



TRIFACTA

User Guide

Version: 8.7.1

Doc Build Date: 09/29/2022

Copyright © Trifacta Inc. 2022 - All Rights Reserved. CONFIDENTIAL

These materials (the “Documentation”) are the confidential and proprietary information of Trifacta Inc. and may not be reproduced, modified, or distributed without the prior written permission of Trifacta Inc.

EXCEPT AS OTHERWISE PROVIDED IN AN EXPRESS WRITTEN AGREEMENT, TRIFACTA INC. PROVIDES THIS DOCUMENTATION AS-IS AND WITHOUT WARRANTY AND TRIFACTA INC. DISCLAIMS ALL EXPRESS AND IMPLIED WARRANTIES TO THE EXTENT PERMITTED, INCLUDING WITHOUT LIMITATION THE IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE AND UNDER NO CIRCUMSTANCES WILL TRIFACTA INC. BE LIABLE FOR ANY AMOUNT GREATER THAN ONE HUNDRED DOLLARS (\$100) BASED ON ANY USE OF THE DOCUMENTATION.

For third-party license information, please select **About Trifacta** from the Help menu.

1. Workflow Basics	8
1.1 Object Overview	10
1.2 Import Basics	16
1.3 Profiling Basics	17
1.4 Transform Basics	21
1.5 Sampling Basics	32
1.6 Running Job Basics	36
1.7 Export Basics	38
2. Common Tasks	39
2.1 Import Tasks	42
2.1.1 Connect to Data	43
2.1.1.1 Share a Connection	46
2.1.2 Import a File	47
2.1.2.1 Change File Encoding	49
2.1.2.2 Remove Initial Structure	50
2.1.3 Import a Table	51
2.1.3.1 Disable Type Inference	52
2.1.4 Import from Another Flow	53
2.1.5 Import Excel Data	55
2.1.6 Import Google Sheets Data	58
2.1.7 Import PDF Data	62
2.1.8 Create Dataset with Parameters	66
2.1.8.1 Parameterize Files for Import	69
2.1.8.2 Parameterize Tables for Import	76
2.1.9 Create Dataset with SQL	82
2.2 Discovery Tasks	90
2.2.1 Explore Suggestions	91
2.2.2 Add or Edit Recipe Steps	95
2.2.3 Filter Data	96
2.2.4 Locate Outliers	98
2.2.5 Compute Counts	104
2.2.6 Calculate Metrics across Columns	108
2.2.7 Compare Strings	111
2.2.8 Analyze across Multiple Columns	115
2.2.9 Parse Fixed-Width File and Infer Columns	118
2.2.10 Generate a Sample	120
2.3 Validation Tasks	125
2.3.1 Profile Your Source Data	126
2.3.2 Validate Your Data	128
2.3.3 Find Bad Data	135
2.3.4 Find Missing Data	140
2.3.5 Manage Null Values	146
2.4 Structuring Tasks	148
2.4.1 Initial Parsing Steps	149
2.4.2 Reshaping Steps	153
2.4.3 Split Column	154
2.4.4 Move Columns	160
2.4.5 Delete Data	164
2.4.6 Select	167
2.4.7 Create Aggregations	169
2.4.8 Pivot Data	172
2.4.9 Unpivot Columns	180
2.4.10 Window Transformations	183
2.4.11 Working with Arrays	194
2.4.12 Working with JSON v2	203
2.4.13 Working with JSON v1	212
2.5 Cleanse Tasks	222
2.5.1 Rename Columns	223
2.5.2 Sanitize Column Names	232
2.5.3 Change Column Data Type	233

2.5.4	Copy and Paste Columns	236
2.5.5	Create Column by Example	237
2.5.6	Remove Data	239
2.5.7	Deduplicate Data	244
2.5.8	Compare Values	247
2.5.9	Replace Cell Values	249
2.5.10	Replace Values Using Patterns	251
2.5.11	Replace Groups of Values	258
2.5.12	Normalize Numeric Values	263
2.5.13	Standardize Using Patterns	269
2.5.14	Modify String Values	273
2.5.15	Manage String Lengths	284
2.5.16	Extract Values	287
2.5.17	Format Dates	296
2.5.18	Apply Conditional Transformations	303
2.5.19	Prepare Data for Machine Processing	306
2.6	Enrichment Tasks	311
2.6.1	Create New Column	312
2.6.2	Add Two Columns	314
2.6.3	Generate Primary Keys	318
2.6.4	Add Lookup Data	322
2.6.5	Append Datasets	325
2.6.6	Join Data	327
2.6.6.1	Configure Range Join	332
2.6.7	Insert Metadata	334
2.6.8	Invoke External Function	337
2.7	Publishing Tasks	339
2.7.1	Create Outputs	340
2.7.1.1	Create Output SQL Scripts	345
2.7.2	Publish Results on Demand	351
2.7.3	Reuse Recipe	352
2.8	Project Management Tasks	354
2.8.1	Take a Snapshot	355
2.8.2	Track Data Changes	357
2.8.3	Add Comments to Your Recipe	361
2.8.4	Create Custom Data Types	362
2.8.5	Create Target	366
2.8.6	Optimize Job Processing	368
2.8.7	Diagnose Failed Jobs	370
2.8.8	Schedule a Job	376
2.8.9	Create Branching Outputs	378
2.8.10	Build Sequence of Datasets	380
2.8.11	Fix Dependency Issues	383
2.8.12	Share a Flow	385
2.8.13	Export Flow	386
2.8.14	Import Flow	388
2.8.14.1	Reconnect Flow to Source Data	391
2.8.14.2	Reconnect Flow to Outputs	392
2.8.15	Create or Replace Macro	393
2.8.16	Apply a Macro	400
2.8.17	Export Macro	402
2.8.18	Import Macro	403
2.8.19	Create Flow Parameter	405
2.8.20	Flag for Review	412
2.8.21	Manage Environment Parameters	415
2.9	User Management Tasks	418
2.9.1	Change Password	419
2.9.2	Configure Your Access to S3	420
3.	Concepts	423

3.1	Feature Overviews	424
3.1.1	Overview of Data Export	425
3.1.2	Overview of Data Import	429
3.1.3	Overview of Storage	433
3.1.4	Overview of Predictive Transformation	438
3.1.5	Overview of the Type System	445
3.1.6	Overview of Standardization	455
3.1.7	Overview of Cluster Clean	460
3.1.8	Overview of Visual Profiling	464
3.1.9	Overview of Sampling	469
3.1.10	Overview of Job Execution	473
3.1.10.1	Trifacta Photon Running Environment	479
3.1.10.2	EMR Running Environment	480
3.1.10.3	AWS Databricks Running Environment	481
3.1.10.4	Azure Databricks Running Environment	482
3.1.10.5	Hadoop Spark Running Environment	483
3.1.11	Overview of TBE	484
3.1.12	Overview of Data Quality	487
3.1.13	Overview of Sharing	
3.1.14	Overview of Job Monitoring	500
3.1.15	Overview of Automator	506
3.1.16	Overview of Parameterization	508
3.1.17	Overview of Authorization	521
3.1.18	Overview of Operationalization	524
3.1.19	Overview of Macros	530
3.1.20	Overview of Deployment Manager	534
3.1.21	Overview of Pattern Matching	543
3.1.22	Overview of RapidTarget	547
3.2	Recipe Development	550
3.2.1	Enriching Data	551
3.3	Using Connections	554
3.3.1	Using Databases	555
3.3.2	Using HDFS	557
3.3.3	Using S3	561
3.3.4	Using SQL DW	565
4.	Reference	567
4.1	UI Reference	568
4.1.1	Home Page	569
4.1.1.1	Download Logs Dialog	574
4.1.2	Flows Page	575
4.1.2.1	Create Flow Page	579
4.1.2.2	Flow View Page	580
4.1.2.2.1	View for Imported Datasets	588
4.1.2.2.2	View for Dataset with Parameters	592
4.1.2.2.3	View for Recipes	595
4.1.2.2.4	View for Reference Datasets	600
4.1.2.2.5	View for Unstructured Datasets	604
4.1.2.2.6	View for Outputs	
4.1.2.2.7	Share Flow Dialog	611
4.1.2.2.8	Change Dataset Dialog	613
4.1.2.2.9	Add Schedule Dialog	615
4.1.2.2.10	Manage Flow Notifications Dialog	617
4.1.2.2.11	Manage Parameters Dialog	619
4.1.2.2.12	Flow Search Panel	624
4.1.2.2.13	Flow Optimization Settings Dialog	626
4.1.3	Plans Page	629
4.1.3.1	Plan View Page	631
4.1.3.1.1	Share Plan Dialog	637
4.1.3.1.2	Manage Plan Notifications Dialog	638

4.1.3.1.3	<i>Plan View for Flow Tasks</i>	640
4.1.3.1.4	<i>Plan View for HTTP Tasks</i>	646
4.1.3.1.5	<i>Plan View for Slack Tasks</i>	649
4.1.4	<i>Library Page</i>	652
4.1.4.1	<i>Dataset Details Page</i>	654
4.1.4.2	<i>Import Data Page</i>	657
4.1.4.2.1	<i>Create Connection Window</i>	663
4.1.4.2.2	<i>Database Browser</i>	668
4.1.4.2.3	<i>File System Browser</i>	671
4.1.4.2.4	<i>HDFS Browser</i>	673
4.1.4.2.5	<i>S3 Browser</i>	675
4.1.4.2.6	<i>File Import Settings</i>	677
4.1.4.2.7	<i>Table Import Settings</i>	679
4.1.4.3	<i>Macros Page</i>	680
4.1.4.3.1	<i>Macro Details Page</i>	682
4.1.5	<i>Connections Page</i>	684
4.1.5.1	<i>Share Connection Dialog</i>	687
4.1.6	<i>Jobs Page</i>	691
4.1.6.1	<i>Job Details Page</i>	696
4.1.6.1.1	<i>Publishing Dialog</i>	707
4.1.6.2	<i>Plan Runs Page</i>	709
4.1.6.3	<i>Sample Jobs Page</i>	712
4.1.7	<i>Transformer Page</i>	714
4.1.7.1	<i>Data Grid Panel</i>	718
4.1.7.1.1	<i>Column Histograms</i>	724
4.1.7.1.2	<i>Data Quality Bars</i>	726
4.1.7.1.3	<i>Lookup Wizard</i>	727
4.1.7.1.4	<i>Standardize Page</i>	730
4.1.7.1.5	<i>Custom Type Dialog</i>	733
4.1.7.2	<i>Recipe Navigator</i>	736
4.1.7.3	<i>Transformer Toolbar</i>	737
4.1.7.4	<i>Column Menus</i>	741
4.1.7.4.1	<i>Choose Datetime Format Dialog</i>	745
4.1.7.4.2	<i>Transformation by Example Page</i>	746
4.1.7.5	<i>Column Browser Panel</i>	749
4.1.7.6	<i>Column Details Panel</i>	753
4.1.7.7	<i>Transform Preview</i>	756
4.1.7.8	<i>Context Panel</i>	759
4.1.7.8.1	<i>Recipe Panel</i>	760
4.1.7.8.2	<i>Transform Builder</i>	765
4.1.7.8.3	<i>Search Panel</i>	770
4.1.7.8.4	<i>Edit History Panel</i>	772
4.1.7.8.5	<i>Samples Panel</i>	774
4.1.7.8.6	<i>Selection Details Panel</i>	780
4.1.7.9	<i>Filter Panel</i>	785
4.1.7.10	<i>Visible Columns Panel</i>	787
4.1.7.11	<i>Join Window</i>	788
4.1.7.12	<i>Union Page</i>	793
4.1.7.13	<i>Run Job Page</i>	797
4.1.7.13.1	<i>SQL Scripts Panel</i>	803
4.1.7.13.2	<i>Redshift Table Settings</i>	805
4.1.7.13.3	<i>Hive Table Settings</i>	806
4.1.7.13.4	<i>Databricks Tables Table Settings</i>	807
4.1.7.13.5	<i>Tableau Server Datasource Settings</i>	809
4.1.7.13.6	<i>Spark Execution Properties Settings</i>	810
4.1.7.13.7	<i>Microsoft SQL Data Warehouse Table Settings</i>	812
4.1.8	<i>Preferences Page</i>	
4.1.8.1	<i>User Profile Page</i>	816
4.1.8.1.1	<i>Storage Config Page</i>	818

4.1.8.2 Account Settings Page	820
4.1.8.3 Email Notifications Page	822
4.1.8.4 Access Tokens Page	824
4.1.8.5 Databricks Settings Page	827
4.1.9 Schedules Page	829
4.1.10 UI Index	831
4.2 Supported File Formats	837
4.3 Column Statistics Reference	843
4.4 Supported Data Types	845
4.4.1 String Data Type	847
4.4.2 Integer Data Type	848
4.4.3 Decimal Data Type	849
4.4.4 Boolean Data Type	850
4.4.5 Social Security Number Data Type	851
4.4.6 Phone Number Data Type	852
4.4.7 Email Address Data Type	853
4.4.8 Credit Card Data Type	854
4.4.9 Gender Data Type	855
4.4.10 Zip Code Data Type	856
4.4.11 State Data Type	857
4.4.12 Object Data Type	858
4.4.13 Array Data Type	859
4.4.14 IP Address Data Type	860
4.4.15 URL Data Type	861
4.4.16 HTTP Code Data Type	862
4.4.17 Datetime Data Type	863
4.5 Supported Numeric Formatting	869
4.6 Type Conversions	872
4.6.1 Avro Data Type Conversions	874
4.6.2 DB2 Data Type Conversions	875
4.6.3 Hive Data Type Conversions	876
4.6.4 Oracle Data Type Conversions	879
4.6.5 MySQL Data Type Conversions	882
4.6.6 Parquet Data Type Conversions	884
4.6.7 Postgres Data Type Conversions	886
4.6.8 Redshift Data Type Conversions	889
4.6.9 Snowflake Data Type Conversions	891
4.6.10 AWS Glue Data Type Conversions	893
4.6.11 Salesforce Data Type Conversions	894
4.6.12 SQL Server Data Type Conversions	895
4.6.13 SQL DW Data Type Conversions	898
4.6.14 Databricks Tables Data Type Conversions	901
4.6.15 Tableau Hyper Data Type Conversions	904
4.6.16 Teradata Data Type Conversions	905
4.6.17 SharePoint Data Type Conversions	908
4.7 Transformation Reference	910
4.8 Plan Metadata References	916
4.9 cron Schedule Syntax Reference	920
4.10 Supported Time Zone Values	926
4.11 Locale Settings	942
4.12 Supported File Encoding Types	944
4.13 Sort Order	945
4.14 Join Types	947
4.15 Join Metrics	951
4.16 Supported SQL Syntax	952
4.17 Sample Types	954
4.18 Support Bundle Contents	957

Workflow Basics

Learn the basics of how to import, wrangle, execute jobs, profile, and export your data from Designer Cloud powered by Trifacta® Enterprise Edition.

Overview

Designer Cloud powered by Trifacta® Enterprise Edition enables analysts, data specialists, and other domain experts to quickly cleanse and transform datasets of varying sizes for use throughout the enterprise. Using an innovative set of web-based tools, you can import complex datasets and wrangle them for use in virtually any target system. Key capabilities include:

- **Import** from flat file, databases, or distributed storage systems
- Locate and remove or modify **missing or mismatched** data
- **Unnest complex** data structures
- Identify statistical **outliers** in your data for review and management
- Perform **lookups** from one dataset into another reference dataset
- Aggregate columnar data using a variety of **aggregation functions**
- **Normalize** column values for more consistent usage and statistical modeling
- Merge datasets with **joins**
- Append one dataset to another through **union** operations

Most of these operations can be executed with a few mouse clicks. This section provides a basic overview of common workflows through Designer Cloud powered by Trifacta Enterprise Edition.

Prerequisites

Before you begin, please verify the following:

- **Trifacta account:** You have a Trifacta account and can login.
- If you are importing data from a location other than your local file system, you must have the appropriate role in your user account. See *Import Basics*.
- **Example data:** You should use a sample set of data during this workflow.

Basic Workflow

1. **Import data:** Integrate data from a variety of sources of data.

Tip: When you login for the first time, you can immediately upload a dataset to begin transforming it.

See *Import Basics*.

2. **Profile your data:** Before, during, and after you transform your data, you can use the visual profiling tools to quickly analyze and make decisions about your data. See *Profiling Basics*.
3. **Build transform recipes:** Use the various views in the Transformer Page to build your transform recipes and preview the results on sampled data. See *Transform Basics*.
4. **Sample your data:** In Designer Cloud powered by Trifacta Enterprise Edition, you create your recipes while working with a sample of your overall dataset. As needed, you can take new samples, which can provide new perspectives and enhance performance in complex flows. See *Sampling Basics*.
5. **Run job:** Launch a job to run your recipe on the full dataset. Review results and iterate as needed. See *Running Job Basics*.
6. **Export results:** Export the generated results data for use outside of Designer Cloud powered by Trifacta Enterprise Edition. See *Export Basics*.

Object overview: You should review the overview of the objects that are created and maintained in Designer Cloud powered by Trifacta Enterprise Edition. See *Object Overview*.

Object Overview

Contents:

- *Flow Structure and Objects*
 - *Flow*
 - *Imported Dataset*
 - *Recipe*
 - *Flow Example*
 - *Working with recipes*
 - *Connections*
 - *Flow Schedules*
 - *Plans*
-

Explore the objects that you create and their relationships. Flows, imported datasets, and recipes are created to transform your sampled data. After you build your output objects, you can run a job to transform the entire dataset based on your recipe and deliver the results according to your output definitions.

Flow Structure and Objects

Within Designer Cloud powered by Trifacta® Enterprise Edition, the basic unit for organizing your work is the flow. The following diagram illustrates the component objects of a flow and how they are related:

Flow Objects

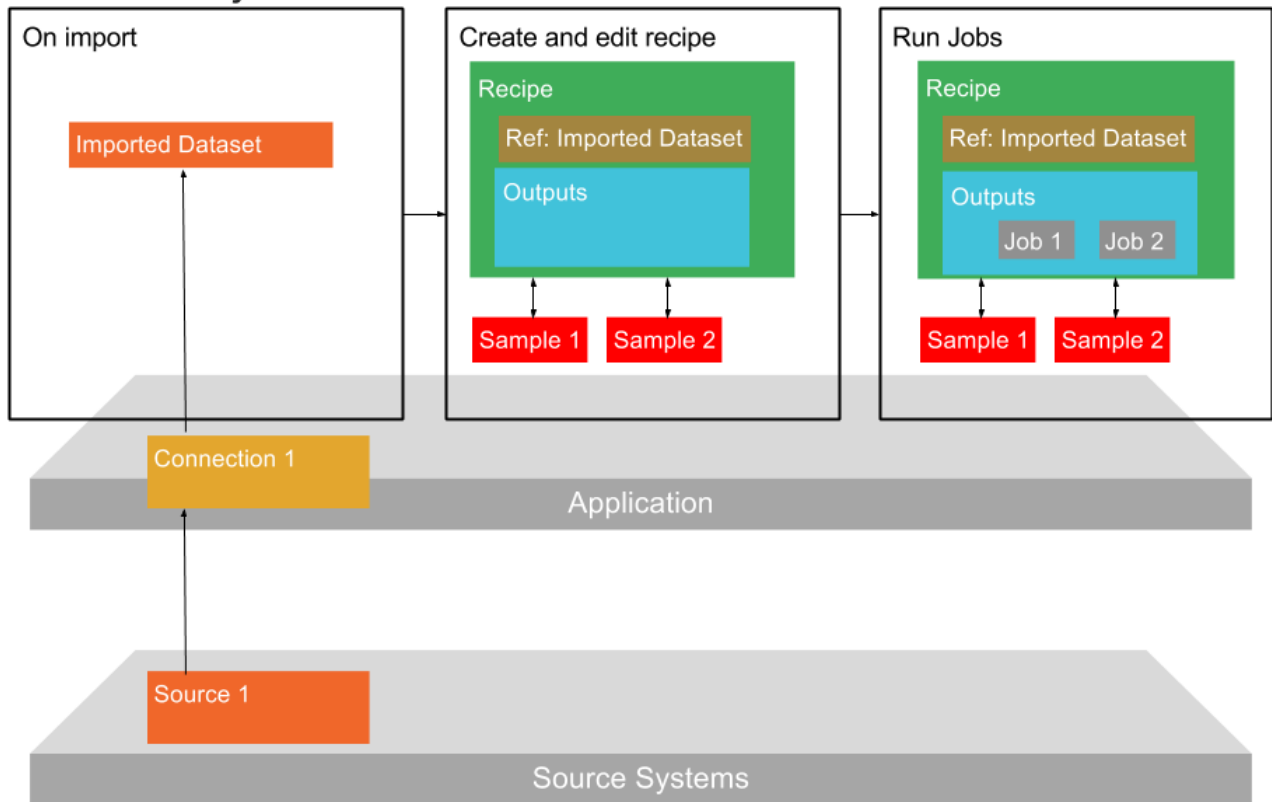


Figure: Objects in a Flow

Flow

A **flow** is a container for holding one or more datasets, associated recipes and other objects. This container is a means for packaging Trifacta objects for the following types of actions:

- Creating relationships between datasets, their recipes, and other datasets.
- Sharing with other users
- Copying
- Execution of pre-configured jobs
- Creating references between recipes and external flows
- Exporting and importing into different instances of the Designer Cloud powered by Trifacta platform

Imported Dataset

Data that is imported to the platform is referenced as an imported dataset. An **imported dataset** is simply a reference to the original data; the data does not exist within the platform. An imported dataset can be a reference to a file, multiple files, database table, or other type of data.

NOTE: An imported dataset is a pointer to a source of data. It cannot be modified or stored within Designer Cloud powered by Trifacta Enterprise Edition.

- An imported dataset can be referenced in recipes.
- Imported datasets are created through the *Import Data Page*.
- For more information on the process, see *Import Basics*.

After you have created an imported dataset, it becomes usable after it has been added to a flow. You can do this as part of the import process or later.

Recipe

A **recipe** is a user-defined sequence of steps that can be applied to transform a dataset.

- A recipe object is created from an imported dataset or another recipe. You can create a recipe from a recipe to chain together recipes.
- Recipes are interpreted by Designer Cloud powered by Trifacta Enterprise Edition and turned into commands that can be executed against data.
- When initially created, a recipe contains no steps. Recipes are augmented and modified using the various visual tools in the *Transformer Page*.
- For more information on the process, see *Transform Basics*.

In a flow, the following objects are associated with each recipe, which are described below:

- Outputs
- References

Outputs and Publishing Destinations

Outputs contain one or more publishing destinations, which define the output format, location, and other publishing options that are applied to the results generated from a job run on the recipe.

When you select a recipe's output object in a flow, you can:

- Define the publishing destinations for outputs that are generated when the recipe is executed. **Publishing destinations** specify output format, location, and other publishing actions. A single recipe can have multiple publishing destinations.
- Run an on-demand job using the specified destinations. The job is immediately queued for execution.

References and Reference Datasets

References allow you to create a reference to the output of the recipe's steps in another dataset. References are not depicted in the above diagram.

When you select a recipe's reference object, you can add it to another flow. This object is then added as a reference dataset in the target flow. A **reference dataset** is a read-only version of the output data generated from the execution of a recipe's steps.

Flow Example

The following diagram illustrates the flexibility of object relationships within a flow.

Flow Example

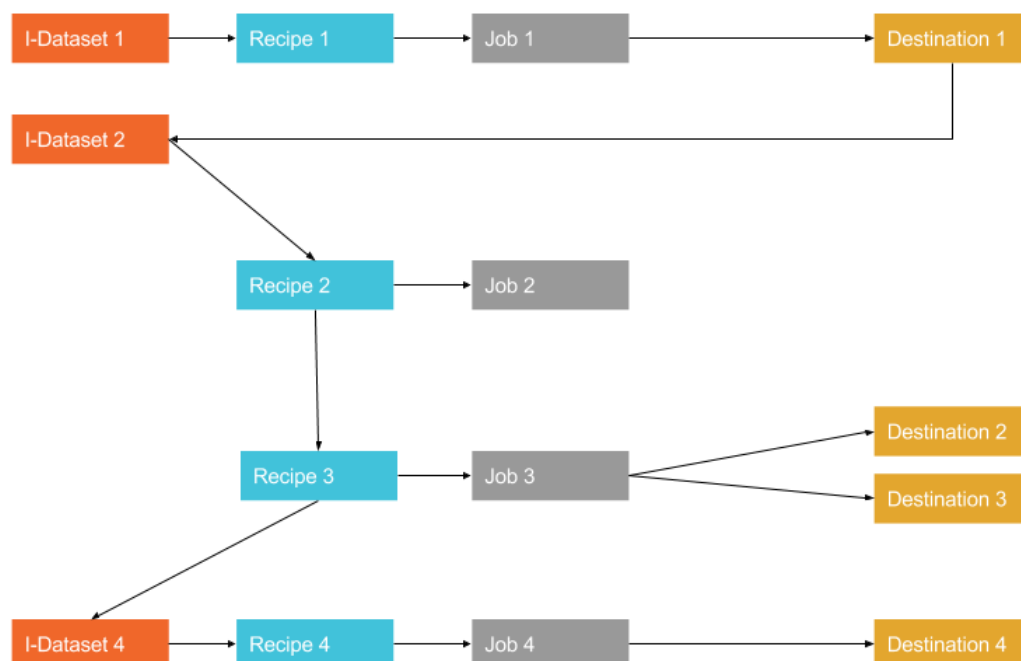


Figure: Flow Example

Type	Datasets	Description
Standard job execution	Recipe 1 /Job 1	Results of the job are used to create a new imported dataset (I-Dataset 2). See <i>Job Details Page</i> .
Create dataset from generated results	Recipe 2 /Job 2	Recipe 2 is created off of I-Dataset 2 and then modified. A job has been specified for it, but the results of the job are unused.
Chaining datasets	Recipe 3 /Job 3	Recipe 3 is chained off of Recipe 2. The results of running jobs off of Recipe 2 include all of the upstream changes as specified in I-Dataset 1/Recipe1 and I-Dataset 2/Recipe 2.
Reference dataset	Recipe 4 /Job 4	I-Dataset 4 is created as a reference off of Recipe 3. It can have its own recipe, job, destinations, and results.

Flows are created in the Flows page. See *Flows Page*.

Working with recipes

Recipes are edited in the Transformer page, which provides multiple methods for quickly selecting and building recipe steps.

Samples: Within the Transformer page, you build the steps of your recipe against a **sample** of the dataset.

- A sample is typically a subset of the entire dataset. For smaller datasets, the sample may be the entire dataset.

- As you build or modify your recipe, the results of each modification are immediately reflected in the sampled data. So, you can rapidly iterate on the steps of your recipe within the same interface.
- As needed, you can generate additional samples, which may offer different perspectives on the data.
- See *Transform Basics*.

Flow parameters: You can specify flow parameters that can be referenced in your recipes. When invoked in a step, a flow parameter replaces its reference with the default string value associated or any override value that you have specified for it. See *Overview of Parameterization*. **Macros:** As needed, you can create reusable sequences of steps that can be parameterized for use in other recipes. For more information, see *Overview of Macros*.

Run Jobs: When you are satisfied with the recipe that you have created in the Transformer page, you can execute a **job**. A job may be composed of one or more of the following job types:

- **Transform job:** Executes the set of recipe steps that you have defined against your sample(s), generating the transformed set of results across the entire dataset.
- **Profile job:** Optionally, you can choose to generate a visual profile of the results of your transform job. This visual profile can provide important feedback on data quality and can be a key for further refinement of your recipe.
- When a job completes, you can review the resulting data and identify data that still needs fixing. See *Job Details Page*.
- For more information on the process, see *Running Job Basics*.

Connections

A **connection** is a configuration object that provides a personal or global integration to an external datastore. Reading data from remote sources and writing results are managed through connections.

- Connections are not associated with individual datasets or flows.
 - Connections are not reflected in the above diagram.
- Most connections can be created by individual users and shared as-needed.
- Depending on the datastore, connections can be read-only, write-only, or both.
- For more information, see *Connections Page*.

Flow Schedules

You can associate a schedule with a flow. A **schedule** is a combination of one or more triggers and the outputs that are generated from them.

NOTE: A flow can have only one schedule associated with it.

- A **trigger** is a scheduled time of execution. When a trigger's time occurs, all of the scheduled output destinations are queued for generation.
 - A schedule can have multiple triggers associated with it. Therefore, a flow can be scheduled for execution at multiple intervals.
- A **scheduled destination** is an output associated with a recipe. This output is generated only when the schedule for the flow is triggered.
 - A scheduled destination is not tied to a specific trigger. When a trigger occurs, all scheduled destinations in the flow are generated.
 - A scheduled destination generates one or more publishing actions (outputs) from the recipe when triggered.
 - A recipe can have only one scheduled destination.
 - Each recipe in a flow can have a scheduled destination.
 - If a flow has a trigger but no scheduled destination, nothing is generated at trigger time.

Below, you can see the object hierarchy within a schedule.

```
+ schedule for Flow 1
+ trigger 1
+ trigger 2
+ scheduled destination a
+ scheduled destination b
+ schedule for Flow 2
+ trigger 3
+ scheduled destination c
+ scheduled destination d
```

For more information, see *Overview of Automator*.

Plans

A **plan** is a sequence of triggers and tasks that can be executed across multiple flows. A plan is executed on a snapshot of all objects at the time that the plan is triggered.

- A **task** is an executable action that is taken as part of a plan's sequence. For example, task #1 could be to execute a flow that imports all of your source data. Task #2 executes the flow that cleans and combines that data. Example task types:
 - A **flow task** is the execution of the recipes in a specified flow, which result in the generation of one or more selected outputs.
 - A **trigger** for a plan is the schedule for its execution.
 - A **snapshot** is a frozen image of the plan. This snapshot of the plan defines the objects that are executed as part of a plan run.
 - An **HTTP task** is a request submitted by the product to a third-party server as part of the sequence of tasks in a plan. For example, an HTTP task could be the submission of a message to a channel in your enterprise's messaging system.

For more information, see *Overview of Operationalization*.

Import Basics

Designer Cloud powered by Trifacta® Enterprise Edition can import data from a variety of flat file formats and other distributed sources.

NOTE: Designer Cloud powered by Trifacta Enterprise Edition does not modify a source. Instead, a set of metadata is associated with the source data, which enables transformation of the source. On export, a new version of the data is written to one or more specified output destinations.

For more information on the formats supported for input, see *Supported File Formats*. For more information on distributed sources of datasets, see *Connection Types*.

Steps:

When data is imported, a reference to it is stored by the platform as an imported dataset. The source data is not modified. In the application, you modify the recipe associated with a dataset to transform the imported data.

NOTE: Any user with a valid user account can import data from a local file.

1. Login to the application.

Tip: When you login for the first time, you can immediately import a dataset to begin transforming it.

2. In the menubar, click **Library**. Click **Import Data**.

3. To add a dataset:

- a. Select the connection where your source is located.
- b. Upload:
 - i. Select **Upload** to upload a file from your local desktop. You can select multiple files to upload. For this example, select only one file.
 - ii. Navigate and select the file or files for your source. Click **Open**.
- c. Backend storage, such as S3:
 - i. Navigate and select the file or files for your source.
 - ii. To queue the dataset for uploading, click the Plus icon next to its name.
 - iii. You can select multiple files.
- d. Select the Add to new flow checkbox. This option creates a new flow, which is a container object for your Trifacta assets. Your imported dataset is added to it.

4. To begin working with your dataset, click **Continue**.

5. The imported dataset and its containing flow are created.

6. You can begin working with the dataset in the Transformer page. For more information, see *Transform Basics*.

Tip: If you are interested, you can create a visual profile of your source data before you begin transforming. For more information, see *Profiling Basics*.

For more information on details about importing, see *Import Tasks*.

Example Dataset

You can download an example dataset from the following URL:

https://trifacta-public-datasets.s3.amazonaws.com/demo-datasets/customers_sample_data.csv

Profiling Basics

Contents:

- *Profile Source Data*
 - *Profiling in the Application*
 - *Status Bar*
 - *Column Header*
 - *Column Histogram*
 - *Column Details - statistics and outliers*
 - *Column Browser - profiles across columns*
 - *Profile Jobs*
 - *Download visual profile*
-

Designer Cloud powered by Trifacta® Enterprise Edition surfaces visual representations of your data for individual columns and the entire dataset. These visual profiles enable you to make quick assessments of problems, unusual patterns, and required changes to your data.

Tip: Visual profiling is especially important in recipe development. When you identify something of interest, you can select the visual representation of it, and the platform prompts you with a set of suggested transforms to add to your recipe. Examples are below.

For more background information, see *Overview of Visual Profiling*.

Profile Source Data

Tip: When you first load your dataset into the application, you might want to run a job to profile your dataset before you build your recipe. The generated results and profile are accessible through the application, which can be useful for seeing how your dataset has changed during development. For more information, see *Profile Your Source Data*.

Profiling in the Application

In the Designer Cloud application, there are a number of features that provide visual information on the status of individual columns, their data, and the overall dataset.

NOTE: Before your job is run, profiling information such as column statistics are exact counts of the sample that is currently loaded. After the job is run, profiled results in the Job Results page might include estimates for some metrics and counts, depending on the scale of the dataset.

Status Bar

Counts on the rows, columns, and data types in the current sample are displayed at the bottom of the page in the status bar.

Tip: Sample counts are used for profiling when in the Transformer page. When a visual profile is generated as part of your job, the counts are taken from the entire dataset.

9 Columns 47 Rows 5 Data Types

Figure: Status Bar

Column Header

The top of each column contains a data quality bar, which identifies the valid, mismatched, and missing values in the column when compared against the specified data type, and column histogram, which identifies the range of values in the column.

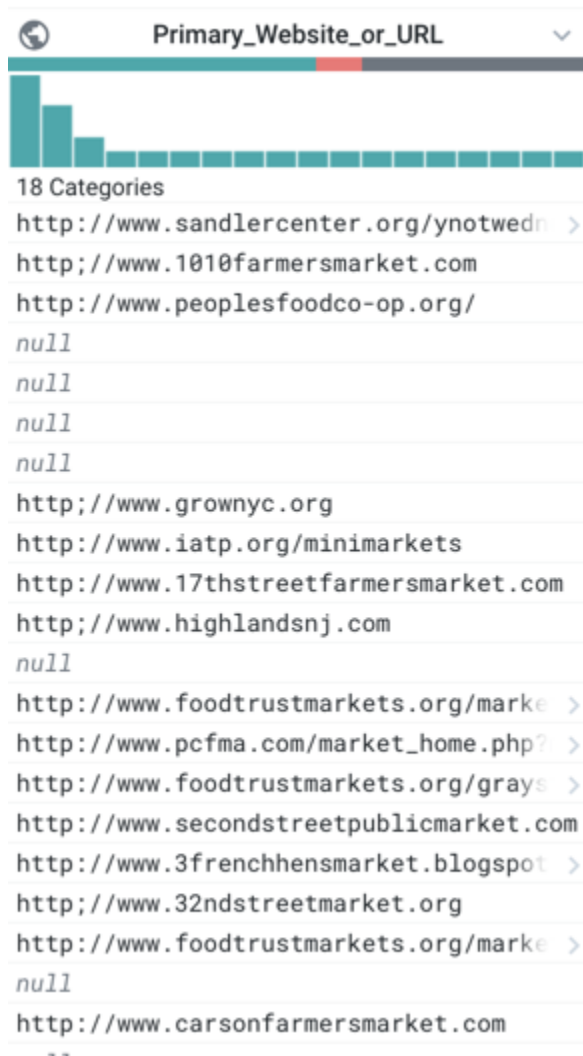


Figure: Example Column

Data Quality Bar - missing and mismatches values

Below the name of the column, the multi-colored band indicates the valid (green), mismatched (red), and missing (gray) values in the column, when matched against the column's data type. In the above image, the data type is set to URL.

Tip: Click the missing or mismatched values in a column's data quality bar. You are prompted with suggestions of transformations to fix or remove these values.

Column Histogram

Each column includes a histogram of the values in the column. In the above image, there are 402 different values in the column, and you can see how some values appear more frequently than others.

Tips:

- In the column histogram, you can select a column value and drag to select a range of values for suggestions on transformations.
- Null values are a special case of missing values. You can use the `ISNULL` function to identify null values in a column, which appear among the category of missing values. See *Manage Null Values*.
- When you select one or more values in the column histogram, you can see the corresponding values for the row values in the histograms for other columns.

See *Column Histograms*.

Column Details - statistics and outliers

In the Column Details window, you can review key statistical information on the values in a column. Displayed statistics are based on the column's data type.

To explore the details for a column's data, select **Column Details** from the drop-down for the specific column in the data grid.

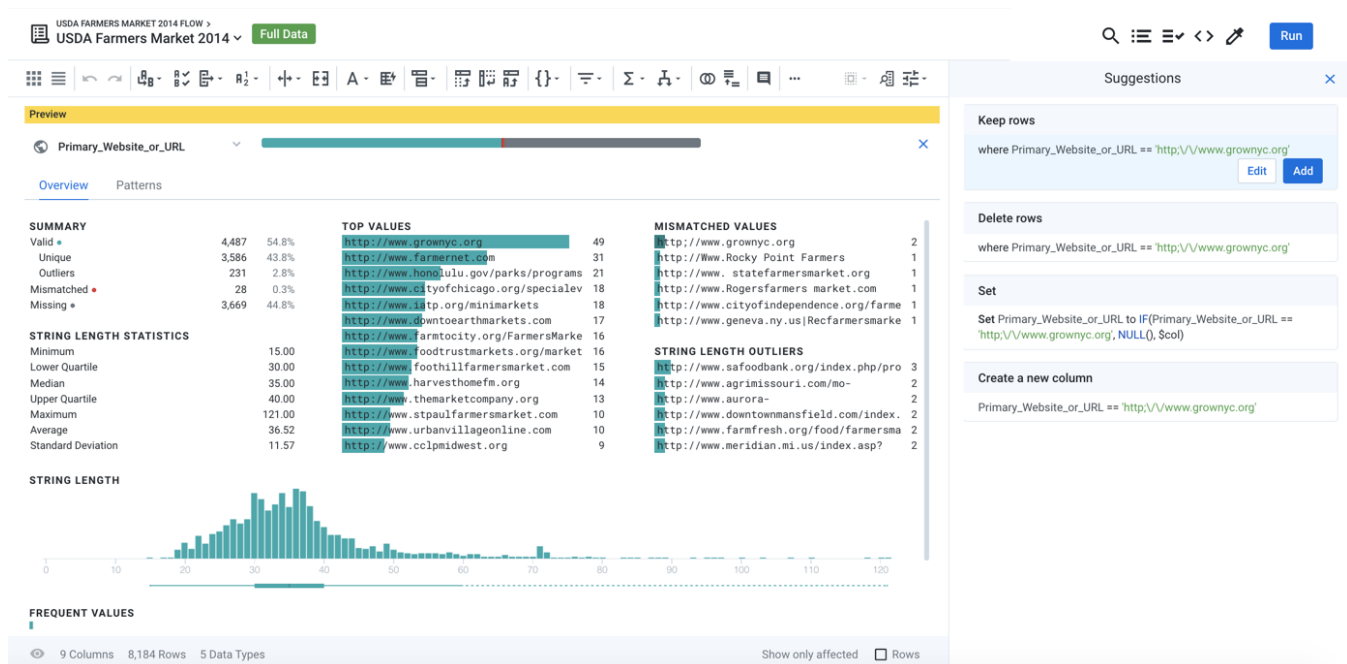


Figure: Column Details

For the selected column, you can review key statistics depending on the data type. The above image shows statistics that apply to the URL data type, which is a variation on String type.

Tips:

- Make a selection from the lists of top, mismatched, and other value lists to be prompted for a set of suggestions for how to modify the selected rows.
- Transform suggestions are updated based on your selection.
- Click the missing values in the data quality bar to prompt for suggestions to address those values in the column.

See *Column Details Panel*.

Column Browser - profiles across columns

In the column browser, you can view visual histograms for each column in the dataset and make selections to identify correlations between values in multiple columns. To open the column browser, click the Columns icon in the Transformer bar.

For more information, see *Column Browser Panel*.

Profile Jobs

When you execute your job, you can generate a visual profile of the entire dataset as part of the job. You can use the generated profile to simplify iteration on your recipe. The optional profiling of the results can take extra time to generate.

Steps:

1. In the Transformer page, click **Run**.
2. Click the Profile Results checkbox.
3. Run the job.
4. When the job finishes, click the Job Id link. Then, click the Profile tab in the Job Details page.

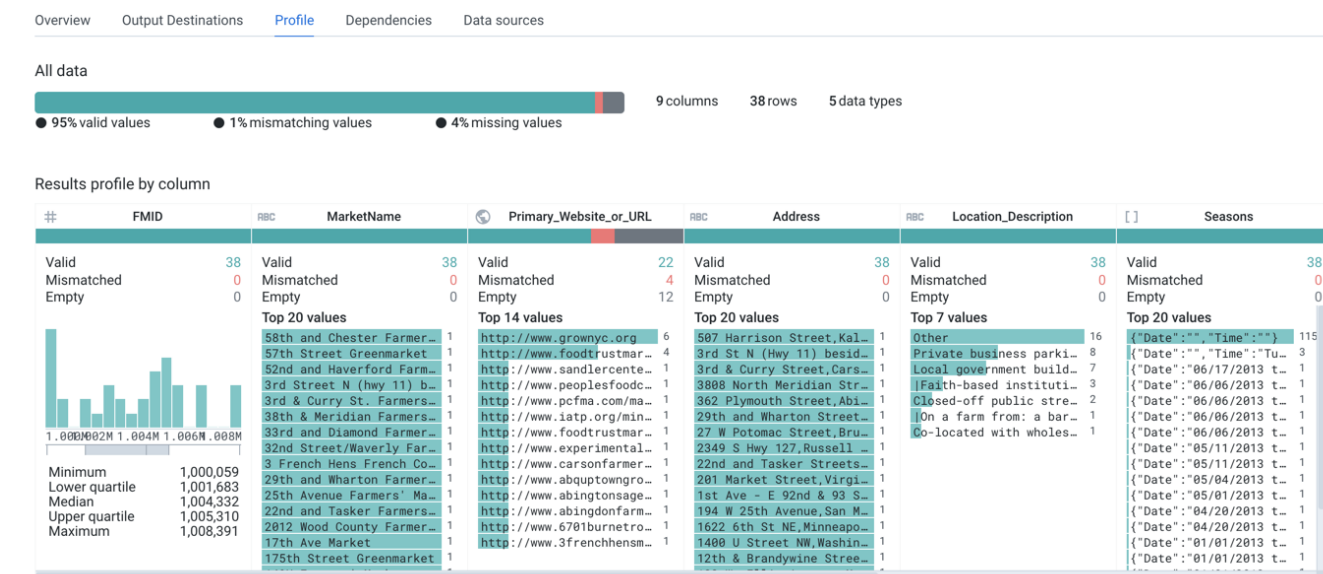


Figure: Visual Profile

This visual profile displays statistics across the entire dataset. Since the data volume of the entire dataset can be quite large, these stats may be approximations.

Download visual profile

From the Profiles tab, you can download your job's visual profile to your desktop.

See *Job Details Page*.

Transform Basics

Contents:

- *Goal*
 - *Recommended Methods for Building Recipes*
 - *Sample*
 - *Cleanse*
 - *Modify*
 - *Enrichment*
 - *Sampling*
 - *Profile*
-

When you edit your dataset's recipe, the Transformer page is opened, where you begin your wrangling tasks on a sample of the dataset. Through this interface, you build your transformation recipe and see the results in real-time as applied to the sample. When you are satisfied with what you see, you can execute a job against the entire dataset.

Goal

Your data transformation is complete when you have done the following:

- Cleansed your data of invalid, missing, or inaccurate values
- Enhanced your dataset as needed with data from other datasets
- Modified your dataset to constrain its values to meet the target schema
- Executed job against the entire dataset
- Exported the results from your dataset and recipe for use in downstream systems

Tip: Before you begin transforming, you should know the target schema that your transformed data must match. A **schema** is the set of columns and their data types, which define the constraints of your dataset.

Tip: If you want to match up against the target schema, you can import a dataset to serve as the target schema to which you are mapping. For more information on this advanced feature, see *Overview of RapidTarget*.

Recommended Methods for Building Recipes

Designer Cloud powered by Trifacta® Enterprise Edition supports the following methods for building recipes. These methods are listed in order of ease of use:

1. **Select something.** When you select elements of data in the Transformer page, you are prompted with a set of suggestions for steps that you can take on the selection or patterns matching the selection. You can select columns or one or more values within columns.

Tip: The easiest method for building recipes is to select items in the application. Over time, the application learns from your selections and prompts you with suggestions based on your previous use. For more information, see *Overview of Predictive Transformation*.

2. **Toolbar and column menus:** In the Transformer page, you can access pre-configured transformations through the Transformer toolbar or through the column context menus.

Tip: Use the toolbar for global transformations across your dataset and the column menu for transformations on one or more selected columns.

- a. When a toolbar item is selected, the Transform Builder is pre-populated with settings and values to get you started. As needed you can modify the step to meet your needs. For more information, see *Transformer Toolbar*.
 - b. The column menu contains the most common transformations for individual or multiple columns. Often, no additional configuration is required. For more information, see *Column Menus*.
 - c. Select multiple columns. Continue selecting columns to be prompted with a different set of suggestions applicable to all of them.
3. **Search and browse for transformations.** Using the Search panel and the Transform Builder, you can rapidly assemble recipe steps through a simple, menu-driven interface. When you choose to add a step, you search for your preferred transformation in the Search panel. When one is selected, the transformation is pre-populated in the Transform Builder for you. See *Search Panel*.

Tip: Use the Transform Builder for performing modifications to the transformation you selected from the Search panel or a suggestion card. See *Transform Builder*.

Sample

Loading very large datasets in Designer Cloud powered by Trifacta Enterprise Edition can overload your browser or otherwise impact performance, so the application is designed to work on a sample of data. After you have finished your recipe working on a sample, you execute the recipe across the entire dataset.

The default sample is the first set of rows of source data in the dataset, the number of which is determined by the platform. For smaller datasets, the entire dataset can be used as your sample. In the Transformer page, it's listed as **Full Data** in the upper-left corner.

In some cases, the default sample might be inadequate or of the wrong type. To generate a new sample, click the name of the sample in the upper-left corner.

NOTE: Collecting new samples requires system resources and storage. In some environments, collecting samples incurs monetary cost.

Tip: You should consider collecting a new sample if you have included a step to change the number of rows in your dataset or have otherwise permanently modified data (keep, delete, lookup, join, or pivot operations). If you subsequently remove the step that made the modification, the generated sample is no longer valid and is removed. This process limits unnecessary growth in data samples.

On the right side of the screen, you can launch a new sampling job on your dataset. For more information, see *Samples Panel*.

Cleanse

Data cleansing tasks address issues in data quality, which can be broadly categorized as follows:

- **Consistency.** Values that describe the same thing should agree with each other. For example, numeric values should have the same precision. String values should be consistently structured to mean the same thing.
- **Validity.** Values should be constrained to the requirements of each field's data type. For example, a DateOfSale field should be a valid date.

- **Reliability.** Values in the same field in different records should mean the same thing. For example, the value 15 in the Temperature field of two different records should not mean Centigrade in one record and Fahrenheit in the other record.

When data is initially imported, it can contain multiple columns, rows, or specific values that you don't need for your final output. Specifically, this phase can involve the following basic activities:

- Remove unused columns
- Address missing and mismatched data
- Change data types
- Improve consistency, validity, and reliability of the data

NOTE: An imported dataset requires about 15 rows to properly infer column data types and the row, if any, to use for column headers.

First recipe steps:

When a dataset sample is first loaded into the Transformer page, Designer Cloud powered by Trifacta Enterprise Edition attempts to split out the unstructured data to form regular, tabular data. If your data appears to contain a header row, it can be used for the titles of the columns.

Figure: Transformer page

In the above image, some initial parsing steps have been applied to structure the data into tabular format, but these steps are not added as formal parts of the recipe. They are hidden from view in the recipe.

- By default, these steps are automatically added to the recipe when you permit the application to detect the structure of the imported data.
- As needed, you can enable visibility of these steps if you need to edit or remove them.
- For more information, see *Initial Parsing Steps*.

The data resulting from these initial transforms is displayed in the **data grid**. See *Data Grid Panel*.

- Your recipe is displayed in the Recipe panel on the right side. You might have to open this panel to see it. See *Recipe Panel*.
- When you select items in the data grid, suggestion cards are displayed for you to begin building transform steps. See *Selection Details Panel*.

- These suggestions can be modified to build more complex or subtle commands in the Transform Builder. See *Transform Builder*.
- Don't forget to use the Transformer toolbar, which pre-configures the Transform Builder with the configuration required for a useful transformation. See *Transformer Toolbar*.
- You can use the column context menu to apply changes to an individual column. See *Column Menus*.

Use a row to create headers:

In most cases, the names of your columns are inferred from the first row of the data in the dataset. If you need to specify a different row, please complete the following:

1. Click the Search icon in the menu bar.
2. In the Search panel textbox, type: header
3. The transformation is displayed in the Transform Builder. Specify the following properties:

Transformation Name	Rename columns
Parameter: Option	Use row as header
Parameter: Row	1

4. If you need to specify a different row to use, you can specify a specific row number to use in the Row textbox.
5. To add this or any transform in development to your recipe, click **Add**. This button is disabled if the step is invalid.

Generate metadata:

On the left side of the data grid, you might notice a set of black dots. If you hover over one of these, the original row number from the source data is listed. Since the data transformation process can change the number of rows or their order, you might want to retain the original order of the rows.

Tip: Some operations, such as unions and joins, can invalidate source row number information. To capture this data into your dataset, it's best to add this transformation early in your recipe.

To retain the original row numbers in a column called, `rowId`, please complete the following:

Transformation Name	New formula
Parameter: Formula type	Single row formula
Parameter: Formula	<code>\$sourcerownumber</code>
Parameter: New column name	<code>rowId</code>

You can use a similar transformation to generate the full path and filename to file-based sources:

Transformation Name	New formula
Parameter: Formula type	Single row formula
Parameter: Formula	<code>\$filepath</code>
Parameter: New column name	<code>filepath</code>

For more information, see *Insert Metadata*.

Delete unused columns:

Your data might contain columns that are not of use to you, so it's in your interest to remove them to simplify the dataset. To delete a column, click the caret next to the column's title and select **Delete**.

Tip: If you are unsure of whether to delete the column, you can use the same caret menu to hide the column for now. Hidden columns do appear in the output.

Tip: You can also delete ranges of columns, too. See *Remove Data*.

Check column data types:

When a dataset is imported, Designer Cloud powered by Trifacta Enterprise Edition attempts to identify the data type of the column from the first set of rows in the column. At times, however, type inference can be incorrect.

Tip: Before you start performing transformations on your data based on mismatched values, you should check the data type for these columns to ensure that they are correct. For more information, see *Supported Data Types*.

For more information, see *Change Column Data Type*.

Display only columns of interest:

You can choose which columns you want to display in the data grid, which can be useful to narrow your focus to problematic areas.

In the Status bar at the bottom of the screen, click the Eye icon.

For more information, see *Visible Columns Panel*.

Review data quality:

After you have removed unused data, you can examine the quality of data within each column just below the column title.

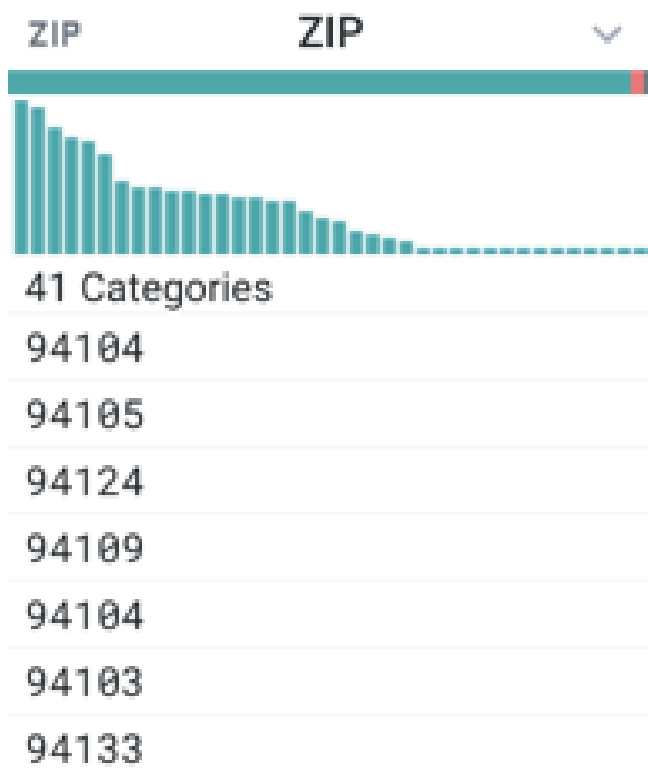


Figure: Column header with data quality bar

The horizontal bar, known, as the **data quality bar**, identifies the quality of the data in the column by the following colors:

Color	Description
green	These values are valid for the column data type.
red	These values do not match those of the column type.
gray	There are no values for the column in these rows.

Tip: When you select values in the data quality bar, those values are highlighted in the sample rows, and suggestions are displayed at the bottom of the screen in the suggestion cards to address the selected rows.

For more information, see *Data Quality Bars*.

Suggestion Cards:

Based on your selections and its knowledge of common data patterns, Designer Cloud powered by Trifacta Enterprise Edition prompts you with suggested transformations. You can then select pre-configured transformations in the right panel of the Transformer page to quickly add steps.

Tip: Where possible, you should try to create your transforms by selecting data and then selecting the appropriate suggestion card. In some cases, you might need to modify the details of the recipe.

In the following example, the missing values in the `SUBSCRIBER_AGE` column have been selected, and a set of suggestion cards is displayed.

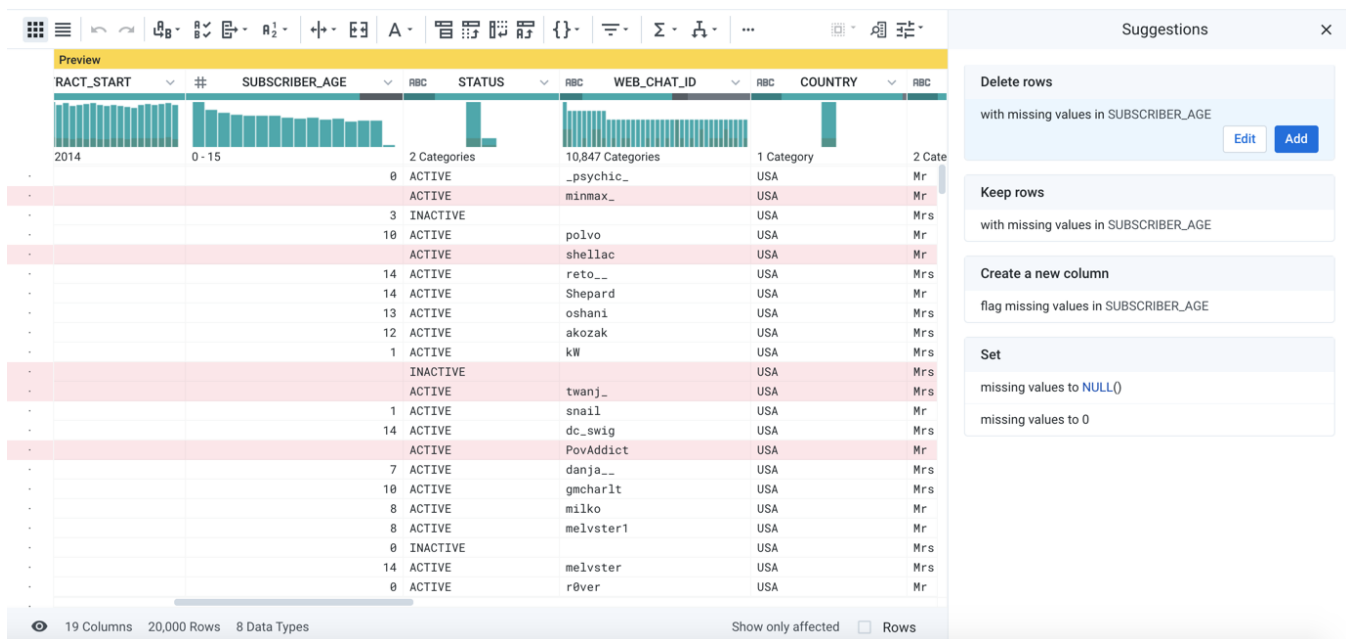


Figure: Selecting missing values

Tip: When previewing a recipe step, you can use the checkboxes in the status bar to display only affected rows, columns, or both, which helps you to assess the effects of your step.

Depending on the nature of the data, you might want to keep, delete, or modify the values. Since the data is missing, the Delete card has been selected.

- To accept this suggest, click **Add**.
- You can modify the step if needed. An example is provided later.

For more information, see *Explore Suggestions*.

For more background information, see *Overview of Predictive Transformation*.

Change data types:

If a column contains a high concentration of mismatched data (red), the column might have been identified as the wrong data type. For example, your dataset includes internal identifiers that are primarily numeric data (e.g. 1000 0022) but have occasional alphabetical characters in some values (e.g. 1000002A). The column for this data might be typed for integer values, when it should be treated as string values. For more information on the available types, see *Supported Data Types*.

Tip: Where possible, you should set the data type for each column to the appropriate type. Designer Cloud powered by Trifacta Enterprise Edition does maintain statistical information and enable some transformation steps based upon data type. See *Column Statistics Reference*.

1. To change a column's data type, click the icon to the left of the column title.
2. Select the new data type.
3. Review the mismatched values for the column to verify that their count has dropped.

For more information, see *Change Column Data Type*.

Explore column details:

As needed, you can explore details about the column's data, including statistical information such as outliers.

- From the caret drop-down next to a column name, select **Column Details**. See *Column Details Panel*.
- For more information on column stats, see *Column Statistics Reference*.

Review histograms:

Just below a column's data quality bar, you can review a histogram of the values found in the column. In the following example, the data histogram on the left applies to the `ZIP` column, while the one on the right applies the `WEB_CHAT_ID` column.

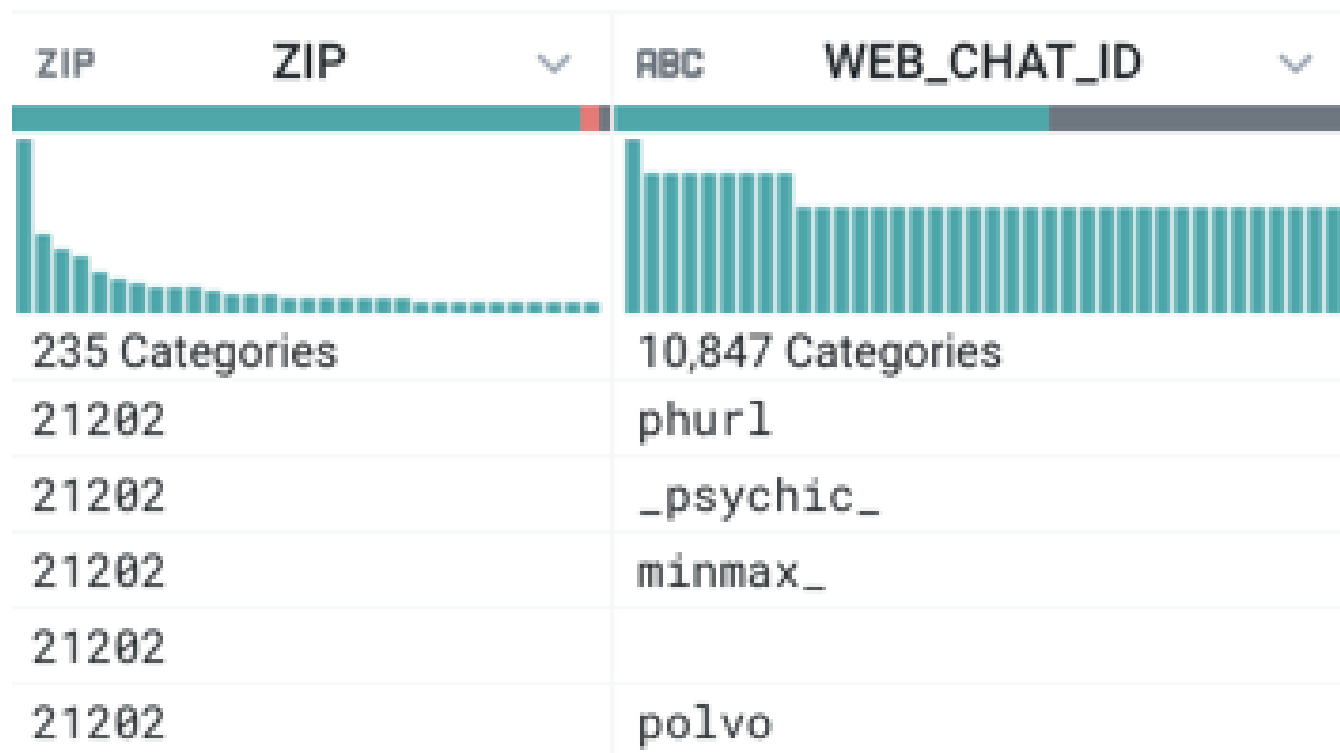


Figure: Column data histogram

When you mouse over the categories in the histogram, you can see the corresponding value, the count of instances in the sample's column, and the percentage of affected rows. In the left one, the bar with the greatest number of instances has been selected; the value `21202` occurs 506 times (21.28%) in the dataset. On the right, the darker shading indicates how rows with `ZIP=21202` map to values in the `WEB_CHAT_ID` column.

Tip: Similar to the data quality bar, you can click values in a data histogram to highlight the affected rows and to trigger a set of suggestions. In this manner, you can use the same data quality tools to apply even more fine-grained changes to individual values in a column.

For a list of common tasks to cleanse your data, see *Cleanse Tasks*.

Modify

After you have performed initial cleansing of your data, you might need to perform modifications to the data to properly format it for the target system, specify the appropriate level of aggregation, or perform some other modification.

Tip: Modification steps are often specific to the downstream use-case for the data. If your source dataset needs to satisfy multiple downstream uses, you might need to make modifications to satisfy each use

case, which are in conflict with each other. It might be easier to cleanse first, create a reference for the recipe object, and then import the reference dataset in each flow for further modification. For more information, see *Flow View Page*.

In the following example, the improperly capitalized word BALTIMORE has been selected, so that you can change it to its propercase spelling (Baltimore). Those rows are highlighted in the row data, and a set of suggestions for how to fix has been provided in the cards at the bottom of the screen. See *Selection Details Panel*.

The screenshot displays a data table with columns: LAST_NAME, SSN, ADDRESS, CITY, and STATE. The 'CITY' column contains several entries, with 'BALTIMORE' highlighted in blue. To the right, a 'Suggestions' panel is open, showing four transformation options: 'Replace', 'Extract values matching', 'Split on values matching', and 'Count values matching'. The 'Replace' suggestion is selected, showing a pattern to replace 'BALTIMORE' with 'Baltimore'.

Figure: Selecting values to modify

Depending on the nature of your data, you might want to keep or change the values, or you can remove the problematic rows altogether.

Tip: When you select one of the suggestion cards, the implied changes are previewed in the Transformer page, so you can see the effects of the change. This previewing capability enables you to review and tweak your changes before they are formally applied. You can always remove a transform step if it is incorrect or even re-run the recipe to generate a corrected set of results, since source data is unchanged. For more information, see *Transform Preview*.

In this case, select the Replace transformation. However, there are a couple of minor issues with the provided suggestion.

- Since the platform has no idea about the meaning of the selection, it might initially suggest removing the text altogether. In this case, you want to change the spelling.
- In the transformation, the Find parameter value contains the pattern used to identify the selection. In this case, it is selecting all values that are capitalized. For now, you only want to fix BALTIMORE.

So, some aspects of this transform must be changed. Click **Edit**.

Transform Builder:

When you modify a transform step, you can make changes in the Transform Builder, which is a simple, menu-driven interface for modifying your transformations:

The screenshot shows the Transform Builder interface. On the left, a data table with columns: LAST_NAME, SSN, RBC, ADDRESS, CITY, CITY, STATE. The table contains 20 rows of data. The 'CITY' column has values like 'Baltimore' and 'BALTIMORE'. The 'STATE' column has values like 'MD'. On the right, the 'Replace text or patterns' panel is open. It shows a 'Find' field with the value 'BALTIMORE' and a 'Replace with' field with the value 'Baltimore'. The 'Advanced options' dropdown is expanded, showing 'Column' and 'required' options. The 'Add' button is highlighted.

Figure: Modifying steps in the Transform Builder

In the Transform Builder, you can replace the pattern with the specific string to locate: BALTIMORE. The new value, which is currently blank, can be populated with the replacement value: Baltimore. Click **Add**.

The step is added to the recipe and automatically applied to the data sample displayed in the Transformer page. For more information, see *Transform Builder*.

See *Cleanse Tasks*.

Enrichment

Before you deliver your data to the target system, you might need to enhance or augment the dataset with new columns or values from other datasets.

Union datasets:

You can append a dataset of identical structure to your currently loaded one to expand the data volume. For example, you can string together daily log data to build weeks of log information. See *Append Datasets*.

Join datasets:

You can also join together two or more datasets based on a common set of values. For example, you are using raw sales data to build a sales commission dataset:

- Your sales transaction dataset contains a column for the salesman's identifier, which indicates the employee who should receive the commission.
- You might want to join your sales transaction dataset to the employee dataset, which provides information on the employee's name and commission rate by the internal identifier.
- If there is no corresponding record in the employee dataset, a commission is not rewarded, and the sales transaction record should not be present in the commission dataset.

This commission dataset is created by performing an inner join between the sales transaction dataset and the employee dataset. In the Search panel, enter *join*. See *Join Data*.

Lookup values:

In some cases, you might need to include or replace values in your dataset with other columns from another dataset. For example, transactional data can reference product and customer by internal identifiers. You can create lookups into your master data set to retrieve user-friendly versions of customer and product IDs.

NOTE: The reference data that you are using for lookups must be loaded as a dataset into Designer Cloud powered by Trifacta Enterprise Edition first.

To perform a lookup for a column of values, click the caret drop-down next to the column title and select **Lookup...**. See *Lookup Wizard*.

For a list of common workflows to enhance your dataset, see *Enrichment Tasks*.

Sampling

The data that you see in the Transformer page is a sample of your entire dataset.

- If your dataset is small enough, the sample is the entire dataset.
- For larger datasets, Designer Cloud powered by Trifacta Enterprise Edition auto-generates an initial sample from the first rows of your dataset.

For larger datasets, you must learn how to generate new samples, which can provide different perspectives on your data and, in complex flows, enhance performance.

Tip: Sampling is an important concept in Designer Cloud powered by Trifacta Enterprise Edition.

For more information, see *Sampling Basics*.

Profile

As part of the transformation process, you can generate and review visual profiles of individual columns and your entire dataset. These interactive profiles can be very helpful in identifying anomalies, outliers, and other issues with your data. For more information, see *Profiling Basics*.

Sampling Basics

Contents:

- *Initial Data*
 - *Take a Sample*
 - *Sampling and Memory*
 - *Sampling Considerations*
 - *Invalid samples*
-

A **sample** is a selection of rows from your dataset, which can be used as the basis for building the transformation steps in your recipe. The Designer Cloud application automatically creates initial data samples of your data whenever you create a new recipe for a dataset and enables you to create additional samples at any time using a variety of sampling techniques.

Initial Data

When you create a new recipe and load it in the Transformer page, the Designer Cloud application displays the initial data sample of the dataset. The **initial data** consists of the first X rows of the datasets, where X is determined by the following factors:

- The number of columns in the dataset
- The amount of data in each cell
- The maximum permitted size of each sample

Take a Sample

These first rows are displayed for you to begin your work in the Transformer page. However, you may begin to run into limitations with this sample. For example, suppose your dataset is organized by date, with earliest dates listed first. There may be significant changes in the data later in the time period that do not appear in the initial sample. You may decide that you need to take a different sample that captures some of these changes.

Steps:

1. In the Transformer page, click the Eyedropper icon at the top of the page.

2. The Samples panel is displayed.

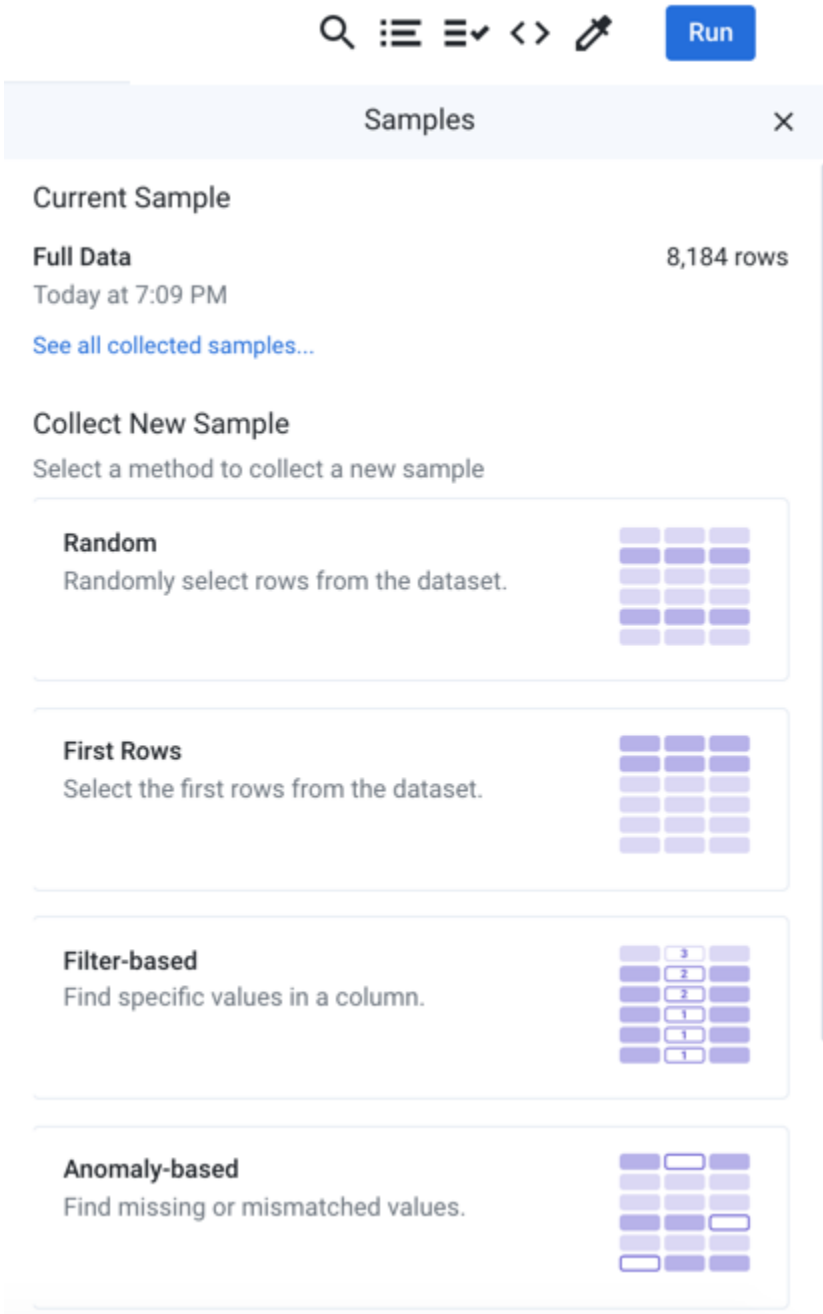


Figure: Samples panel

3. At the top of the panel, you can review the current sample.

Tip: In some cases, the entire dataset is displayed in the data grid. Unless you wish to use a specific sampling technique to filter down your data, sampling may not be useful across the entire dataset.

4. Below the current sample, you can see the available sample types. To take a new random sample:
 - a. Click the Random card.
 - b. Depending on your product edition, you may be able to select Quick Scan or Full Scan.
 - i. Quick Scan creates your sample by making some assumptions about the data when it scans.

- ii. Full Scan creates your sample by scanning across all rows of the dataset. This option can take awhile across a large dataset.
- c. Click **Collect**.
5. The sampling job is queued for execution. When it completes, click **Load Sample**.
6. The data grid is refreshed to display the rows gathered in the new random sample.

For more information, see *Samples Panel*.

Sampling and Memory

NOTE: After you generate a sample, all steps in a recipe that occur after the step selected when you generated the sample are executed in browser memory on the sample data and then displayed in the data grid.

The above statement is best explained by example:

Action	Sampling
1. Create a new recipe and open it in Transformer page.	The initial sample is generated and displayed.
2. Add 3 steps to your recipe.	The 3 new steps are applied to the initial sample in the browser's memory.
3. Generate a new random sample.	The random sample is generated. When you load the sample, it is displayed in the data grid.
4. Add 25 steps to your recipe.	The 25 new steps are applied to the random sample in the browser's memory.
5. Select one of the first 3 steps of your recipe.	The initial sample is loaded and displayed.
6. Insert a new step below the current one.	Now, the first 4 steps are displayed using the initial sample.

Implications:

- As you add steps to your recipe without resampling, your recipe and sample consume more memory in your browser.
- When you perform complex multi-dataset operations, such as joins or unions, your recipe/sample combination consumes a lot more memory.
- If you continue adding steps:
 - Performance in the browser can be impacted. Basic operations such as selection of data or new recipe steps can become slow to respond.
 - The browser can crash.

Sampling Considerations

Tip: When resources permit, it's a good habit to take a new sample after a few multi-dataset operations or operations that otherwise change the number of rows in your dataset have been added to your recipe.

Other considerations:

- **Generating samples takes time.** This is particularly true for Full Scan samples.
- **Sampling can cost money.** In some cloud-based environments, generating a sample costs compute resources, which can add to your computing bill.
- **You may need multiple samples.** For long or complex recipes, you may need to take multiple samples.
- **Reference datasets should begin with a sample.** When you create a recipe for a reference dataset, you should start by generating a new sample for it.

Invalid samples

Samples can become invalid. If your recipe steps change the number of rows or otherwise reshape your dataset using transformations such as pivot or join in the steps leading up to where you took the current sample, your existing sample may no longer be valid.

When the application determines that a sample is invalid:

- The sample can no longer be used. It is now listed under the Unavailable tab in the Samples panel.
- The application automatically reverts to the last known good sample.

NOTE: Depending on when the last known good sample was generated, this reversion could suddenly force a large number of steps to be processed in the browser's memory.

- You should consider generating a new sample immediately.

For more information, see *Overview of Sampling*. For more information on best practices, see <https://community.trifacta.com/s/article/Best-Practices-Managing-Samples-in-Complex-Flows>.

Running Job Basics

Contents:

- *Configure Job*
 - *Run Job*
 - *Iterate*
-

Configure Job

When you are ready to test your recipe against the entire dataset, click **Run** in the Transformer page. In the Run Job page, you specify the output formats and any compression to apply. Unless you are working with a large dataset, compression is unneeded for this basic walkthrough.

Tip: Optionally, you can disable generating a visual profile of your results. While the visual profile is very useful for examining issues in your recipe and iterating, it is a resource-intensive process. If you are working with large datasets that do not require additional debugging, you can consider disabling the profiling of your results. For more information, see *Overview of Visual Profiling*.

Tip: Depending on your product configuration, you may have multiple running environments available to you. In most cases, you should choose to use the default running environment, which is selected for you based on the size of the dataset.

For more information on the job execution options, see *Run Job Page*.

Run Job

To queue the specified job for execution, click **Run**.

The job is queued up for processing.

You can track progress in the Job Details page.

- If visual profiling was enabled for the job, click the Profile tab.
- When the job is completed, you can access results in the Output destinations tab.
- For more information, see *Job Details Page*.

Iterate

In the Profile tab of the Job Details page, you can review the effects of the transformation recipe across the entire dataset. Statistics and data histograms provide overall visibility into the quality of your transformation recipe.

All data



Results profile by column

#	FMID	RBC	MarketName	Primary_Website_or_URL	RBC	Address	RBC	Location_Description	Seasons
Valid	38	Valid	38	Valid	22	Valid	38	Valid	38
Mismatched	0	Mismatched	0	Mismatched	4	Mismatched	0	Mismatched	0
Empty	0	Empty	0	Empty	12	Empty	0	Empty	0
Top 20 values		Top 14 values		Top 20 values		Top 7 values		Top 20 values	
		58th and Chester Farmer...		http://www.grownc.org		587 Harrison Street, Kal...		Other	
Minimum		57th Street Greenmarket...		http://www.foodtrustmar...		3rd St N (Hwy 11) besid...		Private business parki...	
Lower quartile		52nd and Haverford Farm...		http://www.sandlercente...		3rd & Curry Street, Cars...		Local government build...	
Median		3rd Street N (hwy 11) b...		http://www.peoplesfoodc...		3808 North Meridian Str...		Faith-based instituti...	
Upper quartile		3rd & Curry St. Farmers...		http://www.pcfma.com/ma...		362 Plymouth Street, Abi...		Closed-off public stre...	
Maximum		38th & Meridian Farmers...		http://www.iatp.org/min...		29th and Wharton Street...		On a farm from: a bar...	
		33rd and Diamond Farmer...		http://www.foodtrustmar...		27 W Potomac Street, Bru...		Co-located with wholes...	
		32nd Street/Waverly Far...		http://www.experimental...		2349 S Hwy 127, Russell...			
		3 French Hens French Co...		http://www.carsonfarmer...		22nd and Tasker Streets...			
		29th and Wharton Farmer...		http://www.abquptowngro...		201 Market Street, Virgi...			
		25th Avenue Farmers' Ma...		http://www.abingtonsage...		1st Ave - E 92nd & 93 S...			
		22nd and Tasker Farmers...		http://www.abingdonfarm...		194 W 25th Avenue, San M...			
		2012 Wood County Farmer...		http://www.6701burnetro...		1622 6th St NE, Minneapo...			
		17th Ave Market		http://www.3frenchhensm...		1400 U Street NW, Washin...			
		175th Street Greenmarket				12th & Brandywine Stree...			

Figure: Visual Profile

See [Job Details Page](#).

Use the links in the Job Details page to resume working on your dataset sample and the related recipe, generating jobs when you think you are done, until you have generated the appropriate dataset.

Export Basics

After you have iterated on your recipe and generated a result that is to your satisfaction, you can export the transformed data.

Steps:

1. In the left nav bar, click the Jobs icon.
2. In the Jobs page, click the job identifier to open the job in the Job Details page. For more information, see *Job Details Page*.
3. Click the Output Destinations tab.

Export by:

- **Direct file download:** Click the file to download. From its righthand context menu, select **Download result**.

NOTE: Some file types cannot be downloaded.

- **Create new dataset:** You can create a new dataset from a generated output. Click the file. From its righthand context menu, select **Create imported dataset**. For more information, see *Build Sequence of Datasets*.
- **Publish:** If Designer Cloud powered by Trifacta® Enterprise Edition has been integrated with an external datastore, you can publish your results to a designated target. Click **Publish**. For more information, see *Publishing Dialog*.

Checkpoint: Done!

If you walked through this workflow in the application, you have imported, cleansed, transformed, and run a job to generate transformed results. Hopefully, this process has given you insight into the easy-to-use tools at your disposal through Designer Cloud powered by Trifacta Enterprise Edition and how quickly they can be used to transform imported datasets into clean and actionable data for use across the enterprise.

Common Tasks

Contents:

- *Import Tasks*
 - *Discovery Tasks*
 - *Validation Tasks*
 - *Structuring Tasks*
 - *Cleanse Tasks*
 - *Enrichment Tasks*
 - *Publishing Tasks*
 - *Project Management Tasks*
 - *User Management Tasks*
-

This section contains documentation on common methods for performing your data wrangling tasks in Designer Cloud powered by Trifacta® Enterprise Edition.

Import Tasks

- *Connect to Data*
 - *Share a Connection*
- *Import a File*
 - *Change File Encoding*
 - *Remove Initial Structure*
- *Import a Table*
 - *Disable Type Inference*
- *Import from Another Flow*
- *Import Excel Data*
- *Import Google Sheets Data*
- *Import PDF Data*
- *Create Dataset with Parameters*
 - *Parameterize Files for Import*
 - *Parameterize Tables for Import*
- *Create Dataset with SQL*

Discovery Tasks

- *Explore Suggestions*
- *Add or Edit Recipe Steps*
- *Filter Data*
- *Locate Outliers*
- *Compute Counts*
- *Calculate Metrics across Columns*
- *Compare Strings*
- *Analyze across Multiple Columns*
- *Parse Fixed-Width File and Infer Columns*
- *Generate a Sample*

Validation Tasks

- *Profile Your Source Data*
- *Validate Your Data*
- *Find Bad Data*
- *Find Missing Data*

- *Manage Null Values*

Structuring Tasks

- *Initial Parsing Steps*
- *Reshaping Steps*
- *Split Column*
- *Move Columns*
- *Delete Data*
- *Select*
- *Create Aggregations*
- *Pivot Data*
- *Unpivot Columns*
- *Window Transformations*
- *Working with Arrays*
- *Working with JSON v2*
- *Working with JSON v1*

Cleanse Tasks

- *Rename Columns*
- *Sanitize Column Names*
- *Change Column Data Type*
- *Copy and Paste Columns*
- *Create Column by Example*
- *Remove Data*
- *Deduplicate Data*
- *Compare Values*
- *Replace Cell Values*
- *Replace Values Using Patterns*
- *Replace Groups of Values*
- *Normalize Numeric Values*
- *Standardize Using Patterns*
- *Modify String Values*
- *Manage String Lengths*
- *Extract Values*
- *Format Dates*
- *Apply Conditional Transformations*
- *Prepare Data for Machine Processing*

Enrichment Tasks

- *Create New Column*
- *Add Two Columns*
- *Generate Primary Keys*
- *Add Lookup Data*
- *Append Datasets*
- *Join Data*
 - *Configure Range Join*
- *Insert Metadata*
- *Invoke External Function*

Publishing Tasks

- *Create Outputs*
 - *Create Output SQL Scripts*
- *Publish Results on Demand*

- *Reuse Recipe*

Project Management Tasks

- *Take a Snapshot*
- *Track Data Changes*
- *Add Comments to Your Recipe*
- *Create Custom Data Types*
- *Create Target*
- *Optimize Job Processing*
- *Diagnose Failed Jobs*
- *Schedule a Job*
- *Create Branching Outputs*
- *Build Sequence of Datasets*
- *Fix Dependency Issues*
- *Share a Flow*
- *Export Flow*
- *Import Flow*
 - *Reconnect Flow to Source Data*
 - *Reconnect Flow to Outputs*
- *Create or Replace Macro*
- *Apply a Macro*
- *Export Macro*
- *Import Macro*
- *Create Flow Parameter*
- *Flag for Review*
- *Manage Environment Parameters*

User Management Tasks

- *Change Password*
- *Configure Your Access to S3*

Import Tasks

The following workflows pertain to creating imported datasets for use in the product.

An **imported dataset** is a reference to a source of data. It is not a copy of the data.

NOTE: Designer Cloud powered by Trifacta® Enterprise Edition never modifies source data.

For more information, see *Object Overview*.

For more information on importing, see *Import Basics*.

Connect to Data

Contents:

- *Locate Connections*
 - *Use Connections*
 - *Read-Only*
 - *Write*
 - *Create Connection*
 - *Share Connection*
 - *Delete Connection*
 - *Automation*
-

When you import data into Designer Cloud powered by Trifacta® Enterprise Edition, you are creating a reference to a source of data; the source is never touched. When the data is required for use, Designer Cloud powered by Trifacta Enterprise Edition reads a sample of the source data into the application for your use. Data is read into the application through an object called a **connection**.

The following are the supported types of connection for the product:

- **Upload/Download:** You can upload data directly from your local desktop. You can also save it locally on export.
- **Base storage layer:** Your deployed instance of the product is connected to a base storage layer, where you can read sources and write your results.
- **Relational sources:** You can read from database tables into the product.
- **S3:** You can read and write to S3 locations.

Locate Connections

You already have a set of connections that you can use. Connections can be either read-only or read-write.

1. In the Home page, click the Library icon in the left nav bar.
2. In the Library page, click **Import Data**.
3. In the Import Data page, your list of available connections is displayed in the left nav bar.

See *Import Data Page*.

Tip: To get to the Connections page from anywhere in the Designer Cloud application , click the Connections icon in the left nav bar. See *Connections Page*.

Use Connections

Read-Only

1. In the Import Data page, select one of the available connections.
2. Navigate through the connection to select the asset to import.

3. Select the object and click **Open**.
 4. In the Import Data page, review the settings of the asset in the card in the right panel. Make updates as needed.
- See *Import a File*.
 - See *Import a Table*.

For more information, see *Import Data Page*.

Write

NOTE: You cannot write or publish results until connections have been created for you.

Write results:

You write results through a connection by specifying a set of settings.

1. In the Run Job page, click **Add Publishing Action**.
2. In the left nav bar, select the connection.
3. Specify the settings for the publishing action.
4. Run the job.
5. When it successfully completes, the specified results are published through the selected connection.

See *Run Job Page*.

Publish results:

After you have written results from a job, you can publish them to other storage through your configured connections.

1. In the Jobs page, locate the job whose results you wish to publish. Click its link.
2. In the Job Details page, click the Output Destinations tab. Then, click **Publish**.
3. In the Publishing Dialog, select the system to which you would like to publish the results.
4. When the results are successfully published, a message is displayed.

See *Publishing Dialog*.

Create Connection

NOTE: Connections may be created by your Trifacta administrator. Some connections require additional configuration outside of the application. See *Connection Types*.

When a new connection is created, it is initially available only to you.

Prerequisites:

Before you create a new connection, please verify the following:

- On the datastore, you have read and (optionally) write locations.
- You have credentials to use to connect to this datastore. These credentials have permissions on your read /write locations.
- Some datastores require a special connection string, which must be inserted as part of the connection object.

Read-only:

1. In the Import Data page, click the New icon in the left nav bar.
2. In the Create Connection window, specify the parameters of the connection.

For more information, see *Create Connection Window*.

Read-write:

1. In the Connections page, click Create Connection.
2. Click the connection category or search for a specific connection to create.
3. If a connection is grayed out:
 - a. It may already exist. Some connections types permit only one globally available connection.
 - b. It may not be supported in your product.
 - c. It may be read-only.
4. Click the name of the connection.
5. In the Create Connection window, specify the parameters of the connection.

See *Create Connection Window*.

Share Connection

Through the Connections page, you can share your private connection with other users.

NOTE: When you share a connection, you can choose to share your credentials with the connection. Those credentials may provide access to specific areas of the datastore.

1. In the Connections page, locate the connection you wish to share.
2. In the context menu for the connection, select **Share....**
3. Specify the type of sharing from the drop-down and, if applicable, the users with whom you wish to share.

See *Share Connection Dialog*.

Delete Connection

NOTE: You can delete a connection only if you are the connection owner and the connection is not used to import any datasets.

1. In the Connections page, locate the connection to remove.
2. In the context menu, select **Delete....**
3. The connection is deleted.

See *Connections Page*.

Automation

You can create, edit, or delete connections through the APIs. See *API Reference*.

Share a Connection

Contents:

- *Share a Connection*
- *Make a Connection Public*
- *Remove Sharing From a Connection*

This section provides an overview of sharing connections with other users for collaboration.

You can share connections with other users to use the same connection through the Connections page. For more information, see *Connections Page*.

For more information on sharing, see *Overview of Sharing*.

NOTE: Access to the Connections page in the application and privileges on connections is governed by roles in your workspace. For more information, please contact your workspace administrator.

Share a Connection

Steps:

1. From the Connections page, locate the connection to share.
2. From the context menu, select **Share**.
3. In the Share dialog, enter the name or email address of the user with whom you would like to share the connection.
4. Specify the privilege level of the user to whom you are sharing. For more information on sharing privileges, see *Overview of Sharing*.
5. As the owner of a connection, you can specify whether to share your credentials with other users who have access to the connection:
 - a. **Share credentials:** (default) The credentials specified in the connection definition are shared to each user of the connection.
 - b. **Do not share credentials:** The connection credentials are not shared. Each user who is shared the connection must specify their own credentials.
 - c. For more information on the implications of sharing credentials, see *Share Connection Dialog*.
6. Click **Share**.
7. The selected users can now see the connections in the Shared with Me tab of the Connections page and can use the connection.

Make a Connection Public

Only an administrator can make a connection public. For more information, see *Share Connection Dialog*.

Remove Sharing From a Connection

You can remove the sharing from a connection by performing the following steps:

1. From the Share dialog for connections, select the user to remove sharing.
2. From the drop-down next to the user, Select **Remove**.
3. The sharing for the connection is removed.

Import a File

You can import one or more files into the Library or immediately add them to a new or existing flow.

NOTE: When you import a file, the data is not stored in Designer Cloud powered by Trifacta® Enterprise Edition. What you create is an **imported dataset**, which is simply a reference to the source of the data. Designer Cloud powered by Trifacta Enterprise Edition never stores or modifies source data. For more information, see *Object Overview*.

Other import options:

- *Import a Table*
- *Import from Another Flow*

Steps:

- From the menubar, click **Library**.
- In the Library page, click **Import Data**.
- From the left sidebar in the Import Data page, select the connection where your data is located.
 - You must have read permissions on any directory and file that you wish to import.
 - **Upload:** Navigate your local desktop to select the file or files that you wish to upload.

Tip: You can select multiple files in the same directory for uploading at the same time.

- **File-based datastore:** If you are uploading from a file-based backend datastore, navigate the available directories to locate your file.
- **Microsoft Excel:** If you are importing an Excel file that contains multiple worksheets, you must select the worksheets to include as part of your import. For more information, see *Import Excel Data*.
- **Dataset with Parameters:** If you are importing multiple files with similar filenames, you can import them as part of the same dataset using parameters or variables. In this manner, you create a single imported dataset, which automatically includes any new files that appear in the directory and that follow the same filenames pattern. For more information, see *Create Dataset with Parameters*.
- Some aspects of the import process can be modified. In the right panel, click **Edit Settings** for a file that you have imported.
 - By default, the application applies a few steps to file-based imported datasets to attempt to organize them into tabular format and hides these steps from your recipe. As needed, you can disable these automated steps, so that the steps themselves appear in the Recipe panel. For more information, see *Remove Initial Structure*.
 - If your file uses a different file encoding than the default encoding, you can change it for the file during the import process. For more information, see *Change File Encoding*.
- When you are ready to complete the import process:

Tip: If present, you can click the **Add to new flow** checkbox, which adds the imported datasets to an *Untitled* flow. For more information, see *Flow View Page*.

- - If you are importing a single file: click **Continue** to load to add it to a new flow, create a recipe for the imported dataset, and begin editing that recipe. See *Transformer Page*.
 - If you are importing multiple files: click **Continue** to load them. See *Flow View Page*.
- Your files are available as imported datasets.

For more information, see *Import Data Page*.

Change File Encoding

Files are imported based on the default file encoding for Designer Cloud powered by Trifacta® Enterprise Edition. The default file encoding can be configured. For more information, see *Configure Global File Encoding Type*. As needed, you can override this default file encoding during the importing of individual datasets.

NOTE: All output files are written in UTF-8 encoding. For a list of supported types for input, see *Supported File Encoding Types*.

Tip: If you have already imported the dataset and need to change this setting, you can re-import the source and change the settings. In any flows that use the previously imported version of this dataset, you can change the input for any recipe that uses the old version to use this newly imported version. See *Flow View Page*.

Steps:

1. After you have selected or specified the file to import in the Import Data page, click **Edit Settings** for the dataset card in the right panel.
2. From the drop-down, select the preferred encoding to apply to this specific file.
3. Continue the import process.

Remove Initial Structure

When you import a dataset from a file, Designer Cloud powered by Trifacta® Enterprise Edition attempts to detect the structure of the file and to apply an initial set of parsing steps to the data to render it in tabular form for display in the Transformer page. For example, JSON files may be turned into a table of data as long as the structure of the data supports this structuring.

NOTE: Initial parsing steps are applied only to file-based sources of data.

These steps vary based on the file format of data that is being imported. For more information, see *Initial Parsing Steps*.

Depending on the dataset, you may need to modify these steps or rebuild them altogether. You can use the following steps to prevent Designer Cloud powered by Trifacta® Enterprise Edition from detecting the structure and automatically hiding these steps.

Tip: You should allow the product to detect the structure first. If it does not detect the structure well, you can experiment with disabling it and rebuilding the steps to meet your dataset requirements.

Tip: If you have already imported the dataset and need to change this setting, you can re-import the source and change the settings. In any flows that use the previously imported version of this dataset, you can change the input for any recipe that uses the old version to use this newly imported version. See *Flow View Page*.

NOTE: When the steps are completed, the initial parsing steps are listed in any recipe that you create from the imported dataset. If you wish to remove them altogether, you can delete them from the recipe.

Steps:

1. After you have selected or specified the file to import in the Import Data page, click **Edit Settings** for the dataset card in the right panel.
2. Deselect the Detect Structure checkbox.
3. Continue the import process.
4. When the imported dataset is added to a flow, it is listed as an **unstructured dataset**.
5. Select the dataset and click **Create new recipe**.
6. When you select the recipe, the initial parsing steps are listed in the right panel.
7. When the dataset is loaded into the Transformer page, you can modify these steps to improve the parsing or delete them altogether.

NOTE: Any step that breaks up the data into individual rows into individual rows must be the first step in the recipe. To create, enter **Break into rows** in the Search panel. See *Search Panel*.

Import a Table

You can import one or more tables into the Library or immediately add them to a new or existing flow.

NOTE: When you import a file, the data is not stored in Designer Cloud powered by Trifacta® Enterprise Edition. What you create is an **imported dataset**, which is simply a reference to the source of the data. Designer Cloud powered by Trifacta Enterprise Edition never stores or modifies source data. For more information, see *Object Overview*.

Other import options:

- *Import a File*
- *Import from Another Flow*

Steps:

- From the menubar, click **Library**.
- In the Library page, click **Import Data**.
- From the left sidebar in the Import Data page, select the connection to the relational datastore where your data is located.
- Browse the relational datastore to locate the table that you wish to import.
- You must have read permissions on any database and table that you wish to import. For more information, see *Using Databases*.
- **Create Dataset with SQL:** You can apply a custom SQL statement to import from a database. For more information, see *Create Dataset with SQL*.
- Some aspects of the import process can be modified. In the right panel, click **Edit Settings** for a table that you have imported.
 - The application's data types are applied to the table's columns during the import process. If needed, you can disable type inference, so that the data types of the original source are preserved, if possible, during import. For more information, see *Disable Type Inference*.
- When you are ready to complete the import process:

Tip: If present, you can click the **Add to new flow** checkbox, which adds the imported datasets to an `Untitled` flow. For more information, see *Flow View Page*.

- If you are importing a single table: click **Continue** to load to add it to a new flow, create a recipe for the imported dataset, and begin editing that recipe. See *Transformer Page*.
- If you are importing multiple tables: click **Continue** to load them into a new flow. See *Flow View Page*.
- Your tables are available as imported datasets.

For more information, see *Import Data Page*.

Disable Type Inference

When Designer Cloud powered by Trifacta® Enterprise Edition creates an imported dataset from a schematized source, the product applies its own type inferencing to the columns of the imported data. Type inferencing may be reapplied during some operations, such as the creation of samples or when data reshaping transformations are applied in the Transformer page.

If preferred, you can disable this type inferencing on the columns of your imported dataset. When the data is imported, the original types from the source system remain. Any types that do not have a corresponding match with the Trifacta data types must be manually typed in the application.

Methods of disabling:

Column data typing is applied to schematized sources in one of three ways:

1. Globally
2. Per-connection type inference settings override the global setting.
3. Per-file type inference settings override both global and per-connection settings.

For more information on applying global or per-connection type inference settings, see *Configure Type Inference*.

You can use the following steps to disable type inference applied to a specific file during the import process.

Tip: For imported datasets from relational sources, you can identify in Flow View whether type inferencing has been applied to the dataset. When the dataset is selected in Flow View, locate the Type Inference entry in the right panel.

Tip: If you have already imported the dataset and need to change this setting, you can re-import the source and change the settings. In any flows that use the previously imported version of this dataset, you can change the input for any recipe that uses the old version to use this newly imported version. See *Flow View Page*.

Steps:

1. After you have selected or specified the relational table to import in the Import Data page, click **Edit Settings** for the dataset card in the right panel.
2. Deselect the Column Data Type Inference checkbox.
3. Continue the import process.
4. When the dataset is loaded into the Transformer page, no new data typing is applied at all, unless you manually specify the Trifacta data types for the column.

Import from Another Flow

Contents:

- *Import Imported Dataset*
- *Import Reference Dataset*
- *Import Snapshot of Flow Output*

You can use one of the following methods to import data from another flow into your current flow.

NOTE: When you import a file or a reference, the data is not stored in Designer Cloud powered by Trifacta® Enterprise Edition.

Other import options:

- *Import a File*
- *Import a Table*

Import Imported Dataset

If another flow contains an imported dataset that you want to use, you can import it into your current flow.

NOTE: To use an imported dataset from another flow, you must have access to the dataset itself. If you are not the owner of the flow, it must be shared with you. If the connection used to import the dataset is not shared with you, you may have to build your own connection to the source. For more information, see *Overview of Sharing*.

Steps:

1. Open the target flow.
2. In Flow View, select **Add Datasets**.
3. In the Add Datasets to Flow dialog, click the Imported tab.
4. Browse the available datasets:
 - a. Select the one to import.
 - b. If you do not see it, click **Import datasets**. Navigate and select the dataset to import.
5. The dataset is imported into the flow.

Import Reference Dataset

For any flow, you can create a reference to a recipe in it. This **reference** enables the output of the recipe, after execution, to be used elsewhere. When you import this reference into another flow, you create a **reference dataset**.

NOTE: A reference dataset is a dynamic object. If the recipe that is the source of the reference changes, then the reference dataset may change without warning. In the flow that uses the reference dataset, you may see unexpected errors in your recipe. For more information, see *Fix Dependency Issues*.

Steps:

1. In the source flow, locate the recipe whose output you wish to use in another recipe.
2. Right-click and select **Add > Reference**.
3. The reference is created:

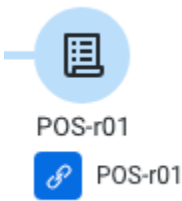


Figure: Reference object

4. In the right panel, click **Add to Flow....**
5. Select the flow to which to add the reference, or create a new one.
6. The reference is used to create the reference dataset in the target flow.



Figure: Reference dataset in a new flow

For more information, see *Flow View Page*.

For more information on references, see *Object Overview*.

Import Snapshot of Flow Output

If you need a snapshot of data at a point in time from another flow, you can do either of the following.

NOTE: Since you are generating an output file in both of the following cases, the imported dataset that you create from these outputs does not receive updated data.

1. **Snapshot of recipe in development:**
 - a. In the source flow, select a specific step in your recipe in the Recipe panel.
 - b. From the panel context menu, select **Download Sample as CSV**.
 - c. The recipe steps up to the selected step are performed on the current sample, and the current state of the sample is download in CSV format to your local desktop.
 - d. Through the Import Data page, you can import this generated file.
 - e. For more information, see *Take a Snapshot*.
2. **Snapshot of job results:**
 - a. In the source flow, select your recipe in the Recipe panel.
 - b. Select the output object icon above the recipe.
 - c. In the side panel, click **Run**. Specify the job outputs. For best results, select a CSV or JSON output. For more information, see *Run Job Page*.
 - d. When the job completes, click the job identifier. The Job Details page opens. See *Job Details Page*.
 - e. In the Job Details page, click the Output Destinations tab. For the generated output, select **Create imported dataset** from its context menu.
 - f. A new imported dataset is created in your Library.
 - g. In the target flow, add this dataset to your flow. See *Flow View Page*.
 - h. For more information, see *Build Sequence of Datasets*.

Import Excel Data

In addition to CSV and other formats, Designer Cloud powered by Trifacta® Enterprise Edition can directly import Microsoft® Excel® workbooks and folders containing workbooks. The worksheets of a workbook can be imported as:

- Individual datasets
- A single dataset
- A dataset with parameters

NOTE: When importing as a parameterized dataset, all selected worksheets are imported into a single dataset.

Limitations

- XLSX and XLS format are supported. Other Excel-related formats, such as XLSM format, are not supported.
- Filenames cannot include the hashtag (#) character.
- Filepath and source row number information is not available from original Excel files. These references return values from the CSV files that have been converted on the backend. For more information, see *Source Metadata References*.
- Source Excel files with cells bracketed by single double quotes may not be properly ingested if any terminating quotes are missing.

Tip: You can check the data quality bars for mismatched values or, for strings, the data histogram bars for anomalous values to see if the above issue is present. If so, deselect Detect Structure on import. Then, use a Split rows transformation applied to the affected column to break up the column as needed.

- Macros in your Excel files are not imported.
 - During import, cell formulas are applied, and the output values are used in the imported dataset.
- You cannot import password-protected Excel files.
- Import of Excel files with protected columns or cells is not supported.
- Compressed Excel files are not supported.
- If loading your Excel-based dataset in the Transformer page results in a blank screen, please take a new sample. The file requires conversion again with each generated sampling.

NOTE: When you share a flow that contains a dataset sourced from Microsoft Excel, the user with whom the flow is shared may receive a `Could not parse` error. In this case, the user does not have access to the original sample. The workaround is to take a new sample or to run a job on the full dataset.

- Latest state of the Excel file may not be reflected in the Transformer page due to caching. When you run a job, the platform always collects the latest version of the data and converts it to CSV for execution.

Use

When Excel data is imported into Designer Cloud powered by Trifacta Enterprise Edition, each sheet in an imported file must be converted to a CSV and then ingested for use.

Steps:

1. In the menu bar, click **Library**.

2. In the Library page, click **Import Data**. Select the connection to use. See *Import Data Page*.

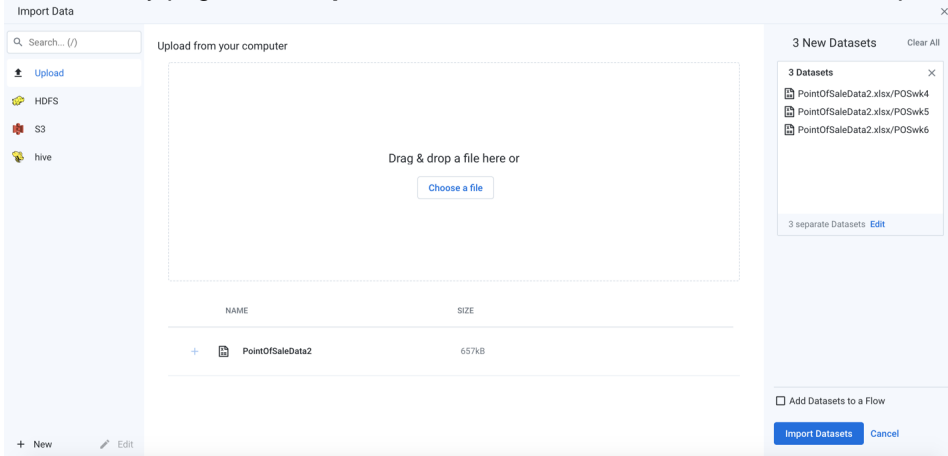


Figure: Import Excel workbook

Tip: If you experience issues uploading large XLS/XLSX files, you can convert the files to CSV files and then upload them.

3. After you select the workbook, it is uploaded and converted to CSV format and stored by the platform. Depending on the size of the workbook, this process may take a while.
4. By default, all worksheets in the workbook are imported as individual datasets. To change how the data is imported, click **Edit** in the right panel.

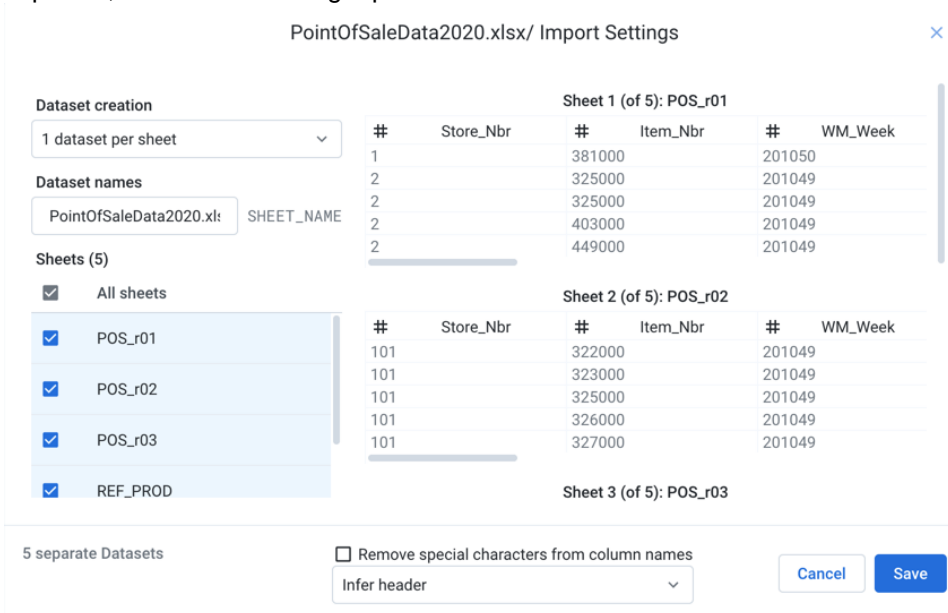


Figure: Import settings for Excel datasets

5. Dataset creation:
- 1 dataset per sheet:** (Default) Each selected sheet in the workbook is imported as a separate dataset.
Specify the base name of the datasets that you are creating. If you are creating a single dataset, the name of the workbook is used.
 - Selected sheets into 1 dataset:** All selected sheets in the workbook are combined and imported as a single dataset.

NOTE: The schemas of each dataset must match. Columns must be listed in the same order in each dataset. The column headers are taken from the first selected dataset.

- c. **All and future sheets into 1 dataset:** If the workbook is updated periodically with new sheets that you would like to add in the future, select this option. After initial selection of sheets, all sheets that are added to the workbook in the future are automatically added as part of the imported dataset.

Tip: Use this option to capture future additional sheets or changes to the names of the current sheets.

NOTE: When an imported dataset based on this option is first loaded into the Transformer page, the data grid displays an initial sample taken from rows in the first sheet only. When you take another sample from the Samples panel, data is collected from other sheets. For more information, see *Samples Panel*.

NOTE: This option is available only if you are connected to a backend file storage system.

6. Selected sheets:

- a. You can select the sheets to import.

NOTE: If you are importing a folder of Excel files, data preview and initial sampling are executed against the first file found in the folder.

- b. To preview the data of an individual sheet, mouse over a dataset and click **Jump to**.
7. Remove special characters from column names: Select this option to remove any special characters from the inferred column headers during import.
 8. From the drop-down, you can specify how you want the application to parse the data for column headers.
 9. To save changes, click **Save**.
 10. After your datasets have been added, you can edit the name and description information for each in the right navigation panel.
 11. Optionally, you can assign the new dataset(s) to an existing flow or create a new one to contain them

Import Google Sheets Data

Designer Cloud powered by Trifacta® Enterprise Edition can import Google® Sheets® spreadsheets.

The sheets of a spreadsheets can be imported as:

- Individual datasets
- A single dataset

Limitations:

NOTE: This integration provides access to all Google Sheets in the connecting user's account. Access includes spreadsheets with disabled options for download, print, or copy as well as hidden sheets within spreadsheets.

- Import-only support

NOTE: After you import a Google Sheet into Designer Cloud powered by Trifacta Enterprise Edition, renaming the source Google Sheet or a tab in it can break your datasets and flows. Details are below.

- Creation of a dataset with parameters from Google Sheets is not supported.
- Connected sheets or embedded external datasources in your Google Sheets are not supported.

Tip: If your connected sheet is linked to a table-based source, you may import that source directly into the product.

- If you have enabled Google Advanced Protection, this connection type does not work.
- A Google Sheet can contain up to 5,000,000 cells. Each cell can contain up to 50,000 characters.
 - Designer Cloud powered by Trifacta Enterprise Edition supports a maximum of 25,000 characters in a cell.
- Filepath and source row number information is not available from original Sheets. These references return values from the CSV files that have been converted on the backend. For more information, see *Source Metadata References*.
- Source Sheets files with cells bracketed by single double quotes may not be properly ingested if any terminating quotes are missing.

Tip: You can check the data quality bars for mismatched values or, for strings, the data histogram bars for anomalous values to see if the above issue is present. If so, deselect Detect Structure on import. Then, use a Split rows transformation applied to the affected column to break up the column as needed.

- If loading your Sheets-based dataset in the Transformer page results in a blank screen, please take a new sample. The file requires conversion again with each generated sampling.
- Latest state of the spreadsheet may not be reflected in the Transformer page due to caching. When you run a job, the platform collects the latest version of the data and converts it to CSV for execution.
- IMPORTRANGE function in Google Sheets is not supported for importing data from another sheet.

Process:

1. A spreadsheet can be read directly from your Google Drive.

NOTE: When you first use the Google Sheets connector, you must enable Designer Cloud powered by Trifacta Enterprise Edition to read all of your Google Drive data. When the connector is used, it locates only the Google Sheets data, including any Sheets that have been shared with you. All other data in Google Drive, including any Microsoft® Workbooks®, is ignored. You can then select the Sheet or Sheets you wish to import.

2. Sheets in a worksheet are ingested and written to Base Storage in CSV format.
3. CSV files are available for selection.
4. These CSV files are the source from which the imported datasets are created.

Steps:

1. In the menu bar, click **Library**.
2. In the Library page, click **Import Data**. Select the Google Sheets connection.

Tip: You can paste links that you gather from Google to select spreadsheets. To access a Google Sheet, edit the path and paste the link. Use this method for publicly available Google Sheets, too.

See *Import Data Page*.

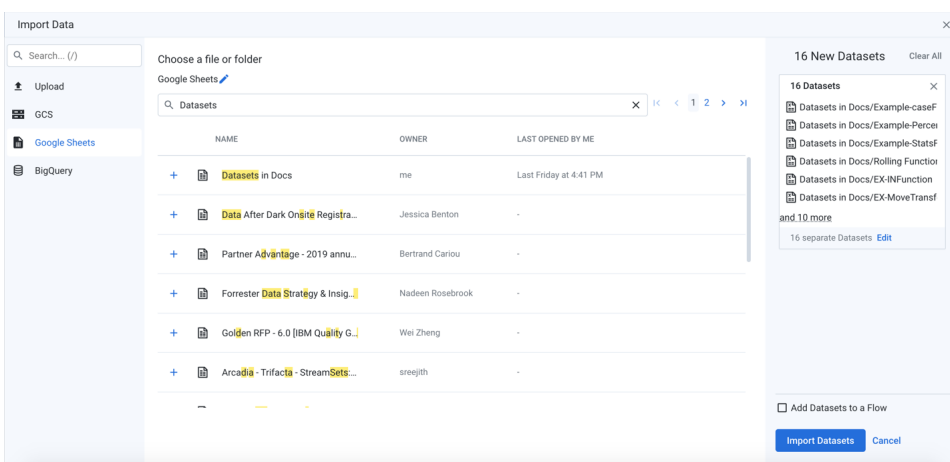


Figure: Import Google Sheets spreadsheet

3. After you select the spreadsheet, it is uploaded and converted to CSV format and stored. Depending on the size of the spreadsheet, this process may take a while.

4. By default, all sheets in the spreadsheet are imported as individual datasets. To change how the data is imported, click **Edit** in the right panel.

Datasets in Docs/ Import Settings ×

Dataset creation

1 dataset per sheet ▼

Dataset names

Datasets in Docs/ SHEET_NAME

Sheets (38)

☒ All sheets

- ☒ Example-SerialNumber
- ☒ ExtractValues-ExtractMatchesIntoArray
- ☒ EXAMPLE-DATEIFunctions
- ☒ Example-

Sheet 7 (of 38): ExtractValues-ExtractMatchesIntoArray

ABC	column2	ABC	column3	ABC
User Name		Location		Customer tweets
Mark		Berlin		Learnt more about the impo
Dave		New York		Learn how #NewYorklife onl
Christy		San Francisco		How can you quickly determ
James		U.K		Excited to announce that we

Sheet 8 (of 38): EXAMPLE-DATEIFunctions

🕒	Date	ABC	Product	#	Units	#:
	3/28/2020		ProductA	4		10
	3/8/2020		ProductB	4		20
	3/12/2020		ProductC	2		30
	3/23/2020		ProductA	0		10
	3/20/2020		ProductB	2		20

Sheet 9 (of 38): Example-TimeZoneConversionFunctions

#	rowId	ABC	datetime
---	-------	-----	----------

38 separate Datasets ☐ Remove special characters from column names

Infer header Cancel Save

Figure: Import settings for Google Sheets datasets

5. Dataset creation:

- a. **1 dataset per sheet:** (Default) Each selected sheet in the spreadsheet is imported as a separate dataset.
Specify the base name of the datasets that you are creating. If you are creating a single dataset, the name of the spreadsheet is used.
- b. **Selected sheets into 1 dataset:** All selected sheets in the spreadsheet are combined and imported as a single dataset.

NOTE: The schemas of each dataset must match. Columns must be listed in the same order in each dataset. The column headers are taken from the first selected dataset.

- c. **All and future sheets into 1 dataset:** If the spreadsheet is updated periodically with new sheets that you would like to add in the future, select this option. After initial selection of sheets, all sheets that are added to the spreadsheet in the future are automatically added as part of the imported dataset.

NOTE: When an imported dataset based on this option is first loaded into the Transformer page, the data grid displays an initial sample taken from rows in the first sheet only. When you take another sample from the Samples panel, data is collected from other sheets. For more information, see *Samples Panel*.

6. Selected sheets:

- a. You can select the sheets to import.

NOTE: Special characters in sheet names are filtered out.

- b. To preview the data of an individual sheet, mouse over a dataset and click **Jump to**.

7. Remove special characters from column names: Select this option to remove any special characters from the inferred column headers during import.

8. You can apply the column headers to your datasets during import. Select the required option from the drop-down list:
 - **Infer header:** (default) When selected, the Designer Cloud application infers the header based on the data in the import.
 - **Use first row as header:** When selected, the first row is used as the column headers.
 - **No header:** When selected, the inference is ignored and column headers are defined using generic names with no headers. For more information, see *File Import Settings*.
9. To save changes, click **Save**.
10. After your datasets have been added, you can edit the name and description information for each in the right navigation panel.
11. Optionally, you can assign the new dataset(s) to an existing flow or create a new one to contain them.

After import:

After you have imported the Google Sheet, you should avoid renaming the Google Sheet or any tab in it that is part of the imported datasets. If you rename a datasource, you can see one or more of the following issues in Designer Cloud powered by Trifacta Enterprise Edition:

- When you open a recipe using the dataset in the Transformer page, you may receive an error that the Base Storage path cannot be loaded.
- Collecting samples in the Transformer page returns a generic error message.

Import PDF Data

NOTE: This feature is in Beta release.

Designer Cloud powered by Trifacta® Enterprise Edition can directly import Adobe® Acrobat® PDF files containing one or more tables. The tables of a PDF can be imported as:

- Individual datasets
- A single dataset
- A dataset with parameters

NOTE: When importing as a parameterized dataset, all selected tables are imported into a single dataset.

PDF files can be uploaded from your local system. If Designer Cloud powered by Trifacta Enterprise Edition is connected to a backend file storage system, you can also import PDF files stored in readable directories.

Limitations

- PDF ingest is limited to 100 MB per file.
- Filepath and source row number information is not available from original PDF files. These references return values from the CSV files that have been converted on the backend. For more information, see *Source Metadata References*.
- You cannot import password-protected PDF files.
- Compressed PDF files are not supported.
- Conversion of large PDF files require non-linear increases in memory requirements on the Trifacta node.
- If loading your PDF-based dataset in the Transformer page results in a blank screen, please take a new sample. The file requires conversion again with each generated sampling.
- Latest state of the PDF file may not be reflected in the Transformer page due to caching. When you run a job, the platform always collects the latest version of the data and converts it to CSV for execution.

Enable

This feature is disabled by default. To enable, please complete the following:

Steps:

1. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
2. Locate the following parameter and set it to `true`:

```
"feature.enablePDFSupport": false,
```

3. Add references to the PDF format to the following parameters:

```
"webapp.convertableExtensions": "xls,XLS,xlsx,XLSX,pdf,PDF",  
"webapp.client.allowedFileExtensions": "<other_options>,pdf,PDF",
```

4. Save your changes and restart the platform.

Table Import

The PDF file format is a publishing format designed around visual layout of information, some of which may include tabular data. Table data in PDF files must be detected and converted into CSV data for proper ingestion in the platform. This ingest process occurs on the backend datastore.

To facilitate ingestion, the following requirements must be met for tables in your source PDF files:

- Non-tabular data in the file is ignored.
- Tables must be enclosed in a border. Each cell in the table must be bordered.
- Tabular data in the PDF cannot be scanned data, which is stored as an image. Data must be written into the file.
- When a table spans multiple pages, it is ingested as two separate CSV files, which can be combined later.
- If a file contains multiple tables, each table is converted as a separate dataset.

Tip: After import, separate datasets can be unioned together or integrated using as a dataset with parameters.

Import Steps

1. In the menu bar, click **Library**.
2. In the Library page, click **Import Data**. Select the connection to use. See *Import Data Page*.

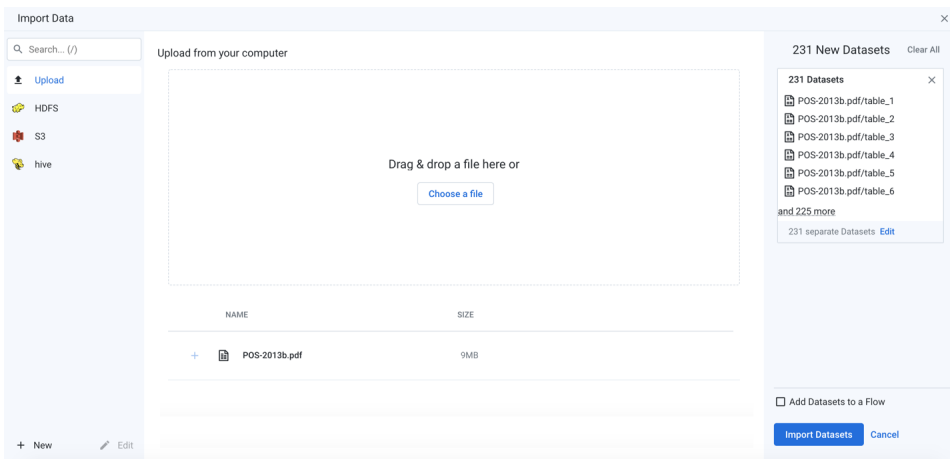


Figure: Import PDF file containing multiple pages

3. After you select the file, it is uploaded and converted to into individual CSV files for each page in the PDF file and then stored by the platform. Depending on the size of the file, this process may take a while.

4. By default, all pages in the PDF are imported as individual datasets. To change how the data is imported, click **Edit** in the right panel.

POS-2013b.pdf/ Import Settings

Dataset creation

1 dataset per table

Dataset names

POS-2013b.pdf/ TABLE_NAME

Tables (231)

☒ All tables

☒ table_1

☒ table_2

☒ table_3

☒ table_4

231 separate Datasets

☐ Remove special characters from column names

Infer header

Cancel Save

Table 1 (of 231): table_1

#	Store_Nbr	#	Item_Nbr	#	WM_Week
1		381000		201050	
2		325000		201049	
2		325000		201049	
2		403000		201049	
2		449000		201049 201049	

Table 2 (of 231): table_2

#	column2	#	column3	#	column4	
4		530000		201049		2/
4		543000		201049		2/
4		548000		201049		2/
4		551000		201049		2/
4		555000		201049		2/

Table 3 (of 231): table_3

Figure: Import settings for PDF datasets

5. Dataset creation:
- 1 dataset per table:** (Default) Each selected table in the PDF is imported as a separate dataset. Specify the base name of the datasets that you are creating. If you are creating a single dataset, the name of the PDF file is used.
 - Selected tables into 1 dataset:** All selected tables in the PDF are combined and imported as a single dataset.

NOTE: The schemas of each dataset must match. Columns must be listed in the same order in each dataset. The column headers are taken from the first selected dataset.

- All and future tables into 1 dataset:** If the PDF is updated periodically with new tables that you would like to add in the future, select this option. After initial selection of the tables to include, all PDF pages that are added to the PDF file in the future are automatically added as part of the imported dataset.

NOTE: This option is available only if you are connected to a backend file storage system.

NOTE: When an imported dataset based on this option is first loaded into the Transformer page, the data grid displays an initial sample taken from rows in the first table only. When you take another sample from the Samples panel, data is collected from other tables. For more information, see *Samples Panel*.

6. Selected tables:
- You can select the tables to import. A table can be a single page, or a single table among multiple on a page.

NOTE: If you are importing a folder of PDF files, data preview and initial sampling are executed against the first file found in the folder.

- b. To preview the data of an individual table, mouse over a dataset and click **Jump to**.
- 7. Remove special characters from column names: Select this option to remove any special characters from the inferred column headers during import.
- 8. You can also choose how to detect column headers from each imported table.
- 9. To save changes, click **Save**.
- 10. After your datasets have been added, you can edit the name and description information for each in the right navigation panel.
- 11. Optionally, you can assign the new dataset(s) to an existing flow or create a new one to contain them.

See *Import Data Page*.

Create Dataset with Parameters

Contents:

- *From File System*
 - *Parameterize bucket names*
 - *From Relational Sources*
 - *Edit Parameter*
 - *Apply Parameter Overrides*
 - *Apply parameter overrides for your flow*
 - *Apply parameter overrides for your job*
 - *Delete Parameter*
-

This section provides an overview on how to parameterize relational sources and files while importing data to your flow.

From File System

When browsing for data on your default storage layer, you can choose to parameterize elements of the path. Through the Import Data page, you can select elements of the path, apply one of the supported parameter types and then create the dataset with parameters.

NOTE: When you import a file, the data is not stored in Designer Cloud powered by Trifacta® Enterprise Edition. What you create is an imported dataset that is simply a reference to the source of the data. Designer Cloud powered by Trifacta Enterprise Edition never stores or modifies source data.

When you create a dataset with parameters in Designer Cloud powered by Trifacta Enterprise Edition, you can replace segments of the input path with parameters. Suppose you have the following files that you'd like to capture through a parameterized dataset:

```
//source/user/me/datasets/month01/2017-01-31-file.csv
//source/user/me/datasets/month02/2017-02-28-file.csv
//source/user/me/datasets/month03/2017-03-31-file.csv
//source/user/me/datasets/month04/2017-04-30-file.csv
//source/user/me/datasets/month05/2017-05-31-file.csv
//source/user/me/datasets/month06/2017-06-30-file.csv
//source/user/me/datasets/month07/2017-07-31-file.csv
//source/user/me/datasets/month08/2017-08-31-file.csv
//source/user/me/datasets/month09/2017-09-30-file.csv
//source/user/me/datasets/month10/2017-10-31-file.csv
//source/user/me/datasets/month11/2017-11-30-file.csv
//source/user/me/datasets/month12/2017-12-31-file.csv
```

A parameterized reference to all of these files would look something like:

```
//source/user/me/datasets/month##/YYYY-MM-DD-file.csv
```

Through the application, you can specify the parameters to match all values for:

- **##** - You can use a wildcard or (better) a pattern to replace these values.
- **YYYY-MM-DD** - A formatted Datetime parameter can replace these values.

For more information, see *Parameterize Files for Import*.

Parameterize bucket names

You can create environment parameters for your bucket names. For more information, see *Parameterize Files for Import*.

From Relational Sources

You can create datasets from a relational source by applying parameters to the custom SQL that pulls the data from the source. During import of database tables through relational connections, you can apply parameters to the SQL query that you use to define the imported dataset. In some scenarios, you may need to define the table to import using a variable parameter or to parameterize the time value associated with a table name. Using parameters, you can define the tables, columns, and conditions of the query that you use to bring in data from a relational database.

For more information, see *Parameterize Tables for Import*.

Edit Parameter

After you have created your dataset with parameters, you can edit the parameter as needed.

Steps:

1. In the left nav bar, select **Library**.
2. In the Library page, locate the dataset. From its context menu, select either of the following:
 - a. **Files:** Select **Edit parameters**. In the Edit Dataset with Parameters, click the parameter to modify its definition. For more information, see *Parameterize Files for Import*.
 - b. **Tables:** Click **Edit Custom SQL**. In the Custom SQL window, you can modify the SQL statement, including any parameters in it. For more information, see *Parameterize Tables for Import*.
 - i. For more information, see *Create Dataset with SQL*.

Apply Parameter Overrides

After you have created a parameterized dataset, you can apply overrides to the default value. These override values can be applied in the following cases.

Case	Precedence	Scenario
Job	1	When you choose to execute a job, you can set a new value for the parameter, which is applied for the specified job only.
Flow	2	<p>If your imported dataset containing a parameter is added to a flow, you can define an override value for the dataset's parameter through Flow View.</p> <p>Whenever a job is executed on the imported dataset within the flow, the override value is applied to the dataset.</p> <div>NOTE: If a job-level override is applied on top of a flow-level override, the job override value is applied to the job.</div>
Default	3	The default value for the parameter is used if no override is applied.

Apply parameter overrides for your flow

Steps:

1. In Flow View, select the dataset with parameters icon.
2. From the context menu, select **Parameter**.

3. In the Manager Parameter dialog, click the Overrides tab.
4. Edit the required values, click **Save**.

For more information, see *Manage Parameters Dialog*.

Apply parameter overrides for your job

You can apply job-level parameter overrides through the Designer Cloud application or through the APIs.

via Designer Cloud application :

1. In Flow View, select the output that you wish to generate.
2. In the right context panel, click **Run Job**.
3. In the Run Job page, you can specify job-level overrides at the bottom of the screen.

For more information, see *Run Job Page*.

via APIs:

For more information, see *API Workflow - Run Job*.

Delete Parameter

Steps:

1. In the Edit Dataset with Parameters screen, select the parameter that you wish to remove.

NOTE: Before you remove parameter, you may want to take note of the default value, which may need to be applied to the path or query after you remove the parameter.

2. In the popup, click **Delete**.
3. Save your changes.
4. The parameter is removed from the imported dataset definition.

Parameterize Files for Import

Contents:

- *Structuring Your Data*
 - *Steps*
 - *Add Datetime Parameter*
 - *Extend Datetime parameter*
 - *Add Variable*
 - *Parameterize bucket names*
 - *Add Pattern Parameter*
-

This section describes how to create datasets and replace segments by parameterizing the input paths to your data in Designer Cloud powered by Trifacta Enterprise Edition .

Structuring Your Data

Each file that is included as part of the dataset with parameters should have identical structures:

- Matching file formats
- Matching column order, naming, and data type
- Matching column headers. Each column in any row that is part of a column header in a dataset with parameters should have a valid value that is consistent with corresponding values across all files in the dataset.

NOTE: If your files have missing or empty values in rows that are used as headers in your recipe, these rows may be treated as data rows during the import process, which may result in unexpected or missing column values.

- Within each column, the data format should be consistent.
 - For example, if the date formats change between files in the source system, you and your recipe may not be able to manage the differences, and it is possible that data in the output may be missing.

NOTE: Avoid creating datasets with parameters where individual files or tables have differing schemas. Either import these sources separately and then correct in the application before performing a union on the datasets, or make corrections in the source application to standardize the schemas.

When working with datasets with parameters, it may be useful to do the following if you expect the underlying datasets to be less than 100% consistent with each other.

- Recreate the dataset with parameters, except deselect the Detect Structure option during the import step.
- In the Transformer page, collect a Random Sample using a full scan. This step attempts to gather data from multiple individual files, which may illuminate problems across the data.

Tip: If you suspect that there is a problem with a specific file or rows of data (e.g. from a specific date), you can create a static dataset from the file in question.

Steps

NOTE: Matching file path patterns in a large directory can be slow. Where possible, avoid using multiple patterns to match a file pattern or scanning directories with a large number of files. To increase matching speed, avoid wildcards in top-level directories and be as specific as possible with your wildcards and patterns.

1. In the Import Data page, navigate your environment to locate one of the files or tables that you wish to parameterize.
2. Click **Create Dataset with Parameters**.

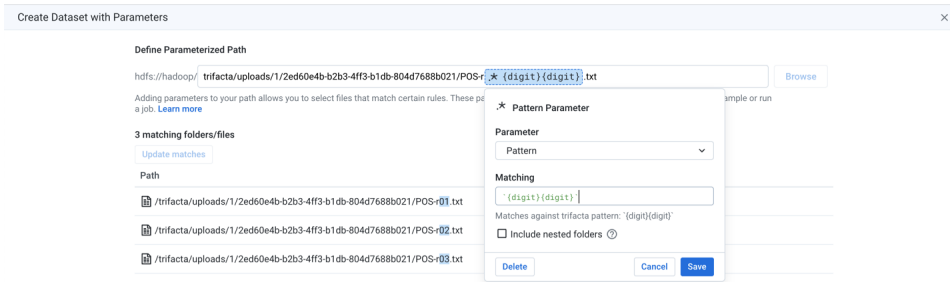


Figure: Create Dataset with Parameters

3. Within the Define Parameterized Path, select a segment of text. Then select one of the following options:
 - a. Add Datetime Parameter
 - b. Add Variable
 - c. Add Pattern Parameter - wildcards and patterns
 - d. For more information on limitations, see *Overview of Parameterization*.
 - e. If you need to navigate elsewhere, select **Browse**.
4. Specify the parameter. Click **Save**.
5. Click **Update matches**. Verify that all of your preferred datasets are matching.

NOTE: If you are matching with more datasets than you wish, you should review your parameters.

6. Click **Create**.
7. The parameterized dataset is loaded. See *Import Data Page*.

A flow containing a dataset with parameters has additional options for managing them. See *Flow View Page*.

Add Datetime Parameter

Datetime parameters require the following elements:

Format: You must specify the format of the matching date and/or time values using alphanumeric patterns. To review a list of example formats, click **Browse Date/Timestamp Patterns**.

You can also create custom formats using patterns. For example, the following regex pattern matches patterns like MM.DD.YYYY:

```
/[0-9][0-9]\.[0-9][0-9]\.[0-9][0-9][0-9][0-9]/
```

Date range: Use these controls to specify the range that matching dates must fall within.

NOTE: Date range parameters are case-insensitive.

Tip: Datetime parameters that you configure here are evaluated at the time of job execution. So, `now` refers to the time when the job is executed.

Time zone: The default time zone is the location of the host of the application. To change the current time zone, click **Change**.

For a list of supported time zone values, see *Supported Time Zone Values*.

Extend Datetime parameter

A parameterized dataset can support only one Datetime parameter. If you have multiple parts of the path that contain date information, you can create a Datetime element for each part.

Steps:

1. Within the Define Parameterized Path, select a segment of text for which to create the first part.
2. Create the Datetime parameter for this element. Remember to use the appropriate format for the part. For example, if you have highlighted a four-digit year for the part, the date format value should be: `YYYY`.
3. Then, select the second element and click the Extend Datetime Parameter icon.

The screenshot shows the 'Define Parameterized Path' interface. At the top, a path is displayed: `hdfs://trifacta/uploads/1/aa112099-4f6e-4925-93fa-d98677233965/`. Two segments are highlighted with blue boxes and icons: `YYYY` (for `2018`) and `MM` (for `04`). Below the path, a table lists matching folders/files. One entry is shown: `/trifacta/uploads/1/aa112099-4f6e-4925-93fa-d98677233965/2018-04-01-orders.txt` with a size of 285.95k. A 'Datetime Parameter' dialog is open for the second segment. It has fields for 'Format' (set to `MM`), 'Example: 2019' (set to `MM`), and 'Example: 01' (set to `01`). The 'Date range' is set to 'Date is last' with a dropdown arrow. Below that, a range of '1' years is specified, with a dropdown arrow. The date range is 'From 01/2018 to 12/2018' and the time zone is 'America/Los_Angeles time zone'. There are 'Cancel' and 'Save' buttons at the bottom.

Figure: Click the *Extend Datetime Parameter* icon to create additional parts to your *Datetime parameter*.

4. In the dialog, you can specify the date format of the second element of your Datetime parameter. Matches are made on the two elements, as well as any static text in between them.

Add Variable

A **variable** parameter is a key-value pair that can be inserted into the path.

- At execution time, the default value is applied, or you can choose to override the value.
- A variable can have an empty default value.

Name: The name of the variable is used to identify its purpose.

NOTE: If multiple datasets within the same flow share the same variable name, they are treated as the same variable.

Tip: Type `env.` to see the environment parameters that can be applied. These parameters are available for use by each user in the environment. For more information, see *Overview of Parameterization*.

Default Value: If the variable value is not overridden at execution time, this value is inserted in the variable location in the path.

NOTE: When you edit an imported dataset, if a variable is renamed, a new variable is created using the new name. Any override values assigned under the old variable name for the dataset must be re-applied. Instances of the variable and override values used in other imported datasets remain unchanged.

Parameterize bucket names

You can create environment parameters to specify your bucket names. An **environment parameter** is a variable name and String value that can be referenced by all users of the environment.

NOTE: A workspace administrator or project owner can create environment parameters. For more information, see *Environment Parameters Page*.

Uses:

- Parameterized bucket names are very useful when you are moving flows between workspaces or projects. When the flow is imported into a new workspace, the environment parameter references the appropriate bucket name in the new workspace.
- If you change source buckets or move data to a new storage bucket, updating the paths to your objects can be as simple as changing the value of the environment parameter where your data is stored.

For example, suppose you have two environments: Dev and Prod. You can create an environment parameter called `env.sourceBucketName` to store the name of the bucket from which all data in the workspace or project is imported.

Environment Name	Source Bucket Name	Environment Parameter Value
Dev	MyCo_Dev	<code>\$env.sourceBucketName = 'MyCo_Dev'</code>
Prod	MyCo_Prod	<code>\$env.sourceBucketName = 'MyCo_Prod'</code>

For more information, see *Overview of Parameterization*.

Add Pattern Parameter

In the screen above, you can see an example of pattern-based parameterization. In this case, you are trying to parameterize the two digits after the value: `POS-r`.

Include nested folders

When you create a wildcard or pattern-based parameter, you have the option to scan any nested folders for matching sources.

- If disabled, the scan stops when the next slash (/) in the path is encountered. Folders are not matched.
- If enabled, the scan continues to any depth of folders.

NOTE: A high number of files and folders to scan can significantly increase the time required to load your dataset with parameters.

Example 1: all text files

Suppose your file and folder structure look like the following:

```
//source/user/me/datasets/thisfile.txt
//source/user/me/datasets/thatfile.txt
//source/user/me/datasets/anotherfile.csv
//source/user/me/datasets/detail/anestedfile.txt
//source/user/me/datasets/detail/anestedfile2.txt
//source/user/me/datasets/detail/anestedfile4.txt
//source/user/me/ahigherfile.txt
```

Since the filenames vary significantly, it may be easiest to create your pattern based on a wildcard. You create a wildcard parameter on the first file in the `//source/user/me/datasets` directory:

```
//source/user/me/datasets/*.txt
```

For the specified directory, the above pattern matches on any text file (.txt). In the example, it matches on the first two files but does not match on the CSV file.

When the Include nested folders checkbox is selected:

- The first two files are matched.
- The next three files inside a nested folder are matched.
- The last file (ahigherfile.txt) is not matched, since it is not inside a nested folder.

Example 2: pattern-based files

Suppose your file and folder structure look like the following:

```
//source/user/me/datasets/file01.csv
//source/user/me/datasets/file02.csv
//source/user/me/datasets/file03.csv
//source/user/me/datasets/detail/file04.csv
//source/user/me/datasets/detail/file05.csv
//source/user/me/datasets/detail/file06.csv
//source/user/me/file07.csv
```

You create a pattern parameter on the first file in the `//source/user/me/datasets` directory with the following pattern-based parameter:

```
`file{digit}+`
```

The above pattern matches on the word `file` and a sequence of one or more digits. For example, suppose `file100.csv` lands in the directory at some point in the future. This pattern would capture it.

In the above example, this pattern matches on the first three files, which are all in the same directory.

When the Include nested folders checkbox is selected:

- The first three files are matched.

- The next three files inside a nested folder are matched.
- The last file (`file07.csv`) is not matched, since it is not inside a nested folder.

Wildcard

The easiest way to is to add a wildcard: *

A **wildcard** can be any value of any length, including an empty string.

Tip: Wildcard matching is very broad. If you are using wildcards, you should constrain them to a very small part of the overall path. Some running environment may place limits on the number of files with which you can match.

Pattern - Regular expression

Instead of a wildcard match, you could specify a regular expression match. **Regular expressions** are a standardized means of expressing patterns.

Regular expressions are specified between forward slashes, as in the following:

```
/my_regular_expression/
```

NOTE: If regular expressions are poorly specified, they can create unexpected matches and results. Use them with care. For a list of limitations of regular expressions for parameterization, see *Overview of Parameterization*.

The following regular expression matches the same two sources in the previous screen:

```
/\[0-9]*\[0-9]*/
```

The above expression matches an underscore (`_`) followed by any number of digits, another underscore, and any number of digits.

Tip: In regular expressions, some characters have special meaning. To ensure that you are referencing the literal character, you can insert a backslash (`\`) before the character in question.

While the above matches the two sources, it also matches any of the following:

```
_2_1
_1
_1231231231231231235245234343_
```

These may not be proper matches. Instead, you can add some specificity to the expression to generate a better match:

```
/\[0-9]{13}\[0-9]{4}/
```

The above pattern matches an underscore, followed by exactly 13 digits, another underscore, and then another 4 digits. This pattern matches the above two sources exactly, without introducing the possibility of matching other numeric patterns.

Pattern - Trifacta pattern match

A Trifacta pattern is a platform-specific mechanism for specifying patterns, which is much simpler to use than regular expressions. These simple patterns can cover much of the same range of pattern expression as regular expressions without the same risks of expression and sometimes ugly syntax. For more information on Trifacta patterns, see *Text Matching*.

Trifacta patterns are specified between back-ticks, as in the following:

```
`my_pattern`
```

In the previous example, the following regular expression was used to match the proper set of files:

```
/\[0-9]{13}\_[0-9]{4}/
```

In a Trifacta pattern, the above can be expressed in a simpler format:

```
`\_ {digit}{13}\_ {digit}{4}`
```

This simpler syntax is easier to parse and performs the same match as the regular expression version.

Parameterize Tables for Import

Contents:

- *Import Parameterized Tables*
 - *Create a custom SQL dataset*
 - *Parameterize dataset with a variable*
 - *Parameterize dataset with a timestamp*
 - *Examples*
 - *Pre-filter rows from a table*
 - *Run a weekly job on daily tables*
 - *Parameterize entire query*
-

This section provides an overview on how to apply parameters to the tables that you import as datasets.

During import of database tables through relational connections, you can apply parameters to the SQL query that you use to define the imported dataset. In some scenarios, you may need to define the table to import using a variable parameter or to parameterize the time value associated with a table name. Using parameters, you can define the specific tables that you use to bring in data from a relational database.

Following are the type of parameters you can apply for relational sources:

- **Timestamps:** Inserts a formatted timestamp when creating a custom SQL query.
- **Variables:** Inserts a value for the variable. This variable has a default value that you assign.

NOTE: Pattern-based parameters are not supported for relational imports.

For more information, see *Overview of Parameterization*.

Import Parameterized Tables

While importing data, you parameterize relational tables by creating custom SQL statements to specify the dataset. By default, when you import a table from a relational source, Designer Cloud powered by Trifacta Enterprise Edition generates a `SELECT *` statement to import the entire table. The Custom SQL enables you to customize the query to pull the data from the source system.

The following are the prerequisites and procedures for parameterizing the relational sources table:

Prerequisites:

- Connections must be created for your target database. See *Connections Page*.
- Verify that you have a read-only or read-write set of connections. For more information, see *Connect to Data*.

Create a custom SQL dataset

You can create a custom SQL dataset through the Import Data page.

Steps:

1. In the Designer Cloud application , click **Library** in the left nav bar.

2. In the Library page, click **Import Data**.
3. From the left side of the Import Data page, select the relational connection from which to import.
4. Depending on the type of relational connection, you may need to select the database or schema to browse.
5. Locate the tables to import. Take note of the table name or names.
6. Click **Create Dataset with SQL**. The Create Dataset with SQL window is displayed.

In this window, you specify the `SELECT` statement to retrieve the data from a table or tables that you specify.

NOTE: When specifying a SQL statement for your database, you are constructing a direct query of the database. You must use the syntax required by the database vendor.

For more information on creating datasets with SQL, see *Create Dataset with SQL*.

Parameterize dataset with a variable

A variable parameter enables you to insert variable into the query statement used to define your dataset. You can replace or highlight elements of the query to add parameters.

How to use variables:

- When a job is executed, the currently specified variable value is passed to the running environment. By default, the value that you specify as part of the dataset creation process is provided.
- You can override this value:
 - You can specify an override for a variable through Flow View. For more information, see *Manage Parameters Dialog*.
 - For any specific job run, you can specify an override value through the Run Job page. See *Run Job Page*.
- In this manner, you can specify the exact data that you wish to retrieve at the flow- or job-level.

Steps:

1. Create a custom dataset using SQL. For more information, see *Create a Custom SQL Dataset* above.
2. In the Create Dataset with SQL window, enter a `SELECT*` statement to retrieve data from the specified table. Click **Validate SQL** to verify that the query is properly specified.
3. Now, highlight the part of the query that you wish to parameterize. Click the Variable icon.

The screenshot shows the 'Create Dataset with SQL' window. At the top, it says 'Connection: postgres'. Below that, the SQL query '1 SELECT col1,col12 from <> varRegion' is displayed. A modal dialog titled '<> Variable' is open, showing the 'Name' field with 'varRegion' and the 'Default value' field with '01'. There are 'Cancel' and 'Save' buttons at the bottom of the dialog. Below the SQL editor, there is a 'Validate SQL' button. At the bottom of the window, there is a checkbox 'Infer column data types' which is checked, and 'Cancel' and 'Create Dataset' buttons.

Figure: Define Variable Parameter

4. In the Variable dialog, enter the following details:

A large empty rectangular box with a light green background, intended for defining the details of the variable parameter.

Tip: Type `env.` to see the environment parameters that can be applied. These parameters are available for use by each user in the environment. For more information, see *Overview of Parameterization*.

- a. **Name:** Enter a display name for the variable.
 - b. **Default value :** Enter a default value for the parameter.
5. Click **Save** to save the parameter.
 6. To verify that your SQL is still valid, click **Validate SQL**.
 7. If the SQL is valid, click **Create Dataset**.

Parameterize dataset with a timestamp

Timestamp parameters can be helpful when you want to filter datasets based on date and time format, time zone, or exact and relative start time. You can apply timestamp parameters based on the specific region or time zone for which the data is generated.

Steps:

1. Create a custom dataset using SQL. For more information, see *Create a Custom SQL Dataset* above.
2. In the Create Dataset with SQL window, enter a `SELECT*` statement to retrieve data from the specified table. Click **Validate SQL** to verify that the query is properly specified.
3. Now, highlight the part of the query that you wish to parameterize. Click the Timestamp icon.

The screenshot shows the 'Create Dataset with SQL' window. The SQL query 'SELECT col1, col2 from ' is entered. A 'Timestamp Parameter' dialog box is open, showing the following fields: 'Timestamp format' with the value 'YYYY-MM-DD_hh_mm', 'Timestamp value' with a dropdown set to 'Relative to job start date', and a section for 'minus' (dropdown), '1' (input), and 'days' (dropdown). Below these fields, an example text states: 'E.g., for a job start time 2020-11-30 12:48, the recorded timestamp would be 2020-11-29_12_48. America/Los_Angeles time zone. Change'. At the bottom of the dialog are 'Cancel' and 'Save' buttons. The main window has a 'Validate SQL' button and a 'Create Dataset' button at the bottom right.

Figure: Define Timestamp Parameter

4. In the Timestamp Parameter dialog, enter the following details:
 - i. **Timestamp format:** Specify the format for timestamp values.
 1. Example: `YYYY-MM-DD_hh_mm`.
 2. Values can express both date and time elements. For more information on the available tokens for formatting date and time values, see *Datetime Data Type*.
 - ii. **Timestamp value:** Select the value to record in the path:
 1. **Exact job start date:** recorded timestamp in path is the start time of the job.
 2. **Relative to the job start date:** recorded timestamp in path is relative to the start time of the job according to the settings that you specify here.
 - iii. **Time zone:** Click **Change** to change the time zone recorded in the timestamp.
 1. Example: `America/Los Angeles` or `Asia/Calcutta`.
 2. For more information on the available time zones, see *Supported Time Zone Values*.
5. Click **Save** to save the parameter.
6. To verify that your SQL is still valid, click **Validate SQL**.
7. If the SQL is valid, click **Create Dataset**.

Examples

In the following examples, you can see how dataset parameters can be used to pre-filter rows or parameterize the tables to include in your dataset.

NOTE: The syntax in these examples uses PostgreSQL syntax. For more information on basic syntax requirements, see *Create Dataset with SQL*.

Pre-filter rows from a table

Suppose you have a set of orders in a single table: `myOrders`. From this table, you want to be able to import a dataset that is pre-filtered for values in the customer identifier (`custId`) column. The following might be a query that you use for the `customerOrders` dataset to retrieve the orders for `custId=0001` from the `myOrders` table in the `transactions` database:

```
SELECT "custId","ordDate","prodId","ordQty","unitPrice" FROM "transactions"."myOrders" WHERE "custId" = "0001"
```

Tip: This example uses a variable parameter.

Steps:

In this case, you can do the following:

1. In the Create Dataset with SQL window, specify the above query. Click **Validate SQL** to verify that it works.
2. Now, highlight the value `0001`.
3. Select the Variable icon.
4. Specify your variable:
 - a. **Name:** `myCustId`
 - b. **Default Value:** `0001`
5. Click **Save**.
6. Before you create the dataset, validate the SQL.

Using the parameter:

- When the job is executed, the `customerOrders` dataset is pre-filtered to retrieve the data for `custId=0001` by default.
- You can override this variable value as needed:
 - At the flow level, you can define an override for the `myCustId` variable. For example, you can set the variable value to: `0002`. Whenever a job is run on this imported dataset, the value `0002` is passed to the running environment, which retrieves only the rows in the table where `custId=0002`.
 - When you run the job through the application, you can specify an override to the variable. This override takes precedence over the flow that was set at the flow level. So, for a specific run, you can set the value to `0003`, generating results for `custId=0003` only.

Tip: Job-level overrides are very useful when you run a job through the APIs. For more information, see *API Workflow - Run Job*.

- Then, the next time that a job is run from the flow using the dataset, the flow override value (`0002`) is used.

Run a weekly job on daily tables

Suppose you have a database that captures log data into separate tables for each date. Each table is named according to the following pattern:

```
20201101-ServerLogs
20201102-ServerLogs
20201103-ServerLogs
20201104-ServerLogs
20201105-ServerLogs
```

Once per week, you want to run a job to ingest and process the log entries from the preceding week.

The following could be a query that you use to retrieve all columns from a single file:

```
SELECT * FROM "logs"."20201101-ServerLogs"
```

Tip: This example uses a timestamp parameter.

Steps:

In this case, you can do the following:

1. In the Create Dataset with SQL window, specify the above query. Click **Validate SQL** to verify that it works.
2. Now, highlight the value 20201101 .
3. Select the Timestamp icon.
4. Specify your variable:
 - a. **Timestamp Format:** YYYYMMDD
 - b. **Timestamp Value:** Relative to job start date
 - i. Select minus, 7, days.
5. Click **Save**.
6. Before you create the dataset, validate the SQL.

Using the parameter:

When the job is executed, the imported dataset includes all of the tables whose timestamp format is within 7 days of the time when the job was started.

- You may need to modify the time zone setting on the Timestamp parameter if the log files were recorded using a different time zone.
- Typically, jobs created on a dataset like this one are executed according to a schedule.
 - For scheduled jobs, the value that is used for the job start date is the timestamp for when the job was scheduled to execute. It's possible that delays in starting the job could create a difference in the timestamps.
 - For more information, see *Schedule a Job*.

Parameterize entire query

You can turn the entire query of your custom SQL statement into a parameter. When you create your dataset with SQL, instead of entering any SQL in the window, create a variable parameter. For example, your parameter could be like the following:

- **Name:** selectCustomersTable
- **Value:**

```
SELECT * from "MDM"."customers"
```

If the SQL validates, then you can create the imported dataset using only this parameter.

How to use this parameter:

- By default, when the dataset is imported, all of the columns from the `customers` table are imported.
- As needed, you can configure overrides at the flow- or job-level to, for example, import only select columns. In the override, you specify the list of columns to gather only the data required for your needs.

Tip: This example uses a variable parameter.

Create Dataset with SQL

Contents:

- *Limitations*
 - *General*
 - *Single Statement*
 - *Multi-Statement*
 - *Enable*
 - *Use*
 - *Create with Variables*
 - *Create with timestamp parameter*
 - *SQL Validation*
 - *SQL Syntax*
 - *Troubleshooting*
 - *Snowflake*
-

As needed, you can insert custom SQL statements as part of the data import process. These custom SQL statements allow you to pre-filter the rows and columns of relational source data within the database, where performance is faster. This query method can also be used for wider operations on relational sources from within Designer Cloud powered by Trifacta® Enterprise Edition.

Limitations

General

All queries are blindly executed. It is your responsibility to ensure that they are appropriate. Queries like `DELETE` and `DROP` can destroy data in the database. Please use caution.

NOTE: Column names in custom SQL statements are case-sensitive. Case mismatches between SQL statement and your datasource can cause jobs to fail.

- SQL statements are stored as part of the query instance for the object. If the same query is being made across multiple users using private connections, the SQL must be shared and entered by individual users.

NOTE: If a dataset created from custom SQL is shared, collaborators are not permitted to edit the custom SQL.

- Each statement must be terminated with a semi-colon (;) and a newline:

```
SELECT * FROM myDB.myTable;
```

- SQL statements must be valid for the syntax of the target relational system. For more information on SQL examples, see *Supported SQL Syntax*.
- If you modify the custom SQL statement when reading from a source, all samples generated based on the previous SQL are invalidated.
- Declared variables are not supported.
- Common Table Expressions (CTEs) are not supported.

- For each SQL statement, all columns must have an explicit name. Example:
 - Function references such as:

```
UPPER(col)
```

- Must be specified as:

```
UPPER(col) as col_name
```

- When using custom SQL to read from a Hive view, the results of a nested function are saved to a temporary name, unless explicitly aliased.
 - If aliases are not used, the temporary column names can cause jobs to fail, on Spark in particular.
 - For more information, see *Hive Connections*.

Single Statement

The following limitations apply to creating datasets from a single statement.

1. All single-statement SQL queries must begin with a `SELECT` statement.
2. Selecting columns with the same name, even with " * ", is not supported and generates an ambiguous column name error.

Tip: You should use fully qualified column names or proper aliasing. See *Column Aliasing* below.

3. Users are encouraged to provide fully qualified path to table being used. Example:

```
SELECT "id", "value" FROM "public"."my_table";
```

4. You should use proper escaping in SQL.

Multi-Statement

These limitations apply to creating datasets using a sequence of multiple SQL statements.

NOTE: Use of multiple SQL statements must be enabled. See *Enable Custom SQL Query*.

1. **Repeatable:** When using multi-statements, you must verify that the statements are repeatable without failure. These statements are run multiple times during validation, datasets creation, data preview, and opening the dataset in the Transformer page.

NOTE: To ensure repeatability, any creation or deletion of data in the database must occur before the final required `SELECT` statement.

2. **Line Termination:** Each statement must terminate with a semi-colon and a new line. Example:

```
SELECT * FROM transactions.orders;
SELECT custId,custName FROM master.customers;
```

3. **Validation:** All statements are run immediately when validating or creating dataset.

NOTE: No DROP or DELETE checking is done prior to statement execution. Statements are the responsibility of the user.

4. **SELECT requirement:** In a multi-statement execution, the last statement must be a SELECT statement.
5. **Database transactions:** All statements are run in a transaction. DDL statements in most dialects (vendors) can't be run within a transaction and might be automatically committed by the driver.

Enable

Steps:

1. You apply this change through the *Workspace Settings Page*. For more information, see *Platform Configuration Methods*.
2. Locate the following setting:

Enable custom SQL Query

Setting	Description
enabled	Set to true to enable the ability to create datasets using customized SQL statements. By default, this feature is enabled.

Use

To use, please complete the following steps.

Steps:

1. In the Library page, click **Import Data**.
2. In the Import Data page, select a connection.
3. Within your source, locate the table from which you wish to import. Do not select the table.
4. Click the Preview icon to review the columns in the dataset.

Tip: You may wish to copy the database, table name, and column names to a text editor to facilitate generating your SQL statement.

5. Click **Create Dataset with SQL**. Enter or paste your SQL statement.

Through the custom SQL interface, it is possible to enter SQL statements that can delete data, change table schemas, or otherwise corrupt the targeted database. Please use this feature with caution.

Connection: hive

1 `SELECT INST#, BUCKET# FROM "AUDSYS"."CLI_SWP$7395268a$1$1"`

[Validate SQL](#)

☒ Infer column data types

[Cancel](#) [Create Dataset](#)

Figure: Create Dataset with SQL dialog

- a. For more information, see *Supported SQL Syntax*.
- b. To test the SQL, click **Validate SQL**. For details, see below.
- c. To apply the SQL to the import process, click **Create Dataset**.
6. The customized source is added to the right panel. To re-edit, click **Custom SQL**.
7. Complete the other steps to define your imported dataset.
8. When the data is imported, it is altered or filtered based on your SQL statement.
 - a. After dataset creation, you can modify the SQL, if needed. See *Dataset Details Page*.

Create with Variables

If parameterization has been enabled, you can specify variables as part of your SQL statement. Suppose you had table names like the following:

```
publish_create_all_types_97912510
publish_create_all_types_97944183
publish_create_all_types_14202824
```

You can insert an inline variable as part of your custom SQL to capture all of these variations.

Create Dataset with SQL

Connection: hive

1 SELECT * from `default`.`publish_create_all_types` iid`

<> Variable

Name

iid

Default value

97912510

Cancel Save

Validate SQL

☒ Infer column data types

Cancel Create Dataset

Figure: Insert variables in your custom SQL

In the above, custom SQL has been added to match the first example table. When the value is highlighted and the icon is clicked, the highlighted value is specified as the default value.

Tip: Type `env.` to see the environment parameters that can be applied. These parameters are available for use by each user in the environment. For more information, see *Overview of Parameterization*.

Provide a name for the variable, and click **Save**.

Through the Run Job page, you can specify overrides for the default value, so the same job definition can be used across all matching tables without much modification. For more information, see *Run Job Page*.

For more information on this feature, see *Overview of Parameterization*.

Create with timestamp parameter

You can insert a timestamp parameter into your custom SQL. These parameters are used to describe timestamp formats for matching timestamps relative to the start of the job at the time of execution.

NOTE: A SQL timestamp parameter only describes the formatting of a timestamp value. It cannot be used to describe actual values. For example, you cannot insert fixed values for the month to parameterize your input using this method. Instead, parameterize the input using multiple input variables, as described in the previous section.

NOTE: Values for seconds in a SQL timestamp parameter are not supported. The finest supported granularity is at the minutes level.

NOTE: When the dataset is created, the current date is used for comparison, instead of the job execution date.

In the following example, the timestamp parameter has been specified as YYYY-MM-DD:

```
SELECT * FROM <YYYY-MM-DD> ;
```

If the job executes on May 28th, 2019, then this parameter resolves as 2019-05-28 and gathers data from that table.

The screenshot shows the 'Create Dataset with SQL' dialog box. At the top, it says 'Connection: hive'. Below that, there's a SQL editor with the text '1 SELECT 1 as'. A clock icon is next to the SQL text. A modal dialog titled 'Timestamp Parameter' is open over the SQL editor. It has two main sections: 'Timestamp format' with a text input field containing 'YYYY-MM-DD', and 'Timestamp value' with a dropdown menu showing 'Exact job start date'. Below these, there's explanatory text: 'E.g., for a job start time 2019-10-18 02:39, the recorded timestamp would be 2019-10-18.' and 'America/Los_Angeles time zone. [Change](#)'. At the bottom of the modal are 'Cancel' and 'Save' buttons. In the background, there's a 'Validate SQL' button and a checkbox labeled 'Infer column data types'. At the bottom right of the main dialog are 'Cancel' and 'Create Dataset' buttons.

Figure: Insert timestamp parameter

Steps:

1. Click the Clock icon in the custom SQL dialog.
2. **Timestamp format:** You can specify the format of the timestamp using supported characters.

Tip: The list and definition of available tokens is available in the help popover.

3. **Timestamp value:** Choose whether the timestamp parameter is to match the exact start time or a time relative to the start of the job.

Tip: You can use relative timestamp parameters to collect data from the preceding week, for example. This relative timestamp allows you to execute weekly jobs for the preceding week's data.

4. To indicate that the timestamps are from a timezone different from the system timezone, click **Change**.
5. To save the specified timestamp parameter, click **Save**.

SQL Validation

You cannot create a SQL-based dataset if any of your SQL statements do not pass validation. Errors must be corrected in the SQL or in the underlying database.

- All **SELECT** statements are planned, which includes syntactical validation. However, these statements are not executed. Validation should be a matter of a few seconds.

- For multi-line statements, all non-SELECT statements are planned and executed. The final SELECT statement is only planned.

NOTE: For multi-line SQL statements, validation may take longer to complete if the non-SELECT statements require significant time to execute.

SQL Syntax

For more information on SQL syntax and supported variations, see *Supported SQL Syntax*.

For more information on SQL syntax for specific connections, please refer to the documentation for the connection type. See *Connection Types*.

Troubleshooting

Snowflake

Selecting time zone data returns null values in profiling and fails in publishing

When you import a column from Snowflake that contains time zone information, you may see the following behavior:

- Sampled data appears to import correctly into the Transformer page for the TIMESTAMP-based column.
- When a job is run, the visual profile for the output column based on this data indicates null values.
- When the data is published back to Snowflake, the publishing job fails.

The above issue is caused by the following:

- When data is imported into the Transformer page, it is automatically converted to UTC timezone during the JDBC ingestion step for displaying the sample in the application. This ingestion process is called by the application and outside of the application's control.
 - During this ingestion process, some auto-recognition and conversion to UTC of Datetime values is applied to the sample for display.
 - Example: You design a recipe step to parse the following Datetime format: 2020-10-11 12:13:14., which has been auto-converted to UTC.
- When a job is run:
 - The application instructs Snowflake to unload the entire dataset from Snowflake and write it the target location, bypassing this automatic conversion process.
 - The recipe that was created to handle the data in the sample does not properly handle the data that is directly unloaded from Snowflake.
 - In the previous example: The Datetime parsing in your recipe may receive an input that looks very different from what you parsed in the displayed sample: 2020-10-11 14:13:14 CEST.

Solution:

For a time stamp with a time zone, you must wrap your reference to it like the following:

```
TO_TIMESTAMP(CONVERT_TIMEZONE('UTC', <timestamp_column_or_function>))
```

Suppose your query was the following:

```
SELECT *, CURRENT_TIMESTAMP() AS current_time FROM MY_TABLE;
```

To address this issue, the query needs to be rewritten as follows:

```
SELECT *, TO_TIMESTAMP(CONVERT_TIMEZONE('UTC', CURRENT_TIMESTAMP())) AS current_time FROM MY_TABLE;
```

When the above wrapper function is applied, the data is imported normally and validated and published as expected.

Discovery Tasks

Use various tools and techniques to identify patterns, anomalies, inconsistencies, and other issues in your datasets.

Explore Suggestions

Contents:

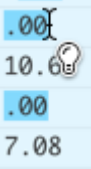

- *Select Something*
 - *Transformation types without suggestions*
- *Suggestion Cards*
- *Decide on the Suggestion*
- *Modify Suggestion*
- *Previews*
- *Iterate*

When you make selections in the Transformer page, Designer Cloud powered by Trifacta® Enterprise Edition responds by posting a set of suggestions for transformations to apply to the selected data in the sample. You can experiment with these suggestions to see what properly transformations your data.

Select Something

Selection Hints:

As you move the cursor around the Transformer page, the cursor changes when it is over a selectable data element.

Icon	Description
	Value or values can be selected.
	Column or columns can be selected.

In the data grid:

- You may select categories of values in a column's data quality bar: Valid, Mismatched, and Missing.
- You may select one or more values in a column's histogram. Use **SHIFT** or **CTRL** to select multiple values.
- Click a column for column-based operations. Click additional columns to add to your selection. Click a selected column to deselect.
- Select a whole or partial cell value to prompt suggestions for managing that specific string of data.

Tip: If you **CTRL**-select multiple partial values in a column of numeric data, the suggestion cards apply to the pattern that matches your selected strings. This does not apply to string data.

NOTE: In the data grid, selection of multiple values in a column is not supported for prompting of suggestions. However, through the Column Details panel, you can review and select patterns to triggers suggestions for sets of multiple values in your column. See *Column Details Panel*.

In the Column Browser or Column Details:

- Select categories of values in the data quality bar.
- Select one or more values in a column's histogram.

Transformation types without suggestions

The following types of transforms are not available through the suggestion cards. In most cases, these operations have too many parameters for a single set of selections to properly suggest the transformation.

- Lookup. See *Lookup Wizard*.
- Join. See *Join Window*.
- Pivot/Unpivot can be suggested when you select a column, instead of a value. See *Pivot Data*.

Suggestion Cards

Based on your selections, relevant suggestions appear in suggestion cards:

Suggestions

Extract values matching

See all

`http;`

`http;` starting after `{start}` ending before `/`

`http;` starting after `{start}` ending before {delim}`

Replace

`http;` with " in Primary_Website_or_URL

Edit

Add

Count values matching

See all

`http;`

`http;` starting after `{start}` ending before `/`

`http;` starting after `{start}` ending before {delim}`

Split on values matching

See all

`http;`

`http;` starting after `{start}` ending before `/`

`http;` starting after `{start}` ending before {delim}`

Extract list of values

Copyright © 2022 Trifacta Inc.

Page #92

Figure: Suggestion Cards

In the suggestion cards, the label at the top identifies the transformation type that is being recommended, followed by a brief preview of how the selection might transform the data.

Tip: A suggestion card may contain multiple variants for each suggestion. For example, in the previous image the `extract` suggestion has many variants, which can be selected and reviewed by selecting the dots at the bottom of the card.

Additional suggestions may be available. Try horizontal scrolling the set of cards to reveal new suggestions.

For more information, see *Selection Details Panel*.

Decide on the Suggestion

Before you decide on the suggestion to follow, you can do one of the following:

- **Select the suggestion to use.** After a suggestion is selected, the changes to the data are previewed in the Transformer page immediately. If there are multiple variants for the suggestion, verify that you are selecting the most appropriate one.
- **Select additional columns or values in the Transformer page.** A different pattern-based set of suggestions is presented to you. Make your transformation selection.
- **Modify the suggestion.** You may need to customize the suggestion to meet more specific requirements.
- **Start over.** If you discover that you have selected the wrong example data, click **Cancel**. Start again.

Modify Suggestion

To make the suggestion work for your specific use, you might need to modify the step. For example, for the selected text, you might need to define a replacement value, which Designer Cloud powered by Trifacta Enterprise Edition may not be able to guess. Click **Edit**.

For more information on how to modify, see *Transform Builder*.

Previews

As soon as you select a suggestion card, the changes are previewed in the Data Grid:

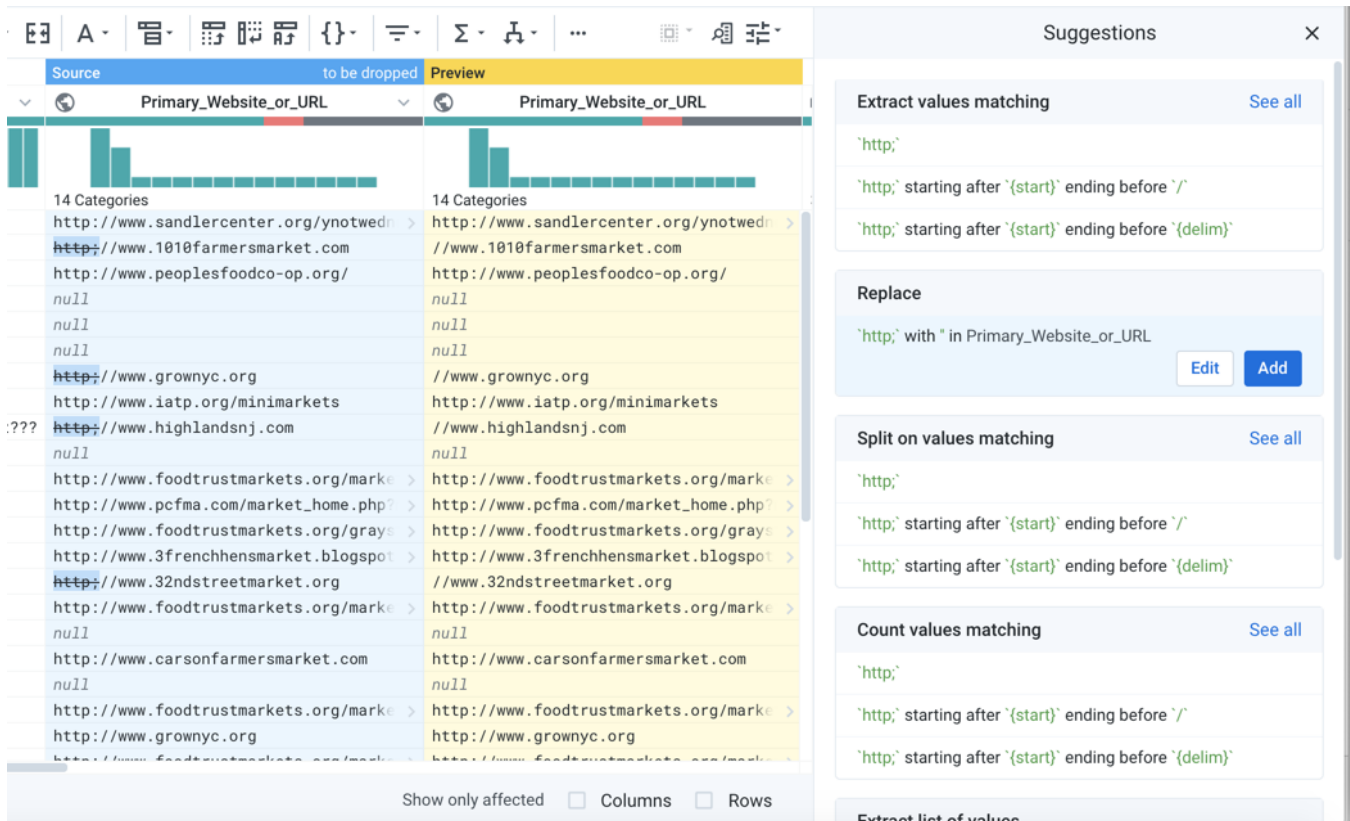


Figure: Previewed suggestion

In this manner, you can review the change before it is applied to the sample.

Tip: You can use the checkboxes in the status bar to display only the rows, columns, or both that are affected by the previewed transformation.

Iterate

Experiment away! Things to keep in mind:

- If you select the wrong thing, you can always cancel the recipe step. Start again.
- To delete a step that has already been added, select the step in the Recipe panel and click the Trash icon to delete it. See *Recipe Panel*.
- To step back a number of steps in the recipe, select the recipe to which you want to revert and start adding steps. Note that any added steps may invalidate the subsequent steps in your recipe.
- You can always undo and redo your most recent actions. Use the buttons on the top of the Recipe panel.
- An executed recipe does not change the source, so you can always step back to your recipe in the Transformer page and revert or modify recipe steps.

Add or Edit Recipe Steps

You can add or edit steps in your recipe through the Recipe panel, which is available on the right side of the Transformer page.

To add or edit steps in your recipe, do the following:

1. If it's not already opened, open the recipe panel:

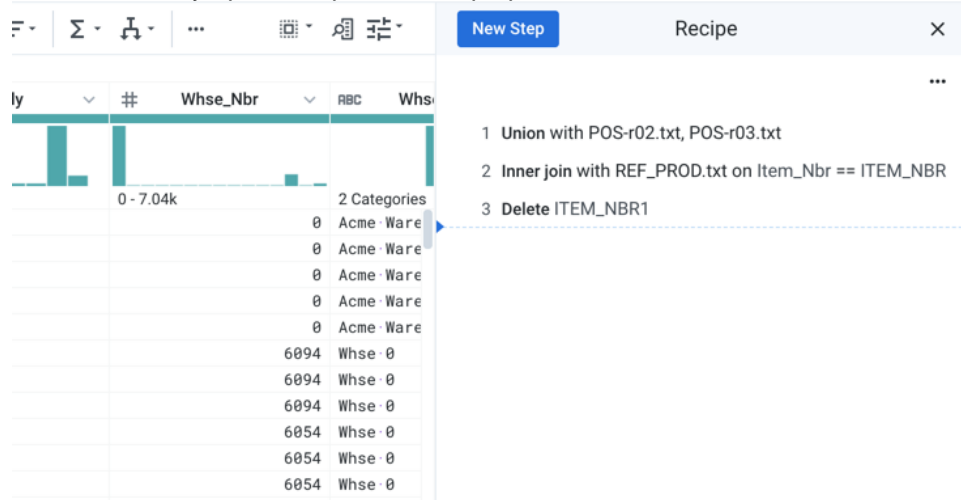


Figure: Recipe Panel

2. Edit a step:
 - a. Select the step in the recipe.
 - b. Click the Pencil icon.
 - c. Skip the next step.
3. Add a step:
 - a. In the recipe, select the step next to where you would like to add the step.
 - b. Select **Insert step before** or **Insert step after** from the drop-down menu.
4. To specify a step, you can:
 - a. Select something in the data grid. A set of suggestions is provided to you. See *Selection Details Panel*.
 - b. Enter some text in the Search panel. For the selected transformation, specify required and optional parameters in the *Transform Builder* to see a preview of the transform.
 - i. See *Search Panel*.
 - ii. See *Transform Builder*.
 - iii. See *Transform Preview*.
 - c. For more information, see *Transform Basics*.
5. After you have specified your step:
 - a. To add it to the recipe as it is currently specified, click **Add**. The step is inserted in the proper location.
 - b. To modify it, click **Edit**. You can edit the step in the Transform Builder.

Filter Data

Contents:

- *Filter Dataset*
 - *Filter Data Grid*
 - *Toggle display of columns*
 - *Filter the data grid*
 - *Filter during previews*
-

In the Transformer page, you can filter data from display in the data grid or from the dataset permanently.

Filter Dataset

You can apply various tools to remove columns of data and rows based on conditions you define. For more information on how to permanently remove rows and columns of data from the sample and the dataset, see *Remove Data*.

Filter Data Grid

You can make selections in the data grid interface to filter the sampled data that is displayed in the data grid.

Depending on your current tasks, you may want to hide columns or rows of the sample, so that you can focus on the task at hand.

- The displayed sample for smaller datasets may be the full dataset.
- Columns or filtered rows that are hidden from view are not removed from the dataset. They are included in any output. Please note that hidden columns can be affected by recipe steps. You should get in the habit of reviewing the Visible Columns panel and the Filters panel before running a job.

NOTE: Data grid filters do not remove any data. They can be used to hide data that is not important for the task at hand. The hidden data is still part of the sample and the full dataset.

Toggle display of columns

- To toggle display of a single column in the data grid, select the drop-down next to the column name. Then, select **Edit column > Hide**.
 - See *Column Menus*.
- To show a hidden column, click the Eye icon in the status bar at the bottom of the page. In the Visible Columns panel, click the Eye icon next to the column name. The column is displayed again in the data grid or column browser.

Tip: You can use the Visible Columns panel to toggle the display of single columns or multiple columns at the same time. See *Visible Columns Panel*.

- You can also hide one or more columns through the Column Browser.
 - In the Transformer page, click **Columns**. In the Column Browser, select the column or columns to hide. From the Actions drop-down, select **Edit > Hide**.
 - See *Column Browser Panel*.
 - Hidden columns must be resurfaced through the Visible Columns panel.

Filter the data grid

In the data grid panel, you can apply row- or column-based filters. At the top of the data grid, click **Filters**. In the drop-down:

- **Columns:** Search for individual columns or filter columns of a specific type. Filtered columns are displayed, and the rest are hidden.
- **Rows:** Highlight search term matches found in any column for a row.

For more information, see *Filter Panel*.

Filter during previews

When you are constructing transforms, the expected results are previewed in the data grid. As needed, you can narrow the display to only the affected rows, columns, or both. Select the appropriate checkbox or checkboxes in the status bar at the bottom of the Transformer page.

For more information, see *Data Grid Panel*.

For more information on previewed transforms, see *Transform Preview*.

Locate Outliers

Contents:

- *Single-column outliers*
 - *Data Histogram*
 - *Column Details*
 - *Tune standard deviation calculations*
 - *Custom functions*
 - *Methods for fixing single-column outliers*
-

Before you begin performing analytics on a dataset, it is important to identify and recognize outlier data patterns and values. Unusual values or patterns in the data can be sources for the following:

- Missing data. See *Find Missing Data*.
- Bad data. See *Find Bad Data*.
- Poorly formatted data
- Mismeasured data
- Data that skews statistics

This section provides guidance in how to locate these patterns of data in individual columns.

NOTE: Analysis of trends and outliers across multiple columns requires different techniques. See *Analyze across Multiple Columns*.

Single-column outliers

For assessing anomalies in individual columns, Designer Cloud powered by Trifacta® Enterprise Edition provides visual features and statistical information to quickly locate them.

Data Histogram

You can use the data quality bar and histogram to locate unusual values in your column data. The following example illustrates a dataset that contains two columns with outlier data. The first two rows are outliers with the subsequent rows to be consistently patterned data:Click to download the *Dataset-Outliers.csv* example data.

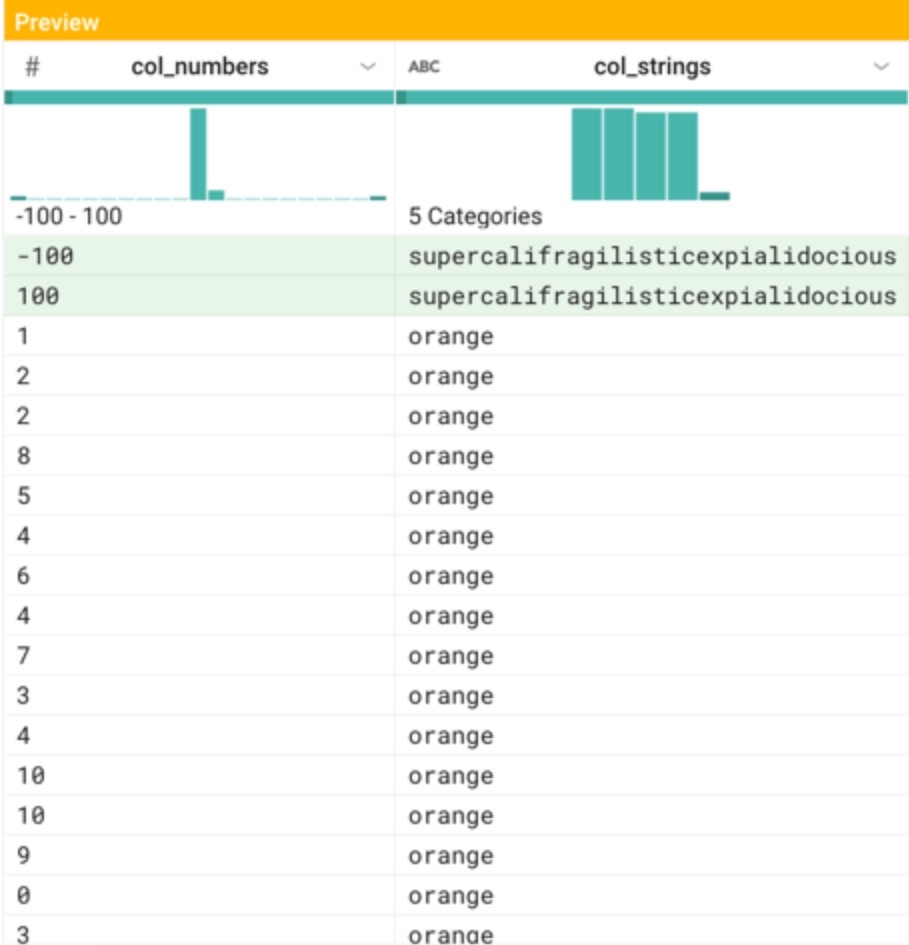


Figure: Numeric and string anomalies

Numeric data

The `col-numbers` column contains 100 random values 0-10, and singleton values -100 and 100.

In the histogram, you can see the outliers at the extremes of the graph. Note the slight visual distinction between the two extreme values and the values next to them, which are not represented in the column data.

Tip: In a histogram for numeric data, the spread between the extreme values and the more frequent values is a visual cue for outliers.

For numeric data, the range of values is displayed as part of the histogram. In this dataset, the extreme values are singletons. If a dataset contains more instances of outlier values, you should investigate further.

NOTE: In numeric datasets, a high count of outlier values may be statistically significant. You should review those values and related data in other columns before you perform operations to change or remove those rows.

Significant counts of unusual values

When your data contains a significant number of specific values, you should review them to see if the values have meaning. They may be placeholders for missing values. See *Find Missing Data*.

For numeric data, you should be skeptical of occurrences of the following values:

Suspicious value	Reason
-1	In system generated data, -1 is often an indicator of a failed result of some kind.
0	Some systems will fill missing numeric values with the number 0. You should verify the meaning of the value of 0 in your dataset.
555-####	In the United States, the phone number prefix 555 never corresponds to a person's phone number. These informational phone numbers and should not be considered as valid values for individuals' data.
65535	<div>In older versions of Microsoft Excel, 65,535 was the maximum number of rows permitted in a single sheet. NOTE: 65,536 is 2^{16}, which is the maximum number of data bits in a 16-bit system.</div>
2147483647	This value is the largest positive integer that can be stored in an <code>int</code> datatype by 32-bit systems, which are still sources of data. If you see these values, the source system may have been unable to represent the true value and wrote this value instead.
4294967295	This value is the largest raw value that can be stored in 32-bit systems. If you see these values, the source system may have been unable to represent the true value and wrote this value instead.
January 1st, 1900	This value is the earliest date recognized by Microsoft Excel. The true date may not be accurately represented in your data.
January 1st, 1904	This value is the earliest date recognized by Microsoft Excel for Macintosh.
00:00:00 UTC on January 1, 1970	This value is the earliest recognized date in UTC timestamp values. UTC timestamps are recorded as the number of milliseconds since this moment in time, stored as a signed 32-bit integer. Since datetime values may be represented in many different formats, you should identify these values for the date formats in your dataset.
03:14:07 UTC on Tuesday, 19 January 2038	<div>This value is the latest recognized date in UTC timestamp values. Since datetime values may be represented in many different formats, you should identify these values for the date formats in your dataset.<ul style="list-style-type: none">This limit is generally known as the "Year 2038" problem.</div>

String data

The `col-strings` column contains approximately 25 values for orange, red, green, yellow, and two instances of `supercalifragilisticexpialidocious`.

NOTE: For string-based data, outliers can be identified as strings with a low count of instances. These are the shorter stacks in the histogram.

Column Details

In the Column Details panel, you can review detailed statistics on the values in the currently selected column, including data on outliers. In the Transformer page, select **Column Details** from a column's drop-down.

Tip: In the Column Details panel, you can select specific outlier values, prompting suggestions, which enables you to take action on values identified by the platform as outliers.

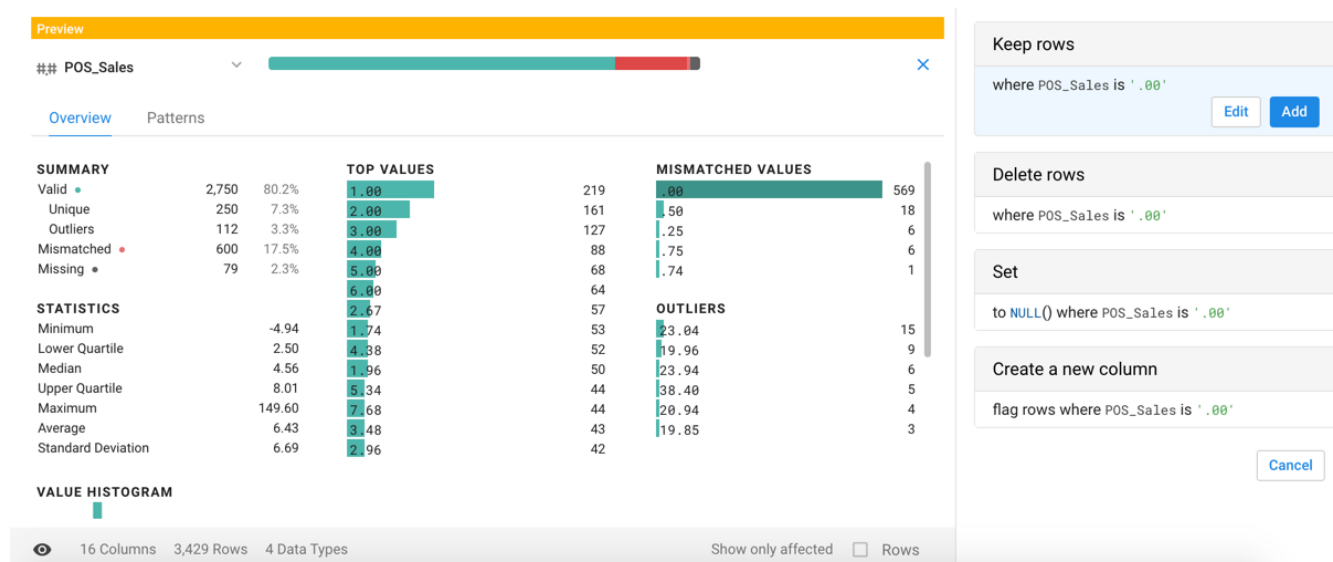


Figure: Outliers in the Column Details

Column Detail Statistics

The Column Details provides information on the following:

- Count of valid, mismatched, and missing values
- Count of value instances
- Min, max, and average
- Outlier values. See below.
- Lowest and highest quartiles
- Standard deviation

NOTE: For string-based data types, these statistics pertain to string length.

Tip: Any green bar in the Column Details can be selected to prompt for suggestions on actions, including values in Outliers, Value Histogram, and Frequent Values graphs. Multi-select values as needed.

See *Column Details Panel*.

Outliers

Designer Cloud powered by Trifacta Enterprise Edition uses a special set of computations to identify values that it designates as outliers.

For more information on these computations and other calculations in the Column Details panel, see *Column Statistics Reference*.

Tune standard deviation calculations

Although standard deviation information is available in the Column Details, you may want to generate your own standard deviation calculation. For example, the following transform generates a new column which computes the number of standard deviations that a column value is from the average value for the column:

Transformation Name	New formula
Parameter: Formula type	Single row formula
Parameter: Formula	<code>(col_numbers - AVERAGE(col_numbers)) / STDEV (col_numbers)</code>

You can then compute your own outlier function, using something like the following, which assumes that the above derived column has been renamed `col_numbers_stdev` and identifies outliers greater than 4 standard deviations from the average:

Transformation Name	New formula
Parameter: Formula type	Single row formula
Parameter: Formula	<code>ABS(col_numbers_stdev) > 4</code>

The above function generates boolean values in a new column, setting the value to `true` if the absolute value of the standard deviation for the `col_numbers_stdev` is more than 4. You can then perform operations based on the values written to this column or leave the column in place for downstream analytics tools.

The variance function is also supported.

Custom functions

If necessary, developers can build their own custom functions in their preferred programming language and then import them into the platform. See *User-Defined Functions*.

Methods for fixing single-column outliers

After you have identified the values that are outliers in your column, you must determine if those values are valid or invalid for your dataset. For example, a value of 0 may be a valid measurement, or it may be a value that was inserted for lack of a valid value.

For invalid values:

- **Fix the values.** The fix may require converting the values to be valid for the column's data type. For example, on import, values for 0 and 1 may be written as `false` or `true`. The following steps converts them back to numeric values:

Transformation Name	Edit column with formula
Parameter: Columns	<code>col_numbers</code>
Parameter: Formula	<code>IF((col_numbers == 'false'),'0', col_numbers)</code>

Transformation Name	Edit column with formula
Parameter: Columns	<code>col_numbers</code>
Parameter: Formula	<code>IF((col_numbers == 'true'),'1',col_numbers)</code>

- **Delete the rows.** If the removal of these records does not skew your data, you can create a simple delete statement. For example, the following deletes rows where the value in the `col_numbers` column is less than 25:

Transformation Name	Filter rows
Parameter: Condition	Custom formula
Parameter: Type of formula	Custom single
Parameter: Condition	<code>col_numbers < 25</code>
Parameter: Action	Delete matching rows

For valid values:

- **Let them be.** If the data is valid, do not remove it unless you have an explicit reason for doing so.
- **Convert to more meaningful values.** You can use the `set` transform to change outlier values to values that are valid for purposes of analysis.

NOTE: Please be aware that changing of values may impact the validity of your statistical analysis.

Example of overwriting values where values in the `col_numbers` column that are below 25 are set to the average value for the column. Otherwise, use the current value:

Transformation Name	Edit column with formula
Parameter: Columns	<code>col_numbers</code>
Parameter: Formula	<code>IF((col_numbers < 25), AVERAGE(col_numbers), col_numbers)</code>

Compute Counts

Contents:

- *Important Note on Counts*
 - *Visual Profiling*
- *Row and Column Counts*
 - *Computed row counts*
- *Count by Pattern*
 - *Count pattern or text*
 - *Count between patterns*
- *Count Functions*
 - *Aggregated counts*
 - *Conditional count functions*

Designer Cloud powered by Trifacta® Enterprise Edition supports computation of counts of rows, columns, and ad-hoc values within your data, so that you can make assessments of the quality, consistency, and statistical validity of your data.

Important Note on Counts

Any computed counts that you see in the Transformer page are computed from the displayed sample.

These computed counts reflect the entire dataset, only if the data grid is displaying the full dataset:



Figure: Data grid sample is the full dataset.

When the job is executed, however, any computations of counts are applied across the entire dataset.

Visual Profiling

When you run a job, you can enable the profiling of the job results, which renders a visual profile and some statistics on the dataset. This profile is available for review through the application. For more information, see *Overview of Visual Profiling*.

Row and Column Counts

In the status bar at the bottom of the data grid, you can review the current count of rows and columns in the displayed sample.

Tip: The row and column counts in the status bar may be useful for comparing the changes to these metrics between steps. For example, you can click step 2 in your recipe and then review these metrics. When you click step 3, these metrics may change.

Row counts: Depending on your method of sampling, the row counts may change. For more information, see *Overview of Sampling*.

Column counts: By default, all columns in the panel are displayed. Column counts should change only if you delete or hide them. For more information on toggling display of columns, see *Visible Columns Panel*.

For more information, see *Data Grid Panel*.

Computed row counts

You can use the following functions to identify and compute the row counts in your dataset.

Function Name	Description
COUNT Function	<div>Generates the count of rows in the dataset. Generated value is of Integer type.</div> <div>Tip Typically, this function is used as part of an aggregation, in which rows are grouped according to shared values in other columns. This function can also be applied without grouping, which is called a flat aggregate. More information on how to apply aggregated counts is below.</div>
ROWNUMBER Function	<div>Generates a new column containing the row number as sorted by the <code>order</code> parameter and optionally grouped by the <code>group</code> parameter.</div>
SOURCE ROWNUMBER Function	<div>Returns the row number of the current row as it appeared in the original source dataset before any steps had been applied.</div> <div>NOTE: This function may fail to return results if the original source row information is not available. For example, if you have performed a join between multiple datasets, the source row number information cannot be computed. Similarly, if you compute this function and then perform a join, the results may not make sense.</div> <div>Tip: You can pair this function later with the MIN or MAX functions to compute the highest and lowest row number information.</div>

Count by Pattern

These transformations allow you to compute counts of literals or patterns in a cell's values. Then, you can perform calculations on this new column of values to compute metrics across the dataset.

Count pattern or text

The following example computes the number of references in the `tweet` column for `My Company`:

Transformation Name	Count matches
Parameter: Option	Text or pattern
Parameter: Text or pattern to count	'My Company'
Parameter: New column name	tweetCompanyReferences

Suppose, however, that the company has multiple ways in which it is reference. It could be:

- My Company
- My Co
- My Company, Inc.

You can modify the above transformation to use a `Pattern` to capture these variations:

--	--

Transformation Name	Count matches
Parameter: Option	Text or pattern
Parameter: Text or pattern to count	`(My Company My Co My Company, Inc.)`
Parameter: New column name	tweetCompanyReferences

If needed, you can use the following to add up all of the counts in `tweetCompanyReferences` to determine the total number.

NOTE: Keep in mind that this sum reflects only the sum of values in the sample in the data grid. When you run a job containing this calculation, it is applied across all rows in the dataset.

Transformation Name	New formula
Parameter: Formula type	Single row formula
Parameter: Formula	<code>SUM(tweetCompanyReferences)</code>
Parameter: New column name	<code>sum_tweetCompanyReferences</code>

Count between patterns

You can also collect counts of values between two patterns within a cell's value. In this manner, you can analyze a more constrained substring of the cell value.

The following transformation calculates the URLs in each row of the `msgText` column, assuming that the URL begins with `http://` or `https://` and ends with `.com` or `.net`:

Transformation Name	Count matches
Parameter: Option	Between two delimiters
Parameter: Starting pattern	`(http\:\/\ https\:\/\)`
Parameter: Include as part of the match	Selected
Parameter: Ending pattern	`(\.com \.net)`
Parameter: Includes as part of the match	Selected
Parameter: Ignore case	Selected
Parameter: New column name	<code>countURLs</code>

Count Functions

Aggregated counts

You can perform calculations based on groups that you define as part of the calculation. These groupings, called **aggregations**, are powerful tools for delivering insightful analysis on your data.

In the following example, several aggregated computations, including the `COUNT` function are performed on transactional data, which is grouped by region (`regionId`) and product (`prodId`):

Transformation Name	Group by
Parameter: Group by 1	regionId
Parameter: Group by 2	prodId
Parameter: Values 1	SUM(sales)
Parameter: Values 2	COUNT()
Parameter: Type	Group by as new column(s)

NOTE: The above calculation inserts two new columns into the dataset. Alternatively, you can choose to do a full replacement of the dataset with these aggregated counts. For more information, see *Pivot Data*.

Conditional count functions

You can use a set of functions that count occurrences, based on conditions. In the following list of functions:

- Some of the conditions are implicit in the function itself. For example, `COUNTA` counts values that are non-null.
- Some conditions are specified as part of the function. For example, `COUNTIF` tabulates counts provided a specified condition is met.

Function Name	Description
<i>COUNTIF Function</i>	Generates the count of rows in each group that meet a specific condition. Generated value is of Integer type.
<i>COUNTA Function</i>	Generates the count of non-null rows in a specified column, optionally counted by group. Generated value is of Integer type.
<i>COUNTAIF Function</i>	Generates the count of non-null values for rows in each group that meet a specific condition.
<i>COUNTDISTINCT Function</i>	Generates the count of distinct values in a specified column, optionally counted by group. Generated value is of Integer type.
<i>COUNTDISTINCTIF Function</i>	Generates the count of distinct non-null values for rows in each group that meet a specific condition.

The following transformation counts the rows where the length of `msgText` is longer than 140 characters, grouped by `userId`:

Transformation Name	Group by
Parameter: Group by 1	userId
Parameter: Values 1	COUNTIF(LEN(msgText)>140)
Parameter: Type	Group by as new column(s)

Calculate Metrics across Columns

You can use a variety of mathematical and statistical functions to calculate metrics within a column.

- See *Aggregate Functions*.
- See *Math Functions*.

To calculate metrics across columns, you can use a generalized version of the following example.

Source:

Your dataset tracks swimmer performance across multiple heats in a race, and you would like to calculate best, worst, and average times in seconds across all three heats. Here's the data:

Racer	Heat1	Heat2	Heat3
Racer X	37.22	38.22	37.61
Racer Y	41.33	DQ	38.04
Racer Z	39.27	39.04	38.85

In the above data, Racer Y was disqualified (DQ) in Heat 2.

Transformation:

To compute the metrics, you must bundle the data into an array, break out the array into separate rows, and then calculate your metrics by grouping. Here are the steps:

1. When the data is imported, you may need to create a header for each row:

Transformation Name	Rename columns with a row
Parameter: Option	Use row as header
Parameter: Row	1

2. The columns containing heat time data may need to be retyped. From the drop-down next to each column name, select Decimal type.
3. The DQ value in the Heat2 column is invalid data for Decimal type. You can use the following transformation to turn it into a missing value. For purposes of calculating averages, you may or may not want to turn invalid data into zeroes or blanks. In this case, replacing the data as 0.00 causes improper calculations for the metrics.

Transformation Name	Replace text or patterns
Parameter: Column	Heat2
Parameter: Find	'DQ'
Parameter: Replace with	' '

4. Use the following to gather all of the heat data into two columns:

Transformation Name	Unpivot columns
Parameter: Columns	Heat1, Heat2, Heat3

Parameter: Group size	1
------------------------------	---

5. You can now rename the two columns. Rename `key` to `HeatNum` and `value` to `HeatTime`.
6. You may want to delete the rows that have a missing value for `HeatTime`:

Transformation Name	Delete rows
Parameter: Condition	ISMISSING([value])

7. You can now perform calculations on this column. The following transformations calculate minimum, average (mean), and maximum times for each racer:

Transformation Name	New formula
Parameter: Formula type	Multiple row formula
Parameter: Formula	MIN(HeatTime)
Parameter: Group rows by	Racer
Parameter: New column name	'BestTime'

Transformation Name	New formula
Parameter: Formula type	Multiple row formula
Parameter: Formula	AVERAGE(HeatTime)
Parameter: Group rows by	Racer
Parameter: New column name	'AvgTime'

Transformation Name	New formula
Parameter: Formula type	Multiple row formula
Parameter: Formula	MAX(HeatTime)
Parameter: Group rows by	Racer
Parameter: New column name	'WorstTime'

8. To make the data look better, you might want to reformat the values in the `AvgTime` column to two decimal points:

Transformation Name	Edit column with formula
Parameter: Columns	AvgTime
Parameter: Formula	NUMFORMAT(AvgTime, '##.00')

Results:

After you use the Move transformation to re-organize your columns, the dataset should look like the following:

Racer	HeatNum	HeatTime	BestTime	WorstTime	AvgTime
Racer X	Heat1	37.22	37.22	38.22	37.68
Racer X	Heat2	38.22	37.22	38.22	37.68
Racer X	Heat3	37.61	37.22	38.22	37.68
Racer Y	Heat1	41.33	38.04	41.33	39.69
Racer Y	Heat3	38.04	38.04	41.33	39.69
Racer Z	Heat1	39.27	38.85	39.27	39.05
Racer Z	Heat2	39.04	38.85	39.27	39.05
Racer Z	Heat3	38.85	38.85	39.27	39.05

Compare Strings

Contents:

- *Find Substrings*
- *Compare String Ends by Pattern*
- *Match Strings*
 - *Exact matching*
 - *Doublemetaphone matching*
- *Compare Strings*

Unlike other types of data, text data has very few restrictions on the kinds of values that appear in a cell. In the application, this data is typically inferred as String data type. As a result, finding string values that mean the same thing can be a challenge, as minor differences in their content or structure can invalidate a match.

This section provides some methods for comparing strings.

- Some target systems may impose limits on the lengths of imported values. For more information on managing the lengths of your strings, see *Manage String Lengths*.

Find Substrings

You can use the following functions to locate sub-strings that are part of a column's value.

Function Name	Description
<i>LEFT Function</i>	Matches the leftmost set of characters in a string, as specified by parameter. The string can be specified as a column reference or a string literal.
<i>RIGHT Function</i>	Matches the right set of characters in a string, as specified by parameter. The string can be specified as a column reference or a string literal.
<i>FIND Function</i>	Returns the index value in the input string where a specified matching string is located in provided column, string literal, or function returning a string. Search is conducted left-to-right.
<i>RIGHTFIND Function</i>	Matches the right set of characters in a string, as specified by parameter. The string can be specified as a column reference or a string literal.
<i>SUBSTRING Function</i>	Matches some or all of a string, based on the user-defined starting and ending index values within the string.

The following transformation checks the left five values of the lowercase version of the ProdId column to see if it matches xxxx-. If the value is detected, then the ProdName value is set to NO_NAME:

Transformation Name	Edit with formula
Parameter: Columns	ProdName
Parameter: Formula	IF(LEFT(LOWER(ProdId,5))=='xxxx-', 'NO_NAME', ProdName)

Compare String Ends by Pattern

You can use the STARTSWITH and ENDSWITH functions to determine if a string begins or ends with a specified pattern.

Tip: These functions are most useful for performing pattern-based checks on strings. For string literals, you can use the `LEFT` and `RIGHT` functions. See below.

The following transformation inserts the value `error` in the `custCodeStatus` column if the `custCode` value begins with six digits in a row:

Transformation Name	Edit with formula
Parameter: Columns	<code>custCodeStatus</code>
Parameter: Formula	<code>IF(STARTSWITH(custCode,`{digit}{6}`), 'error',custCodeStatus)</code>

See *STARTSWITH Function*.

See *ENDSWITH Function*.

Match Strings

Exact matching

You can use the `EXACT` function to compare if two strings are exact matches. String inputs can be literals, column references, or expressions that evaluate to strings.

NOTE: The `EXACT` function evaluates for exact matches. Whitespace or capitalization differences return `false`.

You can nest function expressions inside of the `EXACT` reference to eliminate common and perhaps not useful differences between strings. In the following transformation, a value of `true` is inserted into the `matches` column, if `colA` and `colB` are exact matches, after whitespace and case differences have been removed:

Transformation Name	New formula
Parameter: Formula	<code>IF(EXACT(LOWER(REMOVEWHITESPACE(colA)))=EXACT(LOWER(REMOVEWHITESPACE(colB))), 'true', 'false')</code>
Parameter: New column name	<code>matches</code>

Doublemetaphone matching

The platform also supports the doublemetaphone algorithm for fuzzy matching. This algorithm provides mechanism for proximity matching; the `DOUBLEMETAPHONEEQUALS` function supports an optional second parameter to define the strength of the algorithm.

This algorithm works by generating two separate encodings for each string: a primary encoding and a secondary encoding. You can experiment with these encodings using the `DOUBLEMETAPHONE` function. See *DOUBLEMETAPHONE Function*.

This algorithm can be applied to compare two strings, as in the following transformation.

Transformation Name	New formula
Parameter: Formula	<code>DOUBLEMETAPHONEEQUALS(colA,colB,'strong')</code>
Parameter: New column name	<code>matches</code>

- The first two parameters of the function are the string literals, column references, or functions returning strings to compare.
- The third parameter is optional. It determines the level of matching required to return `true`. Options:

Match threshold	Description
'strong'	Both primary encodings must match.
'normal'	At least one primary encoding must match either of the other string's encodings
'weak'	A primary or secondary encoding from each can match.

- For more information, see *DOUBLEMETAPHONEEQUALS Function*.

Compare Strings

For string values, you can use the string comparison functions to check how strings compare using Latin collation settings.

Tip: Any column can be converted to String data type to use these functions.

Collation refers to the organizing of written content into a standardized order. String comparison functions utilize collation rules for Latin. A summary of the rules:

- Comparisons are case-sensitive.
 - Uppercase letters are greater than lowercase versions of the same letter.
 - However, lowercase letters that are later in the alphabet are greater than the uppercase version of the previous letter.
- Two strings are equal if they match identically.
 - If two strings are identical except that the second string contains one additional character at the end, the second string is greater.
- A **normalized version** of a letter is the unaccented, lowercase version of the letter. In string comparison, it is the lowest value of all of its variants.
 - a is less than .
 - However, when compared to b, a = .
 - The set of Latin normalized characters contains more than 26 characters.

This table illustrates some generalized rules of Latin collation.

Order	Description	Lesser Example	Greater Example
1	whitespace	(space)	(return)
2	Punctuation	'	@
3	Digits	1	2
4	Letters	a	A
5		A	b

Resources:

NOTE: In the following set of charts (linked below), the values at the top of the page are lower than the values listed lower on the page. Similarly, the charts listed in the left nav bar are listed in ascending order.

For more information on the applicable collation rules, see <http://www.unicode.org/charts/collation/>.

Available functions:

- *STRINGLESSTHAN Function*
- *STRINGLESSTHANEQUAL Function*
- *STRINGGREATERTHAN Function*
- *STRINGGREATERTHANEQUAL Function*

Analyze across Multiple Columns

This section describes some techniques for performing analysis across data stored in multiple columns. For example, you may want to analyze combinations of height and weight. Some options:

- **Consolidate dimensions to a single metric.** For example, height and weight can be combined using a BMI (body mass index) calculation. Then, use available outlier analysis capabilities in Designer Cloud powered by Trifacta® Enterprise Edition. Below, you can review a method for bringing together similar data from multiple columns into a single column for easier analysis.
- **Flag outlier values of individual columns**, perhaps giving each column a weighting factor (e.g. 0.5). Sum the outliers and their weights together.
- **Defer analysis** until the data has arrived in the target system.
- **Build a custom function** in another programming language. See *User-Defined Functions*.

If you have homogeneous data across multiple columns, such as multiple individual events recorded in a single row, you can use a different method to calculate metrics. See *Calculate Metrics across Columns*.

In some cases, you may need to identify outliers across multiple columns of data. For example, you have a dataset containing scores from three separate tests taken by a set of individuals. Your columns may look like the following:

- LastName
- FirstName
- TestScore1
- TestScore2
- TestScore3

You can download the *Dataset-TestScores.csv* dataset. Most calculations, such as standard deviation, work for a single column of data. To perform analysis across all three columns, you must reshape the above dataset to look like the following:

- LastName
- FirstName
- TestNumber
- TestScore

This steps below outline the workflow for this example. The full recipe is provided at the bottom of this section.

Steps:

1. Load the TestScores dataset into the Transformer page. It should already be split out into five separate columns.
2. The three columns listed side by side are data that has been organized in a pivot table. To break down this data, you must unpivot the data, which breaks down the data into a *key* column (containing TestScore1, TestScore2, TestScore3) and a *value* column, which contains individual test scores.

Transformation Name	Unpivot columns
Parameter: Columns	TestScore1 , TestScore2 , TestScore3
Parameter: Group size	1

3. Rename the generated column of test scores to TestScore.
4. The numeric information in the *key* column values can be extracted using the following:

Transformation Name	Extract text or pattern
Parameter: Column to extract	key

from	
Parameter: Option	Custom text or pattern
Parameter: Text to extract	`{digit}`

- The key2 column contains just the numeric data now. Rename this column to TestNumber. You can delete the key column now.
- The dataset does not contain a primary key, which field containing a unique identifier for each row. The combination of last name, first name, and test number is a unique identifier for each row in the dataset:

Transformation Name	Merge columns
Parameter: Columns	LastName, FirstName, TestNumber
Parameter: Separator	' - '

- Rename the new column to TestID. Typically, primary keys are listed as the first field in a dataset. You might want to move the column before the LastName column.
- You may have noticed that the data is still organized by name (first and last) and test number, so that an individual's tests are scattered throughout the dataset. To reorganize the information, you can re-aggregate the data using the following:

Transformation Name	Pivot table
Parameter: Row labels	LastName, FirstName, TestNumber, TestID
Parameter: Values	SUM(TestScore)
Parameter: Max number of columns to create	1

Tip: The above retains all instances of tests that have been taken. If you are only interested in the average test score, you can remove the TestNumber and TestID groupings and change the SUM function to AVERAGE. In the results, you have one average for each test taker.

- You may want to rename the aggregation column. Your final dataset should look like the following:

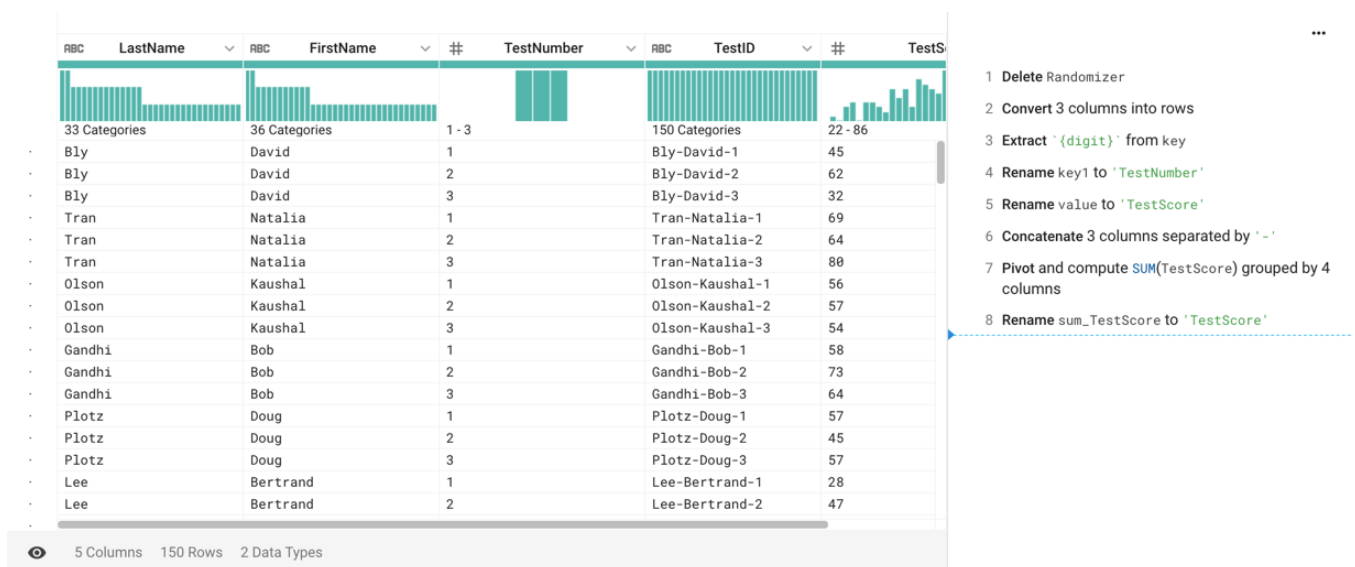


Figure: Single column of test scores

Now that your columns of data have been consolidated to a single column, you can use the single-column transforms and functions to perform analysis.

For more information on identifying outliers in this data, see *Locate Outliers*.

Parse Fixed-Width File and Infer Columns

For datasets that have a fixed width for each row, determining the column breaks can be more challenging, due to the uncertain number of spaces and tabs between each data element. With enhanced pattern matching, the application can help you identify the appropriate locations to break columns and then trim down the data to eliminate the whitespace padding.

Steps:

1. Import your fixed-width dataset through the application and begin wrangling.
2. The data should now look similar to the following:

Figure: Fixed-width dataset after import

3. From the drop-down to the right of the column name, select **Column Details**.
4. In the Column Details panel, click the Patterns tab.
5. Click in the All Patterns area.

NOTE: Selecting a specific pattern token will generate suggestions for only that particular token.

NOTE: If the application has inferred that the dataset is fixed-width, then the All Patterns area is the only available selection. If the dataset is not inferred as fixed-width, you should see multiple categories of patterns.

6. From the suggestion cards, click the Split one.
7. Close the Column Details panel.
8. In the Transform preview window, verify that the column splits look ok.
 - a. If a column contains multiple columns of data, click **Edit**.
 - b. Verify that you are splitting based on position numbers, which means that column splits are done based on the number of characters from the left side of each line.
 - c. Your recipe step might look similar to the following:

Transformation Name	Split columns by positions
Parameter: Column to split	column1
Parameter: Option	By positions
Parameter: Positions	7, 67, 117, 167, 217, 221, 239, 251, 253, 303, 315, 317, 329, 341, 391, 400, 512, 560, 610, 630, 650, 660

- d. In the list of values for positions, insert a new position number for the column or columns that contain multiple columns of data.
 - e. Verify your changes in the Transform Preview panel.
9. Click **Add**.
 10. Verify that the columns are split correctly.
 11. You can use the following step to remove the whitespace from each cell value.

Transformation Name	Edit column with formula
Parameter: Column	*
Parameter: Formula	TRIM(\$col)

12. Click **Add**.

Generate a Sample

Contents:

- *When to Take a New Sample*
 - *Limitations*
 - *Collect a New Sample*
 - *Cancel a Sample*
 - *Example - Random sample*
 - *Example - Filter-based sample*
 - *Example - Anomaly-based sample*
 - *Load a Sample*
 - *Delete a Sample*
 - *Sample Types*
 - *Invalid Samples*
 - *Collected Samples*
 - *Review Sample Jobs*
 - *Best Practices*
-

When you transform your data in the Transformer page, you are performing these transformations on a sample of the total dataset. As needed, you can generate new samples using a variety of algorithms to acquire other slices of your data.

A sample is always collected from the initial rows of the dataset. Whenever you create a recipe and open the dataset in the Transformer page, Designer Cloud application automatically generates the initial sample by loading the first 10 MB of your dataset.

- If your dataset is less than 10 MB, then the entire dataset is loaded as an initial sample.
- For datasets larger than 10 MB, the first 10MB of rows are loaded into the Transformer page.

Tip: On the Transformer page, this first sample is listed as **Initial Data**.

When to Take a New Sample

The initial sample allows you to get started immediately building your recipe steps. However, your recipe and dataset may require additional samples. For example:

- If you have a very long dataset with many rows, there may be statistically significant values that are not part of the first 10MB of data. The recipe steps that you create may not affect those rows properly, since you have not seen any data from them.
- If you have a very wide dataset with many columns, you may need to take additional filter-based samples to focus on the separate segments of your data. For example, if your dataset contains mismatched or missing values, you may consider taking an Anomaly-based sample that can look for mismatched, or missing, or both values in your dataset.
- As you add steps in your recipe, the current state of the Transformer page is rendered based on the currently valid sample (initial sample, in this case) plus all of the recipe steps between the step where the sample was taken and your current step. All of these steps must be rendered in the browser. As you add more recipe steps without taking a sample, browser performance is affected.

Tip: You should utilize sampling as much as possible to improve the browser performance and to get good coverage of the samples across recipes.

NOTE: Generation of a new sample is executed as a job. Quick scan jobs are executed through Trifacta Photon on the Trifacta node, while Full scan jobs are executed on an available clustered running environment. Depending on your deployment, there may be costs associated with generating a sample.

You can generate a new sample when:

- You are working with complex and wide datasets.
- You have complex flows.
- Your dataset has a bad data or outliers that may require a different sample.
- You have datasets with more than 10 MB of data.
- You have added one or more multi-dataset operations with steps, such as a join, union, pivot, or lookup.

Limitations

- Advanced sampling options are available only with a full scan of the dataset.
- Undo/redo do not change the sample state, even if the sample becomes invalid.
- Sort transformations are not preserved for some type of outputs when a new sample is generated. Sort transformations have to be reapplied.
- Samples taken from a dataset with parameters are limited to a maximum of 50 files when executed on the Trifacta Photon running environment. You can modify parameters as they apply to sampling jobs.

Collect a New Sample

You can use the existing loaded sample, or you can collect a new sample to use.

Steps:

1. In the Transformer page, click the Eyedropper icon at the top of the page.
2. From the Samples panel, select the required type of sample. For more information, see *Sample Types*.
3. In the Collect new sample panel, select either Quick or Full scan.
 - a. **Quick:** Creates a sample by partial scanning of the dataset and yields quicker results.

Tip: Quick scan samples are executed by default in the Trifacta Photon running environment. If that environment is not available, the Designer Cloud application may attempt to run the Quick Scan job on an available clustered running environment.

- b. **Full:** Creates a sample by scanning the full dataset. This method takes a longer time depending on the size of the dataset.

Tip: Full scan samples are executed in the cluster running environment.

4. Click **Collect** to collect the same. When the sample is available, a status message is displayed in the Transformer page.
5. Click **Load Sample** to start using the sample.

Cancel a Sample

To cancel a sample collection, click the X next to the progress bar. The interrupted sample is listed as unavailable in the Collected samples panel.

Example - Random sample

Random samples can be generated from a quick or full scan of your dataset.

Tip: A random sample is a fast way to get another randomized slice of your dataset. Often, this can be a first sample to generate after loading a new dataset into the Transformer page.

Steps:

1. In the Transformer page, click the Eyedropper icon at the top of the page.
2. From the Samples panel, select Filter-based sample.
3. In the Collect new sample panel, select the type of scan: Quick or Full.
4. Click **Collect**.
5. When sample collection is complete, a confirmation message is displayed. Click **Load sample**.
6. The random sample is loaded into the Transformer page.

Example - Filter-based sample

The Filter-based sample is helpful when you want to filter the data based on specific values or formulas. The following example filters the required values in the `Region` column for calculating discounts, and then generates a random sample from the matching rows only. For example, you may have a dataset with many values for `Region` such as Atlantic, North East, West Coast and want to calculate discounts only for North East region, you can collect a Filter-based sample.

Steps:

1. In the Transformer page, click the Eyedropper icon at the top of the page.
2. From the Samples panel, select Filter-based sample.
3. In the Collect new sample panel, enter the following details:
 - a. From the Scan column, select **Quick**. For more information, see "Collect a New Sample" above.
 - b. In the Filter field, enter `Region == 'North East'`.
4. Click **Collect**. A confirmation message is displayed.
5. Click **Load sample**. The Filter-based sample is loaded with only the `North East` values for the `Region` column.

Example - Anomaly-based sample

If your dataset has missing values or mismatched values, you can use Anomaly-based sample type to filter the missing values. The following example is based on the missing values in a `Discount` column. When you apply the Anomaly-based sample, the sample displays only rows that have missing values for the `Discount` column.

Steps:

1. In the Transformer page, click the Eyedropper icon at the top of the page.
2. From the Samples panel, select Anomaly-based sample.
3. In the Collect new sample panel, enter the following details:
 - a. From the Scan column, select **Quick**. For more information, see "Collect a New Sample" above.
 - b. Select the required column: `Discount`.
 - c. From the anomaly type, select **Find missing values only**.
4. Click **Collect**. A confirmation message is displayed.
5. Click **Load sample**. The Anomaly-based sample is loaded with the missing values for the `Discount` column.

Load a Sample

You can create as many samples as required based on your dataset. All collected samples are available in the Collected samples panels, where you can review and load them as required.

Steps:

1. In the Samples panel, click **See all collected samples**.
2. From the Collected samples panel, select the required sample from the Available tab. For more information, see "Collected Samples" below.

NOTE: Samples listed under the Unavailable tab are invalid for the current state of your recipe. You cannot select these samples for use.

3. If you want to edit the sample name, click the Pencil icon against the sample.

Delete a Sample

After you have created a sample, you cannot delete it through the application.

NOTE: Designer Cloud powered by Trifacta Enterprise Edition does not support deletion of samples after they have been created. For more information, contact your IT administrator.

Sample Types

For more information, see *Sample Types*.

Invalid Samples

NOTE: Samples are valid based on the state of your flow and recipe at the step where the sample was collected.

Whenever you add or modify a step to the recipe, Designer Cloud powered by Trifacta Enterprise Edition verifies if the current sample is valid. The current sample can become invalid if you add a new step before the step where the sample was created. For example, if you have created a sample in 30th step and if you add a new step that breaks the sample before the 30th step, then the sample becomes invalid.

After the sample becomes invalid, the Transformer page reverts to the recently collected sample that is valid.

NOTE: If the sample is reverted to an earlier sample, then more steps between when that sample was generated and your current location in the recipe are generated in the browser's memory. Browser performance may be impacted.

NOTE: If you modify a SQL statement for an imported dataset, any samples based on the old SQL statement are invalidated.

Collected Samples

The collected samples store the details of your samples collected for your dataset. In the Samples panel, click **See all collected samples** link. The collected samples contain the following tabs:

- **Available:** Displays the available samples that can be used.
- **Unavailable:** Displays the invalid samples, which cannot be selected for use. If subsequent steps make a sample valid again, it is moved to the **Available** tab.
- **All:** Displays both the available and unavailable samples.

Review Sample Jobs

You can review and manage all of your samples like transformation jobs.

Tip: As needed, you can cancel jobs in progress through the Samples panel or the Sample Jobs page.

For more information, see *Sample Jobs Page*.

Best Practices

For more information on best practices, troubleshooting, and browser crashes, see <https://community.trifacta.com/s/article/Best-Practices-Managing-Samples-in-Complex-Flows>.

Validation Tasks

You can detect issues in your data or validate it against source or target schemas.

Profile Your Source Data

You might want to execute a profile of the data that you imported from the source. As soon as you create a recipe from a source, you can execute a job to profile the dataset.

By profiling the data as soon as you load it into the Transformer page, you can assess the following:

- Identify problems in the source and potentially correct them in the source system.
- Create a baseline to evaluate the data wrangling work you do in Designer Cloud powered by Trifacta® Enterprise Edition.
- Identify mismatched or missing values.

Tip: You can also use this technique to generate an output of your source data, which is useful if you do not have read access to the source outside of Designer Cloud powered by Trifacta Enterprise Edition.

Steps:

1. Create an imported dataset from your source. Add it to a flow. See *Import Data Page*.
 - a. Depending on how your data is structured, you may choose to disable Detect Structure. For more information, see *Initial Parsing Steps*.
2. In Flow view, create a recipe for your imported dataset. See *Flow View Page*.
3. In Flow view, edit the newly created recipe. It is opened in the Transformer page. See *Transformer Page*.
4. If needed, add a header step to your dataset.
5. Click **Run**.
6. In the Run Job page, select the following options:
 - a. If you have the option of selecting a running environment, select the default one. This option may not be available in your product.
 - b. CSV format (you need at least one format to generate your dataset's profile).
 - c. Select to profile results.
7. Click **Run**.
8. When the results are generated, click **View Results**.
9. A profile of your dataset is displayed.

In the generated profile, you can identify:

- Missing or mismatched values in each column
- Statistical break-out by quartile
- Beginning dataset size and baseline job execution speed

Tip: You might want to write down the overall statistics for the dataset, which may be useful when validating the changes you have applied through recipe.

You might also download the dataset for recordkeeping. See *Job Details Page*.

Preserve Source Visual Profile

If you wish to preserve the capability of running a profile or gathering results from your source, you can do the following:

1. In Flow View, select the recipe that was used to create the source profile.
2. Rename this recipe to something like, *SourceData*.
3. Create an output off of this recipe. Run the job if you have not yet created the visual profile.
4. Select the recipe again. Now, click **Add New Recipe**.
5. Edit this new recipe and build out your transformation steps.

6. Whenever you need to regenerate the profile for the source, select the `SourceData` recipe and select the output from it. Then, run a job for it.

Tip: This technique is useful if you are replacing the source dataset with refreshed data on a periodic basis.

See *Flow View Page*.

Validate Your Data

Contents:

- *Before You Begin*
 - *Verify downstream requirements*
 - *Identify important fields*
 - *Profile your source data*
 - *Generate a new random sample*
 - *Transformations vs. Data Quality Rules*
 - *Validate Consistency*
 - *Mismatched values*
 - *Outlying values*
 - *Data range checks*
 - *Duplicate rows*
 - *Uniqueness checks*
 - *Permitted character checks*
 - *Validate Completeness*
 - *Missing values*
 - *Null values*
 - *Validate data against other data*
 - *After Transformation*
 - *Generate output profile*
 - *Decisions*
-

The process of cleansing, enhancing, and transforming your data can introduce significant changes to it, some of which might not be intended. This page provides some tips and techniques for validating your dataset, from start to finish for your data wrangling efforts.

Data validation can be broken down into the following categories:

- **Consistency** - Does your data fit into expected values for it? Do field values match the data type for the column? Are values within acceptable ranges? Are rows unique? Duplicated?
- **Completeness** - Are all expected values included in your data? Are some fields missing values? Are there expected values that are not present in the dataset?

Before You Begin

Before you begin building your data pipeline, you should identify your standards for data quality.

NOTE: Depending on your source system, you might be able to generate data quality reports from within it. These reports can be used as the basis for validating your work in Designer Cloud powered by Trifacta® Enterprise Edition.

If your source system does not enable generation of these reports, you should consider profiling your dataset as soon as you load your data into Designer Cloud powered by Trifacta Enterprise Edition.

Verify downstream requirements

Before you begin modifying your dataset, you should review the columns and ranges of values in those columns that are expected by the downstream consumer of your dataset. A quick review can provide guidance to identify the key areas of your dataset that require end-to-end validation.

Identify important fields

For datasets with many columns, it might be problematic to apply consistent validation across all columns. In these situations, you might need to decide the columns whose consistency, completeness, and accuracy are most important.

Profile your source data

Before you get started building your recipe on your dataset, it might be a good idea to create a visual profile of your source data. This process involves creating a minimal recipe on a dataset after you have loaded into the Transformer page. Then, you run a job to generate a profile of the data, which can be used as a baseline for validating the data and as an assistant in debugging the origin of any data problems you discover.

Visual profiling also generates statistics on the values in each column in the dataset. You can use this statistical information to assess overall data quality of the source data. This visual profile information is part of the record for the job, which remains in the system after execution.

For more information, see *Profile Your Source Data*.

Generate a new random sample

When a dataset is first loaded into the Transformer, the default sampling collects the first N rows of data, depending on the size and density of each row. However, your dataset might contain variations in the data that are not present in this first sample. For more information, see *Samples Panel*.

Transformations vs. Data Quality Rules

You can perform data quality rules through the following general methods:

1. **Transformations:** You can verify the quality of your data by creating transformations to check values for consistency and completeness and, if needed, taking action on the data itself for deviations.
 - a. Transformations are built in the Transform Builder in the Transformer page to add steps to your recipe. For more information, see *Transform Builder*.

Tip: If you need to take actions in the data itself based on data quality checks, it may be better to use a transformation.

2. **Data quality rules:** You can create data quality rules, which are persistent checks of columnar data against rules that you define. You can perform a variety of checks that exist outside of the recipe, so as you transform your data, the data quality rules automatically show the effects of your transformations on the overall quality of your data.
 - a. Data quality rules are not recipe steps. They exist outside of recipes and persist in the Transformer page to help you to build steps to transform your data.
 - b. Data quality rules are built in the Data Quality Rules panel in the Transformer page.
 - c. For more information, see *Overview of Data Quality*.

Tip: If you are attempting to transform the data to get all values in a column to pass one or more data quality checks, use data quality rules.

Examples of both types of data quality checks are provided below.

Validate Consistency

Designer Cloud powered by Trifacta Enterprise Edition provides useful features for checking that your data is consistent across its rows. With a few recipe steps, you can create custom validation checks to verify values.

Mismatched values

In the data quality bar at the top of a column, you can review the valid (green), mismatched (red), and missing (gray) values.

When you click the red bar:

- The rows that contain mismatched values are highlighted in the data grid.
- The application provides suggestions in the form of suggestion cards for ways that you can transform your data.

Transformation:

Maybe you are unsure of what to do with your data. If you would like to examine all of the rows together, you can insert a transformation like the following in your recipe.

Transformation Name	New formula
Parameter: Formula type	Single row formula
Parameter: Formula	<code>ismismatched(Primary_Website_or_URL, ['Url'])</code>
Parameter: New column name	<code>mismatched_Primary_Website_or_URL</code>

The above checks the values in the `Primary_Website_or_URL` column against the `Url` data type. If the value in the source column is not a valid URL, then the new column value is `true`. After sorting by this new column, all of the invalid URLs are displayed next to each other in the data grid, where you can review them in detail.

Data quality rule:

The following data quality rule checks the `Primary_Website_or_URL` column against the `Url` data type:

Data Quality Rule	Valid
Parameter: Column	<code>Primary_Website_or_URL</code>
Parameter: Data type	<code>'Url'</code>

Outlying values

Through the Column Details panel, you can review statistical information about individual columns. To open, select **Column Details...** from a column's drop-down menu.

In the Summary area, you can review the count of Outlier values. In Designer Cloud powered by Trifacta Enterprise Edition, an **outlier** is defined as any value that is more than 4 standard deviations from the mean for the set of column values.

The Column Details panel also contains:

- Counts of valid, unique, mismatched, and missing values.
- Breakdowns by quartile and information on maximum, minimum, and mean values.

For more information, see *Column Details Panel*.

Available statistics depend on the data type for the column. For more information, see *Locate Outliers*.

Data range checks

Standard deviation ranges

For example, your range of values does not match the application's definition of an outlier, and you need to identify values that are more than 5 standard deviations from the mean.

You can create your custom transforms to evaluate standard deviations from mean for a specific column. For more information, see *Locate Outliers*.

Fixed value ranges

Transformation:

If you need to test a column of values compared to two fixed values, you can use the following transformation. This one tests evaluates a column value. If the value in `Rating` column is less than 10 or greater than 90, then the generated column value is `true`.

Transformation Name	New formula
Parameter: Formula type	Single row formula
Parameter: Formula	((Rating < 10) (Rating > 90))
Parameter: New column name	Outlier_Rating

Data quality rule:

The following data quality rule performs the same evaluation as the previous transformation yet persists in the Transformer page.

Data Quality Rule	Formula
Parameter: Formula	((Rating < 10) (Rating > 90))
Parameter: Group rows by	(empty)

Duplicate rows

Entire rows can be tested for duplication. The `deduplicate` transform allows you to remove identical rows. Note that whitespace and case differences are evaluated as different rows. For more information, see *Deduplicate Data*.

Uniqueness checks

For an individual column, the column details panel contains an indicator of the number of unique values in the column. If this value does not match the count of values and the count of rows in the sample, then some values are duplicated. Remember that these counts apply to just the sample in the Transformer page and may not be consistent measures across the entire dataset. See *Column Details Panel*.

You can perform ad-hoc tests for uniqueness of individual values. For more information, see *Deduplicate Data*.

Data quality rule:

The following data quality rule verifies that all of the values in the `custId` column are unique:

Data Quality Rule	Unique
-------------------	--------

Parameter: Column	custId
-------------------	--------

Permitted character checks

You can test for the presence of permitted characters in individual columns by using a regular expression test.

Transformation:

The following transformation evaluates to `true` if all of the characters in a column field are alphanumeric or the space character:

Transformation Name	New formula
Parameter: Formula type	Single row formula
Parameter: Formula	<code>MATCHES(MarketName, /^[a-zA-Z0-9]*\$ /)</code>

You can add additional permitted characters inside the square brackets. For more information, see *Text Matching*.

Data quality rule:

This data quality rule performs the same test as the above transformation:

Data Quality Rule	Match
Parameter: Column	MarketName
Parameter: Matches pattern	<code>/^[a-zA-Z0-9]*\$ /</code>

Validate Completeness

Designer Cloud powered by Trifacta Enterprise Edition provides easy methods for identifying if cells are missing values or contain null values. You can also create lookups to identify if values are not represented in your dataset.

Missing values

At the top of each column, the data quality bar includes a gray bar indicating the number of cells in the column that do not contain values. This set of values includes missing values.

Click the gray bar to prompt for a set of suggestion cards for handling those values.

For more information, see *Find Missing Data*.

Null values

While null values are categorized with missing values, they are not the same thing. In some cases, it might be important to distinguish the actual null values within your dataset, and several Wrangle can assist in finding them. See *Manage Null Values*.

Validate data against other data

You can also test if your dataset contains at least one instance of a set of values.

For example, your dataset contains businesses throughout the United States. You might want to check to see if each state is represented in your dataset.

Steps:

1. Create a reference dataset that contains a single instance of each item you are checking. In this example, it'd be a simple CSV file with the name of each state on a separate line.

Tip: To your second dataset, you might want to add a second column containing the value `true`, which allows you to keep separate validation data from the columns that you join.

2. Add this CSV file as a new dataset to your flow.
3. Open your source dataset. In the Search panel, enter `join datasets`.
4. In the Join window:
 - a. Select the reference dataset you just created. Click **Accept**. Click **Next**.
 - b. Select the type of join to perform:
 - i. **Right outer join:** Select this join type if you want to delete rows in your source dataset that do not have a key value in the reference dataset. In the example, all rows that do not have a value in the State column would be removed from the generated dataset.
 - ii. **Full outer join:** Select this type to preserve all data, including the rows in the source that do not contain key values.
 - c. Select the two fields that you want to use to join. In the example, you would select the two fields that identify state values. Click **Next**.
 - d. Select the fields that you want to include in the final dataset. Click **Review**.
 - e. Click **Add to Recipe**.
5. The generated dataset includes all of the fields you specified.
6. For one of your key values, click the gray bar and select the link for the number of affected rows, which loads them into the data grid. Review the missing values in each key column.
7. To remove these rows, select the missing value category in the data quality bar for the appropriate column and apply a delete statement.
8. The generated command should look like the following:

Transformation Name	Delete rows
Parameter: Condition	ISMISSING([State])

For more information, see *Join Window*.

After Transformation

Generate output profile

After you have completed your recipe, you should generate a profile with your executed job. You can open this profile and the profile you created for the source data in separate browser tabs to evaluate how consistent and complete your data remains from beginning to end of the wrangling process.

NOTE: The statistical information in the generated profile should be compared to the statistics generated from the source, so that you can identify if your changes have introduced unwanted changes to these values.

Decisions

After you have performed your data validation checks, you might need to make some decisions about how to address any issues you might have encountered:

- Some problems in the data might have been generated in the source system. If you plan to use additional sources from this system, you should try to get these issues corrected in the source and, if necessary, have your source data regenerated.

- Some data quality issues can be ignored. For the sake of downstream consumers of the data, you might want to annotate your dataset with information about possible issues. Be sure to inform consumers on how to identify this information.

Find Bad Data

Contents:

- *Locate mismatched values*
- *Methods for fixing mismatched data*
 - *Mismatched values in transform code*
- *Trim data*
- *Set values using other columns*
- *Use functions to fix mismatched values*
- *Bad data typing*

You might encounter problems with how data has been structured or formatted that you must fix prior to providing the content to your target system. You can use the methods in this section to locate problems with the content or data typing of your data.

Locate mismatched values

When Designer Cloud powered by Trifacta® Enterprise Edition evaluates a dataset sample, it interprets the values in a column against its expectations for the values. Based on the column's specified data type and internal pattern matching, values are categorized as valid, mismatched, or missing. These value categories are represented in a slender bar at the top of each column.

- A **mismatched value** is any value that seems to be of a different data type than the type specified for the column. For example, if the value `San Francisco` appears in a column of Zip Code type, it would be marked as a mismatched value.

In the data quality bar, mismatched values are identified in red:

Tip: Before you start performing transformations on your data based on mismatched values, you should verify the data type for these columns to ensure that they are correct. The type against which values are checked is displayed to the upper left of the data quality bar. Below, the data type is `ZIP` for U.S. Zip code data. For more information, see *Supported Data Types*.



Figure: Mismatched values in red

Mismatched values can be sourced from a variety of issues:

- Values may be miskeyed into the source system.
- The source system may introduce errors in output, particularly if the data is generated for export using a customized structure.
- Incorrect use of column delimiters may create offsets among fields in individual rows.
- Data may be badly structured across a set of rows.
- The column may be assigned the wrong data type.

Tip: When cleaning up bad data, you should look to work from bigger problems to smaller problems. If a higher percentage of a column's values have been categorized as mismatched data, it may indicate a wider problem with the data. In affected rows, verify if other columns' values are also mismatched. These rows should be reviewed and fixed first. When fixed, other mismatches may be fixed in other rows, too.

To locate data:

NOTE: Remember that you are working on a sample of your data. For small datasets, the Initial Data sample includes all rows of the dataset and is unsampled.

- From the Transformer page, click the mismatched values in a column's data quality bar to see their count, highlight them in the rows of the data grid, and trigger a set of suggestions for your review.
- To refine the data grid view, click the Show Only Affected Rows checkbox in the status bar at the bottom of the screen. Only the rows that are affected by the previewed transform are displayed.

Tip: This step highlights specific values that are mismatched. You can take note of individual values.

- To locate a specific value, click the Filters icon on the right side of the screen. In the Rows tab, enter the specific value to locate. Rows containing this value are highlighted. Back in the data grid, you can select one of these highlighted values to be prompted for suggestions.

Methods for fixing mismatched data

When you discover mismatched data in your dataset, you have the following basic methods of fixing it:

1. **Change the data type.** If the percentage of mismatched rows is significant, you may need to change the data type for a better match.
2. **Replace the values with constant values.** This method works if it is clear to you that the values should be a single, consistent value. Select the mismatched values in the column, and then select one of the highlighted mismatched values. Use the `replace` transform to change the mismatched values to corrected values.

Tip: One easy way to fix isolated problems with mismatched values is to highlight a mismatched value in the data grid. A new set of suggestions is displayed. You can select the `replace` suggestion and then modify it to include the replacement value.

3. **Set values with other columns' values.** You can use the `set` transform to fix mismatched values by replacing them with the corresponding values from other columns.
4. **Use functions.** Data can be fixed by using a function in conjunction with the `set` transform to replace mismatched values.
5. **Delete rows.** Select the mismatched values and use the `delete` transform to remove the problematic rows.
6. **Hide the column for now.** You can remove the column from display if you want to focus on other things. Select **Hide** from the column drop-down. Note that hidden columns appear in any generated output.
7. **Delete the column.** If the column data is unnecessary or otherwise unusable, you can delete the entire column from your dataset. Select **Delete** from the column drop-down.

Tip: Delete unnecessary columns as early as possible. Less data is easier to work with in the application and improves job execution performance.

NOTE: You might need to review and fixed mismatched data problems multiple times in your dataset. For example, if you unnest the data, additional mismatches might be discovered. Similarly, joins and lookups can reveal mismatches in data typing.

Mismatched values in transform code

In your transforms, mismatched data can be identified references as in the following:

Transformation Name	Edit column with formula
Parameter: Columns	postal_code
Parameter: Formula	IF(ISMISMATCHED(postal_code, ['Zipcode']), '00000', postal_code)

Note that the single quotes are important around the value, which identifies the value as a constant.

Tip: In the above, note that the value `Zipcode` identifies the data type that is used for matching the column values. In this case, for greater specificity, you might want to identify the mismatched values in the column against the data type `Integer`, since all U.S. postal codes are positive integers. For more information on how to explicitly reference data types in your steps, see *Valid Data Type Strings*.

Trim data

To trim whitespace out of a column, use the following transformation:

Transformation Name	Edit column with formula
Parameter: Columns	column1
Parameter: Formula	TRIM(\$col)

The `$col` token is a reference to the column name to which the formula is being applied. For more information, see *Source Metadata References*.

This step may increase the number of missing values (for values that contain only whitespace characters) and the number of instances of matching values (for values that have spaces before and after an alphanumeric value).

You can modify the above transformation to trim leading and trailing spaces across all columns in your dataset. The wildcard (*) applies the formula to all columns in the dataset.

Transformation Name	Edit column with formula
Parameter: Columns	*
Parameter: Formula	TRIM(\$col)

You can extend the above transformation further by removing any leading or trailing single- and double-quote marks using the `TRIMQUOTES` function wrapped around the `TRIM` reference:

Tip: Keep in mind that nested functions are evaluated from the inside out. In this case, the `TRIM` function is evaluated first, which removes any surrounding whitespace. Then, the `TRIMQUOTES` function is applied.

Transformation Name	Edit column with formula
Parameter: Columns	*
Parameter: Formula	TRIMQUOTES(TRIM(\$col))

Set values using other columns

You can use values from other columns to replace mismatched values in your current column. Using the previous example, mismatched postal codes are replaced by the corresponding value in the parent entity's postal code column (`parent_postal_code`):

Transformation Name	Edit column with formula
Parameter: Columns	parent_postal_code
Parameter: Formula	IF(ISMISMATCHED(postal_code, ['Zipcode']), '00000', postal_code)

Use functions to fix mismatched values

In your transforms, you can insert a predefined function to replace mismatched data values. In the following example, the value for mismatched values in the `score` column are computed as the average of all values in the column:

Transformation Name	Edit column with formula
Parameter: Columns	score
Parameter: Formula	IF(ISMISMATCHED(score, ['Decimal']), AVERAGE(score), score)

Tip: You can also use the `IFMISMATCHED` function to test for mismatched values. Unlike the above construction, however, `IFMISMATCHED` does not support an else clause when the value does match the listed data type. For more information, see *IFMISMATCHED Function*.

Bad data typing

Tip: Particularly for dates, data is often easiest to manage as String data type. Designer Cloud powered by Trifacta Enterprise Edition has a number of functions that you can deploy to manage strings. After the data has been properly formatted, you can change it to the proper data type. If you change data type immediately, you may have some challenges in reformatting and augmenting it. Do this step last.

For columns that have a high percentage of mismatched values, the column's data type may have been mis-assigned. In the following example, a column containing data on precipitation in inches has been mis-typed as Boolean data:

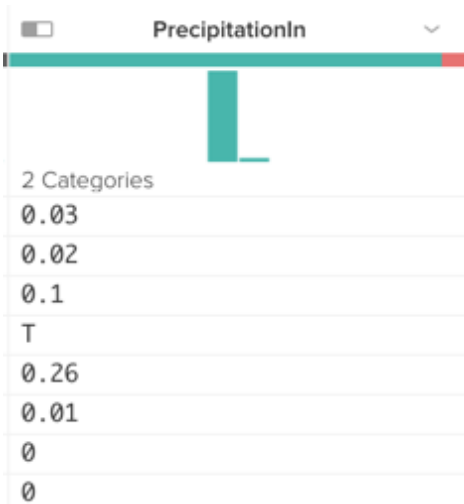


Figure: Mis-typed column data type

To change a column's data type, click the type identifier at the top of the column and select a new type. In this case, you would select `Decimal`.

NOTE: After you change the type, review the data quality bar again. If there are still mismatched values, review them to see if you can categorize the source of the mismatch.

As you can see in the previous example, the precipitation column contains values set to `T`, which may be short for `true`. When the data type is set to `Decimal`, these values now register as mismatched data. To fix, you can replace all `T` values with `1.0` using the `set` transform.

Select an instance of `T` in the column and click the `Set` suggestion card. Click **Modify**. For the `value` in the transform, enter `1.0`. Your transform should look like the following:

Transformation Name	Edit column with formula
Parameter: Columns	PrecipitationIn
Parameter: Formula	IFMATCHES([PrecipitationIn], `{start}{bool}{end}`), '1.0', PrecipitationIn)

Tip: If possible, you should review and refer to an available schema of your dataset, as generated from the source system. If the data has also been mis-typed in the source system, you should fix it there as well, so any future exports from that system show the correct type.

NOTE: As needed, you can create custom data types for use in Designer Cloud powered by Trifacta Enterprise Edition, which enable you to validate column data against a custom set of values. See *Create Custom Data Types*.

Find Missing Data

Contents:

- *Locate missing values*
- *Methods for fixing missing data*
- *Insert constants for missing values*
- *Copy values from another column*
- *Use functions to populate missing values*
- *Manage Missing Metadata*
 - *Example - Change Type*
 - *Example - Insert Year*
 - *Example - Insert Timezone*

When data is imported from another system, you might discover that some values are missing in it. In some cases, these values simply contain no content. In other cases, these values are non-existent. Depending on how the missing values entered the data, you may end up processing them in different ways. This section describes how to identify and manage missing data in your datasets.

NOTE: If you are unsure of the meaning of a column of data that contains missing values, you should attempt to review the source data or contact the individual who generated the data to identify why values may be missing and how to effectively manage them in Designer Cloud powered by Trifacta® Enterprise Edition and downstream systems.

Locate missing values

When your dataset sample is evaluated, each column is validated against the column's type definition. Based on that validation, values in the column are categorized as valid, mismatched, or missing. These values are categorized in the data quality bar at the top of each column.

- A **missing value** is any value that either contains no content or is non-existent.
 - An example of a non-existent value is a cell in a column of integers that has no value in it. In this special case, the missing value is called a **null value**.
 - Null values are converted to missing values during import. For more information, see *Manage Null Values*.
- Values that are spaces (one or more presses of the `SPACEBAR`) or tabs (one or more presses of the `TAB` key) are not missing values.

Tip: To trim whitespace out of a column, use the following transformation:

Transformation Name	Edit column with formula
Parameter: Columns	column1
Parameter: Formula	TRIM(column1)

This step may increase the number of missing values (for values that contain only whitespace characters) and the number of instances of matching values (for values that have spaces before and after an alphanumeric value).

- Return (`\n`) and newline (`\1`) are considered missing.

In the data quality bar, missing values are identified in gray:



Figure: Missing values in gray

Tip: From the Transformer page, click the missing values in a column to see their count, highlight them in the rows of the data grid, and trigger a set of suggestions for your review.

Missing values can be sourced from a variety of issues:

- Values may be miskeyed into the source system.
- The source system may enable optional fields that do not contain values. For example, U.S. zip codes can contain a second, four-digit qualifier for the base 5-digit zip code (an extended Zip+4 code). This second value may not be required and may therefore be missing.
- For columns of generated values, a computation may not be possible from the source data, which may indicate problems with other column data.
- A set of missing values within a row may indicate a problem with the entire record.
- The source system may introduce errors in output, particularly if the data is generated using a customized structure.

Tip: When cleaning up missing data, you should look to work from bigger problems to smaller problems. If a higher percentage of a column's values have been categorized as missing data, you should look across affected rows to see if it's a wider problem. If other records look ok, you should consider deleting the column or figuring out how to manage the missing values, including populating them.

Data may also be considered missing if you don't have sufficient information about the data. For example, timestamps that do not have a timezone identifier may not be usable in the target system.

Methods for fixing missing data

When you discover mismatched data in your dataset, you have the following basic methods of fixing it:

1. **Identify if the column values are required.**
 - a. Check the target system to determine if the field must have a value. If values are not required, don't worry about it. Consider deleting the column.
 - b. Remember that null values imported into Designer Cloud powered by Trifacta Enterprise Edition are exported as missing values, which are easier to consume in most systems.
 - c. Check the column header and data type to determine if values are required. For example, in transactional data, a field called `coupon_code` requires data only if every transaction is processed with one.
 - d. If it's available, check the source system to see if it requires entry into the field. If an entry is required and your data contains missing values, then there is an issue in how the data was exported from the source system.
2. **Insert a constant value.** You can replace a missing value with a constant, which may make it easier to locate more important issues in the application.
3. **Use a function.** Particularly if the missing data can be computed, you can use one of the available functions to populate the missing values.
4. **Copy values from another column.** If a value from another column or a modified form of it can be used for the missing value, you can use the `set` transform to overwrite the missing values.

5. **Delete rows.** Select the missing values bar and use the `delete` transform to remove the problematic rows.

NOTE: Since missing data may not be an explicit problem, you should avoid deleting rows or the column itself until other options have been reviewed.

6. **Hide the column for now.** You can remove the column from display if you want to focus on other things. Select **Hide** from the column drop-down. Note that hidden columns still appear in any generated output.
7. **Delete the column.** If the column data is unnecessary or otherwise unusable, you can delete the entire column from your dataset. Select **Delete** from the column drop-down.

Tip: Delete unnecessary columns as early as possible. Less data is easier to work with in the application and increases job execution performance.

Insert constants for missing values

NOTE: Generally speaking, inserting constants in place of missing values is not a recommended practice, especially if downstream consuming applications and individuals may not be known. In particular, you should not replace missing numeric values with a fixed numeric value, which will skew analysis. Use this method only if your entire data chain is aware of the constants.

Steps:

1. Click the gray missing values segment of the data quality bar for the column to fix.

Tip: Select a missing value in the data grid. Then, select the `replace` suggestion and then modify it to include the replacement value.

2. In the suggestion cards, click the `set` suggestion.
3. By default, this transform sets the missing value to be a null value. Click **Edit**.
4. You might see something like the following:

Transformation Name	Edit column with formula
Parameter: Columns	country
Parameter: Formula	IF(ISMISSING([country]),NULL(),country)

5. The missing data is identified using the `row: ISMISSING` reference. To apply a constant, replace the `NULL()` reference with a constant value, as in the following:

Transformation Name	Edit column with formula
Parameter: Columns	country
Parameter: Formula	IF(ISMISSING([country]),'USA',country)

Note that the single quotes around the value are required, since it identifies the value as a constant.

6. Click **Add**.

Tip: You can also use the `IFMISSING` function to test for empty values. Unlike the above construction, however, `IFMISSING` does not support an else clause when the value is present. For more information, see *IFMISSING Function*.

Copy values from another column

You can populate missing values with values from another column. In the following example, the `nickname` column is populated with the value of `first_name` if it is missing:

Transformation Name	Edit column with formula
Parameter: Columns	nickname
Parameter: Formula	<code>IF(ISMISSING([nickname]),first_name,nickname)</code>

Use functions to populate missing values

Particularly for numeric data, you can use functions to populate missing values. In the following example, missing values for the `unit_price` column are derived from a computation of the `weight_kg` column and the `price` column:

Tip: Be careful using functions such as averages to compute missing values. These computations may factor outliers that have not yet been removed or may fail to account for local trends relative to the data. Study the values and their meaning in the column before performing replacements. When in doubt, a median value may be your best best, assuming outliers and spurious data have been properly addressed.

Transformation Name	Edit column with formula
Parameter: Columns	unit_price
Parameter: Formula	<code>IF(ISMISSING([unit_price]),(price / weight_kg),unit_price)</code>

Manage Missing Metadata

In some cases, a column may contain valid values, but the meaning of those values is missing from the data. For example, your data contains the following Timestamp information:

Timestamp
19 May 02:45:38
19 May 02:42:24
19 May 02:41:33

This timestamp information may be considered problematic for the following reasons:

- The format may be incorrect for the target system.
- There is no year information. If the target system contains multi-year datasets, it may cause issues. The month element should be interpretable by Designer Cloud powered by Trifacta Enterprise Edition.
- There is no timezone information. In what timezone were these entries recorded?

The following examples demonstrate how to insert this information into your timestamps.

Example - Change Type

On import, timestamp data may be classified as String data. For now, this is ok.

Tip: Particularly for dates, data is often easiest to manage as String data type. Designer Cloud powered by Trifacta Enterprise Edition has a number of functions that you can deploy to manage strings. After the data has been properly formatted, you can change it to the proper data type. If you change data type immediately, you may have some challenges in reformatting and augmenting it. Do this step last.

After you have added back missing elements, you can change the data type to Date/Time through the data type drop-down for the column.

Before you begin reformatting your data, you should identify the target date format to which you want to match your timestamps. From the data type drop-down, select **Date/Time**. The dialog shows the following supported date formats:

Tip: When wrangling your data, you should start with the target structure or format of your data and work back to your source. This principle applies to both column management and overall dataset management.

Date / Time Type×

☐ mm

☐ yy

☐ mm-yy

☐ mm-dd

☐ dd-mm

☐ mm-dd-yy

☐ dd-mm-yy

☐ yy-mm-dd

☐ yy-dd-mm

☐ mm-dd-yy hh:mm:ss

☐ dd-mm hh:mm:ss

☐ mm-dd hh:mm:ss

☒ dd-mm-yy hh:mm:ss

☐ yy-mm-dd hh:mm:ss

☐ yy-dd-mm hh:mm:ss

☐ hh:mm:ss

dd*mm*yy*hh:MM:SS.sssa

⌵

Cancel

Save

Figure: Available Date/Time formats

NOTE: Each available option has a set of sub-options in the displayed drop-down.

In this timestamp example, the target format is the following:

```
dd-mm-yy hh:mm:ss (dd*shortMonth*yyyy*HH:MM:SS)
```

Example - Insert Year

The easiest way to handle the insertion of year information is to split out the timestamp data into separate components and then to merge back the content together with the inserted year information. Since the above timestamp data essentially contains three separate fields (Day of Month, Month, and Time), you can use a split command to break this information into three separate columns. Highlight one of the spaces between Day of Month and Month and select the `split` suggestion. The Wrangle step should look similar to the following:

Transformation Name	Split column on delimiter
Parameter: Column	column1
Parameter: Option	By delimiter
Parameter: Delimiter	' '
Parameter: Number of columns to create	2

Now, your data should be stored in three separate columns.

Tip: You may notice that new data types have been applied to the generated columns. The data may be easier to handle if all column types are converted to String type for now.

The next step involves merging all of these columns back into a single field, augmented with the appropriate year information. Select the columns in the order in which you would like to see them in the new timestamp field. In this case, you can select them in the order that they were originally listed. When all three columns are selected, choose the `merge` suggestion.

You may notice that the data has been formatted without spaces (19May02:45:38), and there is no year information yet. You can create new columns containing a year value (`myYear`) then merge the columns together:

Transformation Name	Merge columns
Parameter: Columns	column2, myYear, column3, column4
Parameter: Separator	' '

After you have inserted the year information and merged the columns, you should be able to change the column data type to the appropriate version of Date/Time.

Example - Insert Timezone

Timestamps do not natively support different timezones, so this information must be stored in a separate column. For U.S. data, timezones can be determined based on the zip code.

NOTE: If missing metadata is not supported as part of the value in the target system, you can insert the metadata as a separate column and then apply the metadata to the data inside the target system.

For more information on inserting timezone metadata, see *Add Lookup Data*.

Manage Null Values

Contents:

- *Important notes on null values*
- *Locate null values*
- *High percentage of nulls*
- *Fix null values*
- *Null values in transformations*
- *Write null values*

In general terms, a null value is a definition that points to nothing. A container for a value, such as a row-column combination or a variable, exists, but the container points to no actual value.

Important notes on null values

NOTE: In the platform, null values are a subset of the category identifying missing values. For technical reasons, however, Designer Cloud powered by Trifacta® Enterprise Edition displays null values as missing values and visually treats them as the same. Internally, they are understood to be different values.

Implications:

- Null values are visually represented as missing values.
 - In the data quality bar, null and missing values are represented in the dark bar (missing values).
- Computationally, they are different types of values.
 - Most functions applied to null and missing values return the same results.
 - For example, the `ISMISSING` function returns `true` for null and missing values.
 - However, the `ISNULL` function returns `true` for a null value and `false` for a missing value. See below.
 - If you use a function to generate null values, they are displayed as missing values, although they are recorded as nulls.
 - For example, the following transform generates a column of null values, which are represented as missing values in the data quality bar.

Transformation Name	New formula
Parameter: Formula	NULL()
Parameter: New column name	nulls

- When a set of results is generated, both null and missing values are written as missing values, unless the output format has a specific schema associated with it.

NOTE: When a recipe containing a user-defined function is applied to text data, any null characters cause records to be truncated by the running environment during Photon job execution. In these cases, please execute the job in the Spark running environment.

Locate null values

Null values are displayed with missing values in the Missing values category of the data quality bar (in gray).

You can use the following transform to distinguish between null and missing values. This transform generates a new column of values, which are set to `true` if the value in `isActive` is a null value:

Tip: You can use this transform and a subsequent sorting step on the generated column to filter for null values.

Transformation Name	New formula
Parameter: Formula	<code>ISNULL(isActive)</code>
Parameter: New column name	<code>nulls2</code>

High percentage of nulls

On import, if a column has a high enough percentage of null values, the platform may retype the column as a `String` column, which may yield mismatched values in addition to the missing values that were imported from null values.

Fix null values

See *Find Missing Data*.

Null values in transformations

Functions:

- Applying a null value as an input to a scalar function returns a null value, propagating the null value.
- In aggregate or window functions, null values are ignored, as a single null value could corrupt an entire column of calculations.

Transforms:

- In a join, a null value in one dataset never matches with a null value in another dataset. Rows with null values in join key columns are never included in the output. See *Join Types*.

Write null values

If needed, you can write a null value to a set of data. In the following example, all missing values in a column are replaced by nulls, using the `NULL` function.

NOTE: The `NULL` function is typically used to pass null values into functions that have been designed to specifically address them.

The following example tests all columns in the range between `column1` and `column255` for whether a missing value is detected. If so, a null value is written. Otherwise, the column value is written back to the column:

Transformation Name	Edit column with formula
Parameter: Columns	<code>column1~column255</code>
Parameter: Formula	<code>IF(ISMISSING([\$col]), null(), \$col)</code>

The above transform writes null values, but these values are converted to missing values on export.

Structuring Tasks

These tasks describe different methods for changing the shape of your data. Some of these tasks are applied on data import, while others can be managed through a single transformation in your recipe.

Tip: Some transformations may add or remove data, and the source data is lost. To retain the original data, you may choose to create chains or branching sets of recipes before you apply restructuring steps. For more information, see *Create Branching Outputs*.

Initial Parsing Steps

Contents:

- *File Encoding*
 - *Automatic Structure Detection*
 - *Overview*
 - *Splitting Columns*
 - *Header Row*
 - *Excel, CSV*
 - *JSON*
 - *Avro*
 - *Database Tables*
 - *Known Issues*
 - *Troubleshooting*
 - *Fixing parsing issues from structured source after recipe has been created*
-

When a dataset is initially loaded into the Transformer page, one or more steps may be automatically added to the new recipe in order to assist in parsing the data. The added steps are based on the type of data that is being loaded and the ability of the application to recognize the structure of the data.

File Encoding

When a text file is used as an imported dataset, Designer Cloud powered by Trifacta® Enterprise Edition assumes that the imported files are encoded in UTF-8, by default.

NOTE: Assessing the file encoding type based on parsing an input file is not an accurate method. Instead, Designer Cloud powered by Trifacta Enterprise Edition assumes that the file is encoded in the default encoding. If it is not, the Designer Cloud application should be prompted with the appropriate encoding type.

NOTE: In some cases, imported files are not properly parsed due to issues with encryption types or encryption keys in the source datastore. For more information, please contact your datastore administrator.

- As needed, you can change the encoding to use when parsing individual files. In the Import Data page, click **Edit Settings** in the right-hand panel. For more information, see *Import Data Page*.
- You can change the default encoding type applied to imported files. For more information, see *Configure Global File Encoding Type*.

Automatic Structure Detection

NOTE: By default, these steps do not appear in the recipe panel due to automatic structure detection. If you are having issues with the initial structuring of your dataset, you may choose to re-import the dataset with Detect structure disabled. Then, you can review this section to identify how to manually structure your data. For more information on changing the import settings for a dataset, see *Import Data Page*.

This section provides information on how to apply initial parsing steps to unstructured imported datasets. These steps should be applied through the recipe panel.

NOTE: Imported datasets whose schema has not been detected are labeled, **unstructured datasets**. These datasets are marked in the application. When a recipe for this dataset is first loaded into the Transformer page, the structuring steps are added as the first steps to the associated recipe, where they can be modified as needed.

Overview

When data is first loaded, it is initially contained in a single column, so the initial steps apply to `column1`.

Step 1: Split the rows. In most cases, the first step added to your recipe is a Splitrows transformation, which breaks up the individual rows based on a consistently recognized pattern at the end of each line. Often, this value is a carriage return or a carriage return-new line. These values are written in Wrangle as `\r` and `\r\n`, respectively. See the example below.

Step 2: Split the columns. Next, the application attempts to break up individual rows into columns.

- If the dataset contains no schema, the Split Column transformation used. This transformation attempts to find a single consistent pattern or a sequence of patterns in row data to demarcate the end of individual values (fields).

NOTE: Avoid creating datasets that are wider than 1000 columns. Performance can degrade significantly on very wide datasets.

- If the dataset contains a schema, that information is used to demarcate the columns in the dataset.

When the above steps have been successfully completed, the data can be displayed in tabular format in the data grid.

Step 3: Add column headers. If the first row of data contains a recognizable set of column names, a Rename Columns with Rows transformation might be applied, which turns the first row of values into the names of the columns.

Example recipe:

1.	Transformation Name	Split into rows
	Parameter: Column	column1
	Parameter: Split on	\r
	Parameter: Ignore matches between	\
	Parameter: Quote escape character	\
2.	Transformation Name	Split column
	Parameter: Column	column1
	Parameter: Option	on pattern
	Parameter: Match pattern	', '

Parameter: Number of matches	9
Parameter: Ignore matches between	\ "

3. Transformation Name	Add header
Parameter: Row number	1

After these steps are completed, the data type of each column is inferred from the data in the sample. See *Supported Data Types*.

Splitting Columns

When you import a dataset, the application can automatically split your column into separate columns based on one or more delimiters.

NOTE: Avoid importing datasets that are wider than 1000 columns. Particularly with previewing transformations in the data grid, very wide datasets can consume a significant amount of memory, which can cause browser crashes. Depending on your local environment, you may be able to work with these wide datasets. However, if the dataset is joined with other datasets or shared with other users, crashes can occur.

Tip: If you select the delimiter in a column with a very large number of delimiters, any suggestion card limits the split to a maximum of 250 columns. You can edit the suggested transformation to increase the number of split columns as needed. Increasing the limit can impact browser performance.

Header Row

When a dataset is imported, the application may infer the names of your columns from the first row of the dataset.

Tip: Avoid importing data that contains missing or empty values in the first row. These gaps can cause problems in your headers.

- In some cases, the application may be unable to create this header row. Instead, the columns are titled `column1`, `column2`, `column3` and so on.
- If the column names are split across multiple rows in your dataset, you may need to modify the header transformation step. For more information, see *Rename Columns*.

Excel, CSV

Microsoft Excel files are internally converted to CSV files and then loaded into the Transformer page. CSV files are treated using the general parsing steps. See previous section.

For more information, see *Import Excel Data*.

JSON

If 80% of the records in an imported dataset are valid JSON objects, then the data is parsed as JSON.

Notes:

- For JSON files, it is important to import them in unstructured format.
- Designer Cloud powered by Trifacta® Enterprise Edition requires that JSON files be submitted with one valid JSON object per line.
 - Multi-line JSON import is not supported.
 - Consistently malformed JSON objects or objects that overlap linebreaks might cause import to fail.
- Depending on the shape of your data, you may need to change the following properties. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.

`webapp.maxRecordLength` - determines the maximum length for an individual line.

`webapp.sampleLoadLimit` - determines the maximum size in bytes for a random sample.

NOTE: Be cautious in changing these values. If you set these values too high, you can overload the client and crash the application.

For more information, see *Working with JSON v2*.

Avro

Avro-based sources of data do not require any initial restructuring.

Database Tables

Properly formatted database tables with a provided schema should not require any initial parsing steps.

Known Issues

- Some characters in imported datasets, such as `NUL` (ASCII character 0) characters, may cause problems with recognizing line breaks. If initial parsing is having trouble with line breaks, you may need to fix the issue in the source data prior to import, since the Splitrows transformation must be the first step in your recipe.

Troubleshooting

Fixing parsing issues from structured source after recipe has been created

If you discover that your dataset has issues related to initial parsing of a structured source after you have started creating your recipe, you can use the following steps to attempt to rectify the problem.

Steps:

1. Open the flow containing your recipe.
2. Select the imported dataset. From the context menu, select **Remove structure....**
3. For the imported dataset, click **Add new recipe**.
4. Make any changes to the initial parsing steps in this recipe.
5. Select the recipe you were initially modifying. From its context menu, select the new recipe as its source.

The new initial parsing steps are now inserted into recipe flow before the recipe steps in development.

Reshaping Steps

Recipe steps that change the number of rows in the dataset have additional impacts on your dataset and its samples. These **reshaping steps** include the following transformations:

Transformation	Documentation
Splitrows	<i>Initial Parsing Steps</i>
Expand Arrays into Rows	<i>Working with Arrays</i>
Filter Rows (keep or delete)	<i>Remove Data</i>
Pivot Table	<i>Pivot Data</i>
Unpivot Columns	<i>Unpivot Columns</i>
Join Datasets	<i>Join Window</i>
Union Datasets	<i>Union Page</i>
Select Lookup from the column menu	<i>Lookup Wizard</i>
Remove Duplicate Rows	<i>Remove Data</i>

Samples:

When one of these transformations is applied and rows are removed from your dataset:

- Any samples generated before the step was added are invalidated and cannot be used.
- If you edit steps in your recipe before this added transformation, any samples that you generated after the step are invalidated and cannot be used.
- A valid initial sample is always available for use.

For more information, see *Samples Panel*.

Split Column

Contents:

- *Split by Delimiter*
 - *Split on single delimiter*
 - *Split column by multiple delimiters*
 - *Split column between delimiters*
- *Split by Position*
 - *Split column by positions*
 - *Split columns between positions*
 - *Split column at regular interval*
- *Encoding Issues*
- *Splitting Rows*

For many recipes, the first step is to split data from a single column into multiple columns. This section describes the various methods that can be used for splitting a single column into one or more columns, based on character- or pattern-matching or position within the column's values.

Split by Delimiter

When data is initially imported into Designer Cloud powered by Trifacta® Enterprise Edition, data in each row may be split on a single delimiter. In the following example, you can see that the tab key is a single clear delimiter:

<IMSI^MSIDN^IMEI>	DATETIME/TIMEZONE	OFFSET/DURATION	MSWCNT:BASCNT^BASTRA	CALL_TYPE
/CORRESP_IDN/DISCONNECT REASON				
<310170097665881^13011330554^011808005351311>	2014-12-12T00:06:13/-5/1.55			MSC001:
BSC002^BTS783	MOT/00000000000:11			
<310170097665881^13011330554^011808005351311>	2014-12-12T02:27:26/-5/0.00			MSC001:
BSC002^BTS783	SMS/00000000000:			
<310-170-097665881^13011330554^011808005351311>	2014-12-12T03:24:20/-5/0			MSC001:
BSC001^BTS783	SMS/00000000000:			

However, when this data is imported, it may be rendered in the data grid in the following structure:

column2	column3	column4	column5
<IMSI^MSIDN^IMEI>	DATETIME/TIMEZONE OFFSET/DURATION	MSWCNT: BASCNT^BASTRA	CALL_TYPE/CORRESP_IDN: DISCONNECT REASON
<310170097665881^13011330554^011808005351311>	2014-12-12T00:06:13/-5/1.55	MSC001: BSC002^BTS783	MOT/00000000000:11
<310170097665881^13011330554^011808005351311>	2014-12-12T02:27:26/-5/0.00	MSC001: BSC002^BTS783	SMS/00000000000:
<310-170-097665881^13011330554^011808005351311>	2014-12-12T03:24:20/-5/0	MSC001: BSC001^BTS783	SMS/00000000000:

Notes:

- When the data is first imported, all of it is contained in a single column named column1. The application automatically splits the columns on the tab character for you and removes the original column1.

Tip: This auto-split does not appear in your recipe by default. For most formats, a set of initial steps is automatically applied to the dataset. Optionally, you can review and modify these steps, but you must deselect Detect Structure during the import. See *Initial Parsing Steps*.

- Because the application was unable to determine clear headers for each column's data, generic ones are used. So, before you apply a header to your data, you must split out the data within each column.
- The delimiters within each column vary.
 - column2 uses the caret, while column3 uses the forward slash.
 - column4 and column5 use multiple delimiters.
- There is sparseness in the data. Note that in column5, the second row contains the value 11 at the end, while the other two data rows do not have this value.

Split on single delimiter

For column2, you can split the column into separate columns based on the caret delimiter:

Transformation Name	Split by delimiter
Parameter: Column	column2
Parameter: Option	By delimiter
Parameter: Delimiter	' ^ '
Parameter: Number of columns to create	2

NOTE: The Number of columns to create value reflects the total number of new columns to generate.

Results:

Below is how the data in column2 is transformed:

column1	column6	column7
<IMSI	MSIDN	IMEI>
<310170097665881	13011330554	011808005351311>
<310170097665881	13011330554	011808005351311>
<310-170-097665881	13011330554	011808005351311>

- Since column1 was unused as a name, it re-appears here. column6 and column7 are the next available generic column names.
- There is a small bit of cleanup to do in column1 and column7 to remove the symbols at the beginning and end of these column values. You can do this cleanup before the split in the original column2 if desired.

For column3, suppose that you want to keep the DATETIME and TIMEZONE OFFSET values in the same column, preserving the forward slash to demarcate these two values. The DURATION values are to be split into a separate column:

Transformation Name	Split by delimiter
---------------------	--------------------

Parameter: Column	column2
Parameter: Option	By delimiter
Parameter: Delimiter	' / '
Parameter: Start to split after	`/(-{digit} {digit})`

- The above uses **Patterns** , which are simplified versions of regular expressions for matching patterns.
 - In this case, the expression is the following:

```
`/(-{digit}|{digit})`
```

- For the **Start to split after** value, the above indicates that the application should start to look for matches on the delimiter (forward slash) only after the above pattern has been detected in the column values.
 - In this case, the pattern describes values that appear after a forward slash and could be a negative digit or a positive digit, which matches the pattern for the **TIMEZONE OFFSET** values in the column.
 - For more information on how to use **Patterns** , see *Text Matching*.
- Since you are splitting the column into two columns, you do not need to specify the number of new columns to create. The default is 1.

Split column by multiple delimiters

After splitting column3, the data resembled the following:

column3
DATETIME/TIMEZONE OFFSET
2014-12-12T00:06:13/-5
2014-12-12T02:27:26/-5
2014-12-12T03:24:20/-5

Suppose you want to break down the components of this date-time data into separate columns for year, month, day, hour, minute, second, and offset. The following could be use to do so:

Transformation Name	Split by delimiter
Parameter: Column	column2
Parameter: Option	By multiple delimiters
Parameter: Delimiter 1	' - '
Parameter: Delimiter 2	' - '
Parameter: Delimiter 3	' T '
Parameter: Delimiter 4	' : '
Parameter: Delimiter 5	' : '
Parameter: Delimiter 6	' / '

- Each delimiter is entered on a separate row.
- Delimiters are processed in the listed order.

Split column between delimiters

Suppose that for column4, you want to split the column such that the middle part section is removed. You could use the previous transformation and then delete the middle column. You can also use the following transformation, which identifies that starting and ending delimiters that demarcate the separator between fields, effectively removing the middle column:

Transformation Name	Split by delimiter
Parameter: Column	column4
Parameter: Option	By two delimiters
Parameter: Start delimiter	' : '
Parameter: Include as part of split	Selected
Parameter: End delimiter	' ^ '
Parameter: Include as part of split	Selected

- The separator between the columns is all of the content between the forward slashes. This content is removed from the dataset.
- The two selected options include the forward slashes as part of the separator, which removes them from the dataset.

Split by Position

You can also perform column splits based on numerical positions in column values. These splitting options are useful for highly regular data that is of consistent length.

Suppose you have the following coordination information in three dimensions (x, y, and z). Note that the data is very regular, with leading zeroes for values that are less than 1000.

column1
POSXPOSYPOSZ
000100040001
012405210555
100220046554
202056789011
379274329832

Split column by positions

The above data could be split based on positions within a column's value:

Transformation Name	Split by character position
Parameter: Column	column1
Parameter: Option	By positions

Parameter: Position 1	4
Parameter: Position 2	8

Results:

column2	column3	column4
POSX	POSY	POSZ
0001	0004	0001
0124	0521	0555
1002	2004	6554
2020	5678	9011
3792	7432	9832

Split columns between positions

Suppose that you wish to split the above source data such that the middle column is removed:

Transformation Name	Split by character position
Parameter: Column	column1
Parameter: Option	Between two positions
Parameter: Position 1	4
Parameter: Position 2	8

Results:

column2	column3
POSX	POSZ
0001	0001
0124	0555
1002	6554
2020	9011
3792	9832

Split column at regular interval

The above transformation could be simplified even further, since the splits happen at regular intervals:

Transformation Name	Split by character position
Parameter: Column	column1
Parameter: Option	At regular interval
Parameter: Interval	4
Parameter: Number of times to split	2

Results:

The results would be the same as the first example.

Encoding Issues

If you are attempting to split columns based on non-ASCII characters that appear in the dataset, your transformations may fail.

In these cases, you should change the encoding that is applied to the dataset.

Steps:

1. In the Import Data page, select the dataset to import.
2. When the dataset card appears in the right column, click the Edit Settings link.
3. From the drop-down, select a more appropriate encoding to apply to the file.
4. Import the data and wrangle.
5. Try your split transformation on the dataset.

NOTE: Administrators can change the encoding that is applied by default to all file-based imported to the application. See *Configure Global File Encoding Type*.

Splitting Rows

When a dataset is imported, the application attempts to split the data into individual rows, based on any available end of line delimiters. This transformation is performed automatically and is not included in your initial set of steps.

If the data is not consistently formatted, the rows may not be properly split. If so, you can disable the automatic splitting of rows.

Steps:

1. In the Import Data page, select the dataset to import.
2. When the dataset card appears in the right column, click the Edit Settings link.
3. Deselect the Detect Structure checkbox.
4. Import the data and wrangle.

The steps used to detect structure are listed as the first steps of your recipe, which allows you to modify them as needed. For more information, see *Initial Parsing Steps*.

See *Import Data Page*.

Move Columns

Contents:

- *Cut and Paste Columns*
 - *Move using Column Menus*
 - *Move using Column Icons*
 - *Move using Transform Builder*
 - *Move multiple columns*
 - *Move range of columns*
 - *Move set of columns*
 - *Move using RapidTarget*
-

You can move or reorder individual columns or multiple columns by using the following options:

Cut and Paste Columns

To move an individual column or multiple columns, perform the following:

Steps:

1. Select an individual column or select multiple columns. For example, select Column B and select **Cut** from the column menu.
2. Navigate to the location where you want to paste the column then select **Paste > (Paste before or Paste after)** from the column menu.

In the following example, you can see what happens when Column B is moved after Column D.

Source:

Column A	Column B	Column C	Column D
Cell A.1	Cell B.1	Cell C.1	Cell D.1
Cell A. 2	Cell B.2	Cell C.2	Cell D.2

Results:

Column A	Column C	Column D	Column B
Cell A.1	Cell C.1	Cell D.1	Cell B.1
Cell A. 2	Cell C.2	Cell D.2	Cell B.2

Move using Column Menus

You can use the **Move** option from the drop-down caret of the column context menu to move an individual column or multiple columns.

To move an individual column or multiple columns, perform the following:

Steps:

1. To select an individual column, click its column header. To select multiple columns:
 - a. You can **SHIFT**-click a range of columns.

- b. To select multiple discrete columns, press **CTRL/COMMAND** + click.
2. Select **Move** from the column menu of one of the selected columns. Choose one of the following options to move a column:
 - **to beginning**: Moves the column to the beginning of the dataset.
 - **to end**: Moves the column to the end of the dataset.
 - **after/before**: Moves the column either before or after the specified columns of the dataset.

The specified transformation is displayed in the Transform Builder. For example, the following transformation moves Column A just after Column C:

The Column(s) option defines the method by which you specify the set of columns. In this case, **Multiple** simply means that you specify each column one after another in the transformation. To add this step to your recipe, click **Add**. The columns are moved.

Tip: You can use suggestion cards to select the appropriate transformation to move the columns. For more information on suggestions, see *Explore Suggestions*.

Move using Column Icons

You can use the Column View icon in the Transformer bar to move columns. For more information, see *Column Browser Panel*.

To move an individual column or multiple columns, perform the following:

Steps:

1. When you select an individual column or multiple columns, you are prompted with a set of suggestions.
2. Select the appropriate suggestion from the suggestion cards.
3. **Edit** or **Add** the steps, as required to move columns. For more information, see below examples.

Move using Transform Builder

In the Transform Builder, you can select one or more columns to move using finer-grained controls.

To move an individual column or multiple columns, perform the following:

Steps:

1. Enter **Move columns** in the search context panel. For more information on search panel, see *Search Panel*.
2. Select an individual column or multiple columns, as required. The following options are available when specifying one or more columns in a transformation:
 - **Multiple**: Select one or more columns from the drop-down list. See below example.
 - **Range**: Specify a start column and ending column. All columns inclusive are selected. See below example.
 - **Advanced**: Specify the columns using a comma-separated list. You can combine multiple and range options under Advanced. Ranges of columns can be specified using the tilde (~) character. See below example.
3. Select the required option from the **Option** drop-down list.
4. Select the required column to move after or before the column.
5. Click **Add**. The selected columns are moved based on your inputs.

Move multiple columns

This example moves two discrete columns (Column A and ColumnC), before Column E. These columns are not next to each other, so they can be specified using the Multiple column(s) option.

Source:

Column A	Column B	Column C	Column D	Column E
Cell A.1	Cell B.1	Cell C.1	Cell D.1	Cell E.1
Cell A. 2	Cell B.2	Cell C.2	Cell D.2	Cell E. 2

Transformation:

Transformation Name	Move Columns
Parameter: Column(s)	Multiple
Parameter: Column	A, C
Parameter: Option	Before
Parameter: Column	E

Results:

Column B	Column D	Column A	Column C	Column E
Cell B.1	Cell D.1	Cell A.1	Cell C.1	Cell E.1
Cell B.2	Cell D. 2	Cell A.2	Cell C.2	Cell E.2

Move range of columns

You can move a range of columns to a specified location. For example, you can move Column A through Column C after Column D.

Source:

Column A	Column B	Column C	Column D
Cell A.1	Cell B.1	Cell C.1	Cell D.1
Cell A. 2	Cell B.2	Cell C.2	Cell D.2

Transformation:

Transformation Name	Move Columns
Parameter: Column(s)	Range
Parameter: Column	A~C
Parameter: Option	After
Parameter: Column	D

Results:

Column D	Column A	Column B	Column C
Cell D.1	Cell A.1	Cell B.1	Cell C.1

Cell D. 2	Cell A.2	Cell B.2	Cell C.2
-----------	----------	----------	----------

Move set of columns

Using the Advanced option, you can move combinations of column ranges and discrete columns to a new location. In the following example, ColumnA through ColumnC and ColumnE are moved after ColumnF:

Source:

Column A	Column B	Column C	Column D	Column E	Column F
Cell A.1	Cell B.1	Cell C.1	Cell D.1	Cell E.1	Cell F.1
Cell A. 2	Cell B.2	Cell C.2	Cell D.2	Cell E.2	Cell F.2

Transformation:

In the transformation, you select the Advanced column(s) option where you can specify columns on a single line.

Tip: The tilde character (~) can be used to specify the range of columns between two listed columns. Ranges and individual columns should be separated by a comma.

ColumnA~ColumnC,ColumnE

Transformation Name	Move Columns
Parameter: Column(s)	Advanced
Parameter: Column	A~C, E
Parameter: Option	After
Parameter: Column	F

Results:

Column D	Column F	Column A	Column B	Column C	Column E
Cell D.1	Column F.1	Cell A.1	Cell B.1	Cell C.1	Cell E.1
Cell D.2	Column F.2	Cell A.2	Cell B.2	Cell C.2	Cell E.2

For more information, see *Column Reference Syntax*.

Move using RapidTarget

RapidTarget allows you to associate a target set of columns with your recipe. When you specified a target, you can often reposition your source columns with the targets by clicking in the interface.

- For more information, see *Overview of RapidTarget*.
- For more information, see *Create Target*.

Delete Data

Contents:

- *Delete Columns*
 - *By selection*
 - *Through transformation*
 - *Delete Rows*
 - *By selection*
 - *By custom conditions*
-

A key task in cleaning up your data is to remove unwanted columns and rows, which can simplify future transformations and improve job execution performance. Designer Cloud powered by Trifacta® Enterprise Edition provides multiple mechanisms for removing data from your dataset.

Tip: When you are deleting data, you should consider if that data may have other uses in the future or for other users. If so, you should consider doing the data removal through a separate recipe off of your current recipe, which preserves the data for other uses in the current recipe.

Delete Columns

You can delete one or more columns based on the following:

- By selection
- Through transformation

Tip: When you delete through transformation steps, you have additional controls at your disposal.

By selection

You can delete a single column or multiple columns:

- To delete a column from your dataset, click the column and select **Delete** from the column drop-down.
- If you select **Delete others**, all other remaining columns are deleted except the selected column.

Tip: To delete multiple columns, select them in the data grid or column browser. Then select **Delete** from the column menu.

The column or columns are removed from the data grid, and a new step is added to your recipe.

Through transformation

You can delete columns through the transformation steps.

Steps:

1. In the Transformer page, click **Delete columns**.
2. The Delete columns transformation is populated in the Transformer Builder.
3. Select one or more columns, as required:
 - a. **Multiple:** Select one or more columns from the drop-down list.
 - b. **All:** Select all columns in the dataset.

NOTE: This removes all columns in your dataset.

- c. **Range:** Specify a start and ending columns. All columns inclusive of start and end are deleted.
- d. **Advanced:** Specify the columns using a comma-separated list. Ranges of columns can be specified using the tilde (~) character. Examples:

Entry	Description
Store_Nbr ~ Daily	Columns from Store_Nbr to Daily in the dataset are deleted.
Store_Name,Store_Manager,Store_Nbr ~ Daily	The following columns are deleted: Store_Name Store_Manager Store_Nbr to Daily

- 4. From the **Action** area, select one of the following options:
 - a. **Delete selected columns:** Deletes only the selected columns.
 - b. **Delete unselected columns:** Deletes all other remaining columns except the selected columns.
- 5. To delete columns, click **Add**.

Example transformation:

The following transformation deletes the columns between Store_Nbr and Daily, inclusive.

Transformation Name	Delete columns
Parameter: Columns	Advanced
Parameter: Column	Store_Nbr~Daily
Parameter: Action	Delete selected columns

Delete Rows

Since rows do not have an identifying header, you must identify the rows to remove in your dataset based on a specified condition. You can delete rows based on the following:

- By selection
- By custom conditions

By selection

You can delete rows by selecting values. You are prompted for data filtering suggestions when you select values in:

- column histograms
- column data quality bars
- cells or values within a cell

When you make a selection, select the Delete rows transformation in the context panel. The Transform Builder contains a transformation to filter rows based on the the condition that you have selected. For example, if you selected the value California in the State column, then the transformation is specified to filter out rows in which State=California.

In the Transform Builder, you must decide if the transformation keeps matching rows (deleting all others) or deletes matching rows. In the following example, rows in which State=California are selected for deletion:

Transformation Name	Filter rows
Parameter: Condition	Custom formula

Parameter: Type of formula	Custom single
Parameter: Condition	State == "California"
Parameter: Action	Delete matching rows

By custom conditions

You can delete a set of rows based on a condition specified in the `condition` column . If the conditional expression is `true` , then the selected rows are deleted.

1. In the Transformer page, click the **Recipe** icon. The Recipe panel is displayed.
2. In the Search Transformations panel, enter `Filter` in.
3. In the Filter rows transformation, enter the required details:
 - a. **Condition:** Filter based on the condition type that you select in the drop-down. Some condition types do not support specifying the condition by formula.
 - b. **Column:** The column containing the values to filter. For example, `action_count`.
 - c. **Values or Formula:** Specify the values or the formula used to determine the condition.
 - i. If these values are present, then the condition evaluates to `true`.
 - ii. The formula must evaluate to `true` or `false`.
 - d. **Action:** The action to be performed to the rows based on the specified conditions.
 - e. In the following example, the rows where the `action_count` column values fall between 1 and 10 are deleted:

Transformation Name	Filter rows
Parameter: Condition	Custom formula
Parameter: Type of formula	Custom single
Parameter: Condition	<code>(action_count >= 1) && (action_count <= 10)</code>
Parameter: Action	Delete matching rows

Tip: You can apply logical operators such as `&&` (logical AND) above to build more sophisticated logical tests.

4. To add the recipe to the step, click **Add**. The dataset rows are filtered based on the configured transformation.

Select

You can completely replace the columns in your dataset by selecting source columns, functions computed from the source, and constant values.

NOTE: This transformation completely replaces the existing table, which could have significant effects on any downstream recipes or reference datasets that already exist.

Create Your Table

Steps:

1. In the Transformer page, open the Recipe panel.
2. In the recipe, locate the step where you wish to insert the transformation to create your new table.

NOTE: If your Create Table transformation renames or omits columns, references to them later in your recipe or in other downstream objects may be broken.

3. In the search bar, enter `Select`. Choose the transformation.
4. In the Transform Builder, you can create the columns in order for your new table. For each column:
 - a. In the upper field, enter the source of the column. The source can be one of the following:
 - i. A column name in your source
 - ii. A function. Example:

```
POW(myBaseVal, 5)
```

NOTE: When creating a table, aggregate and window functions are not supported. After you have created your table, you can apply these functions are normal.

- iii. A constant value. Example:

```
&apos;valid&apos;
```

- b. In the lower field, you enter a name for the column in the new table.
5. To add a new column, click **Add**. Repeat the previous steps.
 - a. You can remove columns, if needed. Click **Remove** next to the column entry.
 6. To create the new table when you've specified your columns, click **Add**.

The new table replaces your previous set of columns.

Tip: After you have created your new table, you can disable or delete the step to revert to the previous state.

Use RapidTarget

This transformation is added to your recipe when you perform column matching between your source dataset and a target schema. The results of your column matching work are rendered as a single Create Table transformation in your recipe.

Tip: If you have a target schema to which you can assign to your recipe, you may find it easier to create your new table using RapidTarget, which provides a visual interface for performing these remappings.

NOTE: RapidTarget does not support inserting columns containing constants or generated by functions. You can insert those column as a later step.

For more information, see *Overview of RapidTarget*.

Create Aggregations

Contents:

- *Limitations*
 - *Example Data*
 - *Aggregating across all rows (no grouping)*
 - *Aggregate grouped-by rows*
 - *Generate new aggregation table*
-

You can apply aggregate functions to groups of values in one or more columns to generate aggregated data. Depending on how you configure the Group By transformation, the output of these transformations is a new table or one or more columns in the current dataset.

Limitations

- The Group By transformation does not support nested expressions. You cannot insert multiple nested expressions in your computed value.
- The Group By transformation supports aggregation functions only. For more information, see *Aggregate Functions*.

Example Data

The following table contains test score data from a set of students for four separate tests, spread over two days:

Student	TestDate	TestNum	TestScore
Anna	09/08/2018	1	84
Ben	09/08/2018	1	71
Caleb	09/08/2018	1	76
Danielle	09/08/2018	1	87
Anna	09/08/2018	2	92
Ben	09/08/2018	2	86
Caleb	09/08/2018	2	99
Danielle	09/08/2018	2	73
Anna	09/15/2018	3	86
Ben	09/15/2018	3	99
Caleb	09/15/2018	3	86
Danielle	09/15/2018	3	80
Anna	09/15/2018	4	85
Ben	09/15/2018	4	87
Caleb	09/15/2018	4	79
Danielle	09/15/2018	4	93

Aggregating across all rows (no grouping)

You can perform basic computations across all rows of the dataset. For example, the following transformation creates a new column containing the average test score for all students:

Transformation Name	New formula
Parameter: Formula type	Single row formula
Parameter: Formula	<code>ROUND(AVERAGE(Score), 2)</code>
Parameter: New column name	avg_TestScore

The above results in a new column called, `average_TestScore`, containing the single value 85.19, which is the average of all students' test scores rounded to two decimal places.

NOTE: These types of aggregations are known as **flat aggregations**. In larger datasets, performing flat aggregations can be computationally intensive. Be careful in computing any aggregation functions across a large number of rows.

Aggregate grouped-by rows

For the above example data, suppose you are interested in the average score for each student. In this case, you must compute the average (`AVERAGE(TestScore)`) for each student.

In the previous transformation, you used the New Formula transformation. When you are computing aggregations across groups of values in a column, you must use the Group By transformation:

Transformation Name	Group By
Parameter: Group By	Student
Parameter: Values	<code>AVERAGE(TestScore)</code>
Parameter: Type	Group by as new column(s)

Note that the above transformation does not contain the rounding function. Nested expressions are not supported in the Group By transformation. To round the values, add the following transformation as the next step:

Transformation Name	Edit column with formula
Parameter: Columns	average_TestScore
Parameter: Formula	<code>ROUND(average_TestScore, 2)</code>

You may wish to rename the newly generated column to something like `average_TestScorePerStudent` instead. See *Rename Columns*.

The output data should look like the following:

Student	TestDate	TestNum	TestScore	average_TestScorePerStudent	average_TestScore
Anna	09/08/2018	1	84	86.75	85.19
Ben	09/08/2018	1	71	85.75	85.19
Caleb	09/08/2018	1	76	85	85.19

Danielle	09/08/2018	1	87	83.25	85.19
Anna	09/08/2018	2	92	86.75	85.19
Ben	09/08/2018	2	86	85.75	85.19
Caleb	09/08/2018	2	99	85	85.19
Danielle	09/08/2018	2	73	83.25	85.19
Anna	09/15/2018	3	86	86.75	85.19
Ben	09/15/2018	3	99	85.75	85.19
Caleb	09/15/2018	3	86	85	85.19
Danielle	09/15/2018	3	80	83.25	85.19
Anna	09/15/2018	4	85	86.75	85.19
Ben	09/15/2018	4	87	85.75	85.19
Caleb	09/15/2018	4	79	85	85.19
Danielle	09/15/2018	4	93	83.25	85.19

Generate new aggregation table

Suppose you wish to calculate the minimum, maximum, and average scores for each test. In this case, it may be more useful to create a new table in which the student names have been removed:

Transformation Name	Group By
Parameter: Group By	TestNum
Parameter: Values1	MAX(TestScore)
Parameter: Values2	MIN(TestScore)
Parameter: Values3	AVERAGE(TestScore)
Parameter: Type	Group by as new table

The resulting data looks like the following:

TestNum	max_TestScore	min_TestScore	average_TestScore
1	87	71	79.5
2	99	73	87.5
3	99	80	87.75
4	93	79	86

Tip: In this case, when you replace the existing table with a completely new table, data that is not included in the aggregation is lost. You can add columns to the list of values if you wish to bring forward untouched columns into the new table. You may also consider building aggregation tables in a recipe that is extended from the previous recipe, so that you can continue to work with the other columns in your dataset.

Pivot Data

Contents:

- *Building a Pivot Table*
- *Available Aggregations*
- *Simple Pivot Table*
- *Conditional Aggregations*
- *Multiple Aggregation Levels*
- *Group By*
- *Values to Columns*

A **pivot table** summarizes data that is sourced from another table. Using pivot tables, you can calculate aggregating functions, such as sums, maximums, and averages for one or more columns of data. Optionally, these sums can be performed across groups of values from one column and broken out in columns based on the values in another. In Designer Cloud powered by Trifacta® Enterprise Edition, a pivot table is composed of the following basic elements:

Pivot table element	Description
Column labels	List of one or more columns whose values are represented as the columns in the generated pivot table.
Row labels	List of one or more columns whose values become the rows in the generated pivot table.
Values	Also known as facts , these values are one or more aggregation formulas, which are calculated in the following manner: <i>"Show me the value of this formula computed by each row value for every value represented in the generated table."</i>

NOTE: If your aggregation does not include the kind of transformation listed above, in which the data is pivoted from rows into columns, you can use the Group By transformation. See *Create Aggregations*.

Building a Pivot Table

Pivot tables are very powerful tools for summarizing and visualizing large-scale volumes of data. In Designer Cloud powered by Trifacta Enterprise Edition, search for `pivot table` in the Search panel to create one.

NOTE: A pivot table completely replaces the source table. Data that is not captured in the pivot definition is lost.

In your flows, you may find it useful to create your pivot tables in independent recipes that are chained from your primary recipe. For more information, see *Create Branching Outputs*.

Example Data

Pivot tables are perhaps best explained by example. The following table snippet captures transactional data from a number of stores for a range of products across a set of dates. Transactional values include total sales, quantity, and cost (`POS_Sales`, `POS_Qty`, and `POS_Cost`):

Daily	Store_Nbr	POS_Sales	POS_Qty	POS_Cost	PRODUCT_DESC
2/8/13	1	70	7	4.97	ACME LAWN GARDEN BAG CLEAR

2/7/13	2	10.62	9	8.37	ACME COOKIES CHOC CHIP
2/7/13	2	0	0	0	ACME SANDWICH BAG
2/7/13	2	7.08	6	5.58	ACME SODAS SALTED
2/7/13	2	3.92	2	2.82	ACME SCENTED OIL REFILL-CTRY SUN
2/7/13	2	13.44	7	10.36	ACME LARGE FUDGE GRAHAMS COOKIES
2/7/13	2	0	0	0	ACME SUGAR ICE WAFERS VANILLA
2/7/13	3	3.16	2	2.86	ACME ZOO ANIMAL FRUIT SNACKS 6'S
2/7/13	3	3.16	2	2.78	ACME WAFERS SUGER ICE
2/7/13	3	3.16	2	2.82	ACME SCENTED OIL REFILL-CTRY SUN
2/7/13	3	6.32	4	5.92	ACME RICE CRACKERS ONION
2/2/13	9	150	30	16.2	ACME FROSTED OATMEAL COOKIE SQUA
2/2/13	9	3.5	2	4.86	ACME FRUIT SNACK CASTLE ADVENTRS
2/2/13	9	90	9	8.37	ACME COOKIES CHOC CHIP
2/2/13	9	30	6	3.24	ACME ASSORTED COOKIES DRP
2/2/13	9	70	7	6.51	ACME KITCHEN BAG
2/2/13	9	170	17	15.81	ACME SNACK BAGS RESEALABLE
2/2/13	9	20	4	2.16	ACME CHEDDARY SN CRACKERS/PROCES
2/2/13	9	6.5	2	8.98	ACME RICE CRACKERS TERIYAKI
2/2/13	9	1.5	3	1.62	ACME COOKIE MAPLE LEAF CREME
2/2/13	9	30	6	3.24	ACME RICE CHIPS CHEDDAR
2/1/13	7	190	38	20.52	ACME FROSTED OATMEAL COOKIE SQUA
2/1/13	7	20	2	1.86	ACME COOKIES CHOC CHIP
2/1/13	7	10	1	0.82	ACME DIGESTIVE RICH TEA BISCUITS
2/1/13	7	120	24	12.96	ACME ASSORTED COOKIES DRP
2/1/13	7	120	12	11.16	ACME KITCHEN BAG
2/1/13	7	90	9	8.37	ACME SNACK BAGS RESEALABLE
2/1/13	7	10	1	0.71	ACME FUDGE MINT COOKIES SQUARES
2/1/13	7	9.5	19	10.26	ACME CHEDDARY SN CRACKERS/PROCES
2/1/13	7	10	1	0.82	ACME COOKIES MAPLE CREAM
2/1/13	7	40	8	4.32	ACME COOKIE MAPLE LEAF CREME

Available Aggregations

The Pivot data transformation supports use of any aggregation function. For more information, see *Aggregate Functions*.

Simple Pivot Table

From the above, suppose you are interested in the sales from each store for each product. You can use the following transformation to compute these aggregated calculations:

Transformation Name	Pivot table
Parameter: Column labels	Store_Nbr

Parameter: Row labels	PRODUCT_DESC
Parameter: Values	SUM(POS_Sales)
Parameter: Max number of columns to create	500

In the above transformation:

- The Column labels entry specifies the column whose values make up the calculated columns of the pivot table. The calculation is performed *across each* of these values. In this case, each column contains calculations for separate store numbers.
- The Row labels entry specifies the column whose values define the grouping of the calculations. In this case, the sum of the sales column is performed for each product description value for each store.
- The Values entry specifies the aggregation function to compute for each cell in the new table. In this case, you are generating the sum of sales for each product description in each store.
- By default, this transformation generates a maximum of 50 new columns. However, if the column used for your Column labels contains more than 50 values, you may want to raise this value.

NOTE: Avoid creating datasets wider than 2500 columns. Very wide datasets can cause performance degradation.

Results:

PRODUCT_DESC	sum_POS_Sales_1	sum_POS_Sales_2	sum_POS_Sales_3	sum_POS_Sales_7	sum_POS_Sales_9
ACME LAWN GARDEN BAG CLEAR	70	0	0	0	0
ACME COOKIES CHOC CHIP	0	10.62	0	20	90
ACME SANDWICH BAG	0	0	0	0	0
ACME SODAS SALTED	0	7.08	0	0	0
ACME SCENTED OIL REFILL-CTRY SUN	0	3.92	3.16	0	0
ACME LARGE FUDGE GRAHAMS COOKIES	0	13.44	0	0	0
ACME SUGAR ICE WAFERS VANILLA	0	0	0	0	0
ACME ZOO ANIMAL FRUIT SNACKS 6'S	0	0	3.16	0	0
ACME WAFERS SUGER ICE	0	0	3.16	0	0
ACME RICE CRACKERS ONION	0	0	6.32	0	0
ACME FROSTED OATMEAL COOKIE SQUA	0	0	0	190	150
ACME FRUIT SNACK CASTLE ADVENTRS	0	0	0	0	3.5
ACME ASSORTED	0	0	0	120	30

COOKIES DRP					
ACME KITCHEN BAG	0	0	0	120	70
ACME SNACK BAGS RESEALABLE	0	0	0	90	170
ACME CHEDDARY SN CRACKERS /PROCES	0	0	0	9.5	20
ACME RICE CRACKERS TERIYAKI	0	0	0	0	6.5
ACME COOKIE MAPLE LEAF CREME	0	0	0	40	1.5
ACME RICE CHIPS CHEDDAR	0	0	0	0	30
ACME DIGESTIVE RICH TEA BISCUITS	0	0	0	10	0
ACME FUDGE MINT COOKIES SQUARES	0	0	0	10	0
ACME COOKIES MAPLE CREAM	0	0	0	10	0

Conditional Aggregations

Suppose you are interested in only in the sum of sales for store numbers 1-3. To capture a more limited dataset, you can use the `SUMIF` aggregation function:

Transformation Name	Pivot table
Parameter: Row labels	PRODUCT_DESC
Parameter: Values	<code>SUMIF(POS_Sales, Store_Nbr<4)</code>
Parameter: Max number of columns to create	500

Most aggregation functions have a conditional (`*IF`) variant. See *Aggregate Functions*.

Multiple Aggregation Levels

None of the axes of a pivot table is limited to a single dimension. You can have multiple Column labels, Row labels, and Values (formulas). In the following transformation, aggregations have been further broken out by date, and an additional formula (Value) has been added.

NOTE: Adding multiple Column labels and Values can greatly expand the width of the dataset. Generally, adding Row labels does not expand the total count of rows.

Transformation Name	Pivot table
Parameter: Column labels	Store_Nbr

Parameter: Row labels1	Date
Parameter: Row labels2	PRODUCT_DESC
Parameter: Values1	SUM(POS_Qty)
Parameter: Values2	SUM(POS_Sales)
Parameter: Max number of columns to create	500

Results:

NOTE: Following results table is incomplete. Some columns have been omitted for space reasons.

Daily	PRODUCT_DESC	sum_POS_Qty_1	sum_POS_Sales_1	sum_POS_Qty_2	sum_POS_Sales_2	sum_POS_Qty
2/8/13	ACME LAWN GARDEN BAG CLEAR	7	70	0	0	0
2/7/13	ACME COOKIES CHOC CHIP	0	0	9	10.62	0
2/7/13	ACME SANDWICH BAG	0	0	0	0	0
2/7/13	ACME SODAS SALTED	0	0	6	7.08	0
2/7/13	ACME SCENTED OIL REFILL-CTRY SUN	0	0	2	3.92	2
2/7/13	ACME LARGE FUDGE GRAHAMS COOKIES	0	0	7	13.44	0
2/7/13	ACME SUGAR ICE WAFERS VANILLA	0	0	0	0	0
2/7/13	ACME ZOO ANIMAL FRUIT SNACKS 6'S	0	0	0	0	2
2/7/13	ACME WAFERS SUGER ICE	0	0	0	0	2
2/7/13	ACME RICE CRACKERS ONION	0	0	0	0	4
2/2/13	ACME FROSTED OATMEAL COOKIE SQUA	0	0	0	0	0
2/2/13	ACME FRUIT SNACK CASTLE ADVENTRS	0	0	0	0	0
2/2/13	ACME COOKIES CHOC CHIP	0	0	0	0	0
2/2/13	ACME ASSORTED COOKIES DRP	0	0	0	0	0
2/2/13	ACME KITCHEN BAG	0	0	0	0	0
2/2/13	ACME SNACK BAGS RESEALABLE	0	0	0	0	0
2/2/13	ACME CHEDDARY	0	0	0	0	0

	SN CRACKERS /PROCES					
2/2/13	ACME RICE CRACKERS TERIYAKI	0	0	0	0	0
2/2/13	ACME COOKIE MAPLE LEAF CREME	0	0	0	0	0
2/2/13	ACME RICE CHIPS CHEDDAR	0	0	0	0	0
2/1/13	ACME FROSTED OATMEAL COOKIE SQUA	0	0	0	0	0
2/1/13	ACME COOKIES CHOC CHIP	0	0	0	0	0
2/1/13	ACME DIGESTIVE RICH TEA BISCUITS	0	0	0	0	0
2/1/13	ACME ASSORTED COOKIES DRP	0	0	0	0	0
2/1/13	ACME KITCHEN BAG	0	0	0	0	0
2/1/13	ACME SNACK BAGS RESEALABLE	0	0	0	0	0
2/1/13	ACME FUDGE MINT COOKIES SQUARES	0	0	0	0	0
2/1/13	ACME CHEDDARY SN CRACKERS /PROCES	0	0	0	0	0
2/1/13	ACME COOKIES MAPLE CREAM	0	0	0	0	0
2/1/13	ACME COOKIE MAPLE LEAF CREME	0	0	0	0	0

Group By

If you wish to maintain the original dataset values, you can perform aggregation calculations within a single column. For more information, see *Create Aggregations*.

Values to Columns

Similar to pivot, the Convert values to columns transformation converts individual values within a column to independent columns in the dataset. For each row, if the value represented by the column is present in the original data, one value is added (e.g. Yes). If it's missing, another value is inserted (e.g. No).

Tip: This type of conversion can be useful for preparing data for machine learning systems. You can convert the presence or absence of specific values in a row to 1 or 0, respectively.

In the following, the values in the `Store_Nbr` column have been converted to individual columns:

Transformation Name	Convert values to columns
Parameter: Column	Store_Nbr

Parameter: Fill when present	Yes
Parameter: Max number of columns to create	250

In the above:

- Fill when present identifies the string literal value to insert if the row contains the column's value (yes).
- Fill when missing identifies the string literal value to insert if the row does not contain the column's value (empty).
- Max number of columns to create places a limit on the total number of columns that the application is permitted to create. In this case, the limit is set to 250 since the known number of stores is 250.

Tip: It's a good habit to set limits on the maximum number of columns to create. Data can become sparse or unwieldy if limits are not considered.

Results:

Daily	Store_Nbr	POS_Sales	POS_Qty	POS_Cost	PRODUCT_DESC	column_1	column_2	column_3	colu
2/8/13	1	70	7	4.97	ACME LAWN GARDEN BAG CLEAR	Yes			
2/7/13	2	10.62	9	8.37	ACME COOKIES CHOC CHIP		Yes		
2/7/13	2	0	0	0	ACME SANDWICH BAG		Yes		
2/7/13	2	7.08	6	5.58	ACME SODAS SALTED		Yes		
2/7/13	2	3.92	2	2.82	ACME SCENTED OIL REFILL-CTRY SUN		Yes		
2/7/13	2	13.44	7	10.36	ACME LARGE FUDGE GRAHAMS COOKIES		Yes		
2/7/13	2	0	0	0	ACME SUGAR ICE WAFERS VANILLA		Yes		
2/7/13	3	3.16	2	2.86	ACME ZOO ANIMAL FRUIT SNACKS 6'S			Yes	
2/7/13	3	3.16	2	2.78	ACME WAFERS SUGER ICE			Yes	
2/7/13	3	3.16	2	2.82	ACME SCENTED OIL REFILL-CTRY SUN			Yes	
2/7/13	3	6.32	4	5.92	ACME RICE CRACKERS ONION			Yes	
2/2/13	9	150	30	16.2	ACME FROSTED OATMEAL COOKIE SQUA				Yes
2/2/13	9	3.5	2	4.86	ACME FRUIT SNACK CASTLE ADVENTRS				Yes
2/2/13	9	90	9	8.37	ACME COOKIES CHOC CHIP				Yes
2/2/13	9	30	6	3.24	ACME ASSORTED COOKIES DRP				Yes

2/2/13	9	70	7	6.51	ACME KITCHEN BAG				Yes
2/2/13	9	170	17	15.81	ACME SNACK BAGS RESEALABLE				Yes
2/2/13	9	20	4	2.16	ACME CHEDDARY SN CRACKERS /PROCES				Yes
2/2/13	9	6.5	2	8.98	ACME RICE CRACKERS TERIYAKI				Yes
2/2/13	9	1.5	3	1.62	ACME COOKIE MAPLE LEAF CREME				Yes
2/2/13	9	30	6	3.24	ACME RICE CHIPS CHEDDAR				Yes
2/1/13	7	190	38	20.52	ACME FROSTED OATMEAL COOKIE SQUA				
2/1/13	7	20	2	1.86	ACME COOKIES CHOC CHIP				
2/1/13	7	10	1	0.82	ACME DIGESTIVE RICH TEA BISCUITS				
2/1/13	7	120	24	12.96	ACME ASSORTED COOKIES DRP				
2/1/13	7	120	12	11.16	ACME KITCHEN BAG				
2/1/13	7	90	9	8.37	ACME SNACK BAGS RESEALABLE				
2/1/13	7	10	1	0.71	ACME FUDGE MINT COOKIES SQUARES				
2/1/13	7	9.5	19	10.26	ACME CHEDDARY SN CRACKERS /PROCES				
2/1/13	7	10	1	0.82	ACME COOKIES MAPLE CREAM				
2/1/13	7	40	8	4.32	ACME COOKIE MAPLE LEAF CREME				

Unpivot Columns

Contents:

- *Single-column Unpivot*
- *Multi-column Unpivot*
 - *Ranges*
 - *Wildcards*

You can convert columns into rows of values. The Convert transformation extracts the values from a specified column or columns and turns the column name and each extracted value into key-value pairs.

- Unpivot can be applied to one or more columns.
- Often, this transformation is applied to datasets containing pivoted or aggregated data. For more information, see *Pivot Data*.

NOTE: Depending on the number of source columns, an unpivot operation can significantly increase the number of rows in your dataset.

Single-column Unpivot

When you unpivot a single column of data, the column is separated into two new columns in your dataset:

New column name	Values
key	All values are the name of the source column.
value	Each row contains one of the row values from the source column.

NOTE: These columns replace the source column in the dataset. To retain the source column, create a copy of it first and then unpivot the copied column.

Source:

The following example contains a very simple set of data:

Name	favoriteColor	favoriteDessert
Anna	red	ice cream
Bella	pink	cookies
Callie	blue	pie

Transformation:

You can unpivot these columns one-by-one into row data:

Transformation Name	Unpivot columns
Parameter: Columns	favoriteColor

Parameter: Group size	1
-----------------------	---

Results:

The new unpivoted columns are placed at the end of the dataset, and the source column is removed.

Name	favoriteDessert	key	value
Anna	ice cream	favoriteColor	red
Bella	cookies	favoriteColor	pink
Callie	pie	favoriteColor	blue

Multi-column Unpivot

This example turns the data from multiple columns into a single set of key-value pairs, where the key is the column name associated with the source of the data in the value column.

Source:

The following dataset shows student test scores per test. Each row represents the scores of individual students.

StudentId	test1Score	test2Score	test3Score
001	75	79	77
002	84	81	86
003	79	82	87
004	92	94	92

Transformation:

You can use the following transformation to turn the dataset into one row per student-test combination:

Transformation Name	Unpivot columns
Parameter: Columns	test1Score, test2Score, test3Score
Parameter: Group size	1

Results:

The results are as follows:

StudentId	key	value
001	test1Score	75
002	test2Score	79
003	test3Score	77
001	test1Score	84
002	test2Score	81
003	test3Score	86
001	test1Score	79
002	test2Score	82

003	test3Score	87
001	test1Score	92
002	test2Score	94
003	test3Score	92

You can then rename the `key` and `value` columns as needed. See *Rename Columns*.

Ranges

You can specify a range of columns in your dataset. In the previous example, you can specify the three test score columns using the following value in the Columns textbox:

test1Score~test3Score

All three columns are unpivoted.

Wildcards

NOTE: You can use the asterisk (*) wildcard in the Columns textbox to apply the unpivot to the entire dataset, which generates a `key` and a `value` column, containing all column-row entries from the source columns. However, unpivoting a large number of columns can significantly increase the number of rows in your dataset.

Window Transformations

Contents:

- *Basic Structure*
 - *Group by parameter*
 - *Order by parameter*
 - *Compute over Time Windows*
 - *Calculate over preceding and following rows*
 - *Fill Empty Values*
 - *Calculate Rank*
 - *Calculate Rolling Functions*
 - *Rolling date functions*
-

A **window** transformation performs calculations on a row based on row values that are related to it. Windowing functions can perform calculations based on time, relative row positions, and rolling windows.

For example, you might wish to calculate the average percentage of CPU usage over 24-hour intervals based on log entries. From the rows of data, you can create a window function that calculates the average value in the CPU usage column over the 24-hour period, as defined based on date values for each log entry.

Key distinction:

- In a window function, the output of each row's calculation is specific to the row.
- In an aggregate function, the output for a row is the same value for all rows that are used in the calculation.
- For more information on aggregation, see *Create Aggregations*.

Basic Structure

You can use windowing functions with the following transformation types:

- window - creates a new column called `window`
- New formula - creates a new column that you name
- Edit with formula - modifies the values in a column based on a formula that you specify.

Group by parameter

You can use the Group by parameter to define the column of values by which rows of data are grouped for calculation purposes. For example, if your Group by column contains months, your calculations are computed for each month represented in the column values.

NOTE: Transforms that use the `group` parameter can result in non-deterministic re-ordering in the data grid. However, you should apply the `group` parameter, particularly on larger datasets, or your job may run out of memory and fail. To enforce row ordering, you can use the `sort` transform. For more information, see *Sort Transform*.

Order by parameter

When using window functions, you can use the Order by parameter to specify the column or columns by which to sort the output.

Source:

The following table contains the sales data of a company for all the four regions in the last three months.

Month	Sales	Region
2021-01-01	800	East
2021-01-01	1500	West
2021-01-01	1000	North
2021-01-01	2000	South
2021-02-01	1250	East
2021-02-01	800	West
2021-02-01	1100	North
2021-02-01	700	South
2021-03-01	900	East
2021-03-01	1000	West
2021-03-01	1400	North
2021-03-01	800	South

Transformation:

In the following transformation, you can calculate the rolling average of sales . You apply the `ROLLINGAVERAGE` and specify that the results are to be ordered by the Sales column.

Transformation Name	Window
Parameter: Formulas	<code>ROLLINGAVERAGE (Sales, 0,1)</code>
Parameter: Order by	Sales

Results:

The following dataset shows the `ROLLINGAVERAGE` ordered by Sales column.

Month	Sales	Region	RollingAverage
2021-02-01	700	South	750
2021-01-01	800	East	800
2021-02-01	800	West	800
2021-03-01	800	South	850
2021-03-01	900	East	950
2021-01-01	1000	North	1000
2021-03-01	1000	West	1050
2021-02-01	1100	North	1175
2021-02-01	1250	East	1325
2021-03-01	1400	North	1450
2021-01-01	1500	West	1750
2021-01-01	2000	South	2000

Compute over Time Windows

You may need to create windows of time within your data that are not cleanly segmented by basic units of time measurement. For example, you may need to create a custom time period, called a **session**, based on timestamps recorded in event-based data.

A session is usually defined as a group of events that occur within a given time frame. For example, you may need to perform calculations based on five-minute intervals within your logging data. If a user opens your shopping website, logs in, searches items, and then logs out within a five-minute interval, that can be grouped under a single session. However, if the user's interaction lasted six minutes, the logged events may span multiple windowed sessions in the data.

You can use the `SESSION` function to create time boxes based on a time period that you specify. When the function is applied to your column of timestamp values, the application assigns an ID to events that belong to the same session.

From the following example, you can create a Session ID. After you create the session ID, you can find the volume of data consumed by the individual user.

Source:

User Name	TimeStamp	Activity	Volume (in Kb)
Bob	02/11/21 08:01:13	Read	1024
William	02/11/21 08:01:00	Read	1024
John	02/11/21 08:01:17	Read	1024
Christy	02/11/21 08:01:17	Read	1024
William	02/11/21 08:03:33	Read	520
Christy	02/11/21 08:02:01	Password change	1024
Bob	02/11/21 08:07:23	Adding items to cart	2048
William	02/11/21 08:05:45	Read	520
William	02/11/21 08:11:56	Account settings	2048
John	02/11/21 08:15:11	Password change	2048
Bob	02/11/21 08:34:00	Proceeding to payment	2048
Bob	02/11/21 08:43:03	logout	2048
Christy	02/11/21 09:03:43	Read	1024
Christy	02/11/21 09:10:00	logout	1024

Transformation:

Transformation Name	Window
Parameter: Formulas	<code>SESSION (TimeStamp, 5, minute)</code>
Parameter: Group by	User Name
Parameter: Order by	TimeStamp

Since the new column is named `window`, you should rename it:

Transformation Name	Rename columns
Parameter: Option	Manual rename

Parameter: Column	window
Parameter: New column name	SESSIONID

With this session ID, you can calculate the maximum volume of data consumed by each session ID and by each user.

Transformation Name	New formula
Parameter: Formula type	Multiple row formula
Parameter: Formula	MAX(Volume (in Kb))
Parameter: Sort rows by	SessionID
Parameter: Group rows by	User Name, SessionID
Parameter: New column name	Volume_Consumed (in Kb)

Results:

User Name	TimeStamp	Activity	Volume (in Kb)	SessionID	max_Volume (in Kb)
William	02/11/21 08:01:00	Read	1024	1	1024
William	02/11/21 08:03:33	Read	520	1	1024
William	02/11/21 08:05:45	Read	520	1	1024
William	02/11/21 08:11:56	Account settings	2048	2	2048
Bob	02/11/21 08:01:13	Read	1024	1	1024
Bob	02/11/21 08:07:23	Adding items to cart	2048	2	2048
Bob	02/11/21 08:34:00	Proceeding to payment	2048	3	2048
Bob	02/11/21 08:43:03	logout	2048	4	2048
Christy	02/11/21 08:01:17	Read	1024	1	1024
Christy	02/11/21 08:02:01	Password change	1024	1	1024
Christy	02/11/21 09:03:43	Read	1024	2	1024
Christy	02/11/21 09:10:00	logout	1024	3	1024
John	02/11/21 08:01:17	Read	1024	1	1024
John	02/11/21 08:15:11	Password change	2048	2	2048

Calculate over preceding and following rows

The `PREV` and `NEXT` functions enable you to fetch data from a previous row or a subsequent row, which is helpful for identifying relative changes or trends in your data.

Source:

The following dataset contains orders for different product types over a given time period. You can apply the `PREV` and `NEXT` functions to calculate the previous orders and the next orders to analyze the trend of orders and derive the average of orders for a product group.

--	--	--

Product_Type	Order_date	Order
Laptop	2021-01-05	300
Laptop	2021-01-26	1780
Laptop	2021-01-09	500
Laptop	2021-01-31	1200
SmartPhone	2021-01-24	1400
SmartPhone	2021-01-26	2200
SmartPhone	2021-01-07	700
Tablet	2021-01-21	600
Tablet	2021-01-23	900

Transformation:

You can also calculate the percentage of change in orders over time. The following transformation calculates the change between the current order and the previous one and then divides that value over the previous value to calculate the percent change between the rows:

Transformation Name	Window
Parameter: Formulas	$(\text{Order} - \text{PREV}(\text{Order}, 1)) / \text{PREV}(\text{Order}, 1) * 100$
Parameter: Group by	Product_Type
Parameter: Order by	Order

After you rename the column to ChangeinOrder, you can apply the `NUMFORMAT` function to clean up and format the ChangeinOrder values. The following transformation reformats the ChangeinOrder column to display two decimal places.

Transformation Name	Edit with formula
Parameter: Column	ChangeinOrder
Parameter: Formula	<code>NUMFORMAT(ChangeinOrder, '##.##')</code>

Similarly, you can apply the `NEXT` function and calculate the Change in orders for upcoming months.

Results :

Product_Type	Order_date	Order	NEXTOrder	ChangeinOrder
Laptop	2021-01-05	300	500	
Laptop	2021-01-09	500	1200	66.67
Laptop	2021-01-31	1200	1780	140
Laptop	2021-01-26	1780		48.33
SmartPhone	2021-01-07	700	400	
SmartPhone	2021-01-24	1400	2200	100
SmartPhone	2021-01-26	2200		57.14
Tablet	2021-01-21	600	900	
Tablet	2021-01-23	900		50

See:

- *PREV Function*
- *NEXT Function*
- *NUMFORMAT Function*

Fill Empty Values

You can use the `FILL` function to fill empty or null values in your data with the last non-empty value in the group.

Source:

For example, the following dataset contains the daily orders received. Note the missing values due to weekends. You can assume that the no orders were received for Saturday and Sunday ,

Date	DayOfWeek	OrdersDay	OrdersTotal
2021-03-10	Wednesday	100	100
2021-03-11	Thursday	112	212
2021-03-12	Friday	320	532
2021-03-13	Saturday		
2021-03-14	Sunday		
2021-03-15	Monday	300	832

Transformation:

You have to clean up the data to fill the values for OrdersDay column. You can use the following function to fill the empty and null values. This function tests the the OrdersDay column to check if the column is empty or null. If so, the value ' 0 ' is written in the column, else the value of the column (`$col`) is written.

Transformation Name	Edit with formula
Parameter: Column	OrdersDay
Parameter: Formula	<code>IF(OrdersDay == '' ISNULL(OrdersDay), '0', \$col)</code>

You can see the values of Friday is taken for Saturday and Sunday and filled it accordingly as per the `FILL` function.

Transformation Name	Edit with formula
Parameter: Column	OrdersTotal
Parameter: Formula	<code>IF (OrdersDay == '0', FILL (OrdersTotal, -1,0),\$col)</code>
Parameter: Order by	Date

Results:

Date	DayOfWeek	OrdersDay	OrdersTotal
2021-03-10	Wednesday	100	100
2021-03-11	Thursday	112	212

2021-03-12	Friday	320	532
2021-03-13	Saturday	0	532
2021-03-14	Sunday	0	532
2021-03-15	Monday	300	832

See:

- *FILL Function*

Calculate Rank

The **RANK** function enables you to create rankings in your data based on calculations by returning a ranking value for each row with the specified group of values. When used, some rows might receive the same value as other rows. For example, if there are three tie values in a group, the same rank is assigned to the rows and the next three ranks are skipped.

The **DENSERANK** function enables you to generate a ranked order of values within a group. If there are tie values in a group, it does not skip rank in case of tie values. For example, if two rows are listed as rank 2, then the fourth row receives rank 3.

Source:

The following dataset contains total Sales information by quarter. You can use the **RANK** and **DENSERANK** to identify the quarters with the highest sales.

Year	Quarter	Sales
2018	1	1000
2018	2	2000
2018	3	3000
2018	4	2000
2019	1	1000
2019	2	500
2019	3	9000
2019	4	3000
2020	1	500
2020	2	500
2020	3	200
2020	4	400

Transformation:

RANK:

Transformation Name	Window
Parameter: Formula type	Multiple row formula
Parameter: Formula	RANK()
Parameter: Sort rows by	Sales

Parameter: New column name	SalesRank
-----------------------------------	-----------

DENSERANK:

Transformation Name	Window
Parameter: Formula type	Multiple row formula
Parameter: Formula	DENSERANK ()
Parameter: Sort rows by	Sales
Parameter: New column name	SalesDenseRank

Results:

For the RANK function, when multiple rows share the same rank, the next rank is not consecutive, whereas for the DENSERANK function, the next rank is consecutive.

Year	Quarter	Sales	SalesDenseRank	SalesRank
2020	3	200	1	1
2020	4	400	2	2
2020	2	500	3	3
2020	1	500	3	3
2019	2	500	3	3
2019	1	1000	4	6
2018	1	1000	4	6
2018	4	2000	5	8
2018	2	2000	5	8
2019	4	3000	6	10
2018	3	3000	6	10
2019	3	9000	7	12

See:

- *RANK Function*
- *DENSERANK Function*

Calculate Rolling Functions

Rolling calculations enable you to compute a function over a changing set of rows. Rolling calculations are useful for computing the current state of a measure within your data.

For example, in the above sample data, you can find the rolling sum and rolling average of the sales for the year. You can use the above example data to find the rolling sum and rolling average.

Source:

From the following dataset, you can calculate the rolling calculations such as ROLLINGSUM, ROLLINGAVERAGE, ROLLINGMAX, and ROLLINGMIN.

--	--	--

Year	Quarter	Sales
2018	1	1000
2018	2	2000
2018	3	3000
2018	4	2000
2019	1	1000
2019	2	500
2019	3	9000
2019	4	3000
2020	1	500
2020	2	500
2020	3	200
2020	4	400

Transformation:

Transformation Name	Window
Parameter: Formulas	ROLLINGSUM (Sales, 0,1)
Parameter: Formulas	ROLLINGAVERAGE (Sales, 0,1)
Parameter: Formulas	ROLLINGMAX (Sales, 0, 1)
Parameter: Formulas	ROLLINGMIN (Sales, 0,1)
Parameter: Order by	Sales

You can rename the required columns accordingly.

Results:

Year	Quarter	Sales	RollingSumSales	RollingAverageSales	RollingMinSales	RollingMaxSales
2020	3	200	600	300	200	400
2020	4	400	900	450	400	500
2020	2	500	1000	500	500	500
2020	1	500	1000	500	500	500
2019	2	500	1500	750	500	1000
2019	1	1000	2000	1000	1000	1000
2018	1	1000	3000	1500	1000	2000
2018	4	2000	4000	2000	2000	2000
2018	2	2000	5000	2500	2000	3000
2019	4	3000	6000	3000	3000	3000
2018	3	3000	12000	6000	3000	9000
2019	3	9000	9000	9000	9000	9000

See:

- *ROLLINGAVERAGE Function*
- *ROLLINGMAX Function*
- *ROLLINGMIN Function*
- *ROLLINGMODE Function*

Rolling date functions

The Rolling date functions enable you to calculate forward or backward of the current row within the specified column. For example, when dealing with business calendars, you might want to know if the date falls on a holiday or weekend; based on that, you can roll the date forward or backward according to the business calendar.

Source:

The following example dataset shows the order date, order quantity that belongs to a product group. You are interested in finding the rolling minimum and maximum dates for the product group, as well as the rolling mode value. You can use `ROLLINGMINDATE`, `ROLLINGMAXDATE`, and `ROLLINGMODEDATE` functions.

Order_date	Order_quantity	Product_Group
2021-04-14	750	PG001
2021-07-13	1500	PG001
2021-08-31	355	PG002
2021-02-16	2000	PG002
2021-05-13	867	PG002
2021-06-18	1010	PG002
2021-11-15	909	PG003
2021-10-16	200	PG003
2021-09-09	200	PG004
2021-01-01	900	PG004
2021-12-07	707	PG004

Transformation:

Transformation Name	Window
Parameter: Formulas	<code>ROLLINGSUM (Sales, 0,1)</code>
Parameter: Formulas	<code>ROLLINGMAXDATE (Order_date, 0,1)</code>
Parameter: Formulas	<code>ROLLINGMINDATE (Order_date, 0, 1)</code>
Parameter: Formulas	<code>ROLLINGMODEDATE (Order_date, 0,1)</code>
Parameter: Order by	Order_date

Results:

Order_date	Order_quantity	Product_Group	RollingMaxdate	RollingMindate	RollingModedate
2021-01-01	900	PG004	2021-02-16	2021-01-01	2021-01-01

2021-02-16	2000	PG002	2021-04-14	2021-02-16	2021-02-16
2021-04-14	750	PG001	2021-05-13	2021-04-14	2021-04-14
2021-05-13	867	PG002	2021-06-18	2021-05-13	2021-05-13
2021-06-18	1010	PG002	2021-07-13	2021-06-18	2021-06-18
2021-07-13	1500	PG001	2021-08-31	2021-07-13	2021-07-13
2021-08-31	355	PG002	2021-09-09	2021-08-31	2021-08-31
2021-09-09	200	PG004	2021-10-16	2021-09-09	2021-09-09
2021-10-16	200	PG003	2021-11-15	2021-10-16	2021-10-16
2021-11-15	909	PG003	2021-12-07	2021-11-15	2021-11-15
2021-12-07	707	PG004	2021-12-07	2021-12-07	2021-12-07

See:

- *ROLLINGMAXDATE Function*
- *ROLLINGMINDATE Function*
- *ROLLINGMODEDATE Function*

Working with Arrays

Contents:

- *Array Types*
 - *Create Arrays*
 - *Create by extraction*
 - *Create by nesting*
 - *Create from column values*
 - *Create from Object type*
 - *Read from Arrays*
 - *Compute from Arrays*
 - *Combine Arrays*
 - *Break out Arrays*
 - *Expand arrays into rows*
 - *Unnest array elements into columns*
-

This section describes how to work with the Array data type in the Designer Cloud® application . An **array** is a set of delimited values. Any individual value in the list can be a separate array, which allows for the creation of nested data arrays.

Array Types

To be recognized as an array, a source column must contain values that are:

- Bracketed by square brackets
- Values in cell are delimited by commas

Such columns are likely to be recognized as Array data type.

The following are valid arrays:

```
[1,2,3]
["A","B"]
["C",["D","E"],"F",["G",["H","I"]]]
```

- **Ragged arrays:** If the number of elements varies between two arrays, they are considered ragged. In the above, all three arrays have a different number of top-level elements (3,2,4).
- **Nested arrays:** When an array element is an array itself, the element is considered a nested array. See the last example above.

For more information, see *Array Data Type*.

Create Arrays

Within Designer Cloud powered by Trifacta® Enterprise Edition, you can generate arrays using values from one or more columns to do so.

Create by extraction

You can create an array of values by extracting pattern-based values from a specified column. The following transformation extracts from the `msg` column a list of all values where all letters are capitalized and places them into the new `acronyms` column:

Transformation Name	Extract matches into Array
Parameter: Column	msg
Parameter: Pattern matching elements in the list	`{upper}+`
Parameter: New column name	acronyms

msg	acronyms
SCUBA, IMHO, is the greatest sport in the world.	["SCUBA","IMHO"]
	[]
LOL, that assignment you finished is DOA. You need to fix it PDQ.	["LOL","DOA","Y","PDQ"]

Notes:

- An empty input column value renders an empty array.
- In the final row, the Pattern matches on the "Y" value. To fix this, you can change the Pattern matching value to the following, which matches on two or more uppercase letters in a row:

```
`{upper}{upper}+`
```

Create by nesting

You can create arrays by nesting together the values from multiple columns.

Source:

num1	num2	num3
11	12	13
14	15	16
17	18	19

You want to nest the values in num1 and num2 into a single array and then to nest the array with num3:

NOTE: If you are nesting a multi-level array, you should nest from the lowest level to the top level.

Transformation Name	Nest columns into Objects
Parameter: Columns1	num1
Parameter: Columns2	num2
Parameter: Nest columns to	Array
Parameter: New column name	nest1

Then, you can perform the nesting of the top-level elements:

NOTE: The order in which you list the columns to nest determines the order in which the elements appear in the generated array.

Transformation Name	Nest columns into Objects
Parameter: Columns1	nest1
Parameter: Columns2	num3
Parameter: Nest columns to	Array
Parameter: New column name	nest2

In the generated columns, you notice that all values are quoted, even though these values are integers.

NOTE: Elements that are generated into arrays using a nest transformation are always rendered as quoted values.

You can use the following transformation to remove the quotes from the `nest2` column:

Transformation Name	Replace text or patterns
Parameter: Column	nest2
Parameter: Find	' '
Parameter: Replace	(empty)
Parameter: Match all occurrences	true

After removing the unused `nest1` column, the data looks like the following:

num1	num2	num3	nest2
11	12	13	[[11,12],13]
14	15	16	[[14,15],16]
17	18	19	[[17,18],19]

Create from column values

You can use one of several available functions to create arrays from a column's values.

Source:

listVals
5
TRUE
{"key1":"value1","keys2":"value2"}
[1,2,3]
My String
-5.5

The following transformation generates a new column in which each row contains an array of all of the values of the input column:

--

Transformation Name	New formula
Parameter: Formula type	Single row formula
Parameter: Formula	LIST(listVals,1000)
Parameter: New column name	listOfListVals

Results:

listVals	listOfListVals
5	["5","TRUE",{"key1\":"value1\","keys2\":"value2\"},[1,2,3],"My String",-5.5]
TRUE	["5","TRUE",{"key1\":"value1\","keys2\":"value2\"},[1,2,3],"My String",-5.5]
{"key1\":"value1","keys2\":"value2"}	["5","TRUE",{"key1\":"value1\","keys2\":"value2\"},[1,2,3],"My String",-5.5]
[1,2,3]	["5","TRUE",{"key1\":"value1\","keys2\":"value2\"},[1,2,3],"My String",-5.5]
My String	["5","TRUE",{"key1\":"value1\","keys2\":"value2\"},[1,2,3],"My String",-5.5]
-5.5	["5","TRUE",{"key1\":"value1\","keys2\":"value2\"},[1,2,3],"My String",-5.5]

Notes:

- The second parameter on the LIST function defines the maximum number of values to write. 1000 is the default.
- All values in the generated array are written as String values.
- Quoted values are escaped in the output.

The following functions allow you to generate various types of arrays from a column's set of values.

Function	Description
<i>LIST Function</i>	Extracts the set of values from a column into an array stored in a new column. This function is typically part of an aggregation.
<i>UNIQUE Function</i>	Extracts the set of unique values from a column into an array stored in a new column. This function is typically part of an aggregation.
<i>LISTIF Function</i>	Returns list of all values in a column for rows that match a specified condition.
<i>ROLLINGLIST Function</i>	Computes the rolling list of values forward or backward of the current row within the specified column and returns an array of these values.
<i>RANGE Function</i>	Computes an array of integers, from a beginning integer to an end (stop) integer, stepping by a third parameter. NOTE: The lower bound of the range is included, while the upper bound is not.

Tip: Additional examples are available in the above links for these functions.

Create from Object type

You can extract the keys of an Object column into an array of string values. In an Object type, the values are listed in quoted key/value pairs and can be nested. See *Object Data Type*.

Source:

Suppose your Object data looks like the following:

myObject
{"key1":"value1","key2":"value2","key3":"value3"}
{"apples":"2","oranges":"4" }
{"planes":{"boeing":"5","airbus":"4"},"trains":{"amtrak":"1","SP":"2"},"automobiles":{"toyota":"100","nissan":"50"}}

You can run the following transformation to extract the top-level keys into arrays in a new named column:

NOTE: The KEYS function retrieves only the top-level keys from the Object.

Transformation Name	New formula
Parameter: Formula type	Single row formula
Parameter: Formula	KEYS(myObject)
Parameter: New column name	myObjectKeys

Results:

myObject	myObjectKeys
{"key1":"value1","key2":"value2","key3":"value3"}	["key1","key2","key3"]
{"apples":"2","oranges":"4"}	["apples","oranges"]
{"planes":{"boeing":"5","airbus":"4"},"trains":{"amtrak":"1","SP":"2"},"automobiles":{"toyota":"100","nissan":"50"}}	["planes","trains","automobiles"]

For more information, see *KEYS Function*.

Read from Arrays

You can read values from arrays in your dataset.

NOTE: After an array has been created, you can append to the array or otherwise combine it with another array. You cannot replace values in the array without breaking apart the array and rebuilding it.

Function	Description
<i>IN Function</i>	Returns <code>true</code> if the first parameter is contained in the array of values in the second parameter.
<i>ARRAYELEMENTAT Function</i>	Computes the 0-based index value for an array element in the specified column, array literal, or function that returns an array.
<i>ARRAYLEN Function</i>	Computes the number of elements in the arrays in the specified column, array literal, or function that returns an array.
<i>ARRAYUNIQUE Function</i>	Generates an array of all unique elements among one or more arrays.

Tip: Additional examples are available in the above links for these functions.

Compute from Arrays

You can use the following functions to perform computations on the values in your arrays:

Function	Description
<i>LISTSUM Function</i>	Computes the sum of all numeric values found in input array. Input can be an array literal, a column of arrays, or a function returning an array. Input values must be of Integer or Decimal type.
<i>LISTMAX Function</i>	Computes the maximum of all numeric values found in input array. Input can be an array literal, a column of arrays, or a function returning an array. Input values must be of Integer or Decimal type.
<i>LISTMIN Function</i>	Computes the minimum of all numeric values found in input array. Input can be an array literal, a column of arrays, or a function returning an array. Input values must be of Integer or Decimal type.
<i>LISTAVERAGE Function</i>	Computes the average of all numeric values found in input array. Input can be an array literal, a column of arrays, or a function returning an array. Input values must be of Integer or Decimal type.
<i>LISTVAR Function</i>	Computes the variance of all numeric values found in input array. Input can be an array literal, a column of arrays, or a function returning an array. Input values must be of Integer or Decimal type.
<i>LISTSTDEV Function</i>	Computes the standard deviation of all numeric values found in input array. Input can be an array literal, a column of arrays, or a function returning an array. Input values must be of Integer or Decimal type.
<i>LISTMODE Function</i>	Computes the most common value of all numeric values found in input array. Input can be an array literal, a column of arrays, or a function returning an array. Input values must be of Integer or Decimal type.

Combine Arrays

You can combine arrays together using a variety of methods of combining.

Source:

array1	array2
["1","2","3"]	["A","B","C"]
["4","5","6"]	["D","E","F"]
["7","8","9"]	["G","H","I"]

The following transformation concatenates the above arrays into a single single array:

Transformation Name	New formula
Parameter: Formula type	Single row formula
Parameter: Formula	ARRAYCONCAT([array1,array2])
Parameter: New column name	arrayConcat

Results:

array1	array2	arrayConcat
["1","2","3"]	["A","B","C"]	["1","2","3","A","B","C"]
["4","5","6"]	["D","E","F"]	["4","5","6","D","E","F"]
["7","8","9"]	["G","H","I"]	["7","8","9","G","H","I"]

These functions can be used to combine arrays together:

Function	Description
<i>ARRAYCONCAT Function</i>	Combines the elements of one array with another, listing all elements of the first array before listing all elements of the second array.
<i>ARRAYCROSS Function</i>	Generates a nested array containing the cross-product of all elements in two or more arrays.
<i>ARRAYINTERSECT Function</i>	Generates an array containing all elements that appear in multiple input arrays, referenced as column names or array literals.
<i>ARRAYSTOMAP Function</i>	Combines one array containing keys and another array containing values into an Object of key-value pairs.
<i>ARRAYZIP Function</i>	Combines multiple arrays into a single nested array, with element 1 of array 1 paired with element 2 of array 2 and so on. Arrays are expressed as column names or as array literals.

Tip: Additional examples are available in the above links for these functions.

Break out Arrays

Expand arrays into rows

You can break out arrays into individual values using the following transformations. Here is some example data from the `nest2` column that was generated earlier. The `num3` column is retained for reference:

num3	nest2
13	[[11,12],13]
16	[[14,15],16]
19	[[17,18],19]

You can use the following simple transformation to flatten the values in `nest2` into individual values in each row:

NOTE: Depending on the number of elements in your arrays, you can significantly increase the size of your dataset.

NOTE: If a cell in the source column does not contain an array, an empty value is written into the corresponding row.

Transformation Name	Expand Array to rows
Parameter: column	<code>nest2</code>

Results:

num3	nest2
13	[11,12]

13	13
16	[14,15]
16	16
19	[17,18]
19	19

NOTE: Converting a column of arrays to rows unpacks the top level of the array only. You may have to apply this transformation multiple times.

Unnest array elements into columns

You can break out individual elements of an array into separate columns.

NOTE: Each element that you want broken out into a column must be listed on a separate line in Path to elements.

Source:

arrayNested
["A",["B","C"],"D"]
["H",["I","J"],["K","L"]]
["E","F","G"]

The following transform retrieves the second and third elements of each array:

Transformation Name	Unnest Objects into columns
Parameter: Column	arrayNested
Parameter: Paths to elements1	[1]
Parameter: Paths to elements2	[2]
Parameter: Include original column name	true

This one retrieves the first element of the array that is nested as the second element of the array:

Transformation Name	Unnest Objects into columns
Parameter: Column	arrayNested
Parameter: Paths to elements1	[1][0]
Parameter: Include original column name	true

The resulting data should look like the following:

arrayNested	arrayNested_1	arrayNested_2
["A",["B","C"],"D"]	["B","C"]	B
["H",["I","J"],["K","L"]]	["I","J"],["K","L"]	I

["E","F","G"]	F	
---------------	---	--

Working with JSON v2

Contents:

- *Enable*
 - *Requirements*
 - *JSON input*
 - *JSON output*
 - *Example*
 - *Example 1 - Rows of JSON records*
 - *Example 2 - Top-level array of JSON records*
-

Version 2: This section describes how you can import JSON files into Designer Cloud powered by Trifacta® Enterprise Edition, convert them to tabular format, wrangle them, and then export them back in the same JSON format.

The basic workflow is described by way of example. In the example workflow, the JSON file must be imported into Designer Cloud powered by Trifacta® Enterprise Edition, a new column must be inserted into the JSON, and the resulting JSON must be exported in the same structure.

Enable

This method of working with JSON is enabled by default.

You can choose to continue using the legacy method of working with JSON.

NOTE: The legacy version of JSON import is required if you are working with compressed JSON files or only Newline JSON files.

You should migrate your flows to using the new version.

NOTE: The legacy version of working with JSON is likely to be deprecated in a future release.

For more information on migrating to the new version, see *Working with JSON v1*.

Requirements

JSON input

- Recommended limit of 1 GB in source file size. Since conversion happens within the Trifacta node, this limit may vary depending on the memory of the Trifacta node.
- Filename extensions must be `.json` or `.JSON`.
- Conversion of compressed JSON files is not supported. Compressed JSON files can be imported using the previous method. See *Working with JSON v1*.
- For best results, all keys and values should be quoted and imported as strings.

NOTE: Escape characters that make JSON invalid can cause your JSON file to fail to import.

- You can escape quote values to treat them as literals in your strings using the backslash character. For example: \"
- When the values are imported into the Transformer page, the Designer Cloud application re-infers the data type for each column.

JSON structure	Description	Supported?
Newline	<p>The newline character (\n) denotes the end of a record. Each record can contain the keys (object or array) and values for the JSON object.</p> <div> Tip: This version is supported through both versions of JSON import, but it performs better in v1. If you are using the Newline form of JSON exclusively, you should use v1. </div>	Supported
Top-level object	Top-level row contains keys for mapping JSON objects	Supported
Top-level array	Top-level row contains array of objects	Supported

JSON output

NOTE: JSON-formatted files that are generated by Designer Cloud powered by Trifacta Enterprise Edition are rendered in JSON Lines format, which is a single line per-record variant of JSON. For more information, see <http://jsonlines.org>.

- Designer Cloud powered by Trifacta Enterprise Edition can generate a JSON file as an output for your job. Characteristics of generated JSON files:
 - **Newline-delimited:** The end of each record is the \n character. If your downstream system is expecting comma-delineated records except for the last one, additional work is required outside of the application.
 - **Non-nested:** Each record in the generated file is flat.
 - For multi-level JSON hierarchies, you can nest columns together and leave the top level as a set of columns in the data grid. However, on output, the second and lower hierarchies appear as quoted string values in the output. Additional cleanup is required outside of the application.

Example

Example 1 - Rows of JSON records

The following example contains records of images from a website:

```
{
  "metrics": [
    {
      "rank": "1043",
      "score": "9679"
    }
  ],
  "caption": "Such a good boy!",
  "id": "9kt8ex",
  "url": "https://www.example.com/w285fppl1.jpg",
  "filename": "w285fppl1.jpg"
},
{
  "metrics": [
    {
      "rank": "1042",
      "score": "9681"
    }
  ],
  "caption": "This sweet puppy has transformed our life!",
  "id": "9x2774",
  "url": "https://www.example.com/fml10cy11.jpg",
  "filename": "fml10cy11.jpg"
},
{
  "metrics": [
    {
      "rank": "1041",
      "score": "9683"
    }
  ],
  "caption": "We sure love our fur babies.",
  "id": "a8guou",
  "url": "https://www.example.com/mljnmq521.jpg",
  "filename": "mljnmq521.jpg"
}
```

Notes:

- Each row is a complete JSON record containing keys and values.

Tip: Nested JSON, such as `metrics` above, can be inserted as part of a record. It can then be unnested within the application.

- Each key's value must have a comma after it, except for the final key value in any row.

NOTE: The end of a JSON record is the right curly bracket (}). Commas are not added to the end of each line in this format.

Workflow

1. Import the JSON file.
2. Any nested data must be unnested within columns. Each level in the JSON hierarchy must be un-nested in a separate step.
3. When all of the JSON data is in tabular form, perform any **Wrangle** transformations.
4. If you need to rebuild the loose JSON hierarchy, you must nest the lower levels of the JSON hierarchy back into their original form.
 - a. If it is ok to write out flat JSON records, you can export without nesting the data again.
5. Run the job, generating a JSON output.

Step - Import the file

1. Through the Import Data page, navigate and select your JSON file for import.

NOTE: File formats are detected based on the file extension. Please verify that your file extension is `.json` or `.JSON`, which ensures that it is passed through the conversion service.

- a. The file is passed through the conversion process, which reviews the JSON file and stores it on the base storage layer in a format that can be easily ingested as in row-per-record format. This process happens within the Import Data page. You can track progress on the right side of the screen.
2. After the file has been converted, click the Preview icon on the right side of the screen. In the Preview, you can review the first few rows of the imported file.
 - a. If some rows are missing from the preview, then you may have a syntax error in the first row after the last well-structured row. You should try to fix this in source and re-import.
 - b. If all of the rows are problematic, your data is likely malformed.
 3. Complete the rest of the import process. For more information, see *Import Data Page*.
 4. Add the JSON-based imported dataset to a flow and create a recipe for it. For more information, see *Flow View Page*.
 - a. Select the recipe, and click **Edit Recipe....**

In the Transformer page, the example above should look like the following:

metrics	caption	id	url	filename
[{"rank": "1043", "score": "9679"}]	Such a good boy!	9kt8ex	https://www.example.com/w285fpp11.jpg	w285fpp11.jpg
[{"rank": "1042", "score": "9681"}]	This sweet puppy has transformed our life!	9x2774	https://www.example.com/fmll0cy11.jpg	fmll0cy11.jpg
[{"rank": "1041", "score": "9683"}]	We sure love our fur babies.	a8guou	https://www.example.com/mljnmq521.jpg	mljnmq521.jpg

Step - Unnest JSON records

Your JSON records are in tabular format. If you have nested JSON objects within your JSON records, the next step is to unnest your JSON records.

NOTE: For JSON records that have multiple levels in the hierarchy, you should unnest the top level of the hierarchy first, followed by each successive level.

Tip: The easiest way to unnest is to select the column header for the column containing your nested data. Unnest should be one of the suggested options, and the suggestion should include the specification for the paths to the key values. If not, you can use the following process.

1. In the Recipe panel, click **New Step**.
2. In the Search panel, enter `unnest values into new columns`.
3. Specify the following transformation. Substitute the Paths to elements values below with the top-level keys in your JSON records:

Transformation Name	Unnest values into new columns
Parameter: Column	metrics
Parameter: Path to elements1	[0]

Tip: You can choose to remove the original from the source or not. In deeper or wider JSON files, removing can help to identify what remains to be unnested.

4. In the above transformation, the bracketing array around the set of values has been broken down into raw JSON. This value may now be interpreted as a String data type. From the column drop-down, you can select Object data type.
5. Click the column head again, or specify the following transformation to unnest the Object column:

Transformation Name	Unnest Objects into columns
Parameter: Column	0
Parameter: Path to elements1	rank
Parameter: Path to elements2	score

- a. In the above, each Paths to elements entry specifies a key in the JSON record. The key's associated value becomes the value in the new column, which is given the same name as the key.
 - b. So, this step breaks out the key-value pairs for the specified keys into separate columns in the dataset.
6. Repeat the above process for the next level in the hierarchy.
 7. You can now delete the source columns. In the example, these source columns are named `metrics` and `0`.

Tip: SHIFT + click these columns and then select **Delete columns** from the right panel. Click **Add**.

8. Repeat the above steps for each nested JSON object.

Tip: If the above set of steps needs to be applied to multiple files, you might consider stopping your work and returning to Flow View. Select this recipe and click **Add New Recipe**. If you add successive steps in another recipe, the first one can be used for doing initial processing of your JSON files, separate from any wrangling that you may do for individual files.

Tip: The unnesting process may have moved some columns into positions that are different from their order in the original JSON. Use the **Move** command from the column menu to reposition your columns.

Step - Wrangle your dataset

Your JSON data is ready for wrangling. Continue adding steps until you have transformed your data as needed and are ready to run a job on it.

Step - Nest the JSON records

NOTE: If your desired JSON output does not include multiple hierarchies, you can skip this section. The generated JSON files are a single JSON record per row.

If you ran a job on the example dataset, the output would look like the following:

```
{ "rank":1043,"score":9679,"caption":"Such a good boy!","id":"9kt8ex","url":"https://www.example.com/w285fpp11.jpg","filename":"w285fpp11.jpg" }
{ "rank":1042,"score":9681,"caption":"This sweet puppy has transformed our life!","id":"9x2774","url":"https://www.example.com/fml10cy11.jpg","filename":"fml10cy11.jpg" }
{ "rank":1041,"score":9683,"caption":"We sure love our fur babies.","id":"a8guou","url":"https://www.example.com/mljnmq521.jpg","filename":"mljnmq521.jpg" }
```

Suppose you want to nest the `url` and `filename` columns into a nested array called, `resources`.

Re-nest the lower hierarchies until you have a single flat record, containing some Object type columns that hold the underlying hierarchies. When the re-nested JSON records are exported, secondary hierarchies appear as escaped string values. More details later.

Tip: The following steps reshape your data. You may wish to create a new recipe as an output of the previous recipe where you can add the following steps.

Steps:

1. SHIFT + click the `url` and `filename` columns. Then, select **Nest columns** in the right-hand panel. This transformation should look like the following:

Transformation Name	Nest columns into Objects
Parameter: column1	url
Parameter: column2	filename
Parameter: Nest columns to	Object
Parameter: New column name	column1

2. `column1` now contains an Object mapping of the two columns. You can now nest this column again into an Array:

Transformation Name	Nest columns into Objects
Parameter: Columns	column1
Parameter: Nest columns to	Array

Parameter: New column name	resources
----------------------------	-----------

3. Delete `column1`.
4. Continue nesting other columns in a similar fashion. Repeat the above steps for the next level of the hierarchy in your dataset.
5. You must re-nested from the bottom of the target hierarchy to the top.

NOTE: Do not nest the columns at the top level of the hierarchy.

6. When the column names contain all of the keys that you wish to generate in the top-level JSON output, you can run the job.

Step - Generate JSON output

When you are ready, you can run the job. Create or modify a publishing action to generate a JSON file for output. See [Run Job Page](#).

When the job completes, you can click the JSON link in the Output Destinations tab of the Job Details page to download your JSON file. See [Job Details Page](#).

Output file for the above example should look like the following:

```
{
  "rank":1043,"score":9679,"caption":"Such a good boy!","id":"9kt8ex","url":"https://www.example.com/w285fpp11.jpg",
  "filename":"w285fpp11.jpg","resources":[{"url":"https://www.example.com/w285fpp11.jpg","filename":"w285fpp11.jpg"}]}
{
  "rank":1042,"score":9681,"caption":"This sweet puppy has transformed our life!","id":"9x2774","url":"https://www.example.com/fml10cy11.jpg",
  "filename":"fml10cy11.jpg","resources":[{"url":"https://www.example.com/fml10cy11.jpg","filename":"fml10cy11.jpg"}]}
{
  "rank":1041,"score":9683,"caption":"We sure love our fur babies.","id":"a8guou","url":"https://www.example.com/mljnmq521.jpg",
  "filename":"mljnmq521.jpg","resources":[{"url":"https://www.example.com/mljnmq521.jpg","filename":"mljnmq521.jpg"}]}
}
```

Example 2 - Top-level array of JSON records

Your JSON may be formatted as a single top-level object containing an array of JSON records. The following example contains records of messages about individual diet and exercise achievements:

```
{
  "object": [
    {
      "score": 19669,
      "title": "M/07/1'3\" [23lbs > 13lbs = 10lbs] Still a bit to go, but my owner no longer refers to me as his chunky boy!",
      "ups": 19669,
      "id": "9kt8ex",
      "url": "https://i.redd.it/bzygw285fpp11.jpg",
      "short": "bzygw285fpp11.jpg"
    },
    {
      "score": 19171,
      "title": "M/29/5'11\" [605 pounds > 375 pounds = 230 pounds lost] (14 months) Still considered super morbidly obese but I've made some good progress.",
      "ups": 19171,
      "id": "9x2774",
      "url": "https://i.redd.it/wbbufml10cy11.jpg",
      "short": "wbbufml10cy11.jpg"
    },
    {
      "score": 16778,
      "title": "F/28/5'10\" [233lbs to 130lbs] Got tired of being obese and took control of my life!",
      "ups": 16778,

```

```

    "id": "a8guou",
    "url": "https://i.redd.it/3t0kmljnmq521.jpg",
    "short": "3t0kmljnmq521.jpg"
  },
  {
    "score": 16743,
    "title": "M/22/5'11\" [99lbs > 150lbs = 51lbs] Anorexia my recovery",
    "ups": 16743,
    "id": "atla3n",
    "url": "https://i.redd.it/9t6tvsjs16i21.jpg",
    "short": "9t6tvsjs16i21.jpg"
  }
]
}

```

The outer JSON is a single key-value pair:

- key: object
- value: array of JSON records

When source JSON records structured in this manner are imported, each JSON record in the object is imported into a separate row. You can unnest this data by applying an Unnest values transformation.

NOTE: The object can contain only one nested array of JSON data. If the object contains multiple nested arrays, it is not broken into separate rows. All unnesting must be performed in your recipe steps

Suppose you want to compute the average of all workout scores. First, you must unnest the JSON records and then apply the AVERAGE function.

Steps:

Tip: The easiest way to unnest is to select the column header for the column containing your data. After you select the column header, you are provided with suggestions to Unnest Values into new columns. You can use the Unnest suggestion and click **Add**. The following steps illustrate how to create this transformation manually.

1. In the Recipe panel, click **New Step**.
2. In the Search panel, enter `unnest values into new columns`.
3. Specify the following transformation. Substitute the Paths to elements values below with the top-level keys in your JSON records:

Transformation Name	Unnest values into new columns
Parameter: Column	object
Parameter: Path to elements	id
Parameter: Path to elements	score
Parameter: Path to elements	short
Parameter: Path to elements	title
Parameter: Paths to elements	ups
Parameter: Path to elements	url

- The above step breaks out the key-value pairs for the specified keys into separate columns in the dataset. Each Paths to elements entry specifies a key in the JSON record, which is used to create a new column of the same name. The key's associated value becomes a cell value in the new column.
- You can now delete the source column. In the example, the source column is `object`.

Tip: You can choose to remove the original from the source or not. In deeper or wider JSON files, removing can help to identify what remains to be unnested. When you're done unnesting a column and have removed data from the original, you should have an empty column.

Results:

id	score	short	title	ups	url
9kt 8ex	19669	bzygw285 fpp11.jpg	M/07/1'3" [23lbs > 13lbs = 10lbs] Still a bit to go, but my owner no longer refers to me as his chunky boy!	19669	https://i.redd.it/bzygw285fpp11.jpg
9x2 774	19171	wbbufmll0 cy11.jpg	M/29/5'11" [605 pounds > 375 pounds = 230 pounds lost] (14 months) Still considered super morbidly obese but I've made some good progress.	19171	https://i.redd.it/wbbufmll0cy11.jpg
a8g uou	16778	3t0kmljnm q521.jpg	F/28/5'7" [233lbs to 130lbs] Got tired of being obese and took control of my life!	16778	https://i.redd.it/3t0kmljnmq521.jpg
atla 3n	16743	9t6tvsjs16 i21.jpg	M/22/5'11" [99lbs > 150lbs = 51lbs] Anorexia my recovery	16743	https://i.redd.it/9t6tvsjs16i21.jpg

Now you can find the average score by applying average function.

Transformation Name	New formula
Parameter: Formula type	Single row formula
Parameter: Formula	AVERAGE(score)
Parameter: New column name	Average_score

Results:

id	score	short	title	ups	url	Average_score
9kt 8ex	19669	bzygw285 fpp11.jpg	M/07/1'3" [23lbs > 13lbs = 10lbs] Still a bit to go, but my owner no longer refers to me as his chunky boy!	19669	https://i.redd.it/bzygw285fpp11.jpg	18090.25
9x2 774	19171	wbbufmll0 cy11.jpg	M/29/5'11" [605 pounds > 375 pounds = 230 pounds lost] (14 months) Still considered super morbidly obese but I've made some good progress.	19171	https://i.redd.it/wbbufmll0cy11.jpg	18090.25
a8g uou	16778	3t0kmljnm q521.jpg	F/28/5'7" [233lbs to 130lbs] Got tired of being obese and took control of my life!	16778	https://i.redd.it/3t0kmljnmq521.jpg	18090.25
atla 3n	16743	9t6tvsjs16 i21.jpg	M/22/5'11" [99lbs > 150lbs = 51lbs] Anorexia my recovery	16743	https://i.redd.it/9t6tvsjs16i21.jpg	18090.25

Working with JSON v1

Contents:

- *Enable*
 - *Migrate to v2*
- *JSON Input and Output*
- *Example*
- *JSON Workflow*
 - *Step - Import the file*
 - *Step - Convert to one JSON record per row*
 - *Step - Convert JSON to Object type*
 - *Step - Unnest JSON records*
 - *Step - Wrangle your dataset*
 - *Step - Nest the JSON records*
 - *Step - Generate JSON output*
 - *Step - Final Cleanup*

Version 1: This section describes how you can import JSON files into Designer Cloud powered by Trifacta® Enterprise Edition, convert them to tabular format, wrangle them, and then export them back in the same JSON format.

The basic workflow is described by way of example. In the example workflow, the JSON file must be imported into Designer Cloud powered by Trifacta® Enterprise Edition, a new column must be inserted into the JSON, and the resulting JSON must be exported in the same structure.

Enable

This legacy method of working with JSON is likely to be deprecated in a future release.

If you have existing flows that were created using this legacy method, they should continue to work as expected. However, you should migrate your flows to use the newer version as soon as possible. See [Migrate](#) below.

NOTE: The legacy version of JSON import is required if you are working with compressed JSON files or only Newline JSON files.

This method can be enabled through workspace settings. For more information, see [Workspace Settings Page](#).

Migrate to v2

Any flow that you have created using the v1 version of the JSON importer should work without modification.

In the future, the v1 version will be deprecated. You can use the following method to migrate your flows to use the new version of the JSON importer.

NOTE: The v1 version of JSON import is supported for imported datasets. If you use these datasets in other workflows, they are likely to require modifications that you have done in recipes.

Basic workflow:

This migration workflow creates new versions of these imported datasets and fixes recipes accordingly.

1. Through the Library page, locate the imported datasets that are based on JSON files.
 - a. You may be able to just search for `json`.
2. For each JSON imported dataset:
 - a. Click the link.
 - b. In the Dataset Details page, copy the value for the Location. Paste it into a text file.
 - c. In the Dataset Details page, locate flow or flows where the dataset is in use.

Tip: If you copy the link address of the flow and paste it into a text file, you can paste that later into a browser and jump directly to the flow.

- d. Repeat the above steps for each JSON-based imported dataset.
3. You should now have a list of links to the source data and the flows where your JSON imported datasets are in use.
4. In the Library page, create a new version of each imported dataset:
 - a. Click **Import Data**.
 - b. Click the appropriate connection.
 - c. Paste the link to the Location where the source is stored.
 - d. The data is ingested through the conversion service.

Tip: Click the icon for the dataset in the right panel. All rows in the Preview panel should be properly structured. Nested data may not be broken out into separate columns at this time.

- e. Rename the dataset as needed.

Tip: You should give each new version of the imported dataset a consistent prefix or suffix tag, such as `-v2`. Later, you can locate these new imported datasets easily through search in the Library.

- f. Click **Continue**.
5. Repeat the above steps for each imported dataset that you are updating to v2.
6. For each of these flows:
 - a. Navigate to it.
 - b. Locate the v1 imported dataset in it. You might copy the name.
 - c. Click **Add Datasets**. Search for the v2 imported dataset. Add it to the flow.
7. In Flow View:
 - a. Click the recipe that is in use with the v1 version of the imported dataset. In the context menu in the right panel, select **Make a copy > without inputs**.
 - b. Select the copied recipe.
 - c. In the context menu in the right panel, select **Change input**. Select the v2 imported dataset.
 - d. Your v2 imported dataset is now connected to a version of your recipe.
 - e. Select the recipe object. In the right panel, you should see a preview of the recipe steps.

NOTE: In the recipe, the steps where you modified the imported dataset into tabular format are likely to be broken. This is ok.

8. Click **Edit recipe**.
9. In the Transformer page:
 - a. Disable recipe step 1.

- b. Review the state of the data grid to see if the data is organized in tabular form.
 - c. If not, repeat the above steps for the next step in your recipe.
 - d. Continue until the data is in tabular form.
- 10. After some additional tweaking, your recipe should contain no broken steps, and your data should appear in tabular form.
- 11. You may wish to run a job or download your sample data to compare it to outputs from your v1 imported dataset and steps. You may need to create an output object first.
- 12. You can now integrate these changes in either of the following ways:
 - a. **Apply to existing recipe:** Change the input on the existing to the v2 imported dataset. Apply any disabling of steps and other tweaks to the recipe's connected to the v1 imported dataset.

NOTE: Before applying the above changes, you might want to download the v1 recipe through the Recipe panel.

- b. **Use v2 recipe in the flow:** You could simply switch over to using the new recipe. Caveats:
 - i. You must recreate any outputs and schedules from the v1 recipe.
 - ii. Internal identifiers for the new recipe and its outputs are different from the v1 recipe. These new identifiers may impact API-based automation.
 - iii. Other application objects that reference the v1 recipes, such as flow tasks in your plans, must be fixed to use the new recipe or output objects.
- 13. Run a production job to verify that your flow is producing consistent data with the v2 imported dataset.
- 14. Repeat as needed for other flows.

JSON Input and Output

Input:

- It is easier to work with JSON in which each row of the file is a record. When a record spans multiple rows, additional steps are required in the application to render it into tabular format. The example uses multi-row JSON records.

Output:

NOTE: JSON-formatted files that are generated by Designer Cloud powered by Trifacta Enterprise Edition are rendered in JSON Lines format, which is a single line per-record variant of JSON. For more information, see <http://jsonlines.org>.

- Designer Cloud powered by Trifacta Enterprise Edition can generate a JSON file as an output for your job. Characteristics of generated JSON files:
 - **Newline-delimited:** The end of each record is the `\n` character. If your downstream system is expecting comma-delineated records except for the last one, additional work is required outside of the application.
 - **Non-nested:** Each record in the generated file is flat.
 - For multi-level JSON hierarchies, you can nest columns together and leave the top level as a set of columns in the data grid. However, on output, the second and lower hierarchies appear as quoted string values in the output. Additional cleanup is required outside of the application.

Example

This example dataset contains information on books. In this case:

- The data is submitted as one attribute per row. A single JSON record spans many rows.
- The total number of books is three.
- The JSON data has two hierarchies.

```

"book": {
  "id": "bk101",
  "author": "Guy, Joe",
  "title": "Json Guide",
  "genre": "Computer",
  "price": "44.95",
  "publish_date": "2002-04-26",
  "characteristics": {
    "cover_color": "black",
    "paper_stock": "20",
    "paper_source": "new"
  },
  "description": "An in-depth look at creating applications."
},
"book": {
  "id": "bk102",
  "author": "Nelson, Rogers",
  "title": "When Doves Cry",
  "genre": "Biography",
  "price": "24.95",
  "publish_date": "2016-04-21",
  "characteristics": {
    "cover_color": "white",
    "paper_stock": "15",
    "paper_source": "recycled"
  },
  "description": "Biography of a prince."
},
"book": {
  "id": "bk103",
  "author": "Fitzgerald, F. Scott",
  "title": "The Great Gatsby",
  "genre": "Fiction",
  "price": "9.95",
  "publish_date": "1925-04-10",
  "characteristics": {
    "cover_color": "blue",
    "paper_stock": "20",
    "paper_source": "new"
  },
  "description": "Classic American novel."
}

```

JSON Workflow

1. Import the JSON file.

NOTE: During import, you should deselect the Detect Structure option. You are likely to need to rebuild the initial parsing steps to consume the file properly. Details are provided later.

2. If needed, convert loose JSON to a single JSON record per row.
3. Unnest the data into columns.
 - a. Each level in the JSON hierarchy must be un-nested in a separate step.
4. When all of the JSON data is in tabular form, perform any **Wrangle** transformations.
5. If you need to retain the hierarchy, you must nest the lower levels of the JSON hierarchy back into their original form. Leave the top level un-nested.
 - a. If it is ok to write out flat JSON records, you can export without nesting the data again.
6. Run the job, generating a JSON output.

Step - Import the file

1. Through the Import Data page, navigate and select your JSON file for import.
2. When the file has been loaded, click **Edit settings** for the dataset card in the right panel. In the Import Settings dialog, deselect the Detect Structure checkbox.

3. Complete the rest of the import process. For more information, see *Import Data Page*.
4. Add the JSON-based imported dataset to a flow and create a recipe for it. For more information, see *Flow View Page*.
5. Select the recipe, and click **Edit Recipe....**

Step - Convert to one JSON record per row

NOTE: This step is required only if a single JSON record in your imported dataset spans multiple rows. If you have single-row JSON records in the Transformer page, please skip to the next section.

1. In the Transformer page, you should see your loosely formatted JSON in a single column. Each row contains a separate attribute, and a single record spans multiple rows.
2. Open the Recipe panel on the right side. The initial parsing steps for the data are displayed. For more information, see *Initial Parsing Steps*.
3. In Recipe panel, delete all steps except the first one.
4. The first one is a Break into rows transformation. This transformation can only appear in the first step of a recipe.
5. Select the step, and click the Pencil icon to edit it.
6. In the Transform Builder, the Split on value is probably the `\n` character.
7. The above signals to the application to break up the data into individual rows on the newline (`\n`) character. This transformation then breaks up your loose JSON on every single attribute. You must modify the Split on value so that it captures only the first attribute of each JSON record. For the above dataset, the Split on value must be the following, noting the space after the colon:

```
"book" :
```

8. Click **Add** to save the step again.
9. The above dataset should now have four rows, with the first one an empty row. This empty row is caused by the insertion of the `\n` in front of the first reference to the above string. In the column histogram, select the gray bar, which selects the empty row. In the Suggestions panel, locate the Delete rows suggest, and click **Add**. The row is removed.
10. You now have individual rows for each JSON record.

Step - Convert JSON to Object type

The next step involves converting your JSON records to a column of Object type values. The Object data type is a means of rendering records into key-value pairs. However, its structure is a little different from JSON. The following steps convert your JSON to an Object data type.

- For more information, see *Object Data Type*.
1. Since JSON uses character indentation to convey structure, you should remove these indentations if they appear in your dataset. For our two-layered example, you can use the following transformation:

Transformation Name	Replace text or patterns
Parameter: Column	column1
Parameter: Find	<code>/\n\s*"/</code>
Parameter: Replace with	<code>\</code>
Parameter: Match all occurrences	true

- a. In the above, the key term is the Find pattern, which is a **regular expression**:

```
/\n\s*"/
```

- b. The two forward slashes at the ends define the pattern as a regular expression.
 - c. The content in the middle matches on the pattern of a newline character, an arbitrary number of spaces, and a double quote.
 - d. This pattern is replaced with just the double-quote, removing the preceding part of the pattern from the dataset.
 - e. For more information on matching patterns, see *Text Matching*.
2. In standard JSON, a comma is used to demarcate the end of a line or a record, except for the last one in a set.
- a. In the above example, the first two records have commas at the end of them. Here is a snippet of their ends:

```
... "description":"An in-depth look at creating applications."},  
... "description":"Biography of a prince."},  
... "description":"Classic American novel."}
```

- b. To convert these records to Object type, the commas at the end of the first two rows must be removed:

Transformation Name	Replace text or patterns
Parameter: Column	column1
Parameter: Find	`\n\}, \n{end}`
Parameter: Replace with	}
Parameter: Match all occurrences	true

- i. The above transformation is similar to the previous one. However, in this one, the Find pattern uses a `Pattern` to indicate that the pattern should only be matched at the end of a record:

```
{end}
```

- ii. This token in the pattern prevents it from matching if there are other instances of the pattern nested within the record.
 - iii. For more information, see *Text Matching*.
3. Individual records should look similar to the following:

NOTE: Below, some values are too long for a single line. Single lines that overflow to additional lines are marked with a `\`. The backslash should not be included if the line is used as input.

```
{  
  "id": "bk101",  
  "author": "Guy, Joe",  
  "title": "Json Guide",  
  "genre": "Computer",  
  "price": "44.95",  
  "publish_date": "2002-04-26",  
  "cover_color": "black",  
  "paper_stock": "20",  
  "paper_source": "new",  
  "description": "An in-depth look at creating applications."  
}
```

4. These records are suitable for conversion to Object data type.
5. To change the data type for the column, click the icon to the left of the column header. Select **Object**.
6. The column data type is changed to Object. The step to change data type is added to your recipe, too.
7. If the column histogram now displays some mismatched records.
 - a. Review those records to determine what is malformed.

- b. Delete the recipe step that changes the data type to Object.
- c. Make fixes as necessary.
- d. Switch back to Object data type. Iterate as needed until all records are valid when the column is converted to Object type.

Step - Unnest JSON records

The next step is to convert your JSON records to tabular format.

NOTE: For JSON records that have multiple levels in the hierarchy, you should unnest the top level of the hierarchy first, followed by each successive level.

Tip: The easiest way to unnest is to select the column header for the column containing your Object data. Unnest should be one of the suggested options. If not, you can use the following process.

1. In the Recipe panel, click **New Step**.
2. In the Search panel, enter `unnest object elements`.
3. Specify the following transformation. Substitute the Paths to elements values below with the top-level keys in your JSON records:

Transformation Name	Unnest object elements
Parameter: Column	column1
Parameter: Path to elements1	id
Parameter: Path to elements2	author
Parameter: Path to elements3	title
Parameter: Path to elements4	genre
Parameter: Path to elements5	price
Parameter: Path to elements6	publish_date
Parameter: Path to elements7	description
Parameter: Remove elements from original	true

- a. In the above, each Paths to elements entry specifies a key in the JSON record. The key's associated value becomes the value in the new column, which is given the same name as the key.
- b. So, this step breaks out the key-value pairs for the specified keys into separate columns in the dataset.

Tip: You can choose to remove the original from the source or not. In deeper or wider JSON files, removing can help to identify what remains to be unnested.

4. Repeat the above process for the next level in the hierarchy. In the example, this step means unnesting the `characteristics` node:

Transformation Name	Unnest object elements
----------------------------	------------------------

Parameter: Column	column1
Parameter: Path to elements1	characteristics.cover_color
Parameter: Path to elements2	characteristics.paper_stock
Parameter: Path to elements3	characteristics.paper_source
Parameter: Remove elements from original	true

5. You can now delete `column1`. From the column menu to the right of `column1`, select **Delete**.
6. You have now converted your JSON to tabular format.

Tip: If the above set of steps needs to be applied to multiple files, you might consider stopping your work and returning to Flow View. Select this recipe and click **Add New Recipe**. If you add successive steps in another recipe, the first one can be used for doing initial processing of your JSON files, separate from any wrangling that you may do for individual files.

Tip: The unnesting process may have moved some columns into positions that are different from their order in the original JSON. Use the **Move** command from the column menu to reposition your columns.

Step - Wrangle your dataset

Your JSON data is ready for wrangling.

In the following example, the `discount` column is created. If the publication date is before 01/01/2000, then the discount is 0.1 (10%):

Transformation Name	New formula
Parameter: Formula type	Single row formula
Parameter: Formula	IF(publish_date < DATE(2000, 1, 1), 0.1, 0)
Parameter: New column name	discount

Continue adding steps until you have transformed your data as needed and are ready to run a job on it.

Step - Nest the JSON records

NOTE: If your desired JSON output does not include multiple hierarchies, you can skip this section. The generated JSON files are a single JSON record per row.

If a job is run using the recipe created so far on the example data, a newline-delimited JSON file that has no hierarchies in it can be generated by the application. However, the dataset is a two-level hierarchy, so the elements in the `characteristics` hierarchy are written out in the following manner:

```
"characteristics.cover_color":"black","characteristics.paper_stock":20,"characteristics.paper_source":"new",
"characteristics.cover_color":"white","characteristics.paper_stock":15,"characteristics.paper_source":"
recycled",
"characteristics.cover_color":"blue","characteristics.paper_stock":20,"characteristics.paper_source":"new",
```

You can take one of two approaches:

1. Generate the JSON file with a flat hierarchy. Output looks like the above. Use an external tool to unnest the second and lower hierarchies appropriately.
2. Re-nest the lower hierarchies until have you have a single flat record, containing some Object type columns that hold the underlying hierarchies. When the re-nested JSON records are exported, secondary hierarchies appear as escaped string values. More details later.

If you are re-nesting the lower hierarchies, you can use the following approach.

Tip: The following steps reshape your data. You may wish to create a new recipe as an output of the previous recipe where you can add the following steps.

1. When you re-nest, you want to nest from the lowest to top tier of the hierarchy.
2. In the example, the following columns should be nested together: `characteristics.cover_color`, `characteristics.paper_stock`, and `characteristics.paper_source`:

Transformation Name	Nest columns into Objects
Parameter: column1	<code>characteristics.cover_color</code>
Parameter: column2	<code>characteristics.paper_stock</code>
Parameter: column3	<code>characteristics.paper_source</code>
Parameter: Nest columns to	Object
Parameter: New column name	<code>characteristics</code>

3. In the generated `characteristics` column, you can remove the `characteristics.` from the key value:

Transformation Name	Replace text or patterns
Parameter: Column	<code>characteristics</code>
Parameter: Find	<code>`characteristics.`</code>
Parameter: Replace with	(empty)

4. Now, delete the three source columns:

Transformation Name	Delete columns
Parameter: column1	<code>characteristics.cover_color</code>
Parameter: column2	<code>characteristics.paper_stock</code>
Parameter: column3	<code>characteristics.paper_source</code>

5. Repeat the above steps for the next level of the hierarchy in your dataset.

NOTE: Do not nest the columns at the top level of the hierarchy.

Step - Generate JSON output

When you are ready, you can run the job. Create or modify a publishing action to generate a JSON file for output. See *Run Job Page*.

When the job completes, you can click the JSON link in the Output Destinations tab of the Job Details page to download your JSON file. See *Job Details Page*.

Step - Final Cleanup

Outside the application, you may need to do the following:

1. Since the JSON output is newline delimited, your downstream system may need you to add commas at the end of each record but the last one.
2. If you have re-nested JSON hierarchies into your flat records, the exported JSON for secondary hierarchies appears as quoted strings, like the following:

```
"characteristics":{"cover_color":"black","paper_stock":"20","paper_source":"new"},
"characteristics":{"cover_color":"white","paper_stock":"15","paper_source":"recycled"},
"characteristics":{"cover_color":"blue","paper_stock":"20","paper_source":"new"},
```

The quoted strings can be fixed by simple search and replace.

Cleanse Tasks

The following topics pertain to cleaning data that has been imported into Designer Cloud powered by Trifacta® Enterprise Edition.

Rename Columns

Contents:

- *Name Requirements*
 - *Reserved keywords*
- *Rename Individual Columns*
 - *Rename a column through column menu*
 - *Rename a column through suggestions*
 - *Rename a column through transformation*
 - *Rename a new column*
- *Auto-Generated Column Names*
- *Rename Multiple Columns*
 - *Manual rename multiple columns*
 - *Add prefix*
 - *Add suffix*
 - *Apply rename to all columns*
 - *Convert to lowercase*
 - *Convert to UPPERCASE*
 - *Keep from beginning (left)*
 - *Keep from end (right)*
 - *Find and replace*
 - *Use row(s) as column names*
 - *Combine multiple rows*

In the Designer Cloud® application , you can rename individual columns through the column drop-down. Through transform steps, you can apply renaming to one or more columns.

NOTE: An imported dataset requires about 15 rows to properly infer column data types and the row, if any, to use for column headers.

Name Requirements

- Column names are case-insensitive and cannot begin with whitespace.
- Column names cannot contain escaped characters, such as \n.

NOTE: When publishing to Avro, Parquet, or database tables,

column names support alphanumeric characters and the underscore (_) character only. Column names cannot begin with a numeral. Other characters cause an error to occur.

NOTE: Column names with spaces or special characters in a transformation must be wrapped by curly braces. Example:

```
column1,{Column 2 with space},column3
```

Tip: To prevent potential issues with downstream systems, you should limit your column lengths to no more than 128 characters.

Reserved keywords

The following keywords should not be used as column names, as they may conflict with underlying requirements of the platform or the running environments with which it integrates:

NOTE: This list may not be complete. If your job fails with a duplicate column error, please review your column names to identify potential reserved keywords among them.

- TRIFACTA__LINEAGE_INFO
- TRIFACTA__FILE_LINEAGE_INFO

NOTE: There are two underscore characters in a row (__) after TRIFACTA in each of the above entries.

Rename Individual Columns

Rename a column through column menu

To rename a column, click the drop-down caret next to the column name. Click **Rename**.

Rename a column through suggestions

Steps:

1. If your column already exists, click the name of the column.
2. Click the Rename suggestion card.
3. Click **Modify**.
4. Replace the `newColumnName` value with your preferred column name.

Rename a column through transformation

You can use the following transformation to rename a single column through the Transform Builder. In this case, the Rename columns transformation is used to perform a manual rename of `MySourceCol` to `MyNewCol`.

Transformation Name	Rename columns
Parameter: Option	Manual rename
Parameter: Column	MySourceCol
Parameter: New name	MyNewCol

Rename a new column

Columns that are generated through transform steps are given a default name.

For the following types of transforms, however, you can specify the column name as part of the step:

- derive
- extractkv
- merge
- nest

- `udf` - See *User-Defined Functions*.

When a transform is added to the recipe, an `as:` clause is automatically added to the transform step. You can modify your transform to change the value of the `as:` column. For example, the following transform generates a new column with the first word from the `Name` column. The `as:` value renames this generated column as `FirstName`:

Transformation Name	New formula
Parameter: Formula type	Single row formula
Parameter: Formula	<code>FIND(Name,`{start}` ,false,0)</code>
Parameter: New name	<code>FirstName</code>

Auto-Generated Column Names

When your transforms generate new columns, names are automatically assigned to these columns based on the following pattern.

1. If the transform includes a function reference, the function name is included in the new column. Example:

Transformation Name	New formula
Parameter: Formula type	Single row formula
Parameter: Formula	<code>LEFT(city,3)</code>

New column name: `left_city`

2. If the above step is applied again, a duplicate column is generated with the following name. Example:

Transformation Name	New formula
Parameter: Formula type	Single row formula
Parameter: Formula	<code>LEFT(city,3)</code>

New column name: `left_city1`

3. If the transform does not contain a function reference, the following convention is used:

Transformation Name	New formula
Parameter: Formula type	Single row formula
Parameter: Formula	<code>'A'</code>

New column name: `column1`

Transformation Name	New formula
Parameter: Formula type	Single row formula
Parameter: Formula	'B'

New column name: column2

Rename Multiple Columns

Designer Cloud powered by Trifacta Enterprise Edition enables to rename multiple columns using a single transformation. You can perform this batch renaming using one of the methods described in this section.

NOTE: In macros, Rename Columns transformations do not work. This is a known issue.

Tip: To prevent potential issues with downstream systems, you should limit your column lengths to no more than 128 characters.

Steps:

1. Open the Transform Builder to add a new step to your recipe.
2. From the drop-down in the first textbox, select `Rename columns`.
3. Select your method of renaming. See below.
4. Select the column or columns to which to apply the rename.

Tip: To apply the renaming across all columns in the dataset, select **All**. This option is useful for pattern-based renames, such as adding a prefix or changing case.

5. To add the step to your recipe, click **Add**.

Manual rename multiple columns

For each column that you select, you must add the new name just below the old one.

- To add additional columns to the mapping, click **Add**.
- To remove columns from the mapping, click **Remove**.

Add prefix

For the selected columns, you can apply a specific prefix value to the names.

Old Column Names	Prefix	New Column Names
column1	pre_	pre_column1
column2	pre_	pre_column2
column3	pre_	pre_column3

Transformation:

Transformation Name	Rename columns
----------------------------	----------------

Parameter: Option	Add prefix
Parameter: Column	column1,column2,column3
Parameter: Prefix	pre_

Add suffix

For the selected columns, you can apply a specific suffix value to the names. Example:

Old Column Names	Suffix	New Column Names
column1	_new	column1_new
column2	_new	column2_new
column3	_new	column3_new

Transformation:

Transformation Name	Rename columns
Parameter: Option	Add suffix
Parameter: Column	column1,column2,column3
Parameter: Suffix	_new

Apply rename to all columns

The following transformation performs the same rename as the previous one. Instead, it uses the All option to apply the rename across all columns of the dataset. If the number of columns changes in the future, then the rename is still applied across all of the columns in the dataset.

Transformation:

Transformation Name	Rename columns
Parameter: Option	Add suffix
Parameter: Columns	All
Parameter: Suffix	_new

Convert to lowercase

For the selected columns, you can convert the columns names to lowercase. Example:

Old Column Names	New Column Names
Daily	daily
POS_Cost	pos_cost
Sales_Type	sales_type

Transformation:

Transformation Name	Rename columns
----------------------------	----------------

Parameter: Option	Convert to lowercase
Parameter: Column	Daily, POS_Cost, Sales_Type

For example, if the old column name is `Sales_Type`, then the new column name is renamed to `sales_type`.

Convert to UPPERCASE

For the selected columns, you can convert the columns names to uppercase. Example:

Old Column Names	New Column Names
Daily	DAILY
POS_Cost	POS_COST
Sales_Type	SALES_TYPE

Transformation:

Transformation Name	Rename columns
Parameter: Option	Convert to UPPERCASE
Parameter: Column	Daily, POS_Cost, Sales_Type

For example, if the old column name is `Sales_Type`, then the new column name is renamed to `SALES_TYPE`.

Keep from beginning (left)

For the selected columns, you can specify the number of characters to keep from the beginning (left) of the column names. Based on the number of characters you provide, the column name is updated. Example:

Old Column Names	Number of characters	New Column Names
Daily	3	Dai
POS_Cost	3	POS
Sales_Type	3	Sal

Transformation:

Transformation Name	Rename columns
Parameter: Option	Keep from beginning (left)
Parameter: Column	Daily, POS_Cost, Sales_Type
Parameter: Number of characters	3

For example, if the old column name is `Sales_Type`, then based on the number of characters to keep from the beginning (left) is 3, then new column name is renamed to `Sal`.

Keep from end (right)

For the selected columns, you can specify the number of characters to keep from end (right) of the column names. Based on the number of characters you provide, the column name is updated. Example:

--	--	--

Old Column Names	Number of characters	New Column Names
Daily	4	aily
POS_Cost	4	Cost
Sales_Type	4	Type

Transformation:

Transformation Name	Rename columns
Parameter: Option	Keep from beginning (right)
Parameter: Column	Daily, POS_Cost, Sales_Type
Parameter: Number of characters	4

For example, if the old column name is `Sales_Type`, then based on the number of characters to keep from the end (right) is 4, then new column name is renamed to `Type`.

NOTE: If the number of characters are more than the length of the column names, then the whole name of the column is retained.

Find and replace

You can apply literals, Patterns, or regular expressions to match patterns of text in the source column names. These matching values can then be replaced by a fixed value.

Tip: The default behavior is to replace the first instance. Use the Match all occurrences checkbox to apply the pattern matching across all columns in your set.

For the selected columns, you can specify the number of characters to keep from end (right) of the column names. Based on the number of characters you provide, the column name is updated. Example:

Old Column Names	New Column Names
column1	Field1
column2	Field2
column3	Field3

Transformation:

Transformation Name	Rename columns
Parameter: Option	Find and replace
Parameter: Column	column1, column2, column3
Parameter: Find	'column'
Parameter: Replace with	'Field'

The above uses literal values for find and replace. For more information on pattern-based matching, see *Text Matching*.

Use row(s) as column names

When this method is applied, all of the values in the specified row or rows are used as the new names for each column.

NOTE: This method applies to all columns in the dataset.

Types:

Type	Description
Use a single row to rename columns	Specify the row number in the sample to use as the source for column names. NOTE: Source row number information must be available. See below.
Use the first row in the sample to rename columns	Use the first row in the sample as the name for all columns.
Combine multiple rows to rename columns	Specify two or more rows to combine into column names. Details are below. NOTE: Source row number information must be available. See below.

Source row number information:

NOTE: If source row number information is no longer available, this method cannot be used for column rename.

- If a value is not applied for the source row number, the next row of data is used.
- Source row numbers apply. Current row numbers may not be the same. In the data grid, mouse over the leftmost column to see available row information.
- Each value in the row or combination of values across rows must be unique within the set of new column names.
- The row is removed from its original position.
- If the product is unable to find unique multi-row headers for the column, the first row of the header set is used.

Combine multiple rows

The following transformation renames the columns in the dataset based on the values in rows 3 and 4 of the data:

Transformation Name	Rename columns
Parameter: Option	Use row(s) as column names
Parameter: Type	Combine multiple rows to name columns
Parameter: Row Numbers - row A	3
Parameter: Row Numbers - row B	4
Parameter: Choose your separator	'_'
Parameter: Fill across?	Selected

In the above:

- The separator is defined as an underscore character (_). This value can be empty.
- When Fill across is selected, if any row value is empty, the last non-empty value for the row in a previous column is used as part of the column header.

Sanitize Column Names

If needed, you can clean the names of the columns in your dataset. When column names are sanitized:

- alphanumeric characters and underscores (`_`) are permitted
- spacebars are converted to underscores
- all other characters are removed

Although Designer Cloud powered by Trifacta® Enterprise Edition supports a wider range of characters, you may wish to sanitize your column names to simplify publishing to and import into downstream systems.

Tip: Sanitized names matches the column names supported in Release 5.0 and earlier.

Sanitize during Import

The above sanitization can be applied to your column names when the dataset is imported.

Tip: If you notice issues with references to your column names in your recipes, you may be able to fix them by re-importing the dataset and choosing to sanitize during import.

Steps:

1. In the left nav bar, click the Datasets icon.
2. Click **Import Data**.
3. Select the file or table to import.
4. Click **Edit Settings**.
5. In the dialog, select **Remove special characters from column names**.
6. Complete the import of the dataset.

For more information, see *Import Data Page*.

Sanitize via Transformation

Through the Transform Builder, you can add a step to sanitize column names in your recipe.

Transformation Name	Rename by removing special characters
Parameter: Option	Clean current column names

Tip: If you are sanitizing your column names for downstream systems, you should add this step at the end of your recipe.

You can perform more fine-grained column renaming operations. See *Rename Columns*.

Change Column Data Type

Contents:

- *Change Type*
 - *Change from Column Menus*
 - *Change Data Type for Multiple Columns*
 - *Change Datetime Data Type*
 - *Via column menus*
 - *Via Transform Builder*
-

While transforming your data, you may need to change the data type of one or more columns. For example, data of String type may be the easiest to manipulate. Since there are no mismatched values for String data type, you may wish to change a column's data type to this baseline type.

- Data types that you see in the Transformer page represent types that are understood by the product.
- When data is imported from a separate datastore, Designer Cloud powered by Trifacta Enterprise Edition may apply internal data types to the data. These types may differ from the original data typing in the source. As needed, the inferring of data types can be disabled at the file, connection, or global level. For more information, see *Disable Type Inference*.
- When data is published from the product to a separate datastore, these types may be mapped to different data types in the target. For more information, see *Type Conversions*.

Tip: You can use the Change Column Type transformation to override the data type inferred for a column. However, if a new transformation step is added, the column data type is re-inferred, which may override your specific typing. You should consider applying Change Column Type transformations as late as possible in your recipes.

For more information on the available data types, see *Supported Data Types*.

Change Type

You can change a column's data type in one of the following ways:

Change from Column Menus

You can change the data type for individual columns through the following column menus:

1. To the left of the column name, you can click the icon and select a new data type from the list.

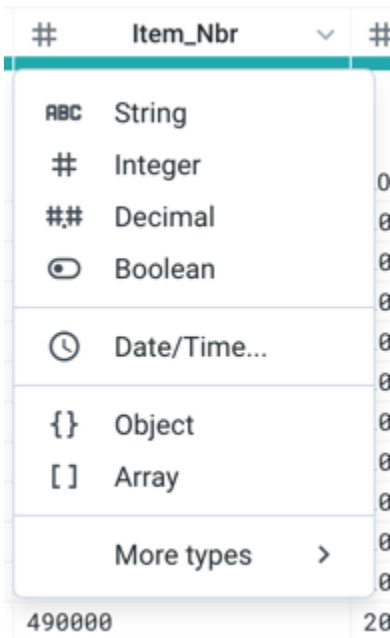


Figure: Column Data Type Menu

2. To the right of the column name, you can click the caret to open the column menu. Select **Change Type** and make a selection from the sub-menu.

Tip: Both of the above methods become individual steps in your recipe.

Change Data Type for Multiple Columns

If you must change the data type for multiple columns to a single data type, you can use a transformation like the following, which changes the columns `LastName`, `FirstName`, and `Address` to `String` data type.

Transformation Name	Change column type
Parameter: Column 1	LastName
Parameter: Column 2	FirstName
Parameter: Column 3	Address
Parameter: New Type	String

NOTE: When specifying a data type by name, you must use the internal value for the data type. The value in the column menu is the display name for the type.

For more information, see *Valid Data Type Strings*.

Change Datetime Data Type

If you are changing a column's data type to `Datetime`, you must also select a format string to apply to the column.

Via column menus

You can apply a Datetime data type through the column menus. When you choose the Datetime data type, you must apply a format for your Datetime values. For more information, see *Choose Datetime Format Dialog*.

Via Transform Builder

In the Transformer Builder, you can apply a specific transformation to format one or more columns to Datetime data type, using a specific format.

Tip: You can use the following transformation to change the format of a Datetime column.

This transformation looks like the following:

Transformation Name	Change column type
Parameter: Columns	Multiple
Parameter: Column 1	myDate
Parameter: New Type	Date/Time
Parameter: Date/time Type	month*dd*yyyy*hh:MMaX

For more information, see *Datetime Data Type*.

Copy and Paste Columns

You can cut, copy, and paste columns or column values in your dataset through the Column Browser panel or the column menus in the data grid.

NOTE: You cannot copy and paste columns between datasets.

Steps:

1. In the Column Browser or the data grid, select the column or columns for your source.
2. After you have selected one or more columns, from the column menu, select one of the following options:

Menu option	Description
Cut	Cut the column(s) to the clipboard. Selection is removed from the dataset temporarily. <div>NOTE: Cut operations do not add steps to your dataset. If you choose to do something other than pasting the column or its values, the source column is left untouched.</div>
Copy	Copy the column(s) to the clipboard.
Paste before	Paste the column(s) in the clipboard before the currently selected column in the dataset.
Paste after	Paste the column(s) in the clipboard after the currently selected column in the dataset.
Paste values only	Paste the values from the column(s) in the clipboard into the selected column(s). <div>NOTE: When values are pasted into the column, the column data type may be re-inferred.</div>

3. Select the column where you wish to move the columns or paste the values.

NOTE: Do not select multiple columns for multi-column pasting. You must select only one column. Multi-column operations are applied to the columns to the bottom/right of the selection.

4. From the column menu, select **Paste**:
 - a. **Paste before:** Paste cut or copied columns before the selected one.
 - b. **Paste after:** Paste column(s) after the selected one.
 - c. **Paste values:** Replace values in the selected column(s) with the values from the column(s) in the clipboard. The number of selected columns on the clipboard and in the selected target area must match. Data types do not have to match.

NOTE: When values are pasted into the column, the column data type may be re-inferred.

For more information, see *Column Menus*.

Create Column by Example

You can create a new column of data from an existing one by providing example values for the new column for values in the source column. With each successive example value, Transformation by Example (TBE) improves the quality of the output values, until you have the desired set of values for your newly generated column.

Limitations:

- Transformation by Example works best for text-based inputs. Non-text inputs are treated as String type by the feature.

NOTE: Multi-value inputs, such as Object or Array data types, must be converted to String data type prior to transformation by example.

- In the Transformer page, TBE is applied across the currently displayed sample. In the entire dataset, there may be outlier values that do not match any of the examples that you have provided.

Tip: If your column data is quite varied, you should collect additional samples to verify that your TBE is properly matching all values in the column.

For more information, see *Overview of TBE*.

Steps:

- In the Transformer page, locate the column to use as your source. From the column menu, select **Create column from examples**.
- In the Transform Builder, enter the new column name.
- In the following example, a new column called `zip` is being created from the `Addresses` column:

Source	Preview
Addresses	zip
1 1881 South Poplar, Santa Ana 92704-4321	92704-4321
2 7772 Chapman, Garden Grove, CA 92841	92841
3 1143 S Nakoma Dr, Santa Ana, CA 92784	92784
4 1398 E. McFadden Avenue, Santa Ana, CA 92785	92785
5 9821 Catherine Ave. Garden Grove, CA 92841	92841
6 9892 Woodbury Rd. Garden Grove, CA 92843	92843
7 688 S Jackson St, Santa Ana, CA 92784	92784
8 13222 Lewis Street, Garden Grove, CA 92843	92843
9 15328 Pickford St. Westminster, CA 92683	92683
10 11383 Sandstone Ave, Fountain Valley, CA 92788	92788
11 15791 Bushard Westminster, CA 92683	92683
12 880 Mustang Way, Calimesa, CA 92320	92320
13 32870 Avenue E, Yucaipa, CA 92399	92399
14 13908 Foster Ave, Baldwin Park, CA 91786	91786
15 811 E Bishop St, Santa Ana, CA 92781	92781
16 12820 Bess Street, Baldwin Park, CA 91786	91786
17 1345 W. 48th Street, San Bernardino, CA 92487	92487
18 5378 North H St., San Bernardino, CA 92487	92487
19 7351 Holder Street Buena, Park, CA 98628	98628
20 12521 Monroe, Garden Grove, CA 92841	92841
21 1488 West 11th Street, 92411-2132	92411-2132
22 13523 2nd Street, Yucaipa, CA 92399	92399
23 489 48th St, San Bernardino, CA 92411	92411

Figure: Selected column and first value is specified

- Double-click an empty cell in the Preview column to populate it with an example. In the above, the zip code from the first value has been entered into the Preview column: 92704-4321.

Tip: You can copy values from the source column and paste them into the Preview column.

- While many of the zip code values from other rows have been accurately populated, there are still some values that need fixing. In the following, you can see that one zip code was not properly extracted. Double-click in the Preview column for the third row and fix the value: 91935:

The screenshot shows the Data Editor interface. On the left, a table with columns 'Source', 'Address', '#', and 'Zip' is displayed. The third row has a zip code of 91935. On the right, a 'Create column from examples' panel is open, showing a grid with 'Address' and 'Zip' columns. The 'Zip' column has a value of 91935.

Figure: Populating multiple example rows improves the overall quality of transformation across all rows

- A quick scroll through the rest of the rows in the sample indicate that you have properly extracted the zip code values for all rows.
- Click **Add to Recipe**.
- The new **Zip** column is added to the dataset.

The screenshot shows the Data Editor interface with a table containing 48 rows and 6 columns. The columns are labeled 'column2', 'column3', 'column4', 'Address', '#', 'Zip', and 'column6'. The 'Zip' column has values ranging from 90.62k to 92.84k. The table is sorted by the 'Zip' column.

Figure: Transformed example column

For more information on previewing changes, see *Transform Preview*.

Remove Data

Contents:

- *Considerations when removing data*
- *Delete columns*
- *Delete rows*
 - *Delete rows based on selections*
 - *Filter rows based on matching conditions*
 - *Filter rows based on data type mismatches*
 - *Delete rows based on multiple blank cells*
- *Remove values*
 - *Using regular expressions*

Through simple selections, you can identify columns to remove, values on which to base row deletion, or strings to remove from your dataset. As needed, these transformations can be modified for more sophisticated removal transformations.

Considerations when removing data

Please keep in mind:

- When data is removed from your dataset, no actual deletion is performed.
 - Designer Cloud powered by Trifacta® Enterprise Edition does not modify source data. All recipe executions generate new sets of data based on the transformations you define, which are applied to a generated version of the source data.
 - Transformation steps are previewed and can be undone on sampled data in the Transformer page, so you should feel free to experiment with data removal.
- In large volume datasets, be careful applying patterns or regular expressions to your data. You should limit your application of these pattern-based changes to the minimum range of columns, rows, or strings required to complete the task.

Delete columns

To delete a column from your dataset, click the column drop-down and select **Delete**. The data is no longer available in the data grid or subsequent recipe steps.

Tip: To delete multiple columns, select them in the data grid or column browser. Then select **Delete** from the column menu.

Tip: To simply remove columns from display, use the **Hide** command. The hidden column still appears in the output.

Manual transformations:

To delete multiple columns, you can specify comma-separated column names in your Delete Columns transformation:

Transformation Name	Delete columns
Parameter: Columns	ColA, ColC, ColE

Parameter: Action	Delete selected columns
--------------------------	-------------------------

To delete a range of columns, use the tilde (~) character between the start and end column names:

Transformation Name	Delete columns
Parameter: Columns	ColA~ColE
Parameter: Action	Delete selected columns

For more information, see *Remove Data*.

Delete rows

You can delete rows in your dataset based on conditional patterns that you specify. The easiest method is to select a string in the appropriate column and then choose the Delete suggestion card.

Delete rows based on selections

Steps:

In the following example, each row contains an entry for a different business, and you want to remove all of the business entries from the city of Tempe.

1. In this case, you could use the column histogram to select the value `Tempe` in the `city` column, or you can use the Filters panel to filter for rows containing the value `Tempe`.
2. Then, select the Delete suggestion card.

The screenshot shows a data transformation interface. On the left, a 'Preview' panel displays a table with columns: RL, RBC, Address, RBC, Location_Description, and Season. The 'Address' column contains values like '300 E Orange Mall, Tempe, Maricopa, Arizona, 85281' and '1840 E Warner Rd, Tempe, Arizona, 85284'. The 'Location_Description' column contains 'Educational institution'. On the right, a configuration panel for 'Delete rows' is shown. It has a 'Keep rows' section with three options: 'values matching `Tempe` to NULL()', 'values matching `(alpha){5}` to NULL()', and 'values matching `(alpha)+` to NULL()'. The 'Delete rows' section has three options: 'with values matching `Tempe`', 'with values matching `(alpha){5}`', and 'with values matching `(alpha)+`'. The 'with values matching `Tempe`' option is selected. There are 'Edit' and 'Add' buttons next to the selected option, and a 'Cancel' button at the bottom.

Figure: Select Tempe in the City column to remove all entries for that city

3. After selecting `Delete`, the application evaluates your selected value and attempt your intention with the selection. Is it a string literal or a pattern? If it's a pattern, what does the pattern represent? You may select one of the variants in the Delete card to find the right match.

NOTE: Be sure to scroll up and down in the data grid to review the values that are affected. In some cases, your selection may turn into a pattern, which could apply to more than just the desired values. In the previous example, selecting `Tempe` may yield a matching pattern of `{alpha}{5}`, which would match any five-letter city name, including `Tempe`. Select other variants in the Delete card to change the matching pattern. Click **Edit** to review the matching string.

4. After defining and modifying your Filter Rows transformation, you can use the preview to see the rows that will be removed, prior to adding the transformation to your recipe.

Tip: You can also use the Filter Rows to retain rows based on a specified condition, effectively deleting the rows that do not match. See *Filter Data*.

Filter rows based on matching conditions

You can delete or keep rows in your dataset based on one or more matching conditions you define.

1. In the Search panel, enter `filter`.
2. Select the type of conditional. You can filter based on:
 - a. Type: missing or mismatched values.
 - b. Matches: literal or pattern matches that are exact matches, partial matches, or matches with the beginning or ending of column values.
 - c. Ranges: Less than (or equal to), greater than (or equal to), or combinations.
 - d. Custom formula: Specify an expression that evaluates to `true` or `false`. If `true`, then the data is filtered.
3. Specify the other parameters, including whether to delete or keep the matching rows.

For more information, see *Filter Data*.

Filter rows based on data type mismatches

You can delete or keep rows based on whether a cell value in the row matches a specified data type. The following example removes rows that do not match the `mm*dd*yy` format for the Datetime data type from the `transactionDate` column.

1. In the Search panel, enter `filter mismatched`.
2. Specify the following transformation:

Transformation Name	Filter mismatched
Parameter: Condition	Is mismatched
Parameter: Column	transactionDate
Parameter: Date/Time type	mm*dd*yy
Parameter: Action	Delete matching rows

3. Review the preview. If it looks good, add it to your recipe.

Delete rows based on multiple blank cells

If you have rows in your dataset that contain no data, you can use the following two steps to remove them. Assuming that you know the starting (`col1`) and ending (`colN`) column names of your dataset, try the following:

NOTE: If at a later time, you reorder or remove the starting or ending columns in a step before this one, these steps are broken.

Transformation Name	New formula
Parameter: Formula type	Single row formula
Parameter: Formula	<code>MERGE([column1~columnN])</code>
Parameter: New column name	<code>'all_blank_vals'</code>

Transformation Name	Delete rows when value is missing
---------------------	-----------------------------------

Parameter: Column	all_blank_vals
Parameter: Action	Delete selected columns

The above merges all values into a single value in the `all_blank_vals` column. The second step removes the row if the value in the merged column is blank.

Remember to delete the `all_blank_vals` column after you are done.

For more information, see *Filter Data*.

Remove values

To delete values from a column, select the values in the data grid. In the suggestion cards, select the **Replace** card. In the following example, the `city` column is removed of all values matching `Tempe`:

Transformation Name	Replace text or patterns
Parameter: Column	city
Parameter: Find	'Tempe'
Parameter: Replace with	' '
Parameter: Match all occurrences	true

The Replace transformation applies only to string values. The rest of a matching row is unaffected.

The above transformation matches all values in the column, even partial values, the match string is removed from the column value. For example, an entry `Tempest` would be turned into `st` if the above transformation was added.

To ensure that only full-column value matches are applied, you can add **Patterns** to indicate the start and end of the column value as in the following:

Transformation Name	Replace text or patterns
Parameter: Column	city
Parameter: Find	`{start}Tempe{end}`
Parameter: Replace with	' '
Parameter: Match all occurrences	true

In the above case, only values of `Tempe` that are the entire column value are matched. For more information on this pattern-based matching, see *Text Matching*.

Using regular expressions

For more sophisticated matching, you can apply regular expressions to your `replace` command. In the following example, all integers from 0-99 are matched in the `qty` column. Because there is no replacement value, they are deleted.

Regular expressions are very powerful pattern matching tools. You should be careful in your use of them. See *Text Matching*.

Character	Definition
^	Beginning of string. Required to prevent matching on the last digit of any numeric value.
\$	End of string. Required to prevent a 2-digit match on three-digit numbers.
\d	A single digit
	Logical or. In this case, it is used to define separate regexes for 1- and 2-digit values.

Deduplicate Data

Contents:

- *Validate Duplicate Data*
- *Remove duplicate rows transformation*
- *Deduplicate Rows Based on a Primary Key*
- *Deduplicate Columns*

As part of your data cleansing steps, you might need to remove duplicate rows of data from your dataset.

Validate Duplicate Data

In some cases, it might be acceptable to have duplicated data. For example, additional records using the same primary key might be included in a dataset as amendments or detail records.

NOTE: Before you remove duplicates from your dataset, you should verify that the data should not contain duplicates at all. If the data structure supports some duplicate elements including key values, you should exercise care in how you identify what constitutes duplicate information.

Remove duplicate rows transformation

Designer Cloud powered by Trifacta® Enterprise Edition provides a single transformation, which can remove identical rows from your dataset:

Tip: If you are attempting to identify if there are duplicate rows, check the row count in your dataset before and after you have added this transformation.

Transformation Name
Remove duplicate rows

Limitations:

- This transformation is case-sensitive. So, if a column has values `Hello` and `HELLO`, the rows containing those values are not considered duplicates and cannot be removed with this transformation.
- Whitespace and the beginning and ending of values is not ignored.

Before applying the `Remove deduplicate rows` transformation, you should attempt to normalize your data. You can use the following techniques to normalize a few columns of data.

NOTE: If you have more than 20 columns of data, you might be better served by trying to identify a primary key method for de-duplicating your dataset. Details are below.

For individual columns, you can use the `trim` function to remove leading and trailing whitespace:

NOTE: To preserve the original column values, use the `New formula` transformation. The `Edit column with formula` transformation replaces the original values.

Transformation Name
New formula

Parameter: Formula type	Single row formula
Parameter: Formula	TRIM(Item)

Since the `Remove deduplicate rows` transformation is case-sensitive, you can use the `LOWER` function to make the case of each entry in a column to be consistent:

Transformation Name	New formula
Parameter: Formula type	Single row formula
Parameter: Formula	LOWER(Description)

For more information, see *Normalize Numeric Values*.

Deduplicate Rows Based on a Primary Key

Another method to deduplicate data might be to delete rows based on one or more columns that you identify as a primary key for the dataset. A **primary key** is an identifier that uniquely identifies a row of data within a dataset. It can be a single field (column) or a combination of columns. For example, in a datasets of restaurant locations, the primary key can be a combination of RestaurantName, Address, and Zip.

NOTE: Before continuing, you must identify a primary key for your dataset. See *Generate Primary Keys*.

When you have identified your primary key, you should identify the appropriate method for your dataset. Please complete the following steps.

Steps:

1. If your primary key spans multiple columns, use the `Merge columns` transformation to bring the values into a single column:

Transformation Name	Merge columns
Parameter: Columns	RestaurantName,Address,Zip
Parameter: Separator	' - '

2. Rename the generated column: `PrimaryKey`.
3. Use the following transformation to generate a new column, comparing each value in the `PrimaryKey` column to the previous one:

Transformation Name	Window
Parameter: Formulas	PREV(PrimaryKey, 1)
Parameter: Order by	PrimaryKey

4. For each row, the value of the new column is the value in the `PrimaryKey` for the previous row. Now, test if this value is the same as the value in the `PrimaryKey` column for the current row:

Transformation Name	New formula
Parameter: Formula type	Single row formula

Parameter: Formula	IF((window==PrimaryKey),true,false)
Parameter: New column name	IsDupe

5. The new column (IsDupe) contains true for duplicate primary keys. Delete the rows that are duplicates:

Transformation Name	Delete rows
----------------------------	-------------

6. Delete any generated columns that are no longer needed.

Deduplicate Columns

While this form of duplicate data is rarer, you might want to check on the possibility of duplicate data between your columns. To check for duplicate column data, you can use a transformation similar to the following:

Transformation Name	New formula
Parameter: Formula type	Single row formula
Parameter: Formula	Column1 == Column2
Parameter: New column name	'dupeColVals'

In the generated column, values that are true indicate duplicate data. If all values are true, then you can remove one of the columns.

Compare Values

Contents:

- *Compare Numeric Values*
- *Compare Boolean Values*
- *Compare Date Values*
- *Compare String Values*

Depending on the data type, you can compare values in separate columns or single columns against fixed values.

Compare Numeric Values

You can use basic comparison operators to perform comparisons on your data. In this example, the `compareCol` column is generated as the evaluation of `3 < 6`, which is `true`:

Transformation Name	New formula
Parameter: Formula type	Single row formula
Parameter: Formula	(3 < 6)
Parameter: New column name	'compareCol '

For more information, see *Comparison Operators*.

Compare Boolean Values

Boolean values can be `true` or `false`, so comparisons like the following can be applied to a Boolean set of values:

Transformation Name	Edit column with formula
Parameter: Columns	Attendance
Parameter: Formula	IF(isSeated == true,true,Attendance)

In the above case, the value in `Attendance` is set to `true` if the value in the `isSeated` column is `true`. Otherwise, the current value in `Attendance` is used.

Compare Date Values

You can use the `DATEDIF` function to compare two date values, as in the following, which compares the number of days between `startCol` and `endCol` values:

NOTE: Both parameters of the `DATEDIF` function must be column references containing valid date values.

Transformation Name	New formula
Parameter: Formula type	Single row formula

Parameter: Formula	DATEDIF(startCol, endCol, 'day')
Parameter: New column name	'DurationInDays'

See *DATEDIF Function*.

Compare String Values

See *Compare Strings*.

Replace Cell Values

You can search and replace for specific values in a column.

If the column is known

Steps:

1. Select the column containing the value you wish to replace.
2. In the Selection Details panel on the right side, click the appropriate bar under Unique Values. All matching values are selected within the column.
3. Right-click the bar and select **Replace Values...**
4. A pre-configured transformation appears in the Transform Builder. Example:

Transformation Name	Replace cells
Parameter: Column	myDates
Parameter: Find	' 2013/02/07 '
Parameter: Replace with	' '

Tip: You can search for multiple items within the same column. Add other search values in additional Find textboxes.

5. Add your value in the Replace with textbox.
6. Click **Add**.
7. All matching cell values in the column are replaced with your entered value.

For more information, see *Selection Details Panel*.

If the column is unknown

Steps:

1. If you do not know the column, click the Filter tool in the Transformer bar.
2. Click the Rows tab and enter the value to locate. Only rows where the value appears are displayed in the data grid. Each instance of the matching value is highlighted.
3. Locate the column containing the specific highlighted values to replace. Select the value.
4. In the Suggestions panel, locate the Replace transformation card.
5. Select the variant of the Replace transformation that contains the specific value you selected. Then, click **Edit**.
6. A pre-configured transformation appears in the Transform Builder. Example:

Transformation Name	Replace text or patterns
Parameter: Column	myString
Parameter: Find	'Red'
Parameter: Replace	' '

7. Enter your replacement value in the Replace with textbox.

8. Click **Add**.
9. All matching values in the column are replaced with your entered value.

For more information, see *Filter Panel*.

Replace Values Using Patterns

Contents:

- *Replace Methods*
 - *Replace by selection*
 - *Replace using Transformer toolbar*
 - *Replace using Column Details panel*
 - *Replace using Transform Builder*
 - *Find Values in a Column*
 - *Examples*
 - *Replace first three characters*
 - *Replace using literal expressions*
 - *Replace string of four digits*
 - *Replace date and time patterns*
 - *Replace based on position*
 - *Replace alpha-numeric and position patterns*
 - *Replace using special patterns*
-

In Designer Cloud powered by Trifacta® Enterprise Edition, Trifacta patterns enable you to identify patterns in cell values and to perform replacements on those found elements of text. This section describes how to use patterns to find text and replace them with preferred values.

Tip: Patterns can also be used to extract values from cell values into a new column. The Trifacta patterns listed on this page can also be applied to the Extract text or pattern transformation. For additional example Trifacta patterns, see *Extract Values*.

- For more information, see *Overview of Pattern Matching*.
- For more information on Pattern syntax, see *Text Matching*.

Replace Methods

You can use the Replace text or patterns transformation to replace values in one or more columns with literal values, Trifacta patterns, or regular expressions through any of the following methods. You can use this transformation to replace missing, mismatched, or bad data using the following methods.

Replace by selection

When you select a piece of text in the data grid, the replace suggestion card displayed in the Selection Details panel on the right side may contain Pattern-based options for finding the selected value and similar values in the column of data. You can use these suggestions to replace column values.

Steps:

1. Select the data you want to replace. The suggestion cards are displayed.
2. In the Selection Details panel on the right side, select the Replace pattern suggestion card and click **Edit**.
3. The Replace text or patterns transformation is specified for you in the Transform Builder, where you can modify the Find value and other parameters as needed. See example below.

Replace using Transformer toolbar

In the Transformer toolbar at the top of the grid, click **Replace > Text or Pattern**. The Replace text or pattern transformation is displayed in the Transform Builder. For more information, see *Transformer Toolbar*.

For more information on procedures, see "Replace using Transform Builder" below.

Replace using Column Details panel

You can review sets of patterns for the selected column in the Column Details panel. When you select a column in the Column Details panel, you are prompted with a set of suggested patterns. For more information, see *Column Details Panel*.

- For more information on suggestions, see *Overview of Predictive Transformation*.

When a pattern suggestion is selected, it is specified in the Transform Builder for review and addition to your recipe. For more information, see "Replace using Transform Builder" below.

Replace using Transform Builder

In the Transform Builder, you can select one or more columns to replace text or patterns.

Steps:

The following steps describe how to build a pattern-based replacement transformation from scratch in the Transform Builder.

Tip: Some selections in the data grid or related tools can lead to suggestions or pre-configured transformations in the Transform Builder.

1. Enter `Replace text or pattern` in the Search panel. For more information, see *Search Panel*.
 2. Select an individual column or multiple columns from the following options:
 - **Multiple:** Select one or more columns from the drop-down list.
 - **All:** Select all columns in the dataset. See below for an example.
 - **Range:** Specify a start column and an ending column. All columns in between are selected.
 - **Advanced:** Specify the columns using a comma-separated list. You can combine multiple and range options under Advanced.
 - Ranges of columns can be specified using the tilde (~) character.
 - The following example range selects from the dataset as displayed in the data grid `column1`, `column3`, and the range of columns between `column5` and `column8`, inclusive:
- ```
column1,column3,column5~column8
```
3. In the Find text box, enter the text value or pattern that matches the value you want to replace. For more information, see "Find Values in a Column" below.
  4. In the Replace text box, enter the value to replace the found text.
  5. For additional controls, click **Advanced Options**:
    - a. **Start search after:** Enter a text or pattern that precedes the value you want to replace. See below example.
    - b. **Start search before:** Enter a text or pattern that follows the value you want to replace. See below example.
    - c. **Ignore case:** If selected, case is ignored when matching.
    - d. **Match all occurrences:** If selected, all occurrences of the found text in the column are matched and replaced.
  6. Click **Add**. The transformation is added to your recipe, and the selected columns are replaced with appropriate patterns in the data grid.

## Find Values in a Column

The `Replace with text or pattern` transformation enables you to replace values within the specified column or columns based on a string literal or Trifacta patterns. When you specify the transformation in the Transform Builder, the Find textbox can be populated with one of the following types of values:

| Find type          | Description                                                                                                                                                                                                                                                                                                                                                                                                    | Delimiter       | Example                                                                                                          |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|------------------------------------------------------------------------------------------------------------------|
| Literal            | A literal pieces of text                                                                                                                                                                                                                                                                                                                                                                                       | single quotes   | <code>&amp;apos;My piece of text&amp;apos;</code>                                                                |
| Trifacta pattern   | A Trifacta pattern represents zero or more characters that match a pattern. In Designer Cloud powered by Trifacta Enterprise Edition, Patterns are a simplified means of expressing regular expressions. For more information on Trifacta pattern syntax, see <i>Text Matching</i> .<br><br><b>Tip:</b> The examples in this section use Trifacta Patterns, which are simpler to use than regular expressions. | back-ticks      | <code>`{start}{digit}{3}`</code>                                                                                 |
| Regular expression | Regular expressions are a standard-based method of describing patterns in values.<br><br><b>NOTE:</b> Regular expressions are considered a developer-level skill. For more information on regular expression, see on <i>RE2</i> and <i>PCRE</i> regular expressions.                                                                                                                                           | forward slashes | <code>/^.{&lt;span class="hljs-number"&gt;0&lt;/span&gt;,&lt;span class="hljs-number"&gt;3&lt;/span&gt;}/</code> |

## Examples

The following examples demonstrate how Trifacta Patterns can be used to find and replace values within a column or set of columns.

### Replace first three characters

This example uses Trifacta Pattern to find the first three characters. In this example, the first three characters of the `Customer ID` column are replaced with the value `CustID-` for the selected column in the dataset.

#### Transformation:

|                                |                            |
|--------------------------------|----------------------------|
| <b>Transformation Name</b>     | Replace text or patterns   |
| <b>Parameter: Column</b>       | CustomerID                 |
| <b>Parameter: Find</b>         | <code>`{start}%{3}`</code> |
| <b>Parameter: Replace with</b> | <code>CustID-</code>       |

#### Results:

| Before   | After        |
|----------|--------------|
| Tri02468 | CustID-02468 |
| Mul2239  | CustID-2239  |
| Zev5521  | CustID-5521  |

## Replace using literal expressions

This example is based on the search and replace content in your dataset using literals. In the following example, the value `##CLT_NAME##` is replaced with `Our Customer, Inc.` across all columns in the dataset.

### Transformation:

|                                  |                          |
|----------------------------------|--------------------------|
| Transformation Name              | Replace text or patterns |
| Parameter: Column                | All                      |
| Parameter: Find                  | '##CLT_NAME##'           |
| Parameter: Replace with          | 'Our Customer, Inc.'     |
| Parameter: Match all occurrences | true                     |

## Replace string of four digits

**Tip:** For privacy reasons or sensitivity reasons, you can mask the sensitive data with the following replacements.

The following example uses Trifacta Patterns to find a string of four digits. The replacement is based on the structure of the data, not on the type of data. If you have data that are not credit card numbers yet follows the four-digit pattern, those values can also be replaced. In this example, the `myCreditCardNumbers` column is masked with `XXXX`.

### Transformation:

|                         |                                                                         |
|-------------------------|-------------------------------------------------------------------------|
| Transformation Name     | Replace text or patterns                                                |
| Parameter: Columns      | myCreditCardNumbers                                                     |
| Parameter: Find         | `{start}{digit}{4}{any}{digit}{4}{any}{digit}{4}{any}({digit}{4}){end}` |
| Parameter: Replace with | XXXX-XXXX-XXXX-\$1                                                      |

### Results:

| Before              | After               |
|---------------------|---------------------|
| 1234-1234-1234-1234 | XXXX-XXXX-XXXX-1234 |
| 1111-1111-1111-1111 | XXXX-XXXX-XXXX-1111 |
| 4321-4321-4321-4321 | XXXX-XXXX-XXXX-4321 |

## Using capture groups

The previous example captures aspects of the found pattern for use during replacement. A **capture group** is a mechanism in Trifacta Patterns or regular expressions to capture one or more parts of the matched values into variables.

In the example, the last four-digit segment of the Trifacta Pattern is surrounded by parentheses:

```
((digit){4}){end}
```

This group of digits is captured as the first (and only) capture group. In the replacement string, it is referenced as:

```
$1
```

You can have multiple capture groups in a single pattern. In the replacement, these capture groups can be referenced sequentially left-to-right from the pattern: \$1, \$2, and so on.

For more information, see *Capture Group References*. See below example.

**Tip:** You can use both {digit} and {#} Trifacta patterns for columns containing numeric values.

## Replace date and time patterns

The following example is based on replacing the date and time using the pre-configured suggestions displayed in the search context panel. In this example, the date Trifacta Patterns `yy/mm/dd` is replaced with `mm/dd/yy`.

### Transformation:

|                         |                                    |
|-------------------------|------------------------------------|
| Transformation Name     | Replace text or patterns           |
| Parameter: Column       | ORDER_DATE                         |
| Parameter: Find         | `({yy}){delim}({MM}){delim}({dd})` |
| Parameter: Replace with | \$2-\$1-\$3                        |

### Results:

| Before   | After    |
|----------|----------|
| 20/11/02 | 11/02/20 |
| 20/11/22 | 11/22/20 |
| 20/11/26 | 11/26/20 |

## Replace based on position

You can specify replacements based on the character position of values in your source column values. This method of finding and replacing values is useful if the source column data is consistently structured.

For example, suppose you have dates in the following format:

| Before     |
|------------|
| 2020-05-01 |
| 2020-05-02 |
| 2020-05-03 |

### Transformation:

Suppose you wanted to replace the value for the month with `Month`, you could add the following transformation step:

|                           |                           |
|---------------------------|---------------------------|
| Transformation Name       | Replace between positions |
| Parameter: Column         | Before                    |
| Parameter: Start position | 6                         |
| Parameter: End position   | 8                         |
| Parameter: Replace with   | Month                     |

#### Results:

| After         |
|---------------|
| 2020-Month-01 |
| 2020-Month-02 |
| 2020-Month-03 |

To replace the four digits of the year, you could perform a basic replace text or pattern transformation with a pattern to find of the following:

```
`{start}{digit}{4}`
```

#### Replace alpha-numeric and position patterns

You can use alpha-numeric and position Trifacta patterns for replacing the customer's address in the dataset. In this example, `{alpha-numeric}` pattern is applied to find the customer's addresses and used `{start}` and `{end}` pattern to mention the position of replacement. For more information on Pattern Syntax, see *Text Matching*.

#### Transformation:

|                               |                                  |
|-------------------------------|----------------------------------|
| Transformation Name           | Replace text or patterns         |
| Parameter: Column             | address_street_number            |
| Parameter: Find               | <code>`{alpha-numeric}`</code>   |
| Parameter: Replace with       | ##                               |
| Parameter: Start search after | <code>`{start}{digit}{2}`</code> |
| Parameter: Stop search before | <code>`{any}`</code>             |

#### Results:

| Before              | After               |
|---------------------|---------------------|
| 3298, Church Street | 32##, Church Street |
| 4132, Park Avenue   | 41##, Park Avenue   |
| 1234, McGrath Road  | 12##, McGrath Road  |

## Replace using special patterns

You can use the following special Trifacta Pattern tokens to search for matches in your source values. In some cases, these Trifacta Patterns are consistent with the patterns used for specific data types.

| Pattern                      | Description                                                                                                                                                                                                                         |
|------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>`{at-username}`</code> | Matches values that begin with an at-sign, such as <code>@trifacta</code> . This Trifacta Pattern can be useful if you need to remap or mask username values.                                                                       |
| <code>`{hashtag}`</code>     | Matches values that begin with a hashtag, such as <code>#dataprep</code> . For an example of this Trifacta Pattern, see <i>Extract Values</i> .                                                                                     |
| <code>`{hex}`</code>         | Matches values that are valid hexadecimal (base-16) numbers. These values contain a string of numerals, letters A-F, and combinations of them, without spaces. Examples: <code>AE00</code> , <code>1F2F</code> , <code>100</code> . |
| <code>`{phone}`</code>       | Matches valid phone numbers within a set of values. For more information on this data type pattern, see <i>Phone Number Data Type</i> .                                                                                             |
| <code>`{email}`</code>       | Matches valid email addresses within a set of values. For more information on this data type pattern, see <i>Email Address Data Type</i> .                                                                                          |
| <code>`{url}`</code>         | Matches valid URL addresses within a set of values. For more information, on this data type pattern, see <i>URL Data Type</i> .                                                                                                     |



# Replace Groups of Values

Contents:

- *Replacement methods*
- *Replace by selection*
- *Mask data*
  - *Delete whole column(s)*
  - *Masking all values*
  - *Partial masking of values*
  - *Mask multiple columns based on data type*
- *Replace with values from another column*
  - *Replace whole column*
  - *Replace partial values from another column*
- *Replace between positions*
- *Search and replace text or pattern*
- *Replace missing values*
  - *Replace missing with zeroes*
  - *Replace missing with average values*
- *Replace mismatched values*
- *Rename columns*

Whether data is missing, mismatched, or simply wrong, you can use a variety of methods in the Designer Cloud® application to replace values in one or more columns with literal values or pattern-based replacements.

## Replacement methods

In the Transformer page, you can use the following methods to replace values:

| Method                 | Description                                                                                                                                                                                                                                                                                                         |
|------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| By selection           | Select a value in the data grid to prompt a series of suggestions on what to do with the data. Typically, replacement options are near the top of the suggestions. <div><b>Tip:</b> You can replace specific values in a column with a preferred value. For more information, see <i>Replace Cell Values</i>.</div> |
| By column menu         | From the menu to the right of the column, select <b>Replace</b> and a sub-menu item to begin configuring a replacement transformation. See <i>Column Menus</i> .                                                                                                                                                    |
| By Transformer toolbar | At the top of the data grid, click the Replace icon in the Transformer toolbar to begin configuring replacements. See <i>Transformer Toolbar</i> .                                                                                                                                                                  |
| By Search panel        | In the Search panel, enter <code>replace</code> to build a replacement transformation from scratch. See <i>Search Panel</i> .                                                                                                                                                                                       |

## Replace by selection

When you select data in the data grid, the replacement suggestions are pre-specified for you, including a number of variants available in the suggestion card.

Notes:

- Suggestions are typically conservative in the scope of their changes. Case-sensitive searches and matching of the first occurrence only are the default settings.
- Order of listing of suggestions in a suggestion card:

- Pattern-based replacements are listed first. These replacements use `Patterns` , instead of regular expressions. Regular expressions can be more difficult to control.
- Literal value replacements are listed below the pattern-based ones.

For more information, see *Overview of Predictive Transformation*.

## Mask data

For privacy reasons or for sensitivity reasons, you may wish to mask sensitive data in one or more columns with fixed strings.

### Delete whole column(s)

If you need to remove the data in an entire column, the easiest method is to delete a column. Select one or more columns and then select **Delete** from the column drop-down. See *Remove Data*.

### Masking all values

You can use a transformation like the following to replace all values in a column with a simple string. In this case, the value `#REDACTED#` has been inserted in place of all values in the column.

**NOTE:** This replacement changes the data type of the column to String. If you must retain the original data type, the replacement value should be valid for the data type.

|                     |                          |
|---------------------|--------------------------|
| Transformation Name | Edit column with formula |
| Parameter: Columns  | transactionValue         |
| Parameter: Formula  | '#REDACTED#'             |

### Partial masking of values

Suppose you wish to partially mask data in a column. In the following example, data for the `AcctNum` column is masked, except for the last four characters (digits):

|                     |                                              |
|---------------------|----------------------------------------------|
| Transformation Name | Edit column with formula                     |
| Parameter: Columns  | AcctNum                                      |
| Parameter: Formula  | value: merge(['XXXX',right(AcctNum, 4)], '') |

### Mask multiple columns based on data type

You can use the following type of transformation to hide data based on data type. In this example, the values in all columns with Social Security Number (SSN) are replaced with a masking value: `XXX-XX-XXXX`:

**This method performs a simple text replacement of the data in the columns(s). After this transformation has been applied to the data, the source data is no longer available, unless you step back to a step before this one. For these kinds of operations, you may find it more secure to apply these kinds of masking operations to the source data in a single recipe and then make that output available to other users to use as an imported dataset.**

|                            |                                                   |
|----------------------------|---------------------------------------------------|
| <b>Transformation Name</b> | Edit column with formula                          |
| <b>Parameter: Columns</b>  | All                                               |
| <b>Parameter: Formula</b>  | if(isvalid(\$col, ['SSN']), 'XXX-XX-XXXX', \$col) |

## Replace with values from another column

### Replace whole column

You can do simple replacements of data from one column into another with transformations like the following. In this example, the values of `colB` are replaced with the values of `colA` with `0.15` added to them:

|                            |                   |
|----------------------------|-------------------|
| <b>Transformation Name</b> | Edit with formula |
| <b>Parameter: Columns</b>  | colB              |
| <b>Parameter: Formula</b>  | colA + 0.15       |

### Replace partial values from another column

You can use the `MERGE` function to blend full or partial sets of columns into a new column. In the following example, the `newBrandId` value is concatenated with the product code in the `ProdId` column to create a new product identifier:

|                            |                                            |
|----------------------------|--------------------------------------------|
| <b>Transformation Name</b> | Edit with formula                          |
| <b>Parameter: Columns</b>  | ProdId                                     |
| <b>Parameter: Formula</b>  | merge([newBrandId, right(prodId, 5)], '-') |

## Replace between positions

You can perform replacements based on character positions that you specify as part of the transformation.

- The beginning character value is specified as a number from 0, which starts on the left.
- The ending character value must be equal to or greater than the beginning character value.

In the following example, the `Whse_Name` column values are prepended with the value `old-`.

|                                  |                     |
|----------------------------------|---------------------|
| <b>Transformation Name</b>       | Replace by position |
| <b>Parameter: Column</b>         | Whse_name           |
| <b>Parameter: Start position</b> | 0                   |
| <b>Parameter: End position</b>   | 0                   |
| <b>Parameter: Replace with</b>   | old-                |

## Search and replace text or pattern

You can search and replace content in your dataset based on literals or patterns. In the following example, the value `##CLT_NAME##` is replaced with `Our Customer, Inc.` across all columns in the dataset:

|                            |                          |
|----------------------------|--------------------------|
| <b>Transformation Name</b> | Replace text or patterns |
|----------------------------|--------------------------|

|                                         |                      |
|-----------------------------------------|----------------------|
| <b>Parameter: Column</b>                | All                  |
| <b>Parameter: Find</b>                  | '##CLT_NAME##'       |
| <b>Parameter: Replace with</b>          | 'Our Customer, Inc.' |
| <b>Parameter: Match all occurrences</b> | true                 |

## Replace missing values

### Replace missing with zeroes

For numeric data, you may choose to replace values that are missing in a column with zeros. The following transformation sets missing values in the `Qty` and `DiscountPct` columns of Decimal data type to 0:

|                            |                                    |
|----------------------------|------------------------------------|
| <b>Transformation Name</b> | Edit column with formula           |
| <b>Parameter: Columns</b>  | Qty,DiscountPct                    |
| <b>Parameter: Formula</b>  | if(ismissing([\$col]), '0', \$col) |

### Replace missing with average values

One of the problems with the above method is that any statistical computations applied to the column are now affected by the zeroing of the missing values. For example, the computation for the `AVERAGE` function does not factor in missing values into the count of rows, which result in skewing of values for your purposes.

The following example creates a new column from the `DiscountPct` column in which empty values are inserted as the average of the values in the source column:

|                                   |                                                                 |
|-----------------------------------|-----------------------------------------------------------------|
| <b>Transformation Name</b>        | New formula                                                     |
| <b>Parameter: Formula type</b>    | Single row formula                                              |
| <b>Parameter: Formula</b>         | if(ismissing([DiscountPct]), average(DiscountPct), DiscountPct) |
| <b>Parameter: New column name</b> | DiscountPct-0toAVG                                              |

In this manner, the new column can be used for some statistical modeling, while preserving the original values in the original column.

## Replace mismatched values

You can perform replacements based on the values in a column that are mismatched against a specified type.

In the following example, Datetime values that do not match the `yyyy*mm*dd`, where the asterisk (\*) is a wildcard value.

|                                         |                           |
|-----------------------------------------|---------------------------|
| <b>Transformation Name</b>              | Replace mismatched values |
| <b>Parameter: Columns</b>               | Multiple                  |
| <b>Parameter: Column 1</b>              | myDate                    |
| <b>Parameter: Data type to evaluate</b> | Date/Time                 |

|                                  |                |
|----------------------------------|----------------|
| <b>Parameter: Date/Time type</b> | yyyy*mm*dd     |
| <b>Parameter: Replace with</b>   | Custom value   |
| <b>Parameter: New value</b>      | '##BAD_DATE##' |

**NOTE:** In the above example, the Date/Time type parameter applies only to replacements that are mismatched against the Date/Time data type. This parameter is used to specify the Datetime format against which the source values are validated. The parameter does not appear in Replace mismatched values transformations for other data types.

## Rename columns

For more information, see *Rename Columns*.

# Normalize Numeric Values

## Contents:

- *Numeric precision*
- *Standardize decimal precision*
- *Standardize units*
  - *Example - Fixed conversion factors*
  - *Dynamic conversion factors*
- *Adjust level of precision*
  - *Adjust data granularity by aggregation*

This section describes techniques to normalize numeric values in your datasets. Ideally, your source systems are configured to capture and deliver data using a consistent set of units in a standardized structure and format. In practice, data from multiple systems can illuminate differences in the level of precision used in numeric data or differences in text entries that reference the same thing. Within Designer Cloud powered by Trifacta® Enterprise Edition, you can use the following techniques to address some of the issues you might encounter in the standardization of units and values for numeric types.

## Numeric precision

In Designer Cloud powered by Trifacta Enterprise Edition, mathematical computations are performed using 64-bit floating point operations to 15 decimals of precision. However, due to rounding off, truncation, and other technical factors, small discrepancies in outputs can be expected. Example:

-636074.22

-2465086.34

Suppose you apply the following transformation:

|                                   |                              |
|-----------------------------------|------------------------------|
| <b>Transformation Name</b>        | New formula                  |
| <b>Parameter: Formula type</b>    | Single row formula           |
| <b>Parameter: Formula</b>         | ( -636074.22 + -2465086.34 ) |
| <b>Parameter: New column name</b> | MySum                        |

The expected output in the MySum column: -3101160.56

The actual output for in the MySum column: -3101160.5599999996

**NOTE:** For 64-bit floating point mathematical operations, deviations like the above are intrinsic to the Decimal data type and how the platform performs computations.

Depending on your precision requirements, you can manage precision across your columns using a transformation like the following, which rounds off MySum to three digits:

|                            |                          |
|----------------------------|--------------------------|
| <b>Transformation Name</b> | Edit column with formula |
| <b>Parameter: Columns</b>  | MySum                    |
|                            |                          |

|                    |                 |
|--------------------|-----------------|
| Parameter: Formula | ROUND(\$col, 3) |
|--------------------|-----------------|

For more information on floating point computations, see [https://en.wikipedia.org/wiki/Numeric\\_precision\\_in\\_Microsoft\\_Excel](https://en.wikipedia.org/wiki/Numeric_precision_in_Microsoft_Excel).

## Standardize decimal precision

If decimal values in a column are of varying levels of precision, you can standardize to a single level of precision.

### Steps:

1. From the column menu, select **Column Details**.
2. In the Column Details panel, select the Patterns tab. Among the patterns, select the following:

```
{digit}.{digit}
```

3. In the Suggestions panel on the right, locate the Edit Column transformation suggestion that uses the ROUND function. Click **Edit**.
4. Change the second parameter of the ROUND function to match the number of digits of precision.

You can generalize this formatting across multiple columns by applying the \$col reference in the transformation's function, as in the following:

|                     |                                            |
|---------------------|--------------------------------------------|
| Transformation Name | Edit column with formula                   |
| Parameter: Columns  | colA, colB, colC                           |
| Parameter: Formula  | IFVALID(\$col, ['Float'], ROUND(\$col, 2)) |

See *Column Details Panel*.

For more information, see *ROUND Function*.

## Standardize units

**Tip:** Each column that contains numeric values should have an identified unit of measurement. Ideally, this information is embedded in the name of the column data. If the unit of measurement is not included, it can be difficult to properly interpret the data.

Designer Cloud powered by Trifacta Enterprise Edition does not impose any units on imported data. For example, a column of values in floating point format could represent centimeters, ounces, or any other unit of measurement. As long as the data conforms to the specified data type for the column, then Designer Cloud powered by Trifacta Enterprise Edition can work with it.

However, this flexibility can present issues for users of the dataset. If data is not clearly labeled and converted to a standardized set of units, its users are forced to make assumptions about the data, which can lead to misuse of it.

**Tip:** The meaning of some units of measure can change over time. For example, a US Dollar in 2010 does not have the same value as a dollar in 2015. When you standardize shifting units of measure, you should account for any time-based differences, if possible.

## Example - Fixed conversion factors

In many cases, units can be converted to other units by applying a fixed conversion factor to a column of data. For example, your dataset has the following three columns of measured data:

| Person | Height_ft | Weight_kg | Arm_Length_in |
|--------|-----------|-----------|---------------|
| Jack   | 5'10"     | 92 kg     | 32            |
| Jill   | 5'2"      | 56 kg     | 29            |
| Joe    | 6'3"      | 101 kg    | 35            |

The above data has the following issues:

1. The Weight and Height columns contain unit identifiers, which forces the values to be treated as strings.
2. Metric data (kg) is mixed with English unit data (ft and in).
3. The Height data is non-numeric.

### Problem 1 - remove units

The `Weight_kg` column contains a unit identifier. On import, these values are treated as strings, which limits their use for analysis.

#### Steps:

1. In the data grid, select an instance of " kg". Note that the space should be selected, too.
2. Among the suggestion cards, select the Replace card.
3. It should automatically choose to replace with nothing, effectively deleting the content. To check, click **Modify**.
4. The transformation should look like the following:

|                                         |                          |
|-----------------------------------------|--------------------------|
| <b>Transformation Name</b>              | Replace text or patterns |
| <b>Parameter: Column</b>                | Weight_kg                |
| <b>Parameter: Find</b>                  | ' kg'                    |
| <b>Parameter: Replace with</b>          | ' '                      |
| <b>Parameter: Match all occurrences</b> | true                     |

5. Add it to your recipe.
6. Verify that the column's data type has been changed to `Integer` or `Decimal`, depending on the values in it.

### Problem 2 - convert English to metric units

To normalize to English units, the first issue is easily corrected by multiplying the Weight values by 2.2, since 1 kg = 2.2 lb:

|                            |                          |
|----------------------------|--------------------------|
| <b>Transformation Name</b> | Edit column with formula |
| <b>Parameter: Columns</b>  | Weight_kg                |
| <b>Parameter: Formula</b>  | (Weight_kg * 2.2)        |

If you want to round the value to the nearest integer, use the following:



|                            |                          |
|----------------------------|--------------------------|
| <b>Transformation Name</b> | Edit column with formula |
| <b>Parameter: Columns</b>  | Weight_kg                |
| <b>Parameter: Formula</b>  | ROUND((Weight_kg * 2.2)) |

After the above is added to the recipe, you should rename the column: Weight\_lbs.

### Problem 3 - convert ft/in to in

The final issue involves converting the Height\_ft values to a single value for inches, so that these values can be used consistently with the other columns in the dataset.

On import, your data for the column might actually look like the following:

| Height_ft |
|-----------|
| "5'10"    |
| "5'2"     |
| "6'3"     |

#### Steps:

1. Select the first quote mark in one of the entries.
2. In the suggestion cards, select the Replace card.
3. Select the variant that deletes all quotes in the column.
4. The full command should look like the following:

|                                         |                          |
|-----------------------------------------|--------------------------|
| <b>Transformation Name</b>              | Replace text or patterns |
| <b>Parameter: Column</b>                | Height_ft                |
| <b>Parameter: Find</b>                  | ` ``                     |
| <b>Parameter: Replace with</b>          | ''                       |
| <b>Parameter: Match all occurrences</b> | true                     |

5. Add it to your recipe.
6. The remaining steps compute the number of inches. Multiply the feet by 12, and then add the number of inches, using new columns of data.
7. Select the single quote mark, and choose the Split suggestion card. This transformation step should split the column into two columns: Height\_ft1 and Height\_ft2.
8. Derive the value in inches:

|                                   |                               |
|-----------------------------------|-------------------------------|
| <b>Transformation Name</b>        | New formula                   |
| <b>Parameter: Formula type</b>    | Single row formula            |
| <b>Parameter: Formula</b>         | ((Height_ft1 *12)+Height_ft2) |
| <b>Parameter: New column name</b> | Height_in                     |

9. You can delete the other, interim columns.

## Dynamic conversion factors

In some cases, the conversion rate between two different units of measures is dynamic. A common example involves mismatches between currency. For example, one dataset can be using U.S. dollars while another represents values in Euros.

### Within a column

If you have inconsistent units within a column, it might be possible to correct these values by applying a multiple. For example, you might be able to determine that some values are in kilometers, instead of meters, based on their much smaller values. Multiplying the kilometer values by 1000 should standardize your units. The following multiplies all values in the column `Distance` that are less than 1000 by 1000.

|                            |                                                                   |
|----------------------------|-------------------------------------------------------------------|
| <b>Transformation Name</b> | Edit column with formula                                          |
| <b>Parameter: Columns</b>  | Distance                                                          |
| <b>Parameter: Formula</b>  | <code>IF((Distance &lt; 1000, (Distance * 1000), Distance)</code> |

Note the implied assumption that there are no distances in kilometers that are over 1000.

**NOTE:** Inconsistency in units within a column indicates a problem in either the source data or how the column data was modified after import. Where possible, you should try to fix these issues in the source data first, as they can introduce problems when the data is used.

## Adjust level of precision

For numeric values that are used for measurement, you can adjust the level of precision within and across columns of values. For example, you have the following columns of data:

| Name     | Width_cm | Height_cm |
|----------|----------|-----------|
| Object 1 | 23.3     | 55.5512   |
| Object 2 | 65.2     | 102.4024  |
| Object 3 | 54.2     | 12.22     |

In the above, you can see the following precision mismatches:

- The Height column contains one value with only two digits of arithmetic precision in measurement.
- The Width column uses two digits of arithmetic precision, while the Height column contains more digits of precision.

Where precision in measurement is important, you should consider rounding to the lowest level of precision. In this case, within the Height column, that level is to two significant digits after the decimal point (e.g. 12.22). However, across all of the columns of the dataset, the level of precision is to one significant digit after the decimal point, as the Width values are all restricted to this level of precision. While you could choose to round off to four digits across all columns, the extra values of 0 do not accurately reflect measurement and are therefore misleading.

You can use the following transformations to perform rounding functions within these columns:

|                            |                                        |
|----------------------------|----------------------------------------|
| <b>Transformation Name</b> | Edit column with formula               |
| <b>Parameter: Columns</b>  | Width_cm                               |
| <b>Parameter: Formula</b>  | <code>NUMFORMAT(Width_cm '#.#')</code> |

|                            |                            |
|----------------------------|----------------------------|
| <b>Transformation Name</b> | Edit column with formula   |
| <b>Parameter: Columns</b>  | Height_cm                  |
| <b>Parameter: Formula</b>  | NUMFORMAT(Height_cm '#.#') |

**NOTE:** The above assumes that the number of significant digits remains fixed in the source data. If this varies over times or uses in your recipe, you might need to revisit these specific transformation steps.

**NOTE:** The above formatting option drops the zero for values like 4.0. As an alternative, you can use a format of '#.0', which always inserts a zero, even in cases where the zero is not present.

## Results:

| Name     | Width_cm | Height_cm |
|----------|----------|-----------|
| Object 1 | 23.3     | 55.5      |
| Object 2 | 65.2     | 102.4     |
| Object 3 | 54.2     | 12.2      |

## Adjust data granularity by aggregation

For data hierarchies, you can use aggregations to adjust the granularity of your data to the appropriate grouping level. For example, you want to join a dataset that is organized by individual products with a dataset that is organized by brand. In most cases, you should aggregate the product-level data in the first dataset to the brand level.

**NOTE:** When aggregation is applied, a new table of data is generated with the columns that you specifically select for inclusion.

For more information, see *Pivot Data*.

# Standardize Using Patterns

## Contents:

- *Example - Phone number patterns*
- *Generic Conversions*
- *Datetime Patterns*
- *Patterns by Example*

This section describes techniques to standardize values in your datasets using patterns. From the Column Details panel in the Designer Cloud® application, you can review and select patterns in the column's data. These selections can be used as the basis for converting all applicable values to the selected format.

**NOTE:** Pattern-based conversions can be applied to any data type.

In the Patterns tab, click the whitespace around a pattern and then review the Convert suggestion to define how the pattern matches can be converted to a single standardized format.

**Tip:** To select, click the whitespace around the pattern and example values.

**NOTE:** The application does not suggest pattern-based conversions that add or remove alphanumeric characters.

The screenshot shows the Designer Cloud application interface. The main panel is titled 'Patterns' and displays a list of patterns for the 'contractDate' column. The patterns are: 'dd / mm / yyyy' (12 matches), 'm / d / yyyy' (4 matches), and 'yyyy - m - dd' (4 matches). Each pattern is accompanied by example values. A right-hand 'Suggestions' panel is open, showing the 'Convert' task configuration for the selected 'dd / mm / yyyy' pattern. It lists example values '3/14/2018' and '2014-3-14', and shows the conversion to the pattern format '14/03/2018'. The 'Delete rows' section is also visible, showing a matching regex pattern. The 'Set' section is partially visible at the bottom.

**Figure: Selecting Datetime patterns in the Patterns tab**

In the above, the pattern block prompts suggestions for Convert tasks based on the selected patterns.

- Click **Edit** to modify the task.

- Click **Add** to add the task as a step to your recipe.

## Example - Phone number patterns

For columns containing phone number data, you can use the Patterns tab to standardize formatting options. Consider the following values, which are valid phone numbers. Next to each value is a pattern representing the value:

| PhoneNum       | Pattern                                              |
|----------------|------------------------------------------------------|
| (415) 555-1212 | \(((\{digit\}{3})\)\) (\{digit\}{3})\-(\{digit\}{4}) |
| 415-555-1212   | (\{digit\}{3})\-(\{digit\}{3})\-(\{digit\}{4})       |
| 415.555.1212   | (\{digit\}{3})\.\{digit\}{3}\.\{digit\}{4}           |
| 415 555-1212   | (\{digit\}{3}) (\{digit\}{3})\-(\{digit\}{4})        |
| 1+415-555-1212 | 1\+(\{digit\}{3})\-(\{digit\}{3})\-(\{digit\}{4})    |

In the Patterns tab, you can select the patterns to which you would like the other patterns in the same pattern group to be converted. Below, the selected **target pattern** becomes the pattern to which other patterns in the column values are converted:

The screenshot shows the Alteryx Patterns tab interface. The main area displays a list of patterns for the 'PHONE' column, with a progress bar indicating 20k rows. The patterns are grouped into three categories: 'All patterns' (16.07k), 'digit 3 - digit 3 - digit 4' (2.69k), and 'digit 4 - digit 3 - digit 4' (1.24k). The 'All patterns' group is selected, showing a list of phone numbers: (240)966-1418, (301)133-0554, (240)664-8565, (410)405-8808, and (301)205-3184. The sidebar on the right contains several options: 'Keep rows' (with values matching '(start)\(((\{digit\}{3})\)\{digit\}{3}\-(\{digit\}{4})\)(end)'), 'Convert' (values like '443 871 4409' to pattern format '443871-4409'), 'Delete rows' (with values matching '(start)\(((\{digit\}{3})\)\{digit\}{3}\-(\{digit\}{4})\)(end)'), 'Set' (values matching '(start)\(((\{digit\}{3})\)\{digit\}{3}\-(\{digit\}{4})\)(end)' to NULL), and 'Create a new column' (flag rows matching '(start)\(((\{digit\}{3})\)\{digit\}{3}\-(\{digit\}{4})\)(end)').

**NOTE:** You may have to modify the phone number values before attempting the conversion, as they may contain extra alphanumeric values. For example, international country codes (such as 044) or a

preceding 1+ required in long-distance numbers, may need to be extracted or removed from the column values prior to conversion.

## Generic Conversions

Below are types of conversions that are supported and not supported.

### Supported:

| Example Source Value | Example Target Value | Notes                                                              |
|----------------------|----------------------|--------------------------------------------------------------------|
| 123.456.7890         | 123-456-7890         | Changing symbolic characters                                       |
| (123) 456-7890       | 123 456-7890         | Removing symbolic characters                                       |
| (123)456-7890        | (123)-456-7890       | Adding symbolic characters                                         |
| 1234567890           | 123-456-7890         | Splitting a long character group and adding symbolic characters    |
| 123-456-7890         | 1234567890           | Merging multiple character groups and removing symbolic characters |

### Not supported:

| Example Source Value | Example Target Value | Notes                                                                                                  |
|----------------------|----------------------|--------------------------------------------------------------------------------------------------------|
| 123.456.7890         | +1.123.456.7890      | Adding a new character group                                                                           |
| +1.123.456.7890      | 123.456.7890         | Deleting a character group (alphanumeric characters cannot be deleted through pattern standardization) |
| Adam Wilson          | A Wilson             | Partial deletion of data from a character group                                                        |
| +1 (123) 456-7890    | +001 (123) 456-7890  | Prepending or appending a character group with specified characters                                    |

## Datetime Patterns

For columns of Datetime type, the available Convert mappings are based upon the supported date formats in the platform. Standardization of Datetime patterns is a specific implementation.

### Notes on Datetime patterns:

Two-digit years (YY) do not yield four-digit year (YYYY) suggestions due to ambiguity. For example, it is unclear if 50 should map to 1950 or 2050.

For performance reasons, a maximum of two semantic standardizations can be applied at once. Examples:

| Source Value | Possible Standardization | Semantic Mappings                                                                   | Status                     |
|--------------|--------------------------|-------------------------------------------------------------------------------------|----------------------------|
| Jan 1, 1981  | 01/01/1981               | <ul style="list-style-type: none"><li>Jan 01</li><li>1 01</li></ul>                 | ok (2 mappings)            |
| Jan 1, 1981  | 01/01/81                 | <ul style="list-style-type: none"><li>Jan 01</li><li>1 01</li><li>1981 81</li></ul> | Not suggested (3 mappings) |

For more information on supported formats, see *Datetime Data Type*.

For more information on converting Datetime values to a different format, see *DATEFORMAT Function*.

## Patterns by Example

You can generate a new column of values based on pattern matches from a source column. When you enter example values to match with source values, other values with similar patterns may also be matched based on your entered example value.

**Tip:** This method provides an easy way to build pattern-based matching for values in a source column.

For more information on transformation by example, see *Overview of TBE*.

# Modify String Values

## Contents:

- *Convert Columns to String*
    - *Available string functions*
  - *Example - Clean up Strings*
    - *Trim strings*
    - *Use missing or mismatched value presets*
    - *Remove a specific sub-string*
    - *Replace double spaces*
    - *Break out CamelCase*
    - *Reduce strings by words*
  - *Other String Cleanup Transformations*
    - *Trim whitespace from text*
    - *Remove whitespace*
    - *Remove symbols*
    - *Remove accents*
    - *Trim quotes*
  - *Pad Values*
    - *Add prefix or suffix to strings*
  - *Standardize String Values*
    - *Standardize case*
  - *Standardize String Lengths*
    - *Pad string values*
    - *Fixed length strings*
  - *Manage Sub-Strings*
  - *Compare Strings*
  - *Reset Types*
- 

This section describes techniques to standardize text values in your datasets. You can use the following techniques to address some common issues you might encounter in the standardization of text and other non-numeric values.

## Convert Columns to String

For manipulation of individual values, it is often easiest to work with the String data type, which is the most flexible. Depending on your approach, you may choose to convert some of your columns into String type:

|                            |                    |
|----------------------------|--------------------|
| <b>Transformation Name</b> | Change column type |
| <b>Parameter: Columns</b>  | col1,col2, col3    |
| <b>Parameter: New type</b> | 'String'           |

For more information, *Valid Data Type Strings*.

## Available string functions

You can edit values in a column by applying one of the available string functions. The following transformation can be modified for any of the available string functions:

|                            |                          |
|----------------------------|--------------------------|
| <b>Transformation Name</b> | Edit column with formula |
| <b>Parameter: Columns</b>  | myCol                    |



|                           |                         |
|---------------------------|-------------------------|
| <b>Parameter: Formula</b> | MyStringFunction(\$col) |
|---------------------------|-------------------------|

**Tip:** The \$col value allows you to reference the current column, which is particularly useful if your transformation is being applied across multiple columns.

For more information see *String Functions*.



## Example - Clean up Strings

In the following example, you can see that there are minor differences between the String values in each row of the dataset. These differences are captured in the Description column.

- Some characters, like tab, cannot be represented in this format.
- You can download this dataset: *Dataset-ExampleStrings.csv*.

| String          | Description                                     |
|-----------------|-------------------------------------------------|
| My String       | Base string: 'My String'                        |
| My String extra | Base string + ' extra'                          |
| My String       | A space in front of base string                 |
| My String       | A space after base string                       |
| MyString        | No space between the two words of base string   |
| My String       | Two spaces between the two words of base string |
| My String       | Base string + a tab character                   |
| My String       | Base string + a return character                |
| My String       | Base string + a newline character               |

When this data is imported, it looks like the following, after minor cleanup:

| ABC | String                                                                            | ABC | Description                                                                        |
|-----|-----------------------------------------------------------------------------------|-----|------------------------------------------------------------------------------------|
|     |  |     |  |
|     | 9 Categories                                                                      |     | 9 Categories                                                                       |
|     | My · String                                                                       |     | Base · string · : · "My · String"                                                  |
|     | My · String · extra                                                               |     | Base · string · + · " · extra"                                                     |
|     | · My · String                                                                     |     | A · space · in · front · of · base · string                                        |
|     | My · String ·                                                                     |     | A · space · after · base · string                                                  |
|     | MyString                                                                          |     | No · space · between · the · two · words · of · base · string                      |
|     | My · · String                                                                     |     | Two · spaces · between · the · two · words · of · base · string                    |
|     | My · String →                                                                     |     | Base · string · + · a · tab · character                                            |
|     | My · String ↵                                                                     |     | Base · string · + · a · return · character                                         |
|     | My · String \n                                                                    |     | Base · string · + · a · newline · character                                        |

**Figure: Example data after import**

**Notes:**

- You can see that white space is demarcated in the imported data. In particular, the line item with two spaces between the words is accurately represented in the data grid.
- Newlines, carriage returns, tabs, and other non-visible characters are represented with icons.

To normalize these text values, you can use some of the techniques listed on this page to match the problematic string values in this dataset and correct them, as needed. The sections below outline a number of techniques for identifying matches and cleaning up your data.

## Trim strings

**NOTE:** Before you begin matching data, you should perform a `TRIM` transform to remove whitespace at the beginning and end of the string, unless the whitespace is significant to the meaning and usage of the string data.

When transforming strings, a key step is to trim off the whitespace at the beginning and ending of the string. For the above dataset, you can use the following command to remove these whitespaces:

|                            |                          |
|----------------------------|--------------------------|
| <b>Transformation Name</b> | Edit column with formula |
| <b>Parameter: Columns</b>  | All                      |
| <b>Parameter: Formula</b>  | TRIM(\$col)              |

The above transform uses the following special values, which are available for some transforms like `set`:

| Special Value | Description                                                                                                                                                                                                 |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| *             | <p>For the Columns textbox under Advanced, you can use this wildcard to reference all columns in the dataset.</p> <div> <b>Tip:</b> You can also select <code>All</code> from the Columns drop-down. </div> |

\$col

When multiple columns are referenced in a transform, this special value allows you to reference the source column in a replacement value.

The previewed data looks like the following, in which five strings are modified and now match the base string:

| Source          | to be dropped   | Preview                                         | Description                                     |
|-----------------|-----------------|-------------------------------------------------|-------------------------------------------------|
| 9 Categories    | 9 Categories    | 9 Categories                                    | 9 Categories                                    |
| My String       | My String       | Base string: "My String"                        | Base string: "My String"                        |
| My String extra | My String extra | Base string + " extra"                          | Base string + " extra"                          |
| My String       | My String       | A space in front of base string                 | A space in front of base string                 |
| My String       | My String       | A space after base string                       | A space after base string                       |
| MyString        | MyString        | No space between the two words of base string   | No space between the two words of base string   |
| My String       | My String       | Two spaces between the two words of base string | Two spaces between the two words of base string |
| My String       | My String       | Base string + a tab character                   | Base string + a tab character                   |
| My String       | My String       | Base string + a return character                | Base string + a return character                |
| My String       | My String       | Base string + a newline character               | Base string + a newline character               |

**Figure: Trim data to improve matches**

To remove all whitespace, including spaces in between, you can use the `REMOVEWHITESPACE` function. See [REMOVEWHITESPACE Function](#).

## Use missing or mismatched value presets

The platform language, Wrangle, provides presets to identify missing or mismatched values in a selection of data.

**Tip:** In a column's histogram, click the missing or mismatched categories to trigger a set of suggestions.

**Missing values preset:** The following transform replaces missing URL values with the text string `http://www.example.com`. The preset `ISMISSING([Primary_WebSite_or_URL])` identifies the rows missing data in the specified column:

|                     |                                                                              |
|---------------------|------------------------------------------------------------------------------|
| Transformation Name | Edit column with formula                                                     |
| Parameter: Columns  | Primary_Website_or_URL                                                       |
| Parameter: Formula  | IF( ISMISSING( [Primary_Website_or_URL] ), 'http://www.example.com', \$col ) |

For more information, see [Find Missing Data](#).

**NOTE:** If the data type for the column is URL, then the replacement text string must be a valid URL, or the new data is registered as mismatched with the data type.

**Mismatched values preset:** This transform converts to 00000 all values in the `Zip` column that are mismatched against the `Zipcode` data type. In this case, the preset `ISMISMATCHED(Zip, ['Zipcode'])` identifies the mismatched values in the column, as compared to the `Zipcode` data type:

|                     |                                                                 |
|---------------------|-----------------------------------------------------------------|
| Transformation Name | Edit column with formula                                        |
| Parameter: Columns  | Zip                                                             |
| Parameter: Formula  | <code>IF(ISMISMATCHED(Zip, ['Zipcode']), '00000', \$col)</code> |

For more information, see *Find Bad Data*.

## Remove a specific sub-string

An entry in the example data contains an additional word: `My String extra`. You can use a simple replace command to remove it:

|                                  |                          |
|----------------------------------|--------------------------|
| Transformation Name              | Replace text or patterns |
| Parameter: Column                | String                   |
| Parameter: Find                  | <code>' extra '</code>   |
| Parameter: Replace with          | <code>''</code>          |
| Parameter: Match all occurrences | <code>true</code>        |

The `global` parameter causes the replacement to be applied to all instances found within a cell value. Otherwise, the replacement occurs only on the first instance.

## Replace double spaces

There are multiple ways of removing double spaces, or any pattern, from text values. For best results, you should limit this change to individual columns.

**NOTE:** For matching string patterns that are short in length, you should be careful to define the scope of match. For example, to remove double spaces from your dataset, you should limit the columns to just the ones containing string values. If you applied the change to all columns in the dataset, meaningful uses of double spacing could be corrupted, such as in JSON data fields.

|                                  |                          |
|----------------------------------|--------------------------|
| Transformation Name              | Replace text or patterns |
| Parameter: Column                | String                   |
| Parameter: Find                  | <code>' '</code>         |
| Parameter: Replace with          | <code>''</code>          |
| Parameter: Match all occurrences | <code>true</code>        |

- In the above, the Find term contains a string with two spaces in it.

**Tip:** If you wish to find two or more spaces, you can use the following Pattern in the Find parameter:

```
`()+`
```

- The Replace term contains no spaces.

## Break out CamelCase

CamelCase refers to text in which multiple words are joined together by removing the spaces between them. In the example data, the entry `MyString` is an example of CamelCase.

**NOTE:** Regular expressions are very powerful pattern-matching tools. If they are poorly specified in a transform, they can have unexpected results. Please use them with caution.

You can use `Patterns` to break up CamelCase entries in a column of values. The following transforms use regular expressions to identify patterns in a set of values:

|                                         |                          |
|-----------------------------------------|--------------------------|
| <b>Transformation Name</b>              | Replace text or patterns |
| <b>Parameter: Column</b>                | String                   |
| <b>Parameter: Find</b>                  | `({alpha})({upper})`     |
| <b>Parameter: Replace with</b>          | '\$1 \$2'                |
| <b>Parameter: Match all occurrences</b> | true                     |

The first transform locates all instances of uppercase letters followed by lower-case letters. Each instance is replaced by a space, followed by the found string (\$2). For more information, see *Text Matching*.

## Reduce strings by words

### Remove last word:

For example, you need to remove the last word of a string and the space before it. You can use the following `replace` transform to do that:

|                                         |                          |
|-----------------------------------------|--------------------------|
| <b>Transformation Name</b>              | Replace text or patterns |
| <b>Parameter: Column</b>                | String                   |
| <b>Parameter: Find</b>                  | ` {alpha}+{end}`         |
| <b>Parameter: Replace with</b>          | ''                       |
| <b>Parameter: Match all occurrences</b> | true                     |

When the above is previewed, however, you might notice that ending punctuation is not captured. For example, periods, exclamation points, and question marks at the end of your values are not captured in the `Pattern`. To capture those values, the Find parameter must be expanded:

|                            |                             |
|----------------------------|-----------------------------|
| <b>Transformation Name</b> | Replace text or patterns    |
| <b>Parameter: Column</b>   | String                      |
| <b>Parameter: Find</b>     | ` {alpha}+([?!;\\]) ){end}` |
|                            |                             |

|                                  |      |
|----------------------------------|------|
| Parameter: Replace with          | ' '  |
| Parameter: Match all occurrences | true |

In the second version, a capture group has been inserted in the middle of the `on` parameter value, as specified by the contents of the parentheses:

- The bracket-colon notation denotes a set of possible individual characters that could appear at this point in the pattern.
  - Note the backward slash before the right parenthesis in the capture group. This value is used to escape a value, so that this parenthesis is interpreted as another character, instead of the end of the capture group.
- The vertical pipe (`|`) denotes a logical OR, meaning that the specified individual characters could appear or the value after the vertical pipe.
- Since the value after the vertical pipe is missing, this capture group finds values with or without punctuation at the end of the line.
- A **capture group** is a method of grouping together sequences of characters as part of a matching pattern and then referencing them programmatically in any replacement value. For more information, see *Capture Group References*.

### Reduce total number of words:

You need to cut each value in a column down to a maximum of two words. You can use the following to identify the first two words using capture groups in a `Pattern` and then write that pattern back out, dropping the remainder of the column value:

|                                  |                                              |
|----------------------------------|----------------------------------------------|
| Transformation Name              | Replace text or patterns                     |
| Parameter: Column                | String                                       |
| Parameter: Find                  | `{start}({alpha}* )({alpha}*) ({any}*{end})` |
| Parameter: Replace with          | '\$1\$2'                                     |
| Parameter: Match all occurrences | true                                         |

For the Find pattern:

- The `start` pattern identifies the start of each value in the `String` column.
- The two `alpha` capture groups identify the first two words in the string. Note that the space after the second capture group is specified outside of the capture group; if it was part of the capture group, a trailing space is written in the replacement value.
- The final capture group identifies the remainder of the value in the cell.
  - `any` captures any single character.
  - The wildcard asterisk captures all values between the `any` character and the `end` of the value.

## Other String Cleanup Transformations

### Trim whitespace from text

You can trim out whitespace from an individual column via transformation. The `TRIM` function applied to string values removes the leading and trailing whitespace:

|                     |                          |
|---------------------|--------------------------|
| Transformation Name | Edit column with formula |
| Parameter: Columns  | myCol                    |
|                     |                          |

|                           |             |
|---------------------------|-------------|
| <b>Parameter: Formula</b> | TRIM(myCol) |
|---------------------------|-------------|

To apply this function across all columns in the dataset, you can use the following:

|                            |                          |
|----------------------------|--------------------------|
| <b>Transformation Name</b> | Edit column with formula |
| <b>Parameter: Columns</b>  | All                      |
| <b>Parameter: Formula</b>  | TRIM(\$col)              |

#### Notes:

- Instead of All above, you can use the asterisk (\*) **wildcard**, which represents all possible value. In this case, both values for Columns matches with all column names in the dataset.
- You may need to move columns or use range values to apply this transformation to only non-numeric column types.
- The \$col entry denotes a reference to the current column. So for any column to which this transformation is applied, the source values are pulled from the column itself and then trimmed.

In some cases, you may wish to remove all spaces, including those in between words or digits, in your strings:

|                            |                          |
|----------------------------|--------------------------|
| <b>Transformation Name</b> | Edit column with formula |
| <b>Parameter: Columns</b>  | All                      |
| <b>Parameter: Formula</b>  | REMOVEWHITESPACE(\$col)  |

#### Remove whitespace

If needed, you can remove all whitespace from a column of values.

**NOTE:** This transformation differs from the TRIM function, which removes only the whitespace at the beginning and end of the string. This transformation removes all whitespace, including space in the middle of the string.

**Tip:** For some of the string comparison functions, you may achieve better results by comparing strings without whitespace.

|                            |                       |
|----------------------------|-----------------------|
| <b>Transformation Name</b> | Remove whitespace     |
| <b>Parameter: Columns</b>  | name                  |
| <b>Parameter: Format</b>   | Remove all whitespace |

#### Remove symbols

The following transformation removes all non-alphanumeric symbols from your string values, including:

- Punctuation
- Numeric value indicators (\$, %, etc.)

**NOTE:** Accented characters may not be removed. If this function fails to remove specific symbols, you may need to remove these symbols manually or change the input encoding on the dataset. For more information, see *Import Data Page*.

|                            |                |
|----------------------------|----------------|
| <b>Transformation Name</b> | Remove symbols |
| <b>Parameter: Columns</b>  | All            |
| <b>Parameter: Format</b>   | Remove symbols |

## Remove accents

The following transformation converts all accented characters (e.g. "ä") to unaccented characters (e.g. "a").

|                            |                          |
|----------------------------|--------------------------|
| <b>Transformation Name</b> | Remove accents from text |
| <b>Parameter: Columns</b>  | All                      |
| <b>Parameter: Format</b>   | Remove accents           |

## Trim quotes

When some files are imported into the application, leading and trailing quotes may remain for some or all columns. You can use the following transformation to remove these quotes from all columns:

**NOTE:** Quotes that appear in the middle of the string value are not removed. Single quotes, such as apostrophes, are not removed.

|                            |                                  |
|----------------------------|----------------------------------|
| <b>Transformation Name</b> | Trim quotes                      |
| <b>Parameter: Columns</b>  | All                              |
| <b>Parameter: Format</b>   | Trim leading and trailing quotes |

## Pad Values

### Add prefix or suffix to strings

You can add fixed-string prefixes or suffixes to your string values. The following adds -0000 to a text version of the Zipcode column:

|                               |                    |
|-------------------------------|--------------------|
| <b>Transformation Name</b>    | Add suffix to text |
| <b>Parameter: Columns</b>     | txtZipCode         |
| <b>Parameter: Format</b>      | Add suffix         |
| <b>Parameter: Text to add</b> | '-0000'            |

## Standardize String Values

### Standardize case

You can use the following steps to set all text values in a column to be the same case.

#### Lower case:

|                            |                          |
|----------------------------|--------------------------|
| <b>Transformation Name</b> | Edit column with formula |
| <b>Parameter: Columns</b>  | myStrings                |



|                           |                  |
|---------------------------|------------------|
| <b>Parameter: Formula</b> | LOWER(myStrings) |
|---------------------------|------------------|

#### Upper case:

|                            |                          |
|----------------------------|--------------------------|
| <b>Transformation Name</b> | Edit column with formula |
| <b>Parameter: Columns</b>  | myStrings                |
| <b>Parameter: Formula</b>  | UPPER(myStrings)         |

#### Proper (sentence) case:

|                            |                          |
|----------------------------|--------------------------|
| <b>Transformation Name</b> | Edit column with formula |
| <b>Parameter: Columns</b>  | myStrings                |
| <b>Parameter: Formula</b>  | PROPER(myStrings)        |

### Standardize String Lengths

#### Pad string values

If you need all of your column values to be of the same length, one technique is to pad each string value at the front sufficiently, such that all string lengths in the column are identical.

This transformation results in adding enough spaces to each row value until the length of each value is 50 characters.

**NOTE:** Strings that are longer than the prescribed maximum are unchanged. You can use the `LEFT` or `RIGHT` functions to change the size of the oversized ones. See below.

|                                         |                                  |
|-----------------------------------------|----------------------------------|
| <b>Transformation Name</b>              | Pad text with leading characters |
| <b>Parameter: Columns</b>               | MyStrings                        |
| <b>Parameter: Format</b>                | Pad with leading characters      |
| <b>Parameter: Character to pad with</b> | ' '                              |
| <b>Parameter: Length</b>                | 50                               |

#### Fixed length strings

You can limit the maximum size of a column or set of columns to a fixed string length. For example:

|                            |                                           |
|----------------------------|-------------------------------------------|
| <b>Transformation Name</b> | Edit column with formula                  |
| <b>Parameter: Columns</b>  | col1,col2                                 |
| <b>Parameter: Formula</b>  | IF(LENGTH(\$col)>32,LEFT(\$col,32),\$col) |

In the above, if the length of either column is longer than 32 characters, then the column value is set to the leftmost 32 characters. For shorter strings, the entire string is used.

For more information, see *Manage String Lengths*.

## Manage Sub-Strings

You can use the following functions to locate values within your strings. These functions can be used as part of New Formula or Edit Formula transformations to create or edit column content:

| Function Name              | Description                                                                                                                                                                                     |
|----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>LEN Function</i>        | Returns the number of characters in a specified string. String value can be a column reference or string literal.                                                                               |
| <i>FIND Function</i>       | Returns the index value in the input string where a specified matching string is located in provided column, string literal, or function returning a string. Search is conducted left-to-right. |
| <i>RIGHTFIND Function</i>  | Returns the index value in the input string where the last instance of a matching string is located. Search is conducted right-to-left.                                                         |
| <i>LEFT Function</i>       | Matches the leftmost set of characters in a string, as specified by parameter. The string can be specified as a column reference or a string literal.                                           |
| <i>RIGHT Function</i>      | Matches the right set of characters in a string, as specified by parameter. The string can be specified as a column reference or a string literal.                                              |
| <i>SUBSTRING Function</i>  | Matches some or all of a string, based on the user-defined starting and ending index values within the string.                                                                                  |
| <i>SUBSTITUTE Function</i> | Replaces found string literal or pattern or column with a string, column, or function returning strings.                                                                                        |

## Compare Strings

The application supports multiple methods of comparing strings. For more information, see *Compare Strings*.

## Reset Types

After modifying non-text values as strings, remember to convert them back to their original types.

# Manage String Lengths

## Contents:

- *Test String Length*
- *Truncate Strings*
- *Specialized String Lengths*
- *Use Rightmost Values*
- *Substring Values*
- *Additional String Functions*

In this example, your target system has a limit on the maximum length for the First Name and Last Name fields. You can use the following transforms to evaluate and truncate your strings based on their length.

## Test String Length

You can use the following command to write a `TOO LONG` message when the length of the `first_name` field exceeds 32 characters:

|                            |                                                  |
|----------------------------|--------------------------------------------------|
| <b>Transformation Name</b> | Edit column with formula                         |
| <b>Parameter: Columns</b>  | String_test                                      |
| <b>Parameter: Formula</b>  | IF(LEN(first_name) > 32, 'TOO LONG',String_test) |

## Truncate Strings

The above test allows you to evaluate individual strings that are too long to see if they are errors or can somehow be shortened. For a large dataset in which you cannot easily solve these problems, you can simply choose to cut off the length of a string at 32 characters:

|                            |                          |
|----------------------------|--------------------------|
| <b>Transformation Name</b> | Edit column with formula |
| <b>Parameter: Columns</b>  | *                        |
| <b>Parameter: Formula</b>  | LEFT(\$col,32)           |

In the above, you can use a wildcard to match all columns in the dataset. The replacement value is defined to be the first 32 characters of the source column (`$col`). By definition of the `LEFT` function, columns that are shorter than 32 characters in length are untouched.

**Tip:** If the field you are truncating is used as a key to your dataset, you should verify that your key still contains unique values after you have applied the truncation. For example, if the combination of `first_name` and `last_name` is a unique identifier in your dataset, you should verify that the column containing these identifiers contains unique values.

## Specialized String Lengths

In some cases, you might want to limit the lengths of text strings. In this example, your dataset contains a column of zip code values, some of which are in Zip+4 format. Your source data might look like the following:

zip\_code

|            |
|------------|
| 94104      |
| 94104-2218 |
| 94105      |

For consistency, you might want to limit the column to use just the first five digits of the zip code.

### Steps:

1. Select the first five digits of one of the nine-digit zip codes.
2. In the suggestion cards, select the Extract card.
3. Select the following variation:

|                                          |                         |
|------------------------------------------|-------------------------|
| <b>Transformation Name</b>               | Extract text or pattern |
| <b>Parameter: Column to extract from</b> | zipcode                 |
| <b>Parameter: Option</b>                 | Custom text or pattern  |
| <b>Parameter: Text to extract</b>        | `{zip}`                 |
| <b>Parameter: Start extracting after</b> | `{start}`               |

4. Click **Add**.

The above solution references two Patterns to identify elements of the cell value. For more information, see *Text Matching*.

For a more generalized approach, you can use some of the following string functions to limit your data length. Values that are shorter than the designated string length are left untouched.

**NOTE:** Transforms that cut down the size of a value might generate mismatched or missing values based on the column's data type. You should verify that you are not creating new missing or mismatched values.

## Use Rightmost Values

Use the following transform to reduce a string to the rightmost 6 characters in any value:

|                            |                          |
|----------------------------|--------------------------|
| <b>Transformation Name</b> | Edit column with formula |
| <b>Parameter: Columns</b>  | prodID                   |
| <b>Parameter: Formula</b>  | RIGHT(prodID, 6)         |

## Substring Values

The `SUBSTRING` function enables you to designate a specific subset of the string's characters to use. You specify the index of the first character in the values and the number of subsequent characters to include. For example, when applied to the value `United States of America` in the `countries` column, the following transform sets the new value to be `States`.

|                            |                          |
|----------------------------|--------------------------|
| <b>Transformation Name</b> | Edit column with formula |
| <b>Parameter: Columns</b>  | countries                |

|                           |                                     |
|---------------------------|-------------------------------------|
| <b>Parameter: Formula</b> | SUBSTRING( <i>countries</i> , 7, 6) |
|---------------------------|-------------------------------------|

Note that the index value begins at zero; to extract from the beginning of the value, replace 7 above with 0.

## Additional String Functions

Wrangle supports other functions, which can be used to transform string values. See *String Functions*.

# Extract Values

## Contents:

- *Extract vs. Split*
  - *Extract methods*
  - *Extract text or patterns*
    - *Extract single values*
    - *Extract values by example*
    - *Constrain matching*
    - *Extract single patterns*
    - *Extract multiple values*
    - *Extract first or last characters*
    - *Extract by positions*
  - *Extract by Data Type*
    - *Extract date values*
    - *Extract numeric values*
    - *Extract components of a URL*
    - *Extract object values*
    - *Extract array values*
  - *Extract Values into a List*
    - *Extract matches into array*
    - *Extract hashtags*
- 

Extracting one or more values from within a column of values can turn data into meaningful and discrete information. This section describes how to extract column data, the methods for which may vary depending on the data type.

## Extract vs. Split

Extract and split transformations do not do the same thing:

- A **split** transformation separates a single column into one or more separate columns based on one or more values in the source column that identify where the data should be split. These delimiters can be determined by the application or specified by the user when defining the transformation.
- An **extract** transformation matches literal or pattern values from a source column and stores it in a separate column.

**NOTE:** The source column is untouched by extract transformations.

## Extract methods

In the Transformer page, you can use the following methods to extract values:

| Method         | Description                                                                                                                                                                                            |
|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| By selection   | Select part of a value in the data grid to prompt a series of suggestions on what to do with the data. Typically, extract options are near the top of the suggestions when you select part of a value. |
| By column menu | From the menu to the right of the column, select <b>Extract</b> and a sub-menu item to begin configuring a transformation. See <i>Column Menus</i> .                                                   |
| By             | At the top of the data grid, click the Extract icon in the Transformer toolbar to begin configuring extract transformations. See                                                                       |

|                     |                                                                                                                   |
|---------------------|-------------------------------------------------------------------------------------------------------------------|
| Transformer toolbar | <i>Transformer Toolbar.</i>                                                                                       |
| By Search panel     | In the Search panel, enter <code>extract</code> to build a transformation from scratch. See <i>Search Panel</i> . |

## Extract text or patterns

A primary use of extraction is to remove literal or patterned values of text from a column of values. Suppose your dataset included a column of LinkedIn updates. You can use one of the following methods to extract keywords from these values.

### Extract single values

The following example transformation extracts the word `#bigdata` from the column `msg_LinkedIn`:

|                                                |                           |
|------------------------------------------------|---------------------------|
| <b>Transformation Name</b>                     | Extract text or pattern   |
| <b>Parameter: Column to extract from</b>       | <code>msg_LinkedIn</code> |
| <b>Parameter: Option</b>                       | Custom text or pattern    |
| <b>Parameter: Text to extract</b>              | <code>'#bigdata'</code>   |
| <b>Parameter: Number of matches to extract</b> | 1                         |

#### Notes:

- The `option` parameter identifies that the pattern to match is a custom one specified by the user.
- The `Number of matches to extract` parameter defaults to 1, meaning that the transformation extracts a maximum of one value from each cell. This value can be set from 1-50.

### Extract values by example

You can generate a new column of values extracted from a source column by entering example values to match with source values. Values with similar patterns may also be matched based on your entered example value.

**Tip:** This method provides an easy way to build pattern-based matching for values in a source column.

For more information on transformation by example, see *Overview of TBE*.

### Constrain matching

Within the extract transformation, you can specify literals or patterns before or after which the match is found. This method can be used to remove parts of each cell value from erroneously matching on the literal or pattern that is desired.

The following example extracts the second three-digit element of a phone number, skipping the area code:

|                                          |                         |
|------------------------------------------|-------------------------|
| <b>Transformation Name</b>               | Extract text or pattern |
| <b>Parameter: Column to extract from</b> | <code>phone_num</code>  |
| <b>Parameter: Option</b>                 | Custom text or pattern  |
| <b>Parameter: Text to extract</b>        | <code>`{digit}`</code>  |

|                                         |                       |
|-----------------------------------------|-----------------------|
| Parameter: Number of matches to extract | 1                     |
| Parameter: Ignore matches between       | `{start}{digit}{3}\-` |

## Extract single patterns

You can also do pattern-based extractions using Trifacta patterns or regular expressions.

- **Regular expressions** are a standards-based method of describing patterns of characters for matching purposes. Regular expressions are very powerful but can be difficult to use.
- A **Trifacta pattern** is a proprietary method of describing patterns, which is much simpler to use than regular expressions.
- For more information on both types of patterns, see *Text Matching*.

The following example extracts all words that begin with # in the `msg_LinkedIn` column:

|                                         |                                         |
|-----------------------------------------|-----------------------------------------|
| Transformation Name                     | Extract text or pattern                 |
| Parameter: Column to extract from       | <code>msg_LinkedIn</code>               |
| Parameter: Option                       | Custom text or pattern                  |
| Parameter: Text to extract              | <code>`\#{alphanum-underscore}+`</code> |
| Parameter: Number of matches to extract | 50                                      |

### Notes:

- The `Text to extract` parameter has changed:

| Element                            | Description                                                                                                                                                            |
|------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Two back-ticks (`)                 | Indicate that the expression between them represents a Trifacta pattern.                                                                                               |
| <code>\#</code>                    | The slash indicates that the character right after it should be interpreted as a character only; it should not be interpreted as any special character in the pattern. |
| <code>{alphanum-underscore}</code> | This Trifacta pattern element is used to indicate a single alphanumeric or underscore character.                                                                       |
| <code>+</code>                     | Adding the plus sign after the above character signifies that the pattern can match on a sequence of alphanumeric or underscore characters of one or more length.      |

- The `Number of matches to extract` parameter has been increased to grab up to 50 hashtags.

## Advanced options

| Option                        | Description                                                                                                                                                                                                                                                                                               |
|-------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Number of patterns to extract | <p>Set this value to the total number of patterns you wish to extract.</p> <div> <p><b>NOTE:</b> This value determines the number of columns that are generated by the extraction. If no value is available, an empty value is written into the corresponding column.</p> </div> <p>The default is 1.</p> |
| Ignore case                   | By default, pattern matching is case-sensitive. Select this checkbox to ignore case when matching.                                                                                                                                                                                                        |
| Ignore matches                | You can enter a pattern here to describe any patterns that should not be part of any match. This option is useful if you                                                                                                                                                                                  |



between | have multiple instances of text but want to ignore the first one, for example.

## Extract multiple values

In your pattern expressions, you can use the vertical pipe character (|) to define multiple patterns to find. The following example extracts any value from the `myDate` column that ends in 7 or 8:

|                                   |                         |
|-----------------------------------|-------------------------|
| Transformation Name               | Extract text or pattern |
| Parameter: Column to extract from | myDate                  |
| Parameter: Text to extract        | `{any}+7 {any}+8`       |
| Parameter: End extracting before  | `{end}`                 |

You can use the vertical pipe in both Trifacta patterns and regular expressions.

## Extract first or last characters

You can extract the first or last set of characters from a column into a new column. In the following example, the first five characters from the `ProductName` column are extracted into a new product identifier column:

|                                            |                      |
|--------------------------------------------|----------------------|
| Transformation Name                        | Extract by positions |
| Parameter: Column to extract from          | ProductName          |
| Parameter: Option                          | First characters     |
| Parameter: Number of characters to extract | 5                    |

You can change the Option value to `Last characters` to extract from the right side of the column value.

## Extract and remove

If you need to remove the characters that you extracted, you can use the following transformation. In this case, the first five characters, which were extracted in the previous transformation, are removed:

|                     |                                        |
|---------------------|----------------------------------------|
| Transformation Name | Edit column with formula               |
| Parameter: Columns  | ProductName                            |
| Parameter: Formula  | RIGHT(ProductName, LEN(ProductName)-5) |

## Extract by positions

You can extract values between specified index positions within a set of column values. In the following example, the text between the fifth and tenth characters in a column are extracted to a new column.

**Tip:** This extraction method is useful if the content before and after the match area is inconsistent and cannot be described using patterns. If it is consistent, you should use the Extract text or pattern transformation.

|                              |                      |
|------------------------------|----------------------|
| Transformation Name          | Extract by positions |
| Parameter: Column to extract | ProductName          |

|                              |                       |
|------------------------------|-----------------------|
| from                         |                       |
| Parameter: Option            | Between two positions |
| Parameter: Starting position | 5                     |
| Parameter: Ending position   | 10                    |

## Extract by Data Type

You can perform extractions that are specific to a data type or based on failures of the data to match a specified data type.

### Extract date values

You can use functions to extract values from Datetime columns. The example below extracts the year value from the `myDate` column:

|                            |                           |
|----------------------------|---------------------------|
| Transformation Name        | New formula               |
| Parameter: Formula type    | Single row formula        |
| Parameter: Formula         | <code>YEAR(myDate)</code> |
| Parameter: New column name | <code>myYear</code>       |

The following functions can be used to extract values from a Datetime column, as long as the values are present in the formatted date:

- *DAY Function*
- *MONTH Function*
- *YEAR Function*
- *HOUR Function*
- *MINUTE Function*
- *SECOND Function*

You can also reformat the whole Datetime column using the `DATEFORMAT` function. The following reformats the column to show only the two-digit year:

|                     |                                       |
|---------------------|---------------------------------------|
| Transformation Name | Edit column with formula              |
| Parameter: Columns  | <code>myDate</code>                   |
| Parameter: Formula  | <code>DATEFORMAT(myDate, "yy")</code> |

### Extract numeric values

You can extract numerical data from text values. In the following example, the first number is extracted from the `address` column, which would correspond to extracting the street number for the address:

|                                         |                      |
|-----------------------------------------|----------------------|
| Transformation Name                     | Extract patterns     |
| Parameter: Column to extract from       | <code>address</code> |
| Parameter: Option                       | Numbers              |
| Parameter: Number of matches to extract | 1                    |

Empty values in this new column might indicate a formatting problem with the address.

**Tip:** If you set the number of patterns to extract to 2 for the `address` column, you might extract apartment or suite information.

## Extract components of a URL

### URL components

Using functions, you can extract specific elements of a valid URL. The following transformation pulls the domain values from the `myURL` column:

|                                   |                            |
|-----------------------------------|----------------------------|
| <b>Transformation Name</b>        | New formula                |
| <b>Parameter: Formula type</b>    | Single row formula         |
| <b>Parameter: Formula</b>         | <code>DOMAIN(myURL)</code> |
| <b>Parameter: New column name</b> | <code>myDomain</code>      |

In some cases, the function may not return values. For example, the `SUBDOMAIN` function returns empty values if there is no sub-domain part of the URL.

The following functions can be used to extract values from a set of URLs:

- *HOST Function*
- *DOMAIN Function*
- *SUBDOMAIN Function*
- *SUFFIX Function*
- *URLPARAMS Function*

### Query parameters

You can extract query parameter values from an URL. The following example extracts the `store_id` value from the `storeURL` field value:

|                                          |                       |
|------------------------------------------|-----------------------|
| <b>Transformation Name</b>               | Extract patterns      |
| <b>Parameter: Column to extract from</b> | <code>storeURL</code> |
| <b>Parameter: Option</b>                 | HTTP Query strings    |
| <b>Parameter: Fields to extract</b>      | <code>store_id</code> |

## Extract object values

If your data includes sets of arrays, you can extract array elements into columns for each key, with the values written to each key column.

Suppose your restaurant dataset includes a set of characteristics in the `restFeatures` column in the following JSON format:

```
{
 "Credit": "Y",
 "Accessible": "Y",
 "Restrooms": "Y",
```

```

"EatIn": "Y",
"ToGo": "N",
"AlcoholBeer": "Y",
"AlcoholHard": "N",
"TotalTables": "10",
"TotalTableSeats": "36",
"Counter": "Y",
"CounterSeats": "8"
}

```

You can use the following transformation to extract the values from `TotalTableSeats` and `CounterSeats` into separate columns:

|                                                |                             |
|------------------------------------------------|-----------------------------|
| <b>Transformation Name</b>                     | Unnest Objects into columns |
| <b>Parameter: Column</b>                       | restFeatures                |
| <b>Parameter: Paths to elements - 1</b>        | TotalTableSeats             |
| <b>Parameter: Paths to elements - 2</b>        | CounterSeats                |
| <b>Parameter: Include original column name</b> | Selected                    |

After the above is executed, you can perform a simple sum of the `TotalTableSeats` and `CounterSeats` columns to determine the total number of seats in the restaurant.

## Extract array values

In some cases, your data may contain arrays of repeated key-value pairs, where each pair would exist on a separate line. Suppose you have a column called, `Events`, which contains date and time information about the musician described in the same row of data. The `Events` column might look like the following:

```

[{"Date": "2018-06-15", "Time": "19:00"}, {"Date": "2018-06-17", "Time": "19:00"}, {"Date": "2018-06-19", "Time": "20:00"}, {"Date": "2018-06-20", "Time": "20:00"}]

```

The following transformation creates a separate row for each entry in the `Events` column, populating the other fields in the new rows with the data from the original row:

**NOTE:** This type of transformation can significantly increase the size of your dataset.

|                            |                         |
|----------------------------|-------------------------|
| <b>Transformation Name</b> | Expand arrays into rows |
| <b>Parameter: Column</b>   | Events                  |

## Extract Values into a List

You can also extract sets of values into an array list of values.

**Tip:** This transformation is useful for extracting types or patterns of information from a single column.

## Extract matches into array

Using Trifacta patterns, you can extract the values of the column to form a new column of arrays. The following example shows the usage of {any} pattern to extract the cell values and form a new array column.

### Transformation:

|                                                         |                            |
|---------------------------------------------------------|----------------------------|
| <b>Transformation Name</b>                              | Extract matches into Array |
| <b>Parameter: Column</b>                                | product                    |
| <b>Parameter: Pattern matching elements in the list</b> | `{any}`                    |
| <b>Parameter: Delimiter separating each element</b>     | `,`                        |

### Results:

| Before              | After                       |
|---------------------|-----------------------------|
| socks, socks, socks | ["socks", "socks", "socks"] |
| pants, pants        | ["pants", "pants"]          |

## Extract hashtags

Suppose you need to extract the hashtags from customer tweets to another column. In such cases, you can use the {hashtag} Trifacta pattern to extract all hashtag values from a customer's tweets into a new column.

### Source:

The following dataset contains a customer tweets across different locations.

| User Name | Location      | Customer tweets                                                                                                                                                                                                             |
|-----------|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| James     | U.K           | Excited to announce that we've transitioned Wrangler from a hybrid desktop application to a completely cloud-based service! #dataprep #businessintelligence #CommitToCleanData # London                                     |
| Mark      | Berlin        | Learnt more about the importance of identifying issues in your data—early and often #CommitToCleanData #predictivetransformations #realbusinessintelligence                                                                 |
| Catherine | Paris         | Clean data is the foundation of your analysis. Learn more about what we consider the five tenets of sound #dataprep, starting with #1a prioritizing and setting targets. #startwiththeuser #realbusinessintelligence #Paris |
| Dave      | New York      | Learn how #NewYorklife<br>onboarded as part of their #bigdata #dataprep initiative to unlock hidden insights and make them accessible across departments.                                                                   |
| Christy   | San Francisco | How can you quickly determine the number of times a user ID appears in your data?#dataprep #pivot #aggregation#machinelearning initiatives #SFO                                                                             |

### Transformation:

The following transformation extracts the hashtag messages from customer tweets.

|                            |                            |
|----------------------------|----------------------------|
| <b>Transformation Name</b> | Extract matches into Array |
|                            |                            |

|                                                  |                 |
|--------------------------------------------------|-----------------|
| Parameter: Column                                | customer_tweets |
| Parameter: Pattern matching elements in the list | `{hashtag}`     |
| Parameter: New column name                       | Hashtag tweets  |

## Results:

| User Name | Location     | Hashtag tweets                                                                         |
|-----------|--------------|----------------------------------------------------------------------------------------|
| James     | U.K          | ["#dataprep", "#businessintelligence", "#CommitToCleanData", " # London"]              |
| Mark      | Berlin       | ["#CommitToCleanData", "#predictivetransformations", "#realbusinessintelligence", "0"] |
| Catherine | Paris        | ["#dataprep", "#startwiththeuser", "#realbusinessintelligence", "# Paris"]             |
| Dave      | New York     | ["#NewYorklife", "dataprep", "bigdata", "0"]                                           |
| Christy   | SanFrancisco | [ "dataprep", "#pivot", "#aggregation", "#machinelearning"]                            |

# Format Dates

## Contents:

- *Recommended Approaches*
    - *Option 1 - Patterns in the Column Details panel*
    - *Option 2 - Patterns based on date format*
    - *Option 3 - Transformation by Example*
    - *Option 4 - Manual fixups*
  - *Custom Datetime Formats*
  - *Normalize Regional Differences*
- 

Datetime values can be imported into Designer Cloud powered by Trifacta® Enterprise Edition in a variety of formats. Below are just a few examples of one date in different acceptable formats:

| myDate              |
|---------------------|
| Mar-14-2018         |
| 03/14/2018          |
| 2018-Mar-03         |
| 3/14/18             |
| 03/14/2018 00:00:00 |
| March 14, 2018      |

This section describes the tools and approaches for standardizing and formatting your date values.

## Recommended Approaches

When you are formatting a column of date values, you can attempt to standardize the values in the following order.

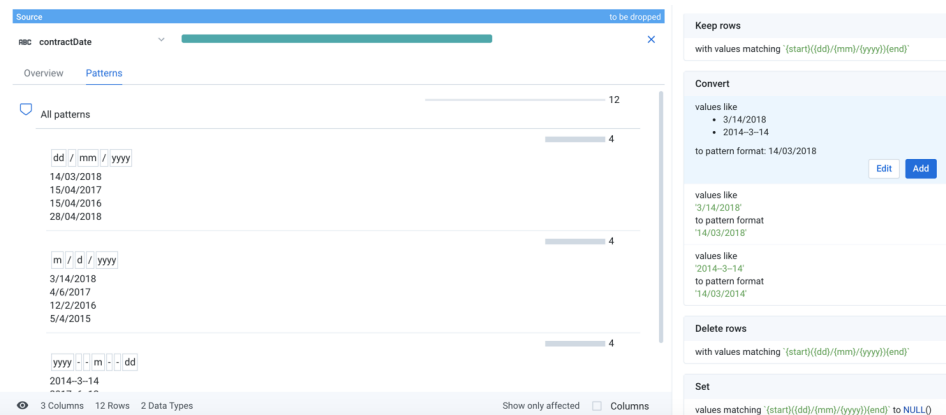
### Option 1 - Patterns in the Column Details panel

Through the Column Details panel, you can review the set of patterns that match the values in your date column and select the ones to apply to standardize the values.

#### Steps:

1. From the column menu for your date column, select **Column Details**.
2. In the Column Details panel, click the Patterns tab.
3. In the Patterns tab, you can review the set of patterns that describe all values that appear in the column. Select one that needs to be corrected.
4. In the right panel, select the Convert card.

**Tip:** If you do not see the Convert card, you might try to generate a new random sample, in which example patterns are more evenly distributed throughout the sample.



**Figure: Select the Convert card in the Patterns tab**

5. Click **Add**.
6. The number of patterns displayed in the Patterns tab is reduced. You can continue to select patterns to standardize values.
7. Iterate until there is only one pattern displayed in the panel. For more information on Datetime patterns, see *Standardize Using Patterns*.

## Option 2 - Patterns based on date format

In some cases, you may not be able to simply select patterns, which generates sufficient suggestions to standardize your date values. A second approach involves keying on mismatched values in the column.

**Tip:** This technique works for columns in which all values are valid Datetime values but are in different date formats. If you have values that are invalid for any date format, you must use Option 3 to correct the syntax errors using patterns first. See below.

In this case, you set the data type for the column to Datetime and use the DATEFORMAT function to match the format of the values that you want to change. Next to the values from the preceding table, you can see the corresponding date format token:

| myDate              | DATEFORMAT value    |
|---------------------|---------------------|
| Mar-14-2018         | MMM-dd-yyyy         |
| 03/14/2018          | MM/dd/yyyy          |
| 2018-Mar-03         | yyyy-MMM-dd         |
| 3/14/18             | M/d/yy              |
| 03/14/2018 00:00:00 | MM/dd/yyyy HH:mm:ss |
| March 14, 2018      | MMMM dd, yyyy       |

For purposes of this example, suppose your `myDate` column contains values in `MM/dd/yyyy` and `M/d/yy` format. You wish to standardize on `MMMM dd, yyyy` format.

### Steps:

1. From the Data Type menu at the top of the `myDate` column, select **Date/Time**.



2. In the dialog, select the Date format that matches values you wish to fix:

Date / Time Type ✕

☐ mm

☐ yy

☐ mm-yy

☐ mm-dd

☐ dd-mm

☒ mm-dd-yy 

month\*dd\*yyyy

☐ dd-mm-yy

☐ yy-mm-dd

☐ yy-dd-mm

☐ mm-dd-yy hh:mm:ss

☐ dd-mm hh:mm:ss

☐ mm-dd hh:mm:ss

☐ dd-mm-yy hh:mm:ss

☐ yy-mm-dd hh:mm:ss

☐ yy-dd-mm hh:mm:ss

☐ hh:mm:ss

Cancel

Save

**Figure: Date/Time format selector**

For more information on the supported Datetime formats, see *Datetime Data Type*.

3. Click **Save**.
4. Now, you need to modify the values that match this format to match the target format (MMMM dd, yyyy). Click the green bar in the column, which matches the values for the currently valid Datetime format., Then click the Set suggestion. Click **Modify**.
5. In the Transform Builder, you have a predefined transformation that sets values based on whether the column values are valid for the currently specified data type and format. You must replace the `NULL( )` entry with the `DATEFORMAT` function which changes these values to the proper format:

|                            |                                                                                           |
|----------------------------|-------------------------------------------------------------------------------------------|
| <b>Transformation Name</b> | Edit with formula                                                                         |
| <b>Parameter: Columns</b>  | myDate                                                                                    |
| <b>Parameter: Formula</b>  | <code>ifvalid(\$col, ['Datetime','yy','yyyy'], dateformat(\$col, 'MMMM dd, yyyy'))</code> |

6. Click **Add**. All values that matched the `MM/dd/yyyy` format are converted to the `MMMM dd, yyyy` format.
7. Repeat the previous steps:
  - a. Set the column's Datetime format to: `M/d/yyyy`.
  - b. Select the green bar in the column data quality bar.
  - c. Select the Set suggestion and modify it.
  - d. For the value in the transformation, insert the following function:

```
ifvalid($col, ['Datetime','M/d/yyyy'], dateformat(myDate, 'MMMM dd, yyyy'))
```

- e. Add the transformation to you recipe.
8. Repeat Step 7 for any other mismatched formats.
  9. You may have some manual fixups to complete at the end. See below.

### Option 3 - Transformation by Example

You can reformat dates by providing example output values for a listed source value. For a column of date values, you can begin providing example outputs for individual values, and Designer Cloud powered by Trifacta Enterprise Edition can perform pattern-based transformations to similarly formatted values. For more information, see *Overview of TBE*.

### Option 4 - Manual fixups

#### Steps:

1. Now that you have selected a specific format for your Datetime values, the rows that do not match this format are now identified as mismatched in the column. Click the red bar at the top of the column.
2. In the Status bar at the bottom of the screen, click **Show only affected rows**.
3. You can now see only the rows that remain mismatched with respect to the preferred Datetime format.
4. Select one of these values. For example, suppose you have quite a few values that are only four-digit year values (YYYY). Select one of the values. Then, select the Replace card. Click **Edit**.
5. Your transformation should look like the following:

|                                |                          |
|--------------------------------|--------------------------|
| <b>Transformation Name</b>     | Replace text or patterns |
| <b>Parameter: Column</b>       | UpdateTime               |
| <b>Parameter: Find</b>         | `{start}{digit}{4}{end}` |
| <b>Parameter: Replace with</b> | ' '                      |

6. You can modify the search and replace patterns to capture and write back the year value:
  - a. In the Find value, put parentheses around the pattern that captures the four digits in a row. Adding parentheses around a matching pattern identifies that sub-pattern as a **capture group**, which can be referenced in any replacement.
  - b. The capture group should look like the following:

```
((digit){4})
```

- c. For the Replace with value, you must insert a month and day value according to the format selected for the column (MM/DD/YYYY), followed by a reference back to the capture group.
- d. Capture groups from the matching pattern can be referenced in the replacement value using references such as \$1, \$2, \$3, and so on. These tokens refer to the first, second, and third capture groups in the Find value.
- e. The Replace value should look like the following:

```
01/01/$1
```

- f. Your transformation should look like the following when done:

|                                |                            |
|--------------------------------|----------------------------|
| <b>Transformation Name</b>     | Replace text or patterns   |
| <b>Parameter: Column</b>       | UpdateTime                 |
| <b>Parameter: Find</b>         | `{start}((digit){4}){end}` |
| <b>Parameter: Replace with</b> | 01/01/\$1                  |

7. Click **Add**.

8. You can repeat these steps for the remaining mismatched values.

## Custom Datetime Formats

You can create your own customized Datetime formats using the `DATEFORMAT` function. For example, the following changes the format of the `lastDate` function to use the `yyyy:MM:dd` format:

|                            |                                    |
|----------------------------|------------------------------------|
| <b>Transformation Name</b> | Edit with formula                  |
| <b>Parameter: Columns</b>  | lastDate                           |
| <b>Parameter: Formula</b>  | DATEFORMAT(lastDate, 'yyyy:MM:dd') |

For more information on the supported codes for specifying your own Datetime formats, see *Datetime Data Type*.

## Normalize Regional Differences

The following date values correspond to the same date but vary in format in different regions of the world:

| Date Value | Region |
|------------|--------|
| 03/14/2018 | U.S.   |
| 14/03/2018 | E.U.   |
| 2014-03-14 | China  |

In the above examples, the delimiters for the U.S. and E.U. values are identical, which makes parsing these values more challenging.

**Tip:** If your dataset contains date values from different regions of the world, you should find or create a separate column to identify the applicable region.

Suppose the previous set of dates was represented in your dataset with the following values:

| contractDate | region |
|--------------|--------|
| 03/14/2018   | USA    |
| 14/03/2018   | EU     |
| 2014-03-14   | CHN    |

In this case, you might try the following generalized solution. You can use conditional transformations to extract the day, month, and year values from the `contractDate` column based on the value in the `region` column.

**NOTE:** This solution assumes that all date values within for a specific region (e.g. USA) are consistently formatted. You should perform those formatting actions first.

### Steps:

1. First, you must split the column based on the cell value's delimiter. Note that the following transformation uses the Pattern `{delim}` to locate the delimiter in the cell value. This delimiter is either a dash or a slash.

|                            |                    |
|----------------------------|--------------------|
| <b>Transformation Name</b> | Split by delimiter |
|                            |                    |

|                             |              |
|-----------------------------|--------------|
| <b>Parameter: Column</b>    | contractDate |
| <b>Parameter: Option</b>    | by Delimiter |
| <b>Parameter: Delimiter</b> | `{delim}`    |

2. Create the following three conditional transformations for extracting the day, month, or year values based on the value in the Region column. Here is the transformation to acquire the year values:

|                                      |                       |
|--------------------------------------|-----------------------|
| <b>Transformation Name</b>           | conditions            |
| <b>Parameter: Condition type</b>     | Case on single column |
| <b>Parameter: Column to evaluate</b> | Region                |
| <b>Parameter: Case 1</b>             | 'EU'                  |
| <b>Parameter: Value 1</b>            | contractDate3         |
| <b>Parameter: Case 2</b>             | 'USA'                 |
| <b>Parameter: Value 2</b>            | contractDate3         |
| <b>Parameter: Case 3</b>             | 'CHN'                 |
| <b>Parameter: Value 1</b>            | contractDate1         |

3. For month:

|                                      |                       |
|--------------------------------------|-----------------------|
| <b>Transformation Name</b>           | conditions            |
| <b>Parameter: Condition type</b>     | Case on single column |
| <b>Parameter: Column to evaluate</b> | Region                |
| <b>Parameter: Case 1</b>             | 'EU'                  |
| <b>Parameter: Value 1</b>            | contractDate2         |
| <b>Parameter: Case 2</b>             | 'USA'                 |
| <b>Parameter: Value 2</b>            | contractDate1         |
| <b>Parameter: Case 3</b>             | 'CHN'                 |
| <b>Parameter: Value 1</b>            | contractDate2         |

4. For day:

|                                      |                       |
|--------------------------------------|-----------------------|
| <b>Transformation Name</b>           | conditions            |
| <b>Parameter: Condition type</b>     | Case on single column |
| <b>Parameter: Column to evaluate</b> | Region                |
| <b>Parameter: Case 1</b>             | 'EU'                  |
| <b>Parameter: Value 1</b>            | contractDate1         |
| <b>Parameter: Case 2</b>             | 'USA'                 |
| <b>Parameter: Value 2</b>            | contractDate2         |

|                           |               |
|---------------------------|---------------|
| <b>Parameter: Case 3</b>  | 'CHN'         |
| <b>Parameter: Value 1</b> | contractDate3 |

5. You can now bring together these three columns:

|                                   |                  |
|-----------------------------------|------------------|
| <b>Transformation Name</b>        | Merge columns    |
| <b>Parameter: Columns</b>         | day, month, year |
| <b>Parameter: Separator</b>       | ' / '            |
| <b>Parameter: New column name</b> | newDate          |

6. You now have your new date column. You may need to reformat it into a preferred format.
7. Delete the columns that were created during this process.

# Apply Conditional Transformations

## Contents:

- *Single- and Multi-Case Transformations*
- *Conditional Functions*
  - *IF function*
  - *CASE function*
- *Logical Operators*

In your recipe steps, you can apply conditional logic to determine if transformational changes should occur. You can build logical tests into your transformations in multiple levels:

- **Single- and multi-case transformations:** Use case-based transformations to test if-then or case logic against your dataset and to apply the specified results.
- **Conditional functions:** IF and CASE functions can be applied to any transformation that accepts functional expressions.
- **Logical operators:** You can use AND or OR logic to build your conditional expressions.

**NOTE:** If you are running your job on Spark, avoid creating single conditional transformations with deeply nested sets of conditions. On Spark, these jobs can time out, and deeply nested steps can be difficult to debug. Instead, break up your nesting into smaller conditional transformations of multiple steps.

## Single- and Multi-Case Transformations

Through the Transform Builder, you can build conditional tests using if/then/else or case logic to manipulate on the data.

1. In the Search panel in the Transformer page, enter `case`.
2. You can choose one of three different logical transformations:
  - a. **If-then-else:** Specify any logical test that evaluates to `true` or `false` and specify values if `true` (then) or if `false` (else).
  - b. **Single-column case:** Test for explicit values in a column and, if true, write specific values to the new column.
  - c. **Custom conditions:** Specify any number of case statements, which can have completely independent expressions:
    - i. Case 1 is tested, and a value is written if `true`.
    - ii. If Case 1 is false, then Case 2 is tested. If `true`, a different value can be written.
    - iii. Supports an arbitrary number of independent conditional cases.
3. Specify the other parameters, including the name of the new column.

After the transformation is added to the recipe, actions can then be taken based on the values in this new column.

## Conditional Functions

You can also apply conditional logical as part of your function definitions for other transformations.

### IF function

For example, the following replaces values in the same column with `IN` if they are greater than 0.5 or `OUT` otherwise:

|                     |                          |
|---------------------|--------------------------|
| Transformation Name | Edit column with formula |
|---------------------|--------------------------|

|                           |                              |
|---------------------------|------------------------------|
| <b>Parameter: Columns</b> | testCol                      |
| <b>Parameter: Formula</b> | IF(\$col >= 0.5, 'IN','OUT') |

In the above, the token \$col is a reference back to the value defined for the column (testCol in this case). However, you can replace it with a reference to any column in the dataset.

You can use the IF function in any transformation that accepts functional inputs. For more information, see *IF Function*.

## CASE function

You can chain together IF functions in the following manner:

|                            |                                                               |
|----------------------------|---------------------------------------------------------------|
| <b>Transformation Name</b> | Edit column with formula                                      |
| <b>Parameter: Columns</b>  | testCol                                                       |
| <b>Parameter: Formula</b>  | IF(\$col >= 0.5, 'IN',(IF(\$col >= 0.35, 'MAYBE IN', 'OUT'))) |

However, these can become problematic to debug. Instead, you can use the CASE function to assist in building more complex logical trees. The following is more legible and easier to manage:

|                            |                                                                |
|----------------------------|----------------------------------------------------------------|
| <b>Transformation Name</b> | Edit column with formula                                       |
| <b>Parameter: Columns</b>  | testCol                                                        |
| <b>Parameter: Formula</b>  | CASE([ \$col >= 0.75, 'IN', \$col >= 0.35, 'MAYBE IN', 'OUT']) |

| If test            | Test          | Output if true |
|--------------------|---------------|----------------|
| If:                | \$col >= 0.75 | IN             |
| If above is false: | \$col >= 0.35 | MAYBE IN       |
| If above is false: | default       | OUT            |

For more information, see *CASE Function*.

## Logical Operators

Logical operators can be applied to your function expressions to expand the range of your logical tests.

In the above example, suppose you have a second column called, Paid, which contains Boolean values. You could expand the previous statement to include a test to see if Paid=true:

|                            |                                                                                                    |
|----------------------------|----------------------------------------------------------------------------------------------------|
| <b>Transformation Name</b> | Edit column with formula                                                                           |
| <b>Parameter: Columns</b>  | testCol                                                                                            |
| <b>Parameter: Formula</b>  | CASE([ (\$col >= 0.75 && Paid == true), 'IN', (\$col >= 0.35 && Paid == true), 'MAYBE IN', 'OUT']) |

The above performs a logical AND operation on the two expressions in each tested case. The logical operator is &&.

You can also reference explicit functions to perform logical tests. The above might be replaced with the following:

|                     |                                                                                                      |
|---------------------|------------------------------------------------------------------------------------------------------|
| Transformation Name | Edit column with formula                                                                             |
| Parameter: Columns  | testCol                                                                                              |
| Parameter: Formula  | CASE([ AND(\$col >= 0.75, Paid == true), 'IN', AND(\$col >= 0.35, Paid == true), 'MAYBE IN', 'OUT']) |

| Logic       | Logical Operator | Logical Function |
|-------------|------------------|------------------|
| Logical AND | (exp1 && exp2)   | AND(exp1,exp2)   |
| Logical OR  | (exp1    exp2)   | OR(exp1,exp2)    |
| Logical NOT | !(exp1 == exp2)  | NOT(exp1,exp2)   |

Depending on the structure of your transformation and your preferences, either form may be used.

- For more information, see *Logical Operators*.
- For more information, see *Logical Functions*.



# Prepare Data for Machine Processing

## Contents:

- *Scaling*
    - *Scale to zero mean and unit variance*
    - *Scale to min-max range*
  - *Outliers*
    - *Identify outliers*
    - *Remove outliers*
    - *Change outliers to mean values*
  - *Binning*
    - *Bins of equal size*
    - *Bins of custom size*
  - *One-Hot Encoding*
- 

Depending on your downstream system, you may need to convert your data into numeric values of the expected form or to standardize the distribution of numeric values. This section summarizes some common statistical transformations that can be applied to columnar data to prepare it for use in downstream analytic systems.

## Scaling

You can scale the values within a column using either of the following techniques.

### Scale to zero mean and unit variance

**Zero mean and unit variance** scaling renders the values in the set to fit a normal distribution with a mean of 0 and a variance of 1. This technique is a common standard for normalizing values into a normal distribution for statistical purposes.

In the following example, the values in the `POS_Sales` column have been normalized to average 0, variance 1.

- **Remove mean:** When selected, the existing mean (average) of the values is used as the center of the distribution curve.

**NOTE:** Re-centering sparse data by removing the mean may remove sparseness.

- **Scale to unit variance:** When selected, the range of values are scaled such that their variance is 1. When deselected, the existing variance is maintained.

**NOTE:** Scaling to unit variance may not work well for managing outliers. Some additional techniques for managing outliers are outlined below.

|                           |                                      |
|---------------------------|--------------------------------------|
| Transformation Name       | Scale column                         |
| Parameter: Column         | POS_Sales                            |
| Parameter: Scaling method | Scale to zero mean and unit variance |
| Parameter: Remove mean    | false                                |
| Parameter: Scale to unit  | true                                 |

|                            |                   |
|----------------------------|-------------------|
| variance                   |                   |
| Parameter: Output options  | Create new column |
| Parameter: New column name | scale_POS_Sales   |

## Scale to min-max range

You can scale column values fitting between a specified minimum and maximum value. This technique is useful for distributions with very small standard deviation values and for preserving 0 values in sparse data.

The following example scales the `TestScores` column to a range of 0 and 1, inclusive.

|                           |                                |
|---------------------------|--------------------------------|
| Transformation Name       | Scale column                   |
| Parameter: Column         | TestScores                     |
| Parameter: Scaling method | Scale to a given min-max range |
| Parameter: Minimum        | 0                              |
| Parameter: Maximum        | 1                              |
| Parameter: Output options | Replace current column         |

## Outliers

You can use several techniques for identifying statistical outliers in your dataset and managing them as needed.

### Identify outliers

Suppose you need to remove the outliers from a column. Assuming a normal bell distribution of values, you can use the following formula to calculate the number of standard deviations a column value is from the column mean (average). In this case, the source column is `POS_Sales`.

|                            |                                                                                                         |
|----------------------------|---------------------------------------------------------------------------------------------------------|
| Transformation Name        | New formula                                                                                             |
| Parameter: Formula type    | Multiple row formula                                                                                    |
| Parameter: Formula         | $(\text{ABS}(\text{POS\_Sales} - \text{AVERAGE}(\text{POS\_Sales}))) / \text{STDEV}(\text{POS\_Sales})$ |
| Parameter: New column name | stdevs_POS_Sales                                                                                        |

### Remove outliers

The new `stdevs_POS_Sales` column now contains the number of standard deviations from the mean for the corresponding value in `POS_Sales`. You can use the following transformation to remove the rows that contain outlier values for this column.

**Tip:** An easier way to select these outlier values is to select the range of values in the `stdevs_POS_Sales` column histogram. Then, select the suggestion to delete these rows. You may want to edit the actual formula before you add it to your recipe.

In the following transformation, all rows that contain a value in `POS_Sales` that is greater than four standard deviations from the mean are deleted:

|  |  |
|--|--|
|  |  |
|--|--|

|                                   |                       |
|-----------------------------------|-----------------------|
| <b>Transformation Name</b>        | Filter rows           |
| <b>Parameter: Condition</b>       | Custom formula        |
| <b>Parameter: Type of formula</b> | Custom single         |
| <b>Parameter: Condition</b>       | 4 <= stdevs_POS_Sales |
| <b>Parameter: Action</b>          | Delete matching rows  |

## Change outliers to mean values

You can also remove the effects of outliers by setting their value to the mean (average), which preserves the data in other columns in the row.

|                            |                                                         |
|----------------------------|---------------------------------------------------------|
| <b>Transformation Name</b> | Edit with formula                                       |
| <b>Parameter: Columns</b>  | POS_Sales                                               |
| <b>Parameter: Formula</b>  | IF(stdevs_POS_Sales > 4, AVERAGE(POS_Sales), POS_Sales) |

## Binning

You can modify your data to fit into bins of equal or custom size. For example, the lowest values in your range would be marked in the 0 bin, with larger values being marked with larger bin numbers.

### Bins of equal size

You can bin numeric values into bins of equal size. Suppose your column contains numeric values 0–1000. You can bin values into equal ranges of 100 by creating 10 bins.

|                                   |                   |
|-----------------------------------|-------------------|
| <b>Transformation Name</b>        | Bin column        |
| <b>Parameter: Column</b>          | MilleBornes       |
| <b>Parameter: Select Option</b>   | Equal Sized Bins  |
| <b>Parameter: Number of Bins</b>  | 10                |
| <b>Parameter: New column name</b> | MilleBornesRating |

### Bins of custom size

You can also create custom bins. In the following example, the TestScores column is binned into the following bins. In a later step, these bins are mapped to grades:

| Bins       | Bin Range | Bin Number | Grade |
|------------|-----------|------------|-------|
| 59         | 0-59      | 0          | F     |
| 69         | 60-69     | 1          | D     |
| 79         | 70-79     | 2          | C     |
| 89         | 80-89     | 3          | B     |
|            | 90+       | 4          | A     |
| (no value) |           |            | I     |

First, you bin values into the bin numbers listed above:

|                            |                 |
|----------------------------|-----------------|
| Transformation Name        | Bin column      |
| Parameter: Column          | TestScores      |
| Parameter: Select option   | Custom bin size |
| Parameter: Bins            | 59,69,79,89     |
| Parameter: New column name | Grades          |

You can then use the following transformation to assign letters in the Grades column:

|                               |                       |
|-------------------------------|-----------------------|
| Transformation Name           | Conditions            |
| Parameter: Condition type     | Case on single column |
| Parameter: Column to evaluate | Grades                |
| Parameter: Case - 0           | 'F'                   |
| Parameter: Case - 1           | 'D'                   |
| Parameter: Case - 2           | 'C'                   |
| Parameter: Case - 3           | 'B'                   |
| Parameter: Case - 4           | 'A'                   |
| Parameter: Default value      | 'I'                   |
| Parameter: New column name    | Grades_letters        |

## One-Hot Encoding

**One-hot encoding** refers to distributing the listed values in a column into individual columns. Within each row of each individual column is a 0 or a 1, depending on whether the value represented by the column appears in the corresponding source column. The source column is untouched. This method of encoding allows for easier consumption of data in target systems.

**Tip:** This transformation is particularly useful for columns containing a limited set of enumerated values.

In the following example, the values in the BrandName column are distributed into separate columns of binary values, with a maximum limit of 50 new columns.

**NOTE:** Be careful applying this to a column containing a wide variety of values, such as Decimal values. Your dataset can expand significantly in size. Use the max columns setting to constrain the upper limit on dataset expansion.

|                                            |                                       |
|--------------------------------------------|---------------------------------------|
| Transformation Name                        | One-hot encoding of values to columns |
| Parameter: Column                          | BrandName                             |
| Parameter: Max number of columns to create | 50                                    |

**Tip:** If needed, you can prepend the names of the resulting columns with a reference to the source column. See *Rename Columns*.

# Enrichment Tasks

These topics cover various approaches to augmenting your data with fixed values, generated values, or data from other datasets.

For more information on the general methods for adding to your dataset, see *Enriching Data*.

# Create New Column

## Contents:

- *New Formula*
    - *Add a column of text values*
    - *Add a column that uses a function*
    - *Add a column that references another column*
    - *Add a column using constants, functions, and column references*
  - *Merge Columns*
  - *Extract Values from a Column*
  - *Split Column Values*
  - *Convert a Column into Multiple Columns*
    - *Unnest*
- 

You can create a new column by adding or editing a formula on any existing column by performing the following steps:

## New Formula

The New Formula transformation allows you to create a new column based upon a formula that you provide to the transformation. Below are some examples.

### Add a column of text values

You can insert a new column containing a string value that you specify as part of the transformation. In the following example, the `status` column is created, and all values in it are set to `ok`.

|                            |                    |
|----------------------------|--------------------|
| Transformation Name        | New formula        |
| Parameter: Formula type    | Single row formula |
| Parameter: Formula         | 'ok'               |
| Parameter: New column name | status             |

### Add a column that uses a function

You can insert a new column by using a function. In the following example, the `currentyear` column is extracted as a new column from the `TransactionDate` column using `YEAR` function. For more information on extracting date information, see *Extract Values*.

|                            |                        |
|----------------------------|------------------------|
| Transformation Name        | New formula            |
| Parameter: Formula type    | Single row formula     |
| Parameter: Formula         | YEAR (TransactionDate) |
| Parameter: New column name | currentyear            |

## Add a column that references another column

You can also insert columns containing references to other columns. In the following example, the `totalCost` column is created called `totalCost`, which is based on the formula using three separate columns: `baseCost + totalTax - totalDiscount`:

|                            |                                                  |
|----------------------------|--------------------------------------------------|
| Transformation Name        | New formula                                      |
| Parameter: Formula type    | Single row formula                               |
| Parameter: Formula         | <code>baseCost + totalTax - totalDiscount</code> |
| Parameter: New column name | <code>totalCost</code>                           |

## Add a column using constants, functions, and column references

You can insert a column by using nested expressions by using constants, functions, and column references. In the following example, the `Three` column is created, which is based on nested functions `ROUND` and `DIVIDE`.

|                            |                                    |
|----------------------------|------------------------------------|
| Transformation Name        | New formula                        |
| Parameter: Formula type    | Single row formula                 |
| Parameter: Formula         | <code>ROUND(DIVIDE(10,3),0)</code> |
| Parameter: New column name | <code>Three</code>                 |

## Merge Columns

You can merge two or more columns together to create a new column containing the merged values. For more information, see *Add Two Columns*.

## Extract Values from a Column

You can extract values based on patterns or literal values from one column and insert them into a new column. See *Extract Values*.

## Split Column Values

You can split the values in a column into separate columns based on delimiters and other conditions that you define. See *Split Column*.

## Convert a Column into Multiple Columns

### Unnest

You can extract values stored in an array into separate columns in your dataset. This type of transformation can be useful for unpacking nested data such as JSON into tabular format.

- For more information, see *Working with JSON v2*.
- For more information, see *Working with Arrays*.



# Add Two Columns

Contents:

- *Check Data Types*
- *Check Values*
- *Syntax of Math Functions*
- *Add One Column into Another*
- *Add Selective Values from One Column into Another*
- *Add Two Columns into a New Third Column*
- *Working with More than Two Columns*
- *Concatenating Columns*
- *Summing Rows*

This section provides an overview of how to perform mathematical operations between columns.

## Check Data Types

Before you begin, you should verify that the data types of the two columns match. Check the icon in the upper left of each column to verify that they match.

To change the data type, you can:

- Click the data type icon.
- Select **Edit data type** from the column menu. See *Column Menus*.

## Check Values

After setting data types, you should address any missing or mismatched values in the column. For example, if you change a column's data type from Decimal to Integer, values that contain decimal points may be reported as mismatched values. Use the `ROUND` function to round them to the nearest integer.

|                     |                          |
|---------------------|--------------------------|
| Transformation Name | Edit column with formula |
| Parameter: Columns  | myColumn                 |
| Parameter: Formula  | ROUND(myColumn)          |

See *ROUND Function*.

**Tip:** You can use the `FLOOR` or `CEILING` functions to force rounding down or up to the nearest integer.

See *FLOOR Function*.

See *CEILING Function*.

## Syntax of Math Functions

You can express mathematical operations using numeric operators or function references. The following two examples perform the same operation, creating a third column that sums the first two.

**Numeric Operators:**

|                                   |                      |
|-----------------------------------|----------------------|
| <b>Transformation Name</b>        | New formula          |
| <b>Parameter: Formula type</b>    | Single row formula   |
| <b>Parameter: Formula</b>         | (colA + colB + colC) |
| <b>Parameter: New column name</b> | 'colD'               |

## Math Functions:

|                                   |                    |
|-----------------------------------|--------------------|
| <b>Transformation Name</b>        | New formula        |
| <b>Parameter: Formula type</b>    | Single row formula |
| <b>Parameter: Formula</b>         | ADD(colA,colB)     |
| <b>Parameter: New column name</b> | 'colD'             |

**NOTE:** Expressions containing numeric operators can contain more than two column references or values, as well as nested expressions. Math functions support two references only.

For more information, see *Numeric Operators*.

For more information, see *Math Functions*.

## Add One Column into Another

To perform math operations, you can use the Edit column with formula transformation to update values in a column based on a math operation. The following transformation multiplies the column by 10 and adds the value of colB:

|                            |                          |
|----------------------------|--------------------------|
| <b>Transformation Name</b> | Edit column with formula |
| <b>Parameter: Columns</b>  | colA                     |
| <b>Parameter: Formula</b>  | ((colA * 10) + colB)     |

All values in colA are modified based on this operation.

## Add Selective Values from One Column into Another

You can use the Edit column with formula transformation to perform math operations based on a condition you define. In the following step, the Cost column is replaced reduced by 10% if the Qty column is more than 100. The expression is rounded down to the nearest integer, so that the type of the column (Integer) is not changed:

|                            |                                        |
|----------------------------|----------------------------------------|
| <b>Transformation Name</b> | Edit column with formula               |
| <b>Parameter: Columns</b>  | Cost                                   |
| <b>Parameter: Formula</b>  | IF(Qty > 100, ROUND(Cost * 0.9), Cost) |

For rows in which Qty is less than 100, the value of Cost is written back to the column (no change).

## Add Two Columns into a New Third Column

To create a new column in which a math operation is performed on two other columns, use the New Formula transformation. The following multiplies `Qty` and `UnitPrice` to yield `Cost`:

|                            |                                                       |
|----------------------------|-------------------------------------------------------|
| Transformation Name        | New formula                                           |
| Parameter: Formula type    | Single row formula                                    |
| Parameter: Formula         | MULTIPLY( <code>Qty</code> , <code>UnitPrice</code> ) |
| Parameter: New column name | 'Cost '                                               |

## Working with More than Two Columns

If you need to work with more than two columns, numeric operators allow you to reference any number of columns and static values in a single expression.

However, you should be careful to avoid making expressions that are too complex, as they can be difficult to parse and debug.

**Tip:** When performing complex mathematic operations, you may want to create a new column to contain the innermost computations of your expression. Then, you can reference this column in the subsequent step, which generates the full expression. In this manner, you can build complex equations in a way that is easier to understand for other users of the recipe. The final step is to delete the generated column.

## Concatenating Columns

If you are concatenating string-based content between multiple columns, use the Merge Columns transformation. In the following example, the Merge Columns transformation is used to bring together the order ID (`ordId`) and product ID (`prodId`) columns, with the dash character used as the delimiter between the two column values:

|                            |                                          |
|----------------------------|------------------------------------------|
| Transformation Name        | Merge columns                            |
| Parameter: Columns         | <code>ordId</code> , <code>prodId</code> |
| Parameter: Separator       | ' - '                                    |
| Parameter: New column name | <code>primaryKey</code>                  |

**Tip:** This method can be used for columns of virtually any type. Change the data type of each column to String and then perform the merge operation.

Array column types can be concatenated with the `ARRAYCONCAT` function. See *ARRAYCONCAT Function*.

**Tip:** You can also use the `MERGE` function to accomplish the above actions. The function method is useful if you are performing a separate transformation action on the data involved. For example, you could use the function if you are using the Edit formula column to modify a column in place. See *MERGE Function*.

## Summing Rows

You can use aggregate functions to perform mathematic operations on sets of rows. Aggregated rows are collapsed and grouped based on the functions that you apply to them. See *Aggregate Functions*.

# Generate Primary Keys

## Contents:

- *The unique row identifier method*
  - *Standardize formatting*
  - *Combine across datasets*
- *The combined field method*

In database terms, a **primary key** is a column or set of columns that uniquely identifies rows in a table. Examples:

- For log data or other transactional data, the timestamp is typically a unique identifier.

**Tip:** If you think you need a primary identifier for your dataset, you should try to identify it or create it before you delete potentially useful columns.

- Product information typically contains an SKU identifier. If that is not available, you may need brand, make, and model combinations, which can be created using the method described below.

A well-organized source of data is likely to contain this information for you, but in some cases, you may be required to generate your own primary key.

**Tip:** In the Transformer page, a quick way to check if there is a primary key in your dataset is to compare the count of categories in the data histograms for string-based data against the count of rows. If the numbers are equal, then the column is suitable for use as a primary key. However, if you ever join with another dataset, you must re-review the suitability of the field and may need to build a new primary key field. Keep in mind that counts apply to the displayed sample, instead of the entire dataset.

This section provides two methods for generating primary keys in your datasets.

## The unique row identifier method

When a dataset is loaded into the Transformer page for the first time, you can see a set of black dots along the left side. Hover over these dots to reveal the row numbers retrieved from the original source, if that information is still available. This method relies on these numbers for generating primary keys and is suitable when your final output contains a relatively few number of combined datasets.

**NOTE:** Some transforms make original row order information unavailable. You cannot retrieve this information from relational sources.

See *Source Metadata References*.

**Tip:** When you first load your dataset into the Transformer page, you should generate a column containing the original row information, such as the following:

|                         |                    |
|-------------------------|--------------------|
| Transformation Name     | New formula        |
| Parameter: Formula type | Single row formula |

|                            |                     |
|----------------------------|---------------------|
| Parameter: Formula         | SOURCEROWNUMBER ( ) |
| Parameter: New column name | origRowId           |

This transform is useful to include after initial inference and structuring of each recipe for all of your datasets.

## Standardize formatting

The output of this column is a list of numeric values from 1 or 2 to the final row of your dataset. As a unique identifier, you might want to standardize these values. For example, you are transforming a set of orders. You may want to prepend your unique row identifiers with a code and to format them based on a fixed length, as in the following:

| origRowId | keyPrefix | primaryKey |
|-----------|-----------|------------|
| 1         | ORD000    | ORD0001    |
| 2         | ORD000    | ORD0002    |
| ...       | ORD000    | ...        |
| 10        | ORD00     | ORD0010    |
| ...       | ORD00     | ...        |
| 99        | ORD00     | ORD0099    |
| 100       | ORD0      | ORD0100    |

This structuring generates primary keys of consistent length. You can use the following steps to standardize their formatting, assuming that you have already created the `origRowId` column.

### Steps:

1. Change this column to be of String type. Select **String** from the data type drop-down for the column.
2. Create a column containing your prepended identifier and the proper number of zeroes. The following bit of logic generates a string with the proper number of zeroes depending on the length of the value in `origRowId`:

|                            |                                                                                                          |
|----------------------------|----------------------------------------------------------------------------------------------------------|
| Transformation Name        | New formula                                                                                              |
| Parameter: Formula type    | Single row formula                                                                                       |
| Parameter: Formula         | IF(LEN(origRowId) > 3, 'ORD', IF(LEN(origRowId) > 2, 'ORD0', IF(LEN(origRowId) > 1, 'ORD00', 'ORD000'))) |
| Parameter: New column name | keyPrefix                                                                                                |

**NOTE:** The following works for up to 10,000 rows in the original dataset. You need to add additional `IF` clauses when your row counts exceed 10,000.

3. Now, you can merge these columns together:

|                     |               |
|---------------------|---------------|
| Transformation Name | Merge columns |
|                     |               |

|                                   |                     |
|-----------------------------------|---------------------|
| <b>Parameter: Columns</b>         | keyPrefix,origRowId |
| <b>Parameter: New column name</b> | primaryKey          |

4. You can now delete the prefix column:

|                            |                         |
|----------------------------|-------------------------|
| <b>Transformation Name</b> | Delete columns          |
| <b>Parameter: Columns</b>  | keyPrefix               |
| <b>Parameter: Action</b>   | Delete selected columns |

These steps should be applied across all datasets that you intend to combine into your output dataset.

## Combine across datasets

After you have combined or enriched your dataset, you can combine these original row ID fields from each dataset to create a super primary key in the combined dataset using the method described below.

### The combined field method

If your final dataset contains more than a few combined datasets, this basic method for creating a primary key is to find a combination of fields that collectively represent a unique identifier from the final dataset. Columns:

- LastName
- FirstName
- TestNumber
- TestScore

Since there are multiple instances of test data for each person, there is no single column to use as a primary key.

#### Steps:

1. Load the dataset into the Transformer page.
2. Identify the columns that together can uniquely identifier a row. In the TestScores-All example, these columns are the following:
  - a. LastName
  - b. FirstName
  - c. TestNumber

**NOTE:** It may be possible to set up a key using LastName and TestNumber, but that is not guaranteed. If the dataset changes over time, a working key based on these columns may become broken.

3. Use the `merge` transform to combine these columns together into a new column, such as the following:

|                                   |                            |
|-----------------------------------|----------------------------|
| <b>Transformation Name</b>        | Merge columns              |
| <b>Parameter: Columns</b>         | LastName,FirstName,TestNum |
| <b>Parameter: Separator</b>       | ' - '                      |
| <b>Parameter: New column name</b> | TestID                     |

The `with` clause identifies the delimiter between the merged column values.

4. Values should look like the following:

| TestID       |
|--------------|
| Smith-Joe-2  |
| Doe-Jane-4   |
| Jones-Jack-1 |

5. In some cases, you may want to delete the source columns for the primary key.



# Add Lookup Data

## Contents:

- *Set up Your Lookup Data*
- *Perform the Lookup*
- *Example - Lookup for Timezones*

You can integrate data from other sources into your current dataset. Based on a key column that you identify in the lookup dataset, you can insert the corresponding values in other columns of the lookup dataset as new columns in your source dataset.

**Tip:** Column lookups are useful for adding reference data based on a column's values.

For example, your data contains the two-letter abbreviations for U.S. states, yet the target system is expecting the full name of each state. You need to replace the XY state abbreviation with the full name of each state in each row.

## Set up Your Lookup Data

Your data table should look like the following:

| State-2Letter | State                |
|---------------|----------------------|
| AL            | Alabama              |
| AK            | Alaska               |
| AZ            | Arizona              |
| AR            | Arkansas             |
| CA            | California           |
| CO            | Colorado             |
| CT            | Connecticut          |
| DE            | Delaware             |
| DC            | District of Columbia |
| FL            | Florida              |
| GA            | Georgia              |
| HI            | Hawaii               |
| ID            | Idaho                |
| IL            | Illinois             |
| IN            | Indiana              |
| IA            | Iowa                 |
| KS            | Kansas               |
| KY            | Kentucky             |
| LA            | Louisiana            |
|               |                      |

|    |                |
|----|----------------|
| ME | Maine          |
| MD | Maryland       |
| MA | Massachusetts  |
| MI | Michigan       |
| MN | Minnesota      |
| MS | Mississippi    |
| MO | Missouri       |
| MT | Montana        |
| NE | Nebraska       |
| NV | Nevada         |
| NH | New Hampshire  |
| NJ | New Jersey     |
| NM | New Mexico     |
| NY | New York       |
| NC | North Carolina |
| ND | North Dakota   |
| OH | Ohio           |
| OK | Oklahoma       |
| OR | Oregon         |
| PA | Pennsylvania   |
| RI | Rhode Island   |
| SC | South Carolina |
| SD | South Dakota   |
| TN | Tennessee      |
| TX | Texas          |
| UT | Utah           |
| VT | Vermont        |
| VA | Virginia       |
| WA | Washington     |
| WV | West Virginia  |
| WI | Wisconsin      |
| WY | Wyoming        |

**Tip:** You can download a version of this table, which also includes some timezone information. See *Dict-TimezoneByState.csv*.

This data table must be uploaded as a new dataset. See *Import Data Page*.

## Perform the Lookup

### Steps:

1. In the Transformer page, click the drop-down on the column that contains your two-letter state abbreviations. Select **Lookup ....**
2. In the Lookup Wizard, select the dataset to use for your lookup.
3. For the lookup key, select the column in the dataset to use as the key value. In the above example, it is `State_2Letter`.
4. Click **Execute Lookup**.
5. The lookup key value is used to locate all of the other column values in the reference dataset. These values are inserted in separate columns to the immediate right of the source column.
6. You might need to delete some of the imported columns. In the above case, you might decide to delete the two-letter state identifier column, which has been replaced by the full state name column.

See *Lookup Wizard*.

## Example - Lookup for Timezones

The CSV linked above also contains timezone information for each state, which you can use to provide higher fidelity information on timestamps.

**U.S. timezones are not consistently demarcated by state lines. Some states are split across multiple timezones. For more accurate representation of timezones, you should download and use a zipcode database, many of which are freely available online. This CSV is provided for demonstration purposes only.**

In this case, you are working with a dataset that contains timestamps, which are stored in different timezones based on the location where an event or transaction occurred. However, the timestamps do not contain any timezone information.

You can use an external source of timezone information to insert timezones into your dataset. In the following example, timezones are derived based on two-letter abbreviations for U.S. state. A more accurate representation would be based on zipcode data.

### Steps:

1. Complete steps 1-5 in the previous section.
2. Delete all columns except the one containing timezone information. The `Time Offsets` column identifies the predominant timezone in each state as an offset of the UTC timezone (Greenwich Mean Time).
3. Move this column to the right of the column containing your timestamps.

**NOTE:** Depending on the requirements of your target system, you can use the `split` transform to break up column data so that only the numerical offset (e.g. `-6:00`) is present. Then, you can use the `DATEDIF` function to apply the timezone offset to your timestamps. In this manner, you can convert timestamps to the source timezone before they are consumed by the target system.

# Append Datasets

If you are wrangling datasets that represent transactional or serialized data, you can append together slices of data to build a larger dataset for richer analysis. For example, you are cleansing log messages on a weekly basis. You can create separate datasets for each day's log messages and then bring them altogether into a single dataset for processing through a single recipe. This method works best for datasets that have identical or very similar structures.

Below, you can see two datasets of contact information. These simplified datasets track customer contact records.

## Dataset01:

| Name        | Email                   | Last Contact |
|-------------|-------------------------|--------------|
| Jack Jones  | jack@example.com        | 06/15/2015   |
| Tina Toms   | tinat@example.com       | 08/02/2015   |
| Larry Lyons | larry.lyons@example.com | 03/22/2015   |

## Dataset02:

| Name           | Last Contact Date | Email                  |
|----------------|-------------------|------------------------|
| Amy Abrams     | 07/24/2015        | amy.abrams@example.com |
| Tina Toms      | 05/12/2015        | tinat@example.com      |
| Samantha Smith | 04/22/2015        | samantha@example.com   |

## Notes:

- There is one overlapping record for Tina Toms.
- There is a mismatch in one column name ("Last Contact" vs. "Last Contact Date").
- The columns are in a different order.

## Steps:

1. Load your first dataset (Dataset01).
2. In the recipe panel, add a step. In the Transformation textbox, enter `union`.
3. In the Union page, you bring together two or more datasets based on a shared set of fields.
  - a. A **union** operation appends datasets together. For more information, see *Union Page*.
4. To add another dataset, click **Add datasets**. Navigate to select the file to add to the union (Dataset02).
5. Initially, fields are mapped based on the column names. However, in this example, the `Last_Contact_Date` field from Dataset02 is not included. You can:
  - a. Click the + icon next to the `Last_Contact_Date` field in the left panel. The field is added as a separate field. However, it is not matched with the other contact date field from the original dataset.
  - b. From the Match columns drop-down menu, select **By Position**. In this case, you can see that there are only three fields, but the order is mismatched.

**Tip:** When possible, you should try to rename or align columns in your datasets prior to building a Union transformation step. Otherwise, you might have to edit the columns after the union has been completed.

To rename a column, click **Rename** from the column drop-down in the Transformer page. You can use the same drop-down to move a column.

6. In this case, you can cancel the union and reposition the `Email` column after the `Last Contact` column in `Dataset01`.
7. Then, open the Union page again and add `Dataset02`. Select **By Position** from the Match columns drop-down menu. Your columns are matched.
8. Click **Add to Recipe**.

`Dataset02` records have now been added to `Dataset01`, which now contains all of the records from both datasets. Note that the record for Tina Toms appears twice in the appended dataset.

- If the appended dataset is a record of all contacts, you should leave the duplicate record in place.
- If the appended dataset is a record of the most recent contact with each customer, you should remove the duplicate record. For more information, see *Deduplicate Data*.

**NOTE:** Be sure to verify that the data type for each column is accurate.

# Join Data

## Contents:

- Overview
  - Create Join
    - Step - Choose dataset or recipe to join
    - Step - Choose join keys and conditions
    - Step - Specify output columns for the join
    - Step - Review join
  - Modify Keys and Conditions
    - Ignore special characters
    - Create fuzzy join
    - Create range join
    - Add multiple join keys
- 

You can join together data based on the presence of one or more keys in your source dataset and the joined-in dataset or recipe. A **join** is a data operation in which two or more tables or datasets are merged into one based on the presence of matching values in one or more key columns that you specify. These shared columns are called the **join keys** of the two sets of rows that you are attempting to join.

## Overview

You can join a recipe or dataset to any of the following objects:

- Another recipe
- An imported dataset
- A reference dataset

For more information on the interface for building joins, see *Join Window*.

## Create Join

You can join datasets through the following mechanisms:

- **Flow View:** Select a dataset or recipe object. Right-click and select **Append Join**.

**Tip:** The join you specify from Flow View is added as the last step to the recipe. If you selected a dataset to which to add the join, a recipe is created from the object, and the join is added as the first step of the new recipe.

See *Flow View Page*.

- **Transform Builder:** Search for and select `join` datasets. See *Transform Builder*.

Joins are created through a workflow process described below. For more information on these steps, see *Join Window*.

## Step - Choose dataset or recipe to join

### Steps:

1. In the Choose dataset or recipe panel:

- a. Search for a dataset or recipe to which you have access. Your search includes objects outside of the current flow.
  - b. You can also select from:
    - i. Recipes in your current flow
    - ii. Datasets in your current flow
    - iii. All datasets to which you have access.
2. When you have found the dataset to use in your join, click **Accept**.

## Step - Choose join keys and conditions

### Steps:

1. Next, you select the join key columns and other conditions from each dataset.
2. **Join type:** Select the type of join to apply. See "Join types" below.
3. **Join keys:** The application attempts to find the best columns to match as the join keys.
  - a. Mouse over the percentage match to get more detailed statistics.

**NOTE:** For formatted data types, such as Datetime, the formatting of the join keys must also match. For example, the values 2021-01-01 and January 01, 2021 may not be interpreted as matching values.

- b. To change a join key, mouse over the key name and then click the Pencil icon. Select your new key.
  - c. For more information on the options, see "Modify Keys and Conditions" below.
  - d. Click **Save & Continue**.
4. Click **Next**.

### Example datasets

For discussion purposes, the following datasets are referenced in the sections below.

- The `CustId` column is shared between both datasets. This column is the **join key**, as there are no matches between the other columns.
- Some values in `CustId` in one dataset do not appear in the other.

### Dataset A:

The first dataset to which you are joining in another is typically called the **left dataset**.

| CustId | LastName | FirstName |
|--------|----------|-----------|
| c001   | Jones    | Jack      |
| c002   | Kim      | Ken       |
| c003   | Lee      | Larry     |
| c004   | Miller   | Mike      |
| c005   |          |           |

### Dataset B:

The second dataset that you are joining in to the first is typically called the **right dataset**.

| CustId | Region | CompanyName  |
|--------|--------|--------------|
| c002   | East   | ACME, Inc.   |
| c003   | West   | Trifax, Inc. |
| c005   | North  | Example Co.  |
|        |        |              |

|      |       |                |
|------|-------|----------------|
| c006 | South | Ace Industries |
|------|-------|----------------|

## Join types

There are multiple types of joins, which generate very different results. When you perform a join, you specify the type of join that is applied. The joined-together rows that appear in the output dataset are determined by the type of join that you selected and matching of values in the join key columns.

The following are the basic join types. The Example column references Dataset A (left) and Dataset B (right) from above.

| Join Type  | Description                                                                                                                                                                                                                                                | Example                                                                                                                                         |
|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| inner join | If a join key value appears in the left dataset and the right dataset, the joined rows are included in the output dataset.                                                                                                                                 | In the above output, rows c002 and c003 are included only.                                                                                      |
| left join  | In a left join, all of the rows that appear in the left dataset appear in the output, even if there is no matching join key value in the right dataset.                                                                                                    | In the above output, rows c001, c002, c003, c004, and c005 are included.<br><br>Rows c006 is excluded.                                          |
| right join | In a right join, all of the rows that appear in the right dataset appear in the output, even if there is no matching join key value in the right dataset.                                                                                                  | In the above output, rows c002, c003, c005, and c006 are included.<br><br>Rows c001 and c004 are excluded.                                      |
| outer join | An outer join combines the effects of a left and a right join. Each key value from both datasets is included in the output. If the key value is not present in one of the datasets, then null values are written into the columns from that dataset.       | In the above output, rows c001, c002, c003, c004, c005, and c006 are included.<br><br>Rows c001, c004, c005, and c006 contain some null values. |
| cross join | A cross join matches every row in the source dataset with a row in the joined-in dataset, regardless of whether the join keys match.<br><br><b>NOTE:</b> A cross join can greatly expand the number of rows in your dataset, which may impact performance. | If Dataset A has 5 rows and Dataset B has 4 rows, the output has 20 rows.                                                                       |
| self join  | A self join matches the rows in the left dataset with a version of itself (dataset or recipe) on the right side. Some limitations apply.                                                                                                                   |                                                                                                                                                 |

For more information, see *Join Types*.

## Step - Specify output columns for the join

### Steps:

1. In the Output columns step, you can specify the columns to include in the output dataset.
  - a. Include All: To include all columns from the left and right datasets, click the checkbox below All.
  - b. Use the Search box to search for specific columns to include or exclude.
2. Advanced options: See below.
3. Click **Review**.

### Apply prefix for column names

In the output dataset, the column names are taken directly from the column names in the source dataset.

Potential issues:



- In some cases, source column names may be an exact match between datasets.
- For development purposes, you may wish to track the source of a column for a period of time.

You can apply a prefix to the column names that are sourced from the left dataset, the right dataset, or both.

- **Name prefix for columns in Current data:** Enter a text value to include as the prefix to any output columns that are sourced from the current (left) dataset. For example, you could enter `left_`.
- **Name prefix for columns in Joined-In data:** Enter a text value to include as the prefix to any output columns that are sourced from the joined-in (right) dataset. For example, you could enter `right_`.

### Apply dynamic updates of selected columns

In the recipe step that produces the join, the columns that you select are mentioned specifically by name. Optionally, you can choose to automatically add in all columns to your output. For example, if your source data for an imported dataset is augmented with 10 new columns, when you re-run your join, those new columns can be automatically added to the output dataset.

**Tip:** You should consider using these options if the schema of your data sources is likely to change in the future.

- **Include all columns from Current data:** When selected, all columns that are subsequently added to the Current (left) dataset are automatically included as part of the join.
- **Include all columns from Joined-In data:** When selected, all columns that are subsequently added to the Joined-In (right) dataset are automatically included as part of the join.

### Step - Review join

#### Steps:

1. In the Review step, you can verify that the specified join is as you expected.
2. You should review the columns that are previewed as in the data grid.
3. To add the join as a recipe step, click **Add to Recipe**.

### Modify Keys and Conditions

**NOTE:** If you modify the selected dataset to join, the joined dataset, the join keys, or the fields to include in the output, subsequent steps in your transform recipe can be broken by the change. After you modify the join, you should select the last step in your recipe to validate all steps in the recipe.

You can apply the following modifications to how keys are matched. To modify a join key and condition, click the Pencil icon in the Join Keys & Conditions panel.

### Ignore special characters

Optionally, you can configure the Designer Cloud application to ignore the following special characters, when matching values in join keys:

- **Ignore case:** Ignore differences in case between values in the join key columns. `MyValue` matches with `MYVALUE`.
- **Ignore special characters:** Ignore special characters that appear in the join key values.
- **Ignore whitespace:** Ignore spaces, tabs, and other whitespace values that may appear in join key values.

### Create fuzzy join

A **fuzzy join** applies a fuzzy matching algorithm to String values in the join key column to account for slight differences in how values are written.

**NOTE:** Fuzzy joins can only be applied to String data types. Other data types cannot be fuzzy-matched using the algorithm.

This algorithm relies on the doublemetaphone function, which attempts to normalize text values based on how the string is spoken by an English speaker. For more information, see [https://en.wikipedia.org/wiki/Metaphone#Double\\_Metaphone](https://en.wikipedia.org/wiki/Metaphone#Double_Metaphone).

- **Fuzzy match:** Enable fuzzy matching based on English language pronunciation using the doublemetaphone function.

### Create range join

**NOTE:** This feature may need to be enabled in your environment. See *Workspace Settings Page*.

Values in the join key columns are matched across a range of values, instead of matching single value to single value. When range joins are enabled, you can set the Condition value between the two join key columns when specifying the join keys. For more information, see *Configure Range Join*.

### Add multiple join keys

For more complex join operations, you can add additional join keys to evaluate. Multi-key joins can be helpful for:

- Providing more finely specified join keys. For example, `lastName` and `firstName`.
- Performance

To add a second join key, click **Add** when modifying the join keys and conditions. Specify the keys in each dataset as needed.

# Configure Range Join

In most join operations, the values in primary keys across two tables must match exactly for the related columns to be included in the join. In a **range join**, you can change the comparative operator for the keys from Equal to one of the following additional comparisons:

- Not equal to
- Greater than
- Greater than or equal to
- Less than
- Less than or equal to

**NOTE:** A Trifacta administrator may need to enable this feature in your environment. See *Workspace Settings Page*.

## Limitations:

**Range joins allow you to include many more matching values and therefore rows in the join. Depending on the matches and the included columns, your resulting dataset can become very large. You should use this feature with some caution.**

- Range joins apply only keys whose data types can be compared.
  - For example, for joins involving keys of Binary data type, you can use Equal to or Not equal to joins.

**Tip:** Range joins cannot be applied to Datetime data type values directly. However, if you convert the values to numeric Unix time values, you should be able to specify a range join. For more information, see *UNIXTIME Function*.

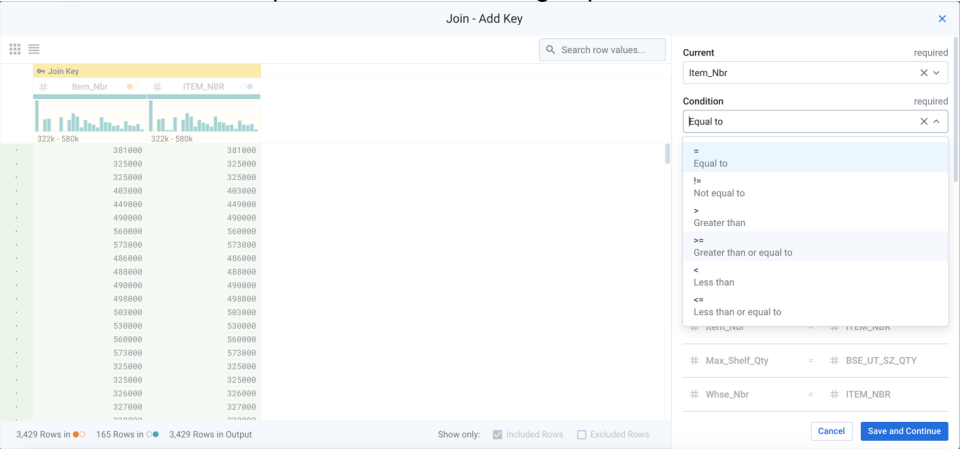
- Any range comparison that includes one or more string columns as keys uses the string comparison greater/less than, not the numerical comparison.

After range joins have been enabled, you can specify them as part of performing any join operation.

## Steps:

1. In the Search panel, enter `join datasets` in the search box.
2. Select the dataset with which to join the current one. Then, click **Accept**.
3. In the Join window, select the join type.
4. In the Join Keys area, click the Pencil icon.
5. Specify the fields in the current dataset and the joined-in dataset.

6. From the Condition drop-down, select the range operator to use:



**Figure: Select range operator**

- 7. Specify other properties for the matching keys.
- 8. Click **Save and Continue**.
- 9. Specify other elements of the join. When finished, click **Add to Recipe**.

For more information, see *Join Window*.

# Insert Metadata

## Contents:

- *Insert filepath*
- *Insert source row number*
- *Insert a single metadata column*
- *Insert multiple columns of metadata*

**Metadata** is data about your data. For example, you might decide that one or more of the following types of information about your dataset should be tracked:

- Source system(s)
- Source filepath and filename
- Source creation date
- Date of import
- Date of wrangling
- Name of person who performed the wrangling

This section provides some methods for how to insert metadata into your dataset.

## Insert filepath

For file-based data sources, you can insert the path to the source file in your dataset using the `$filepath` reference.

**Tip:** Filepath information can be lost when multi-dataset operations, such as unions and joins, are performed on your dataset. These steps should be added very early in your recipe.

In your recipe, insert the following transformation:

|                            |                                |
|----------------------------|--------------------------------|
| Transformation Name        | New formula                    |
| Parameter: Formula type    | Single row formula             |
| Parameter: Formula         | <code>\$filepath</code>        |
| Parameter: New column name | <code>sourceDatasetPath</code> |

For more information, see *Source Metadata References*.

## Insert source row number

You can insert the row number in the source file from which rows in your dataset are sourced, using the `$source rownumber` reference.

**Tip:** Source row number information can be lost when multi-dataset operations, such as unions and joins, are performed on your dataset. These steps should be added very early in your recipe.

In your recipe, insert the following transformation:

|                                   |                    |
|-----------------------------------|--------------------|
| <b>Transformation Name</b>        | New formula        |
| <b>Parameter: Formula type</b>    | Single row formula |
| <b>Parameter: Formula</b>         | \$sourcerownumber  |
| <b>Parameter: New column name</b> | sourceRowNumber    |

For more information, see *Source Metadata References*.

**Tip:** You can derive the current row number in your dataset. For more information, see *ROWNUMBER Function*.

## Insert a single metadata column

The following example describes how to insert a single column of metadata. In this case, the full path to the source is inserted as a new column in the dataset.

### Steps:

1. In the Dataset page, locate the imported dataset that is the source for your recipe. Click the Imported filter to show only the imported datasets.
2. For the imported dataset, click **Details**.
3. In the Dataset Details page, select the entire value for the Location, which is the storage location of the source.

**Tip:** If the full path of the dataset is too long for screen display, be sure to include the ellipsis (...) at the end of the Location value.

4. Copy the value. Paste the value into a text editor. You should see the full path, like the following:

```
<root_dir>/uploads/1/2580298d-3477-4907-bfa7-f71978eace04/SF Restaurants - businesses.csv
```

5. Load the dataset in the Transformer page.
6. Specify the following transformation:

|                                   |                                                                                                 |
|-----------------------------------|-------------------------------------------------------------------------------------------------|
| <b>Transformation Name</b>        | New formula                                                                                     |
| <b>Parameter: Formula type</b>    | Single row formula                                                                              |
| <b>Parameter: Formula</b>         | '<root_dir>\\uploads\\1\\2580298d-3477-4907-bfa7-f71978eace04\\SF Restaurants - businesses.csv' |
| <b>Parameter: New column name</b> | datasetPath                                                                                     |

## Insert multiple columns of metadata

You might need to track more fields of dataset information. While you might be able to perform these kinds of individual inserts, it might be easier to build this information from a separate file.

**NOTE:** This method uses the FILL function, which should be limited to smaller datasets when applied with a single key. Otherwise, there might be performance impacts when running the job against the full dataset.

**Tip:** You can perform a similar merging of datasets using the Join tool. See *Join Window*.

For example, you want to track the following fields as metadata:

- source\_system
- source\_author
- source\_date\_create

You could create a CSV file that looks like the following:

```
source_system,source_author,source_date_create
Excel,Joe Guy,12/9/15
```

In this case, the column headers are in the first line, and the values for each column are in the second line.

#### Steps:

1. Use your CSV file as the source for a new dataset within the flow containing the associated dataset.
2. In the data grid, make sure that the first line of data is treated as the header. If not, add a `header` transform to your recipe.
3. Open the other (source) dataset in the Transformer page.
4. In the recipe panel of the Transformer page, add a new step. In the Transformation textbox, enter `union`.
5. Create a union:
  - a. Include all columns from both datasets.
  - b. Configure the step to perform the union by name, instead of by position.
  - c. See *Union Page*.
6. Add this step to your recipe.
7. You should see one row in the union recipe that contains the new data.
8. Sort your data by a key value (e.g. `business_id`).
9. Determine an appropriate grouping parameter. This step is necessary to simplify the filling process when the job runs at scale. Ideally, you should choose a grouping column that contains a relative few number of values in it (e.g. `region`).
10. Fill values in the data rows with metadata column values. For each metadata column, add the following transformation, done here for the `source_system` column of metadata.

|                            |                                  |
|----------------------------|----------------------------------|
| <b>Transformation Name</b> | Window                           |
| <b>Parameter: Formula</b>  | <code>FILL(source_system)</code> |
| <b>Parameter: Group by</b> | <code>region</code>              |
| <b>Parameter: Order by</b> | <code>business_id</code>         |

11. Repeat the above step for each metadata column you want to insert.
12. Delete the source metadata columns.
13. Rename the `window` columns to use a more appropriate name.
14. Delete the row containing the original metadata values.

# Invoke External Function

## Contents:

- *Prerequisites*
- *Invoke*
- *Examples*
  - *ConcatUDF*
  - *AdderUDF*

---

Through the Search panel, you can access and apply functions that have been developed external to Designer Cloud powered by Trifacta® Enterprise Edition.

## Prerequisites

**Also known as user-defined functions, external functions must be developed in an environment external to the product and then registered for use in it. These steps require developer skills. For more information, see *User-Defined Functions*.**

## Invoke

After an external function has been registered with the product, you can complete the following steps to invoke the function within your recipe.

### Steps:

1. In the Transform Builder, you can search for any of the following:
  - a. `udf`
  - b. `invoke external function`

**NOTE:** You cannot search for the name of the external function.

2. Select **Invoke external function**.
3. The list of available external functions is displayed. Select the function to use.
4. Depending on the function, the following options may be available:
  - a. Columns: specify the column or columns to which to apply the function.
  - b. Arguments: If the function accepts arguments, you can enter them on individual lines.
  - c. New column name: Some functions generate a new column. Enter a new column name.
5. To add the instance of the function to your recipe, click **Add**.
6. The step is added to your recipe.

## Examples

You can create these examples functions in Java for use in the platform. For more information, see *Java UDFs*.

### ConcatUDF

The ConcatUDF function concatenates two strings together.



**Tip:** This function is provided for demonstration purposes only. In practice, you should use the MERGE function instead. See *MERGE Function*.

|                            |                          |
|----------------------------|--------------------------|
| Transformation Name        | Invoke external function |
| Parameter: Column          | colA, colB               |
| Parameter: Arguments       | (empty)                  |
| Parameter: New column name | myConcatUDFColumn        |

## AdderUDF

The AdderUDF function adds an input value to a constant that is submitted by parameter. The following invocation of AdderUDF adds colA and the constant 100.

**Tip:** This function is provided for demonstration purposes only. In practice, you should use the ADD function instead. See *ADD Function*.

|                            |                          |
|----------------------------|--------------------------|
| Transformation Name        | Invoke external function |
| Parameter: Column          | colA                     |
| Parameter: Arguments       | 100                      |
| Parameter: New column name | myAdderUDFColumn         |

# Publishing Tasks

These tasks provide information on the various methods of getting your data out of Designer Cloud powered by Trifacta® Enterprise Edition. These tasks include imported datasets, recipes, generated results, and work-in-progress versions of them.

# Create Outputs

## Contents:

- *Create an Output*
  - *Create a File-Based Output*
  - *Create a Table-Based Output*
  - *Create an Output With Parameters*
    - *Parameterize path or bucket name with a variable*
    - *Parameterize path with a timestamp*
  - *Edit an Output*
  - *Delete an Output*
- 

An output is defined as a set of files or tables, formats, and locations where results are written after a job run on the recipe has completed. To run a job from a flow, you must create an output object that defines where results are delivered after a job is successfully executed.

Every flow requires an output in order to publish results. An output object is composed of one or more publishing actions. A **publishing action** defines the output type, format, location, and other settings where results from a recipe are delivered.

You can create publishing actions in multiple formats and publish those to different databases and file storage formats. The following are the output types:

- File-based outputs such as CSV. For more information, see *Supported File Formats*.
- Table-based outputs such as Oracle or PostgreSQL. For more information, see *Connection Types*.

## Create an Output

You can use either of the following methods to create an output object and its related publishing action:

### From Flow View:

In Flow View, an output object extends from a recipe, indicating the results of the recipe are delivered to the output object.

1. Open your flow in Flow View.
2. In Flow View, you can:
  - a. Right-click a recipe. Select **Add Output to run**.
  - b. If an output already exists, select it.
3. The output is displayed in the Details panel on the right-side.
4. In the Details column under Manual Destinations, click **Edit**.
5. In the Publishing Settings page, click **Add Publishing Action**.

**Tip:** For scheduled runs of your flow, you must define output objects. You must specify output objects to automatically generate the output when the flow is executed by a schedule. For more information on scheduling, see *Overview of Automator*.

### From Run Job page:

You can also create outputs from the Run Job page.

1. In Flow View, click **Run Job**.

2. In the Run Job page, you can edit or add new outputs. To create a new one, click **Add Publishing Action**.

For more information on Flow View, see *Flow View Page*.

For more information on creating output objects, see *View for Outputs*.

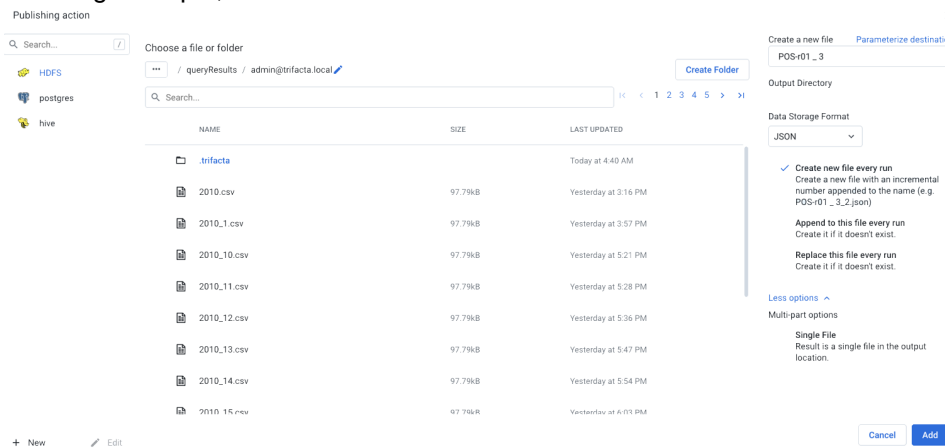
## Create a File-Based Output

You can create a file-based output by performing the following steps.

For more information on creating an output from Flow View and Run Job page, see above sections.

### Steps:

1. In the Publishing action page, select the connection where you wish to write file from the left panel. In the following example, the **HDFS** connection has been selected:



**Figure: Publishing action page for file output**

2. Select the file. You can select the existing file from the search list or click a **Create a new file** in the right panel.
  - a. Enter a file name in the **Create a new file** field.
3. To create output parameters, click the **Parameterize destination** link. See "Create an Output with Parameters" below.
4. From the **Data Storage Format** drop-down list, select the output format for the file.
5. The publishing actions vary based on the options selected. Select the required publishing actions below the drop-down list. For more information, see *Run Job Page*.
6. Update the **Delimiter** field, if required.
7. You can choose to generate the file as a **Single File** or as **Multiple Files**.
8. To apply compression to the file, select the compression type from the **Compression** drop-down list.
9. Click **Add**.

**Tip:** You can define SQL scripts that are executed before or after generation of your output objects. For more information, see *Create Output SQL Scripts*.

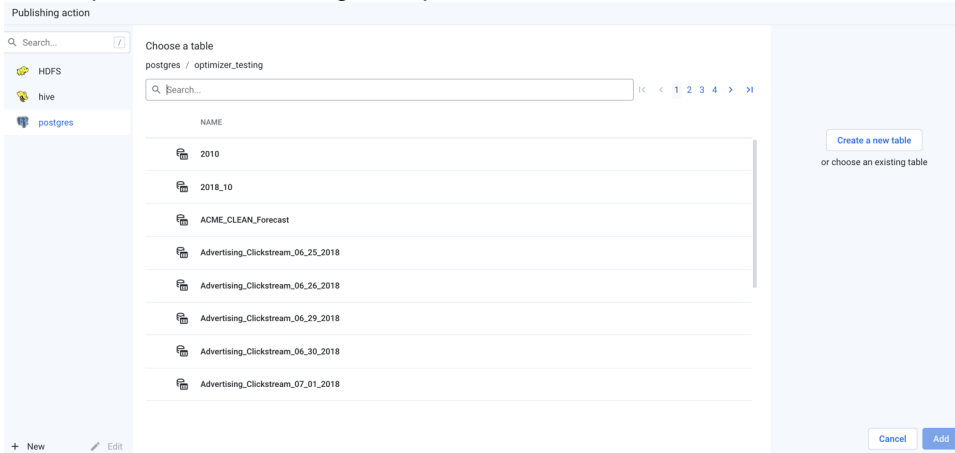
## Create a Table-Based Output

You can create output objects for publishing to tables by performing the following steps:

For more information on creating an output from Flow View and Run Job page, see above sections.

## Steps:

1. In the Publishing action page, select the connection to the database where you wish to store the table from the left panel. In the following example, the `postgres` connection is selected:



**Figure: Publishing action for a table output**

2. Search the table. You can select an existing table from the list or click **Create a new table** in the right panel.
  - a. Enter a table name in the **Create a new table** field.
3. To create output parameters, click the Parameterize destination link. See "Create an Output with Parameters" below.
4. Select the required publishing actions below the drop-down list. For more information, see *Run Job Page*.
5. Click **Add**.

**Tip:** You can define SQL scripts that are executed before or after generation of your output objects. For more information, see *Create Output SQL Scripts*.

## Create an Output With Parameters

For any outputs, you can parameterize elements of the output path. You can parameterize your path with the following options.

**Tip:** You can define multiple parameters per output.

- **Timestamps:** Inserts a formatted timestamp as part of the output path or filename
- **Variables:** Inserts a value for the variable.
  - This variable has a default value that you assign.
  - Whenever you execute a job through the Run Job page, you can pass in the default value or an override value for the variable.

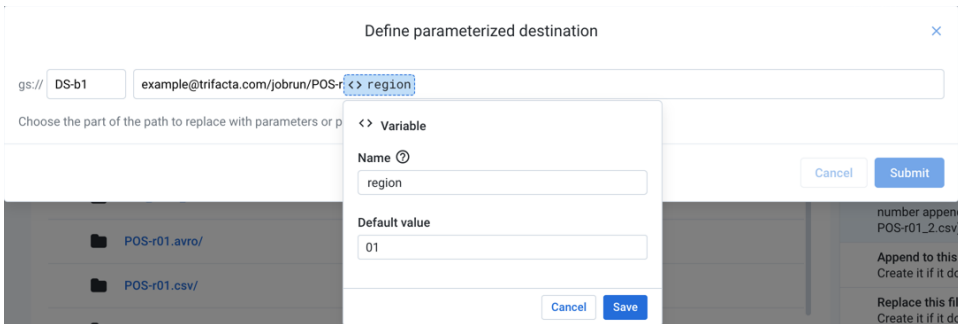
For more information on parameters, see *Overview of Parameterization*.

## Parameterize path or bucket name with a variable

For file- or table-based publishing actions, you can replace the bucket name (if applicable) or elements of the output path with variable values. When you define the output, you replace an element of the output path with the variable name. At runtime, the variable name is replaced by the appropriate value.

**Tip:** You can use environment parameters to parameterize bucket names across your environment. For more information, see *Environment Parameters Page*.

1. In the Publishing action page, click the **Parameterize destination** link. The Define Parameterized destination dialog is displayed.
2. On the listed output path, highlight the part that you wish to parameterize. You can select part of the path or bucket name.
3. Then, select **Add Variable**.



**Figure: Define parameterized destination**

- a. **Name:** Enter a display name for the variable.

**Tip:** Type `env.` to see the environment parameters that can be applied. These parameters are available for use by each user in the environment. For more information, see *Overview of Parameterization*.

**NOTE:** If multiple variables within a flow (or its dependent flows) have the same name then they are treated as the same variable.

- b. **Default value:** Enter a default value for the parameter.
4. Click **Save**.
  5. To save the parameters for the output path, click **Submit**.

The created parameter is displayed in the right context menu of the publishing action page.

**Tip:** If you created a variable parameter, you can apply override values to the variable when you are running a job. For example, you can modify a variable called `baseFileName` to generate an output with a different base filename for your job run. For more information, see *Overview of Parameterization*.

## Parameterize path with a timestamp

Timestamp parameters can be helpful when you want to create outputs based on date and time format, time zone, or exact and relative start time. For file- or table-based publishing actions, you can create outputs based on the specific region or time zone for which the data is generated. When you define the output, you can replace an element of the output path with the timestamp parameters.

### Steps:

1. In the Publishing action page, click the **Parameterize destination** link. The Define Parameterized destination dialog is displayed. See example above.

2. On the listed output path, highlight the part that you wish to parameterize. Then, select **Add Timestamp Parameter**.
3. In the Timestamp Parameter dialog, enter the following details:
  - a. **Timestamp format**: Specify the format for timestamp values.
    - i. Example: YYYY-MM-DD\_hh\_mm.
    - ii. Values can express both date and time elements. For more information on the available tokens for formatting date and time values, see *Datetime Data Type*.
  - b. **Timestamp value**: Select the value to record in the path:
    - i. **Exact job start date**: recorded timestamp in path is the start time of the job.
    - ii. **Relative to the job start date**: recorded timestamp in path is relative to the start time of the job according to the settings that you specify here.
  - c. **Time zone**: Click **Change** to change the time zone recorded in the timestamp.
    - i. Example: America/Los Angeles or Asia/Calcutta.
    - ii. For more information on the available time zones, see *Supported Time Zone Values*.
4. Click **Save**.
5. To save the specified parameter for the output path, click **Submit**.

The created parameter is displayed in the right context menu of the publishing action page.

## Edit an Output

### From Flow View page:

1. Right-click an output object. The object details are displayed in the Details panel.
2. In the Details panel, click **Edit**. The Publishing Actions page is displayed.
3. Make changes as needed in the Publishing Actions page. To save your changes, click **Update**.

### From Run Job page:

In the Run Job page, hover over the publishing action to modify. Click **Edit**.

## Delete an Output

You can delete the output object from the Flow View and from Run Jobs page:

### Flow View page:

1. In the Flow View, select the output.
2. In the right panel, select **Delete Output** from the context menu.

### Run Jobs page:

Select **Delete** from the context menu of the Publishing Actions.

For more information, see *Run Job Page*.

# Create Output SQL Scripts

## Contents:

- *Script Types*
    - *Script execution*
  - *Limitations*
  - *Enable*
  - *Create Output SQL Script*
    - *Parameterize values*
    - *Monitoring execution*
  - *Example Scripts*
    - *Example - log entries*
    - *Example - updates based on job results*
  - *Edit Output SQL Script*
  - *Delete Output SQL Script*
  - *Create Output SQL Script via API*
    - *Create SQL script*
    - *List SQL scripts*
    - *Edit SQL script*
    - *Delete SQL script*
- 

As part of job execution for an output, you can define SQL scripts to run before the job, after it, or both. These SQL scripts are stored as part of the output object definition and can be executed through any database connection to which the user has access. SQL scripts can be applied to file-based and table-based job executions.

- When flows are shared, the shared user can modify SQL Scripts if the user has Editor permissions on the flow. See *Overview of Sharing*.

## Example uses:

- Insert or update log entries in a database log table before or after a job that publishes to file or database destinations.
- Perform custom inserts, updates, and delete logic to other database tables based on job output data that is published to a database.
- Create and refresh tables or materialized views that join the job's output data with data from other tables using `CREATE AS SELECT`.
- Operational tasks such as disabling/enabling indexes and managing partitions on supported databases.

## Script Types

**NOTE:** If one of these scripted steps fails, then all downstream phases of the job also fail.

- **Pre-job:** After a job has been initiated and before data is ingested into the platform, a SQL script can be executed by the Designer Cloud application .
- **Post-job:** After job results have been generated and published, a SQL script can be executed.

**NOTE:** If publishing job fails, then all downstream tasks also fail, including the SQL script, which is not executed and is recorded as a failed phase of the job execution.



## Script execution

- SQL lines in an individual script are executed in the order listed in the script.
- If you have defined multiple scripts of the same type (pre-job, for example), those scripts may be executed in parallel.

**NOTE:** The order of listing of scripts in the Designer Cloud application does not affect the order of execution of those scripts.

## Limitations

**These SQL scripts are executed without validation through the selected database connection. There are no explicit limitations on the types of SQL statements that can be executed. It is possible to do harm through this feature.**

- After each SQL statement in a script, a semi-colon is required.
- SQL validation is not supported for some connection types.
- When flows containing output SQL scripts are imported, the connection to the database where the script is to be executed must exist in the new environment. Otherwise, the SQL script is dropped from the import. See *Import Flow*.
- Output SQL script actions may not be supported for all connection types. If the connection is not available in the dropdown for selecting a connection, then the feature may not be available for the connection type.
- Output SQL script actions are only supported for default connection types provided with the product. Custom connection types modified for a specific customer environment are not supported.

## Enable

This feature may need to be enabled in your environment.

A workspace administrator can enable the use of SQL scripts. For more information, see *Workspace Settings Page*.

## Create Output SQL Script

Through the Designer Cloud application, you add SQL scripts as part of the output object definition.

**Tip:** Depending on the nature of your SQL script, you may choose to test it first in a demo environment on a demo database.

### Where to add:

You can create SQL scripts for the following types of outputs:

- **Manual destinations:** In Flow View, you can select an output object and then modify one of its manual destinations. See *View for Outputs*.
- **Scheduled destinations:** In Flow View, select an output object and then modify one of its scheduled destinations. See *View for Outputs*.

### Steps:

1. In Flow View, select the output object for which you wish to create the SQL script.
2. In the Outputs panel on the right, click the Destinations tab.
3. For the type of destination, click **Edit**.

4. In the SQL Scripts panel at the bottom of the screen, click **Add Script**.
5. In the Add SQL Script window:
  - a. Select the database connection to use for executing the SQL script.
  - b. Enter the SQL script in the panel.
  - c. Choose when you would like to execute the script:
    - i. Run before data ingest - before the job is executed
    - ii. Run after data publish - after the job results have been written
  - d. Before you save your changes, click **Validate SQL**.

**NOTE:** Some connection types do not support SQL validation.

**NOTE:** Validating the SQL does not execute the SQL script on the database. It performs a check of SQL syntax against the selected database.

6. To save your SQL script, click **Add**.

For more information, see *SQL Scripts Panel*.

## Parameterize values

You can add variable or Datetime parameters to your SQL scripts.

- Parameters with the same name that are also defined on input datasets, flow parameters, and output objects can be referenced during job execution to pass the same value for consistency.

**Tip:** You can parameterize values in your SQL script. Parameters can be variables, Datetime parameters, or environment parameters. For more information, see *Overview of Parameterization*.

## Monitoring execution

You can monitor the execution of any SQL scripts that are part of a job execution.

- For more information, see *Job Details Page*.
- For more information, see *Jobs Page*.

## Example Scripts

In the following sections, you can review some common examples for how to use SQL scripts in your data pipelines.

### Example - log entries

In this example, you insert log entries into a log table in your database before and after the execution of your job.

#### Pre-job:

Your SQL script might look like the following:

```
CREATE TABLE IF NOT EXISTS "transactions"."log-tri" (
 timestamp date,
 jobType varchar(255),
 jobStatus varchar(255)
);
INSERT INTO "transactions"."log-tri"(timestamp, jobType, jobStatus)
VALUES ('2021-06-22', 'transformation', 'started');
```

The above script is composed of two statements:

1. **CREATE TABLE IF NOT EXISTS** - This statement creates the `log-tri` table in the transactions database.
  - a. This table is defined with three fields: `timestamp`, `jobType`, and `jobStatus`, each of which is assigned a data type.
  - b. The **IF NOT EXISTS** keyword ensures:
    - i. The table is created if it does not exist.
    - ii. If it exists, then no error is returned, which could stop the job run.

**INSERT INTO** - This statement inserts a record into the `log-tri` table, populating each column with an appropriate **VALUE**:

| Column name | Value            |
|-------------|------------------|
| timestamp   | '2021-06-22'     |
| jobType     | 'transformation' |
| jobStatus   | 'started'        |

**Tip:** In the above example, the value for the `timestamp` is a literal value. If needed, you can parameterize that value, so that a Datetime parameter can be inserted into the record as needed. See "Parameterize values" above.

## Post-job:

After the job results have been published, a post-job SQL script might look like the following:

```
CREATE TABLE IF NOT EXISTS "transactions"."log-tri" (
 timestamp date,
 jobType varchar(255),
 jobStatus varchar(255)
);
INSERT INTO "transactions"."log-tri"(timestamp, jobType, jobStatus)
VALUES ('2021-06-22', 'transformation', 'complete');
```

This script is very similar to the previous:

1. Create the table if it doesn't exist. This statement also provides schema information if you need to make modifications in the future.
2. Inserts a new row in the table, indicating the `transformation` job type is now `complete`.

## Example - updates based on job results

If you write your job results through the same connection where you are executing your SQL script, you can leverage the data directly from your job results into your SQL script.

In the following scenario, a customer account dimension table in the datawarehouse ( `dw.DimCustAccount custdim`) is updated with data enriched through Designer Cloud powered by Trifacta Enterprise Edition in the job results. In this case the `num_emp`, `industry_cd`, and `duns` columns are mapped to the corresponding columns in the `custenr` enriched data table with values where the customer identifier in the customer dimension table (`custdim.custId`) matches the customer identifier in the enriched data table (`custenr.custId`).

```
UPDATE TABLE dw.DimCustAccount custdim
SET num_emp = custenr.empcnt, industry_cd = custenr.ind_cd, duns = custenr.duns_num
FROM tri.cust_enriched custenr
WHERE custdim.custId = custenr.custId;
```

## Edit Output SQL Script

### Steps:

1. In Flow View, select the output object.
2. In the context panel on the right, select the Destinations tab.
3. Click **Edit** next to the type of destination to modify.
4. In the dialog, locate the one to modify in the SQL Scripts panel. Click **Edit**.
5. Make changes as need. Click **Save**.

## Delete Output SQL Script

**After you deleting a SQL script and save the output object, the SQL script is removed permanently. Before deleting, you may wish to copy the script and paste it into a text editor.**

### Steps:

1. In Flow View, select the output object.
2. In the context panel on the right, select the Destinations tab.
3. Click **Edit** next to the type of destination to modify.
4. In the dialog, hover over the one to modify in the SQL Scripts panel. From the More menu, select **Delete**.

## Create Output SQL Script via API

You can create SQL scripts via API. These scripts can then be associated with specific output objects.

## Create SQL script

### Key information:

| Attribute      | Description                                                                                                                                                              |
|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| sqlScript      | Text of the SQL script. You should validate this script before inserting it into the API.                                                                                |
| type           | Set type to be: <ul style="list-style-type: none"><li>• <code>pre</code> - execute before data ingest</li><li>• <code>post</code> - execute after data publish</li></ul> |
| vendor         | The vendor type of the database to which you are connecting. See <i>Connection Types</i> .                                                                               |
| outputObjectId | Internal identifier of the output object to which you are associating the SQL script. When the object is selected in Flow View, the identifier is part of the URL.       |
| connectionId   | Internal identifier of the connection that you are using to execute the SQL script.                                                                                      |

|          |                |
|----------|----------------|
| Endpoint | /v4/sqlScripts |
| Method   | POST           |

For more information, see <https://api.trifacta.com/ee/es.t/index.html#operation/createSqlScript>

## List SQL scripts

List all SQL scripts.

|          |                |
|----------|----------------|
| Endpoint | /v4/sqlScripts |
| Method   | GET            |

For more information, see <https://api.trifacta.com/ee/es.t/index.html#operation/listSqlScripts>

## Edit SQL script

|          |                     |
|----------|---------------------|
| Endpoint | /v4/sqlScripts/{id} |
| Method   | PATCH               |

where:

- {id} is the internal identifier of the SQL script

For more information, see <https://api.trifacta.com/ee/es.t/index.html#operation/patchSqlScript>

## Delete SQL script

|          |                     |
|----------|---------------------|
| Endpoint | /v4/sqlScripts/{id} |
| Method   | DELETE              |

where:

- {id} is the internal identifier of the SQL script

For more information, see <https://api.trifacta.com/ee/es.t/index.html#operation/deleteSqlScript>

# Publish Results on Demand

After a set of results have been generated from a job, you can export those results to different environments in different formats as long as the job remains in Designer Cloud powered by Trifacta® Enterprise Edition. This feature is also known as **ad-hoc publishing**.

## Steps:

- In the left menubar, click the Jobs icon.
- In the Jobs page, select the job whose results you wish to publish. For more information, see *Job Details Page*.

**Tip:** If you enabled profiling of the results, you can click the job identifier to open the visual profile of the job results.

**Tip:** If you know the recipe and flow from which the job was executed, you can also select the recipe in Flow View and then select the output object for that recipe. In the right panel, you can click the job identifier in the Jobs tab to open the job results for export.

- Click the Output Destinations tab.
- Publishing:
  - **Export file:** Click one of the links to the generated output files to download the results in that file format. If you do not see your preferred export format, please rerun the job.
  - **File path:** You can use the provided file path to get the export from the backend datastore outside of the application.
  - **Create dataset:** From the context menu for the output, select **Create imported dataset** to turn the results into a new imported dataset in Designer Cloud powered by Trifacta Enterprise Edition.

**Tip:** In Flow View, you can create a reference object for any recipe in your flow. This reference object makes the output of the recipe available as an input object in other flows. So, you can use this as a method of creating a new dataset from the output of your recipe that is automatically updated without having to regenerate the imported dataset.

- **Publish:** Depending on the connections to which you have access, you can write your job's results to an external datastore. Some datastores do not permit writing the same job results a second time. For more information, see *Publishing Dialog*.
- Close the window when you are done.

# Reuse Recipe

## Contents:

- *Reuse Recipe with the Same Flow*
  - *Reuse Copy of Recipe in the Same Flow*
  - *Move or Copy Recipe to a Different Flow*
  - *Reuse Recipe in a Different Environment*
  - *Download Recipe*
- 

## Reuse Recipe with the Same Flow

The easiest way to reuse a recipe is to change its inputs in Flow View.

**NOTE:** You can create variation in the paths to your imported datasets. For example, if you have File01.csv and File02.csv, you can create a single dataset with parameters to capture both files and apply the same recipe to it. For more information, see *Create Dataset with Parameters*.

### Steps:

1. Open the flow containing the recipe.
2. Select the recipe that you'd like to reuse.
3. From the recipe's context menu, select **Change input....** Select the object to be the input to the recipe.

## Reuse Copy of Recipe in the Same Flow

You can also create copies of recipes within the same flow. This step also copies:

- All outputs attached to the recipe.
- (optionally) All inputs to the recipe.

### Steps:

1. In Flow View, select the recipe to copy.
2. In the context panel on the right, select **Make a copy > Without inputs**.
3. The recipe is copied and added to your flow.
4. To apply the recipe to a dataset, select from the new recipe's context menu, **Change input....**

## Move or Copy Recipe to a Different Flow

You can move a recipe from the current flow a different one. These steps move a recipe from one flow to another.

- If you want to reuse the recipe in a different flow, create a copy of it first. See above.
- In some cases, it may be easier to duplicate the whole flow and then remove objects from the copied flow. For more information on duplicating your flow, see *Flows Page*.

**Tip:** When you move a recipe to a new flow, all attached objects appear in the new flow. If the same objects in the source are used by other recipes, then copies are moved. If the copied object already exists in the target flow, the moved recipe is attached to the corresponding object in the new flow.

### Steps:

1. In Flow View, select the recipe to move.
2. In the context panel on the right, select **Move....**
  - a. To move to a new empty flow, select `Create New Flow`. You can specify a name for the new flow.
  - b. To move to an existing flow to which you have access, select the flow from the drop-down.
3. Click the **Move** button.
4. The recipe is moved, along with any related objects.

### Reuse Recipe in a Different Environment

If you need to reuse a recipe in a different instance of Designer Cloud powered by Trifacta® Enterprise Edition, you have two choices:

1. Export the entire flow and import it into the new environment. Open the flow in the new environment. In Flow View, remove all objects that are not of interest. See *Export Flow*.
2. Turn all of the steps of a recipe into a macro. Export the macro and then import into the new environment. You may choose to remove the macro from the original environment. See *Export Macro*.

### Download Recipe

You can download a recipe in text form in the following ways:

**NOTE:** A downloaded recipe is in a text form of Wrangle (a domain-specific language for data transformation). In this form, it cannot be used in the application. Downloaded recipes are for archival purposes only.

- In Flow View, select the recipe to download. From the context menu, select **Download recipe....** See *Flow View Page*.
- In the Recipe panel in the Transformer page, click the context menu, and select **Download recipe as Wrangle**. See *Recipe Panel*.



# Project Management Tasks

These topics provide guidance on how to better manage your projects in Designer Cloud powered by Trifacta® Enterprise Edition.

# Take a Snapshot

## Contents:

- *Duplicate*
    - *Flows*
    - *Recipes*
  - *Download Work in Progress*
    - *Download Sample Data*
    - *Download Recipe*
  - *Backup*
- 

You can use the following techniques to capture snapshots of your Designer Cloud® application work in progress.

## Duplicate

You can make a copy of individual recipes and flows.

**NOTE:** Copied recipes and datasets are independent objects and do not continue to inherit any changes in the original.

## Flows

In Flow View, click the context menu and select **Duplicate**.

**NOTE:** Sharing permissions are not inherited in the copied flow. You must re-share the flow with any users who need access to the copy.

## Recipes

In Flow View, select a recipe to duplicate. In the right panel, select **Make a copy** from the context menu. You can link the recipe to the same inputs or to no inputs.

**NOTE:** This recipe is still available to all who have access to the flow. If needed, select **Move** to relocate the copied recipe to another flow to which other users do not have access.

Select the copied recipe and click **Edit Recipe** to begin working with the recipe in the Transformer page.

See *Flow View Page*.

## Download Work in Progress

From the Recipe panel in the context panel, you can download your work in progress, including the recipe and the dataset sample as reflected in the current recipe step.

## Download Sample Data

From the Transformer page, you can download the dataset sample as it is currently reflected in the Transformer page.

**NOTE:** A sample downloaded from the Transformer page reflects all recipe steps up to the step that is currently selected. Steps that occur after the current one are not applied to the dataset sample.

**Tip:** You can use this as a work-in-progress backup if you select the final step of the recipe and if the dataset sample represents the entire dataset.

From the Recipe panel, click the context menu and select **Download Sample data as CSV**.

The CSV file is written to your desktop.

## Download Recipe

In the Recipe panel, click the context menu and select **Download recipe as Wrangle**.

The entire recipe is downloaded to your desktop as a text file.

**Tip:** If you are attempting to capture the recipe as a work-in-progress of the dataset sample, you can just delete the steps that aren't executed from the downloaded file.

See *Recipe Panel*.

## Backup

Backups of the Trifacta databases (flows, recipes, and other metadata) and source datastores (imported datasets) should be executed according to your enterprise requirements.

# Track Data Changes

Contents:

- *Create Backup*
- *Track Source Filepath and Filename*
- *Track Source Row Information*
- *Track Steps Affecting a Column*
- *Track Column Value Changes*
- *Track Row Changes*

## Create Backup

After you have created the flow and the datasets within the flow and before applying recipe steps to change the data, create a duplicate of the flow. This becomes a snapshot of your original dataset. Since the imported datasets are not affected, the storage overhead for creating backups is relatively low. See *Flow View Page*.

## Track Source Filepath and Filename

When you first load your dataset in the Transformer page, you can add the following to capture the full path to the original file that is the source of the data:

|                            |                    |
|----------------------------|--------------------|
| Transformation Name        | New formula        |
| Parameter: Formula type    | Single row formula |
| Parameter: Formula         | \$filepath         |
| Parameter: New column name | sourceRowNumber    |

With a few extra steps, you can extract the filename from the above output. For more information, see *Source Metadata References*.

## Track Source Row Information

You can mark the original row numbers of your source data. In the first step in your recipe after initial parsing, add the following:

|                            |                    |
|----------------------------|--------------------|
| Transformation Name        | New formula        |
| Parameter: Formula type    | Single row formula |
| Parameter: Formula         | \$sourcerownumber  |
| Parameter: New column name | sourceRowNumber    |

This step generates a new column that contains the source row number from the source dataset.

**NOTE:** Source row information can become invalid if you perform multi-dataset operations such as lookups, unions, and joins. For more precise tracking of source information, you should consider creating multi-column keys, including the source row number information. For more information, see *Generate Primary Keys*.

See *Source Metadata References*.

## Track Steps Affecting a Column

To see all of the steps in your current recipe that reference a specific column, select **Show related steps...** from the column menu.

All steps are highlighted in the Recipe panel.

**NOTE:** If another column is dependent on the selected column, all steps pertaining to that column are highlighted as well.

For more information, see *Column Menus*.

## Track Column Value Changes

Designer Cloud powered by Trifacta® Enterprise Edition enables you to easily move between steps in your transform recipe so that you can check the state of your dataset at any point during the transformation. In some cases, you may want to be able to track the changes made to an individual column side-by-side with the original column. This section provides a generalized approach for tracking column changes in this manner.

**NOTE:** Use this workflow only if it is important to monitor which values have changed in a column. For most use cases, the Transformer page provides sufficient visibility over your sample data to manage column values.

### Steps:

In the following sequence, the original column is called `String`. For numeric columns, you can perform more detailed analysis between original and modified column values.

1. After you have completed your general setup steps of your transform, create a copy of the original column:

|                                   |                    |
|-----------------------------------|--------------------|
| <b>Transformation Name</b>        | New formula        |
| <b>Parameter: Formula type</b>    | Single row formula |
| <b>Parameter: Formula</b>         | String             |
| <b>Parameter: New column name</b> | String_orig        |

2. You now have a copy of the original column before any manipulations were applied to it.
3. Add any transforms to your recipe, including any that change the values of `String`. In the example below, the following transform has been applied:

|                            |                   |
|----------------------------|-------------------|
| <b>Transformation Name</b> | Edit with formula |
| <b>Parameter: Columns</b>  | String            |
| <b>Parameter: Formula</b>  | TRIM(String)      |

- At the point in your recipe where you would like to test the column for changes, insert the following:

|                                   |                       |
|-----------------------------------|-----------------------|
| <b>Transformation Name</b>        | New formula           |
| <b>Parameter: Formula type</b>    | Single row formula    |
| <b>Parameter: Formula</b>         | String <> String_orig |
| <b>Parameter: New column name</b> | String_changes        |

- The `String_changes` column now contains true values where the values in `String` have been changed from their original values (`String_orig`).
- 

To see just the values that are different, sort in descending order.

**Tip:** You can reposition this test anywhere in your recipe after you have created the `String_orig` column.

- Before you run your recipe, you may want to remove the tracking columns that you generated (`String_orig` and `String_changes` in our example).

| String          | String_orig     | String_changes | Description                       |
|-----------------|-----------------|----------------|-----------------------------------|
| My String       | My String       | false          | "Base string: "My String""        |
| My String extra | My String extra | false          | "Base string + "" extra""         |
| My String       | My String       | true           | A space in front of base string   |
| My String       | My String       | true           | A space after base string         |
| MyString        | MyString        | false          | No space between the two words    |
| My String       | My String       | false          | Two spaces between the two word   |
| "My String"     | "My String"     | false          | Base string + a tab character     |
| "My String."    | "My String."    | false          | Base string + a return character  |
| "My String."    | "My String."    | false          | Base string + a newline character |

1 Rename all columns by converting row 1 to a header  
 2 Create String\_orig from String  
 3 Set String to TRIM(String)  
 4 Create String\_changes from String != String\_orig

4 Columns 9 Rows 2 Data Types

**Figure: Example tracking column changes**

## Track Row Changes

### Steps:

- Create a copy of the flow. In its name, identify that it is your original. See *Flow View Page*.
- In the other flow, create your recipes as normal.
- When done, you can add the following steps:
  - Union the two datasets together.
  - Sort them by a key column.
  - Add the deduplicate transform.

**NOTE:** This method may not work if your recipe includes joins or added or removed columns.

4. If the rows are exact duplicates, they are removed. The remaining rows contain data that has been changed.

# Add Comments to Your Recipe

As needed, you can insert non-functional comments in your recipes. These comments are stored as a Comment transformation but do not make changes to the dataset.

**Tip:** Adding comments to your recipes can be helpful for providing notes or other guidance to yourself for later or to other recipe builders who are reviewing your recipe.

## Steps:

1. In the Transformer page, open the Recipe panel in the context panel.
2. In your list of recipe steps, select the location in the recipe where you wish to insert the comment. From the recipe step context menu, select the appropriate Insert Step command.
3. In the Search panel, enter `comment`.

**Tip:** You can also paste full comments of the following format into the textbox. These comments are reformatted into the supported format:

```
// This is a comment.
```

```
/* This is also a comment. */
```

4. In the comment textbox, enter the comment that you would like to include.
5. Click **Save**.
6. The comment is stored in the recipe as text of a different color.



# Create Custom Data Types

## Contents:

- *Create dictionary file for custom data type*
- *Example - Sizes*
- *Enable*
- *Create the data type*
- *Use a custom data type*

**This method of creating custom data types is likely to be deprecated in a future release. Please consider switching to other types of custom data validation. For more information, see [Validate Your Data](#).**

A custom data type can be created when you create and upload a CSV dictionary file. This dictionary file includes all accepted values for the custom data type.

**NOTE:** The method described in this section validates against a fixed set of values. If you would like to validate your custom type against a pattern, you can specify the pattern using RegEx. See [Create Custom Data Types Using RegEx](#).

A **dictionary** represents one or more columns of reference data, which can be used for data validation. For defining custom types, if a value is included in the dictionary, it is a valid member of the custom type. For example, you may wish to create a custom type called `storeId`, which contains a valid identifiers for stores in your enterprise.

**After a custom type has been added, it cannot be removed or disabled.**

## Create dictionary file for custom data type

For your custom data type, you must create a dictionary of values in your local environment. This file is then uploaded to Designer Cloud powered by Trifacta® Enterprise Edition.

### File characteristics:

- CSV file format
- File can be multi-column, but data validation only uses one of the columns.
- File can contain up to 250,000 values. If your data type contains more values than this limit, you might see values in your dataset identified as members of the data type, when they are not.
- Remove any header row from your file.
- Values are case-insensitive during matching.
- Special restrictions on the newline (`\n`) character are described below.

### Notes and Limitations:

While you can use newline as a delimiter, dictionaries do not support using the newline (`\n`) character within a cell value. If your dictionary includes this character in cell values, it is dropped from use in the generated dictionary. In the following CSV example data, the first row is acceptable, while the second is not:

```
"Arizona"\n"Alaska"\n
"Arizona\n"\n"Alaska"
```

## Example - Sizes

For example, your data contains size information from Extra Small (XS) to Extra Extra Large (XXL). You can create a one-column dictionary file with values for these sizes on separate lines. This dictionary file could be used to validate the custom type `Sizes`. Your data might look like the following. Note that the column has no header.

|                   |
|-------------------|
| Extra Small       |
| Small             |
| Medium            |
| Large             |
| Extra Large       |
| Extra Extra Large |
| XS                |
| S                 |
| M                 |
| L                 |
| XL                |
| XXL               |
| Extra-Small       |
| Extra-Large       |
| Extra-Extra-Large |

You can download this source file: *Dict-Sizes.csv*.

## Enable

To begin, you must enable the use of custom data types:

1. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
2. Locate the following property:

```
"feature.enableCustomTypes": true,
```

3. Save your changes and restart the platform.

## Create the data type

Please complete the following steps to create the custom type.

**NOTE:** After a custom type has been created, a platform restart is required. Please contact your Trifacta administrator.

### Steps:

1. Click the data type drop-down in a column where you want to apply the custom type.
2. Click **More**. Scroll down and click **Custom Type**.

3. The Custom Type dialog is displayed. Click the Create New Custom Type tab.
4. Click **Upload Dictionary**. Select the CSV file you created. Click **Open**.

**NOTE:** After you upload a dictionary file, it cannot be removed. If necessary, upload a new version with a different filename.

5. The file is uploaded:

Use Existing Custom Type   Create New Custom Type   ✕

Name:

Dictionary Name Upload Dictionary

▼ Dict-Sizes.csv

| column1           |
|-------------------|
| Extra Small       |
| Small             |
| Medium            |
| Large             |
| Extra Large       |
| Extra Extra Large |
| XS                |
| S                 |
| M                 |
| L                 |
| XL                |

Cancel Save

**Figure: Custom Data Type dialog**

6. Click the caret next to the filename to review the contents of the dictionary.
7. Select one of the column headers in the left side of the dialog. On the right side, you can review values in the selected column.

**NOTE:** You must expand the custom dictionary to see values before you can save the custom type. This is a known issue.

8. Select the column you want to use for validating the custom type.
9. Enter a name for the data type.

**NOTE:** This name appears in the data type drop-down for each column. Also, it can be referenced explicitly in transforms that utilize a named data type as a parameter.

10. Click **Save**.
11. Restart the platform. See *Start and Stop the Platform*.

For more information, see *Custom Type Dialog*.

## Use a custom data type

### Steps:

1. Select **Custom Type** from the column drop-down.
2. In the Custom Type dialog, click the Use Existing Custom Type tab.

**NOTE:** If you cannot see a recently created custom data type, you may need to logout and login again.

3. Select the name of the custom type. Click **Save**.
4. When the data type is saved, the values in the column are validated against this custom type.
5. Make sure you review the missing and mismatched values for the column.

**Tip:** You can also reference the data type by name in your transforms.

For more information, see *Custom Type Dialog*.

# Create Target

## Contents:

- *Create Schema*
    - *Before you create*
    - *Create*
  - *Update Schema*
  - *Use Schema*
  - *Remove Schema*
- 

To assist in building your recipe, you can associate a target with the recipe. This target schema is displayed in the Target Matching bar in the Transformer page above your column histograms, so that you can track how you are progressing toward completion of the recipe.

- A **target** is the representation of the columns to which you are building your recipe to match. When you know the column order, names, and data types for which you are building your recipe, you can more quickly develop the recipe steps to match this schema.
- For more information, see *Overview of RapidTarget*.

## Create Schema

### Before you create

A target is created from one of the following sources:

- An imported dataset
- A recipe in the current flow
- A dataset reference from another flow

Before you create a new target, you must identify, create, or import one of the above objects. Your target must be in a finished state.

**NOTE:** A target is a snapshot of the target at the time of creation. It does not inherit changes from the source after creation. To update the target for later changes, you must delete and recreate the target. Instructions are provided below.

### Create

Each recipe can have one and only one target. Please use the following steps to create a target for your recipe.

#### Steps:

1. Open the flow containing the recipe. In Flow View, create or select the recipe.
2. If a target already exists for the recipe, select **Remove Target** from the context menu in the right panel.

**NOTE:** A recipe can have one and only one target associated with it.

3. After deleting the old one, from the context menu, select **Assign Target to Recipe**.
  4. Select the imported dataset, recipe, or reference to use as your target for this recipe. Click **Add**.
  5. If the target looks accurate, click **OK**. If not, click **Cancel**.
  6. The target is associated with your recipe.
-

**Tip:** You can also assign a target from the toolbar in the Transformer page. See *Transformer Toolbar*.

## Update Schema

**NOTE:** You cannot edit a target through Designer Cloud powered by Trifacta® Enterprise Edition. To make changes, remove the target and add in a modified target.

If there are have been changes to the source schema of your target, please complete the following steps to update your target.

### Steps:

1. If the source of the target needs to be re-imported into Designer Cloud powered by Trifacta Enterprise Edition, please do so now.
2. In Flow View, select the recipe to which the target is assigned. From the context menu for the recipe, select the following:
  - a. Select **Remove Target** to remove the current target.
  - b. Select **Assign Target to Recipe** to select the new target.

## Use Schema

After a schema has been associated with your recipe, schema information and a few example rows are displayed in the data grid of the Transformer page. These examples serve to guide your transformation operations. See *Data Grid Panel*.

In the Column Browser panel, you can select one or more columns and apply schema-related transformations to them. See *Column Browser Panel*.

## Remove Schema

### Steps:

1. In Flow View, select the recipe whose schema you wish to remove.
2. In the right panel, click the context menu. Select **Remove Target**.

**NOTE:** Removing a target from a recipe does not remove the underlying dataset from the platform.

**NOTE:** Deleting a dataset does not remove any target based off of it. You can still perform alignment operations to match the schema. However, you cannot view example rows from the target in the Transformer page.

# Optimize Job Processing

## Contents:

- *Filter data early*
  - *Perform joins early*
  - *Perform unions late*
  - *Run jobs on the default running environment*
- 

This page contains a set of tips for how to improve the overall performance of job execution.

### Filter data early

If you know that you are deleting some rows and columns from your dataset, add these transformation steps early in your recipe. This reduction simplifies working with the content through the application and, at execution, speeds the processing of the remaining valid data. Since you may be executing your job multiple times before it is finalized, it should also speed your development process.

- To delete columns:
  - Select **Delete** from the column drop-down for individual columns. See *Column Menus*.
  - Use the Delete Columns transformation to remove multiple discrete columns or ranges of columns.
- To delete rows: The following example removes all rows that lack a value for the `id` column:

|                      |                      |
|----------------------|----------------------|
| Transformation Name  | Filter rows          |
| Parameter: Condition | Is missing           |
| Parameter: Column    | id                   |
| Parameter: Action    | delete matching rows |

- To keep rows: The following example keeps all rows that lack a value in the `id` column:

|                      |                    |
|----------------------|--------------------|
| Transformation Name  | Filter rows        |
| Parameter: Condition | Is missing         |
| Parameter: Column    | id                 |
| Parameter: Action    | keep matching rows |

- See *Filter Data*.

### Perform joins early

After you have filtered out unneeded rows and columns, join operations should be performed in your recipe. These steps bring together your data into a single consistent dataset. By doing them early in the process, you reduce the chance of having changes to your join keys impacting the results of your join operations. See *Join Window*.

### Perform unions late

Union operations should generally be performed later in the recipe so that you have a small chance of changes to the union operation, including dataset refreshes, affecting the recipe and the output.

**NOTE:** If your dataset requires a significant amount of data cleaning, you should perform your unions early in your recipe, so that all cleaning steps can be applied once across the dataset.

See *Union Page*.

### **Run jobs on the default running environment**

When configuring a job, Designer Cloud powered by Trifacta Enterprise Edition analyzes the size of your dataset to determine the best of the available running environments on which to execute the job. This option is presented as the default option in the dialog. Unless you have specific reasons for doing otherwise, you should accept the default suggestion.



# Diagnose Failed Jobs

## Contents:

- *Job Types*
- *Identify Job Failures*
  - *Invalid file paths*
  - *Jobs that Hang*
  - *Spark Job Error Messages*
  - *Databricks Job Errors*
- *Try Other Job Options*
- *Review Logs*
  - *Hadoop logs*

## Job Types

The following types of jobs can be executed in Designer Cloud powered by Trifacta® Enterprise Edition:

- **Transform job:** This type of job executes the steps in your recipe against the dataset to generate results in the specified format. When you configure your job, any set of selected output formats causes a transform job to execute according to the job settings.
- **Profile job:** This type of job builds a visual profile of the generated results. When you configure your job, select **Profile Results** to generate a profile job.
- **Publish job:** This job publishes results generated by the platform to a different location or datastore.
- **Ingest job:** This job manages the import of data from a JDBC source into the default datastore for purposes of running a transform or sampling job.

For more information, see *Run Job Page*.

**Tip:** Information on failed plan executions is contained in the `orchestration-service.log` file, which can be acquired in the support bundle. For more information, see *Support Bundle Contents*.

## Identify Job Failures

When a job fails to execute, a failure message appears in following locations:

- Jobs tab in Flow View. See *Flow View Page*.
- Listing in the Jobs page. See *Jobs Page*.

The following is an example from the Jobs page:

|    |                |                               |                                                                |           |                                            |
|----|----------------|-------------------------------|----------------------------------------------------------------|-----------|--------------------------------------------|
| 35 | Automated Demo | airports_info<br>Airport Flow | Transform<br>Finished Today at 10:25 AM<br>Environment: Hadoop | Completed | Today at 10:26 AM<br>Ran for a minute      |
| 33 | Automated Demo | airports_info<br>Airport Flow | Profile<br>Finished Today at 10:25 AM                          | Failed    | Today at 10:24 AM<br>Ran for 2 minutes     |
| 32 | elmer          | Customer_C<br>Getting Star    | Publish<br>Failed Today at 10:25 AM<br>1 Failed                | Completed | Today at 10:24 AM<br>Ran for a few seconds |

**Figure: Publish job failed**

In the above example, the Transform and Profile jobs completed, but the Publish job failed. In this case, the results exist and, if the source of the problem is diagnosed, they can be published separately. From the job's context menu, select **Download Logs**. You can download the jobs logs to look for reasons for the failure. See *Review Logs* below.

## Invalid file paths

When your job uses files as inputs or outputs, you may receive invalid file path errors. Depending on the backend datastore, these can be one of the following:

- Path to the file is invalid for the current user. Path may have been created by another user that had access to the location.
- Path contains invalid characters in it. For more information, see *Supported File Formats*.
- Resource was deleted.

## Jobs that Hang

In some cases, a job may stay in a pending state indefinitely. Typically, these errors are related to a failure of the job tracking service. You can try the following:

- Resubmit the job.
- Have an administrator restart the platform. See *Start and Stop the Platform*.
- Submit the job again.

## Spark Job Error Messages

The following error messages may appear in the Designer Cloud application when a Spark job fails to execute.

### "Aggregate too many columns" error

Your job could not be completed due to one or more Pivot, Window or other Aggregation recipe steps having too many aggregate functions in the `Values` parameter.

**Solution:** Please split these aggregates across multiple Aggregation steps.

### "Binary sort" error

Sorting a nested column such as an array or map is not supported.

### Codegen error

Your job could not be completed due to the complexity of your recipe.

### Tips:

- Look to break up your recipe into sequences of recipes. You can chain recipes together one after another in Flow View.
- If you have complex, multi-dataset operations, you should try to isolate these into smaller recipes.
- Use sampling to checkpoint execution after complex steps.

### "Colon in path" error

Your job references one or more invalid file paths. File and folder names cannot contain the colon character.

### "Invalid input path" error

Your job references one or more invalid file paths. File names cannot begin with characters like dot or underscore.

### "Invalid union" error

Union operations can only be performed on tables with compatible column types.

**Tip:** Edit the union in question. Verify that the columns are properly aligned and have consistent data types. For more information, see *Union Page*.

### "Job service unreachable" error

There was an error communicating with the Spark Job Service.

**Tip:** An administrator can review the contents of the `spark-job-service.log` file for details. See *System Services and Logs*.

### "Oom" error

When you encounter out of memory errors related to job execution, you should review the following general items related to your flow.

#### General Tips:

- Review your recipes to see if you can identify ways to break them up into smaller recipes.
- Operations such as joins and unions can greatly increase the size of your datasets.
- Resource consumption is also affected by the the complexity of your recipe(s).
- If you suspect that there are several jobs running in parallel, you can drop the job launch batch size to 2 or 1 , which serializes job execution while preserving memory. For more information, see *Configure Application Limits*.
- You might be able to configure overrides to the Spark settings to allocate more memory for job execution.
  - This feature may need to be enabled in your environment. See *Enable Spark Job Overrides*.
  - See *Configure User-Specific Props for Cluster Jobs*.

### "Path not found during execution" error

One or more datasources referenced by your job no longer exist.

**Tip:** Review your flow and all of its upstream dependencies to locate the broken datasource. Reference errors for upstream dependencies may be visible in downstream recipes.

### "Too many columns" error

Your job could not be completed due to one or more datasets containing a large number of columns.

**Tip:** A general rule of thumb is to avoid over 1000 columns in your dataset. Depending on your environment, you may experience performance issues and job failures on narrower datasets.

### "Version mismatch" error

The version of Spark installed on your Hadoop cluster does not match the version of Spark that Designer Cloud powered by Trifacta Enterprise Edition is configured to use.

**Tip:** For more information on the appropriate version to configure for the product, see *Configure for Spark*.

### Databricks Job Errors

The following error messages are specific to Spark errors encountered when running jobs on Databricks.

**NOTE:** When a Databricks job fails, the failure is immediately reported in the Designer Cloud application . Collection of the job log files from Databricks occurs afterward in the background.

**Tip:** A platform administrator may be able to download additional logs for help in diagnosing job errors.

### "Runtime cluster" error

There was an error running your job.

### "Staging cluster" error

There was an error launching your job.

### Try Other Job Options

You can try to re-execute the job using different options.

#### Tips:

- **Look to cut data volume.** Some job failures occur due to high data volumes. For jobs that execute across a large dataset, you can re-examine your data to remove unneeded rows and columns of data. Use the Deduplicate transformation to remove duplicate rows. See *Remove Data*.
- **Gather a new sample.** In some cases, jobs can fail when run at scale because the sample displayed in the Transformer page did not include problematic data. If you have modified the number of rows or columns in your dataset, you can generate a new sample, which might illuminate the problematic data. However, gathering a new sample may fail as well, which can indicate a broader problem. See *Samples Panel*.
- **Change the running environment.** If the job failed on Trifacta Photon, try executing it on another running environment.

**Tip:** The Trifacta Photon running environment is not suitable for jobs on large datasets. You should accept the running environment recommended in the Run Job page.

## Review Logs

### Job logs

In the listing for the job on the Jobs page, click **Download Logs** to send the job-related logs to your local desktop.

**NOTE:** If encryption has been enabled for log downloads, you must be an administrator to see a clear-text version of the jobs listed below. For more information, see *Configure Support Bundling*.

When you unzip the ZIP file, you should see a numbered folder with the internal identifier for your job on it. If you executed a transform job and a profile job, the ZIP contains two numbered folders with the lower number representing the transform job.

`job.log`. Review this log file for information on how the job was handled by the application.

**Tip:** Search this log file for `error`.

**Support bundle:** If support bundling has been enabled in your environment, the `support-bundle` folder contains a set of configuration and log files that can be useful for debugging job failures.

**Tip:** Please include this bundle with any request for assistance to *Alteryx Support*.

For more information on configuring the support bundle, see *Configure Support Bundling*.

For more information on the bundle contents, see *Support Bundle Contents*.

### Support logs

For support use, the most meaningful logs and configuration files can be downloaded from the application. Select **Help menu > Download logs**.

**NOTE:** If you are submitting an issue to *Alteryx Support*, please download these files through the application.

For more information, see *Download Logs Dialog*.

The admin version of this dialog enables downloading logs by timeframe, job ID, or session ID. For more information, see *Admin Download Logs Dialog*.

### Trifacta node logs

**NOTE:** You must be an administrator to access these logs. These logs are included when an administrator downloads logs for a failed job. See above.

On the Trifacta node, these logs are located in the following directory:

```
<install_dir>/logs
```

This directory contains the following logs:

- `batch-job-runner.log`. This log contains vital information about the state of any launched jobs.
- `webapp.log`. This log monitors interactions with the web application.

Issues related to jobs running locally on the Trifacta Photon running environment can appear here.

## Hadoop logs

In addition to these logs, you can also use the Hadoop job logs to troubleshoot job failures.

- You can find the Hadoop job logs at port 50070 or 50030 on the node where the ResourceManager is installed.
- The Hadoop job logs contain important information about any Hadoop-specific errors that may have occurred at a lower level than the Designer Cloud application , such as JDK issues or container launch failures.

## Contact Support

If you are unable to diagnose your job failure, please contact *Alteryx Support*.

**NOTE:** When you contact support about a job failure, please be sure to download and include the entire zip file, your recipe, and (if possible) your dataset.

# Schedule a Job

Contents:

- *Add a Schedule*
- *Schedule a Destination*
- *Edit Schedule*
- *Disable Schedule*
- *Delete*
  - *Delete a schedule*
  - *Delete a destination*

After you have finished developing the recipes in your flow, you can define scheduled executions of the recipe or recipes within the flow to deliver outputs to known locations. Using the Automator, you can automate execution of jobs on source data, which can be replenished with fresh data asynchronously.

**NOTE:** Before you begin, you should verify that your data management pipeline into and out of the platform has been appropriately defined. This pipeline includes how data is written to the output location. For more information, see *Overview of Automator*.

When you schedule a job, you create two objects in Flow View:

**NOTE:** You must create both of these objects to schedule a job execution.

| Object                | Description                                                                                                                                                                                    |
|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Schedule              | A schedule applies to the entire flow. It contains one or more intervals at which the recipes of the flow are executed. Recipes are executed if their outputs include a scheduled destination. |
| Scheduled destination | A scheduled destination is an output location, format, and other settings that is populated with the results of executing the related recipes.                                                 |

**Tip:** You can create schedules for datasets with parameters. Any overrides specified through Flow View are automatically applied at runtime for the scheduled job. See *Flow View Page*.

## Add a Schedule

Steps:

1. Open the flow in Flow View.
2. From the context menu, select **Schedule**.
3. In the Add Schedule dialog, select your scheduling options:
  - a. Timezone: Select the timezone to use to determine when to execute the specified schedule.

- b. Frequency: Select the time and frequency of execution: Hourly, Daily, Weekly, Monthly, or cron.

**NOTE:** Scheduling supports a modified version of cron scheduling syntax. For more information, see *cron Schedule Syntax Reference*.

- c. To add another scheduled time, click **Add**.
4. To save your schedule, click **Save**.
5. A Calendar icon appears in Flow View to indicate that the flow has a schedule associated with it.

For more information, see *Add Schedule Dialog*.

## Schedule a Destination

### Steps:

1. To specify a destination for your schedule, click the recipe you wish to execute at the scheduled time.
2. If you have not done so already, click the Output icon next to the recipe to create an output for it.
3. In the right panel, locate the Scheduled destinations header. Click **Add**.
4. Specify an output location, format, and updating method.
5. Click **Save**.

For more information, see *Run Job Page*.

## Edit Schedule

- To edit the scheduled times, click the Calendar icon. Then, click **Edit**. Make changes as needed and save.
- To edit a scheduled destination, select the output in Flow View. In the right panel, click **Edit** next to the appropriate scheduled destination.

## Disable Schedule

- To disable a schedule you control, click the Calendar icon in Flow View. Then, move the slider to disable it.
- Administrators can disable schedules for all flows in the workspace. For more information, see *Schedules Page*.

## Delete

### Delete a schedule

In Flow View, click the Calendar icon. Then, click **Delete**.

### Delete a destination

**Tip:** If you have deleted the schedule for the flow, you do not need to delete the scheduled destination. It cannot run without a schedule.

1. In Flow View, select the output.
2. In the right panel, select **Delete Output** from the context menu.



# Create Branching Outputs

From a single collection of datasets, you may need to generate multiple outputs for downstream purposes. Examples:

- You want to preserve the ability to review and profile your source data. For more information, see *Profile Your Source Data*.
- You need different pivot tables produced from the wrangled data.
- You need to filter down the set of rows or columns to deliver to one user community while delivering a different set of columns to another.

## Reshaping Transformations

If your next step is to add any of the following transformations and you wish to preserve the existing data for other uses, you should consider adding these steps in a separate dedicated recipe.

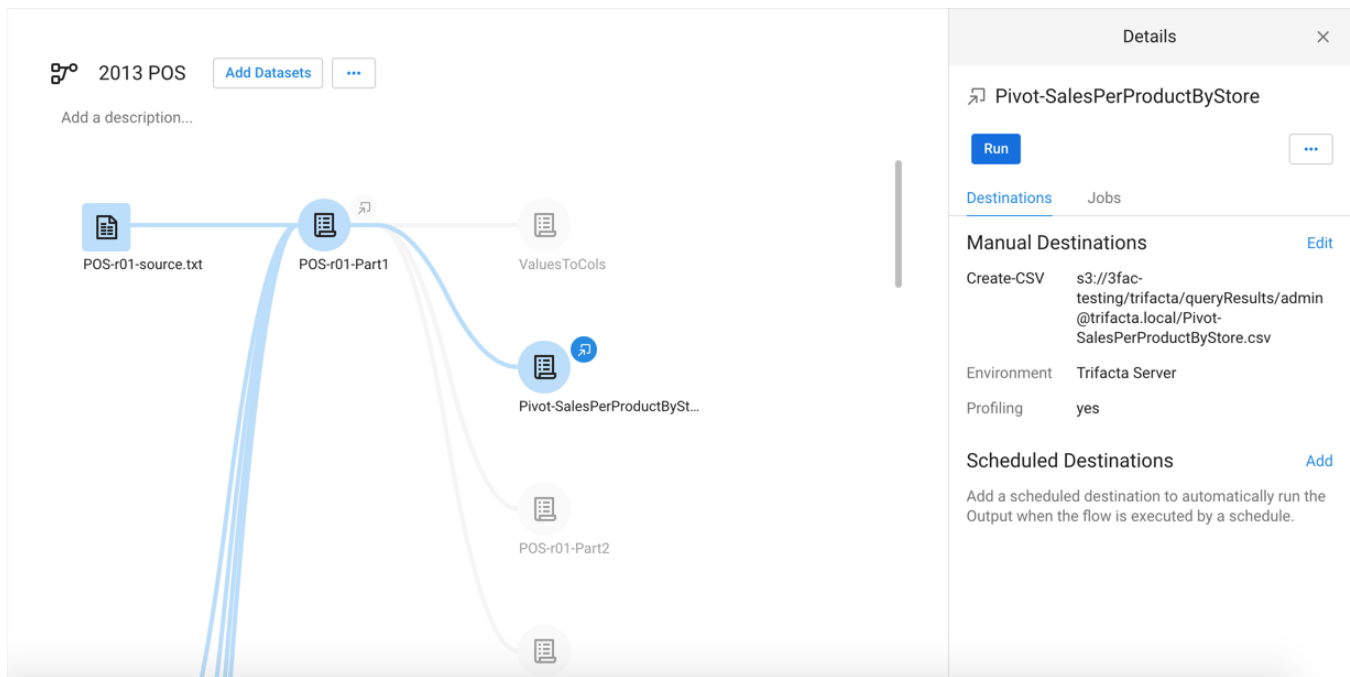
| Transformation Name   | Description                                                                                                                                                                                                                                                                                                                                                                                                            |
|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Union                 | A union appends one or more datasets to your current one. To preserve the original, you may need to create a branching output. See <i>Union Page</i> .                                                                                                                                                                                                                                                                 |
| Join                  | A join combines two datasets based on common values in specified columns in both datasets. These types of transformations can greatly change the shape of your data. See <i>Join Window</i> .<br><br>Similarly, a lookup uses values from a column in your source data to pull in corresponding rows of data from a reference dataset. These transformations add columns to your dataset. See <i>Add Lookup Data</i> . |
| Remove duplicate rows | This transformation removes identical rows from your dataset. However, there may be a set of steps required to standardize values in various columns before applying the de-duplication. You may choose to manage this process in a branching recipe.                                                                                                                                                                  |
| Delete columns        | When a column is removed, it is no longer available for use in any downstream output. See <i>Remove Data</i> .                                                                                                                                                                                                                                                                                                         |
| Filter                | Rows can be filtered from your dataset to render different perspectives. These changes may be best moved to a secondary, branching recipe. See <i>Filter Data</i> .                                                                                                                                                                                                                                                    |
| Pivot data            | When you create a pivot table, all source data that is not explicitly specified in the pivot is dropped from the dataset. For more information, see <i>Pivot Data</i> .                                                                                                                                                                                                                                                |
| Group by              | You can perform aggregation calculations within a table, which may force column data to be dropped. See <i>Create Aggregations</i> .                                                                                                                                                                                                                                                                                   |

## Basic Technique

Whenever you are applying a transformation that destroys data or otherwise reshapes your dataset and you wish to preserve the current state of the dataset, you should do the following:

1. In Flow View, select your current recipe. Click **Add new recipe**.
2. This recipe becomes the source for a branched output. Give the new recipe an appropriate name. For example, `Pivot-SalesPerProductPerStore`.
3. For this recipe, click the Output icon. Specify the appropriate output format and location that you'd like to generate for this branched output.
4. Select your current recipe again. Click **Add new recipe**.
5. This recipe becomes the extension of your current recipe. Give the new recipe an appropriate name. For example, `MyRecipe-Part2`.
6. Select the `Pivot-SalesPerProductPerStore` recipe. Click **Edit recipe**.
7. Build your pivot transformation in this recipe.
8. When ready, run the job. The output should be generated in the appropriate format and location.

**Tip:** When you run a job, all upstream dependencies are generated as part of the job. However, if you have multiple branches in your flow, you must run multiple outputs to generate all of the results. Generating these results may be easier if you create scheduled destinations and then add a schedule to trigger them. For more information, see *Overview of Automator*.



**Figure:** Multiple pivot tables sourced from output of a primary recipe for the flow. POS-r01-Part2 can be used for continued wrangling of primary recipe.

# Build Sequence of Datasets

In some situations, you may need to create a sequence of datasets, in which the output of one recipe becomes the input of another recipe.

Potential uses:

1. You may want to handle data cleanup tasks in one set, before that data is made available to other users for customization for their needs.
2. Columns or rows of data may need to be dropped before the dataset is made available to other users.
3. You may want to have different individuals working on each phase of the data transformation process. For example, one individual may be responsible for cleansing the data, while another may be responsible for transforming the data into final format.

Depending on your situation, you can apply one of the following solutions.

## Chain Recipes in Same Flow

Within a flow, you can chain together recipes. For example, you may wish to use the first recipe for cleansing and then second recipe for transforming. This method is useful if you are using a single imported dataset for multiple types of transformations within the same flow.

**Steps:**

1. Click the imported dataset. Click **Add new recipe**.
2. Click the new recipe. Name it, `Cleanse`.
3. With the new recipe selected, click **Add new recipe**.
4. Click the new recipe. Name it, `Transform`.

The output of `Cleanse` recipe becomes the input of `Transform` recipe.



**Figure: Chained recipes**

## Create Reference Objects

If you need to make the output of a recipe available in other flows, you can create a reference object. This reference is available in other flows that you control.

**Steps:**

1. In Flow View, select the recipe whose output you wish to make available to other flows.

2. Click the Create Reference icon:



Details

×

Transform

Add to Flow...

...

Data Preview

| # | Store_Nbr | Item_Nbr | WM_Week |
|---|-----------|----------|---------|
| 1 |           | 381000   | 201050  |
| 2 |           | 325000   | 201049  |
| 2 |           | 325000   | 201049  |
| 2 |           | 403000   | 201049  |
| 2 |           | 449000   | 201049  |
| 2 |           | 490000   | 201049  |
| 2 |           | 560000   | 201049  |

Updated

Today at 9:56 AM

Created

Today at 9:56 AM

Used in

0 Flows [More details](#)

**Figure: Create reference object**

- To use it in one of your other flows, click **Add to Flow....**
- In the target flow, the reference object appears as a **reference dataset**. It works like an imported dataset with the following considerations.

### Key Considerations:

- When you run a job in a flow that contains a reference dataset, all upstream dependencies of that reference dataset are executed. For the source reference object, all imported datasets and recipes are gathered and executed to populate the reference dataset with fresh data.
- The above has the following implications:
  - If the user running the job in flow #2 does not have permissions to access all of the upstream dependencies of the reference dataset, the job may fail. These dependencies include imported datasets and any connections.
  - If the upstream objects are owned by other users, you may not be able to review these items. For example, if the source recipe is changed by another user, your downstream recipe may break without notice. If you cannot review that recipe, then you can see what was changed and how to fix it.

### Create Imported Dataset from Output

If any of the above considerations are a concern, you can create an imported dataset from the job results of flow #1.

In the Job Details page, click the Output Destinations tab. For the generated output, select **Create imported dataset** from its context menu.

**NOTE:** When the new dataset is created, it is accessible only to the creator. Datasets can be shared with other collaborators. For more information, see *Overview of Sharing*.

From the results of wrangling your first dataset, you can create a new dataset. This dataset is wrangled in a separate recipe, the output of which can become a third dataset. In this manner, you can create sequences of datasets.

### Key Considerations:

- The imported dataset in flow #2 is not refreshed until you run the job that generates it in flow #1.
- If the output of flow #1 uses the same filename each time, you may not know if the data has been refreshed. When the job is executed in flow #2, it collects the source imported dataset and executes, whether the data is new or not. Workarounds:
  - In flow 2, you can create a parameterized dataset, which collects source data, with some variation in parameters. As long as the output of flow #1 follows the naming convention for the parameterized dataset for flow #2, you should be able to run the job on fresh data on-demand. For more information, see *Overview of Parameterization*.
  - After the job in flow #2 executes, rename or remove the output of flow #1 from its target location. That way, whenever job #2 executes again, any data that it collects from the source location is likely to be newer.

See *Job Details Page*.

# Fix Dependency Issues

## Contents:

- *How to Identify*
  - *Dependent datasets*
  - *Broken data integrations*
  - *Hidden breakages*
- *Fixing Dependencies*

Where possible, changes made in one dataset or recipe propagate to the datasets that consume it. Datasets that join, union, or lookup against your dataset are likely to be impacted if you delete columns or rows or otherwise change the data. In some cases, the recipes of these dependent datasets can break.

This section describes how to identify these dependency issues and includes general steps for fixing them.

## How to Identify

### Dependent datasets

When making edits to a recipe, you can verify if your changes potentially impact other recipes or reference datasets that rely on it. In the Transformer page, click the drop-down next to the current dataset's name to open the Recipe Navigator. Select the Flow View tab.

**Tip:** If your current dataset is connected to datasets to the right of it, those datasets are dependent on the current one. After you make changes to the current one, you should use the Recipe Navigator to open recipes and datasets that are connected to it and to the right of it in flow view.

See *Recipe Navigator*.

### Broken data integrations

When you make some changes in an upstream recipe or dataset, the recipes for any downstream datasets can break, such that you cannot generate satisfactory results. In the downstream recipe, you may see errors in the Recipe panel, such as the following:

The screenshot shows a recipe panel with a table of data and a list of recipe steps. The table has columns 'Day' and 'Whse\_Nbr'. The data rows are: 2013/02/07, 0; 2013/02/07, 0; 2013/02/07, 6094. The recipe steps are: 1. Union with POS-r02.txt, POS-r03.txt; 2. Join with REF\_PROD; 3. Delete ITEM\_NBR1; 4. Join with REF\_CAL; 5. Delete Day. An error message is displayed: 'Column Day does not exist in REF\_CAL. Go to REF\_CAL to fix the error'.

**Figure: Dependency error in the Recipe panel**

In the above, the column `Day` does not exist in the current dataset, which is causing problems in the last two recipe steps. These types of errors may be generated when a column in the upstream dataset has been dropped or renamed.

## Steps:

1. Open the object where the column was dropped:
  - a. If the recipe or dataset is from the same flow, you can use the Recipe Navigator in the Transformer Page. See *Recipe Navigator*.
  - b. If the recipe or dataset is in a different flow, use the Flows page to locate it (REF\_CAL.txt in the above). See *Flows Page*.
2. In the Flow View tab, open the dataset referenced in the error message.
3. In the Recipe panel, locate the step where the column was removed.
4. Fix the issue. Details are below.

## Hidden breakages

If you make changes to specific values in a dataset, recipe steps in downstream datasets can break if they rely on detecting specific values. Depending on the usage, the step may not actually be broken, but the generated results are incorrect.

For example, a downstream dataset recipe includes the following step:

|                             |                                   |
|-----------------------------|-----------------------------------|
| <b>Transformation Name</b>  | Filter rows when value is exactly |
| <b>Parameter: Condition</b> | Is exactly                        |
| <b>Parameter: Column</b>    | company_name                      |
| <b>Parameter: Value</b>     | 'My Co.'                          |
| <b>Parameter: Action</b>    | Delete matching rows              |

If the `company_name` column is sourced from another dataset and the `My Co.` value is changed to `My Company`, the downstream dataset that includes this transform doesn't break in an easily noticeable way. The data is simply not removed from the dataset and any generated results.

## Fixing Dependencies

When you locate a dependency issue in the upstream dataset, you can fix it using one of the following methods:

1. Fix the issue in the source dataset. Verify that the change does not impact other datasets.

**NOTE:** If you fix the issue in the source dataset, you should verify if any other downstream datasets are impacted by this change.

2. Change the input dataset to use a dataset that is not broken.

**Tip:** If you must freeze the data in the dataset that you are using as an input, you can create a copy of the dataset as a snapshot. See *Dataset Details Page*.

To use the copy, repair or rebuild the integration using the copied version.

3. Fix the issue in the dataset that depends on it. In this case, you must redefine the transformation that brings in the data.

# Share a Flow

You can allow other users to work on flows that you own. Flow **collaborators** have almost all of the same permissions as flow **owners**.

**NOTE:** Users of a shared flow must have read access to the underlying data sources to access the datasets of a shared flow. If they do not have dataset access, collaborators can still access the flow but have more limited capabilities.

**NOTE:** When you share a flow that contains a dataset sourced from Microsoft Excel, the user with whom the flow is shared may receive a `Could not parse` error. In this case, the user does not have access to the original sample. The workaround is to take a new sample or to run a job on the full dataset.

## Steps:

1. In the Flows page, locate the flow to share.
2. From the context menu on the right side of the screen, select **Share**.
3. In the Share Flow dialog, enter the name of the user or users with whom you would like to share the flow.
  - a. You can specify the privilege level of the user to whom you are sharing.
  - b. For more information, see *Share Flow Dialog*.
4. Click **Add**.
5. The selected users can now see the flow and interact with its objects in the Shared with Me tab of the Flows page. See *Flows Page*.

For more information on the privileges of collaborators, see *Overview of Sharing*.



# Export Flow

As needed, you can export a flow from Designer Cloud powered by Trifacta® Enterprise Edition. An exported flow is stored in a ZIP file that contains all objects needed to use the flow in any instance of that platform that can access the flow's sources.

Exported flows can be imported into the same system or different systems. Flow export is useful for:

- Backups of work in progress

**You cannot import flows that were exported from a version before Release 6.8.** See *Changes to the Object Model*.

- Archiving of completed development work
- Migrating flows from one instance to another
- Deployment of work to Production environments

An exported flow ZIP also includes:

- Any `.data` files, which may be included as artifacts of feature usage.
  - For transformation by example, artifact files include the value transformation information for the TBE step. For more information, see *Overview of TBE*.
  - For cluster clean, artifact files contain the mappings between source values and clustered values. For more information, see *Overview of Cluster Clean*.
- Any configured webhook tasks are part of the flow definition. For more information, see *Create Flow Webhook Task*.

## Export from Flows Page

### Steps:

1. From the menu, select **Flows**.
2. In Flows page, locate the flow to export. From the context menu, select **Export**.
3. To export, click **Download**.
4. The ZIP file is downloaded to the default download location on your local desktop.

**Tip:** You can also export from Flow View. See *Flow View Page*.

**NOTE:** When you import a flow, you import this ZIP file. You cannot import the contents of the ZIP. If your local environment automatically unzips ZIP files, please re-ZIP before you import. For more information, see *Import Flow*.

## Export from Production instance

**Tip:** In general, avoid making changes in a Production environment. Instead, you should make changes in a Development environment, export from there, and reimport into the Production environment.

### Steps:

1. Login to the Production instance. The Deployment Manager is displayed.
2. From the menu, select **Deployments**.
3. Select the deployment that you wish to export.
4. In the list of releases, locate the release to export. From the context menu, select **Export**.
5. Add any optional notes for the export. When the flow is imported into another environment, this notes are displayed in the user interface.
6. To export, click **Download**.
7. The ZIP file is downloaded to the default download location on your local desktop.

This file can be stored for safekeeping or imported back into the instance. For more information, see *Import Flow*.

# Import Flow

## Contents:

- *Limitations*
  - *Define import rules*
  - *Import*
  - *Import into Prod instance*
- 

An exported flow can be imported into Designer Cloud powered by Trifacta® Enterprise Edition.

- **Dev instance:** If you are using an instance of the platform for developing and testing your flows, you can import a new flow through the Flows page.

**NOTE:** Unless your instance of the platform has been specifically configured to support deployment management, you are using a Dev instance of the platform.

**NOTE:** If you are attempting to share a flow with other users on the same instance of the platform, you should use the sharing functions. See *Overview of Sharing*.

- **Prod instance:** If you are importing a flow into a Production instance of the platform, you import it as a package through the Deployment Manager.

**NOTE:** Deployment Manager is a feature that enables segmentation of platform usage between Dev instances and Prod instances. This feature must be enabled and configured. For more information, see *Overview of Deployment Manager*.

## Limitations

**You cannot import flows that were exported before Release 6.8. See *Changes to the Object Model*.**

**NOTE:** You cannot import flows into a version of the product that is earlier than the one from which you exported it. For example, if you develop a flow on free Designer Cloud powered by Trifacta Educational, which is updated frequently, you may not be able to import it into other editions of the product, which are updated less frequently.

Imported flows do not contain the following objects:

**NOTE:** Depending on the import environment, some objects in the flow definition may be incompatible. For example, the connection type may not be valid, or a datasource may not be reachable. In these cases, the objects may be removed from the flow, or you may have to fix a reference in the object definition. After import, you should review the objects in the flow to verify.

- Reference datasets
- Samples
- Connections

**NOTE:** Exported flows do not contain connections. If your flow relies on a connection to the source, you must create the connection in the Prod environment and create an import mapping rule to assign the local connection ID to the import package. Flows that do not require connections may not require remapping before import. See *Define Import Mapping Rules*.

**NOTE:** If the flow's output object uses connections that are not used for importing datasets in the flow, the output is broken on import. Those outputs and their associated connections must be recreated in the environment into which the flow is imported.

Imported datasets that are ingested into backend storage for Designer Cloud powered by Trifacta Enterprise Edition may be broken after the flow has been imported into another instance. These datasets must be reconnected to their source. You cannot use import mapping rules to reconnect these data sources. This issue applies to the following data sources:

- Microsoft Excel workbooks and worksheets. See *Import Excel Data*.
- PDF tables. See *Import PDF Data*.

## Define import rules

Before you import a package, you may need to create import mapping rules to apply to your package. For example, if the Development data is stored in a different location than the Production data, you may need to create import rules to remap paths and connections to use to acquire the data from the Production environment.

**NOTE:** Import rules are applied at the time of import. They cannot be retroactively applied to releases that have already been imported.

For more information, see *Define Import Mapping Rules*.

## Import

**NOTE:** You cannot import into a Dev instance if your account for the instance contains the Deployment role.

**NOTE:** If the exported ZIP file contains a single JSON file, you can just import the JSON file. If the export ZIP also contains other artifact files, you must import the whole flow definition as a ZIP file. For best results, import the entire ZIP file.

### Steps:

1. Export the flow from the source system. See *Export Flow*.
2. Login to the import system, if needed.

3. Click **Flows**.
4. From the context menu in the Flow page, select **Import Flow**.
5. Select the ZIP file containing the exported flow. Click **Open**.

If there are issues with the import, click the Download link to review the missing or malformed objects.

**Tip:** When you import the flow, click the Warnings link to review the list of objects that must be remapped.

The flow is imported and available for use in the Flows page. After import:

- You may need to reconnect your imported datasets to data sources that are available in the new workspace or project. See *Reconnect Flow to Source Data*.
- You may also need to reconnect your outputs. See *Reconnect Flow to Outputs*.

## Import into Prod instance

After creating any import rules in your Prod instance, please do the following.

### Steps:

1. Export the flow from the source system. See *Export Flow*.
2. Login to the Prod instance. The Deployment Manager is displayed.
3. Click **Deployments**.
4. Select or create the deployment into which to import the package.
5. Within the deployment, click **Import Package**.
6. Select the ZIP file containing the exported flow. Click **Open**.
  - a. Any defined import rules are applied to the package during import.
  - b. The package is selected as the active one for the deployment.
  - c. If there are issues with the import, click the Download link to review the missing or malformed objects.

**Tip:** After you import, you should open the flow in Flow View and run a job to verify that the import was successful and the rules were applied. See *Flow View Page*.

# Reconnect Flow to Source Data

When you import a flow into a new project or workspace, you may need to remap the imported datasets in the flow to source data.

**Tip:** When you import the flow, click the Warnings link to review the list of objects that must be remapped.

When the flow is imported from one instance to another instance, the imported datasets may be broken after the flow has been imported into another instance. You must map the imported flow to the corresponding data sources.

## Steps:

1. Open the imported flow.
2. In Flow View, the datasets that need remapping have a red dot in the corner of their icon. This dot means that the Designer Cloud application is unable to connect to the dataset.
3. You may need to recreate the connections that were used to import the dataset in the original workspace. See *Connections Page*.
4. For each broken dataset:
  - a. Right-click the dataset and click **Replace**.
  - b. From the Replace dialog, select the existing dataset or click **Import Datasets** and select the required dataset. For more information, see *Import Data Page*.
  - c. Select **Replace**.
  - d. The selected dataset is replaced with another dataset.
5. Repeat the above steps for each broken dataset in the flow.

You may also need to remap the outputs. For more information, see *Reconnect Flow to Outputs*.

# Reconnect Flow to Outputs

When you import a flow into a new project or workspace, you may be required to remap the flow outputs to accessible publishing destinations.

**Tip:** When you import the flow, click the Warnings link to review the list of objects that must be remapped.

**Tip:** You should remap the data sources first. See *Reconnect Flow to Source Data*.

## Steps:

1. Open the imported flow.
2. In Flow View, for each output:
  - a. Select the required output. The object details are displayed in the Details panel.
    - i. If you cannot connect to the data, you do not have permissions to use the connection specified in the flow or the connection may not be available in the current project or workspace. You must create a new connection to access the source data. See *Reconnect Flow to Source Data*.
  - b. In the Details panel, click **Edit** or **Add**. The Publishing Settings page is displayed.
  - c. Edit the changes as required.
  - d. To save your changes, click **Update**.
3. Repeat the above steps for the other outputs in the flow.
4. To verify, run a job that generates one of the outputs.

# Create or Replace Macro

## Contents:

- *Create Macro*
    - *Define macro inputs*
  - *Edit Macro*
  - *Convert Macro to Steps*
  - *Replace Macro*
    - *Replace macro with another macro*
    - *Replace macro with steps*
    - *Update macro inputs*
  - *Inspect Macro*
  - *Apply Macro*
  - *Manage Macros*
- 

You can create reusable macros from sequences of steps in your recipe. These macros can be applied in other locations of the recipe or in other recipes. If needed, you can modify the steps in an instance of the macro to replace the existing steps, allowing you to make changes and updates to your macros.

## Macro Definition:

**Macros** are user-defined sequences of recipe steps that can be referenced independently and parameterized as needed. A macro is composed of the following types of information:

- **Steps** are the recipe steps that are executed each time that the macro is invoked. A macro contains one or more steps.
- **Inputs** are variables that can be modified wherever the macro is placed. For example, you might have an input that contains the name of a column. This column name may change between recipes, so you can create a macro input to capture the column name, which is the **input value** for the macro input. A macro input can be referenced one or more times in your macro steps.

For more information, see *Overview of Macros*.

## Create Macro

### Steps:

1. In the Recipe panel, select the step or steps to include in your macro.

**NOTE:** Source steps from your recipe do not have to be consecutive. In the macro, steps are listed in the order in which they appear in the recipe.

2. From the recipe toolbar context menu, select **Create or replace macro**.

**NOTE:** The dialog name and options vary based on the selection of create or replace macros.

3. From the drop-down, select **Create a macro**. Enter a Name and an optional Description.

**NOTE:** The Name of the macro appears in the Designer Cloud application. Please verify that the Name is unique.



4. Click **Next**.
5. In the Create macro dialog, you can review the selected steps and the inputs for the macros:

**Figure: Create macro inputs**

6. For each step in the macro:
  - a. Left column: Select the step.
  - b. Middle column: For the selected step, review the values that were specified for the step in the original recipe.
  - c. Right column: As needed, you can provide values for the currently selected inputs from the middle column. For a selected value, you can choose to create a new input or use an existing input.
7. You can review the macro inputs separately in the Inputs tab. For more information, see "Define macro inputs" below.
8. When you have finished specifying your macro and its inputs, click **Create**.
9. The macro is created.
10. In the recipe location where you created it from, the steps from which you created your macro are replaced with an Apply transformation step that references your macro name.

## Define macro inputs

A **macro input** is a variable within the macro whose value can be set to a default or, if needed, modified in each instance of the macro.

When you are specifying a macro, the Designer Cloud application reviews the steps of the macro to identify the values that can be modified in it. In the middle column of Steps tab:

| Value type         | Description                                                                                                                                                                       |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Columns</b><br> | Column names are automatically turned into inputs.                                                                                                                                |
|                    | <p>These two values could be turned into macro inputs but are not currently defined as such.</p> <p>IFMISMATCHED is a function name, which cannot be parameterized as inputs.</p> |
|                    | These two values have been turned into macro inputs.                                                                                                                              |

```
IFMISMATCHED($col, [ABC strDataType ×],
intMismatchReplacementValue ×)
```

### Create macro inputs:

When you are defining a macro, you can create or modify macro inputs.

**Tip:** Macro inputs can be created or modified in the Steps or Inputs tabs.

**Tip:** Column names are always recognized as inputs. They can be modified as needed in each instance of the macro.

### Steps:

1. To create a new macro input, select a value that is not highlighted.
2. In the right column, enter a name for this new macro input.
3. Specify its default value, and click **Create**.
4. The macro input is created. In the middle column, the highlighted value has been replaced by the name of the macro input.

To modify a macro input, click the entry in the middle column. Then, specify values as needed in the right column, and click **Save**.

To delete a macro input, select it in the middle column. In the right column, click **Remove**.

**NOTE:** You cannot delete column names as macro inputs.

### Edit Macro

When you edit a macro, you can modify the name, description of the macro, as well as the names for any of its inputs.

**Tip:** To modify the steps of a macro, you must replace it. See "Replace Macro" below.

### Steps:

1. You can use either of the following methods to edit the macro:
  - a. In the Macros page, click **Edit** from the context menu of the macro.
  - b. From the recipe toolbar context menu, select **Edit macro**.
2. In the Edit macro dialog, modify the name and description as needed.
3. Click **Next**.
4. In the Edit Macro dialog, click the Inputs tab.
5. Review the listed inputs:
  - a. To change the name of any input, select it.
  - b. In the right panel, enter a new name and description value for the input. Click **Save**.
6. Repeat the previous step for other macro inputs as needed.
7. To save your modifications to the macro definition, click **Save**.

## Convert Macro to Steps

After you have created a macro, you may need to convert an instance of a macro to plain steps in your recipe for any of the following reasons:

- The macro definition is going to be changed, and you do not want this instance of the original macro steps to be affected by that change.
- The macro definition is going to be changed, and you want to use this instance as the basis for the new definition. See "Replace Macro" below.

To convert a macro to steps, select the macro instance in your recipe. Then, select **Convert macro to steps** in the context menu of the recipe toolbar.

**NOTE:** This operation converts the selected instance of the macro to a set of steps. It does not modify the definition of the macro. If preferred, you can delete the macro, which forces all instances of the macro in the workspace to be automatically converted to steps. For more information, see *Macros Page*.

## Replace Macro

To modify the steps in your macro, you must perform a replacement of all steps in the current definition.

### Replace macro with another macro

You can replace a macro's steps with all of the steps of a macro that you have exported to your desktop.

**Tip:** This method is useful for publishing changes to a macro from one workspace to other workspaces.

#### Steps:

1. Export a macro definition to your desktop.
2. In the Macros page, find the macro whose steps you'd like to replace with a macro that you've exported to your desktop. From its context menu, select **Replace**.
3. You may need to remap macro inputs in the imported steps to the existing references. See "Update macro inputs" below.

For more information, see *Macros Page*.

### Replace macro with steps

The following method can be used to replace a macro definition with steps that you have created in a recipe.

**Tip:** When replacing a macro, you can create new inputs for new steps and reassign inputs from the previous version to the steps that haven't changed.

Please complete the following steps.

#### Steps:

1. To replace all steps in the macro with new ones:
  - a. Create the steps in a recipe that you wish to use.
  - b. When you are ready to use them to replace a macro, select **Create or replace macro** from the context menu.
2. To modify the steps currently in the macro:
  - a. Open a recipe containing an instance of the macro.
  - b. Select the step that applies the macro. From the context menu, select **Convert macro to steps**.

- c. All of the macro steps are now listed as individual steps in your recipe.
- d. Add, remove, or modify steps to define your new macro.

**Tip:** You may want to remove the comment steps that mark the beginning and ending of the converted macro.

- e. When you are ready to use them to replace the macro, select all of the steps. From the context menu, select **Create or replace macro**.
3. In the Create macro dialog, select **Replace an existing macro** from the drop-down.
  4. From the Replace macro dialog, select the existing macro to replace.
  5. If you want to save the copy of the existing macro, select the corresponding checkbox.

**NOTE:** Replacing an existing macro replaces all the macro steps with the steps of the new macro. All instances of the previous definition of the macro now reference the new macro definition. In some cases, you may need to reassign input values on old instances to align with the inputs in the updated macro definition.

6. Define macro inputs:

- a. If the old version of the macro contained inputs, you should review those inputs and reassign them to values in the new macro definition.

**NOTE:** If you do not reassign the macro inputs from the old definition to the new one, then the values used for those inputs in macro instances created under the old definition are lost. After the replacement version is saved, you must review each instance of the macro to verify that it is working properly.

- b. You can also create new macro inputs that apply to the added or modified steps.
- c. See "Update macro inputs" below.

7. After you have reviewed the input, to replace the macro with the existing inputs, click **Replace**.

- a. If you do not specify a relationship between the existing inputs and the replacement macro's inputs, a warning message is displayed.

**NOTE:** If you discard and save the changes, then any references to those inputs in the instances of the macro in the previous definition are broken.

- b. Click **Discard** to save the macro.

## Update macro inputs

When you are replacing a macro, the macro inputs from the old version are carried over into the new version that you are defining.

**NOTE:** To preserve the values that are stored in the macro inputs from the old version, you must reassign the old macro input to its corresponding input in the new version. If this reassignment is not completed, the input values specified in the old version are lost, and each existing instance of the macro must be reviewed and updated with new macro input values.

**Figure: Reuse existing macro inputs**

1. For each step in the new macro definition,
  - a. Review the inputs in the middle column.
  - b. If a listed input has a corresponding macro input in the old version, select the input. In the right column, select **Use existing input** from the drop down. Then, select the existing input to reassign to the new one. Click **Save**. The input values from the old macro input are preserved.
  - c. If needed, you can create new macro inputs from values in the middle column. See "Define macro inputs" above.
2. Repeat the above steps for each input.

## Inspect Macro

When you inspect a macro definition, you review the steps that comprise the macro.

1. You can use either of the following methods to inspect the macro:
  - a. In the Macros page, click **Inspect** from the context menu of the macro.
  - b. From the recipe toolbar context menu, select **Inspect macro**.
2. The steps of the macro are displayed in raw Wrangle .

**Tip:** You can see the raw Wrangle for your macros in the Library. For more information, see *Macros Page*.

## Apply Macro

You can use macros that you have created in other recipe locations. See *Apply a Macro* .

## Manage Macros

You can manage macros through the Library page. See *Macros Page* .



# Apply a Macro

## Contents:

- *Insert in Recipe*
  - *Modify a Macro Instance*
    - *Replace the macro definition*
- 

After you have created a macro, you can apply it into any of your recipes.

- **Macros** are user-defined sequences of recipe steps that can be referenced independently and parameterized as needed.
- For more information on creating macros, see *Create or Replace Macro*.
- For more information, see *Overview of Macros*.

## Insert in Recipe

### Steps:

1. Through Flow View, edit the recipe into which you are inserting the macro.
2. In the Recipe panel, click the recipe cursor to the location where you are inserting it. See *Recipe Panel*.
3. In the Transformer toolbar, click the Macros icon. See *Transformer Toolbar*.

**Tip:** In the Search panel in the Transform Builder, you can search for `Macro` and then select the macro to use.

4. Search for and select the macro to insert. The macro is displayed in the Transform Builder.
5. Specify any macro input values required for the macro.

**NOTE:** Macro input values must be literal values. Use of flow parameters or metadata references is not supported.

6. To add the macro to the recipe, click **Add**.
7. The macro is added as an **Apply** step.

## Modify a Macro Instance

After a macro has been added to your recipe, the following options are available in the **Apply** step's context menu:

- **Inspect macro:** Click to see the definition of the macro. Definition is displayed in Wrangle .
- **Convert macro to steps:** Convert the instance of the macro to a set of static steps.

**NOTE:** This option converts the instance of the macro. The macro still exists.

## Replace the macro definition

If you want to modify the steps in your macro, please do the following:

1. Convert the macro to steps.
2. Perform your modifications to the steps. You can add or remove steps, too.

3. Select all of the steps that are to be used in the new version of the macro.
4. From the context menu, select **Create or replace macro**.

For more information, see *Create or Replace Macro*.



# Export Macro

As needed, you can export a macro from Designer Cloud powered by Trifacta® Enterprise Edition. An exported macro is stored in a JSON file that contains all of the information required to use the macro in any instance of the product.

**NOTE:** Only the creator of a macro can export it.

Exported macros can be imported into the same system or different systems. Macro export is useful for:

- Backups of work in progress

**You cannot import macros into an earlier release of the product.**

- Archiving of completed development work
- Migrating macros from one instance to another

## Export

### Steps:

1. From the left navigation bar, select **Library**.
2. In the Library page, click **Macros**.
3. In the Macro page, locate the macro that you wish to export. In its context menu, select **Export**.
4. The JSON file is downloaded to the default download location on your local desktop.

When you import a macro, you import this JSON file. For more information, see *Import Macro*.

## Export via API

You can export macro definitions using the APIs.

**Tip:** This method is useful for publishing macro definitions across all deployments in your organization.

For more information, see <https://api.trifacta.com/ee/es.t/index.html#operation/getMacroPackage>

# Import Macro

## Contents:

- *Limitations*
  - *Import*
  - *Import via API*
- 

A macro that has been exported from the Designer Cloud powered by Trifacta® Enterprise Edition can be imported back into the product.

- A **macro** is a reusable set of steps that are specified from within a recipe. For more information, see *Overview of Macros*.
- For more information on creating a macro, see *Create or Replace Macro*.

## Limitations

**You cannot import macros that were exported from a later release of the product.**

- You cannot modify the macro definition JSON file outside of Designer Cloud powered by Trifacta Enterprise Edition.
- If you are importing a flow into a Production instance of the platform, any macros referenced in the imported flow are expanded into their original steps in the recipes where they are referenced.
  - Macros cannot be imported, referenced, or viewed directly in a Production instance.
  - A Production instance is available only if you have enabled the Deployment Manager. For more information, see *Overview of Deployment Manager*.

## Import

**Tip:** If you re-import a macro into the same instance that still contains the source macro, the imported version is named the same as the source and automatically versioned for you.

## Steps:

1. Export the macro from the source system. See *Export Macro*.
2. Login to the import system, if needed.
3. In the left nav bar, click **Library**.
4. In the Library page, click **Macros**.
5. In the Macros page, click **Import Macro**.
6. Select the JSON file containing the exported macro.

**Tip:** You can import multiple macros at the same time. Select each JSON file in the dialog box that you wish to import.

7. Click **Open**.

The macro is imported and available in the Macros page.

To use an imported macro, enter `macro` in the Search panel in the Transform Builder. Select your macro and modify any macro inputs. For more information, see *Apply a Macro*.

## Import via API

If you have exported your macro using the APIs, you can import it into a new environment. For more information, see

<https://api.trifacta.com/ee/es.t/index.html#operation/importMacroPackage>

# Create Flow Parameter

## Contents:

- *Limitations*
    - *Limitations on usage*
  - *Create Parameter*
    - *Parameter Names*
    - *Apply Parameter Override*
    - *Override Evaluation*
  - *Use Parameter*
  - *Examples*
    - *Example - String parameter*
    - *Example - parameter with multiple values*
    - *Example - Date parameter*
  - *Apply Parameter Override via API*
- 

At the flow level, you can define flow parameters to reference in your recipes. A **flow parameter** is a variable that is assigned a String value.

**NOTE:** Flow parameters apply to recipe steps only.

- To flow parameters and parameters of other types, you can apply override values at the flow level through the same interface. Details are below.
- For more information on flow parameters, see *Overview of Parameterization*.

## Limitations

- Flow parameters are of String data type.

**Tip:** You can wrap flow parameter references in your transformations with one of the `PARSE` functions. See "Examples" below.

- Flow parameters are converted to constants in macros. Use of the macro in other recipes results in the constant value being applied.

## Limitations on usage

A flow parameter cannot be used in the following transformation steps or fields.

### Transformations:

- Rename columns: Cannot use a flow parameter as a new column name.

### Transformation fields:

- The `as` clause when creating a New formula transformation.

## Create Parameter

### Steps:

1. Open the flow where you wish to apply the flow parameter.
2. From the Flow View context menu, select **Parameters**.
3. In the Manage Parameters dialog, click the Parameters tab.
4. Click **Add parameter**.
5. Enter a Name for your parameter.

**NOTE:** Name values are case-sensitive. After saving a flow parameter, its name cannot be changed.

6. Enter a default value for this parameter.

**NOTE:** Input Values are evaluated as String type.

7. Click **Save**.

The parameter is available for use in any recipe in your flow. See "Use Parameter."

## Parameter Names

Parameter names can contain alphanumeric characters and spaces. In the following table, you can see how parameter names must be referenced in recipe steps.

| Parameter name | Valid references                         | Notes                                                                                                                                                                                                                                                                                                                                   |
|----------------|------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| paramRegion    | <pre>\$paramRegion \${paramRegion}</pre> | Both references are valid.                                                                                                                                                                                                                                                                                                              |
| param Region   | <pre>\${paramRegion}</pre>               | <p><b>NOTE:</b> If the parameter name contains a space, the curly brackets are required. As a matter of habit, you might want to use the curly brackets for all parameter references. This syntax also helps to distinguish your named parameters from metadata references, which are fixed. See <i>Source Metadata References</i>.</p> |

## Apply Parameter Override

**NOTE:** Parameter overrides that were defined in a pre-Release 7.1 version of the software now appear in the Overrides tab.

You can apply overrides to all parameter types, including flow parameters, at the flow level. An overridden value applies to all references of the parameter within the flow.

**NOTE:** You can apply override values for any parameter of any type that is referenced in the flow: dataset parameters, flow parameters, and object parameters.

- **Upstream parameter values:** Parameter values can be inherited from upstream recipes and datasets.

**NOTE:** Override values applied in a downstream flow are applied to the upstream flow when its objects are invoked for purposes of generating data for use in the downstream flow.

- **Downstream parameter values:** Downstream flows receive parameter values, default or overridden, from upstream flows. These values can be overridden at the flow level.

#### Steps:

1. Open the flow where you wish to apply the flow parameter.
2. From the Flow View context menu, select **Manage parameters.....**
3. In the Manage Parameters dialog, click the Overrides tab.
4. Click **Add override**.
5. Select the parameter to override from the drop-down list.
6. Set the override value for this flow. Click **Save**.
7. Click **Save**.

This override value is applied to all references to the parameter in the flow.

**Tip:** Overrides can also be applied to the recipe parameters that are included when flow tasks are executed as part of a plan. For more information, see *Manage Parameters Dialog*.

#### Override Evaluation

Override values can be applied in multiple locations. Parameter values are evaluated in the following order of precedence (highest to lowest):

1. Overrides at run-time in the Run Job page.
2. Overrides at the flow level.
3. Default values for the flow.
4. Inherited values from upstream flows.

For more information, see *Overview of Parameterization*.

#### Use Parameter

In your recipe step, you can add references to your flow parameter in the following format:

```
${MyRecipeParameter}
```

In a recipe, flow parameters can be applied to:

- Function parameters
- Replacements for String values

#### Examples

Below are examples of how to use flow parameters.

**NOTE:** When a parameter value is displayed in a column, the column type in the data grid may be correctly inferring the type to your desired data type. However, the underlying type is still String type. To convert the underlying type, you must use one of the `PARSE` functions on your String values.

#### Example - String parameter

In this example, data is segmented by time zone. You must create a parameter to capture the following U.S. time zones, which must be specified explicitly:

```
'Hawaii'
'Alaska'
'Pacific'
'Mountain'
'Central'
'Eastern'
```

In your flow, you create the following flow parameter:

| Setting | Value           | Notes                                                                                                                                                                                                                                                                                                                                                                                                                         |
|---------|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Name    | paramTimeZone   | <b>Tip:</b> It's a good habit to specify named variables in an identifiable way. By adding the <code>param</code> prefix, you identify references to it as a parameter. If you change the name to <code>param-recipeTimeZone</code> or similar to distinguish it as a flow parameter, then overrides specified at the flow level do not apply to any other parameter types that are performing the same function in the data. |
| Value   | ##UNSPECIFIED## | Since this value must be specified explicitly, you set this value as the default value. If this value appears in the generated output, then the flow parameter was not specified when the job was run.<br><br><b>NOTE:</b> Before you begin working with this parameter in your dataset, you should consider setting an override for it to a valid value.                                                                     |

In the following transformation, the parameter value is inserted into a new column, `paramTZ` in your dataset:

|                                   |                                |
|-----------------------------------|--------------------------------|
| <b>Transformation Name</b>        | New formula                    |
| <b>Parameter: Formula type</b>    | Single row formula             |
| <b>Parameter: Formula</b>         | <code>\${paramTimeZone}</code> |
| <b>Parameter: New column name</b> | <code>'paramTZ'</code>         |

You can also use the parameter as an input to a function. In the following example, the `paramTimeZone` parameter is merged with the values in the `Store_Nbr` to compute primary key `storeId` field:

**NOTE:** You cannot use the Merge transformation column for the following transformation, since it requires named columns as inputs.

|                                   |                                                      |
|-----------------------------------|------------------------------------------------------|
| <b>Transformation Name</b>        | New formula                                          |
| <b>Parameter: Formula type</b>    | Single row formula                                   |
| <b>Parameter: Formula</b>         | <code>merge([\$paramTimeZone,Store_Nbr], '-')</code> |
| <b>Parameter: New column name</b> | <code>'storeId'</code>                               |

### Example - parameter with multiple values

Suppose you wish to create a flow parameter that contains multiple values. Typically, you must track these values through an array, such as the following containing a set of colors:

```
["red","white","blue","black"]
```

Flow parameters that are literals are String values only. As a workaround, you can define the above as a Pattern .

| Setting | Value                               | Notes                                                                                                                                                                                                       |
|---------|-------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Name    | myColors                            |                                                                                                                                                                                                             |
| Value   | <code>`red white blue black`</code> | Note how the value is specified using backticks (`), which are used to indicate a Pattern .<br><br>The vertical bars are delimiters to separate the values, when they are processed within the application. |

Within your recipe, you can test for the presence of a parameter value. In the following transformation, a value of true is set in the new column isBlue if the value of \$myColors is blue:

|                                   |                                                |
|-----------------------------------|------------------------------------------------|
| <b>Transformation Name</b>        | New formula                                    |
| <b>Parameter: Formula type</b>    | Single row formula                             |
| <b>Parameter: Formula</b>         | <code>MATCHES([blue], \$myColors, true)</code> |
| <b>Parameter: New column name</b> | 'isBlue'                                       |

## Example - Integer parameter

Instead of segmenting the data by named time zone values, suppose your data is segmented by regions, which are numeric in number. Your flow parameter definition could look like the following:

| Setting | Value         | Notes                                                                                                                                                                                           |
|---------|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Name    | paramRegionId | Note the more appropriate name.                                                                                                                                                                 |
| Value   | 0             | In this case, there is no region identifier value 0. You choose to set the default to a value that is valid for the target data type (Integer) but is invalid for the scope of the data itself. |

To use this flow parameter as an integer, you must reference it wrapped in the `PARSEINT` function, which evaluates the input value against the Integer data type:

|                                   |                                           |
|-----------------------------------|-------------------------------------------|
| <b>Transformation Name</b>        | New formula                               |
| <b>Parameter: Formula type</b>    | Single row formula                        |
| <b>Parameter: Formula</b>         | <code>PARSEINT(`\${paramRegionId})</code> |
| <b>Parameter: New column name</b> | paramRegionId                             |

In the column histogram for the `paramRegionId` column, you can verify that the value 0 is present. Set an override outside at the flow level to insert a different value in the column.

For more information, see *PARSEINT Function*.

## Example - Date parameter

Suppose you need to be able to pass a date into the execution of a recipe. If no date is passed in, then the current time is used. The variable is declared as follows:



Instead of segmenting the data by named time zone values, suppose your data is segmented by regions, which are numeric in number. Your flow parameter definition could look like the following:

| Setting | Value     | Notes                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|---------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Name    | paramDate | Note the more appropriate name.                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| Value   |           | <p>In this case, the value is left empty to be overridden as needed in the application with the current timestamp.</p> <p>You should decide on the expected values for this parameter, as you must apply them to:</p> <ul style="list-style-type: none"> <li>Parameter overrides</li> <li>Recipe steps (e.g PARSEDATE function parameters)</li> </ul> <p>It may be easier to insert the format string here as the default value. For example:</p> <div>yyyy-mm-dd HH:MM:SS</div> |

You can use the following to insert the parameter value into your dataset. Note that the value is initially inserted as a String value, so the PARSEDATE function is used as a wrapper:

|                                   |                                                        |
|-----------------------------------|--------------------------------------------------------|
| <b>Transformation Name</b>        | New formula                                            |
| <b>Parameter: Formula type</b>    | Single row formula                                     |
| <b>Parameter: Formula</b>         | PARSEDATE( \${paramDate} , [ 'yyyy-mm-dd HH:MM:SS' ] ) |
| <b>Parameter: New column name</b> | paramDate                                              |

For more information, see *PARSEDATE Function*.

If the inserted value is empty or null, you can insert the current timestamp:

**Tip:** You could also overwrite invalid values in the following manner. However, that may mask problems with your inserted values.

|                            |                                                                 |
|----------------------------|-----------------------------------------------------------------|
| <b>Transformation Name</b> | Edit column with formula                                        |
| <b>Parameter: Columns</b>  | execDate                                                        |
| <b>Parameter: Formula</b>  | IF((execDate == '' )    ISNULL(execDate), NOW('UTC'), execDate) |

In the above, the value in `execDate` is tested to see if it is either:

- empty
- null

If so, the output of the NOW function is written. By default, this function returns the timestamp value at UTC time.

If there is a valid value, then it is written back to the column.

See *NOW Function*.

You can use the following to extract the time value from the parsed date param:

|                            |             |
|----------------------------|-------------|
| <b>Transformation Name</b> | New formula |
|----------------------------|-------------|

|                                   |                                  |
|-----------------------------------|----------------------------------|
| <b>Parameter: Formula type</b>    | Single row formula               |
| <b>Parameter: Formula</b>         | DATEFORMAT(execDate, 'HH:MM:SS') |
| <b>Parameter: New column name</b> | Time                             |

Since this value is not the parameter value specifically, the column name was listed simply as `Time`.

## Apply Parameter Override via API

When you run a job via the APIs, you can apply parameter overrides to the following parameter types:

- dataset parameters
- output parameters
- flow parameters

For more information, see *API Workflow - Run Job*.

# Flag for Review

## Contents:

- *Enable*
  - *Limitations*
  - *Flag for Review*
  - *Context menu*
    - *Mark as reviewed*
    - *Mark as pending review*
    - *Rename review step*
    - *Unflag for review*
- 

As needed, you can flag recipe steps for review in the recipe panel. You can use flags to set up checkpoints in your recipes, which enable flow users to evaluate the data, provide inputs, and sign off before jobs are executed based on the recipe.

## Examples:

- You could flag steps in recipes within flows that other users may copy. These flags and their related descriptions can be used to provide guidelines for how to implement the step in any copy.
- Among your collaborators, you may have experts in specific aspects of the data. You can flag steps for their review, perhaps even including their name in the description value for easy review.

## When you flag a step for review:

- The step is marked for review in the recipe panel.

**NOTE:** A flagged step must be reviewed before you can edit later steps in the recipe or run jobs based on the recipe.

- In Flow View, the recipe icon is highlighted with a warning.
- The Flow View page header summarizes the total number of flagged steps and recipes that are pending for review.
- If you have created a reference dataset, it is also highlighted with a warning wherever it is used.

## Enable

This feature may need to be enabled in your environment.

## Steps:

1. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
2. In the Admin Settings page, set the following to `true`:

```
"feature.haltExecution.enabled": true,
```

3. Save your changes and restart the platform.

For more information, see *Admin Settings Page*.

## Limitations

- When a step is flagged for review, all downstream steps are disabled.
  - Steps must be reviewed in descending, top-to-bottom order.
- You cannot run the job until all flagged steps are reviewed.
- Flags can be applied and cleared one at a time.
- Undo / Redo options are not applicable to flagged steps.
- Flag for Review is not supported for the following features: Join, Union, Standardize, Transform by Example, or Macros.

**Tip:** You can flag the following step or add a comment step and flag the comment if you want to call attention for these transforms.

## Flag for Review

### Steps:

1. In the Recipe panel, select the step to flag.
2. From the Recipe toolbar context menu, select **Flag for review**.
3. In the Flag for review dialog:
  - a. (optional) Enter a name or title for the flag.
  - b. (optional) Enter a description.
  - c. Click **Flag**.
4. A warning icon is displayed over the selected step. You can hover the warning icon to read the description.
5. The step has been marked for review.

**NOTE:** When one or more steps has been flagged in your recipe, the **New Step** and **Run** options are disabled.

## Context menu

The following menu options are available in the context menu for flagged steps.

### Mark as reviewed

From the Recipe toolbar context menu, select the required step and select **Mark as reviewed**. A tick mark is displayed over the reviewed step, indicating that the step has been cleared.

**NOTE:** If there are no additional flagged steps, you can add new recipe steps or run jobs for your recipe.

### Mark as pending review

Revert the Mark as reviewed flag. The tick mark is replaced by the warning icon over the selected step.

**NOTE:** You can toggle between **Mark as reviewed** and **Mark as pending review** options to mark the review as complete or to mark the step as pending review.

### Rename review step

Edit the name and description values.

**Tip:** You can add hyperlinks as part of the Description value.

### Unflag for review

Removes the flag for review from the step.

**NOTE:** The step is now cleared of the flag. If there are no additional flagged steps, you can add new recipe steps or run jobs for your recipe.

# Manage Environment Parameters

## Contents:

- *Create Environment Parameter*
- *Use Environment Parameter*
  - *Limitations*
- *Edit Environment Parameter Value*
- *Delete environment parameter*
- *Export environment parameters*
- *Import environment parameters*

You can define parameters that are applicable across the entire project or workspace environment.

An **environment parameter** is a variable of String type defined by an administrator that any user of the environment can reference in their flows and flow-related objects.

**NOTE:** You must be a project owner or workspace administrator to manage environment parameters.

For more information on parameters, see *Overview of Parameterization*.

## Create Environment Parameter

### Steps:

1. A project owner or workspace administrator can select **User menu > Admin console > Environment parameters**.
2. In the Environment Parameters page, click **Create**.
3. Specify the parameter:
  - a. **Name:** Enter the display name for the parameter.

**NOTE:** All environment parameter names are automatically prepended with `env..`

- b. **Default value:** Enter the default value.

**NOTE:** The default value is stored as a String value.

4. To create another environment parameter, click **Add another**.
5. To save your changes, click **Save**.

For more information, see *Environment Parameters Page*.

## Use Environment Parameter

Environment parameters can be referenced in the following locations:

**Tip:** When specifying a variable, enter `$env` to see the list of available environment parameters.

- Parameterized datasets. See *Create Dataset with Parameters*.
- Datasets created with SQL. See *Create Dataset with SQL*.
- Output paths. See *Create Outputs*.

## Limitations

- You cannot use environment parameters in recipes.
- You cannot use environment parameters in plans.
- Environment parameter names must be unique within the project or workspace. You can apply override values to them at runtime.
- You cannot use environment parameters in Deployment Manager. For more information, see *Overview of Deployment Manager*.

## Edit Environment Parameter Value

Administrators can change the default value for an environment parameter.

**NOTE:** Modifying the default value of an environment parameter immediately applies the change across the entire environment. All subsequent job and plan runs are affected.

**NOTE:** After you have created an environment parameter, you cannot change the name. You must create a new environment parameter.

### Steps:

1. In the Environment Parameters page, locate the parameter whose default value you wish to modify.
2. In the More menu, select **Edit value**.
3. Enter a new value, and click **Save**.

## Delete environment parameter

**When you delete an environment parameter, all references to the parameter are converted to empty string values. Job executions can fail, and recipe steps can break.**

**Tip:** If you delete an environment parameter and then recreate it using the exact same name, references to the parameter are updated with the new default value, which replaces the empty string value for the deleted parameter.

### Steps:

1. In the Environment Parameters page, locate the parameter whose default value you wish to modify.
2. In the More menu, select **Delete**.

## Export environment parameters

You can export the environment parameters from your project or workspace.

**NOTE:** All environment parameters are exported at the same time into a ZIP file. Do not modify this file outside of the Designer Cloud application .

**Steps:**

1. In the Environment Parameters page, select **More menu > Export**.
2. The ZIP file is downloaded to your local desktop.

## Import environment parameters

If you have exported a set of environment parameters, you can import them into another workspace or project.

**NOTE:** If an environment parameter that you are importing has a name that conflicts with an environment parameter that already exists, you must either rename the imported parameter or delete it from the import set.

**Steps:**

1. In the Environment Parameters page, select **More menu > Import**.
2. Select or drag-and-drop the ZIP file. Click **Import**.

**NOTE:** Select the ZIP file or its embedded JSON5 file for import.

3. The Import environment parameters dialog is displayed:

| Name               | Default value     |
|--------------------|-------------------|
| env. region        | 01                |
| env. bucket_name   | myBucket          |
| env. name          | Prod              |
| env. admin_contact | admin@example.com |

[Add another](#)

Cancel Save

**Figure: Import environment parameters dialog**

4. Review each environment variable and its assigned value from the import package:
  - a. Modify values as needed.
  - b. To delete a parameter from the import process, click the Trash icon.
  - c. To add another parameter as part of the import package, click **Add another**.
5. To save your changes and complete the import, click **Save**.
6. The environment parameters and your modifications to them are imported.



# User Management Tasks

The topics below provide information on how to manage aspects of your account in Designer Cloud powered by Trifacta® Enterprise Edition.

# Change Password

To recover your password, click **Forgot password?** in the login screen.

See *Login*.

To change your password after you have logged in, select your name from the User menu. Enter a new password, confirm it, and save your changes. The new password is applied when you next try to log in. See *User Profile Page*.

**The password for the administrator account should be changed immediately after installation is complete. See *Change Admin Password*.**

# Configure Your Access to S3

## Contents:

- *Getting Started*
  - *Credential Provider*
    - *IAM Role*
    - *AWS Key and Secret*
- 

If per-user access to S3 has been enabled in your Trifacta® deployment, you can apply your personal S3 access credentials through the AWS Credentials page. You can use the following properties to define the S3 buckets to use for uploads, job results, and temporary files.

## Getting Started

You can access these settings through the Designer Cloud application .

### Steps:

1. In the menu bar, click the User menu.
2. Select **Storage**. click **Edit** for AWS Credentials and Storage Settings, where you can review and modify your S3 access credentials.

## Credential Provider

### IAM Role

**NOTE:** This role must be created through AWS for you. For more information, please contact your AWS administrator.

**Tip:** This method is recommended for access AWS resources.

AWS Storage Settings

✕

Please see [AWS Config Settings](#) for help completing this form.

**Credential Provider**

✓
IAM Role

AWS Key and Secret

**Available IAM Role ARNs**

**Select Default IAM Role ARN**

**Default S3 Bucket**

This bucket will contain uploaded files, temporary files, and job results.

Cancel

Save

**Figure: Apply your IAM role and credentials**

| Setting                     | Description                                                                                                                                                                                                                                                                 |
|-----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Available IAM Role ARNs     | You can specify the set of IAM Role ARN to use to authenticate to AWS resources.                                                                                                                                                                                            |
| Select Default IAM Role ARN | From the available IAM Role ARNs, you can specify the default one.                                                                                                                                                                                                          |
| Default S3 Bucket           | This bucket is used for storage, unless another bucket is explicitly selected. <div style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;"> <p><b>NOTE:</b> Specify the top-level bucket name only. There should not be any backslashes in your entry.</p> </div> |
| Extra S3 Buckets            | You can specify a comma-separated string of additional S3 buckets that are available for storage. Do not put any quotes around the string. Whitespace between string values is ignored.                                                                                     |

## AWS Key and Secret

Per-user access must be enabled by your Trifacta administrator. See [S3 Access](#).

AWS Storage Settings

×

Please see [AWS Config Settings](#) for help completing this form.

Credential Provider

IAM Role

✓

AWS Key and Secret

AWS Access Key

AWS Secret Key

\*\*\*\*\*

Default S3 Bucket

3fac-testing

This bucket will contain uploaded files, temporary files, and job results.

Cancel

Save

Figure: AWS Storage page

The following settings apply to S3 access.

**NOTE:** The values that you should use for these settings should be provided by your S3 administrator. If they have already been specified, do not modify unless you have been provided instructions to do so.

| Setting           | Description                                                                                                                                                                                                 |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| AWS Access Key    | This key defines the account to use to connect to AWS.                                                                                                                                                      |
| AWS Secret Key    | The secret (or password) associated with the key.                                                                                                                                                           |
| Default S3 Bucket | <div>This bucket is used for storage, unless another bucket is explicitly selected.<div><b>NOTE:</b> Specify the top-level bucket name only. There should not be any backslashes in your entry.</div></div> |
| Extra S3 Buckets  | You can specify a comma-separated string of additional S3 buckets that are available for storage. Do not put any quotes around the string. Whitespace between string values is ignored.                     |

# Concepts

This section contains topics on concepts related to Designer Cloud powered by Trifacta® Enterprise Edition and the underlying principles driving its development.

# Feature Overviews

Review the overall capabilities of significant features of Designer Cloud powered by Trifacta® Enterprise Edition.

**Tip:** Use the links in these sections to access locations in the platform where these features appear.

# Overview of Data Export

## Contents:

- *How to Export*
- *Export Job Results*
  - *Writing to Files*
  - *Writing to Tables*
  - *Parameterized Outputs*
- *Ad-hoc Publishing*
- *Exporting Metadata*
  - *Export flows*
  - *Export recipes*
  - *Export sample data*
  - *Export logs*
- *Export via API*
  - *Job results*

This section provides an overview of exporting data from the Designer Cloud® application to your preferred destinations, such as file-based storage, connected datastores, or your desktop. In addition to exporting of job results, other types of exports are covered in this section.

**Tip:** In most cases, the source of your data does not limit the type of output that you can generate. You can create a file-based imported dataset and generate results to a database table. Some exceptions may apply.

## How to Export

Job results are generated based on the specifications of an output object. An **output object** is a reference object for one or more types of outputs. This reference information includes full path to the output location, file or table name, and other settings. For more information, see *Create Outputs*.

In the Run Job page, you can specify additional settings and overrides. See *Run Job Page*.

## Export Job Results

After you have executed a job, the application writes a set of results to the designated output locations. These results are the application of the recipe's transformation steps to the imported dataset, written to the location or locations specified in the output object in the specified output format.

You can export the results directly from the designated output destination. For more information, see *Job Details Page*.

**Tip:** Job results for your latest job may be exportable from Flow View. For more information, see *View for Outputs*.

## Writing to Files

As a result of job execution, you can publish your outputs to a file-based system.



**NOTE:** You must have write permissions to the location where you are writing your output files. These permissions should be set up during initial configuration of the product. For more information, please contact your administrator.

Defaults for file-based outputs:

- Files are written to your designated output directory on the backend datastore. As needed, you can modify your default output directory. For more information, see *Storage Config Page*.
- Files are written in CSV format to the designated location.

You can modify the publishing action and generate results in your preferred formats.

- For more information on changing file output settings, see *File Settings*.
- For more information on supported file formats, see *Supported File Formats*.

## Writing to Tables

You can export generated results directly to a connected relational database.

**Tip:** Some relational connection types support read-only or write-only connections.

The Designer Cloud application writes results to a database through an object called a connection. A **connection** is a configuration object that defines the interface between the application and the database. Among its properties are a set of credentials that provide access.

**NOTE:** You must have write permissions to the database where you are writing your output tables. These permissions must be enabled by a database administrator outside of the product.

**NOTE:** Connections can be shared among users. When a user chooses to share a connection, the user can also choose to share credentials. If credentials are not shared, other users must provide their own credentials if they wish to use the connection. For more information, see *Share a Connection*.

For relational databases, the Designer Cloud application passes the information in the connection definition to a third-party driver that performs the actual connection. Thereafter, the Designer Cloud application maintains the open connection as long as it is needed to write results. After the results are written, the connection is closed.

When you choose to write results to a table:

- Through the connection, you browse and select the database to which to write the results.
- You can choose to write to an existing table or to a new one.
- You can specify one of the following publishing actions on the table you selected:
  - **New:** Each run generates a new table with a timestamp appended to the name. For example, `myexample_test_1.csv`.
  - **Update:** Each run adds any new results to the end of the table.
  - **Truncate:** With each run, all rows columns of data in a table is removed and retain the empty table as an object.
  - **Load:** With each run, the table is dropped (deleted), and all data is deleted. A new table with the same name is created, and any new results are added to it.
  - **Merge:** Some databases may support merge (upsert) operations.

Additional options may be available, depending on the connection. For more information, see *Relational Table Settings*.

## Parameterized Outputs

For file- or table-based publishing actions, you can parameterize elements of the output path. You can create parameters for your outputs of the following types:

- **Timestamp:** You can insert the timestamp of when the output was written as part of the path to the output location.
- **Variable:** Variable parameters allow you to insert values that you define as part of the output object.

**Tip:** You can optionally override the values of your variable parameters as part of your job definition. For more information, see *Run Job Page*.

For more information on parameters, see *Overview of Parameterization*.

## Ad-hoc Publishing

After a job has successfully completed, you can review and download the set of generated outputs and export results. Optionally, you may be able to publish the generated results to a secondary datastore through the Job Details page.

**NOTE:** Additional configuration may be required.

For more information on ad-hoc publishing, see *Publishing Dialog*.

## Exporting Metadata

In addition to the job results, you can export aspects of the flow definition and other objects that you have created in the Designer Cloud application. These exports can be useful for:

- Migrating flows to other workspaces
- Archiving data
- Taking snapshots of work in progress

## Export flows

You can export a flow from application. An exported flow is stored in a ZIP file that contains references to all objects needed to use the flow in another workspace or project. Exported flows can be imported into the same workspace/project or a different one.

**NOTE:** Users of the imported flow must have access to the datasources and specified output locations. If not, these objects must be remapped in the new environment.

For more information, see *Export Flow*.

## Export recipes

You can download a recipe in text form and reuse it in another flows.

### Reuse recipes in a different environment

If you need to reuse a recipe in a different instance of Designer Cloud powered by Trifacta Enterprise Edition, you can do the following:

- Export the entire flow and import it into the new environment. Open the flow in the new environment. For more information, see *Export Flow*.
- Convert all steps of a recipe into a macro. Export the macro and then import it into the new environment. For more information, see *Export Macro*.

### Download recipes

You can download recipe in a text form of Wrangle (a domain-specific language for data transformation). For more information, see *Recipe Panel*.

### Export sample data

From the recipe panel, you can download the current state of the data grid, which includes the current sample plus any recipe steps that have been applied to it.

**Tip:** When a sample is taken, it is tied to the current recipe step. All steps later in the recipe than the current recipe step are computed in memory using the sample as the baseline. For more information, see *Overview of Sampling*.

For example, if the sample was generated when the recipe cursor was displaying step 7 and you download the data from the recipe when the recipe cursor is on step 10, then you are downloading the state of the recipe at step 10. For more information, see *Recipe Panel*.

**NOTE:** When a flow is shared, its samples are shared with other users. However, if shared users do not have access to the underlying sources that back a sample, they do not have access to the sample. These samples are invalid for the other users, who must create their own.

For more information, see *Samples Panel*.

### Export logs

You can export logs of the following:

- **Download session logs:** You can download logs for your current from the Designer Cloud powered by Trifacta platform through the Help menu. For more information, see *Download Logs Dialog*.

**Tip:** Administrators can download a broader set of platform logs. For more information, see *Admin Download Logs Dialog*.

- **Job logs :** When a job fails, you can download job logs for analysis. For more information, see *Jobs Page*.
- **Sample logs:** If a sample fails to generate, you can retry or download logs for review. For more information, see *Samples Panel*.

### Export via API

#### Job results

After a job has run, you can acquire the path to the results when you query for the job. For more information, see <https://api.trifacta.com/ee/es.t/index.html#operation/runJobGroup>

# Overview of Data Import

## Contents:

- *How to Import*
  - *Types of Import*
    - *Upload*
    - *Import of files*
    - *Import of tables*
  - *Imported Datasets*
  - *Persisted Data*
    - *Samples*
    - *Conversion*
    - *Caching*
  - *Sharing Imported Data*
- 

This section provides an overview of data import and how different types of import are handled in Designer Cloud powered by Trifacta® Enterprise Edition.

## How to Import

You import data for use in the Designer Cloud application through a reference object called an **imported dataset**. An imported dataset is a reference to the source of the data.

**NOTE:** The source data is never modified. In some cases, the source data may be copied to the base storage layer. For example, data that is uploaded from your local desktop must be copied to the base storage layer so that it is accessible to you and potentially other users of the Designer Cloud application.

## Steps:

1. In the Designer Cloud application, click the Library icon in the left navigation bar.
2. In the Library, click **Import Data**.
3. The Import Data page is displayed.
  - a. Select the connection in the left nav bar through which you can access the data.
  - b. For more information, see *Import Data Page*.

After the data has been imported, you can reference it within the application as an imported dataset. For more information, see *Import Basics*.

## Types of Import

You can import datasets or select datasets from sources that are stored on file-based storage, connected datastores, or your desktop. Following are the different types of import that you can perform in the Import Data page.

### Upload

You can upload a variety of flat file formats from your local desktop. You can upload a file up to 1 GB in size.

- You can drag and drop files from your desktop to upload them.
- You can select multiple files in the same directory for uploading at the same time.

For more information, see *Import Data Page*.

## Import of files

Designer Cloud powered by Trifacta Enterprise Edition supports multiple storage environments. You can import one or more files from any backend data storage systems. Each workspace has a default backend storage environment. Each user should be able to import files that are stored in accessible locations in this backend storage area.

**NOTE:** You must have read permissions for these storage environments to import the file. These permissions should be set up during initial configuration of the product. For more information, please contact your administrator.

**NOTE:** During import, the Designer Cloud powered by Trifacta Enterprise Edition identifies file formats based on the extension of the filename. For more information on the formats supported for import, see *Supported File Formats*.

## Import of tables

You can import one or more tables from relational datastores. Through the Import Data page, you can select or create the appropriate connection to the datastore, navigate to the required database and select the files to be imported.

**NOTE:** You must have read permissions for any database that you want to import. These permissions must be enabled by a database administrator outside of the product. For more information, see *Using Databases*.

## Imported Datasets

When you import a file or a table, the data that is imported to the platform is referenced as an imported dataset. An imported dataset is simply a reference to the original data. An **imported dataset** can be a reference to a file, multiple files, database table, or other type of data.

**NOTE:** Designer Cloud powered by Trifacta® Enterprise Edition does not modify the source data. It is only referenced as an imported dataset.

**NOTE:** The imported dataset may be broken if the path or the permissions change for the underlying dataset.

## Persisted Data

In general, the Designer Cloud application does not retain data for a longer time than the data is explicitly needed. For example, when jobs are executed on Trifacta Photon, the source data is streamed to the Trifacta node and transformed, after which results are written. The transformed data is not maintained in the Trifacta node.

**NOTE:** Data is not persisted on the Trifacta node.

More information on persisted data is available below.

## Samples

Samples can be generated within the product through the Samples panel. When a sample is created, it is stored within your storage directory on the backend datastore. You can create a new sample at any time.

- If the source data is larger than 10 MB in size, an initial sample is automatically generated when the recipe is first loaded in the Transformer page. This sample contains the first set of rows in the dataset up to 10 MB in size.
- If the source data contains multiple files, the initial sample for the dataset is generated from the first set of rows in the first filename listed in the directory.
- If that source data is a multi-sheet Excel file, the sample is taken from the first sheet in the file.

For more information on creating samples, see *Overview of Sampling*.

## Conversion

For some file types, the Designer Cloud application must convert the source data into a format that is natively supported by the product. This process happens as part of the importing of data for use in the Designer Cloud application and is managed by the conversion service in the platform. In such scenarios, the data is read from the source and passed through the conversion service, which understands how to read the source format and can write it to a supported text format. This text version of the source data is written to the base storage layer.

For example, when a transformation job is executed, the original source data is passed through the conversion service, and the converted data is used for job execution. When the job results are written, conversion service removes the converted data.

During import, the Designer Cloud application identifies file formats based on the extension of the filename. The conversion process applies for the following type of files:

- **XLS and PDF:** These file types are stored in a proprietary binary format. Conversion service uses a set of libraries to convert files of these types to tabular CSV data and store the files in the base storage layer.
- **JSON:** JSON file through the conversion service provides considerable improvements in terms of quality and performance during ingestion of JSON data.

For more information, see *Supported File Formats*.

## Caching

Caching refers to the process of ingesting and storing data sources in a temporary backend location for a specific period of time in order to perform any additional operations in a faster way.

Instead of reloading the source each time that an object is referenced, the Designer Cloud application checks the cache for a cached version and if the cache is still valid. Based on the results, the Designer Cloud application pulls data from the local cache instead.

**Tip:** Cached objects can be referenced later for faster performance on tasks such as sampling and job execution.

For more information, see *Configure Data Source Caching*.

## Sharing Imported Data

You cannot share an imported dataset as an object; however, you can share connections. If the user has permissions over the dataset that has been shared as a part of the connection then the imported dataset is accessible to the shared user.

**NOTE:** The shared user should have the connection credentials to access the imported dataset.

For more information, see *Overview of Sharing*.

For more information, see *Share a Connection*.

# Overview of Storage

## Contents:

- *Base Storage Layer*
    - *Uses of base storage layer*
    - *Base storage layer directories*
    - *Available base storage layers*
    - *Management of base storage layer*
  - *External Storage*
    - *File-based systems*
    - *Cloud data warehouses*
    - *Relational systems*
    - *Management of external storage*
- 

Designer Cloud powered by Trifacta Enterprise Edition supports different options for reading and writing data from your storage systems.

## Base Storage Layer

The base storage layer is the datastore where Designer Cloud powered by Trifacta Enterprise Edition uploads data, generates profiles, results, and samples. By default, job results are written on the base storage layer. You can configure the base storage layer and other required settings.

**Tip:** The base storage layer must be a file-based system.

**NOTE:** The base storage layer must be enabled and configured during initial installation. After the base storage layer has been configured, it cannot be switched to another environment. For more information, see *Set Base Storage Layer*.

## Uses of base storage layer

In general, all base storage layers provide similar capabilities for storing, creating, reading, and writing datasets.

The base storage layer enables you to perform the following functions:

1. **Storing datasets:** You can upload or store datasets in directories on the base storage layer. See below.
2. **Creating datasets:** You can read in from datasources stored in the storage layer. A source may be a single file or a folder of files.
3. **Storage of samples:** Any samples that you generate are stored in the base storage layer.
4. **Ingested data:** Some data like Excel and PDF are stored as binary (non-text) files. These files must be read and converted to CSVs, which are stored on the base storage layer.
5. **Cached data:** You can enable a cache on the base storage layer, which allows data that has been ingested to remain on the base storage layer for a period of time. This cache allows for faster performance if you need to use the data at a later time.
6. **Writing Results:** After you run the job, you can write the results to the storage layer.



## Base storage layer directories

Designer Cloud powered by Trifacta Enterprise Edition creates and maintains the following directories and their sub-directories on the base storage layer:

| Directory              | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| /trifacta/uploads      | Storage of datasets uploaded through the Designer Cloud application . Directories beneath this one are listed by the internal identifier for each user of the product who has uploaded at least one file.<br><div><b>Avoid using /trifacta/uploads for reading and writing data. This directory is used by the Designer Cloud application .</b></div>                                                                                                                                                                                                                                                                                         |
| /trifacta/queryResults | Default storage of results generated job executions. Directories beneath this one are listed by the internal identifier for each user of the product who has run at least one job.<br>For each user, these sub-directories are the default storage location for job results. These locations can be modified. See <i>Preferences Page</i> . Within the <code>queryResults</code> directory, you may find sub-directories labeled <code>datasourceCache</code> . When data source caching is enabled, data read into the product may be temporarily stored in this directory. For more information, see <i>Configure Data Source Caching</i> . |
| /trifacta/dictionaries | Storage of custom dictionary files uploaded by users.<br><div><b>NOTE:</b> This feature applies to Designer Cloud powered by Trifacta Enterprise Edition only. It is not often used.</div>                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| /trifacta/tempfiles    | Temporary storage location for files required for use of the product.<br><div><b>NOTE:</b> The <code>tempfiles</code> directory is reserved for use by the platform. It is the only directory of these that is actively cleaned by the platform.</div>                                                                                                                                                                                                                                                                                                                                                                                        |

## Minimum Permissions

Designer Cloud powered by Trifacta Enterprise Edition requires the following operating system level permissions on the listed directories and sub-directories:

| Directory              | Owner Min Permissions | Group Min Permissions | World Min Permissions |
|------------------------|-----------------------|-----------------------|-----------------------|
| /trifacta/uploads      | read+write+execute    | none                  | none                  |
| /trifacta/queryResults | read+write+execute    | none                  | none                  |
| /trifacta/dictionaries | read+write+execute    | none                  | none                  |
| /trifacta/tempfiles    | read+write+execute    | none                  | none                  |

## Available base storage layers

Designer Cloud powered by Trifacta Enterprise Edition supports the following base storage layers.

**NOTE:** In some deployments, the base storage layer is pre-configured for you and cannot be modified. After the base storage layer has been defined, you cannot change it.

**NOTE:** For all storage layers, the source data is untouched. Results are written to a location whenever a job is executed on a source dataset.

For more information, see *Storage Deployment Options*.

## S3

Simple Storage Service (S3) is an online data storage service provided by Amazon, which provides low-latency access through web services. For more information, see <https://aws.amazon.com/s3/>.

For more information, see *External S3 Connections*.

## HDFS

HDFS is a scalable file storage system for use across all of the nodes (servers) of a Hadoop cluster. Many interactions with HDFS are similar with desktop interactions with files and folders. However, what looks like a "file" or "folder" in HDFS may be spread across multiple nodes in the cluster. For more information, see [https://en.wikipedia.org/wiki/Apache\\_Hadoop#HDFS](https://en.wikipedia.org/wiki/Apache_Hadoop#HDFS).

**NOTE:** If you are using impersonated access on the base storage layer, then, the group minimum permissions must be read+write+access on all of the above directories and sub-directories.

For more information, see *Using HDFS*.

## ADLS Gen 1

ADLS is a scalable file storage system for use across all of the nodes (servers) of a cluster. Many interactions with ADLS are similar with desktop interactions with files and folders. However, what looks like a "file" or "folder" in ADLS may be spread across multiple nodes in the cluster. For more information, see <https://docs.microsoft.com/en-us/azure/data-lake-store/data-lake-store-overview>.

## ADLS Gen 2

Microsoft Azure deployments can integrate with the next generation of Azure Data Lake Store (ADLS Gen2). Microsoft Azure Data Lake Store Gen2 (ADLS Gen2) combines the power of a high-performance file system with massive scale and economy. Azure Data Lake Storage Gen2 extends Azure Blob Storage capabilities and is optimized for analytics workloads. For more information, see <https://azure.microsoft.com/en-us/services/storage/data-lake-storage/>.

## WASB

WASB is a scalable file storage system for use across all of the nodes (servers) of a cluster. As with HDFS, many interactions with WASB are similar with desktop interactions with files and folders. However, what looks like a "file" or "folder" in WASB may be spread across multiple nodes in the cluster.

## Management of base storage layer

Maintenance of the base storage layer must be in accordance with your enterprise policies.

**Unless the base storage layer is managed by Alteryx, it is the responsibility of the customer to maintain access and perform any required backups of data stored in the base storage layer.**

**NOTE:** Except for temporary files, the Designer Cloud powered by Trifacta platform does not perform any cleanup of the base storage layer.

## External Storage

You can create connections to external storage systems. You can integrate Designer Cloud powered by Trifacta Enterprise Edition with an external datastore. Depending on the type of connection and your permissions, the connection can be:

- read-only
- write-only
- read-write

You can create and edit connections between Designer Cloud powered by Trifacta® Enterprise Edition and external data stores. You can create either file-based or table-based connections to individual storage units, such as databases or buckets.

**NOTE:** In your environment, creation of connections may be limited to administrators only. For more information, contact your workspace administrator.

**Tip:** Administrators can edit any public connection.

**NOTE:** After you create a connection, you cannot change its connection type. You must delete the connection and start again.

For more information, see *Connection Types*.

## File-based systems

In addition to the base storage layer, you may be able to connect to other file-based systems. For example, if your base storage layer is HDFS, you can also connect to S3.

**NOTE:** If HDFS is specified as your base storage layer, you cannot publish to Redshift.

For more information, see *Connection Types*.

## Cloud data warehouses

The Designer Cloud application can be leveraged for loading and transforming data in data warehouses in the cloud. These integrations offer high performance access to reading in datasets from these and other sources, performing transformations, and writing results back to the data warehouse as needed.

## Relational systems

When you are working with relational data, you can configure the database connections after you have completed the platform configuration and have validated that it is working for locally uploaded files.

**NOTE:** Database connections cannot be deleted if their databases host imported datasets that are in use by Designer Cloud powered by Trifacta Enterprise Edition. Remove these imported datasets before deleting the connection.

For more information, see *Using Databases*.

## Management of external storage

To integrate with an external system, the Designer Cloud application requires:

- Basic ability to connect to the hosting node of the external system through your network or cloud-based infrastructure
- Requisite permissions to support the browsing, reading and/or writing of data to the storage system
- A defined connection between the application and the storage system.

Except for cleanup of temporary files, the Designer Cloud application does not maintain external storage systems.

# Overview of Predictive Transformation

## Contents:

- Overview
  - Phases
    - Visualizations
    - Selections
    - Predictive Model
    - Suggestions and Their Variants
    - Previews
    - Additional Steps - Modification
  - Wrangle
- 

Based in academic research, **Predictive Transformation** refers to a set of design and interface principles that serve as the foundation for how Trifacta® users interact with their data. Predictive transformation is the linchpin of the platform. This section provides an overview of the concepts and links to locations where these concepts are surfaced in the interface.

Predictive Transformation is a registered trademark of Alteryx.

## Overview

In essence, Predictive Transformation seeks to bring closer together:

1. the domain knowledge about the data, and
2. the technical knowledge of the sometimes complex operations required to render data into its final usable format.

In data wrangling, the former knowledge set resides with domain experts who understand the meaning of the data, while the latter often requires involvement of IT, which may have no contextual understanding of the data to inform their solution designs.

This process of rendering data from one format into another is generally called **data transformation**, which breaks down into a set of programming-type tasks, with an emphasis on structure, meaning, and the statistical properties of the data. These tasks include:

- statistical manipulation (profiling, outliers, imputation)
- restructuring (data extraction, nesting, pivot/unpivot)
- cleaning (standardization, deduplication, data removal)
- enrichment (join with other data, lookups of reference data)
- distillation (sampling, filtering, aggregation, windowing)

Across large, distributed datasets, these tasks can be technically challenging to properly execute. To move them out of the IT domain, Predictive Transformation seeks to deliver the following capabilities:

1. Features & Visualizations - innovative methods to display and select data of interest
2. Suggestions - based on user selection, suggested transforms are presented to you for selection and configuration
3. Previews - for the selected suggestion, previews of the anticipated change are available for review prior to inclusion in the transformations on the dataset

The above cycle is repeated over and over until the set of transformations is defined and executed to satisfaction.

## Phases

Based on user selection, Predictive Transformation **guides** you toward possible next steps yet allows you to **decide** the step to take and (if necessary) refine the step definition. The core of the guide/decide loop of Predictive Transformation fits into the following iterative phases. When steps are selected, visualizations are updated, and the cycle repeats again.

| Phase     | UI Element                     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|-----------|--------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Visualize | visualizations                 | A critical component of Predictive Transformation is the visual representation of the data, including items of interest for selection. In larger data sets, the visual cues around items of interest and the tools for interacting with them provide information on the meaning of each type of interaction and are critical for a productive and pleasant user experience.                                                                                                                                                        |
| Interact  | selections                     | You interact directly with the visualizations to select values, columns, or other items of interest.                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Predict   | predictive model & suggestions | Automatically, user selections trigger queries into the predictive model. Data, metadata, and the selection of it effectively define queries of the predictive model. The model returns a set of suggested transforms. The suggestions guide you toward recommended actions on items that you has decided, through selection, are interesting. You can then decide which suggestion to undertake, including modification of the specific parameters around the suggestion. Or, you can define a completely different step to take. |
| Present   | previews                       | Whenever the step to take is selected or subsequently modified, the anticipated results of that step are displayed as a preview overlay on top of the data. This method allows for easy development, rapid undoing, and a clearer understanding of the impacts of each step.                                                                                                                                                                                                                                                       |

## Visualizations

In Predictive Transformation, visualizations must be carefully designed to surface selectable data or metadata of interest. In Designer Cloud powered by Trifacta Enterprise Edition, the Transformer page has been designed to represent the underlying dataset while guiding you with selectable items.

| #          | FMID                                                            | MarketName                             | Primary_Website_or_URL                     | Address       |
|------------|-----------------------------------------------------------------|----------------------------------------|--------------------------------------------|---------------|
| 1M - 1.01M | 38 Categories                                                   | 14 Categories                          | 37 Categories                              | 27 Categories |
| 1005969    | ???Y Not Wednesday Farmers Market at Town Center???             | http://www.sandlercenter.org/ynotwed   | 201 Market Street                          | Virginia      |
| 1008044    | 10:10 Farmers Market                                            | http://www.1010farmersmarket.com       | 5960 Stewart Parkway                       | Douglas       |
| 1000618    | 100-Mile Market                                                 | http://www.peoples-op.org/             | 507 Harrison Street                        | Kalamazoo     |
| 1002454    | 112st Madison Avenue                                            | null                                   | 112th Madison Avenue                       | New York      |
| 1005586    | 12th & Brandywine Urban Farm Market                             | null                                   | 12th & Brandywine Streets                  | New Cast      |
| 1008071    | 14&U Farmers' Market                                            | null                                   | 1400 U Street NW                           | District      |
| 1000059    | 175th Street Greenmarket                                        | http://www.grownyc.org                 | W 175 St. & Broadway                       | New York      |
| 1003877    | 17th Ave Market                                                 | http://www.iatp.org/minimarkets        | 1622 6th St NE                             | Hennepin      |
| 1000709    | ???19th Annual Highlands Business Partnership Farmers Market??? | http://www.highlandsnj.com             | 71 Waterwitch Avenue                       | Monmouth      |
| 1003233    | 2012 Wood County Farmers' Market                                | null                                   | 555 W. Grand Ave.                          | Wood          |
| 1005309    | 22nd and Tasker Farmers' Market                                 | http://www.foodtrustmarkets.org/market | 22nd and Tasker Streets                    | Philadelph    |
| 1006576    | 25th Avenue Farmers' Market                                     | http://www.pcfma.com/market_home.php   | 194 W 25th Avenue                          | San Mateo     |
| 1005299    | 29th and Wharton Farmers' Market                                | http://www.foodtrustmarkets.org/gray   | 29th and Wharton Streets                   | Philadelph    |
| 1004950    | 3 French Hens French Country Market                             | http://www.3frenchhensmarket.blogspot  | 123 W. Illinois ave.                       | Grundy        |
| 1005636    | 32nd Street/Waverly Farmers Market                              | http://www.32ndstreetmarket.org        | E. 32nd & Barclay Street                   | Baltimore     |
| 1005310    | 33rd and Diamond Farmers' Market                                | http://www.foodtrustmarkets.org/market | N 33rd and Diamond Streets                 | Philadelph    |
| 1003161    | 38th & Meridian Farmers Market                                  | null                                   | 3808 North Meridian Street                 | Marion        |
| 1004414    | 3rd & Curry St. Farmers Market                                  | http://www.carsonfarmersmarket.com     | 3rd & Curry Street                         | Carson C      |
| 1002860    | 3rd Street N (hwy 11) beside Jack's Restaurant                  | null                                   | 3rd St N (Hwy 11) beside Jack's Restaurant | Etowah        |
| 1005304    | 52nd and Haverford Farmers' Market                              | http://www.foodtrustmarkets.org/market | N 52nd Street and Haverford Avenue         | Philadelph    |
| 1000060    | 57th Street Greenmarket                                         | http://www.grownyc.org                 | W 57 St. & 9 Ave                           | New York      |

**Figure: Transformer Page contains a rich overlay of information and selection cues**

Specific visualization cues:

1. Data rendered into familiar grid format, regardless of underlying structure
  - a. selectable values and columns
2. Color-coded **data quality bars**:
  - a. green: valid
  - b. gray: missing

- c. red: invalid (checked against data type)
  - d. Select a color to select all corresponding values
3. **Histograms** for individual columns:
  - a. Select one or more values in the histogram highlights corresponding values in other column histograms for easy visual comparisons
4. **Metadata** on entire dataset and type and statistical information for individual columns. See *Column Details Panel*.

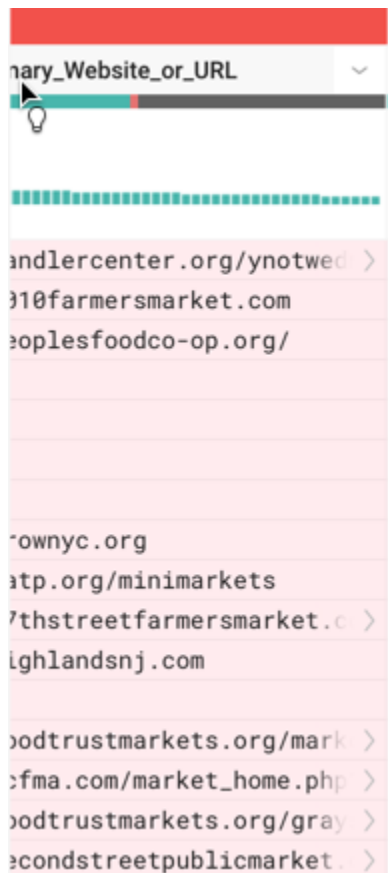
In this manner, this visualization lifts user interaction from the domains of data and code into a more visual representation.

You must still specify via selection; the syntax of the specification is lifted into the visual domain, and the details of crafting the technical query are managed by the application.

**Exploration:** By design, this interaction model supports both detailed specificity and ambiguity. You select, preview the results, and then determine if the preview meets expectations. Additionally, all steps can be undone and removed from the recipe, so that you can explore different steps and entire approaches for transforming data. Solutions that demand more technical interactions often suffer from an intolerance of ambiguity, which limits your ability to express intent without significant experience and/or training. See *Transformer Page*.

## Selections

As you review the visualization, a change in the cursor indicates the items that are available for selection.



**Figure: Selection cursor changes on hover of selectable items**

The following types of selections trigger the subsequent phases:

- cell values and values within a cell

- columns

Selecting a single column in the data grid triggers a visual profile of the column data, as well as a set of suggestions. Selecting multiple columns triggers a different set of suggestions to apply across your selected columns.

- values in a data histogram
- categories of values (valid, invalid, missing) within a data quality bar

Columns and values can be multi-selected.

You are still obligated to make selections in the data, thereby bringing domain-specific expertise to the problem of transforming it. This selection in turn triggers a more complex query through the application to the prediction service.

## Predictive Model

Based on the set of selections, an inference algorithm attempts to interpret the data transformation intent of the selection and generates a ranked set of suggestions and patterns for the selections to match. For example, if you select the first three characters in a cell, the algorithm may produce two transform suggestions for data removal: one to remove the rows containing the specific text and one to keep all rows containing that pattern of text in the column.

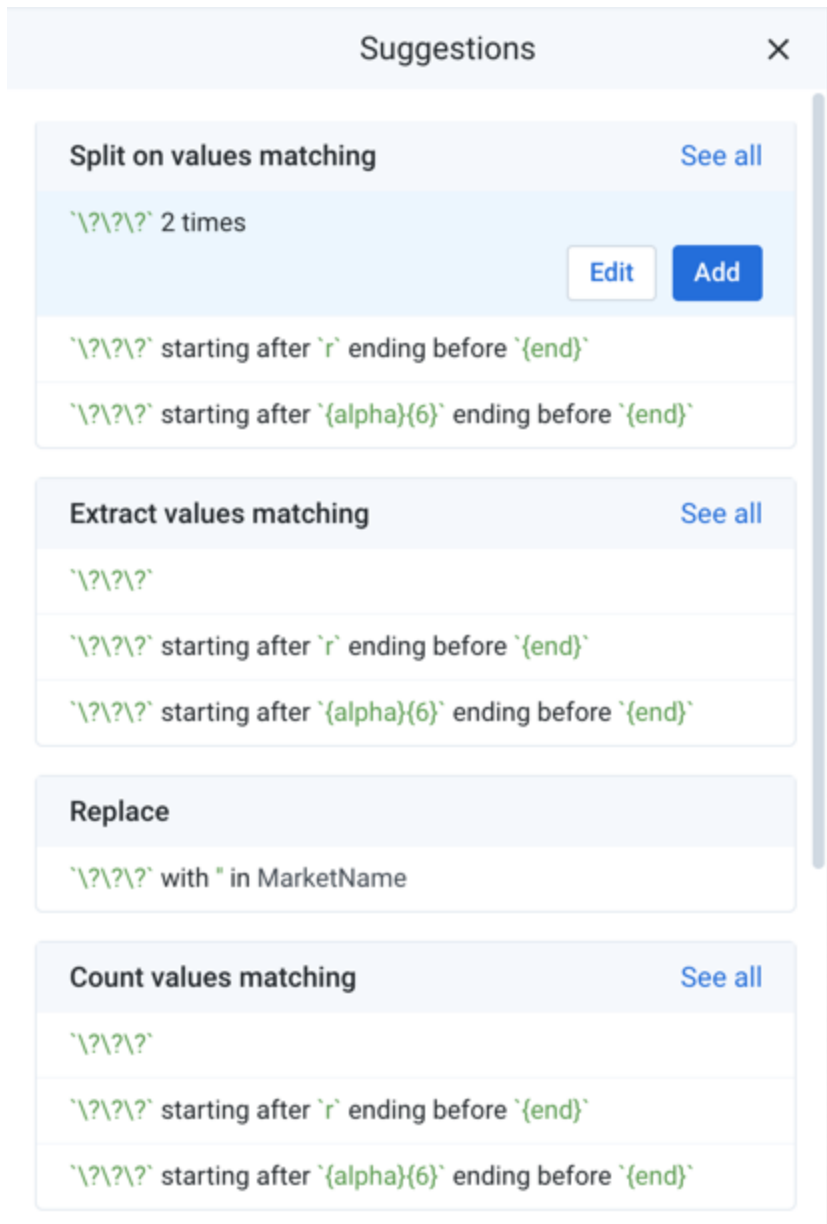
As part of the returned results of the predictive model, matching values for the selection(s) are highlighted in the table.

The predictive model interprets selection to identify intent. Possible intentions are surfaced as one or more suggested transforms in a visual manner that minimizes exposure to the transformation language.

## Suggestions and Their Variants

The set of probable next steps is computed by the predictive model from user interaction, selected data, historical information, and other sources and rendered as a set of suggestions. Since these steps are essentially predictions of user intent, they are surfaced as browsable cards, through which you can explore to disambiguate the uncertainty of intention around their data selections.





**Figure: Suggestion cards - selection guides suggestion**

**Notes:**

- Typically, pattern-based variants to the suggestion are listed first in the suggestion card.
- Pattern-based suggestions are always based on Patterns , which are easier to use than regular expressions.
- Variants using literal expressions are typically listed last. If a column has a high number of literal values, however, literal value variants may be listed first in the card.

Suggestion cards are specific enough for immediate execution. You can choose to modify the transform and its parameters, if additional specification and guidance is needed.

In a suggestion card, you may see multiple **variants** of the selected transformation.

The first variant is the most specific one applicable to the current selection in the data grid. Mouse over the variants to see different versions of the transform. As you mouse over secondary variants, the variants typically become more specific in their changes to the dataset or rarer in their usage.

When you mouse over a different transform variant in the suggestion card, the preview popup is automatically updated to reflect the variation. When you select the variant, the Preview pane is updated. You can always modify the transform to review the detailed differences.

## Collaborative suggestions

Optionally, you can enable the surfacing of collaborative suggestions, which aggregate the transformation steps from users in your workspace to provide an additional category of Recently used suggestion cards. As workspace members continue to transform data that is often related, the set of Recently used suggestions become more relevant to the data on which workspace users are working. This form of data-dependent Predictive Transformation allows Designer Cloud powered by Trifacta Enterprise Edition to improve its understanding of the types of tasks that workspace users are trying to accomplish.

**NOTE:** This feature requires the machine learning service, which is enabled by default. For more information, see *Miscellaneous Configuration*.

Workspace administrators can choose to enable this feature and can configure whether data is aggregated from individual workspace users' transformations or from all workspace users' transformations. See *Workspace Settings Page*.

When this feature is enabled, collaborative suggestions appear as cards under a new **Recently used** category in the suggestions panel.

When the feature is enabled, Individual users can choose to opt-out of sharing their data with the feature. See *User Profile Page*.

## Previews

When a suggestion card is selected, the results of the selected transform are previewed in the data grid, so that you can see in advance the changes to the dataset.

The screenshot displays the Trifacta Designer Cloud interface. On the left, a data grid shows a list of market names under the 'MarketName' column. A 'Preview' column is visible, showing the results of a transformation. On the right, a 'Suggestions' panel is open, displaying several suggestion cards. The first card is 'Split on values matching' with a 'See all' link. Below it are two cards for 'Extract values matching' and 'Replace', each with a 'See all' link. The 'Replace' card shows a transformation rule: '\[?]\?' with 'in MarketName'. At the bottom of the suggestions panel, there is a 'Count values matching' card with a 'See all' link. The bottom of the interface shows a status bar with '13 Columns', '38 Rows', and '6 Data Types'.

**Figure: Previewed effects of transform**

When the transform is added to the recipe, the transform is rendered into the data transformation language and applied in real-time to the dataset, so that you can immediately begin working on the next step of the process.

When a transform is selected, the selected transform and any additional guidance that you provide is translated into a specific, programmatic step in the transformation language. This step, in turn, is rendered into a complex and potentially distributed query that is applied across the dataset. In this manner, additional technical details and the knowledge required to master them are removed from user requirements.

## Additional Steps - Modification

### Modification via Transform Builder

As needed, any selection can be modified, such that you may tweak parameters to further refine intention to reach a specific outcome. In Designer Cloud powered by Trifacta Enterprise Edition, you can click **Edit** to tweak individual transformations in the Transform Builder.

The screenshot shows the Transform Builder interface. On the left, there's a table with two columns: 'Source' and 'Preview'. The 'Source' column contains a list of market names, some of which are highlighted in blue. The 'Preview' column shows the same list of market names, but with some text modified. A modal window titled 'Replace text or patterns' is open on the right. It has a 'Find' field containing the text '222Y' and a 'Replace with' field containing the text 'Y'. There are also 'Advanced options' and 'Cancel'/'Add' buttons.

**Figure: Modifying a transform in the Transform Builder**

## Wrangle

The actual steps of transformation are authored in Wrangle (a domain-specific language for data transformation). Wrangle includes the following characteristics:

- Single-source transformations, with results rendered without modification to the original source data
- General cleaning and transformation operations on numerical and textual data of varying data types
- Structural transformations for managing nested data like JSON
- Multi-dataset transformations such as lookups, joins, and unions
- Transformation of data to metadata, such as pivot and unpivot operations
- Text selection patterns, including regular expressions, as a macro-type set of references. See *Text Matching*.

For a list of available transforms and functions, see *Language Index*.

For more information, see *Wrangle Language*.

# Overview of the Type System

## Contents:

- *How Data Types Are Used*
  - *Data Types*
    - *Logical data types*
    - *Complex data types*
    - *Logical and complex data types*
    - *Custom data types*
  - *Types in Source Data*
    - *Schematized files*
    - *Schematized tables*
    - *Inferred data types*
  - *Type Inference*
    - *Type inference in the application*
  - *Working with Data Types*
    - *Data types in the grid*
    - *Changing the data type*
    - *Changing the data format*
    - *Type functions*
  - *Type Conversions on Export*
    - *Type Conversions*
- 

This section provides an overview of how data types are managed during import, transformation, and export of data in Designer Cloud powered by Trifacta® Enterprise Edition.

## Terms:

A **data type** is a definition of a set of values, including:

- Possible set of values (e.g. Dates from 1/1/1400 to 12/31/2599)
- Meaning of those values (e.g. two-letter codes can represent states of the United States)
- Set of operations that can be applied to those values (e.g. functions applicable to integers)

A **data format** is a representation of the underlying type, which has the same meaning and available operations associated with the data type. For example, the following values are all valid for Datetime data type, but each is represented in a different format:

```
12/31/2021
31-12-2021
December 31, 2021
```

**NOTE:** Some data types can be explicitly formatted through functions. Other data types support different formats implicitly through the application.

## How Data Types Are Used

In the Designer Cloud application , data types are used for the following:

- Anomaly detection (Is the value valid or invalid?)

- Suggestions (What are the available transformation suggestions for this column based on its data type?)
- Standardization (How can all of these valid values be standardized for the column's data type?)
- Pattern recognition (How to identify different formats in the same column?)

## Data Types

Designer Cloud powered by Trifacta Enterprise Edition supports the following categories of data types:

### Logical data types

A **logical** data type is a class of values that is understood by native system representations.

**Tip:** These types are recognized internally by Designer Cloud powered by Trifacta Enterprise Edition. Each running environment to which Designer Cloud powered by Trifacta Enterprise Edition connections natively supports these logical data types.

These data types have no additional specification requirements:

| Data Type                 | Description                                                                                                                                                                                                                                                                                    |
|---------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>String Data Type</i>   | Any non-null value can be typed as String. A String can be anything.                                                                                                                                                                                                                           |
| <i>Integer Data Type</i>  | The Integer data type applies to positive and negative numeric values that have no decimal point.                                                                                                                                                                                              |
| <i>Decimal Data Type</i>  | Decimal data type applies to floating points up to 15 digits in length. <ul style="list-style-type: none"> <li>• In the Designer Cloud application , this data type is referenced as <code>Decimal</code>.</li> <li>• In storage, this data type is written as <code>Double</code>.</li> </ul> |
| <i>Boolean Data Type</i>  | The Boolean data type expresses true or false values.                                                                                                                                                                                                                                          |
| <i>Datetime Data Type</i> | Designer Cloud powered by Trifacta® Enterprise Edition supports a variety of Datetime formats, each of which has additional variations to it.                                                                                                                                                  |
| <i>Object Data Type</i>   | An <b>Object</b> data type is a method for encoding key-value pairs. A single field value may contain one or more sets of key-value pairs.                                                                                                                                                     |
| <i>Array Data Type</i>    | An <b>array</b> is a list of values grouped into a single value. An array may be of variable length; in one record the array field may contain two elements, while in the next record, it contains six elements.                                                                               |

Formatting differences may apply. For example, Designer Cloud powered by Trifacta Enterprise Edition may recognize `Y` and `N` as Boolean data type, while other systems may not.

### Complex data types

A **complex** data type typically is defined by applying additional restrictions on String data type values to define the class of possible values. For example, Designer Cloud powered by Trifacta Enterprise Edition supports a Gender data type, which validates values such as `M` & `F` and `male` and `female` as Gender data type.

The following are the complex data types supported by Designer Cloud powered by Trifacta Enterprise Edition.

| Data Type                               | Description                                                                                                                                            |
|-----------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Social Security Number Data Type</i> | This data type is applied to numeric data following the pattern for United States Social Security numbers.                                             |
| <i>Phone Number Data Type</i>           | This data type is applied to numeric data following common patterns that express telephone numbers and known valid phone numbers in the United States. |

|                                |                                                                                                                              |
|--------------------------------|------------------------------------------------------------------------------------------------------------------------------|
| <i>Email Address Data Type</i> | This data type matches text values that are properly formatted email addresses.                                              |
| <i>Credit Card Data Type</i>   | Credit card numbers are numeric data that follow the 14-digit or 16-digit patterns for credit cards.                         |
| <i>Gender Data Type</i>        | This data type matches a variety of text patterns for expressing male/female distinctions.                                   |
| <i>Zip Code Data Type</i>      | This data type matches five- and nine-digit U.S. zipcode patterns.                                                           |
| <i>State Data Type</i>         | State data type is applied to data that uses the full names or the two-letter abbreviations for states in the United States. |
| <i>IP Address Data Type</i>    | The IP Address data type supports IPv4 address.                                                                              |
| <i>URL Data Type</i>           | URL data type is applied to data that follows generalized patterns of URLs.                                                  |
| <i>HTTP Code Data Type</i>     | Values of these data types are three-digit numeric values, which correspond to recognized HTTP Status Codes.                 |

Complex data types are typically defined based on a regular expression applied to String values. For example, the following regex is used to define the accepted values for the Email Address data type:

```
"regexes": [
 "^[a-z0-9.!#$%&'*/+=?^_`{|}~-]+@[a-z0-9.-]+(?:\\.[a-z0-9-]+)*\\.([a-z]{2,})$"
],
```

**NOTE:** Regex-based data types can be modified, although in most cases, it is unnecessary. Regular expressions are considered a developer-level configuration. For more information, please contact *Alteryx Customer Success and Services*.

## Logical and complex data types

| Data type                               | Category | Internal data type | Notes                                                                                      |
|-----------------------------------------|----------|--------------------|--------------------------------------------------------------------------------------------|
| <i>String Data Type</i>                 | logical  | String             | The default data type. Any non-empty/non-value is valid for String data type.              |
| <i>Integer Data Type</i>                | logical  | Int                | Use <i>NUMFORMAT Function</i> to format these values. Underlying values are not modified.  |
| <i>Decimal Data Type</i>                | logical  | Float              | Use <i>NUMFORMAT Function</i> to format these values. Underlying values are not modified.  |
| <i>Boolean Data Type</i>                | logical  | Bool               |                                                                                            |
| <i>Datetime Data Type</i>               | logical  | Datetime           | Use <i>DATEFORMAT Function</i> to format these values. Underlying values are not modified. |
| <i>Object Data Type</i>                 | logical  | Map/Object         |                                                                                            |
| <i>Array Data Type</i>                  | logical  | Array              |                                                                                            |
| <i>Social Security Number Data Type</i> | complex  | String             | String data type constrained by a regular expression.                                      |
| <i>Phone Number Data Type</i>           | complex  | String             | String data type constrained by a regular expression.                                      |
| <i>Email Address Data Type</i>          | complex  | String             | String data type constrained by a regular expression.                                      |
| <i>Credit Card Data Type</i>            | complex  | String             | String data type constrained by a regular expression.                                      |
| <i>Gender Data Type</i>                 | complex  | String             | String data type constrained by a regular expression.                                      |
| <i>Zip Code Data Type</i>               | complex  | String             | String data type constrained by a regular expression.                                      |
| <i>State Data Type</i>                  | complex  | String             | String data type constrained by a regular expression.                                      |

|                             |         |        |                                                       |
|-----------------------------|---------|--------|-------------------------------------------------------|
| <i>IP Address Data Type</i> | complex | String | String data type constrained by a regular expression. |
| <i>URL Data Type</i>        | complex | String | String data type constrained by a regular expression. |
| <i>HTTP Code Data Type</i>  | complex | String | String data type constrained by a regular expression. |

## Custom data types

If needed, you can create your own data types, which can be selected, validated, and profiled in the Designer Cloud application .

**NOTE:** Custom data types must be defined using regular expressions, which are considered a developer-level configuration. For more information, please contact *Alteryx Customer Success and Services* .

For more information, *Create Custom Data Types*.

## Types in Source Data

Depending on the source system, imported data may be typed to Trifacta data types according to one of the following methods.

**Tip:** For each method, Designer Cloud powered by Trifacta Enterprise Edition attempts to map the source data to one of the above data types. For schematized sources, however, you may wish to use the original data types in the source. Optionally, you can choose to disable the mapping of source to internal data type. See "Type Inference" below.

## Schematized files

Some file formats, such as Avro or Parquet, are stored in a non-readable format. Part of the metadata associated with the file is information identifying the schema of the file. A **schema** represents the data types and other constraints of individual columns. It can be read independently of the data in the source table.

## Schematized tables

For relational sources, schema information is typically stored with the table. This schema information defines data type validation within the datastore and can be read independently from the source table.

**Tip:** Database schemas can be used to define a class of tables to ensure consistency within a database.

## Inferred data types

In most cases, an imported data source is assigned a data type for each column based on a review of a subset of the data. For example, a CSV file contains no information about the data types of individual columns. The data types for each column must be assigned by Designer Cloud powered by Trifacta Enterprise Edition. This process is called type inference. For more information, see "Type Inference" below.

## Type Inference

By default, the Designer Cloud application applies type inference for imported data. The application attempts to infer a column's appropriate data type in the application based on a review of the first lines in the sample.



**Tip:** In many programming languages, a column must be explicitly "type cast" to a data type as part of a functional operation. Wrangle handles this typecasting for you through the process of type inference.

**NOTE:** Mapping source data types to Trifacta data types depends on a sufficient number of values that match the criteria of the internal data type. The mapping of import types to internal data types depends on the data.

- Type inference needs a minimum of 25 rows of data in a column to work consistently.
- If your dataset has fewer than 20 rows, type inference may not have sufficient data to properly infer the column type.

In some datasets, the first 25 rows may be of a data type that is a subset of the best matching type. For example, if the first 25 rows in the initial same match the Integer data type, the column may be typed as Integer, even if the other 2,000 rows match for the Decimal data type. If the column data type is unmodified:

- The data is written out from Designer Cloud powered by Trifacta Enterprise Edition as Integer data type. This works for the first 25 rows.
- The other 2,000 rows are written out as null values, since they do not match the Integer data type. If the source data used decimal notation (e.g. 3.0 in the source), then those values are written out as null values, too.

In this case, it may be easier to disable type inference for this dataset.

**Tip:** If you are having trouble getting your imported dataset to map to expected data types, you can disable type inference for the individual dataset. For more information, see *Import Data Page*.

After data is imported, the Designer Cloud application provides some mechanisms for applying stronger typecasting to the data. Example:

- If all input values are double-quoted, then Designer Cloud powered by Trifacta Enterprise Edition evaluate all columns as String type. As a result, type inference cannot be applied.
- Since non-String data types cannot be inferred, then the first row cannot be detected as anomalous against the inferred type (String). Column headers cannot be automatically detected from double-quoted source files.

**Tip:** The default data type is String. If the Designer Cloud application is unable to evaluate a column's data type, the type is mapped to String data type. Within the application, you can use functions to remap the data type or to parse values according to a specified type.

For more information, see "Working with Data Types" below.

## Disable type inference

For schematized files or tables, the Designer Cloud application inference of data type from the source may result in incorrect initial typing of a dataset's columns in the application. As needed, you can disable type inference for the following:

**NOTE:** When type inference is disabled for imported datasets, it is not disabled within the Designer Cloud application. For more information, see "Type inference in the application" below.



- **Disable for individual files:** In the Import Data page, select the file. In the right-hand column, click **Edit Settings**. For more information, see *File Import Settings*.
- **Disable for individual tables:** In the Import Data page, select the table. In the right-hand column, click **Edit Settings**. For more information, see *Relational Table Settings*.
- **Disable for individual connections:** In the Connections page, edit the connection. In the Edit Connection window, select `Disabled` under Default Column Data Type Inference. By default, all datasets through this connection have type inference disabled. For more information, see *Create Connection Window*.
- **Disable globally:** If desired, you can disable type inference for all users of the workspace. For more information, see *Disable Type Inference*.

## Type inference in the application

Within the Designer Cloud application, column data types may be re-inferred based on your activities in the Transformer page:

**NOTE:** Disabling type inference does not disable the re-inference of types in the Transformer page.

The following general actions may result in column data types being re-inferred:

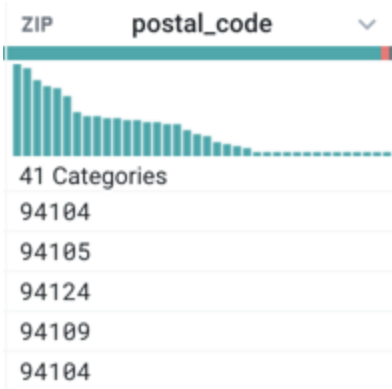
- After a sample is taken, column data types are inferred based on the first set of rows in the sample.
- If a transform or function is provided with a data type that does not match the expected input data type, the values are typecast to the expected output, so you may see changes to the data type of the output to better align with the function.
- Multi-dataset operations generally do not cause re-inferring of data types. However, if there is a mismatch of data types between two columns in a union operation, for example, the data type of the first dataset is preferred.

## Working with Data Types

After data has been imported, you can remap individual column types through recipe steps. For more information, see *Change Column Data Type*.

### Data types in the grid

When a sample is loaded, the data types and their formats for each column are inferred by default. Data types and formatting information is displayed for each column in the Transformer page.



**Figure: Column header example**

At the top of each column, you can review graphical representations of type information:

- **Data type indicator:** To the left of the column name, you can see a graphic of the data type. In the above, the data type is set to Zip code.

**Tip:** Select the data type indicator to change the column to a different data type. This change is added as a step in your recipe. See "Changing the data type."

- **Data quality bar:** Below the column name, you can see a bar indicating the relative percentage of valid, invalid and empty values in the column, compared to the listed data type.
  - Green: Valid for the data type
  - Red: Invalid for the data type
  - Gray: empty or null

**Tip:** Select one of the colored bars to be prompted by a set of transformation suggestions that can be applied to the selected set of values.

- **Column histogram:** Below the data quality bar, you can see the distribution of values within the column. The column histogram may represent the data in different ways, depending on the column's data type.

**Tip:** Click or SHIFT-click values in the histogram to be prompted for transformation suggestions that can be applied to the selected values.

For more information, see *Data Grid Panel*.

For more information, see *Column Menus*.

## Changing the data type

Change the data type through the data type menu at the top of a column.

**Tip:** For some types, such as Datetime type, you must select a data format when you are selecting the type. See below.

**NOTE:** Changing the data type may not change the underlying logical type. For example, if you change a String column to a Gender column, the underlying data is still stored as String values.

See *Change Column Data Type*.

## Changing type across multiple columns

To change the data type for multiple columns, you can a transformation similar to the following, which changes the data type from the `reqProdId` column to the `prodC` column and all columns in between:

|                        |                         |
|------------------------|-------------------------|
| Transformation Name    | Change column data type |
| Parameter: Columns     | Range                   |
| Parameter: Column list | reqProdId~prodC         |
| Parameter: New type    | String                  |

## Changing the data format

You can use the following functions to apply formatting on top of a column of a specified data type. For example, depending on your locale, numbers may require different formatting for use of the decimal point and the digit separator.

**NOTE:** When you apply a formatting function to a column, the data appears in the specified format in the Designer Cloud application , but the underlying data is unmodified. Formatting changes appear as a step in your recipe and are applied to the generated results.

| Formatting Function            | Applicable Data Type | Description                                                                                                                                                                                                                         |
|--------------------------------|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>NUMFORMAT Function</i>      | Integer, Decimal     | Formats a numeric set of values according to the specified number formatting. Source values can be a literal numeric value, a function returning a numeric value, or reference to a column containing an Integer or Decimal values. |
| <i>DATEFORMAT Function</i>     | Datetime             | Formats a specified Datetime set of values according to the specified date format. Source values can be a reference to a column containing Datetime values.                                                                         |
| <i>UNIXTIMEFORMAT Function</i> | Datetime             | Formats a set of Unix timestamps according to a specified date formatting string.                                                                                                                                                   |

## Type functions

The Designer Cloud application provides a set of functions for managing types.

### Validation functions

These functions can be used to test for valid or invalid values against a specific data type.

| Function                  | Description                                                                                                                                                                                                    |
|---------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>VALID Function</i>     | Tests whether a set of values is valid for a specified data type and is not a null value.                                                                                                                      |
| <i>ISINVALID Function</i> | Tests whether a set of values is not valid for a specified data type.                                                                                                                                          |
| <i>IFINVALID Function</i> | The IFINVALID function writes out a specified value if the input expression matches the specified data type. Otherwise, it writes the source value. Input can be a literal, a column reference, or a function. |
|                           |                                                                                                                                                                                                                |

|                                                      |                                                                                                                                                                                                                                          |
|------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>IFMISMATCHED</i><br><i>HED</i><br><i>Function</i> | The IFMISMATCHED function writes out a specified value if the input expression does not match the specified data type or typing array. Otherwise, it writes the source value. Input can be a literal, a column reference, or a function. |
|------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Parsing functions

These functions can be used to parse String values against a specific data type.

| Function                                | Description                                                                                                                                                                                                            |
|-----------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>PARSEINT</i><br><i>Function</i>      | Evaluates a String input against the Integer datatype. If the input matches, the function outputs an Integer value. Input can be a literal, a column of values, or a function returning String values.                 |
| <i>PARSEFLOAT</i><br><i>Function</i>    | Evaluates a String input against the Decimal datatype. If the input matches, the function outputs a Decimal value. Input can be a literal, a column of values, or a function returning String values.                  |
| <i>PARSEBOOL</i><br><i>Function</i>     | Evaluates a String input against the Boolean datatype. If the input matches, the function outputs a Boolean value. Input can be a literal, a column of values, or a function returning String values.                  |
| <i>PARSEDATETIME</i><br><i>Function</i> | Evaluates an input against the default input formats or (if specified) an array of Datetime format strings in their listed order. If the input matches one of the formats, the function outputs a Datetime value.      |
| <i>PARSEARRAY</i><br><i>Function</i>    | Evaluates a String input against the Array datatype. If the input matches, the function outputs an Array value. Input can be a literal, a column of values, or a function returning String values.                     |
| <i>PARSEOBJECT</i><br><i>Function</i>   | Evaluates a String input against the Object datatype. If the input matches, the function outputs an Object value. Input can be a literal, a column of values, or a function returning String values.                   |
| <i>PARSESTRING</i><br><i>Function</i>   | Evaluates an input against the String datatype. If the input matches, the function outputs a String value. Input can be a literal, a column of values, or a function returning values. Values can be of any data type. |

## Managing null and empty values

These functions allow you to generate or test for missing or null values.

| Function                            | Description                                                                                                                                                                                           |
|-------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>ISNULL</i><br><i>Function</i>    | The ISNULL function tests whether a column of values contains null values. For input column references, this function returns true or false.                                                          |
| <i>NULL</i><br><i>Function</i>      | The NULL function generates null values.                                                                                                                                                              |
| <i>IFNULL</i><br><i>Function</i>    | The IFNULL function writes out a specified value if the source value is a null. Otherwise, it writes the source value. Input can be a literal, a column reference, or a function.                     |
| <i>ISMISSING</i><br><i>Function</i> | The ISMISSING function tests whether a column of values is missing or null. For input column references, this function returns true or false.                                                         |
| <i>IFMISSING</i><br><i>Function</i> | The IFMISSING function writes out a specified value if the source value is a null or missing value. Otherwise, it writes the source value. Input can be a literal, a column reference, or a function. |

## Type Conversions on Export

The Designer Cloud application attempts to map the data types that you have specified to match data types in the target platform.

**NOTE:** Values that do not match the data type of the target system for a column are subject to the method by which the target system handles mismatches. Rows could be dropped. Values can be rendered as null values. You should attempt to verify that all columns have valid values before generating results.

**NOTE:** Missing or null values may be treated differently between target systems. Additionally, if these systems feed downstream systems, those systems may have independent rules for managing missing or null values.

## Type Conversions

| Item                                    | Description |
|-----------------------------------------|-------------|
| Avro Data Type Conversions              |             |
| DB2 Data Type Conversions               |             |
| Hive Data Type Conversions              |             |
| Oracle Data Type Conversions            |             |
| MySQL Data Type Conversions             |             |
| Parquet Data Type Conversions           |             |
| Postgres Data Type Conversions          |             |
| Redshift Data Type Conversions          |             |
| Snowflake Data Type Conversions         |             |
| AWS Glue Data Type Conversions          |             |
| Salesforce Data Type Conversions        |             |
| SQL Server Data Type Conversions        |             |
| SQL DW Data Type Conversions            |             |
| Databricks Tables Data Type Conversions |             |
| Tableau Hyper Data Type Conversions     |             |
| Teradata Data Type Conversions          |             |
| SharePoint Data Type Conversions        |             |

For more information, see *Type Conversions*.

# Overview of Standardization

## Contents:

- *Standardization Methods*
- *Invalid Values*
- *Standardize Values by Clustering*
- *Standardize Formatting by Patterns*
- *Standardize Using Functions*
  - *Functions for strings*
  - *Functions for numbers*
  - *Functions for dates*
- *Custom Data Types*
  - *Custom type using a dictionary file*
  - *Custom type using a regular expression*

Designer Cloud powered by Trifacta® Enterprise Edition provides multiple mechanisms for reviewing your column values and identifying patterns in the data format or similar values which mean the same thing. Through simple visual tools, you can select the patterns or clustered value to standardize and, when prompted, the patterns or values to use as their standard. As needed, you can apply formatting or structuring functions to the data for finer grain controls. This section summarizes the available methods of standardization, as well as their recommended uses.

## Standardization Methods

You can use any of the following methods for standardizing values in your dataset's columns. Depending on the situation, you may choose to mix-and-match these methods. Details on these methods are below.

| Method        | Description                                                                                                                                                                                                                   | Recommended Uses                                                                                                                                                                                                                                                         | How to Use                                                   |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------|
| By clustering | Designer Cloud powered by Trifacta Enterprise Edition can identify similar values using one of the available algorithms for comparing values. You can compare values based on spelling or language-independent pronunciation. | <ul style="list-style-type: none"><li>• Standardize values to correct spelling differences, capitalization, whitespace, and other errors.</li><li>• Values must be consistent across rows of the column.</li><li>• Primarily used for string-based data types.</li></ul> | Available through the <i>Standardize Page</i>                |
| By pattern    | Designer Cloud powered by Trifacta Enterprise Edition can identify common patterns in a set of values and suggest transformations to standardize the values to a common format.                                               | <ul style="list-style-type: none"><li>• Standardize values to follow a consistent format, such as phone numbers or social security numbers.</li><li>• Data type follows a somewhat consistent format and needs reshaping.</li></ul>                                      | Available in the Patterns tab in <i>Column Details Panel</i> |
| By function   | You can apply one or more specific functions to cleanse your data of minor errors in formatting or structure.                                                                                                                 | <ul style="list-style-type: none"><li>• Good method for improving the performance of pattern- or algorithm-based matching.</li><li>• Some functions are specific to a data type, while others have more general application.</li></ul>                                   | Edit column with formula in the <i>Transform Builder</i>     |
| Mix-and-match | You can use combinations of the above methods for more complex use cases.                                                                                                                                                     | <ul style="list-style-type: none"><li>• Combine function-based standardization for global changes to</li></ul>                                                                                                                                                           |                                                              |

|  |  |                                                                                         |  |
|--|--|-----------------------------------------------------------------------------------------|--|
|  |  | all values with cluster- or pattern-based standardization for individual value changes. |  |
|--|--|-----------------------------------------------------------------------------------------|--|

## Invalid Values

These standardization techniques assume that your column contains only valid or empty values.

**Tip:** Standardization may help to cut down the number of invalid values. Before you begin standardizing, however, you should select the red bar in the column histogram to review the values that are invalid for the current type and to fix them via suggestion if possible. For more information, see *Find Bad Data*.

## Standardize Values by Clustering

Using one of the supported matching algorithms, Designer Cloud powered by Trifacta Enterprise Edition can cluster together similar column values. You can review the clusters of values to determine if they should be mapped to the same value. If so, you can apply the mapping of these values within the application.

For more information, see *Overview of Cluster Clean*.

## Standardize Formatting by Patterns

For individual columns, Designer Cloud powered by Trifacta Enterprise Edition can analyze column values for patterns and then provide suggestions for how to normalize the patterned values into a consistent format. For example, the same US phone number can be represented in any of the following methods:

```
555-1212
415-555-1212
4155551212
(415) 555-1212
+1 (415) 555-1212
```

**Tip:** Pattern-based standardization is useful for confirming values in a column to a specific format. This method is applicable to data types like phone numbers, dates, social security numbers, and to a lesser extend email addresses and URLs.

You can apply pattern-based standardization through the Patterns tab. See *Column Details Panel*.

## Standardize Using Functions

The following functions can be useful for standardizing values.

### Functions for strings

All values can be converted to string, so these string functions can be applied to any column if its data type is converted to String data type.

**Tip:** The clustering algorithms may apply some of these functions to values in your column for purposes of comparison.

| Category          | Function      | Description                                                                 |
|-------------------|---------------|-----------------------------------------------------------------------------|
| String Conversion | CHAR Function | Generates the Unicode character corresponding to an inputted Integer value. |

|                                    |                                  |                                                                                                                                                                |
|------------------------------------|----------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                    | <i>UNICODE Function</i>          | Generates the Unicode index value for the first character of the input string.                                                                                 |
| <b>Case Conversion</b>             | <i>UPPER Function</i>            | All alphabetical characters in the input value are converted to uppercase in the output value.                                                                 |
|                                    | <i>LOWER Function</i>            | All alphabetical characters in the input value are converted to lowercase in the output value.                                                                 |
|                                    | <i>PROPER Function</i>           | Converts an input string to propercase. Input can be a column reference or a string literal.                                                                   |
| <b>Cleanse Functions</b>           | <i>TRIM Function</i>             | Removes leading and trailing whitespace from a string. Spacing between words is not removed.                                                                   |
|                                    | <i>TRIMQUOTES Function</i>       | Removes leading and trailing quotes or double-quotes from a string. Quote marks in the middle of the string are not removed.                                   |
|                                    | <i>REMOVEWHITESPACE Function</i> | Removes all whitespace from a string, including leading and trailing whitespace and all whitespace within the string.                                          |
|                                    | <i>REMOVESYMBOLS Function</i>    | Removes all characters from a string that are not letters, numbers, accented Latin characters, or whitespace.                                                  |
| <b>String Sizing Functions</b>     | <i>LEFT Function</i>             | Matches the leftmost set of characters in a string, as specified by parameter. The string can be specified as a column reference or a string literal.          |
|                                    | <i>RIGHT Function</i>            | Matches the right set of characters in a string, as specified by parameter. The string can be specified as a column reference or a string literal.             |
|                                    | <i>PAD Function</i>              | Pads string values to be a specified minimum length by adding a designated character to the left or right end of the string. Returned value is of String type. |
| <b>String Comparison Functions</b> |                                  | See <i>Compare Strings</i> .                                                                                                                                   |

### Example:

Designer Cloud powered by Trifacta Enterprise Edition supports nesting functions within each other. The following transformation performs some basic cleanup on all columns in your dataset that are of String type.

|                            |                                                        |
|----------------------------|--------------------------------------------------------|
| <b>Transformation Name</b> | Edit column with formula                               |
| <b>Parameter: Columns</b>  | *                                                      |
| <b>Parameter: Formula</b>  | IFVALID(\$col, 'String', LEFT(UPPER(TRIM(\$col)), 32)) |

- The Columns value is a wildcard, which in this case applies the transformation across all columns in the dataset (\*).
- In the Formula, you see a nested expression. If the value in the column is valid against String data type, then, do the following to the column value:

**NOTE:** The IFVALID function tests each row value for validation against the specified data type. It does not test the column against the data type. See *IFVALID Function*.

- The TRIM function removes leading and trailing whitespace, which may register as a difference between values. See *TRIM Function*.
- The UPPER function then converts the output of the TRIM function to all uppercase. So, differences in capitalization are eliminated. See *UPPER Function*.
- The LEFT function truncates the output of the UPPER function to a maximum of 32 characters. See *LEFT Function*.

The net result of this single step applied to all columns is to eliminate whitespace, convert to uppercase, and then truncate the length of each string to only 32 characters.



For more information, see *Cleanse Tasks*.

## Functions for numbers

You can use the following functions to standardize numeric values.

| Function                     | Description                                                                                                                                                                                                                         |
|------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>ABS Function</i>          | Computes the absolute value of a given numeric value. The value can be a Decimal or Integer literal or a reference to a column containing numeric values.                                                                           |
| <i>ROUND Function</i>        | Rounds input value to the nearest integer. Input can be an Integer, a Decimal, a column reference, or an expression. Optional second argument can be used to specify the number of digits to which to round.                        |
| <i>TRUNC Function</i>        | Removes all digits to the right of the decimal point for any value. Optionally, you can specify the number of digits to which to round. Input can be an Integer, a Decimal, a column reference, or an expression.                   |
| <i>NUMFORMAT AT Function</i> | Formats a numeric set of values according to the specified number formatting. Source values can be a literal numeric value, a function returning a numeric value, or reference to a column containing an Integer or Decimal values. |

### Example:

For the NUMFORMAT function, you can specify the full format to which you want the numeric values in the column to conform. In the following example, all values that contain a decimal point and match with the Decimal (Float) type are forced to add a value before the decimal. This step converts values like .00 to 0.00, which standardizes the format of your numbers.

|                            |                                                                                                            |
|----------------------------|------------------------------------------------------------------------------------------------------------|
| <b>Transformation Name</b> | Edit column with formula                                                                                   |
| <b>Parameter: Columns</b>  | *                                                                                                          |
| <b>Parameter: Formula</b>  | <code>IF(FIND(\$col, '.')&gt;0, IFVALID(\$col, 'Float', NUMFORMAT(ROUND(\$col, 2), '0.00')), \$col)</code> |

- The Columns value is a wildcard, which in this case applies the transformation across all columns in the dataset (\*).
- In the Formula, you see a nested expression, which is a bit more complicated than the preceding String example.
  - The outer IF function tests if the FIND function returns a non-zero value when searching each column value for the period (.). Values that match could possibly be decimals and require further evaluation:
  - If the value in the column is valid against the Decimal (Float) data type then do the following:
    - ROUND the value to two decimal points. For more information, see *ROUND Function*.
    - Format the value in the following manner:

0.00

- The above format includes the two decimal points to which you rounded, adding any extra zeros if they are not present in the input rounded value.
- Additionally, another zero is inserted in front of the decimal if it is missing in the output of the ROUND function.
- For more information on number formats, see *NUMFORMAT Function*.

For more information, see *Normalize Numeric Values*.

## Functions for dates

Since dates are structured patterns of string-based data, the best approach is to begin by using the Patterns tab in the Column Details panel. See below.

For more detailed modifications, you can specify formatting strings that are applied as part of the DATEFORMAT function to the dates in your column.

| Function                   | Description                                                                                                                                                 |
|----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>DATEFORMAT Function</i> | Formats a specified Datetime set of values according to the specified date format. Source values can be a reference to a column containing Datetime values. |

For more information including examples on the DATEFORMAT function, see *Format Dates*.

## Custom Data Types

You can create custom datatypes to use as a form of standardization. Values in a column that do not conform to the custom type are flagged as invalid and can be triaged accordingly.

**NOTE:** A custom data type does not inherently provide a means of standardizing the values. The values flagged as invalid must be converted to valid values or removed.

### Custom type using a dictionary file

You can upload a dictionary file containing the list of accepted values for the custom type.

**NOTE:** This method is likely to be superseded by dictionaries that can be applied through the Standardize page.

For more information, see *Create Custom Data Types*.

### Custom type using a regular expression

A custom data type can be created based on a user-defined regular expression.

**NOTE:** Regular expressions are powerful tools for creating matching patterns. They are considered developer tools.

For more information, see *Create Custom Data Types Using RegEx*.

# Overview of Cluster Clean

## Contents:

- *Example - Multiple methods of clustering*
  - *Clustering Algorithms*
    - *Similar strings*
    - *Pronunciation*
  - *Job Execution*
  - *Disable*
- 

Cluster clean enables users of Designer Cloud powered by Trifacta® Enterprise Edition to standardize values in a column by clustering similar values together. Using one of the supported matching algorithms, Designer Cloud powered by Trifacta Enterprise Edition can cluster together similar column values. You can review the clusters of values to determine if they should be mapped to the same value. If so, you can apply the mapping of these values within the application.

- For more information on how to apply cluster clean, see *Standardize Page*.
- For more information on other methods of standardization, see *Overview of Standardization*.

## Artifacts:

When a cluster clean step is added to your recipe, the number of individual changes can be many megabytes of data. Instead of storing these objects within the recipe definition, they are stored as a set of artifacts in the artifact storage database and referenced from the recipe.

- These artifacts exist outside the scope of the recipe file.
- These artifacts must be stored in a Trifacta database for the step to be editable and exportable.

**NOTE:** If the artifact storage service is disabled, this feature is unusable.

- When a flow is exported, an `artifact.data` file is included as part of the export. This file must be imported with the flow definition, or the cluster clean step in the imported flow is broken. For more information, see *Export Flow*.

## Example - Multiple methods of clustering

### Source:

The following dataset includes some values that could be standardized:

| RowId | Values |
|-------|--------|
| Row01 | Apple  |
| Row02 | pear   |
| Row03 | apple  |
| Row04 | pair   |
| Row05 | Äpple  |
| Row06 | pare   |

When you standardize using a spelling-based algorithm, the following values are clustered:

---

| Source Value | New Value          |
|--------------|--------------------|
|              |                    |
| Apple        |                    |
| apple        |                    |
| Äpple        |                    |
|              | Unclustered values |
| pear         |                    |
| pair         |                    |
| pare         |                    |

After you select the cluster of values at top, you can enter `apple`, in the right context panel to replace that cluster of values with a single string.

In the above, the unclustered values are dissimilar in spelling, but in English, they sound the same (homonyms). When you select the Pronunciation-based algorithm, these values are clustered:

| Source Value | New Value          |
|--------------|--------------------|
|              |                    |
| pear         |                    |
| pair         |                    |
| pare         |                    |
|              | Unclustered values |
| Apple        | apple              |
| apple        | apple              |
| Äpple        | apple              |

When you select the top values clustered by pronunciation, you can enter `pear` in the right context panel.

## Results:

The six source values have been reduced to two final values through two different methods of clustering. See below for more information on the clustering algorithms.

| Source Value | New Value |
|--------------|-----------|
|              |           |
| pear         | pear      |
| pair         | pear      |
| pare         | pear      |
|              |           |
| Apple        | apple     |
| apple        | apple     |
| Äpple        | apple     |

You can apply cluster-based standardization through the Standardize Page. See *Standardize Page*.

## Clustering Algorithms

The following algorithms for clustering values are supported.

### Similar strings

For comparing similar strings, the following methods can be applied:

#### Fingerprint

The fingerprint method compares values in the column by applying the following steps to the data before comparing and clustering:

**NOTE:** These steps are applied to an internal representation of the data. Your dataset and recipe are not changed by this comparison. Changes are only applied if you choose to modify the values and add the mapping.

1. Remove accents from characters, so that only ASCII characters remain.
2. Change all characters to lowercase.
3. Remove whitespace.
4. Split the string on punctuation, any remaining whitespace, and control characters. Remaining characters are assembled into groups called **tokens**.
5. Sort the tokens and remove any duplicates.
6. Join the tokens back together.
7. Compare all tokenized values in the column for purposes of clustering.

#### Fingerprint Ngram

This method follows the same steps as those listed above, except that tokens are broken up based on a specific (N) number of characters. By default, Designer Cloud powered by Trifacta Enterprise Edition uses 2-character tokens.

**Tip:** This method can provide higher fidelity matching, although there may be performance impacts on columns with a high number of unique values.

### Pronunciation

Values are clustered based on a language-independent pronunciation.

This method uses the double metaphone algorithm for string comparison. For more information, see *Compare Strings*.

### Job Execution

When a job is executed, clustering that has been applied through the data grid is applied to the full dataset. Implications:

- If you have auto-standardized values, the most common value that is applied during job execution is the value that appeared most frequently in the sample that was displayed when the cluster clean step was defined. The most common value is not redetermined based on the entire dataset.
- Values that were not part of the displayed sample may not be factored in the standardization process during job execution.

## Disable

This feature is enabled by default. To disable, please complete the following steps.

### Steps:

1. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`.  
For more information, see *Platform Configuration Methods*.
2. Locate the following setting and set it to `false`.

```
"feature.columnStandardization.enabled"
```

3. Save changes and restart the platform.

# Overview of Visual Profiling

## Contents:

- Overview
    - Uses
  - Example
  - Visual Profiling Interfaces
    - Data Grid
    - Column Details
    - Pattern Profiling
    - Job Details
  - Enable
  - Profiling Engine
    - Exact vs. Approximate Metrics in Visual Profiles
- 

## Overview

In Designer Cloud powered by Trifacta® Enterprise Edition, **visual profiling** provides real-time interactive visualizations of your dataset to assist in the discovery, cleansing, and transformation of your data. Visual representations are required for interpreting large volumes of data, and the platform's innovative profiling techniques visualize key statistical information in a dynamic, easy-to-consume format for faster transformation.

- At the individual column level, visual profiles provide interactive statistical information visualized in an appropriate manner for the data type. For example, columns of Zip Code data type can be represented on a geographical map of the United States.
- All visual profiles are interactive, so you can dig into the details of the data. Select one or more elements in a profile, and you can take immediate action on the data, either through steps you define or through transform recommendations provided by the platform.
- The Transformer page displays a set of recommended actions to take based on the values, rows, or columns that you select in the data grid. These recommendations are motivated by platform logic and prior usage information. For more information, see *Overview of Predictive Transformation*.

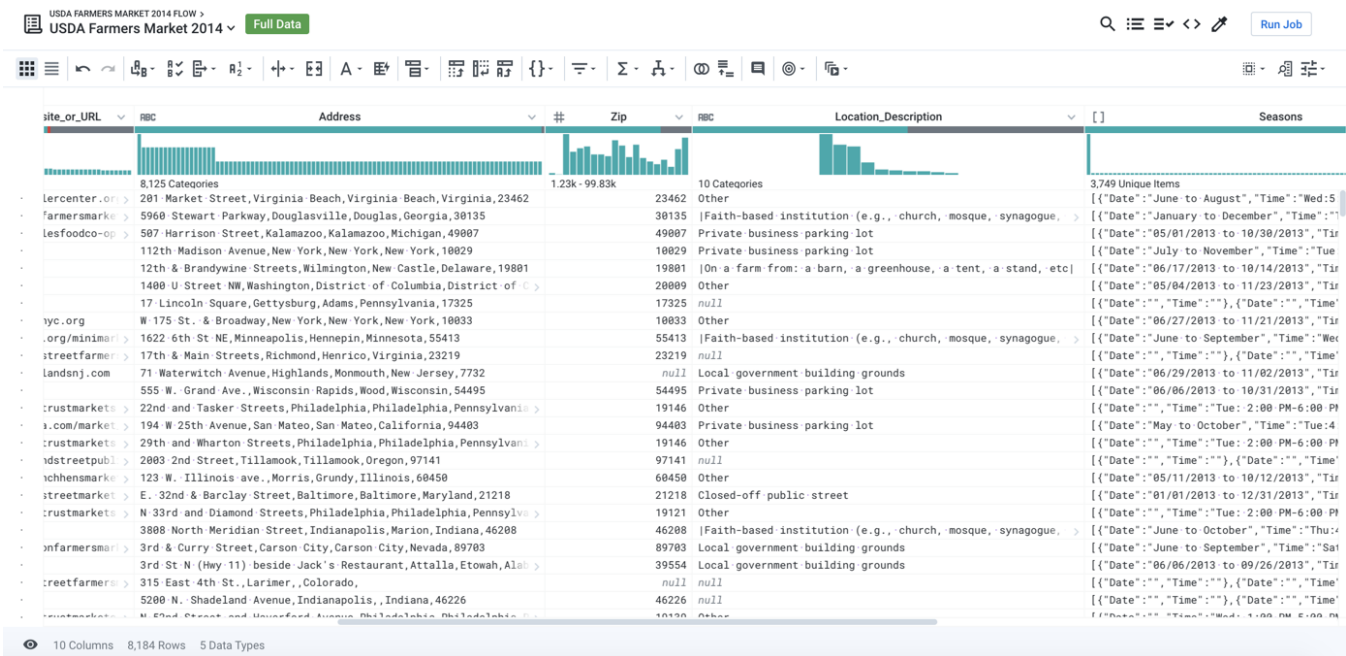
Visual profiles are available while you transform your data in the Transformer page, when you dig into the detail of individual columns, and after you execute your job at scale. Each of these interfaces has different usage patterns designed to accelerate and simplify data transformation for that specific area of the process.

## Uses

- **Locate anomalies.** Visual profiling surfaces missing or invalid data in individual columns. These values can then be selected and transformed as needed.
- **Identify distributions.** In the data grid, you can review value distribution for each column in your dataset. When exploring the column details, you can also identify and select statistical outliers among your column data.
- **Identify areas for further refinement.** After a job has completed, you can review its visual profile through the application and then take action on problematic data.

## Example

In the following example, a dataset containing address information has been loaded in the Transformer page:

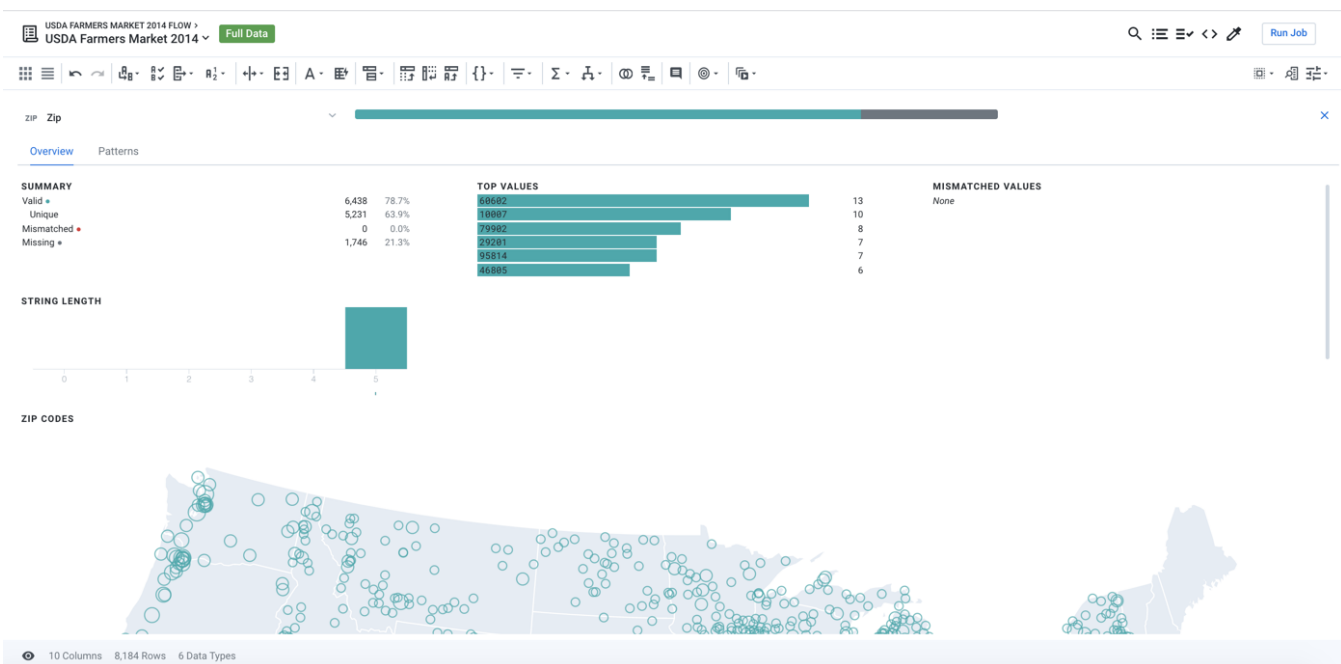


**Figure: Example dataset**

In this example, we are interested in exploring geographic information. From the column drop-down for the Zip column, you select **Column Details**.

**Explore detail on demand.** Generate visual profiles from the column drop-down.

When you explore the column details of the new column, you can see the following representation of the data:



**Figure: Zip Code data type represented as a U.S. map**



In this case, the values in your Zip column are recognized as being of Zipcode data type. The application then represents these values as a U.S. map, which quickly renders numeric data into a format that's much easier to read and analyze.

**Type-specific visualizations.** The profile of the column values is represented in a type-specific visualization to assist in rapid analyzing and taking action on some or all values in the column.

## Visual Profiling Interfaces

Wherever you can interact with data, visual profiling simplifies the process.

**Customized visualizations.** Each interface has been optimized for the scope of the data it is visualizing, whether the data is a single column, the entire sample of a dataset, or generated results.

## Data Grid

In the Transformer page, the **data grid** is a tabular representation of a sample of your dataset. It is the primary interface through which you build your transformation recipes. Profiling tools:

- **Data Quality Bar:** At the top of each column, you can see graphs counting the missing, invalid, and valid values for the column's current data type. Select one of the categories, and you can take immediate action on all of the category's values in the column.
- **Column Histogram:** Individual values in the column are represented in a histogram at the top of the column. You can select one or more of these values, review relevant data, and take action.
- See *Data Grid Panel*.

Whenever a transform is selected or specified, a preview of its effects is displayed in the data grid, including any changes to the data quality bar and column histogram of affected columns. See *Transform Preview*.

For additional details on visual transformation, see *Transform Basics*.

## Column Details

Through the Transformer page, you can explore statistical details about individual columns, visually represented based on the column's data type. From the drop-down for any column, select **Column Details**.

In this interface, you can review the range of values in the column and can optionally select one or more values from other columns to see which values in the current column apply. The visualizations for a column depend on the data type.

See *Column Details Panel*.

## Pattern Profiling

In the Column Details panel, you can review profiling of patterns detected in the values for the selected column. These patterns can be selected, which identifies the relevant values in the column that match the pattern. You can then use these selections as the basis for building transforms that apply to the matching values.

For more information, see *Column Details Panel*.

## Job Details

After the application has successfully executed a job for which profiling is enabled, you can explore a visualization of the generated dataset in the Job Details page. You can download your visual profile and results of your data quality rules on the entire dataset in PDF and JSON format.

For more information on data quality rules, see *Overview of Data Quality*.

For more information on job details, see *Job Details Page*.

### Enable

Visual profiling is enabled by default.

When the feature is enabled, visual profiling is selected on a per-job basis.

### Profiling Engine

Decoupled from the user interface, the profiling engine performs the calculations required to power the visualizations before job execution and after the job results have been generated.

- In the Transformer page, the profile engine is called for incremental changes whenever a step is added to your recipe, so that you can see immediate updates to the visual profile for each column. It utilizes separate algorithms for generating the data quality bars, column histograms, value counts, frequency distributions, and other relevant statistics. When you dig into the column details, the visual profile is up-to-date and can be updated again based on your selections in that interface.
- During job execution, it is queried as a separate job when profiling is executed across the entire dataset.

**NOTE:** When you choose to profile your results, you are creating two distinct tasks: 1) run your transform recipe against your source and 2) profile the results. Due to the computational complexity of generating the interactive results, a profiling task often takes longer to complete than a transformation task and is therefore an optional element of a job run.

### Exact vs. Approximate Metrics in Visual Profiles

Generally, visual profiles represented in the user interface, in places like column histograms and column details, are exact measurements against the current sample.

On generated results, visual profiles tend to favor approximations.

**NOTE:** The computational cost of generating exact visual profiling measurements on large datasets in interactive visual profiles severely impacts performance. Depending on the environment, you may choose to run profiling jobs on generated results as separate jobs. For more information on enabling this feature, see *Profiling Options*.

Below, you can review details on how metrics are calculated in visual profiling performed in different areas of the platform.

#### User Interface

The UI utilizes the local running environment when displaying visual profiles on sampled data.

**NOTE:** Profiles are executed on the currently sampled data. Results may vary when the full transformation job is executed.

| Metric Type         | Measurement |
|---------------------|-------------|
| Frequency (top-k)   | Exact       |
| Unique value counts | Exact       |

|                                           |       |
|-------------------------------------------|-------|
| Numerical histograms                      | Exact |
| Simple statistics (mean, stdev, min, max) | Exact |
| Quartiles                                 | Exact |

### Trifacta Photon Running Environment

| Metric Type                               | Measurement |
|-------------------------------------------|-------------|
| Frequency (top-k)                         | Approximate |
| Numerical histograms                      | Approximate |
| Simple statistics (mean, stdev, min, max) | Exact       |
| Quartiles                                 | Exact       |

### Spark Running Environment

For profiling jobs, the Spark running environment is used for Spark transformation jobs.

Optionally, profiling jobs may be run on Spark for all jobs, regardless of running environment. For more information, see *Profiling Options* .

| Metric Type                               | Measurement |
|-------------------------------------------|-------------|
| Frequency (top-k)                         | Approximate |
| Numerical histograms                      | Approximate |
| Simple statistics (mean, stdev, min, max) | Exact       |
| Quartiles                                 | Approximate |

# Overview of Sampling

## Contents:

- *How Sampling Works*
    - *Initial Data*
    - *Generating samples*
    - *Important notes on sampling*
    - *Parameterization of samples*
    - *Samples management*
    - *Cancel Sample Jobs*
  - *Choosing Samples*
  - *Limitations*
  - *Sample Invalidation*
  - *Best Practices*
    - *Sampling checkpointing*
  - *Sample Types*
- 

To prevent overwhelming the client or significantly impacting performance, Designer Cloud powered by Trifacta® Enterprise Edition generates one or more samples of the data for display and manipulation in the client application. Since Designer Cloud powered by Trifacta Enterprise Edition supports a variety of clients and use cases, you can change the size of samples, the scope of the sample, and the method by which the sample is created. This section provides background information on how the product manages dataset sampling.

## How Sampling Works

### Initial Data

When a dataset is first created, a background job begins to generate a sample using the first set of rows of the dataset. This **initial data** sample is usually very quick to generate, so that you can get to work right away on your transformations.

- The default sample is the initial sample.
- By default, each sample is 10 MB in size or the entire dataset if it's smaller.
  - If the source data is larger than 10MB in size, a random sample is automatically generated for you when the recipe is first loaded in the Transformer page.
  - The initial sample is selected by default. When the automatic random sample has finished generation, it can be manually selected for display.
- If your source of data is a directory containing multiple files, the initial sample for the combined dataset is generated from the first set of rows in the first filename listed in the directory.
  - If the matching file is a multi-sheet Excel file, the sample is taken from the first sheet in the file. If you are wrangling a dataset with parameters, the initial sample loaded in the Transformer page is taken from the first matching dataset.

### Generating samples

Additional samples can be generated from the context panel on the right side of the Transformer page. Sample jobs are independent job executions. When a sample job succeeds or fails, a notification is displayed for you.

As you develop your recipe, you might need to take new samples of the data. For example, you might need to focus on the mismatched or invalid values that appear in a single column. Through the Transformer page, you can specify the type of sample that you wish to create and initiate the job to create the sample. This sampling job occurs in the background. You can create a new sample at any time. When a sample is created, it is stored within your storage directory on the backend datastore. For more information on creating samples, see *Samples Panel*.

## Sampling methods

Depending on the type of sample you select, it may be generated based on one of the following methods, in increasing order of time to create:

1. on a specified set of rows (firstrows)
2. on a quick scan across the dataset
  - a. By default, Quick Scan samples are executed on the Trifacta Photon running environment.
  - b. If Trifacta Photon is not available or is disabled, the Designer Cloud application attempts to execute the Quick Scan sample on an available clustered running environment.
  - c. If the clustered running environment is not available or doesn't support Quick Scan sampling, then the Quick Scan sample job fails.
3. on a full scan of the entire dataset
  - a. Full Scan samples are executed in the cluster running environment.

## Sampling mechanics

When a non-initial sample is executed for a single dataset-recipe combination, the following steps occur:

1. All of the steps of the recipe are executed on the dataset on the backend cluster, up to the currently selected recipe step.
2. The generated sample is executed on the current state of the dataset.

**NOTE:** When a sample is executed from the Samples panel, it is launched based on the steps leading up to current location in the recipe steps. For example, if your recipe includes joining in other datasets, those steps are executed, and the sample is generated with dependencies on these other datasets. As a result, if you change your recipe steps that occur before the step where the sample was generated, you can invalidate your sample. More information is available below.

When your flow contains multiple datasets and flows, all of the preceding steps leading up to the currently selected step of the recipe are executed, which can mean:

- The number of datasets that must be accessed increases.
- The number of recipe steps that must be executed on the backend increases.
- The time to process the sampling job increases.

## Implications:

- When the sample is displayed in the Transformer page, all steps after the one from which it was executed are computed in the web browser. So, if you have a lengthy series of steps or complex operations after the step where you generated a sample, you can cause performance issues of the Transformer page, including the occasional browser crash. Try generating a new sample later in your flow for better performance.
- If you have added an expensive transformation step, such as a complex union or join, you can improve performance of the Transformer page by generating and using a new sample after the transformation step.

**NOTE:** When a flow is shared, its samples are shared with other users. However, if those users do not have access to the underlying files that back a sample, they do not have access to the sample and must create their own.

## Important notes on sampling

- When sampling from compressed data, the data is uncompressed and then expanded. As a result, the sample size reflects the uncompressed data.
- Changes to preceding steps that alter the number of rows or columns in your dataset can invalidate the current sample, which means that the sample is no longer a valid representation of the state of the dataset

in the recipe. In this case, Designer Cloud powered by Trifacta Enterprise Edition automatically switches you back to the most recently collected sample that is currently valid. Details are below.

## Parameterization of samples

Any parameters that are associated with your dataset can be applied to sampling:

- **Parameters:** Subsequent samples generated from the Transformer page are sampled across all datasets matched by parameter values.
- **Variables:** You can apply override values to the defaults for your dataset's variables at sample execution time. In this manner, you can draw your samples from specific sources files within your dataset with parameters.

## Samples management

After you have created a sample, you cannot delete it through the application.

**NOTE:** Designer Cloud powered by Trifacta Enterprise Edition does not delete samples after they have been created. If you are concerned about data accumulation, you should configure periodic purges of the appropriate directories on the base storage layer. For more information, please contact your IT administrator.

For more information, see *Sample Jobs Page*.

## Cancel Sample Jobs

Generating a sample can consume significant time, system resources, and in some deployments cost. As needed, you can cancel a sample job that is in progress in either of the following ways:

- Locate the in-progress sampling job in the Samples panel. Click X.
- Click the Jobs icon in the left nav bar. Select **Sample jobs**. For more information, see *Sample Jobs Page*.

## Choosing Samples

After you have collected multiple samples of multiple types on your dataset, you can choose the proper sample to use for your current task, based on:

1. **How well each sample represents the underlying dataset.** Does the current sample reflect the likely statistics and outliers of the entire dataset at scale?
2. **How well each sample supports your next recipe step.** If you're developing steps for managing bad data or outliers, for example, you may need to choose a different sample.

**Tip:** You can begin work on an outdated yet still valid sample while you generate a new one based on the current recipe.

## Limitations

- Some advanced sampling options are available only with execution across a scan of the full dataset.
- Undo/redo do not change the sample state, even if the sample becomes invalid.
- When a new sample is generated, any Sort transformations that have been applied previously must be re-applied. Depending on the type of output, sort order may not be preserved.

- Samples taken from a dataset with parameters are limited to a maximum of 50 files when executed on the Trifacta Photon running environment. You can modify parameters as they apply to sampling jobs. See *Samples Panel*.

## Sample Invalidation

With each step that is added or modified to your recipe, Designer Cloud powered by Trifacta Enterprise Edition checks to see if the current sample is valid. Samples are valid based on the state of your flow and recipe at the step when the sample was collected. If you add steps before the step where it was created, the currently active sample can be invalidated. For example, if you change the source of data, then the sample in the Transformer page no longer applies, and a new sample must be displayed.

**Tip:** After you have completed a step that significantly changes the number of rows, columns, or both in your dataset, you may need to generate a new sample, factoring in any costs associated with running the job. Performance costs may be displayed in the Transformer page.

**NOTE:** If you modify a SQL statement for an imported dataset, any samples based on the old SQL statement are invalidated.

- The Transformer page reverts to displaying the most recently collected sample that is currently valid.
- You can generate a new sample of the same type through the Samples panel. If no sample is valid, you must generate a new sample before you can open the dataset.
- A sample that is invalidated is listed under the Unavailable tab. It cannot be selected for use. If subsequent steps make it valid again, it re-appears in the Available tab.

## Best Practices

**The data that is displayed in the data grid is based on all of the upstream samples after which all subsequent steps in each upstream recipe are performed in the browser. If you have a large number of steps or complex steps between the recipe locations for your samples in use and your current recipe location, you may experience performance slow-downs or crashes in the data grid. For more information on sampling best practices, see <https://community.trifacta.com/s/article/Best-Practices-Managing-Samples-in-Complex-Flows>.**

## Sampling checkpointing

All steps between the step in your current sample and the currently displayed step must be computed in the browser. As you build more complex recipes, it's a good idea to create samples at various steps in your recipe, particularly after you have executed a complex step. This type of **sample checkpointing** can improve overall performance.

For example, as soon as you load a new recipe, you should take a sample, which can speed up the process of loading.

**Tip:** You can annotate your recipe with comments, such as: `sample: random` and then create a new sample at that location.

## Sample Types

For more information on sample types, see *Sample Types*.

# Overview of Job Execution

## Contents:

- *Jobs Types*
    - *Transformation job types*
    - *Other job types*
  - *Basic Process for Transformation Jobs*
    - *Job preparation*
    - *Job execution*
    - *Job monitoring*
    - *Job cleanup*
    - *Scheduled jobs*
  - *Job Execution Performance*
    - *Job logs*
  - *Running Environments*
- 

This section provides an overview of how jobs of various types are initiated, managed, and executed in Designer Cloud powered by Trifacta® Enterprise Edition. You can also review summaries of the available running environments for your product edition.

**NOTE:** During job execution of any kind, Designer Cloud powered by Trifacta Enterprise Edition never modifies source data. All transformation is performed on requested elements of the data. If the data needs to be retained for any period of time during use or transformation, it is stored in the browser or in the base storage layer. After the data has been used for the intended purpose, it is removed from temporary storage.

When you build your recipe in the Designer Cloud application, you can see in real-time the effects of the transformations that you are creating. When you wish to produce result sets of these transformations, you must run a job, which performs a separate set of execution steps on the data. Job execution is a separate process for the following reasons:

- In the Transformer page, you are working with a sample of your data. For larger volumes of data, the entire dataset cannot be represented in the browser effectively. So, to apply your recipe to the entire dataset, a separate set of actions must be performed.
- When working with large datasets, you need a running environment on a multi-node cluster that has been designed for parallel processing. Modern running environments are designed to break up data transformation jobs into separate pieces, each of which can be executed on a separate node and then returned to be assembled with the other job parts into the finished result set.
- Job execution can occur asynchronously. When you launch a job, a separate lightweight process assembles the necessary pieces for the job to be executed and then distributes these pieces accordingly. You can continue to work in the Transformer page while your results are being prepared with minimal impact on Designer Cloud application performance or your user experience.

## Other features of job execution:

- Change the format of the source to a different format in the output.
- Change the location where the results are generated.
- Change file-based source data into table-based relational data on the output.
- Write multiple versions of the output at the same time.
- Jobs can also be scheduled.



- Jobs can also be executed using REST APIs, which enables automation of job execution. For more information on job execution via API, see *API Workflow - Run Job*.

## Jobs Types

The following types of jobs can be executed as part of normal operations of the product.

### Job locations:

- A **local job** is one that is executed on the Trifacta node using services that are hosted on it.
- A **remote job** is executed through services that are not hosted on the Trifacta node.

### Transformation job types

Informally, a "job" is considered any action that is performed on a set of data. Commonly, jobs refer to the process of transforming source data into output results. However:

- Transformation jobs are composed of a number of sub-jobs, which handle things like ingestion of data, transformation, and writing of results.
- In addition to jobs that transform data, there are other types of jobs. Discussed later.

### Job groups:

For transformation job types, the following terms apply:

- Internal to the product, a job that is executed on one or more recipes in a flow is called a **jobGroup**.
- A jobGroup is composed of one or more of the job types listed below. Internal to the platform, these are called **jobs**.

The following diagram illustrates how these job types are related.

```
+ myJob jobGroup
+ Connect job
+ Request job
+ Ingest job
+ Transform job
+ Transfer job
+ Process job
```

**Tip:** You can have one or more of each of these job types as part of a single jobGroup.

### Connect

A Connect job performs the steps necessary to connection the Designer Cloud application to the datastore that contains source data. These jobs use the connection objects that are native to the platform or that you have created to make the connection to your imported datasets.

**NOTE:** Depending on the running environment, a Connect job may time out after a period of inactivity or failure to connect, and it may be retried one or more times before the job is marked as failed.

### Request

A Request job sends a query or other request to the source datastore for the assets specified in the imported datasets.

## Ingest

Requested data is brought from the external source to the execution layer, which is the temporary storage location as defined for the running environment.

## Convert

Some formats supported for import are not natively understood by the product. These formats must be converted to a format that the platform can quickly process. This process typically converts binary formats, such as XLS or PDF, into CSV files that are stored temporarily in the base storage layer for purposes of job execution. After the job has succeeded or failed, these converted files are removed.

## Transform

After data has been requested and ingested (if needed), a Transform job converts the steps of a recipe into an intermediate scripted format (called CDF). The CDF script is then passed to the appropriate running environment for transformation of the source data. Additional details are provided later.

## Prepare

If the specified job is publishing results to a connection other than the base storage layer, the results are initially prepared on the base storage layer, after which they are written to the target datastore.

This job type does not apply when the base storage layer is the final destination for the results.

## Transfer

A Transfer job writes the results to the appropriate output location, as specified by the output objects referenced when the job was launched.

## Process

When the transfer is complete, a Process job performs final cleanup, including removal of temp files such as intermediate results written to the base storage layer.

## Other job types

### Profiling

When you execute a transformation job, you can optionally choose to create a visual profile of the results of that job. Visual profiling is a separate job that sometimes takes longer to execute than the job itself, but a visual profile can be useful in highlighting characteristics of your data, including metrics and errors on individual columns.

Visual profiles are available for review in the Job Details page. You can also download PDF or JSON versions of your visual profile.

For more information on visual profiling, see *Overview of Visual Profiling*.

### Sampling

When you are interacting with your source data to transform it through the browser, you are working on a sample of the data. As needed, you can take new samples of the data to provide different perspectives on it. Also, for longer and more complex flows, you should get in the habit of taking periodic samples, which can improve performance in the browser.

Through the Samples panel, you can launch a job to collect a new sample of your data. There are multiple types of sampling, which can be executed using one of the following methods:

- **Quick scan:** These sample types are performed based on a scan a limited number of rows of your data.
  - These samples are based on the first set of rows that are accessible and are quick to execute. However, they cannot pick up in the sample any rows that are deeper in your datasets. For

example, if your source data contains multiple files, quick scan samples might not contain any data from the second or later files.

- These samples are executed in Trifacta Photon.
- **Full scan:** A full scan sample is executed across the entire available dataset.
  - Depending on the size of your dataset, this scanning and sample process can take a while to execute on a large dataset.
  - These samples are executed on the clustered running environment with which the Designer Cloud application is connected.

For more information, see *Overview of Sampling*.

## Basic Process for Transformation Jobs

A transformation job is run based on the outputs that you are trying to generate. For a selected output, the executed job runs the transformations for all of the recipes between the output and all of its imported datasets. For example, generation of a single output could require the transformation of five different recipes that use 13 different imported datasets.

### Job preparation

When you initiate a job through the Designer Cloud application, the following steps occur:

1. A jobGroup is created in the database. It consists of the specification of one or more jobs, as described above.
2. The recipe whose output is being executed is requested from the Trifacta database. This recipe is expanded from storage format and later is stored temporarily in the database for reference.
3. The Designer Cloud application verifies access to data sources and output locations.
4. A job execution graph (flow chart) is created for the various jobs required to complete execution of the jobGroup.
  - a. This graph includes jobs for ingest, transformation, conversion, and other steps, as described above.
5. The graph is sent to the batch job runner service in the platform. This service manages the submission, tracking, and completion of all jobs to supported running environments.
6. Batch job runner requests to the Designer Cloud application to return a Common Dataflow Format (CDF) version of the expanded recipe.
  - a. CDF is a domain-specific language for data transformation that runs anywhere that supports Python execution.
  - b. Wrangle is compiled into CDF format at execution time. This CDF script is delivered to the running environment for execution.
  - c. CDF scripts are internal to the platform and are not accessible to users of the platform.
7. Depending on the running environment, additional modifications to the CDF script may be made before the job is submitted.
8. The batch job runner places the job in a queue for submission to the running environment.

### Job execution

When the job is ready to be pulled from the queue, the following tasks are completed:

1. The job definition, CDF script, and associated resources are submitted to the resource coordinating process of the running environment.
  - a. This coordinator is the batch job runner for local jobs or a dedicated service on remote running environments.
  - b. For example, for EMR execution, which is a remote running environment, the job is submitted to the YARN service, which manages the delegation of work tasks to the various nodes in the cluster.
  - c. In the resource coordinator, jobs from the product are labeled as *Trifacta Transformer* or *Trifacta Profiler* (for profiling jobs).
2. Periodically, batch job runner polls the running environment for status on job execution.
  - a. This status information is stored and updated in the Jobs database.
3. The Designer Cloud application queries the Jobs database for updated information.

- a. These updates are stored in the Trifacta databases for internal services to access to present updates.
  - b. Updates can appear in Flow View page and also in the Jobs and Job Details page, so that you can track progress.
4. During execution, the resource manager arranges for the delivery of data and CDF script objects to nodes of the cluster.
  - a. On these individual nodes, portions of the data are processed through the CDF script.
  - b. The results of this processing is messaged back to the resource manager.
  - c. When all of the nodes have reported back that the job processing has been completed, results are written to the location or locations as defined in the output object that was selected during job execution.
5. Batch job runner updates any available job logs as needed based on the results of the job execution. These logs may be available through the Designer Cloud application .

## Job monitoring

**Transformation jobs:** After a transformation job has been launched, you can monitor the state of the job as it passes through separate stages in the process.

- In Flow View, click the output object. Then, click the Jobs tab. See *Flow View Page*.
- In the Jobs page, you can hover over the status of the job to gather more information. See *Jobs Page*.
- Additional information may be available in the Job Details page. See *Job Details Page*.

**Sample jobs:** In-progress sampling jobs can be tracked through the following locations:

- After you have initiated a sample job through the Samples panel, you can track progress there. See *Samples Panel*.
- All of your sample jobs are available through the Designer Cloud application . See *Sample Jobs Page*.

**Plan runs:** When you have launched jobs as part of a plan run, you can track progress through the Designer Cloud application .

See *Plan Runs Page*.

## Job cleanup

After the results have been written, the following tasks are completed:

1. Applicable job logs are updated and written to the appropriate location.
2. The expanded recipe stored in the database is removed.
3. Any temporary files written to the base storage layer are removed.

## Scheduled jobs

You can also schedule the execution of jobs within your flows. This process works as follows:

1. In Flow View, you define the outputs that you wish to deliver when the flow is executed according to a schedule. These outputs are different objects that the outputs you create from your recipes, but you can define them to write to the same locations.
2. You specify the schedule for when the job is to be executed. Date and time information, as well as frequency of execution, can be defined within the flow.

When the specified time is reached, the job is queued for execution, as described above. For more information, see *Overview of Automator*.

## Job Execution Performance

Job execution is a resource-intensive and multi-layered process that transforms data of potentially limitless size. The following factors can affect performance in the Designer Cloud application and during job execution:

- **Long or complex recipes**
  - Consider breaking recipes into smaller steps. You can change recipes together.
- **Number of columns in your data**
  - The entire width of a dataset must be represented in the sample.
  - Delete unnecessary columns early in your recipe.

**Tip:** If your data is sourced in relational systems, you can apply optimizations to your imported datasets to pre-filter out columns in your dataset before they are ingested into the system. See *Flow Optimization Settings Dialog*.

You can also use custom SQL statements to collect only the columns that are needed from source tables. See *Create Dataset with SQL*.

- **Complexity of transformations**
  - Transformations that blend datasets (join and union) or that perform complex transformations on your dataset (aggregate, window, pivot, etc.) can be expensive to process.
  - If your recipe contains too many of them, it can negatively impact job processing. Consider breaking these across multiple recipes instead.

## Job logs

Separate log files are maintained for each jobGroup. As needed, you can acquire these logs from the Designer Cloud application . In the Job Details page, select **Download logs** from the context menu for a job entry. For more information, see *Job Details Page*.

If there are issues with job execution that cannot be resolved by reviewing the job log, workspace administrators can download a support bundle, which contains additional log information from the platform. For more information, see *Support Bundle Contents*.

## Running Environments

# Trifacta Photon Running Environment

Trifacta Photon is an in-memory running environment that is hosted on the Trifacta node. This environment is initialized only when a job is queued for execution on it. Designed for small- to medium-sized jobs, it offers superior performance due to its location on the Trifacta node and its in-memory processing.

When you choose to run a job in the Designer Cloud application, Trifacta Photon is selected as the default running environment if it is available and the job size is estimated to small or medium.

**Tip:** Trifacta Photon is also used for sampling jobs that are configured to use the Quick Scan method. For more information, see *Overview of Sampling*.

**Tip:** In the Run Job page, select **Photon** to run the job on this running environment.

Trifacta Photon is enabled by default but can be disabled as needed.

**NOTE:** Trifacta Photon cannot process numeric values with more than 16 digits. Columns containing such values are converted to String values, and the digits beyond 16 are converted to 0.

**NOTE:** When a recipe containing a user-defined function is applied to text data, any null characters cause records to be truncated during Trifacta Photon job execution. In these cases, please execute the job on Spark.

**NOTE:** For more information on configuring Trifacta Photon, see *Configure Photon Running Environment*.

# EMR Running Environment

Elastic Map Reduce is a service of Amazon Web Service (AWS) for processing large volumes of data using open source technologies such as Spark. EMR integrates easily with other AWS-based services such as S3, IAM, Glue, and more.

When Designer Cloud powered by Trifacta Enterprise Edition is installed in an EC2 instance on AWS, the Designer Cloud application can be integrated with either pre-existing or new EMR clusters for supported versions of EMR. Additional configuration and limitations apply. For more information, see *Configure for EMR*.

**Tip:** In the Run Job page, select **Spark** to run the job on this running environment when the Designer Cloud application has been integrated with it.

For more information, see <https://aws.amazon.com/emr>.

# AWS Databricks Running Environment

Databricks provides the combination of data lakehouse storage, analytics processing, and artificial intelligence capabilities in a single unified platform. For job execution, the Databricks running environment can be hosted in the Azure or AWS ecosystems.

**NOTE:** This running environment is available only if you install Designer Cloud powered by Trifacta Enterprise Edition on AWS.

**Tip:** In the Run Job page, select **Spark (Databricks)** to run the job on this running environment when the Designer Cloud application has been integrated with it.

Additional configuration is required.

**NOTE:** Use of AWS Databricks is not supported on Marketplace installs.

**NOTE:** When executing a job on the AWS Databricks running environment using a relational source, the job fails if one or more columns has been dropped from the underlying source table. As a workaround, the recipe panel may show steps referencing the missing columns, which can be used to either fix the recipe or the source data.

For more information, see *Configure for AWS Databricks*.

For more information on Databricks, see <https://databricks.com/>.



# Azure Databricks Running Environment

Databricks provides the combination of data lakehouse storage, analytics processing, and artificial intelligence capabilities in a single unified platform. For job execution, the Databricks running environment can be hosted in the Azure or AWS ecosystems.

**NOTE:** This running environment is available only if you install Designer Cloud powered by Trifacta Enterprise Edition on Azure.

**Tip:** In the Run Job page, select **Spark (Databricks)** to run the job on this running environment when the Designer Cloud application has been integrated with it.

Additional configuration is required.

**NOTE:** Use of Azure Databricks is not supported on Marketplace installs.

**NOTE:** When executing a job on the Azure Databricks running environment using a relational source, the job fails if one or more columns has been dropped from the underlying source table. As a workaround, the recipe panel may show steps referencing the missing columns, which can be used to either fix the recipe or the source data.

For more information, see *Configure for Azure Databricks*.

For more information on Databricks, see <https://databricks.com/>.

# Hadoop Spark Running Environment

When Designer Cloud powered by Trifacta Enterprise Edition is installed on a supported version of Cloudera or Hortonworks, the Designer Cloud application can be configured to execute larger jobs on the cluster instance of Spark. Spark leverages in-memory capabilities on individual nodes for faster processing of distributed analytics tasks, with spillover to disk as needed.

**Tip:** In the Run Job page, select **Spark** to run the job on this running environment when the Designer Cloud application has been integrated with it.

Spark requires a backend distributed storage layer:

- On AWS-based deployments, this storage layer is S3.
- On Hadoop-based deployments, this storage layer is HDFS.

Additional configuration is required.

**NOTE:** When executing a job on the Spark running environment using a relational source, the job fails if one or more columns has been dropped from the underlying source table. As a workaround, the recipe panel may show steps referencing the missing columns, which can be used to fix to either fix the recipe or the source data.

**NOTE:** The Spark running environment does not support use of multi-character delimiters for CSV outputs. You can switch your job to a different running environment or use single-character delimiters. This issue is fixed in Spark 3.0 and later. For more information on this issue, see <https://issues.apache.org/jira/browse/SPARK-24540>.

For more information, see *Configure for Spark*.

# Overview of TBE

## Contents:

- *Limitations*
  - *Enable*
  - *Column by Example*
    - *CBE for Datetime*
    - *Alternatives*
- 

**Transformation by Example (TBE)** enables you to build recipe objects by mapping example output values for source values. Designer Cloud powered by Trifacta® Enterprise Edition then interprets the differences between the inputs and outputs to determine the transformation required to map them.

TBE leverages pattern-based matching and predictive transformation to derive transformations. When you provide explicit mappings of input value to output, the mapping is passed through predictive transformation to determine the best possible matching pattern.

- For more information on patterns, see *Overview of Pattern Matching*.
- **Predictive Transformation** is a core component of Designer Cloud powered by Trifacta Enterprise Edition. Based upon user input, the platform provides one or more suggestions of ways in which to transform the data.
  - In TBE, these suggestions are rendered as elements of the transformation in progress.
  - For more information, see *Overview of Predictive Transformation*.

## Use cases:

**Tip:** TBE simplifies the process of defining patterns to match all values in your source column. Since you know and can specify the exact desired output, you can leave the details of defining the pattern or patterns required to match input to output to the product.

Transformation by Example works well in the following use cases:

- You are just getting started with the product and would like to get productive quickly to transform your data into known outputs.
- Your data has groups of values, each of which needs transformation in a different way. In a single recipe step, you can perform these transformations across all groups.
- Your data has special-case exceptions that must be transformed.

**Tip:** You can use this feature as a final cleanup for other transformations. If you have a transformation that handles 90% of the cases in a column, you can use this transformation to handle the remainder.

## Artifacts:

When a TBE step is added to your recipe, the number of individual changes can be many megabytes of data. Instead of storing these objects within the recipe definition, they are stored as a set of artifacts in the artifact storage database and referenced from the recipe.

- These artifacts exist outside the scope of the recipe file.
- These artifacts must be stored in a Trifacta database for the step to be editable and exportable.

**NOTE:** If the artifact storage service is disabled, this feature is unusable.

- When a flow is exported, an `artifact.data` file is included as part of the export. This file must be imported with the flow definition, or the TBE step in the imported flow is broken. For more information, see *Export Flow*.

## Limitations

- TBE works best for inputs that are text-based data types (e.g. String, State, URL, etc.).
  - Non-text inputs are treated as String type and may result in unexpected outputs (Integer, Decimal, etc.).
  - You cannot use multi-value inputs, such as Arrays or Objects, or use the feature to create them.

**Tip:** If you have Array or Object input columns, convert them to String type before using TBE.

- TBE bases its transformations on the currently displayed sample.
  - Even if you accurately map all values in your sample, some other values in the full dataset may not be mapped by the transformation.
  - You may need to take additional samples of other parts of the entire dataset to generate a more accurate transformation.
- Arithmetic operations or other numeric functions are not supported.
- You cannot create multiple columns from a single TBE step.

## Enable

This feature can be enabled and disabled through the Workspace Settings page.

### Steps:

1. Login to the application as an administrator.
2. From the left nav bar, select **User menu > Admin console > Workspace settings**.
3. Locate the following setting: **Create column from examples feature**.
4. Set this value to `Enabled`.
5. Save your changes.

See *Workspace Settings Page*.

This feature uses the Artifact Storage service and related database to store and retrieve historical data on TBEs. This database is installed as part of the normal database install or upgrade processes.

## Column by Example

In column-by-example transformations, you create a new column from an existing one by mapping input to output values.

### General workflow:

1. Select the column to use as input data.
2. Change the column to String data type, if needed.
3. From the column menu, select **Create column from examples**. See *Transformation by Example Page*.
4. Transform by example:
  - a. Locate a row containing an example value to transform.
  - b. In the corresponding row in the Preview column, you can enter in the new value to which the input is mapped.
  - c. The transformation in development is updated to accurately capture the mapping you just performed. Additional rows in the output column may be accurately mapped, as well.
5. Repeat the above steps until all values in the output column appear to be accurately mapped.

6. When satisfied, add the transformation to your recipe.
7. Change the data type of the target and the source columns, if needed.
8. Remove the source column, if needed.

For more information, see *Create Column by Example*.

## CBE for Datetime

Column-by-example also works on Datetime columns. When you use a Datetime column as your input, you specify the output values in the date/time format that you wish to use. That input value and all similarly formatted inputs should be converted to the output format. You can then specify additional example outputs for input values in a different format to standardize all of the values in the output column.

**NOTE:** For Datetime formatting to work properly, the input column must be specified as Datetime data type.

## Alternatives

For string-based inputs, the following options in *Wrangle* may assist in performing the same functions that you are trying to do in TBE:

| Wrangle                  | Description                                                                                                  |
|--------------------------|--------------------------------------------------------------------------------------------------------------|
| <i>Extract Transform</i> | You can use the extract transform to retrieve sub-strings from a column and insert into a new column.        |
| <i>String Functions</i>  | Wrangle supports a variety of string manipulation functions, which can be used to gather data from a string. |

# Overview of Data Quality

## Contents:

- *Data Quality Characteristics*
  - *Schema Validation*
    - *Assign targets*
  - *Identify Anomalies*
    - *Data quality bar*
    - *Column histogram*
    - *Column details*
  - *Standardization*
  - *Data Quality Functions*
    - *Type functions*
    - *Count functions*
    - *Aggregation functions*
    - *Statistical functions - single column*
    - *Statistical functions - multi-column*
  - *Data Quality in Job Details*
    - *Visual profiling*
    - *Rules tab*
- 

Designer Cloud powered by Trifacta® Enterprise Edition provides multiple mechanisms to transform and standardize data to meet usage needs, including profile visualizations and type-based quality bars to identify potential anomalies and quality problems. **Data quality** checks can be applied during data import, transformation, or export in the form of visual profiling.

Broadly speaking, data quality identifies the degree to which data is usable and responsive to your use case. When you assess data quality, you are designing tests to assess its suitability for generic usage and for your specific uses.

## Data Quality Characteristics

Data quality covers the following characteristics:

- **Completeness:** values are present where they are needed and expected
- **Accuracy:** data is substantively free of errors
- **Consistency:** a dataset can be matched across different data sources of the enterprise
- **Timeliness:** data values are up-to-date
- **Uniqueness:** aggregate data are free from any duplication via filters or other transformations of source data
- **Validity:** data are structured based on an adequate and rigorous classification system
- **Availability / Accessibility:** data are made available to the relevant stakeholders
- **Traceability:** the history, processing and location of the data under consideration can be easily traced

## Schema Validation

## Type inference

When data is imported, the Designer Cloud powered by Trifacta Enterprise Edition attempts to infer the data types in the source and to type columns in the dataset accordingly. Type inference uses the first 20-25 rows of the initial sample to assess the appropriate data type to apply to the column. For more information, see *Type Conversions*.

Some imported data, such as relational tables, may include schema information to identify the data type of each column. In some cases you can disable type inferencing on imported data:

- **Global:** Trifacta administrators can disable type inferencing for all imported schematized sources. In this manner, the Designer Cloud powered by Trifacta platform uses the schema of the source to define the initial types assigned to the columns of the dataset.
- **Connections:** As part of the definition of a connection, you can optionally choose to disable type inference. For more information, see *Create Connection Window*.
- **Per-dataset:** When you import a dataset, you can modify the import settings for the selected source to disable type inference. See *Import Data Page*.

Assign targets

To assist in your transformation efforts, you can assign a target schema for each recipe. This target schema is super-imposed on the columns of your data. Using visual tools to review differences and select changes, you can rapidly convert the structure of your dataset in development to meet the expected target schema. For more information, see *Overview of RapidTarget*.

Identify Anomalies

In the Transformer page, you can use the available visual tools to review the data quality characteristics of the columns in your data. These data visualizations and type-based quality bars can assist in identifying potential anomalies and quality problems.

Data quality bar

At the top of each column, you can see a data quality bar, which uses the following color coding to validate the column values against the selected column type.

| Color bar | Description                                                   |
|-----------|---------------------------------------------------------------|
| green     | Values that are valid for the current data type of the column |
| red       | Values that are mismatched for the column data type           |
| gray      | Missing or null values                                        |

**Tip:** Click any of the color bars to receive suggestions for transformations to add to your recipe. See *Overview of Predictive Transformation*.

**Tip:** You can change a column's data type in the column header. See *Column Menus*.

For more information, see *Data Quality Bars*.

Column histogram

In the column header, you can review the count and distribution of values in the column. A column's histogram can be useful for identifying anomalies or for selecting specific sets of values in the column for further exploration.



**Tip:** Click and drag over any set of values to receive suggestions for transformations to add to your recipe. See *Overview of Predictive Transformation*.

See *Column Histograms*.

## Column details

Through the Column Details panel, you can explore the quality and distribution of the values in the column. The contents of the panel vary depending on the data type. For example, if the column is typed for Datetime values, then the Column Details panel includes information on the distribution of values across the days of the week and days of the month.

For all data types, you can review useful statistics on statistical quartiles, the uniqueness of values, mismatches, and outliers.

**Tip:** The Column Details panel is very useful for acquiring statistical information on column values in a visual format. Click any data quality bar to be prompted for suggestions of transformation steps. See *Overview of Predictive Transformation*.

For more information, see *Column Details Panel*.

## Standardization

You can use the Standardization tool to standardized clustered sets of column values to values that are common and consistent throughout your enterprise's data. For more information, see *Overview of Standardization*.

## Data Quality Functions

The following functions are available for assessing data quality.

### Type functions

| Item                  | Description                                                                                                                                                                                                                              |
|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| NULL Function         | The NULL function generates null values.                                                                                                                                                                                                 |
| IFNULL Function       | The IFNULL function writes out a specified value if the source value is a null. Otherwise, it writes the source value. Input can be a literal, a column reference, or a function.                                                        |
| IFMISSING Function    | The IFMISSING function writes out a specified value if the source value is a null or missing value. Otherwise, it writes the source value. Input can be a literal, a column reference, or a function.                                    |
| IFMISMATCHED Function | The IFMISMATCHED function writes out a specified value if the input expression does not match the specified data type or typing array. Otherwise, it writes the source value. Input can be a literal, a column reference, or a function. |
| IFVALID Function      | The IFVALID function writes out a specified value if the input expression matches the specified data type. Otherwise, it writes the source value. Input can be a literal, a column reference, or a function.                             |
| ISNULL Function       | The ISNULL function tests whether a column of values contains null values. For input column references, this function returns true or false.                                                                                             |
| ISMISSING Function    | The ISMISSING function tests whether a column of values is missing or null. For input column references, this function returns true or false.                                                                                            |
| ISMISMATCHED Function | Tests whether a set of values is not valid for a specified data type.                                                                                                                                                                    |
| VALID Function        | Tests whether a set of values is valid for a specified data type and is not a null value.                                                                                                                                                |



|                      |                                                                                                                                                                                                                        |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PARSEINT Function    | Evaluates a String input against the Integer datatype. If the input matches, the function outputs an Integer value. Input can be a literal, a column of values, or a function returning String values.                 |
| PARSEBOOL Function   | Evaluates a String input against the Boolean datatype. If the input matches, the function outputs a Boolean value. Input can be a literal, a column of values, or a function returning String values.                  |
| PARSEFLOAT Function  | Evaluates a String input against the Decimal datatype. If the input matches, the function outputs a Decimal value. Input can be a literal, a column of values, or a function returning String values.                  |
| PARSEARRAY Function  | Evaluates a String input against the Array datatype. If the input matches, the function outputs an Array value. Input can be a literal, a column of values, or a function returning String values.                     |
| PARSEOBJECT Function | Evaluates a String input against the Object datatype. If the input matches, the function outputs an Object value. Input can be a literal, a column of values, or a function returning String values.                   |
| PARSESTRING Function | Evaluates an input against the String datatype. If the input matches, the function outputs a String value. Input can be a literal, a column of values, or a function returning values. Values can be of any data type. |

## Count functions

The following functions measure counts of values within a column, optionally counted by group.

| Item                          | Description                                                                                                                              |
|-------------------------------|------------------------------------------------------------------------------------------------------------------------------------------|
| <i>COUNT Function</i>         | Generates the count of rows in the dataset. Generated value is of Integer type.                                                          |
| <i>COUNTA Function</i>        | Generates the count of non-null rows in a specified column, optionally counted by group. Generated value is of Integer type.             |
| <i>COUNTDISTINCT Function</i> | Generates the count of distinct values in a specified column, optionally counted by group. Generated value is of Integer type.           |
| <i>UNIQUE Function</i>        | Extracts the set of unique values from a column into an array stored in a new column. This function is typically part of an aggregation. |

## Aggregation functions

| Item                     | Description                                                                                                                                                                                                      |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>AVERAGE Function</i>  | Computes the average (mean) from all row values in a column or group. Input column can be of Integer or Decimal.<br><br>See also:<br><ul style="list-style-type: none"> <li><i>AVERAGEIF Function</i></li> </ul> |
| <i>SUM Function</i>      | Computes the sum of all values found in all row values in a column. Input column can be of Integer or Decimal.                                                                                                   |
| <i>MIN Function</i>      | Computes the minimum value found in all row values in a column. Input column can be of Integer, Decimal or Datetime.                                                                                             |
| <i>MAX Function</i>      | Computes the maximum value found in all row values in a column. Inputs can be Integer, Decimal, or Datetime.                                                                                                     |
| <i>MODE Function</i>     | Computes the mode (most frequent value) from all row values in a column, according to their grouping. Input column can be of Integer, Decimal, or Datetime type.                                                 |
| <i>MINDATE Function</i>  | Computes the minimum value found in all row values in a Datetime column.                                                                                                                                         |
| <i>MAXDATE Function</i>  | Computes the maximum value found in all row values in a Datetime column.                                                                                                                                         |
| <i>MODEDATE Function</i> | Computes the most frequent (mode) value found in all row values in a Datetime column.                                                                                                                            |

## Statistical functions - single column

Variations in these functions:

- Some of these functions have variations that use the sample population method of computation.

- IF conditional functions can be used to compute statistical computations based on a condition.

## General statistics

| Item                       | Description                                                                                                                                                                   |
|----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>VAR Function</i>        | Computes the variance among all values in a column. Input column can be of Integer or Decimal. If no numeric values are detected in the input column, the function returns 0. |
| <i>STDEV Function</i>      | Computes the standard deviation across all column values of Integer or Decimal type.                                                                                          |
| <i>MEDIAN Function</i>     | Computes the median from all row values in a column or group. Input column can be of Integer or Decimal.                                                                      |
| <i>QUARTILE Function</i>   | Computes a specified quartile across all row values in a column or group. Input column can be of Integer or Decimal.                                                          |
| <i>PERCENTILE Function</i> | Computes a specified percentile across all row values in a column or group. Input column can be of Integer or Decimal.                                                        |

| Item                                  | Description                                                                                                                                 |
|---------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| <i>APPROXIMATEMEDIAN Function</i>     | Computes the approximate median from all row values in a column or group. Input column can be of Integer or Decimal.                        |
| <i>APPROXIMATEQUARTILE Function</i>   | Computes an approximation for a specified quartile across all row values in a column or group. Input column can be of Integer or Decimal.   |
| <i>APPROXIMATEPERCENTILE Function</i> | Computes an approximation for a specified percentile across all row values in a column or group. Input column can be of Integer or Decimal. |

## Statistical functions - multi-column

| Item                   | Description                                                                                                               |
|------------------------|---------------------------------------------------------------------------------------------------------------------------|
| <i>COVAR Function</i>  | Computes the covariance between two columns using the population method. Source values can be of Integer or Decimal type. |
| <i>CORREL Function</i> | Computes the correlation coefficient between two columns. Source values can be of Integer or Decimal type.                |

## Data Quality in Job Details

When you run a job and generate results, you can review the the quality of the data of the generated output.

### Visual profiling

In parallel with executing the job, you can generate a visual profile of the generated results. This visual profile provides graphical representations of the valid and mismatched values against each column's data type, as well as indications about missing values in the output.

**Tip:** Visual profiles can be downloaded in PDF or JSON format for offline analysis.

Visual profiling is selected as part of the job definition process. See *Run Job Page*.

For more information, see *Overview of Visual Profiling*.

## Rules tab

When visual profiling is enabled for your job, the Rules tab in the Job Details page contains the results of the data quality rules for the job's recipes applied across the entire dataset.

**Tip:** Data quality rules are available for download in JSON and PDF format. For more information, see *Job Details Page*.

For more information, see *Job Details Page*.

# Overview of Sharing

Contents:

- *Enable*
- *Sharing Model*
  - *Owners and collaborators*
  - *Workspace role by object type*
  - *Fine-grained sharing privileges for individual shared objects*
- *Shareable Objects*
  - *Sharing Flows*
  - *Share Connections*
  - *Share Plans*

In a collaborative environment, it can be helpful to be able to have multiple users work on the same assets or to create copies of good quality work to serve as templates for others. Designer Cloud powered by Trifacta® Enterprise Edition enables users to collaborate on the same flow objects or to create copies for others to use for independent work.

This section provides an overview of sharing principles, limitations, and approaches.

## Enable

Sharing can be enabled and disabled through Workspace settings by a workspace administrator. To enable, set the following to `Enabled`.

Sharing

For more information, see *Workspace Settings Page*.

## Sharing Model

**NOTE:** You cannot share with users outside of your current workspace, including any account that you may have in a different workspace.

## Owners and collaborators

The following are the basic types of users of a shared object:

| User Type       | Description                                                                                                                                                                                                                                                      |
|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Owner           | <div>Typically, the owner is the original creator of the shared object. This user has maximum permissions on the object.</div> <div><b>NOTE:</b> There can be only one owner on an object. Only the owner or a workspace admin can delete a shared object.</div> |
| Workspace admin | All workspace admins have owner rights on all objects in the workspace.                                                                                                                                                                                          |

|              |                                                                                                                                                                                                                                                    |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Collaborator | <p>Any user who has been shared an object is a <b>collaborator</b>. A collaborator can have the one of the following permissions on the object:</p> <ul style="list-style-type: none"> <li>• Editor</li> <li>• Viewer</li> </ul> <p>See below.</p> |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Workspace role by object type

Individual users can be assigned one or more workspace roles. A **role** is a set of permissions.

For each type of shareable object, a workspace administrator can define within a role the permissions that workspace users have on the object type. Below, you can review the workspace level permissions and the implications on sharing:

| Workspace Permission | Description                                                                    | If object shared, default permissions on the object |
|----------------------|--------------------------------------------------------------------------------|-----------------------------------------------------|
| Author               | Assigned user can create and delete new objects of this type in the workspace. | Editor                                              |
| Editor               | Assigned user can modify objects of this type with limitations. See below.     | Editor                                              |
| Viewer               | Assigned user has read-only access to this type of object.                     | Viewer                                              |

For more information, see *Workspace Roles Page*.

## Fine-grained sharing privileges for individual shared objects

When an object is shared, the user who is sharing the object can specify the privilege level for the target user on the shared object, which provides finer-grained access controls on individual objects:

- Workspace-level privileges on object types can be overridden for individual objects.
- The workspace-level privileges define the maximum set of privileges that you can share on an object with the target user.
- For example, a user with Viewer privileges on flows at the workspace level cannot be given Editor privileges on any individual flow.

### Limitations:

- Fine-grained sharing privileges apply to flows and connections only.
- Users who have received changes in privileges on individual objects should log out and log in again to see those changes.

## Shareable Objects

The following types of objects can be shared with other workspace users:

- Flows
- Connections
- Plans

## Sharing Flows

In the collaborative approach, two or more users can work on the same flow. When a flow is shared, all flow objects are shared, including:

- Imported datasets

**NOTE:** A dataset that is created with parameters cannot be modified by a collaborator. It can only be modified by the owner.

- Recipes
- Output objects
  - If available, any output SQL scripts are also shared.
- Job results
- Webhook tasks

**NOTE:** Sharing of data is managed at the flow level. You cannot share individual recipes or datasets from within a flow.

**NOTE:** You cannot share a flow with yourself.

All collaborators have access to the above objects, as long as they have access to the underlying sources. See below.

### Use cases:

- Distribute the work on a flow with multiple recipes among team members for faster throughput.
- Pass recipes to others for commenting, editing, and general review.
- When stuck, share the flow with the team expert to provide guidance.

### Privileges

**Underlying datasets:** Sharing a flow does not change the permissions to the underlying data. If a user with whom a flow has been shared does not have access to the data on the datastore, the user cannot work with the flow's datasets.

- Datasets that are accessed through private connections cannot be shared, unless the connection is also shared.
- Stricter permissions sets on the datastore can adversely affect users' ability to access shared flows.

**Sharing samples:** A flow's samples are not necessarily available to all users who have been shared the flow. In some cases, if a user who has been shared a flow does not have access to a recipe's sample, the user may have to collect a separate sample to view data or edit the recipe associated with the sample. To enable universal access to shared samples, you can use either of the following permissions schemes:

1. The default output directories for any user can be accessed by any other user. This configuration must be managed in the base storage layer.
2. When the sample is executed, an individual user must set his or her default output directory to a location that shared users of the flow can access.

When flows are shared with you, you can access them through the Shared with Me tab in the Flows page. See *Flows Page*.

#### Editor privileges:

- Datasets
  - Use the imported datasets and references as sources in other flows accessible to the collaborator.
  - Add new imported datasets.
  - Remove existing imported datasets.
  - Change the source of datasets.
  - Edit dataset names and descriptions.
- Recipes
  - Add new recipes.
  - Edit the existing recipes, including multi-dataset operations such as union or join.
  - Delete recipes.
  - Copy recipes within the shared flow.
  - Move recipes to the shared flow.
  - Move recipes out of the shared flow.
  - Run jobs.
- Schedules
  - Create new schedules.
  - Edit schedules.

#### Viewer privileges:

- User can access the flow and run jobs.
- User cannot modify the flow.
- Schedules
  - Create new schedules.
  - Edit schedules.

#### Collaborator (Editor and Viewer) limitations:

Collaborators do not have the following privileges on a flow shared with them:

- Flow
  - Delete the flow
  - Edit the name and description of the flow
  - Remove the flow owner's access to the flow
- Datasets
  - Delete imported datasets
  - Modify imported datasets

**NOTE:** Collaborators cannot modify datasets created with custom SQL.

For more information on the privileges for Viewer and Editor roles, see *Privileges and Roles Reference*.

#### Editing recipes

Owners and Editors have the same privileges to edit recipes in the shared flow. In the Edit History, edits appear under the usernames of the individual contributors.

**NOTE :** Multiple editors cannot make changes to the same recipe at the same time.

**NOTE:** When a column is hidden from a dataset, it is hidden for all users.

**Tip:** You can review the history of changes to a recipe through the Edit History for a recipe. See *Recipe Panel*.

### Removing access

You can remove sharing access to a flow. When a flow is no longer shared with a user, that user:

- Cannot see the flow or its objects
- Cannot access them, if the user knows the location of the objects

**NOTE:** If a dataset from a shared flow is referenced in another flow, when sharing access is removed from the flow, the referenced dataset is still available in the other flow.

**NOTE:** If a flow is unshared with you, you cannot see or access the datasources for any jobs that you have already run on the flow, including any PDF profiles that you generated. You can still access the job results. This is a known issue.

### Share Connections

When initially created, a connection is **private**. It is accessible only to the user who created it.

Through the Connections page, you can share your connections with other users:

- **Share connection with individual users:** You can share your connection with specified workspace users.
  - You can also share connections that have been shared with you.
- **Make connection public:** Public connections are available for use by all users.

**NOTE:** Only a workspace admin can make connections public. After a connection has been made public, it cannot be made private again. You must delete and recreate the connection.

When connections are shared with you, you can access them through the Shared with Me tab in the Connections page. See *Connections Page*.

### Sharing credentials:

When shared, private connections can be shared with or without credentials. If credentials are not shared, new users of the shared connection must supply their own credentials. Those credentials must be permitted access if access to any datasets previously imported through the connection is required.

**NOTE:** A workspace admin has owner-level access to all connections. However, a workspace admin cannot access or use a connection's credentials if those credentials were not shared by the owner of the connection. For more information, see *Workspace Admin Permissions*.



**NOTE:** Password values for credentials are always masked in the user interface.

**NOTE:** For SSO connections, credentials are never shared. Instead, the Kerberos principal of the user with whom the connection is shared is used to connect. That principal must have the appropriate permissions to access the data.

For more information, see *Connections Page*.

### Sharing connections through flows:

When a flow is shared, any connections associated with it are automatically shared to the specified users. If the connection is configured to do so, credentials are included, so that the new users can immediately begin using the flow.

For more information, see *Flow View Page*.

For more information on the privileges for Viewer and Editor roles, see *Privileges and Roles Reference*.

### Share Plans

Plans that you create can be shared with other users. In the Plans page, select **Share** from a plan's context menu.

Depending on whether you created the plan, you may have the following set of privileges:

| You are      | Privileges                                                                                                                                                                                                                                                                                                                                                                   |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Owner        | The owner created the plan and can schedule the plan and has all editor privileges.                                                                                                                                                                                                                                                                                          |
| Collaborator | A collaborator has been shared the plan as a Viewer or Editor. Privileges to the plan that are limited in the following ways: <ul style="list-style-type: none"><li>• Collaborators cannot delete plans that have been shared with them.</li><li>• Collaborator access to the plan may be further filtered based on assignments at the workspace level. See below.</li></ul> |

When a plan is shared with you, you are a collaborator on the plan. A collaborator has the following capabilities based on the plan privileges assigned to your workspace role:

| Plan Privilege | Description                                                                                                                                                                                                                                                       |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Author         | <ul style="list-style-type: none"><li>• Create plans.</li><li>• Delete plans that you create.</li><li>• All Editor privileges.</li></ul>                                                                                                                          |
| Editor         | <ul style="list-style-type: none"><li>• Edit parameters in entitled plans</li><li>• Manage email notifications on entitled plans</li><li>• Update entitled plans names and descriptions</li><li>• Share entitled plans</li><li>• All Viewer privileges.</li></ul> |
| Viewer         | <ul style="list-style-type: none"><li>• View and run entitled plans</li><li>• View runs and jobs from entitled plans</li><li>• Export entitled plans</li></ul>                                                                                                    |

For more information on the privileges for Viewer and Editor privileges, see *Privileges and Roles Reference*.

For more information, see *Share a Plan*.

# Overview of Job Monitoring

## Contents:

- *Monitoring Phases*
    - *Connect*
    - *Request*
    - *Transfer*
    - *Prepare*
    - *Process*
  - *Enable*
  - *Configure*
    - *Enable phases in Data sources tab*
    - *Enable phases in Outputs tab*
  - *Monitoring Jobs in the Application*
    - *Flow View*
    - *Import*
    - *Job Details Page*
- 

Designer Cloud powered by Trifacta® Enterprise Edition supports detailed monitoring of a job throughout each phase of its execution.

## Limitations:

- Applies only to ingest and publishing jobs
- Applies only to JDBC datasets

## Monitoring Phases

These phases apply to ingest and publishing jobs. Information on them is surfaced in the application.

### Connect

In the Connect phases, Designer Cloud powered by Trifacta Enterprise Edition uses the specified connection for the flow to connect to the source of the job.

**NOTE:** Errors in this phase typically involve issues in the connection definition or in the network configuration or availability.

### Request

After the platform has been able to connect to the datastore, the Request phase entails the submission of the request to the datastore for the assets. For example, for JDBC-based datasets, this phase covers the SQL query of the database through the response that the query was successfully executed.

**NOTE:** Errors in this phase typically reflect errors in the SQL query, which can include renaming or moving of assets in the datastore.

**NOTE:** If assets are retrieved via custom SQL query, you may need to review the query and validate it through the Designer Cloud application . For more information, see *Create Dataset with SQL*.

## Transfer

This phase covers the transfer of assets from the datastore to the platform.

**NOTE:** Errors in this phase typically indicate issues with permissions.

## Prepare

**NOTE:** This phase applies to publishing jobs only.

Depending on the destination, the Prepare phase includes the creation of temporary tables, generation of manifest files, and the fetching of extra connections for parallel data transfer.

## Process

After the data has been transferred to the platform, this phase covers the processing of cleanup after data transfer, including the dropping of temporary tables or copying data within the instance.

## Enable

The base feature is enabled by default.

```
"feature.enableJobMonitoring": true,
```

## Configure

Optionally, you can enable the following capabilities in the Designer Cloud application. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.

### Enable phases in Data sources tab

To display separate columns in the Data sources tab of the Job Details page for each phase on an ingest job, set the following parameter to `true`:

```
"jobMonitoring.enablePhasesInDatasourcesTable": true,
```

### Enable phases in Outputs tab

To display separate columns in the Outputs tab of the Job Details page for each phase for a publish job, set the following parameter to `true`:

```
"jobMonitoring.enablePhasesInOutputsTable": true,
```

Save your changes and restart the platform.

## Monitoring Jobs in the Application

When the base feature is enabled, you can monitor jobs in the following locations.

Flow View

- Track phases in the Jobs panel in Flow View. Hover the mouse over the link to the job.
- See *Flow View Page*.

Import

**NOTE:** This feature may require enablement in your deployment. See *Configure JDBC Ingestion*.

Import Data:

For long-loading datasets, you can track the progress of the import through the Import Data page as you specify the import. See *Import Data Page*.

Library:

After specifying the import, if the data is continuing to be ingested, you can track progress through the Library page. See *Library Page*.

Dataset Details Page:

In the Dataset Details page, you can monitor the ingest progress. Hover over the Status link.

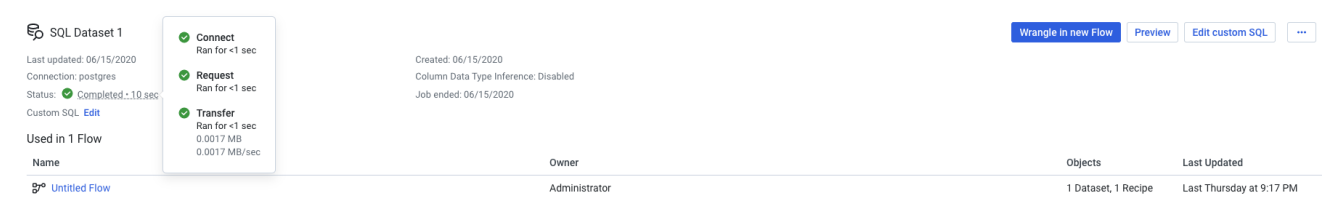


Figure: Dataset Details Page - Job Monitoring

Job Details Page

- Track phases of progress by hovering over the job in progress in the Job Details page.
- Review new and better detail in the Job Details page. Click **View Details** for the job listing.
- For more information, see *Job Details Page*.

Datasources tab - Phased ingest monitoring

If job monitoring phases have been enabled for the Datasources tab, the tab looks like the following:

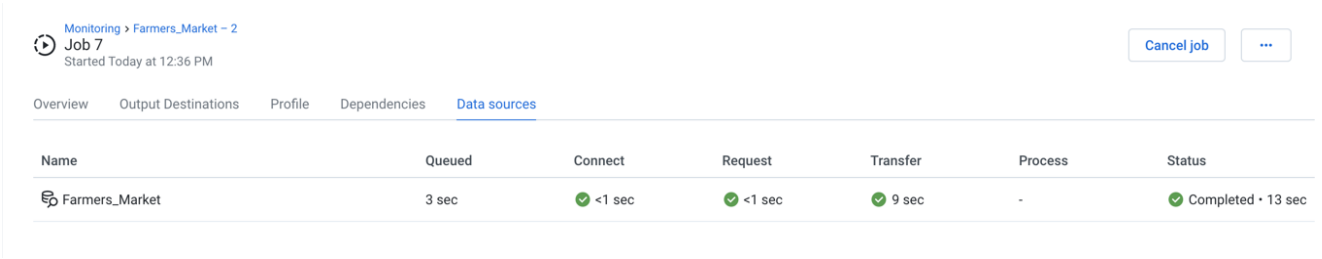



Figure: Job monitoring in the Datasources tab

View details:

If an ingest job succeeds or fails, you can click **View details** in the status column for additional information on each phase of the ingest job:

Details

×

 Farmers\_Market

Details

SQL

---


Type

oracle

Created

June 24th 2019, 11:38 am

Ingestion


 Completed • 13 sec

Ingestion details


Queued

In queue for 3 sec


Connect

 Ran for <1 sec

Request

 Ran for <1 sec

Transfer

 Ran for 9 sec  
 76.8760 MB  
 8.5418 MB/sec

**Figure: View details on monitoring ingest jobs**

#### Output destinations tab - Phased publishing monitoring

If job monitoring phases have been enabled for the Output Destinations tab, the tab looks like the following:

Monitoring > Farmers\_Market - 2

Job 7

Finished Today at 12:37 PM

Publish results
...

Overview







Output Destinations

Profile

Dependencies

Data sources

You can download the generated results locally or [publish](#) to another storage location.

| Name                                                                                                   | Queued | Connect                                                                                  | Prepare                                                                                  | Transfer                                                                                  | Process                                                                                    | Status                                                                                                 |
|--------------------------------------------------------------------------------------------------------|--------|------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------|
|  test_20190624_193621 | 2 sec  |  <1 sec |  <1 sec |  <1 sec |  <1 sec |  Completed • 15 sec |


**Figure: Job monitoring in the Datasources tab**

**View details:**

If a publishing job succeeds or fails, you can click **View details** in the status column for additional information on each phase of the publishing job:

Details

×

 test\_20190624\_193621

Data preview

| # | FM_ID   | RBC | Market_Name                            |
|---|---------|-----|----------------------------------------|
|   | 1005627 |     | Montgomery Farmers' Market             |
|   | 1002686 |     | Hines Veterans Hospital Farmers Market |
|   | 1004509 |     | Montgomery Village Farmers Market      |
|   | 1001779 |     | Farmstand at CHSAS                     |
|   | 1008852 |     | Monticello Farmers' Market             |
|   | 1007417 |     | Farmville Farmers Market               |
|   | 1002538 |     | Monticello Old Jail Market             |

Type

ORACLE

Location

MKOHLLI\_TEST/test\_20190624\_193621


Size

3 columns · 3 types

Start Time

June 24th 2019, 12:36 pm

Publish


 Completed · 51 sec

Publish details


Queued

In queue for 38 sec

Connect

 Ran for <1 sec

Prepare

 Ran for <1 sec

5 connections

**Figure: View details on monitoring ingest jobs**



# Overview of Automator

## Contents:

- *Limitations*
  - *Data Management*
    - *Flows for scheduling*
  - *Schedule a Job*
  - *Job Execution*
    - *Tracking*
  - *Configure*
  - *via API*
- 

As needed, you can use the **Automator** to schedule the execution of recipes in your flows on a recurring basis. For example, if the source file of your flow is updated outside of the application on a weekly basis, you can define a schedule to execute the recipe associated with the related imported dataset after the data has been refreshed. When the scheduled job successfully executes, you can collect the wrangled output in the specified output location, where it is available in the published form that you have specified.

- This feature was formerly known as, "scheduling."

To schedule a job, you must create the following configuration objects:

1. **Define a schedule** - For each flow you can define a schedule. A **schedule** specifies one or more recurring times (**triggers**) when scheduled jobs for the flow are executed. For example, in a single schedule, you can specify daily trigger times for incremental updates and monthly execution times for rollups.

**Tip:** The scheduler supports a modified form of cron job syntax. For more information, see *cron Schedule Syntax Reference*.

2. **Define one or more scheduled destinations** - When you specify a **scheduled destination** for a recipe, the recipe is executed whenever one of the schedule's execution times occurs. Scheduled destinations are specified like regular destinations in flow view.

**NOTE:** When a schedule for a flow is triggered, all of recipes to generate the scheduled destinations are executed. Manual destinations are not generated. You cannot create schedules for individual outputs.

For more information on the scheduling objects, see *Object Overview*.

## Limitations

- One schedule cannot be applied to multiple flows.
- You cannot create separate schedules for individual recipes within a flow. A schedule defined at the flow level applies to all recipes in the flow.
- Only a flow owner can create or modify a flow's schedule.

## Data Management

**NOTE:** Since scheduled destinations are re-populated with each scheduled execution, you must determine how you wish to manage the data that is published to each location. Data management should be done outside of Designer Cloud powered by Trifacta® Enterprise Edition.

- **Import:** Before each scheduled execution, you should refresh the source of the imported dataset with new data outside of Designer Cloud powered by Trifacta Enterprise Edition.
- **Execution:** Please verify that the publishing settings for your scheduled destination are consistent with how you are using the results. For example, if the scheduled destination creates a new file with the same name for each execution (replace), you must move the generated file out of the output location before the next scheduled execution.
- **Output:** You must collect the generated results. While you can export the job's results through the Jobs page, you may find it easier to use an external scheduler to gather the results and forward to the downstream consumer of them.

## Flows for scheduling

**Tip:** When a schedule is executed, all outputs in a flow are generated, even if they are unused. For better performance on larger flows, you can create a separate flow that contains only the references back to the objects in the source flow that you wish to have scheduled. As an additional benefit, this separation keeps development and scheduled execution in separate flows.

## Schedule a Job

Schedules and scheduled destinations are defined through Flow View.

**Tip:** You can create schedules for datasets with parameters and apply overrides through Flow View at runtime. See *Flow View Page*.

For more information, see *Schedule a Job*.

## Job Execution

### Tracking

You can monitor a scheduled job like any other job in the application. See *Jobs Page*.

### Configure

See *Configure Automator*.

### via API

Not supported in this release.

# Overview of Parameterization

## Contents:

- *Environment Parameters*
    - *Limitations*
    - *Example - parameterized bucket names*
    - *Export and Import*
  - *Datasets with Parameters*
    - *Example*
    - *Parameter Types*
    - *Guidelines for Sources*
    - *Mismatched Schemas*
    - *Limitations*
    - *Creating Dataset with Parameters*
    - *Managing Datasets with Parameters*
  - *Flow Parameters*
    - *Limitations*
    - *Example*
    - *Upstream flow parameters*
    - *Creating flow parameters*
    - *Managing flow parameters*
    - *Flow parameters in plans*
  - *Output Parameters*
    - *Parameter Types*
    - *Example*
    - *Creating output parameters*
    - *Using output parameters*
  - *Bucket Name Parameters*
  - *Parameter Overrides*
  - *Order of Parameter Evaluation*
  - *Run Jobs with Parameters*
    - *Runtime Parameter Overrides*
    - *Scheduling jobs on datasets with parameters*
  - *Operationalization with Parameters*
    - *APIs*
  - *Configuration*
    - *Disable*
- 

In Designer Cloud powered by Trifacta® Enterprise Edition, **parameterization** enables you to apply dynamic values to the data that you import and that you generate as part of job execution.

## Parameter types:

- **Environment Parameters:** A workspace administrator or project owner can specify parameters that are available across the environment, including default values for them.
- **Dataset Parameters:** You can parameterize the paths to inputs for your imported datasets, creating datasets with parameters. For file-based imported datasets, you can parameterize the bucket where the source is stored.
- **Flow Parameters:** You can create parameters at the flow level, which can be referenced in any recipe in the flow.
- **Output Parameters:** When you run a job, you can create parameters for the output paths for file- or table-based outputs.

These parameters can be defined by timestamp, patterns, wildcards, or variable values that you specify at runtime.

## Environment Parameters

Project owners or workspace administrators can define parameters that apply across the project or workspace environment. These parameters can be referenced by any user in the environment, but only a user with admin access can define, modify, or delete these parameters.

**Tip:** Environment parameters are a useful means of ensuring that all users of the project or workspace share common reference values to buckets, output locations, and more. Environment parameter definitions can be exported and then imported into other projects or workspaces to ensure commonality across the enterprise. The values assigned to environment parameters can be modified after they have been imported into a new project or workspace.

**NOTE:** You must have admin access to the project or workspace to define environment parameters.

- Names of environment parameters must begin with `env.`

### Limitations

- You cannot use environment parameters in recipes.
- You cannot use environment parameters in plans.
- Environment parameter names are unique within the environment.
- You cannot use environment parameters in Deployment Manager. For more information, see *Overview of Deployment Manager*.

### Example - parameterized bucket names

In this example, you have three Trifacta workspaces, each of which has a different set of resources, although the only difference between them is the name of the S3 bucket in which they are stored:

| Environment Name | S3 Bucket Name |
|------------------|----------------|
| Dev              | myco-s3-dev    |
| Test             | myco-s3-test   |
| Prod             | myco-s3-prod   |

In your Dev workspace, you can create an environment parameter called the following:

```
env.bucket-source
```

The default value for this parameter is set to:

```
myco-s3-dev
```

When creating imported datasets in this workspace, you insert the environment parameter for the source bucket for each one.

**For your Test and Prod environments:**

1. Export your environment parameters from Dev.
2. Import them into Test and Prod. During import, the importing user can map the imported parameters to existing parameters in the environment.
3. In the imported environments, an administrator can manage the imported parameters and values as needed.

When you later export your flows from Dev and move them to Test and Prod, the imported flows automatically connect to the correct bucket for the target environment, since the bucket name is referenced by an environment parameter.

## Export and Import

You can export environment parameters from one environment and import them to another. For example, you may be building your flows in a Dev workspace before they are exported and imported into a Prod workspace. If your flows make use of environment parameters from the Dev space, you may want to export the parameters and their values from the Dev workspace for migration to the Prod workspace.

**NOTE:** As part of the import process, you must reconcile name conflicts between imported environment parameters and the parameters that already exist in the workspace.

For more information, see *Manage Environment Parameters*.

## Datasets with Parameters

In some cases, you may need to be able to execute a recipe across multiple instances of identical datasets. For example, if your source dataset is refreshed each week under a parallel directory with a different timestamp, you can create a variable to replace the parts of the file path that change with each refresh. This variable can be modified as needed at job runtime.

### Example

Suppose you have imported data from a file system source, which has the following source path to weekly transactions:

```
<file_system>:///source/transactions/2018/01/29/transactions.csv
```

In the above, you can infer a date pattern in the form of 2018/01/29, which suggests that there may be a pattern of paths to transaction files. Based on the pattern, it'd be useful to be able to do the following:

- Import data from parallel paths for other weeks' data.
- Sample across all of the available datasets.
- Execute jobs based on runtime variables that you set for other transaction sets fitting the pattern.
- Pass in parameterized values through API to operationalize the execution of jobs across weeks of transaction data.

In this case, you would want to parameterize the date values in the path, such that the dynamic path would look like the following:

```
<file_system>:///source/transactions/YYYY/MM/DD/transactions.csv
```

The above example implements a Datetime parameter on the path values, creating a **dataset with parameters**.

## Parameter Types

You can use the following types of parameters to create datasets with parameters:

- **Datetime parameters:** Apply parameters to date and time values appearing in source paths.
  - When specifying a Datetime parameter, you must also specify a **range**, which limits the range of the Datetime values.
- **Variables:** Define variable names and default values for a dataset with parameters.
  - Variable parameters can be applied to elements of the source path or to the bucket name, if applicable.
  - Modify these values at runtime to parameterize execution.
- **Pattern parameters:**
  - Wildcards: Apply wildcards to replace path values.
  - Regular Expressions: You can apply regular expressions to specify your dataset matches. Please see the limitations section below for more information.
  - Patterns : The platform supports a simplified means of expressing patterns.
    - For more information on Patterns , see *Text Matching*.

For more information, see *Create Dataset with Parameters*.

## Guidelines for Sources

The source files or tables for a dataset with parameters should have consistent structures. Since the sources are parsed with the same recipe or recipes, variations in schema could cause breakages in the recipe or initial parsing steps, which are applied based on the schema of the first matching source.

**NOTE:** All datasets imported through a single parameter are expected to have exactly matching schemas. For more information on variations, see *Mismatched Schemas* below.

**Tip:** If there have been changes to the schema of the sources of your dataset with parameters, you can edit the dataset and update the parameters. See *Library Page*.

Parameters in paths for imported datasets are rendered as regular expressions. Depending on the number of parameters and the comparative depth of them in a parameterized dataset, the process of performing all pattern checks can grow large, impacting import performance.

**Tip:** When specifying an imported dataset with parameters, you should attempt to be as specific as possible in your parameter definitions.

## Mismatched Schemas

Designer Cloud powered by Trifacta Enterprise Edition expects that all datasets imported using a single parameter have schemas that match exactly. The schema for the entire dataset is taken from the first dataset that matches for import.

If schemas do not match:

- When the first dataset contains extra columns at the end, the subsequent datasets that match should import without issues.
- If the subsequent datasets contain extra columns at the end, the datasets may import. Depending on the situation, there may be issues.
- If the subsequent datasets have additional or missing columns in the middle of the dataset, results of the import are unpredictable.
  - If there are extra columns in the middle of the dataset, you may see extra data in the final column, in which the spill-over data has not been split.

- Ideally, you should fix these issues in the source of the data. But if you cannot, you can try the following:

### Tips:

- After import of a dataset with parameters, perform a full scan random sample. When the new sample is selected:
  - Check the last column of your imported to see if you have multiple columns of data. See if you can perform split the columns yourself.
  - Scan the column histograms to see if there are columns where the number of mismatches or anomalous or outlier values has suddenly increased. This could be a sign of mismatches in the schemas.
- Edit the dataset with parameters. Review the parameter definition. Click **Update** to re-infer the data types of the schemas. This step may address some issues.
- You can use the union tool to import the oldest and most recent sources in your dataset with parameters. If you see variations in the schema, you can look to modify the sources to match.
  - If your sources have variation in structure, you should remove the structure from the imported dataset and create your own initial parsing steps to account for the variations. See *Initial Parsing Steps*.

### Limitations

- You cannot create datasets with parameters from uploaded data.
- You cannot create dataset with parameters from multiple file types.
  - File extensions can be parameterized. Mixing of file types (e.g. TXT and CSV) only works if they are processed in an identical manner, which is rare.
  - You cannot create parameters across text and binary file types.
- Parameter and variable names can be up to 255 characters in length.
- For regular expressions, the following reference types are not supported due to the length of time to evaluate:
  - Backreferences. The following example matches on `axa`, `bx`, and `cxc` yet generates an error:
 

```
([a-c])x\1
```
  - Lookahead assertions: The following example matches on `a`, but only when it is part of an `ab` pattern. It generates an error:
 

```
a(?:=b)
```
- For some source file types, such as Parquet, the schemas between source files must match exactly.
- You cannot define import mapping rules for datasets with parameters. If the imported dataset with parameters is still accessible, you should be able to run jobs from it.

## Creating Dataset with Parameters

### From file system

When browsing for data on your default storage layer, you can choose to parameterize elements of the path. Through the Import Data page, you can select elements of the path, apply one of the supported parameter types and then create the dataset with parameters.

**NOTE:** Matching file path patterns in a large directory can be slow. Where possible, avoid using multiple patterns to match a file pattern or scanning directories with a large number of files. To increase matching speed, avoid wildcards in top-level directories and be as specific as possible with your wildcards and patterns.

**Tip:** If your imported dataset is stored in a bucket, you can parameterize the bucket name, which can be useful if you are migrating flows between environments or must change the bucket at some point in the future.

For more information, see *Create Dataset with Parameters*.

### From relational source

If you are creating a dataset from a relational source, you can apply parameters to the custom SQL that pulls the data from the source.

**NOTE:** Avoid using parameters in places in the SQL statement that change the structure of the data. For example, within a SELECT statement, you should not add parameters between the SELECT and FROM keywords.

For more information, see *Create Dataset with SQL*.

### Matching parameters

When a dataset with parameters is imported for use, all matching source files or tables are automatically unioned together.

**NOTE:** Sources for a dataset with parameters should have matching schemas.

The initial sample that is loaded in the Transformer page is drawn from the first matching source file or table. If the initial sample is larger than the first file, rows may be pulled from other source objects.

## Managing Datasets with Parameters

### Datasets with parameters in your flows

After you have imported a dataset with parameters into your flow:

- You can review any parameters that have been applied to the dataset through the Parameterization in Flow view.
- When the dataset with parameters is selected, you can use the right panel to review and edit the parameters that are applied to it.
- You can override the default value applied to the parameter through Flow View. See *Manage Parameters Dialog*.

For more information, see *Flow View Page*.

**Tip:** You can review details on the parameters applied to your dataset. See *Dataset Details Page*.

### Sampling from datasets with parameters

When a dataset with parameters is first loaded into the Transformer page, the initial sample is loaded from the first found match in the range of matching datasets. If this match is a multi-sheet Excel file, the sample is taken from the first sheet in the file.

### With parameters:



To work with data that appears in files other than the first match in the dataset, you must create a new sample in the Transformer page. Any sampling operations performed within the Transformer page sample across all matching sources of the dataset.

### With variables:

If you have created a variable with your dataset, you can apply a variable value to override the default at sampling time. In this manner, you can specify sampling to occur from specific source files from your dataset with parameters.

For more information, see *Overview of Sampling*.

### Scheduling for datasets with parameters

Schedules can be applied to a dataset with parameters. When resolving date range rules for scheduling a dataset with parameters, the schedule time is used.

For more information, see *Add Schedule Dialog*.

### Sharing for datasets with parameters

By default, when a flow containing parameters is copied, any changes to parameter values in the copied flow also affect parameters in the original flow. To separate these parameters, you have the following options:

1. Optionally, when the flow is copied, you can copy the underlying datasets.
2. As a workaround, you can export and import the flow into the same system and replace the datasets in the imported flow.

**NOTE:** For copying flows using parameterized datasets, you should duplicate the datasets, which creates separate copies of parameters and their values in the new flow. If datasets are not copied, then parameter changes in the copied flow modify the values in the source flow.

For more information, see *Overview of Sharing*.

### Housekeeping

Since Designer Cloud powered by Trifacta Enterprise Edition never touches the source data, after a source that is matched for a dataset with parameters has been executed, you should consider removing it from the source system or adjusting any applicable ranges on the matching parameters. Otherwise, outdated data may continue to factor into operations on the dataset with parameters.

**NOTE:** Housekeeping of source data is outside the scope of Designer Cloud powered by Trifacta Enterprise Edition. Please contact your IT staff to assist as needed.

## Flow Parameters

You can specify flow parameters and their default values, which can be invoked in the recipe steps of your flow. Wherever the flow parameter is invoked, it is replaced by the value you set for the parameter. Uses:

- Dynamically affect recipe steps
- Improve flow usability; build fewer flows and recipes to maintain
- Parameters are evaluated at design time in the Transformer page and at runtime during job execution
- All parameter values can be overridden, as needed.

### Flow parameter types:

- Literal values: These values are always of String data type.

**Tip:** You can wrap flow parameter references in your transformations with one of the `PARSE` functions. For more information, see *Create Flow Parameter*.

**NOTE:** Wildcards are not supported.

- Patterns . For more information, see *Text Matching*.
- Regular expressions.

## Limitations

- Flow parameters are converted to constants in macros. Use of the macro in other recipes results in the constant value being applied.
- A flow parameter cannot be used in some transformation steps or fields.

## Example

Suppose you need to process your flow across several regions of your country. These regions are identified using a region ID value: `pacific`, `mountain`, `central`, or `eastern`.

From the Flow View context menu, you select **Manage parameters**. In the Parameters tab, you specify the parameter name:

```
paramRegion
```

You must specify a default value. To verify that this critical parameter is properly specified before job execution, you set the default value to:

```
##UNSPECIFIED##
```

The above setting implies two things:

- If the above value appears in the output, then an override value for the parameter was not specified when the job was executed, which prevents the default value being used erroneously.
- Before the job is executed, you must specify an override value. You can specify an override:
  - At the flow level to assist in recipe development.
  - At run time to insert the proper region value for the job run.

After the flow parameter has been created, you can invoke it in a transformation step using the following syntax.

```
$paramRegion
```

Where the parameter is referenced, the default or applicable override value is applied. For more examples, see *Create Flow Parameter*.

## Upstream flow parameters

If your flow references a recipe or dataset that is sourced from an upstream flow, the flow parameters from that flow are available in your current flow. That value of the parameter at time of execution is passed to the current flow.

**NOTE:** Downstream values and overrides of parameters that share the same name take precedence. When you execute the downstream flow, the parameter value is applied to the current flow and to all upstream objects. For more information, see "Order of Evaluation" below.

## Creating flow parameters

Flow parameters are created at the flow level from the context menu in Flow View. See *Manage Parameters Dialog*.

## Managing flow parameters

Flow parameters can be edited, deleted, and overridden through the Flow View context menu. See *Manage Parameters Dialog*.

## Flow parameters in plans

You can also apply overrides to your flow parameters as part of your plan definition. For more information, see *Plan View Page*.

## Output Parameters

You can specify variable and timestamp parameters to apply to the file or table paths of your outputs.

**NOTE:** Output parameters are independent of dataset parameters.

## Parameter Types

You can create the following types of output parameters:

- **Datetime parameters:** Insert date and time values in output paths based on the job's start time.
- **Variables:** Define variable names and default values for an output parameter. Modify these values at runtime to parameterize execution.

**Tip:** These types of parameters can be applied to file or table paths. An output path can contain multiple parameters.

## Example

Suppose you are generating a JSON file as the results of job execution.

```
/outputs/myFlow/myOutput.json
```

Since this job is scheduled and will be executed on a regular interval, you want to insert a timestamp as part of the output, so that your output filenames are unique and timestamped:

```
/outputs/myFlow/myOutput_<timestamp>.json
```

In this case, you would create an output parameter of timestamp type as part of the write settings for the job you are scheduling.

## Creating output parameters

When you are creating or editing a publishing action in the Run Jobs page, you can click the **Parameterize destination** link that appears in the right panel.

**Tip:** For outputs that are stored in buckets, you can parameterize the name of the bucket.

---

For more information, see *Create Outputs*.

## Using output parameters

Whenever you execute a job using the specified publishing action, the output parameters are applied.

After specifying variable parameters, you can insert new values for them at the time of job execution in the Run Job page.

For more information, see *Run Job Page*.

## Bucket Name Parameters

In addition to parameterizing the paths to imported datasets or outputs, you can also apply parameters to the buckets where these assets are stored. For example, if you are developing flows in one workspace and deploying them into a production workspace, it may be useful to create a parameter for the name of the bucket where outputs are written for the workspace.

Bucket names can be parameterized for the buckets in the following datastores:

- S3
- ADLS Gen2

**Tip:** You can parameterize the user info, host name, and path value fields as separate parameters.

Bucket names can be parameterized as variable parameters or as environment parameters. For more information on examples of parameterized bucket names, see "Environment Parameters" above.

For more information:

- *Parameterize Files for Import*
- *Create Outputs*

## Parameter Overrides

For each of the following types of parameter, you can apply override values as needed.

| Override Type      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| dataset parameters | When you run a job, you can apply override values to variables for your imported datasets. See <i>Run Job Page</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| flow parameters    | <p>At the flow level, you can apply override values to flow parameters. These values are passed into the recipe and the rest of the flow for evaluation during recipe development and job execution.</p> <div><b>NOTE:</b> Overrides applied at the flow level are passed into all recipes and other objects in the flow. Wherever there is case-sensitive match between the name of the overridden parameter and a parameter name in the flow, the override value is applied. These values can be overridden by ad-hoc values. See "Order of Precedence" below.</div> |
| output parameters  | When you define your output objects in Flow View, you can apply override values to the parameterized output paths on an as-needed basis when you specify your job settings. See <i>Run Job Page</i> .                                                                                                                                                                                                                                                                                                                                                                  |

## Order of Parameter Evaluation

Wherever a parameter value or override is specified in the following list, the value is applied to all matching parameters within the execution tree. Suppose you have created a parameter called `varRegion`, which is referenced in your imported dataset, recipe, and output object. If you specify an override value for `varRegion` in the Run Job page, that value is applied to the data you import (dataset parameter), the recipe during execution (flow parameter), and the path of the output that you generate (output parameter). Name matches are case-sensitive.

**NOTE:** Override values are applied to upstream flows, as well. Any overrides specified in the current flow are passed to downstream flows, where they can be overridden as needed.

Parameter values are evaluated based on the following order of precedence (highest to lowest):

**NOTE:** The following does not apply to environment parameters, which cannot be overridden.

1. **Run-time overrides:** Parameter values specified at run-time for jobs.

**NOTE:** The override value is applied to all subsequent operations in the platform. When a job is submitted to the job queue, any overrides are applied at that time. Changes to override values do not affect jobs that are already in flight.

**NOTE:** You can specify run-time override values when executing jobs through the APIs. See *API Workflow - Run Job*.

See *Run Job Page*.

2. **Flow level overrides:** At the flow level, you can specify override values, which are passed into the flow's objects. These values can be overridden by overrides set in the above locations. See *Manage Parameters Dialog*.
3. **Default values:** If no overrides are specified, the default values are applied:
  - a. Imported datasets: See *Create Dataset with Parameters*.
  - b. Flow parameters: See *Manage Parameters Dialog*.
  - c. Output parameters: See *Run Job Page*.
4. **Inherited (upstream) values:** Any parameter values that are passed into a flow can be overridden by any matching override specified within the downstream flow.

## Run Jobs with Parameters

When running a job based on datasets with parameters, results are written into separate folders for each parameterized path.

**NOTE:** During job execution, a canary file is written for each set of results to validate the path. For datasets with parameters, if the path includes folder-level parameterization, a separate folder is created for each parameterized path. During cleanup, only the canary files and the original folder path are removed. The parameterized folders are not removed. This is a known issue.

## Runtime Parameter Overrides

When you choose to run a job on a dataset with parameters from the user interface, any variables are specified using their default values.

Through the Run Job page, you can specify different values to apply to variables for the job.

**NOTE:** Override values applied to a job are not validated. Invalid overrides may cause your job to fail.

**NOTE:** Values applied through the Run Job page to variables override the default values for the current execution of the job. Default values for the next job are not modified.

**NOTE:** When you edit an imported dataset, if a variable is renamed, a new variable is created using the new name. Any override values assigned under the old variable name for the dataset must be re-applied. Instances of the variable and override values used in other imported datasets remain unchanged.

For more information, see *Run Job Page*.

In the Job Details page, click the Parameters tab to view the parameter names and values that were used as part of the job, including the list of matching datasets. See *Job Details Page*.

## Scheduling jobs on datasets with parameters

You can schedule jobs for datasets with parameters.

**NOTE:** When a job is executed, the expected time of execution is used during execution. For scheduled jobs, this value is the scheduled time. For example, if a job scheduled for 08:00 begins execution at 08:05, any parameters that reference "now" time use 08:00 during the job run.

For a scheduled job:

- Parameter values are evaluated based on the scheduled time of execution. Relative times are evaluated based on the scheduled time of execution.
- If there are interruptions in service due to maintenance windows or other reasons, scheduled jobs are queued for execution on restart. These queued jobs are attempted only once.

See *Schedule a Job*.

## Operationalization with Parameters

### APIs

Through the API, you can apply runtime parameters to datasets with parameters during job execution. For more information, see <https://api.trifacta.com/ee/es.t/index.html#operation/runJobGroup> For more information on working with parameters and the APIs, see *API Workflow - Run Job on Dataset with Parameters*.

Use of parameters to create imported datasets through the API is not supported.

For other parameter types, you can apply overrides as key-value pairs in the API request to execute a new job. See *API Workflow - Run Job*.

## Configuration

### Disable

By default, parameterization is enabled. This feature is covered by this setting: **Parameterization**.

For more information on disabling, see *Workspace Settings Page*.

# Overview of Authorization

## Contents:

- *Account Roles*
    - *Member role*
    - *Admin role*
  - *Resource Roles and Privileges*
    - *Standard roles*
    - *Privileges*
  - *Example model*
- 

**Authorization** governs how workspace members can access platform features and user-defined objects in the workspace.

**NOTE:** Authorization manages access to object types. It does not cover access to individual objects of a specified type. For example, access to a specific flow is governed by ownership of the flow (owner) and sharing of the flow by the owner (to a collaborator). If a flow is shared with a user who is not permitted to access flows, then the user cannot access the flow.

## Account Roles

In a workspace, a member can have one of the following roles:

### Member role

The Member role enables the user to access all product functionality that is enabled within the workspace for the product edition.

### Admin role

The Admin role enables all capabilities of the Member role, plus:

- access to all workspace objects, unless specifically limited. See Resource Roles and Privileges below.
- administration functions and settings for the workspace.

**NOTE:** The workspace Admin role is a super-user. It should be granted on a limited basis.

**NOTE:** A platform administrator is automatically granted the workspace Admin role.

## Resource Roles and Privileges

Access to workspace objects is governed by roles in the user account.

- A **role** is a set of zero or more privileges.
- A **privilege** is an access level for a type of object in the workspace.



- A user may have one or more roles in it.

**NOTE:** Roles are additive. If a user has multiple roles, the user has access at the highest level of privileges from each role.

- All accounts are created with the `default` role.

## Standard roles

### default role

All new users are automatically assigned the `default` role. By default, the `default` role enables full access to all workspace objects.

- If you have upgraded from a version of the product that did not support authorization, the `default` role represents no change in behavior. All existing users can access workspace objects as normal.

Since roles in a user account are additive, you may choose to reduce the privileges on the `default` role and then add privileges selectively by creating other roles and assigning them to users. See the example below.

**NOTE:** You can modify the `default` role. You can also remove it from a user account. You cannot delete the role.

**NOTE:** In future releases of the software, additional workspace objects may be made available. A level of access may be defined in the `default` role. No other roles will be modified.

### Workspace admin role

The Workspace admin role is a super-user.

**NOTE:** This role enables for the user owner-level access to all objects in the workspace and access to all admin-level settings and configuration pages in the admin console. This role should not be assigned to many users. At least one user should always have the `Workspace admin` role.

## Privileges

For a complete list of privileges for each type of object, see *Privileges and Roles Reference*.

### Example model

In the following model, three separate roles have been created. Each role enables the highest level of access to a specific type of workspace object.

The `default` object has been modified:

- Since all users are automatically granted the `default` role, the scope of its permissions has been reduced here to view-only.
- There is no `viewer` privilege for Plans ( `none`, `author`).

**NOTE:** Depending on your product edition, some of these privileges may not be applicable.

| Privilege/Role | default | Role A | Role B | Role C | Notes                         |
|----------------|---------|--------|--------|--------|-------------------------------|
| Flows          | viewer  | author | none   | none   |                               |
| Connections    | viewer  | none   | author | none   | Paid product editions only    |
| Plans          | none    | none   | none   | author | Premium product editions only |

### User 1:

Roles: default

- User can see flows in Flows page. User cannot schedule, modify, or create new ones.
- User can see connections in the Connections page. User cannot schedule, modify, or create new ones.
- User cannot access the Plans page.

### User 2:

Roles: default, Role A

- User can create, schedule, modify, run jobs, and delete flows (full privileges).
- User can see connections in the Connections page. User cannot schedule, modify, or create new ones.
- User cannot access the Plans page.

### User 3:

Roles: Role A, Role B, Role C

- User can create, schedule, modify, run jobs, and delete flows (full privileges).
- User can create, modify, and delete connections (full privileges).
- User can create, schedule, modify, run jobs, and delete plans (full privileges).

# Overview of Operationalization

## Contents:

- *Single-Flow Operations*
    - *Parameterization*
    - *Scheduling*
    - *Job monitoring*
    - *Email notifications*
    - *Webhooks*
    - *Deployment Manager*
  - *Orchestration*
    - *Terms*
    - *Task types*
    - *Limitations*
    - *Basic workflow*
    - *Plan scheduling*
    - *Plan execution*
    - *Enable*
    - *Logging*
- 

**Operationalization** refers to a general class of platform features that enable repeated application of Designer Cloud powered by Trifacta® Enterprise Edition on production data. Whether deployed in a single flow or across all flows in your environment, operationalization features broaden the scope of wrangled data, simplify job execution, and enable these processes on a repeated or scheduled basis.

In the following sections, you can review short summaries of specific features and explore more detailed information on them.

## Single-Flow Operations

These features can be applied to individual flows to simplify job execution.

### Parameterization

**Parameterization** enables you to specify parameters that capture variability in your data source paths or names. For example, you can parameterize the names of folders in your filepaths to capture files within multiple folders. Or, you can parameterize your inputs to capture datasets named within a specific time range. Nested folders of data can be parameterized, too.

Parameter types:

- dataset parameters: Parameterize the input paths to your data, allowing you to process data in parallel files and tables through the same flow.
- output parameters: Parameterize the output paths for your results.
- flow parameters: Define parameters that can be applied in your flows, including recipe steps.

**Tip:** You can apply overrides to any parameter at the flow level. These parameter override values are applied to any parameter that is referenced within the flow for any supported parameter type.

Parameter formats:

**NOTE:** Some of the following may not be available in your product edition.

| Parameter Type | Description                                                                                                 |
|----------------|-------------------------------------------------------------------------------------------------------------|
| Pattern        | Use regular expressions or Patterns in your paths or queries to sources to capture a broader set of inputs. |
| Wildcard       | Replace parts of your paths or queries with wildcards.                                                      |
| Datetime       | You can specify parameterized Datetime values in one of the supported formats.                              |
| Variable       | Variable values can be specified as overrides during import, job execution, and output.                     |

Parameterization is available for the following:

#### File systems

| Input     | Output    |
|-----------|-----------|
| Date/time | Timestamp |
| Pattern   | Variable  |
| Variable  |           |

#### Relational sources

| Input     | Output    |
|-----------|-----------|
| Timestamp | Timestamp |
| Variable  | Variable  |

**NOTE:** For relational data, parameterization is applied to custom SQL queries used to import the data. For more information, see *Enable Custom SQL Query*.

For more information, see *Overview of Parameterization*.

#### Scheduling

The scheduling feature, also known as **Automator**, enables you to schedule the execution of individual flows on a specified frequency. Frequencies can be specified through the Designer Cloud application through a simple interface or, if needed, in a modified form of cron syntax.

**Tip:** Automator is often used with parameterization to fully automate data preparation processes in Designer Cloud powered by Trifacta Enterprise Edition.

For more information, see *Overview of Automator*.

#### Job monitoring

After a job has been launched, detailed monitoring permits you to track the progress of your job during all phases of execution. Status, job stats, inputs, outputs and a flow snapshot are available through the Designer Cloud application. For more information, see *Overview of Job Monitoring*.

#### Email notifications

After a job has completed, you can send email notifications to stakeholders based on the success or failure of the job.

**NOTE:** This feature must be enabled. See *Workspace Settings Page*.

These notifications are defined within Flow View. See *Email Notifications Page*.

## Webhooks

Webhook notifications let you define outgoing HTTP messages to any REST API. The message form and body can be customized to include job execution metadata. For more information, see *Create Flow Webhook Task*.

## Deployment Manager

The **Deployment Manager** is a separate environment that can be enabled for the execution of production flows under limited access. Flows in development are exported from your default (Dev) instance and then imported to the Production instance, the Deployment Manager, where you can configure the periodic execution of the flow. For more information, see *Overview of Deployment Manager*.

## Orchestration

**Orchestration** is a set of functionality that supports the scheduled execution of jobs across multiple flows. These jobs could be external processes, other flows, or even HTTP requests.

## Terms

| Term     | Description                                                                                                                                                                                                                  |
|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| plan     | A <b>plan</b> is a sequence of tasks that are executed on one or more flows to which you have access. To orchestrate tasks, you build a plan. A plan can be scheduled for execution, triggered manually, or invoked via API. |
| trigger  | A task is executed based on a trigger. A <b>trigger</b> is a condition under which a task is executed. In many cases, the trigger for a task is based on the schedule for the plan.                                          |
| task     | A <b>task</b> is a unit of execution in the platform. For example, one type of task is the execution of a flow, which executes all recipes in the flow, as well as the flow's upstream dependencies.                         |
| snapshot | A <b>snapshot</b> of the plan is captured, and the plan is executed against this snapshot. For more information on snapshots, see "Plan execution" below.                                                                    |

## Task types

The following types of tasks are available.

| Type       | Description                                                                                                           |
|------------|-----------------------------------------------------------------------------------------------------------------------|
| flow task  | An ad-hoc or scheduled execution of the transformations required to produce one or more selected outputs from a flow. |
| HTTP task  | A request submitted to a third-party server as part of a plan run.                                                    |
| Slack task | Send a message with information about the plan run to a specified Slack channel.                                      |

## Limitations

- You cannot specify parameter overrides to be applied to plans specifically.
  - Plans inherit parameter values from the objects referenced in the plan's tasks.
  - If overrides are applied to flow parameters, those overrides are passed to the plan at the time of flow execution.

**Tip:** Prior to plan execution, you can specify parameter overrides at the flow level. These values are passed through to the plan for execution. For more information, see *Manage Parameters Dialog*.

**For this release:**

- The only type of plan task that is supported is Run Flow.
- Tasks are defined in a linear, non-branching sequence based on the trigger.

**Basic workflow**

You create a plan and schedule it using the following basic workflow.

1. Create the plan. A **plan** is the container for definition of the tasks, triggers, and other objects. See *Plans Page*.
2. In Plan View, you specify the objects that are part of your plan. See *Plan View Page*.
  - a. **Schedule:** The schedule defines the set of triggers that queue the plan for execution.
    - i. **Trigger:** A trigger defines the schedule and frequency at which the plan is executed. A plan can have multiple triggers (e.g. monthly versus weekly executions).
  - b. **Task(s):** Next, you specify the tasks that are executed in order.
    - i. **Flow task:** A flow task includes the specification of the flow to run and the outputs from the flow to generate.

**NOTE:** You can select the outputs from the recipe that you wish to generate. You do not need to generate all outputs.

**NOTE:** When a flow task is executed, the execution plan works back from the selected outputs to execute all of the recipes required to generate the output, including the upstream dependencies of those recipes.

See *Plan View for Flow Tasks*.

- ii. **HTTP task:** An HTTP task is a request issued when it is triggered from the application to a target endpoint. This request supports a variety of API methods. See *Plan View for HTTP Tasks*.
  - iii. **Slack task:** A Slack task is a message between Designer Cloud powered by Trifacta Enterprise Edition and a specified Slack channel that is triggered within the plan. See *Plan View for Slack Tasks*.
  - iv. Continue building tasks in a sequence.
3. As needed, you can apply override values to any flow parameters that are included in the tasks of your recipe. These overrides are applied during a plan run. For more information, see *Manage Parameters Dialog*.
  4. To test:
    - a. Click **Run now**.
    - b. To track progress, click the Runs link.
    - c. In the Run Details page, you can track the progress.
    - d. The first task is executed and completes, before the second task is started.
    - e. Individual tasks are executed as separate jobs, which you can track through the Jobs page. See *Jobs Page*.
    - f. When the plan has completed, you can verify the results through the Job details page. See *Job Details Page*.
  5. If you are satisfied with the plan definition and your test run, the plan will execute according to the scheduled trigger.

## Plan scheduling

Through the Plan View page, you can configure the scheduled executions of the plan. Plan schedules are defined using triggers.

- These schedules are independent of schedules for individual flows.
- You cannot create schedules for individual tasks.

## Plan execution

When a plan is triggered for execution, a **snapshot** of the plan is taken. This snapshot is used to execute the plan. Tasks are executed in the sequence listed in Plan View.

### Important notes:

**NOTE:** Any subsequent changes to the flows, datasets, recipes, and outputs referenced in the plan's tasks can affect subsequent executions of the plan. For example, subsequent removal of a dataset in a flow referenced in a task can cause the task to fail to execute properly.

At the flow level, you can define webhooks and email notifications that are triggered based on the successful generation of outputs. When you execute a plan containing an output with one of these messages, the message is triggered and delivered to stakeholders.

**NOTE:** Webhook messages and email notification cannot be directly triggered based on a plan's execution. However, you can create HTTP-based tasks to send messages based on a plan task's execution.

**Tip:** When a flow email notification is triggered through a plan, the internal identifier for the plan is included in the email.

See "Webhooks" and "Email notifications" above.

## Enable

Enable the following setting:

```
Plans feature
```

Plan sharing, import, and export must also be enabled.

For more information, see *Workspace Settings Page*.

The following flags must be enabled for the orchestration service to correctly function.

### Steps:

1. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
2. Locate the following settings. Verify that they are set to `true`:

```
"webapp.orchestrationWorkers.enabled": true,
"orchestration-service.enabled": true,
"orchestration-service.autoRestart": true,
```

3. You can choose to enable the following task types:

| Task type  | Setting                                                                         | Description                                                                                                                                                                                          |
|------------|---------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| HTTP task  | <code>feature.orchestration.httpTasks.enabled</code>                            | When <code>true</code> , you can configure plan tasks to deliver a REST request over HTTP or HTTPS to a specified endpoint, including endpoints in the Designer Cloud powered by Trifacta platform . |
| Slack task | <code>feature.orchestration.slackTask.enabled</code>                            | When <code>true</code> , you can configure plan tasks to deliver messages to a specified Slack channel.                                                                                              |
| Email task | <code>feature.orchestration.emailTask.enabled</code>                            | This feature is not yet available.                                                                                                                                                                   |
| Flow task  | This feature is automatically enabled when Plans feature is enabled. See above. | These tasks execute a specific output on a selected flow.                                                                                                                                            |

4. Save your changes and restart the platform.

## Logging

Logging information on plan execution is captured in the `orchestration-service.log`. This log file can be downloaded as part of a Support Bundle. For more information, see *Support Bundle Contents*.

You can configure aspects of how this log captures service events. For more information, see *Configure Logging for Services*.



# Overview of Macros

## Contents:

- *Limitations*
  - *Enable*
  - *Examples*
    - *Example 1 - Reformat headers*
    - *Example 2 - Redact data for sensitive column data types*
  - *Create*
    - *Macro inputs*
  - *Apply*
  - *Sharing*
  - *Import/Export*
  - *Manage*
- 

In Designer Cloud powered by Trifacta® Enterprise Edition, a **macro** is a saved sequence of one or more recipe steps that can be reused in other recipes. As needed, values in the recipe steps can be modified, so that instances of the macro can be configured for the recipe requirements.

## Limitations

- You cannot create macros from steps that contain the following:
  - Multi-dataset operations like join, union, and lookup
  - Data-dependent transformations like header, valuestocols, and pivot.
  - Other macros

**NOTE:** In macros, Rename Columns transformations do not work. This is a known issue.

- You cannot create macros in flows that you do not own.
- Macro input limitations on the following types:
  - limits
  - enums
  - arrays
- Sharing of macros is not supported.
  - When working with a flow that was shared with you, you can only use the macros that belong to the flow's owner.
- When a flow containing a macro is imported, the macro steps are expanded.

## Enable

This feature is enabled by default. To disable this feature, please complete the following steps.

### Steps:

1. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
2. Locate the following setting:

```
"feature.macros.enabled": true,
```

3. Export and import of macros is controlled by a separate setting. Locate the following setting and set it to true:

```
"feature.macros.exportable": true,
```

4. Save your changes and restart the platform.

## Examples

### Example 1 - Reformat headers

Suppose one of your downstream systems has the following requirements for column headers:

- No spaces. Underscore is ok.

You can do the following:

1. For the recipe on which you are working, create a new recipe.
2. In this new empty recipe, add the steps to configure your headers according to the above requirements.
  - a. No spaces. Underscores are ok:

|                                         |                                   |
|-----------------------------------------|-----------------------------------|
| <b>Transformation Name</b>              | Rename columns based on a pattern |
| <b>Parameter: Option</b>                | Find and replace                  |
| <b>Parameter: Columns</b>               | All                               |
| <b>Parameter: Find</b>                  | ' '                               |
| <b>Parameter: Replace with</b>          | '_'                               |
| <b>Parameter: Match all occurrences</b> | true                              |

3. Select the above step. In the context menu for it, select **Create or replace macro**.
  - a. Enter a Name and optional Description value. Click **Next**.
  - b. In the Create Macro dialog, you can review the step and its specified field values.
  - c. To save the macro, click **Save**.
4. For any recipe that must generate results for this downstream system, you can insert this macro as the last step before publication. For example, you can delete the recipe where you made the macro and insert the macro reference in the preceding recipe.

### Example 2 - Redact data for sensitive column data types

For security reasons, you may decide that sensitive information must be redacted before it is delivered as an output for downstream consumption. For the following data types, you may wish to remove the sensitive information at the end of your transformation process:

- Credit card numbers
- Social Security numbers

1. For the recipe on which you are working, create a new recipe.
2. In this new empty recipe, add the following steps.
  - a. Redact social security numbers:

|                            |              |
|----------------------------|--------------|
| <b>Transformation Name</b> | Edit formula |
| <b>Parameter: Columns</b>  | All          |

|                           |                                                                |
|---------------------------|----------------------------------------------------------------|
| <b>Parameter: Formula</b> | <code>IF( ISVALID(\$col, 'SSN'), '##REDACTED##', \$col)</code> |
|---------------------------|----------------------------------------------------------------|

- b. Redact credit card numbers: For this one, you can use the following transformation to mask the numbers except for the last four digits using Patterns :

|                                |                                                                                      |
|--------------------------------|--------------------------------------------------------------------------------------|
| <b>Transformation Name</b>     | Replace text or patterns                                                             |
| <b>Parameter: Columns</b>      | All                                                                                  |
| <b>Parameter: Find</b>         | <code>`{start}{digit}{4}{any}{digit}{4}{any}{digit}{4}{any}({digit}{4}){end}`</code> |
| <b>Parameter: Replace with</b> | <code>XXXX-XXXX-XXXX-\$1</code>                                                      |

**NOTE:** The above transformation matches values based on the structure of the data, instead of the data type. If for some reason, you have values that are not credit card numbers yet follow the credit card pattern, those values will be masked as well by this transformation.

3. Select the above steps. In the context menu, select **Create or replace macro** . For more information, see *Create or Replace Macro*.
  - a. Enter a Name and optional Description value. Click **Next**.
  - b. In the Create Macro dialog, you can review the step and its specified field values.
    - i. You may wish to parameterize the Find and Replace with values. For example, for some uses of the macro, you may wish to replace with an empty string or a value like ##REDACTED ## like the previous macro.
  - c. To save the macro, click **Save**.
4. For any recipe that must generate results for this downstream system, you can insert this macro as the last step before publication. For example, you can delete the recipe where you made the macro and insert the macro reference in the preceding recipe.

## Create

A macro is created from a sequence of steps inside a recipe.

- The steps do not have to occur consecutively in the recipe.
- Recipe steps are added to the macro in the order that they are listed in the recipe.
- Some recipe steps cannot be added to a macro, so the option to create a macro with these types of steps is not available.

For more information, see *Create or Replace Macro*.

## Macro inputs

When you create a macro, you can define macro inputs to contain values to be used in the macro's steps. Values for these inputs can be specified with each instance of a macro. For example, if you use MacroA at Step 2 and Step 37 of your recipe, you can specify different values for inputs to MacroA at the Step 2 and Step 37 instance of it.

- **Create macro inputs:** Macro inputs can be defined when you create a new macro.
- **Reuse or replace macro inputs:** When you replace a macro, you can reuse or replace the existing macro inputs in the new version of a macro.
  - If you are reusing the existing macro inputs, you must map them to the new steps in the new version of the macro.

- If you are replacing macro inputs, instances of the macro that were added to your recipes under the old definition must be updated.
- For more information, see *Create or Replace Macro*.

## Apply

After a macro is created, you can apply an instance of it anywhere in your recipes. See *Apply a Macro*.

## Sharing

Macros cannot be independently shared.

### Copy a flow:

All macros are included. Steps are not expanded.

### Share a flow:

When a flow is shared, the flow owner's macros are available for use by any collaborator in the recipes of the shared flow.

## Import/Export

**NOTE:** Exported macros can be imported into a release that is later than the source release of the product. Exported macros cannot be imported into earlier releases.

### Export:

- You can export individual macros from the Macros page. See *Export Macro*.
- When a flow containing a recipe that references macros is exported, macros are exported as expanded steps.

### Import:

- Exported macros can be imported into a new environment through the Macros page. See *Import Macro*.
- When a flow containing macros is imported, the expanded steps are imported normally.

## Manage

After macros have been created, you can manage them through the Library. For more information, see *Macros Page*.

# Overview of Deployment Manager

## Contents:

- *Dev/Test and Prod Deployments*
    - *Implementation in the platform*
    - *Licensing*
  - *Terminology*
    - *Production environment terminology changes*
    - *Deployment Objects*
  - *Enable Deployment*
    - *User management*
  - *Import/Export*
    - *Exported Flows*
    - *Connections*
    - *Import*
    - *Value and Object Mapping Rules*
  - *Production Environment*
    - *Version Management*
    - *Flow View Page*
  - *Example Workflow*
  - *Recommended Practices*
  - *Job Execution*
    - *On-demand jobs*
    - *Scheduled jobs*
  - *Automation*
- 

You can deploy flows that you have created into a separate, production environment where jobs for those flows can be executed on a periodic or scheduled basis. In this manner, you can create separation between your development and production environments and their flows. The **Deployment Manager** includes the tools to migrate your software between environments, manage releases of it, and separately control access to development and production flows.

- Deployment Manager enables the transfer of flows between development and production instances of the platform. A customer may have one or more instances of the platform.
- For managing user access to flows within the same development instance, you can use sharing. See *Overview of Sharing*.
- This feature was formerly known as, "deployment management."

## Key Features:

- Development environment:
  - Export of flows and all dependent objects
  - Import back into Development deployments for further development
- Production environment:
  - Import of flows
    - Import global and object-level mapping rules
  - Manage releases of flows
  - Rollback to previous versions as needed
- APIs to manage deployments

## Dev/Test and Prod Deployments

In a typical environment, deployments may be segmented between Development (Dev), Testing (Test), and Production (Prod) environments. With respect to the Designer Cloud powered by Trifacta platform , these deployments break down into the following:

**NOTE:** In some cases, Dev and Test may be the same instance.

**NOTE:** Multiple browser tabs or windows open to different versions of the product is not supported.

| Platform instance | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Development (Dev) | <p>New flows and recipes are created in a Development instance of the platform. Experiments can be undertaken without concern that production use of the recipe or flow is affected.</p> <div><p><b>Tip:</b> You should do all of your recipe development and testing in Dev/Test. Avoid making changes in a Prod environment.</p></div> <p>Rules should be established on how flows, datasets, and recipes are organized and structured. Where are these assets stored? Where are shared versions of them made available? What are the rules by which items in Dev can be moved to Test /Prod?</p>                                                                                                                                                    |
| Testing (Test)    | <p>In the Testing deployment, the objects in development are subjected to various stress tests. In the Designer Cloud powered by Trifacta platform , this testing can include load testing, malformed inputs, and changes to any parameters affecting the use of the object.</p> <p>For example, scheduled executions of flows should be thoroughly tested in this deployment. When errors are detected, they can be corrected in Dev or Test. Ideally, they are first applied in Test to address the issue at hand. Changes should then be applied back into the Dev deployment, so that future versions can consume the fix.</p>                                                                                                                     |
| Production (Prod) | <p>In the Production deployment, flows and their objects are presumed to be ready for regular, read-only use. After imported flows are reconfigured for the environment, they are ready for immediate use and require no further modification.</p> <ul style="list-style-type: none"><li>• Management of flows and jobs is typically handled via API.</li><li>• The UI should be used for checking and modifying settings and perform on-demand job executions to verify operations.</li></ul> <p>When errors are detected, you can:</p> <ul style="list-style-type: none"><li>• Revert to a previous version of the flow</li><li>• Apply any fixes in the Dev/Test instance for refinement and eventual updating back to the Prod instance.</li></ul> |

## Implementation in the platform

In the Designer Cloud powered by Trifacta platform , deployment management can be addressed in either of the following ways.

| Implementation type                                              | Description                                                                                                                                                                                                                          |
|------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Separate environments:</b> Multiple instances of the platform | <p>Dev, Test, or both environments are separate instances of the Designer Cloud powered by Trifacta platform from the Production environment.</p> <p>Flows are migrated between environments using the export/import mechanisms.</p> |

**NOTE:** Each platform instance is configured to be either a Dev instance or a Prod instance.

**All-in-one:** Single instance of the platform, separate roles

Dev, Test, and Prod are contained in a single instance of the Designer Cloud powered by Trifacta platform . This scenario can apply to cloud-based environments as well.

A user can access either Dev/Test or Prod, but not both at the same time. In this scenario, a user can access Production deployments by having the Deployment account role.

**Tip:** Access to the Production environments should be tightly controlled to prevent inadvertant changes to Production jobs.

## Licensing

**NOTE:** Designer Cloud powered by Trifacta Enterprise Edition is licensed on a per-node and per-user basis. If you have a sufficient number of nodes and users in your license to support multiple instances including any additional cluster nodes for running jobs, no additional licensing is required.

## Terminology

### Production environment terminology changes

A Prod environment focuses on management of the following objects. Differences between how these objects are used in a Dev environment are noted below.

**NOTE:** Some objects are available only in the Production environment. These objects are described later.

| Object | Differences                                                                                                                                                                                                                                                                                                                                                           |
|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Flows  | <p>In a Prod environment, you can review a flow through Flow View.</p> <div><p><b>NOTE:</b> Avoid making changes to your flows in a Prod environment. Any changes in the Prod version should be exported and then imported to the Dev version. Otherwise, when the next release is imported as a package into the Prod environment, those changes are lost.</p></div> |
| Jobs   | <p>In the Prod environment, you can execute jobs against Prod flows. For the version of the flow that is active, you trigger a job for its overall deployment. Details are below.</p> <p>These jobs are accessible through an interface that is very similar to a Dev environment.</p>                                                                                |

The following flow objects from the Dev environment must be replaced in the Prod environment:

| Dev Object        | Replacement                                                                                                                                                                                                                       |
|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Connections       | Any connections used in the Dev system must be recreated or replaced with connections in the Production system.                                                                                                                   |
| Output Objects    | Output objects from the Dev flow must be recreated or replaced in Flow View in the Prod environment.                                                                                                                              |
| Imported Datasets | If the Prod environment is not using the same sources as the Dev environment, you must create import rules to remap the point the flow to use imported datasets that are stored in a different location for the Prod environment. |

## Deployment Objects

In a Prod environment, you can explore the following objects, which are organized in a hierarchy:

| Level | Item       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|-------|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1     | deployment | A <b>deployment</b> is a versioned set of releases that have been uploaded to the Prod instance for use. You can think of it as a production instance of your primary flow and its dependencies.                                                                                                                                                                                                                                                                       |
| 2     | release    | <p>A <b>release</b> is a specific instance of a package that has been imported to the Prod instance. Each time you import, you create a new release within the deployment where you imported.</p> <p>A release is created whenever you import a package into a deployment. A <b>package</b> is a ZIP file containing a flow definition that has been exported from an instance of the Designer Cloud powered by Trifacta platform .</p>                                |
| 3     | flows      | <p>Within a release, you can explore the primary flow and any upstream flows that were included in the package. Each flow can be explored through a version of the Flow View page.</p> <ul style="list-style-type: none"><li>• The <b>primary flow</b> is the flow that you chose to export in the Dev instance.</li><li>• A <b>secondary flow</b> is any flow that is included with the package for the primary flow because the primary one depends on it.</li></ul> |

## Enable Deployment

This feature must be enabled through configuration. When enabled, the user experience of the product changes significantly, and a number of features are no longer available, including the Transformer page and its ability to modify recipes.

**Tip:** When you initially set up a platform instance, you should decide whether it is a Dev instance, a Prod instance or both.

For more information, see *Configure Deployment Manager*.

### User management

For more information on how to configure user accounts for Deployment Manager, see *Configure Deployment Manager*.

### Import/Export

To transfer your flows between instances, you must export the flow from one instance of the platform and import it into the other instance of the platform.

**NOTE:** If Dev and Prod are in the same instance, you must export the flow and import it into a deployment. These are separate processes.

**NOTE:** As part of the import process, you must define rules for how objects and values contained in the imported flow definition are remapped in the Prod environment. See below.

### Exported Flows

Through Flow View or the Flows page, you can export the flow through the context menu. The export is a ZIP file called a **package**.



**NOTE:** You must be the owner of a flow to export it.

A package ZIP contains all objects required to reconstruct and use the flow in a new environment.

- It includes the exported flow and any flows on which it depends.
- It does not include data, samples, or jobs.

### Upstream dependencies

If the outputs of an exported flow require imported datasets or recipes from another flow, that entire flow is included as part of the export package. This package includes objects that may not be required to run the primary exported flow.

### Connections

In the target instance, connections must be created prior to import. You may need to create import mapping rules to use this connections. See *Connections Page*.

### Import

How a flow is imported depends on the environment into which you are importing it and how you intend to use it.

**NOTE:** If a flow is imported into an instance that is different from the instance where it was created, you must first create remapping rules for values and objects contained in the flow definition. More information is provided below.

For more information, see *Import Flow*.

### Value and Object Mapping Rules

When objects are moved between environments, paths and other object-related references may require updating to point to the new environment.

**NOTE:** Import mapping rules do not work for parameterized datasets. If the imported dataset with parameters is still accessible, you should be able to run jobs from it.

For example, a dataset in the Dev environment may be pointing to the following location:

```
hdfs:///mydata-dev/1/00005a1a-81b0-4e4d-9c9b-f42ce55e1dde/Open_Order.csv
```

For the Prod version, the flow may need to be changed to the following:

```
hdfs:///mydata-prod/1/11115z4a-92f5-9f91-7v7f-g22fk99f2rru/Open_Order.csv
```

To support this kind of remapping, you can specify import rules at the level of individual deployments.

**NOTE:** For each deployment that you create, you must define new import remapping rules.

These rules can be specified using literal values, Trifacta patterns, or regular expressions. For more information, see *Define Import Mapping Rules*.

## Production Environment

When a user accesses a Production environment, the UI is changed to include only the following pages:

**NOTE:** You cannot modify recipes within a Prod instance because the Transformer page is not available. The Prod flow must be exported and re-imported into a Dev instance.

| Page                           | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|--------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Deployment Manager Page</i> | On this page, you create deployments, for which you manage import of packages, activation of releases, and rollback to previous release as needed.<br><br>For more information on the deployment objects, see below.                                                                                                                                                                                                                                               |
| <i>Flow View Page</i>          | Within a specific release, you can review and update the flow definition, including specification of outputs and schedules. Flow View for a Prod instance has some restrictions.<br><br><b>NOTE:</b> Use of scheduling through Flow View of a Prod instance is not supported. When a new release of a flow is imported, the schedule still points to the older release and is orphaned until the old release is reactivated or the schedule or release is removed. |
| <i>Jobs Page</i>               | Same as Dev instance. No changes.                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <i>Connections Page</i>        | Connections that have been included as part of imported packages are available for review through the Production environment.                                                                                                                                                                                                                                                                                                                                      |
| <i>Admin Settings Page</i>     | Same as Dev instance. No changes to the interface.<br><br><b>NOTE:</b> In a multi-instance environment, some settings do not apply to the Prod environment.                                                                                                                                                                                                                                                                                                        |

## Version Management

When you explore a deployment, you can see the list of releases pertaining to the deployment, with the active release listed at the top of the list. The **active** release is the one that is triggered for execution when a job is run.

You can roll back to using previous releases. Select **Activate** from the context menu for the desired release.

**NOTE:** Do not use scheduling features available through the user interface in a Production instance. If you have defined schedules through Flow View in the Prod instance and then add a new release, the schedules in the previous release are still available. You must remove them to prevent scheduled executions of outdated flows.

## Flow View Page

In a Prod instance, you can drill into a release to review its flows through Flow View page.

**NOTE:** Avoid making modifications to the flow in a Prod instance.

For more information, see *Flow View Page*.

## Example Workflow

In this example, your environment contains separate Dev and Prod instances, each of which has a different set of users.

| Item            | Dev                                                           | Prod                                 |
|-----------------|---------------------------------------------------------------|--------------------------------------|
| Environment     | http://wrangle-dev.example.com:3005                           | http://wrangle-prod.example.com:3005 |
| User            | User1<br><div><b>NOTE:</b> User1 has no access to Prod.</div> | Admin2                               |
| Source DB       | devWrangleDB                                                  | prodWrangleDB                        |
| Source Table    | Dev-Orders                                                    | Prod-Orders                          |
| Connection Name | Dev Redshift Conn                                             | Prod Redshift Conn                   |

### Example Flow:

User1 is creating a flow, which is used to wrangle weekly batches of orders for the enterprise. The flow contains:

- A single imported dataset that is created from a Redshift database table.
- A single recipe that modifies the imported dataset.
- A single output to a JSON file.
- Production data is hosted in a different Redshift database. So, the Prod connection is different from the Dev connection.

### Steps:

1. **Build in Dev instance:** User1 creates the flow and its steps.
2. **Export:** When User1 is ready to push the flow to production, User1 exports the flow from the Flows page and delivers the export package ZIP to Admin2. See *Export Flow*.
3. **Deploy to Prod instance:**
  - a. Admin2 creates a new deployment in the Prod instance. See *Deployment Manager Page*.
  - b. Admin2 creates a new connection (Prod Redshift Conn) to the Redshift database ProdWrangleDB. See *Create Connection Window*.
  - c. Admin2 creates an import rule to map the old connection (Dev Redshift Conn) to the new one (Prod Redshift Conn). See *Define Import Mapping Rules*.
  - d. Admin2 uploads the export ZIP package provided by User1. See *Import Flow*.
  - e. The deployment now contains a single release.
4. **Test deployment:**
  - a. Through Flow View in the Prod instance, Admin2 runs a job.
  - b. In reviewing the profile results of the job, Admin2 discovers a problem with the recipe. One column contains a number of mismatched values.
  - c. Admin2 chooses to fix in Dev and re-import into Prod.

**NOTE:** Any changes made in Production that must appear in future releases must be applied back in the Dev environment, too. You can either 1) export the flow from Prod and import back into Dev, or 2) manually apply all Prod changes back to the Dev environment and export/import into Prod when ready.

5. **Fix in development:** Back in the Dev environment, Admin2 opens the recipe for the flow.
  - a. Admin2 adds a step to the recipe to delete the rows containing mismatched values for the column.

- b. Admin2 runs a job and verifies that the problem is fixed. In the visual profile for the dataset, the mismatched rows are removed from the dataset.
6. **Deploy again:** Admin2 exports the flow and imports it again as a new release in the deployment.
  - a. Since import rules have already been created for this deployment, the connection is automatically re-mapped for this second import.
  - b. Admin2 runs a job. The results look fine.
  - c. Admin2 removes profiling from the output object, since profiling takes time and is unnecessary in this production environment.
7. **Set schedule:** Using cron, Admin2 sets a schedule to run the active release for this deployment once per week.
  - a. Each week, the Prod-Orders table must be refreshed with data.
  - b. The dataset is now operational in the Prod environment.

## Recommended Practices

- If possible, you should maintain separate instances of the platform for Dev and Prod.
  - If you must use the All-in-One method of managing Dev and Prod instances, you should maintain a small number of non-admin accounts that are specifically used for Deployment Manager.
- Avoid scheduling Prod executions through Flow View. While possible, these schedules continue to exist even if the version of the flow has been replaced by another. Consequently, schedules that were specified through the application continue to execute, even though the flow itself is outdated. Instead, scheduled executions should be specified at the command line through cron jobs pointing at the latest release of each at all times.
- Do not modify Flow View settings through a Prod instance. These settings are not applied back to the Dev version and are lost when the next release package is imported.

## Job Execution

### On-demand jobs

You can configure jobs on-demand through the Flow View page of a Production instance. See *Flow View Page*.

### Scheduled jobs

#### In Dev:

When your flow is exported from a Dev instance, all scheduling-related data is removed from the export package.

#### In Prod:

In a Prod instance, an imported flow contains no schedules. You must configure schedules through the REST APIs to execute on the currently active release for each deployment.

**NOTE:** Do not schedule executions through Flow View in a Prod instance.

- Schedules defined in Flow View are applied to Active and Non-Active releases in Production environments.
- If the scheduled release is deactivated, the schedule still exists, and the jobs are executed on an flow that is now out-of-date.

## Automation

Automation of Deployment Manager is supported through the APIs.

**NOTE:** When you run a deployment, you run the primary flow in the active release for that deployment. Running the flow generates the output objects for all recipes in the flow.

**NOTE:** Scheduled execution of jobs in a deployment environment must be managed through external tools such as cron. For more information on the endpoint to schedule, see <https://api.trifacta.com/ee/es.t/index.html#operation/runDeployment>

For more information on the APIs for Deployment Manager, see *API Reference*.

For more information on an API-based method for deploying flows, see *API Workflow - Deploy a Flow*.

# Overview of Pattern Matching

## Contents:

- Overview
    - Example Patterns
  - Patterns in the Platform
    - Column Profiling
    - Machine Learning
  - Pattern Matching by Data Type
  - Using Patterns
    - Selecting Data
    - Patterns in Column Details
    - Advanced Uses
  - User-Defined Patterns
- 

Designer Cloud powered by Trifacta® Enterprise Edition utilizes columnar pattern matching to identify data patterns of interest to you and to surface them in the interface for use in building your recipes. Additionally, in your recipe steps, you can apply regular expressions or Patterns to locate patterns and transform the matching data in your datasets.

## Overview

A **pattern** is a combination of abstracted character sets and literal characters that can summarize data patterns in a column. Patterns can be applied through one of two methods:

- **Regular expressions** are a standardized method of matching data. The syntax of regular expressions is both powerful and not easy to understand.
- **Trifacta patterns** are pattern-matching widgets that provide a layer of abstraction on top of regular expressions. Instead of having to specify the sometimes complex underlying regular expression, you can specify a simple token to represent the underlying expression.

**Tip:** While regular expressions are a widely used standard, Patterns are powerful simplifications that can limit the sometimes "greedy" matching issues in regular expressions.

- For more information on the supported patterns, see *Text Matching*.

This section provides an overview of the pattern matching features of the platform.

## Example Patterns

Within a row, multiple patterns may be applied at different levels of abstraction to describe the data in all fields (columns) of the row. Suppose you have two records like the following:

```
[cz.laping@gmail.com,3987,1446319063821]
[ajuneauk@gmail.com,5289,1447275151508]
```

The above records can be described by any of the following patterns:

```
[{alpha-numeric}+,{4-digits},{13-digits}]
[{email},{4-digits},{13-digits}]
[{alpha-numeric}+@gmail.com,{4-digits},{13-digits}]
```

**NOTE:** The above patterns utilize the syntax of Patterns . Regular expressions can be used to describe them as well.

In the above case, all three pattern sets capture the data completely. However, please note the differences between the patterns for column 1:

| Pattern                    | Description                                                                                                                                                        |
|----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| {alpha-numeric}+           | This pattern captures alpha-numeric values of one or more characters. So, entries that match on this pattern do not need to be valid email addresses.              |
| {email}                    | This pattern ensures matching only on valid email addresses. So, values that do not match this pattern are likely to be flagged as mismatched within the platform. |
| {alpha-numeric}+@gmail.com | This partial pattern ensures that the only matches are from gmail.com.                                                                                             |

Depending on the specific meaning of the data for your use, any of the above may apply.

## Patterns in the Platform

### Column Profiling

Pattern matching applied to columns can permit users to see the most common patterns and anomalous patterns of data in a column across the entire sample. Since patterns presented to the user encompass the entire set of values in the sample, you can gather detailed information about the consistency of data in the column across the column.

**Tip:** Column pattern profiling is especially useful after you have addressed the mismatched values in the column.

Based on the patterns surfaced for the column, you can take any of the following actions:

- **Filtering a subset of records.** For example, you can review patterns for a column of addresses and filter the rows of data where no street number is provided, based on patterns you select.
- **Standardize values.** You can make selections of patterns for the different patterns for phone numbers. See Pattern Matching by Data Type below.
- **Extract values.** You can break apart column values based on mismatches in structure. For example, apartment numbers from an address field can be extracted into a new column.
- **Variable levels of abstraction.** As demonstrated in the previous example, you may be able to select from multiple matching patterns to determine which one is the best fit for the row values of interest.

### Machine Learning

Additionally, Designer Cloud powered by Trifacta Enterprise Edition collects aggregated information about patterns applied by all users. These patterns are given weight in the set of suggested patterns presented to each user.

### Pattern Matching by Data Type

As part of pattern matching, the platform evaluates the data against the specified data type for the column. Type-specific pattern matching applies to the following data types:

- Datetime
- Phone

See *Standardize Using Patterns*.

## Using Patterns

In the application, patterns can be used as the starting point in building your next recipe step, and you can modify or iterate on a pattern definition to preview the results of the specified transformation. Patterns are used in the following actions:

- Select text to trigger a pattern-based suggestion or suggestions
- Select patterns of varying level of abstraction to modify column data

### Selecting Data

When you select a value in the data grid, your options include pattern-based suggestions. In this manner, you indicate something of interest and enable the platform to interpret your specific interest or broader goal for the selected data. These broader changes are surfaced as pattern-based suggestions in the context panel.

- See *Explore Suggestions*.
- See *Selection Details Panel*.
- For more information on how the platform predicts suggestion cards based on selection, see *Overview of Predictive Transformation*.

### Patterns in Column Details

In the Column Details panel, you can review sets of patterns that describe subsets of the values in the column. When you select one of the patterns, you are prompted with a set of suggested transform steps to apply to the data. See *Column Details Panel*.

### Advanced Uses

In addition to the above basic uses, patterns can be used as the basis for the following advanced uses and more.

| Use                       | Description                                                                                                              |
|---------------------------|--------------------------------------------------------------------------------------------------------------------------|
| Standardize records       | Match values based on a pattern and then change values to fit this pattern. See <i>Standardize Using Patterns</i> .      |
| Filter records            | Keep or delete records based on patterns of values found in row data. See <i>Filter Data</i> .                           |
| Extract values            | Extract values matching a pattern from one column and insert them into a new column of data. See <i>Extract Values</i> . |
| Generate function outputs | Use patterns to generate function outputs in new columns.                                                                |

## User-Defined Patterns

In your recipe steps, you can specify patterns using either of the following methods.

### Regular expressions

Regular expressions (regexes) are sequence of characters that can be used to define a pattern. This pattern can be used in the transformations that support regex to identify patterns in your data of interest to you. Example:

```
replace col: myCol with:/$1/ on:/^\\((\\d\\d\\d)\\)/ global: false
```

In the above step, the matching pattern expressed in the `on` clause evaluates in the following manner:

- The forward slashes around the pattern indicate that it is a regular expression.
- `^` indicates the start of the value in the `myCol` column. So, the matching is only made at the beginning of the column.



- `\(` and `\)` are representations in regular expressions of the literal values for parentheses. So, matches are made on those specific characters.
- The interior set of parentheses are used to define a capture group of values. These values, which correspond to three digits, are captured and inserted as the replacement.

So, the net effect is to search the beginning of a field for values like `( 555 )` and replace them with just the digits: `555`. This replacement removes the parentheses from the area code part of a phone number.

**NOTE:** Regular expressions are very powerful tools for matching patterns. They can also cause unexpected results. Use of regular expressions is considered a developer-level skill. You should use the Patterns described below instead.

Designer Cloud powered by Trifacta Enterprise Edition implements a version of regular expressions based off of *RE2* and *PCRE* regular expressions.

## Patterns

Use `Patterns` to quickly assemble sophisticated patterns to match in your data. The following example includes the equivalent `Pattern` as the previous regular expression:

```
replace col: myCol with:`$1` on:`^\\(({digit}{3})\\)` global: false
```

- The back-ticks around the pattern indicate that it is a `Pattern`.

For more information, see *Text Matching*.

# Overview of RapidTarget

## Contents:

- Overview
    - *Targets in the platform*
    - *Known Limitations*
  - *Creating Targets*
    - *Sources for creating a target*
    - *Creating a target for a recipe*
  - *Using a target*
  - *Running jobs on recipes with assigned targets*
  - *Configure*
    - *Configure fuzzy matching threshold*
    - *Disable*
- 

In Designer Cloud powered by Trifacta® Enterprise Edition, a **target** is the set of columns, their order, and their formats to which you are attempting to wrangle your dataset. This target can be defined through imported or created datasets and must be assigned to an existing recipe. After it is assigned to a recipe, a target appears in the Transformer page to assist in your wrangling efforts. You can also apply changes to selected columns based on the target.

- This feature was formerly known as, "target matching."

## Overview

In general, a target consists of the set of information required to define the expected data in a dataset. Often referred to as a "schema," this target schema information can include:

- Names of columns
- Order of columns
- Column data types
- Data type format
- Example rows of data

A dataset associated with a target is expected to conform to the requirements of the schema. Where there are differences between target schema and dataset schema, a validation indicator (or **schema tag**) is displayed.

## Targets in the platform

In Designer Cloud powered by Trifacta Enterprise Edition, a target is created from the information in a dataset and can be applied to a recipe in a flow. When you are working with the flow, the target information is available for your wrangling activities, so that you can match up columns in your dataset (source) with their corresponding columns in the target. As you make changes in your recipe through the Transformer page, the target schema is available as a reference to see if your latest changes get you closer to matching the dataset to the target.

## Known Limitations

- Targets are applied only after initial type inferencing has been applied to the loaded dataset.

**Tip:** As needed, you can disable initial type inferencing when data is imported into the product. See *Import Data Page*.

- Type-based matching applies a `settype` transform to any selected column. No pattern matching or standardization is applied. For more information, see *Overview of Pattern Matching*.
- Changes to the underlying objects of a target schema are not reflected in the schema. A target schema is a snapshot of the object at the time of its creation. To update, delete the target and create a new one.

**Tip:** If your target schema source is a recipe, then you can modify the recipe as needed and use it as your target again.

## Creating Targets

### Sources for creating a target

The schema used to define a target can be imported and assigned from any of the following objects, including:

- Output of a recipe in the same flow
- A reference dataset from another flow
- An imported dataset

Ideally, the source of the target schema should come from the publishing target. If you are publishing to a pre-existing target, you can create do one of the following:

- **Reference the target:** If the schema is represented in a dataset to which you have access in Designer Cloud powered by Trifacta Enterprise Edition, you can use it as your target schema.
- **Import the target:** Import the target table or schematized source into Designer Cloud powered by Trifacta Enterprise Edition as an imported dataset. Then, it can be selected as the target schema for any recipe to which you have access. See *Import Data Page*.
- **Extract target to a supported format:** If you cannot import the target directly into Designer Cloud powered by Trifacta Enterprise Edition, you could create an extract of a few rows, including the header, for the target into one of the formats supported for import. For more information, see *Supported File Formats*.

### Creating a target for a recipe

You can create a target through one of the following mechanisms:

- **Flow View:** Select a recipe. From the context menu in the right panel, select **Assign Target to Recipe**. See *Flow View Page*.
- **Transformer Page:** Above the data grid, click the Target icon and select **Attach a new Target**.
  - See *Transformer Toolbar*.
  - You can do the same thing in the Column Browser panel. See *Column Browser Panel*.
- **Job Details Page:** After you have successfully run a job, you can create a new dataset from the Output Destinations tab. Through Flow View, this imported dataset can be used as the schema for wrangling. See *Job Details Page*.

For more information, see *Create Target*.

### Using a target

After a target has been attached to a recipe, the target schema appears in a toolbar above the data grid along with a preview of the data. You can then make modifications to the data so that each column matches the definition for the corresponding column in the schema. See *Data Grid Panel*.

Through the data grid and the Column Browser, you can perform operations on selected columns in your dataset to align them with the target schema. For more information, see *Column Browser Panel*.

## Running jobs on recipes with assigned targets

**NOTE:** You can run a job even if there are differences between the schema and your dataset. In Designer Cloud powered by Trifacta Enterprise Edition, no error checking is performed between schema and data prior to job execution. If you are publishing to a target that has a predefined schema, a publication error may be generated.

## Configure

### Configure fuzzy matching threshold

You can experiment with fuzzy matching thresholds to ensure that matches are occurring properly. This parameter applies a specific threshold value when two values are compared for matching. Lower values increase the probability of a match.

#### Steps:

1. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
2. Adjust the value between 0.00 and 0.99 for the following parameter:

```
"feature.targetMatching.fuzzyMatchingThreshold": 0.30;
```

3. Save your changes and restart the platform.

### Disable

If you prefer to disable this feature, please complete the following steps.

**NOTE:** If you are experiencing performance issues with target matching, you can first try to disable fuzzy matching, which can be resource-intensive.

**Tip:** If there is no schema associated with a recipe, then the target schema matching features are not displayed.

#### Steps:

1. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
2. Set the following parameters to `false`:

```
"feature.targetMatching.enabled" : false,
"feature.targetMatching.fuzzyMatchingEnabled" : false,
```

3. Save your changes and restart the platform.

# Recipe Development

This section contains general guidance and overviews of developing your recipes.

# Enriching Data

## Contents:

- *Union*
- *Join*
- *Lookup*
- *Aggregation*

Designer Cloud powered by Trifacta® Enterprise Edition provides multiple tools for bringing data from other sources into your dataset.

## Union

A **union** operation concatenates multiple datasets together. An example is below.

**Tip:** The following example unions two datasets based on the position of the columns. Unions may also be performed based on the column names.

### Dataset 1:

| CName1 | CName2 | CName3 |
|--------|--------|--------|
| C1.1   | C2.1   | C3.1   |
| C1.2   | C2.2   | C3.2   |
| C1.3   | C2.3   | C3.3   |

### Dataset 2:

| CName1 | CName2 | CName4 |
|--------|--------|--------|
| C4.1   | C5.1   | C6.1   |
| C4.2   | C5.2   | C6.2   |
| C4.3   | C5.3   | C6.3   |

When a union is performed based on the position of the columns in each dataset, all of the rows of Dataset 1 are included, followed by all of the rows of Dataset 2. You can choose which columns to include from each of the source datasets.

## Output:

In the above, note that the name of the third column in each dataset is different (CName3 and CName4).

| CName1 | CName2 | CName3 | CName4 |
|--------|--------|--------|--------|
| C1.1   | C2.1   | C3.1   |        |
| C1.2   | C2.2   | C3.2   |        |
| C1.3   | C2.3   | C3.3   |        |
| C4.1   | C5.1   |        | C6.1   |
| C4.2   | C5.2   |        | C6.2   |

**When to use:**

**Tip:** You should perform union operations as early as possible in your recipes.

- If your datasets include event or log information, you can use the union operation to create a longer sequence of those transactions. For example, you might union together all of your log data for a week from daily log files.

To union your dataset to another, enter `Union datasets` the Transformation textbox in the recipe panel. See *Recipe Panel*.

See *Union Page*.

**Join**

A join operation brings together two datasets based on a column that appears in both datasets and contains the same unique values used to identify records. Based on the values in this column, called the **primary key**, records in the second dataset are joined to records in the first dataset. As part of the join definition, you may select the fields from both datasets to include, filtering out any duplicated or unnecessary fields in the combined dataset.

The way in which the two datasets are joined is defined by the type of join:

- inner join - include only the records in which key (**primary key**) values in the first dataset appear as key (**foreign key**) values in the second dataset.
- left join - include only the records that contain a primary key value that appears in the first (left) dataset.
  - If a primary key value from the first dataset does not appear as a foreign key in the second dataset, any columns brought in from the second dataset contain missing values.
  - Foreign key values that appear in the second dataset and not the first one do not generate rows in the output dataset.
- right join - include only the records that contain a foreign key value that appears in the second (right) dataset. The other conditions above apply in reverse.
- outer join - include all records from both datasets. If a key value is missing from either dataset, the column values included from that dataset are missing.
- For more information, see *Join Types*.

**When to use:**

**Tip:** Generally, you should perform join operations as late as possible in your recipes.

- A join is useful for pulling in **selected** fields from a second dataset based on matches of key values. These operations can be expensive to execute but can generate a much wider range of output datasets.

To join your dataset with another, enter `join datasets` in the Search panel. See *Join Window*.

**Lookup**

A **lookup** operation is used to pull in reference fields from another dataset based on the values contained in a selected column of the first dataset. These second datasets are typically static or changing infrequently.

**NOTE:** A lookup is similar to a left join. However, with a lookup, all fields from the reference dataset are brought into the generated dataset and all fields from the original dataset are included automatically. When you create a join, you may specify the fields to include in the output dataset.

For example, you might create a dataset like the following:

| State-2letters | State-full |
|----------------|------------|
| AL             | Alabama    |
| AK             | Alaska     |
| AZ             | Arizona    |
| ...            | ...        |
| WI             | Wisconsin  |
| WY             | Wyoming    |

If you have a dataset containing the two-letter abbreviations, you can perform a lookup into the above dataset to retrieve the corresponding full names, which are inserted as an adjacent column called `State-full` in the original dataset.

**NOTE:** If a value in the column from the first dataset does not appear in the second dataset, there is no corresponding value in the generated `State-full` column.

#### When to use:

- Lookups are useful for referencing shared datasets whose meaning must be consistent across multiple datasets. You can use lookups to pull in customer or product master data (Customer name, address, etc.) based on CustomerId or ProductId values.

To perform a lookup from a column in your dataset, open the column drop-down and select **Lookup....** See *Lookup Wizard*.

## Aggregation

A single-dataset operation, an aggregation is used to perform summary calculations on columns in your dataset, optionally grouping your data by the values in one or more columns.

For example, your dataset contains point-of-sale transactions from all of the stores in your organization. You can use an aggregation to summarize total sales by performing a sum operation on your `Total_Sale` column. If you group this calculation by month and by `StoreId`, you can acquire monthly sales per month per store.

#### When to use:

- An aggregation is useful for performing exploratory calculations on your entire dataset or segments of your dataset.
- You can perform aggregations and run jobs to generate the results. After you have these summary reports, you can return to the Transformer page and remove the aggregation to continue wrangling your data.

For more information on in-column aggregations, see *Create Aggregations*.

For more information on building aggregated pivot tables, see *Pivot Data*.



# Using Connections

This section contains some basic information on using common connections to various types of supported storage.

# Using Databases

## Contents:

- *Before You Begin*
    - *Access*
  - *Storing Data in Relational Databases*
  - *Reading from Database Tables and Views*
    - *Additional Notes on Database Views*
  - *Running Jobs from Database Sources*
  - *Writing to Databases*
- 

This section describes how you interact with your databases through Designer Cloud powered by Trifacta® Enterprise Edition.

- Specific versions of each database are supported.
- Connections must be enabled and configured for each type of supported database.
- See *Connection Types*.

## Before You Begin

- **Read Access:** Your database administrator must configure read permissions to the appropriate databases, tables and views for your use.

**NOTE:** To ensure that all user credentials used to access the database system are securely stored, you must first deploy the encryption key file to the Trifacta node. See *Relational Access*.

- **Write Access:** Some relational connection types support write access. For more information, see *Connection Types*.
  - This feature must be enabled. See *Relational Access*.

## Access

Database access is managed through connections.

- Individual users can create private connections through the application. See *Create Connection Window*.
- An administrator can make your connection public or create public connections through the application.

## Storing Data in Relational Databases

**NOTE:** Designer Cloud powered by Trifacta Enterprise Edition does not modify source data nor store transformed data in the relational systems. Datasets sourced from database tables or views are read without modification from their source locations.

## Reading from Database Tables and Views

You can create a Trifacta dataset from a table or view stored in a connected database.

**Tip:** In some scenarios, you can improve performance of loading from database tables by creating a view on the table to restrict the amount of data loaded to only the required fields. Additionally, you can pre-filter the dataset using custom SQL statements. See *Create Dataset with SQL*.

### Additional Notes on Database Views

- Some metadata, such as row counts, is not available for database views.
- For complex view definitions that require significant processing on the database, there may be a significant delay when previewing the contents of those views. In some cases, the preview may time out waiting for the database to respond with the view contents.

For more information, see *Database Browser*.

## Running Jobs from Database Sources

**NOTE:** When executing a job using a relational source, the job may fail if one or more columns has been dropped from the underlying source table. As a workaround, the recipe panel may show steps referencing the missing columns, which be used to fix to either fix the recipe or the source data.

## Writing to Databases

Relational connections can be configured to support writing results back to the database.

**NOTE:** You can only write to databases from the Run Job page. You cannot ad-hoc publish to a relational database.

**NOTE:** When writing to a new table in a relational target, the first entry in any mapping is used for writing out the value. Subsequent entries in the mapping are used for validation only on writing to new tables. See *Connection Types*.

Natively supported connection types are automatically enabled for writeback.

# Using HDFS

## Contents:

- *Uses of HDFS*
  - *Before You Begin Using HDFS*
    - *Secure Access*
  - *Storing Data in HDFS*
    - *Ingest Caching*
  - *Reading from Sources in HDFS*
  - *Creating Datasets*
  - *Writing Job Results*
    - *Creating a new dataset from results*
  - *Purging Files*
- 

This section describes how you interact through the Designer Cloud powered by Trifacta® platform with your HDFS environment.

- HDFS is a scalable file storage system for use across all of the nodes (servers) of a Hadoop cluster. Many interactions with HDFS are similar with desktop interactions with files and folders. However, what looks like a "file" or "folder" in HDFS may be spread across multiple nodes in the cluster. For more information, see [https://en.wikipedia.org/wiki/Apache\\_Hadoop#HDFS](https://en.wikipedia.org/wiki/Apache_Hadoop#HDFS).

## Uses of HDFS

The Designer Cloud powered by Trifacta platform can use HDFS for the following reading and writing tasks:

1. **Creating Datasets from HDFS Files:** You can read in from a data source stored in HDFS. A source may be a single HDFS file or a folder of identically structured files. See *Reading from Sources in HDFS* below.
2. **Reading Datasets:** When creating a dataset, you can pull your data from another dataset defined in HDFS. See *Creating Datasets* below.
3. **Writing Job Results:** After a job has been executed, you can write the results back to HDFS. See *Writing Job Results* below.

In the Designer Cloud application, HDFS is accessed through the HDFS browser. See *HDFS Browser*.

**NOTE:** When the Designer Cloud powered by Trifacta platform executes a job on a dataset, the source data is untouched. Results are written to a new location, so that no data is disturbed by the process.

## Before You Begin Using HDFS

- **Read/Write Access:** Your Hadoop administrator must configure read/write permissions to locations in HDFS. Please see the HDFS documentation provided with your Hadoop distribution.

**Avoid using `/trifacta/uploads` for reading and writing data. This directory is used by the Designer Cloud application.**

**NOTE:** Use of HDFS in safe mode is not supported.

- Your Hadoop administrator should provide a place or mechanism for raw data to be uploaded to your Hadoop datastore.

- Your Hadoop administrator should provide a writeable home output directory for you, which you can review. See *Storage Config Page*.

## Secure Access

Depending on the security features you've enabled, the technical methods by which Trifacta users access HDFS may vary. For more information, see *Configure Hadoop Authentication*.

## Storing Data in HDFS

Your Hadoop administrator should provide raw data or locations and access for storing raw data within HDFS. All Trifacta users should have a clear understanding of the folder structure within HDFS where each individual can read from and write their job results.

- Users should know where shared data is located and where personal data can be saved without interfering with or confusing other users.

**NOTE:** The Designer Cloud powered by Trifacta platform does not modify source data in HDFS. Sources stored in HDFS are read without modification from their source locations, and sources that are uploaded to the platform are stored in `/trifacta/uploads`.

## Ingest Caching

If JDBC ingest caching has been enabled, users may see a `dataSourceCache` folder in their browser. This folder is used to store per-user caches of JDBC-based data that has been ingested into the platform from its source.

**NOTE:** The `datasourceCache` folder should not be used for reading and writing of datasets, metadata, or results.

For more information, see *Configure JDBC Ingestion*.

## Reading from Sources in HDFS

You can create a dataset from one or more files stored in HDFS.

**NOTE:** To be able to import datasets from the base storage layer, your user account must include the `dataAdmin` role.

## Wildcards:

You can parameterize your input paths to import source files as part of the same imported dataset. For more information, see *Overview of Parameterization*.

## Folder selection:

When you select a folder in HDFS to create your dataset, you select all files in the folder to be included. Notes:

- This option selects all files in all sub-folders. If your sub-folders contain separate datasets, you should be more specific in your folder selection.

- All files used in a single dataset must be of the same format and have the same structure. For example, you cannot mix and match CSV and JSON files if you are reading from a single directory.
- When a folder is selected from HDFS, the following file types are ignored:
  - \*\_SUCCESS and \*\_FAILED files, which may be present if the folder has been populated by Hadoop.
  - If you have stored files in HDFS that begin with an underscore (\_), these files cannot be read during batch transformation and are ignored. Please rename these files through HDFS so that they do not begin with an underscore.

## Creating Datasets

When creating a dataset, you can choose to read data in from a source stored from HDFS or from a local file.

- HDFS sources are not moved or changed.
- Local file sources are uploaded to `/trifacta/uploads` where they remain and are not changed.

Data may be individual files or all of the files in a folder. For more information, see *Reading from Sources in HDFS*.

- In the Import Data page, click the HDFS tab. See *Import Data Page*.

## Writing Job Results

When your job results are generated, they can be stored back in HDFS for you at the location defined for your user account.

- The HDFS location is available through the Output Destinations tab of the Job Details page. See *Job Details Page*.
- Each set of job results must be stored in a separate folder within your HDFS output home directory.
- For more information on your output home directory, see *Storage Config Page*.

**If your deployment is using HDFS, do not use the `trifacta/uploads` directory. This directory is used for storing uploads and metadata, which may be used by multiple users. Manipulating files outside of the Designer Cloud application can destroy other users' data. Please use the tools provided through the interface for managing uploads from HDFS.**

**NOTE:** Users can specify a default output home directory and, during job execution, an output directory for the current job. In an encrypted HDFS environment, these two locations must be in the same encryption zone. Otherwise, writing the job results fails with a `Publish Job Failed` error.

### Access to results:

Depending on how the platform is integrated with HDFS, other users may or may not be able to access your job results.

- If user impersonation is enabled, results are written to HDFS through the HDFS account configured for your use. Depending on the permissions of your HDFS account, you may be the only person who can access these results.
- If user impersonation is not enabled, then each Trifacta user writes results to HDFS using a shared account. Depending on the permissions of that account, your results may be visible to all platform users.

## Creating a new dataset from results

As part of writing job results, you can choose to create a new dataset, so that you can chain together data wrangling tasks.

**NOTE:** When you create a new dataset as part of your job results, the file or files are written to the designated output location for your user account. Depending on how your Hadoop permissions are configured, this location may not be accessible to other users.

## Purging Files

Other than temporary files, the Designer Cloud powered by Trifacta platform does not remove any files that were generated or used by the platform, including:

- Uploaded datasets
- Generated samples
- Generated results

If you are concerned about data accumulation, please contact your HDFS administrator.

# Using S3

## Contents:

- *Uses of S3*
  - *Before You Begin Using S3*
    - *Secure Access*
  - *Storing Data in S3*
  - *Reading from Sources in S3*
  - *Creating Datasets*
  - *Writing Results*
    - *Creating a new dataset from results*
  - *Purging Files*
- 

This section describes how you interact through the Designer Cloud powered by Trifacta® platform with your S3 environment.

- Simple Storage Service (S3) is an online data storage service provided by Amazon, which provides low-latency access through web services. For more information, see <https://aws.amazon.com/s3/>.

## Uses of S3

The Designer Cloud powered by Trifacta platform can use S3 for the following tasks:

1. **Enabled S3 Integration:** The Designer Cloud powered by Trifacta platform has been configured to integrate with your S3 instance. For more information, see *S3 Access*.
2. **Creating Datasets from S3 Files:** You can read in source data stored in S3. An imported dataset may be a single S3 file or a folder of identically structured files. See *Reading from Sources in S3* below.
3. **Reading Datasets:** When creating a dataset, you can pull your data from a source in S3. See *Creating Datasets* below.
4. **Writing Results:** After a job has been executed, you can write the results back to S3. See *Writing Results* below.

In the Designer Cloud application, S3 is accessed through the S3 browser. See *S3 Browser*.

**NOTE:** When the Designer Cloud powered by Trifacta platform executes a job on a dataset, the source data is untouched. Results are written to a new location, so that no data is disturbed by the process.

## Before You Begin Using S3

- **Access:** If you are using system-wide permissions, your administrator must configure access parameters for S3 locations. If you are using per-user permissions, this requirement does not apply. See *S3 Access*.

**Avoid using `/trifacta/uploads` for reading and writing data. This directory is used by the Designer Cloud application.**

- Your administrator should provide a writeable home output directory for you. This directory location is available through your user profile. See *Storage Config Page*.



## Secure Access

Your administrator can grant access on a per-user basis or for the entire Designer Cloud powered by Trifacta platform .

The Designer Cloud powered by Trifacta platform utilizes an S3 key and secret to access your S3 instance. These keys must enable read/write access to the appropriate directories in the S3 instance.

**NOTE:** If you disable or revoke your S3 access key, you must update the S3 keys for each user or for the entire system.

For more information, see [S3 Access](#).

## Storing Data in S3

Your administrator should provide raw data or locations and access for storing raw data within S3. All Trifacta users should have a clear understanding of the folder structure within S3 where each individual can read from and write results.

- Users should know where shared data is located and where personal data can be saved without interfering with or confusing other users.
- The Designer Cloud application stores the results of each job in a separate folder in S3.

**NOTE:** The Designer Cloud powered by Trifacta platform does not modify source data in S3. Source data stored in S3 is read without modification from source locations, and source data uploaded to the Designer Cloud powered by Trifacta platform is stored in `/trifacta/uploads`.

## Reading from Sources in S3

You can create an imported dataset from one or more files stored in S3.

**NOTE:** To be able to import datasets from the base storage layer, your user account must include the `dataAdmin` role.

**NOTE:** Import of glaciated objects is not supported.

### Wildcards:

You can parameterize your input paths to import source files as part of the same imported dataset. For more information, see [Overview of Parameterization](#).

### Folder selection:

When you select a folder in S3 to create your dataset, you select all files in the folder to be included.

Notes:

- This option selects all files in all sub-folders and bundles them into a single dataset. If your sub-folders contain separate datasets, you should be more specific in your folder selection.

- All files used in a single imported dataset must be of the same format and have the same structure. For example, you cannot mix and match CSV and JSON files if you are reading from a single directory.

When a folder is selected from S3, the following file types are ignored:

- `*_SUCCESS` and `*_FAILED` files, which may be present if the folder has been populated by the running environment.

**NOTE:** If you have a folder and file with the same name in S3, search only retrieves the file. You can still navigate to locate the folder.

## Creating Datasets

When creating a dataset, you can choose to read data in from a source stored from S3 or local file.

- S3 sources are not moved or changed.
- Local file sources are uploaded to `/trifacta/uploads` where they remain and are not changed.

Data may be individual files or all of the files in a folder. In the Import Data page, click the S3 tab. See *Import Data Page*.

**Tip:** Users can create secondary connections to specific S3 buckets. For more information, see *External S3 Connections*.

## Writing Results

When you run a job, you can specify the S3 bucket and file path where the generated results are written. By default, the output is generated in your default bucket and default output home directory.

- Each set of results must be stored in a separate folder within your S3 output home directory.
- For more information on your output home directory, see *Storage Config Page*.

**If Trifacta installation is using S3, do not use the `trifacta/uploads` directory. This directory is used for storing uploads and metadata, which may be used by multiple users. Manipulating files outside of the Designer Cloud application can destroy other users' data. Please use the tools provided through the Designer Cloud application interface for managing uploads from S3.**

**NOTE:** When writing files to S3, you may encounter an issue where the UI indicates that the job failed, but the output file or files have been written to S3. This issue may be caused when S3 does not report the files back to the application before the S3 consistency timeout has expired. For more information on raising this timeout setting, see *S3 Access*.

## Creating a new dataset from results

As part of writing results, you can choose to create a new dataset, so that you can chain together data wrangling tasks.

**NOTE:** When you create a new dataset as part of your results, the file or files are written to the designated output location for your user account. Depending on how your permissions are configured, this location may not be accessible to other users.

## Purging Files

Other than temporary files, the Designer Cloud powered by Trifacta platform does not remove any files that were generated or used by the platform, including:

- Uploaded datasets
- Generated samples
- Generated results

If you are concerned about data accumulation, you should create a bucket policy to periodically backup or purge directories in use. For more information, please see the S3 documentation.

# Using SQL DW

## Contents:

- *Limitations*
  - *Uses of Azure Synapse Analytics (Formerly Microsoft SQL DW)*
  - *Before You Begin Using Azure Synapse Analytics (Formerly Microsoft SQL DW)*
    - *Secure Access*
  - *Storing Data in Azure Synapse Analytics (Formerly Microsoft SQL DW)*
  - *Reading from Azure Synapse Analytics (Formerly Microsoft SQL DW)*
  - *Writing to Azure Synapse Analytics (Formerly Microsoft SQL DW)*
- 

This section describes how you interact through the Designer Cloud powered by Trifacta® platform with Azure® Synapse Analytics (Formerly Microsoft® SQL DW)® .

- Azure Synapse Analytics (Formerly Microsoft SQL DW) is a scalable data warehouse solution available through Microsoft Azure. For more information, see <https://docs.microsoft.com/en-us/azure/sql-data-warehouse/sql-data-warehouse-overview-what-is>.
- Azure Synapse Analytics (Formerly Microsoft SQL DW) connections can interact with data stored as managed tables or external tables.
- Azure Synapse Analytics (Formerly Microsoft SQL DW) connections can use dedicated or serverless SQL pools.
- For more information, see *Microsoft SQL Data Warehouse Connections*.

## Limitations

- Azure Synapse Analytics (Formerly Microsoft SQL DW) connections are available only if you have deployed the Designer Cloud powered by Trifacta platform onto Azure.
- The defined length of a table row cannot exceed 1 MB.

**NOTE:** In this release, this connection cannot be created through the APIs. Please create connections of this type through the application.

## Uses of Azure Synapse Analytics (Formerly Microsoft SQL DW)

The Designer Cloud powered by Trifacta platform can use Azure Synapse Analytics (Formerly Microsoft SQL DW) for the following tasks:

1. Create datasets by reading from Azure Synapse Analytics (Formerly Microsoft SQL DW) tables.
2. Write to Azure Synapse Analytics (Formerly Microsoft SQL DW) tables with your job results.
3. Ad-hoc publication of data to Azure Synapse Analytics (Formerly Microsoft SQL DW) .

## Before You Begin Using Azure Synapse Analytics (Formerly Microsoft SQL DW)

- **Enable Access:** Integration requires the following:
  - Installation of the Designer Cloud powered by Trifacta platform on Microsoft Azure.
  - Either ADL or WASB is supported as the base storage layer. For more information, see *Set Base Storage Layer*.
- **Read Access:** Your Azure Synapse Analytics (Formerly Microsoft SQL DW) administrator must configure read permissions. Your administrator should provide a database for upload.

- **Write Access:** You can write and publish jobs results to Azure Synapse Analytics (Formerly Microsoft SQL DW) .

## Secure Access

These connections require SSL access.

## Storing Data in Azure Synapse Analytics (Formerly Microsoft SQL DW)

Your Azure Synapse Analytics (Formerly Microsoft SQL DW) administrator should provide database access for storing datasets. Users should know where shared data is located and where personal data can be saved without interfering with or confusing other users.

**NOTE:** The Designer Cloud powered by Trifacta platform does not modify source data. Datasets sourced from Azure Synapse Analytics (Formerly Microsoft SQL DW) are read without modification from their source locations.

## Reading from Azure Synapse Analytics (Formerly Microsoft SQL DW)

You can create a Trifacta dataset from a table stored in Azure Synapse Analytics (Formerly Microsoft SQL DW) .

For more information, see *Database Browser*.

## Writing to Azure Synapse Analytics (Formerly Microsoft SQL DW)

You can write back data to Azure Synapse Analytics (Formerly Microsoft SQL DW) using one of the following methods:

**NOTE:** Writing and publishing to Azure Synapse Analytics (Formerly Microsoft SQL DW) is not supported if Azure AD SSO has been enabled.

- Job results can be written directly to Azure Synapse Analytics (Formerly Microsoft SQL DW) as part of the normal job execution. Create a new publishing action to write to Azure Synapse Analytics (Formerly Microsoft SQL DW) . See *Run Job Page*.
- As needed, you can publish results to Azure Synapse Analytics (Formerly Microsoft SQL DW) for previously executed jobs.
- For more information on how data is converted to Azure Synapse Analytics (Formerly Microsoft SQL DW) , see *SQL DW Data Type Conversions*.

## Data Validation issues:

- No validation is performed for the connection and any required permissions during job execution. So, you can be permitted to launch your job even if you do not have sufficient connectivity or permissions to access the data. The corresponding publish job fails at runtime.
- No data validation is performed during writing and publication to Azure Synapse Analytics (Formerly Microsoft SQL DW) . Your job fails if the schema for the Trifacta dataset varies from the target schema.
- Prior to publication, no validation is performed on whether a target is a table or a view, so the job that was launched fails at runtime.

# Reference

This section contains reference content on the interface and other aspects of Designer Cloud powered by Trifacta® Enterprise Edition.

# UI Reference

Review reference information on each screen available in the Designer Cloud® application .

- For a summary of each available UI page, see *UI Index*.

**NOTE:** This generated page may take time to load.

- For more information on UI pages that apply to administrators, see *Admin Reference*.

# Home Page

Contents:

- Onboarding Tour
- Recently Updated
- Recent Runs
- Resources
- Left Nav Bar
  - Flows
  - Plans
  - Library
  - Connections
  - Jobs
  - Schedules
  - Help menu
  - User menu

From the Home page, you can create or access your flows, datasets, and jobs, as well as configure settings and find additional resources.

**Tip:** Click the logo at the top of the menu bar to return to the Home page.

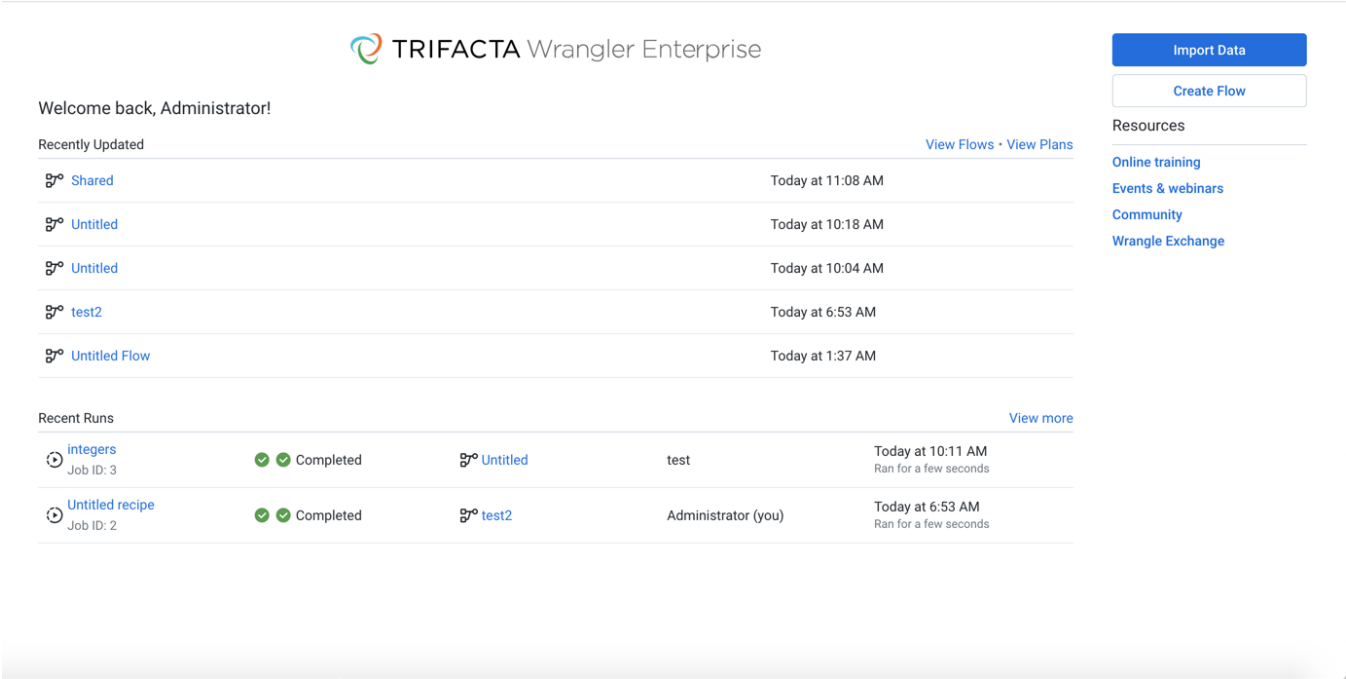


Figure: Home Page



## Onboarding Tour

**NOTE:** This feature may need to be enabled in your environment by an administrator. For more information, see *Enable Onboarding Tour*.

The onboarding tour is available to all users from the Home page. You can toggle display of it as needed.

**Getting Started?** Try out the embedded tour to see how Designer Cloud powered by Trifacta Enterprise Edition can accelerate and enhance your data wrangling.

**NOTE:** Do not have multiple windows open to the application when using the onboarding tour. Avoid using with any form of shared account.

To begin the tour, click **Show Tour**.

**NOTE:** The onboarding tour uses a sample dataset that has been installed for you. When the tour is completed, it is removed from the Home page and cannot be replayed. The flow, dataset, and recipe remain in your account for your reference.

From the Home page, you can quickly access your recent activities in Designer Cloud powered by Trifacta Enterprise Edition or jump to creating flows and importing datasets.

**Tip:** When keyboard shortcuts are enabled, press ? in the application to see the available shortcuts. Individual users must enable them. See *User Profile Page*.

- **Import Data:** Import new datasets into Designer Cloud powered by Trifacta Enterprise Edition. See *Import Data Page*.
- **Create Flow:** Create a new flow to hold your datasets. See *Create Flow Page*.

**Tip:** Use the controls on the left side of the screen to access other areas of the application. For more information, see Menu Bar below.

## Recently Updated

Access the flows that have been recently changed. Click the flow name to open it. See *Flow View Page*.

**Tip:** When an object within a flow has been changed, its timestamp here is updated, so the Home page becomes an easy location where you can monitor changes to the flows to which you have access. Monitored changes include editing a recipe or adding or removing datasets.

### Actions:

- **View more:** See all of your flows.
- For more information on these options, see *Flows Page*.

## Recent Runs

### Recent jobs:

Review jobs that you have been recently queued or completed in Designer Cloud powered by Trifacta Enterprise Edition.

- Click a job ID to view its details. See *Job Details Page*.
- Click the name of the flow to open it. See *Flow View Page*.
- Click the name of a recipe to select it in Flow View. See *Flow View Page*.

### Recent Runs:

Use the links to explore recent plan runs that have been queued or completed. For more information, see *Plan Runs Page*.

### Actions:

- **Cancel job:** If present, you can cancel a job in progress.

**NOTE:** Some jobs cannot be canceled through the Designer Cloud application .

- **View more:** See all of your jobs.
- For more information on these options, see *Jobs Page*.

## Resources

Use the links on the right to explore available resources for Designer Cloud powered by Trifacta Enterprise Edition.

## Left Nav Bar

From the left side of the screen, you can access the top-level pages of the Designer Cloud application .

### Flows

Use the Flows page to create and manage your flows. See *Flows Page*.

- A flow is a container for one or more datasets. See *Create Flow Page*.

### Plans

Use the Plans page to build sequences of tasks that are executed based upon triggers that you create. See *Plans Page*.

- Plans enable the scheduling of execution of task sequences. For more information, see *Overview of Operationalization*.

### Library

From the Library page, you create and manage your datasets. See *Library Page*.

- You can import datasets to begin creating your transformations. See *Import Data Page*.
- Click the linked name of a dataset to transform the data. See *Transformer Page*.

## Connections

Connections enable you to import datasets for use in your flows. Depending on the type of connection, they can also be used for writing or publishing data back to the datastore.

**NOTE:** This option appears only if connectivity must be enabled in your environment.

For more information on creating and editing connections to your data, see *Connections Page*.

## Jobs

After you finish building your transformation recipe, you can run jobs to execute the recipe against your dataset. The Jobs page shows status and history of your jobs. See *Jobs Page*.

You can also review any plan runs that you have executed. See *Plan Runs Page*.

## Schedules

Administrators can review, toggle availability, and delete any schedule in the deployment.

**NOTE:** This page is available to administrators only.

See *Schedules Page*.

## Help menu

Access help resources, including documentation, training, and more.

- **Community:** Explore the online community.
- **Wrangle Exchange:** Explore and download additional resources, such as macros, from the online Wrangle Exchange.
- **Documentation:** Access online product documentation.
- **API documentation:** Access reference documentation for available endpoints and methods.
  - For additional information on API workflows and other API-related content, see *API Reference*.
- **Trifacta Academy:** Explore the training programs and certifications available through Trifacta Academy. Wrangler certification is free.
- **Download logs:** Download logs from the current session. See *Download Logs Dialog*.
  - Admin users can download logs for jobs, sessions, or time periods. See *Admin Download Logs Dialog*.
- **About:** Review information about the product, build number, and licensing information.

- **Keyboard shortcuts:** Review available keyboard shortcuts for the Designer Cloud application .
  - You must enable keyboard shortcuts. See *User Profile Page*.

## User menu

Under the User icon, you can modify settings specific to your account.

### Preferences

Review preferences for your account and other settings. See *Preferences Page*.

- **Profile:** Edit your profile. See *User Profile Page*.

### Admin console

Review and modify settings and users for your workspace. See *Admin Console*.

**Tip:** If you have two open tabs for work on the same dataset, changes made in one browser tab may not be reflected in the other browser tab until you refresh the page. Overwriting results from one tab through another is certainly possible.

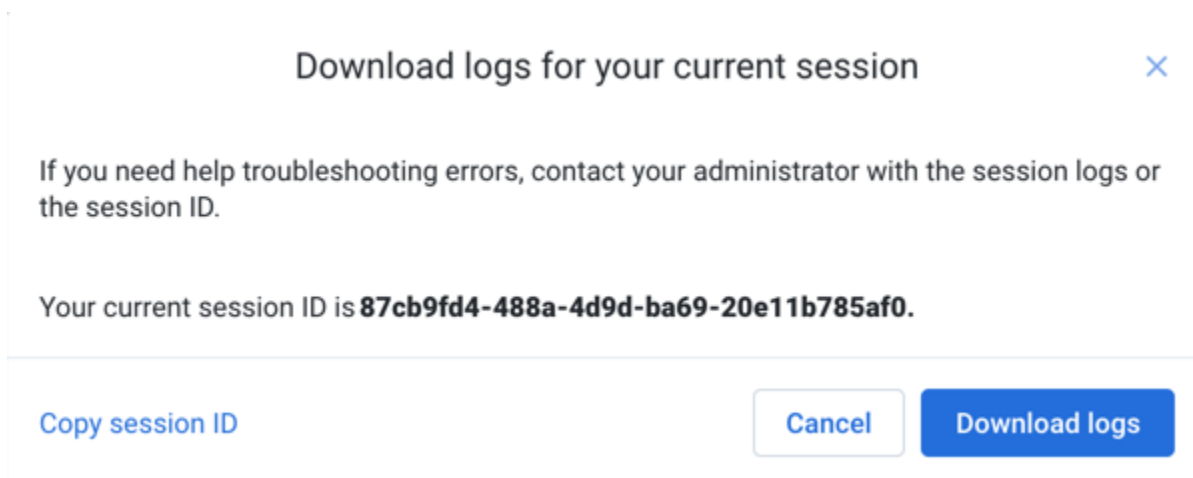
# Download Logs Dialog

You can download logs for your current session in Designer Cloud powered by Trifacta® Enterprise Edition. From the Help menu, select **Download logs**.

**NOTE:** The data downloaded for end users from this dialog is encrypted by default.

**NOTE:** For more information on disabling this feature, see *Configure Support Bundling*.

Administrators have a separate interface for downloading log files, which provides access to a wider set of logging data. For more information, see *Admin Download Logs Dialog*.



**Figure: Download Logs Dialog**

The above dialog contains your current session ID.

**Tip:** Using your session ID, your Trifacta administrator can download a larger set of logs, which can be useful for troubleshooting issues with the product.

- To copy your session identifier to the clipboard, click **Copy session ID**.
- To download logs for your current session, click **Download logs**. Logs pertaining to your current session are bundled in a ZIP and downloaded.

**NOTE:** There is a defined limit to the size of each log file. For more information, please contact your Trifacta administrator.

For more information on the contents of this download, see *Support Bundle Contents*.

# Flows Page

Contents:

- All flows tab
- Owned by me tab
- Shared with me tab
- Folders

The Flows page displays the flows to which you have access and lets you create, review, and manage them. A **flow** is an object for bringing together and organizing the datasets, recipes, and other objects that you use to generate your results.

**NOTE:** Access to the Flows page in the application and privileges on flows is governed by roles in your workspace. For more information, please contact your workspace administrator.

- To create a new flow, click **Create Flow**. To rename the new flow, click the `Untitled` value at the top of the page. See *Create Flow Page*.
- You can also access the flows that have been shared with you.

You can organize your flows into folders. A **folder** is simply a container for your flows. To create a folder, click **Create**. Then, select **Create Folder**. For more information, see "Flows" below.

🔍 Flows

Create...Import Flow

🔍 Search flows

/

All flowsOwned by meShared with me

| Name                                               | Owner         | Contains              | Last updated           |
|----------------------------------------------------|---------------|-----------------------|------------------------|
| 🔍 testflow                                         | Administrator | 6 Datasets, 0 Recipes | Today <div>Share</div> |
| 🔍 [732ac110] implementing cube() function from sql | Administrator | 3 Datasets, 6 Recipes | Today at 4:01 PM       |
| 🔍 [1e92ee00] [def96250] currencyFlow               | Administrator | 4 Datasets, 4 Recipes | Today at 2:58 PM       |
| 🔍 [1e92ee00] [def96250] currencyFlow               | Administrator | 4 Datasets, 4 Recipes | Today at 2:58 PM       |
| 🔍 Untitled Flow                                    | Administrator | 0 Datasets, 0 Recipes | Today at 1:50 PM       |

Figure: Flows Page

## All flows tab

This tab includes all flows accessible to the user, either as owner or collaborator.

## Owned by me tab

This tab contains the flows that you have created.

## Columns:

- **Name:** The name of the flow.
  - Click the flow name to review the flow, its datasets, and its recipes. See *Flow View Page*.
- **Owner:** Indicates the user who is the owner of the flow.
  - If the flow has been shared, you can click this link to see the users with whom the flow has been shared. See *Share Flow Dialog*.
- **Contains:** Count of datasets and recipes in the flow.
- **Last Updated:** Timestamp for the last time that the flow was modified.

## Actions:

- **Create:** From the Create menu, choose to create a flow or a folder for holding flows.
  - For more information on creating a flow, see *Create Flow Page*.
  - For more information on folders, see "Folders" below.
- **Import:** From the context menu, select **Import Flow** to import a flow into this instance. See *Import Flow*.
- **Search:** To search flow names, enter a string in the search bar. Results are highlighted immediately in the Flows page.
- **Sort:** Click a column header to sort the display by the column's entries.

## Flow options:

The following options are available on the right side of a flow's entry:

- **Share:** Enable other users to collaborate on your flows with you or create copies of your flow for their personal use. See *Share Flow Dialog*.
- **Rename:** Change the name and description of the flow.
- **Schedule:** To add a scheduled execution of the recipes in your flow:
  1. Define the scheduled time and interval of execution at the flow level. See *Add Schedule Dialog*.
  2. Define the scheduled destinations for each recipe through its output object. These destinations are targets for the scheduled job. See *Flow View Page*.
- **Email notifications:** Configure types of jobs that generate success or failure emails and who receives the messages. See *Manage Flow Notifications Dialog*.
- **Duplicate:** Create a copy of the flow. The copied flow is owned by the user who copied it.
- **Move:** Move the flow to a new or existing folder. See "Folders" below.
- **Export:** (Available to flow owner only) Export the flow from Designer Cloud powered by Trifacta Enterprise Edition . For more information, see *Export Flow*.
- **Delete:** Delete the flow.

**Deleting a flow removes all recipes and related objects contained in the flow. If copies of these objects exist in other flows, they are not touched. Imported datasets are not deleted by this action.**

For flows that have been shared with you, this command removes your access to them. To regain access, the owner of the flow must share it with you again.

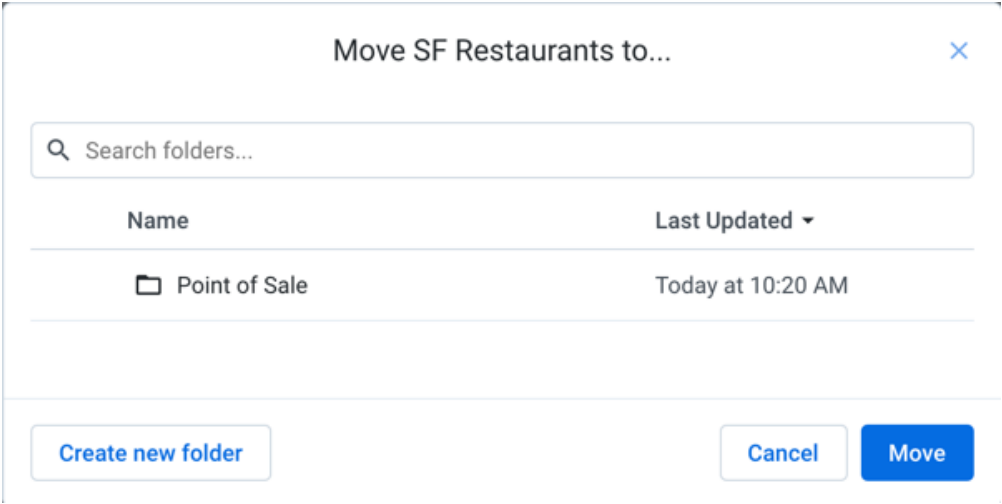
## Shared with me tab

If other users have shared flows with you, you can access them through the Shared with Me tab. Available options are very similar to the Owned by Me tab.

When a flow is shared with you, you are a collaborator in the flow. There are a few restrictions on how you can interact with a shared flow and its assets. See *Overview of Sharing*.

## Folders

You can use folders to organize your flows. For example, you can use folders to group flows by project, by source of data, or by other meaningful grouping.



**Figure: Moving a flow to a folder**

**Limitations:**

- Each flow in a folder is an independent object. Permissions can vary between flows in a folder and should be reviewed after adding them.
- You can only move flows that you own.
- You cannot create nested folders.
- You cannot share folders or modify permissions at the folder level.
- Folders cannot be exported and imported.

**Actions:**

- To create a folder, click the **Create** button. Then, select **Create Folder**. Enter a name and description for the folder. These values appear in the application. Click **Create**.

**Tip:** When you move a flow, you can optionally choose to create a new folder for it.

- To move a flow to a folder, select **Move** from the context menu on the right side of the screen for the flow.
  - To move the flow, select the name of the flow. Click **Move**.
  - To search folder names, enter your search string in the Search textbox.
  - To move the flow into a new folder, click **Create new folder**. Enter a meaningful name and description for the folder. Select **Move to new folder**.
- To delete a folder, select **Delete Folder** from the context menu on the right side of the screen.

**This step deletes the folder and all flows within it. This step cannot be undone.**

**Folder options:**



For folders, the following options are available in the context menu.

- **Import flow:** Import the exported flow to Designer Cloud powered by Trifacta® Enterprise Edition
- **Edit Folder name and description:** Change the name and description of the folder.
- **Delete Folder:** Delete the folder and all flows within it.

**Deleting a folder also removes any flows within it. This action cannot be undone.**

# Create Flow Page

You can use flows to organize your datasets and to track the jobs associated with them.

- A **flow** is a container for datasets, recipes, and related objects.
- To create a new flow click **Create Flow** in the *Flows Page*.

**Tip:** You can also create a flow while importing datasets. See *Import Data Page*.

## Steps:

1. In the Flows page, click **Create Flow**. A new flow is created, with the name `Untitled - x`, where `x` is a number.
2. Click the `Untitled - x` to enter a flow name and description.
3. From the Flow View page, you can add datasets to your flow, or import new ones. You may add multiple datasets at this time and add more later.
  - a. **Add dataset:** You can browse or search for datasets to add to your flow from the available ones.
    - i. This list includes all imported and reference datasets to which you have access.
    - ii. Select a different search filter to display all, imported, or reference datasets.
    - iii. To add a selected dataset, click the checkbox next to it.
  - b. **Import Datasets:** Click this link to import a new dataset into the application. After it is imported, it is automatically added to your flow. See *Import Data Page*.
4. When finished, click **Add**.
5. The datasets are displayed in the flow. For more information, see *Flow View Page*.

# Flow View Page

## Contents:

- *Flow View Organization*
  - *Top Bar*
    - *Flow context menu*
    - *Add Datasets to Flow*
  - *Flow Canvas*
    - *Canvas context menu*
    - *Canvas notes*
    - *Flow objects*
  - *Context Panel*
    - *View for Imported Datasets*
    - *View for Dataset with Parameters*
    - *View for Unstructured Datasets*
    - *View for Recipes*
    - *View for Outputs*
    - *View for Reference Datasets*
- 

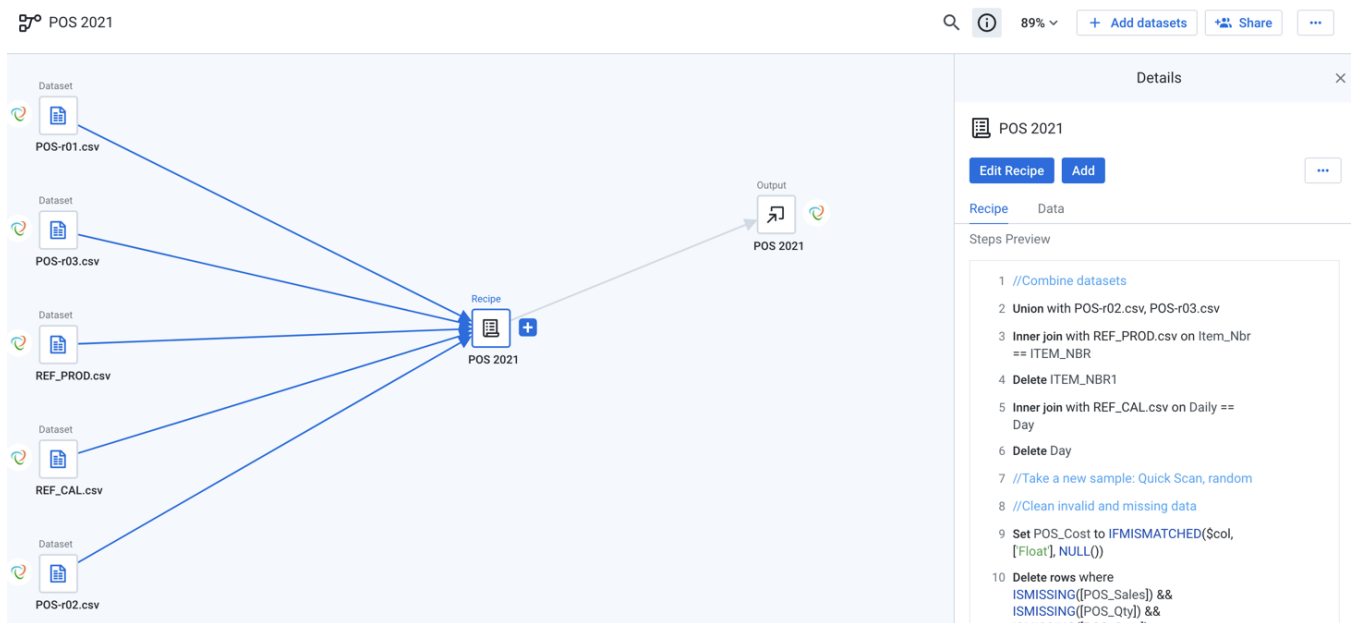
In Flow View, you can access and manage the objects that you have added to or created in the selected flow. You can perform a variety of actions to effectively manage flow development and job execution through a single page in the Designer Cloud® application .

**NOTE:** Access to this page in the application and privileges on its related objects is governed by roles in your workspace. For more information, please contact your workspace administrator.

**If you have enabled Deployment Manager, avoid making changes in Flow View on a Production instance of the platform.**

- **Scheduling executions through Flow View in a Prod environment is not supported. Job executions must be executed through the APIs. See *API Workflow - Deploy a Flow*.**
- **Some Flow View options may not be available in a Prod environment.**
- **You should apply changes to your flow in the Dev instance and then re-deploy to Production. For more information, see *Overview of Deployment Manager*.**

**NOTE:** If the displayed flow has been shared with you, some options are not available.



**Figure: Flow View page**

## Flow View Organization

Flow View is organized into the following basic areas:

| Area                 | Description                                                                                                                                                                                   |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Top bar</b>       | At the top of the screen, you can access a menu of options, which enable adding objects to your flow, configuring aspects of your flow, and other flow management tasks. See "Top bar" below. |
| <b>Flow canvas</b>   | The primary workspace of Flow View is the flow canvas, where you build and organize the objects in your flow. See "Flow Canvas" below.                                                        |
| <b>Context panel</b> | When you select one or more objects in your flow, the object details and relevant context menu options are available in the right-hand panel. See "Context Panel" below.                      |

## Top Bar

From the bar at the top of Flow View, the following options are available:

**Tip:** To rename the flow, click the flow name at the top of Flow View.

- **Search icon:** Click this icon to search for objects in your flow. See *Flow Search Panel*.
- **Details icon:** Click to toggle display of the details panel. This setting is saved for the individual user.
- **Zoom menu:** Flow View attempts to zoom the canvas to display as much of the flow as possible. As needed, you can change the level to zoom in or zoom out on the canvas.

**Tip:** You can access these zoom controls through the context menu for the flow canvas.

- **Zoom in:** Zoom in 10% on the canvas to focus on greater detail.
- **Zoom out:** Zoom out 10% from the canvas to see more of it.
- **Zoom to fit:** Change the zoom level to fit the objects of your flow.
- **Zoom to selection:** Zoom to center the selected object on the canvas.

- **25%, 50%, or 100%:** Change the zoom level to one of the preset levels.
- **Add Datasets:** Click to add new datasets to the flow. See "Add Datasets to Flow" below.
- **Share:** Click **Share** to collaborate with others on the same flow.

**NOTE:** When a flow containing one or more connections is shared, its connections are also shared. By default, credentials are included. If the sharing of credentials has been disabled, the new users must provide their own credentials for the shared connection. See *Configure Sharing*.

**NOTE:** You cannot share with users outside of your current workspace, including any account that you may have in a different workspace.

See *Share Flow Dialog*.

## Flow context menu

- **Rename:** (Available to flow owner only) Change the name and description of the flow.
- **Schedule:** To add a scheduled execution of the recipes in your flow:
  - Flow owners can create schedules. Other users must have the appropriate privileges to create schedules for the flow.
  - Define the scheduled time and interval of execution at the flow level. See *Add Schedule Dialog*.
    - After the schedule has been created, you can review, edit, or delete the schedule through the Clock icon.
    - The time of the next scheduled run is displayed in the time zone of the local browser.
  - Define the scheduled destinations for each recipe through its output object. These destinations are targets for the scheduled job. See *View for Outputs* below.
- **Parameters:** Create and manage recipe parameters, as well as specify overrides for them. See *Manage Parameters Dialog*.
- **Webhooks:** You can define tasks to update third-party applications of the results of jobs executed from this flow. For more information, see *Create Flow Webhook Task*.

**NOTE:** Webhooks may need to be enabled in your environment. For more information, see *Workspace Settings Page*.

- **Email notifications:** Configure types of jobs that generate success or failure emails and who receives the messages. See *Manage Flow Notifications Dialog*.

**NOTE:** This feature uses an SMTP email server to send messages. For more information on configuring the server, see *Enable SMTP Email Server Integration*.

**NOTE:** This feature may need to be enabled in your environment. For more information, see *Workspace Settings Page*.

- **Optimization settings:** You can configure optimizations of your job executions of your flow. For more information, see *Flow Optimization Settings Dialog*.
- **Auto arrange:** Re-organize the custom node layout back to a fresh system generated layout. This option may be helpful for rearranging flow layouts that were originally created in classic Flow View.

**The Auto arrange option cannot be undone.**

- **Duplicate:** Create a copy of the flow.
  - The copied flow is independent of the source flow.
  - Optionally, you can duplicate the datasets from the original flow in the copy.
- Duplicating datasets has some implications on shared flows. See *Overview of Sharing*.

**NOTE:** For flows using parameterized datasets, you should duplicate the datasets, which creates separate copies of parameters and their values in the new flow. If datasets are not copied, then parameter changes in the copied flow modify the values in the source flow.

- **Move:** Move the flow to a new or existing folder. See *Flows Page*.
- **Export:** (Available to flow owner only) Export the flow for archive or transfer. For more information, see *Export Flow*.
- **Delete:** (Available to flow owner only) Delete the flow.

**Deleting a flow cannot be undone.**

**Deleting a flow removes all recipes that are contained in the flow. If copies of these objects exist in other flows, they are not touched. Imported datasets are not deleted by this action.**

## Add Datasets to Flow

From the Flow View page, you can add data through the following objects:

- Imported datasets - data sourced from outside the platform
- Reference datasets - dataset objects that are created from the output of a recipe in the current flow or another flow

**NOTE:** For long-loading relational sources, you can track progress of the load. While the data is loading, some recipe and flow options are disabled.

For more information on enabling long-loading, see *Configure JDBC Ingestion*.

These independent objects can be joined, unioned, or referenced by other datasets in the flow. For more information on these object types, see "View for Objects" below.

×

### Add Datasets to Flow

All (20)

Imported (18)

Reference (2)

|                                     |  | NAME                        | SOURCE | LAST UPDATED     |
|-------------------------------------|--|-----------------------------|--------|------------------|
| <input type="checkbox"/>            |  | TESTDATA                    | oracle | Today at 8:48 AM |
| <input checked="" type="checkbox"/> |  | POS-PivotTable3.xlsx/Sheet1 | HDFS   | Today at 8:40 AM |
| <input type="checkbox"/>            |  | POS-schema.csv              | HDFS   | Today at 8:30 AM |
| <input type="checkbox"/>            |  | POS-r01.txt                 | HDFS   | Today at 8:30 AM |
| <input type="checkbox"/>            |  | POS-r02.txt                 | HDFS   | Today at 8:30 AM |
| <input type="checkbox"/>            |  | REF_CAL.txt                 | HDFS   | Today at 8:30 AM |

Import Datasets

Cancel

Add

**Figure: Add datasets to current flow**

- Search for or select the dataset to add.
  - Use the page view controls to browse for other datasets, or select the appropriate tab to filter the list to a specific type of object.
  - To import new datasets from external sources, click **Import Datasets**. See *Import Data Page*.
- When you have made your selections, click **Add**.
- The object or objects are added as a new object in flow view.

For large relational datasets, you can monitor the import process through the Flow View page.

**NOTE:** This feature may require enablement in your deployment. See *Configure JDBC Ingestion*.

For more information, see *Overview of Job Monitoring*.

## Flow Canvas

The central workspace of Flow View, the **canvas** is where you add and arrange the objects in your flow. In the canvas, you can select one or more objects at a time, drag them to reposition them on screen, and zoom in or out to focus on your current area of development.

When you add an object to the canvas, an icon representing it is added to the flow canvas. This object can be repositioned as needed.

**Tip:** The relative position of objects on the flow view canvas is preserved between screen updates. On refresh, the window on the canvas is repositioned based on the leftmost object on the canvas to focus on the flow to other objects from that one.

- If you create an object from another object, such as an output from a recipe, an arrow connects the recipe to the output.
- For any object, the objects on which it depends are displayed to the left of the original object, and there is a line from the preceding objects to the original object.

**Tip:** When you run a job for a recipe, all of the recipes steps for the preceding datasets are executed as part of the job, and only the results of the terminal dataset are generated.

**NOTE:** Objects marked with a red dot indicate a problem with the object's configuration. Please select the object to begin investigating the error. Error information may be displayed in the right panel.

### Select:

- Click the icon for an object to select it.
  - Object details for the specific type of object are displayed in the context panel.
  - Right-click the object to open its context menu.
- To select multiple objects:
  - Click and drag over a set of objects.
  - To select a discrete set of objects, press **CTRL/COMMAND** + click the objects.
  - When you select multiple objects, the objects are listed in the context panel with options that are applicable to all of the objects. Some objects, like notes, do not have a context panel.
  - Right-click to display the context menu of options that are applicable to the selected objects.
- See "Context Panel" below.

### Move:

- To move an object, click and drag it to a new location. Any arrows connecting to the object are repositioned as well.
- To move multiple objects, select them and then drag them to a new location.

### Canvas context menu

When you right-click an empty part of the canvas, the following options are available:

- **Select All:** Select all objects on the canvas. Options that are relevant to all of the objects are displayed on the context panel.

**Tip:** You can drag these objects to reposition them together.

- **Auto arrange:** Re-organize the custom node layout back to a fresh system generated layout. This option may be helpful for rearranging flow layouts that were originally created in classic Flow View.

**The Auto arrange option cannot be undone.**

- **Add note:** Add a note with text or emojis to the canvas. See "Canvas notes" below.
- **Zoom:** Zoom in or zoom out on the canvas as needed. See "Top Bar" above for more details.

### Canvas notes

As you develop your flow, you can add helpful notes on the canvas in various sections of the flow. For example, if you are collaborating with another user, you can leave status information about objects that are still in progress.

**Tip:** You can drag notes like other objects on the canvas, so they can be repositioned with the related flow object.



From a note's context menu, you can edit or delete the note.

## Flow objects

In the flow canvas, you work with the following types of objects:

- Connections
- Imported datasets
  - Unstructured datasets
- Recipes
- Outputs
- Notes
- Reference datasets
- Datasets with parameters

### Datasets:

- To begin working with data:
  - Click **Add datasets**. Locate your source data and import it into your flow. See "Add Datasets to Flow" above.
  - In Flow View, select the **imported dataset** on the flow canvas. Then, in the context panel, click **Add new recipe**. A new, empty recipe is associated with the dataset.

**Tip:** Double-click an imported dataset to see a preview of it. Some datasets cannot be previewed.

- To open in the Transformer page, click the recipe and select **Edit Recipe**. See *Transformer Page*.
- When created, these objects are connected together by lines flowing between them, which show the relationships between the objects in the flow.

### Recipes:

A recipe is a set of steps to transform source data into the results you desire.

**Tip:** Double-click any recipe to edit it. See *Transformer Page*.

A recipe can be created from the following objects:

- An imported dataset, as described above.
- A **reference dataset** is an object that has been pulled into a flow from another flow. See "View for Reference Datasets" below.
- Another recipe. You can chain together recipes. For example, you may have a set of steps that you always apply at the beginning of transforming a specific type of feed. This recipe can be added into each flow as the first recipe chained to an imported dataset of that feed type.

### Output objects:

The following objects can be created off of a recipe:

- An **output** object is a set of publishing targets for which you can execute jobs.
- A **reference** object is a reference to one of your flow's recipes that can be used in another flow. When a reference object is created, the target flow receives the output of the executed recipe.

- In the target flow, this object appears as a **reference dataset**.
- When a reference dataset is used in a flow, the target flow receives the output of the executed recipe.

For more information on these objects, see *Object Overview*.

## Context Panel

Select an object from your flow to open an object-specific panel on the right side of the screen.

- When multiple objects are selected, the displayed details and options apply to all of the objects.
- If the selected objects are of different types, the available options are limited.

**Tip:** You can right-click any object in Flow View to see the list of available actions that appear when you select it and choose from the right panel.

Depending on the type of object or objects that you select in the canvas, the view in the context panel changes:

**Tip:** The object on the canvas and the context panel display the same set of context menu options. For more information on these options, see the links below.

### View for Imported Datasets

See *View for Imported Datasets*.

### View for Dataset with Parameters

For datasets with parameters, the context panel and available options are different from non-parameterized imported datasets. See *View for Dataset with Parameters*.

### View for Unstructured Datasets

If detecting structure has been disabled for your imported dataset, the structure detecting steps are broken out into the first steps of a recipe that is auto-generated for you. You can review and modify them within the recipe. See *View for Unstructured Datasets*.

### View for Recipes

See *View for Recipes*.

### View for Outputs

Outputs are created from recipes. See *View for Outputs*.

### View for Reference Datasets

A reference dataset is sourced from the outputs of another recipe. A reference dataset can be used in a flow other than the source flow. See *View for Reference Datasets*.

# View for Imported Datasets

When you select an imported dataset in Flow View, you can review its details in the context panel and select options from its context menu.



**Figure: Imported Dataset icon**

## Icon context menu

The following menu options are available when you select the plus icon next to the imported dataset:

- **Add new recipe:** Add a new recipe extending from the current recipe. This new recipe operates on the outputs of the original recipe.
- **Add Join:** Add a join step as the new last step to the recipe. See *Join Window*.
- **Add Union:** Add a union step as the new last step to the recipe. See *Union Page*.

## Details options

The following options are available in the details context menu when you select an imported dataset.

- **Add:**
  - **Recipe:** Add a recipe for this dataset.
  - **Join:** Join this dataset with another recipe or dataset. If this dataset does not have a recipe for it, a new recipe object is created to store this step. See *Join Window*.
  - **Union:** Union this dataset with one or more recipes or datasets. If this dataset does not have a recipe for it, a new recipe object is created to store this step. See *Union Page*.
- **View dataset details:** Explore details of the dataset. See *Dataset Details Page*.
- **Replace:** Replace the dataset with a different dataset or reference dataset.
- **Replace with dataset with Parameters:** For datasets that are not parameterized, you can choose to replace with datasets with parameters.

**Tip:** You may find it useful to create your recipes with a single static dataset and then later replace with a dataset with parameters.

- **Edit name and description:** (Available to flow owner only) Change the name and description for the object.
- **Edit custom SQL:** After you have created a dataset using custom SQL, you can modify the SQL used to construct the imported dataset. See *Create Dataset with Parameters*.
- **Edit parameters:** If your dataset contains parameters, you can change the parameters and their default values.
- **Remove structure:** (If applicable) Remove the initial parsing structure. When the structure is removed:

- The dataset is converted to an unstructured dataset. An **unstructured dataset** is the source data converted into a flat file format.
- All steps to shape the dataset are removed. You must break up columns in manual steps in any recipe created from the object.
- See *View for Unstructured Datasets*.
- **Remove from Flow:** Remove the dataset from the flow.
  - All dependent flows, outputs, and references are not removed from the flow. You can replace the source for these objects as needed.

**NOTE:** References to the deleted dataset in other flows remain broken until the dataset is replaced.

**Tip:** You can also right-click the imported dataset to view all the menu options.

When you select an imported dataset, you can preview the data contained in it, replace the source object, and

Details

POS-schema.csv

Add

View dataset details

Data Preview

| # | Item_Nbr | # | Store_Nbr | # | WM_Week |
|---|----------|---|-----------|---|---------|
|   | 381000   |   | 1         |   | 201050  |
|   | 325000   |   | 2         |   | 201049  |
|   | 325000   |   | 2         |   | 201049  |
|   | 403000   |   | 2         |   | 201049  |
|   | 449000   |   | 2         |   | 201049  |
|   | 490000   |   | 2         |   | 201049  |
|   | 560000   |   | 2         |   | 201049  |

Type

HDFS

Location

hdfs:///trifacta/uploads/1/26149d74-6cf8-46a7-aa8d-a1f02cc3c242/POS-schema.csv

File Size

1.72MB

Size

29 columns · 5 types

Updated

Today at 8:30 AM

more from the right-side panel.

Figure: Imported Dataset view

Key Fields:

- **Data Preview:** In the Data Preview window, you can see a small section of the data that is contained in the imported dataset. This window can be useful for verifying that you are looking at the proper data.

**Tip:** Click the preview to open a larger dialog, where you can select and copy data.

- **Type:** Indicates where the data is sourced or the type of file.
- **Location:** Path to the location of the imported dataset.
- **File Size:** Size of the file. Units may vary.

## Column Data Type Inference:

**NOTE:** This field is only applicable to datasets imported from relational sources.

- **enabled** - Data types have been applied to the dataset during import.
- **disabled** - Data types were not globally applied to the dataset during import. However, some columns may have had overrides applied to them during the import process. See *Import Data Page*.

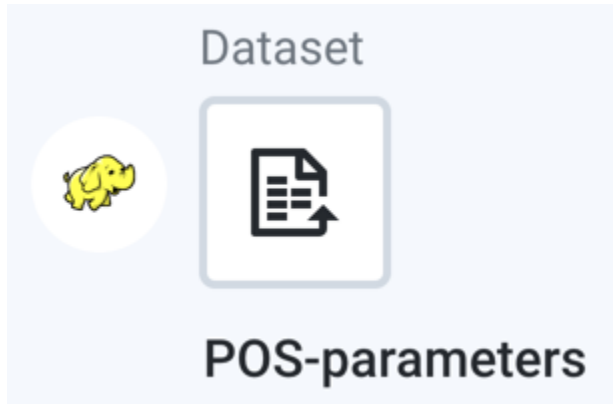
For more information, see *Configure Type Inference*.

- **ConnectionName:** If the data is accessed through a connection, you can click this link to review connection details in the right-side panel. See View for Connections below.
- **More details:** Review details on the flows where the dataset is used.

# View for Dataset with Parameters

When you select a dataset with parameters in Flow View, additional options are available in its context menu and the Details panel.

For more information on these objects, see *Overview of Parameterization*.



**Figure: Dataset with Parameters icon**

## Icon context menu

The following menu options are available when you select the plus icon next to the dataset with parameters:

- **Add new recipe:** Add a new recipe extending from the current recipe. This new recipe operates on the outputs of the original recipe.
- **Add Join:** Add a join step as the new last step to the recipe. See *Join Window*.
- **Add Union:** Add a union step as the new last step to the recipe. See *Union Page*.

## Details options

The following options are available in the Details context menu when you select a dataset with parameters:

- **Add:**
  - **Recipe:** Add a recipe for this dataset.
  - **Join:** Join this dataset with another recipe or dataset. If this dataset does not have a recipe for it, a new recipe object is created to store this step. See *Join Window*.
  - **Union:** Union this dataset with one or more recipes or datasets. If this dataset does not have a recipe for it, a new recipe object is created to store this step. See *Union Page*.
- **View dataset details:** Explore details of the dataset. See *Dataset Details Page*.
- **Edit name and description:** (Available to flow owner only) Change the name and description for the object.
- **Edit parameters:** If your dataset contains parameters, you can change the parameters and their default values.
- **Remove structure:** (If applicable) Remove the initial parsing structure. When the structure is removed:
  - The dataset is converted to an unstructured dataset. An **unstructured dataset** is the source data converted into a flat file format.
  - All steps to shape the dataset are removed. You must break up columns in manual steps in any recipe created from the object.
  - See *View for Unstructured Datasets*.
- **Remove from Flow:** Remove the dataset from the flow.
  - All dependent flows, outputs, and references are not removed from the flow. You can replace the source for these objects as needed.


**NOTE:** References to the deleted dataset in other flows remain broken until the dataset is replaced.

**Tip:** You can also right-click the dataset with parameters to view all the menu options.

When you select a dataset with parameters in Flow View, you can review the parameters that have been specified for the selected dataset in the right panel.

Details

×

 Dataset with Parameters

Add

View dataset details

...

Details

Parameters

Path

hdfs://trifacta/uploads/1/26149d74-6cf8-46a7-aa8d-a1f02cc3c242/POS-r .✱ {digit}{digit} .txt

Parameters

Dataset has 1 parameters

.✱ {digit}{digit}

| Type | Pattern                                            |
|------|----------------------------------------------------|
| Rule | matches against trifacta pattern: `{digit}{digit}` |

**Figure: Parameters tab in Flow View**

**Key Fields:**

Details tab:



- **Type:** Indicates where the data is sourced or the type of file.
- **Size:** Size of the file. Units may vary.
- **More details:** Review details on the flows where the dataset is used.

Parameters tab:

- **Path:** Full path to the target location.
- **Parameters:** Indicates the number of parameters in the dataset.
- **Type:** Type of the pattern.
- **Rule:** Rule applied to the pattern.

# View for Recipes

## Contents:

- *Recipes in the Canvas*
    - *Icon context menu*
    - *Recipes flagged for review*
  - *Recipe Details*
    - *Details options*
    - *Recipe tab*
    - *Data tab*
    - *Target tab*
- 

For each recipe in Flow View, you can review or edit its steps or create new recipes altogether. You can also create references to the recipe, modify outputs, and create new recipes off of the recipe.

## Recipes in the Canvas



**Figure: Recipe icon**

## Icon context menu

The following menu options are available when you select the plus icon next to the recipe:

- **Add new recipe:** Add a new recipe extending from the current recipe. This new recipe operates on the outputs of the original recipe.
- **Create Output to run:** Create an output for the recipe. See *Create Outputs*.
- **Create Reference Dataset:** Create a reference dataset of the recipe. For more information, see *View for Reference Datasets*.
- **Append Join:** Add a join step as the new last step to the recipe. See *Join Window*.
- **Append Union:** Add a union step as the new last step to the recipe. See *Union Page*.

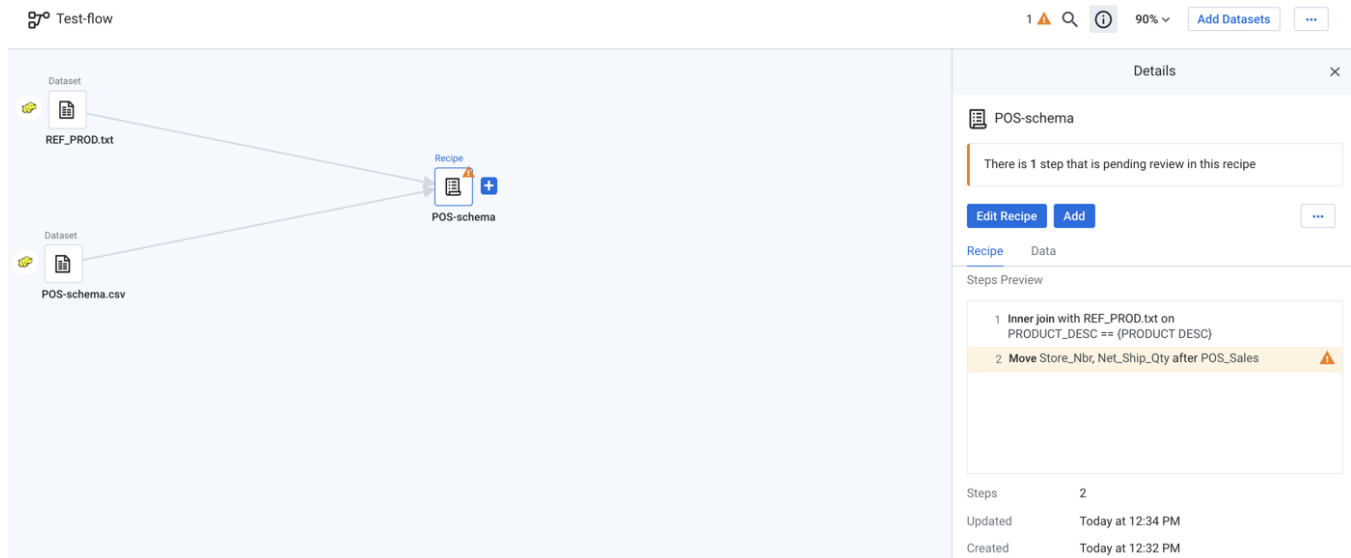
## Recipes flagged for review

You can flag a step or steps in a recipe that requires additional review. Whenever you flag steps or a reference dataset for review, the flow view page is highlighted with a warning icon on each recipe and reference dataset node.

**NOTE:** A recipe that has steps flagged for review cannot be executed. Additional limitations are placed on editing the recipe when some steps are flagged. For more information, see *Flag for Review*.

## Actions:

If you hover the mouse over any recipe and click the warning indicator, the corresponding step that has pending for review icon is highlighted in the Details panel.



**Figure: Recipe icon indicator**

**NOTE:** The Flow View page header summarizes the total number of flagged steps and recipes that are pending for review.


## Recipe Details

When you select a recipe:

- You can create an output object.
- You can create a reference object.
- The following options are available in the context panel.

Details

×

 POS-r01

Edit Recipe

Add new Recipe

...

Recipe

Data

Steps Preview

1 Union with POS-r02.txt, POS-r03.txt

2 Inner join with REF\_PROD.txt on Item\_Nbr == ITEM\_NBR

3 Delete ITEM\_NBR1

4 Inner join with REF\_CAL.txt on Daily == Day

5 Delete Day

Steps

5

Updated

Today at 12:18 PM

Created

Today at 12:01 PM

**Figure: Recipe view**

**NOTE:** If you flag a step for review, the corresponding step is highlighted with a warning icon and displayed in the Details panel. The warning icon in the Details panel header shows the total count of steps that are pending review.

### Details options

The following options are available in the details context menu when you select a recipe.

- **Edit Recipe:** Open the recipe and begin editing. See *Transformer Page*.
- **Add:**
  - **Recipe:** Add a recipe for this dataset.

- **Branch Recipe:** Add a new recipe branching from the recipe. This new recipe operates on the outputs of the recipe from which it was branched.
- **Output:** If the recipe does not have an output, you cannot run a job for it. Select this option to create a default output for your recipe.
- **Reference:** Add a reference dataset from this recipe. A reference dataset can be used in other flows. For more information, see *View for Reference Datasets*.
- **Join:** Join this dataset with another recipe or dataset. If this dataset does not have a recipe for it, a new recipe object is created to store this step. See *Join Window*.
- **Union:** Union this dataset with one or more recipes or datasets. If this dataset does not have a recipe for it, a new recipe object is created to store this step. See *Union Page*.
- **Edit name and description:** (Available to flow owner only) Change the name and description for the object.
- **Append Join:** Add a join step as the new last step to the recipe. See *Join Window*.
- **Append Union:** Add a union step as the new last step to the recipe. See *Union Page*.
- **Assign Target to Recipe:** As needed, you can create a target and assign it to this recipe. For more information, see *Create Target*.
- **Remove Target:** Remove the currently assigned target from this recipe.

**NOTE:** You can toggle between **Assign Target to Recipe** and **Remove Target** to assign a target and to remove the target from the recipe.

- **Change input:** Change the input dataset associated with the recipe.

**NOTE:** This action substitutes only the primary input from a recipe, which does not include any datasets that are integrated from joins, unions, lookups, or other multi-dataset options.

**Tip:** You can swap in dynamic datasets for static datasets, if needed. This feature may not be enabled in your environment. See *Miscellaneous Configuration*.

- **Make a copy:** Create a copy of the recipe and its related objects. You can create the copy with the same inputs or without inputs at all. The copied recipe is owned by the user who copied it.

**NOTE:** The copied recipe is independent of the source recipe. Optionally, you can duplicate the datasets from the original recipe in the copy.

- **Move:** Move the recipe to a different flow, or create a new flow to contain it.
- **Download Recipe:** Download the recipe in Wrangle format to your local desktop.
- **Delete:** Delete the recipe.

**Tip:** When a recipe is deleted, all samples associated with the recipe are also removed, which may significantly reduce the total volume of storage you are using.

**This step cannot be undone.**

**Tip:** You can also right-click the recipe to view all the menu options.

## Recipe tab

Preview the first steps in the recipe.

## Key Fields:

- **Steps:** Total count of the steps in the recipe.

## Data tab

Preview the data as reflected by the recipe.

**NOTE:** To render this data preview, some of the data must be loaded, and all steps in the recipe must be executed to generate the preview. Some delays may be expected.

## Key Fields:

- **Size:** Total count of columns and data types in the dataset.

## Target tab

When a target has been assigned for this recipe, you can review its schema information in the Target tab. This tab appears only after a target has been assigned to the recipe.

To remove the current target, select **Remove Target** from the context menu.

## Columns:

- **Position:** Left-to-right position of the column in the target.
- **Name:** Name of the column in the target.
- **Type:** Trifacta data type of the column in the target.

For more information, see *Overview of RapidTarget*.

# View for Reference Datasets

## Contents:

- *Create reference dataset*
    - *Icon context menu*
    - *Details options*
  - *Add reference dataset in another flow*
    - *Icon context menu*
    - *Details options*
- 

A **reference dataset** is a reference to a recipe's output, which can be added to a flow other than the one where the recipe is located.

## Create reference dataset

Click the plus icon next to the recipe and select **Create Reference Dataset**.

**NOTE:** A reference dataset is a read-only object in the flow where it is referenced. A reference dataset must be created in the source flow from the recipe to use. For more information, see *View for Recipes*.



**Figure:** Reference Dataset icon in source flow

## Icon context menu

The following options are available when you right-click the reference dataset:

- **View details:** Explore details of the reference dataset. See *Dataset Details Page*.
- **Add to Flow:** Click to add the reference dataset to a new or existing flow.
- **Edit name and description:** Change the name and description for the object.
- **Delete Reference Dataset:** Remove the reference dataset from the flow.

**Deleting a reference dataset in the source flow causes all references to it to be broken in the flows where it is referenced. These broken references should be fixed by swapping in new sources.**

## Details options


The following options are available in the details context menu when you select a reference dataset.

- **Add to Flow:** Click to add the reference dataset to a new or existing flow.
- **Edit name and description:** (Available to flow owner only) Change the name and description for the object.

- **Delete Reference Dataset:** Remove the reference dataset from the flow.

Details

×

 POS-r01

Add to Flow...

...

Data Preview

| 🕒 Daily    | # Item_Nbr | # Store_Nbr | # WM_  |
|------------|------------|-------------|--------|
| 2013/02/08 | 381000     | 1           | 201050 |
| 2013/02/07 | 325000     | 2           | 201049 |
| 2013/02/07 | 325000     | 2           | 201049 |
| 2013/02/07 | 403000     | 2           | 201049 |
| 2013/02/07 | 449000     | 2           | 201049 |
| 2013/02/07 | 490000     | 2           | 201049 |
| 2013/02/07 | 560000     | 2           | 201049 |

Updated

Today at 10:18 AM

Created

Today at 10:18 AM

Used in

0 Flows [More details](#)

**Figure: Reference Dataset view**

The following fields appear in the right panel.

**Key Fields:**

**Used In:** Indicates the number of flows where the reference appears. If this number is greater than one, click **More details** to review the flows. See *Dataset Details Page*.

**More details:** Review details on the flows where the dataset is used. See *Dataset Details Page*.

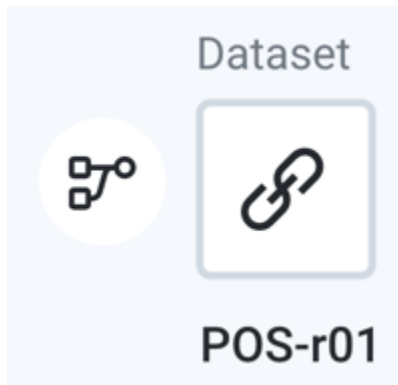


## Add reference dataset in another flow

After you have created a reference dataset, you can use it other flows.

### Options:

- In the source flow, select the reference dataset, and click **Add to Flow** in the Details options.
  - See above procedures on creating a reference dataset.
  - From the **Add - x to** dialog, where **x** is the name of the reference object, select the required flow or click **Create new flow** to add the reference dataset to the flow.
  - For more information on creating new flows, see *Create Flow Page*.
- In a target flow, you add a reference dataset like other datasets. In Flow View, when you add a dataset, reference datasets are listed under the Reference tab. For more information, see *Flow View Page*.



**Figure: Reference Dataset icon in another flow**

### Icon context menu

The following menu options are available when you select the plus icon next to the dataset:

- **Add new recipe:** Add a new recipe extending from the reference dataset.
- **Add Join:** Add a join step as the new last step to the recipe. For more information, see *Join Window*.
- **Add Union:** Add a union step as the new last step to the recipe. For more information, see *Union Page*.

### Details options

The following options are available in the details context menu when you select the reference dataset.


- **Add:**
  - **Recipe:** Add a recipe for this dataset.
  - **Join:** Join this dataset with another recipe or dataset. If this dataset does not have a recipe for it, a new recipe object is created to store this step. See *Join Window*.
  - **Union:** Union this dataset with one or more recipes or datasets. If this dataset does not have a recipe for it, a new recipe object is created to store this step. See *Union Page*.
- **View in library:** Review details on the flows where the dataset is used.
- **Go to original reference:** Open the flow containing the original reference for this dataset.
- **Remove from Flow:** Remove the reference dataset from the flow.
  - All dependent flows, outputs, and references are not removed from the flow. You can replace the source for these objects as needed.

**NOTE:** References to the deleted dataset in other flows remain broken until the dataset is replaced.

**Tip:** You can also right-click the reference dataset to view all the menu options.

Details

×

 POS-schema

Add

View in library

...

Data Preview

| # Item_Nbr | # Store_Nbr | # WM_Week | 🕒 Daily    |
|------------|-------------|-----------|------------|
| 381000     | 1           | 201050    | 2013/02/08 |
| 325000     | 2           | 201049    | 2013/02/07 |
| 325000     | 2           | 201049    | 2013/02/07 |
| 403000     | 2           | 201049    | 2013/02/07 |
| 449000     | 2           | 201049    | 2013/02/07 |
| 490000     | 2           | 201049    | 2013/02/07 |
| 560000     | 2           | 201049    | 2013/02/07 |
| 572000     | 2           | 201049    | 2013/02/07 |

Source Flow

[Test\\_flow](#)

Updated

Today at 2:53 PM

Created

Today at 2:52 PM

**Figure:** View for referenced dataset in a new flow

**NOTE:** Reference datasets marked with a red dot no longer have a source dataset for them in the other flow. These upstream dependencies should be fixed. See *Fix Dependency Issues*.

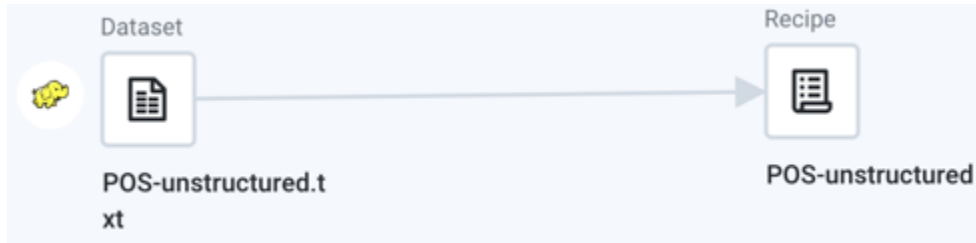
The following fields appear in the right panel.

**Key Fields:**

**Source Flow:** Flow that contains the dataset. Click the link to open the Flow View page for that dataset.

# View for Unstructured Datasets

An **unstructured dataset** is an imported dataset that does not contain any initial parsing steps. All parsing steps must be added through recipes that are applied to the dataset. During the import process, you disable the initial steps that are applied to imported datasets. Instead, these steps are added as the first steps of the auto-generated recipe that appears with the dataset in Flow View.



**Figure: Unstructured Dataset icons**

## Icon context menu

The following menu options are available when you click plus icon next to the unstructured dataset:

- **Add new recipe:** Add a recipe for this dataset.
- **Add Join:** Join this dataset with another recipe or dataset. If this dataset does not have a recipe for it, a new recipe object is created to store this step.
- **Add Union:** Union this dataset with one or more recipes or datasets. If this dataset does not have a recipe for it, a new recipe object is created to store this step.

## Details options


- **Add:**
  - **Recipe:** Add a recipe for this dataset.
  - **Join:** Join this dataset with another recipe or dataset. If this dataset does not have a recipe for it, a new recipe object is created to store this step.
  - **Union:** Union this dataset with one or more recipes or datasets. If this dataset does not have a recipe for it, a new recipe object is created to store this step.
- **View dataset details:** Explore details of the dataset. See *Dataset Details Page*.
- **Edit name and description:** (Available to flow owner only) Change the name and description for the object.
- **Remove from Flow:** Remove the dataset from the flow.
  - All dependent flows, outputs, and references are not removed from the flow. You can replace the source for these objects as needed.

**NOTE:** References to the deleted dataset in other flows remain broken until the dataset is replaced.

**Tip:** You can also right-click the unstructured dataset to view all the menu options.

Details

×

 POS-unstructured.txt

Add

View dataset details

...

Data Preview

```

Store_Nbr,Item_Nbr,WM_Week,Daily,Whse_Nbr,Whse_Name,
1,381000,201050,2013/02/08,0,Acme
Warehouse,7.00,7,4.97,0,0,Regular,24,.98,.71,.98
2,325000,201049,2013/02/07,0,Acme
Warehouse,.00,0,.00,0,7,Rollback,504,1.24,.93,1.24
2,325000,201049,2013/02/07,0,Acme
Warehouse,10.62,9,8.37,0,0,Regular,504,1.24,.93,1.24
2,403000,201049,2013/02/07,0,Acme
Warehouse,.00,0,.00,0,-1,n/a,432,1.24,.93,1.24
2,449000,201049,2013/02/07,0,Acme
Warehouse,7.00,6.5,5.58,72,0,Regular,456,1.24,.93,1.24

```

|           |                                                                             |
|-----------|-----------------------------------------------------------------------------|
| Type      | HDFS                                                                        |
| Location  | hdfs:///trifacta/uploads/1/33ecce5d-b242-4025-ad07-e11c14520fb6/POS-r01.txt |
| File Size | 285.95kB                                                                    |
| Updated   | Today at 2:54 PM                                                            |
| Created   | Today at 2:54 PM                                                            |
| Used in   | 1 Flow <a href="#">More details</a>                                         |

**Figure: Unstructured Dataset view**

#### Key Fields:

**Data Preview:** In the Data Preview window, you can see a small section of the data that is contained in the imported dataset. This window can be useful for verifying that you are looking at the proper data.

**Tip:** Click the preview to open a larger dialog, where you can select and copy data.

**Type:** Indicates where the data is sourced or the type of file.

**Location:** Path to the location of the imported dataset.

**File Size:** Size of the file. Units may vary.

# View for Outputs

## Contents:

- *Icon context menu*
  - *Details options*
  - *Jobs tab*
  - *Destinations tab*
- 

Associated with each recipe is one or more outputs. These publishing destinations can be configured through the context panel in Flow View. Through outputs, you can execute and track jobs for the related recipe.



**Figure: Output icon**

## Icon context menu

The following menu options are available when you right-click the output:

- **View Details:** View details.
- **Delete Output:** Remove this output from the flow. This operation cannot be undone.
  - Removing an output does not remove the jobs associated with the output. You can continue working with those executed jobs. See *Jobs Page*.
- **Run:** Click **Run** to queue for immediate execution a job for the manual destinations. You can track the progress and results of this task through the Jobs tab.

## Details options

The following options are available in the details context menu when you select an output.

**Delete Output:** Remove this output from the flow. This operation cannot be undone.

- Removing an output does not remove the jobs associated with the output. You can continue working with those executed jobs. See *Jobs Page*.
- **Edit:** Click this link to modify the selected destination's properties.

**Tip:** You can also right-click the output to view all the menu options.

Jobs tab

Details

×

Untitled recipe

Run

...

Jobs (1)

Destinations

Latest job

✓

Job 21 • Completed

Administrator • Finished Today at 10:15 AM

| # | Item_Nbr | # | Store_Nbr | # | WM_Week | ⌚ | Daily      |
|---|----------|---|-----------|---|---------|---|------------|
|   | 381000   |   | 1         |   | 201050  |   | 2013/02/08 |
|   | 325000   |   | 2         |   | 201049  |   | 2013/02/07 |
|   | 325000   |   | 2         |   | 201049  |   | 2013/02/07 |
|   | 403000   |   | 2         |   | 201049  |   | 2013/02/07 |
|   | 449000   |   | 2         |   | 201049  |   | 2013/02/07 |
|   | 490000   |   | 2         |   | 201049  |   | 2013/02/07 |
|   | 560000   |   | 2         |   | 201049  |   | 2013/02/07 |
|   | 573000   |   | 2         |   | 201049  |   | 2013/02/07 |
|   | 486000   |   | 3         |   | 201049  |   | 2013/02/07 |
|   | 488000   |   | 3         |   | 201049  |   | 2013/02/07 |

29 columns 8161 rows

Download

View details

The preview above shows the current data in the job destination. It might not reflect the output from this particular job run.

Figure: Jobs tab

Each entry in the Jobs tab identifies a job that has been Queued, Completed, or In Progress for the selected output. You can track the progress, success, or failure of execution. If you have executed no jobs yet, the Jobs tab is empty.

For the latest job:

- You can preview the job results. Click the Preview pane to open the results in a separate window.

**NOTE:** The Preview pane reflects the state of the data at the location specified for the output. If other jobs are also writing to this location, the state of the data may not reflect the output for this specific job.

**NOTE:** This section is not displayed if the job fails. The Preview may not be available if errors occur.

- To download the results from the output location, click the **Download** button.

**NOTE:** This button may not be available for some successful jobs.

- To view job details, click **View details**. For more information, see *Job Details Page*.

You can also view the previous jobs that have been executed for the selected output.

**Tip:** When you hover the mouse over a job link, you can review details of the job in progress. For more information, see *Overview of Job Monitoring*.

When a job has finished execution, click the link to the job to view results. For more information, see *Job Details Page*.

#### Actions:

For a job, you can do the following:

Click the job link to view the results of your completed job. For more information, see *Job Details Page*.

**Cancel job:** Select to cancel a job that is currently being executed

**Delete job:** Delete the job from the platform.

**Deleting a job cannot be undone.**

**NOTE:** This feature may not be enabled in your instance of the platform. For more information, please contact your Trifacta Administrator. See *Miscellaneous Configuration*.

**Download logs:** Download the logs for the job. If the job is in progress, log information is likely to be incomplete.

**Tip:** When jobs fail, the downloaded package includes additional configuration files and service logs to assist in debugging job execution issues. For more information, see *Support Bundle Contents*.


#### Destinations tab

The Destinations tab contains all configured destinations associated with the recipe.

- Manual destinations are executed when the job is run through the application interface.
- Scheduled destinations are populated whenever the flow's schedule is triggered and the destination's recipe is successfully executed.

Details

×

 REF\_PROD

Run

...

Jobs (3)

Destinations

Manual destinations

Edit

Create-CSV

hdfs:///REF\_PROD.csv

Environment

Photon

Profiling

yes

Scheduled destinations

Add

Add a scheduled destination to automatically run the Output when the flow is executed by a schedule.

**Figure: Destinations tab**

#### Key Fields:

- **(Action)-(Format):**
  - Field name describes the output action and the file format in which the results are written.
  - Field value is the location where the results are written.
- **Path:**
  - Full path to the target location.
  - If output parameters have been created for the destination, you can review their names in the path. For more information, see *Overview of Parameterization*.
- **Environment:** The running environment where the job is configured to be executed.
- **Profiling:** If profiling is enabled for this destination, this value is set to *yes*.

For more information, see *Run Job Page*.

#### Scheduled destinations:

If a schedule has been defined for the flow, these destinations are populated with results whenever the schedule is triggered and the associated recipe is successfully executed. If any input datasets are missing, the job is not run.

**NOTE:** Flow collaborators cannot modify publishing destinations.



See *Add Schedule Dialog*.

For more information, see *Overview of Automator*.

# Share Flow Dialog

You can manage access to a flow for other users through the Share Flow dialog. In Flow View, select **Share** from the context menu.

**Tip:** If groups have been enabled in your instance of the Designer Cloud powered by Trifacta platform , you can share flows and connections to LDAP groups. For more information, see *Configure Users and Groups*.

**Tip:** A workspace administrator has owner-level access to flows in the workspace. For more information, see *Workspace Admin Permissions*.

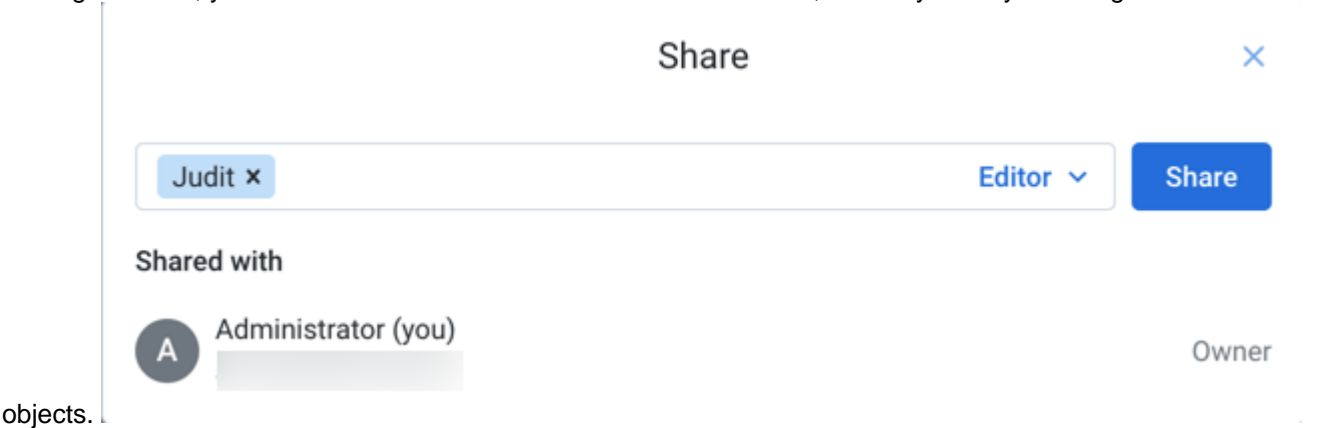
## Manage Access Tab

When you grant another user access to one of your flows, you both can work on the objects of the flow. You can take turns editing the recipes, which allows the team to more rapidly complete the work.

**NOTE:** When a user is given access to a flow, that user is considered a **collaborator** on the flow and has a smaller set of permissions than the **owner** of the flow.

**NOTE:** Any user may be given access to a flow. However, this user must have access to the underlying data. If the imported dataset is accessed from a private location, the user cannot access datasets in the shared flow. For more information, see *Share Connection Dialog*.

Through this tab, you can invite one or more collaborators to the flow, so that you may work together on the same



**Figure: Manage Access Tab**

- To add users as collaborators in your flow, start typing the name of a user with whom you'd like to collaborate. Select the user. Repeat this process to add multiple users.

**Tip:** You can paste a comma-separated list of email addresses to share to multiple users at the same time.

**NOTE:** For privacy reasons, search may not be available in some environments.

- To add a group of users as collaborators, select an entry that includes **(Group)**. Any user in the group has the same permissions as if you shared the flow with the user directly.

**NOTE:** This feature is in Beta release.

**NOTE:** This feature must be enabled. For more information, see *Configure Users and Groups*.

- Select the access level privilege from the drop-down on the right side of the textbox.

To save your changes, click **Save**.

Each selected user now can access the flow through their flows page. See *Flows Page*.

**NOTE:** Collaborators have a reduced set of privileges on the flow. For example, they cannot edit the flow name or description or delete it. See *Overview of Sharing*.

# Change Dataset Dialog

Through the Flow View page, you can change the source that is used for your dataset. In this manner, you can apply the same recipe across datasets with the same schema. When the source dataset has been changed, a new sample is automatically generated for you.

For example, you build your recipe for a week's worth of sales data, which is sourced from an imported dataset based on a CSV called, `Week01-Sales.csv`. When the next week's source data is dropped in the appropriate directory, you can:

1. Import the new dataset,
2. Edit the recipe,
3. Change the source to the new file, and
4. Execute a job immediately to process the new week of data.

**NOTE:** A dataset source can be an imported dataset, a reference dataset, or a recipe. Subsequent changes to the source data affect your dataset in development.

## Notes and Limitations:

- If there are differences between the schemas of the source and the new source, your recipe is likely to break on the dataset when the new dataset is selected.
- You can swap your original source dataset with an imported dataset, reference dataset, or a recipe. If needed, you can swap back to the original source at any time.
- If you have enabled relational connections, swapping relational sources may not work if they are from different database vendors.
- Data-dependent transforms, such as `header` and `valuestocols`, use the data that was present in the sample at the time that they were added to the recipe. This fact can cause unexpected changes or breakages when the recipe is applied to another source.
- You cannot undo or redo source swaps.

## Steps:

1. To change a data source, open the flow containing it.
2. In Flow View, you can:
  - a. Click the imported dataset icon. Then, click **Replace**.

**NOTE:** This action removes the imported dataset and all links (edges) coming out of it. The replacement must be reconnected with any downstream objects.

- b. Click the recipe icon. Then, click **Change input**.

**NOTE:** This action substitutes only the primary input from a recipe, which does not include any datasets that are integrated from joins, unions, lookups, or other multi-dataset options.









3. Select the new source:

**NOTE:** You can select data from any flow to which you have access. Changes to the source are inherited.

Change input for POS-r01

Search...

All (17)Imported (16)Reference (0)Recipe (1)

| NAME                                                                                                                                                                            | SOURCE                                                                                                 | LAST UPDATED      |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------|-------------------|
|  POS-r01                                                                                       |  2013 POS (this flow) | Today at 10:18 AM |
|  twitter.json                                                                                  | HDFS                                                                                                   | Today at 12:12 PM |
|  POS-r01.txt                                                                                   | HDFS                                                                                                   | Today at 12:11 PM |
|  POS-schema.csv                                                                                | HDFS                                                                                                   | Today at 10:17 AM |
|   POS-r02.txt | HDFS                                                                                                   | Today at 10:17 AM |
|  POS-r03.txt                                                                                   | HDFS                                                                                                   | Today at 10:17 AM |

Cancel

Change

**Figure: Change Dataset Dialog**

- a. If replacing an imported dataset, you can import new data as the replacement. Click **Import Datasets**. For more information, see *Import Data Page*.
4. Click **Replace** or **Change**.
5. Your dataset is now using the selected dataset as its source, and the current recipe in the Transformer page is applied to the new source.

# Add Schedule Dialog

To add a schedule to your flow, click the drop-down menu in the Flow View page, and select **Schedule**.

**NOTE:** Flow owners can create schedules. Other users must have the appropriate privileges to create schedules for the flow.

**NOTE:** For a dataset with parameters, scheduled times are used when resolving date range parameters.

**NOTE:** Do not schedule executions through Flow View in a Prod instance when Deployment Manager is enabled.

- Schedules defined in Flow View are applied to Active and Non-Active releases in Production environments.
- If the scheduled release is deactivated, the schedule still exists, and the jobs are executed on an flow that is now out-of-date.
- For more information, see *Overview of Deployment Manager*.

Add Schedule

Scheduling Options

Timezone

America/Los\_Angeles

Frequency

Weekly

 on 

Sunday

 at 

12:00

AM

Variables

Set variable values for this schedule's executions

< > regionNum

01

Cancel

Save

Figure: Add Schedule dialog

## Scheduling Options

- **Timezone:** Select the timezone to apply to the schedule.
  - To use UTC time zone, select `UTC` in the drop-down. For more information, see *Supported Time Zone Values*.
- **Frequency:**
  - **Hourly, Daily, Weekly, Monthly:** Run the schedule at the specified moment for the interval.

**Tip:** For drop-downs showing days of the week or days of the month, you can click multiple values to select them.

- **cron:** Set the schedule according to cron syntax.
  - Time zone settings set in the drop-down are used with the cron schedule.
  - For more information, see *cron Schedule Syntax Reference*.

To add another trigger for the flow's schedule, click **Add**.

To create the schedule, click **Save**.

## Variables

If your flow contains one or more variable parameters, you can apply overrides to any variables. When the scheduled job is executed, the variable value is applied to job at runtime.

For each listed variable, you can modify its value.

For more information, see *Overview of Parameterization*.

# Manage Flow Notifications Dialog

When email notifications are enabled, flow owners and collaborators can configure the delivery of emails to interested stakeholders based on the success or failure of jobs executed within this flow. From the flow menu, select **Email notifications**.

**NOTE:** This feature requires access to an SMTP server to send emails. For more information, see *Enable SMTP Email Server Integration*.

## Settings Tab

In the Settings tab, you configure the types of jobs that generate success or failure emails for jobs executed in this flow.

Manage notifications for 2013 POS

Settings

Watchers

Receive job failure emails

Only from scheduled jobs

Receive job success emails

Never

Cancel

Save

Figure: Manage Flow Notifications - Settings tab

These settings apply to jobs executed on the flow. Default settings are inherited from the workspace settings. For more information, see *Workspace Settings Page*.

- **Receive job failure emails:** Select the type of jobs that generate emails when they fail.

| Setting                  | Description                                                                                                         |
|--------------------------|---------------------------------------------------------------------------------------------------------------------|
| From any job             | Emails are generated for any type of job from this flow when it fails.                                              |
| Only from scheduled jobs | Emails are generated when a scheduled job from this flow fails.                                                     |
| Only from manual jobs    | Emails are generated when a manual job from this flow fails. <div>Tip: Jobs executed via API are manual jobs.</div> |
| Never                    | Emails are never generated when jobs from this flow fail.                                                           |



- **Receive job success emails:** Select the type of jobs that generate emails when they succeed. See above for options.

## Watchers Tab

In the Watchers tab, you can add or remove email addresses for interested stakeholders to receive email notifications.

**Tip:** Any flow collaborator can add or remove watchers from this list.

Manage notifications for 2013 POS

Settings

Watchers

Watchers are the people receiving email notifications about this flow activity.

Flow collaborators

Administrator ( )

Watching

☒

Others

Non-collaborators receive notifications but are not able to access the flow or see job details

Enter a new email

Add

joe@example.com

Cancel

Save

**Figure: Manage Flow Notifications - Settings tab**

### Flow collaborators:

By default, the flow owner and all collaborators receive any email notifications for any job executed for this flow.

Click the checkbox next to the name and email address to toggle whether that collaborator receives flow email notifications.

### Others:

For non-collaborators, you can insert email addresses to receive email messages for jobs from the flow. Enter a valid email address and click **Add**.

To remove a non-collaborator, click the Trash icon next to the address.

**Tip:** Email recipients can remove themselves from receiving notifications on flow jobs using a link at the bottom of the email.

To apply your changes, click **Save**.

# Manage Parameters Dialog

## Contents:

- *Parameters Tab*
  - *Overrides Tab*
    - *A note on upgraded parameters*
  - *Manage Parameters for Plans*
- 

Within a flow, you can create and manage flow parameters, including specifying override values. From the flow menu, select **Parameters**.

- A **flow parameter** is a reference token that can be invoked from within the flow. A flow parameter can be:
  - A string value
  - A Trifacta parameter
  - A regular expression
- Where it is invoked, the default value for the parameter is applied or, if an override value has been set, the override value is applied.
- For more information, see *Overview of Parameterization*.

You specify flow parameters at the flow level. They can be invoked in any recipe within the flow.

**Tip:** Override values apply to all parameters in the flow that share the same name, even if they are output object parameters.

**Tip:** Flow parameters can be inherited from upstream flows. For example, if you create a reference dataset that references flow parameters from its flow, those parameters are passed to the downstream flow. While you cannot change the default value for the downstream instance of the parameter, you can apply override values for all recipes in the downstream flow.

## Parameters Tab

In the Parameters tab, you can manage the flow parameters in your flow.

**NOTE:** Non-flow parameters cannot be edited or deleted from the Parameters tab. These parameters must be modified in the interface for the object to which they apply. Overrides can be applied to these types of parameters through the Overrides tab.

When reviewing parameters for your plan, you may notice that a parameter has multiple values:

- Parameter values that do not conflict are ok. For example, if the parameter is applied to an imported dataset and to an output object which are attached to different recipes, these values are not in conflict.
- If the parameter values are in conflict, then a warning icon is displayed. For example, if a parameter is applied to an imported dataset and to an output object attached to the same recipe, then the conflicting parameter values must be overridden to fix them.

**NOTE:** When a parameter has conflicting values from objects in a plan task, you must fix them by applying a single override value. If these values are not fixed, then the plan fails to execute.



Manage parameters

×

Parameters (2)

Overrides (1)

Parameters used in this flow and upstream flows are displayed below. You can also add parameters to use in your recipes.

| Name             | Source                                                                                      | Value  | Add parameter |
|------------------|---------------------------------------------------------------------------------------------|--------|---------------|
| <> colPrimaryKey |  this flow | userId |               |
| <> storeId       |  this flow | 4      |               |

[Learn more about parameters](#)

Close

Figure: Parameters tab

- To create a new parameter, click **Add parameter**.
  - Specify the Name parameter.

**NOTE:** Name values are case-sensitive. After saving a parameter, you cannot change its name.

- Specify the default Value for the parameter. Examples by parameter type:
  - String literal:

This is my string.

**NOTE:** Flow parameter values that are literal values are String values. You can convert them to other data types after they have been referenced in your recipe.

- Pattern : Patterns can be used to find matches in your recipes. The following pattern matches for two consecutive digits:

``{digit}{digit}``

For more information on Patterns , see *Text Matching*.

- Regular expression: These patterns also can be used for finding matches. The following pattern matches for two consecutive digits:

`/[0-9][0-9]/`

Regular expressions are a standard method for identifying patterns in data. The syntax is based on *RE2* and *PCRE* regular expressions.

- Click **Save**.
- To edit a parameter's default value, hover over its entry, and click the Pencil icon.
- To delete a parameter, hover over its entry and click the Trash icon.

**Deleting a parameter cannot be undone. When you delete a parameter, all recipe steps that reference it are broken.**

**Tip:** If you accidentally delete a flow parameter, you can recreate it with the same case-sensitive name. All references to it should work again.

After you create a parameter, you can insert it into your recipes using the following type of reference:

```
${myRecipeParameterName}
```

For more information, see *Create Flow Parameter*.

## Overrides Tab

In the Overrides tab, you can apply override values to any parameters referenced in your flow, including:

- Flow parameters
- output object parameters
- Dataset parameters
- Parameters that are available through reference datasets

**NOTE:** The parameters listed in this tab have override values applied to them. Parameters that are using their default values in this flow are not listed.

Manage parameters

×

Parameters (2) Overrides (1)

Add an override to replace all variable parameters that share the same name.

| Name       | Value | Add override |
|------------|-------|--------------|
| <> storeId | 3     |              |

[Learn more about parameters](#)

Close

**Figure: Overrides tab**

All of the available parameter overrides are listed.

- To apply an override to all references of a parameter within the flow, click **Add override**.
  - Select the name of the parameter.
  - Enter its override value. Click **Save**.

- To edit a parameter override, hover over its entry and click the Pencil icon. Enter a new value.

**NOTE:** The override value is applied to all subsequent operations in the platform. When a job is submitted to the job queue, any overrides are applied at that time. Changes to override values do not affect jobs that are already in flight.

- To delete an override, hover over its entry and click the Trash icon.

### A note on upgraded parameters

If you have upgraded from a version of the product before Release 7.1, any parameters that were defined in the previous version appear grayed out in the Parameters tab.

- The Parameters tab is used for defining new flow parameters, which is a new type of parameter. These parameters can be referenced inside your recipe steps. See above.

**NOTE:** Parameters that are not of flow parameter type cannot be edited in the Parameters tab. You can apply override values to these parameters through the Overrides tab.

- The Overrides tab is used to specify overrides to the default values for parameters, including parameters defined outside of Flow View.
  - If you add an override using the name of the parameter that was upgraded, the override is applied, based on the matching name.

### Manage Parameters for Plans

You can also specify parameter overrides at the plan level. For the recipes in your plan, you can pass in parameter overrides values, which are used during plan runs.

Manage parameters ×

| Parameter  | From                                                | Created in           | Value |                                                                                                                                                                                                         |
|------------|-----------------------------------------------------|----------------------|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| < > region | Run Datasets - WindowsFunctions Flow<br>flowtask-32 | Datasets - Window... | 02    | <div style="display: flex; justify-content: flex-end; gap: 10px;"> <span>Cancel</span> <span style="background-color: #007bff; color: white; padding: 2px 10px; border-radius: 4px;">Save</span> </div> |

Close

**Figure: Manage Parameters Dialog for plans**

In the Manage Parameters dialog, you can review the parameters that are defined for the flows that are part of your plan.

**NOTE:** You cannot create parameters through Plan View. You must create the parameters within the flows that are part of your plan. After they are created, they are available for overrides through Plan View.

**NOTE:** In a plan, parameters and their overrides apply only to the flow task from where they are sourced. They do not apply to other flow tasks. They do not apply back to the source flows.

**Tip:** To pass a flow parameter value from one recipe to another, you can insert the parameter value in a column in a recipe, export the results as a reference dataset, and then ingest that reference into another flow.

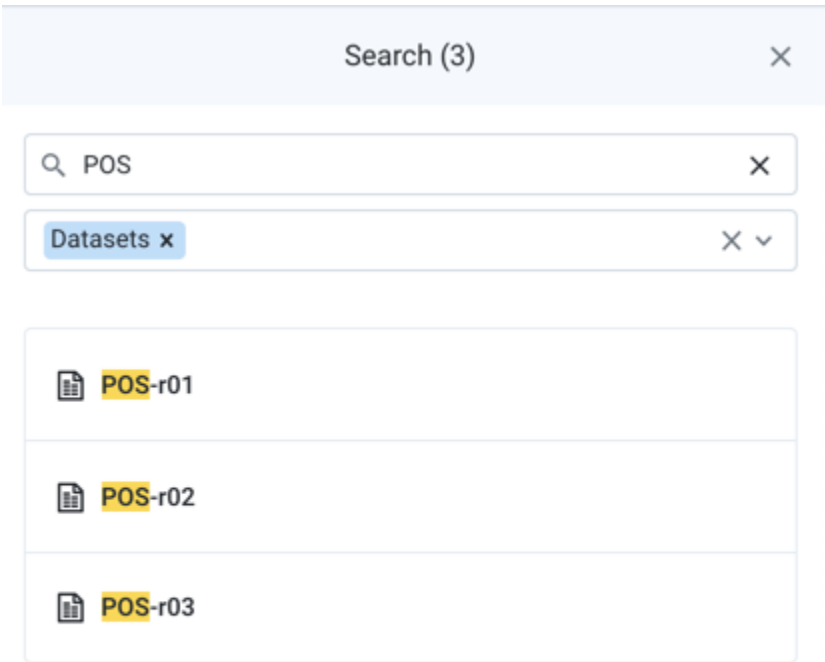
#### Columns:

- **Parameter:** The name of the parameter
- **From:** The source flow in your plan for the parameter
- **Created in:** The object in the source flow where the parameter is defined
- **Value:** The current value for the parameter

**Tip:** To override the inherited value for the parameter, click the Pencil icon. Then, click **Save**. Whenever the flow is executed as part of this current plan, the override value is used to replace the parameter value and any override that are defined within the flow.

# Flow Search Panel

You can search for objects within your flow. In Flow View, click the Magnifying Glass icon at the top of the page.



**Figure: Flow Search panel**

**Steps:**

- 1. From Flow View, click the **Search** icon.
- 2. In the Search panel, enter a search term. As you type your search term, the search results are highlightable in yellow in the panel and in the flow canvas.
- 3. From the **Filter by type** drop-down, you can filter by flow object type:
  - a. Datasets
  - b. Recipes
  - c. Outputs
  - d. References

**NOTE:** If no object type is specified, then all object types are searched.

**Tip:** You may select one or more flow object types to search.

**Context menu**

For each object displayed in the search results, its context menu displays the same options that are available from the object in the canvas, including these additional options:

- **View details:** Open the context panel showing the details of the object.
- **Show in flow:** Center the viewport of Flow View on the object.

The context menu options vary for different types of objects for which you are searching. For more information:

- **Datasets:** See *View for Imported Datasets*.
  - This category includes reference datasets imported from other flows.
- **Recipes:** See *View for Recipes*.
- **Outputs:** See *View for Outputs*.
- **References:** This category covers reference datasets that are created from the recipes in your flow. See *View for Reference Datasets*.



# Flow Optimization Settings Dialog

## Contents:

- *Enable optimization for jobs from this flow*
  - *General Optimizations*
    - *Column pruning optimization*
    - *Filter optimization*
  - *Databases that Support Pushdown*
    - *Column pruning from source*
    - *Filter pushdown*
  - *Other Databases*
    - *Column pruning from source*
- 

In the Flow Optimization Settings dialog, you can configure the following settings, which provide finer-grained control and performance tuning over your flow and its job executions. From the Flow View menu, select **Optimization settings**.

This feature must be enabled at the workspace level. When enabled, the settings in this dialog are applied to the current flow.

- See *Workspace Settings Page*.

These optimizations are designed to improve performance by pre-filtering the volume of data by reducing the columns and rows to the ones that are actually used.

When these filters are enabled, the number of filters successfully applied to a job execution is listed in the Optimization summary in the Job Details page. See *Job Details Page*.

## Enable optimization for jobs from this flow

When enabled, the Designer Cloud application attempts to apply any of the listed optimizations that are enabled to jobs that are executed for this flow.

**NOTE:** When this option is disabled, then no optimization settings are available.

**NOTE:** If two consecutive job executions of a flow fail, then optimizations are skipped for the flow. If the job execution then succeeds, optimizations are automatically disabled for the flow. They can be re-enabled if needed.

## General Optimizations

The following optimizations can be enabled or disabled in general. For individual data sources, you may be able to enable or disable these settings based on your environment and its requirements .

**Tip:** These optimizations are applied at the recipe level. They can be applied on any flow and may improve performance within the Transformer page.

### Column pruning optimization

When enabled, job execution performance is improved by removing any unused or redundant columns based on the recipe that is selected.

### Filter optimization

When this setting is enabled, the Designer Cloud application optimizes job performance on this flow by pushing data filters to recipes.

## Databases that Support Pushdown

Individual types of databases may support one or more of the following pushdowns. Additional restrictions may apply for your specific database.

**Tip:** These optimizations are applied to queries of your relational datasources that support pushdown. These optimizations are applied within the source, which limits the volume of data that is transferred during job execution.

**NOTE:** For each relational connection, you can enable the optimization capabilities to improve the flow and its job execution performance. The optimization settings may vary based on the type of relational connections.

### Column pruning from source

When enabled, job execution performance is improved by removing any unused or redundant columns from the source database.

#### Limitations:

- Column pruning optimizations cannot be applied to imported datasets generated with custom SQL.

### Filter pushdown

When this setting is enabled, the Designer Cloud application optimizes job performance on this flow by pushing data filters directly on the source database.

#### Limitations:

- Filter pushdown optimizations cannot be applied to imported datasets generated with custom SQL.
- Pushdown filters cannot be applied to dates in your relational sources.

**NOTE:** SQL-based filtering is performed on a best-effort basis. When these optimizations are enabled for your flow, there is no guarantee that they will be applied during job execution.

**NOTE:** The connection types may or may not be available in your product edition. For more information, see *Connection Types*.

## Other Databases

Databases that do not support pushdown may support the following optimization settings.

### **Column pruning from source**

When enabled, job execution performance is improved by removing any unused or redundant columns from the source database.

# Plans Page

The Plans page lets you create, review, and manage your plans. A **plan** is a sequence of tasks and the triggers that execute them. Plans can be applied across multiple flows in your workspace.

**NOTE:** Access to the Plans page in the application and privileges on plans is governed by roles in your workspace. For more information, please contact your workspace administrator.

- To create a new plan, click **Create....** To rename the new plan, click the `Untitled` value at the top of the page.
  - The maximum number of plan tasks may vary depending on the plan that you have licensed. For more information, please contact your Trifacta representative.
- Workspace admins can access all plans in the workspace.

| Name                         | Owner | Last updated     | Last run                                               |
|------------------------------|-------|------------------|--------------------------------------------------------|
| <a href="#">Random_Data</a>  |       | Today at 3:22 PM | <a href="#">Today at 3:22 PM</a> <a href="#">Share</a> |
| <a href="#">customerdata</a> |       | Today at 3:19 PM | <a href="#">Today at 3:20 PM</a>                       |
| <a href="#">test2</a>        |       | Today at 3:18 PM | <a href="#">Today at 3:19 PM</a>                       |

**Figure: Plans Page**

## Columns:

- **Name:** The name of the plan.
  - Click the plan name to review it. See *Plan View Page*.
- **Owner:** Owner of the plan.

**NOTE:** Click the link to see the users with whom the plan has been shared. A value of 2 `Users` or more indicates that the plan has been shared. An empty column indicates that you are the owner of the plan and have not shared it with anyone else. See *Share a Plan*.

- **Last Updated:** Timestamp for the last time that the flow was modified.
- **Last Run:** Timestamp for when the plan was last executed.
  - The displayed icon indicates whether the plan executed successfully or not.
  - Click the link to review details of the run. See *Plan View Page*.

## Actions:

- **Create:** From the Create menu, choose to create a plan.

- Enter a name and description for your plan. Click **Create**.
- **Import:** From the context menu, select **Import** to import a plan into this instance. See *Import Plan*.
- **Search:** To search plan names, enter a string in the search bar. Results are highlighted immediately in the Plans page.
- **Sort:** Some column headers can be selected to sort the display by the column's entries.

#### Plan options:

The following options are available on the right side of a plan's entry:

- **Share:** Share the plan with other users. See *Share a Plan*.
- **Rename:** Change the name and description of the plan.
- **Email notifications:** Send email notifications on the plan runs. See *Manage Plan Notifications Dialog*.
- **Export:** Export the plan from Designer Cloud powered by Trifacta Enterprise Edition. See *Export Plan*.
- **Delete Plan:** Delete the plan.

**Deleting a plan removes all objects contained in the plan. Flows referenced in the tasks of the plan are not touched.**

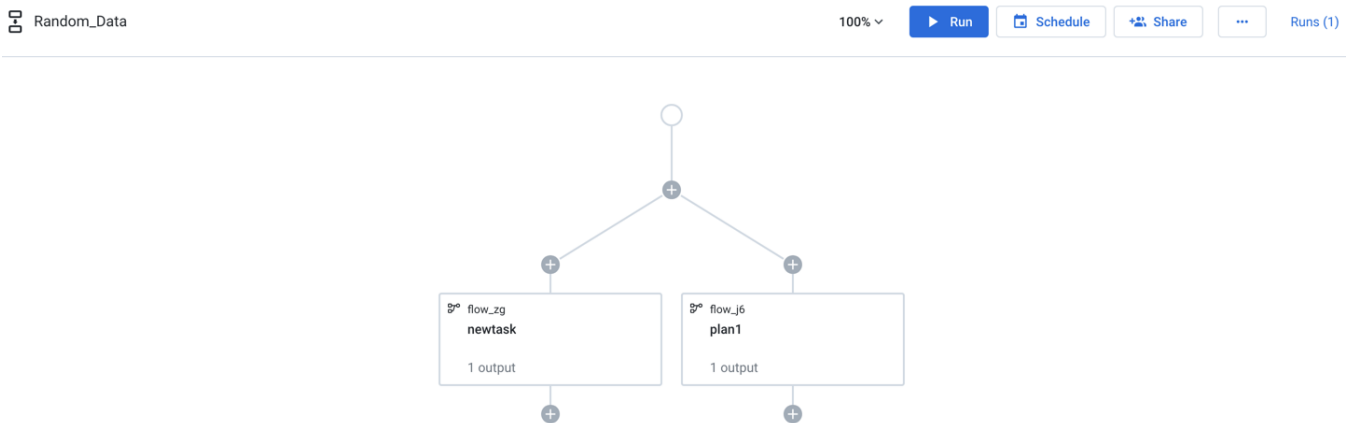
# Plan View Page

**Contents:**

- *Top Bar*
  - *Plan context menu*
- *Workflow*
- *View for Triggers*
- *View for Tasks*
  - *Task types*
- *Run Details*
  - *Task Execution Rule*
  - *Parallel Tasks*
  - *Cancel Plan Run*

In Plan View, you design your plan, which includes the building and sequencing of tasks and the triggers that execute your sequence of tasks.

**NOTE:** Access to this page in the application and privileges on its related objects is governed by roles in your workspace. For more information, please contact your workspace administrator.



**Figure: Plan View Page**

The main panel is the plan canvas, where you build your plans.

On the right side is the context panel. Depending on what you select in the plan canvas, a different set of context options is displayed.

## Top Bar

**Tip:** To rename the plan, click the plan name in the top bar.

## Zoom Options:

You can zoom the plan canvas to display areas of interest in the plan graph.

The zoom control options are available at the right side of the canvas. The following are the available zoom options:

**Tip:** You can use the keyboard shortcuts listed in the zoom options menu to make quick adjustments to the zoom level.

- **Zoom in:** Zoom in 10% on the canvas to focus on greater detail.
- **Zoom out:** Zoom out 10% from the canvas to see more of it.
- **Zoom to fit:** Change the zoom level to fit all of the objects of your plan onto the screen.
- **25%, 50%, or 100%:** Change the zoom level to one of the preset levels.

**NOTE:** By default, the plan view page always opens in Zoom to fit option and it does not remember the previous zoom and position.

#### Other options:

- **Run:** Run the plan. You can track progress of your plan run.

**Tip:** You can apply overrides to flow parameters through the Parameters tab. See *Plan View for Flow Tasks*.

See *Plan Runs Page*.

- **Schedule:** Create or edit the plan schedule with one or more triggers through the right context panel. See "View for Tasks."
- **Share:** Share the plan with other users. See *Share a Plan*.
- **Runs:** The Runs link tracks the current total number of runs that have been queued or executed for this plan. Click this link to track progress on your plan run. See "Run Details" below.

#### Plan context menu

- **Rename:** Modify the name and description for your plan.
- **Parameters:** You can apply overrides to the recipe parameters for your plan tasks during plan job runs. See *Manage Parameters Dialog*.
- **Email notifications:** Send email notifications on the plan runs. See *Manage Plan Notifications Dialog*.
- **Export:** Export the plan from Designer Cloud powered by Trifacta Enterprise Edition. See *Export Plan*.
- **Delete:** Delete your plan.

#### Workflow

For more information, see *Create a Plan*.

#### View for Triggers

To create a trigger, you specify the time and frequency that the plan is to be executed.

**NOTE:** Plan owners can create triggers for their plans. Other users must have the appropriate privileges to create triggers for the plan.

<
Trigger
×

Schedule
Enable
☒

Timezone

America/Los\_Angeles

Frequency
Add another trigger

Weekly

On

Sunday, Wednesday

×

At

12:10

AM

Monthly

On the

1

day(s) of the month

×

at

12:00

AM

Save

**Figure: Create plan trigger**

- **Timezone:** Select the timezone for the time that you are specifying in the trigger's schedule.
  - To use UTC time zone, select `UTC` in the drop-down.
  - For a list of supported timezones, see *Supported Time Zone Values*.
- **Frequency:** Specify how frequently the plan is triggered:
  - **On:** You can specify multiple entries to trigger the plan more frequently.
  - **cron:** Set the schedule according to cron syntax.
    - Time zone settings set in the drop-down are used with the cron schedule.
    - For more information, see *cron Schedule Syntax Reference*.
- To add another trigger, click **Add another trigger**.
- To save your changes, click **Save**.



Trigger

Schedule

Enable ☒ Edit

Weekly: 12:10 AM  
On Sunday and Wednesday

Monthly: 12:00 AM  
On the 1st

**Figure: Saved trigger**

- To disable the trigger, click the slider.

**NOTE:** If you disable a trigger, no new scheduled executions of the tasks in the plan occur. You can still manually trigger plan runs.

- To make changes to the trigger, click **Edit**.

## View for Tasks

When you create or select a task, you can modify its settings through the context panel on the right.

**Tip:** To rename the task, click the task name in the context panel.

### Task context menu options:

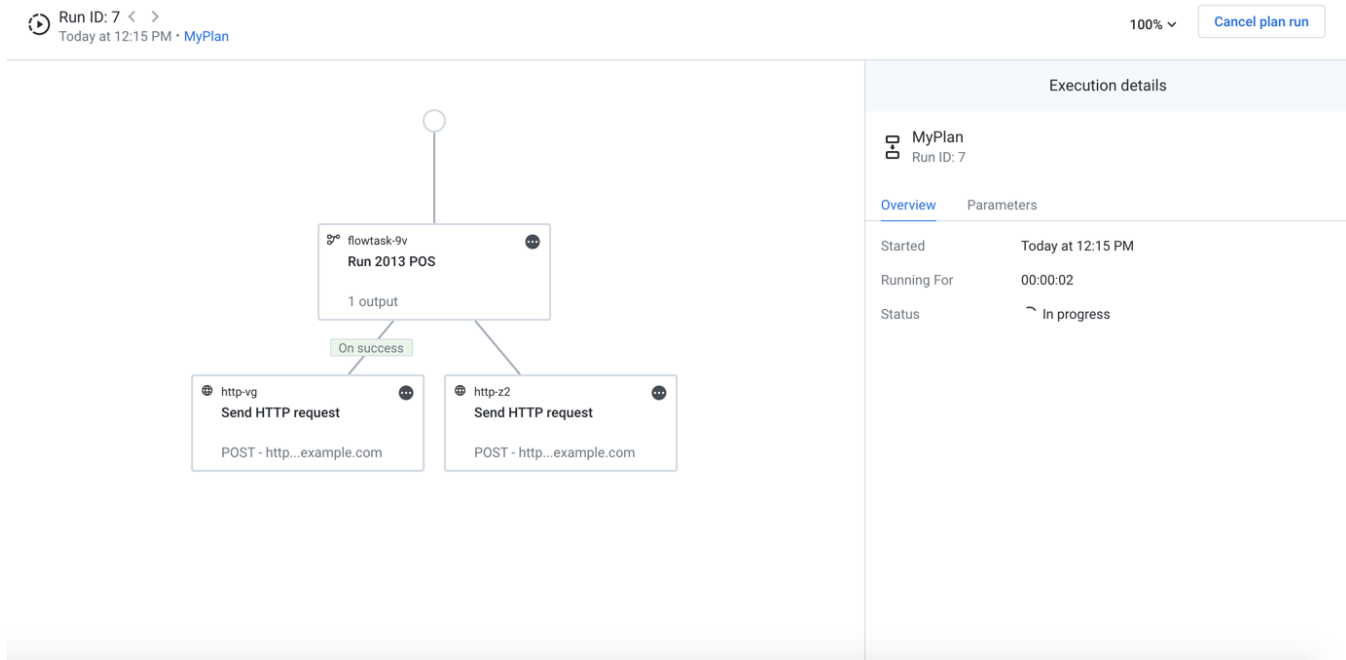
- **View Flow:** Open the flow. See *Flow View Page*.
- **Edit name:** Modify the name of the task.
- **Delete:** Delete the task from your plan.

## Task types

- **Flow task:** Generate all of the defined output objects for a flow. See *Plan View for Flow Tasks*.
- **HTTP task:** Execute a task over HTTP protocol. See *Plan View for HTTP Tasks*.
- **Slack task:** Send a message from the Designer Cloud application to a specified Slack channel. See *Plan View for Slack Tasks*.

## Run Details

You can review the details of individual executions of your plan. Click the **Runs** link in the upper-right corner.



**Figure: Run details**

The latest plan run is displayed. You can review the progress of individual tasks throughout the plan run.

**NOTE:** When a plan run begins, a snapshot is taken of the plan. Subsequent changes to the underlying flows could impact the outcome of the flow tasks when they are later executed during the plan run.

- You can select individual triggers and tasks to review details of the plan run for that object in the context panel.
- To see other runs for the plan, use the angle brackets next to the timestamp at the top of the screen.

You can track all of the runs across all of your plans. See *Plan Runs Page*.

## Task Execution Rule

You can gate the execution of a task based on the completion status of its previous task. Click the line connecting the two tasks and select one of the following options:

- **On success:** Runs if the previous task in the node is successful.
- **On failure:** Runs if the previous task in the node failed.
- **On execution (any status):** Runs the task in the node irrespective of the previous task's status (success or failure).

For more information, see *Create a Plan*.







## Parallel Tasks

You can execute one or more tasks in parallel by clicking the plus node icon and selecting one of the following options:

- **Add a parallel node:** Adds a parallel node to the existing node.
- **Add a node:** Adds a node at the bottom of the existing node.

**NOTE:** Nodes added in a sequence are separated by a plus node. A node added in parallel has two plus nodes separating it from the parent node; one for adding nodes in parallel and other for adding a node in sequence.

The following icons indicate the results of the execution of a task:

| Icon                                                                              | Task Status                                                  |
|-----------------------------------------------------------------------------------|--------------------------------------------------------------|
|  | Task successful                                              |
|  | Task failed                                                  |
|  | Task skipped. Task was not executed due to unmet conditions. |
|  | Task canceled by user                                        |
|  | Task in progress                                             |
|  | Task pending                                                 |

For more information, see *Create a Plan*.

### Cancel Plan Run

To cancel a plan that is currently running, please do the following:

1. In Plan View, click the **Runs (x)** link.
2. In the Run Details page, click **Cancel plan run**.

# Share Plan Dialog

You can manage access to a plan for other users through the Share Plan dialog. From the context menu of the Plans page, select **Share**.

**Tip:** If groups have been enabled in your instance of the Designer Cloud powered by Trifacta platform , you can share flows and connections to LDAP groups. For more information, see *Configure Users and Groups*.

**Tip:** A workspace administrator has owner-level access to plans in the workspace. For more information, see *Workspace Admin Permissions*.

## Permissions

You can grant permissions to other users to access the plan. When a user is given access to a plan, that user is considered a **collaborator** on the plan and has a smaller set of permissions than the **owner** of the plan. Users must have at least the Viewer workspace role viewer permissions for plans.

For more information, see *Workspace Roles Page*.

**NOTE:** If the user does not have access to the underlying flows of the plan, the plan can still be shared and accessed, but the user cannot edit flows and run the plans.

## Actions

Through this Share Plan dialog, you can invite one or more collaborators to the plan, so that you may work together on the same objects.

- **Add:** Add users or email addresses of users with whom you would like to share the plan.

To add users as collaborators in your flow, start typing the name of a user with whom you would like to share the plan. Select the user. Repeat this process to add multiple users.

**NOTE:** For privacy and security reasons, search options are not available.

- **Save:** Click **Save**.
- **Delete:** Select the user or email address in the search by name or email field and click delete using your keyboard.
- **Cancel:** To cancel sharing, click **Cancel**.

Each selected user now can access the plans through the plans page. See *Plans Page*.

# Manage Plan Notifications Dialog

This section provides an overview of sending email notifications to plan owners and collaborators based on the results of plan runs.

When email notifications are enabled, plan owners and collaborators can specify the list of email recipients, based on the status of execution of plans. From the context menu of the Plans page, select **Email notifications**.

- Users who receive notifications for specific plans are considered plan watchers.
- By default, all collaborators receive notifications about plan run failures.
- If plan collaborators have only view permissions, they may not be able to edit the recipients.
- You cannot enable or disable email notifications at the plan level. Workspace administrators can enable or disable email notifications for plans.

**NOTE:** This feature requires access to an SMTP server to send emails. For more information, see *Enable SMTP Email Server Integration*.

**Tip:** Email recipients can remove themselves from receiving notifications on plan runs using a link at the bottom of the email.

Email notifications

Send email notifications when this plan runs

Add recipients

Add

| Recipient                                                                                 | Notify                                        |
|-------------------------------------------------------------------------------------------|-----------------------------------------------|
| <div><div>TU</div><div>Test User 4147080222</div><div>4147080222@trifacta.com</div></div> | <div>On failure</div> <div></div> <div></div> |
| <div><div>A</div><div>Administrator</div><div>admin@trifacta.local</div></div>            | <div>On failure</div> <div></div> <div></div> |

Done

Figure: Manage Plan Notifications

**NOTE:** You can enable or disable the **Send email notifications when this plan runs** option to enable or disable plan email notifications. If this option is disabled, the below options are not available.

For more information, see *Email Notifications Page*.

In the Email notifications dialog, plan owner and collaborators can add the email addresses to stakeholders to receive notifications based on the plan run status:

- **On success:** Emails are generated if the plan run succeeded.
- **On failure:** Emails are generated if the plan run failed.
- **On execution (any status):** Emails are generated whether the plan run succeeded or failure.

**Actions:**

- **Add:** Add email addresses in the **Add recipients** field.

**Tip:** Users can send plan run emails to any valid email address or email alias, even if the user(s) do not have an account in Designer Cloud powered by Trifacta Enterprise Edition.

- **Delete:** To remove a user, click the Trash icon next to the email address.
- **Save:** Click **Done**.

# Plan View for Flow Tasks

## Contents:

- *Sources tab*
  - *Outputs tab*
  - *Parameters tab*
- 

When you create a flow task in Plan View, you first search for the flow that you wish to run. Then, you specify the task to execute on the flow in the right context panel.

When a task is executed, the recipes in the flow that have been selected in the task are executed, and if successful, their selected outputs are generated.

**NOTE:** In a flow, all recipes that you wish to have executed by the corresponding task must have a defined output object. For each output object, you must create at least one write settings or publication object. During plan runs, these objects are not validated, and missing outputs are ignored.

Run flow

X

Q Search flows...

2013 POS

POS-r01 – 2.txt

Run-Job-Publishing-2020-04-16T20:28:40.141Z

allTypes

Run-Job-Publishing-2020-04-16T20:02:41.587Z

allTypes

Untitled Flow

No outputs

flowy

job – 2.log

Test-Notification

carriers\_recipe, Flights\_recipe

Test-Notification

carriers\_recipe, Flights\_recipe


**Figure: Flow task - select flow**

- Enter a search string in the search box.
- When you locate the flow to execute, select it.

Select the outputs that you wish to have generated for the flow.



< Run flow ×

 Run-Job-Publishing-2020-04-16T20:02:41.587Z

Choose outputs to run

☒ allTypes

Cancel Create task

**Figure: Flow task - select outputs**

**Webhooks:** If webhooks are configured for the underlying flow, you can optionally disable execution of the webhooks that are defined in the flow when the flow task in the plan is executed. Click the icon to enable or disable webhook execution.

To save the task, click **Create task**.

The saved task is displayed in the context panel.


**Tip:** To rename the task, click the task name.

Sources tab

Review the datasources for the task, including a full path to the source location.

Run flow

×


 Run Run-Job-Publishing-2020-04-16T20:02:41.587Z

Flow: Run-Job-Publishing-2020-04-16T20:02:41.587Z

...

Sources

Outputs

 allTypes.csv

hdfs://

trifacta/uploads/7/efc8632e-0400-4382-ba3f-88d4d2c6ac8a/allTypes.csv


Figure: Flow task - Sources tab

Outputs tab

Review the outputs that you have selected for the task to generate.

Run flow

×




Run Run-Job-Publishing-2020-04-16T20:02:41.587Z

Flow: Run-Job-Publishing-2020-04-16T20:02:41.587Z

...

Sources

Outputs



allTypes

Profile results

1 publishing action - [Show](#)

[Add/remove outputs](#)

**Figure: Flow task - Outputs tab**

- To review the publications configured for the output, click **Show**.
- To change the outputs generated by the task, click **Add/remove outputs**.


## Parameters tab

If the flow from which the task was created contains parameters, you can review those parameters and apply overrides as necessary.

**NOTE:** Parameter overrides applied to a plan affect only plan execution. These overrides do not apply to any independent job executions of the underlying flows.

Flow task

×



Run 2013 POS

flowtask-9v • [2013 POS](#)

...

Sources

Outputs

Parameters

| Parameter  | Value | <a href="#">Manage</a> |
|------------|-------|------------------------|
| < > region | 02    |                        |

**Figure: Flow task - Parameters tab**

**To apply overrides:**

1. Click **Manage**.
2. In the Manage Parameters dialog, mouse over the Value entry for the desired parameter. Click the Edit icon.

**Tip:** You can use the Search box to locate parameters by name.

3. Enter a new value and click **Save**.

Whenever the flow task is executed, this override value is applied to the execution of the flow task.


# Plan View for HTTP Tasks

In Plan View, you can create HTTP tasks to send request to endpoints before or after the execution of other tasks. These tasks are specified in the right context panel.

<

HTTP task

×

 Send HTTP Request

Test

...

Method

GET


▼

URL

required

https://example.com/v4/connections

https://example.com/endpoint

Headers 

Add

Authorization

<my\_authorization\_key>

Remove

Key

Value

Remove

Secret Key

Secret Key

☒ Validate SSL certificate

Retry

3

times

Cancel

Save

Figure: HTTP task

**Tip:** To rename the task, click the task name.

## Fields:

| Field                    | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Method                   | Select the HTTP method to use to deliver the message. The appropriate method depends on the receiving application. Most use cases require the POST method.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| URL                      | URL where the HTTP request is received by the other application.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| Headers                  | <p>Insert HTTP content headers as key-value pairs. For example, if your body is in JSON format, you should include the following header:</p> <pre>key: Content-Type value: application/json</pre> <p><b>NOTE:</b> You may be required to submit an authentication token as the value for the <code>Authorization</code> key.</p>                                                                                                                                                                                                                                                                                                                 |
| Body                     | <p>( POST , PUT , or PATCH methods only) The body of the request submitted to the receiving application. Request body is structured as follows:</p> <pre>{"text": "My text message to the receiving application."}</pre> <p><b>Tip:</b> As part of the request body or header fields, you can insert metadata references to the plan definition, current run, tasks already executed in the run, and underlying flows, including column data and datasources. For more information on the available metadata, see <i>Plan Metadata References</i>.</p> <p>For examples of requests including metadata examples, see <i>Create HTTP Task</i>.</p> |
| Secret Key               | <p>(Optional) A secret key can be used to verify the request payload. This secret value must be inserted in this location, and it must be included as part of the code used to process the requests in the receiving application. Insert the secret value here as a string without quotes.</p> <p>For more information on how this secret key is used to generate a signature, See <i>Create HTTP Task</i> .</p>                                                                                                                                                                                                                                 |
| Validate SSL Certificate | <p>When set to <code>true</code>, HTTPS (SSL) communications are verified to be using a valid certificate before transmission.</p> <p><b>NOTE:</b> If you must send a request to an endpoint that has an expired/invalid certificate, you must disable SSL verification.</p>                                                                                                                                                                                                                                                                                                                                                                     |
| Retry                    | <p>If the returned status code is outside of the 200-299 range, then the HTTP task is considered to have failed. When this option is enabled, the request is retried.</p> <p>If the request fails, this value defined the number of times that the request should be retried. If this number of retries is reached without success, the task fails.</p>                                                                                                                                                                                                                                                                                          |

## Actions:

- To test if the specified endpoint is reachable, click **Test**.

**Tip:** A status code of 200 indicates that the test was successful.

**Tip:** You can use the GET method for testing purposes. A GET request does not change any data on the target platform but may permit you to specify elements in the request body.

- **Edit task name:** Change the name of the task.

**Tip:** Good naming may include the target platform endpoint and method, as well as the purposes of the task in your plan.

- **Delete:** Delete the task.

**This step cannot be undone.**

For more information, see *Create HTTP Task*.

# Plan View for Slack Tasks

In Plan View, you can create tasks to send messages to a Slack channel. These tasks are specified in the right context panel.

**Tip:** Slack tasks are a specialized form of HTTP tasks.

## Requirements

The following requirements apply to the Slack app that receives the message. For more information on Slack apps, see <https://api.slack.com/apps>.

Please verify that your Slack app has the following:

- Create an OAuth Token that has `chat:write` scopes. This token is inserted into your task definition. There are two types of tokens:

- **Bot Token:** These tokens post a Slack message from the name of the app.

**NOTE:** The Bot Token also requires the `chat:write.public` scope.

**Tip:** A Bot Token is required if you wish to send a direct message through the App category of messages.

- **User Token:** These tokens post a Slack message from the user who authorizes the message.

**Tip:** A User Token is required if you are sending the message to a private channel or to another user (see below).

- The OAuth Token that you create must be installed in your workspace.

**NOTE:** Copy the generated token to a text file and retain it for later. This token must be pasted into the definition of each Slack task where you wish to use it.




Create Slack Task

<

Slack task

×

 Post a message

Test

...

Request

Response

To integrate with Slack, you need a Slack app. See your apps or create a new one [here](#).

OAuth Token

required

xoxb

You will find it in [your app's](#) OAuth & Permissions section

Channel

required

testing

Paste the name of the channel, exactly as you can see it on Slack

Message

Plan {{plan.name}} is running.  
Start time: {{plan.startTime}}

Add metadata to your message by simply pressing \$

Cancel

Save

Figure: Slack task

**Tip:** To rename the task, click the task name.

Fields:

| Field       | Description                                                                                                                                                                                                                                                                                         |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| OAuth Token | The OAuth token to use for posting the message.                                                                                                                                                                                                                                                     |
| Channel     | Paste one of the following values from the Slack workspace for where to post the message: <ul style="list-style-type: none"><li><b>Channel Name:</b> Name of the channel as it appears in Slack.</li><li><b>Channel ID:</b> This value is available in the Settings page for the channel.</li></ul> |

Copyright © 2022 Trifacta Inc.

Page #650

|         |                                                                                                                                                                                                                         |
|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|         | <ul style="list-style-type: none"> <li>• <b>Member ID:</b> You can post the message to a specific user instead of posting to a channel. A user's member ID can be found in the user's Profile page in Slack.</li> </ul> |
| Message | <p>The message to post.</p> <div> <p><b>Tip:</b> Messages can include metadata information about the tasks in the current plan run. For more information, see <i>Plan Metadata References</i>.</p> </div>               |

#### Actions:

- To test if the specified endpoint is reachable, click **Test**.

**Tip:** A status code of 200 indicates that the test was successful.

- **Edit task name:** Change the name of the task.
- **Delete:** Delete the task.

**This step cannot be undone.**

For more information, see *Create Slack Task*.



## Filter by ownership:

For the selected object type, you can filter based on the ownership of the object:

- **All:** All objects of the selected type to which you have access.
- **Owned by me:** All objects of the selected type that you own.
- **Shared with me:** All objects of the type that have been shared with you.

## Columns:

- **Name:** Name of the object.
- **In Flows:** Count of flows in which the object is in use.
- **Source:** Flow or datastore where the object is located.
- **Last Updated:** Timestamp of the last time that the object was modified.

## Actions:

- **Browse:** If displayed, use the page browsing controls to explore the available objects.
- **Search:** To search object names, enter a string in the search bar. Results are highlighted immediately in the Library page.
- **Sort:** Click a column header to sort the display by the column's entries.

## Object Actions:

Hover over an object to reveal these actions on the right side of the screen.

- **Details:** Review details about the dataset. See *Dataset Details Page*.
- **Preview:** Inspect a preview of the dataset.

**NOTE:** Preview is not available for binary format sources.

- **Wrangle in New Flow:** (Imported dataset only) You can create a new flow and begin immediately wrangling the dataset. This step also creates a recipe in the flow.
- **Add to Flow:** Add the dataset to a new or existing flow.
- **Make a copy:** Create a copy of the imported dataset. This option is not available for reference datasets.
- **Edit name and description:** Change the name and description of the dataset.
- **Edit data settings:** If the source of the imported dataset required conversion to an internally supported format, you can modify settings related to that conversion process. For more information, see *File Import Settings*.

**Tip:** This setting applies primarily to binary file formats, such as PDF and Excel, or file formats that may require additional steps to convert into tabular data, such as JSON.

- **Delete Dataset:** Delete the dataset.

**Deleting a dataset cannot be undone.**

# Dataset Details Page

Contents:

- *Imported Dataset*
- *Reference Dataset*
- *Dataset with Parameters*

Use the Dataset Details page to review a dataset's usage and to perform management tasks on it.


## Imported Dataset

For datasets that have been imported into Designer Cloud powered by Trifacta® Enterprise Edition, you can review source location and current usage within flows to which you have access.

**Status:** For large relational datasets, you can track status of the import process. For more information, see *Overview of Job Monitoring*.

**NOTE:** This feature may require enablement in your deployment. See *Configure JDBC Ingestion*.

If the dataset is used in a flow, click the flow name to review its usage in the flow. See *Flow View Page*.

 POS-r01.txt

Wrangle in new Flow

Preview

...

Last updated: Today at 11:28 AM

Created: Today at 11:28 AM

File size: 285.95kB

Size: 16 columns - 4 types

Location:

/POS-r01.txt

Used in 1 Flow


| Name                                                                                         | Objects               | Last Updated      |
|----------------------------------------------------------------------------------------------|-----------------------|-------------------|
|  2013 POS | 6 Datasets, 0 Recipes | Today at 11:28 AM |

Figure: Imported dataset details

Actions:

- **Wrangle in New Flow:** Create a new flow for your dataset and begin wrangling.
- **Preview:** Review the first few rows of the dataset.
- **Edit custom SQL:** After you have created a dataset using custom SQL, you can modify the SQL used to construct the imported dataset.
- **Add to Flow:** Add imported dataset to a new or existing flow.
- **Make a copy:** Create a copy of the imported dataset.
- **Edit name and description:** Edit the name and description for the dataset.
- **Remove structure:** Remove initial steps applied to structure data.
- **Delete Dataset:** Delete the dataset.

Deleting a dataset cannot be undone.

If the dataset was imported with a customized SQL statement, select **Edit custom SQL**. Modify the SQL statement(s) as needed.

Through the custom SQL interface, it is possible to enter SQL statements that can delete data, change table schemas, or otherwise corrupt the targeted database. Please use this feature with caution.

**NOTE:** If you modify the SQL statement for your imported dataset, any samples based on the old SQL statement are invalidated.

For more information, see *Create Dataset with SQL*.  
For more information on the sources from which a dataset was created, see *Flow View Page*.

Reference Dataset

A **reference dataset** is a reference from one flow to the dataset that is sourced in another flow. When the source dataset is modified, the reference dataset automatically receives the changes. For more information on creating a reference dataset, see *Flow View Page*.

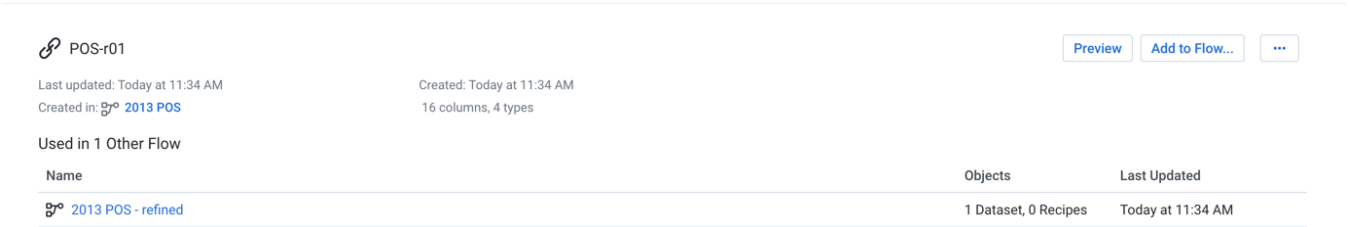


Figure: Reference Dataset details


Actions:

- **Preview:** Review a preview of the first few rows in the dataset.
- **Add to Flow:** Add the reference dataset to a new or existing flow.
- **Edit name and description:** Edit the name and description for the dataset.
- **Delete Reference Dataset:** Delete the reference dataset. The object on which the reference dataset is based is untouched.

Deleting a dataset cannot be undone.

Dataset with Parameters

If your dataset was created with parameters, you can review dataset and parameter information in the details.  
For more information on creating these datasets, see *Create Dataset with Parameters*.  
For more information, see *Overview of Parameterization*.

 Dataset with Parameters

Wrangle in new Flow

Preview

...

Last updated: Today at 11:33 AM

Created: Today at 11:33 AM

Size: 16 columns · 4 types

Parameters

Path: eed8f72c-edcc-40f6-b067-797841e1cc1c/POS-r ★ {digit}{digit}.txt

| Parameters       | Type    | Default value                                      | Name |
|------------------|---------|----------------------------------------------------|------|
| ★ {digit}{digit} | Pattern | matches against trifacta pattern: '{digit}{digit}' |      |

Figure: Dataset with Parameters details

You can review the parameters and variables that have been defined for the dataset.

Action:

- **Wrangle in new Flow:** Create a new flow for your dataset and begin wrangling.
- **Preview:** Review the first few rows of the dataset.
- **Add to Flow:** Add imported dataset to a new or existing flow.
- **Make a copy:** Create a copy of the imported dataset.
- **Edit name and description:** Edit the name and description for the dataset.
- **Edit parameters:**Modify the parameters used to create the dataset. See *Create Dataset with Parameters*.
- **Remove structure:** Remove the initial parsing structure. When the structure is removed:
  - The dataset is converted to an unstructured dataset. An **unstructured dataset** is the source data converted into a flat file format.
  - All steps to shape the dataset are removed. You must break up columns in manual steps in any recipe created from the object. See *Flow View Page*.
- **Delete Dataset:** Delete the dataset.

Deleting a dataset cannot be undone.

# Import Data Page

Contents:

- General Limitations
- Basic Workflow
  - 1. Connect to sources
  - 2. Add datasets
  - 3. Configure selections
  - 4. Import selections

Through the Import Data page, you can upload datasets or select datasets from sources that are stored on connected datastores. From the Library page, click **Import Data**.

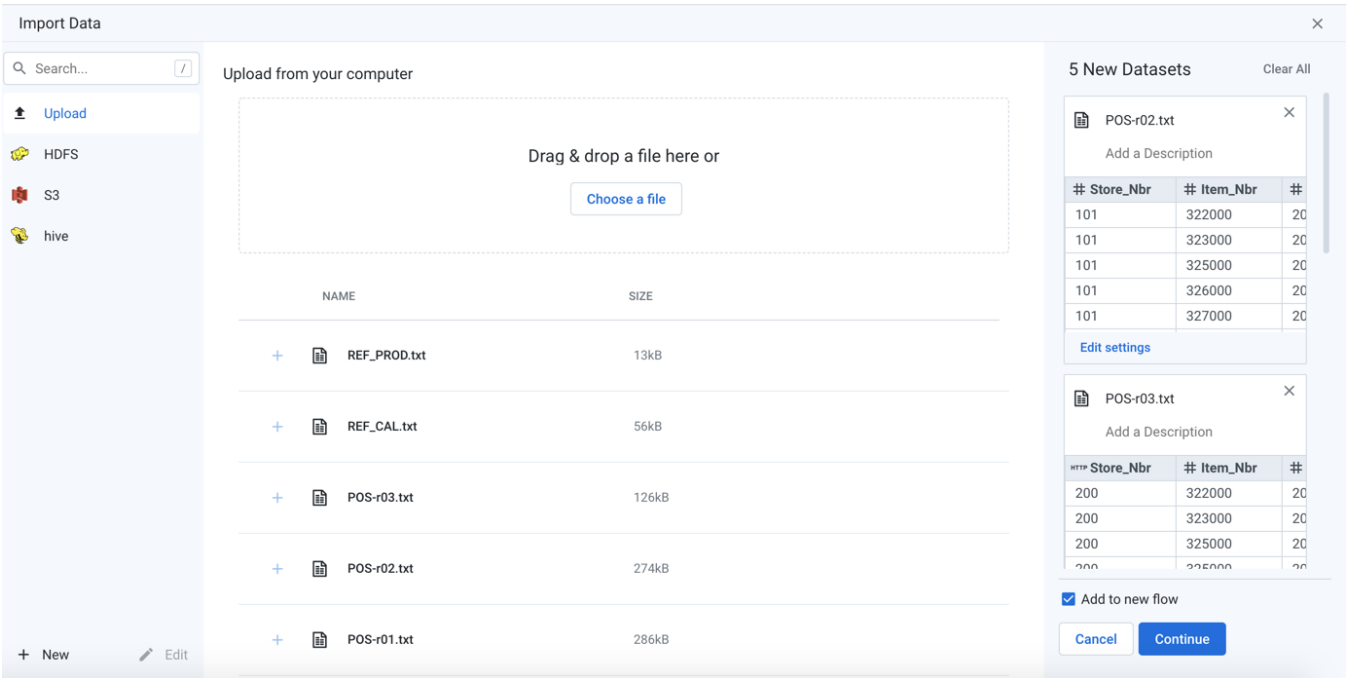


Figure: Import Data page

## General Limitations

**NOTE:** For file-based sources, Designer Cloud powered by Trifacta® Enterprise Edition expects that each row of data in the import file is terminated with a consistent newline character, including the last one in the file.

- For single files lacking this final newline character, the final record may be dropped.
- For multi-file imports lacking a newline in the final record of a file, this final record may be merged with the first one in the next file and then dropped in the Trifacta Photon running environment.



**NOTE:** To be able to import datasets from the base storage layer, your user account must include the `dataAdmin` role.

**NOTE:** An imported dataset requires about 15 rows to properly infer column data types and the row, if any, to use for column headers.

#### File and path limitations:

- The colon character ( `:` ) cannot appear in a filename or a file path.
- Filenames cannot begin with special characters like dot ( `.` ) or underscore ( `_` ).

### Basic Workflow

#### 1. Connect to sources

During import, the Designer Cloud application identifies file formats based on the extension of the filename.

- Compressed files are recognized and can be imported based on their file extensions.
- Filenames that do not have an extension are treated as TXT files.

**Upload:** Designer Cloud powered by Trifacta® Enterprise Edition can also load files from your local file system.

**Tip:** You can drag and drop files from your desktop to to upload them.

**NOTE:** You can upload a file up to 1 GB in size.

**NOTE:** When you upload an updated version of a previously uploaded file, the new file is stored as a separate upload altogether. In your flow, you must swap out the old dataset to point to the new one.

**HDFS:** If connected to a Hadoop cluster, you can select file(s) or folders to import. See *HDFS Browser*.

**Hive:** If connected to a Hive instance, you can load datasets from individual tables within the set of Hive databases. See *Hive Connections*.

**S3:** If connected to an S3 instance, you can browse your S3 buckets to select source files.

**Tip:** For HDFS and S3, you can select folders, which selects each file within the directory as a separate dataset.

See *External S3 Connections*.

**Redshift:** If connected to an S3 data warehouse, you can import source from the connected database. See *Amazon Redshift Connections*.

**WASB:** If enabled, you can import data into your Azure deployment from WASB.

**ADL:** If enabled, you can import data into your Azure deployment from ADLS Gen1.

**ADLS Gen2:** If enabled, you can import data into your Azure deployment from ADLS Gen1.

**Databases:** If connected to a relational datastore, you can load tables or views from your database. See *Database Browser*.

**NOTE:** For long-loading relational sources, you can monitor progress through each stage of ingestion. After these sources are ingested, subsequent steps to import and wrangle the data may be faster.

For more information, see *Configure JDBC Ingestion*.

For more information, see *Overview of Job Monitoring*.

For more information on the supported input formats, see *Supported File Formats*.

**New/Edit:** Click to create or edit a connection. By default, the displayed connections support import.

**Search:** Enter a search term to locate a specific connection.

**NOTE:** This feature may be disabled in your environment. For more information, contact your Trifacta administrator.

See *Create Connection Window*.

## 2. Add datasets

When you have found your source directory or file:

- You can hover over the name of a file to preview its contents.

**NOTE:** Preview may not be available for some sources, such as Parquet.

- Click the Plus icon next to the directory or filename to add it as a dataset.

**Tip:** You can import multiple datasets at the same time. See below.

- **Excel files:** Click the Plus icon next to the parent workbook to add all of the worksheets as a single dataset, or you can add individual sheets as individual datasets. See *Import Excel Data*.
- If custom SQL query is enabled, you can click **Create Dataset with SQL** to enter a customized SQL statement to pre-filter the table within the database to include only the rows and columns of interest.

Through this interface, it is possible to enter SQL statements that can delete data, change table schemas, or otherwise corrupt the targeted database. Please use this feature with caution.

For more information, see *Create Dataset with SQL*.

If parameterization has been enabled, you can apply parameters to the source paths of your datasets to capture a wider set of sources. Click **Create Dataset with Parameters**. See *Create Dataset with Parameters*.

### 3. Configure selections

When a dataset has been selected, the following fields appear on the right side of the screen. Modify as needed:

- **Dataset Name:** This name appears in the interface.
- **Dataset Description:** You may add an optional description that provides additional detail about the dataset. This information is visible in some areas of the interface.

**Tip:** Click the Eye icon to inspect the contents of the dataset prior to importing.

**Tip:** You can select a single dataset or multiple datasets for import.

#### Edit settings

You can edit any additional or optional settings for an individual dataset. Perform the following:

##### Steps:

1. Click **Edit Settings** from the card for an individual dataset in the right panel. The dialog box is displayed.
2. In the dialog box, select the required options and modify the settings.
  - **File Import Settings:** For more information, see *File Import Settings*.
  - **Table Import Settings:** For more information, see *Table Import Settings*.

### 4. Import selections

#### Single dataset

If you have selected a single dataset for import:

**Tip:** If present, you can click the **Add to new flow** checkbox, which adds the imported datasets to an untitled flow. For more information, see *Flow View Page*.

- Click **Continue**. The dataset is imported.
- A recipe is created for it, added to a new flow, and loaded in the Transformer page for wrangling. See *Transformer Page*.

#### Multiple datasets

You can import multiple datasets from multiple sources at the same time. In the Import Data page, continue selecting sources, and additional dataset cards are added to the right panel.

**NOTE:** If you are importing from multiple files at the same time, the files are not necessarily read in a regular or predictable order.

**NOTE:** When you import a dataset with parameters from multiple files, only the first matching file is displayed in the right panel.

In the right panel, you can see a preview of each dataset and make changes as needed.

×

5 New Datasets

Clear All

POS-r02.txt

×

Add a Description

| # Store_Nbr | # Item_Nbr | #  |
|-------------|------------|----|
| 101         | 322000     | 20 |
| 101         | 323000     | 20 |
| 101         | 325000     | 20 |
| 101         | 326000     | 20 |
| 101         | 327000     | 20 |

Edit settings

POS-r01.txt

×

Add a Description

| # Store_Nbr | # Item_Nbr | #  |
|-------------|------------|----|
| 1           | 381000     | 20 |
| 2           | 325000     | 20 |
| 2           | 325000     | 20 |
| 2           | 402000     | 20 |

☐ Add to new flow

Cancel

Continue

**Figure: Import Multiple Datasets**

If you have selected multiple datasets for import:

**Tip:** If present, you can click the **Add to new flow** checkbox, which adds the imported datasets to an untitled flow. For more information, see *Flow View Page*.

- To import the selected datasets, click **Continue**.
  - To begin transforming one of these datasets in Flow View, select it. From its context menu, select **Add new recipe**. Select the recipe. In the context panel on the right, select **Edit Recipe**. See *Transformer Page*.
- To remove a dataset from import, click the X in the dataset card.

# Create Connection Window

Through the Create Connection window, you can create and edit connections between Designer Cloud powered by Trifacta® Enterprise Edition and remote storage.

**NOTE:** Access to this page in the application and privileges on its related objects is governed by roles in your workspace. For more information, please contact your workspace administrator.

This window is available from the following locations:

- **From Import Data page:** By default, the window displays connections that support import. Deselect the checkbox to display all available connection types.
- **From Run Job page:** When you add a new connection as part of a publishing action, the window displays connections that support publishing by default.
- **From the Connections page:** All available connections are displayed.

**NOTE:** Some connection types may not be available for your environment.

**NOTE:** In your environment, creation of connections may be limited to administrators only. For more information, please contact your Trifacta administrator.

**Tip:** Administrators can edit any public connection.

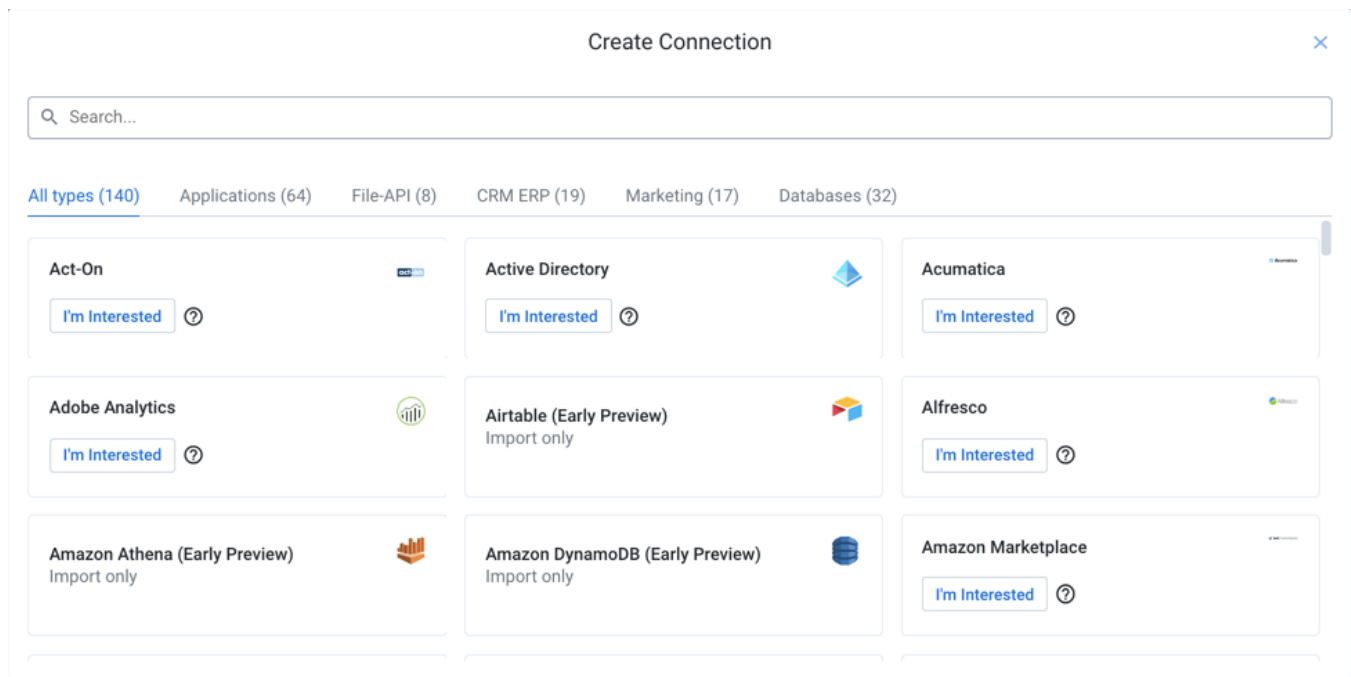
## General Connection Notes:

- After you create a connection, you cannot change its connection type. You must delete the connection and start again.
- Connections can be created, managed, shared, and deleted through the Connections page. See *Connections Page*.

## Database Connection Notes:

- Database connections cannot be deleted if their databases host imported datasets that are in use by Designer Cloud powered by Trifacta Enterprise Edition. Remove these imported datasets before deleting the connection.
- Jobs created for datasets sourced from a database cannot be executed on a Spark-based running environment.

## Connection Type



**Figure: Connection Type window**

In the Connection Type window, you search or browse for the type of connection to create.

### Actions:

- Use the Search bar to perform real-time searches of connection types.
- Click one of the categories to browse for connection types that apply to the listed environment.
- Select the type of connection to continue:
  - **Import only** - Connection can be used only to import data into the platform.
  - **Publish only** - Connection can be used only to publish data from the platform to the connection target.
  - **Import and publish** - Connection can be used to import data and to publish your outputs.

For more information on these connections, See *Connection Types*.

Create Connection

Create Connection

PostgreSQL

Import and publish

SERVER INFORMATION

Host

myserver

Port

5432

Connect String Options (optional)

☒ Enable SSL

Database

myDB

User Name

dbuser

Password

\*\*\*\*\*

Test Connection

< Back

Cancel

Create

Figure: Create Connection Window

| Property               | Description                                                                                                                                                                                                                                      |
|------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Host                   | Host of the database.                                                                                                                                                                                                                            |
| Port                   | Port by which to access the database host.<br>Default values are pre-populated based on the connection type you selected.                                                                                                                        |
| Connect String Options | (optional) If access to the database requires special connection string options, you may paste or enter them here.<br>You only need to provide the parameter and string value. Example:<br><div>" ;transportMode=http;httpPath=cliservice"</div> |
| Enable SSL             | To connect using SSL, click this checkbox.<br><br>If this checkbox is not present, SSL connections for this database type are not supported or are required:                                                                                     |



|                                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                      | <ul style="list-style-type: none"> <li>• SSL connections are not supported for SQL Server or Hive.</li> <li>• SSL connections are required for Redshift and SQL DW.</li> </ul> <p>No additional Connect String Options are required for supported database vendors.</p> <div> <b>NOTE:</b> The database must be configured to receive SSL connections. </div>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| Service Name                                         | (Oracle only) Name of the service. For example, enter <code>orcl</code> here.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| Database                                             | (PostgreSQL only) Name of the database to connect. The name of the default database is the username, so you should change this value in most cases.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| Credential Type                                      | <p>Depending on the type of datastore to which you are connecting, you may have multiple methods of providing credentials for authentication:</p> <ul style="list-style-type: none"> <li>• <code>Basic</code> - Username and password credentials are provided as part of the connection definition.</li> <li>• <code>OAuth 2.0</code> - Connection accesses the datastore using OAuth 2.0 authentication.</li> </ul> <div> <b>NOTE:</b> OAuth 2.0 authentication requires additional configuration. For more information, see <i>Enable OAuth 2.0 Authentication</i>. </div> <div> <b>NOTE:</b> For each type of connection that uses OAuth 2.0, you must create a client app and a client in the Designer Cloud application . See <i>Create OAuth2 Client</i>. </div> <div> <b>NOTE:</b> When you create the connection in this window for an OAuth 2.0 connection, you must click <b>Authenticate</b>, which uses the OAuth 2.0 client to connect to the app. This step is required. </div> |
| User Name                                            | (basic credential type) Username to access the database. This value is encrypted for security.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| Password                                             | (basic credential type) Password for the specified user. This value is encrypted for security.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| OAuth 2.0 Client                                     | <p>(OAuth 2.0 credential type) Select the OAuth 2.0 client to use to connect to the datastore.</p> <div> <b>NOTE:</b> You must create a separate connection for each OAuth 2.0 client that is available in the drop-down list. </div>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| Test Connection                                      | When the above properties are specified, click <b>Test Connection</b> to validate that Designer Cloud powered by Trifacta Enterprise Edition can connect to the database.If the connection test fails, your administrator may need to install a keyfile. See <i>Relational Access</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| Advanced Options: Default Column Data Type Inference | You can choose to enable or disable type inferencing for individual connections, when the connection is created or edited.The default setting for this parameter is defined at the global level. For more information, see <i>Configure Type Inference</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| Connection Name                                      | <p>Display name of the connection, which appears in the application.</p> <div> <b>NOTE:</b> This value must be unique among all connections. </div>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| Connection Description                               | User-friendly description for the connection, which appears in the application.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |

When you've finished, click **Ok** to save the connection.

After you have created your connection, run a simple job on data sourced from it.

**NOTE:** You can make the connection available to all users by sharing it. See *Connections Page*.



# Database Browser

Contents:

- *Browse Databases*
- *Search List*
- *Preview Table Data*
- *Create Dataset with SQL*

The database browser enables you to interact with databases that are connected to Designer Cloud powered by Trifacta® Enterprise Edition.

The database browser appears when:

- You select one of the database tabs to create an imported dataset. See *Import Data Page*.
- You add a publishing action in the Run Job page and choose a database connection through which to write the job results. See *Run Job Page*.

For more information about interacting with databases through the product, see *Using Databases*.

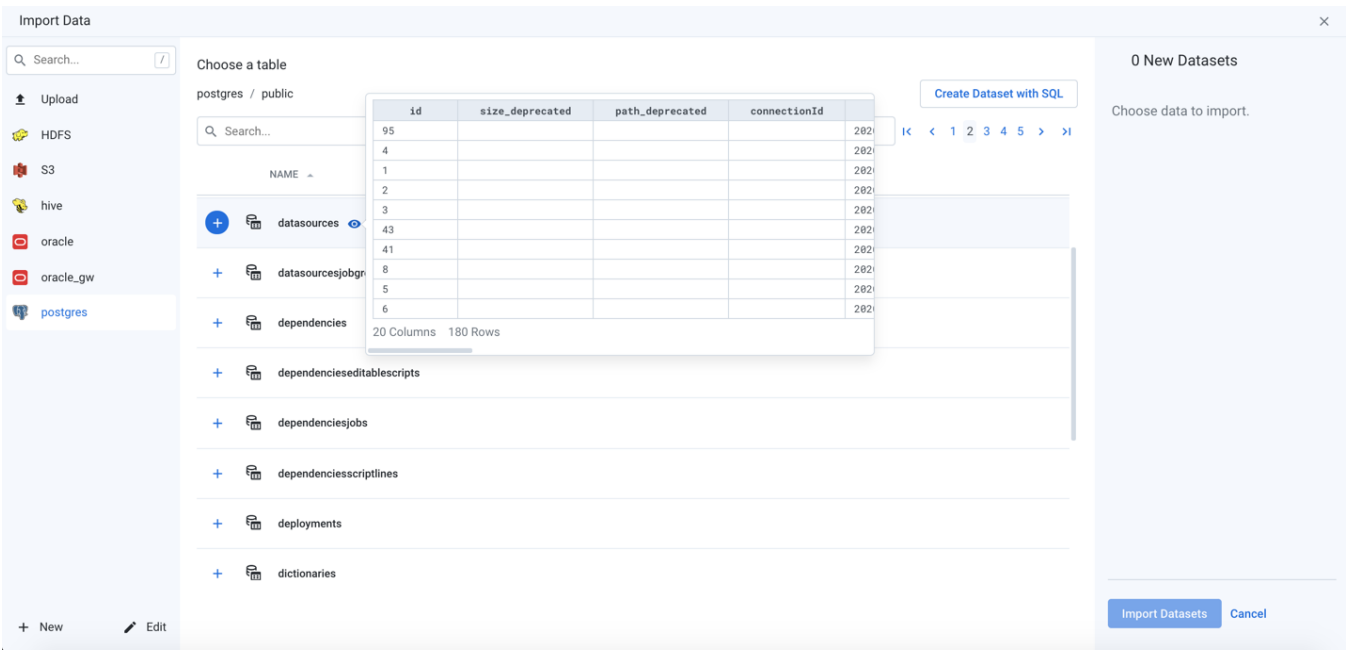






Figure: Database Browser

## Browse Databases

Use the links and icons to browse databases and their tables and views.

**NOTE:** Avoid using the Back button on your browser, which exits the browser without applying changes to your configuration.

| Identifier | Type | Description |
|------------|------|-------------|
|------------|------|-------------|

|                                                                                   |                 |                                                                                                                                                                                                                                                                                                                                                                                              |
|-----------------------------------------------------------------------------------|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  | Database        | Click these links to open a database to reveal its tables and views.                                                                                                                                                                                                                                                                                                                         |
|                                                                                   | Schema          | (Postgres only) Click a schema link to display the tables and views that use the schema.                                                                                                                                                                                                                                                                                                     |
|  | Table           | <p>Click the Plus icon to select this table.</p> <p>To preview its data, hover over the name of the table, and then click the Eye icon.</p> <div> <p><b>Tip:</b> Sizes and update timestamps are calculated and displayed next to tables. They are not displayed next to databases.</p> </div> <div> <p><b>NOTE:</b> Column count information is not available for nested tables.</p> </div> |
|  | View            | <p>Click the Plus icon to select this view as your source.</p> <p>To preview its data, click the Eye icon next to the view name.</p> <div> <p><b>NOTE:</b> Previewing complex views may impact performance.</p> </div>                                                                                                                                                                       |
|  | Page navigation | <p>Use these links to navigate between pages of databases and tables and views.</p> <div> <p><b>NOTE:</b> In some cases, subsequent pages of tables and views may be blank, and counts of tables and views may not match displayed figures. This is a known issue.</p> </div>                                                                                                                |
| postgres / public                                                                 | Breadcrumb      | Click the links in the breadcrumb trail to navigate.                                                                                                                                                                                                                                                                                                                                         |

## Search List

To filter the list, enter a string in the Search box. The filter is applied as you type and matches anywhere in the name of a currently displayed database, table, or view name.

## Preview Table Data

Database tables are displayed by name only. To preview the data in the table, click the Eye icon next to the name of the table.

**Tip:** Table previews include available metadata information, such as column headers and column and row counts.

**NOTE:** Depending on the database type, rows may not be displayed in a specific order.

## Create Dataset with SQL

As needed, you can pre-filter a selected table or view inside the database prior to import. By entering a custom SQL statement, you can remove unnecessary data from the dataset that is extracted from the database, which enables faster and more meaningful imports of your database data. See *Create Dataset with SQL*.

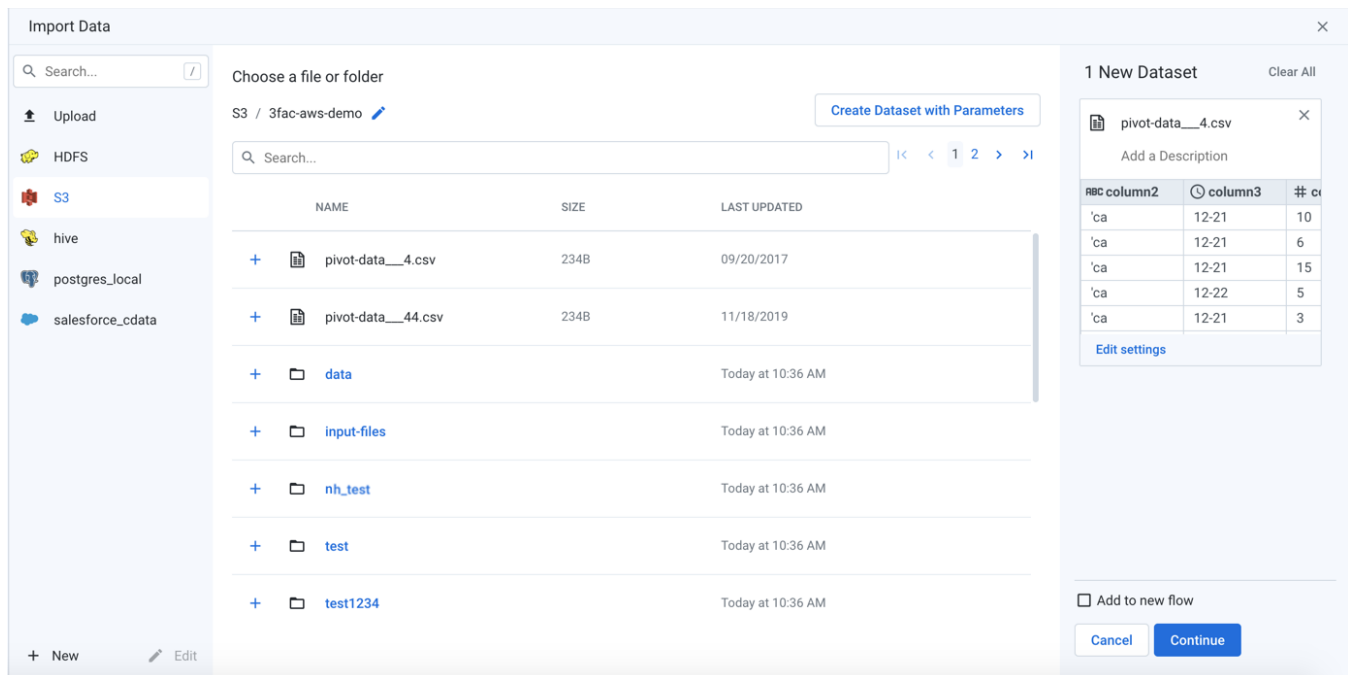
# File System Browser

In Designer Cloud powered by Trifacta® Enterprise Edition, the file system browser lets you browse, select, and filter the sources that you can access through the datastore to which you are connected. You also use the browser to select targets for publishing job results.

Interactions with the connected file system may be determined base on:

- user permissions to specific directories
- features enabled in the product
- any impersonation or kerberos restrictions

For more information, please contact your administrator.



**Figure: File System Browser**

Through the file system browser, you navigate folders and select files through an easy-to-use interface. At a technical level, these objects are typically distributed across multiple servers and may be represented as part files of the whole virtual file.




## Browse:

**NOTE:** Avoid using the Back button on your web browser, which exits the file system browser without applying changes to your configuration.

Use the links and icons to browse for files and folders in the file system tree structure.

**NOTE:** If you do not have the appropriate permissions, you may not be able to browse all of the folders of the directory. However, you may be able to paste in the full path to your location to gain access.

**Tip:** If the displayed file system is the base storage layer, then the path to your output home directory should be available through the browser.

| Identifier                                                                        | Type   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|-----------------------------------------------------------------------------------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  | Bucket | (not always present) In some file systems, the top level browsing object is called a <b>bucket</b> .<br><br><b>NOTE:</b> You cannot add an entire bucket as a source of data for your datasets.                                                                                                                                                                                                                                                                                                                                                    |
|  | Folder | <ul style="list-style-type: none"><li>Click the Plus icon to select all readable files in this folder.</li><li>Click the text link to open the folder and browse further. You must have the appropriate permissions in your account.</li></ul><br><b>Tip:</b> When you open a new folder, a reference to it is added to the Path value. You can modify the path value manually, which may be a faster way to navigate up a deep directory structure.<br><br><b>Tip:</b> Sizes are displayed next to files. They are not displayed next to folders. |
|  | File   | Click the Plus icon to select this file.<br><br>The Last Updated column contains information only for files. It is not available for directories.                                                                                                                                                                                                                                                                                                                                                                                                  |

### Specify Path:

In the browser, you can specify an explicit path to resources. Click the Pencil icon, paste the path value, and click **Go**.

For example, if your home input directory is the following:

```
/mydir/input/username@example.com
```

You should paste the following in the Path textbox:

```
<bucketname>/mydir/input/username@example.com
```

**Tip:** You can retrieve your home directory from your profile.

### Search Files:

To display a subset of files, enter a string in the Search box. The filter is applied as you type and matches anywhere in the name of a currently displayed file or folder.

**NOTE:** If you have a folder and file with the same name, search may only retrieve the file. You can still navigate to locate the folder.

# HDFS Browser

The HDFS browser enables you to browse, select, and filter the files to which you have access in the Hadoop cluster to which Designer Cloud powered by Trifacta® Enterprise Edition is connected.

The HDFS browser appears when you create a dataset in HDFS or in the HDFS tab when you import a dataset. See *Import Data Page*.

**NOTE:** Interactions with HDFS are determined by user permissions and features enabled in the Designer Cloud powered by Trifacta platform . For more information, see *Using HDFS*.

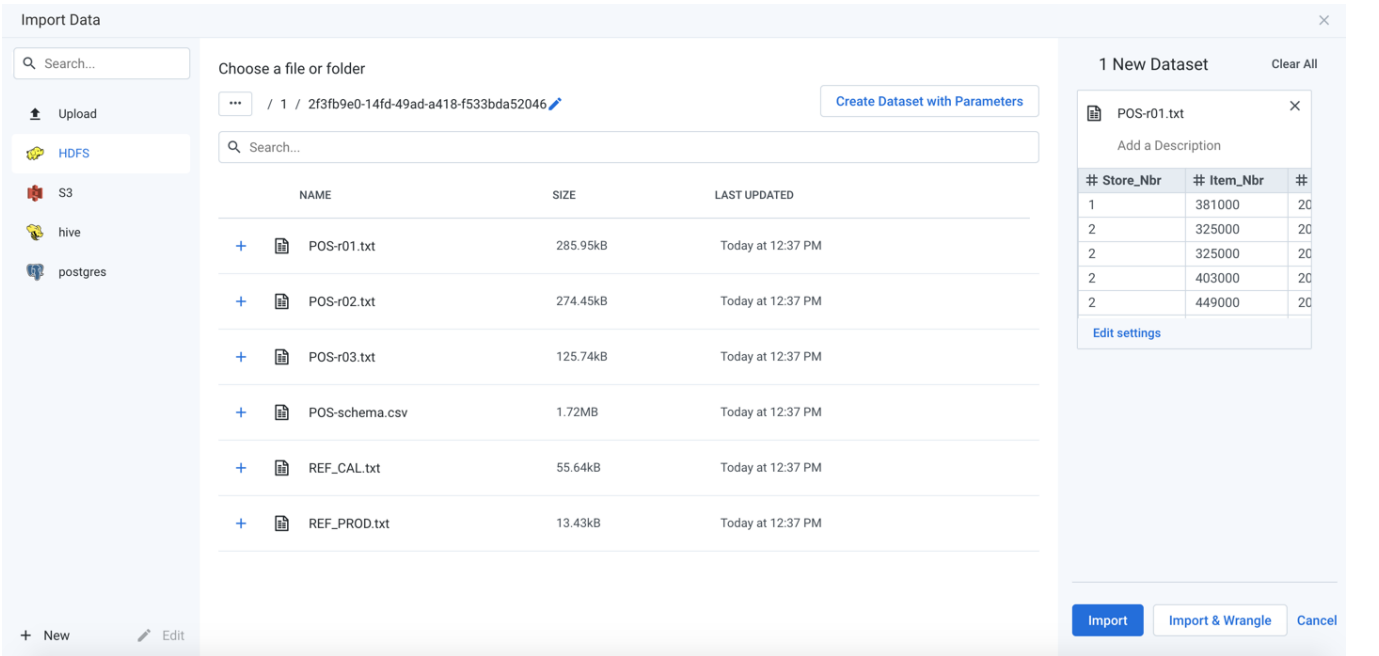




Figure: HDFS Browser

## Browse HDFS:

Use the links and icons to browse for files and folders in the HDFS tree structure.

**NOTE:** Avoid using the Back button on your browser, which exits the HDFS browser without applying changes to your configuration.

| Identifier                                                                          | Type   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|-------------------------------------------------------------------------------------|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  | Folder | <ul style="list-style-type: none"><li>Click the Plus icon to select all readable files in this folder.</li><li>Click the text link to open the folder and browse further.</li></ul> <div><b>Tip:</b> When you open a new folder, a reference to it is added to the Path value. You can modify the path value manually, which may be a faster way to navigate up a deep directory structure.</div> <div><b>Tip:</b> Sizes are displayed next to files. They are not displayed next to folders.</div> |
|  | File   | Click the Plus icon to select this file.                                                                                                                                                                                                                                                                                                                                                                                                                                                            |



---

### Specify HDFS Path:

In the HDFS browser, you can specify an explicit path to resources. Click the Pencil icon, paste the path value, and click **Go**.

```
/trifacta/input/username@example.com
```

You should paste the following in the Path textbox:

```
HDFS/trifacta/input/username@example.com
```

**Tip:** You can retrieve your home directory from your profile. See *Storage Config Page*.

### Filter Files:

To display a subset of files, enter a string in the Search box. The filter is applied as you type and matches anywhere in the name of a currently displayed file or folder.

# S3 Browser

In Designer Cloud powered by Trifacta® Enterprise Edition, the S3 browser lets you browse, select, and filter the sources that you can access through S3. You also use the browser to select targets for publishing job results.

**NOTE:** Interactions with S3 are determined by user permissions and features enabled in Designer Cloud powered by Trifacta Enterprise Edition. For more information, see *Using S3*.

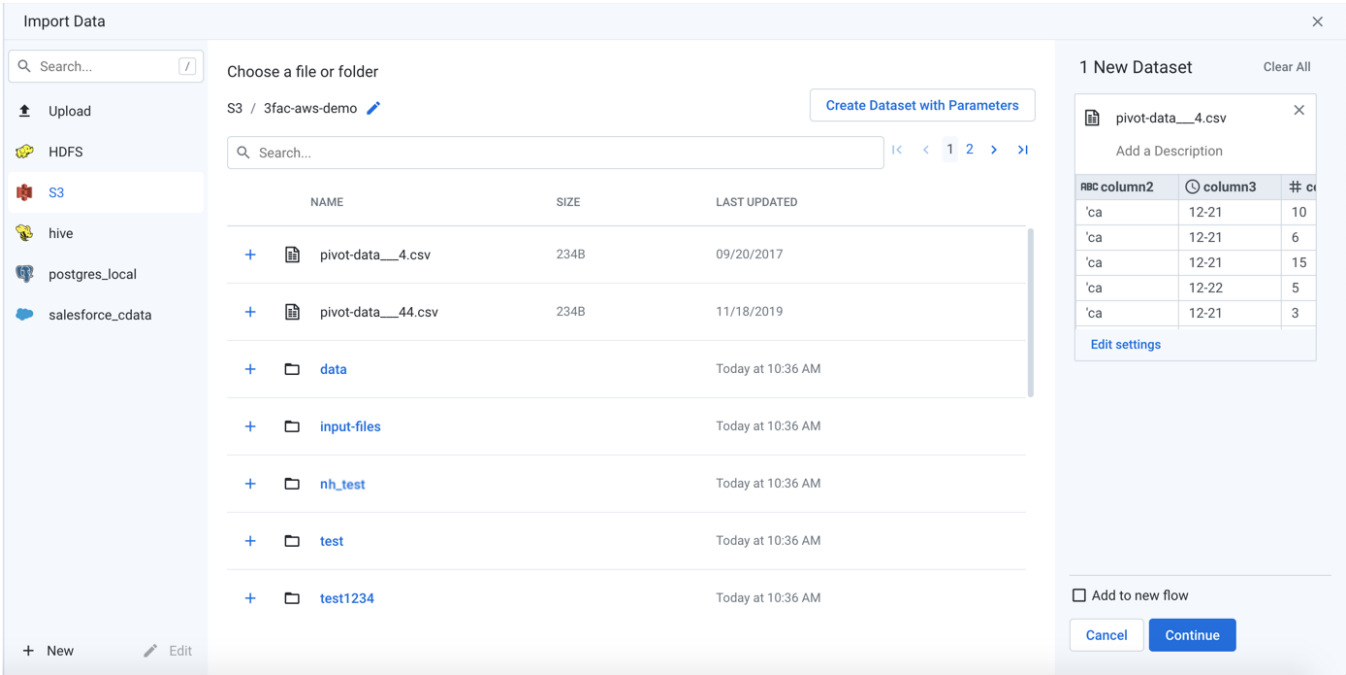


Figure: S3 Browser

## Browse S3:

Use the links and icons to browse buckets for files and folders in the S3 tree structure.



**NOTE:** Permission to browse your buckets is determined by the permissions associated with your S3 credentials. If you do not have the appropriate permissions, you may not be able to browse the bucket. However, you may be able to paste in the full path to your location to gain access.

**Tip:** If S3 is the base storage layer, then the path to your output home directory should be available through the S3 browser. For more information on this path, see *Storage Config Page*.

- The Last Updated column contains information only for files. It is not available for directories.

**NOTE:** Avoid using the Back button on your browser, which exits the S3 browser without applying changes to your configuration.

| Identifier                                                                          | Type   | Description             |
|-------------------------------------------------------------------------------------|--------|-------------------------|
|  | Bucket | Indicates an S3 bucket. |

|                                                                                   |        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|-----------------------------------------------------------------------------------|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                                                   |        | <b>NOTE:</b> You cannot add an entire S3 bucket as a source of data for your datasets.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|  | Folder | <ul style="list-style-type: none"> <li>Click the Plus icon to select all readable files in this folder.</li> <li>Click the text link to open the folder and browse further. You must have the appropriate permissions in your S3 account.</li> </ul> <div> <b>Tip:</b> When you open a new folder, a reference to it is added to the Path value. You can modify the path value manually, which may be a faster way to navigate up a deep directory structure. </div> <div> <b>Tip:</b> Sizes are displayed next to files. They are not displayed next to folders. </div> |
|  | File   | Click the Plus icon to select this file.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |

### Specify S3 Path:

In the S3 browser, you can specify an explicit path to resources. Click the Pencil icon, paste the path value, and click **Go**.

For example, if your home input directory is the following:

```
/mydir/input/username@example.com
```

You should paste the following in the Path textbox:

```
<bucketname>/mydir/input/username@example.com
```

**NOTE:** The name of the bucket (<bucketname>) must appear at the beginning of the path. Do not add a backslash (/) as a prefix.

**Tip:** You can retrieve your home directory from your profile. See *Storage Config Page*.

### Search Files:

To display a subset of files, enter a string in the Search box. The filter is applied as you type and matches anywhere in the name of a currently displayed file or folder.

**NOTE:** If you have a folder and file with the same name in S3, search only retrieves the file. You can still navigate to locate the folder.

# File Import Settings

## Contents:

- *Per-file encoding*
  - *Detect structure*
  - *Remove special characters from column names*
  - *Selecting column headers*
- 

When you edit settings on a selected file in the Import Data page, the following settings are displayed.

You can edit any additional or optional settings for an individual dataset. Perform the following:

1. Click **Edit Settings** from the card for an individual dataset in the right panel. The dialog box is displayed.
2. In the dialog box, select the required options and modify the settings.

## Per-file encoding

By default, Designer Cloud powered by Trifacta Enterprise Edition applies a specified encoding type on the imported file. In some cases, the data preview panel may contain garbled data, due to a mismatch in encodings. In the Data Preview dialog, you can select a different encoding for the file. When the correct encoding is selected, the preview displays the data as expected.

**NOTE:** Assessing the file encoding type based on parsing an input file is not an accurate method. Instead, Designer Cloud powered by Trifacta Enterprise Edition assumes that the file is encoded in the default encoding. If it is not, you should change the encoding type for the file.

**NOTE:** In some cases, imported files are not properly parsed due to issues with encryption types or encryption keys in the source datastore. For more information, please contact your datastore administrator.

For a list of supported encoding types, see *Supported File Encoding Types*.

For more information on supported encodings, see *Configure Global File Encoding Type*.

## Detect structure

By default, Designer Cloud powered by Trifacta Enterprise Edition attempts to interpret the structure of your data during import. This structuring attempts to apply an initial tabular structure to the dataset.

- Unless you have specific problems with the initial structure, you should leave the Detect structure setting enabled. Recipes created from these imported datasets automatically include the structuring as the first, hidden steps. These steps are not available for editing, although you can remove them through the Recipe panel. See *Recipe Panel*.
- When detecting structure is disabled, imported datasets whose schema has not been detected are labeled, **unstructured datasets**. When recipes are created for these unstructured datasets, the structuring steps are added into the recipe and can be edited as needed.
- For more information, see *Initial Parsing Steps*.

## Remove special characters from column names

When selected, characters that are not alphanumeric or underscores are stripped, and space characters are converted to underscores.

**Tip:** This feature matches the column renaming behavior in Release 5.0 and earlier.

For more information, see *Sanitize Column Names*.

## Selecting column headers

You can apply the column headers to your datasets during import. Select the required option from the drop-down list:

- **Infer header:** (default) When selected, the Designer Cloud application infers the header based on the data in the import.
- **Use first row as header:** When selected, the first row is used as the column headers.
- **No header:** When selected, the inference is ignored and column headers are defined using generic names with no headers.

If replacing a file:

- If you replace a dataset in a flow and select the **Use first row as header** option, then the existing header row labels are updated with the new headers.
- Subsequent steps in a pre-existing recipe may be broken if the headers are changed by a replaced file.

**Tip:** After the dataset is imported, you can rename columns manually or using any row in the dataset. For more information, see *Rename Columns*.

# Table Import Settings

When you edit settings for a selected table in the Import Data page, the following settings are displayed.

You can edit any additional or optional settings for an individual dataset. Perform the following:

1. Click **Edit Settings** from the card for an individual dataset in the right panel. The dialog box is displayed.
2. In the dialog box, select the required options and modify the settings.

## Infer column data types

You can choose whether or not to apply Designer Cloud powered by Trifacta Enterprise Edition type inference to table data imported from a database.

- In the preview panel, you can see the data type that is to be applied after the dataset is imported. This data type may change depending on whether column data type inference is enabled or disabled for the dataset.
- To enable Designer Cloud powered by Trifacta Enterprise Edition type inference, select the Infer column data types checkbox.

**Tip:** To see the effects of Designer Cloud powered by Trifacta Enterprise Edition type inference, you can toggle the checkbox and review data type listed at the top of individual columns. To override an individual column's data type, click the data type name and select a new value.




You can configure the default use of type inference at the individual connection level. For more information, see *Create Connection Window*.

For schematized sources that do not require connections, such as uploaded Avro files, the default setting is determined by the global setting for initial type inference. For more information, see *Configure Type Inference*.

# Macros Page

In the Macros page, you can review and manage the macros to which you have access.

A **macro** is a saved sequence of one or more recipe steps that can be reused in other recipes. Values in your macros can be parameterized. For more information, see *Overview of Macros*.

| Macros                                                                                                                                |                     |                  | Find Macros... |
|---------------------------------------------------------------------------------------------------------------------------------------|---------------------|------------------|----------------|
|                                                                                                                                       |                     |                  |                |
| Name                                                                                                                                  | Used in             | Last Updated ▾   |                |
|  <b>InitialCleanv3</b><br>Initial steps cleanup v3   | 1 Flow • 1 Recipe   | Today at 5:36 PM |                |
|  <b>InitialCleanV2</b><br>Initial cleanup steps      | 0 Flows • 0 Recipes | Today at 5:34 PM |                |
|  <b>InitialClean</b><br>Initial recipe cleanup steps | 0 Flows • 0 Recipes | Today at 5:31 PM |                |

**Figure: Macros Page**

To review specifics about the macro, click its name. See *Macro Details Page*.

## Columns:

- **Name:** Name of the macro.
- **Used in:** Count of flows and recipes in which the macro is used.
- **Last Updated:** Timestamp for when the macro was last modified.

## Actions:

- **Import Macro:** Click to import a macro that has been exported to your local desktop. For more information, see *Import Macro*.
- **Get Macros:** Explore and download macros from Wrangle Exchange.
- **Search:** Enter a string in the search box. The list of macros is updated in real-time.
- **Sort:** Click the caret next to any column head to sort the list based on the column.
- **Context Menu:** See below.

## Context Menu Options:

- **Edit:** Modify the name and description for the macro. You can also modify the name, description, and default values for the macro's inputs.
- **Inspect:** Review the recipe steps in the macro.
- **Export:** Export the macro to your local desktop. For more information, see *Export Macro*.
- **Replace:** Replace an existing macro definition with a macro that you have exported to your local desktop. For more information, see *Export Macro*.

**NOTE:** If your imported macro contains macro inputs that are not in the macro that you are replacing, the existing instances of the replaced macro contain a broken step where the macro is referenced but has no data. These references must be fixed in each macro instance. For more information, see *Macro Details Page*.

**Tip:** If you must add more macro inputs or steps to a macro that you have imported, you must convert the macro to steps, modify them, and then perform a replacement. For more information, see *Create or Replace Macro*.

- **Delete:** Delete the macro.

**NOTE:** When a macro is deleted, in any recipe that references it, the macro's steps are expanded into regular recipe steps. Any macro inputs are applied as static values in the expanded recipe steps.

**Deleting a macro cannot be undone.**



# Macro Details Page

Through the Macro Details Page, you can review details about an individual macro. In the Macros page, click the name of the macro to review.

### Actions:

To modify the macro name and description, click **Edit**.

**Tip:** You can also modify the name, description, and default values for the macro's inputs.

### Context menu:

- **Export:** Export the macro to your local desktop. See *Export Macro*.
- **Delete:** Delete the macro.

**NOTE:** When a macro is deleted, in any recipe that references it, the macro's steps are expanded into regular recipe steps. Any macro inputs are applied as static values in the expanded recipe steps.

**Deleting a macro cannot be undone.**

## Overview Tab

In the Overview tab, you can review the steps in the macro.

Macros

InitialCleanv3

Initial steps cleanup v3

Edit

...

Overview

Used in

Macro steps

1

rename

type: findAndReplace

col: \* on: '' with: '' matchAll: true

2

derive

type: single value: \$sourcerownumber as: \$rowindex

3

move

col: \$rowindex position: after after: \$Current\_HO\_Retail

Details

Created

Today at 5:36 PM

Last updated

Today at 5:36 PM

**Figure: Macro Detail Page - Overview tab**

- Steps are displayed in raw Wrangle .
- You can review creation and update timestamps.

## Used In Tab

In the Used In tab, you can review all of the recipes and flows where the macro is referenced.

| <div> <div> <div>Macros</div> <div>InitialCleanv3</div> <div>Initial steps cleanup v3</div> </div> <div> <div>Edit</div> <div>...</div> </div> </div> |              |                  |
|-------------------------------------------------------------------------------------------------------------------------------------------------------|--------------|------------------|
| <div> <div>Overview</div> <div>Used in</div> </div>                                                                                                   |              |                  |
| Name                                                                                                                                                  | Flow         | Last Updated ▾   |
| <div> <div></div> <div>POS-r01</div> </div>                                                                                                           | POS-r01 Flow | Today at 5:36 PM |

**Figure: Macro Detail Page - Used In tab**

- Click the name of the recipe or flow to open the flow. See *Flow View Page*.

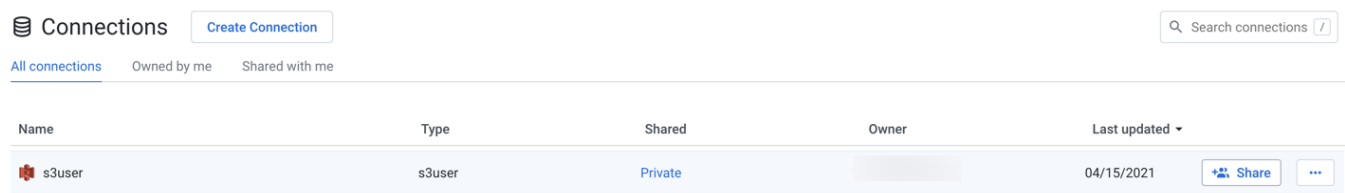
# Connections Page

**Contents:**

- *Top Bar*
- *Connection context menu*
- *Connection Details Panel*

Through the Connections page, you can add new connections or modify the connections that you have already created. From the left nav bar, click the Connections icon.

**NOTE:** Access to the Connections page in the application and privileges on connections is governed by roles in your workspace. For more information, please contact your workspace administrator.



**Figure: Connections page**

**Fields:**

- **Name:** Display name for the connection.

**NOTE:** If the connection has been shared, you can review whether its credentials have also been shared.

- **Type:** The type of connection.

**NOTE:** After you create a connection, you cannot modify its type.

For more information, see *Connection Types*.

- **Shared:** Review the sharing status of the connection:
- Global - connection has been shared with all users of the workspace.

**NOTE:** To make a global connection private, you must delete the connection and recreate it.

- X Users:
  - If this value is 1, the connection is private.

- If this value is greater than 1, the connection has been shared. Click the link in this column to review sharing status. See *Share Connection Dialog*.

## Top Bar

- **Create:** Click **Create Connection** to create a new connection. See *Create Connection Window*.
- **Filter:** In separate tabs, you can review connections that you own, that are shared with you, or all connections to which you have access.
- **Search:** Search connections by name.
- **Review details:** Select a connection or click the icon to review details through the right-side panel.

## Connection context menu

- **Share:** For connections that you own, you can modify the sharing status of them. See *Share Connection Dialog*.
- **View Details:** Open the details of the connection in the side panel. See below.
- **Edit:** If you own the connection, you can review and modify the connection.
  - If the connection has been shared with you, you can edit it to modify the credentials.
  - Administrators can edit public connections.
  - See *Create Connection Window*.
- **Delete:** Delete the connection.


**NOTE:** This option is only available to the connection owner if the connection is not used for any datasets.

## Connection Details Panel

When a connection is selected, you can review its details and make modifications as needed through the panel on the right.

Connection Details

×

 postgres

Edit Connection

...

Connection Type

postgres

Shared

Private

Owner

SteveO

Created

Today at 5:28 PM

Updated

Today at 5:28 PM

Updated by

SteveO

Server Information

Host

myHost

Port

5432

SSL

Disabled

Database

myDB

Username

myUserId

Password

.....

**Figure: Connection Details panel**

**Key Fields:**

- **Shared:** Number of users sharing the connection. If a link is present, click it to modify sharing of the connection. See *Share Connection Dialog*.
- **Server Information:** For server-based connections, you can review the connection properties.

**Actions:**

- **Edit Connection:** If you own the connection, you can review and modify the connection.
  - If the connection has been shared with you, connection properties are read-only.
  - See *Create Connection Window*.
- **Share:** You can share connections that you own or that are shared with you. See *Share Connection Dialog*.
- **Delete:** Delete the connection.

**Deleting a connection cannot be undone.**

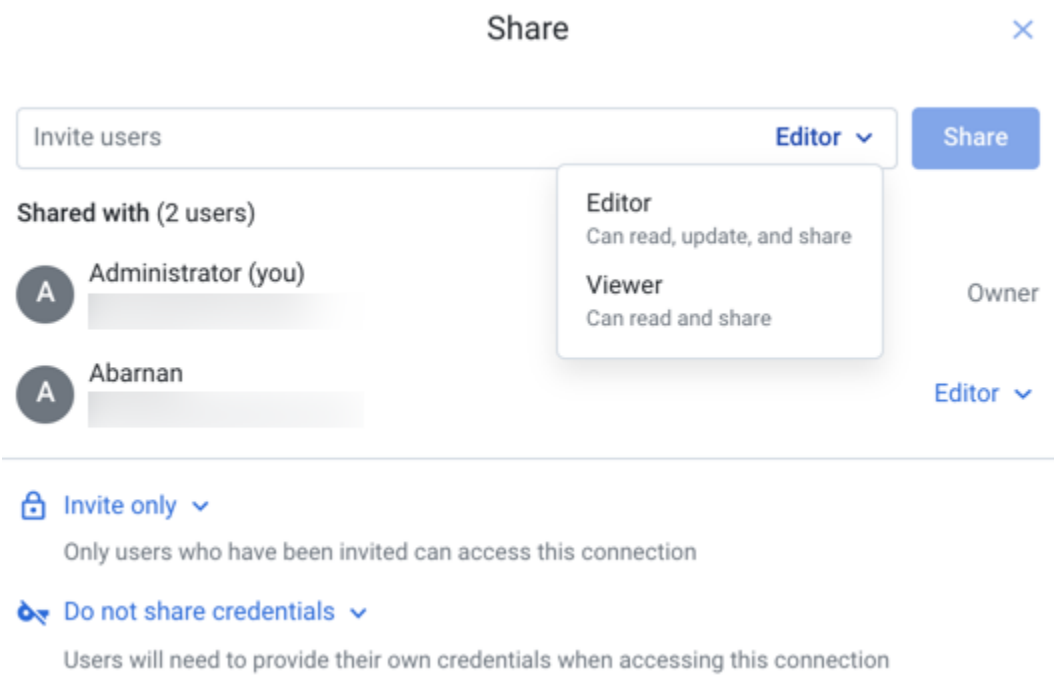
# Share Connection Dialog

Contents:

- *Actions*
- *Find Users*
- *Set Access Level*
- *Privileges*
- *Privacy*
- *Credentials*

Through the Share Connection dialog, users of the selected connection with appropriate privileges can modify who has access to the connection.

**Tip:** A workspace administrator has owner-level access to all connections in the workspace. However, a workspace admin cannot access or use a connection's credentials if those credentials were not shared by the owner of the connection. For more information, see *Workspace Admin Permissions*.



**Figure: Share Connection Dialog**

Actions

Find Users

Start typing names or email addresses of users to see matches.

**Tip:** You can paste a comma-separated list of email addresses to share to multiple users at the same time.

You may be able to browse a list of all user names.

**NOTE:** This feature may need to be enabled in your environment. For more information, see *Workspace Settings Page*.

## Set Access Level

As needed, you can configure the level of access to the connection for users with whom the connection is shared.

**NOTE:** You cannot set a user's access to a level that is higher than the limit set for the user at the workspace level. For example, if the user has Viewer access to connections at the workspace level, you cannot make the user an Editor on your connection.

**NOTE:** Administrators have owner-level access to all connections in the workspace or project. You do not need to share connections with them.

## Privileges

- **Editor:**
  - User can use the connection to read data, update the connection, and share the connection.
  - User has all Viewer privileges.

**NOTE:** Editors cannot delete connections. Only the owner or an admin can delete a connection.

- **Viewer:**
  - User can use the connection to read data.
  - User can share connection.

For more information, see *Overview of Sharing*.

## Privacy

- **Invite only:** Connection can be made available to other users only by invitation through this window.
- **Public:** Connection is available to all workspace users who can access connections.

**NOTE:** Only an administrator can make a connection public.

**NOTE:** After a connection is made public, it cannot be made private again. It must be deleted and recreated.

## Credentials

By default, a connection is shared with credentials. Optionally, sharing of credentials can be disabled when sharing.

**NOTE:** The choice to share credentials or not is applied to all users with whom the connection has been shared, including users with whom the connection has been shared previously.

**NOTE:** Connections that use OAuth 2.0 authentication cannot be shared with credentials.

- **Share credentials:** When selected, the credentials that are specified in the owner's connection definition are shared with other users.

**NOTE:** Password values are always masked in the interface.

- **Do not share credentials:** When this option is selected, users of the shared connection must provide their own credentials.

**NOTE:** To use datasets previously imported through the shared connection, these credentials must provide access the source data. If shared credentials are removed from a connection, then any datasets imported through the connection are not accessible until credentials are provided. This may also apply to re-publishing previously generated results.

### When credentials are shared:

**NOTE:** Users to whom credentials are shared cannot see any passwords in the Designer Cloud application .

- The credentials specified in the connection are shared to the users who are specified in the Share dialog for connections.
  - Users of a shared connection with credentials cannot insert their own credentials. They must create a new connection.
- Sharing of credentials may not guarantee access to the same locations as available to the owner.  
Examples:
  - If your deployment uses Single Sign On, your enterprise login may provide access controls to the same resource that are different from the connection owner.
  - Network infrastructure may whitelist IP addresses for some users and block the same addresses for others.
  - Depending on the datastore, folder or directory permissions may limit access.
  - For more information, please contact your IT administrator.
- The owner of the connection can specify whether credentials are shared or not.
  - A workspace administrator has owner-level access to all connections in the workspace. However, a workspace admin cannot access or use a connection's credentials if those credentials were not shared by the owner of the connection.
- Shared users of the connection can share the connection if they have Editor privileges.



**When credentials are not shared:**

- Each user must provide credentials to use the connection.
  - A user's individual credentials may not provide read access to datasources, which may mean that imported datasets appear to be broken.
  - Individual credentials may not provide write access to the same output locations, which may cause jobs to fail.
- When sharing of credentials is disabled, shared users who share with other users cannot include credentials as part of the share.

# Jobs Page

In the Jobs page, you can track the status of all of your jobs and plan runs.

- **Sample jobs:** For more information on sampling jobs, see *Sample Jobs Page*.
- **Plan runs:** For more information on plan runs, see *Plan Runs Page*.

Jobs can be initiated from:

- Flow View: See *Flow View Page*.
- Transformer page: See *Transformer Page*.

⌚ Flow jobs 1 2 > >> ⌵

| All jobs                             | Completed                   | Failed             | Canceled               | Running                                       | Queued |
|--------------------------------------|-----------------------------|--------------------|------------------------|-----------------------------------------------|--------|
| Job                                  | Status                      | Flow               | User                   | Started                                       |        |
| ⌚ POS-r01<br>Job ID: 2135347         | 🔄 In progress               | 🔗 2013 POS         | Abarna Nagarajan (you) | Today at 9:51 AM <a href="#">Cancel job</a> ⋮ |        |
| ⌚ POS-r01<br>Job ID: 2135265         | ✅✅ Completed                | 🔗 2013 POS         | Abarna Nagarajan (you) | Today at 9:38 AM<br>Ran for 5 minutes         |        |
| ⌚ POS-r01<br>Job ID: 2130018         | ⚠️✅ Completed with warnings | 🔗 2013 POS         | Abarna Nagarajan (you) | Yesterday at 3:05 PM<br>Ran for 7 minutes     |        |
| ⌚ POS-r01<br>Job ID: 2129615         | ⚠️✅ Completed with warnings | 🔗 2013 POS         | Abarna Nagarajan (you) | Yesterday at 1:15 PM<br>Ran for 8 minutes     |        |
| ⌚ POS-r01<br>Job ID: 2129611         | ⚠️✅ Completed with warnings | 🔗 2013 POS         | Abarna Nagarajan (you) | Yesterday at 1:06 PM<br>Ran for 7 minutes     |        |
| ⌚ POS-r01<br>Job ID: 2127857         | ⚠️✅ Completed with warnings | 🔗 2013 POS         | Steve Olson            | Yesterday at 4:04 AM<br>Ran for 7 minutes     |        |
| ⌚ Untitled recipe<br>Job ID: 2105451 | ❌❌ Canceled                 | 🔗 [cf8d8640] test1 | Abarna Nagarajan (you) | Last Friday at 5:40 PM<br>Ran for 5 minutes   |        |
| ⌚ POS-r01<br>Job ID: 1936242         | ✅✅ Completed                | 🔗 2013 POS         | Steve Olson            | 04/24/2021<br>Ran for 8 minutes               |        |

**Figure: Jobs page**

## Job Types:

Each job listed in the Jobs page is a grouping of related jobs acting on the same recipe and dataset(s). Each of these **jobgroups** breaks down into one or more of the following job types.

**Tip:** To review the status of individual jobs within a jobgroup, hover over the icons in the Status column for the jobgroup.

- **Pre-ingest SQL:** These jobs are SQL scripts that execute before the source data is ingested to the platform.
  - For additional details on these jobs, see the SQL scripts tab in the Job Details page. See *Job Details Page*.
  - For more information on these types of SQL scripts, see *Create Output SQL Scripts*.
- **Transform:** These jobs perform transformations on imported datasets based on the recipe from which the job was launched.

- **Profile:** If enabled as part of the job definition, a Profile job generates a visual summary of the results of your transformation job.
  - Profiling jobs may take longer than transformation jobs.
  - Even when selected, profiling jobs may not appear in the Jobs page. In some cases, a profiling job may be folded into a transform job for optimization reasons.

**NOTE:** When the profiling job is run as part of the transform job, there is no listing for profiling in the mouse-over popup.

- See *Job Details Page*.
- **Publish:** Depending on multiple factors, your job may include a second Publish job that occurs after the Transform job. For example, jobgroups can include internal Publish jobs for writing results to the designated location in the base storage layer.

Publishing can also be executed as a separate, post-execution job. As needed, job results can be published from their target location to another location or data store. These jobs are tracked separately as Publish jobs and can be launched from the Job Details page. For more information, see *Job Details Page*.

- **Ingest:** For larger datasets from some relational connections, the platform transfers the data from the source to the default storage layer for faster processing. These ingest jobs occur before any transform or profiling takes place.
- **Post-ingest SQL:** These jobs are SQL scripts that execute after the job results have been published.
  - For additional details on these jobs, see the SQL scripts tab in the Job Details page. See *Job Details Page*.
  - For more information on these types of SQL scripts, see *Create Output SQL Scripts*.

## Tabs and Statuses:

Each of the available tabs corresponds to a possible status for jobs that have been initiated on the platform.

- **All jobs:** All jobs that you have initiated are listed here.
- **Completed:** Job has successfully executed.

**NOTE:** Invalid steps in a recipe are skipped, and it's still possible for the job to be executed successfully.

**NOTE:** A warning icon may indicate that recipe errors were detected during the Transform phase. You can hover over the icon for more information.

- **Failed:** job failed to complete.

**NOTE:** You can re-run a failed job from the Transformer page. If you have since modified the recipe, those changes are applied during the second run. See *Transformer Page*.

- **Publish Failed:** Some failed jobs may be listed under this status, which means that the publishing step of the configured job failed to complete.
- **Canceled:** Job was canceled by the user.
- **Running:** Job is in progress.
- **Queued:** Job has been queued for execution.

## Access:

- You can review and drill into any job that you initiated.
- You can also drill into any job that was initiated from a flow that has been shared with you.
- Administrators can review read-only listings for jobs created by other users.

#### Columns:

- **Job:** Internal identifier for the job. This value is unique for all jobs in your Trifacta® instance.
  - Click the ID number to explore details about the job. See *Job Details Page*.
- **User:** The Trifacta user that initiated the job.
- **Run from:**
  - Location where the job was launched. Click the link to view details.
- **Status:**
  - See Tabs above.
- **Started:** Start timestamp for the job.
  - Scheduled jobs are indicated with an icon.

#### Actions:

- **Filter by status:** Click one of the tabs to filter the display to show only the listings for the selected job status.
- **Filter by type and date:** Click the Funnel icon to filter the list of jobs by source of execution, date range, or both. See below.
- **Search:** Enter text in the search field to filter the listed jobs by job ID, flow name, or dataset name.

#### Context menu:

Next to the job listing, click the options menu to see the following:

- **Cancel Job:** Select to cancel a job that is currently being executed.
- **Delete job:** Delete the job from the platform.

**Deleting a job cannot be undone.**

**NOTE:** This feature may not be enabled in your instance of the platform. For more information, please contact your Trifacta Administrator. See *Miscellaneous Configuration*.

- **Download Logs:** Download the logs for the job. If the job is in progress, log information is likely to be incomplete.

**Tip:** When jobs fail, the downloaded package includes additional configuration files and service logs to assist in debugging job execution issues. For more information, see *Support Bundle Contents*.

Additional options are available for each job. See *Job Details Page*.

## Filter Jobs

To filter the list of jobs based on dates or source of execution, click the Funnel icon. You can use the following dialog to filter the display of jobs.

Filter Jobs

Job type

Only show manual jobs

Date/Time

Started

Started After

MM/DD/YYYY

HH:MM

AM

Ended

Ended Before

MM/DD/YYYY

HH:MM

AM

Clear Filters

Cancel

Apply

Figure: Filter Jobs dialog

**Job type:**

Show jobs based on the following available options:

- Show all jobs
- Only show manual jobs
- Only show scheduled jobs

**Started:**

- Specify the date and time when the jobs to display began.
- If needed, you can specify the start time as a range. Select `Start Between` from the drop-down list and populate both date-time rows.

**Ended:**

- Specify the date and time when the jobs to display ended.
- If needed, you can specify the end time as a range. Select **Ended Between** from the drop-down list and populate both date-time rows.

**Actions:**

- To clear the time period values, click **Clear Filters**.
- To apply the specified time filter to the Jobs page, click **Apply**.

# Job Details Page

## Contents:

- *Overview Tab*
  - *Output Destinations Tab*
    - *Direct file download*
    - *Create imported dataset*
    - *Publish*
  - *SQL scripts Tab*
  - *Profile Tab*
  - *Dependency graph Tab*
  - *Data sources Tab*
  - *Parameters Tab*
  - *Webhooks Tab*
- 

You can use the Job Details page to explore details about successful or failed jobs, including outputs, dependency graph, and other metadata. Download results to your local desktop or, if enabled, explore a visual profile of the data in the results for further iteration on your recipe.

## Page options:

- **Cancel job:** Click this button to cancel your job while it is still in progress.

**NOTE:** This option may not be available for all running environments. Job cancellation is not supported in high availability deployments.

- **Publish results:** Publish your results to an external system. For more information, see *Publishing Dialog*.
- **Delete job:** Delete the job and its results.

**Deleting a job cannot be undone.**

**NOTE:** This feature may not be enabled in your environment. For more information, see *Miscellaneous Configuration*.

- **Download logs:** Download the log files associated with this job.

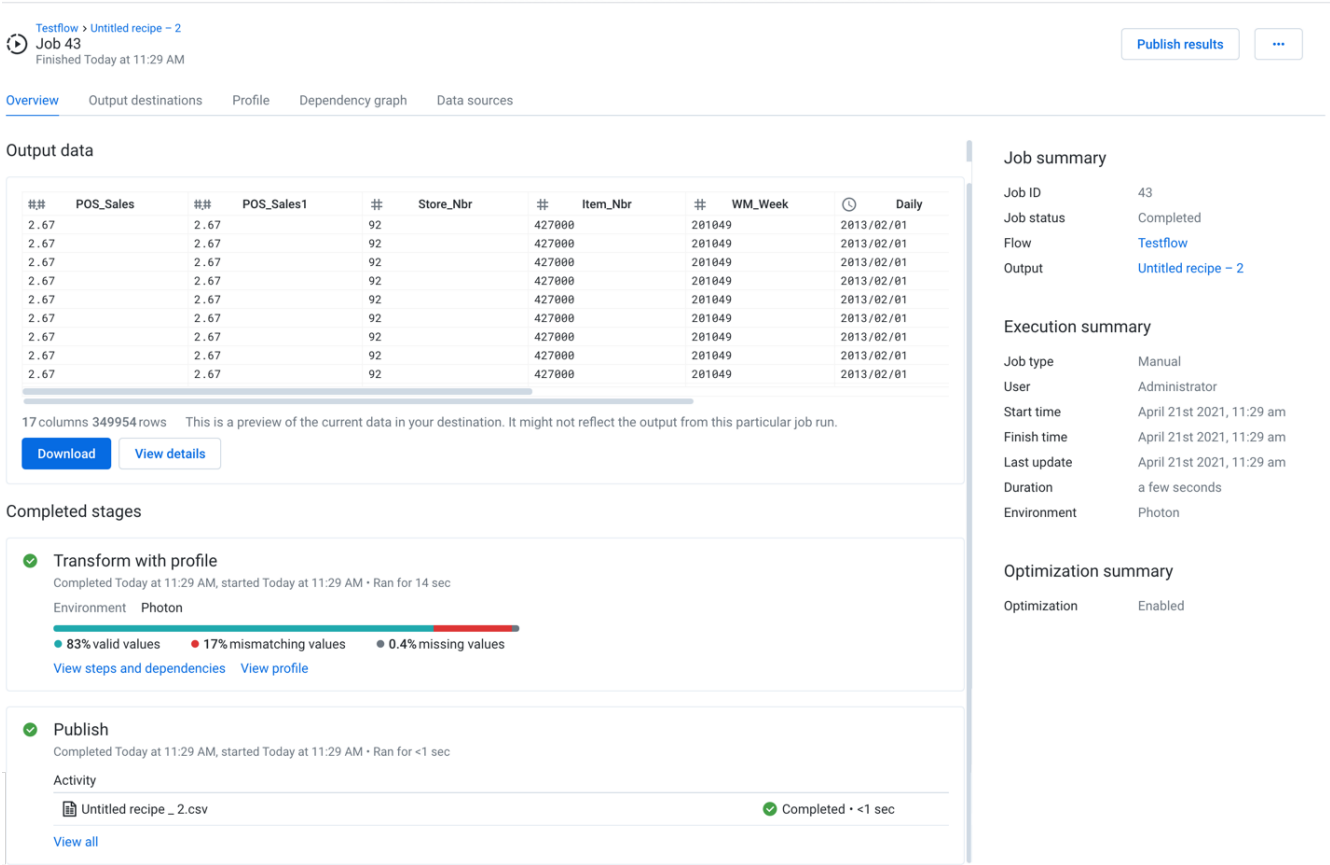
**Tip:** When jobs fail, the downloaded package includes additional configuration files and service logs to assist in debugging job execution issues. For more information, see *Support Bundle Contents*.

- **Download profile as PDF:** If visual profiling was enabled for the job, you can download the profile in PDF format.
- **Download profile as JSON:** If visual profiling was enabled for the job, you can download a JSON representation of the profile to your desktop.

# Overview Tab

In the Overview tab, you can review the job status, its sources, and the details of the job run.

**NOTE:** If your job failed, you may be prompted with an error message indicating a job ID that differs from the listed one. This job ID refers to the sub-job that is part of the job listed in the Job summary.



**Figure: Overview tab**

You can review a snapshot of the results of your job.

## Output Data:

The output data section displays a preview of the generated output of your job.

**NOTE:** This section is not displayed if the job fails.

You can also perform the following:

- **View:** If it is present, you can click the View link to view the job results in the datastore where they were written.

**NOTE:** The View link may not be available for all jobs.



- **Download** : If it is present, click the **Download** link to download the generated job results to your local desktop.
- **View details**: Click **View details** to view the generated results in the side bar. See the Output Destinations below.

### Completed Stages:

**NOTE:** If you chose to generate a profile of your job results, the transformation and profiling tasks may be combined into a single task, depending on your environment. If they are combined and profiling fails, any publishing tasks defined in the job are not launched. You may be able to ad-hoc publish the generated results. See below.

- If present, you can click the **Show Warnings** link to see any warnings pertaining to recipe errors, including the relevant step number. To review the recipe and dependencies in your job, click **View steps and dependencies**. See the Dependencies tab below.
- If you chose to profile results of your job, click **View profile** to review. See Profile tab below.
  - A visual profile provides a graphical snapshot of the results of a successful transformation job for the entire dataset and individual columns in the dataset.
  - For more information on enabling a visual profile job, see *Run Job Page*.
  - For more information, see *Overview of Visual Profiling*.
- If your job output specified SQL scripts to run before or after job execution, you can track their progress in the following stages:
  - **Pre-ingest SQL**: Script that is configured to run before the source data is ingested to the platform.
  - **Post-publish SQL**: Script that is configured to run after the output data has been published.
  - For additional details, see the SQL scripts tab below.
  - For more information on SQL scripts in job execution, see *Create Output SQL Scripts*.

### Job Monitoring:

You can hover over the status of each stage of a job to review breakdowns for individual phases of each stage:

**Tip:** Depending on the operation, you may be able to monitor transfer rate performance for larger datasets.

- **Connect**: The platform is attempting to connect to the datastore hosting the asset sources for the datasets.
- **Request**: The platform is requesting the set of assets to deliver.
- **Ingesting**: Depending on the type of source data, some jobs ingest data to the base storage layer in a converted format before processing begins. This ingested data is purged after job completion.
- **Prepare**: (Publishing only) Depending on the destination, the Prepare phase includes the creation of temporary tables, generation of manifest files, and the fetching of extra connections for parallel data transfer.
- **Transfer**: Assets are transferred to the target, which can be the platform or to the output datastore.
- **Process**: Cleanup after data transfer, including the dropping of temporary tables or copying data within the instance.

For more information, see *Overview of Job Monitoring*.

### Publish:

You can also review the outputs generated as a result of your job. To review and export any of the generated results, click **View all**. See [Outputs Destinations](#) tab below.

#### Job summary:

- **Job ID:** Unique identifier for the job

**Tip:** If you are using the REST APIs, this value can be used to retrieve and modify specifics related to this job. For more information, see [API Reference](#).

- **Job status:** Current status of the job:
  - **Queued:** Job has been queued for execution.
  - **Running:** Job is in progress.
  - **Completed:** Job has successfully executed.

**NOTE:** Invalid steps in a recipe are skipped, and it's still possible for the job to be executed successfully.

- **Failed:** Job failed to complete.

**NOTE:** You can re-run a failed job from the Transformer page. If you have since modified the recipe, those changes are applied during the second run. See [Transformer Page](#).

- **Canceled:** Job was canceled by the user.
- **Flow:** Name of the flow from which the job was executed. Click the link to open the flow. See [Flow View Page](#).
- **Output:** Name of the output object that was used to define the generated results. Click the link to open the output. See [Flow View Page](#).

#### Execution summary:

- **Job type:** The method by which the job was executed:
  - **Manual** - Job was executed through the application interface.
  - **Scheduled** - Job was executed according to a predefined schedule. See [Add Schedule Dialog](#).
- **User:** The user who launched the job
- **Environment:** Where applicable, the running environment where the job was executed is displayed.
- **Start time:** Timestamp for when processing began on the job. This value may not correspond to when the job was queued for execution.
- **Finish time:** Timestamp for when processing ended on the job, successful or not
- **Last update:** Timestamp for when the job was last updated
- **Duration:** Elapsed time of job execution

#### Optimization summary:

For jobs sourced from relational datasets, you can optionally enable SQL-based optimizations, which apply some of the steps specified in your recipe back in the datasource, where they can be executed before the data is transferred to the running environment for execution. Using these optimizations means faster performance based on a lower volume of data transfer.

- Workspace administrators must enable the optimization feature for the workspace. For more information, see [Workspace Settings Page](#).
- When the feature is enabled, optimizations must be enabled for each flow. You can also select the optimizations to apply. For more information, see [Flow Optimization Settings Dialog](#).

When optimizations have been applied to your flow, they are listed on the Overview tab:

- **Optimization:** This setting is displayed if flow optimizations have been enabled for this flow.
- **Columns pruned:** If one or more unused columns have been pruned in the datasource via SQL, the count of columns is listed here.
- **Filters pushed down:** If one or more row filters has been applied in the datasource via SQL, the count of filters is listed here.

If an optimization is disabled or was not applied to the job run, it is not listed.

## Output Destinations Tab

If the job has successfully completed, you can review the set of generated outputs and export results.

test\_flow > REF\_PROD - 3

Job 60

Finished Today at 4:16 PM

Publish results

...

Overview

Output destinations

Profile

Dependency graph

Data sources

You can download the generated results locally or [publish](#) to another storage location.

| Name             | Location                                                                                                                                                                    | Status             |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------|
| REF_PROD _ 3.csv | <a href="hdfs://hadoop:8020/trifacta/queryResults/admin@trifacta.local/REF_PROD _ 3.csv">hdfs://hadoop:8020/trifacta/queryResults/admin@trifacta.local/REF_PROD _ 3.csv</a> | Completed • 13 sec |

**Figure: Output Destinations tab**

### Actions:

For each output, you can do the following:

- **View details:** View details about the generated output in the side bar.

**Tip:** The View details panel contains breakdowns for each phase of a job. If the job fails, you can review error messages, which correspond to entries in the Data Service log file.

- **Download result:** Download the generated output to your local desktop.

**NOTE:** Some file formats may not be downloadable to your desktop. See below.

- **Create imported dataset:** Use the generated output to create a new imported dataset for use in your flows. See below.

**NOTE:** This option is not available for all file formats.

### Direct file download

Click one of the provided links to download the file through your browser to your local desktop.

**NOTE:** If these options are not available, data download may have been disabled by an administrator.

**HYPER:** You can download HYPER formatted outputs to your desktop.

If you have generated output in a Tableau format and have configured a connection to Tableau Server, you can publish directly to the server. See *Publishing Dialog* .

### Create imported dataset

Optionally, you can turn your generated results into new datasets for immediate use in Designer Cloud powered by Trifacta Enterprise Edition. For the generated output, select **Create imported dataset** from its context menu.

**NOTE:** If you generated results in Parquet format only, you cannot create a dataset from it, even if the Create button is present. This is a known issue.

**NOTE:** When you create a new dataset from your job results, the file or files that were written to the designated output location are used as the source. Depending on your backend datastore permissions are configured, this location may not be accessible to other users.

After the new output has been written, you can create new recipes from it. See *Build Sequence of Datasets*.

### Publish

If Designer Cloud powered by Trifacta Enterprise Edition is connected to an external storage system, you may publish your job results to it. Requirements:

- Your version of the product supports publishing.
- Your connection to the storage system includes write permissions.
- Your results are generated in a format that the target system supports for writing.
- All sub-jobs, including profiling, successfully completed.

For more information, see *Publishing Dialog*.

### SQL scripts Tab

If the output for your job included one or more pre- or post-job SQL script executions, you can review the status of their execution during the job.

**NOTE:** If a SQL script fails to execute, all downstream phases of the job fail to execute.

**Tip:** If the SQL script execution for this job encountered errors, you can review those errors through this tab. For more detailed information, click **Download logs**.

2013 POS > Untitled recipe

Job 19

Finished Today at 5:48 PM

Publish results

...

Overview

Output destinations

SQL scripts

Dependency graph

Data sources

| Connection | SQL statement                                                                                                                                                                                                                     | Settings               | Status             |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------|--------------------|
| postgres   | CREATE TABLE IF NOT EXISTS "public"."spotable" ( timestamp date, jobType varchar(255), jobStatus varchar(255) ); INSERT INTO "p<br>ublic"."spotable"(timestamp, jobType, jobStatus) VALUES ('2021-06-22','Trifacta','started');   | Run before data ingest | Completed • 3 sec  |
| postgres   | CREATE TABLE IF NOT EXISTS "public"."spotable" ( timestamp date, jobType varchar(255), jobStatus varchar(255) ); INSERT INTO "p<br>ublic"."spotable"(timestamp, jobType, jobStatus) VALUES ('2021-06-22','Trifacta','completed'); | Run after data publish | Completed • <1 sec |

Figure: SQL scripts tab

Columns:

- **Connection:** Name of the connection through which the script was executed.
- **SQL statement:** The first part of the SQL script that was executed.
- **Settings:**
  - Run before data ingest - script was executed pre-job.
  - Run after data publish - script was executed post-job, after the job results had been written.
- **Status:** Current status and execution duration of the SQL script.

**NOTE:** If you have multiple SQL scripts for each settings, they may execute in parallel. For example, if you created three pre-job SQL scripts, there is no guarantee that they executed in the order in which they are listed.

View details:

Hover over a SQL script entry and click **View details**.

In the SQL script details window, you can review:

- Connection and SQL of the executed script.
- Any error messages that occurred during execution.

**Tip:** To review log information for any error messages, click **Download logs**.

For more information on these types of SQL scripts, see *Create Output SQL Scripts*.

Profile Tab

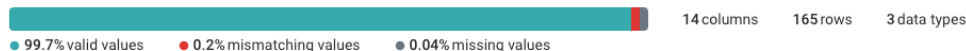
Review the visual profile of your generated results in the Profile tab. Visual profiling can assist in identifying issues in your dataset that require further attention, including outlier values.

**NOTE:** This tab appears only if you selected to profile results in your job definition. See *Run Job Page*.

All data

Download as PDF

Download as JSON



Results profile by column

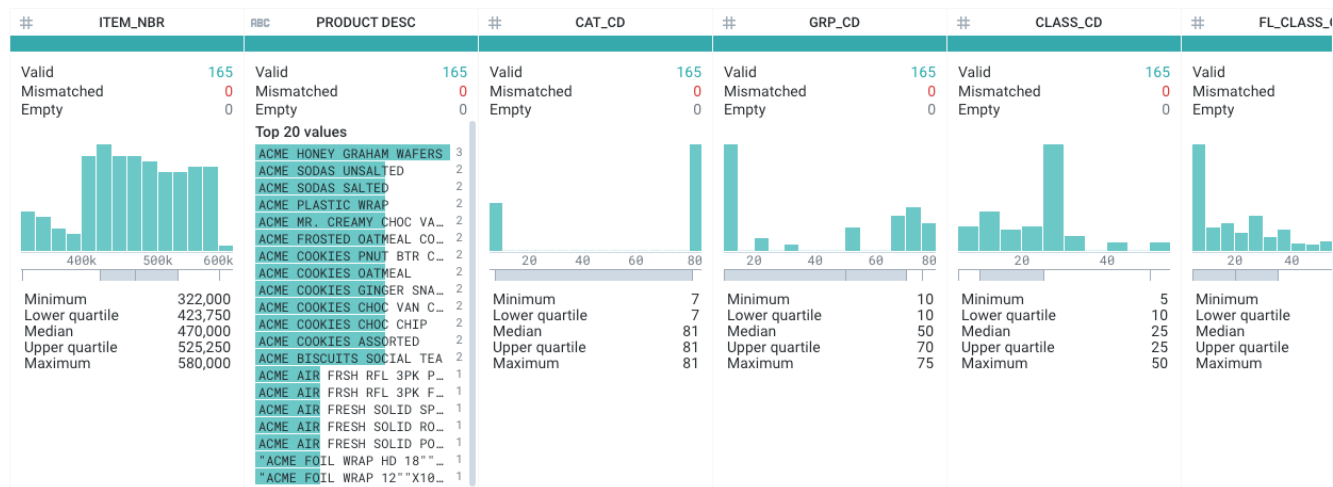


Figure: Profile tab

- **Download as PDF:** Download your visual profile and results of your data quality rules on the entire dataset as a PDF file. For more information, see *Overview of Data Quality*.
- **Download as JSON:** Download your visual profile as a JSON file.

In particular, you should pay attention to the mismatched values and missing values counts, which identify the approximate percentage of affected values across the entire dataset. For more information, see *Overview of Visual Profiling*.

**NOTE:** The computational cost of generating exact visual profiling measurements on large datasets in interactive visual profiles severely impacts performance. As a result, visual profiles across an entire dataset represent statistically significant approximations.

**NOTE:** Designer Cloud powered by Trifacta Enterprise Edition treats null values as missing values. Imported values that are null are generated as missing values in job results (represented in the gray bar). See *Manage Null Values*.

**Tip:** Mouse over the color bars to see counts of values in the category.

**Tip:** Use the horizontal scroll bar to see profiles of all columns in wide datasets.

In the lower section, you can explore details of the transformations of individual columns. Use this area to explore mismatched or missing data elements in individual columns.

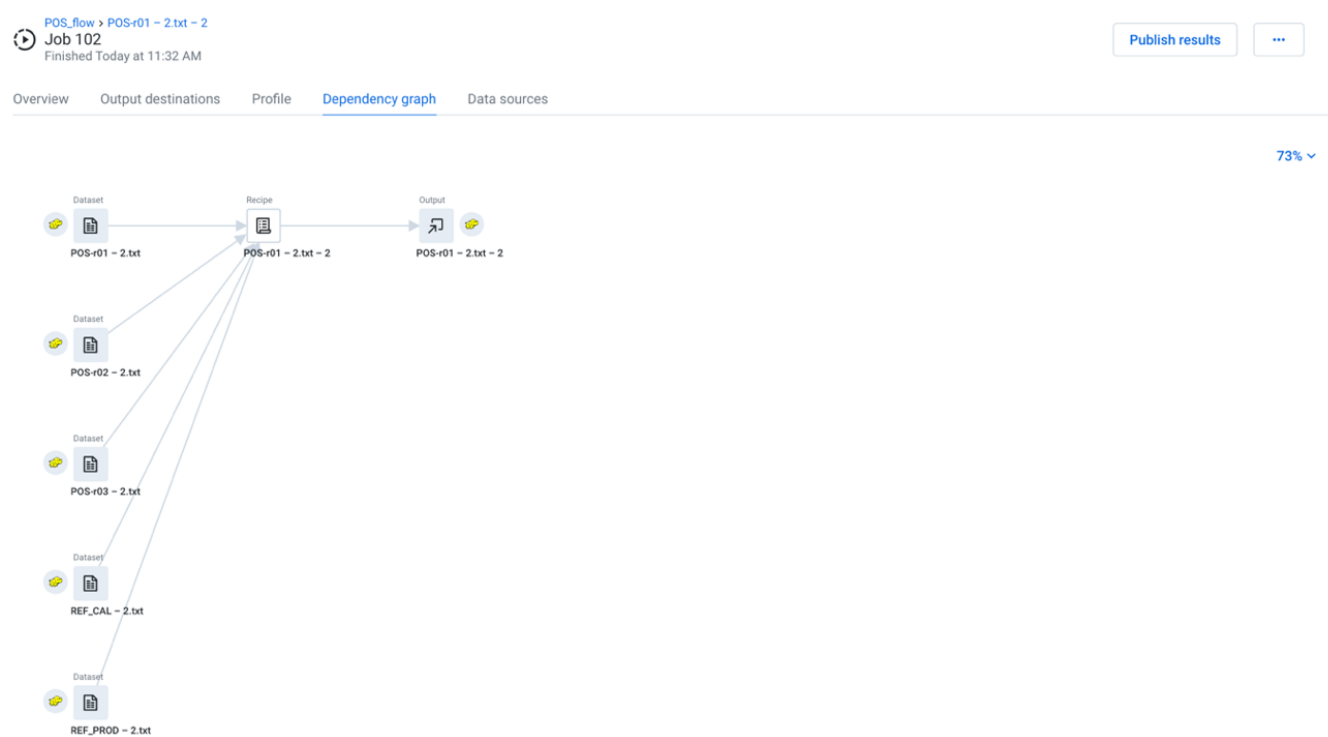
Depending on the data type of the column, varying information is displayed. For more information, see *Column Statistics Reference*.

**Tip:** You should review the type information for each column, which is indicated by the icon to the left of the column.

### Dependency graph Tab

In this tab, you can review a simplified representation of the flow from which the job was executed. This flow view displays only the recipes and datasets that contributed to the generated results.

**Tip:** To open the full flow, you can click its name in the upper-left corner.



**Figure: Dependency graph tab**

#### Zoom menu:

You can zoom the dependency graph canvas to display areas of interest in the flow graph.

The zoom control options are available at the top-right corner of the dependency graph canvas. The following are the available zoom options:

**Tip:** You can use the keyboard shortcuts listed in the zoom options menu to make quick adjustments to the zoom level.

- **Zoom in:** Zoom in 10% on the canvas to focus on greater detail.
- **Zoom out:** Zoom out 10% from the canvas to see more of it.
- **Zoom to fit:** Change the zoom level to fit all of the objects of your flow onto the screen.
- **25%, 50%, or 100%:** Change the zoom level to one of the preset levels.

#### Recipe actions:

- **Download recipe:** Download the text of the recipe in Wrangle .
- **Display Wrangle /natural language:** Toggle display of the recipe in raw language or in readable language.

#### Limitations:

- You can select only recipes in the flow graph.
- Context controls and menus are not available.

## Data sources Tab

In the Data sources tab, you can review all of the sources of data for the executing recipe.

2013 POS > POS-r01  
Job 10  
Finished Today at 5:52 PM

Publish results ...

Overview
Output Destinations
Profile
Rules
Dependency graph
Data sources

| Name         | Location                                                                                 | Status |
|--------------|------------------------------------------------------------------------------------------|--------|
| REF_PROD.txt | hdfs://hadoop:50070/trifacta/uploads/1/0a355cf3-b475-4367-a777-0d841fcab16a/REF_PROD.txt | -      |
| POS-r02.txt  | hdfs://hadoop:50070/trifacta/uploads/1/0a355cf3-b475-4367-a777-0d841fcab16a/POS-r02.txt  | -      |
| REF_CAL.txt  | hdfs://hadoop:50070/trifacta/uploads/1/0a355cf3-b475-4367-a777-0d841fcab16a/REF_CAL.txt  | -      |
| POS-r01.txt  | hdfs://hadoop:50070/trifacta/uploads/1/0a355cf3-b475-4367-a777-0d841fcab16a/POS-r01.txt  | -      |
| POS-r03.txt  | hdfs://hadoop:50070/trifacta/uploads/1/0a355cf3-b475-4367-a777-0d841fcab16a/POS-r03.txt  | -      |

**Figure: Data sources tab**

**NOTE:** If a flow is unshared with you, you cannot see or access the datasources for any jobs that you have already run on the flow, including any PDF profiles that you generated. You can still access the job results. This is a known issue.

## Parameters Tab

If your flow references parameters, you can review the state of the parameters at the time of job execution.

**NOTE:** This tab appears only if the job is sourced from a flow that references parameters. For more information, see *Overview of Parameterization*.



2013 POS - params > Dataset with Parameters - 2

Job 78

Finished Today at 4:22 PM

Publish results

...

Overview

Output Destinations

Profile

Dependency graph

Data sources

Parameters

Parameters

| Type      | Source                      | Resolved value | Name |
|-----------|-----------------------------|----------------|------|
| ✱ Pattern | Dataset with Parameters - 2 | {digit}{digit} |      |

Figure: Parameters tab

Webhooks Tab

2013 POS > POS-r01

Job 47

Finished Today at 4:35 PM

Publish results

...

Overview

Output Destinations

Profile

Dependency graph

Data sources

Webhooks

| Name    | URL                                                          | Status      | Response | Delivered        |
|---------|--------------------------------------------------------------|-------------|----------|------------------|
| Flow-22 | https://hooks.slack.com/services/T024K63KC/BN5SJNJKD/2RAeFCL | ✔ Completed | 200      | Today at 4:35 PM |

Figure: Webhooks Tab

When a webhook task has been triggered for this job, you can review the status of its delivery to the target system.

- Webhooks are defined on a per-flow basis. For more information, see *Create Flow Webhook Task*.

**NOTE:** Webhook notifications may need to be enabled in your environment. See *Workspace Settings Page*.

Columns:

- Name:** Display name for the webhook task.
- URL:** Target URL where the webhook notification is delivered.
- Status:** HTTP status code returned from the delivery of the message.
  - 200 - message was delivered successfully.
- Delivered:** Timestamp for when the webhook was delivered.

# Publishing Dialog

## Contents:

- *Limitations*
- *Publish to Tableau Server*

When a job has successfully completed, you can publish your job results to one of your connected datastores. In the Job Details page, click the Output Destinations tab. Then, click **Publish**.

## Limitations

- You cannot publish ad-hoc results for a job when another publishing job is in progress for the same job through the application. Please wait until the previous job has been published before retrying to publish the failing job. This is a known issue.
- If you run a job and then attempt to export the results to a relational source, Datetime columns are written in the relational table as String values. Direct publication of Datetime columns publishes the output in the designated target data type. For more information, see *Type Conversions*.
- If you run a job with a single relational target and it fails at the publication step, you cannot publish the transformation job through the Export Results window.

JSON-formatted files that are generated by Designer Cloud powered by Trifacta Enterprise Edition are rendered in JSON Lines format, which is a single line per-record variant of JSON. For more information, see <http://jsonlines.org>.

Publish

Hive

Database

default

Table

DatasetsARRAYINDEXOF1\_trifacta

Data Option

Create new table & load data

Publish

## Publish to Tableau Server

If you have created a Tableau Server connection, you can export results that have been generated to the connected server.

### Supported Formats:

- **Hyper:** Results are written to your Tableau Server in Hyper format.

### Options:

- **Connection:** If you have created multiple connections to Tableau Server, please select the connection to use from the list.
  - The Site name is specified as part of the connection. See *Tableau Server Connections*.
- **Project Name:** Name of the Tableau Server project.
- **Datasource Name:** Name of the Tableau Server datasource. This value is displayed for selection in Tableau Server.

### Data Option:

If you are publishing to a pre-existing table, schema validation is automatically performed.

- **Create new datasource:** The platform creates the datasource and then loads it with the results from this job. If you attempt to use this option on a source that already exists, the publishing job fails, and an error is generated in the log.
- **Append data to existing datasource:** The results from this job are appended to the data that is already stored in Tableau Server. If you attempt to append to a source that does not exist, the publishing job fails, and an error is generated in the log. Append operations also fail if you publish to a target with a different schema.
- **Replace contents of existing datasource:** Target datasource is dropped. A new datasource is created using the schema of the generated output and filled with the job results.

### Troubleshooting - Request timeout exception

When publishing to Tableau Server, you may encounter an error similar to the following for a PUT operation in the job log:

```
com.trifacta.clients.http.exceptions.RequestTimeoutException: PUT request to ...
```

### Solution:

In this case, the size of individual chunks submitted to Tableau Server is too large. The PUT operation did not complete before a server timeout was encountered, and the operation failed.

To address this issue, you should lower the size of each chunk that is submitted to Tableau Server for publication. For more information, see *Configure Data Service*.

# Plan Runs Page

In the Plan Runs page, you can track the status of all runs of your plans.

Plan runs are executed from the Plans page. See *Plans Page*.

Plan runs

Search plan runs

/

All runsCompletedFailedCanceledRunning

|                                                                                                        | Status                   | User        | Started                                  |
|--------------------------------------------------------------------------------------------------------|--------------------------|-------------|------------------------------------------|
| <div><div>▼</div><div><div>p1</div><div>Run ID: 5</div></div><div>Run parameters_flow</div></div>      | <div>✔ Completed</div>   | Admin (you) | Today at 2:13 PM<br>Ran for 2 minutes    |
| <div><div>▶</div><div><div>customer - 2</div><div>Job ID: 107</div></div><div>Run POS_flow</div></div> | <div>✔ ✔ Completed</div> |             | Today at 2:13 PM<br>Ran for a minute     |
| <div><div>▶</div><div><div>POS-r01 - 2.txt - 2</div><div>Job ID: 108</div></div></div>                 | <div>✔ ✔ Completed</div> |             | Today at 2:14 PM<br>Ran for a minute     |
| <div><div>▶</div><div><div>POS-r01 - 2.txt - 2</div><div>Job ID: 108</div></div></div>                 | <div>✔ ✔ Completed</div> |             | Today at 2:14 PM<br>Ran for a minute     |
| <div><div>&gt;</div><div><div>new_plan</div><div>Run ID: 4</div></div></div>                           | <div>✔ Completed</div>   | Admin (you) | Today at 1:39 PM<br>Ran for 2 minutes    |
| <div><div>&gt;</div><div><div>new_plan</div><div>Run ID: 3</div></div></div>                           | <div>✔ Completed</div>   | Admin (you) | Today at 12:57 PM<br>Ran for 2 minutes   |
| <div><div>&gt;</div><div><div>new_plan</div><div>Run ID: 2</div></div></div>                           | <div>✔ Completed</div>   | Admin (you) | Yesterday at 4:04 PM<br>Ran for a minute |

Figure: Plan Runs page

Tabs:

Each of the available tabs corresponds to a possible status for plan runs that have been initiated on the platform.

- **All runs:** All runs that you have initiated are listed here.
- **Completed:** Run has successfully executed.
- **Failed:** Run failed to complete.

**NOTE:** You can re-run a failed run from the Plans page. See *Plans Page*.

To download the logs of a failed plan run, select **Download logs** from the run's context menu.

- **Canceled:** Run was canceled by the user.
- **Running:** Run is in progress.

Access:

- You can review and drill into any run that you initiated.
  - For each plan run, you can click the caret to reveal details on the individual flow tasks that were part of the plan run.

- Logs are available for individual flow tasks. Click **Download logs** from the flow task context menu.
- You can also drill into any run that was initiated from a flow that has been shared with you.
- Administrators can review read-only listings for runs created by other users.

#### Columns:

- **Run:** Internal identifier for the plan run. This value is unique for all runs in your Trifacta® instance.
  - Click the ID number to explore details about the plan run. See *Plan View Page*.
- **Status:** The current status of the run. See above.
- **User:** The Trifacta user that initiated the run.
- **Started:** Start timestamp for the run.

#### Actions:

- **Filter by status:** Click one of the tabs to filter the display to show only the listings for the selected status.
- **Filter by type and date:** Click the Funnel icon to filter the list of plans by source of execution, date range, or both. See below.
- **Search:** Enter text in the search field to filter the listed runs by run ID, flow name, or recipe name.

### Filter Plan Runs

To filter the list of plans, click the Funnel icon next to the search bar. You can use the following dialog to filter the display of plans based on dates or source of execution:

Filter plan runs
×

Type

Show all plan runs
▼

Date/Time

Started

Started After
▼

MM/DD/YYYY

📅

HH:MM

AM
▼

Ended

Ended Before
▼

MM/DD/YYYY

📅

HH:MM

AM
▼

Clear Filters

Cancel

Apply

**Figure: Filter plan runs dialog**

**Plan type:**

Show plans based on the following available options:

- **Show all plan runs**
- **Only show manual plan runs**
- **Only show scheduled plan runs**
- **Only show api plan runs**

**Started:**

- Specify the date and time when the plan runs to display started.
- Select **Started Between** or **Started After** from the drop-down list and populate both date-time rows.

**Ended:**

- Specify the date and time when the plan runs to display ended.
- Select **Ended Between** or **Ended After** from the drop-down list and populate both date-time rows.

**Actions:**

- To apply the specified time filter to the Plan Runs page, click **Apply**.
- To clear the time period values, click **Clear Filters**.

# Sample Jobs Page

In the Sample Jobs page, you can track the status of all sample jobs to which you have access.

Sample jobs

Search samples

/

All jobsCompletedFailedCanceledRunningQueued

| Job                               | Status      | Flow                   | User                | Started ▾                                 |
|-----------------------------------|-------------|------------------------|---------------------|-------------------------------------------|
| <b>Anomaly-based</b><br>Job ID: 4 | In progress | <b>test</b><br>POS-r02 | Administrator (you) | Today at 4:27 <a href="#">Cancel job</a>  |
| <b>Filter-based</b><br>Job ID: 2  | Completed   | <b>test</b><br>POS-r03 | Administrator (you) | Today at 4:25 PM<br>Ran for a few seconds |
| <b>Random</b><br>Job ID: 1        | Completed   | <b>test</b><br>POS-r01 | Administrator (you) | Today at 4:24 PM<br>Ran for a few seconds |

Figure: Sample jobs page

Tabs:

Each tab corresponds to a possible status for Sample jobs that have been initiated in the Designer Cloud application .

**Tip:** Select **Download logs** from the sample job's context menu to download the log file for that sample job.

- **All jobs:** All sample jobs that have started.
- **Completed:** Sample jobs that have successfully completed.
- **Failed:** Sample jobs that have failed.
- **Canceled:** Sample jobs that were canceled.

**Tip:** To cancel an in-progress sample job, select the **Cancel job** option while the sample job is running.

- **Running:** Sample jobs that are currently in progress.

Access:

- You can review and drill into any available sample jobs.
  - For each sample job, you can click the caret to reveal details on the sample jobs.

Columns:

- **Jobs:** Sample jobs to which you have access.

**Tip:** Click the link for the sample type to load the sample in the Transformer page.

- **Rows:** Number of rows that have been used to generate a sample.
- **Status:** The current status of the sample jobs.
- **Flow:** Name of the flow in which the samples have been collected. You can click the flow link to go the Flow View Page.
- **User:** The Trifacta user who initiated the sample job.

- **Started:** Start timestamp for the sample job.

#### Actions:

- **Filter by status:** Click one of the tabs to filter the display to show only the listings for the selected status.
- **Filter by type and date:** Click the Funnel icon to filter the list of sample jobs by date range, time of execution, or both. For more information, see below.
- **Search:** Enter text in the search field to filter the listed sample jobs by job ID, sample type, status, or flow.

### Filter Sample Jobs

To filter the list of sample jobs, click the Funnel icon next to the search bar. Use the dialog to filter the display of samples based on date ranges:

Filter sample jobs
×

**Date/Time**

**Started**

Started After
▼

MM/DD/YYYY

HH:MM

AM ▼

**Ended**

Ended Before
▼

MM/DD/YYYY

HH:MM

AM ▼

Clear Filters

Cancel

Apply

**Figure: Sample Job dialog**

#### Started:

- Specify the date and time when the sample jobs have started.
- Select **Started Between** or **Started After** from the drop-down list and populate both date-time rows.

#### Ended:

- Specify the date and time when the sample jobs have ended.
- Select **Ended Between** or **Ended Before** from the drop-down list and populate both date-time rows.

#### Actions:

- To clear the time period values, click **Clear Filters**.



# Transformer Page

## Contents:

- Page Uses
  - Identify and Select Data
  - Get Statistics
- Context Panel
  - Build Recipes
  - Generate Samples
- Launch Jobs
- Transformer Bar

In the Transformer page, you identify the data that you need to transform and build your transformation recipes on samples taken from your currently selected dataset.

When you make changes to your transformation recipe, those changes are immediately applied to your sample, so that you can preview the results of your recipe before you run it against the entire dataset. In this manner, you can quickly build and iterate on the transformations applied to your data.

- In the Library page, click the name of the dataset. See *Library Page*.
- By default, Designer Cloud powered by Trifacta® Enterprise Edition selects the first N of row data as the **he ad sample**. The number of rows depends on the number of columns, data density, and other factors. Depending on the size, this sample may be the full dataset.
- For more information, see *Overview of Sampling*.

The screenshot displays the Transformer Page interface. At the top, there's a header bar with 'USDA FARMERS MARKET 2014 FLOW' and a 'Full Data' button. Below this is a toolbar with various icons for data manipulation. The main area is divided into two panes: 'Source' and 'Preview'. The 'Source' pane shows a table with columns: #, FMID, MarketName, and MarketName. The 'Preview' pane shows a similar table with the same columns. To the right of the main panes is a 'Suggestions' panel with a search bar and several sections: 'Replace', 'Split on values matching', 'Extract values matching', 'Count values matching', and 'Extract list of values'. Each section contains a list of suggestions with a 'See all' link.

Figure: Transformer Page

## Page Uses

**Tip:** When keyboard shortcuts are enabled, press ? in the application to see the available shortcuts. Individual users must enable them. See *User Profile Page*.

### Identify and Select Data

In the main panel of the Transformer page, you can select one or more elements of sampled data, which prompts suggestions for steps that you can apply to transform them. Each of these views provides a different perspective on your data, and the results of any subsequent steps that you select or configure are previewed by default in the data grid:

**NOTE:** Before your job is run, profiling information such as column statistics are exact counts of the sample that is currently loaded. After the job is run, profiled results in the Job Results page might include estimates for some metrics and counts, depending on the scale of the dataset.

**Tip:** Click one or more column headings to be prompted for suggestions that apply to the selected column or columns.

| Panel                      | Description                                                                                                                                                                                                  | Recommended Uses                                                                                                                                              |
|----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Transformer Toolbar</b> | A toolbar of common transformations, filters, and other operations.                                                                                                                                          | Use the tools in these drop-downs to quickly build common recipe steps.                                                                                       |
| <b>Data Grid</b>           | By default, the Transformer page displays previews in a columnar grid, which is the default view. Click <b>Grid</b> .                                                                                        | Use for examining values in a column with appropriate surrounding context. How do missing values in one column compare to values in another column?           |
| <b>Column Details</b>      | For additional statistical information on individual columns, select <b>Column Details</b> from the drop-down next to the column title.                                                                      | Explore values in an individual column, when their context in other rows is not necessary. Useful for managing outliers, reviewing mean, min, and max values. |
| <b>Column Browser</b>      | Use the Column Browser to select the columns to display and review data across columns. Click <b>Columns</b> .                                                                                               | Navigate between columns and toggle their display in the data grid. Good for high-level perspective. Use histograms for selection of ranges of values.        |
| <b>Context Panel</b>       | Depending on the state or the current selection of the data grid, the right side of the page displays one of several contextual panels. These panels cover recipes, suggestions, steps, and more. See below. | Review recipe and edit, create, or delete recipe steps. Review and create samples.                                                                            |

### Get Statistics

You can use the following methods for acquiring statistics on your dataset sample or individual columns in your sample:

- **Sample bar:** At the top of the data grid, you can see the name of the sample currently displayed in the grid. For smaller datasets, this sample is the entire dataset.
- **Status bar:** At the bottom of the page, you can review the number of data types and rows and column information for the sample currently displayed in the data grid. These metrics are updated based on the recipe steps that you apply to the sample.
  - Click the Eye icon to toggle display of individual columns. See *Visible Columns Panel*.
- **Column statistics:** You can review basic statistics on individual columns.
  - Select a column in the data grid. Column information is displayed in the context panel.

- You can also click the Columns icon at the top of the data grid to select your column to review detailed statistics. See *Column Browser Panel*.
- **Profile your data:** When you run a job on your dataset, you can optionally generate a visual profile of the resulting output. A visual profile can be useful for identifying key metrics on individual columns. See *Job Details Page*.
- **Computed statistical functions:** As needed, you can generate aggregated statistics as part of your recipe. See *Aggregate Functions*.

## Context Panel

The following actions are applied through the context panel on the right side of the screen. See *Context Panel*.

## Build Recipes

Use the following methods to add or modify recipe steps in the Transformer page:

**Tip:** To add a new recipe step, press CTRL/COMMAND + K. Enter a search string for your transformation step.

- **Suggestion Cards:** When you select data in the Transformer page, a set of suggested transformations is displayed in cards. Select the appropriate one to preview the results in the data grid. Then, add or edit the selected transformation. See *Selection Details Panel*.
- **Transformer Toolbar:** Select data in the data grid or column browser and then choose your transformation from the Transformer toolbar. The Transform Builder is pre-populated with your transformation. See *Transformer Toolbar*.
- **Search panel and Transform Builder:** Click the + icon in the Transformer page and use the Search panel to locate your preferred transformation. See *Search Panel*.
  - Complete the transform definition in the Transform Builder. See *Transform Builder*.
- **Recipe Panel:** After recipe steps have been created, you can review and edit them through the Recipe panel. See *Recipe Panel*.
- **Transform Preview:** Before a transform step has been added to the recipe, a preview of the transform is displayed in the data grid. See *Transform Preview*.

## Generate Samples

For larger datasets, the Transformer page displays a sample of them, which you use as representative data to build your recipe. As needed, you can generate a new sample, which is useful for polishing your recipe.

**The data that is displayed in the data grid is based on all of the upstream samples after which all subsequent steps in each upstream recipe are performed in the browser. If you have a large number of steps or complex steps between the recipe locations for your samples in use and your current recipe location, you may experience performance slow-downs or crashes in the data grid. For more information on sampling best practices, see <https://community.trifacta.com/s/article/Best-Practices-Managing-Samples-in-Complex-Flows>.**

For more information, see *Samples Panel*.

## Launch Jobs

**Run jobs:** To run a job that executes the transform recipe currently in the Transformer page across the entire dataset, click **Run**. See *Run Job Page*.

## Transformer Bar

The Transformer page contains menus that are different from the standard Trifacta menu bar.



**Figure: Transformer page toolbar**

- **Flow name:** Click to review flow details. See *Flow View Page*.
- **Dataset menu:** Click the caret next to the flow name to open.
  - Review the datasets in the flow or open a different wrangled one.
  - See a mini-map of flow view for the flow.
  - See *Recipe Navigator*.
- **Samples:** Click the description of the current sample to review and create new samples from your dataset. See *Samples Panel*.
- **Search icon:** Search for transformations to add to your recipe. See *Search Panel*.
- **Recipe icon:** Display the current recipe. See *Recipe Panel*.
- **Flow Parameters icon:** Review, create, and modify the parameters of your flow. See *Manage Parameters Dialog*.
- **Samples icon:** Click the dropper icon to review and create new samples. See *Samples Panel*.
- **Run:** Runs the currently specified recipe on the dataset. See *Run Job Page*.

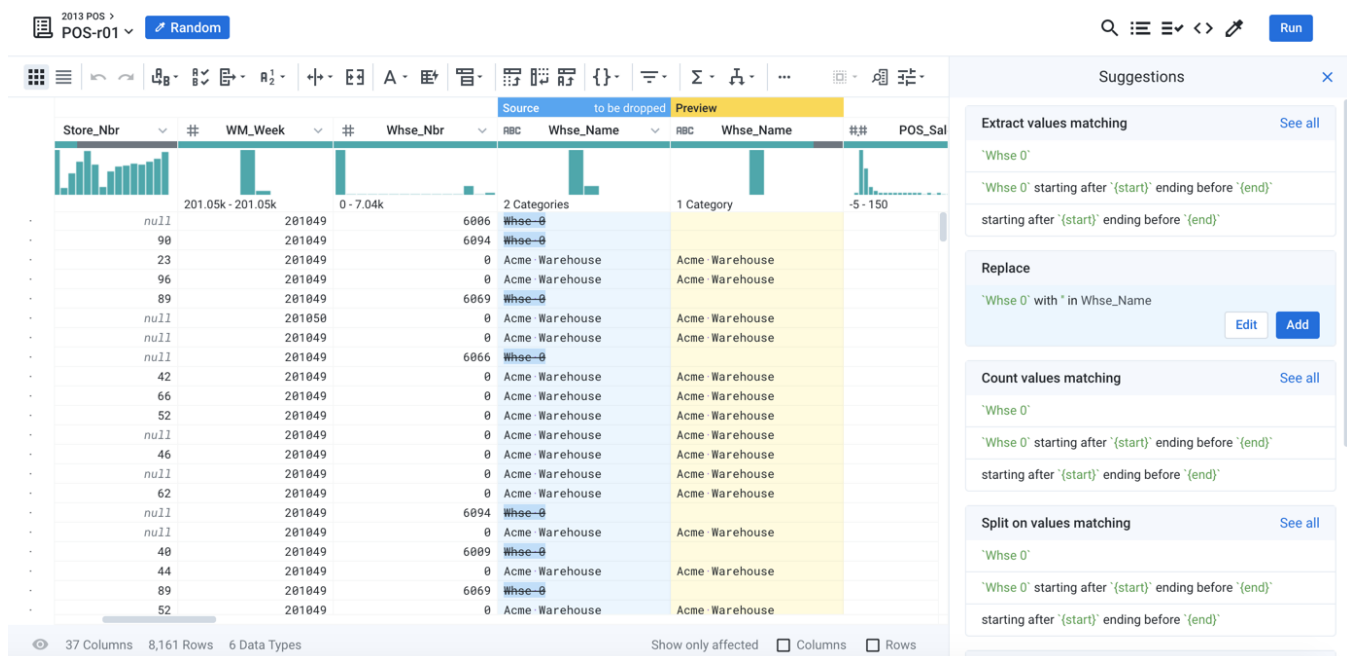
# Data Grid Panel

## Contents:

- *Transformer Toolbar*
- *Status Bar*
- *Find Column*
- *Column Information*
  - *Selecting columns*
  - *Selecting values*
  - *Row Information*
- *Filter Data Grid*
- *Transformation Preview*
- *Target Matching Bar*

The data grid in the Transformer Page displays how your current recipe applies to the data in your currently selected sample.

- The grid is the default view in the Transformer page of Designer Cloud powered by Trifacta® Enterprise Edition.
- To open the data grid, click the Grid View icon in the Transformer bar at the top of the page.



**Figure: Data Grid Panel**

## Select:

- Click column headings to review a visual profile of the column's data and a set of suggestions for transformations to apply to the column.
  - These columns appear in the context panel on the right side of the screen.

**Tip:** Keep clicking columns. You can select multiple columns to prompt for another set of applicable suggestions.

- Suggestions are also generated when you select one or more values in the data histogram for a column or individual values in the displayed rows of the sample.
- See *Selection Details Panel*.
- Select specific values in a column for suggestions on those strings.

**NOTE:** Values in a cell cannot exceed 25,000 characters in length.

**Tip:** If you select a single value in the data grid, the suggestion cards suggest operations specific to that string. If you multi-select multiple values, the suggestions can apply any pattern shared between the values. For example, selecting ", CA" and ", NY" results in suggestions for how to handle state abbreviations in a column.

### Scroll:

- Use the vertical scroll bar to the right of the displayed rows of data to show other rows in the sample. To review rows of the sample data that are not displayed, you may click values in a column and then scroll down through the sampled data.
- Use horizontal scrolling to review additional columns that are off-screen.

**Tip:** If the contents of a cell are too large for the display, you can click the Caret ( > ) icon to the right of the cell value in the data grid to display the entire contents of the cell.

### Add or Edit:

- To add a selected suggestion card to your recipe, select the card. Then, click **Add**.
  - To modify a suggested recipe step, select its suggestion card and click **Edit**. See *Transform Builder*.
- To review details about an individual column, select **Column Details** from the column drop-down. See *Column Details Panel*.
- To review details about a selection of columns, click the Column View icon in the Transformer bar. See *Column Browser Panel*.

### Ordering:

You can reorder the rows based on the values in a column. From the Column menu, select **Edit Column > Sort**. For more information, see *Column Menus*.

**NOTE:** Transforms that use the `group` parameter can result in non-deterministic re-ordering in the data grid. However, you should apply the `group` parameter, particularly on larger datasets, or your job may run out of memory and fail. To enforce row ordering, you can use the `sort` transform. For more information, see *Sort Transform*.

## Transformer Toolbar

At the top of the data grid, you can use the toolbar to quickly build common transformations, filter the display, and other operations. See *Transformer Toolbar*.

## Status Bar

Below the data grid, you can review summary information about the data in your currently selected sample.



### Figure: Sample Status Bar

Click the Eye icon to open the Visible Columns panel, where you can toggle the display of individual columns. For more information, see *Visible Columns Panel*.

The status bar contains metrics about the current dataset sample for the currently selected recipe step.

- For example, if your first recipe step removes 100 rows of data, when you create your next recipe step, the status bar should indicate a row count that is 100 less than the row count at the start of the recipe. The other counts may be affected as well.
- The number of columns reflects the count that is currently displayed in the data grid. Toggling visibility of columns or applying column-based filters changes this value.

**NOTE:** Counts of data types may reflect that varying formats of Datetime columns are considered different types for this computation.

**Tip:** Before you begin transforming your data, you might want to verify the columns and count of data types against the data before it was imported. If there are discrepancies, you might want to investigate the differences. Unless your sample includes the entire dataset, row counts should differ.

**NOTE:** In the Trifacta Photon running environment, results can differ between executions of the same recipe due to its parallel execution and data limiting within the Transformer page. In particular, joins with multiple matches per key can sometimes cause a difference in the number of reported rows when the job is re-executed.

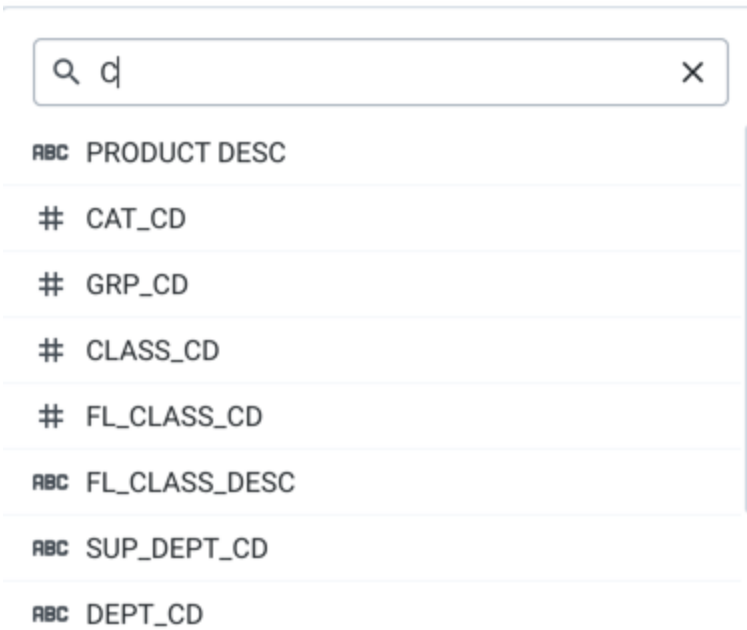
### Show only affected:

When transformation steps are previewed, you can use these checkboxes to display only the previewed changes for affected rows, columns, or both.

**Tip:** These options assist in narrowing the data grid display to only the steps affected by the current recipe step.

## Find Column

In a wide dataset, click the Find icon in the Transformer toolbar to locate the column of interest.

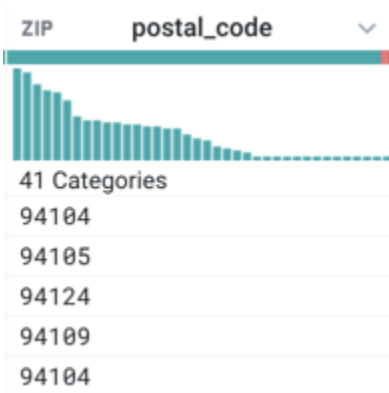


**Figure: Find column search bar**

- Use the up and down arrows to view the list of the columns in the dataset.
- You can start typing a column name to filter the list.

**NOTE:** An imported dataset requires about 15 rows to properly infer column data types and the row, if any, to use for column headers.

## Column Information



**Figure: Column header, data quality bar, and histogram**

- In the column header, counts reflect only the counts in the currently loaded sample. They do not reflect counts across the entire dataset, unless the entire dataset is the sample.
- There are some limitations on column names. For more information, see *Rename Columns*.

| Item      | Description                                                                                                                                              |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| Data type | Identifies the selected data type, which can be inferred by the application based on the contents of the column. Click the icon to change the data type. |



|                  |                                                                                                                                                                                                                                                                                                     |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                  | <p><b>Tip:</b> Before you start performing transformations on your data based on mismatched values, you should check the data type for these columns to ensure that they are correct. For more information, see <i>Supported Data Types</i>.</p> <p>See <i>Supported Data Types</i>.</p>            |
| Column name      | To change the column name, select <b>Rename...</b> from the column menu.                                                                                                                                                                                                                            |
| Column menu      | Depending on the column data type, you can select from a set of predefined recipe steps in the column menu under the caret on the right side of the menu. See <i>Column Menus</i> .                                                                                                                 |
| Data quality bar | <p>The horizontal line shows valid, missing, and mismatched values in the column compared to the column's data type.</p> <p><b>Tip:</b> You can click these colored bars to generate suggestion cards for transformations to act on these types of values.</p> <p>See <i>Data Quality Bars</i>.</p> |
| Column histogram | <p>For each column, you can see the range and frequency of values in the column.</p> <p><b>Tip:</b> You can select one or more values a histogram to generate suggestion cards.</p> <p>See <i>Column Histograms</i>.</p>                                                                            |

## Selecting columns

Through the Column Browser, you can use data quality bars and data type information to perform basic review of data across many columns. You can use these tools to select data of interest for display in the data grid or Column Details views or to prompt for suggestions of recipe steps.

## Selecting values

You can click and drag to select values in a column:

- Select a single value in the column to prompt a set of suggestions.
  - Select multiple values in a single column to receive a different set of suggestions.
  - See *Selection Details Panel*.
- Double-click to select an individual word, and triple-click to select an entire cell value.
- When you select values, some values in other columns may be highlighted in a darker color, which provides some indication of correlation between values.

## Row Information

On the left side of the screen, you can see a column of black dots. If you hover over one of these, you can see the current row number and, if the information is still available, the row number for the row from the original source data. These values apply only to the sample in the current dataset.

**Tip:** To review the original row number for a row, hover over the black dot in the data grid. These values can be referenced using the `$sourcerownumber` reference in your recipe steps. Some transformation steps, such as `pivot` and `union`, may make the original row information invalid or otherwise unavailable, which disables this option. See *Source Metadata References*.

## Filter Data Grid

From the Filters drop-down, you can define filters to apply to columns, rows, or both in the data grid. See *Filter Panel*.

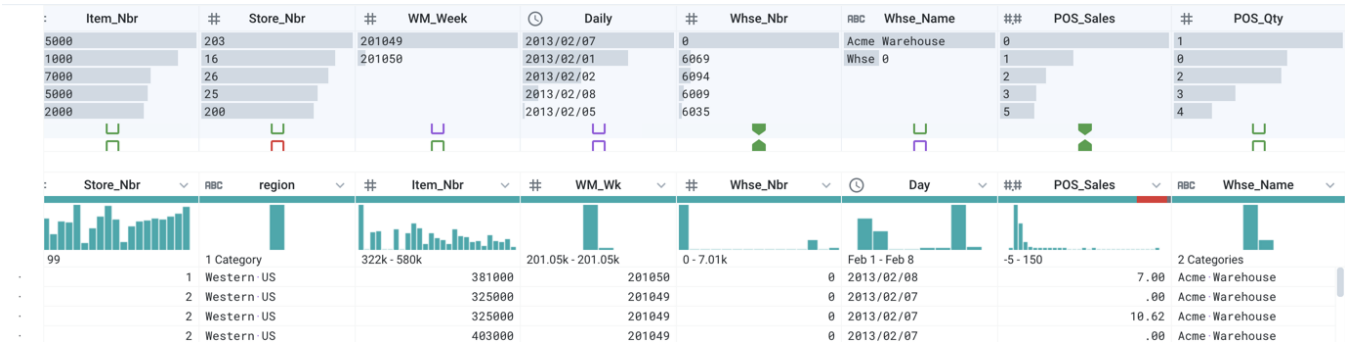
## Transformation Preview

Before a transformation in development has been added to the recipe, a preview of the results is generated in the data grid. See *Transform Preview*.

## Target Matching Bar

When a target has been assigned to your recipe, you can review the column names and data types that are expected for the target in the Target Matching bar above the column histograms.

- You can assign a dataset to be the target for the recipe you are constructing. This imported dataset, reference dataset, or recipe output contains the set of columns to which you are targeting your wrangling activities. When a target has been assigned, it is displayed in the data grid and column browser to assist you in defining your wrangling steps to match the target.
- For more information, see *Overview of Target Matching*.



**Figure: Target Matching Bar**

In the Target Matching bar, you can review how the target above matches the current recipe below. For each column, matching assesses:

- Current column name vs. target column name
- Current column data type vs. target column data type
- Current column position vs. target column position
- Current column values vs. target column values

**Tip:** Two solid green schema tags indicate a perfect match based on the above conditions.

### Actions:

- If you hover over the schema tags between a column and the target above it, you can review the detected differences between the target and the current column and select actions to fix any differences.
- Click the schema tag to auto-fix a mismatch or to select the column with which to match. These actions add a recipe step to create a match between the two columns.

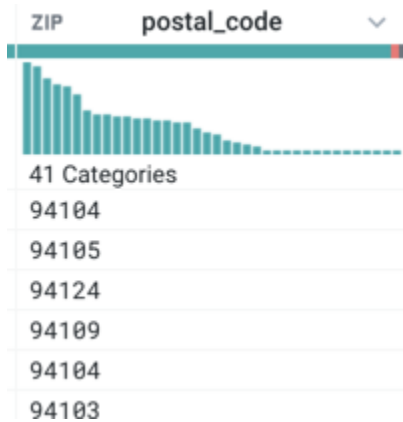
For more information on the schema tags, see *Column Browser Panel*.

# Column Histograms

The bar chart at the top of each column in the Transformer page, called a **histogram**, characterizes the data in that column. Each column histogram displays the count of each detected value in the column (for string data) or the count of values within a numeric range (for number data).

You can use this histogram to identify unusual values or outlier values, which should be removed or corrected.

**NOTE:** Counts in a column histogram reflect only the data in the sample in the data grid. Counts in the entire dataset may differ.



**Figure: Column Histogram**

**Tip:** When you resize the width of a column, the number of bars displayed in the column histogram changes accordingly. You can use this dynamic resizing to change the granularity displayed in histograms.

The contents of the column histogram vary depending on the data type for the column. For example:

- For numerical types (Integer or Decimal type), each bar covers a range of values, and the bars are sorted in numerical order.
  - For a numeric range bar that overlaps values in another bar, values are inclusive on the lower bound and exclusive on the upper bound. For example, if a histogram bar represents the values 0-10, it includes the count of instances of 0 and does not include the count of instances of 10. The count of instances of 10 is part of the adjacent bar in the histogram.
  - The above applies only when there are overlapping values between data ranges. If there are no overlapping values, then the range includes the values of the lower and upper boundaries.
- For non-numerical (i.e., “categorical”) types, each vertical bar covers a single value, ordered from most frequently-occurring values.

**Tip:** If you hover over a bar in the histogram, you can review specific values, the count of that value, and the percentage that value represents of the total count of values in the column.

When you select values:

- For the values represented by the bar(s) you selected, rows containing them are highlighted, and suggestion cards are presented for handling those values.
- Bars in other columns may partially change color. This feature, known as **brushing and linking**, illustrates the fraction of the bar values in other columns that correspond to your selected values. Brushing and linking is useful for identifying correlations in your data.

To select values:

- Use `CTRL` - click to select multiple discrete values.
- Click and drag across a range of values.

# Data Quality Bars

Just below the column name in the data grid is a horizontal band, which identifies data quality issues among the sample values in the column.



**Figure: Data quality bars**

Each color band identifies the relative number of records that fit the following data quality definitions:

| Color | Type       | Description                                                                                                                                                                                            |
|-------|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Green | Valid      | Valid value for the currently selected data type.                                                                                                                                                      |
| Red   | Mismatched | A value that does not match the listed data type. For example, if a column of Zip type contains, MISSING, it is considered a mismatched value.<br><br>For more information, see <i>Find Bad Data</i> . |
| Gray  | Missing    | Value is empty or null. For more information, see <i>Manage Null Values</i> .                                                                                                                          |

You can use a column's data quality bar to build a recipe step to address selected data. For example, click the red set of values in the data quality bar to generate a set of suggestion cards to address mismatched values in the column.

**Tip:** The histogram may also show you unwanted variation in your values. For example, if the column stores latitude data, the precision may be too fine (e.g. 37.764013 and 37.76022 versus 37.76). You can use recipe steps to round your data to a more usable level of precision and thereby reduce the number of unique values in the column to a more manageable count. See *ROUND Function*.

For more information, see *Supported Data Types*.

# Lookup Wizard

## Contents:

- *Lookup Wizard - Step 1*
  - *Lookup Wizard - Step 2*
  - *Column Cleanup*
  - *Auto-updating Lookups*
- 

Through the Transformer page, you can perform lookups from one set of values in your dataset into another set of values in another dataset using a simple wizard. To perform a lookup, select the caret next to a column title, and then select **Lookup....**

A **lookup** compares each value in the selected column against the values in a selected column of the target dataset. Where a match is found, the values in other columns of the target dataset are inserted as new columns in the dataset from which the lookup was executed.

For example, your enterprise is changing the names of all of your products. Instead of performing a complex set of replace transforms, you can perform a lookup from your productName column into a two-column dataset, which contains the original name and the new name in separate columns. When the new name is inserted into your source dataset via lookup, you can delete the source column and continue transforming your data with the new names.

- You cannot perform lookups on columns of Object or Array data type.
- A lookup essentially performs a left join between the first dataset and the second one. However, lookups are less flexible in terms of defining and editing them.








**NOTE:** If column values are non-unique, the resulting dataset can be significantly larger than the original dataset.

This workflow is best demonstrated by example. In this case, your raw sales data records product information in internal numeric identifiers. For analysis, you may want to integrate data from your products master data based on the internal identifier, so that you have a product description and other useful information as part of your dataset.

## Lookup Wizard - Step 1

In the first step, you select the dataset against which you would like to perform your lookup for matching data for the Item\_Nbr column. In this example, the products dataset is selected, since it contains the list of recognized products:

**Tip:** You can search your available flows and datasets. When you search for flows, all datasets in the flow are matched.

| Search...                                                                                                                                                                        | 1      | 2                 | All (29) | Imported (27) | Reference (2) | Recipe (0) |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------|-------------------|----------|---------------|---------------|------------|
| NAME                                                                                                                                                                             | SOURCE | LAST UPDATED      |          |               |               |            |
|  POS-r02.txt                                                                                    | HDFS   | Today at 10:17 AM |          |               |               |            |
|  POS-r03.txt                                                                                    | HDFS   | Today at 10:17 AM |          |               |               |            |
|   REF_PROD.txt | HDFS   | Today at 10:17 AM |          |               |               |            |
|  REF_CAL.txt                                                                                    | HDFS   | Today at 10:17 AM |          |               |               |            |
|  USDA Farmers Market 2014                                                                       | HDFS   | Today at 10:15 AM |          |               |               |            |
|  BOH August.csv                                                                                 | HDFS   | Today at 9:49 AM  |          |               |               |            |

Cancel
Select

**Figure: Lookup Wizard - Step 1**

## Lookup Wizard - Step 2

After you select the dataset against which to perform the lookup, you select the field in the target dataset to use as the lookup key. The **lookup key** provides the set of identifiers for which you are trying to find a match for each value in the source column. In this case, the lookup key column has the same name as the source column: `ITEM_NBR`.

Step 2 of 2 Select Lookup Key ✕

ITEM\_NBR ▼

< Back
Cancel
Execute Lookup

**Figure: Lookup Wizard - Step 2**

## Column Cleanup

When the lookup is executed, for each value in the source `item_nbr` column that can be found in the target dataset's `ITEM_NBR` column, all of the other columns in the corresponding row of the second dataset are inserted as separate columns in the first dataset. These columns are inserted to the immediate right of the column that was used for the lookup:

| #       | Store_Nbr | #           | Item_Nbr                         | ASC           | PRODUCT DESC | #      | CAT_CD | #       | GRP_CD | #      | CLASS_CD | #      | FL_CLASS_CD |
|---------|-----------|-------------|----------------------------------|---------------|--------------|--------|--------|---------|--------|--------|----------|--------|-------------|
| 1 - 250 |           | 322k - 580k |                                  | 70 Categories |              | 7 - 81 |        | 10 - 75 |        | 5 - 50 |          | 5 - 75 |             |
|         | 1         | 381000      | ACME LAWN GARDEN BAG CLEAR       | 07            | 75           | 05     |        |         |        |        |          |        |             |
|         | 2         | 325000      | ACME COOKIES CHOC CHIP           | 81            | 10           | 25     |        |         |        |        |          |        |             |
|         | 2         | 325000      | ACME COOKIES CHOC CHIP           | 81            | 10           | 25     |        |         |        |        |          |        |             |
|         | 2         | 403000      | ACME SANDWICH BAG                | 07            | 70           | 05     |        |         |        |        |          |        |             |
|         | 2         | 449000      | ACME SODAS SALTED                | 81            | 30           | 15     |        |         |        |        |          |        |             |
|         | 2         | 490000      | ACME SCENTED OIL REFILL-CTRY SUN | 07            | 65           | 20     |        |         |        |        |          |        |             |
|         | 2         | 560000      | ACME LARGE FUDGE GRAHAMS COOKIES | 81            | 10           | 25     |        |         |        |        |          |        |             |
|         | 2         | 573000      | ACME SUGAR ICE WAFERS VANILLA    | 81            | 10           | 25     |        |         |        |        |          |        |             |
|         | 3         | 486000      | ACME ZOO ANIMAL FRUIT SNACKS 6'S | 81            | 70           | 30     |        |         |        |        |          |        |             |
|         | 3         | 488000      | ACME WAFERS SUGER ICE            | 81            | 10           | 25     |        |         |        |        |          |        |             |
|         | 3         | 490000      | ACME SCENTED OIL REFILL-CTRY SUN | 07            | 65           | 20     |        |         |        |        |          |        |             |
|         | 3         | 498000      | ACME RICE CRACKERS ONION         | 81            | 20           | 30     |        |         |        |        |          |        |             |
|         | 3         | 503000      | ACME GARBAGE BAG BLACK           | 07            | 75           | 15     |        |         |        |        |          |        |             |
|         | 3         | 530000      | ACME FUDGE DIP CHOC CHIP COOKIE  | 81            | 10           | 25     |        |         |        |        |          |        |             |
|         | 3         | 560000      | ACME LARGE FUDGE GRAHAMS COOKIES | 81            | 10           | 25     |        |         |        |        |          |        |             |
|         | 3         | 573000      | ACME SUGAR ICE WAFERS VANILLA    | 81            | 10           | 25     |        |         |        |        |          |        |             |
|         | 4         | 325000      | ACME COOKIES CHOC CHIP           | 81            | 10           | 25     |        |         |        |        |          |        |             |
|         | 4         | 325000      | ACME COOKIES CHOC CHIP           | 81            | 10           | 25     |        |         |        |        |          |        |             |
|         | 4         | 326000      | ACME DIGESTIVE RICH TEA BISCUITS | 81            | 10           | 25     |        |         |        |        |          |        |             |
|         | 4         | 327000      | ACME ASSORTED COOKIES DRP        | 81            | 10           | 25     |        |         |        |        |          |        |             |
|         | 4         | 328000      | ACME KITCHEN BAG                 | 07            | 75           | 10     |        |         |        |        |          |        |             |

**Figure: Lookup Wizard - Results**

**NOTE:** If the second dataset contains multiple matching entries for individual lookup key values from the first dataset, rows from the first dataset are duplicated in the results.

**NOTE:** You may need to delete some of the columns that have been imported into your dataset.

## Auto-updating Lookups

After you have added a lookup to your recipe, subsequent changes to that reference data are automatically reflected in the dataset.

**Tip:** If you must freeze the data in the dataset that you are using for a lookup, you should create a copy of the dataset as a snapshot. See *Dataset Details Page*.

To use the copy, delete the lookup and rebuild it using the copied version. See *Fix Dependency Issues*.



# Standardize Page

Through the Standardize page, you can review similar column values and standardize them to values that you specify.

For example, master data on customers and products may use different names for the same product. For the Web team, the product may be called, "ACME Cookies Chocolate Chip," while the data from the Accounting team refers to this product as, "Cookies - Choc Chip." Through the Standardize page, you can normalize these values to a single consistent value for easier consumption downstream.

- Standardization can be applied to a single column at a time.
- For more information on how Designer Cloud powered by Trifacta® Enterprise Edition standardizes values, see *Overview of Standardization*.

## Limitations:

- Standardizations applied through this page are stored in a connected database. These standardizations cannot be migrated between instances or workspaces.
- Standardizations cannot be published from a Dev instance to a Prod instance through Deployment Manager.

To open the Standardize page:

- From a specific column, click the column drop-down and then select **Standardize....**
- In the Search panel, enter `standardize` column. Then, select the column whose values you wish to standardize and click **Next**. See *Search Panel*.

USDA FARMERS MARKET 2014 FLOW >  
USDA Farmers Market 2014 ▾ Full Data

Clustering options 🔍 Search values...

| Row count ▾                         | Source value                       | New value                          |                    |
|-------------------------------------|------------------------------------|------------------------------------|--------------------|
| <input checked="" type="checkbox"/> | 4 ACME ZOO ANIMAL FRUIT SNACKS 6'S | ACME ZOO ANIMAL FRUIT SNACKS 6'S   | 2 values · 5 rows  |
| <input checked="" type="checkbox"/> | 1 acme zoo animal fruit snacks 6's | ACME ZOO ANIMAL FRUIT SNACKS 6'S   |                    |
| <input type="checkbox"/>            | 4 ACME FRUIT SNACK CASTLE ADVENTRS | ACME FRUIT SNACK CASTLE ADVENTURES | 2 values · 5 rows  |
| <input type="checkbox"/>            | 1 acme fruit snack castle adventrs | ACME FRUIT SNACK CASTLE ADVENTURES |                    |
| <input type="checkbox"/>            | 4 ACME BISCUITS ASSORTED           | ACME BISCUITS ASSORTED             | 2 values · 5 rows  |
| <input type="checkbox"/>            | 1 acme biscuits assorted           | ACME BISCUITS ASSORTED             |                    |
| <input type="checkbox"/>            | 4 ACME ASSORTED COOKIES DRP        | ACME ASSORTED COOKIES DRP          | 2 values · 5 rows  |
| <input type="checkbox"/>            | 1 acme assorted cookies drp        | ACME ASSORTED COOKIES DRP          |                    |
| <input type="checkbox"/>            | 4 ACME ASSORTED COOKIES            | ACME ASSORTED COOKIES              | 2 values · 5 rows  |
| <input type="checkbox"/>            | 1 acme assorted cookies            | ACME ASSORTED COOKIES              |                    |
| <input type="checkbox"/>            | 8 ACME COOKIES ASSORTED            | ACME COOKIES ASSORTED              | 2 values · 10 rows |
| <input type="checkbox"/>            | 2 acme cookies assorted            | ACME COOKIES ASSORTED              |                    |

43 clusters 129 unique source values 300 rows 2 selected (5 rows)

Standardize

New value

ACME ZOO ANIMAL FRUIT SNACKS 6'S

Revert to source

Apply

Source value

Multiple values

Row count

5

Summary

Source column

Unique new values

Source values updated

Rows updated

ProdName

86

44 / 129 (34.11%)

54 / 300 (18.00%)

Cancel

Add to Recipe

Figure: Standardize page

In the above image, Designer Cloud powered by Trifacta® Enterprise Edition groups the various references to product names into its interpretation of meaningful clusters. This clustering is based on pattern-matching between values in the column.

**NOTE:** The Standardize page displays column values from the currently selected sample only. If the sample does not span the entire dataset, column values that are not captured in the display are not

affected by standardization changes. You may need to take additional samples to capture column values outside the current sample.

In the left side of the screen, you can review the clusters of values that have been detected in the column. In the above image, you can see that the platform has identified a number of clusters based on simple differences in capitalization.

- For each cluster, you can review the number of unique values and the total number of rows where the values appear in the column.
- At the bottom of the left pane, you can review the total number of unique values in the source column and the total number of rows in the displayed sample.

## Toolbar



**Figure: Standardize toolbar**

- **Undo:** Undo the last action in the Standardize page

**NOTE:** This action does not undo recipe steps that have already been added.

- **Redo:** Redo the last undo action.
- **Auto Standardize:** The Wand tool automatically standardizes values based in a cluster to the most common value, as long as the most common value occurs in 25% of the clustered rows.

**NOTE:** For auto-standardization, the most common value is determined based on the cluster of values that are displayed in the current sample.

**Tip:** The Wand tool is recommended for beginning the standardization process. If values within a cluster have been modified, the cluster is not affected by the Wand tool. You can also apply the Wand tool on selected values in a cluster.

- **Clustering options:** By default, values are clustered based on similar spellings. To change the algorithm by which values are clustered, click **Clustering options**.
  - **None:** Do not cluster values. Individual values must be matched.
  - **Similar strings:** Cluster values based on similarities between the text of each value.
  - **Pronunciation:** Cluster values based on phonetic pronunciation of the values.
  - For more information on these options and their variations, see *Overview of Standardization*.
- **Search values:** To locate specific values in the column, enter a search string in the Search textbox.
- To reverse the sort order within your clusters, click Row Count.
  - To sort cluster values alphabetically, click the Source Value header.

## Steps to Standardize

To standardize values:

1. If needed, change the clustering algorithm to apply to the values.
2. From the left panel, select the set of values that you wish to standardize.

**Tip:** Unclustered values are listed at the bottom of the panel. You should review these values when you are selecting clustered values.

- a. To select multiple values, press `COMMAND/CTRL` + click or select multiple values in the left column.

**Tip:** To select all values in a cluster, click the cluster header.

- b. To select a range of sequentially listed values, use `SHIFT` + click, which works across clusters of values.
3. After you have selected all values to standardize, you specify the new value to apply to these selected values in the right panel. This new value applies to all instances of the selected value or values. Changes are previewed next to the source values.

**Tip:** When you have values selected in a cluster, you can use one of the source values as your standardized value. Hover over the value in the left panel, and then click the icon that appears.

- a. Below the value you enter, you can review the number of rows in the sample that are affected by this change.
- b. At the bottom of the right panel, you can review the total effects of standardization on the dataset after this change is applied.
- c. If the new value is empty, then the values are kept as-is. No change is applied.
- d. To apply the standardized value to the affected clustered values, click **Apply**.
- e. At any time, you can revert the changes to the cluster values. Click **Revert to source**.
4. Repeat the previous steps as needed.

**Tip:** You can perform multiple replacements in a single recipe step. So, you can configure all of your standardization steps before adding the single step to your recipe. For debugging purposes, you may want to separate some or all standardization into separate steps.

5. To add the standardizations, click **Add to Recipe**.

**NOTE:** You cannot copy and paste standardization steps in the Recipe panel.

# Custom Type Dialog

## Contents:

- *Dictionary File Format*
  - *Create Custom Type tab*
  - *Use Existing Custom Type tab*
- 

Through the data type menu in each column, you can select different data types, including custom ones. You can also create custom data types in Designer Cloud powered by Trifacta® Enterprise Edition.

A **custom data type** is similar to the pre-defined data types, except that you provide a dictionary file of all accepted values. During data validation, values in a column of a custom data type are validated against the dictionary file to determine if the data is valid or not.

**NOTE:** After a custom type has been created, a platform restart is required. Please contact your Trifacta administrator.

Custom data types can be specified by regular expression patterns or by values defined in an uploaded dictionary file. For more information, see *Create Custom Data Types Using RegEx*.

## Dictionary File Format

For more information on the specification for file-based dictionaries, *Create Custom Data Types*.

## Create Custom Type tab

Upload your dictionary file and select the column in it to use for validation of the data type.

Use Existing Custom Type

Create New Custom Type

×

Name:

| Dictionary Name             |                                                                                                                                                                                                                                                                                                                                                               |         |             |       |        |       |             |                   |    |   |   |   |    |
|-----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|-------------|-------|--------|-------|-------------|-------------------|----|---|---|---|----|
| <div>▼</div> Dict-Sizes.csv | <div>Upload Dictionary</div>                                                                                                                                                                                                                                                                                                                                  |         |             |       |        |       |             |                   |    |   |   |   |    |
| <div>✓</div> column1        | <table> <thead> <tr> <th>column1</th> </tr> </thead> <tbody> <tr><td>Extra Small</td></tr> <tr><td>Small</td></tr> <tr><td>Medium</td></tr> <tr><td>Large</td></tr> <tr><td>Extra Large</td></tr> <tr><td>Extra Extra Large</td></tr> <tr><td>XS</td></tr> <tr><td>S</td></tr> <tr><td>M</td></tr> <tr><td>L</td></tr> <tr><td>XL</td></tr> </tbody> </table> | column1 | Extra Small | Small | Medium | Large | Extra Large | Extra Extra Large | XS | S | M | L | XL |
| column1                     |                                                                                                                                                                                                                                                                                                                                                               |         |             |       |        |       |             |                   |    |   |   |   |    |
| Extra Small                 |                                                                                                                                                                                                                                                                                                                                                               |         |             |       |        |       |             |                   |    |   |   |   |    |
| Small                       |                                                                                                                                                                                                                                                                                                                                                               |         |             |       |        |       |             |                   |    |   |   |   |    |
| Medium                      |                                                                                                                                                                                                                                                                                                                                                               |         |             |       |        |       |             |                   |    |   |   |   |    |
| Large                       |                                                                                                                                                                                                                                                                                                                                                               |         |             |       |        |       |             |                   |    |   |   |   |    |
| Extra Large                 |                                                                                                                                                                                                                                                                                                                                                               |         |             |       |        |       |             |                   |    |   |   |   |    |
| Extra Extra Large           |                                                                                                                                                                                                                                                                                                                                                               |         |             |       |        |       |             |                   |    |   |   |   |    |
| XS                          |                                                                                                                                                                                                                                                                                                                                                               |         |             |       |        |       |             |                   |    |   |   |   |    |
| S                           |                                                                                                                                                                                                                                                                                                                                                               |         |             |       |        |       |             |                   |    |   |   |   |    |
| M                           |                                                                                                                                                                                                                                                                                                                                                               |         |             |       |        |       |             |                   |    |   |   |   |    |
| L                           |                                                                                                                                                                                                                                                                                                                                                               |         |             |       |        |       |             |                   |    |   |   |   |    |
| XL                          |                                                                                                                                                                                                                                                                                                                                                               |         |             |       |        |       |             |                   |    |   |   |   |    |

Cancel

Save

**Figure: Create Custom Type tab**

### Steps:

1. Click **Upload Dictionary**.
2. Select your CSV file. Expand the caret next to the filename. Select the column to use for data validation. Name your new custom data type.
3. Click **Save**.

The new data type is applied to the currently selected column.

### Use Existing Custom Type tab

**NOTE:** If you cannot see a recently created custom data type, you may need to logout and login again.

Select the custom data type to apply to the currently selected column. Click **Save**.

Use Existing Custom Type

Create New Custom Type

×

| Name    | Dictionary     |
|---------|----------------|
| ✓ Sizes | Dict-Sizes.csv |

Cancel

Save

**Figure: Use Custom Type tab**

# Recipe Navigator

Through the Recipe Navigator, you can locate and open any recipe from the current flow in the Transformer page. To open the Recipe Navigator, select the drop-down next to the name of the current recipe in the Transformer page.


This navigator can be helpful in fixing dependency issues between your current recipe and the recipes and datasets that it integrates from the flow. See *Fix Dependency Issues*.


**NOTE:** The listed recipes are constrained only to those that appear in the same flow as the currently loaded recipe.


**NOTE:** You can only open recipes in the Transformer page. To open an imported dataset, you must first create a recipe for it and then edit the recipe. See *Flow View Page*.


Navigate to...


Q Search...

✓  POS-r01

 REF\_PROD

 POS-r03 – 2.txt

 REF\_CAL – 2.txt

 POS-r02 – 2.txt

**Figure: Recipe tab**

To locate a recipe to load, use the Search box or browse the list. Then, select the recipe to load.

# Transformer Toolbar

## Contents:

- *Grid/Columns*
- *Undo/Redo*
- *Replace*
- *Extract*
- *Count Matches*
- *Split Column*
- *Merge Columns*
- *Format*
- *Create column by examples*
- *Group by*
- *Pivot*
- *Unpivot*
- *Convert values to columns*
- *Objects and Arrays*
- *Filter Rows*
- *Functions*
- *Conditions*
- *Join*
- *Union*
- *Comment*
- *Target*
- *Select*
- *Macros*
- *Find Column*
- *Filters*

At the top of the data grid and the column browser, the Transformer toolbar provides quick access to common transformations.

When a tool is selected from one of the drop-downs in the toolbar, a new step is inserted into the current location in your recipe with some of the transformation's parameters pre-specified for you. You can then finish specifying the parameters of the transformation in the Transform Builder.



**Figure: Transformer Toolbar**

## Grid/Columns

Use these buttons to toggle between the data grid and column browser views of the Transformer page.

- See *Data Grid Panel*.
- See *Column Browser Panel*.

## Undo/Redo

Undo or redo previous actions.



## Replace

Replace cell values based on data type validation, string literals, or patterns. See *Replace Groups of Values*.

## Extract

Extract string or numeric values using data validation, literal matches, or patterns. See *Extract Values*.

## Count Matches

Count the number of matching literals or patterns in a column.

## Split Column

Split a single column into multiple columns based on a common delimiter or a pattern-based expression.

## Merge Columns

Merge two or more columns together.

## Format

Format String and Datetime values using a variety of functions.

## Create column by examples

Define transformations by mapping output values from a set of input values. After you specify a few values, Designer Cloud powered by Trifacta Enterprise Edition may be able to interpret the other ones for you. See *Create Column by Example*.

## Group by

Compute aggregation functions for values grouped by columns. Output is new table or one or more columns in the current dataset. For more information, see *Create Aggregations*.

## Pivot

Generate a pivot table of your data based on specified row and column labels. See *Pivot Data*.

## Unpivot

Collapse columns into row data.

## Convert values to columns

Reshape your data by converting unique values into columns or columns into rows.

## Objects and Arrays

Use these tools to manipulate your dataset based on columns of Object or Array type.

## Filter Rows

Remove or keep rows of data based on data type validation, conditionals, source row numbers, or other factors.

- See *Remove Data*.
- See *Deduplicate Data*.

## Functions

Generate a new column of data based on a specified function.

For a list of available functions, see *Language Index*.

## Conditions

Use if/then/else or case logic to apply conditionals to your dataset.

See *Apply Conditional Transformations*.

## Join

Join the current dataset with another dataset using a pair of matching keys.

See *Join Window*.

## Union

Add the rows of data from one or more datasets to your current dataset.

See *Union Page*.

## Comment

Insert a comment in your recipe at the current step. Comments do not perform any modification to the data.

See *Add Comments to Your Recipe*.

## Target

Attach a target schema to the current recipe or remove it. If a target is attached, you can attempt to auto-match all source and target columns.

- For more information on these options, see *Column Browser Panel*.
- For more information on this feature, see *Overview of RapidTarget*.

## Select

**NOTE:** This menu is only available in the Column Browser.

Select all or no columns, or invert the currently selected columns. See *Column Browser Panel*.

## Macros

Click the Macros icon to open the menu:

- **Insert macro:** Search for or select the name of the macro to apply. When a macro is selected, it is opened in the Transformer Builder, where you may specify any parameter values to apply to the macro. See *Apply a Macro*.
- **Manage in Library:** Review available macros. For more information, see *Macros Page*.
- **Create macro:** Enter a name and description for the macro to create. See *Create or Replace Macro*.

For more information, see *Overview of Macros*.

## Find Column

Locate a column in your dataset by typing in the textbox. For more information, see *Data Grid Panel*.

## Filters

Filter the displayed rows and columns based on selection or condition.

**NOTE:** Filtered data is hidden from display. It is not removed from the sample or the dataset during execution.

See *Filter Data*.

# Column Menus

## Contents:

- *Data Type Menu*
  - *Action Menu*
    - *Show in Grid*
    - *Rename*
    - *Change type*
    - *Move*
    - *Hide*
    - *Sort*
    - *Edit with formula*
    - *Format*
    - *Calculate*
    - *Create column from examples*
    - *Group by*
    - *Pivot*
    - *Restructure*
    - *Filter rows*
    - *Replace*
    - *Standardize*
    - *Extract*
    - *Split column*
    - *Column Details*
    - *Show related Steps in Recipe*
    - *Lookup*
    - *Delete*
    - *Delete others*
    - *Copy, Cut, and Paste*
- 

You can use the menus available in a column context menu to perform a variety of actions on the column or columns.

Column menus are available in the following pages:

- *Data Grid Panel*
- *Column Browser Panel*
- *Column Details Panel*

Depending on the data type of the column, the available options may vary.

**NOTE:** In the Column Browser panel, you can select multiple columns at the same time. The displayed column menu items are only the ones that apply to columns of all data types.

## Data Type Menu

On the left side of the column header, in the Data Grid you can click the data type icon to review and select a different data type for the column.

- When you select the Datetime data type, you can choose the format against which values in the column are validated. See *Choose Datetime Format Dialog*.
- For more information, see *Supported Data Types*.

You can also create custom data types for use in the Designer Cloud powered by Trifacta® platform . See *Custom Type Dialog*.

## Action Menu

On the right side of the column header, select the drop-down caret to choose one of the following actions.

- Some options are available only for specific data types.
- In most cases, these actions result in a new step being inserted at the current location in your recipe. Before you apply one, you should verify that you are at the proper position in the recipe to apply this step.

## Show in Grid

When a single column is selected in the Column Browser panel, you can use this option to select the data grid and change its focus to display the selected column.

**Tip:** Use the Column Browser panel to quickly locate and select columns for display or other actions. See *Column Browser Panel*.

## Rename

Rename the column.

For more information on column name requirements, see *Rename Columns*.

## Change type

Change the data type of the column. See *Supported Data Types*.

## Move

Move the column to the beginning or end or to a specified location in the dataset.

## Hide

Hide this column from display in the application. To redisplay this column, you must select it in the Visible Columns panel. See *Visible Columns Panel*.

**NOTE:** Hide/Show commands do not create transformation steps. Hidden columns still exist in the output.

**NOTE:** When a column is hidden from a dataset, it is hidden for all users. You should check the column browser for hidden columns in shared datasets.

## Sort

**Sort:** Sort the values in the column by ascending or descending value. This command does not apply to some data types.

**NOTE:** Sort is intended primarily for display purposes in the Transformer page.

When a column is sorted in ascending order, the bottom values are mismatched and null values, in that order. These values appear at the top of the column when sorted in descending order.

### Edit with formula

Modify the column values based on a formula that you input. For more information, see *Transform Builder*.

### Format

Format the values in the column according to the selection.

**NOTE:** This menu is available only for columns of String and Datetime type.

### Calculate

Calculate a new column containing the values computed from the source column based on the selected function. This command does not apply to all data types.

### Create column from examples

Using a source column, you can create a new column using the source values as examples.

**Tip:** The transformation-by-example feature can be used in place of standardization or extraction transformations and may be simpler to use.

**NOTE:** This feature may need to be enabled in your environment.

See *Create Column by Example*.

For more information on transformation-by-example, see *Overview of TBE*.

### Group by

Group by minimum or maximum value, counts, or arbitrary function. See *Create Aggregations*.

### Pivot

Generate summary table based on computations aggregated across groups.

### Restructure

Restructure your dataset by converting column values to columns, nesting values into objects or arrays, and unpivoting your data.

### Filter rows

Filter the rows in your dataset based on column values, duplicate values in the column, or other condition. See *Filter Data*.

### Replace

Replace values in the selected column with based on text values, patterns, or delimiters. See *Replace Groups of Values*.

## Standardize

Standardize individual and clustered row values for consistency. See *Standardize Page*.

## Extract

Extract values from the selected column based on text values, specific types of data, or patterns in the data. Optionally, extracted data can be removed from the source. See *Extract Values*.

## Split column

Split the column into separate columns based on patterns, delimiters, positions in the value, or regular intervals throughout the value.

## Column Details

Explore the interactive profile of the column details. See *Column Details Panel*.

## Show related Steps in Recipe

Highlight steps in the recipe panel where the selected column is referenced.

- If another column is dependent on the selected column, all steps pertaining to that column are highlighted as well.
- To dismiss the highlighting, click **Clear** at the top of the recipe panel.
- For more information on disabling this feature, see *Miscellaneous Configuration*.

## Lookup

Perform a lookup of the column values against a set of values in another column of another dataset. See *Lookup Wizard*.

## Delete

Remove the column from the dataset.

**Tip:** You can also select **Hide** to remove the column from display in the data grid. The column does remain as part of the dataset and is included in any generated results.

## Delete others

Delete the columns other than the selected column(s) from the dataset.

## Copy, Cut, and Paste

You can copy and paste columns and column values into other areas of your dataset. For more information, see *Copy and Paste Columns*.

# Choose Datetime Format Dialog

In the Datetime Format dialog, you can specify or locate the format that you'd like to use for validation of the values in the current column against the Datetime data type.

For example, if you choose `mmddyyyy` as your format, all values in the column are considered valid for the Datetime data type if they fit the `mmddyyyy` pattern. Otherwise, they are considered invalid values for the Datetime column.

Choose Date/Time Format

Search for formats or type an example

✓

yyyy\*mm\*dd

2019/11/7

mmddyyyy

1172019

mm\*dd\*yy

11/7/19

mm\*dd\*yyyy

11/7/2019

month\*dd\*yy

November 7 19

month\*dd\*yyyy

November 7 2019

Learn about date/time formats

Cancel

Select Format

Figure: Choose Datetime Format dialog

**NOTE:** The list of available Datetime formats is factored based on your current locale settings. Some formats from other locales may be available for selection. For more information, see *Locale Settings*.

To locate a format, you can:

- **Search:** You can start entering your preferred format as tokens. For example, if you enter `yyyy`, you can narrow the list to the formats that support four-digit values for year.
- **Browse:** You can scroll the list to see the available formats.
- **Enter:** Type or paste in an example Datetime value that is in your preferred format.

For more information on the supported tokens for the Datetime data format, see *Datetime Data Type*.

To apply your selected format, click **Select Format**. The values in the column are validated against the selected format for the Datetime data type.



# Transformation by Example Page

## Contents:

- *Transform Builder*
  - *Grid View*
    - *Keyboard shortcuts*
  - *Pattern View*
  - *Toolbar*
- 

In the Transform by Example page, you can build new columns of data by specifying values to map from the selected source column. For the column to transform, select **Create column by examples** from the column menu.

**NOTE:** Transformations by example are developed using the current sample. When the finished transformation is applied across the entire dataset, some source values may not be matched by the patterns you specified using the current sample.

**Tip:** Transformation by example works best for text-based inputs. For non-text inputs, you can convert the column data type to String.

To transform by example, you can use either of the following:

- **Grid View:** Displays Source and Preview column values as they appear in the sample in the data grid.
- **Pattern View:** Displays the Source and Preview values in groups of similar values based on pattern matching by Designer Cloud powered by Trifacta® Enterprise Edition.

For more information on these types of transformations, see *Overview of TBE*.

## Transform Builder

In the Transform Builder panel on the right, you can review and change the Source and Preview columns to transform.

**Tip:** To transform from a different source column, select a new column from the Example column drop-down. This step clears all examples from the transformation you are building. Some columns may not be available for selection. To use such a column, change its type to String first.

## Grid View

In grid view, the Source values for each row in the sample are listed next to an empty Preview column. You can create mappings by selecting a cell in the Preview column and manually entering a value.

DATASETS - TBE FLOW >
Datasets - TBE
Full Data

Run Job

Source
Preview

|    | Phone Number   | Phone_Number_clean |
|----|----------------|--------------------|
| 1  | 510-221-2244   | 510 221 2244       |
| 2  | 510-221-2245   | 510 221 2245       |
| 3  | 510-221-2246   | 510 221 2246       |
| 4  | 510-221-2247   | 510 221 2247       |
| 5  | 510-221-2248   | 510 221 2248       |
| 6  | 510-221-2249   | 510 221 2249       |
| 7  | 510-221-2250   | 510 221 2250       |
| 8  | 510-221-2251   | 510 221 2251       |
| 9  | 510-221-2252   | 510 221 2252       |
| 10 | 510-221-2253   | 510 221 2253       |
| 11 | 510-221-2254   | 510 221 2254       |
| 12 | (510) 434 4404 | 510 434 4404       |
| 13 | (510) 434 4405 | 510 434 4405       |
| 14 | (510) 434 4406 | 510 434 4406       |
| 15 | (510) 434 4407 | 510 434 4407       |
| 16 | (510) 434 4408 | 510 434 4408       |
| 17 | (510) 434 4409 | 510 434 4409       |
| 18 | (510) 434 4410 | 510 434 4410       |
| 19 | (510) 434 4411 | 510 434 4411       |
| 20 | (510) 434 4412 | 510 434 4412       |
| 21 | (510) 434 4413 | 510 434 4413       |
| 22 | (510) 434 4414 | 510 434 4414       |

195 rows 60 unique source values

Create column from examples

Create a new column by providing example values in the grid.

Column

Phone Number

required

New column name

Phone\_Number\_clean

Cancel
Add to Recipe

**Figure: Transformation by Example - Grid View**

After you enter a value, Designer Cloud powered by Trifacta Enterprise Edition attempts to match other values from the Source using the same pattern to generate additional values in the Preview column.

- Values that you manually enter are listed in dark text.
- Values that are inferred by the product are in lighter text.
- Null values indicate that no pattern has been identified to match the value.

**Tip:** Values that have been inferred can be replaced by manual entries for further refinement.

For more information on how to use, see *Create Column by Example*.

### Keyboard shortcuts

- Use the arrow keys to navigate up and down the rows in the Preview column.
- CTRL + up arrow or CTRL + down arrow to jump to the first or last row of the sample
  - In Pattern View, the above shortcuts navigate between groups of values.
- ESC cancels your current edit.
- RETURN submits your current entry as a new example.

**Tip:** You can copy and paste values from the clipboard into the Preview column.

### Pattern View

In Pattern view, Designer Cloud powered by Trifacta Enterprise Edition performs some preliminary pattern detection to group Source values together. Transformations are processed using Patterns .

DATASETS - TBE FLOW >
Full Data

---

| Source         | Preview               |
|----------------|-----------------------|
| Phone Number   | Phone_Number_clean    |
| Pattern 1      | 1 of 1 unique value   |
| 5104289900     | 510 428 9900          |
| Pattern 2      | 5 of 25 unique values |
| (510) 434 4404 | 510 434 4404          |
| (510) 434 4405 | 510 434 4405          |
| (510) 434 4406 | 510 434 4406          |
| (510) 434 4407 | 510 434 4407          |
| (510) 434 4408 | 510 434 4408          |
| Pattern 3      | 5 of 23 unique values |
| 1-510-122-3300 | 510 122 3300          |
| 1-510-122-3301 | 510 122 3301          |
| 1-510-122-3302 | 510 122 3302          |
| 1-510-122-3303 | 510 122 3303          |
| 1-510-122-3304 | 510 122 3304          |
| Pattern 4      | 5 of 11 unique values |
| 510-221-2244   | 510 221 2244          |
| 510-221-2245   | 510 221 2245          |
| 510-221-2246   | 510 221 2246          |
| 510-221-2247   | 510 221 2247          |

195 rows    60 unique source values    4 patterns

### Create column from examples

Create a new column by providing example values in the grid.

Column required

Phone Number

New column name

Phone\_Number\_clean

Cancel
Add to Recipe

**Figure: Transformation by Example - Pattern View**

- This view displays a maximum of five example values per pattern group.
- **Other group:** Values that do not map to any identifiable pattern are placed in a value group labeled, `Other`.
- For more information on patterns, see *Text Matching*.

## Toolbar



**Figure: Transformation by Example toolbar**

- **Grid View:** See previous.
- **Pattern View:** See previous.
- **Undo:** Undo the last change you made to the Preview values.
- **Redo:** Redo the most recently undo change you made to the Preview values.
- **Trash:** Remove all example values from the Preview column and start over.

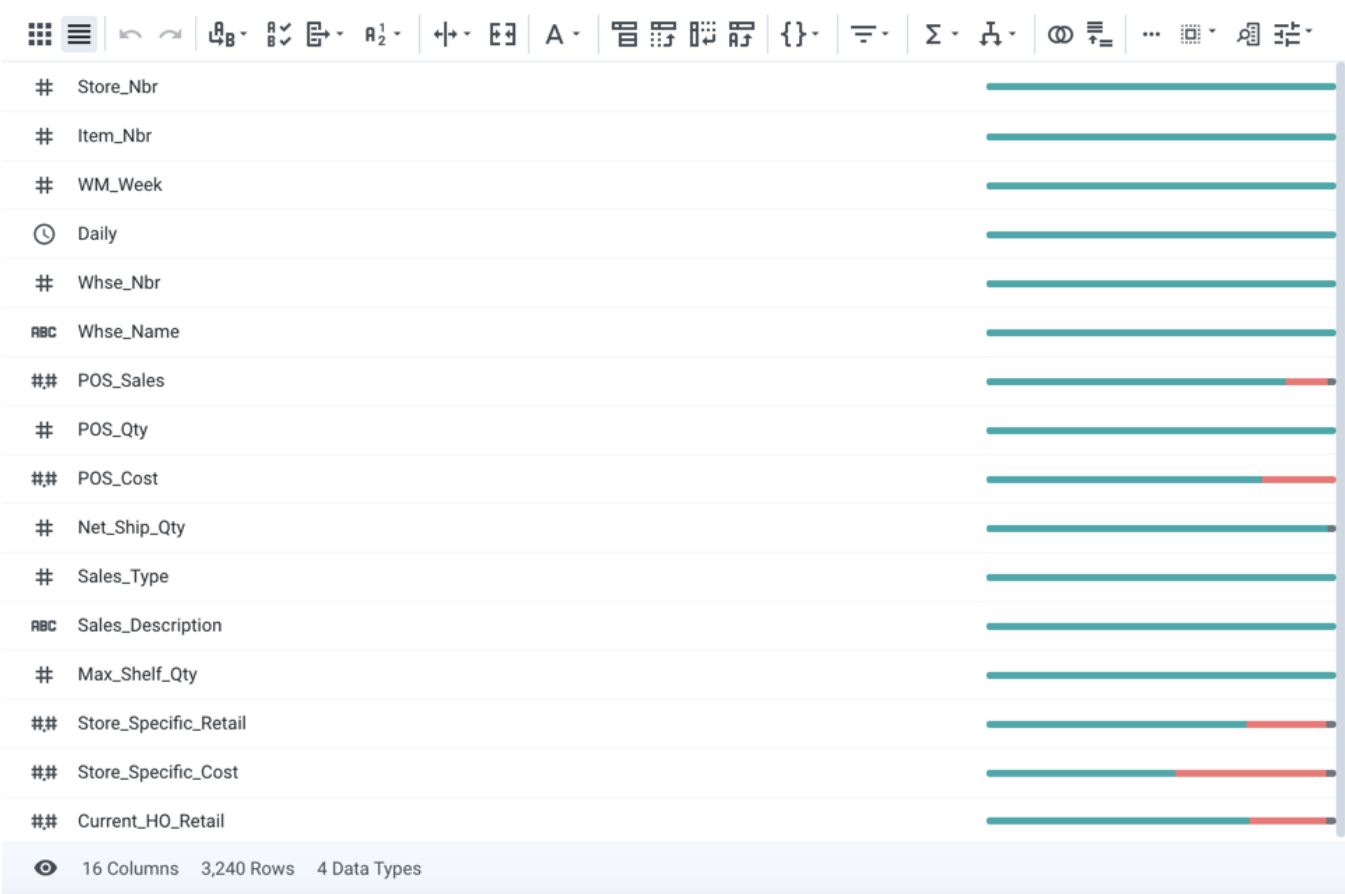
# Column Browser Panel

Contents:

- *Locate Columns*
- *Select Columns*
- *Transformer Toolbar*
- *Column Actions*
- *RapidTarget Matching*

Through the Column Browser, you can use data quality bars and data type information to perform basic review of data across many columns. You can use these tools to select data of interest for display in the data grid or Column Details views or to prompt for suggestions of recipe steps.

- You can also use the Column Browser to toggle the display of individual columns.
- To open the Column Browser, click the Column View icon in the Transformer bar for the Transformer page.



**Figure: Column Browser**

You can select one or more columns in the browser and then perform actions on them.

## Locate Columns

You can apply one or more filters to limit the set of columns displayed in the browser. Click the Filter icon in the Transformer toolbar.

**NOTE:** Filters are additive and persist between the column browser and the data grid.

For more information, see *Filter Panel*.

## Select Columns

You can manually select one or more columns or apply one of the predefined selections.

- To select a range of columns, click a column, press **SHIFT** and then click the ending column.
- To select multiple discrete columns, press **CTRL/COMMAND** and click additional columns.
- To toggle selection of a column, click it again.
- You can copy and paste columns and column values. For more information, see *Copy and Paste Columns*.

## Transformer Toolbar

At the top of the column browser, you can use the toolbar to quickly build common transformations, filter the display, and other operations. See *Transformer Toolbar*.

## Column Actions

For any individual column:

- Click the Eye icon to hide/show of the column in the Transformer page. See *Visible Columns Panel*.

**NOTE:** Hidden columns are only removed from view in the Transformer page. They still appear in any generated output.

- Hover over the color bars in the data quality bar to review counts. See *Data Quality Bars*.
- Right-click a column to display a list of actions in the context menu. Column actions apply only to the selected column and depend on its data type.
  - For multiple selected columns, you can choose an action from the Action menu an option that apply to all of the selected columns.
  - See *Column Menus*.

## RapidTarget Matching

You can associate a target schema with a recipe. A **target schema** is information about the column names, data types, and order of the target dataset for which you are trying to build your recipe. For more information, see *Overview of RapidTarget*.

Actions taken based on the target schema are rendered as a new step in the location in your recipe.



|     |     |     |                      |                                                                                       |
|-----|-----|-----|----------------------|---------------------------------------------------------------------------------------|
| Yes | No  | Yes | Arrow - blue outline | Source column and target column match in name and position, but have different types. |
| Yes | Yes | Yes | Arrow - green solid  | Source and target columns match in name, position, and type.                          |

### Target menu actions:

From the Transformer toolbar, you can select the following options from the Target menu:

- **Attach a new Target:** Assign a target schema to the dataset.
- **Remove Target:** Remove the target from assignment to the source. Target schema dataset is not deleted.
- **Hide/Show Target data:** Toggle display of example rows from the target schema dataset.
- **Align on column name match:** Columns are automatically matched when column names match.
- **Align on fuzzy match:** Match columns based on the data contained in them.

**NOTE:** This method of matching is available if global fuzzy matching has been enabled. For more information, see *Overview of RapidTarget*.

When you have selected one or more source columns, you can additionally perform the following actions:

- **Align selected on column name:** Fix the selected columns to match target columns using the column names.
- **Align selected on fuzzy match:** Fix the selected columns to match target columns using the data contained in them.

**NOTE:** This method of matching is available if global fuzzy matching has been enabled. For more information, see *Overview of RapidTarget*.

# Column Details Panel

## Contents:

- Overview tab
- Patterns tab
  - Pattern reuse

In the Column Details panel, you can review additional details about a column of your dataset. Select **Column Details** from any column menu or the Action menu in the column browser.

**Tip:** Use the Column Details panel to explore values in an individual column, when the context of the value is not important for your current exploration. For example, you can identify outlier values for the column or compare the number of unique values to number of rows to determine whether the column could be a key value.

## Overview tab

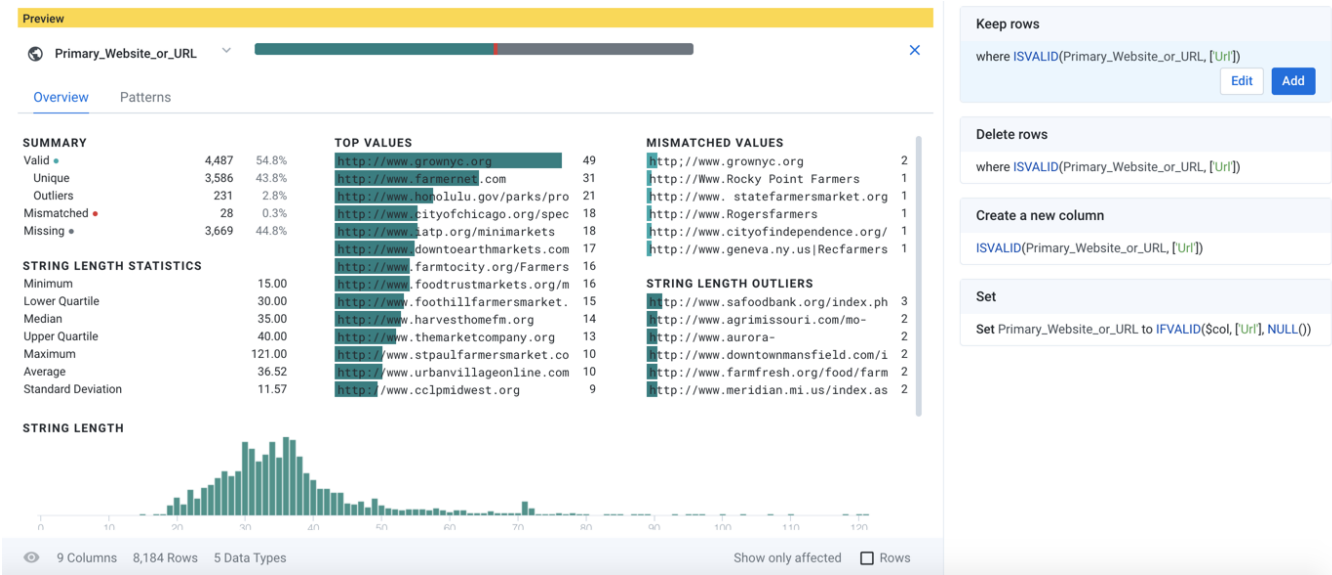


Figure: Column Details panel - Overview tab

## Column statistics:

You can use this view to review basic counts and percentages of the values in the currently selected column. In addition to basic computations on valid, mismatched, and missing values, you can see breakdowns for the most frequent values and outlier values.

**NOTE:** Before your job is run, profiling information such as column statistics are exact counts of the sample that is currently loaded. After the job is run, profiled results in the Job Results page might include estimates for some metrics and counts, depending on the scale of the dataset.



Depending on the data type of the column, additional statistics provide information on data quality and variation. For more information, see *Column Statistics Reference*.

**Actions:**

- To change the data type, click the type indicator next to the column title in the Column Details panel.
- To perform commands on the column, select from the drop-down next to the column title. For more information, see *Column Menus*.
- Use the data quality bar to select categories of values: valid, mismatched, or missing. The context panel is updated based on your selection with recommended recipe steps. See *Selection Details Panel*.

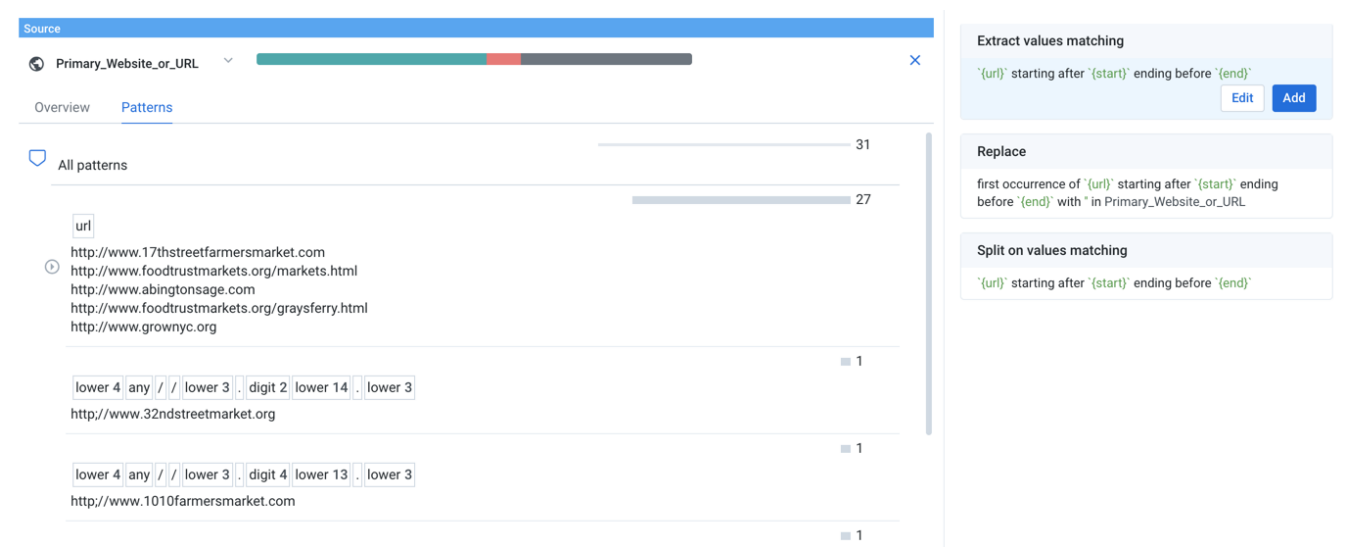
**Patterns tab**

In the Patterns tab, you can review patterns identified by the platform in the selected column's data and then create steps based on patterns that you select. Pattern profiling automatically finds and groups clusters of the column's values based on similarities in format and structure, such as differently formatted phone numbers, addresses, log entries, and name fields. For example, if some of your dataset's address values include apartment numbers, you can create a Split transformation based on a pattern that includes the apartment numbers.

**NOTE:** In this tab, the count of values and the `all patterns` category do not include missing values.

**NOTE:** Wide columns, such as Arrays, Objects, or freeform text, might take a while to profile.

- Each non-blank value in the column is represented by one of the displayed patterns. Patterns are specified as a combination of literal values and `Patterns`. For more information on these patterns, see *Text Matching*.
- Patterns might be more generalized than the constraints of the column's data type.
- Token values are `Patterns` without braces.



**Figure: Column Details panel - Patterns tab**

All non-blank values are captured in the `all patterns` category, which you can expand to display the patterns that capture subsets of all values. Patterns are displayed in a tree structure, with each lower level describing a subset of the parent pattern.

**Tip:** Hover over a pattern or sub-pattern to see the affected values in the example data beneath it.

**Tip:** When you select a pattern group, you may be presented with suggestions for standardizing the values in the column to a single format. In some cases, you might want to remove unnecessary data first. For example, standardization of phone numbers is easier if any +1 country codes are removed from the beginning of values.

**Tip:** Pattern suggestions are created based on the first few thousand rows of data in your sample. For best results, you should generate a random sample with a representative set of patterns in the first rows in the column.

Below the top level, patterns are displayed in order of decreasing frequency in the column, allowing you to choose the level of granularity for which you wish to address data issues in the column. For each pattern, you can review the counts of values matching the pattern.

In the above example, all values that have been identified as matching the `url` Pattern are contained in the first category.

- Select a pattern to trigger a set of suggestion cards to apply to the represented data.
  - When you select values from a pattern's histogram, all suggestions match the pattern. You cannot select the values that do not match the pattern from the histogram.
  - For more information, see *Explore Suggestions*.
- Select a token within a pattern or a highlighted block of text among the example values to trigger suggestion cards that apply the token within the pattern.
- You can modify the selected suggestion in the Transform Builder. See *Transform Builder*.
  - When you apply the transformation to your recipe, the Patterns tab is updated automatically.

**Tip:** When you see a pattern that you wish to reuse, select the pattern and one of its suggestion cards and then modify the step.

- Expand the caret next to any pattern to explore its sub-patterns, which identify subsets of values within the broader pattern.

**NOTE:** The `Other` pattern is a special category that contains values and counts not recognized by the currently selected pattern or sub-pattern. For example, when you select `url` pattern, the `Other` pattern captures the non-URL values. When you explore a sub-pattern of URLs, the `Other` category captures the values not recognized within the sub-pattern.

For more information on pattern standardization, see *Standardize Using Patterns*.

For more information on standardizing numeric values, see *Normalize Numeric Values*.

## Pattern reuse

After patterns have been selected, they can be reused through the Transform Builder.

# Transform Preview

When you create or edit a transform, the data grid displays a preview of results of the transform. **Transform previews** assist in specifying and validating the transformation steps before they are applied.

Although final results may not be exactly represented, a preview gives a good indication of the changes. When you review a preview, keep the following in mind:

- Previewed columns cannot be filtered or hidden.
- You cannot rename or change the data type of a previewed column until you add the change to your recipe.
- Selection of column headers is disabled.

For example, you want to remove leading and trailing quotation marks from all columns. You highlight the quotation marks in one cell and select a suggestion card for Replace:

Source

to be dropped

Preview

| column2          | column3    | column4    | column5      | column5 |
|------------------|------------|------------|--------------|---------|
| ABC              | ABC        | ABC        | ABC          | ABC     |
| column2          | column3    | column4    | column5      | column5 |
| ACCOUNT          | END_DATE   | START_DATE | CUSTOMER_AGE |         |
| 4208107376653317 | 2014-01-03 | 2007-01-03 | 7            |         |
| 4208106585213952 | 2014-10-27 | 2004-10-27 | 9            |         |
| 4208109505943761 | 2012-02-18 | 2010-02-18 | 2            |         |
| 4208113686690378 | 2015-06-30 | 2007-06-30 | 6            |         |
| 4208104023796892 | 2015-08-04 | 2006-08-04 | 7            |         |
| 4208102203151829 | 2015-06-01 | 2003-06-01 | 10           |         |
| 4208112003532067 | 2014-04-23 | 2004-04-23 | 9            |         |
| 4208106783584469 | 2015-06-17 | 2006-06-17 | 7            |         |
| 4208112393379296 | 2014-11-19 | 2003-11-19 | 10           |         |

Affects 17 columns, all rows

|           |              |              |    |    |
|-----------|--------------|--------------|----|----|
| 393379296 | "2014-11-19" | "2003-11-19" | 10 | 10 |
| 840756415 | "2014-01-02" | "2002-01-02" | 12 | 12 |
| 829594575 | "2014-06-16" | "2006-06-16" | 7  | 7  |
| 424063352 | "2014-09-18" | "2005-09-18" | 8  | 8  |
| 992553499 | "2014-11-14" | "2010-11-14" | 3  | 3  |
| 312445947 | "2015-04-30" | "2004-04-30" | 9  | 9  |
| 660745200 | "2014-02-24" | "2007-02-24" | 6  | 6  |
| 568375941 | "2015-04-07" | "2006-04-07" | 7  | 7  |

nns 3,532 Rows 2 Data Types

Show only affected Columns Rows

Replace

" with ' in column5

Edit Add

{start}"|"{end}' with ' in column5

{start}"|"{end}' with ' in all columns

Split on values matching See all

" 2 times

Extract values matching See all

" starting after {start}' ending before {alphanum-underscore}+

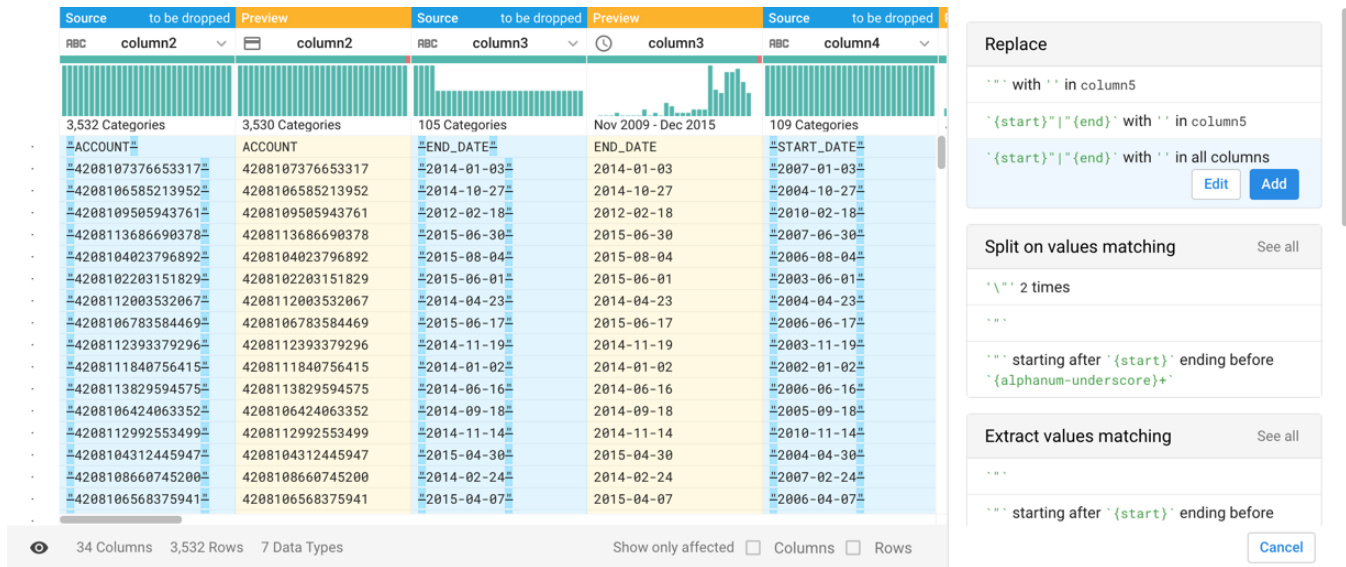
Cancel

Figure: Replace Suggestion Card

To replace in all columns, you click the third option. To verify that it is correct for your needs, you click **Edit**.

The transform preview displays in the data grid for each column to be modified:

**Tip:** The preview also contains updated data quality bars and column histograms. You can use them to test for changes on counts or column values.



**Figure: Transform Preview for Replace**

The transform preview displays for all valid transforms.

**Tip:** Press **ESC** to cancel a preview.

When you modify the transform in the Transform Builder, the preview is updated as you type.

- You can review the preview results using independent scroll bars in the transform preview.
- Previews of transforms are displayed next to each source column.
- You can apply filters to the preview through the filter drop-down. These filters persist after the preview is completed.

Preview color scheme:

| Color   | Meaning              |
|---------|----------------------|
| blue    | matching data        |
| yellow  | updated data         |
| green   | unchanged data       |
| red     | deletions            |
| blurred | transform is invalid |

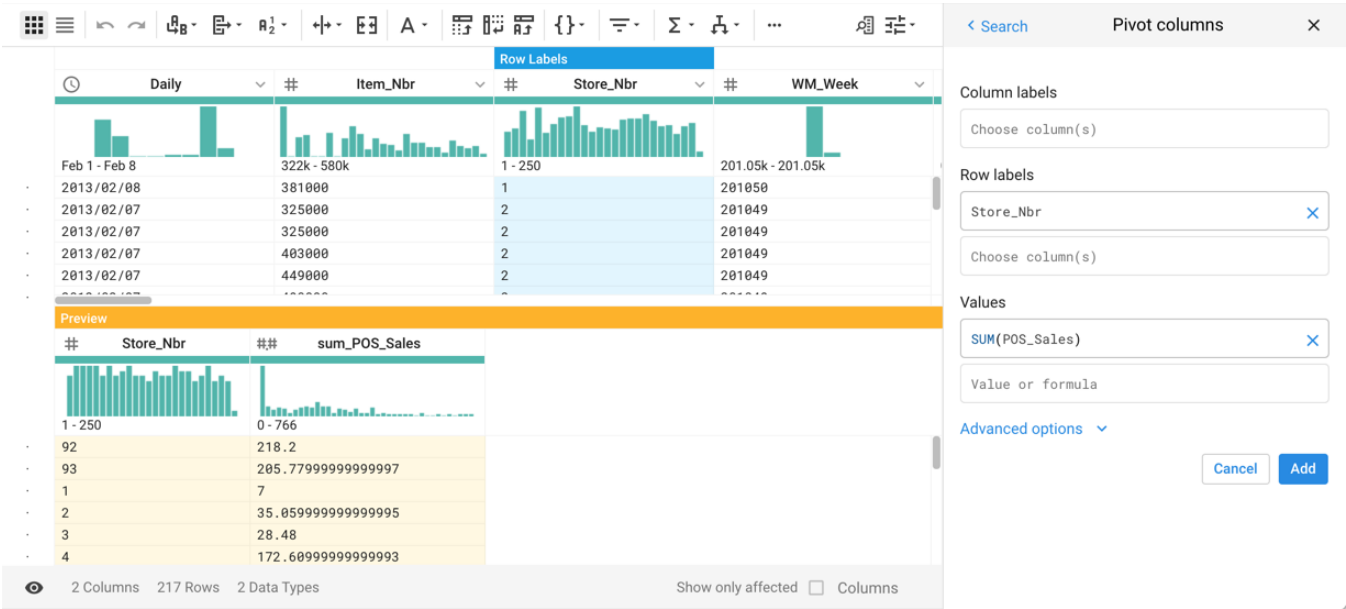
An error might appear while you are typing the transform. Because the transform is not valid, the transform preview appears blurred and does not display the results of the transform.

### Variations:

Previews vary depending on the type of transform. For example, if you create a transform to replace values in a column, the preview displays the results of that transform. If you create a transform to extract values from a column, the preview displays the values that will be extracted.

**Split pane previews:** For transforms that change the number of rows and columns, the results of the transform are previewed in a split pane view in the data grid and in the column browser. The following transform types are previewed in a split pane:

- pivot
- deduplicate
- unpivot



**Figure: Split Pane Preview**

# Context Panel

On the right side of the Transformer page, the context panel displays one of multiple panels, depending on the current state or selection of the data grid.

The following panels may be displayed within the context panel.

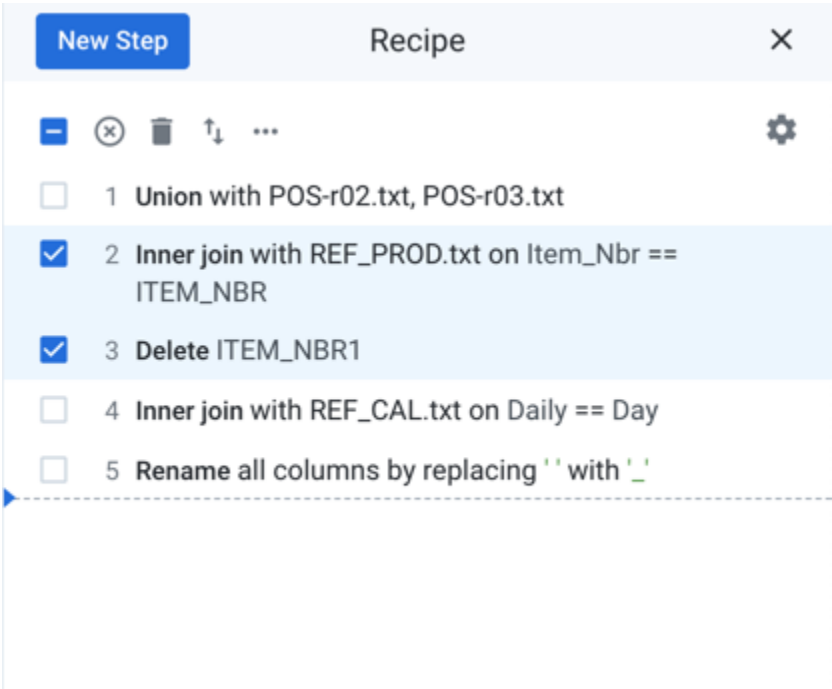
To close the context panel, click the X icon in the upper-right corner.

# Recipe Panel

**Contents:**

- *Recipe Toolbar*
  - *Toolbar context menu*
  - *Recipe Options*
- *Step Options*
  - *Single-step options*
  - *Multi-step options*

Through the Recipe panel, you can review and modify the steps of the recipe that you have already created and add new steps to your recipe at the current location. You can also flag a step that requires review without which jobs cannot be run.



**Figure: Recipe panel**

In the Recipe panel, the dotted horizontal line indicates the state of what is displayed in the data grid.

- To add a new step at the cursor location, click **New Step**. See *Search Panel*.
- This cursor can be moved. Select where you'd like to move to display in the data grid. Your selection can be a step or between two steps. Then, from the context menu in the Recipe panel, select **Go to selected**. Details are below.

**Tip:** You can undo and redo changes to your recipe through the transformer toolbar. See *Transformer Toolbar*.

**Select Steps:**

You can also select one or more steps to move or modify.

- To select a step, click the step or its checkbox to the left.
- Select other step checkboxes to add to your selection.
- You can toggle selecting all steps at the top of your recipe.

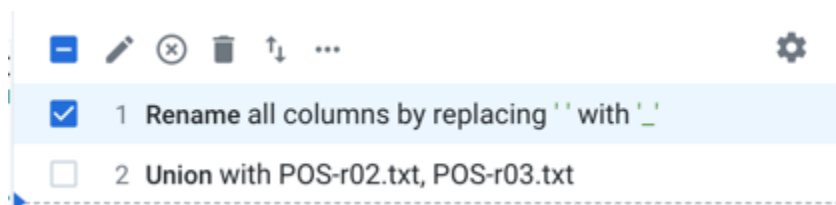
When you select one or more steps, the recipe toolbar is displayed. See below.

#### Keyboard Shortcuts:

| Shortcut             | Description                                                                  |
|----------------------|------------------------------------------------------------------------------|
| COMMAND/CTRL + click | Toggle selection of a step without changing status of other selected steps.  |
| SHIFT + click        | Select the range of steps between the current one and the last selected one. |

## Recipe Toolbar

When you select one or more steps in your recipe, the recipe toolbar is displayed above your steps.



**Figure: Recipe toolbar**

#### Tools:

- **Select/Deselect checkbox:** Select or deselect all steps in your recipe.
- **Edit:** (Single step selected only) Edit the recipe step.
- **Disable/Enable:** Disable the selected step or steps.
- **Delete:** Delete the selected step or steps.
- **Move:** Move the selected steps to the start or end of the recipe or up or down one step in the recipe.

**Moving a recipe can cause steps to break. Some fixups may be required.**

- **Recipe toolbar context menu:** See below.
- **Recipe options:** See below.

#### Toolbar context menu

The following options are available in the toolbar context menu.

**NOTE:** Some of these options may not be available depending on the selected step or whether you have selected multiple steps.

- **Go to:**



- **Selected:** Move the recipe cursor to the selected step. The data grid is updated to display the dataset up to the selected step in the recipe.
- **Start/End:** Go to the first or last step of the recipe.
- **Insert after step:** Insert a new step after the selected one. See *Search Panel*.
- **Duplicate:** Create a copy of the selected step and insert it after the selected one.

**Tip:** You can modify a duplicated step to create variations of your steps.

**NOTE:** Some steps, such as union or join operations, cannot be duplicated.

- **Cut:** Cut the selected step to the clipboard.
- **Copy:** Copy the selected step to the clipboard.

**NOTE:** Some steps, such as union or join operations, cannot be copied.

- **Paste after step:** Paste the step in the clipboard after the selected step. See Notes below.
- **Create macro:** You can create an independent object called a macro out of a selection of steps, which can be applied in other recipes. See *Create or Replace Macro*.

**NOTE:** Some types of steps cannot be included in macros. For more information, see *Overview of Macros*.

### Notes on pasting:

- If you are accessing the application over secure HTTPS, you may be prompted to authorize permission to paste. Some browsers store copied data in the global clipboard for the computer, which causes this security warning to be displayed.
- If you are accessing the application over less-secure HTTP, pasting in Wrangle text that was modified outside of the application is not supported.

### Recipe Options

The following options are available from the Gear menu for any number of selected steps.

- **Display Wrangle /natural language.** Toggle between displaying recipe steps in native Wrangle or in more readable language (default).
- **See Edit History.** Display history of recipe edits by user. See *Edit History Panel*.
- **Download Recipe as Wrangle .** You can download the recipe as text for offline review and storage.

**NOTE:** Wrangle recipes are stored in the Trifacta® database. You cannot upload new or modified recipes to the platform.

- **Download Sample data as CSV.** Download the sample currently displayed in the Transformer page in CSV format.

**Tip:** The downloaded CSV reflects the sample modified up to the currently selected recipe step, so you can use this to acquire and review data transformation in progress. For more information, see *Take a Snapshot*.

## Step Options

**NOTE:** Some of these options may not be available depending on the selected step or whether you have selected multiple steps.

### Single-step options

When you mouse over a step, you can choose to edit or remove the step or perform other operations from the context menu on the right side of the step. You can also right-click the step to open the menu.

**NOTE:** If you go to a step before the latest one in your recipe, the data grid is updated to reflect the state of the sample at that time. All subsequent steps are grayed out in the Recipe panel. When you run a job, all steps in the entire recipe are executed.

**NOTE:** In a dataset that is shared, multiple users cannot make changes to the recipe at the same time.

- **Go to selected:** Move the recipe cursor to the selected step.
- **Edit:** Edit the step.
- **Disable/Enable:** Disable or enable the selected step.
- **Delete:** Delete the selected step.
- **Move:** Move the step to the start or to the end of your recipe. Or, you can move it up or down one step at a time in the recipe.

**Moving or deleting a recipe or disabling/enabling steps a recipe can cause steps to break. Some fixups may be required.**

- **Insert new step after:** Insert a new step after the selected step. See *Search Panel*.
- **Duplicate:** Create a copy of the selected step and insert it after the selected one.

**Tip:** You can modify a duplicated step to create variations of your steps.

**NOTE:** Some steps, such as union or join operations, cannot be duplicated.

- **Cut:** Cut the selected step to the clipboard.
- **Copy:** Copy the selected step to the clipboard.

**NOTE:** Some steps, such as union or join operations, cannot be cut or copied.

- **Paste after:** Paste the step in the clipboard after the selected step.
- **Create macro:** See previous.

### Flag for review

The Flag for review option enables you to flag a step in the recipe for review. When you flag a step for review, a warning icon is displayed for the corresponding step, alerting flow users that the step must be reviewed and cleared of the flag before running the job.

**NOTE:** This feature may need to be enabled in your environment. For more information, see *Flag for Review*.

**NOTE:** When a step is flagged for review, all downstream steps are disabled, and you cannot run the job until all flagged steps are reviewed. Steps must be reviewed in descending, top-to-bottom order. For more information, see *Flag for Review*.

#### Actions:

- **Flag for review:** Marks the current step as pending review.
  - Select the Flag for review option, add a name and description, and click **Flag**.
  - A warning icon is displayed against the corresponding step.
- **Unflag for review:** Removes the flag from the step, unblocking review.
- **Rename review step:** Edit the name and description of the flag.
- **Mark as reviewed /Mark as pending review:** You can toggle between these options to mark the review as complete or to mark the step as pending review. After you select Mark as reviewed, a tick mark is displayed against the reviewed step.

For more information, see *Flag for Review*.

#### Multi-step options

If you select multiple steps in your recipe, the following options are available:

- **Disable/Enable:** Disable or enable the selected steps.
- **Delete:** Delete the selected steps.
- **Move:** Move the steps to the start or to the end of your recipe. Or, you can move it up or down one step at a time in the recipe.

**Moving a recipe can cause steps to break. Some fixups may be required.**

- **Duplicate:** Create copies of the selected steps and insert them after the selected one.
- **Cut:** Cut the selected steps to the clipboard.
- **Copy:** Copy the selected steps to the clipboard.
- **Paste after:** Paste the steps in the clipboard after the selected step.
- **Create macro:** See previous.

# Transform Builder

## Contents:

- *Step 1 - Select transformation in the Search Panel*
  - *Step 2 - Specify the column(s), formula, or condition*
    - *Columns*
    - *Patterns*
    - *Delimiter Groups*
    - *Condition*
  - *Step 3 - Grouping, Ordering, and Naming*
  - *Step 3 - Specify other parameters*
  - *Step 4 - Add the step*
  - *Edit a transform*
- 

The Transform Builder enables you to search for transformations and to rapidly assemble complete transform steps through a simple menu-driven interface.

After you select the transformation to apply, all relevant parameters can be configured through selection or type-ahead fields, so that you can choose from only the elements that are appropriate for the selected transformation.

To open the Transform Builder, begin creating a step through one of the following methods:

- Select a transformation from the Transformer toolbar. See *Transformer Toolbar*.
  - Click the Macros icon in the toolbar to apply a macro as your next step. See *Apply a Macro*.
- Select a transformation from a column menu. See *Column Menus*.
- Search for and select a transformation in the Search panel. See *Search Panel*.
- Click **New Step** in the Recipe panel. See *Recipe Panel*.
- Edit an existing step.

[< Recipe](#)

New formula

×

Formula type

required

Single row formula

Create a new column from a single row formula

Formula

required

SUM(POS\_Sales)

New column name

sumSales

Cancel

Add

## Figure: Transform Builder

### Keyboard shortcuts:

**Tip:** When keyboard shortcuts are enabled, press ? in the application to see the available shortcuts. Individual users must enable them. See *User Profile Page*.

## Step 1 - Select transformation in the Search Panel

From the Search panel, begin typing to see the list of available transformations. Select your preferred one.

Join and union transformations have dedicated pages for configuring this transformations. You can enter `join` datasets or `union` as the search term to open the corresponding tool:

- See *Join Window*.
- See *Union Page*.

For a list of available transformations, see *Transformation Reference*.

## Step 2 - Specify the column(s), formula, or condition

Depending on the transform that you have selected, you must specify one or more of the following in the Transform Builder.

- Some transforms support combinations of the following.
- Some transforms, like `deduplicate`, require no parameters.

The following are general categories of object types:

- **Literal values.** A literal, or constant, value is a fixed numeric, string, Boolean, or other type of value, which does not change depending on the row under evaluation.
- **Functions.** Designer Cloud powered by Trifacta® Enterprise Edition supports a wide variety of numerical, statistical, and other function types. For a list of available transforms and functions, see *Wrangle Language*.
- **Columns.** When a column name is used in a formula, the transform uses the value in the named column for the currently evaluated row.
- **Operators.** You can apply logical, numeric, or comparison operators as part of your formula.
  - See *Logical Operators*.
  - See *Numeric Operators*.
  - See *Comparison Operators*.
- **Parameters:** Add a reference to a flow parameter in your transformation. See *Manage Parameters Dialog*.
- **Metadata.** You can insert special strings that evaluate to references of your dataset's metadata. For more information, see *Source Metadata References*.

### Columns

Using the Columns parameter, you can select or specify the column or columns to which to apply the transform.

The following options are available when specifying one or more columns in a transformation:

- **Multiple:** Select one or more discrete columns from the drop-down list.
- **All:** Select all columns in the dataset.
- **Range:** Specify a start column and ending column. All columns inclusive are selected.
- **Advanced:** Specify the columns using a comma-separated list. You can combine multiple and range options under Advanced. Ranges of columns can be specified using the tilde (~) character. Example:

Store\_Nbr, Item\_Nbr, WM\_Week~POS\_Cost

## Patterns

For some transforms, you can specify patterns to identify conditions or elements of the data on which to take action. These matching patterns can be specified using one of the following types.

| Pattern Type       | Description                                                                                                                                                                                                                                                       | Example                                                                                                                |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------|
| Literal value      | An exact string or value.                                                                                                                                                                                                                                         | The following matches on the exact value between the quotes:<br><br><code>'This is what I want to match.'</code>       |
| Pattern            | Designer Cloud powered by Trifacta Enterprise Edition supports a variety of macro-like pattern identifiers, which can be used in place of more complex regular expressions.                                                                                       | The following matches when two digits appear at the beginning of a value:<br><br><code>`{start}{digit} {digit}`</code> |
| Regular expression | Regular expressions are a standard method of describing matching patterns.<br><br><b>NOTE:</b> The syntax of regular expressions can be complex and can lead to unexpected results if they are improperly specified. Regex is considered a developer-level skill. | The following matches on all numerical values from 0 to 99:<br><br><code>/^\d\$ ^\d\d\$/</code>                        |

For more information on pattern-based matching, see *Text Matching*.

**Flow parameter:** You can also insert a flow parameter into your pattern-based inputs in the Transform Builder. To reference a flow parameter, click the Parameterize icon above any field that accepts pattern-based inputs.

- The parameter values or any overrides applied to those values are applied to the results displayed in the data grid, as well as during job execution.
- For more information on creating flow parameters, see *Manage Parameters Dialog*.
- For more information on parameterization, see *Overview of Parameterization*.

## Delimiter Groups

In the Transform Builder, transforms that require delimiter are organized into delimiter groups, so that you specify only the elements of a pattern that work together. Delimiter groups apply to the following transforms:

- *Countpattern Transform*
- *Extract Transform*
- *Replace Transform*
- *Set Transform*
- *Split Transform*

Delimiter groups are listed below.

| Delimiter group    | Description                                                                                               |
|--------------------|-----------------------------------------------------------------------------------------------------------|
| On delimiter       | Transformation is applied based on a specific literal or pattern.                                         |
| Between delimiters | Transformation is applied on database between two literal or pattern-based delimiters. Details are below. |
| On multiple        |                                                                                                           |

|                     |                                                                                                                                                                                                                             |
|---------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| delimiters          | Transformation is applied based on a sequence of delimiters. An individual pattern can be a string literal, <code>Pattern</code> , or regular expression, and the sequence can contain combinations of these pattern types. |
| Between positions   | Transformation is applied based on a starting index position and an ending index position. Index positions start from 0 on the left side of any cell value.                                                                 |
| On positions        | Transformation is applied based on a sequence of listed index positions. Index positions start from 0 on the left side of any cell value.                                                                                   |
| At regular interval | Transformation is applied at every <i>nth</i> position. Index positions start from 0 on the left side of any cell value.                                                                                                    |

For more information on the underlying syntax for delimiter groups, see *Pattern Clause Position Matching*.

### Between two delimiters

Matches any values that appear between two delimiters. One delimiter describes the beginning of the match, and the other delimiter describes the end of the match.

Each delimiter can either include or exclude the matching value:

| Transform Builder option | Include as part of transform | Include/Exclude      |
|--------------------------|------------------------------|----------------------|
| Start delimiter          | false                        | Excludes sub-pattern |
| Start delimiter          | true                         | Includes sub-pattern |
| End delimiter            | false                        | Excludes sub-pattern |
| End delimiter            | true                         | Includes sub-pattern |

### Condition

A **condition** is an expression that yields a `true` or `false` value. A condition may include all of the elements of a formula. This value determines whether the transformation is applied to the evaluated row.

## Step 3 - Grouping, Ordering, and Naming

A number of transforms support the following parameters.

**NOTE:** Transforms that use the `group` parameter can result in non-deterministic re-ordering in the data grid. However, you should apply the `group` parameter, particularly on larger datasets, or your job may run out of memory and fail. To enforce row ordering, you can use the `sort` transform. For more information, see *Sort Transform*.

**Group parameter:** For transforms that aggregate data, such as `pivot` or `window`, you can specify the column by which you wish to group the computed aggregations. In the following example, all values in the `POS_Sales` column are summed up for each value in the `Store_Nbr` column.

|                       |                |
|-----------------------|----------------|
| Transformation Name   | Pivot columns  |
| Parameter: Row labels | Store_Nbr      |
| Parameter: Values     | sum(POS_Sales) |

Assuming that there are entries in the `Store_Nbr` column, the resulting transform step has 50 rows, each of which contains the total sales for the listed store number.

**Order parameter:** Some transforms support the `order` parameter, which allows you to specify the column of values that are used to sort the output. In the following example, all aggregates `Sales` values are ordered by the contract date and grouped by `State`:

|                          |               |
|--------------------------|---------------|
| Transformation Name      | Pivot columns |
| Parameter: Row labels    | Store_Nbr     |
| Parameter: Column labels | contractDate  |
| Parameter: Values        | sum(Sales)    |

The output can always be ordered using the `sort` transform. See *Sort Transform*.  
**New Column Name parameter:** For transforms that generate new columns, such as `derive` and `extract`, you can optionally specify the name of the new column, which saves adding a step to rename it. In the following example, the values of `colA` and `colB` are summed and written to the new column `colC`:

|                            |                    |
|----------------------------|--------------------|
| Transformation Name        | New formula        |
| Parameter: Formula type    | Single row formula |
| Parameter: Formula         | colA + colB        |
| Parameter: New column name | colC               |

### Step 3 - Specify other parameters

Depending on the transform, you may be presented with other required or optional parameters to specify. See *Transforms*.

### Step 4 - Add the step

When you have finished your transform step, review the preview in the data grid.

If the results look ok, click **Add**.

The step is added to your recipe and applied to the data grid.

- See *Data Grid Panel*.
- See *Transform Preview*.

### Edit a transform

After you have added a step, you can modify it as needed. In the Recipe panel, select the Pencil icon next to the recipe step. The step is displayed for editing in the Transform Builder.



# Search Panel

Through the search context panel, you can locate transformations to specify and add at the current location in your recipe.

You can search for transformations to add in any of the following ways:

- At the top of the Transformer page, click the Magnifying Glass icon.
- When you choose to add a new step to your recipe, the Search panel opens in the context panel.
- To add a new recipe from anywhere in the Transformer page, press `CTRL/COMMAND + K`. Enter a search string for your transformation step.

Enter text or browse the available transformations to begin building your next step.

**Tip:** When you enter a search term, you can choose to use that term to search the product documentation. Select the **Search documentation** entry.

<

Search Transformations

×

🔍 Search...

Formulas

Scale to min max

Scale a column to a specific min max range

One hot encode

Create a column for each unique value indicating its presence ...

Scale to mean

Scale a column to zero mean and unit variance

Bin column

Bin values into ranges of equal or custom size

New formula

Create a new column from the result of a formula

Edit with formula

Set one or more columns to the result of a formula

Window

Perform calculations across multiple ordered rows

Schema

Change column type

Change the data type of a column

### **Figure: Search Panel**

To locate transformations, you can browse or search.

#### **Browse:**

- Headings like **Formulas** indicate categories of transformations.
- For each transformation, you can review a brief description of it.

#### **Search:**

- Enter a few characters of a transformation, function or other object such as a metadata reference for which you are looking. Matches are underlined in the panel.
  - To see a list of all available functions, enter `function`. When selected, a New formula transform is pre-specified using the selected function.
- A copied step can be pasted back into the Search panel and modified from the Transform Builder. Copy and paste may not be supported across different releases of the product.
- If you are familiar with Wrangle transforms, you can enter the transform name in the search bar. For example, type `window`.

**Tip:** At the bottom of your search results, you can click the **Search documentation** link to search the product documentation for the term you entered.

After you have selected the transformation to build, the Transform Builder is pre-populated with some configuration done for you, so you can begin specifying the transformation. For more information, see *Transform Builder*.

# Edit History Panel

Through the Edit History panel, you can review the sequence of edits to the current recipe by individual contributors. This panel assists in determining who made which changes and when they were made.

- If the dataset is part of a shared flow, edits appear created by each user.
- If the dataset has been sent as a copy, all steps made by the user who shared the flow appear as a single edit.
- If the flow containing the dataset has not been shared, the only user listed in the Edit History is the owner of the flow.
- For more information on sharing, see *Overview of Sharing*.
- To open the Edit History panel, select **See Edit History** in the Recipe panel. See *Recipe Panel*.

< Recipe

Edit History

×

Administrator updated today

–

 Delete WM\_Week

SteveO updated today

+

 Delete WM\_Week

–

 Delete Store\_Nbr

+

 Delete Store\_Nbr

SteveO updated today

+

 Delete Day

Administrator updated today

+

 Inner join with REF\_CAL.txt on Daily == Day

+

 Delete ITEM\_NBR1

+

 Inner join with REF\_PROD.txt on Item\_Nbr == ITEM\_NBR

+

 Union with POS-r02.txt, POS-r03.txt

**Figure: Edit History panel**

Sets of changes are grouped by the user who performed them and listed with most recent changes at the top.

Changes to individual transforms are listed within a set:

| Icon | Description |
|------|-------------|
|      |             |

|   |                                                 |
|---|-------------------------------------------------|
| + | The listed step was added (add operation).      |
| - | The listed step was deleted (delete operation). |

Edit operations are represented by a delete operation and then an add operation.

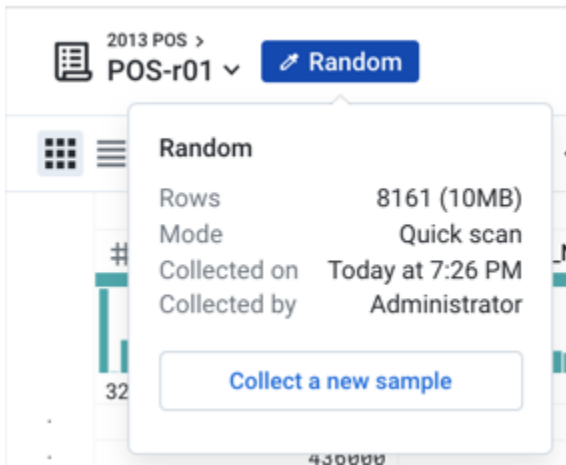
# Samples Panel

## Contents:

- *Collect new sample*
- *Collected samples*
- *Cancel sample jobs*

For smaller datasets, the Transformer page displays the entire dataset. For larger ones, the source data is sampled for use in the Transformer page. Through the Samples panel, you can create new samples and select them for display in the Transformer page.

At the top of the Transformer page, the type of the current sample is displayed next to the dataset name. To open the Samples panel, click the current sample indicator:



**Figure:** Click the current sample button.

In the example above, you can see that the current sample is a Random sample.

- **Full Data:** The entire dataset is small enough to be displayed in the data grid. The data is not sampled.
- **Initial Sample:** The sample is taken from the first set of rows in the first file or table that is part of the dataset. Data from the rest of the first file or table or from other files or tables is not included in the data grid.

**Tip:** For purposes of loading the data, the initial sample is generated and displayed at first. For a better representation of the entire dataset, you should create a new sample.

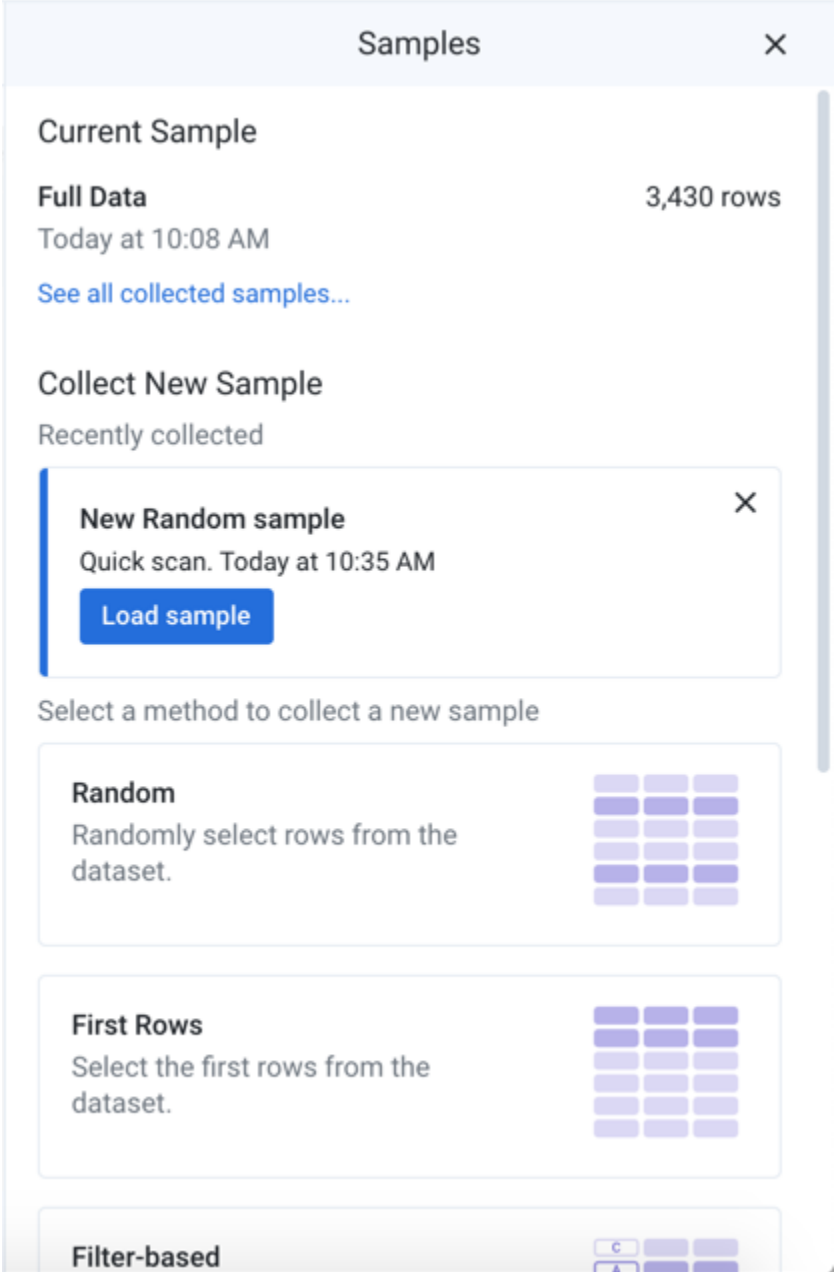
- If the recipe is a child recipe, then the Initial Data sample indicates the selected sample of the parent recipe.

To create a new sample, click **Collect a new sample**.

The Samples panel is displayed on the right side of the screen:

**Tip:** You can also open the Samples panel by clicking the Eyedropper icon at the top of the page.

To review all samples that you have created, see *Sample Jobs Page*.



**Figure: Samples Panel**

**Current sample:**

At the top of the panel, you can review the currently loaded sample. Each user has his own active sample on a dataset.

**NOTE:** When a new sample is generated, any Sort transformations that have been applied previously must be re-applied. Depending on the type of output, sort order may not be preserved.

- **Initial:** By default, the application loads the first N rows of the dataset as the initial sample when the Transformer page is opened. The number of rows depends on column count, data density, and other factors. If the dataset is small enough, the full dataset is used.

**NOTE:** By default, samples may be up to 10 MB in size. For datasets smaller than this limit, the entire dataset is loaded.

- Click the link in the current sample card to see the list of all available samples.

**Tip:** To change the name of a sample, click its card in the list of all available. Then, click the Edit icon.

### New samples:

Below the current sample, you can review the available options for creating new samples. Each type of sample reflects a different method of collection.

**The data that is displayed in the data grid is based on all of the upstream samples after which all subsequent steps in each upstream recipe are performed in the browser. If you have a large number of steps or complex steps between the recipe locations for your samples in use and your current recipe location, you may experience performance slow-downs or crashes in the data grid. For more information on sampling best practices, see <https://community.trifacta.com/s/article/Best-Practices-Managing-Samples-in-Complex-Flows>.**

- To collect a new sample, click the appropriate sample card. See below.

**NOTE:** If a sample fails to generate, you can retry or download logs for review. Click the Download Logs link. These logs may be useful in debugging.

- To cancel a sample collection, click the X next to the progress bar. The interrupted sample is listed as unavailable. You can download the logs from the unfinished sample collection.
- After a sample is created, you can load it at any time, as long as it is still valid. Next to a collected sample, click **Load sample**.
- For more information on sampling methods, see *Overview of Sampling*.

### Status bar:

At the bottom of the Transformer page, you can review the number of rows and columns and count of data types in the currently displayed sample.

**NOTE:** As you add transformation steps to your recipe, the values in the status bar change to reflect the current state of the loaded sample.

**NOTE:** Some operations, such as `union`, may change the row counts without invalidating the sample. If the operation increases the size of the dataset beyond the sample size limit enforced by the application, then a subset of those rows is displayed. This is a known issue.

### Collect new sample

When a new sample is collected, it is gathered based on the current location in the recipe when the sample is gathered. So, if the recipe contains steps that join in other datasets, those joins are performed to bring together the data from which the sample is executed.

< Collect new sample X

Name

Random

Scan required

✓ Quick

Full

Cancel Collect

**Figure: Collect new sample panel**

**NOTE:** Except for the initial sample, all samples are generated based on the steps leading up to the location of the cursor in the recipe. If earlier steps are deleted or modified, the collected sample can be invalidated.

**NOTE:** When sampling from compressed data, the source is uncompressed, and a new sample of it is loaded into the data grid. As a result, the sample size you see in the grid corresponds to the uncompressed data.

#### Steps:

- In the Samples panel, select the type of sample to create. For more information on sample types, see *Overview of Sampling*.
- In the Collect new sample panel, specify the following parameters, some of which may not be required for your sampling method:
- **Choose a sampling method:** Select or enter the type of sample. If you already selected a sampling method, this value is pre-populated for you.
- **Name:** You can enter a new name of the sample as needed.

**Tip:** Naming your samples can assist in tracking them later. For example, you might choose to add a date stamp to the name to track when you captured the sample.

- **Scan Type:** (Does not apply to all sampling methods) Types of scans:
  - **Quick** - performs a random scan of the dataset to extract the appropriate number of rows for the sample.
  - **Full** - gathers the sample from the entire dataset. Depending on the size of the dataset, this method can take a while.



- **Use latest data:** When collecting a Full Scan sample from a JDBC source and performance ingest caching has been enabled, you can choose to override the cached data and to gather all of your data from the original sources.

**NOTE:** If the cached data has expired, the sample is always collected from the original sources, even if this option is not selected.

Click **more details** to review the list of datasets whose cached data will be overridden.

Ingest caching applies to non-native relational (JDBC) sources. For more information, see *Configure JDBC Ingestion*.

- **Column or columns:** (Stratified, Cluster-based) Name of the column from which to gather values to evaluate (Anomaly-based) Specify the name or names of one or more columns containing the anomalies to include in your sample. Multiple columns can be specified by comma-separated values. A column range can be specified using the tilde (~) character.
- **Condition:** (Filter-based, Stratified, Cluster-based, Anomaly-based) Filter the sample based on a specified condition. For example:

```
invoiceDate > 90
```

- **Anomaly type:**(Anomaly-based) Select the type of anomalous values to include in your sample: invalid, missing, or both types.
- **Variable overrides:** If one or more variables is associated with your dataset, you can define the value overrides to be applied when the sample is executed.
  - You can use these overrides to sample data from different source files in your dataset with parameters.
  - A variable can have an empty value.
  - For more information, see *Overview of Parameterization*.
- To begin collecting the sample, click **Collect**.
- You can continue working while the sample is collected. When the sample is available, a status message is displayed in the Transformer page.
- You can click **Load Sample** in the Samples panel to begin using it.

## Collected samples

In the Collected samples panel, you can review the available and unavailable samples. If applicable, you can review the variable override values that were applied during the sampling.

To use one of the available samples, select its card. The sample is loaded in the data grid.

**NOTE:** If you add recipe steps that change the number of rows in your dataset (or a few other edge case steps), some of your existing samples may no longer be valid. When you execute a join, union, or delete action or edit steps before this action, you may be prompted with the Change Recipe dialog, which includes the following message:

**Your change will invalidate some of the currently available samples for this source. The invalid samples will be deactivated.**

For more information on the types of transformations that can invalidate samples, see *Reshaping Steps*.

## Cancel sample jobs

You can cancel a sample job that is currently being executed.

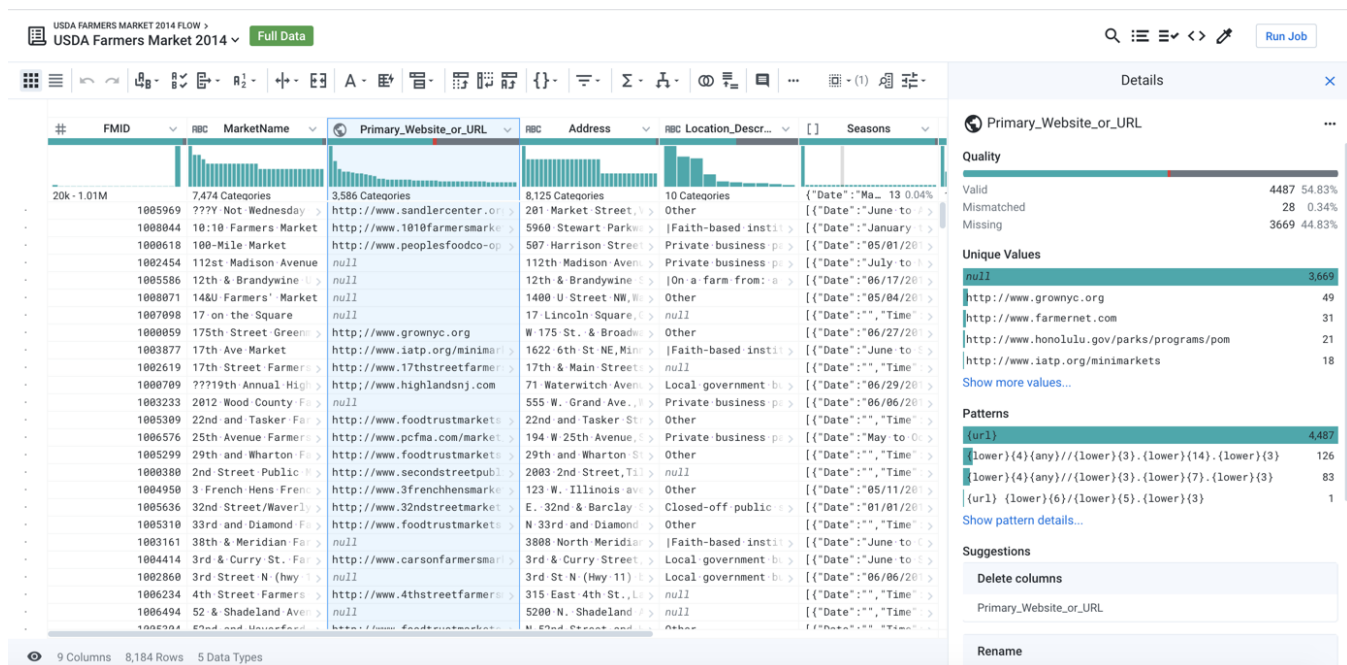
- In the Samples panel, locate the job in-progress. Click the X.
- You can also review and cancel sample jobs through a page in the Designer Cloud application . For more information, see *Sample Jobs Page*.

# Selection Details Panel

## Contents:

- *Select Column*
  - *Data Quality Bar*
  - *Unique Values*
  - *Patterns*
- *Other Selections*
  - *Select Multiple Columns*
  - *Select Column Values*
  - *Select Values in the Data Grid*
- *Suggestions*

In the Selection Details panel, you can review an active profile of your current selection or selections in the data grid or column browser and review patterns and suggestions for transformations.



**Figure: Selection Details Panel for selected column**

Based on what you have selected, the panel is updated with context-specific information about your selection(s) and a set of actions that you can take on the data.

## Select Column

When you select a column, the following sections appear in the Selection Details Panel.

- To change the data type of the column, click the menu to the left of the column name in the panel.
- From the caret to the right, you can make selections from the column menu. Available selections may vary by column data type.
- For more information, see *Column Menus*.

## Data Quality Bar

Review the counts and percentages of valid, mismatched, and missing values in the column.

- Click one of the colored bars to select only the matching values in the column. See [Select Column Values](#) below.

### Context menu:

Right-click the data quality bar to see a set of possible transformations:

- **Keep rows** - Keep rows that match the data quality bar you selected.
- **Delete rows** - Delete rows that match the data quality bar you selected.
- **Create new column flagging** - Create a new column containing `true` for each row that matches the data quality bar you selected. Otherwise, the row value in the new column is `false`.
- **Clear values** - For mismatched or empty rows, you can set the value to be empty.
- **Replace values** - For mismatched or empty rows, you can replace with a specific value. See [Replace Cell Values](#).

## Unique Values

You can review the counts of the most frequently occurring values in the column.

To see all unique values, click **Show more values**.

- Use the Search bar to locate specific values among the list of unique ones.
- Click the Back button to return to the Selection Details panel.

### Context menu:

Right-click any value bar to be prompted for a set of transformations specific to the applicable rows and values.

- **Keep rows with selected values** - Keep rows where the selected value appears. Delete the other rows.
- **Delete rows with selected values** - Delete rows where the selected value appears. Keep the other rows.
- **Create new column flagging** - Create a new column containing `true` for each row that matches the selected value(s). Otherwise, the row value in the new column is `false`.
- **Clear values** - You can set the row value to be empty for the selected value(s).
- **Clear others** - You can set the row value to be empty for all rows that do not match the selected value(s).
- **Replace values** - You can replace the selected value(s) with a specific value. See [Replace Cell Values](#).

## Patterns

Based on your selected column or column values, Designer Cloud powered by Trifacta Enterprise Edition attempts to find patterns that match your selections. These patterns are represented as Trifacta patterns. For more information, see [Text Matching](#).

To see all patterns, click **Show more patterns**. For more information, see [Column Details Panel](#).

### Context menu:

Right-click any pattern bar to be prompted for a set of transformations specific to the rows and values that match the pattern.

- **Keep rows with selected patterns** - Keep rows with values that match the selected pattern(s). Delete the other rows.
- **Delete rows with selected patterns** - Delete rows with values that match the selected pattern(s). Keep the other rows.

- **Create new column flagging** - Create a new column containing `true` for each row that matches the selected pattern(s). Otherwise, the row value in the new column is `false`.
- **Clear values matching patterns** - You can set the row value to be empty for the selected pattern(s).
- **Clear others** - You can set the row value to be empty for all rows that do not match the selected pattern(s).
- **Replace values with matching patterns** - You can replace the selected pattern(s) with a specific value. See *Replace Groups of Values*.

## Other Selections

### Select Multiple Columns

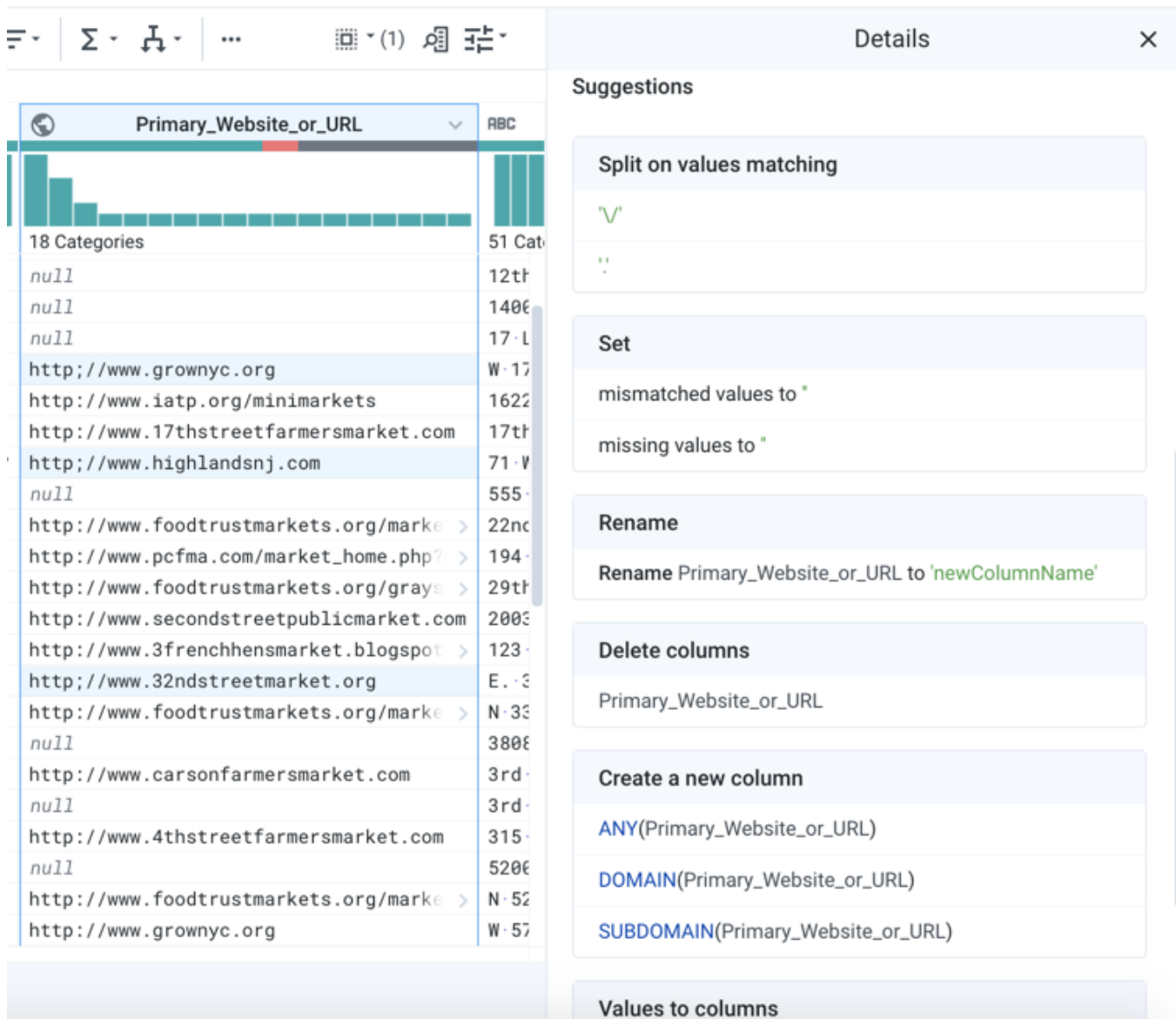
When you select multiple columns, the following changes to the panel apply:

- Profiling of the data is not available, which also means that you cannot take action on individual values within your selection.
- The data type menu is no longer available.

You can access a column menu and review suggestion cards that are applicable to all of your columns.

### Select Column Values

After you have selected a column, you can use the Selection Details panel to select individual values within the column, which updates the panel. Below, the mismatched values in the previously selected column have been selected from the data quality bar:



**Figure: Selection Details Panel - selected column values**

Review and make your choice among the available selections.

## Select Values in the Data Grid

When you select values in the data grid, the Selection Details panel presents a set of suggestions for your review. See below.

## Suggestions

Suggestion cards provide a means to select relevant suggestions for transform steps. The suggestions vary depending on the data you have selected. You can then use the cards to preview the results in the data grid, so that you are confident that the proposed transformation works for your dataset.

**Tip:** As you mouse over areas of the Transformer page and its panels, the Lightbulb icon appears next to the cursor to indicate that suggestions are available. Select the data to see the suggestion cards.

In the following example, your dataset contains a column of addresses. Within one of the values, you can select a zip code, which then triggers an appropriate set of suggestion cards:

**Figure: Suggestion Cards**

In the above image, the first suggestion in the Extract values matching suggestion card has been selected by default, and its parameters have been specified to extract all zip codes from the source column (Address). While useful, this selection may not be your intention. Options:

1. Hover over different suggestions. A mini-preview appears to the left of the suggestion.
2. Click a different suggestion in the same card. In this case, you might click the first suggestion in the Replace card, so that you can remove zip code.

**Tip:** Optionally, suggestions can be provided to you based on your prior transformations or the transformations of other users in your workspace. These suggestions appear under the Recently used heading. For more information on enabling collaborative suggestions, see *Workspace Settings Page*.

**Tip:** Among the suggestion cards, scroll down to see other suggestion cards that are off-screen. If it is present, a See all link displays more suggestions in that card. Variants further down in the suggestion card typically become more specific in their changes to the dataset or rarer in their usage.

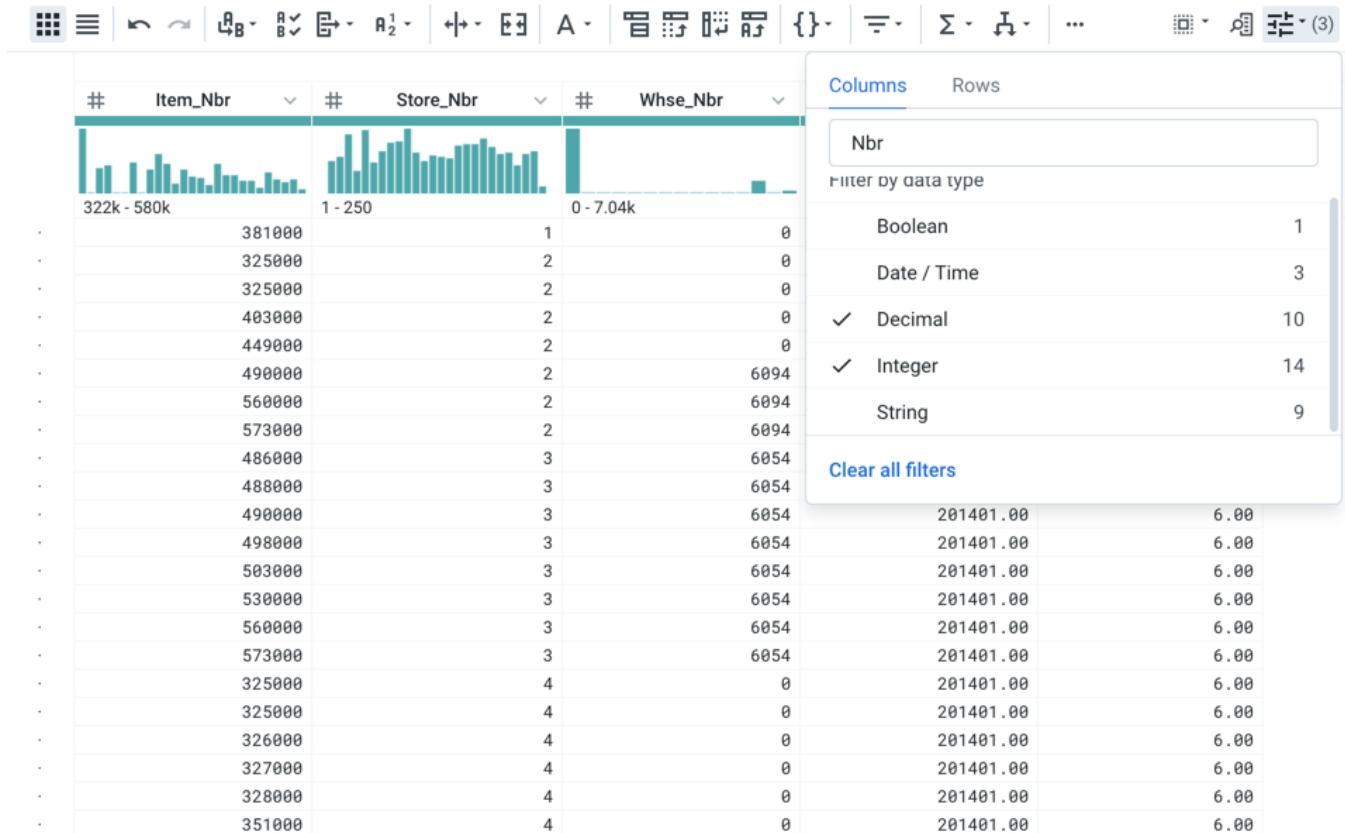
3. Modify the selected transform. To edit the step, click **Edit**. You can fine-tune parameters of the transformation. See *Transform Builder*.

## Actions:

- To apply a suggestion, select the data, and then choose the suggestion to apply from the list of cards. Click **Add**.
- If needed, you can customize the selected suggestion. Click **Edit**. See *Transform Builder*.
- To generate a new set of suggestion cards, click **Cancel**. Then, select a different set of columns or values within a column.

## Filter Panel

As needed, you can filter the rows and columns displayed in the data grid. To review and apply filters, click the Filter icon in the Transformer toolbar.



**Figure: Data Grid Filters**

**Tip:** Use data grid filters to assist in locating and selecting columns or values for selection and selection cards. After you have applied the suggested step to your recipe, remember to remove the filter on the data grid.

**NOTE:** Wildcards are not supported.

**Columns:** You can filter by column name values and data types.

- Enter a text string to immediately filter the display to show only columns with matching values.
- Click next to a data type to show columns of that type.
- Text and data type filters are additive. Both filters are applied to the display.

**NOTE:** If a column is hidden in the Visible Columns panel, it cannot be surfaced using a filter. You must toggle its display first. See *Visible Columns Panel*.



**NOTE:** Previewed columns are always displayed.

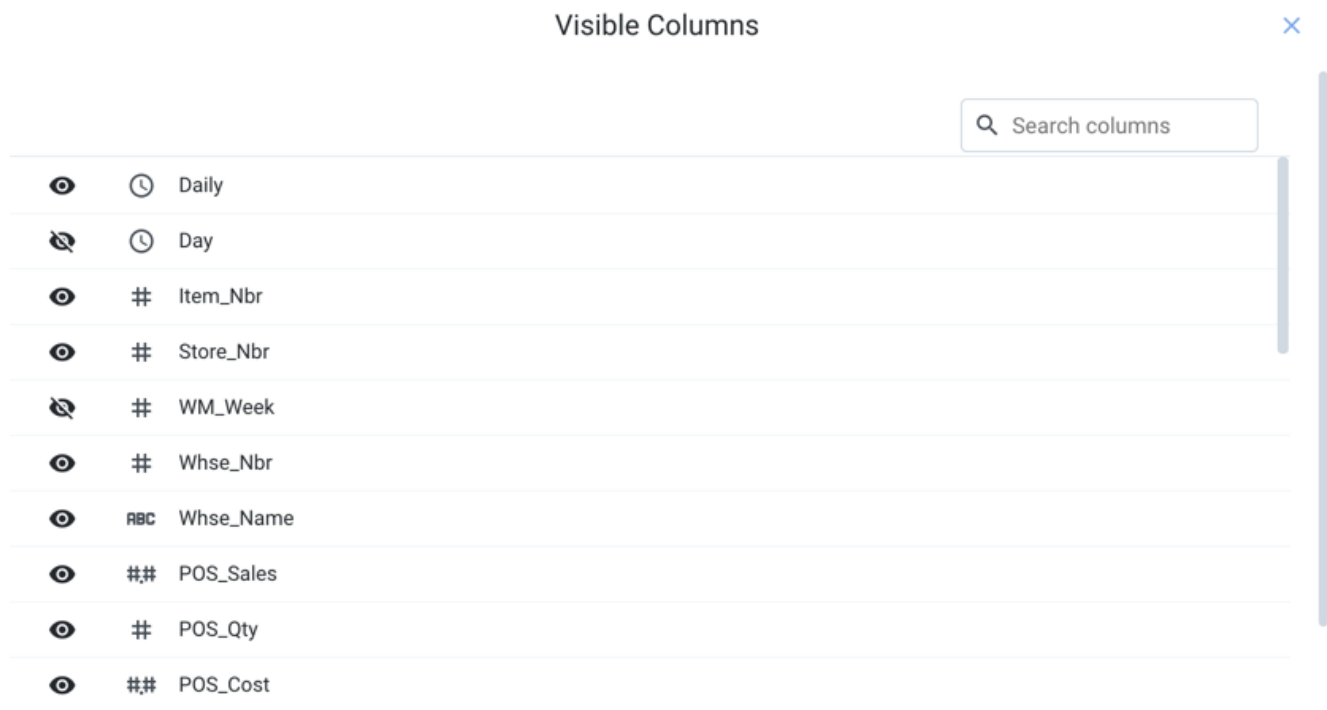
**Rows:** Enter values to highlight the rows where the values are present. Only rows where the values are present are displayed.

To remove all row and column filters, click **Clear all filters**.

For more information on the toolbar, see *Transformer Toolbar*.

# Visible Columns Panel

In the status bar of the Transformer page, click the Eye icon to review the list of visible and hidden columns.



**Figure: Visible Columns Panel**

- Click the Eye icon next to a column name to toggle its visibility in the Transformer page.

**NOTE:** Columns that are not visible in the Transformer page are still generated in the output file. Before you run a job, you should review the Visible Columns dialog.

**NOTE:** Filters applied to the data grid or column browser are also applied in this panel. For more information, see *Filter Panel*.

- To toggle display of multiple columns at the same time, use CTRL or SHIFT to select columns. Then, click the Selected link and choose to show or hide them.
- Use the Search box to find matches for column names.
- To close the dialog, click the X icon.

# Join Window

## Contents:

- *Before You Begin*
  - *Step 1 - Select Dataset or Recipe*
  - *Step 2 - Select Join Conditions*
  - *Step 3 - Select Output Columns*
    - *Advanced options*
  - *Step 4 - Review Join*
- 

In the Join window of the Designer Cloud® application, you can join your current dataset with another dataset or recipe based upon information that is common to both datasets.

For example, you could join together two sets of regional sales data based upon the product identifiers that they both use to track sales. In the Search panel, enter `join datasets` or select the Join icon from the toolbar.

- A **join** is a standard operation for merging the data from two different datasets. For more information, see *Join Types*.
- You cannot perform joins on columns of Object or Array data type.
- A join operation is different from a union operation. In a **union** operation, data from one or more datasets is appended to the current dataset, assuming that the columns are identical or very similar. For more information, see *Union Page*.

**Tip:** Depending on the types of operations you need to perform, you may need to perform joins earlier or later in your recipe. For more information, see *Optimize Job Processing*.

**NOTE:** Unnest, union, or join transforms may significantly increase the number of rows or columns in your dataset. To prevent overloading the browser's memory, the application may apply a limit function to the results to artificially limit the number of rows displayed in your sample. You can generate a new sample if desired. This limitation is not applied during the job execution.

## Before You Begin

- **Review your record counts.** Before you specify the join, you should review your record counts and the uniqueness of your keys, which should provide an idea of the number of records you may see in the output. Note that the number of output records depends on the type of join and the matches between join keys.
- **Review your join key values.** If there are variations in the values in your join keys, you may end up with duplicate records in your joined dataset. Look for mismatched or missing values in your join keys, and correct if possible.
- **Review the granularity of your data.** If you bring together data at a lower fidelity than the source, you can end up with record matches that are not actually matching data. For example, if your timestamps are down-sampled from milliseconds to seconds as part of the join, you may have "matching" timestamps in seconds that were not matches at the millisecond level in the source data.

## Step 1 - Select Dataset or Recipe

In the Search panel, enter `join datasets`. Then, select the dataset or recipe that you wish to join with your current dataset.

Choose dataset or recipe to join with POS-r01 – 2.txt

Search...

Recipes in current flow

Datasets in current flow

All datasets

| Name           | Last Updated      | Source | Data                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |   |          |     |              |        |  |  |                           |        |  |  |                           |        |  |  |                          |        |  |  |                        |        |  |  |                             |        |  |  |                             |        |  |  |                     |        |  |  |                   |
|----------------|-------------------|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---|----------|-----|--------------|--------|--|--|---------------------------|--------|--|--|---------------------------|--------|--|--|--------------------------|--------|--|--|------------------------|--------|--|--|-----------------------------|--------|--|--|-----------------------------|--------|--|--|---------------------|--------|--|--|-------------------|
| POS-r03.txt    | Today at 11:05 AM | HDFS   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |   |          |     |              |        |  |  |                           |        |  |  |                           |        |  |  |                          |        |  |  |                        |        |  |  |                             |        |  |  |                             |        |  |  |                     |        |  |  |                   |
| POS-r02.txt    | Today at 11:05 AM | HDFS   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |   |          |     |              |        |  |  |                           |        |  |  |                           |        |  |  |                          |        |  |  |                        |        |  |  |                             |        |  |  |                             |        |  |  |                     |        |  |  |                   |
| POS-r01.txt    | Today at 11:05 AM | HDFS   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |   |          |     |              |        |  |  |                           |        |  |  |                           |        |  |  |                          |        |  |  |                        |        |  |  |                             |        |  |  |                             |        |  |  |                     |        |  |  |                   |
| REF_CAL.txt    | Today at 11:05 AM | HDFS   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |   |          |     |              |        |  |  |                           |        |  |  |                           |        |  |  |                          |        |  |  |                        |        |  |  |                             |        |  |  |                             |        |  |  |                     |        |  |  |                   |
| ✓ REF_PROD.txt | Today at 11:05 AM | HDFS   | <table> <thead> <tr> <th>#</th> <th>ITEM_NBR</th> <th>RBC</th> <th>PRODUCT DESC</th> </tr> </thead> <tbody> <tr><td>491000</td><td></td><td></td><td>ACME RICE CRACKERS CHEESE</td></tr> <tr><td>474000</td><td></td><td></td><td>ACME RICE CRACKERS SESAME</td></tr> <tr><td>498000</td><td></td><td></td><td>ACME RICE CRACKERS ONION</td></tr> <tr><td>555000</td><td></td><td></td><td>ACME RICE CRACKERS BBQ</td></tr> <tr><td>562000</td><td></td><td></td><td>ACME RICE CRACKERS ORIGINAL</td></tr> <tr><td>352000</td><td></td><td></td><td>ACME RICE CRACKERS TERIYAKI</td></tr> <tr><td>528000</td><td></td><td></td><td>ACME SODAS UNSALTED</td></tr> <tr><td>528000</td><td></td><td></td><td>ACME SODAS SALTED</td></tr> </tbody> </table> | # | ITEM_NBR | RBC | PRODUCT DESC | 491000 |  |  | ACME RICE CRACKERS CHEESE | 474000 |  |  | ACME RICE CRACKERS SESAME | 498000 |  |  | ACME RICE CRACKERS ONION | 555000 |  |  | ACME RICE CRACKERS BBQ | 562000 |  |  | ACME RICE CRACKERS ORIGINAL | 352000 |  |  | ACME RICE CRACKERS TERIYAKI | 528000 |  |  | ACME SODAS UNSALTED | 528000 |  |  | ACME SODAS SALTED |
| #              | ITEM_NBR          | RBC    | PRODUCT DESC                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |   |          |     |              |        |  |  |                           |        |  |  |                           |        |  |  |                          |        |  |  |                        |        |  |  |                             |        |  |  |                             |        |  |  |                     |        |  |  |                   |
| 491000         |                   |        | ACME RICE CRACKERS CHEESE                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |   |          |     |              |        |  |  |                           |        |  |  |                           |        |  |  |                          |        |  |  |                        |        |  |  |                             |        |  |  |                             |        |  |  |                     |        |  |  |                   |
| 474000         |                   |        | ACME RICE CRACKERS SESAME                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |   |          |     |              |        |  |  |                           |        |  |  |                           |        |  |  |                          |        |  |  |                        |        |  |  |                             |        |  |  |                             |        |  |  |                     |        |  |  |                   |
| 498000         |                   |        | ACME RICE CRACKERS ONION                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |   |          |     |              |        |  |  |                           |        |  |  |                           |        |  |  |                          |        |  |  |                        |        |  |  |                             |        |  |  |                             |        |  |  |                     |        |  |  |                   |
| 555000         |                   |        | ACME RICE CRACKERS BBQ                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |   |          |     |              |        |  |  |                           |        |  |  |                           |        |  |  |                          |        |  |  |                        |        |  |  |                             |        |  |  |                             |        |  |  |                     |        |  |  |                   |
| 562000         |                   |        | ACME RICE CRACKERS ORIGINAL                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |   |          |     |              |        |  |  |                           |        |  |  |                           |        |  |  |                          |        |  |  |                        |        |  |  |                             |        |  |  |                             |        |  |  |                     |        |  |  |                   |
| 352000         |                   |        | ACME RICE CRACKERS TERIYAKI                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |   |          |     |              |        |  |  |                           |        |  |  |                           |        |  |  |                          |        |  |  |                        |        |  |  |                             |        |  |  |                             |        |  |  |                     |        |  |  |                   |
| 528000         |                   |        | ACME SODAS UNSALTED                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |   |          |     |              |        |  |  |                           |        |  |  |                           |        |  |  |                          |        |  |  |                        |        |  |  |                             |        |  |  |                             |        |  |  |                     |        |  |  |                   |
| 528000         |                   |        | ACME SODAS SALTED                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |   |          |     |              |        |  |  |                           |        |  |  |                           |        |  |  |                          |        |  |  |                        |        |  |  |                             |        |  |  |                             |        |  |  |                     |        |  |  |                   |

Browse current flow

Cancel

Accept

Figure: Select dataset or recipe to join

You can use the Data tab to preview the data in the selected object.

**NOTE:** You must have read access to the object to join it to your dataset.

- Use the Search bar to locate specific objects.
- Click **Accept**.

Step 2 - Select Join Conditions

In the next step, you specify the type of join and one or more join keys (columns).

Dataset samples

Join - Keys & Conditions

Join Key

#

Item\_Nbr

#

ITEM\_NBR

322k - 580k

322k - 580k

|   |        |        |
|---|--------|--------|
| - | 381000 | 381000 |
| - | 325000 | 325000 |
| - | 325000 | 325000 |
| - | 403000 | 403000 |
| - | 449000 | 449000 |
| - | 490000 | 490000 |
| - | 560000 | 560000 |
| - | 573000 | 573000 |
| - | 486000 | 486000 |
| - | 488000 | 488000 |
| - | 490000 | 490000 |
| - | 498000 | 498000 |
| - | 503000 | 503000 |
| - | 530000 | 530000 |
| - | 560000 | 560000 |
| - | 573000 | 573000 |
| - | 325000 | 325000 |
| - | 325000 | 325000 |

8,161 Rows in

165 Rows in

8,161 Rows in Output

Search row values...

Join type

required

Inner

Join keys

Add

#

Item\_Nbr

= (Equal to)

#

ITEM\_NBR

Suggested Q

99% match

Results summary

Based on current samples

Rows in Current

8161

Rows in Joined-in

165

Rows in Output

8161

Back

Next

Show only:

☒ Included Rows

☐ Excluded Rows

### Figure: Specify join type and join keys

#### Dataset samples:

Mouse over the Dataset samples indicator to see the current samples from the datasets that are part of the join. For more information, see *Samples Panel*.

#### Join type:

From the drop-down, select the type of join to apply. For more information, see *Join Types*.

#### Join keys:

In the above image, the platform has determined that the item number (`Item_Nbr`) field of Region 1 data and the item number (`ITEM_NBR`) field from `REF_PROD` should be used as the keys for performing the join.

**NOTE:** By default, Designer Cloud powered by Trifacta Enterprise Edition displays a maximum of three rows of data for each join key value in your sample. So, when you specify your join, it may seem like there are joined values that are missing from the data grid panel. When the job is run across the entire dataset, however, the join generates the appropriate number of rows. For more information on changing the maximum number of rows that are previewed in the join, see *Miscellaneous Configuration*.

- To make changes to the two join keys, mouse over the specified keys:
  - To remove the two columns as join keys, click the X icon.
  - To edit the keys to use and other key options, click the Pencil icon. See below.
  - To add more join keys, click **Add**.

**NOTE:** Be careful applying multiple join keys. Depending on the join type, this type of join can greatly expand the size of the generated data.

#### Edit keys:

By default, matches between join keys are performed on a strict, case-sensitive matching between key values in the selected columns. In some cases, it may be useful to loosen the conditions under which matches are found. Depending on the type of join, you can specify a range of matching values for your join conditions. For more information, see *Configure Range Join*. The following options are applied to the join key columns in both sources to attempt to find matches. After the join is executed, no data in either column is changed based on these selections.

| Option         | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| fuzzy match    | <p>Use a fuzzy matching algorithm for key value matching.</p> <p><b>Tip:</b> Use this option to perform fuzzy join matching of primary keys between datasets.</p> <p><b>NOTE:</b> Fuzzy joins can only be applied to String data types. Other data types cannot be fuzzy-matched using the algorithm.</p> <p>Fuzzy matching uses the doublemetaphone algorithm for matching strings (keys). Both primary encodings of each key value must match. See <i>DOUBLEMETAPHONEEQUALS Function</i>.</p> |
| Ignore case    | Ignore case differences between the join key values for matching purposes.                                                                                                                                                                                                                                                                                                                                                                                                                      |
| Ignore special | Ignore all characters that are not alphanumeric, accented Latin characters, or whitespace, prior to testing for a match.                                                                                                                                                                                                                                                                                                                                                                        |

|                   |                                                                                           |
|-------------------|-------------------------------------------------------------------------------------------|
| characters        |                                                                                           |
| Ignore whitespace | Ignore all whitespace characters, including spaces, tabs, carriage returns, and newlines. |

### Summary:

You can use these metrics to identify the likelihood of accurate matching between the join keys and the row count generated in the output.

Click **Next**.

## Step 3 - Select Output Columns

From the selected datasets, you can specify the columns to include in the output.

The screenshot displays the 'Join - Output Columns' configuration window. The top section, 'Join Output Preview', shows histograms for various columns: Item\_Nbr (322k - 580k), ITEM\_NBR1 (322k - 580k), Store\_Nbr (1 - 99), WM\_Week (201.05k - 201.05k), Daily (Feb 1 - Feb 8), and Whse\_Nbr (0 - 7.04k). Below these is a table of data rows. The right panel, 'Columns', lists available columns with checkboxes for selection. The 'Advanced options' section at the bottom includes 'Back' and 'Save and Continue' buttons.

**Figure: Select output columns**

### Select columns:

Review the list of available columns, which are displayed for both sources.

- Use the search panel to search for specific columns.
- To include all columns:
  - Click the All, Current, or Join-In tabs.
  - Click the checkbox at the top of the list.

### Advanced options

#### Name prefixes

You can apply prefixes to column names in the joined dataset, which can be helpful for tracking the source of a column in complex datasets. For example, you may wish to prepend each column from a dataset called, salesRegion01 with the prefix: sR01.

- **Name prefix for columns in Current data:** Enter a prefix to apply to the names of columns sourced from your current dataset that appear in the joined output.

- **Name prefix for columns in Joined\_in data:** Enter a prefix to apply to the names of columns sourced from the joined-in dataset that appear in the joined output.

## Dynamically updating Joins

After you have joined in another set of data, subsequent changes to that data can be automatically reflected in the output of the join:

- **Include all columns from Current data:** Dynamic updates always include the latest data from your current dataset.
- **Include all columns from Joined-In data:** Dynamic updates always include the latest data from the dataset that you are joining in.

**NOTE:** After you add your join to the recipe, if the data grid is empty, then the keys that you specified in the join may not have a match in the currently selected sample. You should revisit the keys used in your join. If the join still generates an empty grid on the current sample, you should collect a new sample. See *Samples Panel*.

**Tip:** If you must freeze the data in the dataset that you are joining in, you should create a copy of the dataset as a snapshot and join in the copy. See *Dataset Details Page*.

To join in the copy, edit the join and change the source that is being joined. See *Fix Dependency Issues*.

Click **Save and Continue**.

After you have selected your columns and any advanced settings, click **Review**.

## Step 4 - Review Join

Review the join that you have specified. To modify any aspect of it, click the appropriate **Edit** link.

The screenshot displays the 'Join - Edit Step' interface. At the top, there's a search bar labeled 'Search row values.' Below it, a 'Join Output Preview' section shows a data grid with columns: Item\_Nbr, ITEM\_NBR1, Store\_Nbr, WM\_Week, Daily, and Whse\_Nbr. The grid contains 165 rows of data. To the right of the grid, there's a 'Joined-in data' section showing 'REF\_PROD.txt' with an 'Edit' link. Below that, the 'Join type' is set to 'Inner' with an 'Edit' link. The 'Join keys' section shows two keys: 'Item\_Nbr' and 'ITEM\_NBR' with an '=' operator. The 'Output columns (30)' section shows '16 columns from Current' and '14 columns from Joined-in'. At the bottom right, there's a 'Save to Recipe' button. At the bottom left, there's a status bar showing '8,161 Rows in', '165 Rows in', and '8,161 Rows in Output'.

**Figure: Review join**

To add the specified join to your recipe, click **Add to Recipe**.

# Union Page

Contents:

- Mapping Schema
  - Custom column mappings
- Output Panel
- Updates

In the Union page, you can append data from one or more datasets to an existing dataset.

For example, if you have multiple datasets containing transactional data, such as log files, you can use the union operation to join daily or weekly slices of this data into a single dataset.

In a **union** operation, the Designer Cloud® application attempts to match columns between multiple datasets. As needed, you can perform manual tweaks to the matching and decide which columns to include or exclude in the resulting dataset.

- A union operation is different from a join operation. In a **join** operation, data from two datasets is brought together based on a defined primary key. The type of join determines the columns included in the output. For more information, see *Join Window*.

**Tip:** Depending on the types of operations you need to perform, you should perform your union steps earlier or later in the recipe. See *Optimize Job Processing*.

In the Search panel, enter `union` in the textbox.

Union

CancelAdd to Recipe

Match columnsAdd data

UNION DATA (2)

Union Output

4 Columns in Union

ABCKey2

ABCAntimal2

ABCVegetable2

ABCElement2

1 Dropped columnInclude all

+ABCMineral1

Dataset01 - 3

4 of 4 Columns in Union

ABCKey

ABCAntimal

ABCVegetable

ABCElement

Add column

No Dropped columns

Dataset02 - 3

4 of 5 Columns in Union

ABCKey

ABCAntimal

ABCVegetable

ABCElement

Add column

1 Dropped column

ABCMineral

Figure: Union Page



## Dataset Actions:

- To add data from a dataset, recipe, or reference to the union, click **Add data**.
- Select one or more objects to add to the union and choose one of the following methods to match columns:
  - **Auto Align**. When this option is selected, Designer Cloud powered by Trifacta Enterprise Edition performs intelligent mapping of the columns of the new dataset(s) to the dataset already loaded in the Transformer page. Auto alignment uses the following to map:
    - Edit distance between column names
    - Column data types
    - Similarity between sampled data in the datasets

**NOTE:** Auto align is not available after you have selected the dataset to union. Auto align may add a few seconds to the union operation.

**Add Datasets and Align by Name.** Matches are made based on the name of each column. Partial matches might be identified as matches, as well.

- **Add Datasets and Align by Position.** Matches are made based on horizontal position of each column in each dataset. Extra columns will be dropped. This method might be useful if column names have changed between datasets.
- To remove data from the union, click the X next to its name in the right panel.
  - You cannot remove the original dataset from which the Union page was opened.

## Mapping Schema

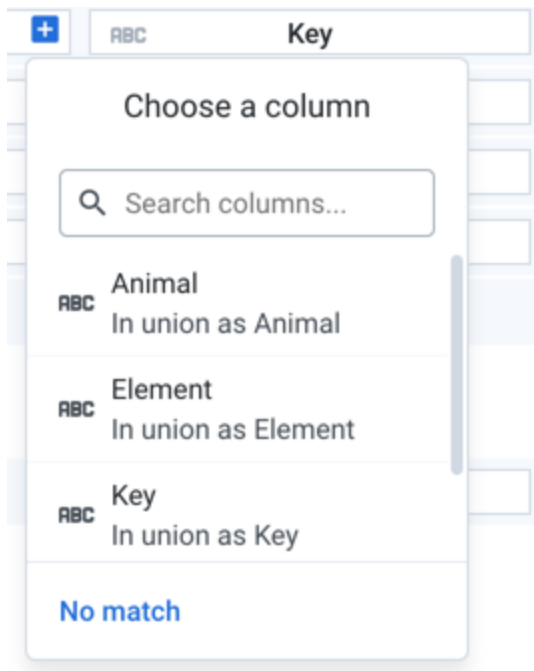
The schema of the output that is to be generated by the union operation is displayed in the left panel.

- The column names of the original dataset are used to populate the column names of the output dataset, where applicable.
- Each object that has been added to the union is displayed in the right panel.

| Panel | Left Side                       | Right Side 1               | Right Side 2               |
|-------|---------------------------------|----------------------------|----------------------------|
| Upper | Output dataset - included cols. | Dataset 1 - included cols. | Dataset 2 - included cols. |
| Lower | Output dataset - excluded cols. | Dataset 1 - excluded cols. | Dataset 2 - excluded cols. |

## Custom column mappings

As needed, you can modify the default column mappings in your dataset. To remap a column, hover over the column entry in the right panel, Then, click the Plus icon:



**Figure: Custom Column Mapping**

In the window, you can select the column in the current dataset that should appear in that location. Use this dialog to remap column order in each dataset.

- Click the Search columns field and begin typing to locate other columns.
- You can also specify that no match should be performed, which results in no data being imported from this column into the unioned dataset.

**Tip:** To map one of the dropped columns in your additional data to one of the source columns, hover over the empty No Match area next to the source column entry. Click the Plus icon to open the above mapping. Then, select the column from your additional data to slot into that location.

## Output Panel

In the left panel, you can review and modify the columns to be included in and excluded from the output. By default, all matching columns are included in the output; if there are no initial matching columns, all columns from the original dataset are included in the output by default. You can see the columns that are sources for the union output column on the same line in the right panel.

- Each column entry contains a data type identifier for the source column. Data types may be re-inferred as part of the union. You can change the data type after the union is completed.
- To the right of the column name, you can see the number of datasets in the union where the column occurs.

### Column Actions:

- To review the top five values for any column, click the Expand icon. You can see the count of each value across all included data.
- To remove a column from the union output, click the X icon to the left of the column entry in the upper panel.
- To add a column to the union output, click the + icon next to the left of the column entry in the lower panel.
- To include all available columns in the output, click **include all**.
- To add the union as specified, click **Add to Recipe**.

**NOTE:** Unnest, union, or join transforms may significantly increase the number of rows or columns in your dataset. To prevent overloading the browser's memory, the application may apply a limit function to the results to artificially limit the number of rows displayed in your sample. You can generate a new sample if desired. This limitation is not applied during the job execution.

## Updates

To modify a union after it has been created, click the Edit icon for the entry in the Recipe panel. See *Recipe Panel*.

After you have added the union to your recipe, changes to the underlying data should automatically propagate to the dataset into which they have been unioned. No refreshing of the data is necessary.

However, it is possible that subsequent changes to your sources can cause problems in the output and downstream references. You can fix these dependency issues.

**Tip:** If you must freeze the data that you are adding in, you should create a copy of it as a snapshot and union in the copy. See *Dataset Details Page*.

To use the copy, edit the `union` transform in the copy and switch the data that is in use. See *Fix Dependency Issues*.

# Run Job Page

## Contents:

- *Running Environment*
  - *Options*
  - *Publishing Actions*
    - *Add publishing action*
    - *Variables*
    - *Output settings*
  - *Run Job*
  - *Automation*
    - *Run jobs via API*
- 

In the Run Job page, you can specify transformation and profiling jobs for the currently loaded recipe. Available options include output formats and output destinations.

You can also configure the environment where the job is to be executed.

**Tip:** Jobs can be scheduled for periodic execution through Flow View page. For more information, see *Add Schedule Dialog*.

**Tip:** Columns that have been hidden in the Transformer page still appear in the generated output. Before you run a job, you should verify that all currently hidden columns are ok to include in the output.

Run Job ×

Running Environment

☒ **Trifacta Photon**  
 Run job on Trifacta Photon (best for small and medium-sized jobs, up to approximately 1 GB of data)

☐ **Spark**  
 Run job on Spark

Options

☒ **Profile results**  
 When enabled, this will generate a profile of your results

☒ **Ignore recipe errors**  
 Allow jobs to be run even if there are errors present in recipe steps

Publishing Actions + Add Action

| Actions    | Location | Settings                                                                  |
|------------|----------|---------------------------------------------------------------------------|
| Create-CSV |          | no compression, single file, with headers, with quotes, with delimiter, . |

SQL Scripts + Add Script

| Connection                                                | SQL statement | Settings |
|-----------------------------------------------------------|---------------|----------|
| No SQL scripts yet.<br><a href="#">Add new SQL script</a> |               |          |

Cancel Run

**Figure: Run Job Page**

## Running Environment

Select the environment where you wish to execute the job. Some of the following environments may not be available to you. These options appear only if there are multiple accessible running environments.

**NOTE:** Running a job executes the transformations on the entire dataset and saves the transformed data to the specified location. Depending on the size of the dataset and available processing resources, this process can take a while.

**Tip:** The application attempts to identify the best running environment for you. You should choose the default option, which factors in the available environments and the size of your dataset to identify the most efficient processing environment.

**Photon:** Executes the job in Photon, an embedded running environment hosted on the same server as the Designer Cloud powered by Trifacta® Enterprise Edition.

**Spark:** Executes the job using the Spark running environment.

### Advanced Execution Options:

- If Spark job overrides have been enabled in your environment, you can apply overrides to the specified job. See *Spark Execution Properties Settings*.
- This setting must be enabled. For more information, see *Enable Spark Job Overrides*.

**Spark (Databricks):** Executes the job on the Databricks cluster with which the platform is integrated.

**NOTE:** Designer Cloud powered by Trifacta platform can integrate with AWS Databricks or Azure Databricks, but not both at the same time.

For more information, see *Configure for AWS Databricks*.

For more information, see *Configure for Azure Databricks*.

**NOTE:** Use of Databricks is not supported on Marketplace installs.

## Options

**Profile Results:** Optionally, you can disable profiling of your output, which can improve the speed of overall job execution. When the profiling job finishes, details are available through the Job Details page, including links to download results.

**NOTE:** Percentages for valid, missing, or mismatched column values may not add up to 100% due to rounding. This issue applies to the Photon running environment.

See *Job Details Page*.

**Ignore recipe errors:** Optionally, you can choose to ignore errors in your recipes and proceed with the job execution.

**NOTE:** When this option is selected, the job may be completed with warning errors. For notification purposes, these jobs with errors are treated as successful jobs, although you may be notified that the job completed with warnings.

Details are available in the Job Details page. For more information, see *Job Details Page*.

## Publishing Actions

You can add, remove, or edit the outputs generated from this job. By default, a CSV output for your home directory on the selected datastore is included in the list of destinations, which can be removed if needed. You must include at least one output destination.

### Columns:

- **Actions:** Lists the action and the format for the output.
- **Location:** The directory and filename or table information where the output is to be written.
- **Settings:** Identifies the output format and any compression, if applicable, for the publication.

### Actions:

- To change format, location, and settings of an output, click the Edit icon.
- To delete an output, click the X icon.

## Add publishing action

From the available datastores in the left column, select the target for your publication.

**Publishing Action**

Search... (/)

Choose a file or folder

... / queryResults /admin [Create Folder](#)

Search...

| NAME             | SIZE | LAST UPDATED      |
|------------------|------|-------------------|
| 📁 .trifacta      |      | Today at 10:26 AM |
| 📁 POS-schema.csv |      | Today at 10:26 AM |

+ New Edit

Create a new file [Parameterize destination](#)

POS-r01

Output Directory

/trifacta/queryResults/admin@trifacta.local

Data Storage Format

CSV

✓ **Create new file every run**  
Create a new file with an incremental number appended to the name (e.g. POS-r01\_2.csv)

**Append to this file every run**  
Create it if it doesn't exist.

**Replace this file every run**  
Create it if it doesn't exist.

[More options](#)

Cancel Add

**NOTE:** Do not create separate publishing actions that apply to the same file or database table.

**New/Edit:** You can create new or modify existing connections. By default, the displayed connections support publishing. See *Create Connection Window*. **Steps:**

1. **Select the publishing target.** Click an icon in the left column.

- a. If Hive publishing is enabled, you must select or specify a database table to which to publish.

Depending on the running environment, results are generated in Avro or Parquet format. See below for details on specifying the action and the target table.

If you are publishing a wide dataset to Hive, you should generate results using Parquet.

For more information on how data is written to Hive, see *Hive Data Type Conversions*.

2. **Locate a publishing destination:** Do one of the following.

- a. **Explore:**

**NOTE:** The publishing location must already exist before you can publish to it. The publishing user must have write permissions to the location.

**NOTE:** If your HDFS environment is encrypted, the default output home directory for your user and the output directory where you choose to generate results must be in the same encryption zone. Otherwise, writing the job results fails with a *Publish Job Failed* error. For more information on your default output home directory, see *Storage Config Page*.

- i. To sort the listings in the current directory, click the carets next to any column name.

- ii. For larger directories, browse using the paging controls.
    - iii. Use the breadcrumb trail to explore the target datastore. Navigate folders as needed.
  - b. **Search:** Use the search bar to search for specific locations in the current folder only.
  - c. **Manual entry:** Click the Edit icon to manually edit or paste in a destination.
3. **Choose an existing file or folder:** When the location is found, select the file to overwrite or the folder into which to write the results.

**NOTE:** You must have write permissions to the folder or file that you select.

- a. To write to a new file, click **Create a new file**.

**Create a new file:** See below.

- 4. **Create Folder:** Depending on the storage destination, you can click it to create a new folder for the job inside the currently selected one. Do not include spaces in your folder name.
- 5. As needed, you can parameterize the outputs that you are creating. Click **Parameterize destination** in the right panel. See Parameterize destination settings below.
- 6. To save the publishing destination, click **Add**.

To update a publishing action, hover over its entry. Then, click **Edit**.

To delete a publishing action, select **Delete** from its context menu.

## Variables

If any variable parameters have been specified for the datasets or outputs of the flow, you can apply overrides to their default values. Click the listed default value and insert a new value. A variable can have an empty value.

**NOTE:** Override values applied to a job are not validated. Invalid overrides may cause your job to fail.

**NOTE:** Unless this output is a scheduled destination, variable overrides apply only to this job. Subsequent jobs use the default variable values, unless specified again. No data validation is performed on entries for override values.

**Tip:** At the flow level, you can specify overrides at the flow level. Override values are applied to parameters of all types that are a case-sensitive match. However, values that are specified at runtime override flow-level overrides. For more information, see *Manage Parameters Dialog*.

For more information on variables, see *Overview of Parameterization*.

## Output settings

Depending on the type of output that you are generating, you must specify additional settings to define location, format, and other settings.

## Run Job

To execute the job as configured, click **Run**. The job is queued for execution. After a job has been queued, you can track its progress toward completion. See *Job Details Page*.



## Automation

### **Run jobs via API**

You can use the available REST APIs to execute jobs for known datasets. For more information, see *API Reference*.

# SQL Scripts Panel

When specifying on-demand or scheduled outputs, you can define SQL scripts to execute before data ingestion, after output publication, or both. These scripts can be executed through any database connection to which you have write access.

**NOTE:** This feature may need to be enabled by a workspace administrator. For more information, see *Workspace Settings Page*.

When specifying your output, you can choose to add SQL scripts.

| SQL Scripts                                                                                |                                                                                                            |                        | <a href="#">+ Add Script</a> |
|--------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------|------------------------|------------------------------|
| Connection                                                                                 | SQL statement                                                                                              | Settings               |                              |
|  teradata | INSERT INTO "dataprep"."joblog" ("TIMESTAMP","JOBTYP","STATUS") VALUES ("210617-150422","orders","begin"); | Run before data ingest |                              |

Figure: SQL Scripts panel

Columns:

- **Connection:** The name of the connection where the script is to be executed.
- **SQL statement:** The first part of the SQL statement to be executed.
- **Settings:**
  - Run before data ingest: during job execution, script is to be run before data is ingested for job execution.
  - Run after data publish: during job execution, script is to be run after job has been executed and data is published.

**NOTE:** If publishing job fails, then all downstream tasks also fail, including the SQL script, which is not executed and is recorded as a failed phase of the job execution.

Actions:

- **Add Script:** To add a new SQL script for this output, click **Add Script**. See below.
- **Edit:** To modify the SQL script, highlight the entry and click **Edit**.
- **Delete:** To removal the SQL script, highlight the entry. Then, click **More menu > Delete**.

## Add SQL Script Window

Enter your SQL statement in the window.

**Figure: Add SQL Script window**

**Steps:**

1. Select the connection through which to apply the SQL statement.
2. Enter your SQL statements in the window:

**NOTE:** Each line must end with a semi-colon (;). Validation fails if otherwise.

- a. You may enter multi-statement SQL scripts.
- b. SQL lines in an individual script are executed in the order listed in the script.
- c. Your SQL statements must comply with the expected syntax of the target system. For more information, see *Supported SQL Syntax*.
3. Choose when to run the SQL script:
  - a. Run before data ingest: SQL script is executed before the data is ingested for a job run.
  - b. Run after data publish: SQL script is executed after that data has been published from a job run.
4. To validate your SQL, click **Validate SQL**.
5. To add the SQL script, click **Add**.

If you have defined multiple scripts of the same type (before data ingest, for example), those scripts may be executed in parallel.

**NOTE:** The order of listing of scripts in the Designer Cloud application does not affect the order of execution of those scripts.

For more information on managing SQL scripts, see *Create Output SQL Scripts*.

# Redshift Table Settings

If you are creating a publishing action for a Redshift database table in the Run Job page, you must provide the following information.

**NOTE:** Some Trifacta data types may be exported to Redshift using different data types. For more information, see *Redshift Data Type Conversions*.

## Steps:

1. **Select location:** Navigate the Redshift browser to select the schema and table to which to publish.
  - a. To create a new table, click **Create a new table**.
2. **Select table options:**
  - a. **Table name:**
    - i. New table: enter a name for it. You may use a pre-existing table name, and schema checks are performed against it.
    - ii. Existing table: you cannot modify the name.
  - b. **Output database:** To change the database to which you are publishing, click the Redshift icon in the sidebar. Select a different database.
  - c. **Publish actions:** Select one of the following.
    - i. **Create new table every run:** Each run generates a new table with a timestamp appended to the name.
    - ii. **Append to this table every run:** Each run adds any new results to the end of the table.
    - iii. **Truncate the table every run:** With each run, all data in the table is truncated and replaced with any new results.
    - iv. **Drop the table every run:** With each run, the table is dropped (deleted), and all data is deleted. A new table with the same name is created, and any new results are added to it.
3. To save the publishing action, click **Add**.

# Hive Table Settings

When publishing to Hive, please complete the following steps to configure the table and settings to apply to the publish action through the Run Job page.

**NOTE:** Some Trifacta data types may be exported to Hive using different data types. For more information on how types are exported to Hive, see *Hive Data Type Conversions*.

## Steps:

1. **Select location:** Navigate the Hive browser to select the database and table to which to publish.
  - a. To create a new table, click **Create a new table**.
2. **Select table options:**
  - a. **Table name:**
    - i. New table: enter a name for it. You may use a pre-existing table name, and schema checks are performed against it.
    - ii. Existing table: you cannot modify the name.
  - b. **Output database:** To change the database to which you are publishing, click the Hive icon in the sidebar. Select a different database.

**NOTE:** You cannot publish to a Hive database that is empty. The database must contain at least one table.

- c. **Publish actions:** Select one of the following.

**NOTE:** If you are writing to unmanaged tables in Hive, create and drop & load actions are not supported.

- i. **Create new table every run:** Each run generates a new table with a timestamp appended to the name.
- ii. **Append to this table every run:** Each run adds any new results to the end of the table.

**Tip:** Optionally, users can be permitted to publish to Hive staging schemas to which they do not have full create and drop permissions. This feature must be enabled. For more information, see *Configure for Hive*.

When enabled, the name of the staging DB must be inserted into your user profile. See *User Profile Page*.

- iii. **Truncate the table every run:** With each run, all data in the table is truncated and replaced with any new results.
  - iv. **Drop the table every run:** With each run, the table is dropped (deleted), and all data is deleted. A new table with the same name is created, and any new results are added to it.
3. To save the publishing action, click **Add**.

# Databricks Tables Table Settings

When you select a Databricks Tables database to store your job results in the Run Job page, you can configure the following options for the generated table.

**NOTE:** Access to Databricks Tables requires integration with Databricks, a Databricks Tables connection, and a Databricks personal access token.

- For more information, see *Configure for Azure Databricks*.
- For more information, see *Configure for AWS Databricks*.

Create a new table

Parameterize destination

POS\_r01\_\_2\_txt

Output Database

default

☒ Use Delta table

☐ Publish as external table

☒ Create new table every run

Create a new table with a timestamp appended to the name (e.g. POS\_r01\_\_2\_txt\_20200421\_113931)

Append to this table every run

Create it if it doesn't exist.

Truncate the table every run

Truncate existing data in the table and append new data.

Drop the table every run

Drop the table and create a new table of the same name.

Cancel

Add

Figure: Databricks Tables table settings

## Steps:

1. **Select location:** Navigate the Databricks Tables browser to select the database and table to which to publish.
  - a. To create a new table, click **Create a new table**.
2. **Select table options:**
  - a. **Table name:**
    - i. New table: enter a name for it. You may use a pre-existing table name, and schema checks are performed against it.
    - ii. Existing table: you cannot modify the name.

**NOTE:** Writing to partitioned tables is not supported.

- b. **Output database:** To change the database to which you are publishing, click the Databricks icon in the sidebar. Select a different database.
    - c. **Optional table types:** Select one or more table types to publish as well:
      - i. **Use Delta table:** Output is stored as a Parquet-based Delta table.

**NOTE:** Versioning and rollback of Delta tables is not supported within the Designer Cloud powered by Trifacta platform . The latest version is always used. You must use external tools to manage versioning and rollback.

- ii. **Publish as external table:** Output is published as an external table to the specified location in your bucket.
      - d. **Publish actions:** Depending on your selection or selections above, the following publishing actions on the table are supported:
        - i. **Create new table every run:** Each run generates a new table with a timestamp appended to the name.
        - ii. **Append to this table every run:** Each run adds any new results to the end of the table.
        - iii. **Truncate the table every run:** With each run, all data in the table is truncated and replaced with any new results.

**NOTE:** Truncating the table is not supported for external tables.

- iv. **Drop the table every run:** With each run, the table is dropped (deleted), and all data is deleted. A new table with the same name is created, and any new results are added to it.

**NOTE:** Dropping the table is not supported for external tables.

3. To save the publishing action, click **Add**.

# Tableau Server Datasource Settings

When publishing to Tableau Server through the Run Job page, please complete the following steps to configure the datasource and settings to apply to the publish action.

- A **datasource** is a table in your Tableau Server datastore that can be used as an input for your Tableau Server projects. For more information, see <https://onlinehelp.tableau.com/current/server/en-us/datasource.htm>.
  - For more information on creating a connection, see *Tableau Server Connections*.
  - For more information on how types are written to Tableau, see *Tableau Hyper Data Type Conversions*.
1. **Select location:** Navigate the Tableau Server browser to select the project and datasource to use for your publication.
    - a. For more information on projects, see <https://onlinehelp.tableau.com/current/server/en-us/projects.htm>.
    - b. To create a new datasource, click **Create a new datasource**.
      - i. For more information, see <https://onlinehelp.tableau.com/current/server/en-us/datasource.htm>.
  2. **Datasource options:**
    - a. **Datasource name:**
      - i. New datasource: enter a datasource for it. You may use a pre-existing datasource name.
      - ii. Existing datasource: you cannot modify the name.
    - b. **Output project:** To change the project to which you are publishing, click the Tableau icon in the sidebar. Select a different project.
    - c. **Publish actions:** Select one of the following.
      - i. **Create new datasource every run:** Each run generates a new datasource with a timestamp appended to the name.
      - ii. **Append to this datasource every run:** Each run adds any new results to the end of the datasource.
      - iii. **Drop the datasource every run:** With each run, the datasource is dropped (deleted), and all data is deleted. A new datasource with the same name is created, and any new results are added to it.
  3. To save the publishing action, click **Add**.

**Tip:** If you generate a Tableau format file as part of your output, you can choose to download and later publish it to Tableau Server. For more information, see *Publishing Dialog*.



# Spark Execution Properties Settings

When you specify a job in the Run Job page, you may pass to the Spark running environment a set of Spark property values to apply to the execution of the job. These property values override the global Spark settings for your deployment.

**NOTE:** A workspace administrator must enable Spark job overrides and configure the set of available parameters. For more information, see *Enable Spark Job Overrides*.

Spark overrides are applied to individual output objects.

- You can specify overrides for ad-hoc jobs through the Run Job page.
- You can specify overrides when you configure a scheduled job execution.

**User-specific Spark overrides:** If you have enabled user-specific overrides for Spark jobs, those settings take precedence over the settings that are applied through this feature. For more information, see *Configure User-Specific Props for Cluster Jobs*.

In the Run Job page, click the Advanced Execution Settings caret.

Run Job

Running Environment

Photon  
Run job on Trifacta Photon (best for small and medium-sized jobs, up to approximately 1 GB of data)

☒ Spark  
Run job on Spark

Advanced environment options ^

Spark Execution Properties ?

Spark Driver Memory

e.g.: 8G

Spark Executor Memory

e.g.: 8G

Spark Executor Cores

e.g.: 4

Transformer Dataframe Checkpoint Threshold

e.g.: 500

Figure: Spark Execution Properties

## Default Spark overrides:

The first four properties are available for all Spark job overrides:

**Before you modify these parameters, you should review with your cluster administrator what are appropriate settings for each parameter. In some cases, you can set these values to cause failures on the cluster. No validation is performed for inputted values.**

| Spark parameter                            | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|--------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Spark Driver Memory                        | <p>Amount of RAM in GB on each Spark node that is made available for the Spark drivers.</p> <p>By raising this number:</p> <ul style="list-style-type: none"><li>• The drivers for your job are allocated more memory on each Spark node.</li><li>• There is less memory available for other uses on the node.</li></ul>                                                                                                                                                                                                                                                                                                                                               |
| Spark Executor Memory                      | <p>Amount of RAM in GB on each Spark node that is made available for the Spark executors.</p> <p>By raising this number:</p> <ul style="list-style-type: none"><li>• The Spark executors for your job are allocated more memory.</li><li>• There is less memory available for other uses on the node.</li></ul>                                                                                                                                                                                                                                                                                                                                                        |
| Spark Executor Cores                       | <p>Number of cores on each Spark executor that is made available to Spark.</p> <p>By raising this number:</p> <ul style="list-style-type: none"><li>• The maximum number of cores available for your job is raised on each Spark executor.</li><li>• There are fewer cores for other uses on the node.</li></ul>                                                                                                                                                                                                                                                                                                                                                       |
| Transformer Dataframe Checkpoint Threshold | <p>When checkpointing is enabled, the Spark DAG is checkpointed when the approximate number of expressions in this parameter has been added to the DAG. Checkpointing assists in managing the volume of work that is processed through Spark at one time; by checkpointing after a set of steps, Designer Cloud powered by Trifacta Enterprise Edition can reduce the chances of execution errors for your jobs.</p> <p>By raising this number:</p> <ul style="list-style-type: none"><li>• You increase the upper limit of steps between checkpoints.</li><li>• You may reduce processing time.</li><li>• It may result in a higher number of job failures.</li></ul> |

For more details on setting these parameters, see *Tune Cluster Performance*.

## Other Spark overrides:

Your workspace administrator may have enabled other Spark properties to be overridden. These parameters appear at the bottom of the list.

Please check with your administrator for appropriate settings for these properties.

# Microsoft SQL Data Warehouse Table Settings

When you select a Azure® Synapse Analytics (Formerly Microsoft® SQL DW)® connection to publish your job results, you can configure the following options for the generated table.

For more information on creating these connections, see *Microsoft SQL Data Warehouse Connections*.

Create a new table    [Parameterize destination](#)

MyExternalTable

Output Schema

INFORMATION\_SCHEMA

☒ Publish as external table    ?

Location required

/    testing\_New\_Folder

Browse

☒ **Create new table every run**  
Create a new table with a timestamp appended to the name (e.g. MyExternalTable\_20220420\_165501)

☐ **Append to this table every run**  
This action is not available for external tables

☐ **Truncate the table every run**  
This action is not available for external tables

☐ **Drop the table every run**

Cancel

Update

**Figure: Microsoft SQL Data Warehouse table settings**

**Steps:**

1. **Select location:** Navigate the Azure Synapse Analytics (Formerly Microsoft SQL DW) browser to select the schema to which to publish.
  - a. To create a new table, click **Create a new table**.
2. **Select table options:**
  - a. **Table name:**

- i. **New table:** enter a name for it. You may use a pre-existing table name, and schema checks are performed against it.
  - ii. **Existing table:** you cannot modify the name.
- b. **Output schema:** To change the schema to which you are publishing, click the connection icon in the sidebar. Select a different schema.
- c. **Optional table types:** The following options may be available:
  - i. **Publish as external table:** Output is published as an external table to the specified location in your storage layer. Otherwise, the table is published as a managed table.

**NOTE:** When publishing to an external table, the output file type is Parquet.

- d. **Publish actions:** Depending on your selection or selections above, the following publishing actions on the table are supported. For more information, see "Publish Actions" below.
  - i. **Create new table every run:** Each run generates a new table with a timestamp appended to the name.
  - ii. **Append to this table every run:** Each run adds any new results to the end of the table.

**NOTE:** Appending the table is not supported for external tables.

- iii. **Truncate the table every run:** With each run, all data in the table is truncated and replaced with any new results.

**NOTE:** Truncating the table is not supported for external tables.

- iv. **Drop the table every run:** With each run, the table is dropped (deleted), and all data is deleted. A new table with the same name is created, and any new results are added to it.

**NOTE:** If the target is an external table, you can only drop the table when you first re-run a job to the target, after which you can choose to recreate the target as a managed or external table.

- 3. To save the publishing action, click **Add**.

# Preferences Page

## Contents:

- *User Preferences*
    - *Profile*
    - *Account*
    - *Email notifications*
  - *Workspace Preferences*
    - *Storage*
    - *Databricks personal access token*
    - *Access tokens*
- 

From the Preferences page, you can manage aspects of your user account and other settings. Select **Help menu** > **Preferences**.

## User Preferences

### Profile

Review and manage your user profile. For more information, see *User Profile Page*.

### Account

Change your password and set your locale among other preferences. See *Account Settings Page*.

### Email notifications

Configure your personal preferences on receiving email notifications about activities in the product.

**NOTE:** This feature requires access to an SMTP server to send emails. For more information, see *Enable SMTP Email Server Integration*.

**NOTE:** This feature may need to be enabled in your workspace. See *Workspace Settings Page*.

For more information, see *Email Notifications Page*.

## Workspace Preferences

### Storage

Review and modify the locations where you store data.

If you are connecting to AWS resources, you can modify your authentication credentials.

See *Storage Config Page*.

## Databricks personal access token

Each user must save a Databricks Personal Access Token to their user account. See *Databricks Settings Page*.

## Access tokens

Create and manage tokens for access to the APIs for Designer Cloud powered by Trifacta Enterprise Edition.

**Tip:** Access tokens provide a simple and secure method of consistent access to the externally available APIs.

For more information, see *Access Tokens Page*.

# User Profile Page

In your user profile, you can review your personal information and update your photo. Select **User menu > Preferences**.

**NOTE:** After saving changes to your user profile and exiting, please refresh the page.

## Profile

Photo



Upload photo

Email

Name

**Figure: User Profile Page**

### Profile settings

**Name:** Display name for your Trifacta® account.

**Email:** Email address associated with your account

**NOTE:** This value is the user ID. It must be a valid email address and cannot be modified after registration.

### Upload photo

**NOTE:** This feature may need to be enabled. See *Miscellaneous Configuration*.

You can upload a preferred image associated with your user account. This image appears wherever the application contains a personal identifier, such as the icon for the User menu.

**Image requirements:**

- Format: JPG (JPEG), PNG, GIF, SVG, BMP, WEBP
- Dimensions: square dimensions work best. If you are using a non-square image, you should center the image details along the shorter edge of the image.

**Steps:**

1. Below the icon at the top of the User Profile page, click **Upload photo**.
2. Navigate your local desktop.
3. Select the file and click **Open**.
4. The icon is replaced by the image from the file you uploaded.



# Storage Config Page

The Storage Config page allows you to configure storage locations for where you upload datasets and generate results.

**Tip:** When editing a directory location, click the Pencil icon. You can paste URLs to storage locations into the textbox.

The following options are available for configuring your storage environment.

**NOTE:** If you cannot modify the values in this page, you may need to enable the feature to modify user paths. For more information, see *Workspace Settings Page*.

## Output home directory

Relative path to the directory where your results are stored by default and where your samples are stored. Click **Edit** to modify the home directory where results are stored.

Full path concatenates Output Protocol/Host value and this value.

**Do not modify this value unless directed to do so. This path is not validated. If you specify a path to a directory to which you do not have appropriate permissions, all job exports will fail.**

**NOTE:** Multiple users cannot share the same home directory.

**NOTE:** If your HDFS environment is encrypted, any location that you specify to write results for a job must be in the same encryption zone as this directory. For more information, please contact your HDFS administrator.

## Upload directory

Relative path to the directory where your uploads are stored. To modify this value, click **Edit**.

**NOTE:** This setting only applies if Designer Cloud powered by Trifacta Enterprise Edition is connected to a backend datastore.

**NOTE:** You cannot upload to locations to which you do not have write access.

## **AWS credentials**

If per-user mode is enabled, this option allows workspace members to apply individual key-secret values and roles to their accounts and modify other personal storage settings. For more information, see *Configure Your Access to S3*.

# Account Settings Page

## Contents:

- *Change Password*
  - *Locale*
  - *Other Settings*
- 

In your Account Settings page, you can change your locale and modify other settings related to your account.

**NOTE:** After saving changes to your account settings and exiting, please refresh the page.

## Change Password

**NOTE:** If Single Sign-On (SSO) has been enabled, then these options are not available.

**Tip:** You can modify your password directly through the `/change-password` URL.

**Old password:** To change your password, enter your current password here.

**New password:** To change your password, enter a new password here.

**Confirm new password:** Confirm the above password before saving.

**Tip:** Forgot your password? Click **Reset password via email** to send a reset password to the email address registered for your account.

## Locale

Select the locale to use when validating data types in the application.

**NOTE:** After saving changes to your locale, refresh your page. Subsequent executions of the data inference service use the new locale settings.

**NOTE:** When locale is changed, data type validation is affected only on subsequent executions of data type inference. If you are using structured datasets, such as schematized files, data types may be attached to the datasets that you have already imported. These data types are not affected.

For more information, see *Locale Settings*.

## Other Settings

**Enable all popup helpers:** Re-enables the popup help windows. You can then re-review the on-boarding tour.

**NOTE:** This change is not applied until you login to the application again.

**NOTE:** This feature may need to be enabled in your environment by an administrator. For more information, see *Enable Onboarding Tour*.

**Enable keyboard shortcuts:** When enabled, you can use keyboard shortcuts in the workspace or Transformer page.

**Tip:** When keyboard shortcuts are enabled, press ? in the application to see the available shortcuts.

**Share usage data to improve product intelligence:** When collaborative suggestions are enabled, anonymized data on how you use the product is aggregated with other workspace users' data to improve the suggestions provided by the product to all workspace users. You can use this setting to opt-out of sharing your data.

**NOTE:** This data is not shared with Alteryx or other users.

**NOTE:** If this setting is not present, the feature is disabled in your workspace. For more information, see *Workspace Settings Page*.

# Email Notifications Page

## Contents:

- *Receive emails about job activity*
  - *Receive emails about plan runs*
  - *Receive emails when a flow or plan is shared with you*
  - *Delivery address*
  - *Timezone*
- 

You can configure the notifications that are sent to your email address based on job success or failure. Notifications can be sent for jobs executed from flows where you are the owner or a collaborator.

**NOTE:** Email notifications requires that you configure the Designer Cloud powered by Trifacta platform to use an available SMTP server. For more information, see *Enable SMTP Email Server Integration*.

**NOTE:** Email notifications may need to be enabled in your environment. You can also configure the type of jobs that generate success or failure emails. For more information, see *Workspace Settings Page*.

**Tip:** If visual profiling has been enabled, a PDF version of the profile of the job results is included as an attachment to the email.

## Receive emails about job activity

When enabled, you can receive email notifications about job activity. You can receive emails from:

- flows where you are an owner or collaborator
- flows where someone has added you as a watcher

Email notifications are configured on a per-flow basis. For more information, see *Manage Flow Notifications Dialog*.

## Receive emails about plan runs

When enabled, you can receive email notifications about plan runs if you are an owner or collaborator.

Email notifications are configured on a per-plan basis. For more information, see *Manage Plan Notifications Dialog*.

## Receive emails when a flow or plan is shared with you

When enabled, you automatically receive notifications to your registered email address when a flow or plan is shared with you.

## Delivery address

If needed, you can send notification emails to a different email address.

**Uses:**

- Your login email address cannot receive email.
- You wish to deliver email to an alias within your enterprise.
- You wish to deliver your email to a third-party application, such as Slack.

**Tip:** You can configure the sender email address and sender name that is used for all emails generated by the Trifacta node. For more information, see *Enable SMTP Email Server Integration*.

**Timezone**

Select the timezone in which email dates are displayed.

# Access Tokens Page

From the Access Token page, you can generate and manage access tokens that apply to your account. Access tokens can be used when accessing the REST APIs.

**Tip:** Access token usage is the preferred method of authenticating from API. See <https://api.trifacta.com/ee/es.t/index.html#section/Authentication>

**NOTE:** If this page is not visible, the feature has not been enabled in your instance of the platform. See *Enable API Access Tokens*.

**NOTE:** Workspace administrators can choose to enable creation and use of access tokens to individual workspace users. For more information, see *Workspace Settings Page*.

**NOTE:** Workspace administrators can delete the access tokens of other users.

Access Tokens

Generate New Token

| Token ID                             | Description | Status | Created             | Expires             | Last used  |
|--------------------------------------|-------------|--------|---------------------|---------------------|------------|
| 0bc1d49f-5475-4c62-a0ba-6ad269389ada | new token   | Active | 2019-01-15 12:58:28 | 2020-01-15 12:58:28 | Never Used |

Figure: Access Tokens Page

Actions:

- **Generate New Token:** Click to generate a new access token for your user account. See below.

Columns:

- **Token ID:** Internal identifier for the token

**NOTE:** This is not the token itself. That value is exposed when you create the token and must be retained for any use outside of the Designer Cloud application .

- **Description:** User-provided description of the token
- **Status:** Current status of the token:
  - Active - Token is active and can be used for access.
  - Expired - Token has expired after its expiration timestamp has been reached.
- **Created:** Timestamp for when the token was created.

- **Expires:** Timestamp for when the token expires.
- **Last Used:** Timestamp for when the token was last used.

#### Context menu:

- **Delete Token:** Click the delete the token.

**Deleting a token cannot be undone.**

**NOTE:** If you delete an active token, any API usage that references the token no longer works.

## Generate Token

When you generate a token, you can provide the following information.

Generate Token

×

Lifetime (days)

Set lifetime to -1 to have the token never expire.

Description (optional)

Cancel

Generate

**Figure: Generate Token Dialog**

- **Lifetime:** Enter the number of days that you would like to use this token without renewal.
  - If the token expires, a new one must be created. You can generate a new token at any time.
  - You can generate any number of tokens.

**Tip:** Depending on your environment, you may be able to set this value to -1 to never expire the token.

- **Description:** (Optional) you can provide a user-friendly description of the purpose of the token. This value is for display purposes only.

To create the token, click **Generate**.

After the token is generated, it is automatically activated. You can have multiple active tokens.

You must copy the token out of the application. Click **Copy Token to clipboard** to copy the text value of the token.

**For security purposes, after you close the My Token screen, the token is no longer accessible. You cannot reopen this dialog. You must copy this value and store it in a secure place for later use.**





# Databricks Settings Page

To access a Databricks cluster for running jobs, each user of the Designer Cloud powered by Trifacta® platform must insert their personal access token into their profile. This configuration enables the user to authenticate to a connected Databricks cluster.

**NOTE:** Each user must insert a personal access token into their profile. Users that do not provide a personal authentication token cannot run jobs on Databricks, including transformation, sampling, and profiling jobs.

**NOTE:** When you reset your personal access token (PAT), a new cluster is created if your new token does not have access to your current cluster. If you are resetting an expired personal access token, no new cluster is created. This new cluster is created when you first request access to the Databricks cluster. When you next use an interface that require access to the cluster, such as the relational browser, it may take some time to load.

## Prerequisites

- Acquire your Databricks personal access token.

**NOTE:** Your Databricks personal access token must be acquired from the same region as your Databricks deployment. This region name is available through the Designer Cloud application .

For more information, see <https://docs.azuredatabricks.net/api/latest/authentication.html#requirements>.

## Steps

1. Login to the application. From the menu bar, select **User menu > Preferences > Databricks**.
2. **Configure URL:**
  - a. **For Azure developments:** Acquire the Azure Databricks personal access token from the same region as your Azure Databricks deployment. The region name is available through the Designer Cloud application . For more information, see *Configure for Azure Databricks*.
  - b. **For AWS developments:** Edit the workspace URL, as required and click **Save**.
    - The existing property `databricks.serviceUrl` is used to configure the URL to the Databricks Service to run Spark jobs. For more information, see *Configure for AWS Databricks*.
    - The `databricks.serviceUrl` defines the default Databricks workspace for all user in the Designer Cloud powered by Trifacta Enterprise Edition workspace.
    - You can override the default settings in this page.
3. **Personal access token:** In the Personal Access Token field, paste your token.
  - a. To use a different token, click **Change**.
4. **Databricks table cluster name:** Each user can specify the name of a cluster to use to browse a Databricks Tables deployment.

**NOTE:** This cluster must be created and maintained by your Databricks administrator. This cluster can be shared among multiple users.

5. **Databricks policy name:** To select the cluster policy to use when you are executing jobs on the cluster, click **Edit**. The available policies are listed in the drop-down.

**NOTE:** Cluster policies determine characteristics of Databricks clusters that are used or created for job execution. This feature requires additional configuration.

- a. For more information, see *Configure for AWS Databricks*.
  - b. For more information, see *Configure for Azure Databricks*.
6. Click **Save**.

# Schedules Page

In the Schedules page, administrators can be review the current set of schedules.

**NOTE:** This page is available to project owners and workspace administrators only.

A **schedule** is automated execution of an output in a flow on a regular basis. Schedules are composed of the following:

- A schedule
- A set of one or more scheduled destinations
- These objects are created from the flow. For more information, see *Flow View Page*.
  - See *Add Schedule Dialog*.

|                     |                                               |                             |         |                   |
|---------------------|-----------------------------------------------|-----------------------------|---------|-------------------|
| Schedules           |                                               |                             |         |                   |
| Owner               | Frequency                                     | Flow                        | State   | Last Updated ▾    |
| Administrator (you) | Weekly: 12:00 AM - On Sunday                  | <a href="#">census data</a> | Enabled | Today at 11:16 AM |
| Administrator (you) | Cron: 0 0 * * TUE * - At 12:00 AM, only on... | <a href="#">2013 POS</a>    | Enabled | Today at 11:14 AM |

Figure: Schedules page

Columns:

- **Owner:** The user that owns the schedule.
- **Frequency:** The frequency of occurrence of the schedule.
  - **Cron:** Cron jobs utilize a modified form of cron scheduling syntax to define execution time. For more information, see *cron Schedule Syntax Reference*.
  - **Weekly/Monthly/Daily:** You can also schedule jobs to execute according to a regular calendar period.
- **Flow:** Name of the flow to which the schedule applies.
  - If available, you can click the link to open the flow. See *Flow View Page*.
- **State:** The current state of the schedule:
  - **Enabled** - The schedule is active and will execute at the next occurrence according to the frequency.
  - **Disabled** - The schedule is inactive and cannot be executed until it is enabled.
- **Last Updated:** Timestamp for the when the schedule was last modified.

Actions:

- **Filter by update:** Click the caret next to Last Updated to sort the list of schedules.
- **Search:** Enter text in the search field to filter the listed jobs by flow name.

Context menu:

Next to the schedule listing, click the options menu to see the following:

- **Enable/Disable Schedule:** Select this option to toggle availability of the schedule.

- **Delete Schedule:** Delete the schedule.

**Deleting a schedule is permanent and cannot be undone.**

# UI Index

Use the links below to access the reference pages for the Designer Cloud application user interface. For more information on UI pages that apply to administrators, see *Admin Reference*.

## Home

| Item                    | Description                                                                                                                                                                                                                                              |
|-------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Home Page</i>        | From the Home page, you can create or access your flows, datasets, and jobs, as well as configure settings and find additional resources.                                                                                                                |
| <i>Flows Page</i>       | The Flows page displays the flows to which you have access and lets you create, review, and manage them. A <b>flow</b> is an object for bringing together and organizing the datasets, recipes, and other objects that you use to generate your results. |
| <i>Plans Page</i>       | The Plans page lets you create, review, and manage your plans. A <b>plan</b> is a sequence of tasks and the triggers that execute them. Plans can be applied across multiple flows in your workspace.                                                    |
| <i>Library Page</i>     | In the Library page, you can review your imported and reference datasets and any macros that you may have created. You can also import new data from this page.                                                                                          |
| <i>Connections Page</i> | Through the Connections page, you can add new connections or modify the connections that you have already created. From the left nav bar, click the Connections icon.                                                                                    |
| <i>Jobs Page</i>        | In the Jobs page, you can track the status of all of your jobs and plan runs.                                                                                                                                                                            |
| <i>Transformer Page</i> | In the Transformer page, you identify the data that you need to transform and build your transformation recipes on samples taken from your currently selected dataset.                                                                                   |
| <i>Preferences Page</i> | From the Preferences page, you can manage aspects of your user account and other settings. Select <b>Help menu &gt; Preferences</b> .                                                                                                                    |
| <i>Schedules Page</i>   | In the Schedules page, administrators can review the current set of schedules.                                                                                                                                                                           |

| Item                        | Description                                                                                                                                                 |
|-----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Download Logs Dialog</i> | You can download logs for your current session in Designer Cloud powered by Trifacta® Enterprise Edition. From the Help menu, select <b>Download logs</b> . |

## Flows

### Flows Page

| Item                    | Description                                                                                                                                                                                                                                                         |
|-------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Create Flow Page</i> | You can use flows to organize your datasets and to track the jobs associated with them.                                                                                                                                                                             |
| <i>Flow View Page</i>   | In Flow View, you can access and manage the objects that you have added to or created in the selected flow. You can perform a variety of actions to effectively manage flow development and job execution through a single page in the Designer Cloud® application. |

## Flow View

### Flow View Page

| Item | Description                                                                                                                                 |
|------|---------------------------------------------------------------------------------------------------------------------------------------------|
|      | When you select an imported dataset in Flow View, you can review its details in the context panel and select options from its context menu. |

|                                          |                                                                                                                                                                                                                                                                                                                                                                                                                    |
|------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>View for Imported Datasets</i>        |                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <i>View for Dataset with Parameters</i>  | When you select a dataset with parameters in Flow View, additional options are available in its context menu and the Details panel.                                                                                                                                                                                                                                                                                |
| <i>View for Recipes</i>                  | For each recipe in Flow View, you can review or edit its steps or create new recipes altogether. You can also create references to the recipe, modify outputs, and create new recipes off of the recipe.                                                                                                                                                                                                           |
| <i>View for Reference Datasets</i>       | A <b>reference dataset</b> is a reference to a recipe's output, which can be added to a flow other than the one where the recipe is located.                                                                                                                                                                                                                                                                       |
| <i>View for Unstructured Datasets</i>    | An <b>unstructured dataset</b> is an imported dataset that does not contain any initial parsing steps. All parsing steps must be added through recipes that are applied to the dataset. During the import process, you disable the initial steps that are applied to imported datasets. Instead, these steps are added as the first steps of the auto-generated recipe that appears with the dataset in Flow View. |
| <i>View for Outputs</i>                  | Associated with each recipe is one or more outputs. These publishing destinations can be configured through the context panel in Flow View. Through outputs, you can execute and track jobs for the related recipe.                                                                                                                                                                                                |
| <i>Share Flow Dialog</i>                 | You can manage access to a flow for other users through the Share Flow dialog. In Flow View, select <b>Share</b> from the context menu.                                                                                                                                                                                                                                                                            |
| <i>Change Dataset Dialog</i>             | Through the Flow View page, you can change the source that is used for your dataset. In this manner, you can apply the same recipe across datasets with the same schema. When the source dataset has been changed, a new sample is automatically generated for you.                                                                                                                                                |
| <i>Add Schedule Dialog</i>               | To add a schedule to your flow, click the drop-down menu in the Flow View page, and select <b>Schedule</b> .                                                                                                                                                                                                                                                                                                       |
| <i>Manage Flow Notifications Dialog</i>  | When email notifications are enabled, flow owners and collaborators can configure the delivery of emails to interested stakeholders based on the success or failure of jobs executed within this flow. From the flow menu, select <b>Email notifications</b> .                                                                                                                                                     |
| <i>Manage Parameters Dialog</i>          | Within a flow, you can create and manage flow parameters, including specifying override values. From the flow menu, select <b>Parameters</b> .                                                                                                                                                                                                                                                                     |
| <i>Flow Search Panel</i>                 | You can search for objects within your flow. In Flow View, click the Magnifying Glass icon at the top of the page.                                                                                                                                                                                                                                                                                                 |
| <i>Flow Optimization Settings Dialog</i> | In the Flow Optimization Settings dialog, you can configure the following settings, which provide finer-grained control and performance tuning over your flow and its job executions. From the Flow View menu, select <b>Optimization settings</b> .                                                                                                                                                               |

## Plans

| Item                  | Description                                                                                                                                   |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Plan View Page</i> | In Plan View, you design your plan, which includes the building and sequencing of tasks and the triggers that execute your sequence of tasks. |

## Library

### Library Page

| Item | Description |
|------|-------------|
|------|-------------|

|                             |                                                                                                                                                                                  |
|-----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Dataset Details Page</i> | Use the Dataset Details page to review a dataset's usage and to perform management tasks on it.                                                                                  |
| <i>Import Data Page</i>     | Through the Import Data page, you can upload datasets or select datasets from sources that are stored on connected datastores. From the Library page, click <b>Import Data</b> . |
| <i>Macros Page</i>          | In the Macros page, you can review and manage the macros to which you have access.                                                                                               |

## Import Data Page

| Item                            | Description                                                                                                                                                                                                                                                                     |
|---------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Create Connection Window</i> | Through the Create Connection window, you can create and edit connections between Designer Cloud powered by Trifacta® Enterprise Edition and remote storage.                                                                                                                    |
| <i>Database Browser</i>         | The database browser enables you to interact with databases that are connected to Designer Cloud powered by Trifacta® Enterprise Edition.                                                                                                                                       |
| <i>File System Browser</i>      | In Designer Cloud powered by Trifacta® Enterprise Edition, the file system browser lets you browse, select, and filter the sources that you can access through the datastore to which you are connected. You also use the browser to select targets for publishing job results. |
| <i>HDFS Browser</i>             | The HDFS browser enables you to browse, select, and filter the files to which you have access in the Hadoop cluster to which Designer Cloud powered by Trifacta® Enterprise Edition is connected.                                                                               |
| <i>S3 Browser</i>               | In Designer Cloud powered by Trifacta® Enterprise Edition, the S3 browser lets you browse, select, and filter the sources that you can access through S3. You also use the browser to select targets for publishing job results.                                                |
| <i>File Import Settings</i>     | When you edit settings on a selected file in the Import Data page, the following settings are displayed.                                                                                                                                                                        |
| <i>Table Import Settings</i>    | When you edit settings for a selected table in the Import Data page, the following settings are displayed.                                                                                                                                                                      |

## Macros Page

| Item                      | Description                                                                                                                                  |
|---------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Macro Details Page</i> | Through the Macro Details Page, you can review details about an individual macro. In the Macros page, click the name of the macro to review. |

## Jobs

| Item                    | Description                                                                                                                                                                                                                                                                                    |
|-------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Job Details Page</i> | You can use the Job Details page to explore details about successful or failed jobs, including outputs, dependency graph, and other metadata. Download results to your local desktop or, if enabled, explore a visual profile of the data in the results for further iteration on your recipe. |
| <i>Plan Runs Page</i>   | In the Plan Runs page, you can track the status of all runs of your plans.                                                                                                                                                                                                                     |
| <i>Sample Jobs Page</i> | In the Sample Jobs page, you can track the status of all sample jobs to which you have access.                                                                                                                                                                                                 |

## Transformer

### Transformer Page

| Item | Description                                                                                                                   |
|------|-------------------------------------------------------------------------------------------------------------------------------|
|      | The data grid in the Transformer Page displays how your current recipe applies to the data in your currently selected sample. |



|                              |                                                                                                                                                                                                                                                                                                |
|------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Data Grid Panel</i>       |                                                                                                                                                                                                                                                                                                |
| <i>Recipe Navigator</i>      | Through the Recipe Navigator, you can locate and open any recipe from the current flow in the Transformer page. To open the Recipe Navigator, select the drop-down next to the name of the current recipe in the Transformer page.                                                             |
| <i>Transformer Toolbar</i>   | At the top of the data grid and the column browser, the Transformer toolbar provides quick access to common transformations.                                                                                                                                                                   |
| <i>Column Menus</i>          | You can use the menus available in a column context menu to perform a variety of actions on the column or columns.                                                                                                                                                                             |
| <i>Column Browser Panel</i>  | Through the Column Browser, you can use data quality bars and data type information to perform basic review of data across many columns. You can use these tools to select data of interest for display in the data grid or Column Details views or to prompt for suggestions of recipe steps. |
| <i>Column Details Panel</i>  | In the Column Details panel, you can review additional details about a column of your dataset. Select <b>Column Details</b> from any column menu or the Action menu in the column browser.                                                                                                     |
| <i>Transform Preview</i>     | When you create or edit a transform, the data grid displays a preview of results of the transform. <b>Transform previews</b> assist in specifying and validating the transformation steps before they are applied.                                                                             |
| <i>Context Panel</i>         | On the right side of the Transformer page, the context panel displays one of multiple panels, depending on the current state or selection of the data grid.                                                                                                                                    |
| <i>Filter Panel</i>          | As needed, you can filter the rows and columns displayed in the data grid. To review and apply filters, click the Filter icon in the Transformer toolbar.                                                                                                                                      |
| <i>Visible Columns Panel</i> | In the status bar of the Transformer page, click the Eye icon to review the list of visible and hidden columns.                                                                                                                                                                                |
| <i>Join Window</i>           | In the Join window of the Designer Cloud® application , you can join your current dataset with another dataset or recipe based upon information that is common to both datasets.                                                                                                               |
| <i>Union Page</i>            | In the Union page, you can append data from one or more datasets to an existing dataset.                                                                                                                                                                                                       |
| <i>Run Job Page</i>          | In the Run Job page, you can specify transformation and profiling jobs for the currently loaded recipe. Available options include output formats and output destinations.                                                                                                                      |

## Data Grid Panel

| Item                      | Description                                                                                                                                                                                                                                                                                          |
|---------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Column Histograms</i>  | The bar chart at the top of each column in the Transformer page, called a <b>histogram</b> , characterizes the data in that column. Each column histogram displays the count of each detected value in the column (for string data) or the count of values within a numeric range (for number data). |
| <i>Data Quality Bars</i>  | Just below the column name in the data grid is a horizontal band, which identifies data quality issues among the sample values in the column.                                                                                                                                                        |
| <i>Lookup Wizard</i>      | Through the Transformer page, you can perform lookups from one set of values in your dataset into another set of values in another dataset using a simple wizard. To perform a lookup, select the caret next to a column title, and then select <b>Lookup....</b>                                    |
| <i>Standardize Page</i>   | Through the Standardize page, you can review similar column values and standardize them to values that you specify.                                                                                                                                                                                  |
| <i>Custom Type Dialog</i> | Through the data type menu in each column, you can select different data types, including custom ones. You can also create custom data types in Designer Cloud powered by Trifacta® Enterprise Edition.                                                                                              |

## Column Menus

| Item | Description |
|------|-------------|
|      |             |

|                                       |                                                                                                                                                                                                                             |
|---------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Choose Datetime Format Dialog</i>  | In the Datetime Format dialog, you can specify or locate the format that you'd like to use for validation of the values in the current column against the Datetime data type.                                               |
| <i>Transformation by Example Page</i> | In the Transform by Example page, you can build new columns of data by specifying values to map from the selected source column. For the column to transform, select <b>Create column by examples</b> from the column menu. |

## Context Panel

| Item                           | Description                                                                                                                                                                                                                                                     |
|--------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Recipe Panel</i>            | Through the Recipe panel, you can review and modify the steps of the recipe that you have already created and add new steps to your recipe at the current location. You can also flag a step that requires review without which jobs cannot be run.             |
| <i>Transform Builder</i>       | The Transform Builder enables you to search for transformations and to rapidly assemble complete transform steps through a simple menu-driven interface.                                                                                                        |
| <i>Search Panel</i>            | Through the search context panel, you can locate transformations to specify and add at the current location in your recipe.                                                                                                                                     |
| <i>Edit History Panel</i>      | Through the Edit History panel, you can review the sequence of edits to the current recipe by individual contributors. This panel assists in determining who made which changes and when they were made.                                                        |
| <i>Samples Panel</i>           | For smaller datasets, the Transformer page displays the entire dataset. For larger ones, the source data is sampled for use in the Transformer page. Through the Samples panel, you can create new samples and select them for display in the Transformer page. |
| <i>Selection Details Panel</i> | In the Selection Details panel, you can review an active profile of your current selection or selections in the data grid or column browser and review patterns and suggestions for transformations.                                                            |

## Preferences

### Preferences Page

| Item                            | Description                                                                                                                                                                                                                                                       |
|---------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>User Profile Page</i>        | In your user profile, you can review your personal information and update your photo. Select <b>User menu &gt; Preferences</b> .                                                                                                                                  |
| <i>Account Settings Page</i>    | In your Account Settings page, you can change your locale and modify other settings related to your account.                                                                                                                                                      |
| <i>Email Notifications Page</i> | You can configure the notifications that are sent to your email address based on job success or failure. Notifications can be sent for jobs executed from flows where you are the owner or a collaborator.                                                        |
| <i>Access Tokens Page</i>       | From the Access Token page, you can generate and manage access tokens that apply to your account. Access tokens can be used when accessing the REST APIs.                                                                                                         |
| <i>Databricks Settings Page</i> | To access a Databricks cluster for running jobs, each user of the Designer Cloud powered by Trifacta® platform must insert their personal access token into their profile. This configuration enables the user to authenticate to a connected Databricks cluster. |

### User Profile Page

| Item                       | Description                                                                                                           |
|----------------------------|-----------------------------------------------------------------------------------------------------------------------|
| <i>Storage Config Page</i> | The Storage Config page allows you to configure storage locations for where you upload datasets and generate results. |

## Connections Page

| Item                           | Description                                                                                                                                    |
|--------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Share Connection Dialog</i> | Through the Share Connection dialog, users of the selected connection with appropriate privileges can modify who has access to the connection. |

# Supported File Formats

## Contents:

- *Filenames*
- *Native Input File Formats*
  - *Converted file formats*
- *Native Output File Formats*
- *Compression Algorithms*
  - *Read Native File Formats*
  - *Write Native File Formats*
  - *Snappy compression formats*
- *Additional Configuration for File Format Support*
  - *Publication of some formats requires execute permissions*

This section contains information on the file formats and compression schemes that are supported for input to and output of Designer Cloud powered by Trifacta® Enterprise Edition.

**NOTE:** To work with formats that are proprietary to a desktop application, such as Microsoft Excel, you do not need the supporting application installed on your desktop.

## Filenames

**NOTE:** During import, the Designer Cloud application identifies file formats based on the extension of the filename. If no extension is provided, the Designer Cloud application assumes that the submitted file is a text file of some kind. Non-text file formats, such as Avro and Parquet, require filename extensions.

**NOTE:** Filenames that include special characters can cause problems during import or when publishing to a file-based datastore.

## Forbidden characters in import filenames:

The following list of characters present issues in the listed area of the product. If you encounter issues, the following listings may provide some guidance on where the issue occurred.

**Tip:** You should avoid using any of these characters in your import filenames. This list may not be complete for all available running environments.

- **General:**

" / "

- **Web browser:**

" \ "

- **Excel filenames:**

```
"##"
```

- **Spark-based running environment:**

```
"{" , " * " , " \ "
```

## Native Input File Formats

Designer Cloud powered by Trifacta® Enterprise Edition can read and import directly these file formats:

- CSV
- JSON v1, including nested

**NOTE:** JSON files can be read natively but often require additional work to properly structure into tabular format. Depending on how the Designer Cloud application is configured (v1 or v2), JSON files may require conversion before they are available for use in the application. See "Converted file formats" below.

**NOTE:** Designer Cloud powered by Trifacta Enterprise Edition requires that JSON files be submitted with one valid JSON object per line. Consistently malformed JSON objects or objects that overlap linebreaks might cause import to fail. See *Initial Parsing Steps* in the User Guide

- Plain Text
- LOG
- TSV
- Parquet

**NOTE:** When working with datasets sourced from Parquet files, lineage information and the `$sourcerownumber` reference are not supported.

- Avro
- XML

**Tip:** XML files can be ingested as unstructured text.

- Google Sheets

**NOTE:** Individual users must enable access to their Google Drive. No data other than Google Sheets is read from Google Drive.

See *Import Google Sheets Data*.

For more information on data is handled initially, see *Initial Parsing Steps* in the User Guide.

## Converted file formats

Files of the following type are not read into the product in their native format. Instead, these file types are converted using the Conversion Service into a file format that is natively supported, stored in the base storage layer, and then ingested for use in the product.

**NOTE:** Compressed files that require conversion of the underlying file format are not supported for use in the product.

Converted file formats:

- Excel (XLS/XLSX)

**NOTE:** Other Excel-related formats, such as XLSM format, are not supported.

**Tip:** You may import multiple worksheets from a single workbook at one time. See *Import Excel Data* in the User Guide.

- Google Sheets

**Tip:** You may import multiple sheets from a single Google Sheet at one time. See *Import Google Sheets Data* in the User Guide.

- PDF

**NOTE:** PDF support may need to be enabled in your environment. See *Import PDF Data*.

- PDF is supported for import only.
- See *Import PDF Data* in the User Guide.

- JSON v2

### Notes on JSON:

There are two methods of ingesting JSON files for use in the product.

- JSON v2 - This newer version reads the JSON source file through the Conversion Service, which stores a restructured version of the data in tabular format on the base storage layer for quick and simple use within the application.

**Tip:** This method is enabled by default and is recommended. For more information, see *Working with JSON v2*.

- JSON v1 - This older version reads JSON files directly into the platform as text files. However, this method often requires additional work to restructure the data into tabular format. For more information, see *Working with JSON v1*.

## Native Output File Formats

Designer Cloud powered by Trifacta Enterprise Edition can write to these file formats:

- CSV
- JSON
- Tableau Hyper

**NOTE:** Publication of results in Hyper format may require additional configuration. See below.

- Avro

**NOTE:** The Trifacta Photon and Spark running environments apply Snappy compression to this format.

- Parquet

**NOTE:** The Trifacta Photon and Spark running environments apply Snappy compression to this format.

## Compression Algorithms

When a file is imported, the Designer Cloud application attempts to infer the compression algorithm in use based on the filename extension. For example, `.gz` files are assumed to be compressed with GZIP.

**NOTE:** Import of a compressed file whose underlying format requires conversion through the Conversion Service is not supported.

**NOTE:** Importing a compressed file with a high compression ratio can overload the available memory for the application. In such cases, you can decompress the file before uploading. If decompression fails, you should contact your administrator about increasing the Java Heap Size memory.

**NOTE:** Publication of results in Snappy format may require additional configuration. See below.

**NOTE:** GZIP files on Hadoop are not split across multiple nodes. As a result, jobs can crash when processing it through a single Hadoop task. This is a known issue with GZIP on Hadoop.

Where possible, limit the size of your GZIP files to 100 MB or less, or use BZIP2 as an alternative compression method. As a workaround, you can try to run the job on the unzipped file. You may also disable profiling for the job. See *Run Job Page* in the User Guide.

**Tip:** If preferred, you can configure the Designer Cloud application to infer the compression scheme based on first few bytes of the file. For more information, see *Miscellaneous Configuration*.

## Read Native File Formats

|         | GZIP          | BZIP          | Snappy        | Notes                                   |
|---------|---------------|---------------|---------------|-----------------------------------------|
| CSV     | Supported     | Supported     | Supported     |                                         |
| JSON v2 | Not supported | Not supported | Not supported | A converted file format. See above.     |
| JSON v1 | Supported     | Supported     | Supported     | Not a converted file format. See above. |
| Avro    |               |               | Supported     |                                         |
| Hive    |               |               | Supported     |                                         |

## Write Native File Formats

|      | GZIP      | BZIP      | Snappy               |
|------|-----------|-----------|----------------------|
| CSV  | Supported | Supported | Supported            |
| JSON | Supported | Supported | Supported            |
| Avro |           |           | Supported; always on |
| Hive |           |           | Supported; always on |

## Snappy compression formats

Designer Cloud powered by Trifacta Enterprise Edition supports the following variants of Snappy compression format:

| File extension | Format name          | Notes                                                                                                                                                                                                                                   |
|----------------|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| .sz            | Framing2 format      | See: <a href="https://github.com/google/snappy/blob/master/framing_format.txt">https://github.com/google/snappy/blob/master/framing_format.txt</a>                                                                                      |
| .snappy        | Hadoop-snappy format | See: <a href="https://code.google.com/p/hadoop-snappy/">https://code.google.com/p/hadoop-snappy/</a><br><br><b>NOTE:</b> Xerial's snappy-java format, which is also written with a .snappy file extension by default, is not supported. |

## Additional Configuration for File Format Support

### Publication of some formats requires execute permissions

When job results are generated and published in the following formats, the Designer Cloud powered by Trifacta platform includes a JAR, from which is extracted a binary executable into a temporary directory. From this directory, the binary is then executed to generate the results in the proper format. By default, this directory is set to `/tmp` on the Trifacta node.

In many environments, execute permissions are disabled on `/tmp` for security reasons. Use the steps below to specify the temporary directory where this binary can be moved and executed.

#### Steps:

1. Login to the application as an administrator.
2. From the menu, select **User menu > Admin console > Admin settings**.
3. For each of the following file formats, locate the listed parameter, where the related binary code can be executed:



| File Format | Parameter                     | Setting to Add                                            |
|-------------|-------------------------------|-----------------------------------------------------------|
| Snappy      | "data-service.<br>jvmOptions" | -Dorg.xerial.snappy.tmpdir=<some executable<br>directory> |
| Hyper       | See previous.                 | See previous.                                             |

4. Save your changes and restart the platform.
5. Run a job configured for direct publication of the modified file format.

# Column Statistics Reference

## Contents:

- *General Column Counts*
- *General Column Statistics*
- *Aggregation Functions*

This page describes the statistical information available for individual columns of data.

- Statistics may vary depending on the column's data type. For example, the statistics retained for states may be different from the statistics for strings.
- Most of these statistics are available in the column details panel, which can be opened from the left side of the Transformer page. See *Column Details Panel*.

Below, you can review general statistics maintained for each data type, followed by breakdowns of statistics for each specific type of data.

**NOTE:** Before your job is run, profiling information such as column statistics are exact counts of the sample that is currently loaded. After the job is run, profiled results in the Job Results page might include estimates for some metrics and counts, depending on the scale of the dataset.

## General Column Counts

For any selection of values in a column, the following counts are generally available.

| Count Name        | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid Values      | Count of values that are valid for the column's data type                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| Unique Values     | Count of unique values. Duplicate values are not counted.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| Outlier Values    | Count of values that qualify as outliers. An <b>outlier</b> value is either: <ul style="list-style-type: none"><li>• <math>&lt; (25\text{th percentile}) - (2 * \text{IQR})</math></li><li>• <math>&gt; (75\text{th percentile}) + (2 * \text{IQR})</math></li><li>• <b>IQR (interquartile range)</b> is the range of values between the two middle quarters, which is equivalent to the range between the 25th and 75th percentiles. Thus, in the above computations, the IQR factor ensures that the outliers are at the extremes of the entire range.</li></ul> |
| Mismatched Values | Count of values that do not confirm to the column's data type. For example, an Integer column with a value of "MISSING" results in a mismatched value.                                                                                                                                                                                                                                                                                                                                                                                                             |
| Missing Values    | Count of values that are not populated                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |

## General Column Statistics

These statistics are available for most types of data through the Column Browser.

- For string types (String, Phone Number, Social Security Number, Boolean, Email Address, Credit Card Number, Gender, IP Address, URL, HTTP Code, Date/Time), these stats measure string length.

- For structured string types (Phone Number, Social Security Number, Boolean, Gender, IP Address, HTTP Code, Date/Time), any variation in these numbers indicates data problems.
- Does not apply to: State

| Statistic Name     | Description                                                                                                                                                                                                                                                                              |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Minimum            | Lowest value in the column                                                                                                                                                                                                                                                               |
| Lower Quartile     | The median of the lower half of values (25th percentile)                                                                                                                                                                                                                                 |
| Median             | <p>The middle value of the selected set. For example, in a set of 21 values, the median value is the 11th value in ascending order.</p> <ul style="list-style-type: none"> <li>• For datasets with an even number of values, the median is the mean of the two middle values.</li> </ul> |
| Upper Quartile     | The median of the upper half of values (75th percentile)                                                                                                                                                                                                                                 |
| Maximum            | Highest value in the column                                                                                                                                                                                                                                                              |
| Average            | Average value in the column                                                                                                                                                                                                                                                              |
| Standard Deviation | The computed standard deviation for the selected values.                                                                                                                                                                                                                                 |

## Aggregation Functions

The following functions can be applied to a set of columnar data taken from one or more columns in your dataset. Unless otherwise noted, these functions apply to numeric data. For more information, see *Aggregate Functions*.

# Supported Data Types

Designer Cloud powered by Trifacta® Enterprise Edition supports the following data types.

**Tip:** Transforms that include functions work only if the inputs are of a data type and format valid for the function. You should clean up the type and format of your columns before you apply transformations to them.

For more information on how to explicitly reference these data types in your steps, see *Valid Data Type Strings*.

## Supported Data Types

| Item                                    | Description                                                                                                                                                                                                                                                                                 |
|-----------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>String Data Type</i>                 | Any non-null value can be typed as String. A String can be anything.                                                                                                                                                                                                                        |
| <i>Integer Data Type</i>                | The Integer data type applies to positive and negative numeric values that have no decimal point.                                                                                                                                                                                           |
| <i>Decimal Data Type</i>                | Decimal data type applies to floating points up to 15 digits in length. <ul style="list-style-type: none"><li>• In the Designer Cloud application , this data type is referenced as <code>Decimal</code>.</li><li>• In storage, this data type is written as <code>Double</code>.</li></ul> |
| <i>Boolean Data Type</i>                | The Boolean data type expresses true or false values.                                                                                                                                                                                                                                       |
| <i>Social Security Number Data Type</i> | This data type is applied to numeric data following the pattern for United States Social Security numbers.                                                                                                                                                                                  |
| <i>Phone Number Data Type</i>           | This data type is applied to numeric data following common patterns that express telephone numbers and known valid phone numbers in the United States.                                                                                                                                      |
| <i>Email Address Data Type</i>          | This data type matches text values that are properly formatted email addresses.                                                                                                                                                                                                             |
| <i>Credit Card Data Type</i>            | Credit card numbers are numeric data that follow the 14-digit or 16-digit patterns for credit cards.                                                                                                                                                                                        |
| <i>Gender Data Type</i>                 | This data type matches a variety of text patterns for expressing male/female distinctions.                                                                                                                                                                                                  |
| <i>Zip Code Data Type</i>               | This data type matches five- and nine-digit U.S. zipcode patterns.                                                                                                                                                                                                                          |
| <i>State Data Type</i>                  | State data type is applied to data that uses the full names or the two-letter abbreviations for states in the United States.                                                                                                                                                                |
| <i>Object Data Type</i>                 | An <b>Object</b> data type is a method for encoding key-value pairs. A single field value may contain one or more sets of key-value pairs.                                                                                                                                                  |
| <i>Array Data Type</i>                  | An <b>array</b> is a list of values grouped into a single value. An array may be of variable length; in one record the array field may contain two elements, while in the next record, it contains six elements.                                                                            |
| <i>IP Address Data Type</i>             | The IP Address data type supports IPv4 address.                                                                                                                                                                                                                                             |
| <i>URL Data Type</i>                    | URL data type is applied to data that follows generalized patterns of URLs.                                                                                                                                                                                                                 |
| <i>HTTP Code Data Type</i>              | Values of these data types are three-digit numeric values, which correspond to recognized HTTP Status Codes.                                                                                                                                                                                |
| <i>Datetime Data Type</i>               | Designer Cloud powered by Trifacta® Enterprise Edition supports a variety of Datetime formats, each of which has additional variations to it.                                                                                                                                               |

## Custom Types

If you have created a custom type, it is available for selection from the column type drop-down.

**NOTE:** After a custom type has been created, a platform restart is required. Please contact your Trifacta administrator.

- A custom type can be created based on a dictionary file. If a value is contained in the type's dictionary, the value is considered valid. See *Create Custom Data Types*.
- Developers may also define custom data types using regular expressions. See *Create Custom Data Types Using RegEx*.

# String Data Type

Any non-null value can be typed as String. A String can be anything.

**NOTE:** If a column of values fails to match another data type, the column is typed as String data.

The length of an individual string value is limited only by the limit applied to an entire row of data, which is approximately 1 MB of characters.

**NOTE:** On export to relational systems, string lengths are limited to 256 for performance reasons.

As needed, this limit can be raised for all applicable relational systems. For more information, see *Configure Data Service*.

# Integer Data Type

The Integer data type applies to positive and negative numeric values that have no decimal point.

- Punctuation such as commas and dollar signs (\$) are not supported. These markers must be removed from numeric values through transform steps before you can change the type to Integer.
- The following range is considered safe for values of this type. There may be inconsistencies in output for values outside this range:
  - **Safe minimum:**  $-9007199254740991$  ( $-2^{53} + 1$ )
  - **Safe maximum:**  $9007199254740991$  ( $2^{53} - 1$ )

**NOTE:** Scientific notation is not supported for Integer data type. Please use Decimal data type instead.

## Examples:

- 4
- -23
- 1234567890123456
- -1234567890123456

# Decimal Data Type

Decimal data type applies to floating points up to 15 digits in length.

- In the Designer Cloud application , this data type is referenced as `Decimal`.
- In storage, this data type is written as `Double`.

Punctuation such as commas and dollar signs (\$) are not supported. These markers must be removed from numeric values through transformation steps before you can change the type to Decimal.

The following range is considered safe for values of this type. There may be inconsistencies in output for values outside this range:

- **Safe minimum:** `4.9406564584124654e-324`
- **Safe maximum:** `1.7976931348623157e+308`

## Notes:

- Decimal values that are longer than 15 digits are treated as String values and may appear as mismatched values in a Decimal column.
- Scientific notation is supported.
- Designer Cloud powered by Trifacta Enterprise Edition utilizes Java's Float data type for its Decimal data validation, which may result in some loss of precision in rare cases.

## Examples:

- `123.45`
- `3000.00`
- `1.2345678E+22`
- `7.4423e-12`
- `-4.123e+12`



# Boolean Data Type

The Boolean data type expresses true or false values.

## Supported Values:

Values can also be expressed as combinations of the following (e.g. `True` / `f`).

- `true` / `false`
- `True` / `False`
- `t`/`f`
- `T`/`F`
- `yes` / `no`
- `Yes` / `No`
- `y`/`n`
- `Y` / `N`
- `on` / `off`
- `1` / `0`

# Social Security Number Data Type

This data type is applied to numeric data following the pattern for United States Social Security numbers.

**Supported Patterns and Delimiters:**

| Delimiter | Example     |
|-----------|-------------|
| dash      | ###-##-#### |
| space     | ### ## #### |
| no space  | #####       |

## Data Validation

When values are validated against the Social Security Number data type, the Designer Cloud® application validates the values using regular expressions. This regular expression method checks for general Social Security Number patterns and is fast to evaluate.

However, some values may follow the regular expression validation pattern but are not accurate social security numbers. These values may be detected as valid values.

# Phone Number Data Type

This data type is applied to numeric data following common patterns that express telephone numbers and known valid phone numbers in the United States.

- Parentheses are optional around area code values: ( ### ) ###-####
- Phone numbers may or may not include the 1 at their beginning.
- Dashes and spaces are optional between groups of numbers.

**NOTE:** Spaces within a group of values are not supported and will result in a mismatched data type entry.

## Standard Domestic U.S.:

```
###-###-####
(###)-###-####

1#####
```

## Standard Domestic U.S. with extensions:

```
(###)-###-#### x ###
```

- The dash in ###-#### is required when extension is present.
- The extension indicator can be one of the following: x, X, ext, or EXT.

## International U.S.:

```
+1(###)-###-####
```

## Not supported:

- Local U.S. format not supported:

```
###-####
```

- International format:

```
+##-##-###-####
```

- Internal international format not supported:

```
##-##-##-###-####
```

## Data Validation

In addition to validation against common phone patterns, values are checked against permitted U.S. phone numbers. For example, an area code value of 123 is invalid, as this area code value does not exist in the U.S. phone system.

# Email Address Data Type

This data type matches text values that are properly formatted email addresses.

Data in the following format is likely to be typed as email addresses:

```
String@String.aaa
String@String.aaaa
```

where:

- `String` - any set of readable characters of 1 or more characters in length
- `aaa` or `aaaa` - three-letter or four-letter suffix

# Credit Card Data Type

Credit card numbers are numeric data that follow the 14-digit or 16-digit patterns for credit cards.

## Pattern:

```

####-####-####-####

####-####-####-##
```

## Notes:

Some numerical combinations in the above patterns may be:

- widely known test values
- invalid for credit card numbers

For more information, see *Data Type Validation Patterns*.

## Data Validation

When values are validated against the Credit Card data type, the Designer Cloud® application validates the values using regular expressions. This regular expression method checks for general Credit Card patterns and is fast to evaluate.

However, some values may follow the regular expression validation pattern but are not accurate credit card numbers. These values may be detected as valid values.

# Gender Data Type

This data type matches a variety of text patterns for expressing male/female distinctions.

The following gender identifiers are supported:

## Supported Values:

- m / f
- M / F
- male / female
- Male / Female

# Zip Code Data Type

This data type matches five- and nine-digit U.S. zipcode patterns.

Zip code data follows the following possible patterns:

```

#####-####
```

Some numerical combinations of the above are not valid zipcodes. For more information, see *Data Type Validation Patterns*.

# State Data Type

State data type is applied to data that uses the full names or the two-letter abbreviations for states in the United States.

## Supported Patterns and Examples:

| Pattern       | Examples                                                                                          |
|---------------|---------------------------------------------------------------------------------------------------|
| Full names    | <ul style="list-style-type: none"><li>• California</li><li>• Arizona</li><li>• New York</li></ul> |
| Abbreviations | <ul style="list-style-type: none"><li>• CA</li><li>• AZ</li><li>• NY</li></ul>                    |

## Also supported:

| Abbreviation | Full Name            |
|--------------|----------------------|
| DC           | District of Columbia |
| PR           | Puerto Rico          |
| VI           | Virgin Islands       |



# Object Data Type

An **Object** data type is a method for encoding key-value pairs. A single field value may contain one or more sets of key-value pairs.

An Object data type is identified as a set of nested objects in the following format:

```
{ "key": "value" }
{ "New York": "NY" }
{ "California": "CA" }
```

## Notes:

- Individual values of an Object type must have unique keys. Values, however, may be repeated.
- The order of key-value pairs is not guaranteed.
- The maximum size of an Object is determined by the maximum size of a cell's value in the data grid. For more information, see *Data Grid Panel*.

**NOTE:** The Designer Cloud application recognizes up to 250 unique keys in a column of Object data type.

**NOTE:** The `\n` and `\t` escaped characters are supported in inputs for this data type. For more information, see *Supported Special Regular Expression Characters*.

# Array Data Type

An **array** is a list of values grouped into a single value. An array may be of variable length; in one record the array field may contain two elements, while in the next record, it contains six elements.

Arrays must be wrapped in square brackets; parentheses are not supported.

The maximum size of an Object is determined by the maximum size of a cell's value in the data grid. For more information, see *Data Grid Panel*.

## Examples:

```
["123" , "456" , "789"]
["Hello" , "Goodbye"]
["abc" , "2"]
["abc" , "3"]
["A" , ["B" , "C"] , "D"]
```

**Ragged arrays:** The above arrays are collectively a set of ragged arrays. The number of elements in each array varies. When arrays are ragged, some array functions may return null or empty values.

**Nested arrays:** The last example above is a nested array, in which one array is nested inside of another.

**NOTE:** The `\n` and `\t` escaped characters are supported in inputs for this data type. For more information, see *Supported Special Regular Expression Characters*.

# IP Address Data Type

The IP Address data type supports IPv4 address.

Supported format:

```
###.###.###.###
```

# URL Data Type

URL data type is applied to data that follows generalized patterns of URLs.

The domain and suffix of the URL are the only required elements (e.g. `example.com`).

## Supported URL Elements:

```
scheme:[//[user:password@]sub-domain.domain.sfx[:port]][/]path[?query][#fragment]
```

- scheme
- username
- password
- full-domain (sub-domain + domain above)
- sub-domain
- domain (required)
- suffix (sfx above) (required)
- port
- path
- query
- fragment

## Examples:

```
example.com
http://example.com
https://docs.example.com/dosearchsite.action?where=HOME&spaceSearch=true&queryString=Support
```

# HTTP Code Data Type

Values of these data types are three-digit numeric values, which correspond to recognized HTTP Status Codes.

If your column contains three-digit numbers in the following ranges, it may be typed as Http Code.

## Supported Values:

- 100 - 102
- 200 - 208
- 300 - 308
- 400 - 409
- 400 - 459
- 500 - 509

# Datetime Data Type

## Contents:

- *Date Range*
  - *Data Validation*
  - *Formatting Tokens*
    - *Two-digit year values*
  - *Supported Datetime Formats*
  - *Supported Time Zones*
  - *Job Execution*
    - *Differences between Trifacta Photon and Spark running environments*
  - *Datetime Schema via API*
- 

Designer Cloud powered by Trifacta® Enterprise Edition supports a variety of Datetime formats, each of which has additional variations to it.

## Date Range

### Supported Date Ranges:

- **Earliest:** January 1, 1400

**NOTE:** Two-digit values for the year that are older than 80 years from the current year are forward-ported into the future. For example, in a job run on Dec 31, 2021, the date `01/01/41` is interpreted as 01/01/1941. However, if the job is run the next day (January 01, 2022), then the same data is interpreted as 01/01/2041. See "Two-digit year values" below.

- **Latest:** December 31, 2599

**NOTE:** The supported date ranges can be modified if needed. For more information, see *Configure Application Limits*.

You can use dates in the Gregorian calendar system only. Dates in the Julian calendar are not supported.

## Data Validation

When values are validated against the Datetime data type, the Designer Cloud application does not compare them to an underlying calendar system. Instead, the application validates the values using regular expressions. This regular expression method checks for general Datetime validation and is fast to evaluate.

However, some values may follow the regular expression validation pattern but are not accurate dates. For example, every four years, February 29 is a valid date. When this date is validated against the Datetime data type, it may be detected as a valid value, while the date is changed in the application to be incremented to a close accurate date, such as March 1 in this example.

## Formatting Tokens

You can use the following tokens to change the format of a column of dates:

---

| Letter | Date or Time Component                                                               | Presentation | Examples                                                                                                                                                                                                                                                                                                                                                                                         |
|--------|--------------------------------------------------------------------------------------|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| M      | Month in year                                                                        | Number       | 1                                                                                                                                                                                                                                                                                                                                                                                                |
| MM     | Month in year                                                                        | Number       | 01                                                                                                                                                                                                                                                                                                                                                                                               |
| MMMM   | Month in year                                                                        | Month        | January                                                                                                                                                                                                                                                                                                                                                                                          |
| MMM    | Month in year                                                                        | Month        | Jan                                                                                                                                                                                                                                                                                                                                                                                              |
| yy     | Year                                                                                 | Number       | 16 <div> <b>NOTE:</b> Two-digit values for the year that are older than 80 years from the current year are forward-ported into the future. For example, in a job run on Dec 31, 2021, the date 01 / 01 / 41 is interpreted as 01/01/1941. However, if the job is run the next day (January 01, 2022), then the same data is interpreted as 01/01/2041. See "Two-digit year values" below. </div> |
| yyyy   | Year                                                                                 | Number       | 2016                                                                                                                                                                                                                                                                                                                                                                                             |
| D      | Day in year                                                                          | Number       | 352                                                                                                                                                                                                                                                                                                                                                                                              |
| d      | Day in month                                                                         | Number       | 9                                                                                                                                                                                                                                                                                                                                                                                                |
| dd     | Day in a month                                                                       | Number       | 09                                                                                                                                                                                                                                                                                                                                                                                               |
| EEE    | Day in week (three-letter abbreviation)                                              | Text         | Wed                                                                                                                                                                                                                                                                                                                                                                                              |
| EEEE   | Day in week                                                                          | Text         | Wednesday                                                                                                                                                                                                                                                                                                                                                                                        |
| h      | Hour in day (1-12) <div> <b>NO TE:</b> Requires an AM /PM indicator (a ). </div>     | Number       | 2                                                                                                                                                                                                                                                                                                                                                                                                |
| hh     | Hour in am /pm (01-12) <div> <b>NO TE:</b> Requires an AM /PM indicator (a ). </div> | Number       | 02                                                                                                                                                                                                                                                                                                                                                                                               |
| H      | Hour in day                                                                          | Number       | 2                                                                                                                                                                                                                                                                                                                                                                                                |

|     |                    |                    |          |
|-----|--------------------|--------------------|----------|
|     | (1-12)             |                    |          |
| HH  | Hour in day (0-23) | Number             | 20       |
| m   | Minute in an hour  | Number             | 9        |
| mm  | Minute in an hour  | Number             | 09       |
| s   | Second in a minute | Number             | 3        |
| ss  | Second in a minute | Number             | 03       |
| SSS | Millisecond        | Number             | 218      |
| X   | Time zone          | ISO 8601 time zone | -08 : 00 |
| a   | AM/PM indicator    | String             | AM       |

**NOTE:** When publishing to relational targets, Datetime values are written as date/time values in newly created tables. If you are appending to a relational table column that is in timestamp format, Datetime values can be written as timestamps.

**Tip:** If your DateTime column contains data in multiple formats, you must change the format of the DateTime column to one format and then add a transformation to convert that data to the other format. When all formats of your source date values are converted to a single format, the application should infer the appropriate date and time format.

### Supported Separators:

- Date separators: blank space, comma, single hyphen, or forward slash
- Time separators: blank space, comma, single hyphen, colon, t or T
- Non-delimited Datetime values are supported. For example, `yyyymmdd, yyyymmddThhmmssX`.

### ISO 8601 Time Zone Notes:

- Support for timezone offset from UTC indicated by `+hh:mm`, `+hhmm`, or `+hh`. For example, the date '2013-11-18 11:55-04:00' is recognized as a DateTime value.
- Datetime part functions (for example, Hour) truncate time zones and return local time.
- If you have a column with multiple time zones, you can convert the column to Unixtime so you can perform Date/Time operations with a standardized time zone. If you want to work with local times, you can truncate the time zone or use other Datetime functions. See *UNIXTIME Function*.

### Two-digit year values

Depending on the system, a two-digit value for year in a Datetime value is subject to different interpretations. In Designer Cloud powered by Trifacta Enterprise Edition, two-digit values for the year that are older than 80 years from the current year are forward-ported into the future. For example, in a job run on Dec 31, 2021, the date `01/01/41` is interpreted as 01/01/1941. However, if the job is run the next day (January 01, 2022), then the same data is interpreted as 01/01/2041.

Other systems use different limits for backward versus forward porting of year values:



- In BigQuery, if no century value is provided, then the year value has a century value applied to it based on a fixed range. See [https://cloud.google.com/bigquery/docs/reference/standard-sql/format-elements#:~:text=The%20year%20without%20century%20as%20a%20decimal%20number%20\(00%2D99\)](https://cloud.google.com/bigquery/docs/reference/standard-sql/format-elements#:~:text=The%20year%20without%20century%20as%20a%20decimal%20number%20(00%2D99))
- Snowflake permits customization of two-digit year values at the Account, Session, or Object level. See <https://docs.snowflake.com/en/sql-reference/parameters.html#two-digit-century-start>.

As a result, it can be a challenge to manage these system-dependent two-digit years in a consistent manner.

**Tip:** For best results, you should format year values as four-digit values before the data is ingested into Designer Cloud powered by Trifacta Enterprise Edition. Four-digit years are consistently represented across all systems.

If the above is not possible, you can create replacement steps in your recipe to convert two-digit years to four-digit values. In the following example, 00–39 is interpreted as a 19XX year, while 40–99 is interpreted as a 20XX year:

|                                |                         |
|--------------------------------|-------------------------|
| <b>Transformation Name</b>     | Replace text or pattern |
| <b>Parameter: Column</b>       | myDateColumn            |
| <b>Parameter: Find</b>         | /\b([456789][0-9])\b\$/ |
| <b>Parameter: Replace with</b> | 19\$1                   |

and

|                                |                         |
|--------------------------------|-------------------------|
| <b>Transformation Name</b>     | Replace text or pattern |
| <b>Parameter: Column</b>       | myDateColumn            |
| <b>Parameter: Find</b>         | /\b([0123][0-9])\b\$/   |
| <b>Parameter: Replace with</b> | 20\$1                   |

## Supported Datetime Formats

For more information on the available formats and examples of each one, see *Datetime Formats (PDF)*.

For more information on supported date formatting strings, see *DATEFORMAT Function*.

## Supported Time Zones

For more information, see *Supported Time Zone Values*.

## Job Execution

Datetime data typing involves the basic type definition, plus any supported formatting options. Depending on where the job is executed, there may be variation in how the Datetime data type is interpreted.

- Some running environments may perform additional inference on the typing.

**NOTE:** During job execution on Spark, inputs of Datetime data type may result in row values being inferred for data type individually. For example, the String value 01/10/2020 may be inferred by date transformations as 1st Oct, 2020 or 10th Jan, 2020. Resulting outputs of Datetime values may not be deterministic in this scenario.

- Some formatting options may not be supported.

## Differences between Trifacta Photon and Spark running environments

If your Datetime data does not contain time zone information, by default:

- Spark uses the time zone of the Trifacta node for Datetime values.
- Trifacta Photon uses the UTC time zone for Datetime values.

This difference in how the values are treated can result in differences in Datetime-based calculations, such as the DATEDIF function.

### Workarounds:

You can do one of the following:

- Set the time zone for the Trifacta node to be UTC. You must also set the time zone for your Spark running environment to UTC.
- Apply the following Spark property overrides:

```
"spark": {
 "props": {
 ...
 "spark.driver.extraJavaOptions" : "-Duser.timezone=\"UTC\"",
 "spark.executor.extraJavaOptions" : "-Duser.timezone=\"UTC\""
 }
 ...
}
```

For more information, see *Spark Execution Properties Settings*.

## Datetime Schema via API

When Datetime data is returned via API calls, the schema for this information is returned as a three-element array. The additional elements to the specific are required to account for formatting options of for Datetime values.

**Tip:** Schema information for data types is primarily available via API calls. You may find schema information for columns in JSON versions of the visual profile and flow definitions when they are exported.

Example:

```
"end_date": [
 "Datetime",
 "mm-dd-yy",
 "mm*dd*yyyy"
]
```

| Array Element | Description                                                                                                           | Example 1      | Example 2  |
|---------------|-----------------------------------------------------------------------------------------------------------------------|----------------|------------|
| Data type     | The internal name for the data type. For Datetime columns, this schema value should always be <code>Datetime</code> . | "Datetim<br>e" | "Datetime" |

|             |                                              |                  |                        |
|-------------|----------------------------------------------|------------------|------------------------|
| Sub-format  | The general format category of the data type | "mm-dd-yy"       | "mm-dd-yy"             |
| Format type | The specific formatting for the data type    | "mm*dd*y<br>yyy" | "shortMonth*d<br>d*yy" |

# Supported Numeric Formatting

## Contents:

- *Supported Key Codes*
- *Key Codes as Separator Values*
  - *Separators for locales*
  - *Example Separators*

The following formatting can be applied to Integer and Decimal types or to String values that are being converted to numeric types.

**Tip:** Designer Cloud powered by Trifacta Enterprise Edition supports Java number formatting strings, with some exceptions.

## Supported Key Codes

| Code    | Description                                                                                                                                                                                                                                                                                                                                | Example Format String                                  | Example Inputs                     | Example Outputs                              |
|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------|------------------------------------|----------------------------------------------|
| #       | Insert a digit if it is present in the data.                                                                                                                                                                                                                                                                                               | '###,###'                                              | 99<br>999<br>1000<br>10000         | 99<br>999<br>1,000<br>10,000                 |
| 0       | Indicate required digits. If a digit is not available in the source, inserts zero in the data.                                                                                                                                                                                                                                             | '00.##'                                                | 20<br>7.1                          | 20.00<br>07.1                                |
| \$      | You can add constants values to the expression. For example, you can insert currency markers at the beginning of your expression.<br><br><b>NOTE:</b> The following currency formats are supported: \$, "€", "£", "¥", "", "", "NT\$", "R\$", "R", "Rs", "Kr<br><br>Whitespace is respected, except in the following case.                 | '\$ ##.##'                                             | 20<br>2514.22<br>6.6666            | \$ 20<br>\$ 2514.22<br>\$ 6.67               |
| (space) | You can use space as a grouping separator. When space is used to group sets of digits, all other whitespace in the value is trimmed.                                                                                                                                                                                                       | '\$ ###.##' where space is used as grouping separator. | 123456.78<br>£<br>123456.78        | \$123<br>456.78<br>\$123<br>456.78           |
| %       | Percentage expressions can be at the back of the number formatting expression.<br><br><b>NOTE:</b> When the percentage sign is added to the format string, the value is automatically multiplied by 100. When the format string is used with the NUMVALUE function, the value is automatically divided by 100 to return the decimal value. | '##.## %'                                              | 0.20<br>14.22<br>6.6666            | 20 %<br>1422 %<br>666.67 %                   |
| -       | Negative value indicators can be added to the front part of the number formatting string. <ul style="list-style-type: none"><li>• Negative value indicators at the end of the number are not supported.</li></ul>                                                                                                                          | '-###,###.00'                                          | 123<br>-123<br>1234.56<br>-1234.56 | -123.00<br>--123.00<br>-1234.56<br>--1234.56 |

- If the source value is positive, the negative value is rendered.

**NOTE:** In this case, the source value is formatted to appear as its opposite.

- If the source value is negative, a second dash is added to the front of the value. See examples.

**NOTE:** In this case, the value is formatted as a non-numeric value. You can add a second step to your recipe to remove the second dash from column values.

**NOTE:** After the formatting has been applied, type inference may be re-applied to the column, which can change the data type of the column.

## Key Codes as Separator Values

Some functions support the use of specifying key codes for grouping and decimal separators.

- For more information, see *NUMFORMAT Function*.
- For more information, see *NUMVALUE Function*.

**NOTE:** Separators must be specified when using the NUMVALUE function.

## Separators for locales

Grouping and decimal separators can be used to format values for specific locales. Below, you can see how you can format values for locales.

| Example Locale | Grouping Separator | Decimal Separator | Example Formatting                                                           | Example Output |
|----------------|--------------------|-------------------|------------------------------------------------------------------------------|----------------|
| U.S locale     | Comma ( , )        | Period ( . )      | <code>NUMFORMAT(SUM(1000000,DIVIDE(1,100)), '###,###.00',' ',' ',' ')</code> | 1,000,000.01   |
| Spanish locale | Period ( . )       | Comma ( , )       | <code>NUMFORMAT(SUM(1000000,DIVIDE(1,100)), '###,###.00',' ',' ',' ')</code> | 1.000.000,01   |
| French locale  | Space              | Comma ( , )       | <code>NUMFORMAT(SUM(1000000,DIVIDE(1,100)), '###,###.00',' ',' ',' ')</code> | 1 000 000,01   |

## Example Separators

| Input | Example Format String | Grouping Separator | Decimal Separator | Output |
|-------|-----------------------|--------------------|-------------------|--------|
|       |                       |                    |                   |        |

|         |          |       |   |          |
|---------|----------|-------|---|----------|
| 123.45  | ##.00    | ,     | . | 123.45   |
| 123.4   | ##.00    | ,     | . | 123.40   |
| 1234    | #,###    | ,     | . | 1,234    |
| 1234.5  | #,###.#  | ,     | . | 1,234.5  |
| 1234.56 | #,###.## | ,     | . | 1,234.56 |
| 1234    | ###,#    | .     | , | 1.234    |
| 1234.56 | ###,#    | .     | , | 1.234,56 |
| 1234    | #,##     | .     | , | 1.234    |
| 1234    | #.###,0  | .     | , | 1.234,0  |
| 123.45  | ##,#     | space | , | 123,45   |
| 1234    | # ###    | space | , | 1 234    |
| 1234.5  | # ###,#  | space | , | 1 234,5  |
| 1234.56 | # ###,## | space | , | 1 234,56 |

# Type Conversions

## Contents:

- *Import*
  - *Type Inference*
  - *Export*
  - *Supported Data Types*
- 

## Import

When data is imported:

- Supported data types from the source are converted to corresponding data types supported by the application, based upon the conversions listed in this section.
- Types that are not supported but are recognized by the application are converted to String types.
- Data for types that cannot be read from the source due to technical reasons are converted to null values on import.

## Type Inference

By default, the Designer Cloud application applies type inference for imported data. The application attempts to infer a column's appropriate data type in the application based on a review of the first lines in the sample. For information on how data types are inferred, see *Overview of the Type System*.

## Export

On export from the Designer Cloud application :

- The application maps the internal Trifacta data type to the explicit type listed in the appropriate page in this section.
- Unmapped types are converted to the equivalent of strings.

**Tip:** You can import a target schema to assist in lining up your columns with the expected target. For more information, see *Overview of RapidTarget*.

## Supported Data Types

| Item                                    | Description                                                                                                                                                                                                                                                                                 |
|-----------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>String Data Type</i>                 | Any non-null value can be typed as String. A String can be anything.                                                                                                                                                                                                                        |
| <i>Integer Data Type</i>                | The Integer data type applies to positive and negative numeric values that have no decimal point.                                                                                                                                                                                           |
| <i>Decimal Data Type</i>                | Decimal data type applies to floating points up to 15 digits in length. <ul style="list-style-type: none"><li>• In the Designer Cloud application , this data type is referenced as <code>Decimal</code>.</li><li>• In storage, this data type is written as <code>Double</code>.</li></ul> |
| <i>Boolean Data Type</i>                | The Boolean data type expresses true or false values.                                                                                                                                                                                                                                       |
| <i>Social Security Number Data Type</i> | This data type is applied to numeric data following the pattern for United States Social Security numbers.                                                                                                                                                                                  |

|                                |                                                                                                                                                                                                                  |
|--------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Phone Number Data Type</i>  | This data type is applied to numeric data following common patterns that express telephone numbers and known valid phone numbers in the United States.                                                           |
| <i>Email Address Data Type</i> | This data type matches text values that are properly formatted email addresses.                                                                                                                                  |
| <i>Credit Card Data Type</i>   | Credit card numbers are numeric data that follow the 14-digit or 16-digit patterns for credit cards.                                                                                                             |
| <i>Gender Data Type</i>        | This data type matches a variety of text patterns for expressing male/female distinctions.                                                                                                                       |
| <i>Zip Code Data Type</i>      | This data type matches five- and nine-digit U.S. zipcode patterns.                                                                                                                                               |
| <i>State Data Type</i>         | State data type is applied to data that uses the full names or the two-letter abbreviations for states in the United States.                                                                                     |
| <i>Object Data Type</i>        | An <b>Object</b> data type is a method for encoding key-value pairs. A single field value may contain one or more sets of key-value pairs.                                                                       |
| <i>Array Data Type</i>         | An <b>array</b> is a list of values grouped into a single value. An array may be of variable length; in one record the array field may contain two elements, while in the next record, it contains six elements. |
| <i>IP Address Data Type</i>    | The IP Address data type supports IPv4 address.                                                                                                                                                                  |
| <i>URL Data Type</i>           | URL data type is applied to data that follows generalized patterns of URLs.                                                                                                                                      |
| <i>HTTP Code Data Type</i>     | Values of these data types are three-digit numeric values, which correspond to recognized HTTP Status Codes.                                                                                                     |
| <i>Datetime Data Type</i>      | Designer Cloud powered by Trifacta® Enterprise Edition supports a variety of Datetime formats, each of which has additional variations to it.                                                                    |

For more information on the data types that are supported within the Designer Cloud application , see *Supported Data Types*.



# Avro Data Type Conversions

**NOTE:** The Trifacta® data types listed in this page reflect the raw data type of the converted column. Depending on the contents of the column, the Transformer Page may re-infer a different data type, when a dataset using this type of source is loaded.

## Import

| Avro Data Type | Trifacta Data Type | Notes |
|----------------|--------------------|-------|
| string         | String             |       |
| int            | Integer            |       |
| long           | Integer            |       |
| float          | Decimal            |       |
| double         | Decimal            |       |
| boolean        | Bool               |       |
| map            | Object             |       |
| array          | Array              |       |
| record         | Object             |       |
| enum           | String             |       |
| fixed          | String             |       |

## Export

On export, Trifacta data types are exported to their corresponding Avro types, with the following specific mappings:

| Trifacta Data Type | Avro Data Type | Notes |
|--------------------|----------------|-------|
| Boolean            | BOOLEAN        |       |
| Integer            | LONG           |       |
| Decimal            | DOUBLE         |       |
| String             | STRING         |       |

The fallback data type on export is STRING.

# DB2 Data Type Conversions

**NOTE:** The Trifacta® data types listed in this page reflect the raw data type of the converted column. Depending on the contents of the column, the Transformer Page may re-infer a different data type, when a dataset using this type of source is loaded.

## Access/Read

| Source Data Type | Supported | Trifacta Data Type |
|------------------|-----------|--------------------|
| BOOLEAN          | Y         | Bool               |
| VARCHAR          | Y         | String             |
| INTEGER          | Y         | Integer            |

## Publish/Write

| Trifacta Data Type | DB2 Data Type | Notes |
|--------------------|---------------|-------|
| Bool               | BOOLEAN       |       |
|                    | VARCHAR       |       |
| Integer            | BIGINT        |       |
|                    | INT           |       |
|                    | SMALLINT      |       |
|                    | VARCHAR       |       |
| String             | VARCHAR(1024) |       |
| Datetime           | TIMESTAMP     |       |
|                    | VARCHAR       |       |
| Time               | VARCHAR(1024) |       |
| Decimal            | DECIMAL       |       |
|                    | NUMERIC       |       |
|                    | VARCHAR       |       |
| Map                | VARCHAR(1024) |       |
| Array              | VARCHAR(1024) |       |

# Hive Data Type Conversions

## Contents:

- *Access/Read*
- *Write/Publish*
  - *Create new table*
  - *Append to existing table*
  - *Notes on Datetime columns*

**NOTE:** The Trifacta® data types listed in this page reflect the raw data type of the converted column. Depending on the contents of the column, the Transformer Page may re-infer a different data type, when a dataset using this type of source is loaded.

## Access/Read

When a Hive data type is imported, its JDBC data type is remapped according to the following table.

**Tip:** Data precision may be lost during conversion. You may want to generate min and max values and compute significant digits for values in your Hive tables and then compute the same in the Designer Cloud application .

| Source Data Type | Supported? | Trifacta Data Type | Notes                                                                                                                                                                                                                             |
|------------------|------------|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| array            | Y          | Array              |                                                                                                                                                                                                                                   |
| bigint           | Y          | Integer            | <b>NOTE:</b> The Designer Cloud powered by Trifacta platform may infer bigint columns containing very large or very small values as String data type.                                                                             |
| binary           | Y          | String             |                                                                                                                                                                                                                                   |
| boolean          | Y          | Bool               |                                                                                                                                                                                                                                   |
| char             | Y          | String             |                                                                                                                                                                                                                                   |
| date             | Y          | Datetime           |                                                                                                                                                                                                                                   |
| decimal          | Y          | Decimal            |                                                                                                                                                                                                                                   |
| double           | Y          | Decimal            |                                                                                                                                                                                                                                   |
| float            | Y          | Decimal            | <b>NOTE:</b> On import, some float columns may be interpreted as Integer data type in the Designer Cloud powered by Trifacta platform . To fix, you can explicitly set the column's data type to Decimal in the Transformer page. |
| int              | Y          | Integer            |                                                                                                                                                                                                                                   |
| map              | Y          | Object             |                                                                                                                                                                                                                                   |
| smallint         | Y          | Integer            |                                                                                                                                                                                                                                   |
| string           | Y          | String             |                                                                                                                                                                                                                                   |
| struct           | Y          | Object             |                                                                                                                                                                                                                                   |

|           |   |          |  |
|-----------|---|----------|--|
| timestamp | Y | Datetime |  |
| tinyint   | Y | Integer  |  |
| uniontype | N |          |  |
| varchar   | Y | String   |  |

## Write/Publish

### Create new table

**NOTE:** By default, the maximum length of values published to VARCHAR columns is 256 characters. As needed, this limit can be changed for multiple publication targets. For more information, see *Configure Application Limits*.

| Trifacta Data Type | Hive Data Type                                          | Notes                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|--------------------|---------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| String             | string                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Integer            | bigint                                                  | <p><b>NOTE:</b> The Designer Cloud powered by Trifacta platform may infer Integer columns containing very large or very small values as String data type. Before you publish, you should verify that your columns containing extreme values are interpreted as Integer type. You can import a target schema to assist in lining up your columns with the expected target. For more information, see <i>Overview of RapidTarget</i>.</p> |
| Decimal            | double                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Bool               | boolean                                                 |                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Datetime           | Timestamp /string (see Notes on Datetime columns below) | Target data type is based on the underlying data. Time zone information is retained.                                                                                                                                                                                                                                                                                                                                                    |
| Object             | string                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Array              | string                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                         |

### Append to existing table

If you are publishing to a pre-existing table, the following data type conversions apply:

- **Columns:** Trifacta data types
- **Rows:** Target table data types

In any table cell, a Y indicates that the append operation for that data type mapping is supported.

**NOTE:** You cannot append to Hive map and array column types from Trifacta columns of Map and Array type, even if you imported data from this source.

|      | String | Integer | Datetime | Bool | Decimal | Map | Array | Out of Range error |
|------|--------|---------|----------|------|---------|-----|-------|--------------------|
| CHAR | Y      | Y       | Y        | Y    | Y       | Y   | Y     |                    |

|           |   |   |   |   |   |   |   |      |
|-----------|---|---|---|---|---|---|---|------|
| VARCHAR   | Y | Y | Y | Y | Y | Y | Y |      |
| STRING    | Y | Y | Y | Y | Y | Y | Y |      |
| INT       |   | Y |   |   |   |   |   | NULL |
| BIGINT    |   | Y |   |   |   |   |   | n/a  |
| TINYINT   |   |   |   |   |   |   |   | NULL |
| SMALLINT  |   |   |   |   |   |   |   | NULL |
| DECIMAL   |   | Y |   |   | Y |   |   | NULL |
| DOUBLE    |   | Y |   |   | Y |   |   | n/a  |
| FLOAT     |   |   |   |   | Y |   |   | NULL |
| TIMESTAMP |   |   | Y |   |   |   |   |      |
| BOOLEAN   |   |   |   | Y |   |   |   |      |

## Notes on Datetime columns

### Run Job

Columns in new tables created for output of Datetime columns are written with the Hive timestamp data type. These columns can be appended.

- Before release 4.2.1, Datetime columns were written to Hive as type String. Jobs that were created in these releases and that write to pre-existing tables continue to behave this way.
- A single job cannot write Datetime values to one table as String type and to another table as Timestamp type. This type of job should be split into multiple types. The table schemas may require modification.
  - The above issue may appear as the following error when executing the job:

```
Unable to publish due to datetime data type conflict in column XXXX
```

### Ad-Hoc Publishing

- When you export pre-generated results to Hive, all new tables created for Datetime column values continue to store String data type in Hive for Release 4.2.1. These columns can be appended with new String data.
- When you publish results from a job through the Publishing dialog to Hive, all Datetime column values are written as String type.
- If you are appending to a Timestamp column, the exported Datetime column must be in the following format: yyyy-MM-dd HH:mm:ss.xxxx

# Oracle Data Type Conversions

**NOTE:** The Trifacta® data types listed in this page reflect the raw data type of the converted column. Depending on the contents of the column, the Transformer Page may re-infer a different data type, when a dataset using this type of source is loaded.

**NOTE:** Dots (.) in the names of Oracle tables or table columns are not supported.

## Access/Read

| Source Data Type       | Supported | Trifacta Data Type |
|------------------------|-----------|--------------------|
| ANYDATA                | N         |                    |
| ANYDATASET             | N         |                    |
| ANYTYPE                | N         |                    |
| BFILE                  | N         |                    |
| BINARY_DOUBLE          | Y         | Decimal            |
| BINARY_FLOAT           | Y         | Decimal            |
| BLOB                   | N         |                    |
| CHAR                   | Y         | String             |
| CLOB                   | Y         | String             |
| DATE                   | Y         | Date/Time          |
| DBURIType              | Y         | String             |
| FLOAT                  | Y         | Decimal            |
| HTTPURIType            | Y         | String             |
| INTERVAL_DAY TO SECOND | Y         | String             |
| INTERVAL_YEAR TO MONTH | Y         | String             |
| LONG RAW               | Y         | String (base64)    |
| <media types>          | N         |                    |
| NCHAR                  | Y         | String             |
| NCLOB                  | Y         | String             |
| NUMBER                 | Y         | Decimal / Integer  |
| NVARCHAR2              | Y         | Integer            |
| RAW                    | Y         | String (base64)    |
| ROWID                  | N         |                    |
| SDO_GEOMETRY           | N         |                    |
| SDO_GEORASTER          | N         |                    |
| SDO_TOPO_GEOMETRY      | N         |                    |

|                               |   |        |
|-------------------------------|---|--------|
| TIMESTAMP                     | Y | String |
| TIMESTAMP WITH LOCAL TIMEZONE | Y | String |
| TIMESTAMP WITH TIMEZONE       | Y | String |
| URITYPE                       | Y | String |
| UROWID                        | N |        |
| VARCHAR2                      | Y | String |
| XDBURType                     | Y | String |
| XMLType                       | N |        |

**NOTE:** Implementation of Oracle custom types is a custom engagement. For more information, please contact *Alteryx Customer Success and Services*.

## Write/Publish

| Trifacta Data Type | Oracle Column Type | Notes                                                                                                                                                                            |
|--------------------|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Bool               | VARCHAR (256)      | Oracle VARCHAR column type must include the maximum number of permitted characters.                                                                                              |
| Integer            | INT                |                                                                                                                                                                                  |
|                    | INTEGER            |                                                                                                                                                                                  |
|                    | SMALLINT           |                                                                                                                                                                                  |
|                    | DECIMAL            |                                                                                                                                                                                  |
|                    | DEC                |                                                                                                                                                                                  |
|                    | NUMERIC            |                                                                                                                                                                                  |
|                    | NUMBER             |                                                                                                                                                                                  |
|                    | FLOAT              |                                                                                                                                                                                  |
|                    | VARCHAR (256)      | Oracle VARCHAR column type must include the maximum number of permitted characters.                                                                                              |
| String             | VARCHAR (256)      | Oracle VARCHAR column type must include the maximum number of permitted characters.                                                                                              |
| Datetime           | TIMESTAMP          |                                                                                                                                                                                  |
|                    | VARCHAR (256)      | Oracle VARCHAR column type must include the maximum number of permitted characters.                                                                                              |
| Timestamp          | VARCHAR (256)      | Oracle VARCHAR column type must include the maximum number of permitted characters.                                                                                              |
| Float              | DECIMAL (38,9)     | Oracle DECIMAL column type must include the scale (total number of permitted digits) and precision (total number of digits permitted to the right of the decimal) as parameters. |
|                    | DEC                |                                                                                                                                                                                  |
|                    | NUMERIC            |                                                                                                                                                                                  |
|                    | NUMBER             |                                                                                                                                                                                  |
|                    | FLOAT              |                                                                                                                                                                                  |

|       |                  |                                                                                     |
|-------|------------------|-------------------------------------------------------------------------------------|
|       | VARCHAR<br>(256) | Oracle VARCHAR column type must include the maximum number of permitted characters. |
| Map   | VARCHAR<br>(256) | Oracle VARCHAR column type must include the maximum number of permitted characters. |
| Array | VARCHAR<br>(256) | Oracle VARCHAR column type must include the maximum number of permitted characters. |

**NOTE:** If you are appending to an existing table where a column's Trifacta data type is not mapped to the column data type in the target table, a validation error is thrown, as the platform writes unmapped types as String data type.



# MySQL Data Type Conversions

**NOTE:** The Trifacta® data types listed in this page reflect the raw data type of the converted column. Depending on the contents of the column, the Transformer Page may re-infer a different data type, when a dataset using this type of source is loaded.

## Access/Read

| Source Data Type                 | Supported? | Trifacta Data Type |
|----------------------------------|------------|--------------------|
| array                            | N          |                    |
| bigint                           | Y          | Integer            |
| tinyint                          | Y          | Integer            |
| mediumint                        | Y          | Integer            |
| smallint                         | Y          | Integer            |
| int                              | Y          | Integer            |
| bit [ (n) ]                      | Y          | String             |
| float                            | Y          | Float              |
| numeric                          | Y          | Float              |
| decimal                          | Y          | Decimal            |
| real                             | Y          | Float              |
| boolean                          | Y          | Bool               |
| character varying(n), varchar(n) | Y          | String             |
| character(n), char(n)            | Y          | String             |
| date                             | Y          | Date               |
| double                           | Y          | Decimal            |
| enum                             | N          |                    |
| set                              | N          |                    |
| json                             | Y          | String             |
| text                             | Y          | String             |
| tinytext                         | Y          | String             |
| mediumtext                       | Y          | String             |
| longtext                         | Y          | String             |
| time                             | Y          | Datetime           |
| datetime                         | Y          | Datetime           |
| timestamp                        | Y          | Datetime           |
| year                             | Y          | String             |

## Write/Publish

| Trifacta Data Type | PostgreSQL Column Type | Notes |
|--------------------|------------------------|-------|
| Bool               | BIT                    |       |
|                    | VARCHAR                |       |
|                    | TINYINT(1)             |       |
| Integer            | BIGINT                 |       |
|                    | INT                    |       |
|                    | SMALLINT               |       |
|                    | MEDIUMINT              |       |
|                    | TINYINT                |       |
|                    | VARCHAR                |       |
|                    | VARCHAR                |       |
|                    | VARCHAR                |       |
| String             | VARCHAR                |       |
|                    | TINYTEXT               |       |
|                    | TEXT                   |       |
|                    | MEDIUMTEXT             |       |
|                    | LONGTEXT               |       |
| Datetime           | TIMESTAMP              |       |
|                    | DATETIME               |       |
|                    | DATE                   |       |
|                    | VARCHAR                |       |
| Timestamp          | VARCHAR                |       |
| Float              | FLOAT                  |       |
|                    | DECIMAL                |       |
|                    | REAL                   |       |
|                    | NUMERIC                |       |
|                    | VARCHAR                |       |
| Map                | VARCHAR                |       |
| Array              | VARCHAR                |       |

**NOTE:** If you are appending to an existing table where a column's Trifacta data type is not mapped to the column data type in the target table, a validation error is thrown, as the platform writes unmapped types as String data type.

# Parquet Data Type Conversions

**NOTE:** The Trifacta® data types listed in this page reflect the raw data type of the converted column. Depending on the contents of the column, the Transformer Page may re-infer a different data type, when a dataset using this type of source is loaded.

## Import

**NOTE:** Designer Cloud powered by Trifacta Enterprise Edition does not support ingest of Parquet files with nested values, which can occur for Map or Object data types.

| Parquet Data Type | Trifacta Data Type | Notes |
|-------------------|--------------------|-------|
| STRING            | String             |       |
| INT               | Integer            |       |
| DECIMAL           | Decimal            |       |
| DATE              | Datetime           |       |
| TIME              | Datetime           |       |
| TIMESTAMP         | Datetime           |       |
| LIST              | Array              |       |
| MAP               | Object             |       |

### Limitations on import:

The Parquet data format supports the use of row groups for organizing chunks of data. This row grouping is helpful for processing across distributed systems.

Designer Cloud powered by Trifacta Enterprise Edition places limitations on the volume of data that can be displayed in the browser. By default, these limits are set to 10 MB.

If Parquet row groups are greater than 10 MB:

- You cannot preview data from the file before import.
- When a Parquet-based dataset is loaded in the Transformer page, the screen may be blank.

**Tip:** You can create a new sample from inside the Transformer page. The sample is displayed normally.

Other product functions work as expected with Parquet format.

## Export

On export, Trifacta data types are exported to their corresponding Parquet types, with the following specific mappings:

| Trifacta Data Type | Parquet Data Type | Notes |
|--------------------|-------------------|-------|
| Boolean            | BOOLEAN           |       |
|                    |                   |       |

|         |                     |  |
|---------|---------------------|--|
| Integer | INT64               |  |
| Decimal | DOUBLE              |  |
| String  | BYTE_ARRAY (STRING) |  |

The fallback data type on export is BYTE\_ARRAY (STRING).

**NOTE:** Optionally, you can enable publication of Datetime values to Datetime/Timestamp values for Parquet outputs. For more information, see *Miscellaneous Configuration*.

# Postgres Data Type Conversions

**NOTE:** The Trifacta® data types listed in this page reflect the raw data type of the converted column. Depending on the contents of the column, the Transformer Page may re-infer a different data type, when a dataset using this type of source is loaded.

## Access/Read

| Source Data Type                 | Supported? | Trifacta Data Type |
|----------------------------------|------------|--------------------|
| array                            | N          |                    |
| bigint                           | Y          | Integer            |
| bigserial                        | Y          | Integer            |
| bit [ (n) ]                      | Y          | String             |
| bit varying [ (n) ]              | Y          | String             |
| boolean                          | Y          | Bool               |
| box                              | Y          | String             |
| bytea                            | Y          | String             |
| character varying(n), varchar(n) | Y          | String             |
| character(n), char(n)            | Y          | String             |
| cidr                             | Y          | String             |
| circle                           | Y          | String             |
| composite                        | N          |                    |
| date                             | Y          | Date               |
| daterange                        | N          |                    |
| decimal                          | Y          | Decimal            |
| double precision                 | Y          | Decimal            |
| enum                             | N          |                    |
| inet                             | Y          | String             |
| int4range                        | N          |                    |
| int8range                        | N          |                    |
| integer                          | Y          | Integer            |
| interval [ fields ] [ (p) ]      | Y          | String             |
| json                             | Y          | Object             |
| line                             | N          |                    |
| lseg                             | Y          | String             |
| macaddr                          | Y          | String             |
| money                            | Y          | Decimal            |
| numeric                          | Y          | Decimal/Integer    |
|                                  |            |                    |

|                                  |   |         |
|----------------------------------|---|---------|
| numrange                         | N |         |
| oid                              | N |         |
| path                             | Y | String  |
| point                            | Y | String  |
| polygon                          | Y | String  |
| real                             | Y | Decimal |
| serial                           | Y | Integer |
| smallint                         | Y | Integer |
| smallserial                      | Y | Integer |
| text                             | Y | String  |
| time [ (p) ]                     | Y | Date    |
| time [ (p) ] with time zone      | Y | String  |
| timestamp [ (p) ]                | Y | Date    |
| timestamp [ (p) ] with time zone | Y | Date    |
| tsquery                          | Y | String  |
| tsrange                          | N |         |
| tstzrange                        | N |         |
| tsvector                         | Y | String  |
| txid_snapshot                    | Y | String  |
| uuid                             | Y | String  |
| xml                              | Y | String  |

## Write/Publish

| Trifacta Data Type | PostgreSQL Column Type | Notes |
|--------------------|------------------------|-------|
| Bool               | BOOL                   |       |
|                    | VARCHAR                |       |
| Integer            | BIGINT                 |       |
|                    | INT                    |       |
|                    | SMALLINT               |       |
|                    | VARCHAR                |       |
| String             | VARCHAR                |       |
| Datetime           | TIMESTAMP              |       |
|                    | VARCHAR                |       |
| Float              | DECIMAL                |       |
|                    | NUMERIC                |       |
|                    | VARCHAR                |       |
| Map                | VARCHAR                |       |
| Array              | VARCHAR                |       |

**NOTE:** If you are appending to an existing table where a column's Trifacta data type is not mapped to the column data type in the target table, a validation error is thrown, as the platform writes unmapped types as String data type.

# Redshift Data Type Conversions

## Contents:

- *Access/Read*
- *Write/Publish*
  - *Create new table*
  - *Append to existing table*

**NOTE:** The Trifacta® data types listed in this page reflect the raw data type of the converted column. Depending on the contents of the column, the Transformer Page may re-infer a different data type, when a dataset using this type of source is loaded.

## Access/Read

| Source Data Type | Supported | Trifacta data type |
|------------------|-----------|--------------------|
| string           | Y         | String             |
| bigint           | Y         | Integer            |
| double precision | Y         | Decimal            |
| bool             | Y         | Boolean            |
| date             | Y         | DateTime           |
| timestamp        | Y         | DateTime           |

## Write/Publish

### Create new table

**NOTE:** By default, the maximum length of values published to VARCHAR columns is 256 characters. As needed, this limit can be changed for multiple publication targets. For more information, see *Configure Application Limits*.

| Trifacta Data Type | Redshift Data Type | Notes                                                                                                                                     |
|--------------------|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| String             | varchar            |                                                                                                                                           |
| Integer            | bigint             |                                                                                                                                           |
| Decimal            | double precision   |                                                                                                                                           |
| Bool               | bool               |                                                                                                                                           |
| Datetime           | timestamp          | When you publish results from a job through the Export Results window to Redshift, all Datetime column values are written as String type. |
| Object             | varchar            |                                                                                                                                           |
| Array              | varchar            |                                                                                                                                           |

**NOTE:** Object and Array types are written in the supported format, if escaped commas and backslashes are unescaped in your recipe.



## Append to existing table

If you are publishing to a pre-existing table, the following data type conversions apply:

- **Columns:** Trifacta data types
- **Rows:** Target table data types

In any table cell, a Y indicates that the append operation for that data type mapping is supported.

|                   | String | Integer | Datetime | Bool | Float | Map | Array |
|-------------------|--------|---------|----------|------|-------|-----|-------|
| TEXT              | Y      | Y       | Y        | Y    | Y     | Y   | Y     |
| VARCHAR           | Y      | Y       | Y        | Y    | Y     | Y   | Y     |
| NVARCHAR          | Y      | Y       | Y        | Y    | Y     | Y   | Y     |
| BPCHAR            | Y      | Y       | Y        | Y    | Y     | Y   | Y     |
| NCHAR             |        |         |          |      |       |     |       |
| CHAR              |        |         |          |      |       |     |       |
| CHARACTER VARYING | Y      | Y       | Y        | Y    | Y     | Y   | Y     |
| SMALLINT          |        |         |          |      |       |     |       |
| INT2              |        |         |          |      |       |     |       |
| INTEGER           |        |         |          |      |       |     |       |
| INT               |        |         |          |      |       |     |       |
| INT4              |        |         |          |      |       |     |       |
| BIGINT            |        | Y       |          |      |       |     |       |
| INT8              |        | Y       |          |      |       |     |       |
| DECIMAL           |        | Y       |          |      | Y     |     |       |
| NUMERIC           |        | Y       |          |      | Y     |     |       |
| DOUBLE_PRECISION  |        | Y       |          |      | Y     |     |       |
| FLOAT             |        | Y       |          |      | Y     |     |       |
| FLOAT4            |        |         |          |      |       |     |       |
| FLOAT8            |        |         |          |      | Y     |     |       |
| REAL              |        |         |          |      |       |     |       |
| BOOL              |        |         |          | Y    |       |     |       |
| BOOLEAN           |        |         |          | Y    |       |     |       |
| TIMESTAMP         |        |         | Y        |      |       |     |       |
| TIMESTAMPTZ       |        |         | Y        |      |       |     |       |

# Snowflake Data Type Conversions

**NOTE:** The Trifacta® data types listed in this page reflect the raw data type of the converted column. Depending on the contents of the column, the Transformer Page may re-infer a different data type, when a dataset using this type of source is loaded.

## Access/Read

| Source Type      | Supported       | Trifacta Data Type |
|------------------|-----------------|--------------------|
| NUMBER           | Y               | Integer            |
| DECIMAL          | Y               | Integer            |
| NUMERIC          | Y               | Integer            |
| INT              | Y               | Integer            |
| INTEGER          | Y               | Integer            |
| BIGINT           | Y               | Integer            |
| SMALLINT         | Y               | Integer            |
| FLOAT            | Y               | Decimal            |
| FLOAT4           | Y               | Decimal            |
| FLOAT8           | Y               | Decimal            |
| DOUBLE           | Y               | Decimal            |
| DOUBLE_PRECISION | Y               | Decimal            |
| REAL             | Y               | Decimal            |
| VARCHAR          | Y               | String             |
| CHAR             | Y               | String             |
| CHARACTER        | Y               | String             |
| STRING           | Y               | String             |
| TEXT             | Y               | String             |
| BINARY           | Y               | String             |
| VARBINARY        | Y               | String             |
| BOOLEAN          | Y               | Bool               |
| DATE             | Y               | String             |
| DATETIME         | Y               | String             |
| TIME             | Y               | String             |
| TIMESTAMP        | Y               | String             |
| TIMESTAMP_TZ     | Y               | Datetime           |
| TIMESTAMP_LTZ    | Y <sup>1)</sup> | Datetime           |
| TIMESTAMP_NTZ    | Y <sup>2)</sup> | Datetime           |
| VARIANT          | N               |                    |
| OBJECT           | N               |                    |
|                  |                 |                    |

|       |   |  |
|-------|---|--|
| ARRAY | N |  |
|-------|---|--|

#### Notes:

1. Convert to Datetime using local timezone of the Snowflake connector
2. Convert to Datetime using UTC

#### Write/Publish

| Trifacta Data Type | Supported | Snowflake Type | Notes      |
|--------------------|-----------|----------------|------------|
| Array              | N         |                |            |
| Bool               | Y         | BOOLEAN        |            |
| Date               | Y         | TIMESTAMP      | See below. |
| Datetime           | Y         | TIMESTAMP      |            |
| Time               | Y         | TIME           |            |
| Float              | Y         | FLOAT          |            |
| Integer            | Y         | BIGINT         |            |
| Map                | N         |                |            |
| String             | Y         | VARCHAR        |            |

#### Notes on Date publishing:

**NOTE:** Trifacta Date columns can be published to existing Snowflake columns of Date, Datetime, and String type.

When Dates are published to Snowflake, the following date formats are supported. In some cases, missing data is inserted into the output value.

| Trifacta date format | Snowflake date format |
|----------------------|-----------------------|
| yy-dd-mm             | yy-dd-mm              |
| yy-mm-dd             | yy-mm-dd              |
| dd-mm-yy             | dd-mm-yy              |
| mm-dd-yy             | mm-dd-yy              |
| mm-yy                | mm-yy                 |
| dd-mm                | 1970-dd-mm            |
| mm-dd                | 1970-mm-dd            |
| mm-yy                | mm-yy-01              |

- On publication, all dates are written in the following format: `yyyy-mm-dd`.
- No other date formats are supported for writing to Snowflake as date values.

# AWS Glue Data Type Conversions

**NOTE:** The Trifacta® data types listed in this page reflect the raw data type of the converted column. Depending on the contents of the column, the Transformer Page may re-infer a different data type, when a dataset using this type of source is loaded.

## Access/Read

| Source Data Type | Supported | Trifacta Data Type | Notes                                                                                                                                                                                                                                                                                                      |
|------------------|-----------|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| BIGINT           | Y         | Integer            | Tables generated by the Glue crawler from double-quoted CSV files with columns that are inferred by Glue to be this data type result in empty columns when imported to the Designer Cloud powered by Trifacta platform . The workaround is to change the column types for these columns in Glue to STRING. |
| INT              | Y         | Integer            |                                                                                                                                                                                                                                                                                                            |
| STRING           | Y         | String             |                                                                                                                                                                                                                                                                                                            |
| STRUCT           | Y         | Object             |                                                                                                                                                                                                                                                                                                            |
| DOUBLE           | Y         | Decimal            | Tables generated by the Glue crawler from double-quoted CSV files with columns that are inferred by Glue to be this data type result in empty columns when imported to the Designer Cloud powered by Trifacta platform . The workaround is to change the column types for these columns in Glue to STRING. |
| TIMESTAMP        | Y         | Datetime           |                                                                                                                                                                                                                                                                                                            |
| BOOLEAN          | Y         | Bool               | Tables generated by the Glue crawler from double-quoted CSV files with columns that are inferred by Glue to be this data type result in empty columns when imported to the Designer Cloud powered by Trifacta platform . The workaround is to change the column types for these columns in Glue to STRING. |

## Write/Publish

Not supported.

# Salesforce Data Type Conversions

**NOTE:** The Trifacta® data types listed in this page reflect the raw data type of the converted column. Depending on the contents of the column, the Transformer Page may re-infer a different data type, when a dataset using this type of source is loaded.

## Access/Read

| Source Data Type | Supported? | Trifacta Data Type |
|------------------|------------|--------------------|
| BOOL             | Y          | Bool               |
| CHAR             | Y          | String             |
| NUMBER           | Y          | Integer            |
| BINARY_FLOAT     | Y          | Float              |
| BINARY_DOUBLE    | Y          | Float              |

## Publish/Write

Publishing to this datastore is not supported in this release.

# SQL Server Data Type Conversions

**NOTE:** The Trifacta® data types listed in this page reflect the raw data type of the converted column. Depending on the contents of the column, the Transformer Page may re-infer a different data type, when a dataset using this type of source is loaded.

## Access/Read

| Source Data Type  | Supported? | Trifacta Data Type |
|-------------------|------------|--------------------|
| Bigint            | Y          | Integer            |
| BIGINT IDENTITY   | Y          | Integer            |
| Binary            | Y          | String (Base64)    |
| Bit               | Y          | Bool               |
| Char              | Y          | String             |
| Cursor            | N          |                    |
| Date              | Y          | Date               |
| Datetime          | Y          | Date               |
| Datetime2         | Y          | Date               |
| Datetimeoffset    | Y          | String             |
| Decimal           | Y          | Integer/Decimal    |
| DECIMAL IDENTITY  | Y          | Integer            |
| Float             | Y          | Decimal            |
| GEOGRAPHY         | Y          | String             |
| GEOMETRY          | Y          | String             |
| HIERARCHYID       | Y          | String             |
| Image             | Y          | String             |
| Int               | Y          | Integer            |
| INT IDENTITY      | Y          | Integer            |
| Money             | Y          | Decimal            |
| Nchar             | Y          | String             |
| Ntext             | Y          | String             |
| Numeric           | Y          | Integer/Decimal    |
| NUMERIC IDENTITY  | Y          | Integer            |
| Nvarchar          | Y          | String             |
| Nvarchar(max)     | Y          | String             |
| Real              | Y          | Decimal            |
| Smalldatetime     | Y          | Date               |
| Smallint          | Y          | Integer            |
| SMALLINT IDENTITY | Y          | Integer            |
|                   |            |                    |

|                  |   |                 |
|------------------|---|-----------------|
| Smallmoney       | Y | Decimal         |
| Sql_variant      | N |                 |
| Table            | N |                 |
| Text             | Y | String          |
| Time             | Y | String          |
| Timestamp        | Y | String          |
| Tinyint          | Y | Integer         |
| TINYINT IDENTITY | Y | Integer         |
| Uniqueidentifier | Y | String          |
| Varbinary        | Y | String (Base64) |
| Varbinary(max)   | Y | String (Base64) |
| Varchar          | Y | String          |
| Varchar(max)     | Y | String          |
| Xml              | Y | String          |

## Write/Publish

| Trifacta Data Type | SQL Server Column Type | Notes |
|--------------------|------------------------|-------|
| Bool               | BIT                    |       |
|                    | VARCHAR(256)           |       |
| Integer            | BIGINT                 |       |
|                    | INT                    |       |
|                    | SMALLINT               |       |
|                    | DECIMAL                |       |
|                    | VARCHAR(256)           |       |
| String             | VARCHAR(256)           |       |
| Datetime           | DATETIME               |       |
|                    | TIMESTAMP              |       |
|                    | VARCHAR(256)           |       |
| Timestamp          | TIME                   |       |
|                    | VARCHAR(256)           |       |
| Float              | FLOAT                  |       |
|                    | DECIMAL                |       |
|                    | NUMERIC                |       |
|                    | VARCHAR(256)           |       |
| Map                | VARCHAR(256)           |       |
| Array              | VARCHAR(256)           |       |

**NOTE:** If you are appending to an existing table where a column's Trifacta data type is not mapped to the column data type in the target table, a validation error is thrown, as the platform writes unmapped types as String data type.



# SQL DW Data Type Conversions

Contents:

- Access/Read
- Write/Publish
  - Create new table
  - Append to existing table

**NOTE:** The Trifacta® data types listed in this page reflect the raw data type of the converted column. Depending on the contents of the column, the Transformer Page may re-infer a different data type, when a dataset using this type of source is loaded.

## Access/Read

| Source Data Type | Supported | Trifacta Data Type | Notes                                                                                                                                                                                                                                                                                   |
|------------------|-----------|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| INT              | Y         | Integer            |                                                                                                                                                                                                                                                                                         |
| TINYINT          | Y         | Integer            |                                                                                                                                                                                                                                                                                         |
| SMALLINT         | Y         | Integer            |                                                                                                                                                                                                                                                                                         |
| BIGINT           | Y         | Integer            | <b>NOTE:</b> The Designer Cloud powered by Trifacta platform may infer bigint columns containing very large or very small values as String data type. If needed, you can disable type inference for individual schematized sources. For more information, see <i>Import Data Page</i> . |
| FLOAT            | Y         | Float              |                                                                                                                                                                                                                                                                                         |
| REAL             | Y         | Float              |                                                                                                                                                                                                                                                                                         |
| BIT              | Y         | Bool               |                                                                                                                                                                                                                                                                                         |
| SMALLMONEY       | Y         | String             |                                                                                                                                                                                                                                                                                         |
| MONEY            | Y         | String             |                                                                                                                                                                                                                                                                                         |
| DECIMAL          | Y         | Float              |                                                                                                                                                                                                                                                                                         |
| NUMERIC          | Y         | String             |                                                                                                                                                                                                                                                                                         |
| DATETIMEOFFSET   | Y         | String             |                                                                                                                                                                                                                                                                                         |
| TIME             | Y         | String             |                                                                                                                                                                                                                                                                                         |
| DATE             | Y         | String             |                                                                                                                                                                                                                                                                                         |
| DATETIME         | Y         | String             |                                                                                                                                                                                                                                                                                         |
| DATETIME2        | Y         | String             |                                                                                                                                                                                                                                                                                         |
| SMALLDATETIME    | Y         | String             |                                                                                                                                                                                                                                                                                         |
| CHAR             | Y         | String             |                                                                                                                                                                                                                                                                                         |
| VARCHAR          | Y         | String             |                                                                                                                                                                                                                                                                                         |

|                      |   |        |  |
|----------------------|---|--------|--|
| NCHAR                | Y | String |  |
| NVARCH<br>AR         | Y | String |  |
| SYSNAME              | Y | String |  |
| BINARY               | Y | String |  |
| VARBINA<br>RY        | Y | String |  |
| UNIQUEI<br>DENTIFIER | Y | String |  |
| TIMESTA<br>MP        | N |        |  |
| GEOGRA<br>PHY        | N |        |  |
| GEOMET<br>RY         | N |        |  |
| HIERARC<br>HYID      | N |        |  |
| IMAGE                | N |        |  |
| TEXT                 | N |        |  |
| NTEXT                | N |        |  |
| XML                  | N |        |  |
| CURSOR               | N |        |  |
| ROWVER<br>SION       | N |        |  |
| SQL_VAR<br>IANT      | N |        |  |

## Write/Publish

### Create new table

**NOTE:** By default, the maximum length of values published to VARCHAR columns is 256 characters. As needed, this limit can be changed for multiple publication targets. For more information, see *Configure Application Limits*.

| Trifacta<br>Data<br>Type | SQL<br>DW<br>Data<br>Type | Notes                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|--------------------------|---------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| String                   | VARC<br>HAR               |                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Integer                  | BIGINT                    | <p><b>NOTE:</b> The Designer Cloud powered by Trifacta platform may infer Integer columns containing very large or very small values as String data type. Before you publish, you should verify that your columns containing extreme values are interpreted as Integer type. You can import a target schema to assist in lining up your columns with the expected target. For more information, see <i>Overview of RapidTarget</i>.</p> |
| Float                    | FLOAT                     |                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Bool                     | BIT                       |                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Datetime                 | DATET                     | If a time-only value is published as an append to a pre-defined DATETIME2 column, then the output column is                                                                                                                                                                                                                                                                                                                             |

|                                    |             |                                                                                                                                                                                                                                                                                                                                |
|------------------------------------|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                    | IME2        | prepended with 1900-01-01.                                                                                                                                                                                                                                                                                                     |
| Datetime<br>(with time value only) | TIME        | <p>If a value is published as an append to a pre-defined TIME column, then any date information is dropped from the output.</p> <div> <b>NOTE:</b> The platform publishes to Timestamp columns only for append operations to pre-existing tables. </div>                                                                       |
| Datetime                           | VARC<br>HAR | <p>If a Datetime value is published as an append to a pre-defined VARCHAR column, then the output column contains the string value of whatever appears in the Transformer page.</p> <p>When you publish results from a job through the Publishing dialog to SQL DW, all Datetime column values are written as String type.</p> |
| Map                                | VARC<br>HAR |                                                                                                                                                                                                                                                                                                                                |
| Array                              | VARC<br>HAR |                                                                                                                                                                                                                                                                                                                                |
| Date                               | VARC<br>HAR |                                                                                                                                                                                                                                                                                                                                |
| All Other Data Types               | VARC<br>HAR |                                                                                                                                                                                                                                                                                                                                |

## Append to existing table

If you are publishing to a pre-existing table, the following data type conversions apply:

- **Columns:** Trifacta data types
- **Rows:** Target table data types

In any table cell, a  $\mathbb{Y}$  indicates that the append operation for that data type mapping is supported.

|           | String | Integer | Datetime | Bool | Float | Map | Array |
|-----------|--------|---------|----------|------|-------|-----|-------|
| VARCHAR   | Y      | Y       | Y        | Y    | Y     | Y   | Y     |
| NVARCHAR  | Y      | Y       | Y        | Y    | Y     | Y   | Y     |
| CHAR      | Y      | Y       | Y        | Y    | Y     | Y   | Y     |
| NCHAR     | Y      | Y       | Y        | Y    | Y     | Y   | Y     |
| TINYINT   |        |         |          |      |       |     |       |
| SMALLINT  |        |         |          |      |       |     |       |
| INT       |        |         |          |      |       |     |       |
| BIGINT    |        | Y       |          |      |       |     |       |
| DATETIME2 |        |         | Y        |      |       |     |       |
| TIME      |        |         | Y        |      |       |     |       |
| BIT       |        |         |          | Y    |       |     |       |
| FLOAT     |        | Y       |          |      | Y     |     |       |
| REAL      |        | Y       |          |      | Y     |     |       |
| DECIMAL   |        | Y       |          |      | Y     |     |       |

# Databricks Tables Data Type Conversions

## Contents:

- *Access/Read*
- *Write/Publish*
  - *Create new table*
  - *Append to existing table*
  - *Notes on Datetime columns*

**NOTE:** The Trifacta® data types listed in this page reflect the raw data type of the converted column. Depending on the contents of the column, the Transformer Page may re-infer a different data type, when a dataset using this type of source is loaded.

## Access/Read

When a Databricks Tables data type is imported, its JDBC data type is remapped according to the following table.

**Tip:** Data precision may be lost during conversion. You may want to generate min and max values and compute significant digits for values in your Hive tables and then compute the same in the Designer Cloud application .

| Source Data Type | Supported? | Trifacta Data Type | Notes                                                                                                                                                                                                                             |
|------------------|------------|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| array            | Y          | Array              |                                                                                                                                                                                                                                   |
| bigint           | Y          | Integer            | <b>NOTE:</b> The Designer Cloud powered by Trifacta platform may infer bigint columns containing very large or very small values as String data type.                                                                             |
| binary           | Y          | String             |                                                                                                                                                                                                                                   |
| boolean          | Y          | Bool               |                                                                                                                                                                                                                                   |
| char             | Y          | String             |                                                                                                                                                                                                                                   |
| date             | Y          | Datetime           |                                                                                                                                                                                                                                   |
| decimal          | Y          | Decimal            |                                                                                                                                                                                                                                   |
| double           | Y          | Decimal            |                                                                                                                                                                                                                                   |
| float            | Y          | Decimal            | <b>NOTE:</b> On import, some float columns may be interpreted as Integer data type in the Designer Cloud powered by Trifacta platform . To fix, you can explicitly set the column's data type to Decimal in the Transformer page. |
| int              | Y          | Integer            |                                                                                                                                                                                                                                   |
| map              | Y          | Object             |                                                                                                                                                                                                                                   |
| smallint         | Y          | Integer            |                                                                                                                                                                                                                                   |
| string           | Y          | String             |                                                                                                                                                                                                                                   |
| struct           | Y          | Object             |                                                                                                                                                                                                                                   |

|           |   |          |  |
|-----------|---|----------|--|
| timestamp | Y | Datetime |  |
| tinyint   | Y | Integer  |  |
| uniontype | N |          |  |
| varchar   | Y | String   |  |

## Write/Publish

### Create new table

**NOTE:** By default, the maximum length of values published to VARCHAR columns is 256 characters. As needed, this limit can be changed for multiple publication targets. For more information, see *Configure Application Limits*.

| Trifacta Data Type | Databricks Tables Data Type                             | Notes                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|--------------------|---------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| String             | string                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Integer            | bigint                                                  | <p><b>NOTE:</b> The Designer Cloud powered by Trifacta platform may infer Integer columns containing very large or very small values as String data type. Before you publish, you should verify that your columns containing extreme values are interpreted as Integer type. You can import a target schema to assist in lining up your columns with the expected target. For more information, see <i>Overview of RapidTarget</i>.</p> |
| Decimal            | double                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Bool               | boolean                                                 |                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Datetime           | Timestamp /string (see Notes on Datetime columns below) | Target data type is based on the underlying data. Time zone information is retained.                                                                                                                                                                                                                                                                                                                                                    |
| Object             | string                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Array              | string                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                         |

### Append to existing table

If you are publishing to a pre-existing table, the following data type conversions apply:

- **Columns:** Trifacta data types
- **Rows:** Target table data types

In any table cell, a Y indicates that the append operation for that data type mapping is supported.

**NOTE:** You cannot append to Databricks Tables map and array column types from Trifacta columns of Map and Array type, even if you imported data from this source.

|         | String | Integer | Datetime | Bool | Decimal | Map | Array | Out of Range error |
|---------|--------|---------|----------|------|---------|-----|-------|--------------------|
| CHAR    | Y      | Y       | Y        | Y    | Y       | Y   | Y     |                    |
| VARCHAR | Y      | Y       | Y        | Y    | Y       | Y   | Y     |                    |
| STRING  | Y      | Y       | Y        | Y    | Y       | Y   | Y     |                    |

|           |  |   |   |   |   |  |      |
|-----------|--|---|---|---|---|--|------|
| INT       |  | Y |   |   |   |  | NULL |
| BIGINT    |  | Y |   |   |   |  | n/a  |
| TINYINT   |  |   |   |   |   |  | NULL |
| SMALLINT  |  |   |   |   |   |  | NULL |
| DECIMAL   |  | Y |   |   | Y |  | NULL |
| DOUBLE    |  | Y |   |   | Y |  | n/a  |
| FLOAT     |  |   |   |   | Y |  | NULL |
| TIMESTAMP |  |   | Y |   |   |  |      |
| BOOLEAN   |  |   |   | Y |   |  |      |

## Notes on Datetime columns

### Run Job

Columns in new tables created for output of `Datetime` columns are written with the Databricks Tables `timestamp` data type. These columns can be appended.

A single job cannot write `Datetime` values to one table as `String` type and to another table as `Timestamp` type. This type of job should be split into multiple types. The table schemas may require modification.

- The above issue may appear as the following error when executing the job:

```
Unable to publish due to datetime data type conflict in column XXXX
```

# Tableau Hyper Data Type Conversions

**NOTE:** The Trifacta® data types listed in this page reflect the raw data type of the converted column. Depending on the contents of the column, the Transformer Page may re-infer a different data type, when a dataset using this type of source is loaded.

## Access/Read

Directing reading of Tableau Hyper data is not supported in the platform.

## Publish/Write

| Trifacta Data Type   | Tableau Data Type         | Notes |
|----------------------|---------------------------|-------|
| String               | SqlType.text()            |       |
| Integer              | SqlType.bigInt()          |       |
| Decimal              | sqlType.doublePrecision() |       |
| Bool                 | SqlType.bool()            |       |
| All other data types | SqlType.text()            |       |

For more information on SqlTypes, see [https://help.tableau.com/current/api/hyper\\_api/en-us/reference/java/index.html](https://help.tableau.com/current/api/hyper_api/en-us/reference/java/index.html).

# Teradata Data Type Conversions

**NOTE:** The Trifacta® data types listed in this page reflect the raw data type of the converted column. Depending on the contents of the column, the Transformer Page may re-infer a different data type, when a dataset using this type of source is loaded.

## Access/Read

| Source Data Type              | Supported | Trifacta Data Type |
|-------------------------------|-----------|--------------------|
| BYTE[(n)]                     | N         |                    |
| VARBYTE[(n)]                  | N         |                    |
| BYTEINT                       | Y         | Integer            |
| SMALLINT                      | Y         | Integer            |
| BIGINT                        | Y         | Integer            |
| INTEGER                       | Y         | Integer            |
| DECIMAL [(n[,m])]             | Y         | Decimal            |
| NUMERIC [(n[,m])]             | Y         | Decimal            |
| REAL                          | Y         | Decimal            |
| DOUBLE PRECISION              | Y         | Decimal            |
| FLOAT                         | Y         | Decimal            |
| NUMBER(n[,m])                 | Y         | Decimal            |
| NUMBER[(*,m)]                 | Y         | Decimal            |
| DATE                          | Y         | String             |
| Time(0)                       | Y         | Datetime           |
| Time(0) WITH TIME ZONE        | Y         | Datetime           |
| TIME (n)                      | Y         | String             |
| TIME (n) WITH TIME ZONE       | Y         | String             |
| TIMESTAMP (0)                 | Y         | String             |
| Timestamp(0) WITH TIME ZONE   | Y         | Datetime           |
| TIMESTAMP (n)                 | Y         | Datetime           |
| TIMESTAMP (n) WITH TIME ZONE  | Y         | String             |
| INTERVAL DAY [(n)]            | Y         | String             |
| INTERVAL DAY [(n)] TO HOUR    | Y         | String             |
| INTERVAL DAY [(n)] TO MINUTE  | Y         | String             |
| INTERVAL DAY [(n)] TO SECOND  | Y         | String             |
| INTERVAL HOUR [(n)]           | Y         | String             |
| INTERVAL HOUR [(n)] TO MINUTE | Y         | String             |
| INTERVAL HOUR [(n)] TO SECOND | Y         | String             |
| INTERVAL MINUTE [(n)]         | Y         | String             |
|                               |           |                    |



|                                       |   |        |
|---------------------------------------|---|--------|
| INTERVAL MINUTE [(n)] TO SECOND [(m)] | Y | String |
| INTERVAL MONTH                        | Y | String |
| INTERVAL SECOND [(n],[m])]            | Y | String |
| INTERVAL YEAR [(n)]                   | Y | String |
| INTERVAL YEAR [(n)] TO MONTH          | Y | String |
| CHAR[(n)]                             | Y | String |
| CHARACTER(n) CHARACTER SET GRAPHIC    | N |        |
| VARCHAR(n)                            | Y | String |
| CHAR VARYING(n)                       | Y | String |
| LONG VARCHAR                          | Y | String |
| LONG VARGRAPHIC                       | N |        |
| VARGRAPHIC                            | N |        |
| PERIOD(DATE)                          | N |        |
| PERIOD(TIME (0))                      | N |        |
| PERIOD(TIME (0)) WITH TIME ZONE       | N |        |
| PERIOD(TIME (n))                      | N |        |
| PERIOD(TIME (n)) WITH TIME ZONE       | N |        |
| PERIOD(TIMESTAMP(0))                  | N |        |
| PERIOD(TIMESTAMP(0)) WITH TIME ZONE   | N |        |
| PERIOD(TIMESTAMP(n))                  | N |        |
| PERIOD(TIMESTAMP(n)) WITH TIME ZONE   | N |        |
| BLOB                                  | N |        |
| CLOB                                  | Y | String |
| XML                                   | Y | String |
| ARRAY                                 | Y | Array  |
| VARARRAY                              | Y | Array  |
| TD_ANYTYPE                            | N |        |
| VARIANT_TYPE                          | N |        |
| Distinct                              | N |        |
| Structured                            | N |        |
| ST_CircularString                     | N |        |
| ST_CompoundCurve                      | N |        |
| ST_Curve                              | N |        |
| ST_CurvePolygon                       | N |        |
| ST_GeomCollection                     | N |        |
| ST_Geometry                           | N |        |
| ST_LineString                         | N |        |
| ST_MultiCurve                         | N |        |
| ST_MultiLineString                    | N |        |

|                 |   |        |
|-----------------|---|--------|
| ST_MultiPoint   | N |        |
| ST_MultiPolygon | N |        |
| ST_MultiSurface | N |        |
| ST_Point        | N |        |
| ST_Polygon      | N |        |
| ST_Surface      | N |        |
| JSON            | Y | String |

## Write/Publish

| Trifacta Data Type | Teradata Column Type | Notes |
|--------------------|----------------------|-------|
| Bool               | VARCHAR(256)         |       |
| Integer            | INT                  |       |
|                    | INTEGER              |       |
|                    | SMALLINT             |       |
|                    | VARCHAR(256)         |       |
| String             | VARCHAR(256)         |       |
| Datetime           | TIMESTAMP            |       |
|                    | VARCHAR(256)         |       |
| Timestamp          | VARCHAR(256)         |       |
| Float              | FLOAT                |       |
|                    | DECIMAL              |       |
|                    | NUMERIC              |       |
|                    | VARCHAR(256)         |       |
| Map                | VARCHAR(256)         |       |
| Array              | VARCHAR(256)         |       |

**NOTE:** If you are appending to an existing table where a column's Trifacta data type is not mapped to the column data type in the target table, a validation error is thrown, as the Designer Cloud powered by Trifacta platform writes unmapped types as String data type.

# SharePoint Data Type Conversions

**NOTE:** The Trifacta® data types listed in this page reflect the raw data type of the converted column. Depending on the contents of the column, the Transformer Page may re-infer a different data type, when a dataset using this type of source is loaded.

## Access/Read

**NOTE:** Image, Link, and Address fields from SharePoint are imported into Designer Cloud powered by Trifacta Enterprise Edition as JSON strings. Incorrect formatting or modification of these strings within the product may result in errors when published back to SharePoint Lists.

| Source Data Type       | Supported? | Trifacta Data Type | Notes                                                                               |
|------------------------|------------|--------------------|-------------------------------------------------------------------------------------|
| Choice (menu)          | Y          | String             |                                                                                     |
| Currency               | Y          | Float              |                                                                                     |
| Date and Time          | Y          | Datetime           |                                                                                     |
| Link and Image         | Y          | String (JSON)      |                                                                                     |
| Lookup                 | Y          | String             |                                                                                     |
| Address                | Y          | String (JSON)      |                                                                                     |
| Multiple lines of text | Y          | String             | Lines in the source are demarcated in the String value using the newline character. |
| Number                 | Y          | Float              |                                                                                     |
| Person or Group        | Y          | String (JSON)      |                                                                                     |
| Single line of text    | Y          | String             |                                                                                     |
| Task outcome           | Y          | String             |                                                                                     |
| Yes/No                 | Y          | Boolean            |                                                                                     |

## Publish/Write

| Trifacta Data Type | SharePoint List Data Types                         | Notes                                                                                                    |
|--------------------|----------------------------------------------------|----------------------------------------------------------------------------------------------------------|
| String             | String<br>Note<br>Text<br>Varchar<br>URL<br>Choice | <b>NOTE:</b> Secondary types are published only if appending or truncating an existing table. See below. |
| Integer            | Integer                                            |                                                                                                          |
| Decimal            | Float<br>Currency                                  |                                                                                                          |
| Datetime           | Date and Time                                      |                                                                                                          |

|                      |                                   |                                                                                                                       |
|----------------------|-----------------------------------|-----------------------------------------------------------------------------------------------------------------------|
| Bool                 | Boolean                           |                                                                                                                       |
| Map                  | Note<br>String<br>Text<br>Varchar | <div> <b>NOTE:</b> Secondary types are published only if appending or truncating an existing table. See below. </div> |
| Array                | Note<br>String<br>Text<br>Varchar | <div> <b>NOTE:</b> Secondary types are published only if appending or truncating an existing table. See below. </div> |
| All other data types | String                            |                                                                                                                       |

For more information, see [http://cdn.cdata.com/help/RSF/jdbc/pg\\_datatypemapping.htm](http://cdn.cdata.com/help/RSF/jdbc/pg_datatypemapping.htm).

Depending on the publishing action, output data types may vary.

### Appending or truncating a SharePoint List:

If you are appending or truncating an existing SharePoint List, all column data types are published to the target data types as expected, as long as the data being published can be parsed and published by SharePoint.

### Writing to a new SharePoint List:

Some data types in Designer Cloud powered by Trifacta Enterprise Edition do not have a direct mapping to SharePoint List data types. Examples:

- Address, Link, Person, Choice, and Image data types are published to SharePoint Lists as a multi-line String.
- Currency is published as a Number.
- Datetime values in Designer Cloud powered by Trifacta Enterprise Edition are written back to SharePoint Lists as DateTime types, even if the source data from SharePoint is Date type.

# Transformation Reference

## Contents:

- *Scale to min max*
- *One hot encode*
- *Scale to mean*
- *Bin column*
- *Change column type*
- *Comment*
- *Conditional column*
- *Convert patterns*
- *Count matches*
- *Count matches between delimiters*
- *Remove duplicate rows*
- *New formula*
- *Delete columns*
- *Extract between delimiters*
- *Extract text or pattern*
- *Extract first*
- *Convert key/value to Object*
- *Extract last*
- *Extract matches to Array*
- *Extract between positions*
- *Extract mismatched*
- *Extract numbers*
- *Extract query strings*
- *Filter contains*
- *Filter custom formula*
- *Filter ends with*
- *Filter exact*
- *Filter not equals*
- *Filter from top*
- *Filter greater than*
- *Filter at interval*
- *Filter less than*
- *Filter missing*
- *Filter mismatched*
- *Filter in*
- *Filter range*
- *Filter starts with*
- *Expand Array to rows*
- *Group by*
- *Join datasets*
- *Lowercase text*
- *Pad with leading*
- *Merge columns*
- *Move columns*
- *Nest columns*
- *Pivot*
- *Prefix text*
- *Propercase text*
- *Remove symbols in text*
- *Remove whitespace in text*
- *Remove accents in text*
- *Rename columns*
- *Rename with pattern*
- *Rename with prefix*

- *Rename with row(s)*
- *Rename with suffix*
- *Rename to UPPERCASE*
- *Rename to lowercase*
- *Rename from beginning*
- *Rename from end*
- *Rename by removing special characters*
- *Replace cells*
- *Replace text or pattern*
- *Replace between delimiters*
- *Replace between positions*
- *Replace mismatched*
- *Replace missing*
- *Select*
- *Edit with formula*
- *Sort rows*
- *Split on text or pattern*
- *Split between delimiters*
- *Split with multiple delimiters*
- *Split at positions*
- *Split at interval*
- *Split between positions*
- *Split into rows*
- *Suffix text*
- *Trim whitespace*
- *Trim quotes*
- *Invoke external function*
- *Uppercase text*
- *Date format*
- *Union datasets*
- *Standardize column*
- *Create column from examples*
- *Unnest elements*
- *Unpivot*
- *Convert values to columns*
- *Window*
- *sourcerownumber*
- *filepath*

This section contains reference information on the transformations available in Designer Cloud powered by Trifacta® Enterprise Edition.

**Tip:** Use the values in the Title column as search strings in the Search panel to begin specifying these transformations.

| Name                 | Title                   | Description                                                                                                                |
|----------------------|-------------------------|----------------------------------------------------------------------------------------------------------------------------|
| scaleminmax          | <b>Scale to min max</b> | Scale a column to a specific min max range. See <i>Prepare Data for Machine Processing</i> .                               |
| onehotencode         | <b>One hot encode</b>   | Create a column for each unique value indicating its presence or absence. See <i>Prepare Data for Machine Processing</i> . |
| scalestandar<br>dize | <b>Scale to mean</b>    | Scale a column to zero mean and unit variance. See <i>Prepare Data for Machine Processing</i> .                            |
| bincolumn            | <b>Bin column</b>       | Bin values into ranges of equal or custom size. See <i>Prepare Data for Machine Processing</i> .                           |
| changetype           |                         | Changes the data type of a column [settype]. See <i>Change Column Data Type</i> .                                          |

|                            |                                         |                                                                                                                                      |
|----------------------------|-----------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------|
|                            | <b>Change column type</b>               |                                                                                                                                      |
| comment                    | <b>Comment</b>                          | Adds a comment to your recipe [comment]. See <i>Add Comments to Your Recipe</i> .                                                    |
| conditions                 | <b>Conditional column</b>               | Returns values based on conditions such as if-then-else or case statements. See <i>Apply Conditional Transformations</i> .           |
| convertpattern             | <b>Convert patterns</b>                 | Finds one or more patterns or text literals and replaces them with specified pattern values. See <i>Standardize Using Patterns</i> . |
| countmatches               | <b>Count matches</b>                    | Counts the number of matches [countpattern]. See <i>Compute Counts</i> .                                                             |
| countmatchbetween          | <b>Count matches between delimiters</b> | Counts the number of matches [countpattern]. See <i>Compute Counts</i> .                                                             |
| deduplicate                | <b>Remove duplicate rows</b>            | Removes duplicate rows where values in every column are the same. See <i>Deduplicate Data</i> .                                      |
| derive                     | <b>New formula</b>                      | Creates a new column with the result of a formula.                                                                                   |
| drop                       | <b>Delete columns</b>                   | Delete one or more columns. See <i>Remove Data</i> .                                                                                 |
| extractbetween delimiters  | <b>Extract between delimiters</b>       | Extracts text found between two patterns. See <i>Extract Values</i> .                                                                |
| extractcustom              | <b>Extract text or pattern</b>          | Extracts text found between two patterns. Variant: Custom text or pattern. See <i>Extract Values</i> .                               |
| extractfirst characters    | <b>Extract first</b>                    | Extracts text according to its position. Variant: Extract the first n characters. See <i>Extract Values</i> .                        |
| extractkv                  | <b>Convert key/value to Object</b>      | Extracts key-value pairs into an Object [extractkv]. See <i>Extract Values</i> .                                                     |
| extractlast characters     | <b>Extract last</b>                     | Extracts key-value pairs into an Object [extractkv]. Variant: Extract the last n characters. See <i>Extract Values</i> .             |
| extractlist                | <b>Extract matches to Array</b>         | Extracts a list into an Array [extractlist]. See <i>Extract Values</i> .                                                             |
| extractrange of characters | <b>Extract between positions</b>        | Extracts text according to its position. Variant: Extract the last n characters. See <i>Extract Values</i> .                         |
| extractmismatched          | <b>Extract mismatched</b>               | Extracts a list into an Array [extractlist]. Variant: The data type to match against. See <i>Extract Values</i> .                    |
| extractnumbers             | <b>Extract numbers</b>                  | Extracts a list into an Array [extractlist]. Variant: Extract numbers from a text. See <i>Extract Values</i> .                       |
| extractquery strings       | <b>Extract query strings</b>            | Extracts a list into an Array [extractlist]. Variant: Extract fields from an URL query string. See <i>Extract Values</i> .           |
| filtercontains             | <b>Filter contains</b>                  | Filter rows that satisfy a condition. Variant: Filter rows that contain a specified value or pattern. See <i>Filter Data</i> .       |
| filtercustom               | <b>Filter custom formula</b>            | Filter rows that satisfy a condition. Variant: Filter rows that satisfy an arbitrary formula. See <i>Filter Data</i> .               |
| filterendswith             | <b>Filter ends with</b>                 | Filter rows that satisfy a condition. Variant: Filter rows that ends with a specified value or pattern. See <i>Filter Data</i> .     |
| filterexactly              | <b>Filter exact</b>                     | Filter rows that satisfy a condition. Variant: Filter rows that match exactly a specified value. See <i>Filter Data</i> .            |
| filternot                  | <b>Filter not equals</b>                | Filters rows that do not satisfy a condition. See <i>Filter Data</i> .                                                               |
| filterfromtop              | <b>Filter from top</b>                  | Filter rows by their position. Variant: Filter rows from the top. See <i>Filter Data</i> .                                           |
| filtergreaterthan          | <b>Filter greater than</b>              | Filter rows that satisfy a condition. Variant: Filter rows with values greater than (or equal                                        |

|                  |                                  |                                                                                                                                                          |
|------------------|----------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| an               |                                  | to) a specified value. See <i>Filter Data</i> .                                                                                                          |
| filterinterval   | <b>Filter at interval</b>        | Filter rows by their position. Variant: . Variant: The size of the interval to filter rows at. See <i>Filter Data</i> .                                  |
| filterlessthan   | <b>Filter less than</b>          | Filter rows that satisfy a condition. Variant: Filter rows with values less than (or equal to) a specified value. See <i>Filter Data</i> .               |
| filtermissing    | <b>Filter missing</b>            | Filter rows that satisfy a condition. Variant: Filter rows with missing values. See <i>Remove Data</i> .                                                 |
| filtermismatched | <b>Filter mismatched</b>         | Filter rows that satisfy a condition. Variant: Filter rows with mismatched values. See <i>Filter Data</i> .                                              |
| filteroneof      | <b>Filter in</b>                 | Filter rows that satisfy a condition. Variant: Filter rows that match any of the specified values. See <i>Filter Data</i> .                              |
| filterrange      | <b>Filter range</b>              | Filter rows by their position. Variant: Filter rows within a range. See <i>Filter Data</i> .                                                             |
| filterstartswith | <b>Filter starts with</b>        | Filter rows that satisfy a condition. Variant: Filter rows that starts with a specified value or pattern. See <i>Filter Data</i> .                       |
| flatten          | <b>Expand Array to rows</b>      | Converts each element in an Array into a new row. See <i>Working with Arrays</i> .                                                                       |
| groupby          | <b>Group by</b>                  | Group data and perform aggregated calculations on it. See <i>Create Aggregations</i> .                                                                   |
| join             | <b>Join datasets</b>             | Adds additional columns from other data sources [join]. See <i>Join Window</i> .                                                                         |
| lowercase        | <b>Lowercase text</b>            | Format text in columns. Variant: Convert text in column to lowercase. See <i>Modify String Values</i> .                                                  |
| leftpad          | <b>Pad with leading</b>          | Format text in columns. Variant: Add the necessary number of characters to each value to make them of the same length. See <i>Modify String Values</i> . |
| merge            | <b>Merge columns</b>             | Concatenates the values from two or more columns into a new column [merge]. See <i>Add Two Columns</i> .                                                 |
| move             | <b>Move columns</b>              | Moves one or more columns before or after another column [move]. See <i>Move Columns</i> .                                                               |
| nest             | <b>Nest columns</b>              | Converts columns into an Object or Array [nest]. See <i>Working with Arrays</i> .                                                                        |
| pivot            | <b>Pivot</b>                     | Creates a new column for each unique value in a column [pivot]. See <i>Pivot Data</i> .                                                                  |
| prefix           | <b>Prefix text</b>               | Format text in columns. Variant: Specify a prefix to be added at the beginning of each selected column name. See <i>Modify String Values</i> .           |
| propercase       | <b>Propercase text</b>           | Format text in columns. Variant: Convert text in column to ProperCase. See <i>Modify String Values</i> .                                                 |
| removesymbols    | <b>Remove symbols in text</b>    | Format text in columns. Variant: Remove all non-alphanumeric characters from the text. See <i>Remove Data</i> .                                          |
| removewhitespace | <b>Remove whitespace in text</b> | Format text in columns. Variant: Remove all whitespace found in the text. See <i>Remove Data</i> .                                                       |
| removeaccents    | <b>Remove accents in text</b>    | Remove accent marks from text. See <i>Modify String Values</i> .                                                                                         |
| rename           | <b>Rename columns</b>            | Renames one or more columns [rename]. See <i>Rename Columns</i> .                                                                                        |
| renamepattern    | <b>Rename with pattern</b>       | Renames one or more columns [rename]. See <i>Rename Columns</i> .                                                                                        |
| renameprefix     | <b>Rename with prefix</b>        | Renames one or more columns [rename]. See <i>Rename Columns</i> .                                                                                        |
| renameheader     | <b>Rename with row(s)</b>        | Renames one or more columns [rename]. See <i>Rename Columns</i> .                                                                                        |
| renamesuffix     | <b>Rename with suffix</b>        | Renames one or more columns [rename]. See <i>Rename Columns</i> .                                                                                        |
| renameupper      |                                  | Renames one or more columns [rename]. See <i>Rename Columns</i> .                                                                                        |



|                          |                                              |                                                                                                                                                                 |
|--------------------------|----------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                          | <b>Rename to UPPERCASE</b>                   |                                                                                                                                                                 |
| renamelower              | <b>Rename to lowercase</b>                   | Renames one or more columns [rename]. See <i>Rename Columns</i> .                                                                                               |
| renamekeepleft           | <b>Rename from beginning</b>                 | Renames one or more columns [rename]. See <i>Rename Columns</i> .                                                                                               |
| renamekeepright          | <b>Rename from end</b>                       | Renames one or more columns [rename]. See <i>Rename Columns</i> .                                                                                               |
| renamesanitize           | <b>Rename by removing special characters</b> | Renames one or more columns [rename]. See <i>Rename Columns</i> .                                                                                               |
| replacecell              | <b>Replace cells</b>                         | Renames one or more columns [rename]. See <i>Rename Columns</i> .                                                                                               |
| replacepattern           | <b>Replace text or pattern</b>               | Replace text matching a pattern. See <i>Replace Cell Values</i> .                                                                                               |
| replacebetweenpatterns   | <b>Replace between delimiters</b>            | Replace text between delimiters. Variant: Replace text between delimiters. See <i>Replace Cell Values</i> .                                                     |
| replacebetweenpositions  | <b>Replace between positions</b>             | Replace text between delimiters. Variant: Replaces text based on position. See <i>Replace Cell Values</i> .                                                     |
| replacemismatched        | <b>Replace mismatched</b>                    | Replace mismatched values. See <i>Replace Cell Values</i> .                                                                                                     |
| replacemissing           | <b>Replace missing</b>                       | Replace missing values. See <i>Replace Cell Values</i> .                                                                                                        |
| select                   | <b>Select</b>                                | Create a new table of columns <i>Selectd</i> from your current dataset. See <i>Select</i> .                                                                     |
| set                      | <b>Edit with formula</b>                     | Sets the values of one or more columns to the result of a formula [set].                                                                                        |
| sort                     | <b>Sort rows</b>                             | Sorts the rows based on the values in one or more columns.                                                                                                      |
| splitondelimiter         | <b>Split on text or pattern</b>              | Split by delimiter. Variant: Text or pattern. See <i>Split Column</i> .                                                                                         |
| splitbetween delimiters  | <b>Split between delimiters</b>              | Split by delimiter. Variant: Between two delimiters. See <i>Split Column</i> .                                                                                  |
| splitmultiple delimiters | <b>Split with multiple delimiters</b>        | Split by delimiter. Variant: By multiple delimiters. See <i>Split Column</i> .                                                                                  |
| splitpositions           | <b>Split at positions</b>                    | Split by character position. Variant: By positions. See <i>Split Column</i> .                                                                                   |
| splitevery               | <b>Split at interval</b>                     | Split by character position. Variant: At regular interval. See <i>Split Column</i> .                                                                            |
| splitbetween positions   | <b>Split between positions</b>               | Split by character position. Variant: Between two positions. See <i>Split Column</i> .                                                                          |
| splitrows                | <b>Split into rows</b>                       | Splits raw data into rows [splitrows]. See <i>Split Column</i> .                                                                                                |
| suffix                   | <b>Suffix text</b>                           | Format text in columns. Variant: Specify a suffix to be added to the end of each selected column name. See <i>Modify String Values</i> .                        |
| trimwhitespace           | <b>Trim whitespace</b>                       | Format text in columns. Variant: Remove all whitespaces found at the beginning and end of the text. See <i>Modify String Values</i> .                           |
| trimquotes               | <b>Trim quotes</b>                           | Format text in columns. Variant: Remove quotes found at the beginning and end of the text. See <i>Modify String Values</i> .                                    |
| udf                      | <b>Invoke external function</b>              | Creates a new column with the result of an external function.<br><div></div> |

|                 |                                    |                                                                                                                                                                                                                                                    |
|-----------------|------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                 |                                    | <p><b>Feature Availability:</b> This feature may not be available in all product editions. For more information on available features, see <i>Compare Editions</i>.</p> <p><b>NOTE:</b> This transformation requires additional configuration.</p> |
| uppercase       | <b>Uppercase text</b>              | Format text in columns. Variant: Convert text in column to UPPERCASE. See <i>Modify String Values</i> .                                                                                                                                            |
| dateformat      | <b>Date format</b>                 | Change format for Datetime columns. See <i>Format Dates</i> .                                                                                                                                                                                      |
| union           | <b>Union datasets</b>              | Adds additional rows from other data source [union]. See <i>Union Page</i> .                                                                                                                                                                       |
| standardize     | <b>Standardize column</b>          | Single-column standardization for standardizing column values. See <i>Standardize Page</i> .                                                                                                                                                       |
| columnbyexample | <b>Create column from examples</b> | Create a new column by providing example values. See <i>Create Column by Example</i> .                                                                                                                                                             |
| unnest          | <b>Unnest elements</b>             | Extracts elements from an Object or Array into columns. See <i>Working with Arrays</i> .                                                                                                                                                           |
| unpivot         | <b>Unpivot</b>                     | Turns columns into rows. Produces a key column with unnested values. See <i>Pivot Data</i> .                                                                                                                                                       |
| valuestocols    | <b>Convert values to columns</b>   | Creates a new column for each unique value in a column [valuestocols]. See <i>Pivot Data</i> .                                                                                                                                                     |
| window          | <b>Window</b>                      | Performs row-based calculations across multiple ordered rows [window]. See <i>Window Functions</i> .                                                                                                                                               |
| sourcerownumber | <b>sourcerownumber</b>             | Generate a new column containing the row number for each row from the source, if available. See <i>Source Metadata References</i> .                                                                                                                |
| filepath        | <b>filepath</b>                    | Generate a new column containing the path to the source file, if available. See <i>Source Metadata References</i> .                                                                                                                                |

# Plan Metadata References

## Contents:

- *General Syntax*
  - *\$plan References*
  - *\$http References*
    - *Response references*
  - *\$slack References*
  - *\$flow References*
    - *Output references*
  - *Additional References*
- 

In the body and header of HTTP tasks in your plans, you can reference the following elements of metadata from the plan run for additional contextual information.

## General Syntax

All plan metadata references follow the following basic syntax:

```
{{ $plan.path.to.reference }}
```

- All references can be entered with \$ in the Designer Cloud application . These references are turned into { \$ in the code definition. The double-curly braces forms the environment for metadata replacement.

**Tip:** In the Designer Cloud application , you can start by typing \$.

- Nodes in the tree are separated with a . period.

Reference values that contain whitespace must be listed in the following manner:

```
{{ $plan.path['path with white space in it'].rest.of.path }}
```

## Notes:

- In the Designer Cloud application , you can use double-quotes when specifying a whitespace value. However, these double-quotes get escaped in the actual request. It is safer and more consistent to use single quotes.

Whitespace values typically appear when referencing the display name values for underlying objects, like recipes executed as part of a flow task.

## \$plan References

These references apply to the plan definition or current plan run.

Text to enter:

```
$plan.
```

| Reference | Description                                                                                                                                                                                                         |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| name      | Name of the plan that is run.                                                                                                                                                                                       |
| duration  | Length of time that the plan ran or has run so far <div> <b>Tip:</b> To return a more readable form of this duration value, use the following reference: <pre>{{ \$plan.duration   humanizeDuration }}</pre> </div> |
| startTime | Timestamp for when the plan run began                                                                                                                                                                               |
| runId     | Internal identifier for this run of the plan                                                                                                                                                                        |
| user      | Internal identifier of the user who launched this run.                                                                                                                                                              |
| taskCount | Count of tasks in the plan run.                                                                                                                                                                                     |

## \$http References

These references apply to HTTP tasks in the plan run.

Enter the following, after which you can see the two-letter codes for the HTTP tasks that have already executed in the current plan run:

```
$http_ax.
```

| Reference  | Description                                                                |
|------------|----------------------------------------------------------------------------|
| name       | Name of the HTTP task                                                      |
| status     | Current status of the task execution                                       |
| duration   | Length of time that the task ran or has run so far                         |
| startTime  | Timestamp for when the task began. A null value if the task has not begun. |
| endTime    | Timestamp for when the task ended. A null value if it has not ended yet.   |
| statusCode | Status code (if any) returned from the receiving endpoint                  |
| response   | Response information. See below.                                           |

## Response references

These references apply to the response returned as part of the task execution.

Enter the following, after which you can see the two-letter codes for the HTTP tasks that have already executed in the current plan run:

```
$http_ax.response.
```

| Reference | Description                            |
|-----------|----------------------------------------|
| body      | Body of the response                   |
| json      | JSON-formatted version of the response |
| headers   | Headers returned with the response     |

## \$slack References

You can reference metadata from Slack tasks in the current plan run using the following reference types:

```
$slack_ax.
```

Supported metadata is identical to the metadata for HTTP tasks. See the previous section for details.

## \$flow References

These references apply to flow tasks in the plan run.

Enter the following, after which you can see the two-letter codes for the HTTP tasks that have already executed in the current plan run:

```
$flow_ax.
```

| Reference | Description                                                                |
|-----------|----------------------------------------------------------------------------|
| name      | Name of the flow task                                                      |
| status    | Current status of the task execution                                       |
| duration  | Length of time that the task ran or has run so far                         |
| startTime | Timestamp for when the task began. A null value if the task has not begun. |
| endTime   | Timestamp for when the task ended. A null value if it has not ended yet.   |
| jobIds    | Internal identifiers for the jobs that were run as part of this flow task  |
| flowName  | Name of the flow underlying this flow task                                 |
| output    | Metadata from the flow task's output. See below.                           |
| params    | Parameters created in the flow can be referenced in the task.              |

## Output references

These references apply to the outputs that are generated in the flow tasks of the plan run.

Enter the following for flow task 7p with output My Output Name:

```
$flow_7p['My Output Name'].
```

| Reference | Description      |
|-----------|------------------|
| name      | Name of the flow |
|           |                  |

|                       |                                                                                            |
|-----------------------|--------------------------------------------------------------------------------------------|
| status                | Current status of the flow                                                                 |
| duration              | Length of time that the flow execution ran or has run so far                               |
| startTime             | Timestamp for when the flow execution began. A null value if the run has not begun.        |
| endTime               | Timestamp for when the flow execution ended. A null value if it has not ended yet.         |
| lastUpdate            | Timestamp for when the flow was last modified                                              |
| jobId                 | Internal identifier for the job that was run or is running for the flow                    |
| user                  | Internal identifier for the user who executed the job                                      |
| jobType               |                                                                                            |
| fileSize              | If the output generates a file or files, this value captures the size in KB of the output. |
| environment           | Running environment where the job was executed                                             |
| columnCount           | Count of columns generated in the output                                                   |
| rowCount              | Count of rows generated in the output                                                      |
| dataTypeCount         | Count of Trifacta data types detected in the output                                        |
| validValuesCount      | Count of valid values in the output                                                        |
| mismatchedValuesCount | Count of mismatched values in the output                                                   |
| emptyValuesCount      | Count of missing or empty values in the output                                             |
| columns               | Column information from the selected output for the flow.                                  |
| sources               | Source filename and table information from the imported datasets.                          |

## Additional References

Plan metadata reference information leverages the Nunjucks templating language, which provides additional capabilities such as loops, conditions, filters, and helper functions.

**NOTE:** These additional capabilities are available through the language, but their implementation in the D esigner Cloud application has not been certified. For Nunjucks capabilities not listed on this page, you should experiment with them in a development environment first.

For more information, see <https://mozilla.github.io/nunjucks/templating.html>.

# cron Schedule Syntax Reference

Contents:

- Overview of cron
  - Cron syntax
  - Special characters
- Examples
  - Hourly
  - Daily
  - Weekly
  - Weekdays
  - Monthly
  - Yearly
  - Other examples
  - Unsupported cron expressions
  - Invalid cron expressions

This section describes the syntax for defining scheduled executions using cron in Designer Cloud powered by Trifacta® Enterprise Edition. Typically, this method is used for repeated schedules.

Flow schedules:

- Flow owners can define scheduled executions of flows from within the Flow View page.
- Collaborators can review and cannot edit schedules.
- See *Flow View Page*.

**NOTE:** Time zone settings defined in the Designer Cloud application page where you are specifying your cron schedule are used with the schedule. To use UTC time zone, select `UTC` in the drop-down. For more information, see *Supported Time Zone Values*.

## Overview of cron

Designer Cloud powered by Trifacta Enterprise Edition allows you to make use of cron, a widely used syntax, for specifying times that recur at regular intervals. You can use cron to specify schedules on a per-minute or annual basis and arbitrary intervals in between.

### Cron syntax

A cron scheduled is defined as a space-separated string of values. The following cron example defines a schedule to be triggered at 11:30:00pm on February 1:

| minute | hour | day of month | month | day of week |
|--------|------|--------------|-------|-------------|
| 30     | 23   | 1            | 2     | *           |

When all values are matched, the cron job is triggered.

**NOTE:** Specification of seconds is not supported.

### Wildcards:

In the above cron expression, the wildcard \* can be used to match any accepted value, which means that the cron value type is not a factor in determining this schedule. Since the wildcard is applied to the day of week value, the schedule can be triggered on any day of the week.

**NOTE:** You must use the \* character in either the day-of-week or day-of-month fields. Specifying both fields in the same cron expression is not supported.

### Legend:

Except for the final field (year), all fields are required in the cron expression. Special characters are described below the table.

| Value | Type         | Description                                                                                         | Supported Special Characters |
|-------|--------------|-----------------------------------------------------------------------------------------------------|------------------------------|
| 30    | minute       | 0-59                                                                                                | , - * /                      |
| 23    | hour         | 0-23                                                                                                | , - * /                      |
| 1     | day of month | 1-31                                                                                                | , - * / L W                  |
| 2     | month        | 1-12                                                                                                | , - * /                      |
| *     | day of week  | 0 - 6 or Sun - Sat<br>0, Sun, SUN = Sunday<br>1, Mon, MON = Monday<br>...<br>6, Sat, SAT = Saturday | , - * / L #                  |
| *     | year         | (Optional) You can specify year settings if needed. Default is *.                                   | , - * /                      |

### Special characters

You can use the following special characters in your cron expressions.

| Character | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| *         | ("all values") - Wildcard to match all possible values in the field. For example, if you wanted your trigger to fire every minute of the 10pm hour, the minute character in the expression is *. An example is below.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| -         | Specify a range of values. For example, you could use 1 – 5 in the day-of-week field to match the work days of the week (Monday through Friday). An example is below.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| ,         | Specify a discrete set of values. For example, an entry of 1 , 10 , 20 , 30 for the day of month field is triggered on the 1st, 10th, 20th, and 30th (if possible) of the month.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| /         | Specify increments of the field in the units of the field. For example, 5 / 20 in the minutes field matches on the 5th, 25th, and 45th minute of each hour.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| L         | Last value accepted in the range is accepted in the following fields: <ul style="list-style-type: none"> <li>Day-of-month: Specifies the last day of the month for the currently selected month value. <ul style="list-style-type: none"> <li>In January, this value matches with 31.</li> <li>In February, this value matches with 28 for non-leap years.</li> <li>In April, this value matches with 30.</li> </ul> </li> <li>Day-of-week: <ul style="list-style-type: none"> <li>By itself, it specifies the last day of the week, which matches with 6 (Saturday).</li> <li>When used with another value, it specifies the last matching value for the month. For example, 3L is the last Wednesday of the month.</li> </ul> </li> </ul> |
| W         | Specifies the nearest matching weekday. For example, an entry of 22W in the day-of-month field matches on the nearest weekday to the 22nd of the month. If the 22nd is a Saturday, then the cron job matches on the 24th (the following Monday).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |



**Tip:** LW can be used in the day-of-month field to match on the last weekday of the month.

# Specifies the nth day of the month. Examples for the day-of-week field:

- 3#4 - fourth Tuesday of the month
- 5#2 - second Thursday of the month

## Examples

Below are some example cron schedules.

### Hourly

Runs at minute 15 of every hour:

```
15 * * * *
```

### Daily

Runs every day at 10pm:

```
0 22 * * *
```

Runs every minute of the 10pm hour every day:

```
* 22 * * *
```

### Weekly

Runs every Tuesday at 3am:

```
0 3 * * 2
```

### Weekdays

Runs each weekday at 8pm:

```
0 20 * * 1-5
```

Note that the above schedule runs at 10pm on Monday night and each night of the week at that time.

To refresh the flow for each weekday morning, you might choose to start the schedules on Sunday, in which the day-of-week value starts with 0 and ends with 4.

### Monthly

Runs the first day of each month at 2:30am:

```
30 2 1 * *
```

Runs at 3:30pm on the nearest weekday (W) to the 25th of the month:

```
30 15 25W * *
```

- If the 25th is a Saturday, the above triggers on Friday the 24th.
- If the 25th is a Sunday, the above triggers on Monday the 26th.

## Yearly

Runs at midnight of January 1 each year:

```
0 0 1 1 * *
```

## Other examples

| Expression                   | Meaning                                                                                                                             |
|------------------------------|-------------------------------------------------------------------------------------------------------------------------------------|
| <pre>0 12 * * *</pre>        | Fire at 12pm (noon) every day                                                                                                       |
| <pre>15 10 * * *</pre>       | Fire at 10:15am every day                                                                                                           |
| <pre>15 10 * * *</pre>       | Fire at 10:15am every day                                                                                                           |
| <pre>15 10 * * * *</pre>     | Fire at 10:15am every day                                                                                                           |
| <pre>15 10 * * * 2017</pre>  | Fire at 10:15am every day during the year 2017                                                                                      |
| <pre>* 14 * * *</pre>        | Fire every minute starting at 2pm and ending at 2:59pm, every day                                                                   |
| <pre>0/5 14 * * *</pre>      | Fire every 5 minutes starting at 2pm and ending at 2:55pm, every day                                                                |
| <pre>0/5 14,18 * * *</pre>   | Fire every 5 minutes starting at 2pm and ending at 2:55pm, AND fire every 5 minutes starting at 6pm and ending at 6:55pm, every day |
| <pre>0-5 14 * * *</pre>      | Fire every minute starting at 2pm and ending at 2:05pm, every day                                                                   |
| <pre>10,44 14 * 3 WED</pre>  | Fire at 2:10pm and at 2:44pm every Wednesday in the month of March.                                                                 |
| <pre>15 10 * * MON-FRI</pre> | Fire at 10:15am every Monday, Tuesday, Wednesday, Thursday and Friday                                                               |

|                           |                                                                                          |
|---------------------------|------------------------------------------------------------------------------------------|
|                           |                                                                                          |
| 15 10 15 * *              | Fire at 10:15am on the 15th day of every month                                           |
| 15 10 L * *               | Fire at 10:15am on the last day of every month                                           |
| 15 10 L-2 * *             | Fire at 10:15am on the 2nd-to-last last day of every month                               |
| 15 10 * * 5L              | Fire at 10:15am on the last Friday of every month                                        |
| 15 10 * * 5L<br>2017-2019 | Fire at 10:15am on every last friday of every month during the years 2017, 2018 and 2019 |
| 15 10 * * 5#3             | Fire at 10:15am on the third Friday of every month                                       |
| 0 12 1/5 * *              | Fire at 12pm (noon) every 5 days every month, starting on the first day of the month.    |
| 11 11 11 11 *             | Fire every November 11th at 11:11am.                                                     |

## Unsupported cron expressions

**NOTE:** The Designer Cloud application does not support mixing / and – special characters in the same expressions.

Instead of expressing ranges in your cron syntax, you can reference all possible options.

| Invalid expression | Valid expression      |
|--------------------|-----------------------|
| 0 23 * 1-11/2 * *  | 0 23 * 2,4,6,8,10 * * |

## Invalid cron expressions

| Expression          | Meaning                                        | Reason                                                                                 |
|---------------------|------------------------------------------------|----------------------------------------------------------------------------------------|
| 15 10 * * *<br>2001 | Fire at 10:15am every day during the year 2001 | This cron expression is invalid because it will not generate any events in the future. |
|                     | -                                              | The cron expression should contain 6 or 7 fields.                                      |

|       |  |  |
|-------|--|--|
| * * * |  |  |
|-------|--|--|

# Supported Time Zone Values

## American Time Zones

The following American time zones are mapped to the time zone values supported in Designer Cloud powered by Trifacta® Enterprise Edition:

**Tip:** Designer Cloud powered by Trifacta Enterprise Edition does make adjustments for Daylight Savings Time.

| US Time Zone | Time Zone Value       | Standard Time (UTC) | Daylight Savings Time (UTC) | Included States and Territories                                                                                                                                                                                                                                             | Notes                                                                                          |
|--------------|-----------------------|---------------------|-----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------|
| Atlantic     | America / Puerto_Rico | UTC-04:00           | n/a                         | Puerto Rico, US Virgin Islands                                                                                                                                                                                                                                              | Daylight Savings Time is not observed.                                                         |
| Eastern      | US / Eastern          | UTC-05:00           | UTC-04:00                   | Entire: Connecticut, Delaware, Georgia, Maine, Maryland, Massachusetts, New Hampshire, New Jersey, New York, North Carolina, Ohio, Pennsylvania, Rhode Island, South Carolina, Vermont, Virginia, West Virginia<br>Partial: Florida, Indiana, Kentucky, Michigan, Tennessee |                                                                                                |
| Central      | US / Central          | UTC-06:00           | UTC-05:00                   | Entire: Alabama, Arkansas, Illinois, Iowa, Louisiana, Minnesota, Mississippi, Missouri, Oklahoma, Wisconsin<br>Partial: Florida, Indiana, Kansas, Kentucky, Michigan, Nebraska, North Dakota, South Dakota, Tennessee, Texas                                                |                                                                                                |
| Mountain     | US / Mountain         | UTC-07:00           | UTC-06:00                   | Entire: Arizona, Colorado, Montana, New Mexico, Utah, Wyoming<br>Partial: Idaho, Kansas, Nebraska, Nevada, North Dakota, Oregon, South Dakota, Texas                                                                                                                        | Most of Arizona does not observe Daylight Savings Time. Use US / Arizona in this area.         |
| Pacific      | US / Pacific          | UTC-08:00           | UTC-07:00                   | Entire: California, Washington<br>Partial: Idaho, Nevada, Oregon                                                                                                                                                                                                            |                                                                                                |
| Alaska       | US / Alaska           | UTC-09:00           | UTC-08:00                   | Partial: Alaska                                                                                                                                                                                                                                                             | Daylight Savings Time is not observed in the Aleutian Islands. Use US / Aleutian in this area. |
| Hawaii       | US / Hawaii           | UTC-10:00           | UTC-09:00                   | Entire: Hawaii<br>Partial: Alaska                                                                                                                                                                                                                                           | Daylight Savings Time is not observed in Hawaii.                                               |

## Global Time Zone Values

For the functions that support use of specified time zones, you can apply the following string values as parameters to specify the time zone:

| Time Zone Value | Standard time | Daylight Saving | Partial country coverage |
|-----------------|---------------|-----------------|--------------------------|
|-----------------|---------------|-----------------|--------------------------|

|                      | (UTC)     | time (UTC) |                           |
|----------------------|-----------|------------|---------------------------|
| Africa/Abidjan       | UTC+00:00 | UTC+00:00  |                           |
| Africa/Accra         | UTC+00:00 | UTC+00:00  |                           |
| Africa/Addis_Ababa   | UTC+03:00 | UTC+03:00  |                           |
| Africa/Algiers       | UTC+01:00 | UTC+01:00  |                           |
| Africa/Asmara        | UTC+03:00 | UTC+03:00  |                           |
| Africa/Bamako        | UTC+00:00 | UTC+00:00  |                           |
| Africa/Bangui        | UTC+01:00 | UTC+01:00  |                           |
| Africa/Banjul        | UTC+00:00 | UTC+00:00  |                           |
| Africa/Bissau        | UTC+00:00 | UTC+00:00  |                           |
| Africa/Blantyre      | UTC+02:00 | UTC+02:00  |                           |
| Africa/Brazzaville   | UTC+01:00 | UTC+01:00  |                           |
| Africa/Bujumbura     | UTC+02:00 | UTC+02:00  |                           |
| Africa/Cairo         | UTC+02:00 | UTC+02:00  |                           |
| Africa/Casablanca    | UTC+01:00 | UTC+01:00  |                           |
| Africa/Ceuta         | UTC+01:00 | UTC+01:00  | Ceuta, Melilla            |
| Africa/Conakry       | UTC+00:00 | UTC+00:00  |                           |
| Africa/Dakar         | UTC+00:00 | UTC+00:00  |                           |
| Africa/Dar_es_Salaam | UTC+03:00 | UTC+03:00  |                           |
| Africa/Djibouti      | UTC+03:00 | UTC+03:00  |                           |
| Africa/Douala        | UTC+01:00 | UTC+01:00  |                           |
| Africa/EI_Aaiun      | UTC+00:00 | UTC+01:00  |                           |
| Africa/Freetown      | UTC+00:00 | UTC+00:00  |                           |
| Africa/Gaborone      | UTC+02:00 | UTC+02:00  |                           |
| Africa/Harare        | UTC+02:00 | UTC+02:00  |                           |
| Africa/Johannesburg  | UTC+02:00 | UTC+02:00  |                           |
| Africa/Juba          | UTC+03:00 | UTC+03:00  |                           |
| Africa/Kampala       | UTC+03:00 | UTC+03:00  |                           |
| Africa/Khartoum      | UTC+02:00 | UTC+02:00  |                           |
| Africa/Kigali        | UTC+02:00 | UTC+02:00  |                           |
| Africa/Kinshasa      | UTC+01:00 | UTC+01:00  | Dem. Rep. of Congo (west) |
| Africa/Lagos         | UTC+01:00 | UTC+01:00  |                           |
| Africa/Libreville    | UTC+01:00 | UTC+01:00  |                           |
| Africa/Lome          | UTC+00:00 | UTC+00:00  |                           |
| Africa/Luanda        | UTC+01:00 | UTC+01:00  |                           |
| Africa/Lubumbashi    | UTC+02:00 | UTC+02:00  | Dem. Rep. of Congo (east) |
| Africa/Lusaka        | UTC+02:00 | UTC+02:00  |                           |
| Africa/Malabo        | UTC+01:00 | UTC+01:00  |                           |
| Africa/Maputo        | UTC+02:00 | UTC+02:00  |                           |
| Africa/Maseru        | UTC+02:00 | UTC+02:00  |                           |
|                      |           |            |                           |

|                                      |           |           |                                                        |
|--------------------------------------|-----------|-----------|--------------------------------------------------------|
| Africa/Mbabane                       | UTC+02:00 | UTC+02:00 |                                                        |
| Africa/Mogadishu                     | UTC+03:00 | UTC+03:00 |                                                        |
| Africa/Monrovia                      | UTC+00:00 | UTC+00:00 |                                                        |
| Africa/Nairobi                       | UTC+03:00 | UTC+03:00 |                                                        |
| Africa/Ndjamena                      | UTC+01:00 | UTC+01:00 |                                                        |
| Africa/Niamey                        | UTC+01:00 | UTC+01:00 |                                                        |
| Africa/Nouakchott                    | UTC+00:00 | UTC+00:00 |                                                        |
| Africa/Ouagadougou                   | UTC+00:00 | UTC+00:00 |                                                        |
| Africa/Porto-Novo                    | UTC+01:00 | UTC+01:00 |                                                        |
| Africa/Sao_Tome                      | UTC+00:00 | UTC+00:00 |                                                        |
| Africa/Timbuktu                      | UTC+00:00 | UTC+00:00 |                                                        |
| Africa/Tripoli                       | UTC+02:00 | UTC+02:00 |                                                        |
| Africa/Tunis                         | UTC+01:00 | UTC+01:00 |                                                        |
| Africa/Windhoek                      | UTC+02:00 | UTC+02:00 |                                                        |
| America/Adak                         | UTC10:00  | UTC09:00  | Aleutian Islands                                       |
| America/Anchorage                    | UTC09:00  | UTC08:00  | Alaska (most areas)                                    |
| America/Anguilla                     | UTC04:00  | UTC04:00  |                                                        |
| America/Antigua                      | UTC04:00  | UTC04:00  |                                                        |
| America/Araguaina                    | UTC03:00  | UTC03:00  | Tocantins                                              |
| America/Argentina<br>/Buenos_Aires   | UTC03:00  | UTC03:00  | Buenos Aires (BA, CF)                                  |
| America/Argentina<br>/Catamarca      | UTC03:00  | UTC03:00  | Catamarca (CT); Chubut (CH)                            |
| America/Argentina<br>/ComodRivadavia | UTC03:00  | UTC03:00  |                                                        |
| America/Argentina/Cordoba            | UTC03:00  | UTC03:00  | Argentina (most areas: CB, CC, CN, ER, FM, MN, SE, SF) |
| America/Argentina/Jujuy              | UTC03:00  | UTC03:00  | Jujuy (JY)                                             |
| America/Argentina/La_Rioja           | UTC03:00  | UTC03:00  | La Rioja (LR)                                          |
| America/Argentina/Mendoza            | UTC03:00  | UTC03:00  | Mendoza (MZ)                                           |
| America/Argentina<br>/Rio_Gallegos   | UTC03:00  | UTC03:00  | Santa Cruz (SC)                                        |
| America/Argentina/Salta              | UTC03:00  | UTC03:00  | Salta (SA, LP, NQ, RN)                                 |
| America/Argentina<br>/San_Juan       | UTC03:00  | UTC03:00  | San Juan (SJ)                                          |
| America/Argentina/San_Luis           | UTC03:00  | UTC03:00  | San Luis (SL)                                          |
| America/Argentina/Tucuman            | UTC03:00  | UTC03:00  | Tucuman (TM)                                           |
| America/Argentina/Ushuaia            | UTC03:00  | UTC03:00  | Tierra del Fuego (TF)                                  |
| America/Aruba                        | UTC04:00  | UTC04:00  |                                                        |
| America/Asuncion                     | UTC04:00  | UTC03:00  |                                                        |
| America/Atikokan                     | UTC05:00  | UTC05:00  | EST - ON (Atikokan); NU (Coral H)                      |
| America/Atka                         | UTC10:00  | UTC09:00  |                                                        |
| America/Bahia                        | UTC03:00  | UTC03:00  | Bahia                                                  |
| America/Bahia_Banderas               | UTC06:00  | UTC05:00  | Central Time - Bahia de Banderas                       |

|                       |           |           |                                        |
|-----------------------|-----------|-----------|----------------------------------------|
| America/Barbados      | UTC04:00  | UTC04:00  |                                        |
| America/Belem         | UTC03:00  | UTC03:00  | Para (east); Amapa                     |
| America/Belize        | UTC06:00  | UTC06:00  |                                        |
| America/Blanc-Sablon  | UTC04:00  | UTC04:00  | AST - QC (Lower North Shore)           |
| America/Boa_Vista     | UTC04:00  | UTC04:00  | Roraima                                |
| America/Bogota        | UTC05:00  | UTC05:00  |                                        |
| America/Boise         | UTC07:00  | UTC06:00  | Mountain - ID (south); OR (east)       |
| America/Buenos_Aires  | UTC03:00  | UTC03:00  |                                        |
| America/Cambridge_Bay | UTC07:00  | UTC06:00  | Mountain - NU (west)                   |
| America/Campo_Grande  | UTC04:00  | UTC03:00  | Mato Grosso do Sul                     |
| America/Cancun        | UTC05:00  | UTC05:00  | Eastern Standard Time - Quintana Roo   |
| America/Caracas       | UTC04:00  | UTC04:00  |                                        |
| America/Catamarca     | UTC03:00  | UTC03:00  |                                        |
| America/Cayenne       | UTC03:00  | UTC03:00  |                                        |
| America/Cayman        | UTC05:00  | UTC05:00  |                                        |
| America/Chicago       | UTC06:00  | UTC05:00  | Central (most areas)                   |
| America/Chihuahua     | UTC07:00  | UTC06:00  | Mountain Time - Chihuahua (most areas) |
| America/Coral_Harbour | UTC05:00  | UTC05:00  |                                        |
| America/Cordoba       | UTC03:00  | UTC03:00  |                                        |
| America/Costa_Rica    | UTC06:00  | UTC06:00  |                                        |
| America/Creston       | UTC07:00  | UTC07:00  | MST - BC (Creston)                     |
| America/Cuiaba        | UTC04:00  | UTC03:00  | Mato Grosso                            |
| America/Curacao       | UTC04:00  | UTC04:00  |                                        |
| America/Danmarkshavn  | UTC+00:00 | UTC+00:00 | National Park (east coast)             |
| America/Dawson        | UTC08:00  | UTC07:00  | Pacific - Yukon (north)                |
| America/Dawson_Creek  | UTC07:00  | UTC07:00  | MST - BC (Dawson Cr, Ft St John)       |
| America/Denver        | UTC07:00  | UTC06:00  | Mountain (most areas)                  |
| America/Detroit       | UTC05:00  | UTC04:00  | Eastern - MI (most areas)              |
| America/Dominica      | UTC04:00  | UTC04:00  |                                        |
| America/Edmonton      | UTC07:00  | UTC06:00  | Mountain - AB; BC (E); SK (W)          |
| America/Eirunepe      | UTC05:00  | UTC05:00  | Amazonas (west)                        |
| America/El_Salvador   | UTC06:00  | UTC06:00  |                                        |
| America/Ensenada      | UTC08:00  | UTC07:00  |                                        |
| America/Fort_Nelson   | UTC07:00  | UTC07:00  | MST - BC (Ft Nelson)                   |
| America/Fort_Wayne    | UTC05:00  | UTC04:00  |                                        |
| America/Fortaleza     | UTC03:00  | UTC03:00  | Brazil (northeast: MA, PI, CE, RN, PB) |
| America/Glace_Bay     | UTC04:00  | UTC03:00  | Atlantic - NS (Cape Breton)            |
| America/Godthab       | UTC03:00  | UTC02:00  | Greenland (most areas)                 |
| America/Goose_Bay     | UTC04:00  | UTC03:00  | Atlantic - Labrador (most areas)       |
| America/Grand_Turk    | UTC05:00  | UTC04:00  |                                        |
|                       |           |           |                                        |



|                              |          |          |                                                                |
|------------------------------|----------|----------|----------------------------------------------------------------|
| America/Grenada              | UTC04:00 | UTC04:00 |                                                                |
| America/Guadeloupe           | UTC04:00 | UTC04:00 |                                                                |
| America/Guatemala            | UTC06:00 | UTC06:00 |                                                                |
| America/Guayaquil            | UTC05:00 | UTC05:00 | Ecuador (mainland)                                             |
| America/Guyana               | UTC04:00 | UTC04:00 |                                                                |
| America/Halifax              | UTC04:00 | UTC03:00 | Atlantic - NS (most areas); PE                                 |
| America/Havana               | UTC05:00 | UTC04:00 |                                                                |
| America/Hermosillo           | UTC07:00 | UTC07:00 | Mountain Standard Time - Sonora                                |
| America/Indiana/Indianapolis | UTC05:00 | UTC04:00 | Eastern - IN (most areas)                                      |
| America/Indiana/Knox         | UTC06:00 | UTC05:00 | Central - IN (Starke)                                          |
| America/Indiana/Marengo      | UTC05:00 | UTC04:00 | Eastern - IN (Crawford)                                        |
| America/Indiana/Petersburg   | UTC05:00 | UTC04:00 | Eastern - IN (Pike)                                            |
| America/Indiana/Tell_City    | UTC06:00 | UTC05:00 | Central - IN (Perry)                                           |
| America/Indiana/Vevay        | UTC05:00 | UTC04:00 | Eastern - IN (Switzerland)                                     |
| America/Indiana/Vincennes    | UTC05:00 | UTC04:00 | Eastern - IN (Da, Du, K, Mn)                                   |
| America/Indiana/Winamac      | UTC05:00 | UTC04:00 | Eastern - IN (Pulaski)                                         |
| America/Indianapolis         | UTC05:00 | UTC04:00 |                                                                |
| America/Inuvik               | UTC07:00 | UTC06:00 | Mountain - NT (west)                                           |
| America/Iqaluit              | UTC05:00 | UTC04:00 | Eastern - NU (most east areas)                                 |
| America/Jamaica              | UTC05:00 | UTC05:00 |                                                                |
| America/Jujuy                | UTC03:00 | UTC03:00 |                                                                |
| America/Juneau               | UTC09:00 | UTC08:00 | Alaska - Juneau area                                           |
| America/Kentucky/Louisville  | UTC05:00 | UTC04:00 | Eastern - KY (Louisville area)                                 |
| America/Kentucky/Monticello  | UTC05:00 | UTC04:00 | Eastern - KY (Wayne)                                           |
| America/Knox_IN              | UTC06:00 | UTC05:00 |                                                                |
| America/Kralendijk           | UTC04:00 | UTC04:00 |                                                                |
| America/La_Paz               | UTC04:00 | UTC04:00 |                                                                |
| America/Lima                 | UTC05:00 | UTC05:00 |                                                                |
| America/Los_Angeles          | UTC08:00 | UTC07:00 | Pacific                                                        |
| America/Louisville           | UTC05:00 | UTC04:00 |                                                                |
| America/Lower_Princes        | UTC04:00 | UTC04:00 |                                                                |
| America/Maceio               | UTC03:00 | UTC03:00 | Alagoas, Sergipe                                               |
| America/Managua              | UTC06:00 | UTC06:00 |                                                                |
| America/Manaus               | UTC04:00 | UTC04:00 | Amazonas (east)                                                |
| America/Marigot              | UTC04:00 | UTC04:00 |                                                                |
| America/Martinique           | UTC04:00 | UTC04:00 |                                                                |
| America/Matamoros            | UTC06:00 | UTC05:00 | Central Time US - Coahuila, Nuevo Leon, Tamaulipas (US border) |
| America/Mazatlan             | UTC07:00 | UTC06:00 | Mountain Time - Baja California Sur, Nayarit, Sinaloa          |
| America/Mendoza              | UTC03:00 | UTC03:00 |                                                                |
| America/Menominee            | UTC06:00 | UTC05:00 | Central - MI (Wisconsin border)                                |

|                                 |          |          |                                                                       |
|---------------------------------|----------|----------|-----------------------------------------------------------------------|
| America/Merida                  | UTC06:00 | UTC05:00 | Central Time - Campeche, Yucatan                                      |
| America/Metlakatla              | UTC09:00 | UTC08:00 | Alaska - Annette Island                                               |
| America/Mexico_City             | UTC06:00 | UTC05:00 | Central Time                                                          |
| America/Miquelon                | UTC03:00 | UTC02:00 |                                                                       |
| America/Moncton                 | UTC04:00 | UTC03:00 | Atlantic - New Brunswick                                              |
| America/Monterrey               | UTC06:00 | UTC05:00 | Central Time - Durango; Coahuila, Nuevo Leon, Tamaulipas (most areas) |
| America/Montevideo              | UTC03:00 | UTC03:00 |                                                                       |
| America/Montreal                | UTC05:00 | UTC04:00 |                                                                       |
| America/Montserrat              | UTC04:00 | UTC04:00 |                                                                       |
| America/Nassau                  | UTC05:00 | UTC04:00 |                                                                       |
| America/New_York                | UTC05:00 | UTC04:00 | Eastern (most areas)                                                  |
| America/Nipigon                 | UTC05:00 | UTC04:00 | Eastern - ON, QC (no DST 196773)                                      |
| America/Nome                    | UTC09:00 | UTC08:00 | Alaska (west)                                                         |
| America/Noronha                 | UTC02:00 | UTC02:00 | Atlantic islands                                                      |
| America/North_Dakota /Beulah    | UTC06:00 | UTC05:00 | Central - ND (Mercer)                                                 |
| America/North_Dakota /Center    | UTC06:00 | UTC05:00 | Central - ND (Oliver)                                                 |
| America/North_Dakota /New_Salem | UTC06:00 | UTC05:00 | Central - ND (Morton rural)                                           |
| America/Ojinaga                 | UTC07:00 | UTC06:00 | Mountain Time US - Chihuahua (US border)                              |
| America/Panama                  | UTC05:00 | UTC05:00 |                                                                       |
| America/Pangnirtung             | UTC05:00 | UTC04:00 | Eastern - NU (Pangnirtung)                                            |
| America/Paramaribo              | UTC03:00 | UTC03:00 |                                                                       |
| America/Phoenix                 | UTC07:00 | UTC07:00 | MST - Arizona (except Navajo)                                         |
| America/Port_of_Spain           | UTC04:00 | UTC04:00 |                                                                       |
| America/Port-au-Prince          | UTC05:00 | UTC04:00 |                                                                       |
| America/Porto_Acre              | UTC05:00 | UTC05:00 |                                                                       |
| America/Porto_Velho             | UTC04:00 | UTC04:00 | Rondonia                                                              |
| America/Puerto_Rico             | UTC04:00 | UTC04:00 |                                                                       |
| America/Punta_Arenas            | UTC03:00 | UTC03:00 | Region of Magallanes                                                  |
| America/Rainy_River             | UTC06:00 | UTC05:00 | Central - ON (Rainy R, Ft Frances)                                    |
| America/Rankin_Inlet            | UTC06:00 | UTC05:00 | Central - NU (central)                                                |
| America/Recife                  | UTC03:00 | UTC03:00 | Pernambuco                                                            |
| America/Regina                  | UTC06:00 | UTC06:00 | CST - SK (most areas)                                                 |
| America/Resolute                | UTC06:00 | UTC05:00 | Central - NU (Resolute)                                               |
| America/Rio_Branco              | UTC05:00 | UTC05:00 | Acre                                                                  |
| America/Rosario                 | UTC03:00 | UTC03:00 |                                                                       |
| America/Santa_Isabel            | UTC08:00 | UTC07:00 |                                                                       |
| America/Santarem                | UTC03:00 | UTC03:00 | Para (west)                                                           |
| America/Santiago                | UTC04:00 | UTC03:00 | Chile (most areas)                                                    |
|                                 |          |          |                                                                       |

|                           |           |           |                                                        |
|---------------------------|-----------|-----------|--------------------------------------------------------|
| America/Santo_Domingo     | UTC04:00  | UTC04:00  |                                                        |
| America/Sao_Paulo         | UTC03:00  | UTC03:00  | Brazil (southeast: GO, DF, MG, ES, RJ, SP, PR, SC, RS) |
| America/Scoresbysund      | UTC01:00  | UTC+00:00 | Scoresbysund/Ittoqqortoormiit                          |
| America/Shiprock          | UTC07:00  | UTC06:00  |                                                        |
| America/Sitka             | UTC09:00  | UTC08:00  | Alaska - Sitka area                                    |
| America/St_Barthelemy     | UTC04:00  | UTC04:00  |                                                        |
| America/St_Johns          | UTC03:30  | UTC02:30  | Newfoundland; Labrador (southeast)                     |
| America/St_Kitts          | UTC04:00  | UTC04:00  |                                                        |
| America/St_Lucia          | UTC04:00  | UTC04:00  |                                                        |
| America/St_Thomas         | UTC04:00  | UTC04:00  |                                                        |
| America/St_Vincent        | UTC04:00  | UTC04:00  |                                                        |
| America/Swift_Current     | UTC06:00  | UTC06:00  | CST - SK (midwest)                                     |
| America/Tegucigalpa       | UTC06:00  | UTC06:00  |                                                        |
| America/Thule             | UTC04:00  | UTC03:00  | Thule/Pituffik                                         |
| America/Thunder_Bay       | UTC05:00  | UTC04:00  | Eastern - ON (Thunder Bay)                             |
| America/Tijuana           | UTC08:00  | UTC07:00  | Pacific Time US - Baja California                      |
| America/Toronto           | UTC05:00  | UTC04:00  | Eastern - ON, QC (most areas)                          |
| America/Tortola           | UTC04:00  | UTC04:00  |                                                        |
| America/Vancouver         | UTC08:00  | UTC07:00  | Pacific - BC (most areas)                              |
| America/Virgin            | UTC04:00  | UTC04:00  |                                                        |
| America/Whitehorse        | UTC08:00  | UTC07:00  | Pacific - Yukon (south)                                |
| America/Winnipeg          | UTC06:00  | UTC05:00  | Central - ON (west); Manitoba                          |
| America/Yakutat           | UTC09:00  | UTC08:00  | Alaska - Yakutat                                       |
| America/Yellowknife       | UTC07:00  | UTC06:00  | Mountain - NT (central)                                |
| Antarctica/Casey          | UTC+11:00 | UTC+11:00 | Casey                                                  |
| Antarctica/Davis          | UTC+07:00 | UTC+07:00 | Davis                                                  |
| Antarctica/DumontDURville | UTC+10:00 | UTC+10:00 | Dumont-d'Urville                                       |
| Antarctica/Macquarie      | UTC+11:00 | UTC+11:00 | Macquarie Island                                       |
| Antarctica/Mawson         | UTC+05:00 | UTC+05:00 | Mawson                                                 |
| Antarctica/McMurdo        | UTC+12:00 | UTC+13:00 | New Zealand time - McMurdo, South Pole                 |
| Antarctica/Palmer         | UTC03:00  | UTC03:00  | Palmer                                                 |
| Antarctica/Rothera        | UTC03:00  | UTC03:00  | Rothera                                                |
| Antarctica/South_Pole     | UTC+12:00 | UTC+13:00 |                                                        |
| Antarctica/Syowa          | UTC+03:00 | UTC+03:00 | Syowa                                                  |
| Antarctica/Troll          | UTC+00:00 | UTC+02:00 | Troll                                                  |
| Antarctica/Vostok         | UTC+06:00 | UTC+06:00 | Vostok                                                 |
| Arctic/Longyearbyen       | UTC+01:00 | UTC+02:00 |                                                        |
| Asia/Aden                 | UTC+03:00 | UTC+03:00 |                                                        |
| Asia/Almaty               | UTC+06:00 | UTC+06:00 | Kazakhstan (most areas)                                |
| Asia/Amman                | UTC+02:00 | UTC+03:00 |                                                        |
|                           |           |           |                                                        |

|                  |           |           |                                                       |
|------------------|-----------|-----------|-------------------------------------------------------|
| Asia/Anadyr      | UTC+12:00 | UTC+12:00 | MSK+09 - Bering Sea                                   |
| Asia/Aqtau       | UTC+05:00 | UTC+05:00 | Mangghystau/Mankistau                                 |
| Asia/Aqtobe      | UTC+05:00 | UTC+05:00 | Aqtobe/Aktobe                                         |
| Asia/Ashgabat    | UTC+05:00 | UTC+05:00 |                                                       |
| Asia/Ashkhabad   | UTC+05:00 | UTC+05:00 |                                                       |
| Asia/Atyrau      | UTC+05:00 | UTC+05:00 | Atyrau/Atirau/Gur'yev                                 |
| Asia/Baghdad     | UTC+03:00 | UTC+03:00 |                                                       |
| Asia/Bahrain     | UTC+03:00 | UTC+03:00 |                                                       |
| Asia/Baku        | UTC+04:00 | UTC+04:00 |                                                       |
| Asia/Bangkok     | UTC+07:00 | UTC+07:00 |                                                       |
| Asia/Barnaul     | UTC+07:00 | UTC+07:00 | MSK+04 - Altai                                        |
| Asia/Beirut      | UTC+02:00 | UTC+03:00 |                                                       |
| Asia/Bishkek     | UTC+06:00 | UTC+06:00 |                                                       |
| Asia/Brunei      | UTC+08:00 | UTC+08:00 |                                                       |
| Asia/Calcutta    | UTC+05:30 | UTC+05:30 |                                                       |
| Asia/Chita       | UTC+09:00 | UTC+09:00 | MSK+06 - Zabaykalsky                                  |
| Asia/Choibalsan  | UTC+08:00 | UTC+08:00 | Dornod, Sukhbaatar                                    |
| Asia/Chongqing   | UTC+08:00 | UTC+08:00 |                                                       |
| Asia/Chungking   | UTC+08:00 | UTC+08:00 |                                                       |
| Asia/Colombo     | UTC+05:30 | UTC+05:30 |                                                       |
| Asia/Dacca       | UTC+06:00 | UTC+06:00 |                                                       |
| Asia/Damascus    | UTC+02:00 | UTC+03:00 |                                                       |
| Asia/Dhaka       | UTC+06:00 | UTC+06:00 |                                                       |
| Asia/Dili        | UTC+09:00 | UTC+09:00 |                                                       |
| Asia/Dubai       | UTC+04:00 | UTC+04:00 |                                                       |
| Asia/Dushanbe    | UTC+05:00 | UTC+05:00 |                                                       |
| Asia/Famagusta   | UTC+02:00 | UTC+02:00 | Northern Cyprus                                       |
| Asia/Gaza        | UTC+02:00 | UTC+03:00 | Gaza Strip                                            |
| Asia/Harbin      | UTC+08:00 | UTC+08:00 |                                                       |
| Asia/Hebron      | UTC+02:00 | UTC+03:00 | West Bank                                             |
| Asia/Ho_Chi_Minh | UTC+07:00 | UTC+07:00 |                                                       |
| Asia/Hong_Kong   | UTC+08:00 | UTC+08:00 |                                                       |
| Asia/Hovd        | UTC+07:00 | UTC+07:00 | Bayan-Olgii, Govi-Altai, Hovd, Uvs, Zavkhan           |
| Asia/Irkutsk     | UTC+08:00 | UTC+08:00 | MSK+05 - Irkutsk, Buryatia                            |
| Asia/Istanbul    | UTC+03:00 | UTC+03:00 |                                                       |
| Asia/Jakarta     | UTC+07:00 | UTC+07:00 | Java, Sumatra                                         |
| Asia/Jayapura    | UTC+09:00 | UTC+09:00 | New Guinea (West Papua / Irian Jaya); Maluku/Moluccas |
| Asia/Jerusalem   | UTC+02:00 | UTC+03:00 |                                                       |
| Asia/Kabul       | UTC+04:30 | UTC+04:30 |                                                       |
| Asia/Kamchatka   | UTC+12:00 | UTC+12:00 | MSK+09 - Kamchatka                                    |
|                  |           |           |                                                       |

|                    |           |           |                                                                           |
|--------------------|-----------|-----------|---------------------------------------------------------------------------|
| Asia/Karachi       | UTC+05:00 | UTC+05:00 |                                                                           |
| Asia/Kashgar       | UTC+06:00 | UTC+06:00 |                                                                           |
| Asia/Kathmandu     | UTC+05:45 | UTC+05:45 |                                                                           |
| Asia/Katmandu      | UTC+05:45 | UTC+05:45 |                                                                           |
| Asia/Khandyga      | UTC+09:00 | UTC+09:00 | MSK+06 - Tomponsky, Ust-Maysky                                            |
| Asia/Kolkata       | UTC+05:30 | UTC+05:30 |                                                                           |
| Asia/Krasnoyarsk   | UTC+07:00 | UTC+07:00 | MSK+04 - Krasnoyarsk area                                                 |
| Asia/Kuala_Lumpur  | UTC+08:00 | UTC+08:00 | Malaysia (peninsula)                                                      |
| Asia/Kuching       | UTC+08:00 | UTC+08:00 | Sabah, Sarawak                                                            |
| Asia/Kuwait        | UTC+03:00 | UTC+03:00 |                                                                           |
| Asia/Macao         | UTC+08:00 | UTC+08:00 |                                                                           |
| Asia/Macau         | UTC+08:00 | UTC+08:00 |                                                                           |
| Asia/Magadan       | UTC+11:00 | UTC+11:00 | MSK+08 - Magadan                                                          |
| Asia/Makassar      | UTC+08:00 | UTC+08:00 | Borneo (east, south); Sulawesi/Celebes, Bali, Nusa Tenggara; Timor (west) |
| Asia/Manila        | UTC+08:00 | UTC+08:00 |                                                                           |
| Asia/Muscat        | UTC+04:00 | UTC+04:00 |                                                                           |
| Asia/Novokuznetsk  | UTC+07:00 | UTC+07:00 | MSK+04 - Kemerovo                                                         |
| Asia/Novosibirsk   | UTC+07:00 | UTC+07:00 | MSK+04 - Novosibirsk                                                      |
| Asia/Omsk          | UTC+06:00 | UTC+06:00 | MSK+03 - Omsk                                                             |
| Asia/Oral          | UTC+05:00 | UTC+05:00 | West Kazakhstan                                                           |
| Asia/Phnom_Penh    | UTC+07:00 | UTC+07:00 |                                                                           |
| Asia/Pontianak     | UTC+07:00 | UTC+07:00 | Borneo (west, central)                                                    |
| Asia/Pyongyang     | UTC+09:00 | UTC+09:00 |                                                                           |
| Asia/Qatar         | UTC+03:00 | UTC+03:00 |                                                                           |
| Asia/Qyzylorda     | UTC+05:00 | UTC+05:00 | Qyzylorda/Kyzylorda/Kzyl-Orda                                             |
| Asia/Rangoon       | UTC+06:30 | UTC+06:30 |                                                                           |
| Asia/Riyadh        | UTC+03:00 | UTC+03:00 |                                                                           |
| Asia/Saigon        | UTC+07:00 | UTC+07:00 |                                                                           |
| Asia/Sakhalin      | UTC+11:00 | UTC+11:00 | MSK+08 - Sakhalin Island                                                  |
| Asia/Samarkand     | UTC+05:00 | UTC+05:00 | Uzbekistan (west)                                                         |
| Asia/Seoul         | UTC+09:00 | UTC+09:00 |                                                                           |
| Asia/Shanghai      | UTC+08:00 | UTC+08:00 | Beijing Time                                                              |
| Asia/Singapore     | UTC+08:00 | UTC+08:00 |                                                                           |
| Asia/Srednekolymsk | UTC+11:00 | UTC+11:00 | MSK+08 - Sakha (E); North Kuril Is                                        |
| Asia/Taipei        | UTC+08:00 | UTC+08:00 |                                                                           |
| Asia/Tashkent      | UTC+05:00 | UTC+05:00 | Uzbekistan (east)                                                         |
| Asia/Tbilisi       | UTC+04:00 | UTC+04:00 |                                                                           |
| Asia/Tehran        | UTC+03:30 | UTC+04:30 |                                                                           |
| Asia/Tel_Aviv      | UTC+02:00 | UTC+03:00 |                                                                           |
| Asia/Thimbu        | UTC+06:00 | UTC+06:00 |                                                                           |

|                        |           |           |                                 |
|------------------------|-----------|-----------|---------------------------------|
| Asia/Thimphu           | UTC+06:00 | UTC+06:00 |                                 |
| Asia/Tokyo             | UTC+09:00 | UTC+09:00 |                                 |
| Asia/Tomsk             | UTC+07:00 | UTC+07:00 | MSK+04 - Tomsk                  |
| Asia/Ujung_Pandang     | UTC+08:00 | UTC+08:00 |                                 |
| Asia/Ulaanbaatar       | UTC+08:00 | UTC+08:00 | Mongolia (most areas)           |
| Asia/Ulan_Bator        | UTC+08:00 | UTC+08:00 |                                 |
| Asia/Urumqi            | UTC+06:00 | UTC+06:00 | Xinjiang Time                   |
| Asia/Ust-Nera          | UTC+10:00 | UTC+10:00 | MSK+07 - Oymyakonsky            |
| Asia/Vientiane         | UTC+07:00 | UTC+07:00 |                                 |
| Asia/Vladivostok       | UTC+10:00 | UTC+10:00 | MSK+07 - Amur River             |
| Asia/Yakutsk           | UTC+09:00 | UTC+09:00 | MSK+06 - Lena River             |
| Asia/Yangon            | UTC+06:30 | UTC+06:30 |                                 |
| Asia/Yekaterinburg     | UTC+05:00 | UTC+05:00 | MSK+02 - Urals                  |
| Asia/Yerevan           | UTC+04:00 | UTC+04:00 |                                 |
| Atlantic/Azores        | UTC01:00  | UTC+00:00 | Azores                          |
| Atlantic/Bermuda       | UTC04:00  | UTC03:00  |                                 |
| Atlantic/Canary        | UTC+00:00 | UTC+01:00 | Canary Islands                  |
| Atlantic/Cape_Verde    | UTC01:00  | UTC01:00  |                                 |
| Atlantic/Faeroe        | UTC+00:00 | UTC+01:00 |                                 |
| Atlantic/Faroe         | UTC+00:00 | UTC+01:00 |                                 |
| Atlantic/Jan_Mayen     | UTC+01:00 | UTC+02:00 |                                 |
| Atlantic/Madeira       | UTC+00:00 | UTC+01:00 | Madeira Islands                 |
| Atlantic/Reykjavik     | UTC+00:00 | UTC+00:00 |                                 |
| Atlantic/South_Georgia | UTC02:00  | UTC02:00  |                                 |
| Atlantic/St_Helena     | UTC+00:00 | UTC+00:00 |                                 |
| Atlantic/Stanley       | UTC03:00  | UTC03:00  |                                 |
| Australia/ACT          | UTC+10:00 | UTC+11:00 |                                 |
| Australia/Adelaide     | UTC+09:30 | UTC+10:30 | South Australia                 |
| Australia/Brisbane     | UTC+10:00 | UTC+10:00 | Queensland (most areas)         |
| Australia/Broken_Hill  | UTC+09:30 | UTC+10:30 | New South Wales (Yancowinna)    |
| Australia/Canberra     | UTC+10:00 | UTC+11:00 |                                 |
| Australia/Currie       | UTC+10:00 | UTC+11:00 | Tasmania (King Island)          |
| Australia/Darwin       | UTC+09:30 | UTC+09:30 | Northern Territory              |
| Australia/Eucla        | UTC+08:45 | UTC+08:45 | Western Australia (Eucla)       |
| Australia/Hobart       | UTC+10:00 | UTC+11:00 | Tasmania (most areas)           |
| Australia/LHI          | UTC+10:30 | UTC+11:00 |                                 |
| Australia/Lindeman     | UTC+10:00 | UTC+10:00 | Queensland (Whitsunday Islands) |
| Australia/Lord_Howe    | UTC+10:30 | UTC+11:00 | Lord Howe Island                |
| Australia/Melbourne    | UTC+10:00 | UTC+11:00 | Victoria                        |
| Australia/North        | UTC+09:30 | UTC+09:30 |                                 |
|                        |           |           |                                 |

|                      |           |           |                                |
|----------------------|-----------|-----------|--------------------------------|
| Australia/NSW        | UTC+10:00 | UTC+11:00 |                                |
| Australia/Perth      | UTC+08:00 | UTC+08:00 | Western Australia (most areas) |
| Australia/Queensland | UTC+10:00 | UTC+10:00 |                                |
| Australia/South      | UTC+09:30 | UTC+10:30 |                                |
| Australia/Sydney     | UTC+10:00 | UTC+11:00 | New South Wales (most areas)   |
| Australia/Tasmania   | UTC+10:00 | UTC+11:00 |                                |
| Australia/Victoria   | UTC+10:00 | UTC+11:00 |                                |
| Australia/West       | UTC+08:00 | UTC+08:00 |                                |
| Australia/Yancowinna | UTC+09:30 | UTC+10:30 |                                |
| Brazil/Acre          | UTC05:00  | UTC05:00  |                                |
| Brazil/DeNoronha     | UTC02:00  | UTC02:00  |                                |
| Brazil/East          | UTC03:00  | UTC02:00  |                                |
| Brazil/West          | UTC04:00  | UTC04:00  |                                |
| Canada/Atlantic      | UTC04:00  | UTC03:00  |                                |
| Canada/Central       | UTC06:00  | UTC05:00  |                                |
| Canada/Eastern       | UTC05:00  | UTC04:00  |                                |
| Canada/Mountain      | UTC07:00  | UTC06:00  |                                |
| Canada/Newfoundland  | UTC03:30  | UTC02:30  |                                |
| Canada/Pacific       | UTC08:00  | UTC07:00  |                                |
| Canada/Saskatchewan  | UTC06:00  | UTC06:00  |                                |
| Canada/Yukon         | UTC08:00  | UTC07:00  |                                |
| CET                  | UTC+01:00 | UTC+02:00 |                                |
| Chile/Continental    | UTC04:00  | UTC03:00  |                                |
| Chile/EasterIsland   | UTC06:00  | UTC05:00  |                                |
| CST6CDT              | UTC06:00  | UTC05:00  |                                |
| Cuba                 | UTC05:00  | UTC04:00  |                                |
| EET                  | UTC+02:00 | UTC+03:00 |                                |
| Egypt                | UTC+02:00 | UTC+02:00 |                                |
| Eire                 | UTC+00:00 | UTC+01:00 |                                |
| EST                  | UTC05:00  | UTC05:00  |                                |
| EST5EDT              | UTC05:00  | UTC04:00  |                                |
| Etc/GMT              | UTC+00:00 | UTC+00:00 |                                |
| Etc/GMT+0            | UTC+00:00 | UTC+00:00 |                                |
| Etc/GMT+1            | UTC01:00  | UTC01:00  |                                |
| Etc/GMT+10           | UTC10:00  | UTC10:00  |                                |
| Etc/GMT+11           | UTC11:00  | UTC11:00  |                                |
| Etc/GMT+12           | UTC12:00  | UTC12:00  |                                |
| Etc/GMT+2            | UTC02:00  | UTC02:00  |                                |
| Etc/GMT+3            | UTC03:00  | UTC03:00  |                                |
| Etc/GMT+4            | UTC04:00  | UTC04:00  |                                |
|                      |           |           |                                |

|                   |           |           |                                            |
|-------------------|-----------|-----------|--------------------------------------------|
| Etc/GMT+5         | UTC05:00  | UTC05:00  |                                            |
| Etc/GMT+6         | UTC06:00  | UTC06:00  |                                            |
| Etc/GMT+7         | UTC07:00  | UTC07:00  |                                            |
| Etc/GMT+8         | UTC08:00  | UTC08:00  |                                            |
| Etc/GMT+9         | UTC09:00  | UTC09:00  |                                            |
| Etc/GMT0          | UTC+00:00 | UTC+00:00 |                                            |
| Etc/GMT-0         | UTC+00:00 | UTC+00:00 |                                            |
| Etc/GMT-1         | UTC+01:00 | UTC+01:00 |                                            |
| Etc/GMT-10        | UTC+10:00 | UTC+10:00 |                                            |
| Etc/GMT-11        | UTC+11:00 | UTC+11:00 |                                            |
| Etc/GMT-12        | UTC+12:00 | UTC+12:00 |                                            |
| Etc/GMT-13        | UTC+13:00 | UTC+13:00 |                                            |
| Etc/GMT-14        | UTC+14:00 | UTC+14:00 |                                            |
| Etc/GMT-2         | UTC+02:00 | UTC+02:00 |                                            |
| Etc/GMT-3         | UTC+03:00 | UTC+03:00 |                                            |
| Etc/GMT-4         | UTC+04:00 | UTC+04:00 |                                            |
| Etc/GMT-5         | UTC+05:00 | UTC+05:00 |                                            |
| Etc/GMT-6         | UTC+06:00 | UTC+06:00 |                                            |
| Etc/GMT-7         | UTC+07:00 | UTC+07:00 |                                            |
| Etc/GMT-8         | UTC+08:00 | UTC+08:00 |                                            |
| Etc/GMT-9         | UTC+09:00 | UTC+09:00 |                                            |
| Etc/Greenwich     | UTC+00:00 | UTC+00:00 |                                            |
| Etc/UCT           | UTC+00:00 | UTC+00:00 |                                            |
| Etc/Universal     | UTC+00:00 | UTC+00:00 |                                            |
| Etc/UTC           | UTC+00:00 | UTC+00:00 |                                            |
| Etc/Zulu          | UTC+00:00 | UTC+00:00 |                                            |
| Europe/Amsterdam  | UTC+01:00 | UTC+02:00 |                                            |
| Europe/Andorra    | UTC+01:00 | UTC+02:00 |                                            |
| Europe/Astrakhan  | UTC+04:00 | UTC+04:00 | MSK+01 - Astrakhan                         |
| Europe/Athens     | UTC+02:00 | UTC+03:00 |                                            |
| Europe/Belfast    | UTC+00:00 | UTC+01:00 |                                            |
| Europe/Belgrade   | UTC+01:00 | UTC+02:00 |                                            |
| Europe/Berlin     | UTC+01:00 | UTC+02:00 | Germany (except for Bisingen am Hochrhein) |
| Europe/Bratislava | UTC+01:00 | UTC+02:00 |                                            |
| Europe/Brussels   | UTC+01:00 | UTC+02:00 |                                            |
| Europe/Bucharest  | UTC+02:00 | UTC+03:00 |                                            |
| Europe/Budapest   | UTC+01:00 | UTC+02:00 |                                            |
| Europe/Busingen   | UTC+01:00 | UTC+02:00 | Bisingen am Hochrhein                      |
| Europe/Chisinau   | UTC+02:00 | UTC+03:00 |                                            |
| Europe/Copenhagen | UTC+01:00 | UTC+02:00 |                                            |
|                   |           |           |                                            |



|                    |           |           |                           |
|--------------------|-----------|-----------|---------------------------|
| Europe/Dublin      | UTC+00:00 | UTC+01:00 |                           |
| Europe/Gibraltar   | UTC+01:00 | UTC+02:00 |                           |
| Europe/Guernsey    | UTC+00:00 | UTC+01:00 |                           |
| Europe/Helsinki    | UTC+02:00 | UTC+03:00 |                           |
| Europe/Isle_of_Man | UTC+00:00 | UTC+01:00 |                           |
| Europe/Istanbul    | UTC+03:00 | UTC+03:00 |                           |
| Europe/Jersey      | UTC+00:00 | UTC+01:00 |                           |
| Europe/Kaliningrad | UTC+02:00 | UTC+02:00 | MSK01 - Kaliningrad       |
| Europe/Kiev        | UTC+02:00 | UTC+03:00 | Ukraine (most areas)      |
| Europe/Kirov       | UTC+03:00 | UTC+03:00 | MSK+00 - Kirov            |
| Europe/Lisbon      | UTC+00:00 | UTC+01:00 | Portugal (mainland)       |
| Europe/Ljubljana   | UTC+01:00 | UTC+02:00 |                           |
| Europe/London      | UTC+00:00 | UTC+01:00 |                           |
| Europe/Luxembourg  | UTC+01:00 | UTC+02:00 |                           |
| Europe/Madrid      | UTC+01:00 | UTC+02:00 | Spain (mainland)          |
| Europe/Malta       | UTC+01:00 | UTC+02:00 |                           |
| Europe/Mariehamn   | UTC+02:00 | UTC+03:00 |                           |
| Europe/Minsk       | UTC+03:00 | UTC+03:00 |                           |
| Europe/Monaco      | UTC+01:00 | UTC+02:00 |                           |
| Europe/Moscow      | UTC+03:00 | UTC+03:00 | MSK+00 - Moscow area      |
| Asia/Nicosia       | UTC+02:00 | UTC+03:00 | Cyprus (most areas)       |
| Europe/Oslo        | UTC+01:00 | UTC+02:00 |                           |
| Europe/Paris       | UTC+01:00 | UTC+02:00 |                           |
| Europe/Podgorica   | UTC+01:00 | UTC+02:00 |                           |
| Europe/Prague      | UTC+01:00 | UTC+02:00 |                           |
| Europe/Riga        | UTC+02:00 | UTC+03:00 |                           |
| Europe/Rome        | UTC+01:00 | UTC+02:00 |                           |
| Europe/Samara      | UTC+04:00 | UTC+04:00 | MSK+01 - Samara, Udmurtia |
| Europe/San_Marino  | UTC+01:00 | UTC+02:00 |                           |
| Europe/Sarajevo    | UTC+01:00 | UTC+02:00 |                           |
| Europe/Saratov     | UTC+04:00 | UTC+04:00 | MSK+01 - Saratov          |
| Europe/Simferopol  | UTC+03:00 | UTC+03:00 | Crimea                    |
| Europe/Skopje      | UTC+01:00 | UTC+02:00 |                           |
| Europe/Sofia       | UTC+02:00 | UTC+03:00 |                           |
| Europe/Stockholm   | UTC+01:00 | UTC+02:00 |                           |
| Europe/Tallinn     | UTC+02:00 | UTC+03:00 |                           |
| Europe/Tirane      | UTC+01:00 | UTC+02:00 |                           |
| Europe/Tiraspol    | UTC+02:00 | UTC+03:00 |                           |
| Europe/Ulyanovsk   | UTC+04:00 | UTC+04:00 | MSK+01 - Ulyanovsk        |
| Europe/Uzhgorod    | UTC+02:00 | UTC+03:00 | Ruthenia                  |
|                    |           |           |                           |

|                     |           |           |                                                |
|---------------------|-----------|-----------|------------------------------------------------|
| Europe/Vaduz        | UTC+01:00 | UTC+02:00 |                                                |
| Europe/Vatican      | UTC+01:00 | UTC+02:00 |                                                |
| Europe/Vienna       | UTC+01:00 | UTC+02:00 |                                                |
| Europe/Vilnius      | UTC+02:00 | UTC+03:00 |                                                |
| Europe/Volgograd    | UTC+04:00 | UTC+04:00 | MSK+01 - Volgograd                             |
| Europe/Warsaw       | UTC+01:00 | UTC+02:00 |                                                |
| Europe/Zagreb       | UTC+01:00 | UTC+02:00 |                                                |
| Europe/Zaporozhye   | UTC+02:00 | UTC+03:00 | Zaporozh'ye/Zaporizhia; Lugansk/Luhansk (east) |
| Europe/Zurich       | UTC+01:00 | UTC+02:00 |                                                |
| GB                  | UTC+00:00 | UTC+01:00 |                                                |
| GB-Eire             | UTC+00:00 | UTC+01:00 |                                                |
| GMT                 | UTC+00:00 | UTC+00:00 |                                                |
| GMT+0               | UTC+00:00 | UTC+00:00 |                                                |
| GMT0                | UTC+00:00 | UTC+00:00 |                                                |
| GMT-0               | UTC+00:00 | UTC+00:00 |                                                |
| Greenwich           | UTC+00:00 | UTC+00:00 |                                                |
| Hongkong            | UTC+08:00 | UTC+08:00 |                                                |
| HST                 | UTC10:00  | UTC10:00  |                                                |
| Iceland             | UTC+00:00 | UTC+00:00 |                                                |
| Indian/Antananarivo | UTC+03:00 | UTC+03:00 |                                                |
| Indian/Chagos       | UTC+06:00 | UTC+06:00 |                                                |
| Indian/Christmas    | UTC+07:00 | UTC+07:00 |                                                |
| Indian/Cocos        | UTC+06:30 | UTC+06:30 |                                                |
| Indian/Comoro       | UTC+03:00 | UTC+03:00 |                                                |
| Indian/Kerguelen    | UTC+05:00 | UTC+05:00 |                                                |
| Indian/Mahe         | UTC+04:00 | UTC+04:00 |                                                |
| Indian/Maldives     | UTC+05:00 | UTC+05:00 |                                                |
| Indian/Mauritius    | UTC+04:00 | UTC+04:00 |                                                |
| Indian/Mayotte      | UTC+03:00 | UTC+03:00 |                                                |
| Indian/Reunion      | UTC+04:00 | UTC+04:00 |                                                |
| Iran                | UTC+03:30 | UTC+04:30 |                                                |
| Israel              | UTC+02:00 | UTC+03:00 |                                                |
| Jamaica             | UTC05:00  | UTC05:00  |                                                |
| Japan               | UTC+09:00 | UTC+09:00 |                                                |
| Kwajalein           | UTC+12:00 | UTC+12:00 |                                                |
| Libya               | UTC+02:00 | UTC+02:00 |                                                |
| MET                 | UTC+01:00 | UTC+02:00 |                                                |
| Mexico/BajaNorte    | UTC08:00  | UTC07:00  |                                                |
| Mexico/BajaSur      | UTC07:00  | UTC06:00  |                                                |
| Mexico/General      | UTC06:00  | UTC05:00  |                                                |
|                     |           |           |                                                |

|                      |           |           |                               |
|----------------------|-----------|-----------|-------------------------------|
| MST                  | UTC07:00  | UTC07:00  |                               |
| MST7MDT              | UTC07:00  | UTC06:00  |                               |
| Navajo               | UTC07:00  | UTC06:00  |                               |
| NZ                   | UTC+12:00 | UTC+13:00 |                               |
| NZ-CHAT              | UTC+12:45 | UTC+13:45 |                               |
| Pacific/Apia         | UTC+13:00 | UTC+14:00 |                               |
| Pacific/Auckland     | UTC+12:00 | UTC+13:00 | New Zealand (most areas)      |
| Pacific/Bougainville | UTC+11:00 | UTC+11:00 | Bougainville                  |
| Pacific/Chatham      | UTC+12:45 | UTC+13:45 | Chatham Islands               |
| Pacific/Chuuk        | UTC+10:00 | UTC+10:00 | Chuuk/Truk, Yap               |
| Pacific/Easter       | UTC06:00  | UTC05:00  | Easter Island                 |
| Pacific/Efate        | UTC+11:00 | UTC+11:00 |                               |
| Pacific/Enderbury    | UTC+13:00 | UTC+13:00 | Phoenix Islands               |
| Pacific/Fakaofu      | UTC+13:00 | UTC+13:00 |                               |
| Pacific/Fiji         | UTC+12:00 | UTC+13:00 |                               |
| Pacific/Funafuti     | UTC+12:00 | UTC+12:00 |                               |
| Pacific/Galapagos    | UTC06:00  | UTC06:00  | Galapagos Islands             |
| Pacific/Gambier      | UTC09:00  | UTC09:00  | Gambier Islands               |
| Pacific/Guadacanal   | UTC+11:00 | UTC+11:00 |                               |
| Pacific/Guam         | UTC+10:00 | UTC+10:00 |                               |
| Pacific/Honolulu     | UTC10:00  | UTC10:00  | Hawaii                        |
| Pacific/Johnston     | UTC10:00  | UTC10:00  |                               |
| Pacific/Kiritimati   | UTC+14:00 | UTC+14:00 | Line Islands                  |
| Pacific/Kosrae       | UTC+11:00 | UTC+11:00 | Kosrae                        |
| Pacific/Kwajalein    | UTC+12:00 | UTC+12:00 | Kwajalein                     |
| Pacific/Majuro       | UTC+12:00 | UTC+12:00 | Marshall Islands (most areas) |
| Pacific/Marquesas    | UTC09:30  | UTC09:30  | Marquesas Islands             |
| Pacific/Midway       | UTC11:00  | UTC11:00  | Midway Islands                |
| Pacific/Nauru        | UTC+12:00 | UTC+12:00 |                               |
| Pacific/Niue         | UTC11:00  | UTC11:00  |                               |
| Pacific/Norfolk      | UTC+11:00 | UTC+11:00 |                               |
| Pacific/Noumea       | UTC+11:00 | UTC+11:00 |                               |
| Pacific/Pago_Pago    | UTC11:00  | UTC11:00  |                               |
| Pacific/Palau        | UTC+09:00 | UTC+09:00 |                               |
| Pacific/Pitcairn     | UTC08:00  | UTC08:00  |                               |
| Pacific/Pohnpei      | UTC+11:00 | UTC+11:00 | Pohnpei/Ponape                |
| Pacific/Ponape       | UTC+11:00 | UTC+11:00 |                               |
| Pacific/Port_Moresby | UTC+10:00 | UTC+10:00 | Papua New Guinea (most areas) |
| Pacific/Rarotonga    | UTC10:00  | UTC10:00  |                               |
| Pacific/Saipan       | UTC+10:00 | UTC+10:00 |                               |
|                      |           |           |                               |

|                   |           |           |                 |
|-------------------|-----------|-----------|-----------------|
| Pacific/Samoa     | UTC11:00  | UTC11:00  |                 |
| Pacific/Tahiti    | UTC10:00  | UTC10:00  | Society Islands |
| Pacific/Tarawa    | UTC+12:00 | UTC+12:00 | Gilbert Islands |
| Pacific/Tongatapu | UTC+13:00 | UTC+14:00 |                 |
| Pacific/Truk      | UTC+10:00 | UTC+10:00 |                 |
| Pacific/Wake      | UTC+12:00 | UTC+12:00 | Wake Island     |
| Pacific/Wallis    | UTC+12:00 | UTC+12:00 |                 |
| Pacific/Yap       | UTC+10:00 | UTC+10:00 |                 |
| Poland            | UTC+01:00 | UTC+02:00 |                 |
| Portugal          | UTC+00:00 | UTC+01:00 |                 |
| PRC               | UTC+08:00 | UTC+08:00 |                 |
| PST8PDT           | UTC08:00  | UTC07:00  |                 |
| ROC               | UTC+08:00 | UTC+08:00 |                 |
| ROK               | UTC+09:00 | UTC+09:00 |                 |
| Singapore         | UTC+08:00 | UTC+08:00 |                 |
| Turkey            | UTC+03:00 | UTC+03:00 |                 |
| UCT               | UTC+00:00 | UTC+00:00 |                 |
| Universal         | UTC+00:00 | UTC+00:00 |                 |
| US/Alaska         | UTC09:00  | UTC08:00  |                 |
| US/Aleutian       | UTC10:00  | UTC09:00  |                 |
| US/Arizona        | UTC07:00  | UTC07:00  |                 |
| US/Central        | UTC06:00  | UTC05:00  |                 |
| US/Eastern        | UTC05:00  | UTC04:00  |                 |
| US/East-Indiana   | UTC05:00  | UTC04:00  |                 |
| US/Hawaii         | UTC10:00  | UTC10:00  |                 |
| US/Indiana-Starke | UTC06:00  | UTC05:00  |                 |
| US/Michigan       | UTC05:00  | UTC04:00  |                 |
| US/Mountain       | UTC07:00  | UTC06:00  |                 |
| US/Pacific        | UTC08:00  | UTC07:00  |                 |
| US/Pacific-New    | UTC08:00  | UTC07:00  |                 |
| US/Samoa          | UTC11:00  | UTC11:00  |                 |
| UTC               | UTC+00:00 | UTC+00:00 |                 |
| WET               | UTC+00:00 | UTC+01:00 |                 |
| W-SU              | UTC+03:00 | UTC+03:00 |                 |
| Zulu              | UTC+00:00 | UTC+00:00 |                 |

# Locale Settings

Designer Cloud powered by Trifacta® Enterprise Edition supports a tier-based scheme for applying locale settings.

**NOTE:** Locale settings apply only to the inference and validation of data in the Designer Cloud application . Underlying data is not affected by changing locale.

**NOTE:** After saving changes to your locale, refresh your page. Subsequent executions of the data inference service use the new locale settings.

**NOTE:** When locale is changed, data type validation is affected only on subsequent executions of the data type inference service. If you are using structured datasets, such as schematized JDBC sources, data types may be attached to the datasets that you have already imported. These data types are not affected.

| Tier      | Description                                                                                                                |
|-----------|----------------------------------------------------------------------------------------------------------------------------|
| System    | The default locale for the system is United States.                                                                        |
| Workspace | For all members of the workspace, administrators can be configure the locale setting. See <i>Workspace Settings Page</i> . |
| User      | Individual users can choose their own locale settings. For more information, see <i>User Profile Page</i> .                |

## Supported Locales

The following locales are available:

- United States
- Australia
- Austria
- Belgium
- Bulgaria
- Canada
- Croatia
- Cyprus
- Czech Republic
- Denmark
- Estonia
- Finland
- France
- Germany
- Hungary
- Greece
- Ireland
- Italy
- Japan
- Latvia
- Lithuania
- Luxembourg
- Malta
- New Zealand
- Netherlands
- Poland
- Portugal

- Romania
- Singapore
- Slovakia
- Slovenia
- Spain
- Sweden
- United Kingdom

## Product Areas Affected by Locale

### Datetime

- Data type inference more accurately recognizes Datetime values based on locale settings.
- Suggestion cards containing Datetime transformations and example previews utilize locale settings.
- Number of weeks in the year may vary. For example, the WEEKNUM function calculated on the last few dates of the calendar year may return different values depending on the locale where the calculation is performed:

**NOTE:** The output calculations for number of weeks may vary between browser, Trifacta Photon, and the running environment where the job may be executed, based upon locale.

- **EN-US:** Maximum of 52 weeks
- **ISO 8601:** Maximum of 53 weeks
- You can add steps similar to the following, which adds a 53rd week as a possible output for WEEKNUM:

```
derive type: single value: dateformat(date(year(myDate), 1, 1), 'yyyy\MM\dd') as: 'NewYearsDayforMyYear'
```

```
derive type: single value: if(datedif(NewYearsDayforMyYear, myDate, day) > (52 * 7), 53, weeknum(myDate)) as: 'weekNumforMyDate'
```

- For more information, see *WEEKNUM Function*.

# Supported File Encoding Types

## Supported Global File Encoding Types for Input

- UTF-8 (default)
- IBM00858
- IBM437
- IBM775
- IBM850
- IBM852
- IBM855
- IBM857
- IBM862
- IBM866
- ISO-8859-1
- ISO-8859-2
- ISO-8859-3
- ISO-8859-4
- ISO-8859-5
- ISO-8859-6
- ISO-8859-7
- ISO-8859-8
- ISO-8859-9
- ISO-8859-13
- ISO-8859-15
- KOI8-R
- KOI8-U
- US-ASCII
- UTF-16
- UTF-16BE
- UTF-16LE
- UTF-32
- windows-1250
- windows-1251
- windows-1252
- windows-1253
- windows-1254
- windows-1255
- windows-1256
- windows-1257
- x-IBM737
- x-IBM874
- x-UTF-16LE-BOM

## Supported Global File Encoding Types for Output

Output files are written in UTF-8 encoding.

# Sort Order

## Contents:

- *General Sort Order*
  - *Sort Order for Strings*
  - *Sort Order for Integers and Decimals*
- 

This section describes how values are sorted within Designer Cloud powered by Trifacta® Enterprise Edition. Sorting can be applied through the following mechanisms:

- Clicking a column header in a workspace table, such as the Flows, Library, or Jobs pages.
- Applying a Sort transform. For more information, see *Sort Transform*.
- Applying the ARRAYSORT function to an array. For more information, see *ARRAYSORT Function*.

**NOTE:** Following listings represent sorting in ascending order. Descending order sorting is in the reverse of the listings below.

## General Sort Order

For any column, sorting is performed in the following order:

1. Sorting based on data type
2. Mismatched values
3. Null/empty values

## Sort Order for Strings

Since all values are valid for String data type, this sort order represents the most common representation for sort order.

1. Sorting based on data type:
  - a. Numeric values (low value to high value)
  - b. Whitespace
  - c. Special characters
  - d. Alphabetical
    - i. Case-insensitive
    - ii. Accented characters (ä) are below unaccented character (a)
2. Mismatched values
3. Null/empty values

## Sort Order for Integers and Decimals

For Integers and Decimals, sorting happens in the following order:

1. Sorting based on data type:
  - a. Numeric values (low value to high value)



2. Mismatched values
3. Null/empty values

# Join Types

## Contents:

- *Inner Join*
  - *Left Join*
  - *Right Join*
  - *Full Outer Join*
  - *Cross Join*
  - *Self Join*
  - *Joins Together*
- 

The following types of joins are supported. For example, the following tables contains information about employees and departments.

### Employee table:

| Name          | DepartmentID | Role                      |
|---------------|--------------|---------------------------|
| Dave Smith    | 001          | Product Marketing Manager |
| Julie Jones   | 002          | Software Engineer         |
| Scott Tanner  | 001          | Director of Demand Gen    |
| Ted Connors   | 002          | Software Engineer         |
| Margaret Lane | 001          | VP of Marketing           |
| Mary Martin   | 004          | Receptionist              |

### Department table:

| Name        | DepartmentID |
|-------------|--------------|
| Marketing   | 001          |
| Engineering | 002          |
| Accounting  | 003          |

In the above example, `DepartmentID` is the key to use in both tables for any joins.

## Inner Join

An **inner join** requires that key values exist in both tables for the records to appear in the results table. Records appear in the merge only if there are matches in both tables for the key values.

- If you want to include rows containing non-matching values, you must use some form of an outer join. See below.

For the preceding example tables, an inner join on the `DepartmentID` table produces the following result table:

| Employee.Name | Employee.DepartmentID | Employee.Role             | Department.Name | Department.DepartmentID |
|---------------|-----------------------|---------------------------|-----------------|-------------------------|
| Dave Smith    | 001                   | Product Marketing Manager | Marketing       | 001                     |
| Julie Jones   | 002                   | Software Engineer         | Engineering     | 002                     |
| Scott Tanner  | 001                   | Director of Demand Gen    | Marketing       | 001                     |

|               |     |                   |             |     |
|---------------|-----|-------------------|-------------|-----|
| Ted Connors   | 002 | Software Engineer | Engineering | 002 |
| Margaret Lane | 001 | VP of Marketing   | Marketing   | 001 |

#### Notes:

- All fields are included in the merged result set. Fields from the first dataset are listed first.
- The row for Mary Martin is excluded, since there is no reference in the Department table for her department identifier. The row for Accounting is excluded, since there is no reference in the Employee table for the department identifier.
  - To include these rows, you either need to augment the data or perform a form of an outer join.
- A null value in one table does not match a null value in another table. So, rows with null values in a join key are never included in an inner join. These values should be fixed.

**Tip:** An inner join can be used to eliminate rows with null values in their key fields.

## Left Join

A left join (or left outer join) does not require that there be matching records for each value in the key value of the source (left) table. Each row in the left table appears in the results, regardless of whether there are matches in the right table.

For the preceding example tables, a left join on the `DepartmentID` table produces the following result table:

| Employee.Name | Employee.DepartmentID | Employee.Role             | Department.Name | Department.DepartmentID |
|---------------|-----------------------|---------------------------|-----------------|-------------------------|
| Dave Smith    | 001                   | Product Marketing Manager | Marketing       | 001                     |
| Julie Jones   | 002                   | Software Engineer         | Engineering     | 002                     |
| Scott Tanner  | 001                   | Director of Demand Gen    | Marketing       | 001                     |
| Ted Connors   | 002                   | Software Engineer         | Engineering     | 002                     |
| Margaret Lane | 001                   | VP of Marketing           | Marketing       | 001                     |
| Mary Martin   | 004                   | Receptionist              | NULL            | NULL                    |

#### Notes:

- In this left join, the Mary Martin row has been added to the result, since her record in the Employee table does contain an entry for the `DepartmentID`. However, since there are no corresponding values in the Department table, the corresponding fields in the result table are `NULL` values.

## Right Join

A right join (or right outer join) is the reverse of a left join. A right join does not require that there be matching records for each value in the key value of the secondary (right) table. Each row in the right table appears in the results, regardless of whether there are matches in the left table.

For the preceding example tables, a right join on the `DepartmentID` table produces the following result table:

| Employee.Name | Employee.DepartmentID | Employee.Role             | Department.Name | Department.DepartmentID |
|---------------|-----------------------|---------------------------|-----------------|-------------------------|
| Dave Smith    | 001                   | Product Marketing Manager | Marketing       | 001                     |
| Julie Jones   | 002                   | Software Engineer         | Engineering     | 002                     |
| Scott Tanner  | 001                   | Director of Demand Gen    | Marketing       | 001                     |
| Ted Connors   | 002                   | Software Engineer         | Engineering     | 002                     |
|               |                       |                           |                 |                         |

|               |      |                 |            |     |
|---------------|------|-----------------|------------|-----|
| Margaret Lane | 001  | VP of Marketing | Marketing  | 001 |
| NULL          | NULL | NULL            | Accounting | 003 |

#### Notes:

- In this right join, the Accounting entry is added. However, since there is no entry in the Employee table for the DepartmentID value, those fields are NULL values in the result set.

## Full Outer Join

A **full outer join** combines the effects of a left join and a right join. If there is a match between the key values, a row is written in the result.

- If there is no match for a key value that appears in either table, a single record is written to the result, with NULL values inserted for the fields from the other table.

| Employee.Name | Employee.DepartmentID | Employee.Role             | Department.Name | Department.DepartmentID |
|---------------|-----------------------|---------------------------|-----------------|-------------------------|
| Dave Smith    | 001                   | Product Marketing Manager | Marketing       | 001                     |
| Julie Jones   | 002                   | Software Engineer         | Engineering     | 002                     |
| Scott Tanner  | 001                   | Director of Demand Gen    | Marketing       | 001                     |
| Ted Connors   | 002                   | Software Engineer         | Engineering     | 002                     |
| Margaret Lane | 001                   | VP of Marketing           | Marketing       | 001                     |
| Mary Martin   | 004                   | Receptionist              | NULL            | NULL                    |
| NULL          | NULL                  | NULL                      | Accounting      | 003                     |

#### Notes:

- Any duplicated rows between joining from left-to-right and from right-to-left are removed from the results.

## Cross Join

A **cross join** combines each row of the first data set with each row of the second dataset, where every combination is represented in the output. As a result, the number of total rows in the join are:

```
Rows(DatasetA) * Rows(DatasetB)
```

**NOTE:** Depending on the size of your datasets, a cross join can greatly expand the size of the output, which may increase costs in some environments.

## Self Join

A **self join** is a join operation between a dataset and a copy of itself. For example, you can use a self-join to invert the structure of hierarchical data, such as brand-product or manager-employee.

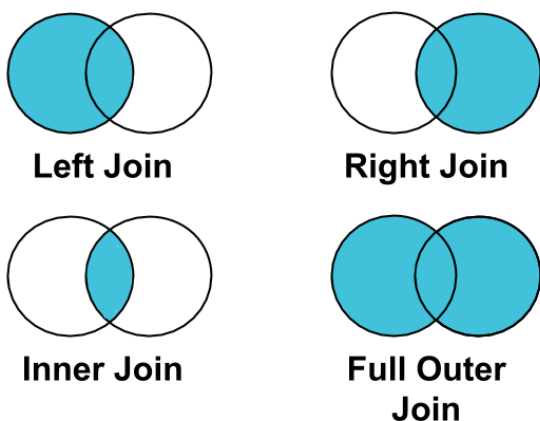
Designer Cloud powered by Trifacta Enterprise Edition supports joins between a recipe and any upstream recipe or dataset.

- You cannot join a recipe to itself.
- You can join it to its source imported dataset. When a self-join is performed with a recipe connected to its source dataset, only one line connects the imported dataset with the recipe in Flow View. This is as designed.

- You can join a recipe to any recipe upstream of it. Examples:
  - You can create an empty recipe after the recipe from which you wish to self-join. In this new empty recipe, you add the join step back to the original recipe.
  - You can insert an empty recipe between an imported dataset and the recipe where the self-join is performed. When you perform the self join in the first recipe, you join to the empty recipe you just created in between.
  - In both examples, you can see multiple lines in Flow View to indicate the self-join. See *Flow View Page*.

## Joins Together

The following diagram summarizes the relationships between the types of supported joins. In each venn diagram, the area of intersection is the set of records that contain shared key values.



**Figure: Join Types**

# Join Metrics

When you create a join, Designer Cloud powered by Trifacta® Enterprise Edition attempts to match up columns as the keys in your join. For each set of join keys, you can review the following metrics related to the join.

## Match percentage:

When you hover over the percentage of matches between key values, you can see the details that make up the calculation:

| Metric      | Description                                                                                                                             |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| All Rows    | Total count of rows in the dataset                                                                                                      |
| Matches     | Total count of values in the join key of the selected column with matching values in the join key of the other dataset.                 |
| Non-Matches | Total count of values in the join key of the selected column with values that do not have a match in the join key of the other dataset. |

The percentage is calculated by summing the count of matches for both datasets and dividing that by the total count of rows across both datasets:

```
(Matches_Current_Dataset + Matches_Joined-in-Dataset) /
(All_Rows_Current_Dataset + All_Rows_Joined-in-Dataset)
```

## Rows in output:

When you hover over the Rows in Output metric, you can see the following values:

| Metric   | Description                                                        |
|----------|--------------------------------------------------------------------|
| All Rows | Total count of rows in each dataset.                               |
| Included | Count of rows from each dataset that are included in the output.   |
| Excluded | Count of rows from each dataset that are excluded from the output. |

# Supported SQL Syntax

## Contents:

- *Basic Syntax*
    - *Supported syntax by datastore*
  - *General Examples*
    - *Column aliasing*
    - *Collect whole table*
    - *Filter columns*
    - *Filter rows*
    - *Multi-line statement for imported datasets*
- 

This section provides general information on how the Designer Cloud powered by Trifacta Enterprise Edition uses SQL to interact with your databases, including syntax requirements and examples.

## Basic Syntax

Your SQL statements must be valid for the syntax expected by the target relational system. In particular, object delimiters may vary between systems.

**NOTE:** The proper syntax depends on your database system. Please consult the documentation for your product for details.

**Tip:** Although some relational systems do not require object delimiters around column names, it is recommended that you add them to all applicable objects.

**Tip:** Avoid using column type identifiers (e.g. `int`) and other SQL keywords as object names. Some systems may generate invalid SQL errors.

**NOTE:** In the following sections, Oracle syntax is used in the examples. Please modify the examples for your target system.

## Supported syntax by datastore

For more information on the supported syntax for your datastore, please see the documentation for the connection type. See *Connection Types*.

## General Examples

Here are some basic SQL examples to get started.

## Column aliasing

If your select statement results in multiple columns with same name, the query fails to validate or fails on execution, such as selecting all columns in a JOIN. In these cases, columns must be properly aliased.

**NOTE:** This error will be caught either during validating or during dataset import.

For example, in the following JOIN, the EMPLOYEE and DEPARTMENT tables have column names department\_id and department\_id.

```
SELECT * FROM EMPLOYEE INNER JOIN DEPARTMENT ON (department_id = department_id);
```

The above query generates an error. Columns must be properly aliased, as in the following:

```
SELECT e.id, e.department_id, e.first_name, e.last_name, d.department_name FROM EMPLOYEE AS E INNER JOIN
DEPARTMENT d ON (e.department_id = d.department_id);
```

## Collect whole table

```
SELECT * FROM "DB1"."table2";
```

## Filter columns

```
SELECT lastName,firstName FROM "DB1"."table2";
```

## Filter rows

```
SELECT lastName,firstName FROM "DB1"."table2" WHERE invoiceAmt > 10000;
```

## Multi-line statement for imported datasets

The following example uses a multi-line SQL sequence to import a dataset:

**NOTE:** Multi-line SQL support is considered an advanced use case. This feature must be enabled.

The following example inserts values in the TABLE\_INVENTORY table and then queries the table. It utilizes Oracle syntax:

```
INSERT INTO "SALES"."TABLE_INVENTORY" ("ID", "AVAILABILITY") VALUES (1, 10);
SELECT * FROM "SALES"."TABLE_INVENTORY";
```



# Sample Types

## Contents:

- *Initial Data Samples*
  - *First Rows Samples*
  - *Random Samples*
  - *Filter-Based Samples*
  - *Anomaly-Based Samples*
  - *Stratified Samples*
  - *Cluster-Based Samples*
- 

This section provides an overview of the types of samples that Designer Cloud powered by Trifacta® Enterprise Edition can generate.

## Sample filters:

Several sampling types support the application of filters to the source data. In this case, a **filter** can be defined to limit the scope of rows that are used to generate the sample. For example, suppose you apply a filter like the following:

```
orderId == '100'
```

The rows of data available for generating the sample are reduced to include only the rows where the value of the `orderId` column is 100.

**Tip:** Sample filters are very useful for allowing you to generate samples that are much more specific to the steps that are trying to build at the present time in your recipe.

## Scan method:

Depending on the type of sample, you may be able to select the method by which the data is scanned:

- **Quick Scan:** Representative sample is scanned and executed in-memory on the Trifacta node. Although the scope of the scanned data is smaller, these samples are much faster to generate.
  - If a Quick Scan sample fails, the Designer Cloud application may attempt to perform the scan on an available clustered running environment.
- **Full Scan:** Data is sampled from the full set of available data. The sampling job is executed on an available clustered running environment. These sampling jobs can take longer to execute. Depending on your environment, additional costs may be incurred.

## Initial Data Samples

These samples are collected automatically when you first load a new dataset into the Transformer page. These sample contain the first 10 MB of data from the first file or table in the dataset.

**Tip:** In the Transformer page, these samples are labeled as **Initial Data**.

**Tip:** If the recipe is a child recipe, the Initial Data sample indicates the selected sample of the parent recipe.

## First Rows Samples

**NOTE:** The First rows sampling technique requires the Trifacta Photon running environment. If the running environment is disabled, please set `feature.enableFirstRowsSample` to `false`.

This sample is taken from the first set of rows in the imported dataset based on the current cursor location in the recipe. The first N rows in the dataset are collected based on the recipe steps up to the configured sample size.

- This sample may span multiple datasets and files, depending on how the recipe is constructed.
- The first rows sample is different from the initial sample, which is gathered without reference to any recipe steps.

These samples are fast to generate. These samples may load faster in the application than samples of other types.

**Tip:** If you have chained together multiple recipes, all steps in all linked recipes must be run to provide visual updates. If you are experiencing performance problems related to this kind of updating, you can select a recipe in the middle of the chain of recipes and switch it off the initial sample to a different sample. When invoked, the recipes from the preceding datasets do not need to be executed, which can improve performance.

## Random Samples

Random selection of a subset of rows in the dataset. These samples are comparatively fast to generate. You can apply quick scan or full scan to determine the scope of the sample.

## Filter-Based Samples

Find specific values in one or more columns. From the rows that have matching set of values, a random sample is generated.

You must define your filter in the Filter textbox.

## Anomaly-Based Samples

Find mismatched or missing data or both in one or more columns.

You specify one or more columns and whether the anomaly is:

1. mismatched
2. missing
3. either of the above

Optionally, you can define an additional filter on any column.

## Stratified Samples

Find all unique values within a column and create a sample that contains the unique values, up to the sample size limit. The distribution of the column values in the sample reflects the distribution of the column values in the dataset. Sampled values are sorted by frequency, relative to the specified column.

Optionally, you can apply a filter to this one.

**Tip:** Collecting samples containing all unique values can be useful if you are performing mapping transformations, such as values to columns. If your mapping contains too many unique values among

your key-value pairs, you can try to delete all columns except the one containing key-value pairs in a step, collect the sample, add the mapping step, and then delete the step where all other columns are removed.

## Cluster-Based Samples

Cluster sampling collects contiguous rows in the dataset that correspond to a random selection from the unique values in a column. All rows corresponding to the selected unique values appear in the sample, up to the maximum sample size. This sampling is useful for time-series analysis and advanced aggregations.

Optionally, you can apply an advanced filter to the column.

# Support Bundle Contents

## Contents:

- *Job Logs*
    - *cdf script*
    - *job.log*
    - *photon flags file*
    - *photon cli info file*
    - *photon cli log info file*
  - *Support Bundle*
    - *conf files folder*
    - *configuration service folder*
    - *service logs folder*
    - *Process files*
    - *ulimit.txt*
    - *version.txt*
    - *Binary files*
- 

When you download job log files, the following contents may be included in the exported support bundle.

## Example ZIP contents:

- 75 - job log folder
- 76 - job log folder
- support-bundle - support bundle folder

**Tip:** If the support bundle contents fails to generate, please review `log-bundle-creation-errors.txt` for details, which is located inside the `support-bundle` folder.

## Job Logs

When you execute a job, it is broken down into individual for each phase of the process.

**NOTE:** The job log files downloaded from the Designer Cloud application may contain unnecessary messages from other executed jobs. In some cases, it may not be possible to filter out these messages.

There may be separate folders for each of the following processes:

- **Ingest:** Data is ready into the platform from the sources.
- **Convert:** Some imported datasets must be converted from their source format to a format that is natively readable by the product. Typically, these jobs convert binary files into CSVs for use.

**NOTE:** The files generated during Convert jobs are retained only for the duration of job execution, after which they are purged.

- **Transform:** All recipes in the job are executed at scale against datasources.
- **Profile:** Results of the transformation are profiled, if profiling has been enabled for the job.
- **Publish:** Results are written to the specified output locations and formats.

## **cdf script**

This file contains the script that is passed to the running environment to transform your dataset.

**NOTE:** This script is in a compiled language that is passed to the running environment for job execution. It is not in Wrangle and is not intended for user consumption.

### **Example filenames(s):**

```
cdf-script-7932315341561207019.py
```

## **job.log**

Log file for the specific job.

### **Example filenames(s):**

```
job.log
```

## **photon flags file**

This file identifies the settings used by the Photon running environment during job execution.

### **Example filenames(s):**

```
photon-3840706514533636937.flags
```

## **photon cli info file**

Location and format of the log files for the instance of Photon where the job was executed.

### **Example filenames(s):**

```
photon-cli.bin.INFO
```

## **photon cli log info file**

Location and format of the log files for the instance of Photon where the job was executed.

### **Example filenames(s):**

```
photon-cli.bin.ip-10-0-0-9.us-west-2.compute.internal.trifacta.log.INFO.20191101-234021.16850
```

## Support Bundle

When support bundling is enabled, the following folders and files are included in the download. Support bundling is enabled by default.

**NOTE:** Log files that are configured for JSON output format cannot be included in the support bundle.

**NOTE:** You can disable or configure the contents of the support bundle. For more information, see *Configure Support Bundling*.

### conf files folder

Current version and archived versions of `trifacta-conf.json`, the core platform configuration file. If the platform is connected to a Hadoop cluster, additional cluster configuration files are retrieved from the local Trifacta node and included in the support bundle.

#### Example filenames(s):

```
archives/trifacta-conf.json_2019-11-01T073409Z
archives/trifacta-conf.json_20191024-000_2019-11-01T073419.404Z
archives/trifacta-conf.json_20191024-000_2019-11-01T073435.390Z
```

### configuration service folder

Configuration files extracted from the configuration service, which governs configuration at the system, workspace, and user level.

#### Example filenames(s):

```
actual-settings.json
default-settings.json
system-overrides.json
user-overrides.json
workspace-overrides.json
workspace-tier-overrides.json
```

### service logs folder

Log files for services of Designer Cloud powered by Trifacta Enterprise Edition.

#### Example filenames(s):

```
artifact-storage-service.log
batch-job-runner.log
configuration-service.log
orchestration-service.log
```

**Tip:** If you are experiencing issues with the execution of plans, please review `orchestration-service.log` for messages.

**Tip:** Your downloaded bundle may also include access logs for the services.

For more information on these log files, see *System Services and Logs*.

## Process files

The following files include information on the processes that are created and used by the Designer Cloud powered by Trifacta platform :

### Example filenames(s):

- `process-info.json` - list of processes that are run under the Trifacta user

## ulimit.txt

Output of executing the `ulimit` operating system command. `ulimit` returns the values for key operating system parameters.

### Example filenames(s):

```
ulimit.txt
```

## version.txt

Build number of the software from which this bundle was downloaded.

### Example filenames(s):

```
version.txt
```

## Binary files

The following files are stored in binary format and not intended for customer consumption.

### Example filename(s):

```
iv.data
key.data.enc
support-bundle.zip.enc
```



Copyright © 2022 - Trifacta, Inc.  
All rights reserved.