alteryx | TRIFACTA

User Guide

Version: 9.7

Doc Build Date: 12/31/2022

Basics

Contents:

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 tasks 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.

- **Example data:** You should use a sample set of data during this task.

Basic Task

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 *Application Asset Overview*.

Application Asset Overview

Contents:

Explore the assets that you create and their relationships. Flows, imported datasets, and recipes are created to transform your sampled data. After you build your output object, 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 Assets

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

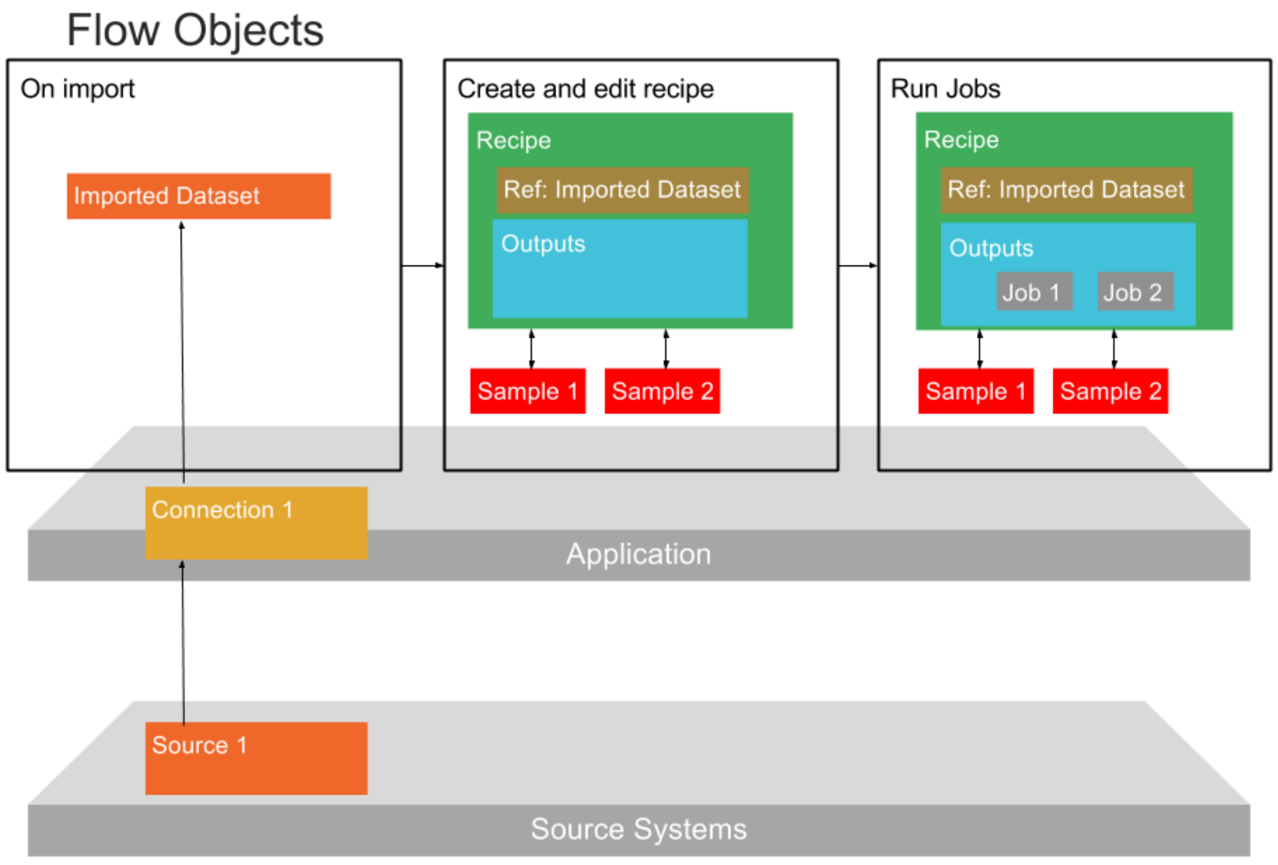


Figure: Assets in a Flow

Flow

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

- Creating relationships between datasets, their recipes, and other datasets.

- Copying
- Execution of pre-configured jobs
- Creating references between recipes and external flows

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 Design er 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 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 asset types 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 objects 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.

Reference Datasets

When you select a recipe's reference, you can add it to another flow. This asset 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 relationships between assets within a flow.

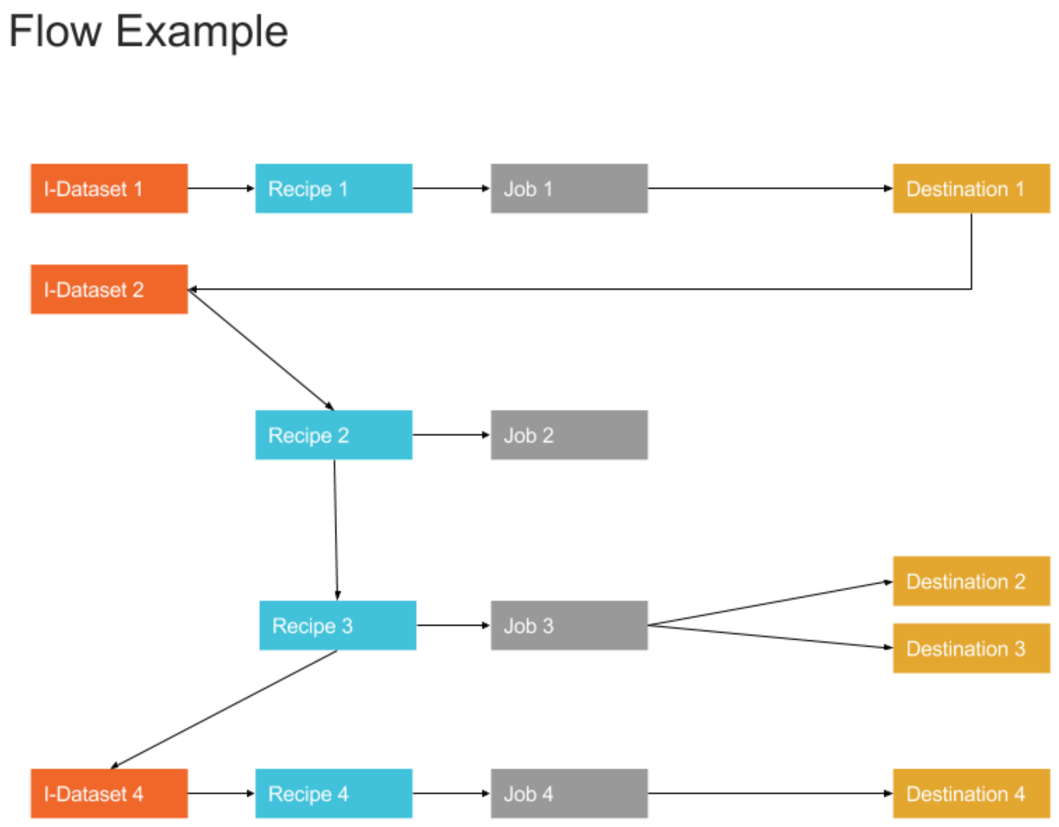


Figure: Flow Example

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) from the Job Details page.
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.

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 *Sampling Basics*.

Macros: As needed, you can create reusable sequences of steps that can be parameterized for use in other recipes.

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 in the Job Details page.
- For more information on the process, see *Running Job Basics*.

Connections

Feature Availability: This feature may not be available in all product editions. For more information on available features, see *Compare Editions*.

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.
- Connections are created in the Connections page.

Flow Schedules

Feature Availability: This feature may not be available in all product editions. For more information on available features, see *Compare Editions*.

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 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
```

Schedules are created for a flow through Flow View page.

Plans

Feature Availability: This feature may not be available in all product editions. For more information on available features, see *Compare Editions*.

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 assets 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 assets 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.

Plans are created through the Plans page.

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.

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 a sample of the imported data.

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

1. Login to the application.
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*.

Flow Basics

Contents:

In Designer Cloud Powered by Trifacta® Enterprise Edition, a flow is a container for holding the objects you create for your end-to-end data pipeline.

With flows, you can perform the following types of actions:

- Combine datasets by performing union or join functions.
- Share the flows with other users.
- Create output definitions and generate job results for them.
- Schedule the execution of jobs.
- Create references between recipes and external flows.
- Export and import into different instances of Designer Cloud Powered by Trifacta Enterprise Edition platform.

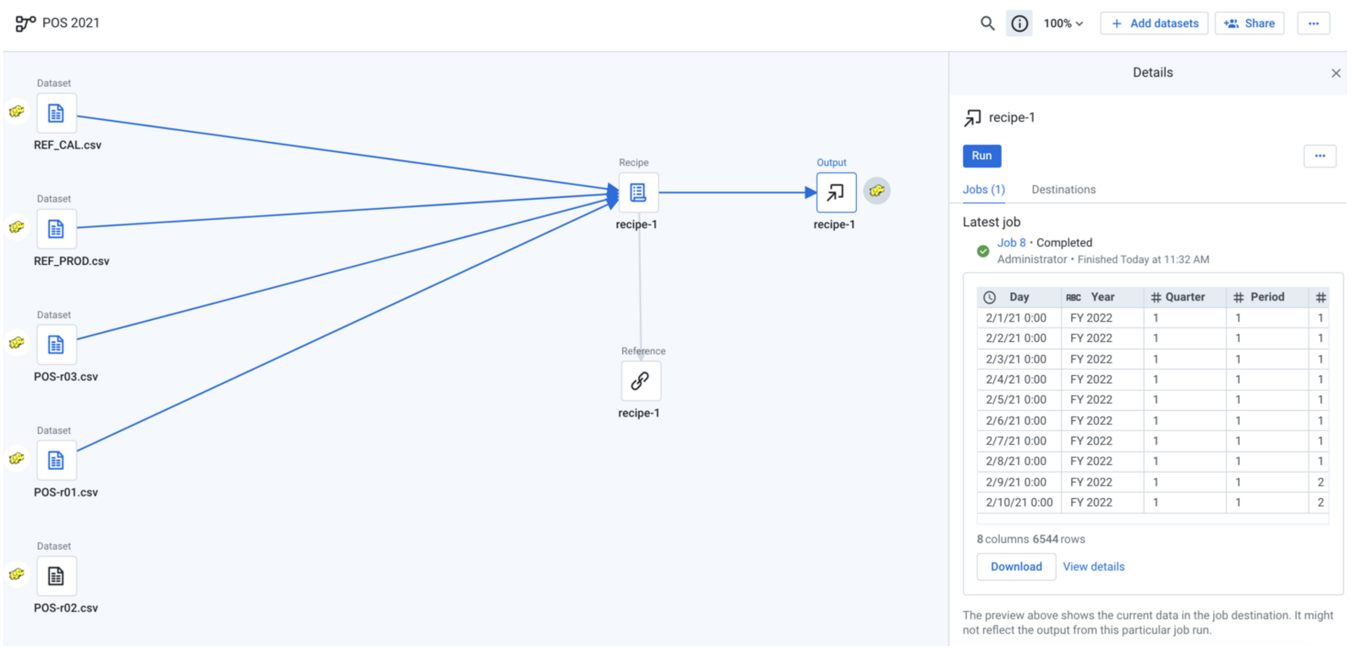
To create a flow:

1. You can get started with flows by clicking the **Flows** icon in the left nav bar.
2. The Flows page displays the flows to which you have access and enables you to create, review, import flows.
3. To create a new flow, click **Create** in the Flows page. An empty flow is displayed in Flow View.

For more information, see *Flow View Page*.

Flow View

In Flow View, you create the objects that are part of your flow.



Latest job

Job 8 • Completed
Administrator • Finished Today at 11:32 AM

Day	Year	Quarter	Period	
2/1/21 0:00	FY 2022	1	1	1
2/2/21 0:00	FY 2022	1	1	1
2/3/21 0:00	FY 2022	1	1	1
2/4/21 0:00	FY 2022	1	1	1
2/5/21 0:00	FY 2022	1	1	1
2/6/21 0:00	FY 2022	1	1	1
2/7/21 0:00	FY 2022	1	1	1
2/8/21 0:00	FY 2022	1	1	1
2/9/21 0:00	FY 2022	1	1	2
2/10/21 0:00	FY 2022	1	1	2

8 columns 6544 rows

The preview above shows the current data in the job destination. It might not reflect the output from this particular job run.

Figure: Flow View page

Top bar

From the top bar of Flow View, you can access a variety of options to perform the following functions:

- Share your flow
- Add datasets to your flow
- Zoom in and out from the canvas
- Search your flow for objects

Flow canvas

The primary workspace of Flow View is the flow canvas. From the flow canvas, you can add new objects and organize them:

- Add a recipe for an imported dataset
    - Add a reference dataset for your recipe
- Add an output for a recipe
- Add more imported datasets
- Auto arrange object icons
- Add a note to the canvas
- Zoom the canvas

Context panel

When you select an object in the flow, you can find additional options in the right panel.

> **NOTE**: The available options depend on the type of object that you selected.

Flow Objects

A flow can contain the following types of objects.

Required flow objects

To create an end-to-end data pipeline, you must create the following objects:

1. **Imported dataset:** A reference to a file- or table-based source of data.
2. **Recipe:** A set of steps to transform your data.
3. **Output:** A definition for the type, location, and format of the results generated by your recipe, plus additional settings.

Imported datasets

An **imported dataset** is reference to a dataset that can be imported into **Designer Cloud Powered by Trifacta Enterprise Edition** . You can import datasets from a variety of flat-file formats and other distributed sources. An imported dataset can be a reference to a file, multiple files, database table, or other types of data.

> **NOTE:** When you import a dataset, you are creating a reference to the source dataset. The source data is never modified.

For more information, see *Import Basics*.

Recipes

A **recipe** is a series of user-defined data transformation steps that can be applied to your data. In Flow View, recipes can be created from other objects:

- You can create a recipe from an imported dataset. The steps in this recipe are applied to the data referenced in the imported dataset.
- You can create a recipe from another recipe to join recipes together.

When you create a recipe in Flow View, an empty recipe object is created for you. You can edit the recipe to add or modify steps in it.

For more information, see *Transformer Page*.

Reference Datasets

A **reference dataset** is a reference to the output of a recipe, which can be used in another flow. When you modify the source recipe, the reference dataset that is used in another flow is automatically updated with the updated data.

To create a reference dataset, select the recipe icon in Flow View. Then, click the **+** icon and select **Create reference dataset**.

A reference dataset is created. When you import data into another flow, you can select the reference dataset for import.

For more information, see *View for Reference Datasets*.

Outputs

An output defines the set of files or tables, formats, and locations where the results of a recipe's transformations are written after a job run has been completed. Every flow requires an output in order to publish results.

An **output** object is composed of settings related to how the job results are generated, including one or more publishing actions. A **publishing action** defines how the results are written, including:

- output location

- output format
- format-specific settings

> **Tip:** You can apply parameters to your output locations.

You can create publishing actions in multiple formats for file-or table-based outputs and publish them to different locations. For more information, see *Flow View Page*.

Search Flows

You can search for specific objects in your flow, which helps to filter complex flows. When you start typing a search term, search results are highlighted in yellow, both in the Flow View canvas and in the search panel.

Steps:

1. From Flow View, click the Search icon.
2. In the Search panel, enter a search term. When you start typing, matching results are highlighted in yellow in the canvas and in the search panel.
3. By default, all object types are searched. You can filter the search results by object type. From the **Filter by type** drop-down, select one or more of the following:
   a. Datasets
   b. Recipes
   c. Outputs
   d. References
4. The search results are displayed.

Flow Sharing

You can share your flows with other users in your project or workspace. Click the **Share** button to share the flow with other users.

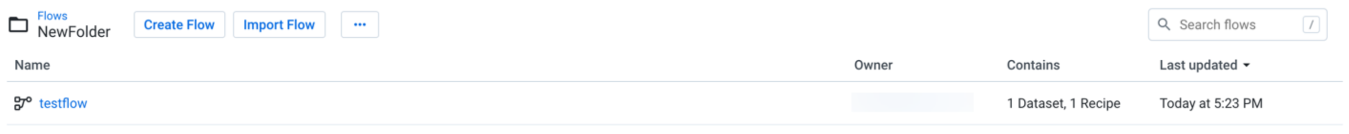
> **NOTE:** When a flow is shared, the receiving users must have access to the flow. Flow access is based on varying levels of authorization to shared flows.

In the Owner column of the Flows page, a shared icon indicates that the flow has been shared.

For more information, see *Share a Flow*.

Organize Flows into Folders

You can organize your flows into folders. A **folder** is a container for storing your flows. From the Flows page, click **Create > Create Folder** to create a folder.



Name	Owner	Contains	Last updated
testflow		1 Dataset, 1 Recipe	Today at 5:23 PM

Figure: Flows in a folder

After creating a folder, you can use the **Move** option in the Flows Page or Flow View to move your flows into a folder.

Profiling Basics

Contents:

Designer Cloud Powered by Trifacta® Enterprise Edition surfaces visual representations of your data for individual columns and the entire dataset and provides mechanisms for taking immediate action on issues in the data.

Profiling Source Data

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 Job Details page in the Designer Cloud application . This profile of your source can be useful later in seeing how your dataset has changed during development.

Profiling in the Application

When you identify something of interest in the Designer Cloud application , you can select the visual representation of it, and the platform prompts you with a set of suggested transforms to add to your recipe. These visual profiles enable you to make quick assessments of problems, unusual patterns, and required changes to your 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.

Status Bar

The number of rows, columns, and data types in the current sample are displayed at the bottom of the page in the 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.

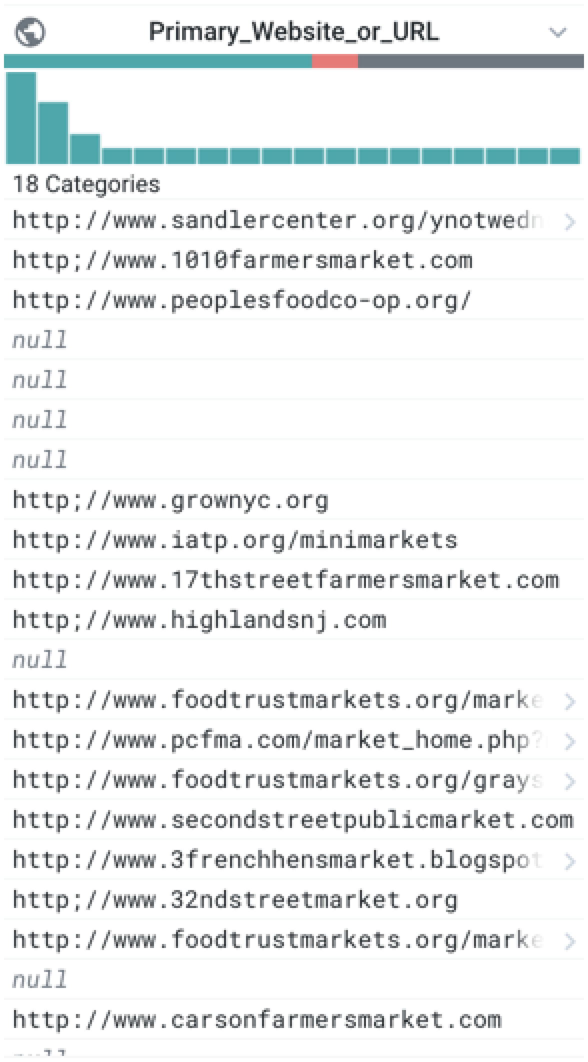

```
Primary_Website_or_URL
18 Categories
http://www.sandlercenter.org/ynotwed
http;//www.1010farmersmarket.com
http://www.peoplesfoodco-op.org/
null
null
null
null
http;//www.grownyc.org
http://www.iatp.org/minimarkets
http://www.17thstreetfarmersmarket.com
http;//www.highlandsnj.com
null
http://www.foodtrustmarkets.org/mark
http://www.pcfma.com/market_home.php
http://www.foodtrustmarkets.org/gray
http://www.secondstreetpublicmarket.com
http://www.3frenchhensmarket.blogspo
http;//www.32ndstreetmarket.org
http://www.foodtrustmarkets.org/mark
null
http://www.carsonfarmersmarket.com
```

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. 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

The bar chart at the top of each column in the Transformer page is called a histogram. 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 can be corrected or removed.

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. Select Column Details from the drop-down for the specific column in the data grid.

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.

Profiling in Job Results

When you run a job, you can choose to generate a visual profile based on the job results from the Run Job page.

Download visual profile

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

Transform Basics

Contents:

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. You can import this target schema as a dataset and use it during recipe development to serve as a mapping for your transformations.

A Quick Tour

The Transformer page experience is based on the design principles of predictive transformation. **Predictive transformation** allows you as the user to explore your data and guides you to possible next steps based on your explorations. Predictive transformation works in the following phases:

1. **Visualize**: Display your sampled data in tabular format with visualizations to indicate areas of interest, such as missing or invalid data or outlier values.
2. **Interact:** You can directly interact with the data to select things of interest to you. For example, you can click a black bar in a column visualization to select all of the missing values in the column.

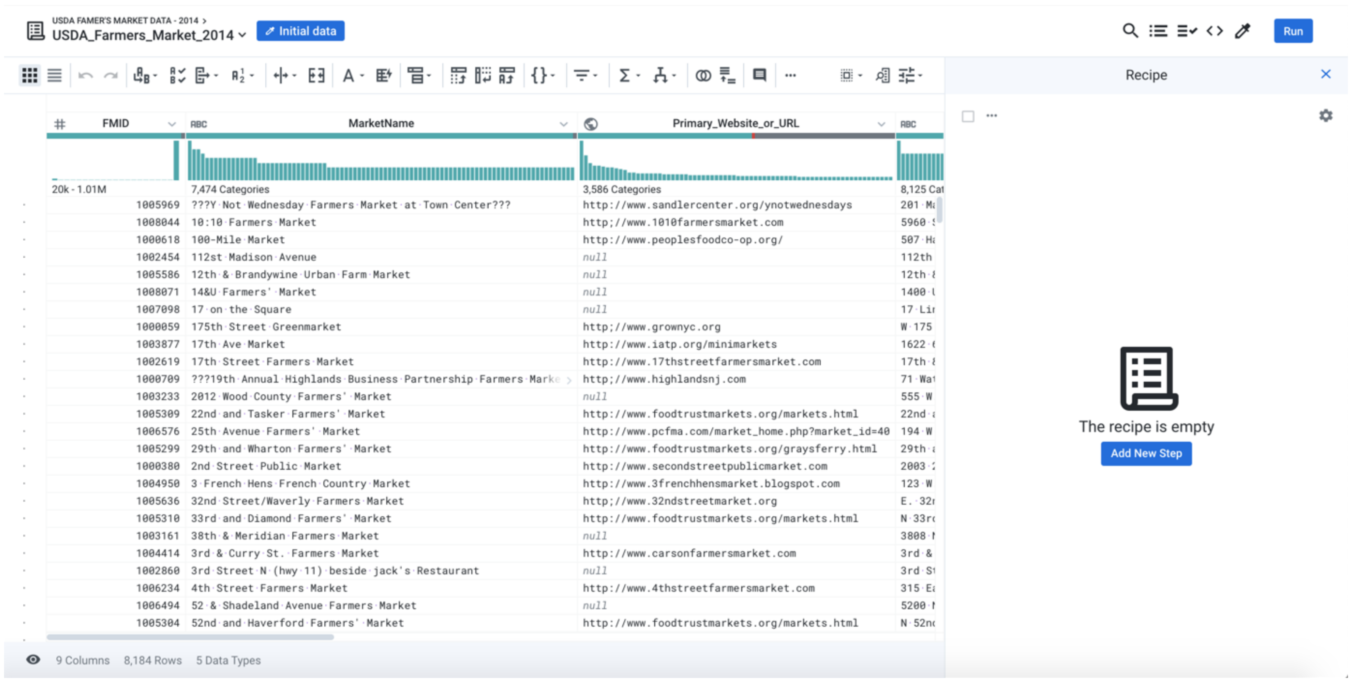
3. **Predict:** Based on your selection(s), the predictive transformation model offers a set of suggested transformations. You can select one from the list. A suggestion may have multiple variations for further refinement. If you want to take it a step further, you can edit the suggested transformation directly to fine-tune the step.
4. **Present:** Each time you make a selection in the data grid or among the suggestion cards, the effects of the selected transformation are previewed in the data grid.

If all looks good, you add the transformation. The change is added to your recipe, which is the sequence of steps to transform your data.

Then, you repeat the above steps for the next data of interest. You can see this process in action below.

1 - Visualize

Below, you can see the Transformer page displaying a sample from the USDA-recognized Farmer's Markets in the United States as of the year 2014.



FMID	MarketName	Primary_Website_or_URL	
20k - 1.01M	7,474 Categories	3,586 Categories	8,125 Cat
1005969	???Y Not Wednesday Farmers Market at Town Center???	http://www.sandlercenter.org/ynotwednesdays	201 Ma
1008044	10:10 Farmers Market	http;//www.1010farmersmarket.com	5960
1000618	100-Mile Market	http://www.peoplesfoodco-op.org/	507 Ha
1002454	112st Madison Avenue	null	112th
1005586	12th & Brandywine Urban Farm Market	null	12th
1008071	14&U Farmers' Market	null	1400
1007098	17 on the Square	null	17 Lir
1000059	175th Street Greenmarket	http;//www.grownyc.org	W 175
1003877	17th Ave Market	http://www.iatp.org/minimarkets	1622
1002619	17th Street Farmers Market	http://www.17thstreetfarmersmarket.com	17th
1000709	???19th Annual Highlands Business Partnership Farmers Marke	http;//www.highlandsnj.com	71 Wat
1003233	2012 Wood County Farmers' Market	null	555 W
1005309	22nd and Tasker Farmers' Market	http://www.foodtrustmarkets.org/markets.html	22nd
1006576	25th Avenue Farmers' Market	http://www.pcfma.com/market_home.php?market_id=40	194 W
1005299	29th and Wharton Farmers' Market	http://www.foodtrustmarkets.org/graysferry.html	29th
1000380	2nd Street Public Market	http://www.secondstreetpublicmarket.com	2003
1004950	3 French Hens French Country Market	http://www.3frenchhensmarket.blogspot.com	123 W
1005636	32nd Street/Waverly Farmers Market	http;//www.32ndstreetmarket.org	E. 32r
1005310	33rd and Diamond Farmers' Market	http://www.foodtrustmarkets.org/markets.html	N 33rc
1003161	38th & Meridian Farmers Market	null	3808
1004414	3rd & Curry St. Farmers Market	http://www.carsonfarmersmarket.com	3rd &
1002860	3rd Street N (hwy 11) beside jack's Restaurant	null	3rd St
1006234	4th Street Farmers Market	http://www.4thstreetfarmersmarket.com	315 Ea
1006494	52 & Shadeland Avenue Farmers Market	null	5200
1005304	52nd and Haverford Farmers' Market	http://www.foodtrustmarkets.org/markets.html	N 52nc

9 Columns 8,184 Rows 5 Data Types

Figure: Visualize: Farmer's Market Data

The source of this data is an Avro file, which in its raw form does not look like this nice tabular representation. When a dataset is loaded into the Transformer page, the Designer Cloud application attempts to represent it in tabular form, so that it is easy to navigate and analyze.

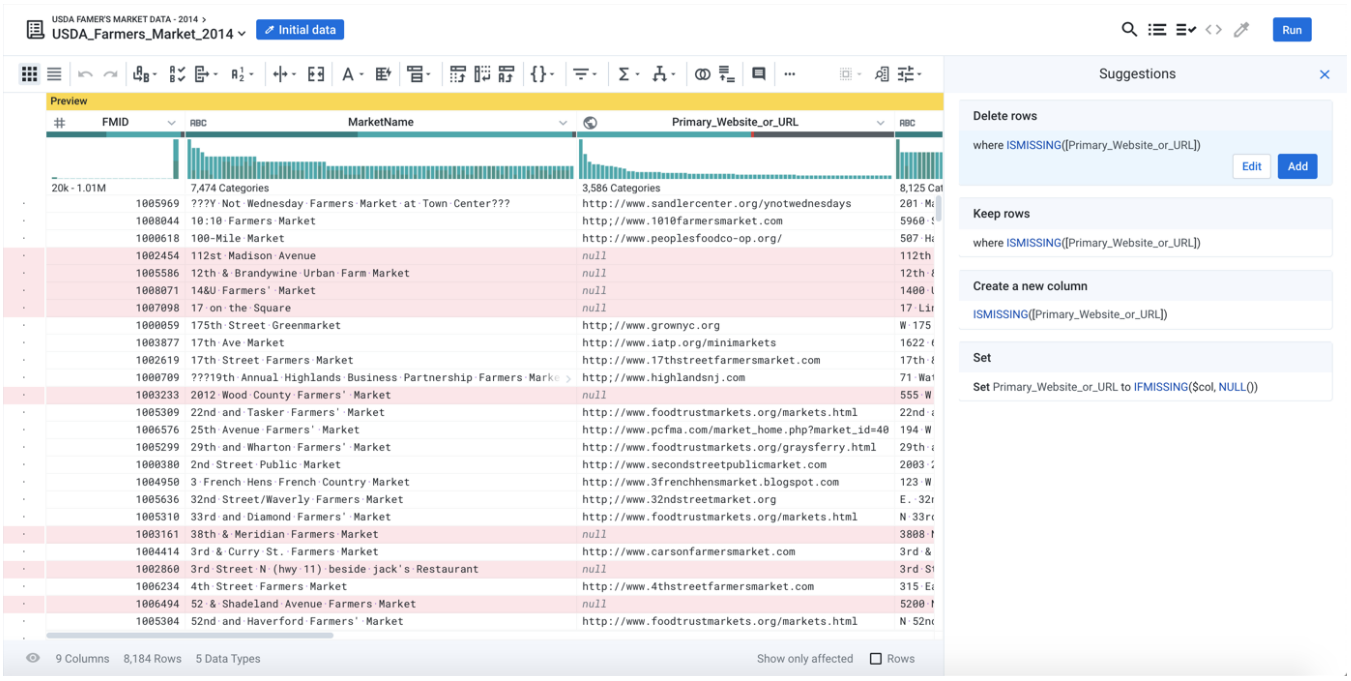
A few key things to notice:

1. **Initial Data:** This button indicates the current sample. Unless your dataset is small, the Transformer page displays in its **data grid** a sample taken from the first set of rows in the first file in your dataset. Be sure to read the section on Sampling below.
   a. The Initial Data button is part of the **menu bar**, which provide access to key application features and capabilities. The menu bar is described later.
   b. Below the menu bar is the **toolbar**, which contains a set of tools that you can select to apply to your data. The toolbar is described later.
2. **The recipe is empty:** This area is the recipe panel, where you can see the steps that you have added to your recipe. A **recipe** is the set of steps that you create to transform your data.
3. **Primary_Website_or_URL:** Above the data in each column you can see two bars.
   a. **Histogram:** The lower bar is a histogram of values in the column.

b. **Data quality bar:** Above the histogram is the data quality bar, which displays the comparative percentage of valid (green), invalid (red), and missing (black) values in a column, as compared to the data type of the column.

2 - Interact

The data grid is not just for display; you can interact with it to select data elements of interest to you. For example, in the Farmer's Market dataset, you may have noticed that the `Primary_Website_or_URL` column has a large black bar, which indicates many missing values in the column. A reasonable interpretation of these missing values is that these Farmer's Markets do not have a known web site. To explore these markets, you click the black bar:



FMID	MarketName	Primary_Website_or_URL	
20k - 1.01M	7,474 Categories	3,586 Categories	8,125 Cat
1005969	???Y Not Wednesday Farmers Market at Town Center???	http://www.sandlercenter.org/ynotwednesdays	201 Ma
1008044	10:10 Farmers Market	http;//www.1010farmersmarket.com	5960
1000618	100-Mile Market	http://www.peoplesfoodco-op.org/	507 Ha
1002454	112st Madison Avenue	null	112th
1005586	12th & Brandywine Urban Farm Market	null	12th
1008071	14&U Farmers' Market	null	1400
1007098	17 on the Square	null	17 Lin
1000059	175th Street Greenmarket	http;//www.grownyc.org	W 175
1003877	17th Ave Market	http://www.iatp.org/minimarkets	1622
1002619	17th Street Farmers Market	http://www.17thstreetfarmersmarket.com	17th
1000709	???19th Annual Highlands Business Partnership Farmers Marke	http;//www.highlandsnj.com	71 Wat
1003233	2012 Wood County Farmers' Market	null	555 W
1005309	22nd and Tasker Farmers' Market	http://www.foodtrustmarkets.org/markets.html	22nd
1006576	25th Avenue Farmers' Market	http://www.pcfma.com/market_home.php?market_id=40	194 W
1005299	29th and Wharton Farmers' Market	http://www.foodtrustmarkets.org/graysferry.html	29th
1000380	2nd Street Public Market	http://www.secondstreetpublicmarket.com	2003
1004950	3 French Hens French Country Market	http://www.3frenchhensmarket.blogspot.com	123 W
1005636	32nd Street/Waverly Farmers Market	http;//www.32ndstreetmarket.org	E. 32r
1005310	33rd and Diamond Farmers' Market	http://www.foodtrustmarkets.org/markets.html	N 33rc
1003161	38th & Meridian Farmers Market	null	3808
1004414	3rd & Curry St. Farmers Market	http://www.carsonfarmersmarket.com	3rd &
1002860	3rd Street N (hwy 11) beside jack's Restaurant	null	3rd St
1006234	4th Street Farmers Market	http://www.4thstreetfarmersmarket.com	315 Ea
1006494	52 & Shadeland Avenue Farmers Market	null	5200
1005304	52nd and Haverford Farmers' Market	http://www.foodtrustmarkets.org/markets.html	N 52nd

9 Columns 8,184 Rows 5 Data Types Show only affected ☐ Rows

Delete rows
where ISMISSING([Primary_Website_or_URL])

Keep rows
where ISMISSING([Primary_Website_or_URL])

Create a new column
ISMISSING([Primary_Website_or_URL])

Set
Set Primary_Website_or_URL to IFMISSING($col, NULL())

Figure: Interact: select missing values in a column

A couple of things to notice:

- The right panel now displays a list of suggestions. These suggestions correspond to transformations that you can apply to the current selection. In this case, the selected data are the missing values in the `Primary_Website_or_URL` column.
- The first suggestion is selected by default. In the data grid, you can see a preview of what would happen if this step was added to the recipe. The highlighted rows would be removed.

Things you can select:

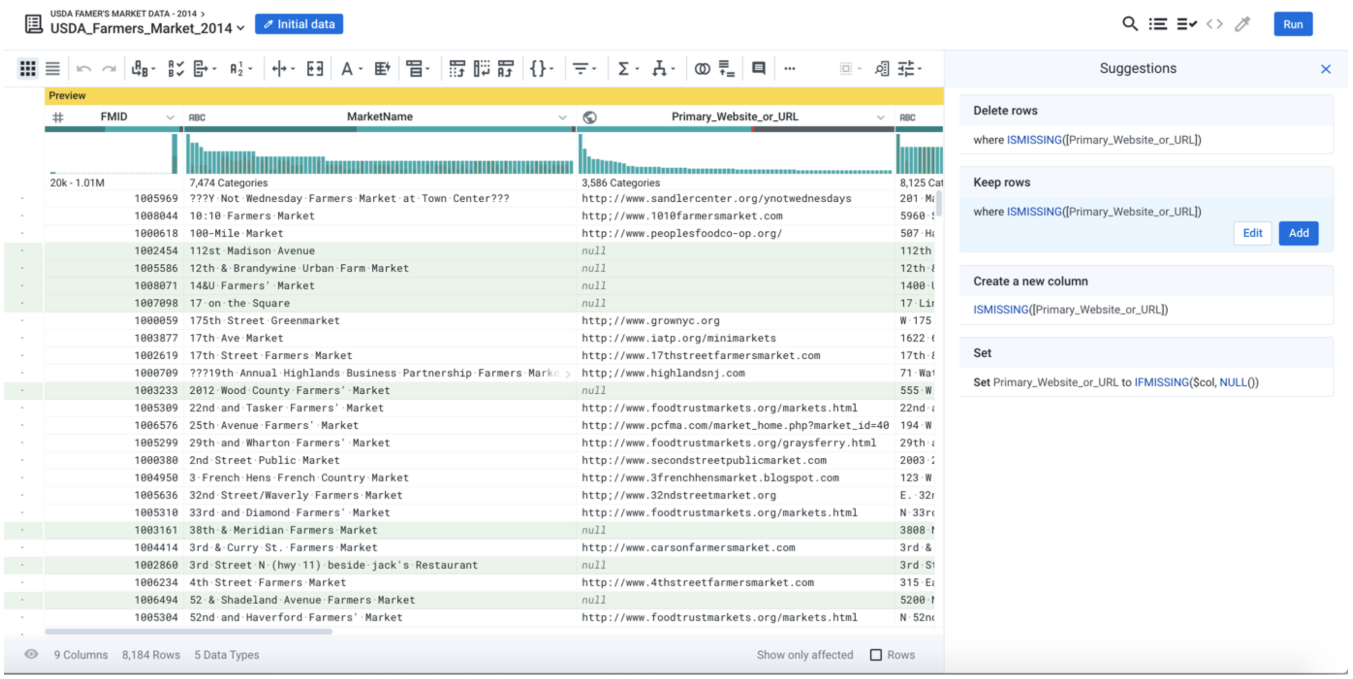
- **Column(s):** Click a column header. Use `SHIFT` or `CTRL/COMMAND` to select multiple columns.
- **Row(s):** To select a row, click the dot on the left side of the data grid.
- **Cell values:** Click and drag to select part of a cell value. Double-click to select the whole cell value.
- **Histogram values:**You can select one or more values in the histogram. Based on your selection(s), a set of suggested changes is presented to you. These changes apply to the rows where the selected values occur.
- **Data quality bar:** You can select bars in the data quality bar.

3 - Predict

In the above example, predictive transformation predicted that based on your selection, you wanted to delete the rows where the website URL column was empty. This prediction is based on multiple factors, including:

- Platform algorithms that interpret the meaning of user selections
- Tracking of previous user interactions with your project or workspace of a similar nature.

So, the prediction makes sense; in most cases, if data is missing, you don't want the rest of the incomplete data in the row. However, suppose your interest in the data is different. Suppose that you are a website builder. You want to find the farmer's markets that don't have a website as potential customers. In this case, the second suggested transformation makes more sense: Keep these rows with missing values:



FMID	MarketName	Primary_Website_or_URL	
20k - 1.01M	7,474 Categories	3,586 Categories	8,125 Cat
1005969	???Y Not Wednesday Farmers Market at Town Center???	http://www.sandlercenter.org/ynotwednesdays	201 Ma
1008044	10:10 Farmers Market	http;//www.1010farmersmarket.com	5960
1000618	100-Mile Market	http://www.peoplesfoodco-op.org/	507 Ha
1002454	112st Madison Avenue	null	112th
1005586	12th & Brandywine Urban Farm Market	null	12th &
1008071	14&U Farmers' Market	null	1400 U
1007098	17 on the Square	null	17 Lir
1000059	175th Street Greenmarket	http;//www.grownyc.org	W 175
1003877	17th Ave Market	http://www.iatp.org/minimarkets	1622
1002619	17th Street Farmers Market	http://www.17thstreetfarmersmarket.com	17th &
1000709	???19th Annual Highlands Business Partnership Farmers Marke	http;//www.highlandsnj.com	71 Wat
1003233	2012 Wood County Farmers' Market	null	555 W
1005309	22nd and Tasker Farmers' Market	http://www.foodtrustmarkets.org/markets.html	22nd a
1006576	25th Avenue Farmers' Market	http://www.pcfma.com/market_home.php?market_id=40	194 W
1005299	29th and Wharton Farmers' Market	http://www.foodtrustmarkets.org/graysferry.html	29th a
1000380	2nd Street Public Market	http://www.secondstreetpublicmarket.com	2003
1004950	3 French Hens French Country Market	http://www.3frenchhensmarket.blogspot.com	123 W
1005636	32nd Street/Waverly Farmers Market	http;//www.32ndstreetmarket.org	E. 32r
1005310	33rd and Diamond Farmers' Market	http://www.foodtrustmarkets.org/markets.html	N 33rc
1003161	38th & Meridian Farmers Market	null	3808 I
1004414	3rd & Curry St. Farmers Market	http://www.carsonfarmersmarket.com	3rd &
1002860	3rd Street N (hwy 11) beside jack's Restaurant	null	3rd St
1006234	4th Street Farmers Market	http://www.4thstreetfarmersmarket.com	315 Ea
1006494	52 & Shadeland Avenue Farmers Market	null	5200 I
1005304	52nd and Haverford Farmers' Market	http://www.foodtrustmarkets.org/markets.html	N 52nc

9 Columns 8,184 Rows 5 Data Types Show only affected Rows

Suggestions

Delete rows
where ISMISSING([Primary_Website_or_URL])

Keep rows
where ISMISSING([Primary_Website_or_URL])
Edit Add

Create a new column
ISMISSING([Primary_Website_or_URL])

Set
Set Primary_Website_or_URL to IFMISSING($col, NULL())

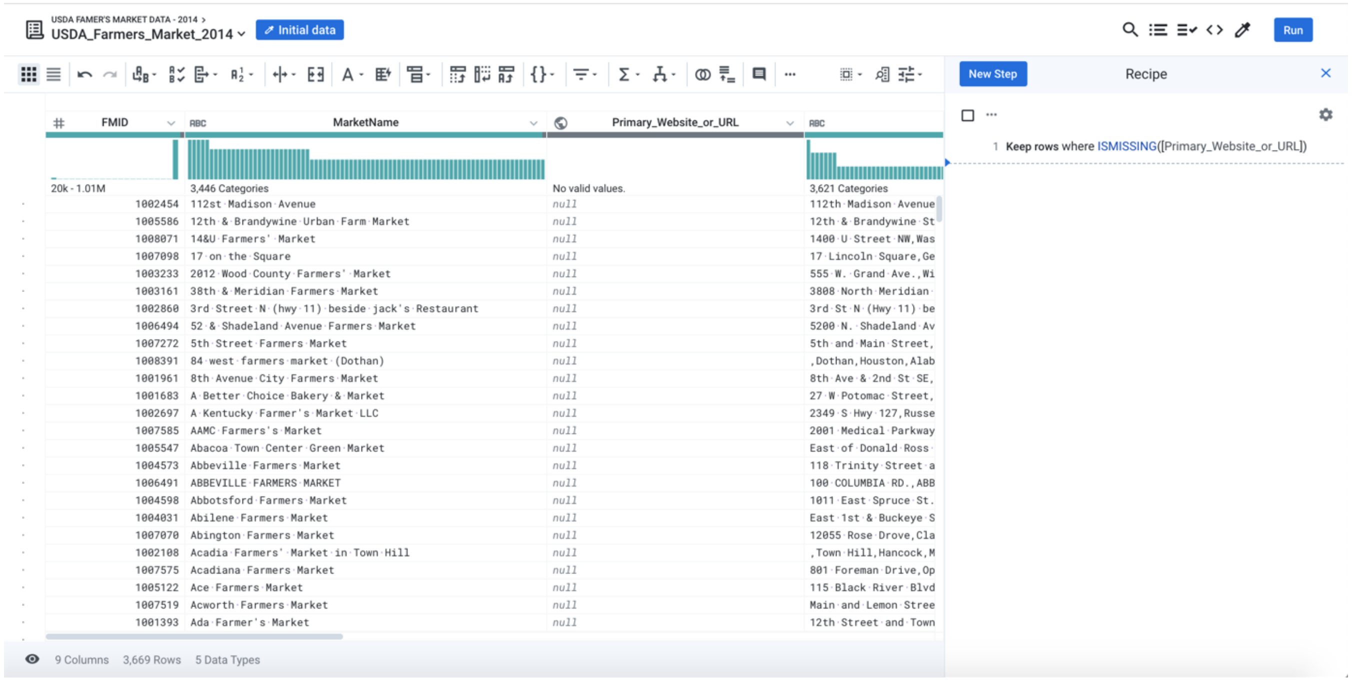
Figure: Predict: Multiple suggestions serve as guides to possible useful transformations

When the second suggestion is selected, the affected rows are highlighted in green, indicating that they will be retained.

Tip: To visualize assist in reviewing these rows, click the Show only affected checkbox at the bottom of the screen. Only the rows are displayed where the suggested transformation is to be applied.

4 - Present

If the selected suggestion looks good, click **Add**:



1 Keep rows where ISMISSING([Primary_Website_or_URL])

FMID	MarketName	Primary_Website_or_URL	
20k - 1.01M	3,446 Categories	No valid values.	3,621 Categories
1002454	112st Madison Avenue	null	112th Madison Avenue
1005586	12th & Brandywine Urban Farm Market	null	12th & Brandywine St
1008071	14&U Farmers' Market	null	1400 U Street NW,Was
1007098	17 on the Square	null	17 Lincoln Square,Ge
1003233	2012 Wood County Farmers' Market	null	555 W. Grand Ave.,Wi
1003161	38th & Meridian Farmers Market	null	3808 North Meridian
1002860	3rd Street N (hwy 11) beside jack's Restaurant	null	3rd St N (Hwy 11) be
1006494	52 & Shadeland Avenue Farmers Market	null	5200 N. Shadeland Av
1007272	5th Street Farmers Market	null	5th and Main Street,
1008391	84 west farmers market (Dothan)	null	,Dothan,Houston,Alab
1001961	8th Avenue City Farmers Market	null	8th Ave & 2nd St SE,
1001683	A Better Choice Bakery & Market	null	27 W Potomac Street,
1002697	A Kentucky Farmer's Market LLC	null	2349 S Hwy 127,Russe
1007585	AAMC Farmers's Market	null	2001 Medical Parkway
1005547	Abacoa Town Center Green Market	null	East of Donald Ross
1004573	Abbeville Farmers Market	null	118 Trinity Street a
1006491	ABBEVILLE FARMERS MARKET	null	100 COLUMBIA RD.,ABB
1004598	Abbotsford Farmers Market	null	1011 East Spruce St.
1004031	Abilene Farmers Market	null	East 1st & Buckeye S
1007070	Abington Farmers Market	null	12055 Rose Drove,Cla
1002108	Acadia Farmers' Market in Town Hill	null	,Town Hill,Hancock,M
1007575	Acadiana Farmers Market	null	801 Foreman Drive,Op
1005122	Ace Farmers Market	null	115 Black River Blvd
1007519	Acworth Farmers Market	null	Main and Lemon Stree
1001393	Ada Farmer's Market	null	12th Street and Town

9 Columns 3,669 Rows 5 Data Types

Figure: Present: Data grid is updated based on the transformation that has been applied

The transformation has been added to your recipe. Note at there is now a step in the recipe panel at right, which was previously empty. You've added a recipe step, and your sampled data is transformed.

The Cycle

The above represents the basic cycle of using the Transformer page:

1. **Display & Interact:** Locate data of interest and select it.
2. **Predict:** Review the suggested transformations.
   a. Click them to preview the results.
   b. You can also click **Edit** to make modifications to the suggested transformation before you add it.
3. **Present:** Add the selected and modified suggestion to your recipe. The sample of data in the data grid is transformed.

Repeat the above steps.

Recommended Methods for Building Recipes

Designer Cloud Powered by Trifacta® Enterprise Edition supports the following methods for building recipes in the Transformer page. 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.

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 Transformer 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.
b. The column menus contain the most common transformations for individual or multiple columns. Often, no additional configuration is required.
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 Transform Builder is pre-populated from your selection in the Search panel.

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

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 **Initial 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 Transformer page, you can launch a new sampling job on your dataset from the Samples panel. You may have to open it first.

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.

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	`$sourcerownumber`
Parameter: New column name	`rowId`

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	`$filepath`
Parameter: New column name	`filepath`

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 multiple columns, including ranges of columns.

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.

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.

Review data quality:

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

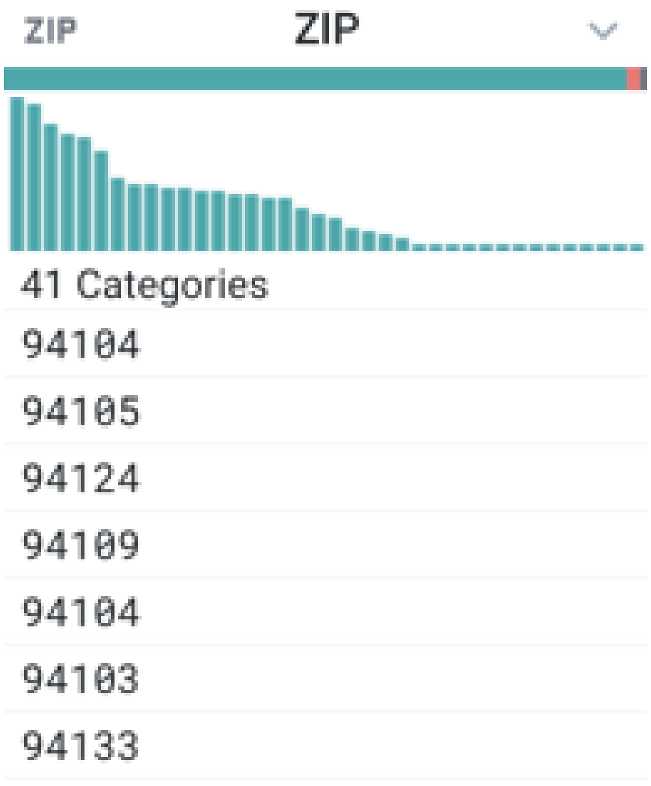


Figure: Column header with data quality bar

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.

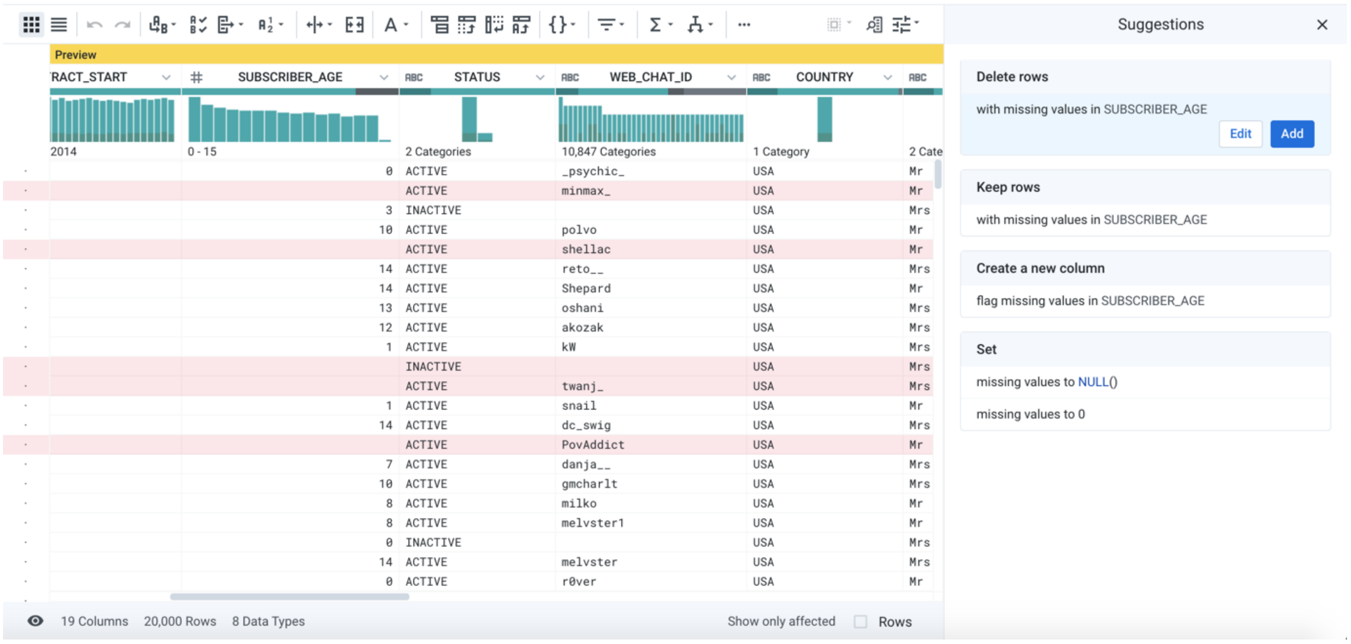
Tip: When you select values in the data quality bar, those values are highlighted in the sample rows, and suggestions are displayed at the bottom of the screen in the suggestion cards to address the selected rows.

Example - 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.



Preview

RACT_START	SUBSCRIBER_AGE	STATUS	WEB_CHAT_ID	COUNTRY	
2014	0 - 15	2 Categories	10,847 Categories	1 Category	2 Cate
	0	ACTIVE	_psychic_	USA	Mr
		ACTIVE	minmax_	USA	Mr
	3	INACTIVE		USA	Mrs
	10	ACTIVE	polvo	USA	Mr
		ACTIVE	shellac	USA	Mr
	14	ACTIVE	reto__	USA	Mrs
	14	ACTIVE	Shepard	USA	Mr
	13	ACTIVE	oshani	USA	Mrs
	12	ACTIVE	akozak	USA	Mrs
	1	ACTIVE	kW	USA	Mrs
		INACTIVE		USA	Mrs
		ACTIVE	twanj_	USA	Mrs
	1	ACTIVE	snail	USA	Mr
	14	ACTIVE	dc_swig	USA	Mrs
		ACTIVE	PovAddict	USA	Mr
	7	ACTIVE	danja__	USA	Mrs
	10	ACTIVE	gmcharlt	USA	Mrs
	8	ACTIVE	milko	USA	Mr
	8	ACTIVE	melvster1	USA	Mr
	0	INACTIVE		USA	Mrs
	14	ACTIVE	melvster	USA	Mrs
	0	ACTIVE	r0ver	USA	Mr

19 Columns 20,000 Rows 8 Data Types Show only affected ☐ Rows

Suggestions

Delete rows
with missing values in SUBSCRIBER_AGE
Edit Add

Keep rows
with missing values in SUBSCRIBER_AGE

Create a new column
flag missing values in SUBSCRIBER_AGE

Set
missing values to NULL()
missing values to 0

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 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. `10000022`) 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.

Tip: Designer Cloud Powered by Trifacta Enterprise Edition maintains statistical information and enable some transformation steps based upon data type.

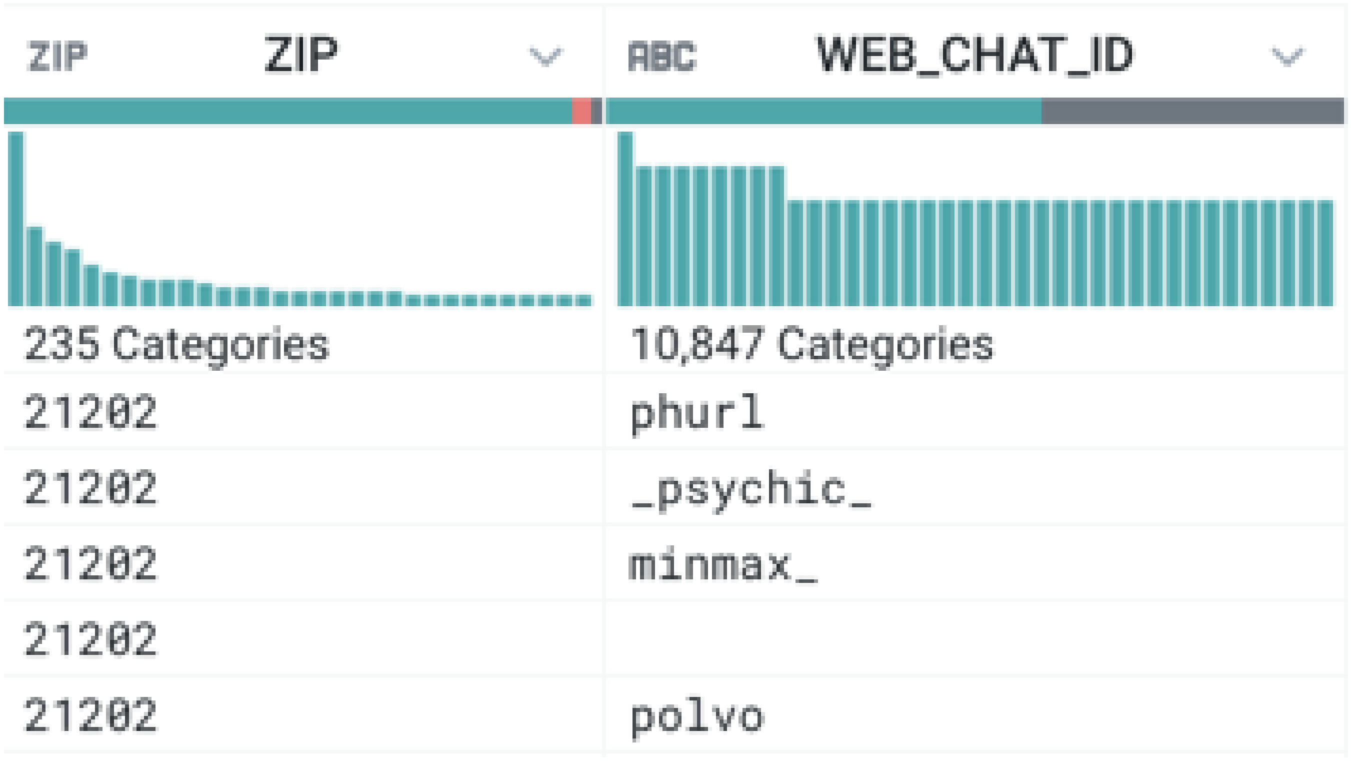
1. To change a column's data type, click the icon to the left of the column title.
2. Select the new data type.
3. Review the mismatched values for the column to verify that their count has dropped.

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**.

Review Histograms

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



ZIP	WEB_CHAT_ID
21202	phurl
21202	_psychic_
21202	minmax_
21202	
21202	polvo

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.

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 using the Union page.

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` to 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...** .

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 data 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.

Profile

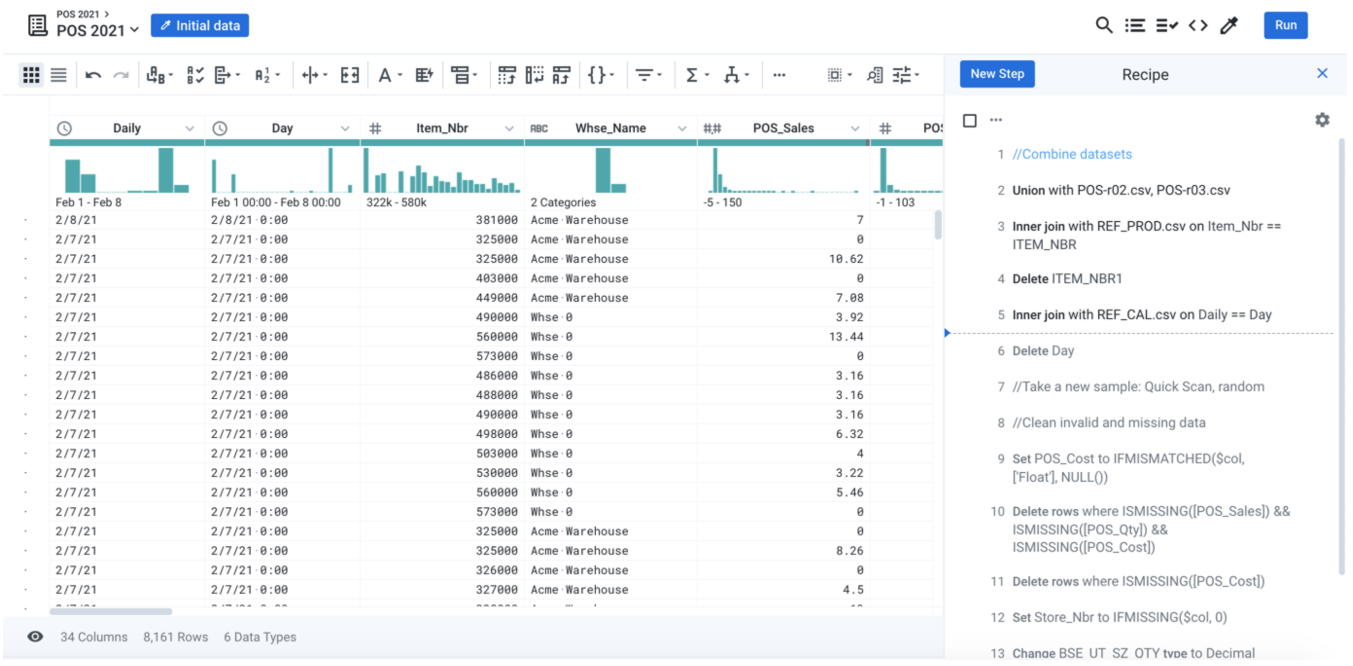
As part of the transformation process, you can generate and review visual profiles of individual columns and your entire dataset. These interactive profiles can be very helpful in identifying anomalies, outliers, and other issues with your data.

Recipe Basics

Contents:

In the Transformer page, every transformation that you apply to your data adds a new step to your recipe. A **recipe** is the sequence of steps that are applied to your imported datasets to transform them for output.

When you select items and apply suggestions or perform other transformative actions in the Transformer page, you are creating new steps for your recipe. Your recipe in development is available through the Recipe panel. If the panel on the right below is not visible, click the **Recipe** icon in the toolbar at the top of the screen:



Daily	Day	Item_Nbr	Whse_Name	POS_Sales
Feb 1 - Feb 8	Feb 1 00:00 - Feb 8 00:00	322k - 580k	2 Categories	-5 - 150
2/8/21	2/8/21 0:00	381000	Acme Warehouse	7
2/7/21	2/7/21 0:00	325000	Acme Warehouse	0
2/7/21	2/7/21 0:00	325000	Acme Warehouse	10.62
2/7/21	2/7/21 0:00	403000	Acme Warehouse	0
2/7/21	2/7/21 0:00	449000	Acme Warehouse	7.08
2/7/21	2/7/21 0:00	490000	Whse 0	3.92
2/7/21	2/7/21 0:00	560000	Whse 0	13.44
2/7/21	2/7/21 0:00	573000	Whse 0	0
2/7/21	2/7/21 0:00	486000	Whse 0	3.16
2/7/21	2/7/21 0:00	488000	Whse 0	3.16
2/7/21	2/7/21 0:00	490000	Whse 0	3.16
2/7/21	2/7/21 0:00	498000	Whse 0	6.32
2/7/21	2/7/21 0:00	503000	Whse 0	4
2/7/21	2/7/21 0:00	530000	Whse 0	3.22
2/7/21	2/7/21 0:00	560000	Whse 0	5.46
2/7/21	2/7/21 0:00	573000	Whse 0	0
2/7/21	2/7/21 0:00	325000	Acme Warehouse	0
2/7/21	2/7/21 0:00	325000	Acme Warehouse	8.26
2/7/21	2/7/21 0:00	326000	Acme Warehouse	0
2/7/21	2/7/21 0:00	327000	Acme Warehouse	4.5

```
1  //Combine datasets
2  Union with POS-r02.csv, POS-r03.csv
3  Inner join with REF_PROD.csv on Item_Nbr ==
   ITEM_NBR
4  Delete ITEM_NBR1
5  Inner join with REF_CAL.csv on Daily == Day
6  Delete Day
7  //Take a new sample: Quick Scan, random
8  //Clean invalid and missing data
9  Set POS_Cost to IFMISMATCHED($col,
   ['Float'], NULL())
10 Delete rows where ISMISSING([POS_Sales]) &&
   ISMISSING([POS_Qty]) &&
   ISMISSING([POS_Cost])
11 Delete rows where ISMISSING([POS_Cost])
12 Set Store_Nbr to IFMISSING($col, 0)
13 Change BSE_UT_SZ_QTY type to Decimal
```

Figure: Transformer page with Recipe panel

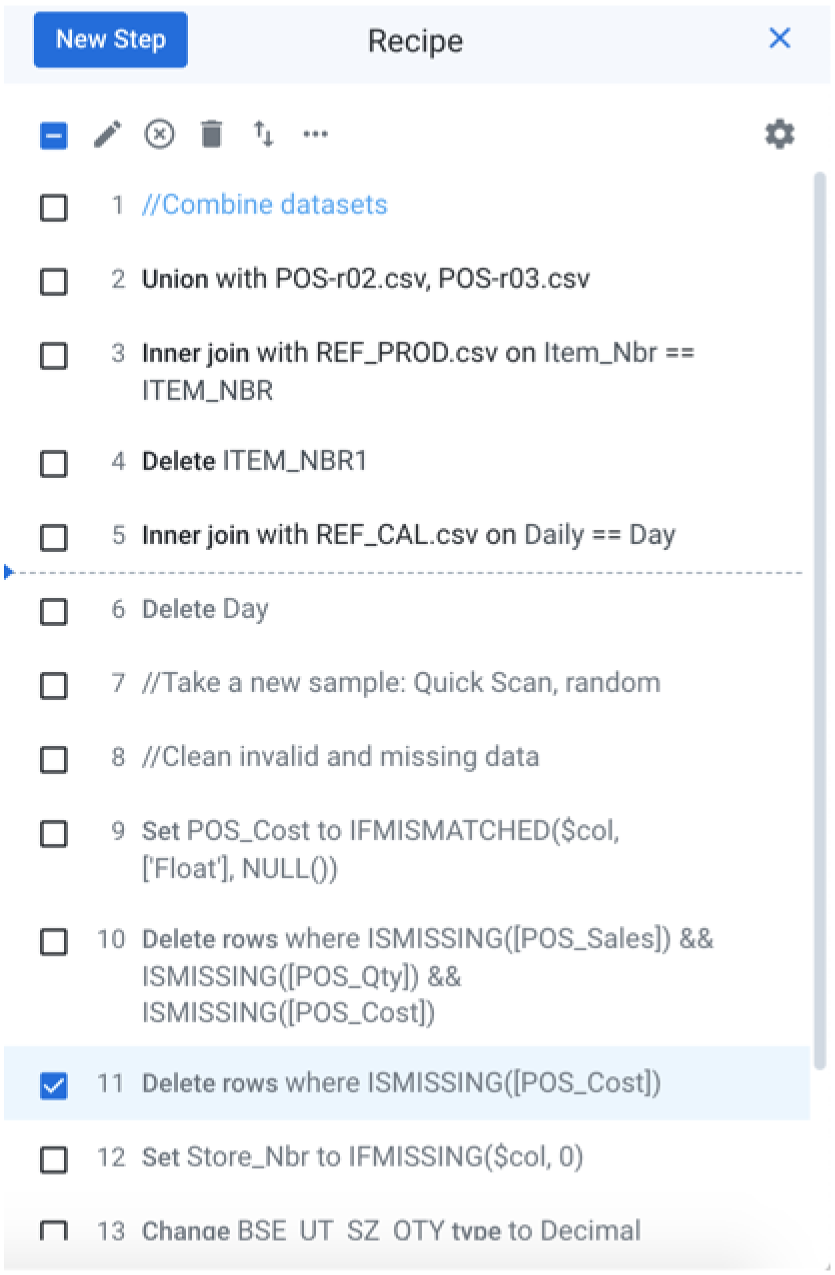
Steps are crafted in a proprietary language. Wrangle (a domain-specific language for data transformation) is created for use in Designer Cloud Powered by Trifacta Enterprise Edition.

- By default, recipe steps are displayed in a readable, more natural language than the underlying Wrangle .
- From the Gear icon, you can choose **Display language** to view the underlying language steps.

Recipe Panel

In the Recipe panel, you can review and manipulate your recipe steps, including re-ordering them, editing them, and disabling them. The list of steps in the Recipe panel represent the current state of your recipe. Each numeric item corresponds to:

- The sequence number in the recipe when the step is executed.
- The description of the transformation in the step.



1 //Combine datasets
2 Union with POS-r02.csv, POS-r03.csv
3 Inner join with REF_PROD.csv on Item_Nbr == ITEM_NBR
4 Delete ITEM_NBR1
5 Inner join with REF_CAL.csv on Daily == Day
6 Delete Day
7 //Take a new sample: Quick Scan, random
8 //Clean invalid and missing data
9 Set POS_Cost to IFMISMATCHED($col, ['Float'], NULL())
10 Delete rows where ISMISSING([POS_Sales]) && ISMISSING([POS_Qty]) && ISMISSING([POS_Cost])
11 Delete rows where ISMISSING([POS_Cost])
12 Set Store_Nbr to IFMISSING($col, 0)
13 Change BSE_UT_SZ_QTY type to Decimal

Figure: Recipe panel

Key elements of the recipe panel:

- **Recipe cursor:** Represented by the dotted line in the above image, the recipe cursor indicates the state of the data in the data grid at that step. The current data is the current sample of data, plus all of the steps between the sample and the recipe cursor.

- **Selected step:** The step with the selected checkbox next to it is the currently selected step. In the context menu of the selected step are a set of commands that you can apply to it. See below.

  Tip: You can select multiple steps using `SHIFT` + click or `CMD/CTRL` + click. The available context menu options may change for multiple steps.

  Tip: If you make a mistake, press `CMD/CTRL` + `Z` to try to undo it. Some recipe actions cannot be undone.

Step Actions

Add a new step

Steps:

1. Move the recipe cursor to the location after which you wish to add a step.

   Tip: You can also select a step and then select **Insert after step** to insert a new step.

2. Click **Add New Step**.
3. In the Search panel, search for the transformation or function to add. Select the appropriate one from the list.
4. Specify the transformation settings as needed.
5. Click **Add**.

Edit steps

- To edit a step, select the step. From the step context menu, select **Edit**.

  NOTE: You cannot change the base transformation of a step. Create a new step instead.

- To disable a step, select **Disable** from the step context menu.

  Tip: Disabling a step is a good way to experiment with different steps at that location. After disabling the step, you can add a new step, which is automatically enabled. If the new step works, then you can safely delete the disabled step.

- To delete a step, select **Delete** from the step context menu.

  Deleting a step cannot be undone.

Move steps

Steps:

1. Select the step or steps that you wish to move. You can select multiple steps if needed.
2. In the step context menu, select **Move**. In the sub-menu, choose where you wish to move the step or steps.

Tip: You can also click the Up/Down arrows in the recipe menu bar.

3. Make additional selections to move as needed. You can undo the move if needed.

Cut, copy, and paste steps

From the step context menu, you can cut, copy, and paste steps.

Recipe Toolbar

Some of the above commands are also available through the toolbar at the top of the recipe. Additional commands are available through the Gear icon.

Review edit history

You can see the history of changes and who made them to the recipe. From the Gear icon, select **See edit history**.

Download recipe

You can download the current recipe as a text file containing the Wrangle steps. From the Gear icon, select **Download Recipe as wrangle**.

Download sample

You can download the current state of the data grid in CSV format. Select the step that you'd like to see in the data grid. Then, from the Gear icon, select **Download Sample data as CSV**.

Sampling Basics

Contents:

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

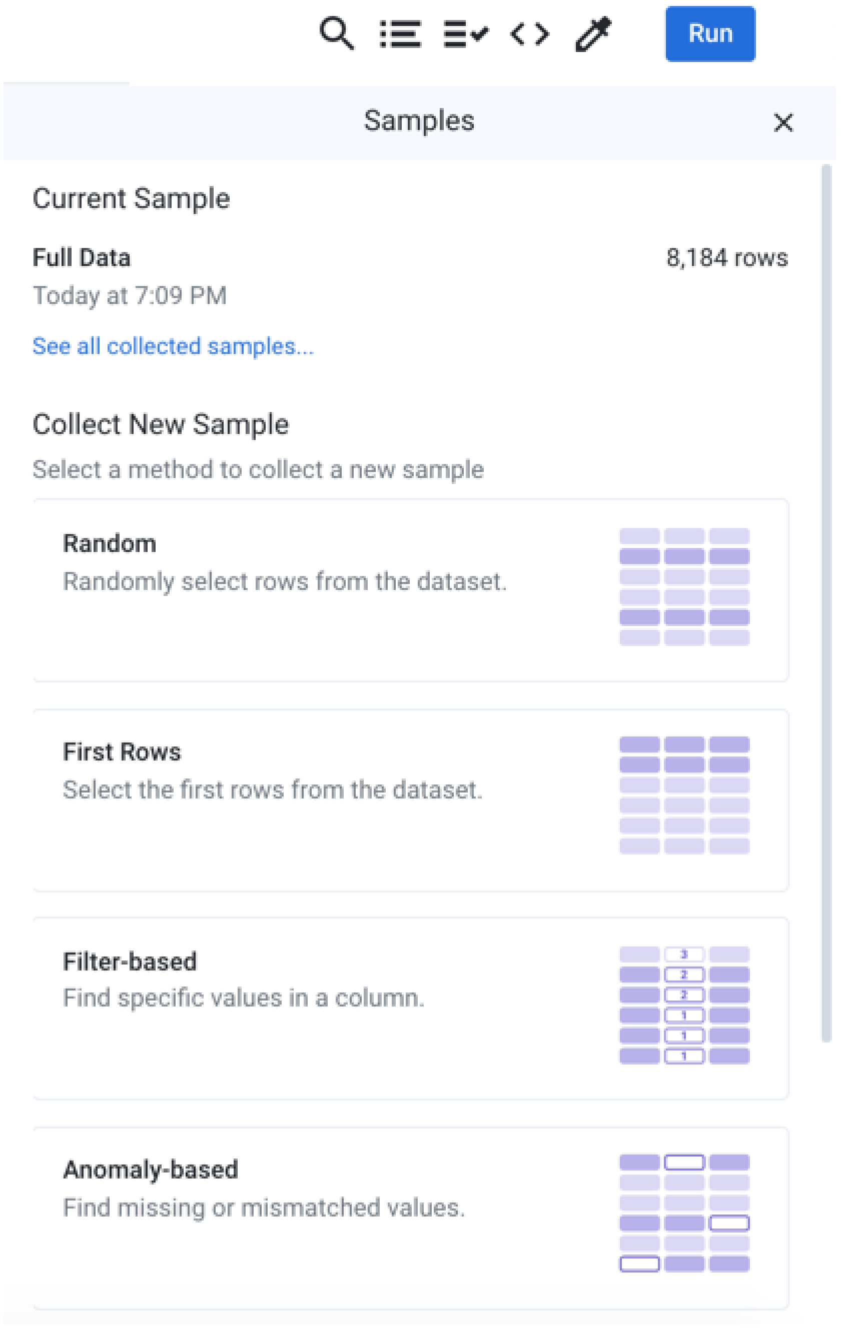
Take a Sample

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.



Current Sample

Full Data — 8,184 rows

Today at 7:09 PM

See all collected samples...

Collect New Sample

Select a method to collect a new sample

Random — Randomly select rows from the dataset.

First Rows — Select the first rows from the dataset.

Filter-based — Find specific values in a column.

Anomaly-based — Find missing or mismatched values.

Figure: Samples panel

3. At the top of the panel, you can review the Current Sample.

> **Tip:** In some cases, then 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.

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.

Best Practices

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

Running Job Basics

Contents:

This section provides an overview of running and executing job basics.

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. If you do not specify a publishing action, which includes output information, a default one is created for you.

For more information, 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*.

Visual Profiling

When you define your job, you can choose to generate a visual profile, which provides visual information on the quality of your results, including statistical information about each column.

Tip: Visual profiles can be useful for troubleshooting wrangling issues and for summarizing your datasets for other analysis.

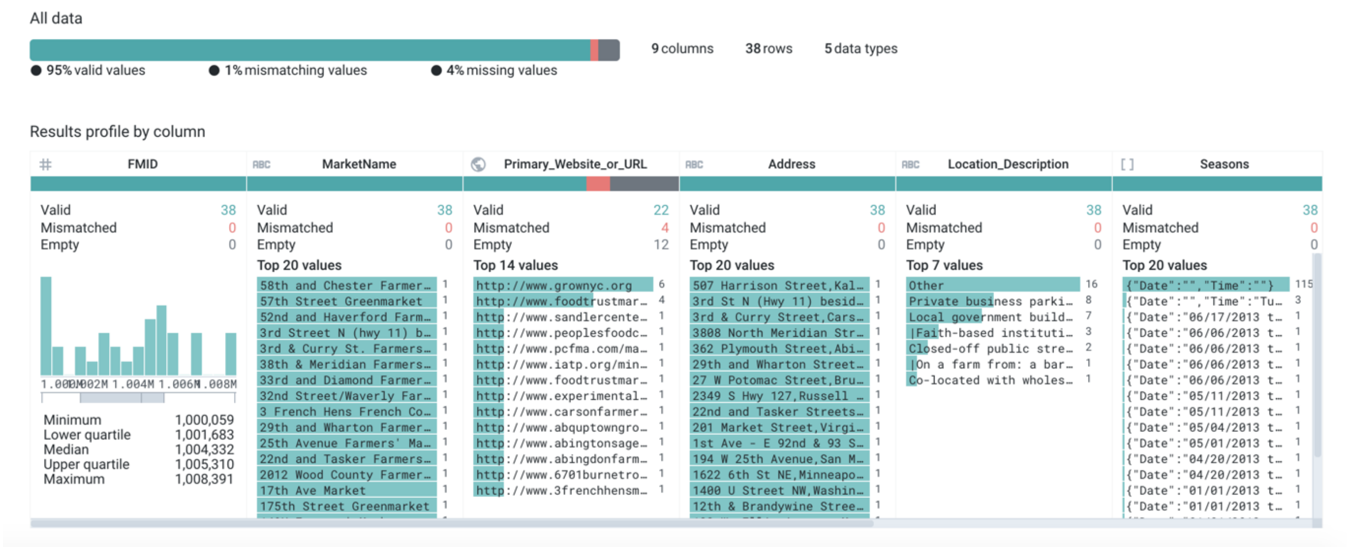
Tip: Optionally, you can disable generating a visual profile of your results. While the visual profile is very useful for examining issues in your recipe and iterating, it is a resource-intensive process. If you are working with large datasets that do not require additional debugging, you can consider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

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

9 columns 38 rows 5 data types

● 95% valid values ● 1% mismatching values ● 4% missing values

Results profile by column

FMID

Valid	38
Mismatched	0
Empty	0
Minimum	1,000,059
Lower quartile	1,001,683
Median	1,004,332
Upper quartile	1,005,310
Maximum	1,008,391

MarketName

Valid	38
Mismatched	0
Empty	0
Top 20 values	
58th and Chester Farmer…	1
57th Street Greenmarket	1
52nd and Haverford Farm…	1
3rd Street N (hwy 11) b…	1
3rd & Curry St. Farmers…	1
38th & Meridian Farmers…	1
33rd and Diamond Farmer…	1
32nd Street/Waverly Far…	1
3 French Hens French Co…	1
29th and Wharton Farmer…	1
25th Avenue Farmers' Ma…	1
22nd and Tasker Farmers…	1
2012 Wood County Farmer…	1
17th Ave Market	1
175th Street Greenmarket	1

Primary_Website_or_URL

Valid	22
Mismatched	4
Empty	12
Top 14 values	
http://www.grownyc.org	6
http://www.foodtrustmar…	4
http://www.sandlercente…	1
http://www.peoplesfoodc…	1
http://www.pcfma.com/ma…	1
http://www.iatp.org/min…	1
http://www.foodtrustmar…	1
http://www.experimental…	1
http://www.carsonfarmer…	1
http://www.abquptowngro…	1
http://www.abingtonsage…	1
http://www.abingdonfarm…	1
http://www.6701burnetro…	1
http://www.3frenchhensm…	1

Address

Valid	38
Mismatched	0
Empty	0
Top 20 values	
507 Harrison Street,Kal…	1
3rd St N (Hwy 11) besid…	1
3rd & Curry Street,Cars…	1
3808 North Meridian Str…	1
362 Plymouth Street,Abi…	1
29th and Wharton Street…	1
27 W Potomac Street,Bru…	1
2349 S Hwy 127,Russell …	1
22nd and Tasker Streets…	1
201 Market Street,Virgi…	1
1st Ave - E 92nd & 93 S…	1
194 W 25th Avenue,San M…	1
1622 6th St NE,Minneapo…	1
1400 U Street NW,Washin…	1
12th & Brandywine Stree…	1

Location_Description

Valid	38
Mismatched	0
Empty	0
Top 7 values	
Other	16
Private business parki…	8
Local government build…	7
|Faith-based instituti…	3
Closed-off public stre…	2
|On a farm from: a bar…	1
Co-located with wholes…	1

Seasons

Valid	38
Mismatched	0
Empty	0
Top 20 values	
{"Date":"","Time":""}	115
{"Date":"","Time":"Tu…	3
{"Date":"06/17/2013 t…	1
{"Date":"06/06/2013 t…	1
{"Date":"06/06/2013 t…	1
{"Date":"06/06/2013 t…	1
{"Date":"06/06/2013 t…	1
{"Date":"05/11/2013 t…	1
{"Date":"05/11/2013 t…	1
{"Date":"05/04/2013 t…	1
{"Date":"05/01/2013 t…	1
{"Date":"04/20/2013 t…	1
{"Date":"04/20/2013 t…	1
{"Date":"01/01/2013 t…	1
{"Date":"01/01/2013 t…	1

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.

Create Output

Every job requires an output object, which defines the format, location, and other settings of the output file(s) or table(s) that are produced when the job completes.

Tip: When you run the first job on a recipe, a default output object is created for you to produce a CSV in your default output location.

You can modify or define outputs to meet your pipeline requirements. For more information, see *Create Outputs*.

Export Results

During job execution, you can monitor progress on the job through the Job Details page. When the job is complete, your results are ready in the designated output location and format.

As needed, you can download results for offline use or use them to create a new dataset.

For more information, see *Export Basics*.

Schedule Jobs

Feature Availability: This feature may not be available in all product editions. For more information on available features, see *Compare Editions*.

In many cases, the same job needs to be executed on a periodic basis. For example, your source dataset may be updated on a weekly basis to include a fresh set of transactions. As needed, you can schedule the execution of your jobs to refresh the output data. For more information, see *Schedule a Job*.

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 Job History icon.
2. In the Job History page, click the job identifier to open the job in the 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**.

- **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**.

Common Tasks

Contents:

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

Import Tasks

- *Import from Another Flow*
- *Import Excel Data*
- *Import Google Sheets Data*
- *Import PDF Data*
- *Create Flow*

Discovery Tasks

- *Explore Suggestions*
- *Review Column Statistics*
- *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*
    - *Change Recipe Sample Size*

Validation Tasks

- *Profile Your Source Data*
- *Validate Your Data*
    - *Validate Column Values against a Dataset*
- *Find Bad Data*
- *Find Missing Data*
- *Manage Null Values*

Structuring Tasks

- *Initial Parsing Steps*
- *Transform a Column*
- *Transform a Row*

- *Split Column*
- *Move Columns*
- *Add New Column*
- *Delete Data*
- *Sort Rows*
- *Reshaping Steps*
- *Select*
- *Create Aggregations*
- *Nest Your Data*
- *Unnest Your Data*
- *Pivot Data*
    - *Values to Columns*
- *Unpivot Columns*
- *Window Transformations*
- *Working with Arrays*
- *Working with Objects*
- *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

- *Run Jobs*
- *Create Outputs*
    - *Create Output SQL Scripts*

- *Profile Results*
- *Publish Results on Demand*
- *Reuse Recipe*

Project Management Tasks

- *Take a Snapshot*
- *Track Data Changes*
- *Add Comments to Your Recipe*
- *Create Target*
- *Optimize Job Processing*
- *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*
- *Create Flow Parameter*
- *Flag for Review*
- *Export Data*

Operationalization Tasks

- *Create Flow Webhook Task*

Import Tasks

These tasks 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.

Import from Another Flow

Contents:

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.

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.

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 in Flow View, locate the recipe whose output you wish to use in another recipe.
2. Right-click and select **Add > Reference**.

3. The reference is created:

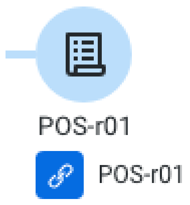


Figure: Reference object

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.

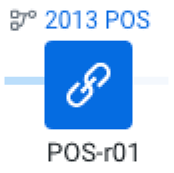


Figure: Reference dataset in a new flow

For more information, see *View for Reference Datasets*.

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 in the Run Job page.
   d. When the job completes, click the job identifier. The Job Details page opens.
   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.
   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 one or more Excel files as a parameterized dataset, you select worksheets to include from the first file. If there are worksheets in other Excel files that match the names of the worksheets that you selected, those worksheets are also imported. All worksheets are unioned together into a single imported dataset with parameters. Pattern-based parameters are not supported for import of Excel worksheets.

Limitations

> **NOTE:** Before you begin, you should review information on file formats supported for import, which cause your files to fail to import or to be properly ingested and formatted. For more information, see *Supported File Formats*.

- XLSX and XLS format are supported. Other Excel-related formats, such as XLSM format, are not supported.
  - Strict Open XML format is not supported. You can save the file as an XLSX file and then import it.
- Some characters, such as hashtags (#) and curly braces (`{}`) cannot be used in filenames. For more information, see *Supported File Formats*.
- 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.

- Conversion of large Excel files require non-linear increases in memory requirements on the Trifacta node.

- 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.

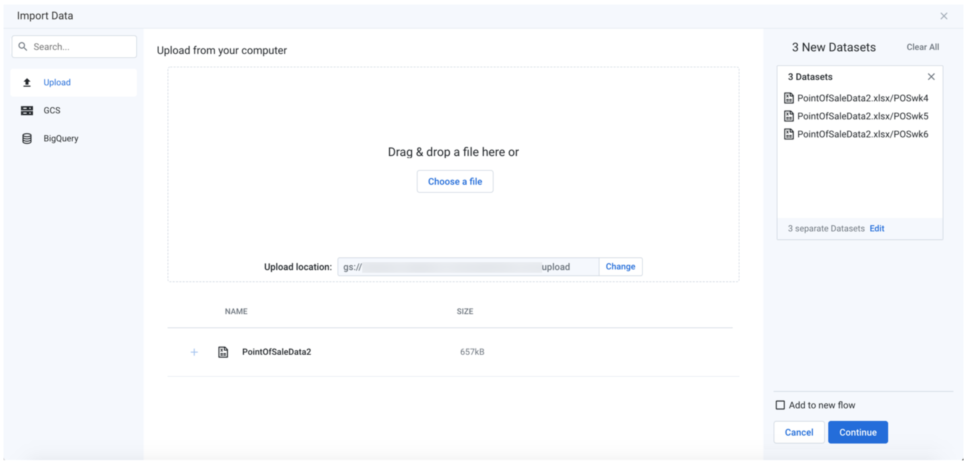
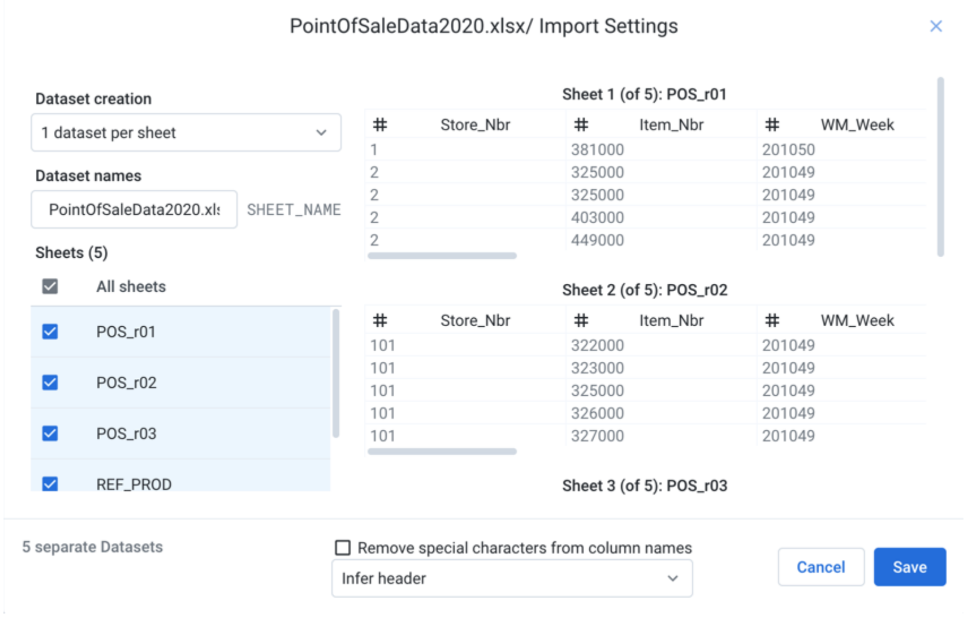


Figure: Import Excel workbook

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.



PointOfSaleData2020.xlsx/ Import Settings

Dataset creation: 1 dataset per sheet

Dataset names: PointOfSaleData2020.xls SHEET_NAME

Sheets (5)

- [✓] All sheets
- [✓] POS_r01
- [✓] POS_r02
- [✓] POS_r03
- [✓] REF_PROD

Sheet 1 (of 5): POS_r01

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

Sheet 2 (of 5): POS_r02

# Store_Nbr	# Item_Nbr	# WM_Week
101	322000	201049
101	323000	201049
101	325000	201049
101	326000	201049
101	327000	201049

Sheet 3 (of 5): POS_r03

5 separate Datasets

[ ] Remove special characters from column names

Infer header

Cancel Save

Figure: Import settings for Excel datasets

5. Dataset creation:
   a. **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.
   b. **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.

   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.

For more information, see *Import Data Page*.

Import Google Sheets Data

Feature Availability: This feature may not be available in all product editions. For more information on available features, see *Compare Editions*.

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.

NOTE: Before you begin, you should review information on file formats supported for import, which can cause your files to fail to import or to be properly ingested and formatted. For more information, see *Supported File Formats*.

- 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.

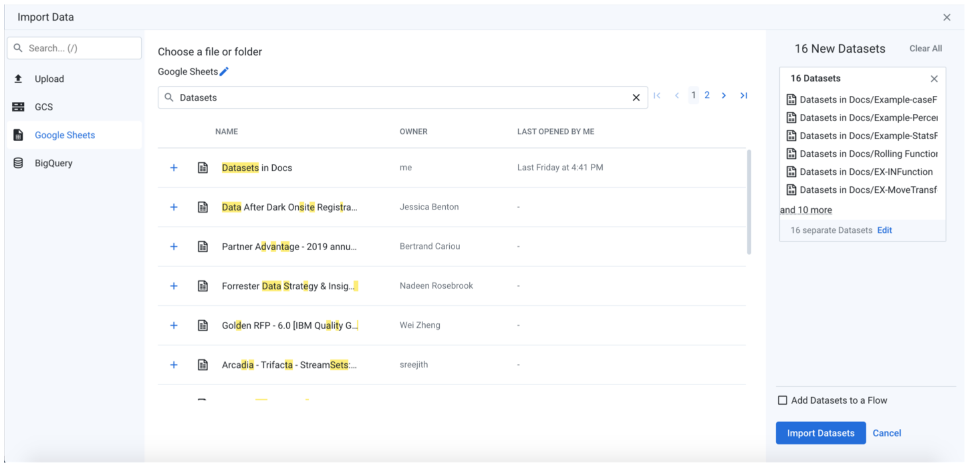
   > **NOTE:** When you first use the Google Sheets connector, you must enable Designer Cloud Powered by Trifacta Enterprise Edition to read all of your Google Drive data. When the connector is used, it locates only the Google Sheets data, including any Sheets that have been shared with you. All other data 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.



Datasets

NAME	OWNER	LAST OPENED BY ME
Datasets in Docs	me	Last Friday at 4:41 PM
Data After Dark Onsite Registra...	Jessica Benton	-
Partner Advantage - 2019 annu...	Bertrand Cariou	-
Forrester Data Strategy & Insig...	Nadeen Rosebrook	-
Golden RFP - 6.0 [IBM Quality G...	Wei Zheng	-
Arcadia - Trifacta - StreamSets:...	sreejith	-

16 Datasets
- Datasets in Docs/Example-caseF
- Datasets in Docs/Example-Percei
- Datasets in Docs/Example-StatsF
- Datasets in Docs/Rolling Functioı
- Datasets in Docs/EX-INFunction
- Datasets in Docs/EX-MoveTransf

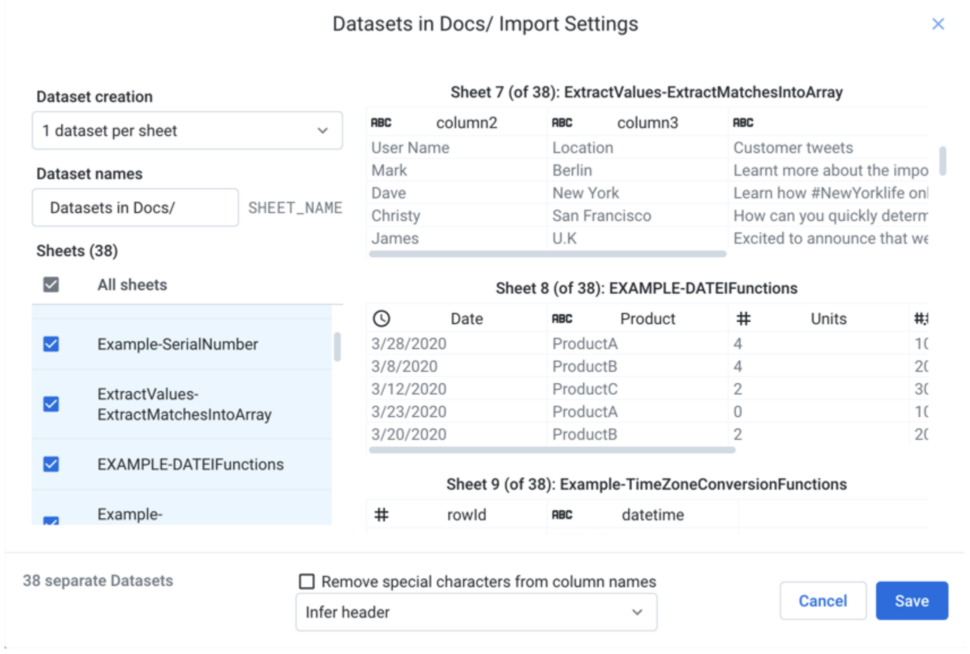
and 10 more

16 separate Datasets Edit

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

column2	column3	
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	Product	Units	
3/28/2020	ProductA	4	1(
3/8/2020	ProductB	4	2(
3/12/2020	ProductC	2	3(
3/23/2020	ProductA	0	1(
3/20/2020	ProductB	2	2(

Sheet 9 (of 38): Example-TimeZoneConversionFunctions

rowId	datetime

38 separate Datasets

- [ ] Remove special characters from column names

Infer header

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.

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.
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.

For more information, see *Import Data Page*.

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 De signer 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

Contents:

> **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.

Limitations

> **NOTE:** Before you begin, you should review information on file formats supported for import, which can cause your files to fail to import or to be properly ingested and formatted. For more information, see *Supported File Formats*.

- 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.

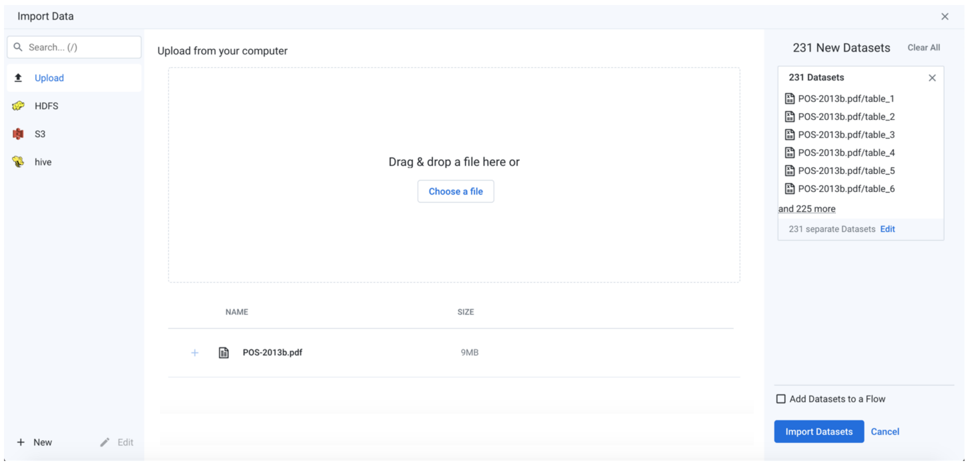
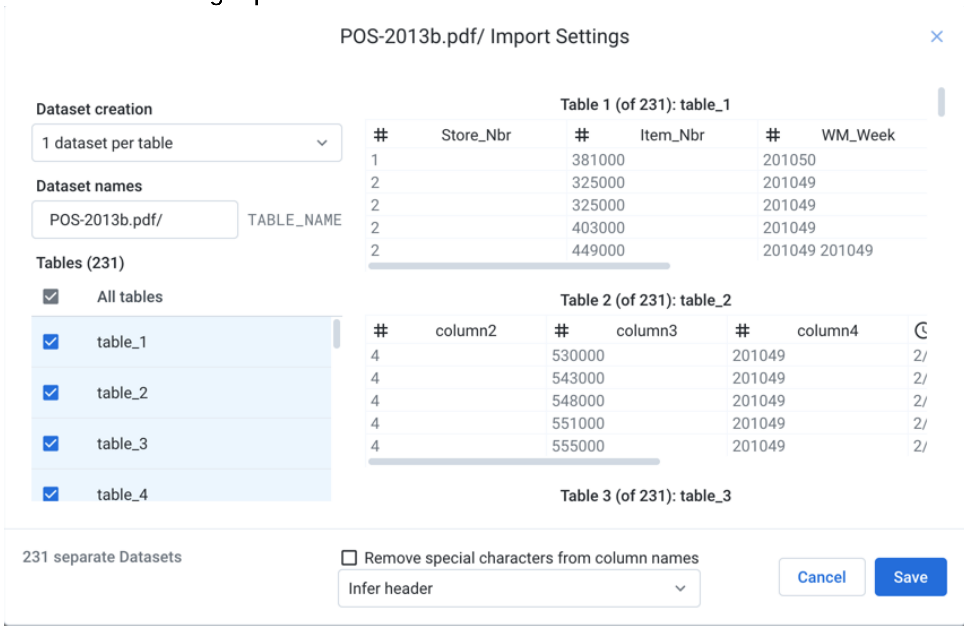


Figure: Import PDF file containing multiple pages

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

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

231 separate Datasets

☐ Remove special characters from column names

Infer header

Figure: Import settings for PDF datasets

5. Dataset creation:
    a. **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.
    b. **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.

    c. **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.

6. Selected tables:
    a. 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.

For more information, see *Import Data Page*.

Create Flow

You can create flows to organize your datasets, recipes, imported reference objects that you use to generate your results. You can create flows from the Flows page and Import Data page.

To create an end-to-end data pipeline, in your flow must create at least one of the following objects:

- **Imported dataset:** A reference to source data.
- **Recipe:** A set of transformation steps to apply to your source data.
- **Output:** A definition for how the output is to be generated, where it is to be located, and the output format.

Steps

1. From the menubar, click **Flows**.
2. In the Flows page, click **Create** > **Create Flow**. A new flow is created, with the name `Untitled - X`, where X is a number.

   > **Tip**: You can click the `Untitled - X` to enter a flow name and description.

3. **Imported Dataset:**
   a. In Flow View, click **Add Datasets**.
   b. From the Add datasets to flow dialog, you can add existing datasets or import new datasets.
      i. **Add datasets**:
         - You can browse or search for the datasets and add them to your flow.
         - To add a dataset, click the checkbox next to it.
      i. **Import datasets**: Click **import datasets** and add datasets from your local desktop or from the connected datastores.
      ii. For more information, see *Import Basics*.
   c. When finished, click **Add**.
4. **Recipe:**
   a. In Flow View, right-click the imported dataset and select **Add > Recipe**. A new empty recipe is created for the dataset.

      > **Tip:** You should rename your recipe as soon as you create it.

   b. To edit a recipe, select the recipe and click **Edit Recipe** in the right panel. The Transformer page is loaded with the initial data sample. For more information, see *Transform Basics*.
   c. To add a new step to your recipe:
      i. Click **Add New Step** in the recipe panel.
      ii. In the Search panel, you can search for the transformations and add the required transformations.
5. **Output:**
   a. In Flow View, right-click the recipe and select **Add Output**. An output is created on the flow canvas. For more information, see *Create Outputs*.
6. **Run a job:** A **job** executes the steps of the recipe on the imported dataset to generate the results, as defined by the output.
   a. To run a job on the output, right-click the output and select **Run**.

      > **Tip**: You can also click **Run** from the right side panel fo the Jobs panel.

b. In the Run Job page, specify the job that you wish to run. By default, an output object is pre-defined to generate a CSV output file. Click **Run**.
c. See *Run Jobs*.

Discovery Tasks

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

Explore Suggestions

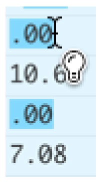
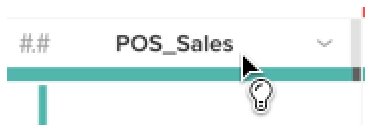
Contents:

When you make selections in the Transformer page, Designer Cloud Powered by Trifacta® Enterprise Edition res ponds 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.

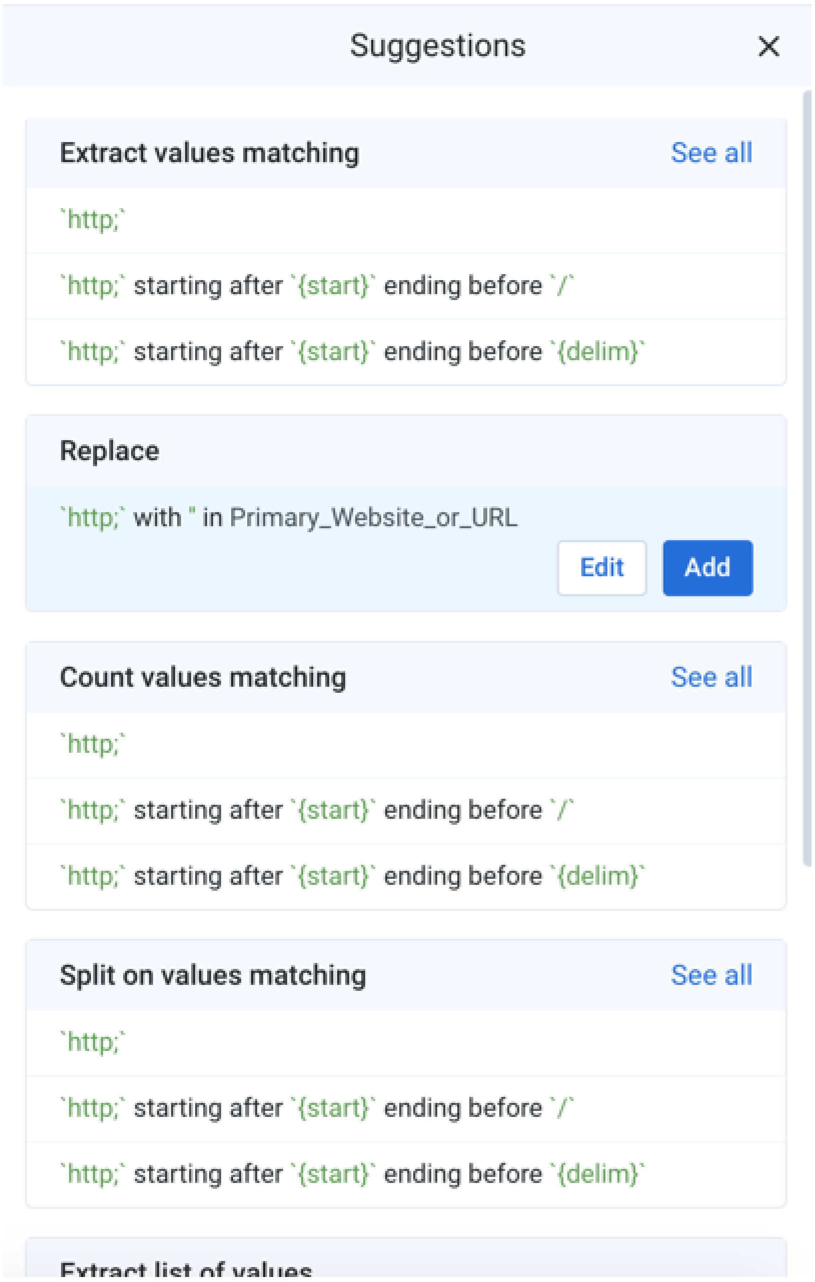
In the Column Browser or Column Details:

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

NOTE: Some complex transformations, such as joins and unions, cannot be suggested based on selection of values in the data grid.

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

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.

Decide on the Suggestion

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

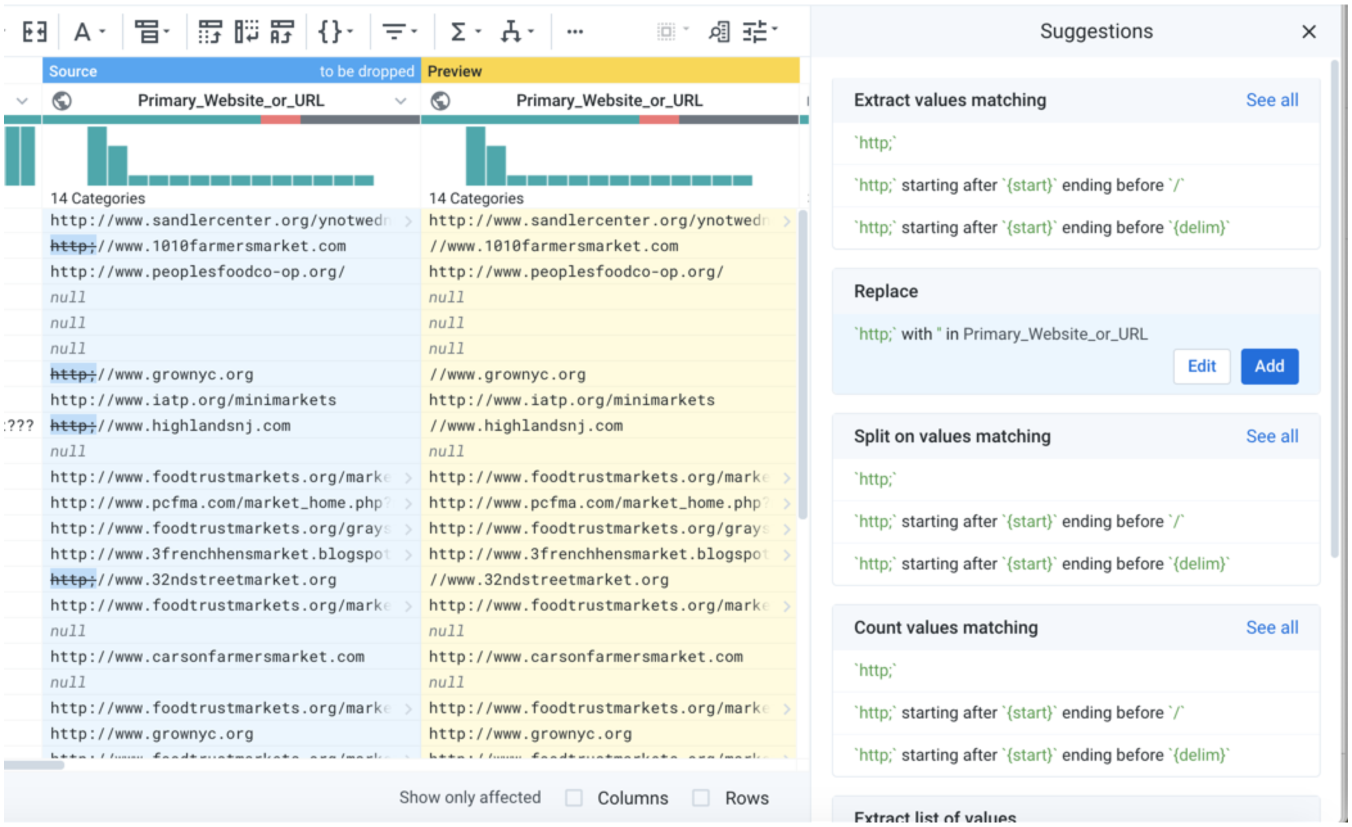
- **Select the suggestion to use.** After a suggestion is selected, the changes to the data are previewed in the Transformer page immediately. If there are multiple variants for the suggestion, verify that you are selecting the most appropriate one.
- **Select additional columns or values in the Transformer 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**. The Transform Builder is displayed, where you can edit the details of the transformation.

Previews

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



Source (to be dropped): Primary_Website_or_URL	Preview: Primary_Website_or_URL
14 Categories	14 Categories
http://www.sandlercenter.org/ynotwed	http://www.sandlercenter.org/ynotwed
~~http:~~//www.1010farmersmarket.com	//www.1010farmersmarket.com
http://www.peoplesfoodco-op.org/	http://www.peoplesfoodco-op.org/
null	null
null	null
null	null
~~http:~~//www.grownyc.org	//www.grownyc.org
http://www.iatp.org/minimarkets	http://www.iatp.org/minimarkets
~~http:~~//www.highlandsnj.com	//www.highlandsnj.com
null	null
http://www.foodtrustmarkets.org/marke	http://www.foodtrustmarkets.org/marke
http://www.pcfma.com/market_home.php	http://www.pcfma.com/market_home.php
http://www.foodtrustmarkets.org/grays	http://www.foodtrustmarkets.org/grays
http://www.3frenchhensmarket.blogspot	http://www.3frenchhensmarket.blogspot
~~http:~~//www.32ndstreetmarket.org	//www.32ndstreetmarket.org
http://www.foodtrustmarkets.org/marke	http://www.foodtrustmarkets.org/marke
null	null
http://www.carsonfarmersmarket.com	http://www.carsonfarmersmarket.com
null	null
http://www.foodtrustmarkets.org/marke	http://www.foodtrustmarkets.org/marke
http://www.grownyc.org	http://www.grownyc.org

Show only affected ☐ Columns ☐ Rows

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

Split on values matching — See all
- \`http:\`
- \`http:\` starting after \`{start}\` ending before \`/\`
- \`http:\` starting after \`{start}\` ending before \`{delim}\`

Count values matching — See all
- \`http:\`
- \`http:\` starting after \`{start}\` ending before \`/\`
- \`http:\` starting after \`{start}\` ending before \`{delim}\`

Extract list of values

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.
- 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.

Review Column Statistics

Contents:

Designer Cloud Powered by Trifacta® Enterprise Edition retains real-time statistics on the data in individual columns in the Transformer page and, when a job is executed, can deliver detailed statistics as part of the output.

Column Statistics in the Transformer Page

In the Transformer page, you can access statistics on individual columns through the data grid or Column Browser.

NOTE: All statistics in the Transformer page reference the currently displayed sample. If the sample is not the entire dataset, then these statistics may not accurately statistics of your generated results.

Statistics in the data grid

At the top of each column, you can see a visual representation of the data quality and histogram of values in the column.

- **Data quality bar:** This horizontal bar indicates the comparative volumes of valid values (green), invalid values (red), and missing values (black).
- **Column histogram:** You can see the distribution of values within the column, as indicated by the height of individual columns.

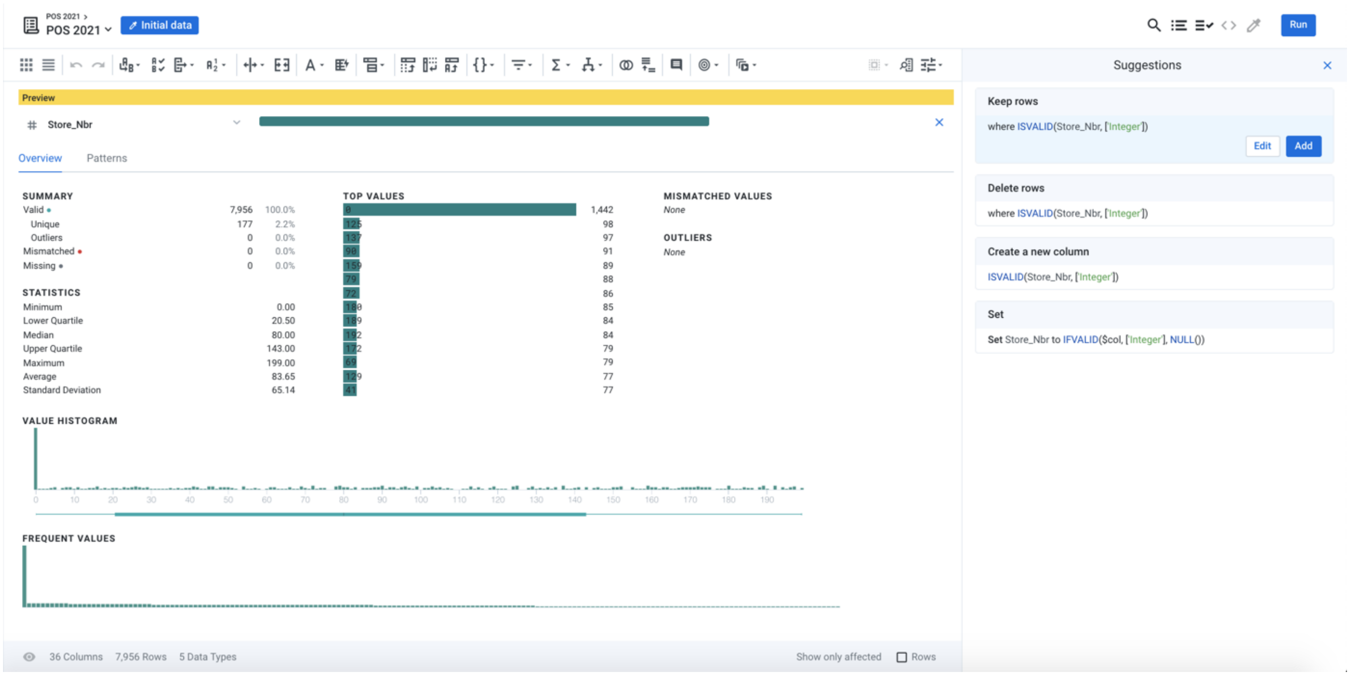
Tip: Mouse over any bar to see the number and percentage of a value in the sampled column.

Tip: You can select one or more of these bars to prompt for suggestions on how to transform the data.

For more information, see *Find Bad Data*.

Statistics in Column Details

In the Transformer toolbar, select a column. From the column menu, select **Details**. In Column Details, you can explore additional statistics derived from the values in the column for the currently selected sample.



SUMMARY		
Valid	7,956	100.0%
Unique	177	2.2%
Outliers	0	0.0%
Mismatched	0	0.0%
Missing	0	0.0%

STATISTICS	
Minimum	0.00
Lower Quartile	20.50
Median	80.00
Upper Quartile	143.00
Maximum	199.00
Average	83.65
Standard Deviation	65.14

TOP VALUES	
0	1,442
125	98
137	97
98	91
159	89
79	88
72	86
180	85
189	84
192	84
172	79
69	79
129	77
41	77

MISMATCHED VALUES
None

OUTLIERS
None

Keep rows
where ISVALID(Store_Nbr, ['Integer'])

Delete rows
where ISVALID(Store_Nbr, ['Integer'])

Create a new column
ISVALID(Store_Nbr, ['Integer'])

Set
Set Store_Nbr to IFVALID($col, ['Integer'], NULL())

Figure: Column Details panel

The statistics displayed in the Column Details vary depending on the column data type. For example, numeric metrics appear for Integer and Decimal columns, and fewer of them appear for columns of String type.

Tip: Selecting one or more bars in the Column Details panel prompts for a set of suggested transformations.

Tip: You may wish to explore the Patterns tab where you can review patterns in the data and to make transformations based on them.

Column Statistics in Transformations

You can also generate statistics for your dataset at any point time during recipe development.

NOTE: The values computed by these metrics in the Transformer page apply only to the displayed sample. Values are computed across the entire dataset when a job is run.

Flat Aggregations

As a transformation step, you can perform custom aggregated calculations on your dataset. For example, you can compute the average of all values in a column and add that as a new column.

Aggregations

You can also create aggregations that group computations by specified value. These computations provide deeper insights into your data.

Tip: For exploratory calculations, you might choose to create a secondary recipe after your current one. In the secondary recipe, you perform these calculations, allowing you to continue to work in the first one on performing your primary transformations.

For more information, see *Create Aggregations*.

Column Statistics in Results

When you generate results, you can choose to generate a visual profile of your output data, which is delivered alongside the results that you specify.

- A **visual profile** is visual representation of column and dataset statistics on the output.
  - Visual profiles can provide clues to continuing issues in the data and to overall trends.
  - Visual profiles are available in the application and can be downloaded in PDF or JSON format.
- When enabled, a visual profile job is executed after the results have been generated.

To generate a visual profile, click the **Profiling** checkbox in the Run Job page.

When the job is complete, your visual profile is available in the Profile tab of the Job Details page.

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

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.

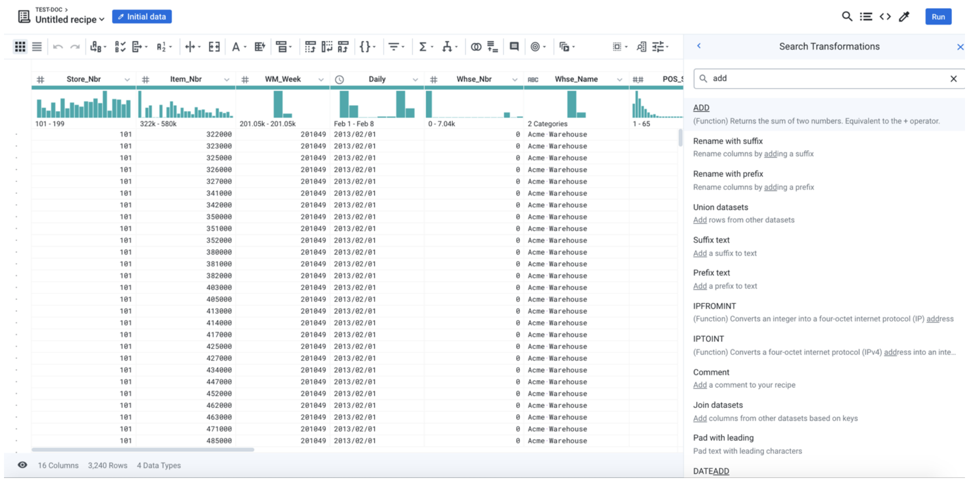
Steps:

To edit the recipe:

1. Select the recipe, and click **Edit Recipe** in the right panel.
2. The Transformer page is loaded. The initial data sample is loaded in the data grid. This sample comes from the first rows of the first file or table in the dataset.
3. On the right side of the screen is the empty recipe panel. You have no steps in your recipe. You can add recipe steps in any of the following ways:
4. Select something:
    a. You can highlight a cell value, a column, a row, or even part of a cell value, and Designer Cloud Powered by Trifacta® Enterprise Edition provides a list of suggestions in the panel to the right. For more information, see *Transform Basics*.
5. Search for a function or transformation:
    a. If the recipe panel is not displayed on the right side of the screen, click the Recipe icon in the menubar at the top of the screen.
    b. In the panel on the right side of the Transformer page, click **Add New Step**. The Search transformation panel is displayed.

    > **Tip**: Press `CTRL` / `COMMAND + K` to add a new step.

    c. In the Search panel, you can search for the transformations. Enter the search term, the corresponding transformation or function is underlined in the search panel. The following example shows a search for **add**.



Store_Nbr	Item_Nbr	WM_Week	Daily	Whse_Nbr	Whse_Name	POS_
101 - 199	322k - 580k	201.05k - 201.05k	Feb 1 - Feb 8	0 - 7.04k	2 Categories	1 - 65
101	322000	201049	2013/02/01	0	Acme Warehouse	
101	323000	201049	2013/02/01	0	Acme Warehouse	
101	325000	201049	2013/02/01	0	Acme Warehouse	
101	326000	201049	2013/02/01	0	Acme Warehouse	
101	327000	201049	2013/02/01	0	Acme Warehouse	
101	341000	201049	2013/02/01	0	Acme Warehouse	
101	342000	201049	2013/02/01	0	Acme Warehouse	
101	350000	201049	2013/02/01	0	Acme Warehouse	
101	351000	201049	2013/02/01	0	Acme Warehouse	
101	352000	201049	2013/02/01	0	Acme Warehouse	
101	380000	201049	2013/02/01	0	Acme Warehouse	
101	381000	201049	2013/02/01	0	Acme Warehouse	
101	382000	201049	2013/02/01	0	Acme Warehouse	
101	403000	201049	2013/02/01	0	Acme Warehouse	
101	405000	201049	2013/02/01	0	Acme Warehouse	
101	413000	201049	2013/02/01	0	Acme Warehouse	
101	414000	201049	2013/02/01	0	Acme Warehouse	
101	417000	201049	2013/02/01	0	Acme Warehouse	
101	425000	201049	2013/02/01	0	Acme Warehouse	
101	427000	201049	2013/02/01	0	Acme Warehouse	
101	434000	201049	2013/02/01	0	Acme Warehouse	
101	447000	201049	2013/02/01	0	Acme Warehouse	
101	452000	201049	2013/02/01	0	Acme Warehouse	
101	462000	201049	2013/02/01	0	Acme Warehouse	
101	463000	201049	2013/02/01	0	Acme Warehouse	
101	471000	201049	2013/02/01	0	Acme Warehouse	
101	485000	201049	2013/02/01	0	Acme Warehouse	

16 Columns 3,240 Rows 4 Data Types

add

ADD
(Function) Returns the sum of two numbers. Equivalent to the + operator.

Rename with suffix
Rename columns by adding a suffix

Rename with prefix
Rename columns by adding a prefix

Union datasets
Add rows from other datasets

Suffix text
Add a suffix to text

Prefix text
Add a prefix to text

IPFROMINT
(Function) Converts an integer into a four-octet internet protocol (IP) address

IPTOINT
(Function) Converts a four-octet internet protocol (IPv4) address into an inte...

Comment
Add a comment to your recipe

Join datasets
Add columns from other datasets based on keys

Pad with leading
Pad text with leading characters

DATEADD

Figure: Add transformation

> **Tip**: To see a list of all available functions, enter `function` . The list of available functions is displayed.

    d. After you select the required transformation or function, the Transform Builder is displayed with the pre-specified function.
6. Edit the recipe directly:

a. After you have added a few steps and are comfortable with the process, you can begin to explore editing the recipe directly.

Filter Data

Contents:

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**.
- 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.

- You can also hide one or more columns through the Column Browser:
  - In the Transformer page, click the Columns icon in the toolbar.
  - In the Column Browser, select the column or columns to hide. From the Actions drop-down, select **E dit > Hide**.
  - 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 Filter panel:

- **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.

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.

Locate Outliers

Contents:

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.
- Bad data.
- Poorly formatted data
- Mismeasured data
- Data that skews statistics

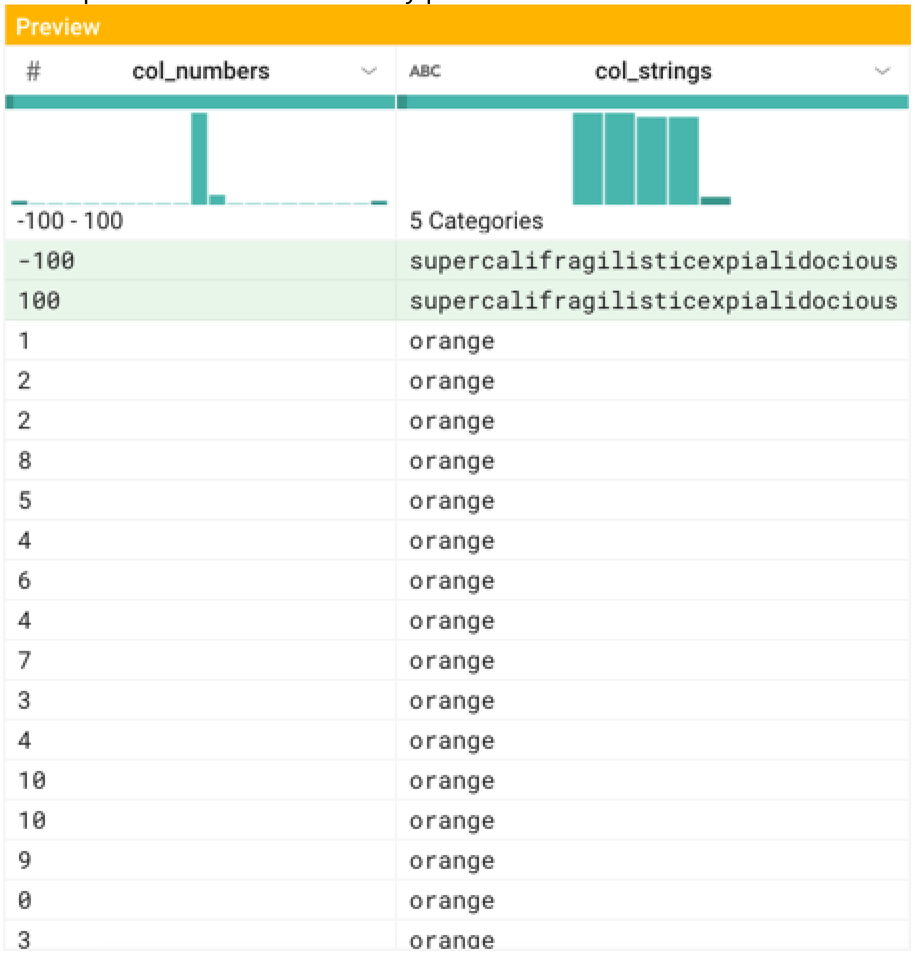
This section provides guidance in how to locate these patterns of data in individual columns.

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.



col_numbers	col_strings
-100	supercalifragilisticexpialidocious
100	supercalifragilisticexpialidocious
1	orange
2	orange
2	orange
8	orange
5	orange
4	orange
6	orange
4	orange
7	orange
3	orange
4	orange
10	orange
10	orange
9	orange
0	orange
3	orange

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.

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`	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.
`2147483647`	This value is the largest positive integer that can be stored in an `int` 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`	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. • This limit is generally known as the "Year 2038" problem.

String data

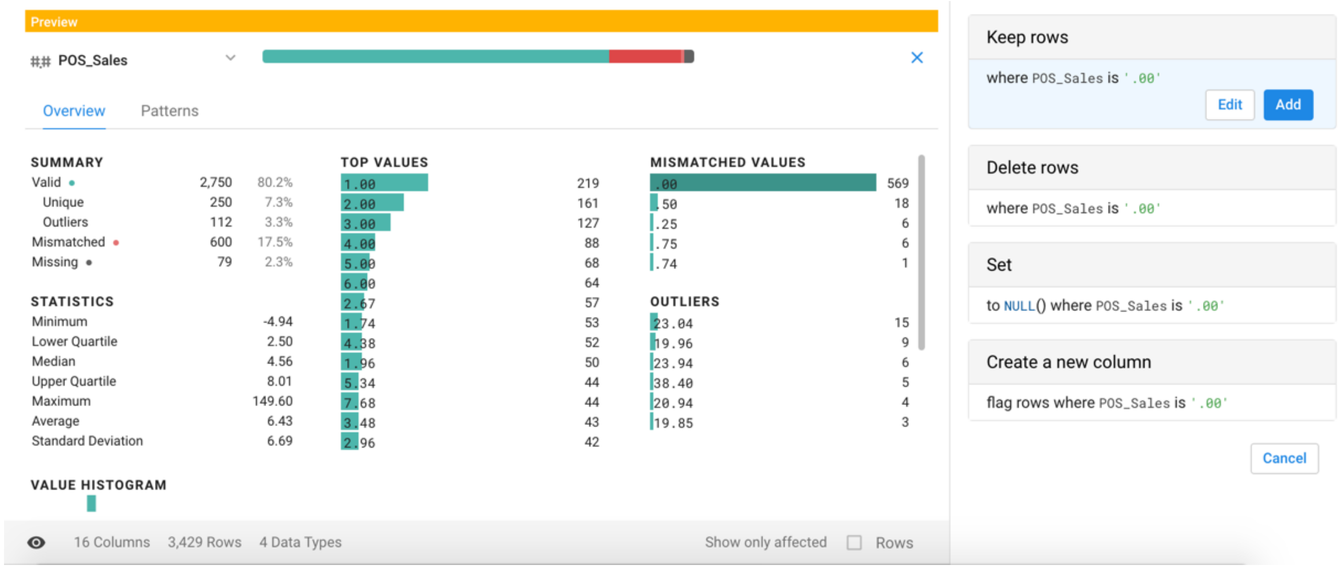
The `col-strings` column contains approximately 25 values for `orange`, `red`, `green`, `yellow`, and two instances of `supercalifragilisticexpialidocious`.

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.


Preview

POS_Sales

Overview | Patterns

SUMMARY

Valid ●	2,750	80.2%
Unique	250	7.3%
Outliers	112	3.3%
Mismatched ●	600	17.5%
Missing ●	79	2.3%

STATISTICS

Minimum	-4.94
Lower Quartile	2.50
Median	4.56
Upper Quartile	8.01
Maximum	149.60
Average	6.43
Standard Deviation	6.69

VALUE HISTOGRAM

TOP VALUES

1.00	219
2.00	161
3.00	127
4.00	88
5.00	68
6.00	64
2.67	57
1.74	53
4.38	52
1.96	50
5.34	44
7.68	44
3.48	43
2.96	42

MISMATCHED VALUES

.00	569
.50	18
.25	6
.75	6
.74	1

OUTLIERS

23.04	15
19.96	9
23.94	6
38.40	5
20.94	4
19.85	3

16 Columns 3,429 Rows 4 Data Types Show only affected ☐ Rows

Keep rows
where POS_Sales is '.00'
Edit Add

Delete rows
where POS_Sales is '.00'

Set
to NULL() where POS_Sales is '.00'

Create a new column
flag rows where POS_Sales is '.00'

Cancel

Figure: Outliers in the Column Details

Column Detail Statistics

The Column Details panel 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 panel can be selected to prompt for suggestions on actions, including values in Outliers, Value Histogram, and Frequent Values graphs. Multi-select values as needed.

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	`(col_numbers - AVERAGE(col_numbers)) / STDEV`

	(col_numbers)

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	`ABS(col_numbers_stdev)> 4`

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	`col_numbers`
Parameter: Formula	`IF((col_numbers == 'false'),'0', col_numbers)`

Transformation Name	`Edit column with formula`
Parameter: Columns	`col_numbers`
Parameter: Formula	`IF( (col_numbers == 'true'),'1',col_numbers)`

- **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	`col_numbers < 25`
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	`col_numbers`
Parameter: Formula	`IF((col_numbers < 25), AVERAGE(col_numbers), col_numbers)`

Compute Counts

Contents:

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:

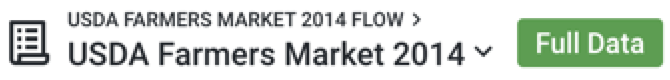


Figure: Data grid sample is the full dataset.

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.

Computed row counts

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

Function Name	Description
COUNT Function	Generates the count of rows in the dataset. Generated value is of Integer type. **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.
ROWNUMBER Function	Generates a new column containing the row number as sorted by the `order` parameter and optionally grouped by the `group` parameter.
SOURCEROWNUMBER Function	Returns the row number of the current row as it appeared in the original source dataset before any steps had been applied. **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. **Tip:** You can pair this function later with the `MIN` or `MAX` functions to compute the highest and lowest row number information.

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	`SUM(tweetCompanyReferences)`
Parameter: New column name	`sum_tweetCompanyReferences`

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	`countURLs`

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

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:

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**
LEFT Function	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.
RIGHT Function	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.
FIND Function	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.
RIGHTFIND Function	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.
SUBSTRING Function	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	`custCodeStatus`
Parameter: Formula	`` IF(STARTSWITH(custCode,`{digit}{6}`), 'error',custCodeStatus) ``

Function Name	Description
STARTSWITH Function	Returns `true` if the leftmost set of characters of a column of values matches a pattern. The source value can be any data type, and the pattern can be a Pattern , regular expression, or a string.
ENDSWITH Function	Returns `true` if the rightmost set of characters of a column of values matches a pattern. The source value can be any data type, and the pattern can be a Pattern , regular expression, or a string.

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	`IF(EXACT(LOWER(REMOVEWHITESPACE(colA)))==EXACT(LOWER(REMOVEWHITESPACE(colB))),'true','false')`
Parameter: New column name	`matches`

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	`DOUBLEMETAPHONEEQUALS(colA,colB,'strong')`
Parameter: New column name	`matches`

- 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 steps 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`, `TestScore1`) 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 from	`key`
Parameter: Option	`Custom text or pattern`
Parameter: Text to extract	``` `{digit}` ```

5. The `key2` column contains just the numeric data now. Rename this column to `TestNumber`. You can delete the `key` column now.
6. 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:

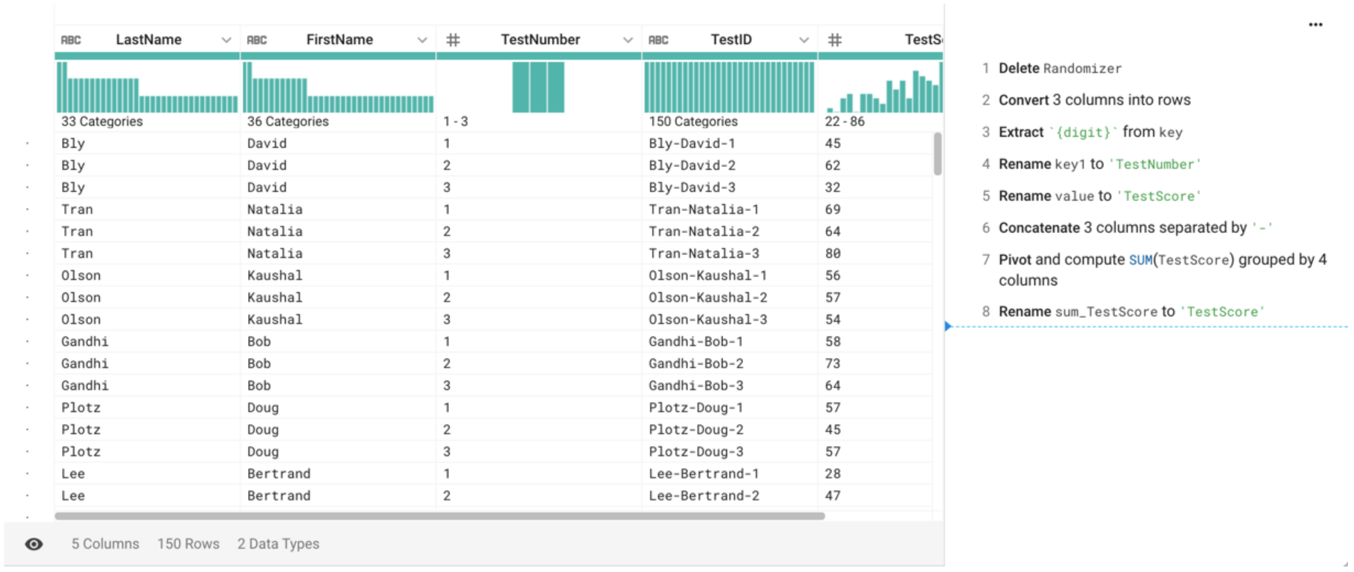
Transformation Name	`Merge columns`
Parameter: Columns	`LastName,FirstName,TestNumber`
Parameter: Separator	`'-'`

7. 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.
8. 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 the change the `SUM` function to `AVERAGE`. In the results, you have one average for each test taker.

9. You may want to rename the aggregation column. Your final dataset should look like the following:



LastName	FirstName	TestNumber	TestID	TestS
33 Categories	36 Categories	1 - 3	150 Categories	22 - 86
Bly	David	1	Bly-David-1	45
Bly	David	2	Bly-David-2	62
Bly	David	3	Bly-David-3	32
Tran	Natalia	1	Tran-Natalia-1	69
Tran	Natalia	2	Tran-Natalia-2	64
Tran	Natalia	3	Tran-Natalia-3	80
Olson	Kaushal	1	Olson-Kaushal-1	56
Olson	Kaushal	2	Olson-Kaushal-2	57
Olson	Kaushal	3	Olson-Kaushal-3	54
Gandhi	Bob	1	Gandhi-Bob-1	58
Gandhi	Bob	2	Gandhi-Bob-2	73
Gandhi	Bob	3	Gandhi-Bob-3	64
Plotz	Doug	1	Plotz-Doug-1	57
Plotz	Doug	2	Plotz-Doug-2	45
Plotz	Doug	3	Plotz-Doug-3	57
Lee	Bertrand	1	Lee-Bertrand-1	28
Lee	Bertrand	2	Lee-Bertrand-2	47

5 Columns 150 Rows 2 Data Types

1 **Delete** `Randomizer`
2 **Convert** 3 columns into rows
3 **Extract** ``` `{digit}` ``` from `key`
4 **Rename** `key1` to `'TestNumber'`
5 **Rename** `value` to `'TestScore'`
6 **Concatenate** 3 columns separated by `'-'`
7 **Pivot** and compute `SUM(TestScore)` grouped by 4 columns
8 **Rename** `sum_TestScore` to `'TestScore'`

Figure: Single column of test scores

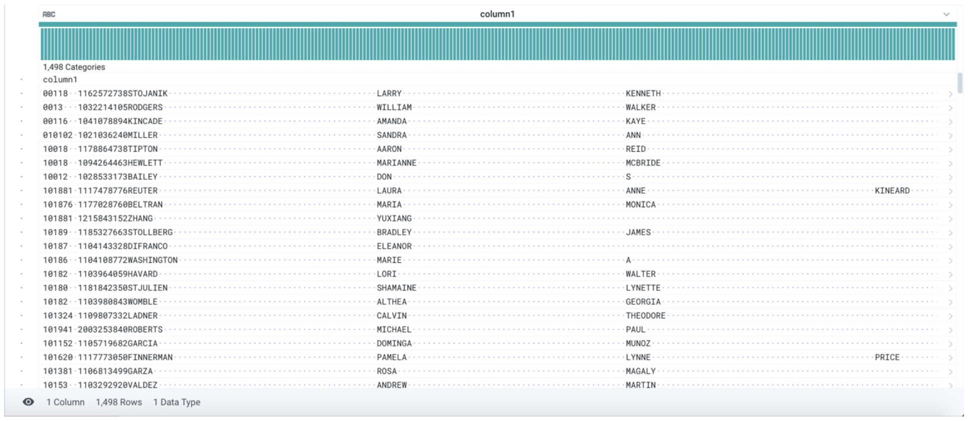
Now that your columns of data have been consolidated to a single column, you can use the single-column transforms and functions to perform analysis, such as locating outliers.

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:



```
column1
00118  1162572738STOJANIK          LARRY         KENNETH
0013   1032214105RODGERS           WILLIAM       WALKER
00116  1041078894KINCADE           AMANDA        KAYE
010102 1021036240MILLER            SANDRA        ANN
10018  1178864738TIPTON            AARON         REID
10018  1094264463HEWLETT           MARIANNE      MCBRIDE
10012  1028533173BAILEY            DON           S
101881 1117478776REUTER            LAURA         ANNE          KINEARD
101876 1177028760BELTRAN           MARIA         MONICA
101881 1215843152ZHANG             YUXIANG
10189  1185327663STOLLBERG         BRADLEY       JAMES
10187  1104143328DIFRANCO          ELEANOR
10186  1104108772WASHINGTON        MARIE         A
10182  1103964059HAVARD            LORI          WALTER
10180  1181842350STJULIEN          SHAMAINE      LYNETTE
10182  1103980843WOMBLE            ALTHEA        GEORGIA
101324 1109807332LADNER            CALVIN        THEODORE
101941 2003253840ROBERTS           MICHAEL       PAUL
101152 1105719682GARCIA            DOMINGA       MUNOZ
101620 1117773050FINNERMAN         PAMELA        LYNNE         PRICE
101381 1106813499GARZA             ROSA          MAGALY
10153  1103292920VALDEZ            ANDREW        MARTIN
```

1 Column   1,498 Rows   1 Data Type

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 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.

The initial data sample is 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 default, the initial sample is the first 10 MB of your dataset.
    - The size of the sample can be modified by an administrator.
    - For file-based sources, the initial sample is taken from a limited number of files.
        - By default, this limit is set to `10` files.
        - The maximum number of files from which a sample can be generated can be defined by an administrator.
- If your dataset is less than 10 MB, then the entire dataset may be 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**. For more information on how this special sampling type is generated, see *Overview of Sampling*.

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 Anamoly-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.

Change Sample Size

If you are encountering low-memory conditions related to sampling or wish to improve the performance of the sampling process, you can adjust the size of the samples that are displayed in the browser for your current recipe. For more information, see *Change Recipe Sample Size*.

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.
- When a new sample is generated, sort transformations are not preserved for some type of outputs. Sort transformations must be reapplied. See *Sort Rows*.
- When executed on the Trifacta Photon running environment, samples taken from a dataset with parameters are limited to a maximum of 50 files.

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 sample. A sample job ID is generated for each sample you collect. When the sample is available, the Load Sample message is displayed in the Transformer page.
5. To load the sample, click **Load Sample**.

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.

Cancel 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.

Load 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 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.

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.

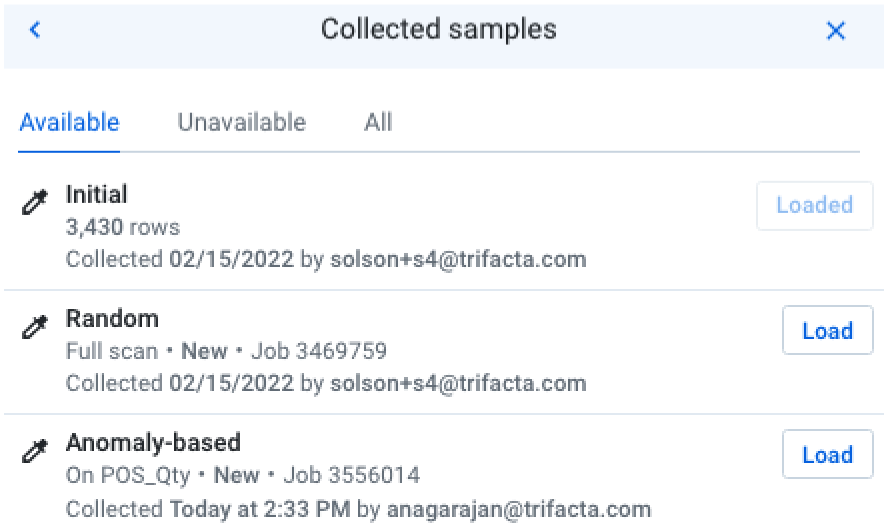
After the sample becomes invalid, the Transformer page reverts to the recently collected sample that is valid.

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.



Initial
3,430 rows
Collected 02/15/2022 by solson+s4@trifacta.com

Random
Full scan • New • Job 3469759
Collected 02/15/2022 by solson+s4@trifacta.com

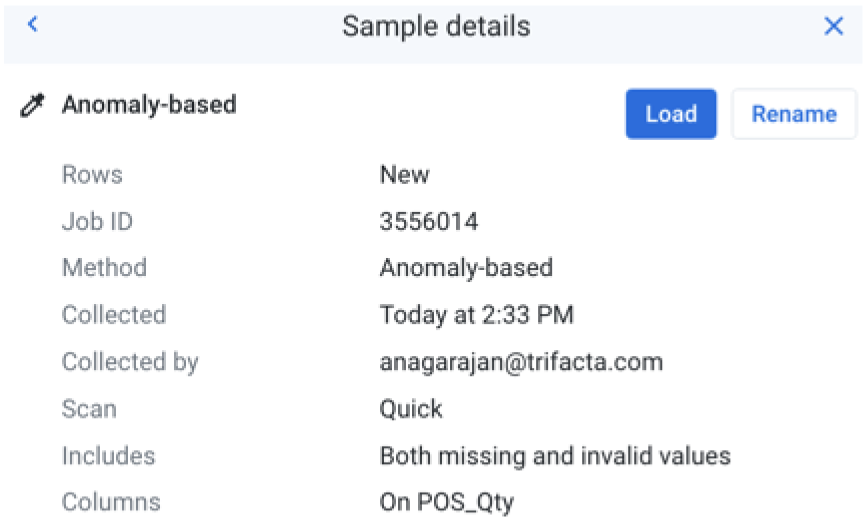
Anomaly-based
On POS_Qty • New • Job 3556014
Collected Today at 2:33 PM by anagarajan@trifacta.com

Figure: Collected samples

The collected samples contain the following tabs:

- **Available**: Displays the available samples that can be used. You can click **Load** to load the required sample.
- **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.

You can click the sample name to view the sample details.



Anomaly-based

Rows	New
Job ID	3556014
Method	Anomaly-based
Collected	Today at 2:33 PM
Collected by	anagarajan@trifacta.com
Scan	Quick
Includes	Both missing and invalid values
Columns	On POS_Qty

Figure: Sample details

- **Load**: Click **Load** to load the sample.
- **Rename**: Click **Rename** to rename the sample

Review Sample Jobs

You can review and manage all of your samples like transformation jobs. 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*.

Change Recipe Sample Size

By default, samples displayed in the Designer Cloud® application can be up to 10 MB in size. In some cases, you may want to increase or decrease the size of samples displayed in the browser to include more data or to prevent browser or Trifacta Photon memory issues.

> **NOTE:** Samples are still generated using the preset limit and stored in full on the base storage layer. This setting changes the volume of the data delivered to the browser.

> **NOTE:** Trifacta administrators can increase the maximum size of the Trifacta Photon samples up to 40 MB. For more information, see *Configure Application Limits*.

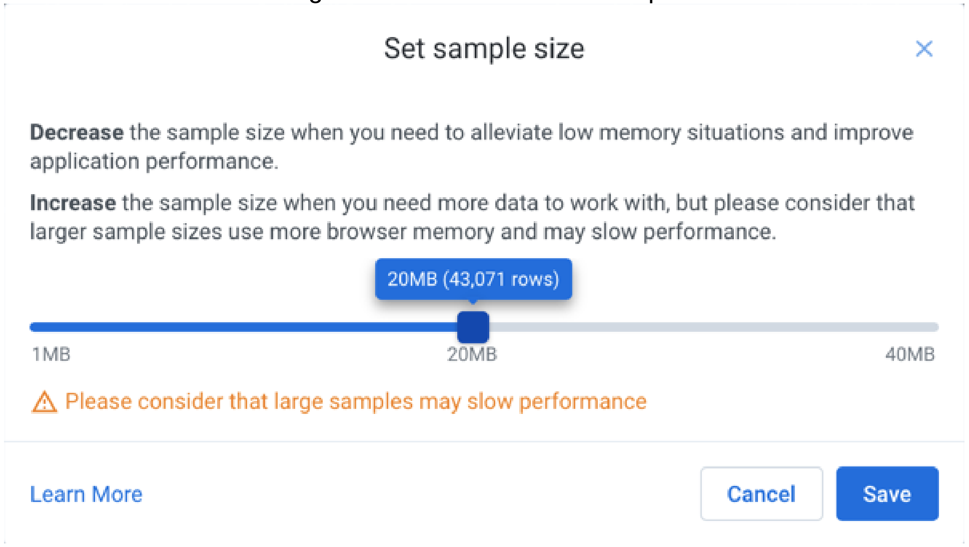
This size reflects the maximum permitted size of a sample that is delivered to the browser. If the available data is less than the maximum permitted size, then the actual sample size may be smaller.

- When this setting is reduced, the sample currently loaded in the browser is immediately reduced to the new maximum size. Other available samples for the recipe are resized when they are reloaded.
- If you raise the maximum sample size, the volume of the currently loaded sample can be expanded to the maximum size, if the data is available in the sample.

> **NOTE:** Maximum sample sizes are configured on a per-recipe basis. Changes to your sample size affect sample sizes in downstream recipes. For example, if the maximum sample size on Recipe A is set to 5MB, any sample from Recipe A that is used in Recipe B, which is downstream of Recipe A, is also constrained to 5MB in maximum size. However, Recipe B may have a different maximum sample size, so you can generate a new sample in Recipe B to acquire a different-sized sample.

Steps:

1. In the Transformer page, click the name of the sample in the top menu.
2. The Sample Indicator is displayed
3. In the Sample Indicator, click **Edit**.
4. Use the slider bar to change the maximum size of samples delivered to the browser for this recipe:



Set sample size

Decrease the sample size when you need to alleviate low memory situations and improve application performance.

Increase the sample size when you need more data to work with, but please consider that larger sample sizes use more browser memory and may slow performance.

20MB (43,071 rows)

1MB 20MB 40MB

Please consider that large samples may slow performance

Learn More Cancel Save

Figure: Set sample size

> **NOTE**: The range of the slider indicates that maximum available data for the sample. For example, if the sample is 7 MB then the slider shows a maximum of 7 MB.

NOTE: A warning message may displayed if the sample size exceeds the recommended size.

NOTE: For datasources that are uncompressed or converted when ingested to the backend, the actual storage size may exceed the specified maximum limit.

Tip: The Designer Cloud application provides a recommendation for the new sample size. You should set your sample size to this value or a smaller value.

5. To apply your changes, click **Save**.
6. The current sample is immediately updated to reflect the new maximum sample size. The size of all subsequent samples that are delivered to the browser for this recipe are capped at this new maximum size. Samples are still generated at the preset size in backend storage.

For more information, see *Sample Indicator*.

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. In the Import Data page, create an imported dataset from your source. Add it to a flow.
2. In Flow View, create a recipe for your imported dataset.
3. In Flow View, edit the newly created recipe. It is opened in the 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 the Profile tab in the Job Details page.
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 can download the profile and output for review.

For more information, 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 to create 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.

Validate Your Data

Contents:

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.

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. New samples can be generated through the 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 Transformer page to add steps to your recipe.

    > **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	`ismismatched(Primary_Website_or_URL, ['Url'])`
Parameter: New column name	`mismatched_Primary_Website_or_URL`

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`.

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	`Primary_Website_or_URL`
Parameter: Data type	`'Url'`

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.

Available statistics depend on the data type for the column.

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.

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.

You can perform ad-hoc tests for uniqueness of individual values.

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	`MATCHES(MarketName, /^[a-zA-Z0-9 ]*$/)`

You can add additional permitted characters inside the square brackets. For more information, see *Text Matching*.

Data quality rule:

This data quality performs the same test as the above transformation:

Data Quality Rule	`Match`
Parameter: Column	`MarketName`
Parameter: Matches pattern	`/^[a-zA-Z0-9 ]*$/`

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.

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.

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 a detailed example, see *Validate Column Values against a Dataset*.

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.

Validate Column Values against a Dataset

Contents:

When needed, you can validate a column of values against a pre-defined set of values maintained in a separate dataset. This method of data validation is most useful for String-based data that does not easily map to a specific pattern of values.

Tip: This method is a suitable replacement for custom data types maintained using a dictionary file.

Overview

This method of validation is completed through the following general steps:

1. **Prepare your validation dataset.** Create a dataset containing the unique values against which you wish to validate.
2. **Import your validation dataset.** After you have prepared the dataset externally, you should import it into the Designer Cloud application .
3. **Join your data to your validation dataset.** You perform a join from the dataset you're wrangling to the validation dataset that you imported. Validation errors should be identifiable as missing values in the validation column.
4. **Triage defects as necessary.** For rows that cannot be resolved, additional wrangling may be necessary.
5. **Standardize data.** You can use the Standardize tool to review the differences between invalid data and valid data.

Example

This approach is best demonstrated by example. Below, you can see a set of orders for product.

productName	customerName	Qty	totalSales
Product ADA	Customer ABC	2	26
Product AEV	Customer DEF	4	100
Product DXL	Customer EFG	6	42
Product EDM	Customer ABC	1	26
Product JTO	Customer DEF	3	75
Product JUB	Customer EFG	5	35
Product NRS	Customer ABC	6	26
product NSE	Customer DEF	8	200
Product ZZZ	Customer EFG	10	80

Notes:

- You can see that this set of orders is spread across 10 different products for three different customers.
- In the productName column, there is a mismatch in capitalization.
- The final productName value (`Product ZZZ`) does not exist.

The product names in this list must be validated against a dataset containing the list of all available products. This list is 100 product names long.

You can use the links below to download these datasets as CSV files for exploration in your project or workspace.

- *Dataset-ProductNames.csv*
- *Dataset-ProductNames-Orders.csv*

Prepare Validation Dataset

If you haven't done so already, you should prepare your validation dataset for use in the Designer Cloud application . Below, you can see the first 10 rows of the ProductNames dataset:

productName
Product ADA
Product AEV
Product ANH
Product ARA
Product ARM
Product AUJ
Product BAD
Product BAP
Product BEI
Product BEZ

Notes:

- A column header is provided in the dataset. This is helpful for identifying the column to use later as the join key.
- You may wish to enter a validation column, simply contains the value `TRUE`.

  Tip: If your dataset does not contain this column, you can create a new formula within the Transform Builder to insert this value. This step is covered later.

Import Validation Dataset

If you have prepared your dataset, you must import into in the application.

Steps:

1. In the Designer Cloud application , click **Library**.
2. Click **Import Data**.
3. Navigate to the file or files to import.

Tip: If you are using the example files, you can right-click them above, download them to your desktop, and then drag and drop them into the Import Data page.

4. Import the file as a new dataset.

   Tip: If you are using the example datasets, you can call it `Reference-ProductNames`.

5. It may be helpful to import the file into a new file and create a recipe from it.

For the example dataset, there is a single column of values. To make this dataset useful as a validation dataset, add the following transformation, which adds a second column called `validation` containing the value `true` for each row.

Transformation Name	`New formula`
Parameter: Formula type	`Single row formula`
Parameter: Formula	`'true'`
Parameter: New column name	`validation`

The example reference dataset now looks like the following:

productName	validation
Product ADA	TRUE
Product AEV	TRUE
Product ANH	TRUE
Product ARA	TRUE
Product ARM	TRUE
Product AUJ	TRUE
Product BAD	TRUE
Product BAP	TRUE
Product BEI	TRUE
Product BEZ	TRUE

Join with Validation Dataset

Now you can join your existing dataset with the new validation dataset. A **join** performs a comparison of the column values in one dataset compared to the column values in another dataset. Where matches are detected, columns and values of the joined-in dataset are inserted into the source dataset. For more information on joins, see *Join Types*.

Steps:

1. Edit the recipe of the source dataset.
2. As a new step for the recipe, enter `join datasets` in the Search panel.
3. Select the source of the joined-in data:
   a. If you created a recipe and added steps (as in the example), then click the Recipes in current flow tab.
   b. If you imported a clean dataset, then click one of the Datasets tabs.

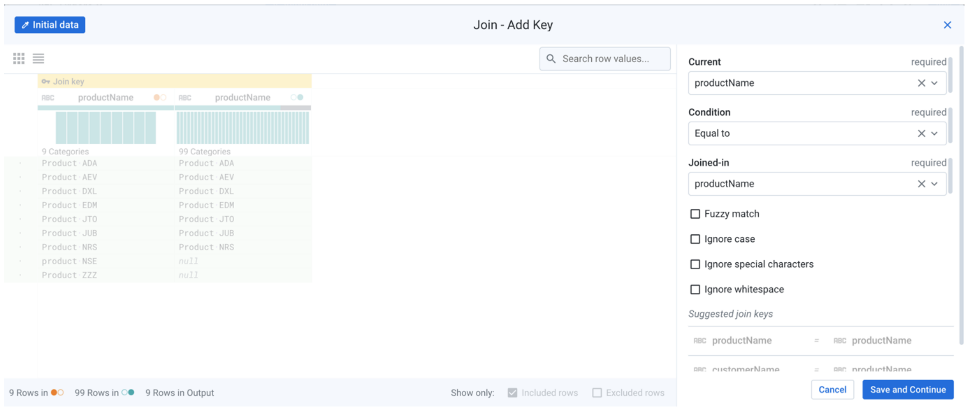
4. For Join type, select **Left**.

> **Tip:** A left join includes all rows from the left (source) dataset and only the matching rows from the right dataset for a specified set of column values (**join keys**) in the left dataset. If a column value in the left dataset does not exist in the right dataset, then null values are listed for that row's entry for all columns imported from the right dataset.

5. For the Join keys, select the column containing values to check from the left (source) dataset and then column containing the reference values in the right dataset.

> **Tip:** In the example datasets, both of these columns are called `ProductNames`, which assists the join tool in identifying the join key columns.

6. Under Join Keys, hover over one of the column names. Then, click the Pencil icon.



productName	productName
9 Categories	99 Categories
Product·ADA	Product·ADA
Product·AEV	Product·AEV
Product·DXL	Product·DXL
Product·EDM	Product·EDM
Product·JTO	Product·JTO
Product·JUB	Product·JUB
Product·NRS	Product·NRS
product·NSE	null
Product·ZZZ	null

Current (required): productName
Condition (required): Equal to
Joined-in (required): productName
☐ Fuzzy match
☐ Ignore case
☐ Ignore special characters
☐ Ignore whitespace

Suggested join keys: productName = productName; customerName = productName

9 Rows in ●○ 99 Rows in ○● 9 Rows in Output
Show only: ✓ Included rows ☐ Excluded rows

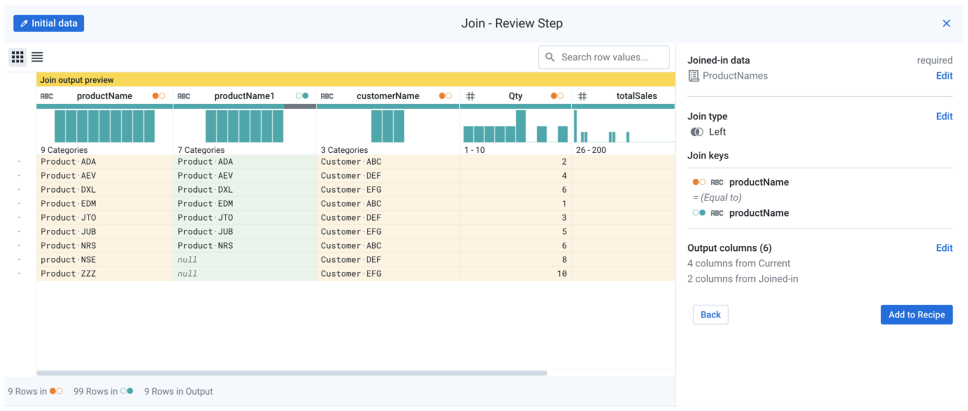
Figure: Edit join keys

7. When you edit the join keys, you can specify the Condition, which defines the type of comparison that is performed to determine a match.
8. The other options allow you to fine-tune how matching is performed. In particular, the **Ignore case** option is off, which means that by default, joins are case-sensitive. So, `product 01` does not match to `Product 01`.

> **Tip:** In the example data, you can see that the `product NSE` entry does not have a match, which is due to differences in case. If Ignore case is enabled, then this entry may find a match. However, you may wish to maintain case-sensitive searches to ensure that you can clean up the data correctly.

9. Click **Save and continue**.
10. Click **Next**.

11. Select all columns, and click **Review**.



Join - Review Step

Join output preview

productName	productName1	customerName	Qty	totalSales
9 Categories	7 Categories	3 Categories	1 - 10	26 - 200
Product·ADA	Product·ADA	Customer·ABC	2	
Product·AEV	Product·AEV	Customer·DEF	4	
Product·DXL	Product·DXL	Customer·EFG	6	
Product·EDM	Product·EDM	Customer·ABC	1	
Product·JTO	Product·JTO	Customer·DEF	3	
Product·JUB	Product·JUB	Customer·EFG	5	
Product·NRS	Product·NRS	Customer·ABC	6	
product·NSE	null	Customer·DEF	8	
Product·ZZZ	null	Customer·EFG	10	

9 Rows in ●○ 99 Rows in ○● 9 Rows in Output

Joined-in data — required
ProductNames — Edit

Join type — Edit
Left

Join keys
●○ productName
= (Equal to)
○● productName

Output columns (6) — Edit
4 columns from Current
2 columns from Joined-in

Figure: Review join

> **Tip:** In the example, you can see that two rows failed to match.

12. Click **Add to recipe**.

Triage Invalid Data

You should now have a dataset containing all columns from both datasets.

> **Tip:** In the example dataset, a second column called `validation` was added. This column contains null values for the mismatched rows. So, you can delete the duplicate `ProductName` column, which contains null values in two rows.

You can make some decisions on how to triage the invalid values in the `ProductNames` column.

Insert error messages

You can use a transformation to replace the null values in the `validation` column with a meaningful message:

Transformation Name	`Edit column with formula`
Parameter: Columns	`validation`
Parameter: Formula	`IF ($col == NULL(),'Error - invalid Product Name','ok')`

The transformation replaces the null values with `Error - invalid Product Name` and writes `ok` for the other rows.

Delete invalid rows

If the entire row of data is invalid because of the invalid value, then you simply delete the row. This transformation deletes rows, where the `validation` column contains a null value:

Transformation Name	`Filter rows`
Parameter: Condition	`Custom formula`

Parameter: Type of formula	`Custom single`
Parameter: Condition	`(validation == NULL())`
Parameter: Action	`Delete matching rows`

Update validation dataset with new values

If you discover that some of the mismatched rows should be part of your validation dataset, you can follow these general steps to add them.

Steps:

1. In Flow View, add a new recipe off of the recipe where the join occurs.
2. Edit this new recipe.
3. Insert this transformation to filter the dataset down to the rows containing the new values:

Transformation Name	`Filter rows`
Parameter: Condition	`Custom formula`
Parameter: Type of formula	`Custom single`
Parameter: Condition	`(validation == NULL())`
Parameter: Action	`Keep matching rows`

4. You can then delete all columns except (in the example) the `ProductName` column. From the column menu of the `ProductName` column, select **Delete others**.
5. You should now have a single column containing the missing values. You can create a union between this dataset and the validation dataset to add the values. See *Append Datasets*.
6. Run a job on this recipe to generate the output file.
7. This output file can then be used to replace the source data. In Flow View, select the imported dataset. From the context panel, open the More menu and select **Replace**.

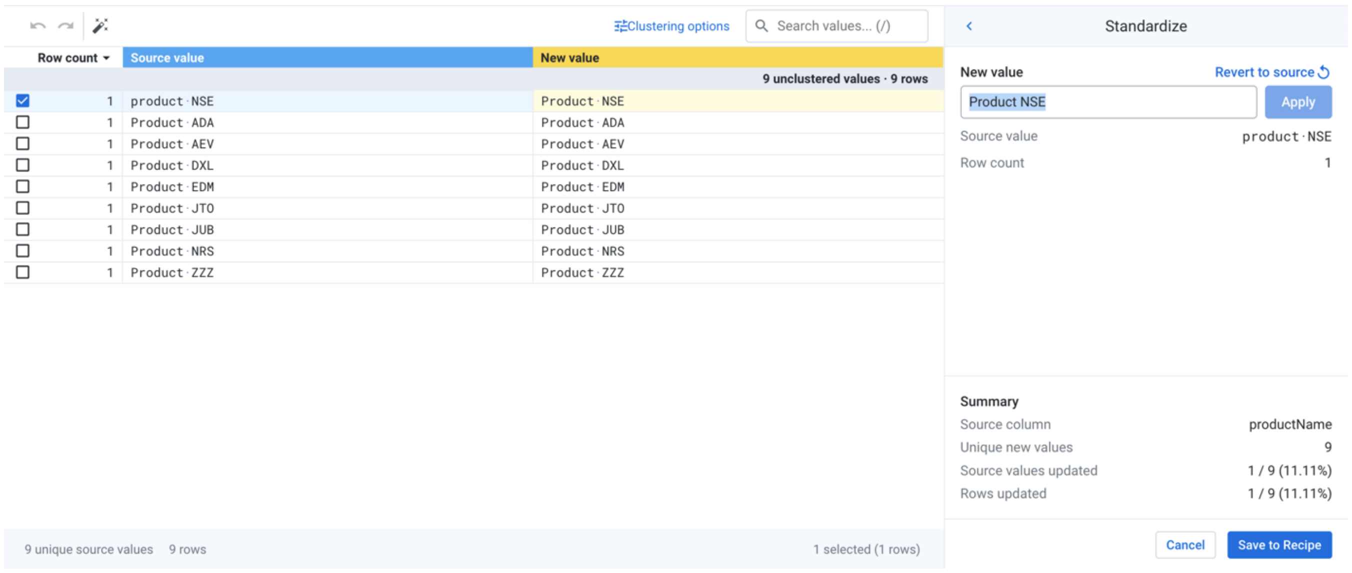
Standardize invalid data

You can standardize values in join key column for your source data, which may address some of the invalid data. The Standardization tool attempts to cluster similar values together within the column, so that you can try to identify if your values in your source dataset can be matched to values in the target dataset.

Tip: After you have identified issues, this step may be best to apply before the join. By applying it before the join, some or all of the mismatched value issues may be addressed.

For your source column, you can select **Standardize** from the column menu.

In the example, the `ProductNames` column values are being standardized. In this case, the lower case `product NSE` value has been corrected to be `Product NSE`.



Row count	Source value	New value
		9 unclustered values · 9 rows
✓ 1	product·NSE	Product·NSE
1	Product·ADA	Product·ADA
1	Product·AEV	Product·AEV
1	Product·DXL	Product·DXL
1	Product·EDM	Product·EDM
1	Product·JTO	Product·JTO
1	Product·JUB	Product·JUB
1	Product·NRS	Product·NRS
1	Product·ZZZ	Product·ZZZ

9 unique source values 9 rows 1 selected (1 rows)

New value: Product NSE (Revert to source, Apply)

Source value	product·NSE
Row count	1

Summary

Source column	productName
Unique new values	9
Source values updated	1 / 9 (11.11%)
Rows updated	1 / 9 (11.11%)

Cancel Save to Recipe

Figure: Standardize join key values

For more information, see *Overview of Cluster Clean*.

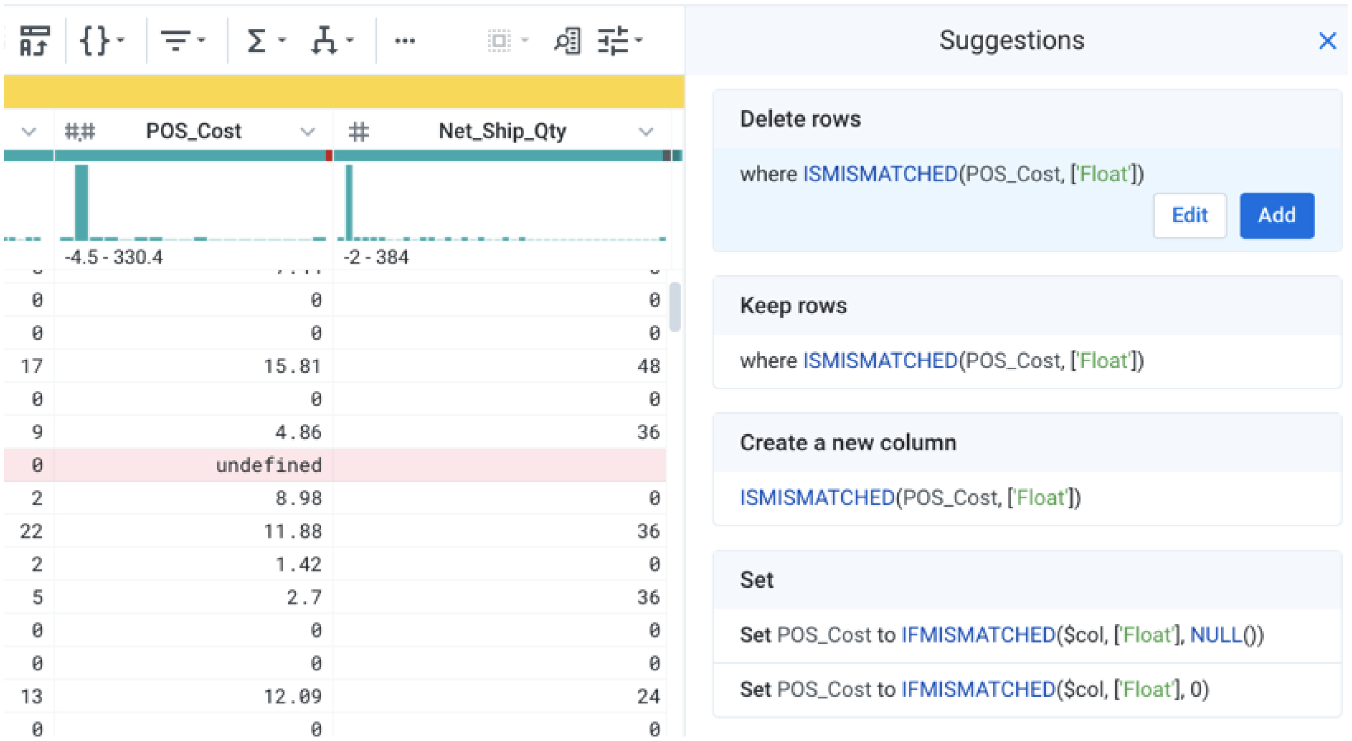
Find Bad Data

Contents:

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.

In the Designer Cloud application , it is very easy to identify where there are errors in your data. What is truly innovative is how you correct them:

1. Identify missing or mismatched data by **color-coded bars** in column data.
2. Select a bar.
3. **Suggestions are offered in a set of cards** on the right panel.
4. Click a suggestion, and **immediately see the effects** of the suggested transformation previewed in the data grid.
    a. If the transformation needs tweaking, you can **edit the transformation as needed**.
    b. If the transformation is not the correct one, **click another suggestion**.
5. When satisfied, you add the transformation, and your sample of data is transformed.



	POS_Cost	Net_Ship_Qty
	-4.5 - 330.4	-2 - 384
0	0	0
0	0	0
17	15.81	48
0	0	0
9	4.86	36
0	undefined	
2	8.98	0
22	11.88	36
2	1.42	0
5	2.7	36
0	0	0
0	0	0
13	12.09	24
0	0	0

Suggestions

Delete rows
where ISMISMATCHED(POS_Cost, ['Float'])
Edit Add

Keep rows
where ISMISMATCHED(POS_Cost, ['Float'])

Create a new column
ISMISMATCHED(POS_Cost, ['Float'])

Set
Set POS_Cost to IFMISMATCHED($col, ['Float'], NULL())
Set POS_Cost to IFMISMATCHED($col, ['Float'], 0)

Figure: Select errors in your data, and review AI-driven suggestions for how to correct. Make the change on the spot.

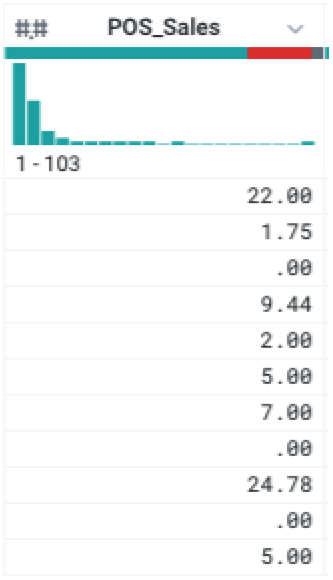
Through this series of seeing, selecting, and refining issues in your sampled data, you can address basic errors in data mismatches, missing data, non-standard values, outlier values, and much more to improve the overall consistency and quality of your data.

Find bad data

In the Transformer page, above each column of data is a data quality bar and histogram.

The top bar is the data quality bar. The **data quality bar** segments the values found in the column into three color-coded bands:

Color bar	Description
green	Valid values for the current data type of the column
red	Invalid values for the current data type of the column
black	Missing values could be empty or null.



```
22.00
1.75
.00
9.44
2.00
5.00
7.00
.00
24.78
.00
5.00
```

Figure: Mismatched values in a column are indicated in red

Change column data type

In the image above, you can identify the **data type** of the column based on the icon to the left of the column name (`POS_Sales`). In this case, the data type is Decimal.

In some cases, invalid data can be fixed by simply changing the column data type. You can click the current data type indicator to review and select a more appropriate data type.

> **Tip:** You can change the data type of the column by click the data type icon for the column.

> **Tip:** No value is invalid for the String data type.

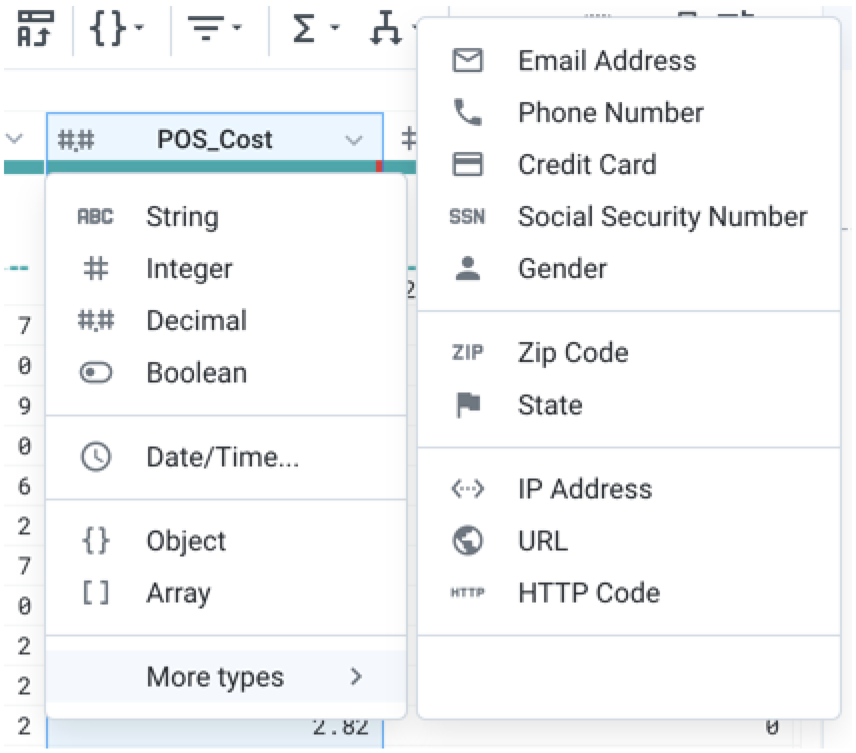
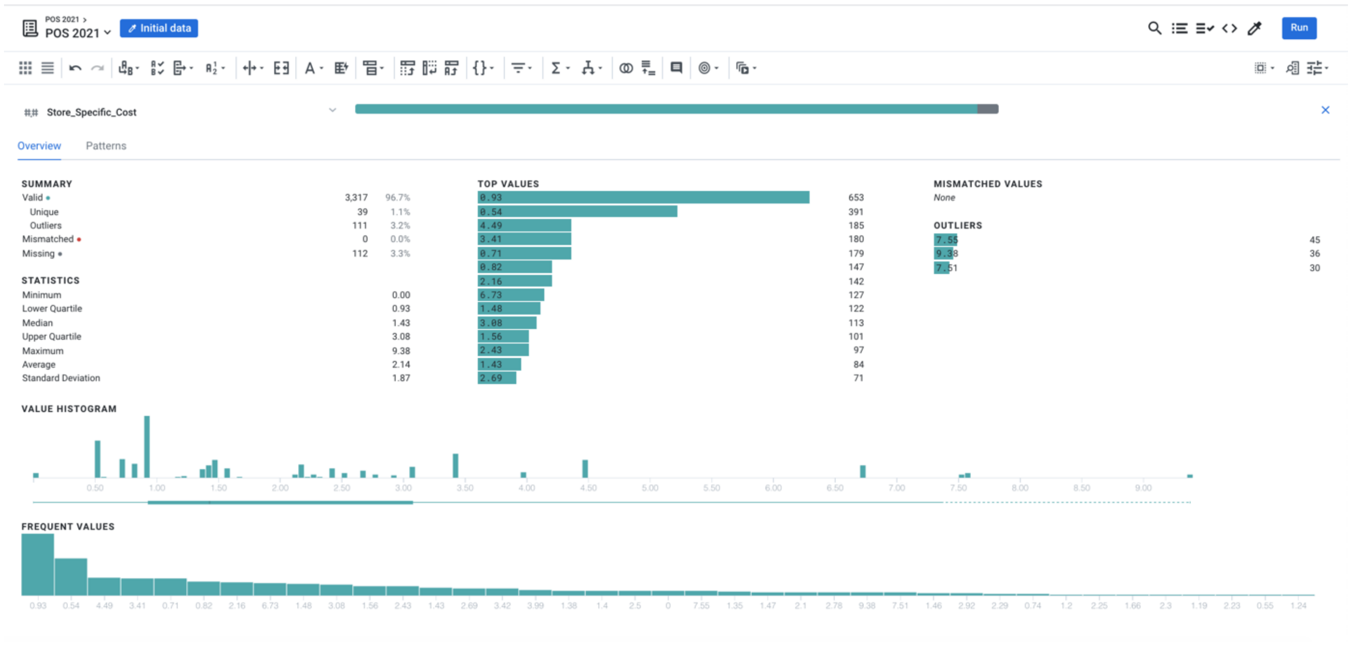


Figure: Change column data type

Find outlier data

You can explore the details of a column of data to review statistical metrics on the data and to locate outlier values. In the column menu, select **Column Details**.



SUMMARY		
Valid ●	3,317	96.7%
Unique	39	1.1%
Outliers	111	3.2%
Mismatched ●	0	0.0%
Missing ●	112	3.3%

STATISTICS	
Minimum	0.00
Lower Quartile	0.93
Median	1.43
Upper Quartile	3.08
Maximum	9.38
Average	2.14
Standard Deviation	1.87

TOP VALUES	
0.93	653
0.54	391
4.49	185
3.41	180
0.71	179
0.82	147
2.16	142
6.73	127
1.48	122
3.08	113
1.56	101
2.43	97
1.43	84
2.69	71

MISMATCHED VALUES

None

OUTLIERS	
7.55	45
9.38	36
7.51	30

Figure: Column Details

Tip: When these bars are clicked or `SHIFT`-clicked, the selected values are used to prompt suggestions for how to transform them.

Tip: You can explore the patterns in the data in the Patterns tab, where you can also use these patterns to standardize the formatting of your data.

Fix 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.

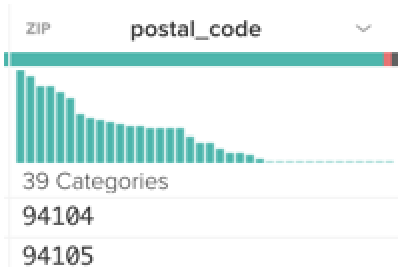


Figure: Mismatched values in red

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	`Edit column with formula`

Name	
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:

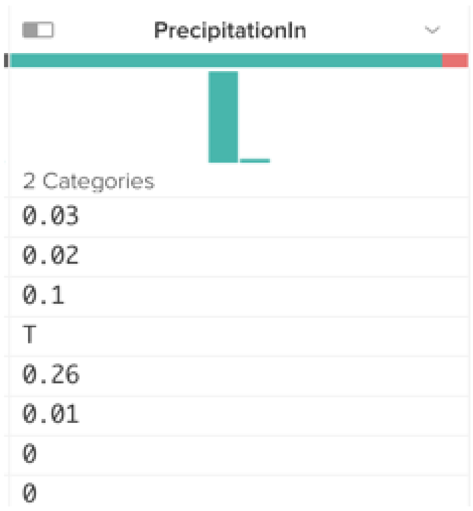
Transformation Name	`Edit column with formula`
Parameter: Columns	`score`
Parameter: Formula	`IF(ISMISMATCHED(score, ['Decimal']),AVERAGE(score), score)`

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.

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:



```
0.03
0.02
0.1
T
0.26
0.01
0
0
```

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.

Find Missing Data

Contents:

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.
- 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 (`\l`) are considered missing.

In the data quality bar, missing values are identified in gray:

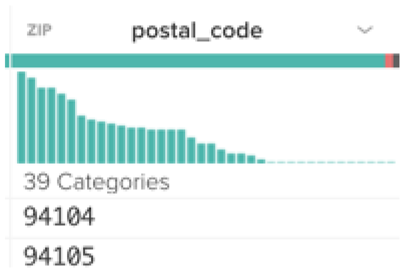


Figure: Missing values in gray

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 seem 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.

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	`IF(ISMISSING([nickname]),first_name,nickname)`

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	`IF(ISMISSING([unit_price]),(price / weight_kg), unit_price)`

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

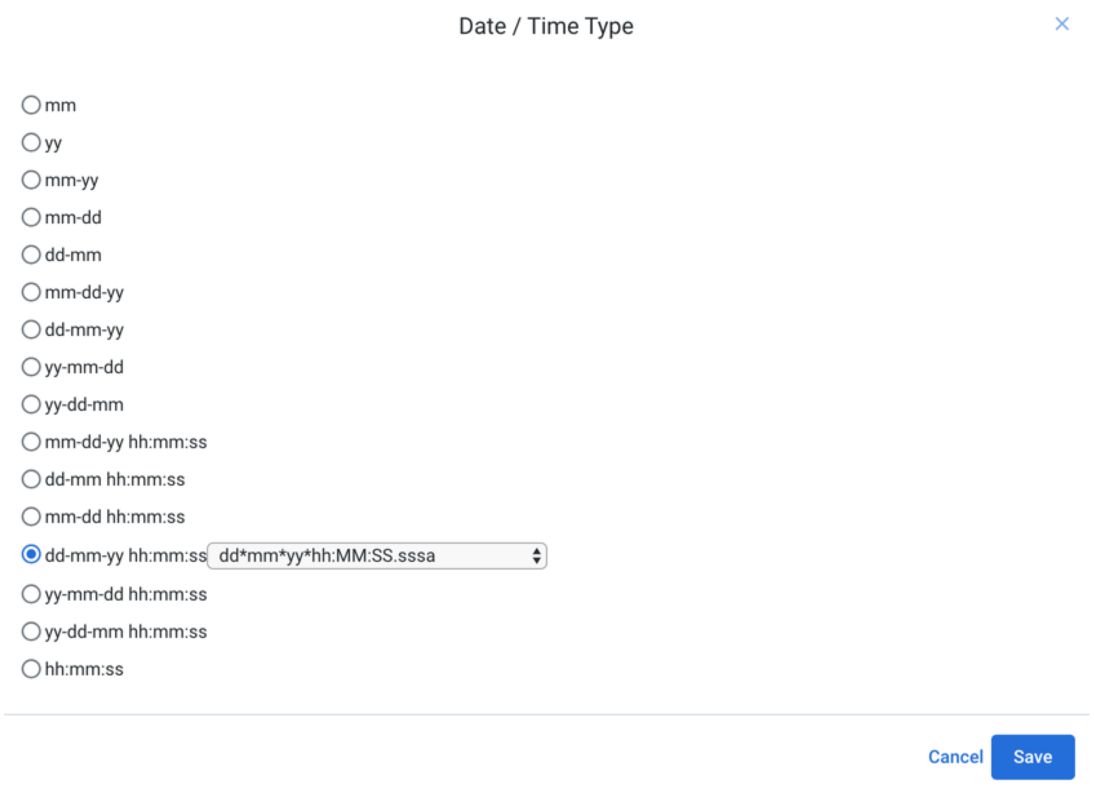
On import, timestamp data may be classified as String data. For now, this is ok.

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] dd*mm*yy*hh:MM:SS.sssa
- yy-mm-dd hh:mm:ss
- yy-dd-mm hh:mm:ss
- hh:mm:ss

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.

Manage Null Values

Contents:

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 when the job is run on Trifacta Photon. 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:

Transformation Name	`New formula`
Parameter: Formula	`ISNULL(isActive)`
Parameter: New column name	`nulls2`

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.

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	`column1~column255`
Parameter: Formula	`IF(ISMISSING([$col]), null(), $col)`

The above transform writes null values, but these values are converted to missing values on export.

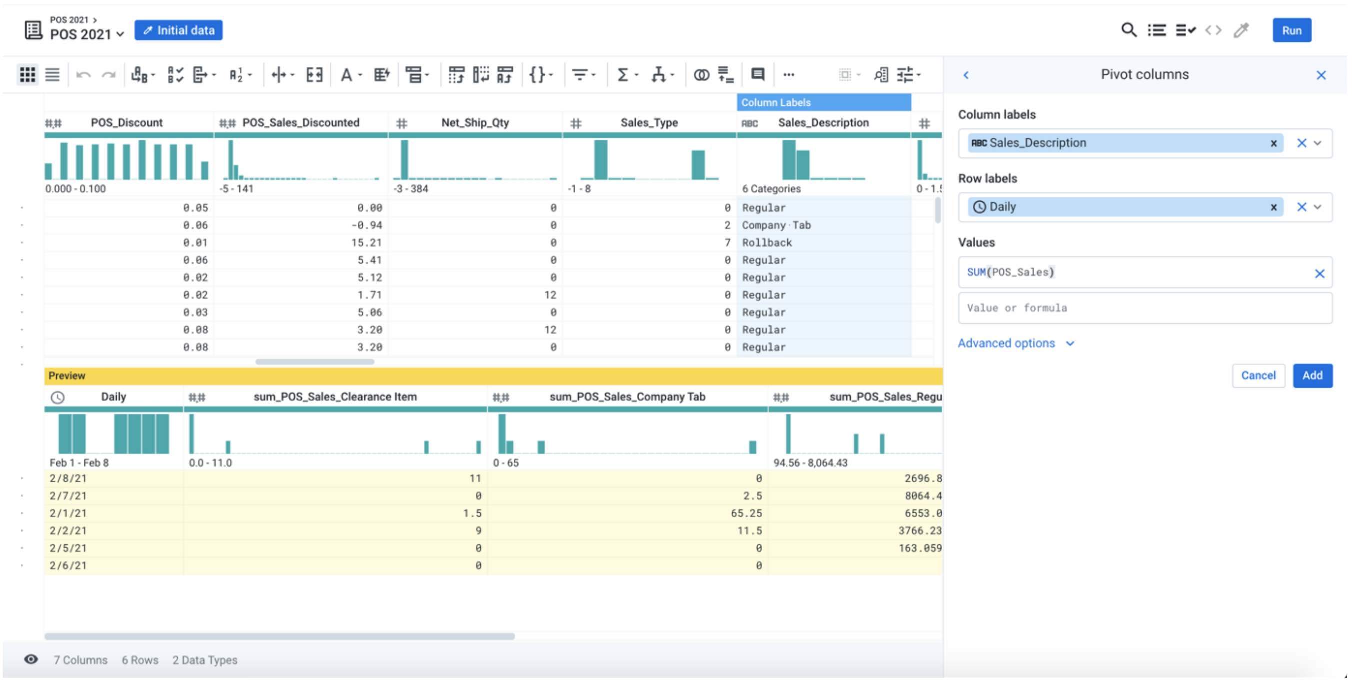
Structuring Tasks

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 your apply restructuring steps. For more information, see *Create Branching Outputs*.

Reshape Your Dataset

You can change the composition of rows and columns in your dataset through transformation.



POS_Discount	POS_Sales_Discounted	Net_Ship_Qty	Sales_Type	Sales_Description
0.000 - 0.100	-5 - 141	-3 - 384	-1 - 8	6 Categories
0.05	0.00	0	0	Regular
0.06	-0.94	0	2	Company Tab
0.01	15.21	0	7	Rollback
0.06	5.41	0	0	Regular
0.02	5.12	0	0	Regular
0.02	1.71	12	0	Regular
0.03	5.06	0	0	Regular
0.08	3.20	12	0	Regular
0.08	3.20	0	0	Regular

Daily	sum_POS_Sales_Clearance Item	sum_POS_Sales_Company Tab	sum_POS_Sales_Regu
Feb 1 - Feb 8	0.0 - 11.0	0 - 65	94.56 - 8,064.43
2/8/21	11	0	2696.8
2/7/21	0	2.5	8064.4
2/1/21	1.5	65.25	6553.0
2/2/21	9	11.5	3766.23
2/5/21	0	0	163.059
2/6/21	0	0	

Pivot columns

Column labels: Sales_Description

Row labels: Daily

Values: SUM(POS_Sales)

Value or formula

Advanced options

Cancel Add

Figure: Change the structure of your data through menu-driven selection of rows, columns, and formulas.

The following types of transformations can be used to reshape or completely replace the columns and rows in your dataset:

- **Split:** Split a column based on one or more known delimiters or based on index positions in the data. See *Split Column*
- **Aggregation:** Perform computations across a set of grouped rows, generating the results in a new column or a reshaped table. See *Create Aggregations*.
- **Pivot:** Create pivot tables based on one or more calculations and selected fields. See *Pivot Data*.
- **Select:** Select a set of columns to completely replace the current dataset. See *Select*.

Nest and Unnest

Data in separate columns can be combined together into single columns as arrays or objects (maps). Similarly, columns of these object types can be expanded as new columns or new rows in your dataset.

- See *Nest Your Data*.
- See *Unnest Your Data*.

Initial Parsing Steps

Contents:

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 assume s 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.

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.

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.

NOTE: The maximum permitted length of any individual record on input is 20 MB.

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 column naming transformation step.

Converted data

Some formats, such as binary data or JSON, are converted to a format that is natively understood by the product before the data is available for sampling and transformation.

Excel

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.

JSON

If 80% of the records in an imported dataset are valid JSON objects, then the data is parsed as JSON through a conversion process.

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.

For more information, see *Working with JSON v2*.

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.

Transform a Column

Contents:

The Transformer page provides several ways of rapidly transforming your data at the column level.

Tip: Except for finding and selecting columns, all of these actions add transformations to your recipe. After the step has been added, you can edit the recipe to adjust the step, including manually adding or removing columns.

Find Columns

For wide datasets, you can locate using either of the following methods.

Find by name

Steps:

1. In the Transformer toolbar, click the **Find column** icon.
2. In the Find column panel, begin typing the name of the column.

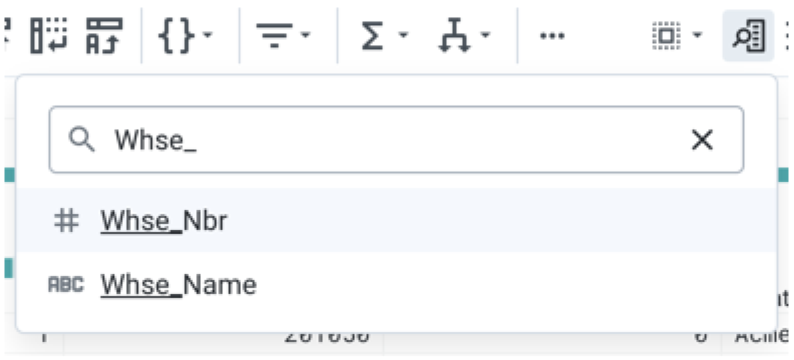


Figure: Find column panel

3. Select the column.
4. The data grid is moved to display the selected column on your screen.

Find in Column Browser

You may find it easier to use Column Browser to find and select columns for transformation.

Steps:

1. In the Transformer toolbar, click the **Columns** icon on the left side. Column Browser is displayed:

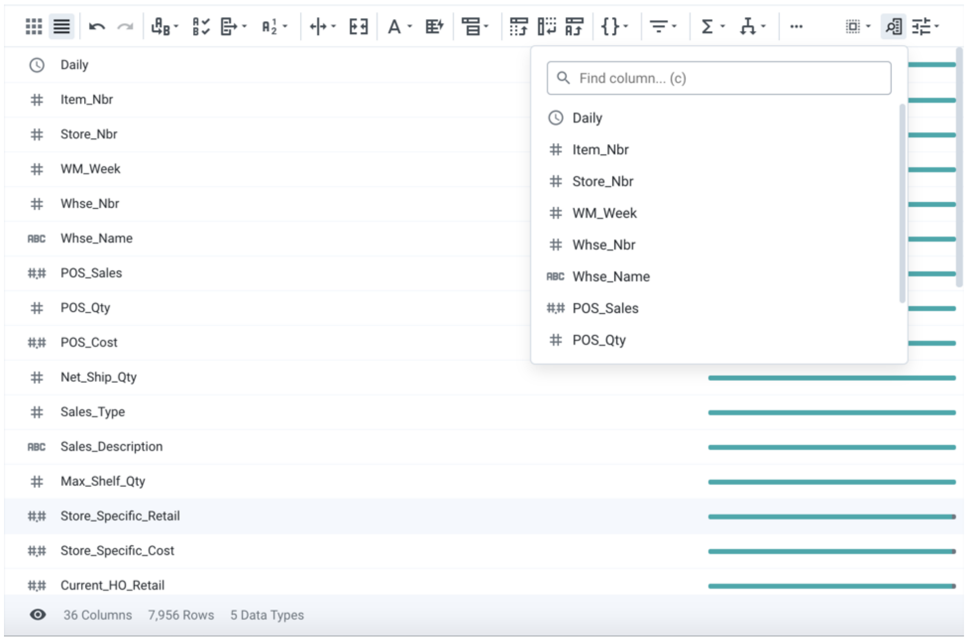


Figure: Column Browser

2. You can scroll the columns to select one or more columns for transformation.

> **Tip:** The Find column panel can also be used to locate columns in this list.

Select Columns

When you locate a column or columns of interest, you can select using one or more of the following methods:

- Click an individual column header to select the column.
- Press `SHIFT` and click to select consecutive columns.
- Press `CMD/CTRL` to select multiple discrete columns.

A set of suggestion cards is displayed on the right side that reflect the column or columns that you have selected.

Column menu

When you select a single column, you can explore additional transformations through the column menu.

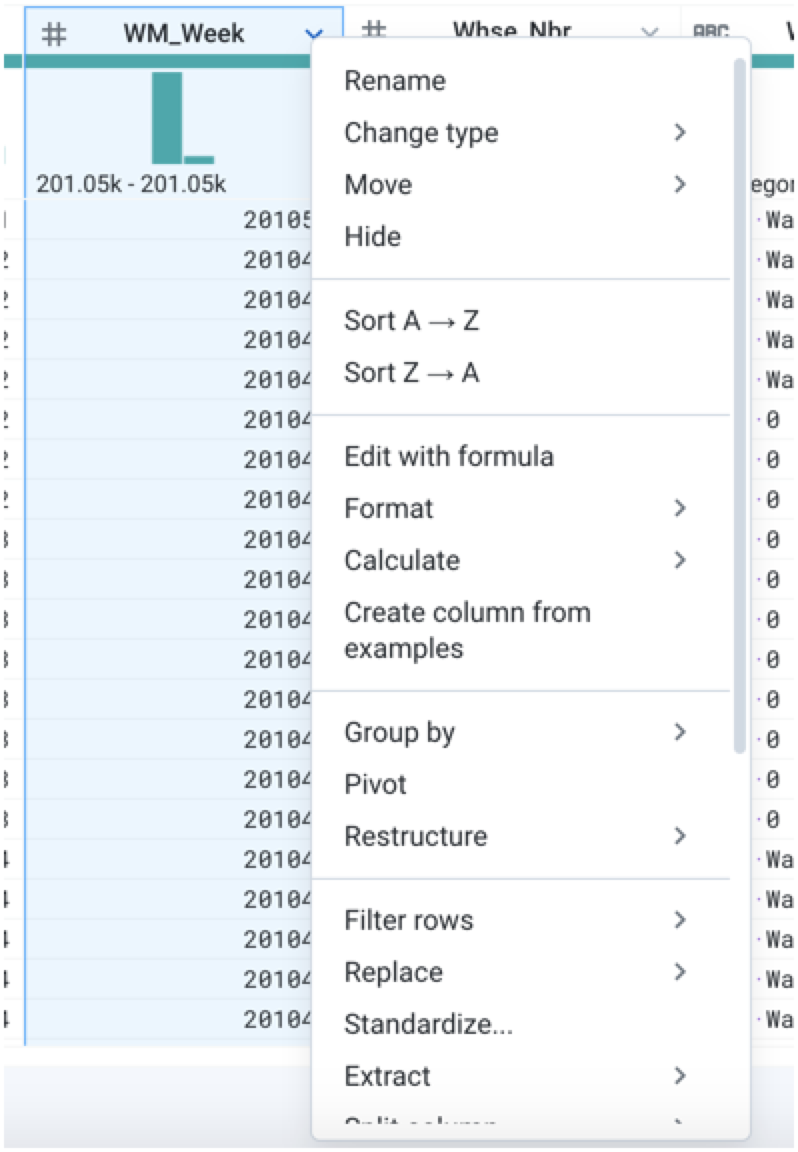


Figure: Column menu

The options available in the column menu depend upon the currently selected data type of the column. For example, the options for Integer columns are different than the options for String columns.

Tip: The displayed column menu applies only to the column from which it was opened. You cannot apply column menu transformations to multiple columns, although you can modify the transformation after it has been added to your recipe to apply to multiple columns.

Cut, Copy and Paste Columns

Steps:

1. Click the **Columns** icon in the Transformer toolbar.
2. In Column Browser, select the header of the column.
3. From the column menu, select **Cut** or **Copy**.
4. Select the column where you wish to paste the column.
5. Right-click and select the appropriate **Paste** option: before, after, or paste values.

Move Columns

You can move one or more columns to a new location in your dataset.

Steps:

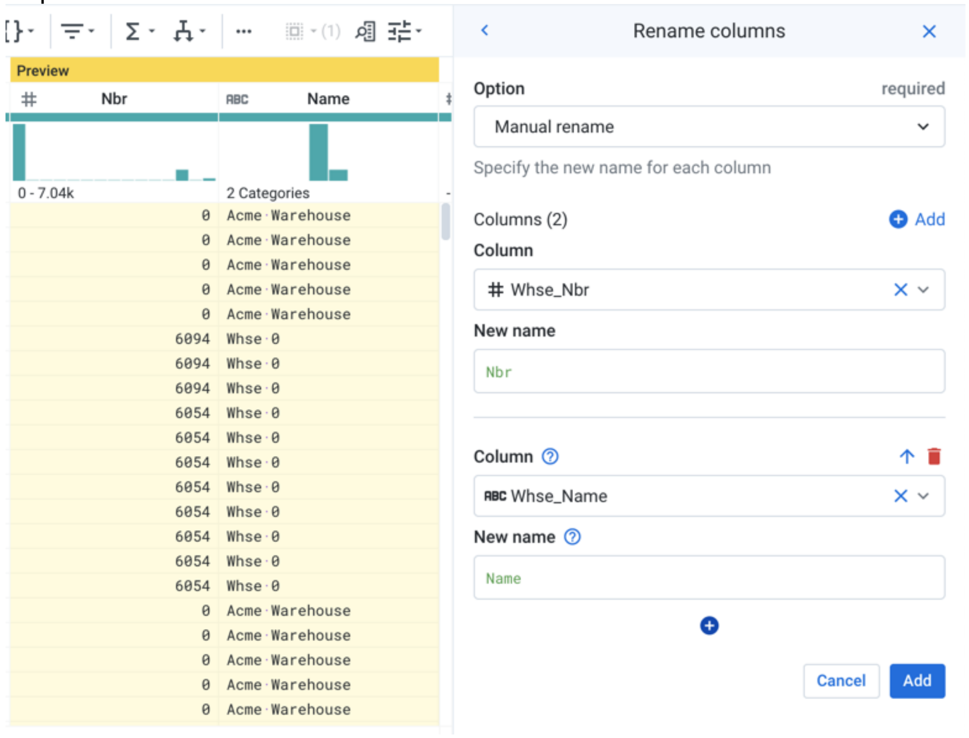
1. Click the **Columns** icon in the Transformer toolbar.
2. In Column Browser, select the column or columns to move.
3. Right-click and select the appropriate **Move** option:
    a. To beginning: Columns are moved to the beginning of columns in the dataset.
    b. To end: Columns are moved to the end of columns in the dataset.
    c. Before/after: In the Transform Builder, you can specify exactly where you wish to move the columns.
4. When the transformation is added to your recipe, the columns are moved.

Rename Columns

You can rename one or more columns in a single step.

Steps:

1. In the Transformer toolbar, click the **Grid** icon.
2. In the data grid, click the name of the column.
3. In the data grid, select **Rename** from the column menu.
4. Change the name as needed.
5. In the Transform Builder, you can add additional columns to be renamed.
    a. Click **+ Add**.
    b. Enter the name of a column to rename.
    c. Enter the new name.
    d. Repeat as needed.



Preview

Nbr	Name
0 - 7.04k	2 Categories
0	Acme·Warehouse
0	Acme·Warehouse
0	Acme·Warehouse
0	Acme·Warehouse
0	Acme·Warehouse
6094	Whse·0
6094	Whse·0
6094	Whse·0
6054	Whse·0
6054	Whse·0
6054	Whse·0
6054	Whse·0
6054	Whse·0
6054	Whse·0
6054	Whse·0
6054	Whse·0
0	Acme·Warehouse
0	Acme·Warehouse
0	Acme·Warehouse
0	Acme·Warehouse
0	Acme·Warehouse

Rename columns

Option: required
Manual rename
Specify the new name for each column

Columns (2) Add

Column: Whse_Nbr
New name: Nbr

Column: Whse_Name
New name: Name

Cancel Add

Figure: Rename multiple columns

6. When the columns have been renamed, click **Add** to add the step to your recipe.

Change Column Data Type

By default, Designer Cloud Powered by Trifacta® Enterprise Edition attempts to infer a column's data type based on the data contained in it.

A **data type** defines the expected values in a column. For example, when Designer Cloud Powered by Trifacta Enterprise Edition detects numeric values in a column, the column may be assigned the Integer data type or the Decimal data type, depending on the data in it.

In some cases, you may need to re-assign a column's data type. For example, if you are modifying values and sub-values in the column, you may find it easier to work with the data temporarily setting it to String data type.

Tip: To change the data type of multiple columns to the same type, you may find it easier to use the Column Browser. Select the columns. Then, right-click the select and select the appropriate type from the **Change type** menu.

Steps:

1. If you are changing the data type for a single column, select the column header.
2. From the data type menu, select the new type:

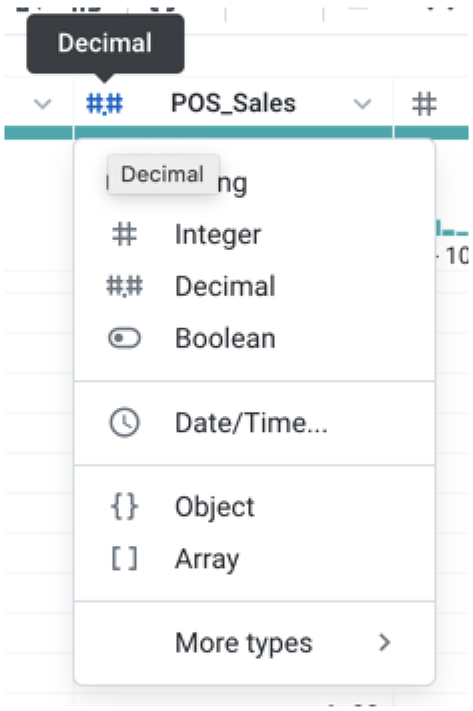


Figure: Change data type

3. A step is added to your recipe to change the selected column to the new type.

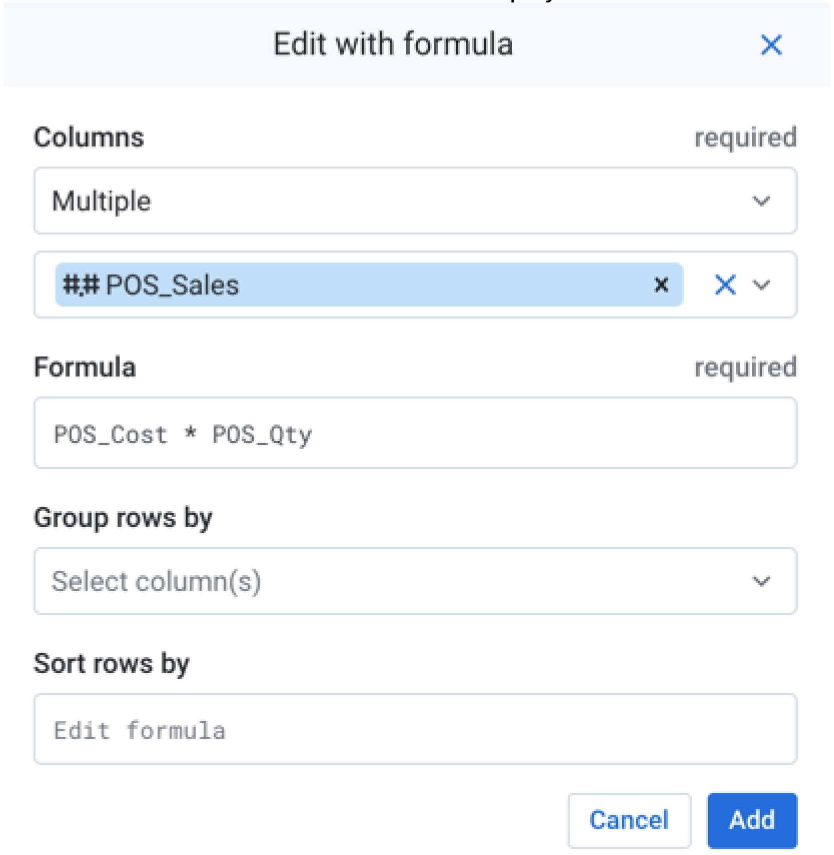
Tip: You can edit the recipe step to apply the new data type to more columns.

Edit Column

You can modify the contents of a column through transformation.

Steps:

1. Select the column. From the column menu, select **Edit with formula**.
2. The Edit with Formula transformation is displayed in the Transform Builder:



Edit with formula

Columns required

Multiple

POS_Sales

Formula required

```
POS_Cost * POS_Qty
```

Group rows by

Select column(s)

Sort rows by

```
Edit formula
```

Figure: Edit with formula

   a. **Formula:** Enter a valid formula. You can edit functions, column references, constants, or combinations of these objects.
   b. **Group rows by:** The formula can be calculated based on the groupings of values of a selected column. For example, you may wish to compute based on separate values in the `Region` column.
3. To edit the column with the new formula, click **Add**.

Delete Columns

Select the column or columns. From the column menu, select **Delete**.

Tip: To restore the columns, you can click the Undo icon immediately in the Transformer toolbar, or you can delete the recipe step through the Recipe panel.

Merge Columns

You can concatenate non-numeric data together from multiple columns into a new column. You can apply this technique to columns of different un-nested data types.

Steps:

1. In the Transformer page, identify the columns that you wish to merge.
2. Click the **Merge columns** icon in the toolbar.

Tip: You can also search for `Merge columns` in the Search panel.

3. In the Merge columns transformation, enter the required details:
   - **Columns**: The columns containing the values you wish to merge. You can select from the following options:
     - **Multiple**: Select one or more columns from the drop-down list.
     - **Range**: Specify a start column and ending column. All columns inclusive are selected.
     - **All**: Select all columns in the dataset.

       NOTE: This option merges the data in all of the columns into a new single column.

     - **Advanced**: Specify the columns using a comma-separated list. You can combine multiple and range options under Advanced. Example:

       ```
       c1,c3,c5~c8
       ```

       Above specifies the following columns: `c1,c3,c5,c6,c7,c8`.
   - **Separator**: The separator to be placed between each value. This value can be empty.
   - **New column name**: The name of the new column.
4. To add the step to your recipe, click **Add**.
5. The new column is added to your dataset.
6. As needed, you can delete the source columns.

Transform a Row

Contents:

You can transform rows in your dataset based on conditions you specify or by using a custom formula.

Deduplicate Rows

You can use the `Remove duplicate rows` transformation to remove identical rows from the dataset.

Steps:

1. In the Transformer page, click the **Recipe** icon. The Recipe Panel is displayed.
2. In the Search Transformations panel, enter `Remove duplicate rows`. The duplicated rows are highlighted in the Transformer page.
3. To remove the duplicate rows, click **Add**.

Limitations:

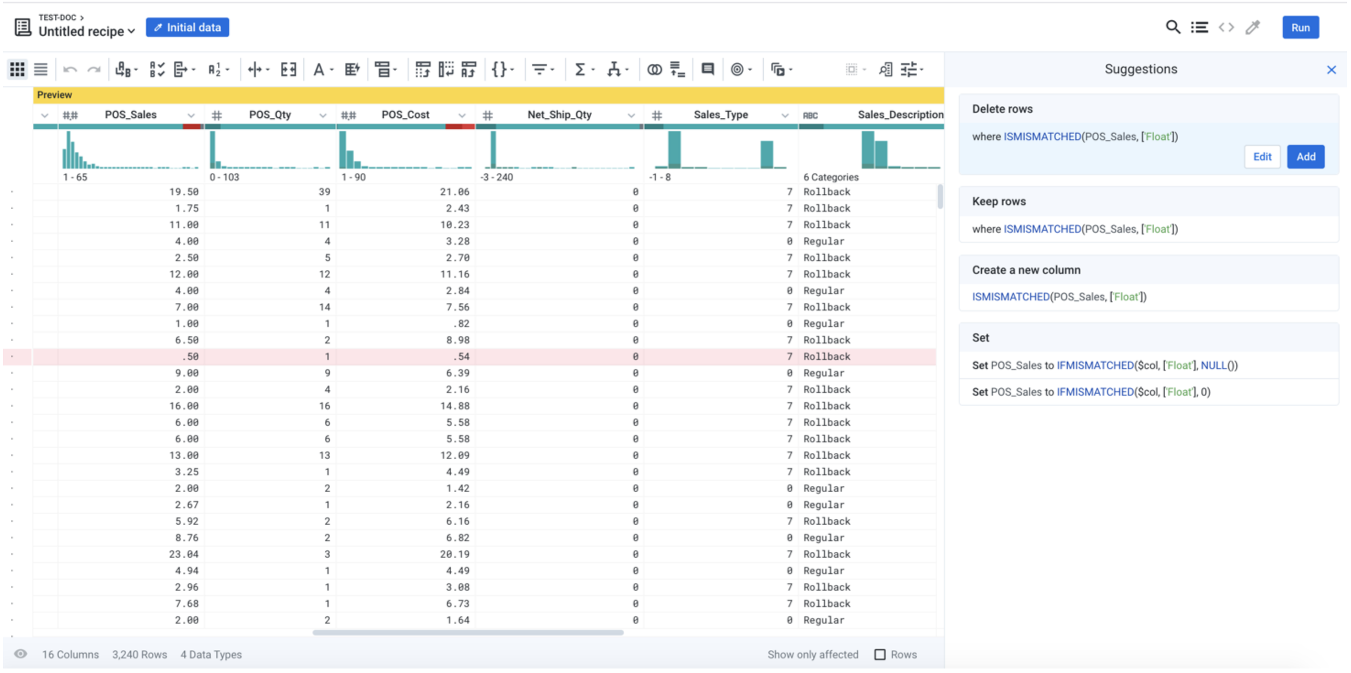
- This transformation is case-sensitive. So, if a column has values `Hash` and `HASH`, the rows containing those values are not considered duplicates and cannot be removed with this transformation.
- Whitespace and the beginning and ending of values is not ignored.

Delete Rows on a Condition

By selection

You can delete the rows for any column based on a condition that you select. When you select values in the data quality bar that for a column, a set of predictive suggestions is displayed in the right panel. Locate the Delete rows transformation, which contains a `where` clause to indicate the condition on which a row is deleted.

For example, if your data contains mismatched values, select the red bar in the data quality bar and use the Delete rows transformation to delete the mismatched values from the dataset:



TEST-DOC >
Untitled recipe
Initial data

Preview

POS_Sales	POS_Qty	POS_Cost	Net_Ship_Qty	Sales_Type	Sales_Description
1 - 65	0 - 103	1 - 90	-3 - 240	-1 - 8	6 Categories
19.50	39	21.06	0	7	Rollback
1.75	1	2.43	0	7	Rollback
11.00	11	10.23	0	7	Rollback
4.00	4	3.28	0	0	Regular
2.50	5	2.70	0	7	Rollback
12.00	12	11.16	0	7	Rollback
4.00	4	2.84	0	0	Regular
7.00	14	7.56	0	7	Rollback
1.00	1	.82	0	0	Regular
6.50	2	8.98	0	7	Rollback
.50	1	.54	0	7	Rollback
9.00	9	6.39	0	0	Regular
2.00	4	2.16	0	7	Rollback
16.00	16	14.88	0	7	Rollback
6.00	6	5.58	0	7	Rollback
6.00	6	5.58	0	7	Rollback
13.00	13	12.09	0	7	Rollback
3.25	1	4.49	0	7	Rollback
2.00	2	1.42	0	0	Regular
2.67	1	2.16	0	0	Regular
5.92	2	6.16	0	7	Rollback
8.76	2	6.82	0	0	Regular
23.04	3	20.19	0	7	Rollback
4.94	1	4.49	0	0	Regular
2.96	1	3.08	0	7	Rollback
7.68	1	6.73	0	7	Rollback
2.00	2	1.64	0	0	Regular

16 Columns 3,240 Rows 4 Data Types Show only affected Rows

Suggestions

Delete rows
where ISMISMATCHED(POS_Sales, ['Float'])
Edit Add

Keep rows
where ISMISMATCHED(POS_Sales, ['Float'])

Create a new column
ISMISMATCHED(POS_Sales, ['Float'])

Set
Set POS_Sales to IFMISMATCHED($col, ['Float'], NULL())
Set POS_Sales to IFMISMATCHED($col, ['Float'], 0)

Figure: Deleting rows based on values that are invalid for the column's data type

By custom condition

You can also delete a set of rows based on a condition expression that you specify. If the conditional expression evaluates to `true`, then the row is deleted.

In the following example, you can delete a set of `POS_Sales` values based on a conditional parameter. All `POS_Sales` where the value is less than or equal to 5 are deleted, and the rest are retained.

Store_Nbr	Item_Nbr	POS_Sales	POS_Qty	POS_Cost
200	322000	22.00	44	23.76
200	323000	1.75	1	2.43
200	325000	0.00	0	0.00
200	325000	9.44	8	7.44
200	326000	2.00	2	1.64
200	327000	5.00	10	5.40
200	328000	7.00	7	6.51
200	341000	0.00	0	0.00
200	341000	7.78	21	19.53
200	342000	0.00	0	0.00
200	350000	5.00	10	5.40

Steps:

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, `POS_Sales`.
    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. In this case, select `Delete matching rows`.

Transformation Name	`Filter rows`
Parameter: Condition	`Less than or equal to`
Parameter: Column	`POS_Sales`
Parameter: Value	`POS_Sales <= 5`
Parameter: Action	`Delete matching rows`

4. To add the recipe to the step, click **Add**. The selected rows are deleted from the dataset.

Results:

Store_Nbr	Item_Nbr	POS_Sales	POS_Qty	POS_Cost
200	322000	22.00	44	23.76
200	325000	9.44	8	7.44
200	328000	7.00	7	6.51
200	341000	7.78	21	19.53

Filter the Data Grid

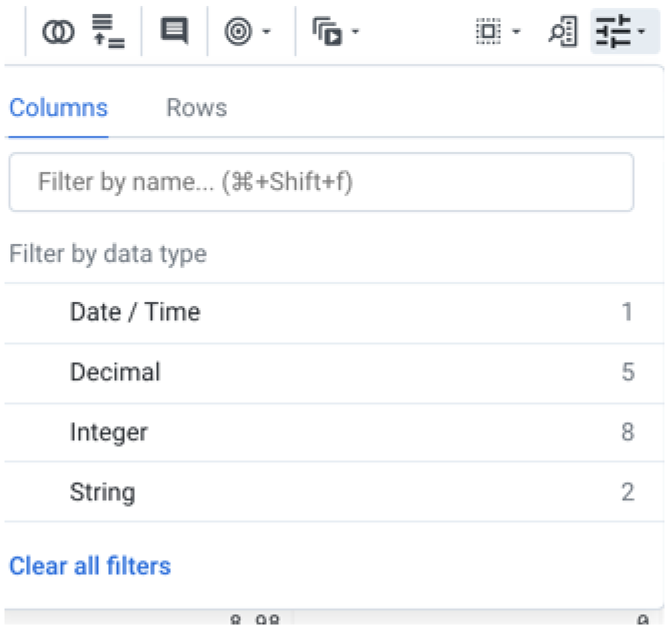
You can filter the sampled data to display only a subset of rows in the Transformer page. You can filter the dataset by rows, columns, and data types.

Columns or rows that are filtered are hidden only from the view. The hidden data is part of the sample and the dataset and is included in the output.

Steps:

1. In the Transformer page, click the **Filter** icon.

2. In the Filter by name panel, enter the required options:



Columns | Rows

Filter by name... (⌘+Shift+f)

Filter by data type

Date / Time	1
Decimal	5
Integer	8
String	2

Clear all filters

Figure: Filter by name

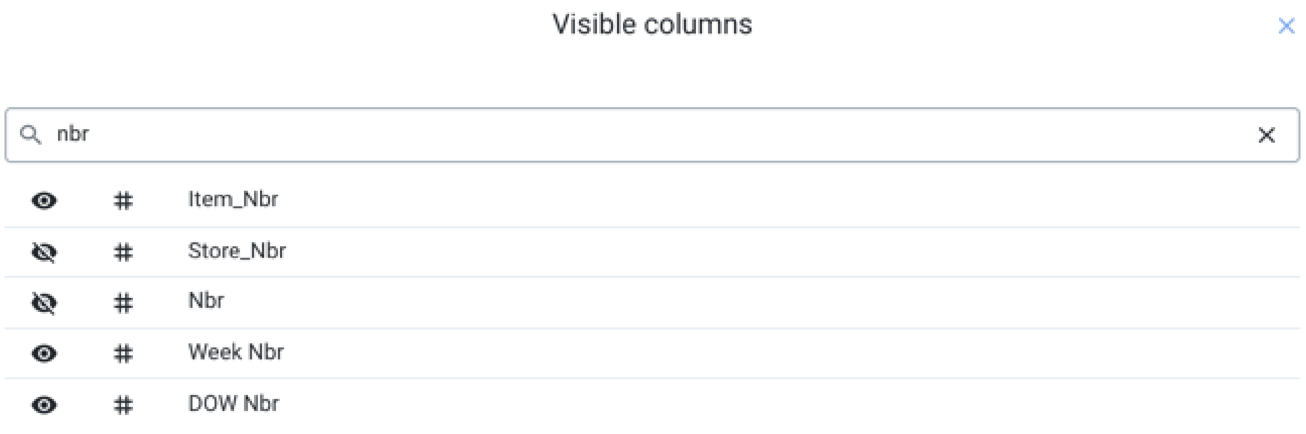
   a. **Columns:** Search for individual columns or filter columns of a specific type. Filtered columns are displayed, and the rest are hidden.
   b. **Rows:** Highlight search term matches found in any column for a row.

   > **NOTE**: You can also filter the rows and columns by data type.

3. To clear all filters, click **Clear all filters**.

Hide and show columns

In the Transformer page, click the **Eye** icon in the status bar at the bottom of the page. Click one or more columns to show or hide them.



Visible columns

nbr

👁	#	Item_Nbr
⊘	#	Store_Nbr
⊘	#	Nbr
👁	#	Week Nbr
👁	#	DOW Nbr

Figure: Visible Columns panel

> **Tip:** You can also use the **Columns** menu in the Transformer page to show or hide columns.

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.

Filter during previews

When you are creating or editing transformations, the data grid displays a preview of results of the transformation in development. As needed, you can narrow the display to show only the affected rows, columns, or both. Select the **Show only affected rows** checkbox in the status bar at the bottom of the Transformer page.

Split Column

Contents:

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.

Tip: When this transform appears in a suggestion card, the maximum number of suggested columns to split is 250, which may prevent the browser from crashing. If your dataset requires additional column splits, you can edit the transformation and increase the maximum number of splits. Avoid creating datasets that are wider than 1000 columns.

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>	DATETTIME/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>`	`DATETTIME/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.

- 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
`DATETTIME/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 editing 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.

Tip: When specifying numeric positions, you do not have to list the positions in numeric order. You can now do faster iteration since you can add new positions as needed when previewing the transformation.

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.

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.

Move Columns

Contents:

You can move or reorder individual columns or multiple columns through multiple methods.

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 explore and select the appropriate transformation to move the columns. For more information on suggestions, see *Explore Suggestions*.

Move using Column Icons

Select the Column View icon in the Transformer bar to move columns in the 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 panel of the Transform Builder.
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. S ee 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 Target Schema Mapping

Target schema mapping 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 Target Schema Mapping*.

Add New Column

Contents:

Through the Transformer page, you can create a new column of data, containing constant values, references to other columns or metadata, or functions that you specify. Some transformations also create new columns of data.

Adding a new column can insert any of the following types of data into your dataset:

- **Constant or literal values.** Example formulas:

  ```
  "clean"
  ```

  ```
  5
  ```

  > **NOTE:** When a new column of data is inserted, the Designer Cloud® application attempts to infer the data type of the values in the column. In the above examples, the inferred data types would be String and Integer, respectively.

- **References to another column.** You can create a new column of data that contains a reference to another column, which effectively copies and pastes the column values into the new column. In the Formula, you type the name of the source column.
    - For columns containing numeric data types, you can create a new column by applying a formula to combine the data. You can apply formulas like:

      ```
      column1 + column2
      ```

      To round the above computation to two decimal places, use the following:

      ```
      round(column1 + column2,2)
      ```

- **Metadata about your dataset.** You can insert references to metadata about your source dataset. For more information on adding metadata, see *Insert Metadata*.
- **Functions.** Your new column can contain the results of a function calculations. Example formulas:

  ```
  1+2+3
  ```

  ```
  stdev(circumferenceCM)
  ```

- **Combinations.** Constants, literals, and functions can be combined into a single formula. Example:

  ```
  multiply(add(1,sales_tax_rate),sales_pre_tax)
  ```

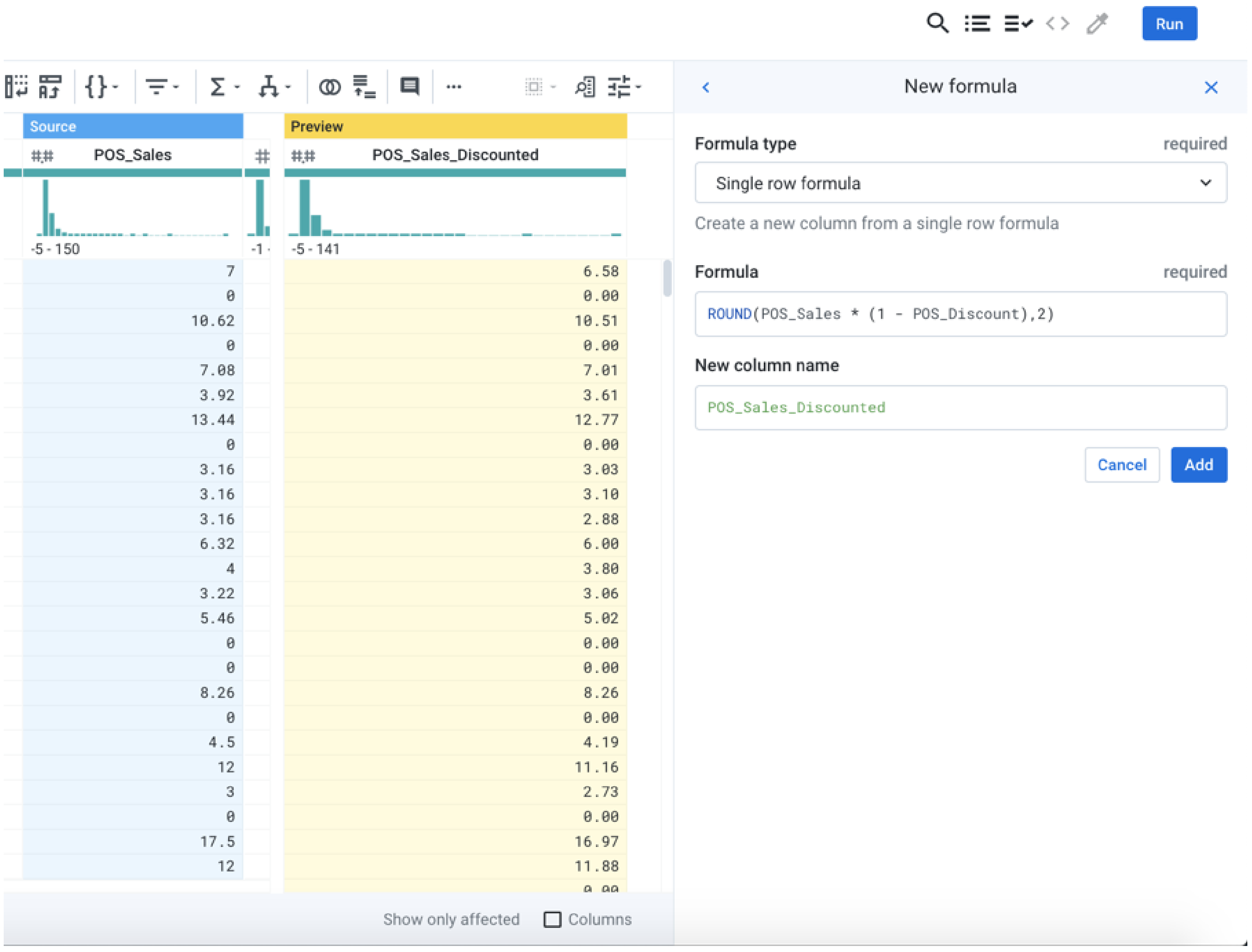
NOTE: As indicated in the above example, functions can be nested inside of each other.

Add New Data

You can add in new data to your dataset through the following methods.

New columns

Create new columns in your dataset containing literal values, function outputs, or values from other columns, including extraction of values into new columns.



POS_Sales	POS_Sales_Discounted
7	6.58
0	0.00
10.62	10.51
0	0.00
7.08	7.01
3.92	3.61
13.44	12.77
0	0.00
3.16	3.03
3.16	3.10
3.16	2.88
6.32	6.00
4	3.80
3.22	3.06
5.46	5.02
0	0.00
0	0.00
8.26	8.26
0	0.00
4.5	4.19
12	11.16
3	2.73
0	0.00
17.5	16.97
12	11.88

Formula type: Single row formula

Create a new column from a single row formula

Formula:

```
ROUND(POS_Sales * (1 - POS_Discount),2)
```

New column name: POS_Sales_Discounted

Figure: Build a New Formula transformation to craft a new column of data containing custom functions or literal values.

Steps:

1. In the Transformer page, click the **Recipe** icon. The Recipe Panel is displayed.
2. In the Recipe panel, click **New Step**.
3. To add a recipe step, enter `New Formula`. This transformation adds a new column of data.
4. In the New Formula panel, enter the required details:
    a. **Formula type**: Type of formula in the transformation.
        i. Single row formula: Formula calculations are contained within a single row of values.

ii. Multiple row formula: Formula calculations involve multiple rows of inputs. In this formula type, you can sort and group the rows.

b. **Formula**: The formula you want to enter to create a new column. See above for examples.
c. **New column name**: The name of the new column.

5. To create a new column, click **Add**. A new column is created.

Metadata

You can insert references to metadata about your datasources within your dataset. Source row and path information can be added as new data.

For more information, see *Insert Metadata*.

Delete Data

Contents:

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 step 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	`(action_count >= 1) && (action_count <= 10)`
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.

Sort Rows

Contents:

You can sort the ordering of rows in your dataset based on the values in one or more columns.

Sort Order

The order of rows is determined by how the application orders for a sort column's data type.For more information, see *Sort Order*.

Example Dataset

rowId	lastName	firstName	startDate
3	Jones	Tim	2022-05-01
1	Jones	Cary	2022-03-15
2	Abbott	Michael	2022-02-07

Sort by a Single Column

To sort rows by the values in a single column, click the column menu and then select one of the following options:

- **Sort A Z:** Sort the rows in the dataset in ascending order, as determined by the data type of the column.
- **Sort Z A:** Sort the rows in descending order, as determined by the data type of the column.

Transformation:

For the example dataset above, suppose you want to sort the rows based on the rowId value in ascending order. From the rowId column menu, you could select Sort A Z, which yields the following step in your recipe.

Transformation Name	`Sort rows`
Parameter: Sort by	`rowId`

Results:

The resulting dataset looks like the following:

rowId	lastName	firstName	startDate
1	Jones	Cary	2022-03-15
2	Abbott	Michael	2022-02-07
3	Jones	Tim	2022-05-01

For more information, see *Column Menus*.

Reversing Sort Order

Suppose you wish to reverse the sort order of the above transformation to organize the rows in descending order based on the rowId column.

Transformation:

You can edit the recipe step and insert a dash (`-`) in front of the rowId value:

Transformation Name	`Sort rows`
Parameter: Sort by	`-rowId`

Results:

rowId	lastName	firstName	startDate
3	Jones	Tim	2022-05-01
2	Abbott	Michael	2022-02-07
1	Jones	Cary	2022-03-15

Sort by Multiple Columns

You can sort by multiple columns in a single step.

Tip: To perform multi-column sorts, search for `Sort rows` in the Transform Builder and specify the columns manually.

In this example, you wish to sort the lastName column and then descending order of the firstName column.

Transformation:

This transformation looks like the following:

Transformation Name	`Sort rows`
Parameter: Sort by	`lastName`
Parameter: Sort by	`-firstName`

Results:

rowId	lastName	firstName	startDate
2	Abbott	Michael	2022-02-07
3	Jones	Tim	2022-05-01
1	Jones	Cary	2022-03-15

Reshaping Steps

Contents:

You can reshape the row and column structure of your data through a variety of transformations.

In the Transformer page:

- You can create new columns, modify them, and delete them to re-scope the size of your data to the most meaningful information.
- You can reshape your data through pivots and aggregations.
- Nested data in the form of Arrays or Objects (key-value pairs) can be un-nested across columns and rows for easier manipulation. As needed, patterned data can be re-nested through transformations that are easy to select and manipulate.

Tip: When reshaping your data from its original form, you may find it useful to build your pivots and aggregations as separate recipes created off of your current recipe. In this manner, you can preserve the original structure and explore more significant transformations as needed.

Transformations to Reshape Your Dataset

Recipe steps can change the number of rows in the dataset and apply wider impacts to your dataset and its samples.

These **reshaping steps** include the following transformations:

Transformation	Documentation
Splitrows	*Initial Parsing Steps*
Expand Arrays into Rows	*Working with Arrays*
Filter Rows (keep or delete)	*Remove Data*
Pivot Table	*Pivot Data*
Unpivot Columns	*Unpivot Columns*
Join Datasets	*Join Window*
Union Datasets	*Union Page*
Select **Lookup** from the column menu	*Lookup Wizard*
Remove Duplicate Rows	*Remove Data*

Samples and reshaping your datasets

When one of these transformations is applied and rows are removed from your dataset:

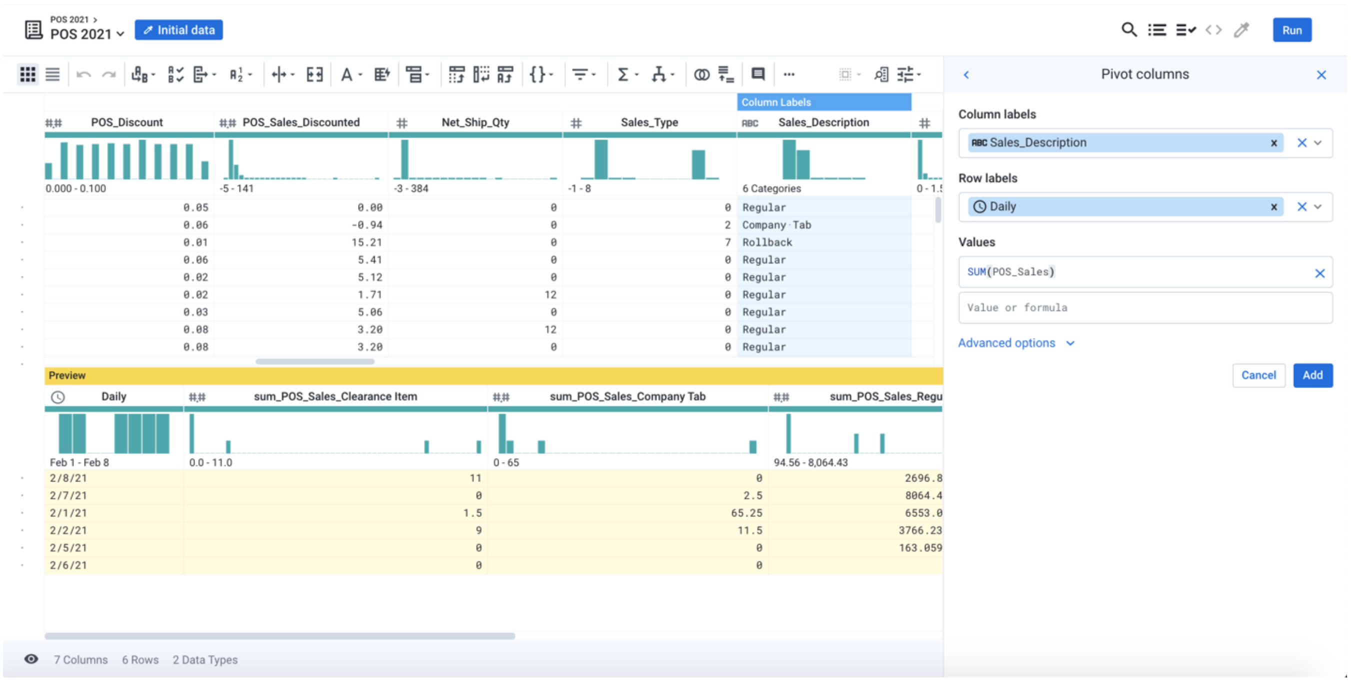
- Any samples generated before the step was added are invalidated and cannot be used.
- If you edit steps in your recipe before this added transformation, any samples that you generated after the step are invalidated and cannot be used.
- A valid initial sample is always available for use.

For more information, see *Samples Panel* .

Build Pivot Tables

You can reshape your data by building pivot tables. Pivot tables are useful when you want to calculate aggregation functions, such as sums, maximums, and averages for one or more columns of data.

In the following example, the data is reshaped to include the sum of `POS_Sales` for each distinct value in the `Daily` column across the values in the `Sales_Description` column:



POS_Discount	POS_Sales_Discounted	Net_Ship_Qty	Sales_Type	Sales_Description
0.000 - 0.100	-5 - 141	-3 - 384	-1 - 8	6 Categories
0.05	0.00	0	0	Regular
0.06	-0.94	0	2	Company Tab
0.01	15.21	0	7	Rollback
0.06	5.41	0	0	Regular
0.02	5.12	0	0	Regular
0.02	1.71	12	0	Regular
0.03	5.06	0	0	Regular
0.08	3.20	12	0	Regular
0.08	3.20	0	0	Regular

Daily	sum_POS_Sales_Clearance Item	sum_POS_Sales_Company Tab	sum_POS_Sales_Regu
Feb 1 - Feb 8	0.0 - 11.0	0 - 65	94.56 - 8,064.43
2/8/21	11	0	2696.8
2/7/21	0	2.5	8064.4
2/1/21	1.5	65.25	6553.0
2/2/21	9	11.5	3766.23
2/5/21	0	0	163.059
2/6/21	0	0	

Figure: Reshape your data using pivot tables

For more information, see *Pivot Data*.

Create Aggregations

An **aggregation** is a computation across a grouped set of rows. Designer Cloud Powered by Trifacta Enterprise Edition provides a wide range of aggregation functions that you can apply:

- To an entire column (called a **flat aggregation**)
- To generate a new column
- To use to reshape your entire table

For more information, see *Create Aggregations*.

Nest and Unnest

You can combine data in separate columns into single-column values stored in Arrays or Objects (maps). Similarly, data from an Array or Object column can be converted into new rows or columns based on the keys in the source data. For more information:

- *Nest Your Data*
- *Unnest Your Data*

Select Columns

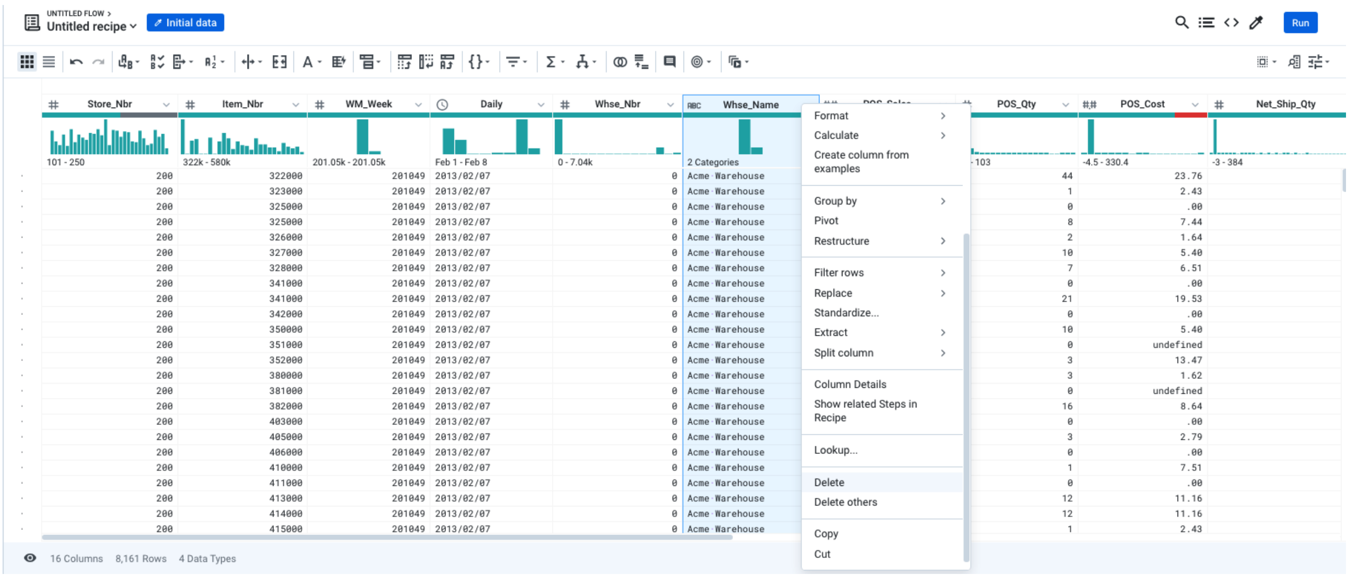
You can select a set of columns to replace the current dataset completely. See *Select*.

Delete Columns

You can reshape your data by deleting unwanted columns in the dataset. You can delete a single column or multiple columns.

- To delete a column from your dataset, click the required 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.



Store_Nbr	Item_Nbr	WM_Week	Daily	Whse_Nbr	Whse_Name	POS_Qty	POS_Cost	Net_Ship_Qty
101 - 250	322k - 580k	201.05k - 201.05k	Feb 1 - Feb 8	0 - 7.04k	2 Categories	- 103	-4.5 - 330.4	-3 - 384
200	322000	201049	2013/02/07	0	Acme Warehouse	44	23.76	
200	323000	201049	2013/02/07	0	Acme Warehouse	1	2.43	
200	325000	201049	2013/02/07	0	Acme Warehouse	0	.00	
200	325000	201049	2013/02/07	0	Acme Warehouse	8	7.44	
200	326000	201049	2013/02/07	0	Acme Warehouse	2	1.64	
200	327000	201049	2013/02/07	0	Acme Warehouse	10	5.40	
200	328000	201049	2013/02/07	0	Acme Warehouse	7	6.51	
200	341000	201049	2013/02/07	0	Acme Warehouse	0	.00	
200	341000	201049	2013/02/07	0	Acme Warehouse	21	19.53	
200	342000	201049	2013/02/07	0	Acme Warehouse	0	.00	
200	350000	201049	2013/02/07	0	Acme Warehouse	10	5.40	
200	351000	201049	2013/02/07	0	Acme Warehouse	0	undefined	
200	352000	201049	2013/02/07	0	Acme Warehouse	3	13.47	
200	380000	201049	2013/02/07	0	Acme Warehouse	3	1.62	
200	381000	201049	2013/02/07	0	Acme Warehouse	0	undefined	
200	382000	201049	2013/02/07	0	Acme Warehouse	16	8.64	
200	403000	201049	2013/02/07	0	Acme Warehouse	0	.00	
200	405000	201049	2013/02/07	0	Acme Warehouse	3	2.79	
200	406000	201049	2013/02/07	0	Acme Warehouse	0	.00	
200	410000	201049	2013/02/07	0	Acme Warehouse	1	7.51	
200	411000	201049	2013/02/07	0	Acme Warehouse	0	.00	
200	413000	201049	2013/02/07	0	Acme Warehouse	12	11.16	
200	414000	201049	2013/02/07	0	Acme Warehouse	12	11.16	
200	415000	201049	2013/02/07	0	Acme Warehouse	1	2.43	

16 Columns 8,161 Rows 4 Data Types

Figure: Reshape your data using Delete columns

The above menu choices get turned into recipe steps that use the `Delete columns` transformation.

Transformation Name	`Delete columns`
Parameter: Columns	`Multiple`
Parameter: Columns	`Whse_Name`

Parameter: Action	`Delete selected columns`

Tip: While using Delete columns transformation, you can use the tilde (~) character between the start and end column names to delete a range of columns.

See *Delete Data*.

Split Columns

You can split a column based on one or more known delimiters or based on index positions in the data. See *Split Column*.

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 Target Schema Mapping

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 target schema mapping, which provides a visual interface for performing these remappings.

NOTE: Target schema mapping 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 Target Schema Mapping*.

Create Aggregations

Contents:

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.

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	`ROUND(AVERAGE(Score),2)`
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	`AVERAGE(TestScore)`
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	`ROUND(average_TestScore,2)`

You may wish to rename the newly generated column to something like `average_TestScorePerStudent` instead.

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.

Nest Your Data

Contents:

In Designer Cloud Powered by Trifacta® Enterprise Edition, you can nest columns into arrays and objects (maps) using a variety of transformations.

Nested data types include the following types of data:

- **Arrays** are lists of values, grouped into a single value. Examples:

```
[1,2,3,4]
```

```
[1,2,[a,b],4]
```

- **Objects** are collections of key-value pairs. Examples:

```
{"make":"Subaru","model":"Outback","year":"2015","color":"black"}
```

```
{"make":"Subaru","model":"Outback","year":{"first":"2015","last":"2021"},"color":"black"}
```

In the above examples:

- The first example is a **flat** or **unnested** Array or Object.
- The first example is a **nested** Array or Object.

Nest Columns into Array

This section provides simple examples of nesting columns into Arrays by extracting values from a column or nesting one or more columns into an Array column.

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:

num1	num2	num3
11	12	13
14	15	16
17	18	19

You can 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`

num1	num2	num3	nest2
11	12	13	[[11,12],13]
14	15	16	[[14,15],16]
17	18	19	[[17,18],19]

Nest Columns into Objects

You can nest multiple columns into a single column of objects using `nest` transform.

This section provides a simple example of nesting columns into a new column of Object data type.

Source:

In the following example, furniture product dimensions are stored in separate columns in cm.

Category	Name	Length_cm	Width_cm	Height_cm
bench	Hooska	118.11	74.93	46.34
lamp	Tansk	30.48	30.48	165.1
bookshelf	Brock	27.94	160.02	201.93
couch	Loafy	95	227	83

Transformation:

Use the `nest` transform to bundle the data into a single column.

Transformation Name	`Nest columns into Objects`
Parameter: Columns	`Length_cm,Width_cm,Height_cm`
Parameter: Nest columns to	`Object`
Parameter: New column name	`'Dimensions'`

Results:

Category	Name	Length_cm	Width_cm	Height_cm	Dimensions
bench	Hooska	118.11	74.93	46.34	{"Length_cm":"118.11","Width_cm":"74.93","Height_cm":"46.34"}
lamp	Tansk	30.48	30.48	165.1	{"Length_cm":"30.48","Width_cm":"30.48","Height_cm":"165.1"}
bookshelf	Brock	27.94	160.02	201.93	{"Length_cm":"27.94","Width_cm":"160.02","Height_cm":"201.93"}
couch	Loafy	95	227	83	{"Length_cm":"95,"Width_cm":"227","Height_cm":"83"}

Hybrids of Arrays and Objects

A single Array or Object can contain other Arrays, Objects, or both. In fact, this structuring is essentially the JSON file format. In this example, you can see how data, when laid out in a single table, can be assembled into a set of JSON records. See *EXAMPLE - Nest JSON Records*.

Unnest Your Data

Contents:

You can unnest Array or Object values into separate rows or columns using the following transformations.

Flatten Array Values into Rows

Array values can be flattened into individual values in separate rows.

This section describes how to flatten the values in an Array into separate rows in your dataset.

Source:

In the following example dataset, students took the same test three times, and their scores were stored in any array in the `Scores` column.

LastName	FirstName	Scores
Adams	Allen	[81,87,83,79]
Burns	Bonnie	[98,94,92,85]
Cannon	Chris	[88,81,85,78]

Transformation:

When the data is imported, you might have to re-type the `Scores` column as an array:

Transformation Name	`Change column data type`
Parameter: Columns	`Scores`
Parameter: New type	`Array`

You can now flatten the `Scores` column data into separate rows:

Transformation Name	`Expand Array into rows`
Parameter: Column	`Scores`

Results:

LastName	FirstName	Scores
Adams	Allen	81
Adams	Allen	87
Adams	Allen	83

Adams	Allen	79
Burns	Bonnie	98
Burns	Bonnie	94
Burns	Bonnie	92
Burns	Bonnie	85
Cannon	Chris	88
Cannon	Chris	81
Cannon	Chris	85
Cannon	Chris	78

Tip: You can use aggregation functions on the above data to complete values like average, minimum, and maximum scores. When these aggregation calculations are grouped by student, you can perform the calculations for each student.

Unnest Array Values into New Columns

You can also split out the individual values in an array into separate columns.

This section describes how to unnest the values in an Array into separate columns in your dataset.

Source:

In the following example dataset, students took the same test three times, and their scores were stored in any array in the `Scores` column.

LastName	FirstName	Scores
Adams	Allen	[81,87,83,79]
Burns	Bonnie	[98,94,92,85]
Cannon	Chris	[88,81,85,78]

Transformation:

When the data is imported, you might have to re-type the `Scores` column as an array:

Transformation Name	`Change column data type`
Parameter: Columns	`Scores`
Parameter: New type	`Array`

You can now unnest the `Scores` column data into separate columns:

Transformation Name	`Unnest Objects into columns`
Parameter: Column	`Scores`
Parameter: Parameter: Paths to elements	`[0]`
Parameter: Parameter: Paths to elements	`[1]`

Parameter: Parameter: Paths to elements	`[2]`
Parameter: Parameter: Paths to elements	`[3]`
Parameter: Remove elements from original	`true`
Parameter: Include original column name	`true`

In the above transformation:

- Each path is specified in a separate row.
  - The `[x]` syntax indicates that the path is the *xth* element of the array.
  - The first element of an array is referenced using `[0]`.
- You can choose to delete the element from the original or not. Deleting the element can be a helpful way of debugging your transformation. If all of the elements are gone, then the transformation is complete.
- If you include the original column name in the output column names, you have some contextual information for the outputs.

Results:

LastName	FirstName	Scores_0	Scores_1	Scores_2	Scores_3
Adams	Allen	81	87	83	79
Burns	Bonnie	98	94	92	85
Cannon	Chris	88	81	85	78

Flatten and Unnest Together

The following example illustrates how flatten and unnest can be used together to reshape your data.

This example illustrates you to use the flatten and unnest transforms.

Source:

You have the following data on student test scores. Scores on individual scores are stored in the `Scores` array, and you need to be able to track each test on a uniquely identifiable row. This example has two goals:

1. One row for each student test
2. Unique identifier for each student-score combination

LastName	FirstName	Scores
Adams	Allen	[81,87,83,79]
Burns	Bonnie	[98,94,92,85]
Cannon	Charles	[88,81,85,78]

Transformation:

When the data is imported from CSV format, you must add a `header` transform and remove the quotes from the `Scores` column:

Transformation Name	`Rename column with row(s)`

Parameter: Option	`Use row(s) as column names`
Parameter: Type	`Use a single row to name columns`
Parameter: Row number	`1`

Transformation Name	`Replace text or pattern`
Parameter: Column	`colScores`
Parameter: Find	`'\"'`
Parameter: Replace with	`''`
Parameter: Match all occurrences	`true`

Validate test date: To begin, you might want to check to see if you have the proper number of test scores for each student. You can use the following transform to calculate the difference between the expected number of elements in the `Scores` array (4) and the actual number:

Transformation Name	`New formula`
Parameter: Formula type	`Single row formula`
Parameter: Formula	`(4 - arraylen(Scores))`
Parameter: New column name	`'numMissingTests'`

When the transform is previewed, you can see in the sample dataset that all tests are included. You might or might not want to include this column in the final dataset, as you might identify missing tests when the recipe is run at scale.

Unique row identifier: The `Scores` array must be broken out into individual rows for each test. However, there is no unique identifier for the row to track individual tests. In theory, you could use the combination of `LastName-FirstName-Scores` values to do so, but if a student recorded the same score twice, your dataset has duplicate rows. In the following transform, you create a parallel array called `Tests`, which contains an index array for the number of values in the `Scores` column. Index values start at `0`:

Transformation Name	`New formula`
Parameter: Formula type	`Single row formula`
Parameter: Formula	`range(0,arraylen(Scores))`
Parameter: New column name	`'Tests'`

Also, we will want to create an identifier for the source row using the `sourcerownumber` function:

Transformation Name	`New formula`
Parameter: Formula type	`Single row formula`
Parameter: Formula	`sourcerownumber()`
Parameter: New column name	`'orderIndex'`

One row for each student test: Your data should look like the following:

LastName	FirstName	Scores	Tests	orderIndex
Adams	Allen	[81,87,83,79]	[0,1,2,3]	2
Burns	Bonnie	[98,94,92,85]	[0,1,2,3]	3
Cannon	Charles	[88,81,85,78]	[0,1,2,3]	4

Now, you want to bring together the `Tests` and `Scores` arrays into a single nested array using the `arrayzip` fun ction:

Transformation Name	`New formula`
Parameter: Formula type	`Single row formula`
Parameter: Formula	`arrayzip([Tests,Scores])`

Your dataset has been changed:

LastName	FirstName	Scores	Tests	orderIndex	column1
Adams	Allen	[81,87,83,79]	[0,1,2,3]	2	[[0,81],[1,87],[2,83],[3,79]]
Adams	Bonnie	[98,94,92,85]	[0,1,2,3]	3	[[0,98],[1,94],[2,92],[3,85]]
Cannon	Charles	[88,81,85,78]	[0,1,2,3]	4	[[0,88],[1,81],[2,85],[3,78]]

Use the following to unpack the nested array:

Transformation Name	`Expand arrays to rows`
Parameter: Column	`column1`

Each test-score combination is now broken out into a separate row. The nested Test-Score combinations must be broken out into separate columns using the following:

Transformation Name	`Unnest Objects into columns`
Parameter: Column	`column1`
Parameter: Paths to elements	`'[0]','[1]'`

After you delete `column1`, which is no longer needed you should rename the two generated columns:

Transformation Name	`Rename columns`
Parameter: Option	`Manual rename`
Parameter: Column	`column_0`
Parameter: New column name	`'TestNum'`

Transformation Name	`Rename columns`
Parameter: Option	`Manual rename`
Parameter: Column	`column_1`

Parameter: New column name	`'TestScore'`

Unique row identifier: You can do one more step to create unique test identifiers, which identify the specific test for each student. The following uses the original row identifier `OrderIndex` as an identifier for the student and the `TestNumber` value to create the `TestId` column value:

Transformation Name	`New formula`
Parameter: Formula type	`Single row formula`
Parameter: Formula	`(orderIndex * 10) + TestNum`
Parameter: New column name	`'TestId'`

The above are integer values. To make your identifiers look prettier, you might add the following:

Transformation Name	`Merge columns`
Parameter: Columns	`'TestId00','TestId'`

Extending: You might want to generate some summary statistical information on this dataset. For example, you might be interested in calculating each student's average test score. This step requires figuring out how to properly group the test values. In this case, you cannot group by the `LastName` value, and when executed at scale, there might be collisions between first names when this recipe is run at scale. So, you might need to create a kind of primary key using the following:

Transformation Name	`Merge columns`
Parameter: Columns	`'LastName','FirstName'`
Parameter: Separator	`'-'`
Parameter: New column name	`'studentId'`

You can now use this as a grouping parameter for your calculation:

Transformation Name	`New formula`
Parameter: Formula type	`Single row formula`
Parameter: Formula	`average(TestScore)`
Parameter: Group rows by	`studentId`
Parameter: New column name	`'avg_TestScore'`

Results:

After you delete unnecessary columns and move your columns around, the dataset should look like the following:

TestId	LastName	FirstName	TestNum	TestScore	studentId	avg_TestScore
TestId0021	Adams	Allen	0	81	Adams-Allen	82.5
TestId0022	Adams	Allen	1	87	Adams-Allen	82.5

TestId0023	Adams	Allen	2	83	Adams-Allen	82.5
TestId0024	Adams	Allen	3	79	Adams-Allen	82.5
TestId0031	Adams	Bonnie	0	98	Adams-Bonnie	92.25
TestId0032	Adams	Bonnie	1	94	Adams-Bonnie	92.25
TestId0033	Adams	Bonnie	2	92	Adams-Bonnie	92.25
TestId0034	Adams	Bonnie	3	85	Adams-Bonnie	92.25
TestId0041	Cannon	Chris	0	88	Cannon-Chris	83
TestId0042	Cannon	Chris	1	81	Cannon-Chris	83
TestId0043	Cannon	Chris	2	85	Cannon-Chris	83
TestId0044	Cannon	Chris	3	78	Cannon-Chris	83

Unnest Object Values into New Columns

This example shows how you can unnest Object data into separate columns. The example contains vehicle identifiers, and the `Properties` column contains key-value pairs describing characteristics of each vehicle.

This example shows how you can unpack data nested in an Object into separate columns.

Source:

You have the following information on used cars. The `VIN` column contains vehicle identifiers, and the `Properties` column contains key-value pairs describing characteristics of each vehicle. You want to unpack this data into separate columns.

VIN	Properties
XX3 JT4522	year=2004,make=Subaru,model=Impreza,color=green,mileage=125422,cost=3199
HT4 UJ9122	year=2006,make=VW,model=Passat,color=silver,mileage=102941,cost=4599
KC2 WZ9231	year=2009,make=GMC,model=Yukon,color=black,mileage=68213,cost=12899
LL8 UH4921	year=2011,make=BMW,model=328i,color=brown,mileage=57212,cost=16999

Transformation:

Add the following transformation, which identifies all of the key values in the column as beginning with alphabetical characters.

- The `valueafter` string identifies where the corresponding value begins after the key.
- The `delimiter` string indicates the end of each key-value pair.

Transformation Name	`Convert keys/values into Objects`
Parameter: Column	`Properties`
Parameter: Key	`` `{alpha}+` ``
Parameter: Separator between key and value	`` `=` ``
Parameter: Delimiter between pair	`','`

Now that the Object of values has been created, you can use the `unnest` transform to unpack this mapped data. In the following, each key is specified, which results in separate columns headed by the named key:

> **NOTE**: Each key must be entered on a separate line in the Path to elements area.

Transformation Name	`Unnest Objects into columns`
Parameter: Column	`extractkv_Properties`
Parameter: Paths to elements	`year`
Parameter: Paths to elements	`make`
Parameter: Paths to elements	`model`
Parameter: Paths to elements	`color`
Parameter: Paths to elements	`mileage`
Parameter: Paths to elements	`cost`

Results:

When you delete the unnecessary Properties columns, the dataset now looks like the following:

VIN	year	make	model	color	mileage	cost
XX3 JT4522	2004	Subaru	Impreza	green	125422	3199
HT4 UJ9122	2006	VW	Passat	silver	102941	4599
KC2 WZ9231	2009	GMC	Yukon	black	68213	12899
LL8 UH4921	2011	BMW	328i	brown	57212	16999

Extract a Set of Values

This example shows how to extract values (for example, hashtag values) from a column and convert them into a column of arrays.

In this example, you extract one or more values from a source column and assemble them in an Array column.

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 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 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.

Transformation Name	`Extract matches into Array`
Parameter: Column	`customer_tweets`
Parameter: Pattern matching elements in the list	`` `{hashtag}` ``
Parameter: New column name	`Hashtag tweets`

Then, the source column can be deleted.

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"]

Pivot Data

Contents:

A **pivot table** summaries 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: *"Show me the value of this formula computed by each row value for every value represented in the generated table."*

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 and an aggregate function. 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.

Tip: In your flows, you may find it useful to create your pivot tables in independent recipes that are chained from your primary recipe.

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 COOKIES DRP	0	0	0	120	30
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	`SUMIF(POS_Sales, Store_Nbr<4)`
Parameter: Max number of columns to create	`500`

Most aggregation functions have a conditional (`*IF`) variant.

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 SN CRACKERS /PROCES	0	0	0	0	0
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 apply an aggregate function within a single column.

Values to Columns

You can convert individual values from a column to separate columns in your dataset.

Individual row values become the headers of the new columns that are created. If the value represented by the column is present in the original data, a consistent value is inserted (For example, `Yes`) in the corresponding row in the newly created column. If it's missing, another value is inserted (For example, `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. These values can also be used for filtering out rows as needed.

Source:

The following example contains transactional information about products sold in individual stores. In the transformation, the values in the `Store_Nbr` column are converted to columns.

Daily	Store_Nbr	POS_Sales	POS_Qty	PRODUCT_DESC
2/8/13	1	70	7	ACME LAWN GARDEN BAG CLEAR
2/7/13	2	10.62	9	ACME COOKIES CHOC CHIP
2/7/13	2	0	0	ACME SANDWICH BAG
2/7/13	2	7.08	6	ACME SODAS SALTED
2/7/13	2	3.92	2	ACME SCENTED OIL REFILL-CTRY SUN
2/7/13	2	13.44	7	ACME LARGE FUDGE GRAHAMS COOKIES
2/7/13	2	0	0	ACME SUGAR ICE WAFERS VANILLA
2/7/13	3	3.16	2	ACME ZOO ANIMAL FRUIT SNACKS 6'S
2/7/13	3	3.16	2	ACME WAFERS SUGER ICE
2/7/13	3	3.16	2	ACME SCENTED OIL REFILL-CTRY SUN
2/7/13	3	6.32	4	ACME RICE CRACKERS ONION
2/2/13	9	150	30	ACME FROSTED OATMEAL COOKIE SQUA
2/2/13	9	3.5	2	ACME FRUIT SNACK CASTLE ADVENTRS
2/2/13	9	90	9	ACME COOKIES CHOC CHIP
2/2/13	9	30	6	ACME ASSORTED COOKIES DRP
2/2/13	9	70	7	ACME KITCHEN BAG
2/2/13	9	170	17	ACME SNACK BAGS RESEALABLE
2/2/13	9	20	4	ACME CHEDDARY SN CRACKERS/PROCES
2/2/13	9	6.5	2	ACME RICE CRACKERS TERIYAKI
2/2/13	9	1.5	3	ACME COOKIE MAPLE LEAF CREME
2/2/13	9	30	6	ACME RICE CHIPS CHEDDAR
2/1/13	7	190	38	ACME FROSTED OATMEAL COOKIE SQUA
2/1/13	7	20	2	ACME COOKIES CHOC CHIP
2/1/13	7	10	1	ACME DIGESTIVE RICH TEA BISCUITS
2/1/13	7	120	24	ACME ASSORTED COOKIES DRP
2/1/13	7	120	12	ACME KITCHEN BAG

2/1/13	7	90	9	ACME SNACK BAGS RESEALABLE
2/1/13	7	10	1	ACME FUDGE MINT COOKIES SQUARES
2/1/13	7	9.5	19	ACME CHEDDARY SN CRACKERS/PROCES
2/1/13	7	10	1	ACME COOKIES MAPLE CREAM
2/1/13	7	40	8	ACME COOKIE MAPLE LEAF CREME

Transformation:

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	PRODUCT_DESC	column_1	column_2	column_3	column_9	colu
2/8/13	1	70	7	ACME LAWN GARDEN BAG CLEAR	Yes				
2/7/13	2	10.62	9	ACME COOKIES CHOC CHIP		Yes			
2/7/13	2	0	0	ACME SANDWICH BAG		Yes			
2/7/13	2	7.08	6	ACME SODAS SALTED		Yes			
2/7/13	2	3.92	2	ACME SCENTED OIL REFILL-CTRY SUN		Yes			
2/7/13	2	13.44	7	ACME LARGE FUDGE GRAHAMS COOKIES		Yes			
2/7/13	2	0	0	ACME SUGAR ICE WAFERS VANILLA		Yes			
2/7/13	3	3.16	2	ACME ZOO ANIMAL FRUIT SNACKS 6'S			Yes		
2/7/13	3	3.16	2	ACME WAFERS SUGER ICE			Yes		

2/7/13	3	3.16	2	ACME SCENTED OIL REFILL-CTRY SUN			Yes		
2/7/13	3	6.32	4	ACME RICE CRACKERS ONION			Yes		
2/2/13	9	150	30	ACME FROSTED OATMEAL COOKIE SQUA				Yes	
2/2/13	9	3.5	2	ACME FRUIT SNACK CASTLE ADVENTRS				Yes	
2/2/13	9	90	9	ACME COOKIES CHOC CHIP				Yes	
2/2/13	9	30	6	ACME ASSORTED COOKIES DRP				Yes	
2/2/13	9	70	7	ACME KITCHEN BAG				Yes	
2/2/13	9	170	17	ACME SNACK BAGS RESEALABLE				Yes	
2/2/13	9	20	4	ACME CHEDDARY SN CRACKERS /PROCES				Yes	
2/2/13	9	6.5	2	ACME RICE CRACKERS TERIYAKI				Yes	
2/2/13	9	1.5	3	ACME COOKIE MAPLE LEAF CREME				Yes	
2/2/13	9	30	6	ACME RICE CHIPS CHEDDAR				Yes	
2/1/13	7	190	38	ACME FROSTED OATMEAL COOKIE SQUA					Yes
2/1/13	7	20	2	ACME COOKIES CHOC CHIP					Yes
2/1/13	7	10	1	ACME DIGESTIVE RICH TEA BISCUITS					Yes
2/1/13	7	120	24	ACME ASSORTED COOKIES DRP					Yes
2/1/13	7	120	12	ACME KITCHEN BAG					Yes
2/1/13	7	90	9	ACME SNACK BAGS RESEALABLE					Yes
2/1/13	7	10	1	ACME FUDGE MINT COOKIES SQUARES					Yes
2/1/13	7	9.5	19	ACME CHEDDARY SN CRACKERS /PROCES					Yes
2/1/13	7	10	1	ACME COOKIES MAPLE CREAM					Yes
2/1/13	7	40	8	ACME COOKIE MAPLE LEAF CREME					Yes

Unpivot Columns

Contents:

You can convert columns into rows of values. A conversion transformations 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.

> **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.

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:

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	`ROLLINGAVERAGE (Sales, 0,1)`
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	`SESSION (TimeStamp, 5, minute)`
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	`(Order - PREV(Order, 1)) / PREV(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	`NUMFORMAT(ChangeinOrder, '##.##')`

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

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	`IF(OrdersDay == '' || ISNULL(OrdersDay), '0', $col)`

You can see the values of Friday is taken for Saturday and Sunday and filled it accordingly as per the `FILL` functi on.

Transformation Name	`Edit with formula`
Parameter: Column	`OrdersTotal`
Parameter: Formula	`IF (OrdersDay == '0', FILL (OrdersTotal, -1,0),$col)`
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

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

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

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	`ROLLINGSUM (Sales, 0,1)`
Parameter: Formulas	`ROLLINGMAXDATE (Order_date, 0,1)`
Parameter: Formulas	`ROLLINGMINDATE (Order_date, 0, 1)`
Parameter: Formulas	`ROLLINGMODEDATE (Order_date, 0,1)`
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

Working with Arrays

Contents:

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
LIST Function	Extracts the set of values from a column into an array stored in a new column. This function is typically part of an aggregation.
UNIQUE Function	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.
LISTIF Function	Returns list of all values in a column for rows that match a specified condition.
ROLLINGLIST Function	Computes the rolling list of values forward or backward of the current row within the specified column and returns an array of these values.
RANGE Function	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
IN Function	Returns `true` if the first parameter is contained in the array of values in the second parameter.
ARRAYELEMENTAT Function	Computes the 0-based index value for an array element in the specified column, array literal, or function that returns an array.
ARRAYLEN Function	Computes the number of elements in the arrays in the specified column, array literal, or function that returns an array.
ARRAYUNIQUE Function	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
LISTSUM Function	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.
LISTMAX Function	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.
LISTMIN Function	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.
LISTAVERAGE Function	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.
LISTVAR Function	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.
LISTSTDEV Function	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.
LISTMODE Function	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
ARRAYCONCAT Function	Combines the elements of one array with another, listing all elements of the first array before listing all elements of the second array.
ARRAYCROSS Function	Generates a nested array containing the cross-product of all elements in two or more arrays.
ARRAYINTERSECT Function	Generates an array containing all elements that appear in multiple input arrays, referenced as column names or array literals.
ARRAYSTOMAP Function	Combines one array containing keys and another array containing values into an Object of key-value pairs.
ARRAYZIP Function	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	`nest2`

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 Objects

Contents:

This section describes how to work with the Object data type. An **object** (or **map**) is a set of key-value pairs.

Any individual value can contain another set of key-value pairs, which enables the creation of nested data objects.

> **Tip:** As one of its values, an object can contain an array, which can in turn contain other objects or arrays. In this manner, you can created nested hybrid data objects by combining these two data types.

Structure of Objects

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. A simple example:

```
{"Texas":"TX"},
{"New York":"NY"},
{"California":"CA"},
```

Notes:

- The above example features repeated data in a repeated format across each line.
- Effectively, these are records of data, mapping a state's formal name (e.g. `Texas`) to its two-letter abbreviation (e.g. TX).
- Data structures of the Object data type can be more complex.

> **NOTE:** The Designer Cloud application can recognize up to 250 unique keys in a column of Object data type.

Import Objects

Import Object columns

When a column is identified as a set of key-value pairs during import, the column may be typed as an Object data type column. These key-value pairs can be extracted and converted into rows and columns in the dataset using transformations and functions in the application.

Import JSON files

The Object data type can be the basis for entire JSON files. When JSON files are formatted in a way that can be parsed by the the Designer Cloud application , they can be converted into tabular format as part of the import process. If the preceding example is the entire file, the conversion process may display the dataset in the Transformer page as the following:

column1	column2
Texas	TX
New York	NY
California	CA

For more information, see *Working with JSON v2*.

Create Objects

Within the Designer Cloud application , you can use functions and transformations to create columns that are recognized as Object data type.

Create by nesting

You can nest multiple columns into a single column of objects using the `nest` transform.

This section provides a simple example of nesting columns into a new column of Object data type.

Source:

In the following example, furniture product dimensions are stored in separate columns in cm.

Category	Name	Length_cm	Width_cm	Height_cm
bench	Hooska	118.11	74.93	46.34
lamp	Tansk	30.48	30.48	165.1
bookshelf	Brock	27.94	160.02	201.93
couch	Loafy	95	227	83

Transformation:

Use the `nest` transform to bundle the data into a single column.

Transformation Name	`Nest columns into Objects`
Parameter: Columns	`Length_cm,Width_cm,Height_cm`
Parameter: Nest columns to	`Object`
Parameter: New column name	`'Dimensions'`

Results:

Category	Name	Length_cm	Width_cm	Height_cm	Dimensions
bench	Hooska	118.11	74.93	46.34	{"Length_cm":"118.11","Width_cm":"74.93","Height_cm":"46.34"}
lamp	Tansk	30.48	30.48	165.1	{"Length_cm":"30.48","Width_cm":"30.48","Height_cm":"165.1"}

bookshelf	Brock	27.94	160.02	201.93	{"Length_cm":"27.94","Width_cm":"160.02","Height_cm":"201.93"}
couch	Loafy	95	227	83	{"Length_cm":"95,"Width_cm":"227","Height_cm":"83"}

Create by Filtering Strings

You can create objects by filtering strings by using the `FILTEROBJECT` function.

You can create nested objects by filtering strings. In this example, column headers and column values are nested into a single entity in a new column of Object data type.

Functions:

Item	Description
FILTEROBJECT Function	Filters the keys and values from an Object data type column based on a specified key value.
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.

Source:

The following table shows a series of requests for inventory on three separate products. These are rolling requests, so inventory levels in the subsequent request are decreased based on the previous request.

date	reqProdId	reqValue	prodA	prodB	prodC
5/10/21	prodA	10	90	100	100
5/10/21	prodC	20	90	100	80
5/10/21	prodA	15	75	100	80
5/11/21	prodB	25	75	75	80
5/11/21	prodA	5	70	75	80
5/11/21	prodC	30	70	75	50
5/12/21	prodB	10	70	65	50

You must create a column containing the request information and the inventory level information for the requested product after the request has been fulfilled.

Transformation:

The five data columns must be nested into an Object. The generated column is called `inventoryLevels`.

Transformation Name	`Nest columns into Objects`
Parameter: Columns	`reqProdId,reqValue,prodA,prodB,prodC`
Parameter: Nest columns to	`Object`
Parameter: New column name	`inventoryLevels`

You can then build the inventory response column (`inventoryResponse`) using the FILTEROBJECT function:

Transformation Name	`New formula`
Parameter: Formula type	`Single row formula`

Parameter: Formula	`filterobject(parseobject(inventoryRequest), ['reqProdId','reqValue',reqProdId])`
Parameter: New column name	`inventoryResponse`

Results:

The `inventoryResponse` column contains the request information and the response information after the request has been fulfilled.

date	reqProdId	reqValue	prodA	prodB	prodC	inventoryLevels	inventoryResponse
5/10/21	prodA	10	90	100	100	{"reqProdId":"prodA","reqValue":"10","prodA":"90","prodB":"100","prodC":"100"}	{"reqProdId":"prodA","reqValue":"10","prodA":"90"}
5/10/21	prodC	20	90	100	80	{"reqProdId":"prodC","reqValue":"20","prodA":"90","prodB":"100","prodC":"80"}	{"reqProdId":"prodC","reqValue":"20","prodC":"80"}
5/10/21	prodA	15	75	100	80	{"reqProdId":"prodA","reqValue":"15","prodA":"75","prodB":"100","prodC":"80"}	{"reqProdId":"prodA","reqValue":"15","prodA":"75"}
5/11/21	prodB	25	75	75	80	{"reqProdId":"prodB","reqValue":"25","prodA":"75","prodB":"75","prodC":"80"}	{"reqProdId":"prodB","reqValue":"25","prodB":"75"}
5/11/21	prodA	5	70	75	80	{"reqProdId":"prodA","reqValue":"5","prodA":"70","prodB":"75","prodC":"80"}	{"reqProdId":"prodA","reqValue":"5","prodA":"70"}
5/11/21	prodC	30	70	75	50	{"reqProdId":"prodC","reqValue":"30","prodA":"70","prodB":"75","prodC":"50"}	{"reqProdId":"prodC","reqValue":"30","prodC":"50"}
5/12/21	prodB	10	70	65	50	{"reqProdId":"prodB","reqValue":"10","prodA":"70","prodB":"65","prodC":"50"}	{"reqProdId":"prodB","reqValue":"10","prodB":"65"}

Convert from Arrays

You can create objects by converting two arrays of key value pairs by using the the `ARRAYSTOMAP` function.

This example illustrates how to use the ARRAYSTOMAP and KEYS functions to convert values in Array or Object data type of key-value pairs.

Functions:

Item	Description
ARRAYSTOMAP Function	Combines one array containing keys and another array containing values into an Object of key-value pairs.
KEYS Function	Extracts the key values from an Object data type column and stores them in an array of String values.

Source:

Your dataset contains master product data with product properties stored in two arrays of keys and values.

ProdId	ProdCategory	ProdName	ProdKeys	ProdProperties
S001	Shirts	Crew Neck T-Shirt	["type", "color", "fabric", "sizes"]	["crew","blue","cotton","S,M,L","in stock","padded"]

S002	Shirts	V-Neck T-Shirt	["type", "color", "fabric", "sizes"]	["v-neck","white","blend","S,M,L,XL","in stock","discount - seasonal"]
S003	Shirts	Tanktop	["type", "color", "fabric", "sizes"]	["tank","red","mesh","XS,S,M","discount - clearance","in stock"]
S004	Shirts	Turtleneck	["type", "color", "fabric", "sizes"]	["turtle","black","cotton","M,L,XL","out of stock","padded"]

Transformation:

When the above data is loaded into the Transformer page, you might need to clean up the two array columns.

Using the following transform, you can map the first element of the first array as a key for the first element of the second, which is its value. You might notice that the number of keys and the number of values are not consistent. For the extra elements in the second array, the default key of `ProdMiscProperties` is used:

Transformation Name	`New formula`
Parameter: Formula type	`Single row formula`
Parameter: Formula	`ARRAYSTOMAP(ProdProperties, ProdValues, 'ProdMiscProperties')`
Parameter: New column name	`'prodPropertyMap'`

You can now use the following steps to generate a new version of the keys:

Transformation Name	`Delete columns`
Parameter: Columns	`ProdKeys`
Parameter: Action	`Delete selected columns`

Transformation Name	`New formula`
Parameter: Formula type	`Single row formula`
Parameter: Formula	`KEYS(prodPropertyMap)`
Parameter: New column name	`'ProdKeys'`

Results:

ProdId	ProdCategory	ProdName	ProdKeys	ProdProperties	prodPropertyMap
S001	Shirts	Crew Neck T-Shirt	["type", "color", "fabric", "sizes","ProdMiscProperties"]	["crew","blue","cotton","S,M,L","in stock","padded"]	`{ "type": [ "crew" ], "color": [ "blue" ], "fabric": [ "cotton" ], "sizes": [ "S,M,L" ], "ProdMiscProperties": [ "in stock", "padded" ] }`
S002	Shirts	V-Neck T-Shirt	["type", "color", "fabric", "sizes","ProdMiscProperties"]	["v-neck","white","blend","S,M,L,XL"," in stock","discount - seasonal"]	`{ "type": [ "v-neck" ],`

					"color": ["white"], "fabric": ["blend"], "sizes": ["S,M,L,XL"], "ProdMiscProperties": ["in stock", "discount - seasonal"] }
S003	Shirts	Tanktop	["type", "color", "fabric", "sizes","ProdMiscProperties"]	["tank","red","mesh","XS,S,M"," discount - clearance","in stock"]	{ "type": ["tank"], "color": ["red"], "fabric": ["mesh"], "sizes": ["XS,S,M"], "ProdMiscProperties": ["discount - clearance", "in stock"] }
S004	Shirts	Turtleneck	["type", "color", "fabric", "sizes","ProdMiscProperties"]	["turtle","black","cotton","M,L,XL","out of stock","padded"]	{ "type": ["turtle"], "color": ["black"], "fabric": ["cotton"], "sizes": ["M,L,XL"], "ProdMiscProperties": ["out of stock", "padded"] }

Read from Objects

When a column is recognized as an Object data type, you can apply transformations to extract the keys, the values, or both from the column for use in a new column. You can use pattern-based matching to acquire the values of interest for further analysis or cleaning.

Extract Keys

You can extract keys from objects from the Object data and and stores them in an array of String values.

You can extract the keys from an Object column into an Array of String values.

Functions:

Item	Description
KEYS Function	Extracts the key values from an Object data type column and stores them in an array of String values.

Source:

The following dataset contains configuration blocks for individual features, each of which has a different configuration. These example blocks are of Object type.

Tip: In the following example configuration, the **keys** are the values on the left (e.g. `enabled`, `maxRows`, and `maxCols`), while the **values** for those keys are on the right side.

Code formatting has been applied to the Object data to improve legibility.

FeatureName	Configuration

Whiz Widget	`{`
	`"enabled": "true",`
	`"maxRows": "1000",`
	`"maxCols": "100"`
	`}`

Magic Button	`{`
	`"enabled": "false",`
	`"startDirectory": "/home",`
	`"maxDepth": "15"`
	`}`

Happy Path Finder	`{`
	`"enabled": "true"`
	`}`

Transformation:

The following transformation extracts the keys from the Object data in the `Configuration` column.

Transformation Name	`New formula`
Parameter: Formula type	`Single row formula`
Parameter: Formula	`keys(Configuration)`
Parameter: New column name	`'keys_Configuration'`

Results:

The `keys_Configuration` column contains the arrays of the key values.

FeatureName	Configuration	keys_Configuration

Whiz Widget	`{`
	`"enabled": "true",`
	`"maxRows": "1000",`
	`"maxCols": "100"`
	`}`
		`["enabled","maxRows","maxCols"]`

Magic Button	`{`
	`"enabled": "false",`
	`"startDirectory": "/home",`
	`"maxDepth": "15"`
	`}`
		`["enabled","startDirectory","maxDepth"]`

Happy Path Finder	`{ "enabled": "true" }`	`["enabled"]`

Extract Object Values

You can extract an object's values in to a new column.

This simple example demonstrates how to extract nested values from Object elements into a separate column.

Source:

For example, suppose your restaurant dataset includes a set of characteristics in the `restFeatures` column in the following JSON format, from which you are interested in the total number of seats in the restaurant.

This example contains the data for a single restaurant, formatted as regular JSON, for simplicity:

```
{
  "Credit":"Y",
  "Accessible":"Y",
  "Restrooms":"Y",
  "EatIn":"Y",
  "ToGo":"N",
  "AlcoholBeer":"Y",
  "AlcoholHard":"N",
  "TotalTables":"10",
  "TotalTableSeats":"36",
  "Counter":"Y",
  "CounterSeats":"8"
}
```

Transformation:

You can use the following transformation to extract the values from `TotalTableSeats` and `CounterSeats` into separate columns:

> **NOTE:** Change the column type to Object before applying the following transformation.

> **NOTE**: Each key must be entered on a separate line in the Path to elements area.

Transformation Name	`Unnest Objects into columns`
Parameter: Column	`restFeatures`
Parameter: Paths to elements	`TotalTableSeats`
Parameter: Paths to elements	`CounterSeats`
Parameter: Include original column name	`Selected`

Results:

restFeatures_TotalTable Seats	restFeatures_CounterSeats

36	8

After converting into separate columns, you can perform a simple sum of the `TotalTableSeats` and `CounterSeats` columns to determine the total number of seats in the restaurant.

The final table looks like the following:

restFeatures_TotalTable Seats	restFeatures_CounterSeats	TotalSeats_Restaurant
36	8	44

Convert to String

Depending on the use, it may be easier to work with your objects as String values. While Strings have no inherent structure, they do have a wide range of functions that you can use to find and extract information from the values. Some useful functions:

Function	Description
FIND Function	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.
RIGHTFIND Function	Returns the index value in the input string where the last instance of a matching string is located. Search is conducted right-to-left.
FINDNTH Function	Returns the position of the nth occurrence of a letter or pattern in the input string where a specified matching string is located in the provided column. You can search either from left or right.
SUBSTITUTE Function	Replaces found string literal or pattern or column with a string, column, or function returning strings.

Unnest Objects

You can unnest the object data type to create new rows or columns based on the keys in the source data. The following example shows how to unnest object values into separate columns.

This example shows how you can unpack data nested in an Object into separate columns.

Source:

You have the following information on used cars. The `VIN` column contains vehicle identifiers, and the `Properties` column contains key-value pairs describing characteristics of each vehicle. You want to unpack this data into separate columns.

VIN	Properties
XX3 JT4522	year=2004,make=Subaru,model=Impreza,color=green,mileage=125422,cost=3199
HT4 UJ9122	year=2006,make=VW,model=Passat,color=silver,mileage=102941,cost=4599
KC2 WZ9231	year=2009,make=GMC,model=Yukon,color=black,mileage=68213,cost=12899
LL8 UH4921	year=2011,make=BMW,model=328i,color=brown,mileage=57212,cost=16999

Transformation:

Add the following transformation, which identifies all of the key values in the column as beginning with alphabetical characters.

- The `valueafter` string identifies where the corresponding value begins after the key.
- The `delimiter` string indicates the end of each key-value pair.

Transformation Name	`Convert keys/values into Objects`
Parameter: Column	`Properties`
Parameter: Key	`` `{alpha}+` ``
Parameter: Separator between key and value	`` `=` ``
Parameter: Delimiter between pair	`','`

Now that the Object of values has been created, you can use the `unnest` transform to unpack this mapped data. In the following, each key is specified, which results in separate columns headed by the named key:

NOTE: Each key must be entered on a separate line in the Path to elements area.

Transformation Name	`Unnest Objects into columns`
Parameter: Column	`extractkv_Properties`
Parameter: Paths to elements	`year`
Parameter: Paths to elements	`make`
Parameter: Paths to elements	`model`
Parameter: Paths to elements	`color`
Parameter: Paths to elements	`mileage`
Parameter: Paths to elements	`cost`

Results:

When you delete the unnecessary Properties columns, the dataset now looks like the following:

VIN	year	make	model	color	mileage	cost
XX3 JT4522	2004	Subaru	Impreza	green	125422	3199
HT4 UJ9122	2006	VW	Passat	silver	102941	4599
KC2 WZ9231	2009	GMC	Yukon	black	68213	12899
LL8 UH4921	2011	BMW	328i	brown	57212	16999

Working with JSON v2

Contents:

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 task is described by way of example. In the example task, the JSON file must be imported into Designe r 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.

> **NOTE:** When this feature is enabled, all JSON imported datasets created under the legacy method must be recreated to behave like v2 datasets with respect to conversion and schema management. Features developed in the future may not retroactively be supported in the v1 legacy mode. You should convert to using the v2 method.

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.
- Each JSON record must be less than 20 MB in size.

NOTE: This maximum record length can be modified. For more information, see *Configure Application Limits*.

- 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.
- 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	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. **Tip:** This version is supported through 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.	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/w285fpp11.jpg", "filename": "w285fpp11.jpg"}
{"metrics": [{"rank": "1042", "score" : "9681"}], "caption": "This sweet puppy has transformed our life!",
"id": "9x2774", "url": "https://www.example.com/fmll0cy11.jpg", "filename": "fmll0cy11.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.

Task

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.
4. In Flow View, add the JSON-based imported dataset to your flow and create a recipe for it.
   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":"	This sweet puppy has transformed our	9x2774	https://www.example.com/fmll0cy11.jpg	fmll0cy11.jpg

9681"}]	life!			
[{"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/fmll0cy11.jpg","filename":"fmll0cy11.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 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.

> **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/fmll0cy11.jpg","filename":"fmll0cy11.jpg","resources":[{"url":"https://www.example.com
/fmll0cy11.jpg","filename":"fmll0cy11.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/wbbufmll0cy11.jpg",
      "short": "wbbufmll0cy11.jpg"
    },
    {
      "score": 16778,
      "title": "F/28/5\u20197\u201d [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 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`

4. 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.
5. 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
9kt8ex	19669	bzygw285fpp11.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
9x2774	19171	wbbufmll0cy11.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
a8guou	16778	3t0kmljnmq521.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
atla3n	16743	9t6tvsjs16i21.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
9kt8ex	19669	bzygw285fpp11.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
	19171			19171		18090.25

9x2774		wbbufmll0cy11.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.		https://i.redd.it/wbbufmll0cy11.jpg	
a8guou	16778	3t0kmljnmq521.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
atla3n	16743	9t6tvsjs16i21.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:

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 method is described by way of example. In the example, 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. Features developed in the future, such as schema management, are unlikely to support this method of JSON import.

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 flows, they are likely to require modifications that you have done in recipes.

Basic process:

This migration process 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 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 Ingestion Process

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.
4. In Flow View, add the JSON-based imported dataset to your flow and create a recipe for it.
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.
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. For more information, see *Object Data Type*.

Steps:

The following steps convert your JSON to an 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	`/\n\s*"/`
Parameter: Replace with	`\"`
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.
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	`characteristics.cover_color`
Parameter: column2	`characteristics.paper_stock`
Parameter: column3	`characteristics.paper_source`
Parameter: Nest columns to	`Object`
Parameter: New column name	`characteristics`

3. In the generated `characteristics` column, you can remove the `characteristics.` from the key value:

Transformation Name	`Replace text or patterns`
Parameter: Column	`characteristics`
Parameter: Find	``` `characteristics.` ```
Parameter: Replace with	`(empty)`

4. Now, delete the three source columns:

Transformation Name	`Delete columns`
Parameter: column1	`characteristics.cover_color`
Parameter: column2	`characteristics.paper_stock`
Parameter: column3	`characteristics.paper_source`

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:

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`

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	``FIND(Name,`{start} `,false,0)``
Parameter: New name	`FirstName`

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	`LEFT(city,3)`

   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	`LEFT(city,3)`

   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	`'A'`

   New column name: `column1`

Transformation Name	`New formula`
Parameter: Formula	`Single row formula`

type	
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.

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. From the menubar, click **Library**.
2. In the Library page, click **Import Data**.
3. In the Import Data page, 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 *File Import Settings*.

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:

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.

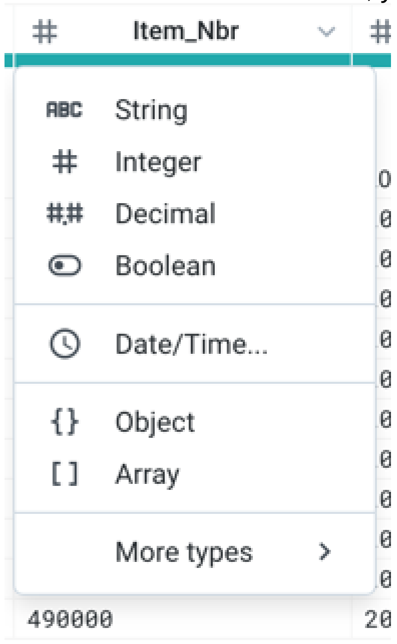


Figure: Column Data Type Menu

2. To the right of the column name, you can click the caret to open the column menu. Select **Change Type** an d make a selection from the sub-menu.

Tip: Both of the above methods become individual steps in your recipe.

Change through Transform Builder

You can change data type for a single column or multiple columns through the Transform Builder. 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: You can lock the data type for columns to prevent it from being updated when the data is transformed in subsequent steps.

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*.

Lock Data Type

You can lock a column's data type through the Transform Builder. When a column's data type is locked, the data type is no longer automatically checked and updated by the Designer Cloud application .

Tip: If you do not wish to have the data types modified, you can add a transformation to lock all of them in a single step. Details are below.

Via Transform Builder

1. In the Search panel, enter `lock column type`.
2. From the **Columns** drop-down, select any one of the following options:
    a. **Multiple**: Select one or more columns from the drop-down list.
    b. **Range**: Specify a start column and ending column. All columns inclusive are selected.
    c. **All**: Select all columns in the dataset.

        NOTE: This option locks all the column's data type.

    d. **Advanced**: Specify the columns using a comma-separated list. You can combine multiple and range options under Advanced. Example:

    ```
    c1,c3,c5~c8
    ```
3. Specify the other parameters.
4. To add the step to your recipe, click **Add**.

Example - lock a column's data type

This transformation locks the column data type:

Transformation Name	`lock column to current type`
Parameter: Columns	`Multiple`
Parameter: Column 1	`Store_Nbr, Whse_Name`

Example - lock the data types for all columns

This transformation locks the data types for all columns:

Tip: Many transformations support the Advanced option for column selection. You can specify column ranges, including all columns using the asterisk (`*`) wildcard. See the following.

Transformation Name	`lock column to current type`
Parameter: Columns	`Advanced`
Parameter: Column 1	`*`

Unlock Data Type

You can unlock a column's data type by following any one of these methods:

Via Transform Builder

In the Transformer Builder, you can select unlock to the current type option to apply the unlock feature to one or more columns.

This transformation unlocks the column data type:

Transformation Name	`unlock column to current type`
Parameter: Columns	`Multiple`
Parameter: Column 1	`Store_Nbr, Whse_Name`

Via column menus

You can unlock the data type for individual columns through the following column menus:

- To the left of the column name, you can click the icon and select **Automatically update**. The selected column is unlocked.

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`

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. **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.
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). **NOTE:** When values are pasted into the column, the column data type may be re-inferred.

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.

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.

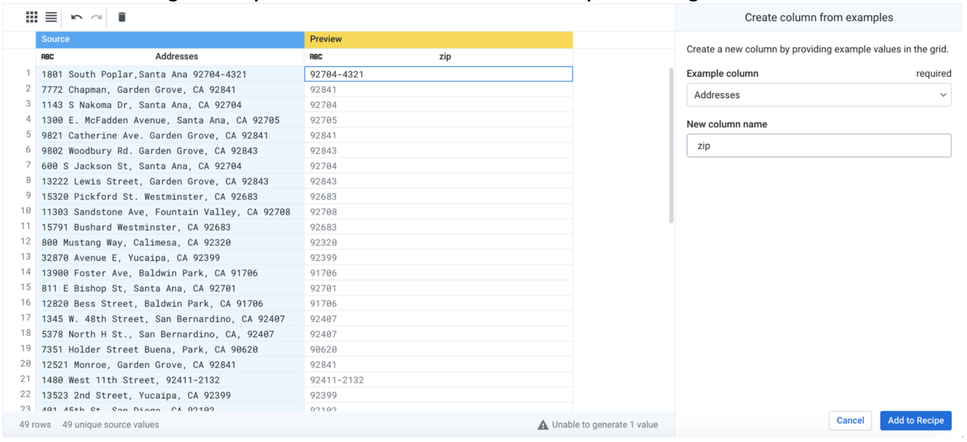
  > **NOTE:** Multi-value inputs, such as Object or Array data types, must be converted to String data type prior to transformation by example.

- 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:

1. In the Transformer page, locate the column to use as your source. From the column menu, select **Create column from examples**.
2. In the Transform Builder, enter the new column name.
3. In the following example, a new column called `zip` is being created from the `Addresses` column:


	Source	Preview
	Addresses	zip
1	1801 South Poplar,Santa Ana 92704-4321	92704-4321
2	7772 Chapman, Garden Grove, CA 92841	92841
3	1143 S Nakoma Dr, Santa Ana, CA 92704	92704
4	1300 E. McFadden Avenue, Santa Ana, CA 92705	92705
5	9821 Catherine Ave. Garden Grove, CA 92841	92841
6	9802 Woodbury Rd. Garden Grove, CA 92843	92843
7	600 S Jackson St, Santa Ana, CA 92704	92704
8	13222 Lewis Street, Garden Grove, CA 92843	92843
9	15320 Pickford St. Westminster, CA 92683	92683
10	11303 Sandstone Ave, Fountain Valley, CA 92708	92708
11	15791 Bushard Westminster, CA 92683	92683
12	800 Mustang Way, Calimesa, CA 92320	92320
13	32870 Avenue E, Yucaipa, CA 92399	92399
14	13900 Foster Ave, Baldwin Park, CA 91706	91706
15	811 E Bishop St, Santa Ana, CA 92701	92701
16	12820 Bess Street, Baldwin Park, CA 91706	91706
17	1345 W. 48th Street, San Bernardino, CA 92407	92407
18	5378 North H St., San Bernardino, CA, 92407	92407
19	7351 Holder Street Buena, Park, CA 90620	90620
20	12521 Monroe, Garden Grove, CA 92841	92841
21	1480 West 11th Street, 92411-2132	92411-2132
22	13523 2nd Street, Yucaipa, CA 92399	92399

49 rows 49 unique source values

Unable to generate 1 value

Create column from examples

Create a new column by providing example values in the grid.

Example column: Addresses (required)

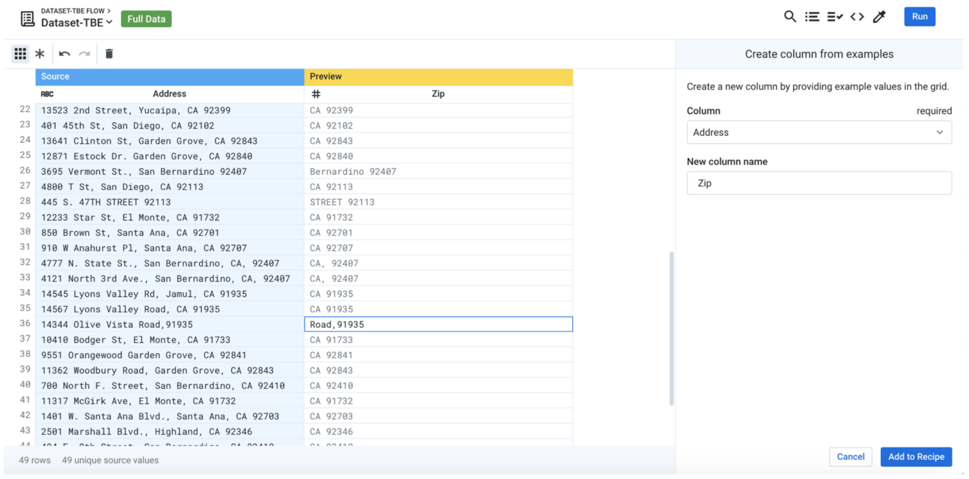
New column name: zip

Figure: Selected column and first value is specified

4. 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.

5. 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`:



DATASET-TBE FLOW > Dataset-TBE Full Data

Create column from examples

Create a new column by providing example values in the grid.

Column: Address (required)

New column name: Zip

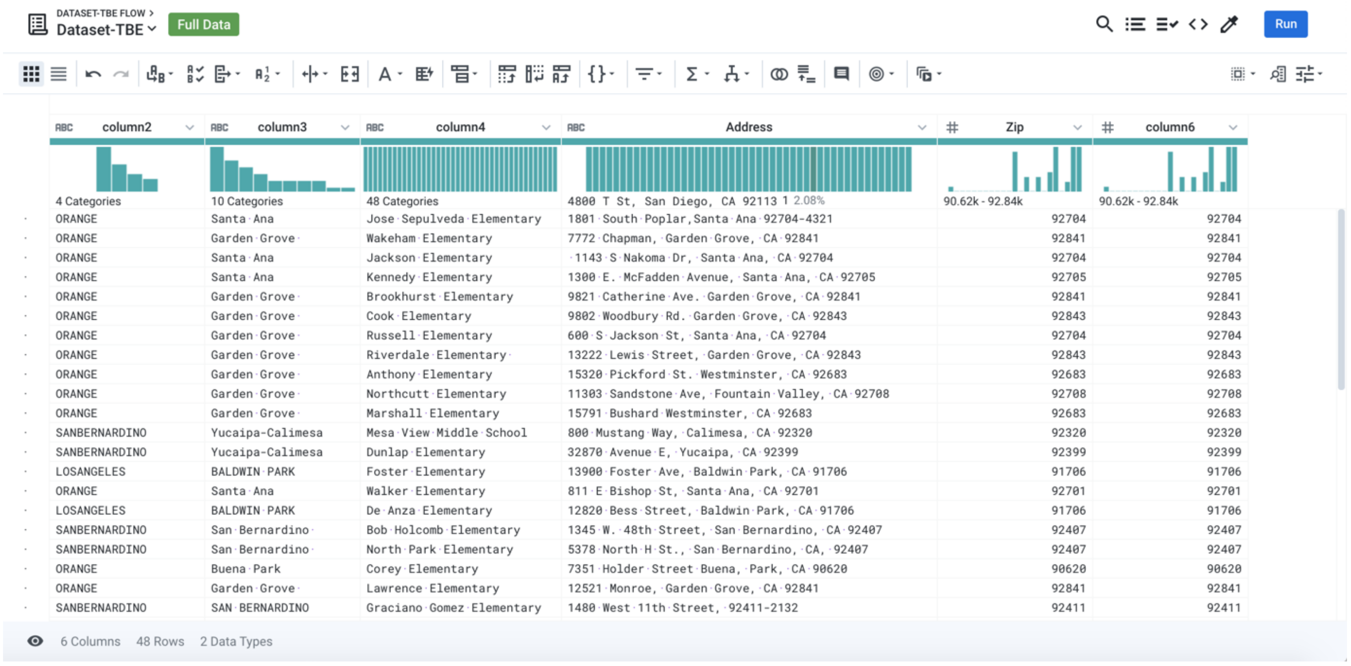
	Source: Address	Preview: Zip
22	13523 2nd Street, Yucaipa, CA 92399	CA 92399
23	401 45th St, San Diego, CA 92102	CA 92102
24	13641 Clinton St, Garden Grove, CA 92843	CA 92843
25	12871 Estock Dr. Garden Grove, CA 92840	CA 92840
26	3695 Vermont St., San Bernardino 92407	Bernardino 92407
27	4800 T St, San Diego, CA 92113	CA 92113
28	445 S. 47TH STREET 92113	STREET 92113
29	12233 Star St, El Monte, CA 91732	CA 91732
30	850 Brown St, Santa Ana, CA 92701	CA 92701
31	910 W Anahurst Pl, Santa Ana, CA 92707	CA 92707
32	4777 N. State St., San Bernardino, CA, 92407	CA, 92407
33	4121 North 3rd Ave., San Bernardino, CA, 92407	CA, 92407
34	14545 Lyons Valley Rd, Jamul, CA 91935	CA 91935
35	14567 Lyons Valley Road, CA 91935	CA 91935
36	14344 Olive Vista Road,91935	Road,91935
37	10410 Bodger St, El Monte, CA 91733	CA 91733
38	9551 Orangewood Garden Grove, CA 92841	CA 92841
39	11362 Woodbury Road, Garden Grove, CA 92843	CA 92843
40	700 North F. Street, San Bernardino, CA 92410	CA 92410
41	11317 McGirk Ave, El Monte, CA 91732	CA 91732
42	1401 W. Santa Ana Blvd., Santa Ana, CA 92703	CA 92703
43	2501 Marshall Blvd., Highland, CA 92346	CA 92346

49 rows 49 unique source values

Cancel Add to Recipe

Figure: Populating multiple example rows improves the overall quality of transformation across all rows

6. 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.
7. Click **Add to Recipe**.
8. The new `Zip` column is added to the dataset.



DATASET-TBE FLOW > Dataset-TBE Full Data

column2	column3	column4	Address	Zip	column6
4 Categories	10 Categories	48 Categories	4800 T St, San Diego, CA 92113 1 2.08%	90.62k - 92.84k	90.62k - 92.84k
ORANGE	Santa Ana	Jose Sepulveda Elementary	1801 South Poplar,Santa Ana 92704-4321	92704	92704
ORANGE	Garden Grove	Wakeham Elementary	7772 Chapman, Garden Grove, CA 92841	92841	92841
ORANGE	Santa Ana	Jackson Elementary	1143 S Nakoma Dr, Santa Ana, CA 92704	92704	92704
ORANGE	Santa Ana	Kennedy Elementary	1300 E. McFadden Avenue, Santa Ana, CA 92705	92705	92705
ORANGE	Garden Grove	Brookhurst Elementary	9821 Catherine Ave. Garden Grove, CA 92841	92841	92841
ORANGE	Garden Grove	Cook Elementary	9802 Woodbury Rd. Garden Grove, CA 92843	92843	92843
ORANGE	Garden Grove	Russell Elementary	600 S Jackson St, Santa Ana, CA 92704	92704	92704
ORANGE	Garden Grove	Riverdale Elementary	13222 Lewis Street, Garden Grove, CA 92843	92843	92843
ORANGE	Garden Grove	Anthony Elementary	15320 Pickford St. Westminster, CA 92683	92683	92683
ORANGE	Garden Grove	Northcutt Elementary	11303 Sandstone Ave, Fountain Valley, CA 92708	92708	92708
ORANGE	Garden Grove	Marshall Elementary	15791 Bushard Westminster, CA 92683	92683	92683
SANBERNARDINO	Yucaipa-Calimesa	Mesa View Middle School	800 Mustang Way, Calimesa, CA 92320	92320	92320
SANBERNARDINO	Yucaipa-Calimesa	Dunlap Elementary	32870 Avenue E, Yucaipa, CA 92399	92399	92399
LOSANGELES	BALDWIN PARK	Foster Elementary	13900 Foster Ave, Baldwin Park, CA 91706	91706	91706
ORANGE	Santa Ana	Walker Elementary	811 E Bishop St, Santa Ana, CA 92701	92701	92701
LOSANGELES	BALDWIN PARK	De Anza Elementary	12820 Bess Street, Baldwin Park, CA 91706	91706	91706
SANBERNARDINO	San Bernardino	Bob Holcomb Elementary	1345 W. 48th Street, San Bernardino, CA 92407	92407	92407
SANBERNARDINO	San Bernardino	North Park Elementary	5378 North H St., San Bernardino, CA, 92407	92407	92407
ORANGE	Buena Park	Corey Elementary	7351 Holder Street Buena, Park, CA 90620	90620	90620
ORANGE	Garden Grove	Lawrence Elementary	12521 Monroe, Garden Grove, CA 92841	92841	92841
SANBERNARDINO	SAN BERNARDINO	Graciano Gomez Elementary	1480 West 11th Street, 92411-2132	92411	92411

6 Columns 48 Rows 2 Data Types

Figure: Transformed example column

Remove Data

Contents:

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`

Delete rows

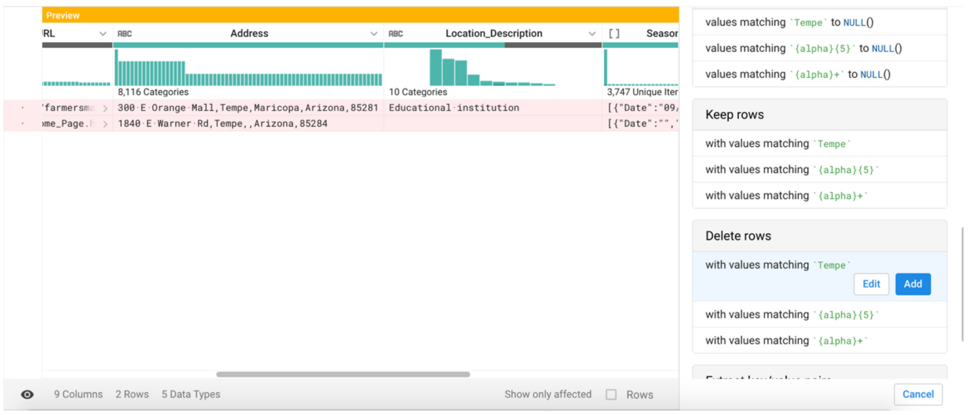
You can delete rows in your dataset based on conditional patterns that you specify. The easiest method is to select a string in the appropriate column and then choose the Delete suggestion card.

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.



```
'farmersma  300 E Orange Mall,Tempe,Maricopa,Arizona,85281  Educational institution  [{"Date":"09,
ome_Page.h  1840 E Warner Rd,Tempe,,Arizona,85284                                    [{"Date":"",'
```

values matching `Tempe` to NULL()
values matching `{alpha}{5}` to NULL()
values matching `{alpha}+` to NULL()

Keep rows
with values matching `Tempe`
with values matching `{alpha}{5}`
with values matching `{alpha}+`

Delete rows
with values matching `Tempe`
with values matching `{alpha}{5}`
with values matching `{alpha}+`

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	`MERGE([column1~columnN])`
Parameter: New column name	`'all_blank_vals'`

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.

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:

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)`

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:

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'`

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 **E dit**.
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:

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 on suggestions, see *Overview of Predictive Transformation*.

Replace using Transform Builder

When a pattern suggestion is selected, it is specified in the Transform Builder for review and addition to your recipe. 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.
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	`&apos;My piece of text&apos;`
Trifacta pattern	A Trifacta pattern represents zero or more characters that match a pattern. In Designe r Cloud Powered by Trifacta Enterprise Edition, Patterns are a simplified means of expressing regular expressions. For more information on Trifacta pattern syntax, see *Text Matching* . **Tip:** The examples in this section use Trifacta Patterns, which are simpler to use than regular expressions.	back-ticks	`` `{start}{digit}{3}` ``
Regula r expres sion	Regular expressions are a standard-based method of describing patterns in values. **NOTE:** Regular expressions are considered a developer-level skill. For more information on regular expression, see on *RE2* and *PCRE* regular expressions.	forward slashes	`/^.{<span class="hljs-number">0</span>,<span class="hljs-number">3</span>}/`

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:

Transformation Name	`Replace text or patterns`
Parameter: Column	`CustomerID`
Parameter: Find	`` `{start}%{3}` ``
Parameter: Replace with	`CustID-`

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*.

> **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 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	``` `{alpha-numeric}` ```
Parameter: Replace with	`##`
Parameter: Start search after	``` `{start}{digit}{2}` ```
Parameter: Stop search before	``` `{any}` ```

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
``` `{at-username}` ```	Matches values that begin with an at-sign, such as `@trifacta`. This Trifacta Pattern can be useful if you need to remap or mask username values.

`` `{hashtag}` ``	Matches values that begin with a hashtag, such as `#dataprep`. For an example of this Trifacta Pattern, see *Extract Values*.
`` `{hex}` ``	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: `AE00`, `1F2F`, `100`.
`` `{phone}` ``	Matches valid phone numbers within a set of values. For more information on this data type pattern, see *Phone Number Data Type*.
`` `{email}` ``	Matches valid email addresses within a set of values. For more information on this data type pattern, see *Email Address Data Type*.
`` `{url}` ``	Matches valid URL addresses within a set of values. For more information, on this data type pattern, see *URL Data Type*.

Replace Groups of Values

Contents:

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. **Tip:** You can replace specific values in a column with a preferred value. For more information, see *Replace Cell Values*.
By column menu	From the column menu, select **Replace** and a sub-menu item to begin configuring a replacement transformation.
By Transformer toolbar	At the top of the data grid, click the Replace icon in the Transformer toolbar to begin configuring replacements.
By Search panel	In the Search panel, enter `replace` to build a replacement transformation from scratch.

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.

Transformation Name	`Edit column with formula`
Parameter: Columns	`All`
Parameter: Formula	`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:

Transformation Name	`Edit with formula`
Parameter: Columns	`colB`
Parameter: Formula	`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:

Transformation Name	`Edit with formula`
Parameter: Columns	`ProdId`
Parameter: Formula	`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-`.

Transformation Name	`Replace by position`
Parameter: Column	`Whse_name`
Parameter: Start position	`0`
Parameter: End position	`0`
Parameter: Replace with	`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:

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 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`:

Transformation Name	`Edit column with formula`
Parameter: Columns	`Qty,DiscountPct`
Parameter: Formula	`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:

Transformation Name	`New formula`
Parameter: Formula type	`Single row formula`
Parameter: Formula	`if(ismissing([DiscountPct]), average(DiscountPct), DiscountPct)`
Parameter: New column name	`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.

Transformation Name	`Replace mismatched values`
Parameter: Columns	`Multiple`
Parameter: Column 1	`myDate`
Parameter: Data type to evaluate	`Date/Time`
Parameter: Date/Time type	`yyyy*mm*dd`
Parameter: Replace with	`Custom value`
Parameter: New value	`'##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.

Normalize Numeric Values

Contents:

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:

Transformation Name	`New formula`
Parameter: Formula type	`Single row formula`
Parameter: Formula	`(-636074.22 + -2465086.34)`
Parameter: New column name	`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:

Transformation Name	`Edit column with formula`
Parameter: Columns	`MySum`

Parameter: Formula	`ROUND($col,3)`

For more information on floating point computations, see *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))`

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 **Modi fy**.
4. The transformation should look like the following:

Transformation Name	`Replace text or patterns`
Parameter: Column	`Weight_kg`
Parameter: Find	`' kg'`
Parameter: Replace with	`''`
Parameter: Match all occurrences	`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:

Transformation Name	`Edit column with formula`
Parameter: Columns	`Weight_kg`
Parameter: Formula	`(Weight_kg * 2.2)`

If you want to round the value to the nearest integer, use the following:

Transformation Name	`Edit column with formula`
Parameter: Columns	`Weight_kg`
Parameter: Formula	`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:

Transformation Name	`Replace text or patterns`
Parameter: Column	`Height_ft`
Parameter: Find	`` `"` ``
Parameter: Replace with	`''`
Parameter: Match all occurrences	`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:

Transformation Name	`New formula`
Parameter: Formula type	`Single row formula`
Parameter: Formula	`((Height_ft1 *12)+Height_ft2)`
Parameter: New column name	`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.

Transformation Name	`Edit column with formula`
Parameter: Columns	`Distance`
Parameter: Formula	`IF((Distance < 1000,(Distance * 1000), Distance)`

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:

Transformation Name	`Edit column with formula`
Parameter: Columns	`Width_cm`
Parameter: Formula	`NUMFORMAT(Width_cm '#.#')`

Transformation Name	`Edit column with formula`
Parameter: Columns	`Height_cm`
Parameter: Formula	`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:

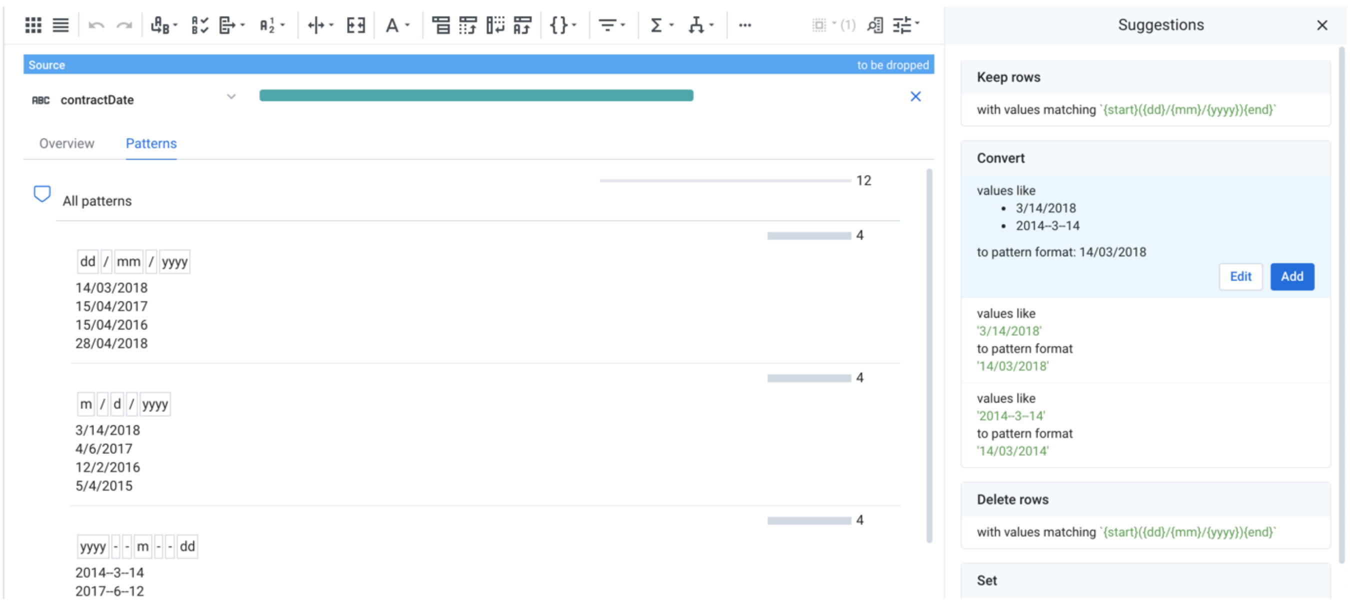
This section describes techniques to standardize values in your datasets using patterns. From the Column Details panel in the Designer Cloud® application , you can review and select patterns in the column's data. These selections can be used as the basis for converting all applicable values to the selected 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.



All patterns 12

dd / mm / yyyy 4
14/03/2018
15/04/2017
15/04/2016
28/04/2018

m / d / yyyy 4
3/14/2018
4/6/2017
12/2/2016
5/4/2015

yyyy - - m - - dd 4
2014--3--14
2017--6--12

Suggestions

Keep rows
with values matching `{start}({dd}/{mm}/{yyyy}){end}`

Convert
values like
- 3/14/2018
- 2014--3--14

to pattern format: 14/03/2018

Edit Add

values like
'3/14/2018'
to pattern format
'14/03/2018'

values like
'2014--3--14'
to pattern format
'14/03/2014'

Delete rows
with values matching `{start}({dd}/{mm}/{yyyy}){end}`

Set

Figure: Selecting Datetime patterns in the Patterns tab

In the above, the pattern block prompts suggestions for Convert tasks based on the selected patterns.

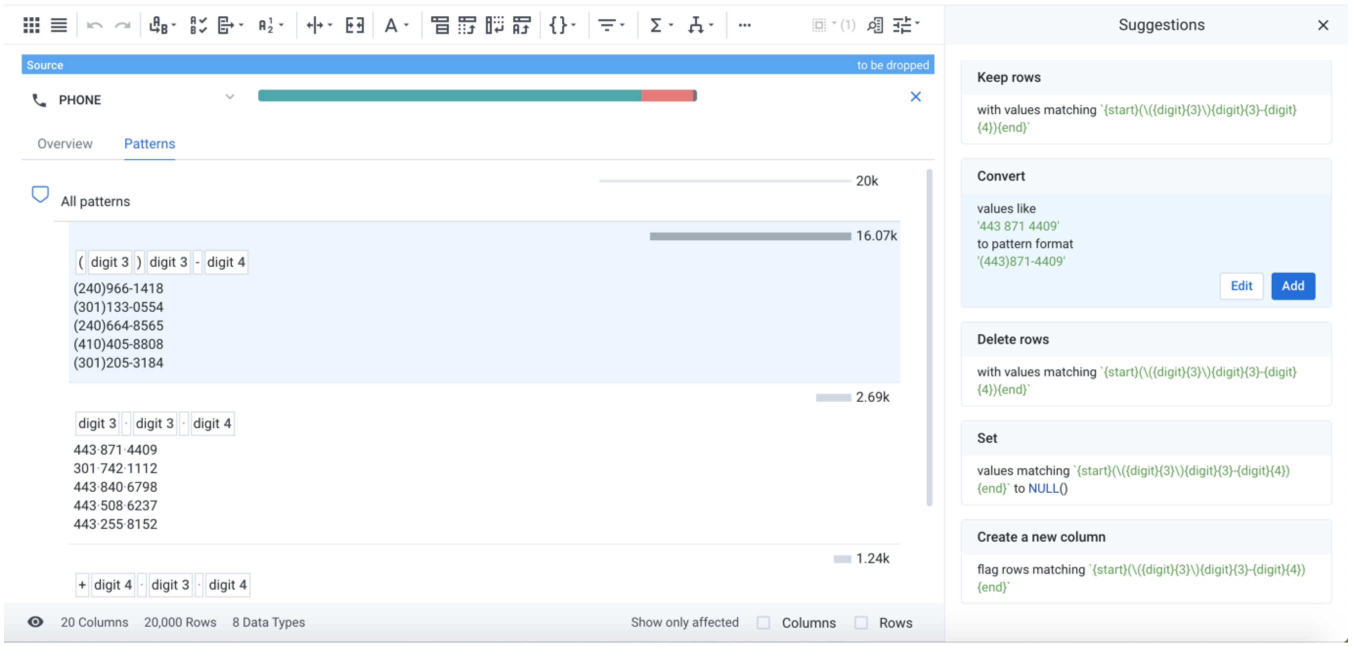
- Click **Edit** to modify the task.
- Click **Add** to add the task as a step to your recipe.

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}{3})`
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:



(240)966-1418
(301)133-0554
(240)664-8565
(410)405-8808
(301)205-3184

443·871·4409
301·742·1112
443·840·6798
443·508·6237
443·255·8152

Keep rows
with values matching `{start}(\({digit}{3}\){digit}{3}-{digit}{4}){end}`

Convert
values like
'443 871 4409'
to pattern format
'(443)871-4409'

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()

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 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`	• `Jan 01` • `1 01`	ok (2 mappings)
`Jan 1, 1981`	`01/01/81`	• `Jan 01` • `1 01` • `1981 81`	Not suggested (3 mappings)

Tip: Use the DATEFORMAT function to convert Datetime values to different date formats.

For more information on supported formats, see *Datetime Data Type*.

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:

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:

Transformation Name	`Change column type`
Parameter: Columns	`col1,col2, col3`
Parameter: New type	`'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:

Transformation Name	`Edit column with formula`
Parameter: Columns	`myCol`

Parameter: Formula	`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*.

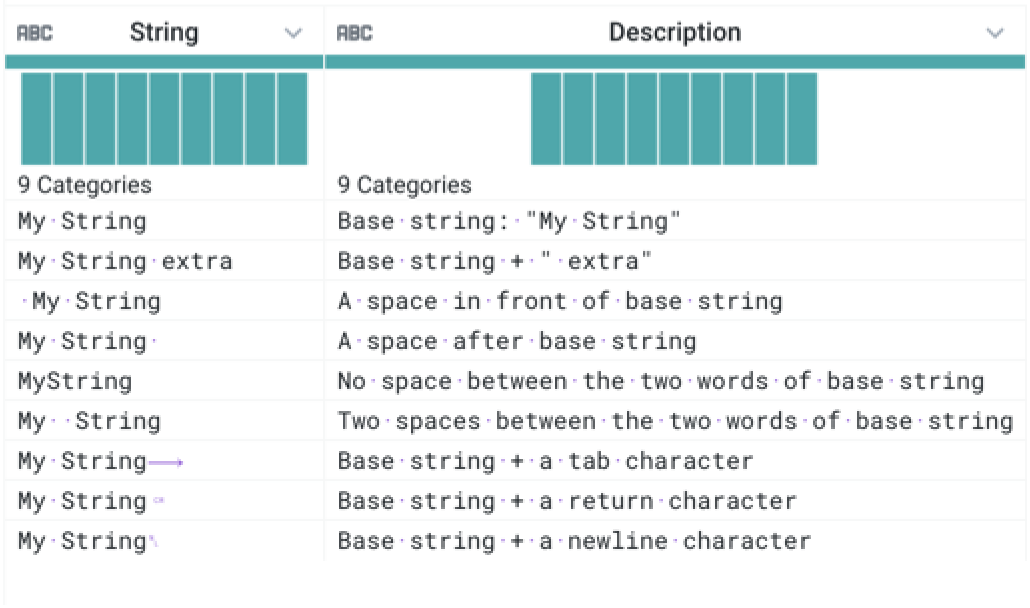
Example - Clean up Strings

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:



String	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	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.

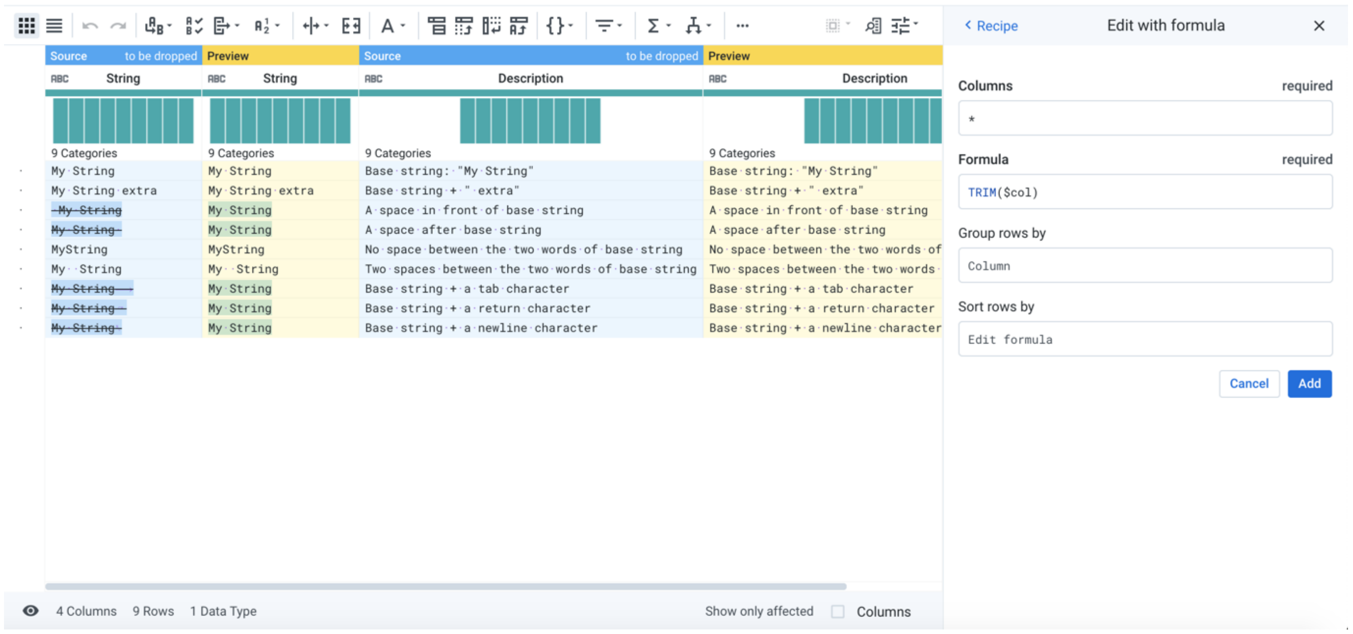
When transforming strings, a key step is to trim off the whitespace at the beginning and ending of the string. For the above dataset, you can use the following command to remove these whitespaces:

Transformation Name	`Edit column with formula`
Parameter: Columns	`All`
Parameter: Formula	`TRIM($col)`

The above transform uses the following special values, which are available for some transforms like `set`:

Special Value	Description
`*`	For the Columns textbox under Advanced, you can use this wildcard to reference all columns in the dataset. **Tip:** You can also select `All` from the Columns drop-down.
`$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	Source · to be dropped	Preview
String	String	Description	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
My··String	My··String	Two·spaces·between·the·two·words·of·base·string	Two·spaces·between·the·two·words·
~~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

4 Columns 9 Rows 1 Data Type Show only affected Columns

Edit with formula

Columns (required)	*
Formula (required)	TRIM($col)
Group rows by	Column
Sort rows by	Edit formula

Figure: Trim data to improve matches

Tip: To remove all whitespace, including spaces in between, you can use the `REMOVEWHITESPACE` functi on.

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:

	`Edit column with formula`

Transformation Name	
Parameter: Columns	`Zip`
Parameter: Formula	`IF(ISMISMATCHED(Zip, ['Zipcode']),'00000',$col)`

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	`' extra'`
Parameter: Replace with	`''`
Parameter: Match all occurrences	`true`

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	`'  '`
Parameter: Replace with	`''`
Parameter: Match all occurrences	`true`

- 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:

Transformation Name	Replace text or patterns
Parameter: Column	String
Parameter: Find	`({alpha})({upper})`
Parameter: Replace with	'$1 $2'
Parameter: Match all occurrences	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:

Transformation Name	Replace text or patterns
Parameter: Column	String
Parameter: Find	` {alpha}+{end}`
Parameter: Replace with	''
Parameter: Match all occurrences	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:

Transformation Name	Replace text or patterns
Parameter: Column	String
Parameter: Find	` {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`
Parameter: Formula	`TRIM(myCol)`

To apply this function across all columns in the dataset, you can use the following:

Transformation Name	`Edit column with formula`
Parameter: Columns	`All`
Parameter: Formula	`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:

Transformation Name	`Edit column with formula`
Parameter: Columns	`All`
Parameter: Formula	`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.

Transformation Name	`Remove whitespace`
Parameter: Columns	`name`
Parameter: Format	`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 through the Import Data page.

Transformation Name	`Remove symbols`
Parameter: Columns	`All`

Parameter: Format	`Remove symbols`

Remove accents

The following transformation converts all accented characters (e.g."ä") to unaccented characters (e.g "a").

Transformation Name	`Remove accents from text`
Parameter: Columns	`All`
Parameter: Format	`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.

Transformation Name	`Trim quotes`
Parameter: Columns	`All`
Parameter: Format	`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:

Transformation Name	`Add suffix to text`
Parameter: Columns	`txtZipCode`
Parameter: Format	`Add suffix`
Parameter: Text to add	`'-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:

Transformation Name	`Edit column with formula`
Parameter: Columns	`myStrings`
Parameter: Formula	`LOWER(myStrings)`

Upper case:

Transformation Name	`Edit column with formula`
Parameter: Columns	`myStrings`
Parameter: Formula	`UPPER(myStrings)`

Proper (sentence) case:

Transformation Name	`Edit column with formula`
Parameter: Columns	`myStrings`
Parameter: Formula	`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 that the prescribed maximum are unchanged. You can use the `LEFT` or `RIGHT` functions to change the size of the oversized ones. See below.

Transformation Name	`Pad text with leading characters`
Parameter: Columns	`MyStrings`
Parameter: Format	`Pad with leading characters`
Parameter: Character to pad with	`' '`
Parameter: Length	`50`

Fixed length strings

You can limit the maximum size of a column or set of columns to a fixed string length. For example:

Transformation Name	`Edit column with formula`
Parameter: Columns	`col1,col2`
Parameter: Formula	`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**
LEN Function	Returns the number of characters in a specified string. String value can be a column reference or string literal.
FIND Function	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.
RIGHTFIND Function	Returns the index value in the input string where the last instance of a matching string is located. Search is conducted right-to-left.
LEFT Function	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.
RIGHT Function	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.
SUBSTRING Function	Matches some or all of a string, based on the user-defined starting and ending index values within the string.
SUBSTITUTE Function	Replaces found string literal or pattern or column with a string, column, or function returning strings.

Reset Types

After modifying non-text values as strings, remember to convert them back to their original types.

Manage String Lengths

Contents:

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:

Transformation Name	`Edit column with formula`
Parameter: Columns	`String_test`
Parameter: Formula	`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:

Transformation Name	`Edit column with formula`
Parameter: Columns	`*`
Parameter: Formula	`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:

Transformation Name	`Extract text or pattern`
Parameter: Column to extract from	`zipcode`
Parameter: Option	`Custom text or pattern`
Parameter: Text to extract	`` `{zip}` ``
Parameter: Start extracting after	`` `{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:

Transformation Name	`Edit column with formula`
Parameter: Columns	`prodID`
Parameter: Formula	`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`.

Transformation Name	`Edit column with formula`
Parameter: Columns	`countries`

Parameter: Formula	`SUBSTRING(countries, 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:

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 **Extract** and a sub-menu item to begin configuring a transformation.
By Transformer toolbar	At the top of the data grid, click the Extract icon in the Transformer toolbar to begin configuring extract transformations.

By Search panel	In the Search panel, enter `extract` to build a transformation from scratch.

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`:

Transformation Name	`Extract text or pattern`
Parameter: Column to extract from	`msg_LinkedIn`
Parameter: Option	`Custom text or pattern`
Parameter: Text to extract	`'#bigdata'`
Parameter: Number of matches to extract	`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:

Transformation Name	`Extract text or pattern`
Parameter: Column to extract from	`phone_num`
Parameter: Option	`Custom text or pattern`
Parameter: Text to extract	``` `{digit}` ```
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	`msg_LinkedIn`
Parameter: Option	`Custom text or pattern`
Parameter: Text to extract	`` `\#{alphanum-underscore}+` ``
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.
`\#`	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.
`{alphanum-underscore}`	This Trifacta pattern element is used to indicate a single alphanumeric or underscore character.
`+`	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	Set this value to the total number of patterns you wish to extract. **NOTE:** 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. The default is `1`.
Ignore case	By default, pattern matching is case-sensitive. Select this checkbox to ignore case when matching.
Ignore matches between	You can enter a pattern here to describe any patterns that should not be part of any match. This option is useful if you 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` pr in `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 from	`ProductName`
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	`YEAR(myDate)`
Parameter: New column name	`myYear`

The following functions can be used to extract values from a Datetime column, as long as the values are present in the formatted date:

Function Name	Description
DAY Function	Derives the numeric day value from a Datetime value. Source value can be a a reference to a column containing Datetime values or a literal.
MONTH Function	Derives the month integer value from a Datetime value. Source value can be a a reference to a column containing Datetime values or a literal.
YEAR Function	Derives the four-digit year value from a Datetime value. Source value can be a a reference to a column containing Datetime values or a literal.
HOUR Function	Derives the hour value from a Datetime value. Generated hours are expressed according to the 24-hour clock.
MINUTE Function	Derives the minutes value from a Datetime value. Minutes are expressed as integers from 0 to 59.
SECOND Function	Derives the seconds value from a Datetime value. Source value can be a a reference to a column containing Datetime values or a literal.

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	`myDate`
Parameter: Formula	`DATEFORMAT(myDate, "yy")`

Extract numeric values

You can extract numerical data from text values. In the following example, the first number is extracted from the a `ddress` column, which would correspond to extracting the street number for the address:

Transformation Name	`Extract patterns`

Parameter: Column to extract from	`address`
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:

Transformation Name	`New formula`
Parameter: Formula type	`Single row formula`
Parameter: Formula	`DOMAIN(myURL)`
Parameter: New column name	`myDomain`

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:

Function Name	Description
HOST Function	Finds the host value from a valid URL. Input values must be of URL or String type and can be literals or column references.
DOMAIN Function	Finds the value for the domain from a valid URL. Input values must be of URL or String type.
SUBDOMAIN Function	Finds the value a subdomain value from a valid URL. Input values must be of URL or String type.
SUFFIX Function	Finds the suffix value after the domain from a valid URL. Input values must be of URL or String type.
URLPARAMS Function	Extracts the query parameters of a URL into an Object. The Object keys are the parameter's names, and its values are the parameter's values. Input values must be of URL or String type.

Query parameters

You can extract query parameter values from an URL. The following example extracts the `store_id` value from the `storeURL` field value:

Transformation Name	`Extract patterns`
Parameter: Column to extract from	`storeURL`

Parameter: Option	`HTTP Query strings`
Parameter: Fields to extract	`store_id`

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:

Transformation Name	`Unnest Objects into columns`
Parameter: Column	`restFeatures`
Parameter: Paths to elements - 1	`TotalTableSeats`
Parameter: Paths to elements - 2	`CounterSeats`
Parameter: Include original column name	`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.

Transformation Name	`Expand arrays into rows`
Parameter: Column	`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:

Transformation Name	`Extract matches into Array`
Parameter: Column	`product`
Parameter: Pattern matching elements in the list	`` `{any}` ``
Parameter: Delimiter separating each element	`` `,` ``

Results:

Before	After
socks, socks, socks	["socks", "socks", "socks"]
pants, pants	["pants", "pants"]

Extract hashtags

In this example, you extract one or more values from a source column and assemble them in an Array column.

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 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

		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.

Transformation Name	`Extract matches into Array`
Parameter: Column	`customer_tweets`
Parameter: Pattern matching elements in the list	``` `{hashtag}` ```
Parameter: New column name	`Hashtag tweets`

Then, the source column can be deleted.

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:

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.

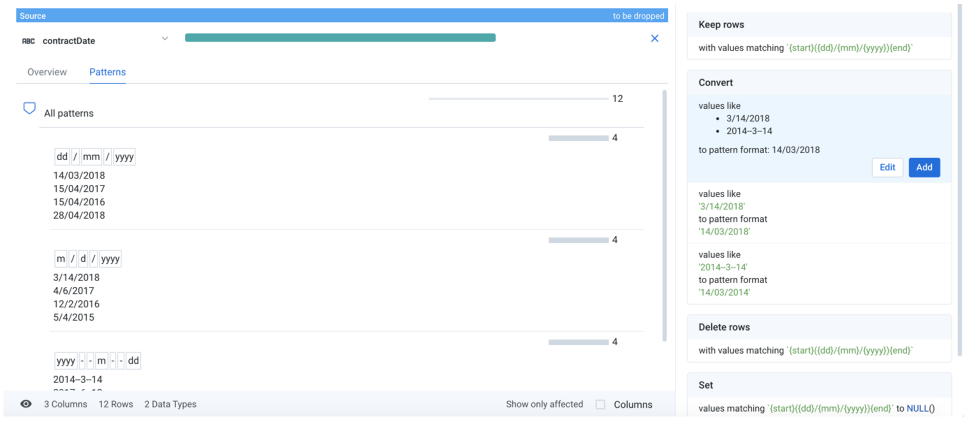
Option 1 - Patterns in the Column Details panel

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.



All patterns 12

dd / mm / yyyy 4
14/03/2018
15/04/2017
15/04/2016
28/04/2018

m / d / yyyy 4
3/14/2018
4/6/2017
12/2/2016
5/4/2015

yyyy - - m - - dd 4
2014--3--14

Keep rows
with values matching `{start}({dd}/{mm}/{yyyy}){end}`

Convert
values like
- 3/14/2018
- 2014--3--14

to pattern format: 14/03/2018

values like
'3/14/2018'
to pattern format
'14/03/2018'

values like
'2014--3--14'
to pattern format
'14/03/2014'

Delete rows
with values matching `{start}({dd}/{mm}/{yyyy}){end}`

Set
values matching `{start}({dd}/{mm}/{yyyy}){end}` to NULL()

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:

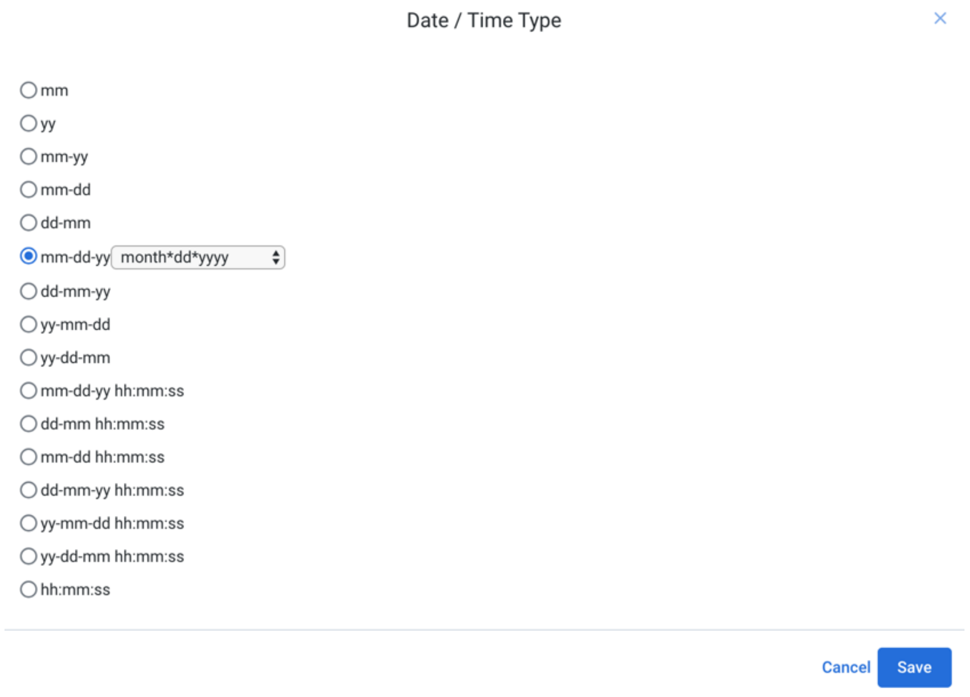
myDate	DATEFORMAT value
Mar-14-2018	MMM-dd-yyyy
03/14/2018	MM/dd/yyyy
2018-Mar-03	yyyy-MMM-dd
3/14/18	M/d/yy
03/14/2018 00:00:00	MM/dd/yyyy HH:mm:ss
March 14, 2018	MMMM dd, yyyy

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:



- ( ) 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

Figure: Date/Time format selector

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()` entr y with the `DATEFORMAT` function which changes these values to the proper format:

Transformation Name	`Edit with formula`
Parameter: Columns	`myDate`
Parameter: Formula	`ifvalid($col, ['Datetime','yy','yyyy'], dateformat($col, 'MMMM dd, yyyy'))`

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:

Transformation Name	`Replace text or patterns`
Parameter: Column	`UpdateTime`
Parameter: Find	`` `{start}{digit}{4}{end}` ``
Parameter: Replace with	`''`

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:

Transformation Name	`Replace text or patterns`
Parameter: Column	`UpdateTime`
Parameter: Find	`` `{start}({digit}{4}){end}` ``
Parameter: Replace with	`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:

Transformation Name	`Edit with formula`
Parameter: Columns	`lastDate`
Parameter: Formula	`DATEFORMAT(lastDate, 'yyyy:MM:dd')`

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.

Transformation Name	`Split by delimiter`
Parameter: Column	`contractDate`
Parameter: Option	`by Delimiter`

Parameter: Delimiter	`` `{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:

Transformation Name	`conditions`
Parameter: Condition type	`Case on single column`
Parameter: Column to evaluate	`Region`
Parameter: Case 1	`'EU'`
Parameter: Value 1	`contractDate3`
Parameter: Case 2	`'USA'`
Parameter: Value 2	`contractDate3`
Parameter: Case 3	`'CHN'`
Parameter: Value 1	`contractDate1`

3. For month:

Transformation Name	`conditions`
Parameter: Condition type	`Case on single column`
Parameter: Column to evaluate	`Region`
Parameter: Case 1	`'EU'`
Parameter: Value 1	`contractDate2`
Parameter: Case 2	`'USA'`
Parameter: Value 2	`contractDate1`
Parameter: Case 3	`'CHN'`
Parameter: Value 1	`contractDate2`

4. For day:

Transformation Name	`conditions`
Parameter: Condition type	`Case on single column`
Parameter: Column to evaluate	`Region`
Parameter: Case 1	`'EU'`
Parameter: Value 1	`contractDate1`
Parameter: Case 2	`'USA'`
Parameter: Value 2	`contractDate2`
Parameter: Case 3	`'CHN'`
Parameter: Value 1	`contractDate3`

5. You can now bring together these three columns:

Transformation Name	`Merge columns`
Parameter: Columns	`day, month, year`
Parameter: Separator	`'/'`
Parameter: New column name	`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:

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`
Parameter: Columns	`testCol`
Parameter: Formula	`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.

CASE function

You can chain together IF functions in the following manner:

Transformation Name	`Edit column with formula`
Parameter: Columns	`testCol`
Parameter: Formula	`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:

Transformation Name	`Edit column with formula`
Parameter: Columns	`testCol`
Parameter: Formula	`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`

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`:

Transformation Name	`Edit column with formula`
Parameter: Columns	`testCol`
Parameter: Formula	`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.

Prepare Data for Machine Processing

Contents:

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	`(ABS(POS_Sales - AVERAGE(POS_Sales))) / STDEV (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:

Transformation Name	`Filter rows`
Parameter: Condition	`Custom formula`
Parameter: Type of formula	`Custom single`
Parameter: Condition	`4 <= stdevs_POS_Sales`
Parameter: Action	`Delete matching rows`

Change outliers to mean values

You can also remove the effects of outliers be setting their value to the mean (average), which preserves the data in other columns in the row.

Transformation Name	`Edit with formula`
Parameter: Columns	`POS_Sales`
Parameter: Formula	`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.

Transformation Name	`Bin column`
Parameter: Column	`MilleBornes`
Parameter: Select Option	`Equal Sized Bins`
Parameter: Number of Bins	`10`
Parameter: New column name	`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 rename the columns to prepend the names with a reference to the source column.

Enrichment Tasks

Contents:

These topics cover various approaches to augmenting your data with fixed values, generated values, or data from other datasets.

Add New Columns

You can add new columns of data that you specify within the application. See *Create New Column*.

This new column can be created by combining multiple columns of existing data. See *Add Two Columns*.

Insert Metadata

You can insert data about your dataset into the dataset itself. See *Insert Metadata*.

Tip: A common use of available metadata is to create primary keys for each record in your dataset. See *Generate Primary Keys*.

Append Datasets

You may need to add the rows of one dataset to your current dataset. You can use a union operation to bring in the rows of a second dataset into your current dataset. For example, if you have transactional data, such as orders or log events, you may wish to string these rows together in a single dataset. See *Union Page*.

Combine Datasets

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
C4.3	C5.3		C6.3

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 *Append Datasets*.

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 (**fo reign 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 Data*.

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*.

Reshape Datasets

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*.

Pivot tables

For more information on building aggregated pivot tables, see *Pivot Data*.

Combine Datasets with Parameterization

You can combine individual source files or tables together into a single imported dataset. For example, if you have sets of log files or transactions stored in separate files or tables in the same area, you can use a **parameterized dataset** to combine these datasets together.

Combine Datasets with SQL

Feature Availability: This feature may not be available in all product editions. For more information on available features, see *Compare Editions*.

For relational sources, you can combine data from multiple tables into a single dataset using custom SQL statements that you create. At import, these SQL statements are applied to the datastore, returning the customized dataset for use in Designer Cloud Powered by Trifacta Enterprise Edition.

Create New Column

Contents:

You can create a new column by adding or editing a formula on any existing column.

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.

Transformation Name	`New formula`
Parameter: Formula type	`Single row formula`
Parameter: Formula	`YEAR (TransactionDate)`
Parameter: New column name	`currentyear`

For more information on extracting date information, see *Extract Values*.

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	`baseCost + totalTax - totalDiscount`
Parameter: New column name	`totalCost`

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	`ROUND(DIVIDE(10,3),0)`
Parameter: New column name	`Three`

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:

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.

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)`

Tip: You can use the `FLOOR` or `CEILING` functions to force rounding down or up to the nearest integer.

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:

Transformation Name	`New formula`
Parameter: Formula type	`Single row formula`
Parameter: Formula	`(colA + colB + colC)`

Parameter: New column name	`'colD'`

Math Functions:

Transformation Name	`New formula`
Parameter: Formula type	`Single row formula`
Parameter: Formula	`ADD(colA,colB)`
Parameter: New column name	`'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.

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`:

Transformation Name	`Edit column with formula`
Parameter: Columns	`colA`
Parameter: Formula	`((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:

Transformation Name	`Edit column with formula`
Parameter: Columns	`Cost`
Parameter: Formula	`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(Qty,UnitPrice)`

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	`ordId, prodId`
Parameter: Separator	`'-'`
Parameter: New column name	`primaryKey`

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.

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.

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.

Generate Primary Keys

Contents:

This section describes how you can create primary keys for each row in your dataset.

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 and datasources, like relational sources, make original row order information unavailable.

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`
Parameter: Columns	`keyPrefix,origRowId`
Parameter: New column name	`primaryKey`

4. You can now delete the prefix column:

Transformation Name	`Delete columns`
Parameter: Columns	`keyPrefix`
Parameter: Action	`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:

Transformation Name	`Merge columns`
Parameter: Columns	`LastName,FirstName,TestNum`
Parameter: Separator	`'-'`
Parameter: New column name	`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:

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 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.

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 transformation 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.

A **union** appends two or more similar datasets together. Rows of the second dataset are added to the end of the first.

- You can include the following objects in a union:
  - **A recipe:** The output columns of a recipe are used in the union. Note that changes to the recipe can affect the union.
  - **A dataset:** You can union directly with a dataset to which you have access.
  - **A reference dataset:** A reference dataset is a link to the output of a recipe from another flow.

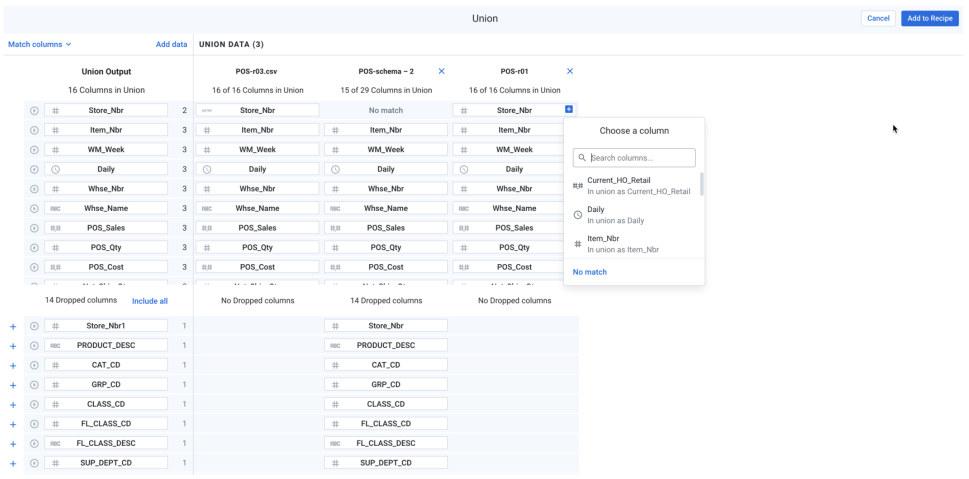
    Tip: You can create a reference dataset from a recipe. In Flow View, click the recipe and select **Create Reference Dataset**.

- In a union, the Designer Cloud® application can match columns for you automatically.
- You can also manually tweak the matching and decide the columns to include or exclude in the resulting dataset.

Steps

1. To start a join, begin from Dataset A:
   a. In Flow View, right-click a dataset and select **Add > Union**.
   b. If your first dataset already has a recipe, click the recipe, and select **Append Union**.
   c. In the Transformer page, you can enter `union` in the Search panel.
2. The Union dialog previews the columns in the output on the left side and the column from the source datasets on the right side.

   a. To add data from a dataset, recipe, or reference, click **Add data**.
   b. From the Choose data to Union dialog, select or search the datasets to union from the available options.
   c. From the drop-down, you can choose how you want to align the columns between datasets. Select o ne of the aligning methods such as **Auto Align**, **Align by Name**, and **Align by Position** to map the columns of the new datasets to the existing datasets.
   d. Click **Apply**.
3. A preview of the columns in the union is displayed.



Union Output		POS-r03.csv	POS-schema – 2	POS-r01
16 Columns in Union		16 of 16 Columns in Union	15 of 29 Columns in Union	16 of 16 Columns in Union
Store_Nbr	2	Store_Nbr	No match	Store_Nbr
Item_Nbr	3	Item_Nbr	Item_Nbr	Item_Nbr
WM_Week	3	WM_Week	WM_Week	WM_Week
Daily	3	Daily	Daily	Daily
Whse_Nbr	3	Whse_Nbr	Whse_Nbr	Whse_Nbr
Whse_Name	3	Whse_Name	Whse_Name	Whse_Name
POS_Sales	3	POS_Sales	POS_Sales	POS_Sales
POS_Qty	3	POS_Qty	POS_Qty	POS_Qty
POS_Cost	3	POS_Cost	POS_Cost	POS_Cost
14 Dropped columns		No Dropped columns	14 Dropped columns	No Dropped columns
Store_Nbr1	1		Store_Nbr	
PRODUCT_DESC	1		PRODUCT_DESC	
CAT_CD	1		CAT_CD	
GRP_CD	1		GRP_CD	
CLASS_CD	1		CLASS_CD	
FL_CLASS_CD	1		FL_CLASS_CD	
FL_CLASS_DESC	1		FL_CLASS_DESC	
SUP_DEPT_CD	1		SUP_DEPT_CD	

Current_HO_Retail
In union as Current_HO_Retail
Daily
In union as Daily
Item_Nbr
In union as Item_Nbr

Figure: Union Summary

4. Review the output columns in the left panel.
    a. To remove a column from the output, click the **X** icon next to the column in the left panel.
5. If needed, you can change the columns and their order in the output dataset based on selections in the right panel:
    a. To completely remove the dataset from the union, click the **X** icon next to the dataset name in the right panel.
    b. To add a column to the union output, click the **Plus** icon to the right of the column entry.
    c. To include all available columns in the output, click **Include all** or select the respective **Plus** icon against the column.
6. To add the union as specified, click **Add to Recipe**.

The rows of data in the unioned datasets are added to the original dataset, and the `union` transformation is inserted in the appropriate recipe.

Example

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. Th is 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.
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_Da te` 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 with the Deduplicate transformation. For more information, see *Deduplicate Data*.

> **NOTE:** Be sure to verify that the data type for each column is accurate.

Join Data

Contents:

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 standard data operation that brings together multiple datasets based on one or more fields that they have in common to render a new dataset.

- Two datasets are compared for joining based on one or more shared columns, called **join keys**.
- There are multiple types of join. The simplest one to understand is the **inner join**. In an inner join, a value that is in the join key column of Dataset A must appear in the join key column of Dataset B.
    - If the join key value appears in both source datasets, a row in which values from the rows in both source datasets is included in the new dataset.
    - If the join key value does not appear in one of the source datasets, in an inner join, the row data from both datasets is excluded from the new dataset.

Overview

Using a Join transformation, you can join a recipe or dataset to any of the following objects:

- Another recipe
- An imported dataset
- A reference dataset

Create Join

You can join datasets through the following mechanisms:

- **Flow View:**

    1. To start a join, begin from Dataset A:
    2. In Flow View, right-click a dataset and select **Add > Join**.
    3. If your first dataset already has a recipe, you can right-click the recipe, and select **Append Join**.
    4. In the Transformer page, you can enter `join` in the Search panel.
    5. **Choose dataset or recipe to join:** select the recipe or dataset to which you want to join to Dataset A

        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.

- **Transform Builder**: Search for and select `Join`.

Joins are created through the Join window. This task is described below.

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.

      NOTE:

   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 colums.
- 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` a nd `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. 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. 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. 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. **NOTE:** 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.	

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*.

- **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 that specifies a range of matching values.

Comparitive operators:

- 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:

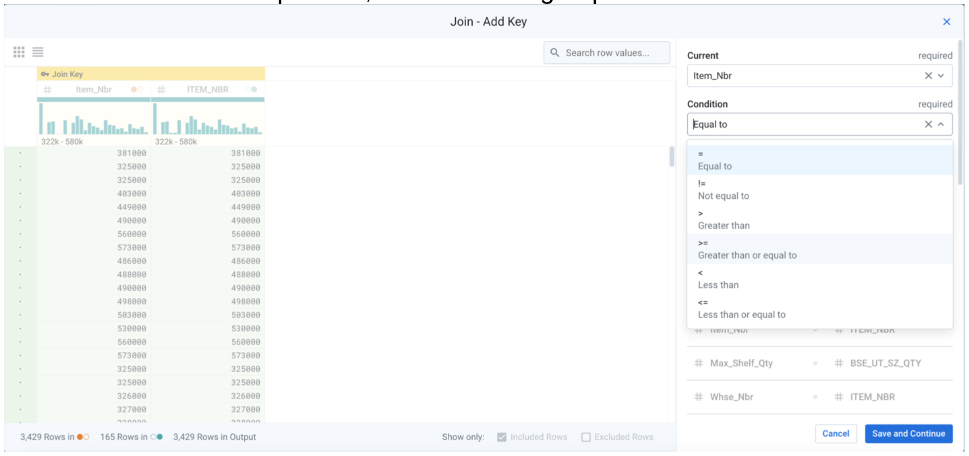


Figure: Select range operator

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**.

Insert Metadata

Contents:

Metadata is data about your data. In some use cases, you may need to insert data about your data into your dataset for downstream consumption.

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 (for non-uploaded files)
- 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 that were loaded from a dedicated storage layer, 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	`$filepath`
Parameter: New column name	`sourceDatasetPath`

Insert source row number

You can insert the row number in the source file from which rows in your dataset are sourced, using the `$sourcerownumber` 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:

Transformation Name	`New formula`
Parameter: Formula type	`Single row formula`
Parameter: Formula	`$sourcerownumber`
Parameter: New column name	`sourceRowNumber`

Tip: Use the ROWNUMBER function to derive the current row number in your dataset.

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:

Transformation Name	`New formula`
Parameter: Formula type	`Single row formula`
Parameter: Formula	`'<root_dir>\/uploads\/1\/2580298d-3477-4907-bfa7-f71978eace04\/SF Restaurants - businesses.csv'`
Parameter: New column name	`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 datasets` transformation. See *Join Data*.

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 using the Union transformation:
    a. Include all columns from both datasets.
    b. Configure the step to perform the union by name, instead of by position.
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.

Transformation Name	`Window`
Parameter: Formula	`FILL(source_system)`
Parameter: Group by	`region`
Parameter: Order by	`business_id`

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:

Through the Search panel, you can access and apply functions that have been developed external to Designer Cloud Powered by Trifacta® Enterprise Edition.

> **NOTE:** This method of invocation applies only to Java UDFs created and applied to a specific deployment of 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.

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.

Transformation Name	`Invoke external function`
Parameter: Column	`colA`
Parameter: Arguments	`100`
Parameter: New column name	`myAdderUDFColumn`

Publishing Tasks

Contents:

When you are ready to apply your recipe across your entire dataset, you run a job. When your recipe is finalized, you can schedule a job for regular execution, so that downstream stakeholders are assured of having fresh data.

Job Execution Process

A **job** is a complex set of tasks to ingest your data from its datasources and deliver your data and recipe to the selected running environment for execution.

A **running environment** is an execution engine designed for transforming large datasets based on a set of scripted steps. A running environment can be:

- Trifacta® Photon is an in-memory running environment, local to the Trifacta node. Trifacta Photon enables faster execution of small- to medium-sized jobs.
- **Remote running environments** provide cloud- or cluster-based running environments for execution of large jobs of any scale. Designer Cloud Powered by Trifacta Enterprise Edition supports various remote running environments, depending on your deployment.

Output objects

A job is executed through an **output object**, which is required for every job.

Tip: If an output object does not exist for the job you are trying to run, the Designer Cloud application creates one for you.

An output object definition includes the following:

- **Running environment:** For best results, select the default.
- **Visual profiling:** Select the visual profiling option to generate a visual profile of the job results. Visual profiling is handled as a separate job executed after the transformation job is complete.
- **Publishing actions**: Define one or more publishing actions to specify:
    - Output datastore, path/database, and output file or table name
    - Output format
    - Update action: create new, append, or replace.
    - Parameterization: create output parameters as needed.
- Other output settings

Tip: A "job" encompasses multiple sub-jobs, which manage the processes of ingestion, conversion, transfer, transformation, profiling, and generating of results as needed to complete the job.

Job types

Jobs can be of the following types:

- **Manual:** Need results? Click **Run** to launch a job right now.
- **Scheduled:** If you need results to be scheduled at a specific time, you can set up a scheduled execution.

NOTE: Both types of jobs require output objects. For any recipe, you can create different output destinations for manual or scheduled jobs.

Tip: Jobs can also be triggered using REST APIs, if you prefer to handle job scheduling outside of the De signer Cloud application .

Run Job to Generate Results

NOTE: Running a job consumes resources. Depending on your environment, resource consumption may cost money. Your project owner or workspace administrator may be able to provide guidance on resources and their costs.

To run a job right now, you can do either of the following:

1. In the Transformer page, click **Run**.
2. In Flow View, click the output to generate. In the right panel, click **Run**.

Tip: By default, a manual job generates a CSV with visual profiling to the default output location using the optimal running environment for the job size. In the Run Job page, you can define or update your output object and its publishing actions, as needed.

For more information, see *Run Job*.

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:

Run Jobs

When your recipe is ready, you can execute a job to generate output results of your transformations applied to your dataset.

For more information, see *Overview of Job Execution*.

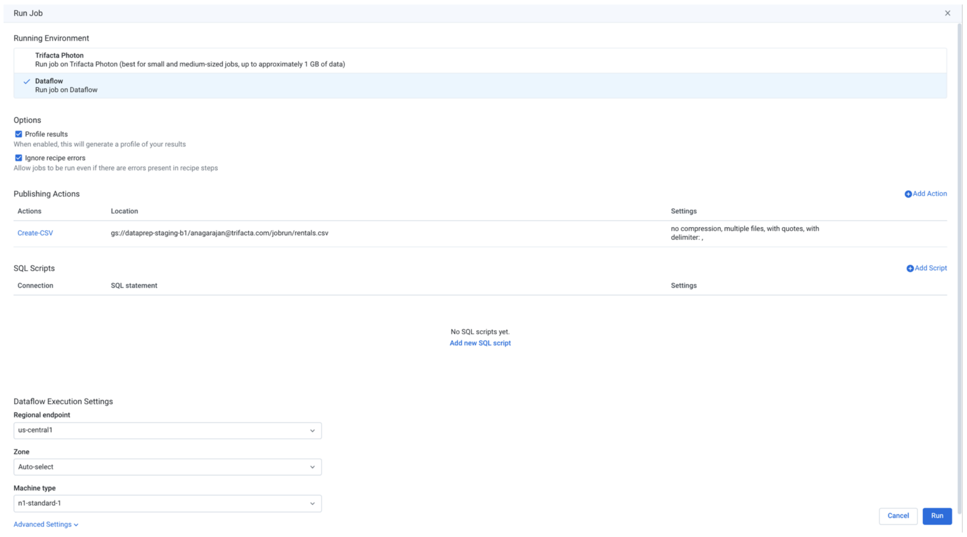
Prerequisites

- To generate results, you must define an output object, which includes information about output format, location, and other settings.
    - You can define an output as part of the process of running a job.
    - For more information, see *Create Outputs*.

Run Job

1. From the Transformer page, click **Run**.
2. In the Run Job page, a running environment is selected for you.

    Tip: Unless you have a specific reason to do so, you should use the default running environment, which is chosen based on the size of the job for you.

    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)

    ✓ **Dataflow**
    Run job on Dataflow

    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

Actions	Location	Settings
Create-CSV	gs://dataprep-staging-b1/anagarajan@trifacta.com/jobrun/rentals.csv	no compression, multiple files, with quotes, with delimiter: ,

    SQL Scripts

Connection	SQL statement	Settings

    No SQL scripts yet.

    Dataflow Execution Settings

    Regional endpoint: us-central1

    Zone: Auto-select

    Machine type: n1-standard-1

    Figure: Run job page

    NOTE: When you select the Spark environment, the **Advanced environment options** are displayed. These options pertain to various limits that Spark can apply to the runtime execution of the job.

3. Options:
    a. **Profile results:** A visual profile of your output results can be generated for review. For more information, see *Profile Results*.
    b. **Ignore recipe errors:** When selected, the job is permitted to execute even when there are known errors present in the recipe.
4. To add a new publishing destination, click **Add Action**.
    a. In the publishing action page, specify the connection and location.
    b. For the specified output, set the preferred settings in the right-side panel and click **Add**.
    c. See *Create Outputs*.

5. If needed, you can define SQL scripts that are executed before or after generation of your output objects. T o add SQL scripts, click **Add Scripts**.
6. To execute the job to generate the specified output, click **Run**.

To cancel a job in progress, click **Cancel job** in the Job Details page.

For more information, see *Run Job Page*.

Create Outputs

Contents:

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.
- Table-based outputs such as Oracle or PostgreSQL.

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 Settings, click **Edit.**
5. In the Publishing Settings page, click **Add Publishing Action**.

Tip: For scheduled runs of your flow, you must specify Scheduled Settings to automatically generate the output when the flow is executed by a schedule. For more information on scheduling, see *Overview of Scheduling*.

From Run Job page:

For an existing output, you can create new destinations from the Run Job page.

1. In Flow View, click **Run Job**.
2. In the Run Job page, click **Add Action** to add a new destination.

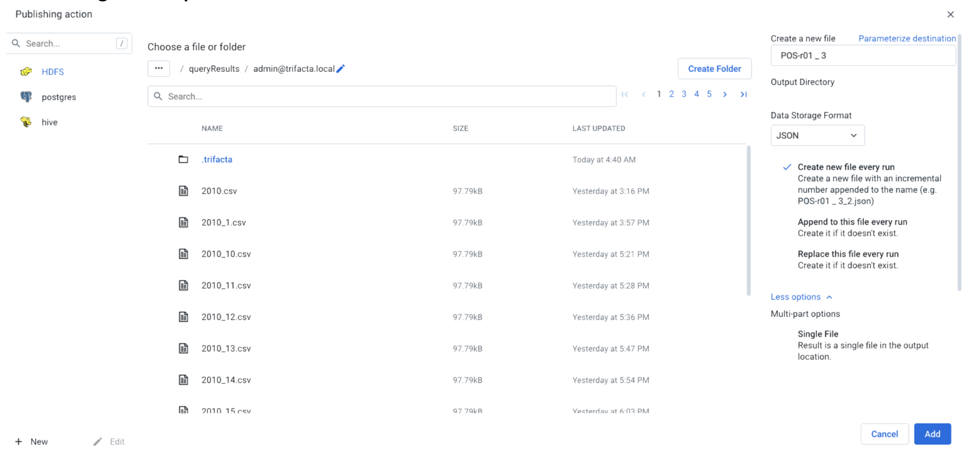
Create a File-Based Output

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:



NAME	SIZE	LAST UPDATED
.trifacta		Today at 4:40 AM
2010.csv	97.79kB	Yesterday at 3:16 PM
2010_1.csv	97.79kB	Yesterday at 3:57 PM
2010_10.csv	97.79kB	Yesterday at 5:21 PM
2010_11.csv	97.79kB	Yesterday at 5:28 PM
2010_12.csv	97.79kB	Yesterday at 5:36 PM
2010_13.csv	97.79kB	Yesterday at 5:47 PM
2010_14.csv	97.79kB	Yesterday at 5:54 PM
2010_15.csv	97.79kB	Yesterday at 6:03 PM

Create a new file: POS-r01 _ 3
Parameterize destination
Output Directory
Data Storage Format: JSON

✓ Create new file every run
Create a new file with an incremental number appended to the name (e.g. POS-r01 _ 3_2.json)

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.

Less options

Multi-part options

Single File
Result is a single file in the output location.

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 *File Settings*.
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*.

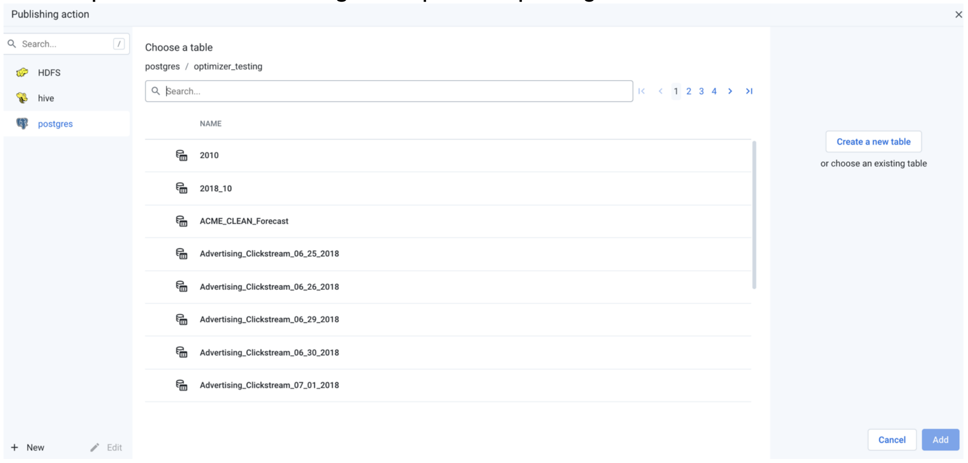
Create a Table-Based Output

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:



2010
2018_10
ACME_CLEAN_Forecast
Advertising_Clickstream_06_25_2018
Advertising_Clickstream_06_26_2018
Advertising_Clickstream_06_29_2018
Advertising_Clickstream_06_30_2018
Advertising_Clickstream_07_01_2018

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 *Relational Table Settings*.
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**.

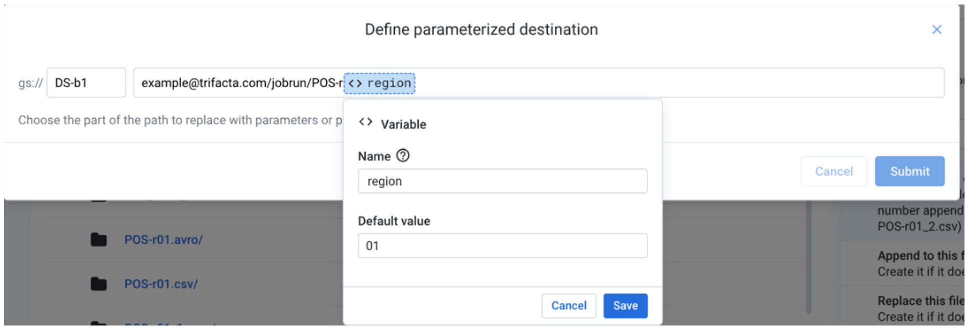

Define parameterized destination
gs://
DS-b1
example@trifacta.com/jobrun/POS-r
region
Choose the part of the path to replace with parameters or p
Variable
Name
region
Default value
01
Cancel
Save
Cancel
Submit
POS-r01.avro/
POS-r01.csv/

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.

   > **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.

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 context panel.
2. In the context panel, select the Manual Settings tab. Then, 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 Job 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 Job page:

In the Run Job page, you can delete publishing actions. From the context menu for a publishing action, select **Del ete**.

Create Output SQL Scripts

Contents:

Feature Availability: This feature may not be available in all product editions. For more information on available features, see *Compare Editions*.

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.

- A warning message is displayed if the pre/post SQL scripts do not have any valid connection.

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.
- 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 Settings destinations:** In Flow View, you can select an output object and then modify one of its Manual Settings destinations.
- **Scheduled Settings destinations:** In Flow View, select an output object and then modify one of its Scheduled Settings destinations.

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 Manual Settings 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 *Overview of Job Monitoring*.

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 Manual Settings 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 Manual Settings 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: • `pre` - execute before data ingest • `post` - execute after data publish
`vendor`	The vendor type of the database to which you are connecting. See *Connection Types*.
`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*

Profile Results

Contents:

When you run a job, you can optionally choose to generate a visual profile of the results of the job. A visual profile provides visual and statistical summary information on each column in your output dataset.

A visual profile provides visualizations on your dataset that you can use to refine your recipes or analyze the outputs of them. Since Designer Cloud Powered by Trifacta® Enterprise Edition can manage large volumes of data, visual profiling assists in analysis by summarizing column data in useful metrics, as determined by data type. For example, columns of Zip Code data type can be represented on a geographical map of the United States.

> **NOTE:** Generating a visual profile can take more time and system resources to complete.

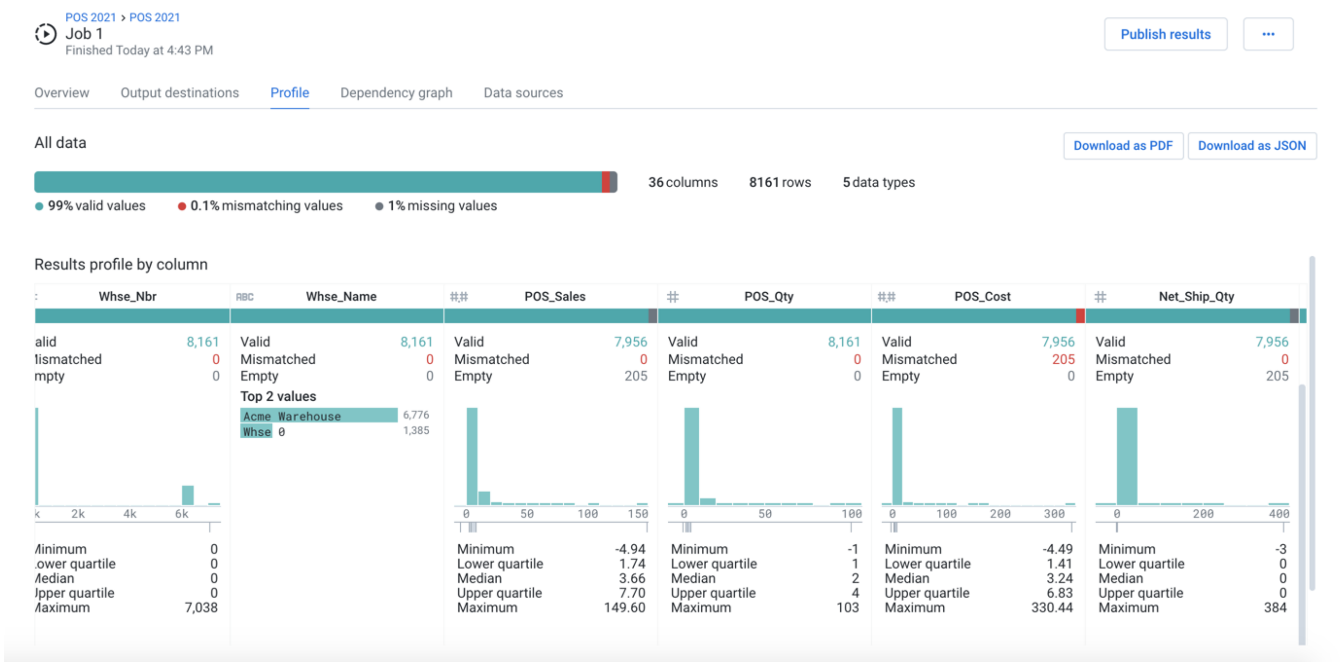
For more information, see *Overview of Visual Profiling*.

Generate Visual Profile

When you are specifying your job in the Run Job page, select the **Profile results** checkbox. Then, run the job. See *Run Jobs*.

Access Visual Profile

After the job has completed, you can review the visual profile in the Profiles tab of the Job Details page.



POS 2021 > POS 2021
Job 1
Finished Today at 4:43 PM

All data

36 columns 8161 rows 5 data types

● 99% valid values ● 0.1% mismatching values ● 1% missing values

Results profile by column

	Whse_Nbr	Whse_Name	POS_Sales	POS_Qty	POS_Cost	Net_Ship_Qty
Valid	8,161	8,161	7,956	8,161	7,956	7,956
Mismatched	0	0	0	0	205	0
Empty	0	0	205	0	0	205
Minimum	0		-4.94	-1	-4.49	-3
Lower quartile	0		1.74	1	1.41	0
Median	0		3.66	2	3.24	0
Upper quartile	0		7.70	4	6.83	0
Maximum	7,038		149.60	103	330.44	384

Whse_Name — Top 2 values:

Acme Warehouse	6,776
Whse 0	1,385

Figure: Example visual profile in Job Details page

In the above example, you can see summary statistics for the entire result dataset and breakdowns for individual columns based on data type:

- Columns containing numeric data types include common statistical metrics including a histogram indicating frequency of values.
- Since any value can be a String value, these columns illustrate only a histogram of values in the column.
- Valid (green), mismatched (red), and empty (black) values are noted in column histogram bars.

Download Visual Profile

Visual profiles can become analytical datasets for use in other systems. You can download your visual profile in PDF or JSON format from the Job Details page.

Tip: JSON versions of the visual profile can be loaded into other systems for downstream uses.

Review Profiles in Columns

You can also review visual profiles of individual columns in development in the Transformer page and Column Details.

Tip: Visual profiles of column data in the Transformer page and Column Details page are interactive. When you select bars in the column histogram or data quality bar, suggestions for transformations are displayed, enabling you to rapidly transform your data of interest.

- For more information, see *Data Grid Panel*.
- For more information, see *Column Statistics Reference*.

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 Job History icon.
- In the Job History page, select the job whose results you wish to publish.

> **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.

- In the Job Details page, 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.

- Close the window when you are done.

Reuse Recipe

Contents:

This section describes multiple ways in which you can leverage a recipe developed in one flow in other flows.

Reuse Recipe with the Same Flow

The easiest way to reuse a recipe is to change its inputs in Flow View.

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. In the Flow View menu, select **Duplicate** from the flow context menu.

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...**.
- In the Recipe panel in the Transformer page, click the context menu, and select **Download recipe as Wrangle** .

Project Management Tasks

These topics provide guidance on how to better manage your data wrangling efforts in Designer Cloud Powered by Trifacta® Enterprise Edition.

Take a Snapshot

Contents:

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.

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.

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:

You can use these techniques for tracking changes to your datasets over time.

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.

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.

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*.

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.

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 task 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:

Transformation Name	`New formula`
Parameter: Formula type	`Single row formula`
Parameter: Formula	`String`
Parameter: New column name	`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:

Transformation Name	`Edit with formula`
Parameter: Columns	`String`
Parameter: Formula	`TRIM(String)`

4. At the point in your recipe where you would like to test the column for changes, insert the following:

Transformation Name	`New formula`

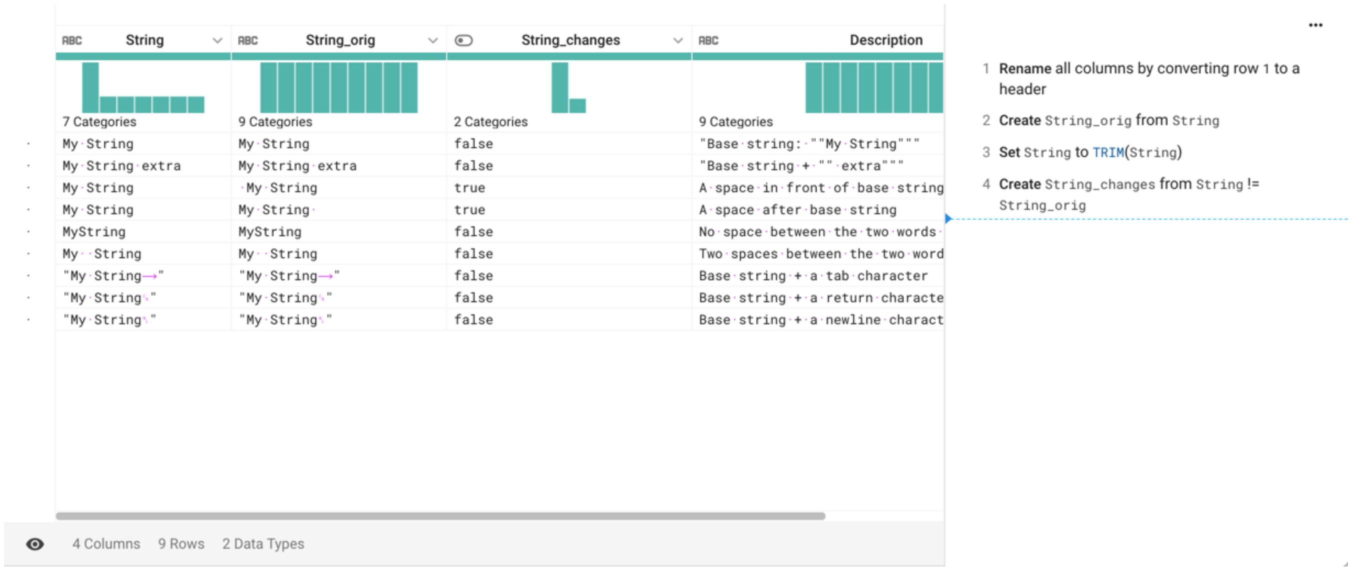
Parameter: Formula type	`Single row formula`
Parameter: Formula	`String <> String_orig`
Parameter: New column name	`String_changes`

5. The `String_changes` column now contains `true` values where the values in `String` have been changed from their original values (`String_orig`).
6.

   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.

7. 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
7 Categories	9 Categories	2 Categories	9 Categories
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·characte
"My·String"	"My·String"	false	Base·string·+·a·newline·charact

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:

1. In Flow View, create a copy of the flow. Create a name for it that identifies it as your original.
2. In the other flow, create your recipes as normal.
3. When done, you can add the following steps:
   a. Union the two datasets together.
   b. Sort them by a key column.
   c. 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 Target

Contents:

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 Target Schema Mapping*.

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 Transformer toolbar in the Transformer page.

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

- **Data grid:** 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.
- **Column Browser:** In the Column Browser panel, you can select one or more columns and apply schema-related transformations to them.

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:

This page contains a set of tips for how to improve the overall performance of job execution.

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.

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.
  - 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 Data*.

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 *Append Datasets*.

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.

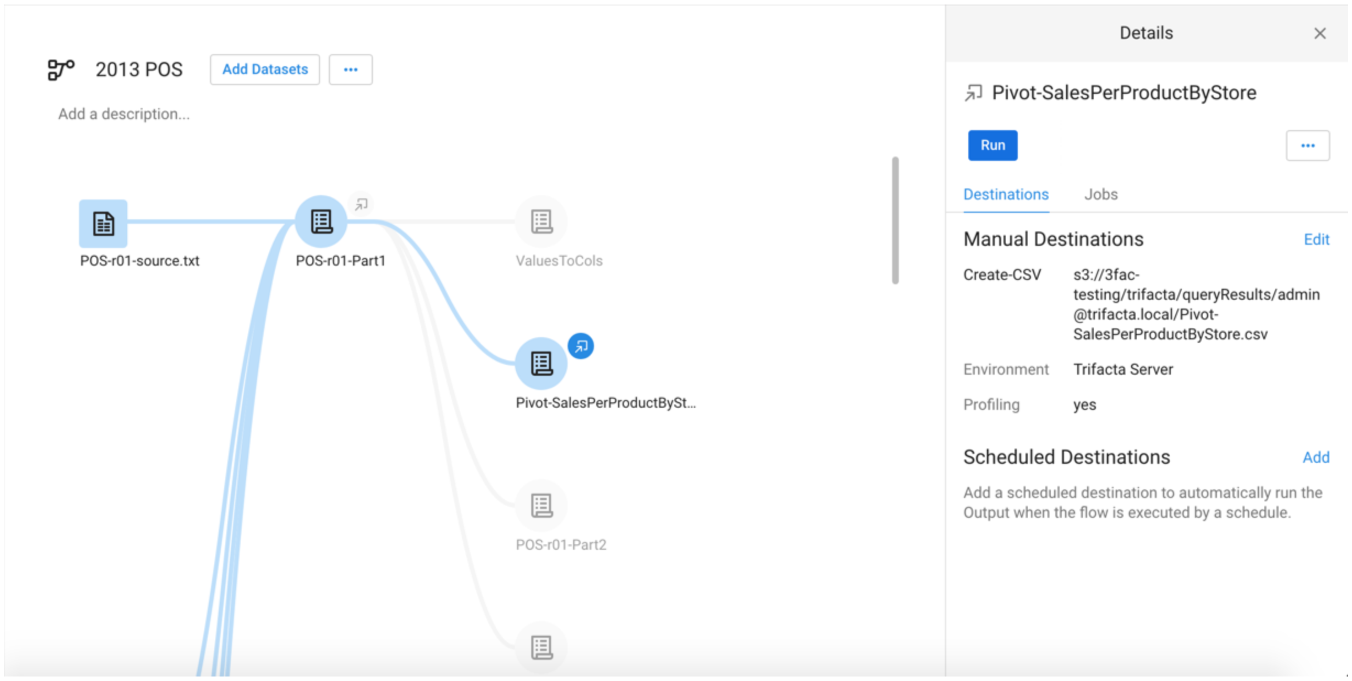
Transformation Name	Description
Union	A union appends one or more datasets to your current one. To preserve the original, you may need to create a branching output. See *Append Datasets*.
Join	A join combines two datasets based on common values in specified columns in both datasets. These types of transformations can greatly change the shape of your data. See *Join Data*. 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 *Add Lookup Data*.
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 *Remove Data*.
Filter	Rows can be filtered from your dataset to render different perspectives. These changes may be best moved to a secondary, branching recipe. See *Filter Data*.
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 *Pivot Data*.
Group by	You can perform aggregation calculations within a table, which may force column data to be dropped. See *Create Aggregations*.

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 Scheduling*.



Pivot-SalesPerProductByStore

Manual Destinations Edit

Create-CSV	s3://3fac-testing/trifacta/queryResults/admin@trifacta.local/Pivot-SalesPerProductByStore.csv
Environment	Trifacta Server
Profiling	yes

Scheduled Destinations Add

Add a scheduled destination to automatically run the Output when the flow is executed by a schedule.

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

Contents:

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.

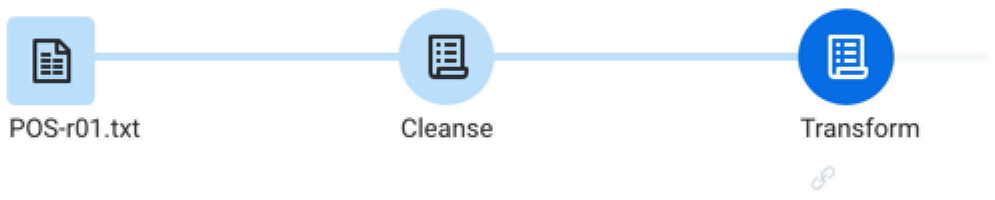


Figure: Chained recipes

Create Reference Objects

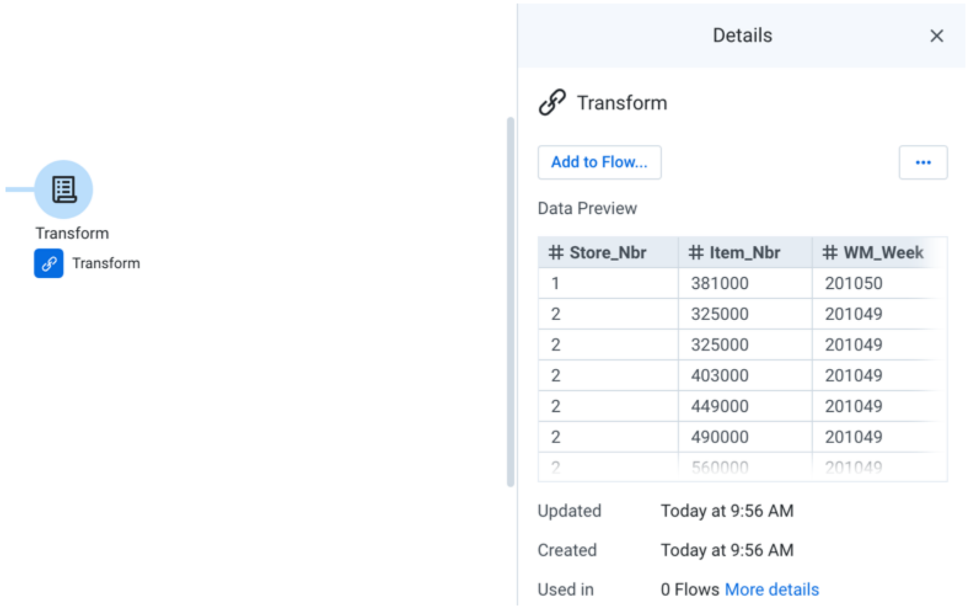
A **reference dataset** is a reference to a recipe's output, which can be used in another flow like an imported dataset. When you modify the source recipe, the reference dataset that is used in another flow automatically receives the changes.

If you need to make the output of a recipe available in other flows, you can create a reference dataset. This reference is available in other flows that you control.

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 object.

Steps:

1. In Flow View, select the recipe whose output you wish to make available to other flows.
2. Click the Create Reference icon:



# 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

3. To use it in one of your other flows, click **Add to Flow...**.
4. 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.

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:
    - **Dataset with parameters:** 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.

    - 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.

Fix Dependency Issues

Contents:

This section describes how to identify dependency issues between your current recipe and other recipes or datasets and includes general steps for fixing them.

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.

How to Identify

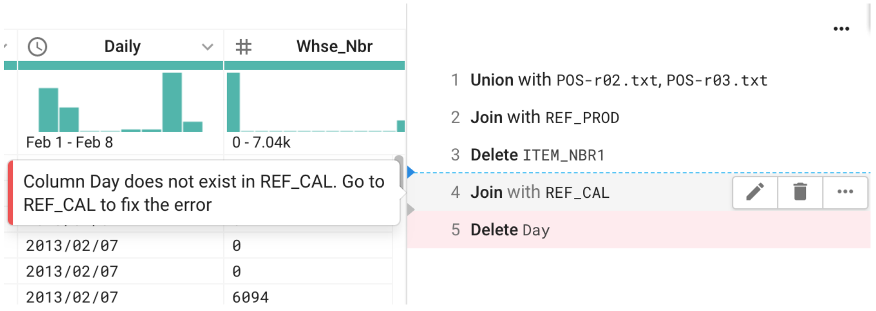
Dependent datasets

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.

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:



Column Day does not exist in REF_CAL. Go to REF_CAL to fix the error

```
2013/02/07   0
2013/02/07   0
2013/02/07   6094
```

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

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).
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:

Transformation Name	`Filter rows when value is exactly`
Parameter: Condition	`Is exactly`
Parameter: Column	`company_name`
Parameter: Value	`'My Co.'`
Parameter: Action	`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 from the 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.

   Tip: You can also share the flow from the Flow view page through the flow context menu.

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.

   NOTE: For privacy reasons, search may not be available in some environments. In such cases, you have to manually enter the user name in the **Invite users** field.

4. You can specify the privilege level of the user to whom you are sharing.
5. Click **Add**.
6. The selected users can now see the flow and interact with its objects in the Shared with Me tab of the Flows page.

NOTE: When a flow is shared, its samples are shared with other users. If the shared users do not have access to the underlying source datasets, they do not have access to samples generated from those datasets and must recreate new samples.

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 and outputs. Exported flows can be imported into different projects and workspaces.

Flow export is useful for:

- Backups of work in progress

> **You cannot import flows that were exported from a version before Release 6.8.**

- 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.

> **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:

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

> **NOTE:** 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

  NOTE: Reference dataset objects may refer to connections and parameters that are not specifically part of the flow being imported. These references must be remapped in the new project or workspace.

- Samples
- Connections

  Tip: When you import an exported flow, you can perform a remapping of the connections listed in the flow to connections in the new project or workspace. Details are below.

  NOTE: If you are importing your flow into Deployment Manager, 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 through APIs. For more information, see *API Task - Define Deployment Import Mappings*.

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*.

Before You Begin

Dry-run import

Tip: You can gather some of the following information by performing a dry run of importing the flow. In the Connections Mapping dialog, you can review the objects that need to be mapped and try to identify items that need to be fixed. Make sure you cancel out of the import instead of confirming it.

Define import rules

During import of a package into Deployment Manager, 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 *API Task - Define Deployment Import Mappings*.

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.

Import flow 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**.

   Tip: You can import multiple flows (ZIP files) through the file browser or through drag-and-drop. Press `CTRL/COMMAND` + click or `SHIFT` + click to select multiple files for import.

5. Select the ZIP file containing the exported flow. Click **Open**.

Flow mappings

When you import a flow into a new project or workspace, you may need to remap the connections and environment parameters to corresponding objects from the import.

These mappings can be performed via API. For more information, see *API Task - Define Deployment Import Mappings*.

Otherwise, you can fix them manually after import by repairing the broken references that appear in the flow and recipe. See below.

Finishing up

To complete the import process, click **Confirm.**If there are issues with the import, click the Download link to review the missing or malformed objects.The flow is imported and available for use in the Flows page.

Post-import

- If you were unable to complete mappings through the dialog:
  - 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*.
  - An administrator may need to create environment parameters for you. See *Environment Parameters Page*.
- Samples are not included in an exported flow. Samples must be recreated in the new project or workspace. See *Samples Panel*.

NOTE: If you have imported a flow from an earlier version of the application, you may receive warnings of schema drift during job execution when there have been no changes to the underlying schema. This is a known issue. The workaround is to create a new version of the underlying imported dataset and use it in the imported flow.

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

After 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, you can use the displayed dialog to perform remappings. Use the steps below if you need to perform updates after the import is completed.

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. In the Connections page, you may need to recreate the connections that were used to the import the dataset in the original workspace.
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.
    c. Select **Replace**.
    d. The selected dataset is replaced with an another dataset.
5. Repeat the above steps for each broken dataset in the flow.

You may also need to remap the flow's outputs.

Reconnect Flow to Outputs

After you import a flow into a new project or workspace, you may need required to remap the flow outputs to accessible publishing destinations.

Tip: When you import the flow, you can use the displayed dialog to perform remappings. Use the steps below if you need to perform updates after the import is completed.

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.
    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:

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.

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:**Macro Definition:**

- **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.

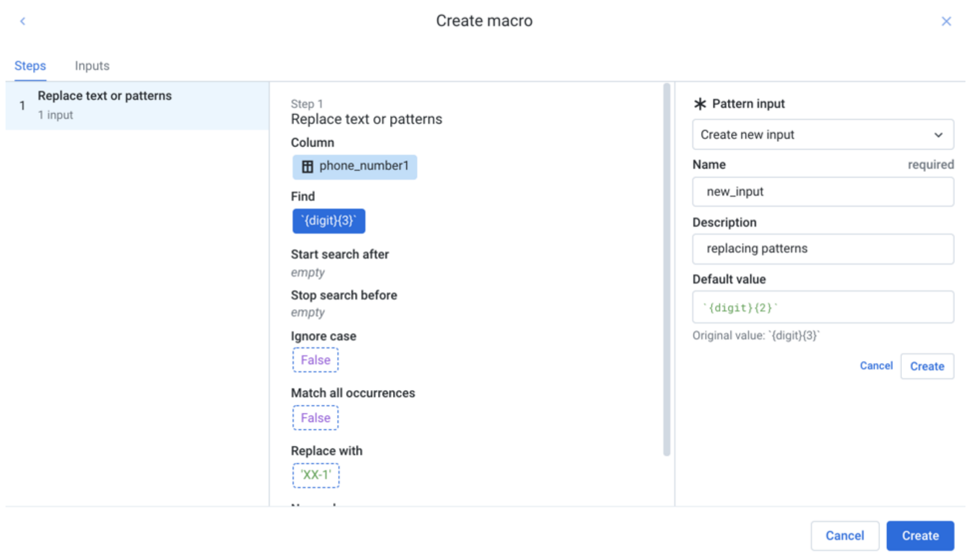
2. From the recipe toolbar context menu, select **Create or replace macro**.

    > **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:



Create macro

Steps | Inputs

1 Replace text or patterns
1 input

Step 1
Replace text or patterns

Column: phone_number1

Find: \`{digit}{3}\`

Start search after: *empty*

Stop search before: *empty*

Ignore case: False

Match all occurrences: False

Replace with: 'XX-1'

Pattern input: Create new input

Name (required): new_input

Description: replacing patterns

Default value: \`{digit}{2}\`

Original value: \`{digit}{3}\`

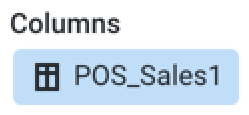
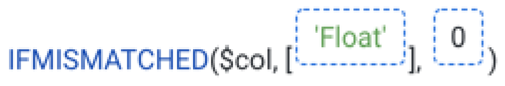
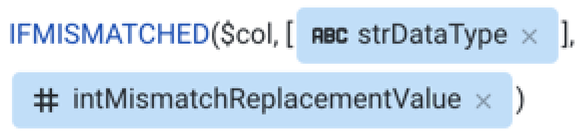
Figure: Create macro inputs

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
 Columns POS_Sales1	Column names are automatically turned into inputs.
 IFMISMATCHED($col, ['Float'], 0)	These two values could be turned into macro inputs but are not currently defined as such. `IFMISMATCHED` is a a function name, which cannot be parameterized as inputs.
 IFMISMATCHED($col, [ABC strDataType ×], # intMismatchReplacementValue ×)	These two values have been turned into macro inputs.

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 from the Macros page, which forces all instances of the macro in the workspace to be automatically converted to steps.

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.

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.

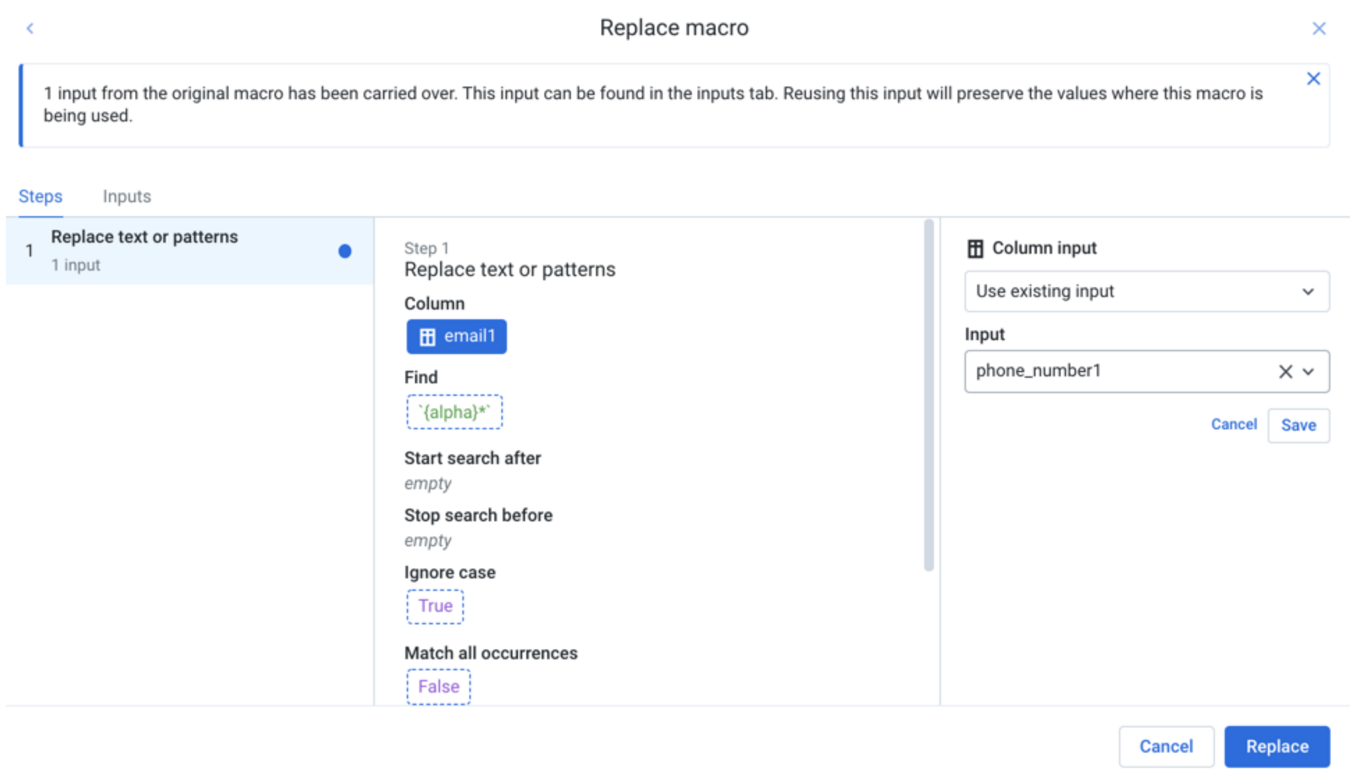
      NOTE: If you discard and save the changes, then any references to those inputs in the instances of the macro in the previous definition are broken.

   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.


Replace macro

1 input from the original macro has been carried over. This input can be found in the inputs tab. Reusing this input will preserve the values where this macro is being used.

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 Macros page in the Library.

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.

Apply a Macro

Contents:

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, 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.
3. In the Transformer toolbar, click the Macros icon.

    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**.

Create Flow Parameter

Contents:

At the flow level, you can define flow parameters to reference in your recipes. A **flow parameter** is a (Variable) value or a set of enumerated values (Selector type).

> **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. (optional) Specify a user-friendly Description value.
7. For Type, select the type of parameter. See below.

Variable type parameter

Variable type parameters are single values that can be one of the following formats:

Format	Example
String	`Here is my String value.`
Trifacta patterns	Any string of letters: `` `{letter}+` ``
regular expression	Any string of letters: `/[a-zA-z]+/`

Additional steps:

1. Enter a default value for this parameter.

   NOTE: Input Values are evaluated as String type.

2. Click **Save**.

The parameter is available for use in any recipe in your flow. See "Use Parameter."

Selector type parameter

Selector type parameters are lists of one or more permitted values.

Additional steps:

1. Under Choices, enter a value.

Tip: The first value that you add is the default value.

2. Click **Add**.
3. Repeat the above steps until all permitted values have been added.
4. 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`	`$paramRegion` `${paramRegion}`	Both references are valid.
`param Region`	`${param Region}`	**NOTE:** 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 *Source Metadata References*.

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.

NOTE: For selector type flow parameters, the override value must be a value from the defined list of values.

- **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:** Through Flow View, overrides can also be applied to the recipe parameters that are included when flow tasks are executed as part of a plan.

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.

Variable type 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:

Transformation Name	`New formula`
Parameter: Formula type	`Single row formula`
Parameter: Formula	`PARSEINT(${paramregionId})`
Parameter: New column name	`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.

Variable type 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`		In this case, the value is left empty to be overridden as needed in the application with the current timestamp. You should decide on the expected values for this parameter, as you must apply them to: • Parameter overrides • Recipe steps (e.g PARSEDATE function parameters) It may be easier to insert the format string here as the default value. For example: `yyyy-mm-dd HH:MM:SS`

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:

Transformation Name	`New formula`
Parameter: Formula type	`Single row formula`
Parameter: Formula	`PARSEDATE(${paramDate},['yyyy-mm-dd HH:MM:SS'])`
Parameter: New column name	`paramDate`

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.

Transformation Name	`Edit column with formula`
Parameter: Columns	`execDate`
Parameter: Formula	`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.

You can use the following to extract the time value from the parsed date param:

Transformation Name	`New formula`
Parameter: Formula type	`Single row formula`
Parameter: Formula	`DATEFORMAT(execDate, 'HH:MM:SS')`
Parameter: New column name	`Time`

Since this value is not the parameter value specifically, the column name was listed simply as `Time`.

Variable type example - parameter with range of values

Suppose you need to create a parameter that can contain any two-digit value from 0 to 99. To do so, you create a flow parameter of variable type, using a Pattern as the value:

```
`{digit}{digit}`
```

The above matches two-digit values only. It does not match single-digit values. To expand the pattern to also match single digit values, you can use this one:

```
`({digit}|{digit}{digit})`
```

- Note how the value is specified using backticks (`` ` ``), which are used to indicate a Pattern .
- The vertical bars are delimiters to separate the values, when they are processed within the application.

This parameter now accepts the following values:

```
0-9
00-99
```

Selector type example - parameter with multiple values

Suppose you wish to create a flow parameter that can contain one of multiple values for color. Typically, you must track these values through an array, such as the following containing a set of colors:

```
["red","white","blue","black"]
```

Using a selector type of flow parameter, you can specify these values as choices in your permitted values list for the `$myColors` flow parameter:

```
red
white
blue
black
```

In this scenario, `red` is the default value.

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`:

Transformation Name	`New formula`
Parameter: Formula type	`Single row formula`
Parameter: Formula	`F($myColors == "blue",true,false)`
Parameter: New column name	`'isBlue'`

Tip: When you reference a selector type flow parameter in a transformation step, you are presented the set of accepted values from which you apply the preferred one.

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 parmeters
- flow parameters

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

Flag for Review

Contents:

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 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.

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.

Export Data

Contents:

You can export the generated job results from Designer Cloud® application to your preferred destination. You can also download your work-in-progress sampled data to your local desktop.

Export Job Results

Steps:

1. From the left navigation bar, click the **Job history** icon.
2. In the Flow Jobs page, click the link to the job that generated the desired outputs.
3. In the Job Details page, click the **Output destinations** tab.
4. For each output, the following options are available in the context menu:
    a. **Download result**: To download the results from a generated output, click **Download result**. The result set is downloaded to your local environment.
    b. **Create new dataset**: You can create a new dataset from a generated output, click **Create new dataset**. A new imported dataset is created and added to the Library.

    > **Tip:** Click the link that is displayed to open the dataset. You may wish to rename the dataset with a more meaningful name.

Publish datasets

You can publish the results to the preferred target if Designer Cloud Powered by Trifacta Enterprise Edition is integrated with an external datastore.

To publish the results to the preferred target, click **Publish results**.

Export Sample Data

From the Transformer page, you can export what is currently displayed in the data grid to your local desktop. Downloads are in CSV format.

At any point in time, the data grid displays the currently selected sample with the recipe steps applied to it, up to the currently selected recipe step. For example, if you download the data from the recipe when the recipe cursor is on step 10, then you are downloading the currently selected sample with the state of the recipe at step 10 applied to it.

> **Tip:** This method is useful for checkpointing or assessing status on your wrangling project.

Steps:

1. In the Transformer page, open the Recipe panel.
2. Select the step where you want to see the sampled dataset. From the step's context menu, select **Go to selected**. The data grid is updated to show the state of the sample and the recipe at the point after the selected step has been executed.

3. In the Recipe panel, click the **Recipe Options** icon.
4. From the Recipe Options icon, select **Download Sample data as CSV**. The contents of the data grid are downloaded to your local desktop.

Operationalization Tasks

These topics cover how to operationalize your Trifacta® flows for use in your enterprise data pipelines.

Operationalization includes all tasks that occur after principle flow development has completed. After you have your flow working and delivering useful data, operationalization can include:

- **Scheduling.** For more information, see *Schedule a Job*.
- **SQL scripting.** Before or after job execution, you can configure your job to execute a specified SQL script. For more information, see *Create Output SQL Scripts*.
- **Macros.** You can build reusable sets of recipe steps, which can be used in other flows and workspaces for consistency. See *Create or Replace Macro*.
- **Flow Webhooks**. After a flow has executed, you can configure a webhook task to deliver messages about the flow execution to other systems. See below.
- **Plans.** A plan is a sequence of tasks that can be executed based on logic and can be scheduled. See below.

Create Flow Webhook Task

Feature Availability: This feature may not be available in all product editions. For more information on available features, see *Compare Editions*.

Contents:

You can send webhook messages to third-party applications based on the results of job executions in your flow.

- A **webhook task** is a callback message between Designer Cloud Powered by Trifacta® Enterprise Edition and another application. They are typically delivered using JSON over HTTP and can be interpreted by the receiving application to take action.

  NOTE: Your receiving application may require that you whitelist the host and port number or IP address of the platform. Please refer to the documentation for your application.

- A webhook task is defined at the flow level, although an individual webhook task can be restricted to specific outputs. It is shared between ad-hoc and scheduled executions.
- This capability may need to be enabled in your environment. For more information, see *Workspace Settings Page*.
- Additional configuration may be required. See *Configure Webhooks*.

For more information on how to orchestrate execution of your flows, see *Overview of Operationalization*.

Limitations

- Custom security certificates cannot be used.
- HTTP-based requests have a 30-second timeout limit.
- Webhook tasks are not included when a flow is copied. They are available to collaborators for review, editing, and execution, when a flow is shared.

  Tip: You can export and import the flow, which includes the webhook task definition.

- You can create a maximum of 50 webhooks per flow.

NOTE: Administrators can change this limit as needed. For *Configure Webhooks*.

Prerequisites

NOTE: It's possible that webhook requests can be submitted back to the platform to execute API tasks within the platform. However, there are security concerns. Additional configuration is required. For more information, see *Configure Webhooks*.

Requirements for receiving application

To send webhooks to a target application, the application must be configured to receive the webhook:

- Incoming webhooks must be enabled.

  NOTE: Your receiving application may require that you whitelist the host and port number or IP address of the platform. Please refer to the documentation for your application.

- You must acquire the URL of the endpoint to which to send the webhook request.
- You must acquire any HTTP headers that must be inserted with each webhook request.
- If the request must be signed, additional configuration is required. Details are below.

Steps

1. Open your flow in Flow View. From the flow context menu, select **Webhooks**.
2. In the right panel, select **Create webhook task**.
3. Set the following parameters:

Parameter	Description
Name	User-visible name of the task.
Url	URL where the webhook message is received by the other application.
Trigger event	Select the event that triggers the message.
Trigger object	Select the object or objects that can trigger the message: `Any job executed in this flow` - Any scheduled or ad-hoc job triggers the message `Only specific objects` - Select the output or outputs whose success or failure triggers the message

Headers	Insert HTTP content headers as key-value pairs. For example, if your body is in JSON format, you should include the following header:
	`key: Content-Type`
	`value: application/json`
	NOTE: You may be required to submit an authentication token as the value for the `Authorization` key.
	Please refer to the documentation for your receiving application about the required headers.

Body	(`POST`, `PUT`, or `PATCH` methods only) The body of the request submitted to the receiving application. In the body, you can use the following references: `jobId` - the internal identifier for the jobGroup that was executed. `jobStatus` - the status for the job after execution. For more information, see *Job History Page*. You can apply

	metadata references to the flow in the Body text. See below for examples.
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.
Secret key	(Optional) A secret key can be used to verify the webhook 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. For more information on how this secret key is used to generate a signature, see Verify Webhook Signatures below.
Validate SSL certificate	When set to `true`, HTTPS (SSL) communications are verified to be using a valid certificate before transmission. **NOTE:** If you must send a request to an endpoint that has an expired/invalid certificate, you must disable SSL verification.
Retry on failure	If the returned status code is outside of the 200-299 range, then the webhook is considered to have failed. When this option is enabled, the request is retried.The number of retry attempts can be configured. See *Configure Webhooks*.

4. To test the connection, click **Test**. A success message is displayed.
5. To add the webhook task to the flow, click **Create**.
6. When the job is executed:
    a. Depending on the outcome, the webhook task is executed through the other application.
    b. The webhook is listed in the Job Details page.

Flow metadata references in body

In the body of your webhook, you can use the following references:

Reference	Description
`$jobId`	Internal identifier to the job in the platform.
`$jobStatus`	The current status of the webhook job. For more information on job status messages, see *Job History Page*.

Examples

Run another job

You can create a webhook task to run another job on the successful execution of this one.

Tip: Use this method to create conditional sequences of job executions.

As needed, you can specify webhook overrides as part of a launching a job via API. For more information, see *API Task - Run Job*.

Prerequisites

NOTE: For this example, the platform must be whitelisted to receive webhooks from itself. Additional configuration is required. For more information, see *Configure Webhooks*.

You must acquire the recipe identifier for the next job to execute.

1. Open the flow containing the next recipe.
2. In Flow View, click the recipe whose outputs you wish to generate.
3. Review the URL for the recipe object. In the example below, the recipe Id value is 4:

```
http://www.example.com:3005/flows/1?recipe=4&tab=recipe
```

4. Retain this value for below.

Define the flow webhook task

Parameter	Description
Name	This name appears in the Designer Cloud application only.
Url	Specify the URL as follows, replacing the example values with your own: `http://www.example.com:3005/v4/jobGroups/`
Trigger event	Select `Job success`.
Trigger object	Select the `any` option to execute all jobs in the target flow, or you can specify individual jobs to execute.

Headers	Insert the following two headers:
	 `key: Content-Type`
	 `value: application/json`
	 `key: Authorization`
	 `value: Bearer <paste your access token here>`
	 NOTE: The token value must be preceded by the string: `Bearer`.

Body	In the body, insert the recipe Id for the value for `wrangledDataset`, which is the internal platform term for recipe: `{ "wrangledDataset": { "id": 4 } }`
Method	Select the `POST` method.

Verify

1. Run the job for which the webhook was created.
2. When the job successfully completes, open the flow containing the other job to execute.
3. When you select the target recipe, a new job should be queued, in-progress, or completed.

Slack channel message

You can create a webhook task to deliver a text message to a Slack channel of your choice.

Prerequisites

Set up your Slack installation to receive webhook messages:

1. If needed, create a Slack channel to receive your messages.
2. Create an app.
3. Activate incoming webhook messages for your app.
4. Specify the channel to receive your incoming webhook messages.
5. Copy the URL for the incoming webhook from the cURL statement.

Define the flow webhook task

Parameter	Description
Name	This name appears in the Designer Cloud application only.
Url	Paste the URL that you copied from Slack.

Headers	Copy the content headers from the Slack cURL command:
	`key: Content-Type`
	`value: application/json`

Body	`{"text":"Job $jobId has completed. Status: $jobStatus."}`
Method	Select the `POST` method.

Verify

1. Click **Test** to validate that this webhook task will work.
2. Run a job:
    a. Check the Slack channel for a message.
    b. Check the Webhook tab in the Job Details page.

Verify Webhook Signatures

Depending on the target application, implementing Webhook signature verification may require developer skills.

Optionally, you can configure the platform to sign the Webhook requests sent for a flow. Signed requests guarantee that the requests are sent from the platform, instead of a third party.

Below, you can review how the signature is created, so that you can configure the receiving application to properly process the signature and its related request.

Webhook Signature Header

Webhook requests are signed by inserting the `X-Webhook-Signature` header in the request. These signatures are in the following form:

```
X-Webhook-Signature: t=<timestamp>,sha256=<signature>
```

where:

- `<timestamp>` - Timestamp when the signature was sent. Value is in UNIX time.

- `<signature>` - SHA256 signature. The platform generates this signature using a hash-based message authentication code (HMAC) with SHA-256.

More information on these values is available below.

Example:

```
X-Webhook-Signature: t=1568818215724,sha256=55fa71b2e391cd3ccba8413fb51ad16984a38edb3cccfe81f381c4b8197ee07a
```

Check Application Tools

Depending on the application, you may need to complete one of the following sets of tasks to verify the Webhook signatures:

> **NOTE:** You may need to whitelist the platform in your application. See the application's documentation for details.

You may be required to create some custom coding for your application. Below, you can review details on how to do so, including a JavaScript example.

Process Signed Requests

Timestamp

The timestamp value (`t=<timestamp>`) appears at the beginning of the header value to prevent replay attacks, where an attacker could intercept a valid payload and its signature and re-transmit them.

- To avoid such attacks, a timestamp is included in the signature header and is also embedded as part of the signed payload.
- Since the timestamp is part of the signed payload, an attacker cannot change the timestamp value without invalidating the signature.
  - If the signature is valid but the timestamp is too old, you can then choose to reject the request.
  - For example, if you receive a request with a timestamp that corresponds to a date from one hour ago, you should probably reject the request.
- For more information on replay attacks, see *https://en.wikipedia.org/wiki/Replay_attack*.

Signature

The Webhook signature includes as part of its hashed value:

- The secret key (entered above)
- The timestamp value
- Request data:
  - (POST/PUT/PATCH) - the body of the request
  - (GET/DELETE) - URL of the request

Step 1 - Extract the timestamp and signatures

Split the `X-Webhook-Signature` header:

1. Split values using the , character as a separator.
2. Split each of the parts using the = character.
3. Extract the values for the timestamp and signature. From the above example:
   a. timestamp: `1568818215724`
   b. signature: `55fa71b2e391cd3ccba8413fb51ad16984a38edb3cccfe81f381c4b8197ee07a`

Step 2 - Create the expected signature

In the receiving application, you can recompute the signature to verify that the request was sent from the platform.

1. Concatenate the timestamp, the dot character `.` and the request body (POST/PUT/PATCH methods) or the url (GET/DELETE methods).
2. Suppose the above example is the signature for a `POST` request, and the request body is `test`. The concatenated value is the following:

```
1568818215724.test
```

3. You can now compute the HMAC authentication code in your receiving application. In the following JavaScript example, the secret key value is `mySecret`:

```
const crypto = require('crypto');

const message = '1568818215724.test'; // as defined above

const hmac = crypto.createHmac('sha256', 'mySecret');
hmac.update(message)
const expectedSignature = hmac.digest('hex');
```

Step 3 - Compare the signatures

The value returned by your code and the value included as the signature in the `X-Webhook-Signature` header should be compared:

- If the values do not match, reject the request.
- If the values do match, compute the difference between the current timestamp and the timestamp in the header. If the difference is outside of your permitted limit, reject the request.
- Otherwise, process the request normally in your application.

Concepts

This section contains conceptual information on the Designer Cloud powered by Trifacta® platform and its underlying features.

Feature Overviews

These sections provide overviews of key features and capabilities of the Designer Cloud powered by Trifacta® platform .

Tip: Use the links in these sections to access locations in the platform where these features appear.

Overview of Data Export

Contents:

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 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.

- 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*.

F or 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.

> **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 Resources 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 *Job History 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:

This section provides an overview of data import and how different types of import are handled in the Designer Cloud powered by Trifacta® platform .

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 as an imported dataset.

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.

Import of files

The Designer Cloud powered by Trifacta platform 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 platform   identifies file formats based on the extension of the filename.

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 from which 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**: The Designer Cloud powered by Trifacta platform 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 application and transformed, after which results are written. The transformed data is not maintained in the application.

> **NOTE:** Data is not persisted in the Designer Cloud application .

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.

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 pull s 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 shared 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:

The Designer Cloud powered by Trifacta® platform supports different options for reading and writing data from your storage systems.

Base Storage Layer

The base storage layer is the datastore where the Designer Cloud powered by Trifacta platform 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

The following directories and their sub-directories are created and maintained 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. **Avoid using `/trifacta/uploads` for reading and writing data. This directory is used by the Designer Cloud application .**
`/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. For each user, these sub-directories are the default storage location for job results. These locations can be modified. See *Preferences Page*. Within the `queryResults` directory, you may find sub-directories labeled `datasourceCache`. When data source caching is enabled, data read into the product may be temporarily stored in this directory. For more information, see *Configure Data Source Caching*.
`/trifacta/dictionaries`	Storage of custom dictionary files uploaded by users. **NOTE:** This feature applies to Designer Cloud Powered by Trifacta Enterprise Edition only. It is not often used.
`/trifacta/tempfiles`	Temporary storage location for files required for use of the product. **NOTE:** The `tempfiles` directory is reserved for use by the platform. It is the only directory of these that is actively cleaned by the platform.

User-specific directories

The following directories are created by default on the base storage layer for the Designer Cloud powered by Trifacta platform to store user data.

By default, these directories are stored in the following:

```
<bucket_name>/<userId>
```

where:

- `<bucket_name>` is the name of the bucket where user data is stored.
- `userId>` is the username that is used to log in to the product.

Directory	Description
`jobrun`	Storage of generated samples.
`temp`	Temporary storage
`upload`	Depending on your configuration, uploaded files may be stored in this per-user directory.

These directories may be modified by individual users. For more information, see *Storage Config Page*.

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

The Designer Cloud powered by Trifacta platform 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* .

> **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.

Encryption on base storage layer

For data that is transferred to and from the base storage layer:

- Data in transit is encrypted using HTTPS.
- Data at rest is unencrypted by default.

> **NOTE:** Server-side encryption can be applied when the product is writing results to an S3 bucket. For more information, see *AWS Settings Page*.

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 the Designer Cloud powered by Trifacta platform 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 the Designer Cloud powered by Trifacta platform 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.

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 the Designer Cloud powered by Trifacta platform . 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 Parameterization

Contents:

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 of multiple types 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

- Environment parameter names are unique within the environment.
- You cannot use environment parameters in recipes.
- You cannot use environment parameters in plans.
- You cannot use environment parameters in schedule triggers.

- 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.

Example - parameterized database names

In the previous example, bucket names differed between the development and production workspaces. Similarly, there may be different values for the database to use between development and production.

Tip: If you are creating datasets using custom SQL, this method can be used for migrating flows between workspaces. If the flows reference imported datasets that use custom SQL, you only need to update the value of the environment parameter in the target workspace and reset the connection identifier in order to reconnect the imported flow to its data sources.

Environment Name	Database Name
Dev	`myco-DB-dev`
Test	`myco-DB-test`
Prod	`myco-DB-prod`

In your Dev workspace, you can create the environment parameter as follows:

```
env.db-source
```

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.

NOTE: When importing one or more Excel files as a parameterized dataset, you select worksheets to include from the first file. If there are worksheets in other Excel files that match the names of the worksheets that you selected, those worksheets are also imported. All worksheets are unioned together into a single imported dataset with parameters. Pattern-based parameters are not supported for import of Excel worksheets.

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.

- For datasources that require conversion, such as Excel, PDF, or JSON files, you can create a dataset with parameters from a maximum of 500 converted files.
- 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`, `bxb`, 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` patter n. 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:** For best results when parameterizing directories in your file path, include the trailing slash (`/`) as part of your parameterized value.

Options:

- You can choose to search nested folders for files that match your specified pattern.

> **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 been 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 permitted 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:

- **Variable type** flow parameters:
    - 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.

      > **NOTE:** Wildcards are not supported.

    - Patterns . For more information, see *Text Matching*.
    - Regular expressions.
- **Selector type** flow parameters:
    - These parameters are specified as a list of permitted values.
    - The first value is the default value.
    - The parameter values are treated as String values.

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 - variable type

Suppose you need to provide variable date ranges to your transformations. For example, you may need to be able to transform the data for the preceding 30, 60, 90, or custom days.

You can create a Variable parameter called:

```
$paramDaysBack
```

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 DaysBack value for the job run.
- Since all values are treated as String values, you must interpret the numeric values within your transformation steps as Integer data types.

For more examples, see *Create Flow Parameter*.

Example - selector type

Suppose you need to process your flow across several regions of your country. These regions are identified using a region ID value.

From the Flow View context menu, you select **Manage parameters**. In the Parameters tab, you specify **Selector** f or the Type.

Specify the parameter name:

```
paramRegion
```

You can then add Choice values for the permitted regions:

```
WEST
MOUNTAIN
CENTRAL
EAST
OTHER
```

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.

You can test for the parameter by using the following conditional in your transformation step and selecting from one of the listed values:

```
$paramRegion == <selected_value>
```

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

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	When you run a job, you can apply override values to variables for your imported datasets. See *Run Job Page*.

parameters	
flow parameters	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. **NOTE:** 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. When flows are referenced in a plan task, their parameters and default values are available for override.
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 *Run Job Page*.
environment parameters	Environment parameters cannot be overridden.
Triggers	In flow and plan triggers, you can apply overrides to parameter values that are inherited from the object being triggered. **NOTE:** In a trigger, displayed parameter values may be inherited from the plan or flow that is being triggered. To ensure that the proper value is used, you should set a specific value for the override in the trigger. This is a known issue.

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. **Runtime or trigger-based overrides:** Parameter values specified at runtime for ad-hoc or scheduled 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 Task - Run Job*.

   a. For more information on runtime overrides, see *Run Job Page*.
   b. For more information for trigger-based overrides for flows, see *Add Schedule Dialog*.
   c. For more information on trigger-based overrides for plans, see *Plan View for Triggers*.

2. **Flow- or plan-level overrides:** At the flow or plan 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 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*.

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*.

Parameters in Job Details

In the Job Details page:

- **Data sources tab:** For file-based parameterized datasets, you can review the files that were matched at runtime for the specified parameters.
- **Parameters tab:** View the parameter names and values that were used as part of the job, including the list of matching datasets.

See *Job Details Page*.

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 Task - 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 Task - 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 Predictive Transformation

Contents:

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.

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.

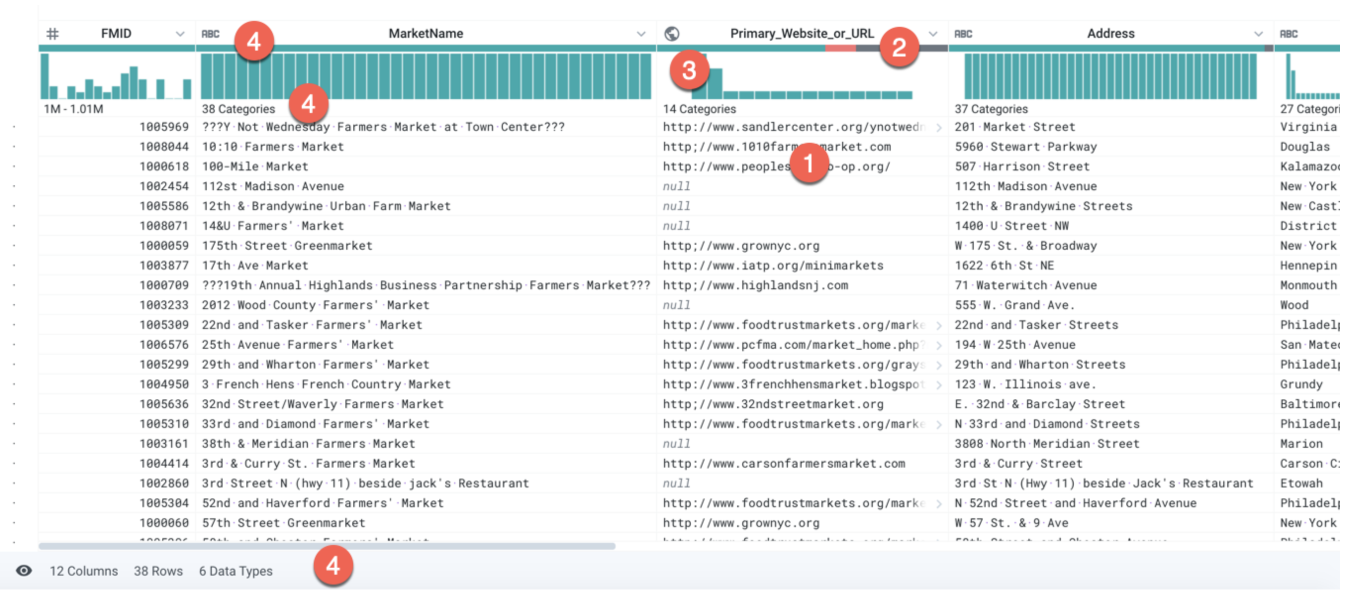
Phases

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	ABC
1M - 1.01M	38 Categories	14 Categories	37 Categories	27 Categori
1005969	???Y Not Wednesday Farmers Market at Town Center???	http://www.sandlercenter.org/ynotwedn	201 Market Street	Virginia
1008044	10:10 Farmers Market	http;//www.1010farm[illegible]market.com	5960 Stewart Parkway	Douglas
1000618	100-Mile Market	http://www.peoples[illegible]o-op.org/	507 Harrison Street	Kalamazo
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/marke	22nd and Tasker Streets	Philadel
1006576	25th Avenue Farmers' Market	http://www.pcfma.com/market_home.php?	194 W 25th Avenue	San Mate
1005299	29th and Wharton Farmers' Market	http://www.foodtrustmarkets.org/grays	29th and Wharton Streets	Philadel
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	Baltimor
1005310	33rd and Diamond Farmers' Market	http://www.foodtrustmarkets.org/marke	N 33rd and Diamond Streets	Philadel
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/marke	N 52nd Street and Haverford Avenue	Philadel
1000060	57th Street Greenmarket	http://www.grownyc.org	W 57 St. & 9 Ave	New York

12 Columns 38 Rows 6 Data Types

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 selects 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.

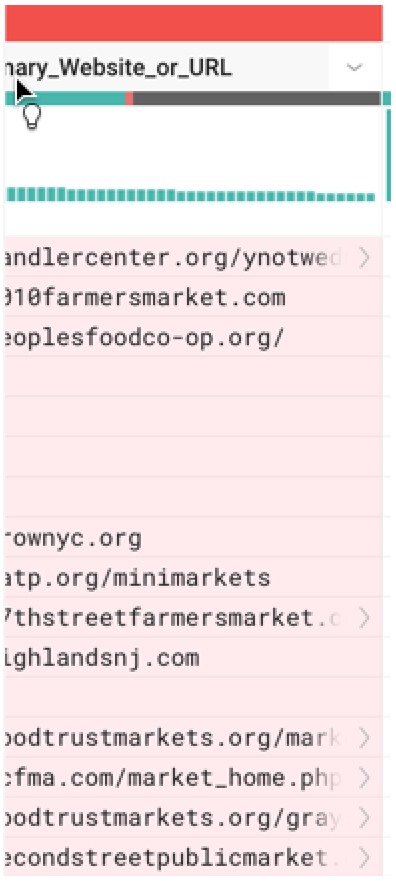


Figure: Selection cursor changes on hover of selectable items

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

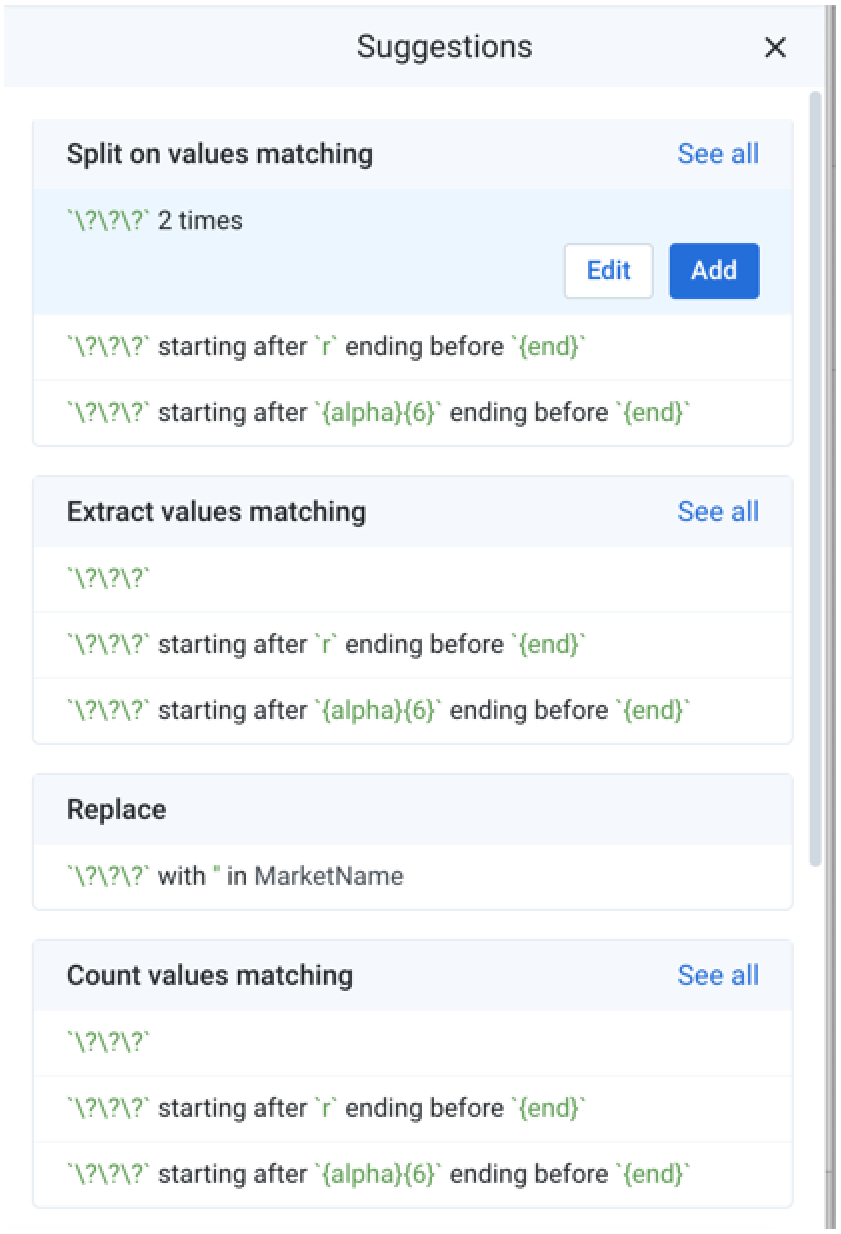
Based on the set of selections, an inference algorithm attempts to interpret the data transformation intent of the selection and generates a ranked set of suggestions and patterns for the selections to match. For example, if you select the first three characters in a cell, the algorithm may produce two 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.



Suggestions

Split on values matching — See all

`\?\?\?` 2 times

Edit Add

`\?\?\?` starting after `r` ending before `{end}`

`\?\?\?` starting after `{alpha}{6}` ending before `{end}`

Extract values matching — See all

`\?\?\?`

`\?\?\?` starting after `r` ending before `{end}`

`\?\?\?` starting after `{alpha}{6}` ending before `{end}`

Replace

`\?\?\?` with '' in MarketName

Count values matching — See all

`\?\?\?`

`\?\?\?` starting after `r` ending before `{end}`

`\?\?\?` starting after `{alpha}{6}` ending before `{end}`

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*.

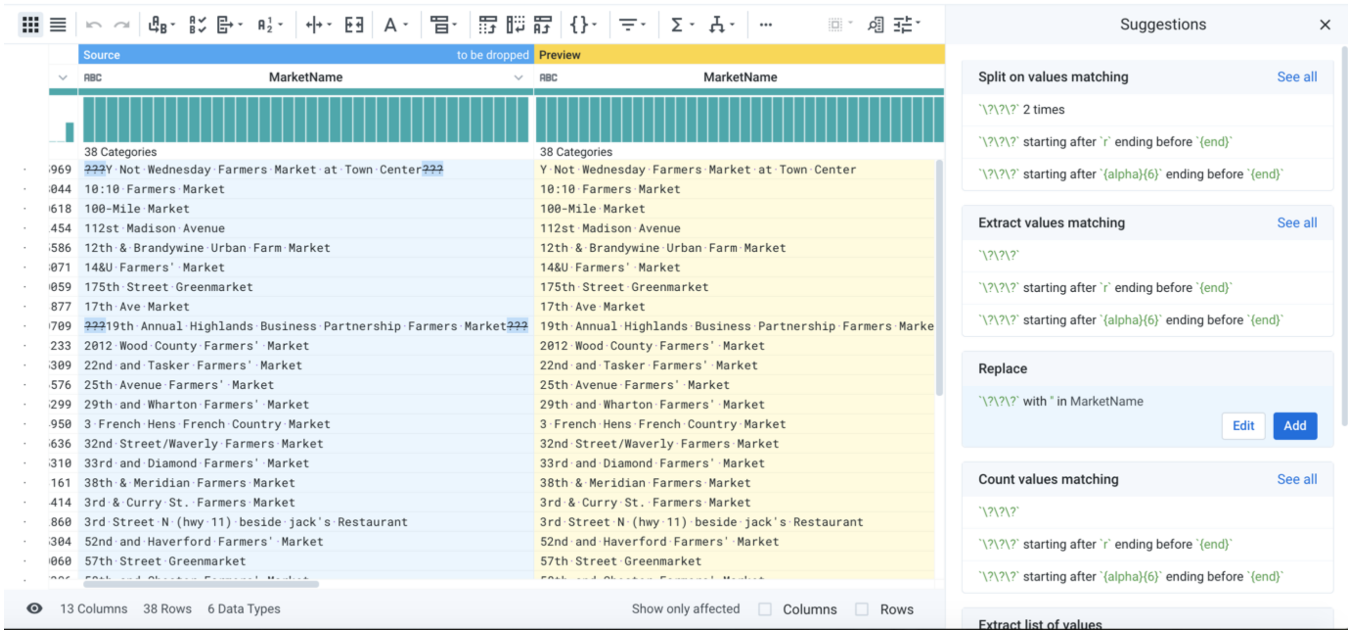
Workspace administrators can choose to enable this feature and can configure whether data is aggregated from individual workspace users' transformations or from all workspace users' transformations. See *Workspace Settings Page*.

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.



	Source — to be dropped ABC MarketName 38 Categories	Preview ABC MarketName 38 Categories
969	~~???~~Y Not Wednesday Farmers Market at Town Center~~???~~	Y Not Wednesday Farmers Market at Town Center
044	10:10 Farmers Market	10:10 Farmers Market
618	100-Mile Market	100-Mile Market
454	112st Madison Avenue	112st Madison Avenue
586	12th & Brandywine Urban Farm Market	12th & Brandywine Urban Farm Market
071	14&U Farmers' Market	14&U Farmers' Market
059	175th Street Greenmarket	175th Street Greenmarket
877	17th Ave Market	17th Ave Market
709	~~???~~19th Annual Highlands Business Partnership Farmers Market~~???~~	19th Annual Highlands Business Partnership Farmers Marke
233	2012 Wood County Farmers' Market	2012 Wood County Farmers' Market
309	22nd and Tasker Farmers' Market	22nd and Tasker Farmers' Market
576	25th Avenue Farmers' Market	25th Avenue Farmers' Market
299	29th and Wharton Farmers' Market	29th and Wharton Farmers' Market
950	3 French Hens French Country Market	3 French Hens French Country Market
636	32nd Street/Waverly Farmers Market	32nd Street/Waverly Farmers Market
310	33rd and Diamond Farmers' Market	33rd and Diamond Farmers' Market
161	38th & Meridian Farmers Market	38th & Meridian Farmers Market
414	3rd & Curry St. Farmers Market	3rd & Curry St. Farmers Market
860	3rd Street N (hwy 11) beside jack's Restaurant	3rd Street N (hwy 11) beside jack's Restaurant
304	52nd and Haverford Farmers' Market	52nd and Haverford Farmers' Market
060	57th Street Greenmarket	57th Street Greenmarket

13 Columns 38 Rows 6 Data Types — Show only affected ☐ Columns ☐ Rows

Suggestions

Split on values matching — See all
- `\?\?\?` 2 times
- `\?\?\?` starting after `r` ending before `{end}`
- `\?\?\?` starting after `{alpha}{6}` ending before `{end}`

Extract values matching — See all
- `\?\?\?`
- `\?\?\?` starting after `r` ending before `{end}`
- `\?\?\?` starting after `{alpha}{6}` ending before `{end}`

Replace
- `\?\?\?` with '' in MarketName — Edit Add

Count values matching — See all
- `\?\?\?`
- `\?\?\?` starting after `r` ending before `{end}`
- `\?\?\?` starting after `{alpha}{6}` ending before `{end}`

Extract list of values

Figure: Previewed effects of transform

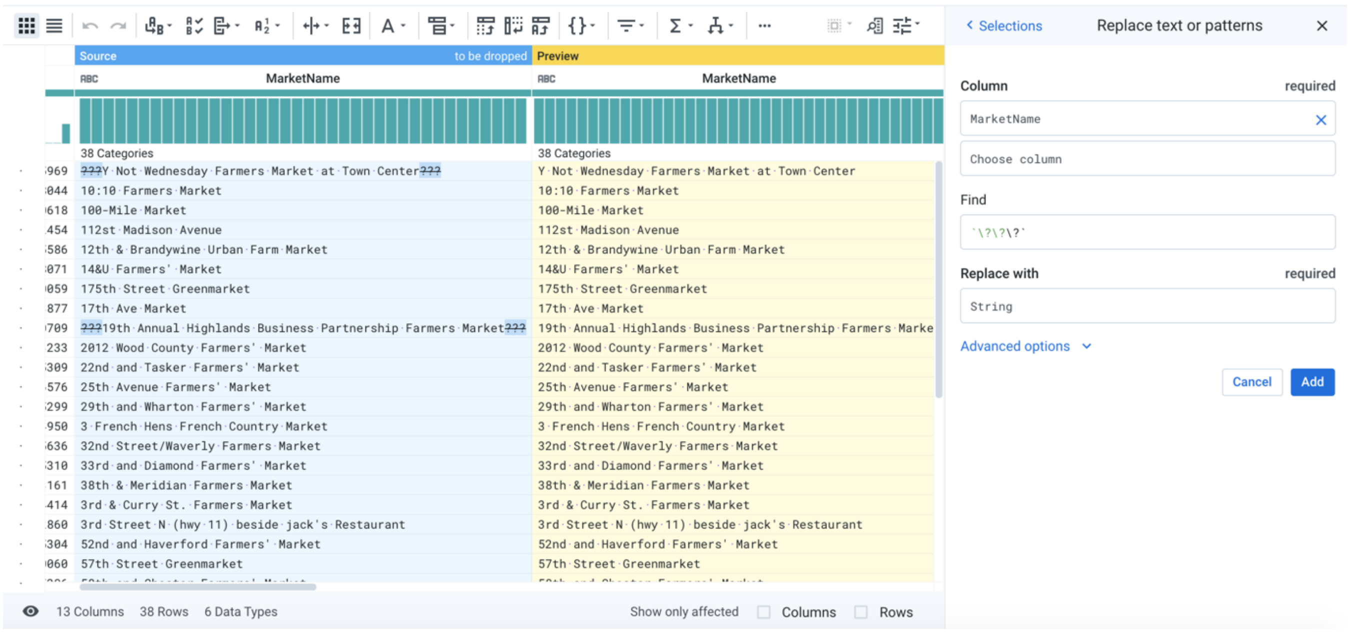
When the transform is added to the recipe, the transform is rendered into the data transformation language and applied in real-time to the dataset, so that you can immediately begin working on the next step of the process.

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.



	Source — to be dropped: MarketName (38 Categories)	Preview: MarketName (38 Categories)
969	~~???~~Y Not Wednesday Farmers Market at Town Center~~???~~	Y Not Wednesday Farmers Market at Town Center
044	10:10 Farmers Market	10:10 Farmers Market
618	100-Mile Market	100-Mile Market
454	112st Madison Avenue	112st Madison Avenue
586	12th & Brandywine Urban Farm Market	12th & Brandywine Urban Farm Market
071	14&U Farmers' Market	14&U Farmers' Market
059	175th Street Greenmarket	175th Street Greenmarket
877	17th Ave Market	17th Ave Market
709	~~???~~19th Annual Highlands Business Partnership Farmers Market~~???~~	19th Annual Highlands Business Partnership Farmers Marke
233	2012 Wood County Farmers' Market	2012 Wood County Farmers' Market
309	22nd and Tasker Farmers' Market	22nd and Tasker Farmers' Market
576	25th Avenue Farmers' Market	25th Avenue Farmers' Market
299	29th and Wharton Farmers' Market	29th and Wharton Farmers' Market
950	3 French Hens French Country Market	3 French Hens French Country Market
636	32nd Street/Waverly Farmers Market	32nd Street/Waverly Farmers Market
310	33rd and Diamond Farmers' Market	33rd and Diamond Farmers' Market
161	38th & Meridian Farmers Market	38th & Meridian Farmers Market
414	3rd & Curry St. Farmers Market	3rd & Curry St. Farmers Market
860	3rd Street N (hwy 11) beside jack's Restaurant	3rd Street N (hwy 11) beside jack's Restaurant
304	52nd and Haverford Farmers' Market	52nd and Haverford Farmers' Market
060	57th Street Greenmarket	57th Street Greenmarket

13 Columns 38 Rows 6 Data Types — Show only affected ☐ Columns ☐ Rows

Selections — Replace text or patterns

Column (required): MarketName

Choose column

Find: `` `\?\?\?` ``

Replace with (required): String

Advanced options

Cancel Add

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:

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
String Data Type	Any non-null value can be typed as String. A String can be anything.
Integer Data Type	The Integer data type applies to positive and negative numeric values that have no decimal point.
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`.
Boolean Data Type	The Boolean data type expresses true or false values.
Datetime Data Type	Designer Cloud Powered by Trifacta® Enterprise Edition supports a variety of Datetime formats, each of which has additional variations to it.
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.
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.

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
Social Security Number Data Type	This data type is applied to numeric data following the pattern for United States Social Security numbers.
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.
	This data type matches text values that are properly formatted email addresses.

Email Address Data Type	
Credit Card Data Type	Credit card numbers are numeric data that follow the 14-digit or 16-digit patterns for credit cards.
Gender Data Type	This data type matches a variety of text patterns for expressing male/female distinctions.
Zip Code Data Type	This data type matches five- and nine-digit U.S. zipcode 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.
IP Address Data Type	The IP Address data type supports IPv4 address.
URL Data Type	URL data type is applied to data that follows generalized patterns of URLs.
HTTP Code Data Type	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
String Data Type	logical	String	The default data type. Any non-empty/non-value is valid for String data type.
Integer Data Type	logical	Int	Use *NUMFORMAT Function* to format these values. Underlying values are not modified.
Decimal Data Type	logical	Float	Use *NUMFORMAT Function* to format these values. Underlying values are not modified.
Boolean Data Type	logical	Bool	
Datetime Data Type	logical	Datetime	Use *DATEFORMAT Function* to format these values. Underlying values are not modified.
Object Data Type	logical	Map/Object	
Array Data Type	logical	Array	
Social Security Number Data Type	complex	String	String data type constrained by a regular expression.
Phone Number Data Type	complex	String	String data type constrained by a regular expression.
Email Address Data Type	complex	String	String data type constrained by a regular expression.
Credit Card Data Type	complex	String	String data type constrained by a regular expression.
Gender Data Type	complex	String	String data type constrained by a regular expression.
Zip Code Data Type	complex	String	String data type constrained by a regular expression.
State Data Type	complex	String	String data type constrained by a regular expression.
IP Address Data Type	complex	String	String data type constrained by a regular expression.

URL Data Type	complex	String	String data type constrained by a regular expression.
HTTP Code Data Type	complex	String	String data type constrained by a regular expression.

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 s 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 **Edi t 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.

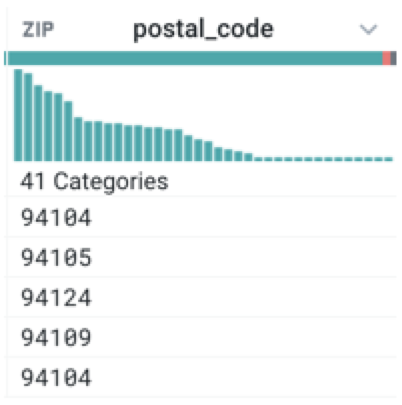


Figure: Column header example

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.

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
NUMFORMAT Function	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.
DATEFORMAT Function	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.
UNIXTIMEFORMAT Function	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
VALID Function	Tests whether a set of values is valid for a specified data type and is not a null value.
ISMISMATCHED Function	Tests whether a set of values is not valid for a specified data type.
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.
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.

Parsing functions

These functions can be used to parse String values against a specific data type.

Function	Description
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.
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.
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.
PARSEDATE Function	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.
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.

Managing null and empty values

These functions allow you to generate or test for missing or null values.

Function	Description
ISNULL Function	The `ISNULL` function tests whether a column of values contains null values. For input column references, this function returns `true` or `false`.
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.
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`.
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.

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	This section covers data type conversions between the Designer Cloud® application and the Avro file format.
DB2 Data Type Conversions	This section covers data type conversions between the Designer Cloud® application and DB2 databases.
Hive Data Type Conversions	This section covers data type conversions between the Designer Cloud® application and Hive.
Oracle Data Type Conversions	This section covers data type conversions between the Designer Cloud® application and Oracle databases.
MySQL Data Type Conversions	This section covers data type conversions between the Designer Cloud® application and MySQL databases.
Parquet Data Type Conversions	This section covers data type conversions between the Designer Cloud® application and the Parquet file format.
Postgres Data Type Conversions	This section covers data type conversions between the Designer Cloud® application and PostgreSQL databases.

Redshift Data Type Conversions	This section covers data type conversions between the Designer Cloud® application and Redshift.
Snowflake Data Type Conversions	This section covers data type conversions between the Designer Cloud® application and Snowflake databases.
AWS Glue Data Type Conversions	This section covers data type conversions between the Designer Cloud® application and AWS Glue.
Salesforce Data Type Conversions	This section covers data type conversions between the Designer Cloud® application and Salesforce.
SQL Server Data Type Conversions	This section covers data type conversions between the Designer Cloud® application and SQL Server databases.
SQL DW Data Type Conversions	This section covers data type conversions between the Designer Cloud® application and SQL DW datastores.
Databricks Tables Data Type Conversions	This section covers data type conversions between the Designer Cloud® application and Databricks Tables.
Tableau Hyper Data Type Conversions	This section covers data type conversions between the Designer Cloud® application and Tableau Hyper format.
Teradata Data Type Conversions	This section covers data type conversions between the Designer Cloud® application and Teradata databases.
SharePoint Data Type Conversions	This section covers data type conversions between the Designer Cloud® application and SharePoint.

For more information, see *Type Conversions*.

Overview of Schema Management

Contents:

A **schema** refers to the sequence and data type of columns in a dataset. Schemas are applicable to relational tables and some file formats. This section provides an overview of how Designer Cloud Powered by Trifacta® Enterprise Edition enables the capture and tracking of changes of input schemas as well as the methods available for transforming your data to match a target schema.

Overview of Schemas

A schema is a skeleton structure that represents the logical view of the dataset. The dataset can be a file, table, or a SQL query in a database. A schema defines how the data is structured and organized. Schema information includes:

- Column names
- Column ordering
- Column data types

Schemas may apply to relational tables and schematized file formats such as Avro and Parquet.

Input type conversions

Depending on the data source, Designer Cloud Powered by Trifacta® Enterprise Edition can read in native data types into Trifacta data types. For more information, see *Type Conversions*.

Schema Validation

Feature Availability: This feature may not be available in all product editions. For more information on available features, see *Compare Editions*.

Over time, schema sources may change in major and minor ways, often without warning. From within the Designe r Cloud application , schema changes may appear as broken recipe steps and can cause data corruption downstream. Schema validation can be applied to:

- relational datasets (tables and views)
- schematized files (e.g. Parquet)

- file-based datasets (e.g. CSV files)

To assist with these issues, the Designer Cloud application can be configured to monitor schema changes on your dataset. Schema validation performs the following actions on your dataset:

- On read, the schema information from the dataset is captured and stored separately in the Trifacta database. This information identifies column names, data types, and ordering of the dataset.
- When the dataset is read during job execution, the new schema information is read and compared to the stored version, which enables identification of changes to the dataset.

  > **Tip:** This check occurs as the first step of the job execution process and is labeled as **Schema validation**.

- You can configure the Designer Cloud application to halt job execution when schema validation issues have been encountered.

  > **Tip:** Configuration settings can be overridden for individual jobs.

Limitations

> **NOTE:** If you attempt to refresh the schema of a parameterized dataset based on a set of files, only the schema for the first file is checked for changes. If changes are detected, the other files are assumed to contain those changes as well. This can lead to changes being assumed or undetected in later files and potential data corruption in the flow.

Enable

Schema management service

If you are not enabling schema validation, the Schema Management service can be disabled.

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 `false`:

```
"schema-management-service.enabled": true,
```

3. Save your changes and restart the platform.

Settings

At the project or workspace level, an administrator can set the default settings for outputs to validate schemas or not.

> **Tip:** Workspace-level defaults can be overridden at the job level, even if the workspace-level settings are disabled. For more information, see *Run Job Page*.

File settings

During the creation of an imported dataset, you can configure the following settings for schema validation:

Steps:

1. After a file has been selected in the Import Data page, click **Edit settings**.
2. In the Edit settings dialog:

Setting	Effects on schema validation
Detect structure	When enabled, the structure of the first chunk from the imported dataset is used for determining the schema of the dataset. **NOTE:** If the imported dataset is composed of multiple files, only the first file is used for schema validation purposes. If there are changes to the schema of the second or later files, they are undetected. When disabled, the structure of the file is ignored, and all data is imported as a single column. Schema validation is effectively disabled for the dataset.
Infer header	The first row of data is used as the column headers.
No headers	Default column names are used in the stored schema: `column1` , `column2`, and so on.

For more information, see *File Import Settings*.

Use

When a job is launched, the schema validation check is performed in parallel with the data ingestion step. Schema validation checks for:

1. Changes to the order of columns
2. Columns that have been deleted
3. Columns that have been added

The results of the schema validation check are reported in the Job Details page in the Schema validation stage.

NOTE: Jobs may be configured to fail if schema validation checks fail. If jobs are not configured to fail, jobs may complete with warnings and publish output data to the specified targets, when schema validation fails.

For more information, see *Job Details Page*.

When schema validation detects differences in the Job Details page, those findings can be explored in detail. See *Schema Changes Dialog*.

Job-level overrides

You can override the project or workspace level settings for schema validation for individual jobs. For more information, see *Run Job Page*.

Job results

In the Job Details page, you can review schema validation checks for the datasets in the job. For more information, see *Job Details Page*.

Schema Refresh

Feature Availability: This feature may not be available in all product editions. For more information on available features, see *Compare Editions*.

Schema refresh enables on-demand updating of your imported dataset schemas to capture changes to columns. For example, when you are working with datasets in a flow view, you can refresh your imported datasets' schemas by checking the source schema for changes. Schema refresh automatically generates a new initial sample, which allows you to gather fresh data in the Transformer page.

Schema refresh applies to:

- Relational schemas
- Schematized files
- Delimited files

  > **NOTE:** Delimited files include CSVs and TSVs and can include other files whose delimiters can be inferred by the Designer Cloud application during import. Delimited files do not contain data type information; data types are inferred by the Designer Cloud application for these file types.

- Converted file formats:
  - JSON files that are converted during ingest. For more information, see *Working with JSON v2*.
  - PDF. See *Import PDF Data*.
  - Excel. See *Import Excel Data*.
  - Google Sheets. See *Import Google Sheets Data*.
  - For multi-sheet sources:
    - If each sheet is converted into a separate dataset, the schema of a dataset is refreshed from the source sheet.
    - If multiple sheets are combined into a single dataset, the schema of a dataset is refreshed from the first sheet in the source.

Key Benefits:

- Reduces the number of duplicate or invalid datasets created from the same source.
- Reduces challenges of replacing datasets and retaking samples.

Limitations

- If a column's data type is modified and other changes, such as column name changes, are not detected, this change is not considered a schema drift error.
- You cannot refresh the schemas of reference datasets or uploaded sources.
- Schema refresh does not apply to any file formats that require conversion to native formats.

  > **NOTE:** Schema management does not work for JSON-based imported datasets that were created under the v1 legacy method of JSON import. All JSON imported datasets created under the legacy method (v1) of JSON import must be recreated to behave like v2 datasets with respect to conversion and schema management. Features developed in the future may not retroactively be supported in the v1 legacy mode. For more information, see *Working with JSON v2*.

> **NOTE:** If you have imported a flow from an earlier version of the application, you may receive warnings of schema drift during job execution when there have been no changes to the underlying schema. This is a known issue. The workaround is to create a new version of the underlying imported dataset and use it in the imported flow.

Limitations for parameterized datasets

Parameterized files:

> **NOTE:** If you attempt to refresh the schema of a parameterized dataset based on a set of files, only the schema for the first file is checked for changes. If changes are detected, the other files are assumed to

contain those changes as well. This can lead to changes being assumed or undetected in later files and potential data corruption in the flow.

Parameterized tables:

NOTE: Refreshing the schema of a parameterized dataset using custom SQL is not supported.

Effects of refreshing schemas

When you choose to refresh a schema, the schema is refreshed without checking for changes, which forces the deletion of all samples and recollection of a new initial sample. All pre-existing samples must be recreated. In some environments, this sample collection incurs costs.

When you refresh the schema in the Designer Cloud application :

- The source schema is applied to the imported dataset in all cases.
    - All the existing samples are invalidated.
    - A new initial sample is generated, which updates the previewed data. This may take some time.
- Addition or removal of columns may cause recipe steps to break, which can cause any transformation jobs to fail. You must fix these broken steps in the Recipe panel.

    Tip: For some data-dependent recipe steps, such as joins and pivots, that are listed as broken, you may be able to edit the step and immediately save it to repair the step.

Refresh your schemas

For more information on how to refresh the schemas of your datasets, see:

- *Library Page*
- *Dataset Details Page*
- *View for Imported Datasets*

Via API:

For more information, see *https://api.trifacta.com/ee/es.t/index.html#operation/asyncRefreshSchema*

Output Schemas

Output type conversions

Depending on the output system, Designer Cloud Powered by Trifacta Enterprise Edition can deliver your results in columns and data types native to the target. For more information, see *Type Conversions*.

Target schemas

As needed, you can import a dataset the columns of which can serve as the target of your transformation efforts. When this target schema is imported, it is super-imposed on the columns of your dataset in the Transformer page, allowing you to quickly change the naming, order, and data typing of your columns to match the target schema. For more information, see *Overview of Target Schema Mapping*.

Overview of Standardization

Contents:

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. This section summarizes the available methods of standardization, as well as their recommended uses.

Standardization Methods

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.

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.	• Standardize values to correct spelling differences, capitalization, whitespace, and other errors. • Values must be consistent across rows of the column. • Primarily used for string-based data types.	Available through the *Standardize Page*
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.	• Standardize values to follow a consistent format, such as phone numbers or social security numbers. • Data type follows a somewhat consistent format and needs reshaping.	Available in the Patterns tab in *Column Details Panel*
By function	You can apply one or more specific functions to cleanse your data of minor errors in formatting or structure.	• Good method for improving the performance of pattern- or algorithm-based matching. • Some functions are specific to a data type, while others have more general application.	Edit column with formula in the *Transform Builder* .
Mix-and-match	You can use combinations of the above methods for more complex use cases.	• Combine function-based standardization for global changes to	

		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.

	UNICODE Function	Generates the Unicode index value for the first character of the input string.
Case Conversion	*UPPER Function*	All alphabetical characters in the input value are converted to uppercase in the output value.
	LOWER Function	All alphabetical characters in the input value are converted to lowercase in the output value.
	PROPER Function	Converts an input string to propercase. Input can be a column reference or a string literal.
Cleanse Functions	*TRIM Function*	Removes leading and trailing whitespace from a string. Spacing between words is not removed.
	TRIMQUOTES Function	Removes leading and trailing quotes or double-quotes from a string. Quote marks in the middle of the string are not removed.
	REMOVEWHITESPACE Function	Removes all whitespace from a string, including leading and trailing whitespace and all whitespace within the string.
	REMOVESYMBOLS Function	Removes all characters from a string that are not letters, numbers, accented Latin characters, or whitespace.
String Sizing Functions	*LEFT Function*	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.
	RIGHT Function	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.
	PAD Function	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.
String Comparison Functions		See *Compare Strings*.

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 cleanup.

Transformation Name	`Edit column with formula`
Parameter: Columns	`*`
Parameter: Formula	`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
ABS Function	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.
ROUND Function	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.
TRUNC Function	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.
NUMFORMAT Function	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 confirm. 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.

Transformation Name	`Edit column with formula`
Parameter: Columns	`*`
Parameter: Formula	`IF(FIND($col, '.')>0, IFVALID($col, 'Float',NUMFORMAT(ROUND($col, 2), '0.00')),$col)`

- 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
DATEFORMAT Function	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 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:

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	Åpple
Row06	pare

When you standardize using a spelling-based algorithm, the following values are clustered:

Source Value	New Value
Apple	
apple	
Åpple	
	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
Åpple	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
Åpple	apple

You can apply cluster-based standardization through the 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:

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*.

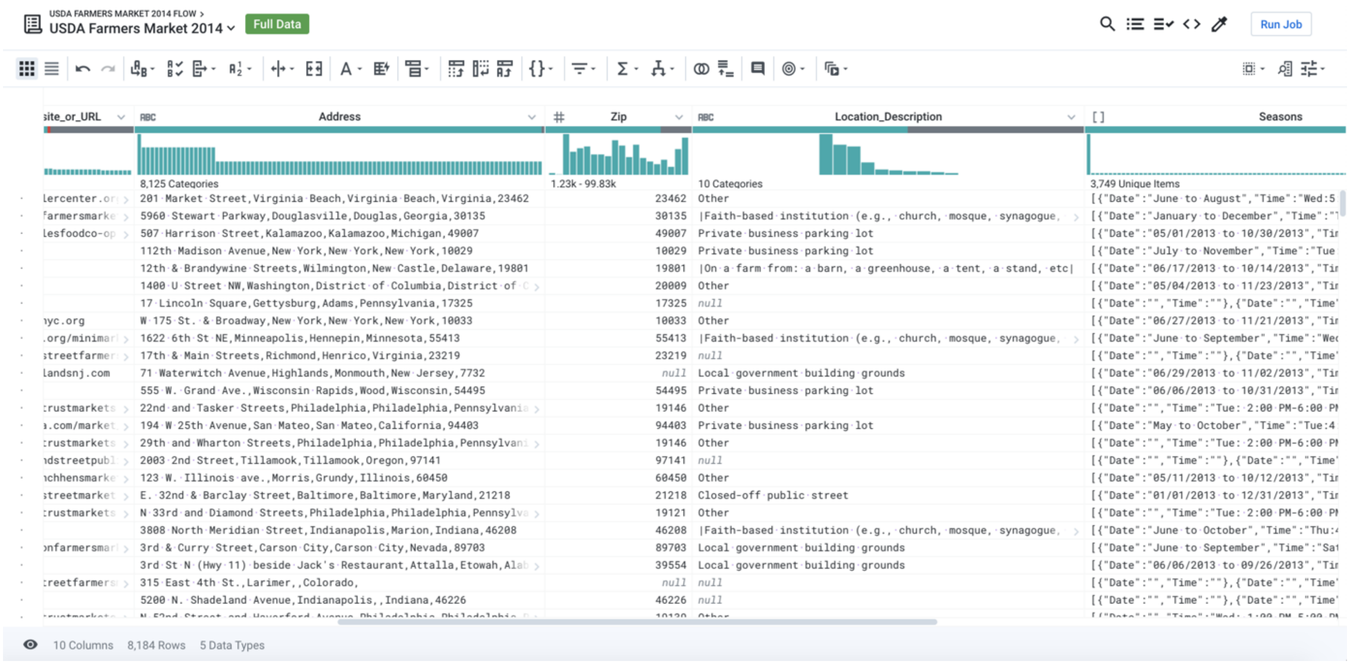
Visual profiles are available while you transform your data in the Transformer page, when you dig into the detail of individual columns, and after you execute your job at scale. Each of these interfaces has different usage patterns designed to accelerate and simplify data transformation for that specific 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:


USDA FARMERS MARKET 2014 FLOW >
USDA Farmers Market 2014
Full Data
Run Job

site_or_URL	Address	Zip	Location_Description	Seasons
	8,125 Categories	1.23k - 99.83k	10 Categories	3,749 Unique Items
lercenter.or	201 Market Street,Virginia Beach,Virginia Beach,Virginia,23462	23462	Other	[{"Date":"June to August","Time":"Wed:5
farmersmarke	5960 Stewart Parkway,Douglasville,Douglas,Georgia,30135	30135	|Faith-based institution (e.g., church, mosque, synagogue,	[{"Date":"January to December","Time":"T
lesfoodco-op	507 Harrison Street,Kalamazoo,Kalamazoo,Michigan,49007	49007	Private business parking lot	[{"Date":"05/01/2013 to 10/30/2013","Tin
	112th Madison Avenue,New York,New York,New York,10029	10029	Private business parking lot	[{"Date":"July to November","Time":"Tue
	12th & Brandywine Streets,Wilmington,New Castle,Delaware,19801	19801	|On a farm from: a barn, a greenhouse, a tent, a stand, etc|	[{"Date":"06/17/2013 to 10/14/2013","Tin
	1400 U Street NW,Washington,District of Columbia,District of C	20009	Other	[{"Date":"05/04/2013 to 11/23/2013","Tin
	17 Lincoln Square,Gettysburg,Adams,Pennsylvania,17325	17325	*null*	[{"Date":"","Time":""},{"Date":"","Time"
nyc.org	W 175 St. & Broadway,New York,New York,New York,10033	10033	Other	[{"Date":"06/27/2013 to 11/21/2013","Tin
.org/minimar	1622 6th St NE,Minneapolis,Hennepin,Minnesota,55413	55413	|Faith-based institution (e.g., church, mosque, synagogue,	[{"Date":"June to September","Time":"Wed
streetfarmer	17th & Main Streets,Richmond,Henrico,Virginia,23219	23219	*null*	[{"Date":"","Time":""},{"Date":"","Time"
landsnj.com	71 Waterwitch Avenue,Highlands,Monmouth,New Jersey,7732	*null*	Local government building grounds	[{"Date":"06/29/2013 to 11/02/2013","Tin
	555 W. Grand Ave.,Wisconsin Rapids,Wood,Wisconsin,54495	54495	Private business parking lot	[{"Date":"06/06/2013 to 10/31/2013","Tin
trustmarkets	22nd and Tasker Streets,Philadelphia,Philadelphia,Pennsylvania	19146	Other	[{"Date":"","Time":"Tue: 2:00 PM-6:00 PM
a.com/market	194 W 25th Avenue,San Mateo,San Mateo,California,94403	94403	Private business parking lot	[{"Date":"May to October","Time":"Tue:4
trustmarkets	29th and Wharton Streets,Philadelphia,Philadelphia,Pennsylvani	19146	Other	[{"Date":"","Time":"Tue: 2:00 PM-6:00 PM
ndstreetpubl	2003 2nd Street,Tillamook,Tillamook,Oregon,97141	97141	*null*	[{"Date":"","Time":""},{"Date":"","Time"
nchhensmarke	123 W. Illinois ave.,Morris,Grundy,Illinois,60450	60450	Other	[{"Date":"05/11/2013 to 10/12/2013","Tin
streetmarket	E. 32nd & Barclay Street,Baltimore,Baltimore,Maryland,21218	21218	Closed-off public street	[{"Date":"01/01/2013 to 12/31/2013","Tin
trustmarkets	N 33rd and Diamond Streets,Philadelphia,Philadelphia,Pennsylva	19121	Other	[{"Date":"","Time":"Tue: 2:00 PM-6:00 PM
	3808 North Meridian Street,Indianapolis,Marion,Indiana,46208	46208	|Faith-based institution (e.g., church, mosque, synagogue,	[{"Date":"June to October","Time":"Thu:4
onfarmersmar	3rd & Curry Street,Carson City,Carson City,Nevada,89703	89703	Local government building grounds	[{"Date":"June to September","Time":"Sat
	3rd St N (Hwy 11) beside Jack's Restaurant,Attalla,Etowah,Alab	39554	Local government building grounds	[{"Date":"06/06/2013 to 09/26/2013","Tin
treetfarmers	315 East 4th St.,Larimer,,Colorado,	*null*	*null*	[{"Date":"","Time":""},{"Date":"","Time"
	5200 N. Shadeland Avenue,Indianapolis,,Indiana,46226	46226	*null*	[{"Date":"","Time":""},{"Date":"","Time"

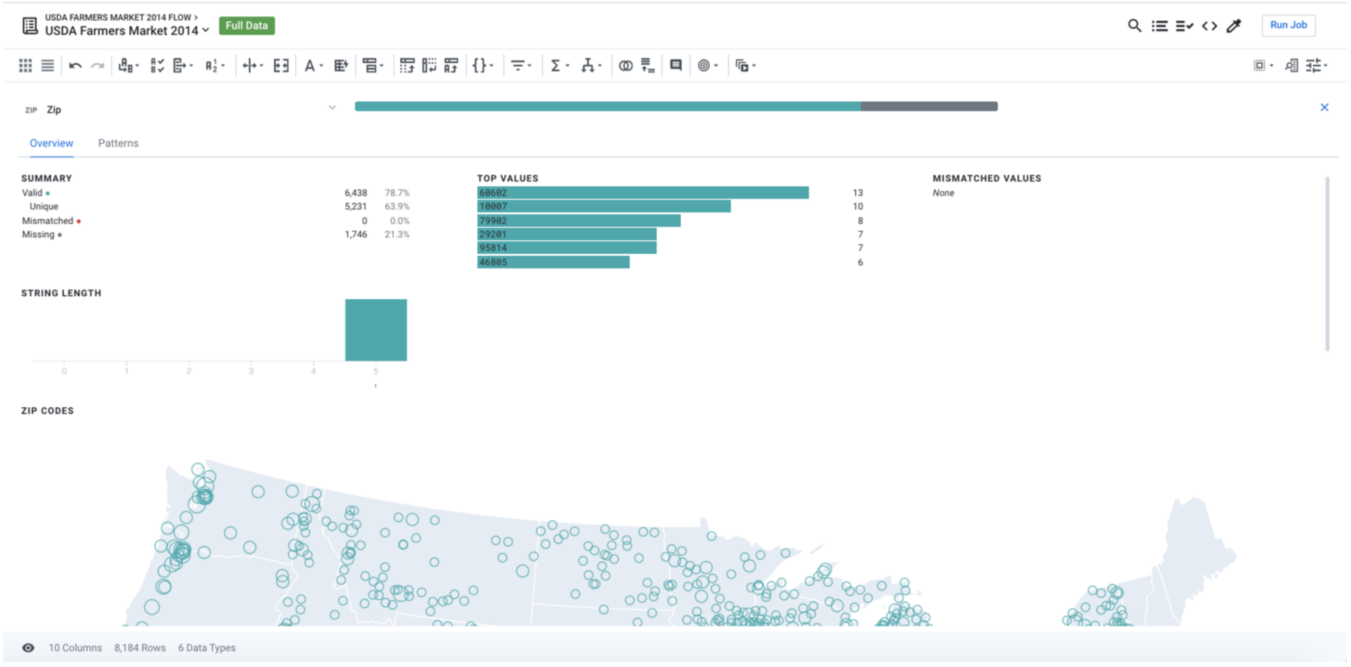
10 Columns 8,184 Rows 5 Data Types

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:


USDA FARMERS MARKET 2014 FLOW >
USDA Farmers Market 2014
Full Data
Run Job

ZIP Zip

Overview Patterns

SUMMARY

Valid ●	6,438	78.7%
Unique	5,231	63.9%
Mismatched ●	0	0.0%
Missing ●	1,746	21.3%

TOP VALUES

60602	13
10007	10
79902	8
29201	7
95814	7
46805	6

MISMATCHED VALUES
None

STRING LENGTH

ZIP CODES

10 Columns 8,184 Rows 6 Data Types

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.

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 on a per-job basis. See *Run Job Page*.

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.

> **NOTE:** It is not possible to calculate the accuracy of approximate calculations due to variations in dataset sizes and job execution environments.

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

Snowflake

For jobs executed in Snowflake, profiling jobs may also be executed in Snowflake.

NOTE: The option to pushdown profiling to Snowflake is selected for individual flows and is only applied if the job successfully executes on Snowflake. Additional limitations may apply. For more information, see *Flow Optimization Settings Dialog*.

NOTE: In Snowflake, calculations of quartiles uses a different algorithm than the same calculations in Spark. Some differences in values should be expected.

Metric Type	Measurement
Frequency (top-k)	Approximate
Numerical histograms	Approximate
Simple statistics (mean, stdev, min, max)	Exact
Quartiles	Approximate

Overview of Sampling

Contents:

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

NOTE: Generated samples are created by executing jobs on the applicable running environment. Quick Scan samples are executed in Trifacta Photon. Full Scan samples are generated in the applicable running environment on the cluster. Each running environment has a proprietary method of calculating the available volume of data in memory which is used for executing the sampling job that is launched in the running environment. As a result, the number of rows returned for the same sample type across different running environments can vary significantly.

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.
    - If the recipe is a child recipe, then the Initial Data sample indicates the selected sample of the parent recipe.
- 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.
    - The maximum number of files in a directory that can be read in the initial sample is limited by parameter for performance reasons.
        - For more information, see *Workspace Settings Page*.
    - If you are wrangling a dataset with parameters, the initial sample loaded in the Transformer page is taken from the first matching dataset.

- If the matching file is a multi-sheet Excel file, the sample is taken from the first sheet in the file.
- By default, each initial sample is either:
  - 10 MB in size
  - Limited by the maximum number of files
  - The entire dataset

- 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.

- To change the sample size, see *Change Recipe Sample Size*.

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.

> **NOTE:** The Initial Data sample contains raw data from the source. Any generated sample is stored in JSONLines format with additional metadata on the sample. These different storage formats can result is differences between initial and generated sample sizes.

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.

Sample storage

When a sample is generated, it is stored in the default storage layer in the `jobrun` directory, assigned to the user who initiated the sample. For more information, see *Overview of Storage*.

Changing sample sizes

If needed, you can change the size of samples that are loaded into the browser your current recipe. You may need to reduce these sizes if you are experiencing performance problems or memory issues in the browser. For more information, see *Change Recipe Sample Size*.

Important notes on sampling

- Depending on the running environment, sampling jobs may incur costs. These costs may vary between Trif acta Photon and your clustered running environments, depending on type of sample and cost of job execution.
- 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 Job History 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. See *Sort Rows*.
- 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 ch ecks 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:

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 Task - 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.

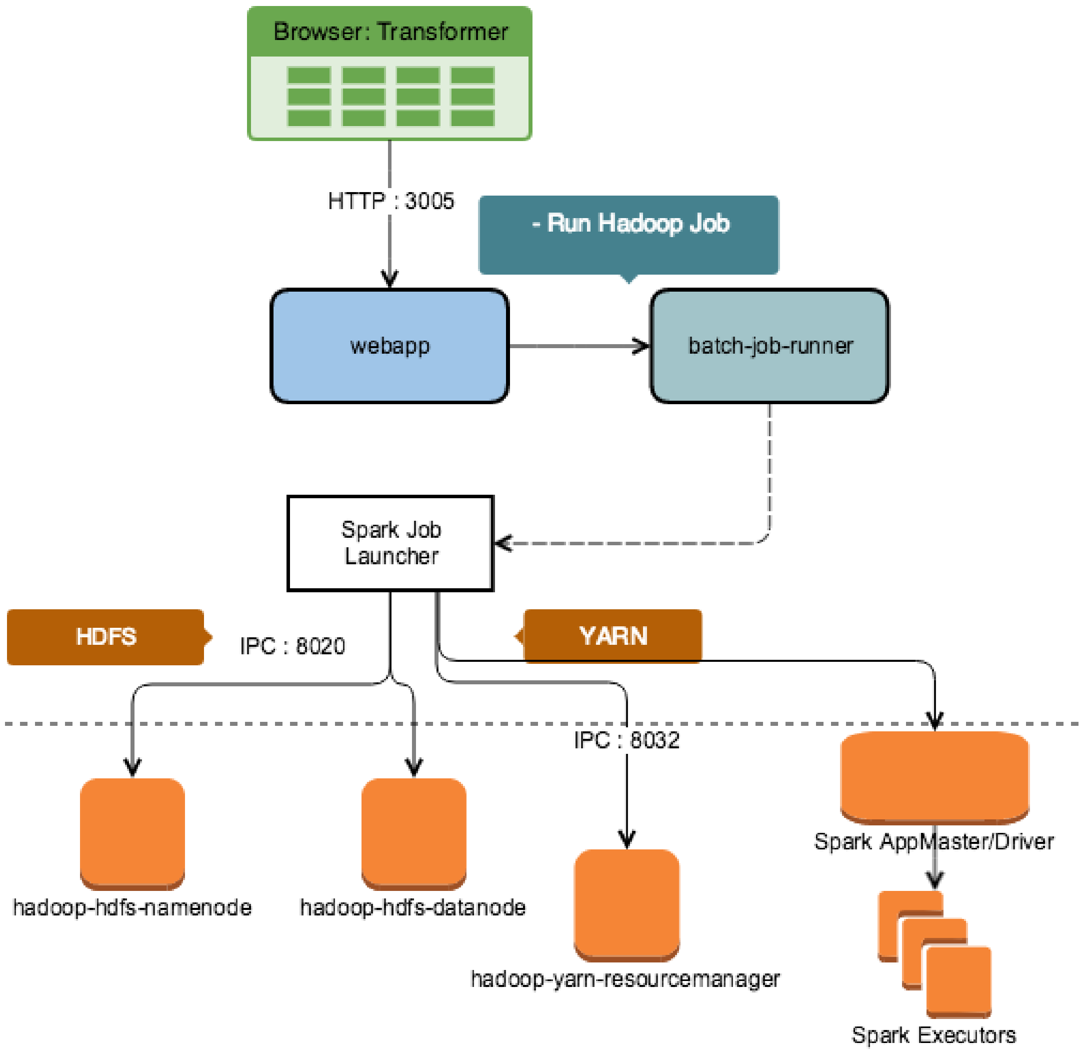


Figure: Flow chart on how a Spark-based job is executed on a Hadoop cluster

Steps:

1. **Job preparation:** A jobGroup entry is created in the database, and a job execution graph is created for each job within the jobGroup and submitted to the batch job runner service. This service requests an expanded version of the recipe in CDF format from the Designer Cloud application . This and other assets are placed in a queue for processing by the batch job runner.
2. **Job preparation:** When resources are available, the job is pulled from the queue and submitted to the resource coordinator of the batch job runner for execution on Trifacta Photon or the remote running environment. The job is placed in another queue for execution on the appropriate running environment.
3. **Job execution:** When cluster resources are available, the job definition, CDF script and other resources are submitted to the resource coordinating process for the selected running environment (In the above diagram, this is the Spark Job Launcher, which coordinates with YARN). This process submits parts of the job to separate nodes on the cluster for execution. Periodically, batch job runner polls this process for status. When all nodes of the cluster have completed their execution, the job results are written to the designated location, and batch job runner finishes the job execution by updating the Jobs database.

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 Job History page, you can hover over the status of the job to gather more information. See *Job History 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 . A plan run may consist of flow-based transformation jobs, as well as other tasks.

Feature Availability: This feature may not be available in all product editions. For more information on available features, see *Compare Editions*.

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 Scheduling*.

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

Feature Availability: This feature may not be available in all product editions. For more information on available features, see *Compare Editions*.

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.

Limitations

- Scheduled jobs cannot be run on Trifacta Photon.
- 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.
- 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 another running environment.
- For most array functions, such as ARRAYUNIQUE and KEYS functions, the order of elements in the output cannot be guaranteed.

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 Design er 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*.

Snowflake Running Environment

Contents:

Snowflake provides cloud-based data storage and analytics as a service. Among other infrastructures, Snowflake runs on Amazon S3. If all of your source datasets and outputs are in Snowflake locations and other conditions are met, then the entire execution of the transformations can occur in Snowflake.

Transferring the execution steps from the Trifacta node to Snowflake yields the following benefits:

- A minimum of data (recipe steps and associated metadata) is transferred between systems. Everything else remains in Snowflake.
- Recipe steps are converted into SQL that is understandable and native to Snowflake. Execution times are much faster.
- Depending on your environment, total cost of executing the job may be lower in Snowflake.

In this scenario, the recipe steps are converted to SQL, which is sequentially executed your source data in temporary tables, from which the results that you have defined for your output are written.

Tip: When running a job in Snowflake, your data never leaves Snowflake.

Tip: Execution on datasets created with custom SQL is supported.

If the requirements and limitations are met, the Designer Cloud application automatically executes the job in Snowflake.

Requirements

Feature Availability: This feature may not be available in all product editions. For more information on available features, see *Compare Editions*.

General

- This feature must be enabled by the workspace admin. See below.
- Designer Cloud application must be integrated with Snowflake. See *Snowflake Connections*.
    - The permission to execute jobs in Snowflake must be enabled.
- All sources and outputs must reside in Snowflake.
- `Spark + Snowflake` must be selected as running environment. See *Run Job Page*.
- Jobs are executed in the virtual warehouse that is specified as part of the Snowflake connection.

> **NOTE:** Job execution requires significantly more resources than ingest or publish jobs on Snowflake. Before you begin using Snowflake, you should verify that your Snowflake virtual warehouse has sufficient resources to handle the expected load. For more information, see *Snowflake Connections*.

- In your flow, you must enable all general and Snowflake-specific flow optimizations. When all of these optimizations are enabled, the job can be pushed down to Snowflake for execution. See "Flow Optimizations" below.

For Designer Cloud Powered by Trifacta Enterprise Edition

For customer-managed deployments, the following additional requirements apply:

- **S3:** Base storage layer must be S3. See *Set Base Storage Layer*.
- **AWS running environment:** The Trifacta node must be integrated with a running environment that is compatible with AWS.
    - For more information, see *Configure for EMR*.
    - For more information, see *Configure for AWS Databricks*.

Requirements across multiple Snowflake connections

If you are executing a job on Snowflake that utilizes multiple connections, the following requirements must also be met for execution of the job on Snowflake:

- All Snowflake connections used in the job must utilize to the same Snowflake account.
- All Snowflake connections used in the job must be backed by the same Snowflake primary role. For more information, see *https://docs.snowflake.com/en/user-guide/security-access-control-overview.html#enforcement-model-the-primary-role-and-secondary-roles* .

Limitations

Snowflake as a running environment requires that pushdowns be enabled for the workspace and for the specific flow for which the job is executed. If the flow and the workspace are properly configured, the job is automatically executed in Snowflake.

> **NOTE:** Snowflake is not a running environment that you explicitly select or specify as part of a job. If all of the requirements are met, then the job is executed in Snowflake when you select EMR.

- Except as noted below, all datasources and all outputs specified in a job must be located within Snowflake.

> **NOTE:** Datasources that require conversion are not supported for execution in Snowflake.

- All recipe steps, including all Wrangle functions in the recipe, must be translatable to SQL.

NOTE: When attempting to execute a job in Snowflake, Designer Cloud application executes each recipe in Snowflake, until it reaches a step that cannot be executed there. At that point, data is transferred to EMR, where the remainder of the job is executed.

- If the schemas have changed for your datasets, pushdown execution on Snowflake is not supported. Desi gner Cloud Powered by Trifacta Enterprise Edition falls back to submitting the job through another running environment.
- Some transformations and functions are not currently supported for execution in Snowflake. See below.
- If your recipe includes data quality rules, the job cannot be fully executed in Snowflake.
- Visual profiling is supported with the following conditions or requirements.
  - Visual profiles are unloaded to a stage in an S3 bucket.
  - If a stage is named in the connection, it is used. This stage must point to the default S3 bucket in use.
  - If no stage is named, a temporary stage is be created in the `PUBLIC` schema. The connecting user must have write access to `PUBLIC`.

    NOTE: Creating a temporary stage requires temporary credentials from AWS. These credentials are valid for 1 hour only. If a job is expected to run longer than one hour, you should define a named stage.

  - For more information, see *Snowflake Connections*.
- Sampling in Snowflake is supported with the following limitations:
  - The following sampling methods are not supported:
    - Stratified
    - Cluster-based
  - The following file formats are not supported for sampling in Snowflake when stored in S3:
    - CSV
    - JSON
  - Other JDBC-based sources are not supported.

Enable

Workspace Settings

The following setting must be enabled in the workspace. Select **User menu > Admin console > Workspace settings**.

Optimization	Description
Logical and physical optimization of jobs	When enabled, the Designer Cloud application attempts to optimize job execution through logical optimizations of your recipe and physical optimizations of your recipes interactions with data.

For more information, see *Workspace Settings Page*.

Flow Optimizations

You must enable the Snowflake optimizations in your flow. In Flow View, select **More menu > Optimization settings**.

NOTE: All general optimizations must be enabled for your flow, as well as the following optimizations, which are specific to Snowflake.

Optimization	Description
Snowflake > Column pruning from source	When enabled, job execution performance is improved by removing any unused or redundant columns from the source database.

Snowflake > 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.
Snowflake > Full pushdown	When this setting is enabled, all supported pushdown operations, including full transformation and profiling job execution, is pushed down to Snowflake , where possible.
Full execution for S3	If requirements are met for data sourced from S3, you can enable execution of your S3 datasources in Snowfl ake . **NOTE:** Snowflake pushdown is not supported for external S3 connections.

For more information, see *Flow Optimization Settings Dialog*.

Run Job

To execute a job in Snowflake in the Designer Cloud application :

- Your job must meet the requirements listed above.

  Tip: Some S3-based jobs can be executed in Snowflake. Additional requirements are listed below.

- Your job must not include the functions, transformations, or other unsupported elements that are listed below.
- You must select `Snowflake + Spark` as your running environment in the Run Job page.

  NOTE: If this running environment option does not appear in the Run Job page, then all required optimization settings have not been enabled for the workspace or the flow (see above) or the data or recipes do not meet the criteria for execution.

  See *Run Job Page*.

Tip: After launching the job, you can monitor job execution through the Job Details page, which includes a link to the corresponding job in the Snowflake console.

S3 File Support

⚠ **Feature Availability:** This feature may not be available in all product editions. For more information on available features, see *Compare Editions*.

In addition to Snowflake sources**,** you can execute jobs in Snowflake on source files from S3.

- The Full execution for S3 file option must enabled in your flow. For more information, see *Flow Optimization Settings Dialog*.

- For more information, see *S3 Access*.

Tip: The Snowflake running environment also supports hybrid sources, so you can use as sources S3 files and Snowflake tables in the same flow.

Requirements

- S3 or TFS is supported as the default storage layer.
- Publishing actions must be defined to be a target in Snowflake .
- The policies in use must support all of the requirements for Snowflake access and execution. See *Required AWS Account Permissions*.
- In the Designer Cloud application , the following flow optimization settings must be enabled at the flow level:
  - Snowflake optimization
  - Full execution for S3 file
  - See *Flow Optimization Settings Dialog*.
- In the Run Job page, the **Spark + Snowflake** running environment must be selected.

  > **Tip:** If this option is not available, one or more requirements for S3 file execution on Snowflake hav e not been met.

  For more information, see *Run Job Page*.

Execution requirements

> **NOTE:** For execution of S3 jobs in Snowflake , AWS credentials are passed in encrypted format as part of the SQL that is executed within Snowflake .

- Jobs executed on an empty directory fail. Each directory that is read must contain at least one file.
- When defining a publishing action, the schema that is used for the publication must match the staging schema defined in the connection. See *Snowflake Connections*.
- If temporary credentials are used to access S3, these credentials are valid for one hour only and cannot be refreshed. As a result, any ingestion process from S3 into Snowflake that takes longer than one hour is likely to cause the job to fail.
- Datasets with parameters can be composed of a maximum of 1000 files.
- All files referenced in a dataset with parameters must use a single and consistent delimiter.

Supported file formats from S3

- CSV: Files that fail to meet the following requirements may cause job failures when executed in Snowflake , even though they can be imported into Designer Cloud Powered by Trifacta Enterprise Edition. Requirem ents:
  - For job execution of CSV files in Snowflake , source CSV files must be well-formatted.
  - Newlines must be inserted.
  - Fields must be demarcated with quotes and commas.

    > **NOTE:** Escaped quotes in field values must be represented as double quotes (`""`). Escaped quotes with a backslash is not supported.

  - Each row must have the same number of columns.
- TSV
- JSON (newline-delimited)

  > **NOTE:** Snowflake only supports UTF-8 encoding for JSON files.

- TXT
- gzip and bz2 compressed formats are supported.

NOTE: Snappy compression is not supported for S3 execution on Snowflake .

Supported file encodings:

- UTF-8
- ISO-8859-1

Supported delimiters:

- Comma
- Tab
- Pipe

Supported quote characters:

- No quotes
- double quotes

Uploaded File Support

When a file is uploaded from your desktop, ingested, and stored in a storage layer that is supported for file pushdown, jobs that reference datasets created from that file are eligible for execution in Snowflake . For example, if your base storage layer is S3, then files uploaded from your desktop could be used for jobs that execute like S3 files in Snowflake . The requirements and limitations listed in the previous section apply.

Unsupported Wrangle for Snowflake Execution

The following transformations and functions are not currently supported for execution in Snowflake.

NOTE: If your recipe contains any of the following transformations or functions, full job execution in Snowflake is not possible at this time. These transformations are expected to be supported and removed from this list in future releases.

General limitations

For more information on limitations on specific push-downs, see *Flow Optimization Settings Dialog*.

Unsupported input data types

The following Snowflake data types are not supported for input into Designer Cloud Powered by Trifacta Enterprise Edition:

- BINARY
- VARBINARY
- GEOGRAPHY

NOTE: For mismatched values in columns of Integer data type, the value is published, instead of a NULL value. This is a known issue.

Unsupported Trifacta data types

None.

Unsupported transformations

The following Wrangle functions are not currently supported for execution in Snowflake.

- None.

Unsupported functions

The following Wrangle functions are not currently supported for execution in Snowflake.

Aggregate functions

KTHLARGEST
KTHLARGESTIF
KTHLARGESTUNIQUE
KTHLARGESTUNIQUEIF
APPROXIMATEMEDIAN
APPROXIMATEPERCENTILE
APPROXIMATEQUARTILE
QUARTILE

For more information, see *Aggregate Functions*.

Math functions

LCM
NUMVALUE

Partially supported:

NUMFORMAT: Only supported when used for rounding.

For more information, see *Math Functions*.

Nested functions

Partially supported:

> **NOTE:** For most array functions, such as ARRAYUNIQUE and KEYS functions, the order of elements in the output cannot be guaranteed.

Date functions

NETWORKDAYS
NETWORKDAYSINTL
WORKDAY
WORKDAYINTL
KTHLARGESTDATE
KTHLARGESTUNIQUEDATE
KTHLARGESTUNIQUEDATEIF

KTHLARGESTDATEIF
EOMONTH
SERIALNUMBER

String functions

DOUBLEMETAPHONEEQUALS
TRANSLITERATE

For more information, see *String Functions*.

Type functions

Partially supported:

IFMISSING

> **NOTE:** When the IFMISSING function immediately follows the PREV function in your recipe steps, Snowflake generates an incorrect value. This is a known issue and will be fixed in a future Snowflake release.

Window functions

SESSION

For more information, see *Window Functions*.

Verify Execution

To verify execution in Snowflake, please do the following:

Steps:

1. In the left nav bar, click the Jobs link.
2. In the Job History page, select the job that you executed.
3. In the Overview tab, the value for Environment under the Execution summary should be: `Snowflake`.

For more information, see *Job Details Page*.

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, 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:

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 Settings page in the Admin console.

Locate the following setting:

```
Create examples
```

Set this value to `Enabled`.

Column by Example

In column-by-example transformations, you create a new column from an existing one by mapping input to output values.

General task:

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**
Extract Transform	You can use the extract transform to retrieve sub-strings from a column and insert into a new column.
String Functions	Wrangle supports a variety of string manipulation functions, which can be used to gather data from a string.

Overview of Data Quality

Contents:

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 Target Schema Mapping*.

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.

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 *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
COUNT Function	Generates the count of rows in the dataset. Generated value is of Integer type.
COUNTA Function	Generates the count of non-null rows in a specified column, optionally counted by group. Generated value is of Integer type.
COUNTDISTINCT Function	Generates the count of distinct values in a specified column, optionally counted by group. Generated value is of Integer type.
UNIQUE Function	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
AVERAGE Function	Computes the average (mean) from all row values in a column or group. Input column can be of Integer or Decimal. See also: • *AVERAGEIF Function*
SUM Function	Computes the sum of all values found in all row values in a column. Input column can be of Integer or Decimal.
MIN Function	Computes the minimum value found in all row values in a column. Input column can be of Integer, Decimal or Datetime.
MAX Function	Computes the maximum value found in all row values in a column. Inputs can be Integer, Decimal, or Datetime.
MODE Function	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.
MINDATE Function	Computes the minimum value found in all row values in a Datetime column.
MAXDATE Function	Computes the maximum value found in all row values in a Datetime column.
MODEDATE Function	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
VAR Function	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`.
STDEV Function	Computes the standard deviation across all column values of Integer or Decimal type.
MEDIAN Function	Computes the median from all row values in a column or group. Input column can be of Integer or Decimal.
QUARTILE Function	Computes a specified quartile across all row values in a column or group. Input column can be of Integer or Decimal.
PERCENTILE Function	Computes a specified percentile across all row values in a column or group. Input column can be of Integer or Decimal.

Item	Description
APPROXIMATEMEDIAN Function	Computes the approximate median from all row values in a column or group. Input column can be of Integer or Decimal.
APPROXIMATEQUARTILE Function	Computes an approximation for a specified quartile across all row values in a column or group. Input column can be of Integer or Decimal.
APPROXIMATEPERCENTILE Function	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
COVAR Function	Computes the covariance between two columns using the population method. Source values can be of Integer or Decimal type.
CORREL Function	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*.

For more information, see *Job Details Page*.

Overview of Job Monitoring

Contents:

The Designer Cloud powered by Trifacta® platform supports detailed monitoring of a job throughout each phase of its execution.

Limitations:

- Applies only to ingest and publishing jobs
- Applies 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, the Designer Cloud powered by Trifacta platform 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 issued 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:

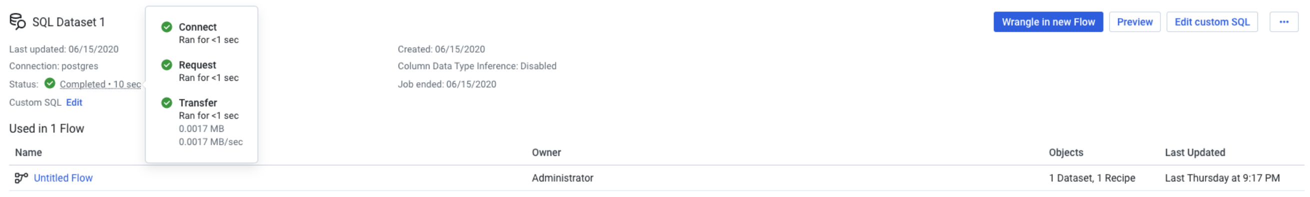
For long-loading datasets, you can track the progress of the import through the Import Data page as you specify the import. See *Import Data Page*.

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.


SQL Dataset 1

Last updated: 06/15/2020
Connection: postgres
Status: Completed • 10 sec
Custom SQL Edit

Connect
Ran for <1 sec

Request
Ran for <1 sec

Transfer
Ran for <1 sec
0.0017 MB
0.0017 MB/sec

Created: 06/15/2020
Column Data Type Inference: Disabled
Job ended: 06/15/2020

Used in 1 Flow

Name	Owner	Objects	Last Updated
Untitled Flow	Administrator	1 Dataset, 1 Recipe	Last Thursday at 9:17 PM

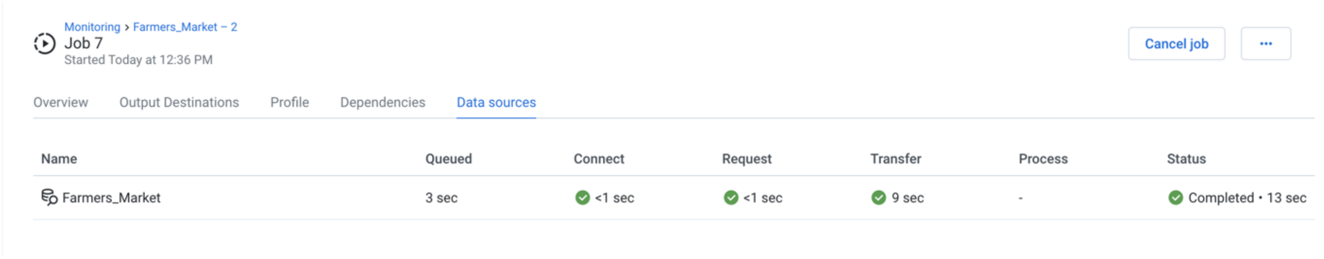
Figure: Dataset Details Page - Job Monitoring

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:

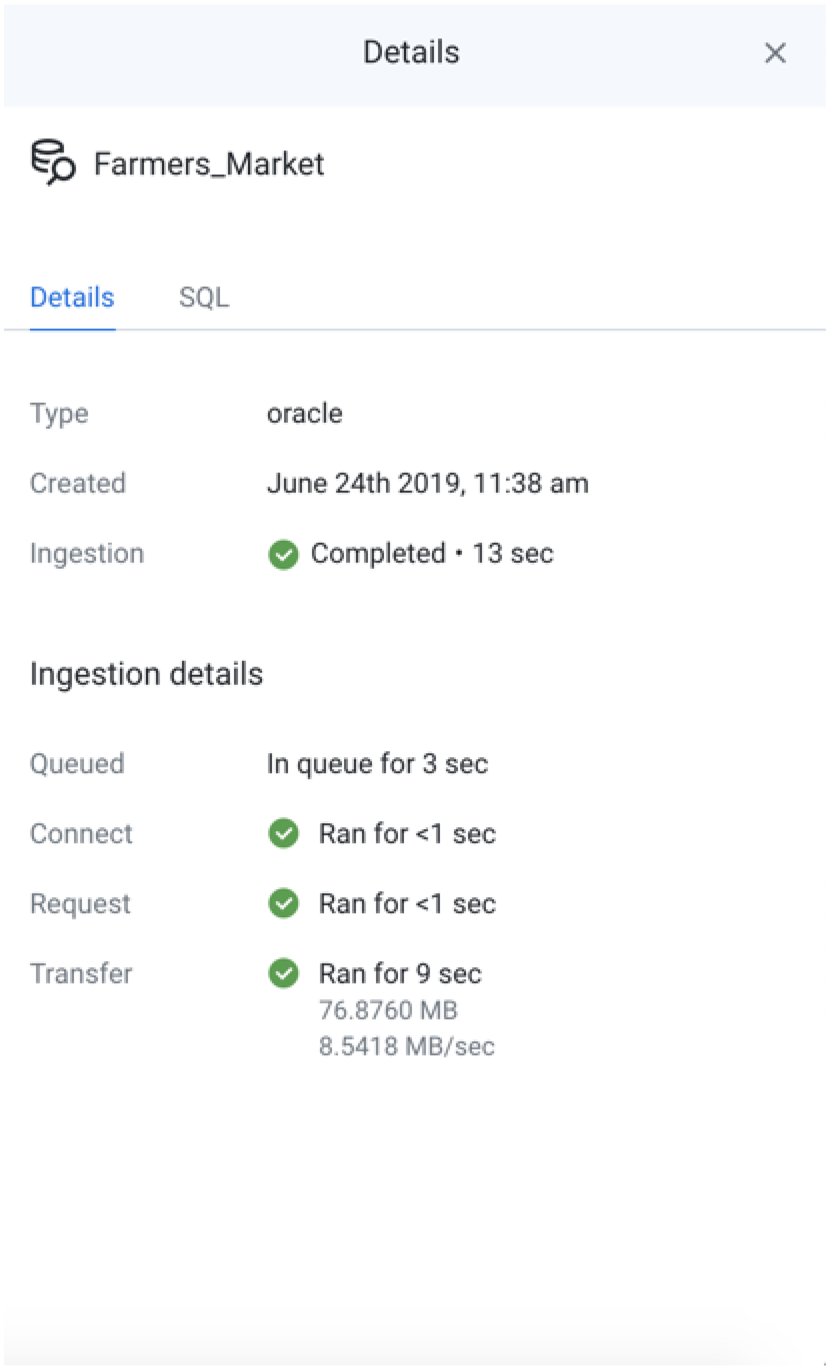

Monitoring › Farmers_Market – 2
Job 7
Started Today at 12:36 PM

Name	Queued	Connect	Request	Transfer	Process	Status
Farmers_Market	3 sec	<1 sec	<1 sec	9 sec	-	Completed • 13 sec

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

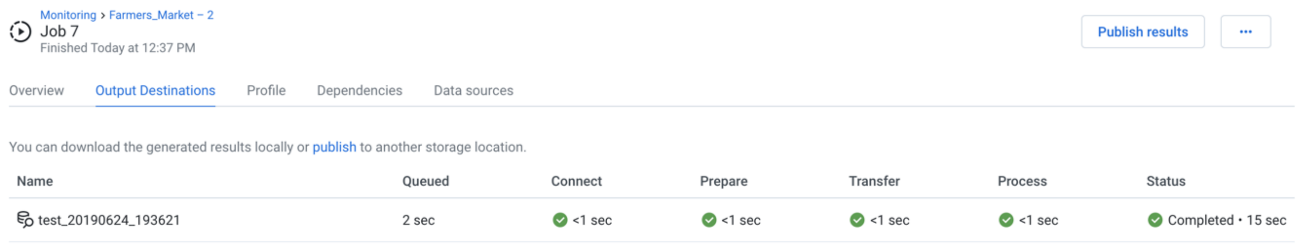
Ingestion details

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:



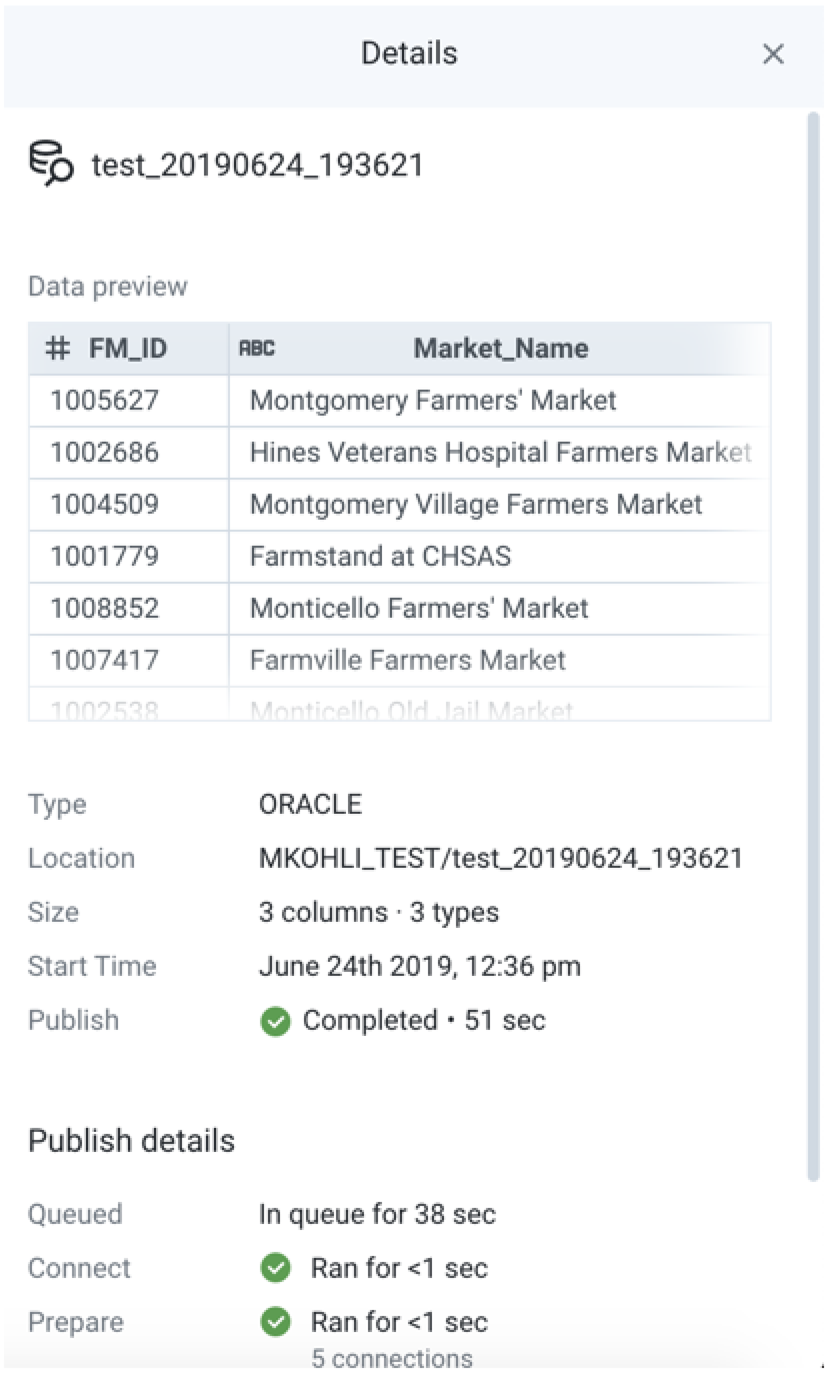
You can download the generated results locally or publish to another storage location.

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	ABC 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	MKOHLI_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 Sharing

Contents:

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 project or workspace, including any account that you may have in a different project or workspace.

> **NOTE:** You may not be permitted to share objects with users who have not yet logged into the product.

Owners and collaborators

The following are the basic types of users of a shared object:

User Type	Description
Owner	Typically, the owner is the original creator of the shared object. This user has maximum permissions on the object. **NOTE:** There can be only one owner on an object. Only the owner or a workspace admin can delete a shared object.
Workspace admin	All workspace admins have owner rights on all objects in the workspace.

Collaborator	Any user who has been shared an object is a **collaborator**. A collaborator can have the one of the following permissions on the object: • Editor • Viewer See below.

Role by object type

Individual users can be assigned one or more roles. A **role** is a set of privileges (permissions).

For each type of shareable object, an administrator can define within a role the privileges that users have on the object type. Below, you can review the basic privilege levels and the implications on sharing:

Privilege	Description	If object shared, default privileges on the object
Author	Assigned user can create and delete new objects of this type.	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 *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:

- The high-level privileges define the maximum set of privileges that you can share on an object with the target user.
- Project- or workspace-level privileges on object types can be overridden for individual objects.
- For example, a user with Viewer privileges on flows at the project or workspace level cannot be given Editor privileges on any individual flow.

Limitations:

- Fine-grained sharing privileges apply to flows, plans, 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 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.

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.

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

⚠ **Feature Availability:** This feature may not be available in all product editions. For more information on available features, see *Compare Editions*.

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 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 an 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 on the privileges for Viewer and Editor roles, see *Privileges and Roles Reference*.

Share Plans

⚠

Feature Availability: This feature may not be available in all product editions. For more information on available features, see *Compare Editions*.

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: • Collaborators cannot delete plans that have been shared with them. • Collaborator access to the plan may be further filtered based on assignments at the project or workspace level. See below.

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 role:

Plan Privilege	Description
Author	• Create plans. • Delete plans that you create. • All Editor privileges.
Editor	• Edit parameters in entitled plans • Manage email notifications on entitled plans • Update entitled plans names and descriptions • Share entitled plans • All Viewer privileges.
Viewer	• View and run entitled plans • View runs and jobs from entitled plans • Export entitled plans

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 Authorization

Contents:

Feature Availability: This feature may not be available in all product editions. For more information on available features, see *Compare Editions*.

Authorization governs how the Designer Cloud powered by Trifacta® platform users can access platform features and user-defined objects in the Designer Cloud application .

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.

Resource Roles and Privileges

Access to Trifacta objects is governed by roles in the user account.

- A **role** is a set of zero or more privileges. A user may have one or more assigned roles.

  NOTE: Roles are additive. If a user has multiple roles, the user has access at the highest level of privileges from each role.

- A **privilege** is an access level for a type of object. A role may have one or more privileges assigned to it.
- All accounts are created with the `default` role, which provides a set of basic privileges.

Standard roles

default role

All new users are automatically assigned the `default` role. By default, the `default` role enables full access to all types of Trifacta 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 Trifacta 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 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
This admin role is a super-user. The admin role enables all capabilities of the `default` role, plus:

- access to all Designer Cloud application objects, unless specifically limited. See Resource Roles and Privileges below.
- administration functions and settings within the Designer Cloud application .

NOTE: This role enables for the user owner-level access to all objects in the project or 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 this role.

NOTE: A platform administrator is automatically granted the admin role.

Custom role(s)

As needed, administrators can create custom roles for users of the project or workspace. For more information, see *Create 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 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	
Plans	none	none	none	author	
User defined functions	viewer	none	none	author	Dataprep by Trifacta 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 can invoke UDFs but cannot create, modify or delete them.

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 can invoke UDFs but cannot create, modify or delete them.

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).
- User can create, modify, and delete UDFs.

Overview of Operationalization

Contents:

⚠ **Feature Availability:** This feature may not be available in all product editions. For more information on available features, see *Compare Editions*.

Operationalization refers to a general class of platform features that enable repeated application of Designer Cloud Powered by Trifacta® Enterprise Edition on production data. 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. 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 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:** Scheduling 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 Scheduling*.

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

Feature Availability: This feature may not be available in all product editions. For more information on available features, see *Compare Editions*.

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*.

Plans

A plan is a sequence of tasks executed in the Designer Cloud powered by Trifacta platform . Plans can be used to build more complex sequences of transformation jobs, including external messaging, and much more. For more information, see *Overview of Orchestration*.

Overview of Macros

Contents:

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:

Transformation Name	`Rename columns based on a pattern`
Parameter: Option	`Find and replace`
Parameter: Columns	`All`
Parameter: Find	`' '`
Parameter: Replace with	`'_'`
Parameter: Match all occurrences	`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:

Transformation Name	`Edit formula`
Parameter: Columns	`All`

Parameter: Formula	`IF(ISVALID($col,'SSN'),'##REDACTED##',$col)`

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 :

Transformation Name	`Replace text or patterns`
Parameter: Columns	`All`
Parameter: Find	``` `{start}{digit}{4}{any}{digit}{4}{any}{digit}{4}{any}({digit}{4}){end}` ```
Parameter: Replace with	`XXXX-XXXX-XXXX-$1`

> **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**.
   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.

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:

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)	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. **Tip:** You should do all of your recipe development and testing in Dev/Test. Avoid making changes in a Prod environment. 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?
Testing (Test)	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. 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.
Production (Prod)	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. • Management of flows and jobs is typically handled via API. • The UI should be used for checking and modifying settings and perform on-demand job executions to verify operations. When errors are detected, you can: • Revert to a previous version of the flow • Apply any fixes in the Dev/Test instance for refinement and eventual updating back to the Prod instance.

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
Separate environments: Multiple instances of the platform	Dev, Test, or both environments are separate instances of the Designer Cloud powered by Trifacta platform from the Production environment. Flows are migrated between environments using the export/import mechanisms. **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.

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	In a Prod environment, you can review a flow through Flow View. **NOTE:** 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.
Jobs	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. These jobs are accessible through an interface that is very similar to a Dev environment.

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 **deployment** 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	A **release** 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. A release is created whenever you import a package into a deployment. A **package** is a ZIP file containing a flow definition that has been exported from an instance of the Designer Cloud powered by Trifacta platform .
3	flows	

		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. • The **primary flow** is the flow that you chose to export in the Dev instance. • A **secondary flow** is any flow that is included with the package for the primary flow because the primary one depends on it.

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.

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 *API Task - Define Deployment Import Mappings*.

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
Deployment Manager Page	On this page, you create deployments, for which you manage import of packages, activation of releases, and rollback to previous release as needed. For more information on the deployment objects, see below.
Flow View Page	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.

	NOTE: 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.
Job History Page	Same as Dev instance. No changes.
Connections Page	Connections that have been included as part of imported packages are available for review through the Production environment.
Admin Settings Page	Same as Dev instance. No changes to the interface. **NOTE:** 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.

Example Task

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 **NOTE:** User1 has no access to Prod.	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 *API Task - Define Deployment Import Mappings*.
   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 Task - Deploy a Flow*.

Overview of Pattern Matching

Contents:

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 *Standardize Using Patterns*.
Filter records	Keep or delete records based on patterns of values found in row data. See *Filter Data*.
Extract values	Extract values matching a pattern from one column and insert them into a new column of data. See *Extract Values*.
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 .

Overview of Target Schema Mapping

Contents:

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.

- 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.
- **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.

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 .

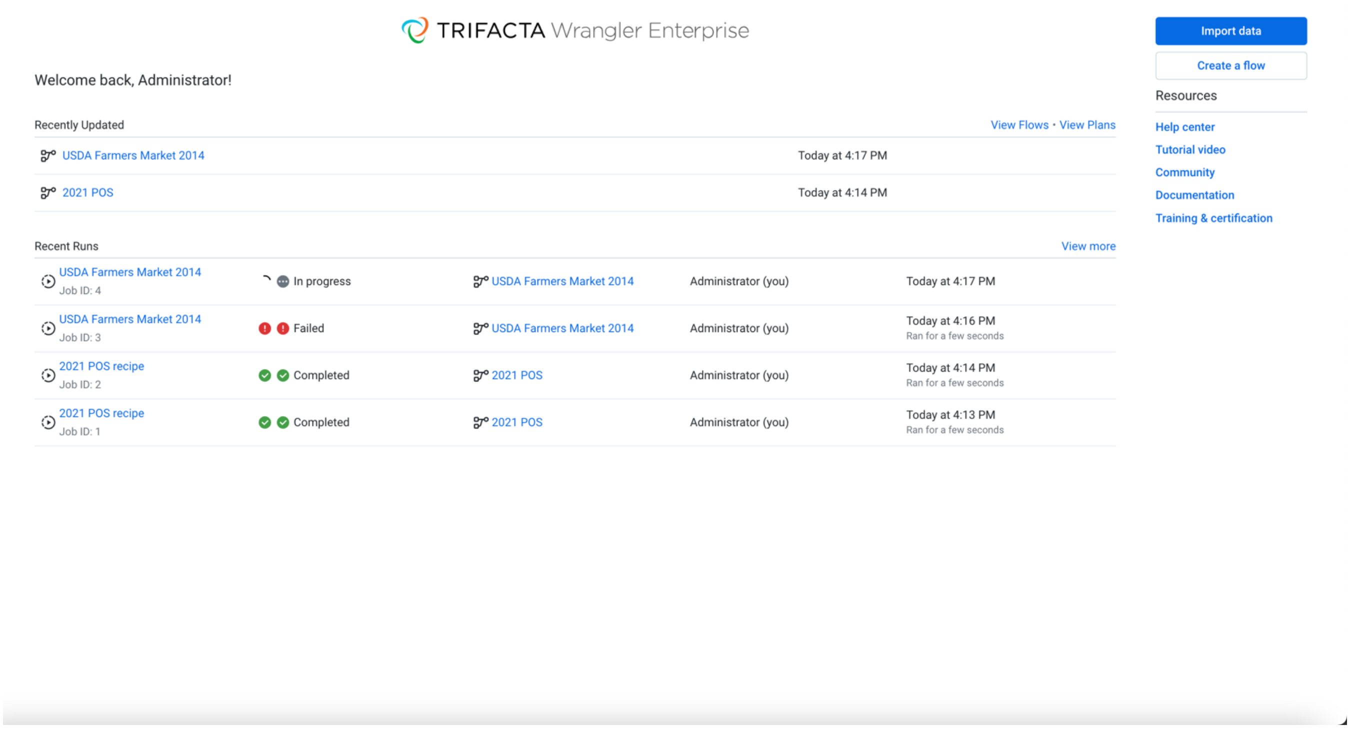
For more information on UI pages that apply to administrators, see *Admin Reference*.

Home Page

Contents:

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.



TRIFACTA Wrangler Enterprise

Welcome back, Administrator!

Recently Updated — View Flows • View Plans

USDA Farmers Market 2014	Today at 4:17 PM
2021 POS	Today at 4:14 PM

Recent Runs — View more

USDA Farmers Market 2014 Job ID: 4	In progress	USDA Farmers Market 2014	Administrator (you)	Today at 4:17 PM
USDA Farmers Market 2014 Job ID: 3	Failed	USDA Farmers Market 2014	Administrator (you)	Today at 4:16 PM Ran for a few seconds
2021 POS recipe Job ID: 2	Completed	2021 POS	Administrator (you)	Today at 4:14 PM Ran for a few seconds
2021 POS recipe Job ID: 1	Completed	2021 POS	Administrator (you)	Today at 4:13 PM Ran for a few seconds

Figure: Home Page

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.

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 *Job History 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.

- A flow is a container for one or more datasets. See *Create Flow Page*.

Plans

Feature Availability: This feature may not be available in all product editions. For more information on available features, see *Compare Editions*.

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*.

Job history

After you finish building your transformation recipe, you can run jobs to execute the recipe against your dataset. The Job History page shows status and history of your jobs. See *Job History 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*.

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*.

Support

Select **Support** to chat with our support team.

Resources menu

Access help resources, including documentation, training, and more.

- **Help center:** Search for answers to your questions.
- **Tutorial video:** Review a tutorial video for how to get started with Designer Cloud Powered by Trifacta Enterprise Edition.
- **Community:** Explore the online community.
- **Documentation:** Access online product documentation.

- **API documentation:** Access reference documentation for available endpoints and methods.
  - For additional information on API tasks and other API-related content, see *API Reference*.

- **Training & certification:** 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.

  Tip: You can acquire the name of your product edition from the About screen.

- **Keyboard shortcuts:** Review available keyboard shortcuts for the Designer Cloud application .
  - You must enable keyboard shortcuts. See *User Profile Page*.

Expand/Collapse menu

Click the icon at the bottom of the left nav bar to expand or collapse the bar.

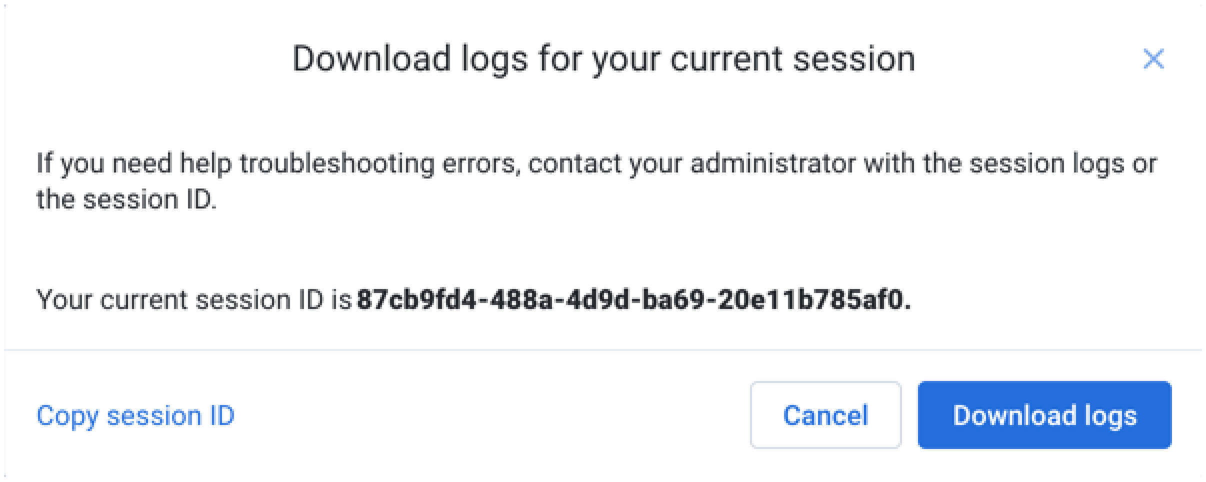
Download Logs Dialog

You can download logs for your current session in Designer Cloud Powered by Trifacta® Enterprise Edition. From the Resources 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*.



Download logs for your current session

If you need help troubleshooting errors, contact your administrator with the session logs or the session ID.

Your current session ID is **87cb9fd4-488a-4d9d-ba69-20e11b785af0.**

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

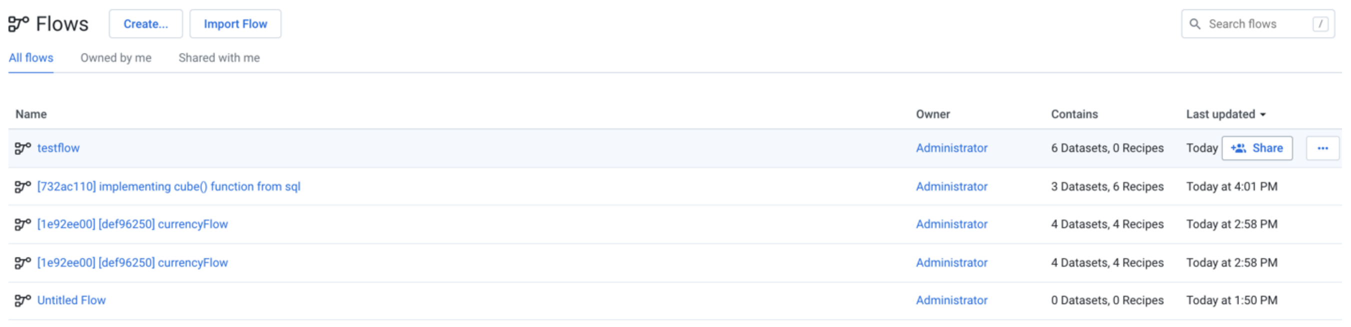
Contents:

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 hover over the shared icon link next to the flow to view the name of shared users (up to three shared users) and the total number of shared users. Also, when you click the shared icon link, the share dialog is displayed.

You can organize your flows into folders. A **folder** is simply a container for your flows. To create a folder, click **Cre ate**. Then, select **Create Folder**. For more information, see "Flows" below.



Name	Owner	Contains	Last updated
testflow	Administrator	6 Datasets, 0 Recipes	Today
[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.

- **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*.

  Tip: You can import multiple flows (ZIP files) through the file browser or through drag-and-drop. Press `CTRL/COMMAND` + click or `SHIFT` + click to select multiple files for import.

- **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*.
- **Transfer ownership:** (Available to owner or admin only) Transfer ownership of this asset to another user. See *Transfer Asset Ownership*.
- **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.

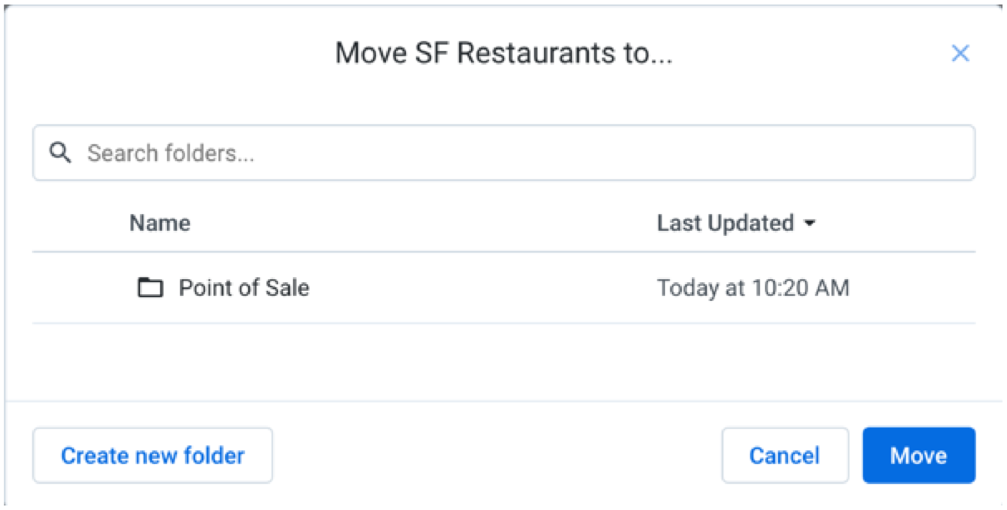


Figure: Moving a flow to a folder

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.

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:

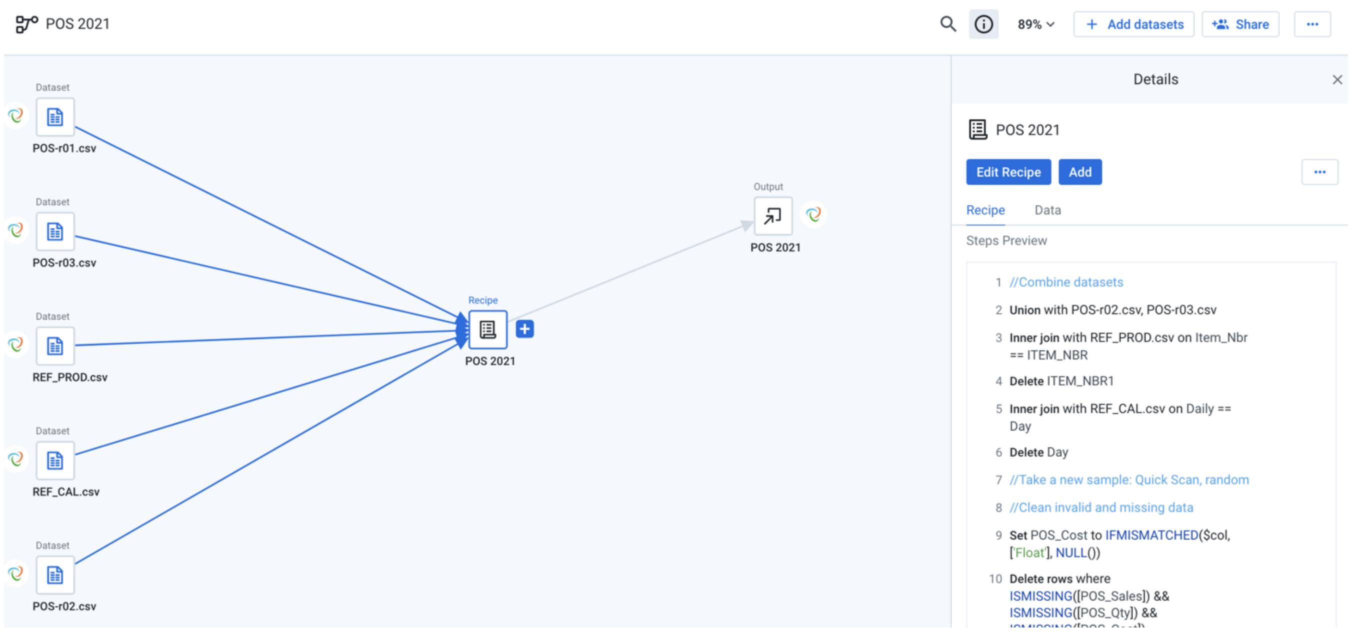
In Flow View, you can access and manage the objects that you have added to or created in the selected flow. You can perform a variety of actions to effectively manage flow development and job execution through a single page in the Designer Cloud® application .

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 Task - 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.



```
1  //Combine datasets
2  Union with POS-r02.csv, POS-r03.csv
3  Inner join with REF_PROD.csv on Item_Nbr
   == ITEM_NBR
4  Delete ITEM_NBR1
5  Inner join with REF_CAL.csv on Daily ==
   Day
6  Delete Day
7  //Take a new sample: Quick Scan, random
8  //Clean invalid and missing data
9  Set POS_Cost to IFMISMATCHED($col,
   ['Float'], NULL())
10 Delete rows where
   ISMISSING([POS_Sales]) &&
   ISMISSING([POS_Qty]) &&
```

Figure: Flow View page

Flow View Organization

Flow View is organized into the following basic areas:

Area	Description
Top bar	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 and other flow management tasks. See "Top bar" below.
Flow canvas	The primary workspace of Flow View is the flow canvas, where you build and organize the objects in your flow. See "Flow Canvas" below.
Context panel	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:** Export the flow for archive or transfer. For more information, see *Export Flow*.
- **Transfer ownership:** (Available to owner or admin only) Transfer ownership of this asset to another user. See *Transfer Asset Ownership*.
- **Delete:** 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

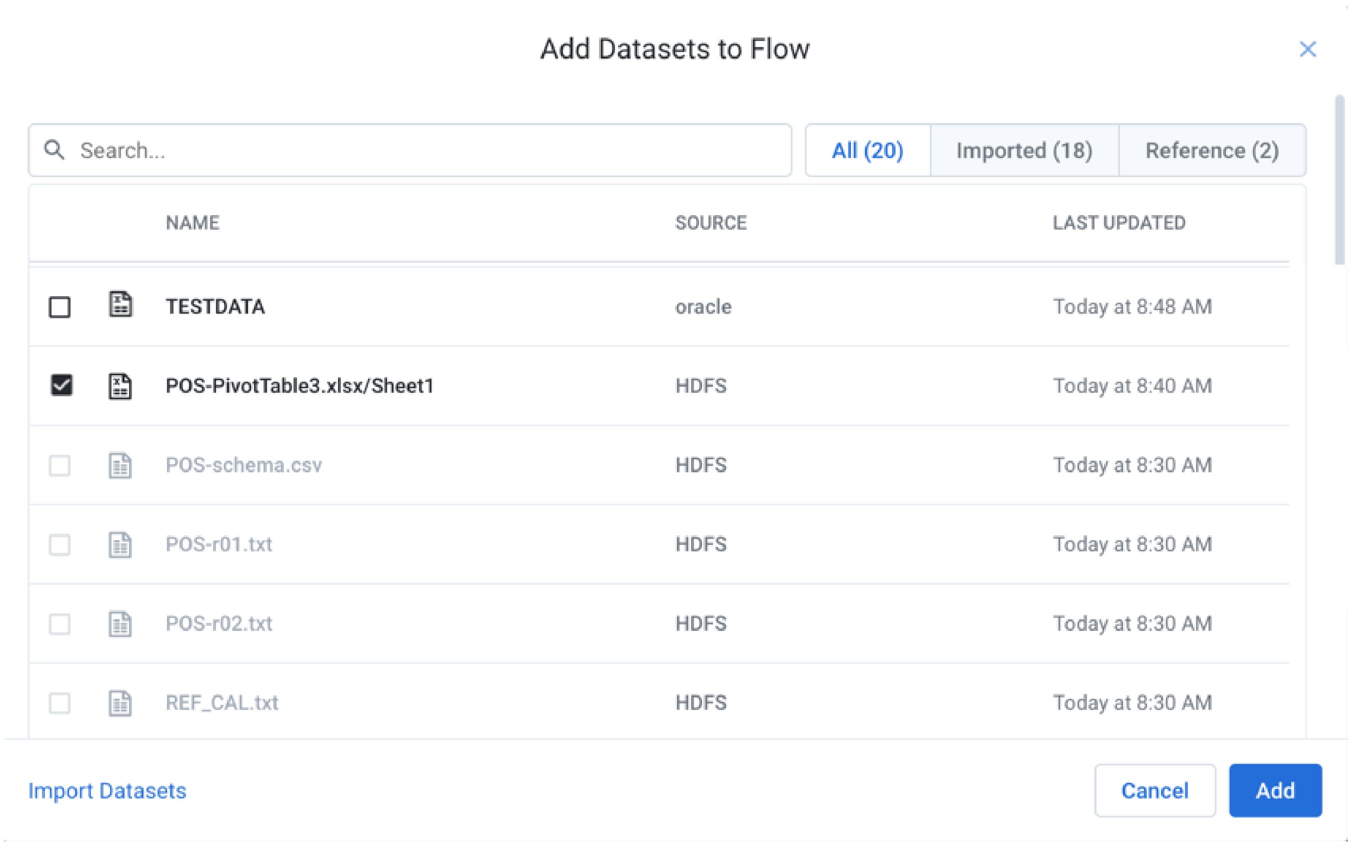
From the Flow View page, you can add data through the following objects:

- 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

Search...

All (20) | Imported (18) | Reference (2)

	NAME	SOURCE	LAST UPDATED
☐	TESTDATA	oracle	Today at 8:48 AM
☑	POS-PivotTable3.xlsx/Sheet1	HDFS	Today at 8:40 AM
☐	POS-schema.csv	HDFS	Today at 8:30 AM
☐	POS-r01.txt	HDFS	Today at 8:30 AM
☐	POS-r02.txt	HDFS	Today at 8:30 AM
☐	REF_CAL.txt	HDFS	Today at 8:30 AM

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 or Parquet 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.

- **Add dataset:** Add new datasets to the flow. See "Add Datasets to Flow".
- **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.

    Tip: You should rename your recipe as soon as you create it.

  - 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.

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 *Application Asset 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 Connections

In Flow View, you can review the details of the connections used to access the flow's imported datasets, whether you created it or it was shared with you.

Select the imported dataset that uses the connection.

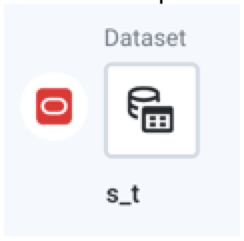


Figure: Connection icon

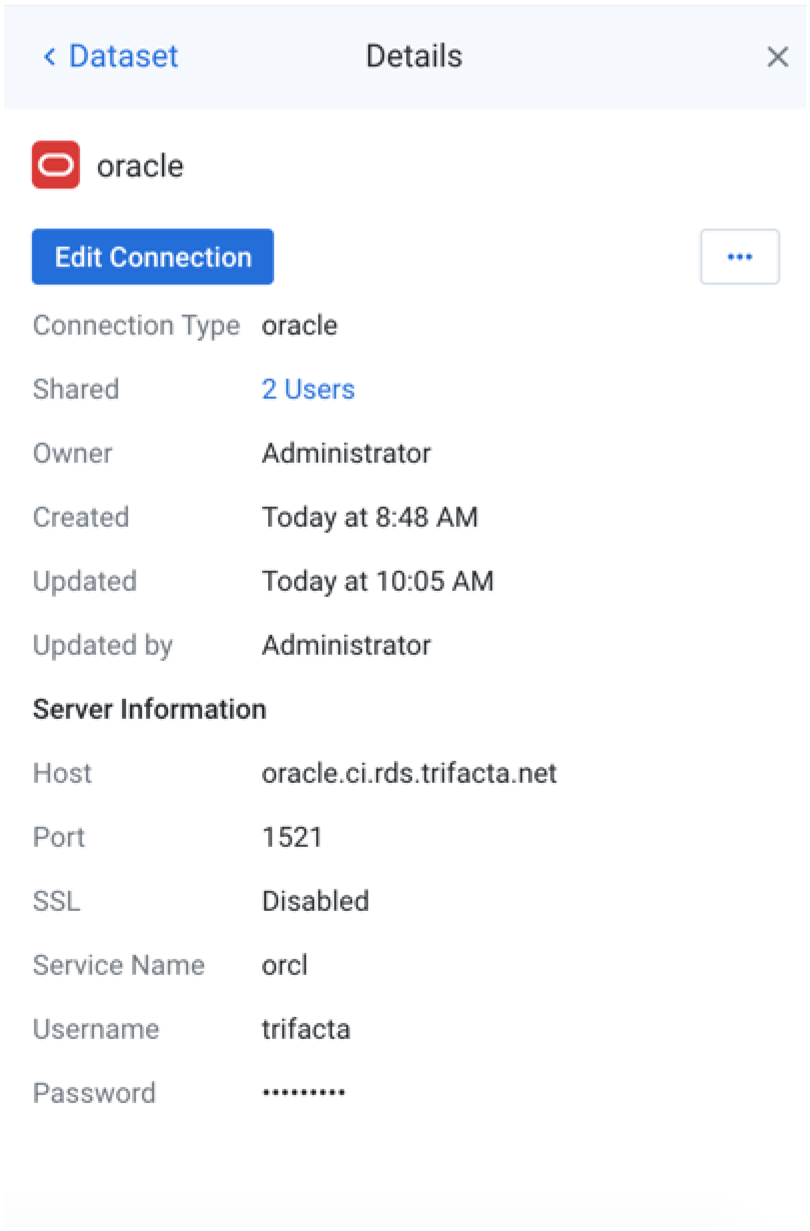
Icon context menu

Right-click the connection icon next to the imported dataset. The context menu items apply to the imported dataset. For more information, see *View for Imported Datasets*.

Details options

Select the connection to view the details options. Most options pertain to the dataset imported through the connection.

To review details about the connection, click the connection name in the Details panel.



Connection Type	oracle
Shared	2 Users
Owner	Administrator
Created	Today at 8:48 AM
Updated	Today at 10:05 AM
Updated by	Administrator

Server Information

Host	oracle.ci.rds.trifacta.net
Port	1521
SSL	Disabled
Service Name	orcl
Username	trifacta
Password	•••••••••

Figure: Connection details

Key Fields:Connection Type: For more information, see *Connection Types*.

Owner: User that owns the connection. This user can modify connection properties.

Server information: You can review information about the source to which the connection links.

Shared:

`Private` - Connection is available for use only for specified users of the platform.

`Public` - Connection is available for all users.

For more information, see *Share Connection Dialog*.

Details options

Edit Connection: If you have permissions, this option enables you to modify the connection. See

Create Connection Window.

NOTE: For shared connections, you may only modify the username and password if they were not provided to you. All other fields are read-only.

Share: Click to share the connection with other users.

Feature Availability: This feature may not be available in all product editions. For more information on available features, see *Compare Editions*.

NOTE: You can share connections that have been shared with you. You cannot modify their properties.

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*.

See *Share Connection Dialog*.

Delete: If you are the connection owner, you can delete the connection.

Deleting a connection cannot be undone. All datasets that use the connection are no longer accessible.

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.

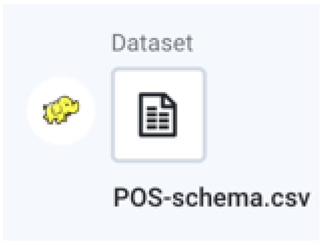


Figure: Imported Dataset icon

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.

  > **Tip:** You should rename your recipe as soon as you create it.

- **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.
  - **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*.
- **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.

- **Refresh Dataset:** If available, this option refreshes the dataset's metadata with the latest source schema.

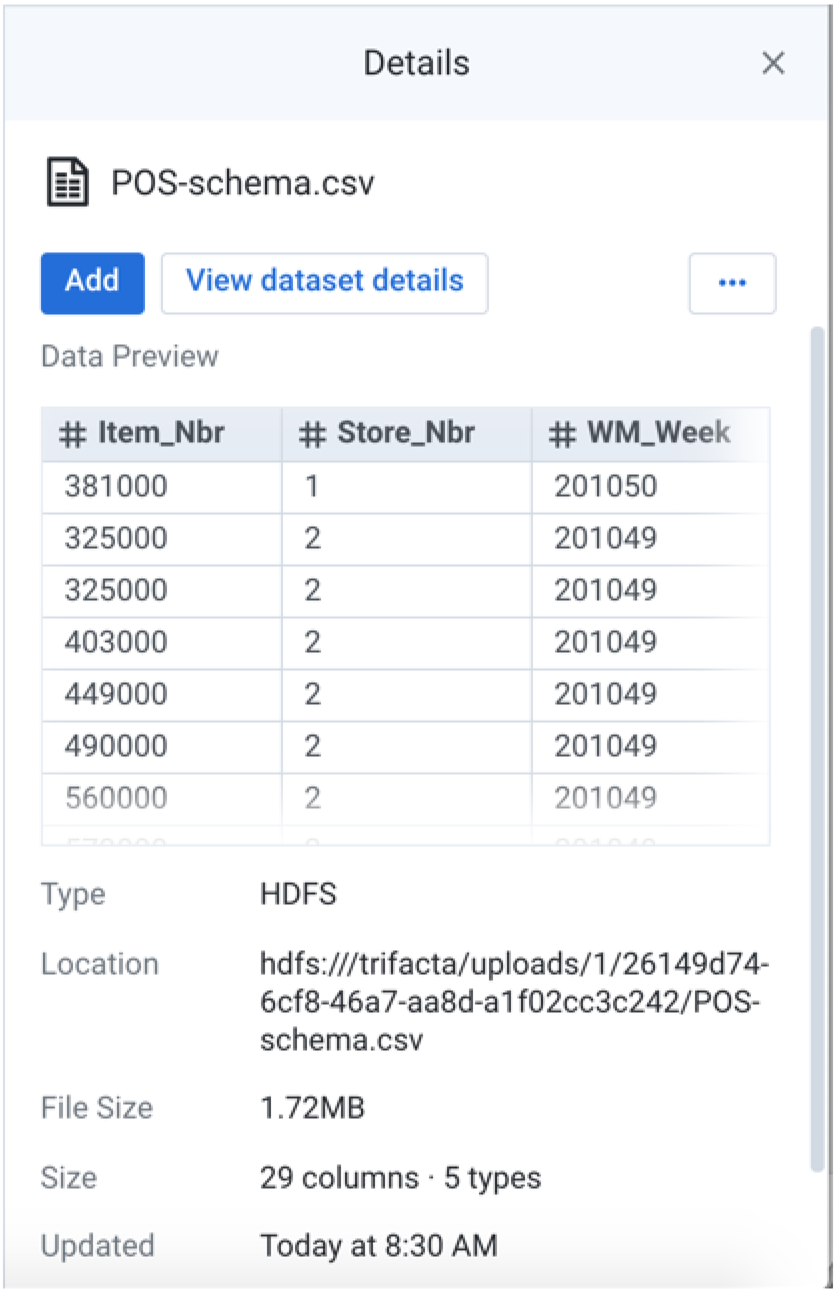
  NOTE: When a dataset is refreshed, all samples associated with the dataset are deleted, whether the dataset has changed. Samples must be recreated in their recipes.

  NOTE: If you attempt to refresh the schema of a parameterized dataset based on a set of files, only the schema for the first file is checked for changes. If changes are detected, the other files are contain those changes as well. This can lead to changes being assumed or undetected in later files and potential data corruption in the flow.

  For more information, see *Overview of Schema Management*.

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 more from the right-side panel.


Details

POS-schema.csv

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

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.
- **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*.

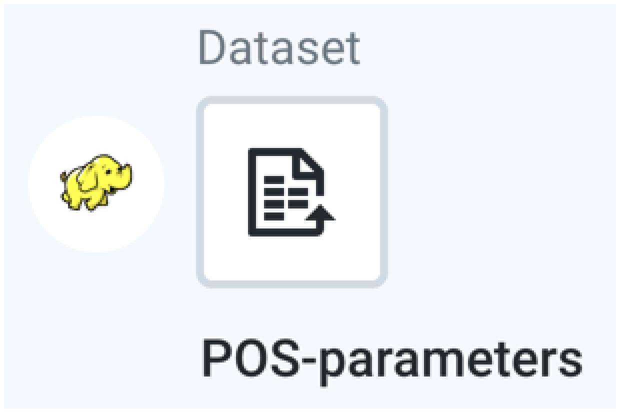


Figure: Dataset with Parameters icon

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.

  > **Tip:** You should rename your recipe as soon as you create it.

- **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:

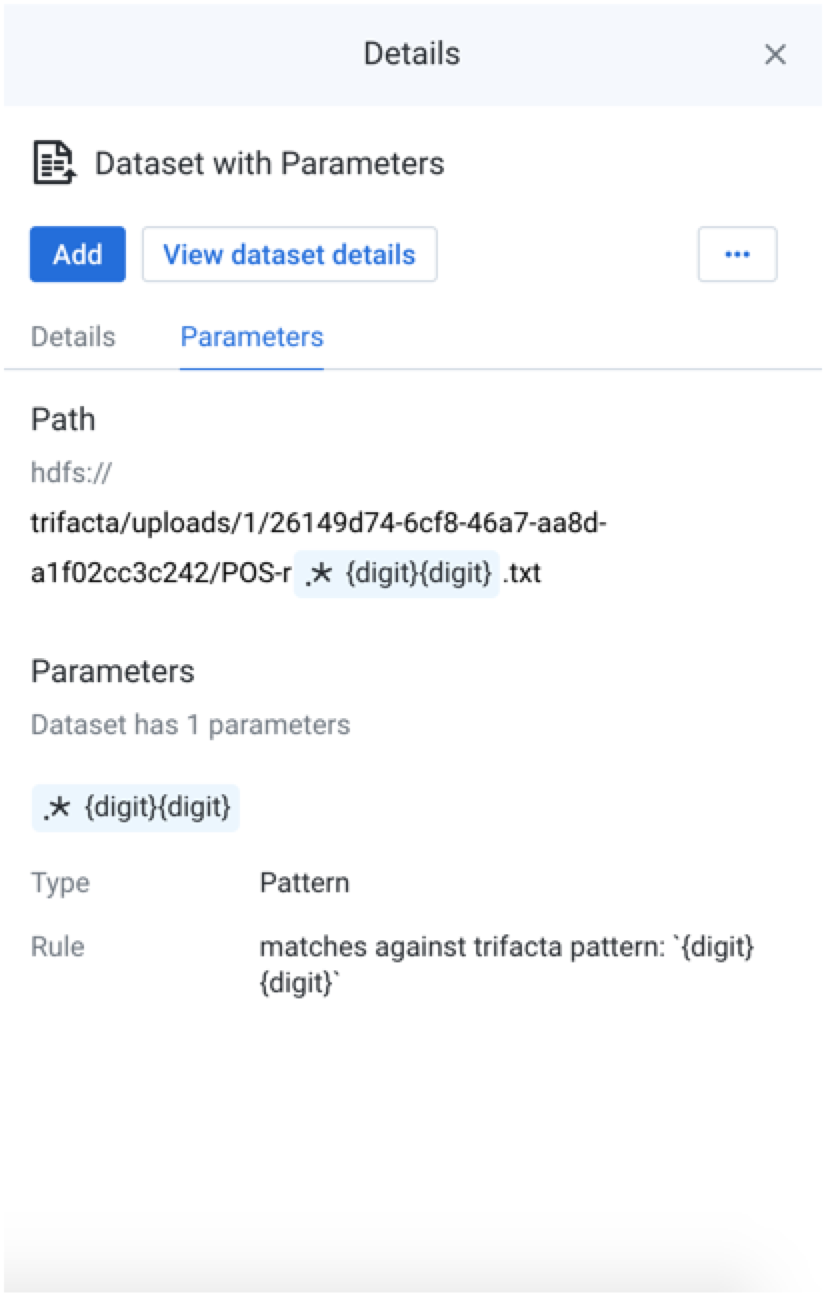
- **Add**:
  - **Recipe:** Add a recipe for this dataset.
  - **Join:** Join this dataset with another recipe or dataset. If this dataset does not have a recipe for it, a new recipe object is created to store this step.
  - **Union:** Union this dataset with one or more recipes or datasets. If this dataset 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.
- **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.



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:

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.

  > **Tip:** You should rename your recipe as soon as you create it.

- **Create Output to run**: Create an output for the recipe.
  - A recipe can have only one output.
  - See *Create Outputs*.
- **Create Reference Dataset**: Create a reference dataset of the recipe.
  - A recipe can have only one reference dataset.
  - 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*.

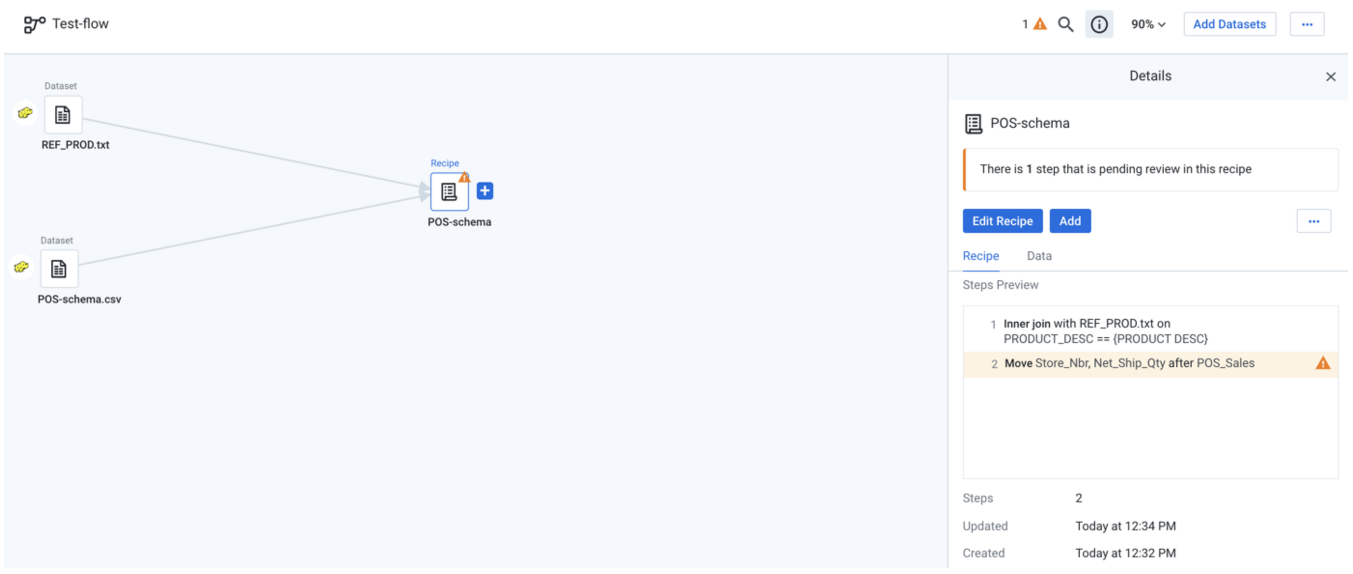
Recipes flagged for review

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.



POS-schema

There is 1 step that is pending review in this recipe

1 Inner join with REF_PROD.txt on PRODUCT_DESC == {PRODUCT DESC}

2 Move Store_Nbr, Net_Ship_Qty after POS_Sales

Steps	2
Updated	Today at 12:34 PM
Created	Today at 12:32 PM

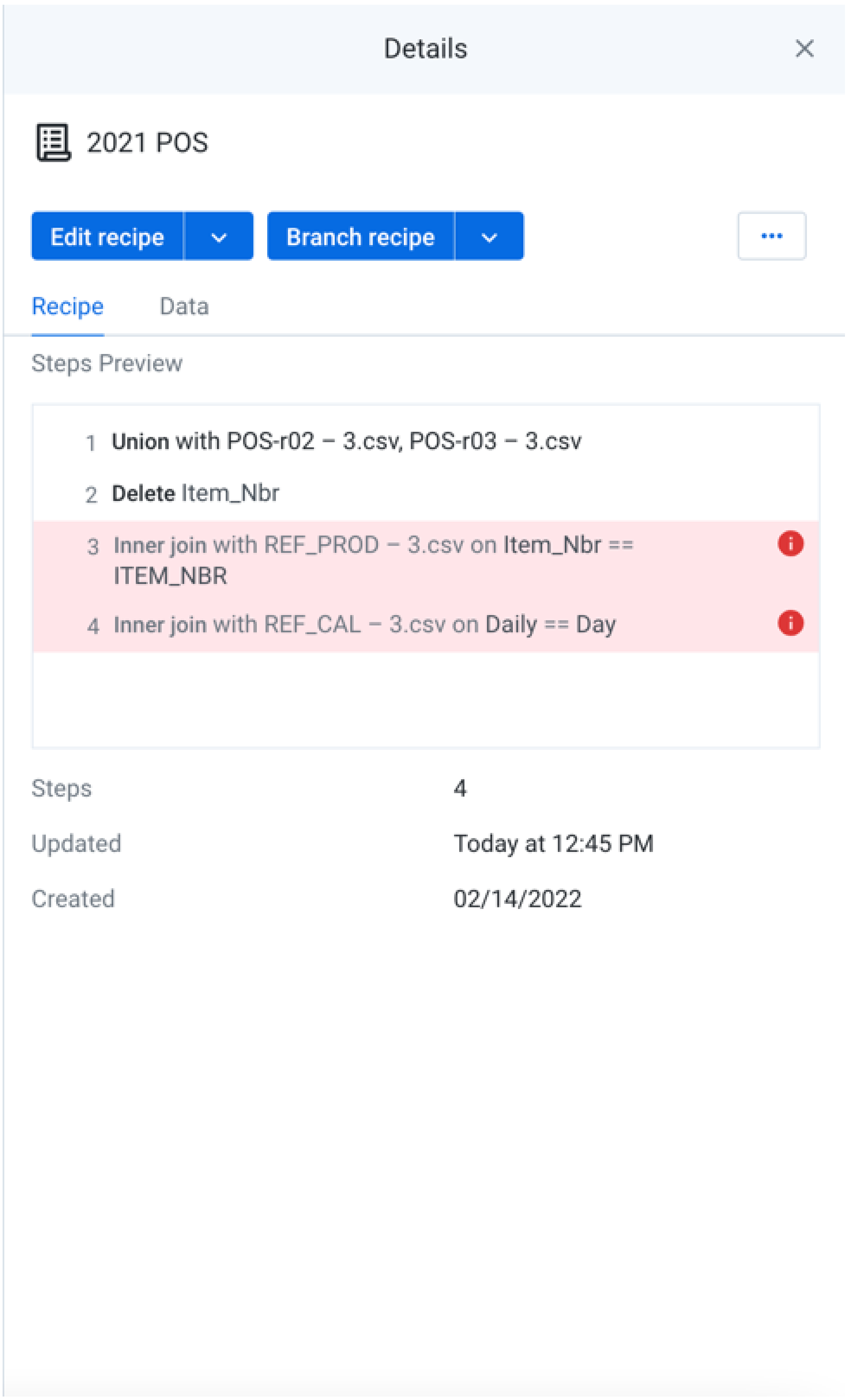
Figure: Recipe icon indicator

> **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.



1 **Union** with POS-r02 – 3.csv, POS-r03 – 3.csv
2 **Delete** Item_Nbr
3 Inner join with REF_PROD – 3.csv on Item_Nbr == ITEM_NBR
4 Inner join with REF_CAL – 3.csv on Daily == Day

Steps	4
Updated	Today at 12:45 PM
Created	02/14/2022

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*.
    - **Edit recipe without datagrid:** When this feature is enabled, you can load the recipe in the Transformer page without loading a sample, which enables rapid updating of your recipes for simple, known changes.

        NOTE: This feature may need to be enabled in your environment. For more information, see *Workspace Settings Page*.

- **Branch recipe:** Create a new recipe that branches from the selected recipe. The new recipe consumes the results of the original recipe.

    Tip: You can use branched recipes to create sequences of recipes for more modular recipe development.

- **Add recipe:** See previous. Add recipe applies if you have not created an output for the selected recipe.
- **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.
- **Append Union:** Add a union step as the new last step to the recipe.
- **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 Target Schema Mapping*.

View for Reference Datasets

Contents:

A **reference dataset** is a reference to a recipe's output, which can been 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*.

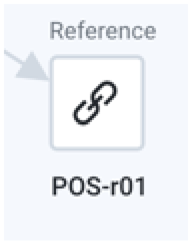


Figure: Reference Dataset icon in source flow

Icon context menu

The following options are available when you right-click the reference dataset:

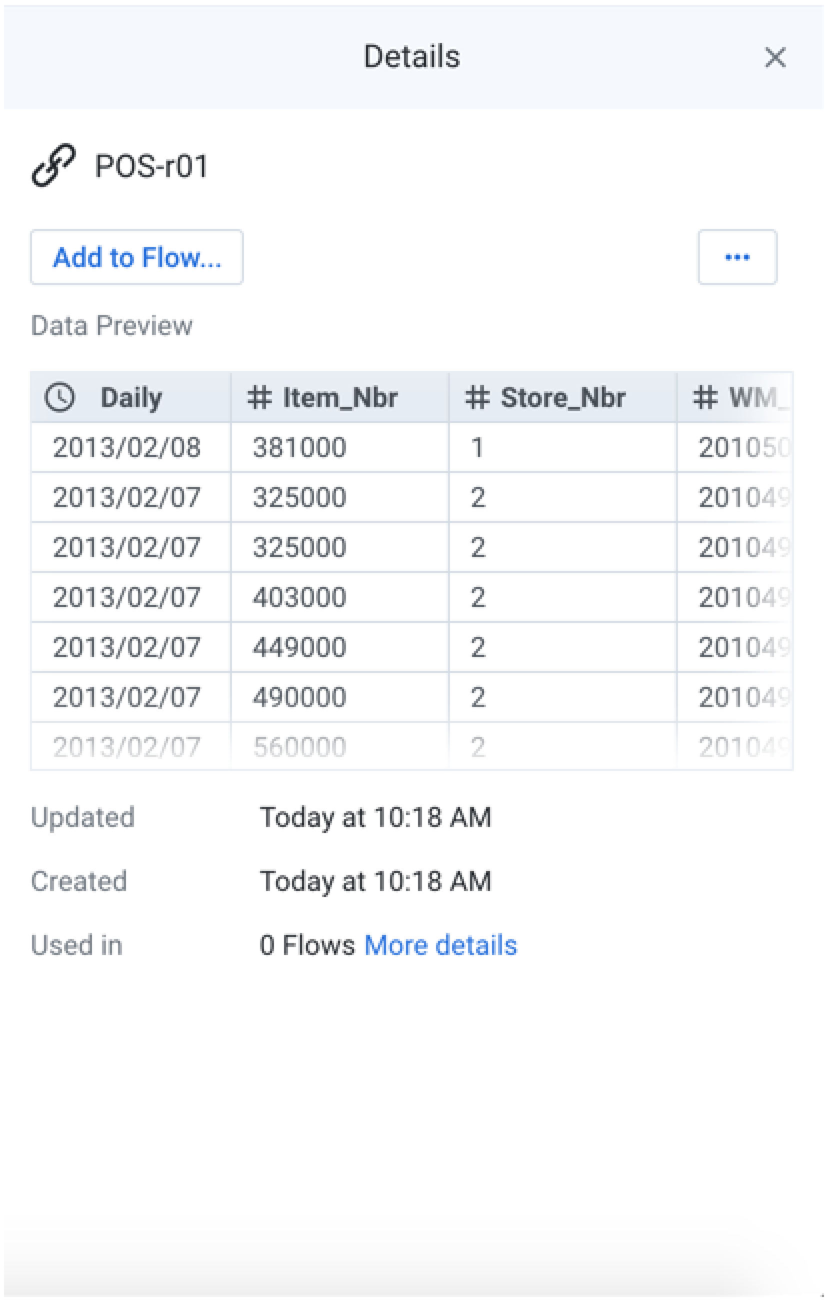
- **View details**: Explore details of the reference dataset. See *Dataset Details Page*.
- **Add to Flow:** Click to add the reference dataset to a new or existing flow.
- **Edit name and description:** Change the name and description for the object.
- **Delete Reference Dataset:** Remove the reference 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

Data Preview

Daily	Item_Nbr	Store_Nbr	WM_
2013/02/08	381000	1	20105
2013/02/07	325000	2	20104
2013/02/07	325000	2	20104
2013/02/07	403000	2	20104
2013/02/07	449000	2	20104
2013/02/07	490000	2	20104
2013/02/07	560000	2	20104

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 **Mor e details** to review the flows. See *Dataset Details Page*.

More details: Review details on the flows where the dataset is used.

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*.

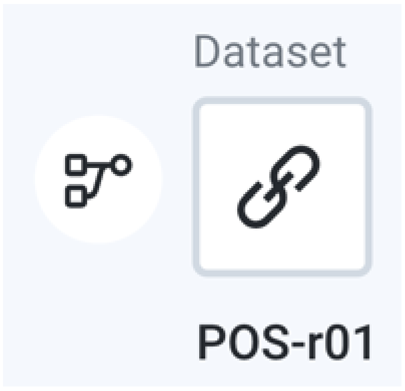


Figure: Reference Dataset icon in another flow

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*.

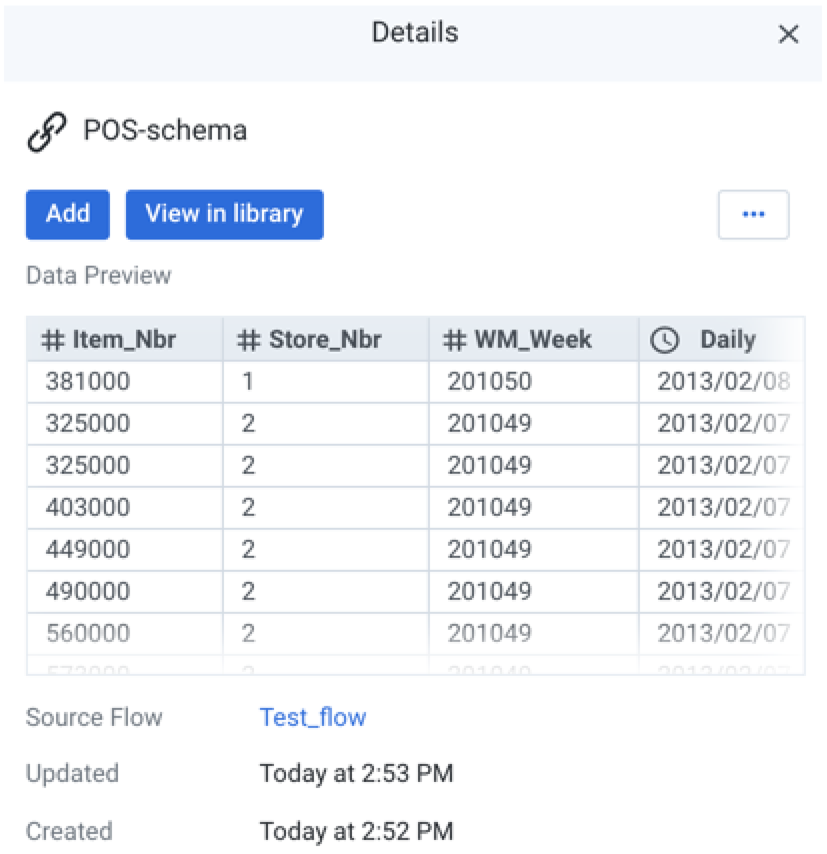
Details options

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.
    - **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 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.



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

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.

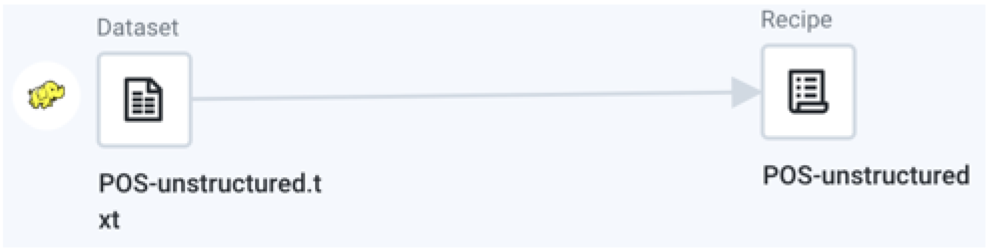


Figure: Unstructured Dataset icons

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. See *Join Window*.
- **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. See *Union Page*.

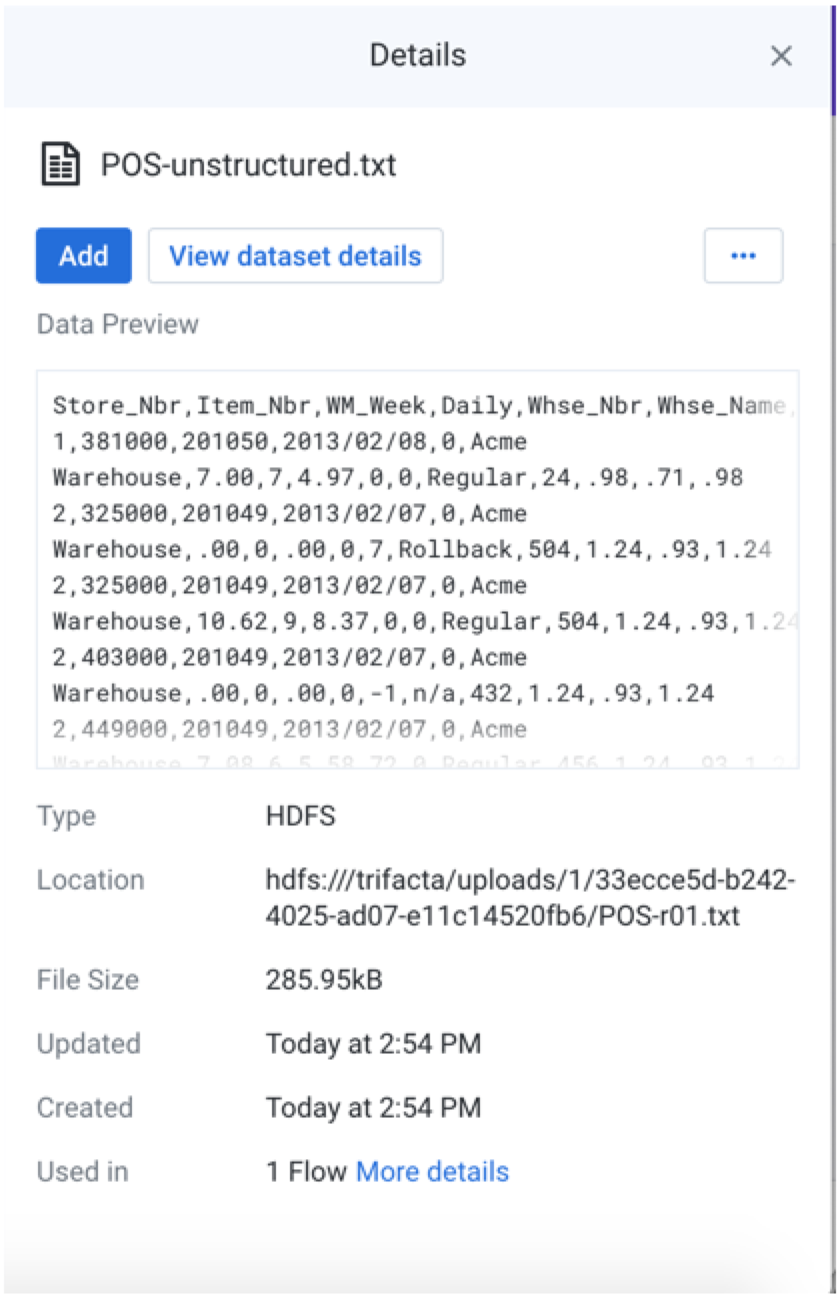
Details options

- **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.



```
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
```

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 More details

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:

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

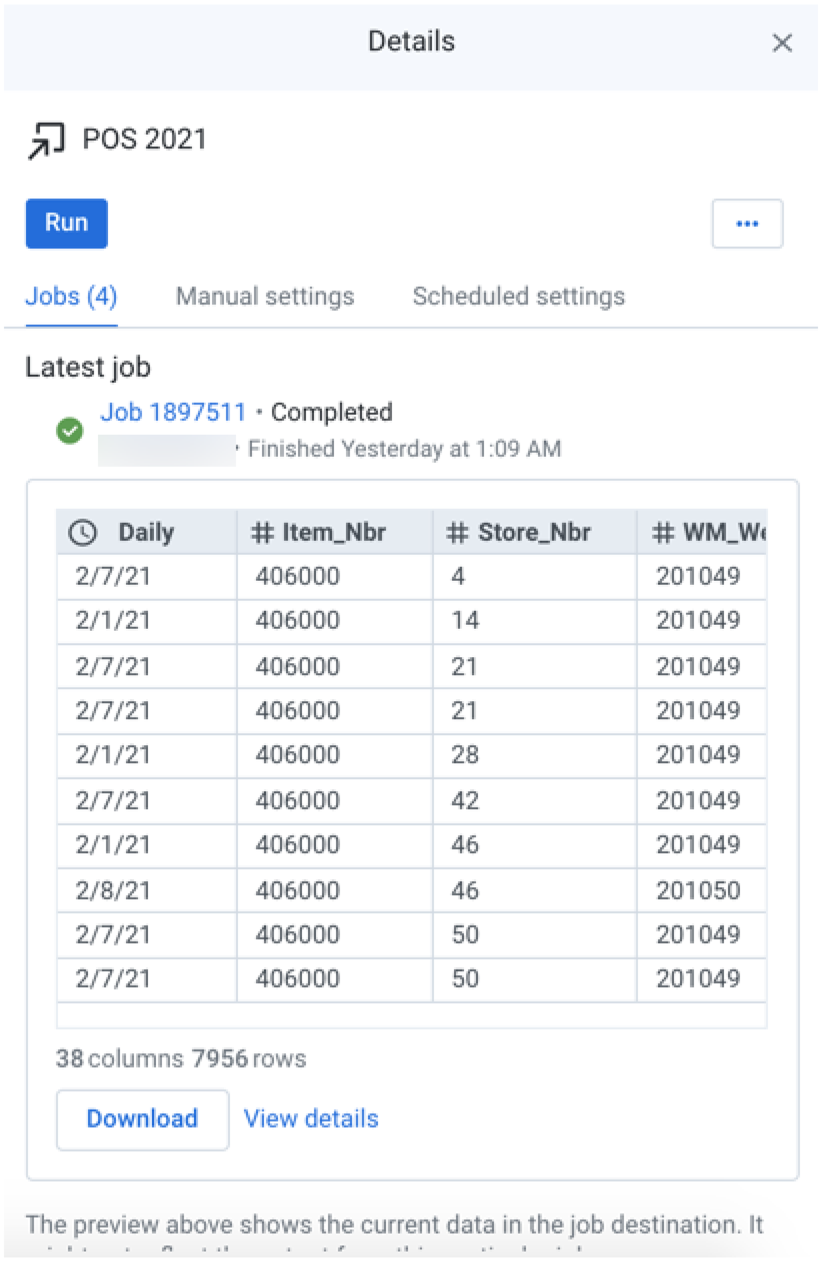
In the context panel, the following options are available:

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.

In the context menu:

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 *Job History Page*.

Jobs tab


Details

POS 2021

Latest job

Job 1897511 • Completed
[redacted] • Finished Yesterday at 1:09 AM

Daily	# Item_Nbr	# Store_Nbr	# WM_W
2/7/21	406000	4	201049
2/1/21	406000	14	201049
2/7/21	406000	21	201049
2/7/21	406000	21	201049
2/1/21	406000	28	201049
2/7/21	406000	42	201049
2/1/21	406000	46	201049
2/8/21	406000	46	201050
2/7/21	406000	50	201049
2/7/21	406000	50	201049

38 columns 7956 rows

The preview above shows the current data in the job destination. It

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.

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.

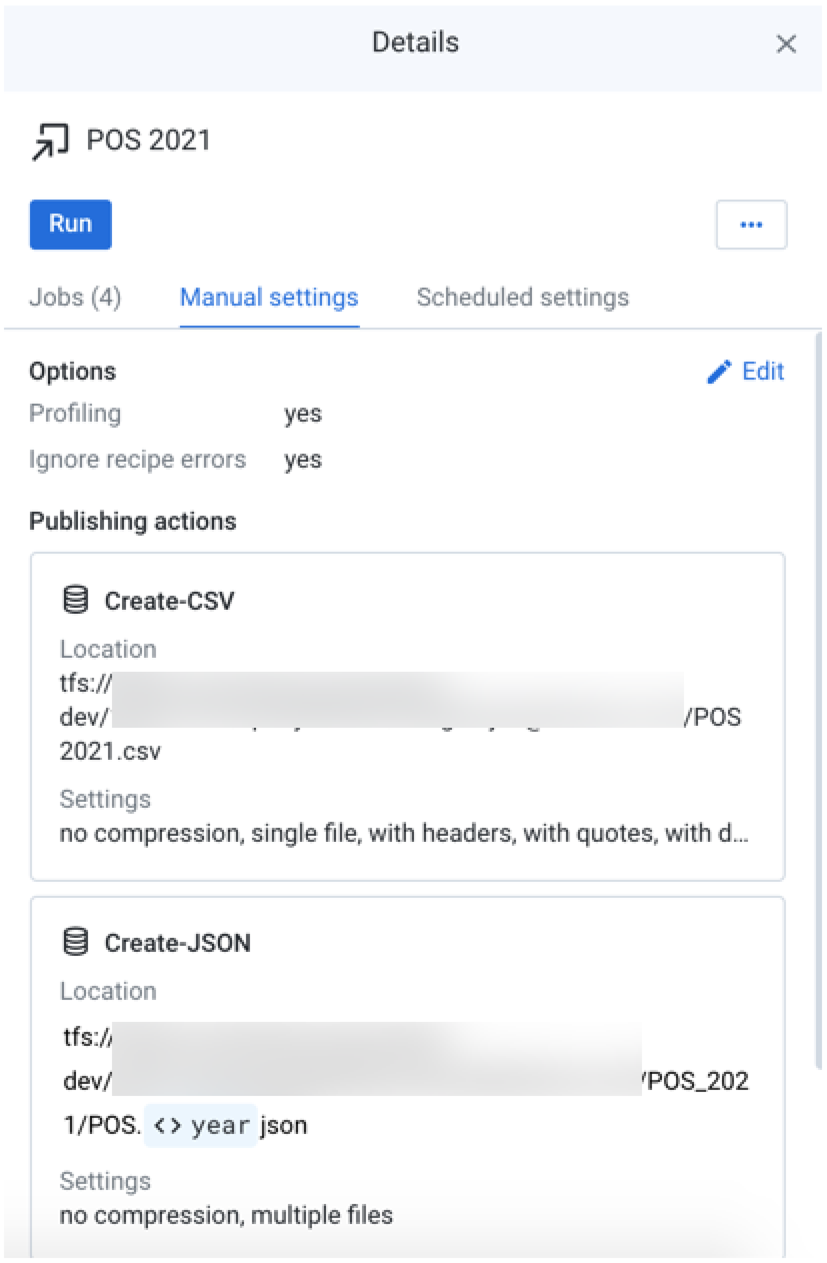
NOTE: This feature may not be enabled in your instance of the platform. For more information, please contact your Trifacta Administrator. See *Miscellaneous Configuration*.

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*.

Manual Settings tab

The Manual Settings tab contains configured outputs for manual execution of jobs through the application interface.


Details

POS 2021

Run

Jobs (4) | Manual settings | Scheduled settings

Options — Edit

Profiling: yes

Ignore recipe errors: yes

Publishing actions

Create-CSV

Location

tfs://[redacted]
dev/[redacted]/POS
2021.csv

Settings

no compression, single file, with headers, with quotes, with d...

Create-JSON

Location

tfs://[redacted]
dev/[redacted]/POS_202
1/POS. <> year json

Settings

no compression, multiple files

Figure: Manual Settings tab

Options:

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`.

Ignore recipe errors: When enabled, non-fatal errors encountered in a recipe during job execution are ignored. These errors are available for review in the Job Details page.

- To create a new manual destination, click **Add**.
- To edit the current manual setting, click **Edit.**

For more information on these settings, see *Run Job Page*.

Publishing actions:

For the manual destination, this section outlines any additional publishing actions to be taken when generating the output.

Location: 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*.

SQL scripts:

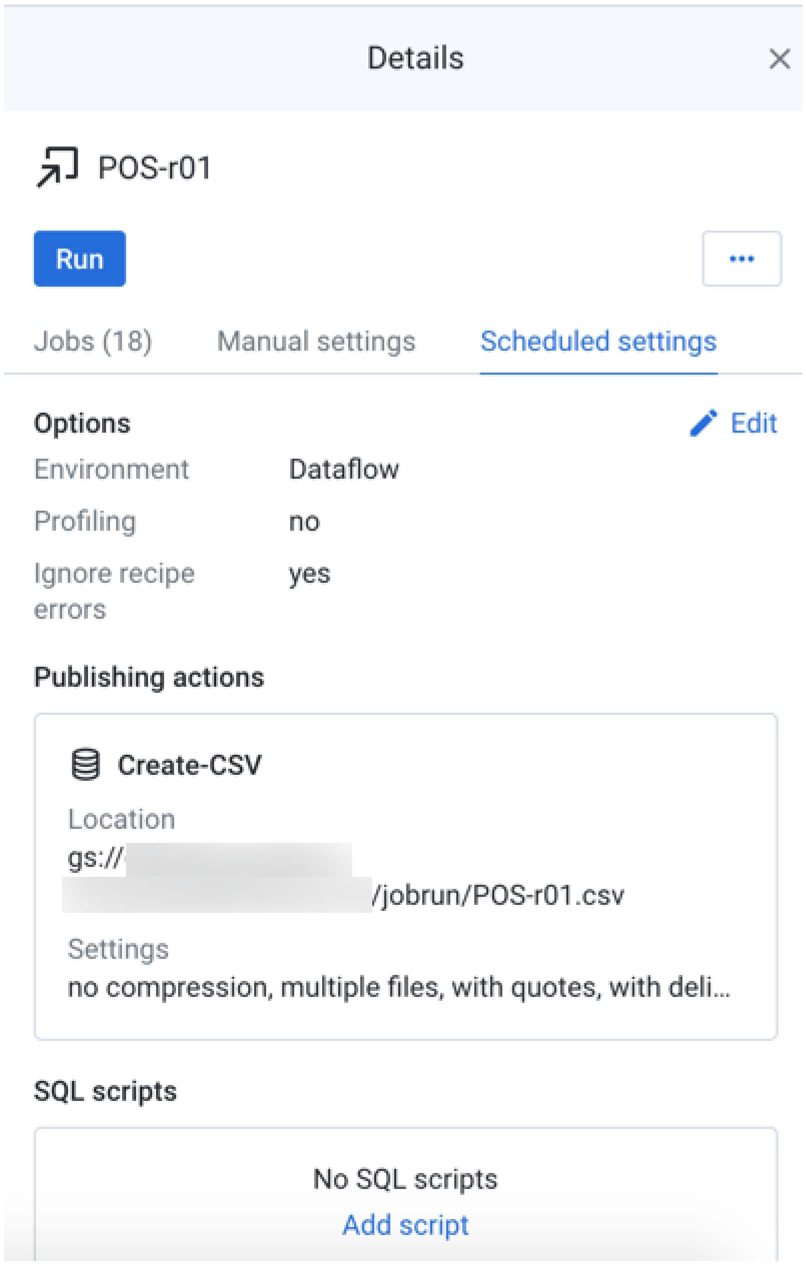
> **NOTE:** The SQL Scripts feature may need to be enabled in your environment by an administrator.

Before or after a job, you can specify one or more SQL scripts to execute. These scripts can be used for tasks like staging data for job execution or to updating an audit table on job execution. For more information, see *SQL Scripts Panel*.

Scheduled Settings tab

If a schedule has been defined for the flow, you can define a separate set of destinations, which are populated whenever the schedule is triggered and the associated recipe is successfully executed. If any input datasets are missing, the job is not run.

> **NOTE:** The Scheduling feature may need to be enabled in your environment by an administrator.



Options

Environment	Dataflow
Profiling	no
Ignore recipe errors	yes

Publishing actions

Create-CSV

Location
gs://[redacted]/jobrun/POS-r01.csv

Settings
no compression, multiple files, with quotes, with deli...

SQL scripts

No SQL scripts
Add script

Figure: Scheduled Settings tab

> **NOTE:** Flow collaborators cannot modify publishing destinations.

See *Add Schedule Dialog*.

For more information, see *Overview of Scheduling*.

Share Flow Dialog

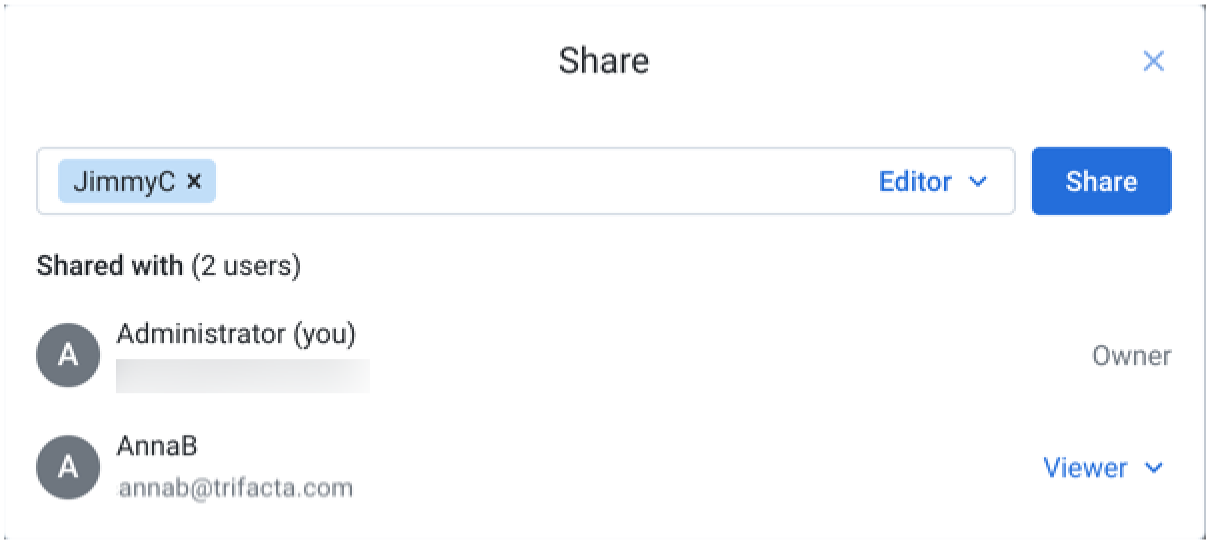
You can manage access to a flow for other users through the Share Flow dialog. In Flow View, select **Share** from the context menu.

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*.

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*.



Shared with (2 users)

A	Administrator (you)	Owner
A	AnnaB annab@trifacta.com	Viewer

Figure: Share Flow Dialog

Find users:

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.

Set access level:

For each user, you can configure their level of access to the shared flow:

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 flows at the workspace level, you cannot make the user an Editor on your flow.

NOTE: Workspace administrators have Owner-level access to all flows in the workspace. You do not need to share flows with them.

- `Editor`:
  - Modify the flow.
  - Share the flow.
  - All Viewer privileges.
- `Viewer`:
  - Read-only access to the flow.
  - Run jobs in the flow.

For more information, see *Overview of Sharing*.

To save your changes, click **Save**.

Each selected user now can access the flow through their flows page. See *Flows Page*.

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 objects.

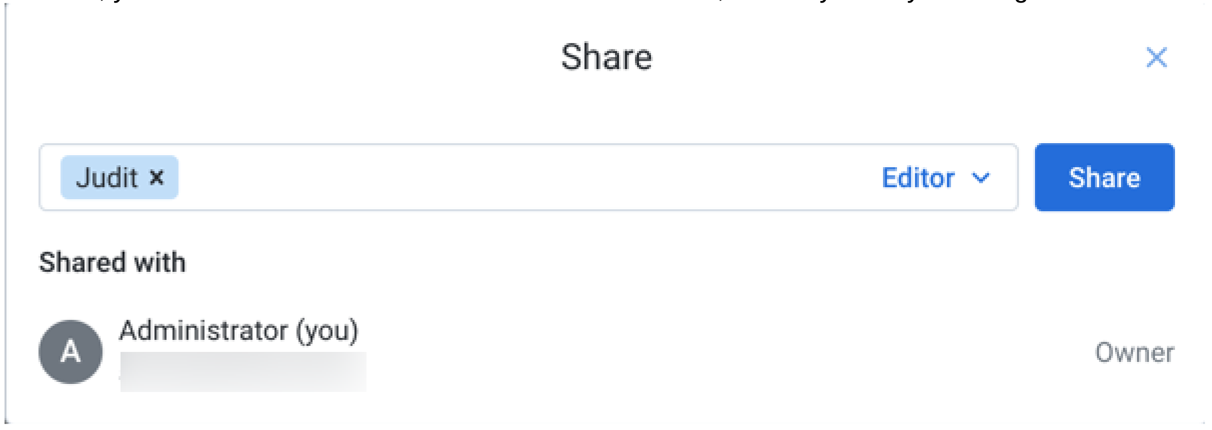


Figure: Manage Access Tab

- 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**.

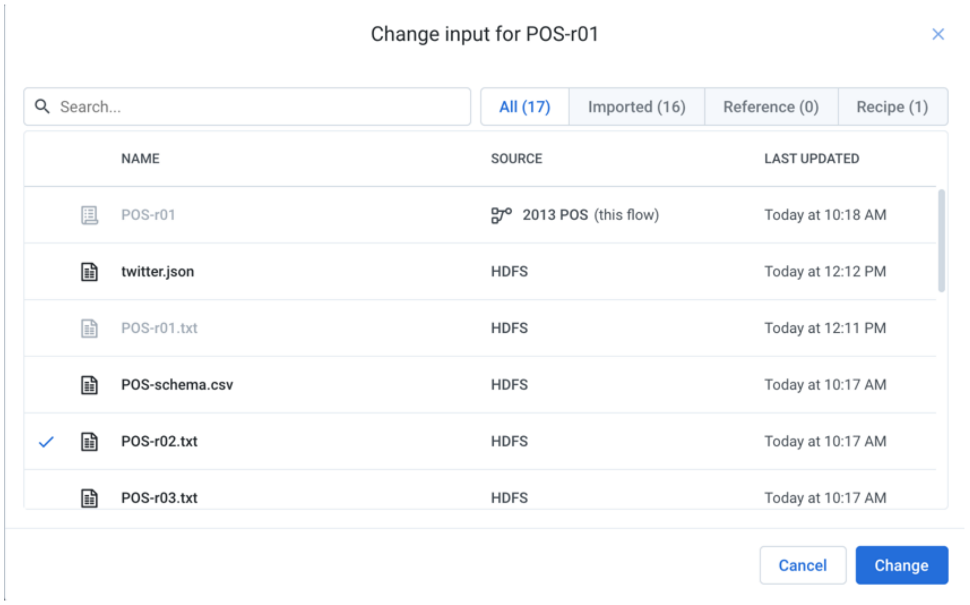
    > **NOTE:** This action removes the imported dataset and all links (edges) coming out of it. The replacement must be reconnected with any downstream objects.

    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

	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

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.

Manage Flow Notifications Dialog

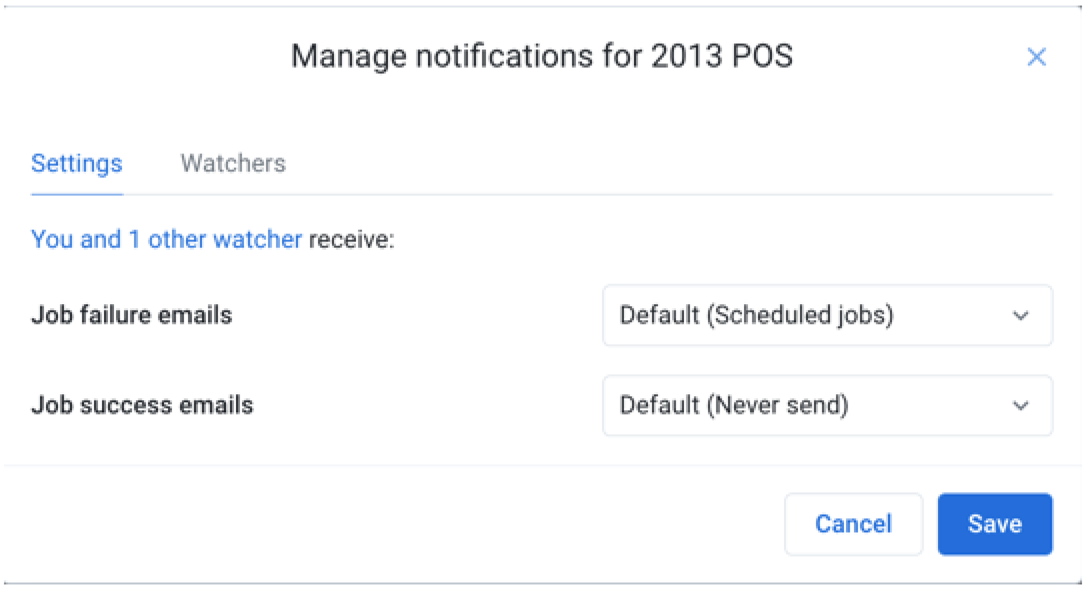
Feature Availability: This feature may not be available in all product editions. For more information on available features, see *Compare Editions*.

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 when success or failure emails for jobs executed in this flow are generated and delivered.



You and 1 other watcher receive:

Job failure emails: Default (Scheduled jobs)

Job success emails: Default (Never send)

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. **Tip:** Jobs executed via API are `manual` jobs.

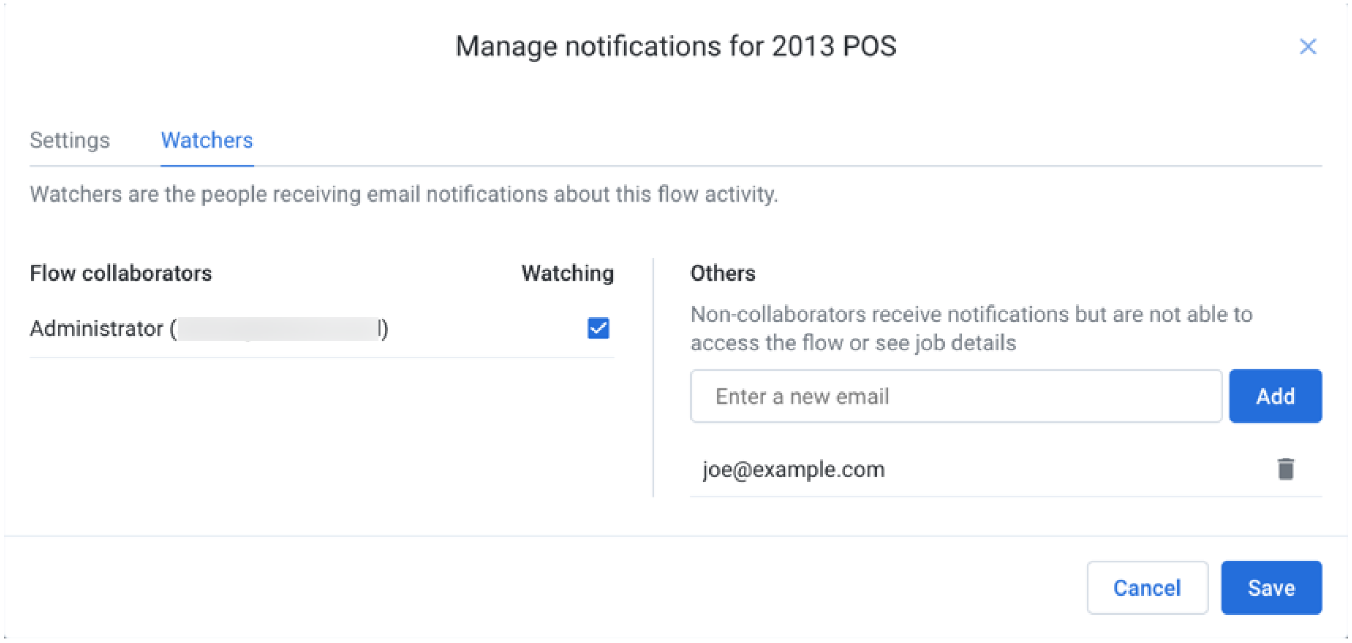
Never	Emails are never generated when jobs from this flow fail.

- **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	Watching
Administrator ([redacted])	✓

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:

Feature Availability: This feature may not be available in all product editions. For more information on available features, see *Compare Editions*.

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 one of the following types:
  - **Variable** flow parameter, which can be:
    - A string value
    - A Trifacta parameter
    - A regular expression
  - **Selector** flow parameter.
    - A selector type parameter is a set of one or more permitted values.
- 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

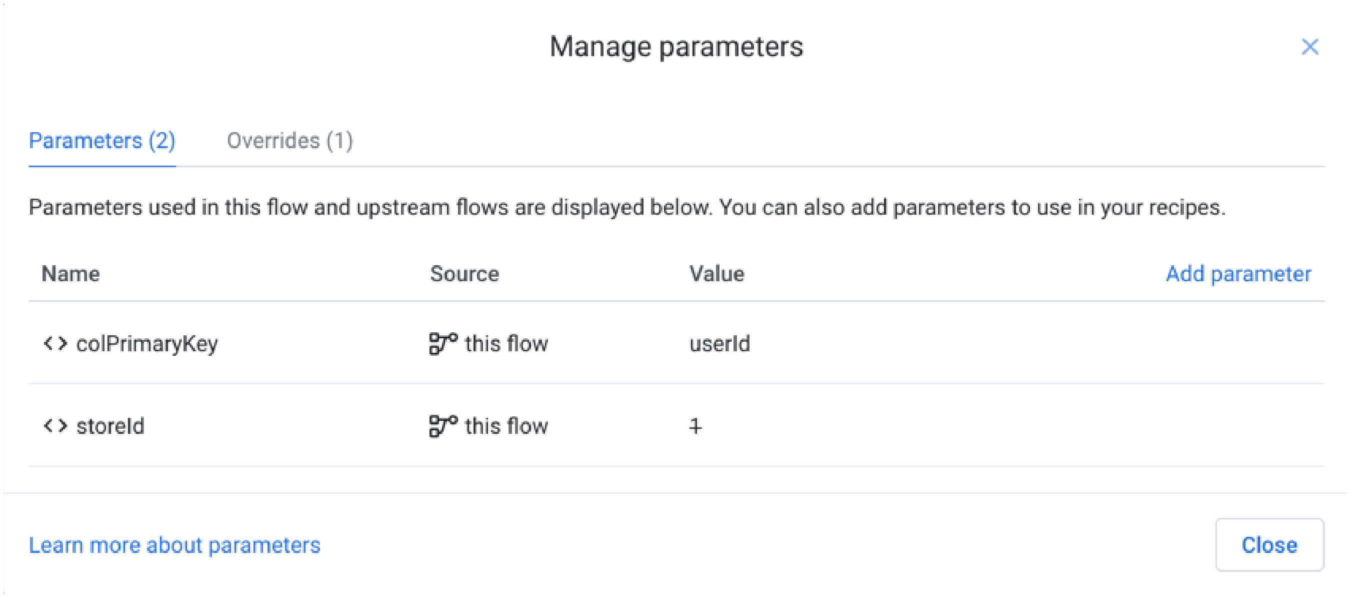
In the Parameters tab, you can manage the flow parameters in your flow.

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.

Parameters for plans: 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 used in this flow and upstream flows are displayed below. You can also add parameters to use in your recipes.

Name	Source	Value
<> colPrimaryKey	this flow	userId
<> storeId	this flow	1

Figure: Parameters tab

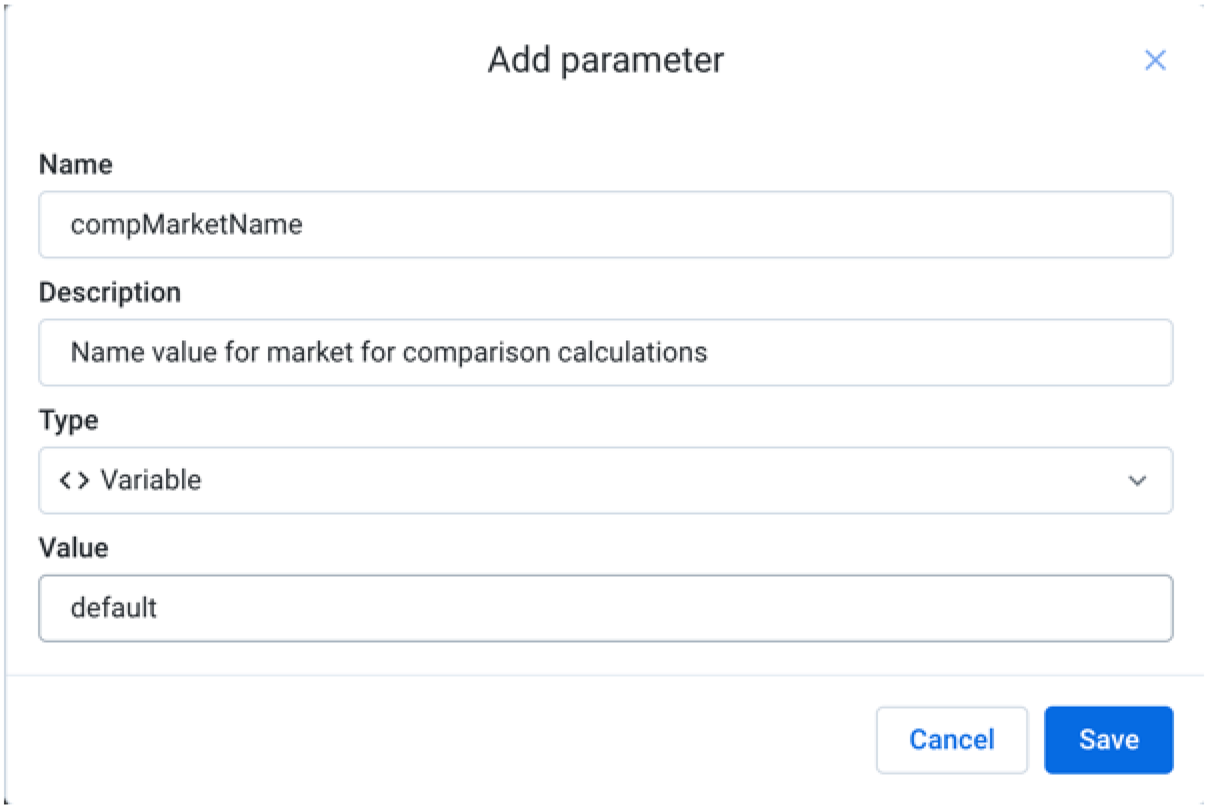
- To create a new parameter, click **Add parameter**.
  - Specify the parameter details as needed. Additional information is below.
  - Click **Save**.
- To edit a parameter's default value, hover over its entry, and click the Pencil icon.
- Additional options are available in the context menu.
  - Select **Edit description** to modify the Description value.
  - (Selector type only) Select **Edit choices** to modify the permitted values for the parameter.
- 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.

Variable type parameters

Variable type parameters are single values. Values can be specified as String values, Trifacta patterns, or regular expression patterns.



Name: compMarketName

Description: Name value for market for comparison calculations

Type: <> Variable

Value: default

Figure: Specify a flow parameter of variable type

- Specify the Name value and optional Description value.

  > **NOTE:** Name values are case-sensitive. After saving a parameter, you cannot change its name.

- For type, select **Variable**.
- 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]/
    ```

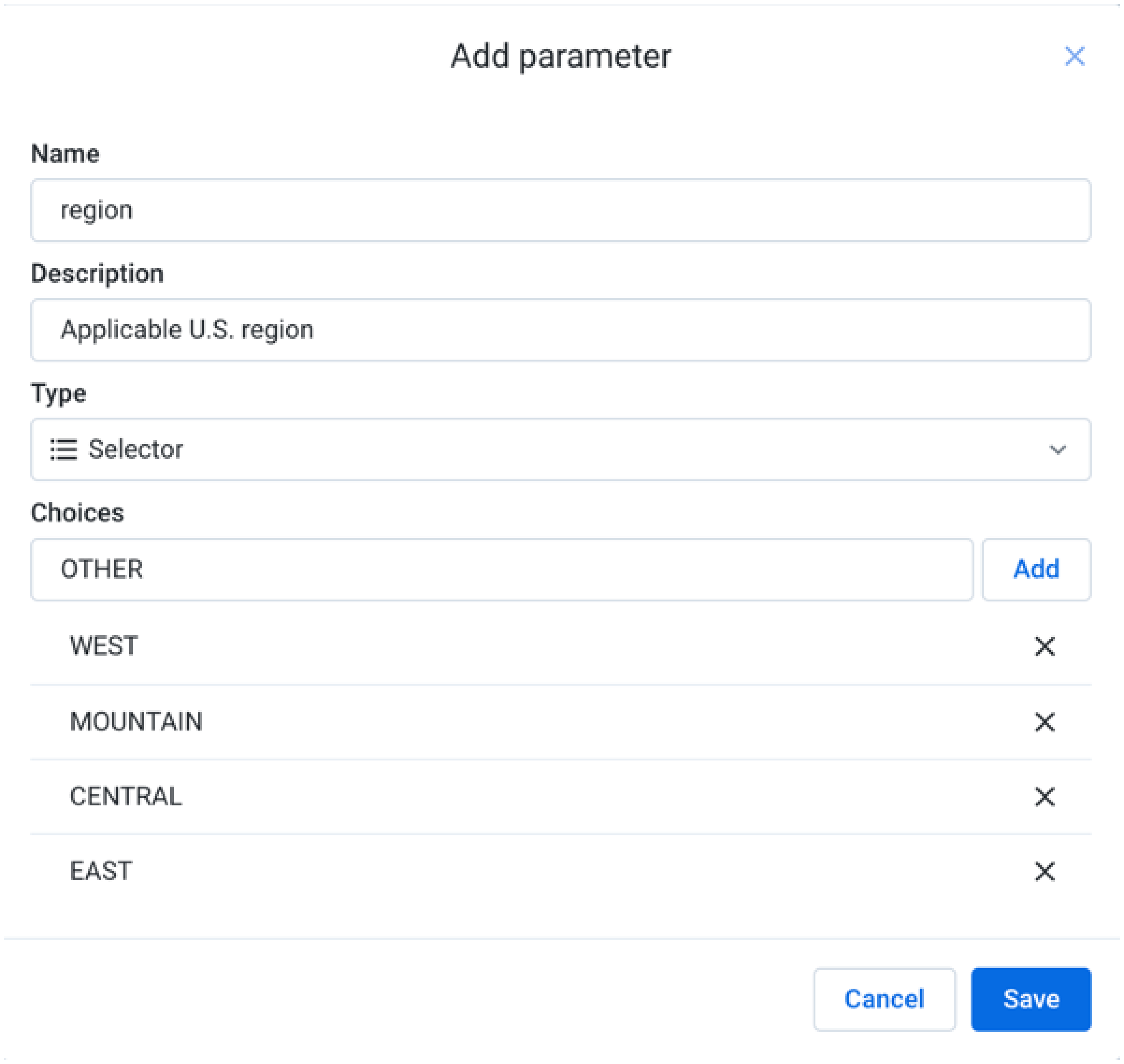
Regular expressions are a standard method for identifying patterns in data. The syntax is based on *RE2* and *PCRE* regular expressions.

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*.

Selector type parameters



Add parameter

Name: region

Description: Applicable U.S. region

Type: Selector

Choices: OTHER

WEST

MOUNTAIN

CENTRAL

EAST

Figure: Specify a flow parameter of Selector type

- Specify the Name value and optional Description value.

> **NOTE:** Name values are case-sensitive. After saving a parameter, you cannot change its name.

- For type, select **Selector**.
- For Choices, type or paste the permitted values for your selector type parameter.

Tip: The first value that you add is the default value.

- Each value must be added separately.
- Click the X next to a value to remove it from the permitted list.

After you create a parameter, you can insert it into your recipes using the following type of reference and then selecting a value from the list of permitted values:

```
${myRecipeParameterName} == <selected_value>
```

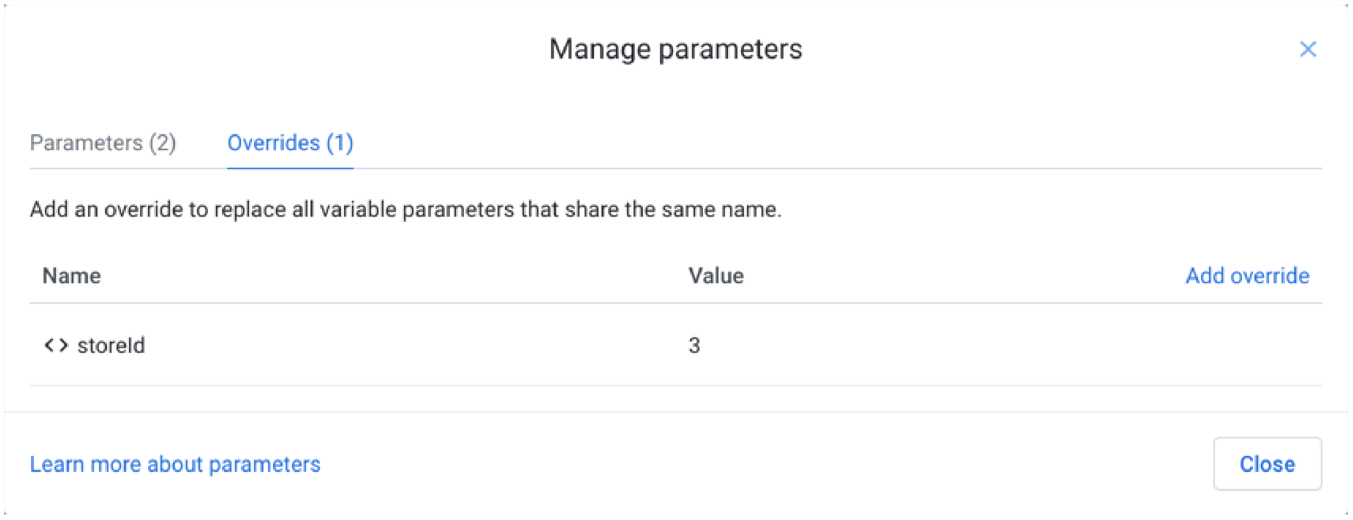
For more information, see *Create Flow Parameter*.

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.



Add an override to replace all variable parameters that share the same name.

Name	Value
<> storeId	3

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.

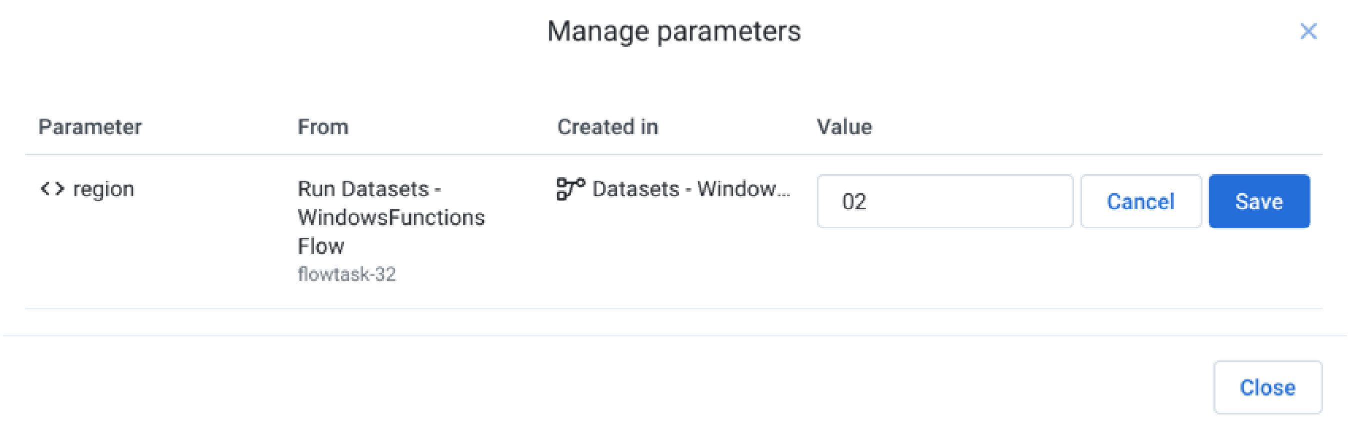
  NOTE: Parameters that are not of flow parameter type cannot be edited in the Parameters tab. You can apply override values to these parameters through the Overrides tab.

- 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.

Feature Availability: This feature may not be available in all product editions. For more information on available features, see *Compare Editions*.



Parameter	From	Created in	Value
region	Run Datasets - WindowsFunctions Flow flowtask-32	Datasets - Window...	02

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

Manage Parameters for Plans

See *Manage Parameters Dialog for Plans*.

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.

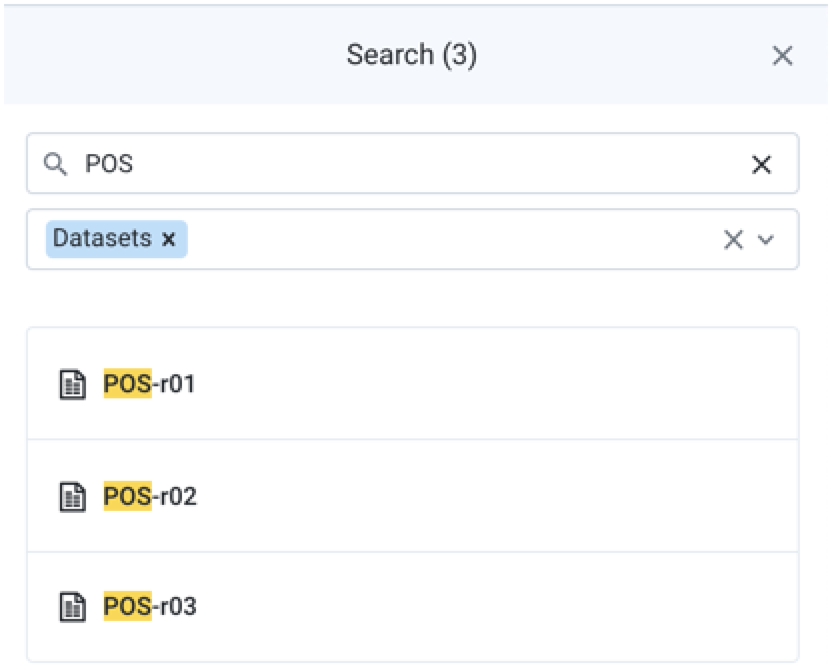


Figure: Flow Search panel

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 highlighted 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:

Feature Availability: This feature may not be available in all product editions. For more information on available features, see *Compare Editions*.

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.

Tip: In general, all of these optimizations should be enabled for each flow. As needed, you can selectively disable optimizations if you are troubleshooting execution issues.

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.

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.

Other optimizations

Additional optimizations can be enabled or disabled for specific types of transformations or jobs.

File-Based Optimizations

Full execution for S3 files

When enabled, jobs for this flow that are sourced from files stored in S3 can be executed in Snowflake .

NOTE: For execution of S3 jobs in Snowflake , AWS credentials are passed in encrypted format as part of the SQL that is executed within Snowflake .

NOTE: Additional limitations and requirements may apply for file-based job execution.

For more information, see *Snowflake Running Environment*.

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*.

Sample pushdown

When this setting is enabled, the Designer Cloud application optimizes job performance by executing sampling jobs directly on the source database.

> **NOTE**: All pushdowns must be enabled to ensure sample jobs run in the database.

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.

Transformer Page

Contents:

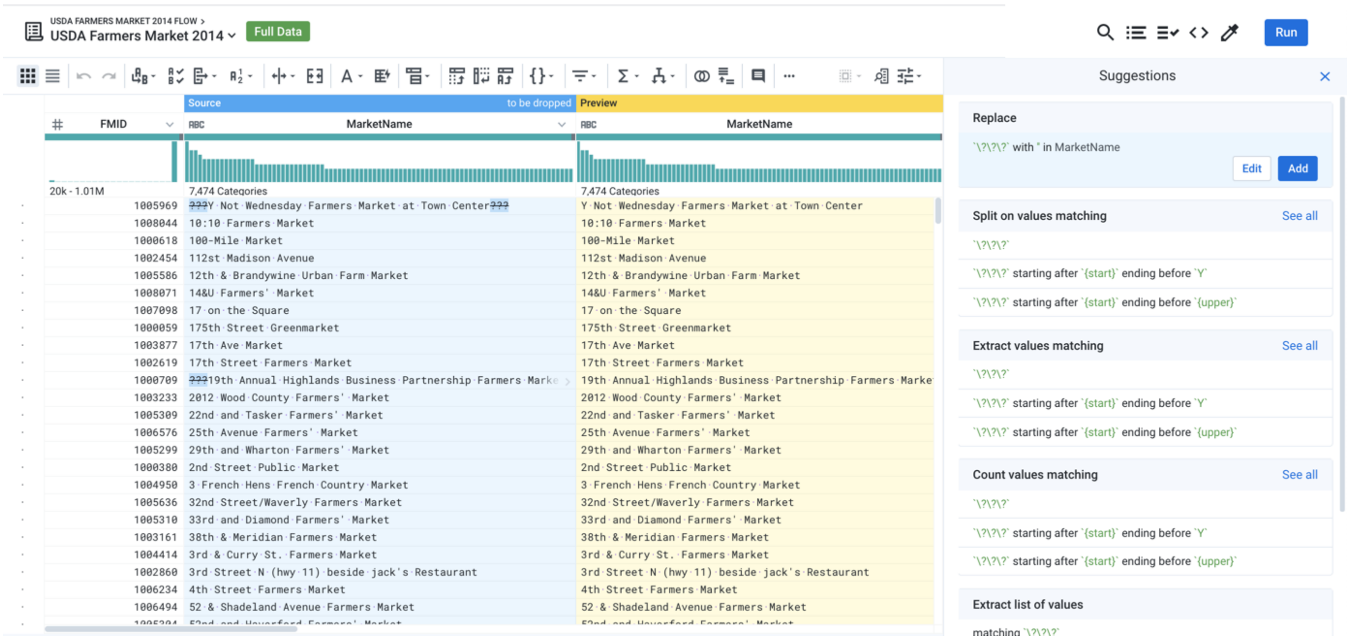
In the Transformer page, you identify the data that you need to transform and build your transformation recipes on samples taken from your currently selected dataset.

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 **head 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*.

Data not displayed: From Flow View you can choose to edit your recipe without loading a sample for faster editing. You can select a different sample with the data grid off. If needed, click **Enable data grid** to load the currently active sample.

> **NOTE:** This feature can be enabled or disabled by an administrator. For more information, see *Workspace Settings Page*.



FMID	MarketName (Source)	MarketName (Preview)
1005969	~~???~~Y Not Wednesday Farmers Market at Town Center~~???~~	Y Not Wednesday Farmers Market at Town Center
1008044	10:10 Farmers Market	10:10 Farmers Market
1000618	100-Mile Market	100-Mile Market
1002454	112st Madison Avenue	112st Madison Avenue
1005586	12th & Brandywine Urban Farm Market	12th & Brandywine Urban Farm Market
1008071	14&U Farmers' Market	14&U Farmers' Market
1007098	17 on the Square	17 on the Square
1000059	175th Street Greenmarket	175th Street Greenmarket
1003877	17th Ave Market	17th Ave Market
1002619	17th Street Farmers Market	17th Street Farmers Market
1000709	~~???~~19th Annual Highlands Business Partnership Farmers Marke	19th Annual Highlands Business Partnership Farmers Marke
1003233	2012 Wood County Farmers' Market	2012 Wood County Farmers' Market
1005309	22nd and Tasker Farmers' Market	22nd and Tasker Farmers' Market
1006576	25th Avenue Farmers' Market	25th Avenue Farmers' Market
1005299	29th and Wharton Farmers' Market	29th and Wharton Farmers' Market
1000380	2nd Street Public Market	2nd Street Public Market
1004950	3 French Hens French Country Market	3 French Hens French Country Market
1005636	32nd Street/Waverly Farmers Market	32nd Street/Waverly Farmers Market
1005310	33rd and Diamond Farmers' Market	33rd and Diamond Farmers' Market
1003161	38th & Meridian Farmers Market	38th & Meridian Farmers Market
1004414	3rd & Curry St. Farmers Market	3rd & Curry St. Farmers Market
1002860	3rd Street N (hwy 11) beside jack's Restaurant	3rd Street N (hwy 11) beside jack's Restaurant
1006234	4th Street Farmers Market	4th Street Farmers Market
1006494	52 & Shadeland Avenue Farmers Market	52 & Shadeland Avenue Farmers Market

Suggestions

Replace
`\?\?\?` with '' in MarketName
Edit Add

Split on values matching See all
- `\?\?\?`
- `\?\?\?` starting after `{start}` ending before `Y`
- `\?\?\?` starting after `{start}` ending before `{upper}`

Extract values matching See all
- `\?\?\?`
- `\?\?\?` starting after `{start}` ending before `Y`
- `\?\?\?` starting after `{start}` ending before `{upper}`

Count values matching See all
- `\?\?\?`
- `\?\?\?` starting after `{start}` ending before `Y`
- `\?\?\?` starting after `{start}` ending before `{upper}`

Extract list of values
matching `\?\?\?`

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
Transformer Toolbar	A toolbar of common transformations, filters, and other operations.	Use the tools in these drop-downs to quickly build common recipe steps.
Data Grid	By default, the Transformer page displays previews in a columnar grid, which is the default view. Click **Grid**.	Use for examining values in a column with appropriate surrounding context. How do missing values in one column compare to values in another column?
Column Details	For additional statistical information on individual columns, select **Column Details** 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.
Column Browser	Use the Column Browser to select the columns to display and review data across columns. Click **Columns**.	Navigate between columns and toggle their display in the data grid. Good for high-level perspective. Use histograms for selection of ranges of values.
Context Panel	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 Indicator:** A t 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. Click this name to display statistics about the currently loaded sample. See *Sample Indicator*.
- **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. For more information, see *Data Grid 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.

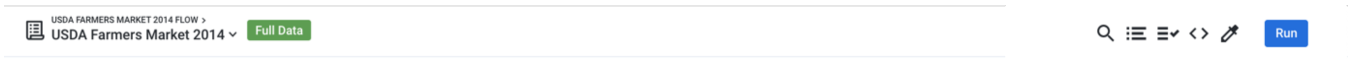


Figure: Transformer page toolbar

- **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.
- **Recipe icon:** Display the current recipe. See

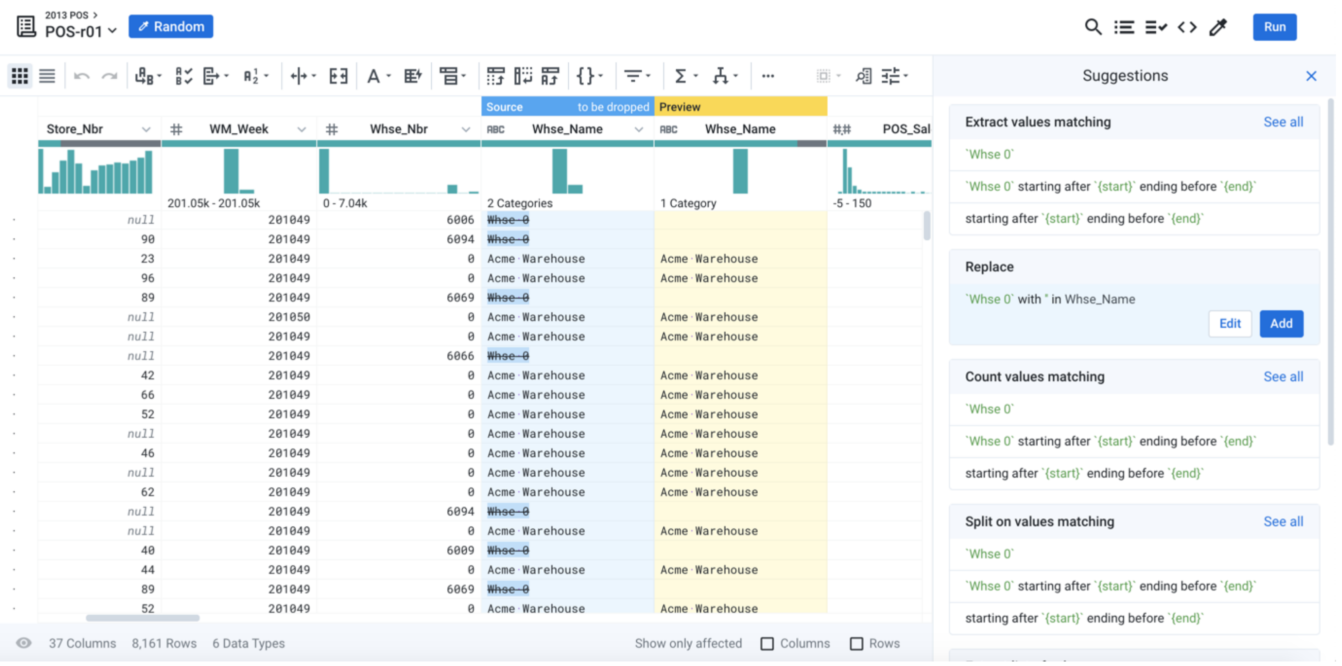
- **Flow Parameters icon:** Review, create, and modify the parameters of your flow. See *Manage Parameters Dialog*.
- **Samples icon:** Click the dropper icon to review and create new samples. See *Samples Panel*.
- **Run:** Runs the currently specified recipe on the dataset. See *Run Job Page*.

Data Grid Panel

Contents:

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.



Store_Nbr	WM_Week	Whse_Nbr	Whse_Name (Source)	Whse_Name (Preview)	POS_Sal
	201.05k - 201.05k	0 - 7.04k	2 Categories	1 Category	-5 - 150
null	201049	6006	~~Whse 0~~		
90	201049	6094	~~Whse 0~~		
23	201049	0	Acme Warehouse	Acme Warehouse	
96	201049	0	Acme Warehouse	Acme Warehouse	
89	201049	6069	~~Whse 0~~		
null	201050	0	Acme Warehouse	Acme Warehouse	
null	201049	0	Acme Warehouse	Acme Warehouse	
null	201049	6066	~~Whse 0~~		
42	201049	0	Acme Warehouse	Acme Warehouse	
66	201049	0	Acme Warehouse	Acme Warehouse	
52	201049	0	Acme Warehouse	Acme Warehouse	
null	201049	0	Acme Warehouse	Acme Warehouse	
46	201049	0	Acme Warehouse	Acme Warehouse	
null	201049	0	Acme Warehouse	Acme Warehouse	
62	201049	0	Acme Warehouse	Acme Warehouse	
null	201049	6094	~~Whse 0~~		
null	201049	0	Acme Warehouse	Acme Warehouse	
40	201049	6009	~~Whse 0~~		
44	201049	0	Acme Warehouse	Acme Warehouse	
89	201049	6069	~~Whse 0~~		
52	201049	0	Acme Warehouse	Acme Warehouse	

37 Columns 8,161 Rows 6 Data Types   Show only affected ☐ Columns ☐ Rows

Extract values matching See all
- `Whse 0`
- `Whse 0` starting after `{start}` ending before `{end}`
- starting after `{start}` ending before `{end}`

Replace
- `Whse 0` with '' in Whse_Name Edit Add

Count values matching See all
- `Whse 0`
- `Whse 0` starting after `{start}` ending before `{end}`
- starting after `{start}` ending before `{end}`

Split on values matching See all
- `Whse 0`
- `Whse 0` starting after `{start}` ending before `{end}`
- starting after `{start}` ending before `{end}`

Figure: Data Grid Panel

Data not displayed: From Flow View you can choose to edit your recipe without loading a sample for faster editing. You can select a different sample with the data grid off. If needed, click **Enable data grid** to load the currently active sample.

> **NOTE:** This feature can be enabled or disabled by an administrator.
>
> For more information, see *Workspace Settings Page*.

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 **Sort A Z**. 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. 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.

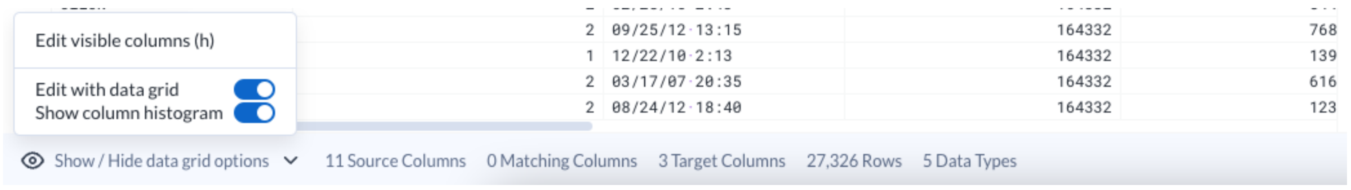


Figure: Sample Status bar

- **Show/Hide Data grid options**: Click the drop-down to:
    - **Edit visible columns**: Show or hide the visible columns. For more information, see *Visible Columns Panel*.
    - **Edit with data grid**: You can toggle updating of the data grid based on the changes to your recipe. When disabled, the data grid is frozen. You can also enable the data grid by using the reload icon available.

      > **NOTE:** Some transformations cannot be added or modified when the data grid is disabled.

      This feature can be enabled or disabled by an administrator. For more information, see *Workspace Settings Page*.

    - **Show column histogram**: You can toggle to show or hide the histograms over each column in the data grid.

      > **NOTE**: Histograms cannot be toggled when the data grid is disabled.

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.

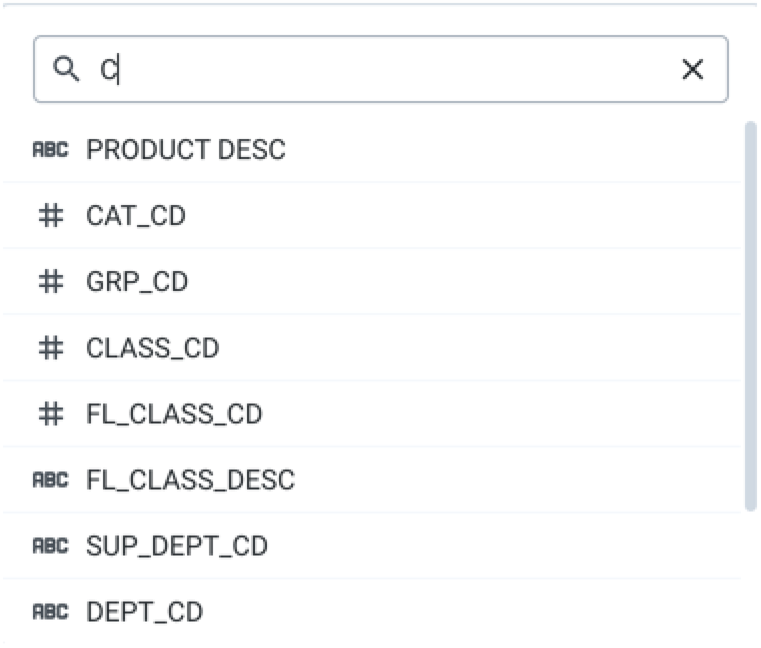


Figure: Find column search bar

- 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

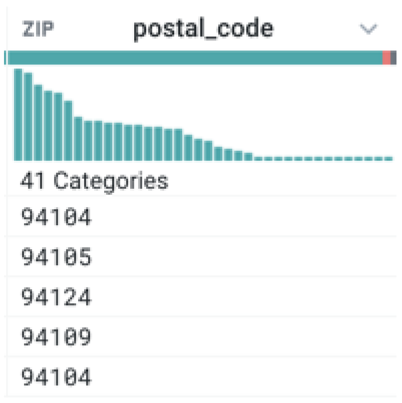


Figure: Column header, data quality bar, and histogram

- 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. **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*. See *Supported Data Types*.
Column name	To change the column name, select **Rename...** 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 *Column Menus*.
Data quality bar	The horizontal line shows valid, missing, and mismatched values in the column compared to the column's data type. **Tip:** You can click these colored bars to generate suggestion cards for transformations to act on these types of values. See *Data Quality Bars*.
Column histogram	For each column, you can see the range and frequency of values in the column. **Tip:** You can select one or more values a histogram to generate suggestion cards. See *Column Histograms*.

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*.

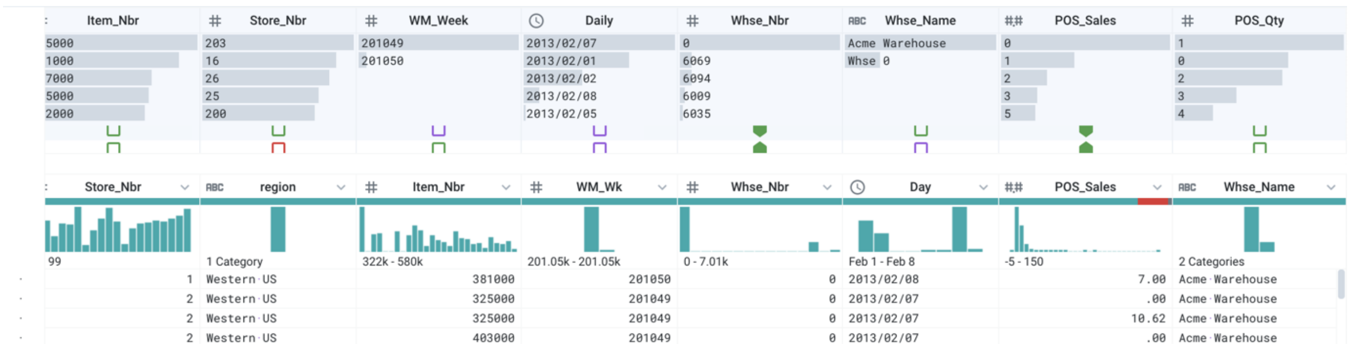
Transformation Preview

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*.



Item_Nbr	Store_Nbr	WM_Week	Daily	Whse_Nbr	Whse_Name	POS_Sales	POS_Qty
5000	203	201049	2013/02/07	0	Acme Warehouse	0	1
1000	16	201050	2013/02/01	6069	Whse 0	1	0
7000	26		2013/02/02	6094		2	2
5000	25		2013/02/08	6009		3	3
2000	200		2013/02/05	6035		5	4

Store_Nbr	region	Item_Nbr	WM_Wk	Whse_Nbr	Day	POS_Sales	Whse_Name
99	1 Category	322k - 580k	201.05k - 201.05k	0 - 7.01k	Feb 1 - Feb 8	-5 - 150	2 Categories
1	Western US	381000	201050	0	2013/02/08	7.00	Acme Warehouse
2	Western US	325000	201049	0	2013/02/07	.00	Acme Warehouse
2	Western US	325000	201049	0	2013/02/07	10.62	Acme Warehouse
2	Western US	403000	201049	0	2013/02/07	.00	Acme Warehouse

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.

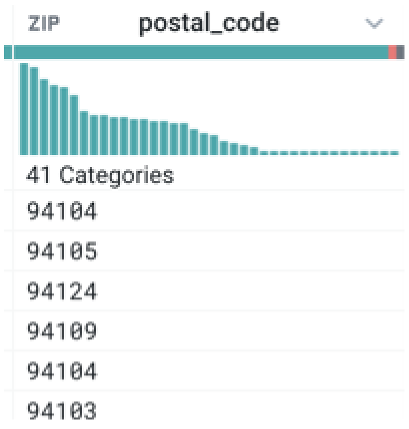


Figure: Column Histogram

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.

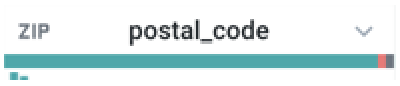


Figure: Data quality bars

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. For more information, see *Find Bad Data*.
Gray	Missing	Value is empty or null. For more information, see *Manage Null Values*.

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 the ROUND function 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.

For more information, see *Supported Data Types*.

Lookup Wizard

Contents:

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.

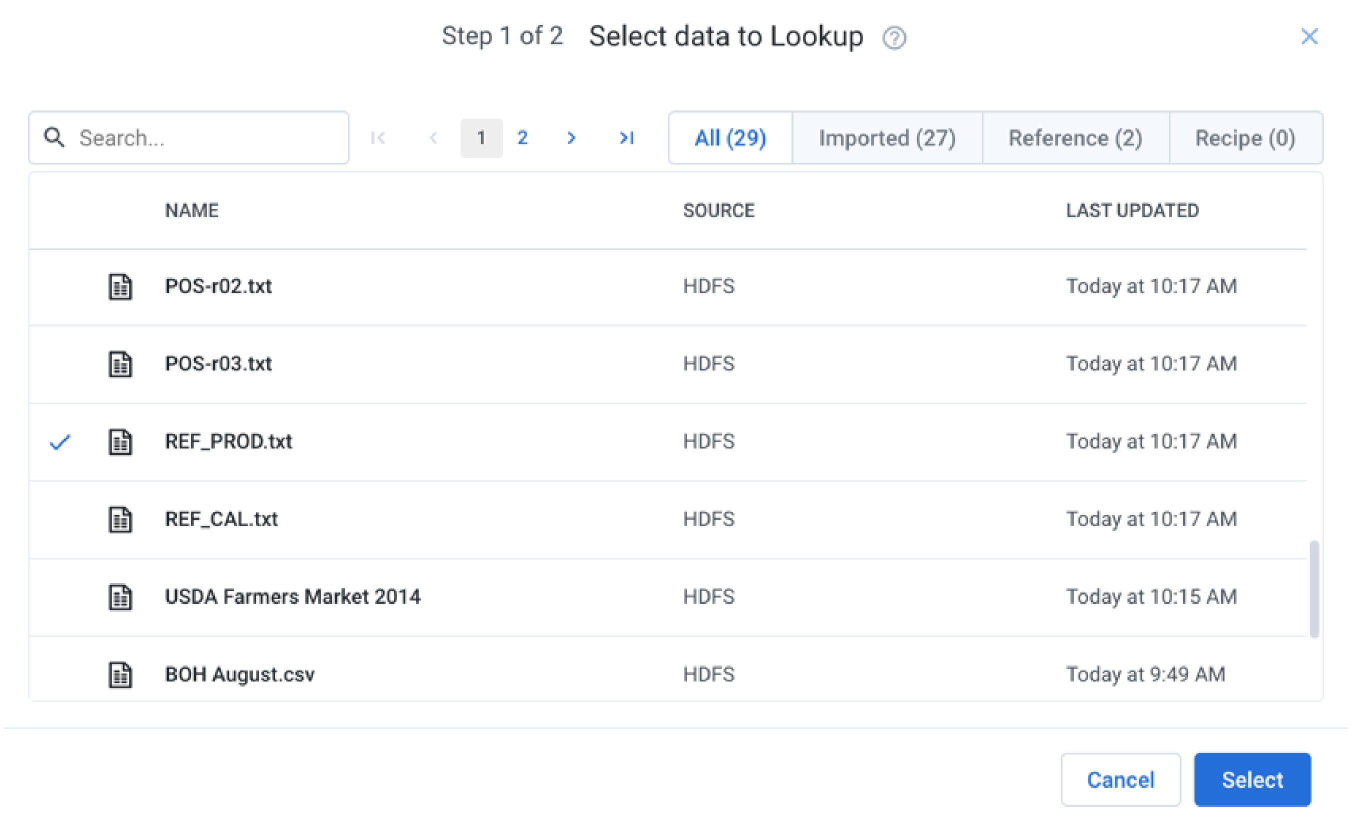
  > **NOTE:** If column values are non-unique, the resulting dataset can be significantly larger than the original dataset.

This task 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.

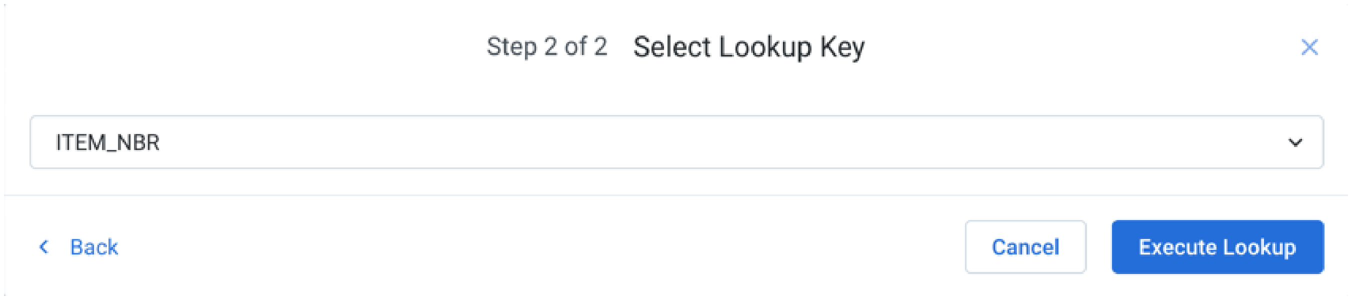


	NAME	SOURCE	LAST UPDATED
	POS-r02.txt	HDFS	Today at 10:17 AM
	POS-r03.txt	HDFS	Today at 10:17 AM
✓	REF_PROD.txt	HDFS	Today at 10:17 AM
	REF_CAL.txt	HDFS	Today at 10:17 AM
	USDA Farmers Market 2014	HDFS	Today at 10:15 AM
	BOH August.csv	HDFS	Today at 9:49 AM

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`.

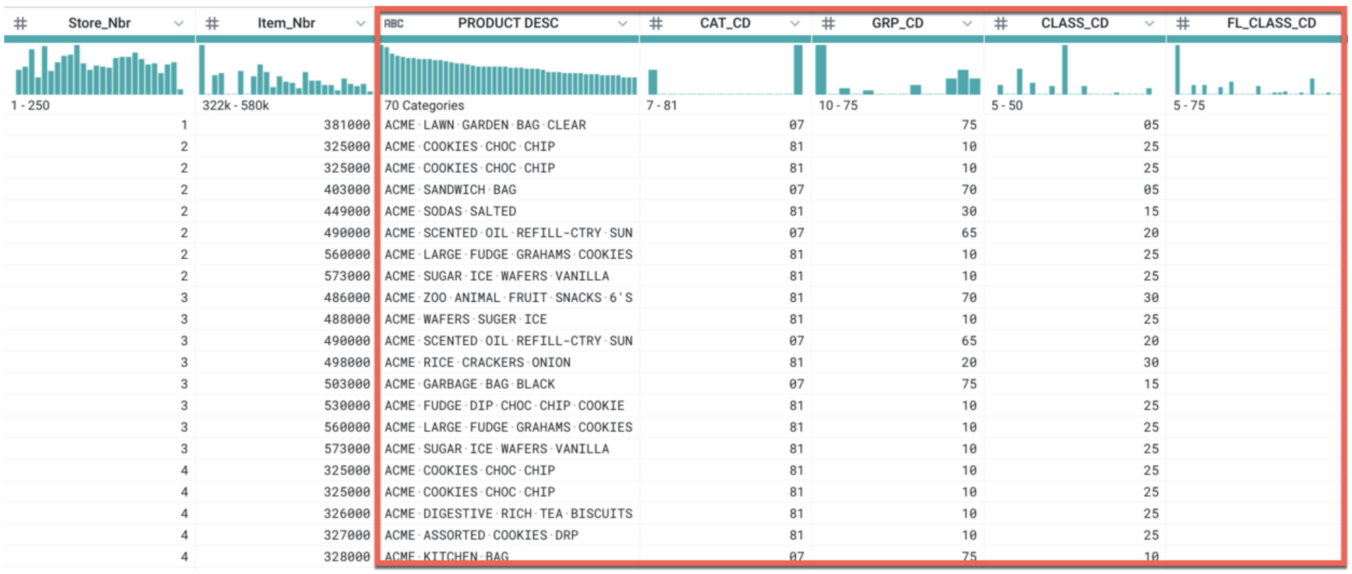


ITEM_NBR

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	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*.

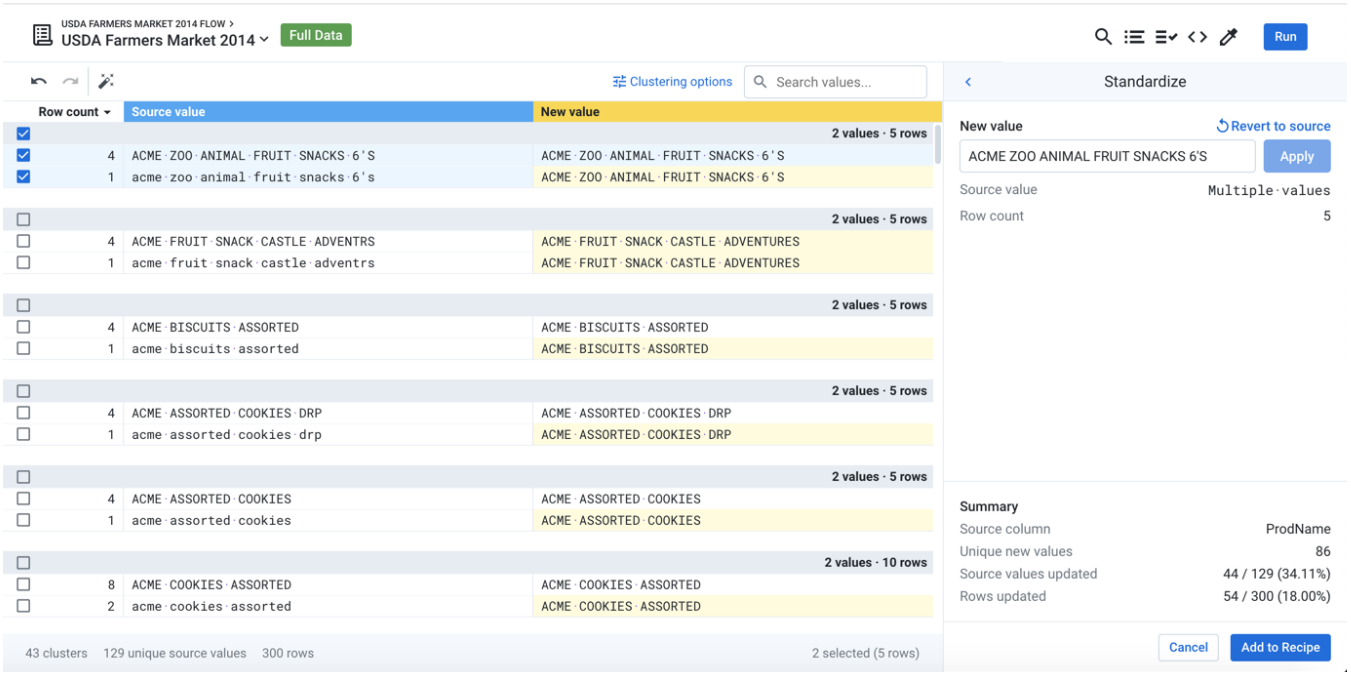
Limitations:

- 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*.



	Row count	Source value	New value
✓			2 values · 5 rows
✓	4	ACME·ZOO·ANIMAL·FRUIT·SNACKS·6'S	ACME·ZOO·ANIMAL·FRUIT·SNACKS·6'S
✓	1	acme·zoo·animal·fruit·snacks·6's	ACME·ZOO·ANIMAL·FRUIT·SNACKS·6'S
☐			2 values · 5 rows
☐	4	ACME·FRUIT·SNACK·CASTLE·ADVENTRS	ACME·FRUIT·SNACK·CASTLE·ADVENTURES
☐	1	acme·fruit·snack·castle·adventrs	ACME·FRUIT·SNACK·CASTLE·ADVENTURES
☐			2 values · 5 rows
☐	4	ACME·BISCUITS·ASSORTED	ACME·BISCUITS·ASSORTED
☐	1	acme·biscuits·assorted	ACME·BISCUITS·ASSORTED
☐			2 values · 5 rows
☐	4	ACME·ASSORTED·COOKIES·DRP	ACME·ASSORTED·COOKIES·DRP
☐	1	acme·assorted·cookies·drp	ACME·ASSORTED·COOKIES·DRP
☐			2 values · 5 rows
☐	4	ACME·ASSORTED·COOKIES	ACME·ASSORTED·COOKIES
☐	1	acme·assorted·cookies	ACME·ASSORTED·COOKIES
☐			2 values · 10 rows
☐	8	ACME·COOKIES·ASSORTED	ACME·COOKIES·ASSORTED
☐	2	acme·cookies·assorted	ACME·COOKIES·ASSORTED

43 clusters   129 unique source values   300 rows   2 selected (5 rows)

New value   Revert to source
ACME ZOO ANIMAL FRUIT SNACKS 6'S   Apply

Source value	Multiple·values
Row count	5

Summary

Source column	ProdName
Unique new values	86
Source values updated	44 / 129 (34.11%)
Rows updated	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

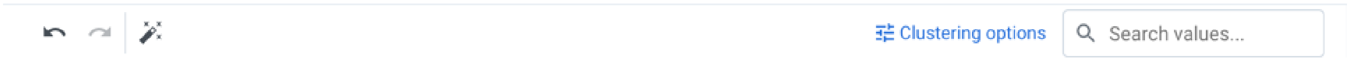


Figure: Standardize toolbar

- **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.

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*.

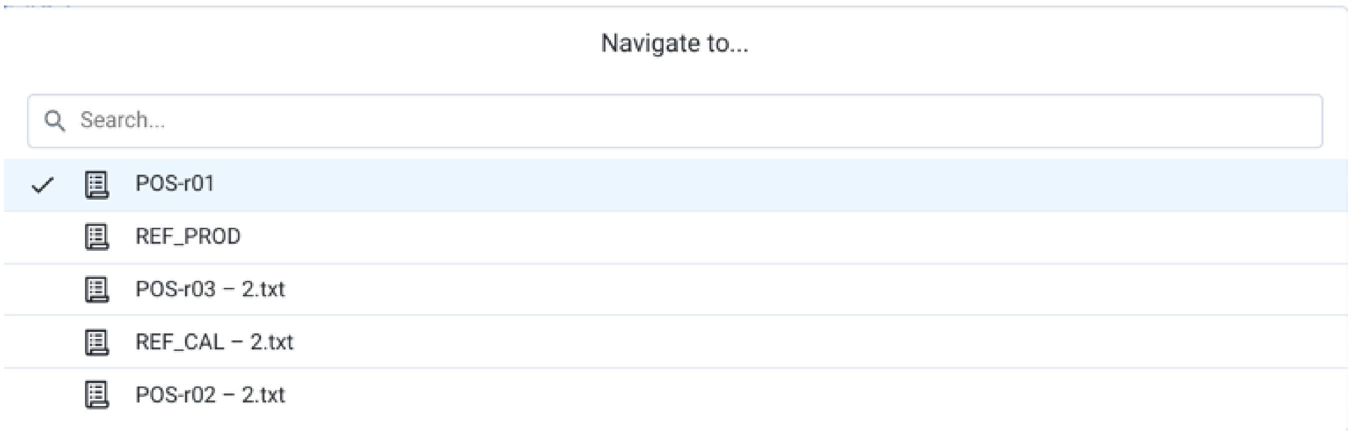
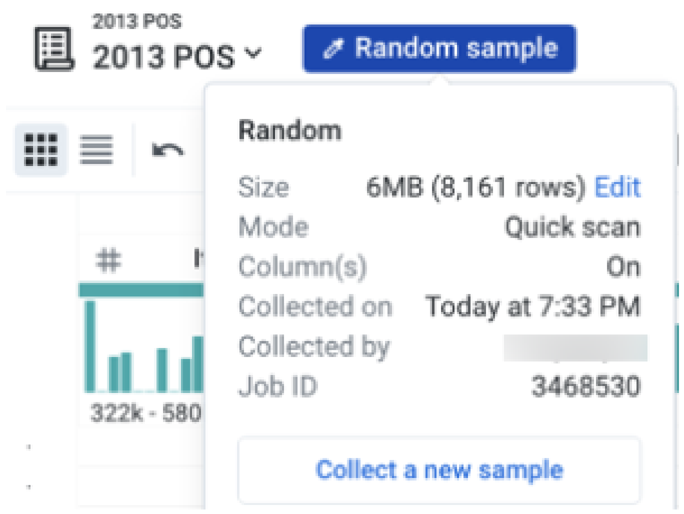


Figure: Recipe tab

To locate a recipe to load, use the Search box or browse the list. Then, select the recipe to load.

Sample Indicator

At the top of the Transformer page, you can see the Sample Indicator showing the name of the currently displayed sample. If you have not taken a sample, this value is **Initial data**. Click this button to show more information about the sample.



Random

Size	6MB (8,161 rows) Edit
Mode	Quick scan
Column(s)	On
Collected on	Today at 7:33 PM
Collected by	[redacted]
Job ID	3468530

Figure: Sample Indicator

In the Sample Indicator, you can review the number of rows collected in the same and information about the user and time when the sample was collected.

- As needed, you can change the size of samples loaded into the browser for your current recipe, which may address performance issues. Click **Edit**. For more information, see *Change Recipe Sample Size*.
- To generate a new sample, click **Collect a new sample**. The Samples panel is displayed. For more information, see *Samples Panel*.

Transformer Toolbar

Contents:

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.

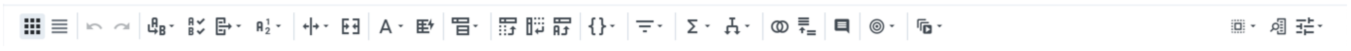


Figure: Transformer Toolbar

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 Target Schema Mapping*.

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:

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*.

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*.

Automatically update

Unlock the column's data type. For more information, see *Change Column Data Type*.

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.

For more information, see *Sort Rows*.

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.

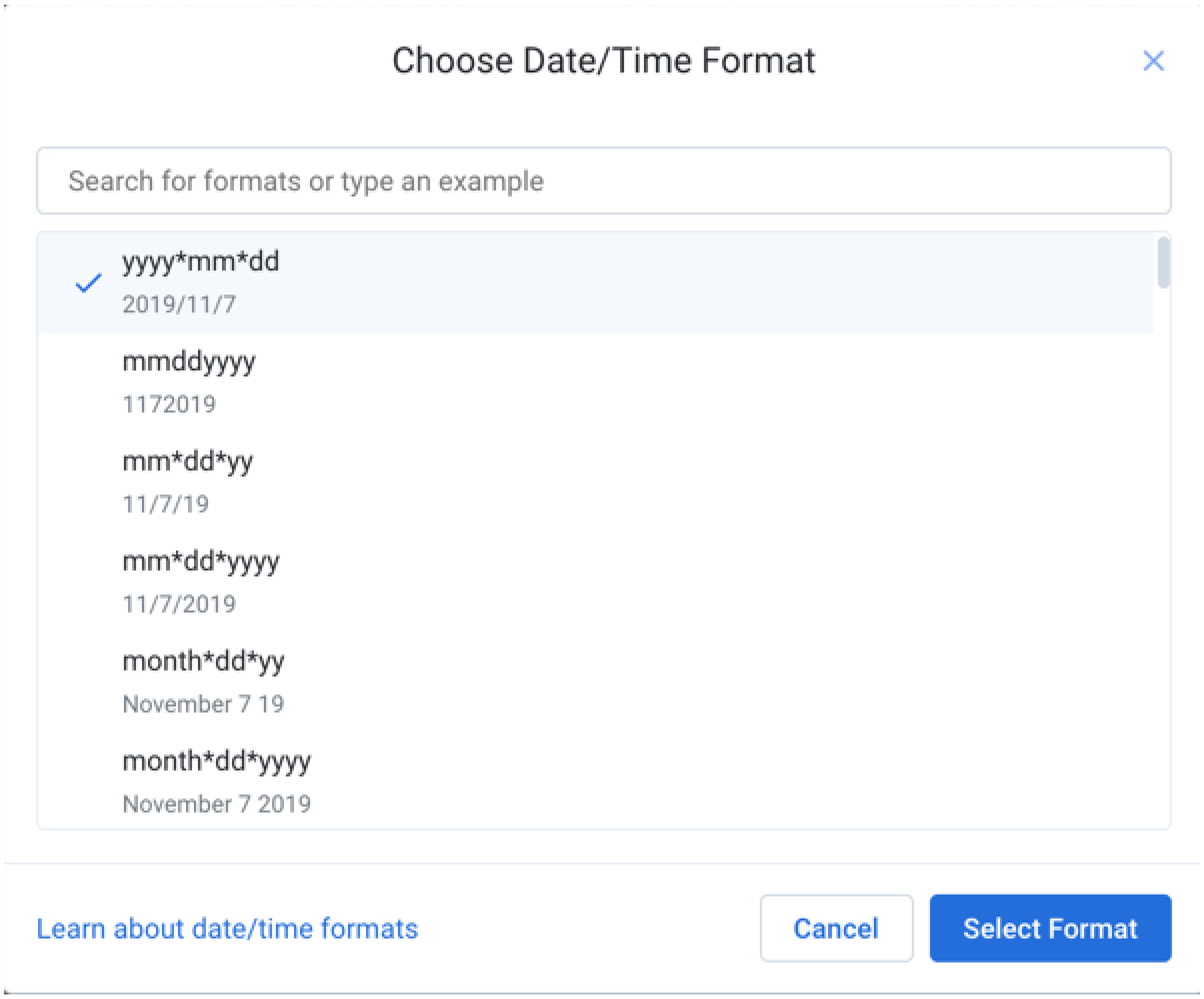
Copy, Cut, and Paste

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

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:

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*.

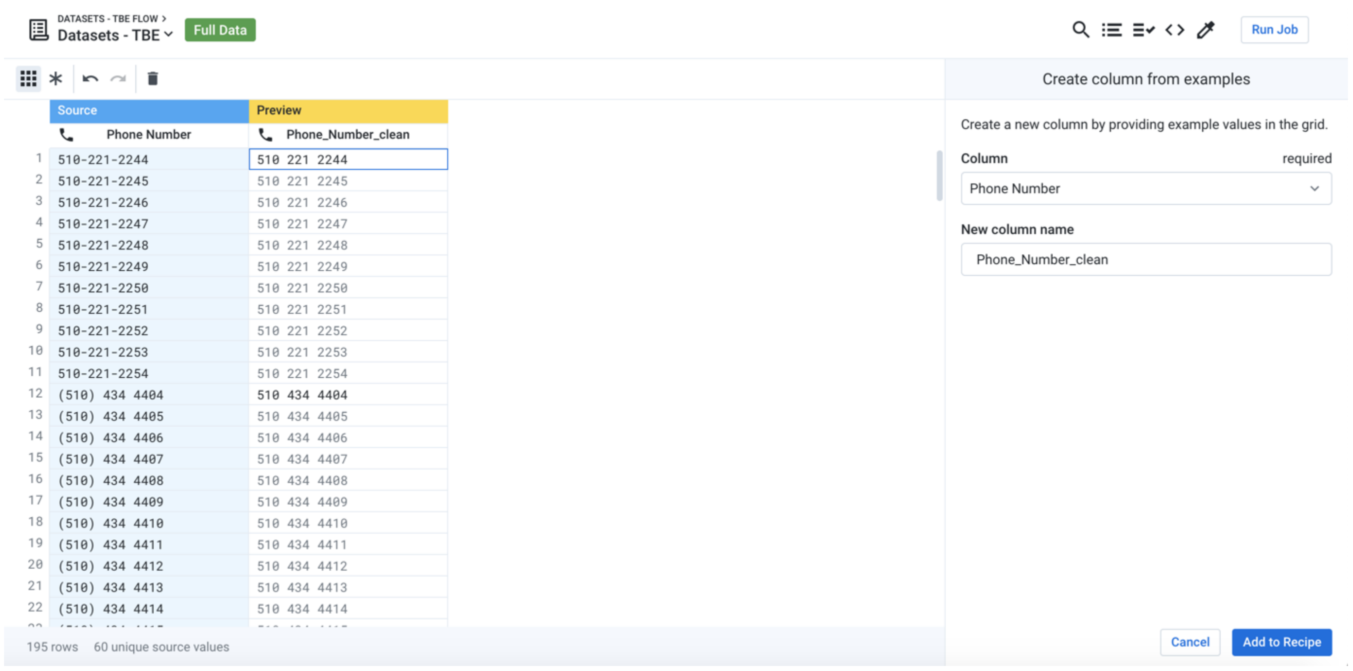
Transform Builder

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.



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

	Source: Phone Number	Preview: 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

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*.

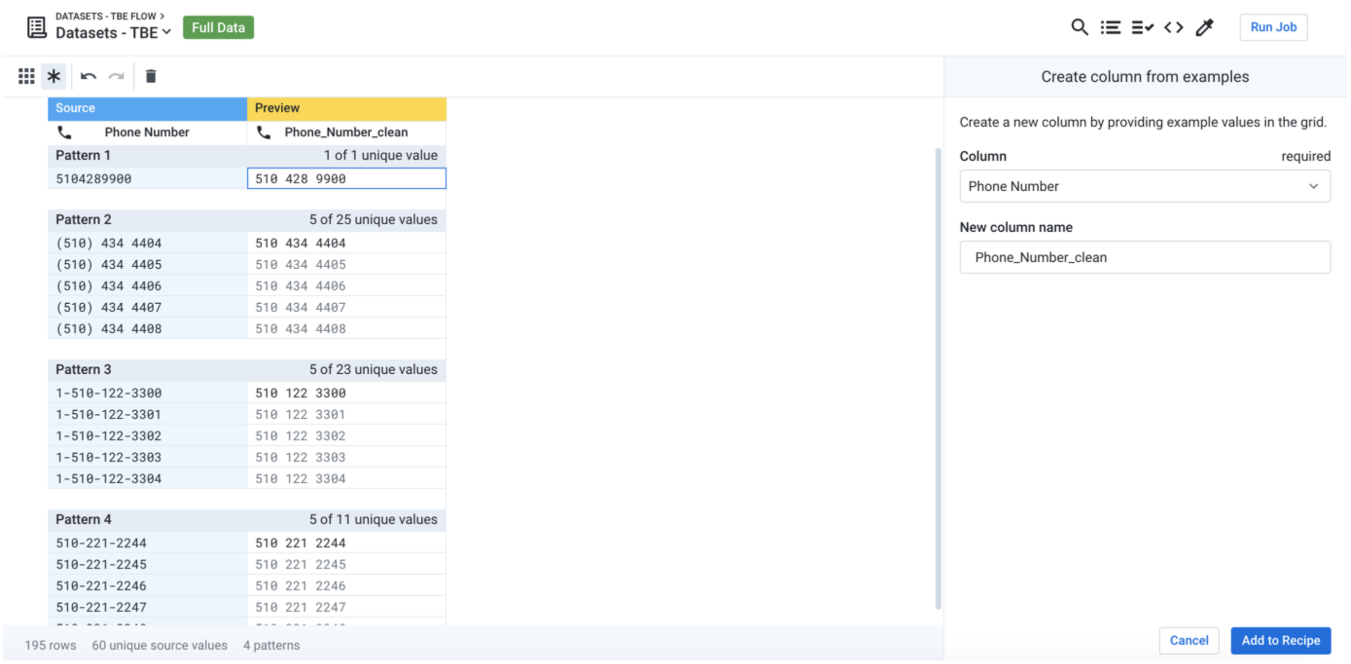
Keyboard shortcuts

- 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 >
Datasets - TBE
Full Data
Run Job

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: Phone Number (required)

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

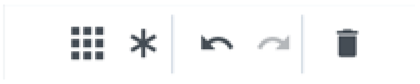


Figure: Transformation by Example toolbar

- **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:

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.

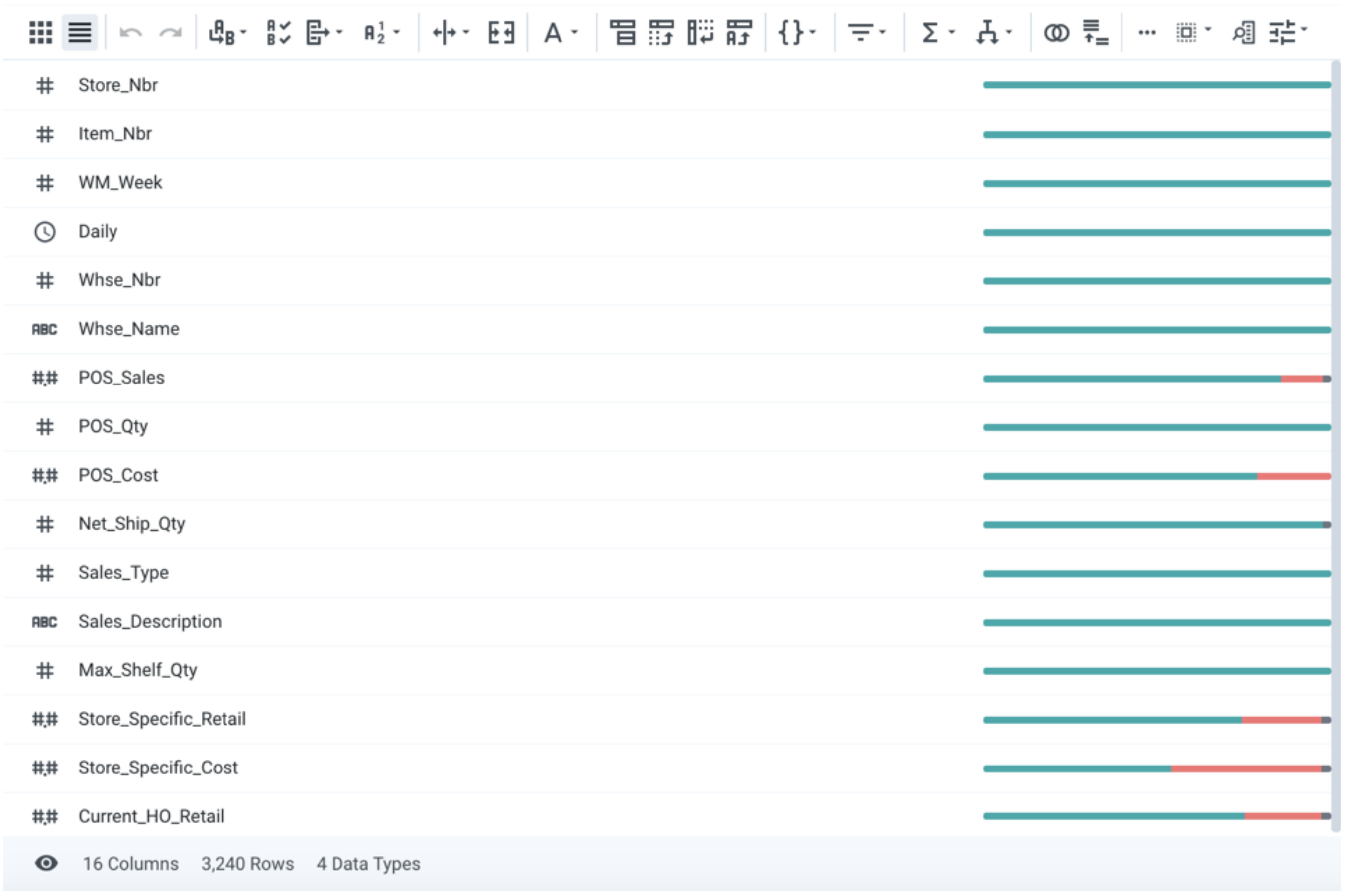


Figure: Column Browser

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*.

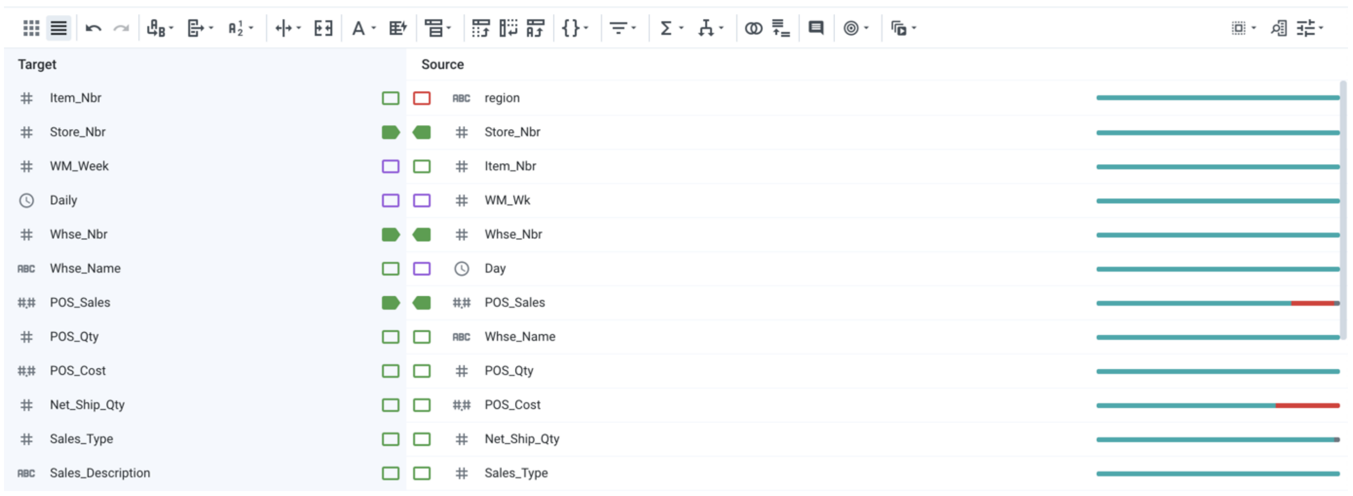
  NOTE: Hidden columns are only removed from view in the Transformer page. They still appear in any generated output.

- 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*.

Target Schema Mapping

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 Target Schema Mapping*.

Actions taken based on the target schema are rendered as a new step in the location in your recipe.



Target	Source
Item_Nbr	region
Store_Nbr	Store_Nbr
WM_Week	Item_Nbr
Daily	WM_Wk
Whse_Nbr	Whse_Nbr
Whse_Name	Day
POS_Sales	POS_Sales
POS_Qty	Whse_Name
POS_Cost	POS_Qty
Net_Ship_Qty	POS_Cost
Sales_Type	Net_Ship_Qty
Sales_Description	Sales_Type

Figure: Target Schema Mapping Panel in Column Browser

In the Target Schema Mapping panel, the columns of the target schema assigned to your recipe are displayed on the left side. Columns are listed side-by-side in order:

Side	Description
Left	Target column
Right	Source column (current recipe)

Schema Tags:

Between the two sets of fields are a set of tags that describe the matching between the two columns in the order listed in each dataset. Each tag describes the match found for the corresponding target or source column.

- Matches by position are arrows.
- Matches by data type are green in color.
- Partial name matches are indicated by the color purple. In the cases, the platform has found a match when column names are inexact matches.
- Red color indicates no match at all.

Tip: Mouse over tags that are not solid green to learn the nature of the column mismatch. You can then click the tag to match the two columns.

Match by Name	Match by Data Type	Match by Position	Icon	Description
No	-	-	Square - red outline	Target column has no identifiable name match in source. Column appears to be missing from source. **Tip:** Click the source or target rectangle to manually assign a matching column.
Yes	No	No	Square - blue outline	Source column name has a matching target column name, but it is in the incorrect location.
Partial	No	No	Square - purple outline	Source column name has a partial match with a target column name, but it is in a different location.
Yes	Yes	No	Square - green outline	Source column name and data type have a match in the target, but it is in a different position.

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 Target Schema Mapping*.

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 Target Schema Mapping*.

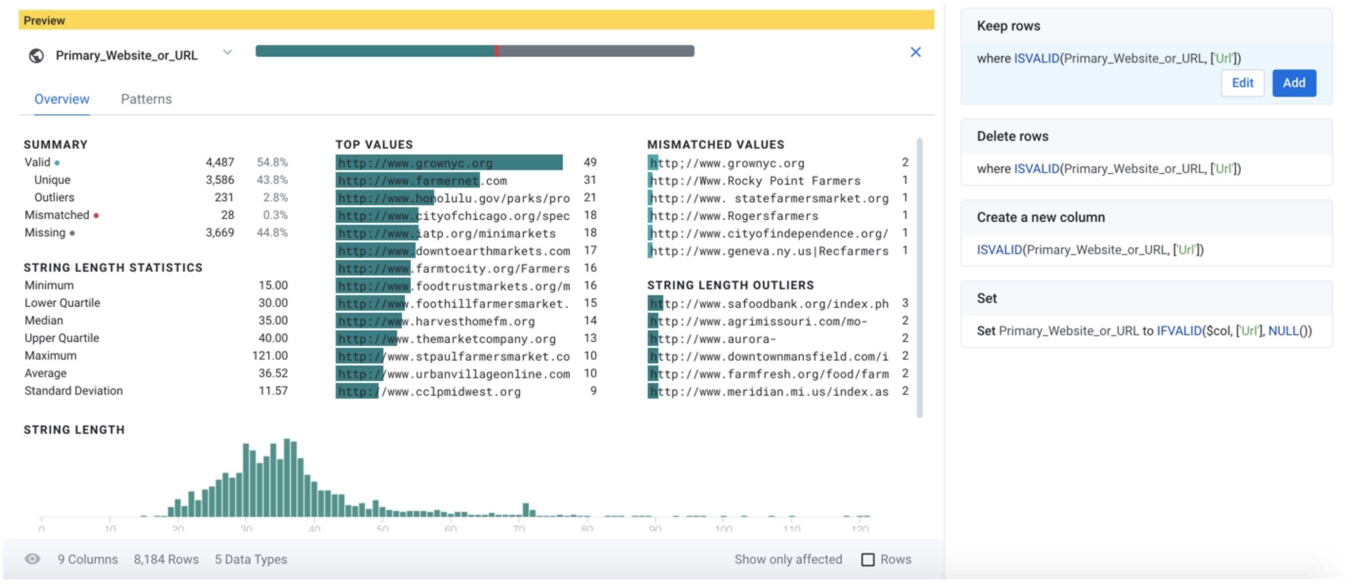
Column Details Panel

Contents:

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



Preview
Primary_Website_or_URL
Overview
Patterns
STRING LENGTH
9 Columns
8,184 Rows
5 Data Types
Show only affected
Rows
Keep rows
where ISVALID(Primary_Website_or_URL, ['Url'])
Edit
Add
Delete rows
where ISVALID(Primary_Website_or_URL, ['Url'])
Create a new column
ISVALID(Primary_Website_or_URL, ['Url'])
Set
Set Primary_Website_or_URL to IFVALID($col, ['Url'], NULL())

SUMMARY

Valid ●	4,487	54.8%
Unique	3,586	43.8%
Outliers	231	2.8%
Mismatched ●	28	0.3%
Missing ●	3,669	44.8%

STRING LENGTH STATISTICS

Minimum	15.00
Lower Quartile	30.00
Median	35.00
Upper Quartile	40.00
Maximum	121.00
Average	36.52
Standard Deviation	11.57

TOP VALUES

http://www.grownyc.org	49
http://www.farmernet.com	31
http://www.honolulu.gov/parks/pro	21
http://www.cityofchicago.org/spec	18
http://www.iatp.org/minimarkets	18
http://www.downtoearthmarkets.com	17
http://www.farmtocity.org/Farmers	16
http://www.foodtrustmarkets.org/m	16
http://www.foothillfarmersmarket.	15
http://www.harvesthomefm.org	14
http://www.themarketcompany.org	13
http://www.stpaulfarmersmarket.co	10
http://www.urbanvillageonline.com	10
http://www.cclpmidwest.org	9

MISMATCHED VALUES

http;//www.grownyc.org	2
http://Www.Rocky Point Farmers	1
http://www. statefarmersmarket.org	1
http://www.Rogersfarmers	1
http://www.cityofindependence.org/	1
http://www.geneva.ny.us|Recfarmers	1

STRING LENGTH OUTLIERS

http://www.safoodbank.org/index.ph	3
http://www.agrimissouri.com/mo-	2
http://www.aurora-	2
http://www.downtownmansfield.com/i	2
http://www.farmfresh.org/food/farm	2
http://www.meridian.mi.us/index.as	2

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*.

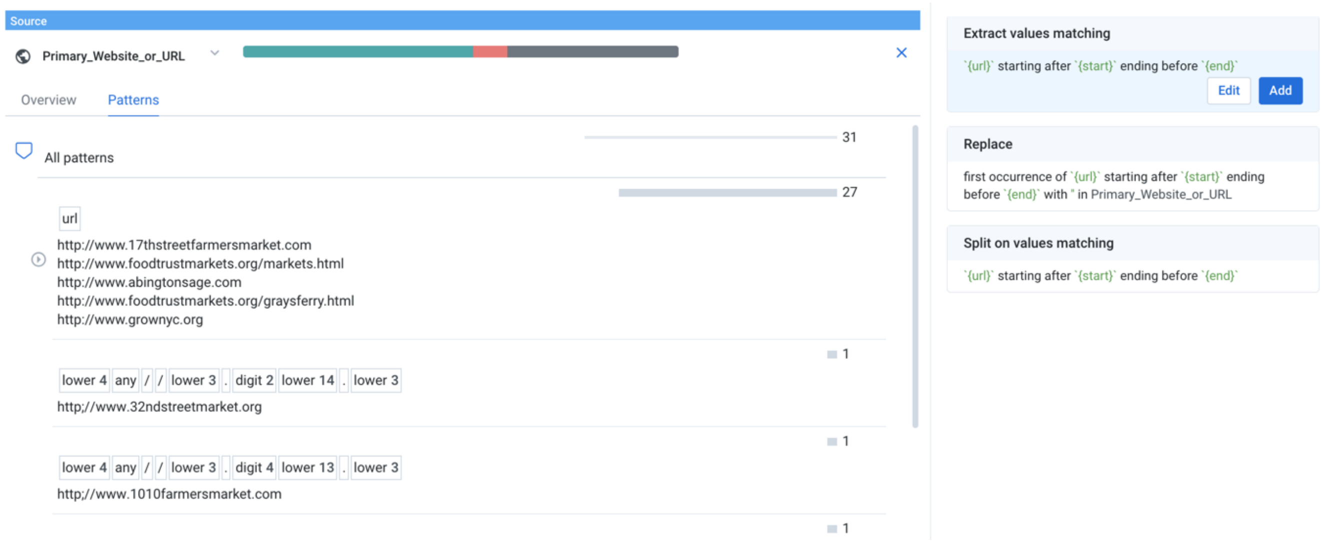
Patterns tab

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.



http://www.17thstreetfarmersmarket.com
http://www.foodtrustmarkets.org/markets.html
http://www.abingtonsage.com
http://www.foodtrustmarkets.org/graysferry.html
http://www.grownyc.org

lower 4 | any | / | / | lower 3 | . | digit 2 | lower 14 | . | lower 3
http;//www.32ndstreetmarket.org

lower 4 | any | / | / | lower 3 | . | digit 4 | lower 13 | . | lower 3
http;//www.1010farmersmarket.com

Extract values matching
`{url}` starting after `{start}` ending before `{end}`
Edit Add

Replace
first occurrence of `{url}` starting after `{start}` ending before `{end}` with '' in Primary_Website_or_URL

Split on values matching
`{url}` starting after `{start}` ending before `{end}`

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.

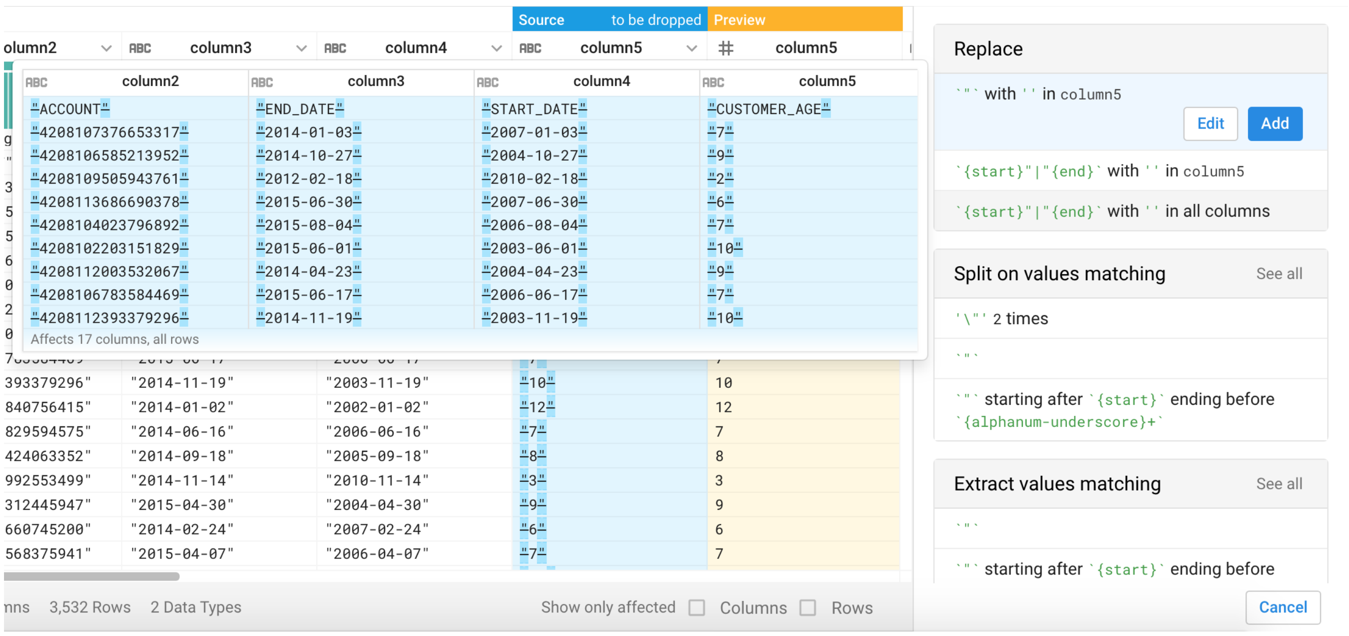
Transform Preview

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:



column2	column3	column4	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

olumn2	column3	column4	column5	column5
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

`"`

`"` starting after `{start}` ending before `{alphanum-underscore}+`

Extract values matching — See all

`"`

`"` starting after `{start}` ending before

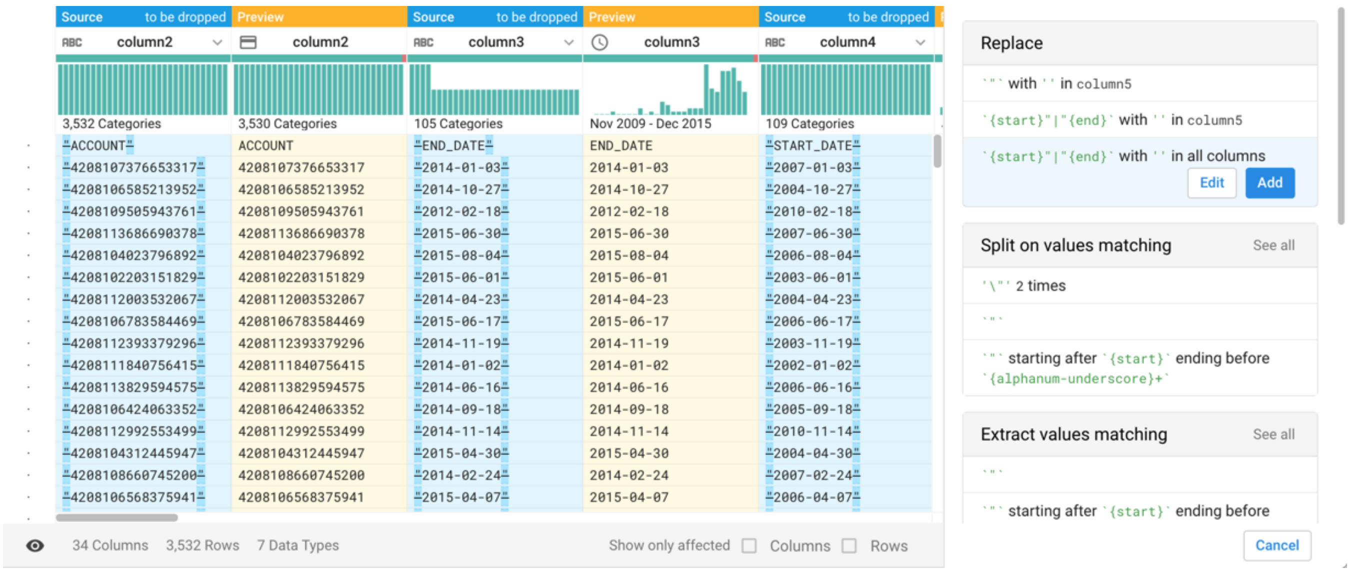
Cancel

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.



Source to be dropped	Preview	Source to be dropped	Preview	Source to be dropped
column2	column2	column3	column3	column4
3,532 Categories	3,530 Categories	105 Categories	Nov 2009 - Dec 2015	109 Categories
"ACCOUNT"	ACCOUNT	"END_DATE"	END_DATE	"START_DATE"
"4208107376653317"	4208107376653317	"2014-01-03"	2014-01-03	"2007-01-03"
"4208106585213952"	4208106585213952	"2014-10-27"	2014-10-27	"2004-10-27"
"4208109505943761"	4208109505943761	"2012-02-18"	2012-02-18	"2010-02-18"
"4208113686690378"	4208113686690378	"2015-06-30"	2015-06-30	"2007-06-30"
"4208104023796892"	4208104023796892	"2015-08-04"	2015-08-04	"2006-08-04"
"4208102203151829"	4208102203151829	"2015-06-01"	2015-06-01	"2003-06-01"
"4208112003532067"	4208112003532067	"2014-04-23"	2014-04-23	"2004-04-23"
"4208106783584469"	4208106783584469	"2015-06-17"	2015-06-17	"2006-06-17"
"4208112393379296"	4208112393379296	"2014-11-19"	2014-11-19	"2003-11-19"
"4208111840756415"	4208111840756415	"2014-01-02"	2014-01-02	"2002-01-02"
"4208113829594575"	4208113829594575	"2014-06-16"	2014-06-16	"2006-06-16"
"4208106424063352"	4208106424063352	"2014-09-18"	2014-09-18	"2005-09-18"
"4208112992553499"	4208112992553499	"2014-11-14"	2014-11-14	"2010-11-14"
"4208104312445947"	4208104312445947	"2015-04-30"	2015-04-30	"2004-04-30"
"4208108660745200"	4208108660745200	"2014-02-24"	2014-02-24	"2007-02-24"
"4208106568375941"	4208106568375941	"2015-04-07"	2015-04-07	"2006-04-07"

34 Columns 3,532 Rows 7 Data Types Show only affected ☐ Columns ☐ Rows

Replace

`"` with '' in column5

`{start}"|"{end}` with '' in column5

`{start}"|"{end}` with '' in all columns

Split on values matching

'\"' 2 times

`"`

`"` starting after `{start}` ending before `{alphanum-underscore}+`

Extract values matching

`"`

`"` starting after `{start}` ending before

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

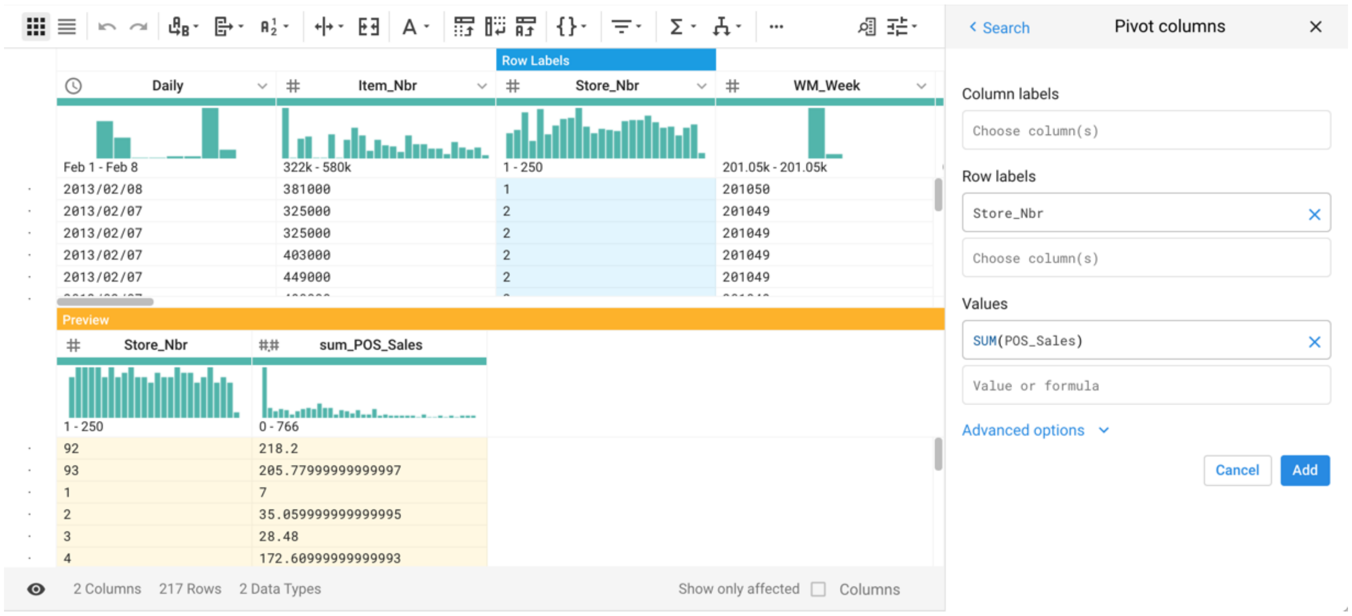
An error might appear while you are typing the transform. Because the transform is not valid, the transform preview appears blurred and does not display the results of the transform.

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



Daily	Item_Nbr	Store_Nbr	WM_Week
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

Store_Nbr	sum_POS_Sales
92	218.2
93	205.77999999999997
1	7
2	35.059999999999995
3	28.48
4	172.60999999999993

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.

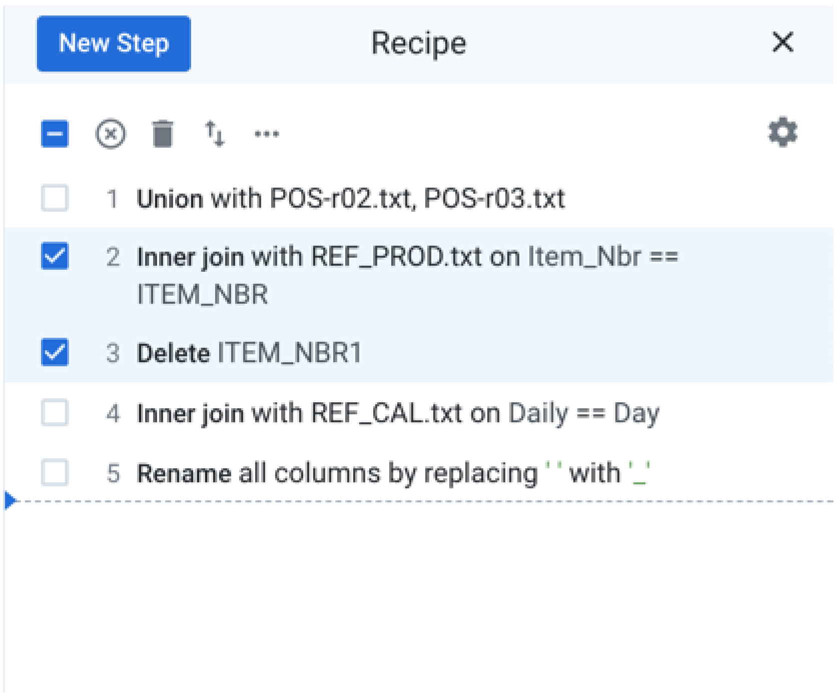
The following panels may be displayed within the context panel.

To close the context panel, click the X icon in the upper-right corner.

Recipe Panel

Contents:

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.



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 Rename all columns by replacing ' ' with '_'

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.

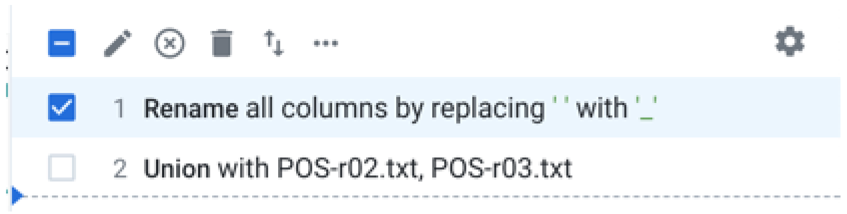
When you select one or more steps, the recipe toolbar is displayed. See below.

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.



1 Rename all columns by replacing ' ' with '_'

2 Union with POS-r02.txt, POS-r03.txt

Figure: Recipe toolbar

Tools:

- **Select/Deselect checkbox:** Select or deselect all steps in your recipe.
- **Edit:** (Single step selected only) Edit the recipe step.

> **NOTE**: When data grid updating is disabled, you may not be able to edit some recipe steps. For steps that you can edit, select **Preview** to see the effects of the step on the data.

- **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*.
- **Edit with data grid**: Select to enable or disable display of data with the data grid. For more information, see *Data Grid 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.

> **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.

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

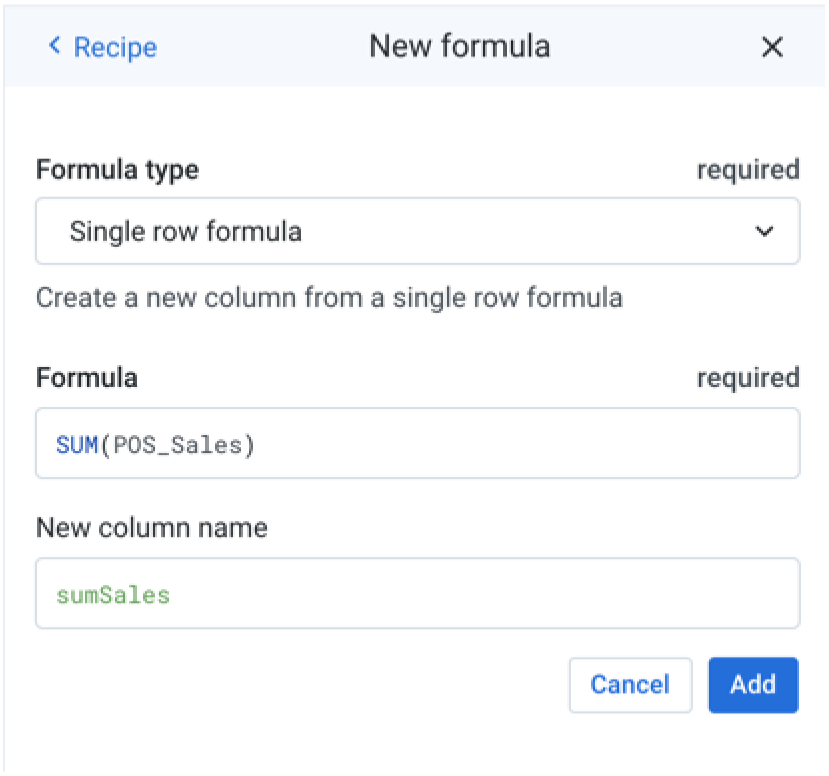
Contents:

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.



Formula type required

Single row formula

Create a new column from a single row formula

Formula required

`SUM(POS_Sales)`

New column name

`sumSales`

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: `'This is what I want to match.'`
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: `` `{start}{digit}{digit}` ``
Regular expression	Regular expressions are a standard method of describing matching patterns. **NOTE:** 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: `/^\d$\|^\d\d$/`

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, Pattern , 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 *nth* 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` col umn 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)`

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

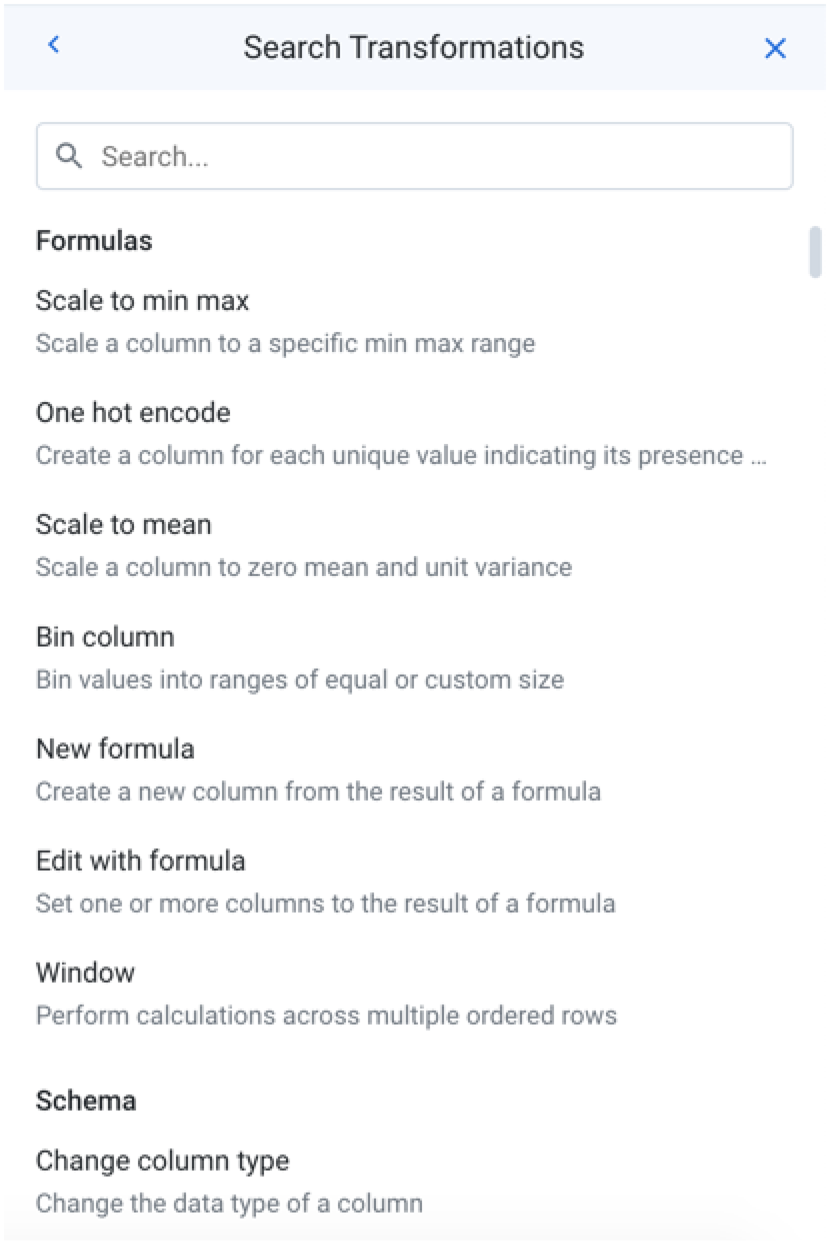
Through the search context panel, you can locate transformations to specify and add at the current location in your recipe.

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.



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.

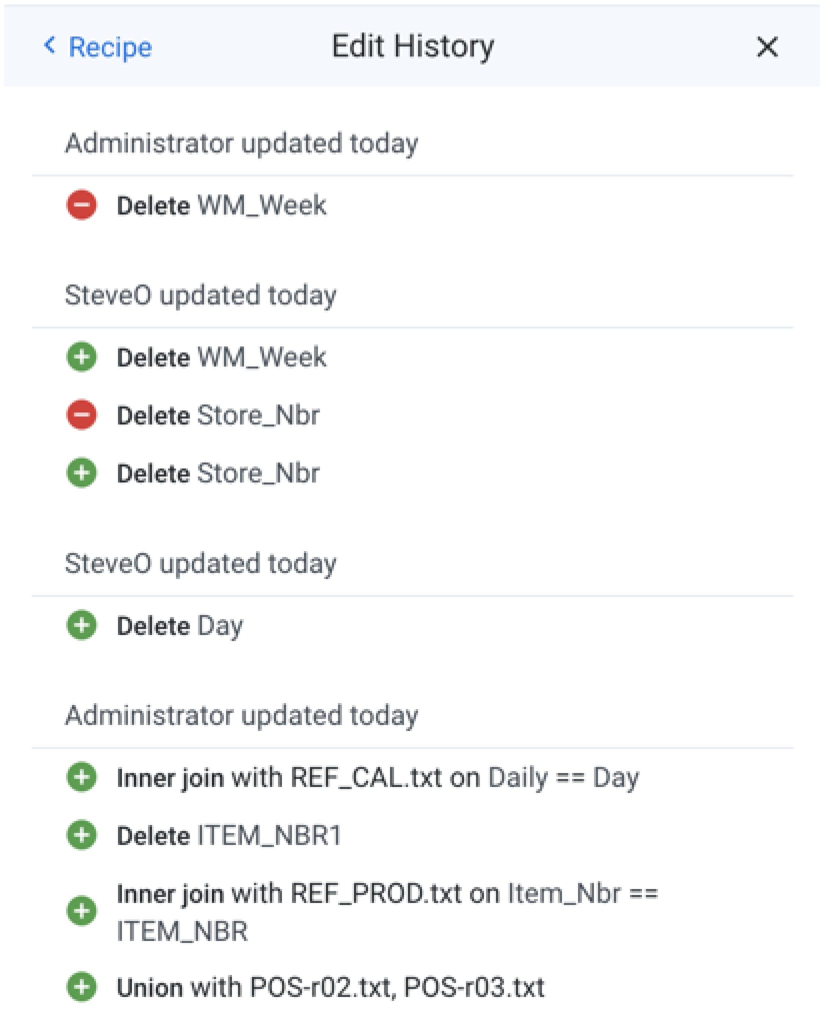
- If you are familiar with Wrangle transforms, you can enter the transform name in the search bar. For example, type `window`.

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*.



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).

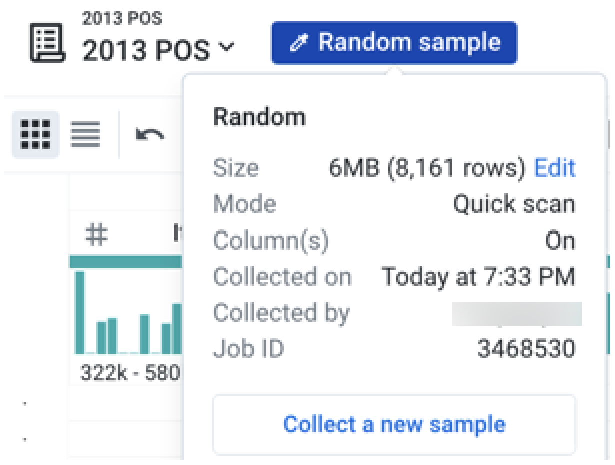
Edit operations are represented by a delete operation and then an add operation.

Samples Panel

Contents:

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:



Random

Size 6MB (8,161 rows) Edit
Mode Quick scan
Column(s) On
Collected on Today at 7:33 PM
Collected by
Job ID 3468530

Figure: Click the current sample button.

In the example above, you can see that the current sample is a Random sample.

Initial Data: The sample is taken from the first set of rows in the first file or table that is part of the dataset.

- In some cases, the Initial Data sample is the entire dataset.

  > **Tip:** For purposes of loading the data, the initial data sample is generated and displayed at first. For a better representation of the entire dataset, you should create a new sample.

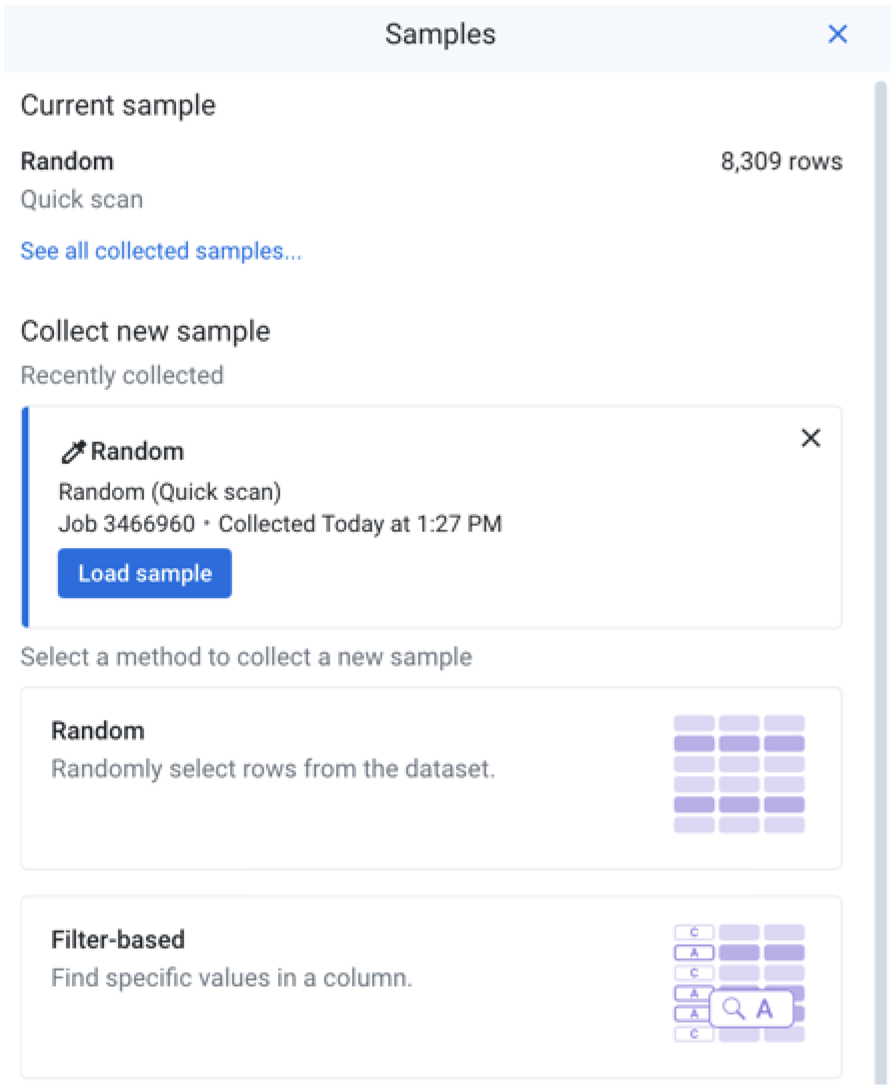
- In other cases, the Initial Data sample is generated from a collection of files.
- If the recipe is a child recipe, then the Initial Data sample indicates the selected sample of the parent recipe.
- For more information on this special sampling type, see *Overview of Sampling*.

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*.


Samples

Current sample

Random 8,309 rows

Quick scan

See all collected samples...

Collect new sample

Recently collected

Random

Random (Quick scan)

Job 3466960 • Collected Today at 1:27 PM

Load sample

Select a method to collect a new sample

Random

Randomly select rows from the dataset.

Filter-based

Find specific values in a column.

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 Data:** By default, the application loads the first N rows of the dataset as the initial data 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 or may be limited based on the maximum number of files that can be scanned. For datasets smaller than this limit, the entire dataset is loaded. See *Overview of Sampling*.

- 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.

  Tip: A sample execution is a type of job. Any issues related to the execution of a sampling can be reviewed through the job logs.

- 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.

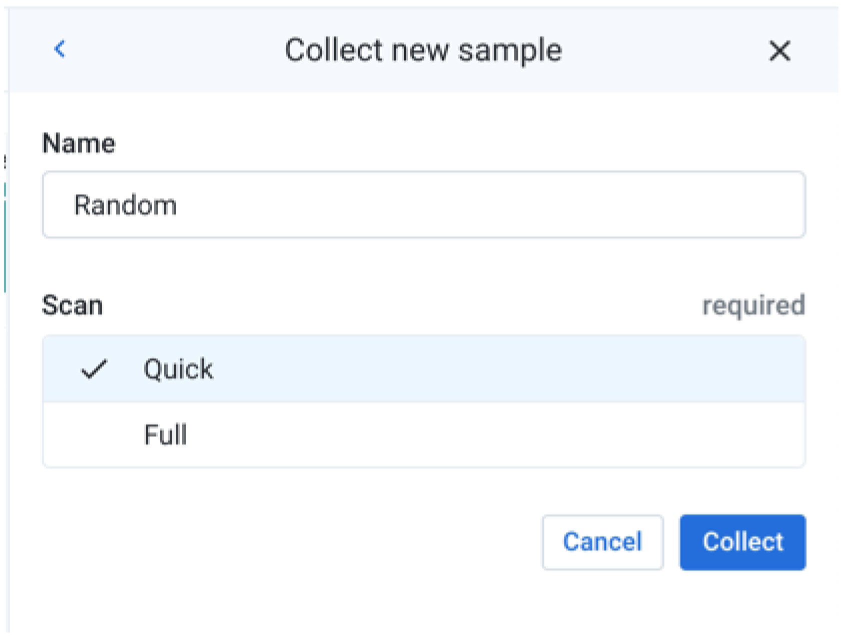


Figure: Collect new sample panel

NOTE: Except for the initial data 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, click **Load**. The sample is loaded in the data grid. For more information, see *Generate a Sample*.

> **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

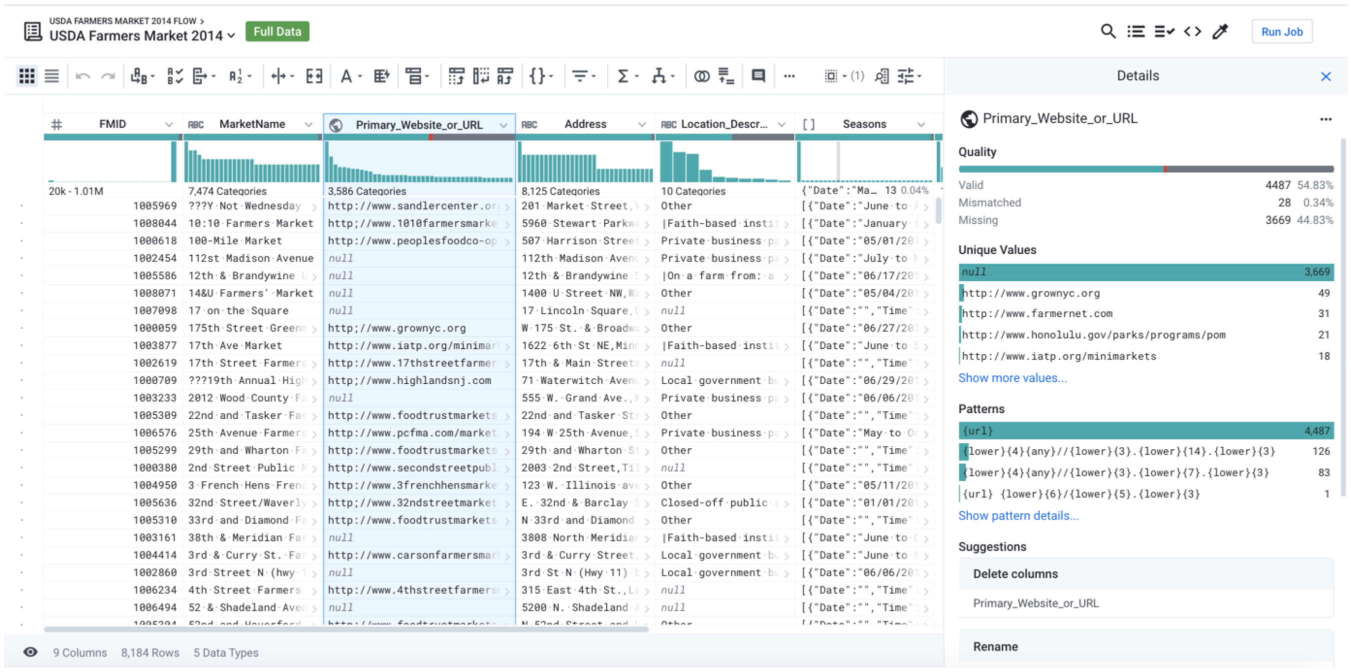
You can cancel a sample job that is currently being executed.

- 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:

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.



FMID	MarketName	Primary_Website_or_URL	Address	Location_Descr...	Seasons
20k - 1.01M	7,474 Categories	3,586 Categories	8,125 Categories	10 Categories	{"Date":"Ma… 13 0.04%
1005969	???Y Not Wednesday	http://www.sandlercenter.or	201 Market Street,	Other	[{"Date":"June to
1008044	10:10 Farmers Market	http;//www.1010farmersmarke	5960 Stewart Parkwa	|Faith-based insti	[{"Date":"January
1000618	100-Mile Market	http://www.peoplesfoodco-op	507 Harrison Street	Private business pa	[{"Date":"05/01/201
1002454	112st Madison Avenue	null	112th Madison Avenu	Private business pa	[{"Date":"July to
1005586	12th & Brandywine U	null	12th & Brandywine S	|On a farm from: a	[{"Date":"06/17/201
1008071	14&U Farmers' Market	null	1400 U Street NW,Wa	Other	[{"Date":"05/04/201
1007098	17 on the Square	null	17 Lincoln Square,C	null	[{"Date":"","Time"
1000059	175th Street Greenm	http;//www.grownyc.org	W 175 St. & Broadwa	Other	[{"Date":"06/27/201
1003877	17th Ave Market	http://www.iatp.org/minimar	1622 6th St NE,Minr	|Faith-based insti	[{"Date":"June to
1002619	17th Street Farmers	http://www.17thstreetfarmer	17th & Main Streets	null	[{"Date":"","Time"
1000709	???19th Annual High	http;//www.highlandsnj.com	71 Waterwitch Avenu	Local government bu	[{"Date":"06/29/201
1003233	2012 Wood County Fa	null	555 W. Grand Ave.,W	Private business pa	[{"Date":"06/06/201
1005309	22nd and Tasker Far	http://www.foodtrustmarkets	22nd and Tasker Str	Other	[{"Date":"","Time"
1006576	25th Avenue Farmers	http://www.pcfma.com/market	194 W 25th Avenue,S	Private business pa	[{"Date":"May to Oc
1005299	29th and Wharton Fa	http://www.foodtrustmarkets	29th and Wharton St	Other	[{"Date":"","Time"
1000380	2nd Street Public M	http://www.secondstreetpubl	2003 2nd Street,Til	null	[{"Date":"","Time"
1004950	3 French Hens Frenc	http://www.3frenchhensmarke	123 W. Illinois ave	Other	[{"Date":"05/11/201
1005636	32nd Street/Waverly	http;//www.32ndstreetmarket	E. 32nd & Barclay S	Closed-off public s	[{"Date":"01/01/201
1005310	33rd and Diamond Fa	http://www.foodtrustmarkets	N 33rd and Diamond	Other	[{"Date":"","Time"
1003161	38th & Meridian Far	null	3808 North Meridian	|Faith-based insti	[{"Date":"June to C
1004414	3rd & Curry St. Far	http://www.carsonfarmersmar	3rd & Curry Street,	Local government bu	[{"Date":"June to S
1002860	3rd Street N (hwy 1	null	3rd St N (Hwy 11)	Local government bu	[{"Date":"06/06/201
1006234	4th Street Farmers	http://www.4thstreetfarmers	315 East 4th St.,La	null	[{"Date":"","Time"
1006494	52 & Shadeland Aven	null	5200 N. Shadeland A	null	[{"Date":"","Time"

9 Columns 8,184 Rows 5 Data Types

Primary_Website_or_URL

Quality

Valid	4487	54.83%
Mismatched	28	0.34%
Missing	3669	44.83%

Unique Values

null	3,669
http://www.grownyc.org	49
http://www.farmernet.com	31
http://www.honolulu.gov/parks/programs/pom	21
http://www.iatp.org/minimarkets	18

Show more values...

Patterns

{url}	4,487
{lower}{4}{any}//{lower}{3}.{lower}{14}.{lower}{3}	126
{lower}{4}{any}//{lower}{3}.{lower}{7}.{lower}{3}	83
{url} {lower}{6}/{lower}{5}.{lower}{3}	1

Show pattern details...

Suggestions

Delete columns

Primary_Website_or_URL

Rename

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 attempt s 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

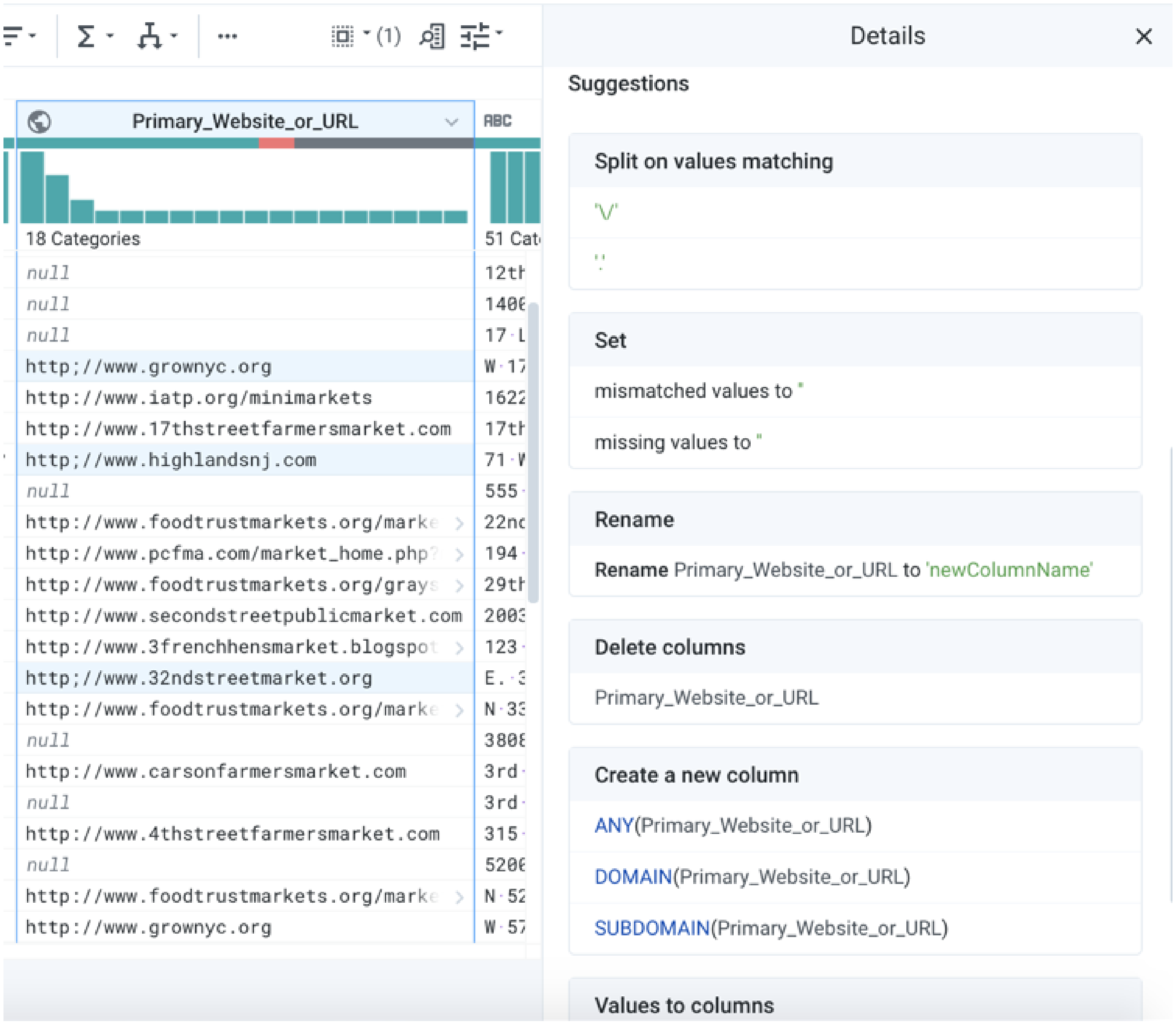
When you select multiple columns, the following changes to the panel apply:

- 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:



```
null
null
null
http;//www.grownyc.org
http://www.iatp.org/minimarkets
http://www.17thstreetfarmersmarket.com
http;//www.highlandsnj.com
null
http://www.foodtrustmarkets.org/marke
http://www.pcfma.com/market_home.php?
http://www.foodtrustmarkets.org/grays
http://www.secondstreetpublicmarket.com
http://www.3frenchhensmarket.blogspot
http;//www.32ndstreetmarket.org
http://www.foodtrustmarkets.org/marke
null
http://www.carsonfarmersmarket.com
null
http://www.4thstreetfarmersmarket.com
null
http://www.foodtrustmarkets.org/marke
http://www.grownyc.org
```

Figure: Selection Details Panel - selected column values

Review and make your choice among the available selections.

Select Values in the Data Grid

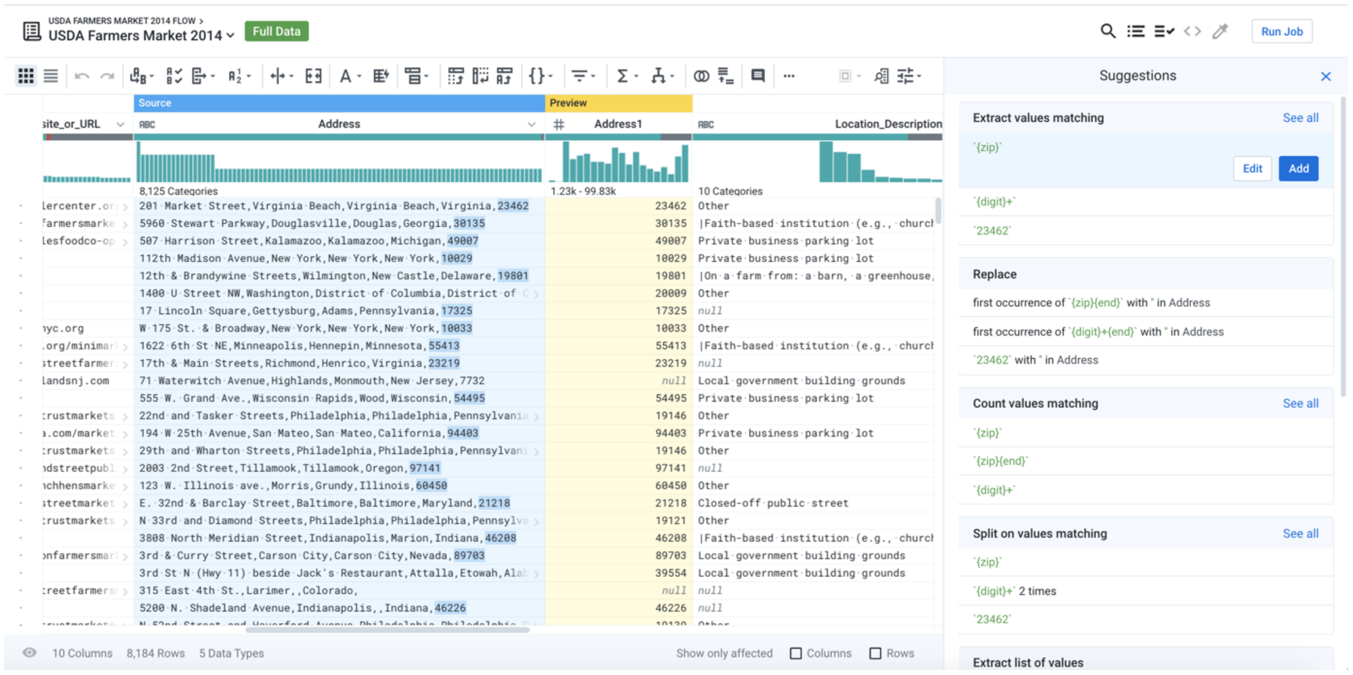
When you select values in the data grid, the Selection Details panel presents a set of suggestions for your review. See below.

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:



site_or_URL	Address	Address1	Location_Description
	8,125 Categories	1.23k - 99.83k	10 Categories
lercenter.or	201 Market Street,Virginia Beach,Virginia Beach,Virginia,23462	23462	Other
farmersmarke	5960 Stewart Parkway,Douglasville,Douglas,Georgia,30135	30135	|Faith-based institution (e.g., churcl
lesfoodco-op	507 Harrison Street,Kalamazoo,Kalamazoo,Michigan,49007	49007	Private business parking lot
	112th Madison Avenue,New York,New York,New York,10029	10029	Private business parking lot
	12th & Brandywine Streets,Wilmington,New Castle,Delaware,19801	19801	|On a farm from: a barn, a greenhouse,
	1400 U Street NW,Washington,District of Columbia,District of C	20009	Other
	17 Lincoln Square,Gettysburg,Adams,Pennsylvania,17325	17325	null
nyc.org	W 175 St. & Broadway,New York,New York,New York,10033	10033	Other
.org/minimar	1622 6th St NE,Minneapolis,Hennepin,Minnesota,55413	55413	|Faith-based institution (e.g., churcl
streetfarmer	17th & Main Streets,Richmond,Henrico,Virginia,23219	23219	null
landsnj.com	71 Waterwitch Avenue,Highlands,Monmouth,New Jersey,7732	null	Local government building grounds
	555 W. Grand Ave.,Wisconsin Rapids,Wood,Wisconsin,54495	54495	Private business parking lot
trustmarkets	22nd and Tasker Streets,Philadelphia,Philadelphia,Pennsylvania	19146	Other
a.com/market	194 W 25th Avenue,San Mateo,San Mateo,California,94403	94403	Private business parking lot
trustmarkets	29th and Wharton Streets,Philadelphia,Philadelphia,Pennsylvani	19146	Other
ndstreetpubl	2003 2nd Street,Tillamook,Tillamook,Oregon,97141	97141	null
nchhensmarke	123 W. Illinois ave.,Morris,Grundy,Illinois,60450	60450	Other
streetmarket	E. 32nd & Barclay Street,Baltimore,Baltimore,Maryland,21218	21218	Closed-off public street
trustmarkets	N 33rd and Diamond Streets,Philadelphia,Philadelphia,Pennsylva	19121	Other
	3808 North Meridian Street,Indianapolis,Marion,Indiana,46208	46208	|Faith-based institution (e.g., churcl
onfarmersmar	3rd & Curry Street,Carson City,Carson City,Nevada,89703	89703	Local government building grounds
	3rd St N (Hwy 11) beside Jack's Restaurant,Attalla,Etowah,Alab	39554	Local government building grounds
treetfarmers	315 East 4th St.,Larimer,,Colorado,	null	null
	5200 N. Shadeland Avenue,Indianapolis,,Indiana,46226	46226	null

10 Columns 8,184 Rows 5 Data Types Show only affected ☐ Columns ☐ Rows

Suggestions

Extract values matching See all
- `{zip}` Edit Add
- `{digit}+`
- `23462`

Replace
- first occurrence of `{zip}{end}` with '' in Address
- first occurrence of `{digit}+{end}` with '' in Address
- `23462` with '' in Address

Count values matching See all
- `{zip}`
- `{zip}{end}`
- `{digit}+`

Split on values matching See all
- `{zip}`
- `{digit}+` 2 times
- `23462`

Extract list of values

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.

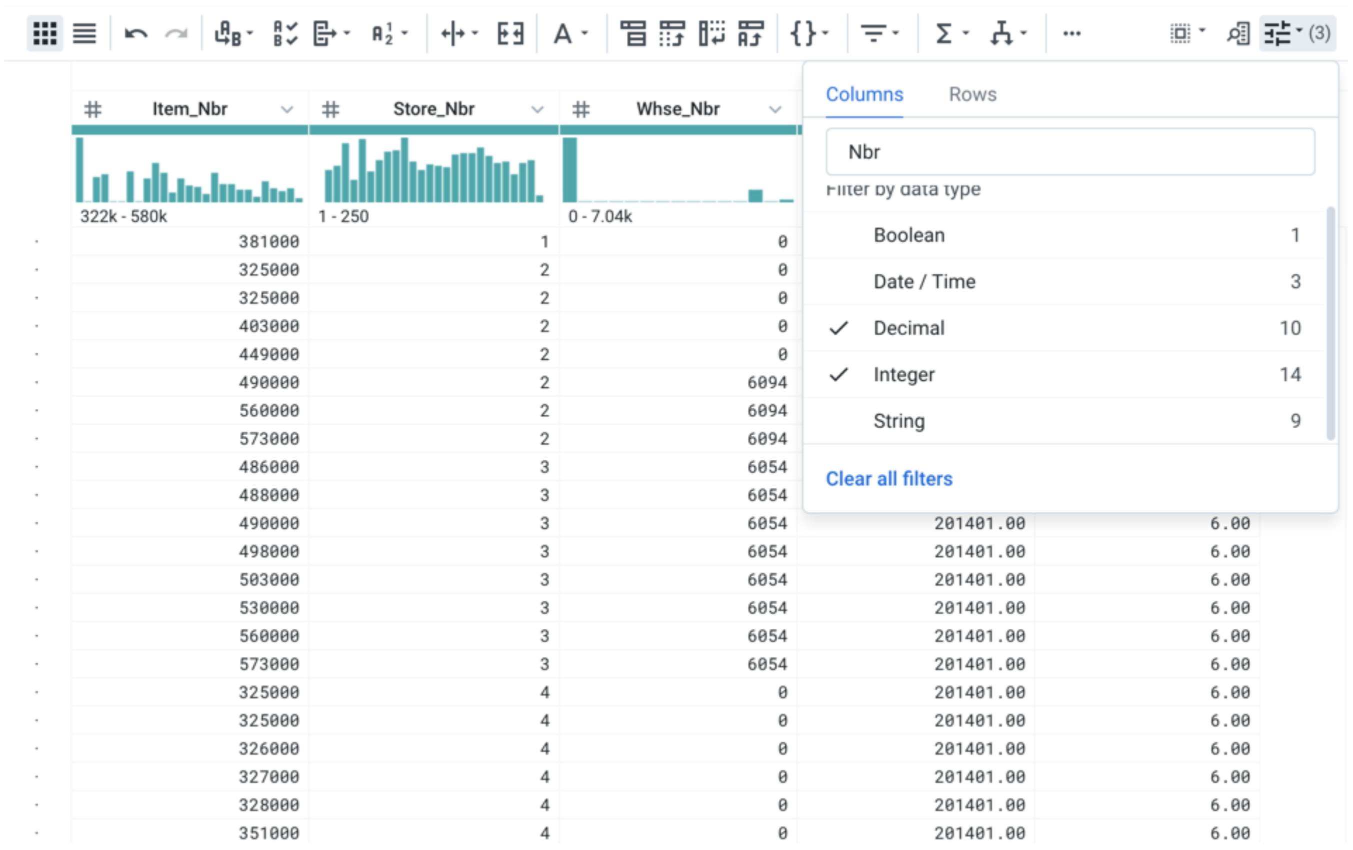
3. Modify the selected transform. To edit the step, click **Edit**. You can fine-tune parameters of the transformation. See *Transform Builder*.

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**.
- 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.



Columns Rows

Nbr

Filter by data type

	Boolean	1
	Date / Time	3
✓	Decimal	10
✓	Integer	14
	String	9

Clear all filters

Item_Nbr	Store_Nbr	Whse_Nbr		
381000	1	0		
325000	2	0		
325000	2	0		
403000	2	0		
449000	2	0		
490000	2	6094		
560000	2	6094		
573000	2	6094		
486000	3	6054		
488000	3	6054		
490000	3	6054	201401.00	6.00
498000	3	6054	201401.00	6.00
503000	3	6054	201401.00	6.00
530000	3	6054	201401.00	6.00
560000	3	6054	201401.00	6.00
573000	3	6054	201401.00	6.00
325000	4	0	201401.00	6.00
325000	4	0	201401.00	6.00
326000	4	0	201401.00	6.00
327000	4	0	201401.00	6.00
328000	4	0	201401.00	6.00
351000	4	0	201401.00	6.00

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.

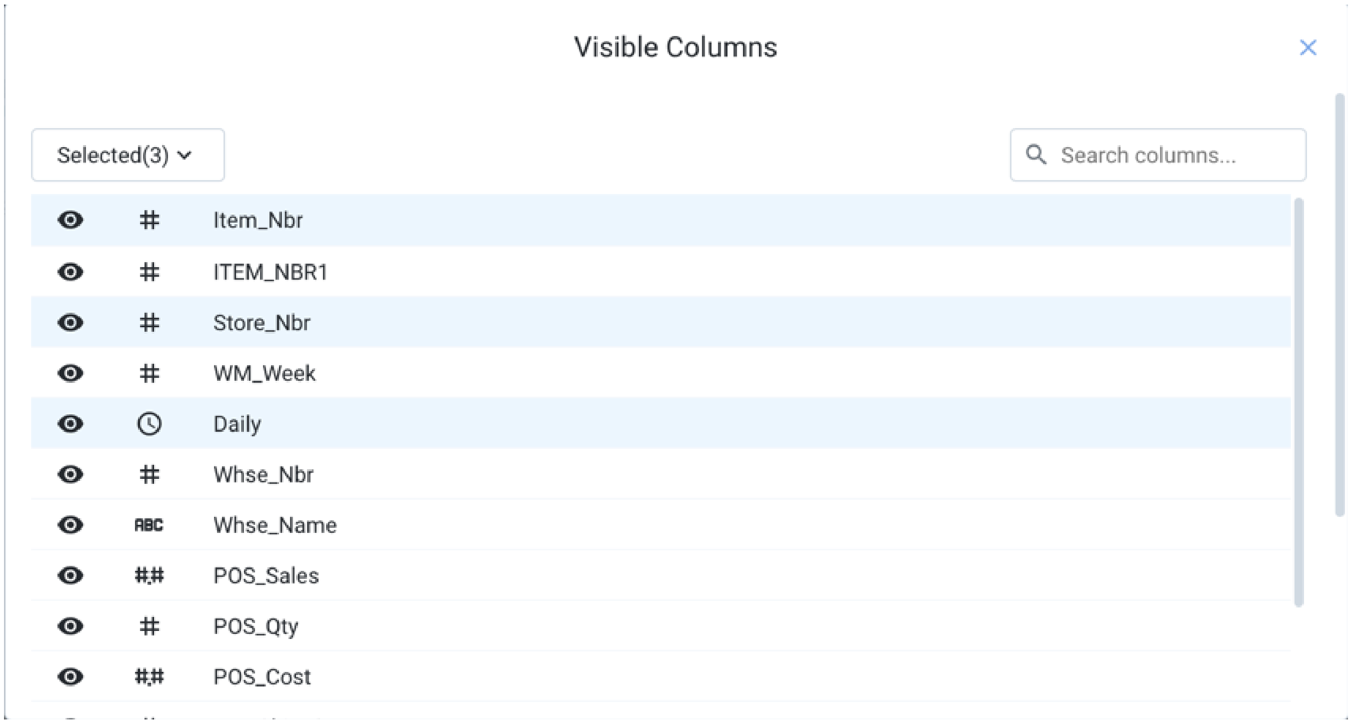


Figure: Visible Columns Panel

- 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:

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*.

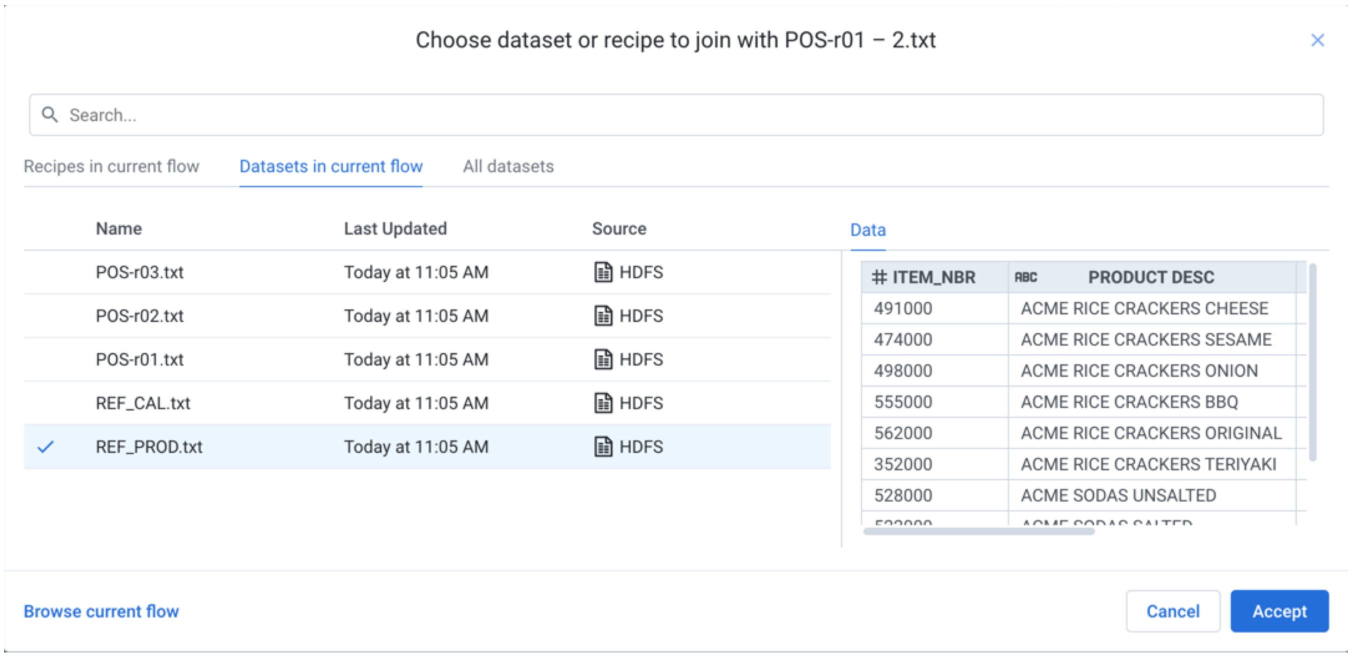
NOTE: Unnest, union, or join transforms may significantly increase the number of rows or columns in your dataset. To prevent overloading the browser's memory, the application may apply a limit function to the results to artificially limit the number of rows displayed in your sample. You can generate 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
	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

Data

# ITEM_NBR	ABC 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

Figure: Select dataset or recipe to join

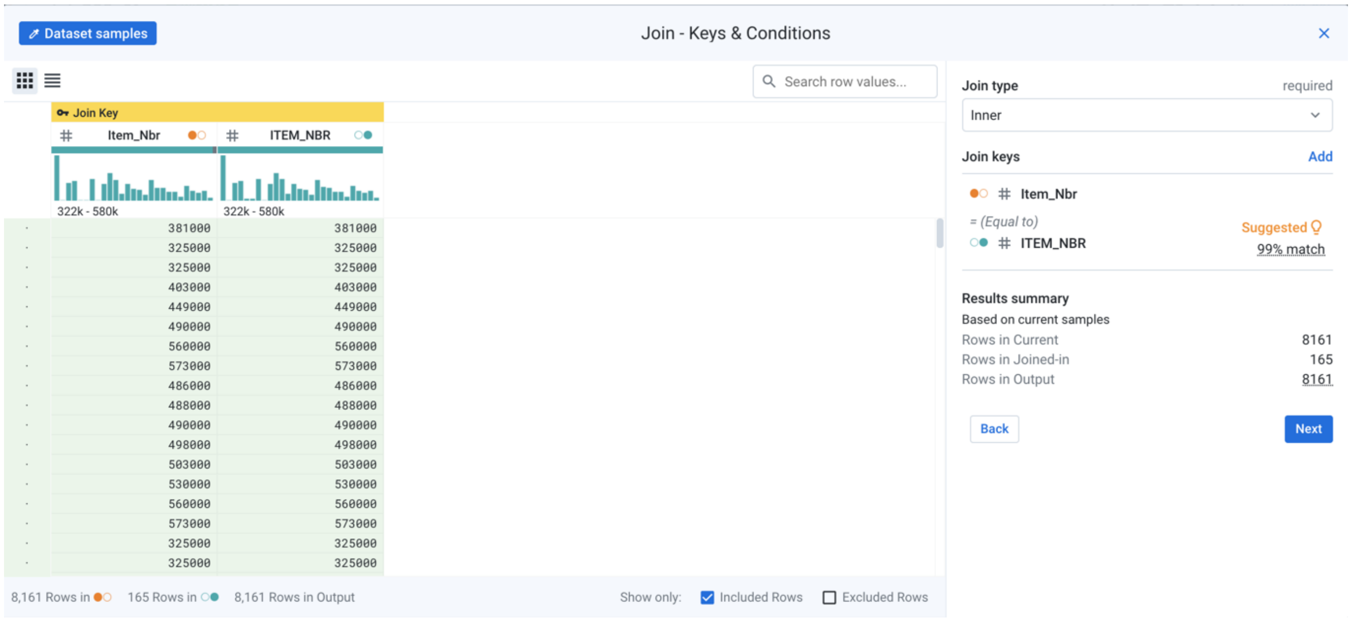
You can use the Data tab to preview the data in the selected object.

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).



# Item_Nbr	# ITEM_NBR
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

Show only: ✓ Included Rows ☐ Excluded Rows

Join type (required): Inner

Join keys — Add

●○ # Item_Nbr
= (Equal to) Suggested 99% match
○● # ITEM_NBR

Results summary
Based on current samples

Rows in Current	8161
Rows in Joined-in	165
Rows in Output	8161

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.

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	Use a fuzzy matching algorithm for key value matching. **Tip:** Use this option to perform fuzzy join matching of primary keys between datasets. **NOTE:** Fuzzy joins can only be applied to String data types. Other data types cannot be fuzzy-matched using the algorithm. Fuzzy matching uses the doublemetaphone algorithm for matching strings (keys). Both primary encodings of each key value must match. See *DOUBLEMETAPHONEEQUALS Function*.
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.

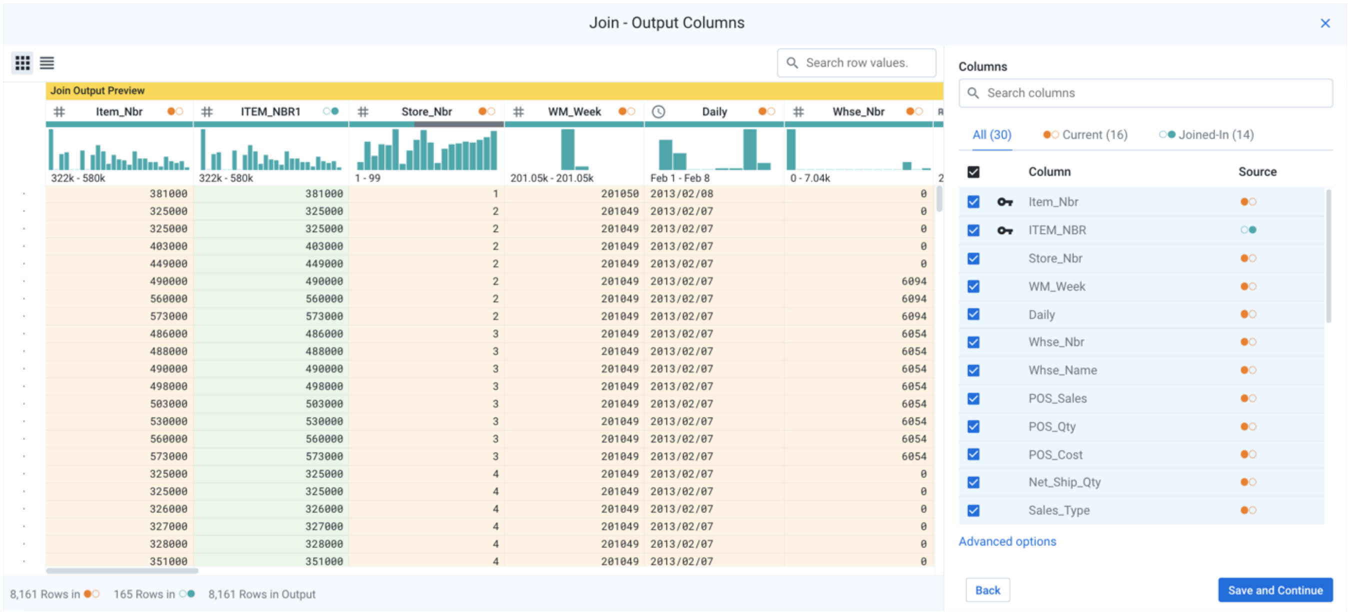
Summary:

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.


Join - Output Columns

Join Output Preview

Item_Nbr	ITEM_NBR1	Store_Nbr	WM_Week	Daily	Whse_Nbr
322k - 580k	322k - 580k	1 - 99	201.05k - 201.05k	Feb 1 - Feb 8	0 - 7.04k
381000	381000	1	201050	2013/02/08	0
325000	325000	2	201049	2013/02/07	0
325000	325000	2	201049	2013/02/07	0
403000	403000	2	201049	2013/02/07	0
449000	449000	2	201049	2013/02/07	0
490000	490000	2	201049	2013/02/07	6094
560000	560000	2	201049	2013/02/07	6094
573000	573000	2	201049	2013/02/07	6094
486000	486000	3	201049	2013/02/07	6054
488000	488000	3	201049	2013/02/07	6054
490000	490000	3	201049	2013/02/07	6054
498000	498000	3	201049	2013/02/07	6054
503000	503000	3	201049	2013/02/07	6054
530000	530000	3	201049	2013/02/07	6054
560000	560000	3	201049	2013/02/07	6054
573000	573000	3	201049	2013/02/07	6054
325000	325000	4	201049	2013/02/07	0
325000	325000	4	201049	2013/02/07	0
326000	326000	4	201049	2013/02/07	0
327000	327000	4	201049	2013/02/07	0
328000	328000	4	201049	2013/02/07	0
351000	351000	4	201049	2013/02/07	0

8,161 Rows in ●○ 165 Rows in ○● 8,161 Rows in Output

Columns

All (30) ●○ Current (16) ○● Joined-In (14)

✓	Column	Source
✓	Item_Nbr	●○
✓	ITEM_NBR	○●
✓	Store_Nbr	●○
✓	WM_Week	●○
✓	Daily	●○
✓	Whse_Nbr	●○
✓	Whse_Name	●○
✓	POS_Sales	●○
✓	POS_Qty	●○
✓	POS_Cost	●○
✓	Net_Ship_Qty	●○
✓	Sales_Type	●○

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*.

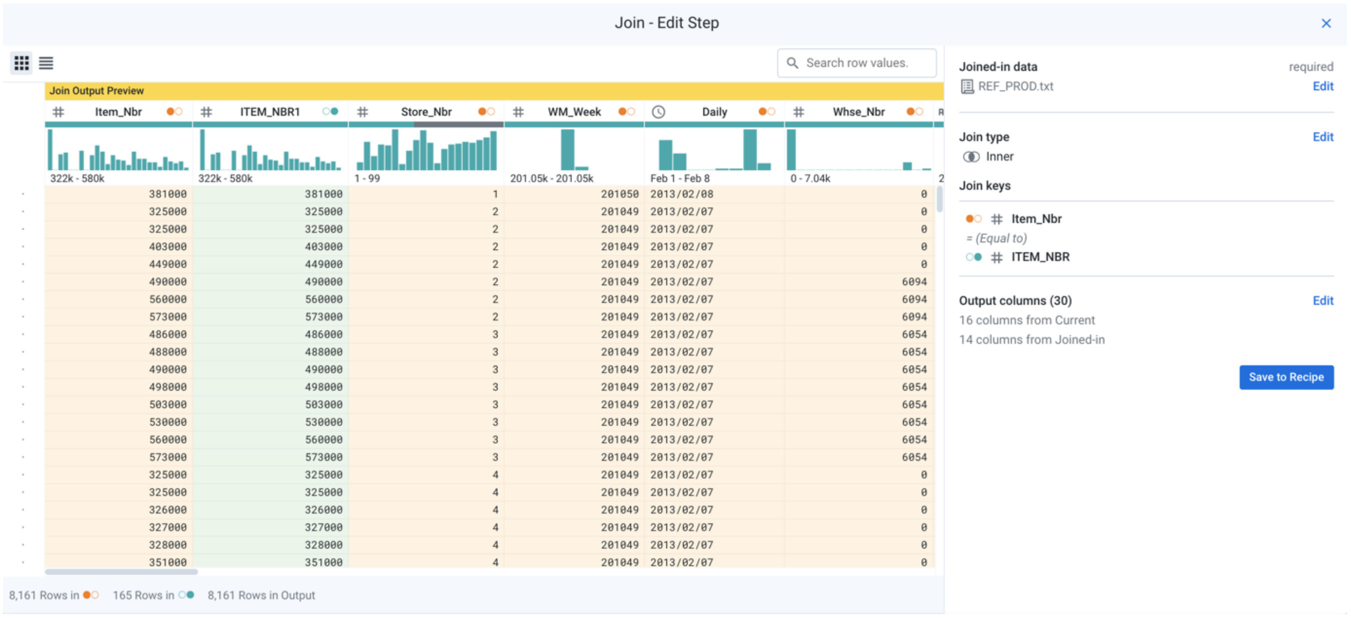
To join in the copy, edit the join and change the source that is being joined. See *Fix Dependency Issues*.

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.



Join - Edit Step

Join Output Preview

Item_Nbr	ITEM_NBR1	Store_Nbr	WM_Week	Daily	Whse_Nbr
322k - 580k	322k - 580k	1 - 99	201.05k - 201.05k	Feb 1 - Feb 8	0 - 7.04k
381000	381000	1	201050	2013/02/08	0
325000	325000	2	201049	2013/02/07	0
325000	325000	2	201049	2013/02/07	0
403000	403000	2	201049	2013/02/07	0
449000	449000	2	201049	2013/02/07	0
490000	490000	2	201049	2013/02/07	6094
560000	560000	2	201049	2013/02/07	6094
573000	573000	2	201049	2013/02/07	6094
486000	486000	3	201049	2013/02/07	6054
488000	488000	3	201049	2013/02/07	6054
490000	490000	3	201049	2013/02/07	6054
498000	498000	3	201049	2013/02/07	6054
503000	503000	3	201049	2013/02/07	6054
530000	530000	3	201049	2013/02/07	6054
560000	560000	3	201049	2013/02/07	6054
573000	573000	3	201049	2013/02/07	6054
325000	325000	4	201049	2013/02/07	0
325000	325000	4	201049	2013/02/07	0
326000	326000	4	201049	2013/02/07	0
327000	327000	4	201049	2013/02/07	0
328000	328000	4	201049	2013/02/07	0
351000	351000	4	201049	2013/02/07	0

8,161 Rows in ●○ 165 Rows in ○● 8,161 Rows in Output

Joined-in data required
REF_PROD.txt Edit

Join type Edit
Inner

Join keys
●○ Item_Nbr
= (Equal to)
○● ITEM_NBR

Output columns (30) Edit
16 columns from Current
14 columns from Joined-in

Figure: Review join

To add the specified join to your recipe, click **Add to Recipe**.

Union Page

Contents:

In the Union page, you can append data from one or more datasets to an existing dataset.

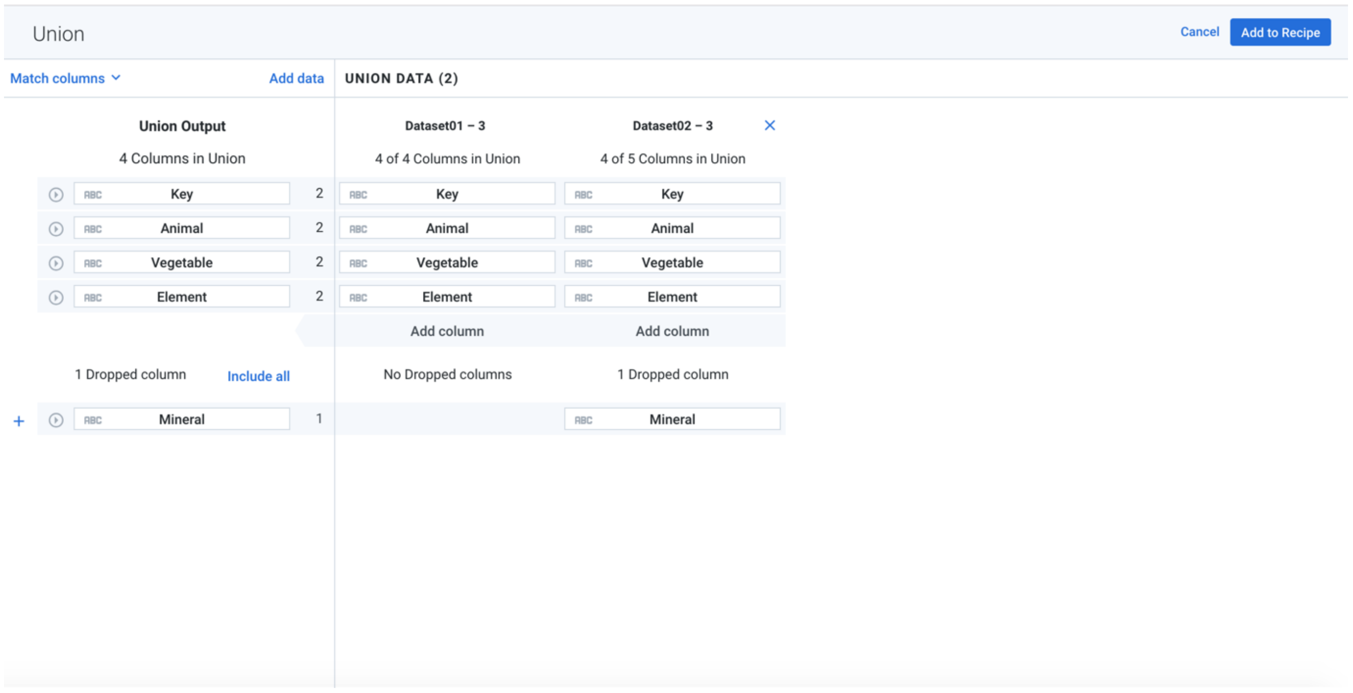
For example, if you have multiple datasets containing transactional data, such as log files, you can use the union operation to join daily or weekly slices of this data into a single dataset.

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 Output		Dataset01 – 3	Dataset02 – 3
4 Columns in Union		4 of 4 Columns in Union	4 of 5 Columns in Union
Key	2	Key	Key
Animal	2	Animal	Animal
Vegetable	2	Vegetable	Vegetable
Element	2	Element	Element
		Add column	Add column
1 Dropped column Include all		No Dropped columns	1 Dropped column
Mineral	1		Mineral

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:

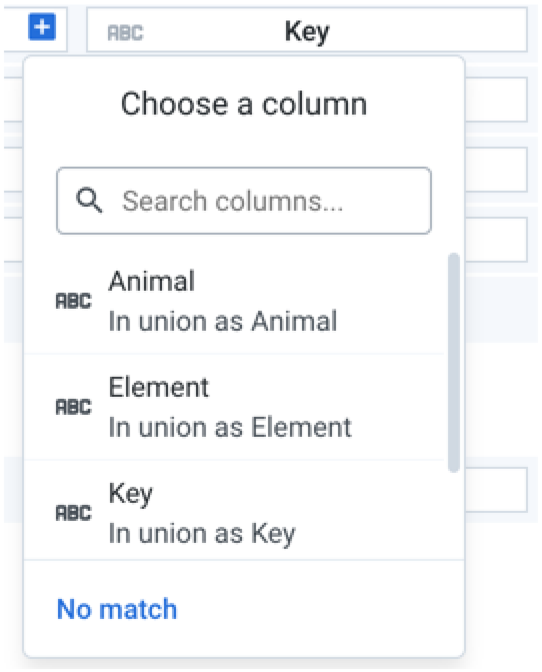


Figure: Custom Column Mapping

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:

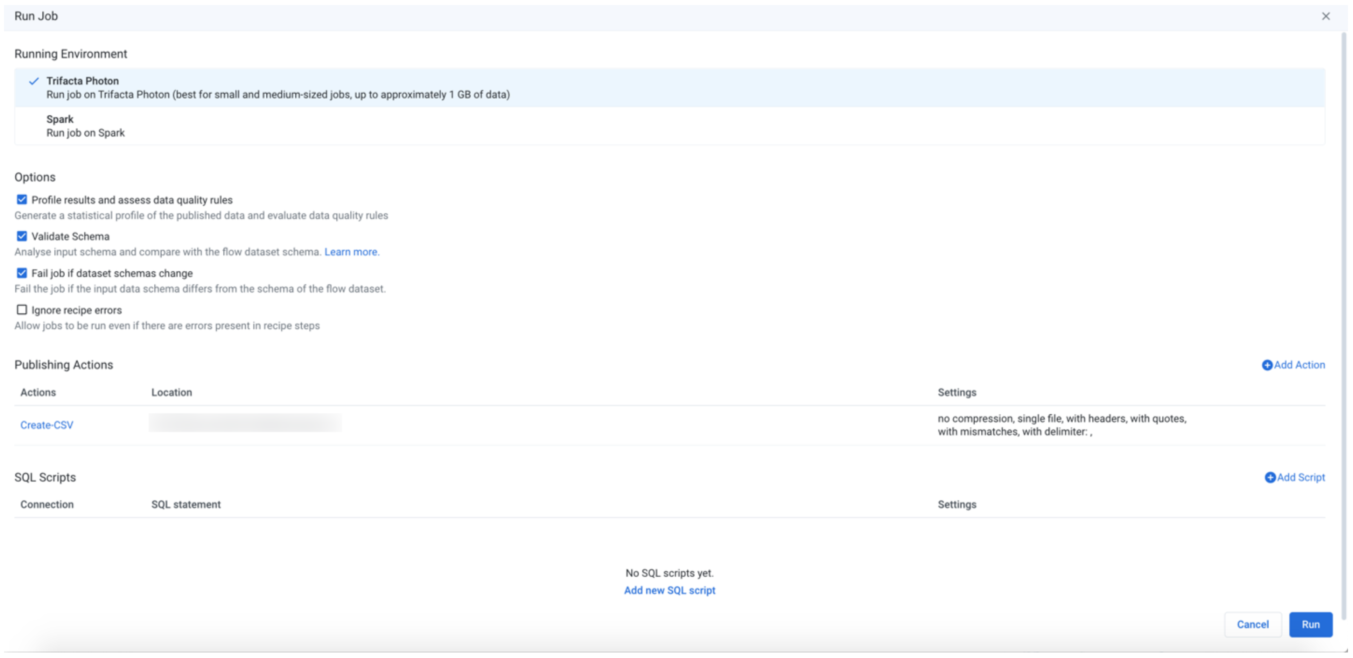
In the Run Job page, you can specify transformation and profiling jobs for the currently loaded recipe. Available options include output formats and output destinations.

You can also configure the environment where the job is to be executed.

NOTE: If the job is executed in an environment other than Trifacta Photon, the job is queued for execution in the environment. Jobs executed on a remote cluster may incur additional overhead to spin up execution nodes, which is typically within 10-15 seconds. During job execution, Designer Cloud Powered by Trifacta Enterprise Edition observes the job in progress and reports progress as needed back into the application. Designer Cloud Powered by Trifacta Enterprise Edition does not control the execution of the job.

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 and assess data quality rules
Generate a statistical profile of the published data and evaluate data quality rules

[✓] Validate Schema
Analyse input schema and compare with the flow dataset schema. Learn more.

[✓] Fail job if dataset schemas change
Fail the job if the input data schema differs from the schema of the flow dataset.

[ ] Ignore recipe errors
Allow jobs to be run even if there are errors present in recipe steps

Publishing Actions

Actions	Location	Settings
Create-CSV	[redacted]	no compression, single file, with headers, with quotes, with mismatches, with delimiter: ,

SQL Scripts

Connection	SQL statement	Settings

No SQL scripts yet.
Add new SQL script

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 + Snowflake : Executes job on Snowflake .

- Snowflake as a running environment must be enabled in the project by a project owner. See *Configure Running Environments*.
- Individual users must enable pushdowns within their flows. See *Flow Optimization Settings Dialog*.

> **NOTE:** Supported file formats are CSV, TXT, and JSON only.

For more information on limitations, see *Snowflake Running Environment*.

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 enable this option to generate a visual profile of your job results.

When the profiling job finishes, details are available through the Job Details page, including links to download results.

- Disabling profiling of your output can improve the speed of overall job execution.
- See *Job Details Page*.

> **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.

Validate Schema: When enabled, the schemas of the datasources for this job are checked for any changes since the last time that the datasets were loaded. Differences are reported in the Job Details page as a Schema validation stage.

> **Tip:** A **schema** defines the column names, data types, and ordering in a dataset.

Fail job if dataset schemas change: When Validate Schema is enabled, you can set this flag to automatically fail the job if there are differences between the stored schemas for your datasets and the schemas that are detected when the job is launched.

NOTE: If you attempt to refresh the schema of a parameterized dataset based on a set of files, only the schema for the first file is checked for changes. If changes are detected, the other files are contain those changes as well. This can lead to changes being assumed or undetected in later files and potential data corruption in the flow.

Tip: This setting prevents data corruption for downstream consumers of your executed jobs.

Tip: The default for validate schema is set at the workspace level. In the Run Job page, these settings are overrides for individual jobs.

For more information, see *Overview of Schema Management*.

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 that are generated from this job. For more information, see *Publishing Actions*.

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*.

Publishing Actions

Contents:

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:

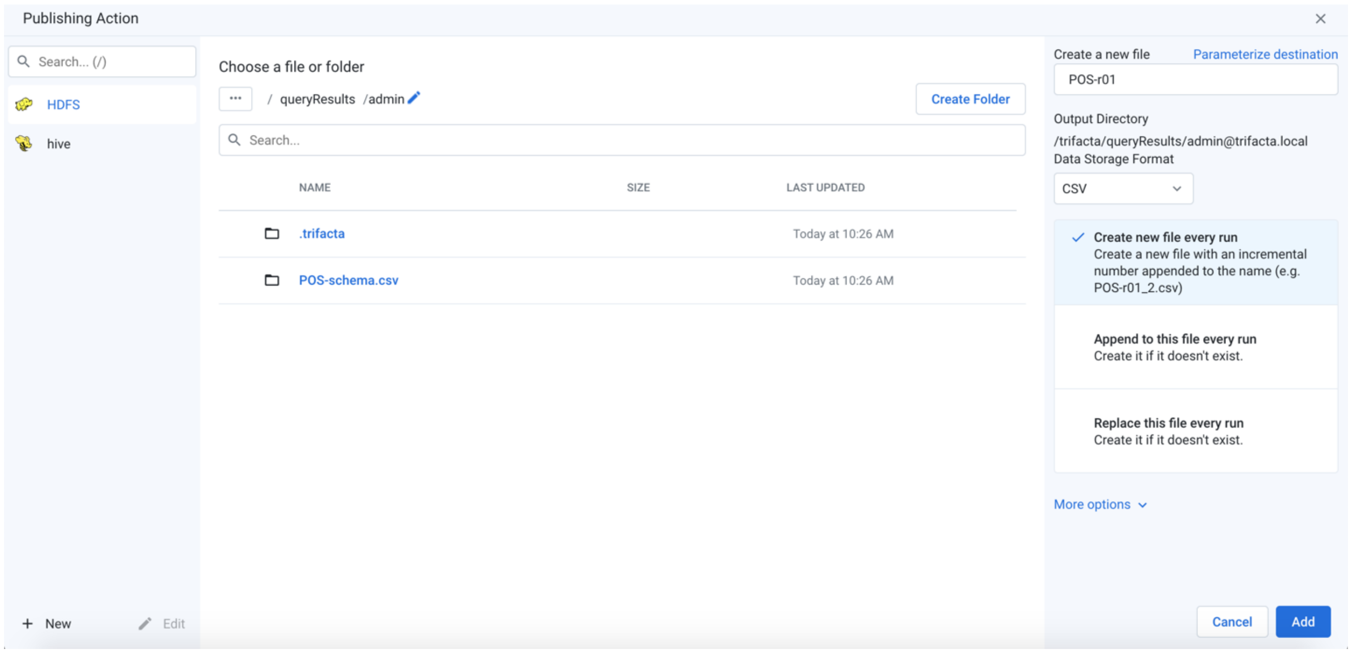
- **Actions:** Lists the action and the format for the output.
- **Location:** The directory and filename or table information where the output is to be written.
- **Settings:** Identifies the output format and any compression, if applicable, for the publication.

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.



NAME	SIZE	LAST UPDATED
.trifacta		Today at 10:26 AM
POS-schema.csv		Today at 10:26 AM

Create a new file: 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.

F

> **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*.
    b. **Snowflake :** You can published your results to the current project or to a different one to which you have access.

        NOTE: You must have read and write access to any Snowflake database to which you are publishing.

        To publish to a different project, click the Snowflake link at the front of the breadcrumb trail. Then, enter the identifier for the project where you wish to publish your job results.

        Tip: Your projects and their identifiers are available for review through the Designer Cloud Powered by Trifacta Enterprise Edition menu bar. See *UI Reference*.

        Click **Go**. Navigate to the database where you wish to write your Snowflake results.

        For more information, see *Snowflake Connections*.
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. **Create a new file:** Enter the filename under which to save the dataset.
    a. Select the Data Storage Format.
    b. For more information, see Output Settings below.
6. As needed, you can parameterize the outputs that you are creating. Click **Parameterize destination** in the right panel. See Parameterize destination settings below.
7. 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.

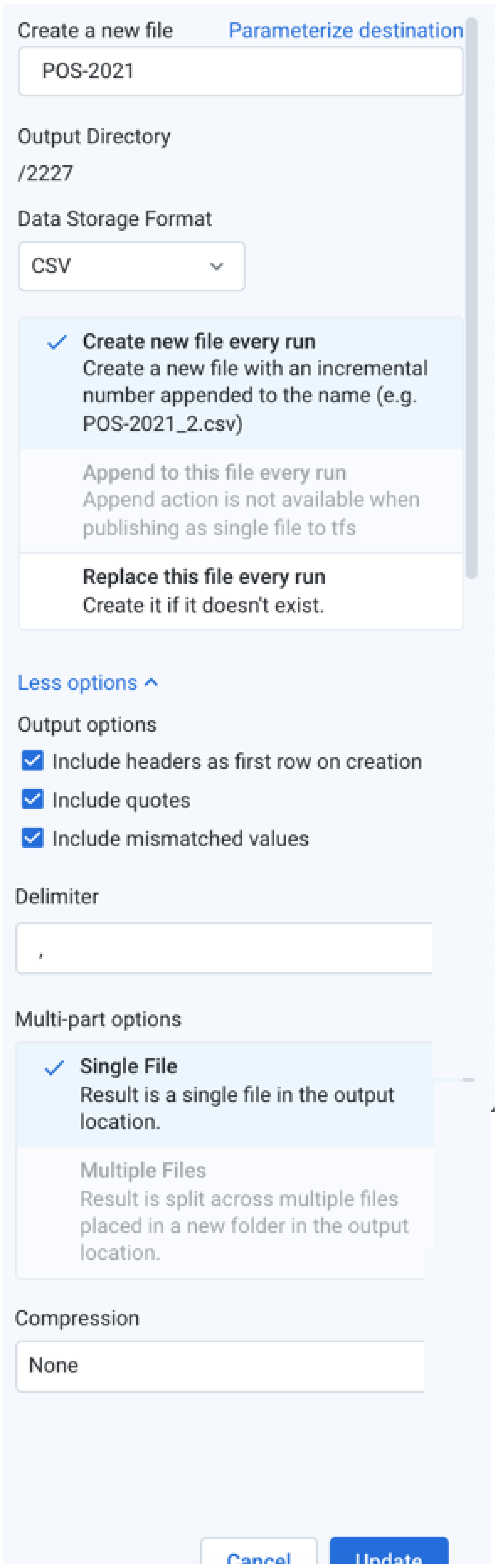
- See *File Settings*.
- See *Relational Table Settings*.

- *Snowflake Table Settings*

File Settings

When you generate file-based results, you can configure the filename, storage format, compression, number of files, and the updating actions in the right-hand panel.

NOTE: By default, when scheduled or API jobs are executed, no validations are performed of any write settings objects for file-based outputs. Issues with these objects may cause failures during transformation or publishing stages of job execution. Jobs of these types should be tested through the Designer Cloud application first. A workspace administrator can disable the skipping of these validations.


Create a new file

POS-2021

Output Directory

/2227

Data Storage Format

CSV

✓ **Create new file every run**
Create a new file with an incremental number appended to the name (e.g. POS-2021_2.csv)

Append to this file every run
Append action is not available when publishing as single file to tfs

Replace this file every run
Create it if it doesn't exist.

Output options

- ✓ Include headers as first row on creation
- ✓ Include quotes
- ✓ Include mismatched values

Delimiter

,

Multi-part options

✓ **Single File**
Result is a single file in the output location.

Multiple Files
Result is split across multiple files placed in a new folder in the output location.

Compression

None

Figure: Output File Settings

Configure the following settings.

1. **Create a new file:** Enter the filename to create. A filename extension is automatically added for you, so you should omit the extension from the filename.
   a. File output paths can have a maximum length of 2048 characters.
2. **Output directory:** Read-only value for the current directory. To change it, navigate to the proper directory.

   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 the canary files and the original folder path are removed. The parameterized folders are not removed. This is a known issue.

3. **Data Storage Format:** Select the output format you want to generate for the job.
   a. **Avro:** This open source format is used widely for data serialization and data transfer between systems.
   b. **CSV and JSON:** These formats are supported for all types of imported datasets and all running environments.

      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*.

   c. **Parquet:** This format is a columnar storage format.

   d. **HYPER:** Choose HYPER to generate results that can be imported into Tableau.

      Feature Availability: This feature may not be available in all product editions. For more information on available features, see *Compare Editions*.

      If you have created a Tableau Server connection, you can write results to Tableau Server or publish them after they have been generated in Hyper format.

      NOTE: If you encounter errors generating results in Hyper format, additional configuration may be required. See *Supported File Formats*.

   e. For more information, see *Supported File Formats*.
4. **Publishing action:** Select one of the following:

   NOTE: If multiple jobs are attempting to publish to the same filename, a numeric suffix (`_N`) is added to the end of subsequent filenames (e.g. `filename_1.csv`).

   NOTE: If a single user executes two jobs with the same output settings except for different methods (e.g. create vs. replace) on the same output destination, the generated results and potential error conditions are unpredictable. Please wait for the first job to complete execution before changing the configuration for the second job.

a. **Create new file every run:** For each job run with the selected publishing destination, a new file is created with the same base name with the job number appended to it (e.g. `myOutput_2.csv`, `myOutput_3.csv`, and so on).
b. **Append to this file every run:** For each job run with the selected publishing destination, the same file is appended, which means that the file grows until it is purged or trimmed.

> **NOTE:** The `append` action is not supported when publishing to S3.

> **NOTE:** When publishing single files to WASB, the `append` action is not supported.

> **NOTE:** When appending data into a Hive table, the columns displayed in the Transformer page must match the order and data type of the columns in the Hive table.

> **NOTE:** Compression of published files is not supported for an `append` action.

c. **Replace this file every run:** For each job run with the selected publishing destination, the existing file is overwritten by the contents of the new results.

5. **More Options:**

a. **Include headers as first row on creation:** For CSV outputs, you can choose to include the column headers as the first row in the output. For other formats, these headers are included automatically.

> **NOTE:** Headers cannot be applied to compressed outputs.

b. **Include quotes:** For CSV outputs, you can choose to include double quote marks around all values, including headers.
c. **Include mismatched values:** For CSV outputs, you can choose to include any value that is mismatched for its column data type. When disabled, mismatched values are written as null values.
d. **Delimiter:** For CSV outputs, you can enter the delimiter that is used to separate fields in the output. The default value is the global delimiter, which you can override on a per-job basis in this field.

> **Tip:** If needed for your job, you can enter Unicode characters in the following format: `\uXXXX`.

> **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. For more information on this issue, see *https://issues.apache.org/jira/browse/SPARK-24540*.

e. **Single File:** Output is written to a single file. Default setting for smaller, file-based jobs.
f. **Multiple Files:** Output is written to multiple files. Default setting for larger file-based jobs.

6. **Compression:** For text-based outputs, compression can be applied to significantly reduce the size of the output. Select a preferred compression format for each format you want to compress.

> **NOTE:** If you encounter errors generating results using Snappy, additional configuration may be required. See *Supported File Formats*.

7. To save the publishing action, click **Add**.

Parameterize Destination Settings

For file- or table-based publishing actions, you can parameterize elements of the output path. Whenever you execute a job, you can pass in parameter values through the Run Job page.

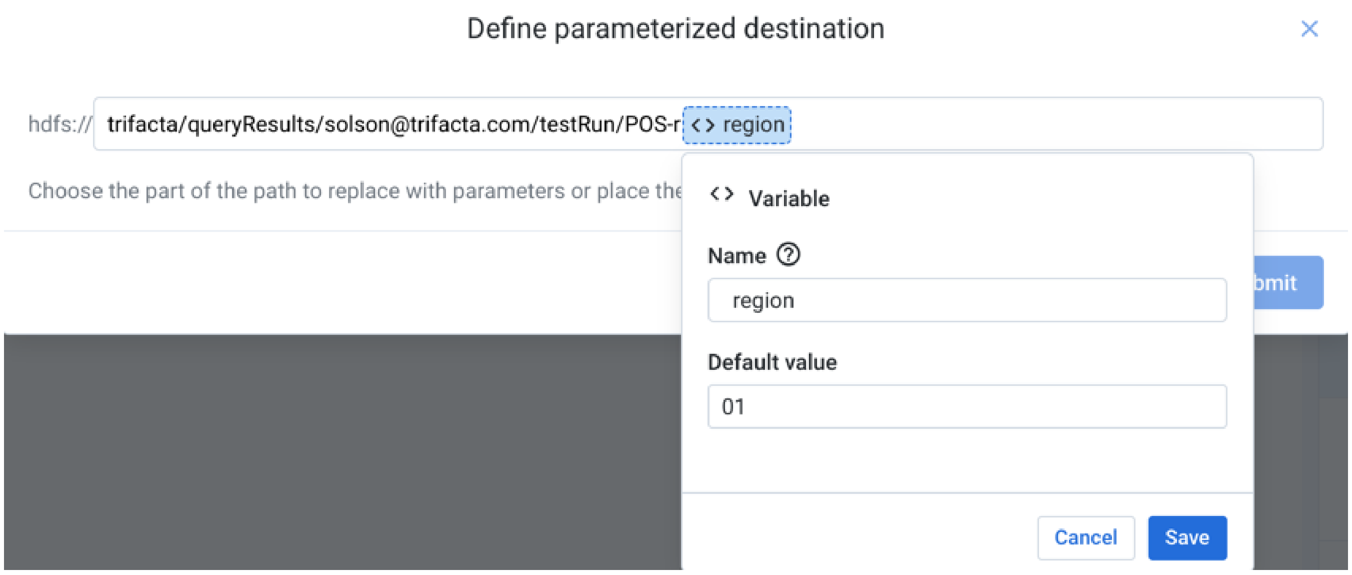
NOTE: Output parameters are independent of dataset parameters. However, two variables of different types with the same name should resolve to the same value.

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 the canary files and the original folder path are removed. The parameterized folders are not removed. This is a known issue.

Supported parameter types:

- Timestamp
- Variable

For more information, see *Overview of Parameterization*.


Define parameterized destination

hdfs:// trifacta/queryResults/solson@trifacta.com/testRun/POS-r <> region

Choose the part of the path to replace with parameters or place the

<> Variable

Name: region

Default value: 01

Figure: Define destination parameter

Steps:

1. When you add or edit a publishing action, click **Parameterize destination** in the right panel.
2. On the listed output path, highlight the part that you wish to parameterize. Then, choose the type of parameter.
3. For Timestamp parameters:
    a. Timestamp format: Specify the format for the timestamp value.
    b. Timestamp value: You can choose to record the exact job start time or the time when the results are written relative to the job start time.
    c. Timezone: To change the timezone recorded in the timestamp, click **Change**.
4. For Variable parameters:

    a. Name: Enter a display name for the variable.

    > **NOTE:** Variable names do not have to be unique. Two variables with the same name should resolve to the same value.

    b. Default value: Enter a default value for the parameter.
5. To save your output parameter, click **Save**.
6. You can create multiple output parameters for the same output.
7. To save all of your parameters for the output path, click **Submit**.
8. The parameter or parameters that you have created are displayed at the bottom of the screen. You can change the value for each parameter whenever you run the job.

> **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*.

Relational Table Settings

For relational connections that support writing directly to the database, you can configure the following settings to specify the output table in the Run Job page.

> **NOTE:** You cannot write to multiple relational outputs from the same job.

Steps:

1. **Select location:** Navigate the database 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 database 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** or **Update**.

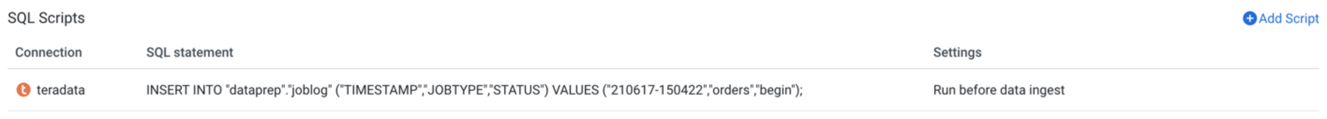
SQL Scripts Panel

Feature Availability: This feature may not be available in all product editions. For more information on available features, see *Compare Editions*.

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.



Connection	SQL statement	Settings
teradata	INSERT INTO "dataprep"."joblog" ("TIMESTAMP","JOBTYPE","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.

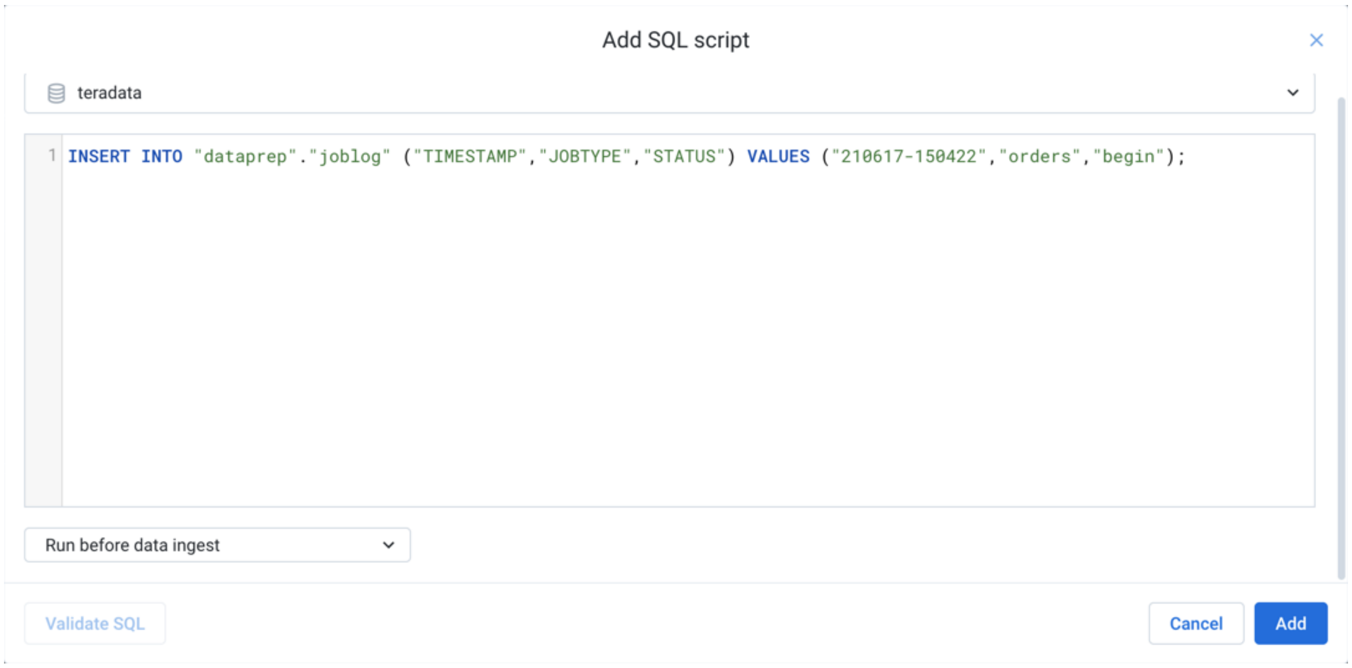
    NOTE: If publishing job fails, then all downstream tasks also fail, including the SQL script, which is not executed and is recorded as a failed phase of the job execution.

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.



```
INSERT INTO "dataprep"."joblog" ("TIMESTAMP","JOBTYPE","STATUS") VALUES ("210617-150422","orders","begin");
```

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*.

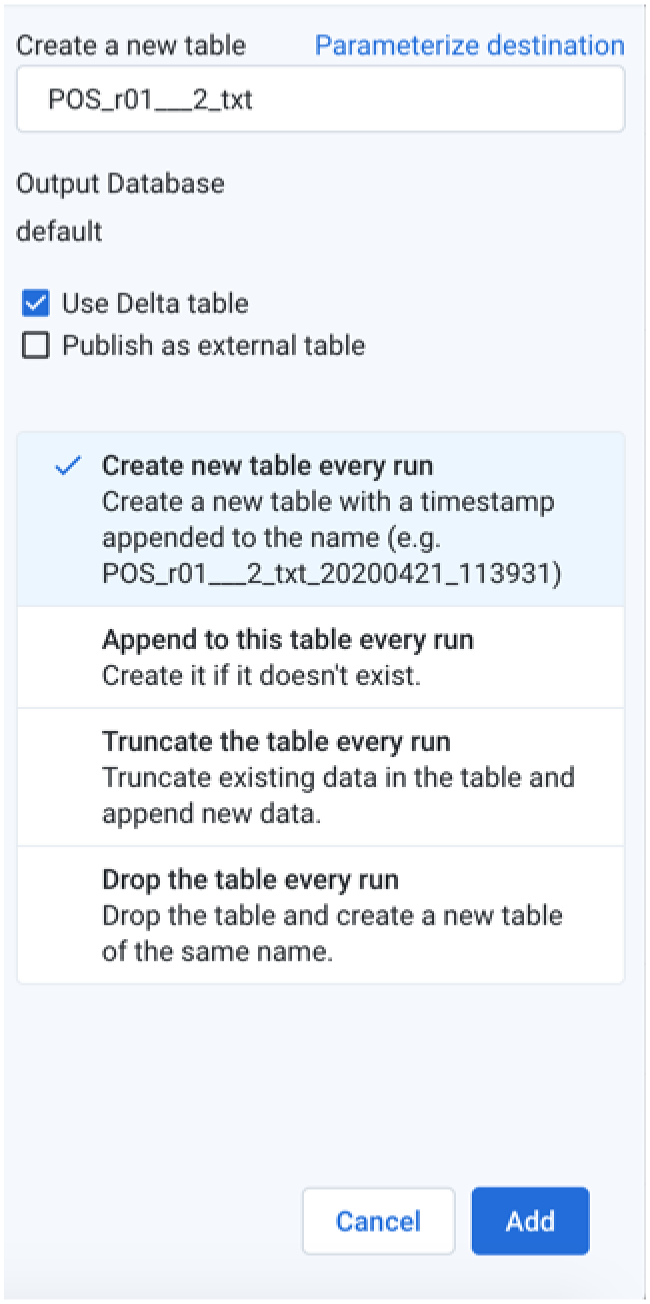
        iii. **Truncate the table every run:** With each run, all data in the table is truncated and replaced with any new results.
        iv. **Drop the table every run:** With each run, the table is dropped (deleted), and all data is deleted. A new table with the same name is created, and any new results 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.

Figure: Databricks Tables table settings

Steps:

1. **Select location:** Navigate the Databricks Tables browser to select the database and table to which to publish.
    1. To create a new table, click **Create a new table**.
2. **Select table options:**
    1. **Table name:**
        1. New table: enter a name for it. You may use a pre-existing table name, and schema checks are performed against it.
        2. Existing table: you cannot modify the name.

            > **NOTE:** Writing to partitioned tables is not supported.

    2. **Output database:** To change the database to which you are publishing, click the Databricks icon in the sidebar. Select a different database.
    3. **Optional table types:** Select one or more table types to publish as well:
        1. **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.

        2. **Publish as external table:** Output is published as an external table to the specified location in your bucket.
    4. **Publish actions:** Depending on your selection or selections above, the following publishing actions on the table are supported:
        1. **Create new table every run:** Each run generates a new table with a timestamp appended to the name.
        2. **Append to this table every run:** Each run adds any new results to the end of the table.
        3. **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.

        4. **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

⚠ **Feature Availability:** This feature may not be available in all product editions. For more information on available features, see *Compare Editions*.

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.

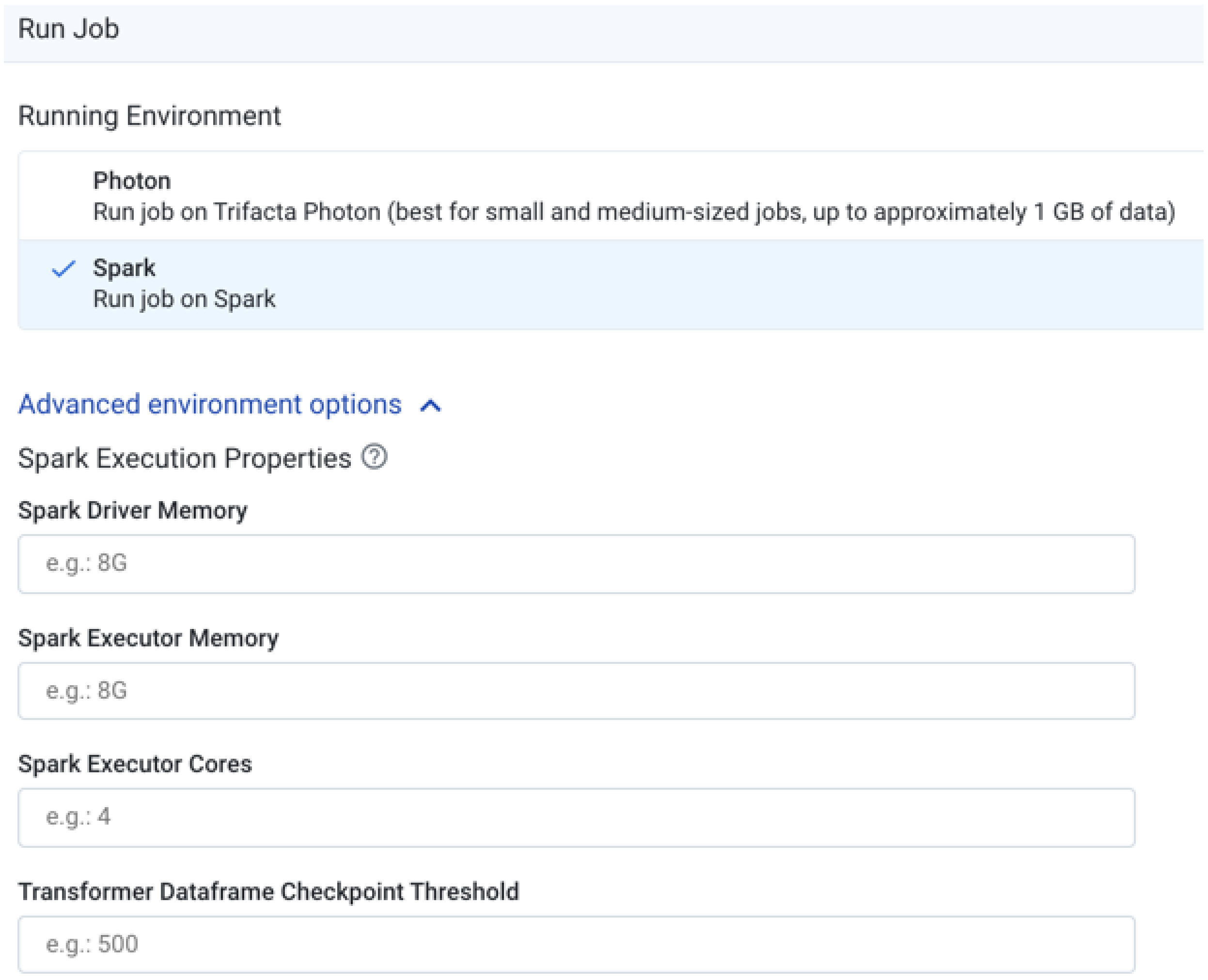
> **NOTE:** A workspace administrator must enable Spark job overrides and configure the set of available parameters. For more information, see *Enable Spark Job Overrides*.

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	Amount of physical RAM in GB on each Spark node that is made available for the Spark drivers. By raising this number: • The drivers for your job are allocated more memory on each Spark node. • There is less memory available for other uses on the node.
Spark Executor Memory	Amount of physical RAM in GB on each Spark node that is made available for the Spark executors. By raising this number: • The Spark executors for your job are allocated more memory. • There is less memory available for other uses on the node.
Spark Executor Cores	Number of cores on each Spark executor that is made available to Spark. By raising this number: • The maximum number of cores available for your job is raised on each Spark executor. • There are fewer cores for other uses on the node.
Transformer Dataframe Checkpoint Threshold	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. By raising this number: • You increase the upper limit of steps between checkpoints. • You may reduce processing time. • It may result in a higher number of job failures.

For more details on setting these parameters, see *Tune Cluster Performance*.

Other Spark overrides:

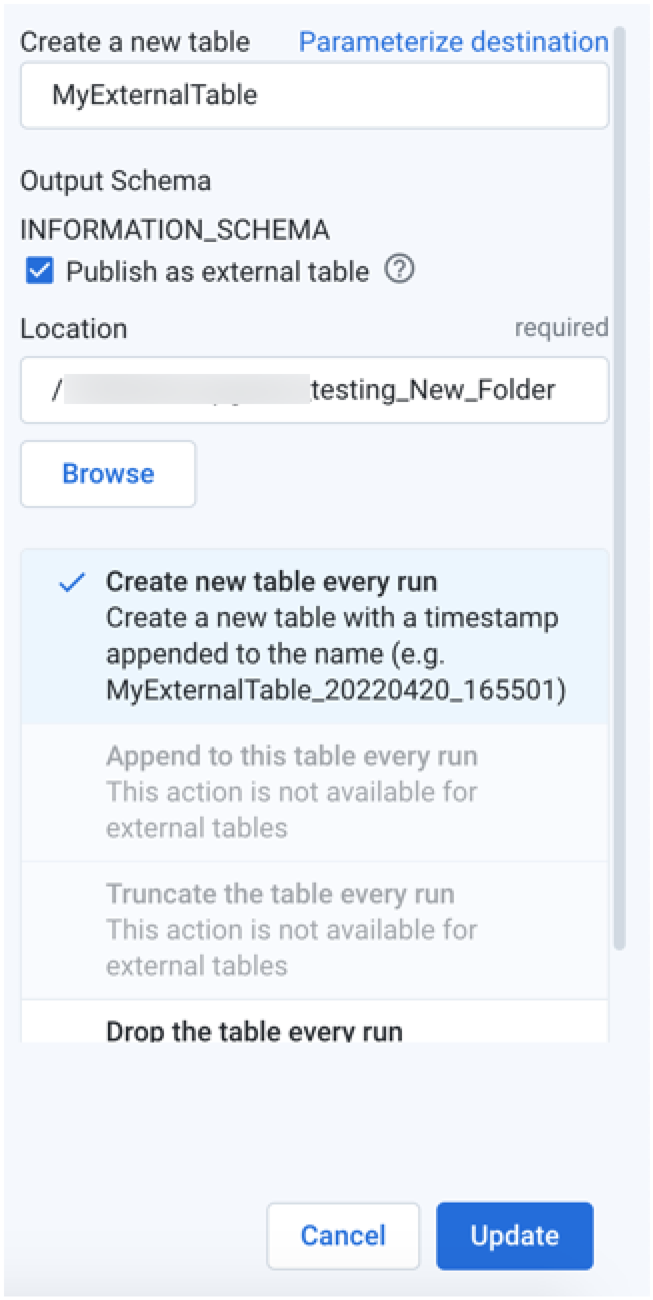
Your workspace administrator may have enabled other Spark properties to be overridden. These parameters appear at the bottom of the list.

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

/[redacted]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.

> **NOTE:** When publishing to an external table under ADLS user mode, the system credentials are used to write to the storage location and must have the appropriate permissions.

    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**.

Snowflake Table Settings

When publishing to Snowflake , please complete the following steps to configure the table and settings to apply to the publish action.

Steps:

1. **Select location:** Navigate the Snowflake 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. **Publish actions:** Select one of the following.
        i. **Append to this table every run:** Each run adds any new results to the end of the table.
        ii. **Truncate the table every run:** With each run, all data in the table is truncated and replaced with any new results.
        iii. **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.
        iv. **Merge the table every run:** This publishing option merges the rows in your results with any existing rows in the target Snowflake table. For more information, see Merge Table Operations below.
3. To save the publishing action, click **Add** or **Update**.

Merge Table Operations

NOTE: When a merge operation is performed on a Snowflake table, profiling information displayed in the Job Details page is incorrectly based on source data. Downloaded profile information is accurate. This is a known issue. As a workaround, you can create a second flow whose input is the output of the first. This flow can be run when profiling information is needed. See *Build Sequence of Datasets*.

NOTE: When a merge operation is applied to a Snowflake table that is referenced in another publication action, the merge operation should be blocked by the Designer Cloud application . This is a known issue.

The publishing option to merge table with every run allows you to update existing rows of data in the target table with corresponding values from your results (merge) and optionally to insert or delete matching rows from your results into the table.

Steps:

1. In the Table Settings panel, select **Merge the table every run**.
2. **Primary keys for matching:** Select one or more columns whose values determine if a row in your source results matches a row in the target. When these key values match, the following columns are updated.

    NOTE: Columns of Array or Object data type cannot be used as key columns for merge operations.

    NOTE: Null values in a key column in the generated output are not considered matches with null values in the key columns of the target table. Instead, the merge operation inserts a new row in the target table containing only the values from the generated output.

    a. If the matching columns have duplicate rows in the target table, all rows in the target are updated.

    b. If the matching columns have duplicate rows in the source, the job fails.
3. **Action on target table for matched rows:** Select the action to apply to the target record when a match is found between the key columns:
    a. **Update:** The values from your results are updated into the columns specified below.
    b. **Delete:** The row in the target table is deleted.
4. **Keys to be updated:** Select one or more columns whose values are updated from your source results when values from the previous set of columns match. These are the columns that are merged into the table.

    Tip: If All Columns is selected, all columns other than the matching columns are updated on a match. All columns continue to be updated even if the schema changes, and the matching columns remain in the schema.

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
Home Page	From the Home page, you can create or access your flows, datasets, and jobs, as well as configure settings and find additional resources.
Flows Page	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.
Transformer Page	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.

Item	Description
Download Logs Dialog	You can download logs for your current session in Designer Cloud Powered by Trifacta® Enterprise Edition. From the Resources menu, select **Download logs.**

Flows

Flows Page

Item	Description
Create Flow Page	You can use flows to organize your datasets and to track the jobs associated with them.
Flow View Page	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
View for Connections	In Flow View, you can review the details of the connections used to access the flow's imported datasets, whether you created it or it was shared with you.
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.
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.
View for Recipes	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.
View for Reference Datasets	A **reference dataset** is a reference to a recipe's output, which can been added to a flow other than the one where the recipe is located.

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.
View for Outputs	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.
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.
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.
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**.
Manage Parameters Dialog	Within a flow, you can create and manage flow parameters, including specifying override values. From the flow menu, select **Parameters**.
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.
Flow Optimization Settings Dialog	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**.

Plans

Library

Library Page

Item	Description
Imported Datasets Page	You can review the imported datasets that you own or have been shared with you.
Dataset Details Page	Use the Dataset Details page to review a dataset's usage and to perform management tasks on it.
Import Data Page	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**.
References Page	A reference dataset is a link to a recipe's output, which can been added to other flows.
Macros Page	In the Macros page, you can review and manage the macros to which you have access.

Import Data Page

Item	Description
Database Browser	The database browser enables you to interact with databases that are connected to the Designer Cloud powered by Trifacta® platform .
File System Browser	In the Designer Cloud powered by Trifacta® platform , 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.

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 De signer Cloud Powered by Trifacta® Enterprise Edition is connected.
S3 Browser	In the Designer Cloud powered by Trifacta® platform , 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.
File Import Settings	When you edit settings on a selected file in the Import Data page, the following settings are displayed.
Table Import Settings	When you edit settings for a selected table in the Import Data page, the following settings are displayed.

Macros Page

Item	Description
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.

Jobs

Transformer

Transformer Page

Item	Description
Data Grid Panel	The data grid in the Transformer Page displays how your current recipe applies to the data in your currently selected sample.
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.
Sample Indicator	At the top of the Transformer page, you can see the Sample Indicator showing the name of the currently displayed sample. If you have not taken a sample, this value is **Initial data**. Click this button to show more information about the sample.
Transfor mer Toolbar	At the top of the data grid and the column browser, the Transformer toolbar provides quick access to common transformations.
Column Menus	You can use the menus available in a column context menu to perform a variety of actions on the column or columns.
Column Browser Panel	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.
Column Details Panel	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.
Transfor m 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.
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.
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.
Visible Columns Panel	In the status bar of the Transformer page, click the Eye icon to review the list of visible and hidden columns.
Join Window	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.

Union Page	In the Union page, you can append data from one or more datasets to an existing dataset.
Run Job Page	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
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).
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.
Lookup Wizard	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...**.
Standardize Page	Through the Standardize page, you can review similar column values and standardize them to values that you specify.

Column Menus

Item	Description
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.
Transformation by Example Page	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.

Context Panel

Item	Description
Recipe Panel	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.
Transform Builder	The Transform Builder enables you to search for transformations and to rapidly assemble complete transform steps through a simple menu-driven interface.
Search Panel	Through the search context panel, you can locate transformations to specify and add at the current location in your recipe.
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.
Samples Panel	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.
Selection Details Panel	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
User Profile Page	In your user profile, you can review your personal information and update your photo. Select **User menu > Preferences**.
Account Settings Page	In your Account Settings page, you can change your locale and modify other settings related to your account.
Email Notifications Page	You can configure the notifications that are sent to your email address and other related settings.
Sessions Page	The Sessions page enables you to view the number of devices signed into your account. You can use this information to make sure no one else has signed in to your account. You can review the devices that are authorized and revoke any unfamiliar devices.
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.
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.

User Profile Page

Item	Description
Storage Config Page	The Storage Config page allows you to configure storage locations for where you upload datasets and generate results.

Connections Page

Item	Description
Create Connection Window	Through the Create Connection window, you can create and edit connections between the Designer Cloud powered by Trifacta® platform and remote storage.
Share Connection Dialog	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:

This section contains information on the fie 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.

File path length limits

Maximum character limits for file paths:

- **File paths to sources for imported datasets:** `1024`

  Tip: This limit (`storagelocations`) applies to both files and tables.

- **File paths to output files:** `2048`

  Tip: This limit (`writesettings`) applies to files stored on any file-based storage location.

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:**

  ```
  "/"
  ```

- **Seb 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.

- Plain Text
- LOG
- TSV
- Parquet

  > **NOTE:** When working with datasets sourced from Parquet files, lineage information and the `$sourcerownumber` reference are not supported.

- Avro

  NOTE: When working with datasets sourced from Avro files, lineage information and the `$sourcerownumber` reference are not supported.

- Google Sheets

  ⚠ **Feature Availability:** This feature may not be available in all product editions. For more information on available features, see *Compare Editions*.

  NOTE: Individual users must enable access to their Google Drive. No data other than Google Sheets is read from Google Drive.

- XML

  Tip: XML files can be ingested as unstructured text.

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. If you are encountering issues, try to Save As to XLS or XLSX from within the Microsoft Excel application.

  Tip: You may import multiple worksheets from a single workbook at one time. See *Import Excel Data* in the User Guide.

- Google Sheets

  ⚠ **Feature Availability:** This feature may not be available in all product editions. For more information on available features, see *Compare Editions*.

  Tip: You may import multiple sheets from a single Google Sheet at one time.

- 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*.

Parsing Limits

After a file is passed into the Designer Cloud application , a set of transformations is applied to it to prepare it for use in the application. These transformations may impose additional limits on the file that is imported. When these transformations fail, the file may be imported into the application as a single column of data. For more information, see *Initial Parsing Steps*.

Native Output File Formats

Designer Cloud Powered by Trifacta Enterprise Edition can write to these file formats:

NOTE: Some output formats may need to be enabled by an administrator. See *Workspace Settings Page*.

- CSV
- JSON

- 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 of 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: *https://github.com/google/snappy/blob/master/framing_format.txt*
.snappy	Hadoop-snappy format	See: *https://code.google.com/p/hadoop-snappy/* **NOTE:** 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.jvmOptions"`	`-Dorg.xerial.snappy.tempdir=<some executable 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

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.

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 **outlier** value is either: • < (25th percentile) - (2 * IQR) • > (75th percentile) + (2* IQR) • **IQR (interquarterile range)** 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.
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	

	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. • For datasets with an even number of values, the median is the mean of the two middle values.
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.

Supported Data Types

This section provides information on supported data types in Designer Cloud Powered by Trifacta® Enterprise Edition.

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.

Supported Data Types

Item	Description
String Data Type	Any non-null value can be typed as String. A String can be anything.
Integer Data Type	The Integer data type applies to positive and negative numeric values that have no decimal point.
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`.
Boolean Data Type	The Boolean data type expresses true or false values.
Social Security Number Data Type	This data type is applied to numeric data following the pattern for United States Social Security numbers.
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.
Email Address Data Type	This data type matches text values that are properly formatted email addresses.
Credit Card Data Type	Credit card numbers are numeric data that follow the 14-digit or 16-digit patterns for credit cards.
Gender Data Type	This data type matches a variety of text patterns for expressing male/female distinctions.
Zip Code Data Type	This data type matches five- and nine-digit U.S. zipcode 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.
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.
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.
IP Address Data Type	The IP Address data type supports IPv4 address.
URL Data Type	URL data type is applied to data that follows generalized patterns of URLs.
HTTP Code Data Type	Values of these data types are three-digit numeric values, which correspond to recognized HTTP Status Codes.
Datetime Data Type	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.

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.

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 valida tes 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:

```
#### #### #### ####
####-####-####-####
#### #### #### ##
####-####-####-##
```

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.

Some numerical combinations in the above patterns may be:

- widely known test values
- invalid for credit card numbers

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.

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	• `California` • `Arizona` • `New York`
Abbreviations	• `CA` • `AZ` • `NY`

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 `\n` and `\t` escaped characters are supported in inputs for this data type.

NOTE: The Designer Cloud application recognizes up to 250 unique keys in a column of Object data type.

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 `\n` and `\t` escaped characters are supported in inputs for this data type.

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.

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:

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 in Designer Cloud Powered by Trifacta Enterprise Edition. This behavior may be different from source and target systems. 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 **NOTE:** Two-digit values for the year that are older than 80 years from the current year are forward-ported into the future in Designer Cloud Powered by Trifacta Enterprise Edition. This behavior may be different from source and target systems. See "Two-digit year values" below.
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) **NOTE:** Requires an AM/PM indicator (a).	Number	2
hh	Hour in am/pm (01-12) **NOTE:** Requires an AM/PM indicator (a).	Number	02
H	Hour in day (1-12)	Number	2
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 using the UNIXTIME function. If you want to work with local times, you can truncate the time zone or use other Datetime functions.

Two-digit year values

Depending on the system, a two-digit value for year in a Datetime value is subject to different interpretations. In D esigner 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)*.
- 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 D esigner 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:

Transformation Name	`Replace text or pattern`
Parameter: Column	`myDateColumn`
Parameter: Find	`/\b([456789][0-9])\b$/`
Parameter: Replace with	`19$1`

and

Transformation Name	`Replace text or pattern`
Parameter: Column	`myDateColumn`
Parameter: Find	`/\b([0123][0-9])\b$/`
Parameter: Replace with	`20$1`

Supported Datetime Formats

For more information on the available formats and examples of each one, see *Datetime Formats (PDF)*.

You can use the DATEFORMAT function to modify the formatting of your Datetime values.

Supported Time Zones

Time zones values (e.g. `UTC-08:00`) are supported.

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.

  Tip: This use of time zone applies to any Spark-based running environment, such as EMR.

- 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 execution properties from the Run Job page:

```
"spark":
  "props": {
    ...
    "spark.driver.extraJavaOptions" : "-Duser.timezone=\"UTC\"",
    "spark.executor.extraJavaOptions" : "-Duser.timezone=\"UTC\""
  }
  ...
}
```

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 `Datetime`.	`"Datetime"`	`"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*yyyy"`	`"shortMonth*dd*yy"`

Supported Numeric Formatting

Contents:

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.	`'###,###'`	99 999 1000 10000	99 999 1,000 10,000
0	Indicate required digits. If a digit is not available in the source, inserts zero in the data.	`'00.##'`	20 7.1	20.00 07.1
$	You can add constants values to the expression. For example, you can insert currency markers at the beginning of your expression. **NOTE**: The following currency formats are supported: $", "€", "£", "¥", "", "", "NT$", "R$", "R", "Rs", "Kr Whitespace is respected, except in the following case.	`'$ ##.##'`	20 2514.22 6.6666	$ 20 $ 2514.22 $ 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 £ 123456.78	$123 456.78 $123 456.78
%	Percentage expressions can be at the back of the number formatting expression. **NOTE:** 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 14.22 6.6666	20 % 1422 % 666.67 %
-	Negative value indicators can be added to the front part of the number formatting string.	'-###,###.00'	123 -123	-123.00 --123.00

	• Negative value indicators at the end of the number are not supported. • 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.		1234.56 -1234.56	-1234.56 --1234.56

Key Codes as Separator Values

Some functions support the use of specifying key codes for grouping and decimal separators:

- `NUMFORMAT Function` 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.
- `NUMVALUE Function` Converts a string formatted as a number into an Integer or Decimal value by parsing out the specified decimal and group separators. A string or a function returning formatted numbers of String type or a column containing formatted numbers of string type can be inputs.

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 (`.`)	`NUMFORMAT(SUM(1000000,DIVIDE(1,100)),'###,###.00',',','.')`	`1,000,000.01`
Spanish locale	Period (`.`)	Comma (`,`)	`NUMFORMAT(SUM(1000000,DIVIDE(1,100)),'###,###.00','.',',')`	`1.000.000,01`
French locale	Space	Comma (`,`)	`NUMFORMAT(SUM(1000000,DIVIDE(1,100)),'###,###.00',' ',',')`	`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:

This section provides information on type conversions in Designer Cloud application .

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 and how those lines are interpreted by the type system.

Type inference can be disabled for individual datasets. For more information, see *Disable Type Inference*.

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.

Type Conversions

Item	**Description**
Avro Data Type Conversions	This section covers data type conversions between the Designer Cloud® application and the Avro file format.
DB2 Data Type Conversions	This section covers data type conversions between the Designer Cloud® application and DB2 databases.
Hive Data Type Conversions	This section covers data type conversions between the Designer Cloud® application and Hive.
Oracle Data Type Conversions	This section covers data type conversions between the Designer Cloud® application and Oracle databases.
MySQL Data Type Conversions	This section covers data type conversions between the Designer Cloud® application and MySQL databases.
Parquet Data Type Conversions	This section covers data type conversions between the Designer Cloud® application and the

	Parquet file format.
Postgres Data Type Conversions	This section covers data type conversions between the Designer Cloud® application and PostgreSQL databases.
Redshift Data Type Conversions	This section covers data type conversions between the Designer Cloud® application and Redshift.
Snowflake Data Type Conversions	This section covers data type conversions between the Designer Cloud® application and Snowflake databases.
AWS Glue Data Type Conversions	This section covers data type conversions between the Designer Cloud® application and AWS Glue.
Salesforce Data Type Conversions	This section covers data type conversions between the Designer Cloud® application and Salesforce.
SQL Server Data Type Conversions	This section covers data type conversions between the Designer Cloud® application and SQL Server databases.
SQL DW Data Type Conversions	This section covers data type conversions between the Designer Cloud® application and SQL DW datastores.
Databricks Tables Data Type Conversions	This section covers data type conversions between the Designer Cloud® application and Databricks Tables.
Tableau Hyper Data Type Conversions	This section covers data type conversions between the Designer Cloud® application and Tableau Hyper format.
Teradata Data Type Conversions	This section covers data type conversions between the Designer Cloud® application and Teradata databases.
SharePoint Data Type Conversions	This section covers data type conversions between the Designer Cloud® application and SharePoint.

Avro Data Type Conversions

This section covers data type conversions between the Designer Cloud® application and the Avro file format.

> **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

Feature Availability: This feature may not be available in all product editions. For more information on available features, see *Compare Editions*.

This section covers data type conversions between the Designer Cloud® application and DB2 databases.

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:

This section covers data type conversions between the Designer Cloud® application and Hive.

> **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	**NOTE:** 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	**NOTE:** On import, some float columns may be interpreted as Integer data type in the D esigner 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	**NOTE:** 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 *Overview of Target Schema Mapping*.
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

Feature Availability: This feature may not be available in all product editions. For more information on available features, see *Compare Editions*.

This section covers data type conversions between the Designer Cloud® application and Oracle databases.

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.

NOTE: You cannot disable type inference for Oracle sources. This is a known issue.

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
XDBUIRType	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 (256)	Oracle VARCHAR column type must include the maximum number of permitted characters.
Map	VARCHAR (256)	Oracle VARCHAR column type must include the maximum number of permitted characters.
Array	VARCHAR (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

Feature Availability: This feature may not be available in all product editions. For more information on available features, see *Compare Editions*.

This section covers data type conversions between the Designer Cloud® application and MySQL databases.

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
lontext	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	
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

This section covers data type conversions between the Designer Cloud® application and the Parquet file format.

> **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).

Postgres Data Type Conversions

Feature Availability: This feature may not be available in all product editions. For more information on available features, see *Compare Editions*.

This section covers data type conversions between the Designer Cloud® application and PostgreSQL databases.

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:

This section covers data type conversions between the Designer Cloud® application and Redshift.

> **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

This section covers data type conversions between the Designer Cloud® application and Snowflake databases.

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 1)	Datetime
TIMESTAMP_NTZ	Y 2)	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

This section covers data type conversions between the Designer Cloud® application and AWS Glue.

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

Feature Availability: This feature may not be available in all product editions. For more information on available features, see *Compare Editions*.

This section covers data type conversions between the Designer Cloud® application and Salesforce.

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

Feature Availability: This feature may not be available in all product editions. For more information on available features, see *Compare Editions*.

This section covers data type conversions between the Designer Cloud® application and SQL Server databases.

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:

This section covers data type conversions between the Designer Cloud® application and SQL DW datastores.

> **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	**NOTE:** 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 *Import Data Page*.
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	
NVARCHAR	Y	String	
SYSNAME	Y	String	
BINARY	Y	String	
VARBINARY	Y	String	
UNIQUEIDENTIFIER	Y	String	
TIMESTAMP	N		
GEOGRAPHY	N		
GEOMETRY	N		
HIERARCHYID	N		
IMAGE	N		
TEXT	N		
NTEXT	N		
XML	N		
CURSOR	N		
ROWVERSION	N		
SQL_VARIANT	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 Data Type	SQL DW Data Type	Notes
String	VARCHAR	
Integer	BIGINT	**NOTE:** 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 *Overview of Target Schema Mapping*.

Float	FLOAT	
Bool	BIT	
Datetime	DATETIME2	If a time-only value is published as an append to a pre-defined DATETIME2 column, then the output column is prepended with `1900-01-01`.
Datetime (with time value only)	TIME	If a value is published as an append to a pre-defined TIME column, then any date information is dropped from the output. **NOTE:** The platform publishes to Timestamp columns only for append operations to pre-existing tables.
Datetime	VARCHAR	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. When you publish results from a job through the Publishing dialog to SQL DW, all Datetime column values are written as String type.
Map	VARCHAR	
Array	VARCHAR	
Date	VARCHAR	
All Other Data Types	VARCHAR	

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
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:

This section covers data type conversions between the Designer Cloud® application and Databricks Tables.

> **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	**NOTE:** 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	**NOTE:** On import, some float columns may be interpreted as Integer data type in the D esigner 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	**NOTE:** 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 *Overview of Target Schema Mapping*.
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

This section covers data type conversions between the Designer Cloud® application and Tableau Hyper format.

> **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*.

Teradata Data Type Conversions

Feature Availability: This feature may not be available in all product editions. For more information on available features, see *Compare Editions*.

This section covers data type conversions between the Designer Cloud® application and Teradata databases.

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

Feature Availability: This feature may not be available in all product editions. For more information on available features, see *Compare Editions*.

This section covers data type conversions between the Designer Cloud® application and SharePoint.

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 Note Text Varchar URL Choice	**NOTE:** Secondary types are published only if appending or truncating an existing table. See below.

Integer	Integer	
Decimal	Float Currency	
Datetime	Date and Time	
Bool	Boolean	
Map	Note String Text Varchar	**NOTE:** Secondary types are published only if appending or truncating an existing table. See below.
Array	Note String Text Varchar	**NOTE:** Secondary types are published only if appending or truncating an existing table. See below.
All other data types	String	

For more information, see *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:

- *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*
- *Unlock type*
- *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	**Scale to min max**	Scale a column to a specific min max range. See *Prepare Data for Machine Processing*.
onehotencode	**One hot encode**	Create a column for each unique value indicating its presence or absence. See *Prepare Data for Machine Processing*.
scalestandardize	**Scale to mean**	Scale a column to zero mean and unit variance. See *Prepare Data for Machine Processing*.
bincolumn		Bin values into ranges of equal or custom size. See *Prepare Data for Machine Processing*.

	Bin column	
changetype	**Change column type**	Changes the data type of a column [settype]. See *Change Column Data Type*.
comment	**Comment**	Adds a comment to your recipe [comment]. See *Add Comments to Your Recipe*.
conditions	**Conditional column**	Returns values based on conditions such as if-then-else or case statements. See *Apply Conditional Transformations*.
convertpattern	**Convert patterns**	Finds one or more patterns or text literals and replaces them with specified pattern values. See *Standardize Using Patterns*.
countmatches	**Count matches**	Counts the number of matches [countpattern]. See *Compute Counts*.
countmatchesbetween	**Count matches between delimiters**	Counts the number of matches [countpattern]. See *Compute Counts*.
deduplicate	**Remove duplicate rows**	Removes duplicate rows where values in every column are the same. See *Deduplicate Data*.
derive	**New formula**	Creates a new column with the result of a formula.
drop	**Delete columns**	Delete one or more columns. See *Remove Data*.
extractbetweendelimiters	**Extract between delimiters**	Extracts text found between two patterns. See *Extract Values*.
extractcustom	**Extract text or pattern**	Extracts text found between two patterns. Variant: Custom text or pattern. See *Extract Values*.
extractfirstcharacters	**Extract first**	Extracts text according to its position. Variant: Extract the first n characters. See *Extract Values*.
extractkv	**Convert key/value to Object**	Extracts key-value pairs into an Object [extractkv]. See *Extract Values*.
extractlastcharacters	**Extract last**	Extracts key-value pairs into an Object [extractkv]. Variant: Extract the last n characters. See *Extract Values*.
extractlist	**Extract matches to Array**	Extracts a list into an Array [extractlist]. See *Extract Values*.
extractrangeofcharacters	**Extract between positions**	Extracts text according to its position. Variant: Extract the last n characters. See *Extract Values*.
extractmismatched	**Extract mismatched**	Extracts a list into an Array [extractlist]. Variant: The data type to match against. See *Extract Values*.
extractnumbers	**Extract numbers**	Extracts a list into an Array [extractlist]. Variant: Extract numbers from a text. See *Extract Values*.
extractquerystrings	**Extract query strings**	Extracts a list into an Array [extractlist]. Variant: Extract fields from an URL query string. See *Extract Values*.
filtercontains	**Filter contains**	Filter rows that satisfy a condition. Variant: Filter rows that contain a specified value or pattern. See *Filter Data*.
filtercustom	**Filter custom formula**	Filter rows that satisfy a condition. Variant: Filter rows that satisfy an arbitrary formula. See *Filter Data*.
filterendswith	**Filter ends with**	Filter rows that satisfy a condition. Variant: Filter rows that ends with a specified value or pattern. See *Filter Data*.
filterexactly	**Filter exact**	Filter rows that satisfy a condition. Variant: Filter rows that match exactly a specified value. See *Filter Data*.
filternot	**Filter not equals**	Filters rows that do not satisfy a condition. See *Filter Data*.
filterfromtop	**Filter from top**	Filter rows by their position. Variant: Filter rows from the top. See *Filter Data*.

filtergreaterth an	**Filter greater than**	Filter rows that satisfy a condition. Variant: Filter rows with values greater than (or equal to) a specified value. See *Filter Data*.
filterinterval	**Filter at interval**	Filter rows by their position. Variant: . Variant: The size of the interval to filter rows at. See *Filter Data*.
filterlessthan	**Filter less than**	Filter rows that satisfy a condition. Variant: Filter rows with values less than (or equal to) a specified value. See *Filter Data*.
filtermissing	**Filter missing**	Filter rows that satisfy a condition. Variant: Filter rows with missing values. See *Remove Data*.
filtermismatc hed	**Filter mismatched**	Filter rows that satisfy a condition. Variant: Filter rows with mismatched values. See *Filter Data*.
filteroneof	**Filter in**	Filter rows that satisfy a condition. Variant: Filter rows that match any of the specified values. See *Filter Data*.
filterrange	**Filter range**	Filter rows by their position. Variant: Filter rows within a range. See *Filter Data*.
filterstartswith	**Filter starts with**	Filter rows that satisfy a condition. Variant: Filter rows that starts with a specified value or pattern. See *Filter Data*.
flatten	**Expand Array to rows**	Converts each element in an Array into a new row. See *Working with Arrays*.
groupby	**Group by**	Group data and perform aggregated calculations on it. See *Create Aggregations*.
join	**Join datasets**	Adds additional columns from other data sources [join]. See *Join Window*.
locktype	**Lock type**	Lock column to current type.
lowercase	**Lowercase text**	Format text in columns. Variant: Convert text in column to lowercase. See *Modify String Values*.
leftpad	**Pad with leading**	Format text in columns. Variant: Add the necessary number of characters to each value to make them of the same length. See *Modify String Values*.
merge	**Merge columns**	Concatenates the values from two or more columns into a new column [merge]. See *Add Two Columns*.
move	**Move columns**	Moves one or more columns before or after another column [move]. See *Move Columns*.
nest	**Nest columns**	Converts columns into an Object or Array [nest]. See *Working with Arrays*.
pivot	**Pivot**	Creates a new column for each unique value in a column [pivot]. See *Pivot Data*.
prefix	**Prefix text**	Format text in columns. Variant: Specify a prefix to be added at the beginning of each selected column name. See *Modify String Values*.
propercase	**Propercase text**	Format text in columns. Variant: Convert text in column to ProperCase. See *Modify String Values*.
removesymb ols	**Remove symbols in text**	Format text in columns. Variant: Remove all non-alphanumerical characters from the text. See *Remove Data*.
removewhite space	**Remove whitespace in text**	Format text in columns. Variant: Remove all whitespace found in the text. See *Remove Data*.
removeacce nts	**Remove accents in text**	Remove accent marks from text. See *Modify String Values*.
rename	**Rename columns**	Renames one or more columns [rename]. See *Rename Columns*.
renamepattern	**Rename with pattern**	Renames one or more columns [rename]. See *Rename Columns*.
renameprefix	**Rename with prefix**	Renames one or more columns [rename]. See *Rename Columns*.
renamehead er	**Rename with row(s**	Renames one or more columns [rename]. See *Rename Columns*.

renamesuffix	**Rename with suffix**	Renames one or more columns [rename]. See *Rename Columns*.
renameupper	**Rename to UPPERCASE**	Renames one or more columns [rename]. See *Rename Columns*.
renamelower	**Rename to lowercase**	Renames one or more columns [rename]. See *Rename Columns*.
renamekeepleft	**Rename from beginning**	Renames one or more columns [rename]. See *Rename Columns*.
renamekeepright	**Rename from end**	Renames one or more columns [rename]. See *Rename Columns*.
renamesanitize	**Rename by removing special characters**	Renames one or more columns [rename]. See *Rename Columns*.
replacecell	**Replace cells**	Renames one or more columns [rename]. See *Rename Columns*.
replacepattern	**Replace text or pattern**	Replace text matching a pattern. See *Replace Cell Values*.
replacebetweenpatterns	**Replace between delimiters**	Replace text between delimiters. Variant: Replace text between delimiters. See *Replace Cell Values*.
replacebetweenpositions	**Replace between positions**	Replace text between delimiters. Variant: Replaces text based on position. See *Replace Cell Values*.
replacemismatched	**Replace mismatched**	Replace mismatched values. See *Replace Cell Values*.
replacemissing	**Replace missing**	Replace missing values. See *Replace Cell Values*.
select	**Select**	Create a new table of columns *Select*d from your current dataset. See Select.
set	**Edit with formula**	Sets the values of one or more columns to the result of a formula [set].
sort	**Sort rows**	Sorts the rows based on the values in one or more columns.
splitondelimiter	**Split on text or pattern**	Split by delimiter. Variant: Text or pattern. See *Split Column*.
splitbetweendelimiters	**Split between delimiters**	Split by delimiter. Variant: Between two delimiters. See *Split Column*.
splitmultipledelimiters	**Split with multiple delimiters**	Split by delimiter. Variant: By multiple delimiters. See *Split Column*.
splitpositions	**Split at positions**	Split by character position. Variant: By positions. See *Split Column*.
splitevery	**Split at interval**	Split by character position. Variant: At regular interval. See *Split Column*.
splitbetweenpositions	**Split between positions**	Split by character position. Variant: Between two positions. See *Split Column*.
splitrows	**Split into rows**	Splits raw data into rows [splitrows]. See *Split Column*.
suffix	**Suffix text**	Format text in columns. Variant: Specify a suffix to be added to the end of each selected column name. See *Modify String Values*.
trimwhitespace	**Trim whitespace**	Format text in columns. Variant: Remove all whitespaces found at the beginning and end of the text. See *Modify String Values*.
trimquotes	**Trim quotes**	Format text in columns. Variant: Remove quotes found at the beginning and end of the text. See *Modify String Values*.
udf		Creates a new column with the result of an external function.

	Invoke external function	**Feature Availability:** This feature may not be available in all product editions. For more information on available features, see *Compare Editions*. **NOTE:** This transformation requires additional configuration.
uppercase	**Uppercase text**	Format text in columns. Variant: Convert text in column to UPPERCASE. See *Modify String Values*.
dateformat	**Date format**	Change format for Datetime columns. See *Format Dates*.
union	**Union datasets**	Adds additional rows from other data source [union]. See *Union Page*.
standardize	**Standardize column**	Single-column standardization for standardizing column values. See *Standardize Page*.
columnbyexample	**Create column from examples**	Create a new column by providing example values. See *Create Column by Example*.
unlocktype	**Unlock type**	Unlock column type.
unnest	**Unnest elements**	Extracts elements from an Object or Array into columns. See *Working with Arrays*.
unpivot	**Unpivot**	Turns columns into rows. Produces a key column with unnested values. See *Pivot Data*.
valuestocols	**Convert values to columns**	Creates a new column for each unique value in a column [valuestocols]. See *Pivot Data*.
window	**Window**	Performs row-based calculations across multiple ordered rows [window]. See *Window Functions*.
sourcerownumber	**sourcerownumber**	Generate a new column containing the row number for each row from the source, if available. See *Source Metadata References*.
filepath	**filepath**	Generate a new column containing the path to the source file, if available. See *Source Metadata References*.

Character Encoding

Contents:

This section describes how Designer Cloud Powered by Trifacta® Enterprise Edition manages character encoding on import, within the application, and on export.

Overview of Character Encoding

Character encoding refers to the mechanism by which numeric digital data is used to represent characters, including alphanumeric characters and punctuation, in languages around the world. To ensure that different machines can represent the same thing on-screen, each machine can reference one or more of the supported file encoding types, which are standards for representation of characters. For example, a machine in the United Kingdom will represent the letter "A" sent from a machine in the United States if they are using the same encoding file encoding types.

In many languages around the world, the representation of all characters requires hundreds and even thousands of characters. As a result, encodings for these regions may require a larger number of bits to represent all aspects of the language.

The platform supports a global file encoding type. By default, this encoding type is UTF-8. For more information, see *Configure Global File Encoding Type*.

Character Encoding on Input

By default, Designer Cloud Powered by Trifacta Enterprise Edition supports UTF-8 on input. As needed, individual users can change the file encoding of input files. For example, a file that is ingested with a double-byte encoding can be identified as such for the product in the file settings during import, so that the data can be properly parsed during input.

Character Encoding within the Application

Within the Designer Cloud application , you can use the following functions to modify character encodings:

Item	Description
BASE64ENCODE Function	Converts an input value to base64 encoding with optional padding with an equals sign (=). Input can be of any type. Output type is String.
BASE64DECODE Function	Converts an input base64 value to text. Output type is String.
UNICODE Function	Generates the Unicode index value for the first character of the input string.

Character Encoding on Output

All files are published with UTF-8 encoding.

Supported File Encoding Types

The following is the list of supported global file encoding types for input and output.

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:

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 Job History pages.
- Applying a Sort transform.
- Applying the ARRAYSORT function to an array.

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:

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.

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.

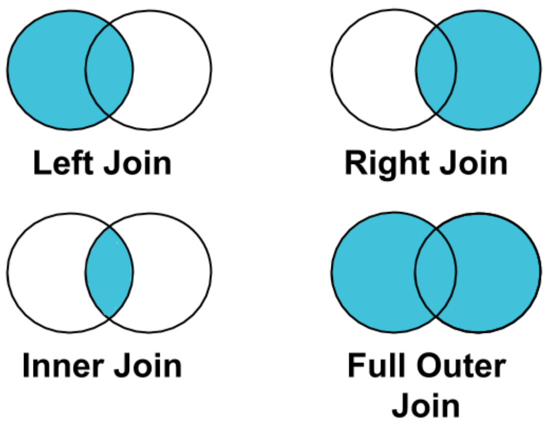


Figure: Join Types

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.

Sample Types

Contents:

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 == &apos;100&apos;
```

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 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.

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:

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
```