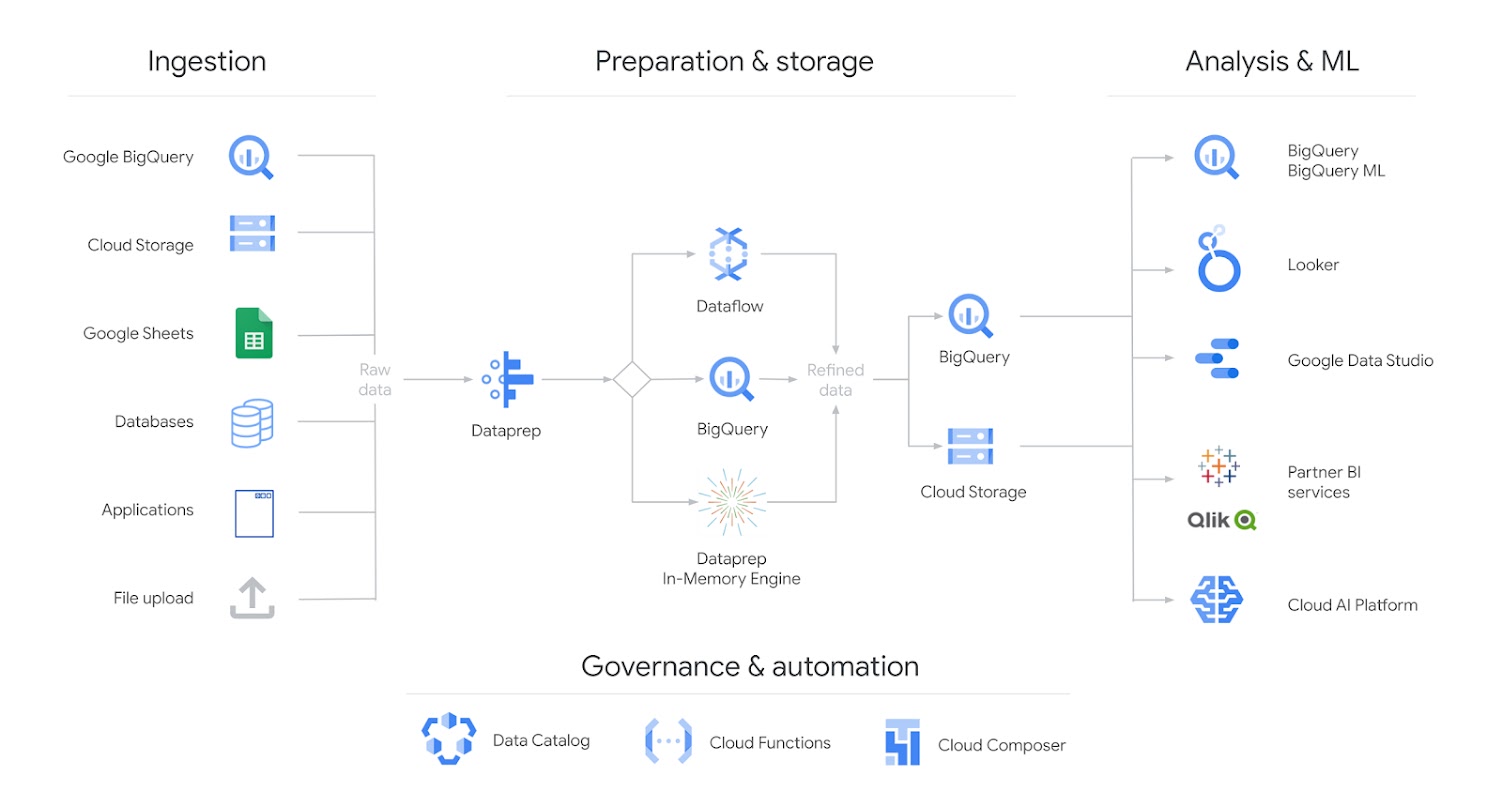
Cloud Dataprep

Cloud Dataprep is **an intelligent data preparation** web application on the Google Cloud Platform that has been offered by Trifacta, a privately owned software company with headquarters in San Francisco.

This tool is very useful for data analysts and data scientists to **visually explore, clean, and transform a dataset** using a simple workflow of drag and drop transformations in a user-friendly interface instead of writing complicated code.

It is a serverless and fully managed service that can drastically reduce and simplify the amount of work required to get the data ready for analysis, reporting, and machine learning.

In this notes, I´ll give an introduction to this tool. The target audience is people on the beginner and intermediate levels.



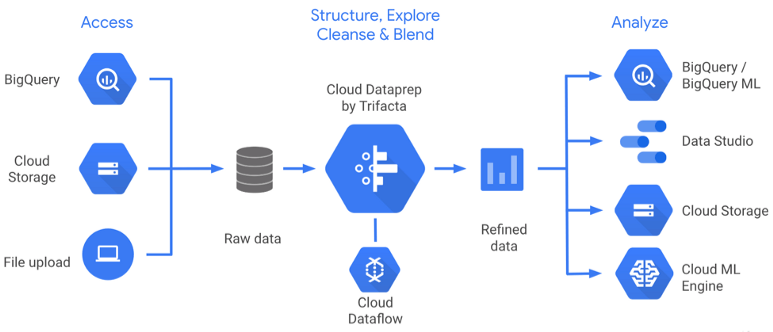


Photo by [Trifacta](https://www.trifacta.com/blog/general-availability-google-cloud-dataprep/" \t "_blank)

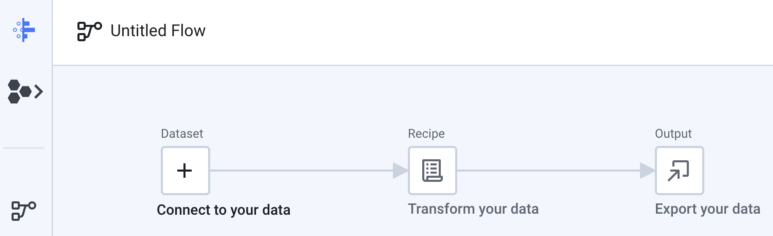
Part A — Create Flows and Import Datasets

In the first step, it is required to create a flow, a container object to access and manipulate a dataset.

It is possible to import multiple datasets from a staging google storage bucket, by uploading a local file or directly from BigQuery.

The file formats supported are CSV, JSON, Plain text, LOG, TSV, Parquet, Avro, Google Sheets, or Excel (XLS/XLSX).

Once the data has been imported, Dataprep will automatically detect the schema, the data type, the distribution of values, and outliers.



Here below you can find a summarized sequence of steps required to create a flow and import a dataset:

Step 1 — Create and name a flow (Blank or from Template)

Step 2 — Click on connect your data

Step 3 — Click on Import a dataset by selecting the appropriate source

Step 4 — Click on import and add to flow

Part B — Apply Recipe Transformations

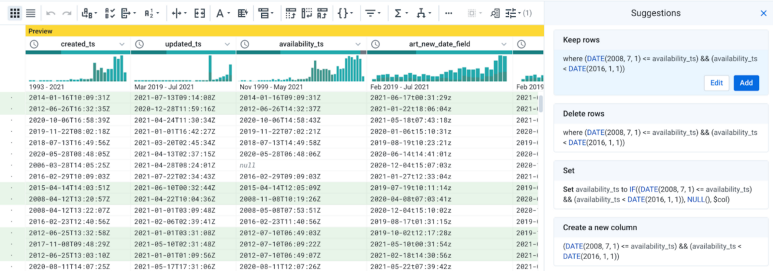
By clicking on edit transformation, Dataprep will load an initial random sample of the dataset. This initial sample will be the first 10MB of rows in the dataset (or the complete dataset if it is smaller than 10MB).

Additional samples based on certain criteria might be created after for deeper analysis of certain features, by clicking on the top left corner on “initial data” and then clicking to collect a new sample.

The samples can be created at random, by selecting the first rows, by a filter-based condition, anomaly-based, stratified, or cluster-based condition. To filter the column by data type it is required to analyze the column of a specific type (for example all the timestamp columns).

It is possible to select a specific interval of data by directly clicking on the distribution graph of a particular column.

Once you select a column, a column category, and/or eventually a specified range, on the right will appear a suggestion box with a list of potentially relevant transformations to be applied.



*Dataprep Wrangler Page with Suggestion Box*

The available transformations can be found on the top bar and include for example the following main functionalities:

Data cleaning:

* Change the name of columns
* Change the data type of a column
* Removing duplicates
* Modify, extract or concatenate strings
* Dates formatting

**Data Discovery:**

* Filter data according to specified conditions
* Identify outliers
* Calculate aggregations like count or sum

Data Enrichment:

* Generate primary keys
* Joining multiple tables
* Adding metadata and lookup tables

Advanced Operations:

* Split columns
* Pivoting tables
* Manipulating arrays and JSON files
* Identify missing and mismatched data

A reference explanation of all the transformations available in Dataprep can be found [here](https://docs.trifacta.com/display/DP/Transformation+Reference).



*Dataprep Wrangler Page*

On the screenshot above you can see in the red box the available functions and in the blue box the column filter.

Here below you can find a summarized sequence of the steps required to apply the transformation to a dataset:

Step 5 — Click on edit recipe

Step 6 — Select a column or a column group from the filter icon

Step 7 — Eventually select a range of values from the distribution graph or by editing the transformation wrangler code

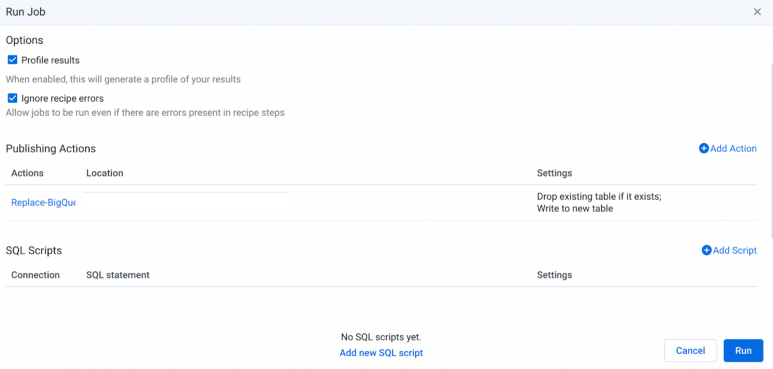
Step 8 — Apply multiple recipe transformations and/or filter columns

Step 9 — Create a new sample of the data and repeat the steps above until you are satisfied

Part C — Run Dataflow Job

Once you are satisfied with the current transformation recipe (i.e. the list of transformations to be applied), eventually applied to multiple samples, you can consolidate the dataset into a new file by running a dataflow job.

The output location is set by default to a new cloud storage bucket, but it can also be imported directly into Big Query if necessary. You can follow the dataflow job directly into the Dataflow console. The data can be exported in a compressed CSV, a compressed JSON file, or in Avro. It is also possible to use Automator to schedule the execution in Dataflow periodically or according to a specific schedule.



*Dataprep Run Job Page*

Here below you can find a summarized sequence of the steps required to export the transformed dataset:

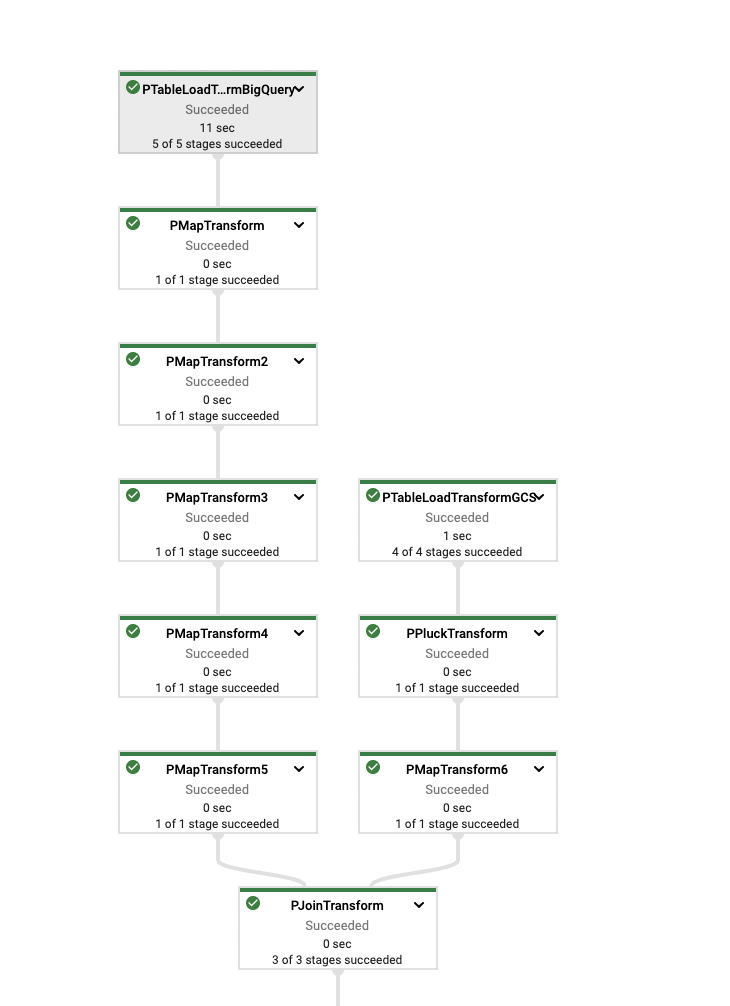
Step 10 — Click on Run on the top right corner

Step 11 — Inside publishing action select the destination file name and type (either in Cloud Storage or Big Query). It needs to be in the same location of the source file

Step 12 — Eventually add a SQL script

Step 13 — Click on Run on the bottom right corner

Step 14 — Wait for the Dataflow Job to complete and you will find your new transformed dataset in the specified location



*Dataflow Pipeline*

Limitations

The size of the samples is limited to 10MB and it is possible to directly upload a file of a maximum of 1 GB.

Although the import supports a variety of file encoding types, the application only displays UTF-8 encodings (UTF-32 is not supported).

Furthermore, the maps generated in visual profiling with a State and a Zip Code column only display data for the United States.

According to the edition and the [Dataprep subscription](https://docs.trifacta.com/display/DP/Product+Editions" \t "_blank) selected there might be further limitations.

Conclusion

In this blog post, the Cloud Dataprep Tool was briefly introduced, with a simple introduction of steps in order to use it, some examples of potential transformation to be applied and its main limitations. This tool is particularly useful to clean and analyze a dataset before doing any other complicated ETL transformation.

The best way to familiarize yourself with Cloud Dataprep is by trying it out directly. Have fun with Data Analysis!

More detailed information about specific Dataprep functions might be published in future blog posts from datadice.

**Creating a Data Pipeline using Google Cloud Dataprep**

**Purpose of the article:**  
This article uses to create a data pipeline using cloud dataprep to load from google cloud storage bucket to google bigquery by performing cleaning and transformation of data.

**Intended Audience:**  
Everyone

**Tools and Technology:**  
Tool-Cloud Dataprep       
Technology- Google cloud platform

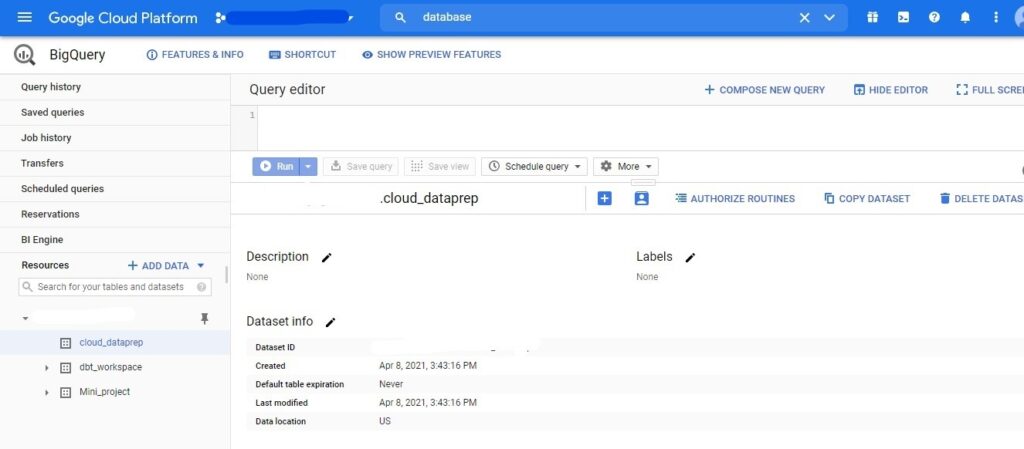
**Keywords:**  
BigQuery, GCS, Dataprep, Dataflow recipe, Data quality, Dataset, Cleaning, enrich, transformation.

**Description:**  
Cloud Dataprep isa google cloud service for visually exploring, cleaning, and transforming structured and unstructured data for analytics, reporting, and machine learning**.** This tool dynamically scales the resources. Let’s create a data pipeline to transform data from google cloud storage to google BigQuery using Cloud Dataprep.

**TASK-1:**

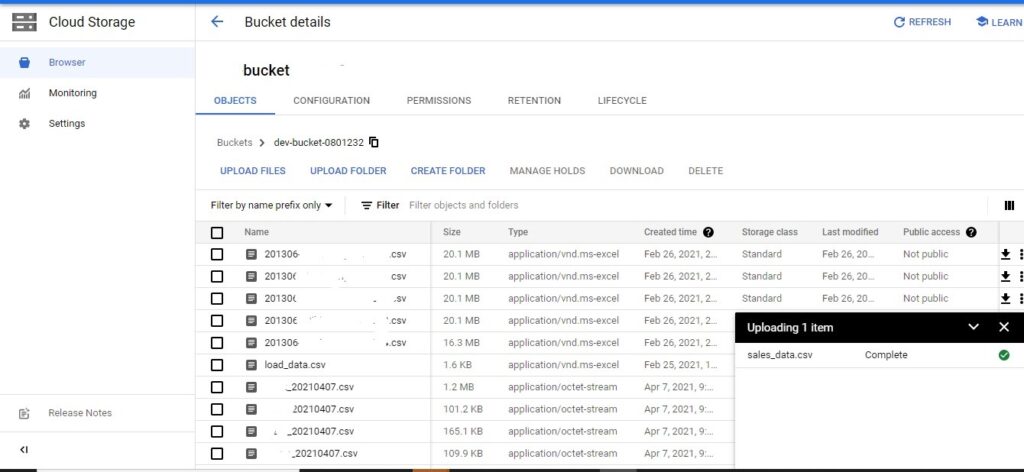
In the Navigation menu, go to the BigQuery and Create a dataset in BigQuery and load the sales\_data file into the source, i.e., google cloud storage bucket.

**Step 1:**  
Navigate to BigQuery and create a dataset, enter the dataset\_id and click on create a dataset to create a dataset in BigQuery.



**Step 2:**

* Now navigate to the google cloud storage bucket and load the data into the GCS bucket.

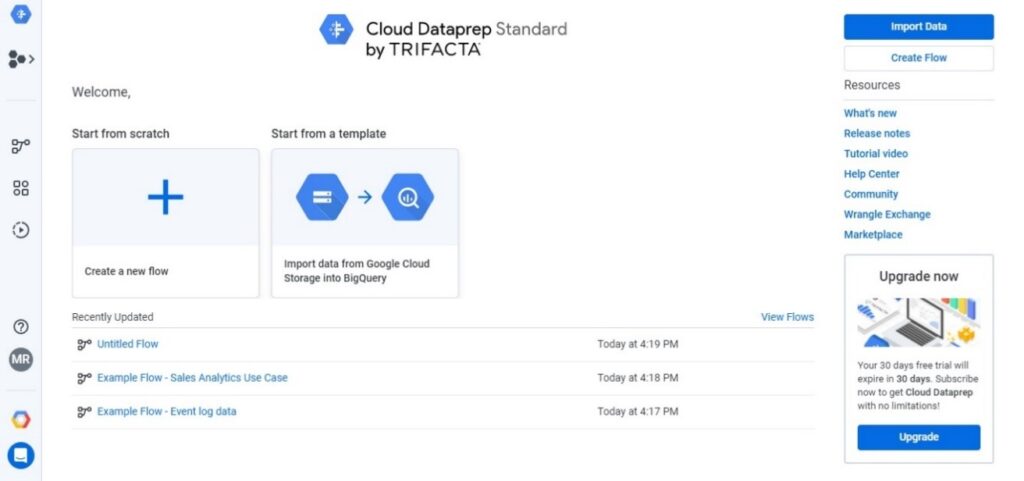


**TASK-2:**

Transforming the data from GCS to BigQuery using cloud data prep below are the steps to be followed.

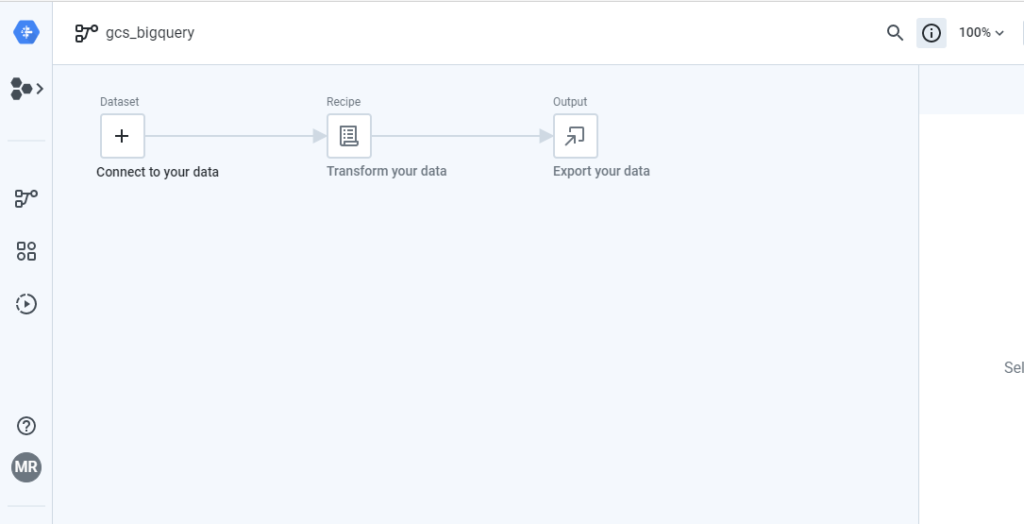
**STEP 1:**

* Click on create a new flow.



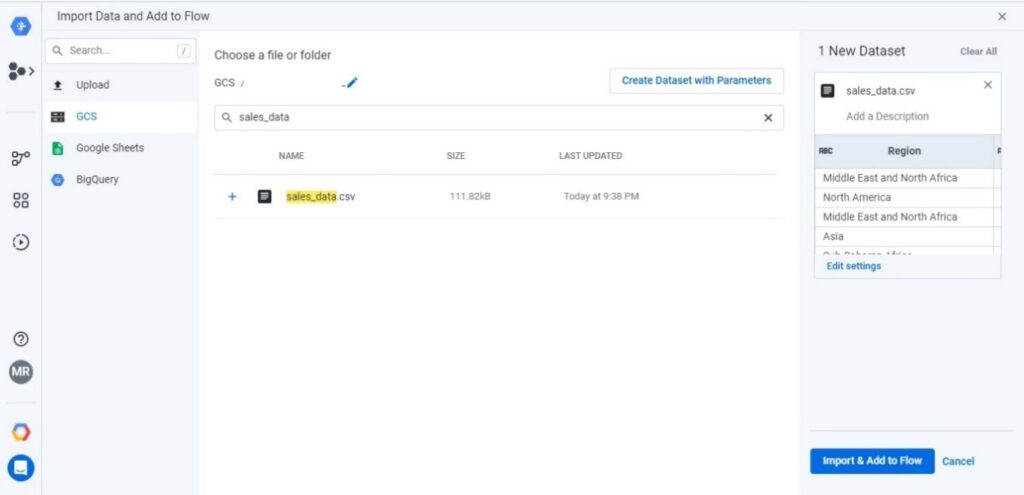
**Step 2:**

* Click on the Add Dataset to get connected to your data.

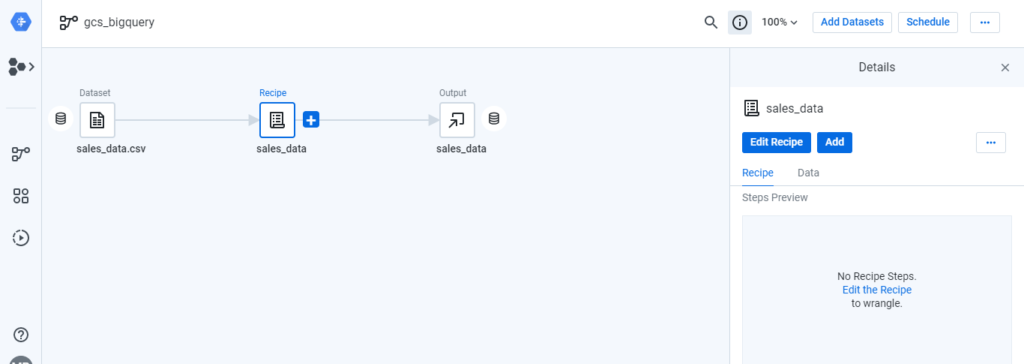


**Step 3:**

* Click on import datasets, Here select GCS as our source data is google cloud storage bucket.
* Click on IMPORT & ADD TO FLOW to add your data.

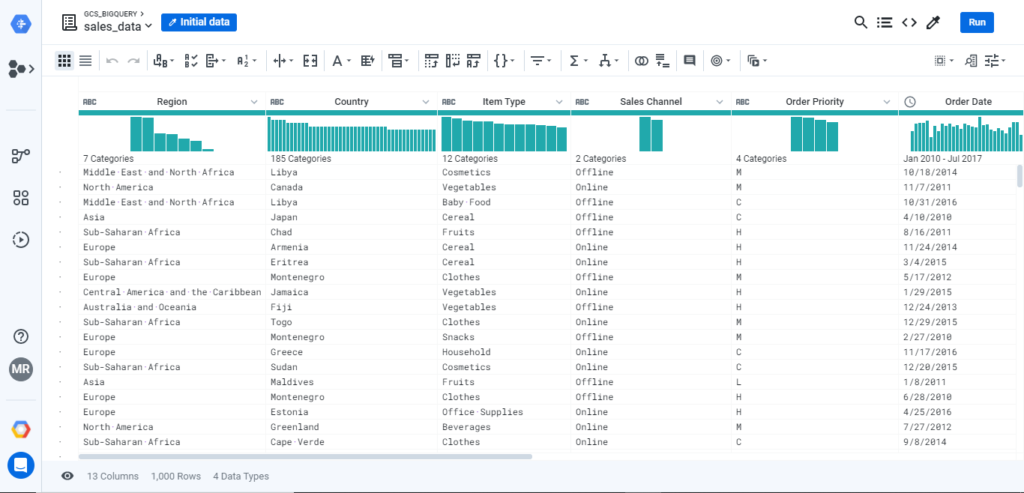


**Step 4:** Imported data added to the recipe. You can even add one or more datasets to the recipe according to your requirement



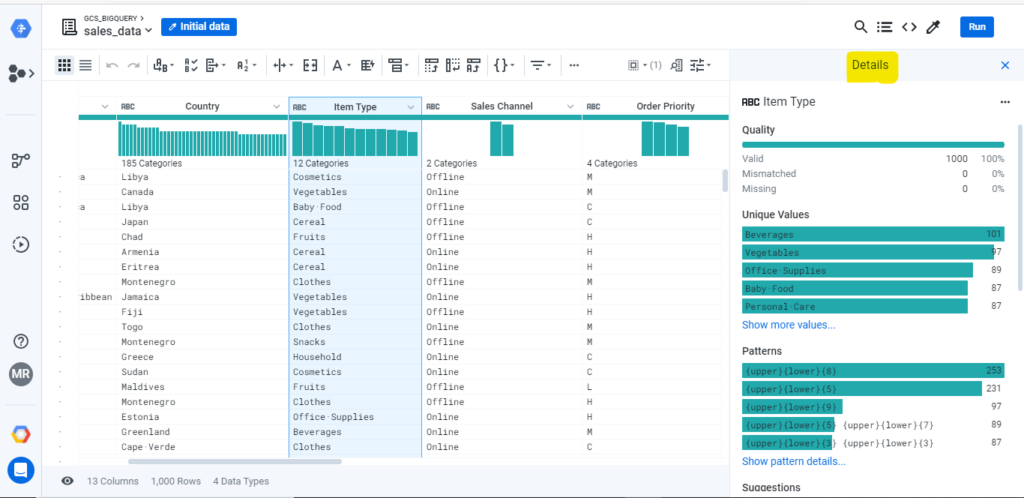
**Step 5:**

* Click on edit recipe. Once you click on edit recipe, you can view the details of the data in the recipe.  
  You can clean, enrich, and transform the data in the recipe.



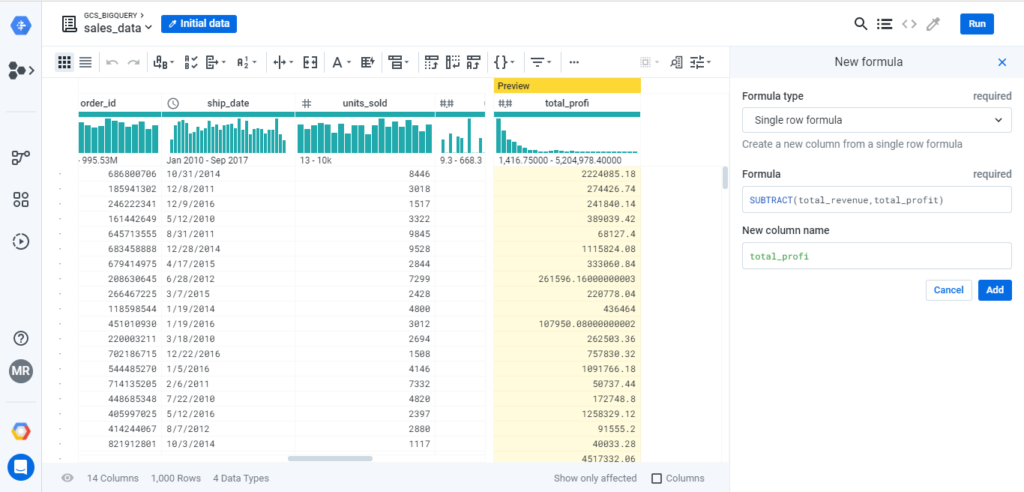
**Step 6 :**

* Click on a single column. It gives us all the details of a column like the Quality of the data, Unique values, and the data pattern if it is a string.



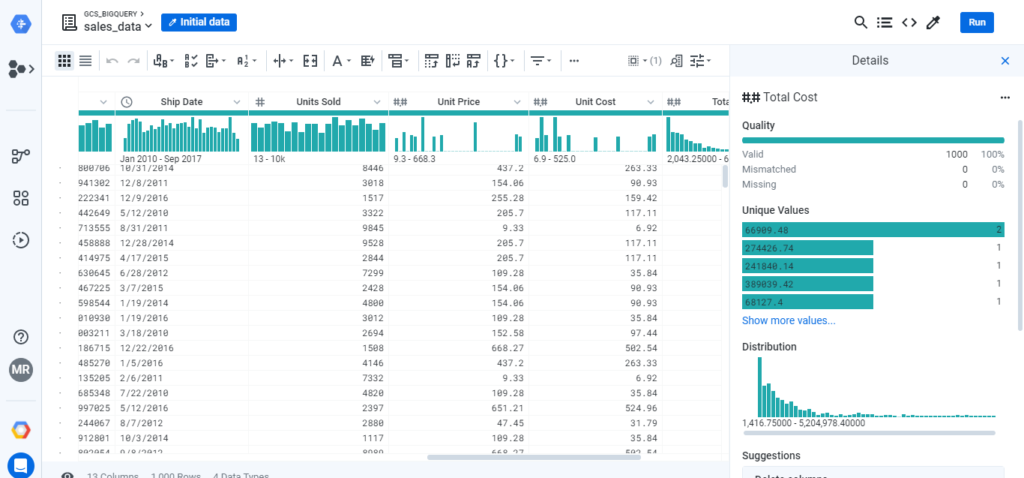
**Step 7:**

* Now click on functions to add a new column total\_profit to the sales\_data added to the dataset.
* Here total\_profit is the difference between total\_revenue and total\_cost
* You can even perform operations like merging the data, perform aggregations, etc.



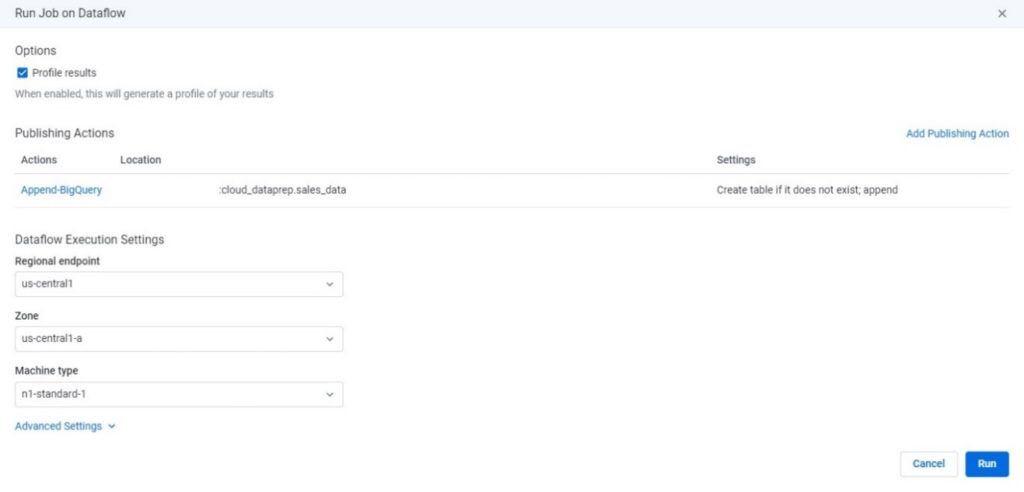
**Step 8:**

* A new column total\_profit is added to the recipe now. After completing the transformation of data, click on Run.



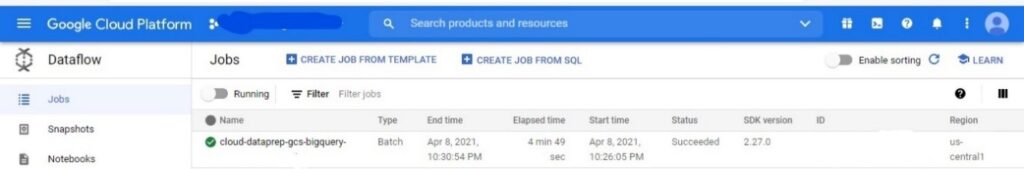
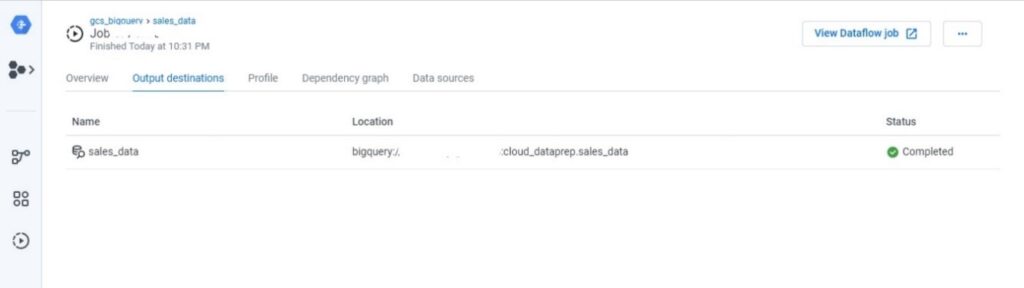
**Step 9:**

* Now add the dataflow execution details and add the output file location.



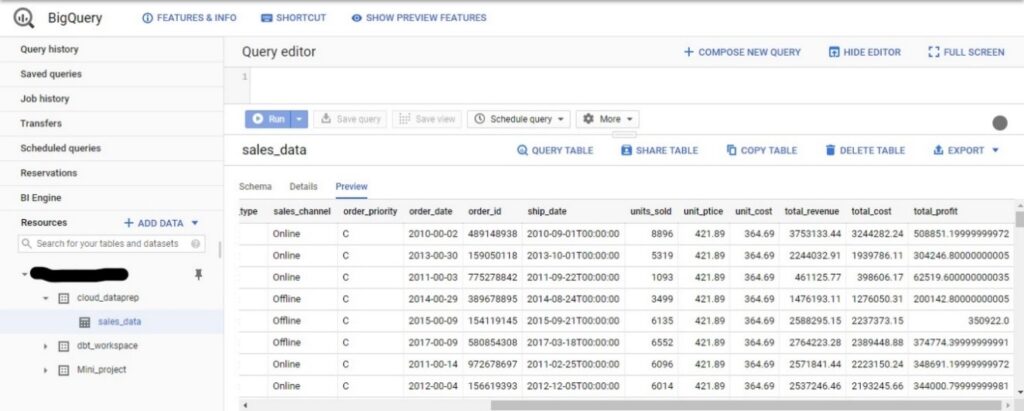
**Step 10:**

* Now click on **RUN.**A dataflow job will create along with the data prep workflow.  
  Dataflow Job uses to extract, transform, and load the data and execute the details of the job flow.



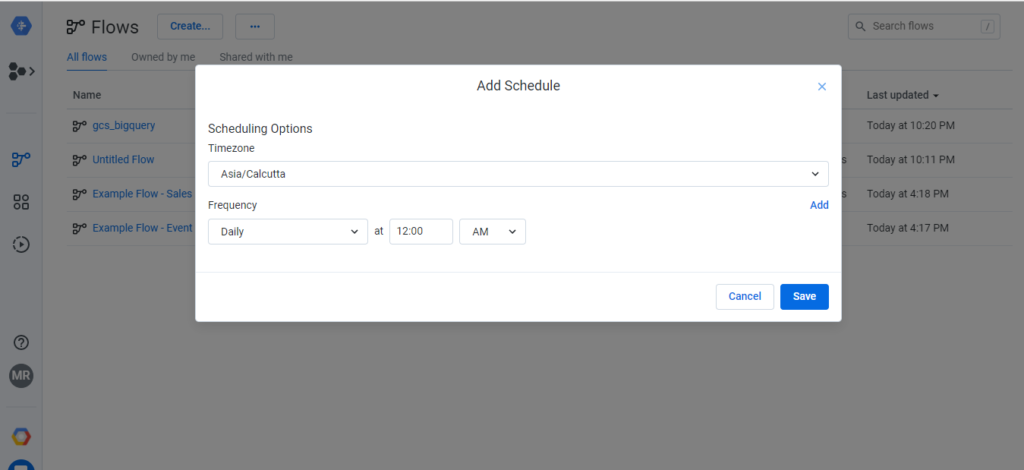
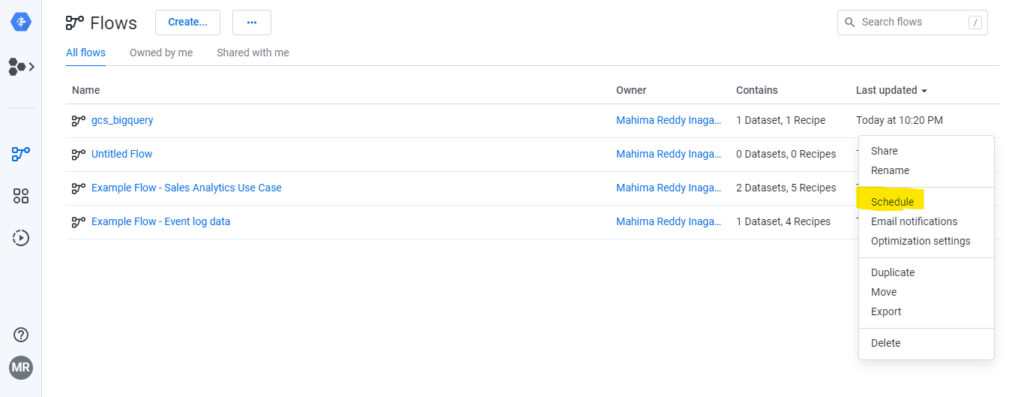
**Step 11:**

* Successfully, the output table developed in BigQuery along with the changes.



**Step 12:**

* Now, please navigate to the options of the dataprep workflow to schedule it.
* Select the scheduling options and click on save to schedule the job.



References / Sources of the information referred:

Google cloud documents

# How to Automatically Deploy a Google Cloud Dataprep Pipeline Between Workspaces

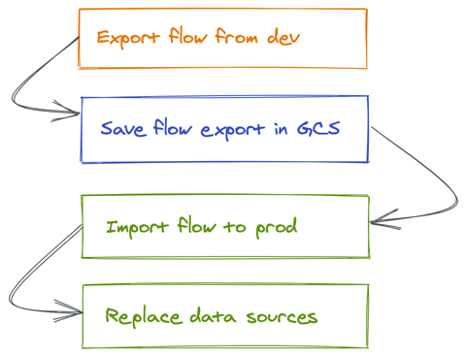
**SPONSORED CONTENT BY TRIFACTA**

Connor Carreras



This article explains how to use Cloud Composer to automate Cloud Dataprep flow migration between two workspaces. This process can be leveraged for your Cloud Data Warehouse project to move from development, test, and production following what is known as Continuous Integration and Continuous Delivery (CI/CD) pipeline in agile development.

At a high level, this pipeline will follow the process shown below:

[](https://2s7gjr373w3x22jf92z99mgm5w-wpengine.netdna-ssl.com/wp-content/uploads/2021/01/Picture1-5.png)

[On Github you can download the Cloud Composer Directed Acyclic Graph (DAG) example file](https://github.com/ccat435/cloud_dataprep_ci-cd/blob/master/dataprep_cicd2.py) that we will be explaining during this article.

We will be using a simple flow, with one Cloud Dataprep input and one recipe, as our example. You’ll be able to leverage this framework to extend it to support you specific environments.

### **Initial Setup**

1. **Prerequisites**

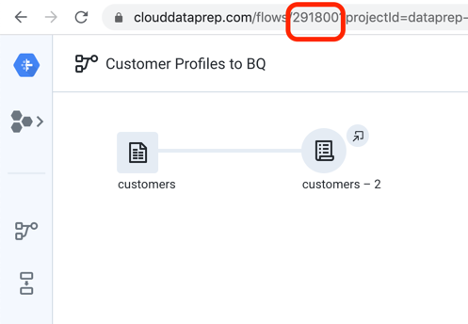
Before creating this CI/CD pipeline, you should ensure that you have performed the following steps in your Google Cloud Platform (GCP) environment:

* You have Cloud Dataprep Premium enabled in each of your projects. Cloud Dataprep Premium is needed to benefit from the APIs needed in the automation process.
* You have “Owner” privileges on each of the GCP projects where Cloud Dataprep Premium is enabled. Using the Cloud Dataprep Premium APIs requires “Owner” privileges.
* You have enabled Cloud Composer in one of your projects. Note that Cloud Composer does not necessarily need to reside in the same project as Cloud Dataprep Premium.

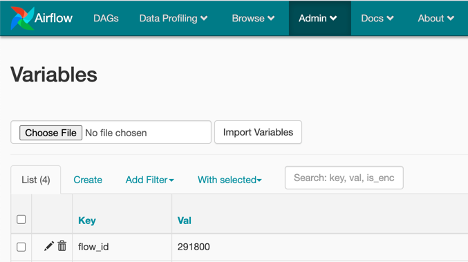
1. **Configure the variables in Cloud Composer**

We need to set up four variables in Cloud Composer. These variables allow you to pass static values into a DAG.

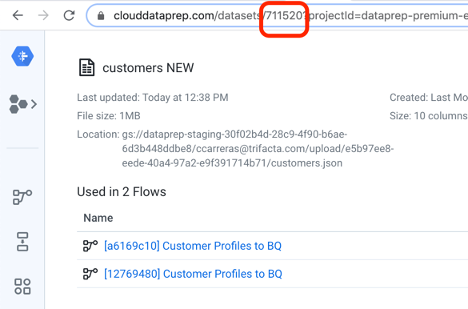
First, we need to store the ID for the flow that we will be exporting from the development environment. From the Cloud Dataprep Premium UI, open the flow that you want to export. In the URL bar, you can find the flow ID, as shown below:

[](https://2s7gjr373w3x22jf92z99mgm5w-wpengine.netdna-ssl.com/wp-content/uploads/2021/01/Picture1-6.png)

Copy this ID. Open the Cloud Composer Admin/Variables interface and create a new variable containing this value.

[](https://2s7gjr373w3x22jf92z99mgm5w-wpengine.netdna-ssl.com/wp-content/uploads/2021/01/Picture1-7.png)

Next, we need to store an ID for the dataset in the production environment that will be associated with your flow. As part of a CI/CD pipeline, we want to fully automate the process of preparing the production flow for deployment. This means replacing the references to datasets in the development environment with dataset references in the production environment. Open your Cloud Dataprep production project and navigate to the source dataset that you want to use. In the URL bar, you can find the dataset ID, as shown below:

[](https://2s7gjr373w3x22jf92z99mgm5w-wpengine.netdna-ssl.com/wp-content/uploads/2021/01/Picture1-8.png)

Copy the dataset ID. Open the Cloud Composer Admin/Variables interface and create a new variable containing this value. If you have multiple source datasets, you should store the IDs for each dataset as an array object inside a single Cloud Composer variable.

The final variables that we need to store in Cloud Composer are the API access tokens for the Cloud Dataprep development and production projects. You can find instructions about how to create API access tokens here: [API Access Token documentation](https://cloud.google.com/dataprep/docs/html/Manage-API-Access-Tokens_145281444). Store each of these access tokens in Cloud Composer following this format: Bearer <token>

1. **Set up an HTTP connection in Cloud Composer**

In Cloud Composer, you need to create a new HTTP connection to point to the Cloud Dataprep API endpoint. Ensure that the connection points to [https://api.clouddataprep.com](https://api.clouddataprep.com/).

### **Developing the Cloud Composer DAG**

[You can access the complete Cloud Composer DAG here.](https://github.com/ccat435/cloud_dataprep_ci-cd/blob/master/dataprep_cicd2.py)

The full DAG includes seven tasks that export the flow from the development environment, save the flow in GCS, import the flow into the production environment, and programmatically identify the correct data sources in the production environment that should be associated with the flow.

1. **Define variable values by their IDs**

As the first step in the DAG, pass in each of the variables that you defined in Cloud Composer. This allows you to invoke the variables in each task in your DAG.

[](https://2s7gjr373w3x22jf92z99mgm5w-wpengine.netdna-ssl.com/wp-content/uploads/2021/01/Picture1-9.png)

1. **Create the task to export the flow and store it in a zip file in GCS**

The first task that we want to perform is exporting the flow from the development project. To do this, you will invoke the getFlowPackage API: <https://api.trifacta.com/dataprep-premium/index.html#operation/getFlowPackage>

Cloud Composer automatically provisions a new GCS bucket for maintaining copies of the DAGs, and storing any input/output data from executed pipelines. The GCS folder that Cloud Composer can access is /home/airflow/gcs/data. The example task shown below writes the API response to the Cloud Composer GCS data folder. This will be the zip file containing the exported flow.

[](https://2s7gjr373w3x22jf92z99mgm5w-wpengine.netdna-ssl.com/wp-content/uploads/2021/01/Picture1-10.png)

Note: As an extension to this task, you could potentially move the exported flow from GCS into a Git environment.

1. **Create the task to import the flow to the production environment**

Once you have saved the flow zip file to GCS, you need to invoke the importPackage API to import the flow zip file into the production Cloud Dataprep environment. Instructions for using this API can be found here: <https://api.trifacta.com/dataprep-premium/index.html#operation/importPackage>

[](https://2s7gjr373w3x22jf92z99mgm5w-wpengine.netdna-ssl.com/wp-content/uploads/2021/01/Picture1-11.png)

As an output, this API call returns metadata about the imported flow, including the ID for the flow in the new environment. This metadata will be critical for the next steps in this pipeline. Our example DAG saves the API response to Xcom, which allows for communication between tasks in the pipeline.

1. **Create the task to retrieve the list of data sources for the imported flow**

When you import your flow into a new environment, the flow retains all of the references to the data sources from the original environment. This means that without replacing the data sources, the imported flow will not be able to execute. We need to replace each of the original data sources in the flow with the corresponding data sources in the new environment.

As a first step, you need to invoke the getFlowInputs API. This will return information about all of the data sources currently associated with the newly imported flow. Note that even though the data sources are invalid, they retain the correct connections with downstream objects.

The code shown below passes the flow ID into the getFlowInput API endpoint. The API call that you made in step 3 would have returned this flow ID as part of the response.

[](https://2s7gjr373w3x22jf92z99mgm5w-wpengine.netdna-ssl.com/wp-content/uploads/2021/01/Picture1-12.png)

Refer to the API documentation (<https://api.trifacta.com/dataprep-premium/index.html#operation/getFlowInputs>) for details about the response from this call. Once again, we need to retain this response for use in downstream tasks.

1. **Create the task to retrieve the list of nodes for the imported flow**

Trifacta’s API for replacing data sources requires you to know the ID for the recipe directly connected to the data source that you want to replace. In order to determine this recipe ID, you need to connect information returned from two API calls: flow nodes and flow edges.

The first set of information that you need to return is the flow nodes data. Flow nodes assign unique IDs to all of the individual objects in a flow. Invoke the listFlows API with an embedded resource named flownodes to retrieve a list of all the nodes in a flow.

Documentation for the listFlows API can be found here: <https://api.trifacta.com/dataprep-premium/index.html#operation/listFlows>

[](https://2s7gjr373w3x22jf92z99mgm5w-wpengine.netdna-ssl.com/wp-content/uploads/2021/01/Picture1-13.png)

This call will return a JSON object structured as shown below:

[](https://2s7gjr373w3x22jf92z99mgm5w-wpengine.netdna-ssl.com/wp-content/uploads/2021/01/Picture1-14.png)

[](https://2s7gjr373w3x22jf92z99mgm5w-wpengine.netdna-ssl.com/wp-content/uploads/2021/01/Picture1-15.png)

You will want to store this response for use in a downstream task.

1. **Create the task to retrieve the list of edges for the imported flow**

Next, we need to return information about flow edges, which identify how each node in a flow is connected to other nodes. Flow edges contain an input node ID, and an output node ID. The flow edges information is the final detail that we need to know in order to connect our data source to a recipe.

As in step 5, you will be invoking the listFlows API. However, this time you will want to embed a call for flowEdges.

[](https://2s7gjr373w3x22jf92z99mgm5w-wpengine.netdna-ssl.com/wp-content/uploads/2021/01/Picture1-16.png)

This call returns the following JSON object:

[](https://2s7gjr373w3x22jf92z99mgm5w-wpengine.netdna-ssl.com/wp-content/uploads/2021/01/Picture1-17.png)

Once again, store this response for use in a downstream task.

1. **Create task to identify the recipe associated with the source dataset**

Now that you have returned information about the flow nodes, flow edges, and imported dataset ID, you are ready to use these outputs to identify the ID for the recipe connected to your data source. You will need to chain the outputs together in the following way:

* When the parsingRecipe.id value returned from step 4 matches the recipe.id value returned from step 5, return the id for that flownode.
* When the flownode id matches the inputFlownode.id returned from step 6, return the outputFlownode id.

The output flownode ID becomes the value that you will enter in the swap datasource API call.

The following Python task shows how you would iteratively navigate through the JSON objects returned in steps 4, 5, and 6, and return the output flownode ID.

[](https://2s7gjr373w3x22jf92z99mgm5w-wpengine.netdna-ssl.com/wp-content/uploads/2021/01/Picture1-18.png)

Note: This code assumes that you only had a single data source in your flow. If your flow contains more data sources, you would want to loop through the array of data sources returned in step 4, and match each data source to the corresponding parsingRecipeID, flownode ID, and output flownode ID.

1. **Create the task to replace the source dataset in the imported flow**

Use the recipe ID that you identified in step 7 to invoke the updateInputDataset API. This API call allows you to replace the existing dev source dataset references with the updated production source dataset references.

You can reference the API documentation here: <https://api.trifacta.com/dataprep-premium/index.html#operation/updateInputDataset>

[](https://2s7gjr373w3x22jf92z99mgm5w-wpengine.netdna-ssl.com/wp-content/uploads/2021/01/Picture1-19.png)

1. **Sequence the tasks**

Finally, to invoke the end-to-end pipeline, you will need to sequence these tasks in your DAG definition:

[](https://2s7gjr373w3x22jf92z99mgm5w-wpengine.netdna-ssl.com/wp-content/uploads/2021/01/Picture1-20.png)

You have now created an end-to-end CI/CD pipeline for exporting a flow from a development project, importing that flow into a production project, and replacing the data sources in the imported flow.

To further extend your CI/CD pipeline, you could trigger automated job execution through Cloud Composer and the Cloud Dataprep APIs. You can refer to this blog for details about how to trigger Cloud Dataprep jobs using Cloud Composer: <https://cloud.google.com/blog/products/data-analytics/how-to-orchestrate-cloud-dataprep-jobs-using-cloud-composer>

# Exploring DataPrep: A Python Library For Data Preparation & EDA

14/08/2021



###### **READ NEXT**

[](https://analyticsindiamag.com/how-nvidia-is-going-all-out-with-its-omniverse-platform/)

##### [**How NVIDIA Is Going All Out With Its Omniverse Platform**](https://analyticsindiamag.com/how-nvidia-is-going-all-out-with-its-omniverse-platform/)

[](https://dataandanalyticsconclave.com/register-now/)

Big Data comes along with its complications. Collecting and managing data properly and the methods used to do so play an important role. With such underlying concerns, the method of Data Preparation becomes very helpful and a crucial aspect to begin with. A good data preparation procedure allows for [efficient analysis](https://analyticsindiamag.com/comprehensive-guide-to-time-series-analysis-using-arima/), limits and minimizes errors and inaccuracies that can occur during processing, making the processed data more accessible to all users. Lately, the process of Data Preparation has gotten easier with new tools and technologies that enable anyone to cleanse and clarify data independently. Data preparation is the process of cleaning and transforming the raw data before preprocessing and analysis. It is a small yet important step before processing and often involves reformatting the data, making corrections, and combining multiple data sets to enrich the present data. Data preparation is often considered a lengthy undertaking for data professionals or business users, but it is an essential prerequisite to put data in context to turn it into insights that might help in decision making, eliminating the bias resulting from poor data quality.

The data preparation process first begins with finding the right data. This can come from an existing data catalogue, warehouse or can be added ad-hoc. After the data is collected, it is important to discover and explore each dataset to prepare and process. This step is essential and helps get to know the data and understand what has to be done before the data can be called useful in a particular context. [Cleaning](https://analyticsindiamag.com/how-to-paraphrase-text-using-pegasus-transformer/) up the data is traditionally another most time-consuming part of the data preparation process, but it’s crucial for removing forward and dealing with faulty data to help fill in the gaps present. Data cleansing is a process where you go through all of the data to be processed and either remove or update information that is considered to be incomplete, incorrect, improperly formatted, duplicated, or highly irrelevant. The Data cleansing process usually also involves cleaning up all the data compiled previously in one area. The data cleansing process is done all at once and can take quite a while if the information has been piling and stacking up for years. That’s why it’s important to perform data cleansing and taking care of the data regularly.

[**Register for this Session>>**](https://register.gotowebinar.com/register/3309501831520626448)

Then, data is further transformed by updating the format or value entries to reach a clean and well-defined outcome or to make the data more easily understood by a wider spectrum of audiences. Once all the mentioned processes are done, the data is prepared. This data can be stored or inculcated into a third-party application, such as a business intelligence tool, clearing the way for processing and analysis. A thorough data preparation process can give an organization many advantages and rather give it a headstart. It must be clean and free of errors before using data for analysis or plugging it into[dashboards](https://analyticsindiamag.com/step-by-step-guide-to-data-analysis-using-sweetviz/) for [visualizations](https://analyticsindiamag.com/complete-guide-to-augly-a-modern-data-augmentation-library/). Preparing data for analysis will help avoid mistakes, saving more time to be invested down the line. These errors will be much more difficult to catch and fix after the data has been transferred out of its original format. Using properly cleaned and formatted data while building [data models](https://analyticsindiamag.com/complete-guide-to-vit-augreg-a-pytorch-image-model-descriptive-predictions/) or applications will ensure top-quality reporting and analysis with proper accuracy. This eventually helps receive game-changing and revolutionary business insights.

## What is DataPrep?

DataPrep is an open-source library available for python that lets you prepare your data using a single library with only a few lines of code. DataPrep can be used to address multiple data-related problems, and the library provides numerous features through which every problem can be solved and taken care of. Using the DataPrep Library, one can collect data from multiple data sources using the dataprep.connector module, perform intense exploratory analysis using the dataprep.eda module and clean and standardize datasets using the dataprep.clean module. DataPrep automatically detects and highlights the insights present in the data, such as missing data, distinct count and statistics.  A whole detailed profile report can be created in a matter of seconds by using just a single line of code, which makes it ten times faster than other libraries to perform data preparation or EDA on.

## Getting Started with the Code

In this article, we will be exploring the different functionalities of the DataPrep library for ease in Data Preparation and EDA, which will help us understand the library even better. The following implementation is partially inspired by the official DataPrep Documentation, which can be accessed using the link [here](https://pypi.org/project/dataprep/).

##### **Installing the Library**

To install the library, you can use the following line of code,

# Run the code to install DataPrep

!pip install dataprep

##### **Installing the Dependencies**

Further, from the DataPrep library itself, we can import the required dependencies for the task to be performed,

import pandas as pd

from dataprep.eda import plot, plot\_correlation, plot\_missing

As we want to create different plots from our dataset, we have imported plot, plot\_correlation to create correlation graphs,plot\_missing to plot the number of missing data.

Now, Lets load our data into the data frame,

#loading data in to the DataPrep dataframe

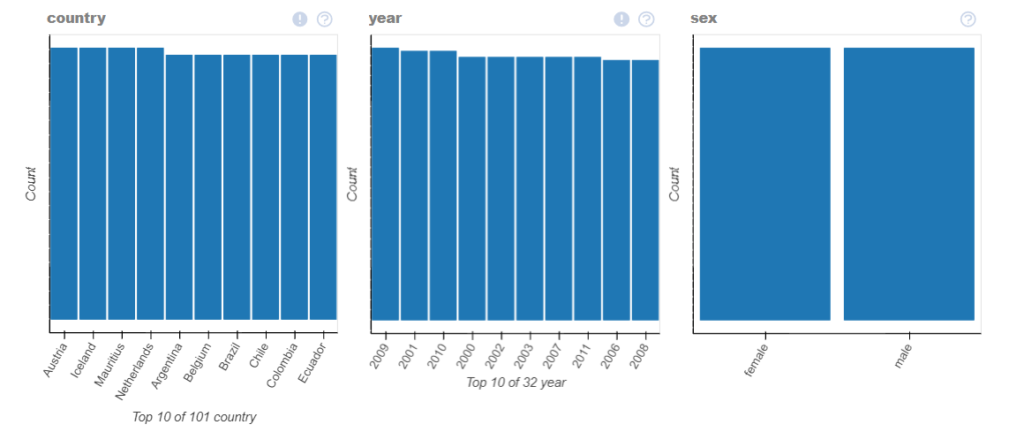
df = pd.read\_csv("/content/master.csv")

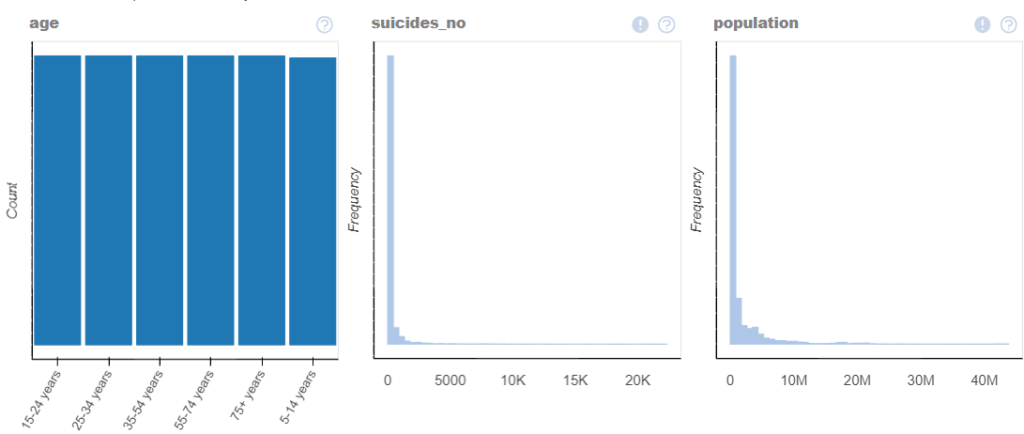
##### **Creating Visualizations**

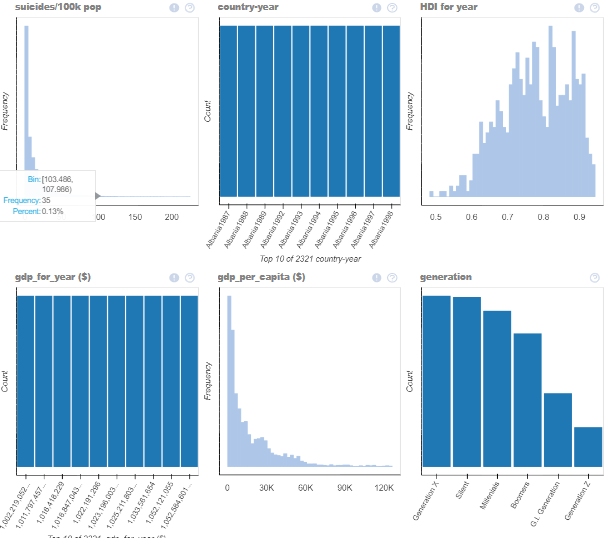
Using DataPrep, we can create all the possible visualizations for the data using just a single line of code. Let us plot our loaded data and see what it looks like,

df["year"] = df["year"].astype("category")

plot(df)





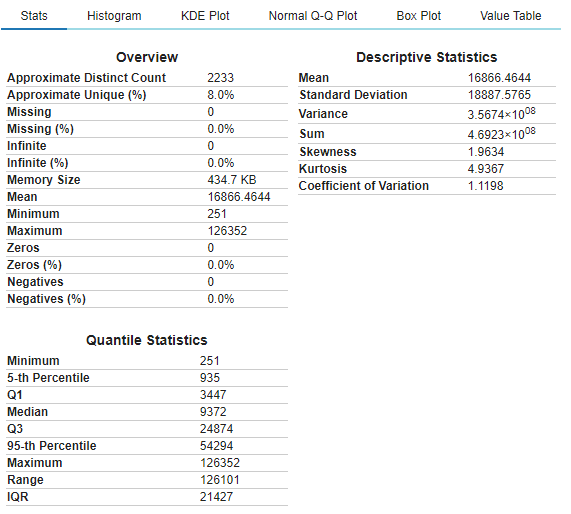


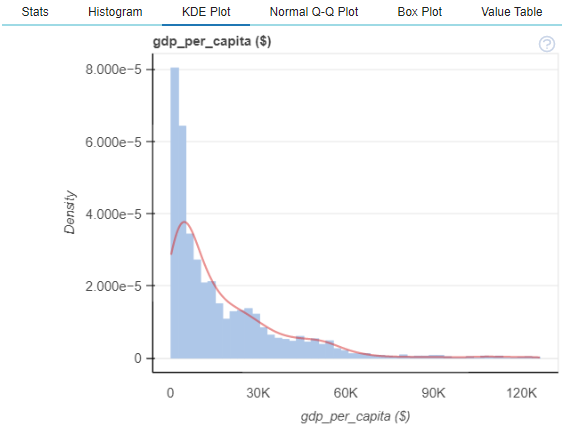
As we can see, the library itself detects the data and plots all the necessary data graphs in a single-window itself!

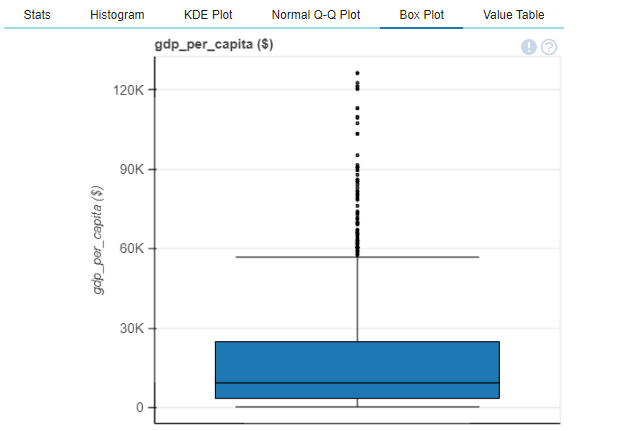
##### **Auto Insight Generation**

You can also get a detailed plot for a single column with all its statistics to understand the column better,

plot(df, "gdp\_per\_capita ($)")







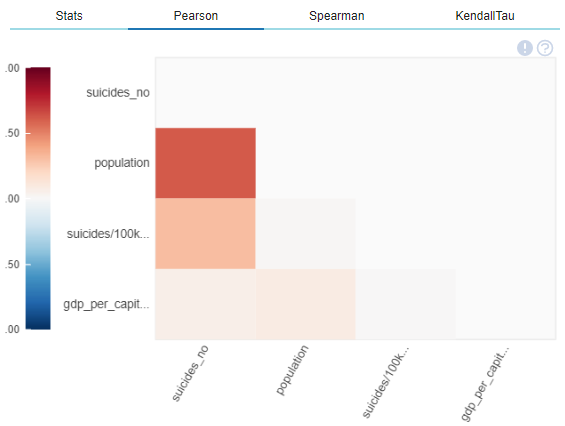
We also create a dataframe, taking care of the missing values and then create a correlation plot,

df\_without\_missing = df.dropna('columns')

plot\_correlation(df\_without\_missing)

plot\_correlation(df\_without\_missing, k=1)

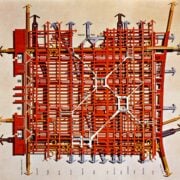
plot\_correlation(df\_without\_missing, value\_range=(0,1)



Whether it be Pearson, Spearman or Kendall-Tau, any correlation graph can be easily plotted using the DataPrep library.

##### **More Operations with DataPrep**

Let us now explore some more operations on another dataset. Here I have used the titanic dataset to perform further functional operations on using DataPrep.

SEE ALSO[](https://analyticsindiamag.com/a-guide-to-multilevel-modeling-in-machine-learning/)

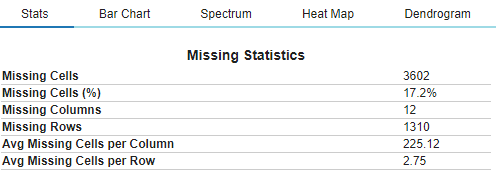
[DEVELOPERS CORNER](https://analyticsindiamag.com/category/developers_corner/)

###### [A Guide to Multilevel Modeling in Machine Learning](https://analyticsindiamag.com/a-guide-to-multilevel-modeling-in-machine-learning/)

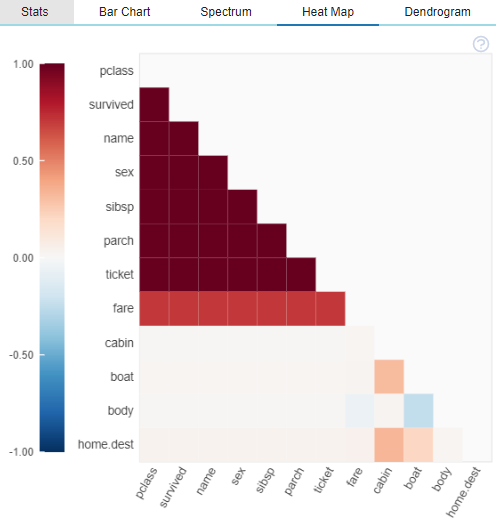
Exploring the missing data in the dataset,

#plot missing

plot\_missing(train\_df)

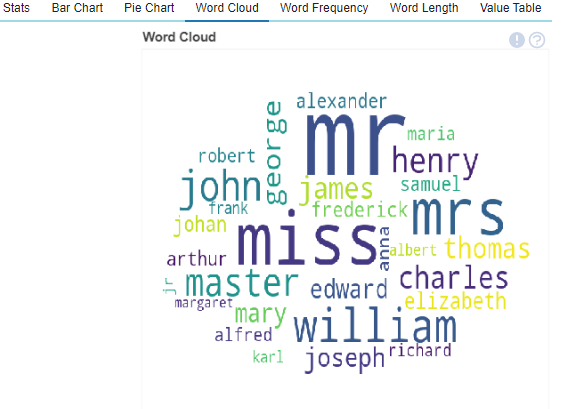


DataPrep will automatically analyse the data and provide the necessary graph such as a Bar Chart, Heat Map or a Dendogram.



Creating Word Clouds have never been easier as well, such functionalities can be used in NLP tasks which will be highly helpful.

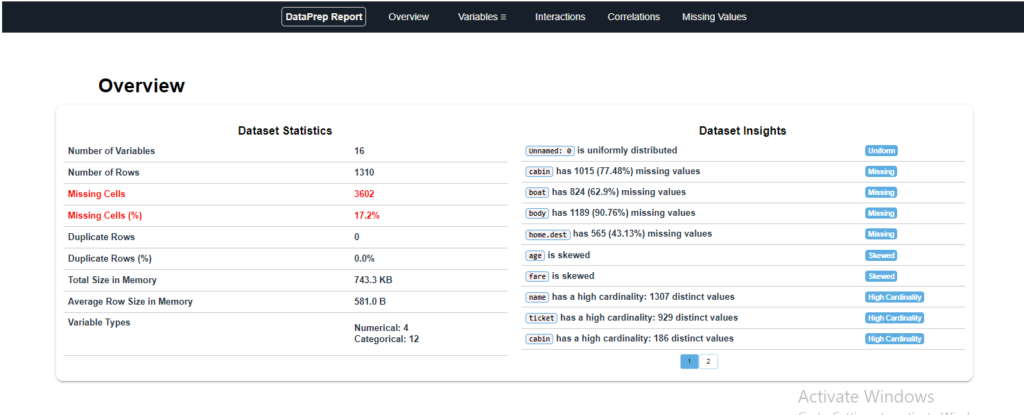
plot(train\_df,'name' )



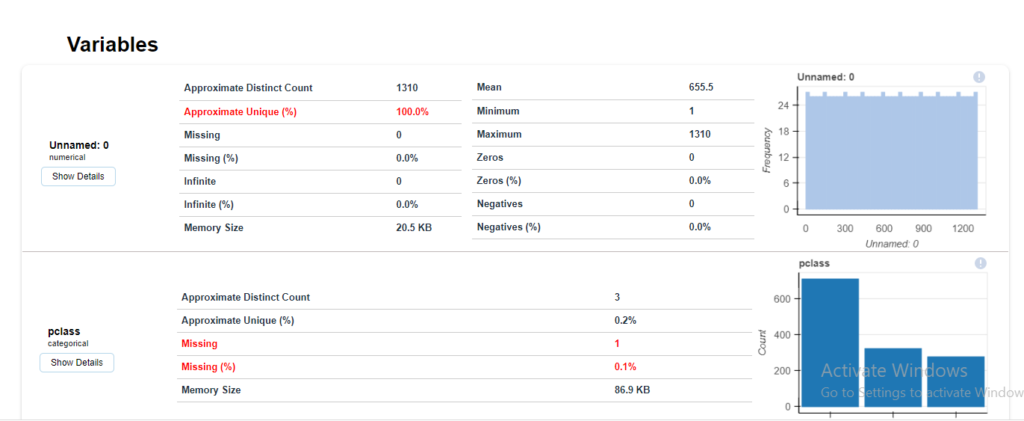
To further analyse and understand what necessary steps would be needed to take on the loaded dataset, we can generate an instant report of the dataset in one go, which will provide us with all the necessary information and metrics to analyse where the focus during the data preparation stage must be, particular columns if any.

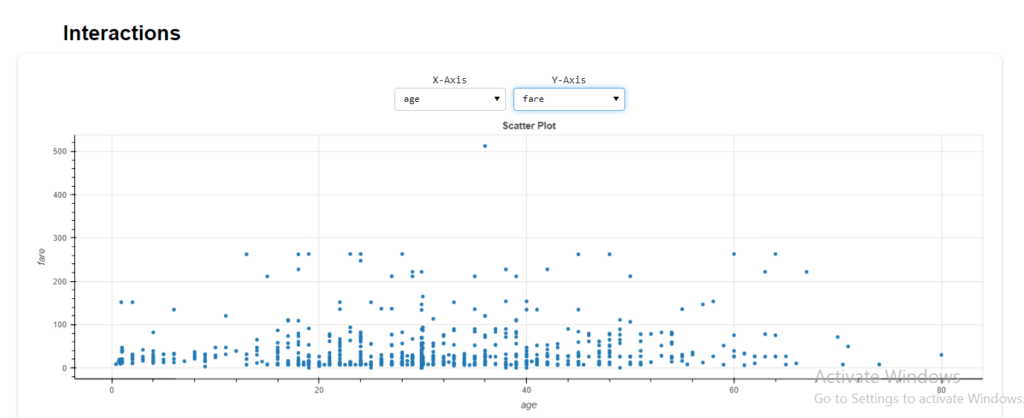
#creating a full report

create\_report(train\_df)



The detailed statistics for each column is generated with options to perform interactions between columns, check correlation or plots of missing values.





## End Notes

In this article, we understood the importance of Data Preparation in Big Data Analytics and the necessary steps required to do so. We also explored a library known as DataPrep, and tested its different functionalities that might help during the Data Preparation and EDA phase. Although there is still a lot more the DataPrep library can do, I would recommend encouraging the reader to explore further and understand the library’s immense power. The following implementations above can be found as Colab notebooks in two separate notebooks. You can access them using the links here:[Notebook 1](https://colab.research.google.com/drive/1J0an4CDLri_NRFkliYIen26hW6banu9u?usp=sharing) – [Titanic Dataset Notebook 2](https://colab.research.google.com/drive/1SDz-4e7Rf1jaQAwYBwUW8KZA5r6I1duo?usp=sharing).

Happy Learning!

## References

* [PyPi DataPrep](https://pypi.org/project/dataprep/)
* [Official DataPrep Website](https://docs.dataprep.ai/user_guide/user_guide.html)

What Do You Think?

## Subscribe to our Newsletter

Get the latest updates and relevant offers by sharing your email.

[**Join our Telegram Group. Be**](https://t.me/joinchat/TrPapV7gnN2Oz1AZ)