BigQuery in JupyterLab on Vertex AI 2.5

## Overview

The purpose of this lab is to show learners how to instantiate a Jupyter notebook running on Google Cloud Platform's Vertex AI service. To aid in the demonstration, a dataset with various flight departure and arrival times will be leveraged.

### **Objectives**

In this lab, you learn to perform the following tasks:

* Instantiate a Jupyter notebook on Vertex AI.
* Execute a BigQuery query from within a Jupyter notebook and process the output using Pandas.

### **Open BigQuery Console**

1. In the Google Cloud Console, select **Navigation menu** > **BigQuery**.

The **Welcome to BigQuery in the Cloud Console** message box opens. This message box provides a link to the quickstart guide and lists UI updates.

1. Click **Done**.

## Start a JupyterLab Notebook Instance

1. Click on the **Navigation Menu**.
2. Navigate to **Artificial Intelligence**, **Vertex AI**, then to **Workbench**.
3. You'll be redirected to **User-Managed Notebooks** tab on the main page for **Notebooks** on Vertex AI.

When the tab loads if you notice a link entitled **Enable Notebooks API**, click that link to allow the background Notebooks API to be upgraded. The upgrade will occur promptly.

Click on the **New Notebook** icon on the top of the page.

1. In the menu that pops down, select the **Python 3** option.
2. A screen entitled **New notebook** will be shown. Leave the default options and click on **Create**.
3. After a few minutes, the Vertex AI console will have your instance name followed by **Open Jupyterlab**. Click **Open Jupyterlab**.
4. A new tab will open in your browser with the JupyterLab environment. Select **Python 3** under **Notebook**.

Your notebook is now set up.

Click Check my progress to verify the objective.

Start a JupyterLab Notebook Instance.

Check my progress

## Execute a BigQuery query

1. Execute the following Python install command by hitting **Shift + Enter** in the first cell of the notebook to install the google-cloud-bigquery library at version 1.25.0.

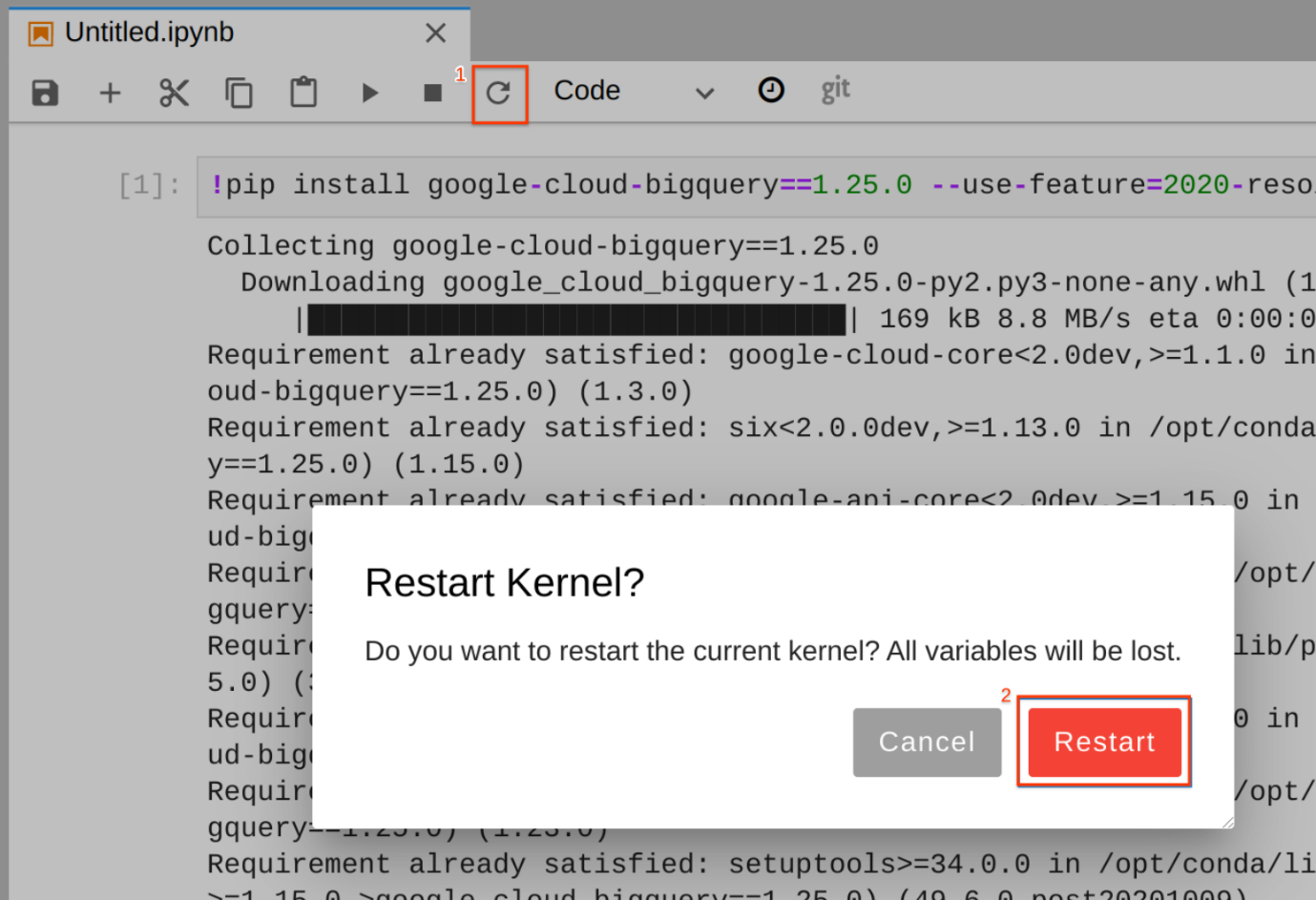
!pip install google-cloud-bigquery==1.25.0 --use-feature=2020-resolver

Copied!

content\_copy

**Note:** You may safely ignore the following notifications: **WARNING: --use-feature=2020-resolver...** and **ERROR: pip's dependency resolver...**.

Restart the kernel by clicking **Restart kernel** icon > **Restart**.



1. Enter the following query in the second cell of the notebook.

%%bigquery df

SELECT

departure\_delay,

COUNT(1) AS num\_flights,

APPROX\_QUANTILES(arrival\_delay, 10) AS arrival\_delay\_deciles

FROM

`bigquery-samples.airline\_ontime\_data.flights`

GROUP BY

departure\_delay

HAVING

num\_flights > 100

ORDER BY

departure\_delay ASC

Copied!

content\_copy

The command makes use of the magic function %%bigquery. Magic functions in notebooks provide an alias for a system command. In this case, %%bigquery runs the query in the cell in BigQuery and stores the output in a Pandas DataFrame object named df.

1. Run the cell by hitting **Shift + Enter**, when the cursor is in the cell. Alternatively, if you navigate to the **Run** tab you can click on **Run Selected Cells**. Note the keyboard shortcut for this action in case it is not Shift + Enter. There should be no output when executing the command.

Click Check my progress to verify the objective.

Execute a BigQuery query

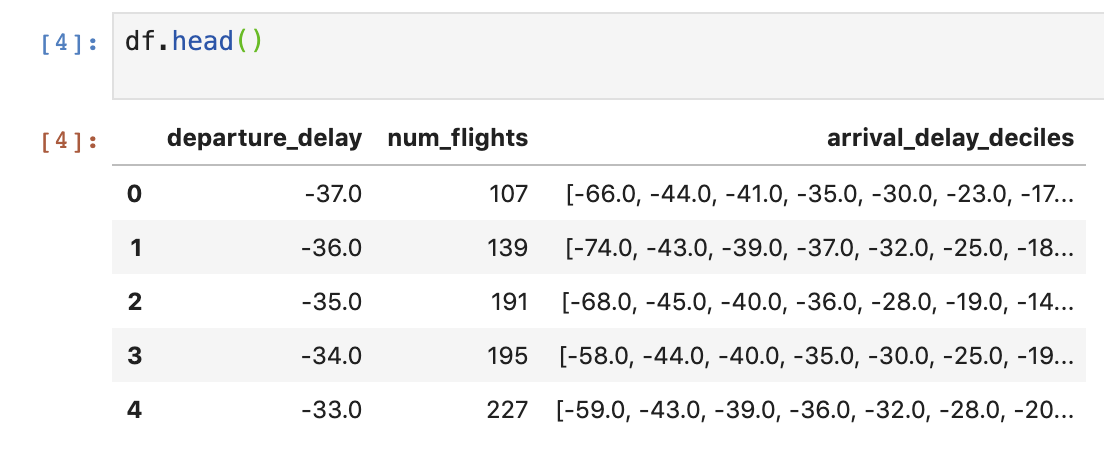
Check my progress

1. View the first five rows of the query's output by executing the following code in a new cell:

df.head()

Copied!

content\_copy



## Make a Plot with Pandas

We're going to use the Pandas DataFrame containing our query output to build a plot that depicts how arrival delays correspond to departure delays. Before continuing, if you are unfamiliar with Pandas the [Ten Minute Getting Started Guide](https://pandas.pydata.org/pandas-docs/stable/user_guide/10min.html) is recommended reading.

1. To get a DataFrame containing the data we need we first have to wrangle the raw query output. Enter the following code in a new cell to convert the list of arrival\_delay\_deciles into a Pandas Series object. The code also renames the resulting columns.

import pandas as pd

percentiles = df['arrival\_delay\_deciles'].apply(pd.Series)

percentiles.rename(columns = lambda x : '{0}%'.format(x\*10), inplace=True)

percentiles.head()

Copied!

content\_copy

1. Since we want to relate departure delay times to arrival delay times we have to concatenate our percentiles table to the departure\_delay field in our original DataFrame. Execute the following code in a new cell:

df = pd.concat([df['departure\_delay'], percentiles], axis=1)

df.head()

Copied!

content\_copy

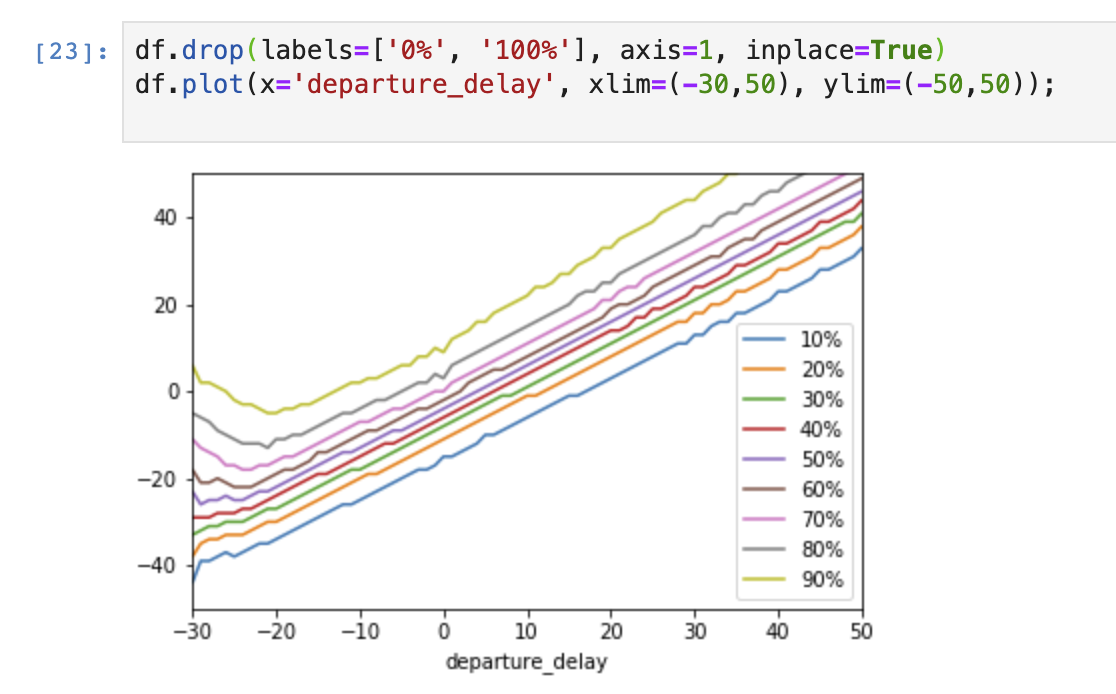
1. Before plotting the contents of our DataFrame, we'll want to drop extreme values stored in the 0% and 100% fields. Execute the following code in a new cell:

df.drop(labels=['0%', '100%'], axis=1, inplace=True)

df.plot(x='departure\_delay', xlim=(-30,50), ylim=(-50,50));

Copied!

content\_copy



## End your lab