Reference guide: Import datasets with Python

In your career as a data professional, you will come across various datasets that have different file types or are stored in various databases. As you've learned previously, it is critical for you to know what these data types are and how to import data using Python. Below you will find examples of importing both databases through connections and data files into Python.

Although you will use the Coursera platform for Python coding, you will need to know how to work with and import CSV files if you'd like to download and open them outside of Coursera.

How to import a dataset from a CSV file

For this example, find a CSV file on your computer. If you don't have one, you can use a dataset of unicorn companies (companies that reached a valuation of $1 billion USD) from this course's Resources Opens in a new tab .

There are several different ways to import a CSV file into Python, but we will only review some of the more common ways. Start by using a `with` statement and `open()` function. Pass the **file name (or file path)** of the CSV file to the `open()` function along with an argument for the `mode` parameter of the function.

```
with open('file_path/file_name, mode=)
```

The mode is telling the Python library what to do with the file. When defining the **mode**, you use one of the following options:

- 'r' – read
- 'w' – write
- 'a' – append
- '+' – create new file

Typically, you'll be defining the mode inside the `with open()` argument field as `'r'` because you want Python to open and read the CSV file.

Next, we'll add `as file` to the end, which is assigning the result to a variable name. In this case, we'll name it `data`.

```
with open('example_filepath/file', mode='r') as file:
      data = file.read()
```

Importing a CSV file using pandas

Instead of using Python's standard library to read a file, you can use pandas to import the CSV file into a dataframe. First, of course, you'll want to import the pandas library into your Python notebook.

```
import pandas as pd
```

Next, you'll use the `read_csv()` function to load the data into a dataframe. The file path then goes in the argument field.

```
df = pd.read_csv('file_path/file_name')
```

Note: *You can also use this same syntax for importing a CSV file that is stored on the internet. In the place of the filename, you would simply copy and paste the url.*

How to import data by connecting to a database

There are a number of database solutions that you can connect to with Python, such as BigQuery, MySQL, SQLite, and Oracle. Databases are a convenient way for companies and organizations to store huge amounts of data.

If the dataset is small enough, it can be downloaded and manipulated locally on your computer. However, often the datasets kept in databases are too large to access in their entirety on a personal computer. In this case you have a number of different options, most of which involve querying the database with SQL to obtain specific tables of interest. In other words, you extract select parts—usually specified rows and/or columns—from the whole dataset. The manner in

which querying is done can vary with respect to systems, platforms, and interfaces. Because of this variability, this reference guide will only present a couple of different ways to query databases. Specifically, it will explore BigQuery, Google's data warehouse that provides a wide range of tools and services to facilitate analysis.

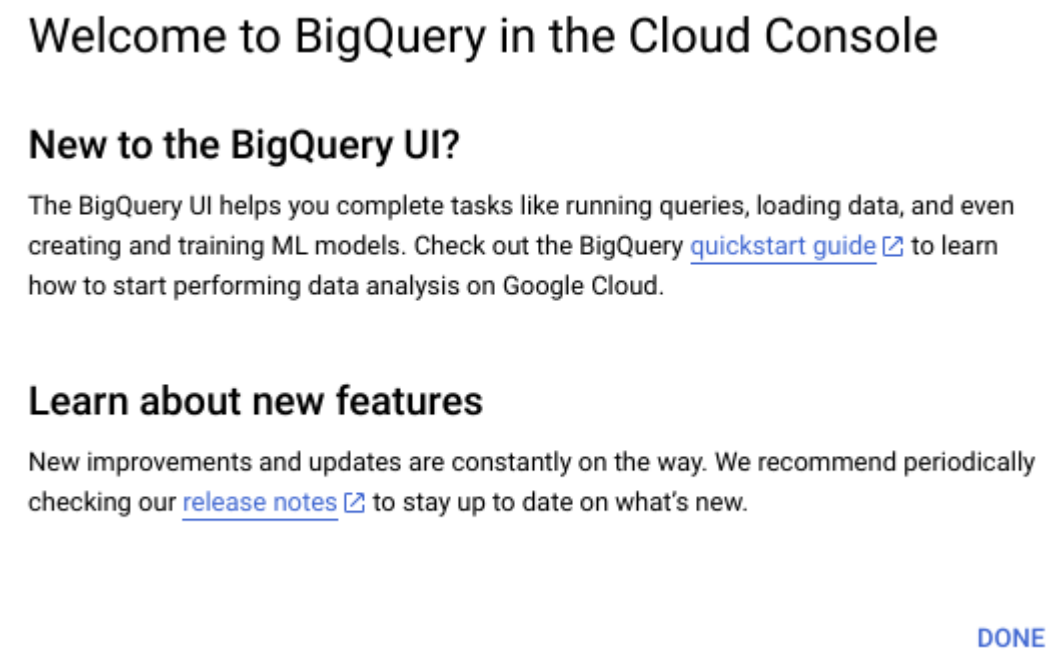
Downloading data from BigQuery

Step 1: Access BigQuery

BigQuery allows you to upload data for storage, and it also has a number of publicly available datasets to explore. You can access these public datasets for free using BigQuery Sandbox, which requires a free Google account. Sandbox gives you 10 GB of active storage and 1 TB of processed query data each month for free.

Step 2: Perform a query

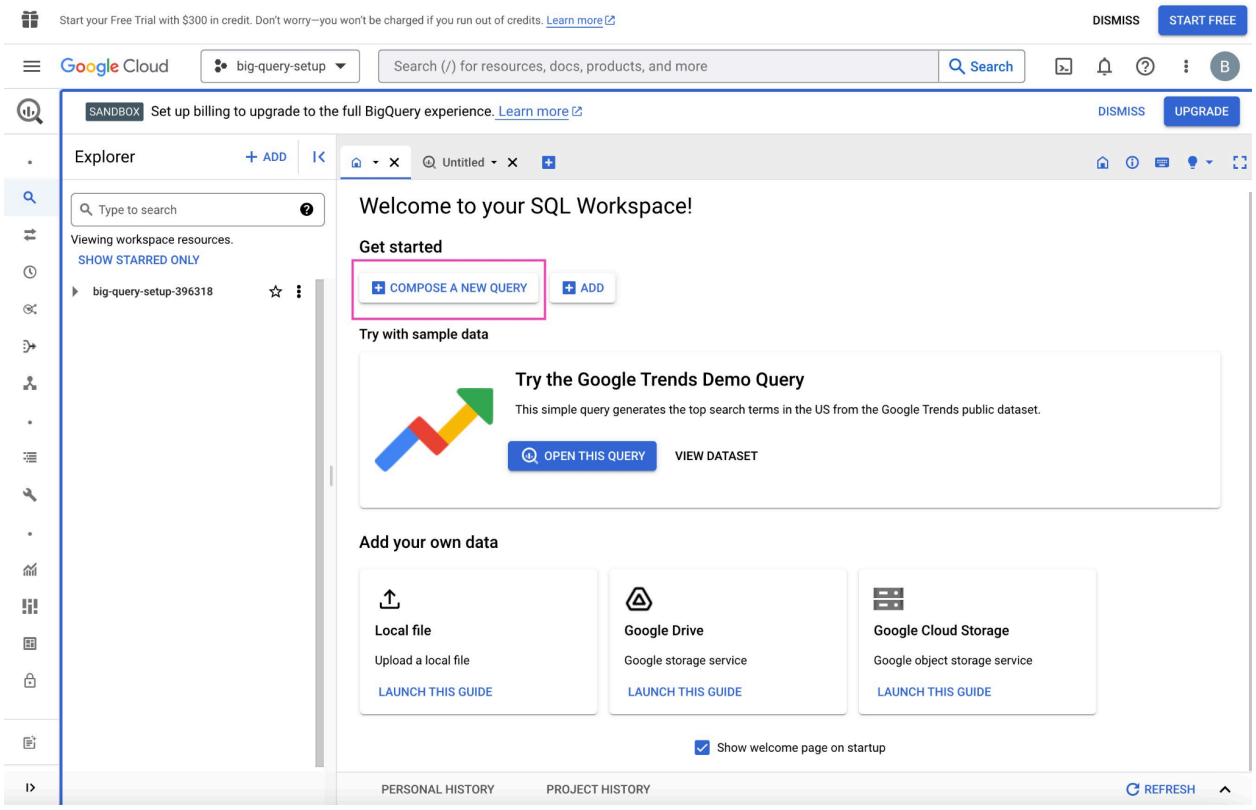
Once you have authenticated your account and created a new project as indicated in the instructions linked in step one, you're ready to query a database. Note that if it is your first time logging in, you may encounter a window asking "New to the BigQueryUI?" with a link to a quickstart guide.



[**Alt-text:** Screenshot of the "New to BigQuery UI?" window]

The quickstart guide will guide you through the same steps as those presented to you here.

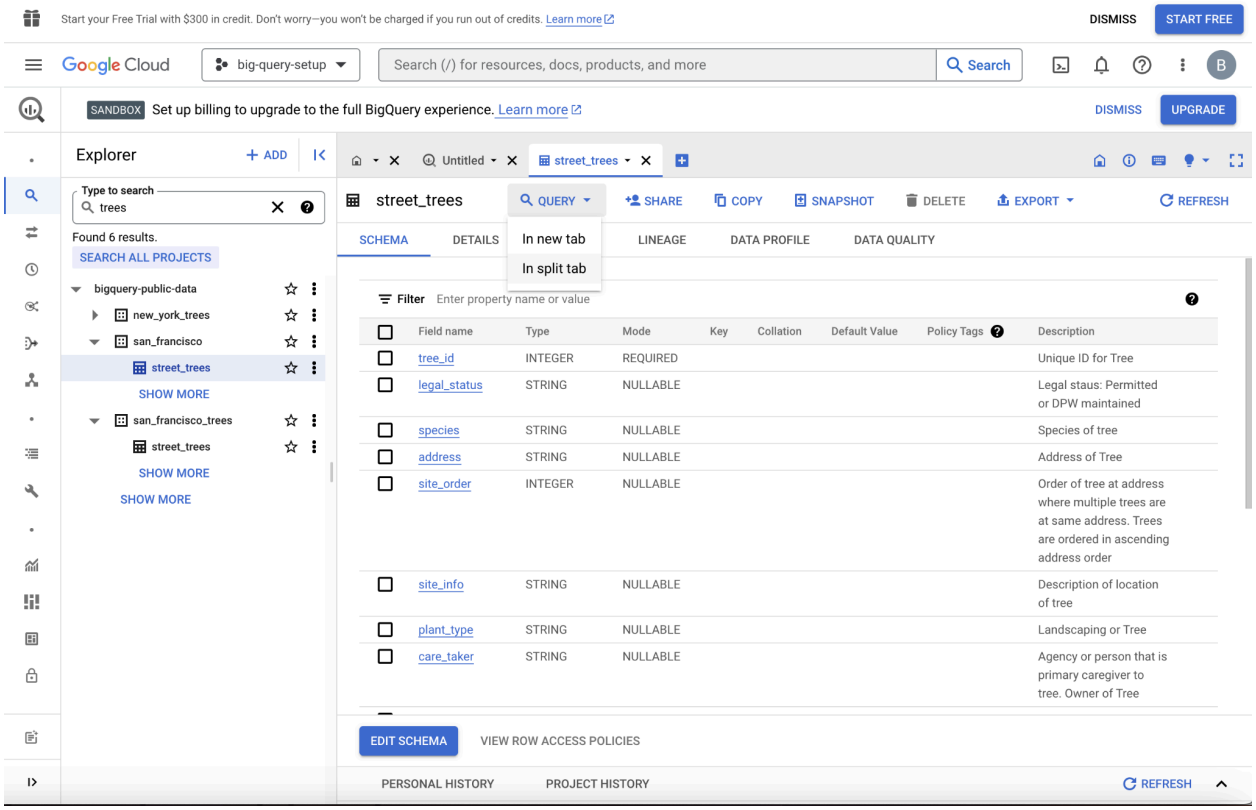
From the “Welcome to your SQL Workspace!” page, click the “Compose a new query” button.



[**Alt-text:** Screenshot of the “Welcome to your SQL Workspace!” page]

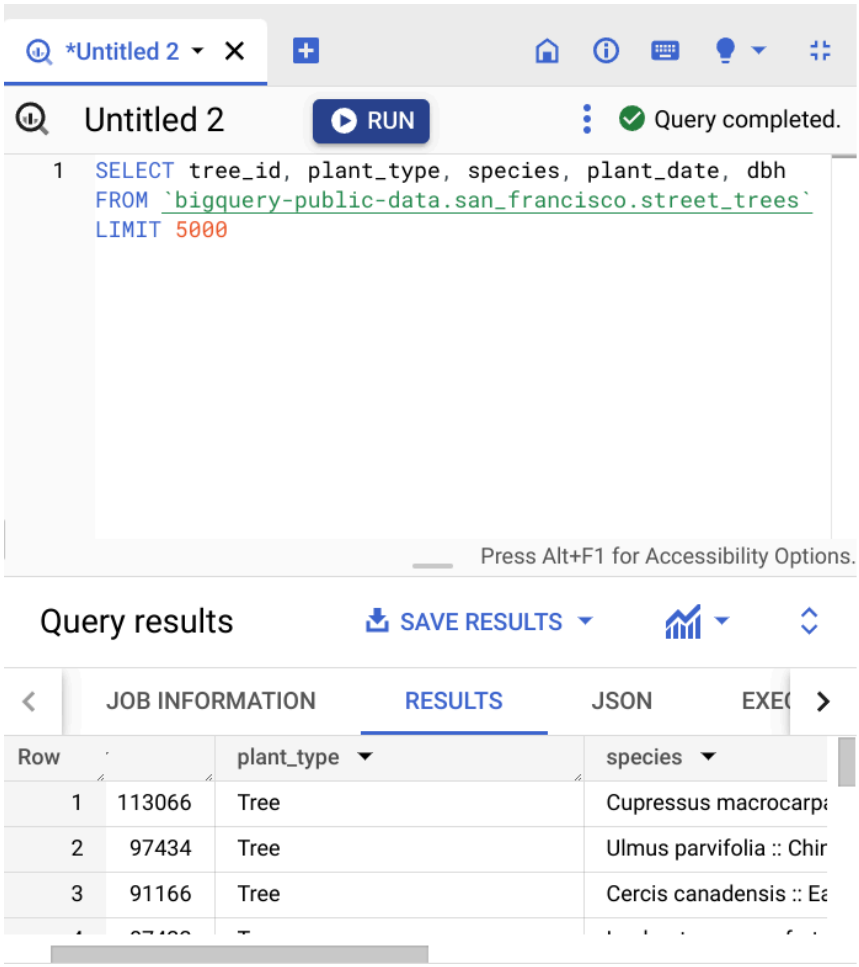
Click into the search bar in the Explorer on the left side of the page. For example, you can search for “trees.” Initially, this will return zero results. However, click “Search all projects” and it will return applicable datasets from the `biquery-public-data` project and premade tables from those datasets.

Click the `street_trees` table in the `san_francisco` dataset. The metadata for this table will appear in a panel to the right. Then, click “Query” from the menu at the top of the metadata panel. You can opt to query in a new tab or in a split-pane of the current window.



[**Alt-text:** Screenshot of the San Francisco street trees query page]

Now, you can query the table using SQL. For example, the query in the following screenshot selects 5,000 rows with columns of `tree_id`, `plant_type`, `species`, `plant_date`, and `dbh` – defined as "depth, height."



[**Alt-text:** Screenshot of the SQL query]

Once you are satisfied with your query, click the “Run” button at the top of the SQL query panel. The results will display below, and there is a button to “Save results,” which allows you to save the resulting table in different locations and formats. From there, you can read the data into your notebook.

Using notebooks within BigQuery

Another way to access data on BigQuery is by using the tools within the BigQuery platform itself. This workflow more closely resembles what data professionals would use when working with very large datasets stored in the cloud. Essentially, you set up a virtual machine on BigQuery. A virtual machine is a computer that has its own CPU, memory, software, etc., just like any other computer, only it does not have its own dedicated hardware; they most often exist as a

partition on a server. You can work in a Jupyter notebook on the virtual machine on the BigQuery platform, from which you can query and pull in data directly.

This process requires you to set up a payment method. However, new users get a $300 credit, and a ML instance is only a few cents per minute, so you'll get approximately 2,000 hours of free usage before incurring any charges. There are a lot of great tutorials for setting this up. For instance, if you search for "How to use Jupyter notebook in Google Cloud AI," you'll find a number of useful videos and blogs on the topic.

Using notebooks outside of BigQuery

It's also possible to query data on BigQuery from notebooks that are not on the BigQuery platform. However, the details of this process are dependent on a number of factors, including the platform that is hosting the notebook, the operating environment, and the specific location of the data being accessed. Therefore, we will not go into depth on this method. Feel free to explore this on your own, though. You'll find many helpful online resources that are just a search away.

Key takeaways

There are lots of different kinds of data, which means there are numerous ways to import data. Learning several methods to import data, whether it be from a data file or a database, will build a solid foundation for your career as a data professional.

Resources for more information

To learn more about importing data into Python, you can refer to the following links:

- An overview of importing data in Python
- How to connect to BigQuery from a Colab

Citations:

#	Title	Link
1.	An Overview of importing data in Python	https://towardsdatascience.com/an-overview-of-importing-data-in-python-ac6aa46e0889
2.	Downloading BigQuery data to pandas using the BigQuery Storage API	https://cloud.google.com/bigquery/docs/bigquery-storage-python-pandas

---