

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](#).

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 importing the Python library "CSV." You can do that by opening a notebook and typing the following:

```
import csv
```

Next you can use the `with` statement and `open()` function to define the **file name (or file path)** and then the **mode**:

```
With open("filename / file path", 'mode')
```

So the syntax so far is:

```
import csv  
with open("filename / file path", '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 you'll recall is giving the file a data frame name, in this case, we'll name it `csv_df`.

```
import csv
with open("Downloads:\\eda_missing_data_dataset1.csv",'r') as file:
    csv_df = file.read()
```

Importing a CSV file using pandas

Instead of using the CSV package, you can also use pandas to import a CSV data file. First, of course, you'll want to import pandas in your Python notebook.

```
import pandas as pd
```

Next, you'll use the “`read_csv`” function to load the data into a dataframe. Most data professionals use abbreviations, like “DF.” The file path then goes in the argument field.

```
df = pd.read_csv ('#input file path here')
```

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.

Data from databases can be accessed and analyzed via a query within your code, rather than downloading a file. Data professionals are regularly tasked with analyzing data from databases. Below you will find a step-by-step guide for connecting to datasets in BigQuery, which is a large online database.

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.

Welcome to BigQuery in the Cloud Console

New to the BigQuery UI?

The BigQuery UI helps you complete tasks like running queries, loading data, and even creating and training ML models. Check out the BigQuery [quickstart guide](#) to learn how to start performing data analysis on Google Cloud.

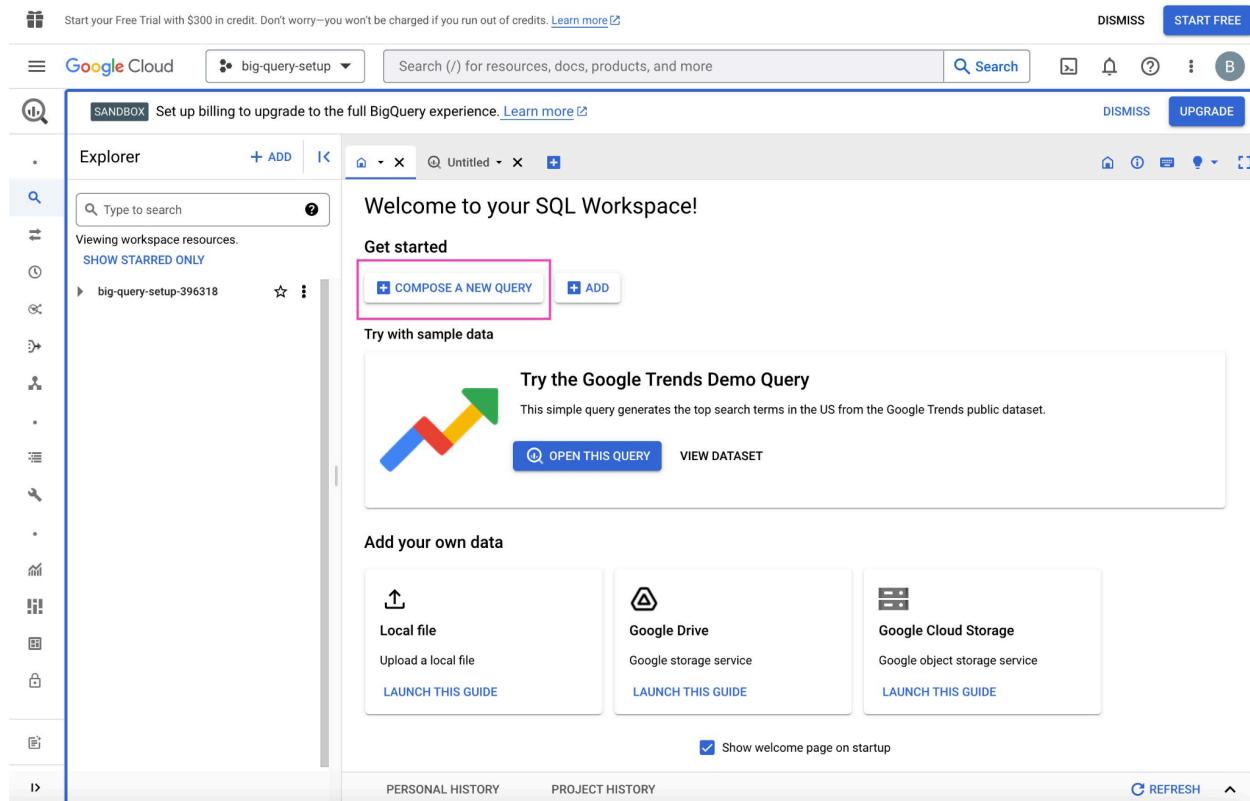
Learn about new features

New improvements and updates are constantly on the way. We recommend periodically checking our [release notes](#) to stay up to date on what's new.

[DONE](#)

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.



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 `bigrquery-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.

The screenshot shows the Google Cloud BigQuery interface. In the top navigation bar, there is a message about a free trial and a 'START FREE' button. Below the navigation bar, the search bar contains 'big-query-setup'. The main area is titled 'street_trees' and shows the table schema. The schema includes columns for tree_id (INTEGER, REQUIRED), legal_status (STRING, NULLABLE), species (STRING, NULLABLE), address (STRING, NULLABLE), site_order (INTEGER, NULLABLE), site_info (STRING, NULLABLE), plant_type (STRING, NULLABLE), and care_taker (STRING, NULLABLE). There are tabs for SCHEMA, DETAILS, LINEAGE, DATA PROFILE, and DATA QUALITY. A modal window is open over the schema table, with the 'In new tab' option selected from a dropdown menu.

Field name	Type	Mode	Key	Collation	Default Value	Policy Tags	Description
tree_id	INTEGER	REQUIRED					Unique ID for Tree
legal_status	STRING	NULLABLE					Legal status: Permitted or DPW maintained
species	STRING	NULLABLE					Species of tree
address	STRING	NULLABLE					Address of tree
site_order	INTEGER	NULLABLE					Order of tree at address where multiple trees are at same address. Trees are ordered in ascending address order
site_info	STRING	NULLABLE					Description of location of tree
plant_type	STRING	NULLABLE					Landscaping or Tree
care_taker	STRING	NULLABLE					Agency or person that is primary caregiver to tree. Owner of Tree

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.”

The screenshot shows the BigQuery web interface. At the top, there's a toolbar with icons for file operations, navigation, and help. Below the toolbar, the title bar says "Untitled 2" and has a "RUN" button. A message "Query completed." with a checkmark is displayed. The main area contains a single query:

```
1 SELECT tree_id, plant_type, species, plant_date, dbh  
FROM `bigquery-public-data.san_francisco.street_trees`  
LIMIT 5000
```

Below the query, a message says "Press Alt+F1 for Accessibility Options." The results section is titled "Query results" and includes a "SAVE RESULTS" button and a chart icon. The results table has columns: Row, plant_type, and species. The data rows are:| Row | plant_type | species |
| --- | --- | --- |
| 1 | Tree | Cupressus macrocarpa |
| 2 | Tree | Ulmus parvifolia :: Chir |
| 3 | Tree | Cercis canadensis :: Ea |

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.

Using Notebook:

Step 1: Access BigQuery

First off when you access BigQuery, you'll start a new Google Cloud project. Make sure the billing is enabled. You'll also need to set up your authentication in BigQuery which is described in Google Cloud's resource, [Getting started with authentication](#).

Step 2: Open your Python notebook

Next you should open a Python notebook. You can use IPython magic commands to connect to BigQuery, which makes the import quite simple. Begin by opening a Jupyter notebook and loading the magic commands for big query:

```
%load_ext google.cloud.bigquery
```

From there, you can input the following magic command “%%bigquery” along with the name of any set of data in the database.

```
%%bigquery - country_names_area df
```

Step 3: Select your data

You can then use SQL commands like `SELECT` and `FROM` to select the data you want to use.

```
SELECT * FROM `country_names_area`
```

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.