BigQuery Data Ingestion

Overview

Information access uses multiple formats, and BigQuery makes working with multiple data sources simple. In this lab you will get started with sports data science by importing external sports data sources into BigQuery tables. This will give you the basis for building more sophisticated analytics in subsequent labs.

The data used in this lab originates from the following sources:

- Pappalardo et al., (2019) **A public data set of spatio-temporal match events in soccer competitions**, Nature Scientific Data 6:236, https://www.nature.com/articles/s41597-019-0247-7
- Pappalardo et al. (2019) **PlayerRank: Data-driven Performance Evaluation and Player Ranking in Soccer via a Machine Learning Approach**. ACM Transactions on Intelligent Systems and Technologies (TIST) 10, 5, Article 59 (September 2019), 27 pages. DOI: https://doi.org/10.1145/3343172

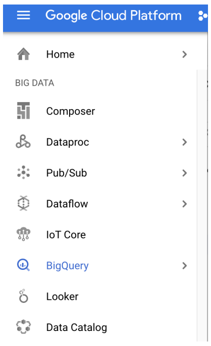
In this lab, you will learn how to:

- Upload files from Google Cloud Storage (GCS) into BigQuery tables using the Cloud Console.
- Use the Cloud Console to access information derived from BigQuery tables.
- Understand how to write queries on the uploaded tables.

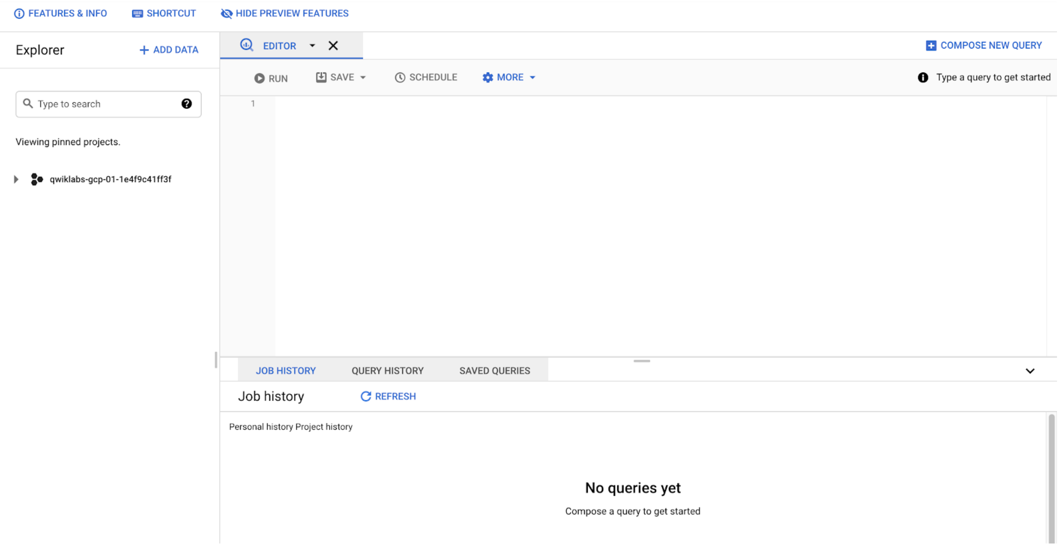
Open BigQuery

The BigQuery console provides an interface to query tables, including public datasets offered by BigQuery.

1. In the Cloud Console, from the **Navigation menu** select **BigQuery**:



2. The **Welcome to BigQuery in the Cloud Console** message box opens. This message box provides a link to the quickstart guide and the release notes.
3. Click **Done**.
4. The BigQuery console opens.



No queries yet

Compose a query to get started

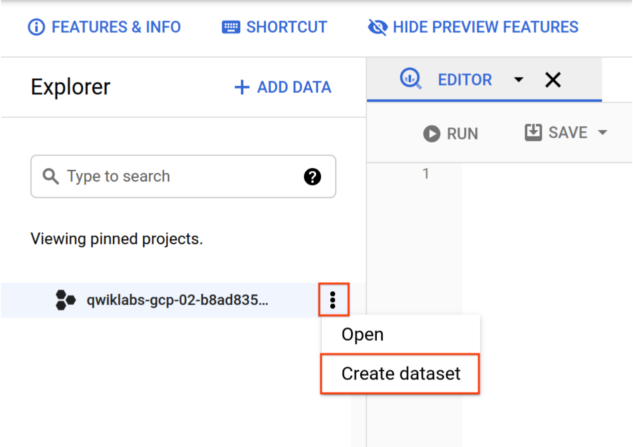
In this section the BigQuery interface was used to access the console. The console provides a convenient way to add information to a dataset. BigQuery uses tables to represent data in a structured way.

In the next section learn more about BigQuery and how to create custom tables.

Create custom tables

In this section, you will create a dataset. The dataset is used to add data to the project. Datasets utilize **tables** and **views** to help control access to data within a project.

1. In the BigQuery console, observe the **Explorer** section.
2. Click on the **View actions** icon next to your project ID and select **Create dataset**.



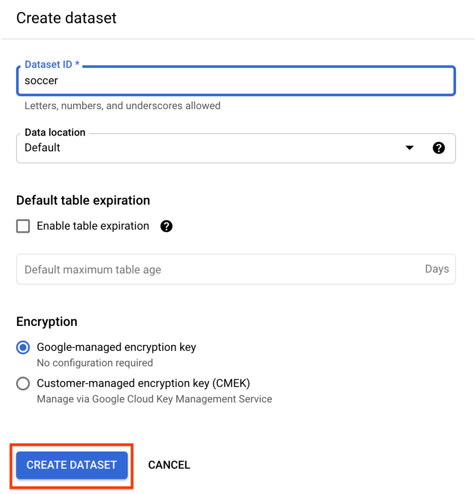
Viewing pinned projects.

A dataset is contained within a specific project. Datasets are top-level containers that are used to organize and control access to your tables and views. A table or view must belong to a dataset, so you need to create at least one dataset before loading data into BigQuery. Reference: BigQuery datasets introduction

3. On the **Create dataset** page fill in the following:

Field	Value
Dataset ID	soccer
Data location	United States (US)
Default table expiration	*Default*

4. The BigQuery **Create dataset** screen will display information similar to below:



Create dataset

Dataset ID * soccer

Letters, numbers, and underscores allowed

Data location Default

Default table expiration

□ Enable table expiration

Default maximum table age Days

Encryption

● Google-managed encryption key
No configuration required

○ Customer-managed encryption key (CMEK)
Manage via Google Cloud Key Management Service

Currently, the public datasets are stored in the US multi-region data locations. For simplicity, place your dataset in the same location.

5. Click **Create dataset** at the bottom of the panel.

In this section a new dataset was created using BigQuery. During this process, BigQuery needs to know where to store the information to be created. It also provides the option to include customer managed encryption, if required. In the next section learn how to populate the created dataset with JavaScript Object Notation (JSON) a common data format.

Load JSON Data

Now you will load the tables created previously with soccer data into the dataset. BigQuery provides support for a number of import formats. In this lab use JSON with the dataset created in the previous section.

1. Create a table by clicking on the **View actions** icon next to your `soccer` dataset in the Explorer section.
2. Select **Open**, then click **Create table**.

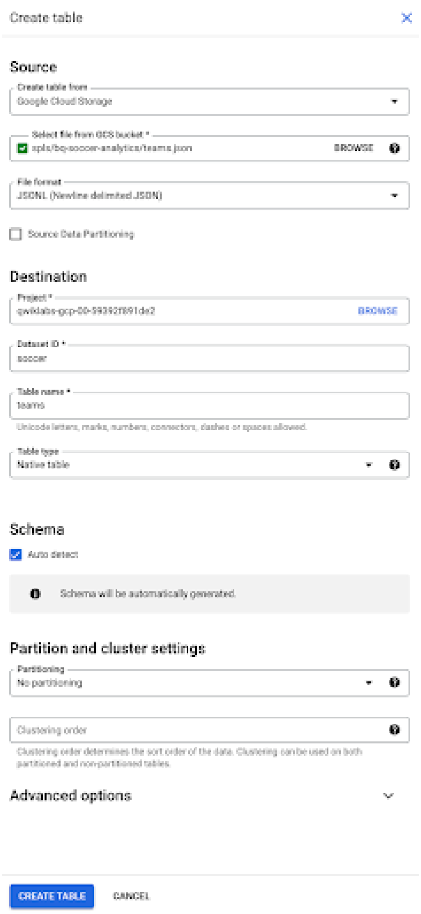
In the following section use the default values for all settings unless otherwise indicated. The data is stored in a public Google Cloud Storage (GCS) bucket.

3. On the **Create table** page add the following information:

Source	Google Cloud Storage
Select file from GCS bucket	spls/bq-soccer-analytics/competitions.json
File format	JSONL (Newline delimited JSON)
Table name	competitions
Schema	Check the box marked Schema `Auto detect`

Note: *When using Cloud Storage buckets with BigQuery, it does not require the prefix of* `gs://` *to be applied.*

4. The BigQuery **Create table** screen will display information similar to below



Create table

Source

Create table from: Google Cloud Storage

Select file from GCS bucket *: ☑ spls/bq-soccer-analytics/teams.json

File format: JSONL (Newline delimited JSON)

□ Source Data Partitioning

Destination

Project *: qwiklabs-gcp-00-59392f891de2

Dataset ID *: soccer

Table name *: teams

Unicode letters, marks, numbers, connectors, dashes or spaces allowed.

Table type: Native table

Schema

☑ Auto detect

Schema will be automatically generated.

Partition and cluster settings

Partitioning: No partitioning

Clustering order

Clustering order determines the sort order of the data. Clustering can be used on both partitioned and non-partitioned tables.

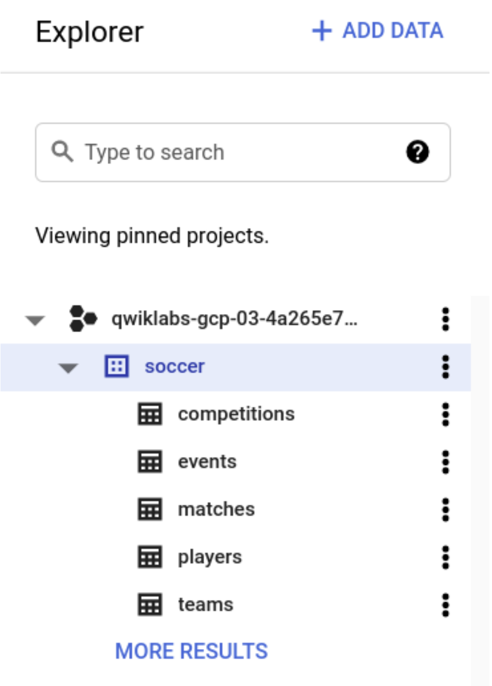
Advanced options

5. Click **Create table**.
6. Wait for BigQuery to create the table and load the data.
7. A pop up notification message saying **"competitions" created** is displayed.
8. **The table** will show up after the data is loaded.
9. Repeat the steps above for the other JSON data to be ingested.

GCS bucket file	**Table name**
spls/bq-soccer-analytics/matches.json	matches
spls/bq-soccer-analytics/teams.json	teams
spls/bq-soccer-analytics/players.json	players
spls/bq-soccer-analytics/events.json	events

Use the exact Cloud Storage bucket files and table names shown.

10. Once the tables are created the display will be similar to below:



In this section new tables were created using BigQuery. During this process, BigQuery used Cloud Storage as the source for the JSON files. Cloud Storage provides a good intermediate storage option for object files. In the next section learn how to populate the created dataset with a comma-separated values (CSV) file, that is another common data format.

Load CSV data

In this section, load another table of soccer data into the dataset. The load process will this time be sourced from a comma-separated values (CSV) file stored in Cloud Storage.

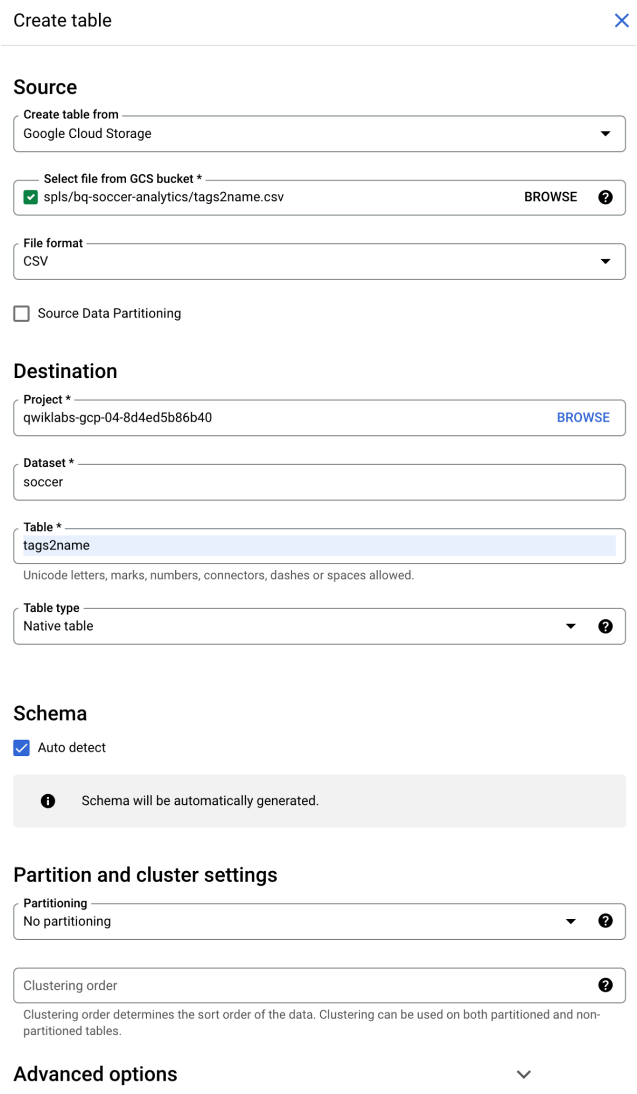
1. Create a table by clicking on the **View actions** icon next to your `soccer` dataset in the Explorer section. Select **Open**, then click **Create table**.

Use the default values for all settings unless otherwise indicated.

2. On the **Create table** page add the following information:

Source	Google Cloud Storage
Select file from GCS bucket	spls/bq-soccer-analytics/tags2name.csv
File format	CSV
Table name	tags2name
Schema	Check the box marked Auto detect

3. The BigQuery **Create table** screen will display information similar to below:



Create table

Source

Create table from: Google Cloud Storage

Select file from GCS bucket *: ☑ spls/bq-soccer-analytics/tags2name.csv

File format: CSV

□ Source Data Partitioning

Destination

Project *: qwiklabs-gcp-04-8d4ed5b86b40

Dataset *: soccer

Table *: tags2name

Unicode letters, marks, numbers, connectors, dashes or spaces allowed.

Table type: Native table

Schema

☑ Auto detect

Schema will be automatically generated.

Partition and cluster settings

Partitioning: No partitioning

Clustering order

Clustering order determines the sort order of the data. Clustering can be used on both partitioned and non-partitioned tables.

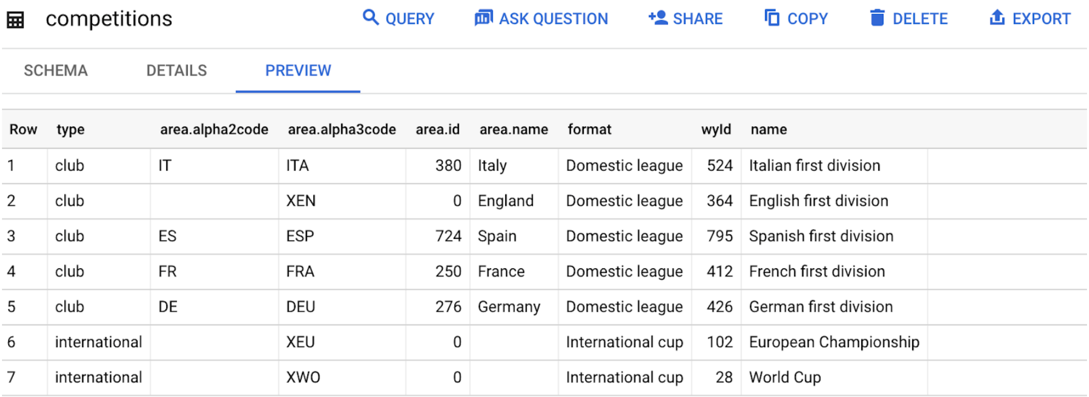
Advanced options

4. Click **Create table** (at the bottom of the window).
5. Wait for BigQuery to create the table and load the data.
6. A pop up message will appear saying "tags2name" created.
7. **The table** will show up after the data is loaded.

In this section a new table was created using BigQuery. During this process, BigQuery used Cloud Storage as the source for the CSV file. Cloud Storage provides a good intermediate storage option for object files.

review tables

1. In the left pane, select **soccer** > **competitions** in the navigation panel.
2. In the Details panel, click the **Preview** tab.



Row	type	area.alpha2code	area.alpha3code	area.id	area.name	format	wyId	name
1	club	IT	ITA	380	Italy	Domestic league	524	Italian first division
2	club		XEN	0	England	Domestic league	364	English first division
3	club	ES	ESP	724	Spain	Domestic league	795	Spanish first division
4	club	FR	FRA	250	France	Domestic league	412	French first division
5	club	DE	DEU	276	Germany	Domestic league	426	German first division
6	international		XEU	0		International cup	102	European Championship
7	international		XWO	0		International cup	28	World Cup

3. Click through the other uploaded tables from the navigation panel.
4. Check the **Schema**, **Details**, and **Preview** tabs to learn more about the data in each table.

BigQuery provides a convenient way to store data previously held in a variety of formats. To learn more about data ingestion techniques for BigQuery read "Choosing a data ingestion method".

In the next couple of sections learn how to query the datasets created in BigQuery.

Query Player data

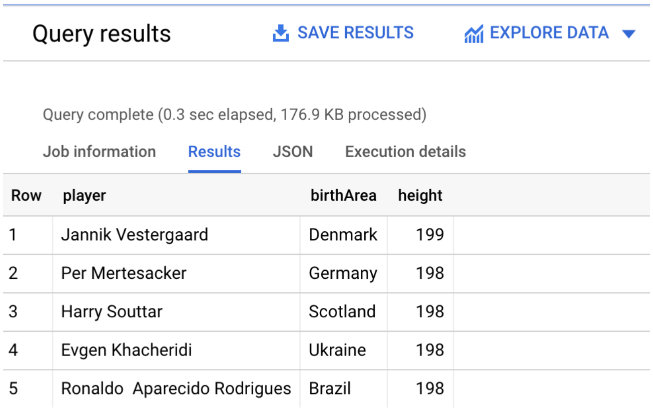
Now that you've loaded data into your tables, you can run queries against it. Next, create a query that retrieves the top 10 tallest defenders (for whom height is available) in the players table.

1. In the Query editor, click **Compose new query**.
2. Copy and paste the following query into the query Editor.

```
SELECT
  (firstName || ' ' || lastName) AS player,
  birthArea.name AS birthArea,
  height
FROM
  `soccer.players`
WHERE
  role.name = 'Defender'
ORDER BY
  height DESC
LIMIT 5
```

In the above query, use BigQuery to retrieve information relating to soccer players. Specifically query for a specific player role to understand the general characteristics of a defender.

3. Click **Run**. The results are displayed below the query window.



Query complete (0.3 sec elapsed, 176.9 KB processed)

Row	player	birthArea	height
1	Jannik Vestergaard	Denmark	199
2	Per Mertesacker	Germany	198
3	Harry Souttar	Scotland	198
4	Evgen Khacheridi	Ukraine	198
5	Ronaldo Aparecido Rodrigues	Brazil	198

Understanding how to perform queries in BigQuery is essential. Running queries in BigQuery provides a simple interface to extract powerful data insights.

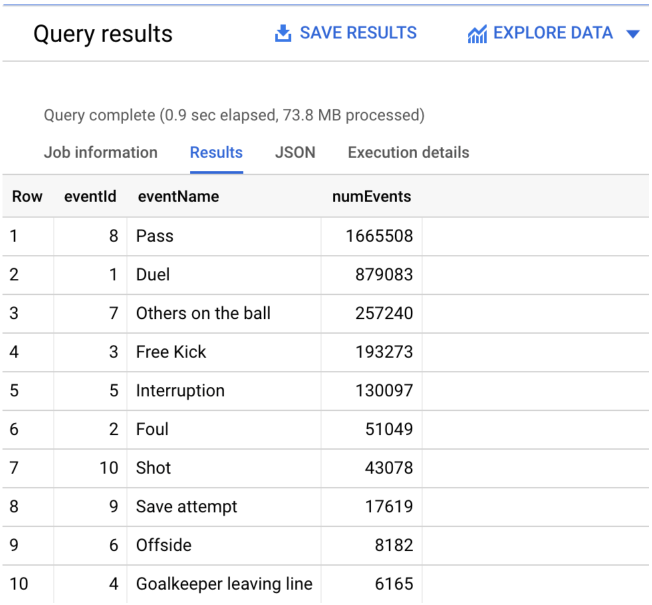
Query Events data

Create a query to retrieve counts of all event types that are found in the **events** table.

1. Copy and paste the following query into the query Editor.

```
SELECT
  eventId,
  eventName,
  COUNT(id) AS numEvents
FROM
  `soccer.events`
GROUP BY
  eventId, eventName
ORDER BY
  numEvents DESC
```

2. Click **Run**. The results are displayed below the query window.



Query complete (0.9 sec elapsed, 73.8 MB processed)

Row	eventId	eventName	numEvents
1	8	Pass	1665508
2	1	Duel	879083
3	7	Others on the ball	257240
4	3	Free Kick	193273
5	5	Interruption	130097
6	2	Foul	51049
7	10	Shot	43078
8	9	Save attempt	17619
9	6	Offside	8182
10	4	Goalkeeper leaving line	6165

In the above query, use BigQuery to retrieve information relating to events occurring within a soccer match. Specifically query for the frequency of events like passes and shots.