

Introduction to SQL for BigQuery and Cloud SQL

Task 1. The basics of SQL

Databases and tables

As mentioned earlier, SQL allows you to get information from "structured datasets". Structured datasets have clear rules and formatting and often times are organized into tables, or data that's formatted in rows and columns.

An example of *unstructured data* would be an image file. Unstructured data is inoperable with SQL and cannot be stored in BigQuery datasets or tables (at least natively.) To work with image data (for instance), you would use a service like [Cloud Vision](#), perhaps through its [API](#) directly.

The following is an example of a structured dataset—a simple table:

User	Price	Shipped
Sean	\$35	Yes
Rocky	\$50	No

If you've had experience with Google Sheets, then the above should look quite similar. The table has columns for User, Price, and Shipped and two rows that are composed of filled in column values.

A Database is essentially a *collection of one or more tables*. SQL is a structured database management tool, but quite often (and in this lab) you will be running queries on one or a few tables joined together—not on whole databases.

SELECT and FROM

SQL is phonetic by nature and before running a query, it's always helpful to first figure out what question you want to ask your data (unless you're just exploring for fun.)

SQL has predefined *keywords* which you use to translate your question into the pseudo-english SQL syntax so you can get the database engine to return the answer you want.

The most essential keywords are SELECT and FROM:

- Use SELECT to specify what fields you want to pull from your dataset.
- Use FROM to specify what table or tables you want to pull our data from.

	A	B	C
1	USER	PRICE	SHIPPED
2	SEAN	\$35	YES
3	ROCKY	\$50	NO
4	AMANDA	\$20	YES
5	EMMA	\$65	YES
6	ANDRES	\$10	NO
7	CASEY	\$55	YES
8	HANNAH	\$15	NO
9	JOCELYN	\$30	NO

An example may help understanding. Assume that you have the following table example_table, which has columns USER, PRICE, and SHIPPED:

And say that you want to just pull the data that's found in the USER column. You can do this by running the following query that uses SELECT and FROM:

```
SELECT USER FROM example_table
```

Copied!

content_copy

If you executed the above command, you would select all the names from the USER column that are found in example_table.

You can also select multiple columns with the SQL SELECT keyword. Say that you want to pull the data that's found in the USER and SHIPPED columns. To do this, modify the previous query by adding another column value to our SELECT query (making sure it's separated by a comma!):

```
SELECT USER, SHIPPED FROM example_table
```

Copied!

content_copy

Running the above retrieves the USER and the SHIPPED data from memory:

And just like that you've covered two fundamental SQL keywords! Now to make things a bit more interesting.

WHERE

The WHERE keyword is another SQL command that filters tables for specific column values. Say that you want to pull the names from `example_table` whose packages were shipped. You can supplement the query with a WHERE, like the following:

```
SELECT USER FROM example_table WHERE SHIPPED='YES'
```

Copied!

content_copy

Running the above returns all USERS whose packages have been SHIPPED from memory:

Now that you have a baseline understanding of SQL's core keywords, apply what you've learned by running these types of queries in the BigQuery console.

Test your understanding

The following are some multiple choice questions to reinforce your understanding of the concepts covered so far. Answer them to the best of your abilities.

Task 2. Exploring the BigQuery console

The BigQuery paradigm

BigQuery is a fully-managed petabyte-scale data warehouse that runs on the Google Cloud. Data analysts and data scientists can quickly query and filter large datasets, aggregate results, and perform complex operations without having to worry about setting up and managing servers. It comes in the form of a command line tool (pre installed in cloudshell) or a web console—both ready for managing and querying data housed in Google Cloud projects.

In this lab, you use the web console to run SQL queries.

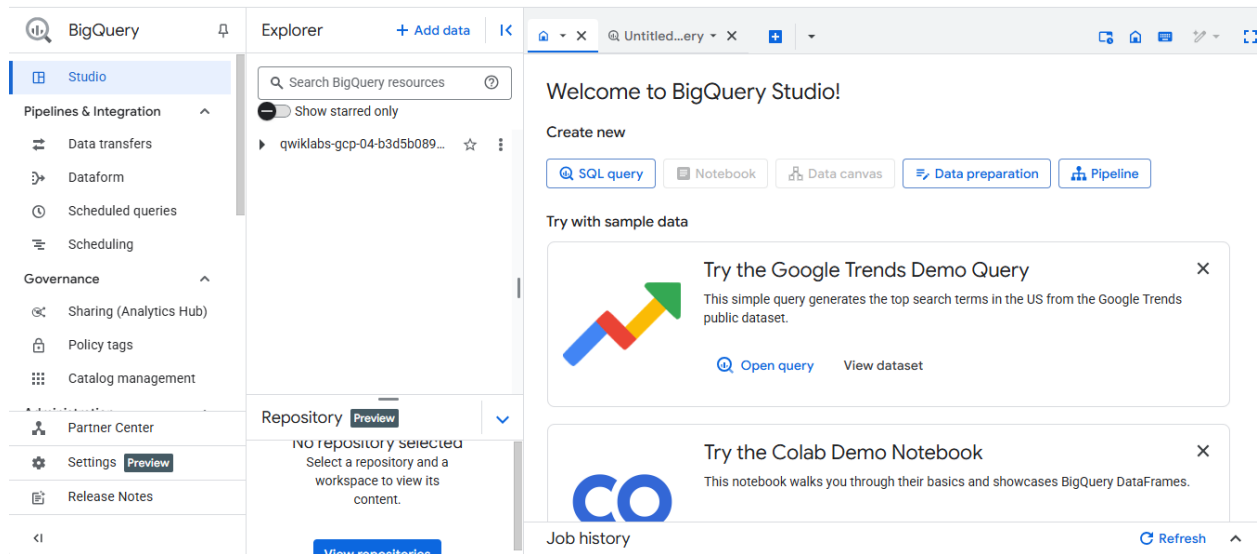
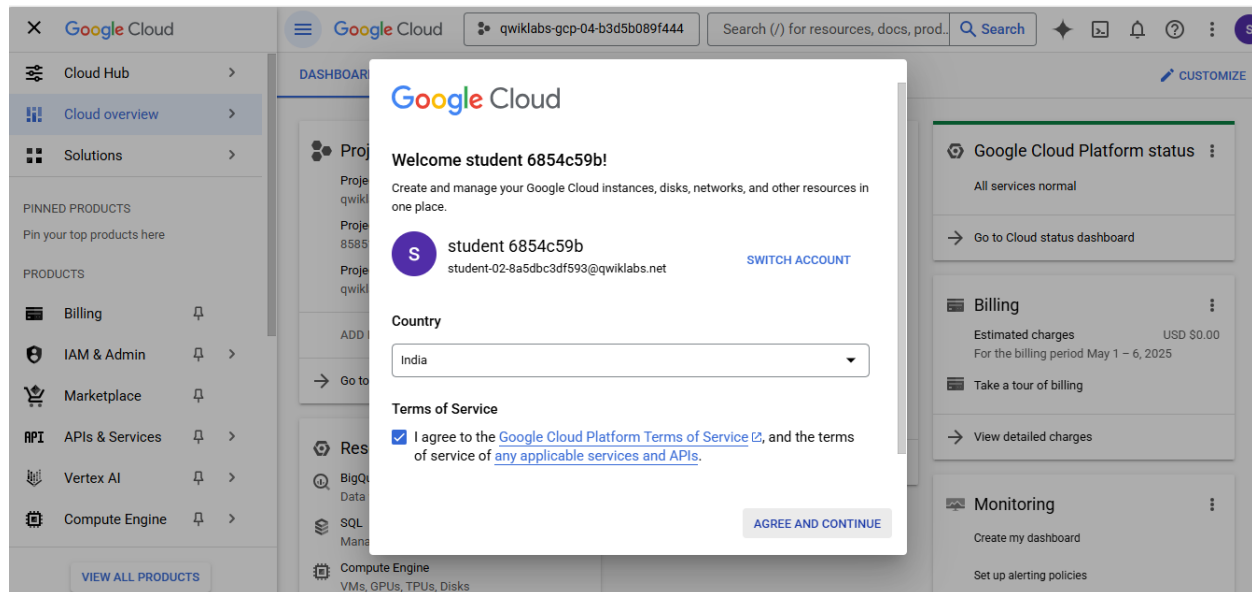
Open the BigQuery console

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 the release notes.

2. Click **Done**.

The BigQuery console opens.



Take a moment to note some important features of the UI. The right-hand side of the console houses the query "Editor". This is where you write and run SQL commands like the examples covered earlier. Below that is "Query history", which is a list of queries you ran previously.

The left pane of the console is the **Navigation menu**. Apart from the self-explanatory query history, saved queries, and job history, there is the *Explorer* tab.

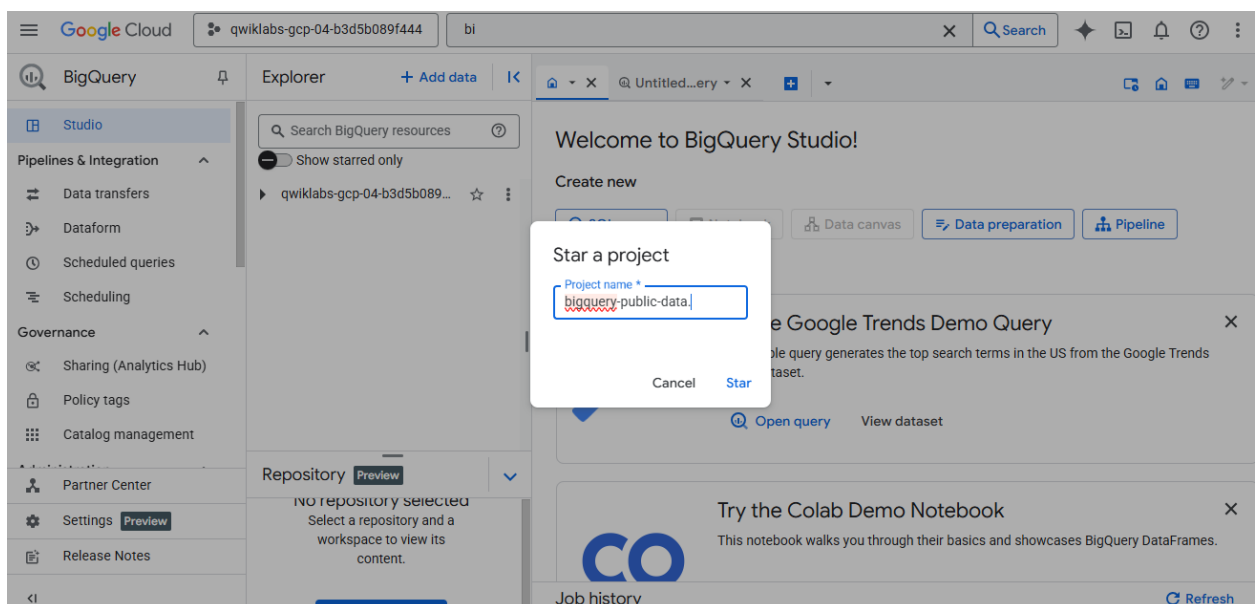
The highest level of resources in the *Explorer* tab contain Google Cloud projects, which are just like the temporary Google Cloud projects you sign into and use with each Google Cloud Skills Boost lab. As you can see in your console and in the last screenshot, you only have your project housed in the Explorer tab. If you try clicking on the arrow next to the project name, nothing will show up.

This is because your project contains no datasets or tables, you have nothing that can be queried. Earlier you learned datasets contain tables. When you add data to your project, note that in BigQuery, *projects contain datasets, and datasets contain tables*. Now that you better understand the project > dataset > table paradigm and the intricacies of the console, you can load up some queryable data.

Uploading queryable data

In this section you pull in some public data into your project so you can practice running SQL commands in BigQuery.

1. Click on **+ ADD**.
2. Choose **Star a project by name**.
3. Enter project name as **bigquery-public-data**.
4. Click **STAR**.



It's important to note that you are still working out of your lab project in this new tab. All you did was pull a publicly accessible project that contains datasets and tables into BigQuery for analysis — you didn't *switch* over to that project. All of your jobs and services are still tied

to your Google Cloud Skills Boost account. You can see this for yourself by inspecting the project field near the top of the console:

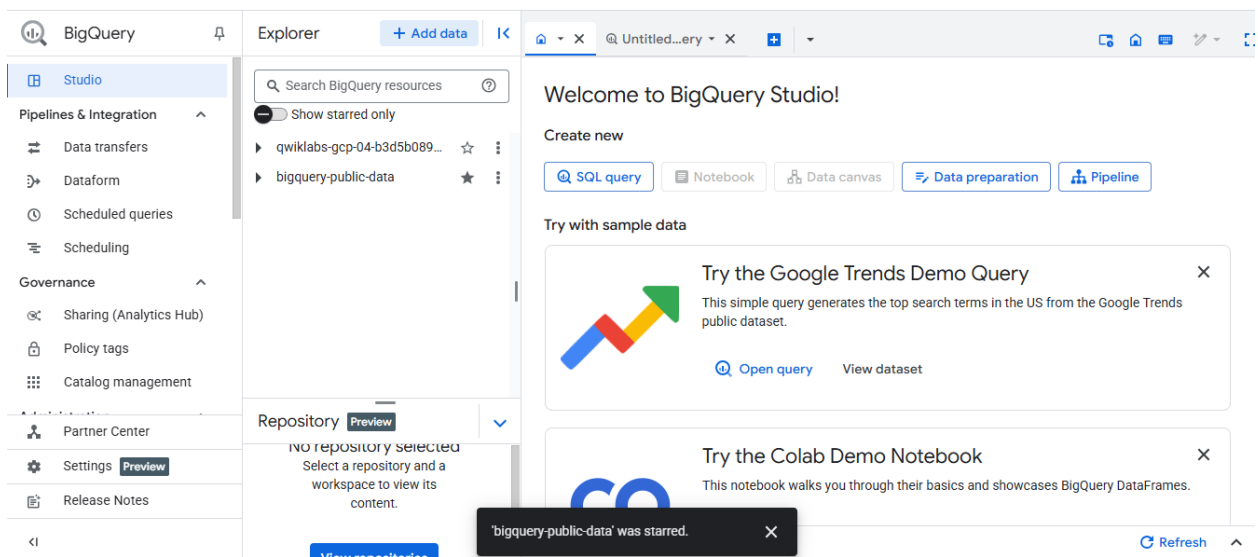
5. You now have access to the following data:

- Google Cloud Project → bigquery-public-data
- Dataset → london_bicycles

6. Click on the **london bicycles** dataset to reveal the associated tables

- Table → cycle_hire
- Table → cycle_stations

In this lab you will use the data from **cycle_hire**. Open the cycle_hire table, then click the **Preview** tab. Your page should resemble the following:



The screenshot shows the BigQuery Studio interface. On the left, the 'Studio' tab is active. The Explorer pane shows a search for 'london_bicycles' with results for 'bigquery-public-data' and 'london_bicycles', including 'cycle_hire' and 'cycle_stations'. The main pane displays the 'cycle_hire' table schema with columns: rental_id, duration, duration_ms, bike_id, bike_model, end_date, end_station_id, end_station_name, and start_date. The 'end_station_name' column is highlighted as the key.

Field name	Type	Mode	Key	Collation	Default Value
rental_id	INTEGER	REQUIRED	-	-	-
duration	INTEGER	NULLABLE	-	-	-
duration_ms	INTEGER	NULLABLE	-	-	-
bike_id	INTEGER	NULLABLE	-	-	-
bike_model	STRING	NULLABLE	-	-	-
end_date	TIMESTAMP	NULLABLE	-	-	-
end_station_id	INTEGER	NULLABLE	-	-	-
end_station_name	STRING	NULLABLE	-	-	-
start_date	TIMESTAMP	NULLABLE	-	-	-

Inspect the columns and values populated in the rows. You are now ready to run some SQL queries on the `cycle_hire` table.

Running **SELECT**, **FROM**, and **WHERE** in BigQuery

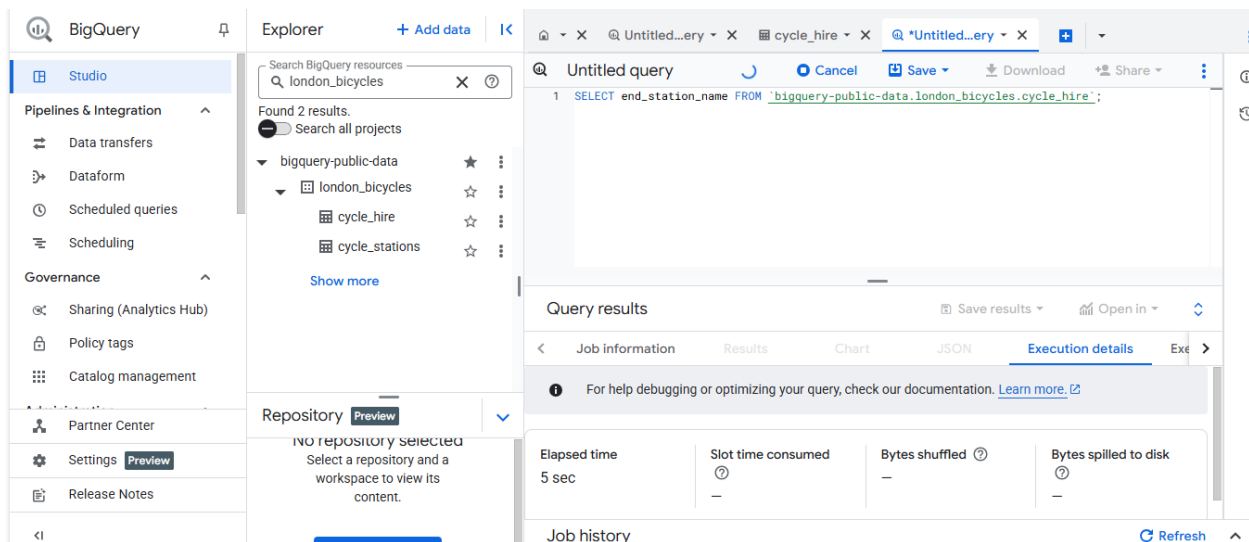
You now have a basic understanding of SQL querying keywords and the BigQuery data paradigm and some data to work with. Run some SQL commands using this service.

If you look at the bottom right corner of the console, you will notice that there are **83,434,866** rows of data, or individual bikeshare trips taken in London between 2015 and 2017 (not a small amount by any means!)

Now take note of the ninth column key: `end_station_name`, which specifies the end destination of bikeshare rides. Before getting too deep, run a simple query to isolate the `end_station_name` column.

1. Copy and paste the following command into the query **Editor**:

```
SELECT end_station_name FROM `bigquery-public-data.london_bicycles.cycle_hire` ;
```

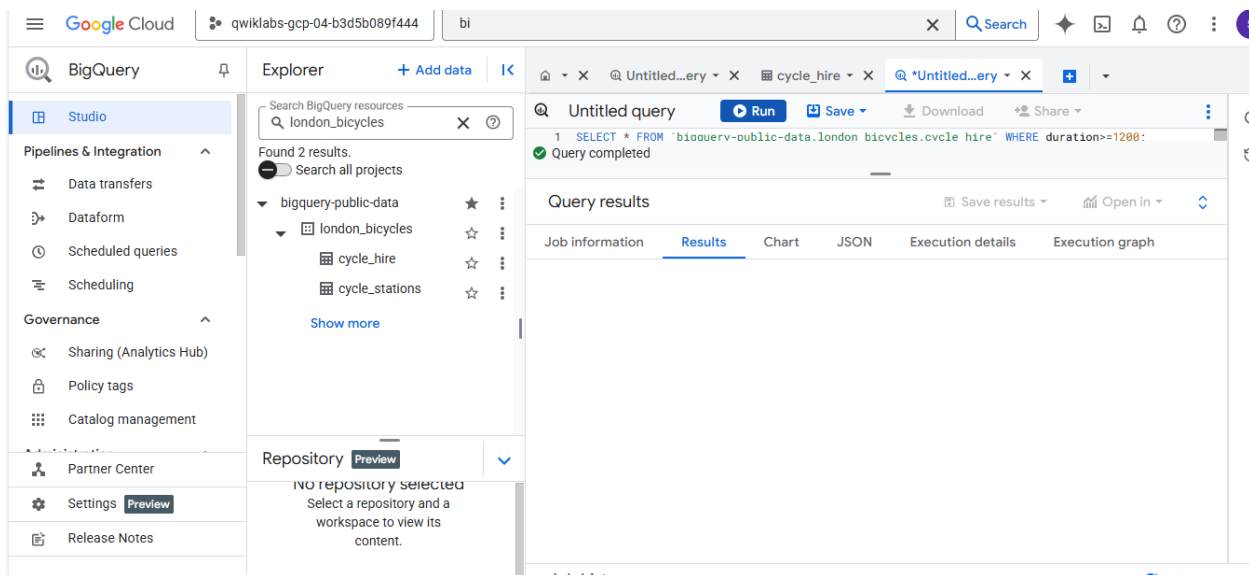


2. Then click **Run**.

After ~20 seconds, you should be returned with 83434866 rows that contain the single column you queried for: end_station_name.

3. Clear the query from the editor, then run the following query that utilizes the WHERE keyword:

```
SELECT * FROM `bigquery-public-data.london_bicycles.cycle_hire` WHERE duration>=1200;
```



This query may take a minute or so to run.

SELECT * returns all column values from the table. Duration is measured in seconds, which is why you used the value 1200 (60 * 20).

If you look in the bottom right corner you see that **26,441,016** rows were returned. As a fraction of the total (26441016/83434866), this means that ~30% of London bikeshare rides lasted 20 minutes or longer (they're in it for the long haul!)

Task 3. More SQL Keywords: GROUP BY, COUNT, AS, and ORDER BY

GROUP BY

The GROUP BY keyword will aggregate result-set rows that share common criteria (e.g. a column value) and will return all of the unique entries found for such criteria.

This is a useful keyword for figuring out categorical information on tables.

1. To get a better picture of what this keyword does, clear the query from the editor, then copy and paste the following command:

```
SELECT start_station_name FROM `bigquery-public-data.london_bicycles.cycle_hire`  
GROUP BY start_station_name;
```

The screenshot shows the Google BigQuery Studio interface. On the left is the 'Explorer' sidebar with a search bar containing 'london_bicycles'. It lists 'bigquery-public-data' > 'london_bicycles' > 'cycle_hire'. The main editor area shows a query: `SELECT start_station_name FROM `bigquery-public-data.london_bicycles.cycle_hire` GROUP BY start_station_name;`. Below the query editor, a 'Query completed' message is visible. The 'Query results' section shows a table with 8 rows of unique station names: Cardinal Place, Victoria; Godliman Street, St. Paul's; Whitehall Place, Strand; Fournier Street, Whitechapel; Somerset House, Strand; Eversholt Street, Camden Town; Great Suffolk Street, The Borou...; and Snow Hill, Farringdon. At the bottom right, it indicates 'Results per page: 50' and '1 - 50 of 954'.

2. Click **Run**.

Your results are a list of unique (non-duplicate) column values.

Without the GROUP BY, the query would have returned the full **83,434,866** rows. GROUP BY will output the unique column values found in the table. You can see this for yourself by looking in the bottom right corner. You will see **954** rows, meaning there are 954 distinct London bikeshare starting points.

COUNT

The COUNT() function will return the number of rows that share the same criteria (e.g. column value). This can be very useful in tandem with a GROUP BY.

Add the COUNT function to our previous query to figure out how many rides begin at each starting point.

- Clear the query from the editor, then copy and paste the following command and then click **Run**:

```
SELECT      start_station_name,      COUNT(*)      FROM      `bigquery-public-  
data.london_bicycles.cycle_hire` GROUP BY start_station_name;
```

The screenshot shows the Google Cloud BigQuery Studio interface. On the left is a sidebar with navigation options like 'Studio', 'Pipelines & Integration', 'Governance', and 'Partner Center'. The main area is divided into three panes. The top pane shows the 'Explorer' with a search for 'london_bicycles' and a list of results including 'bigquery-public-data' and 'london_bicycles'. The middle pane shows the 'Repository' section with a message 'no repository selected'. The right pane shows the 'Query editor' with a query: `SELECT start_station_name, COUNT(*) FROM `bigquery-public-data.london_bicycles.cycle_hire` GROUP BY start_station_name;`. Below the editor, a 'Query completed' message is shown. The bottom pane displays the 'Query results' in a table format. The table has two columns: 'start_station_name' and 'count(*)'. The results are sorted by count in descending order.

Row	start_station_name	count(*)
1	Podium, Queen Elizabeth Olym...	691
2	Black Lion Gate, Kensington Ga...	459716
3	Holborn Circus, Holborn	226050
4	Queen's Circus, Battersea Park	196722
5	Queen Street 2, Bank	200845
6	Macclesfield Rd, St Lukes	116742
7	Sail Street, Vauxhall	63454
8	Milrov Walk, South Bank	149449

Your output shows how many bikeshare rides begin at each starting location.

AS

SQL also has an AS keyword, which creates an *alias* of a table or column. An alias is a new name that's given to the returned column or table—whatever AS specifies.

1. Add an AS keyword to the last query you ran to see this in action. Clear the query from the editor, then copy and paste the following command:

```
SELECT      start_station_name,      COUNT(*) AS num_starts      FROM      `bigquery-public-  
data.london_bicycles.cycle_hire` GROUP BY start_station_name;
```

The screenshot shows the BigQuery Studio interface. The Explorer pane on the left shows a search for 'london_bicycles' with two results: 'bigquery-public-data' and 'london_bicycles'. The main editor shows a SQL query: 'SELECT start_station_name, COUNT(*) AS num_starts FROM `bigquery-public-data.london_bicycles.cycle_hire` GROUP BY start_station_name;'. The query is completed, and the results are displayed in a table with 8 rows. The table has columns 'start_station_name' and 'num_starts'. The results are sorted by 'num_starts' in descending order.

Row	start_station_name	num_starts
1	Podium, Queen Elizabeth Olym...	139591
2	Black Lion Gate, Kensington Ga...	459716
3	Holborn Circus, Holborn	226050
4	Queen's Circus, Battersea Park	196722
5	Queen Street 2, Bank	200845
6	Macclesfield Rd, St Lukes	116742
7	Sail Street, Vauxhall	63454
8	Milrov Walk, South Bank	149449

2. Click **Run**.

For Results, the right column name changed from COUNT(*) to num_starts.

As you see, the COUNT(*) column in the returned table is now set to the alias name num_starts. This is a handy keyword to use especially if you are dealing with large sets of data — forgetting that an ambiguous table or column name happens more often than you think!

ORDER BY

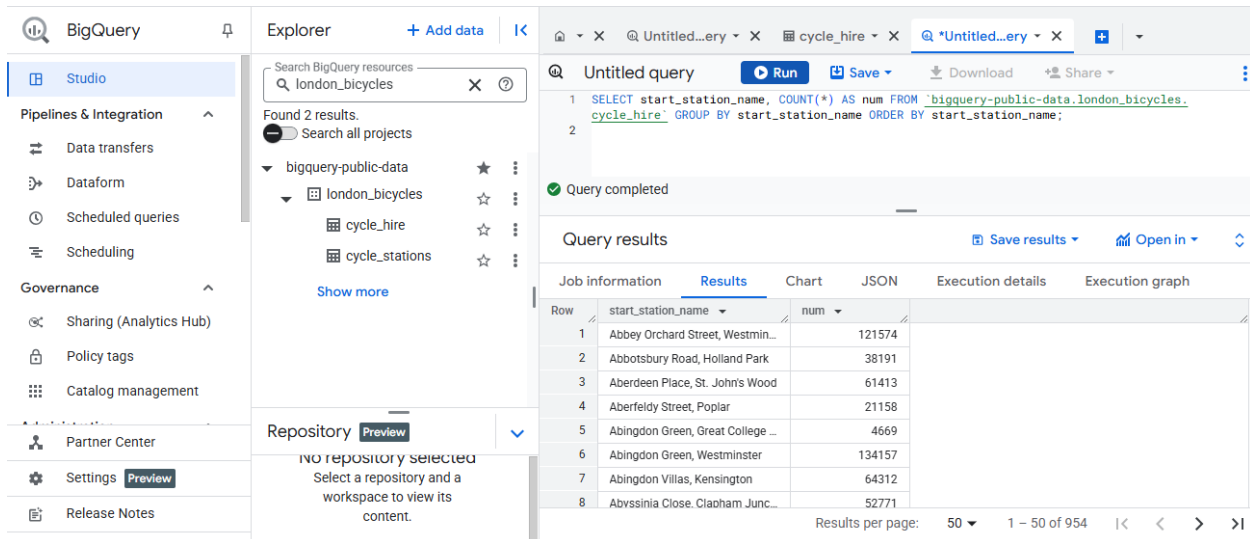
The ORDER BY keyword sorts the returned data from a query in ascending or descending order based on a specified criteria or column value. Add this keyword to our previous query to do the following:

- Return a table that contains the number of bikeshare rides that begin at each starting station, organized alphabetically by the starting station.
- Return a table that contains the number of bikeshare rides that begin at each starting station, organized numerically from lowest to highest.
- Return a table that contains the number of bikeshare rides that begin at each starting station, organized numerically from highest to lowest.

Each of the commands below is a separate query. For each command:

1. Clear the query **Editor**.
2. Copy and paste the command into the query **Editor**.
3. Click **Run**. Examine the results.

```
SELECT start_station_name, COUNT(*) AS num FROM `bigquery-public-data.london_bicycles.cycle_hire` GROUP BY start_station_name ORDER BY start_station_name;
```



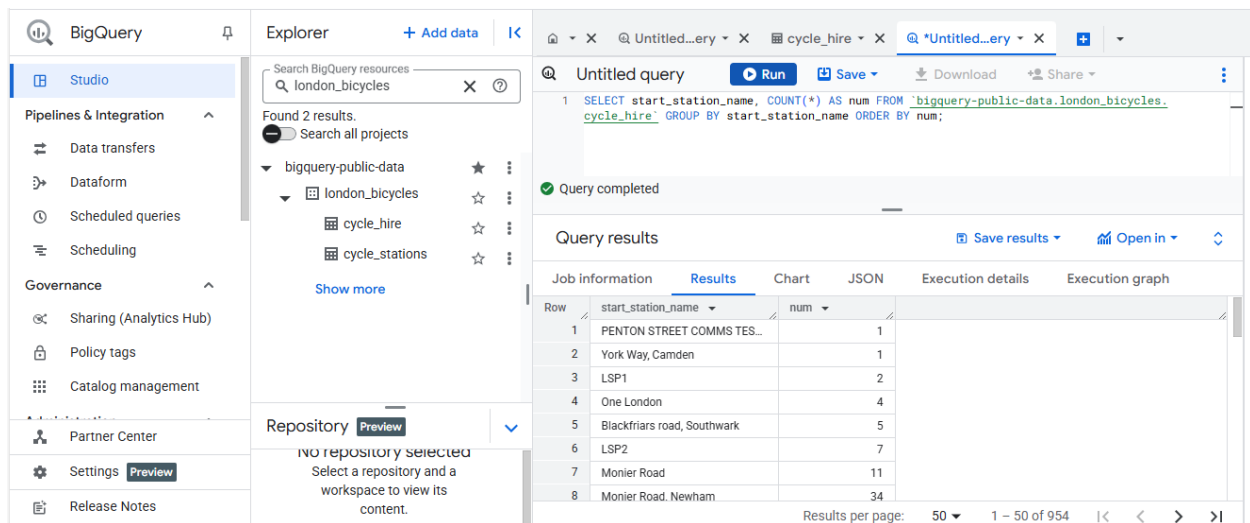
The screenshot shows the BigQuery Studio interface. On the left is the navigation menu with sections like 'Pipelines & Integration' and 'Governance'. The 'Explorer' pane shows a search for 'london_bicycles' with two results: 'bigquery-public-data' and 'london_bicycles'. The 'Query Editor' pane contains the following SQL query:

```
1 SELECT start_station_name, COUNT(*) AS num FROM `bigquery-public-data.london_bicycles.cycle_hire` GROUP BY start_station_name ORDER BY start_station_name;
```

The 'Query results' pane shows the results of the query in a table format. The table has two columns: 'start_station_name' and 'num'. The results are ordered by 'start_station_name'.

Row	start_station_name	num
1	Abbey Orchard Street, Westmin...	121574
2	Abbotsbury Road, Holland Park	38191
3	Aberdeen Place, St. John's Wood	61413
4	Aberfeldy Street, Poplar	21158
5	Abingdon Green, Great College ...	4669
6	Abingdon Green, Westminster	134157
7	Abingdon Villas, Kensington	64312
8	Abyssinia Close, Clapham Junc...	52771

```
SELECT start_station_name, COUNT(*) AS num FROM `bigquery-public-data.london_bicycles.cycle_hire` GROUP BY start_station_name ORDER BY num;
```



The screenshot shows the BigQuery Studio interface. The 'Query Editor' pane contains the following SQL query:

```
1 SELECT start_station_name, COUNT(*) AS num FROM `bigquery-public-data.london_bicycles.cycle_hire` GROUP BY start_station_name ORDER BY num;
```

The 'Query results' pane shows the results of the query in a table format. The table has two columns: 'start_station_name' and 'num'. The results are ordered by 'num' in descending order.

Row	start_station_name	num
1	PENTON STREET COMMMS TES...	1
2	York Way, Camden	1
3	LSP1	2
4	One London	4
5	Blackfriars road, Southwark	5
6	LSP2	7
7	Monier Road	11
8	Monier Road, Newham	34

```
SELECT start_station_name, COUNT(*) AS num FROM `bigquery-public-data.london_bicycles.cycle_hire` GROUP BY start_station_name ORDER BY num DESC;
```

The screenshot shows the BigQuery Studio interface. On the left is the navigation menu with sections like 'Pipelines & Integration' and 'Governance'. The 'Explorer' pane shows a search for 'london_bicycles' with results under 'bigquery-public-data'. The main pane displays an 'Untitled query' with the following SQL:

```
1 SELECT start_station_name, COUNT(*) AS num FROM `bigquery-public-data.london_bicycles.cycle_hire` GROUP BY start_station_name ORDER BY num DESC;
```

The query is completed, and the 'Query results' pane shows a table with 8 rows. The table has columns 'start_station_name' and 'num'.

Row	start_station_name	num
1	Hyde Park Corner, Hyde Park	671688
2	Belgrove Street, King's Cross	593065
3	Waterloo Station 3, Waterloo	527112
4	Albert Gate, Hyde Park	461108
5	Black Lion Gate, Kensington Ga...	459716
6	Waterloo Station 1, Waterloo	419334
7	Wellington Arch, Hyde Park	392889
8	Hoo Exchange, The Borough	385926

At the bottom, it indicates 'Results per page: 50' and '1 - 50 of 954'.

The results of the last query lists start locations by the number of starts from that location.

You see that "Hyde Park Corner, Hyde Park" has the highest number of starts. However, as a fraction of the total (671688/83434866), you see that < 1% of rides start from this station.

Task 4. Working with Cloud SQL

Exporting queries as CSV files

Cloud SQL is a fully-managed database service that makes it easy to set up, maintain, manage, and administer your relational PostgreSQL and MySQL databases in the cloud. There are two formats of data accepted by Cloud SQL: dump files (.sql) or CSV files (.csv). You will learn how to export subsets of the cycle_hire table into CSV files and upload them to Cloud Storage as an intermediate location.

Back in the BigQuery Console, this should have been the last command that you ran:

```
SELECT start_station_name, COUNT(*) AS num FROM `bigquery-public-data.london_bicycles.cycle_hire` GROUP BY start_station_name ORDER BY num DESC;
```

The screenshot shows the BigQuery Studio interface. In the Explorer pane, the 'bigquery-public-data' project is expanded, and 'london_bicycles' is selected. The main editor displays a query that counts the number of bike hires by start station name. The Query Results pane shows the following data:

Row	start_station_name	num
1	Hyde Park Corner, Hyde Park	671688
2	Belgrove Street, King's Cross	593065
3	Waterloo Station 3, Waterloo	527112
4	Albert Gate, Hyde Park	461108
5	Black Lion Gate, Kensington Ga...	459716
6	Waterloo Station 1, Waterloo	419334
7	Wellington Arch, Hyde Park	392889
8	Hop Exchange, The Borough	385926

1. In the Query Results section click **SAVE RESULTS > CSV(local file)**. This initiates a download, which saves this query as a CSV file. Note the location and the name of this downloaded file—you will need it soon.
2. Clear the query Editor, then copy and run the following in the query editor:

```
SELECT end_station_name, COUNT(*) AS num FROM `bigquery-public-data.london_bicycles.cycle_hire` GROUP BY end_station_name ORDER BY num DESC;
```

The screenshot shows the BigQuery Studio interface with the second query. The Query Results pane shows the following data:

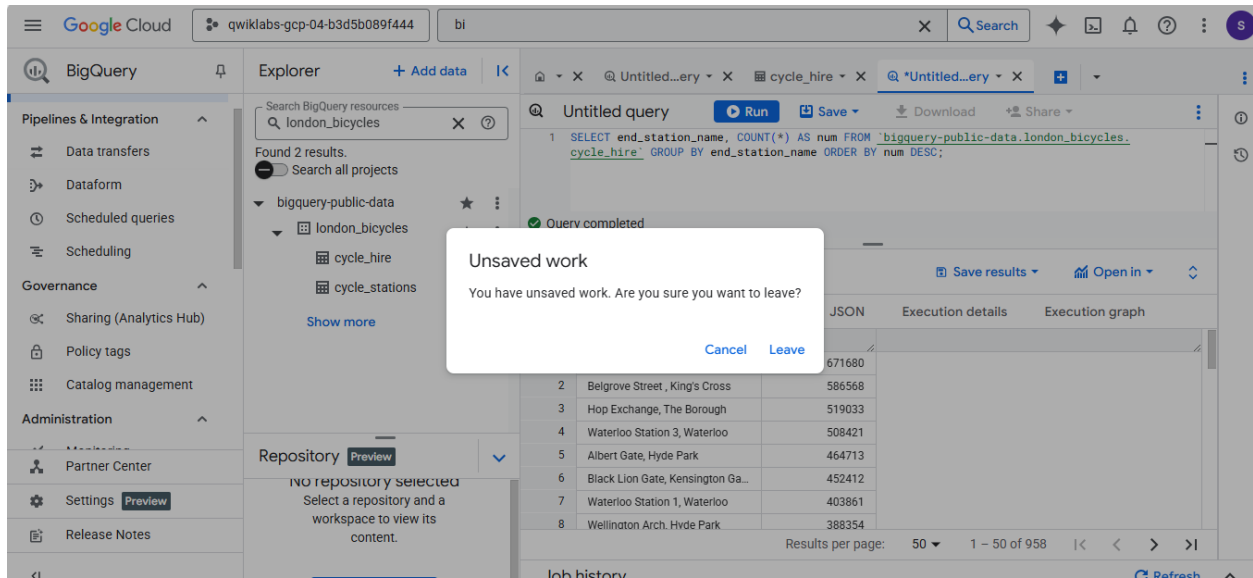
Row	end_station_name	num
1	Hyde Park Corner, Hyde Park	671680
2	Belgrove Street, King's Cross	586568
3	Hop Exchange, The Borough	519033
4	Waterloo Station 3, Waterloo	508421
5	Albert Gate, Hyde Park	464713
6	Black Lion Gate, Kensington Ga...	452412
7	Waterloo Station 1, Waterloo	403861
8	Wellington Arch, Hyde Park	388354

This returns a table that contains the number of bikeshare rides that finish in each ending station and is organized numerically from highest to lowest number of rides.

3. In the Query Results section click **SAVE RESULTS > CSV(local file)**. This initiates a download, which saves this query as a CSV file. Note the location and the name of this downloaded file—you will need it in the following section.

Upload CSV files to Cloud Storage

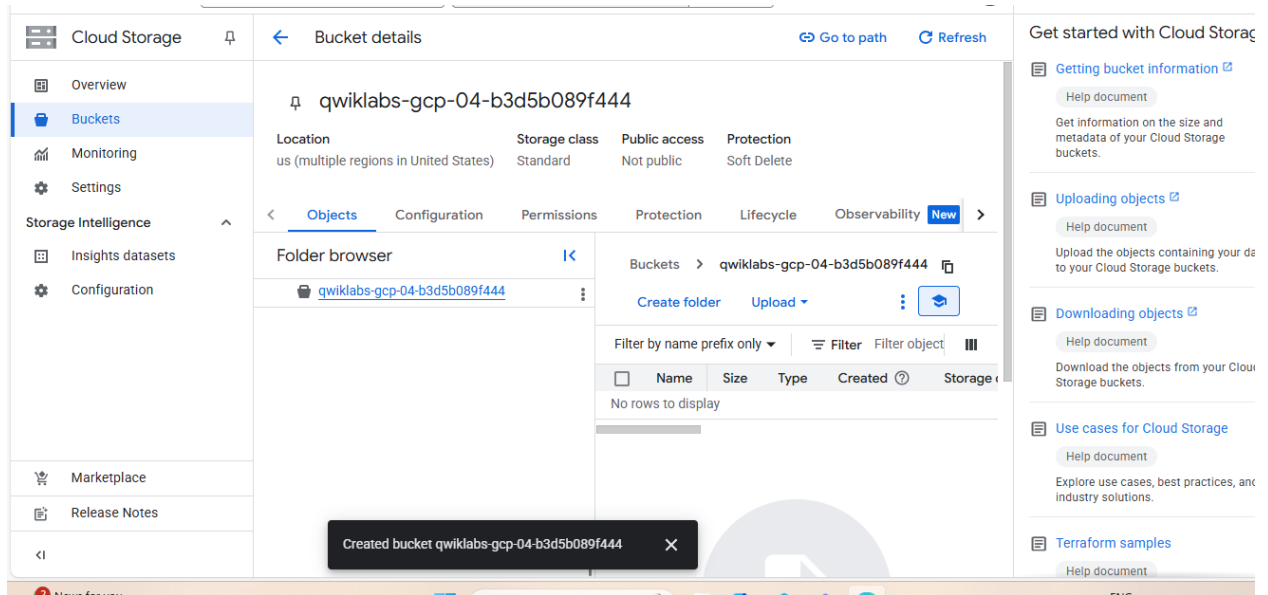
1. Go to the Cloud Console where you'll create a storage bucket where you can upload the files you just created.
2. Select **Navigation menu > Cloud Storage > Buckets**, and then click **CREATE BUCKET**.



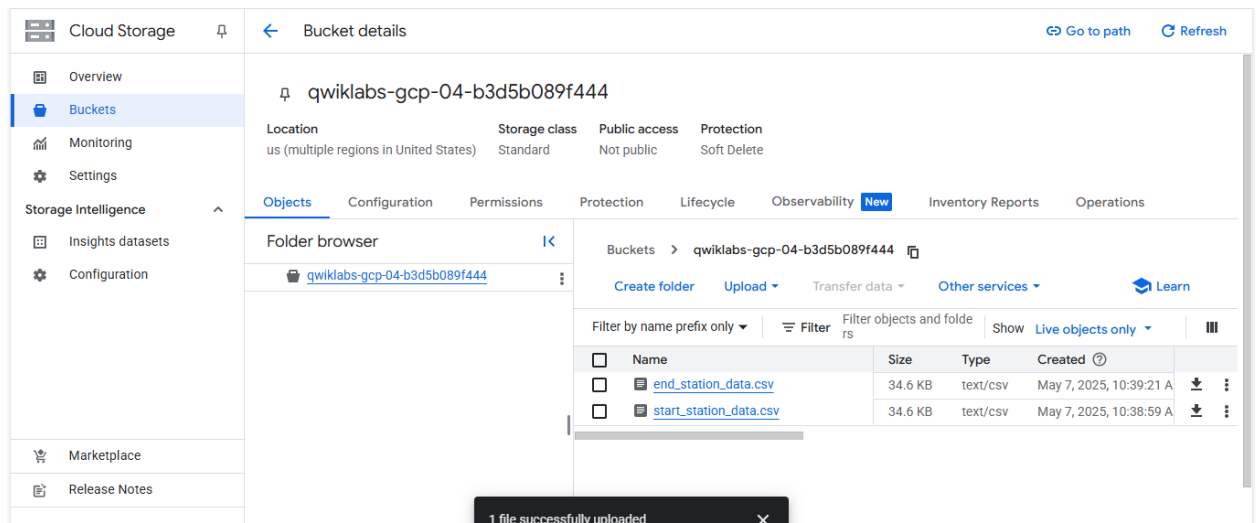
3. Enter a unique name for your bucket, keep all other settings as default, and click **Create**.
4. If prompted, click **Confirm** for Public access will be prevented dialog.

You should now be in the Cloud Console looking at your newly created Cloud Storage Bucket.

1. Click **UPLOAD > Upload files** and select the CSV that contains start_station_name data.
2. Then click **Open**. Repeat this for the end_station_name data.
3. Rename your start_station_name file by clicking on the three dots next to on the far side of the file and click **rename**. Rename the file to start_station_data.csv.



4. Rename your `end_station_name` file by clicking on the three dots next to on the far side of the file and click **rename**. Rename the file to `end_station_data.csv`.



You should now see `start_station_data.csv` and `end_station_data.csv` in the **Objects** list on the **Bucket details** page.

Task 5. Create a Cloud SQL instance

In the console, select **Navigation menu** > **SQL**.

1. Click **CREATE INSTANCE** > **Choose MySQL**.
2. Enter instance id as **my-demo**.

3. Enter a secure password in the **Password** field (remember it!).
4. Select the database version as **MySQL 8**.
5. For **Choose a Cloud SQL edition**, select **Enterprise**.
6. For **Edition preset**, select **Development** (4 vCPU, 16 GB RAM, 100 GB Storage, Single zone).

The image shows two screenshots of the Google Cloud SQL console. The top screenshot is the 'Cloud SQL' overview page, which includes a 'Get started' section with tabs for 'MYSQL', 'POSTGRESQL', and 'SQL SERVER'. It displays two instance presets: 'Sandbox' (MySQL 8.0, 2 vCPUs, 8 GB RAM, 100 GB SSD storage) and 'Development' (MySQL 8.0, 4 vCPUs, 32 GB RAM, 250 GB SSD storage). A notification '1 file successfully uploaded' is visible. The bottom screenshot shows the 'Overview' page for a specific instance named 'my-demo'. It indicates that the instance is being created and provides a timeline for the operation. A chart for 'CPU utilization' is shown, but it displays 'No data is available for the selected time frame.' The timeline ranges from UTC+5:30 to 10:00 AM on May 7.

7. Set the **Region** field as <Lab Region>.
8. Set the **Multi zones (Highly available) > Primary Zone** field as <Lab Zone>.
9. Click **CREATE INSTANCE**.
10. Click on the Cloud SQL instance. The **SQL Overview** page opens.

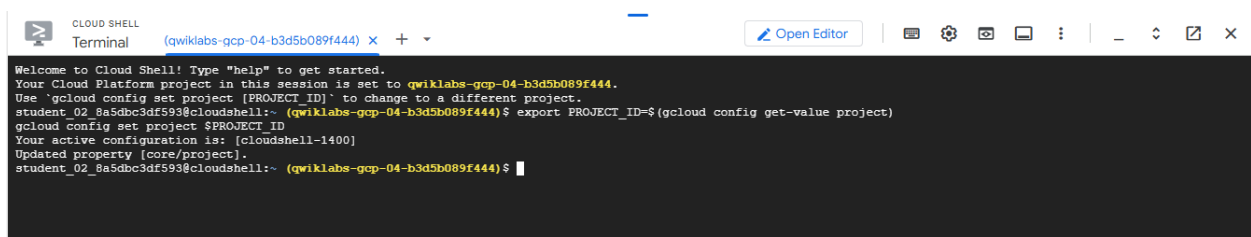
Task 6. New queries in Cloud SQL

CREATE keyword (databases and tables)

Now that you have a Cloud SQL instance up and running, create a database inside of it using the Cloud Shell Command Line.

1. Open Cloud Shell by clicking on the icon in the top right corner of the console.
2. Run the following command to set your project ID as an environment variable:

```
export PROJECT_ID=$(gcloud config get-value project)
gcloud config set project $PROJECT_ID
```

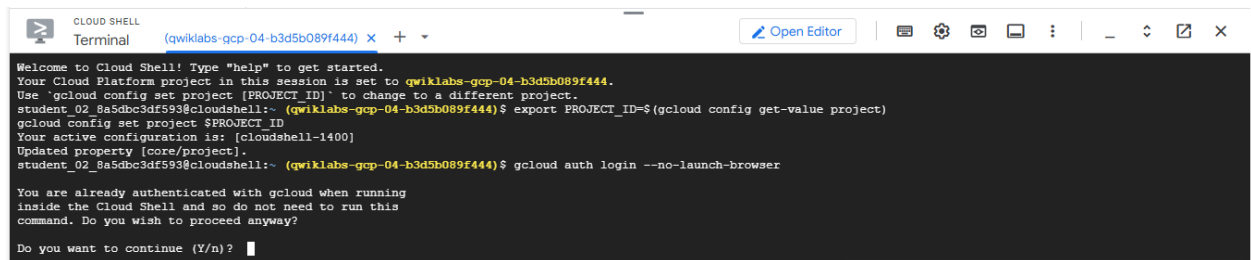


A screenshot of the Cloud Shell terminal interface. The terminal shows a welcome message and the execution of the command `gcloud config set project $(gcloud config get-value project)`. The output indicates that the project ID is `qwiklabs-gcp-04-b3d5b089f444` and that the configuration has been updated.

Create a database in Cloud Shell

1. Run the following command in Cloud Shell to setup auth without opening up a browser.

```
gcloud auth login --no-launch-browser
```



A screenshot of the Cloud Shell terminal interface. The terminal shows the execution of the command `gcloud auth login --no-launch-browser`. The output indicates that the user is already authenticated and asks if they want to proceed anyway. The prompt is `Do you want to continue (Y/n)?`.

If prompted [Y/n], press **Y** and then **ENTER**.

This will give you a link to open in your browser. Open the link in the same browser where you are logged in to the qwiklabs account. Once you login you will get a verification code to copy. Paste that code in the cloud shell.

2. Run the following command to connect to your SQL instance, replacing my-demo if you used a different name for your instance:

```
gcloud sql connect my-demo --user=root --quiet
```



```

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> CREATE DATABASE bike;
Query OK, 1 row affected (0.22 sec)

mysql> USE bike;
Database changed
mysql> CREATE TABLE london1 (start_station_name VARCHAR(255), num INT);
Query OK, 0 rows affected (0.27 sec)

mysql> USE bike;
Database changed
mysql> CREATE TABLE london2 (end_station_name VARCHAR(255), num INT);
Query OK, 0 rows affected (0.26 sec)

mysql> 

```

This statement uses the CREATE keyword, but this time it uses the TABLE clause to specify that it wants to build a table instead of a database. The USE keyword specifies a database that you want to connect to. You now have a table named "london1" that contains two columns, "start_station_name" and "num". VARCHAR(255) specifies variable length string column that can hold up to 255 characters and INT is a column of type integer.

2. Create another table named "london2" by running the following command:

```

USE bike;
CREATE TABLE london2 (end_station_name VARCHAR(255), num INT);

```

3. Now confirm that your empty tables were created. Run the following commands at the MySQL server prompt:

```

SELECT * FROM london1;
SELECT * FROM london2;

```

```
mysql> USE bike;
Database changed
mysql> CREATE TABLE london1 (start_station_name VARCHAR(255), num INT);
Query OK, 0 rows affected (0.27 sec)

mysql> USE bike;
Database changed
mysql> CREATE TABLE london2 (end_station_name VARCHAR(255), num INT);
Query OK, 0 rows affected (0.26 sec)

mysql> SELECT * FROM london1;
Empty set (0.22 sec)

mysql> SELECT * FROM london2;
Empty set (0.21 sec)

mysql> 
```

Return to the Cloud SQL console. You will now upload the start_station_name and end_station_name CSV files into your newly created london1 and london2 tables.

1. In your Cloud SQL instance page, click **IMPORT**.
2. In the Cloud Storage file field, click **Browse**, and then click the arrow next to your bucket name, and then click start_station_data.csv. Click **Select**.
3. Select **CSV** as File format.
4. Select the bike database and type in london1 as your table.
5. Click **Import**.

SQL

Primary instance

- Overview
- Cloud SQL Studio
- System insights
- Query insights
- Connections
- Users
- Databases
- Backups
- Replicas
- Operations
- Release Notes

Import data from Cloud Storage

File format

☐ SQL
A plain text file with a sequence of SQL commands, like the output of mysqldump

☒ CSV
If your Cloud Storage file is a CSV file, select CSV. The CSV file should be a plain text file with one line per row and comma-separated fields.

Destination

Choose the database and table in your instance for this file to import into. [Learn more](#)

Database *
bike

Table *
london1
Enter the name of an existing table in the database to house your CSV file

When you import, a Cloud SQL service account will be granted read access to the selected file and bucket, which will be reflected in your permissions.

IMPORTING CANCEL

Do the same for the other CSV file.

1. In your Cloud SQL instance page, click **IMPORT**.

2. In the Cloud Storage file field, click **Browse**, and then click the arrow next to your bucket name, and then click end_station_data.csv Click **Select**.
3. Select **CSV** as File format.
4. Select the bike database and type in london2 as your table.
5. Click **Import**.

You should now have both CSV files uploaded to tables in the bike database.

1. Return to your Cloud Shell session and run the following command at the MySQL server prompt to inspect the contents of london1:

```
SELECT * FROM london1;
```

```
| Exhibition Road, South Kensington | 270 |
| Import Dock | 142 |
| Crimscott Street, Bermondsey | 139 |
| Pop Up Dock 1 | 126 |
| Wansey Street, Walworth | 97 |
| Cartier Circle, Canary Wharf | 75 |
| Pop Up Dock 2 | 70 |
| Contact Centre, Southbury House | 60 |
| Allington street, Off Victoria Street, Westminster | 55 |
| Monier Road, Newham | 34 |
| Monier Road | 11 |
| LSP2 | 7 |
| Blackfriars road, Southwark | 5 |
| One London | 4 |
| LSP1 | 2 |
| PENTON STREET COMMS TEST TERMINAL _ CONTACT MATT McNULTY | 1 |
| York Way, Camden | 1 |
+-----+-----+
955 rows in set (0.43 sec)
```

You should receive 955 lines of output, one for each unique station name.

2. Run the following command to make sure that london2 has been populated:

```
SELECT * FROM london2;
```

```

| Import Dock | 146 |
| Crimscott Street, Bermondsey | 133 |
| Pop Up Dock 1 | 110 |
| Contact Centre, Southbury House | 60 |
| Cartier Circle, Canary Wharf | 58 |
| Pop Up Dock 2 | 52 |
| Allington street, Off Victoria Street, Westminster | 51 |
| Monier Road, Newham | 36 |
| Monier Road | 14 |
| PENTON STREET COMMS TEST TERMINAL _ CONTACT MATT McNULTY | 11 |
| LSP1 | 7 |
| Blackfriars road, Southwark | 5 |
| LSP2 | 5 |
| One London | 4 |
| Electrical Workshop PS | 2 |
| York Way, Camden | 1 |
| Canada Water Station | 1 |
+-----+-----+
959 rows in set (0.22 sec)

```

You should receive 959 lines of output, one more each unique station name.

DELETE keyword

Here are a couple more SQL keywords that help us with data management. The first is the DELETE keyword.

- Run the following commands in your MySQL session to delete the first row of the london1 and london2:

```
DELETE FROM london1 WHERE num=0;
DELETE FROM london2 WHERE num=0;
```

```

mysql> DELETE FROM london1 WHERE num=0;
Query OK, 1 row affected (0.22 sec)

mysql> DELETE FROM london2 WHERE num=0;
Query OK, 1 row affected (0.22 sec)

```

You can also insert values into tables with the INSERT INTO keyword.

- Run the following command to insert a new row into london1, which sets start_station_name to "test destination" and num to "1":

```
INSERT INTO london1 (start_station_name, num) VALUES ("test destination", 1);
```

```

mysql> INSERT INTO london1 (start_station_name, num) VALUES ("test destination", 1);
Query OK, 1 row affected (0.22 sec)

```

The INSERT INTO keyword requires a table (london1) and will create a new row with columns specified by the terms in the first parenthesis (in this case "start_station_name" and "num"). Whatever comes after the "VALUES" clause will be inserted as values in the new row.

If you run the query SELECT * FROM london1; you will see an additional row added at the bottom of the "london1" table.

```
SELECT start_station_name AS top_stations, num FROM london1 WHERE num>100000
UNION
```

```
SELECT end_station_name, num FROM london2 WHERE num>100000
ORDER BY top_stations DESC;
```

```
| All Saints Church, Portobello | 139718 |
| All Saints Church, Portobello | 137984 |
| Aldersgate Street, Barbican | 190311 |
| Aldersgate Street, Barbican | 183256 |
| Alderney Street, Pimlico | 128339 |
| Alderney Street, Pimlico | 120167 |
| Albert Gate, Hyde Park | 464713 |
| Albert Gate, Hyde Park | 461108 |
| Albert Bridge Road, Battersea Park | 131622 |
| Albert Bridge Road, Battersea Park | 127936 |
| Albany Street, The Regent's Park | 101765 |
| Ada Street, Hackney Central | 106514 |
| Ada Street, Hackney Central | 111076 |
| Abingdon Green, Westminster | 159305 |
| Abingdon Green, Westminster | 134157 |
| Abbey Orchard Street, Westminster | 121574 |
| Abbey Orchard Street, Westminster | 145521 |
+-----+
629 rows in set (0.22 sec)
```

As you see, 13/14 stations share the top spots for rideshare starting and ending points. With some basic SQL keywords you were able to query a sizable dataset, which returned data points and answers to specific questions.