Modernizing Data Lakes and Data Warehouses with Google Cloud

Course · 8 hours
50% complete
home

Course overview

- Loading Taxi Data into Google Cloud SQL 2.5 1 hourNo cost

Overview

In this lab, you will learn how to import data from CSV text files into Cloud SQL and then carry out some basic data analysis using simple queries. The dataset used in this lab is collected by the NYC Taxi and Limousine Commission and includes trip records from all trips completed in Yellow and Green taxis in NYC from 2009 to present, and all trips in for-hire vehicles (FHV) from 2015 to present. Records include fields capturing pick-up and drop-off dates/times, pick-up and drop-off locations, trip distances, itemized fares, rate types, payment types, and driver-reported passenger counts.
This dataset can be used to demonstrate a wide range of data science concepts and techniques and will be used in several of the labs in the Data Engineering curriculum.

Objectives

- Create Cloud SQL instance
- Create a Cloud SQL database
- Import text data into Cloud SQL
- Check the data for integrity

Setup and requirements

For each lab, you get a new Google Cloud project and set of resources for a fixed time at no cost.

1. Sign in to Qwiklabs using an **incognito window**.

2. Note the lab's access time (for example, `1:15:00`), and make sure you can finish within that time.
There is no pause feature. You can restart if needed, but you have to start at the beginning.

3. When ready, click **Start lab**.

4. Note your lab credentials (**Username** and **Password**). You will use them to sign in to the Google Cloud Console.

5. Click **Open Google Console**.

6. Click **Use another account** and copy/paste credentials for **this** lab into the prompts.
If you use other credentials, you'll receive errors or **incur charges**.

7. Accept the terms and skip the recovery resource page.

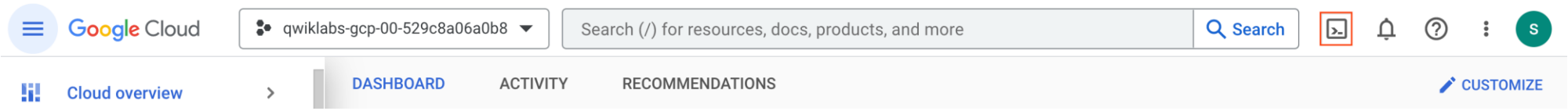
Note: Do not click **End Lab** unless you have finished the lab or want to restart it. This clears your work and removes the project.

Activate Google Cloud Shell

Google Cloud Shell is a virtual machine that is loaded with development tools. It offers a persistent 5GB home directory and runs on the Google Cloud.

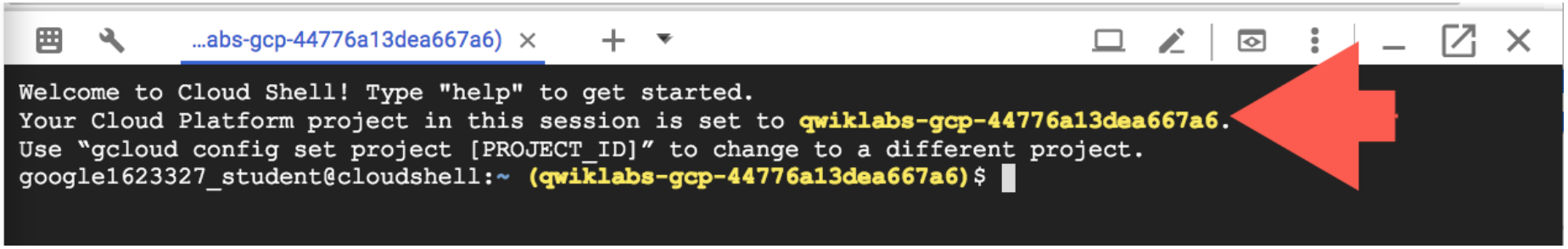
Google Cloud Shell provides command-line access to your Google Cloud resources.

1. In Cloud console, on the top right toolbar, click the Open Cloud Shell button.



2. Click **Continue**.

It takes a few moments to provision and connect to the environment. When you are connected, you are already authenticated, and the project is set to your *PROJECT_ID*. For example:



gcloud is the command-line tool for Google Cloud. It comes pre-installed on Cloud Shell and supports tab-completion.

- You can list the active account name with this command:

```
gcloud auth list
```

Copied!
content_copy
Output:

```
Credentialed accounts:
 - @.com (active)
```

Example output:

```
Credentialed accounts:
```

```
 - xxxxxxx27_student@qwiklabs.net
```

- You can list the project ID with this command:

```
gcloud config list project
```

Copied!
content_copy
Output:

```
[core]
project =
```

Example output:

```
[core]
project = qwiklabs-gcp-4xxxxxxxxa6
```

Note: Full documentation of **gcloud** is available in the gcloud CLI overview guide .

Task 1. Preparing your environment

- Create environment variables that will be used later in the lab for your project ID and the storage bucket that will contain your data:

```
export PROJECT_ID=$(gcloud info --format='value(config.project)')
export BUCKET=${PROJECT_ID}-ml
```

Copied!
content_copy

Task 2. Create a Cloud SQL instance

1. Enter the following commands to create a Cloud SQL instance:

```
gcloud sql instances create taxi \
    --tier=db-n1-standard-1 --activation-policy=ALWAYS
```

Copied!
content_copy
This will take a few minutes to complete.

Test completed task

Click **Check my progress** to verify your performed task. If you have completed the task successfully you will be granted an assessment score.

Create a Cloud SQL instance.

Check my progress

2. Set a root password for the Cloud SQL instance:

```
gcloud sql users set-password root --host % --instance taxi \
 --password Passw0rd
```

Copied!
content_copy

3. When prompted for the password type `Passw0rd` and press enter this will update root password.

4. Now create an environment variable with the IP address of the Cloud Shell:

```
export ADDRESS=$(wget -qO - http://ipecho.net/plain)/32
```

Copied!
content_copy

5. Whitelist the Cloud Shell instance for management access to your SQL instance:

```
gcloud sql instances patch taxi --authorized-networks $ADDRESS
```

Copied!
content_copy

6. When prompted press **Y** to accept the change.

Test completed task

Click **Check my progress** to verify your performed task. If you have completed the task successfully you will be granted an assessment score.

Whitelist the Cloud Shell instance to access your SQL instance.

Check my progress

7. Get the IP address of your Cloud SQL instance by running:

```
MYSQLIP=$(gcloud sql instances describe \
taxi --format="value(ipAddresses.ipAddress)")
```

Copied!
content_copy

8. Check the variable MYSQLIP:

```
echo $MYSQLIP
```

Copied!
content_copy
You should get an IP address as an output.

9. Create the taxi trips table by logging into the `mysql` command line interface:

```
mysql --host=$MYSQLIP --user=root \
      --password --verbose
```

Copied!
content_copy

10. When prompted for a password enter `Passw0rd`.

11. Paste the following content into the command line to create the schema for the `trips` table:

```
create database if not exists bts;
use bts;

drop table if exists trips;

create table trips (
  vendor_id VARCHAR(16),
  pickup_datetime DATETIME,
  dropoff_datetime DATETIME,
  passenger_count INT,
  trip_distance FLOAT,
  rate_code VARCHAR(16),
  store_and_fwd_flag VARCHAR(16),
  payment_type VARCHAR(16),
  fare_amount FLOAT,
```

```
  extra FLOAT,
  mta_tax FLOAT,
  tip_amount FLOAT,
  tolls_amount FLOAT,
  imp_surcharge FLOAT,
  total_amount FLOAT,
  pickup_location_id VARCHAR(16),
  dropoff_location_id VARCHAR(16)
);
```
Copied!
content_copy
Test completed task

Click **Check my progress** to verify your performed task. If you have completed the task successfully you will be granted an assessment score.

Create a bts database and trips table.

Check my progress

12. In the `mysql` command line interface check the import by entering the following commands:

```
describe trips;
```
Copied!
content_copy

13. Query the `trips` table:

```
select distinct(pickup_location_id) from trips;
```
Copied!
content_copy
This will return an empty set as there is no data in the database yet.

14. Exit the `mysql` interactive console:

```
exit
```
Copied!
content_copy

Task 3. Add data to Cloud SQL instance

Now you'll copy the New York City taxi trips CSV files stored on Cloud Storage locally. To keep resource usage low, you'll only be working with a subset of the data (~20,000 rows).

1. Run the following in the command line:

```
gcloud storage cp gs://cloud-training/OCBL013/nyc_tlc_yellow_trips_2018_subset_1.csv trips.csv-1
gcloud storage cp gs://cloud-training/OCBL013/nyc_tlc_yellow_trips_2018_subset_2.csv trips.csv-2
```

Copied!
content_copy

2. Connect to the `mysql` interactive console to load local infile data:

```
mysql --host=$MYSQLIP --user=root  --password  --local-infile
```

Copied!
content_copy

3. When prompted for a password enter `Passw0rd`.

4. In the `mysql` interactive console select the database:

```
use bts;
```

Copied!
content_copy

5. Load the local CSV file data using `local-infile`:

```
LOAD DATA LOCAL INFILE 'trips.csv-1' INTO TABLE trips
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
IGNORE 1 LINES
(vendor_id,pickup_datetime,dropoff_datetime,passenger_count,trip_distance,rate_code,store_and_fwd_flag,payment_typ
e,fare_amount,extra,mta_tax,tip_amount,tolls_amount,imp_surcharge,total_amount,pickup_location_id,dropoff_location
_id);
```

Copied!
content_copy

```
LOAD DATA LOCAL INFILE 'trips.csv-2' INTO TABLE trips
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
IGNORE 1 LINES
(vendor_id,pickup_datetime,dropoff_datetime,passenger_count,trip_distance,rate_code,store_and_fwd_flag,payment_typ
e,fare_amount,extra,mta_tax,tip_amount,tolls_amount,imp_surcharge,total_amount,pickup_location_id,dropoff_location
_id);
```

Copied!
content_copy

Task 4. Checking for data integrity

Whenever data is imported from a source it's always important to check for data integrity. Roughly, this means making sure the data meets your expectations.

1. Query the `trips` table for unique pickup location regions:

```
select distinct(pickup_location_id) from trips;
```

Copied!
content_copy
This should return 159 unique ids.

2. Let's start by digging into the `trip_distance` column. Enter the following query into the console:

```
select
  max(trip_distance),
  min(trip_distance)
from
  trips;
```

Copied!
content_copy
One would expect the trip distance to be greater than 0 and less than, say 1000 miles. The maximum trip distance returned of 85 miles seems reasonable but the minimum trip distance of 0 seems buggy.

3. How many trips in the dataset have a trip distance of 0?

```
select count(*) from trips where trip_distance = 0;
```

Copied!
content_copy
There are 155 such trips in the database. These trips warrant further exploration. You'll find that these trips have non-zero payment amounts associated with them. Perhaps these are fraudulent transactions?

4. Let's see if we can find more data that doesn't meet our expectations. We expect the `fare_amount` column to be positive. Enter the following query to see if this is true in the database:

```
select count(*) from trips where fare_amount < 0;
```

Copied!
content_copy
There should be 14 such trips returned. Again, these trips warrant further exploration. There may be a reasonable explanation for why the fares take on negative numbers. However, it's up to the data engineer to ensure there are no bugs in the data pipeline that would cause such a result.

5. Finally, let's investigate the `payment_type` column.

```
select
  payment_type,
  count(*)
from
  trips
group by
  payment_type;
```

Copied!
content_copy
The results of the query indicate that there are four different payment types, with:

- Payment type = 1 has 13863 rows
- Payment type = 2 has 6016 rows
- Payment type = 3 has 113 rows
- Payment type = 4 has 32 rows

Digging into the documentation, a payment type of 1 refers to credit card use, payment type of 2 is cash, and a payment type of 4 refers to a dispute. The figures make sense.

6. Exit the 'mysql' interactive console:

```
exit
```

Copied!
content_copy

End your lab

When you have completed your lab, click **End Lab**. Google Cloud Skills Boost removes the resources you've used and cleans the account for you.

You will be given an opportunity to rate the lab experience. Select the applicable number of stars, type a comment, and then click **Submit**.

The number of stars indicates the following:

- 1 star = Very dissatisfied
- 2 stars = Dissatisfied
- 3 stars = Neutral
- 4 stars = Satisfied
- 5 stars = Very satisfied

You can close the dialog box if you don't want to provide feedback.

For feedback, suggestions, or corrections, please use the **Support** tab.