Loading Taxi Data into Google Cloud SQL 2.5

## 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](https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page) 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

### **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 GCP resources.

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



1. 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 Platform. 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:

- <myaccount>@<mydomain>.com (active)

Example output:

Credentialed accounts:

- google1623327\_student@qwiklabs.net

You can list the project ID with this command:

gcloud config list project

Copied!

content\_copy

Output:

[core]

project = <project\_ID>

Example output:

[core]

project = qwiklabs-gcp-44776a13dea667a6

Full documentation of **gcloud** is available on [Google Cloud gcloud Overview](https://cloud.google.com/sdk/gcloud).

## 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

## Create a Cloud SQL instance

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 granted with an assessment score.

Create a Cloud SQL instance.

Check my progress

Set a root password for the Cloud SQL instance:

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

--password Passw0rd

Copied!

content\_copy

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

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

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

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

Copied!

content\_copy

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 granted with an assessment score.

Whitelist the Cloud Shell instance to access your SQL instance.

Check my progress

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

MYSQLIP=$(gcloud sql instances describe \

taxi --format="value(ipAddresses.ipAddress)")

Copied!

content\_copy

Check the variable MYSQLIP:

echo $MYSQLIP

Copied!

content\_copy

you should get an IP address as an output.

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

mysql --host=$MYSQLIP --user=root \

--password --verbose

Copied!

content\_copy

When prompted for a password enter Passw0rd. 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 granted with an assessment score.

Create a bts database and trips table.

Check my progress

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

describe trips;

Copied!

content\_copy

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.

Exit the mysql interactive console:

exit

Copied!

content\_copy

## 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).

Run the following in the command line:

gsutil cp gs://cloud-training/OCBL013/nyc\_tlc\_yellow\_trips\_2018\_subset\_1.csv trips.csv-1

gsutil cp gs://cloud-training/OCBL013/nyc\_tlc\_yellow\_trips\_2018\_subset\_2.csv trips.csv-2

Copied!

content\_copy

Import the CSV file data into Cloud SQL using mysql:

mysqlimport --local --host=$MYSQLIP --user=root --password \

--ignore-lines=1 --fields-terminated-by=',' bts trips.csv-\*

Copied!

content\_copy

When prompted for a password enter Passw0rd.

Connect to the mysql interactive console:

mysql --host=$MYSQLIP --user=root --password

Copied!

content\_copy

When prompted for a password enter Passw0rd.

## 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.

In the mysql interactive console select the database:

use bts;

Copied!

content\_copy

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. 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. 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? 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.

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](https://www1.nyc.gov/assets/tlc/downloads/pdf/data_dictionary_trip_records_yellow.pdf), 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.

Exit the 'mysql' interactive console:

exit