# Module 2 - LAB: Loading Taxi Data into Google Cloud SQL

**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



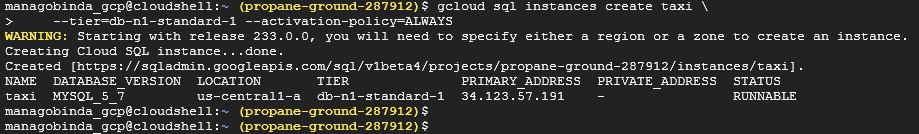
## Create a Cloud SQL instance

Enter the following commands to create a Cloud SQL instance:

gcloud sql instances create taxi \

--tier=db-n1-standard-1 --activation-policy=ALWAYS

This will take a few minutes to complete.



Set a root password for the Cloud SQL instance:

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

--password Passw0rd



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:

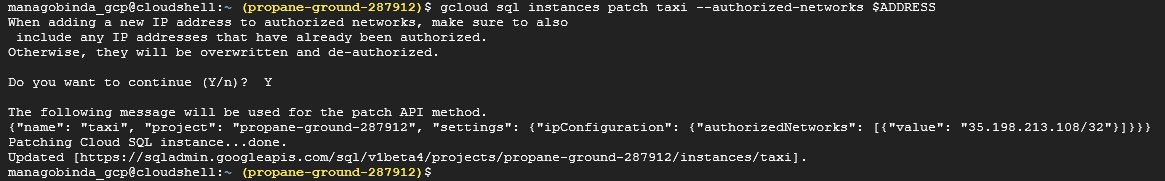
export ADDRESS=$(wget -qO - <http://ipecho.net/plain)/32>



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

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

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



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

MYSQLIP=$(gcloud sql instances describe \

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



Check the variable MYSQLIP:

echo $MYSQLIP

you should get an IP address as an output.

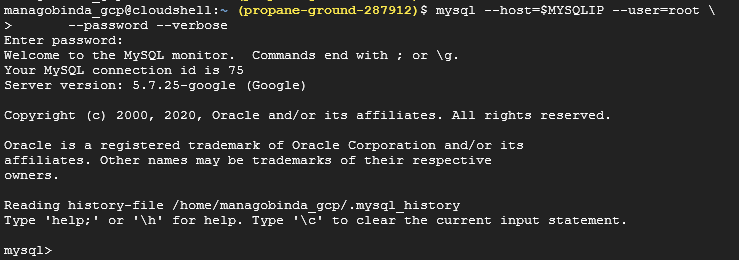


Create the taxi trips table by logging into the mysql command line interface.

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

--password –verbose

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,

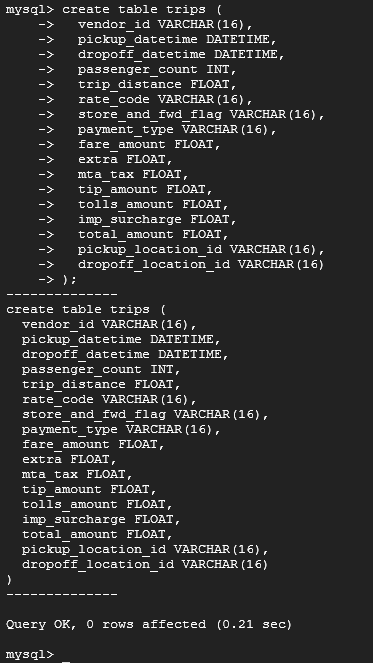
imp\_surcharge FLOAT,

total\_amount FLOAT,

pickup\_location\_id VARCHAR(16),

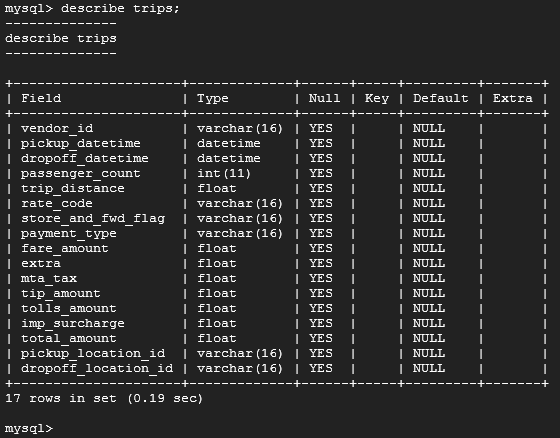
dropoff\_location\_id VARCHAR(16)

);



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

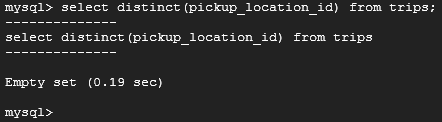
describe trips;



Query the trips table:

select distinct(pickup\_location\_id) from trips;

This will return an empty set as there is no data in the database yet.



Exit the mysql interactive console:

Exit



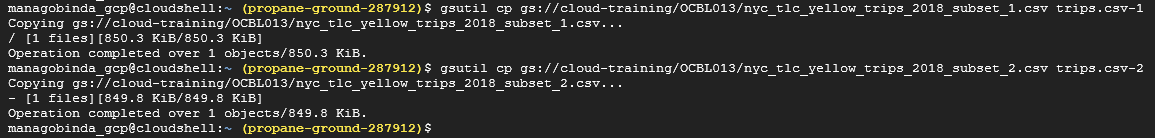
## Add data to Cloud SQL instance

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

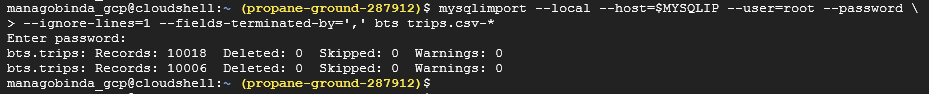


Import the CSV file data into Cloud SQL using mysql:

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

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

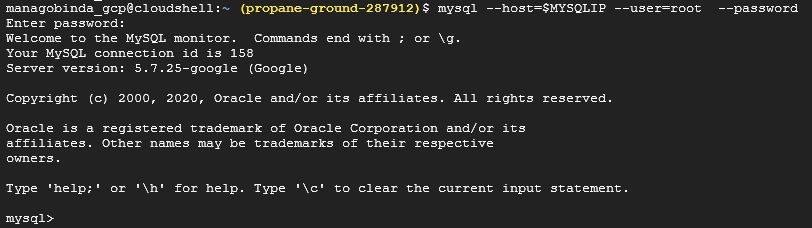
When prompted for a password enter Passw0rd.



Connect to the mysql interactive console:

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

When prompted for a password enter Passw0rd.



## Checking for data integrity

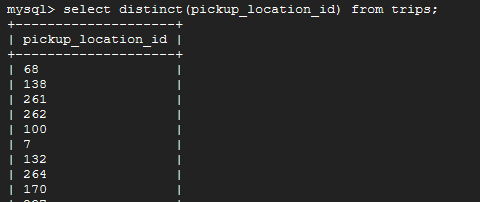
Whenever data is imported from a source it's always important to check for data integrity. Roughly, this means making sure the data meets your expectations.

In the mysql interactive console select the database:

use bts;

Query the trips table for unique pickup location regions:

select distinct(pickup\_location\_id) from trips;





This should return 159 unique ids. Let's start by digging into the trip\_distance column. Enter the following query into the console:

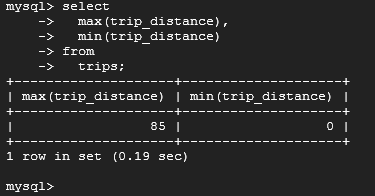
select

max(trip\_distance),

min(trip\_distance)

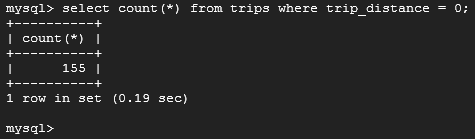
from

trips;



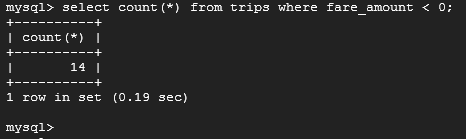
One would expect the trip distance to be greater than 0 and less than, say 1000 miles. The maximum trip distance returned of 85 miles seems reasonable but the minimum trip distance of 0 seems buggy. How many trips in the dataset have a trip distance of 0?

select count(\*) from trips where trip\_distance = 0;



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;



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;

