

# Lab5-Spark ML

## Introduction

In order to provide recommendations, whether in real time while customers browse or through email later on, several things need to happen. At first, while you know little about your users' tastes and preferences, you might base recommendations on item attributes alone. But your system needs to be able to learn from your users, collecting data about their tastes and preferences. Over time and with enough data, you can use machine learning algorithms to perform useful analysis and deliver meaningful recommendations. Other users' inputs can also improve the results, enabling the system to be retrained periodically. This solution deals with a recommendations system that already has enough data to benefit from machine learning algorithms.

A recommendation engine typically processes data through the following four phases:



## What you need

To complete this lab, you need:

- A project created on Google Cloud Platform
- A Cloud Storage bucket created

## What you learn

In this lab, you:

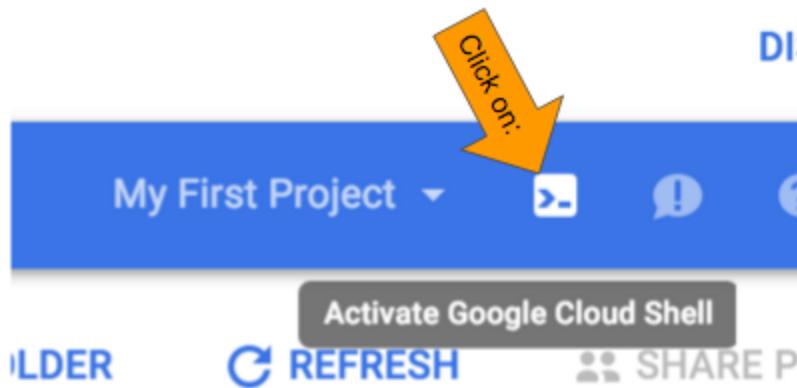
- Create Cloud SQL instance
- Create database tables by importing .sql files from Cloud Storage
- Populate the tables by importing .csv files from Cloud Storage
- Allow access to Cloud SQL
- Explore the rentals data using SQL statements from CloudShell

## Explore lab data in Cloud Shell

To explore the labs code in Cloud Shell:

## Step 1

Start Cloud Shell if you have not already done so. On GCP console, click on the **Activate Google Cloud Shell** button:



## Step 2

On the command-line, type:

```
git clone https://github.com/GoogleCloudPlatform/training-data-analyst
```

This downloads the code from github

## Step 3

Navigate to the folder corresponding to this lab:

```
cd training-data-analyst/CPB100/lab3a
```

## Step 4

Examine the table creation file using less:

```
less cloudsql/table_creation.sql
```

The less command allows you to view the file (Press the **spacebar** to scroll down; the letter **b** to back up a page; the letter **q** to quit).

Fill out this information (the first line has been filled out for you):

Table Name	Columns
Accommodation	Id, title, location, price, rooms, rating, type

\_\_\_\_\_?  
?

\_\_\_\_\_?  
?

How do these relate to the rentals recommendation scenario? Fill the following blanks:

- When a user rates a house (giving it four stars for example), an entry is added to the \_\_\_\_\_ table.
- General information about houses, such as the number of rooms they have and their average rating is stored in the \_\_\_\_\_ table.
- The job of the recommendation engine is to fill out the \_\_\_\_\_ table for each user and house -- this is the predicted rating of that house by that user.

## Step 5

Examine the data files using head:

```
head cloudsql/*.csv
```

The head command shows you the first few lines of each file.

## Stage .sql and .csv files into Cloud Storage

Stage the table definition and data files into Cloud Storage, so that you can later import them into Cloud SQL:

### Step 1

From Cloud Shell within the lab3a directory, type:

```
gsutil cp cloudsql/* gs://<BUCKET-NAME>/sql/
```

substituting the name of the bucket that you created in Lab 2b.

### Step 2

From the GCP console, go to Storage, navigate to your bucket and verify that the .sql and .csv files now exist on Cloud Storage.

## 5. Create Cloud SQL instance

To create Cloud SQL instance:

### Step 1

From the GCP console main menu (horizontal bars), select **SQL** (in the Storage section).

### Step 2

Click on **Create Instance**

### Step 3

Choose **MySQL**, then click Next. On the next dialog, choose Cloud SQL **Second Generation**

### Step 4

Give the instance the ID **rentals**

Instance ID  
ID is permanent. Use lowercase letters and numbers only.

### Step 5

Scroll down and specify a root password. Before you forget, note down the root password (please don't do this in real-life!) in a table that looks like this.

Machine	Password/IP Address
Root password	_____? ?
Cloud Shell (client)	(Step 7)
Cloud SQL (MySQL server)	(Step 10)

### Step 6

Scroll down and click on the **Add network** button

---

## + Add network

---

### Step 7

From Cloud Shell within the lab3a directory, find your IP address by typing:

```
bash ./find_my_ip.sh
```

Note this down in the second row of a table like this:

Machine	Password/IP Address
Root password	(Step 5)
Cloud Shell (client)	_____
	?
Cloud SQL (MySQL server)	(Step 10)

### Step 8

In the Add Network box, plug your IP address (Name can be anything) and click **Done**

New network

Name (Optional)

cloudshell

Network

Use [CIDR notation](#). [↗](#)

104.198.15

The IP address you got from ./find\_my\_ip.sh goes here

Done

Cancel

**Note:** If you lose your Cloud Shell VM due to inactivity, you will have to reauthorize your new Cloud Shell VM with Cloud SQL. For your convenience, lab3a includes a script called **authorize\_cloudshell.sh** that you can run.

## Step 9

Click **Create** to create the instance. It will take a minute or so for your Cloud SQL instance to be provisioned..

## Step 10

Note down the IP address of your Cloud SQL instance (from the browser window) in the *third* row of the table you started:

Note this down in the second row of a table like this:

Machine	Password/IP Address
Root password	(Step 5)
Cloud Shell (client)	(Step 7)
Cloud SQL (MySQL server)	_____? ?

## 6. Create tables

To import table definitions from Cloud Storage:

### Step 1

Click on the hyperlink named **rentals** i.e. your Cloud SQL instance name

### Step 2

Click on **Import** (on the top menu bar)

### Step 3

Click on the **Browse** button and browse to table\_creation.sql and **Select** it

### Step 4

Click **Import**.

## Populate tables

To import CSV files from Cloud Storage:

### Step 1

From the GCP console page with the Cloud SQL instance details, click on **Import** (top menu)

### Step 2

Click on the **Browse** button, browse to accommodation.csv and **Select** it. Fill out the rest of the dialog as follows:

The Database is recommendation\_spark

The Table is **Accommodation**


Import data from Cloud Storage

Choose a Cloud Storage file to import into your Cloud SQL instance. [Learn more](#)


Cloud Storage file

☒ bucket /accommodations.csv

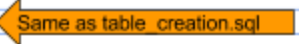
Format of import

☐ SQL ☒ CSV  Select CSV

Database

recommendation\_spark  Same as table\_creation.sql

Table

Accommodation  Same as table\_creation.sql

When you click Import, we will grant a Cloud SQL service account read access to your Cloud Storage file and the bucket that contains it. Your bucket and file permissions will reflect this access.

### Step 3

Repeat the **Import** for rating.csv, changing the table name to **Rating**.

## Explore Cloud SQL

To explore Cloud SQL, you can use the tool mysql from Cloud Shell:

### Step 1

In the Cloud Shell window, type:

```
mysql --host=<MySQLIP> --user=root --password
```

The IP address is the one for the database server (i.e. the third row in the notes). You can also find it from the instance details on the cloud console.

MySQL will prompt you for the root password. Type that into the prompt when prompted.

## Step 2

In the Cloud Shell, at the mysql prompt, type:

```
use recommendation_spark;
```

This sets the database in the mysql session.

## Step 3

View the list of tables you created. This will be helpful to prevent any typos in your query in step 4.

```
show tables;
```

## Step 4

Let's verify that the data was loaded:

```
select * from Rating;
```

Example output:

```
| 23 | 99 | 5 |
```

```
| 4  | 99 | 4 |
```

```
| 7  | 99 | 5 |
```

```
| 8  | 99 | 5 |
```

```
+-----+-----+-----+
```

1186 rows in set (0.03 sec)

## Step 5

Let's see if there is a great deal out there somewhere:

```
select * from Accommodation where type = 'castle' and price < 1500;
```

All the cheap castles are rated poorly. Oh well.

Now let's you use spark in Dataproc to train the recommendations machine learning model based on users' previous ratings. You then apply that model to create a list of recommendations for every user in the database.

In this lab, you will:

- Launch Dataproc
- Train and apply ML model written in PySpark to create product recommendations
- Explore inserted rows in Cloud SQL



## Launch Dataproc

To launch Dataproc and configure it so that each of the machines in the cluster can access Cloud SQL:

### Step 1

From the GCP console menu (three horizontal bars), select SQL and note the region of your Cloud SQL instance:

Instance ID ?	Type	IP address	Storage used ?	Storage type	Failover	Location
rentals	Second Generation	104.198.222.91	—	SSD	—	us-central1

In the snapshot above, the region is us-central1.

### Step 2

From the GCP console menu (three horizontal bars), select Dataproc and click **Create cluster**.

### Step 3

Change the zone to be in the same region as your Cloud SQL instance. This will minimize network latency between the cluster and the database. For example, I would have chosen us-central1-b (any zone in the us-central1 region).

### Step 4


Change the machine type of **both** the Master and the Worker nodes to n1-standard-2. That is sufficient for this job.

### Step 5

Click **Create**, accepting all the other defaults. It will take 1-2 minutes to provision your cluster.

### Step 6

Note the name, zone and number of workers in your cluster. It might be:

<input type="checkbox"/>	Name ^	Zone	Total worker nodes
<input type="checkbox"/>	 cluster-1	us-east1-b	2

## Step 7

In Cloud Shell, navigate to the folder corresponding to this lab and authorize all the Dataproc nodes to be able to access your Cloud SQL instance:

```
cd ~/training-data-analyst/CPB100/lab3b
```

```
bash authorize_dataproc.sh cluster-1 <Zone> 2
```

Change the cluster-name, zone or number of workers as necessary.

## Run ML model

To create a trained model and apply it to all the users in the system:

### Step 1

Edit the model training file using nano:

```
nano sparkml/train_and_apply.py
```

Change the fields marked CHANGE at the top of the file (scroll down using the down arrow key) to match your Cloud SQL setup, and save the file using Ctrl+X

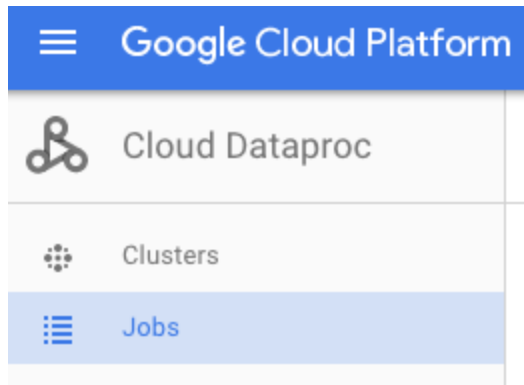
### Step 2

Copy this file to your Cloud Storage bucket using:

```
gsutil cp sparkml/tr*.py gs://<bucket-name>/
```

### Step 3

On the left-hand menu of the Dataproc section, click on **Jobs**



## Step 4

Click on **Submit job**, change the Job type to **PySpark**, and specify the location of the Python file you uploaded to your bucket.

Cluster  
cluster-1

Job type  
PySpark

Main python file  
gs://<bucket-name>/train\_and\_apply.py

gs://<bucket-name>/train\_and\_apply.py

## Step 5

Click **Submit** and wait for the job Status to change from Running (**this will take up to 5 minutes**) to Succeeded

Elapsed time	Status
22 sec	Running

If the job Failed, please troubleshoot using the logs and fix the errors. You may need to re-upload the changed Python file to Cloud Storage and clone the failed job to resubmit.

## Explore inserted rows

To view the new rows in the table, you can use the tool mysql from Cloud Shell:

## Step 1

In Cloud Shell, authorize your CloudShell VM to access the Cloud SQL instance. This will also deauthorize the Dataproc cluster.

```
bash ../lab3a/authorize_cloudshell.sh
```

## Step 2

Connect to your Cloud SQL instance (make sure to replace the MySQL IP address):

```
mysql --host=<MySQLIP> --user=root --password
```

When prompted, enter the root password.

## Step 3

At the mysql prompt, type:

```
use recommendation_spark;
```

This sets the database in the mysql session.

## Step 4

Find the recommendations for some user:

```
select r.userid,  
       r.accoid,  
       r.prediction,  
       a.title,  
       a.location,  
       a.price,  
       a.rooms,  
       a.rating,  
       a.type  
from Recommendation as r,  
     Accommodation as a  
where r.accoid = a.id and  
       r.userid = 10;
```

These are the five accommodations that we would recommend to her. Note that the quality of the recommendations are not great because our dataset was so small (note that the predicted ratings are not very high). Still, this codelab illustrates the process you'd go through to create product recommendations.

## Clean up

We will not use CloudSQL and Dataproc any more in this course, so clean up so as to avoid wasting those computational resources:

## Step 1

In GCP console, navigate to Cloud SQL, click on the hyperlink corresponding to the rentals instance and click on Delete in the top menu bar.

## Step 2

In GCP console, navigate to Dataproc, click on the checkbox corresponding to cluster-1 and click on Delete in the top menu bar.

---

NB:

In this exercise you make use of the ALS algorithm is based on three different sets of data:

- **Training set:** Contains data with known output. This set is what a perfect result would look like. In this solution, it contains the user ratings.
- **Validating set:** Contains data that will help tune the training to pick the right combination of parameters and choose the best model.
- **Testing set:** Contains data that will be used to evaluate the performance of the best trained model. This would be equivalent to running the analysis in a real-world example.

And in the code you can see the call:

```
# train the model
model = ALS.train(dfRates.rdd, 20, 20) # you could tune these numbers, but these are reasonable choices
print("trained ...")
```

If you want to know more about this algorithm, please have a look [here](#), but just take into account that this is beyond the current course.