

# Movie Recommendations in BigQuery ML

1 hour 7 Credits

[Rate Lab](#)

## Overview

**BigQuery is Google's fully managed, NoOps, low cost analytics database. With BigQuery you can query terabytes and terabytes of data without having any infrastructure to manage or needing a database administrator. BigQuery uses SQL and can take advantage of the pay-as-you-go model. BigQuery allows you to focus on analyzing data to find meaningful insights.**

**BigQuery Machine Learning** (BQML, product in beta) is a new feature in BigQuery where data analysts can create, train, evaluate, and predict with machine learning models with minimal coding.

Collaborative filtering provides a way to generate product recommendations for users, or user targeting for products. The starting point is a table with three columns: a user id, an item id, and the rating that the user gave the product. This table can be sparse -- users don't have to rate all products. Then, based on just the ratings, the technique finds similar users and similar products and determines the rating that a user would give an unseen product. Then, we can recommend the products with the highest predicted ratings to users, or target products at users with the highest predicted ratings.

To illustrate recommender systems in action, we'll use the MovieLens dataset. This is a dataset of movie reviews released by **GroupLens**, a research lab in the Department of Computer Science and Engineering at the University of Minnesota, through funding by the US National Science Foundation.

## Objectives

In this lab, you learn to perform the following tasks:

- Train a recommendation model in BigQuery
- Make product predictions for both single users and batch users

# Set up your environment

What you'll need

To complete this lab, you'll need:

- Access to a standard internet browser (Chrome browser recommended).
- Time. Note the lab's Completion time in Qwiklabs. This is an estimate of the time it should take to complete all steps. Plan your schedule so you have time to complete the lab. Once you start the lab, you will not be able to pause and return later (you begin at step 1 every time you start a lab).
- The lab's Access time is how long your lab resources will be available. If you finish your lab with access time still available, you will be able to explore the Google Cloud Platform or work on any section of the lab that was marked "if you have time". Once the Access time runs out, your lab will end and all resources will terminate.
- You DO NOT need a Google Cloud Platform account or project. An account, project and associated resources are provided to you as part of this lab.
- If you already have your own GCP account, make sure you do not use it for this lab.

- **If your lab prompts you to log into the console, use only the student account provided to you by the lab. This prevents you from incurring charges for lab activities in your personal GCP account.**

### **Start your lab**

**When you are ready, click Start Lab. You can track your lab's progress with the status bar at the top of your screen.**

**Important** What is happening during this time? Your lab is spinning up GCP resources for you behind the scenes, including an account, a project, resources within the project, and permission for you to control the resources needed to run the lab. This means that instead of spending time manually setting up a project and building resources from scratch as part of your lab, you can begin learning more quickly.

### **Find Your Lab's GCP Username and Password**

**To access the resources and console for this lab, locate the Connection**

**Details panel in Qwiklabs. Here you will find the account ID and password for the account you will use to log in to the Google Cloud Platform:**

## Open Google Console

**Caution:** When you are in the console, do not deviate from the lab instructions. Doing so may cause your account to be blocked. [Learn more.](#)

Username

google2876526\_student@qwiklabs.n



Password

TG959yrKDX



GCP Project ID

qwiklabs-gcp-0855e773352d3560



[New to labs? View our introductory video!](#)

If your lab provides other resource identifiers or connection-related information, it will appear on this panel as well.

**Log in to Google Cloud Console**

Using the Qwiklabs browser tab/window or the separate browser you are using for the Qwiklabs session, copy the Username from the Connection Details panel and click the Open Google Console button.

**You'll be asked to Choose an account. Click Use another account.**



## Choose an account



gcpstaging10382\_student@qwiklabs.net  
*Signed out*



gcpstaging10408\_student@qwiklabs.net  
*Signed out*



Use another account

**Paste in the Username, and then the Password as prompted:**



## Sign in

to continue to Google Cloud Platform

Enter your email

gcpstaging277-student@qwiklabs.net

[More options](#)

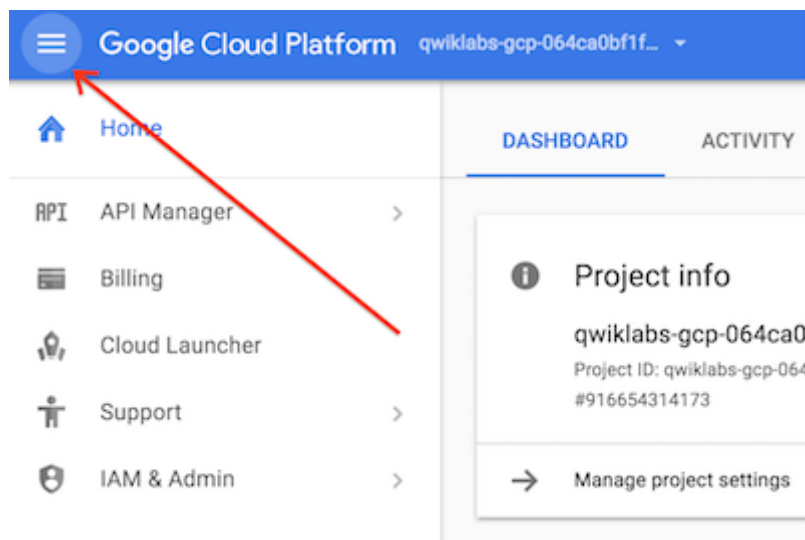
NEXT

**Accept the terms and conditions.**

Since this is a temporary account, which you will only have to access for this one lab:

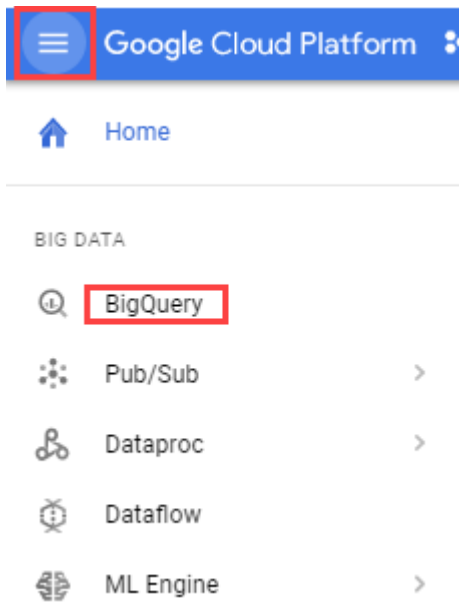
- Do not add recovery options
- Do not sign up for free trials

Note: You can view the list of services by clicking the GCP Navigation menu button at the top-left next to “Google Cloud Platform”.



## Open BigQuery Console

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 lists UI updates.

Click Done.


## Task 1: Get MovieLens Data

First, we're going to use the command line to create a BigQuery dataset to store the MovieLens data. The MovieLens data will then be downloaded and placed in the dataset.



## Start the Cloud Shell Editor

To create a BigQuery dataset and download the MovieLens data the Cloud Shell Editor is used.

1. In the GCP Console, click Activate Cloud Shell (  ).
2. If prompted, click Start Cloud Shell.

## Create a BigQuery Dataset

1. Run the following command to create a BigQuery dataset named `movies`:

```
bq --location=EU mk --dataset movies
```

## Download Data with `curl` and load to Dataset

1. Run the following command in Cloud Shell Editor:

```
curl -O 'http://files.grouplens.org/datasets/movielens/ml-20m.zip'
```

```
unzip ml-20m.zip
```

```
bq --location=EU load --source_format=CSV \
```

```
--autodetect movies.movielens_ratings ml-20m/ratings.csv
```

```
bq --location=EU load --source_format=CSV \
```

```
--autodetect movies.movielens_movies_raw ml-20m/movies.csv
```

Click *Check my progress* to verify the objective.

Get MovieLens Data

Check my progress

## Task 2: Explore the Data

1. In BigQuery's Query editor execute the following query:

```
SELECT
```

```
COUNT(DISTINCT userId) numUsers,
```

```
COUNT(DISTINCT movieId) numMovies,
```

```
COUNT(*) totalRatings
```

```
FROM
```

```
movies.movielens_ratings
```

You should confirm that the dataset consists of over 138 thousand users, nearly 27 thousand movies, and a little more than 20 million ratings.

2. Examine the first few movies using the query:

```
SELECT
```

```
*
```

```
FROM
```

```
movies.movielens_movies_raw
```

```
WHERE
```

```
movieId < 5
```

Row	movieId	title	genres
1	3	Grumpier Old Men (1995)	Comedy Romance
2	4	Waiting to Exhale (1995)	Comedy Drama Romance
3	2	Jumanji (1995)	Adventure Children Fantasy
4	1	Toy Story (1995)	Adventure Animation Children Comedy Fantasy

3. We can see that the genres column is a formatted string. Parse the genres into an array and rewrite the results into a table named `movielens_movies`.

```
CREATE OR REPLACE TABLE
```

```
movies.movielens_movies AS
```

```
SELECT
```

```
* REPLACE(SPLIT(genres, "|") AS genres)
```

```
FROM
```

```
movies.movielens_movies_raw
```

Feel free to perform additional queries until you are comfortable with the dataset.

Click *Check my progress* to verify the objective.

Explore the Data

Check my progress

## Task 3: Collaborative Filtering

Matrix factorization is a collaborative filtering technique that relies on two vectors called the user factors and the item factors. The user factors is a

low-dimensional representation of a `user_id` and the item factors similarly represents an `item_id`.

To perform a matrix factorization of our data we use the typical BigQuery ML syntax except that the `model_type` is `matrix_factorization` and that we have to identify which columns play what roles in the collaborative filtering setup.

1. Apply matrix factorization to the movie ratings data by executing the following query:

```
CREATE OR REPLACE MODEL
```

```
  movies.movie_recommender
```

```
OPTIONS
```



```
(model_type='matrix_factorization',
```

```
user_col='userId',
```

```
item_col='movieId',
```

```
rating_col='rating',
```

```
l2_reg=0.2,
```

```
num_factors=16) AS
```

```
SELECT
```

```
userId,
```

```
movieId,
```

```
rating
```

```
FROM
```

```
movies.movielens_ratings
```

Note, the `num_factors` and `l2_reg` options have been selected after much experimentation to speed up training of the model. Still, the model will take some 40 minutes to train. Fortunately, a fully trained model has been saved that you can access in the `cloud-training-prod-bucket:movies` public dataset.

2. Stop the query by clicking on Stop.
3. To view metrics for the trained model enter the following query:

```
SELECT * FROM ML.EVALUATE(MODEL  
`cloud-training-prod-bucket.movies.movie_recommender`)
```

## Task 4: Making Recommendations

With the trained model, we can now provide recommendations.

1. Let's find the best comedy movies to recommend to the user whose `userId` is 903. Enter the query below:

SELECT

\*

FROM

ML.PREDICT(MODEL `cloud-training-prod-bucket.movies.movie\_recommender`,

(

```
SELECT
```

```
movieId,
```

```
title,
```

```
903 AS userId
```

```
FROM
```

```
`movies.movielens_movies`,
```

```
UNNEST(genres) g
```

```
WHERE
```

```
g = 'Comedy' ))
```

```
ORDER BY
```

**predicted\_rating DESC**

**LIMIT**

**5**

Row	predicted_rating	movieId	title	userId
1	6.305484877897655	82978	Neighbors (1920)	903
2	5.659955887029915	26136	Hallelujah Trail, The (1965)	903
3	5.608127858593018	69075	Trojan War (1997)	903
4	5.423441457257417	3337	I'll Never Forget What's'isname (1967)	903
5	5.301408212165985	6167	Stand-In (1937)	903

2. This result includes movies the user has already seen and rated in the past. Let's remove them:

**SELECT**

\*

FROM

```
ML.PREDICT(MODEL `cloud-training-prod-bucket.movies.movie_recommender`,
```

(

WITH



```
seen AS (
```

```
SELECT
```

```
ARRAY_AGG(movieId) AS movies
```

```
FROM
```

```
movies.movielens_ratings
```

```
WHERE
```

```
userId = 903 )
```

```
SELECT
```

```
movieId,
```

```
title,
```

```
903 AS userId
```

```
FROM
```

```
movies.movielens_movies,
```

```
UNNEST(genres) g,
```

```
seen
```

```
WHERE
```

```
g = 'Comedy'
```

```
AND movieId NOT IN UNNEST(seen.movies) ))
```

```
ORDER BY
```

```
predicted_rating DESC
```

**LIMIT**

**5**

For this user, this happens to yield the same set of movies -- the top predicted ratings didn't include any of the movies the user has already seen.

Click *Check my progress* to verify the objective.

Making Recommendations

Check my progress

## Task 5: Customer Targeting

In the previous section, we looked at how to identify the top-rated movies for a specific user. Sometimes, we have a product and have to find the customers who are likely to appreciate it.

1. We wish to get more reviews for `movieId=96481` which has only one rating and we wish to send coupons to the 100 users who are likely to rate it the highest. Identify those users using:

SELECT

\*

FROM

ML.PREDICT(MODEL `cloud-training-prod-bucket.movies.movie\_recommender`,

(

WITH

allUsers AS (

SELECT

DISTINCT userId

FROM

```
movies.movielens_ratings )
```

```
SELECT
```

```
96481 AS movieId,
```

```
(
```

```
SELECT
```



```
title
```

```
FROM
```

```
movies.movielens_movies
```

```
WHERE
```

```
movieId=96481) title,
```

```
userId
```

```
FROM
```

```
allUsers ))
```

```
ORDER BY
```

```
predicted_rating DESC
```

**LIMIT**

**100**

The result gives us 100 users to target, the top 5 of whom are:

Row	predicted_rating	movieId	title	userId
1	6.000193988615432	96481	American Mullet (2001)	104104
2	5.92811262777923	96481	American Mullet (2001)	57703
3	5.902559169949699	96481	American Mullet (2001)	22625
4	5.882101585633906	96481	American Mullet (2001)	118093
5	5.740621111206273	96481	American Mullet (2001)	37594

Click *Check my progress* to verify the objective.

Customer Targeting

Check my progress

## Task 6: Batch predictions for all users and movies

What if we wish to carry out predictions for every user and movie combination? Instead of having to pull distinct users and movies as in the previous query, a convenience function is provided to carry out batch predictions for all `movieId` and `userId` encountered during training.

1. Enter the following query to obtain batch predictions:

```
SELECT
```

```
*
```

```
FROM
```

```
ML.RECOMMEND(MODEL `cloud-training-prod-bucket.movies.movie_recommender`)
```

In this case, the result will be too large to return given the default settings. Impose a `LIMIT` to see a few output results.

As seen in a section above, it is possible to filter out movies the user has already seen and rated in the past. The reason already seen movies aren't filtered out by default is that there are situations (think of restaurant recommendations, for example) where it is perfectly expected that we would need to recommend restaurants the user has liked in the past.

## End your lab