

Assignment 1

PSTAT 135/235

Name: **Zijian Wan**_

Perm Number: **4422853**_

MovieLens Dataset

In this assignment, we will be working on a new dataset. To download it paste the following URL into your laptop's browser: <http://files.grouplens.org/datasets/movielens/ml-latest.zip>. Alternatively, you can also go to <https://grouplens.org/datasets/movielens/> and download [ml-latest.zip](#).

This dataset has around 27 million ratings on about 58,000 movies done by over 280,000 users and last updated on 9/2018. Unzip this 288 MB file. For the purpose of this assignment we will be using only two of the files that are included:

1. [movies.csv](#) (2.9 MB)
2. [ratings.csv](#) (760 MB).

Question 1: Uploading Data to BigQuery

Upload these two files into a dataset in BigQuery and call it [movie_ratings](#).


Create a new dataset and call it [movie_ratings](#). We will load these two files into the newly created dataset two ways: using the web interface and again using cloud shell.

Question 1a: [movies](#) table

To create [movies](#) table from [movies.csv](#) file,

1. Download the zipped file
2. Unzip the archive
3. In your BigQuery interface, select in the resources list `<YOUR-PROJECT-ID>` > [movie_ratings](#) > click **"CREATE TABLE"** button
4. [Create table from: Upload](#)
[Select file:](#) BROWSE and find [movies.csv](#) from your computer
[Table:](#) [movies](#)
[Schema](#) [Auto detect](#): check

Find your LOAD job information from [PROJECT HISTORY](#) (next to [PERSONAL HISTORY](#)) at the bottom. Mine looks like @fig-job-info

load-job-info{#fig-job-info}

Post screenshot of your LOAD job information here:

Load job details

| | |
|-----------------------|--|
| Job ID | pstat235-zw:US.bquxjob_613aede1_185eb68d8c2 |
| User | zijianwan@ucsb.edu |
| Location | US |
| Creation time | Jan 25, 2023, 4:09:57 PM UTC-8 |
| Start time | Jan 25, 2023, 4:09:57 PM UTC-8 |
| End time | |
| Duration | 2 sec |
| Auto-detect schema | true |
| Ignore unknown values | false |
| Source format | CSV |
| Max bad records | 0 |
| Destination table | pstat235-zw.movie_ratings.movies |

[REPEAT LOAD JOB](#)[CLOSE](#)

{#fig-movies-job-info}

Question 1b: **ratings** table

Follow the same procedure as Question 1a to crate **ratings** table from **ratings.csv**. What happens?

Local uploads are limited to 100 MB. We are advised to use Google Cloud Storage for larger files.

PSTAT 135 Students: Upload **ratings.csv** file to Cloud Storage and create **ratings** table from it using the web interface. Then, post the screenshot of your LOAD job information here:

Replace this text with your screenshot image

PSTAT 235 Students: Upload **ratings.csv** file to Cloud Storage and create **ratings** table using the command line tools: **bq** and **gsutil**.

1. Verify the location of **ratings.csv** file using Cloud Storage command:

```
gsutil ls gs://<YOUR-BUCKET-NAME>
```

Note your the path to your **ratings.csv** file (referred to as **<RATINGS-FILE-LOCATION>** below).

2. Create an empty table with `bq`. Read the documentation, `bq mk --help` to fill-in the blanks in the code below:

```
bq mk _____
```

3. Using `bq` command to load `movie_ratings.ratings` table with contents from `<RATINGS-FILE-LOCATION>`. Read the documentation, `bq load --help` to fill-in the blanks in the code below:

```
bq load --autodetect _____
```

Replace the section below with your own commands:

```
gsutil ls gs://pstat235-zw  
bq mk movie_ratings.ratings  
bq load --autodetect movie_ratings.ratings gs://pstat235-zw/ratings.csv
```

Also, post screenshot of your LOAD job information here:

Load job details

| | |
|-----------------------|---|
| Job ID | pstat235-zw:US.bqjob_r1043fd657450a053_00000185eb7993e4_1 |
| User | zijianwan@ucsb.edu |
| Location | US |
| Creation time | Jan 25, 2023, 4:28:11 PM UTC-8 |
| Start time | Jan 25, 2023, 4:28:12 PM UTC-8 |
| End time | Jan 25, 2023, 4:28:51 PM UTC-8 |
| Duration | 39 sec |
| Auto-detect schema | true |
| Ignore unknown values | |
| Source format | |
| Max bad records | 0 |
| Destination table | pstat235-zw.movie_ratings.ratings |

REPEAT LOAD JOB

CLOSE

{#fig-ratings-job-info}

Question 2: ratings table number of rows

How many rows are there in ratings table? C

- A. 27753445
- B. 27000001
- C. 27753444
- D. 27000000

Question 3: movies table number of rows

How many rows are there in the movies table? D

- A. 57999
- B. 58000
- C. 58097
- D. 58098

Question 3: number of unique movies

How many unique `movieId`'s are in `ratings` table? **C**

- A. 52019
- B. Around 27 million
- C. 53889**
- D. 58097

What is your SQL code to obtain the info?

```
SELECT COUNT(DISTINCT movieId) FROM pstat235-zw.movie_ratings.ratings
```

Question 4: highly rated movies

Which one of these movies are among top 10 highly rated movies, with at least 10,000 reviews? (select all that apply) **C**

- A. Star Wars: Episode IV - A New Hope (1977)
- B. Chinatown (1974)
- C. Godfather**
- D. Casablanca (1942)

What is your SQL code to obtain the info?

```
SELECT title, COUNT(*) AS review_count, AVG(rating) AS avg_rating FROM pstat235-zw.movie_ratings.ratings INNER JOIN pstat235-zw.movie_ratings.movies ON pstat235-zw.movie_ratings.movies.movieId = pstat235-zw.movie_ratings.ratings.movieId GROUP BY title HAVING review_count > 10000 ORDER BY avg_rating DESC LIMIT 10;
```

Question 5: most watched movies

Which movie is the most watched? Make an assumption that number of ratings is strongly correlated with number of people watching it. **A**

- A. Shawshank Redemption**
- B. Forrest Gump (1994)
- C. Matrix
- D. Toy Story (1995)

What is your SQL code to obtain the info?

To find the most watched movie,

```
SELECT title, COUNT(userId) AS user_count, AVG(rating) AS avg_rating FROM pstat235-zw.movie_ratings.ratings INNER JOIN pstat235-zw.movie_ratings.movies ON pstat235-zw.movie_ratings.movies.movieId = pstat235-zw.movie_ratings.ratings.movieId GROUP BY title ORDER BY user_count DESC LIMIT 10;
```

To examine the correlation between the number of ratings and the number of people watching,

```
SELECT CORR(rating_count, user_count) AS corr FROM ( SELECT title, COUNT(*) AS rating_count, COUNT(userId) AS user_count FROM pstat235-zw.movie_ratings.ratings INNER JOIN pstat235-
```

```
zw.movie_ratings.movies ON pstat235-zw.movie_ratings.movies.movieid = pstat235-  
zw.movie_ratings.ratings.movieid GROUP BY title );
```

The correlation is 1. The number of ratings is strongly correlated with the number of people watching it.