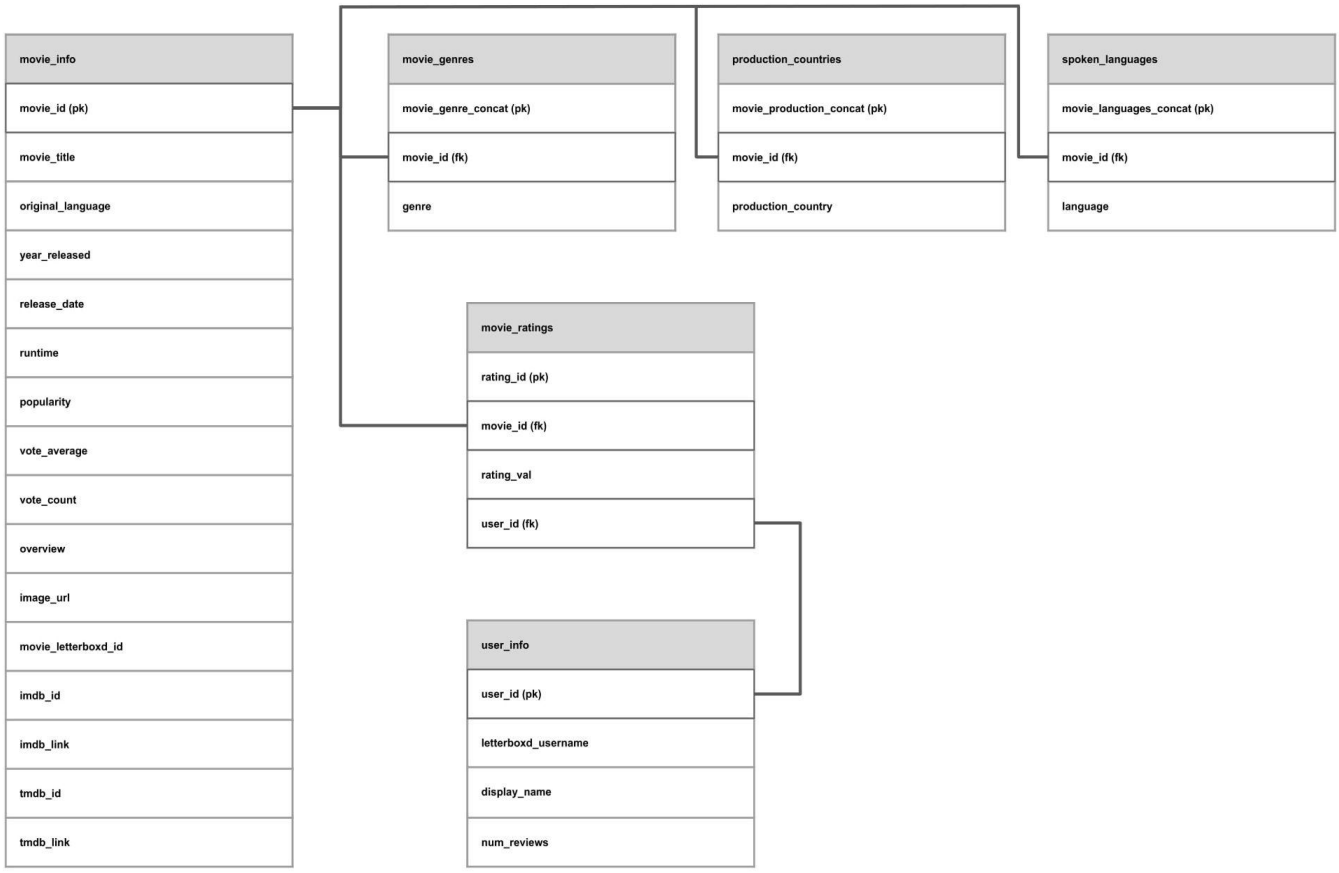
Rachel Lee

4/7/2022

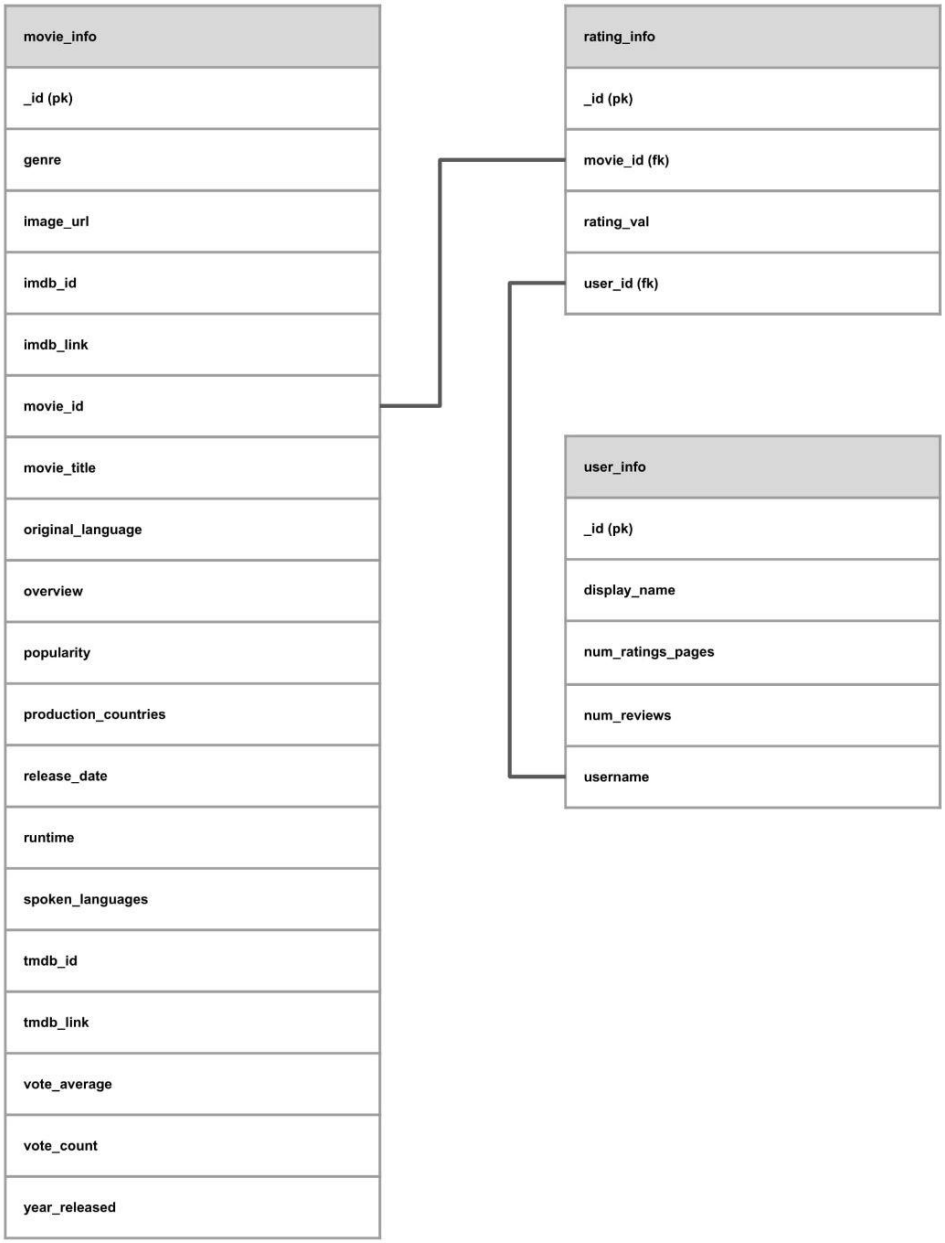
CMSC B383 Databases

Homework 4: Putting Data

1. PSQL Schema Diagram



2. Mongo Schema Diagram



3. Details

To bring my data into PostGreSQL, I first used a SQL file (letterboxd_DDL.sql) to create database tables in my “rlee” database. I then used python (CSVtoPSQL.py) to parse through the CSV files, clean up the data, and populate the database tables.

Overall, I have 6 tables in PSQL: “movie_info”, “movie_genres”, “production_countries”, “spoken_languages”, “user_info”, and “movie_ratings”.

The “movie_info” table has the bulk of the movie data, containing almost all of the columns that were originally in the “movie_export.csv” file (found on this site). This table includes information like movie title, year released, and IMDB link. The “movie_genres”, “production_countries”, and “spoken_languages” tables contain the rest of the data from that CSV file. I chose to separate these columns into other tables since these columns contain arrays with multiple values (e.g. a film might be considered as both action and science-fiction genre), and PSQL is not good for array datatypes within individual database columns. These three tables have primary keys that were generated from string concatenations: “movie_genre_concat”, “movie_production_concat”, and “movie_languages_concat” respectively. The “movie_info” table uses an integer as a primary key, “movie_id”, which was generated as indices. The “movie_genres”, “production_countries”, and “spoken_languages” tables all have a “movie_id” column that acts as a foreign key referencing the “movie_info” table.

The “user_info” table contains all of the relevant information from the “user_export.csv” file, such as their Letterboxd site username and the number of film reviews they have written. I excluded one column as it wasn’t super important, and was very often null so my SQL code did not handle putting those rows into the PSQL database well. The table uses an integer primary key, “user_id”, which was generated as indices.

The “movie_ratings” table contains all of the information from the “ratings_export.csv” file, such as the movie being rated and the rating value. No columns were excluded. The table uses an integer primary key, “rating_id”, which was generated as indices. This table has a foreign key, “movie_id”, which references the primary key in the “movie_info” table, also named “movie_id”. This table also has a foreign key,

“user_id”, which references the primary key in the “user_info” table, also named “user_id”. Referencing the unique integer identifiers from those other tables made the most sense and likely will not lead to any conflict issues.

To bring data into my Mongo collections, I used a python file (CSVtoMongo.py) to create a database, create collections, and then populate those collections with documents parsed from the CSV files. The database is named “rlee”, and the collections are “movie_info”, “user_info”, and “rating_info”. I only have 3 collections in Mongo versus 6 tables in PostGreSQL because Mongo handles array datatypes much easier. The “movie_info” collection in my Mongo database contains the equivalent of all the information in the “movie_info”, “movie_genres”, “production_countries”, and “spoken_languages” tables in my PSQL database. The “user_info” collection contains all the information from the “user_export.csv” file, similar to the PSQL database. The “rating_info” collection contains all the information from the “rating_export.csv” file, also similar to the PSQL database. The “rating_info” collection has 2 foreign keys referencing the “movie_id” field in the “movie_info” collection and the “username” field in the “user_info” collection. Having all the information in fewer collections than PSQL resulted in fewer total interaction points, as you can see in the Mongo schema diagram as compared to the PSQL schema diagram.

Part 2 Query Descriptions

1. Find the first 10 users' display names titles according to indices. The index is used to find what correlates to the first 10 entries in the original CSV.
2. Find the number of 1996 documentary films.
3. Find 25 ratings by the user with the Letterboxd username 'deathproof'. For each of the ratings, display the Letterboxd unique movie id and the user's rating.