Ananya Vittal and Kathleen Gendotti

CPSC 408

22 May 2020

Final Report

Problem:

During this global pandemic, people all over the world are stuck inside with limited options of what they can do. Many people have turned to rely on online streaming services such as Netflix, Disney+, and Hulu to make use of their time inside. A common problem with these services is that there are so many entertainment options available, that it can be hard to choose what to watch. Technologies such as the Netflix Party Google Chrome extension also make it easy to watch Netflix remotely with multiple friends. However, it can be difficult to choose a movie or TV show that multiple people can agree on watching. With our project, we have combined data from multiple streaming platforms such as Netflix and Disney+ so that users can search through a larger database of entertainment content in order to decide what to watch, as well as insert their own movie data and user ratings to keep track of their watch history.

Related applications/work:

The related applications of our project basically already exist in streaming platforms. For example, streaming platforms contain databases of movies and TV shows, along with information such as actors, directors, genres, ratings, and more. Netflix algorithms analyze a user's watch history and provide recommendations for what to watch next based on factors such as actors, genre, rating, and country. Similarly, Hulu's algorithm incorporates a like and dislike system where users can give individual movies or TV shows a thumbs-up or thumbs-down

depending on whether they like the content. Hulu is more likely to display movies and shows that are similar to items where someone has previously given a thumbs-up. Disney+ also analyzes user trends to personalize recommendations and can detect whether users belong to different categories and age groups based on user interaction. For example, families with children are more likely to subscribe to Disney+ for animated and family-friendly movies, whereas other market segments may be more interested in content such as Star Wars or ESPN.

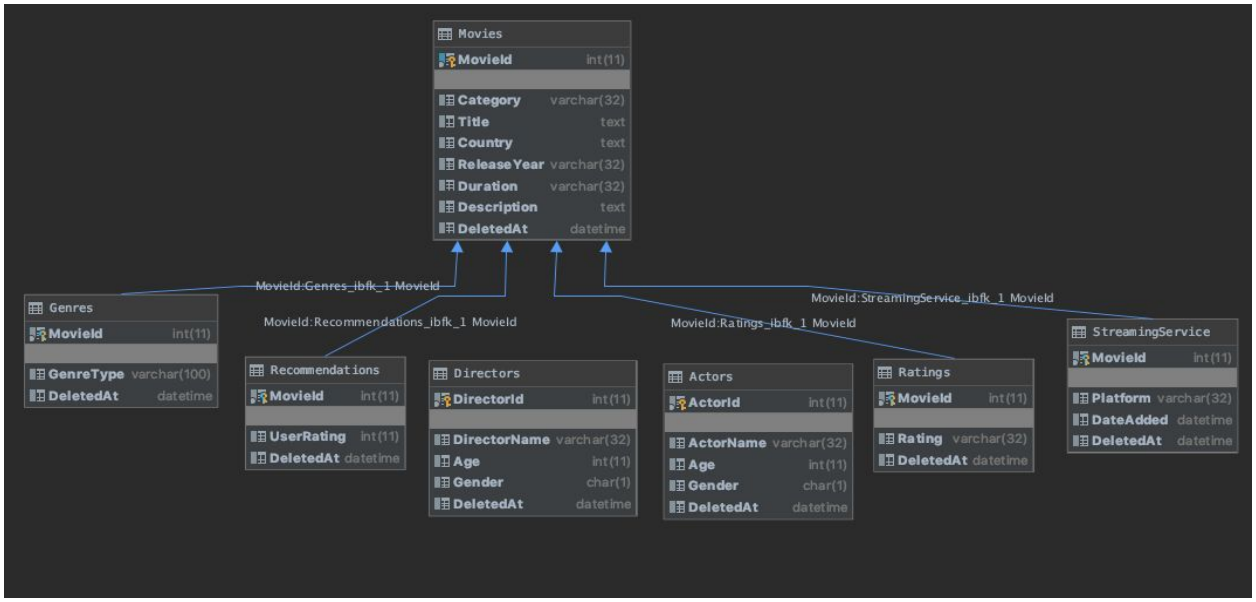
Elements of solution (i.e. framework, algorithms):

For our project, we decided to build our web application using Flask, which is a micro web framework written in Python. We wanted to choose a framework that was based in Python, because we had already been using Pycharm to connect to our MySQL and SQLite databases. At the beginning of our project, we had looked into using Django and Kivy, however Flask is much more minimal in design, has substantial documentation available, and also provided more flexibility in terms of customization. By default, Flask uses Jinja2, a modern and design-friendly template engine for Python. Jinja2 also allows for template inheritance, so that templates can be inherited from a base template in order to share content across multiple HTML files without redundancies. Working with Flask for this project was a great learning experience, because this was our first time building a front-end web application interface and connecting it to a back-end database server.

Results, schema diagrams, and functionality:

Our final project is a web application that allows a user to access a database that consists of data from streaming platforms such as Netflix and Disney+. After creating our DDL statements in DataGrip, we downloaded over 5,000 rows of data from Kaggle databases. We

then used the programming language R to clean the data, choose 250 random records from each streaming service, and combine the data. We then created a data importer tool to import the data from a flat csv file into our empty database. Our MySQL database consists of seven tables: Movies, Actors, Directors, Genres, Ratings, Recommendations, and StreamingService. We also created an index for MovieId so that queries can efficiently retrieve data from the database. MovieId is the primary key used to uniquely identify each record in the Movies table. It is also used as a foreign key so that the other tables can reference the Movies table, thereby maintaining referential integrity.



The application allows the user to login with a username and password, which brings them to the home page. The layout of the home page consists of seven tabs, as well as four buttons. Depending on the tab they click on, the user can display all records from the particular table. The user can also select a table from a drop down list which exports all the records in the selected table to a csv file. If the user would like to insert a new record, they can click on the "Insert new record" button which leads them to a page that allows them to insert new data into

the tables, which would then be committed to the database. When a new movie/TV show/episode is created, the records tied to the same MovieId will all be inserted into their individual tables. Before allowing the user to successfully insert a new movie, we checked for duplicates by using subqueries and a triple join. The user can also insert a recommendation by inputting a user rating from 1-5 for a movie/TV show/episode that they have already watched. In our code, if a user does not select which category they would like to insert new data into, the transaction would be rollbacked in order to avoid any errors. A user can also delete records by clicking on the "Delete" button next to the record they would like to delete. If a record from Movies is deleted, it will also be deleted from the other tables such as Genre, Rating, and StreamingService that are connected through MovieId. For deleting, we utilized the soft delete method which updates the DeletedAt column with the timestamp of when the record was deleted. This is hidden when the tables are displayed so the record appears to be deleted if the DeletedAt column is not null. For updating, the user can click on the "Update record" button or "Edit" button in the Movies table to update a record with new data for each attribute. The record is updated according to the matching title, and all the other tables containing MovieId will also be updated. To search the database or filter for attributes, the user can click on the "Filtering" button on the home page. This allows them to filter all the content in the database based on categories, TV ratings, countries, genres, and/or streaming services. This feature uses checkboxes so the user can choose multiple selections from each category and multiple categories at a time in order to get customized results. There is also a feature on the filtering page that allows the user to search through the database based on either a title, year, actor, director and user rating. The filtering function uses triple left joins to check for existing records in the database.

Originally we wanted to be able to take in user data and make suggestions on what they should watch next. It would be unique from how other streaming services provide recommendations to their users because we would allow multiple users to input their favorite shows and generate recommendations that everyone will enjoy. Although we did not have time to create a recommendation system for the combined interests of multiple users, we did create a button that would display suggestions for a single user. The “Suggestions” button uses group-by clauses to provide recommendations to the user based on the top TV ratings and genres associated with the content they have given the highest user ratings to.

Overall this was a really interesting project. It was great to combine all the knowledge from this semester into a web application with a MySQL backend, and we definitely learned a lot from this experience.