ajurewic, ajnorell, ikgatt
SAT 3210
Tim VanWagner
November 29, 2023

MTU Filmboard's Expanded Cinematic Universe (The Sequel) Database

Our group decided to create a database of a movie theater chain. This database would allow multiple locations in one database that holds information on movies playing, schedule, types of theaters they have, types of concessions, and employee information. A movie theater has a lot of interlocking data, what concessions are available and at what location, what locations have what employees, what employees have what training, etc. To achieve this we decided to use MariaDB and Python to create our database. This allowed us to use Python to create an insertion script to easily create our database as well as upload a CSV file with the information to the database.

The database name we decided to go with is Movies. We decided this because the full name of the company we created would be too long. This database contains 8 tables. When creating this database we faced some challenges. Learning how to code to create and modify the database was a bit of a challenge due to the limited knowledge the entire group had in programming. Another challenge was getting the database into 3rd form, to meet the requirement we found repetitive attribute columns, and once we removed those, our primary keys required multiple attributes. To combat this, we had to break a larger employee table into a separate salary and training table. We also found a completely repetitive movie table, in which we only needed one extra attribute in the movies table, instead of 2 separate tables.

Our menu to navigate through the database is written in Python. The creation of menus grew difficult to navigate as we had poor prior planning, and tried to build off of as was needed, instead of trying to anticipate all needs from the beginning. We used a lot of comments in the code in order to keep our navigation menu understandable from the back end. In the end, we have a functioning navigation menu that does leave room for improvement.

With this, our database is still not complete, it can use more work. Something to help improve the database would be to implement better security. To improve security we would need to add different users, where the different users could represent different levels in the company and thus have different permissions. For example, the manager would have access to payroll, but a regular ticket counter worker would only need access to the table with the movies.

Additionally, we could give a menu of expected values that the user would need to input. If we did this method, we would have to write code in order to only accept what the preset options are,

because otherwise through inspect element a user could manually add HTML form lines that open us up to SQL injection.

We would also want better abstraction to improve useability. While we were breaking out our tables, they started to become fairly cluttered. Specifically, the movies table that keeps track of the schedule. Through the web application and the programming that comes with it, it becomes a human-readable chart, however, the table itself is very confusing when listed in the database. Additionally, the training table is made in boolean, which to an average user would be mostly incomprehensible. In the future, we would try to clean up the movies table, and try to create a new way to store the showtimes of all the movies, and we would also change the training table to either a string or program a solution to make it so whenever a user pulls an employee's training history, the programming gives the user the string of what trainings they have instead of a binary chart.