Shrey, Satyaprakash, Justin, Joseph
DATA 225 Report
18 December 2023

TABLE OF CONTENTS

Application Introduction

Today, movies have evolved into a modern source of entertainment that captivates millions of audiences worldwide with its breathtaking visuals and unique storylines. With the rise in CGIs and technology, the need for a movie database has become just as important as well. A movie database is an organized collection that serves as an accessible resource for accessing information about movies. The database that we have created allows users to select specific movies in the database, where the user can then rate the movie(s) based on opinion. Each user will be able to save their ratings inside their account, where they will be able to register and login to the system. Finally, users will also be able to visualize important trends and meaningful insights on the movies based on charts, tables, and graphs created.

Data Sources

The application consists of two main modules: Operational Module and Analytical Module. Here are the sources that were used:

- Mockaroo - https://www.mockaroo.com/
- Kaggle - https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset
- Microsoft Excel
- ERD Plus
- MySQL Workbench
- PyQt Designer
- Python3

Summary of the Operational Module

The user can register and log in to access various modules and functions in the database. Once the user is logged in, users can search for movies inside the database and be able to retrieve in-depth information about the movie, such as title, genre, average rating, number of reviews, and the generated critique for a specific movie. Users can also contribute by selecting reviews and movies, assigning a rating with a slider, and the option to create a new critique ID. This system also enables the user to view, edit, and delete the past reviews. The system maintains a log for users, ensuring the record of critiques and reviews.

Summary of the Analytical Module

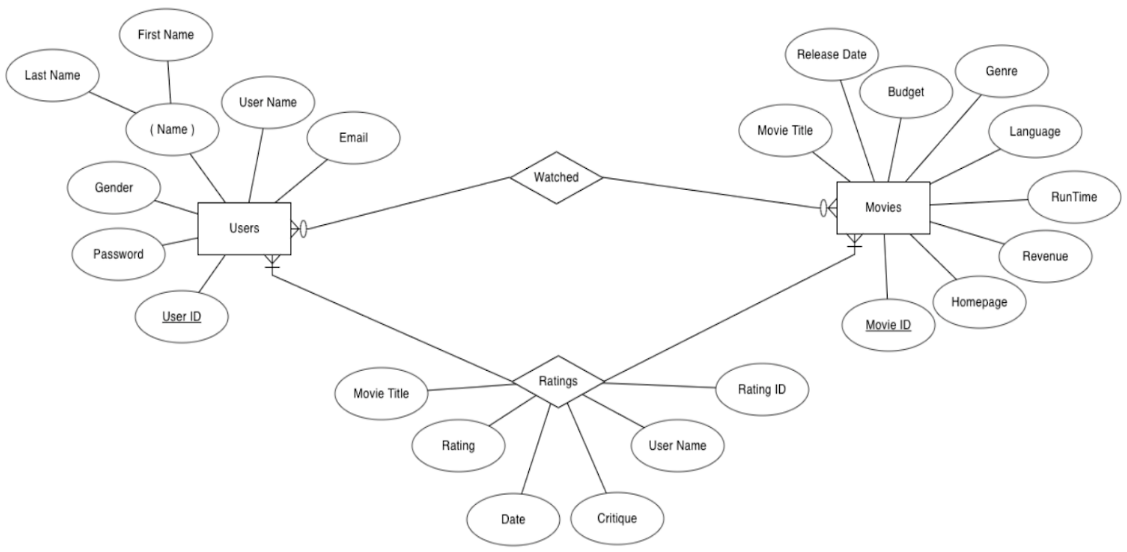
The analytical module is designed to compare and analyze two movies across key metrics. The Movie Analytical Dashboard allows users to compare between rating, runtime, and revenue which are shown using a bar chart. The database also offers insights into the distribution of the movies within the genres, with a bar graph. This allows viewers to make decisions about movie choices based on preferences by previous audiences. Additionally, another bar graph shows the yearly average rating between genres and would help users understand the popularity of different movies and time periods. Overall, the database should allow users and influence them to choose the type of movie to watch based on the visualizations provided.

ETL Process

We first obtained our dataset from Kaggle using the link provided above. However, this dataset, specifically the movies_metadata.csv, contained dictionary keys that were hard to extract and use for analysis. Combined with the fact that there were other columns that were not needed (movie keywords), we decided to use Mockaroo to generate data. After doing so, we cleaned our dataset using Microsoft Excel. For example, the 'Date' column in the Date.csv was not in SQL form, so it converted from a date format to an integer format, which was easier to use later on. In addition, the Date table had duplicate dates, so using SQL code, these were removed (the code is shown in the application queries section). We also used ERD plus to create our ER Diagrams and were later used to create relational schema and star schema. Now that we have our dataset, SQL workbench was used to connect to the SJSU school server using VPN, PyQt Designer was used for the operational module, and Python3 was used for the analytical module.

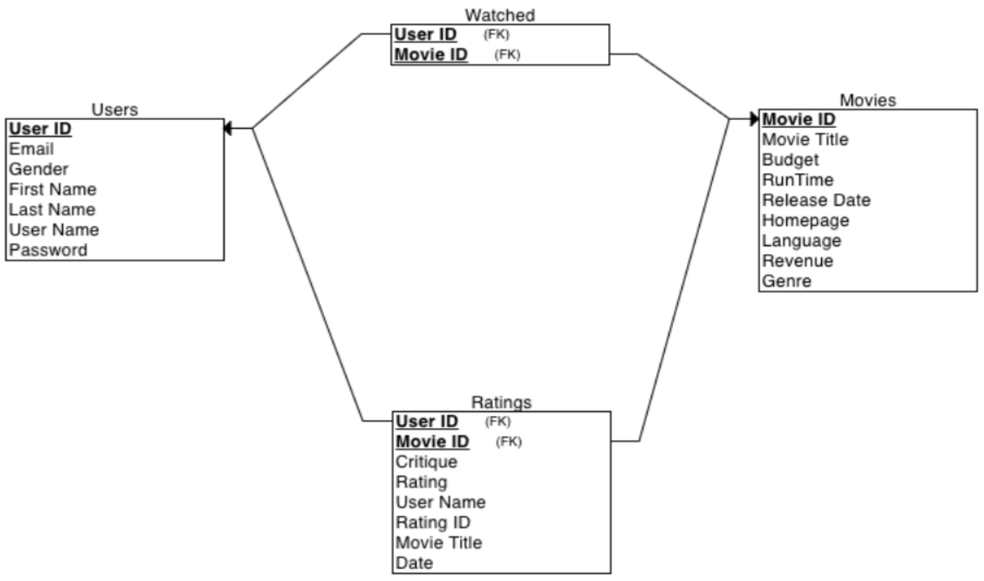
Database Design

Figure 1: ER Diagram



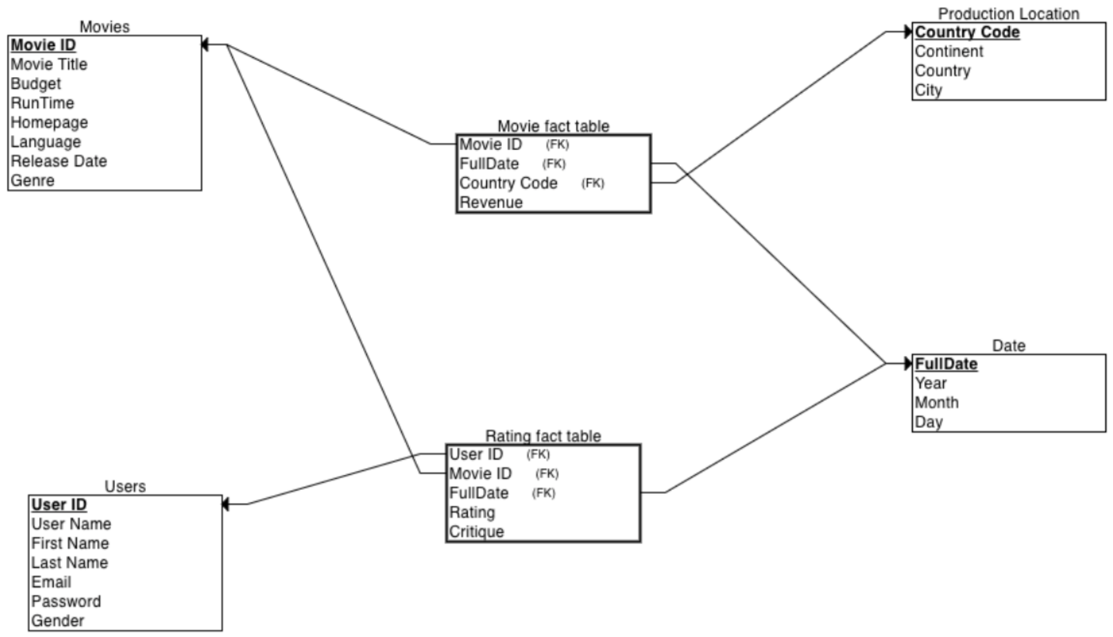
The ER Diagram contains two main entities: users and movies. In each entity, there are attributes. For the Users entity, there is information and attributes such as Password, Gender, UserName, Email, Name, and User ID. The Name attribute is a composite attribute split intro First Name and Last Name. The primary key is the User ID. For the movies entity, there are attributes about each movie such as Movie Title, Release Date, Budget, Genre, Language, RunTime, Revenue, Homepage, and Movie ID, which is the primary key. In the users-movies relationship there is many-to-many, where users rate movies and movies are rated by users, which then provides attributes such as Movie Title, Rating, Date, User Name, Rating ID, and the Critique the user has for the movie.

Figure 2: Relational Schema



The relational schema is similar to the ER Diagram. It has two main entities: users and movies, a many-to-many relationship of ratings. See above for more details.

Figure 3: Star Schema



The star schema has 4 dimensional tables: movies, users, production location, and date. There are two fact tables: rating and movie. The movie fact table is connected with the Movies, Date, and Production Location dimension tables. The rating fact table is connected with the Movies, Users, and Date dimension tables. The Movies dimensional table contains information about movies such as Movie ID, Movie title, budget, runtime, homepage, language, release date, and genre. The users dimensional table contains information about users who watched movies, such as user Id, user name, first name, last name, email, password, gender. The Production location is the location the movie was produced such as country code, continent, country, and city. The Date is the information about the time, specifically year, month, day. The production location and date are in order of specificity so it allows for drill down or drill up in the later steps.

Use Case of the Operational Module

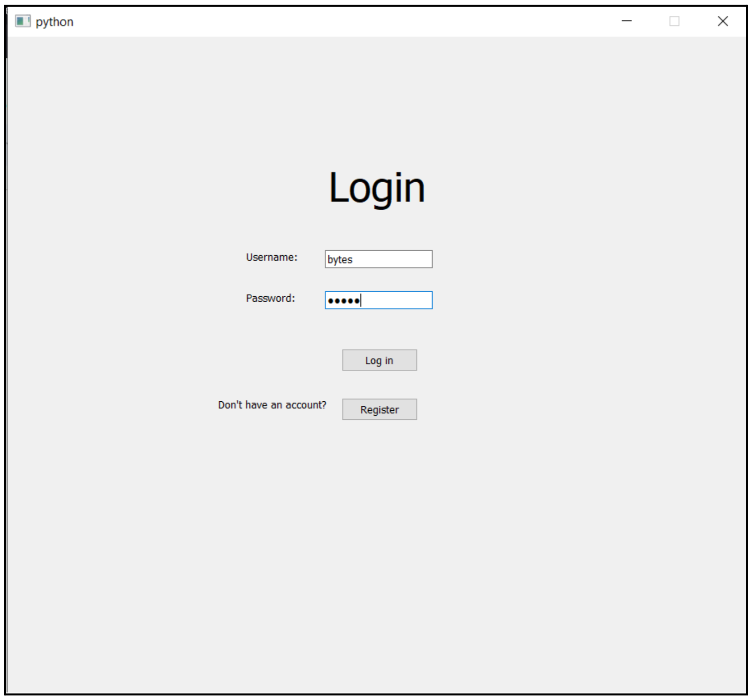
Username: *bytes*
Password: *bytes*

Since this is a user based system, the end user is able to register for an account and log in. Once logged in, they are able to perform a few functions:

First, the user is able to search up a movie, and if it is in the database, it will return the title, genre, average rating, number of reviews, and all reviews made for that particular movie.

The user can also be able to create a review for any movie in the database by first selecting a movie, giving it a rating using a slider, and inputting a critique, if any. Should the user choose to give the review a unique identification, the review is given a review ID upon creation.

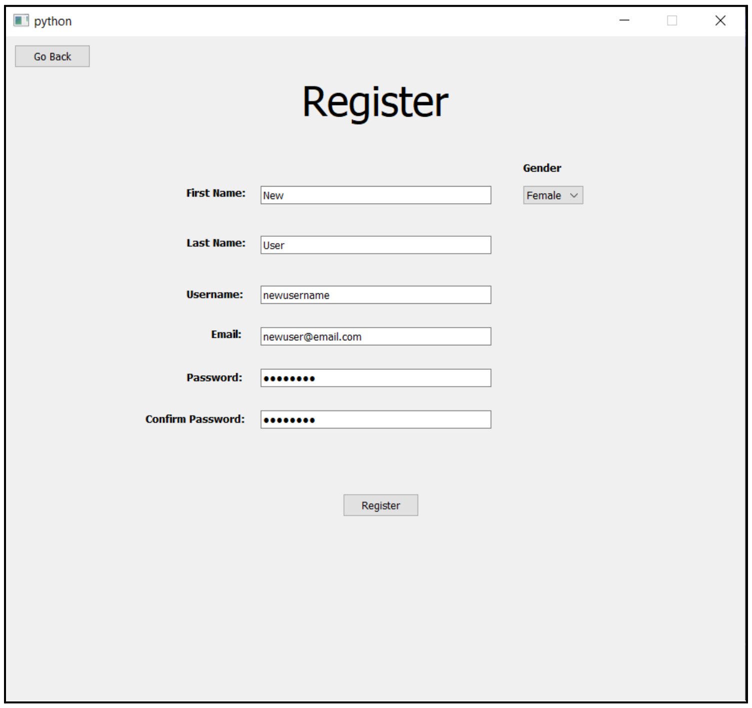
In order to allow the user to view all past reviews they have made, they are able to view their reviews. In addition they are able to edit their reviews or delete them if they wish.



Username: bytes
Password: •••••

Figure 4: Login Account
The operational GUI has a login page that requires a username and password.

If the account does not have a matching username and password in the database, the user can register for an account.

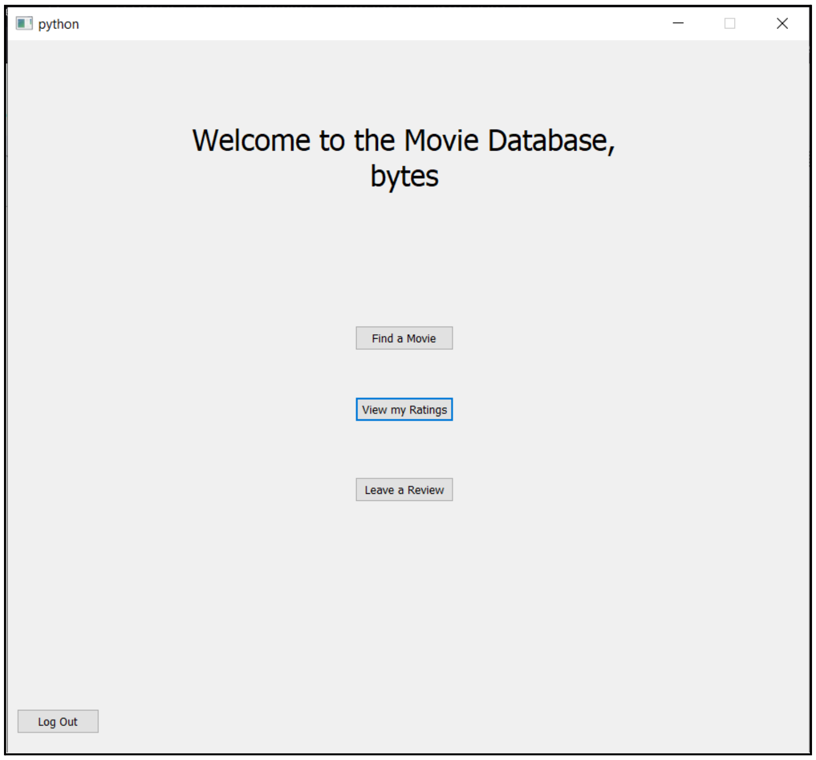


First Name: New
Last Name: User
Username: newusername
Email: newuser@email.com
Password: ••••••••
Confirm Password: ••••••••
Gender: Female

Figure 5: Register Account
The user is able to register for an account by filling out the necessary fields. Note that all fields are required.

If any fields are left blank, or the username or email have already been used, or the passwords do not match, then an error message will appear.

Upon successful creation, the user will be sent back to the login page.



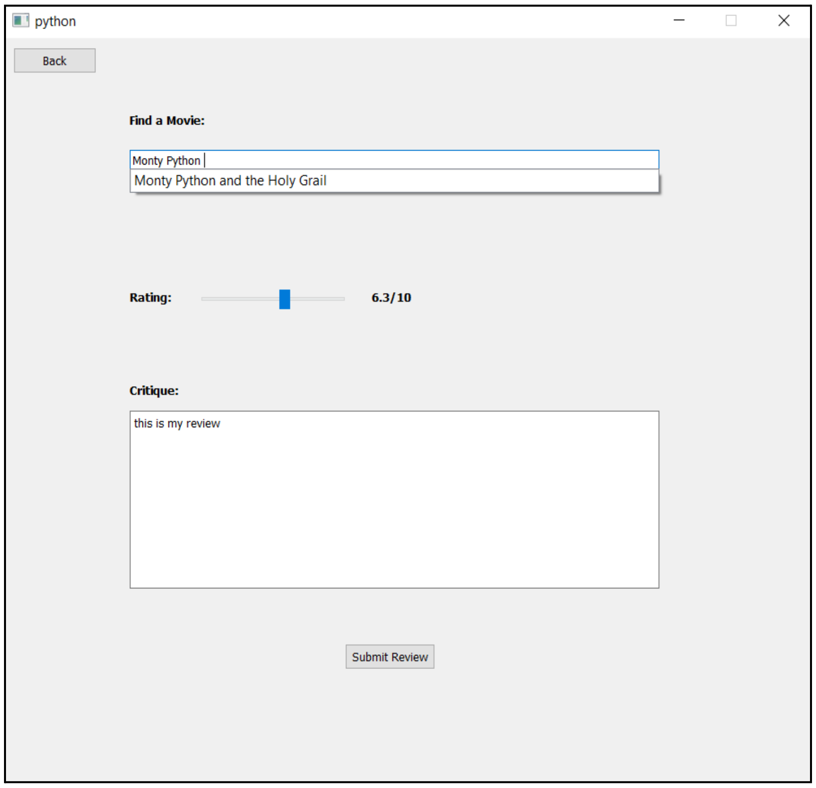
Welcome to the Movie Database,
bytes

Figure 6: Access features
Upon successful login, there are three main functionalities:

1. Search movies by clicking on the **Find a Movie** button.

2. View all ratings the logged-in user has made with **View my Ratings**

3. Create a review using **Leave a Review**



Monty Python

Monty Python and the Holy Grail

6.3/10

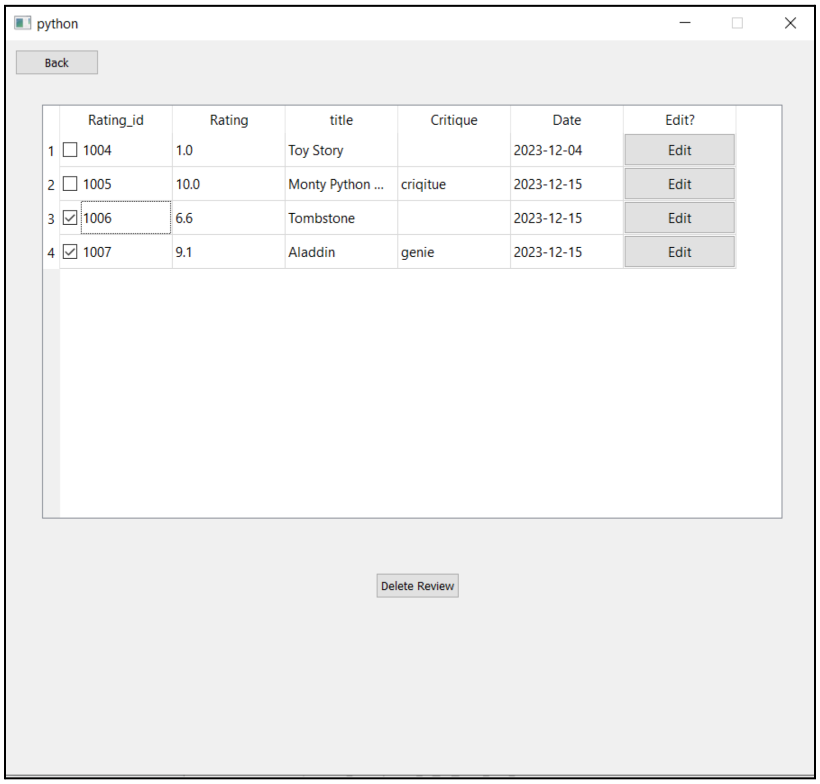
this is my review

Figure 7: Rate a Movie
Here are 3 steps to create a review:

1. Selecting a movie in the database using the search bar

2. Giving a rating out of 10 using the slider

3. Leaving a critique using the text box. This step is optional.

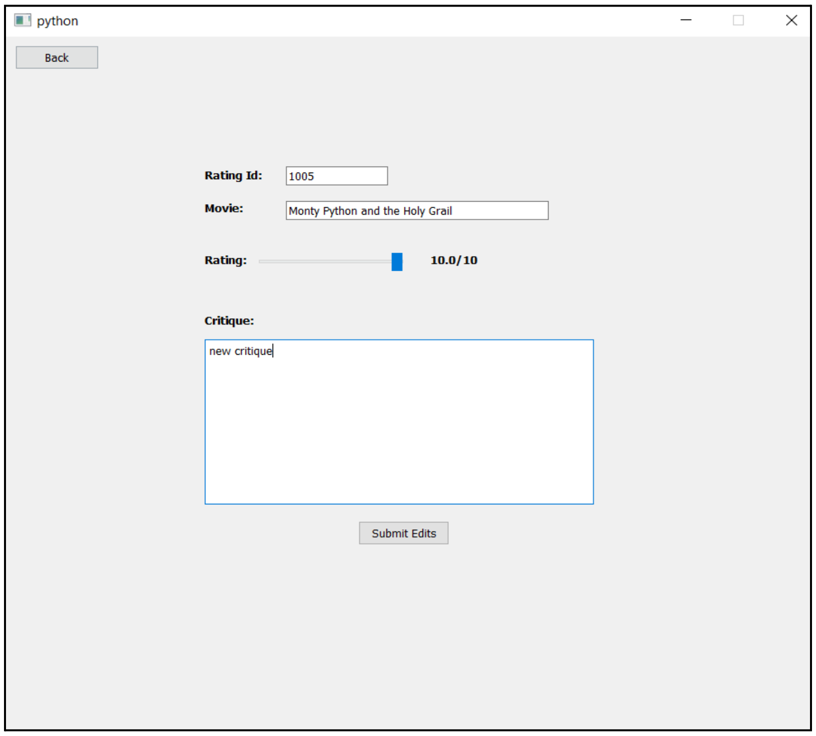


	Rating_id	Rating	title	Critique	Date	Edit?
1	☐ 1004	1.0	Toy Story		2023-12-04	Edit
2	☐ 1005	10.0	Monty Python ...	criqitue	2023-12-15	Edit
3	☑ 1006	6.6	Tombstone		2023-12-15	Edit
4	☑ 1007	9.1	Aladdin	genie	2023-12-15	Edit

Figure 8: Past Reviews
Here the user can view all of their reviews.

Using the checkboxes, the user can select one or many ratings to delete or edit. Note that at least one checkbox must be selected for the delete function to go through.

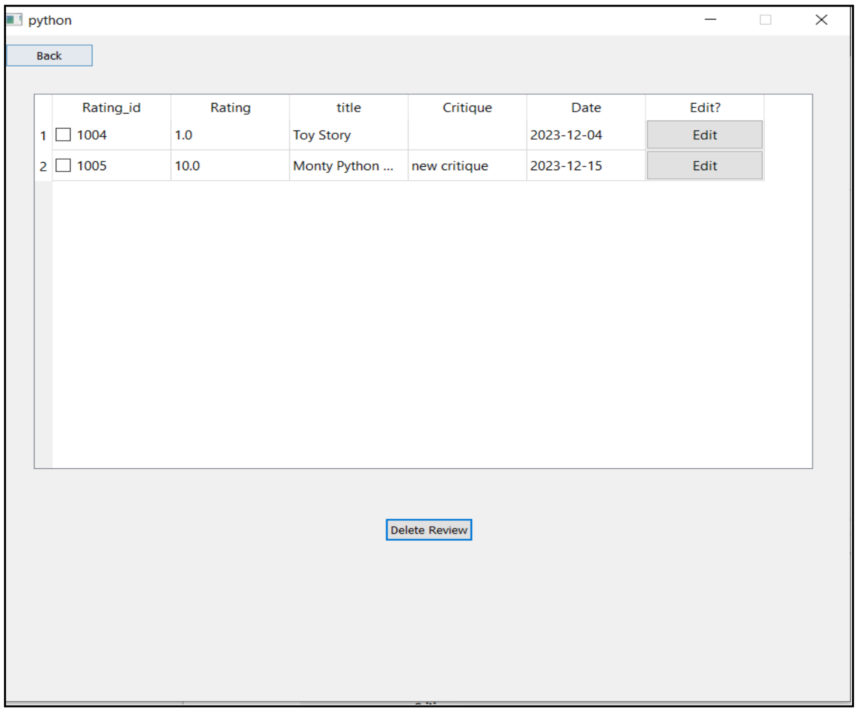
Users can also click on the edit button to bring up a new page to update their reviews.



Rating Id: 1005
Movie: Monty Python and the Holy Grail
Rating: 10.0/10
Critique:
new critique

Figure 9: Change Review
This is the Edit page. The rating ID and movie title are already filled in. These cannot be changed on this page. This ensures that the user understands which review they are changing.

As the example shown, for the movie *Monty Python and the Holy Grail,* the user rated 10/10 and gave his/her review.



python

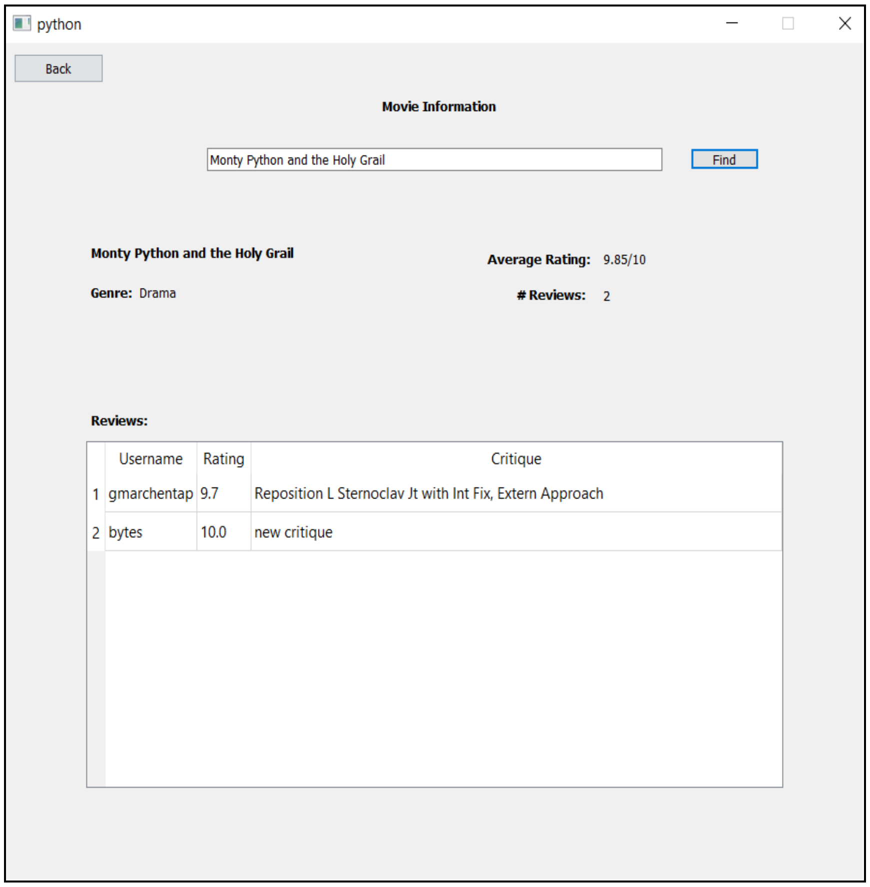
Back

	Rating_id	Rating	title	Critique	Date	Edit?
1	1004	1.0	Toy Story		2023-12-04	Edit
2	1005	10.0	Monty Python ...	new critique	2023-12-15	Edit

Delete Review

Figure 10: Edit History
This shows the viewer rating table from above after deletes and edits have been made.

This ensures that a history log has been created for the user, in case the user wants to look back at his/her critiques.



python

Back

Movie Information

Monty Python and the Holy Grail

Find

Monty Python and the Holy Grail

Average Rating: 9.85/10

Genre: Drama

# Reviews: 2

Reviews:

	Username	Rating	Critique
1	gmarchentap	9.7	Reposition L Sternoclav Jt with Int Fix, Extern Approach
2	bytes	10.0	new critique

Figure 11: Search Movie
This allows the user to search for a movie using the search bar. If a movie is not in the database or nothing is inputted, an error message is printed.

Otherwise, the database is queried and the title, genre, average rating, number of reviews, and all reviews made by users on the system will be shown.

Use Case of the Analytical Module

The analytical database focuses on the comparison between two movies that are chosen by the user. This is an important part of the database, providing users with advanced tools for comparison and in-depth analysis.

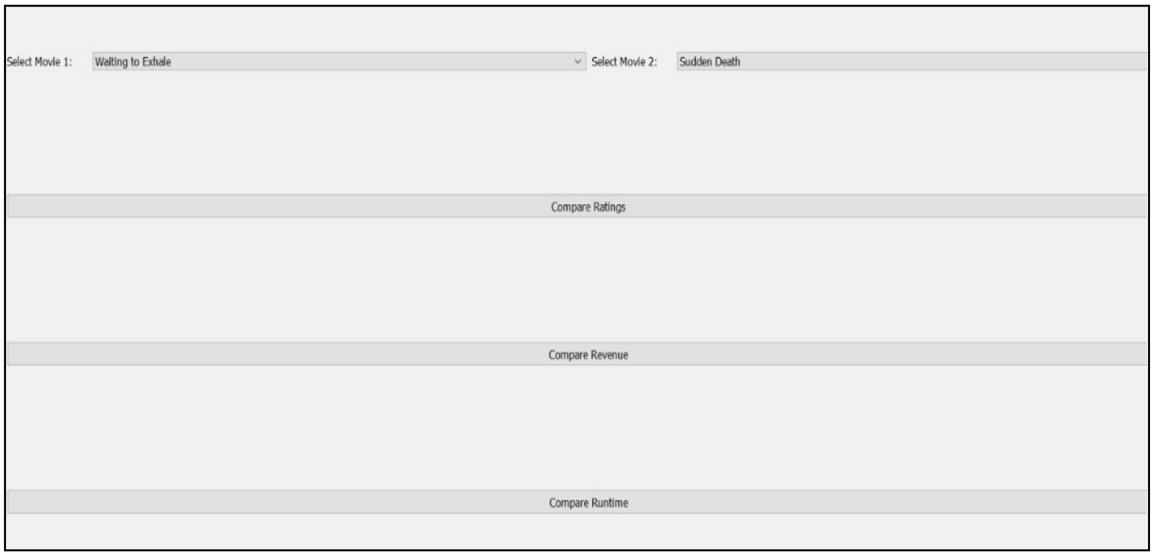


Figure 12: Movie Analytics Dashboard

This is the first thing the user will see. Then, the user can compare the movies accordingly. After selecting any two of the movies, users can select the three buttons to find movie comparisons.

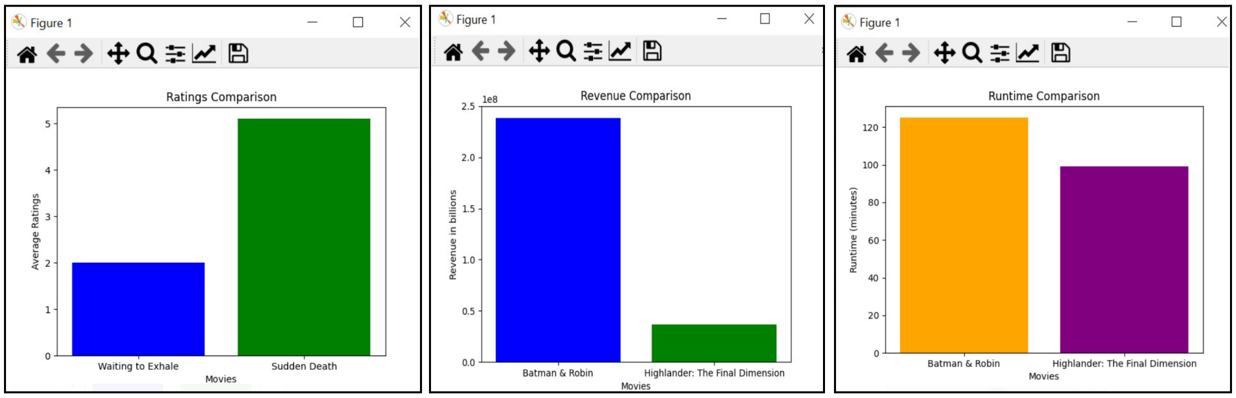
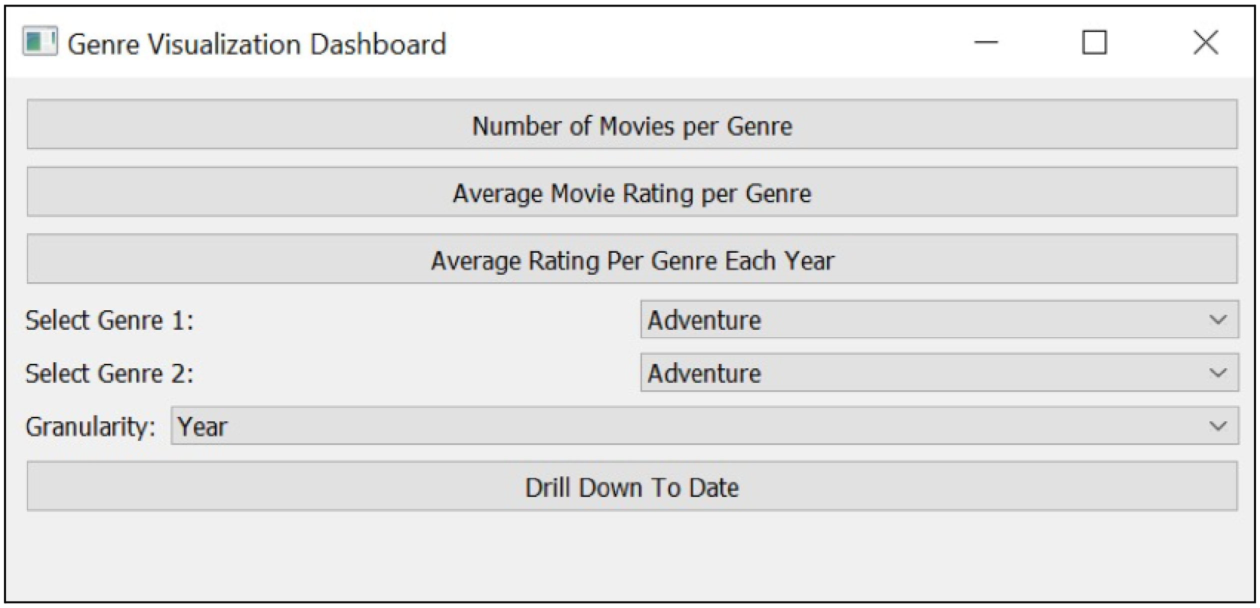


Figure 13: Bar Charts of Ratings, Revenue, and Runtime

For movie enthusiasts, the process of comparing two films can be advantageous. Analyzing ratings, box office revenue, and runtime provides valuable insights into a movie's commercial success, critical acclaim, and audience engagement.

Besides comparing between two movies on ratings, revenue, and runtime, the user can get information about the genre of the movie. Specifically, the user can view the number of movies per genre, average ratings per genre, and a time series with drill down for two selected genres.



Genre Visualization Dashboard

Number of Movies per Genre
Average Movie Rating per Genre
Average Rating Per Genre Each Year

Select Genre 1: Adventure

Select Genre 2: Adventure

Granularity: Year

Figure 14: Genre Visualization Dashboard

This is what the user will see. The user will have options to click on certain features and select the different genres and the granularity.

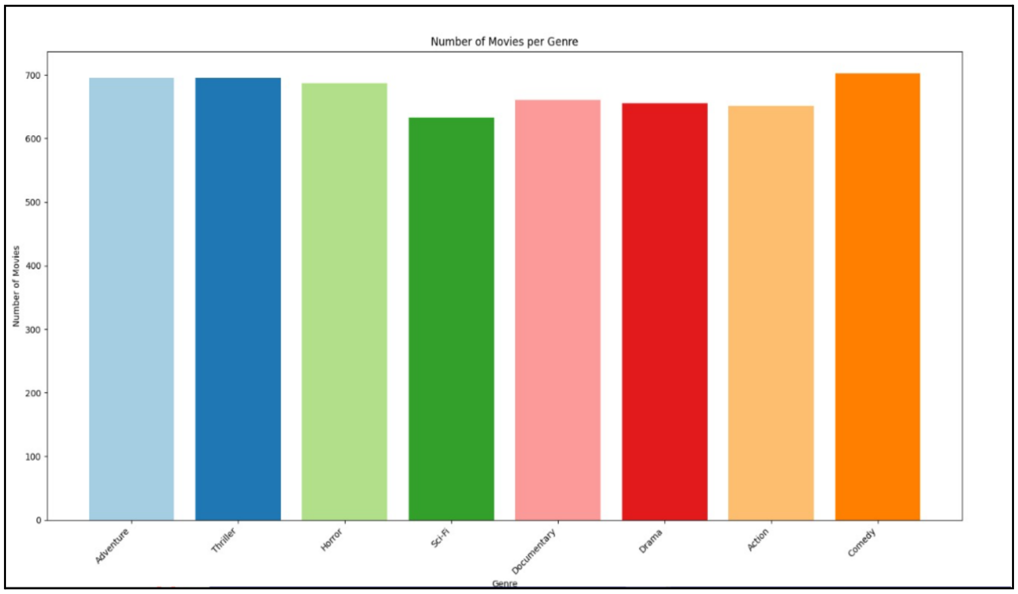


Figure 15: Number of Movies per Genre

Another bar graph to show the number of movies within specific genres.

There were more Comedy movies than Sci-Fi movies. Note this data is not real.

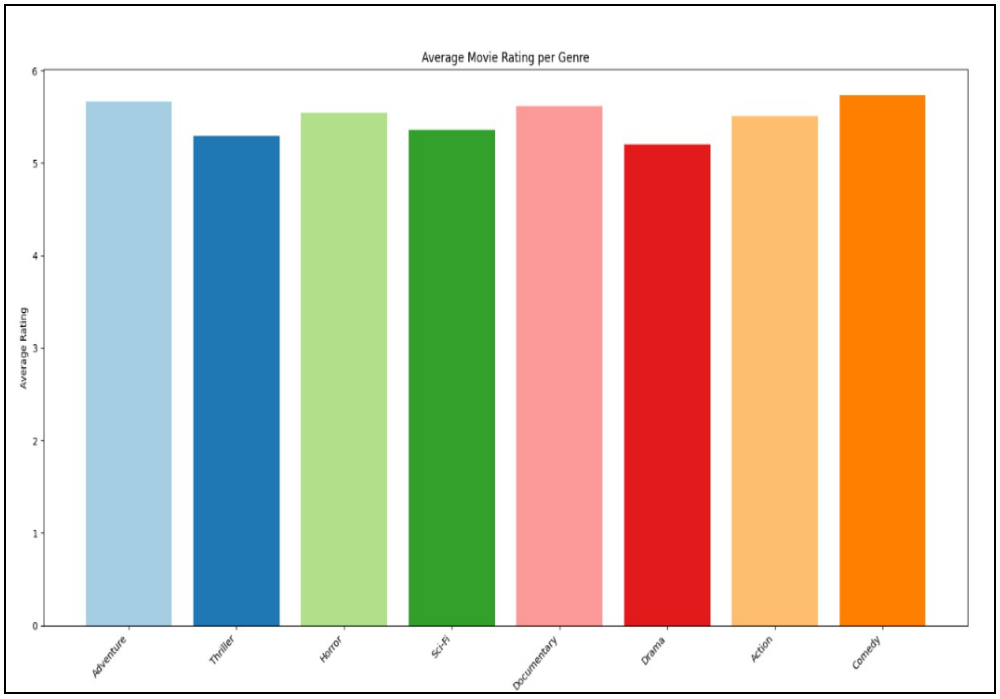


Figure 16: Average Genre Rating

Genres include action, drama, comedy, science fiction, horror, and romance.

Drill Down:

The following figures show the drill down function for year, month, and day.

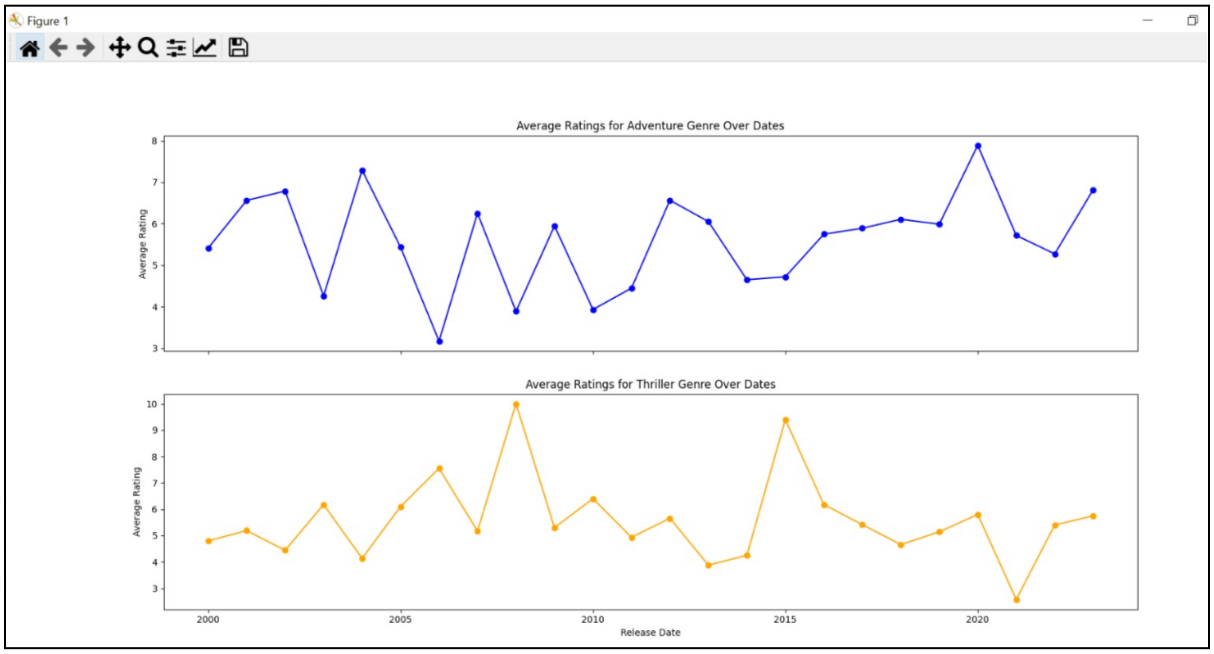


Figure 17: Average Ratings by Year

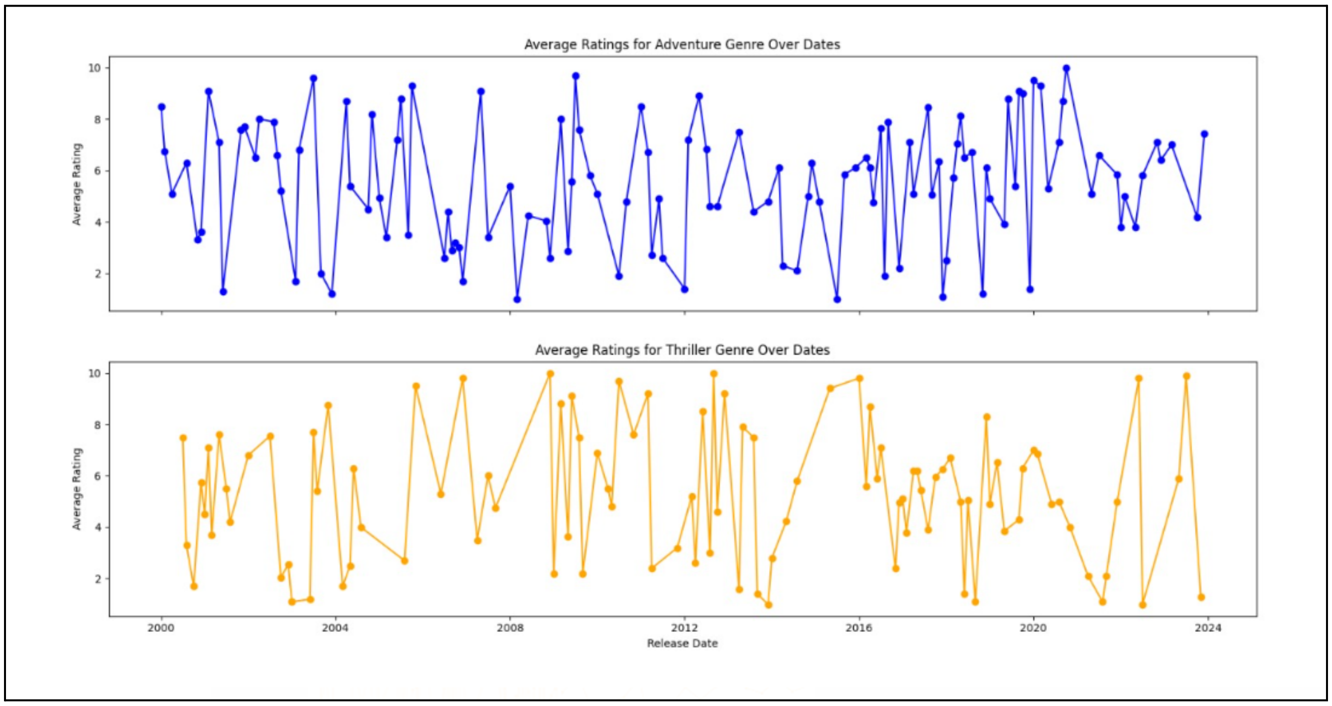


Figure 18: Average Rating by Month

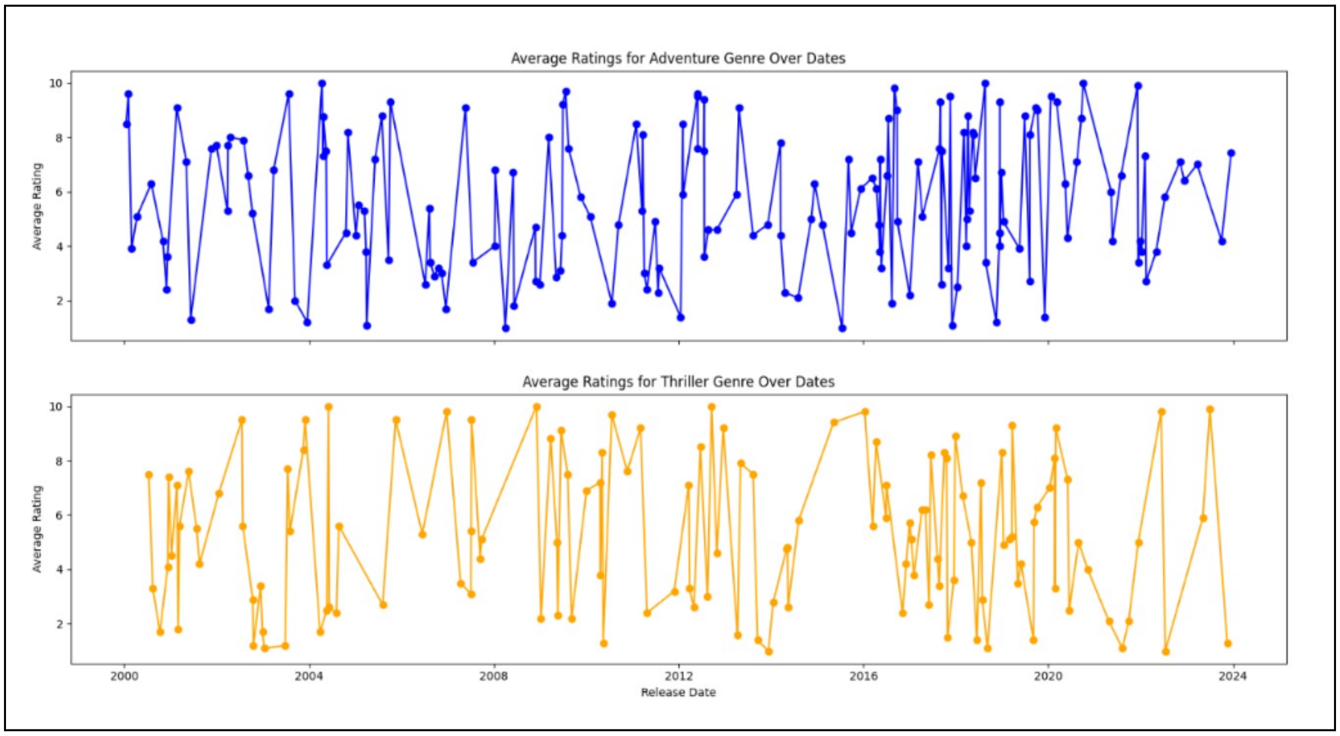


Figure 19: Average Ratings by Day

Note that the above figures may look inconsistent because the data was generated and not real data. The database also conducted ratings of distinct genres over a yearly basis. By checking average ratings by genre over time, the user can see each type of movie as time goes on. This way, viewers can know what movies are most-liked in a particular year and choose to watch what people are enjoying the most.

This helps us see which types of movies are more popular. It's not just about what people like, but it also tells us about what's happening in the movie world and how well different types of movies are doing.

Technical Aspects

The launching of the application can be completed using the Terminal on the computer. Note that the necessary data tables are on the school server. The files should be stored inside a folder and executed with the commands shown:

1. Operational: python operational.py
2. Analytical: python Movies_Analytical_Dashboard.py
   python Genre_Visualization.py

Here are the files contained in each module:

- Operational:
    - DATA225utils → this is the utility file that has common functions required
    - operational.py → the main py file used for running the Operational GUI
    - operational.ini → the utility file used to connect to the database on the server
    - edit_dialog.ui → the user interface file for the edit rating page
    - login_dialog_ui → the user interface file for the login page
    - mainMenu.dialog.ui → the user interface file for the menu that links to the movie finder, rating submission, and rating viewer pages
    - movie_dialog.ui → the user interface for the movie finder page
    - rating_dialog.ui → the user interface for the rating submission page
    - register_dialog.ui → the user interface for the account registration page
    - view_dialog.ui → the user interface for the user’s rating viewer page

- Analytical:
    - Movies_Analytical_Dashboard.py → this file contains the comparison between any two movies from the dataset by metrics like revenue, average ratings and run time of selected movies.
    - Genre_Visualization.py → this shows movie count by genre, average rating by genre, and average rating over years of selected genres displayed in fig 14.

Application Queries

SQL Code that do not show up in the Python files
For the rest of the queries used, please refer to python files

ETL for date table. There were some duplicate dates, so this removed them

```
DELETE d FROM date d
inner join (SELECT *,ROW_NUMBER() OVER (PARTITION BY Date ORDER BY Date) AS RN
FROM date) test on d.Date = test.Date
WHERE RN > 1
```

Stored Procedure for updating analytical database based on operational database :
**Note: This stored procedure is in the school server warehouse.*

```
CREATE DEFINER=`bytes_user`@`%` PROCEDURE `update_warehouse`()
BEGIN
INSERT INTO users_password
SELECT *
FROM bytes_db.users
WHERE bytes_db.users.user_id NOT IN (SELECT user_id FROM users_password);

SET SQL_SAFE_UPDATES = 0;
delete from rating
WHERE not exists(select * from bytes_db.ratings where rating.rating_id =
bytes_db.ratings.rating_id
and rating.Rating = bytes_db.ratings.rating and rating.Critique =
bytes_db.ratings.Critique
and rating.Date = bytes_db.ratings.Date);
SET SQL_SAFE_UPDATES = 1;

INSERT INTO rating
SELECT *
FROM bytes_db.ratings
WHERE bytes_db.ratings.rating_id NOT IN (SELECT rating_id FROM rating);

SET SQL_SAFE_UPDATES = 0;
DELETE FROM rating
WHERE NOT EXISTS (SELECT * FROM bytes_db.ratings WHERE rating.rating_id =
bytes_db.ratings.rating_id);
```

```
SET SQL_SAFE_UPDATES = 1;

INSERT INTO date
SELECT DISTINCT a.Date, YEAR(a.Date), MONTH(a.Date), DAY(a.Date) FROM
bytes_db.ratings a
WHERE NOT EXISTS(SELECT Date from date WHERE Date = a.Date);

END
```

References

"ERDPlus." ERDPlus, www.erdplus.com.

Mak, Ron. "Apropos Logic - Database Class." Apropos Logic, www.apropo s-logic.com/dbclass/.

Mak, Ron. "DATA 225: Principles and Techniques of Data Science." San Jose State University Computer Science Department, www.cs.sjsu.edu/~mak/DATA225/.

"Mockaroo." Mockaroo, www.mockaroo.com

"The Movies Dataset." Kaggle, www.kaggle.com/datasets/rounakbanik/the-movie s-dataset.