ENSF 692: Final Project

Barrett Sapunjis, Marley Cheema

Instructor: Sarah Shah

A Movie Database Query CLI & Database Statical Analysis

1.0 Executive Summary and Introduction

For the ENSF 692 final project, this team set out to use the concepts, learning outcomes, and tools provided by the ENSF 692 class to answer the following problem statement:

To build an interactive CLI that empowers the user to find movies that match exactly what they are looking for based on the inputs they provide, and eventually provide them with a list of these movies with corresponding data associated with them, such as their ratings, genre, and actors and actresses, to name a few.

Early in the conceptualization process, this team realized that in order to make a comprehensive CLI, a lot of good data would be required. After experimenting with many different resources and analyzing different data sets, this team managed to find a combination of three data sets that would provide enough information to not only fulfill the requirements of the project but also be comprehensive enough to provide a great database of movies for the user. This will be further expanded upon within section “2.0 Data Sets” of this report.

Additionally, based on the data obtained, this team realized that there was potential to extract a ton of valuable insights from the data, as essentially the data represents every movie accounted for on IMDB since 1980. Through performing analysis on the data set as a whole, the team was able to identify and unlock information about trends in movie ratings, make strong comparisons between genres and the quality of movies that exist within them through ratings, and better understand the relationship of factors that influence movie ratings through performing linear regressions. This team felt that due to the great magnitude of data, this was critical to complete.

Ultimately, this allows the team's project to be thought of as segmented into two separate parts; however, they are connected as all work with the same constructed, complete Pandas Multi-Index DataFrame, which was constructed from raw data. All data manipulation, functions, and analysis for both parts exist in the same Python file called: DataHandler.py.

Part 1: Movie Data Base Filtering Based on User Input (which internally named for consistency in this report) reflects the CLI itself, which is interactive with the user and is the “foreground” of our project. As an overview, it allows for a DataFrame of specific movies to be returned, which is fully dependent on the user’s choices and all logic within the CLI based on their inputs. The interface exists within the Python file called: app.py and calls upon functions that exist within DataHandler.py, which utilizes Pandas operations that empower the user to receive a smaller DataFrame of movies that match their criteria. Some major components of Part 1 reflect

operations that use masks, hierarchical indexing, slicing, grouping, and more, which all in unison enable the concatenation of the complete DataFrame into one that corresponds to their inputs.

Part 2: Statical Analysis of IMDB Movies Since 1980 - reflects the statistical analysis, which is again performed on the complete constructed DataFrame, referenced in the code as data. The execution of the statistical analysis occurs within the file DataStatsScript, which again calls upon functions that exist in DataHandler.py. However, the output of the statistical analysis is called upon in our app.py file and presented to the user as a nice bonus of movie insights after they request their final DataFrame of movies (indicating they are done sorting). However, these stats are independent of the user input, as they are done on the complete DataFrame to unlock insights on movies as a collective since 1980. This serves to strengthen the quality of our program for the user, adding more value, and enables us to successfully meet all the requirements of the project, which include the statistical analysis portion, and strengthen our skills in this regard when working with a large data set.

2.0 Data Sets

This team pulled data that comes directly from IMDB (Internet Movie Database) through an external website that is also maintained by IMDB [1]. In the early stages of development, we had acquired movie data from Kaggle.com, but after further research, we quickly realized that, within the scope of this assignment, the source we used was both the most reputable and comprehensive, ultimately improving the quality of our project.

From https://datasets.imdbws.com/, we acquired three sets of data that were most pertinent to our goals within the CLI. What made this dataset so appealing was that they all shared a commonality: each file included a movie ID (a two-letter and seven-number code), which we recognized could be very powerful for merging the files and constructing a complete Pandas DataFrame. An overview of the files and their preprocessing modifications can be found in the table below.

Original File From Source	**Preprocessing Modifications**	**Justification**	**Post Processing File Name (as seen in project folder)**
name.basics.tsv.gz	This file was modified to only contain names of people whose role indicated they were an actor or actress.	The reason for this was that we were not concerned with additional individuals for our database (such as directors, producer ect).	namesActorActressOnly.csv
title.ratings.tsv.gz	No modifications performed	This file was already super clean, and all the information was valuable.	ratings.csv
title.basics.tsv.gz	This file was modified to only contain movies released after 1980	Ultimately as the files were so large, we knew that we had to limit our data to some capacity. By concatenating this file, we knew it could serve to govern the other through our data merge and construction.	titles1980.csv

3.0 Constructing Data & Merge

After completing the acquisition and preprocessing of our data, as outlined above, the next and most critical step was constructing a DataFrame. This was achieved through the construct_data() function, which can be found in DataHandler.py.

This function reads the CSV files and performs some additional preprocessing to ensure proper formatting. This includes: adding principals to ensure actors and actresses are connected to all the movies they were a part of and separating them into different columns, dropping columns deemed irrelevant, converting actors and actresses into lists within their respective movie columns, accounting for void data, and finally merging the three different datasets into one final Pandas DataFrame with appropriate indexing.

We decided to set the first layer of indexing within our DataFrame hierarchy to be the date of release, and the second layer to be the title of the movie itself. Within our final DataFrame, we included 445,427 rows (representing 445,427 different movies), 11 columns, and two layers of indexing, which our team felt very satisfied met all the requirements for this portion of the project.

A table depicting the key features of our complete DataFrame, created by the construct_data() function, can be seen below:

DataFrame Function	**Shape**	**Levels of Indexing**	**Number of Columns**	**Number of Rows**	**dtype**
construct_data():	(445427, 11)	2	11	445427	‘object’

4.0 Interface

The interface of our project lies within the file app.py. The goal of this interface was to ensure that the experience the user had when working with our CLI was truly user-centered. This was so the user could choose exactly how many times and by which methods they wanted to manipulate the DataFrame, as outlined above, into a much smaller DataFrame of movies that met the conditions they wanted to apply. After each splicing of the DataFrame based on their chosen filtering method, they could see how many movies remained that met their requirements and could choose to filter further based on another condition or receive their final list of movies.

We gave users the option to filter their movies by the following four conditions:

1.) Filter by Genre
2.) Filter by Release Date
3.) Filter by Actors / Actresses
4.) Filter by Minimum Rating

Followed by another 4 options once they were satisfied with the filtering they applied.

5.) Export data to excel
6.) Get user data analysis
7.) Reset data
0.) Exit

Additionally, to ensure invalid entries were handled, if one of the filtering options above was not selected, a ValueErrorwould be raised, allowing the user to continue being prompted until a valid selection was made. Additionally, within each of these filtering options, if an incorrect input (respective to each option) was entered, an error message would be printed, and the user would be returned to the original filtering prompt.

Each of the filter options has an associated function beginning with “get” that can be found in DataHandler.py, which takes in the current DataFrame (this is important to ensure that previous filters are continuously applied) and returns a new, smaller DataFrame based on the filtering method and input, which is then used for all future filtering.

An example of a potential logic process from the user can be seen below, which comes directly from our code.

```
Please choose from the following options:
[1] Filter by genre
[2] Filter by release date
[3] Filter by actor/actress
[4] Filter by minimum rating
[5] Export data to excel
[6] Get user data analysis
[7] Reset data
[0] Exit
Enter a number (0-7): 1
Enter a genre to filter by: Action

🔍 Currently based on your selections your output list of movies has 36382 entries that match the criteria!
Would you like to apply another filter on this list or select a new option? (Y/N): y
Please choose from the following options:
[1] Filter by genre
[2] Filter by release date
[3] Filter by actor/actress
[4] Filter by minimum rating
[5] Export data to excel
[6] Get user data analysis
[7] Reset data
[0] Exit
Enter a number (0-7): 4
Enter minimum rating threshold (e.g. 7.5): 3

🔍 Currently based on your selections your output list of movies has 25604 entries that match the criteria!
Would you like to apply another filter on this list or select a new option? (Y/N): y
Please choose from the following options:
[1] Filter by genre
[2] Filter by release date
[3] Filter by actor/actress
[4] Filter by minimum rating
[5] Export data to excel
[6] Get user data analysis
[7] Reset data
[0] Exit
Enter a number (0-7): 3
Enter actor/actress name to filter by: robert downey jr.

🔍 Currently based on your selections your output list of movies has 16 entries that match the criteria!
Would you like to apply another filter on this list or select a new option? (Y/N): n
```

5.0 Operations & Requirements

By the time this team had completed the project, a total of 19 functions were created—the majority of which related to **Part 1: Movie Database Filtering Based on User Input**, and six of which performed statistical analysis on the entire dataset, pertaining to Part 2 of our project: **Statistical Analysis of IMDB Movies Since 1980**.

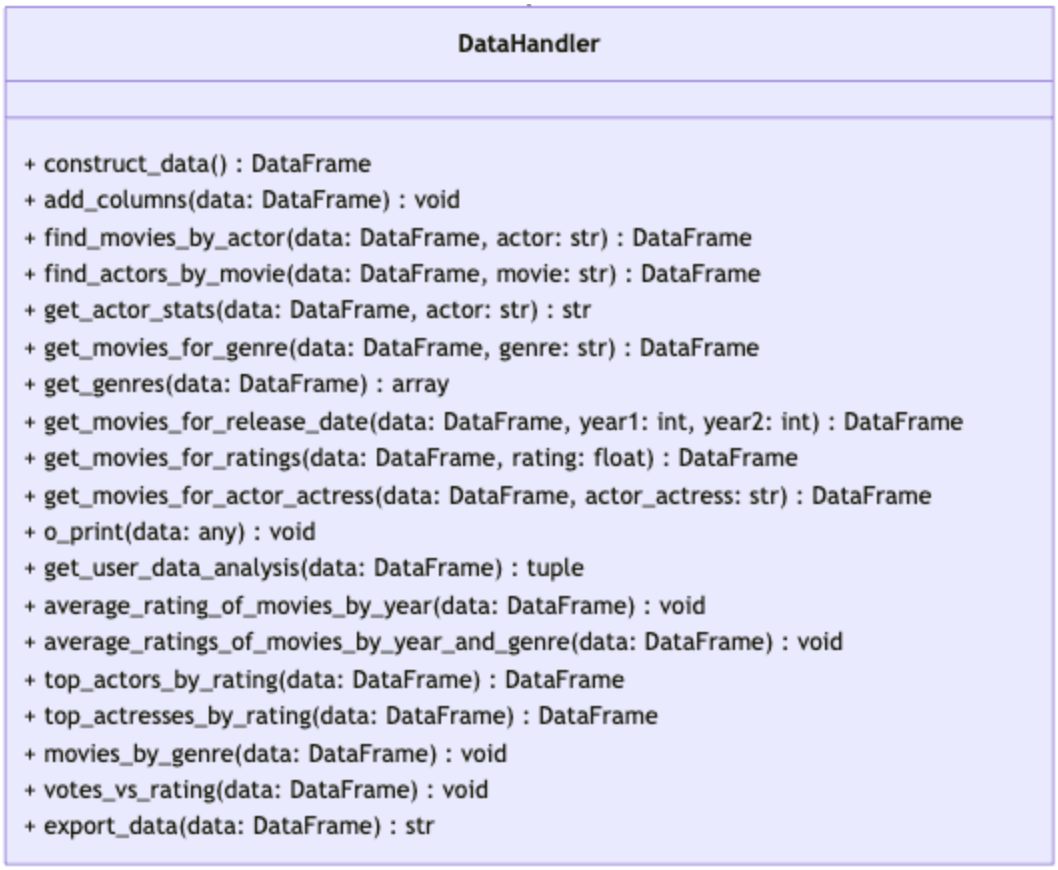
While each function has its own specific functionality and returns, a cohesive concept existed in how we built the functions: each function would take a DataFrame as a parameter and perform some sort of manipulation on it.

For functions pertaining to Part 1, along with the DataFrame, the input for each function was generally accompanied by a variable representing the user input. Based on that input, filtering was applied through operations such as grouping, slicing (.loc), and applying masks—all of which are utilized in different capacities.

For functions that pertained to Part 2, these functions contained an array of methods that extracted value from the data as a whole. Our team used a lot of creativity in thinking about how to analyze the data, as there were so many possibilities. This included using different techniques such as value counts, applying means, grouping data, and utilizing Matplotlib, Seaborn, and Pandas pivot tables to further visualize and statistically analyze the data. Ultimately, through the functions used in Part 2 of our project, we successfully managed to meet all the additional requirements of the project that were not fulfilled solely by Part 1, including data visualization methods and the use of pivot tables.

Lastly, it is important to acknowledge a reference to ChatGPT for helping aid this portion, as it provided significant inspiration and excellent information on how to best utilize Matplotlib and Seaborn to produce great results for the user—though, as always, not for specific code generation.

A list of all our functions and their parameters can be found below:

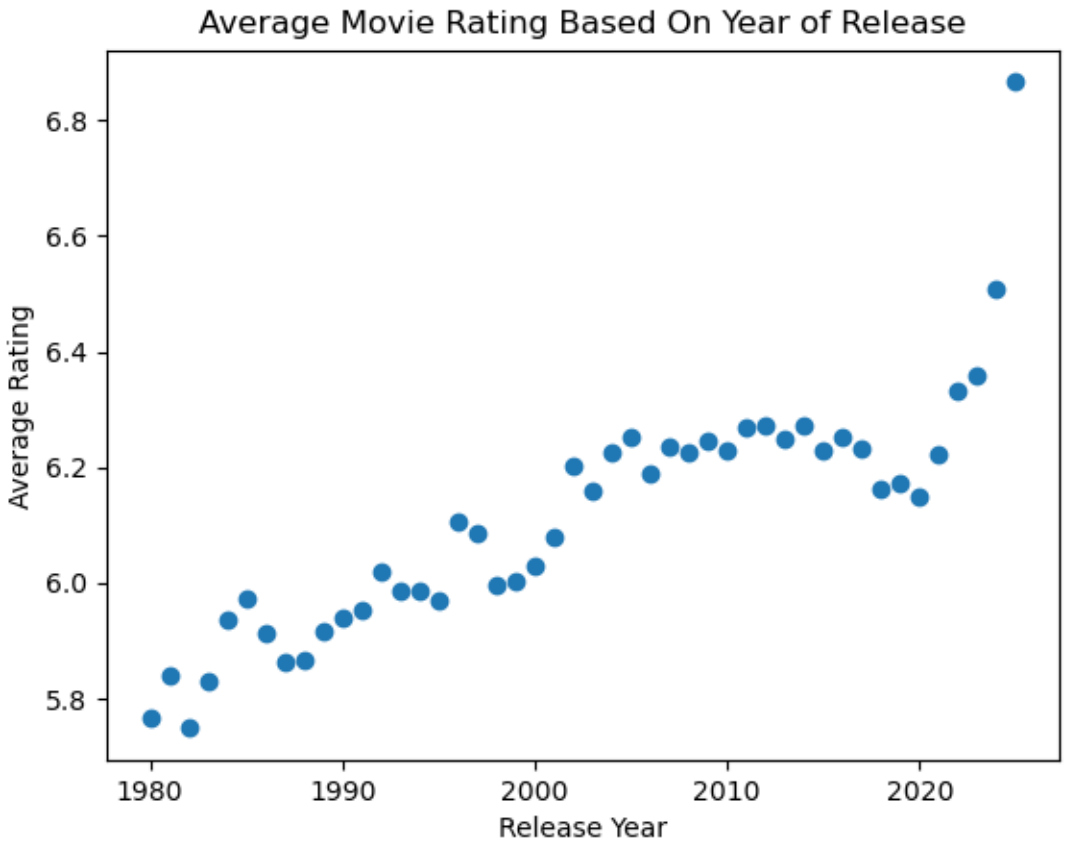

DataHandler
+ construct_data() : DataFrame
+ add_columns(data: DataFrame) : void
+ find_movies_by_actor(data: DataFrame, actor: str) : DataFrame
+ find_actors_by_movie(data: DataFrame, movie: str) : DataFrame
+ get_actor_stats(data: DataFrame, actor: str) : str
+ get_movies_for_genre(data: DataFrame, genre: str) : DataFrame
+ get_genres(data: DataFrame) : array
+ get_movies_for_release_date(data: DataFrame, year1: int, year2: int) : DataFrame
+ get_movies_for_ratings(data: DataFrame, rating: float) : DataFrame
+ get_movies_for_actor_actress(data: DataFrame, actor_actress: str) : DataFrame
+ o_print(data: any) : void
+ get_user_data_analysis(data: DataFrame) : tuple
+ average_rating_of_movies_by_year(data: DataFrame) : void
+ average_ratings_of_movies_by_year_and_genre(data: DataFrame) : void
+ top_actors_by_rating(data: DataFrame) : DataFrame
+ top_actresses_by_rating(data: DataFrame) : DataFrame
+ movies_by_genre(data: DataFrame) : void
+ votes_vs_rating(data: DataFrame) : void
+ export_data(data: DataFrame) : str

6.0 Analysis & Results

As for Part 2 of our project, we managed to derive some key insights into movies released since 1980 and wanted to provide an overview and potential explanations of our findings.

6.1 Average Movie Rating Based on Year of Release

To begin our statistical analysis, our team wanted to gain a general understanding of IMDB rating trends. This led us to our first piece of analysis: examining whether there was any connection between the year of a movie's release and its rating. By calculating the mean rating of movies grouped by their release year and plotting it against the year, we produced the plot below.

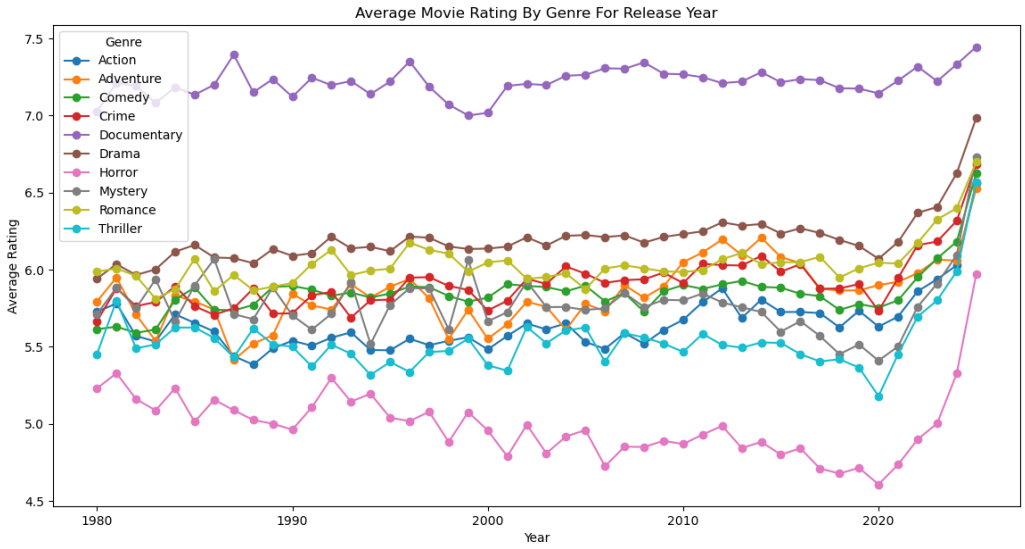


This graph is quite insightful, as it shows that, generally, ratings have been slowly increasing since 1980. While this is not enough to fully conclude this trend as definitive—and does not account for additional factors—it is certainly interesting to observe. This initial analysis sparked further questions and helped guide the subsequent analyses presented below.

6.2 Average Movie Rating by Genre Based on Year of Release

Next, we wanted to see how ratings varied across different movie genres.

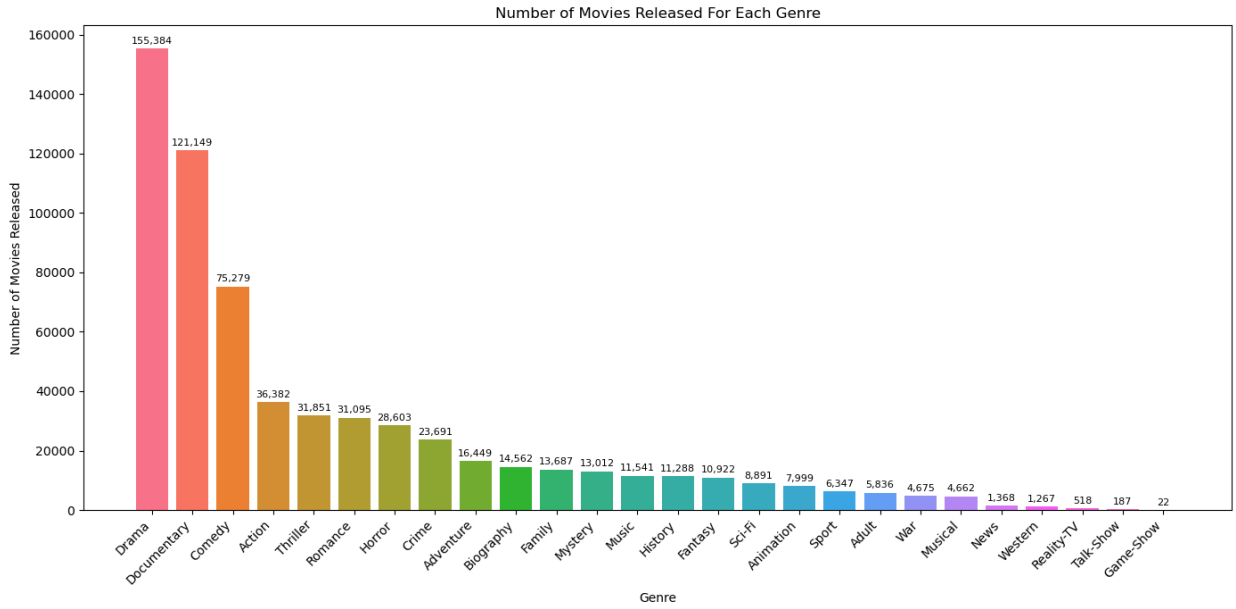
This was achieved by grouping our data by both genre and year of release, calculating the mean rating, and plotting it against the release year.



Our results were certainly interesting, and it became very evident that, historically, documentaries have consistently been ranked the highest on IMDB. This trend has remained consistent through the years. Additionally, horror movies have historically performed far worse than almost all other genres in IMDB ratings as a collective; however, they appear to be experiencing a significant increase since 2020. Almost all other genres fall somewhere in the middle, with relatively stagnant and consistent average ratings compared to other genres, while also experiencing an average rating increase since 2020 like the horror genre.

6.3 Number of Movies Released by Genre Since 1980

To dive deeper next, our team wanted to see exactly how our data split by genre to give further insight. We thought that by using a bar graph we could effectively represent this information best. Through grouping by genre and using value.counts() built-in function we were able to determine that IMDB represents all movies through 26 different genres (although a movie can belong to more than 1) and that since 1980 the genre with the most released movies falls under Drama with over 155 384 releases and the smallest genre of movies is Game Shows with 22 releases.

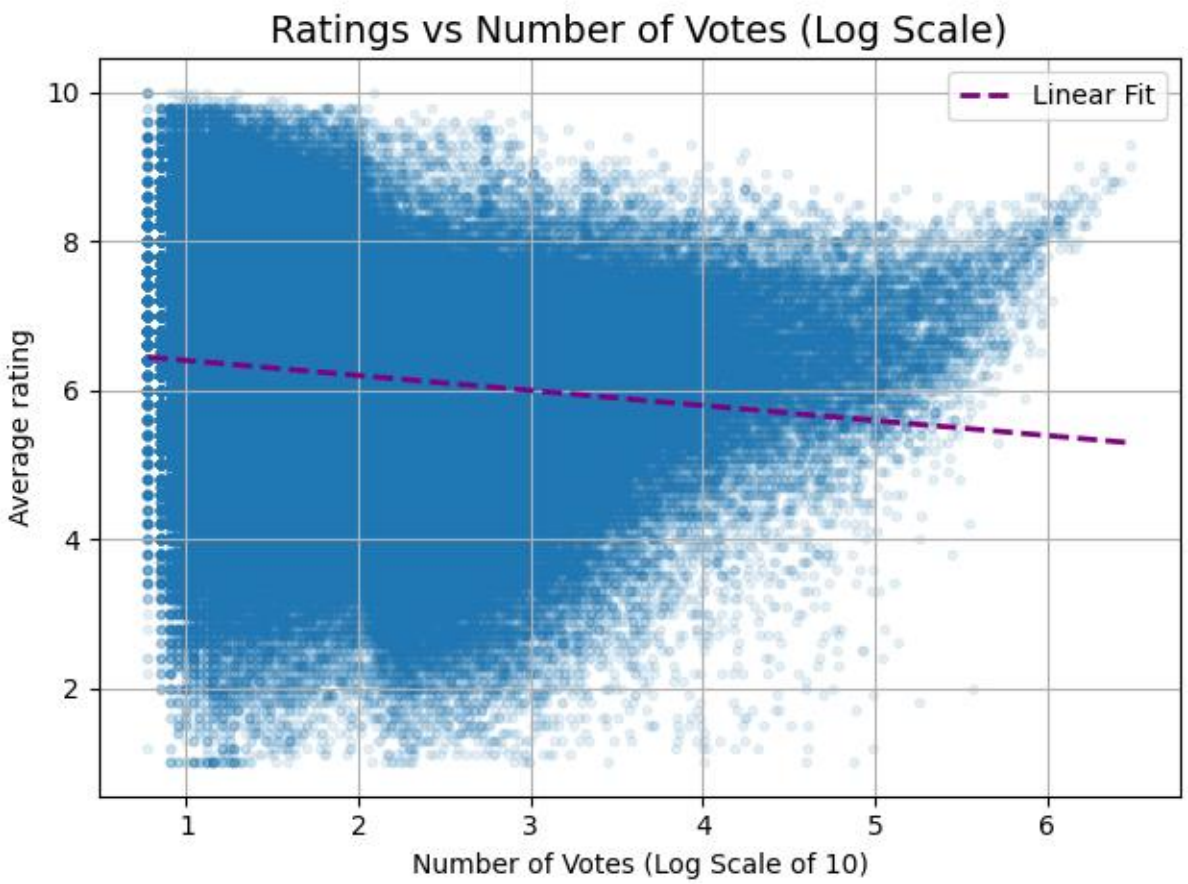


6.4 Linear Regression Looking at The Relationship Between Rating and Number of Votes (contributing to rating)

To end our analysis through visualization methods, we decided to take things further and apply linear regression to our data. IMDB ratings are based on crowd input, where individuals with an IMDb account are able to give their own rating of a movie. Each movie's rating then effectively becomes the average of all individuals who contributed their rating. Our team found this fascinating and wanted to explore whether there was a relationship between the number of votes (a column within our data) and the rating.

Our hypothesis was that movies with a lower number of votes would have higher ratings on average, as these movies could be considered more niche due to lower public interest. Individuals who watch these more niche movies are more likely to have a higher affinity for the film for a specific reason—i.e., they know someone involved or have an appreciation for the director or genre—and thus would be more positively biased in their reviews. In contrast, more popular movies (reflected by a higher number of votes) are subjected to a wider audience and are more likely to receive lower ratings due to less positive bias.

Due to the wide discrepancy in vote counts (some movies having over 100,000 votes and others fewer than 100), we decided to plot the number of votes on a log scale and again determine a line of best fit. A linear regression was then applied to the data using "Ordinary Least Squares" to determine a linear fit. This can be seen below.



Based on this analysis, a negative slope (-0.201) can be seen, which in fact confirmed our original hypothesis that the more votes a movie had, the lower its rating would be on average compared to a movie that had a lower number of contributed votes to its IMDB rating.

However, after statistically determining the R^2 value (also known as the coefficient of determination), this regression returned an R^2 value of 0.017 (which can be found in the terminal of our program when running at the end). This indicates that only 1.7% of the variation in the data can be explained by our regression. It can therefore be said that there is no strong relationship between the number of votes and ratings for movies since 1980—or at least not one that is linear and statistically significant.

7.0 Conclusion

Ultimately, our team really enjoyed working on this project. It gave us a great opportunity to apply skills developed in ENSF 692 to a project of our own design with a real-world application (or at least we believe so). We also got to use GitHub for a collaborative project, which was a first for both of us, and we learned a lot about the process. Work was split up evenly and completed in a timely fashion (for the most part), with constant communication that gave both members confidence and certainty in the success of the project—which we believe we more than achieved!

Our final project is something we are both proud of, and we think it adds a lot of value to the user. Additionally, it meets all the requirements, contains two separate parts—an interactive interface and a statistical analysis—both stemming from a single multi-level DataFrame that we constructed from acquired data.

We hope you enjoyed it and thank you so much for a wonderful semester!

Sincerely,
Marley Cheema & Barrett Sapunjis

8.0 References

[1] “IMDb data files download,” IMDb Data Files Download, https://datasets.imdbws.com/ (accessed Jun. 23, 2025).

[2] ChatGPT

It is important that we reference ChatGPT for our assignment. This became an excellent tool for helping us learn and execute things in our code that we were not familiar with or had limited experience. It was especially beneficial for debugging merges and debugging in general for that matter, finding errors that got us stuck with functions and helping improve our statistical analysis especially in respect to using matplotlib to its full potential, implementing seaborn (which gave our graphs wonderful colors) and helping us learn how to implement a linear regression through python.

However as always, ChatGPT was only used to debug code, provide inspiration and aid when needed but no code generation. All conceptual ideas, architecture and logic along with the physical writing of code for our project come from our own minds.