Data Management Project Report: Movie Analysis

Rowyda Askalani 888387
Nastaran Janialiabadi 922779
Sushil Kumar Chaurasiya 925329

1. Introduction

The movie industry generates vast amounts of data, ranging from audience ratings to financial performance. Understanding what drives higher ratings can help filmmakers and production companies optimize their content.
This project aims to investigate the differences in the evaluation of movies from 2006 to 2016 on the IMDB and TMDB websites by comparing their ratings, metascores, and popularity to understand the relationship between these attributes.

Main research questions

These questions will be explored through queries that analyze how these attributes are correlated and whether there are trends among ratings, metascores, popularity, and revenue.

1) Do Popularity and Metascore Correspond to the Ratings Received?
2) Is There a Correlation Between Ratings, Popularity, and Revenue?
3) Is There a Correlation Between Genres and Metascores?

2. Methodology

Setup

The project begins by importing all the necessary libraries. In particular, the Pandas library is the primary Python library used in this project to facilitate data analysis and improvement.
Next, the drive is mounted and connected to a shared folder for easier acquisition of datasets and seamless exportation to a common drive.

Data Acquisition

To obtain the data for the project, two primary sources were used:

1) IMDB Dataset from Kaggle: The first source was the Kaggle website, where we selected a dataset titled “IMDB data from 2006 to 2016.” This dataset contains the 1,000 most popular movies on IMDB from 2006 to 2016. The data points included are:
    a) Title, Genre, Description, Director, Actors, Year, Runtime, Rating, Votes, Revenue, Metascore
2) TMDB API Data: The second source was the TMDB website’s API. Data acquisition for this source was implemented in Python, where we

fetched the most popular movies from the same year range as the IMDB dataset for better comparison and integration. The steps taken to acquire the data are as follows:

a) Initialization:
    i) The file name was defined for exportation.
    ii) The API key was used to retrieve and access data from TMDB.
    iii) The base URL from which the data was extracted was set.
b) Fetching Movie Data:
    i) The fetch_movies function was used to request movie data in batches of 20 movies per page from the TMDB API, as this was the request limit per batch.
    ii) The function retrieved movies based on a release date range (2006–2016) to match the IMDB dataset and sorted them by popularity in descending order.
    iii) A total of 2,000 movies were collected to increase the likelihood of matching with IMDB data.
c) Fetching Additional Movie Details:
    i) The fetch_additional_details function retrieved metadata for each movie, including:
        (1) Genres: Genre IDs were matched with corresponding names for easier analysis.
        (2) Runtime: The total duration of the movie.
        (3) Revenue: Converted into millions for consistency and simpler analysis.
        (4) Language: The original language of the movie. Unlike genres, this attribute required retrieval of a CSV file from the website to match abbreviations with their English names.
d) Fetching Cast and Crew Information
    i) The fetch_movie_credits function extracted key personnel involved in each film:
        (1) Director: Retrieved from the movie's crew list.
        (2) Top 5 actors: Collected from the cast list.
e) Extract Movie Data
    i) Data extracted from all pages included:
        (1) Rank, ID, Title, Genre, Description, Director, Actors, Runtime in Minutes, Rating, Votes, Revenue in Millions, Original Title, Popularity, Release Year, Language
f) Data Storage
    i) All extracted movies were stored in a Pandas DataFrame and then exported as a CSV file (tmdb_movies_with_details.csv) for further analysis. Storing the data in the drive ensured easier access and eliminated the need to rerun the API every time data retrieval was required.

Data Pre-integration

To begin the pre-integration process, we first standardized the column names in both datasets to ensure consistency and prevent integration issues. In the IMDB dataset, we renamed the columns "Runtime (Minutes)" and "Revenue (Millions)" to "Runtime_Minutes" and "Revenue_Millions," respectively.

In the TMDB dataset, a more extensive renaming was performed to harmonize the data. Specifically, we updated the following columns:

- id → ID
- original_title → Original_Title
- Runtime (Minutes) → Runtime_Minutes
- Revenue (Millions) → Revenue_Millions
- popularity → Popularity
- release_year → Year
- language → Language

After ensuring that the column names were consistent across both datasets, we created a new column called “Title_Year” in each dataset. This column serves as a unique key for merging by combining the movie title and its release year. The process involved converting the movie title to lowercase, stripping any extra whitespace, and concatenating it with the release year converted to a string using an underscore as the delimiter. This method helps differentiate movies with similar titles that were released in different years, thereby facilitating an accurate merge between the IMDB and TMDB datasets.

Data Integration

The integration process begins by leveraging the newly defined “Title_Year” column present in both datasets. Using the Pandas library’s merge function, we performed an inner join on this key to retain only the movies common to both IMDB and TMDB. To clearly distinguish columns with matching names across the two sources, we appended the suffixes “_imdb” and “_tmdb” respectively. This merging step not only aligns the datasets but also enriches the IMDB data by incorporating additional details from TMDB, thereby deepening the dataset for further analysis.

To avoid duplicate columns and ensure the completeness of our data, we then built a new integrated dataset. We started by initializing an empty list called “integrated_rows” to store the final records. Next, we iterated over each row of the merged data frame using a for loop. For each row, we created a dictionary that served as a blueprint for the integrated record.

Within the loop, we assigned a unique ID using the row’s index. We then mapped the “Title” and “Year” columns directly from the IMDB data, using the

get method to extract values reliably. This approach was similarly applied to other straightforward columns such as “Original_Title”, “Rating_imdb”, “Votes_imdb”, “Rating_tmdb”, “Votes_tmdb”, “Metascore”, “Popularity”, and “Language”.

For fields where data availability might differ between the two sources, namely “Runtime_Minutes”, “Description”, “Director”, and “Revenue_Millions”, we used a fallback approach: extracting the value from whichever dataset provided non-null data.

Handling the genre information required a slightly different method. We extracted the genre strings from both datasets, cleaned and split them by commas, and then combined them into a set to eliminate duplicates. From this unified set, we selected the top three genres which is a common practice given the typical focus on a limited number of key genres, and stored them in the “Genre” field. A similar strategy was applied to the “Actors” column, with the difference that we did not restrict the number of actors to capture a fuller cast list.

After processing each row in this manner, we appended the resulting dictionary to the “integrated_rows” list. Once all rows were processed, we converted this list into a new dataframe, replacing the original merged dataframe. Finally, we transformed the “Year” column into a datetime format, focusing solely on the year to simplify subsequent analyses.

This structured approach ensures that our final dataset is coherent, enriched, and primed for deeper analysis without the clutter of duplicate columns.

Data Profiling & Quality Improvement

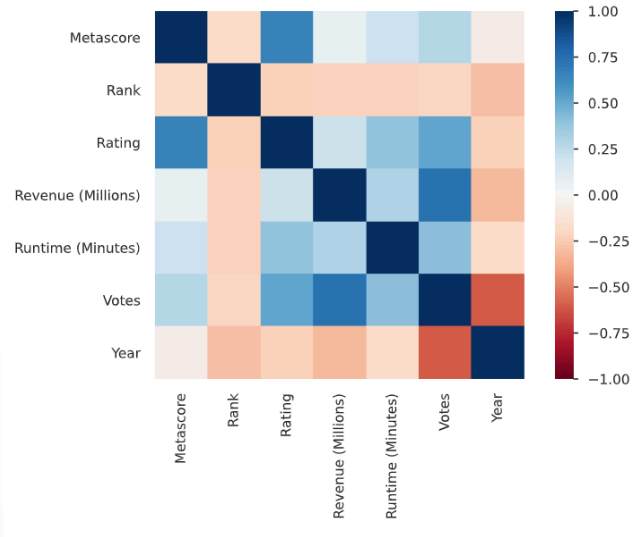
To thoroughly explore the datasets, we used the ydata_profiling library, which provided insights into completeness, uniqueness, and statistical attributes of each column. Profiling was conducted separately on both datasets before integration and then again post-integration to assess data quality.

1) IMDB Dataset Profiling:
    a) We begin by exploring the IMDB dataset, where we discovered missing values in the revenue column, with 128 movies lacking revenue data. To enhance the dataset's quality, we conducted a manual acquisition of the available revenue through online sources, including the IMDB website, The Numbers, and Wikipedia when available. Despite this effort, 35 movies still had missing revenue values, which were addressed post-integration. Additionally, 64 metascores were missing and were similarly handled after integration. Regarding duplicates, we identified two movies that shared the same title; however, they represented different films and were thus retained. Profiling results indicated that only 0.8% of the entire dataset contained missing values, reflecting a relatively high level of data completeness.

b) In the Alerts section, we can notice that there are several variables that are highly correlated with each other indicating that there's a possible relation between them whether negative or positive, that is:

i) The Metascore is highly overall correlated with Rating
ii) The Rating is highly overall correlated with both Metascore and Votes
iii) Revenue is highly overall correlated with Votes
iv) Votes is highly overall correlated with Rating, Revenue and Year
v) Subsequently, Year is highly overall correlated with Votes

This can be further observed through the Interaction section or through the Heatmap or Table created through profiling which is displayed below:



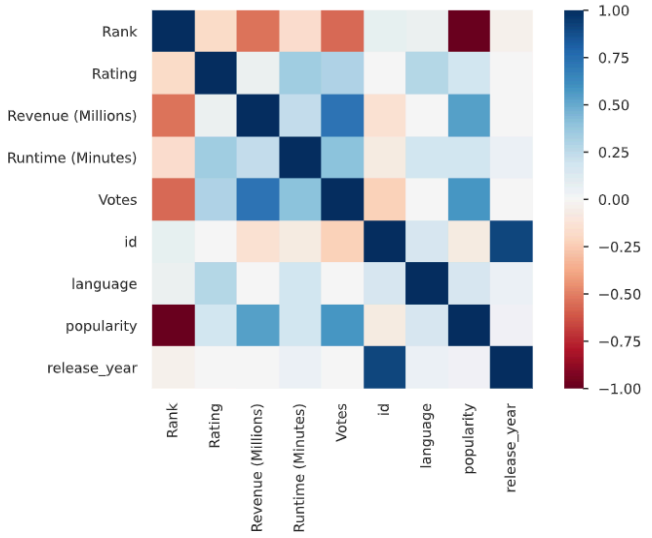
2) TMDB Dataset Profiling:

a) Next, we explore the TMDB dataset and observe that the percentage of missing cells is nearly double that of the IMDB dataset, reaching 1.5% of the total number of cells. A detailed analysis of the columns containing missing values reveals that five attributes are affected: revenue (438 missing values), genre (5 missing values), description (2 missing values), director (3 missing values), and actors (1 missing value). Additionally, we identify 9 pairs of movies that share the same title, leading to potential duplication issues. These missing values and duplicates were addressed post-integration to enhance data consistency and completeness.

b) In the Alerts section, we can notice that there are several variables that are highly correlated with each other indicating that there's a possible relation between them whether negative or positive, that is:

i) The Revenue is highly overall correlated with both Votes and Popularity
ii) Votes is highly overall correlated with Revenue and Popularity
iii) Popularity is highly overall correlated with Revenue and Votes

This can be further observed through the Interaction section or through the Heatmap or Table created through profiling which is displayed below:



Another key observation from the Alerts Section is the imbalance in the Language variable. Our analysis reveals that 72.1% of the movies in the merged dataset are in English, suggesting a potential spike in English-language films during the given year range.

3) Merged Dataset Profiling:

To assess data loss resulting from the integration, we analyzed how many movies from IMDB did not find a corresponding match in TMDB. Our findings revealed that 273 titles from the IMDB dataset remained unmatched, while 1,265 titles from the TMDB dataset were not linked. This indicates a greater data loss on the TMDB side.

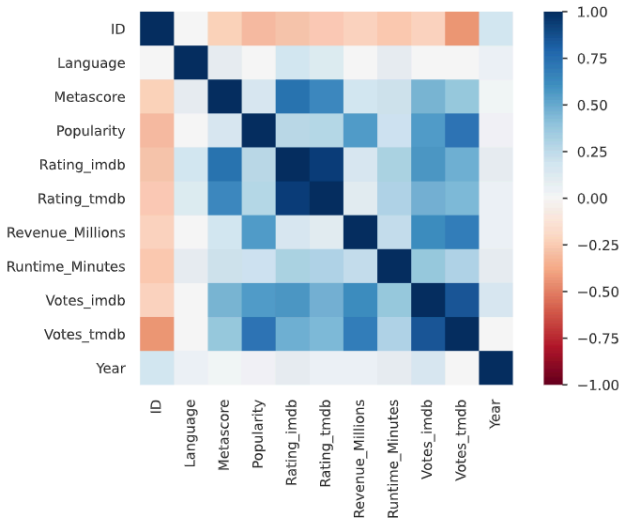
We also examined the presence of missing values, with particular focus on the Revenue and Metascore columns. The analysis showed that Revenue had 5 missing values, while Metascore was missing 24 data points. To address the missing values in the Revenue column, we first calculated both the mean and median, then analyzed the distribution of values. The data exhibited left skewness, with a significant difference between the mean and the median. Given this skewness, we opted to use the mean to fill in the missing values. A similar approach was applied to the Metascore column, but in this case, the difference between the

mean and median was minimal. Consequently, we filled the missing Metascore values using the integer form of the mean. Additionally, we investigated the presence of duplicate movie titles in the merged dataset. Our analysis identified two pairs of duplicate titles. However, upon closer inspection, we determined that these pairs referred to entirely different movies rather than actual duplicates. As a result, no entries were removed from the dataset.

a) In the Alerts section, we can notice that there are several variables that are highly correlated with each other indicating that there's a possible relation between them whether negative or positive, that is:

   i) Metascore is highly overall correlated with Rating_imdb and Rating_tmdb
   ii) Popularity is highly overall correlated with Revenue_Millions, Votes_imdb, and Votes_tmdb
   iii) Rating_imdb is highly overall correlated with Metascore, Rating_tmdb, Votes_imdb
   iv) Rating_tmdb is highly overall correlated with Metascore, Rating_imdb
   v) Revenue_Millions is highly overall correlated with Popularity, Votes_imdb, and Votes_tmdb
   vi) Votes_imdb is highly overall correlated with Popularity, Ratin_imdb, Revenue_Millions, and Votes_tmdb
   vii) Votes_tmdb is highly overall correlated with Popularity, Revenue_Millions, and Votes_imdb

   This can be further observed through the Interaction section or through the Heatmap or Table created through profiling which is displayed below:



   Another key observation from the Alerts Section is the imbalance in the Language variable. Our analysis shows that 93.6% of the movies in the merged dataset are in English, suggesting that a

significant number of foreign-language films were lost during the integration process.

Data Exportation

The final cleaned version of the merged dataset was then exported in CSV format for further analysis through MySQL application.

Data storage and Queries

The cleaned dataset was stored in MySQL for efficient querying and management. This allowed for further exploration of the data, answering research questions, and uncovering potential trends. MySQL was chosen for its structured format, efficient data retrieval, and tabular structure.

Database and Table Creation

The following steps were taken in order to create the database and store the data that was obtained from the colab notebook into tables:

1. To house the integrated movies dataset, we first created a dedicated database. This step ensures that all related tables and queries are contained within a single, well-organized schema. The database was created using the following SQL commands:
    a. CREATE DATABASE IF NOT EXISTS movies_db;
       USE movies_db;

    The CREATE DATABASE IF NOT EXISTS movies_db; command creates a new database named movies_db if it does not already exist. The USE movies_db; command then sets this database as the context for subsequent operations.
2. Next, we established a table to store the dataset by creating a table named Movies. The table schema was designed to reflect the structure of the final integrated dataset, with each column assigned an appropriate data type. The id column was set as the primary key to ensure that each record is uniquely identifiable. The table was created with the following SQL commands:
    a. CREATE TABLE IF NOT EXISTS Movies (
       id INT AUTO_INCREMENT,
       Title VARCHAR(255),
       Genre VARCHAR(255),
       Description TEXT,
       Director VARCHAR(255),
       Actors TEXT,
       Year INT,
       Runtime_Minutes INT,
       Rating_imdb FLOAT,
       Votes_imdb INT,
       Revenue_Millions FLOAT,
       Metascore FLOAT,

```
    Rating_tmdb FLOAT,
    Votes_tmdb INT,
    original_title VARCHAR(255),
    popularity FLOAT,
    language VARCHAR(50),
    PRIMARY KEY (id)
);
```

This query defines the structure of the Movies table. It includes the various attributes each with a data type that matches the nature of the data. The id column is auto-incremented and designated as the primary key to uniquely identify each entry.

3. Once the table was ready, the next step was to import the dataset from a CSV file stored on our local machine. To avoid potential importation issues, such as misinterpreting column headers, the first row, which contains the column names, was skipped. The data importation was performed with the following SQL commands:
    a.
```
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/Final_df02.csv'
INTO TABLE Movies
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS
(id, Title, Genre, Description, Director, Actors, Year, Runtime_Minutes, Rating_imdb, Votes_imdb, Revenue_Millions, Metascore, Rating_tmdb, Votes_tmdb, original_title, popularity, language);
```

The LOAD DATA INFILE command loads data from the specified CSV file into the Movies table. The parameters ensure that fields are correctly separated by commas, text is enclosed in quotes, and lines are appropriately terminated. The IGNORE 1 ROWS clause skips the header row to prevent it from being imported as data.

4. To analyze the relationship between genres and other variables more effectively, we decided to decompose the composite Genre column into three distinct columns: “Genre_1”, “Genre_2”, and “Genre_3”. Each new column represents an individual genre extracted from the original, comma-separated list. This is done through 2 Steps:
    a. We first altered the Movies table to add the new columns through this SQL commands:
```
ALTER TABLE movies
ADD COLUMN Genre_1 VARCHAR(255) NULL,
ADD COLUMN Genre_2 VARCHAR(255) NULL,
ADD COLUMN Genre_3 VARCHAR(255) NULL;
```

This query adds three new columns to the Movies table. These columns are set to allow NULL values in cases where fewer than three genres are present for a movie.

b. After adding the columns, we updated the table to populate them by extracting individual genres using string functions and using this SQL commands:
```
UPDATE movies
SET
   Genre_1 = SUBSTRING_INDEX(Genre, ',', 1),
   Genre_2 =
NULLIF(SUBSTRING_INDEX(SUBSTRING_INDEX(Genre, ',', 2), ',', -1), Genre_1),
   Genre_3 = NULLIF(SUBSTRING_INDEX(Genre, ',', -1), Genre_2);
```

1. “Genre_1”: Extracts the first genre from the comma-separated list using SUBSTRING_INDEX(Genre, ',', 1).
2. “Genre_2”: Extracts the second genre by first obtaining the first two genres with SUBSTRING_INDEX(Genre, ',', 2), and then isolating the second genre. The NULLIF function prevents duplicate entries if the second genre is identical to the first.
3. “Genre_3”: Similarly extracts the third genre from the list. The use of NULLIF ensures that if the third genre is identical to the second, it is set to NULL.

Queries and Research

After defining the tables and successfully loading the data, the next phase in our methodology involves executing a series of SQL queries to address our initial research questions. These queries are crafted to analyze the trends and patterns present within the refined database, enabling us to gain valuable insights into the dataset. By interrogating the data through targeted queries, we can explore relationships between variables, identify significant trends, and ultimately answer the research questions that motivated this project.

1. Research Question 1: Do Popularity and Metascore Correspond to the Ratings Received?
   To explore whether a movie's popularity and metascore are indicative of its ratings, we executed the following query:
   a.
```
SELECT title, rating_imdb, rating_tmdb, popularity, metascore
FROM movies
ORDER BY popularity DESC
LIMIT 10;
```
   We obtain the following results:

title	rating_imdb	rating_tmdb	popularity	metascore
Interstellar	8.6	8.4	321.485	74
Fifty Shades of Grey	4.1	5.881	159.473	46
The Maze Runner	6.8	7.187	151.891	57
The Dark Knight	9	8.5	151.801	82
The Avengers	8.1	7.7	137.126	69
The Wolf of Wall Street	8.2	8	129.295	75
Harry Potter and the Deathly Hallows: Part 1	7.7	7.7	126.877	65
The Equalizer	7.2	7.282	126.167	57
Ratatouille	8	7.826	125.767	96
Inside Out	8.2	7.912	122.429	94

- The results revealed notable discrepancies between ratings across different platforms. For example, the second most popular movie “Fifty Shades of Grey” registered an IMDB score of 4.1 and a TMDB score of 5.8, alongside a low metascore of 46. Despite its high popularity, its ratings vary significantly, suggesting that popularity does not always align with critical acclaim or audience evaluation. Another instance is The Maze Runner presented in the third row, which exhibits high popularity, yet its metascore of 57 contrasts with its IMDB and TMDB ratings of 6.8 and 7.18, respectively. These findings indicate that ratings, popularity, and metascore can sometimes diverge, emphasizing the complexity of audience and critic responses.

2. Research Question 2: Is There a Correlation Between Ratings, Popularity, and Revenue?
   To investigate the potential correlations among ratings, popularity, and revenue, we executed the following query:
   a. SELECT
      ROUND(CORR(rating_imdb, revenue_millions), 2) AS correlation_rating_revenue,
      ROUND(CORR(popularity, revenue_millions), 2) AS correlation_popularity_revenue,
      ROUND(CORR(rating_imdb, popularity), 2) AS correlation_rating_popularity
      FROM movies;

   We obtain the following results:

correlation_rating_revenue	correlation_popularity_revenue	correlation_rating_popularity	correlation_tmdb_rating_revenue	correlation_tmdb_rating_popularity	correlation_tmdb_rating_imdb
0.22	0.54	0.29	0.19	0.34	0.93

- The analysis revealed a high correlation between popularity and revenue, suggesting that movies with greater popularity tend to generate higher revenue. In contrast, the correlations between IMDB rating and revenue as well as between IMDB rating and popularity were found to be low. This implies that while a film’s popularity might drive its financial success, it does not necessarily translate to high ratings. Similarly, the TMDB ratings exhibited low correlations with both popularity and revenue. Notably, however, there was a strong correlation of 0.93 between TMDB and IMDB ratings, indicating that despite platform differences, audience preferences tend to be consistent across different rating systems.

3. Research Question 3: Is There a Correlation Between Genres and Metascores?
   To assess how genres may influence critical acclaim as measured by metascores, we ran the following query:
   a. SELECT genre_1 AS genre, AVG(metascore) AS avg_metascore
      FROM movies
      WHERE genre_1 IS NOT NULL
      GROUP BY genre_1
      ORDER BY avg_metascore DESC;

   We obtain the following results:

genre	avg_metascore
Animation	71.96
Biography	68.9
History	68.49
Musical	65.67
Drama	65.45
War	64.38
Sport	62.18
Music	61.4
Western	61.38
Mystery	59.35
Family	59.34
Sci-Fi	59.21
Science Fiction	58.44
Adventure	58.28
Crime	58.27
Horror	57.98
Thriller	57.54
Comedy	57.33
Fantasy	56.73
Romance	54.67
Action	54.6

   - Based on these results, animation movies have the highest average metascore, followed by biography and historical films. In contrast, the worst-performing genres, with the lowest average metascores, are action, romance, and fantasy movies. This is expected, as action films often aim to be blockbusters focused on commercial success rather than critical acclaim.

3. Possible improvements

To enhance the project, the following improvements could be made:

1. Optimizing the integration process to retain as much information as possible, particularly from the TMDB dataset, which experienced greater data loss during merging. While this approach would likely result in a higher number of missing values, these could be addressed using data imputation techniques such as mean/median substitution or more advanced machine learning methods to predict and fill in missing values. This refinement would increase the completeness of the dataset and improve the reliability of subsequent analyses.
2. Additionally, this approach would help mitigate the imbalance in the Language variable, where a high percentage of movies in English suggests potential data loss for foreign-language films. By preserving more TMDB data, particularly from international films, the dataset would become more linguistically diverse, reducing biases toward English-language movies. This

increased diversity would, in turn, expand the geographical scope of the analysis, enabling deeper insights into global film trends, cross-cultural comparisons, and industry patterns beyond English-speaking markets.

4. Conclusions

In conclusion, our analysis of movies from 2006 to 2016 using data from IMDB and TMDB provided clear insights into the relationships between ratings, popularity, revenue, and genres. We observed that high popularity often leads to increased revenue, while critical scores and audience ratings can vary. The study also showed that animation, biography, and historical films generally receive more favorable reviews than action, romance, and fantasy films. These findings offer valuable guidance for filmmakers and production companies seeking to optimize their strategies. Future work should focus on reducing data loss and including a broader range of international films.