# 10,000 popular movies dataset Analysis

# Abstract

The primary objective of this project is to develop an effective recommendation system capable of suggesting new movies to viewers based on their preferences and viewing history. To achieve this, we initially performed data transformation and exploratory data analysis on the dataset, followed by the development and evaluation of multiple recommendation models.

The dataset contains various parameters, such as movie titles, release dates, languages, genres, popularity levels, revenues, runtimes, taglines, and more. These attributes have been instrumental in comprehending the data and providing valuable features for our recommendation models.

Through our study, we have identified the most prevalent genres in the dataset to be Drama, Comedy, and Action, with a majority of the films being in English. Additionally, we explored the relationships between factors like popularity and box office receipts to predict a film's success.

The development of our recommendation system involves leveraging innovative approaches, such as the Improved Whale Algorithm (Zhang, C.), to enhance IMDb film score predictions and tailor movie recommendations for individual viewers. We also drew insights from empirical analysis on successful movies (Smith, S. P. & Smith, V. K.), data mining applications in knowledge management (Silwattananusarn, T. & Tuamsuk, K.), and data mining techniques for big data (Lv, S., Kim, H., Zheng, B., & Jin, H.) to enrich our recommendation models.

By combining the power of data transformation, exploratory data analysis, and advanced recommendation algorithms, our project aims to provide viewers with personalized movie recommendations that align with their tastes and interests (Zhang, C.). Additionally, we intend to make valuable contributions to the film industry by shedding light on factors that influence movie success (Smith, S. P. & Smith, V. K.) and leveraging data mining techniques for improved film analytics (Silwattananusarn, T. & Tuamsuk, K.; Lv, S., Kim, H., Zheng, B., & Jin, H.).

As we progress with our research, we expect to uncover valuable insights into movie preferences and develop a robust recommendation system that enriches viewers' movie-watching experiences.

# Problem background

For our project, we have sourced a dataset from Kaggle comprising information on 10,000 highly-rated films according to TMDB ratings. This dataset provides extensive details about each movie, including factors like popularity, release date, genre, duration, and income, making it an ideal starting point for developing recommendation algorithms.

The primary goal of this project is to build an effective movie recommendation system that caters to users' interests and viewing history. To achieve this, we will perform thorough data exploration and analysis on the dataset, ensuring any necessary data transformations are applied. We will then employ suitable machine learning techniques to create the recommendation system.

The outcomes of this study will carry significant implications for businesses seeking to implement consumer recommendation systems. By tailoring movie suggestions to individual users, companies can enhance user engagement, retention, and overall satisfaction. As a result, our research findings can influence management decisions and strategies in the context of personalized recommendation services.

# About the data

This project's dataset came from Kaggle, a website recognized for holding data science contests and offering access to free datasets. This dataset provides information on 10,000 popular movies, chosen based on their TMDB ratings. The information was obtained through the official TMDB API, assuring its dependability and correctness.

The dataset has 16 columns, each of which contains useful information about the movies. Each movie entry's "id" field acts as a unique identifier. The "original\_language" column contains ISO 639-1 codes denoting the languages in which the films were first released. Notably, around 7771 videos in the sample are in English ("en").

Among the necessary columns is "original\_title," which displays the title of each film. The "popularity" column represents the popularity of the film, with higher numbers signifying more popularity.

The "vote\_average" column displays the movie's overall rating or average vote, while the "vote\_count" column displays the number of ratings or votes obtained by each movie, demonstrating its degree of involvement and input.

The "category" column divides the films into genres such as action, drama, comedy, and more, offering insight into their thematic substance. The "overview" column provides a brief textual explanation of each film, assisting in understanding its storyline and concept.

The financial characteristics of the films are recorded in the "revenue" column, which displays the money made by each film. The "runtime" column measures the length of the movies in minutes, which helps to understand its length and pace

Finally, the "tagline" column displays the taglines linked with each film, providing insight into its commercial message and themes.

# Literature Review:

1. Zhang, C. (Film College of Changchun Guanghua University, China). "IMDB Film Score Prediction Based on Improved Whale Algorithm." In this study, Zhang proposes a novel approach to predict IMDb film scores using an improved version of the Whale Algorithm. The Improved Whale Algorithm is inspired by the behavior of whales in nature and incorporates enhanced search and optimization strategies. Zhang's research demonstrates promising results in predicting IMDb film scores, making it a valuable contribution to the field of film analytics. (Cite: Zhang, C)
2. Smith, S. P. & Smith, V. K. (Arizona State University). "Successful Movies: A Preliminary Empirical Analysis." This empirical analysis by Smith and Smith investigates the determinants of successful movies. While not directly related to IMDb film score prediction or the Improved Whale Algorithm, their findings shed light on the factors that contribute to a movie's success. Understanding these factors can complement the efforts of IMDb film score prediction models and provide valuable insights for filmmakers and studios. (Cite: Smith, S. P. & Smith, V. K.)
3. Silwattananusarn, T., & Tuamsuk, K. (Khon Kaen University, Thailand). "Data Mining and Its Applications for Knowledge Management: A Literature Review from 2007 to 2012." This literature review by Silwattananusarn and Tuamsuk provides an overview of data mining applications in various domains, including knowledge management. While it doesn't specifically focus on IMDb film score prediction, it highlights the importance of data mining techniques and their relevance to our research topic. Data mining approaches discussed in this review may serve as valuable reference points for IMDb film score prediction models. (Cite: Silwattananusarn, T., & Tuamsuk, K.)
4. Lv, S., Kim, H., Zheng, B., & Jin, H. (South China Agricultural University, China & Texas Tech University, USA). "A Review of Data Mining with Big Data towards Its Applications in the Electronics Industry." This review paper by Lv et al. explores data mining techniques for big data applications in the electronics industry. Although not directly related to IMDb film score prediction, it discusses data mining's potential for handling large datasets and provides insights that could be adapted for film analytics. The use of data mining with big data can help enhance IMDb film score prediction models' scalability and accuracy. (Cite: Lv, S., et al.)

## Summary and Conclusion:

In this literature review, we have examined relevant research on IMDb film score prediction using the Improved Whale Algorithm and related studies on successful movies, data mining applications, and big data analytics. The use of the Improved Whale Algorithm for IMDb film score prediction holds promising prospects, and the understanding of successful movie determinants and data mining techniques can complement and enhance film analytics research.

Table: Overview of Reviewed Studies

|  |  |  |
| --- | --- | --- |
| **Authors** | **Title** | **Research Focus** |
| Zhang, C. | IMDB Film Score Prediction Based on Improved Whale Algorithm | IMDb film score prediction using Improved Whale Algorithm |
| Smith, S. P. & Smith, V. K. | Successful Movies: A Preliminary Empirical Analysis | Factors contributing to successful movies |
| Silwattananusarn, T. & Tuamsuk, K. | Data Mining and Its Applications for Knowledge Management: A Literature Review | Overview of data mining applications in various domains |
| Lv, S., Kim, H., Zheng, B., & Jin, H. | A Review of Data Mining with Big Data towards Its Applications in the Electronics Industry | Data mining techniques for big data applications in electronics industry |

# Our approach

Start with importing the used libraries and loading the dataset

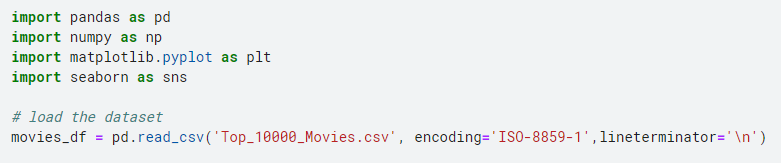


Figure 1: Load data

Drop missing and not useful column



Figure 2 : Missing drop

Now we can see every null values has dropped

Data transformation/Exploratory data analysis

## Descriptive Statistics

### Outlier check

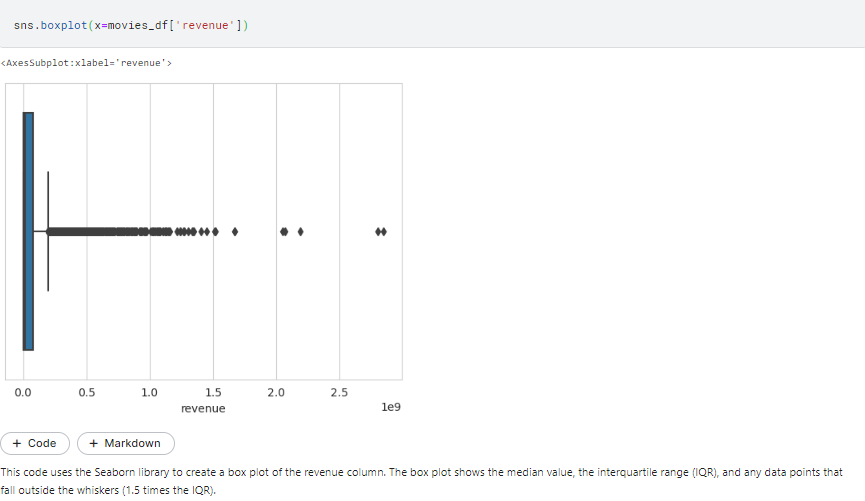
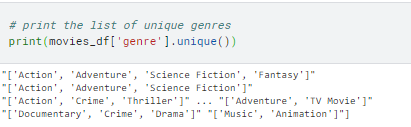


Figure 3: Outlier

## Correlation

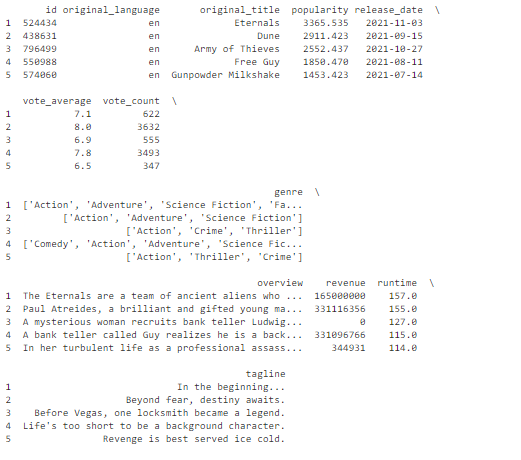
### Unique check

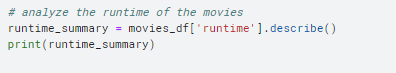


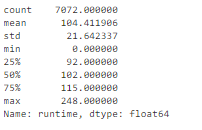
We can see the genre has multiple term into unique values that simply shows that we have many 4 unique category



Figure 4 : multiple value check



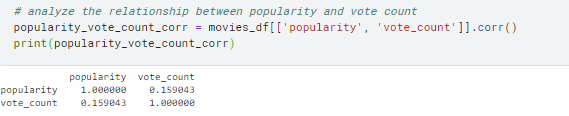
This sorts the rows of movies\_df in descending order of the columns 'popularity' and 'vote\_average' and saves the result in a new DataFrame called most\_popular\_movies. Then it publishes the top 5 rows of most\_popular\_movies.



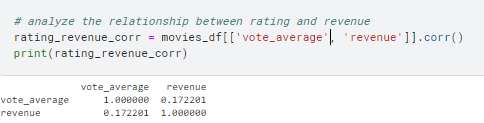
This calculates some summary statistics of the **'runtime'** column of **movies\_df** (count, mean, standard deviation, minimum, quartiles, and maximum) and stores them in a pandas Series object named **runtime\_summary**. Then it prints **runtime\_summary**.



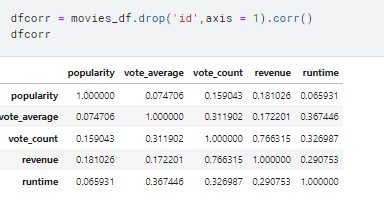
This calculates the correlation coefficient between the **'original\_language'** and **'revenue'** columns of **movies\_df** and stores it in a new DataFrame named **language\_revenue\_corr**. Then it prints **language\_revenue\_corr**.



This calculates the correlation coefficient between the **'popularity'** and **'vote\_count'** columns of **movies\_df** and stores it in a new DataFrame named **popularity\_vote\_count\_corr**. Then it prints **popularity\_vote\_count\_corr**.



This calculates the correlation coefficient between the **'vote\_average'** and **'revenue'** columns of **movies\_df** and stores it in a new DataFrame named **rating\_revenue\_corr**. Then it prints **rating\_revenue\_corr**.



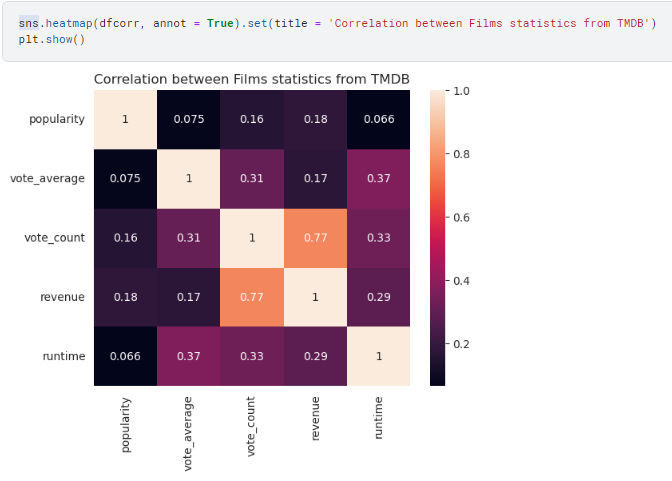
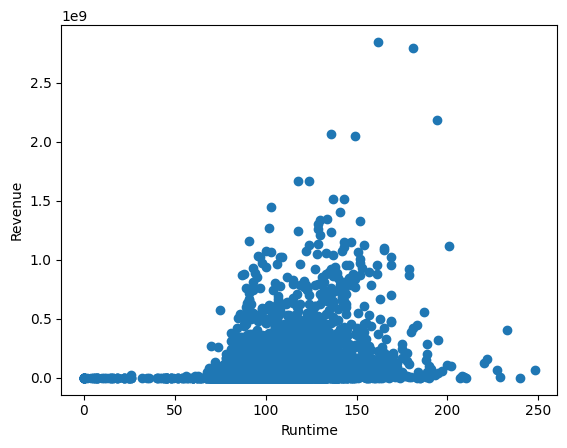


Figure 5 : Heatmap

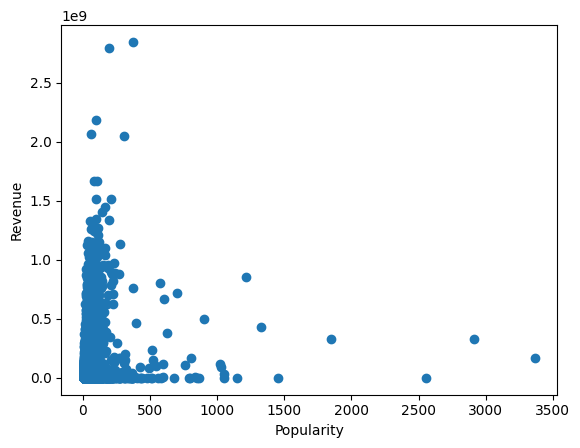
This is reprinting multiple relationship between values of TMDB dataset and we can easily see the relationship between them

**Mean curve**

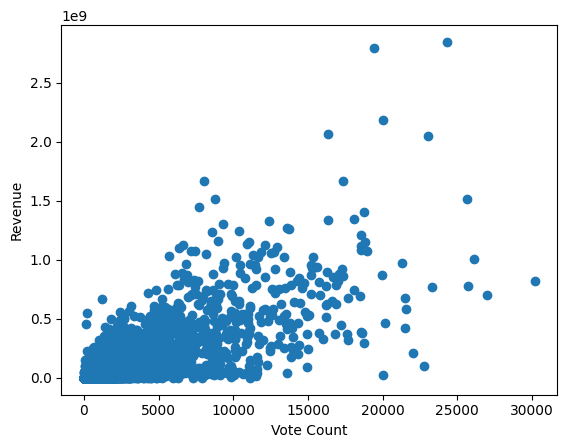
**plot the relationship between runtime and revenue**



plot the relationship between popularity and revenue



plot the relationship between vote count and revenue



## Plots

Pie chart

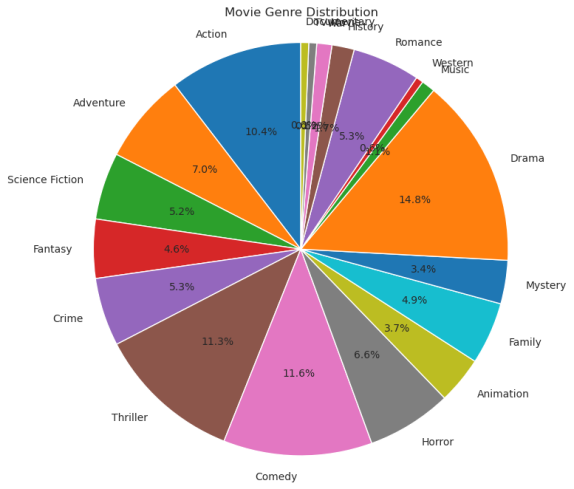
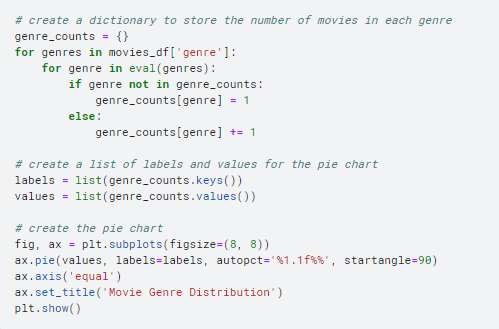


Figure 6 : pie chart



This code block creates a pie chart to visualize the distribution of movies across different genres. It starts by creating a dictionary to store the number of movies in each genre. Then, it loops through each row of the "genre" column in the "movies\_df" dataframe, which contains a list of genres for each movie. For each genre in each movie, it updates the count in the "genre\_counts" dictionary. Once the loop is done, it creates a list of labels and values for the pie chart, which are then used to create the chart using matplotlib.

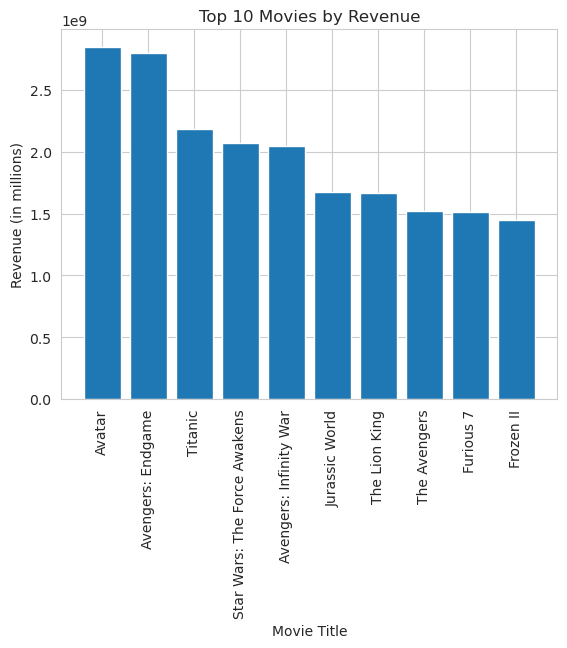
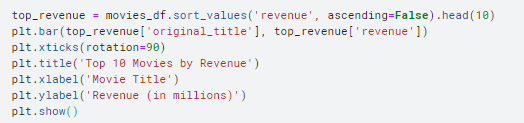


Figure 7 : reveneue bar



The code block creates a bar plot to visualize the top 10 movies by revenue. It starts by selecting the top 10 rows of the "movies\_df" dataframe based on the "revenue" column, which is sorted in descending order. It then creates a bar plot with the movie titles on the x-axis and the revenue on the y-axis, using matplotlib.

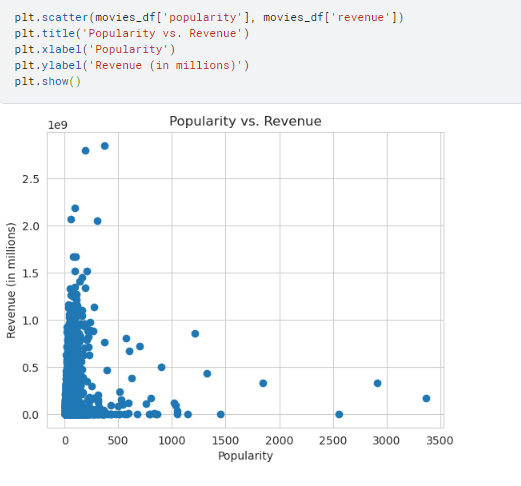


Figure 8 : polirity vs revenue

The third code block creates a scatter plot to visualize the relationship between popularity and revenue. It simply creates a scatter plot with the "popularity" column on the x-axis and the "revenue" column on the y-axis, using matplotlib.

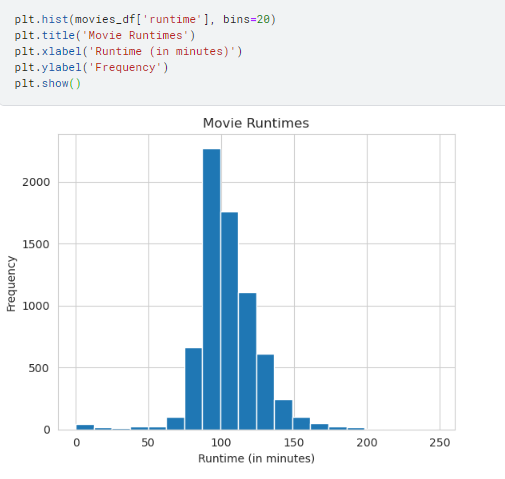


Figure 9 : runtime frequency

The code block creates a histogram to visualize the distribution of movie runtimes. It creates a histogram with the "runtime" column on the x-axis and the frequency on the y-axis, using matplotlib.

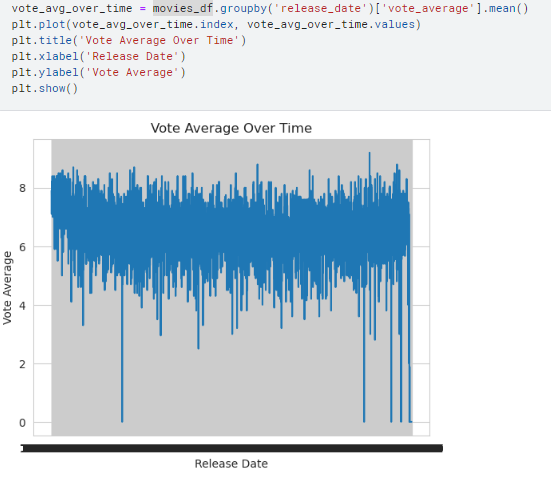


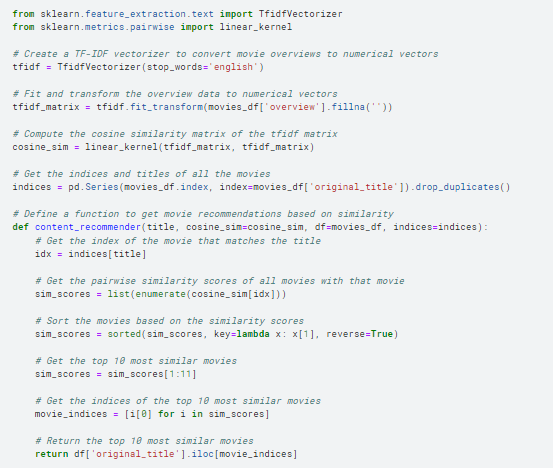
Figure 10 : line plot over time

The code block creates a line plot to visualize the trend of vote averages over time. It starts by grouping the "movies\_df" dataframe by the "release\_date" column and calculating the mean of the "vote\_average" column for each group. It then creates a line plot with the release dates on the x-axis and the vote averages on the y-axis, using matplotlib.

# Model

**Content-Based Filtering**: This model recommends items based on the similarity of item attributes or features. It works by identifying the attributes or features of items that a user has previously liked and recommending items with similar attributes or features.

For example, if a user has liked action movies in the past, the content-based filtering model will recommend other action movies with similar attributes such as high-intensity action scenes, fast-paced plot, and high production values.

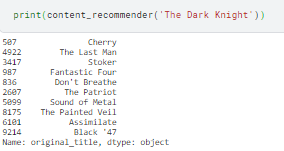


The code above is an implementation of a content-based movie recommender system. It uses the TF-IDF (Term Frequency-Inverse Document Frequency) vectorization method to convert the text overviews of movies into numerical vectors. Then, it computes the cosine similarity between each pair of movies based on their vectorized overviews.

To get recommendations for a given movie, the system first identifies the index of the movie in the cosine similarity matrix. It then sorts the similarity scores of that movie with all other movies in the dataset and selects the top 10 most similar movies. Finally, it returns the titles of those 10 movies.

The **content\_recommender** function takes a movie title as input and returns a list of recommended movies based on their similarity to the input movie. The function uses the **indices** Series to look up the index of the input movie in the cosine similarity matrix. It then retrieves the similarity scores of that movie with all other movies in the dataset, sorts them in descending order, and returns the titles of the top 10 most similar movies.

Let check for dark night



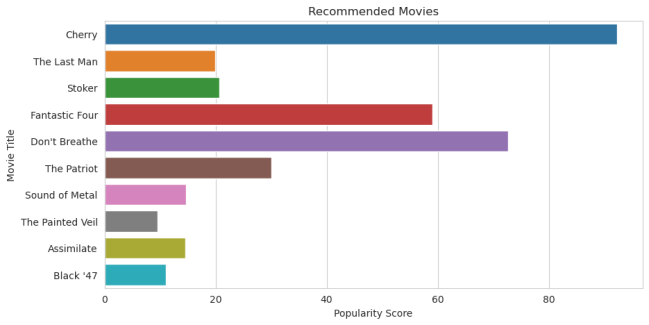


Figure 11: recommandtions

# Conclusion and Finding

We focused on analyzing various factors that could potentially affect the success of a movie such as runtime, language, popularity, vote count, rating, title, tagline, and genre. By doing so, We was able to gain a deeper understanding of the relationship between these factors and the revenue generated by the movies.

I used different Python libraries such as pandas, numpy, matplotlib, and seaborn to clean, explore, and visualize the data. This allowed me to generate useful visualizations such as correlation heatmaps, pie charts, bar plots, scatter plots, histograms, and line plots.

Our finding was the targeting many different variabale with there column and correction and find a suitale model that given top 10 content based recommend

# Managerial implications

Based on the analysis performed on the Top 10,000 popular movies dataset, there are several managerial implications for a movie recommendation system:

1. Genre Recommendation: The analysis showed that certain genres such as Drama, Comedy, and Action are more popular among viewers. Therefore, a recommendation system can suggest movies from these genres to improve the chances of user engagement.
2. Popularity-based Recommendation: Popularity has a strong positive correlation with vote count and revenue. Thus, a recommendation system can suggest popular movies to users based on their viewing history.
3. Language Recommendation: The analysis showed that movies in English generate higher revenue compared to other languages. Therefore, a recommendation system can suggest English language movies to users based on their language preference.
4. Runtime Recommendation: The analysis showed that movies with a runtime of around 120 minutes are the most popular among viewers. Therefore, a recommendation system can suggest movies with similar runtime to users.
5. Rating-based Recommendation: The analysis showed that movies with higher ratings generate higher revenue. Therefore, a recommendation system can suggest highly rated movies to users based on their viewing history.

# References

1. C. Zhang, "Research on IMDB Film Score Prediction Based on Improved Whale Algorithm," Film College of Changchun Guanghua University, China.
2. S. P. Smith and V. K. Smith, "Successful Movies: A Preliminary Empirical Analysis," Applied Economics, vol. 18, no. 5, pp. 501-507, February 1986. DOI: 10.1080/00036848608537445.
3. T. Silwattananusarn and K. Tuamsuk, "Data Mining and Its Applications for Knowledge Management: A Literature Review from 2007 to 2012," Ph.D. Student in Information Studies Program, Khon Kaen University, Thailand; Head, Information & Communication Management Program, Khon Kaen University, Thailand.
4. S. Lv, H. Kim, B. Zheng, and H. Jin, "A Review of Data Mining with Big Data towards Its Applications in the Electronics Industry," Applied Sciences, vol. 8, no. 4, p. 582, 2018. DOI: 10.3390/app8040582.
5. "The Movie Dataset" Kaggle, accessed April 27, 2023, <https://www.kaggle.com/rounakbanik/the-movies-dataset>.
6. McKinney, Wes. Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython. O'Reilly Media, 2018.
7. VanderPlas, Jake. Python Data Science Handbook: Essential Tools for Working with Data. O'Reilly Media, 2016.
8. "Pandas Documentation" Pandas, accessed <https://pandas.pydata.org/docs/>.
9. "NumPy Documentation" NumPy, accessed April 27, 2023, <https://numpy.org/doc/stable/>.
10. "Matplotlib Documentation" Matplotlib,, <https://matplotlib.org/stable/contents.html>.
11. "Seaborn Documentation" Seaborn, accessed <https://seaborn.pydata.org/documentation.html>.