Faculty of Engineering, Natural and Medical Sciences / Department of Information Technologies

IT 323 - DATA MINING PROJECT

Exploratory Data Analysis on IMDB Movies Dataset

Milestone 2

Arnela Sokolić

04/05/2025

Sarajevo

Contents

1. INTRODUCTION

For this milestone, I explored the data to find cool patterns, like which genres are most common or which actors and directors show up the most in top movies. I also looked at how ratings and popularity have changed over time. One thing I found interesting was seeing how some directors had a lot of top-rated movies. I made many charts and graphs to help visualize the data, like bar graphs and pie charts. These made it easier to understand the trends and compare things like genre popularity or rating averages.

2. PERFORMING DATA ANALYSIS

2.1 Distribution of IMDB Ratings

```
#MILESTONE 2 STARTING FROM HERE


import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Set the aesthetic style of the plots
sns.set_style("whitegrid")
```

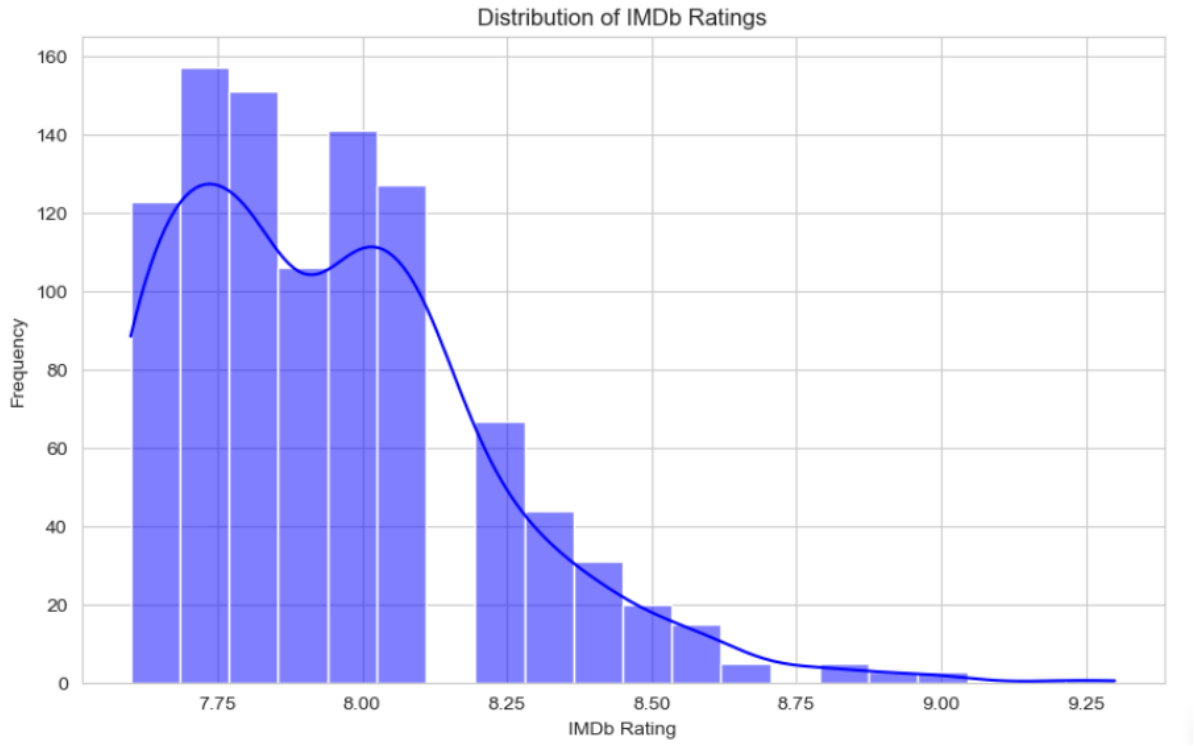
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load dataset
file_path = r"C:\Users\Korisnik\Downloads\archive\imdb_top_1000.csv"
df = pd.read_csv(file_path)

# Set style
sns.set_style("whitegrid")
```

In the first code snippet, I imported the necessary libraries: pandas for data handling, matplotlib.pyplot for creating plots, and seaborn for statistical visualizations. I then set the plot style to "whitegrid" to make graphs more readable with a clean background and grid lines.

In the second code snippet, I repeated the imports but also loaded the IMDb dataset. I specified the file path as a raw string to prevent errors with backslashes and used pd.read_csv(file_path) to read the data into a pandas DataFrame. Finally, I set the "whitegrid" style again to ensure consistency in visualization aesthetics.

```
plt.figure(figsize=(10, 6))
sns.histplot(df['IMDB_Rating'], bins=20, kde=True, color='blue')
plt.title('Distribution of IMDb Ratings')
plt.xlabel('IMDb Rating')
plt.ylabel('Frequency')
plt.show()
```

I created a histogram to show how IMDb ratings are distributed. The bars represent how many movies fall into different rating ranges, and the smooth curve helps visualize the overall trend. I also added labels and a title to make the chart clear, and set the color to blue for better visibility.



Picture 1. Distribution of IMDB Ratings

This graph shows the distribution of IMDb ratings for movies. Most movies have ratings between 7.7 and 8.1, with the highest frequency around these values. The smooth curve shows the general trend, indicating that higher-rated movies (above 8.5) are much less common. The distribution is right-skewed, meaning fewer movies have very high ratings.

2.2 Top 10 highest-rated movies

```
top_rated = df.nlargest(10, 'IMDB_Rating')[['Series_Title', 'IMDB_Rating']]
print(top_rated)
```

This code selects the top 10 highest-rated movies from the dataset based on their IMDb ratings and displays their titles and ratings.

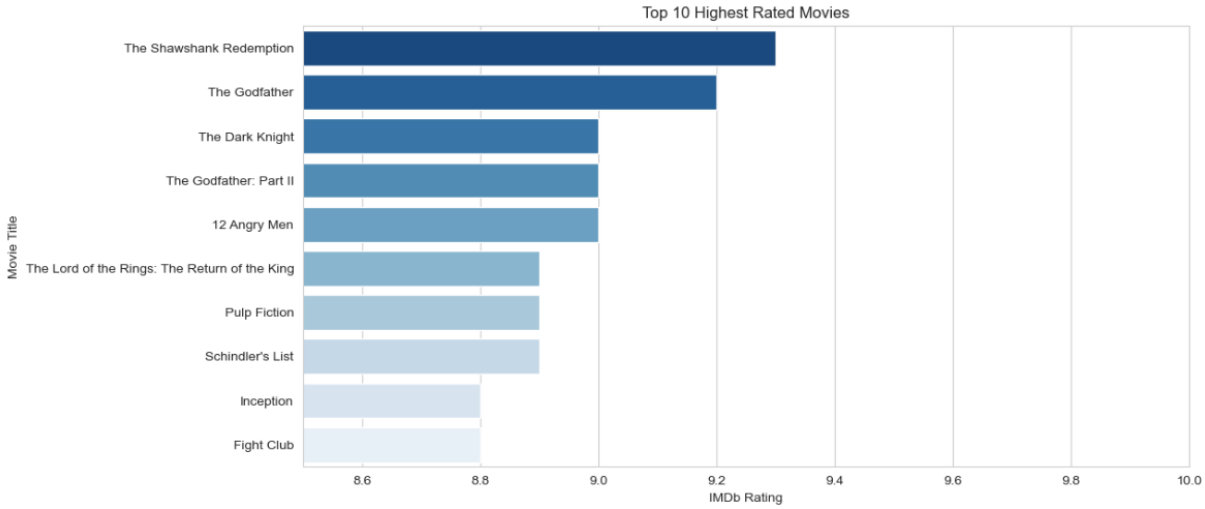
```
                                  Series_Title  IMDB_Rating
0                     The Shawshank Redemption          9.3
1                                The Godfather          9.2
2                              The Dark Knight          9.0
3                      The Godfather: Part II          9.0
4                                  12 Angry Men          9.0
5  The Lord of the Rings: The Return of the King          8.9
6                                  Pulp Fiction          8.9
7                            Schindler's List          8.9
8                                   Inception          8.8
9                                  Fight Club          8.8
```

The output shows the top 10 movies with the highest IMDb ratings, with The Shawshank Redemption ranked highest at 9.3, followed by The Godfather at 9.2. Several other classics, like The Dark Knight, 12 Angry Men, and Pulp Fiction, also appear in the list with high ratings above 8.8.

```
top_rated = df.nlargest(10, 'IMDB_Rating')[['Series_Title', 'IMDB_Rating']]

# Plot top-rated movies
plt.figure(figsize=(12, 6))
sns.barplot(y=top_rated['Series_Title'], x=top_rated['IMDB_Rating'], palette='Blues_r')
plt.title('Top 10 Highest Rated Movies')
plt.xlabel('IMDb Rating')
plt.ylabel('Movie Title')
plt.xlim(8.5, 10)
plt.show()
```

This code selects the top 10 highest-rated movies from the dataset based on their IMDb ratings and then creates a horizontal bar chart to visualize them.

First, it extracts the movie titles and their IMDb ratings. Then, it sets the figure size and uses Seaborn to create a bar plot with the movie titles on the y-axis and their IMDb ratings on the x-axis. The palette='Blues_r' makes the bars appear in shades of blue. The x-axis is limited to a range of 8.5 to 10 to focus on high ratings. Finally, the plot is displayed, making it easy to compare the highest-rated movies visually.



Picture 2. Top 10 Highest Rated Movies

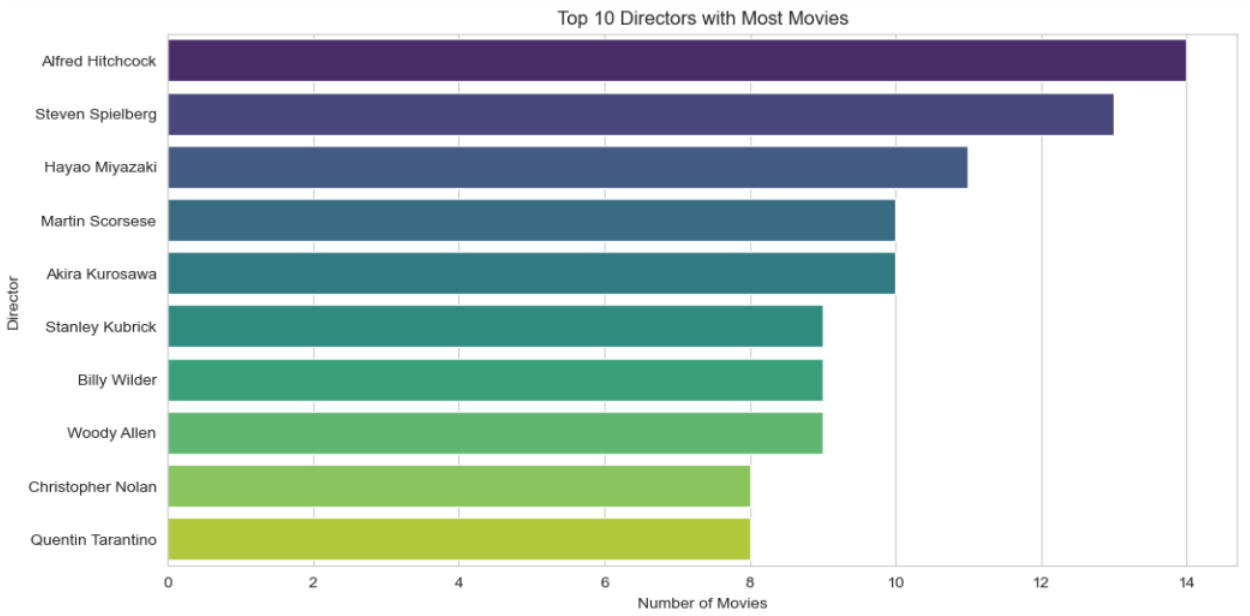
This bar chart shows the top 10 highest-rated movies based on IMDb ratings. "The Shawshank Redemption" has the highest rating, followed closely by "The Godfather" and "The Dark Knight."

The bars represent the IMDb ratings of each movie, with darker shades of blue indicating higher ratings. The x-axis ranges from 8.5 to 10, focusing only on the highest-rated films. This visualization makes it easy to compare the ratings of these top movies briefly.

2.3 Top 10 Directors with Most Movies

```
plt.figure(figsize=(12, 6))
sns.countplot(y=df['Director'], order=df['Director'].value_counts().head(10).index, palette='viridis')
plt.title('Top 10 Directors with Most Movies')
plt.xlabel('Number of Movies')
plt.ylabel('Director')
plt.show()
```

This code creates a bar chart that shows the top 10 directors with the most movies in the dataset. It first counts how many movies each director has and selects the top 10. The bars represent the number of movies directed by each person, with the longest bars showing the most prolific directors. The viridis color palette is used to make the chart visually appealing, and labels are added for clarity. This visualization helps identify which directors have the highest number of films in the IMDb Top 1000 list.



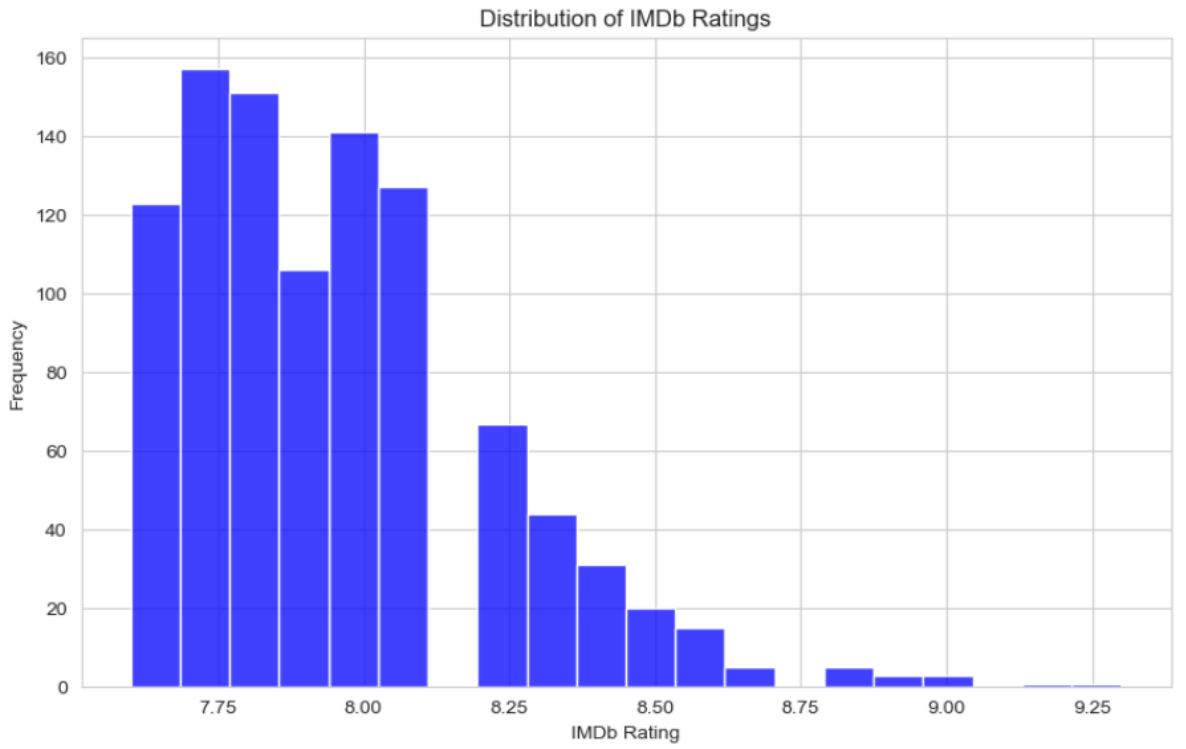
Picture 3. Top 10 Directors with Most Movies

This bar chart shows the top 10 directors with the most movies in the IMDb Top 1000 list. Alfred Hitchcock leads with the highest number of films, followed closely by Steven Spielberg and Hayao Miyazaki. The bars represent the number of movies each director has in the dataset, with directors like Martin Scorsese, Akira Kurosawa, and Christopher Nolan also making the list. The viridis color gradient helps distinguish between different counts, making it easy to compare. This visualization highlights the most influential directors in highly rated cinema.

2.4 Distribution of IMDb Ratings

```
plt.figure(figsize=(10, 6))
sns.histplot(df['IMDB_Rating'], bins=20, kde=False, color='blue')
plt.title('Distribution of IMDb Ratings')
plt.xlabel('IMDb Rating')
plt.ylabel('Frequency')
plt.show()
```

This code creates a histogram to show the distribution of IMDb ratings in the dataset. It divides the ratings into 20 bins (intervals) and counts how many movies fall into each range, displaying the frequency on the y-axis. The histogram is colored blue, and the kde=False setting means no smooth density curve is added. The plot helps visualize how IMDb ratings are spread, showing common rating ranges and whether the distribution is skewed or balanced.



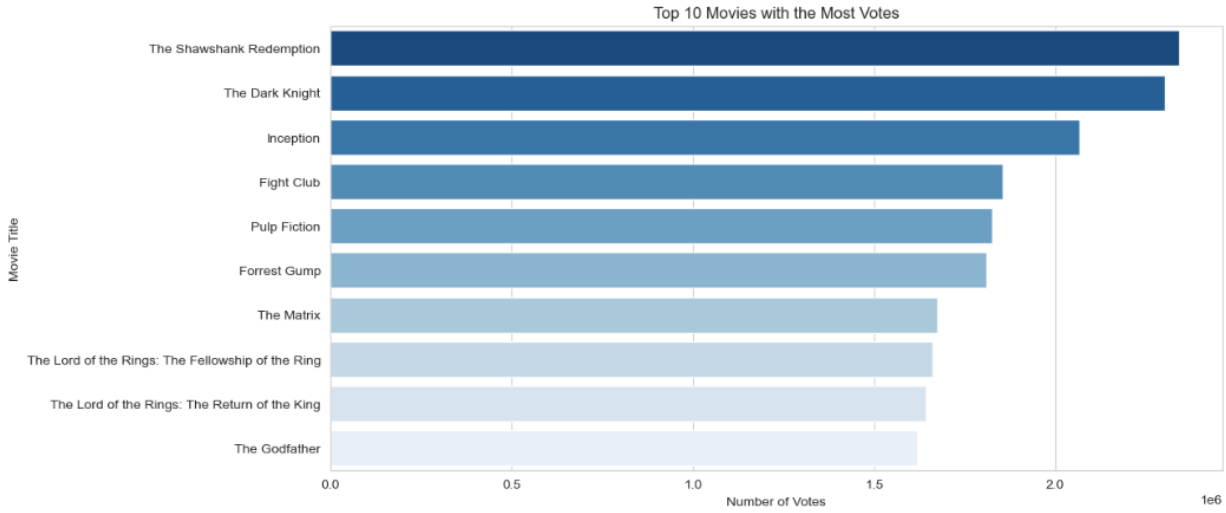
Picture 4. Distribution of IMDB Ratings - 2

This histogram shows how IMDb ratings are distributed. The x-axis represents IMDb ratings, while the y-axis shows how many movies fall into each rating range. Most movies have ratings between 7.5 and 8.2, with fewer movies rated higher than 8.5. The distribution is right-skewed, meaning high ratings (above 9) are rare.

2.5 Top 10 Movies With The Highest Number Of Votes

```
top_voted_movies = df.nlargest(10, 'No_of_Votes')[['Series_Title', 'No_of_Votes']]

plt.figure(figsize=(12, 6))
sns.barplot(y=top_voted_movies['Series_Title'], x=top_voted_movies['No_of_Votes'], palette='Blues_r')
plt.title('Top 10 Movies with the Most Votes')
plt.xlabel('Number of Votes')
plt.ylabel('Movie Title')
plt.show()
```

This code selects the top 10 movies with the highest number of votes from the dataset. It then creates a bar chart where the movie titles are displayed on the y-axis and the number of votes on the x-axis. The bars are colored using a blue palette for better visualization. Finally, the chart is displayed with a title and labeled axes for clarity.



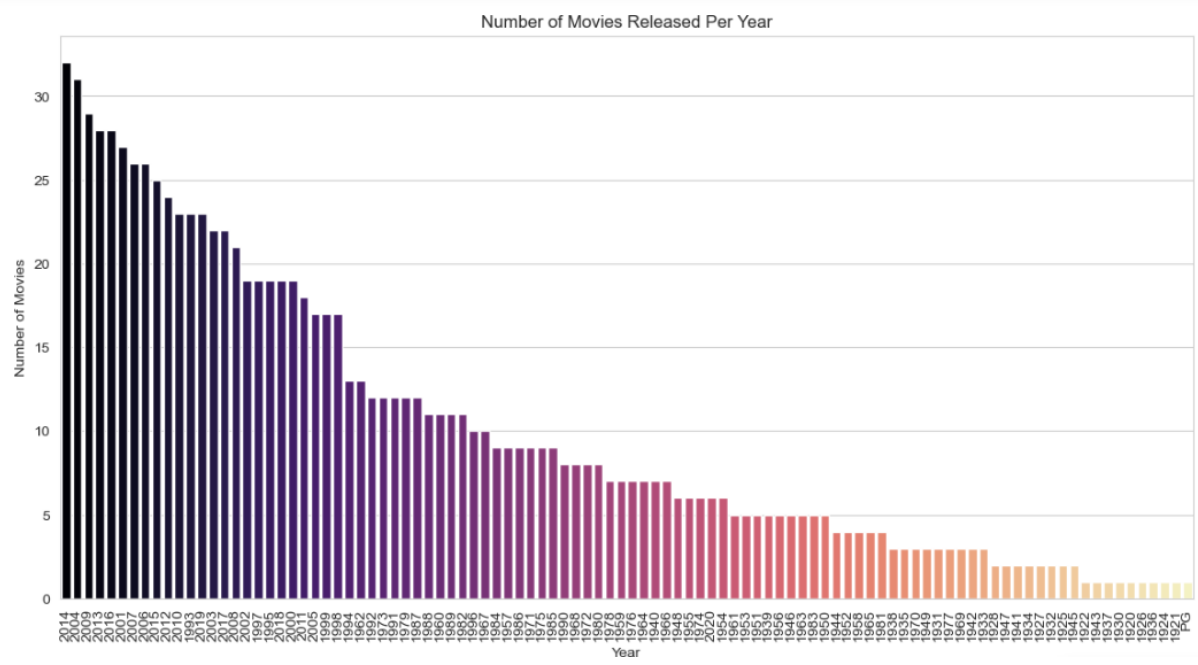
Picture 5. Top 10 Movies with the Most Votes

This bar chart shows the top 10 movies with the highest number of votes. The x-axis represents the number of votes, while the y-axis lists the movie titles. Darker shades of blue indicate movies with more votes. "The Shawshank Redemption" has the most votes, followed by "The Dark Knight" and "Inception."

2.6 Number of Movies Released Per Year

```
plt.figure(figsize=(14, 7))
sns.barplot(x=df['Released_Year'].value_counts().index,
            y=df['Released_Year'].value_counts().values,
            palette='magma')
plt.title('Number of Movies Released Per Year')
plt.xlabel('Year')
plt.ylabel('Number of Movies')
plt.xticks(rotation=90)
plt.show()
```

This code creates a bar chart showing the number of movies released each year. The x-axis represents the release years, and the y-axis represents the number of movies released in each year. The bars are colored using the "magma" palette. The figure size is set to 14x7 for better visibility, and the x-axis labels are rotated 90 degrees to make them easier to read.



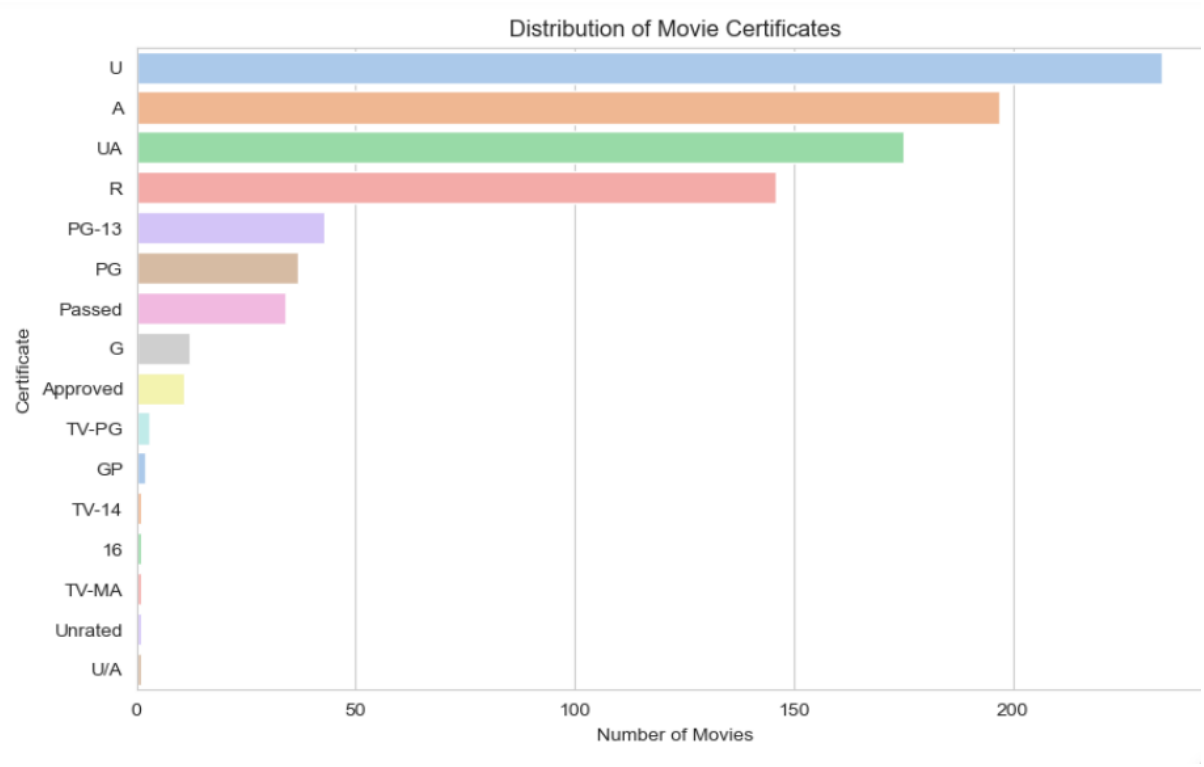
Picture 6. Number of Movies Released per Year

This bar chart shows the number of movies released per year. The x-axis represents the years, and the y-axis represents the number of movies released in each year. The bars are colored using the "magma" palette, transitioning from dark to light shades. The data suggests that more movies were released in recent years compared to older years. The x-axis labels are rotated to make them readable.

2.7 Distribution of Movie Certificates

```
plt.figure(figsize=(10, 6))
sns.barplot(y=df['Certificate'].value_counts().index,
            x=df['Certificate'].value_counts().values,
            palette='pastel')
plt.title('Distribution of Movie Certificates')
plt.xlabel('Number of Movies')
plt.ylabel('Certificate')
plt.show()
```

The code creates a horizontal bar chart showing the distribution of movie certificates. It first calculates the count of each unique certificate in the dataset and then plots these values using Seaborn. The x-axis represents the number of movies, while the y-axis represents different certificate categories. The pastel color palette is used for visualization.



Picture 7. Distribution of Movie Certificates

The graph visually represents the frequency of movie certificates, showing that the most common certificates are "U," "A," "UA," and "R." These categories have the highest number of movies, while other certifications like "TV-MA" and "Unrated" appear less frequently. This distribution highlights the dominance of general audience and restricted content ratings in the dataset.

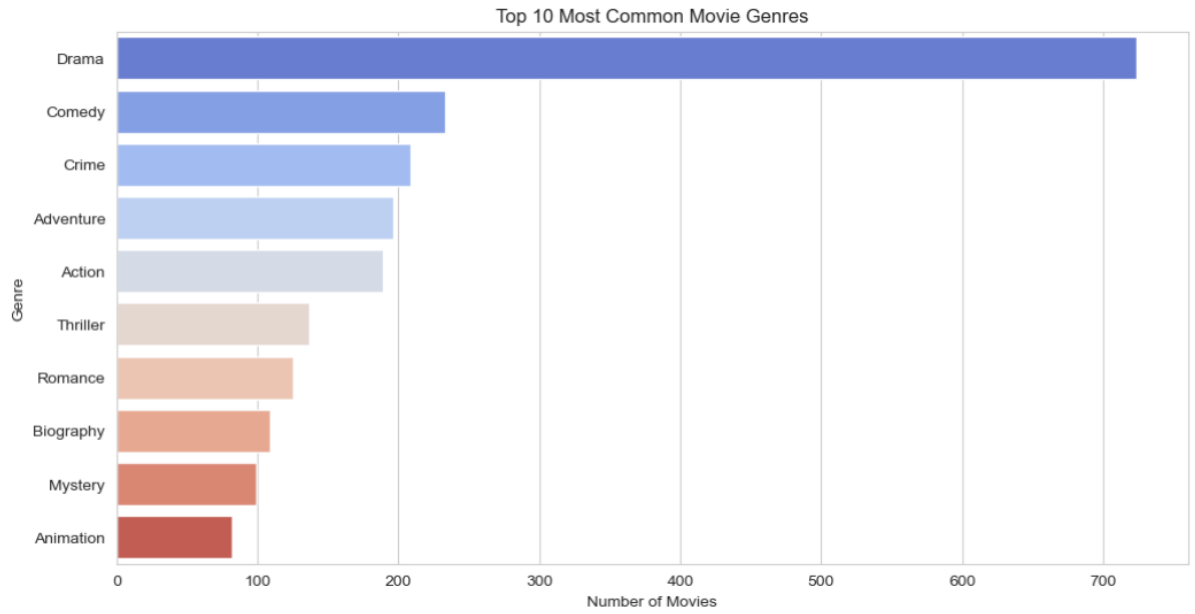
```
df = pd.read_csv(r"C:\Users\Korisnik\Downloads\archive\imdb_top_1000.csv")

print(df.head())  # Check if Genre column has values again
```

This code reads a CSV file named "imdb_top_1000.csv" from a specified directory using pandas and stores it in a DataFrame called df. The raw string notation (r"") ensures that backslashes in the file path are treated correctly. The df.head() function prints the first five rows of the DataFrame to check if the data, specifically the "Genre" column, is loaded correctly. The comment suggests that the user is verifying whether the "Genre" column has values after reloading the dataset.

2.8 Top 10 Movies Genres

```
df_exploded = df.assign(Genre=df['Genre'].str.split(', ')).explode('Genre')
genre_counts = df_exploded['Genre'].value_counts().head(10)

plt.figure(figsize=(12, 6))
sns.barplot(x=genre_counts.values, y=genre_counts.index, palette='coolwarm')
plt.title('Top 10 Most Common Movie Genres')
plt.xlabel('Number of Movies')
plt.ylabel('Genre')
plt.show()
```

This code processes the "Genre" column in the dataset to analyze the distribution of movie genres. Since some movies have multiple genres listed together, the code first splits these entries and expands them into separate rows using the explode function. Then, it counts the occurrences of each genre and selects the top 10 most common ones. Finally, it visualizes the results using a bar chart, where the x-axis represents the number of movies, and the y-axis lists the genres. This visualization helps identify the most prevalent genres in the dataset.



Picture 8. Number of Movies Released per Year

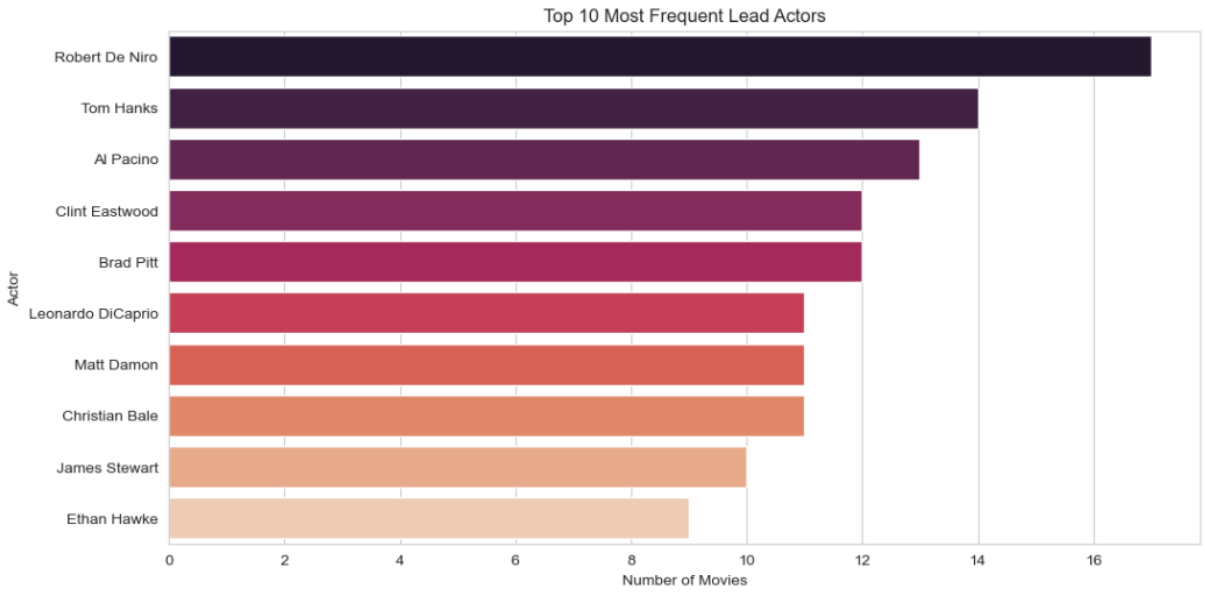
This bar chart visualizes the top movie genres based on their frequency in the dataset. Drama is the most common genre, followed by Comedy and Crime. The x-axis represents the number of movies, while the y-axis lists the genres. The chart helps identify the most

frequently occurring genres in the dataset, giving insights into the dominant trends in popular movies.

2.9 Top 10 Most Frequent Lead Actors

```
# Combine all stars into a single column
all_actors = pd.concat([df['Star1'], df['Star2'], df['Star3'], df['Star4']])

# Count the occurrences of each actor
actor_counts = all_actors.value_counts().head(10)

# Plot
plt.figure(figsize=(12, 6))
sns.barplot(y=actor_counts.index, x=actor_counts.values, palette='rocket')
plt.title('Top 10 Most Frequent Lead Actors')
plt.xlabel('Number of Movies')
plt.ylabel('Actor')
plt.show()
```

This code analyzes the most frequently appearing actors in the dataset. It first combines the four columns representing lead actors into a single column. Then, it counts the occurrences of each actor and selects the top 10. Finally, it visualizes the results using a horizontal bar chart, where the x-axis represents the number of movies and the y-axis lists the most frequently appearing actors. This helps identify which actors have starred in the most movies in the dataset.



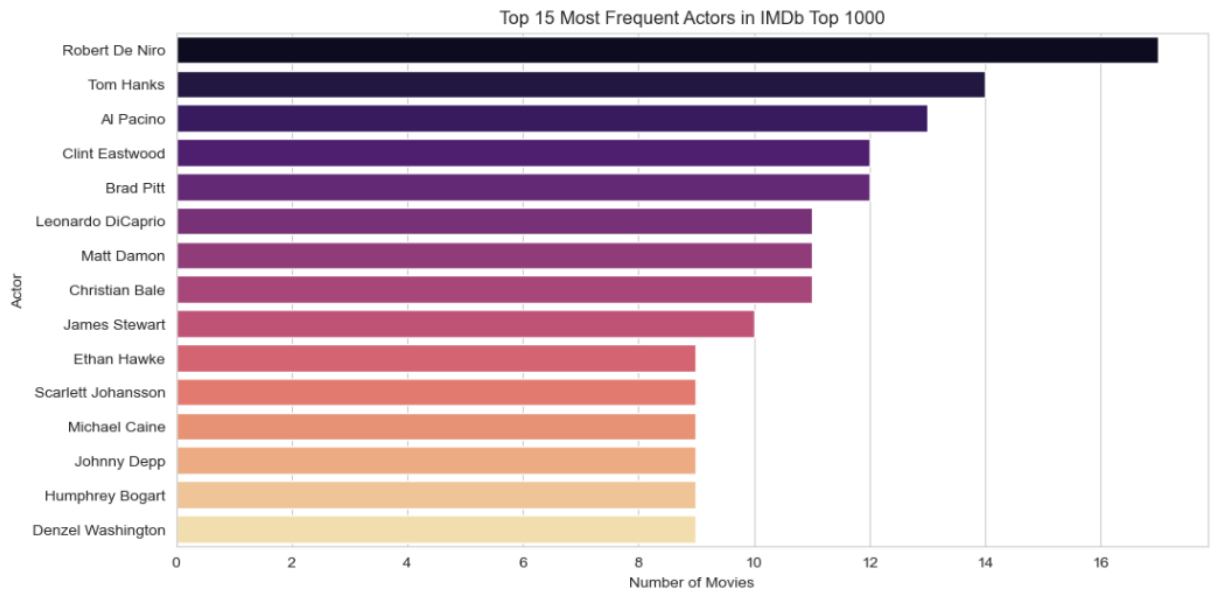
Picture 9. Top 10 Most Frequent Lead Actors

This bar chart visualizes the top 10 most frequently appearing lead actors in the dataset. The x-axis represents the number of movies, while the y-axis lists the actors. Robert De Niro appears in the highest number of films, followed by Tom Hanks, Al Pacino, and others. The chart provides insight into which actors have the most appearances in top-rated movies, highlighting their prominence in the film industry.

2.10 Top 15 Most Frequent Actors in IMDb Top 1000

```
# Combine all actors into a single column
all_actors = pd.concat([df['Star1'], df['Star2'], df['Star3'], df['Star4']])

# Count occurrences
actor_counts = all_actors.value_counts().head(15)

# Plot
plt.figure(figsize=(12, 6))
sns.barplot(y=actor_counts.index, x=actor_counts.values, palette='magma')
plt.title('Top 15 Most Frequent Actors in IMDb Top 1000')
plt.xlabel('Number of Movies')
plt.ylabel('Actor')
plt.show()
```

This code analyzes the most frequently appearing actors in the IMDb Top 1000 movies. It first combines the actor columns into a single column, ensuring that all listed actors are included. Then, it counts how many times each actor appears in the dataset and selects the top fifteen. Finally, it creates a horizontal bar chart where the x-axis represents the number of movies, and the y-axis lists the actors. The "magma" color palette is used to enhance the visualization. This helps in identifying which actors appear most often in highly-rated films.



Picture 10. Top 15 Most Frequent Actors in IMDb Top 1000

This bar chart shows the top 15 actors who appear most frequently in the IMDb Top 1000 movies. Robert De Niro appears the most, followed by Tom Hanks and Al Pacino. The x-axis represents the number of movies each actor has starred in, while the y-axis lists the actors. The "magma" color palette gives the bars a gradient effect, with darker shades for higher counts. This visualization highlights which actors have had the most consistent presence in highly rated films.

2.11 Al Pacino Movie Analysis

```
# Filter movies where Al Pacino is in any of the star columns
al_pacino_movies = df[(df['Star1'] == 'Al Pacino') |
                      (df['Star2'] == 'Al Pacino') |
                      (df['Star3'] == 'Al Pacino') |
                      (df['Star4'] == 'Al Pacino')]

# Count movies
num_al_pacino_movies = len(al_pacino_movies)

print(f'Al Pacino has acted in {num_al_pacino_movies} movies in the IMDb Top 1000.')

# Display movies
print(al_pacino_movies[['Series_Title', 'Released_Year', 'IMDB_Rating']])
```

This code filters movies where Al Pacino is listed in any of the four-star columns. It then counts how many times he appears and prints the result. Finally, it displays a table with the

movie title, release year, and IMDb rating for the films he has acted in. This helps analyze Al Pacino's presence in top-rated movies.

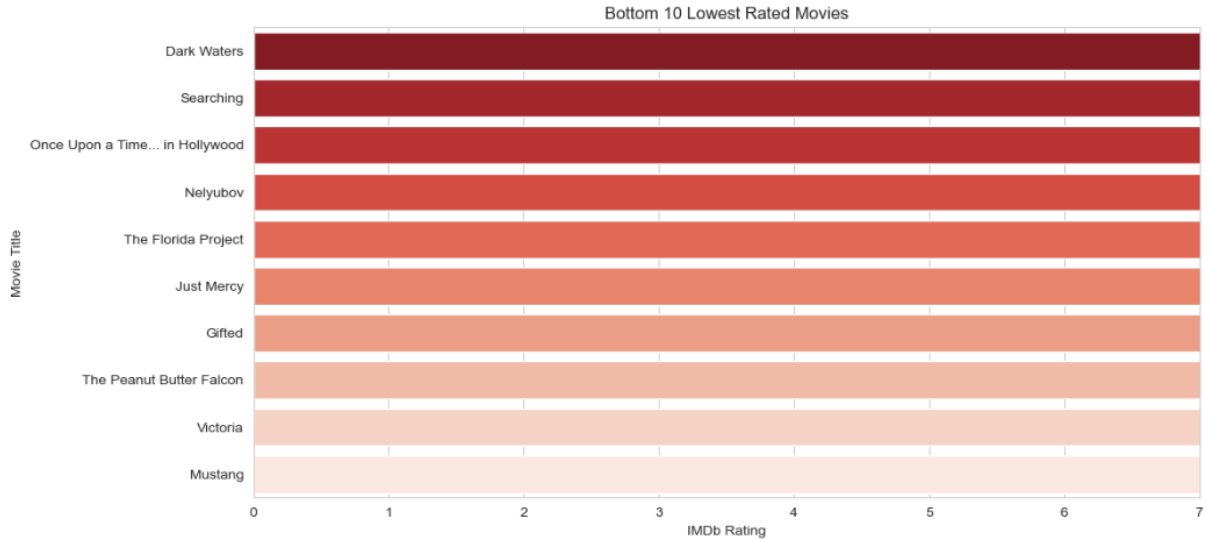
```
Al Pacino has acted in 13 movies in the IMDb Top 1000.
                Series_Title Released_Year  IMDB_Rating
1               The Godfather          1972          9.2
3     The Godfather: Part II           1974          9.0
108                  Scarface          1983          8.3
164                      Heat          1995          8.2
398          Scent of a Woman          1992          8.0
416         Dog Day Afternoon          1975          8.0
484              The Irishman          2019          7.9
523             Carlito's Way          1993          7.9
649               The Insider          1999          7.8
809             Donnie Brasco          1997          7.7
823       Glengarry Glen Ross          1992          7.7
849                   Serpico          1973          7.7
974  The Godfather: Part III           1990          7.6
```

The output shows that Al Pacino has appeared in 13 movies that are ranked among IMDb's top 1000. The table lists these movies along with their release years and IMDb ratings. His highest-rated films include *The Godfather* (9.2) and *The Godfather: Part II* (9.0), while *The Godfather: Part III* has the lowest rating at 7.6. The list highlights his most iconic performances in crime and drama films spanning several decades.

2.12 Bottom 10 Lowest Rated Movies

```python
# Get the lowest-rated movies
lowest_rated_movies = df.nsmallest(10, 'IMDB_Rating')[['Series_Title', 'IMDB_Rating', 'Released_Year']]

# Plot
plt.figure(figsize=(12, 6))
sns.barplot(y=lowest_rated_movies['Series_Title'], x=lowest_rated_movies['IMDB_Rating'], palette='Reds_r')
plt.title('Bottom 10 Lowest Rated Movies')
plt.xlabel('IMDb Rating')
plt.ylabel('Movie Title')
plt.xlim(0, 7)  # Adjust x-axis
plt.show()

# Display movies
print(lowest_rated_movies)
```

This code finds the 10 lowest-rated movies in the dataset based on IMDb ratings. It selects the movie titles, ratings, and release years. Then, it creates a horizontal bar chart to show these movies, using a red color palette. The x-axis is adjusted to range from 0 to 7. Finally, it prints the list of these lowest-rated movies.



Picture 11. Bottom 10 Lowest Rated Movies

This bar chart shows the 10 lowest-rated movies in the IMDb Top 1000. The movies are listed on the y-axis, and their IMDb ratings are on the x-axis. Dark Waters has the lowest rating among them, while Mustang has the highest rating in this group. The red color gradient represents the rating intensity, with darker shades indicating lower-rated movies.

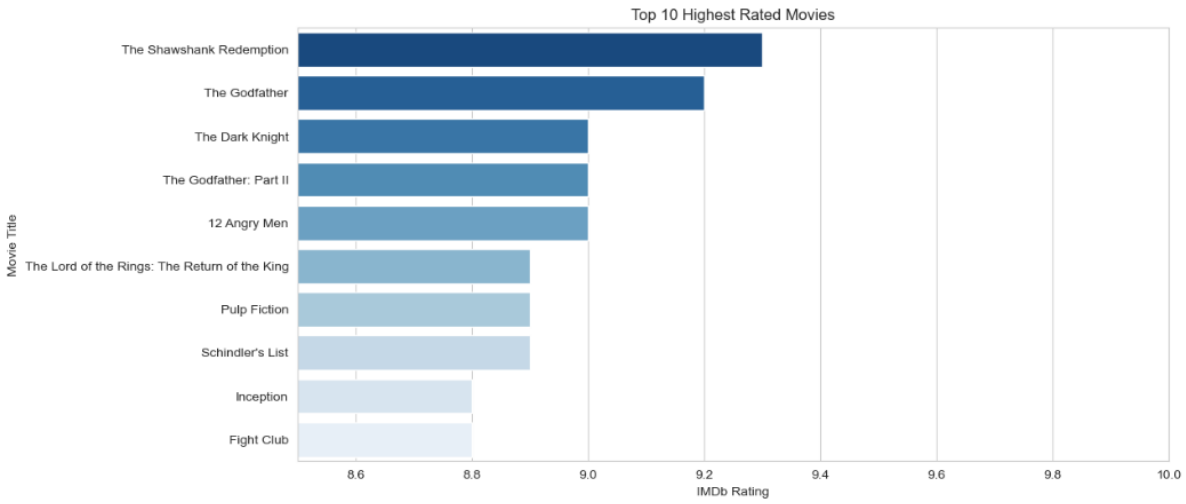
	Series_Title	IMDB_Rating	Released_Year
877	Dark Waters	7.6	2019
878	Searching	7.6	2018
879	Once Upon a Time... in Hollywood	7.6	2019
880	Nelyubov	7.6	2017
881	The Florida Project	7.6	2017
882	Just Mercy	7.6	2019
883	Gifted	7.6	2017
884	The Peanut Butter Falcon	7.6	2019
885	Victoria	7.6	2015
886	Mustang	7.6	2015

This table displays the 10 lowest-rated movies in the IMDb Top 1000. Each movie has an IMDb rating of 7.6, making them the lowest in the dataset. The table includes three columns: the movie title (Series_Title), its IMDb rating (IMDB_Rating), and the year it was released (Released_Year). All these movies were released between 2015 and 2019.

2.13 Top 10 Highest Rated Movies

```
# Get top-rated movies
top_rated_movies = df.nlargest(10, 'IMDB_Rating')[['Series_Title', 'IMDB_Rating', 'Released_Year']]

# Plot
plt.figure(figsize=(12, 6))
sns.barplot(y=top_rated_movies['Series_Title'], x=top_rated_movies['IMDB_Rating'], palette='Blues_r')
plt.title('Top 10 Highest Rated Movies')
plt.xlabel('IMDb Rating')
plt.ylabel('Movie Title')
plt.xlim(8.5, 10)  # Adjust x-axis for better visualization
plt.show()

# Display movies
print(top_rated_movies)
```

This code finds the 10 highest-rated movies in the IMDb Top 1000. It selects movies with the highest IMDb ratings and displays them in a table. The code also creates a bar chart to visualize these top-rated movies, using blue shades for the bars. The x-axis is adjusted to better display the ratings, ranging from 8.5 to 10.



Picture 12. Top 10 Highest Rated Movies

This chart shows the top 10 highest-rated movies on IMDb. *The Shawshank Redemption* has the highest rating, followed by *The Godfather* and *The Dark Knight*. The x-axis represents IMDb ratings, and the darker bars indicate higher ratings. All movies have ratings above 8.6.

	Series_Title	IMDB_Rating	Released_Year
0	The Shawshank Redemption	9.3	1994
1	The Godfather	9.2	1972
2	The Dark Knight	9.0	2008
3	The Godfather: Part II	9.0	1974
4	12 Angry Men	9.0	1957
5	The Lord of the Rings: The Return of the King	8.9	2003
6	Pulp Fiction	8.9	1994
7	Schindler's List	8.9	1993
8	Inception	8.8	2010
9	Fight Club	8.8	1999

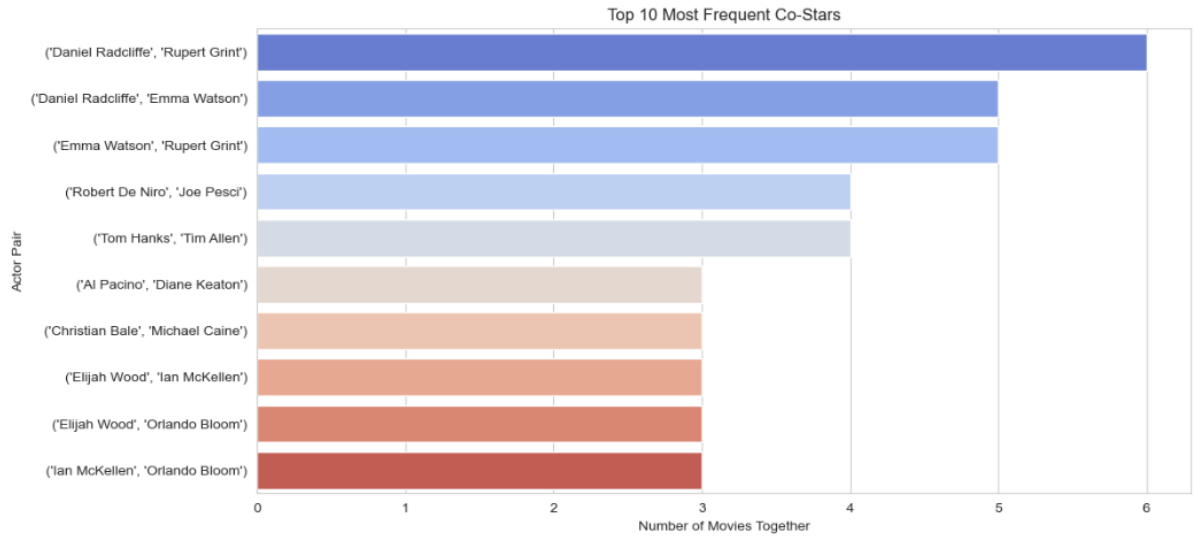
This table lists the top 10 highest-rated movies on IMDb. *The Shawshank Redemption* is ranked highest with a 9.3 rating, followed by *The Godfather* at 9.2. Other highly rated movies include *The Dark Knight*, *12 Angry Men*, and *The Lord of the Rings: The Return of the King*. The ratings range from 8.8 to 9.3.

2.14 Top 10 Most Frequent Co-Stars

```
from itertools import combinations
from collections import Counter

# Create actor pairs from each movie
actor_pairs = []
for i, row in df.iterrows():
    actors = [row['Star1'], row['Star2'], row['Star3'], row['Star4']]
    pairs = combinations(actors, 2)  # Create all 2-actor combinations
    actor_pairs.extend(pairs)

# Count occurrences
pair_counts = Counter(actor_pairs)
top_pairs = pair_counts.most_common(10)

# Convert to DataFrame
top_pairs_df = pd.DataFrame(top_pairs, columns=['Pair', 'Count'])

# Plot
plt.figure(figsize=(12, 6))
sns.barplot(y=top_pairs_df['Pair'].astype(str), x=top_pairs_df['Count'], palette='coolwarm')
plt.title('Top 10 Most Frequent Co-Stars')
plt.xlabel('Number of Movies Together')
plt.ylabel('Actor Pair')
plt.show()
```

This code analyzes the most frequently paired actors in movies. It extracts actor pairs from each movie by selecting up to four main actors and generating all possible two-actor combinations.

It then counts how many times each pair appears together across different films. The most common pairs are stored in a DataFrame, which is then visualized using a bar chart to highlight the top ten most frequent co-stars.



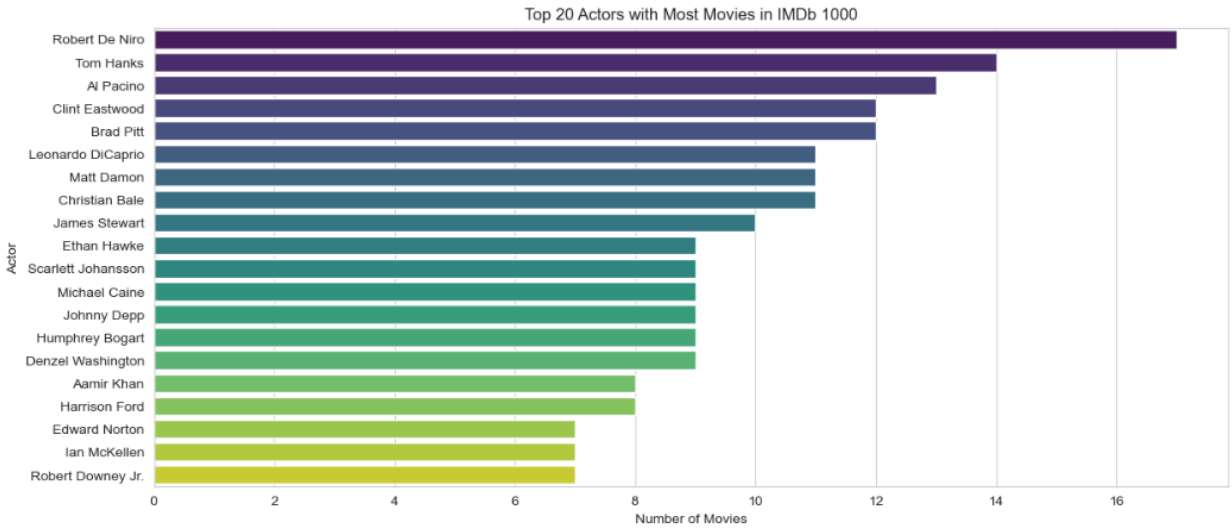
Picture 13. Top 10 Most Frequent Co -Stars

This chart displays the top ten most frequently paired actors in movies. Daniel Radcliffe, Rupert Grint, and Emma Watson appear together the most, reflecting their roles in the Harry Potter series. Other frequent co-stars include Robert De Niro and Joe Pesci, known for their collaborations in crime films, as well as Tom Hanks and Tim Allen, who voiced characters in the Toy Story franchise. The list also includes actors from The Godfather, The Dark Knight trilogy, and The Lord of the Rings, highlighting famous recurring duos in cinema.

2.15 Al Pacino Movie Analysis

```
# Combine all actor columns into a single column
actors = pd.concat([df['Star1'], df['Star2'], df['Star3'], df['Star4']])

# Count occurrences of each actor
actor_counts = actors.value_counts().head(20)  # Top 20 most frequent actors

# Plot
plt.figure(figsize=(14, 6))
sns.barplot(y=actor_counts.index, x=actor_counts.values, palette='viridis')
plt.title('Top 20 Actors with Most Movies in IMDb 1000')
plt.xlabel('Number of Movies')
plt.ylabel('Actor')
plt.show()
```

This code identifies the most frequently appearing actors in the dataset. It combines actor columns into a single list and counts how many times each actor appears. The top 20 actors with the most movies are then displayed in a bar chart. The chart visually represents which actors have been in the highest number of films in the dataset.



Picture 14. Top 20 Actors with Most Movies

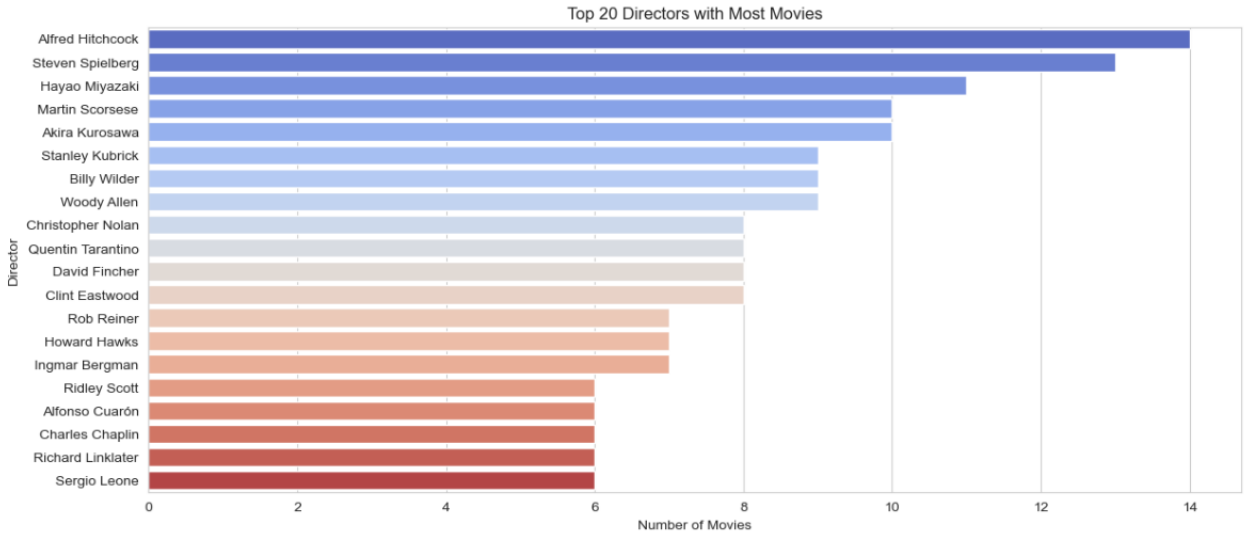
This bar chart shows the top 20 actors who have appeared in the most movies within the dataset. Robert De Niro has the highest number of films, followed by Tom Hanks and Al Pacino. The visualization highlights actors with consistent appearances in highly rated movies.

2.16 Top 20 Directors with Most Movies

```
director_counts = df['Director'].value_counts().head(20)  # Top 20 directors

plt.figure(figsize=(14, 6))
sns.barplot(y=director_counts.index, x=director_counts.values, palette='coolwarm')
plt.title('Top 20 Directors with Most Movies')
plt.xlabel('Number of Movies')
plt.ylabel('Director')
plt.show()
```

This code calculates the number of movies directed by each director and selects the top 20 with the most films. It then creates a bar chart displaying these directors along with their

movie counts. The visualization helps identify the most frequently featured directors in the dataset.



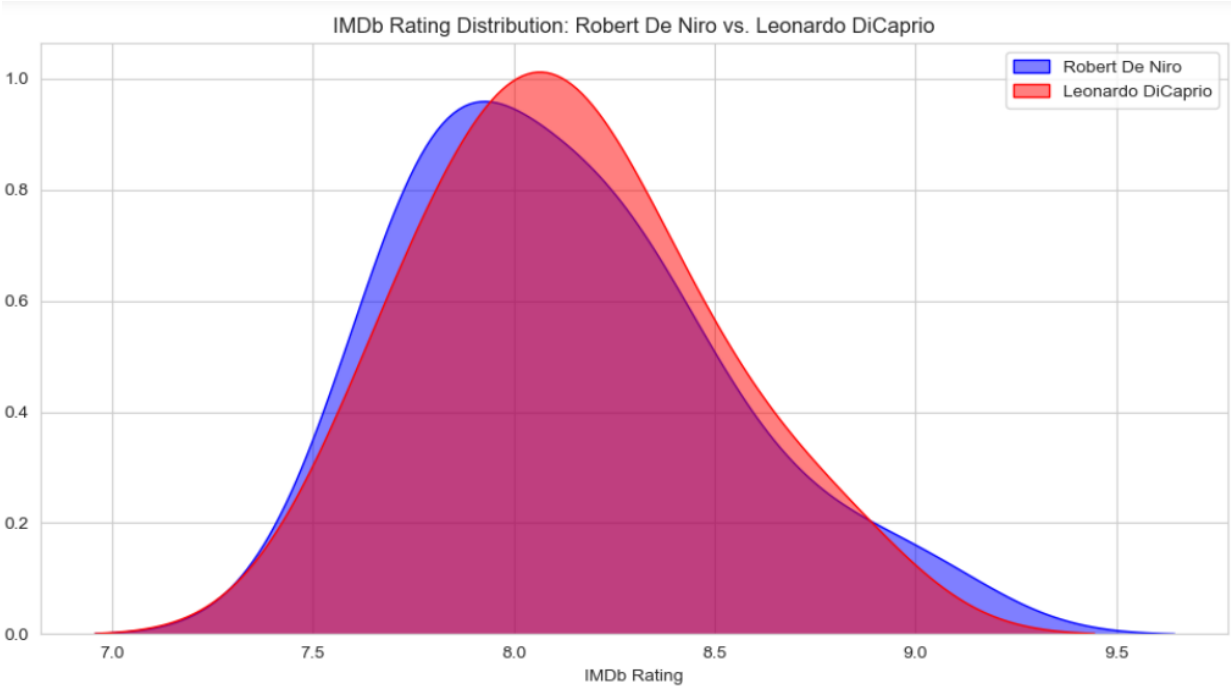
Picture 15. Top 20 Directors with Most Movies

This bar chart displays the top 20 directors with the most movies in the dataset. Alfred Hitchcock, Steven Spielberg, and Hayao Miyazaki are among the most frequently featured directors. The visualization provides insight into the most influential filmmakers based on the number of their movies included.

2.17 IMDb Rating Distribution -Robert De Niro vs. Leonardo DiCaprio

```
# Filter movies with Robert De Niro and Leonardo DiCaprio
de_niro_movies = df[df[['Star1', 'Star2', 'Star3', 'Star4']].eq("Robert De Niro").any(axis=1)]
dicaprio_movies = df[df[['Star1', 'Star2', 'Star3', 'Star4']].eq("Leonardo DiCaprio").any(axis=1)]

# Compare IMDb ratings
plt.figure(figsize=(12, 6))
sns.kdeplot(de_niro_movies['IMDB_Rating'], label='Robert De Niro', fill=True, color='blue', alpha=0.5)
sns.kdeplot(dicaprio_movies['IMDB_Rating'], label='Leonardo DiCaprio', fill=True, color='red', alpha=0.5)
plt.title('IMDb Rating Distribution: Robert De Niro vs. Leonardo DiCaprio')
plt.xlabel('IMDb Rating')
plt.ylabel('Density')
plt.legend()
plt.show()
```

This code filters movies featuring Robert De Niro and Leonardo DiCaprio. It then compares their IMDb rating distributions using a density plot. The blue area represents De Niro's movies, while the red area represents DiCaprio's movies. The visualization helps analyze the rating patterns of films starring these two actors.



Picture 16. Top 10 IMDb Rating Distribution -Robert De Niro vs. Leonardo DiCaprio

This density plot compares the IMDb rating distributions of movies starring Robert De Niro (blue) and Leonardo DiCaprio (red).

Observations:

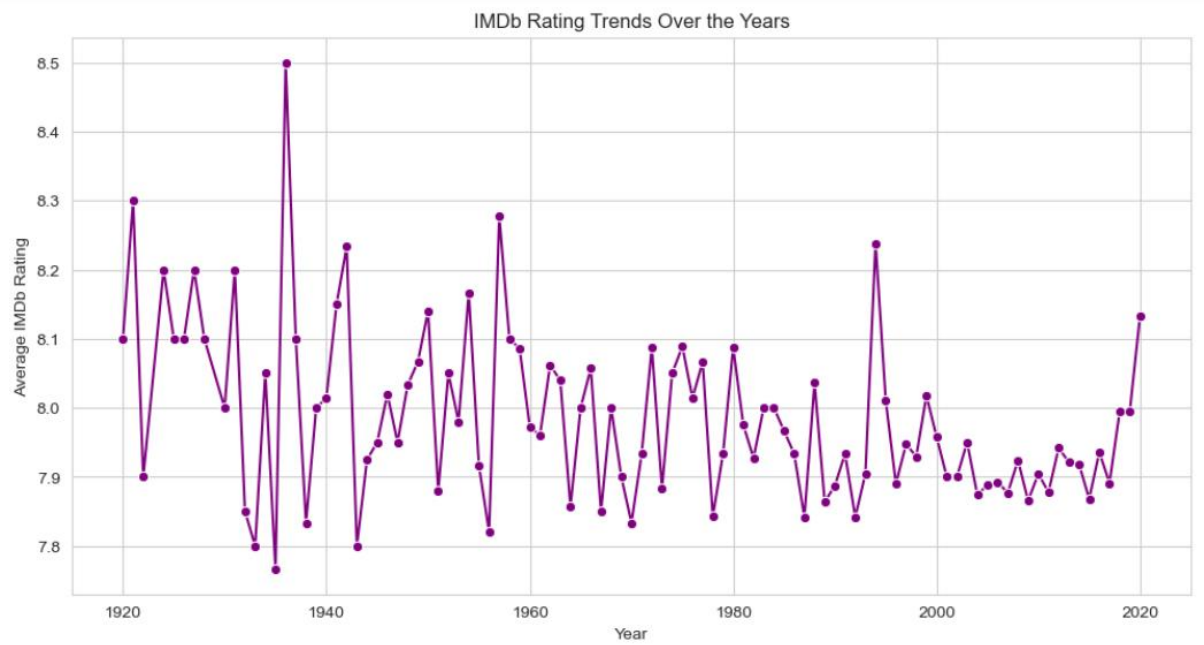
- Both actors have a similar rating distribution, mostly ranging from **7.0 to 9.5**.
- De Niro's movies peak around **7.8**, whereas DiCaprio's movies peak slightly higher, around **8.0**.
- DiCaprio's movies appear to have a more concentrated distribution around **8.0**, while De Niro's ratings show a wider spread.
- Both actors have high-rated films, with some going beyond **9.0**.

This suggests that both actors consistently appear in highly rated films, with DiCaprio's movies tending to have slightly higher ratings on average.

2.18 IMDb Rating Trends Over the Years

```
#If movies are getting better or worse by the time
# Convert Released_Year to numeric
df['Released_Year'] = pd.to_numeric(df['Released_Year'], errors='coerce')

# Average IMDb Rating per year
yearly_ratings = df.groupby('Released_Year')['IMDB_Rating'].mean()

plt.figure(figsize=(12, 6))
sns.lineplot(x=yearly_ratings.index, y=yearly_ratings.values, marker='o', color='purple')
plt.title('IMDb Rating Trends Over the Years')
plt.xlabel('Year')
plt.ylabel('Average IMDb Rating')
plt.grid(True)
plt.show()
```

This code tracks how IMDb ratings have changed over the years. It converts release years to numbers, finds the average rating per year, and then plots a **line graph** to show trends over time.



Picture 17. IMDb Rating Over the Years

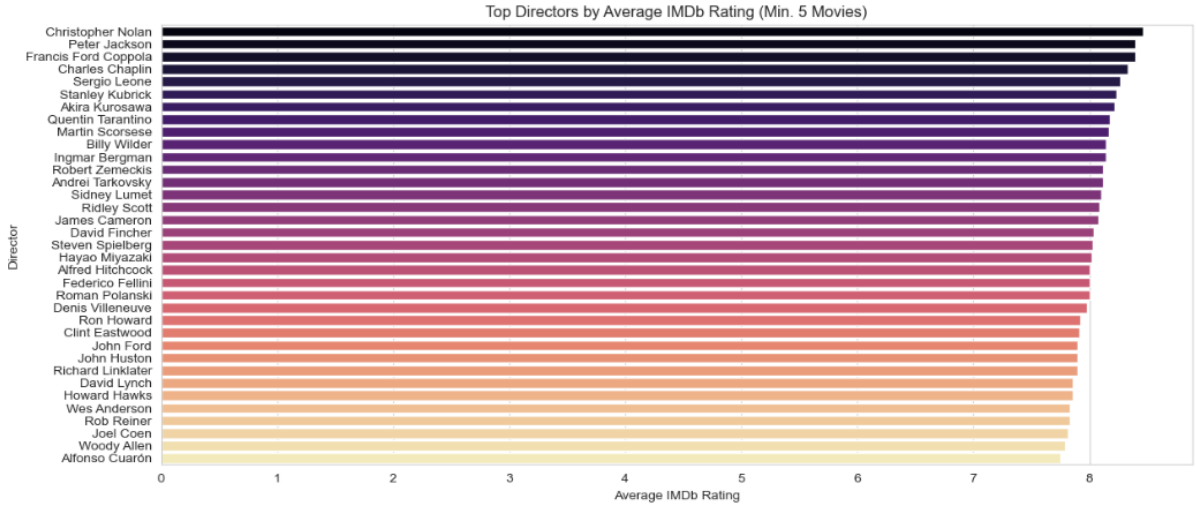
This graph shows how the average IMDb rating of movies has changed over time. Each point represents the average rating of movies released in a specific year. There are many ups and downs, suggesting that some years had stronger movies while others had weaker ones. The ratings for older movies, especially before the 1950s, vary a lot, possibly due to

fewer movies being made or shifts in movie quality. From the 1980s to the 2010s, movie ratings appear more stable, without extreme fluctuations. A sharp increase in ratings after 2015 suggests that recent movies might be receiving better ratings, possibly due to modern trends or fan-based voting.

2.19 Top Directors by Average IMDb Rating

```
#WHO IS CONSTANTLY MAKING GREAT MOVIES
# Get directors with at least 5 movies
top_directors = df['Director'].value_counts()[df['Director'].value_counts() >= 5].index
director_ratings = df[df['Director'].isin(top_directors)].groupby('Director')['IMDB_Rating'].mean().sort_values(ascending=False)

plt.figure(figsize=(14, 6))
sns.barplot(y=director_ratings.index, x=director_ratings.values, palette='magma')
plt.title('Top Directors by Average IMDb Rating (Min. 5 Movies)')
plt.xlabel('Average IMDb Rating')
plt.ylabel('Director')
plt.show()
```

This code finds the best directors based on average IMDb ratings, but only if they've directed at least five movies. First, it filters out directors with fewer than five movies. Then, it calculates the average IMDb rating for each qualifying director. Finally, it creates a horizontal bar chart, ranking directors from highest to lowest average IMDb rating.



Picture 18. Top Directors by Average IMDb Rating

This bar chart shows the top directors ranked by their average IMDb rating, considering only those who have directed at least five movies. Christopher Nolan has the highest average rating, followed closely by Peter Jackson and Francis Ford Coppola. The color gradient from dark to light represents the ranking, with darker bars indicating higher ratings.

2.20 Filtering Nolan Movies with Leonardo DiCaprio as a Star

```
# Filter movies where Christopher Nolan is the director and Leonardo DiCaprio is one of the stars
leo_nolan_movies = df[(df['Director'] == 'Christopher Nolan') &
                      ((df['Star1'] == 'Leonardo DiCaprio') |
                       (df['Star2'] == 'Leonardo DiCaprio') |
                       (df['Star3'] == 'Leonardo DiCaprio') |
                       (df['Star4'] == 'Leonardo DiCaprio'))]

# Display results
print(leo_nolan_movies[['Series_Title', 'Released_Year', 'IMDB_Rating']])
```

```
  Series_Title  Released_Year  IMDB_Rating
8    Inception         2010.0          8.8
```

This code filters the dataset to find movies directed by Christopher Nolan that also star Leonardo DiCaprio. It checks if the Director column is "Christopher Nolan" and if DiCaprio appears in any of the Star1, Star2, Star3, or Star4 columns. The result shows "Inception" (2010) with an IMDb rating of 8.8, which is the only movie in the dataset meeting these criteria.

2.21 Filtering Nolan Movies with Leonardo Al Pacino as a Star

```
# Filter movies where Christopher Nolan is the director and Al Pacino is one of the stars
pacino_nolan_movies = df[(df['Director'] == 'Christopher Nolan') &
                         ((df['Star1'] == 'Al Pacino') |
                          (df['Star2'] == 'Al Pacino') |
                          (df['Star3'] == 'Al Pacino') |
                          (df['Star4'] == 'Al Pacino'))]

# Display results
print(pacino_nolan_movies[['Series_Title', 'Released_Year', 'IMDB_Rating']])
```

```
Empty DataFrame
Columns: [Series_Title, Released_Year, IMDB_Rating]
Index: []
```

This code attempts to find movies directed by Christopher Nolan where Al Pacino is one of the stars. It filters the dataset by checking if the Director is "Christopher Nolan" and if Al Pacino appears in any of the Star1, Star2, Star3, or Star4 columns. However, the output is an empty DataFrame, meaning that no movies in the dataset match both conditions. This suggests that Al Pacino has never starred in a Christopher Nolan-directed movie.

2.22 Movie with the Longest runtime

```
# Convert 'Runtime' to numeric (removing ' min')
df['Runtime'] = df['Runtime'].str.replace(' min', '').astype(float)

# Find the movie with the longest runtime
longest_movie = df.nlargest(1, 'Runtime')[['Series_Title', 'Runtime', 'Released_Year', 'IMDB_Rating']]

# Display result
print(longest_movie)
```

```
           Series_Title  Runtime  Released_Year  IMDB_Rating
140  Gangs of Wasseypur    321.0         2012.0          8.2
```

This code converts the 'Runtime' column into a numeric format by removing the " min" text and changing the data type to float. Then, it finds the movie with the longest runtime using the nlargest(1, 'Runtime') function. The output shows that "Gangs of Wasseypur" (2012) is the longest movie in the dataset, with a runtime of 321 minutes and an IMDb rating of 8.2.

2.23 Movie with the Shortest runtime

```
# Find the movie with the shortest runtime
shortest_movie = df.nsmallest(1, 'Runtime')[['Series_Title', 'Runtime', 'Released_Year', 'IMDB_Rating']]

# Display result
print(shortest_movie)
```

```
     Series_Title  Runtime  Released_Year  IMDB_Rating
194  Sherlock Jr.     45.0         1924.0          8.2
```

This code identifies the movie with the shortest runtime by using the nsmallest(1, 'Runtime') function. The output shows that "Sherlock Jr." (1924) has the shortest runtime in the dataset, lasting 45 minutes, with an IMDb rating of 8.2.

2.24 Movie with the Highest Number of Votes

```
# Find the movie with the highest number of votes
most_voted_movie = df.nlargest(1, 'No_of_Votes')[['Series_Title', 'No_of_Votes', 'Released_Year', 'IMDB_Rating']]

# Display result
print(most_voted_movie)
```

```
              Series_Title  No_of_Votes  Released_Year  IMDB_Rating
0  The Shawshank Redemption      2343110         1994.0          9.3
```

This code finds the movie with the highest number of votes using nlargest(1, 'No_of_Votes'). The output reveals that "The Shawshank Redemption" (1994) holds the most votes, with 2,343,110 votes and an impressive IMDb rating of 9.3.

2.25 Movie with the Lowest IMDb rating

```
# Find the movie with the lowest IMDb rating
lowest_rated_movie = df.nsmallest(1, 'IMDB_Rating')[['Series_Title', 'IMDB_Rating', 'Released_Year', 'No_of_Votes']]

# Display result
print(lowest_rated_movie)
```

```
    Series_Title  IMDB_Rating  Released_Year  No_of_Votes
877  Dark Waters          7.6         2019.0        60408
```

This code identifies the movie with the lowest IMDb rating using nsmallest(1, 'IMDB_Rating'). The output shows that "Dark Waters" (2019) has the lowest rating in the dataset, with an IMDb rating of 7.6 and 60,408 votes.

2.26 Movie with the Longest overview

```
# Create a new column for overview length
df['Overview_Length'] = df['Overview'].astype(str).apply(len)

# Find the movie with the longest overview
longest_overview_movie = df.nlargest(1, 'Overview_Length')[['Series_Title', 'Overview', 'Overview_Length', 'IMDB_Rating']]

# Display result
print(longest_overview_movie)
```

```
       Series_Title                                           Overview  \
935  Jeux d'enfants  As adults, best friends Julien and Sophie cont...

     Overview_Length  IMDB_Rating
935              313          7.6
```

This code creates a new column Overview_Length, which stores the character length of each movie's overview. Then, it finds the movie with the longest overview using nlargest(1, 'Overview_Length'). The result shows that "Jeux d'enfants" (also known as *Love Me If You Dare*) has the longest overview with 313 characters and an IMDb rating of 7.6.

2.27 Movie with the Shortest overview

```
# Ensure 'Overview' is a string and calculate its length
df['Overview_Length'] = df['Overview'].astype(str).apply(len)

# Find the movie with the shortest overview
shortest_overview_movie = df.nsmallest(1, 'Overview_Length')[['Series_Title', 'Overview', 'Overview_Length', 'IMDB_Rating']]

# Display result
print(shortest_overview_movie)
```

```
        Series_Title                                Overview  \
831  The Last Emperor  The story of the final Emperor of China.

     Overview_Length  IMDB_Rating
831               40          7.7
```

This code ensures that the Overview column is treated as a string and calculates its length, storing it in a new column called Overview_Length. It then identifies the movie with the shortest overview using nsmallest(1, 'Overview_Length'). The result shows that "The Last Emperor" has the shortest overview with only 40 characters and an IMDb rating of 7.7.

2.28 Filtering Movies Where Leonardo DiCaprio Is One of the Stars

```
# Filter movies where Leonardo DiCaprio is in any of the star columns
leo_movies = df[(df['Star1'] == 'Leonardo DiCaprio') |
                (df['Star2'] == 'Leonardo DiCaprio') |
                (df['Star3'] == 'Leonardo DiCaprio') |
                (df['Star4'] == 'Leonardo DiCaprio')]

# Select relevant columns
leo_movies = leo_movies[['Series_Title', 'Released_Year', 'IMDB_Rating', 'No_of_Votes']]

# Sort by release year
leo_movies = leo_movies.sort_values(by='Released_Year')

# Display all Leonardo DiCaprio movies
print(leo_movies)
```

```
                        Series_Title  Released_Year  IMDB_Rating  No_of_Votes
658       What's Eating Gilbert Grape         1993.0          7.8       215034
652                           Titanic         1997.0          7.8      1046089
243               Catch Me If You Can         2002.0          8.1       832846
37                        The Departed         2006.0          8.5      1189773
361                     Blood Diamond         2006.0          8.0       499439
8                           Inception         2010.0          8.8      2067042
145                    Shutter Island         2010.0          8.2      1129894
62                   Django Unchained         2012.0          8.4      1357682
147           The Wolf of Wall Street         2013.0          8.2      1187498
343                       The Revenant         2015.0          8.0       705589
879  Once Upon a Time... in Hollywood         2019.0          7.6       551309
```

This code filters a dataset (df) to extract all movies in which Leonardo DiCaprio is listed as one of the stars. It checks four columns (Star1, Star2, Star3, Star4) to find his name. After filtering, it selects only the relevant columns: movie title, release year, IMDb rating, and number of votes. The movies are then sorted by release year in ascending order and displayed.

Explanation of the Output

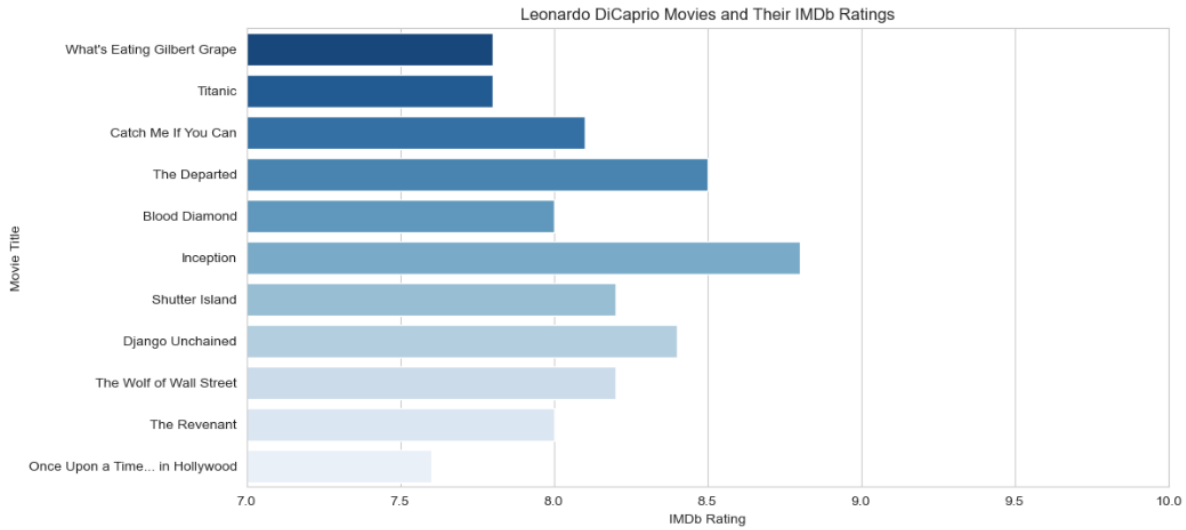
The output is a list of Leonardo DiCaprio's movies, ordered by release year. Each row contains:

- Series_Title: The movie's name.
- Released_Year: The year it was released.
- IMDB_Rating: The IMDb rating of the movie.
- No_of_Votes: The total number of votes the movie received on IMDb.

2.29 Leonardo DiCaprio Movies and Their IMDb Ratings

```
plt.figure(figsize=(12, 6))
sns.barplot(y=leo_movies['Series_Title'], x=leo_movies['IMDB_Rating'], palette='Blues_r')

plt.title('Leonardo DiCaprio Movies and Their IMDb Ratings')
plt.xlabel('IMDb Rating')
plt.ylabel('Movie Title')
plt.xlim(7, 10)  # Adjust the x-axis for better visualization

plt.show()
```

This code creates a horizontal bar chart to visualize Leonardo DiCaprio's movies and their IMDb ratings using seaborn and matplotlib. It first sets the figure size to make the chart wider. Then, it creates a bar plot where the y-axis represents the movie titles, and the x-axis represents the IMDb ratings, using a reversed blue color gradient. The chart title is set as "Leonardo DiCaprio Movies and Their IMDb Ratings," and the x-axis and y-axis are labeled accordingly. The x-axis range is limited to values between 7 and 10 to improve visualization. Finally, the plot is displayed. The output is a bar chart where each bar represents a movie, showing its IMDb rating with different shades of blue.



Picture 19. Leonardo DiCaoprio Movies And Their IMDb Ratings

The graph shows Leonardo DiCaprio's movies and their IMDb ratings. The x-axis represents ratings from 7.0 to 10.0, while the y-axis lists movie titles. Longer bars mean higher ratings. *Inception* has the highest rating, while *What's Eating Gilbert Grape* and *Titanic* have the lowest. Most of his movies are rated above 7.5, showing he has starred in well-received films.

2.30 Filtering Movies Where Richard Gere Is One of the Stars

```
# Filter movies where Richard Gere is in any of the star columns
gere_movies = df[(df['Star1'] == 'Richard Gere') |
                 (df['Star2'] == 'Richard Gere') |
                 (df['Star3'] == 'Richard Gere') |
                 (df['Star4'] == 'Richard Gere')]

# Select relevant columns
gere_movies = gere_movies[['Series_Title', 'Released_Year', 'IMDB_Rating', 'No_of_Votes']]

# Sort by release year
gere_movies = gere_movies.sort_values(by='Released_Year')

# Display all Richard Gere movies
print(gere_movies)
```

```
           Series_Title  Released_Year  IMDB_Rating  No_of_Votes
690       Days of Heaven         1978.0          7.8        52852
811          Primal Fear         1996.0          7.7       189716
228  Hachi: A Dog's Tale         2009.0          8.1       253575
```

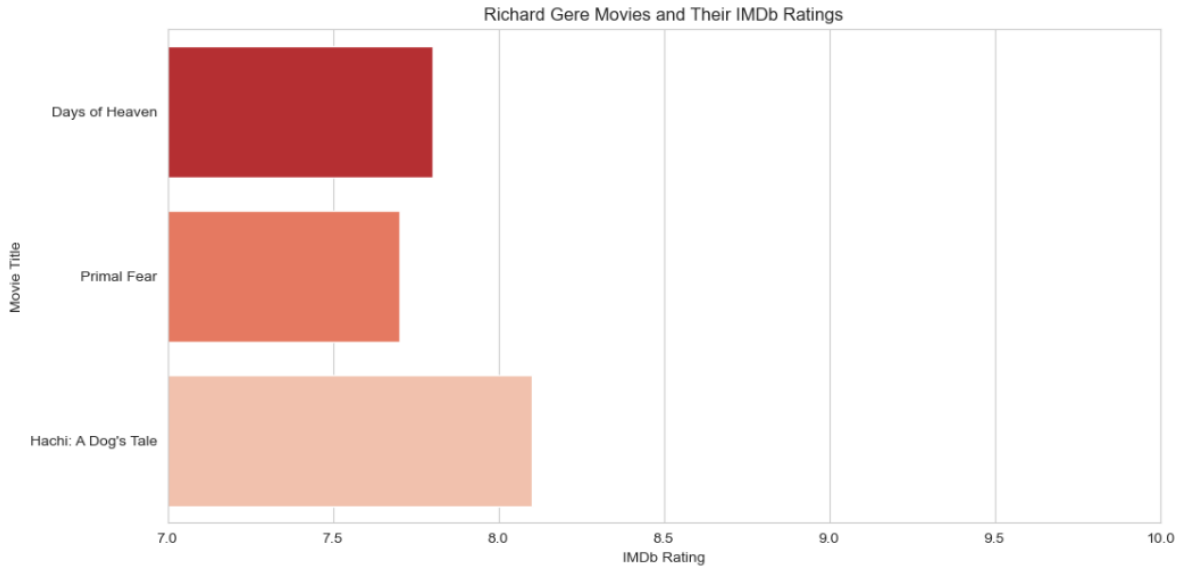
This code filters movies where Richard Gere is listed as one of the four stars. It then selects key columns: title, release year, IMDb rating, and number of votes. The movies are sorted by release year. Finally, it prints the results, showing three Richard Gere movies:

1. *Days of Heaven* (1978) – IMDb 7.8
2. *Primal Fear* (1996) – IMDb 7.7
3. *Hachi: A Dog's Tale* (2009) – IMDb 8.1

2.31 Richard Gere Movies and Their IMDb Ratings

```
plt.figure(figsize=(12, 6))
sns.barplot(y=gere_movies['Series_Title'], x=gere_movies['IMDB_Rating'], palette='Reds_r')

plt.title('Richard Gere Movies and Their IMDb Ratings')
plt.xlabel('IMDb Rating')
plt.ylabel('Movie Title')
plt.xlim(7, 10)

plt.show()
```

This code creates a horizontal bar chart showing Richard Gere's movies and their IMDb ratings. The figure size is set to 12x6 inches. The movie titles are on the Y-axis, and IMDb ratings are on the X-axis. A red color palette is used for the bars. The X-axis is limited between 7 and 10. Finally, the chart is displayed.



Picture 20. Richard Gere Movies and Their IMDb Ratings

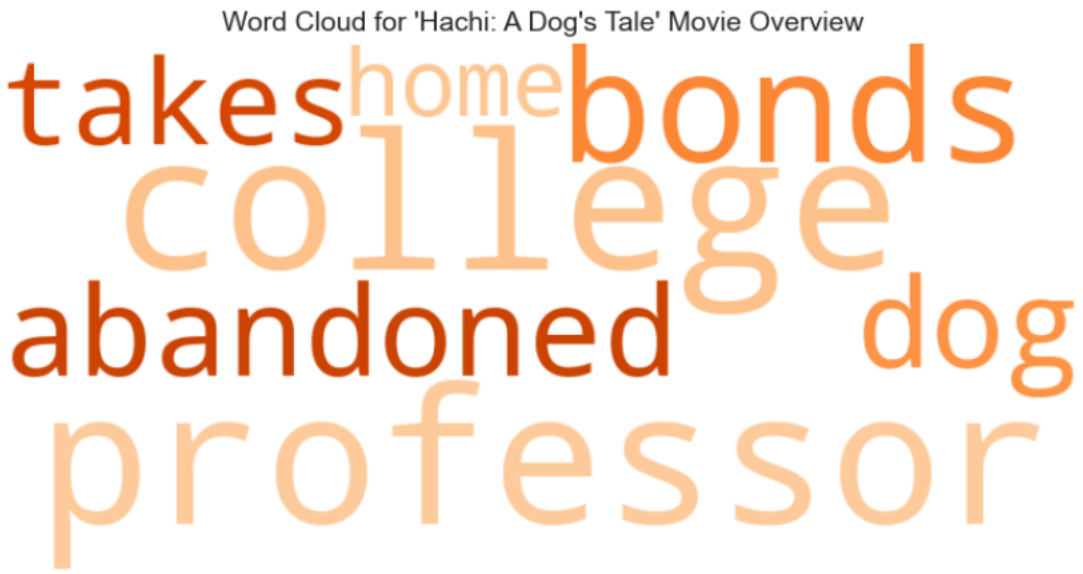
This graph shows Richard Gere's movies and their IMDb ratings using a horizontal bar chart. The movie titles are on the Y-axis, and the IMDb ratings are on the X-axis, ranging from 7 to 10. Darker shades of red represent lower ratings, while lighter shades indicate higher ratings. *Hachi: A Dog's Tale* has the highest rating at 8.1, followed by *Days of Heaven* at 7.8 and *Primal Fear* at 7.7.

2.32 WordCloud for "Hachi" Movie

```
import re
from wordcloud import WordCloud
import matplotlib.pyplot as plt

# Filter dataset for "Hachi: A Dog's Tale"
hachi_movie = df[df['Series_Title'].str.contains('Hachi: A Dog', case=False, na=False)]

# Check if the movie exists in the dataset
if not hachi_movie.empty:
    # Extract the overview text
    hachi_overview = hachi_movie['Overview'].values[0]

    # Clean the text: remove punctuation & lowercase all words
    cleaned_text = re.sub(r'[^\w\s]', '', hachi_overview.lower())

    # Create a WordCloud object
    wordcloud = WordCloud(width=800, height=400, background_color='white', colormap='Oranges').generate(cleaned_text)

    # Display the Word Cloud
    plt.figure(figsize=(10, 5))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis('off')  # Hide axes
    plt.title("Word Cloud for 'Hachi: A Dog's Tale' Movie Overview", fontsize=14)
    plt.show()
else:
    print("Movie 'Hachi: A Dog's Tale' not found in the dataset.")
```

This code generates a word cloud for the movie *Hachi: A Dog's Tale* based on its overview text. It first filters the dataset to find the movie. If found, it extracts and cleans the text by removing punctuation and converting it to lowercase.

A word cloud is then created with an orange color theme and displayed using Matplotlib. If the movie is not found, it prints a message saying so.



Picture 21. WordCloud For “Hachi” Movie

2.33 Column Names

```
# Display all column names
print(df.columns)
```

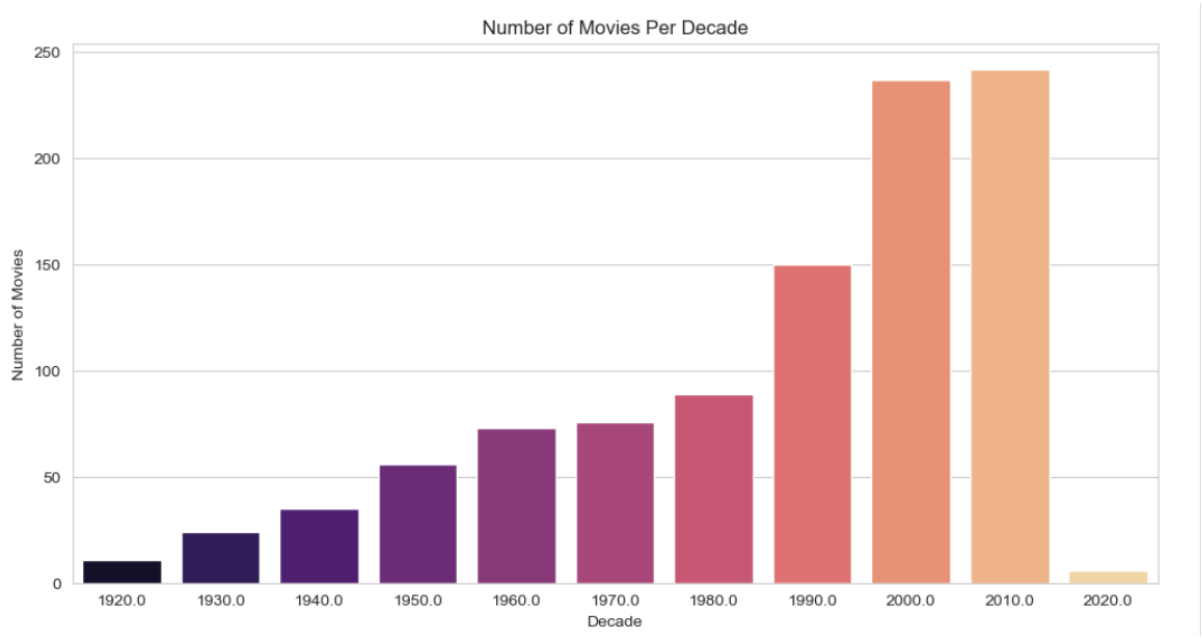
```
Index(['Poster_Link', 'Series_Title', 'Released_Year', 'Certificate',
       'Runtime', 'Genre', 'IMDB_Rating', 'Overview', 'Meta_score', 'Director',
       'Star1', 'Star2', 'Star3', 'Star4', 'No_of_Votes', 'Gross',
       'Overview_Length'],
      dtype='object')
```

This code displays all column names in the DataFrame (df). The dataset includes movie-related details such as title, release year, genre, IMDb rating, overview, director, and main actors. It also contains numerical data like runtime, number of votes, gross earnings, and overview length. The output confirms that these columns exist in the dataset.

2.34 Number of Movies Per Decade

```
df['Decade'] = (df['Released_Year'] // 10) * 10

plt.figure(figsize=(12, 6))
sns.barplot(x=df['Decade'].value_counts().index, y=df['Decade'].value_counts().values, palette='magma')

plt.title('Number of Movies Per Decade')
plt.xlabel('Decade')
plt.ylabel('Number of Movies')

plt.show()
```

This code groups movies by decade and creates a bar chart to show the number of movies released per decade. It calculates the decade by dividing the release year by 10 and multiplying by 10. A bar plot is created using the magma color palette, with decades on the X-axis and movie counts on the Y-axis. The graph is displayed with appropriate labels and a title.



Picture 22. Number of Movies Per Decade

This bar chart shows the number of movies released per decade. The X-axis represents the decades from the 1920s to the 2020s, while the Y-axis represents the number of movies produced in each decade. The trend shows a steady increase in movie production over time, with the highest number of movies released in the 2000s and 2010s. The 2020s have

significantly fewer movies, likely because the dataset is incomplete, or the decade is still ongoing. The magma color palette visually emphasizes the increasing trend in movie production.

2.35 Filtering Christopher Nolan Movies

```
# Filter movies directed by Christopher Nolan
nolan_movies = df[df['Director'] == 'Christopher Nolan']

# Select relevant columns
nolan_movies = nolan_movies[['Series_Title', 'Released_Year', 'IMDB_Rating']]

# Sort by release year
nolan_movies = nolan_movies.sort_values(by='Released_Year')

# Display the movies
print(nolan_movies)
```

```
              Series_Title  Released_Year  IMDB_Rating
69                 Memento         2000.0          8.4
155          Batman Begins         2005.0          8.2
36             The Prestige         2006.0          8.5
2           The Dark Knight         2008.0          9.0
8                 Inception         2010.0          8.8
63   The Dark Knight Rises         2012.0          8.4
21             Interstellar         2014.0          8.6
573                Dunkirk         2017.0          7.8
```

Filtering Movies: It selects only rows where the Director column is "Christopher Nolan".

Selecting Columns: It keeps only three columns:

- Series_Title (Movie title)
- Released_Year (Year of release)
- IMDB_Rating (Movie rating)

Sorting by Release Year: The movies are sorted in ascending order by their release year.

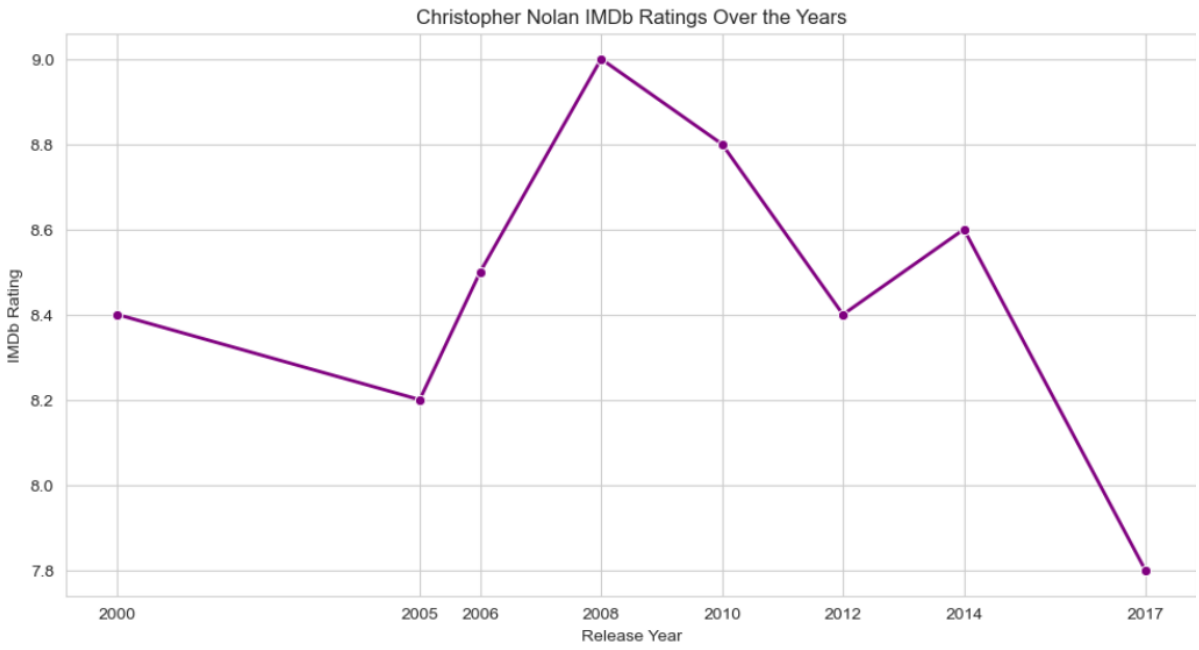
Displaying the Results: It prints the list of Christopher Nolan's movies, showing their release year and IMDb ratings. From the output, movies like Memento (2000), The Dark Knight (2008), and Interstellar (2014) are included, with The Dark Knight having the highest IMDb rating (9.0).

2.36 Christopher Nolan IMDb Ratings Over the Years

```
plt.figure(figsize=(12, 6))
sns.lineplot(x=nolan_movies['Released_Year'], y=nolan_movies['IMDB_Rating'], marker='o', color='purple', linewidth=2)

plt.xticks(nolan_movies['Released_Year'])  # Show every release year
plt.title('Christopher Nolan IMDb Ratings Over the Years')
plt.xlabel('Release Year')
plt.ylabel('IMDb Rating')
plt.grid(True)

plt.show()
```

This code creates a line plot to visualize Christopher Nolan's IMDb ratings over the years. It sets the figure size, then uses sns.lineplot() to plot release years on the x-axis and IMDb ratings on the y-axis, with purple lines and circular markers.

The x-axis labels are set to display every release year. A title, x-label, and y-label are added, and a grid is enabled for better readability. Finally, plt.show() displays the plot.



Picture 23. Christopher Nolan IMDb Ratings Over the Years

This line plot shows the IMDb ratings of Christopher Nolan's movies over the years. The x-axis represents the release years of his films, while the y-axis represents their IMDb ratings. Each point on the line marks a movie's rating, connected by a purple line to show the trend. The ratings fluctuate, peaking at 9.0 for The Dark Knight (2008) and showing a decline in later years, with Dunkirk (2017) having the lowest rating in the dataset.

2.37 Christopher Nolan's First Movie By Release Year

```
# Find Nolan's first movie by release year
first_nolan_movie = df[df['Director'] == 'Christopher Nolan'].nsmallest(1, 'Released_Year')

# Display result
print(first_nolan_movie[['Series_Title', 'Released_Year']])
```

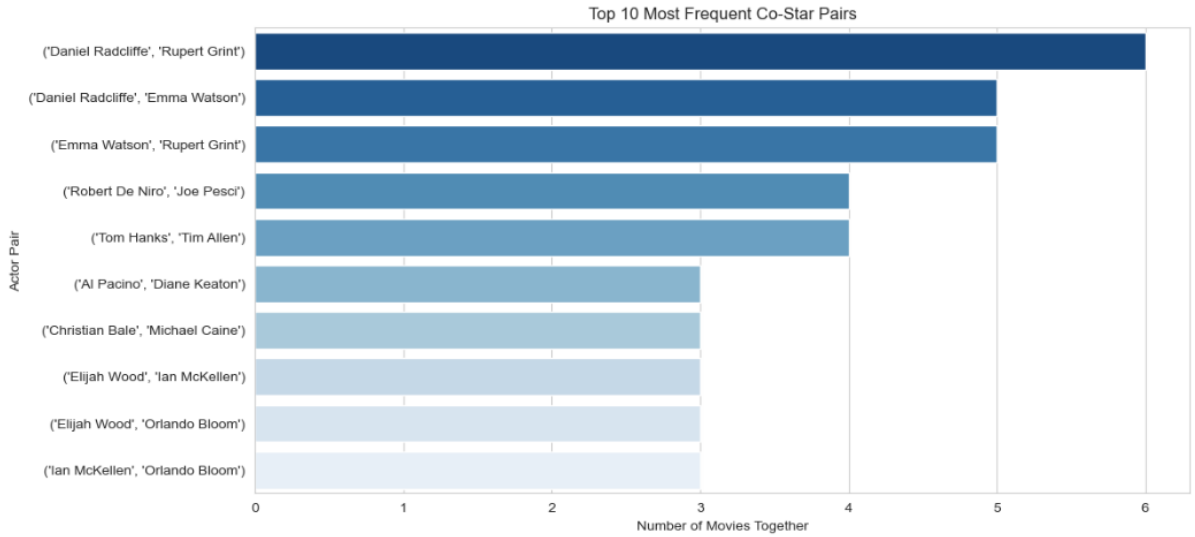
```
   Series_Title  Released_Year
69      Memento         2000.0
```

This code finds Christopher Nolan's first movie by release year. It filters the dataset for movies directed by him and selects the one with the earliest release year using nsmallest(1, 'Released_Year'). The output shows that Memento (released in 2000) is his first movie in the dataset.

```
from itertools import combinations
from collections import Counter

# Create actor pairs from each movie
actor_pairs = []
for i, row in df.iterrows():
    actors = [row['Star1'], row['Star2'], row['Star3'], row['Star4']]
    pairs = combinations(actors, 2)  # Create all 2-actor combinations
    actor_pairs.extend(pairs)

# Count occurrences
pair_counts = Counter(actor_pairs)
top_pairs = pair_counts.most_common(10)

# Convert to DataFrame
top_pairs_df = pd.DataFrame(top_pairs, columns=['Pair', 'Count'])

# Plot
plt.figure(figsize=(12, 6))
sns.barplot(y=top_pairs_df['Pair'].astype(str), x=top_pairs_df['Count'], palette='Blues_r')

plt.title('Top 10 Most Frequent Co-Star Pairs')
plt.xlabel('Number of Movies Together')
plt.ylabel('Actor Pair')

plt.show()
```

This code analyzes the most frequent co-star pairs in movies. It extracts the main actors from each movie and generates all possible two-actor combinations. It then counts how often each pair appears across different movies. The top 10 most frequent actor pairs are stored in a DataFrame and visualized using a bar plot, where the x-axis represents the number of movies they appeared in together, and the y-axis lists the actor pairs.

2.38 Top 10 Most Frequent Co-Star Pairs



Picture 24. Top 10 Most Frequent Co-Star Pairs

This bar chart shows the top 10 most frequent co-star pairs in movies.

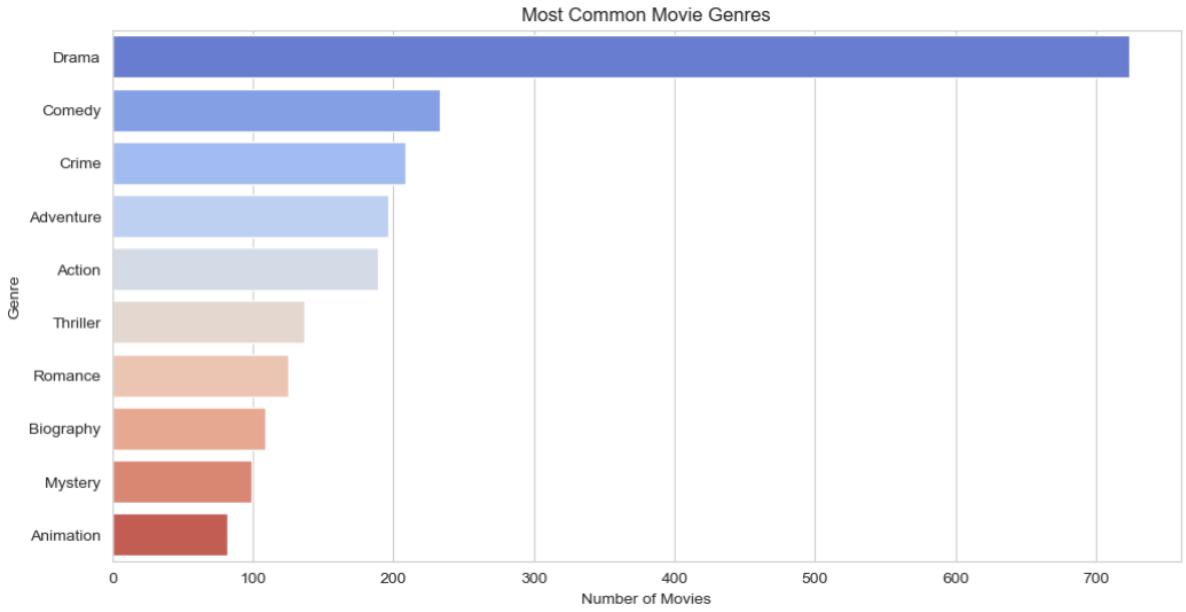
The x-axis represents the number of movies in which each pair appeared together, while the y-axis lists the actor pairs. Daniel Radcliffe and Rupert Grint have worked together the most, appearing in six movies. Other frequent co-star pairs include Emma Watson with both Daniel Radcliffe and Rupert Grint, as well as classic actor duos like Robert De Niro and Joe Pesci. The chart visually highlights the strongest on-screen partnerships in the dataset.

2.39 Most Common Movie Genres

```
df_exploded = df.assign(Genre=df['Genre'].str.split(', ')).explode('Genre')

top_genres = df_exploded['Genre'].value_counts().head(10)

plt.figure(figsize=(12, 6))
sns.barplot(y=top_genres.index, x=top_genres.values, palette='coolwarm')

plt.title('Most Common Movie Genres')
plt.xlabel('Number of Movies')
plt.ylabel('Genre')

plt.show()
```

This code analyzes the most common movie genres. It first splits the 'Genre' column where multiple genres are listed and then expands them into separate rows. After that, it counts the occurrences of each genre and selects the top 10 most frequent ones. A horizontal bar plot is then created, where the x-axis represents the number of movies, and the y-axis

represents the genres. The visualization helps identify which genres are most popular in the dataset.



Picture 25. Most Common Movie Genres

This bar chart shows the most common movie genres in the dataset. The x-axis represents the number of movies, while the y-axis lists different genres. Drama is the most frequent genre, appearing in the highest number of movies, followed by Comedy and Crime. The chart helps visualize which genres are most popular.

2.40 The Highest-rated genre and The Lowest-rated genre

```
: # Explode genres so each genre has its own row
df_exploded = df.assign(Genre=df['Genre'].str.split(', ')).explode('Genre')

# Calculate the average IMDb rating for each genre
genre_avg_rating = df_exploded.groupby('Genre')['IMDB_Rating'].mean().sort_values(ascending=False)

# Get the best and worst rated genres
best_genre = genre_avg_rating.idxmax()
worst_genre = genre_avg_rating.idxmin()

print(f"The highest-rated genre is: {best_genre} with an average rating of {genre_avg_rating.max():.2f}")
print(f"The lowest-rated genre is: {worst_genre} with an average rating of {genre_avg_rating.min():.2f}")
```

```
The highest-rated genre is: War with an average rating of 8.01
The lowest-rated genre is: Horror with an average rating of 7.89
```

This code analyzes the average IMDb ratings of different movie genres. First, it ensures that each genre has its own row. Then, it calculates the average rating for each genre and

sorts them in descending order. The best-rated genre is determined as "War" with an average rating of 8.01, while the lowest-rated genre is "Horror" with an average rating of 7.89.

2.41 Unique Genres

```
# Explode genres so each genre has its own row
df_exploded = df.assign(Genre=df['Genre'].str.split(', ')).explode('Genre')

# Get unique genres
unique_genres = df_exploded['Genre'].unique()

# Display all genres
print("All unique genres in the dataset:")
print(unique_genres)
```

```
All unique genres in the dataset:
['Drama' 'Crime' 'Action' 'Adventure' 'Biography' 'History' 'Sci-Fi'
 'Romance' 'Western' 'Fantasy' 'Comedy' 'Thriller' 'Animation' 'Family'
 'War' 'Mystery' 'Music' 'Horror' 'Musical' 'Film-Noir' 'Sport']
```

This code extracts all unique movie genres from a dataset. It first ensures that each genre appears in its own row by splitting and expanding them. Then, it finds all distinct genres in the dataset and prints them. The output lists various genres such as Drama, Crime, Action, Adventure, Sci-Fi, and many more.

2.42 Filtering Movies Where Leonardo DiCaprio Is One of the Stars

In this code, I counted the number of movies in each genre by exploding the genre column (in case movies have multiple genres). The output shows that Drama is the most common genre with 724 movies, followed by Comedy (233 movies) and Crime (209 movies). The least frequent genres are Musical (17 movies) and Sport (19 movies). This provides insights into the distribution of movie genres in the dataset.

```
# Count number of movies per genre
genre_counts = df_exploded['Genre'].value_counts()

# Display
print("\nNumber of movies in each genre:")
print(genre_counts)
```

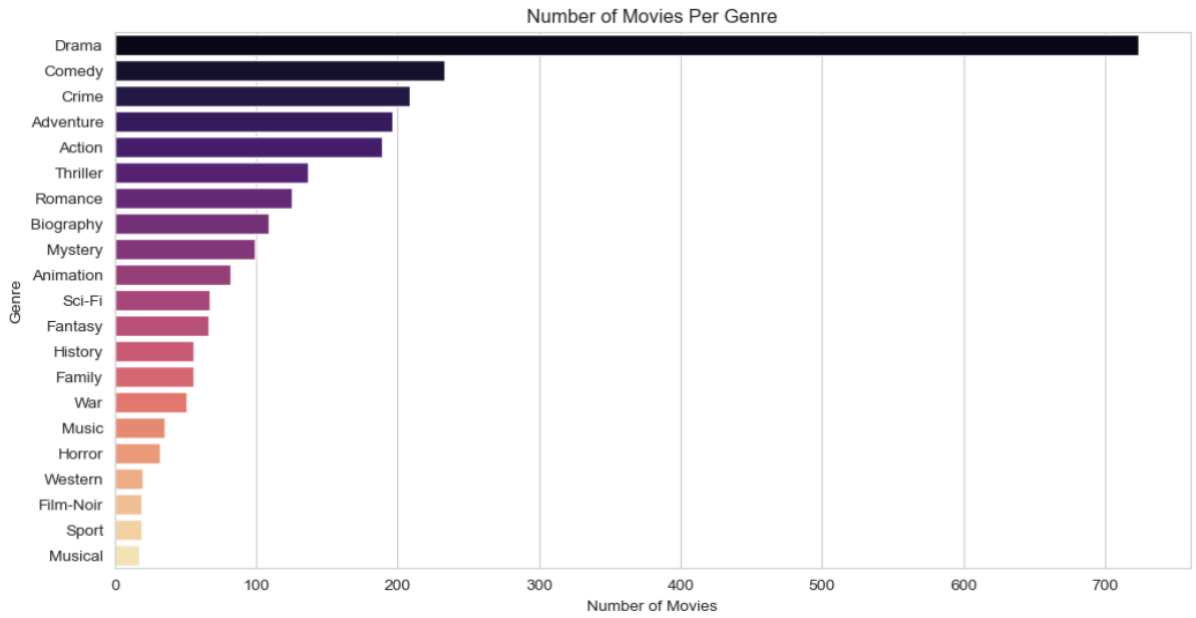
```
Number of movies in each genre:
Genre
Drama        724
Comedy       233
Crime        209
Adventure    196
Action       189
Thriller     137
Romance      125
Biography    109
Mystery       99
Animation     82
Sci-Fi        67
Fantasy       66
History       56
Family        56
War           51
Music         35
Horror        32
Western       20
Film-Noir     19
Sport         19
Musical       17
Name: count, dtype: int64
```

2.43 Number of Movies Per Genre

```
plt.figure(figsize=(12, 6))
sns.barplot(y=genre_counts.index, x=genre_counts.values, palette='magma')

plt.title('Number of Movies Per Genre')
plt.xlabel('Number of Movies')
plt.ylabel('Genre')

plt.show()
```

This code generates a horizontal bar plot using **Seaborn** to visualize the number of movies per genre. The magma color palette is used, which provides a **warm gradient color scheme**. The figure size is set to **(12,6)** for better readability.

The labels and title are clearly defined, making it an effective visualization for understanding the genre distribution in the dataset.



Picture 26. Number of Movies Per Genre

This bar plot effectively visualizes the number of movies per genre, with Drama having the highest count, followed by Comedy and Crime. The magma color palette helps distinguish between different genres, with darker shades representing higher counts and lighter shades indicating fewer movies. The horizontal layout improves readability, making it easy to compare genres.

2.44 Filtering Movies Where Leonardo DiCaprio Is In Any of Star Columns

```
# Filter movies where Leonardo DiCaprio is in any of the star columns
leo_movies = df[(df['Star1'] == 'Leonardo DiCaprio') |
                (df['Star2'] == 'Leonardo DiCaprio') |
                (df['Star3'] == 'Leonardo DiCaprio') |
                (df['Star4'] == 'Leonardo DiCaprio')]

# Explode genres so each genre gets its own row
leo_genres = leo_movies.assign(Genre=leo_movies['Genre'].str.split(', ')).explode('Genre')

# Get unique genres
leo_unique_genres = leo_genres['Genre'].unique()

# Display result
print("Genres Leonardo DiCaprio has acted in:")
print(leo_unique_genres)
```

```
Genres Leonardo DiCaprio has acted in:
['Action' 'Adventure' 'Sci-Fi' 'Crime' 'Drama' 'Thriller' 'Western'
 'Mystery' 'Biography' 'Romance' 'Comedy']
```

This code snippet extracts all movies featuring Leonardo DiCaprio by checking if his name appears in any of the four "Star" columns. Then, it processes the genres associated with these movies to determine the unique ones he has acted in. Leonardo DiCaprio has acted in 10 different genres, including Action, Adventure, Drama, Thriller, Crime, Sci-Fi, and Comedy.

2.45 Count of Movies Per Genre – Leonardo DiCaprio

```
# Count movies per genre for Leonardo DiCaprio
leo_genre_counts = leo_genres['Genre'].value_counts()

# Display
print("\nNumber of Leonardo DiCaprio movies in each genre:")
print(leo_genre_counts)
```

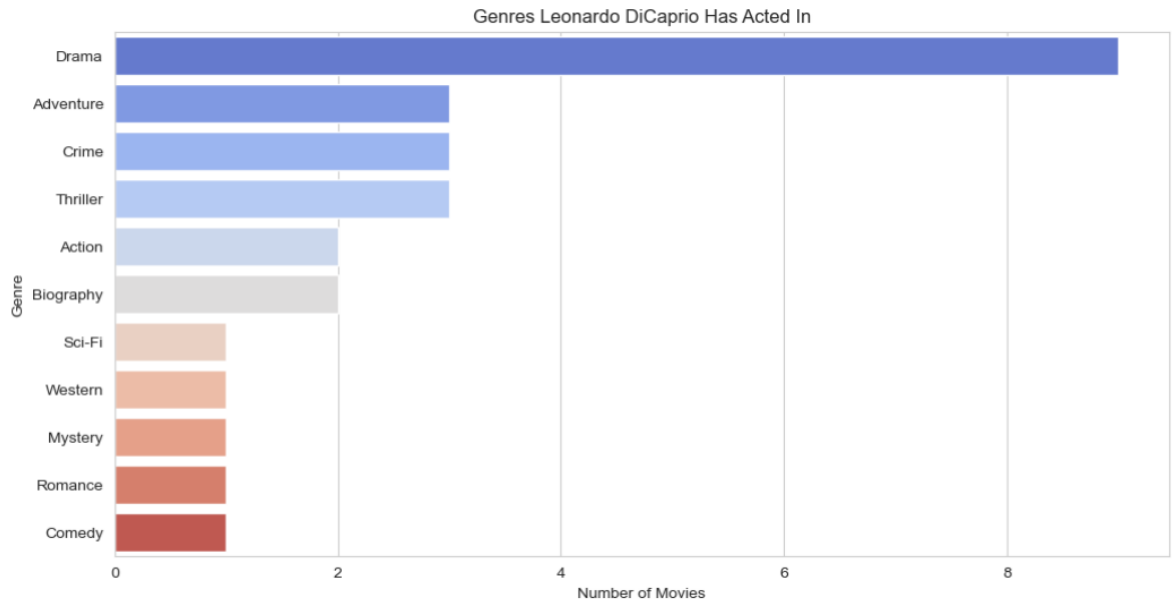
```
Number of Leonardo DiCaprio movies in each genre:
Genre
Drama        9
Adventure    3
Crime        3
Thriller     3
Action       2
Biography    2
Sci-Fi       1
Western      1
Mystery      1
Romance      1
Comedy       1
Name: count, dtype: int64
```

The code finds all movies with Leonardo DiCaprio, splits their genres into separate rows, and lists the unique genres he has acted in. Then, it counts how many movies he has done in each genre. The results show he acted in 9 drama movies, 3 adventure movies, and others.

2.46 Genres Leonardo DiCaprio Has Acted In

```
plt.figure(figsize=(12, 6))
sns.barplot(y=leo_genre_counts.index, x=leo_genre_counts.values, palette='coolwarm')

plt.title('Genres Leonardo DiCaprio Has Acted In')
plt.xlabel('Number of Movies')
plt.ylabel('Genre')

plt.show()
```

This code creates a bar chart showing the number of movies Leonardo DiCaprio has acted in for each genre. It uses the "coolwarm" color palette and labels the axes properly.



Picture 27. Genres Leonardo DiCaprio Has Acted In

This bar chart visualizes the number of movies Leonardo DiCaprio has acted in for each genre. Drama is the most frequent genre, followed by Adventure, Crime, and Thriller. The "coolwarm" color palette differentiates genres with varying shades.

2.47 Filtering Movies Where Richard Gere Is In Any of Star Columns

```
# Filter movies where Richard Gere is in any of the star columns
gere_movies = df[(df['Star1'] == 'Richard Gere') |
                 (df['Star2'] == 'Richard Gere') |
                 (df['Star3'] == 'Richard Gere') |
                 (df['Star4'] == 'Richard Gere')]

# Explode genres so each genre gets its own row
gere_genres = gere_movies.assign(Genre=gere_movies['Genre'].str.split(', ')).explode('Genre')

# Get unique genres
gere_unique_genres = gere_genres['Genre'].unique()

# Display result
print("Genres Richard Gere has acted in:")
print(gere_unique_genres)
```

```
Genres Richard Gere has acted in:
['Biography' 'Drama' 'Family' 'Romance' 'Crime' 'Mystery']
```

This code filters movies where Richard Gere appears as one of the stars, extracts the genres he has acted in, and lists the unique genres. The result shows that Richard Gere has acted in Biography, Drama, Family, Romance, Crime, and Mystery genres.

2.48 Genres Richard Gere Has Acted In

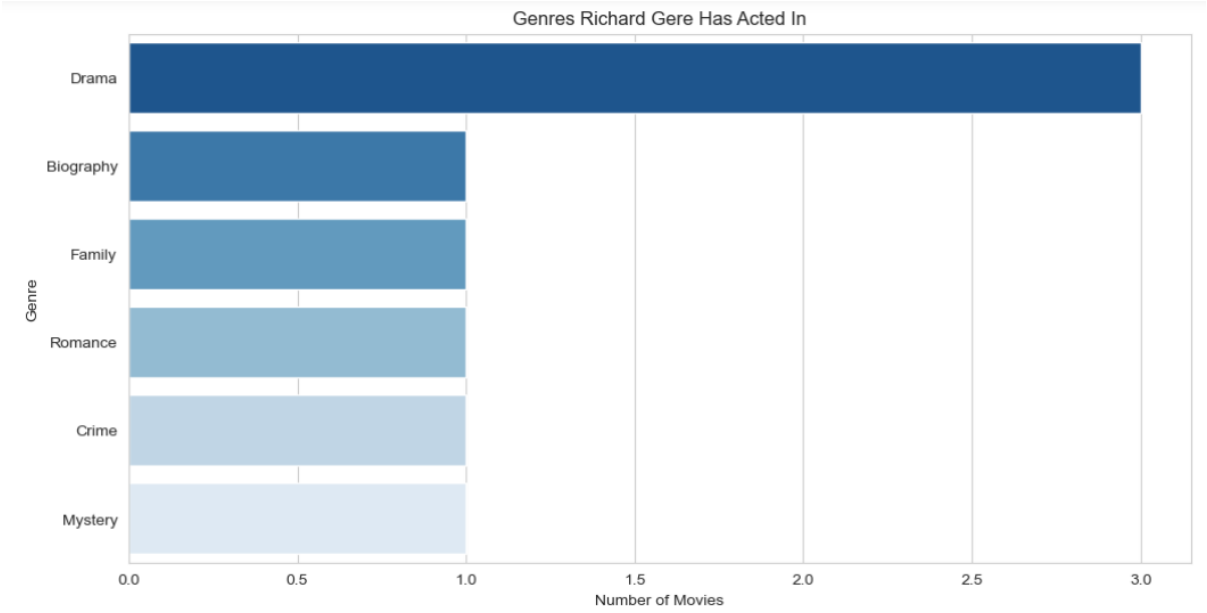
```
# Count movies per genre for Richard Gere
gere_genre_counts = gere_genres['Genre'].value_counts()

# Display
print("\nNumber of Richard Gere movies in each genre:")
print(gere_genre_counts)
```

```
Number of Richard Gere movies in each genre:
Genre
Drama        3
Biography    1
Family       1
Romance      1
Crime        1
Mystery      1
Name: count, dtype: int64
```

This code counts the number of movies Richard Gere has acted in for each genre. The result shows that he has acted in 3 Drama movies, and 1 movie each in Biography, Family, Romance, Crime, and Mystery genres.

```
plt.figure(figsize=(12, 6))
sns.barplot(y=gere_genre_counts.index, x=gere_genre_counts.values, palette='Blues_r')

plt.title('Genres Richard Gere Has Acted In')
plt.xlabel('Number of Movies')
plt.ylabel('Genre')

plt.show()
```

I created a bar chart that visually represents the number of movies Richard Gere has acted in for each genre. It uses the Seaborn library to generate the plot, with the genres listed on the y-axis and the number of movies on the x-axis. The palette 'Blues_r' is applied, which provides a reversed blue color gradient, making the bars darker for higher values and lighter for lower values. The figure size is set to (12,6), ensuring clarity. The title 'Genres Richard Gere Has Acted In' is displayed, along with labeled axes for better readability.



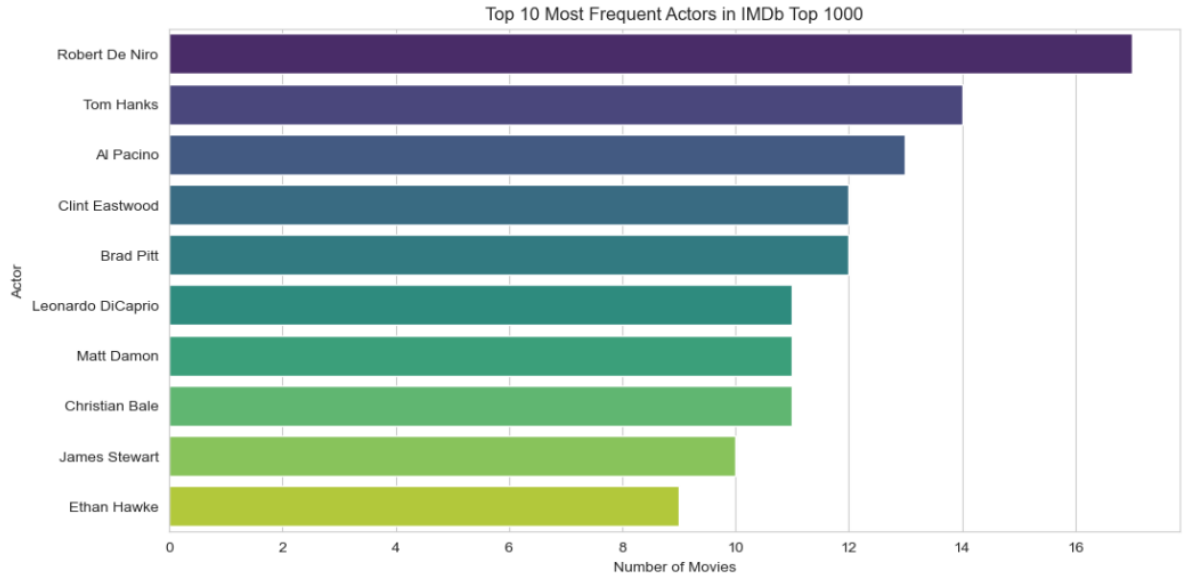
Picture 28. Genres Richard Gere Has Acted In

This bar chart visually represents the number of movies Richard Gere has acted in, categorized by genre. The y-axis lists different genres, while the x-axis represents the number of movies in each genre. The reversed blue color palette emphasizes the distribution, with darker shades indicating higher counts. Drama stands out as the most frequent genre, with three movies, while the other genres have only one each.

2.49 Top 10 Most Frequent Actors in IMDb Top 1000

```
# Count occurrences of each actor in the dataset
actor_counts = all_actors.value_counts().head(10)

# Plot
plt.figure(figsize=(12, 6))
sns.barplot(y=actor_counts.index, x=actor_counts.values, palette='viridis')

plt.title('Top 10 Most Frequent Actors in IMDb Top 1000')
plt.xlabel('Number of Movies')
plt.ylabel('Actor')

plt.show()
```

This code analyzes the top 10 most frequent actors in the IMDb Top 1000 movies by counting how many times each actor appears, selecting the top 10, and visualizing the results using a bar chart with actor names on the y-axis and their movie count on the x-axis, styled with the "viridis" color scheme for better visuals.



Picture 29. Top 10 Most Frequent Actors

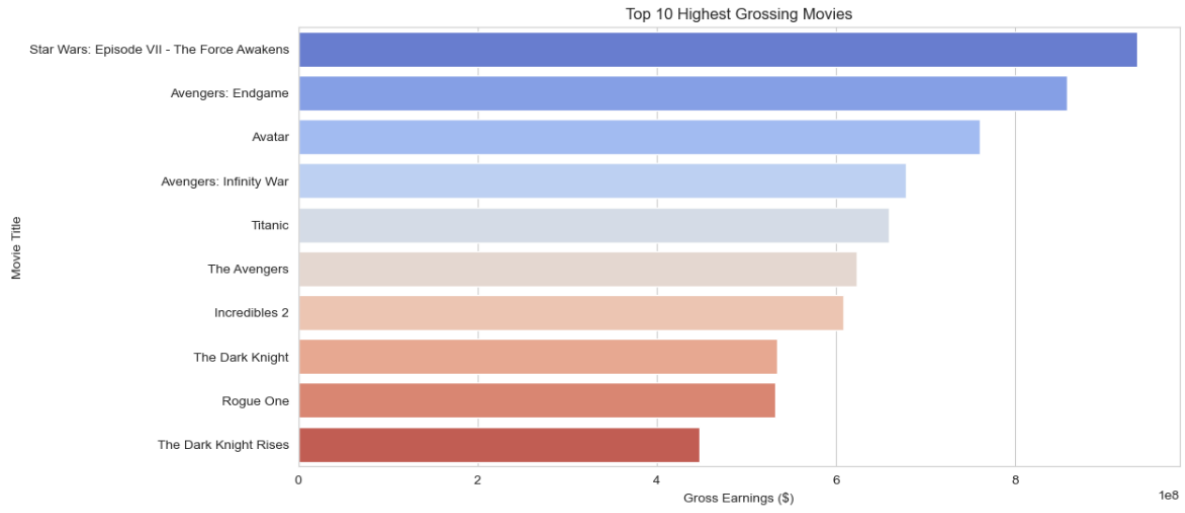
This bar chart shows the top 10 most frequent actors in the IMDb Top 1000 movies. The x-axis represents the number of movies each actor has appeared in, while the y-axis lists the actors' names. The chart uses the "viridis" color palette, where darker shades indicate higher frequencies.

Robert De Niro appears the most, followed by Tom Hanks and Al Pacino. Other notable actors in the list include Clint Eastwood, Brad Pitt, and Leonardo DiCaprio. The visualization highlights actors who have consistently played roles in top-rated films.

2.50 Top 10 Highest Grossing Movies

```
df['Gross'] = df['Gross'].str.replace(',', '').astype(float)

top_grossing = df.nlargest(10, 'Gross')[['Series_Title', 'Gross']]

plt.figure(figsize=(12, 6))
sns.barplot(y=top_grossing['Series_Title'], x=top_grossing['Gross'], palette='coolwarm')

plt.title('Top 10 Highest Grossing Movies')
plt.xlabel('Gross Earnings ($)')
plt.ylabel('Movie Title')

plt.show()
```

This code processes and visualizes the top 10 highest-grossing movies by converting the "Gross" column into a numeric format, selecting the highest earners, and creating a bar chart with movie titles on the y-axis and earnings on the x-axis using the "coolwarm" color scheme, while labeling the axes and adding a title for clarity.



Picture 30. Top 10 Highest Grossing Movies

This bar chart visualizes the top 10 highest-grossing movies, with "Star Wars: Episode VII - The Force Awakens" at the top, followed by "Avengers: Endgame" and "Avatar," displaying their earnings in descending order using a gradient color scheme from cool to warm for emphasis. The x-axis represents gross earnings in dollars, while the y-axis lists the movie titles.

2.51 Movies Released In The 1990s

```
# Filter movies released in the 1990s
movies_90s = df[(df['Released_Year'] >= 1990) & (df['Released_Year'] <= 1999)]

# Find the movie with the most votes
most_popular_90s = movies_90s.nlargest(1, 'No_of_Votes')[['Series_Title', 'Released_Year', 'IMDB_Rating', 'No_of_Votes'

# Display result
print("Most popular movie from the 1990s:")
print(most_popular_90s)
```

```
Most popular movie from the 1990s:
              Series_Title  Released_Year  IMDB_Rating  No_of_Votes
0  The Shawshank Redemption         1994.0          9.3      2343110
```

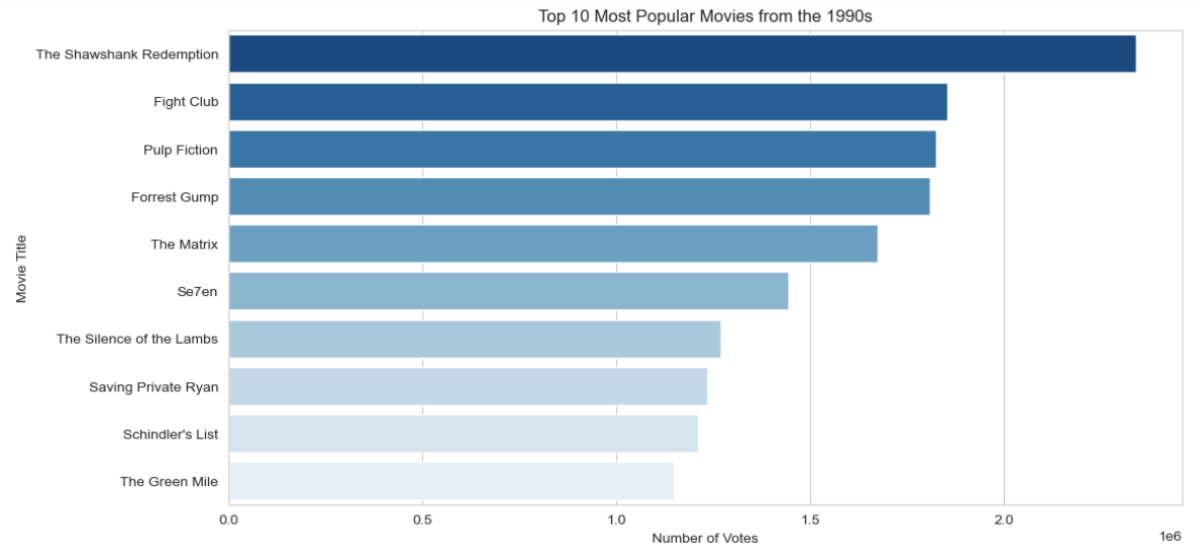
2.52 Top 10 Most Voted Movies From The 1990s

This code filters movies released in the 1990s and selects the one with the highest number of votes. It finds that "The Shawshank Redemption" (1994) is the most popular movie of the decade, with an IMDb rating of 9.3 and 2,343,110 votes. The result is printed as a small table displaying the title, release year, rating, and vote count.

```
# Get the top 10 most voted movies from the 1990s
top_90s_movies = movies_90s.nlargest(10, 'No_of_Votes')[['Series_Title', 'No_of_Votes']]

plt.figure(figsize=(12, 6))
sns.barplot(y=top_90s_movies['Series_Title'], x=top_90s_movies['No_of_Votes'], palette='Blues_r')

plt.title('Top 10 Most Popular Movies from the 1990s')
plt.xlabel('Number of Votes')
plt.ylabel('Movie Title')

plt.show()
```

This code selects the top 10 most voted movies from the 1990s and visualizes them using a horizontal bar chart. It first filters the dataset to find the movies with the highest number of votes, then plots them with movie titles on the y-axis and the number of votes on the x-

axis. The chart uses the "Blues_r" color palette and has a title, x-label, and y-label for clarity.



Picture 31. Top 10 Most Popular Movies from the 1990s

This bar chart visualizes the top 10 most popular movies from the 1990s based on the number of votes. "The Shawshank Redemption" received the highest number of votes, followed by "Fight Club" and "Pulp Fiction." The movies are ranked in descending order, with the number of votes displayed on the x-axis and movie titles on the y-axis. The color scheme (Blues_r) represents the intensity of popularity.

2.53 Filtering Movies Where Tom Cruise Is In Any of Star Columns

```
# Filter movies where Tom Cruise is in any of the star columns
tom_cruise_movies = df[(df['Star1'] == 'Tom Cruise') |
                       (df['Star2'] == 'Tom Cruise') |
                       (df['Star3'] == 'Tom Cruise') |
                       (df['Star4'] == 'Tom Cruise')]

# Filter only movies from the 1990s
tom_cruise_90s = tom_cruise_movies[(tom_cruise_movies['Released_Year'] >= 1990) &
                                   (tom_cruise_movies['Released_Year'] <= 1999)]

# Find the highest-rated Tom Cruise movie in the 90s
best_tom_cruise_90s = tom_cruise_90s.nlargest(1, 'IMDB_Rating')[['Series_Title', 'Released_Year', 'IMDB_Rating']]

# Display result
print("Tom Cruise's highest-rated movie in the 1990s:")
print(best_tom_cruise_90s)
```

```
Tom Cruise's highest-rated movie in the 1990s:
    Series_Title  Released_Year  IMDB_Rating
383     Magnolia         1999.0          8.0
```

This code filters movies where Tom Cruise is listed as one of the stars and further narrows them down to those released in the 1990s. It then selects the highest-rated Tom Cruise movie from that decade based on IMDb ratings. The result shows that "Magnolia" (1999) was his highest-rated movie in the 1990s, with an IMDb rating of 8.0.

2.54 Longest Movie Titles

```
df['Title_Length'] = df['Series_Title'].apply(len)

# Get the longest movie titles
longest_titles = df.nlargest(10, 'Title_Length')[['Series_Title', 'Title_Length']]

# Display
print("Movies with the longest titles:")
print(longest_titles)
```

```
Movies with the longest titles:
                                        Series_Title  Title_Length
78   Dr. Strangelove or: How I Learned to Stop Worr...            68
246  Shin seiki Evangelion Gekijô-ban: Air/Magokoro...            58
376  Pirates of the Caribbean: The Curse of the Bla...            54
10   The Lord of the Rings: The Fellowship of the Ring            49
610    Das weiße Band - Eine deutsche Kindergeschichte            47
733    Birdman or (The Unexpected Virtue of Ignorance)            47
16      Star Wars: Episode V - The Empire Strikes Back            46
977     The Naked Gun: From the Files of Police Squad!            46
5        The Lord of the Rings: The Return of the King            45
226       Harry Potter and the Deathly Hallows: Part 2            44
```

This code calculates the length of each movie title and selects the 10 movies with the longest titles. The longest title is "Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb" with 68 characters. Other long titles include entries from "Pirates of the Caribbean," "Lord of the Rings," and "Harry Potter" franchises.

2.55 Movie With The Shortest Title

```
# Create a new column for title length
df['Title_Length'] = df['Series_Title'].apply(len)

# Find the movie with the shortest title
shortest_title_movie = df.nsmallest(1, 'Title_Length')[['Series_Title', 'Released_Year', 'IMDB_Rating', 'Title_Length']]

# Display result
print("Movie with the shortest title:")
print(shortest_title_movie)
```

```
Movie with the shortest title:
    Series_Title  Released_Year  IMDB_Rating  Title_Length
146           Up         2009.0          8.2             2
```

This code finds the movie with the shortest title in the dataset. The result shows that "Up" (2009) has the shortest title, with only 2 characters. It also has an IMDB rating of 8.2.