IMDB DATA ANALYSIS

ABSTRACT:

This report presents an analysis of the IMDb TOP 1000 movie dataset using Python's Pandas and Matplotlib libraries. The dataset contains information about the top-rated movies on IMDb, including details such as series title, director, genre, IMDb-rating, and runtime. The objective of this analysis is to gain insights into the movie dataset, such as the distribution of genres, IMDb-ratings, year-wise movie count,top 10 highest-rated movies, top 10 longest movies, directors with the most movies, and directors with the longest movies. The report provides a step-by-step explanation of the Python code used for data analysis, along with visualizations to illustrate the results. Additionally, it discusses potential areas for future work and extensions to this analysis.

Keywords: *IMDb, movie dataset, data analysis, Pandas, Matplotlib, visualization, movie genres, movie ratings, movie directors.*

INTRODUCTION:

The IMDb TOP 1000 movie dataset contains a rich set of information about the highest-rated movies on IMDb, making it an excellent resource for exploring and analyzing trends in the film industry. This report aims to delve into the dataset and extract meaningful insights to better understand the characteristics of top-rated movies. By analyzing aspects like movie genres, ratings, and directors, we can gain valuable knowledge about the preferences of IMDb users and the contributions of directors to the movie industry.

IMPORTING LIBRARIES

Importing the necessary libraries such as Python Pandas for Data Analysis and Matplotlib for Data Visualization.

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

LOADING DATA

The code begins by loading the IMDb TOP 1000 movie dataset from a CSV file using the 'pd.read_csv()' function from the Pandas library. The first few rows of the dataset are displayed to provide an overview.

```
# Load the IMDb Top 1000 movie dataset from CSV
imdb_data = pd.read_csv('imdb_top_1000.csv')
```

DATA EXPLORATION

Data exploration involves getting a high-level overview of the dataset and understanding its structure. In this code, we start by loading the IMDb TOP 1000 movie dataset from the CSV file and displaying the first few rows using the 'head()' method.

The 'pd.read_csv()' function loads the data from the CSV file into a Pandas DataFrame called imdb_df. The 'head()' method displays the first 5 rows of the DataFrame by default, giving us a quick look at the dataset's columns and the data it contains.

```
# Basic data exploration
print("Sample data:")
print(imdb_data.head())
```

```
Sample data:
                                         Poster_Link  \
0  https://m.media-amazon.com/images/M/MV5BMDFkYT...
1  https://m.media-amazon.com/images/M/MV5BM2MyNj...
2  https://m.media-amazon.com/images/M/MV5BMTMxNT...
3  https://m.media-amazon.com/images/M/MV5BMWMwMG...
4  https://m.media-amazon.com/images/M/MV5BMWU4N2...

               Series_Title Released_Year Certificate  Runtime  \
0  The Shawshank Redemption          1994           A  142 min
1             The Godfather          1972           A  175 min
2           The Dark Knight          2008          UA  152 min
3    The Godfather: Part II          1974           A  202 min
4              12 Angry Men          1957           U   96 min

                  Genre  IMDB_Rating  \
0                 Drama          9.3
1          Crime, Drama          9.2
2  Action, Crime, Drama          9.0
3          Crime, Drama          9.0
4          Crime, Drama          9.0

                                            Overview  Meta_score  \
0  Two imprisoned men bond over a number of years...        80.0
1  An organized crime dynasty's aging patriarch t...       100.0
2  When the menace known as the Joker wreaks havo...        84.0
3  The early life and career of Vito Corleone in ...        90.0
4  A jury holdout attempts to prevent a miscarria...        96.0

               Director           Star1           Star2          Star3  \
0        Frank Darabont     Tim Robbins  Morgan Freeman     Bob Gunton
1  Francis Ford Coppola   Marlon Brando        Al Pacino     James Caan
2     Christopher Nolan  Christian Bale     Heath Ledger  Aaron Eckhart
3  Francis Ford Coppola       Al Pacino  Robert De Niro  Robert Duvall
4          Sidney Lumet     Henry Fonda     Lee J. Cobb  Martin Balsam

            Star4  No_of_Votes        Gross
0  William Sadler      2343110   28,341,469
1    Diane Keaton      1620367  134,966,411
2   Michael Caine      2303232  534,858,444
3    Diane Keaton      1129952   57,300,000
4    John Fiedler       689845    4,360,000
```

DATA INFORMATION

Data information provides essential details about the dataset, such as the number of rows and columns, data types of each column, and the presence of missing values. The code snippet below demonstrates how to access this information using the 'info()' method.

The 'info()' method gives a concise summary of the DataFrame, showing the number of non-null values, data types, and memory usage. It is useful for identifying missing values (non-null counts) and ensuring that the data types are appropriate for further analysis.

```
# Data information
print("\nData Information:")
print(imdb_data.info())
```

```
Data Information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 16 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
```

```
 0   Poster_Link    1000 non-null   object 
 1   Series_Title   1000 non-null   object 
 2   Released_Year  1000 non-null   object 
 3   Certificate    899 non-null    object 
 4   Runtime        1000 non-null   object 
 5   Genre          1000 non-null   object 
 6   IMDB_Rating    1000 non-null   float64
 7   Overview       1000 non-null   object 
 8   Meta_score     843 non-null    float64
 9   Director       1000 non-null   object 
 10  Star1          1000 non-null   object 
 11  Star2          1000 non-null   object 
 12  Star3          1000 non-null   object 
 13  Star4          1000 non-null   object 
 14  No_of_Votes    1000 non-null   int64  
 15  Gross          831 non-null    object 
dtypes: float64(2), int64(1), object(13)
memory usage: 125.1+ KB
None
```

STATISTICS SUMMARY

To get an overview of the numerical attributes in the dataset, we can calculate basic summary statistics using the 'describe()' method. This provides statistics like mean, standard deviation, minimum, 25th percentile (Q1), median (50th percentile or Q2), 75th percentile (Q3), and maximum for each numeric column.

The 'describe()' method generates a summary statistics table that includes count, mean, standard deviation, min, quartiles, and max values for all numerical columns in the DataFrame. This information helps in understanding the distribution and range of numerical data in the dataset.

```
# Summary statistics
print("\nSummary Statistics:")
print(imdb_data.describe())
```

```
Summary Statistics:
       IMDB_Rating  Meta_score   No_of_Votes
count  1000.000000  843.000000  1.000000e+03
mean      7.949300   77.971530  2.736929e+05
std       0.275491   12.376099  3.273727e+05
min       7.600000   28.000000  2.508800e+04
25%       7.700000   70.000000  5.552625e+04
50%       7.900000   79.000000  1.385485e+05
75%       8.100000   87.000000  3.741612e+05
max       9.300000  100.000000  2.343110e+06
```

DATA ANALYSIS

Data analysis involves examining and interpreting the data to draw meaningful insights, identify patterns, and answer specific research questions. In the context of the IMDb TOP 1000 movie dataset, we can perform various types of data analysis to explore trends, relationships, and characteristics of the movies. Below are some examples of data analysis that can be conducted:

1. No. of movies in each Genre.
2. Average Rating of movies.
3. TOP 10 Highest-rated movies.
4. TOP 10 Longest mocvies.
5. TOP Directors with most movies.

6. Year-wise movie count.
7. Directors with longest movies.

1. NO. OF MOVIES IN EACH GENRE

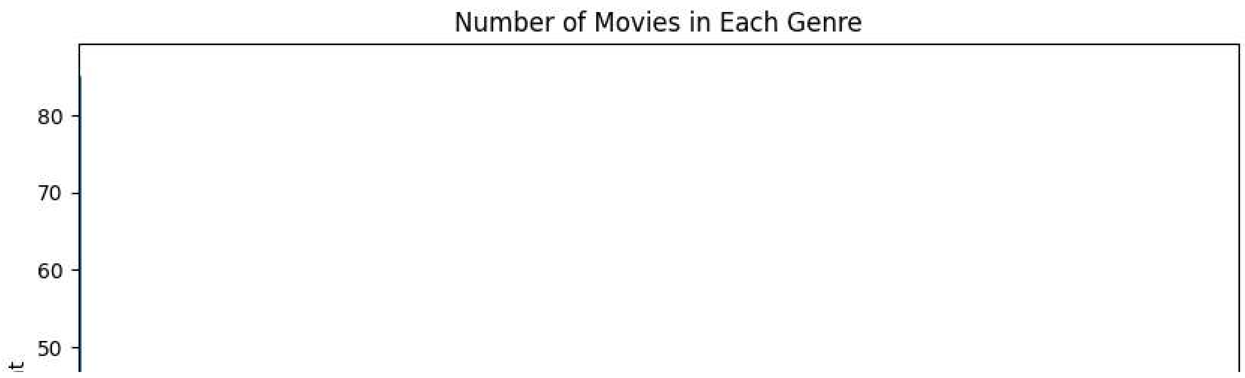
This analysis aims to count the number of movies in each genre to understand the distribution of genres among the top-rated movies.

The code below uses the 'value_counts()' function on the 'Genre' column of the IMDb DataFrame ('imdb_df') to count the occurrences of each genre. The result is stored in the genre_counts variable.

```
# Find the number of movies in each Genre
Genre_counts = imdb_data['Genre'].value_counts()
print("\nNumber of movies in each Genre:")
print(Genre_counts)
```

```
Number of movies in each Genre:
Drama                        85
Drama, Romance               37
Comedy, Drama                35
Comedy, Drama, Romance       31
Action, Crime, Drama         30
                             ..
Adventure, Thriller           1
Animation, Action, Sci-Fi     1
Action, Crime, Comedy         1
Animation, Crime, Mystery     1
Adventure, Comedy, War        1
Name: Genre, Length: 202, dtype: int64
```

```
# Plot the number of movies in each genre
Genre_counts.plot(kind='bar', figsize=(10, 6))
plt.title('Number of Movies in Each Genre')
plt.xlabel('Genre')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```



2. AVERAGE RATING OF MOVIES

This analysis calculates the average rating of movies in the dataset.

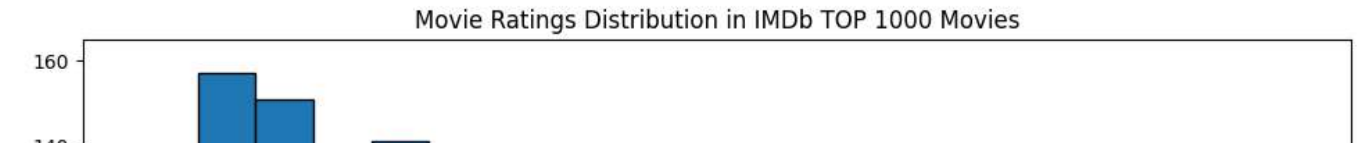
The code below calculates the mean (average) f the 'Rating' column of the IMDb DataFrame ('imdb_df'). The result is stored in the 'average_rating' variable.

```
# Average rating of movies
average_rating = imdb_data['IMDB_Rating'].mean()
print("\nAverage rating of movies:", average_rating)
```

```
     Average rating of movies: 7.949299999999999
```

MOVIES RATING DISTRIBUTION PLOT

```
# Movie Ratings Distribution
plt.figure(figsize=(10, 6))
plt.hist(imdb_df['IMDB_Rating'], bins=20, edgecolor='black')
plt.xlabel('IMDB_Rating')
plt.ylabel('Number of Movies')
plt.title('Movie Ratings Distribution in IMDb TOP 1000 Movies')
plt.tight_layout()
plt.show()
```



3. TOP 10 HIGHEST-RATED MOVIES

This analysis identifies the top 10 highest-rated movies based on their ratings.

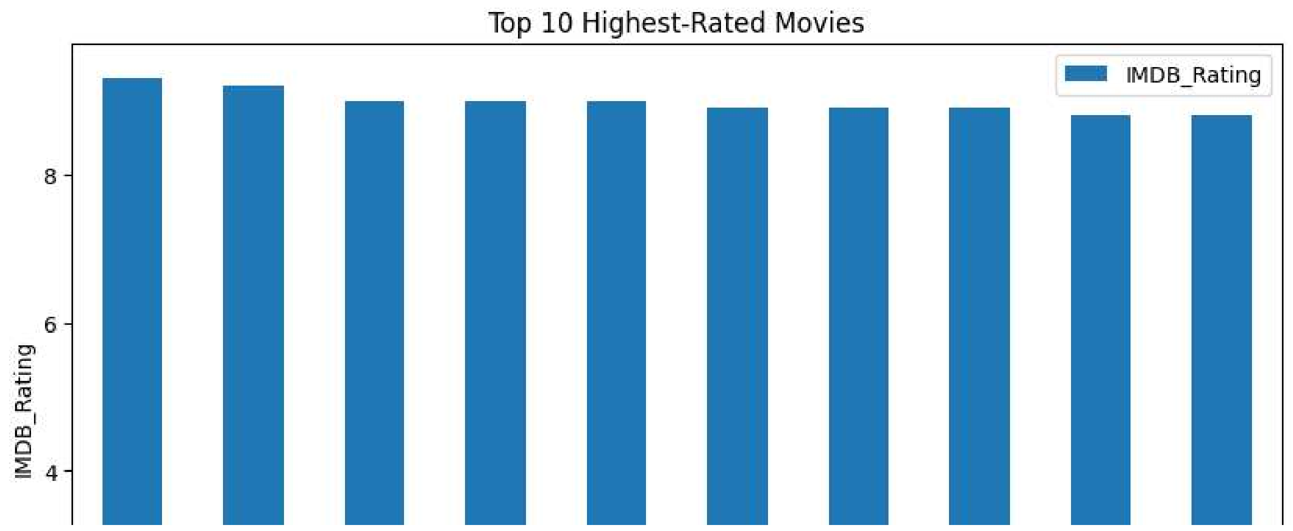
The code below uses the 'nlargest()' function to retrieve the 10 rows with the highest values in the 'Rating' column of the IMDb DataFrame ('imdb_df'). The result is stored in the 'top_10_movies' variable.

```
# Find the top 10 highest-rated movies
top_rated_movies = imdb_data.nlargest(10, 'IMDB_Rating')[['Series_Title', 'IMDB_Rating']]
print("\nTop 10 highest-rated movies:")
print(top_rated_movies)
```

```
Top 10 highest-rated movies:
                                    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
```

TOP 10 HIGHEST-RATED MOVIES PLOT

```
# Plot the top 10 highest-rated movies
top_rated_movies.plot(x='Series_Title', y='IMDB_Rating', kind='bar', figsize=(10, 6))
plt.title('Top 10 Highest-Rated Movies')
plt.xlabel('Series Title')
plt.ylabel('IMDB_Rating')
plt.xticks(rotation=45)
plt.show()
```



4. TOP 10 LONGEST MOVIES

This analysis finds the top 10 longest movies in terms of runtime.

The code below uses the 'nlargest()' function to retrieve the 10 rows with the highest values in the 'Runtime (Minutes)' column of the IMDb DataFrame ('imdb_df'). The result is stored in the 'top_10_longest_movies' variable.

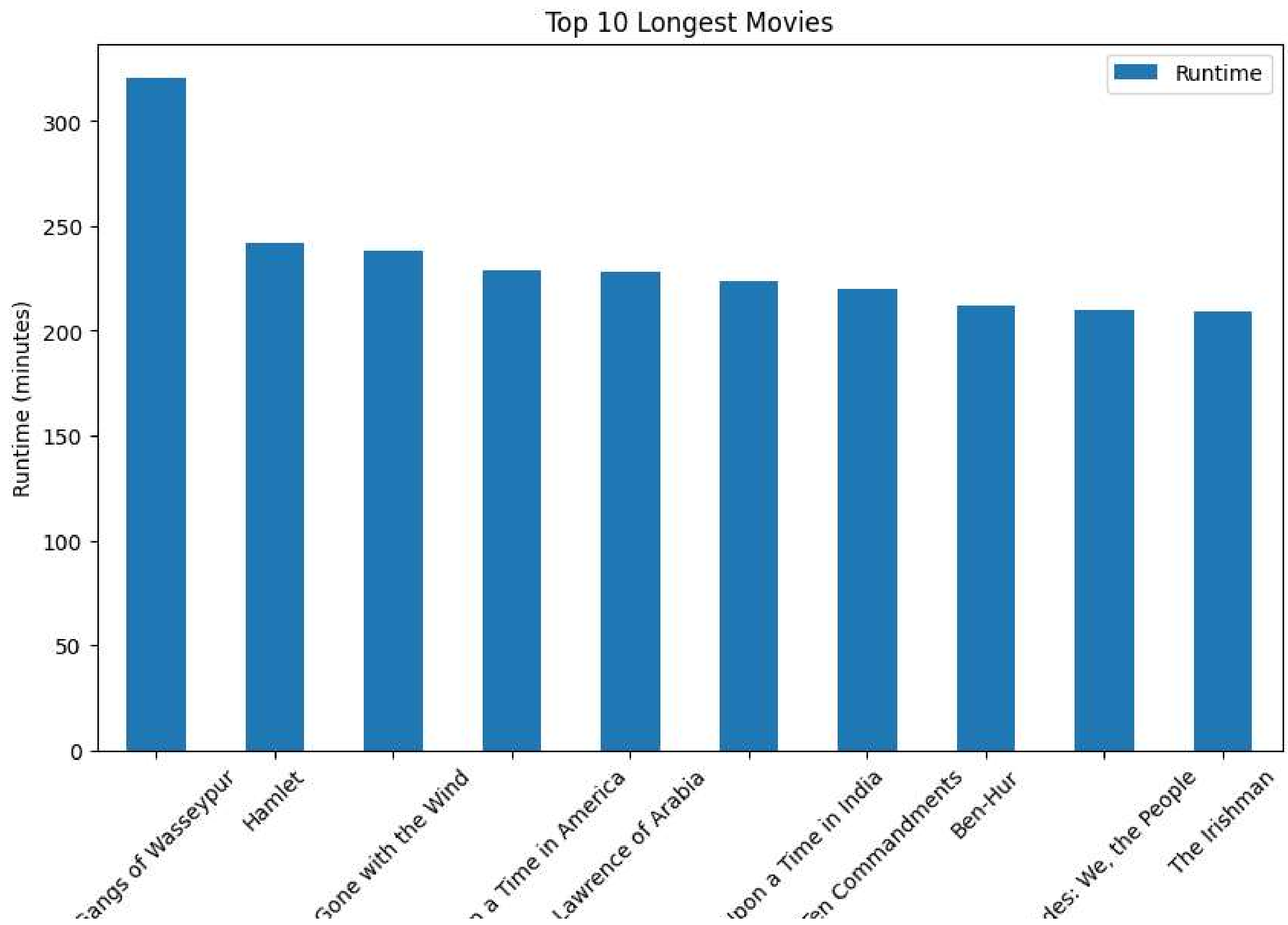
```
# Convert the 'Runtime' column to numeric (integer) and handle missing values
imdb_data['Runtime'] = pd.to_numeric(imdb_data['Runtime'].str.split().str[0], errors='coerce')

# Find the top 10 longest movies
top_longest_movies = imdb_data.nlargest(10, 'Runtime')[['Series_Title', 'Runtime']]
print("\nTop 10 longest movies:")
print(top_longest_movies)
```

```
Top 10 longest movies:
                        Series_Title  Runtime
140                Gangs of Wasseypur      321
812                            Hamlet      242
314                Gone with the Wind      238
71          Once Upon a Time in America      229
116                Lawrence of Arabia      228
247  Lagaan: Once Upon a Time in India      224
552                 The Ten Commandments      220
300                           Ben-Hur      212
156             Swades: We, the People      210
484                       The Irishman      209
```

TOP 10 LONGEST MOVIES PLOT

```
# Plot the top 10 longest movies
top_longest_movies.plot(x='Series_Title', y='Runtime', kind='bar', figsize=(10, 6))
plt.title('Top 10 Longest Movies')
plt.xlabel('Series Title')
plt.ylabel('Runtime (minutes)')
plt.xticks(rotation=45)
plt.show()
```



5. TOP DIRECTORS WITH MOST MOVIES

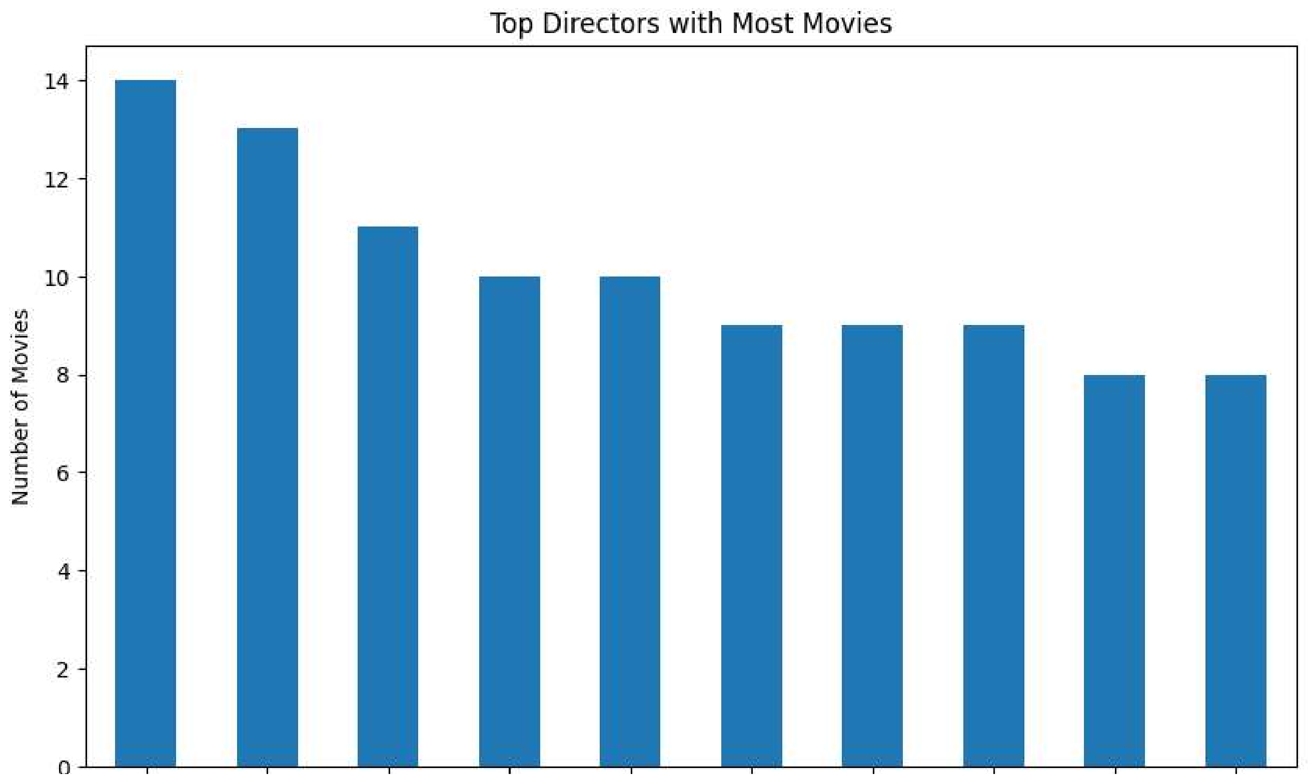
This analysis identifies the top directors who have directed the most number of movies in the IMDb TOP 1000 dataset.

The code below uses the 'value_counts()' function on the 'Director' column of the IMDb DataFrame ('imdb_df') to count the occurrences of each director. The 'head(10)' method retrieves the top 10 directors with the most movies, and the result is stored in the 'director_counts' variable.

```
# Find the director with the most movies
director_counts = imdb_data['Director'].value_counts()
most_movies_director = director_counts.idxmax()
```

TOP DIRECTORS WITH MOST MOVIES PLOT

```
# Plot the top directors with the most movies
top_directors = director_counts.nlargest(10)  # Top 10 directors
top_directors.plot(kind='bar', figsize=(10, 6))
plt.title('Top Directors with Most Movies')
plt.xlabel('Director')
plt.ylabel('Number of Movies')
plt.xticks(rotation=45)
plt.show()
```



```
# Find the director with the most movies
director_counts = imdb_data['Director'].value_counts()
most_movies_director = director_counts.idxmax()
print("\nDirector with the most movies:", most_movies_director)
```

```
    Director with the most movies: Alfred Hitchcock
```

6. YEAR-WISE MOVIE COUNT

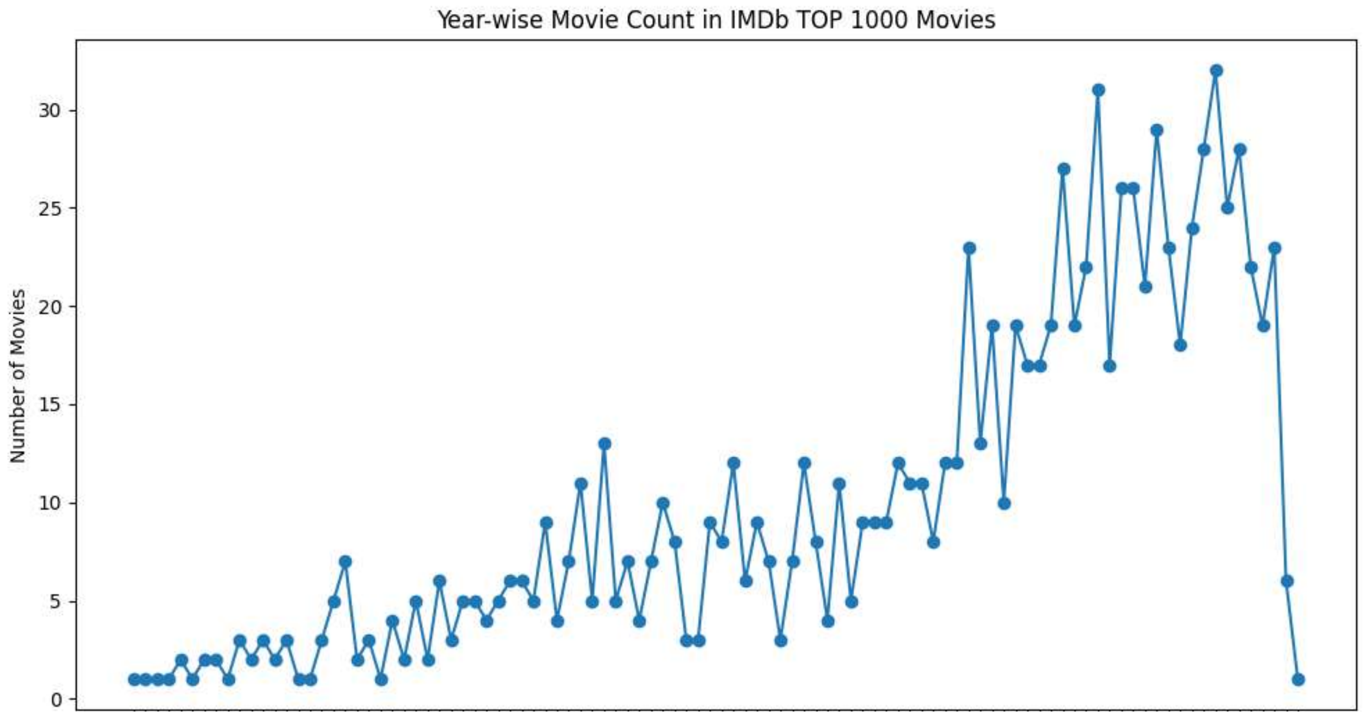
This analysis calculates the number of movies released in each year and visualizes the year-wise movie count using a line plot.

The code below groups the movies in the IMDb DataFrame ('imdb_df') by release year ('Year' column) using 'value_counts()'. Then, it sorts the result by year using 'sort_index()', and the year-wise movie counts are stored in the 'year_counts' variable.

```
# Year-wise Movie Count
year_counts = imdb_df['Released_Year'].value_counts().sort_index()
```

YEAR-WISE MOVIE COUNT PLOT

```
# Plot Year-wise Movie Count
plt.figure(figsize=(10, 6))
plt.plot(year_counts.index, year_counts.values, marker='o')
plt.xlabel('Released_Year')
plt.ylabel('Number of Movies')
plt.title('Year-wise Movie Count in IMDb TOP 1000 Movies')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```



```
# Find the year with the most movies released
year_counts = imdb_data['Released_Year'].value_counts()
year_with_most_movies = year_counts.idxmax()
print("\nYear with the most movies released:", year_with_most_movies)
```

```
Year with the most movies released: 2014
```

7. DIRECTORS WITH LONGEST MOVIES

This analysis calculates the average runtime of movies for each director and identifies the top directors with the longest average movie runtimes.

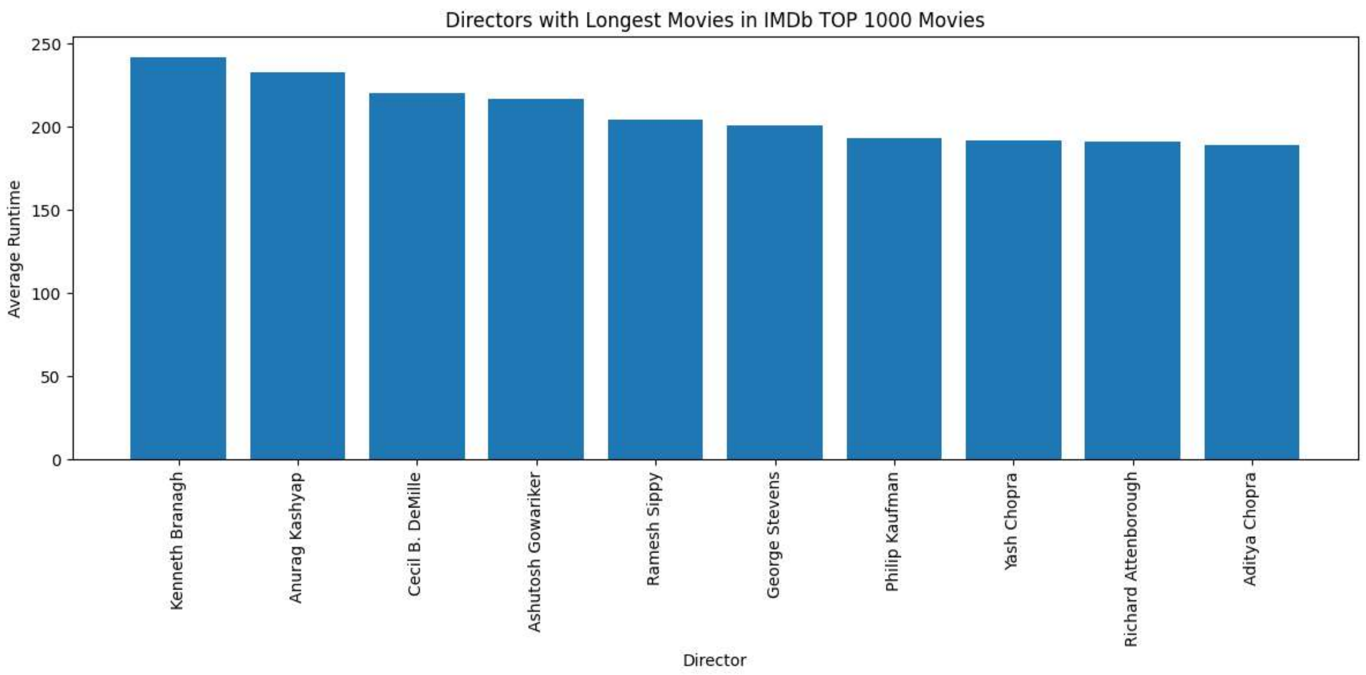
The code below uses the 'groupby()' function on the 'Director' column of the IMDb DataFrame ('imdb_df') to group the data by director. Then, it calculates the mean of the 'Runtime (Minutes)' column for each director using mean(). Finally, it retrieves the top 10 directors with the longest average runtime using 'nlargest()' and stores the result in the 'director_avg_runtime' variable.

```
# Clean the "Runtime (Minutes)" column and convert it to numeric values
imdb_df['Runtime'] = imdb_df['Runtime'].str.replace(' min', '').astype(float)
```

```
# Directors with Longest Movies
director_avg_runtime = imdb_df.groupby('Director')['Runtime'].mean().nlargest(10)
```

DIRECTORS WITH LONGEST MOVIES PLOT

```
# Plot Directors with Longest Movies
plt.figure(figsize=(12, 6))
plt.bar(director_avg_runtime.index, director_avg_runtime.values)
plt.xlabel('Director')
plt.ylabel('Average Runtime')
plt.title('Directors with Longest Movies in IMDb TOP 1000 Movies')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```



RESULTS:

The results of the data analysis provide valuable insights into the IMDb TOP 1000 movie dataset, such as the distribution of genres, average rating, highest-rated and longest movies, directors with the most movies, and year-wise movie counts. These insights can be used to understand user preferences, identify popular movies and directors, and observe trends in the film industry.

FUTURE WORKS:

For further analysis, More in-depth analyses and visualizations can be conducted to explore specific aspects of the dataset further. Additionally, incorporating other datasets or external data sources can enrich the analysis and provide a more comprehensive understanding of factors influencing movie ratings, genres, and directors' contributions. Moreover, machine learning techniques can be applied to predict movie ratings based on various features or to recommend movies to users based on their preferences.

THANK YOU..!!!

✓ 0s completed at 10:50 AM

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.