**Project Description:**

The project involves analyzing a dataset of movies to extract insights regarding popular genres, best directors, and favorite actors among critics and audiences. The dataset contains information about movies such as their titles, directors, actors, genres, ratings, and other details. Through this project, we aim to gain a deeper understanding of the movie industry and provide insights that can be useful for various stakeholders such as producers, directors, and actors.

**Approach:**

The project involved importing the dataset into Jupyter Notebook and using Python libraries such as Pandas and Matplotlib for data cleaning, analysis, and visualization. We started by exploring the dataset and identifying the relevant columns for our analysis. Then, we cleaned the data by removing duplicates, missing values, and irrelevant columns. After that, we performed various analysis and visualizations to extract insights.

**Tech-Stack Used:**

For this project, we used Jupyter Notebook as the coding environment and Python libraries such as Pandas, Matplotlib, and Seaborn for data analysis and visualization. We also used Microsoft Excel for exporting and formatting the data..

**Insights:**

Through our analysis, we found that Drama, Comedy, Action, Thriller, and Adventure are the most popular genres among audiences and critics. We also identified the top 10 directors with the highest mean IMDb score and found that Christopher Nolan, Quentin Tarantino, and Stanley Kubrick are among them. In addition, we identified the favorite actors among critics and audiences, with Leonardo DiCaprio, Meryl Streep, and Brad Pitt being the most popular. We also observed the change in the number of voted users over decades and found that the number of votes has significantly increased in recent years.

**Result:**

The project helped us gain insights into the movie industry and provided useful information for stakeholders such as producers, directors, and actors. Through our analysis, we identified popular genres, best directors, and favorite actors, which can be useful for making informed decisions about movie production and casting. The project also helped us improve our data analysis and visualization skills using Python libraries.

**1)Cleaning the data::** PThis is one of the most important step to perform before moving forward with the analysis. Use your knowledge learned till now to do this. (Dropping columns, removing null values, etc.)

**Your task: Clean the data**

import pandas as pd

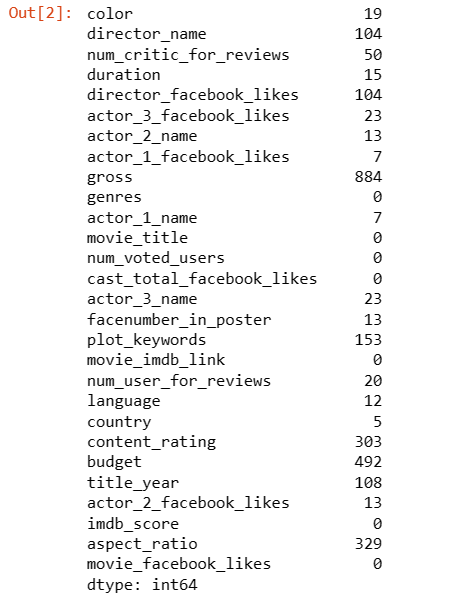
# load the data set

df = pd.read\_csv(r'C:\Users\karan\Downloads\IMDB\_Movies.csv')

---------------------------------------------------------------------------------

# check for missing or null values

df.isnull().sum()

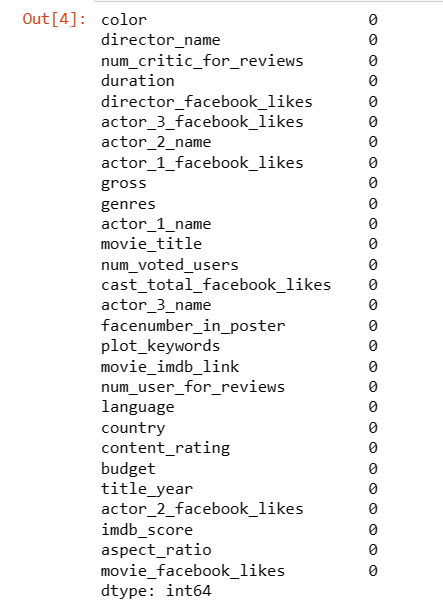


# drop rows with missing values

df.dropna(inplace=True)

# check for missing or null values

df.isnull().sum()



**2)Movies with highest profit:** Create a new column called profit which contains the difference of the two columns: gross and budget. Sort the column using the profit column as reference. Plot profit (y-axis) vs budget (x- axis) and observe the outliers using the appropriate chart type.

**Your task: Find the movies with the highest profit?**

import matplotlib.pyplot as plt

# create a new column for profit

df['profit'] = df['gross'] - df['budget']

# sort the data set by profit

df.sort\_values(by='profit', ascending=False, inplace=True)

# plot profit vs budget

plt.scatter(df['budget'], df['profit'])

plt.xlabel('Budget')

plt.ylabel('Profit')

plt.show()import matplotlib.pyplot as plt

# create a new column for profit

df['profit'] = df['gross'] - df['budget']

# sort the data set by profit

df.sort\_values(by='profit', ascending=False, inplace=True)

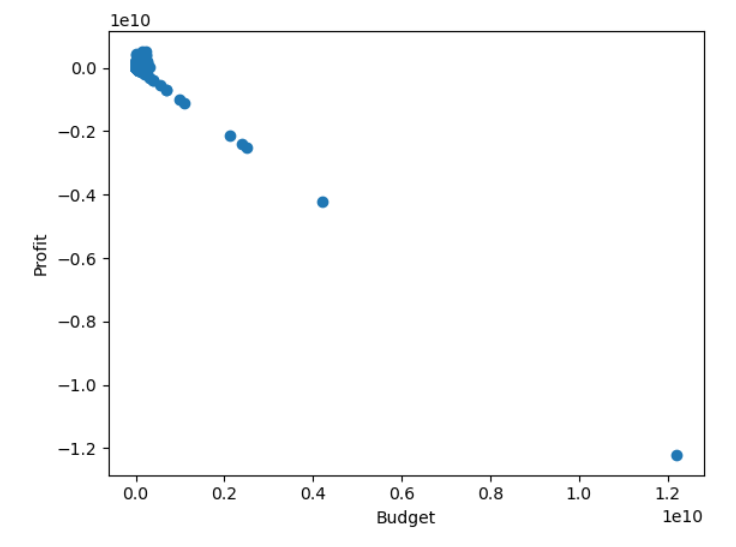
# plot profit vs budget

plt.scatter(df['budget'], df['profit'])

plt.xlabel('Budget')

plt.ylabel('Profit')

plt.show()



import pandas as pd

# Load the dataset

movies = pd.read\_csv(r'C:\Users\karan\Downloads\IMDB\_Movies.csv')

# Create a new column 'profit'

movies['profit'] = movies['gross'] - movies['budget']

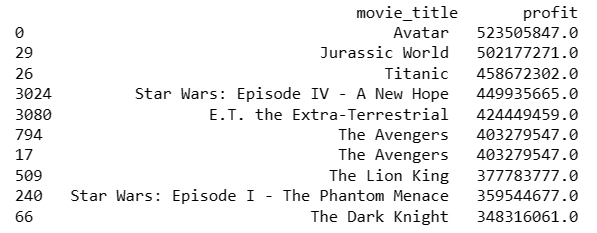
# Sort the dataset by profit in descending order

sorted\_movies = movies.sort\_values(by='profit', ascending=False)

# Display the top 10 movies with the highest profit

top\_profit\_movies = sorted\_movies[['movie\_title', 'profit']].head(10)

print(top\_profit\_movies)



**3)Top 250:** Create a new column IMDb\_Top\_250 and store the top 250 movies with the highest IMDb Rating (corresponding to the column: imdb\_score). Also make sure that for all of these movies, the num\_voted\_users is greater than 25,000. Also add a Rank column containing the values 1 to 250 indicating the ranks of the corresponding films.

Extract all the movies in the IMDb\_Top\_250 column which are not in the English language and store them in a new column named Top\_Foreign\_Lang\_Film. You can use your own imagination also!

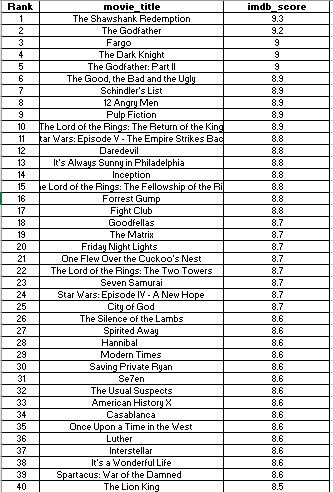
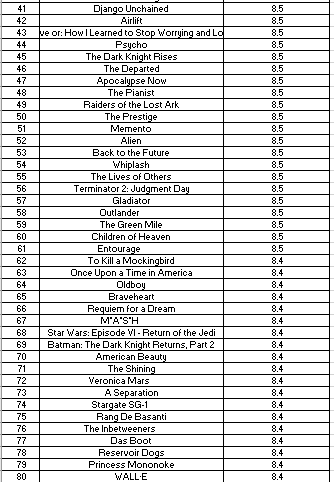
**Your task: Find IMDB Top 250**

top250\_movies = movies[movies['num\_voted\_users']>25000].sort\_values(by='imdb\_score',ascending=False)[:250]

top250\_movies['IMDb\_Top\_250'] = 1

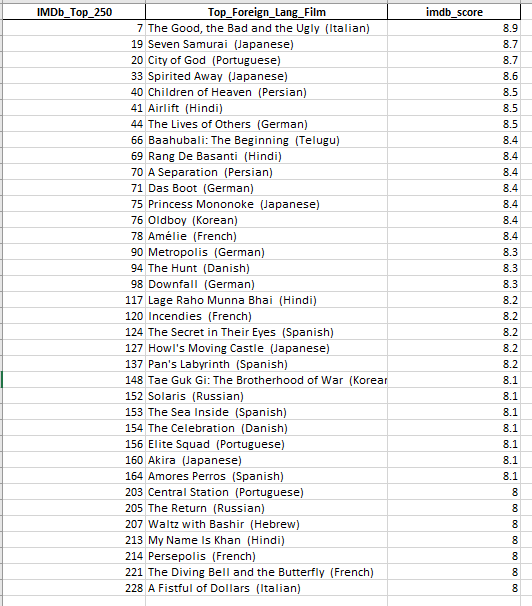
top250\_movies['Rank'] = range(1, 251)

top250\_movies.to\_excel('top250\_movies.xlsx', index=False)



****

To extract all the movies in the IMDb\_Top\_250 column which are not in the English language and store them in a new column named Top\_Foreign\_Lang\_Film:

****

**# First, let's filter out the movies that are not in the English language**

**top\_foreign\_lang\_movies = top250\_movies[top250\_movies['language'] != 'English'].copy()**

**# Next, let's create the new column Top\_Foreign\_Lang\_Film**

**top\_foreign\_lang\_movies.loc[:, 'Top\_Foreign\_Lang\_Film'] = top\_foreign\_lang\_movies['movie\_title'] + ' (' + top\_foreign\_lang\_movies['language'] + ')'**

# Finally, let's save the filtered data to a new excel file

top\_foreign\_lang\_movies.to\_excel('IMDb\_Top\_Foreign\_Lang\_Films.xlsx', index=False)

****

**4)Best Directors:** TGroup the column using the director\_name column.

Find out the top 10 directors for whom the mean of imdb\_score is the highest and store them in a new column top10director. In case of a tie in IMDb score between two directors, sort them alphabetically.

**Your task: Find the best directors.**

# Reset the index to start from 1 instead of 0

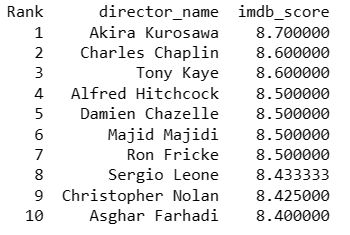
top10directors = top10directors.reset\_index(drop=True)

# Add a new column 'Rank' to show the rank of each director

top10directors['Rank'] = range(1, 11)

# Print the top 10 directors with rank

print(top10directors[['Rank', 'director\_name', 'imdb\_score']])



**5)Popular Genres:** Perform this step using the knowledge gained while performing previous steps.

**Your task: Find popular genres**

# Step 1: Create a new column 'genre\_list' by splitting the 'genres' column on '|' separator

df['genre\_list'] = df['genres'].str.split('|')

# Step 2: Explode the 'genre\_list' column to create a new row for each genre

df\_exploded = df.explode('genre\_list')

# Step 3: Group the data by genre and calculate the mean imdb\_score

genre\_scores = df\_exploded.groupby('genre\_list')['imdb\_score'].mean().reset\_index()

# Step 4: Sort the data by mean imdb\_score in descending order and add a Rank column

genre\_scores = genre\_scores.sort\_values(by='imdb\_score', ascending=False)

genre\_scores['Rank'] = range(1, len(genre\_scores)+1)

# Step 5: Select the top 10 genres with highest mean imdb\_score and print them with their rank

top10genres = genre\_scores.head(10)

print(top10genres[['Rank', 'genre\_list', 'imdb\_score']])

****

**6)Charts:** Create three new columns namely, Meryl\_Streep, Leo\_Caprio, and Brad\_Pitt which contain the movies in which the actors: 'Meryl Streep', 'Leonardo DiCaprio', and 'Brad Pitt' are the lead actors. Use only the actor\_1\_name column for extraction. Also, make sure that you use the names 'Meryl Streep', 'Leonardo DiCaprio', and 'Brad Pitt' for the said extraction.

Append the rows of all these columns and store them in a new column named Combined.

Group the combined column using the actor\_1\_name column.

Find the mean of the num\_critic\_for\_reviews and num\_users\_for\_review and identify the actors which have the highest mean.

Observe the change in number of voted users over decades using a bar chart. Create a column called decade which represents the decade to which every movie belongs to. For example, the title\_year year 1923, 1925 should be stored as 1920s. Sort the column based on the column decade, group it by decade and find the sum of users voted in each decade. Store this in a new data frame called df\_by\_decade.

**Your task: Find the critic-favorite and audience-favorite actors**

Create three new columns namely, Meryl\_Streep, Leo\_Caprio, and Brad\_Pitt which contain the movies in which the actors: 'Meryl Streep', 'Leonardo DiCaprio', and 'Brad Pitt' are the lead actors. Use only the actor\_1\_name column for extraction. Also, make sure that you use the names 'Meryl Streep', 'Leonardo DiCaprio', and 'Brad Pitt' for the said extraction.

Append the rows of all these columns and store them in a new column named Combined.

Group the combined column using the actor\_1\_name column.

**# Create new columns for each actor**

meryl\_streep\_movies = movies[movies['actor\_1\_name'] == 'Meryl Streep'].copy()

meryl\_streep\_movies.loc[:, 'Meryl\_Streep'] = meryl\_streep\_movies['movie\_title']

leo\_movies = movies[movies['actor\_1\_name'] == 'Leonardo DiCaprio'].copy()

leo\_movies.loc[:, 'Leo\_Caprio'] = leo\_movies['movie\_title']

brad\_movies = movies[movies['actor\_1\_name'] == 'Brad Pitt'].copy()

brad\_movies.loc[:, 'Brad\_Pitt'] = brad\_movies['movie\_title']

# Concatenate the dataframes and create the Combined column

combined\_movies = pd.concat([meryl\_streep\_movies, leo\_movies, brad\_movies], sort=True)

combined\_movies['Combined'] = combined\_movies['Meryl\_Streep'].fillna('') + combined\_movies['Leo\_Caprio'].fillna('') + combined\_movies['Brad\_Pitt'].fillna('')

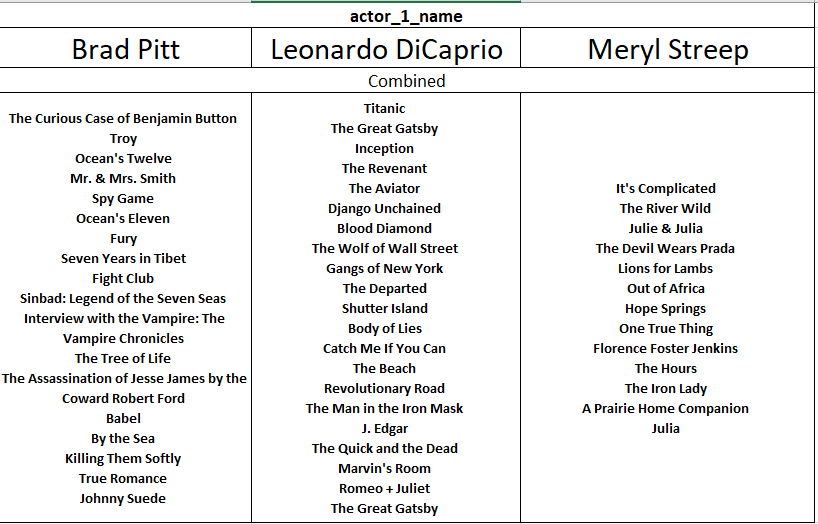
# Group by actor name and show the Combined column

grouped = combined\_movies.groupby('actor\_1\_name')['Combined'].apply(lambda x: '\n'.join(x)).reset\_index()

# Save to Excel

grouped.to\_excel('grouped\_data2.xlsx', index=False)

**----------------------------------------------------------------------------------------------**

****

**Find the mean of the num\_critic\_for\_reviews and num\_users\_for\_review and identify the**

**actors which have the highest mean.**

# Group movies by actor and calculate the mean of the num\_critic\_for\_reviews and num\_users\_for\_review columns

actor\_review\_means = movies.groupby('actor\_1\_name')[['num\_critic\_for\_reviews', 'num\_user\_for\_reviews']].mean()

# Add a column for the total review mean

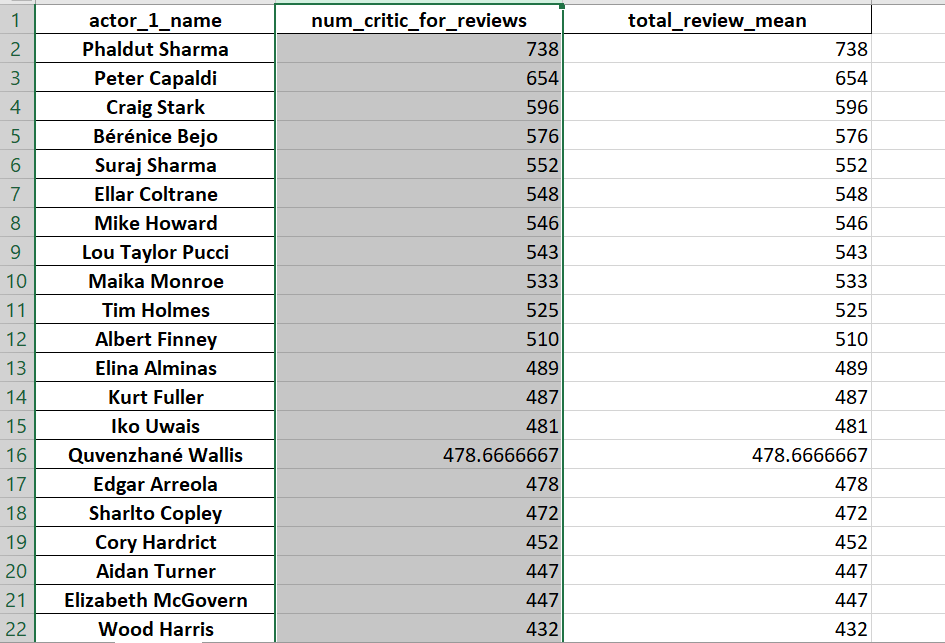
actor\_review\_means['total\_review\_mean'] = actor\_review\_means.mean(axis=1)

# Sort by the total\_review\_mean in descending order

actor\_review\_means = actor\_review\_means.sort\_values(by='total\_review\_mean', ascending=False)

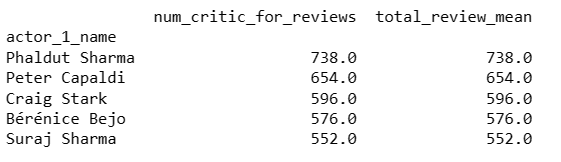
# Export as excel

actor\_review\_means.to\_excel('actor\_review\_means.xlsx')



top\_five\_actors = actor\_review\_means.head()

print(top\_five\_actors)



**top five actors with the highest mean**

# Group movies by actor and calculate the mean of the num\_user\_for\_reviews column

actor\_user\_review\_means = movies.groupby('actor\_1\_name')['num\_user\_for\_reviews'].mean()

# Convert the mean values to strings before concatenating with actor names

top\_five\_actors = actor\_user\_review\_means.apply(lambda x: str(round(x, 2))).sort\_values(ascending=False).head(5)

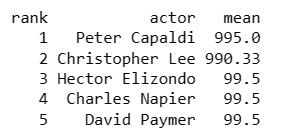
# Create a DataFrame with the top five actors and their mean values

df = pd.DataFrame({'rank': range(1, 6), 'actor': top\_five\_actors.index, 'mean': top\_five\_actors.values})

movies['num\_user\_for\_reviews'] = pd.to\_numeric(movies['num\_user\_for\_reviews'], errors='coerce').fillna(0)

# Print the DataFrame

print(df.to\_string(index=False))



**Observe the change in number of voted users over decades using a bar chart.**

import matplotlib.pyplot as plt

# Extract the decade from the release year

movies['decade'] = (movies['title\_year'] // 10) \* 10

# Group movies by decade and sum the number of voted users

voted\_users\_by\_decade = movies.groupby('decade')['num\_voted\_users'].sum()

# Create a bar chart

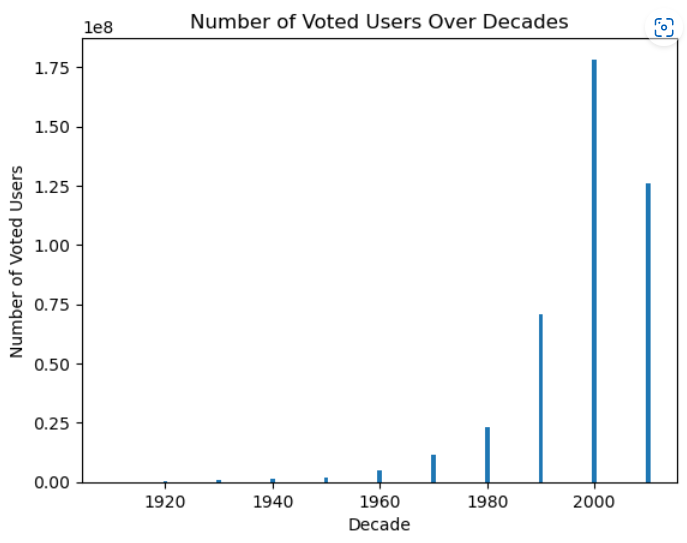
plt.bar(voted\_users\_by\_decade.index, voted\_users\_by\_decade.values)

plt.xlabel('Decade')

plt.ylabel('Number of Voted Users')

plt.title('Number of Voted Users Over Decades')

plt.show()



**Create a column called decade which represents the decade to which every movie belongs to. For example, the title\_year year 1923, 1925 should be stored as 1920s. Sort the column based on the column decade, group it by decade and find the sum of users voted in each decade. Store this in a new data frame called df\_by\_decade.**

# Create a column 'decade' representing the decade to which each movie belongs

movies['decade'] = (movies['title\_year'] // 10 \* 10).fillna(-1).astype(int).astype(str).replace('-1', 'Unknown') + 's'

# Sort the column 'decade' and group the movies by decade, finding the sum of users voted in each decade

df\_by\_decade = movies.sort\_values('decade').groupby('decade')['num\_voted\_users'].sum().reset\_index()

# Print the new data frame

print(df\_by\_decade)

