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()
```

```
Out[2]: color                        19
        director_name               104
        num_critic_for_reviews       50
        duration                     15
        director_facebook_likes     104
        actor_3_facebook_likes       23
        actor_2_name                 13
        actor_1_facebook_likes        7
        gross                       884
        genres                        0
        actor_1_name                  7
        movie_title                   0
        num_voted_users               0
        cast_total_facebook_likes     0
        actor_3_name                 23
        facenumber_in_poster         13
        plot_keywords               153
        movie_imdb_link               0
        num_user_for_reviews         20
        language                     12
        country                       5
        content_rating              303
        budget                      492
        title_year                  108
        actor_2_facebook_likes       13
        imdb_score                    0
        aspect_ratio                329
        movie_facebook_likes          0
        dtype: int64
```

drop rows with missing values

df.dropna(inplace=True)

check for missing or null values

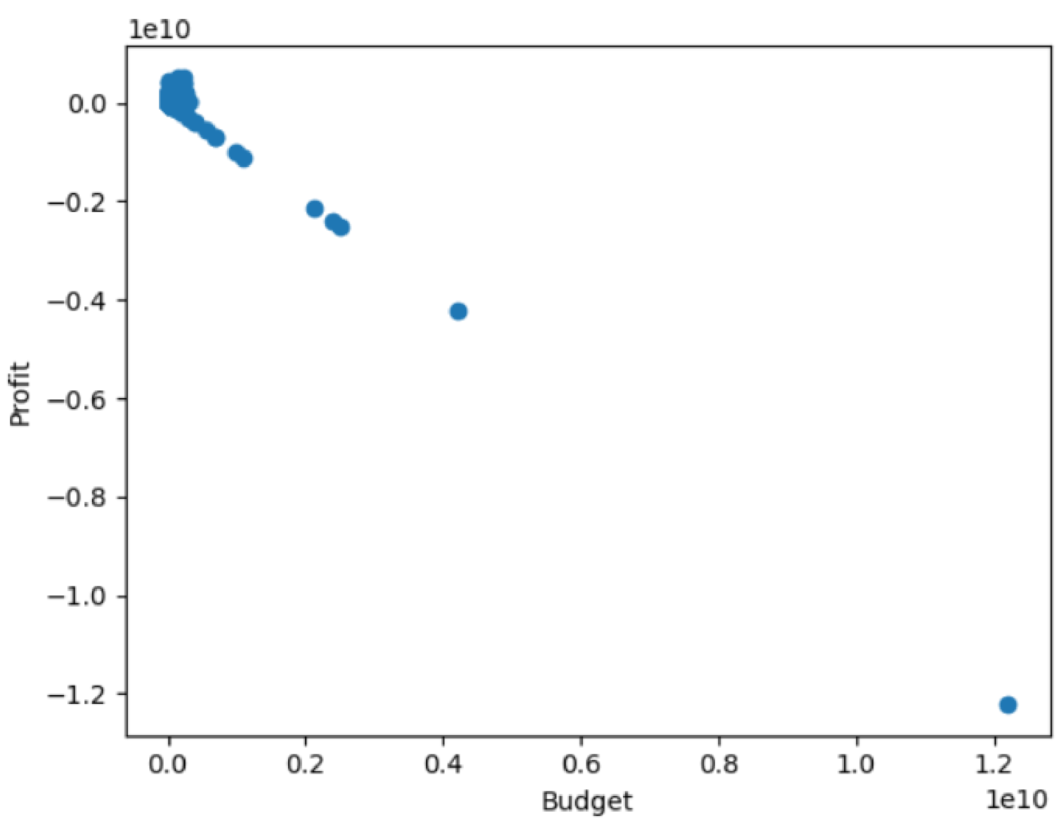
df.isnull().sum()

```
Out[4]: color                        0
        director_name                0
        num_critic_for_reviews       0
        duration                     0
        director_facebook_likes      0
        actor_3_facebook_likes       0
        actor_2_name                 0
        actor_1_facebook_likes       0
        gross                        0
        genres                       0
        actor_1_name                 0
        movie_title                  0
        num_voted_users              0
        cast_total_facebook_likes    0
        actor_3_name                 0
        facenumber_in_poster         0
        plot_keywords                0
        movie_imdb_link              0
        num_user_for_reviews         0
        language                     0
        country                      0
        content_rating               0
        budget                       0
        title_year                   0
        actor_2_facebook_likes       0
        imdb_score                   0
        aspect_ratio                 0
        movie_facebook_likes         0
        dtype: int64
```

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)
```

```
                                  movie_title       profit
0                                     Avatar   523505847.0
29                            Jurassic World   502177271.0
26                                   Titanic   458672302.0
3024        Star Wars: Episode IV - A New Hope   449935665.0
3080              E.T. the Extra-Terrestrial   424449459.0
794                             The Avengers   403279547.0
17                              The Avengers   403279547.0
509                            The Lion King   377783777.0
240   Star Wars: Episode I - The Phantom Menace   359544677.0
66                           The Dark Knight   348316061.0
```

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)[:2
50]
top250_movies['IMDb_Top_250'] = 1
top250_movies['Rank'] = range(1, 251)
top250_movies.to_excel('top250_movies.xlsx', index=False)
```

Rank	movie_title	imdb_score
1	The Shawshank Redemption	9.3
2	The Godfather	9.2
3	Fargo	9
4	The Dark Knight	9
5	The Godfather: Part II	9
6	The Good, the Bad and the Ugly	8.9
7	Schindler's List	8.9
8	12 Angry Men	8.9
9	Pulp Fiction	8.9
10	The Lord of the Rings: The Return of the King	8.9
11	tar Wars: Episode V - The Empire Strikes Bac	8.8
12	Daredevil	8.8
13	It's Always Sunny in Philadelphia	8.8
14	Inception	8.8
15	e Lord of the Rings: The Fellowship of the Ri	8.8
16	Forrest Gump	8.8
17	Fight Club	8.8
18	Goodfellas	8.7
19	The Matrix	8.7
20	Friday Night Lights	8.7
21	One Flew Over the Cuckoo's Nest	8.7
22	The Lord of the Rings: The Two Towers	8.7
23	Seven Samurai	8.7
24	Star Wars: Episode IV - A New Hope	8.7
25	City of God	8.7
26	The Silence of the Lambs	8.6
27	Spirited Away	8.6
28	Hannibal	8.6
29	Modern Times	8.6
30	Saving Private Ryan	8.6
31	Se7en	8.6
32	The Usual Suspects	8.6
33	American History X	8.6
34	Casablanca	8.6
35	Once Upon a Time in the West	8.6
36	Luther	8.6
37	Interstellar	8.6
38	It's a Wonderful Life	8.6
39	Spartacus: War of the Damned	8.6
40	The Lion King	8.5
41	Django Unchained	8.5
42	Airlift	8.5
43	ve or: How I Learned to Stop Worrying and Lo	8.5
44	Psycho	8.5
45	The Dark Knight Rises	8.5
46	The Departed	8.5
47	Apocalypse Now	8.5
48	The Pianist	8.5
49	Raiders of the Lost Ark	8.5
50	The Prestige	8.5
51	Memento	8.5
52	Alien	8.5
53	Back to the Future	8.5
54	Whiplash	8.5
55	The Lives of Others	8.5
56	Terminator 2: Judgment Day	8.5
57	Gladiator	8.5
58	Outlander	8.5
59	The Green Mile	8.5
60	Children of Heaven	8.5
61	Entourage	8.5
62	To Kill a Mockingbird	8.4
63	Once Upon a Time in America	8.4
64	Oldboy	8.4
65	Braveheart	8.4
66	Requiem for a Dream	8.4
67	M*A*S*H	8.4
68	Star Wars: Episode VI - Return of the Jedi	8.4
69	Batman: The Dark Knight Returns, Part 2	8.4
70	American Beauty	8.4
71	The Shining	8.4
72	Veronica Mars	8.4
73	A Separation	8.4
74	Stargate SG-1	8.4
75	Rang De Basanti	8.4
76	The Inbetweeners	8.4
77	Das Boot	8.4
78	Reservoir Dogs	8.4
79	Princess Mononoke	8.4
80	WALL·E	8.4
80	WALL·E	8.4
81	Psych	8.4
82	Baahubali: The Beginning	8.4
83	Amélie	8.4
84	Aliens	8.4
85	Lawrence of Arabia	8.4
86	Room	8.3
87	Downfall	8.3
88	The Sting	8.3
89	Life	8.3
90	Snatch	8.3
91	Scarface	8.3
92	Some Like It Hot	8.3
93	Good Will Hunting	8.3
94	Batman Begins	8.3
95	Singin' in the Rain	8.3
96	2001: A Space Odyssey	8.3
97	Monty Python and the Holy Grail	8.3
98	Raging Bull	8.3
99	The Hunt	8.3
100	Metropolis	8.3
101	The Apartment	8.3
102	Amadeus	8.3
103	Inside Job	8.3
104	Judgment at Nuremberg	8.3
105	Inside Out	8.3
106	Eternal Sunshine of the Spotless Mind	8.3
107	Up	8.3
108	The Great Escape	8.3
109	Indiana Jones and the Last Crusade	8.3
110	Toy Story	8.3
111	Toy Story 3	8.3
112	Inglourious Basterds	8.3
113	L.A. Confidential	8.3
114	Taxi Driver	8.3
115	Snatch	8.3
116	Unforgiven	8.3
117	The Bridge on the River Kwai	8.2
118	Pan's Labyrinth	8.2
119	Incendies	8.2
120	The Deer Hunter	8.2
121	A Beautiful Mind	8.2
122	Buffy the Vampire Slayer	8.2
123	Trainspotting	8.2
124	Warrior	8.2
125	Into the Wild	8.2
126	Gone with the Wind	8.2
127	The Elephant Man	8.2
128	Howl's Moving Castle	8.2
129	The Big Lebowski	8.2
130	The Secret in Their Eyes	8.2
131	Gran Torino	8.2
132	Blade Runner	8.2
133	The Wolf of Wall Street	8.2
134	It Happened One Night	8.2
135	Captain America: Civil War	8.2
136	Casino	8.2
137	On the Waterfront	8.2
138	V for Vendetta	8.2
139	Finding Nemo	8.2
140	Rebecca	8.2
141	Lock, Stock and Two Smoking Barrels	8.2
142	Mr. Smith Goes to Washington	8.2
143	The Thing	8.2
144	How to Train Your Dragon	8.2
145	Die Hard	8.2
146	Lage Raho Munna Bhai	8.2
147	No Country for Old Men	8.1
148	es of the Caribbean: The Curse of the Black F	8.1
149	High Noon	8.1
150	Kill Bill: Vol. 1	8.1
151	Jurassic Park	8.1
152	The Wizard of Oz	8.1
153	Shutter Island	8.1
154	The Martian	8.1
155	Rocky	8.1
156	Sin City	8.1
157	Solaris	8.1
158	The Help	8.1
159	Prisoners	8.1
160	The Man Who Shot Liberty Valance	8.1
161	The Celebration	8.1
162	The Sixth Sense	8.1
163	The Avengers	8.1
164	Hotel Rwanda	8.1
165	Deadpool	8.1
166	Mad Max: Fury Road	8.1
167	The Princess Bride	8.1
168	Cat on a Hot Tin Roof	8.1
169	The Avengers	8.1
170	Butch Cassidy and the Sundance Kid	8.1
171	Gone Girl	8.1
172	Platoon	8.1
173	Barry Lyndon	8.1
174	The Imitation Game	8.1
175	The Terminator	8.1
176	Hachi: A Dog's Tale	8.1
177	Monsters, Inc.	8.1
178	Elite Squad	8.1
179	The Truman Show	8.1

180	Stand by Me	8.1
181	The Sea Inside	8.1
182	Annie Hall	8.1
183	Touching the Void	8.1
184	Before Sunrise	8.1
185	There Will Be Blood	8.1
186	The Best Years of Our Lives	8.1
187	Million Dollar Baby	8.1
188	Guardians of the Galaxy	8.1
189	Network	8.1
190	The Grand Budapest Hotel	8.1
191	Akira	8.1
192	Tae Guk Gi: The Brotherhood of War	8.1
193	Groundhog Day	8.1
194	Gandhi	8.1
195	Amores Perros	8.1
196	Rush	8.1
197	The Revenant	8.1
198	Spotlight	8.1
199	A Christmas Story	8.1
200	Donnie Darko	8.1
201	The Bourne Ultimatum	8.1
202	12 Years a Slave	8.1
203	The Incredibles	8
204	Fiddler on the Roof	8
205	The Perks of Being a Wallflower	8
206	The Diving Bell and the Butterfly	8
207	Black Swan	8
208	True Romance	8
209	Slumdog Millionaire	8
210	Casino Royale	8
211	Sicko	8
212	The King's Speech	8
213	Blood Diamond	8
214	Cinderella Man	8
215	Patton	8
216	Mulholland Drive	8
217	In Bruges	8
218	Dancer in the Dark	8
219	Brazil	8
220	The Straight Story	8
221	My Name Is Khan	8
222	Jaws	8
223	Casino Royale	8
224	The Return	8
225	The Exorcist	8
226	The Sound of Music	8
227	Persepolis	8
228	Doctor Zhivago	8
229	Star Trek	8
230	A Streetcar Named Desire	8
231	The Iron Giant	8
232	X-Men: Days of Future Past	8
233	Rosemary's Baby	8

233	Rosemary's Baby	8
234	Bowling for Columbine	8
235	Rain Man	8
236	Days of Heaven	8
237	JFK	8
238	Central Station	8
239	Ratatouille	8
240	Catch Me If You Can	8
241	Young Frankenstein	8
242	Serenity	8
243	Waltz with Bashir	8
244	Aladdin	8
245	Magnolia	8
246	Before Sunset	8
247	Big Fish	8
248	Mystic River	8
249	The Hustler	8
250	District 9	8

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'] + ')'
```

IMDb_Top_250	Top_Foreign_Lang_Film	imdb_score
7	The Good, the Bad and the Ugly (Italian)	8.9
19	Seven Samurai (Japanese)	8.7
20	City of God (Portuguese)	8.7
33	Spirited Away (Japanese)	8.6
40	Children of Heaven (Persian)	8.5
41	Airlift (Hindi)	8.5
44	The Lives of Others (German)	8.5
66	Baahubali: The Beginning (Telugu)	8.4
69	Rang De Basanti (Hindi)	8.4
70	A Separation (Persian)	8.4
71	Das Boot (German)	8.4
75	Princess Mononoke (Japanese)	8.4
76	Oldboy (Korean)	8.4
78	Amélie (French)	8.4
90	Metropolis (German)	8.3
94	The Hunt (Danish)	8.3
98	Downfall (German)	8.3
117	Lage Raho Munna Bhai (Hindi)	8.2
120	Incendies (French)	8.2
124	The Secret in Their Eyes (Spanish)	8.2
127	Howl's Moving Castle (Japanese)	8.2
137	Pan's Labyrinth (Spanish)	8.2
148	Tae Guk Gi: The Brotherhood of War (Korear	8.1
152	Solaris (Russian)	8.1
153	The Sea Inside (Spanish)	8.1
154	The Celebration (Danish)	8.1
156	Elite Squad (Portuguese)	8.1
160	Akira (Japanese)	8.1
164	Amores Perros (Spanish)	8.1
203	Central Station (Portuguese)	8
205	The Return (Russian)	8
207	Waltz with Bashir (Hebrew)	8
213	My Name Is Khan (Hindi)	8
214	Persepolis (French)	8
221	The Diving Bell and the Butterfly (French)	8
228	A Fistful of Dollars (Italian)	8

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']])
```

```
 Rank      director_name  imdb_score
    1     Akira Kurosawa    8.700000
    2    Charles Chaplin    8.600000
    3         Tony Kaye     8.600000
    4   Alfred Hitchcock    8.500000
    5    Damien Chazelle    8.500000
    6       Majid Majidi    8.500000
    7         Ron Fricke    8.500000
    8      Sergio Leone     8.433333
    9 Christopher Nolan     8.425000
   10     Asghar Farhadi    8.400000
```

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']])

```
Rank    genre_list  imdb_score
   1     Film-Noir    7.700000
   2       History    7.174510
   3     Biography    7.157741
   4           War    7.067532
   5   Documentary    6.988889
   6       Western    6.793220
   7         Drama    6.791513
   8     Animation    6.702551
   9       Musical    6.596875
  10         Sport    6.593243
```

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)
```

---

actor_1_name		
Brad Pitt	Leonardo DiCaprio	Meryl Streep
Combined		
The Curious Case of Benjamin Button Troy Ocean's Twelve Mr. & Mrs. Smith Spy Game Ocean's Eleven Fury Seven Years in Tibet Fight Club Sinbad: Legend of the Seven Seas Interview with the Vampire: The Vampire Chronicles The Tree of Life The Assassination of Jesse James by the Coward Robert Ford Babel By the Sea Killing Them Softly True Romance Johnny Suede	Titanic The Great Gatsby Inception The Revenant The Aviator Django Unchained Blood Diamond The Wolf of Wall Street Gangs of New York The Departed Shutter Island Body of Lies Catch Me If You Can The Beach Revolutionary Road The Man in the Iron Mask J. Edgar The Quick and the Dead Marvin's Room Romeo + Juliet The Great Gatsby	It's Complicated The River Wild Julie & Julia The Devil Wears Prada Lions for Lambs Out of Africa Hope Springs One True Thing Florence Foster Jenkins The Hours The Iron Lady A Prairie Home Companion Julia

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')
```

1	actor_1_name	num_critic_for_reviews	total_review_mean
2	Phaldut Sharma	738	738
3	Peter Capaldi	654	654
4	Craig Stark	596	596
5	Bérénice Bejo	576	576
6	Suraj Sharma	552	552
7	Ellar Coltrane	548	548
8	Mike Howard	546	546
9	Lou Taylor Pucci	543	543
10	Maika Monroe	533	533
11	Tim Holmes	525	525
12	Albert Finney	510	510
13	Elina Alminas	489	489
14	Kurt Fuller	487	487
15	Iko Uwais	481	481
16	Quvenzhané Wallis	478.6666667	478.6666667
17	Edgar Arreola	478	478
18	Sharlto Copley	472	472
19	Cory Hardrict	452	452
20	Aidan Turner	447	447
21	Elizabeth McGovern	447	447
22	Wood Harris	432	432

```
top_five_actors = actor_review_means.head()
print(top_five_actors)
```

```
                    num_critic_for_reviews  total_review_mean
actor_1_name
Phaldut Sharma                       738.0              738.0
Peter Capaldi                        654.0              654.0
Craig Stark                          596.0              596.0
Bérénice Bejo                        576.0              576.0
Suraj Sharma                         552.0              552.0
```

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

```
rank             actor    mean
   1     Peter Capaldi   995.0
   2 Christopher Lee  990.33
   3 Hector Elizondo    99.5
   4   Charles Napier    99.5
   5     David Paymer    99.5
```

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

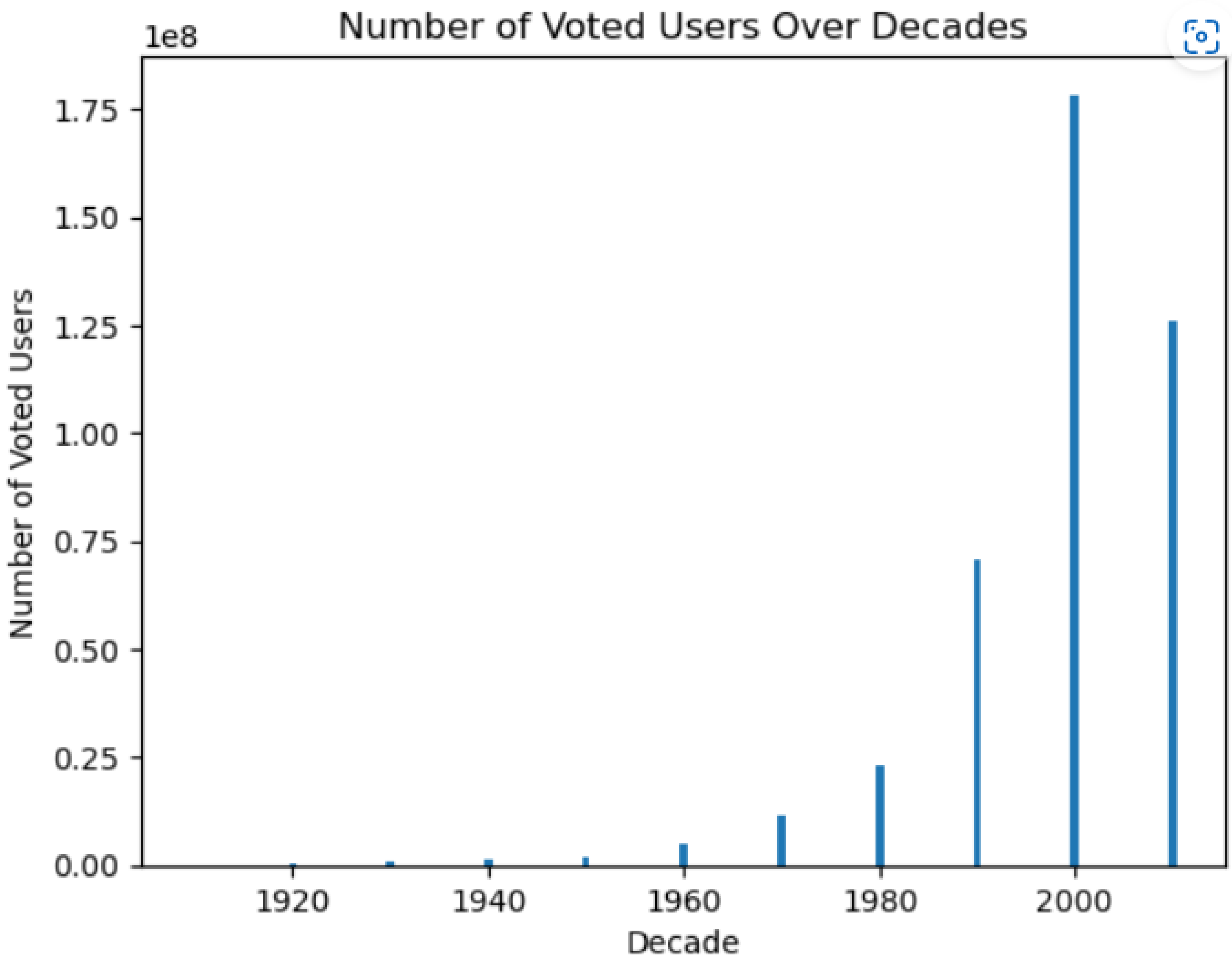
import matplotlib.pyplot as plt

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)
```

```
      decade  num_voted_users
0      1910s            10718
1      1920s           128672
2      1930s           984397
3      1940s          1211888
4      1950s          1638504
5      1960s          5153052
6      1970s         11312705
7      1980s         23176169
8      1990s         70633270
9      2000s        178354686
10     2010s        126202706
11  Unknowns          3131768
```