Exploratory Data Analysis on IMDb dataset

Description:

The primary objective of this project is to conduct in-depth exploratory data analysis on the IMDb dataset to extract meaningful insights about movies and TV shows. Through data visualization and statistical analysis, we aim to answer various questions and detect patterns within datasets.

Importing the important packages

In [ ]:
```
import numpy as np
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
%matplotlib inline
```

Loading the dataset

In [ ]:
```
data = pd.read_csv('/content/IMDB_Movies.csv')
# getting all the data column
pd.set_option('display.max_columns', None)

data.head()
```

Out[ ]:

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_lik
0	Color	James Cameron	723.0	178.0	0.0	85
1	Color	Gore Verbinski	302.0	169.0	563.0	100
2	Color	Sam Mendes	602.0	148.0	0.0	16
3	Color	Christopher Nolan	813.0	164.0	22000.0	2300
4	NaN	Doug Walker	NaN	NaN	131.0	Na

A. Cleaning the data: This is one of the most important steps before proceeding with the analysis. To do this use the knowledge you have learned so far. (deleting columns, removing null values, etc.)

Task: Clean the data

In [ ]:
```
# getting the duplicated value and sorting it in descending order
data.isnull().sum().sort_values(ascending = False)
```

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

In [ ]:
```
data.shape
```

Out[ ]:
```
(5043, 28)
```

Extracting only those columns that are important

In [ ]:
```
df = data[['director_name', 'num_critic_for_reviews', 'gross', 'genres', 'actor_1_name',
           'num_voted_users', 'num_user_for_reviews', 'language', 'budget', 'title_year',
           'movie_facebook_likes']]
df.head()
```

Out[ ]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	mov
0	James Cameron	723.0	760505847.0	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	
1	Gore Verbinski	302.0	309404152.0	Action|Adventure|Fantasy	Johnny Depp	Pi Car At
2	Sam Mendes	602.0	200074175.0	Action|Adventure|Thriller	Christoph Waltz	
3	Christopher Nolan	813.0	448130642.0	Action|Thriller	Tom Hardy	T
4	Doug Walker	NaN	NaN	Documentary	Doug Walker	Sta Epis - Th Awa

Data Description

1. **director_name**: Name of the director who directed the movie.
2. **num_critic_for_review**: Critic Review
3. **gross**: Total revenue genreated by the movie
4. **genres**: Category of the movie
5. **actor_1_name**: Lead actor of the movie
6. **movie_title**: Name of the movie
7. **num_voted_users**: Number of people who have voted
8. **num_user_for_reviews**: User Review
9. **language**: Language of the movie
10. **imdb_score**: Score obtained by the movie
11. **movie_facebook_likes**: Total facebook likes

In [ ]:
```
df.shape
```

Out[ ]:
```
(5043, 13)
```

In [ ]:
```
df.isnull().sum().sort_values(ascending = False)
```

Out[ ]:
```
gross                      884
budget                     492
title_year                 108
director_name              104
num_critic_for_reviews      50
num_user_for_reviews        20
language                    12
actor_1_name                 7
genres                       0
movie_title                  0
num_voted_users              0
imdb_score                   0
movie_facebook_likes         0
dtype: int64
```

In [ ]:
```
df.isnull().sum(axis = 1).sort_values(ascending = False)
```

Out[ ]:
```
2342    6
2370    6
279     6
4634    5
2765    5
       ..
1702    0
1701    0
1700    0
1699    0
5042    0
Length: 5043, dtype: int64
```

Keeping only values that are not null for gross and budget columns

In [ ]:
```
df = df[df['gross'].notna()]
df = df[df['budget'].notna()]
```

In [ ]:
```
df.isnull().sum().sort_values(ascending = False)
```

Out[ ]:
```
actor_1_name               3
language                   3
num_critic_for_reviews     1
director_name              0
gross                      0
genres                     0
movie_title                0
num_voted_users            0
num_user_for_reviews       0
budget                     0
title_year                 0
imdb_score                 0
movie_facebook_likes       0
dtype: int64
```

In [ ]:
```
# Getting only null values in actors column
df[df['actor_1_name'].isnull()]
```

Out[ ]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_v
4502	Léa Pool	23.0	24784.0	Documentary	NaN	Pink Ribbons, Inc.	
4720	U. Roberto Romano	3.0	2245.0	Documentary	NaN	The Harvest/La Cosecha	
4837	Pan Nalin	15.0	16892.0	Documentary	NaN	Ayurveda: Art of Being	

In [ ]:
```
df['language'].value_counts().iloc[0:5]
```

Out[ ]:
```
English    3707
French       37
Spanish      26
Mandarin     15
German       13
Name: language, dtype: int64
```

Replacing the null values in the language column with English as it has the highest frequency

In [ ]:
```
df['language'].replace(np.nan, 'English', inplace = True)
```

In [ ]:
```
df.isnull().sum().sort_values(ascending = False)
```

Out[ ]:
```
actor_1_name          3
num_critic_for_reviews    1
director_name         0
gross                 0
genres                0
movie_title           0
num_voted_users       0
num_user_for_reviews      0
language              0
budget                0
title_year            0
imdb_score            0
movie_facebook_likes      0
dtype: int64
```

In [ ]:
```
df[df['num_critic_for_reviews'].isnull()]
```

Out[ ]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_
4711	Gene Teigland	NaN	23616.0	Mystery|Thriller	Kendyl Joi	Arnolds Park	

Getting mathematical answers for num_critic_for_reviews

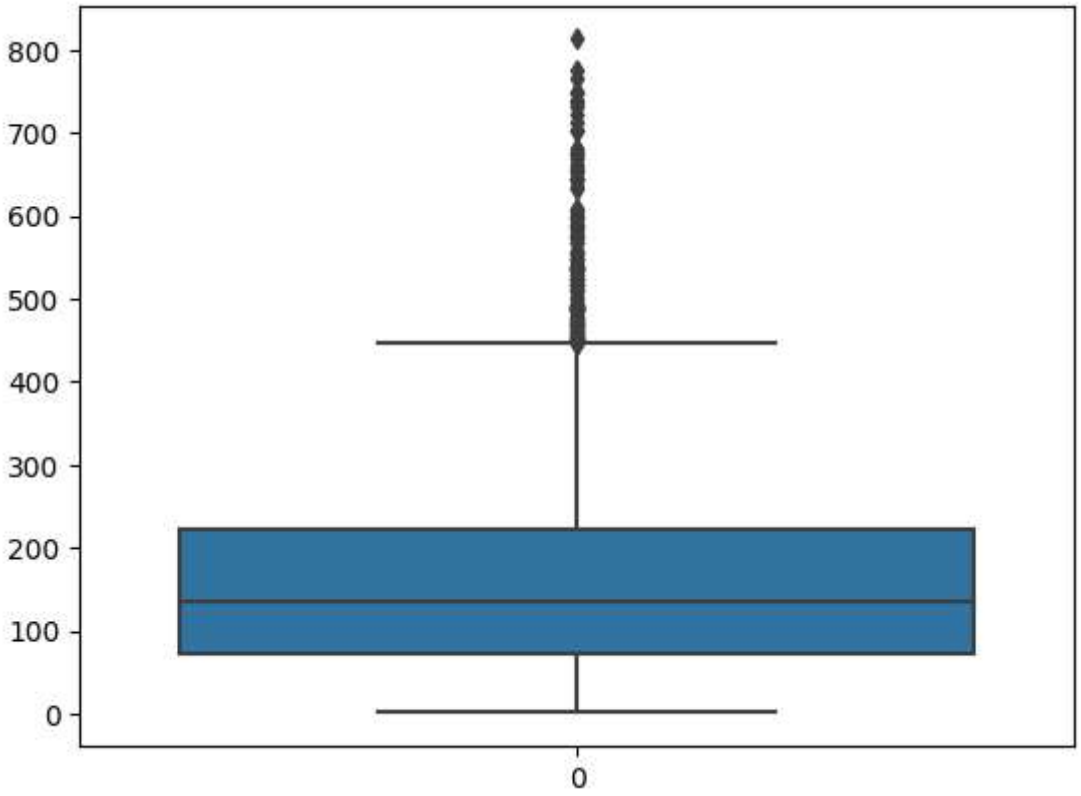
In [ ]:
```
df['num_critic_for_reviews'].describe()
```

Out[ ]:
```
count    3890.000000
mean      163.234704
std       124.053735
min         1.000000
25%        72.250000
50%       134.000000
75%       221.750000
max       813.000000
Name: num_critic_for_reviews, dtype: float64
```

Finding the outliers

In [ ]:
```
sns.boxplot(df['num_critic_for_reviews'])
```

Out[ ]:
```
<Axes: >
```



Dropping the null values

In [ ]:
```
df = df.dropna()
```

In [ ]:
```
df.shape
```

Out[ ]:
```
(3887, 13)
```

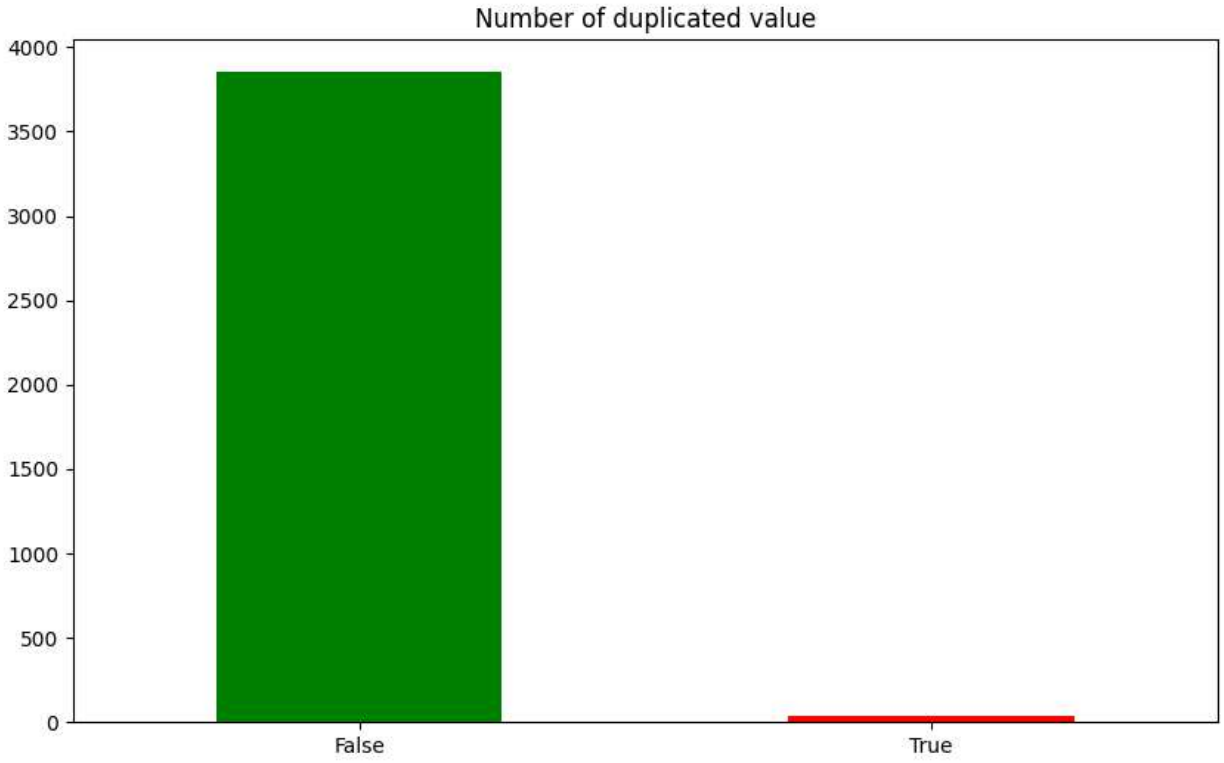
Finding the duplicated values

In [ ]:
```
dup = df.duplicated().value_counts()
dup
```

Out[ ]:
```
False    3852
True       35
dtype: int64
```

In [ ]:
```
plt.figure(figsize = (10,6))
dup.plot(kind = 'bar', color = ['g', 'r'])
plt.xticks(rotation = 360)
plt.title("Number of duplicated value")
```

Out[ ]:
```
Text(0.5, 1.0, 'Number of duplicated value')
```



Dropping the duplicated values

In [ ]:
```
df = df.drop_duplicates()
df.shape
```

Out[ ]:
```
(3852, 13)
```

In [ ]:
```
df.describe().style.background_gradient()
```

Out[ ]:

	num_critic_for_reviews	gross	num_voted_users	budget	title_year	imd
count	3852.000000	3852.000000	3852.000000	3852.000000	3852.000000	3852
mean	163.036085	50975542.371236	102442.806594	45253902.591121	2003.063344	6
std	123.937734	69326510.589619	150309.201024	223449575.930116	10.010103	1
min	1.000000	162.000000	5.000000	218.000000	1920.000000	1
25%	72.000000	6815231.500000	17308.500000	10000000.000000	1999.000000	5
50%	134.000000	27900000.000000	50588.000000	24000000.000000	2005.000000	6
75%	221.000000	65508766.750000	124194.250000	50000000.000000	2010.000000	7
max	813.000000	760505847.000000	1689764.000000	12215500000.000000	2016.000000	9

In [ ]:
```
df.isnull().sum()
```

Out[ ]:
```
director_name              0
num_critic_for_reviews     0
gross                      0
genres                     0
actor_1_name               0
movie_title                0
num_voted_users            0
num_user_for_reviews       0
language                   0
budget                     0
title_year                 0
imdb_score                 0
movie_facebook_likes       0
dtype: int64
```

Data is clean and ready for visualization

B. 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 a reference. Plot profit (y-axis) vs budget (x-axis) and observe the outliers using the appropriate chart type.

Task: Find the movies with the highest profit?

First let's create a column revenue

In [ ]:
```
df['profit'] = df['gross'] - df['budget']
df.head()
```

```
<ipython-input-25-c3018e466a0e>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user
_guide/indexing.html#returning-a-view-versus-a-copy
  df['profit'] = df['gross'] - df['budget']
```

Out[ ]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	mov
0	James Cameron	723.0	760505847.0	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	
1	Gore Verbinski	302.0	309404152.0	Action|Adventure|Fantasy	Johnny Depp	Pi Car At
2	Sam Mendes	602.0	200074175.0	Action|Adventure|Thriller	Christoph Waltz	
3	Christopher Nolan	813.0	448130642.0	Action|Thriller	Tom Hardy	T
5	Andrew Stanton	462.0	73058679.0	Action|Adventure|Sci-Fi	Daryl Sabara	John

Sorting the profit column in decending order

In [ ]:
```
top_profitable_movie = df.sort_values(['profit'], axis = 0, ascending = False)
top_profitable_movie.head()
```

Out[ ]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	r
0	James Cameron	723.0	760505847.0	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	
29	Colin Trevorrow	644.0	652177271.0	Action|Adventure|Sci-Fi|Thriller	Bryce Dallas Howard	
26	James Cameron	315.0	658672302.0	Drama|Romance	Leonardo DiCaprio	
3024	George Lucas	282.0	460935665.0	Action|Adventure|Fantasy|Sci-Fi	Harrison Ford	I
3080	Steven Spielberg	215.0	434949459.0	Family|Sci-Fi	Henry Thomas	

In [ ]:
```
# Getting top 10 values
top_10_profit = top_profitable_movie.iloc[:10]
top_10_profit[['movie_title', 'profit']]
```

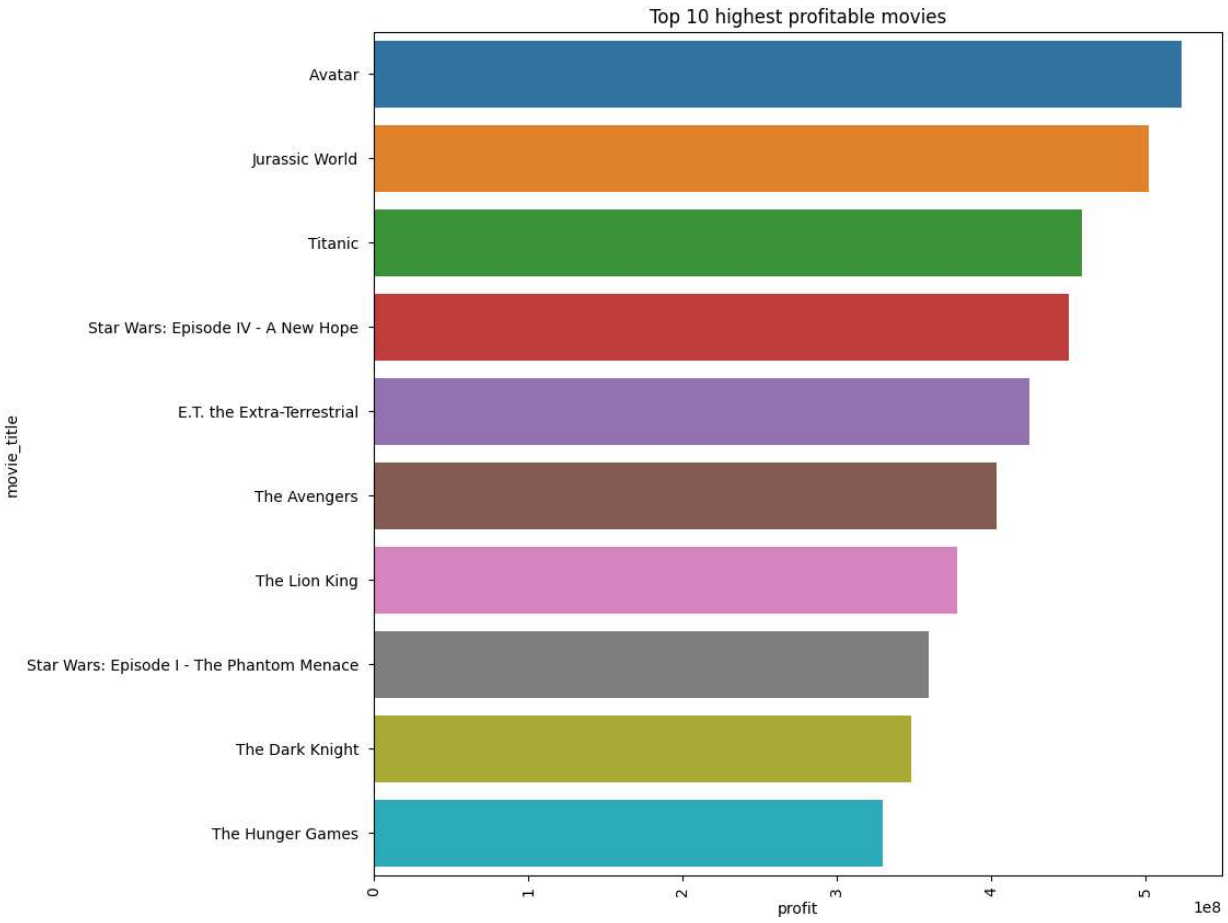
Out[ ]:

	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
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
439	The Hunger Games	329999255.0

In [ ]:
```
top_10_profit.keys()
```

Out[ ]:
```
Index(['director_name', 'num_critic_for_reviews', 'gross', 'genres',
       'actor_1_name', 'movie_title', 'num_voted_users',
       'num_user_for_reviews', 'language', 'budget', 'title_year',
       'imdb_score', 'movie_facebook_likes', 'profit'],
      dtype='object')
```

In [ ]:
```
plt.figure(figsize = (10,10))
sns.barplot(data = df, y = top_10_profit['movie_title'], x = top_10_profit['profit'])
plt.xticks(rotation = 90)
plt.title("Top 10 highest profitable movies")
plt.show()
```



Observations

Avatar is the most profitable movie which is followed by Jurrasic world.

C. 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!

Task: Find IMDB Top 250

In [ ]:
```
# Listing the data which has num_voted_users more than 25000
IMDb_Top_250 = df[df['num_voted_users'] > 25000]
IMDb_Top_250.head()
```

Out[ ]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	mov
0	James Cameron	723.0	760505847.0	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	
1	Gore Verbinski	302.0	309404152.0	Action|Adventure|Fantasy	Johnny Depp	Pi Car At
2	Sam Mendes	602.0	200074175.0	Action|Adventure|Thriller	Christoph Waltz	
3	Christopher Nolan	813.0	448130642.0	Action|Thriller	Tom Hardy	T
5	Andrew Stanton	462.0	73058679.0	Action|Adventure|Sci-Fi	Daryl Sabara	John

In [ ]:
```
# SOrting the values in descending order
IMDb_Top_250 = IMDb_Top_250.sort_values(["imdb_score"],
                        axis = 0, ascending = False)
IMDb_Top_250.head()
```

Out[ ]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	mo
1937	Frank Darabont	199.0	28341469.0	Crime|Drama	Morgan Freeman	Sh Re
3466	Francis Ford Coppola	208.0	134821952.0	Crime|Drama	Al Pacino	(
2837	Francis Ford Coppola	149.0	57300000.0	Crime|Drama	Robert De Niro	G
66	Christopher Nolan	645.0	533316061.0	Action|Crime|Drama|Thriller	Christian Bale	
4498	Sergio Leone	181.0	6100000.0	Western	Clint Eastwood	T the

Getting top 250 movies with highest IMDb score

In [ ]:
```
IMDb_Top_250 = IMDb_Top_250.iloc[:250]
pd.set_option('display.max_rows', 500)
IMDb_Top_250[['movie_title', 'imdb_score']]
```

Out[ ]:

	movie_title	imdb_score
1937	The Shawshank Redemption	9.3
3466	The Godfather	9.2
2837	The Godfather: Part II	9.0
66	The Dark Knight	9.0
4498	The Good, the Bad and the Ugly	8.9
3355	Pulp Fiction	8.9
1874	Schindler's List	8.9
339	The Lord of the Rings: The Return of the King	8.9
836	Forrest Gump	8.8
97	Inception	8.8
683	Fight Club	8.8
2051	Star Wars: Episode V - The Empire Strikes Back	8.8
270	The Lord of the Rings: The Fellowship of the R...	8.8
654	The Matrix	8.7
3867	One Flew Over the Cuckoo's Nest	8.7
1903	Goodfellas	8.7
4747	Seven Samurai	8.7
3024	Star Wars: Episode IV - A New Hope	8.7
340	The Lord of the Rings: The Two Towers	8.7
4029	City of God	8.7
1600	Se7en	8.6
96	Interstellar	8.6
2373	Spirited Away	8.6
648	Saving Private Ryan	8.6
4427	Modern Times	8.6
2158	The Silence of the Lambs	8.6
3175	American History X	8.6
3592	The Usual Suspects	8.6
3716	Memento	8.5
1233	The Prestige	8.5
2152	Raiders of the Lost Ark	8.5
4028	Whiplash	8.5
1448	The Pianist	8.5
712	The Green Mile	8.5
4259	The Lives of Others	8.5

	movie_title	imdb_score
1571	Apocalypse Now	8.5
361	The Departed	8.5
3	The Dark Knight Rises	8.5
4921	Children of Heaven	8.5
288	Terminator 2: Judgment Day	8.5
296	Django Unchained	8.5
3277	Alien	8.5
2363	Back to the Future	8.5
283	Gladiator	8.5
509	The Lion King	8.5
2242	Psycho	8.5
2323	Princess Mononoke	8.4
3849	Requiem for a Dream	8.4
2970	Das Boot	8.4
1714	Once Upon a Time in America	8.4
58	WALL·E	8.4
2486	Aliens	8.4
1536	Star Wars: Episode VI - Return of the Jedi	8.4
2606	American Beauty	8.4
573	Braveheart	8.4
2644	Lawrence of Arabia	8.4
4105	Oldboy	8.4
4659	A Separation	8.4
1329	Baahubali: The Beginning	8.4
4496	Reservoir Dogs	8.4
1298	Amélie	8.4
1906	Scarface	8.3
78	Inside Out	8.3
3017	Snatch	8.3
2734	Metropolis	8.3
588	Inglourious Basterds	8.3
2425	Raging Bull	8.3
2223	Eternal Sunshine of the Spotless Mind	8.3
1416	L.A. Confidential	8.3
1588	Toy Story	8.3

	movie_title	imdb_score
2866	Room	8.3
3668	The Sting	8.3
4033	The Hunt	8.3
4155	Some Like It Hot	8.3
4795	Monty Python and the Holy Grail	8.3
1039	Indiana Jones and the Last Crusade	8.3
67	Up	8.3
43	Toy Story 3	8.3
3089	Good Will Hunting	8.3
2407	Amadeus	8.3
2829	Downfall	8.3
120	Batman Begins	8.3
3079	2001: A Space Odyssey	8.3
2760	Unforgiven	8.3
1359	The Thing	8.2
2551	Pan's Labyrinth	8.2
508	A Beautiful Mind	8.2
338	Finding Nemo	8.2
2250	Into the Wild	8.2
4458	Lock, Stock and Two Smoking Barrels	8.2
960	V for Vendetta	8.2
4638	On the Waterfront	8.2
1869	Gran Torino	8.2
920	Casino	8.2
1754	Die Hard	8.2
27	Captain America: Civil War	8.2
4000	The Secret in Their Eyes	8.2
3550	Incendies	8.2
3970	Gone with the Wind	8.2
2047	Howl's Moving Castle	8.2
4050	Trainspotting	8.2
1978	Warrior	8.2
308	The Wolf of Wall Street	8.2
4066	The Bridge on the River Kwai	8.2
2703	The Big Lebowski	8.2

	movie_title	imdb_score
93	How to Train Your Dragon	8.2
1777	Blade Runner	8.2
179	The Revenant	8.1
128	Mad Max: Fury Road	8.1
2830	The Sea Inside	8.1
278	The Martian	8.1
2755	Groundhog Day	8.1
2614	The Imitation Game	8.1
2480	Hotel Rwanda	8.1
2651	The Princess Bride	8.1
1213	Sin City	8.1
183	The Bourne Ultimatum	8.1
1578	The Grand Budapest Hotel	8.1
2914	Tae Guk Gi: The Brotherhood of War	8.1
3584	Butch Cassidy and the Sundance Kid	8.1
715	Gone Girl	8.1
812	Deadpool	8.1
4238	The Best Years of Our Lives	8.1
1885	No Country for Old Men	8.1
452	Shutter Island	8.1
4190	Before Sunrise	8.1
4461	The Celebration	8.1
4157	The Wizard of Oz	8.1
1911	There Will Be Blood	8.1
697	Jurassic Park	8.1
3889	Annie Hall	8.1
3854	Donnie Darko	8.1
1181	The Sixth Sense	8.1
1875	The Help	8.1
17	The Avengers	8.1
719	The Truman Show	8.1
2174	12 Years a Slave	8.1
238	Monsters, Inc.	8.1
1373	Rush	8.1
3582	Platoon	8.1

	movie_title	imdb_score
3575	The Terminator	8.1
205	Pirates of the Caribbean: The Curse of the Bla...	8.1
3553	Elite Squad	8.1
855	Kill Bill: Vol. 1	8.1
2194	Spotlight	8.1
95	Guardians of the Galaxy	8.1
3423	Akira	8.1
3362	Stand by Me	8.1
4530	Rocky	8.1
1604	Million Dollar Baby	8.1
1084	Prisoners	8.1
4267	Amores Perros	8.1
1374	Magnolia	8.0
1601	District 9	8.0
2545	The Artist	8.0
223	Life of Pi	8.0
2539	Dead Poets Society	8.0
2058	Her	8.0
1747	Aladdin	8.0
2356	Dances with Wolves	8.0
1603	Mystic River	8.0
2607	The King's Speech	8.0
3179	The Straight Story	8.0
349	The Incredibles	8.0
3906	Boyhood	8.0
3456	Persepolis	8.0
911	Catch Me If You Can	8.0
1334	Serenity	8.0
851	The Pursuit of Happyness	8.0
307	Blood Diamond	8.0
3714	Dallas Buyers Club	8.0
3741	Shaun of the Dead	8.0
3821	Sling Blade	8.0
47	X-Men: Days of Future Past	8.0
3344	My Name Is Khan	8.0

	movie_title	imdb_score
4054	Bowling for Columbine	8.0
4144	Central Station	8.0
4158	Young Frankenstein	8.0
602	Big Fish	8.0
4266	Before Sunset	8.0
4284	Waltz with Bashir	8.0
391	Cinderella Man	8.0
4897	A Fistful of Dollars	8.0
3359	The Sound of Music	8.0
858	Kill Bill: Vol. 2	8.0
3287	Sicko	8.0
2917	Jaws	8.0
160	Star Trek	8.0
2700	Brazil	8.0
2712	In Bruges	8.0
2723	Mulholland Drive	8.0
2762	Slumdog Millionaire	8.0
2835	Black Swan	8.0
2884	The Perks of Being a Wallflower	8.0
2898	True Romance	8.0
1217	JFK	8.0
3280	Fiddler on the Roof	8.0
2916	The Exorcist	8.0
2907	Dancer in the Dark	8.0
1868	Rain Man	8.0
119	Ratatouille	8.0
2944	Casino Royale	8.0
286	Casino Royale	8.0
3026	Doctor Zhivago	8.0
1008	The Iron Giant	8.0
1606	The Notebook	7.9
927	Shrek	7.9
1813	The Right Stuff	7.9
525	Children of Men	7.9
1807	The Blues Brothers	7.9

	movie_title	imdb_score
1735	Walk the Line	7.9
1171	Hero	7.9
716	The Bourne Identity	7.9
1713	Glory	7.9
845	Captain Phillips	7.9
706	The Hateful Eight	7.9
1748	Straight Outta Compton	7.9
788	Almost Famous	7.9
639	The Insider	7.9
0	Avatar	7.9
1871	Taken	7.9
89	Big Hero 6	7.9
162	How to Train Your Dragon 2	7.9
2666	The Remains of the Day	7.9
2667	Boogie Nights	7.9
2863	Letters from Iwo Jima	7.9
3080	E.T. the Extra-Terrestrial	7.9
3193	Crash	7.9
3264	Amour	7.9
3357	Nightcrawler	7.9
3361	Little Miss Sunshine	7.9
99	The Hobbit: An Unexpected Journey	7.9
3510	Veer-Zaara	7.9
3595	The Wrestler	7.9
2558	Hot Fuzz	7.9
75	Edge of Tomorrow	7.9
3677	The Chorus	7.9
3680	Do the Right Thing	7.9
69	Iron Man	7.9
3768	Moon	7.9
4082	Before Midnight	7.9
23	The Hobbit: The Desolation of Smaug	7.9
4415	Nine Queens	7.9
4640	4 Months, 3 Weeks and 2 Days	7.9
4821	Halloween	7.9

	movie_title	imdb_score
4931	Once	7.9
2605	Crouching Tiger, Hidden Dragon	7.9
3029	The Fighter	7.9
2177	Edward Scissorhands	7.9
2487	My Fair Lady	7.9

In [ ]:
```
IMDb_Top_250.groupby(['imdb_score'])['movie_title'].value_counts().iloc[:250]
```

Out[ ]:

imdb_score	movie_title	
7.9	4 Months, 3 Weeks and 2 Days	1
	Almost Famous	1
	Amour	1
	Avatar	1
	Before Midnight	1
	Big Hero 6	1
	Boogie Nights	1
	Captain Phillips	1
	Children of Men	1
	Crash	1
	Crouching Tiger, Hidden Dragon	1
	Do the Right Thing	1
	E.T. the Extra-Terrestrial	1
	Edge of Tomorrow	1
	Edward Scissorhands	1
	Glory	1
	Halloween	1
	Hero	1
	Hot Fuzz	1
	How to Train Your Dragon 2	1
	Iron Man	1
	Letters from Iwo Jima	1
	Little Miss Sunshine	1
	Moon	1
	My Fair Lady	1
	Nightcrawler	1
	Nine Queens	1
	Once	1
	Shrek	1
	Straight Outta Compton	1
	Taken	1
	The Blues Brothers	1
	The Bourne Identity	1
	The Chorus	1
	The Fighter	1
	The Hateful Eight	1
	The Hobbit: An Unexpected Journey	1
	The Hobbit: The Desolation of Smaug	1
	The Insider	1
	The Notebook	1
	The Remains of the Day	1
	The Right Stuff	1
	The Wrestler	1
	Veer-Zaara	1
	Walk the Line	1
8.0	Casino Royale	2
	A Fistful of Dollars	1
	Aladdin	1
	Before Sunset	1
	Big Fish	1
	Black Swan	1
	Blood Diamond	1
	Bowling for Columbine	1
	Boyhood	1
	Brazil	1
	Catch Me If You Can	1
	Central Station	1
	Cinderella Man	1
	Dallas Buyers Club	1

	Dancer in the Dark	1
	Dances with Wolves	1
	Dead Poets Society	1
	District 9	1
	Doctor Zhivago	1
	Fiddler on the Roof	1
	Her	1
	In Bruges	1
	JFK	1
	Jaws	1
	Kill Bill: Vol. 2	1
	Life of Pi	1
	Magnolia	1
	Mulholland Drive	1
	My Name Is Khan	1
	Mystic River	1
	Persepolis	1
	Rain Man	1
	Ratatouille	1
	Serenity	1
	Shaun of the Dead	1
	Sicko	1
	Sling Blade	1
	Slumdog Millionaire	1
	Star Trek	1
	The Artist	1
	The Exorcist	1
	The Incredibles	1
	The Iron Giant	1
	The King's Speech	1
	The Perks of Being a Wallflower	1
	The Pursuit of Happyness	1
	The Sound of Music	1
	The Straight Story	1
	True Romance	1
	Waltz with Bashir	1
	X-Men: Days of Future Past	1
	Young Frankenstein	1
8.1	12 Years a Slave	1
	Akira	1
	Amores Perros	1
	Annie Hall	1
	Before Sunrise	1
	Butch Cassidy and the Sundance Kid	1
	Deadpool	1
	Donnie Darko	1
	Elite Squad	1
	Gone Girl	1
	Groundhog Day	1
	Guardians of the Galaxy	1
	Hotel Rwanda	1
	Jurassic Park	1
	Kill Bill: Vol. 1	1
	Mad Max: Fury Road	1
	Million Dollar Baby	1
	Monsters, Inc.	1
	No Country for Old Men	1
	Pirates of the Caribbean: The Curse of the Black Pearl	1
	Platoon	1
	Prisoners	1

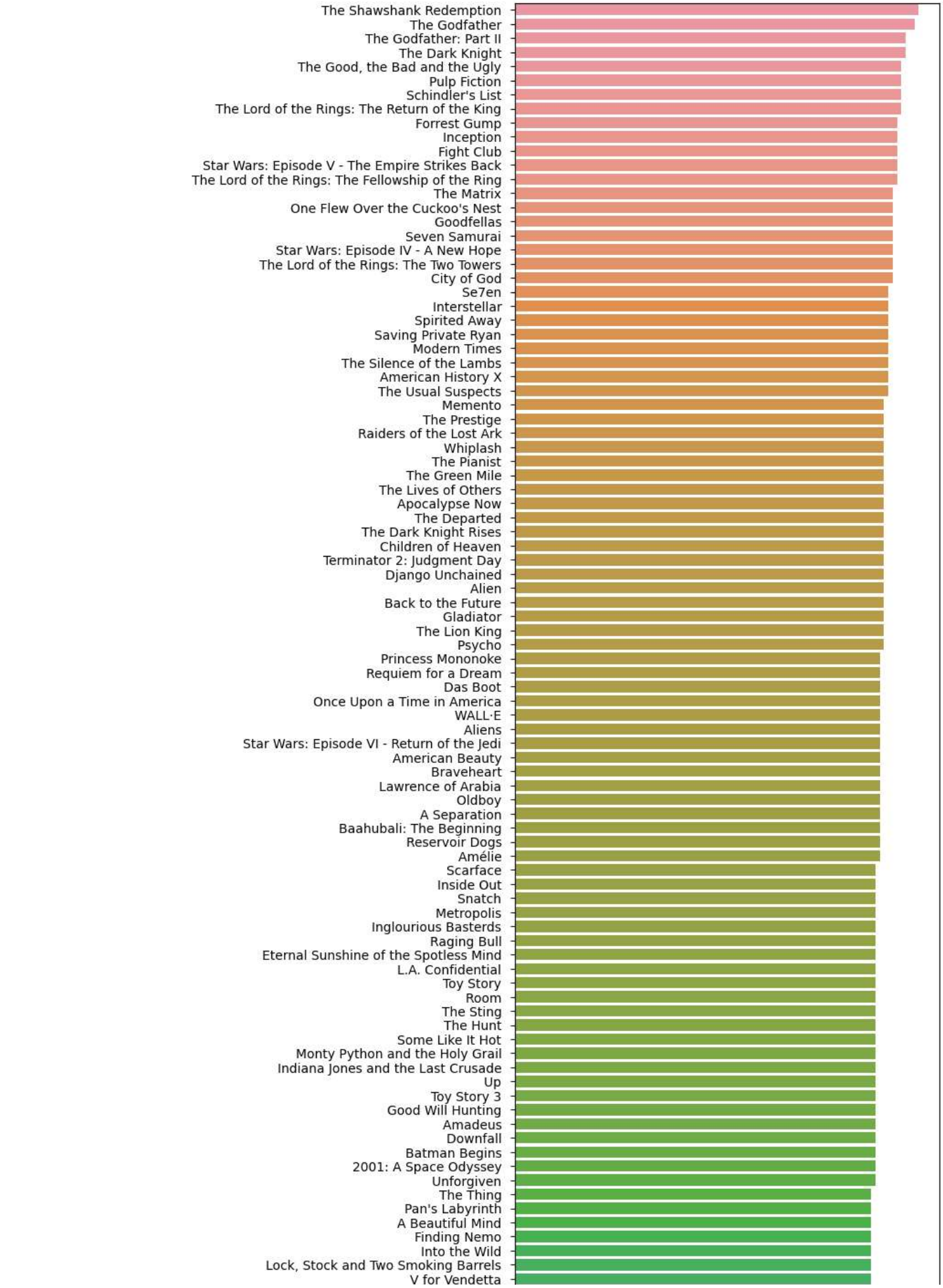
	Rocky	1
	Rush	1
	Shutter Island	1
	Sin City	1
	Spotlight	1
	Stand by Me	1
	Tae Guk Gi: The Brotherhood of War	1
	The Avengers	1
	The Best Years of Our Lives	1
	The Bourne Ultimatum	1
	The Celebration	1
	The Grand Budapest Hotel	1
	The Help	1
	The Imitation Game	1
	The Martian	1
	The Princess Bride	1
	The Revenant	1
	The Sea Inside	1
	The Sixth Sense	1
	The Terminator	1
	The Truman Show	1
	The Wizard of Oz	1
	There Will Be Blood	1
8.2	A Beautiful Mind	1
	Blade Runner	1
	Captain America: Civil War	1
	Casino	1
	Die Hard	1
	Finding Nemo	1
	Gone with the Wind	1
	Gran Torino	1
	How to Train Your Dragon	1
	Howl's Moving Castle	1
	Incendies	1
	Into the Wild	1
	Lock, Stock and Two Smoking Barrels	1
	On the Waterfront	1
	Pan's Labyrinth	1
	The Big Lebowski	1
	The Bridge on the River Kwai	1
	The Secret in Their Eyes	1
	The Thing	1
	The Wolf of Wall Street	1
	Trainspotting	1
	V for Vendetta	1
	Warrior	1
8.3	2001: A Space Odyssey	1
	Amadeus	1
	Batman Begins	1
	Downfall	1
	Eternal Sunshine of the Spotless Mind	1
	Good Will Hunting	1
	Indiana Jones and the Last Crusade	1
	Inglourious Basterds	1
	Inside Out	1
	L.A. Confidential	1
	Metropolis	1
	Monty Python and the Holy Grail	1
	Raging Bull	1
	Room	1

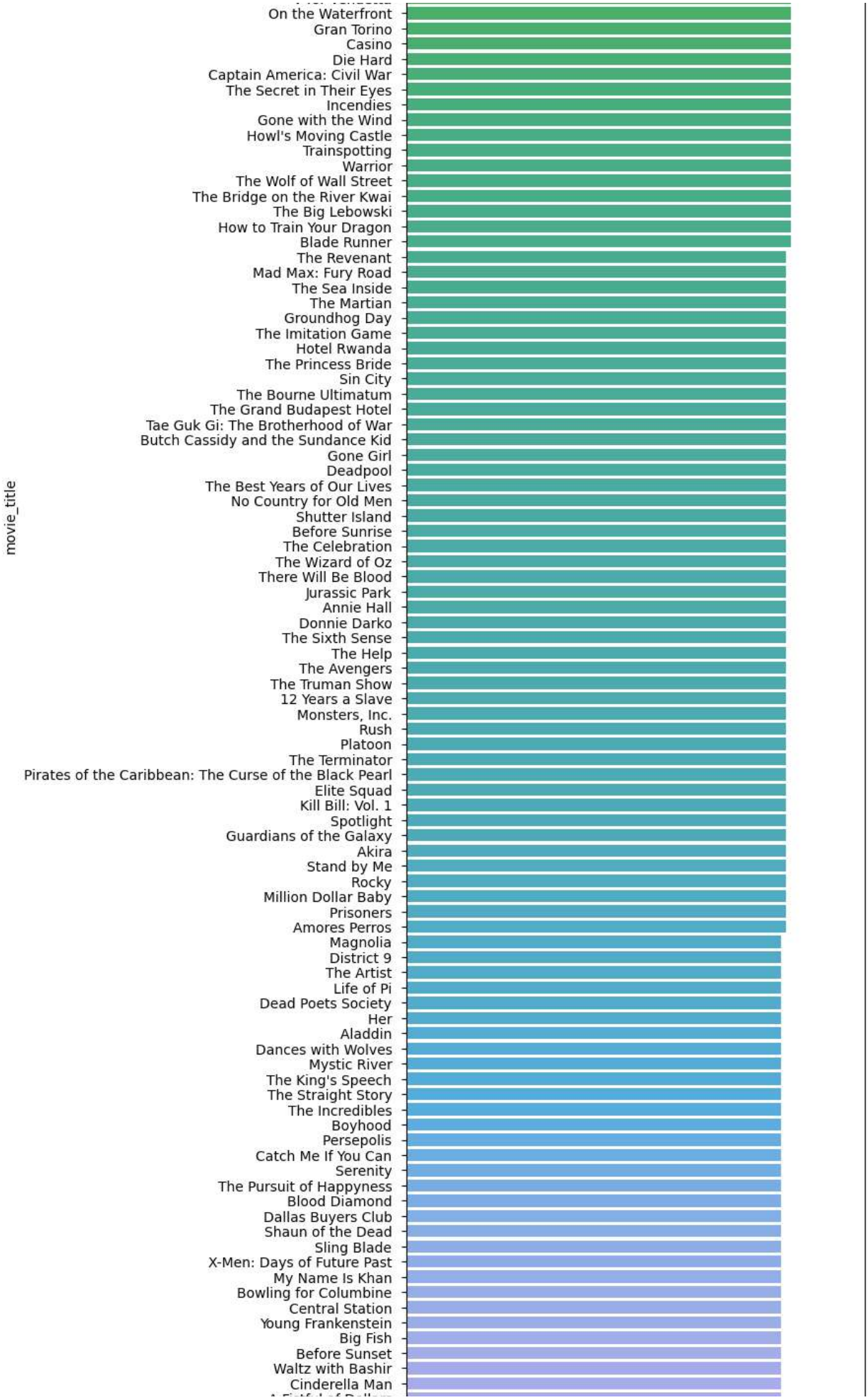
	Scarface	1
	Snatch	1
	Some Like It Hot	1
	The Hunt	1
	The Sting	1
	Toy Story 3	1
	Toy Story	1
	Unforgiven	1
	Up	1
8.4	A Separation	1
	Aliens	1
	American Beauty	1
	Amélie	1
	Baahubali: The Beginning	1
	Braveheart	1
	Das Boot	1
	Lawrence of Arabia	1
	Oldboy	1
	Once Upon a Time in America	1
	Princess Mononoke	1
	Requiem for a Dream	1
	Reservoir Dogs	1
	Star Wars: Episode VI - Return of the Jedi	1
	WALL·E	1
8.5	Alien	1
	Apocalypse Now	1
	Back to the Future	1
	Children of Heaven	1
	Django Unchained	1
	Gladiator	1
	Memento	1
	Psycho	1
	Raiders of the Lost Ark	1
	Terminator 2: Judgment Day	1
	The Dark Knight Rises	1
	The Departed	1
	The Green Mile	1
	The Lion King	1
	The Lives of Others	1
	The Pianist	1
	The Prestige	1
	Whiplash	1
8.6	American History X	1
	Interstellar	1
	Modern Times	1
	Saving Private Ryan	1
	Se7en	1
	Spirited Away	1
	The Silence of the Lambs	1
	The Usual Suspects	1
8.7	City of God	1
	Goodfellas	1
	One Flew Over the Cuckoo's Nest	1
	Seven Samurai	1
	Star Wars: Episode IV - A New Hope	1
	The Lord of the Rings: The Two Towers	1
	The Matrix	1
8.8	Fight Club	1
	Forrest Gump	1
	Inception	1

```
                Star Wars: Episode V - The Empire Strikes Back                1
                The Lord of the Rings: The Fellowship of the Ring             1
8.9             Pulp Fiction                                                  1
                Schindler's List                                              1
                The Good, the Bad and the Ugly                                1
                The Lord of the Rings: The Return of the King                 1
9.0             The Dark Knight                                               1
                The Godfather: Part II                                        1
9.2             The Godfather                                                 1
9.3             The Shawshank Redemption                                      1
Name: movie_title, dtype: int64
```

In [ ]:
```
plt.figure(figsize = (6,50))
sns.barplot(data = IMDb_Top_250, y = IMDb_Top_250['movie_title'], x = IMDb_Top_250['imdb
```

Out[ ]:
```
<Axes: xlabel='imdb_score', ylabel='movie_title'>
```


The Shawshank Redemption
The Godfather
The Godfather: Part II
The Dark Knight
The Good, the Bad and the Ugly
Pulp Fiction
Schindler's List
The Lord of the Rings: The Return of the King
Forrest Gump
Inception
Fight Club
Star Wars: Episode V - The Empire Strikes Back
The Lord of the Rings: The Fellowship of the Ring
The Matrix
One Flew Over the Cuckoo's Nest
Goodfellas
Seven Samurai
Star Wars: Episode IV - A New Hope
The Lord of the Rings: The Two Towers
City of God
Se7en
Interstellar
Spirited Away
Saving Private Ryan
Modern Times
The Silence of the Lambs
American History X
The Usual Suspects
Memento
The Prestige
Raiders of the Lost Ark
Whiplash
The Pianist
The Green Mile
The Lives of Others
Apocalypse Now
The Departed
The Dark Knight Rises
Children of Heaven
Terminator 2: Judgment Day
Django Unchained
Alien
Back to the Future
Gladiator
The Lion King
Psycho
Princess Mononoke
Requiem for a Dream
Das Boot
Once Upon a Time in America
WALL·E
Aliens
Star Wars: Episode VI - Return of the Jedi
American Beauty
Braveheart
Lawrence of Arabia
Oldboy
A Separation
Baahubali: The Beginning
Reservoir Dogs
Amélie
Scarface
Inside Out
Snatch
Metropolis
Inglourious Basterds
Raging Bull
Eternal Sunshine of the Spotless Mind
L.A. Confidential
Toy Story
Room
The Sting
The Hunt
Some Like It Hot
Monty Python and the Holy Grail
Indiana Jones and the Last Crusade
Up
Toy Story 3
Good Will Hunting
Amadeus
Downfall
Batman Begins
2001: A Space Odyssey
Unforgiven
The Thing
Pan's Labyrinth
A Beautiful Mind
Finding Nemo
Into the Wild
Lock, Stock and Two Smoking Barrels
V for Vendetta


movie_title
On the Waterfront
Gran Torino
Casino
Die Hard
Captain America: Civil War
The Secret in Their Eyes
Incendies
Gone with the Wind
Howl's Moving Castle
Trainspotting
Warrior
The Wolf of Wall Street
The Bridge on the River Kwai
The Big Lebowski
How to Train Your Dragon
Blade Runner
The Revenant
Mad Max: Fury Road
The Sea Inside
The Martian
Groundhog Day
The Imitation Game
Hotel Rwanda
The Princess Bride
Sin City
The Bourne Ultimatum
The Grand Budapest Hotel
Tae Guk Gi: The Brotherhood of War
Butch Cassidy and the Sundance Kid
Gone Girl
Deadpool
The Best Years of Our Lives
No Country for Old Men
Shutter Island
Before Sunrise
The Celebration
The Wizard of Oz
There Will Be Blood
Jurassic Park
Annie Hall
Donnie Darko
The Sixth Sense
The Help
The Avengers
The Truman Show
12 Years a Slave
Monsters, Inc.
Rush
Platoon
The Terminator
Pirates of the Caribbean: The Curse of the Black Pearl
Elite Squad
Kill Bill: Vol. 1
Spotlight
Guardians of the Galaxy
Akira
Stand by Me
Rocky
Million Dollar Baby
Prisoners
Amores Perros
Magnolia
District 9
The Artist
Life of Pi
Dead Poets Society
Her
Aladdin
Dances with Wolves
Mystic River
The King's Speech
The Straight Story
The Incredibles
Boyhood
Persepolis
Catch Me If You Can
Serenity
The Pursuit of Happyness
Blood Diamond
Dallas Buyers Club
Shaun of the Dead
Sling Blade
X-Men: Days of Future Past
My Name Is Khan
Bowling for Columbine
Central Station
Young Frankenstein
Big Fish
Before Sunset
Waltz with Bashir
Cinderella Man

C. 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!

Task: Find IMDB Top 250

Listing with language which is not english

In [ ]:
```
non_english = IMDb_Top_250[IMDb_Top_250['language'] != 'English']
non_english.head()
```

Out[ ]:

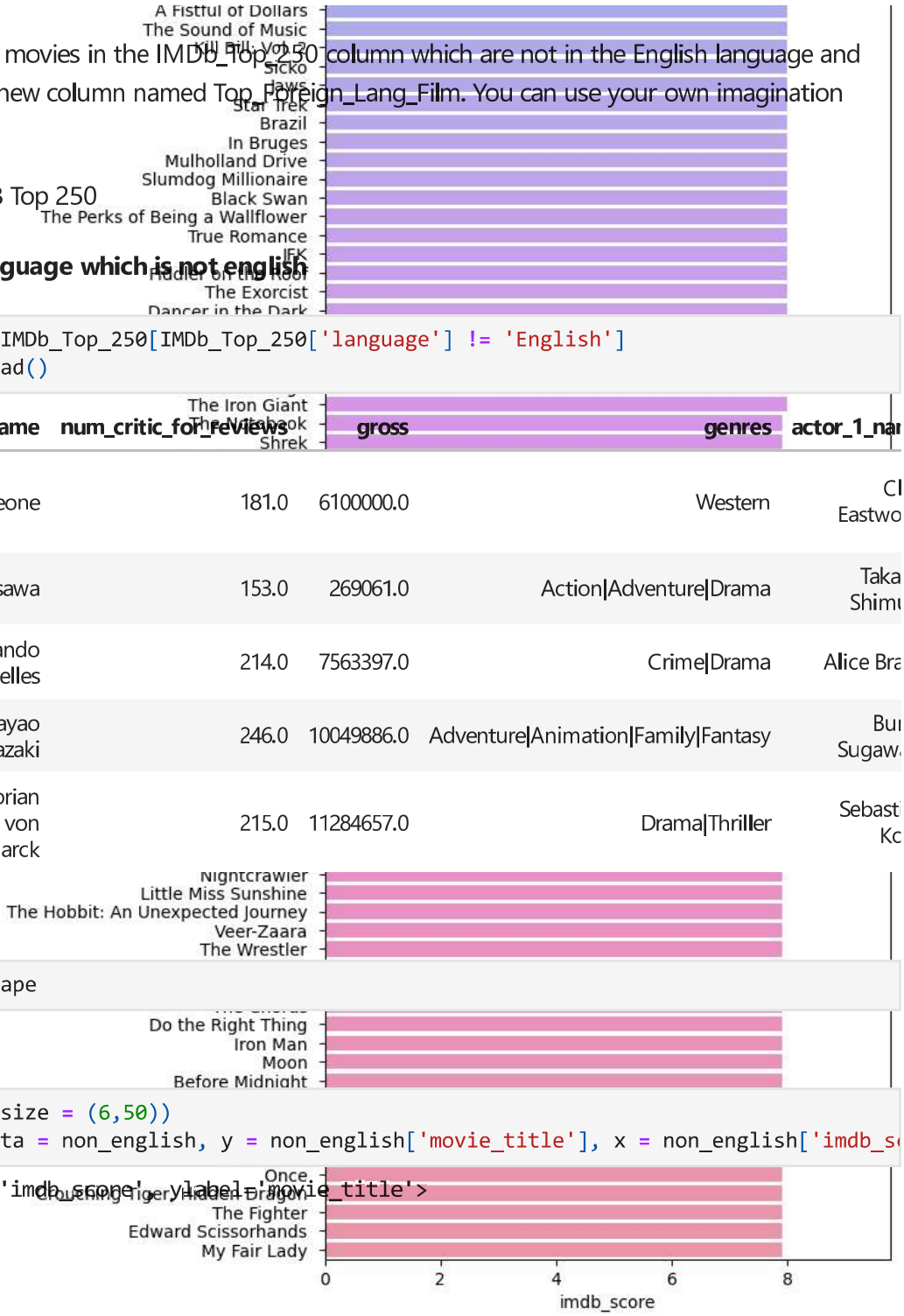
	director_name	num_critic_for_reviews	gross	genres	actor_1_nar
4498	Sergio Leone	181.0	6100000.0	Western	Cl Eastwo
4747	Akira Kurosawa	153.0	269061.0	Action|Adventure|Drama	Taka Shimu
4029	Fernando Meirelles	214.0	7563397.0	Crime|Drama	Alice Bra
2373	Hayao Miyazaki	246.0	10049886.0	Adventure|Animation|Family|Fantasy	Bur Sugaw
4259	Florian Henckel von Donnersmarck	215.0	11284657.0	Drama|Thriller	Sebast Kc

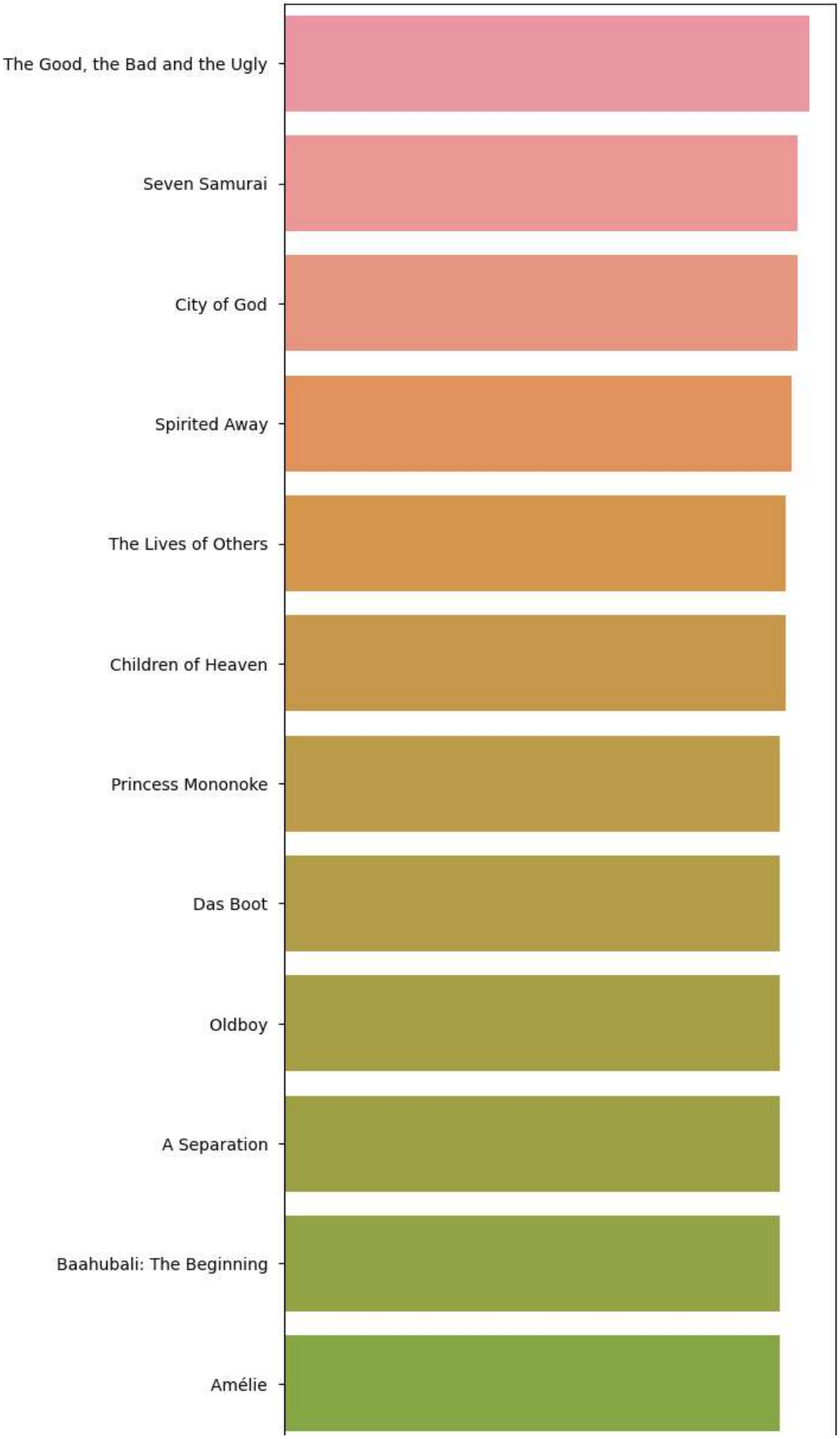
In [ ]:
```
non_english.shape
```

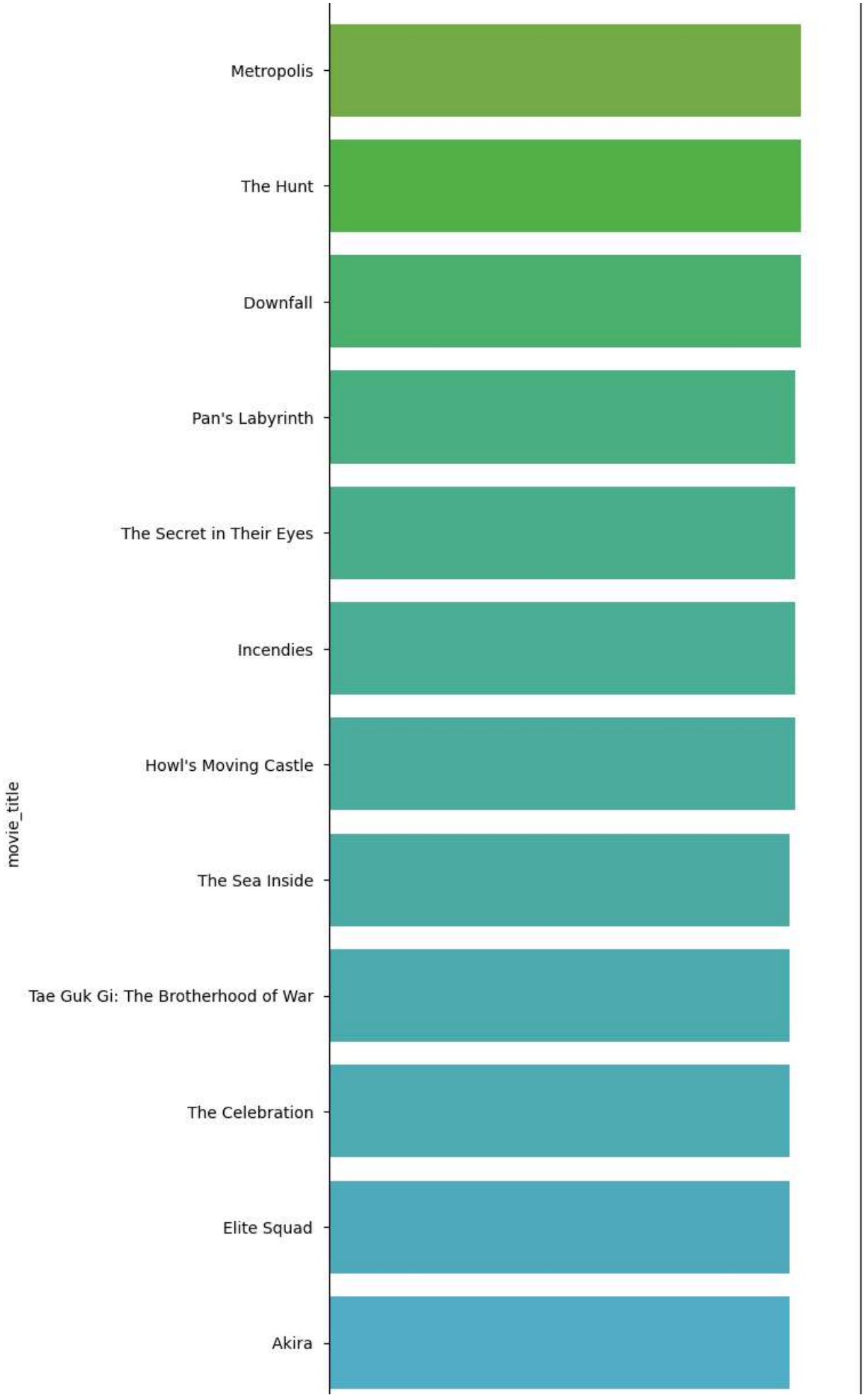
Out[ ]:
```
(38, 14)
```

In [ ]:
```
plt.figure(figsize = (6,50))
sns.barplot(data = non_english, y = non_english['movie_title'], x = non_english['imdb_s
```

Out[ ]:
```
<Axes: xlabel='imdb_score', ylabel='movie_title'>
```




The Good, the Bad and the Ugly
Seven Samurai
City of God
Spirited Away
The Lives of Others
Children of Heaven
Princess Mononoke
Das Boot
Oldboy
A Separation
Baahubali: The Beginning
Amélie


movie_title
Metropolis
The Hunt
Downfall
Pan's Labyrinth
The Secret in Their Eyes
Incendies
Howl's Moving Castle
The Sea Inside
Tae Guk Gi: The Brotherhood of War
The Celebration
Elite Squad
Akira

D. 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.

Task: Find the best directors

Grouping with directors name and imdb score and sorting in descending order

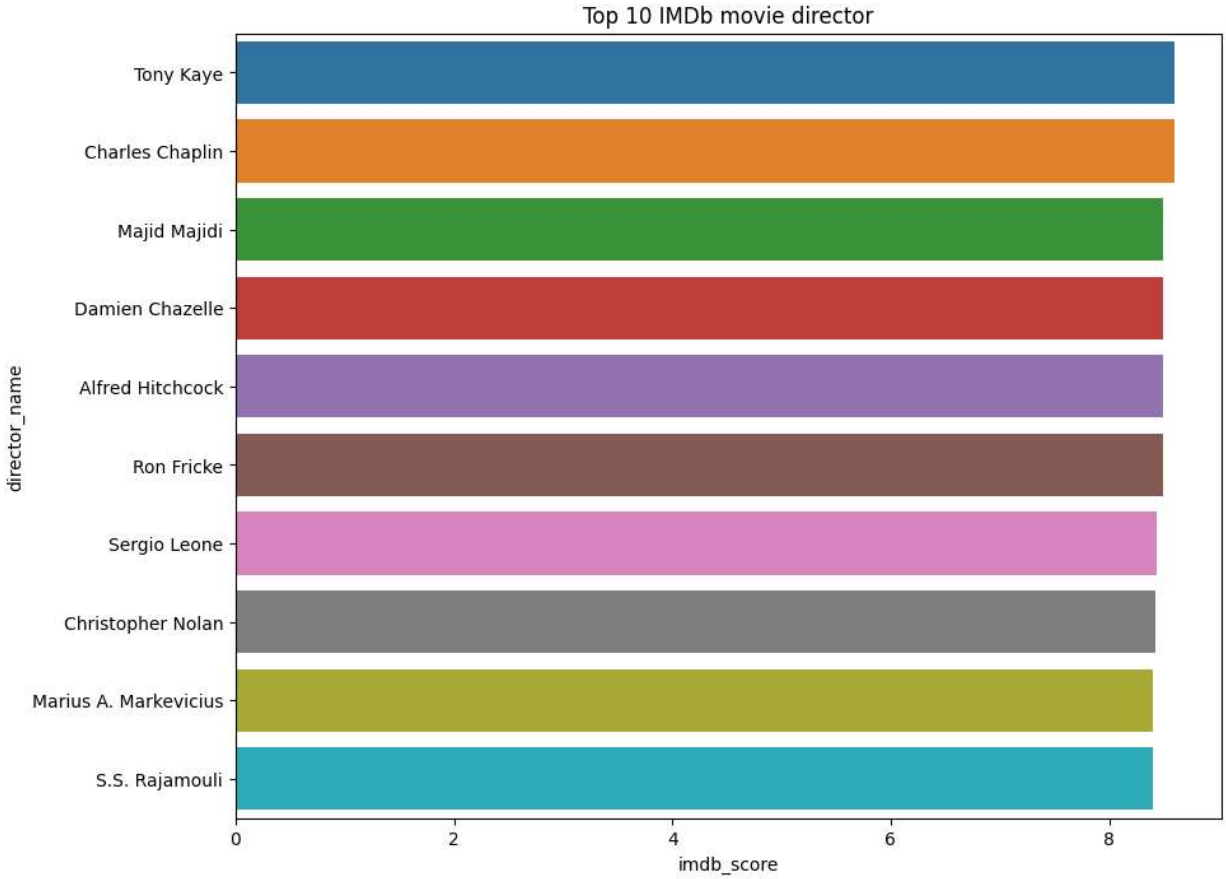
In [ ]:
```
top_10_director = df.groupby(['director_name'])['imdb_score'].mean().reset_index()
top_10_director = top_10_director.sort_values(['imdb_score'], axis = 0, ascending = Fals
top_10_director.head(10)
```

Out[ ]:

	director_name	imdb_score
1672	Tony Kaye	8.600000
216	Charles Chaplin	8.600000
1015	Majid Majidi	8.500000
302	Damien Chazelle	8.500000
45	Alfred Hitchcock	8.500000
1437	Ron Fricke	8.500000
1495	Sergio Leone	8.433333
260	Christopher Nolan	8.425000
1033	Marius A. Markevicius	8.400000
1464	S.S. Rajamouli	8.400000

In [ ]:
```
plt.figure(figsize = (10,8))
sns.barplot(data = top_10_director, x = top_10_director['imdb_score'].iloc[:10],
                                    y = top_10_director['director_name'].iloc[:10])
plt.title("Top 10 IMDb movie director")
```

Out[ ]:
```
Text(0.5, 1.0, 'Top 10 IMDb movie director')
```



Observation

Tony kane is the director with highest IMDb score

In [ ]:
```
df.duplicated().sum()
```

Out[ ]:
```
0
```

E. Popular Genres: Perform this step using the knowledge gained while performing previous steps.

Task: Find popular genres

In [ ]:
```
IMDb_Top_250.head()
```

Out[ ]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	m
1937	Frank Darabont	199.0	28341469.0	Crime|Drama	Morgan Freeman	Sh Re
3466	Francis Ford Coppola	208.0	134821952.0	Crime|Drama	Al Pacino	(
2837	Francis Ford Coppola	149.0	57300000.0	Crime|Drama	Robert De Niro	G
66	Christopher Nolan	645.0	533316061.0	Action|Crime|Drama|Thriller	Christian Bale	
4498	Sergio Leone	181.0	6100000.0	Western	Clint Eastwood	T th

Sorting the top Genres with respect to IMDb score

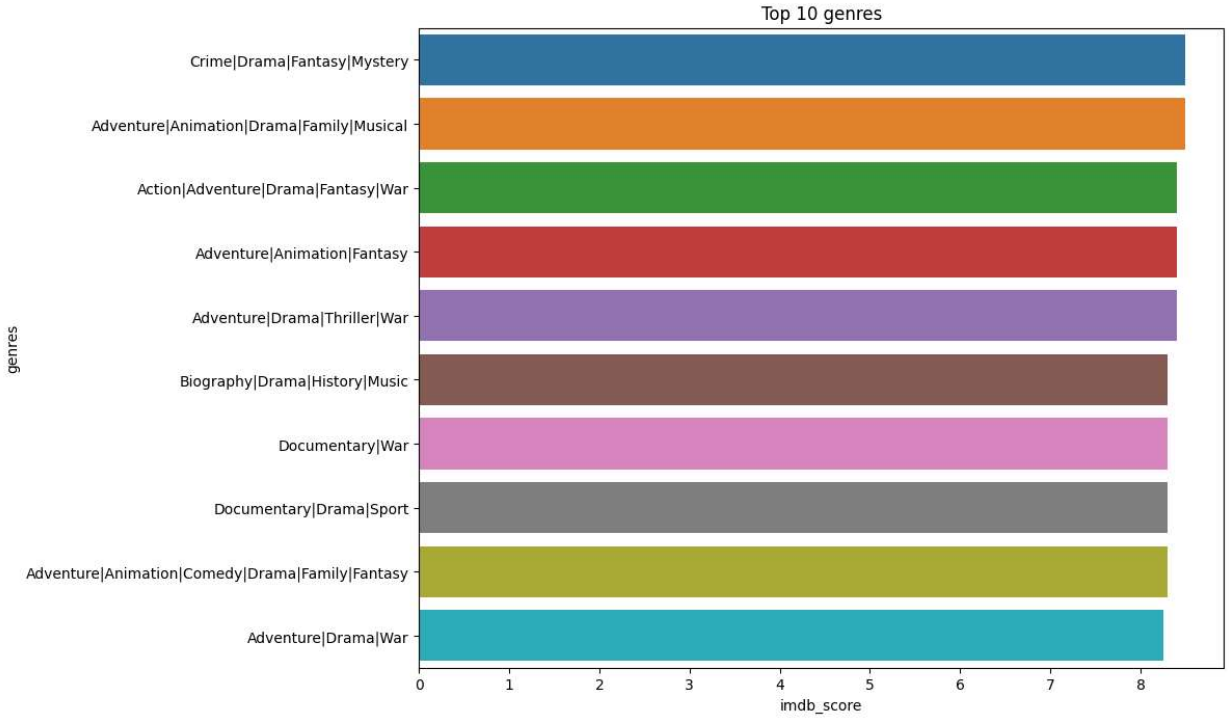
In [ ]:
```
popular_genres = df.groupby(['genres'])['imdb_score'].mean().reset_index()
popular_genres.sort_values(['imdb_score'], axis = 0, ascending = False, inplace = True)
popular_genres.head(15)
```

Out[ ]:

	genres	imdb_score
607	Crime|Drama|Fantasy|Mystery	8.50
281	Adventure|Animation|Drama|Family|Musical	8.50
60	Action|Adventure|Drama|Fantasy|War	8.40
290	Adventure|Animation|Fantasy	8.40
372	Adventure|Drama|Thriller|War	8.40
468	Biography|Drama|History|Music	8.30
647	Documentary|War	8.30
641	Documentary|Drama|Sport	8.30
258	Adventure|Animation|Comedy|Drama|Family|Fantasy	8.30
374	Adventure|Drama|War	8.25
713	Drama|Mystery|War	8.20
444	Biography|Crime|Documentary|History	8.20
675	Drama|Fantasy|War	8.20
373	Adventure|Drama|Thriller|Western	8.10
116	Action|Animation|Sci-Fi	8.10

In [ ]:
```
plt.figure(figsize = (10,8))
sns.barplot(data = popular_genres, x = popular_genres['imdb_score'].iloc[:10],y = popul
plt.title("Top 10 genres")
```

Out[ ]: Text(0.5, 1.0, 'Top 10 genres')



Observation

Looks like Crime, drama, fantasy and mystery is the most liked genre

F. 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.

Task: Find the critic-favorite and audience-favorite actors

In [ ]:
```
df.head()
```

Out[ ]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	mov
0	James Cameron	723.0	760505847.0	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	
1	Gore Verbinski	302.0	309404152.0	Action|Adventure|Fantasy	Johnny Depp	Pi Car At
2	Sam Mendes	602.0	200074175.0	Action|Adventure|Thriller	Christoph Waltz	
3	Christopher Nolan	813.0	448130642.0	Action|Thriller	Tom Hardy	T
5	Andrew Stanton	462.0	73058679.0	Action|Adventure|Sci-Fi	Daryl Sabara	John

Creating column with Meryl Streep, Leonardo DiCaprio and Brad Pitt

In [ ]:
```
Meryl_Streep = df[df['actor_1_name'] == 'Meryl Streep']
Leo_Caprio = df[df['actor_1_name'] == 'Leonardo DiCaprio']
Brad_Pitt = df[df['actor_1_name'] == 'Brad Pitt']
```

In [ ]:
```
df.head()
```

Out[ ]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	mov
0	James Cameron	723.0	760505847.0	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	
1	Gore Verbinski	302.0	309404152.0	Action|Adventure|Fantasy	Johnny Depp	Pi Car At
2	Sam Mendes	602.0	200074175.0	Action|Adventure|Thriller	Christoph Waltz	
3	Christopher Nolan	813.0	448130642.0	Action|Thriller	Tom Hardy	T
5	Andrew Stanton	462.0	73058679.0	Action|Adventure|Sci-Fi	Daryl Sabara	John

Appending Meryl streep with Leonardo dicaprio and Brad Pitt and storing it in combined variable

In [ ]:
```
combined = Meryl_Streep.append([Leo_Caprio,Brad_Pitt])
combined.head()
```

```
<ipython-input-47-f2d52e287952>:1: FutureWarning: The frame.append method is deprecated
and will be removed from pandas in a future version. Use pandas.concat instead.
  combined = Meryl_Streep.append([Leo_Caprio,Brad_Pitt])
```

Out[ ]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name
410	Nancy Meyers	187.0	112703470.0	Comedy|Drama|Romance	Meryl Streep
1106	Curtis Hanson	42.0	46815748.0	Action|Adventure|Crime|Thriller	Meryl Streep
1204	Nora Ephron	252.0	94125426.0	Biography|Drama|Romance	Meryl Streep
1408	David Frankel	208.0	124732962.0	Comedy|Drama|Romance	Meryl Streep
1483	Robert Redford	227.0	14998070.0	Drama|Thriller|War	Meryl Streep

In [ ]:
```
df['num_critic_for_reviews'].describe()
```

Out[ ]:
```
count    3852.000000
mean      163.036085
std       123.937734
min         1.000000
25%        72.000000
50%       134.000000
75%       221.000000
max       813.000000
Name: num_critic_for_reviews, dtype: float64
```

Changing the datatype of num_critic_for_reviews with int

In [ ]:
```
combined.num_critic_for_reviews = combined.num_critic_for_reviews.astype(int)
```

Finding the mean after grouping actor with num_critic_for_reviews

In [ ]:
```
combined.groupby(['actor_1_name'])['num_critic_for_reviews'].mean().reset_index()
```

Out[ ]:

	actor_1_name	num_critic_for_reviews
0	Brad Pitt	245.000000
1	Leonardo DiCaprio	330.190476
2	Meryl Streep	181.454545

Changing the datatype of num_user_for_reviews with int

In [ ]:
```
combined['num_user_for_reviews'] = combined['num_user_for_reviews'].astype(int)
```

Finding the mean after grouping actor with num_user_for_reviews

In [ ]:
```
combined.groupby(['actor_1_name'])['num_user_for_reviews'].mean().reset_index()
```

Out[ ]:

	actor_1_name	num_user_for_reviews
0	Brad Pitt	742.352941
1	Leonardo DiCaprio	914.476190
2	Meryl Streep	297.181818

In [ ]:
```
combined.groupby(['actor_1_name'])['num_critic_for_reviews','num_user_for_reviews'].mea
```

```
<ipython-input-53-a9ed58255e46>:1: FutureWarning: Indexing with multiple keys (implicit
ly converted to a tuple of keys) will be deprecated, use a list instead.
  combined.groupby(['actor_1_name'])['num_critic_for_reviews','num_user_for_reviews'].m
ean()
```

Out[ ]:

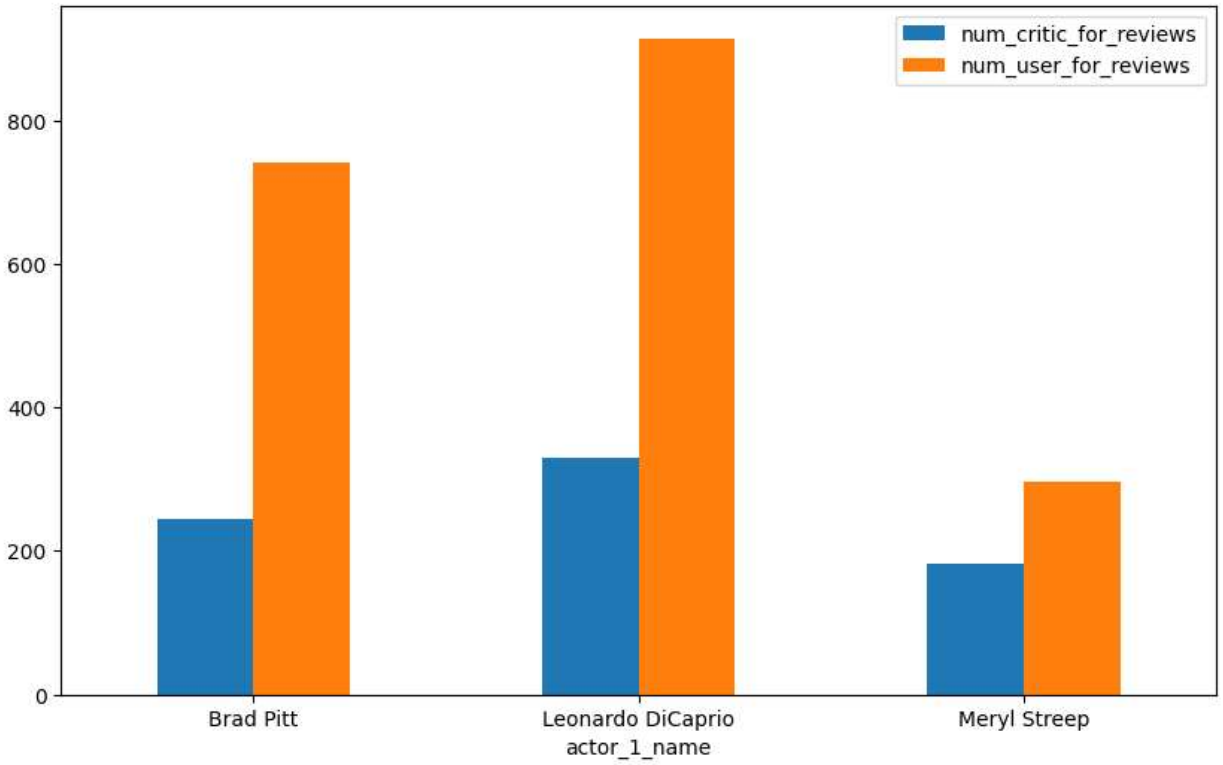
actor_1_name	num_critic_for_reviews	num_user_for_reviews
Brad Pitt	245.000000	742.352941
Leonardo DiCaprio	330.190476	914.476190
Meryl Streep	181.454545	297.181818

In [ ]:
```
combined.groupby(['actor_1_name'])['num_critic_for_reviews','num_user_for_reviews'].mea

plt.xticks(rotation = 360)
```

```
<ipython-input-54-8b413ec37ed1>:1: FutureWarning: Indexing with multiple keys (implicit
ly converted to a tuple of keys) will be deprecated, use a list instead.
  combined.groupby(['actor_1_name'])['num_critic_for_reviews','num_user_for_reviews'].m
ean().plot(kind = 'bar',
```

Out[ ]:
```
(array([0, 1, 2]),
 [Text(0, 0, 'Brad Pitt'),
  Text(1, 0, 'Leonardo DiCaprio'),
  Text(2, 0, 'Meryl Streep')])
```



Observation

Looks like Leonardo DiCaprio has the highest user and critic reviews.

In [ ]:
```
df.head()
```

Out[ ]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	mov
0	James Cameron	723.0	760505847.0	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	
1	Gore Verbinski	302.0	309404152.0	Action|Adventure|Fantasy	Johnny Depp	Pi Car At
2	Sam Mendes	602.0	200074175.0	Action|Adventure|Thriller	Christoph Waltz	
3	Christopher Nolan	813.0	448130642.0	Action|Thriller	Tom Hardy	T
5	Andrew Stanton	462.0	73058679.0	Action|Adventure|Sci-Fi	Daryl Sabara	John

Changing the title_year with int

In [ ]:
```
df['title_year'] = df['title_year'].astype(int)
```

Grouping it with title year after indexing it in nearest to 10ths value

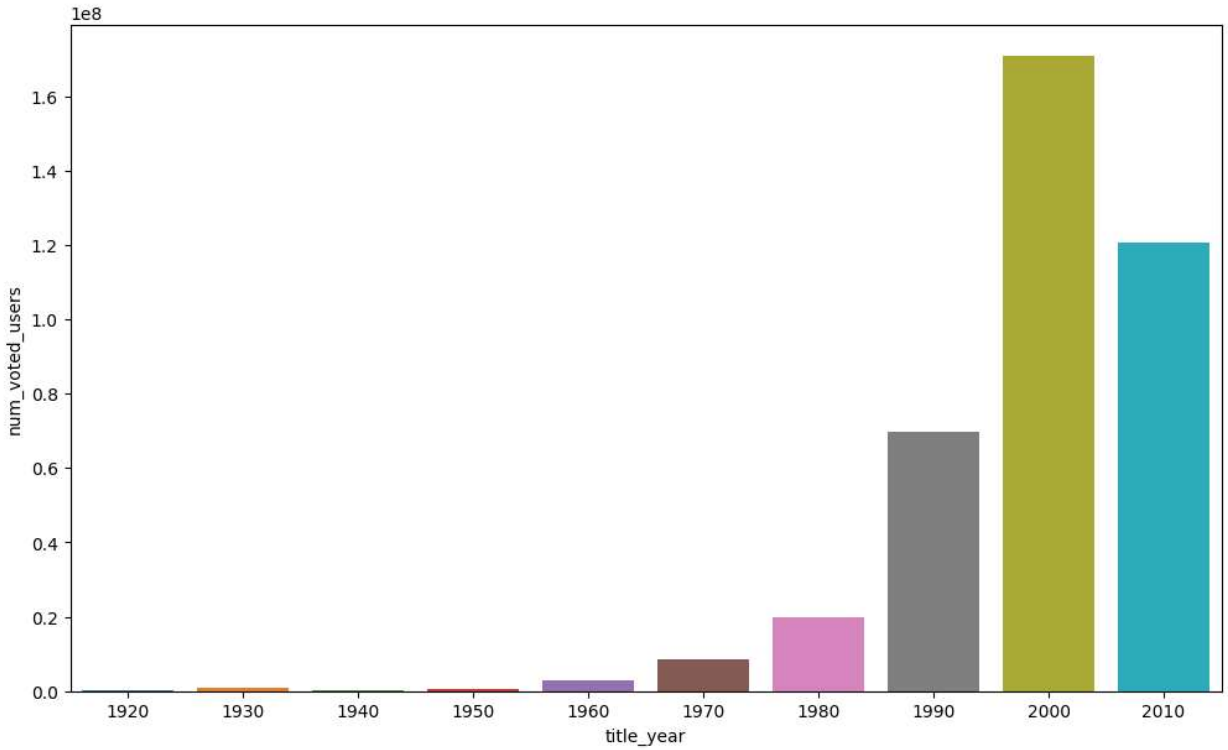
In [ ]:
```
df_num_voted = df.groupby([(df['title_year'] // 10) * 10])['num_voted_users'].sum().res
df_num_voted
```

Out[ ]:

	title_year	num_voted_users
0	1920	116392
1	1930	804839
2	1940	230838
3	1950	678336
4	1960	2983442
5	1970	8524102
6	1980	19987476
7	1990	69735679
8	2000	170908241
9	2010	120640346

In [ ]:
```
plt.figure(figsize = (12,7))
sns.barplot(data = df_num_voted, x = df_num_voted['title_year'], y = df_num_voted['num_
```

Out[ ]:
```
<Axes: xlabel='title_year', ylabel='num_voted_users'>
```



Observation

In 2000 people voted the most followed by 2010

In [ ]: