In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
import io
%cd "C:\Users\coolr\Dropbox\PC\Desktop\IMDBCinema"
```

```
C:\Users\coolr\Dropbox\PC\Desktop\IMDBCinema
```

In [3]:
```
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

```
C:\Users\coolr\anaconda3\lib\site-packages\scipy\__init__.py:155: UserWarnin
g: A NumPy version >=1.18.5 and <1.26.0 is required for this version of SciP
y (detected version 1.26.4
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

Loading the dataset

In [4]:
```
imdb = pd.read_csv("IMDB_Movies.csv")
```

In [5]:
```
imdb.head()
```

Out[5]:

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_faceb
0	Color	James Cameron	723.0	178.0	0.0	
1	Color	Gore Verbinski	302.0	169.0	563.0	
2	Color	Sam Mendes	602.0	148.0	0.0	
3	Color	Christopher Nolan	813.0	164.0	22000.0	
4	NaN	Doug Walker	NaN	NaN	131.0	

5 rows × 28 columns

```
In [6]: imdb.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5043 entries, 0 to 5042
Data columns (total 28 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   color                      5024 non-null   object 
 1   director_name              4939 non-null   object 
 2   num_critic_for_reviews     4993 non-null   float64
 3   duration                   5028 non-null   float64
 4   director_facebook_likes    4939 non-null   float64
 5   actor_3_facebook_likes     5020 non-null   float64
 6   actor_2_name               5030 non-null   object 
 7   actor_1_facebook_likes     5036 non-null   float64
 8   gross                      4159 non-null   float64
 9   genres                     5043 non-null   object 
 10  actor_1_name               5036 non-null   object 
 11  movie_title                5043 non-null   object 
 12  num_voted_users            5043 non-null   int64  
 13  cast_total_facebook_likes  5043 non-null   int64  
 14  actor_3_name               5020 non-null   object 
 15  facenumber_in_poster       5030 non-null   float64
 16  plot_keywords              4890 non-null   object 
 17  movie_imdb_link            5043 non-null   object 
 18  num_user_for_reviews       5023 non-null   object 
 19  language                   5031 non-null   object 
 20  country                    5038 non-null   object 
 21  content_rating             4740 non-null   object 
 22  budget                     4551 non-null   float64
 23  title_year                 4935 non-null   float64
 24  actor_2_facebook_likes     5030 non-null   float64
 25  imdb_score                 5043 non-null   float64
 26  aspect_ratio               4714 non-null   float64
 27  movie_facebook_likes       5043 non-null   int64  
dtypes: float64(12), int64(3), object(13)
memory usage: 1.1+ MB
```

Cleaning the data

```
In [7]: #Viewing the duplicate value and sorting in descending order
        imdb.isnull().sum().sort_values(ascending= False)
```

```
Out[7]: 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
```

```
In [8]: imdb.shape
```

```
Out[8]: (5043, 28)
```

EXTRACTING ONLY THOSE COLUMNS WHICH ARE IMPORTANT

```
In [9]: imdb.columns
```

```
Out[9]: Index(['color', 'director_name', 'num_critic_for_reviews', 'duration',
               'director_facebook_likes', 'actor_3_facebook_likes', 'actor_2_name',
               'actor_1_facebook_likes', 'gross', 'genres', 'actor_1_name',
               'movie_title', 'num_voted_users', 'cast_total_facebook_likes',
               'actor_3_name', 'facenumber_in_poster', 'plot_keywords',
               'movie_imdb_link', 'num_user_for_reviews', 'language', 'country',
               'content_rating', 'budget', 'title_year', 'actor_2_facebook_likes',
               'imdb_score', 'aspect_ratio', 'movie_facebook_likes'],
              dtype='object')
```

```
In [10]: df = imdb[['director_name', 'num_critic_for_reviews','gross', 'genres', 'actor
                'movie_title', 'num_voted_users','num_user_for_reviews','language','buc
                    'imdb_score','movie_facebook_likes']]
```

Data Description

1.director_name: Name of the director who directed the movie. 2.num_critic_for_reviews:Critic Review 3.gross:Total Revenue generated 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.language:Language of the movie 9.budget: 10.title_year: 11.imdb_score: Score obtained by the movie 12.movie_facebook_likes: Total facebook likes

In [11]:
```
df.head()
```

Out[11]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name
0	James Cameron	723.0	760505847.0	Action|Adventure|Fantasy|Sci-Fi	CCH Pounde
1	Gore Verbinski	302.0	309404152.0	Action|Adventure|Fantasy	Johnny Depp
2	Sam Mendes	602.0	200074175.0	Action|Adventure|Thriller	Christoph Waltz
3	Christopher Nolan	813.0	448130642.0	Action|Thriller	Tom Hardy
4	Doug Walker	NaN	NaN	Documentary	Doug Walke

In [12]:
```
df.shape
```

Out[12]:
```
(5043, 13)
```

In [13]:
```
df.isnull().sum().sort_values(ascending= False) # This is for column null valu
```

Out[13]:
```
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 [14]:
```
df.isnull().sum(axis = 1).sort_values(ascending= False)
#This is for row null value
```

Out[14]:
```
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 which are not null gross & budget columns

In [15]:
```
df = df[df['gross'].notna()]
df = df[df['budget'].notna()]
```

In [16]:
```
df.isnull().sum().sort_values(ascending = False)
```

Out[16]:
```
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 [17]:
```
# Getting only null values in actor column
df[df['actor_1_name'].isnull()]
```

Out[17]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	n
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 [18]:
```
df['language'].value_counts().iloc[0:5]
```

Out[18]:
```
English     3707
French        37
Spanish       26
Mandarin      15
German        13
Name: language, dtype: int64
```

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

In [19]:
```
df['language'].replace(np.nan,'English', inplace = True )
```

In [20]:
```
df.isnull().sum().sort_values(ascending = False)
```

Out[20]:
```
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 [21]:
```
df[df['num_critic_for_reviews'].isnull()]
```

Out[21]:

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

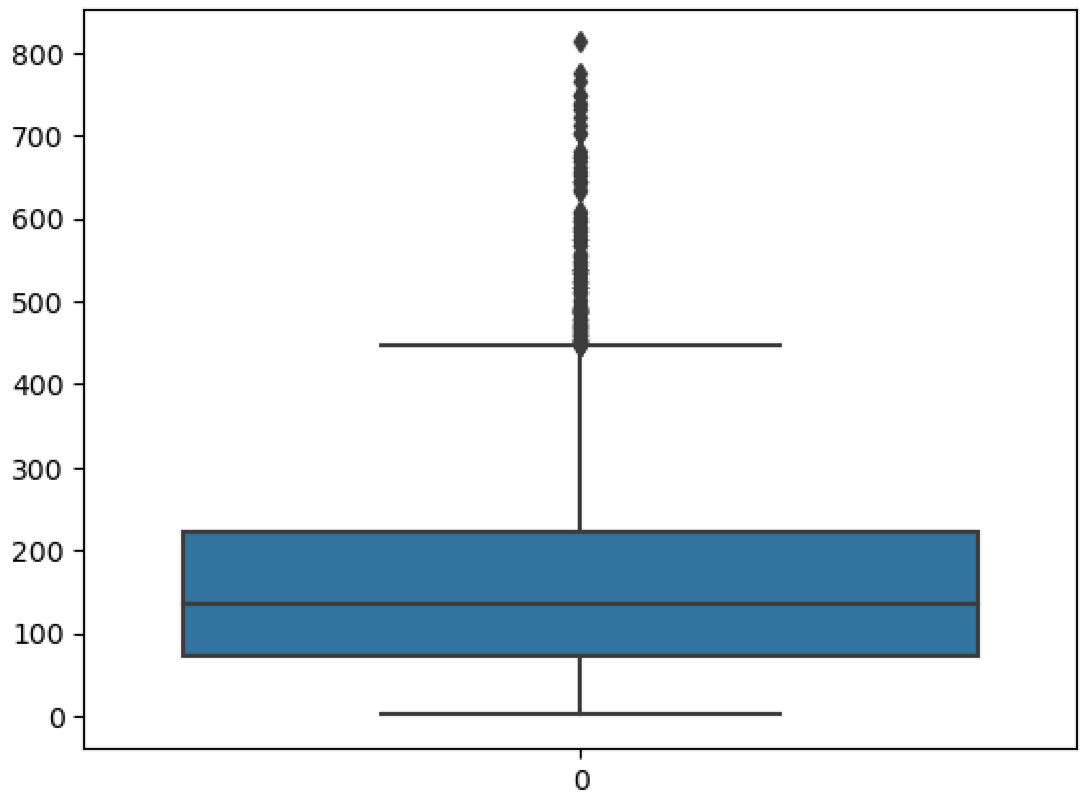
In [22]:
```
df['num_critic_for_reviews'].describe()
```

Out[22]:
```
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 [23]: sns.boxplot(df['num_critic_for_reviews'])
```

```
Out[23]: <AxesSubplot: >
```



```
In [24]: df = df.dropna()
```

```
In [25]: df.shape
```

```
Out[25]: (3887, 13)
```

```
In [26]: df.isnull().sum()
```

```
Out[26]: 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
```

Finding the duplicated values

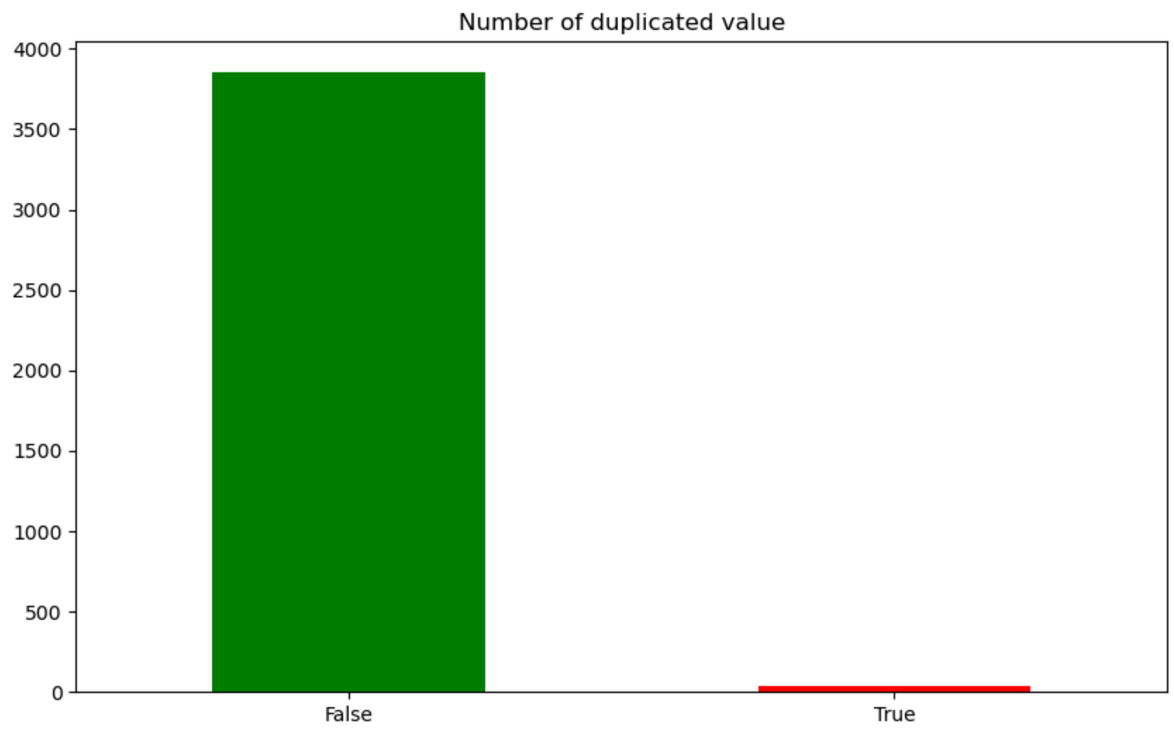
```
In [27]: dup = df.duplicated().value_counts()
```

```
In [28]: dup
```

```
Out[28]: False    3852
         True       35
         dtype: int64
```

```
In [29]: plt.figure(figsize= (10,6))
         dup.plot(kind = 'bar', color = ['g','r'])
         plt.xticks(rotation =360)
         plt.title("Number of duplicated value")
```

```
Out[29]: Text(0.5, 1.0, 'Number of duplicated value')
```



Dropping the duplicated values

```
In [30]: df = df.drop_duplicates()
         df.shape
```

```
Out[30]: (3852, 13)
```

In [31]:
```
df.describe().style.background_gradient()
```

Out[31]:

	num_critic_for_reviews	gross	num_voted_users	budget	title_
count	3852.000000	3852.000000	3852.000000	3852.000000	3852.00
mean	163.036085	50975542.371236	102442.806594	45253902.591121	2003.06
std	123.937734	69326510.589619	150309.201024	223449575.930116	10.01
min	1.000000	162.000000	5.000000	218.000000	1920.00
25%	72.000000	6815231.500000	17308.500000	10000000.000000	1999.00
50%	134.000000	27900000.000000	50588.000000	24000000.000000	2005.00
75%	221.000000	65508766.750000	124194.250000	50000000.000000	2010.00
max	813.000000	760505847.000000	1689764.000000	12215500000.000000	2016.00

In [32]:
```
df.isnull().sum()
```

Out[32]:
```
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
```

The data is clean and is 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 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?

lets create a column revenue

In [33]:
```
df.head()
```

Out[33]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name
0	James Cameron	723.0	760505847.0	Action|Adventure|Fantasy|Sci-Fi	CCH Pounde
1	Gore Verbinski	302.0	309404152.0	Action|Adventure|Fantasy	Johnny Dep
2	Sam Mendes	602.0	200074175.0	Action|Adventure|Thriller	Christoph Walt
3	Christopher Nolan	813.0	448130642.0	Action|Thriller	Tom Hard
5	Andrew Stanton	462.0	73058679.0	Action|Adventure|Sci-Fi	Daryl Sabar

In [34]:
```
df['profit'] = df['gross'] - df['budget']
```

In [35]:
```
df.head()
```

Out[35]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name
0	James Cameron	723.0	760505847.0	Action|Adventure|Fantasy|Sci-Fi	CCH Pounde
1	Gore Verbinski	302.0	309404152.0	Action|Adventure|Fantasy	Johnny Dep
2	Sam Mendes	602.0	200074175.0	Action|Adventure|Thriller	Christoph Walt
3	Christopher Nolan	813.0	448130642.0	Action|Thriller	Tom Hard
5	Andrew Stanton	462.0	73058679.0	Action|Adventure|Sci-Fi	Daryl Sabar

Sorting the profit column in decending order

In [36]:
```
top_profitable_movie = df.sort_values(['profit'],axis = 0,ascending=False)
top_profitable_movie.head()
```

Out[36]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_n
0	James Cameron	723.0	760505847.0	Action|Adventure|Fantasy|Sci-Fi	CCH Pou
29	Colin Trevorrow	644.0	652177271.0	Action|Adventure|Sci-Fi|Thriller	Bryce D Hov
26	James Cameron	315.0	658672302.0	Drama|Romance	Leon DiCa
3024	George Lucas	282.0	460935665.0	Action|Adventure|Fantasy|Sci-Fi	Harrison
3080	Steven Spielberg	215.0	434949459.0	Family|Sci-Fi	Henry Tho

In [37]:
```
# Getting top 10 values
top_10_profit = top_profitable_movie.iloc[:10]
top_10_profit[['movie_title','profit']]
```

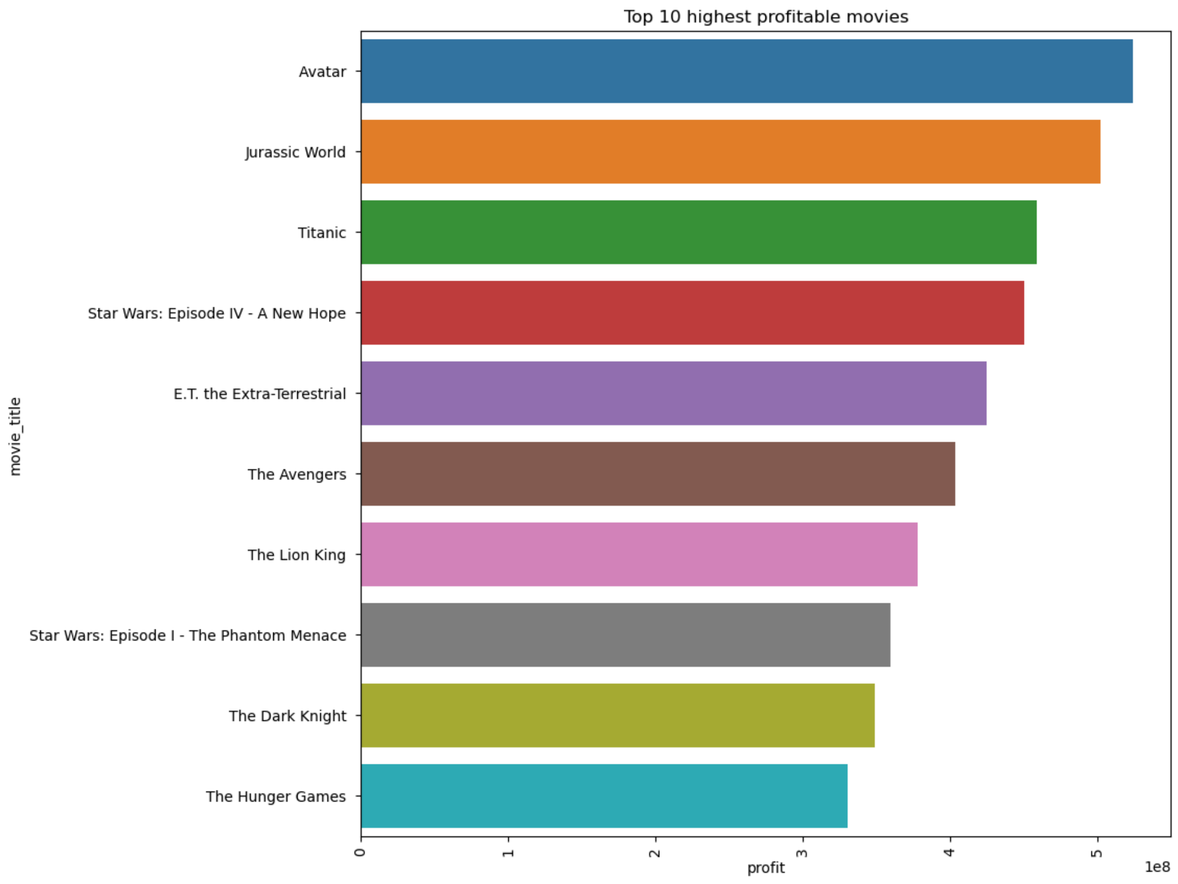
Out[37]:

	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 [38]:
```
top_10_profit.keys()
```

Out[38]:
```
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 [39]: plt.figure(figsize= (10,10))
         sns.barplot(data = df, y = top_10_profit['movie_title'], x = top_10_profit['pr
         plt.xticks(rotation = 90)
         plt.title("Top 10 highest profitable movies")
         plt.show()
```



Observations

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!

Your task: Find IMDB Top 250

In [40]:
```
#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[40]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name
0	James Cameron	723.0	760505847.0	Action|Adventure|Fantasy|Sci-Fi	CCH Pounde
1	Gore Verbinski	302.0	309404152.0	Action|Adventure|Fantasy	Johnny Dep
2	Sam Mendes	602.0	200074175.0	Action|Adventure|Thriller	Christoph Walt
3	Christopher Nolan	813.0	448130642.0	Action|Thriller	Tom Hard
5	Andrew Stanton	462.0	73058679.0	Action|Adventure|Sci-Fi	Daryl Sabar

In [41]:
```
IMDB_Top_250.tail()
```

Out[41]:

	director_name	num_critic_for_reviews	gross	genres	acto
4977	Morgan Spurlock	193.0	11529368.0	Comedy|Documentary|Drama	
5008	Kevin Smith	136.0	3151130.0	Comedy	Jas
5012	David Ayer	233.0	10499968.0	Action|Crime|Drama|Thriller	M
5033	Shane Carruth	143.0	424760.0	Drama|Sci-Fi|Thriller	Sha
5035	Robert Rodriguez	56.0	2040920.0	Action|Crime|Drama|Romance|Thriller	

```
In [42]: # 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[42]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_nam
1937	Frank Darabont	199.0	28341469.0	Crime|Drama	Morga Freema
3466	Francis Ford Coppola	208.0	134821952.0	Crime|Drama	Al Pacir
2837	Francis Ford Coppola	149.0	57300000.0	Crime|Drama	Robert De Ni
66	Christopher Nolan	645.0	533316061.0	Action|Crime|Drama|Thriller	Christian Ba
4498	Sergio Leone	181.0	6100000.0	Western	Clint Eastwoc

```
In [43]: IMDB_Top_250.shape
```

```
Out[43]: (2609, 14)
```

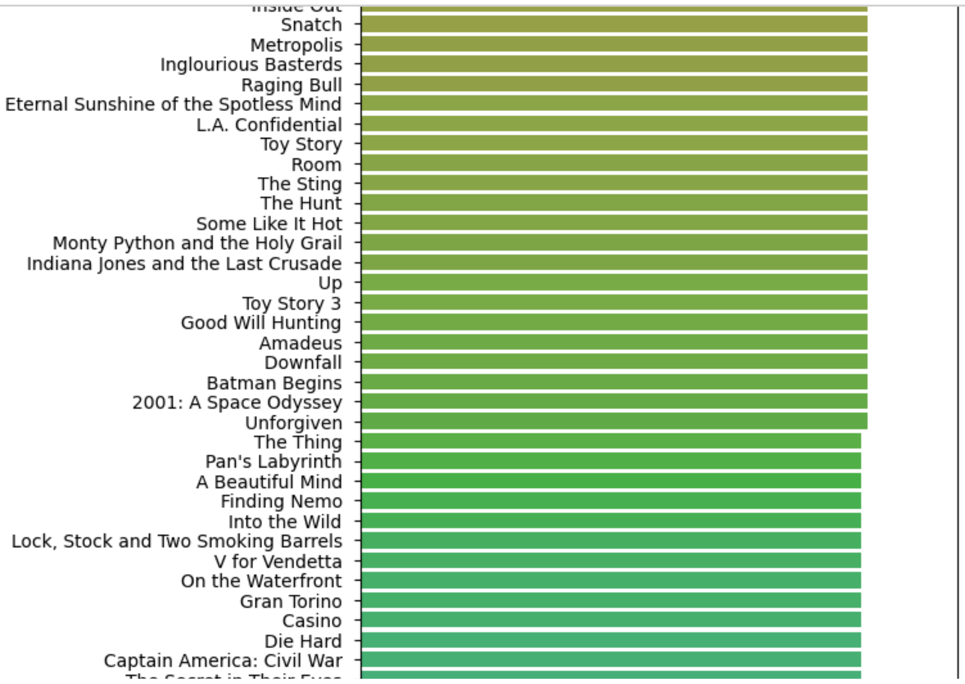
```
In [44]: IMDB_Top_250 = IMDB_Top_250.iloc[:250]
         pd.set_option('display.max_rows',500)
         IMDB_Top_250[['movie_title','imdb_score']]
```

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

```
In [45]: IMDB_Top_250.groupby(['imdb_score'])['movie_title'].value_counts().iloc[:250]
```

```
Out[45]: 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
```

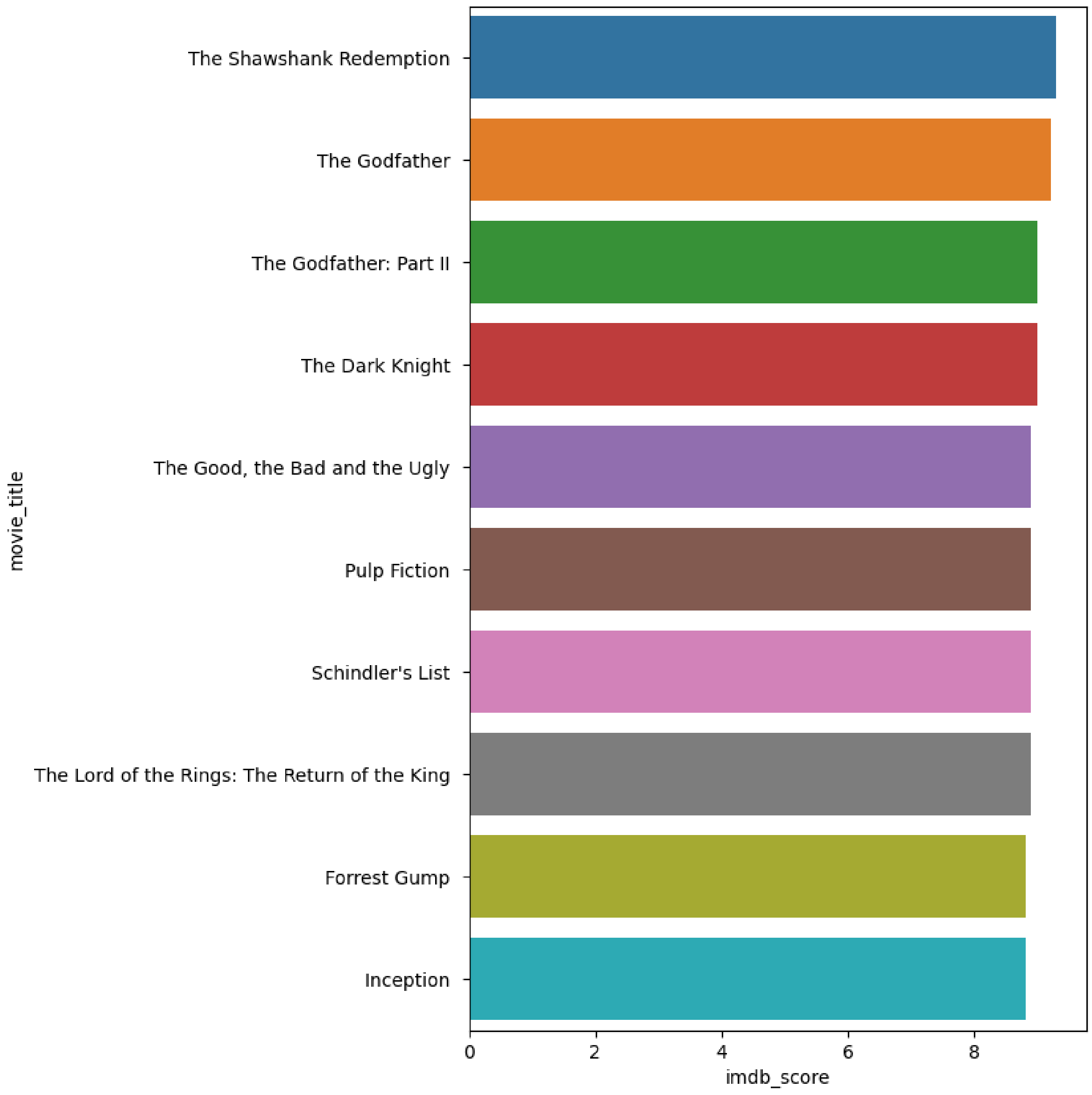
```
In [46]: plt.figure(figsize = (6,50))
         sns.barplot(data = IMDB_Top_250, y = IMDB_Top_250['movie_title'],
                     x = IMDB_Top_250['imdb_score'])
```



```
In [47]: # Top 10 profit movies

         plt.figure(figsize = (6,10))
         sns.barplot(data = IMDB_Top_250, y = IMDB_Top_250['movie_title'].iloc[:10],
                     x = IMDB_Top_250['imdb_score'].iloc[:10])

Out[47]: <AxesSubplot: xlabel='imdb_score', ylabel='movie_title'>
```



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!

In [48]:
```
non_english = IMDB_Top_250[IMDB_Top_250['language'] != 'English']
non_english.head()
```

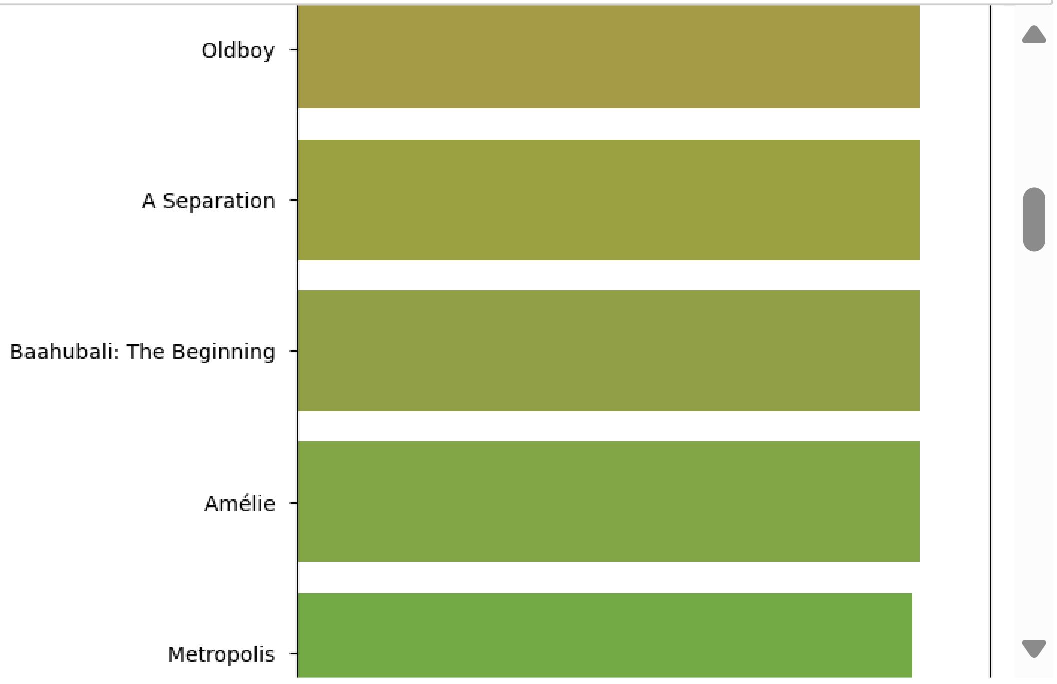
Out[48]:

	director_name	num_critic_for_reviews	gross	genres	actor
4498	Sergio Leone	181.0	6100000.0	Western	Clint
4747	Akira Kurosawa	153.0	269061.0	Action|Adventure|Drama	
4029	Fernando Meirelles	214.0	7563397.0	Crime|Drama	Al
2373	Hayao Miyazaki	246.0	10049886.0	Adventure|Animation|Family|Fantasy	
4259	Florian Henckel von Donnersmarck	215.0	11284657.0	Drama|Thriller	

In [49]:
```
non_english.shape
```

Out[49]:
```
(38, 14)
```

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



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.

Your task: Find the best directors

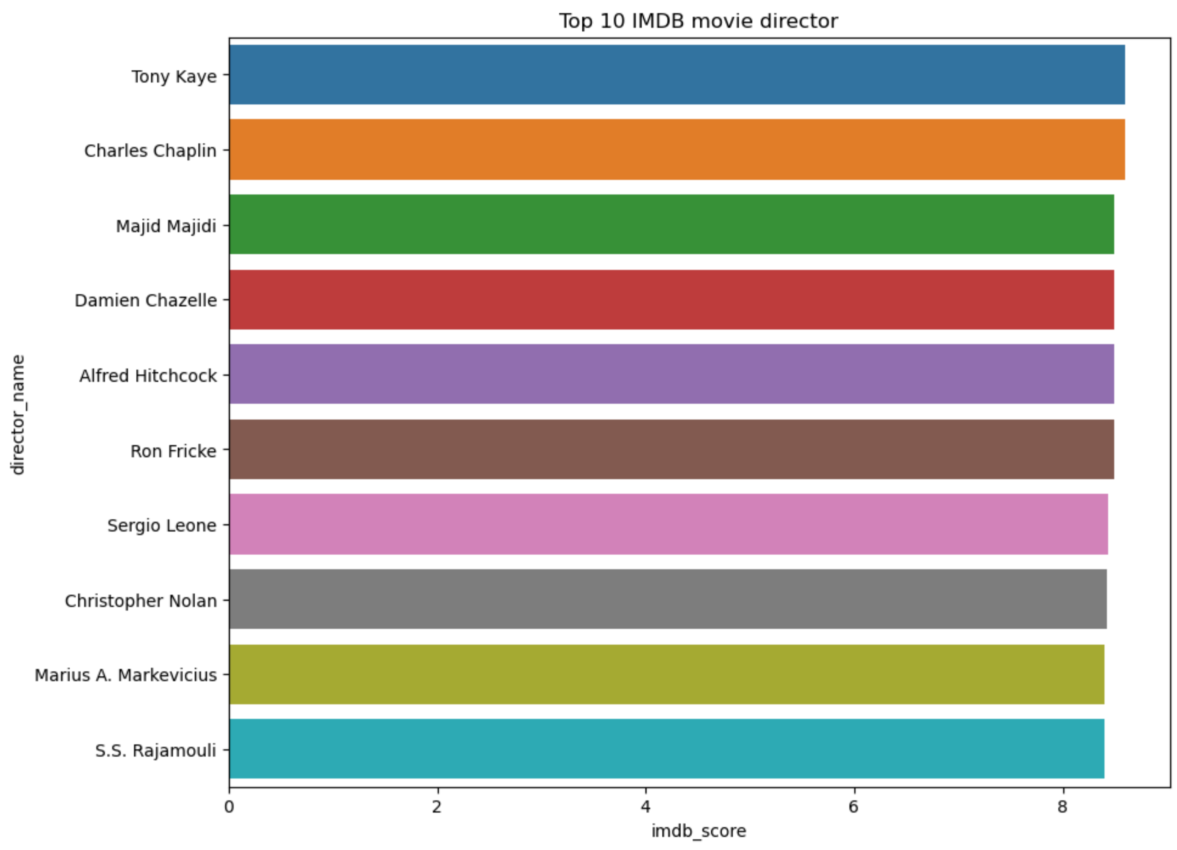
In [51]:
```
top_10_director = df.groupby(['director_name'])['imdb_score'].mean().reset_inc
top_10_director = top_10_director.sort_values(['imdb_score'],axis = 0,
                                               ascending = False)
top_10_director.head(10)
```

Out[51]:

	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 [52]:
```
plt.figure(figsize = (10,8))
sns.barplot(data = top_10_director , x = top_10_director['imdb_score'].iloc[:1
            y = top_10_director['director_name'].iloc[:10])
plt.title("Top 10 IMDB movie director")
```

Out[52]:
```
Text(0.5, 1.0, 'Top 10 IMDB movie director')
```



Observation

tony kane is the director with highest imdb score

E.Popular Genres: Perform this step using the knowledge gained while performing previous steps. Your task: Find popular genres

In [53]:
```
IMDB_Top_250
```

Out[53]:

	director_name	num_critic_for_reviews	gross	
1937	Frank Darabont	199.0	28341469.0	
3466	Francis Ford Coppola	208.0	134821952.0	
2837	Francis Ford Coppola	149.0	57300000.0	
66	Christopher Nolan	645.0	533316061.0	Action|Crime
4498	Sergio Leone	181.0	6100000.0	
3355	Quentin Tarantino	215.0	107930000.0	

Sorting the top Genres with respect to IMDB score

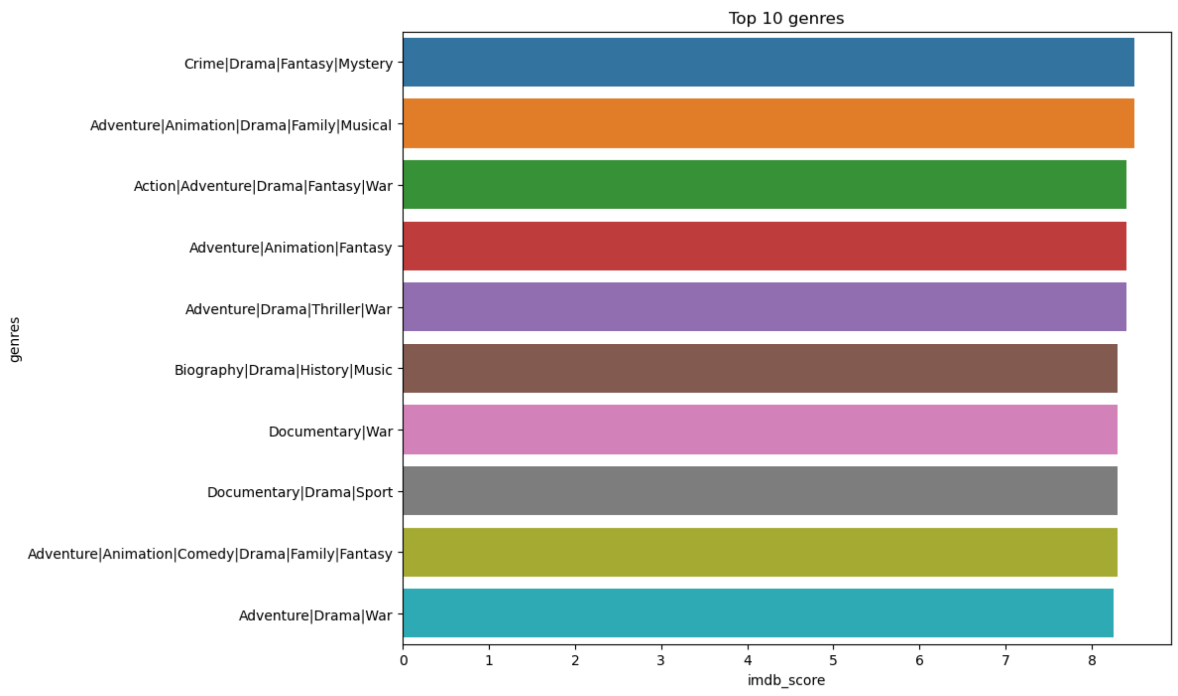
In [54]:
```
popular_genres = df.groupby(['genres'])['imdb_score'].mean().reset_index()
popular_genres.sort_values(['imdb_score'],axis = 0, ascending = False,
                           inplace = True)
popular_genres.head(15)
```

Out[54]:

	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 [55]:
```
plt.figure(figsize = (10,8))
sns.barplot(data = popular_genres , x = popular_genres['imdb_score'].iloc[:10]
            y = popular_genres['genres'].iloc[:10])
plt.title("Top 10 genres")
```

Out[55]:
```
Text(0.5, 1.0, 'Top 10 genres')
```



Observation

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.

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

```
In [56]: df.head()
```

Out[56]:

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

Creating column with Meryl streep,Leonardo Dicaprio and Brad Pitt

```
In [57]: 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 [58]: df.head()
```

Out[58]:

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

Appending Meryl Streep with Leanardo dicaprio and Brad Pitt and storing it in combined variable

```
In [59]: combined = Meryl_Streep.append([Leo_Caprio,Brad_Pitt])
         combined.head()
```

Out[59]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_
410	Nancy Meyers	187.0	112703470.0	Comedy|Drama|Romance	Meryl S
1106	Curtis Hanson	42.0	46815748.0	Action|Adventure|Crime|Thriller	Meryl S
1204	Nora Ephron	252.0	94125426.0	Biography|Drama|Romance	Meryl S
1408	David Frankel	208.0	124732962.0	Comedy|Drama|Romance	Meryl S
1483	Robert Redford	227.0	14998070.0	Drama|Thriller|War	Meryl S

```
In [60]: combined['actor_1_name'].unique()

Out[60]: array(['Meryl Streep', 'Leonardo DiCaprio', 'Brad Pitt'], dtype=object)
```

```
In [61]: df['num_critic_for_reviews'].describe()
```

```
Out[61]: 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 [62]: combined.num_critic_for_reviews = combined.num_critic_for_reviews.astype(int)
```

Finding the mean after grouping actor with num_critic_for_reviews

```
In [63]: combined.groupby(['actor_1_name'])['num_critic_for_reviews'].mean().reset_inde
```

Out[63]:

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

```
In [64]: combined.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 49 entries, 410 to 2898
Data columns (total 14 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   director_name           49 non-null     object 
 1   num_critic_for_reviews  49 non-null     int32  
 2   gross                   49 non-null     float64
 3   genres                  49 non-null     object 
 4   actor_1_name            49 non-null     object 
 5   movie_title             49 non-null     object 
 6   num_voted_users         49 non-null     int64  
 7   num_user_for_reviews    49 non-null     object 
 8   language                49 non-null     object 
 9   budget                  49 non-null     float64
 10  title_year              49 non-null     float64
 11  imdb_score              49 non-null     float64
 12  movie_facebook_likes    49 non-null     int64  
 13  profit                  49 non-null     float64
```

Changing the datatype of num_user_for_reviews with int

```
In [65]: combined['num_user_for_reviews'] = combined['num_user_for_reviews'].astype(int
```

Finding the mean after grouping actor with num_user_for_reviews

```
In [66]: combined.groupby(['actor_1_name'])['num_user_for_reviews'].mean().reset_index(
```

```
Out[66]:
```

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

```
In [67]: combined.groupby(['actor_1_name'])['num_critic_for_reviews','num_user_for_revi
```

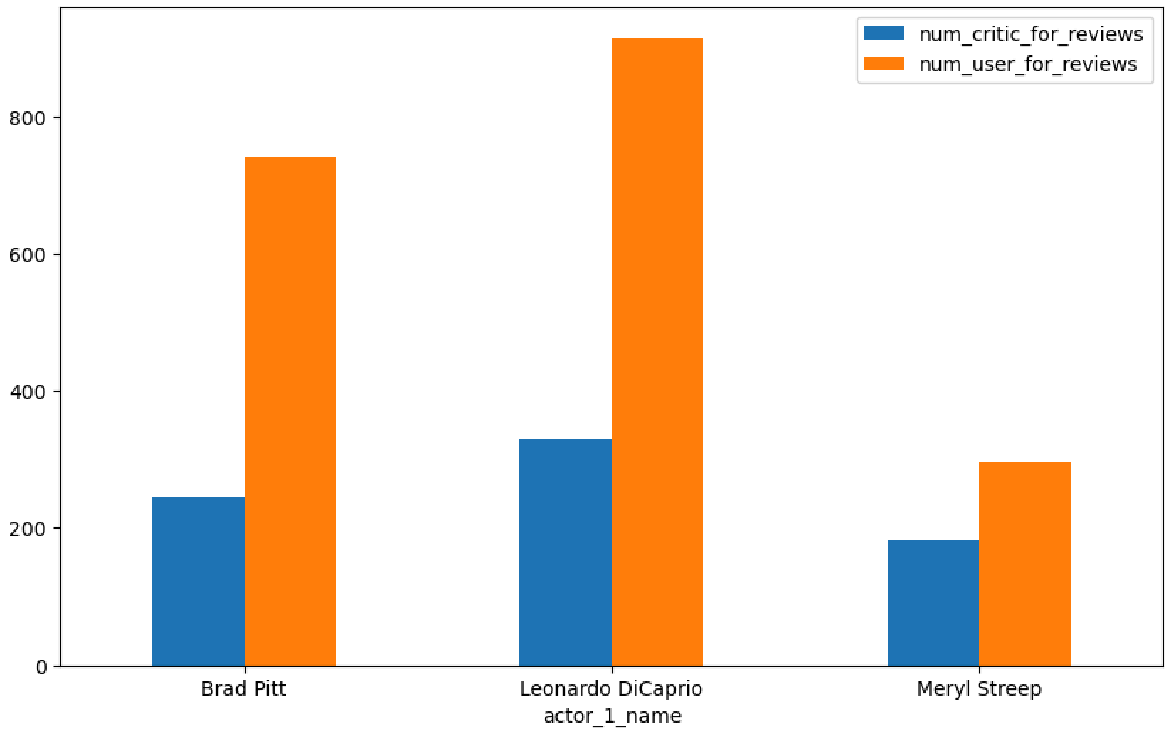
```
Out[67]:
```

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

In [68]:
```
combined.groupby(['actor_1_name'])['num_critic_for_reviews','num_user_for_revi

plt.xticks(rotation = 360)
```

Out[68]:
```
(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 [69]:
```
df.head()
```

Out[69]:

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

Changing the title_year with int

```
In [70]: df['title_year'] = df['title_year'].astype(int)
```

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

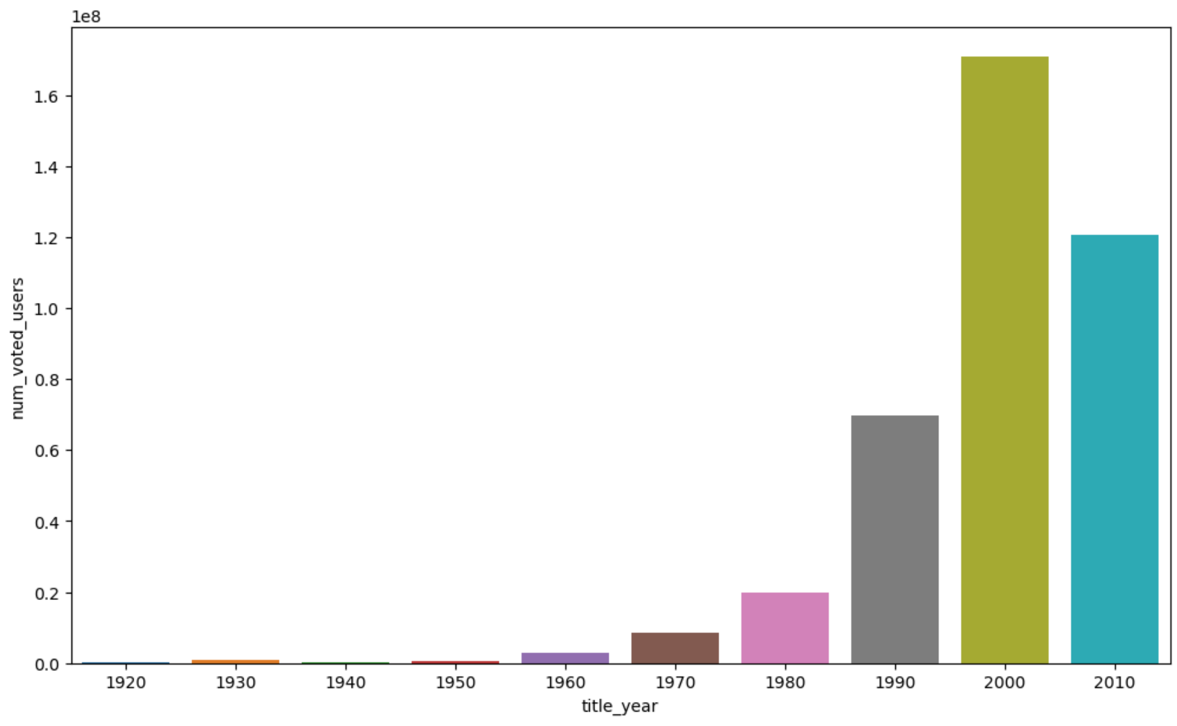
```
In [71]: df_num_voted = df.groupby([(df['title_year'] // 10) * 10])['num_voted_users'].
         df_num_voted
```

Out[71]:

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

```
Out[72]: <AxesSubplot: xlabel='title_year', ylabel='num_voted_users'>
```



Observation

In 2000 people voted the most followed by 2010