```
In [3]:  #importing the libraries
          import numpy as np
          import pandas as pd
          import matplotlib.pyplot as plt
          import seaborn as sns
```

```
In [55]:  #1) Import the data and read
          movies= pd.read_csv("IMDB_Movies.csv")
          origdata = movies
          movies
```

Out[55]:

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes
0	Color	James Cameron	723.0	178.0	0.0	855.0
1	Color	Gore Verbinski	302.0	169.0	563.0	1000.0
2	Color	Sam Mendes	602.0	148.0	0.0	161.0
3	Color	Christopher Nolan	813.0	164.0	22000.0	23000.0
4	NaN	Doug Walker	NaN	NaN	131.0	NaN
...	...	...	...	...	...	...
5038	Color	Scott Smith	1.0	87.0	2.0	318.0
5039	Color	NaN	43.0	43.0	NaN	319.0
5040	Color	Benjamin Roberds	13.0	76.0	0.0	0.0
5041	Color	Daniel Hsia	14.0	100.0	0.0	489.0
5042	Color	Jon Gunn	43.0	90.0	16.0	16.0

5043 rows × 28 columns

```
In [56]:  #1.1) Inspect the dataframe's Columns, Variables type etc
          movies.shape
```

```
Out[56]: (5043, 28)
```

In [7]:
```
movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5043 entries, 0 to 5042
Data columns (total 28 columns):
color                        5024 non-null object
director_name                4939 non-null object
num_critic_for_reviews       4993 non-null float64
duration                     5028 non-null float64
director_facebook_likes      4939 non-null float64
actor_3_facebook_likes       5020 non-null float64
actor_2_name                 5030 non-null object
actor_1_facebook_likes       5036 non-null float64
gross                        4159 non-null float64
genres                       5043 non-null object
actor_1_name                 5036 non-null object
movie_title                  5043 non-null object
num_voted_users              5043 non-null int64
cast_total_facebook_likes    5043 non-null int64
actor_3_name                 5020 non-null object
facenumber_in_poster         5030 non-null float64
plot_keywords                4890 non-null object
movie_imdb_link              5043 non-null object
num_user_for_reviews         5023 non-null object
language                     5031 non-null object
country                      5038 non-null object
content_rating               4740 non-null object
budget                       4551 non-null float64
title_year                   4935 non-null float64
actor_2_facebook_likes       5030 non-null float64
imdb_score                   5043 non-null float64
aspect_ratio                 4714 non-null float64
movie_facebook_likes         5043 non-null int64
dtypes: float64(12), int64(3), object(13)
memory usage: 1.1+ MB
```

In [ ]:
```
# Cleaning the data
#1.2) Inspect NULL Values
```

In [6]:
```
#For Column wise NULL Count
movies.isnull().sum(axis=0).sort_values(ascending=False)
```

Out[6]:
```
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
movie_facebook_likes           0
genres                         0
movie_title                    0
num_voted_users                0
movie_imdb_link                0
imdb_score                     0
cast_total_facebook_likes      0
dtype: int64
```

In [57]:
```
#For row wise Null Values count
movies.isnull().sum(axis=1).sort_values(ascending=False)
```

Out[57]:
```
279     15
4       13
4945    11
2241    11
2342    10
        ..
2708     0
2707     0
2706     0
2705     0
0        0
Length: 5043, dtype: int64
```

In [50]:
```
#For Column wise Null percentages
movies.isnull().sum(axis=0).sort_values(ascending=False)/len(movies)*100
```

Out[50]:
```
gross                        17.529248
budget                        9.756098
aspect_ratio                  6.523895
content_rating                6.008328
plot_keywords                 3.033908
title_year                    2.141582
director_name                 2.062265
director_facebook_likes       2.062265
num_critic_for_reviews        0.991473
actor_3_name                  0.456078
actor_3_facebook_likes        0.456078
num_user_for_reviews          0.396589
color                         0.376760
duration                      0.297442
facenumber_in_poster          0.257783
actor_2_name                  0.257783
actor_2_facebook_likes        0.257783
language                      0.237954
actor_1_name                  0.138806
actor_1_facebook_likes        0.138806
country                       0.099147
movie_facebook_likes          0.000000
genres                        0.000000
movie_title                   0.000000
num_voted_users               0.000000
movie_imdb_link               0.000000
imdb_score                    0.000000
cast_total_facebook_likes     0.000000
dtype: float64
```

In [ ]:
```
#1.3) Drop Unnecessary Columns, In this assignment we are analysing the movies with resp
ect to ratings,gross collection, popularity of the movis wtc
#so many of the columns in this dataframe not required. we can drop those columns
```

In [58]:
```
movies = movies.drop([
    'color',
    'director_facebook_likes',
    'actor_1_facebook_likes',
    'actor_2_facebook_likes',
    'actor_3_facebook_likes',
    'actor_2_name',
    'cast_total_facebook_likes',
    'actor_3_name',
    'duration',
    'facenumber_in_poster',
    'content_rating',
    'country',
    'movie_imdb_link',
    'aspect_ratio',
    'plot_keywords'],axis=1)
```

```
In [9]: movies
```

Out[9]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_ti
0	James Cameron	723.0	760505847.0	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	Ava
1	Gore Verbinski	302.0	309404152.0	Action|Adventure|Fantasy	Johnny Depp	Pirates t Caribbe At Worl E
2	Sam Mendes	602.0	200074175.0	Action|Adventure|Thriller	Christoph Waltz	Spec
3	Christopher Nolan	813.0	448130642.0	Action|Thriller	Tom Hardy	The Da Knig Ris
4	Doug Walker	NaN	NaN	Documentary	Doug Walker	Star Wa Episode - The For Awake
...	...	...	...	...	...	
5038	Scott Smith	1.0	NaN	Comedy|Drama	Eric Mabius	Sigr Sea Delivei
5039	NaN	43.0	NaN	Crime|Drama|Mystery|Thriller	Natalie Zea	T Follow
5040	Benjamin Roberds	13.0	NaN	Drama|Horror|Thriller	Eva Boehnke	A Plag Pleas
5041	Daniel Hsia	14.0	10443.0	Comedy|Drama|Romance	Alan Ruck	Shang Call
5042	Jon Gunn	43.0	85222.0	Documentary	John August	My Da with Dr

5043 rows × 13 columns

```
In [ ]: #1.4) Drop Unnecessary rows using columns with high Null percentages
        # Now,we might notice that some columns have larger percentage(greater than 5%) of Null
        values.Drop all the such rows which have Null values
```

```
In [10]: round(movies.isnull().sum().sort_values(ascending=False)/len(movies)*100,2)
```

```
Out[10]: gross                     17.53
         budget                     9.76
         title_year                 2.14
         director_name              2.06
         num_critic_for_reviews     0.99
         num_user_for_reviews       0.40
         language                   0.24
         actor_1_name               0.14
         movie_facebook_likes       0.00
         imdb_score                 0.00
         num_voted_users            0.00
         movie_title                0.00
         genres                     0.00
         dtype: float64
```

```
In [59]: movies=movies[movies['gross'].notnull()]
```

```
In [60]: movies=movies[movies['budget'].notnull()]
```

```
In [61]: round(movies.isnull().sum().sort_values(ascending=False)/len(movies)*100,2)
```

```
Out[61]: language                  0.08
         actor_1_name              0.08
         num_critic_for_reviews    0.03
         movie_facebook_likes      0.00
         imdb_score                0.00
         title_year                0.00
         budget                    0.00
         num_user_for_reviews      0.00
         num_voted_users           0.00
         movie_title               0.00
         genres                    0.00
         gross                     0.00
         director_name             0.00
         dtype: float64
```

```
In [ ]: #1.5) Drop Unnecessary rows,some of the rows might have greater than 5 Nan values.
        #such rows aren't of much use for the analysis and hence should be removed
```

```
In [62]: (movies.isnull().sum(axis=1).sort_values(ascending=False) >5).sum()
```

```
Out[62]: 0
```

In [63]:
```
movies=movies[movies.isnull().sum(axis=1).sort_values(ascending=False) <=5]
movies
```

```
C:\Users\Parashu\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: UserWarning: Bool
ean Series key will be reindexed to match DataFrame index.
  """Entry point for launching an IPython kernel.
```

Out[63]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	n
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	(
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	J
...	...	...	...	...	...	
5033	Shane Carruth	143.0	424760.0	Drama|Sci-Fi|Thriller	Shane Carruth	
5034	Neill Dela Llana	35.0	70071.0	Thriller	Ian Gamazon	
5035	Robert Rodriguez	56.0	2040920.0	Action|Crime|Drama|Romance|Thriller	Carlos Gallardo	E
5037	Edward Burns	14.0	4584.0	Comedy|Drama	Kerry Bishé	N
5042	Jon Gunn	43.0	85222.0	Documentary	John August	

3891 rows × 13 columns

In [ ]:
```
#1.6) Fill Nan Values
#you might notice that the language column has some Nan values, Here on inspection we no
tice that we can replace with English
```

```
In [64]: round(movies.isnull().sum().sort_values(ascending=False)/len(movies)*100,2)
```

```
Out[64]: language                 0.08
         actor_1_name             0.08
         num_critic_for_reviews   0.03
         movie_facebook_likes     0.00
         imdb_score               0.00
         title_year               0.00
         budget                   0.00
         num_user_for_reviews     0.00
         num_voted_users          0.00
         movie_title              0.00
         genres                   0.00
         gross                    0.00
         director_name            0.00
         dtype: float64
```

```
In [59]: #Why to replace with English only
         movies.groupby('language').language.count().sort_values(ascending=False)
```

```
Out[59]: language
         English       3707
         French          37
         Spanish         26
         Mandarin        15
         German          13
         Japanese        12
         Hindi           10
         Cantonese        8
         Italian          7
         Korean           5
         Portuguese       5
         Norwegian        4
         Hebrew           3
         Persian          3
         Dutch            3
         Danish           3
         Thai             3
         Dari             2
         Indonesian       2
         Aboriginal       2
         Icelandic        1
         Hungarian        1
         Arabic           1
         Aramaic          1
         Bosnian          1
         Telugu           1
         Czech            1
         Swedish          1
         Russian          1
         Romanian         1
         Dzongkha         1
         None             1
         Filipino         1
         Mongolian        1
         Maya             1
         Kazakh           1
         Vietnamese       1
         Zulu             1
         Name: language, dtype: int64
```

```
In [17]: movies.language.describe()
```

```
Out[17]: count        3888
         unique         38
         top       English
         freq         3707
         Name: language, dtype: object
```

```
In [18]: movies.language=movies.language.fillna('English')
```

In [65]:
```
round(movies.isnull().sum().sort_values(ascending=False)/len(movies)*100,2)
```

Out[65]:
```
language                  0.08
actor_1_name              0.08
num_critic_for_reviews    0.03
movie_facebook_likes      0.00
imdb_score                0.00
title_year                0.00
budget                    0.00
num_user_for_reviews      0.00
num_voted_users           0.00
movie_title               0.00
genres                    0.00
gross                     0.00
director_name             0.00
dtype: float64
```

In [20]:
```
#1.5) Check the number of retained rows
# there still 2 columns have misssing data viz 1) actor_1_nmae and 2) num_critic_for_rev
iews have small percentage of Nan values left
# as of now we can keep like that # We still have 77% of rows are present
len(movies)/len(origdata)*100
```

Out[20]:
```
77.15645449137418
```

In [125]:
```
#2) Data Analysis
#2.1) Change the unit of budget and gross columns from $ to million$
movies['budget']=movies['budget']/1000000
movies['gross']=movies['gross']/1000000
```

```
In [22]: movies
```

Out[22]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	m
0	James Cameron	723.0	760.505847	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	
1	Gore Verbinski	302.0	309.404152	Action|Adventure|Fantasy	Johnny Depp	C A
2	Sam Mendes	602.0	200.074175	Action|Adventure|Thriller	Christoph Waltz	
3	Christopher Nolan	813.0	448.130642	Action|Thriller	Tom Hardy	
5	Andrew Stanton	462.0	73.058679	Action|Adventure|Sci-Fi	Daryl Sabara	Jo
...	...	...	...	...	...	
5033	Shane Carruth	143.0	0.424760	Drama|Sci-Fi|Thriller	Shane Carruth	
5034	Neill Dela Llana	35.0	0.070071	Thriller	Ian Gamazon	
5035	Robert Rodriguez	56.0	2.040920	Action|Crime|Drama|Romance|Thriller	Carlos Gallardo	El
5037	Edward Burns	14.0	0.004584	Comedy|Drama	Kerry Bishé	N
5042	Jon Gunn	43.0	0.085222	Documentary	John August	\

3891 rows × 13 columns

```
In [ ]: #2.2) Find the movies with highest profit
In [126]: #2.2.1) Create a new column called profit which contains difference between two columns
          gross and budget
          movies['profit']= movies['gross']-movies['budget']
          movies
```

Out[126]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	mc
0	James Cameron	723.0	7.605058e-04	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	
1	Gore Verbinski	302.0	3.094042e-04	Action|Adventure|Fantasy	Johnny Depp	F Ca A
2	Sam Mendes	602.0	2.000742e-04	Action|Adventure|Thriller	Christoph Waltz	
3	Christopher Nolan	813.0	4.481306e-04	Action|Thriller	Tom Hardy	-
5	Andrew Stanton	462.0	7.305868e-05	Action|Adventure|Sci-Fi	Daryl Sabara	Jol
...	...	...	...	...	...	
5033	Shane Carruth	143.0	4.247600e-07	Drama|Sci-Fi|Thriller	Shane Carruth	
5034	Neill Dela Llana	35.0	7.007100e-08	Thriller	Ian Gamazon	
5035	Robert Rodriguez	56.0	2.040920e-06	Action|Crime|Drama|Romance|Thriller	Carlos Gallardo	El
5037	Edward Burns	14.0	4.584000e-09	Comedy|Drama	Kerry Bishé	Ne
5042	Jon Gunn	43.0	8.522200e-08	Documentary	John August	w

3856 rows × 16 columns

In [127]:

```python
#2.2.2)sort the data using profit  column as reference
movies.sort_values(by='profit',ascending=False)
```

Out[127]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name
0	James Cameron	723.0	7.605058e-04	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder
29	Colin Trevorrow	644.0	6.521773e-04	Action|Adventure|Sci-Fi|Thriller	Bryce Dallas Howard
26	James Cameron	315.0	6.586723e-04	Drama|Romance	Leonardo DiCaprio
3024	George Lucas	282.0	4.609357e-04	Action|Adventure|Fantasy|Sci-Fi	Harrison Ford
3080	Steven Spielberg	215.0	4.349495e-04	Family|Sci-Fi	Henry Thomas
...	...	...	...	...	...
2334	Katsuhiro Ôtomo	105.0	4.103880e-07	Action|Adventure|Animation|Family|Sci-Fi|Thriller	William Hootkins
2323	Hayao Miyazaki	174.0	2.298191e-06	Adventure|Animation|Fantasy	Minnie Driver
3005	Lajos Koltai	73.0	1.958880e-07	Drama|Romance|War	Marcell Nagy
3859	Chan-wook Park	202.0	2.116670e-07	Crime|Drama	Min-sik Choi
2988	Joon-ho Bong	363.0	2.201412e-06	Comedy|Drama|Horror|Sci-Fi	Doona Bae

3856 rows × 16 columns

In [128]:
```
#2.2.3) movies with highest profit
top10=movies.sort_values(by='profit',ascending=False).head(10)
top10
```

Out[128]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name
0	James Cameron	723.0	0.000761	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder
29	Colin Trevorrow	644.0	0.000652	Action|Adventure|Sci-Fi|Thriller	Bryce Dallas Howard
26	James Cameron	315.0	0.000659	Drama|Romance	Leonardo DiCaprio
3024	George Lucas	282.0	0.000461	Action|Adventure|Fantasy|Sci-Fi	Harrison Ford
3080	Steven Spielberg	215.0	0.000435	Family|Sci-Fi	Henry Thomas
17	Joss Whedon	703.0	0.000623	Action|Adventure|Sci-Fi	Chris Hemsworth
509	Roger Allers	186.0	0.000423	Adventure|Animation|Drama|Family|Musical	Matthew Broderick
240	George Lucas	320.0	0.000475	Action|Adventure|Fantasy|Sci-Fi	Natalie Portman
66	Christopher Nolan	645.0	0.000533	Action|Crime|Drama|Thriller	Christian Bale
439	Gary Ross	673.0	0.000408	Adventure|Drama|Sci-Fi|Thriller	Jennifer Lawrence

In [ ]:
```
#There are some duplicates are present inorder to proceed we have to remove duplicates
#2.3) Duplicate records removing
```

In [68]:
```
movies.drop_duplicates(keep='first',inplace=True)
```

In [ ]:
```
#3)Find IMDB Top250
#1)Create a new column IMDb_Top_250 and store the top 250 movies with the highest IMDb R
ating (corresponding to the column: imdb_score). Also make sure that for all of these mo
vies, 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.

#2)Extract all the movies in the IMDb_Top_250 column which are not in the English langua
ge and store them in a new column named Top_Foreign_Lang_Film. You can use your own imag
ination also!
```

```
In [27]: movies
```

Out[27]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	m
0	James Cameron	723.0	760.505847	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	
1	Gore Verbinski	302.0	309.404152	Action|Adventure|Fantasy	Johnny Depp	C A
2	Sam Mendes	602.0	200.074175	Action|Adventure|Thriller	Christoph Waltz	
3	Christopher Nolan	813.0	448.130642	Action|Thriller	Tom Hardy	
5	Andrew Stanton	462.0	73.058679	Action|Adventure|Sci-Fi	Daryl Sabara	Jo
...	...	...	...	...	...	
5033	Shane Carruth	143.0	0.424760	Drama|Sci-Fi|Thriller	Shane Carruth	
5034	Neill Dela Llana	35.0	0.070071	Thriller	Ian Gamazon	
5035	Robert Rodriguez	56.0	2.040920	Action|Crime|Drama|Romance|Thriller	Carlos Gallardo	El
5037	Edward Burns	14.0	0.004584	Comedy|Drama	Kerry Bishé	Ne
5042	Jon Gunn	43.0	0.085222	Documentary	John August	V

3856 rows × 14 columns

In [29]:
```python
top10=movies.sort_values(by='profit',ascending=False).head(10)
top10
```

Out[29]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_nam
0	James Cameron	723.0	760.505847	Action|Adventure|Fantasy|Sci-Fi	CCH Pounde
29	Colin Trevorrow	644.0	652.177271	Action|Adventure|Sci-Fi|Thriller	Bryce Dalla Howar
26	James Cameron	315.0	658.672302	Drama|Romance	Leonard DiCapri
3024	George Lucas	282.0	460.935665	Action|Adventure|Fantasy|Sci-Fi	Harrison For
3080	Steven Spielberg	215.0	434.949459	Family|Sci-Fi	Henry Thoma
17	Joss Whedon	703.0	623.279547	Action|Adventure|Sci-Fi	Chri Hemswort
509	Roger Allers	186.0	422.783777	Adventure|Animation|Drama|Family|Musical	Matthe Broderic
240	George Lucas	320.0	474.544677	Action|Adventure|Fantasy|Sci-Fi	Natali Portma
66	Christopher Nolan	645.0	533.316061	Action|Crime|Drama|Thriller	Christian Bal
439	Gary Ross	673.0	407.999255	Adventure|Drama|Sci-Fi|Thriller	Jennife Lawrenc

In [29]:
```
movies
```

Out[29]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	m
0	James Cameron	723.0	760.505847	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	
1	Gore Verbinski	302.0	309.404152	Action|Adventure|Fantasy	Johnny Depp	C A
2	Sam Mendes	602.0	200.074175	Action|Adventure|Thriller	Christoph Waltz	
3	Christopher Nolan	813.0	448.130642	Action|Thriller	Tom Hardy	
5	Andrew Stanton	462.0	73.058679	Action|Adventure|Sci-Fi	Daryl Sabara	Jo
...	...	...	...	...	...	
5033	Shane Carruth	143.0	0.424760	Drama|Sci-Fi|Thriller	Shane Carruth	
5034	Neill Dela Llana	35.0	0.070071	Thriller	Ian Gamazon	
5035	Robert Rodriguez	56.0	2.040920	Action|Crime|Drama|Romance|Thriller	Carlos Gallardo	El
5037	Edward Burns	14.0	0.004584	Comedy|Drama	Kerry Bishé	Ne
5042	Jon Gunn	43.0	0.085222	Documentary	John August	\

3856 rows × 14 columns

In [129]:
```
IMDB_top_250 = movies[movies['num_voted_users']>25000].sort_values(by='imdb_score',ascen
ding=False).head(250)
IMDB_top_250
```

Out[129]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_t
1937	Frank Darabont	199.0	0.000028	Crime|Drama	Morgan Freeman	T Shawsha Redempt
3466	Francis Ford Coppola	208.0	0.000135	Crime|Drama	Al Pacino	T Godfat
2837	Francis Ford Coppola	149.0	0.000057	Crime|Drama	Robert De Niro	T Godfath Pa
66	Christopher Nolan	645.0	0.000533	Action|Crime|Drama|Thriller	Christian Bale	The D Kni
4498	Sergio Leone	181.0	0.000006	Western	Clint Eastwood	The Go the Bad a the U
...	...	...	...	...	...	
4931	John Carney	232.0	0.000009	Drama|Music|Romance	Glen Hansard	Or
2605	Ang Lee	287.0	0.000128	Action|Drama|Romance	Chen Chang	Crouch Tiger, Hid Drag
3029	David O. Russell	410.0	0.000094	Biography|Drama|Sport	Christian Bale	The Figh
2177	Tim Burton	111.0	0.000056	Fantasy|Romance	Johnny Depp	Edw Scissorhar
2487	George Cukor	82.0	0.000072	Drama|Family|Musical|Romance	Jeremy Brett	My Fair La

250 rows × 16 columns

In [33]:
```python
IMDB_top_250['Rank']=IMDB_top_250['imdb_score'].rank(method='first',ascending=False)
IMDB_top_250
```

Out[33]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie
1937	Frank Darabont	199.0	28.341469	Crime|Drama	Morgan Freeman	Shaws Redem
3466	Francis Ford Coppola	208.0	134.821952	Crime|Drama	Al Pacino	God
2837	Francis Ford Coppola	149.0	57.300000	Crime|Drama	Robert De Niro	Godf I
66	Christopher Nolan	645.0	533.316061	Action|Crime|Drama|Thriller	Christian Bale	The k
4498	Sergio Leone	181.0	6.100000	Western	Clint Eastwood	The (the Ba the
...	...	...	...	...	...	
4931	John Carney	232.0	9.437933	Drama|Music|Romance	Glen Hansard	
2605	Ang Lee	287.0	128.067808	Action|Drama|Romance	Chen Chang	Crou Tiger, H D
3029	David O. Russell	410.0	93.571803	Biography|Drama|Sport	Christian Bale	The F
2177	Tim Burton	111.0	56.362352	Fantasy|Romance	Johnny Depp	Ec Scissorh
2487	George Cukor	82.0	72.000000	Drama|Family|Musical|Romance	Jeremy Brett	My Fair

250 rows × 15 columns

In [34]:
```python
IMDB_top_250.to_csv('IMDB_Top_250.csv')
```

```
In [32]: Top_Foreign_Lang_Film = IMDB_top_250[IMDB_top_250['language']!='English']
         Top_Foreign_Lang_Film
```

Out[32]:

	director_name	num_critic_for_reviews	gross	genres	act
4498	Sergio Leone	181.0	6.100000	Western	Clir
4747	Akira Kurosawa	153.0	0.269061	Action|Adventure|Drama	
4029	Fernando Meirelles	214.0	7.563397	Crime|Drama	
2373	Hayao Miyazaki	246.0	10.049886	Adventure|Animation|Family|Fantasy	
4259	Florian Henckel von Donnersmarck	215.0	11.284657	Drama|Thriller	
4921	Majid Majidi	46.0	0.925402	Drama|Family	
2323	Hayao Miyazaki	174.0	2.298191	Adventure|Animation|Fantasy	M
2970	Wolfgang Petersen	96.0	11.433134	Adventure|Drama|Thriller|War	
4105	Chan-wook Park	305.0	2.181290	Drama|Mystery|Thriller	I
4659	Asghar Farhadi	354.0	7.098492	Drama|Mystery	
1329	S.S. Rajamouli	44.0	6.498000	Action|Adventure|Drama|Fantasy|War	
1298	Jean-Pierre Jeunet	242.0	33.201661	Comedy|Romance	
2734	Fritz Lang	260.0	0.026435	Drama|Sci-Fi	E
4033	Thomas Vinterberg	349.0	0.610968	Drama	
2829	Oliver Hirschbiegel	192.0	5.501940	Biography|Drama|History|War	K
2551	Guillermo del Toro	406.0	37.623143	Drama|Fantasy|War	Iva
4000	Juan José Campanella	262.0	20.167424	Drama|Mystery|Thriller	Ri
3550	Denis Villeneuve	226.0	6.857096	Drama|Mystery|War	Lu
2047	Hayao Miyazaki	212.0	4.710455	Adventure|Animation|Family|Fantasy	Cl
2830	Alejandro Amenábar	157.0	2.086345	Biography|Drama|Romance	B
2914	Je-kyu Kang	86.0	1.110186	Action|Drama|War	I
4461	Thomas Vinterberg	98.0	1.647780	Drama	

	director_name	num_critic_for_reviews	gross	genres	act
3553	José Padilha	142.0	0.008060	Action|Crime|Drama|Thriller	Wa
3423	Katsuhiro Ôtomo	150.0	0.439162	Action|Animation|Sci-Fi	M
4267	Alejandro G. Iñárritu	157.0	5.383834	Drama|Thriller	
3456	Vincent Paronnaud	242.0	4.443403	Animation|Biography|Drama|War	
3344	Karan Johar	210.0	4.018695	Adventure|Drama|Thriller	
4144	Walter Salles	71.0	5.595428	Drama	I
4284	Ari Folman	231.0	2.283276	Animation|Biography|Documentary|Drama|History|War	
4897	Sergio Leone	122.0	3.500000	Action|Drama|Western	Clir
1171	Yimou Zhang	283.0	0.084961	Action|Adventure|History	
2863	Clint Eastwood	251.0	13.753931	Drama|History|War	Yuk
3264	Michael Haneke	447.0	0.225377	Drama|Romance	
3510	Yash Chopra	29.0	2.921738	Drama|Musical|Romance	
3677	Christophe Barratier	112.0	3.629758	Drama|Music	Je
4415	Fabián Bielinsky	94.0	1.221261	Crime|Drama|Thriller	Ri
4640	Cristian Mungiu	233.0	1.185783	Drama	
2605	Ang Lee	287.0	128.067808	Action|Drama|Romance	C

In [ ]:

```
#4) Find out the top 10 directors for whom the mean of imdb_score is the highest and sto
re them in a new column top10director.
#In case of a tie in IMDb score between two directors, sort them alphabetically
```

In [70]:
```
#top 10 directors
top10director=movies.groupby('director_name').imdb_score.mean().sort_values(ascending=Fa
lse).head(10)
top10director
```

Out[70]:
```
director_name
Charles Chaplin       8.600000
Tony Kaye             8.600000
Ron Fricke            8.500000
Damien Chazelle       8.500000
Majid Majidi          8.500000
Alfred Hitchcock      8.500000
Sergio Leone          8.433333
Christopher Nolan     8.425000
Asghar Farhadi        8.400000
Richard Marquand      8.400000
Name: imdb_score, dtype: float64
```

In [ ]:
```
#5) Find popular genres, Perform this step using the knowledge gained while performing p
revious steps.
```

In [75]:
```
TempGenre=movies.genres.str.split('|',expand=True).iloc[:,0:2]
TempGenre.columns = ['genre_1','genre_2']
TempGenre
```

Out[75]:

	genre_1	genre_2
0	Action	Adventure
1	Action	Adventure
2	Action	Adventure
3	Action	Thriller
5	Action	Adventure
...	...	...
5033	Drama	Sci-Fi
5034	Thriller	None
5035	Action	Crime
5037	Comedy	Drama
5042	Documentary	None

3856 rows × 2 columns

```
In [76]: # replace None from genre_2 column
         TempGenre.genre_2.fillna(TempGenre.genre_1,inplace=True)
         TempGenre
```

Out[76]:

	genre_1	genre_2
0	Action	Adventure
1	Action	Adventure
2	Action	Adventure
3	Action	Thriller
5	Action	Adventure
...	...	...
5033	Drama	Sci-Fi
5034	Thriller	Thriller
5035	Action	Crime
5037	Comedy	Drama
5042	Documentary	Documentary

3856 rows × 2 columns

In [77]:
```
movies=pd.concat([movies,TempGenre],axis=1)
movies
```

Out[77]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	m
0	James Cameron	723.0	760.505847	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	
1	Gore Verbinski	302.0	309.404152	Action|Adventure|Fantasy	Johnny Depp	C
2	Sam Mendes	602.0	200.074175	Action|Adventure|Thriller	Christoph Waltz	
3	Christopher Nolan	813.0	448.130642	Action|Thriller	Tom Hardy	
5	Andrew Stanton	462.0	73.058679	Action|Adventure|Sci-Fi	Daryl Sabara	Jo
...	...	...	...	...	...	
5033	Shane Carruth	143.0	0.424760	Drama|Sci-Fi|Thriller	Shane Carruth	
5034	Neill Dela Llana	35.0	0.070071	Thriller	Ian Gamazon	
5035	Robert Rodriguez	56.0	2.040920	Action|Crime|Drama|Romance|Thriller	Carlos Gallardo	El
5037	Edward Burns	14.0	0.004584	Comedy|Drama	Kerry Bishé	Ne
5042	Jon Gunn	43.0	0.085222	Documentary	John August	

3856 rows × 15 columns

In [47]:
```
movies = movies.drop([
    'genre_1','genre_2'],axis=1)
```

```
In [122]: movies
```

Out[122]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	m
0	James Cameron	723.0	760.505847	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	
1	Gore Verbinski	302.0	309.404152	Action|Adventure|Fantasy	Johnny Depp	C
2	Sam Mendes	602.0	200.074175	Action|Adventure|Thriller	Christoph Waltz	
3	Christopher Nolan	813.0	448.130642	Action|Thriller	Tom Hardy	
5	Andrew Stanton	462.0	73.058679	Action|Adventure|Sci-Fi	Daryl Sabara	Jo
...	...	...	...	...	...	
5033	Shane Carruth	143.0	0.424760	Drama|Sci-Fi|Thriller	Shane Carruth	
5034	Neill Dela Llana	35.0	0.070071	Thriller	Ian Gamazon	
5035	Robert Rodriguez	56.0	2.040920	Action|Crime|Drama|Romance|Thriller	Carlos Gallardo	El
5037	Edward Burns	14.0	0.004584	Comedy|Drama	Kerry Bishé	Ne
5042	Jon Gunn	43.0	0.085222	Documentary	John August	

3856 rows × 14 columns

In [83]:
```
movies= pd.concat([movies,TempGenre],axis=1)
movies
```

Out[83]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	m
0	James Cameron	723.0	760.505847	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	
1	Gore Verbinski	302.0	309.404152	Action|Adventure|Fantasy	Johnny Depp	C
2	Sam Mendes	602.0	200.074175	Action|Adventure|Thriller	Christoph Waltz	
3	Christopher Nolan	813.0	448.130642	Action|Thriller	Tom Hardy	
5	Andrew Stanton	462.0	73.058679	Action|Adventure|Sci-Fi	Daryl Sabara	Jo
...	...	...	...	...	...	
5033	Shane Carruth	143.0	0.424760	Drama|Sci-Fi|Thriller	Shane Carruth	
5034	Neill Dela Llana	35.0	0.070071	Thriller	Ian Gamazon	
5035	Robert Rodriguez	56.0	2.040920	Action|Crime|Drama|Romance|Thriller	Carlos Gallardo	El
5037	Edward Burns	14.0	0.004584	Comedy|Drama	Kerry Bishé	Ne
5042	Jon Gunn	43.0	0.085222	Documentary	John August	

3856 rows × 15 columns

In [113]:
```
PopGen=movies.groupby(['genre_1','genre_2']).gross.mean().sort_values(ascending=False).h
ead(5)
PopGen
```

Out[113]:
```
genre_1    genre_2
Family     Sci-Fi       434.949459
Adventure  Sci-Fi       228.627758
           Family       118.919540
           Animation    116.998550
Action     Adventure    109.595465
Name: gross, dtype: float64
```

In [112]:
```
PopGen.to_csv('Pop_gen.csv')
```

```
C:\Users\Parashu\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: FutureWarning: Th
e signature of `Series.to_csv` was aligned to that of `DataFrame.to_csv`, and argument
'header' will change its default value from False to True: please pass an explicit valu
e to suppress this warning.
  """Entry point for launching an IPython kernel.
```

In [ ]:
```
#6) Create three new columns namely, Meryl_Streep, Leo_Caprio, and Brad_Pitt which conta
in 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 th
at you use the names 'Meryl Streep', 'Leonardo DiCaprio', and 'Brad Pitt' for the said e
xtraction.

#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 a
ctors which have the highest mean.
```

In [85]:
```
Meryl_Streep=movies[movies['actor_1_name']=='Meryl Streep']
Leo_Caprio=movies[movies['actor_1_name']=='Leonardo DiCaprio']
Brad_Pitt=movies[movies['actor_1_name']=='Brad Pitt']
```

In [86]:
```
Combined=Meryl_Streep.append([Leo_Caprio,Brad_Pitt])
Combined
```

Out[86]:

	director_name	num_critic_for_reviews	gross	genres	act
410	Nancy Meyers	187.0	112.703470	Comedy|Drama|Romance	M
1106	Curtis Hanson	42.0	46.815748	Action|Adventure|Crime|Thriller	M
1204	Nora Ephron	252.0	94.125426	Biography|Drama|Romance	M
1408	David Frankel	208.0	124.732962	Comedy|Drama|Romance	M
1483	Robert Redford	227.0	14.998070	Drama|Thriller|War	M
1575	Sydney Pollack	66.0	87.100000	Biography|Drama|Romance	M
1618	David Frankel	234.0	63.536011	Comedy|Drama|Romance	M
1674	Carl Franklin	64.0	23.209440	Drama	M
1925	Stephen Daldry	174.0	41.597830	Drama|Romance	M
2781	Phyllida Lloyd	331.0	29.959436	Biography|Drama|History	M
3135	Robert Altman	211.0	20.338609	Comedy|Drama|Music	M
26	James Cameron	315.0	658.672302	Drama|Romance	
50	Baz Luhrmann	490.0	144.812796	Drama|Romance	
97	Christopher Nolan	642.0	292.568851	Action|Adventure|Sci-Fi|Thriller	
179	Alejandro G. Iñárritu	556.0	183.635922	Adventure|Drama|Thriller|Western	
257	Martin Scorsese	267.0	102.608827	Biography|Drama	
296	Quentin Tarantino	765.0	162.804648	Drama|Western	
307	Edward Zwick	166.0	57.366262	Adventure|Drama|Thriller	
308	Martin Scorsese	606.0	116.866727	Biography|Comedy|Crime|Drama	
326	Martin Scorsese	233.0	77.679638	Crime|Drama	
361	Martin Scorsese	352.0	132.373442	Crime|Drama|Thriller	
452	Martin Scorsese	490.0	127.968405	Mystery|Thriller	
641	Ridley Scott	238.0	39.380442	Action|Drama|Thriller	
911	Steven Spielberg	194.0	164.435221	Biography|Crime|Drama	
990	Danny Boyle	118.0	39.778599	Adventure|Drama|Thriller	

	director_name	num_critic_for_reviews	gross	genres	acto
1114	Sam Mendes	323.0	22.877808	Drama|Romance	
1422	Randall Wallace	83.0	56.876365	Action|Adventure	
1453	Clint Eastwood	392.0	37.304950	Biography|Crime|Drama	
1560	Sam Raimi	63.0	18.636537	Action|Thriller|Western	
2067	Jerry Zaks	45.0	12.782508	Drama	
2757	Baz Luhrmann	106.0	46.338728	Drama|Romance	
3476	Baz Luhrmann	490.0	144.812796	Drama|Romance	
101	David Fincher	362.0	127.490802	Drama|Fantasy|Romance	
147	Wolfgang Petersen	220.0	133.228348	Adventure	
254	Steven Soderbergh	198.0	125.531634	Crime|Thriller	
255	Doug Liman	233.0	186.336103	Action|Comedy|Crime|Romance|Thriller	
382	Tony Scott	142.0	0.026871	Action|Crime|Thriller	
400	Steven Soderbergh	186.0	183.405771	Crime|Thriller	
470	David Ayer	406.0	85.707116	Action|Drama|War	
611	Jean-Jacques Annaud	76.0	37.901509	Adventure|Biography|Drama|History|War	
683	David Fincher	315.0	37.023395	Drama	
792	Patrick Gilmore	98.0	26.288320	Adventure|Animation|Comedy|Drama|Family|Fantas...	
940	Neil Jordan	120.0	105.264608	Drama|Fantasy|Horror	
1490	Terrence Malick	584.0	13.303319	Drama|Fantasy	
1722	Andrew Dominik	273.0	3.904982	Biography|Crime|Drama|History|Western	
2204	Alejandro G. Iñárritu	285.0	34.300771	Drama	
2333	Angelina Jolie Pitt	131.0	0.531009	Drama|Romance	
2682	Andrew Dominik	414.0	14.938570	Crime|Thriller	

	director_name	num_critic_for_reviews	gross	genres	actc
2898	Tony Scott	122.0	12.281500	Action|Crime|Drama|Romance|Thriller	

```
In [ ]: #num_critic_for_reviews and num_users_for_reviews
```

```
In [87]: Combined.groupby('actor_1_name').num_critic_for_reviews.mean()
```

```
Out[87]: actor_1_name
         Brad Pitt            245.000000
         Leonardo DiCaprio    330.190476
         Meryl Streep         181.454545
         Name: num_critic_for_reviews, dtype: float64
```

```
In [88]: Combined.groupby('actor_1_name').num_user_for_reviews.mean()
```

```
---------------------------------------------------------------------------
DataError                                 Traceback (most recent call last)
<ipython-input-88-d884db1b60d4> in <module>
----> 1 Combined.groupby('actor_1_name').num_user_for_reviews.mean()

~\Anaconda3\lib\site-packages\pandas\core\groupby\groupby.py in mean(self, *args, **kwa
rgs)
   1203         try:
   1204             return self._cython_agg_general(
-> 1205                 "mean", alt=lambda x, axis: Series(x).mean(**kwargs), **kwargs
   1206             )
   1207         except GroupByError:

~\Anaconda3\lib\site-packages\pandas\core\groupby\groupby.py in _cython_agg_general(sel
f, how, alt, numeric_only, min_count)
    886
    887         if len(output) == 0:
--> 888             raise DataError("No numeric types to aggregate")
    889
    890         return self._wrap_aggregated_output(output, names)

DataError: No numeric types to aggregate
```

```
In [89]: Combined.num_user_for_reviews=Combined.num_user_for_reviews.astype('int')
         Combined.num_user_for_reviews
```

```
Out[89]: 410      214
         1106      69
         1204     277
         1408     631
         1483     298
         1575     200
         1618     178
         1674     112
         1925     660
         2781     350
         3135     280
         26      2528
         50       753
         97      2803
         179     1188
         257      799
         296     1193
         307      657
         308     1138
         326     1166
         361     2054
         452      964
         641      263
         911      667
         990      548
         1114     414
         1422     244
         1453     279
         1560     216
         2067      71
         2757     506
         3476     753
         101      822
         147     1694
         254      627
         255      798
         382      361
         400      845
         470      701
         611      119
         683     2968
         792       91
         940      406
         1490     975
         1722     415
         2204     908
         2333      61
         2682     369
         2898     460
         Name: num_user_for_reviews, dtype: int32
```

In [90]:
```
Combined.groupby('actor_1_name').num_user_for_reviews.mean()
```

Out[90]:
```
actor_1_name
Brad Pitt            742.352941
Leonardo DiCaprio    914.476190
Meryl Streep         297.181818
Name: num_user_for_reviews, dtype: float64
```

In [91]:
```
Combined.groupby('actor_1_name')[['num_critic_for_reviews','num_user_for_reviews']].mean
()
```

Out[91]:

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 [ ]:
```
#6.1)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 d
ecade.
#Store this in a new data frame called df_by_decade.
```

In [131]:
```
df_decade=movies.copy(deep=True)
```

In [106]:
```
movies.title_year= movies.title_year.astype('category')
movies.title_year
```

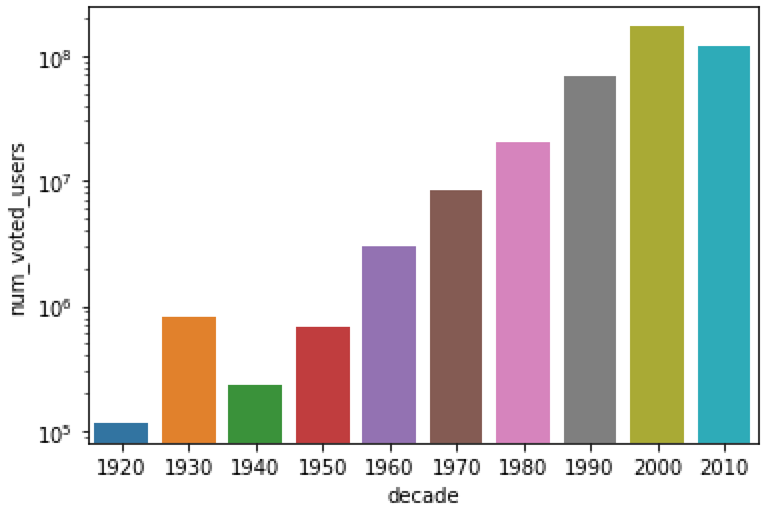
Out[106]:
```
0       2009
1       2007
2       2015
3       2012
5       2012
        ...
5033    2004
5034    2005
5035    1992
5037    2011
5042    2004
Name: title_year, Length: 3856, dtype: category
Categories (75, int64): [1920, 1927, 1929, 1933, ..., 2013, 2014, 2015, 2016]
```

In [132]:
```
df_decade['decade']=df_decade['title_year'].apply(lambda x:10*(int(x/10)))
```

In [133]:
```
df_decade=df_decade.groupby('decade',as_index=False)['num_voted_users'].sum().sort_value
s(by="decade")
```

In [47]:
```
import seaborn as sns
```

In [134]:
```
ax=sns.barplot(x='decade',y='num_voted_users',data=df_decade)
plt.yscale('log')
plt.ylabel("num_voted_users")
plt.xlabel("decade")
plt.show()
```



In [ ]:
```
```