Name: Pranav Sanjay Vispute

Email: pranavvispute671@gmail.com

Project Name: IMDB Case Study

In [47]:

```
# Supress Warnings

import warnings
warnings.filterwarnings('ignore')
```

In [48]:

```
# Import the numpy and pandas packages

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

Task 1: Reading and Inspection

- ## Subtask 1.1: Import and read

Import and read the movie database. Store it in a variable called `movies` .

In [49]:

```
movies = pd.read_csv("IMDB_Movies.csv")
movies
```

Out[49]:

	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

- **Subtask 1.2: Inspect the dataframe**

Inspect the dataframe's columns, shapes, variable types etc.

In [50]:

```
# Write your code for inspection here
movies.columns
```

Out[50]:

```
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 [51]:

```
movies.dtypes
```

Out[51]:

```
color                       object
director_name               object
num_critic_for_reviews     float64
duration                   float64
director_facebook_likes    float64
actor_3_facebook_likes     float64
actor_2_name                object
actor_1_facebook_likes     float64
gross                      float64
genres                      object
actor_1_name                object
movie_title                 object
num_voted_users              int64
cast_total_facebook_likes    int64
actor_3_name                object
facenumber_in_poster       float64
plot_keywords               object
movie_imdb_link             object
num_user_for_reviews        object
language                    object
country                     object
content_rating              object
budget                     float64
title_year                 float64
actor_2_facebook_likes     float64
imdb_score                 float64
aspect_ratio               float64
movie_facebook_likes         int64
dtype: object
```

In [52]:

```
movies.content_rating.describe()
```

Out[52]:

```
count     4740
unique      18
top          R
freq      2118
Name: content_rating, dtype: object
```

In [53]:

```
movies.duration.describe()
```

Out[53]:

```
count    5028.000000
mean      107.201074
std        25.197441
min         7.000000
25%        93.000000
50%       103.000000
75%       118.000000
max       511.000000
Name: duration, dtype: float64
```

In [54]:

```
movies.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
```

In [55]:

```
movies.shape
```

Out[55]:

```
(5043, 28)
```

In [56]:

```
movies.describe()
```

Out[56]:

	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	act
count	4993.000000	5028.000000	4939.000000	5020.000000	
mean	140.194272	107.201074	686.509212	645.009761	
std	121.601675	25.197441	2813.328607	1665.041728	
min	1.000000	7.000000	0.000000	0.000000	
25%	50.000000	93.000000	7.000000	133.000000	
50%	110.000000	103.000000	49.000000	371.500000	
75%	195.000000	118.000000	194.500000	636.000000	
max	813.000000	511.000000	23000.000000	23000.000000	

Task 2: Cleaning the Data

- ## Subtask 2.1: Inspect Null values

Find out the number of Null values in all the columns and rows. Also, find the percentage of Null values in each column. Round off the percentages upto two decimal places.

In [57]:

```
# Write your code for column-wise null count here
movies.isnull().sum()
```

Out[57]:

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

In [58]:

```
# Write your code for row-wise null count here
for i in range(len(movies.index)):
    print("There are ",movies.iloc[i].isnull().sum()," null values in row :",i)
```

```
There are  0  null values in row : 0
There are  0  null values in row : 1
There are  0  null values in row : 2
There are  0  null values in row : 3
There are  13  null values in row : 4
There are  0  null values in row : 5
There are  0  null values in row : 6
There are  0  null values in row : 7
There are  0  null values in row : 8
There are  0  null values in row : 9
There are  0  null values in row : 10
There are  0  null values in row : 11
There are  0  null values in row : 12
There are  0  null values in row : 13
There are  0  null values in row : 14
There are  0  null values in row : 15
There are  0  null values in row : 16
There are  0  null values in row : 17
There are  0  null values in row : 18
```

In [59]:

```
# Write your code for column-wise null percentages here
round(movies.isnull().mean()*100,2)
```

Out[59]:

```
color                        0.38
director_name                2.06
num_critic_for_reviews       0.99
duration                     0.30
director_facebook_likes      2.06
actor_3_facebook_likes       0.46
actor_2_name                 0.26
actor_1_facebook_likes       0.14
gross                       17.53
genres                       0.00
actor_1_name                 0.14
movie_title                  0.00
num_voted_users              0.00
cast_total_facebook_likes    0.00
actor_3_name                 0.46
facenumber_in_poster         0.26
plot_keywords                3.03
movie_imdb_link              0.00
num_user_for_reviews         0.40
language                     0.24
country                      0.10
content_rating               6.01
budget                       9.76
title_year                   2.14
actor_2_facebook_likes       0.26
imdb_score                   0.00
aspect_ratio                 6.52
movie_facebook_likes         0.00
dtype: float64
```

- ## Subtask 2.2: Drop unecessary columns

For this assignment, you will mostly be analyzing the movies with respect to the ratings, gross collection, popularity of movies, etc. So many of the columns in this dataframe are not required. So it is advised to drop the following columns.

- 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

In [60]:

```
# Write your code for dropping the columns here. It is advised to keep inspecting the dataf
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"], inplace=True, axis=1, errors='ignore')
movies
```

Out[60]:

	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
1	Gore Verbinski	302.0	309404152.0	Action|Adventure|Fantasy	Johnny I
2	Sam Mendes	602.0	200074175.0	Action|Adventure|Thriller	Chris \
3	Christopher Nolan	813.0	448130642.0	Action|Thriller	Tom H
4	Doug Walker	NaN	NaN	Documentary	Doug W
...	...	...	...	...	
5038	Scott Smith	1.0	NaN	Comedy|Drama	Eric Ma
5039	NaN	43.0	NaN	Crime|Drama|Mystery|Thriller	Natalie
5040	Benjamin Roberds	13.0	NaN	Drama|Horror|Thriller	Eva Boe
5041	Daniel Hsia	14.0	10443.0	Comedy|Drama|Romance	Alan
5042	Jon Gunn	43.0	85222.0	Documentary	John Au

5043 rows × 13 columns

- **Subtask 2.3: Drop unecessary rows using columns with high Null percentages**

Now, on inspection you might notice that some columns have large percentage (greater than 5%) of Null values. Drop all the rows which have Null values for such columns.

In [61]:

```
# Write your code for dropping the rows here
round(movies.isnull().mean()*100,2)
movies.dropna(subset=['budget','gross'], inplace=True)
round(movies.isnull().mean()*100,2)
```

Out[61]:

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

- **Subtask 2.4: Fill NaN values**

You might notice that the `language` column has some NaN values. Here, on inspection, you will see that it is safe to replace all the missing values with `'English'`.

In [62]:

```
# Write your code for filling the NaN values in the 'language' column here
movies['language'].replace(np.nan, 'English', inplace=True)
```

- **Subtask 2.5: Check the number of retained rows**

You might notice that two of the columns viz. `num_critic_for_reviews` and `actor_1_name` have small percentages of NaN values left. You can let these columns as it is for now. Check the number and percentage of the rows retained after completing all the tasks above.

In [63]:

```
# Write your code for checking number of retained rows here
movies.shape
3891/5043*100
```

Out[63]:

```
77.15645449137418
```

Checkpoint 1: You might have noticed that we still have around `77%` of the rows!

Task 3: Data Analysis

- ## Subtask 3.1: Change the unit of columns

Convert the unit of the `budget` and `gross` columns from `$` to `million $`.

In [64]:

```
# Write your code for unit conversion here
movies['budget']=movies['budget']/1000000
movies['gross']=movies['gross']/1000000
movies
```

Out[64]:

	director_name	num_critic_for_reviews	gross	genres	acto
0	James Cameron	723.0	760.505847	Action|Adventure|Fantasy|Sci-Fi	CC
1	Gore Verbinski	302.0	309.404152	Action|Adventure|Fantasy	Jo
2	Sam Mendes	602.0	200.074175	Action|Adventure|Thriller	
3	Christopher Nolan	813.0	448.130642	Action|Thriller	
5	Andrew Stanton	462.0	73.058679	Action|Adventure|Sci-Fi	Da
...	...	...	...	...	
5033	Shane Carruth	143.0	0.424760	Drama|Sci-Fi|Thriller	Sha
5034	Neill Dela Llana	35.0	0.070071	Thriller	Ian
5035	Robert Rodriguez	56.0	2.040920	Action|Crime|Drama|Romance|Thriller	
5037	Edward Burns	14.0	0.004584	Comedy|Drama	I
5042	Jon Gunn	43.0	0.085222	Documentary	J

3891 rows × 13 columns

- ## Subtask 3.2: Find the movies with highest profit

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

4. Extract the top ten profiting movies in descending order and store them in a new dataframe - `top10`

In [65]:

```
# Write your code for creating the profit column here
movies['profit']=movies['gross']-movies['budget']
movies
```

Out[65]:

	director_name	num_critic_for_reviews	gross	genres	acto
0	James Cameron	723.0	760.505847	Action|Adventure|Fantasy|Sci-Fi	CC
1	Gore Verbinski	302.0	309.404152	Action|Adventure|Fantasy	Jo
2	Sam Mendes	602.0	200.074175	Action|Adventure|Thriller	
3	Christopher Nolan	813.0	448.130642	Action|Thriller	
5	Andrew Stanton	462.0	73.058679	Action|Adventure|Sci-Fi	D
...	...	...	...	...	
5033	Shane Carruth	143.0	0.424760	Drama|Sci-Fi|Thriller	Sha
5034	Neill Dela Llana	35.0	0.070071	Thriller	Ian
5035	Robert Rodriguez	56.0	2.040920	Action|Crime|Drama|Romance|Thriller	
5037	Edward Burns	14.0	0.004584	Comedy|Drama	I
5042	Jon Gunn	43.0	0.085222	Documentary	J

3891 rows × 14 columns

In [66]:

```
# Write your code for sorting the dataframe here
movies.sort_values(by=['profit'], inplace=True, ascending =False)
movies
```

Out[66]:

	director_name	num_critic_for_reviews	gross	genres	ac
0	James Cameron	723.0	760.505847	Action|Adventure|Fantasy|Sci-Fi	C
29	Colin Trevorrow	644.0	652.177271	Action|Adventure|Sci-Fi|Thriller	
26	James Cameron	315.0	658.672302	Drama|Romance	
3024	George Lucas	282.0	460.935665	Action|Adventure|Fantasy|Sci-Fi	H
3080	Steven Spielberg	215.0	434.949459	Family|Sci-Fi	H
...	...	...	...	...	
2334	Katsuhiro Ôtomo	105.0	0.410388	Action|Adventure|Animation|Family|Sci-Fi|Thriller	
2323	Hayao Miyazaki	174.0	2.298191	Adventure|Animation|Fantasy	I
3005	Lajos Koltai	73.0	0.195888	Drama|Romance|War	I
3859	Chan-wook Park	202.0	0.211667	Crime|Drama	
2988	Joon-ho Bong	363.0	2.201412	Comedy|Drama|Horror|Sci-Fi	

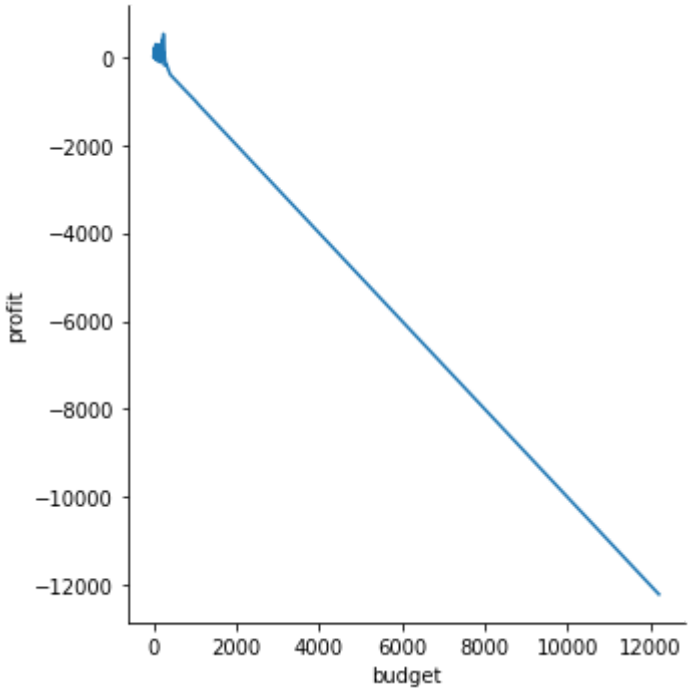
3891 rows × 14 columns

In [67]:

```
# Write code for profit vs budget plot here
sns.relplot(x=movies.budget, y=movies.profit, kind='line')
```

Out[67]:

```
<seaborn.axisgrid.FacetGrid at 0x1c5450ea1f0>
```



In [68]:

```
top10 = movies.head(10)
top10
```

Out[68]:

	director_name	num_critic_for_reviews	gross	genres
0	James Cameron	723.0	760.505847	Action|Adventure|Fantasy|Sci-Fi
29	Colin Trevorrow	644.0	652.177271	Action|Adventure|Sci-Fi|Thriller
26	James Cameron	315.0	658.672302	Drama|Romance
3024	George Lucas	282.0	460.935665	Action|Adventure|Fantasy|Sci-Fi
3080	Steven Spielberg	215.0	434.949459	Family|Sci-Fi
794	Joss Whedon	703.0	623.279547	Action|Adventure|Sci-Fi
17	Joss Whedon	703.0	623.279547	Action|Adventure|Sci-Fi
509	Roger Allers	186.0	422.783777	Adventure|Animation|Drama|Family|Musical
240	George Lucas	320.0	474.544677	Action|Adventure|Fantasy|Sci-Fi
66	Christopher Nolan	645.0	533.316061	Action|Crime|Drama|Thriller

- ## Subtask 3.3: Drop duplicate values

After you found out the top 10 profiting movies, you might have noticed a duplicate value. So, it seems like the dataframe has duplicate values as well. Drop the duplicate values from the dataframe and repeat `Subtask 3.2`. Note that the same `movie_title` can be there in different languages.

In [69]:

```
# Write your code for dropping duplicate values here
movies.drop_duplicates(keep=False, inplace=True)
movies
```

Out[69]:

	director_name	num_critic_for_reviews	gross	genres	ac
0	James Cameron	723.0	760.505847	Action|Adventure|Fantasy|Sci-Fi	C
29	Colin Trevorrow	644.0	652.177271	Action|Adventure|Sci-Fi|Thriller	
26	James Cameron	315.0	658.672302	Drama|Romance	
3024	George Lucas	282.0	460.935665	Action|Adventure|Fantasy|Sci-Fi	H
3080	Steven Spielberg	215.0	434.949459	Family|Sci-Fi	H
...	...	...	...	...	
2334	Katsuhiro Ôtomo	105.0	0.410388	Action|Adventure|Animation|Family|Sci-Fi|Thriller	
2323	Hayao Miyazaki	174.0	2.298191	Adventure|Animation|Fantasy	I
3005	Lajos Koltai	73.0	0.195888	Drama|Romance|War	I
3859	Chan-wook Park	202.0	0.211667	Crime|Drama	
2988	Joon-ho Bong	363.0	2.201412	Comedy|Drama|Horror|Sci-Fi	

3821 rows × 14 columns

In [70]:

```
# Write code for repeating subtask 2 here
movies.sort_values(by=['profit'], inplace=True, ascending=False)
movies.head(10)
```

Out[70]:

	director_name	num_critic_for_reviews	gross	genres
0	James Cameron	723.0	760.505847	Action|Adventure|Fantasy|Sci-Fi
29	Colin Trevorrow	644.0	652.177271	Action|Adventure|Sci-Fi|Thriller
26	James Cameron	315.0	658.672302	Drama|Romance
3024	George Lucas	282.0	460.935665	Action|Adventure|Fantasy|Sci-Fi
3080	Steven Spielberg	215.0	434.949459	Family|Sci-Fi
509	Roger Allers	186.0	422.783777	Adventure|Animation|Drama|Family|Musical
240	George Lucas	320.0	474.544677	Action|Adventure|Fantasy|Sci-Fi
66	Christopher Nolan	645.0	533.316061	Action|Crime|Drama|Thriller
439	Gary Ross	673.0	407.999255	Adventure|Drama|Sci-Fi|Thriller
812	Tim Miller	579.0	363.024263	Action|Adventure|Comedy|Romance|Sci-Fi

Checkpoint 2: You might spot two movies directed by `James Cameron` in the list.

- ## Subtask 3.4: Find IMDb Top 250

1. Create a new dataframe `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.
2. Extract all the movies in the `IMDb_Top_250` dataframe which are not in the English language and store them in a new dataframe named `Top_Foreign_Lang_Film` .

In [71]:

```
# Write your code for extracting the top 250 movies as per the IMDb score here. Make sure t
# and name that dataframe as 'IMDb_Top_250'
movies.sort_values(by=['imdb_score'], inplace=True, ascending=False)

user25k=movies[movies['num_voted_users']>25000]
IMDb_Top_250=user25k.head(250)
IMDb_Top_250.insert(0, 'Rank', range(1,1+len(IMDb_Top_250)))
IMDb_Top_250
```

Out[71]:

	Rank	director_name	num_critic_for_reviews	gross	
1937	1	Frank Darabont	199.0	28.341469	Crin
3466	2	Francis Ford Coppola	208.0	134.821952	Crin
2837	3	Francis Ford Coppola	149.0	57.300000	Crin
66	4	Christopher Nolan	645.0	533.316061	Action|Crime|Dran
4498	5	Sergio Leone	181.0	6.100000	
...	...	...	...	...	
1735	246	James Mangold	291.0	119.518352	Biography|Drama|Music|
927	247	Andrew Adamson	212.0	267.652016	Adventure|Animation|Comedy|Famil
1606	248	Nick Cassavetes	177.0	0.064286	Drama|
525	249	Alfonso Cuarón	372.0	35.286428	Drama|Sci-
353	250	John Lasseter	191.0	245.823397	Adventure|Animation|Comedy|Famil

250 rows × 15 columns

In [72]:

```
Top_Foreign_Lang_Film = IMDb_Top_250[IMDb_Top_250['language']!='English']
Top_Foreign_Lang_Film
```

Out[72]:

	Rank	director_name	num_critic_for_reviews	gross	
4498	5	Sergio Leone	181.0	6.100000	
4747	17	Akira Kurosawa	153.0	0.269061	Action
4029	18	Fernando Meirelles	214.0	7.563397	
2373	26	Hayao Miyazaki	246.0	10.049886	Adventure|Animati
4921	33	Majid Majidi	46.0	0.925402	
4259	34	Florian Henckel von Donnersmarck	215.0	11.284657	
1329	54	S.S. Rajamouli	44.0	6.498000	Action|Adventure|D
2323	55	Hayao Miyazaki	174.0	2.298191	Adventure|
1298	56	Jean-Pierre Jeunet	242.0	33.201661	
4659	58	Asghar Farhadi	354.0	7.098492	
4105	60	Chan-wook Park	305.0	2.181290	Dran
2970	61	Wolfgang Petersen	96.0	11.433134	Adventure|[
4033	71	Thomas Vinterberg	349.0	0.610968	
2829	82	Oliver Hirschbiegel	192.0	5.501940	Biography|[
2734	83	Fritz Lang	260.0	0.026435	
4000	84	Juan José Campanella	262.0	20.167424	Dran
3550	89	Denis Villeneuve	226.0	6.857096	D
2047	100	Hayao Miyazaki	212.0	4.710455	Adventure|Animati
2551	104	Guillermo del Toro	406.0	37.623143	D
3553	114	José Padilha	142.0	0.008060	Action|Cr

	Rank	director_name	num_critic_for_reviews	gross	
2914	118	Je-kyu Kang	86.0	1.110186	
2830	127	Alejandro Amenábar	157.0	2.086345	Biography
4461	137	Thomas Vinterberg	98.0	1.647780	
3423	138	Katsuhiro Ôtomo	150.0	0.439162	Actio
4267	145	Alejandro G. Iñárritu	157.0	5.383834	
3344	154	Karan Johar	210.0	4.018695	Advent
4144	181	Walter Salles	71.0	5.595428	
3456	182	Vincent Paronnaud	242.0	4.443403	Animation|Biog
4284	183	Ari Folman	231.0	2.283276	Animation|Biography|Documentary|D
4897	200	Sergio Leone	122.0	3.500000	Acti
3677	211	Christophe Barratier	112.0	3.629758	
4640	214	Cristian Mungiu	233.0	1.185783	
4415	215	Fabián Bielinsky	94.0	1.221261	Cr
3510	223	Yash Chopra	29.0	2.921738	Drama
3264	224	Michael Haneke	447.0	0.225377	
2863	234	Clint Eastwood	251.0	13.753931	D
2605	238	Ang Lee	287.0	128.067808	Actior

Checkpoint 3: Can you spot `Veer-Zaara` in the dataframe?

- ## Subtask 3.5: Find the best directors

1. Group the dataframe using the `director_name` column.
2. Find out the top 10 directors for whom the mean of `imdb_score` is the highest and store them in a new dataframe `top10director`. Incase of a tie in IMDb score between two directors, sort them alphabetically.

In [73]:

```
# Write your code for extracting the top 10 directors here
movies_mean=movies.groupby(['director_name'])[['imdb_score']].mean()
movies_mean.sort_values(by=['imdb_score', 'director_name'], inplace=True, ascending=[False,
top10director=movies_mean.head(10)
top10director
```

Out[73]:

director_name	imdb_score
Charles Chaplin	8.600000
Tony Kaye	8.600000
Alfred Hitchcock	8.500000
Damien Chazelle	8.500000
Majid Majidi	8.500000
Ron Fricke	8.500000
Sergio Leone	8.433333
Christopher Nolan	8.425000
Asghar Farhadi	8.400000
Marius A. Markevicius	8.400000

Checkpoint 4: No surprises that `Damien Chazelle` (director of Whiplash and La La Land) is in this list.

- ## Subtask 3.6: Find popular genres

You might have noticed the `genres` column in the dataframe with all the genres of the movies seperated by a pipe (|). Out of all the movie genres, the first two are most significant for any film.

1. Extract the first two genres from the `genres` column and store them in two new columns: `genre_1` and `genre_2` . Some of the movies might have only one genre. In such cases, extract the single genre into both the columns, i.e. for such movies the `genre_2` will be the same as `genre_1` .
2. Group the dataframe using `genre_1` as the primary column and `genre_2` as the secondary column.
3. Find out the 5 most popular combo of genres by finding the mean of the gross values using the `gross` column and store them in a new dataframe named `PopGenre` .

In [74]:

```
# Write your code for extracting the first two genres of each movie here
column = movies['genres'].apply(lambda x: pd.Series(x.split('|'))).rename(columns={0:'genre
movies["genre_1"] = movies["genres"].str.split("|").str.get(0)
movies["genre_2"] = movies["genres"].str.split("|").str.get(1)
movies["genre_2"] = movies["genre_2"].fillna(movies["genre_1"])
movies.head()
```

Out[74]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_nam
1937	Frank Darabont	199.0	28.341469	Crime|Drama	Morga Freema
3466	Francis Ford Coppola	208.0	134.821952	Crime|Drama	Al Pacin
2837	Francis Ford Coppola	149.0	57.300000	Crime|Drama	Robert De Nir
66	Christopher Nolan	645.0	533.316061	Action|Crime|Drama|Thriller	Christian Bal
4498	Sergio Leone	181.0	6.100000	Western	Clint Eastwoo

In [75]:

```
# Write your code for grouping the dataframe here
movies_by_segment = movies.groupby(['genre_1','genre_2'])
movies_by_segment
```

Out[75]:

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x000001C5476107F0>
```

In [76]:

```
# Write your code for getting the 5 most popular combo of genres here
PopGenre = pd.DataFrame(movies_by_segment.gross.mean()).sort_values("gross", ascending = Fa
PopGenre
```

Out[76]:

		gross
genre_1	**genre_2**	
Family	**Sci-Fi**	434.949459
Adventure	**Sci-Fi**	228.627758
	Family	121.463622
	Animation	116.998550
Action	**Adventure**	108.789045

Checkpoint 5: Well, as it turns out. `Family + Sci-Fi` is the most popular combo of genres out there!

- ## Subtask 3.7: Find the critic-favorite and audience-favorite actors

1. Create three new dataframes 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.
2. Append the rows of all these dataframes and store them in a new dataframe named `Combined` .
3. Group the combined dataframe using the `actor_1_name` column.
4. Find the mean of the `num_critic_for_reviews` and `num_users_for_review` and identify the actors which have the highest mean.
5. 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 dataframe 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` .

In [77]:

```
# Write your code for creating three new dataframes here
# Include all movies in which Meryl_Streep is the lead
Meryl_Streep = movies[movies['actor_1_name']=="Meryl Streep"]
Meryl_Streep
```

Out[77]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_n
1925	Stephen Daldry	174.0	41.597830	Drama|Romance	Meryl St
1575	Sydney Pollack	66.0	87.100000	Biography|Drama|Romance	Meryl St
1674	Carl Franklin	64.0	23.209440	Drama	Meryl St
1204	Nora Ephron	252.0	94.125426	Biography|Drama|Romance	Meryl St
1408	David Frankel	208.0	124.732962	Comedy|Drama|Romance	Meryl St
3135	Robert Altman	211.0	20.338609	Comedy|Drama|Music	Meryl St
410	Nancy Meyers	187.0	112.703470	Comedy|Drama|Romance	Meryl St
2781	Phyllida Lloyd	331.0	29.959436	Biography|Drama|History	Meryl St
1106	Curtis Hanson	42.0	46.815748	Action|Adventure|Crime|Thriller	Meryl St
1618	David Frankel	234.0	63.536011	Comedy|Drama|Romance	Meryl St
1483	Robert Redford	227.0	14.998070	Drama|Thriller|War	Meryl St

In [78]:

```
 # Include all movies in which Leo_Caprio is the lead
Leo_Caprio = movies[movies['actor_1_name']=="Leonardo DiCaprio"]
Leo_Caprio
```

Out[78]:

	director_name	num_critic_for_reviews	gross	genres	actor_1
97	Christopher Nolan	642.0	292.568851	Action|Adventure|Sci-Fi|Thriller	Le D
361	Martin Scorsese	352.0	132.373442	Crime|Drama|Thriller	Le D
296	Quentin Tarantino	765.0	162.804648	Drama|Western	Le D
308	Martin Scorsese	606.0	116.866727	Biography|Comedy|Crime|Drama	Le D
452	Martin Scorsese	490.0	127.968405	Mystery|Thriller	Le D
179	Alejandro G. Iñárritu	556.0	183.635922	Adventure|Drama|Thriller|Western	Le D
307	Edward Zwick	166.0	57.366262	Adventure|Drama|Thriller	Le D
911	Steven Spielberg	194.0	164.435221	Biography|Crime|Drama	Le D
26	James Cameron	315.0	658.672302	Drama|Romance	Le D
326	Martin Scorsese	233.0	77.679638	Crime|Drama	Le D
257	Martin Scorsese	267.0	102.608827	Biography|Drama	Le D
3476	Baz Luhrmann	490.0	144.812796	Drama|Romance	Le D
50	Baz Luhrmann	490.0	144.812796	Drama|Romance	Le D
1114	Sam Mendes	323.0	22.877808	Drama|Romance	Le D
641	Ridley Scott	238.0	39.380442	Action|Drama|Thriller	Le D
2757	Baz Luhrmann	106.0	46.338728	Drama|Romance	Le D
2067	Jerry Zaks	45.0	12.782508	Drama	Le D
990	Danny Boyle	118.0	39.778599	Adventure|Drama|Thriller	Le D
1453	Clint Eastwood	392.0	37.304950	Biography|Crime|Drama	Le D
1560	Sam Raimi	63.0	18.636537	Action|Thriller|Western	Le D
1422	Randall Wallace	83.0	56.876365	Action|Adventure	Le D

In [79]:

```
# Include all movies in which Brad_Pitt is the lead
Brad_Pitt =movies[movies["actor_1_name"]=='Brad Pitt']
Brad_Pitt
```

Out[79]:

	director_name	num_critic_for_reviews	gross	
683	David Fincher	315.0	37.023395	
2898	Tony Scott	122.0	12.281500	Action|Crime|Drama|Roman
400	Steven Soderbergh	186.0	183.405771	Crir
101	David Fincher	362.0	127.490802	Drama|Fantasy|
940	Neil Jordan	120.0	105.264608	Drama|Fanta
470	David Ayer	406.0	85.707116	Action|D
2204	Alejandro G. Iñárritu	285.0	34.300771	
1722	Andrew Dominik	273.0	3.904982	Biography|Crime|Drama|Histor
147	Wolfgang Petersen	220.0	133.228348	/
382	Tony Scott	142.0	0.026871	Action|Crir
611	Jean-Jacques Annaud	76.0	37.901509	Adventure|Biography|Drama|H
792	Patrick Gilmore	98.0	26.288320	Adventure|Animation|Comedy|Drama|Family
1490	Terrence Malick	584.0	13.303319	Dram
255	Doug Liman	233.0	186.336103	Action|Comedy|Crime|Roman
254	Steven Soderbergh	198.0	125.531634	Crir
2682	Andrew Dominik	414.0	14.938570	Crir
2333	Angelina Jolie Pitt	131.0	0.531009	Drama|

In [80]:

```
# Write your code for combining the three dataframes here
Combined=pd.concat([Meryl_Streep, Leo_Caprio, Brad_Pitt], axis=0)
Combined
```

Out[80]:

	director_name	num_critic_for_reviews	gross	genres	act
1925	Stephen Daldry	174.0	41.597830	Drama|Romance	M
1575	Sydney Pollack	66.0	87.100000	Biography|Drama|Romance	M
1674	Carl Franklin	64.0	23.209440	Drama	M
1204	Nora Ephron	252.0	94.125426	Biography|Drama|Romance	M
1408	David Frankel	208.0	124.732962	Comedy|Drama|Romance	M
3135	Robert Altman	211.0	20.338609	Comedy|Drama|Music	M
410	Nancy Meyers	187.0	112.703470	Comedy|Drama|Romance	M

In [81]:

```
# Write your code for grouping the combined dataframe here
ThreeActorsMovies=Combined.groupby('actor_1_name')
ThreeActorsMovies.head()
```

Out[81]:

	director_name	num_critic_for_reviews	gross	genres	acto
1925	Stephen Daldry	174.0	41.597830	Drama|Romance	M
1575	Sydney Pollack	66.0	87.100000	Biography|Drama|Romance	M
1674	Carl Franklin	64.0	23.209440	Drama	M
1204	Nora Ephron	252.0	94.125426	Biography|Drama|Romance	M
1408	David Frankel	208.0	124.732962	Comedy|Drama|Romance	M
97	Christopher Nolan	642.0	292.568851	Action|Adventure|Sci-Fi|Thriller	
361	Martin Scorsese	352.0	132.373442	Crime|Drama|Thriller	
296	Quentin Tarantino	765.0	162.804648	Drama|Western	
308	Martin Scorsese	606.0	116.866727	Biography|Comedy|Crime|Drama	
452	Martin Scorsese	490.0	127.968405	Mystery|Thriller	
683	David Fincher	315.0	37.023395	Drama	
2898	Tony Scott	122.0	12.281500	Action|Crime|Drama|Romance|Thriller	
400	Steven Soderbergh	186.0	183.405771	Crime|Thriller	
101	David Fincher	362.0	127.490802	Drama|Fantasy|Romance	
940	Neil Jordan	120.0	105.264608	Drama|Fantasy|Horror	

In [82]:

```
# Write the code for finding the mean of critic reviews and audience reviews here
Combined.groupby(['actor_1_name']) ['num_critic_for_reviews','num_user_for_reviews'].mean()
```

Out[82]:

actor_1_name	num_critic_for_reviews
Brad Pitt	245.000000
Leonardo DiCaprio	330.190476
Meryl Streep	181.454545

Checkpoint 6: `Leonardo` has aced both the lists!

In [83]:

```
# Write the code for calculating decade here
movies['decade']= (movies['title_year']//10)*10
movies.sort_values(by=['decade'], inplace=True, ascending=False)
movies
```

Out[83]:

	director_name	num_critic_for_reviews	gross	genres	actor_1
2834	Jon M. Chu	84.0	73.000942	Documentary|Music	Ra
4987	Lena Dunham	113.0	0.389804	Comedy|Drama|Romance	Lena D
37	Michael Bay	378.0	245.428137	Action|Adventure|Sci-Fi	Bing
2357	Jake Kasdan	286.0	100.292856	Comedy	Tim
1410	Seth Gordon	275.0	134.455175	Comedy|Crime	Jon F
...	...	...	...	...	
4449	William Cottrell	145.0	184.925485	Animation|Family|Fantasy|Musical	C
4786	Lloyd Bacon	65.0	2.300000	Comedy|Musical|Romance	Ginger
4812	Harry Beaumont	36.0	2.808000	Musical|Romance	Ani
4958	Harry F. Millarde	1.0	3.000000	Crime|Drama	Steph
2734	Fritz Lang	260.0	0.026435	Drama|Sci-Fi	Brigit

3821 rows × 17 columns

In [84]:

```
# Write your code for creating the data frame df_by_decade here
df_by_decade=movies.groupby(['decade'])['num_voted_users'].sum()
df_by_decade
```

Out[84]:

```
decade
1920.0       116392
1930.0       804839
1940.0       230838
1950.0       678336
1960.0      2983442
1970.0      8366245
1980.0     19883171
1990.0     69382588
2000.0    168789887
2010.0    118833749
Name: num_voted_users, dtype: int64
```

In [86]:

```
# Write your code for plotting number of voted users vs decade
sns.set(style="darkgrid")
df_by_decade.plot.bar('decade','num_voted_users')
```

Out[86]:

```
<AxesSubplot:xlabel='decade'>
```