PROJECT DESCRIPTION

- The project is about the movies rated by the IMDB and other Information.I will inspect the Data and will find out the Net Profit,Best Directors,Popular Genres,critic-favorite and Audience-favorite actors

APPROACH

- Importing Module
- Reading the Dataset into Pandas Dataframe
- We have divided the features into small segments and anlyzed segment-wise using a smaller dataframe containing only relevant categories.
- Data Cleaning, Missing Data Handling, Type casting are done segment-wise.
- Plots and percentages and means used to get the desired results

TECH STACK USED

- Jupyter Notebook V: 6.4.5
- Pandas V: 1.3.4
- Numpy V: 1.20.3
- Matplotlib V: 3.4.3
- Seaborn V: 0.11.2

INSIGHTS

- Movies with higher budgets are not necessarily profitable. when we plotted the graph of Profit V Budget we saw that movies which have higher budgets doesnot necessarily profitable. Around 1800 movies incurred losses or net Loss out of 3891 movies data
- James cameron's Avatar earned the highest profit and
- Frank Darabon's The Shawshank Redemption had the most IMDB rating of 9.3
- Top Foreign Language Film was The Good, the Bad and the Ugly by Sergio Leone(Director) in the language Italian
- Charles Chaplin was the best director with Imdb Mean of 8.6
- Family and Sci-Fi were the most Popular genres
- critic-favorite and audience-favorite actor was none other than Leonardo DiCaprio
- the dataframe consisted the movies from the ERA 1920 - 2010

RESULT

- Mainly I achivied while making the project was read the data carefully and each and every step should be given time to understand the result.It helped me understanding the data and was able to give rightfull insights about the data and helped me getting developed in my data analytical skills

DRIVE LINK

https://drive.google.com/drive/folders/1elXpPfhV3LddIEHh2tlqzz_YNJ69q0jA?usp=sharing
(https://drive.google.com/drive/folders/1elXpPfhV3LddIEHh2tlqzz_YNJ69q0jA?usp=sharing)

In [1]:
```
# Supress Warnings

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

In [2]:
```
# Import the numpy and pandas packages

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

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 [3]:
```
movies = pd.read_csv("IMDB_Movies.csv")
movies
```

Out[3]:

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	actor_1_facebook
0	Color	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore	1
1	Color	Gore Verbinski	302.0	169.0	563.0	1000.0	Orlando Bloom	40
2	Color	Sam Mendes	602.0	148.0	0.0	161.0	Rory Kinnear	11
3	Color	Christopher Nolan	813.0	164.0	22000.0	23000.0	Christian Bale	27
4	NaN	Doug Walker	NaN	NaN	131.0	NaN	Rob Walker	
...	...	...	...	...	...	...	...	
5038	Color	Scott Smith	1.0	87.0	2.0	318.0	Daphne Zuniga	
5039	Color	NaN	43.0	43.0	NaN	319.0	Valorie Curry	
5040	Color	Benjamin Roberds	13.0	76.0	0.0	0.0	Maxwell Moody	
5041	Color	Daniel Hsia	14.0	100.0	0.0	489.0	Daniel Henney	
5042	Color	Jon Gunn	43.0	90.0	16.0	16.0	Brian Herzlinger	

5043 rows × 28 columns

- **Subtask 1.2: Inspect the dataframe**

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

In [4]:
```
# Write your code for inspection here
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 [5]:
```
movies.columns
```

Out[5]:
```
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 [6]:
```
movies.dtypes
```

Out[6]:
```
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 [7]:
```
movies.shape
```

Out[7]:
```
(5043, 28)
```

In [8]:
```
movies.index
```

Out[8]:
```
RangeIndex(start=0, stop=5043, step=1)
```

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 [9]:
```
# Write your code for column-wise null count here
columnwise_null_count = movies.isnull().sum().sort_values(ascending=False)
columnwise_null_count
```

Out[9]:
```
gross                        884
budget                       492
aspect_ratio                 329
content_rating               303
plot_keywords                153
title_year                   108
director_name                104
director_facebook_likes      104
num_critic_for_reviews        50
actor_3_name                  23
actor_3_facebook_likes        23
num_user_for_reviews          20
color                         19
duration                      15
facenumber_in_poster          13
actor_2_name                  13
actor_2_facebook_likes        13
language                      12
actor_1_name                   7
actor_1_facebook_likes         7
country                        5
cast_total_facebook_likes      0
num_voted_users                0
movie_title                    0
movie_imdb_link                0
genres                         0
imdb_score                     0
movie_facebook_likes           0
dtype: int64
```

In [10]:
```
# Write your code for row-wise null count here
rowwise_null_count = movies.isnull().sum(axis=1).sort_values(ascending=False)
rowwise_null_count
```

Out[10]:
```
279     15
4       13
4945    11
2241    11
2342    10
        ..
1703     0
1702     0
1701     0
1700     0
5042     0
Length: 5043, dtype: int64
```

In [11]:
```
# Write your code for column-wise null percentages here
columnwise_null_percentages = ((columnwise_null_count/len(movies))*100).round(decimals=2)
columnwise_null_percentages #Rounding  off the percentages upto two decimal places.
```

Out[11]:
```
gross                        17.53
budget                        9.76
aspect_ratio                  6.52
content_rating                6.01
plot_keywords                 3.03
title_year                    2.14
director_name                 2.06
director_facebook_likes       2.06
num_critic_for_reviews        0.99
actor_3_name                  0.46
actor_3_facebook_likes        0.46
num_user_for_reviews          0.40
color                         0.38
duration                      0.30
facenumber_in_poster          0.26
actor_2_name                  0.26
actor_2_facebook_likes        0.26
language                      0.24
actor_1_name                  0.14
actor_1_facebook_likes        0.14
country                       0.10
cast_total_facebook_likes     0.00
num_voted_users               0.00
movie_title                   0.00
movie_imdb_link               0.00
genres                        0.00
imdb_score                    0.00
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 [12]:
```
# Write your code for dropping the columns here. It is advised to keep inspecting the dataframe after each set of oper

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 [13]:
```
movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5043 entries, 0 to 5042
Data columns (total 13 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   director_name           4939 non-null   object 
 1   num_critic_for_reviews  4993 non-null   float64
 2   gross                   4159 non-null   float64
 3   genres                  5043 non-null   object 
 4   actor_1_name            5036 non-null   object 
 5   movie_title             5043 non-null   object 
 6   num_voted_users         5043 non-null   int64  
 7   num_user_for_reviews    5023 non-null   object 
 8   language                5031 non-null   object 
 9   budget                  4551 non-null   float64
 10  title_year              4935 non-null   float64
 11  imdb_score              5043 non-null   float64
 12  movie_facebook_likes    5043 non-null   int64  
dtypes: float64(5), int64(2), object(6)
memory usage: 512.3+ KB
```

In [14]:
```
movies.columns
```

Out[14]:
```
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'],
      dtype='object')
```

- **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 [15]:
```
# Write your code for dropping the rows here
filter_rows = columnwise_null_percentages[columnwise_null_percentages > 5].index
filter_rows
```

Out[15]:
```
Index(['gross', 'budget', 'aspect_ratio', 'content_rating'], dtype='object')
```

In [16]:
```
movies.head()
```

Out[16]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted_users	num_user_for_re
0	James Cameron	723.0	760505847.0	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	Avatar	886204	
1	Gore Verbinski	302.0	309404152.0	Action|Adventure|Fantasy	Johnny Depp	Pirates of the Caribbean: At World's End	471220	
2	Sam Mendes	602.0	200074175.0	Action|Adventure|Thriller	Christoph Waltz	Spectre	275868	
3	Christopher Nolan	813.0	448130642.0	Action|Thriller	Tom Hardy	The Dark Knight Rises	1144337	
4	Doug Walker	NaN	NaN	Documentary	Doug Walker	Star Wars: Episode VII - The Force Awakens ...	8	

In [17]:
```
drop_rows = movies.dropna(subset = ['gross','budget']) # 'aspect_ratio', 'content_rating' are already dropped in subtc
drop_rows
```

Out[17]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted_users	num_u
0	James Cameron	723.0	760505847.0	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	Avatar	886204	
1	Gore Verbinski	302.0	309404152.0	Action|Adventure|Fantasy	Johnny Depp	Pirates of the Caribbean: At World's End	471220	
2	Sam Mendes	602.0	200074175.0	Action|Adventure|Thriller	Christoph Waltz	Spectre	275868	
3	Christopher Nolan	813.0	448130642.0	Action|Thriller	Tom Hardy	The Dark Knight Rises	1144337	
5	Andrew Stanton	462.0	73058679.0	Action|Adventure|Sci-Fi	Daryl Sabara	John Carter	212204	
...	...	...	...	...	...	...	...	
5033	Shane Carruth	143.0	424760.0	Drama|Sci-Fi|Thriller	Shane Carruth	Primer	72639	
5034	Neill Dela Llana	35.0	70071.0	Thriller	Ian Gamazon	Cavite	589	
5035	Robert Rodriguez	56.0	2040920.0	Action|Crime|Drama|Romance|Thriller	Carlos Gallardo	El Mariachi	52055	
5037	Edward Burns	14.0	4584.0	Comedy|Drama	Kerry Bishé	Newlyweds	1338	
5042	Jon Gunn	43.0	85222.0	Documentary	John August	My Date with Drew	4285	

3891 rows × 13 columns

In [18]:
```
movies
```

Out[18]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted_users	num_user_for
0	James Cameron	723.0	760505847.0	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	Avatar	886204	
1	Gore Verbinski	302.0	309404152.0	Action|Adventure|Fantasy	Johnny Depp	Pirates of the Caribbean: At World's End	471220	
2	Sam Mendes	602.0	200074175.0	Action|Adventure|Thriller	Christoph Waltz	Spectre	275868	
3	Christopher Nolan	813.0	448130642.0	Action|Thriller	Tom Hardy	The Dark Knight Rises	1144337	
4	Doug Walker	NaN	NaN	Documentary	Doug Walker	Star Wars: Episode VII - The Force Awakens ...	8	
...	...	...	...	...	...	...	...	
5038	Scott Smith	1.0	NaN	Comedy|Drama	Eric Mabius	Signed Sealed Delivered	629	
5039	NaN	43.0	NaN	Crime|Drama|Mystery|Thriller	Natalie Zea	The Following	73839	
5040	Benjamin Roberds	13.0	NaN	Drama|Horror|Thriller	Eva Boehnke	A Plague So Pleasant	38	
5041	Daniel Hsia	14.0	10443.0	Comedy|Drama|Romance	Alan Ruck	Shanghai Calling	1255	
5042	Jon Gunn	43.0	85222.0	Documentary	John August	My Date with Drew	4285	

5043 rows × 13 columns

- **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 [19]:
```
# Write your code for filling the NaN values in the 'language' column here

drop_rows['language'].fillna('English',inplace=True)
```

In [20]:
```
# A check that there are no null vales left in language column
drop_rows['language'].isnull().sum()
```

Out[20]:
```
0
```

In [21]:
```
movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5043 entries, 0 to 5042
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   director_name         4939 non-null   object 
 1   num_critic_for_reviews  4993 non-null   float64
 2   gross                 4159 non-null   float64
 3   genres                5043 non-null   object 
 4   actor_1_name          5036 non-null   object 
 5   movie_title           5043 non-null   object 
 6   num_voted_users       5043 non-null   int64  
 7   num_user_for_reviews  5023 non-null   object 
 8   language              5031 non-null   object 
 9   budget                4551 non-null   float64
 10  title_year            4935 non-null   float64
 11  imdb_score            5043 non-null   float64
 12  movie_facebook_likes  5043 non-null   int64  
dtypes: float64(5), int64(2), object(6)
memory usage: 512.3+ KB
```

- **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 [22]:
```
# Write your code for checking number of retained rows here
no_of_rows = len(drop_rows)
no_of_rows
```

Out[22]: 3891

In [23]:
```
percentage_of_rows = ((no_of_rows/len(movies))*100)
percentage_of_rows
```

Out[23]: 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 [24]:
```
movies = drop_rows # making movies dataframe as drops_rows
```

In [25]:
```
movies.budget=movies.budget/1000000
movies.gross=movies.gross/1000000
```

In [26]:
```
movies.head()
```

Out[26]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted_users	num_user_for_rev
0	James Cameron	723.0	760.505847	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	Avatar	886204	
1	Gore Verbinski	302.0	309.404152	Action|Adventure|Fantasy	Johnny Depp	Pirates of the Caribbean: At World's End	471220	
2	Sam Mendes	602.0	200.074175	Action|Adventure|Thriller	Christoph Waltz	Spectre	275868	
3	Christopher Nolan	813.0	448.130642	Action|Thriller	Tom Hardy	The Dark Knight Rises	1144337	
5	Andrew Stanton	462.0	73.058679	Action|Adventure|Sci-Fi	Daryl Sabara	John Carter	212204	

- **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 [27]:
```
movies["gross"].isnull().sum()
```

Out[27]: 0

In [28]:
```
movies["budget"].isnull().sum()
```

Out[28]: 0

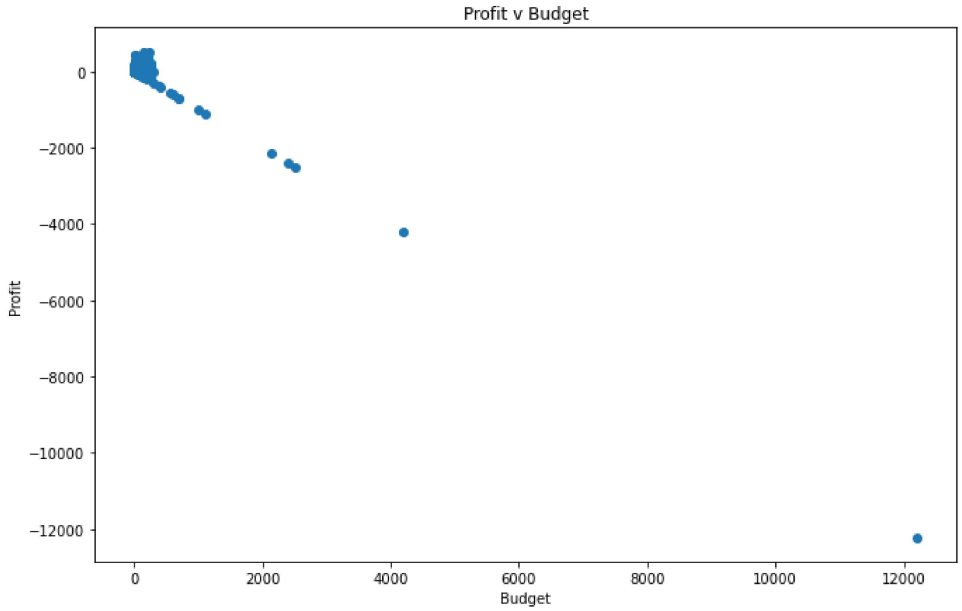
In [29]:
```
# Create the new column named 'profit' by subtracting the 'budget' column from the 'gross' column
movies["profit"] = movies["gross"]-movies["budget"]
```

In [30]:
```
# Sort the dataframe with the 'profit' column as reference using the 'sort_values' function. Make sure to set the argu
#'ascending' to 'False'
movies.sort_values(by="profit",ascending = False,inplace = True)
movies.head()
```

Out[30]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted_users	num_user_for_
0	James Cameron	723.0	760.505847	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	Avatar	886204	
29	Colin Trevorrow	644.0	652.177271	Action|Adventure|Sci-Fi|Thriller	Bryce Dallas Howard	Jurassic World	418214	
26	James Cameron	315.0	658.672302	Drama|Romance	Leonardo DiCaprio	Titanic	793059	
3024	George Lucas	282.0	460.935665	Action|Adventure|Fantasy|Sci-Fi	Harrison Ford	Star Wars: Episode IV - A New Hope	911097	
3080	Steven Spielberg	215.0	434.949459	Family|Sci-Fi	Henry Thomas	E.T. the Extra-Terrestrial	281842	

In [31]:
```
#Plot profit vs budget
plt.figure(figsize=[11,7])
plt.scatter(movies.budget,movies.profit)
plt.title("Profit v Budget")
plt.xlabel("Budget")
plt.ylabel("Profit")
plt.show()
#My Observation: Movies with higher budgets are not necessarily profitable
```



My Observation:

Movies with higher budgets are not necessarily profitable.
scatter plot tells a different story. You can notice that there are some movies with negative profit. Althoug
h good movies do incur losses, but there appear to be quite a few movie with losses. What can be the reason b
ehind this? Lets have a closer look at this by finding the movies with negative profit

In [32]:
```
#Find the movies with negative profit
neg_profit = movies[movies["profit"]<0]
neg_profit
```

Out[32]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted_users	nu
4311	Hunter Richards	34.0	0.012667	Drama|Romance	Jason Statham	London	19336	
4933	Michel Orion Scott	29.0	0.155984	Documentary	Temple Grandin	The Horse Boy	586	
5037	Edward Burns	14.0	0.004584	Comedy|Drama	Kerry Bishé	Newlyweds	1338	
2826	Martin McDonagh	401.0	14.989761	Comedy|Crime	Abbie Cornish	Seven Psychopaths	185845	
4283	John Cameron Mitchell	160.0	1.984378	Comedy|Drama|Romance	Sook-Yin Lee	Shortbus	27346	
...	...	...	...	...	...	...	...	
2334	Katsuhiro Ôtomo	105.0	0.410388	Action|Adventure|Animation|Family|Sci-Fi|Thriller	William Hootkins	Steamboy	13727	
	Hayao					Princess		

In [ ]:

In [ ]:

In [33]:
```
# Get the top 10 profitable movies by using position based indexing. Specify the rows till 10 (0-9)
top10=movies.iloc[0:10]
top10
```

Out[33]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted_users	nu
0	James Cameron	723.0	760.505847	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	Avatar	886204	
29	Colin Trevorrow	644.0	652.177271	Action|Adventure|Sci-Fi|Thriller	Bryce Dallas Howard	Jurassic World	418214	
26	James Cameron	315.0	658.672302	Drama|Romance	Leonardo DiCaprio	Titanic	793059	
3024	George Lucas	282.0	460.935665	Action|Adventure|Fantasy|Sci-Fi	Harrison Ford	Star Wars: Episode IV - A New Hope	911097	
3080	Steven Spielberg	215.0	434.949459	Family|Sci-Fi	Henry Thomas	E.T. the Extra-Terrestrial	281842	
794	Joss Whedon	703.0	623.279547	Action|Adventure|Sci-Fi	Chris Hemsworth	The Avengers	995415	
17	Joss Whedon	703.0	623.279547	Action|Adventure|Sci-Fi	Chris Hemsworth	The Avengers	995415	
509	Roger Allers	186.0	422.783777	Adventure|Animation|Drama|Family|Musical	Matthew Broderick	The Lion King	644348	
240	George Lucas	320.0	474.544677	Action|Adventure|Fantasy|Sci-Fi	Natalie Portman	Star Wars: Episode I - The Phantom Menace	534658	
66	Christopher Nolan	645.0	533.316061	Action|Crime|Drama|Thriller	Christian Bale	The Dark Knight	1676169	

- **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 [34]:
```
# Write your code for dropping duplicate values here
movies.drop_duplicates(inplace=True)
movies.head()
```

Out[34]:

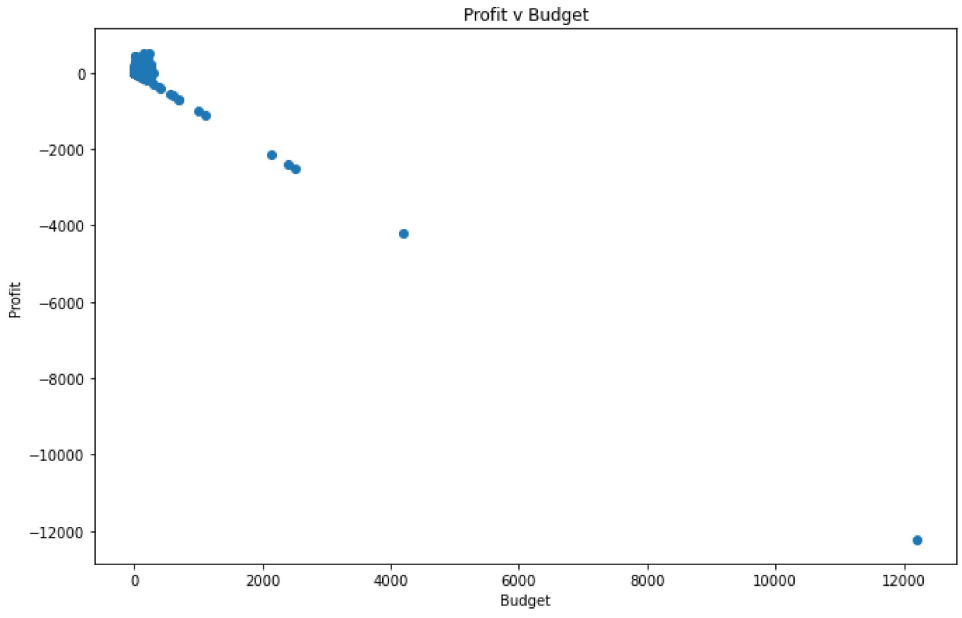
	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted_users	num_user_for_
0	James Cameron	723.0	760.505847	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	Avatar	886204	
29	Colin Trevorrow	644.0	652.177271	Action|Adventure|Sci-Fi|Thriller	Bryce Dallas Howard	Jurassic World	418214	
26	James Cameron	315.0	658.672302	Drama|Romance	Leonardo DiCaprio	Titanic	793059	
3024	George Lucas	282.0	460.935665	Action|Adventure|Fantasy|Sci-Fi	Harrison Ford	Star Wars: Episode IV - A New Hope	911097	
3080	Steven Spielberg	215.0	434.949459	Family|Sci-Fi	Henry Thomas	E.T. the Extra-Terrestrial	281842	

In [ ]:
```
```

Write code for repeating subtask 2 here

repeating

In [35]:
```
#Plot profit vs budget
plt.figure(figsize=[11,7])
plt.scatter(movies.budget,movies.profit)
plt.title("Profit v Budget")
plt.xlabel("Budget")
plt.ylabel("Profit")
plt.show()
#My Observation: Movies with higher budgets are not necessarily profitable
```



My Observation:

Movies with higher budgets are not necessarily profitable.
scatter plot tells a different story. You can notice that there are some movies with negative profit. Althoug
h good movies do incur losses, but there appear to be quite a few movie with losses. What can be the reason b
ehind this? Lets have a closer look at this by finding the movies with negative profit

In [36]:
```
#Find the movies with negative profit
neg_profit = movies[movies["profit"]<0]
neg_profit
```

Out[36]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted_users	num_u
4311	Hunter Richards	34.0	0.012667	Drama|Romance	Jason Statham	London	19336	
4933	Michel Orion Scott	29.0	0.155984	Documentary	Temple Grandin	The Horse Boy	586	
5037	Edward Burns	14.0	0.004584	Comedy|Drama	Kerry Bishé	Newlyweds	1338	
2826	Martin McDonagh	401.0	14.989761	Comedy|Crime	Abbie Cornish	Seven Psychopaths	185845	
4283	John Cameron Mitchell	160.0	1.984378	Comedy|Drama|Romance	Sook-Yin Lee	Shortbus	27346	
...	...	...	...	...	...	...	...	
2334	Katsuhiro Ôtomo	105.0	0.410388	Action|Adventure|Animation|Family|Sci-Fi|Thriller	William Hootkins	Steamboy	13727	
2323	Hayao Miyazaki	174.0	2.298191	Adventure|Animation|Fantasy	Minnie Driver	Princess Mononoke	221552	
3005	Lajos Koltai	73.0	0.195888	Drama|Romance|War	Marcell Nagy	Fateless	5603	
3859	Chan-wook Park	202.0	0.211667	Crime|Drama	Min-sik Choi	Lady Vengeance	53508	
2988	Joon-ho Bong	363.0	2.201412	Comedy|Drama|Horror|Sci-Fi	Doona Bae	The Host	68883	

1823 rows × 14 columns

In [37]:
```
top10=movies.iloc[0:10]
top10
```

Out[37]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted_users	nu
0	James Cameron	723.0	760.505847	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	Avatar	886204	
29	Colin Trevorrow	644.0	652.177271	Action|Adventure|Sci-Fi|Thriller	Bryce Dallas Howard	Jurassic World	418214	
26	James Cameron	315.0	658.672302	Drama|Romance	Leonardo DiCaprio	Titanic	793059	
3024	George Lucas	282.0	460.935665	Action|Adventure|Fantasy|Sci-Fi	Harrison Ford	Star Wars: Episode IV - A New Hope	911097	
3080	Steven Spielberg	215.0	434.949459	Family|Sci-Fi	Henry Thomas	E.T. the Extra-Terrestrial	281842	
794	Joss Whedon	703.0	623.279547	Action|Adventure|Sci-Fi	Chris Hemsworth	The Avengers	995415	
509	Roger Allers	186.0	422.783777	Adventure|Animation|Drama|Family|Musical	Matthew Broderick	The Lion King	644348	
240	George Lucas	320.0	474.544677	Action|Adventure|Fantasy|Sci-Fi	Natalie Portman	Star Wars: Episode I - The Phantom Menace	534658	
66	Christopher Nolan	645.0	533.316061	Action|Crime|Drama|Thriller	Christian Bale	The Dark Knight	1676169	
439	Gary Ross	673.0	407.999255	Adventure|Drama|Sci-Fi|Thriller	Jennifer Lawrence	The Hunger Games	701607	

FROM the top 10 dataframe,row 17 got removed and 439 got added

In [ ]:
```

```

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 [38]:
```
# Write your code for extracting the top 250 movies as per the IMDb score here. Make sure that you store it in a new d
# and name that dataframe as 'IMDb_Top_250'
IMDb_Top_250 = movies[movies['num_voted_users'] > 25000].sort_values(by='imdb_score',ascending=False).head(250)
IMDb_Top_250
```

Out[38]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted_users	num_user_f
1937	Frank Darabont	199.0	28.341469	Crime|Drama	Morgan Freeman	The Shawshank Redemption	1689764	
3466	Francis Ford Coppola	208.0	134.821952	Crime|Drama	Al Pacino	The Godfather	1155770	
2837	Francis Ford Coppola	149.0	57.300000	Crime|Drama	Robert De Niro	The Godfather: Part II	790926	
66	Christopher Nolan	645.0	533.316061	Action|Crime|Drama|Thriller	Christian Bale	The Dark Knight	1676169	
339	Peter Jackson	328.0	377.019252	Action|Adventure|Drama|Fantasy	Orlando Bloom	The Lord of the Rings: The Return of the King	1215718	
...	...	...	...	...	...	...	...	
4640	Cristian Mungiu	233.0	1.185783	Drama	Anamaria Marinca	4 Months, 3 Weeks and 2 Days	44763	
2492	John Carpenter	318.0	47.000000	Horror|Thriller	Jamie Lee Curtis	Halloween	157857	
4821	John Carpenter	318.0	47.000000	Horror|Thriller	Jamie Lee Curtis	Halloween	157863	
639	Michael Mann	209.0	28.965197	Biography|Drama|Thriller	Al Pacino	The Insider	133526	
3029	David O. Russell	410.0	93.571803	Biography|Drama|Sport	Christian Bale	The Fighter	275869	

250 rows × 14 columns

In [39]:
```
IMDb_Top_250['Rank'] = IMDb_Top_250['imdb_score'].rank(method='first',ascending=False)
IMDb_Top_250
```

Out[39]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted_users	num_user_f
1937	Frank Darabont	199.0	28.341469	Crime|Drama	Morgan Freeman	The Shawshank Redemption	1689764	
3466	Francis Ford Coppola	208.0	134.821952	Crime|Drama	Al Pacino	The Godfather	1155770	
2837	Francis Ford Coppola	149.0	57.300000	Crime|Drama	Robert De Niro	The Godfather: Part II	790926	
66	Christopher Nolan	645.0	533.316061	Action|Crime|Drama|Thriller	Christian Bale	The Dark Knight	1676169	
339	Peter Jackson	328.0	377.019252	Action|Adventure|Drama|Fantasy	Orlando Bloom	The Lord of the Rings: The Return of the King	1215718	
...	...	...	...	...	...	...	...	
4640	Cristian Mungiu	233.0	1.185783	Drama	Anamaria Marinca	4 Months, 3 Weeks and 2 Days	44763	
2492	John Carpenter	318.0	47.000000	Horror|Thriller	Jamie Lee Curtis	Halloween	157857	
4821	John Carpenter	318.0	47.000000	Horror|Thriller	Jamie Lee Curtis	Halloween	157863	
639	Michael Mann	209.0	28.965197	Biography|Drama|Thriller	Al Pacino	The Insider	133526	
3029	David O. Russell	410.0	93.571803	Biography|Drama|Sport	Christian Bale	The Fighter	275869	

250 rows × 15 columns

In [40]:
```
Top_Foreign_Lang_Film = IMDb_Top_250[IMDb_Top_250['language']!='English']
Top_Foreign_Lang_Film
```

Out[40]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_vo
4498	Sergio Leone	181.0	6.100000	Western	Clint Eastwood	The Good, the Bad and the Ugly	
4029	Fernando Meirelles	214.0	7.563397	Crime|Drama	Alice Braga	City of God	
4747	Akira Kurosawa	153.0	0.269061	Action|Adventure|Drama	Takashi Shimura	Seven Samurai	
2373	Hayao Miyazaki	246.0	10.049886	Adventure|Animation|Family|Fantasy	Bunta Sugawara	Spirited Away	
4921	Majid Majidi	46.0	0.925402	Drama|Family	Bahare Seddiqi	Children of Heaven	
4259	Florian Henckel von Donnersmarck	215.0	11.284657	Drama|Thriller	Sebastian Koch	The Lives of Others	
1329	S.S. Rajamouli	44.0	6.498000	Action|Adventure|Drama|Fantasy|War	Tamannaah Bhatia	Baahubali: The Beginning	
4659	Asghar Farhadi	354.0	7.098492	Drama|Mystery	Shahab Hosseini	A Separation	
1298	Jean-Pierre Jeunet	242.0	33.201661	Comedy|Romance	Mathieu Kassovitz	Amélie	
4105	Chan-wook Park	305.0	2.181290	Drama|Mystery|Thriller	Min-sik Choi	Oldboy	
2323	Hayao Miyazaki	174.0	2.298191	Adventure|Animation|Fantasy	Minnie Driver	Princess Mononoke	
2970	Wolfgang Petersen	96.0	11.433134	Adventure|Drama|Thriller|War	Jürgen Prochnow	Das Boot	
2734	Fritz Lang	260.0	0.026435	Drama|Sci-Fi	Brigitte Helm	Metropolis	
4033	Thomas Vinterberg	349.0	0.610968	Drama	Thomas Bo Larsen	The Hunt	
2829	Oliver Hirschbiegel	192.0	5.501940	Biography|Drama|History|War	Thomas Kretschmann	Downfall	
3550	Denis Villeneuve	226.0	6.857096	Drama|Mystery|War	Lubna Azabal	Incendies	
4000	Juan José Campanella	262.0	20.167424	Drama|Mystery|Thriller	Ricardo Darín	The Secret in Their Eyes	
2551	Guillermo del Toro	406.0	37.623143	Drama|Fantasy|War	Ivana Baquero	Pan's Labyrinth	
2047	Hayao Miyazaki	212.0	4.710455	Adventure|Animation|Family|Fantasy	Christian Bale	Howl's Moving Castle	
3553	José Padilha	142.0	0.008060	Action|Crime|Drama|Thriller	Wagner Moura	Elite Squad	
3423	Katsuhiro Ôtomo	150.0	0.439162	Action|Animation|Sci-Fi	Mitsuo Iwata	Akira	
2914	Je-kyu Kang	86.0	1.110186	Action|Drama|War	Min-sik Choi	Tae Guk Gi: The Brotherhood of War	
4461	Thomas Vinterberg	98.0	1.647780	Drama	Ulrich Thomsen	The Celebration	
4267	Alejandro G. Iñárritu	157.0	5.383834	Drama|Thriller	Adriana Barraza	Amores Perros	
2830	Alejandro Amenábar	157.0	2.086345	Biography|Drama|Romance	Belén Rueda	The Sea Inside	
4284	Ari Folman	231.0	2.283276	Animation|Biography|Documentary|Drama|History|War	Ari Folman	Waltz with Bashir	
3456	Vincent Paronnaud	242.0	4.443403	Animation|Biography|Drama|War	Catherine Deneuve	Persepolis	
3344	Karan Johar	210.0	4.018695	Adventure|Drama|Thriller	Shah Rukh Khan	My Name Is Khan	
4897	Sergio Leone	122.0	3.500000	Action|Drama|Western	Clint Eastwood	A Fistful of Dollars	
4144	Walter Salles	71.0	5.595428	Drama	Fernanda Montenegro	Central Station	
3264	Michael Haneke	447.0	0.225377	Drama|Romance	Isabelle Huppert	Amour	
2863	Clint Eastwood	251.0	13.753931	Drama|History|War	Yuki Matsuzaki	Letters from Iwo Jima	

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_vo
2605	Ang Lee	287.0	128.067808	Action|Drama|Romance	Chen Chang	Crouching Tiger, Hidden Dragon	
3510	Yash Chopra	29.0	2.921738	Drama|Musical|Romance	Shah Rukh Khan	Veer-Zaara	
4415	Fabián Bielinsky	94.0	1.221261	Crime|Drama|Thriller	Ricardo Darín	Nine Queens	
3677	Christophe Barratier	112.0	3.629758	Drama|Music	Jean-Baptiste Maunier	The Chorus	
2493	Yimou Zhang	283.0	0.084961	Action|Adventure|History	Jet Li	Hero	
4640	Cristian Mungiu	233.0	1.185783	Drama	Anamaria Marinca	4 Months, 3 Weeks and 2 Days	

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 [41]:
```
# Write your code for extracting the top 10 directors here
top10director = movies.groupby('director_name').imdb_score.mean().sort_values(ascending=False).head(10)
top10director
```

Out[41]:
```
director_name
Charles Chaplin          8.600000
Tony Kaye                8.600000
Alfred Hitchcock         8.500000
Ron Fricke               8.500000
Damien Chazelle          8.500000
Majid Majidi             8.500000
Sergio Leone             8.433333
Christopher Nolan        8.425000
S.S. Rajamouli           8.400000
Marius A. Markevicius    8.400000
Name: imdb_score, dtype: float64
```

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 [42]:
```
# Write your code for extracting the first two genres of each movie here
TempGenre = movies.genres.str.split('|',expand=True).iloc[:,0:2]
TempGenre.columns=['genre_1','genre_2']
TempGenre.genre_2.fillna(TempGenre.genre_1,inplace=True)
TempGenre
```

Out[42]:

	genre_1	genre_2
0	Action	Adventure
29	Action	Adventure
26	Drama	Romance
3024	Action	Adventure
3080	Family	Sci-Fi
...	...	...
2334	Action	Adventure
2323	Adventure	Animation
3005	Drama	Romance
3859	Crime	Drama
2988	Comedy	Drama

3856 rows × 2 columns

In [43]:
```
movies_by_segment =pd.concat([movies,TempGenre],axis=1)
movies_by_segment
```

Out[43]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted_users	num_u
0	James Cameron	723.0	760.505847	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	Avatar	886204	
29	Colin Trevorrow	644.0	652.177271	Action|Adventure|Sci-Fi|Thriller	Bryce Dallas Howard	Jurassic World	418214	
26	James Cameron	315.0	658.672302	Drama|Romance	Leonardo DiCaprio	Titanic	793059	
3024	George Lucas	282.0	460.935665	Action|Adventure|Fantasy|Sci-Fi	Harrison Ford	Star Wars: Episode IV - A New Hope	911097	
3080	Steven Spielberg	215.0	434.949459	Family|Sci-Fi	Henry Thomas	E.T. the Extra-Terrestrial	281842	
...	...	...	...	...	...	...	...	
2334	Katsuhiro Ôtomo	105.0	0.410388	Action|Adventure|Animation|Family|Sci-Fi|Thriller	William Hootkins	Steamboy	13727	
2323	Hayao Miyazaki	174.0	2.298191	Adventure|Animation|Fantasy	Minnie Driver	Princess Mononoke	221552	
3005	Lajos Koltai	73.0	0.195888	Drama|Romance|War	Marcell Nagy	Fateless	5603	
3859	Chan-wook Park	202.0	0.211667	Crime|Drama	Min-sik Choi	Lady Vengeance	53508	
2988	Joon-ho Bong	363.0	2.201412	Comedy|Drama|Horror|Sci-Fi	Doona Bae	The Host	68883	

3856 rows × 16 columns

In [44]:
```
PopGenre = movies_by_segment.groupby(['genre_1','genre_2']).gross.mean().sort_values(ascending=False).head(5)
PopGenre
```

Out[44]:
```
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
```

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 [45]:
```
# Write your code for creating three new dataframes here
Meryl_Streep = movies[movies['actor_1_name']=='Meryl Streep']
```

In [46]:
```
Leo_Caprio = movies[movies['actor_1_name']=='Leonardo DiCaprio']
```

In [47]:
```
Brad_Pitt = movies[movies['actor_1_name']=='Brad Pitt']
```

In [48]:
```
# Write your code for combining the three dataframes here
Combined = Meryl_Streep.append([Leo_Caprio,Brad_Pitt])
Combined
```

Out[48]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_v
1408	David Frankel	208.0	124.732962	Comedy|Drama|Romance	Meryl Streep	The Devil Wears Prada	
1575	Sydney Pollack	66.0	87.100000	Biography|Drama|Romance	Meryl Streep	Out of Africa	
1204	Nora Ephron	252.0	94.125426	Biography|Drama|Romance	Meryl Streep	Julie & Julia	
1618	David Frankel	234.0	63.536011	Comedy|Drama|Romance	Meryl Streep	Hope Springs	
410	Nancy Meyers	187.0	112.703470	Comedy|Drama|Romance	Meryl Streep	It's Complicated	
2781	Phyllida Lloyd	331.0	29.959436	Biography|Drama|History	Meryl Streep	The Iron Lady	
1925	Stephen Daldry	174.0	41.597830	Drama|Romance	Meryl Streep	The Hours	
3135	Robert Altman	211.0	20.338609	Comedy|Drama|Music	Meryl Streep	A Prairie Home Companion	

In [49]:
```
# Write your code for grouping the combined dataframe here
Combined.groupby('actor_1_name')
Combined
```

Out[49]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted
1408	David Frankel	208.0	124.732962	Comedy|Drama|Romance	Meryl Streep	The Devil Wears Prada	
1575	Sydney Pollack	66.0	87.100000	Biography|Drama|Romance	Meryl Streep	Out of Africa	
1204	Nora Ephron	252.0	94.125426	Biography|Drama|Romance	Meryl Streep	Julie & Julia	
1618	David Frankel	234.0	63.536011	Comedy|Drama|Romance	Meryl Streep	Hope Springs	
410	Nancy Meyers	187.0	112.703470	Comedy|Drama|Romance	Meryl Streep	It's Complicated	
2781	Phyllida Lloyd	331.0	29.959436	Biography|Drama|History	Meryl Streep	The Iron Lady	
1925	Stephen Daldry	174.0	41.597830	Drama|Romance	Meryl Streep	The Hours	
3135	Robert Altman	211.0	20.338609	Comedy|Drama|Music	Meryl Streep	A Prairie Home Companion	
1106	Curtis Hanson	42.0	46.815748	Action|Adventure|Crime|Thriller	Meryl Streep	The River Wild	
1674	Carl Franklin	64.0	23.209440	Drama	Meryl Streep	One True Thing	
1483	Robert Redford	227.0	14.998070	Drama|Thriller|War	Meryl Streep	Lions for Lambs	
26	James Cameron	315.0	658.672302	Drama|Romance	Leonardo DiCaprio	Titanic	
97	Christopher Nolan	642.0	292.568851	Action|Adventure|Sci-Fi|Thriller	Leonardo DiCaprio	Inception	1
911	Steven Spielberg	194.0	164.435221	Biography|Crime|Drama	Leonardo DiCaprio	Catch Me If You Can	
296	Quentin Tarantino	765.0	162.804648	Drama|Western	Leonardo DiCaprio	Django Unchained	
179	Alejandro G. Iñárritu	556.0	183.635922	Adventure|Drama|Thriller|Western	Leonardo DiCaprio	The Revenant	
452	Martin Scorsese	490.0	127.968405	Mystery|Thriller	Leonardo DiCaprio	Shutter Island	
361	Martin Scorsese	352.0	132.373442	Crime|Drama|Thriller	Leonardo DiCaprio	The Departed	
50	Baz Luhrmann	490.0	144.812796	Drama|Romance	Leonardo DiCaprio	The Great Gatsby	
3476	Baz Luhrmann	490.0	144.812796	Drama|Romance	Leonardo DiCaprio	The Great Gatsby	
2757	Baz Luhrmann	106.0	46.338728	Drama|Romance	Leonardo DiCaprio	Romeo + Juliet	
1422	Randall Wallace	83.0	56.876365	Action|Adventure	Leonardo DiCaprio	The Man in the Iron Mask	
308	Martin Scorsese	606.0	116.866727	Biography|Comedy|Crime|Drama	Leonardo DiCaprio	The Wolf of Wall Street	
1453	Clint Eastwood	392.0	37.304950	Biography|Crime|Drama	Leonardo DiCaprio	J. Edgar	
257	Martin Scorsese	267.0	102.608827	Biography|Drama	Leonardo DiCaprio	The Aviator	
2067	Jerry Zaks	45.0	12.782508	Drama	Leonardo DiCaprio	Marvin's Room	
990	Danny Boyle	118.0	39.778599	Adventure|Drama|Thriller	Leonardo DiCaprio	The Beach	
1114	Sam Mendes	323.0	22.877808	Drama|Romance	Leonardo DiCaprio	Revolutionary Road	
1560	Sam Raimi	63.0	18.636537	Action|Thriller|Western	Leonardo DiCaprio	The Quick and the Dead	
326	Martin Scorsese	233.0	77.679638	Crime|Drama	Leonardo DiCaprio	Gangs of New York	
641	Ridley Scott	238.0	39.380442	Action|Drama|Thriller	Leonardo DiCaprio	Body of Lies	
307	Edward Zwick	166.0	57.366262	Adventure|Drama|Thriller	Leonardo DiCaprio	Blood Diamond	
400	Steven Soderbergh	186.0	183.405771	Crime|Thriller	Brad Pitt	Ocean's Eleven	
255	Doug Liman	233.0	186.336103	Action|Comedy|Crime|Romance|Thriller	Brad Pitt	Mr. & Mrs. Smith	
940	Neil Jordan	120.0	105.264608	Drama|Fantasy|Horror	Brad Pitt	Interview with the Vampire: The Vampire Chroni...	

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted
470	David Ayer	406.0	85.707116	Action|Drama|War	Brad Pitt	Fury	
254	Steven Soderbergh	198.0	125.531634	Crime|Thriller	Brad Pitt	Ocean's Twelve	
2204	Alejandro G. Iñárritu	285.0	34.300771	Drama	Brad Pitt	Babel	
2682	Andrew Dominik	414.0	14.938570	Crime|Thriller	Brad Pitt	Killing Them Softly	
2898	Tony Scott	122.0	12.281500	Action|Crime|Drama|Romance|Thriller	Brad Pitt	True Romance	
2333	Angelina Jolie Pitt	131.0	0.531009	Drama|Romance	Brad Pitt	By the Sea	
1490	Terrence Malick	584.0	13.303319	Drama|Fantasy	Brad Pitt	The Tree of Life	
101	David Fincher	362.0	127.490802	Drama|Fantasy|Romance	Brad Pitt	The Curious Case of Benjamin Button	
683	David Fincher	315.0	37.023395	Drama	Brad Pitt	Fight Club	1
1722	Andrew Dominik	273.0	3.904982	Biography|Crime|Drama|History|Western	Brad Pitt	The Assassination of Jesse James by the Coward...	
611	Jean-Jacques Annaud	76.0	37.901509	Adventure|Biography|Drama|History|War	Brad Pitt	Seven Years in Tibet	
792	Patrick Gilmore	98.0	26.288320	Adventure|Animation|Comedy|Drama|Family|Fantas...	Brad Pitt	Sinbad: Legend of the Seven Seas	
147	Wolfgang Petersen	220.0	133.228348	Adventure	Brad Pitt	Troy	
382	Tony Scott	142.0	0.026871	Action|Crime|Thriller	Brad Pitt	Spy Game	

In [50]:
```
Combined.num_user_for_reviews = Combined.num_user_for_reviews.astype('int')
Combined.num_user_for_reviews
```

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

In [51]:
```
Combined.groupby('actor_1_name').num_user_for_reviews.mean()
```

Out[51]:
```
actor_1_name
Brad Pitt            742.352941
Leonardo DiCaprio    914.476190
Meryl Streep         297.181818
Name: num_user_for_reviews, dtype: float64
```

In [52]:
```
Combined.groupby('actor_1_name').num_critic_for_reviews.mean()
```

Out[52]:
```
actor_1_name
Brad Pitt            245.000000
Leonardo DiCaprio    330.190476
Meryl Streep         181.454545
Name: num_critic_for_reviews, dtype: float64
```

In [53]:
```
# 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[53]:

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

Checkpoint 6: Leonardo has aced both the lists!

In [57]:
```
# Write the code for calculating decade here
df_by_decade=movies.copy(deep=True)
df_by_decade['decade']=df_by_decade['title_year'].apply(lambda x:10*(int(x/10)))
df_by_decade
```

Out[57]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted_users	num_u
0	James Cameron	723.0	760.505847	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	Avatar	886204	
29	Colin Trevorrow	644.0	652.177271	Action|Adventure|Sci-Fi|Thriller	Bryce Dallas Howard	Jurassic World	418214	
26	James Cameron	315.0	658.672302	Drama|Romance	Leonardo DiCaprio	Titanic	793059	
3024	George Lucas	282.0	460.935665	Action|Adventure|Fantasy|Sci-Fi	Harrison Ford	Star Wars: Episode IV - A New Hope	911097	
3080	Steven Spielberg	215.0	434.949459	Family|Sci-Fi	Henry Thomas	E.T. the Extra-Terrestrial	281842	
...	...	...	...	...	...	...	...	
2334	Katsuhiro Ôtomo	105.0	0.410388	Action|Adventure|Animation|Family|Sci-Fi|Thriller	William Hootkins	Steamboy	13727	
2323	Hayao Miyazaki	174.0	2.298191	Adventure|Animation|Fantasy	Minnie Driver	Princess Mononoke	221552	
3005	Lajos Koltai	73.0	0.195888	Drama|Romance|War	Marcell Nagy	Fateless	5603	
3859	Chan-wook Park	202.0	0.211667	Crime|Drama	Min-sik Choi	Lady Vengeance	53508	
2988	Joon-ho Bong	363.0	2.201412	Comedy|Drama|Horror|Sci-Fi	Doona Bae	The Host	68883	

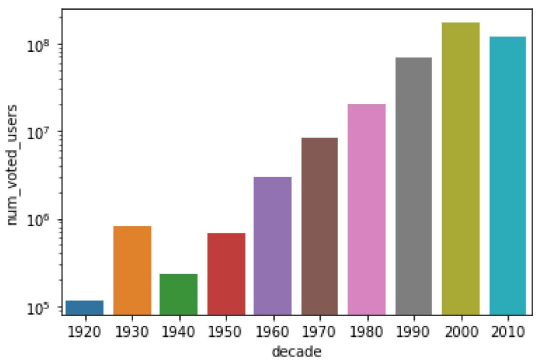
3856 rows × 15 columns

In [61]:
```
# Write your code for creating the data frame df_by_decade here
df_by_decade=df_by_decade.groupby('decade',as_index=False)['num_voted_users'].sum().sort_values(by="decade")
df_by_decade
```

Out[61]:

	decade	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	170908676
9	2010	120640994

In [60]:
```
# Write your code for plotting number of voted users vs decade
ax=sns.barplot(x='decade',y='num_voted_users',data=df_by_decade)
plt.yscale('log')
plt.ylabel("num_voted_users")
plt.xlabel("decade")
plt.show()
```



In [ ]: