In [ ]:
```
# Supress Warnings

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

In [1]:
```
# 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 [2]:
```
movies = pd.read_csv(r"F:\stige projects\pandas stige\imdb movie rating\IMDB_Movies.csv
movies
```

Out[2]:

| | color | director_name | num_critic_for_reviews | duration | director_facebook_likes | actor_3_facebook_li |
|---|---|---|---|---|---|---|
| **0** | Color | James Cameron | 723.0 | 178.0 | 0.0 | 8 |
| **1** | Color | Gore Verbinski | 302.0 | 169.0 | 563.0 | 10 |
| **2** | Color | Sam Mendes | 602.0 | 148.0 | 0.0 | 1 |
| **3** | Color | Christopher Nolan | 813.0 | 164.0 | 22000.0 | 230 |
| **4** | NaN | Doug Walker | NaN | NaN | 131.0 | N |
| **...** | ... | ... | ... | ... | ... | |
| **5038** | Color | Scott Smith | 1.0 | 87.0 | 2.0 | 3 |
| **5039** | Color | NaN | 43.0 | 43.0 | NaN | 3 |
| **5040** | Color | Benjamin Roberds | 13.0 | 76.0 | 0.0 | |
| **5041** | Color | Daniel Hsia | 14.0 | 100.0 | 0.0 | 4 |
| **5042** | Color | Jon Gunn | 43.0 | 90.0 | 16.0 | |

5043 rows × 28 columns

- ### Subtask 1.2: Inspect the dataframe

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

In [4]:
```
print('Columns')
print(movies.columns)
print('Shapes')
print(movies.shape)
print('Variable_types')
print(movies.dtypes)
```

```
Columns
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')
Shapes
(5043, 28)
Variable_types
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
```

# 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 [5]:
```
# Write your code for column-wise null count here

print(movies.isnull().sum(axis = 0))
```

```
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 [6]:
```
# Write your code for row-wise null count here

print(movies.isnull().sum(axis = 1))
```

```
0        0
1        0
2        0
3        0
4       13
       ..
5038     4
5039     5
5040     4
5041     2
5042     0
Length: 5043, dtype: int64
```

In [9]:
```
# Write your code for column-wise null percentages here
movies.isnull().sum(axis=0).sort_values(ascending=False)/len(movies)*100
```

Out[9]:
```
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
cast_total_facebook_likes   0.000000
num_voted_users             0.000000
movie_title                 0.000000
movie_imdb_link             0.000000
genres                      0.000000
imdb_score                  0.000000
movie_facebook_likes        0.000000
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 [18]:

```
# Write your code for dropping the columns here. It is advised to keep inspecting the d
movies = movies.drop(['color','director_facebook_likes','actor_1_facebook_likes',
                      'actor_2_facebook_likes','actor_3_facebook_likes','actor_2_na
                     'duration','facenumber_in_poster','content_rating','movie_imdb
movies_new
```

Out[18]:

| | director_name | num_critic_for_reviews | gross | genres | actor_1_name | mc |
|---|---|---|---|---|---|---|
| **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 | Ca A |
| **2** | Sam Mendes | 602.0 | 200074175.0 | Action\|Adventure\|Thriller | Christoph Waltz | |
| **3** | Christopher Nolan | 813.0 | 448130642.0 | Action\|Thriller | Tom Hardy | |
| **4** | Doug Walker | NaN | NaN | Documentary | Doug Walker | S Ep - T Aw |
| **...** | ... | ... | ... | ... | ... | |
| **5038** | Scott Smith | 1.0 | NaN | Comedy\|Drama | Eric Mabius | |
| **5039** | NaN | 43.0 | NaN | Crime\|Drama\|Mystery\|Thriller | Natalie Zea | F |
| **5040** | Benjamin Roberds | 13.0 | NaN | Drama\|Horror\|Thriller | Eva Boehnke | So |
| **5041** | Daniel Hsia | 14.0 | 10443.0 | Comedy\|Drama\|Romance | Alan Ruck | |
| **5042** | Jon Gunn | 43.0 | 85222.0 | Documentary | John August | w |

5043 rows × 14 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 [19]:
```
round(movies.isnull().sum().sort_values(ascending=False)/len(movies)*100,2)
```

Out[19]:
```
actor_1_name          0.08
language              0.08
```

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

In [20]:
```
movies = movies[movies['gross'].notnull()]
movies = movies[movies['budget'].notnull()]
```

In [21]:
```
round(movies.isnull().sum().sort_values(ascending=False)/len(movies)*100,2)
```

Out[21]:
```
actor_1_name              0.08
language                  0.08
num_critic_for_reviews    0.03
director_name             0.00
gross                     0.00
genres                    0.00
movie_title               0.00
num_voted_users           0.00
num_user_for_reviews      0.00
country                   0.00
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 [22]:
```
# Write your code for filling the NaN values in the 'Language' column here
round(movies.isnull().sum().sort_values(ascending=False)/len(movies)*100,2)
```

Out[22]:
```
actor_1_name              0.08
language                  0.08
num_critic_for_reviews    0.03
director_name             0.00
gross                     0.00
genres                    0.00
movie_title               0.00
num_voted_users           0.00
num_user_for_reviews      0.00
country                   0.00
budget                    0.00
title_year                0.00
imdb_score                0.00
```

```
movie_facebook_likes      0.00
dtype: float64
```

- ### 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 [23]:
```
# Write your code for checking number of retained rows here
movies.groupby('language').language.count().sort_values(ascending=False)
```

Out[23]:
```
language
English       3707
French          37
Spanish         26
Mandarin        15
German          13
Japanese        12
Hindi           10
Cantonese        8
Italian          7
Portuguese       5
Korean           5
Norwegian        4
Dutch            3
Persian          3
Danish           3
Hebrew           3
Thai             3
Indonesian       2
Aboriginal       2
Dari             2
Czech            1
Vietnamese       1
Aramaic          1
Telugu           1
Swedish          1
Bosnian          1
Russian          1
Romanian         1
Arabic           1
Icelandic        1
Dzongkha         1
None             1
Mongolian        1
Maya             1
Filipino         1
Hungarian        1
Kazakh           1
Zulu             1
Name: language, dtype: int64
```

In [24]:
```
movies.language.describe()
```

Out[24]:
```
count        3888
unique         38
```

```
top       English
freq         3707
Name: language, dtype: object
```

In [26]:
```
movies.language = movies.language.fillna('English')
```

In [27]:
```
round(movies.isnull().sum().sort_values(ascending=False)/len(movies)*100,2)
```

Out[27]:
```
actor_1_name              0.08
num_critic_for_reviews    0.03
director_name             0.00
gross                     0.00
genres                    0.00
movie_title               0.00
num_voted_users           0.00
num_user_for_reviews      0.00
language                  0.00
country                   0.00
budget                    0.00
title_year                0.00
imdb_score                0.00
movie_facebook_likes      0.00
dtype: float64
```

**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 [34]:
```
# Write your code for unit conversion here
movies['budget'] = movies['budget']/1000000
movies['gross'] = movies['gross']/1000000
movies
```

Out[34]:

| | director_name | num_critic_for_reviews | gross | genres | actor_1_name |
|---|---|---|---|---|---|
| **0** | James Cameron | 723.0 | 7.605058e-28 | Action\|Adventure\|Fantasy\|Sci-Fi | CCH Pounde |
| **1** | Gore Verbinski | 302.0 | 3.094042e-28 | Action\|Adventure\|Fantasy | Johnny Depp |
| **2** | Sam Mendes | 602.0 | 2.000742e-28 | Action\|Adventure\|Thriller | Christoph Walt |
| **3** | Christopher Nolan | 813.0 | 4.481306e-28 | Action\|Thriller | Tom Hardy |

| | director_name | num_critic_for_reviews | gross | genres | actor_1_name |
|---|---|---|---|---|---|
| **5** | Andrew Stanton | 462.0 | 7.305868e-29 | Action\|Adventure\|Sci-Fi | Daryl Sabara |
| **...** | ... | ... | ... | ... | ... |
| **5033** | Shane Carruth | 143.0 | 4.247600e-31 | Drama\|Sci-Fi\|Thriller | Shane Carruth |
| **5034** | Neill Dela Llana | 35.0 | 7.007100e-32 | Thriller | Ian Gamazon |
| **5035** | Robert Rodriguez | 56.0 | 2.040920e-30 | Action\|Crime\|Drama\|Romance\|Thriller | Carlos Gallardo |
| **5037** | Edward Burns | 14.0 | 4.584000e-33 | Comedy\|Drama | Kerry Bishe |
| **5042** | Jon Gunn | 43.0 | 8.522200e-32 | Documentary | John Augus |

3891 rows × 15 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 [37]:
```
# Write your code for creating the profit column here
movies['profit'] = movies['gross'] - movies['budget']
movies
```

Out[37]:

| | director_name | num_critic_for_reviews | gross | genres | actor_1_name |
|---|---|---|---|---|---|
| **0** | James Cameron | 723.0 | 7.605058e-28 | Action\|Adventure\|Fantasy\|Sci-Fi | CCH Pounde |
| **1** | Gore Verbinski | 302.0 | 3.094042e-28 | Action\|Adventure\|Fantasy | Johnny Depp |
| **2** | Sam Mendes | 602.0 | 2.000742e-28 | Action\|Adventure\|Thriller | Christoph Walt |
| **3** | Christopher Nolan | 813.0 | 4.481306e-28 | Action\|Thriller | Tom Hardy |

| | director_name | num_critic_for_reviews | gross | genres | actor_1_name |
|---|---|---|---|---|---|
| **5** | Andrew Stanton | 462.0 | 7.305868e-29 | Action\|Adventure\|Sci-Fi | Daryl Sabara |
| **...** | ... | ... | ... | ... | ... |
| **5033** | Shane Carruth | 143.0 | 4.247600e-31 | Drama\|Sci-Fi\|Thriller | Shane Carruth |
| **5034** | Neill Dela Llana | 35.0 | 7.007100e-32 | Thriller | Ian Gamazon |
| **5035** | Robert Rodriguez | 56.0 | 2.040920e-30 | Action\|Crime\|Drama\|Romance\|Thriller | Carlos Gallardo |
| **5037** | Edward Burns | 14.0 | 4.584000e-33 | Comedy\|Drama | Kerry Bishe |
| **5042** | Jon Gunn | 43.0 | 8.522200e-32 | Documentary | John Augus |

3891 rows × 15 columns

In [38]:
```
# Write your code for sorting the dataframe here
movies.sort_values(by='profit',ascending=False)
```

Out[38]:

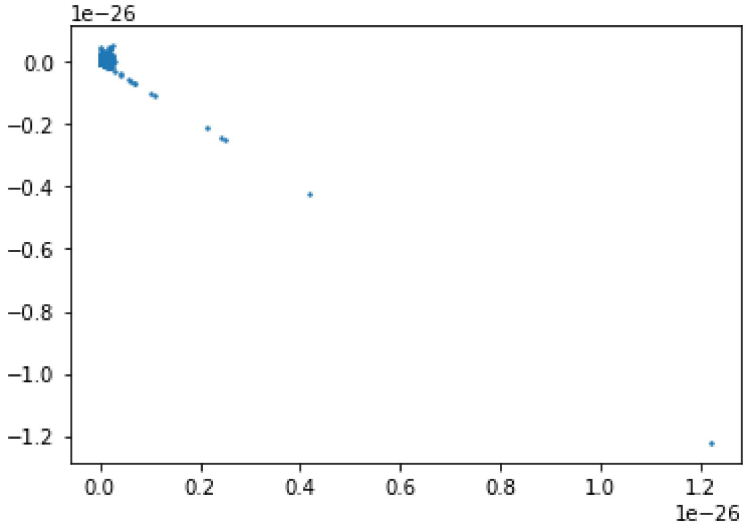
| | director_name | num_critic_for_reviews | gross | genres | actor_1_nam |
|---|---|---|---|---|---|
| **0** | James Cameron | 723.0 | 7.605058e-28 | Action\|Adventure\|Fantasy\|Sci-Fi | CCH Pounc |
| **29** | Colin Trevorrow | 644.0 | 6.521773e-28 | Action\|Adventure\|Sci-Fi\|Thriller | Bryce Dal Howa |
| **26** | James Cameron | 315.0 | 6.586723e-28 | Drama\|Romance | Leonar DiCap |
| **3024** | George Lucas | 282.0 | 4.609357e-28 | Action\|Adventure\|Fantasy\|Sci-Fi | Harrison Fc |
| **3080** | Steven Spielberg | 215.0 | 4.349495e-28 | Family\|Sci-Fi | Henry Thon |
| **...** | ... | ... | ... | ... | |
| **2334** | Katsuhiro Ôtomo | 105.0 | 4.103880e-31 | Action\|Adventure\|Animation\|Family\|Sci-Fi\|Thriller | Willi Hootk |
| **2323** | Hayao Miyazaki | 174.0 | 2.298191e-30 | Adventure\|Animation\|Fantasy | Minnie Dri |
| **3005** | Lajos Koltai | 73.0 | 1.958880e-31 | Drama\|Romance\|War | Marcell Na |
| **3859** | Chan-wook Park | 202.0 | 2.116670e-31 | Crime\|Drama | Min-sik Cl |

| | director_name | num_critic_for_reviews | gross | genres | actor_1_nai |
|---|---|---|---|---|---|
| **2988** | Joon-ho Bong | 363.0 | 2.201412e-30 | Comedy\|Drama\|Horror\|Sci-Fi | Doona E |

3891 rows × 15 columns

In [39]:
```
# Write code for profit vs budget plot here
plt.scatter(movies.budget,movies.profit,s=2)
plt.figure(figsize=(20,20))
plt.show
```

Out[39]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```




```
<Figure size 1440x1440 with 0 Axes>
```

In [40]:
```
top10 = movies.sort_values(by='profit',ascending=False).head(10)
top10
```

Out[40]:

| | director_name | num_critic_for_reviews | gross | genres | actor_1 |
|---|---|---|---|---|---|
| **0** | James Cameron | 723.0 | 7.605058e-28 | Action\|Adventure\|Fantasy\|Sci-Fi | CCH P |
| **29** | Colin Trevorrow | 644.0 | 6.521773e-28 | Action\|Adventure\|Sci-Fi\|Thriller | Bryce H |
| **26** | James Cameron | 315.0 | 6.586723e-28 | Drama\|Romance | Le Di |
| **3024** | George Lucas | 282.0 | 4.609357e-28 | Action\|Adventure\|Fantasy\|Sci-Fi | Harrisc |
| **3080** | Steven Spielberg | 215.0 | 4.349495e-28 | Family\|Sci-Fi | Henry T |

| | director_name | num_critic_for_reviews | gross | genres | actor_1 |
|---|---|---|---|---|---|
| **794** | Joss Whedon | 703.0 | 6.232795e-28 | Action\|Adventure\|Sci-Fi | Hem |
| **17** | Joss Whedon | 703.0 | 6.232795e-28 | Action\|Adventure\|Sci-Fi | Hem |
| **509** | Roger Allers | 186.0 | 4.227838e-28 | Adventure\|Animation\|Drama\|Family\|Musical | M Brc |
| **240** | George Lucas | 320.0 | 4.745447e-28 | Action\|Adventure\|Fantasy\|Sci-Fi | Pc |
| **66** | Christopher Nolan | 645.0 | 5.333161e-28 | Action\|Crime\|Drama\|Thriller | Christia |

- ### 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 [41]:
```
# Write your code for dropping duplicate values here
movies.drop_duplicates(keep='first',inplace=True)
```

In [42]:
```
# Write code for repeating subtask 2 here
movies
```

Out[42]:

| | director_name | num_critic_for_reviews | gross | genres | actor_1_name |
|---|---|---|---|---|---|
| **0** | James Cameron | 723.0 | 7.605058e-28 | Action\|Adventure\|Fantasy\|Sci-Fi | CCH Pounde |
| **1** | Gore Verbinski | 302.0 | 3.094042e-28 | Action\|Adventure\|Fantasy | Johnny Depp |
| **2** | Sam Mendes | 602.0 | 2.000742e-28 | Action\|Adventure\|Thriller | Christopł Walt: |
| **3** | Christopher Nolan | 813.0 | 4.481306e-28 | Action\|Thriller | Tom Hardy |
| **5** | Andrew Stanton | 462.0 | 7.305868e-29 | Action\|Adventure\|Sci-Fi | Daryl Sabara |
| **...** | ... | ... | ... | ... | ... |

| | director_name | num_critic_for_reviews | gross | genres | actor_1_name |
|---|---|---|---|---|---|
| **5033** | Shane Carruth | 143.0 | 4.247600e-31 | Drama\|Sci-Fi\|Thriller | Shane Carruth |
| **5034** | Neill Dela Llana | 35.0 | 7.007100e-32 | Thriller | Ian Gamazor |
| **5035** | Robert Rodriguez | 56.0 | 2.040920e-30 | Action\|Crime\|Drama\|Romance\|Thriller | Carlos Gallardo |
| **5037** | Edward Burns | 14.0 | 4.584000e-33 | Comedy\|Drama | Kerry Bishe |
| **5042** | Jon Gunn | 43.0 | 8.522200e-32 | Documentary | John Augus |

3856 rows × 15 columns

**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 [43]:
```
# Write your code for extracting the top 250 movies as per the IMDb score here. Make su
# and name that dataframe as 'IMDb_Top_250'
IMDb_Top_250 = movies[movies['num_voted_users'] > 25000].sort_values(by='imdb_score',as
IMDb_Top_250
```

Out[43]:

| | director_name | num_critic_for_reviews | gross | genres | actor_1_name | m |
|---|---|---|---|---|---|---|
| **1937** | Frank Darabont | 199.0 | 2.834147e-29 | Crime\|Drama | Morgan Freeman | S Re |
| **3466** | Francis Ford Coppola | 208.0 | 1.348220e-28 | Crime\|Drama | Al Pacino | |
| **2837** | Francis Ford Coppola | 149.0 | 5.730000e-29 | Crime\|Drama | Robert De Niro | ( |
| **66** | Christopher Nolan | 645.0 | 5.333161e-28 | Action\|Crime\|Drama\|Thriller | Christian Bale | |
| **4498** | Sergio Leone | 181.0 | 6.100000e-30 | Western | Clint Eastwood | th |
| **...** | ... | ... | ... | ... | ... | ... |

| | director_name | num_critic_for_reviews | gross | genres | actor_1_name | m |
|---|---|---|---|---|---|---|
| **4931** | John Carney | 232.0 | 9.437933e-30 | Drama\|Music\|Romance | Glen Hansard | |
| **2605** | Ang Lee | 287.0 | 1.280678e-28 | Action\|Drama\|Romance | Chen Chang | |
| **3029** | David O. Russell | 410.0 | 9.357180e-29 | Biography\|Drama\|Sport | Christian Bale | T |
| **2177** | Tim Burton | 111.0 | 5.636235e-29 | Fantasy\|Romance | Johnny Depp | Sci |
| **2487** | George Cukor | 82.0 | 7.200000e-29 | Drama\|Family\|Musical\|Romance | Jeremy Brett | My |

250 rows × 15 columns

In [44]:
```
IMDb_Top_250['Rank'] = IMDb_Top_250['imdb_score'].rank(method='first',ascending=False)
IMDb_Top_250
```

Out[44]:

| | director_name | num_critic_for_reviews | gross | genres | actor_1_name | m |
|---|---|---|---|---|---|---|
| **1937** | Frank Darabont | 199.0 | 2.834147e-29 | Crime\|Drama | Morgan Freeman | S Re |
| **3466** | Francis Ford Coppola | 208.0 | 1.348220e-28 | Crime\|Drama | Al Pacino | |
| **2837** | Francis Ford Coppola | 149.0 | 5.730000e-29 | Crime\|Drama | Robert De Niro | ( |
| **66** | Christopher Nolan | 645.0 | 5.333161e-28 | Action\|Crime\|Drama\|Thriller | Christian Bale | |
| **4498** | Sergio Leone | 181.0 | 6.100000e-30 | Western | Clint Eastwood | th |
| **...** | ... | ... | ... | ... | ... | |
| **4931** | John Carney | 232.0 | 9.437933e-30 | Drama\|Music\|Romance | Glen Hansard | |
| **2605** | Ang Lee | 287.0 | 1.280678e-28 | Action\|Drama\|Romance | Chen Chang | |
| **3029** | David O. Russell | 410.0 | 9.357180e-29 | Biography\|Drama\|Sport | Christian Bale | T |
| **2177** | Tim Burton | 111.0 | 5.636235e-29 | Fantasy\|Romance | Johnny Depp | Sci |

| | director_name | num_critic_for_reviews | gross | genres | actor_1_name | m |
|---|---|---|---|---|---|---|
| **2487** | George Cukor | 82.0 | 7.200000e-29 | Drama\|Family\|Musical\|Romance | Jeremy Brett | My |

250 rows × 16 columns

In [47]:
```
Top_Foreign_Lang_Film = IMDb_Top_250[IMDb_Top_250['language']!='English']
Top_Foreign_Lang_Film
```

Out[47]:

| | director_name | num_critic_for_reviews | gross | genre |
|---|---|---|---|---|
| **4498** | Sergio Leone | 181.0 | 6.100000e-30 | Western |
| **4747** | Akira Kurosawa | 153.0 | 2.690610e-31 | Action\|Adventure\|Drama |
| **4029** | Fernando Meirelles | 214.0 | 7.563397e-30 | Crime\|Drama |
| **2373** | Hayao Miyazaki | 246.0 | 1.004989e-29 | Adventure\|Animation\|Family\|Fantasy |
| **4259** | Florian Henckel von Donnersmarck | 215.0 | 1.128466e-29 | Drama\|Thriller |
| **4921** | Majid Majidi | 46.0 | 9.254020e-31 | Drama\|Family |
| **2323** | Hayao Miyazaki | 174.0 | 2.298191e-30 | Adventure\|Animation\|Fantasy |
| **2970** | Wolfgang Petersen | 96.0 | 1.143313e-29 | Adventure\|Drama\|Thriller\|War |
| **4105** | Chan-wook Park | 305.0 | 2.181290e-30 | Drama\|Mystery\|Thriller |
| **4659** | Asghar Farhadi | 354.0 | 7.098492e-30 | Drama\|Mystery |
| **1329** | S.S. Rajamouli | 44.0 | 6.498000e-30 | Action\|Adventure\|Drama\|Fantasy\|War |
| **1298** | Jean-Pierre Jeunet | 242.0 | 3.320166e-29 | Comedy\|Romance |
| **2734** | Fritz Lang | 260.0 | 2.643500e-32 | Drama\|Sci-Fi |
| **4033** | Thomas Vinterberg | 349.0 | 6.109680e-31 | Drama |
| **2829** | Oliver Hirschbiegel | 192.0 | 5.501940e-30 | Biography\|Drama\|History\|War |

| | director_name | num_critic_for_reviews | gross | genres |
|---|---|---|---|---|
| **2551** | Guillermo del Toro | 406.0 | 3.762314e-29 | Drama\|Fantasy\|Wa |
| **4000** | Juan José Campanella | 262.0 | 2.016742e-29 | Drama\|Mystery\|Thrille |
| **3550** | Denis Villeneuve | 226.0 | 6.857096e-30 | Drama\|Mystery\|Wa |
| **2047** | Hayao Miyazaki | 212.0 | 4.710455e-30 | Adventure\|Animation\|Family\|Fantasy |
| **2830** | Alejandro Amenábar | 157.0 | 2.086345e-30 | Biography\|Drama\|Romance |
| **2914** | Je-kyu Kang | 86.0 | 1.110186e-30 | Action\|Drama\|Wa |
| **4461** | Thomas Vinterberg | 98.0 | 1.647780e-30 | Drama |
| **3553** | José Padilha | 142.0 | 8.060000e-33 | Action\|Crime\|Drama\|Thrille |
| **3423** | Katsuhiro Ôtomo | 150.0 | 4.391620e-31 | Action\|Animation\|Sci-F |
| **4267** | Alejandro G. Iñárritu | 157.0 | 5.383834e-30 | Drama\|Thrille |
| **3456** | Vincent Paronnaud | 242.0 | 4.443403e-30 | Animation\|Biography\|Drama\|Wa |
| **3344** | Karan Johar | 210.0 | 4.018695e-30 | Adventure\|Drama\|Thrille |
| **4144** | Walter Salles | 71.0 | 5.595428e-30 | Drama |
| **4284** | Ari Folman | 231.0 | 2.283276e-30 | Animation\|Biography\|Documentary\|Drama\|History\|Wa |
| **4897** | Sergio Leone | 122.0 | 3.500000e-30 | Action\|Drama\|Western |
| **1171** | Yimou Zhang | 283.0 | 8.496100e-32 | Action\|Adventure\|History |
| **2863** | Clint Eastwood | 251.0 | 1.375393e-29 | Drama\|History\|Wa |
| **3264** | Michael Haneke | 447.0 | 2.253770e-31 | Drama\|Romance |
| **3510** | Yash Chopra | 29.0 | 2.921738e-30 | Drama\|Musical\|Romance |
| **3677** | Christophe Barratier | 112.0 | 3.629758e-30 | Drama\|Music |

| | director_name | num_critic_for_reviews | gross | genre |
|---|---|---|---|---|
| **4415** | Fabián Bielinsky | 94.0 | 1.221261e-30 | Crime\|Drama\|Thrille |
| **4640** | Cristian Mungiu | 233.0 | 1.185783e-30 | Drama |
| **2605** | Ang Lee | 287.0 | 1.280678e-28 | Action\|Drama\|Romance |

**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 [48]:
```
# Write your code for extracting the top 10 directors here
top10director = movies.groupby('director_name').imdb_score.mean().sort_values(ascending
top10director
```

Out[48]:
```
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 [49]:
```
# 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[49]:

| | 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 [51]:
```
movies = pd.concat([movies,TempGenre],axis=1)
movies
```

Out[51]:

| | director_name | num_critic_for_reviews | gross | genres | actor_1_name |
|---|---|---|---|---|---|
| **0** | James Cameron | 723.0 | 7.605058e-28 | Action\|Adventure\|Fantasy\|Sci-Fi | CCH Pounde |
| **1** | Gore Verbinski | 302.0 | 3.094042e-28 | Action\|Adventure\|Fantasy | Johnny Depp |
| **2** | Sam Mendes | 602.0 | 2.000742e-28 | Action\|Adventure\|Thriller | Christopł Walt: |
| **3** | Christopher Nolan | 813.0 | 4.481306e-28 | Action\|Thriller | Tom Hardy |

| | director_name | num_critic_for_reviews | gross | genres | actor_1_name |
|---|---|---|---|---|---|
| **5** | Andrew Stanton | 462.0 | 7.305868e-29 | Action\|Adventure\|Sci-Fi | Daryl Sabara |
| **...** | ... | ... | ... | ... | ... |
| **5033** | Shane Carruth | 143.0 | 4.247600e-31 | Drama\|Sci-Fi\|Thriller | Shane Carruth |
| **5034** | Neill Dela Llana | 35.0 | 7.007100e-32 | Thriller | Ian Gamazon |
| **5035** | Robert Rodriguez | 56.0 | 2.040920e-30 | Action\|Crime\|Drama\|Romance\|Thriller | Carlos Gallardo |
| **5037** | Edward Burns | 14.0 | 4.584000e-33 | Comedy\|Drama | Kerry Bishe |
| **5042** | Jon Gunn | 43.0 | 8.522200e-32 | Documentary | John Augus |

3856 rows × 17 columns

In [53]:
```
PopGenre = movies.groupby(['genre_1','genre_2']).gross.mean().sort_values(ascending=Fal
PopGenre
```

Out[53]:
```
genre_1    genre_2
Family     Sci-Fi       4.349495e-28
Adventure  Sci-Fi       2.286278e-28
           Family       1.189195e-28
           Animation    1.169985e-28
Action     Adventure    1.095955e-28
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 [54]:
```
# Write your code for creating three new dataframes here

Meryl_Streep = movies[movies['actor_1_name']=='Meryl Streep']
```

In [58]:
```
Leo_Caprio = movies[movies['actor_1_name']=='Leonardo DiCaprio']
print(Leo_Caprio)
```

```
             director_name  num_critic_for_reviews         gross  \
26             James Cameron                   315.0  6.586723e-28
50              Baz Luhrmann                   490.0  1.448128e-28
97         Christopher Nolan                   642.0  2.925689e-28
179   Alejandro G. Iñárritu                   556.0  1.836359e-28
257          Martin Scorsese                   267.0  1.026088e-28
296        Quentin Tarantino                   765.0  1.628046e-28
307             Edward Zwick                   166.0  5.736626e-29
308          Martin Scorsese                   606.0  1.168667e-28
326          Martin Scorsese                   233.0  7.767964e-29
361          Martin Scorsese                   352.0  1.323734e-28
452          Martin Scorsese                   490.0  1.279684e-28
641             Ridley Scott                   238.0  3.938044e-29
911          Steven Spielberg                   194.0  1.644352e-28
990               Danny Boyle                   118.0  3.977860e-29
1114              Sam Mendes                   323.0  2.287781e-29
1422         Randall Wallace                    83.0  5.687637e-29
1453          Clint Eastwood                   392.0  3.730495e-29
1560               Sam Raimi                    63.0  1.863654e-29
2067              Jerry Zaks                    45.0  1.278251e-29
2757            Baz Luhrmann                   106.0  4.633873e-29
3476            Baz Luhrmann                   490.0  1.448128e-28

                              genres       actor_1_name  \
26                     Drama|Romance  Leonardo DiCaprio
50                     Drama|Romance  Leonardo DiCaprio
97    Action|Adventure|Sci-Fi|Thriller  Leonardo DiCaprio
179   Adventure|Drama|Thriller|Western  Leonardo DiCaprio
257                   Biography|Drama  Leonardo DiCaprio
296                     Drama|Western  Leonardo DiCaprio
307           Adventure|Drama|Thriller  Leonardo DiCaprio
308       Biography|Comedy|Crime|Drama  Leonardo DiCaprio
326                       Crime|Drama  Leonardo DiCaprio
361               Crime|Drama|Thriller  Leonardo DiCaprio
452                   Mystery|Thriller  Leonardo DiCaprio
641              Action|Drama|Thriller  Leonardo DiCaprio
911              Biography|Crime|Drama  Leonardo DiCaprio
990           Adventure|Drama|Thriller  Leonardo DiCaprio
1114                    Drama|Romance  Leonardo DiCaprio
1422                 Action|Adventure  Leonardo DiCaprio
1453             Biography|Crime|Drama  Leonardo DiCaprio
1560           Action|Thriller|Western  Leonardo DiCaprio
2067                             Drama  Leonardo DiCaprio
2757                    Drama|Romance  Leonardo DiCaprio
3476                    Drama|Romance  Leonardo DiCaprio

                    movie_title  num_voted_users num_user_for_reviews  \
```

| | | | |
|---|---|---|---|
| 26 | Titanic | 793059 | 2528 |
| 50 | The Great Gatsby | 362912 | 753 |
| 97 | Inception | 1468200 | 2803 |
| 179 | The Revenant | 406020 | 1188 |
| 257 | The Aviator | 264318 | 799 |
| 296 | Django Unchained | 955174 | 1193 |
| 307 | Blood Diamond | 400292 | 657 |
| 308 | The Wolf of Wall Street | 780588 | 1138 |
| 326 | Gangs of New York | 314033 | 1166 |
| 361 | The Departed | 873649 | 2054 |
| 452 | Shutter Island | 786092 | 964 |
| 641 | Body of Lies | 174248 | 263 |
| 911 | Catch Me If You Can | 525801 | 667 |
| 990 | The Beach | 176169 | 548 |
| 1114 | Revolutionary Road | 152591 | 414 |
| 1422 | The Man in the Iron Mask | 125219 | 244 |
| 1453 | J. Edgar | 102728 | 279 |
| 1560 | The Quick and the Dead | 69197 | 216 |
| 2067 | Marvin's Room | 20163 | 71 |
| 2757 | Romeo + Juliet | 167750 | 506 |
| 3476 | The Great Gatsby | 362933 | 753 |

| | language | country | budget | title_year | imdb_score | \ |
|---|---|---|---|---|---|---|
| 26 | English | USA | 2.000000e-28 | 1997.0 | 7.7 | |
| 50 | English | Australia | 1.050000e-28 | 2013.0 | 7.3 | |
| 97 | English | USA | 1.600000e-28 | 2010.0 | 8.8 | |
| 179 | English | USA | 1.350000e-28 | 2015.0 | 8.1 | |
| 257 | English | USA | 1.100000e-28 | 2004.0 | 7.5 | |
| 296 | English | USA | 1.000000e-28 | 2012.0 | 8.5 | |
| 307 | English | Germany | 1.000000e-28 | 2006.0 | 8.0 | |
| 308 | English | USA | 1.000000e-28 | 2013.0 | 8.2 | |
| 326 | English | USA | 1.000000e-28 | 2002.0 | 7.5 | |
| 361 | English | USA | 9.000000e-29 | 2006.0 | 8.5 | |
| 452 | English | USA | 8.000000e-29 | 2010.0 | 8.1 | |
| 641 | English | USA | 7.000000e-29 | 2008.0 | 7.1 | |
| 911 | English | USA | 5.200000e-29 | 2002.0 | 8.0 | |
| 990 | English | USA | 5.000000e-29 | 2000.0 | 6.6 | |
| 1114 | English | USA | 3.500000e-29 | 2008.0 | 7.3 | |
| 1422 | English | USA | 3.500000e-29 | 1998.0 | 6.4 | |
| 1453 | English | USA | 3.500000e-29 | 2011.0 | 6.6 | |
| 1560 | English | Japan | 3.200000e-29 | 1995.0 | 6.4 | |
| 2067 | English | USA | 2.300000e-29 | 1996.0 | 6.7 | |
| 2757 | English | USA | 1.450000e-29 | 1996.0 | 6.8 | |
| 3476 | English | Australia | 1.050000e-28 | 2013.0 | 7.3 | |

| | movie_facebook_likes | profit | genre_1 | genre_2 |
|---|---|---|---|---|
| 26 | 26000 | 4.586723e-28 | Drama | Romance |
| 50 | 115000 | 3.981280e-29 | Drama | Romance |
| 97 | 175000 | 1.325689e-28 | Action | Adventure |
| 179 | 190000 | 4.863592e-29 | Adventure | Drama |
| 257 | 0 | -7.391173e-30 | Biography | Drama |
| 296 | 199000 | 6.280465e-29 | Drama | Western |
| 307 | 14000 | -4.263374e-29 | Adventure | Drama |
| 308 | 138000 | 1.686673e-29 | Biography | Comedy |
| 326 | 0 | -2.232036e-29 | Crime | Drama |
| 361 | 29000 | 4.237344e-29 | Crime | Drama |
| 452 | 53000 | 4.796840e-29 | Mystery | Thriller |
| 641 | 0 | -3.061956e-29 | Action | Drama |
| 911 | 15000 | 1.124352e-28 | Biography | Crime |
| 990 | 0 | -1.022140e-29 | Adventure | Drama |

```
1114                   0 -1.212219e-29      Drama    Romance
1422                   0  2.187637e-29     Action  Adventure
1453               16000  2.304950e-30  Biography      Crime
1560                   0 -1.336346e-29     Action   Thriller
2067                1000 -1.021749e-29      Drama      Drama
2757               10000  3.183873e-29      Drama    Romance
3476              115000  3.981280e-29      Drama    Romance
```

In [61]:

```
Brad_Pitt = movies[movies['actor_1_name']=='Brad Pitt']
print(Brad_Pitt)
```

```
             director_name  num_critic_for_reviews         gross  \
101           David Fincher                   362.0  1.274908e-28
147       Wolfgang Petersen                   220.0  1.332283e-28
254       Steven Soderbergh                   198.0  1.255316e-28
255              Doug Liman                   233.0  1.863361e-28
382              Tony Scott                   142.0  2.687100e-32
400       Steven Soderbergh                   186.0  1.834058e-28
470              David Ayer                   406.0  8.570712e-29
611     Jean-Jacques Annaud                    76.0  3.790151e-29
683           David Fincher                   315.0  3.702339e-29
792         Patrick Gilmore                    98.0  2.628832e-29
940              Neil Jordan                   120.0  1.052646e-28
1490        Terrence Malick                   584.0  1.330332e-29
1722         Andrew Dominik                   273.0  3.904982e-30
2204  Alejandro G. Iñárritu                   285.0  3.430077e-29
2333    Angelina Jolie Pitt                   131.0  5.310090e-31
2682         Andrew Dominik                   414.0  1.493857e-29
2898              Tony Scott                   122.0  1.228150e-29

                                            genres actor_1_name  \
101                        Drama|Fantasy|Romance    Brad Pitt
147                                    Adventure    Brad Pitt
254                              Crime|Thriller    Brad Pitt
255          Action|Comedy|Crime|Romance|Thriller    Brad Pitt
382                        Action|Crime|Thriller    Brad Pitt
400                              Crime|Thriller    Brad Pitt
470                           Action|Drama|War    Brad Pitt
611       Adventure|Biography|Drama|History|War    Brad Pitt
683                                      Drama    Brad Pitt
792   Adventure|Animation|Comedy|Drama|Family|Fantas...    Brad Pitt
940                        Drama|Fantasy|Horror    Brad Pitt
1490                             Drama|Fantasy    Brad Pitt
1722       Biography|Crime|Drama|History|Western    Brad Pitt
2204                                     Drama    Brad Pitt
2333                             Drama|Romance    Brad Pitt
2682                             Crime|Thriller    Brad Pitt
2898          Action|Crime|Drama|Romance|Thriller    Brad Pitt

                                      movie_title  num_voted_users  \
101   The Curious Case of Benjamin Button           459346
147                                   Troy           381672
254                          Ocean's Twelve          284852
255                       Mr. & Mrs. Smith           348861
382                              Spy Game            121259
400                          Ocean's Eleven          402645
470                                   Fury           303185
611                  Seven Years in Tibet             96385
683                              Fight Club          1347461
```

```
792                  Sinbad: Legend of the Seven Seas            36144
940   Interview with the Vampire: The Vampire Chroni...          239752
1490                                 The Tree of Life            136367
1722  The Assassination of Jesse James by the Coward...          136104
2204                                           Babel             243799
2333                                        By the Sea               7976
2682                                Killing Them Softly            111625
2898                                       True Romance            163492

     num_user_for_reviews language  country        budget  title_year  \
101                   822  English      USA  1.500000e-28      2008.0
147                  1694  English      USA  1.750000e-28      2004.0
254                   627  English      USA  1.100000e-28      2004.0
255                   798  English      USA  1.200000e-28      2005.0
382                   361  English  Germany  9.200000e-29      2001.0
400                   845  English      USA  8.500000e-29      2001.0
470                   701  English      USA  6.800000e-29      2014.0
611                   119  English      USA  7.000000e-29      1997.0
683                  2968  English      USA  6.300000e-29      1999.0
792                    91  English      USA  6.000000e-29      2003.0
940                   406  English      USA  6.000000e-29      1994.0
1490                  975  English      USA  3.200000e-29      2011.0
1722                  415  English      USA  3.000000e-29      2007.0
2204                  908  English   France  2.500000e-29      2006.0
2333                   61  English      USA  1.000000e-29      2015.0
2682                  369  English      USA  1.500000e-29      2012.0
2898                  460  English      USA  1.300000e-29      1993.0

      imdb_score  movie_facebook_likes        profit    genre_1    genre_2
101          7.8                 23000 -2.250920e-29      Drama    Fantasy
147          7.2                     0 -4.177165e-29  Adventure  Adventure
254          6.4                     0  1.553163e-29      Crime   Thriller
255          6.5                     0  6.633610e-29     Action     Comedy
382          7.0                     0 -9.197313e-29     Action      Crime
400          7.8                     0  9.840577e-29      Crime   Thriller
470          7.6                 82000  1.770712e-29     Action      Drama
611          7.0                     0 -3.209849e-29  Adventure  Biography
683          8.8                 48000 -2.597661e-29      Drama      Drama
792          6.7                   880 -3.371168e-29  Adventure  Animation
940          7.6                 11000  4.526461e-29      Drama    Fantasy
1490         6.7                 39000 -1.869668e-29      Drama    Fantasy
1722         7.5                     0 -2.609502e-29  Biography      Crime
2204         7.5                     0  9.300771e-30      Drama      Drama
2333         5.3                     0 -9.468991e-30      Drama    Romance
2682         6.2                 20000 -6.143000e-32      Crime   Thriller
2898         8.0                 15000 -7.185000e-31     Action      Crime
```

In [73]:

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

Combined
```

Out[73]:

| | director_name | num_critic_for_reviews | gross | genres |
|---|---|---|---|---|
| **410** | Nancy Meyers | 187.0 | 1.127035e-28 | Comedy\|Drama\|Romance |
| **1106** | Curtis Hanson | 42.0 | 4.681575e-29 | Action\|Adventure\|Crime\|Thriller |

| | director_name | num_critic_for_reviews | gross | genres |
|---|---|---|---|---|
| **1204** | Nora Ephron | 252.0 | 9.412543e-29 | Biography\|Drama\|Romance |
| **1408** | David Frankel | 208.0 | 1.247330e-28 | Comedy\|Drama\|Romance |
| **1483** | Robert Redford | 227.0 | 1.499807e-29 | Drama\|Thriller\|War |
| **1575** | Sydney Pollack | 66.0 | 8.710000e-29 | Biography\|Drama\|Romance |
| **1618** | David Frankel | 234.0 | 6.353601e-29 | Comedy\|Drama\|Romance |
| **1674** | Carl Franklin | 64.0 | 2.320944e-29 | Drama |
| **1925** | Stephen Daldry | 174.0 | 4.159783e-29 | Drama\|Romance |
| **2781** | Phyllida Lloyd | 331.0 | 2.995944e-29 | Biography\|Drama\|History |
| **3135** | Robert Altman | 211.0 | 2.033861e-29 | Comedy\|Drama\|Music |
| **26** | James Cameron | 315.0 | 6.586723e-28 | Drama\|Romance |
| **50** | Baz Luhrmann | 490.0 | 1.448128e-28 | Drama\|Romance |
| **97** | Christopher Nolan | 642.0 | 2.925689e-28 | Action\|Adventure\|Sci-Fi\|Thriller |
| **179** | Alejandro G. Iñárritu | 556.0 | 1.836359e-28 | Adventure\|Drama\|Thriller\|Western |
| **257** | Martin Scorsese | 267.0 | 1.026088e-28 | Biography\|Drama |
| **296** | Quentin Tarantino | 765.0 | 1.628046e-28 | Drama\|Western |
| **307** | Edward Zwick | 166.0 | 5.736626e-29 | Adventure\|Drama\|Thriller |
| **308** | Martin Scorsese | 606.0 | 1.168667e-28 | Biography\|Comedy\|Crime\|Drama |
| **326** | Martin Scorsese | 233.0 | 7.767964e-29 | Crime\|Drama |
| **361** | Martin Scorsese | 352.0 | 1.323734e-28 | Crime\|Drama\|Thriller |
| **452** | Martin Scorsese | 490.0 | 1.279684e-28 | Mystery\|Thriller |
| **641** | Ridley Scott | 238.0 | 3.938044e-29 | Action\|Drama\|Thriller |

| | director_name | num_critic_for_reviews | gross | genres |
|---|---|---|---|---|
| **911** | Steven Spielberg | 194.0 | 1.644352e-28 | Biography\|Crime\|Drama |
| **990** | Danny Boyle | 118.0 | 3.977860e-29 | Adventure\|Drama\|Thriller |
| **1114** | Sam Mendes | 323.0 | 2.287781e-29 | Drama\|Romance |
| **1422** | Randall Wallace | 83.0 | 5.687637e-29 | Action\|Adventure |
| **1453** | Clint Eastwood | 392.0 | 3.730495e-29 | Biography\|Crime\|Drama |
| **1560** | Sam Raimi | 63.0 | 1.863654e-29 | Action\|Thriller\|Western |
| **2067** | Jerry Zaks | 45.0 | 1.278251e-29 | Drama |
| **2757** | Baz Luhrmann | 106.0 | 4.633873e-29 | Drama\|Romance |
| **3476** | Baz Luhrmann | 490.0 | 1.448128e-28 | Drama\|Romance |
| **101** | David Fincher | 362.0 | 1.274908e-28 | Drama\|Fantasy\|Romance |
| **147** | Wolfgang Petersen | 220.0 | 1.332283e-28 | Adventure |
| **254** | Steven Soderbergh | 198.0 | 1.255316e-28 | Crime\|Thriller |
| **255** | Doug Liman | 233.0 | 1.863361e-28 | Action\|Comedy\|Crime\|Romance\|Thriller |
| **382** | Tony Scott | 142.0 | 2.687100e-32 | Action\|Crime\|Thriller |
| **400** | Steven Soderbergh | 186.0 | 1.834058e-28 | Crime\|Thriller |
| **470** | David Ayer | 406.0 | 8.570712e-29 | Action\|Drama\|War |
| **611** | Jean-Jacques Annaud | 76.0 | 3.790151e-29 | Adventure\|Biography\|Drama\|History\|War |
| **683** | David Fincher | 315.0 | 3.702339e-29 | Drama |
| **792** | Patrick Gilmore | 98.0 | 2.628832e-29 | Adventure\|Animation\|Comedy\|Drama\|Family\|Fantas... |

| | director_name | num_critic_for_reviews | gross | genres |
|---|---|---|---|---|
| **940** | Neil Jordan | 120.0 | 1.052646e-28 | Drama\|Fantasy\|Horror |
| **1490** | Terrence Malick | 584.0 | 1.330332e-29 | Drama\|Fantasy |
| **1722** | Andrew Dominik | 273.0 | 3.904982e-30 | Biography\|Crime\|Drama\|History\|Western |
| **2204** | Alejandro G. Iñárritu | 285.0 | 3.430077e-29 | Drama |
| **2333** | Angelina Jolie Pitt | 131.0 | 5.310090e-31 | Drama\|Romance |
| **2682** | Andrew Dominik | 414.0 | 1.493857e-29 | Crime\|Thriller |
| **2898** | Tony Scott | 122.0 | 1.228150e-29 | Action\|Crime\|Drama\|Romance\|Thriller |

In [71]:
```
# Write the code for finding the mean of critic reviews and audience reviews here
Combined.groupby('actor_1_name').num_critic_for_reviews.mean()
```

Out[71]:
```
actor_1_name
Brad Pitt            245.000000
Leonardo DiCaprio    330.190476
Meryl Streep         181.454545
Name: num_critic_for_reviews, dtype: float64
```

In [75]:
```
Combined
```

Out[75]:

| | director_name | num_critic_for_reviews | gross | genres |
|---|---|---|---|---|
| **410** | Nancy Meyers | 187.0 | 1.127035e-28 | Comedy\|Drama\|Romance |
| **1106** | Curtis Hanson | 42.0 | 4.681575e-29 | Action\|Adventure\|Crime\|Thriller |
| **1204** | Nora Ephron | 252.0 | 9.412543e-29 | Biography\|Drama\|Romance |
| **1408** | David Frankel | 208.0 | 1.247330e-28 | Comedy\|Drama\|Romance |
| **1483** | Robert Redford | 227.0 | 1.499807e-29 | Drama\|Thriller\|War |
| **1575** | Sydney Pollack | 66.0 | 8.710000e-29 | Biography\|Drama\|Romance |

| | director_name | num_critic_for_reviews | gross | genres |
|---|---|---|---|---|
| **1618** | David Frankel | 234.0 | 6.353601e-29 | Comedy\|Drama\|Romance |
| **1674** | Carl Franklin | 64.0 | 2.320944e-29 | Drama |
| **1925** | Stephen Daldry | 174.0 | 4.159783e-29 | Drama\|Romance |
| **2781** | Phyllida Lloyd | 331.0 | 2.995944e-29 | Biography\|Drama\|History |
| **3135** | Robert Altman | 211.0 | 2.033861e-29 | Comedy\|Drama\|Music |
| **26** | James Cameron | 315.0 | 6.586723e-28 | Drama\|Romance |
| **50** | Baz Luhrmann | 490.0 | 1.448128e-28 | Drama\|Romance |
| **97** | Christopher Nolan | 642.0 | 2.925689e-28 | Action\|Adventure\|Sci-Fi\|Thriller |
| **179** | Alejandro G. Iñárritu | 556.0 | 1.836359e-28 | Adventure\|Drama\|Thriller\|Western |
| **257** | Martin Scorsese | 267.0 | 1.026088e-28 | Biography\|Drama |
| **296** | Quentin Tarantino | 765.0 | 1.628046e-28 | Drama\|Western |
| **307** | Edward Zwick | 166.0 | 5.736626e-29 | Adventure\|Drama\|Thriller |
| **308** | Martin Scorsese | 606.0 | 1.168667e-28 | Biography\|Comedy\|Crime\|Drama |
| **326** | Martin Scorsese | 233.0 | 7.767964e-29 | Crime\|Drama |
| **361** | Martin Scorsese | 352.0 | 1.323734e-28 | Crime\|Drama\|Thriller |
| **452** | Martin Scorsese | 490.0 | 1.279684e-28 | Mystery\|Thriller |
| **641** | Ridley Scott | 238.0 | 3.938044e-29 | Action\|Drama\|Thriller |
| **911** | Steven Spielberg | 194.0 | 1.644352e-28 | Biography\|Crime\|Drama |
| **990** | Danny Boyle | 118.0 | 3.977860e-29 | Adventure\|Drama\|Thriller |
| **1114** | Sam Mendes | 323.0 | 2.287781e-29 | Drama\|Romance |
| **1422** | Randall Wallace | 83.0 | 5.687637e-29 | Action\|Adventure |

| | director_name | num_critic_for_reviews | gross | genres |
|---|---|---|---|---|
| **1453** | Clint Eastwood | 392.0 | 3.730495e-29 | Biography\|Crime\|Drama |
| **1560** | Sam Raimi | 63.0 | 1.863654e-29 | Action\|Thriller\|Western |
| **2067** | Jerry Zaks | 45.0 | 1.278251e-29 | Drama |
| **2757** | Baz Luhrmann | 106.0 | 4.633873e-29 | Drama\|Romance |
| **3476** | Baz Luhrmann | 490.0 | 1.448128e-28 | Drama\|Romance |
| **101** | David Fincher | 362.0 | 1.274908e-28 | Drama\|Fantasy\|Romance |
| **147** | Wolfgang Petersen | 220.0 | 1.332283e-28 | Adventure |
| **254** | Steven Soderbergh | 198.0 | 1.255316e-28 | Crime\|Thriller |
| **255** | Doug Liman | 233.0 | 1.863361e-28 | Action\|Comedy\|Crime\|Romance\|Thriller |
| **382** | Tony Scott | 142.0 | 2.687100e-32 | Action\|Crime\|Thriller |
| **400** | Steven Soderbergh | 186.0 | 1.834058e-28 | Crime\|Thriller |
| **470** | David Ayer | 406.0 | 8.570712e-29 | Action\|Drama\|War |
| **611** | Jean-Jacques Annaud | 76.0 | 3.790151e-29 | Adventure\|Biography\|Drama\|History\|War |
| **683** | David Fincher | 315.0 | 3.702339e-29 | Drama |
| **792** | Patrick Gilmore | 98.0 | 2.628832e-29 | Adventure\|Animation\|Comedy\|Drama\|Family\|Fantas... |
| **940** | Neil Jordan | 120.0 | 1.052646e-28 | Drama\|Fantasy\|Horror |
| **1490** | Terrence Malick | 584.0 | 1.330332e-29 | Drama\|Fantasy |

| | director_name | num_critic_for_reviews | gross | genres |
|---|---|---|---|---|
| **1722** | Andrew Dominik | 273.0 | 3.904982e-30 | Biography\|Crime\|Drama\|History\|Western |
| **2204** | Alejandro G. Iñárritu | 285.0 | 3.430077e-29 | Drama |
| **2333** | Angelina Jolie Pitt | 131.0 | 5.310090e-31 | Drama\|Romance |
| **2682** | Andrew Dominik | 414.0 | 1.493857e-29 | Crime\|Thriller |
| **2898** | Tony Scott | 122.0 | 1.228150e-29 | Action\|Crime\|Drama\|Romance\|Thriller |

In [76]:
```
Combined.num_user_for_reviews = Combined.num_user_for_reviews.astype('int')
Combined.num_user_for_reviews
```

Out[76]:
```
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 [77]:
```
Combined.groupby('actor_1_name').num_user_for_reviews.mean()
```

Out[77]:
```
actor_1_name
Brad Pitt            742.352941
Leonardo DiCaprio    914.476190
Meryl Streep         297.181818
Name: num_user_for_reviews, dtype: float64
```

In [78]:
```
Combined.groupby('actor_1_name')[['num_critic_for_reviews','num_user_for_reviews']].mea
```

Out[78]:

| 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 [86]:
```
# Write the code for calculating decade here
Movies_by_decade=movies.copy(deep=True)
Movies_by_decade['decade']=Movies_by_decade['title_year'].apply(lambda x:10*(int(x/10))
Movies_by_decade
```

Out[86]:

| | director_name | num_critic_for_reviews | gross | genres | actor_1_name |
|---|---|---|---|---|---|
| **0** | James Cameron | 723.0 | 7.605058e-28 | Action\|Adventure\|Fantasy\|Sci-Fi | CCH Pounde |
| **1** | Gore Verbinski | 302.0 | 3.094042e-28 | Action\|Adventure\|Fantasy | Johnny Depp |
| **2** | Sam Mendes | 602.0 | 2.000742e-28 | Action\|Adventure\|Thriller | Christopl Walt |

| | director_name | num_critic_for_reviews | gross | genres | actor_1_name |
|---|---|---|---|---|---|
| **3** | Christopher Nolan | 813.0 | 4.481306e-28 | Action\|Thriller | Tom Hardy |
| **5** | Andrew Stanton | 462.0 | 7.305868e-29 | Action\|Adventure\|Sci-Fi | Daryl Sabara |
| **...** | ... | ... | ... | ... | ... |
| **5033** | Shane Carruth | 143.0 | 4.247600e-31 | Drama\|Sci-Fi\|Thriller | Shane Carruth |
| **5034** | Neill Dela Llana | 35.0 | 7.007100e-32 | Thriller | Ian Gamazon |
| **5035** | Robert Rodriguez | 56.0 | 2.040920e-30 | Action\|Crime\|Drama\|Romance\|Thriller | Carlos Gallardo |
| **5037** | Edward Burns | 14.0 | 4.584000e-33 | Comedy\|Drama | Kerry Bishe |
| **5042** | Jon Gunn | 43.0 | 8.522200e-32 | Documentary | John Augus |

3856 rows × 18 columns

In [87]:
```
# Write your code for creating the data frame df_by_decade here
Movies_by_decade = Movies_by_decade.groupby('decade',as_index=False)['num_voted_users']
Movies_by_decade
```

Out[87]:

| | 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 [88]:
```
# Write your code for plotting number of voted users vs decade
sns.lineplot(data= Movies_by_decade, x='decade',y='num_voted_users')
plt.yscale('log')
plt.ylabel("num_voted_users")
```

```
plt.xlabel("decade")
plt.show()
```