IMDB Movie Rating

Project Description: This project contains the dataset from the IMDB website. It contains movie , their rating, their genre, their net profit, etc. This project contains a handful amount of problems to be solved for the practice and get the insight from it.

Approach: I loaded the dataset, understand it, clean it and perform the required task.

Tech-Stack Used: I have used the google online notebook Collab for this project. The purpose behind using Collab is that we don't need to install any notebook software locally on my system.

Insights: With this project, I learnt how to approach the problem, how to convert the logic into code and hidden insights I can get via libraries like matplotlib which helps to plot the graphs.

Result: Now I am comfortable working with python libraries like Numpy and Pandas, because this project contains a whole lot of problems which I solved.

DATASET LINK

PROBLEMS LINK

MY DRIVE LINK

All the solution of problems are provided below:

```
# Supress Warnings

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

```
# Import the numpy and pandas packages

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

Task 1: Reading and Inspection

- ## Subtask 1.1: Import and read

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

```
movies = pd.read_csv('IMDB_Movies.csv')
movies.head()
```

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

Subtask 1.2: Inspect the dataframe

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

```
# Check the number of rows and columns in the dataframe
movies.shape
```

```
(5043, 28)
```

```
# Check the column-wise info of the dataframe
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
```

```
# Check the summary for the numeric columns
movies.describe()
```

	num_critic_for_reviews	duration	director_facebook_likes	actor_3_faceboc
count	4993.000000	5028.000000	4939.000000	502

mean	140.194272	107.201074	686.509212	64
std	121.601675	25.197441	2813.328607	166
min	1.000000	7.000000	0.000000	
25%	50.000000	93.000000	7.000000	13
50%	110.000000	103.000000	49.000000	37
75%	195.000000	118.000000	194.500000	63

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.

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

```
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
```

```
# Write your code for row-wise null count here
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
```

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

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

Subtask 2.2: Drop unecessary columns

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

- color
- director_facebook_likes
- actor_1_facebook_likes
- actor_2_facebook_likes

- actor_3_facebook_likes
- actor_2_name
- cast_total_facebook_likes
- actor_3_name
- duration
- facenumber_in_poster
- content_rating
- country
- movie_imdb_link
- aspect_ratio
- plot_keywords

```
# Write your code for dropping the columns here. It is advised to keep inspecting the data
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)
round((movies.isnull().sum()/len(movies.index)*100),2)
```

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

```
movies.head()
```

	director_name	num_critic_for_reviews	gross	genres	act
0	James Cameron	723.0	760505847.0	Action|Adventure|Fantasy|Sci-Fi	C(

1	Gore Verbinski	302.0	309404152.0	Action|Adventure|Fantasy	J
2	Sam Mendes	602.0	200074175.0	Action|Adventure|Thriller	
3	Christopher	813.0	448130642.0	Action|Thriller	

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.

```
# Write your code for dropping the rows here
movies=movies[~np.isnan(movies['budget'])]
movies=movies[~np.isnan(movies['gross'])]

round((movies.isnull().sum()/len(movies.index)*100),2)
```

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

Subtask 2.4: Fill NaN values

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

```
# Write your code for filling the NaN values in the 'language' column here
movies.loc[pd.isnull(movies['language']),['language']]="English"
round((movies.isnull().sum()/len(movies.index)*100))
```

```
director_name             0.0
num_critic_for_reviews    0.0
gross                     0.0
genres                    0.0
actor_1_name              0.0
movie_title               0.0
```

```
num_voted_users           0.0
num_user_for_reviews      0.0
language                  0.0
budget                    0.0
title_year                0.0
imdb_score                0.0
movie_facebook_likes      0.0
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.

```
# Write your code for checking number of retained rows here
print(movies.shape)

print('Number of rows retained after completing all the tasks above: ',movies.shape[0])
print(len(movies.index)/5043*100)       #Dividing movies.index with total number of rows(5
```

```
(3891, 13)
Number of rows retained after completing all the tasks above:  3891
77.15645449137418
```

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

```
movies.head()
```

Task 3: Data Analysis

- ## Subtask 3.1: Change the unit of columns

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

```
# Write your code for unit conversion here
movies['budget']=movies['budget'].apply(lambda x: round(x/1000000,2))
movies['gross']=movies['gross'].apply(lambda x: round(x/1000000,2))
movies.head()
```

	director_name	num_critic_for_reviews	gross	genres	actor_1_
0	James Cameron	723.0	760.51	Action|Adventure|Fantasy|Sci-Fi	CCH Po
1	Gore Verbinski	302.0	309.40	Action|Adventure|Fantasy	Johnny
2	Sam Mendes	602.0	200.07	Action|Adventure|Thriller	Chri
3	Christopher Nolan	813.0	448.13	Action|Thriller	Tom I
5	Andrew Stanton	462.0	73.06	Action|Adventure|Sci-Fi	Daryl S

- ## 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`

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

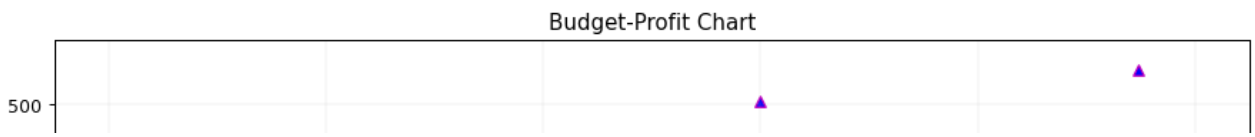
	director_name	num_critic_for_reviews	gross	genres	actor_1_
0	James Cameron	723.0	760.51	Action|Adventure|Fantasy|Sci-Fi	CCH Po

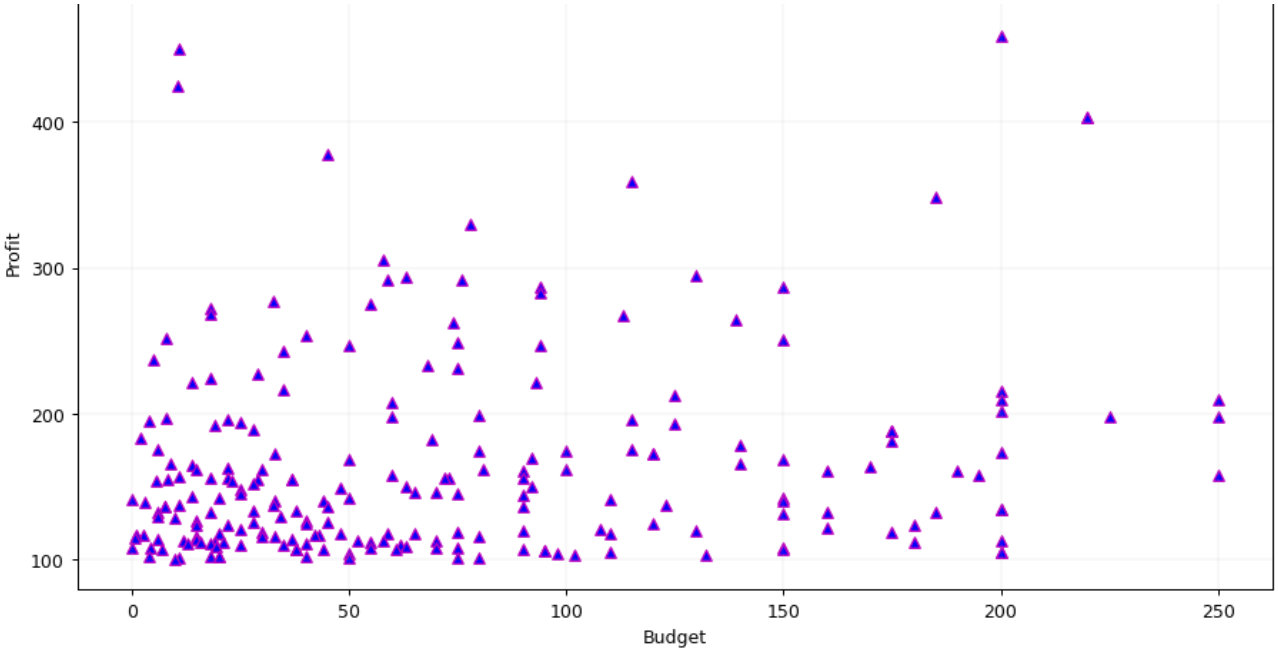
1	Gore Verbinski	302.0	309.40	Action|Adventure|Fantasy	Johnny
2	Sam Mendes	602.0	200.07	Action|Adventure|Thriller	Chri
3	Christopher Nolan	813.0	448.13	Action|Thriller	Tom I
5	Andrew Stanton	462.0	73.06	Action|Adventure|Sci-Fi	Daryl S

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

	director_name	num_critic_for_reviews	gross	genres	actor
0	James Cameron	723.0	760.51	Action|Adventure|Fantasy|Sci-Fi	CCH
29	Colin Trevorrow	644.0	652.18	Action|Adventure|Sci-Fi|Thriller	Bry
26	James Cameron	315.0	658.67	Drama|Romance	
3024	George Lucas	282.0	460.94	Action|Adventure|Fantasy|Sci-Fi	Harr
3080	Steven Spielberg	215.0	434.95	Family|Sci-Fi	Henry

```
# Write code for profit vs budget plot here
plt.figure(num=None, figsize=(12,7), dpi=90)
movie=movie[movie.profit>100]
plt.scatter(movie['budget'], movie['profit'],marker ="^",edgecolors = 'm',facecolor='b') #
plt.xlabel("Budget") #Label x
plt.ylabel("Profit") #Label y
plt.title("Budget-Profit Chart")
plt.grid( linestyle='-', linewidth=0.25, alpha=0.5)
plt.show()
```






```
# Write your code to get the top 10 profiting movies here
top10=movie[['movie_title']][0:10]  # CreatingNew Dataframe
top10
```

	movie_title
0	Avatar
29	Jurassic World
26	Titanic
3024	Star Wars: Episode IV - A New Hope
3080	E.T. the Extra-Terrestrial
794	The Avengers
17	The Avengers
509	The Lion King
240	Star Wars: Episode I - The Phantom Menace
66	The Dark Knight

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.

```
# Write your code for dropping duplicate values here
movies=movies.drop_duplicates()
movies
```

	director_name	num_critic_for_reviews	gross	genres
0	James Cameron	723.0	760.51	Action|Adventure|Fantasy|Sci-Fi
1	Gore Verbinski	302.0	309.40	Action|Adventure|Fantasy
2	Sam Mendes	602.0	200.07	Action|Adventure|Thriller
3	Christopher Nolan	813.0	448.13	Action|Thriller
5	Andrew Stanton	462.0	73.06	Action|Adventure|Sci-Fi
...	...	...	...	...
5033	Shane Carruth	143.0	0.42	Drama|Sci-Fi|Thriller
5034	Neill Dela Llana	35.0	0.07	Thriller
5035	Robert Rodriguez	56.0	2.04	Action|Crime|Drama|Romance|Thriller
5037	Edward Burns	14.0	0.00	Comedy|Drama
5042	Jon Gunn	43.0	0.09	Documentary

3856 rows × 14 columns

```
# Write code for repeating subtask 2 here
movies['profit']=movies['gross']-movies['budget']
movie=movies.sort_values(by=['profit'],ascending=False)

top10=movie[['director_name','movie_title']]
top10.head(10)
```

	director_name	movie_title
0	James Cameron	Avatar

29	Colin Trevorrow	Jurassic World
26	James Cameron	Titanic
3024	George Lucas	Star Wars: Episode IV - A New Hope
3080	Steven Spielberg	E.T. the Extra-Terrestrial
17	Joss Whedon	The Avengers
509	Roger Allers	The Lion King

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

66	Christopher Nolan	The Dark Knight

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`.

```
# Write your code for extracting the top 250 movies as per the IMDb score here. Make sure
# and name that dataframe as 'IMDb_Top_250'
IMDb_Top_250=movies[['imdb_score','num_voted_users','movie_title','language']]

IMDb_sort= IMDb_Top_250.sort_values(by=['imdb_score'],ascending=False)
IMDb_Top_250=IMDb_sort[IMDb_Top_250.num_voted_users>25000]
IMDb_Top_250.head(250)
```

	imdb_score	num_voted_users	movie_title	language
1937	9.3	1689764	The Shawshank Redemption	English

3466	9.2	1155770	The Godfather	English

```
#Rank column containing the values 1 to 250 indicating the ranks of the corresponding film
IMDb_Top_250["Rank"]=IMDb_Top_250['movie_title'].rank()
IMDb_Top_250['Rank']=IMDb_Top_250['Rank'].sort_values(ascending=True).values
IMDb_Top_250.head()
```

	imdb_score	num_voted_users	movie_title	language	Rank
1937	9.3	1689764	The Shawshank Redemption	English	1.0
3466	9.2	1155770	The Godfather	English	2.0
2837	9.0	790926	The Godfather: Part II	English	3.0
66	9.0	1676169	The Dark Knight	English	4.0
339	8.9	1215718	The Lord of the Rings: The Return of the King	English	5.0

250 rows × 4 columns

```
#Setting the Rank column as the index
IMDb_Top_250=IMDb_Top_250.set_index('Rank')
IMDb_Top_250.head(250)
```

	imdb_score	num_voted_users	movie_title	language
Rank				
1.0	9.3	1689764	The Shawshank Redemption	English
2.0	9.2	1155770	The Godfather	English
3.0	9.0	790926	The Godfather: Part II	English
4.0	9.0	1676169	The Dark Knight	English
5.0	8.9	1215718	The Lord of the Rings: The Return of the King	English
...	...	...	...	...
246.0	7.9	483756	Taken	English
247.0	7.9	483540	The Hobbit: The Desolation of Smaug	English
248.0	7.9	219008	The Untouchables	English
249.0	7.9	44763	4 Months, 3 Weeks and 2 Days	Romanian
250.0	7.9	90827	Once	English

```
# Write your code to extract top foreign language films from 'IMDb_Top_250' here
Top_Foreign_Lang_Film = IMDb_Top_250[(IMDb_Top_250.language=='Hindi') ] #Extracting foreig
Top_Foreign_Lang_Film[0:250]
```

	imdb_score	num_voted_users	movie_title	language
Rank				

170.0	8.0	69759	My Name Is Khan	Hindi

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.

```
# Write your code for extracting the top 10 directors here
mov=movies.groupby('director_name')

top10director=pd.DataFrame(mov['imdb_score'].mean().sort_values(ascending=False))  #Conver
top10director=top10director.head(10)

top10director=top10director.sort_values(['imdb_score','director_name'],ascending=(False,Tr
top10director
```

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

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

Subtask 3.6: Find popular genres

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

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

```
# Write your code for extracting the first two genres of each movie here
movies['genres']=movies.genres.str.split('|')
movies['genre_1']=movies['genres'].apply(lambda x: x[0])  #Extracting one Genre
movies['genre_2']=movies['genres'].apply(lambda x: x[1] if len(x)>1 else x[0])   #Extracti
movies.head()
```

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_tit
0	James Cameron	723.0	760.51	[Action, Adventure, Fantasy, Sci-Fi]	CCH Pounder	Avat
1	Gore Verbinski	302.0	309.40	[Action, Adventure, Fantasy]	Johnny Depp	Pirates of tl Caribbea At Worlc E
2	Sam Mendes	602.0	200.07	[Action, Adventure, Thriller]	Christoph Waltz	Spect
3	Christopher Nolan	813.0	448.13	[Action, Thriller]	Tom Hardy	The Da Knight Ris
5	Andrew Stanton	462.0	73.06	[Action, Adventure, Sci-Fi]	Daryl Sabara	John Cart

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

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x7f892bcb9fd0>
```

```
# Write your code for getting the 5 most popular combo of genres here
PropGenre=pd.DataFrame(movies_by_segment.gross.mean().sort_values(ascending=False) )
PropGenre[0:5]
```

```
PropGenre[0.5]
```

genre_1	genre_2	gross
Family	**Sci-Fi**	434.950000
Adventure	**Sci-Fi**	228.628750
	Family	118.918824
	Animation	116.998462
Action	**Adventure**	109.595510

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`.

```
# Write your code for creating three new dataframes here
Meryl_Streep=movies[['actor_1_name','movie_title','num_critic_for_reviews','num_user_for_r
Leo_Caprio=movies[['actor_1_name','movie_title','num_critic_for_reviews','num_user_for_rev
Brad_Pitt=movies[['actor_1_name','movie_title','num_critic_for_reviews','num_user_for_revi

# Include all movies in which Meryl_Streep is the lead
Meryl_Streep=Meryl_Streep.loc[Meryl_Streep['actor_1_name']=='Meryl Streep',:]
Meryl_Streep.head()
```

	actor_1_name	movie_title	num_critic_for_reviews	num_user_for_reviews
410	Meryl Streep	It's Complicated	187.0	214

1106	Meryl Streep	The River Wild	42.0	69
1204	Meryl Streep	Julie & Julia	252.0	277
1408	Meryl Streep	The Devil Wears Prada	208.0	631
1483	Meryl Streep	Lions for Lambs	227.0	298

```
# Include all movies in which Leo_Caprio is the lead
Leo_Caprio=Leo_Caprio.loc[Leo_Caprio['actor_1_name']=='Leonardo DiCaprio',:]
Leo_Caprio.head()
```

	actor_1_name	movie_title	num_critic_for_reviews	num_user_for_reviews
26	Leonardo DiCaprio	Titanic	315.0	2528
50	Leonardo DiCaprio	The Great Gatsby	490.0	753
97	Leonardo DiCaprio	Inception	642.0	2803

```
# Include all movies in which Brad_Pitt is the lead
Brad_Pitt=Brad_Pitt.loc[Brad_Pitt['actor_1_name']=='Brad Pitt',:]
Brad_Pitt.head()
```

	actor_1_name	movie_title	num_critic_for_reviews	num_user_for_reviews
101	Brad Pitt	The Curious Case of Benjamin Button	362.0	822
147	Brad Pitt	Troy	220.0	1694
254	Brad Pitt	Ocean's Twelve	198.0	627
255	Brad Pitt	Mr. & Mrs. Smith	233.0	798
382	Brad Pitt	Spy Game	142.0	361

```
# Write your code for combining the three dataframes here
Combined=Meryl_Streep.append(Leo_Caprio).append(Brad_Pitt)
Combined
```

	actor_1_name	movie_title	num_critic_for_reviews	num_user_for_reviews
410	Meryl Streep	It's Complicated	187.0	214

1106	Meryl Streep	The River Wild	42.0	69
1204	Meryl Streep	Julie & Julia	252.0	277
1408	Meryl Streep	The Devil Wears Prada	208.0	631
1483	Meryl Streep	Lions for Lambs	227.0	298
1575	Meryl Streep	Out of Africa	66.0	200
1618	Meryl Streep	Hope Springs	234.0	178
1674	Meryl Streep	One True Thing	64.0	112
1925	Meryl Streep	The Hours	174.0	660
2781	Meryl Streep	The Iron Lady	331.0	350
3135	Meryl Streep	A Prairie Home Companion	211.0	280
26	Leonardo DiCaprio	Titanic	315.0	2528
50	Leonardo DiCaprio	The Great Gatsby	490.0	753
97	Leonardo DiCaprio	Inception	642.0	2803
179	Leonardo DiCaprio	The Revenant	556.0	1188
257	Leonardo DiCaprio	The Aviator	267.0	799
296	Leonardo DiCaprio	Django Unchained	765.0	1193
307	Leonardo DiCaprio	Blood Diamond	166.0	657
308	Leonardo DiCaprio	The Wolf of Wall Street	606.0	1138
326	Leonardo DiCaprio	Gangs of New York	233.0	1166
361	Leonardo DiCaprio	The Departed	352.0	2054
452	Leonardo DiCaprio	Shutter Island	490.0	964
641	Leonardo DiCaprio	Body of Lies	238.0	263
911	Leonardo DiCaprio	Catch Me If You Can	194.0	667
990	Leonardo DiCaprio	The Beach	118.0	548
	Leonardo			

1114	Leonardo DiCaprio	Revolutionary Road	323.0	414
1422	Leonardo DiCaprio	The Man in the Iron Mask	83.0	244
1453	Leonardo DiCaprio	J. Edgar	392.0	279
1560	Leonardo DiCaprio	The Quick and the Dead	63.0	216
2067	Leonardo DiCaprio	Marvin's Room	45.0	71
2757	Leonardo DiCaprio	Romeo + Juliet	106.0	506

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

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x7f892bb43650>
```

```
# Write the code for finding the mean of critic reviews and audience reviews here

# mean of critic reviews
Critic_reviews=Actor_name['num_critic_for_reviews'].mean().sort_values(ascending=False)
print(Critic_reviews.head())
print()
# mean of audience reviews
Audience_reviews=Actor_name['num_user_for_reviews'].mean().sort_values(ascending=False)
print(Audience_reviews.head())
```

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

actor_1_name
Leonardo DiCaprio    1.204168e+67
Brad Pitt            4.836291e+49
Meryl Streep         1.951753e+30
Name: num_user_for_reviews, dtype: float64
```

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

```
# Write the code for calculating decade here
movies['decade']=movies['title_year'].apply(lambda x: (x//10) *10).astype(np.int64)  #asty
movies['decade']=movies['decade'].astype(str)+'s'    #astype(str)+'s' to add s to decade


movies=movies.sort_values(['decade'])
movies
```

director_name num_critic_for_reviews gross genres actor_1_name movie

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_
4812	Harry Beaumont	36.0	2.81	[Musical, Romance]	Anita Page	Bro M
4958	Harry F. Millarde	1.0	3.00	[Crime, Drama]	Stephen Carr	Over t Poor
2734	Fritz Lang	260.0	0.03	[Drama, Sci-Fi]	Brigitte Helm	Met
4157	Victor Fleming	213.0	22.20	[Adventure, Family, Fantasy, Musical]	Margaret Hamilton	The \
4706	Mark Sandrich	66.0	3.00	[Comedy, Musical, Romance]	Ginger Rogers	T
...	...	...	...	...	...	
3470	Steven Soderbergh	324.0	113.71	[Comedy, Drama]	Channing Tatum	Magi
781	Martin Campbell	258.0	43.29	[Crime, Drama, Mystery, Thriller]	Bojana Novakovic	E Da
2495	Malcolm D. Lee	56.0	70.49	[Comedy, Drama]	Harold Perrineau	Th Man H
1668	Steven Soderbergh	450.0	32.15	[Crime, Drama, Thriller]	Channing Tatum	Side I
3264	Michael Haneke	447.0	0.23	[Drama, Romance]	Isabelle Huppert	

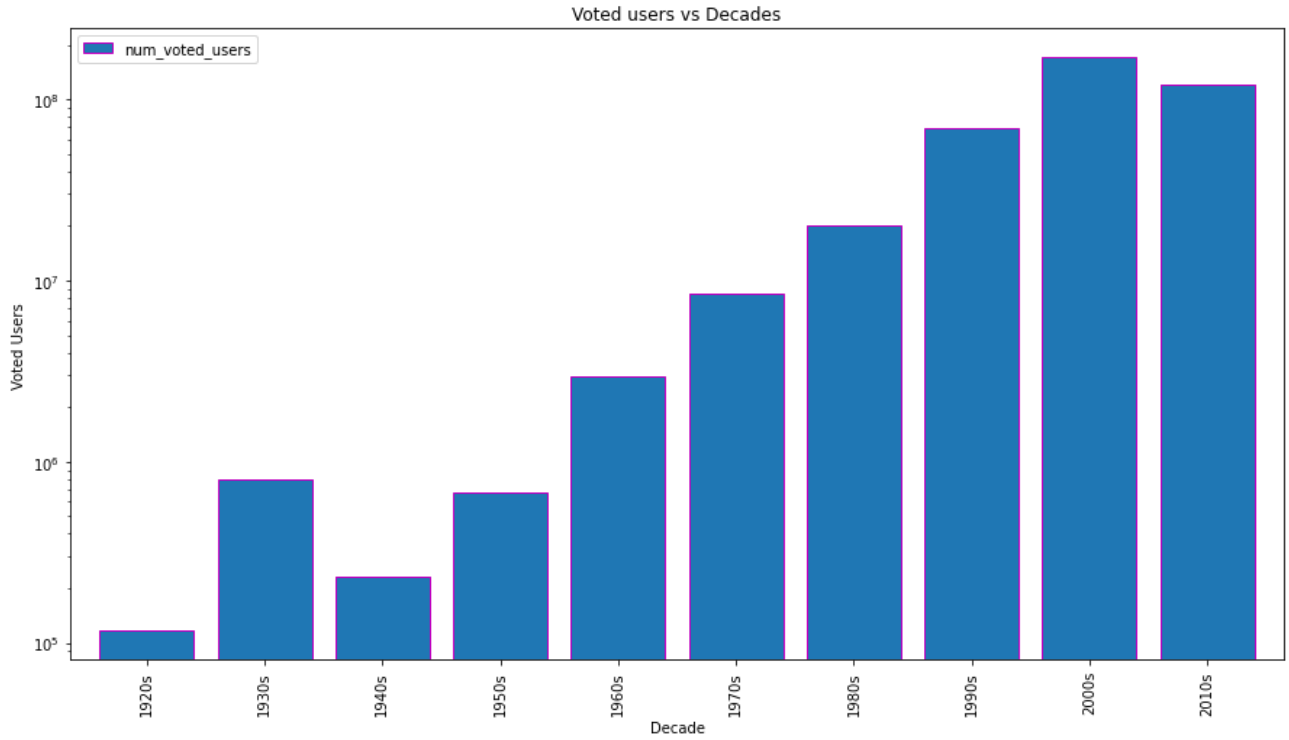
3856 rows × 17 columns

```
# Write your code for creating the data frame df_by_decade
df_by_decade=movies.groupby('decade')
df_by_decade['num_voted_users'].sum()
#Convert to Dafaframe
df_by_decade=pd.DataFrame(df_by_decade['num_voted_users'].sum())
df_by_decade
```

decade	num_voted_users

1920s	116392
1930s	804839
1940s	230838
1950s	678336
1960s	2983442
1970s	8524102
1980s	19987476
1990s	69735679

```
# Write your code for plotting number of voted users vs decade
df_by_decade.plot.bar(figsize=(15,8),width=0.8,edgecolor='m')   #Figure size, width of bar
plt.xlabel("Decade")
plt.ylabel("Voted Users")
plt.title("Voted users vs Decades")
plt.yscale('log')  #Changing the Y scale to see the actual difference
plt.show()
```



✓ 0s completed at 3:16 AM