IMDB Movie Analysis

Final Project-1

Description:

For the Final Project - 1, we are provided with a dataset having various columns of different IMDB Movies. We are required to Frame the problem. For this task, we will need to define a problem we want to shed some light on.

Project Approach Used:

This project is quite challenging and different from the type of project I have worked on previously, provided by the team. I am very happy to work on this project and finish it by bringing out insights that will be useful for the company to make better decisions.

Tech Stack Used:

In this project, I used

1. **Python,**
2. **Google Collab and**
3. **MS Excel**

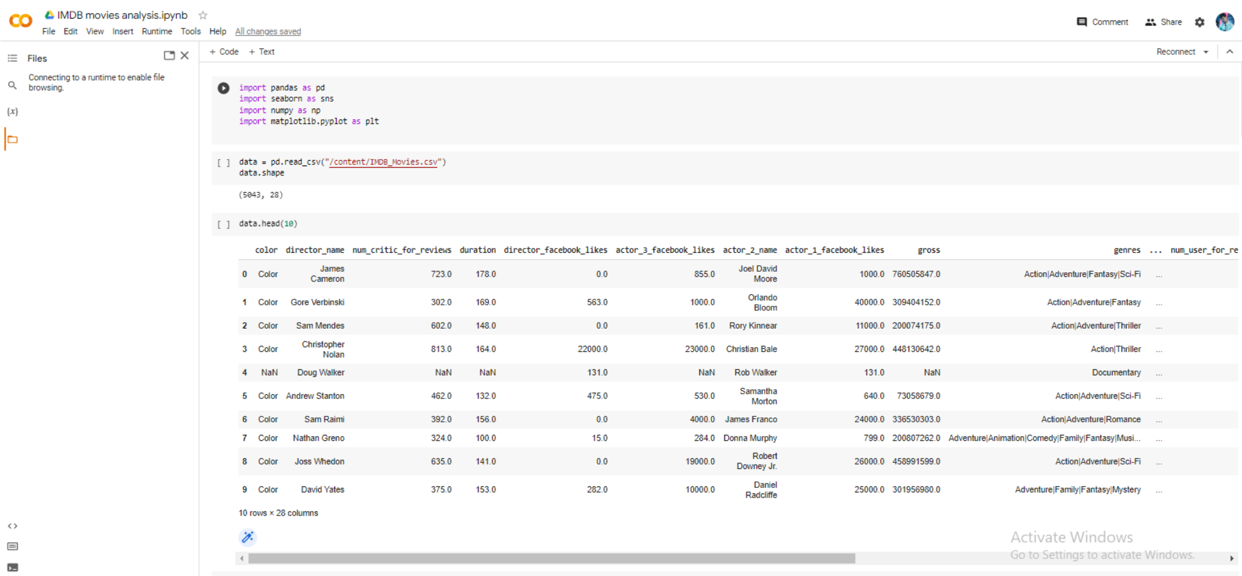
To solve the given problems.

In this project, I achieved some new things like how to get results from huge amounts of data.

The dataset provided by the team has various columns of different IMDB movies. First I started working on the 5 WHY aspect of the dataset.

1. Cleaning the Data

First, we start by exploring the dataset.



IMDB movies analysis.ipynb

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

```
data = pd.read_csv("/content/IMDB_Movies.csv")
data.shape
```

```
(5043, 28)
```

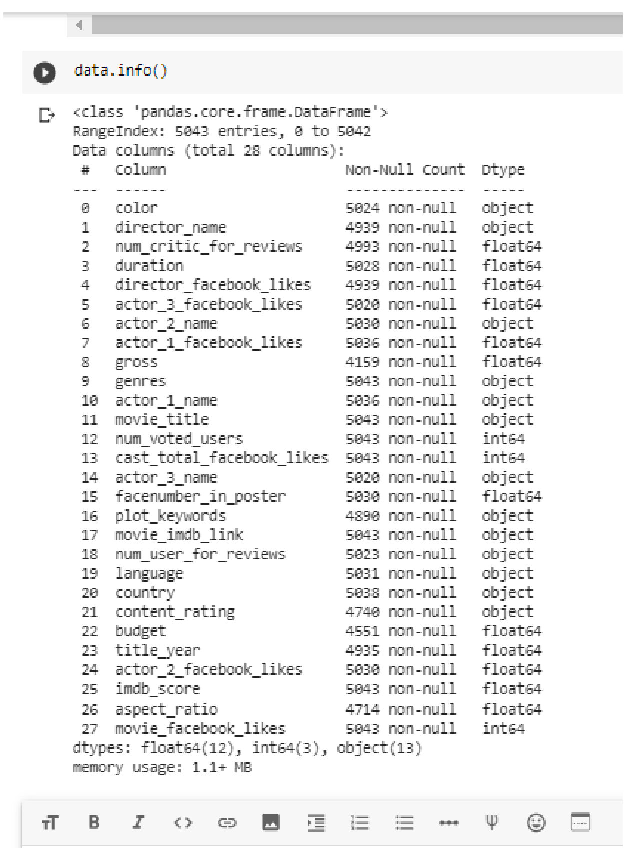
```
data.head(10)
```

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	actor_1_facebook_likes	gross	genres	...	num_user_for_re
0	Color	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore	1000.0	760505847.0	Action|Adventure|Fantasy|Sci-Fi	...	
1	Color	Gore Verbinski	302.0	169.0	563.0	1000.0	Orlando Bloom	40000.0	309404152.0	Action|Adventure|Fantasy	...	
2	Color	Sam Mendes	602.0	148.0	0.0	161.0	Rory Kinnear	11000.0	200074175.0	Action|Adventure|Thriller	...	
3	Color	Christopher Nolan	813.0	164.0	22000.0	23000.0	Christian Bale	27000.0	448130642.0	Action|Thriller	...	
4	NaN	Doug Walker	NaN	NaN	131.0	NaN	Rob Walker	131.0	NaN	Documentary	...	
5	Color	Andrew Stanton	462.0	132.0	475.0	530.0	Samantha Morton	640.0	73058679.0	Action|Adventure|Sci-Fi	...	
6	Color	Sam Raimi	392.0	156.0	0.0	4000.0	James Franco	24000.0	336530303.0	Action|Adventure|Romance	...	
7	Color	Nathan Greno	324.0	100.0	15.0	284.0	Donna Murphy	799.0	200807262.0	Adventure|Animation|Comedy|Family|Fantasy|Musi...	...	
8	Color	Joss Whedon	635.0	141.0	0.0	19000.0	Robert Downey Jr.	26000.0	458991599.0	Action|Adventure|Sci-Fi	...	
9	Color	David Yates	375.0	153.0	282.0	10000.0	Daniel Radcliffe	25000.0	301956980.0	Adventure|Family|Fantasy|Mystery	...	

10 rows × 28 columns

Then we tried to find out some information regarding the dataset.

Using – data.info() command.



```
data.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
```

Then we tried to find out the number of unique rows in each feature.

Using – data.nunique().sort_values() command

```
#checking the number of of unique rows in each feature
data.nunique().sort_values()
```

```
color                          2
content_rating                18
facenumber_in_poster          19
aspect_ratio                  22
language                      47
country                       65
imdb_score                    78
title_year                    91
duration                     191
director_facebook_likes      435
budget                       439
num_critic_for_reviews       528
movie_facebook_likes         876
actor_1_facebook_likes       878
actor_3_facebook_likes       906
genres                       914
actor_2_facebook_likes       917
num_user_for_reviews         955
actor_1_name                2097
director_name               2398
actor_2_name                3032
actor_3_name                3521
cast_total_facebook_likes   3978
gross                       4035
plot_keywords               4760
num_voted_users             4826
movie_title                 4917
movie_imdb_link             4919
dtype: int64
```

After that we tried to find out the missing values are available or not, and if available we printed it.

Using the command – data.isnull()

```
# Check the missing values are avalable or not
print ("Any missing value?",data.isnull().values.any())
```

```
Any missing value? True
```

```
data.isnull()
```

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	actor_1_facebook_likes	gross	genres	...	num_user_for_reviews	language	country	content_rating
0	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False
4	True	False	True	True	False	True	False	False	True	False	...	False	True	True	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
5038	False	False	False	False	False	False	False	False	True	False	...	False	False	False	True
5039	False	True	False	False	True	False	False	False	True	False	...	False	False	False	False
5040	False	False	False	False	False	False	False	False	True	False	...	False	False	False	True
5041	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False
5042	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False

5043 rows × 28 columns

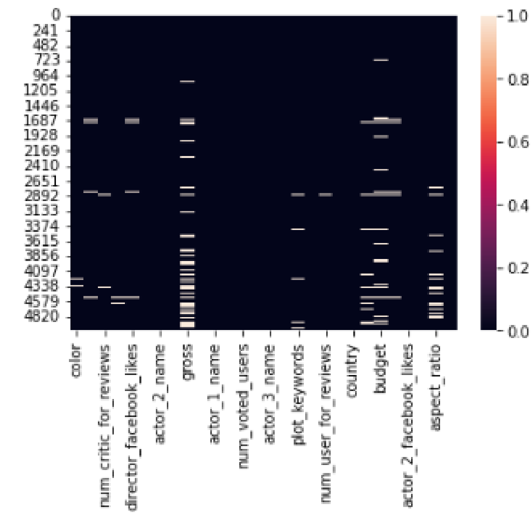
Then we tried to establish the columns with missing values with their respective sum.

```python
#Finding the number of missing values in every variable
data.isnull().sum().sort_values(ascending = False)
```

```
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
```

```python
sns.heatmap(data.isnull())
```

```
<Axes: >
```




```python
#Trying to find missing values in percentagte

per_missing = data.isnull().sum().sort_values(ascending = False) * 100 / len(data)
per_missing
```

```
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
```

Here, we showed the percentage of missing values in each column.

Gross having highest missing values, followed by budget and aspect_ratio.

Till now the dataset have **5043 rows × 28 columns**, altogether including all the missing values, duplicate values and the unnecessary columns not needed for our desired results.

Now we progressed towards removing or dropping the missing values.

```
[ ] # Drop missing values

    data.dropna(axis = 0, inplace = True)
    data.shape

    (3756, 28)
```

After dropping the missing values finally we are left with **3756 rows * 28 columns.**

Later we progressed towards identifying is there any duplicate values available, and if available remove from the dataset for cleaning the dataset.

```
# Check for duplicate dataset

dup_data=data.duplicated().any()

print ("Are there any duplicate values?",dup_data)

Are there any duplicate values? True
```

```
[ ] #since we dont have any duplicate data in our dataframe, we no need to process it further for duplicate data.
    # If any duplicate data was present , we coud have use the data.drop_duplicated() fuinction

    data = data.drop_duplicates()
    data.shape

    (3723, 28)
```

After dropping the duplicate values finally we are left with **3723 rows * 28 columns.**

Now dropping unnecessary columns, which is not required for our work.

Initial columns = 28

```
data.keys()

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')
```

After deleting unnecessary columns left = 17

```
#Now removing unnecessary columns

cleaned_data = data.drop(['color', 'director_facebook_likes', 'actor_3_facebook_likes',
       'actor_1_facebook_likes', 'facenumber_in_poster', 'plot_keywords',
       'movie_imdb_link', 'actor_2_facebook_likes', 'cast_total_facebook_likes',
       'aspect_ratio', 'movie_facebook_likes'], axis = 1)

cleaned_data
```

	director_name	num_critic_for_reviews	duration	actor_2_name	gross	genres	actor_1_name	movie_title	num_voted_users	actor_3_name	num_user_for_reviews	language	country	content_rating	budget	title_year	imdb_
0	James Cameron	723.0	178.0	Joel David Moore	760505847.0	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	Avatar	886204	Wes Studi	3054	English	USA	PG-13	237000000.0	2009.0	
1	Gore Verbinski	302.0	169.0	Orlando Bloom	309404152.0	Action|Adventure|Fantasy	Johnny Depp	Pirates of the Caribbean: At World's End	471220	Jack Davenport	1238	English	USA	PG-13	300000000.0	2007.0	
2	Sam Mendes	602.0	148.0	Rory Kinnear	200074175.0	Action|Adventure|Thriller	Christoph Waltz	Spectre	275868	Stephanie Sigman	994	English	UK	PG-13	245000000.0	2015.0	
3	Christopher Nolan	813.0	164.0	Christian Bale	448130642.0	Action|Thriller	Tom Hardy	The Dark Knight Rises	1144337	Joseph Gordon-Levitt	2701	English	USA	PG-13	250000000.0	2012.0	
5	Andrew Stanton	462.0	132.0	Samantha Morton	73058679.0	Action|Adventure|Sci-Fi	Daryl Sabara	John Carter	212204	Polly Walker	738	English	USA	PG-13	263700000.0	2012.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
5026	Olivier Assayas	81.0	110.0	Béatrice Dalle	136007.0	Drama|Music|Romance	Maggie Cheung	Clean	3924	Don McKellar	39	French	France	R	4500.0	2004.0	
5027	Jafar Panahi	64.0	90.0	Nargess Mamizadeh	673780.0	Drama	Fereshteh Sadre Orafaiy	The Circle	4555	Mojgan Faramarzi	26	Persian	Iran	Not Rated	10000.0	2000.0	
5033	Shane Carruth	143.0	77.0	David Sullivan	424760.0	Drama|Sci-Fi|Thriller	Shane Carruth	Primer	72639	Casey Gooden	371	English	USA	PG-13	7000.0	2004.0	
5035	Robert Rodriguez	56.0	81.0	Peter Marquardt	2040920.0	Action|Crime|Drama|Romance|Thriller	Carlos Gallardo	El Mariachi	52055	Consuelo Gómez	130	Spanish	USA	R	7000.0	1992.0	
5042	Jon Gunn	43.0	90.0	Brian Herzlinger	85222.0	Documentary	John August	My Date with Drew	4285	Jon Gunn	84	English	USA	PG	1100.0	2004.0	

3723 rows × 17 columns

Finally to make the data more readable and usable we re-ordered the columns.

'movie_title', 'director_name', 'actor_1_name', 'actor_2_name', 'actor_3_name','genres', 'country','language', 'content_rating', 'title_year', 'duration', 'num_critic_for_reviews','num_user_for_reviews', 'num_voted_users', 'imdb_score', 'budget', 'gross',

```
# altering the DataFrame
ordered_cleaned_data = cleaned_data[['movie_title', 'director_name',  'actor_1_name',  'actor_2_name', 'actor_3_name',
                                     'genres', 'country','language', 'content_rating', 'title_year', 'duration', 'num_critic_for_reviews',
                                     'num_user_for_reviews', 'num_voted_users',  'imdb_score', 'budget', 'gross',
                                    ]]
ordered_cleaned_data
```

	movie_title	director_name	actor_1_name	actor_2_name	actor_3_name	genres	country	language	content_rating	title_year	duration	num_critic_for_reviews	num_user_for_reviews	num_voted_users	imdb_score	budget	
0	Avatar	James Cameron	CCH Pounder	Joel David Moore	Wes Studi	Action|Adventure|Fantasy|Sci-Fi	USA	English	PG-13	2009.0	178.0	723.0	3054	886204	7.9	237000000.0	760505
1	Pirates of the Caribbean: At World's End	Gore Verbinski	Johnny Depp	Orlando Bloom	Jack Davenport	Action|Adventure|Fantasy	USA	English	PG-13	2007.0	169.0	302.0	1238	471220	7.1	300000000.0	309404
2	Spectre	Sam Mendes	Christoph Waltz	Rory Kinnear	Stephanie Sigman	Action|Adventure|Thriller	UK	English	PG-13	2015.0	148.0	602.0	994	275868	6.8	245000000.0	200074
3	The Dark Knight Rises	Christopher Nolan	Tom Hardy	Christian Bale	Joseph Gordon-Levitt	Action|Thriller	USA	English	PG-13	2012.0	164.0	813.0	2701	1144337	8.5	250000000.0	448130
5	John Carter	Andrew Stanton	Daryl Sabara	Samantha Morton	Polly Walker	Action|Adventure|Sci-Fi	USA	English	PG-13	2012.0	132.0	462.0	738	212204	6.6	263700000.0	73058
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
5026	Clean	Olivier Assayas	Maggie Cheung	Béatrice Dalle	Don McKellar	Drama|Music|Romance	France	French	R	2004.0	110.0	81.0	39	3924	6.9	4500.0	136
5027	The Circle	Jafar Panahi	Fereshteh Sadre Orafaiy	Nargess Mamizadeh	Mojgan Faramarzi	Drama	Iran	Persian	Not Rated	2000.0	90.0	64.0	26	4555	7.5	10000.0	673
5033	Primer	Shane Carruth	Shane Carruth	David Sullivan	Casey Gooden	Drama|Sci-Fi|Thriller	USA	English	PG-13	2004.0	77.0	143.0	371	72639	7.0	7000.0	424
5035	El Mariachi	Robert Rodriguez	Carlos Gallardo	Peter Marquardt	Consuelo Gómez	Action|Crime|Drama|Romance|Thriller	USA	Spanish	R	1992.0	81.0	56.0	130	52055	6.9	7000.0	2040
5042	My Date with Drew	Jon Gunn	John August	Brian Herzlinger	Jon Gunn	Documentary	USA	English	PG	2004.0	90.0	43.0	84	4285	6.6	1100.0	85

3723 rows × 17 columns

B. Movies with highest profit:

Here, I need to create a new column called profit, which contains the difference of the two columns: gross and budget. Sort the column using the profit column as reference. Plot profit (y-axis) vs budget (x- axis) and observe the outliers using the appropriate chart type.

2. Movies with Highest Profit

```
[66] ordered_cleaned_data['budget'] =  ordered_cleaned_data['budget'] / 1000000
     ordered_cleaned_data['gross'] =  ordered_cleaned_data['gross'] / 1000000
```

```
ordered_cleaned_data
```

:tor_1_name	actor_2_name	actor_3_name	genres	country	language	content_rating	title_year	duration	num_critic_for_reviews	num_user_for_reviews	num_voted_users	imdb_score	budget	gross
:CH Pounder	Joel David Moore	Wes Studi	Action|Adventure|Fantasy|Sci-Fi	USA	English	PG-13	2009.0	178.0	723.0	3054	886204	7.9	237.0000	760.505847
Johnny Depp	Orlando Bloom	Jack Davenport	Action|Adventure|Fantasy	USA	English	PG-13	2007.0	169.0	302.0	1238	471220	7.1	300.0000	309.404152
Christoph Waltz	Rory Kinnear	Stephanie Sigman	Action|Adventure|Thriller	UK	English	PG-13	2015.0	148.0	602.0	994	275868	6.8	245.0000	200.074175
Tom Hardy	Christian Bale	Joseph Gordon-Levitt	Action|Thriller	USA	English	PG-13	2012.0	164.0	813.0	2701	1144337	8.5	250.0000	448.130642
Daryl Sabara	Samantha Morton	Polly Walker	Action|Adventure|Sci-Fi	USA	English	PG-13	2012.0	132.0	462.0	738	212204	6.6	263.7000	73.058679
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
Maggie Cheung	Béatrice Dalle	Don McKellar	Drama|Music|Romance	France	French	R	2004.0	110.0	81.0	39	3924	6.9	0.0045	0.136007
Fereshteh ;adre Orafaiy	Nargess Mamizadeh	Mojgan Faramarzi	Drama	Iran	Persian	Not Rated	2000.0	90.0	64.0	26	4555	7.5	0.0100	0.673780
hane Carruth	David Sullivan	Casey Gooden	Drama|Sci-Fi|Thriller	USA	English	PG-13	2004.0	77.0	143.0	371	72639	7.0	0.0070	0.424760
Carlos Gallardo	Peter Marquardt	Consuelo Gómez	Action|Crime|Drama|Romance|Thriller	USA	Spanish	R	1992.0	81.0	56.0	130	52055	6.9	0.0070	2.040920
John August	Brian Herzlinger	Jon Gunn	Documentary	USA	English	PG	2004.0	90.0	43.0	84	4285	6.6	0.0011	0.085222

Here my task is to find the movies with the highest profit.

We found out that “Avatar “ is the highest profit generating movie according to given dataset, with a total profit of **$523.505847 million.**

```
#movies with highest profit

ordered_cleaned_data['profit'] = ordered_cleaned_data['gross'] - ordered_cleaned_data['budget']
ordered_cleaned_data
```

title	director_name	actor_1_name	actor_2_name	actor_3_name	genres	country	language	content_rating	title_year	duration	num_critic_for_reviews	num_user_for_reviews	num_voted_users	imdb_score	budget	gross	profit
\vatar	James Cameron	CCH Pounder	Joel David Moore	Wes Studi	Action|Adventure|Fantasy|Sci-Fi	USA	English	PG-13	2009.0	178.0	723.0	3054	886204	7.9	237.0000	760.505847	523.505847
of the ibean: /orld's End	Gore Verbinski	Johnny Depp	Orlando Bloom	Jack Davenport	Action|Adventure|Fantasy	USA	English	PG-13	2007.0	169.0	302.0	1238	471220	7.1	300.0000	309.404152	9.404152
pectre	Sam Mendes	Christoph Waltz	Rory Kinnear	Stephanie Sigman	Action|Adventure|Thriller	UK	English	PG-13	2015.0	148.0	602.0	994	275868	6.8	245.0000	200.074175	-44.925825
: Dark Rises	Christopher Nolan	Tom Hardy	Christian Bale	Joseph Gordon-Levitt	Action|Thriller	USA	English	PG-13	2012.0	164.0	813.0	2701	1144337	8.5	250.0000	448.130642	198.130642
Carter	Andrew Stanton	Daryl Sabara	Samantha Morton	Polly Walker	Action|Adventure|Sci-Fi	USA	English	PG-13	2012.0	132.0	462.0	738	212204	6.6	263.7000	73.058679	-190.641321
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
Clean	Olivier Assayas	Maggie Cheung	Béatrice Dalle	Don McKellar	Drama|Music|Romance	France	French	R	2004.0	110.0	81.0	39	3924	6.9	0.0045	0.136007	0.131507
Circle	Jafar Panahi	Fereshteh Sadre Orafaiy	Nargess Mamizadeh	Mojgan Faramarzi	Drama	Iran	Persian	Not Rated	2000.0	90.0	64.0	26	4555	7.5	0.0100	0.673780	0.663780
?rimer	Shane Carruth	Shane Carruth	David Sullivan	Casey Gooden	Drama|Sci-Fi|Thriller	USA	English	PG-13	2004.0	77.0	143.0	371	72639	7.0	0.0070	0.424760	0.417760
iriachi	Robert Rodriguez	Carlos Gallardo	Peter Marquardt	Consuelo Gómez	Action|Crime|Drama|Romance|Thriller	USA	Spanish	R	1992.0	81.0	56.0	130	52055	6.9	0.0070	2.040920	2.033920
e with Drew	Jon Gunn	John August	Brian Herzlinger	Jon Gunn	Documentary	USA	English	PG	2004.0	90.0	43.0	84	4285	6.6	0.0011	0.085222	0.084122

columns

C. Top 250:

Create a new column 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.

Extract all the movies in the IMDb_Top_250 column which are not in the English language and store them in a new column named Top_Foreign_Lang_Film. You can use your own imagination also!

Here my task is to Find IMDB Top 250.

a) Here are the list of top 250 IMDB movies with the highest IMDb Rating (corresponding to the column: imdb_score) – Eg.

The Shawshank Redemption
The Godfather
The Dark Knight
The Godfather: Part II
FargoÂ
The Lord of the Rings: The Return of the KingÂ

b) Also it was made sure that that for all of these movies, the num_voted_users is greater than 25,000.

c) And at last a rank column is also added demontratring each movies rank with other details.

d) Also, extraction of all the movies in the IMDb_Top_250 column which are not in the English language is performed and stored in a new column named Top_Foreign_Lang_Film. – **Buffy the Vampire Slayer**

e) Detailed result can be viewed in the excel file question 3 sheet.

	A	B	C	D	E	F	G
1	movie_title	num_voted_users	imdb_score	language	Ranking		Top_Foreign_Lang_Film
3	The Shawshank Redemption	1689764	9.3	English	1		
4	The Godfather	1155770	9.2	English	2		Buffy the Vampire SlayerÂ
8	The Dark Knight	1676169	9	English	3		
9	The Godfather: Part II	790926	9	English	4		
10	FargoÂ	170055	9	English	5		
11	The Lord of the Rings: The Return of the KingÂ	1215718	8.9	English	6		
12	Pulp FictionÂ	1324680	8.9	English	7		
13	Schindler's ListÂ	865020	8.9	English	8		
14	The Good, the Bad and the UglyÂ	503509	8.9	English	9		
15	12 Angry MenÂ	447785	8.9	English	10		
16	Forrest GumpÂ	1251222	8.8	English	11		
17	Star Wars: Episode V - The Empire Strikes BackÂ	837759	8.8	English	12		
18	The Lord of the Rings: The Fellowship of the RingÂ	1238746	8.8	English	13		
19	InceptionÂ	1468200	8.8	English	14		
20	DaredevilÂ	213483	8.8	English	15		
21	It's Always Sunny in PhiladelphiaÂ	133415	8.8	English	16		
22	Fight ClubÂ	1347461	8.8	English	17		
23	Star Wars: Episode IV - A New HopeÂ	911097	8.7	English	18		
24	The Lord of the Rings: The Two TowersÂ	1100446	8.7	English	19		
25	The MatrixÂ	1217752	8.7	English	20		
26	One Flew Over the Cuckoo's NestÂ	680041	8.7	English	21		
27	GoodfellasÂ	728685	8.7	English	22		
28	City of GodÂ	533200	8.7	English	23		
29	Friday Night LightsÂ	42746	8.7	English	24		
35	Seven SamuraiÂ	229012	8.7	English	25		
36	Saving Private RyanÂ	881236	8.6	English	26		
37	The Silence of the LambsÂ	887467	8.6	English	27		
38	Se7enÂ	1023511	8.6	English	28		
39	InterstellarÂ	928227	8.6	English	29		

imdb

D. Best Directors:

Group the column using the director_name column.

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

Here my task is to find the best directors.

The top 10 best directors are

```
director_name         Mean_imdb_scores
1.Akira Kurosawa          8.700000
2.Charles Chaplin         8.600000
3.Tony Kaye               8.600000
4.Damien Chazelle         8.500000
5.Majid Majidi            8.500000
6.Alfred Hitchcock        8.500000
7.Ron Fricke              8.500000
8.Sergio Leone            8.433333
9.Christopher Nolan       8.425000
10.Richard Marquand       8.400000
```

Q4. Best Director

```
#Finding the best directors
ordered_cleaned_data.groupby('director_name').imdb_score.mean().sort_values(ascending = False)
```

```
director_name
Akira Kurosawa        8.7
Charles Chaplin       8.6
Tony Kaye             8.6
Damien Chazelle       8.5
Majid Majidi          8.5
                     ...
Aaron Seltzer         2.7
Jason Friedberg       2.6
Roger Christian       2.4
Alex Zamm             2.3
Vondie Curtis-Hall    2.1
Name: imdb_score, Length: 1659, dtype: float64
```

```
[69] top_10_directors = ordered_cleaned_data.groupby('director_name').imdb_score.mean().sort_values(ascending = False).head(10)
     top_10_directors
```

```
director_name
Akira Kurosawa        8.700000
Charles Chaplin       8.600000
Tony Kaye             8.600000
Damien Chazelle       8.500000
Majid Majidi          8.500000
Alfred Hitchcock      8.500000
Ron Fricke            8.500000
Sergio Leone          8.433333
Christopher Nolan     8.425000
Richard Marquand      8.400000
Name: imdb_score, dtype: float64
```

E. Popular Genres:

Perform this step using the knowledge gained while performing previous steps.

Here our work is to find popular genres.

These are the genres available in the IMDB movies database.

Adventure

Action

Fantasy etc.

```
popular_genres = ordered_cleaned_data.genres.str.split('|', expand = True)
popular_genres
```

	0	1	2	3	4	5	6	7
0	Action	Adventure	Fantasy	Sci-Fi	None	None	None	None
1	Action	Adventure	Fantasy	None	None	None	None	None
2	Action	Adventure	Thriller	None	None	None	None	None
3	Action	Thriller	None	None	None	None	None	None
5	Action	Adventure	Sci-Fi	None	None	None	None	None
...	...	...	...	...	...	...	...	...
5026	Drama	Music	Romance	None	None	None	None	None
5027	Drama	None	None	None	None	None	None	None
5033	Drama	Sci-Fi	Thriller	None	None	None	None	None
5035	Action	Crime	Drama	Romance	Thriller	None	None	None
5042	Documentary	None	None	None	None	None	None	None

3723 rows × 8 columns

```python
genres = []

for i in data['genres']:
    genres += i.split('|')

unique_gen = list(set(genres))
unique_gen
```

Following are the unique geners that are listed in the IMDB mivies database:

Western

Sport

Drama

Action

Adventure etc.

```
['Western',
 'Sport',
 'Comedy',
 'Crime',
 'Musical',
 'Horror',
 'Mystery',
 'War',
 'Family',
 'Music',
 'Adventure',
 'Film-Noir',
 'Action',
 'Fantasy',
 'Biography',
 'Romance',
 'Sci-Fi',
 'Drama',
 'History',
 'Thriller',
 'Animation',
 'Documentary']
```

The following list is the Popularity of the various genres given in the Dataset

```python
for gen in unique_gen:
  c = 0
  for genres in data['genres']:
      if (gen in genres):
          c += 1

  print(gen , c)
```

```
Western 57
Sport 147
Comedy 1455
Crime 704
Musical 96
Horror 386
Mystery 378
War 150
Family 440
Music 231
Adventure 773
Film-Noir 1
Action 951
Fantasy 504
Biography 238
Romance 851
Sci-Fi 492
Drama 1876
History 147
Thriller 1105
Animation 196
Documentary 45
```

This the final list of unique genres with its popularity based on number of times that genre appeared in the different movies listed on the IMDB dataset given to us.

	A	B
1	Genres	Popularity
2	Drama	1876
3	Comedy	1455
4	Thriller	1105
5	Action	951
6	Romance	851
7	Adventure	773
8	Crime	704
9	Fantasy	504
10	Sci-Fi	492
11	Family	440
12	Horror	386
13	Mystery	378
14	Biography	238
15	Music	231
16	Animation	196
17	War	150
18	Sport	147
19	History	147
20	Musical	96
21	Western	57
22	Documentary	45
23	Film-Noir	1
24		

The following list is the Popularity of the various genres:

Drama - 1876

Comedy - 1455

Thriller – 1105

Action - 951

Conclusion:

According to the following result, Drama movies are the most popular movies followed by Comedy and Thriller movies.

F. Charts:

Create three new columns 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.

Append the rows of all these columns and store them in a new column named Combined.

Group the combined column using the actor_1_name column.

Find the mean of the num_critic_for_reviews and num_users_for_review and identify the actors which have the highest mean.

Observe the change in number of voted users over decades using a bar chart. Create a column called decade which represents the decade to which every movie belongs to. For example, the title_year year 1923, 1925 should be stored as 1920s. Sort the column based on the column decade, group it by decade and find the sum of users voted in each decade. Store this in a new data frame called df_by_decade.

Here our task is to find the critic-favorite and audience-favorite actors.

Solution:

List of, movies where Meryl Steep is the lead actor, along with user and critic reviews.

F. Charts:

[56]
```
# Write the code for creating three new dataframes here
Meryl_Streep=data[['actor_1_name','movie_title','num_critic_for_reviews','num_user_for_reviews']]
Leo_Caprio=data[['actor_1_name','movie_title','num_critic_for_reviews','num_user_for_reviews']]
Brad_Pitt=data[['actor_1_name','movie_title','num_critic_for_reviews','num_user_for_reviews']]

# 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

List of, movies where Leonardo DiCaprio is the lead actor, along with user and critic reviews.

```
# 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
179	Leonardo DiCaprio	The Revenant	556.0	1188
257	Leonardo DiCaprio	The Aviator	267.0	799

List of, movies where Leonardo DiCaprio is the lead actor, along with user and critic reviews.

```
[58] # 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

Grouping the combined data frame using the actor_1_name column.

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

Combined
```

```
<ipython-input-71-fd26915cbd35>:2: FutureWarning: The frame.append method is deprecated and will be removed from pandas i
  Combined=Meryl_Streep.append(Leo_Caprio).append(Brad_Pitt)
<ipython-input-71-fd26915cbd35>:2: FutureWarning: The frame.append method is deprecated and will be removed from pandas i
  Combined=Meryl_Streep.append(Leo_Caprio).append(Brad_Pitt)
```

	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

Find the mean of the num_critic_for_reviews and num_users_for_review and identify the actors which have the highest mean.

```
[63] # mean of audience reviews
     Audience_reviews=Actor_name['num_user_for_reviews'].mean().sort_values(ascending=False)
     Audience_reviews.head()
     #Leonardo has more Critic Reviews and Audience reveiws.
```

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

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

Sorting the data frame based on the column decade, and grouping it by decade and finding the sum of users voted in each decade.

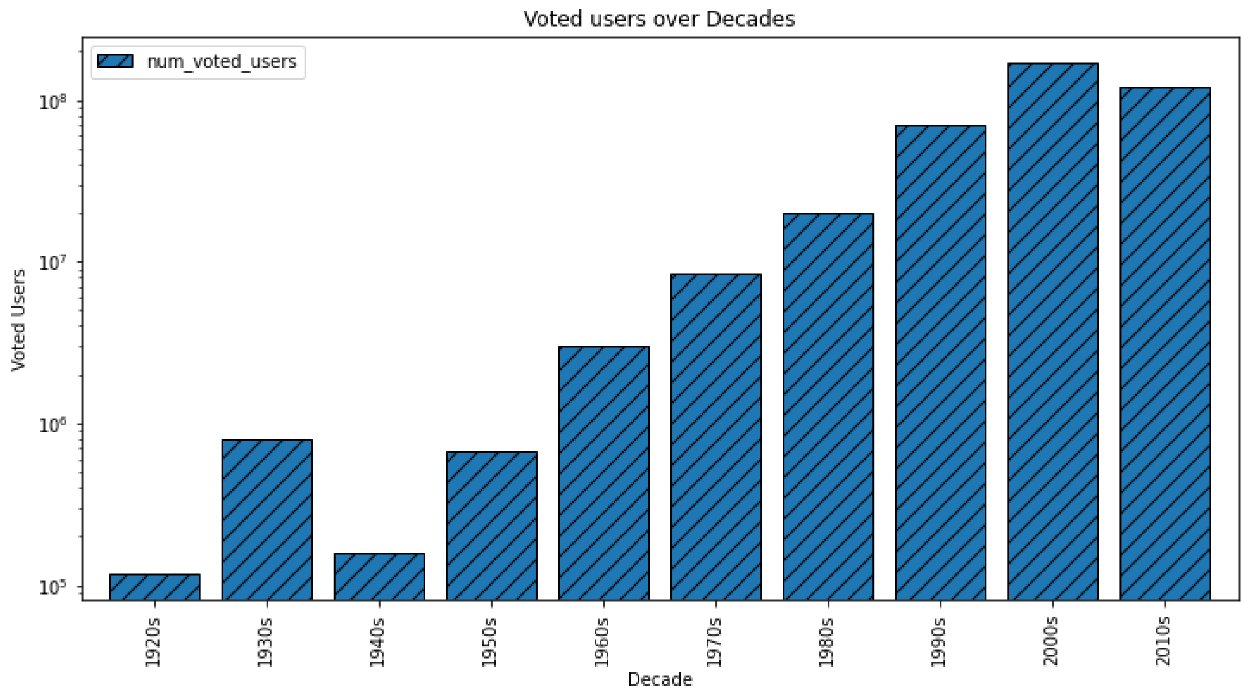
```
# Write your code for creating the data frame df_by_decade
df_by_decade = data.groupby('decade')
df_by_decade['num_voted_users'].sum()
#Convert to Dafaframe
df_by_decade=pd.DataFrame(df_by_decade['num_voted_users'].sum())
df_by_decade
```

decade	num_voted_users
1920s	116387
1930s	804839
1940s	159517
1950s	678336
1960s	2982551
1970s	8523299
1980s	19987476
1990s	69581866
2000s	170711435
2010s	119432961

Observing the change in the number of voted users over decades using a bar chart.

```
# Write your code for plotting number of voted users vs decade
import matplotlib.pyplot as plt

df_by_decade.plot.bar(figsize=(12,6),width=0.8,hatch="//",edgecolor='k')   #Figure size, width of
plt.xlabel("Decade")
plt.ylabel("Voted Users")
plt.title("Voted users over Decades")
plt.yscale('log')  #Changing y scale
plt.show()
```



Conclusion:

- Leonardo DiCaprio is the most voted actor by both metrics num_critic_for_reviews and num_users_for_review, followed by Brad Pitt and Meryl Streep.

- According to the bar chart, the 2010s was the decade when the most number users voted.

RESULT

In the making of this report, we used both of our Python and Microsoft Excel knowledge in a real-world example.

In this Project, I achieved Some new things like how to get results from huge amount of data.

DRIVE LINK

https://drive.google.com/drive/folders/1PixXwW7TEEnwKRBcR5FXddomipsKARzi?usp=share_link

For a further detailed report, please visit the. ipypnb file in the drive where I have uploaded the file where I built the project using Google collab.