IMDB Movie Analysis

--By Sagir Mehmood

Kindly download my Jyputer Notebook:

https://colab.research.google.com/drive/159hm2tehbj8bjAfWc4Ams73ETLaZC6TE?usp=sharing

Project Description:

This dataset having various columns of different IMDB Movies. I have to use my knowledge of statistics and use different formulas in excel and draw necessary conclusions about the data.

Approach:

I analyzed it in the manner described below:

- Data cleaning
- EDA
- Analysis and conclusion

Tech Used: Here I sought assistance from Python Jupyter Notebook.

Insights & Results:

QA. Data Cleaning:

1. Dropping unnecessary columns (Not needed for this project):

'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'

2. Removing the duplicate movie names, which are entered more than once.

```
]: mc=data['movie_title'].value_counts()

]: duplicates = mc[mc.values>1]
   duplicates
```

```
Eddie the Eagle                          2
The Lovers                               2
Creepshow                                2
Crossroads                               2
Hamlet                                   2
Death at a Funeral                       2
Point Break                              2
Hercules                                 2
My Soul to Take                          2
Big Fat Liar                             2
A Dog's Breakfast                        2
Lucky Number Slevin                      2
Dekalog                                  2
Carrie                                   2
Snitch                                   2
Twilight                                 2
Ghostbusters                             2
The Island                               2
The Day the Earth Stood Still            2
The Love Letter                          2
```

```
]: len(duplicates)

]: 119
```

```
9]: data.shape

9]: (5043, 13)

1]: data.drop_duplicates(subset ="movie_title", keep ='first', inplace = True)

2]: data.shape

2]: (4917, 13)
```

3. Checking for null values:

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

```
director_name              2.07
num_critic_for_reviews     1.00
gross                     17.55
genres                     0.00
actor_1_name               0.14
movie_title                0.00
num_voted_users            0.00
num_user_for_reviews       0.41
language                   0.24
budget                     9.84
title_year                 2.16
imdb_score                 0.00
movie_facebook_likes       0.00
dtype: float64
```

Here 'gross' and 'budget' columns have a large percentage (greater than 5%) of Null values. Here I dropped all the null rows in these two columns. Left are negligible.

After cleaning:

```
data= data[~np.isnan(data['gross'])]
data= data[~np.isnan(data['budget'])]
```

```
round(data.isnull().sum()/len(data)*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
```

4. Data formatting:

```
3]: data.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 3789 entries, 0 to 5042
Data columns (total 13 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   director_name          3789 non-null   object
 1   num_critic_for_reviews 3788 non-null   float64
 2   gross                  3789 non-null   float64
 3   genres                 3789 non-null   object
 4   actor_1_name           3786 non-null   object
 5   movie_title            3789 non-null   object
 6   num_voted_users        3789 non-null   int64
 7   num_user_for_reviews   3789 non-null   object
 8   language               3786 non-null   object
 9   budget                 3789 non-null   float64
 10  title_year             3789 non-null   float64
 11  imdb_score             3789 non-null   float64
 12  movie_facebook_likes   3789 non-null   int64
dtypes: float64(5), int64(2), object(6)
memory usage: 414.4+ KB
```

Here num_user_for_reviews column's data are stored in object format, and I converted it to float type.

After converting:

```
: data['num_user_for_reviews'] = data['num_user_for_reviews'].astype(float)

: data.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 3789 entries, 0 to 5042
Data columns (total 13 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   director_name          3789 non-null   object
 1   num_critic_for_reviews 3788 non-null   float64
 2   gross                  3789 non-null   float64
 3   genres                 3789 non-null   object
 4   actor_1_name           3786 non-null   object
 5   movie_title            3789 non-null   object
 6   num_voted_users        3789 non-null   int64
 7   num_user_for_reviews   3789 non-null   float64
 8   language               3786 non-null   object
 9   budget                 3789 non-null   float64
 10  title_year             3789 non-null   float64
 11  imdb_score             3789 non-null   float64
 12  movie_facebook_likes   3789 non-null   int64
dtypes: float64(6), int64(2), object(5)
memory usage: 414.4+ KB
```

B.

Movies with the highest profit:

Profit = Gross – Budget

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

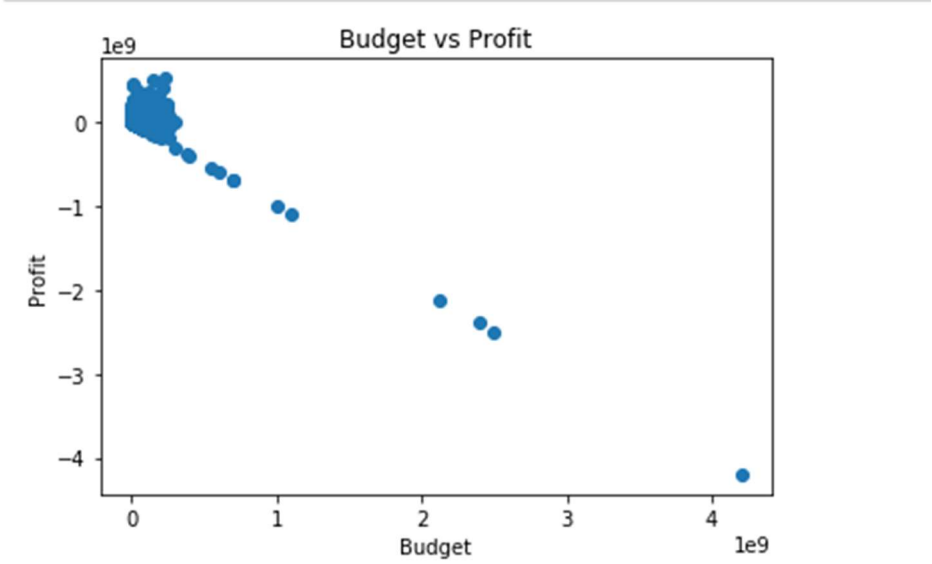
Here to 10 highest-earned movies are

```
]: top10 = data.sort_values(by='profit', ascending=False)[['movie_title','director_name','profit']].head(10)
   top10
```

]:

	movie_title	director_name	profit
0	Avatar	James Cameron	523505847.0
29	Jurassic World	Colin Trevorrow	502177271.0
26	Titanic	James Cameron	458672302.0
3024	Star Wars: Episode IV - A New Hope	George Lucas	449935665.0
3080	E.T. the Extra-Terrestrial	Steven Spielberg	424449459.0
17	The Avengers	Joss Whedon	403279547.0
509	The Lion King	Roger Allers	377783777.0
240	Star Wars: Episode I - The Phantom Menace	George Lucas	359544677.0
66	The Dark Knight	Christopher Nolan	348316061.0
439	The Hunger Games	Gary Ross	329999255.0

Plot profit (y-axis) vs budget (x-axis)



C.

Top 250 movies with the highest IMDb Rating for all of these movies, the num_voted_users is greater than 25,000:

```
In [35]: IMDB_Top_250 = data[data['num_voted_users']>25000][['movie_title','director_name','language','imdb_score']]
         IMDB_Top_250 = IMDB_Top_250.sort_values(by='imdb_score',ascending=False)
         IMDB_Top_250 = IMDB_Top_250.head(250)
         IMDB_Top_250['Rank'] = range(1,251)
         IMDB_Top_250
```

Out[35]:

	movie_title	director_name	language	imdb_score	Rank
1937	The Shawshank Redemption	Frank Darabont	English	9.3	1
3466	The Godfather	Francis Ford Coppola	English	9.2	2
66	The Dark Knight	Christopher Nolan	English	9.0	3
2837	The Godfather: Part II	Francis Ford Coppola	English	9.0	4
3355	Pulp Fiction	Quentin Tarantino	English	8.9	5
1874	Schindler's List	Steven Spielberg	English	8.9	6
339	The Lord of the Rings: The Return of the King	Peter Jackson	English	8.9	7
4498	The Good, the Bad and the Ugly	Sergio Leone	Italian	8.9	8
97	Inception	Christopher Nolan	English	8.8	9
2051	Star Wars: Episode V - The Empire Strikes Back	Irvin Kershner	English	8.8	10
683	Fight Club	David Fincher	English	8.8	11

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:

```
]: Top_Foreign_Lang_Film = IMDB_Top_250[IMDB_Top_250['language'] != 'English'].sort_values(by='imdb_score',ascending=False)
   Top_Foreign_Lang_Film
```

]:

	movie_title	director_name	language	imdb_score	Rank
4498	The Good, the Bad and the Ugly	Sergio Leone	Italian	8.9	8
4029	City of God	Fernando Meirelles	Portuguese	8.7	20
4747	Seven Samurai	Akira Kurosawa	Japanese	8.7	15
2373	Spirited Away	Hayao Miyazaki	Japanese	8.6	24
4259	The Lives of Others	Florian Henckel von Donnersmarck	German	8.5	35
4921	Children of Heaven	Majid Majidi	Persian	8.5	45
4105	Oldboy	Chan-wook Park	Korean	8.4	57
1298	Amélie	Jean-Pierre Jeunet	French	8.4	60
2323	Princess Mononoke	Hayao Miyazaki	Japanese	8.4	61
2970	Das Boot	Wolfgang Petersen	German	8.4	54
4659	A Separation	Asghar Farhadi	Persian	8.4	50
1329	Baahubali: The Beginning	S.S. Rajamouli	Telugu	8.4	47
4033	The Hunt	Thomas Vinterberg	Danish	8.3	78
2829	Downfall	Oliver Hirschbiegel	German	8.3	80
2734	Metropolis	Fritz Lang	German	8.3	82

D. Best Director (TOP-10):

```
top10director = pd.DataFrame(data.groupby('director_name')['imdb_score'].mean())
top10director = top10director.sort_values(by=['imdb_score','director_name'],ascending=[False,True])
top10director = top10director.head(10)
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
Asghar Farhadi	8.400000
Marius A. Markevicius	8.400000

E. Popular Genres:

```
PopularGenre = pd.DataFrame(data.groupby([ 'genres'])['imdb_score'].mean())
PopularGenre = PopularGenre.sort_values(by=['imdb_score','genres'],ascending=[False,True]).head(10)
PopularGenre
```

genres	imdb_score
Adventure|Animation|Drama|Family|Musical	8.50
Crime|Drama|Fantasy|Mystery	8.50
Action|Adventure|Drama|Fantasy|War	8.40
Adventure|Animation|Fantasy	8.40
Adventure|Drama|Thriller|War	8.40
Adventure|Animation|Comedy|Drama|Family|Fantasy	8.30
Biography|Drama|History|Music	8.30
Documentary|Drama|Sport	8.30
Documentary|War	8.30
Adventure|Drama|War	8.25

F. Charts:

Movies by Leonardo DiCaprio, Brad Pitt, and Meryl Streep:

Lead: Meryl Streep

Lead: Meryl Streepis

```
9]: Meryl_Streep = data[data['actor_1_name']=='Meryl Streep'][['movie_title','actor_1_name','num_critic_for_reviews','num_user_for_re
    Meryl_Streep
```

9]:

	movie_title	actor_1_name	num_critic_for_reviews	num_user_for_reviews
410	It's Complicated	Meryl Streep	187.0	214.0
1106	The River Wild	Meryl Streep	42.0	69.0
1204	Julie & Julia	Meryl Streep	252.0	277.0
1408	The Devil Wears Prada	Meryl Streep	208.0	631.0
1483	Lions for Lambs	Meryl Streep	227.0	298.0
1575	Out of Africa	Meryl Streep	66.0	200.0
1618	Hope Springs	Meryl Streep	234.0	178.0
1674	One True Thing	Meryl Streep	64.0	112.0
1925	The Hours	Meryl Streep	174.0	660.0
2781	The Iron Lady	Meryl Streep	331.0	350.0
3135	A Prairie Home Companion	Meryl Streep	211.0	280.0

Lead: Leonardo Di Caprio

Lead: Leonardo DiCaprio

```
: Leo_Caprio =data[data['actor_1_name']=='Leonardo DiCaprio'][['movie_title','actor_1_name','num_critic_for_reviews','num_user_for_
  Leo_Caprio
```

:

	movie_title	actor_1_name	num_critic_for_reviews	num_user_for_reviews
26	Titanic	Leonardo DiCaprio	315.0	2528.0
50	The Great Gatsby	Leonardo DiCaprio	490.0	753.0
97	Inception	Leonardo DiCaprio	642.0	2803.0
179	The Revenant	Leonardo DiCaprio	556.0	1188.0
257	The Aviator	Leonardo DiCaprio	267.0	799.0
296	Django Unchained	Leonardo DiCaprio	765.0	1193.0
307	Blood Diamond	Leonardo DiCaprio	166.0	657.0
308	The Wolf of Wall Street	Leonardo DiCaprio	606.0	1138.0
326	Gangs of New York	Leonardo DiCaprio	233.0	1166.0
361	The Departed	Leonardo DiCaprio	352.0	2054.0
452	Shutter Island	Leonardo DiCaprio	490.0	964.0
641	Body of Lies	Leonardo DiCaprio	238.0	263.0
911	Catch Me If You Can	Leonardo DiCaprio	194.0	667.0

Lead: Brad Pitt

Lead: Brad Pitt

```
.]: Brad_Pitt= data[data['actor_1_name']=='Brad Pitt'][['movie_title','actor_1_name','num_critic_for_reviews','num_user_for_reviews']
Brad_Pitt
```

.]:

	movie_title	actor_1_name	num_critic_for_reviews	num_user_for_reviews
101	The Curious Case of Benjamin Button	Brad Pitt	362.0	822.0
147	Troy	Brad Pitt	220.0	1694.0
254	Ocean's Twelve	Brad Pitt	198.0	627.0
255	Mr. & Mrs. Smith	Brad Pitt	233.0	798.0
382	Spy Game	Brad Pitt	142.0	361.0
400	Ocean's Eleven	Brad Pitt	186.0	845.0
470	Fury	Brad Pitt	406.0	701.0
611	Seven Years in Tibet	Brad Pitt	76.0	119.0
683	Fight Club	Brad Pitt	315.0	2968.0
792	Sinbad: Legend of the Seven Seas	Brad Pitt	98.0	91.0
940	Interview with the Vampire: The Vampire Chroni...	Brad Pitt	120.0	406.0
1490	The Tree of Life	Brad Pitt	584.0	975.0
1722	The Assassination of Jesse James by the Coward...	Brad Pitt	273.0	415.0

COMBINED:

```
[46]: Combined=pd.concat([Meryl_Streep,Leo_Caprio,Brad_Pitt])
Combined
```

[46]:

	movie_title	actor_1_name	num_critic_for_reviews	num_user_for_reviews
410	It's Complicated	Meryl Streep	187.0	214.0
1106	The River Wild	Meryl Streep	42.0	69.0
1204	Julie & Julia	Meryl Streep	252.0	277.0
1408	The Devil Wears Prada	Meryl Streep	208.0	631.0
1483	Lions for Lambs	Meryl Streep	227.0	298.0
1575	Out of Africa	Meryl Streep	66.0	200.0
1618	Hope Springs	Meryl Streep	234.0	178.0
1674	One True Thing	Meryl Streep	64.0	112.0
1925	The Hours	Meryl Streep	174.0	660.0
2781	The Iron Lady	Meryl Streep	331.0	350.0
3135	A Prairie Home Companion	Meryl Streep	211.0	280.0
26	Titanic	Leonardo DiCaprio	315.0	2528.0
50	The Great Gatsby	Leonardo DiCaprio	490.0	753.0
97	Inception	Leonardo DiCaprio	642.0	2803.0
179	The Revenant	Leonardo DiCaprio	556.0	1188.0
257	The Aviator	Leonardo DiCaprio	267.0	799.0

The mean of critic reviews and audience reviews from the above data:

```
a = pd.DataFrame(Combined.groupby('actor_1_name')['num_critic_for_reviews'].mean())
a.sort_values(by='num_critic_for_reviews',ascending=False)
```

actor_1_name	num_critic_for_reviews
Leonardo DiCaprio	322.200000
Brad Pitt	245.000000
Meryl Streep	181.454545

```
b = pd.DataFrame(Combined.groupby('actor_1_name')['num_user_for_reviews'].mean())
b.sort_values(by='num_user_for_reviews',ascending=False)
```

actor_1_name	num_user_for_reviews
Leonardo DiCaprio	922.550000
Brad Pitt	742.352941
Meryl Streep	297.181818

The mean of critic reviews and audience reviews from overall data:

```
e = pd.DataFrame(data.groupby('actor_1_name')['num_critic_for_reviews'].mean())
e.sort_values(by='num_critic_for_reviews',ascending=False).head(10)
```

actor_1_name	num_critic_for_reviews
Albert Finney	750.0
Phaldut Sharma	738.0
Peter Capaldi	654.0
Craig Stark	596.0
Bérénice Bejo	576.0
Suraj Sharma	552.0
Ellar Coltrane	548.0
Mike Howard	546.0
Lou Taylor Pucci	543.0
Joel Courtney	539.0

```
f = pd.DataFrame(data.groupby('actor_1_name')['num_user_for_reviews'].mean())
f.sort_values(by='num_user_for_reviews',ascending=False).head(10)
```

actor_1_name	num_user_for_reviews
Heather Donahue	3400.0
Christo Jivkov	2814.0
Steve Bastoni	2789.0
Phaldut Sharma	1885.0
Orlando Bloom	1842.0
Keir Dullea	1736.0
Chen Chang	1641.0
Nick Stahl	1562.0
Albert Finney	1498.0
Kevin Rankin	1445.0

The sum of users voted in each decade:

Decade= year- remainder(decade/10)

Eg. Year= 2013

So, Remainder(2013/10) = 3

So, Decade = 2013-3 = 2010

```
data['decade']=data['title_year']-(data['title_year']%10)
```

```
df_by_decade = pd.DataFrame(data.groupby('decade')['num_voted_users'].sum())
df_by_decade.sort_values(by='num_voted_users',ascending=False)
```

decade	num_voted_users
2000.0	165976608
2010.0	116240252
1990.0	69635863
1980.0	19344369
1970.0	8269828
1960.0	2983442
1930.0	804839
1950.0	678336
1940.0	230838
1920.0	116392