MINOR PROJECT

Done By: NOTAM KEDARI

kedarinotam3@gmail.com

TMDb MOVIES

The main objective of this project is to go through the dataset and the general data analysis process using numpy, pandas and matplotlib. This data set contains information about 10,000 movies collected from The Movie Database (TMDb), including user ratings and revenue.

In [18]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [19]:
```
movies = pd.read_csv("C:\\Users\\NOTAM KEDARI\\Desktop\\tmdb_movies.csv")
```

In [20]:
```
movies.head()
```

Out[20]:

	id	imdb_id	popularity	budget	revenue	original_title	cast	homepage	director
0	135397	tt0369610	32.985763	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	http://www.jurassicworld.com/	Colin Trevorrow
1	76341	tt1392190	28.419936	150000000	378436354	Mad Max: Fury Road	Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...	http://www.madmaxmovie.com/	George Miller
2	262500	tt2908446	13.112507	110000000	295238201	Insurgent	Shailene Woodley|Theo James|Kate Winslet|Ansel...	http://www.thedivergentseries.movie/#insurgent	Robert Schwentke
3	140607	tt2488496	11.173104	200000000	2068178225	Star Wars: The Force Awakens	Harrison Ford|Mark Hamill|Carrie Fisher|Adam D...	http://www.starwars.com/films/star-wars-episod...	J.J. Abrams
4	168259	tt2820852	9.335014	190000000	1506249360	Furious 7	Vin Diesel|Paul Walker|Jason Statham|Michelle ...	http://www.furious7.com/	James Wan

5 rows × 21 columns

In [21]:
```
movies.shape
```

Out[21]:
```
(10866, 21)
```

In [22]:
```
movies.columns.tolist()
```

Out[22]:
```
['id',
 'imdb_id',
 'popularity',
 'budget',
 'revenue',
 'original_title',
 'cast',
 'homepage',
 'director',
 'tagline',
 'keywords',
 'overview',
 'runtime',
 'genres',
 'production_companies',
 'release_date',
 'vote_count',
 'vote_average',
 'release_year',
 'budget_adj',
 'revenue_adj']
```

In [23]:

```python
movies.describe()
```

Out[23]:

	id	popularity	budget	revenue	runtime	vote_count	vote_average	release_year	budget_adj	rever
count	10866.000000	10866.000000	1.086600e+04	1.086600e+04	10866.000000	10866.000000	10866.000000	10866.000000	1.086600e+04	1.0866
mean	66064.177434	0.646441	1.462570e+07	3.982332e+07	102.070863	217.389748	5.974922	2001.322658	1.755104e+07	5.1364
std	92130.136561	1.000185	3.091321e+07	1.170035e+08	31.381405	575.619058	0.935142	12.812941	3.430616e+07	1.4463
min	5.000000	0.000065	0.000000e+00	0.000000e+00	0.000000	10.000000	1.500000	1960.000000	0.000000e+00	0.0000
25%	10596.250000	0.207583	0.000000e+00	0.000000e+00	90.000000	17.000000	5.400000	1995.000000	0.000000e+00	0.0000
50%	20669.000000	0.383856	0.000000e+00	0.000000e+00	99.000000	38.000000	6.000000	2006.000000	0.000000e+00	0.0000
75%	75610.000000	0.713817	1.500000e+07	2.400000e+07	111.000000	145.750000	6.600000	2011.000000	2.085325e+07	3.3697
max	417859.000000	32.985763	4.250000e+08	2.781506e+09	900.000000	9767.000000	9.200000	2015.000000	4.250000e+08	2.8271

In [24]:

```python
movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10866 entries, 0 to 10865
Data columns (total 21 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   id                    10866 non-null  int64  
 1   imdb_id               10856 non-null  object 
 2   popularity            10866 non-null  float64
 3   budget                10866 non-null  int64  
 4   revenue               10866 non-null  int64  
 5   original_title        10866 non-null  object 
 6   cast                  10790 non-null  object 
 7   homepage              2936 non-null   object 
 8   director              10822 non-null  object 
 9   tagline               8042 non-null   object 
 10  keywords              9373 non-null   object 
 11  overview              10862 non-null  object 
 12  runtime               10866 non-null  int64  
 13  genres                10843 non-null  object 
 14  production_companies  9836 non-null   object 
 15  release_date          10866 non-null  object 
 16  vote_count            10866 non-null  int64  
 17  vote_average          10866 non-null  float64
 18  release_year          10866 non-null  int64  
 19  budget_adj            10866 non-null  float64
 20  revenue_adj           10866 non-null  float64
dtypes: float64(4), int64(6), object(11)
memory usage: 1.7+ MB
```

movies.isnull().sum()

Which are the movies with the third lowest and third highest budget?

In [25]:

```python
# print("The third lowest budget is ",movies.sort_values(['budget']).iloc[2]["budget"])
print("The movie with the third lowest budget is :- ", movies.sort_values(['budget']).iloc[2]['original_title'],'
print("-"*120,"\n")
print("The third highest budget is ",movies.sort_values(['budget']).iloc[-3]["budget"])
print("The movie with the third highest budget is :- ",movies.sort_values(['budget']).iloc[-3]['original_title'],
```

```
The movie with the third lowest budget is :-  Lego Batman: The Movie - DC Super Heroes Unite .
------------------------------------------------------------------------------------------------------------------
-------

The third highest budget is  300000000
The movie with the third highest budget is :-  Pirates of the Caribbean: At World's End .
```

What is the average number of words in movie titles between the year 2000-2005?

In [26]:

```
movies_req = movies[movies['release_year'].isin([2000, 2001, 2002, 2003, 2004, 2005])]

words = 0

# Going through each row which has "release_year" between 2000-2005.
for i in range(movies_req.shape[0]):

    # Splitting based on the Empty Space
    list_of_words = movies_req['original_title'].values[i].split(' ')

    words = words + len(list_of_words)

# Computing the Average
avg = words/movies_req.shape[0]

# Rounding off the Number
avg = round(avg)
```

In [27]:

```
print("The average number of words in movie titles between the year 2000-2005 are ", avg,".")
```

```
The average number of words in movie titles between the year 2000-2005 are  3 .
```

Which are the movies with most and least earned revenue?

In [28]:

```
least = movies.sort_values(['revenue']).iloc[0]["revenue"]
print("The least earned revenue value is",least)

print("The movies with the least earned revenue are :-")
for i in range(movies.shape[0]//1000):
    if movies.sort_values(['revenue']).iloc[i]['revenue'] == least:
        print(movies.sort_values(['revenue']).iloc[i]['original_title'])

print("-"*120,"\n")

most = movies.sort_values(['revenue']).iloc[-1]["revenue"]
print("The most earned revenue value is", most)

print("The movie with the most earned revenue is : ", movies.sort_values(['revenue']).iloc[-1]['original_title'])
```

```
The least earned revenue value is 0
The movies with the least earned revenue are :-
Manos: The Hands of Fate
A Turtleâ€™s Tale 2: Sammyâ€™s Escape From Paradise
Truth or Dare
Laurence Anyways
Io e te
Much Ado About Nothing
London 2012 Olympic Opening Ceremony: Isles of Wonder
Radio Rebel
So Undercover
Maximum Conviction
------------------------------------------------------------------------------------------------------------------
-------

The most earned revenue value is 2781505847
The movie with the most earned revenue is :  Avatar
```

What is the average runtime of movies in the year 2006?

In [29]:
```
run = movies.loc[ (movies["release_year"]== 2006), "runtime" ].tolist()
```

In [30]:
```
avg= np.mean(run)
```

In [31]:
```
print("The average runtime of movies in the year 2006 is :",avg)
```

```
The average runtime of movies in the year 2006 is : 101.68382352941177
```

END