Introduction

The dataset I chose is The Movie Database (TMDB). It contains over 10,000 movies. The objectove is to explore in order to find out useful insights from the dataset.

Questions

1. Which genres are most popular from year to year?
2. What kind of properties are associated with movies that high revenues?
3. How many movies are produced annually?
4. Movies with the highest budget
5. Top 10 movies with the highest revenue.
6. 10 most popular movies
7. 10 most profitable movies
8. 10 Least profitable movies.
9. Genres with the highest release.

```
#import the modules
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Data Wrangling

The dataset will be loaded and converted to a dataframe for investigation and exploration.

```
#load the movie dataset
df_movie = pd.read_csv('movies.csv')
df_movie.head()
```

	id	imdb_id	popularity	budget	revenue	original_title	
0	135397	tt0369610	32.985763	150000000	1513528810	Jurassic World	Chris Pratt| I Howard Kha
1	76341	tt1392190	28.419936	150000000	378436354	Mad Max: Fury Road	Hardy|Ch Theron K Byrne
2	262500	tt2908446	13.112507	110000000	295238201	Insurgent	Sh Woodley Jame Winslet|A
3	140607	tt2488496	11.173104	200000000	2068178225	Star Wars: The Force Awakens	Ha Ford Hamill| Fisher|Ada
4	168259	tt2820852	9.335014	190000000	1506249360	Furious 7	Vin Diese Walker| Statham|Mi

```
df_movie.tail()
```

	id	imdb_id	popularity	budget	revenue	original_title	
10861	21	tt0060371	0.080598	0	0	The Endless Summer	Mi Hynson|R August|Lord Hc
10862	20379	tt0060472	0.065543	0	0	Grand Prix	James Garne Marie Saint| Montand|Tc
10863	39768	tt0060161	0.065141	0	0	Beregis Avtomobilya	Innok Smoktunovskiy

Determine how many rows and columns the movie data frame have.

```
df_movie.shape
```

```
(10866, 21)
```

Determine the datatypes and not null objects of each column in the dataframe

10865	22293	tt0060666	0.035919	19000	0	Hands of Fate	Nevmanl

```
df_movie.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
```

```
genre_split =df_movie['genres'].str.split('|', n=3, expand=True)
genre_split
```

	0	1	2	3
0	Action	Adventure	Science Fiction	Thriller
1	Action	Adventure	Science Fiction	Thriller
2	Adventure	Science Fiction	Thriller	None
3	Action	Adventure	Science Fiction	Fantasy
4	Action	Crime	Thriller	None
...	...	...	...	...
10861	Documentary	None	None	None
10862	Action	Adventure	Drama	None
10863	Mystery	Comedy	None	None
10864	Action	Comedy	None	None
10865	Horror	None	None	None

10866 rows × 4 columns

Data Cleaning

1. Check for duplicates

```
df_movie.duplicated().sum()
```

```
1
```

```
#drop the duplicates
df_movie.drop_duplicates(inplace=True)
```

```
df_movie.shape
```

```
(10865, 21)
```

```
df_movie.duplicated().sum()
```

```
0
```

```
##Remove the columns that will not be used in this investigation

df_movie.drop(['budget_adj','revenue_adj','overview','imdb_id','homepage','tagline'],axis =1,
```

```
df_movie.shape
```

```
(10865, 15)
```

Check for null values

```
df_movie.isna().sum()
```

```
id                        0
popularity                0
budget                    0
revenue                   0
original_title            0
cast                     76
director                 44
keywords               1493
runtime                   0
genres                   23
production_companies   1030
release_date              0
vote_count                0
vote_average              0
release_year              0
dtype: int64
```

Replace the null values with zero

```
# Checking for an zero values in the budget and revenue columns
df_movie[(df_movie['budget']==0)].shape[0]
```

```
5696
```

```
df_movie[(df_movie['revenue']==0)].shape[0]
```

```
6016
```

replace zero with nan

```
df_movie.replace(0,np.nan,inplace=True)
```

```
#Drop rows that have zero values
```

```
df_movie.dropna(inplace=True)
```

```
df_movie.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3677 entries, 0 to 10848
Data columns (total 15 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   id                    3677 non-null   int64  
 1   popularity            3677 non-null   float64
 2   budget                3677 non-null   float64
 3   revenue               3677 non-null   float64
 4   original_title        3677 non-null   object 
 5   cast                  3677 non-null   object 
 6   director              3677 non-null   object 
 7   keywords              3677 non-null   object 
 8   runtime               3677 non-null   float64
 9   genres                3677 non-null   object 
 10  production_companies  3677 non-null   object 
 11  release_date          3677 non-null   object 
 12  vote_count            3677 non-null   int64  
 13  vote_average          3677 non-null   float64
 14  release_year          3677 non-null   int64  
dtypes: float64(5), int64(3), object(7)
memory usage: 459.6+ KB
```

```
## **Exploratory Data Analysis**
```

Exploratory Data Analysis

Descriptive Statistics of the numerical columns of the dataframe

```
df_movie.describe()
```

	id	popularity	budget	revenue	runtime	vote_count
count	3677.000000	3677.000000	3.677000e+03	3.677000e+03	3677.000000	3677.000000
mean	39224.526244	1.226051	3.811465e+07	1.114405e+08	109.561327	547.702203
std	67249.633137	1.498897	4.267577e+07	1.793625e+08	19.855075	894.954704
min	5.000000	0.010335	1.000000e+00	2.000000e+00	26.000000	10.000000
25%	5470.000000	0.481276	1.000000e+07	1.489942e+07	96.000000	78.000000
50%	11017.000000	0.830597	2.500000e+07	4.806344e+07	106.000000	219.000000
75%	37958.000000	1.411147	5.000000e+07	1.298324e+08	120.000000	596.000000
max	417859.000000	32.985763	4.250000e+08	2.781506e+09	338.000000	9767.000000

```
df_movie.hist(figsize=(10, 8))
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7fa58f0c7ed0>,
```

Question 1: Which genres are most popular from year to year?

```
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fa58ee86590>,
```

Create a function to extract the unique genres in the genre column as a list

```
def separate_count(column):
    genre_split = pd.Series(df_movie[column].str.cat(sep = '|').split('|'))

    genre = list(genre_split.unique())
    #genre2 = genre.replace(' ',',')
    return  genre
```

```
genre = separate_count('genres')

print(genre)
```

```
['Action', 'Adventure', 'Science Fiction', 'Thriller', 'Fantasy', 'Crime', 'Western', '
```

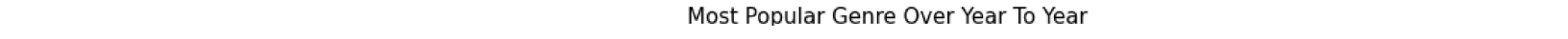
```
genre_list = list(map(str,(df_movie['genres'])))


#make the numpy array of year and popularity which contain all the rows of release_year and p
year = np.array(df_movie['release_year'])
film_popularity = np.array(df_movie['popularity'])

#make a null dataframe which indexs are genres and columns are years.
df_popular = pd.DataFrame(index = genre, columns = range(1960, 2016))
#The range year is between the min year and the max year according to the descriptive statist
#change all the values of the dataframe from NAN to zero.
df_popular = df_popular.fillna(value = 0.0)

x = 0
for y in genre_list:
    split_genre = list(map(str,y.split('|')))
    df_popular.loc[split_genre, year[x]] = df_popular.loc[split_genre, year[x]] + film_popula
    x+=1


# function to calculate the standard deviation for the accurate results.
def calculate(x):
    return (x-x.mean())/x.std(ddof=0)

popular_gen = calculate(df_popular)
popular_gen.head()
```

	1960	1961	1962	1963	1964	1965	1966
Action	1.521076	-0.195604	1.850915	1.485508	0.973797	1.612008	1.769465
Adventure	0.666668	1.385628	2.640234	1.866965	0.973797	1.612008	1.084220
Science Fiction	-0.740233	-0.792705	-0.645666	-0.780206	-0.954579	-0.813665	-0.006546
Thriller	1.221445	-0.792705	1.495611	2.634687	1.379459	1.016124	0.704383
Fantasy	-0.740233	-0.792705	-0.645666	-0.780206	0.260867	-0.813665	-0.760334

5 rows × 56 columns

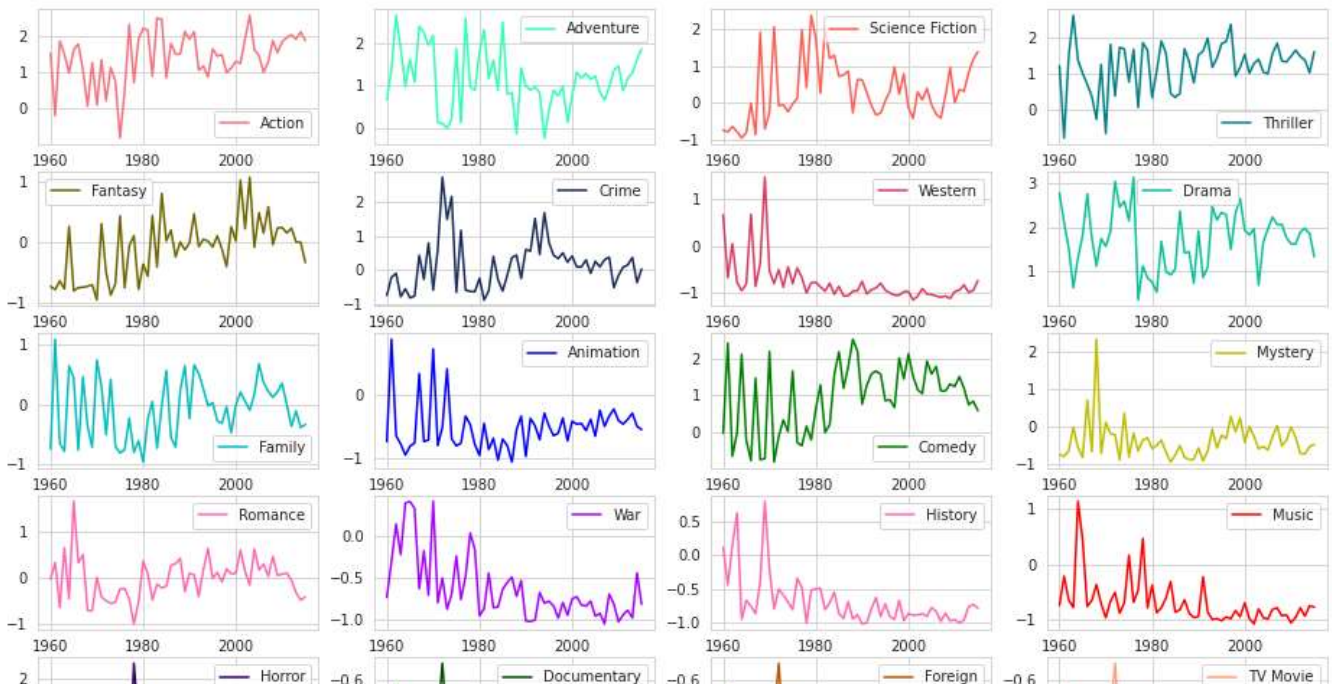
```
#plot the barh plot of the standardised data.
popular_gen.iloc[0:,53:].plot(kind='bar',figsize = (15,6),fontsize=13)


#setup the title and labels of the plot.
plt.title("Most Popular Genre Over Year To Year",fontsize=15)
plt.xlabel("Popularity",fontsize=15)
plt.ylabel("Genres",fontsize = 15)
```

```
Text(0, 0.5, 'Genres')
```




```
sns.set_style("whitegrid")
#make a subplot of size 3,3.
fig, ax = plt.subplots(5,4,figsize = (16,10))

#set the title of the subplot.
fig.suptitle('Genre Popularity Over Year To Year',fontsize = 16)

#plot the 'Drama' genre plot see the popularity difference over year to year.
popular_gen.loc['Action'].plot(label = "Action",color = '#f67280',ax = ax[0][0],legend=True)

#plot the 'Action' genre plot see the popularity difference over year to year.
popular_gen.loc['Adventure'].plot(label = "Adventure",color='#33FFB5',ax = ax[0][1],legend=Tr

#plot the 'Comedy' genre plot see the popularity difference over year to year.
popular_gen.loc['Science Fiction'].plot(label = "Science Fiction",color='#fe5f55',ax = ax[0][

#accordig to the previous plot the 'Thriller','Science Fiction', 'Romance', 'Music','Adventur
popular_gen.loc['Thriller'].plot(label = "Thriller",color='#00818a',ax = ax[0][3],legend=True
popular_gen.loc['Fantasy'].plot(label = "Fantasy",color='#6f6600',ax = ax[1][0],legend=True)
popular_gen.loc['Crime'].plot(label = "Crime",color='#1a2c5b',ax = ax[1][1],legend=True)
popular_gen.loc['Western'].plot(label = "Western",color='#db3b61',ax = ax[1][2],legend=True)
popular_gen.loc['Drama'].plot(label = "Drama",color='#08c299',ax = ax[1][3],legend=True)
popular_gen.loc['Family'].plot(label = "Family",color='c',ax = ax[2][0],legend=True)
popular_gen.loc['Animation'].plot(label = "Animation",color='b',ax = ax[2][1],legend=True)
popular_gen.loc['Comedy'].plot(label = "Comedy",color='g',ax = ax[2][2],legend=True)
popular_gen.loc['Mystery'].plot(label = "Mystery",color='y',ax = ax[2][3],legend=True)
popular_gen.loc['Romance'].plot(label = "Romance",color='#FF69B4',ax = ax[3][0],legend=True)
popular_gen.loc['War'].plot(label = "War",color='#aa00ff',ax = ax[3][1],legend=True)
popular_gen.loc['History'].plot(label = "History",color='#FF69B4',ax = ax[3][2],legend=True)
popular_gen.loc['Music'].plot(label = "Music",color='#ff0000',ax = ax[3][3],legend=True)
popular_gen.loc['Horror'].plot(label = "Horror",color='#330066',ax = ax[4][0],legend=True)
popular_gen.loc['Documentary'].plot(label = "Documentary",color='#004d00',ax = ax[4][1],leger
popular_gen.loc['Foreign'].plot(label = "Foreign",color='#b35900',ax = ax[4][2],legend=True)
popular_gen.loc['TV Movie'].plot(label = "TV Movie",color='#ff9f80',ax = ax[4][3],legend=True
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa58eb11410>
```



From the above plots; The bar plot and the line plot. The bar plot shows the popularity of a genre from 2013-2015. The three popular genres are Action, Adventure and Thriller.

The line plots is a more comprehensive data showing the poplularity of a genre from year to year. There are twenty genres in this data set. The year range is from 1960-2015. It shows the popularity from Action genre to TV Movie genre.

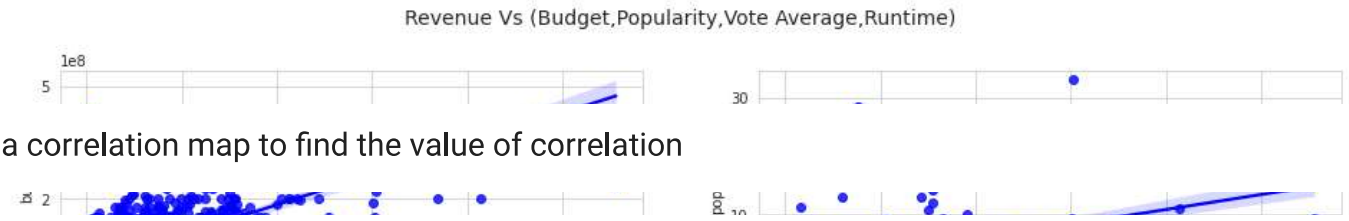
One would notice that there is a spike in the 60s for almost all the genres. One would also notice that the top three popular genres here are Action, Adventure and thriller. Science Fiction follows closely behind in the fourth position

Question 2: What kinds of properties are associated with movies that have high revenues?

```
revenue = pd.DataFrame(df_movie['revenue'].sort_values(ascending=False))
movie_properties = ['id','popularity','budget','original_title','cast','director','runtime','
for i in movie_properties:
    revenue[i] = df_movie[i]
revenue.head()
```

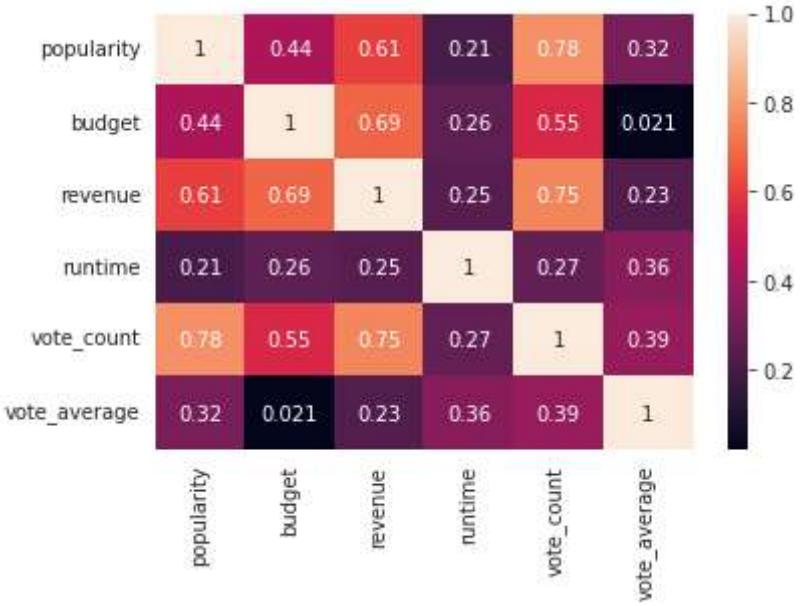
	revenue	id	popularity	budget	original_title	cast	d
1386	2.781506e+09	19995	9.432768	237000000.0	Avatar	Sam Worthington|Zoe Saldana|Sigourney Weaver|S...	(
3	2.068178e+09	140607	11.173104	200000000.0	Star Wars: The Force Awakens	Harrison Ford|Mark Hamill|Carrie Fisher|Adam D...	
5231	1.845034e+09	597	4.355219	200000000.0	Titanic	Kate Winslet|Leonardo DiCaprio|Frances Fisher|...	(
4361	1.519558e+09	24428	7.637767	220000000.0	The Avengers	Robert Downey Jr.|Chris Evans|Mark Ruffalo|Chr...	
						Chris Pratt|Bryce Dallas	

```
fig, axes = plt.subplots(2,2,figsize = (16,6))
fig.suptitle("Revenue Vs (Budget,Popularity,Vote Average,Runtime)",fontsize=14)

sns.regplot(x=df_movie['revenue'], y=df_movie['budget'],color='b',ax=axes[0][0])
sns.regplot(x=df_movie['revenue'], y=df_movie['popularity'],color='b',ax=axes[0][1])
sns.regplot(x=df_movie['revenue'], y=df_movie['vote_average'],color='b',ax=axes[1][0])
sns.regplot(x=df_movie['revenue'], y=df_movie['runtime'],color='b',ax=axes[1][1])

sns.set_style("dark")
```



Plot a correlation map to find the value of correlation


```
df = pd.DataFrame(df_movie,columns=['popularity','budget','revenue','runtime','vote_count','\
corrMatrix = df.corr()
sns.heatmap(corrMatrix, annot=True)
plt.show()
```



Plot 1: In the regression plot, you would see that the budget is correlated to the revenue. The higher the budget, the higher the revenue.

- Correlation =0.69

Plot2: Popularity is slightly correlated to the revenue.

- Correlation =0.61

Plot 3: Vote_average is very loosely correlated to the revenue

- Correlation Score=0.23

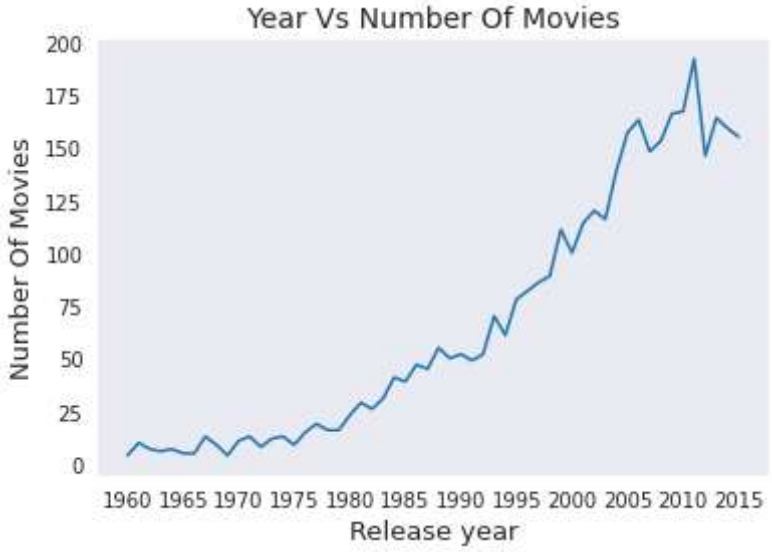
Plot 4: Runtime is also very loosely correlated to the revenue with a correlation score of 0.23

Question3: Number of movies created in a year

```
# make group for each year and count the number of movies in each year
movie_number=df_movie.groupby('release_year').count()['id']
print(movie_number.tail(10))
```

```
#make group of the data according to their release year and count the total number of movies
df_movie.groupby('release_year').count()['id'].plot(xticks = np.arange(1960,2016,5))

#set the figure size and labels
sns.set(rc={'figure.figsize':(10,5)})
plt.title("Year Vs Number Of Movies",fontsize = 14)
plt.xlabel('Release year',fontsize = 13)
plt.ylabel('Number Of Movies',fontsize = 13)
#set the style sheet
sns.set_style("dark")
```

```
release_year
2006    163
2007    148
2008    153
2009    166
2010    167
2011    192
2012    146
2013    164
2014    159
2015    155
Name: id, dtype: int64
```



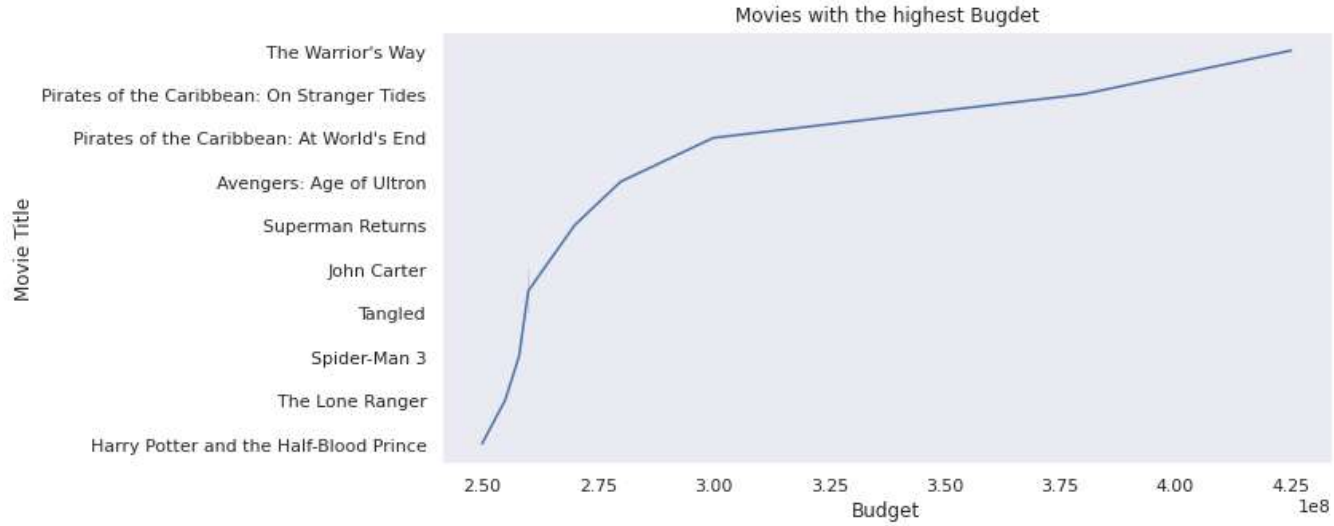
The highest number of movies was released in 2011 with 192 movies in total. The second highest was in 2009 with 166 movies.

Question 4: Movies with the highest budget

```
budget=  pd.DataFrame(df_movie['budget'].sort_values(ascending = False))
budget['original_title'] =df_movie['original_title']
data = list(map(str,(budget['original_title'])))
x = list(budget['budget'][:10])
y =list(data[:10])
```

```
sns.lineplot( x=x, y=y)
plt.title("Movies with the highest Bugdet")
plt.xlabel("Budget")
plt.ylabel("Movie Title")
```

```
Text(0, 0.5, 'Movie Title')
```



The the top three movies with the highest budget, the warrior way, pirate of the carribean: Stranger tides and The world's end.

Question5: Movies with the highest revenue

```
revenue =  pd.DataFrame(df_movie['revenue'].sort_values(ascending = False))
revenue['original_title'] =df_movie['original_title']
data = list(map(str,(revenue['original_title'])))
x = list(revenue['revenue'][:10])
y =list(data[:10])
sns.lineplot( x=x, y=y)
plt.title("Movies with the highest revenue")
plt.xlabel("Revenue")
plt.ylabel("Movie Title")
```

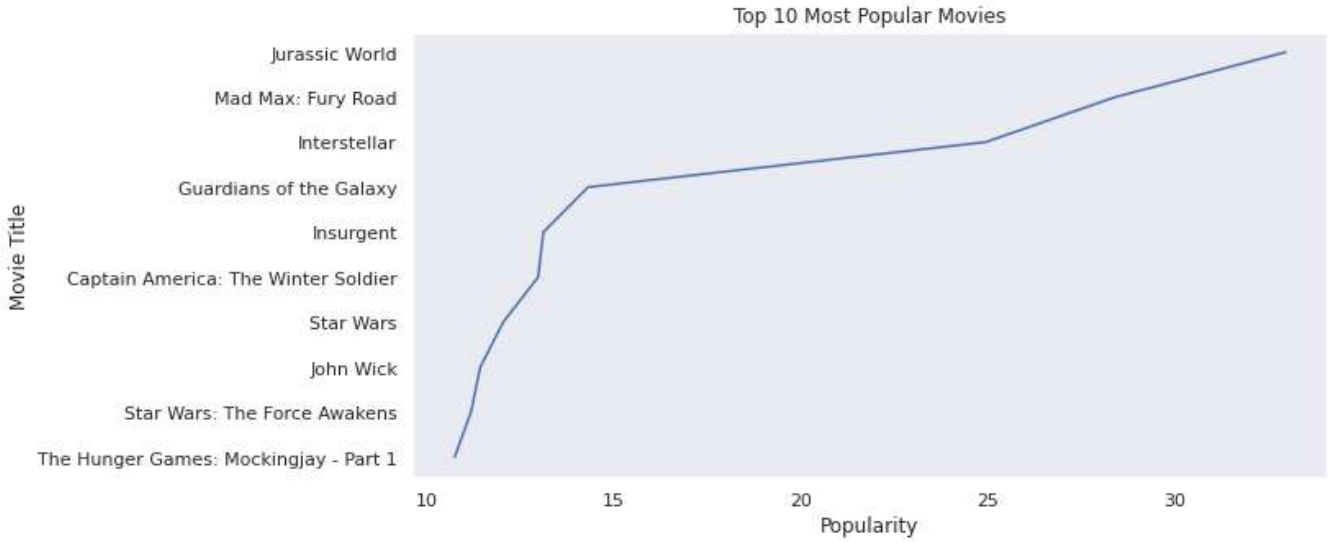
```
Text(0, 0.5, 'Movie Title')
```

The highest reveneu is Avartar, Star wars and Titanic

Question 6: Top 10 most popular movies

```
popular =  pd.DataFrame(df_movie['popularity'].sort_values(ascending = False))
popular['original_title'] =df_movie['original_title']
data = list(map(str,(popular['original_title'])))
x = list(popular['popularity'][:10])

y =list(data[:10])
sns.lineplot( x=x, y=y)
plt.title("Top 10 Most Popular Movies")
plt.xlabel("Popularity")
plt.ylabel("Movie Title")
```

```
Text(0, 0.5, 'Movie Title')
```

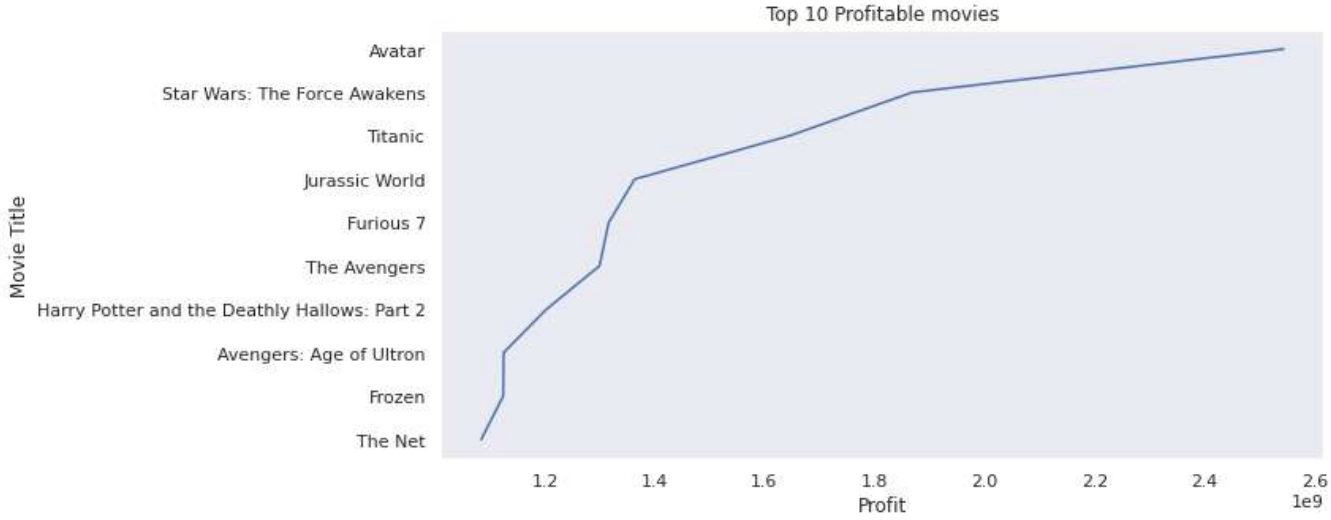


The most popular films are Jurassic world, Mad Max fury and Interstellar.

Qustion 7: Profitable movies

```
#To get the proft, fine the difference between the revenue and the budget
df_movie['Profit'] = df_movie['revenue'] - df_movie['budget']
profit = pd.DataFrame(df_movie['Profit'].sort_values(ascending = False))
profit['original_title'] = df_movie['original_title']
data = list(map(str,(profit['original_title'])))
title= list(data[:10])
highgross = list(profit['Profit'][:10])
sns.lineplot( x=highgross, y=title)
plt.title("Top 10 Profitable movies")
plt.xlabel("Profit")
plt.ylabel("Movie Title")
```

```
Text(0, 0.5, 'Movie Title')
```



Avartar is the highest profitable movie followed by starwarw, titanic and Jurassic world.

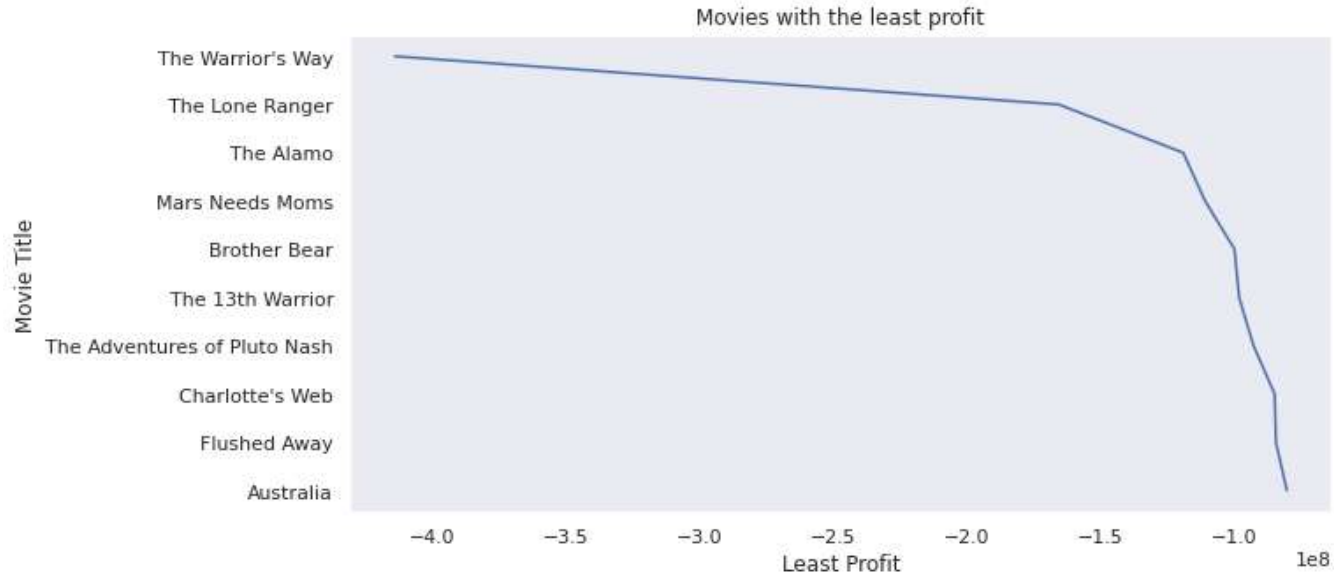
Double-click (or enter) to edit

Question 8.Top 10 lowest profit movies

```
least_profit = pd.DataFrame(df_movie['Profit'].sort_values(ascending = True))
least_profit['original_title'] = df_movie['original_title']
data = list(map(str,(least_profit['original_title'])))
high_Gross = list(data[:10])
title = list(least_profit['Profit'][:10])
```

```
ax = sns.lineplot( x=title, y=high_Gross)
plt.title("Movies with the least profit")
plt.xlabel("Least Profit")
plt.ylabel("Movie Title")
```

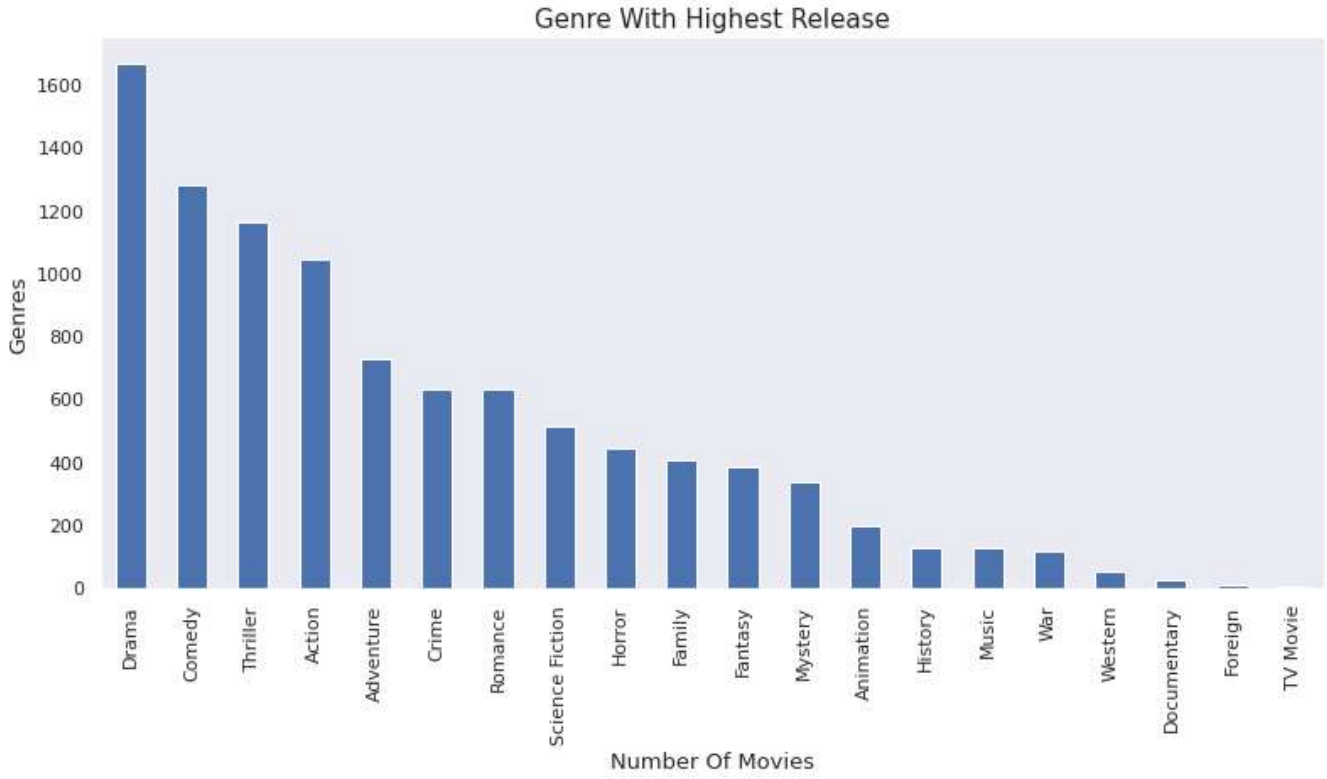
```
Text(0, 0.5, 'Movie Title')
```



The lowest grossing movie is Warrior way followed by lone ranger, Alamo and Mars need Moms.

Question 9. Genre with the highest release

```
def calculate(x):
    #concatenate all the rows of the genrs.
    genre_plot = df_movie[x].str.cat(sep = '|')
    data = pd.Series(genre_plot.split('|'))

    info = data.value_counts(ascending=False)
    return info

#call the function for counting the movies of each genre.
total_genre_movies = calculate('genres')

total_genre_movies.plot(kind= 'bar',figsize = (13,6))

#setup the title and the labels of the plot.
plt.title("Genre With Highest Release",fontsize=15)
plt.xlabel('Number Of Movies',fontsize=13)
plt.ylabel("Genres",fontsize= 13)
sns.set_style("dark")
```



Drama has the highest release of genre followed by comedy.

Conclusions:

1. Action is the most popular movie produced annually.
2. Budget was hightly correlated to the revenue with a correlation score of 0.69 followed by popularity with a correlation score of 0.61
3. Year 2011 had the highest amount of movies produced with 192 movies released.
4. The movie with the highest budget is The Warrior's way followed by Pirates of the Carribean: On Stranger Tides.
5. The movie that has the highest revenue is Avatar followed by Star Wars and Titanic.
6. The most popular movie is Jurassic World followed by Mad Max
7. The most profitable movie is Avatar.
8. The least profitable movie is The Warrior Way. Even though it had the highest budget, it still did not make it profitable.
9. Genre with the highest release is Drama followed by the Comedy genre

Limitations

- One would see from the datasets. In the beginning of the investigation, there were about 10,000 datasets, but to get insights from the data. we had to delete the null data which would have affected the results leaving a little less than 4000 movie datasets.
- The genre had a seperator character("|") which had to be split during the data cleaning process. To investigate the data in this category took longer time than if it did not have this character.

✓ 0s completed at 12:17 PM