MARVEL VS DC

Importing Libraries

In [240…

```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
plt.style.use('dark_background')
import warnings
warnings.filterwarnings(action = 'ignore')
```

Importing Dataset

```
In [198… data = pd.read_csv('C:/Users/WELCOME
 DURGESH/Downloads/SEM_6/Subjects/Machine_Learni
```

Functions head() function is used to print the first 5 rows of

the dataset.

In [199…

```
data.head()
```

Out[199…

	Sno	Original_Title	Company	Rate	Metascore	Minutes	Release	Budget	Opening_Weekend
0	1	Iron Man	Marvel	7.9	79	126	2008	140000000	9861
		First Avenger							

Using **shape** we can observe the dimensions of the data.

```
data.shape
```

```
(40, 11)
```

	Sno	Original_Title	Company	Rate	Metascore	Minutes	Release	Budget	Opening_Weekend
1	2	The Incredible Hulk	Marvel	6.7	61	112	2008	150000000	5541
2	3	Iron Man 2	Marvel	7.0	57	124	2010	200000000	12812
3	4	Thor	Marvel	7.0	57	115	2011	150000000	6572
4	5	Captain America: The	Marvel	6.9	66	124	2011	140000000	6505

In [200…

Out[200…

info() method shows some of the characteristics of the data such as Column Name, No. of nonnull values of our columns, Dtype of the data, and Memory Usage.

In [201…

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 40 entries, 0 to 39
Data columns (total 11 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
0    Sno                 40 non-null     int64
1    Original_Title      40 non-null     object
2    Company             40 non-null     object
3    Rate                40 non-null     float64
4    Metascore           40 non-null     int64
5    Minutes             40 non-null     int64
6    Release             40 non-null     int64
7    Budget              40 non-null     int64
8    Opening_Weekend_USA 40 non-null     int64
9    Gross_USA           40 non-null     int64    10   Gross_Worldwide      40 non-
     null     int64
dtypes: float64(1), int64(8), object(2)
memory usage: 3.6+ KB
```

columns attribute of pandas dataframe shows the column names in the dataset.

In [202…

```
data.columns
```

Out[202…

```
Index(['Sno', 'Original_Title', 'Company', 'Rate', 'Metascore', 'Minutes',
       'Release', 'Budget', 'Opening_Weekend_USA', 'Gross_USA',
       'Gross_Worldwide'],
      dtype='object')
```

isnull() Checking if the dataset has missing values.

In [203…

```
data.isnull().sum()
```

Out[203…

```
Sno                    0
Original_Title         0
Company                0
Rate                   0
Metascore              0
Minutes                0
Release                0
Budget                 0
Opening_Weekend_USA    0
Gross_USA              0
Gross_Worldwide        0
dtype: int64
```

Checking for the duplicate values.

In [204…

```
dv = data.duplicated()
print(dv.sum())
data[dv]
```

```
0
```

Out[204…

Sno	Original_Title	Company	Rate	Metascore	Minutes	Release	Budget	Opening_Weekend_USA

describe() function is used to find the count, mean, median, mode, standard deviation and quantiles of the dataset.

In [205…

```
data.describe(include='all')
```

Out[205…

	Sno	Original_Title	Company	Rate	Metascore	Minutes	Release	B
count	40.000000	40	40	40.000000	40.000000	40.000000	40.000000	4.00000
unique	NaN	40	2	NaN	NaN	NaN	NaN	
top	NaN	Spider-Man: Homecoming	Marvel	NaN	NaN	NaN	NaN	
freq	NaN	1	23	NaN	NaN	NaN	NaN	
mean	20.500000	NaN	NaN	7.242500	63.425000	134.550000	2013.950000	1.87750
std	11.690452	NaN	NaN	1.090492	13.767087	24.976861	4.343873	6.76346
min	1.000000	NaN	NaN	3.300000	27.000000	81.000000	2004.000000	4.70000
25%	10.750000	NaN	NaN	6.900000	55.750000	120.750000	2011.000000	1.50000
50%	20.500000	NaN	NaN	7.300000	66.500000	131.000000	2015.000000	1.75000
75%	30.250000	NaN	NaN	7.900000	72.250000	143.000000	2017.250000	2.21250
max	40.000000	NaN	NaN	9.000000	88.000000	240.000000	2020.000000	3.56000

unique() function is used to find the different unique values in the given series or dataframe.

In [206…

```
data['Company'].unique()
```

Out[206…

```
array(['Marvel', 'DC'], dtype=object)
```

nunique() function is used to find the total unique values in all of the columns.

In [207…

```
data.nunique()
```

Out[207…

```
Sno               40
Original_Title    40
Company            2
Rate              25
Metascore         29
Minutes           31
Release           16
```

```
Budget                 23
Opening_Weekend_USA    40
Gross_USA              40
Gross_Worldwide        40
dtype: int64
```

value_counts() function is used to find the frequency of each occurrence of different categorical values in the column.

In [208…

```
data['Company'].value_counts()
```

Out[208…

```
Marvel    23
DC        17
Name: Company, dtype: int64
```

Central Tendency mean()

In [209…

```
data['Budget'].mean()
```

Out[209…

```
187750000.0
```

median()

In [210…

```
data['Budget'].median()
```

Out[210…

```
175000000.0
```

mode()

In [211…

```
data['Budget'].mode()
```

Out[211…

```
0    200000000
dtype: int64
```

quantiles()

In [212…

```
data['Budget'].quantile([0.25,0.50,0.75])
```

Out[212…

```
0.25    150000000.0
0.50    175000000.0
0.75    221250000.0
Name: Budget, dtype: float64
```

astype() function is used to convert the data from one type to another.

In [213…

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40 entries, 0 to 39
Data columns (total 11 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
```

```
0   Sno                 40 non-null     int64
1   Original_Title      40 non-null     object
2   Company             40 non-null     object
3   Rate                40 non-null     float64
4   Metascore           40 non-null     int64
5   Minutes             40 non-null     int64
6   Release             40 non-null     int64
7   Budget              40 non-null     int64
8   Opening_Weekend_USA 40 non-null     int64
9   Gross_USA           40 non-null     int64    10  Gross_Worldwide      40 non-

memory usage: 3.6+ KB
```

```
data['Minutes'] = data['Minutes'].astype(float)
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40 entries, 0 to 39
    null     int64   dtypes: float64(1), int64(8), object(2)
```

In [54]:

In [214…

```
Data columns (total 11 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
0   Sno                 40 non-null     int64
1   Original_Title      40 non-null     object
2   Company             40 non-null     object
3   Rate                40 non-null     float64
4   Metascore           40 non-null     int64
5   Minutes             40 non-null     int64
6   Release             40 non-null     int64
7   Budget              40 non-null     int64
8   Opening_Weekend_USA 40 non-null     int64
9   Gross_USA           40 non-null     int64    10  Gross_Worldwide      40 non-
    null     int64
dtypes: float64(1), int64(8), object(2)
memory usage: 3.6+ KB
```

drop() is used to drop a column

In [215…

```
data.drop('Opening_Weekend_USA', axis = 1)
```

Out[215…

	Sno	Original_Title	Company	Rate	Metascore	Minutes	Release	Budget	Gross_USA	Gros
0	1	Iron Man	Marvel	7.9	79	126	2008	140000000	318604126	
1	2	The Incredible Hulk	Marvel	6.7	61	112	2008	150000000	134806913	
2	3	Iron Man 2	Marvel	7.0	57	124	2010	200000000	312433331	
3	4	Thor	Marvel	7.0	57	115	2011	150000000	181030624	

4	5	Captain America: The First Avenger	Marvel	6.9	66	124	2011	140000000	176654505
5	6	The Avengers	Marvel	8.0	69	143	2012	220000000	623357910
6	7	Iron Man Three	Marvel	7.2	62	130	2013	200000000	409013994
7	8	Thor: The Dark World	Marvel	6.9	54	112	2013	170000000	206362140
8	9	Captain America: The Winter Soldier	Marvel	7.7	70	136	2014	170000000	259766572
9	10	Guardians of the Galaxy	Marvel	8.0	76	121	2014	170000000	333176600
10	11	Avengers: Age of Ultron	Marvel	7.3	66	141	2015	250000000	459005868
11	12	Ant-Man	Marvel	7.3	64	117	2015	130000000	180202163
12	13	Captain America: Civil War	Marvel	7.8	75	147	2016	250000000	408084349
13	14	Doctor Strange	Marvel	7.5	72	115	2016	165000000	232641920
14	15	Guardians of the Galaxy Vol. 2	Marvel	7.6	67	136	2017	200000000	389813101
15	16	Spider-Man: Homecoming	Marvel	7.4	73	133	2017	175000000	334201140
16	17	Thor:Ragnarok	Marvel	7.9	74	130	2017	180000000	315058289
17	18	Black Panther	Marvel	7.3	88	134	2018	200000000	700059566
18	19	Avengers: Infinity War	Marvel	8.5	68	149	2018	321000000	678815482

	Sno	**Original_Title**	**Company**	**Rate**	**Metascore**	**Minutes**	**Release**	**Budget**	**Gross_USA**	**Gros**
19	20	Ant-Man and the Wasp	Marvel	7.1	70	118	2018	162000000	216648740	
20	21	Captain Marve	Marvel	6.9	64	123	2019	175000000	426829839	
21	22	Avengers: Endgame	Marvel	8.5	78	181	2019	356000000	858373000	
22	23	Spider-Man: Far from Home	Marvel	7.6	69	129	2019	160000000	390532085	

23	24	Catwoman	DC	3.3	27	104	2004	100000000	40202379
24	25	Batman Begins	DC	8.2	70	140	2005	150000000	206852432
25	26	Superman Returns	DC	6.0	72	154	2006	270000000	200081192
26	27	The Dark Knight	DC	9.0	84	152	2008	185000000	535234033
27	28	Watchmen	DC	7.6	56	162	2009	130000000	107509799
28	29	Jonah Hex	DC	4.7	33	81	2010	47000000	10547117
29	30	Green Lantern	DC	5.5	39	114	2011	200000000	116601172
30	31	The Dark Knight Rises	DC	8.4	78	164	2012	250000000	448139099
31	32	Man of Steel	DC	7.1	55	143	2013	225000000	291045518
32	33	Batman v Superman: Dawn of Justice	DC	6.5	44	151	2016	250000000	330360194
33	34	Suicide Squad	DC	6.0	40	123	2016	175000000	325100054
34	35	Wonder Woman	DC	7.4	76	141	2017	149000000	412563408
35	36	Justice League	DC	6.4	45	120	2017	300000000	229024295
36	37	Aquaman	DC	7.0	55	143	2018	160000000	335061807
37	38	Shazam!	DC	7.1	71	132	2019	100000000	140371656
38	39	Joker	DC	8.7	59	122	2019	55000000	333204580
39	40	Zack Snyder's Justice League	DC	8.8	54	240	2020	330000000	35456000

CHARTS

boxplot() function is used to draw a box plot on the data

In [261…

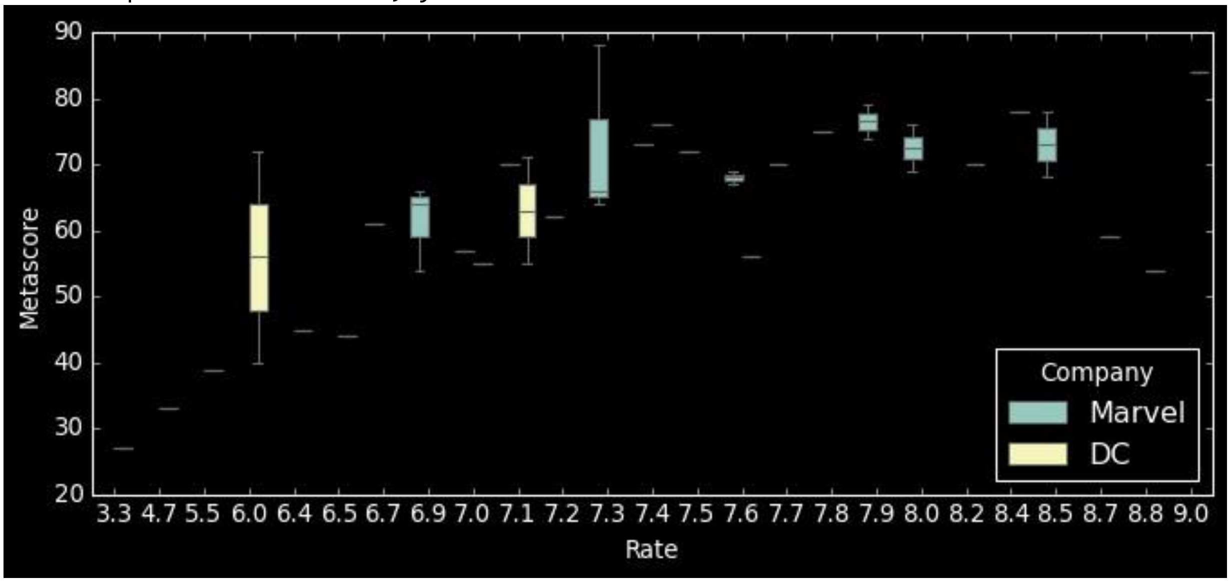
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40 entries, 0 to 39
Data columns (total 11 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
0    Sno                  40 non-null     int64
1    Original_Title       40 non-null     object
2    Company              40 non-null     object
```

```
 3   Rate                40 non-null     float64
 4   Metascore           40 non-null     int64
 5   Minutes             40 non-null     int64
 6   Release             40 non-null     int64
 7   Budget              40 non-null     int64
 8   Opening_Weekend_USA  40 non-null     int64
 9   Gross_USA           40 non-null     int64    10  Gross_Worldwide      40 non-
     null     int64   dtypes: float64(1), int64(8), object(2) memory usage: 3.6+ KB
```

In [278…
```
plt.figure(figsize=(10,4)) sns.boxplot(x='Rate',
  y='Metascore',data = data, hue="Company")
```

Out[278…
```
<AxesSubplot:xlabel='Rate', ylabel='Metascore'>
```



In [217…
```
correlation = data.corr()
```

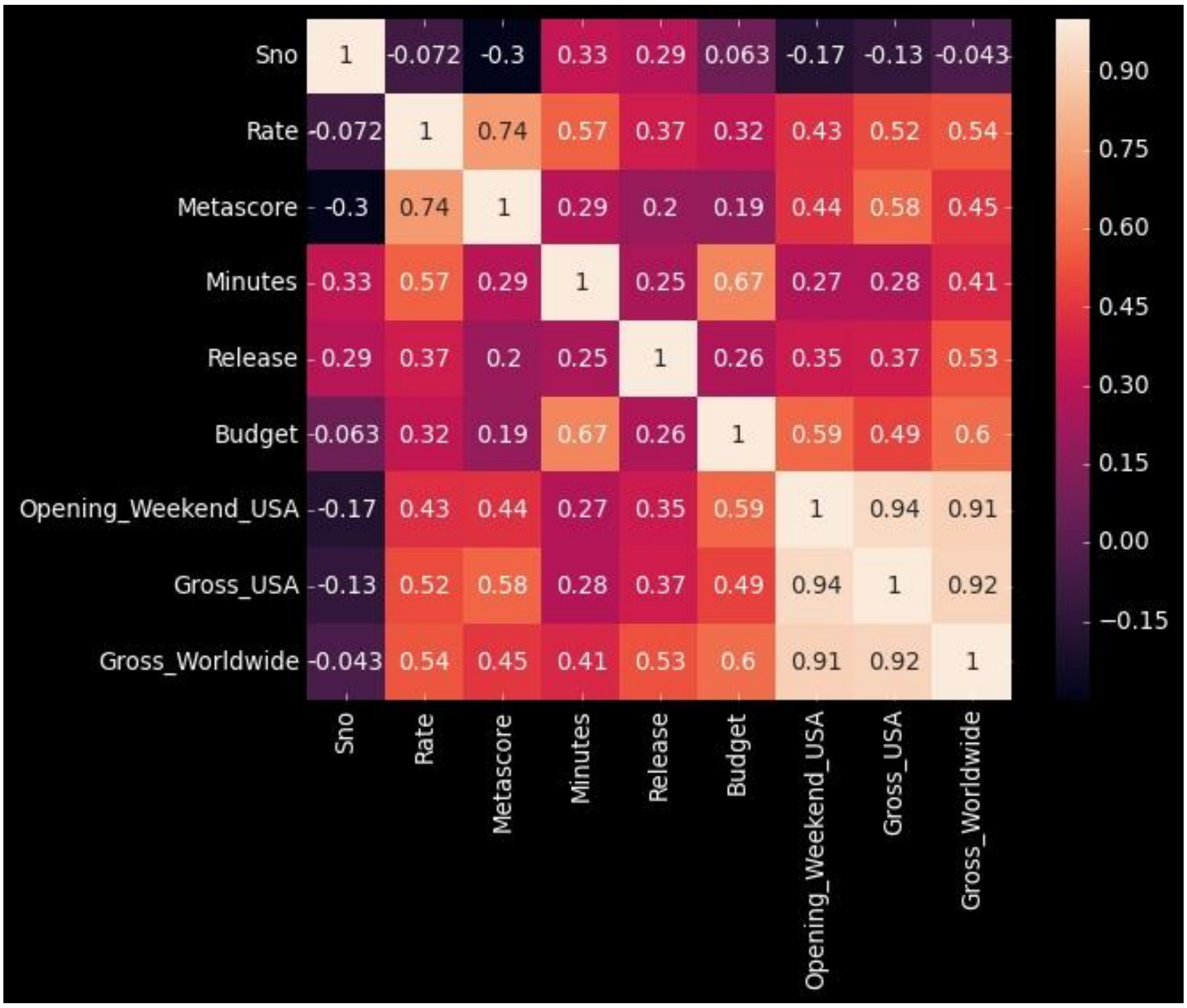
The resulting coefficient value lies between -1 and 1 where:-

- 1: Total +ve linear correlation.
- 0: No linear correlation, the two variables are not likely to affect each other.
- -1: Total -ve linear correlation.

The **heatmap** is used to visualise the corelation between colums of our data.

In [242…
```
sns.heatmap(correlation,xticklabels=correlation.columns,
yticklabels=correlation.col
```

Out[242…
```
<AxesSubplot:>
```



corr() function is used to find the correlation between different columns. We can find the **pairwise correlation** between the diffrent columns of the data using the **corr()** method.

In [75]:
```
data.corr()
```

Out[75]:

	Rate	Metascore				Minutes	Release	Budget	Opening We
Sno	1.000000	-0.143460	-0.281745	0.200610	0.247660	-0.032107			
Rate	-0.143460	1.000000	0.786901	0.583813	0.331977	0.265655			
Metascore	-0.281745	0.786901	1.000000	0.509780	0.232213	0.242570			
Minutes	0.200610	0.583813	0.509780	1.000000	0.138387	0.638160			
Release	0.247660	0.331977	0.232213	0.138387	1.000000	0.204316			
Budget	-0.032107	0.265655	0.242570	0.638160	0.204316	1.000000			
Opening_Weekend_USA	-0.112277	0.521689	0.425888	0.637006	0.433480	0.741009			
Gross_USA	-0.062961	0.609582	0.575244	0.630699	0.449439	0.631972			
Gross_Worldwide	-0.031942	0.565348	0.450119	0.603935	0.552735	0.656332			

barh() function is used to draw a horizontal bar chart on the given data.

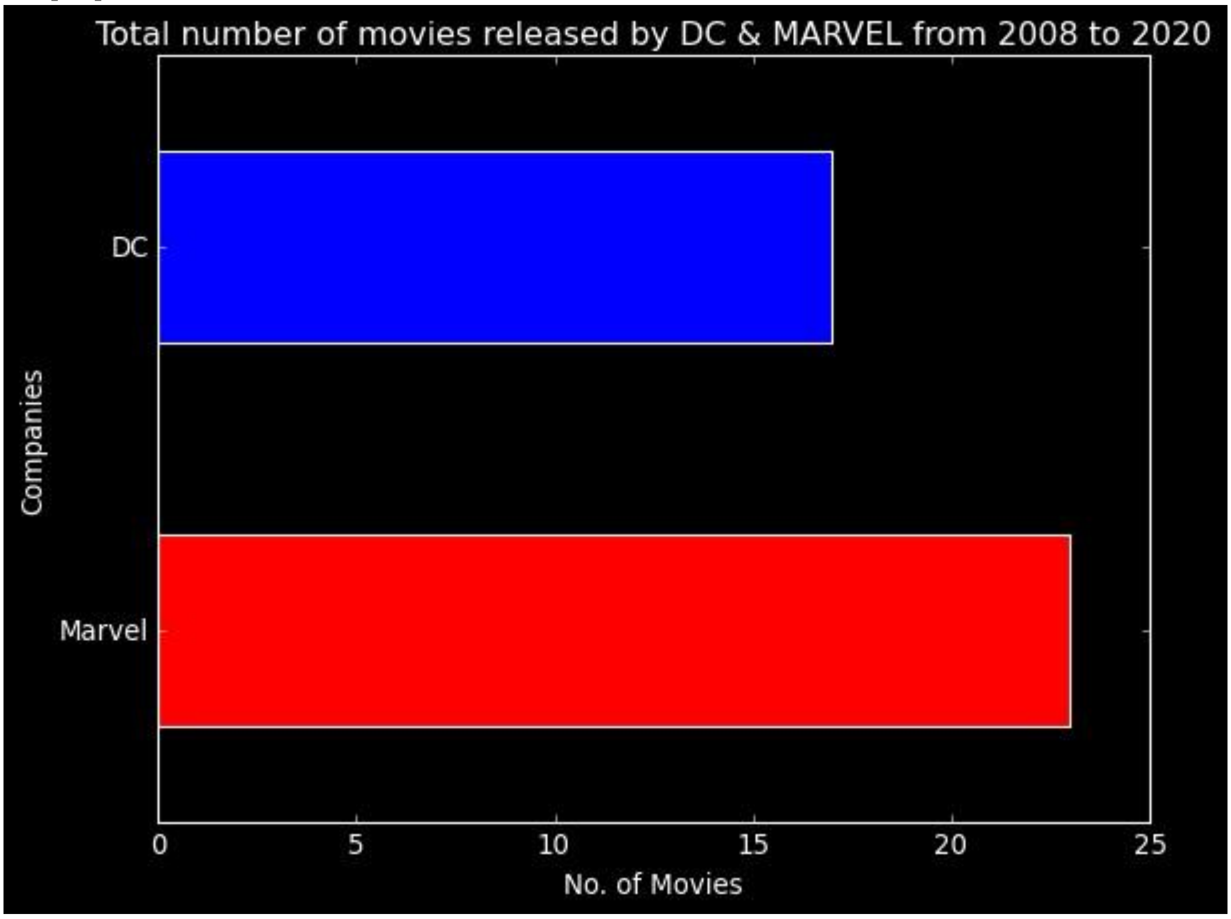
```
In [245…

data.Company.value_counts(normalize= False).plot(kind = 'barh' ,                  '
color=['Red','Blue plt.xlabel('No. of Movies')

 plt.ylabel('Companies')
 plt.title('Total number of movies released by DC & MARVEL from 2008 to 2020')
```

```
Out[245… Text(0.5, 1.0, 'Total number of movies released by DC & MARVEL from 2008 to 2020')
```

```
In [94]:
```

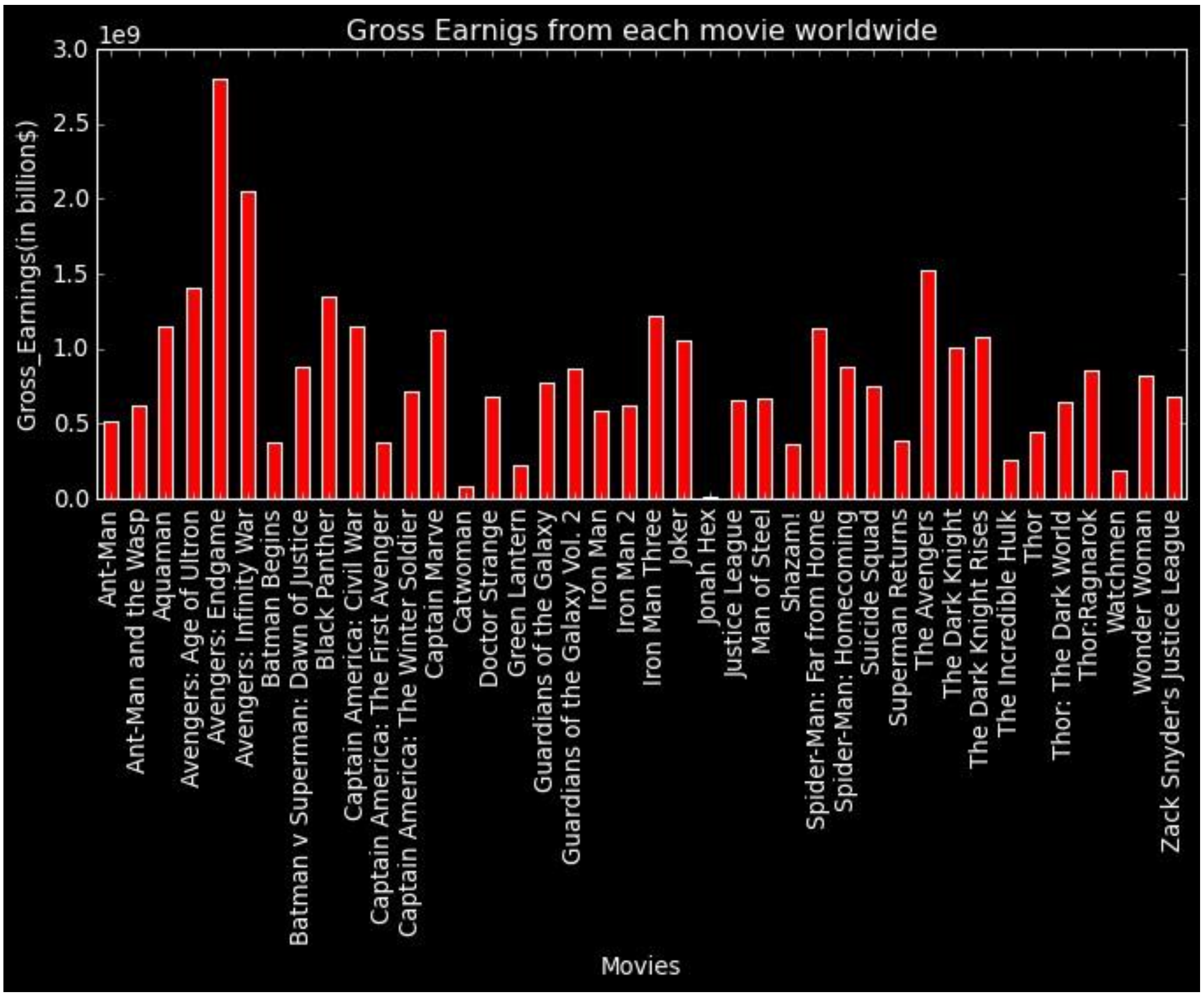


```
data['Original_Title'] = data['Original_Title'].astype(object)
```

bar() function is used to draw a bar chart on the given data.

```
In [244… plt.figure(figsize=(10,4))
data.groupby('Original_Title')['Gross_Worldwide'].mean()
.plot(kind = 'bar', color=[' plt.xlabel('Movies')
plt.ylabel('Gross_Earnings(in billion$)')
plt.title('Gross Earnigs from each movie worldwide')
```

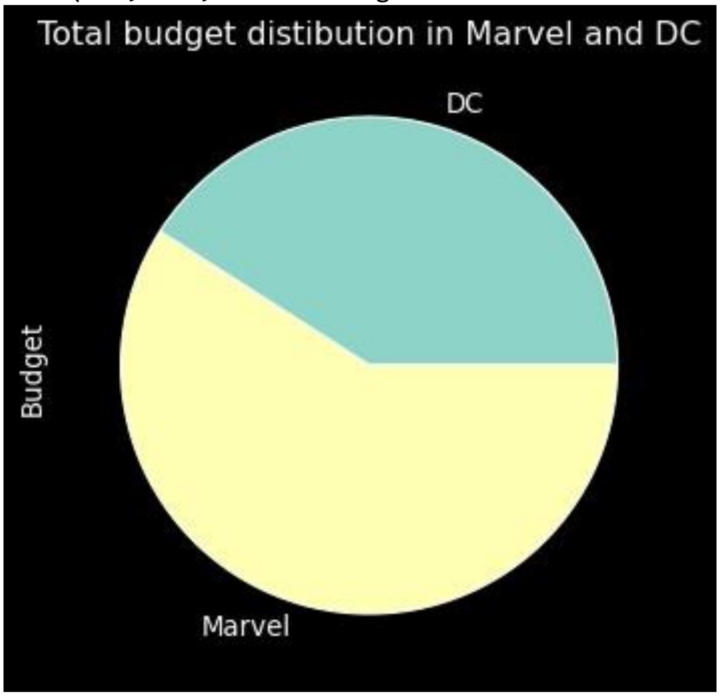
```
Out[244… Text(0.5, 1.0, 'Gross Earnigs from each movie worldwide')
```



pie() function is used to draw a pie chart on the given data.

```
In [246…
plt.figure(figsize=(5,5))
data.groupby('Company')['Budget'].sum().plot(kind = 'pie')
plt.title('Total budget distibution in Marvel and DC')
```

```
Out[246… Text(0.5, 1.0, 'Total budget distibution in Marvel and DC')
```

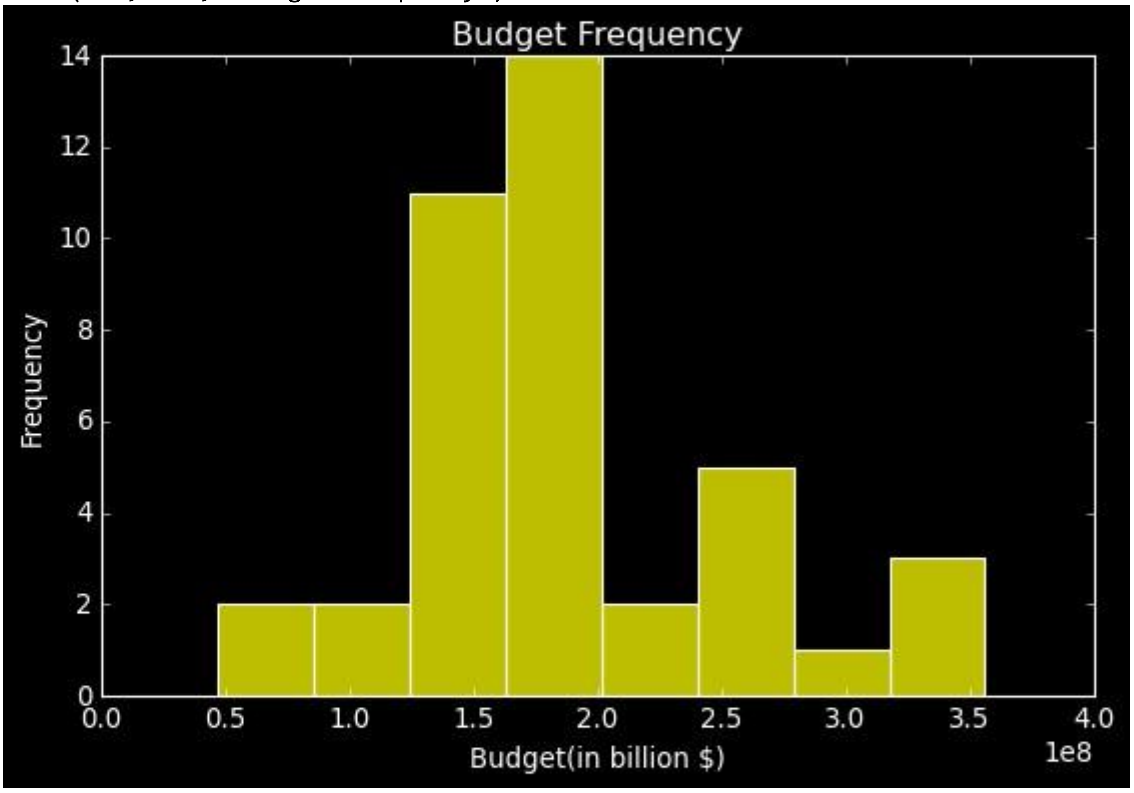


histplot() is used to draw histogram on the given data set.

```
plt.figure(figsize=(8,5))
sns.histplot(data['Budget'], color='yellow')
plt.ylabel('Frequency')
plt.xlabel('Budget(in billion $)')
plt.title('Budget Frequency')
```

In [255…

Out[255…
```
Text(0.5, 1.0, 'Budget Frequency')
```



jointplot() function is used to draw a histogram and scatter plot together

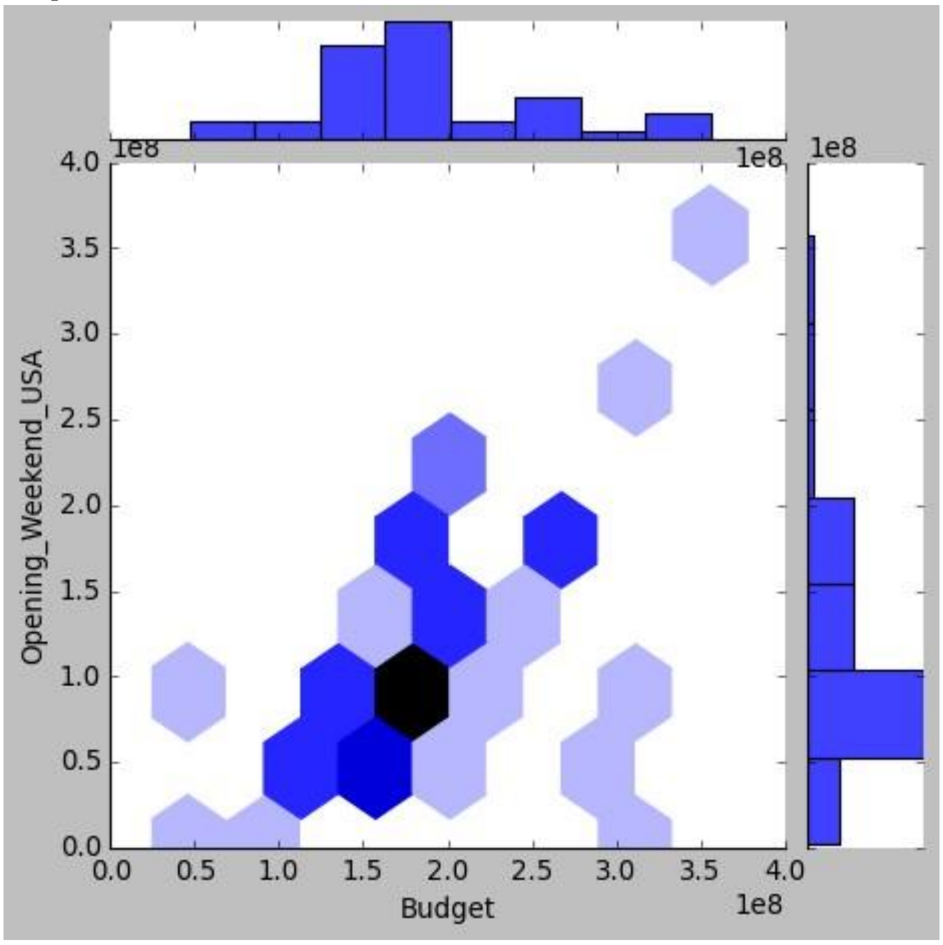
In [236…

```
plt.figure(figsize=(6,5))
plt.style.use('classic')
sns.jointplot(data['Budget'][0:100],
data['Opening_Weekend_USA'][0:100],kind='hex')
```

Out[236…
```
<seaborn.axisgrid.JointGrid at 0x1d3b89348e0>

<Figure size 480x400 with 0 Axes>
```

In [225…

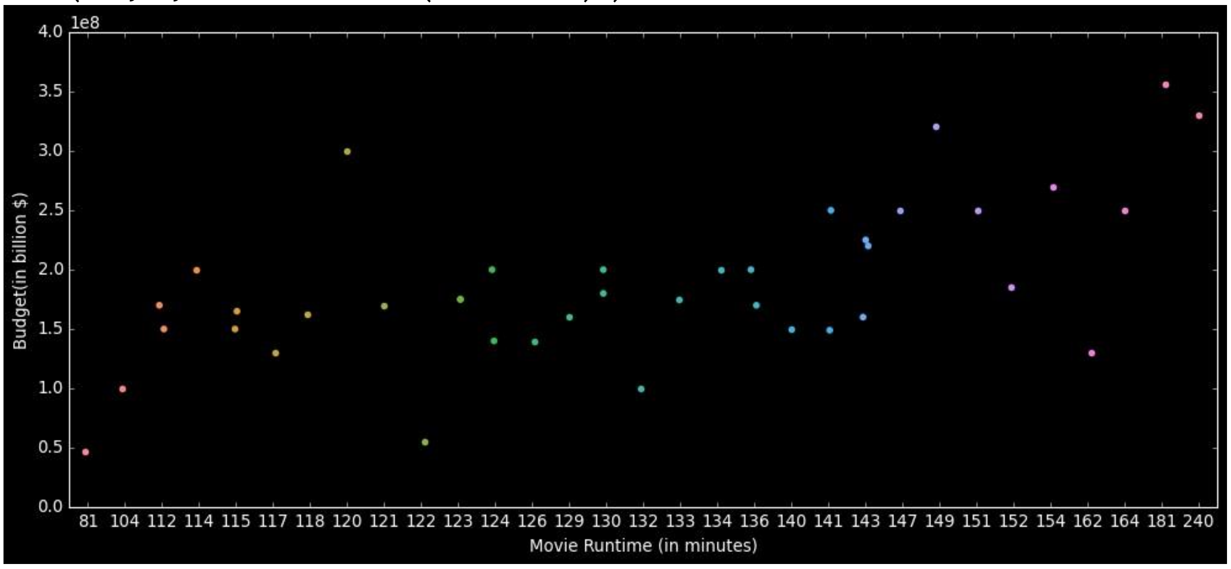


```
plt.style.use('dark_background')
```

stripplot() function is used to draw strip charts which shows the distribution of the values.

```
In [226… plt.figure(figsize=(15,6))
sns.stripplot(x='Minutes', y='Budget',
data=data) plt.ylabel('Budget(in billion
$)') plt.xlabel('Movie Runtime (in
minutes)')
```

```
Out[226… Text(0.5, 0, 'Movie Runtime (in minutes)')
```



rename() function is used to rename the column of the dataframe.

In [283…

```
data.rename(columns={'Original_Title':'Title'}, inplace=True)
```

In [284…

```
data.columns
```

Out[284…

```
Index(['Sno', 'Title', 'Company', 'Rate', 'Metascore', 'Minutes', 'Release',
 'Budget', 'Opening_Weekend_USA', 'Gross_USA', 'Gross_Worldwide'],
 dtype='object')
```

Conclusion

From the analysis, we can conclude that Budget is out target because it is highly correlated with Minutes, Gross_Worldwide, Opening_weekend_USA, and Gross_USA.

Also we can say that Marvel Studios as been performing better than DC as the releases per year is greater than DC and they also focus on their budgeting as they increase their budget their Earnings also goes up.

DC have less releases in the time span of 2008-2020 as compared to marvel and the Rating and Metacritics score are more in marvel studios.

As the Runtime of a movie increases the Budget of the movie also goes up distinctively.