

IMDB Movies Dataset & Analysis

This dataset is about top 1k movies based on IMDB rating, Meta_Score, No_of_votes, gross, overviews and genre from Kaggle. This will help to find top rated movies for people. There are 1000 rows and 16 columns with heading Poster_Link, Series_Title, Released_Year, Certificate, Runtime, Genre, IMDB_Rating, Overview, Meta_Score, Director, Stars, No_of_votes, and Gross.

Project Members

1. Aniket Bhavar(321)
2. Deepchand Prajapati(322)
3. Abhishek Hulke(323)

Problem Statement

Analysis of last 94yrs. (i.e., from 1920 – 2014) movies data IMDB and come up with the success factors of any movie and their correlation.

Data Description

The dataset contains the following Columns:

- Poster Link: Link of the poster that imdb using
- Series Title: Name of the movie
- Released Year: Year at which that movie released
- Certificate: Certificate earned by that movie
- Runtime: Total runtime of the movie
- Genre: Genre of the movie
- IMDB Rating: Rating of the movie at IMDB site
- Overview: mini story/ summary
- Meta score: Score earned by the movie
- Director: Name of the Director Star1, Star2, Star3, Star4: Name of the Stars
- No of votes: Total number of votes
- Gross: Money earned by that movie

```
In [1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
In [2]: df=pd.read_csv('imdb_top_1000.csv')
```

```
In [3]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Poster_Link      1000 non-null   object  
 1   Series_Title     1000 non-null   object  
 2   Released_Year    1000 non-null   object  
 3   Certificate      899 non-null   object  
 4   Runtime          1000 non-null   object  
 5   Genre            1000 non-null   object  
 6   IMDB_Rating      1000 non-null   float64 
 7   Overview         1000 non-null   object  
 8   Meta_score       843 non-null   float64 
 9   Director         1000 non-null   object  
 10  Star1            1000 non-null   object  
 11  Star2            1000 non-null   object  
 12  Star3            1000 non-null   object  
 13  Star4            1000 non-null   object  
 14  No_of_Votes     1000 non-null   int64  
 15  Gross            831 non-null   object  
dtypes: float64(2), int64(1), object(13)
memory usage: 125.1+ KB
```

Shape of the data

```
In [4]: print("The shape of the data is: {} rows and {} columns".format(df.shape[0], df.shape[1]))
```

The shape of the data is: 1000 rows and 16 columns

```
In [5]: df.shape
```

```
Out[5]: (1000, 16)
```

```
In [6]: df.head(3)
```

Out[6]:

	Poster_Link	Series_Title	Released_Year	Certificate	Runtime	Genre	IM
0	https://m.media-amazon.com/images/M/MV5BMDFkYT...	The Shawshank Redemption	1994	A	142 min	Drama	
1	https://m.media-amazon.com/images/M/MV5BM2MyNj...	The Godfather	1972	A	175 min	Crime, Drama	
2	https://m.media-amazon.com/images/M/MV5BMTMxNT...	The Dark Knight	2008	UA	152 min	Action, Crime, Drama	



Number of Null values

In [7]: `df.isnull().sum()`

```
Out[7]:
```

Poster_Link	0
Series_Title	0
Released_Year	0
Certificate	101
Runtime	0
Genre	0
IMDB_Rating	0
Overview	0
Meta_Score	157
Director	0
Star1	0
Star2	0
Star3	0
Star4	0
No_of_Votes	0
Gross	169
dtype:	int64

Data Cleaning

In [8]: `df.drop(['Certificate', 'Star1', 'Star2', 'Star3', 'Star4'], axis=1, inplace=True)`
`df.head(2)`

Out[8]:

	Poster_Link	Series_Title	Released_Year	Runtime	Genre	IMDB_Rating
--	-------------	--------------	---------------	---------	-------	-------------

0	https://m.media-amazon.com/images/M/MV5BMDFkYT...	The Shawshank Redemption	1994	142 min	Drama	9.3
---	---	--------------------------	------	---------	-------	-----

1	https://m.media-amazon.com/images/M/MV5BM2MyNj...	The Godfather	1972	175 min	Crime, Drama	9.2
---	---	---------------	------	---------	--------------	-----

In [9]: `data = df.rename(columns={"Series_Title": "Movies_Titile"})`In [10]: `data.tail(2)`

Out[10]:

	Poster_Link	Movies_Titile	Released_Year	Runtime	Genre	IMDB_Rat
--	-------------	---------------	---------------	---------	-------	----------

998	https://m.media-amazon.com/images/M/MV5BZTBmMj...	Lifeboat	1944	97 min	Drama, War	
-----	---	----------	------	--------	------------	--

999	https://m.media-amazon.com/images/M/MV5BMTY5OD...	The 39 Steps	1935	86 min	Crime, Mystery, Thriller	
-----	---	--------------	------	--------	--------------------------	--

Performing Different Operation

In [11]: `type(df)`Out[11]: `pandas.core.frame.DataFrame`In [12]: `df[['IMDB_Rating', 'Meta_score']]`

Out[12]:

	IMDB_Rating	Meta_score
0	9.3	80.0
1	9.2	100.0
2	9.0	84.0
3	9.0	90.0
4	9.0	96.0
...
995	7.6	76.0
996	7.6	84.0
997	7.6	85.0
998	7.6	78.0
999	7.6	93.0

1000 rows × 2 columns

In [13]:

`df.loc[[432, 435], ['Runtime', 'Meta_score']]`

Out[13]:

	Runtime	Meta_score
432	138 min	91.0
435	174 min	95.0

In [14]:

`df.iloc[2:4]`

Out[14]:

	Poster_Link	Series_Title	Released_Year	Runtime	Genre	IMDB_Rating
2	https://m.media-amazon.com/images/M/MV5BMXTMxNT... amazon.com/images/M/MV5BMXTMxNT...	The Dark Knight	2008	152 min	Action, Crime, Drama	9.0
3	https://m.media-amazon.com/images/M/MV5BMWMMwMG... amazon.com/images/M/MV5BMWMMwMG...	The Godfather: Part II	1974	202 min	Crime, Drama	9.0

In [15]:

`# It will drop all the null values
df.dropna()`

Out[15]:

		Poster_Link	Series_Title	Released_Year	Runtime	Genre	IMDB_I
0		https://m.media-amazon.com/images/M/MV5BMDFkYT...	The Shawshank Redemption	1994	142 min	Drama	
1		https://m.media-amazon.com/images/M/MV5BM2MyNj...	The Godfather	1972	175 min	Crime, Drama	
2		https://m.media-amazon.com/images/M/MV5BMTMxNT...	The Dark Knight	2008	152 min	Action, Crime, Drama	
3		https://m.media-amazon.com/images/M/MV5BMWWMwMG...	The Godfather: Part II	1974	202 min	Crime, Drama	
4		https://m.media-amazon.com/images/M/MV5BMWU4N2...	12 Angry Men	1957	96 min	Crime, Drama	
...
990		https://m.media-amazon.com/images/M/MV5BYjRmY2...	Giù la testa	1971	157 min	Drama, War, Western	
991		https://m.media-amazon.com/images/M/MV5BMzAyND...	Kelly's Heroes	1970	144 min	Adventure, Comedy, War	
992		https://m.media-amazon.com/images/M/MV5BMjAwMT...	The Jungle Book	1967	78 min	Animation, Adventure, Family	

		Poster_Link	Series_Title	Released_Year	Runtime	Genre	IMDB_Rating
994		https://m.media-amazon.com/images/M/MV5BZjQyMG...	A Hard Day's Night	1964	87 min	Comedy, Music, Musical	7.9

997		https://m.media-amazon.com/images/M/MV5BM2U3Yz...	From Here to Eternity	1953	118 min	Drama, Romance, War	8.1
-----	--	---	-----------------------	------	---------	---------------------	-----

750 rows × 11 columns

◀	▶
---	---

In [16]: `df.describe()`

Out[16]:

	IMDB_Rating	Meta_score	No_of_Votes
count	1000.000000	843.000000	1.000000e+03
mean	7.949300	77.971530	2.736929e+05
std	0.275491	12.376099	3.273727e+05
min	7.600000	28.000000	2.508800e+04
25%	7.700000	70.000000	5.552625e+04
50%	7.900000	79.000000	1.385485e+05
75%	8.100000	87.000000	3.741612e+05
max	9.300000	100.000000	2.343110e+06

In [17]: `df.describe().transpose()`

Out[17]:

	count	mean	std	min	25%	50%	75%	max
IMDB_Rating	1000.0	7.94930	0.275491	7.6	7.70	7.9	8.10	9.3
Meta_score	843.0	77.97153	12.376099	28.0	70.00	79.0	87.00	100.0
No_of_Votes	1000.0	273692.91100	327372.703934	25088.0	55526.25	138548.5	374161.25	2343110.0

◀	▶
---	---

In [18]: `type(df['Meta_score'])`

Out[18]: `pandas.core.series.Series`

In [19]: `df_avg = df['IMDB_Rating'].mean()`

```
print('The overall average of all the IMDB_Rating of out of 10 is: ', df_avg)
```

The overall average of all the IMDB_Rating of out of 10 is: 7.949300000000012

In [20]: `var=df['Series_Title'].value_counts()`

```
print(var)

Drishyam           2
The Shawshank Redemption  1
Awakenings          1
Tombstone           1
The Sandlot          1
..
Guardians of the Galaxy  1
Blade Runner 2049     1
Her                  1
Bohemian Rhapsody    1
The 39 Steps          1
Name: Series_Title, Length: 999, dtype: int64
```

In [21]: `df.Director.value_counts().head(10)`

```
Out[21]: Alfred Hitchcock    14
Steven Spielberg      13
Hayao Miyazaki        11
Martin Scorsese       10
Akira Kurosawa        10
Stanley Kubrick       9
Billy Wilder          9
Woody Allen           9
Christopher Nolan     8
Quentin Tarantino     8
Name: Director, dtype: int64
```

In [22]: `var.index`

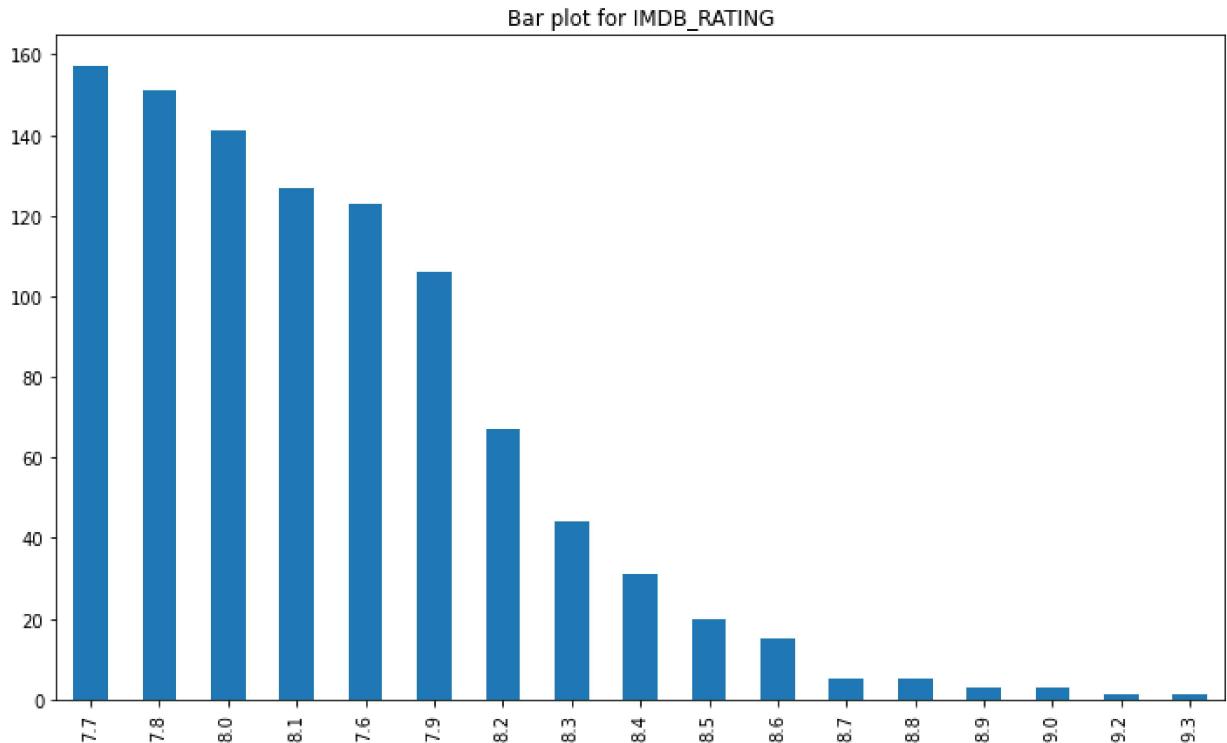
```
Out[22]: Index(['Drishyam', 'The Shawshank Redemption', 'Awakenings', 'Tombstone',
   'The Sandlot', 'The Remains of the Day', 'Naked', 'The Fugitive',
   'A Bronx Tale', 'Batman: Mask of the Phantasm',
   ...
   'Gully Boy', 'Special Chabbis', 'Short Term 12',
   'Serbuhan maut 2: Berandal', 'The Imitation Game',
   'Guardians of the Galaxy', 'Blade Runner 2049', 'Her',
   'Bohemian Rhapsody', 'The 39 Steps'],
  dtype='object', length=999)
```

Data Visualization

In [23]: `var = df['IMDB_Rating'].value_counts()
print(var)
var.plot(kind='bar', figsize = (12,7), title='Bar plot for IMDB_RATING')`

```

7.7    157
7.8    151
8.0    141
8.1    127
7.6    123
7.9    106
8.2    67
8.3    44
8.4    31
8.5    20
8.6    15
8.7    5
8.8    5
8.9    3
9.0    3
9.2    1
9.3    1
Name: IMDB_Rating, dtype: int64
<AxesSubplot:title={'center':'Bar plot for IMDB_RATING'}>
Out[23]:
```

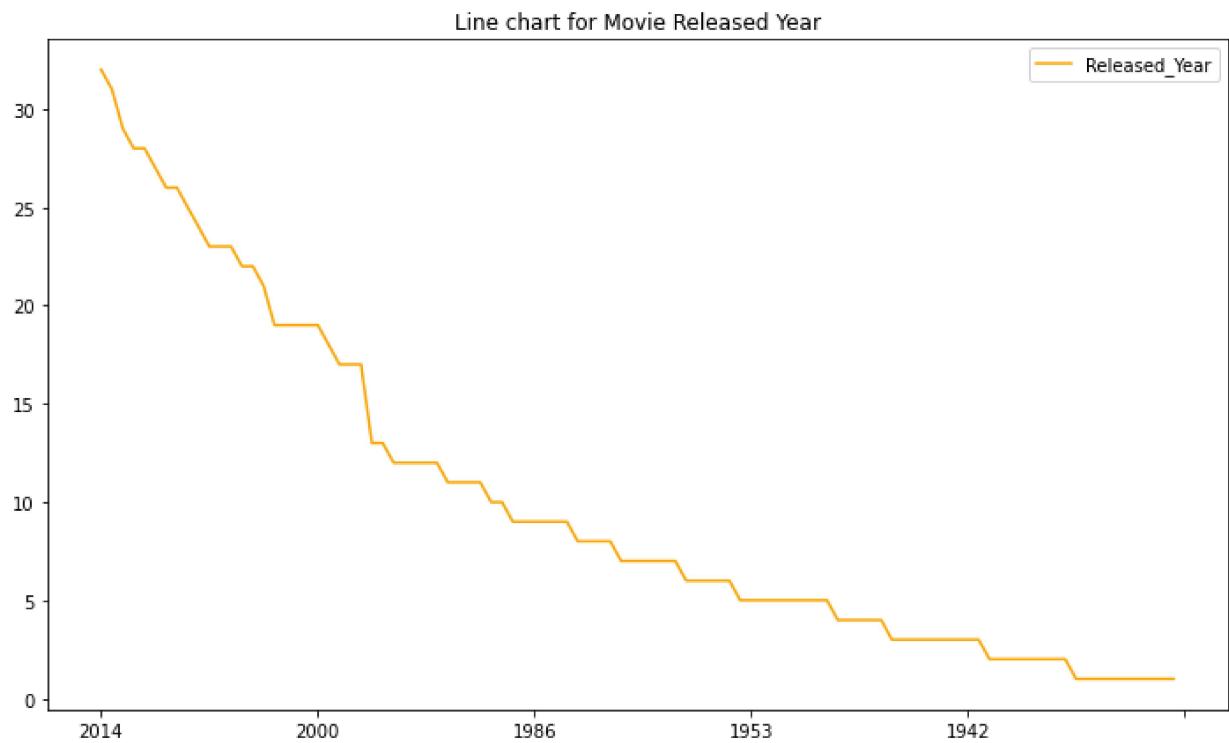


```
In [24]: Year = df['Released_Year'].value_counts()
print(Year)
Year.plot.line(title='Line chart for Movie Released Year',color = ('Orange'), figsize
```

```

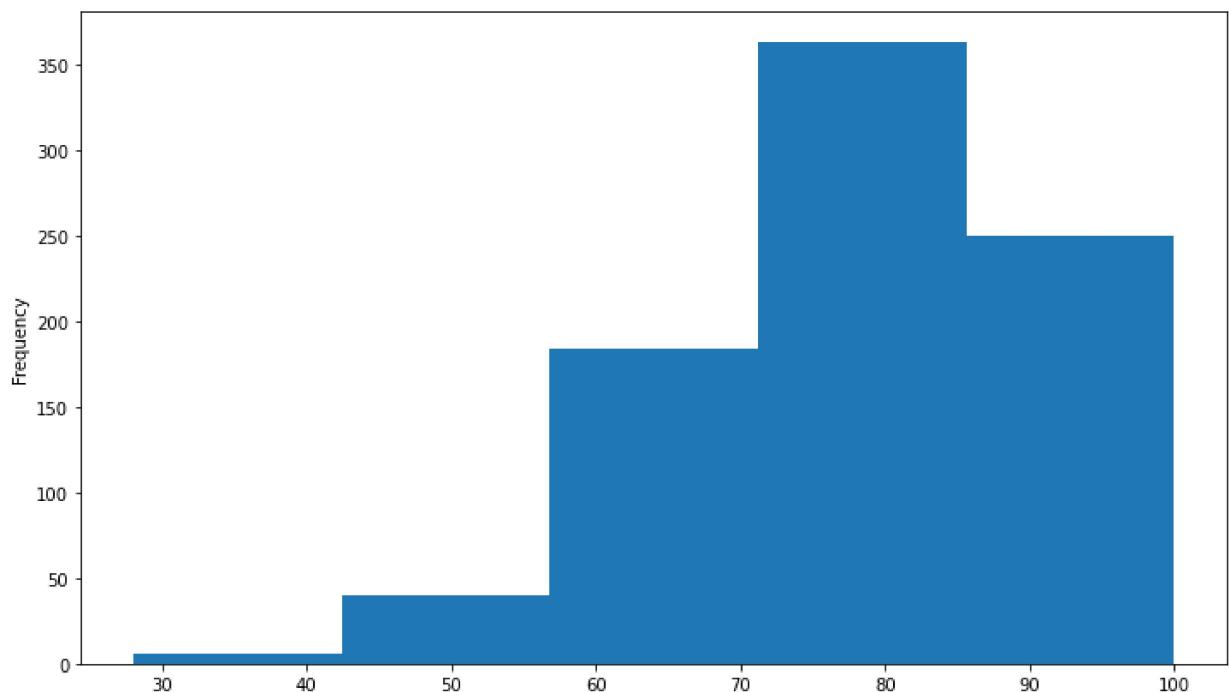
2014    32
2004    31
2009    29
2013    28
2016    28
...
1926    1
1936    1
1924    1
1921    1
PG      1
Name: Released_Year, Length: 100, dtype: int64
```

```
Out[24]: <AxesSubplot:title={'center':'Line chart for Movie Released Year'}>
```



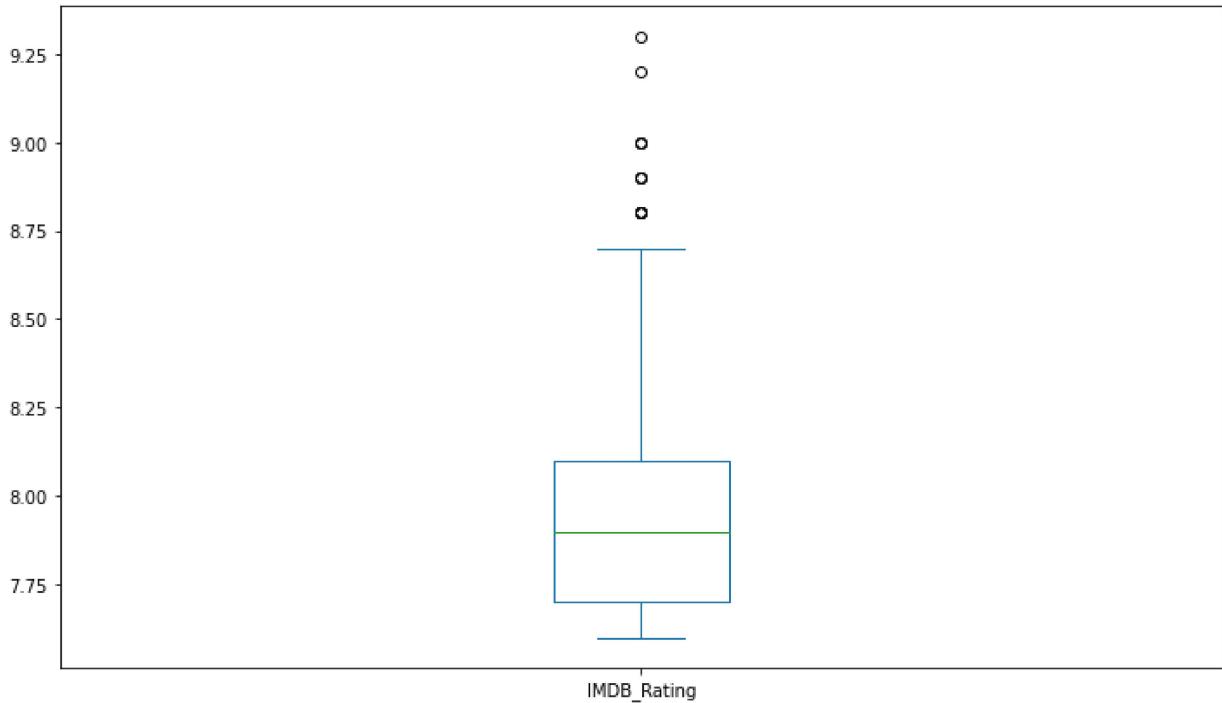
```
In [25]: df.Meta_score.plot(kind = 'hist', figsize = (12,7), bins = 5 )
```

```
Out[25]: <AxesSubplot:ylabel='Frequency'>
```



```
In [26]: df.IMDB_Rating.plot(kind = 'box', figsize = (12,7))
```

```
Out[26]: <AxesSubplot:>
```



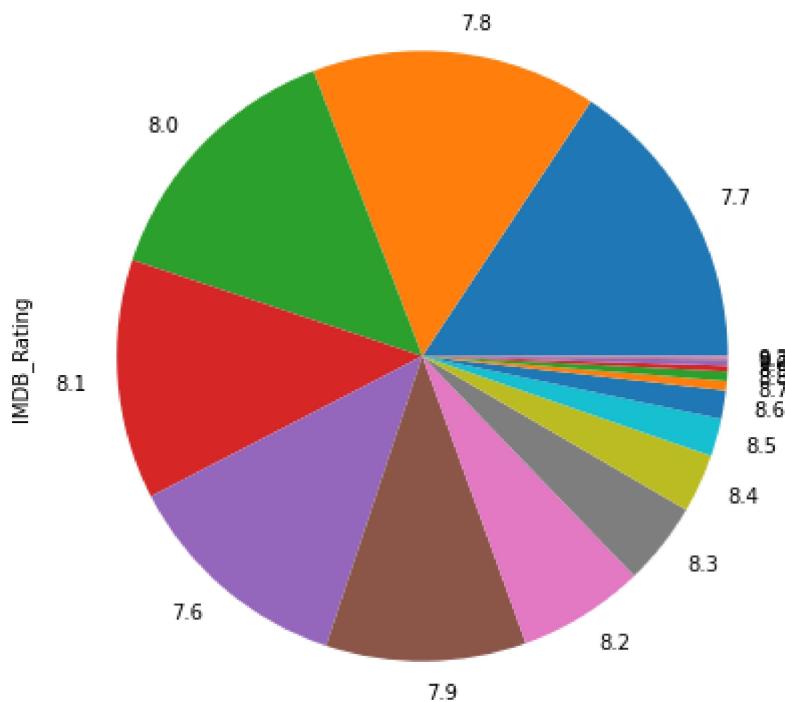
```
In [27]: IR=df['IMDB_Rating'].value_counts()
print(IR)
IR.plot.pie(title='Pie chart for IMDB_Rating', figsize = (12,7), legend=False)
```

7.7	157
7.8	151
8.0	141
8.1	127
7.6	123
7.9	106
8.2	67
8.3	44
8.4	31
8.5	20
8.6	15
8.7	5
8.8	5
8.9	3
9.0	3
9.2	1
9.3	1

Name: IMDB_Rating, dtype: int64

```
Out[27]: <AxesSubplot:title={'center':'Pie chart for IMDB_Rating'}, ylabel='IMDB_Rating'>
```

Pie chart for IMDB_Rating



Looking For Correlation

```
In [28]: print(df['Gross'].head(1))
df['Gross'] = df['Gross'].str.replace(',', '')
print(df['Gross'].head(1))

df['Gross'] = df['Gross'].astype('float64')
df['Gross'] = df['Gross'].replace(np.nan, 0)
```

```
0    28,341,469
Name: Gross, dtype: object
0    28341469
Name: Gross, dtype: object
```

```
In [29]: df['Gross'] = df['Gross'].astype(int)
```

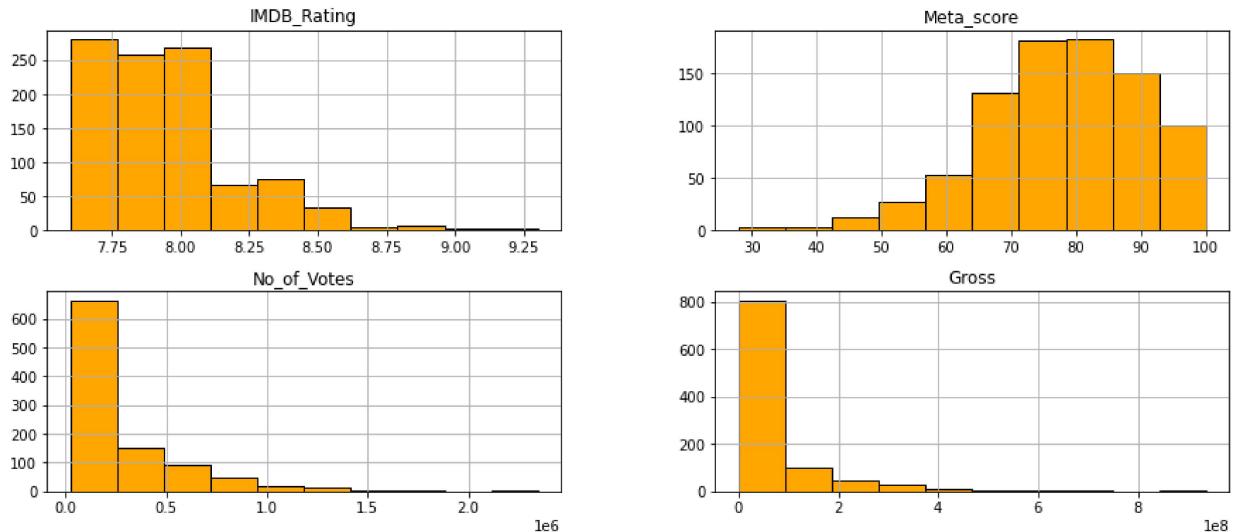
```
In [30]: df['Gross'].dtype
```

```
Out[30]: dtype('int32')
```

```
In [31]: df.corr()
```

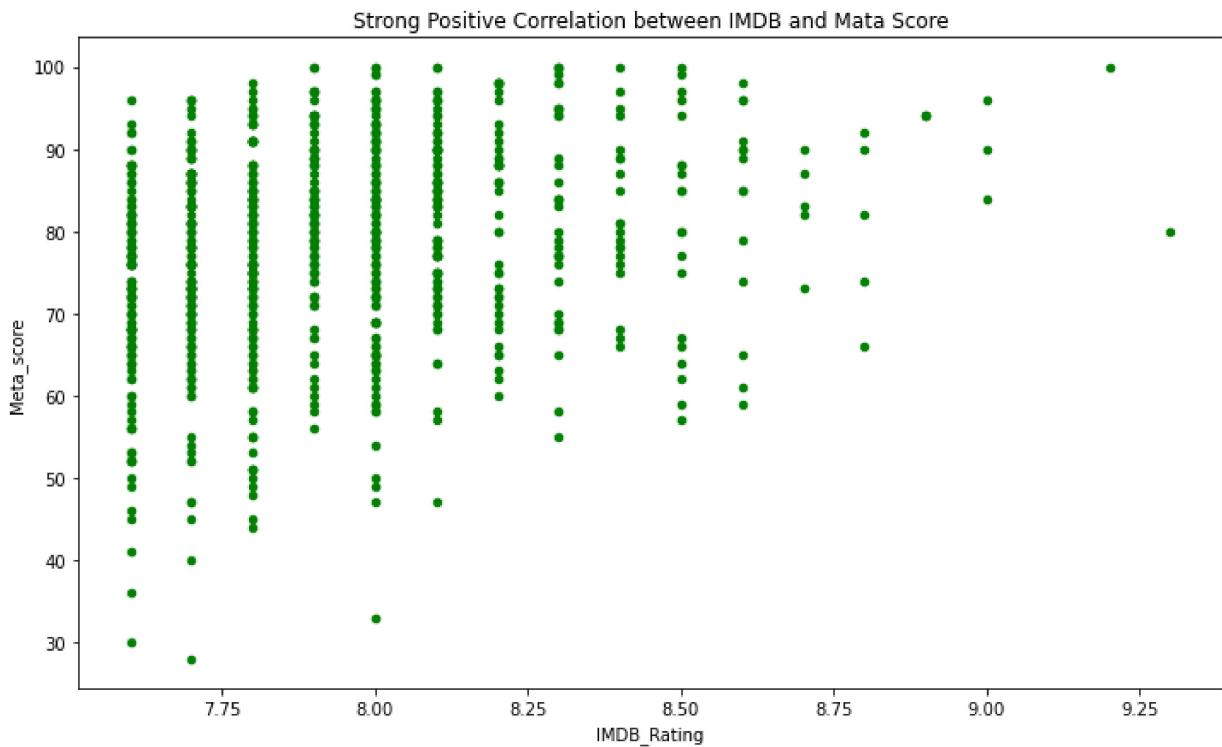
	IMDB_Rating	Meta_score	No_of_Votes	Gross
IMDB_Rating	1.000000	0.268531	0.494979	0.082381
Meta_score	0.268531	1.000000	-0.018507	-0.053659
No_of_Votes	0.494979	-0.018507	1.000000	0.602128
Gross	0.082381	-0.053659	0.602128	1.000000

```
In [32]: numerical_attributes = ['IMDB_Rating', 'Meta_score', 'No_of_Votes', 'Gross']
df[numerical_attributes].hist(figsize = (15, 6), color = 'orange', edgecolor = 'black')
```



```
In [33]: df.plot(kind='scatter', x = 'IMDB_Rating', y = 'Meta_score', title = ("Strong Positive Correlation between IMDB and Mata Score"))
```

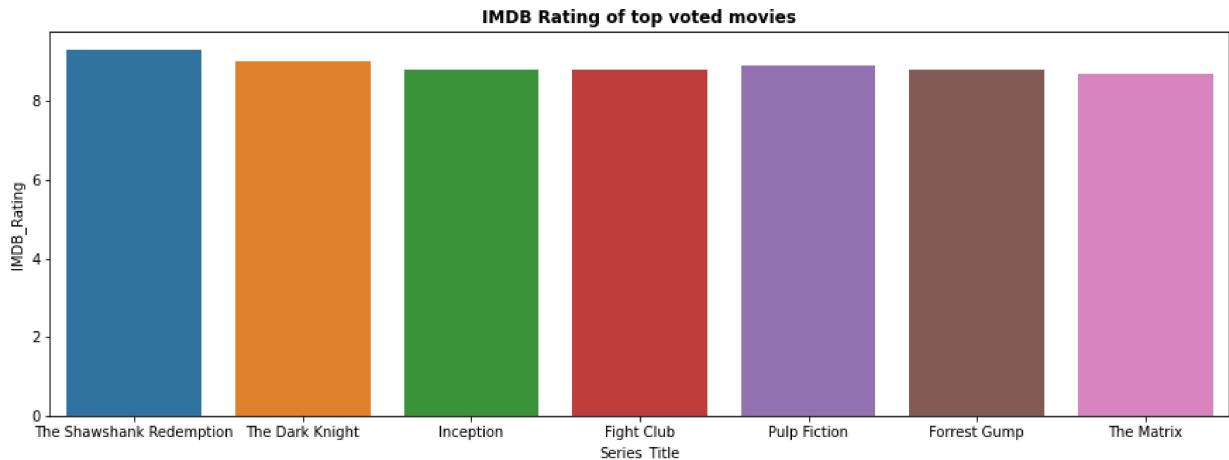
```
Out[33]: <AxesSubplot:title={'center':'Strong Positive Correlation between IMDB and Mata Score'}, xlabel='IMDB_Rating', ylabel='Meta_score'>
```



Top voted movies rating

```
In [34]: top_voted = df.sort_values(['No_of_Votes'], ascending = False)
```

```
In [35]: fig,axs=plt.subplots(figsize=(15,5))
g=sns.barplot(x=top_voted['Series_Title'][:7],y=top_voted['IMDB_Rating'][:7])
g.set_title("IMDB Rating of top voted movies", weight = "bold")
plt.show()
```



In []:

Most occurred directors

```
In [36]: fig,axs=plt.subplots(figsize=(20,5))
g=sns.barplot(x=data['Director'].value_counts()[:10].index,y=data['Director'].value_c
g.set_title("Most Occurred Directors", weight = "bold")
g.set_xlabel("Directors", weight = "bold")
g.set_ylabel("Movies", weight = "bold")
plt.show()
```

