IMPORTING DATASET

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import math
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
df = pd.read_csv("/content/drive/MyDrive/Movies/movies.csv", encoding='latin1')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15509 entries, 0 to 15508
Data columns (total 10 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Name      15509 non-null  object
 1   Year      14981 non-null  object
 2   Duration  7240 non-null   object
 3   Genre     13632 non-null  object
 4   Rating    7919 non-null   float64
 5   Votes     7920 non-null   object
 6   Director  14984 non-null  object
 7   Actor 1   13892 non-null  object
 8   Actor 2   13125 non-null  object
 9   Actor 3   12365 non-null  object
dtypes: float64(1), object(9)
memory usage: 1.2+ MB
```

```
df.isnull().sum()
```

```
Name           0
Year         528
Duration    8269
Genre       1877
Rating      7590
Votes       7589
Director     525
Actor 1     1617
Actor 2     2384
Actor 3     3144
dtype: int64
```

```
df.head(10)
```

	Name	Year	Duration	Genre	Rating	Votes	Director	Actor 1	Actor 2	Actor 3
0		NaN	NaN	Drama	NaN	NaN	J.S. Randhawa	Manmauji	Birbal	Rajendra Bhatia
1	#Gadhvi (He thought he was Gandhi)	(2019)	109 min	Drama	7.0	8	Gaurav Bakshi	Rasika Dugal	Vivek Ghamande	Arvind Jangid
2	#Homecoming	(2021)	90 min	Drama, Musical	NaN	NaN	Soumyajit Majumdar	Sayani Gupta	Plabita Borthakur	Roy Angana
3	#Yaaram	(2019)	110 min	Comedy,	4.4	35	Ovais Khan	Prateik	Ishita Rai	Siddhant

Data Cleaning

	Name	Year	Duration	Genre	Rating	Votes	Director	Actor 1	Actor 2	Actor 3
	Aur Pyaar Ho			Comedy				Bobby	Aishwarya Rai	Shammi
				Drama			Shoojit	Jimmy	Minissha	Yashpal

```
df.dropna(subset=df.columns[1:9],how='all',inplace=True)
```

```
df.dropna(subset=['Name','Year'],how='all',inplace=True)
```

```
df.drop_duplicates(['Name','Year'],keep='first',inplace=True)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 15477 entries, 0 to 15508
Data columns (total 10 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Name      15477 non-null  object 
 1   Year      14958 non-null  object 
 2   Duration  7235 non-null   object 
 3   Genre     13614 non-null  object 
 4   Rating    7915 non-null   float64
 5   Votes     7916 non-null   object 
 6   Director  14962 non-null  object 
 7   Actor 1   13875 non-null  object 
 8   Actor 2   13110 non-null  object 
 9   Actor 3   12355 non-null  object 
dtypes: float64(1), object(9)
memory usage: 1.3+ MB
```

```
df.dropna(subset=['Year'],inplace=True)
```

```
df['Year']=df['Year'].str.extract(r'([0-9].{0,3})',expand=False)
```

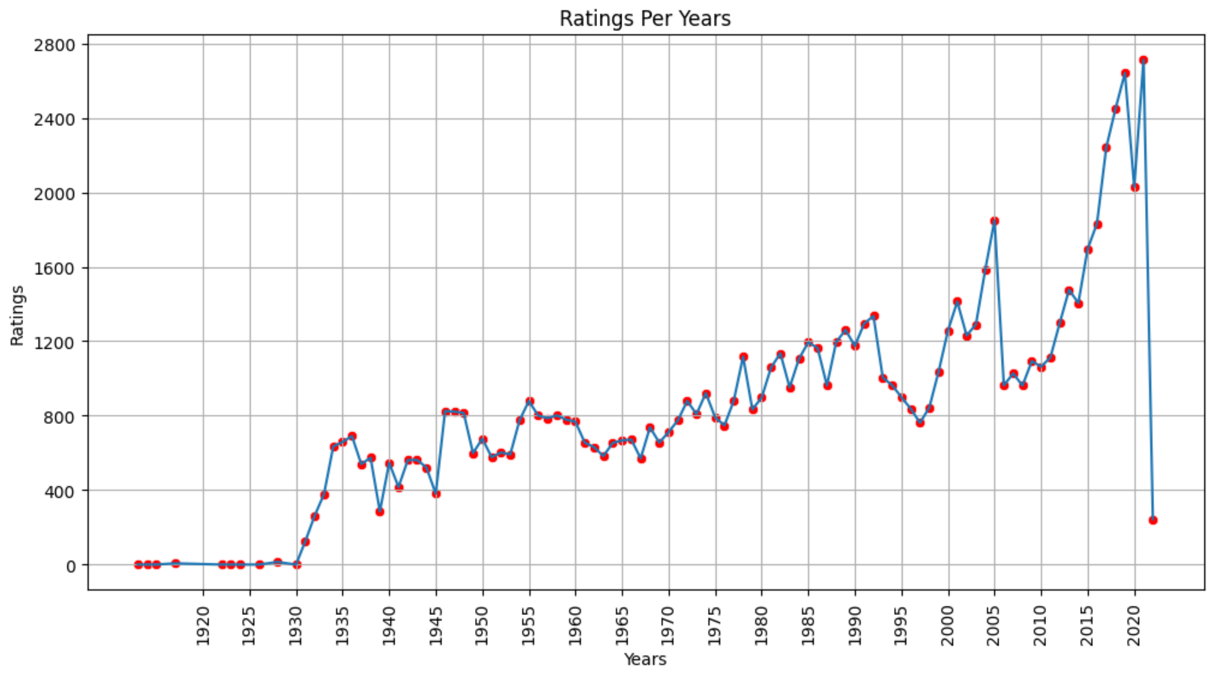
```
df['Duration']=df['Duration'].str.extract(r'([0-9]+)',expand=False)
```

```
def get_mode_with_default(x):
    mode_result = x.mode()
    if not mode_result.empty:
        return mode_result[0]
    else:
        return 'unknown'

df['Actor 1']=df['Actor 1'].fillna(df.groupby('Year')['Actor 1'].transform(get_mode_with_default))
df['Actor 2']=df['Actor 2'].fillna(df.groupby('Year')['Actor 2'].transform(get_mode_with_default))
df['Actor 3']=df['Actor 3'].fillna(df.groupby('Year')['Actor 3'].transform(get_mode_with_default))


df['Director']=df.groupby(['Year','Actor 1','Actor 2','Actor 3'])['Director'].transform(get_mode_with_default)


df['Duration']=pd.to_numeric(df['Duration'])


def get_mean_with_default(x):
    mean_result = x.mean()
    if not math.isnan(mean_result):
            return round(mean_result)
    else:
        return 0
df['Duration']=df.groupby(['Year','Director','Actor 1','Actor 2','Actor 3'])['Duration'].transform(get_mean_with_default)


df['Rating']=df.groupby(['Director','Actor 1'])['Rating'].transform(lambda x:x.mean())
df['Rating']=df.groupby(['Director','Actor 2'])['Rating'].transform(lambda x:x.mean())
df['Rating']=df.groupby(['Director','Actor 3'])['Rating'].transform(lambda x:x.mean())
df['Rating']=df.groupby(['Year','Director'])['Rating'].transform(lambda x:x.mean())
df['Rating']=df.groupby('Year')['Rating'].transform(lambda x:x.mean())
df['Year']=pd.to_numeric(df['Year'])


df['Votes']=df['Votes'].str.extract(r'([0-9]+)',expand=False)
df['Votes']=pd.to_numeric(df['Votes'])


df['Votes']=df.groupby(['Year','Rating'])['Votes'].transform(lambda x:x.mean())


df['Votes']=df.groupby('Year')['Votes'].transform(lambda x:x.mean())


df.info()
```

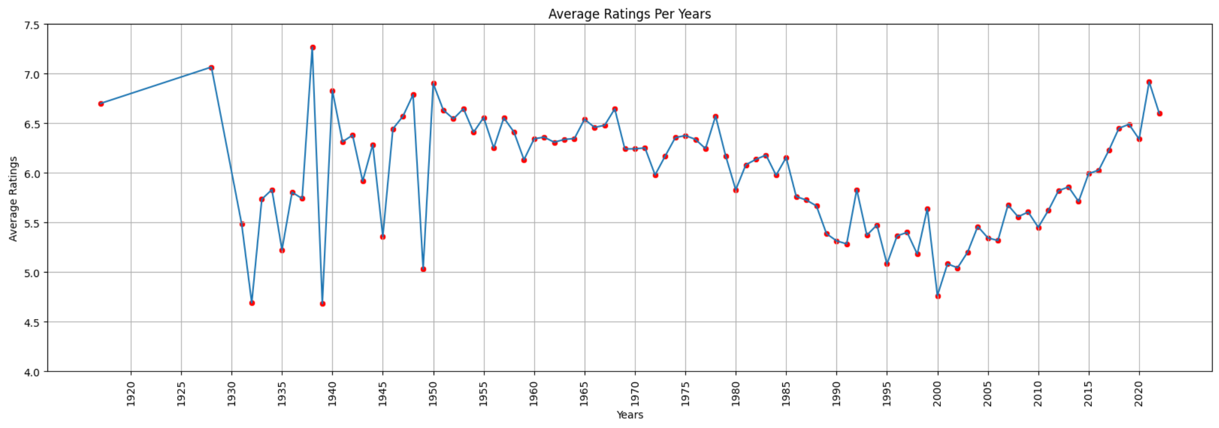
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 14958 entries, 1 to 15508
Data columns (total 10 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Name      14958 non-null  object
 1   Year      14958 non-null  int64
 2   Duration  14958 non-null  int64
 3   Genre     13123 non-null  object
 4   Rating    14947 non-null  float64
 5   Votes     14908 non-null  float64
 6   Director  14958 non-null  object
 7   Actor 1   14958 non-null  object
 8   Actor 2   14958 non-null  object
 9   Actor 3   14958 non-null  object
dtypes: float64(2), int64(2), object(6)
memory usage: 1.3+ MB
```

EDA

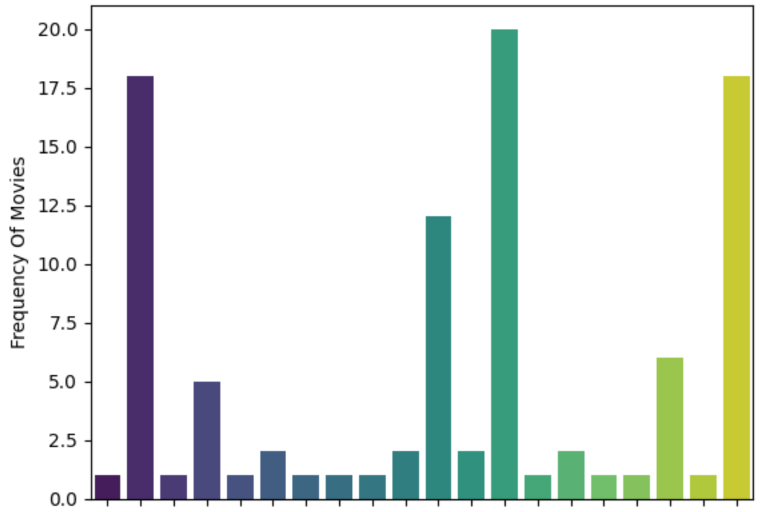
```
#Year with best rating
rating_sum=df.groupby('Year')['Rating'].sum().reset_index()

plt.figure(figsize=(12,6))
sns.lineplot(x='Year',y='Rating',data=rating_sum)
sns.scatterplot(x='Year',y='Rating',data=rating_sum,color='r')
plt.yticks(np.arange(0,3000,400))
plt.xticks(np.arange(1920,2025,5))
plt.ylabel('Ratings')
plt.xlabel('Years')
plt.title('Ratings Per Years')
plt.xticks(rotation=90)
plt.grid(True)
plt.show()
```



```
#Year with best average rating
rating_avg=df.groupby('Year')['Rating'].mean().reset_index()

plt.figure(figsize=(20,6))
sns.lineplot(x='Year',y='Rating',data=rating_avg)
sns.scatterplot(x='Year',y='Rating',data=rating_avg,color='r')
plt.yticks(np.arange(4,8,0.5))
plt.xticks(np.arange(1920,2025,5))
plt.ylabel('Average Ratings')
plt.xlabel('Years')
plt.title('Average Ratings Per Years')
plt.xticks(rotation=90)
plt.grid(True)
plt.show()
```

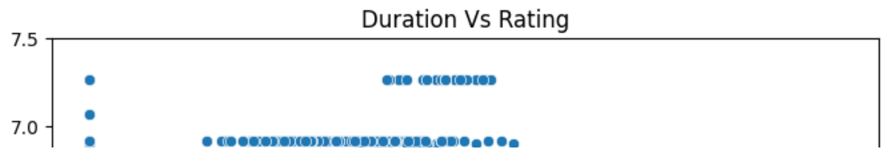


```
#Top 20 Directors by Frequency of Movies
top_20=df.groupby('Director')['Name'].count()[0:20]

sns.barplot(x=top_20.index,y=top_20.values,data=df,palette='viridis')
plt.xticks(rotation=90)
plt.ylabel('Frequency Of Movies')
plt.xlabel('Director')
plt.show()
```



```
#Does length of movie have any impact with the rating
corr_leng_rat=df['Duration'].corr(df['Rating'])
print(f"Correlation Of Duration And Rating is {corr_leng_rat}")
#show there is no impact of duration on rating

plt.figure(figsize=(8,6))
sns.scatterplot(x='Duration',y='Rating',data=df)
plt.xlabel('Duration')
plt.ylabel('Rating')
plt.title('Duration Vs Rating')
plt.yticks(np.arange(4,8,0.5))
plt.show()
```

```
Correlation Of Duration And Rating is -0.07511162035794038
```



```
#Top 10 movies according to rating per year and overall.
overall=df.nlargest(10,'Rating')
overall=overall.reset_index(drop=True)
print("Top 10 Movies Overall:")
overall
```

```
Top 10 Movies Overall:
```

	Name	Year	Duration	Genre	Rating	Votes	Director	Actor 1	Actor 2	Actor 3
0	Abhagin	1938	151	NaN	7.266085	9.555556	Prafulla Roy	Molina Devi	Prithviraj Kapoor	Vijay Kumar
1	Abhilasha	1938	134	NaN	7.266085	9.555556	Zia Sarhadi	Mahendra Thakore	M. Kumar	Bibbo
2	Adhikar	1938	132	NaN	7.266085	9.555556	P.C. Barua	P.C. Barua	Jamuna	Pahadi Sanyal
3	Baazigar	1938	152	NaN	7.266085	9.555556	Mohan Dayaram Bhavnani	K.L. Saigal	Ashok Kumar	Bibbo
4	Baghban	1938	159	Drama	7.266085	9.555556	Abdul Rashid Kardar	Bimla Kumari	B. Nandrekar	Sitara Devi
5	Bahadur Kisan	1938	0	NaN	7.266085	9.555556	Master Bhagwan	Chandrarao	Chandrarao	Hansa Wadkar
6	Ban Ki Chidiya	1938	0	Action	7.266085	9.555556	Jayant Desai	Madhuri	Eddie Billimoria	Ishwarlal

```
top_10_per_year = pd.DataFrame()
for year in df['Year'].unique():
    year_df = df[df['Year'] == year]
    top_10_year = year_df.nlargest(10, 'Rating').sort_values(by='Rating', ascending=False)
    top_10_per_year = top_10_per_year.append(top_10_year)


top_10_per_year = top_10_per_year.reset_index(drop=True)
print("\nTop 10 Movies Per Year:")
top_10_per_year
```

```
<ipython-input-28-54e74d6d9c9c>:5: FutureWarning: The frame.append method is deprecated and will be removed fron
  top_10_per_year = top_10_per_year.append(top_10_year)
<ipython-input-28-54e74d6d9c9c>:5: FutureWarning: The frame.append method is deprecated and will be removed fron
  top_10_per_year = top_10_per_year.append(top_10_year)
<ipython-input-28-54e74d6d9c9c>:5: FutureWarning: The frame.append method is deprecated and will be removed fron
  top_10_per_year = top_10_per_year.append(top_10_year)
<ipython-input-28-54e74d6d9c9c>:5: FutureWarning: The frame.append method is deprecated and will be removed fron
  top_10_per_year = top_10_per_year.append(top_10_year)
<ipython-input-28-54e74d6d9c9c>:5: FutureWarning: The frame.append method is deprecated and will be removed fron
  top_10_per_year = top_10_per_year.append(top_10_year)
<ipython-input-28-54e74d6d9c9c>:5: FutureWarning: The frame.append method is deprecated and will be removed fron
  top_10_per_year = top_10_per_year.append(top_10_year)
<ipython-input-28-54e74d6d9c9c>:5: FutureWarning: The frame.append method is deprecated and will be removed fron
  top_10_per_year = top_10_per_year.append(top_10_year)
<ipython-input-28-54e74d6d9c9c>:5: FutureWarning: The frame.append method is deprecated and will be removed fron
  top_10_per_year = top_10_per_year.append(top_10_year)
<ipython-input-28-54e74d6d9c9c>:5: FutureWarning: The frame.append method is deprecated and will be removed fron
  top_10_per_year = top_10_per_year.append(top_10_year)
<ipython-input-28-54e74d6d9c9c>:5: FutureWarning: The frame.append method is deprecated and will be removed fron
  top_10_per_year = top_10_per_year.append(top_10_year)
<ipython-input-28-54e74d6d9c9c>:5: FutureWarning: The frame.append method is deprecated and will be removed fron
  top_10_per_year = top_10_per_year.append(top_10_year)
<ipython-input-28-54e74d6d9c9c>:5: FutureWarning: The frame.append method is deprecated and will be removed fron
  top_10_per_year = top_10_per_year.append(top_10_year)
<ipython-input-28-54e74d6d9c9c>:5: FutureWarning: The frame.append method is deprecated and will be removed fron
  top_10_per_year = top_10_per_year.append(top_10_year)
<ipython-input-28-54e74d6d9c9c>:5: FutureWarning: The frame.append method is deprecated and will be removed fron
  top_10_per_year = top_10_per_year.append(top_10_year)
<ipython-input-28-54e74d6d9c9c>:5: FutureWarning: The frame.append method is deprecated and will be removed fron
  top_10_per_year = top_10_per_year.append(top_10_year)
<ipython-input-28-54e74d6d9c9c>:5: FutureWarning: The frame.append method is deprecated and will be removed fron
  top_10_per_year = top_10_per_year.append(top_10_year)
<ipython-input-28-54e74d6d9c9c>:5: FutureWarning: The frame.append method is deprecated and will be removed fron
  top_10_per_year = top_10_per_year.append(top_10_year)
<ipython-input-28-54e74d6d9c9c>:5: FutureWarning: The frame.append method is deprecated and will be removed fron
  top_10_per_year = top_10_per_year.append(top_10_year)
<ipython-input-28-54e74d6d9c9c>:5: FutureWarning: The frame.append method is deprecated and will be removed fron
  top_10_per_year = top_10_per_year.append(top_10_year)
<ipython-input-28-54e74d6d9c9c>:5: FutureWarning: The frame.append method is deprecated and will be removed fron
  top_10_per_year = top_10_per_year.append(top_10_year)
<ipython-input-28-54e74d6d9c9c>:5: FutureWarning: The frame.append method is deprecated and will be removed fron
  top_10_per_year = top_10_per_year.append(top_10_year)
<ipython-input-28-54e74d6d9c9c>:5: FutureWarning: The frame.append method is deprecated and will be removed fron
  top_10_per_year = top_10_per_year.append(top_10_year)
<ipython-input-28-54e74d6d9c9c>:5: FutureWarning: The frame.append method is deprecated and will be removed fron
  top_10_per_year = top_10_per_year.append(top_10_year)
<ipython-input-28-54e74d6d9c9c>:5: FutureWarning: The frame.append method is deprecated and will be removed fron
  top_10_per_year = top_10_per_year.append(top_10_year)
<ipython-input-28-54e74d6d9c9c>:5: FutureWarning: The frame.append method is deprecated and will be removed fron
  top_10_per_year = top_10_per_year.append(top_10_year)
<ipython-input-28-54e74d6d9c9c>:5: FutureWarning: The frame.append method is deprecated and will be removed fron
  top_10_per_year = top_10_per_year.append(top_10_year)
<ipython-input-28-54e74d6d9c9c>:5: FutureWarning: The frame.append method is deprecated and will be removed fron
  top_10_per_year = top_10_per_year.append(top_10_year)
<ipython-input-28-54e74d6d9c9c>:5: FutureWarning: The frame.append method is deprecated and will be removed fron
  top_10_per_year = top_10_per_year.append(top_10_year)
```

```
#Number of popular movies released each year.
rat_bool=df['Rating']>=6
vot_bool=df['Votes']>110
pop_df=df[vot_bool & rat_bool]
pop_df
```

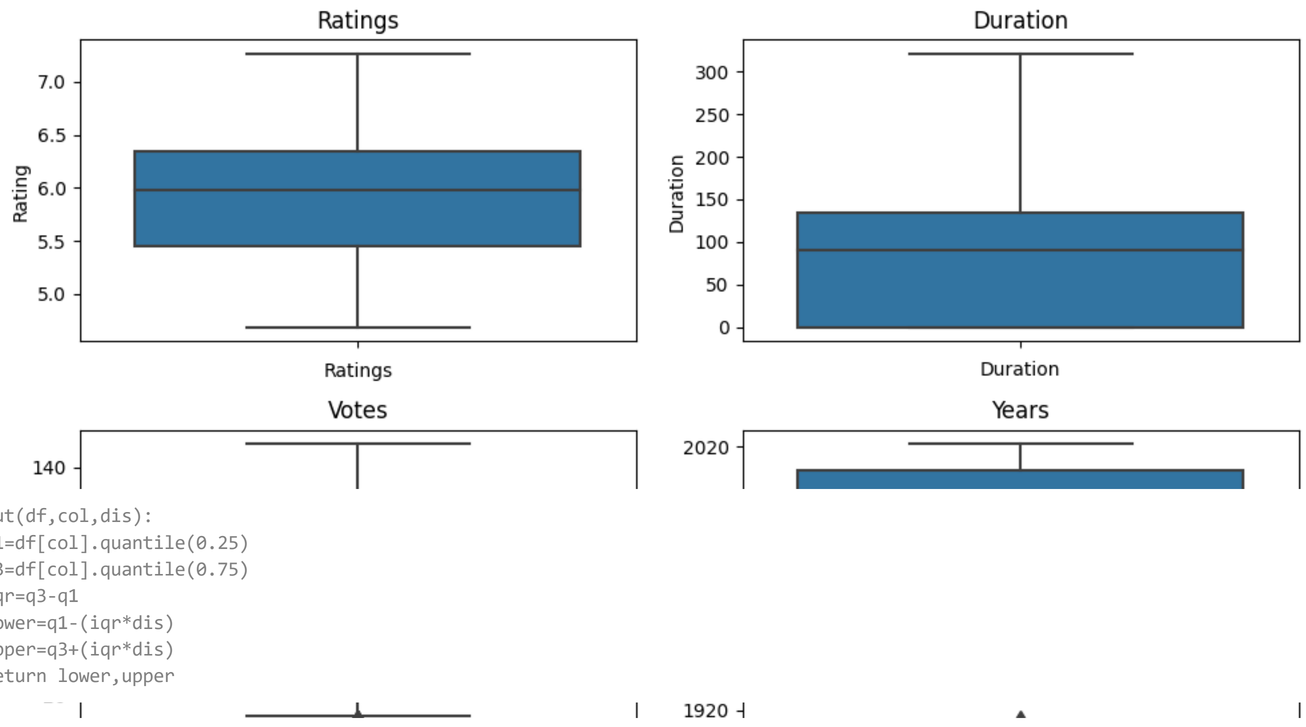
```
                                  Name  Year  Duration          Genre    Rating       Votes          Director        Actor 1         Actor 2          Actor 3
 1   #Gadhvi (He thought he was Gandhi)  2019       109          Drama  6.489797  121.770116     Gaurav Bakshi    Rasika Dugal  Vivek Ghamande    Arvind Jangid
```

```
#ML
```

```
df.dropna(inplace=True)
df.isnull().sum()
```

```
    Name        0
    Year        0
    Duration    0
    Genre       0
    Rating      0
    Votes       0
    Director    0
    Actor 1     0
    Actor 2     0
    Actor 3     0
    dtype: int64
```

```
#df.reset_index()
fig,ax=plt.subplots(nrows=2,ncols=2,figsize=(10,6))

sns.boxplot(data=df,y='Rating',ax=ax[0][0])
ax[0][0].set_title('Ratings')
ax[0][0].set_xlabel('Ratings')

sns.boxplot(data=df,y='Duration',ax=ax[0][1])
ax[0][1].set_title('Duration')
ax[0][1].set_xlabel('Duration')

sns.boxplot(data=df,y='Votes',ax=ax[1][0])
ax[1][0].set_title('Votes')
ax[1][0].set_xlabel('Votes')

sns.boxplot(data=df,y='Year',ax=ax[1][1])
ax[1][1].set_title('Years')
ax[1][1].set_xlabel('Years')

plt.tight_layout()

plt.show()
```



```
def out(df,col,dis):
    q1=df[col].quantile(0.25)
    q3=df[col].quantile(0.75)
    iqr=q3-q1
    lower=q1-(iqr*dis)
    upper=q3+(iqr*dis)
    return lower,upper
```

```
votes_low,votes_up=out(df,'Votes',1.5)
```

```
vote_out_count=(df['Votes'] > votes_up) | (df['Votes'] < votes_low)
```

```
df['Votes'][vote_out_count].count()
```

```
12
```

```
df=df[(df['Votes']>votes_low) & (df['Votes']<votes_up)]
```

```
year_low,year_upper=out(df,'Year',1.5)
```

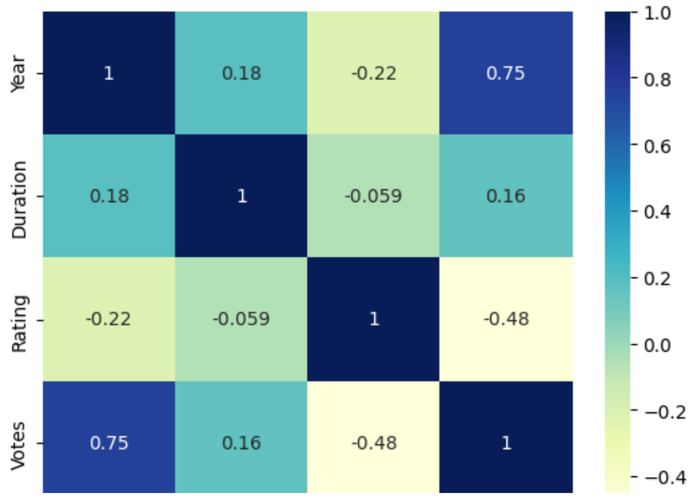
```
year_out_count=(df['Year']>year_upper) | (df['Year']<year_low)
```

```
df['Year'][year_out_count].count()
```

```
1
```

```
sns.heatmap(df.corr(),cmap='YlGnBu',annot=True)
```

```
<ipython-input-41-a7848fed4585>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only
  sns.heatmap(df.corr(),cmap='YlGnBu',annot=True)
<Axes: >
```



```
df=df[(df['Year']>year_low) &(df['Year']<year_upper)]
df.shape
```

```
(13071, 10)
```

Applying ML

```
from sklearn.preprocessing import LabelEncoder
LB=LabelEncoder()
df['Name']=LB.fit_transform(df['Name'])
df['Genre']=LB.fit_transform(df['Genre'])
df['Director']=LB.fit_transform(df['Director'])
df['Actor 1']=LB.fit_transform(df['Actor 1'])
df['Actor 2']=LB.fit_transform(df['Actor 2'])
df['Actor 3']=LB.fit_transform(df['Actor 3'])
```

```
from sklearn.linear_model import LinearRegression
LR=LinearRegression()
```

```
from sklearn.model_selection import train_test_split
x=df.drop('Rating',axis=1)
y=df['Rating']
```

```
train_x,test_x,train_y,test_y=train_test_split(x,y,test_size=0.3,random_state=42)
```