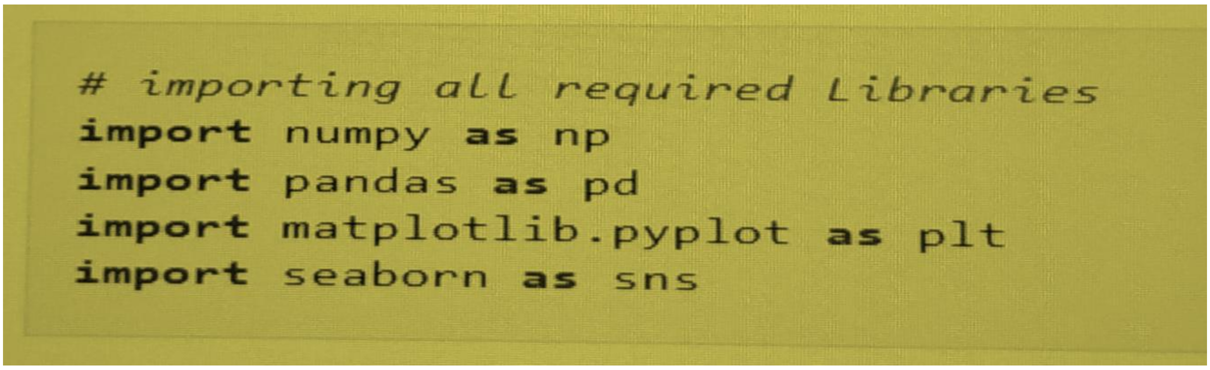
Exploratory Data Analysis on Top 250 TV Shows

Step – 1: Import all required Libraries

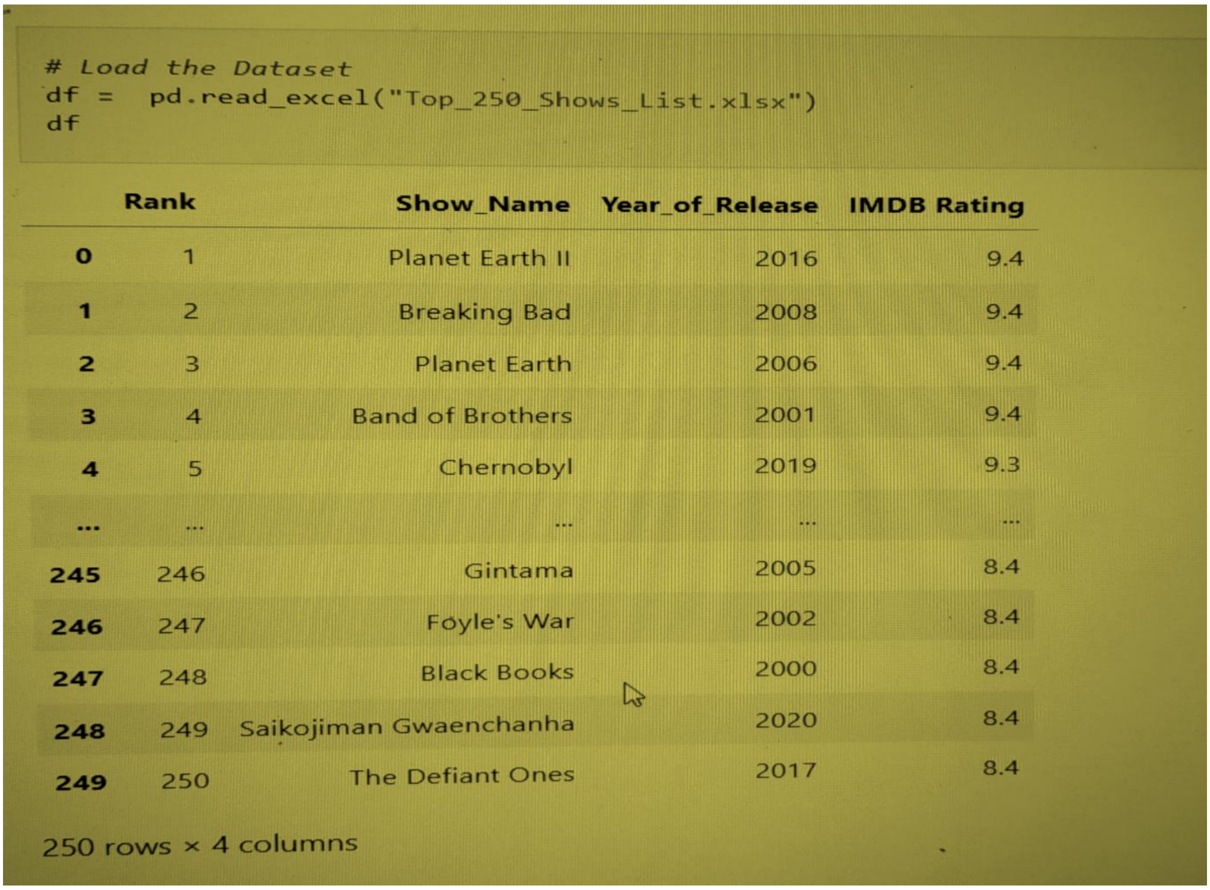
To Perform EDA on Dataset, import all libraries which are required for our data analysis, such as Data Loading, Statistical analysis, Visualizations.



```
# importing all required Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Step – 2: Load the Dataset

After importing all required Libraries, we read the data from Excel file into the DataFrame.



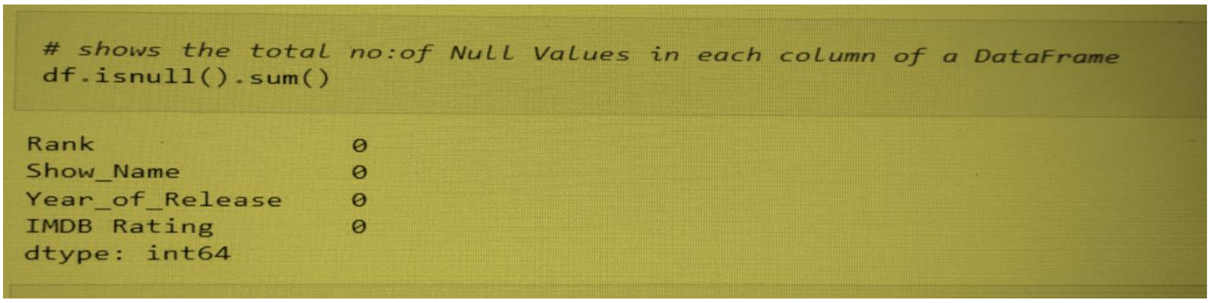
```
# Load the Dataset
df =  pd.read_excel("Top_250_Shows_List.xlsx")
df
```

```
     Rank               Show_Name  Year_of_Release  IMDB Rating
0       1          Planet Earth II            2016          9.4
1       2             Breaking Bad            2008          9.4
2       3             Planet Earth            2006          9.4
3       4        Band of Brothers            2001          9.4
4       5                Chernobyl            2019          9.3
...   ...                      ...             ...          ...
245   246                  Gintama            2005          8.4
246   247              Foyle's War            2002          8.4
247   248              Black Books            2000          8.4
248   249  Saikojiman Gwaenchanha            2020          8.4
249   250          The Defiant Ones            2017          8.4

250 rows × 4 columns
```

Step – 3: Data Cleaning

Now, we can use isnull () method for checking the null values in a DataFrame and we check for number of Null Values in each column for that we will use isnull (). sum () function.

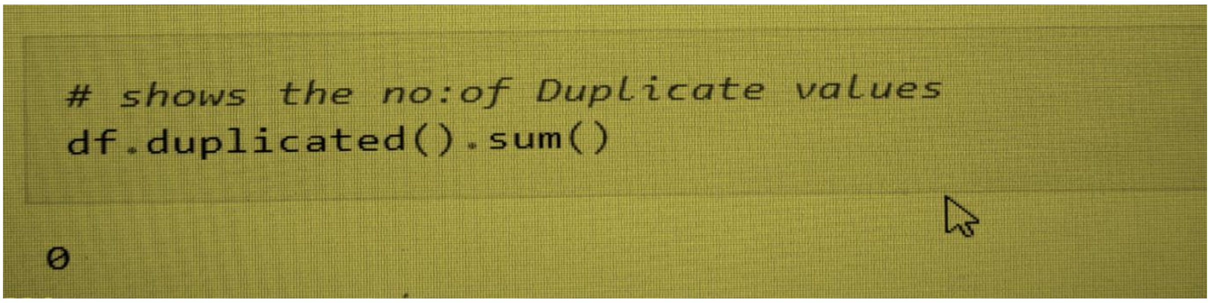


```
# shows the total no:of Null Values in each column of a DataFrame
df.isnull().sum()

Rank                 0
Show_Name            0
Year_of_Release      0
IMDB Rating          0
dtype: int64
```

There are no Null Values in each column shows zero null values.

After checking the Null Values, number of duplicate records can be identified by using duplicated (). Sum ().



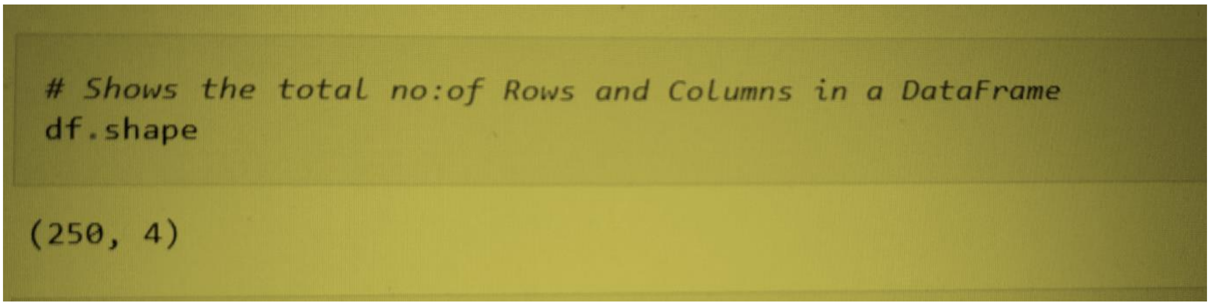
```
# shows the no:of Duplicate values
df.duplicated().sum()

0
```

There are no Duplicate Records in a DataFrame.

Step – 4: Data Exploration

For checking the number of Rows and Columns in a DataFrame, we will use shape attribute.

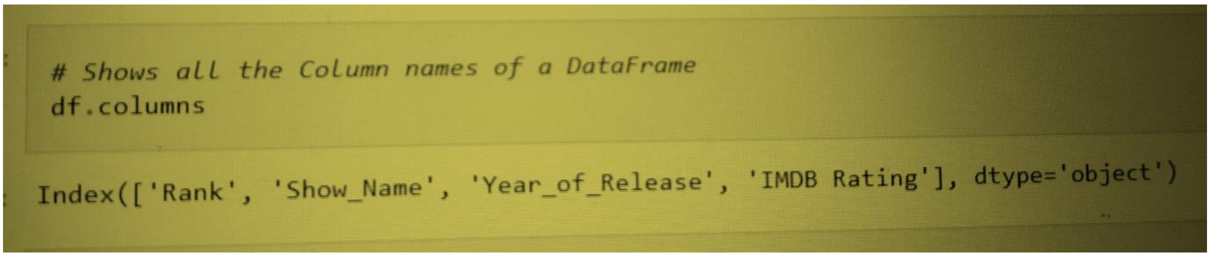


```
# Shows the total no:of Rows and Columns in a DataFrame
df.shape

(250, 4)
```

DataFrame has 250 Records and 4 Columns.

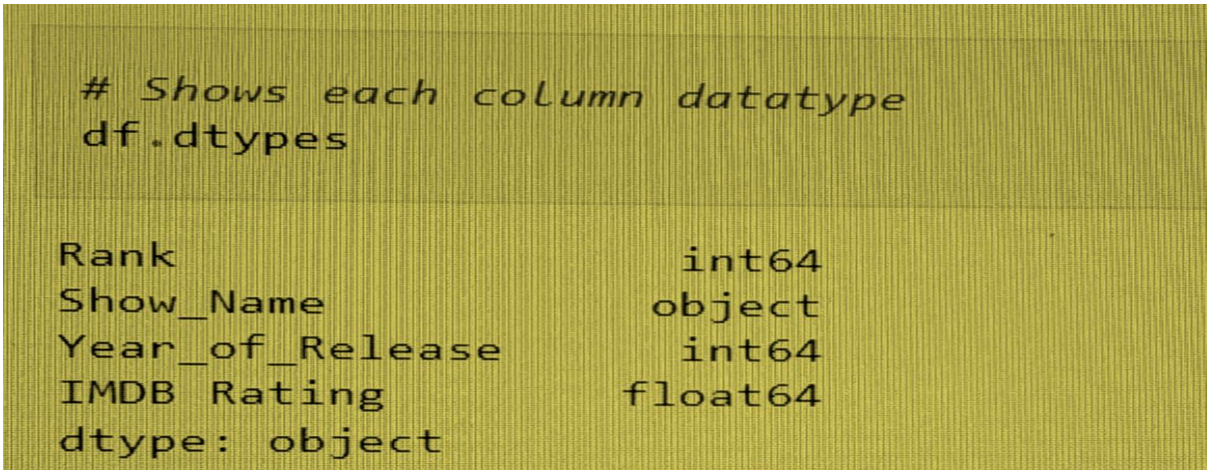
We can see all the Column names of a DataFrame by using columns attribute.



```
# Shows all the Column names of a DataFrame
df.columns
```

```
Index(['Rank', 'Show_Name', 'Year_of_Release', 'IMDB Rating'], dtype='object')
```

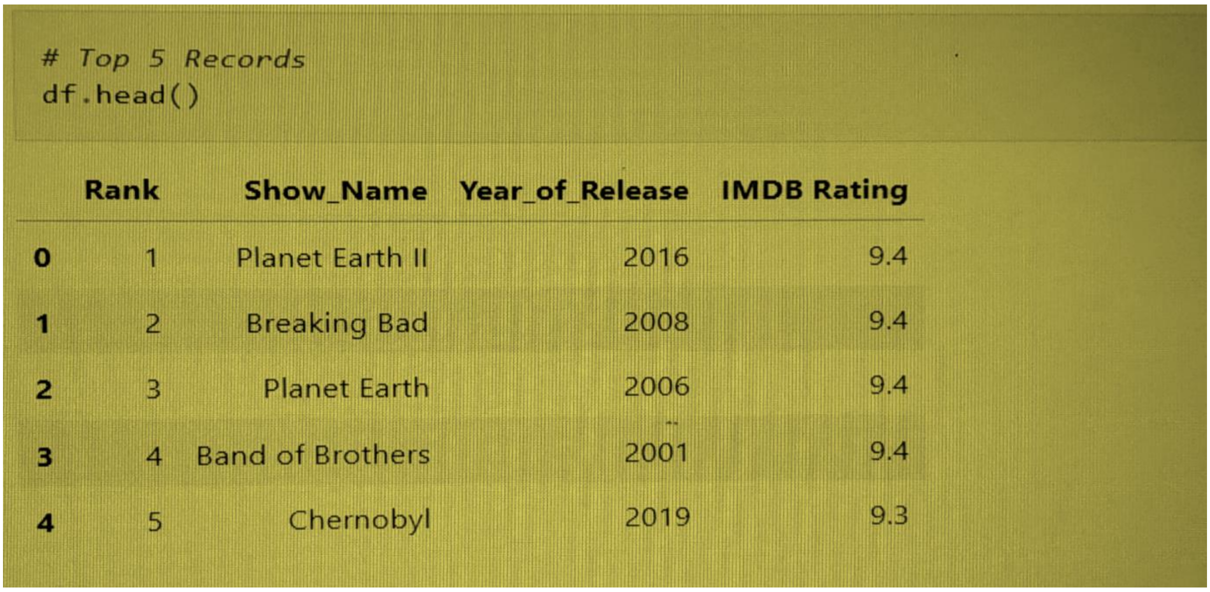
We can also know the datatype of each column in a DataFrame with the help of dtypes attribute.



```
# Shows each column datatype
df.dtypes
```

```
Rank                 int64
Show_Name           object
Year_of_Release      int64
IMDB Rating        float64
dtype: object
```

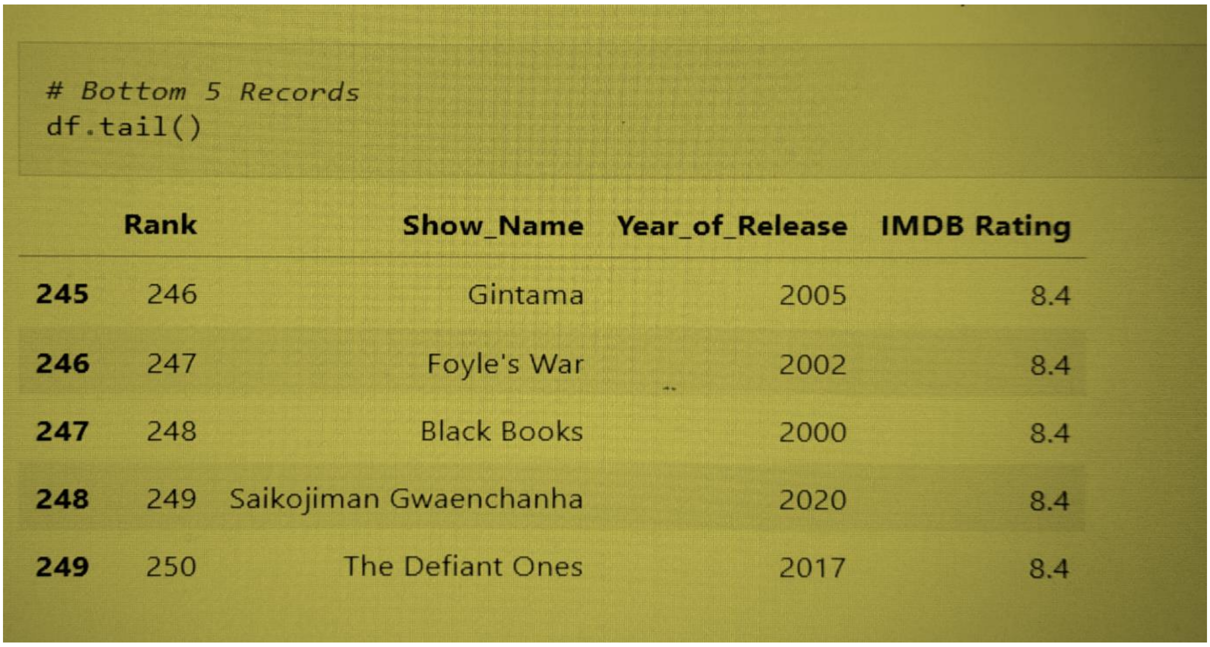
If we want to see the Top Five Records from the DataFrame, we will use head () function.



```
# Top 5 Records
df.head()
```

```
   Rank         Show_Name  Year_of_Release  IMDB Rating
0     1   Planet Earth II             2016          9.4
1     2      Breaking Bad             2008          9.4
2     3      Planet Earth             2006          9.4
3     4  Band of Brothers             2001          9.4
4     5         Chernobyl             2019          9.3
```

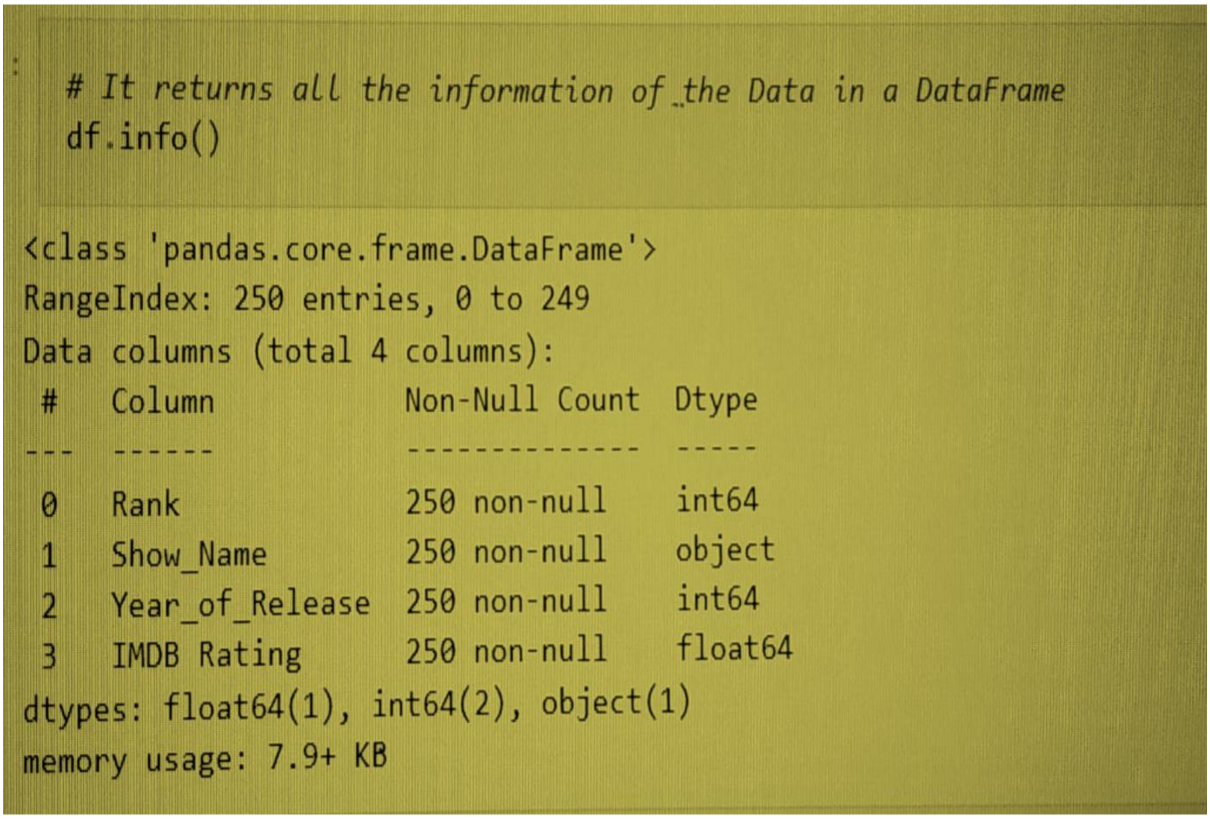
If we want to see the Bottom 5 Records from the DataFrame, we will use tail () method.



```
# Bottom 5 Records
df.tail()
```

	Rank	Show_Name	Year_of_Release	IMDB Rating
245	246	Gintama	2005	8.4
246	247	Foyle's War	2002	8.4
247	248	Black Books	2000	8.4
248	249	Saikojiman Gwaenchanha	2020	8.4
249	250	The Defiant Ones	2017	8.4

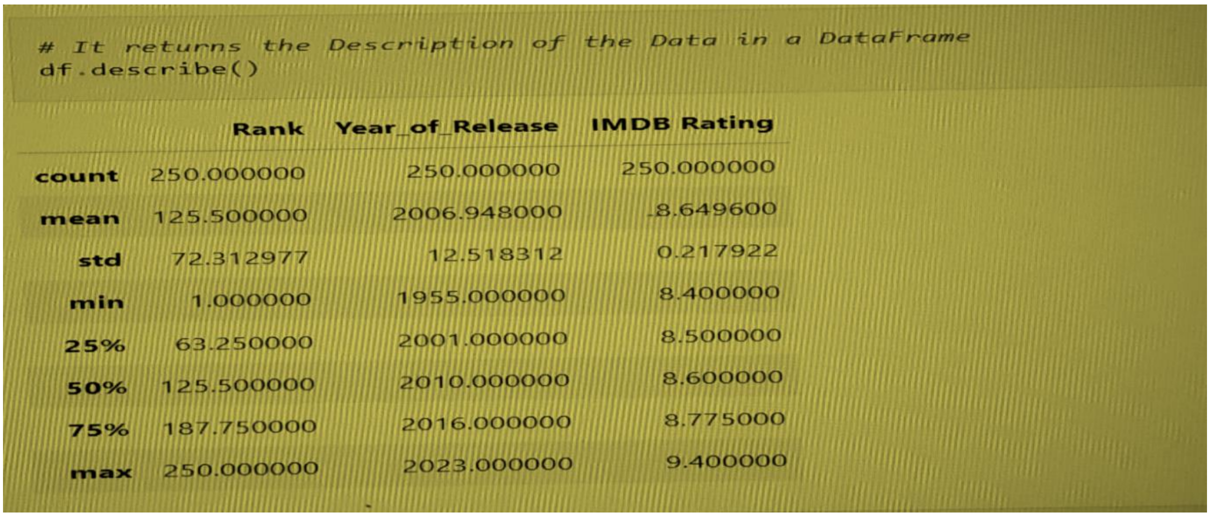
If we want to know all the information it means number of Columns, data type of each column, number of records in each column having any Null Values or not, dtypes, and memory usage about the DataFrame, we can use info () method.



```
# It returns all the information of the Data in a DataFrame
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 250 entries, 0 to 249
Data columns (total 4 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Rank             250 non-null    int64  
 1   Show_Name        250 non-null    object 
 2   Year_of_Release  250 non-null    int64  
 3   IMDB Rating      250 non-null    float64
dtypes: float64(1), int64(2), object(1)
memory usage: 7.9+ KB
```

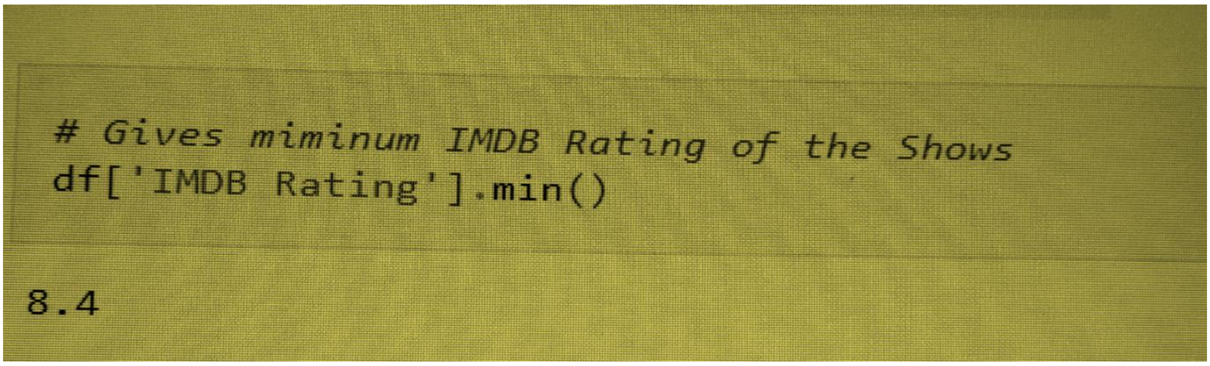
We can see mean, median, Percentiles, minimum values and maximum values of all the Numerical variables by using describe () method.



```
# It returns the Description of the Data in a DataFrame
df.describe()
```

```
             Rank  Year_of_Release  IMDB Rating
count  250.000000       250.000000   250.000000
mean   125.500000      2006.948000     8.649600
std     72.312977        12.518312     0.217922
min      1.000000      1955.000000     8.400000
25%     63.250000      2001.000000     8.500000
50%    125.500000      2010.000000     8.600000
75%    187.750000      2016.000000     8.775000
max    250.000000      2023.000000     9.400000
```

If we want to see the minimum value of IMDB Rating, we will use min () function.

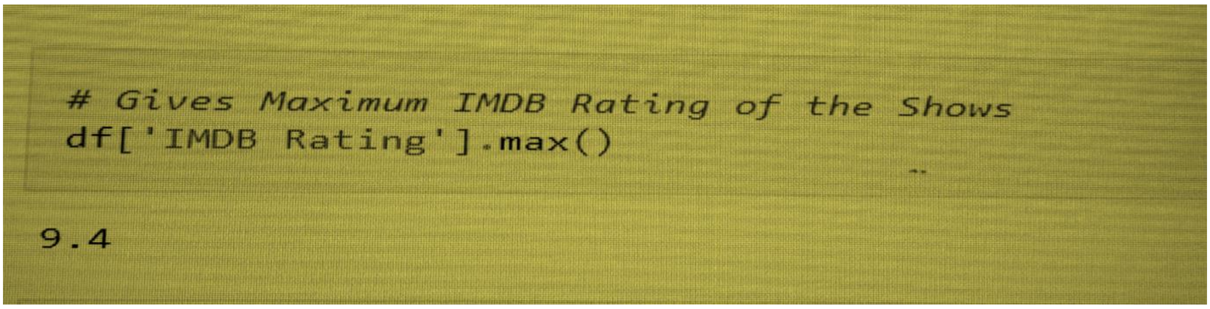


```
# Gives miminum IMDB Rating of the Shows
df['IMDB Rating'].min()
```

```
8.4
```

8.4 is the Lowest IMDB Rating of the Top 250 Shows.

If we want to see the maximum value in IMDB Rating column, we will use max () function.

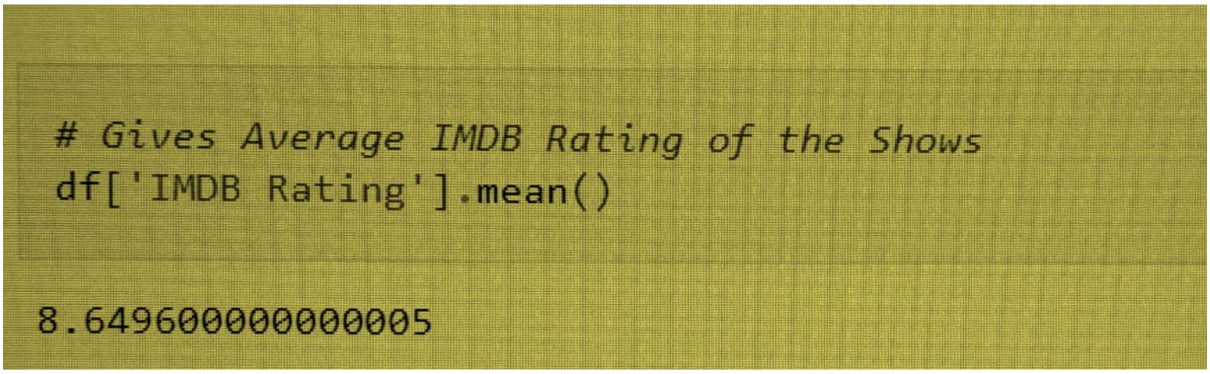


```
# Gives Maximum IMDB Rating of the Shows
df['IMDB Rating'].max()
```

```
9.4
```

9.4 is the Highest IMDB Rating of the Top 250 Shows.

If we want to see the Average value of IMDB Rating in IMDB Rating column, we will use mean () function.

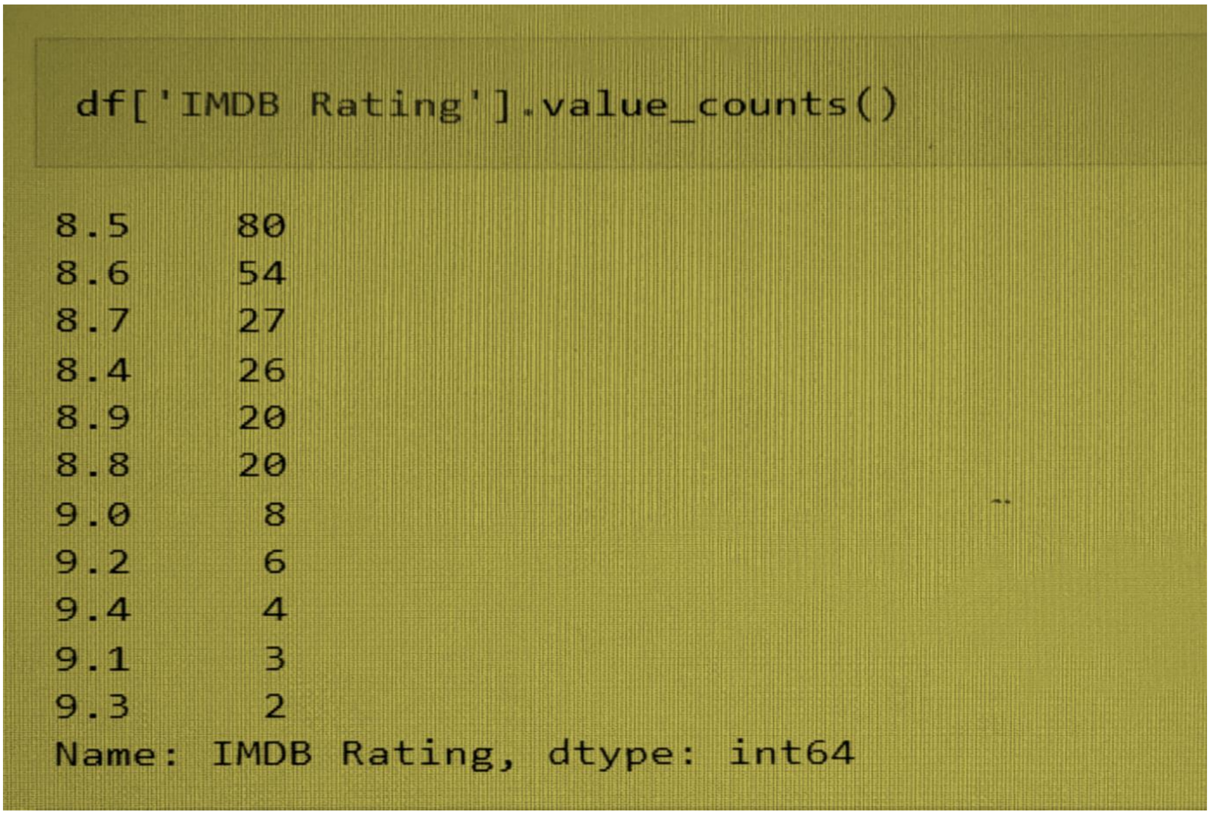


```
# Gives Average IMDB Rating of the Shows
df['IMDB Rating'].mean()
```

```
8.649600000000005
```

Average IMDB Rating of Top 250 Shows is 8.64.

We can return a Series of IMDB Rating Unique Values Count by using value_counts () function.



```
df['IMDB Rating'].value_counts()
```

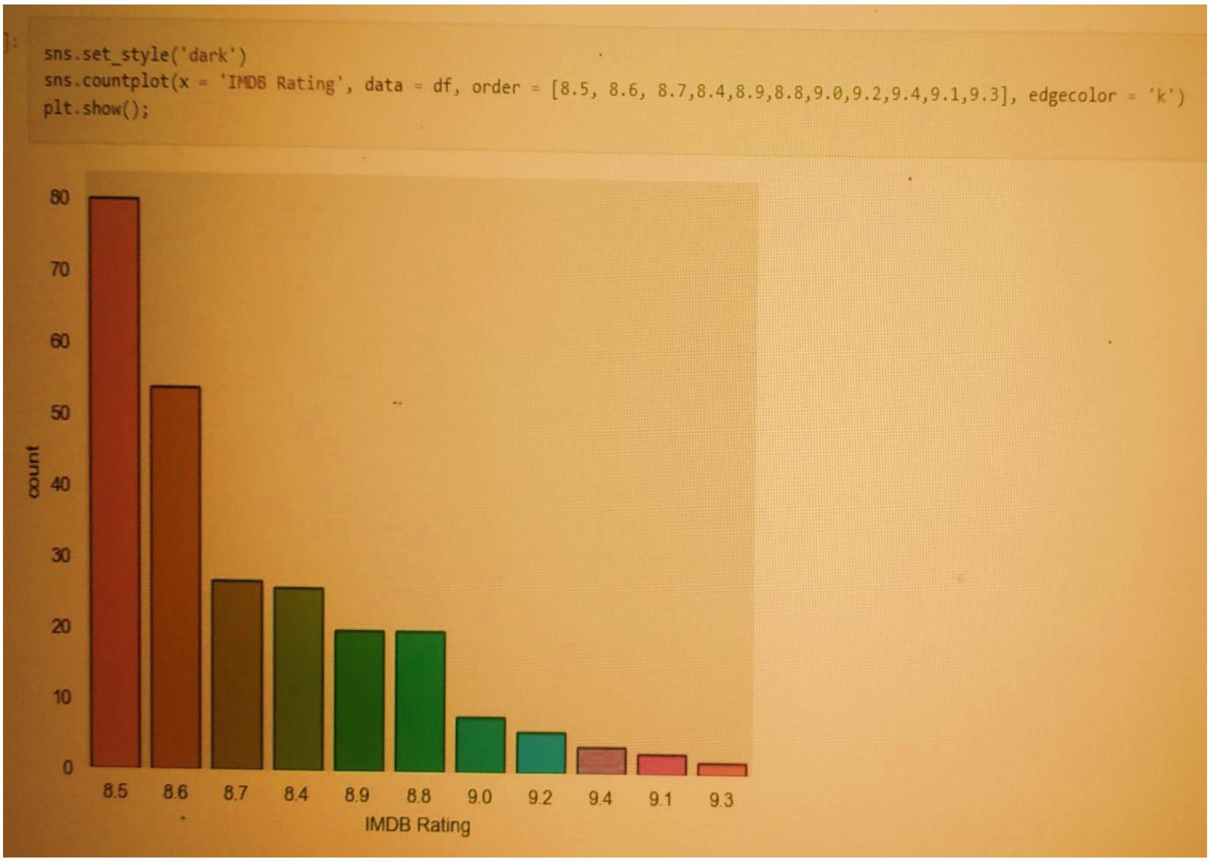
```
8.5    80
8.6    54
8.7    27
8.4    26
8.9    20
8.8    20
9.0     8
9.2     6
9.4     4
9.1     3
9.3     2
Name: IMDB Rating, dtype: int64
```

8.5 IMDB Rating Shows are 80 in number and More Shows has got 8.5 IMDB Rating. Secondly, the shows got 8.6 IMDB Rating are 54 in number. Third, 8.7 IMDB Rating Shows are 27 in number.

Step – 5: Data Visualization using Matplotlib and Seaborn Libraries

Univariate Analysis:

If we want to know How many numbers of TV Shows got same rating, we need to plot the count plot.



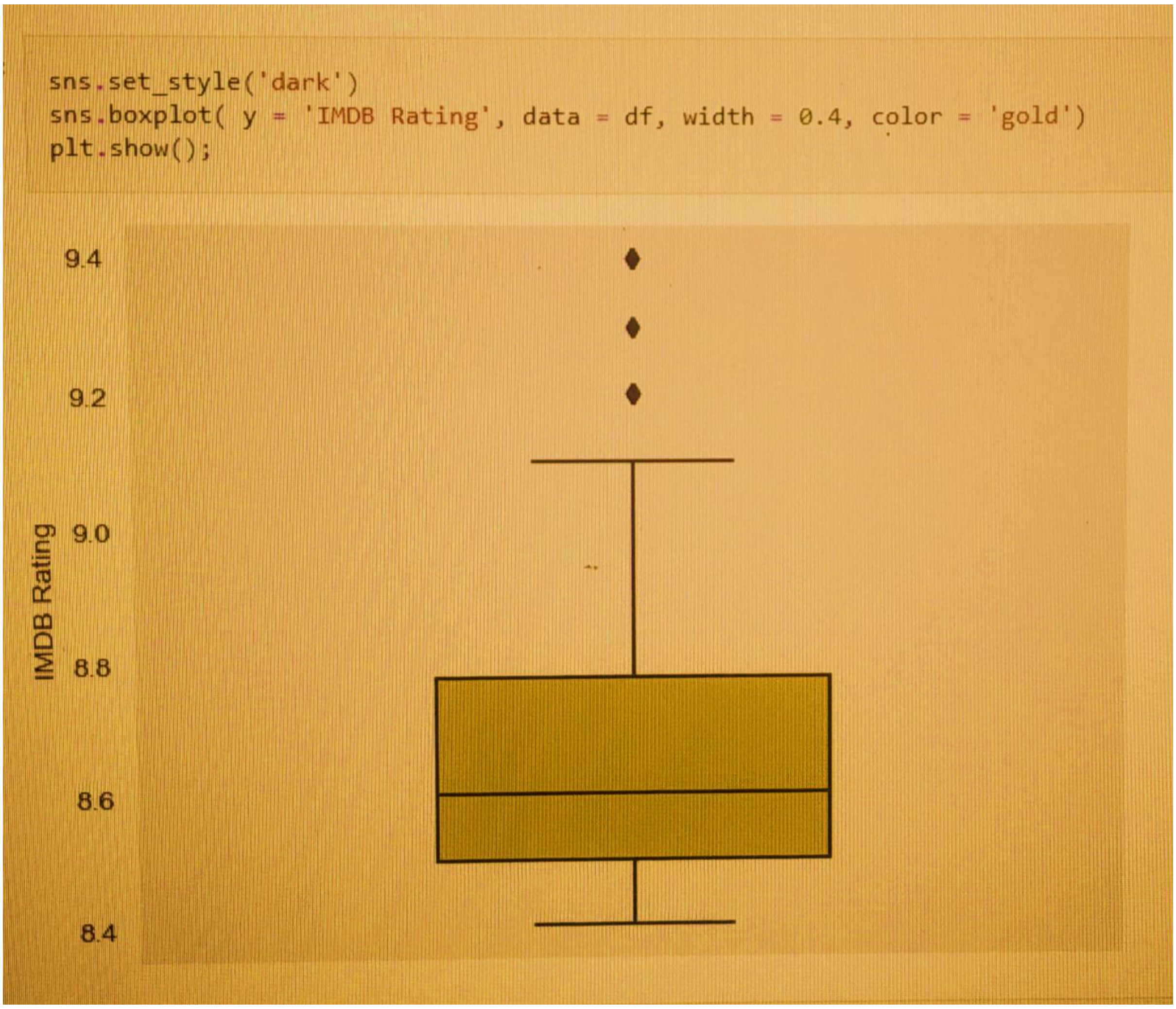
```
sns.set_style('dark')
sns.countplot(x = 'IMDB Rating', data = df, order = [8.5, 8.6, 8.7,8.4,8.9,8.8,9.0,9.2,9.4,9.1,9.3], edgecolor = 'k')
plt.show();
```

From the figure, we get some insights.

Insights:

1. The Shows which are rated as 8.5 is more in number as compared to other rating Shows that is 80 TV Shows are rated as 8.6.
2. 8.8 and 8.9 rating TV Shows are equal in number that is 20 TV Shows rated as 8.8 as well as 20 TV Shows rated as 8.9.
3. 9.3 Rating TV Shows are less in number as compared to all other rating TV Shows. Only, Two TV Shows are rated as 9.3.
4. Only, four TV Shows rated as 9.4 which is the highest IMDB Rating in the Top 250 TV Shows.

If we want to know most of the IMDB Ratings of TV Shows are in between some Ratings, plot a box plot.



```
sns.set_style('dark')
sns.boxplot( y = 'IMDB Rating', data = df, width = 0.4, color = 'gold')
plt.show();
```

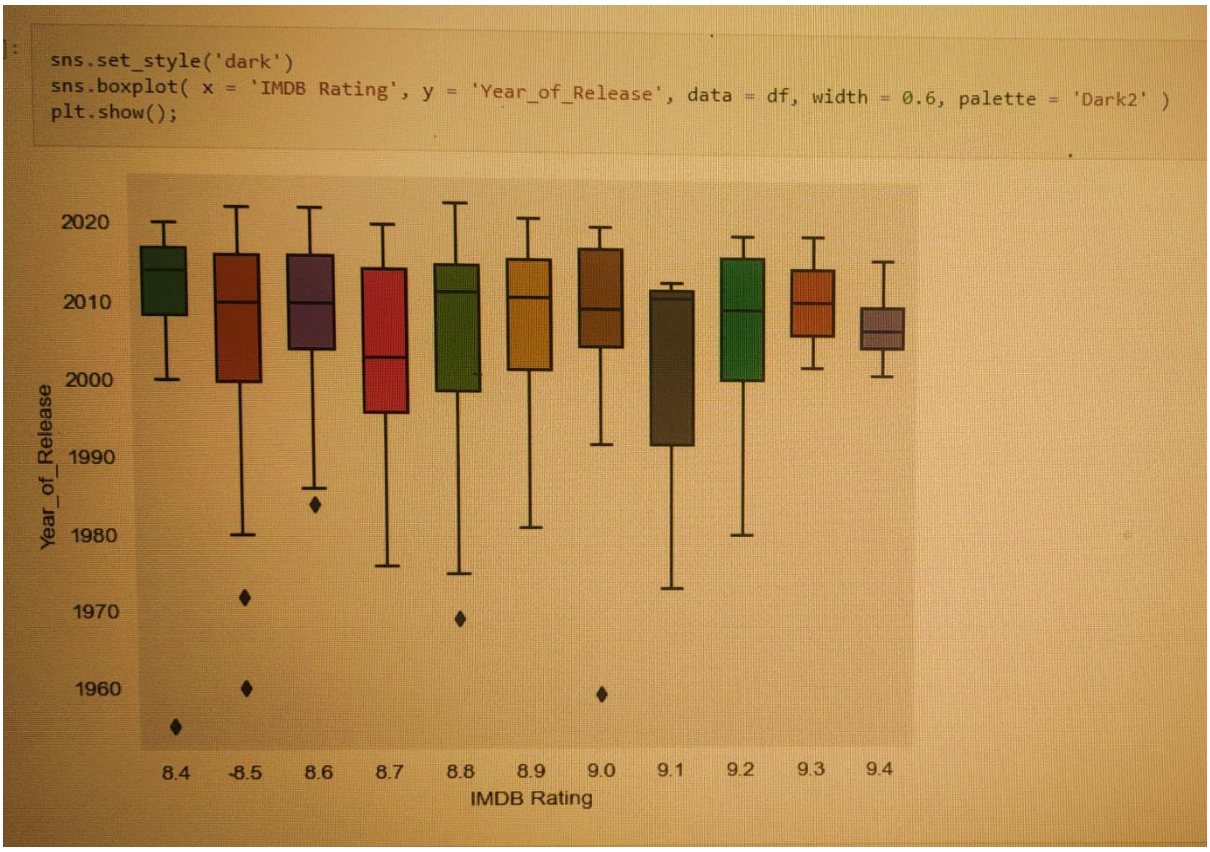
If we observe the box plot for IMDB Rating of Top 250 TV Shows, we get some information.

Insights:

1. TV Shows got Minimum IMDB Rating as 8.4.
2. 25% to 75% of the IMDB Ratings for the Top 250 TV Shows are in between 8.5 to 8.8.
3. 9.2 and Above 9.2 IMDB Rating of TV Shows are considered as Outliers because these ratings are far from the mean and a smaller number of TV shows got above 9.0 IMDB Rating.
4. TV Shows got Maximum IMDB Rating as 9.4.
5. TV Shows got Average IMDB Rating as 8.6.

Bivariate Analysis:

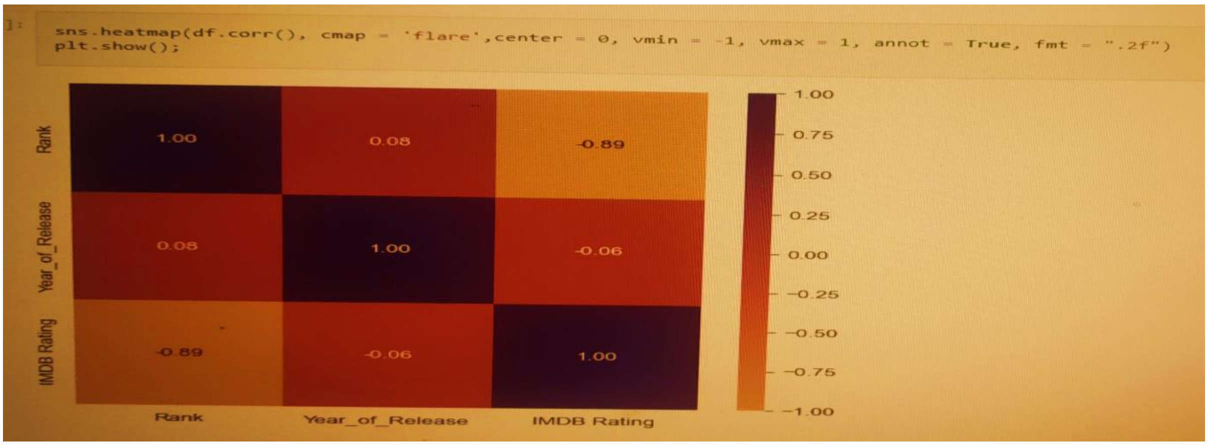
If we want to see the IMDB Ratings with respect to Year of Released TV Shows, we plot box plot and take IMDB Rating Variable in X – Axis and year of Released variable in Y – Axis.



```
sns.set_style('dark')
sns.boxplot( x = 'IMDB Rating', y = 'Year_of_Release', data = df, width = 0.6, palette = 'Dark2' )
plt.show();
```

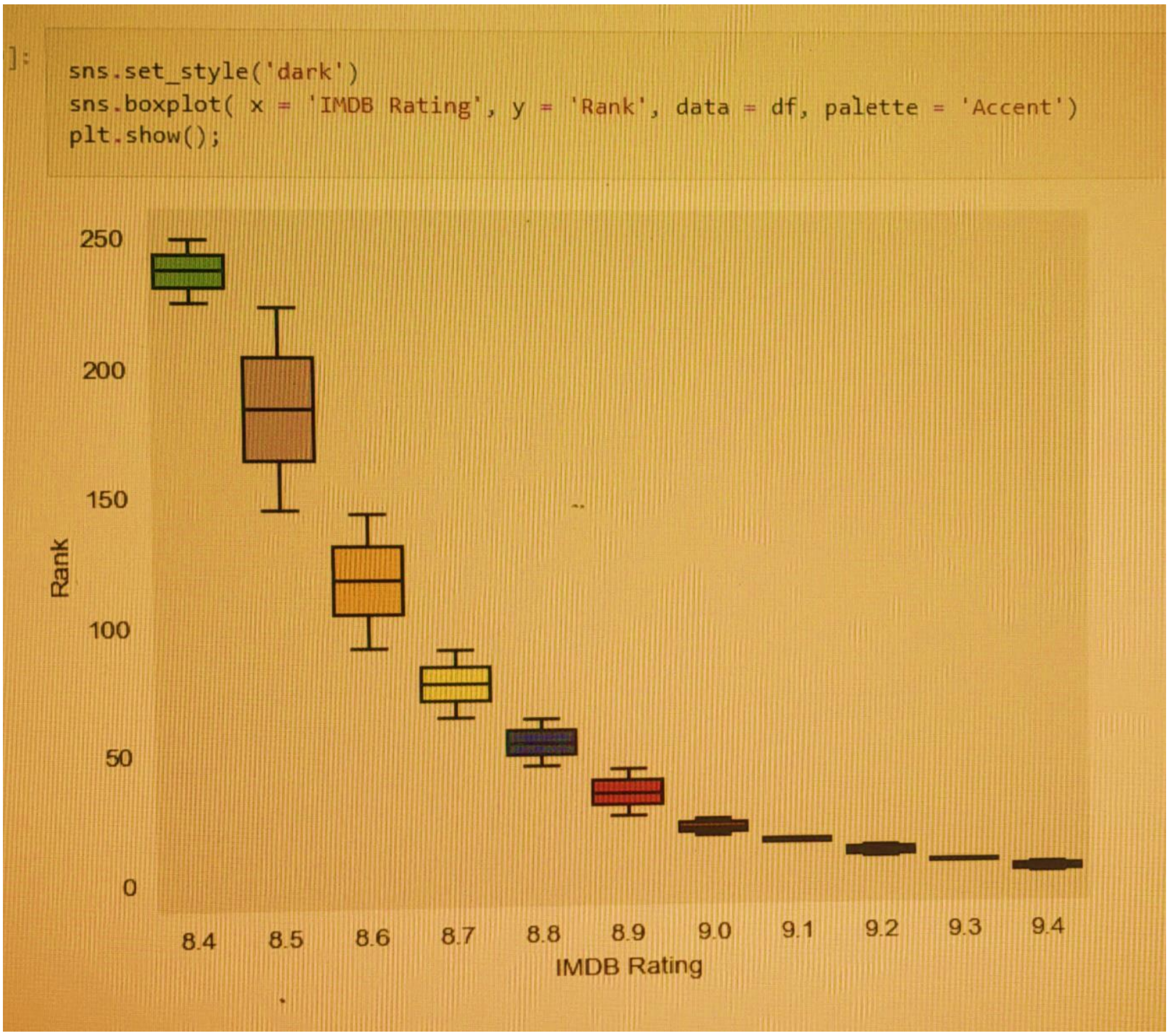
Insight:

If we observe the above figure, we can say that, from last 30 years the TV Shows have been getting Good IMDB Ratings.



```
sns.heatmap(df.corr(), cmap = 'flare',center = 0, vmin = -1, vmax = 1, annot = True, fmt = ".2f")
plt.show();
```

If we want to see the Rank with respect to IMDB Rating, we plot box plot and take IMDB Rating Variable in X – Axis and Rank variable in Y – Axis..



```
sns.set_style('dark')
sns.boxplot( x = 'IMDB Rating', y = 'Rank', data = df, palette = 'Accent')
plt.show();
```

After seeing the above figure, we can strongly say that, if Higher the IMDB Rating Better the Rank.