1. **DATA SET –**

Game of Thrones Dataset –

dataset of a famous TV show

2. URL: https://www.kaggle.com/bakar31/game-of-thronesgot

3. No of Instances – 73

No of Attributes – 10

```
In [6]:
1 #instances and attributes
2 no_of_instances=list(df.shape)[0]
3 no_of_cols=list(df.shape)[1]
```

```
In [54]:
1 no_of_instances,no_of_cols

Out[54]: (73, 10)
```

Datatype –

```
In [8]:
1 #attribute types
2 # the first 3 are integer type, next 5 are object type(string) and next 2 are float
3 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 73 entries, 0 to 72
Data columns (total 10 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   No. overall            73 non-null     int64
 1   No. in season          73 non-null     int64
 2   Season                 73 non-null     int64
 3   Title                  73 non-null     object
 4   Directed by            73 non-null     object
 5   Written by             73 non-null     object
 6   Novel(s) adapted       73 non-null     object
 7   Original air date      61 non-null     object
 8   U.S. viewers(millions) 70 non-null     float64
 9   Imdb rating            73 non-null     float64
dtypes: float64(2), int64(3), object(5)
memory usage: 5.8+ KB
```

Missing Values –

```
In [9]:
1
2 #null values
3 df.isnull().sum()
```

```
Out[9]: No. overall              0
        No. in season            0
        Season                   0
        Title                    0
        Directed by              0
        Written by               0
        Novel(s) adapted         0
        Original air date       12
        U.S. viewers(millions)   3
        Imdb rating              0
        dtype: int64
```

4. Language Chosen for Implementation – **Python**

5. a)

Mean, Median (50%), Mode, Std, variance –

```
In [55]:
1 df.describe()
2
```

Out[55]:

	No. overall	No. in season	Season	U.S. viewers(millions)	Imdb rating
count	59.000000	59.000000	59.000000	59.000000	59.000000
mean	36.355932	5.203390	4.135593	6.422203	8.832203
std	21.300609	2.857414	2.177135	2.858102	1.012413
min	1.000000	1.000000	1.000000	2.200000	4.000000
25%	19.500000	3.000000	2.000000	4.050000	8.700000
50%	35.000000	5.000000	4.000000	6.590000	8.900000
75%	53.500000	8.000000	6.000000	7.810000	9.400000
max	73.000000	10.000000	8.000000	13.610000	9.900000

```
In [11]:
1 df[['Title','Directed by','Written by','Novel(s) adapted','Original air date']].describe()
```

Out[11]:

	Title	Directed by	Written by	Novel(s) adapted	Original air date
count	73	73	73	73	61
unique	73	20	5	6	61
top	"Breaker of Chains"	David Nutter	David Benioff & D. B. Weiss	A Storm of Swords	3-May-15
freq	1	9	51	20	1

```
In [43]:  1  df.var()

Out[43]: No. overall                453.715956
         No. in season                8.164816
         Season                       4.739918
         U.S. viewers(millions)       8.168745
         Imdb rating                  1.024980
         dtype: float64
```

b) Boxplot, Scatter Plot

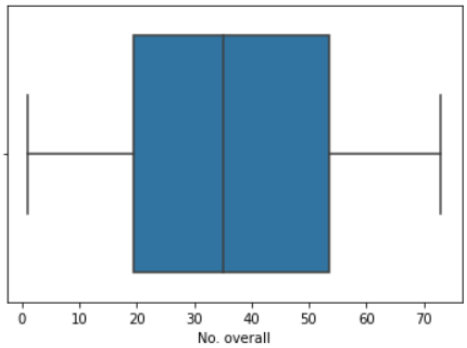
Boxplot –

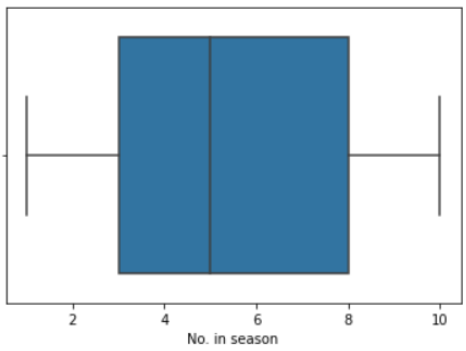
Boxplot visualizes the distribution of each column and shows if outliers are present.

Scatterplot –

Scatterplot represents the scattered distribution of one variable on another variable.

```
In [63]:  1 import seaborn as sns
          2 import matplotlib.pyplot as plt
          3 for i in ['No. overall','No. in season','Season','U.S. viewers(millions)','Imdb rating']:
          4   sns.boxplot(df[i])
          5   plt.show()
```

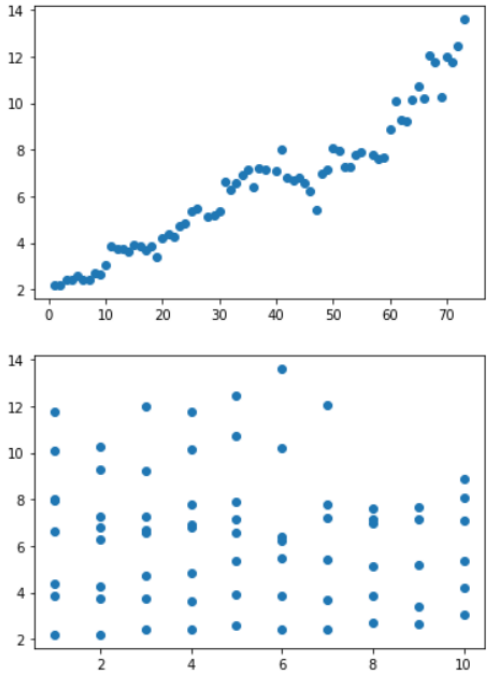




In [13]:

```
for i in ['No. overall','No. in season','Season','U.S. viewers(millions)','Imdb rating']:
  plt.scatter(df[i],df['U.S. viewers(millions)'])
  plt.show()

```



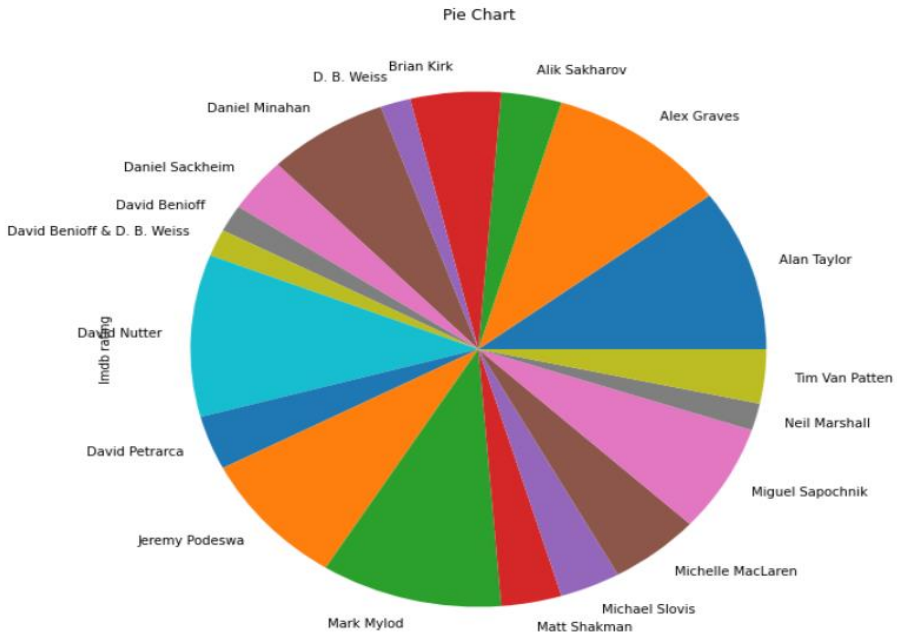
C) Any two Visualisation techniques –

Pie chart –

The directors of each episode of the show are grouped to visualize the share of each directors in the show.

In [50]:

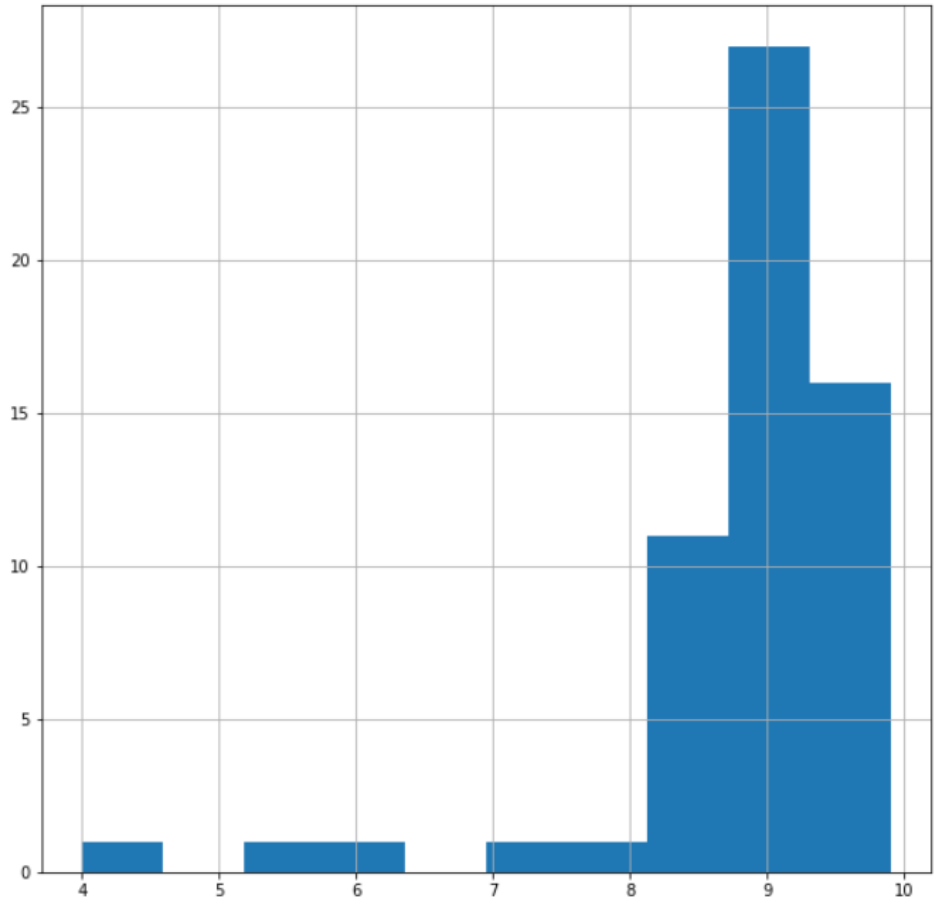
```
# visualization Technique
plt.figure(figsize=(10,10))
grouped=df.groupby('Directed by').count()['Imdb rating']
grouped.plot(kind='pie')
plt.title('Pie Chart')
plt.show()
```



Histogram –

Histogram is drawn for the IMDB ratings column so that , the range an the no. of values between the range are visualized.

```
In [53]:
1 plt.figure(figsize=(10,10))
2 df['Imdb rating'].hist()
3 plt.show()
```



6. Handling Missing Values

- Original air date and U.S. viewers(millions) consists of missing values. Since the 'Original air date' is categorical we added a new category for the missing values in the column as 'missing'.

For the 'U.S. viewers(millions)' the column is numerical, so the missing values are filled with median.

In [19]:
```
df.isnull().sum()
```

Out[19]:
```
No. overall               0
No. in season             0
Season                    0
Title                     0
Directed by               0
Written by                0
Novel(s) adapted          0
Original air date        12
U.S. viewers(millions)    3
Imdb rating               0
dtype: int64
```

In [20]:
```
df['Original air date']=df['Original air date'].fillna('missing')
df['U.S. viewers(millions)']=df['U.S. viewers(millions)'].fillna(df['U.S. viewers(millions)'].median())
```

In [21]:
```
df.isnull().sum()
```

Out[21]:
```
No. overall               0
No. in season             0
Season                    0
Title                     0
Directed by               0
Written by                0
Novel(s) adapted          0
Original air date         0
U.S. viewers(millions)    0
Imdb rating               0
dtype: int64
```

7. Normalization technique

- The normalization technique used is Standardization using Standard Scaler

- subtracting each value with mean and dividing by standard deviation.

In [36]:
```
df1.head()
```

Out[36]:

	No. overall	No. in season	Season	U.S. viewers(millions)	Imdb rating
0	1	1	1	2.22	9.1
1	2	2	1	2.20	8.8
2	3	3	1	2.44	8.7
3	4	4	1	2.45	8.8
4	5	5	1	2.58	9.1

In [22]:
```
from sklearn.preprocessing import StandardScaler

scale=StandardScaler()
df1=df[['No. overall','No. in season','Season','U.S. viewers(millions)','Imdb rating']]
df1=scale.fit_transform(df1)
```

In [23]:
```
normalized_df=pd.DataFrame(df1,columns=['No. overall','No. in season','Season','U.S. viewers(millions)','Imdb rating'])
```

In [24]:
```
normalized_df.head()
```

Out[24]:

	No. overall	No. in season	Season	U.S. viewers(millions)	Imdb rating
0	-1.708484	-1.495765	-1.472543	-1.515978	0.282259
1	-1.661026	-1.138932	-1.472543	-1.523118	-0.034740
2	-1.613569	-0.782099	-1.472543	-1.437434	-0.140406
3	-1.566111	-0.425266	-1.472543	-1.433864	-0.034740
4	-1.518653	-0.068434	-1.472543	-1.387452	0.282259