5) Implement K-means 2 Clustering on a proper dataset of your choice

K-Means Clustering : Perform clustering for the crime data and identify the number of clusters formed and draw inferences. Refer to crime_data.csv dataset.

In [1]:
```
import pandas as pd                  # for Data Manipulation
import matplotlib.pyplot as plt      # for Visualization
import numpy as np                   #for Mathematical calculations
import seaborn as sns                #for Advanced visualizations

crime = pd.read_csv("crime_data.csv")
```

In [2]:
```
crime.head()
```

Out[2]:

	Unnamed: 0	Murder	Assault	UrbanPop	Rape
0	Alabama	13.2	236	58	21.2
1	Alaska	10.0	263	48	44.5
2	Arizona	8.1	294	80	31.0
3	Arkansas	8.8	190	50	19.5
4	California	9.0	276	91	40.6

In [3]:
```
# We see the columns in the dataset
crime['State'] = crime.iloc[:,0]
crime = crime.iloc[:, [5,1,2,3,4]]
```

In [4]:
```
crime.head()
```

Out[4]:

	State	Murder	Assault	UrbanPop	Rape
0	Alabama	13.2	236	58	21.2
1	Alaska	10.0	263	48	44.5
2	Arizona	8.1	294	80	31.0
3	Arkansas	8.8	190	50	19.5
4	California	9.0	276	91	40.6

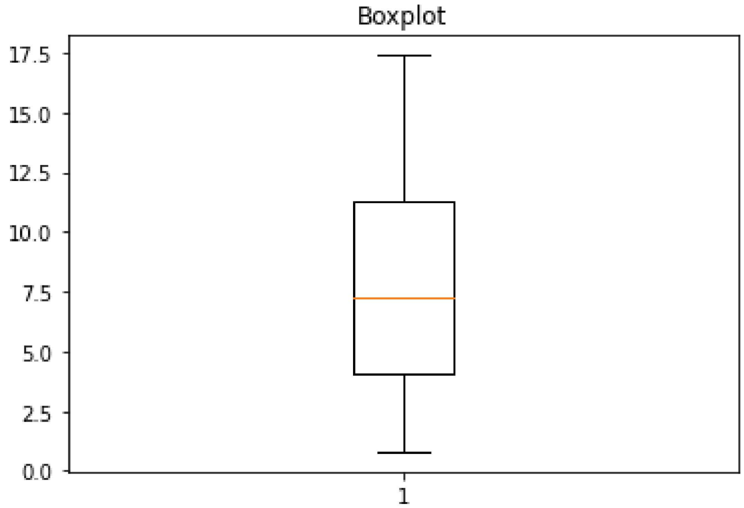
In [5]:
```
# As a part of the Data cleansing we check the data for any missing/ na values
crime.isna().sum()
```

Out[5]:
```
State       0
Murder      0
Assault     0
UrbanPop    0
Rape        0
dtype: int64
```

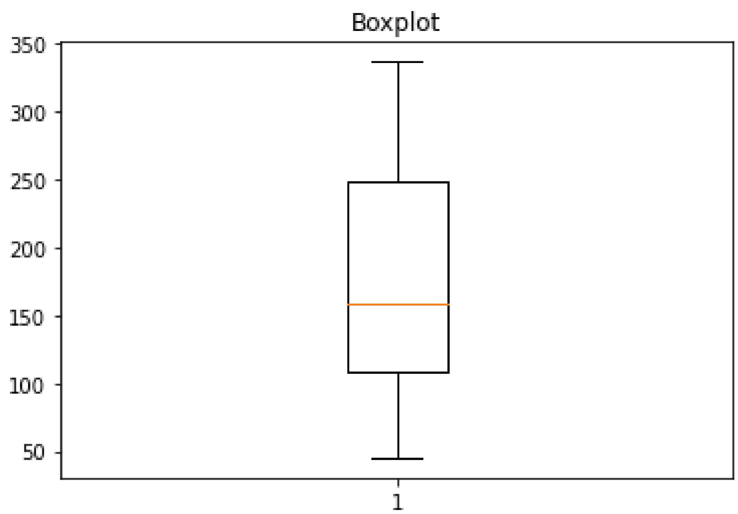
In [6]:
```
# Additionally we check the data for any duplicate values, now this can be an optional
crime1 = crime.duplicated()
sum(crime1)
```

Out[6]:
```
0
```

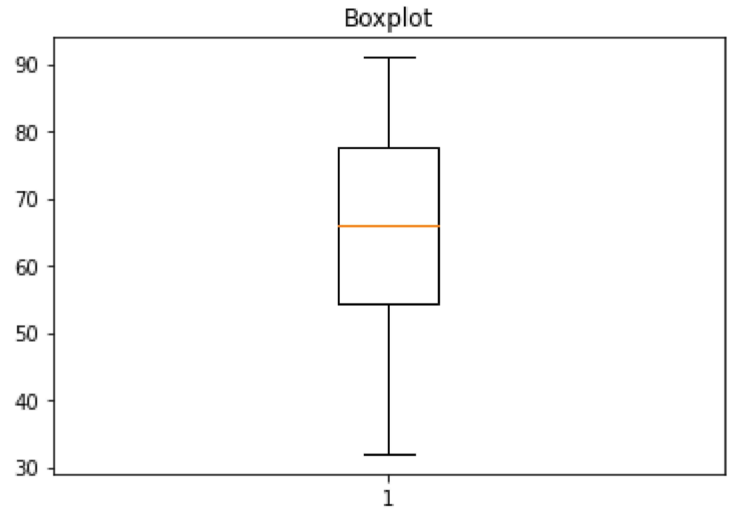
In [7]:
```
# We now plot the boxplot for the data using each feature independently and check for
plt.boxplot(crime.Murder);plt.title('Boxplot');plt.show()

# We see that there are Outliers present for "Balance" Feature
```



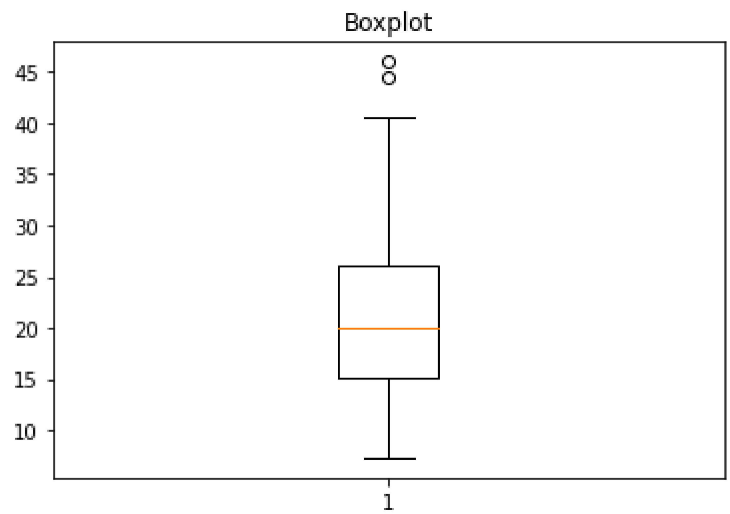
In [8]:
```
plt.boxplot(crime.Assault);plt.title('Boxplot');plt.show()  # outliers present
```



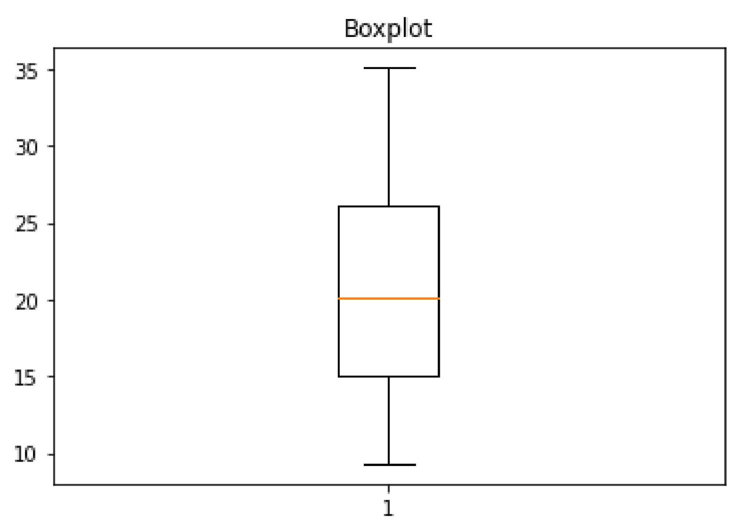
In [9]:
```
plt.boxplot(crime.UrbanPop);plt.title('Boxplot');plt.show()  # No outliers
```



In [10]:
```
plt.boxplot(crime.Rape);plt.title('Boxplot');plt.show()  # outliers present
```



In [11]:
```
from scipy.stats.mstats import winsorize

crime['Rape'] = winsorize(crime.Rape, limits=[0.07, 0.093])
plt.boxplot(crime['Rape']);plt.title('Boxplot');plt.show()
```



In [12]:
```
# Now we check the data for zero variance values
(crime == 0).all()
```

Out[12]:
```
State       False
Murder      False
Assault     False
UrbanPop    False
Rape        False
dtype: bool
```

In [13]:
```
# We see the data again now to check whether the data is in scale
crime.describe

# we notice that the data needs to be normalise, using normalization
```

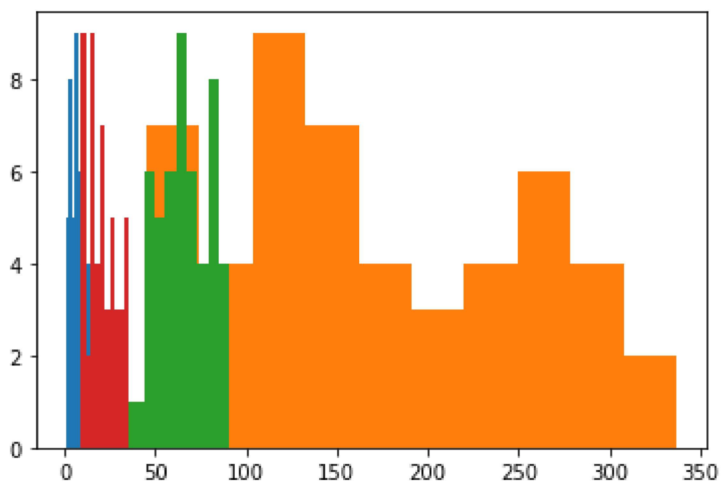
Out[13]:
```
<bound method NDFrame.describe of             State  Murder  Assault  UrbanPop  Rape
0          Alabama    13.2      236        58  21.2
1           Alaska    10.0      263        48  35.1
2          Arizona     8.1      294        80  31.0
3         Arkansas     8.8      190        50  19.5
4       California     9.0      276        91  35.1
5         Colorado     7.9      204        78  35.1
6      Connecticut     3.3      110        77  11.1
7         Delaware     5.9      238        72  15.8
8          Florida    15.4      335        80  31.9
9          Georgia    17.4      211        60  25.8
10          Hawaii     5.3       46        83  20.2
11           Idaho     2.6      120        54  14.2
12        Illinois    10.4      249        83  24.0
13         Indiana     7.2      113        65  21.0
14            Iowa     2.2       56        57  11.3
15          Kansas     6.0      115        66  18.0
16        Kentucky     9.7      109        52  16.3
17       Louisiana    15.4      249        66  22.2
18           Maine     2.1       83        51   9.3
19        Maryland    11.3      300        67  27.8
20   Massachusetts     4.4      149        85  16.3
21        Michigan    12.1      255        74  35.1
22       Minnesota     2.7       72        66  14.9
23     Mississippi    16.1      259        44  17.1
24        Missouri     9.0      178        70  28.2
25         Montana     6.0      109        53  16.4
26        Nebraska     4.3      102        62  16.5
27          Nevada    12.2      252        81  35.1
28   New Hampshire     2.1       57        56   9.5
29      New Jersey     7.4      159        89  18.8
30      New Mexico    11.4      285        70  32.1
31        New York    11.1      254        86  26.1
32  North Carolina    13.0      337        45  16.1
33    North Dakota     0.8       45        44   9.3
34            Ohio     7.3      120        75  21.4
35        Oklahoma     6.6      151        68  20.0
36          Oregon     4.9      159        67  29.3
37    Pennsylvania     6.3      106        72  14.9
38    Rhode Island     3.4      174        87   9.3
39  South Carolina    14.4      279        48  22.5
40    South Dakota     3.8       86        45  12.8
41       Tennessee    13.2      188        59  26.9
42           Texas    12.7      201        80  25.5
43            Utah     3.2      120        80  22.9
44         Vermont     2.2       48        32  11.2
45        Virginia     8.5      156        63  20.7
46      Washington     4.0      145        73  26.2
47   West Virginia     5.7       81        39   9.3
48       Wisconsin     2.6       53        66  10.8
49         Wyoming     6.8      161        60  15.6>
```

In [14]:
```
def norm_func(i):
    x = (i - i.min())   / (i.max() - i.min())
    return (x)

# Normalized data frame (considering the numerical part of data)
df_norm = norm_func(crime.iloc[:,1:])
```

In [15]:
```
##########################Univariate, Bivariate################
plt.hist(crime["Murder"])   #Univariate

plt.hist(crime["Assault"])

plt.hist(crime["UrbanPop"])

plt.hist(crime["Rape"])

crime.skew(axis = 0, skipna = True)

crime.kurtosis(axis = 0, skipna = True)
```

```
<ipython-input-15-e62e0e231209>:10: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns before calling the reduction.
  crime.skew(axis = 0, skipna = True)
<ipython-input-15-e62e0e231209>:12: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns before calling the reduction.
  crime.kurtosis(axis = 0, skipna = True)
```

Out[15]:
```
Murder     -0.827488
Assault    -1.053848
UrbanPop   -0.738360
Rape       -0.883786
dtype: float64
```

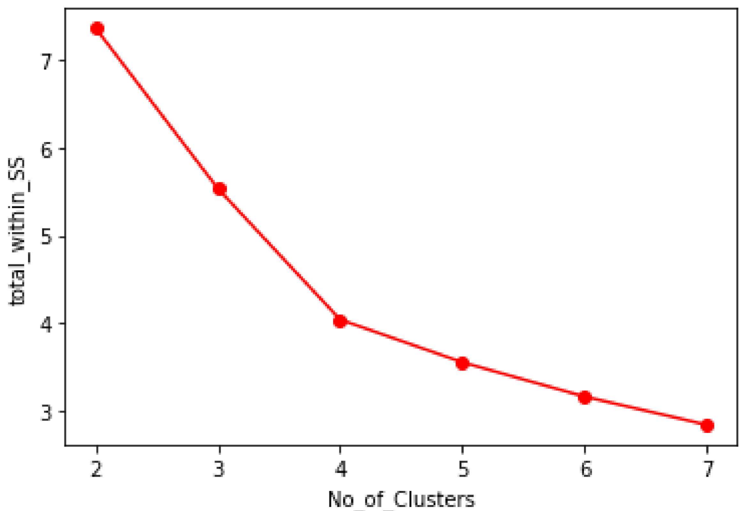


In [16]:
```
# calculating TWSS - Total within SS using different cluster range
from sklearn.cluster import KMeans

TWSS = []
k = list(range(2, 8))

for i in k:
    kmeans = KMeans(n_clusters = i)
    kmeans.fit(df_norm)
    TWSS.append(kmeans.inertia_)

TWSS
```

Out[16]:
```
[7.358376498536079,
 5.532071995078602,
 4.0407678952238815,
 3.5539811127025747,
 3.1628651131109455,
 2.8417637970747243]
```

In [17]:
```
# Plotting the Scree plot using the TWSS from above defined function
plt.plot(k, TWSS, 'ro-');plt.xlabel("No_of_Clusters");plt.ylabel("total_within_SS")
```

Out[17]:
```
Text(0, 0.5, 'total_within_SS')
```



In [18]:
```
# Selecting 4 clusters from the above scree plot which is the optimum number of cluste
# as the curve is seemingly bent or showinf an elbow format at K = 4

model = KMeans(n_clusters = 4)
model.fit(df_norm)
```

Out[18]:
```
KMeans(n_clusters=4)
```

In [19]:
```
model.labels_ # getting the labels of clusters assigned to each row
```

Out[19]:
```
array([2, 1, 1, 2, 1, 1, 3, 3, 1, 2, 3, 0, 1, 3, 0, 3, 0, 2, 0, 1, 3, 1,
       0, 2, 1, 0, 0, 1, 0, 3, 1, 1, 2, 0, 3, 3, 3, 3, 3, 2, 0, 2, 1, 3,
       0, 3, 3, 0, 0, 3])
```

In [20]:
```
mb = pd.Series(model.labels_)  # converting numpy array into pandas series object
```

In [21]:
```
crime['clust'] = mb # creating a  new column and assigning it to new column
```

In [22]:
```
crime.head()
```

Out[22]:

	State	Murder	Assault	UrbanPop	Rape	clust
0	Alabama	13.2	236	58	21.2	2
1	Alaska	10.0	263	48	35.1	1
2	Arizona	8.1	294	80	31.0	1
3	Arkansas	8.8	190	50	19.5	2
4	California	9.0	276	91	35.1	1

In [23]:

```
crime = crime.iloc[:,[5,0,1,2,3,4]]
crime.head()
```

Out[23]:

	clust	State	Murder	Assault	UrbanPop	Rape
0	2	Alabama	13.2	236	58	21.2
1	1	Alaska	10.0	263	48	35.1
2	1	Arizona	8.1	294	80	31.0
3	2	Arkansas	8.8	190	50	19.5
4	1	California	9.0	276	91	35.1

In [24]:

```
# We can clearly see that we have the labels in the dataset in the form of a column ca
```

In [26]:

```
# In order to see the clusters we aggregate the records within the clusters and group
# 4 nos of clear cluster formed
crime.iloc[:, 1:6].groupby(crime.clust).mean()
```

Out[26]:

clust	Murder	Assault	UrbanPop	Rape
0	3.600000	78.538462	52.076923	12.446154
1	10.815385	257.384615	76.000000	30.930769
2	13.937500	243.625000	53.750000	21.412500
3	5.656250	138.875000	73.875000	18.843750

In [ ]: