In [1]:
```
#Task:Perform Clustering for the crime data and identify the number of clusters f
```

1. Import Nessary Libraries

In [2]:
```
import pandas as pd
import seaborn as  sns
from sklearn.preprocessing import normalize
from sklearn.cluster import KMeans
from scipy.cluster.hierarchy import linkage
import scipy.cluster.hierarchy as sch
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

2. Import DataSet

In [3]:
```
data=pd.read_csv('crime_data.csv')
data.head()
```

Out[3]:

	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

3. Data Undestanding

In [4]:
```
data.isnull().sum()
```

Out[4]:
```
Unnamed: 0    0
Murder        0
Assault       0
UrbanPop      0
Rape          0
dtype: int64
```

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

Out[5]:
```
(50, 5)
```

In [6]:
```
data.dtypes
```

Out[6]:
```
Unnamed: 0     object
Murder        float64
Assault         int64
UrbanPop        int64
Rape          float64
dtype: object
```

In [7]:
```
data.describe() #Describe the function
```

Out[7]:

	Murder	Assault	UrbanPop	Rape
count	50.00000	50.000000	50.000000	50.000000
mean	7.78800	170.760000	65.540000	21.232000
std	4.35551	83.337661	14.474763	9.366385
min	0.80000	45.000000	32.000000	7.300000
25%	4.07500	109.000000	54.500000	15.075000
50%	7.25000	159.000000	66.000000	20.100000
75%	11.25000	249.000000	77.750000	26.175000
max	17.40000	337.000000	91.000000	46.000000

In [8]:
```
#Correlation between each columns.
data.corr()
```

Out[8]:

	Murder	Assault	UrbanPop	Rape
Murder	1.000000	0.801873	0.069573	0.563579
Assault	0.801873	1.000000	0.258872	0.665241
UrbanPop	0.069573	0.258872	1.000000	0.411341
Rape	0.563579	0.665241	0.411341	1.000000

4. Data Preparing

In [9]:
```
x=data.iloc[:,1:]
#we are taking only numeric columns for further process
```

In [10]:
```
# Normalization function
def norm_fun(i):
    x=(i-i.min())/(i.max()-i.min())
    return x
```

In [11]:
```
df_norm=norm_fun(x)
```

In [12]:
```
z=linkage(y=df_norm, method='complete', metric='euclidean')
```

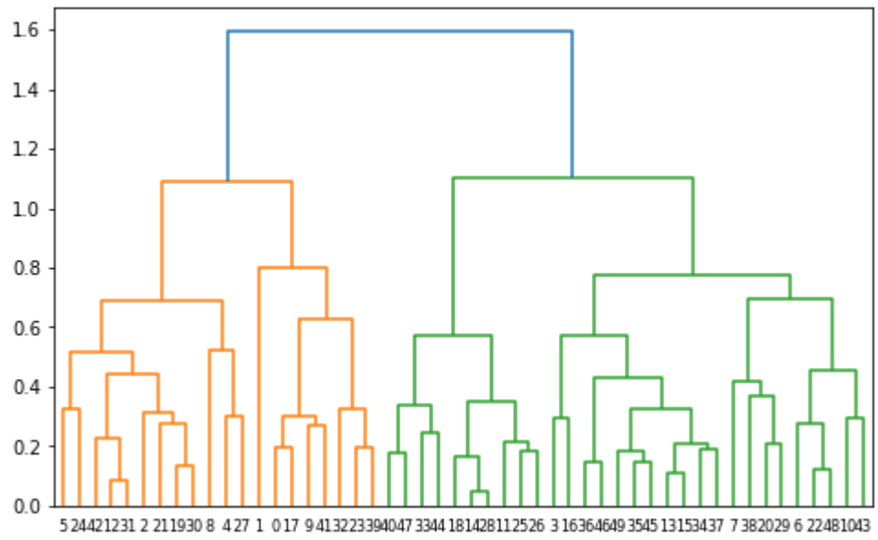
5. Plot the DENDOGRAM

In [13]:
```
###### DENDOGRAM #############

plt.figure(figsize=(8, 5))

sch.dendrogram(z,leaf_font_size=8,leaf_rotation=0)
plt.plot()
```

Out[13]: []



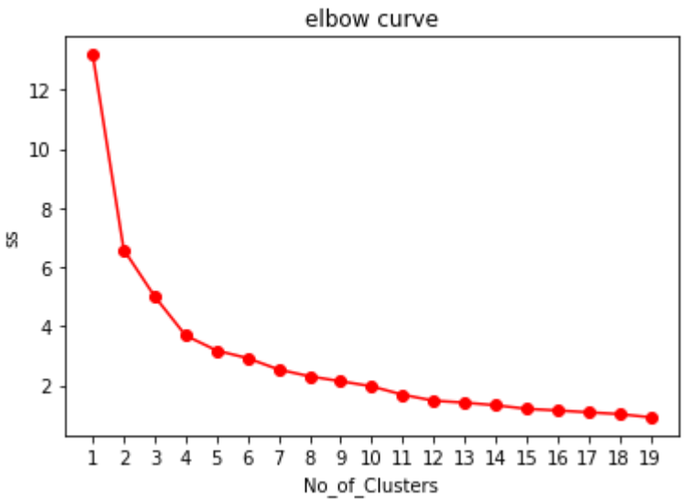
6.Elbow Curve

In [14]:
```
wcss=[]

#for find out best number of k
for i in range(1,20):
    knn=KMeans(n_clusters=i)
    knn.fit(df_norm)
    knn.predict(df_norm)
    wcss.append(knn.inertia_)
print(wcss)  # it print inertia of k value start from 1 to 20
#after some point value of ineria has  no major drop that's we call elbow point.
#that point indicate ideal number of k we should pick up.
```

```
[13.184122550256445, 6.596893867946199, 5.010878493006419, 3.690820410392112,
3.183157731676654, 2.9248045477302056, 2.5407532357086757, 2.310883669570136,
2.1447887918682382, 1.9730078263292596, 1.6895797320703811, 1.4870840680949293,
1.420970293682285, 1.33301591163183, 1.2114203782124078, 1.1563065475652206, 1.
094463763899287, 1.033954618347078, 0.9244592254098558]
```

In [15]:
```
plt.plot(range(1,20),wcss,'ro-')
plt.title("elbow curve")
plt.xlabel("No_of_Clusters")
plt.ylabel("ss")
plt.xticks(range(1,20))
plt.show()                    #ploting range 1 to 20 in x label and  inertia of c
```



The elbow appear to be smoothening out after four clusters indicating that the optimal number of clusters is 4.

7. Taking number of clusters is 4

In [16]:
```
model=KMeans(n_clusters=4)  # taking number of cluster as 4 for prediction
model.fit(df_norm)
```

Out[16]:
```
KMeans(n_clusters=4)
```

In [17]:
```
model.inertia_
```

Out[17]:
```
3.690820410392112
```

In [18]:
```
model.labels_ #Row wise indication of cluster group
```

Out[18]:
```
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 [19]:
```
model.cluster_centers_
```

Out[19]:
```
array([[0.1686747 , 0.11485774, 0.34028683, 0.12601868],
       [0.60333642, 0.72734457, 0.74576271, 0.66905188],
       [0.79141566, 0.6802226 , 0.36864407, 0.36466408],
       [0.29254518, 0.32148973, 0.70974576, 0.29667313]])
```

In [20]:
```
X = data[['Murder', 'Assault', 'Rape', 'UrbanPop']]
clusters = KMeans(4)  # 4 clusters!
clusters.fit( X )
clusters.cluster_centers_
clusters.labels_
data['clusters'] = clusters.labels_

data.sort_values(by=['clusters'],ascending = True)
data.head()
```

Out[20]:

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

In [21]:
```
data1=data.sort_values( 'Murder',ascending=True) #sort the value according murder
data1.head()
```

Out[21]:

	Unnamed: 0	Murder	Assault	UrbanPop	Rape	clusters
33	North Dakota	0.8	45	44	7.3	2
28	New Hampshire	2.1	57	56	9.5	2
18	Maine	2.1	83	51	7.8	2
14	Iowa	2.2	56	57	11.3	2
44	Vermont	2.2	48	32	11.2	2

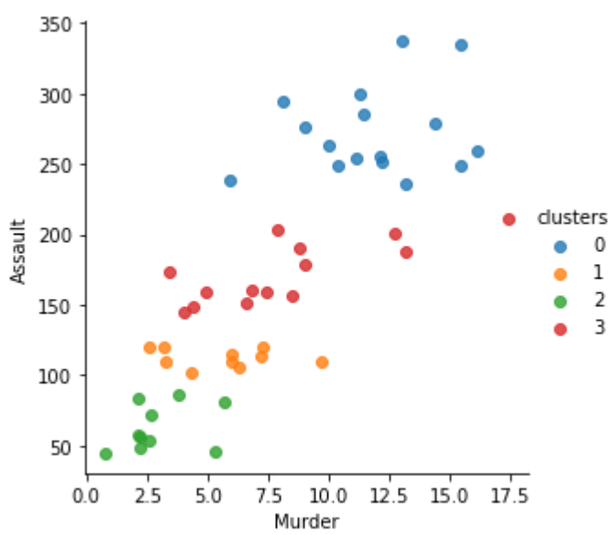
8. Ploting of Diffrent Columns.

Plot :1

In [22]:
```
#plot between pair Murder~Accault

sns.lmplot('Murder','Assault',hue='clusters',data=data,fit_reg=False, size = 4)
```

Out[22]: <seaborn.axisgrid.FacetGrid at 0x204f0596640>

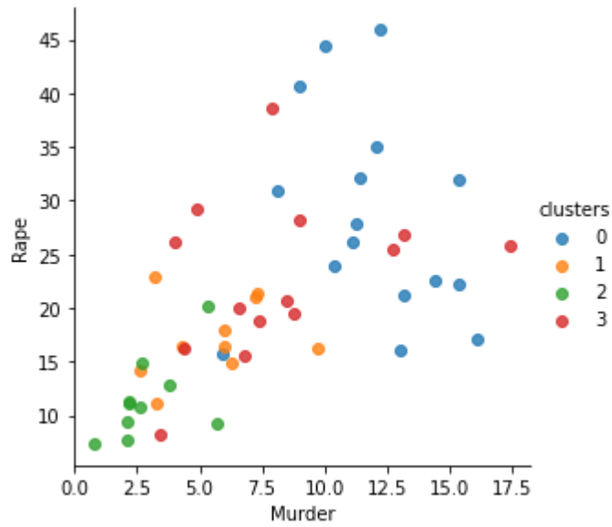


Plot :2

In [23]:
```
# Plot between pairs Murder~Rape

sns.lmplot('Murder','Rape',hue='clusters',data=data1,fit_reg=False,size=4)
```

Out[23]: <seaborn.axisgrid.FacetGrid at 0x204f0cb3790>



Plot:3

In [25]:
```
#plot between Assault~Rape

sns.lmplot('Assault','Rape',hue='clusters',data=data1,fit_reg=False,size=4)
```

Out[25]: <seaborn.axisgrid.FacetGrid at 0x204f05dff70>