In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:

```
# load the data
data = pd.read_csv("crime_data.csv")
data
```

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
5	Colorado	7.9	204	78	38.7
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	7.8
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	46.0
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	7.3
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	8.3
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

EDA

In [3]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Unnamed: 0  50 non-null     object 
 1   Murder      50 non-null     float64
 2   Assault     50 non-null     int64  
 3   UrbanPop    50 non-null     int64  
 4   Rape        50 non-null     float64
dtypes: float64(2), int64(2), object(1)
memory usage: 2.1+ KB
```

In [4]:

```
# see the null values
data.isnull().sum()
```

Out[4]:

```
Unnamed: 0    0
Murder        0
Assault       0
UrbanPop      0
Rape          0
dtype: int64
```

In [5]:

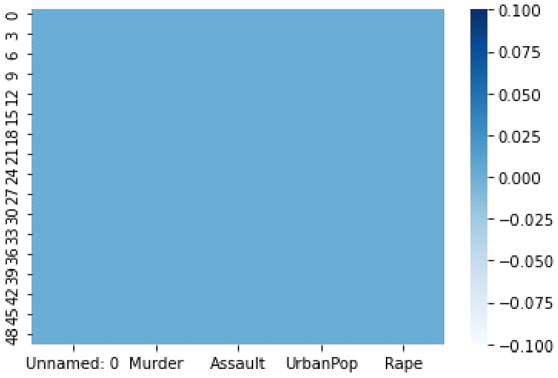
```
import seaborn as sns
```

In [6]:

```
sns.heatmap(data.isnull(),cmap='Blues')
```

Out[6]:

```
<AxesSubplot:>
```



In [7]:

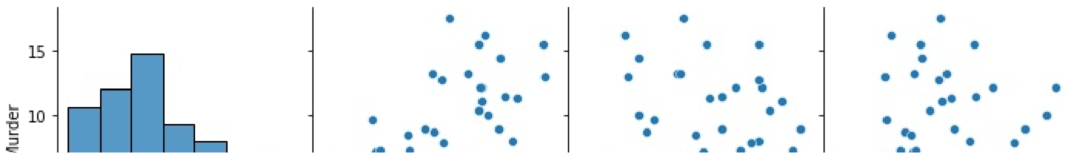
```
# there are no null values
```

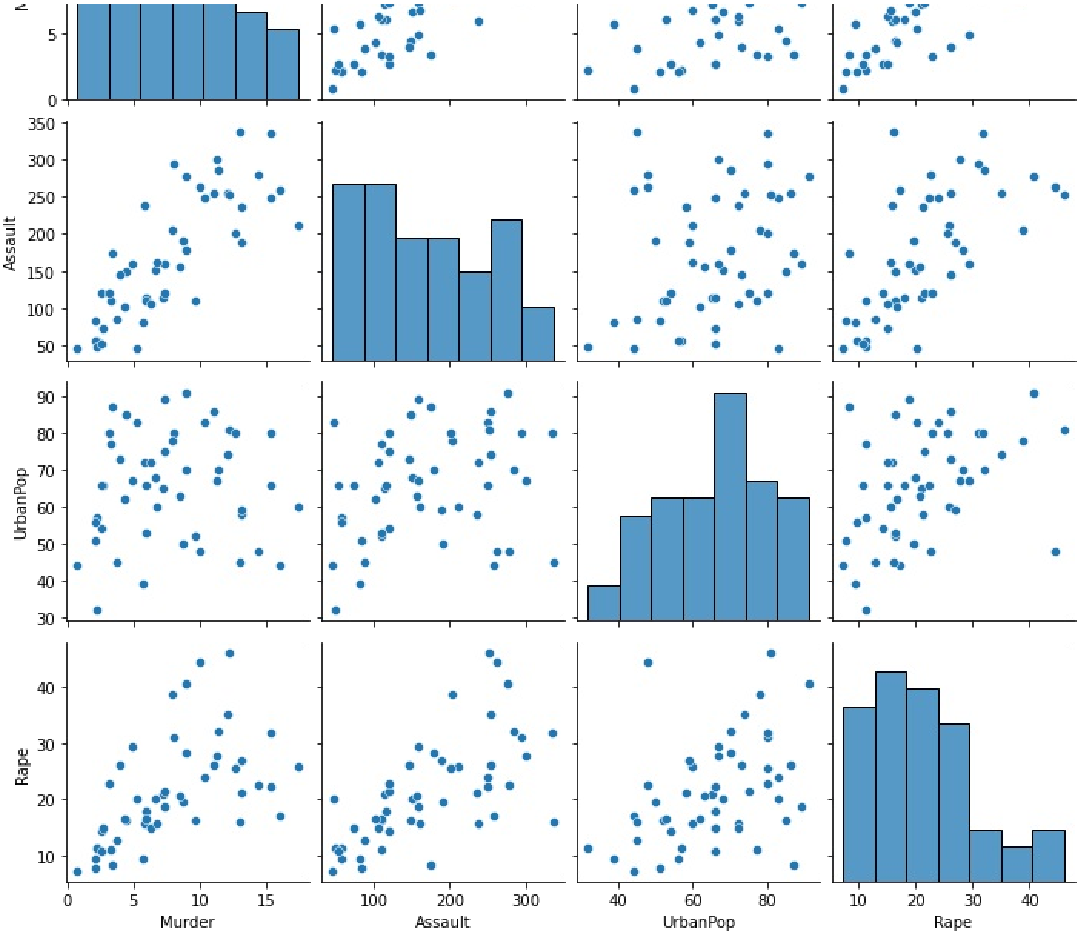
In [8]:

```
sns.pairplot(data)
```

Out[8]:

```
<seaborn.axisgrid.PairGrid at 0x293ecd39310>
```





In [9]:

```
data.value_counts()
```

Out[9]:

```
Unnamed: 0      Murder  Assault  UrbanPop  Rape
Alabama         13.2    236      58        21.2    1
Pennsylvania    6.3     106      72        14.9    1
Nevada          12.2    252      81        46.0    1
New Hampshire   2.1     57       56        9.5     1
New Jersey      7.4     159      89        18.8    1
New Mexico      11.4    285      70        32.1    1
New York        11.1    254      86        26.1    1
North Carolina  13.0    337      45        16.1    1
North Dakota    0.8     45       44        7.3     1
Ohio            7.3     120      75        21.4    1
Oklahoma        6.6     151      68        20.0    1
Oregon          4.9     159      67        29.3    1
Rhode Island    3.4     174      87        8.3     1
Alaska          10.0    263      48        44.5    1
South Carolina  14.4    279      48        22.5    1
South Dakota    3.8     86       45        12.8    1
Tennessee       13.2    188      59        26.9    1
Texas           12.7    201      80        25.5    1
Utah            3.2     120      80        22.9    1
Vermont         2.2     48       32        11.2    1
Virginia        8.5     156      63        20.7    1
Washington      4.0     145      73        26.2    1
West Virginia   5.7     81       39        9.3     1
Wisconsin       2.6     53       66        10.8    1
Nebraska        4.3     102      62        16.5    1
Montana         6.0     109      53        16.4    1
Missouri        9.0     178      70        28.2    1
Mississippi     16.1    259      44        17.1    1
Arizona         8.1     294      80        31.0    1
Arkansas        8.8     190      50        19.5    1
California      9.0     276      91        40.6    1
Colorado        7.9     204      78        38.7    1
Connecticut     3.3     110      77        11.1    1
Delaware        5.9     238      72        15.8    1
Florida         15.4    335      80        31.9    1
Georgia         17.4    211      60        25.8    1
Hawaii          5.3     46       83        20.2    1
Idaho           2.6     120      54        14.2    1
Illinois        10.4    249      83        24.0    1
Indiana         7.2     113      65        21.0    1
Iowa            2.2     56       57        11.3    1
Kansas          6.0     115      66        18.0    1
Kentucky        9.7     109      52        16.3    1
Louisiana       15.4    249      66        22.2    1
```

```
Maine           2.1     83       51        7.8      1
Maryland        11.3    300      67        27.8     1
Massachusetts   4.4     149      85        16.3     1
Michigan        12.1    255      74        35.1     1
Minnesota       2.7     72       66        14.9     1
Wyoming         6.8     161      60        15.6     1
dtype: int64
```

In [10]:

```
data.describe()
```

Out[10]:

	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 [11]:

```
# drop the unnamed coloums
```

In [12]:

```
d = data.drop("Unnamed: 0",axis=1)
```

In [13]:

```
d.head()
```

Out[13]:

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

In [14]:

```
# see the duplicates
data.duplicated()   # there are no duplicates
```

Out[14]:

```
0      False
1      False
2      False
3      False
4      False
5      False
6      False
7      False
8      False
9      False
10     False
11     False
12     False
13     False
14     False
15     False
16     False
17     False
18     False
19     False
20     False
21     False
22     False
23     False
24     False
25     False
26     False
27     False
28     False
29     False
30     False
```

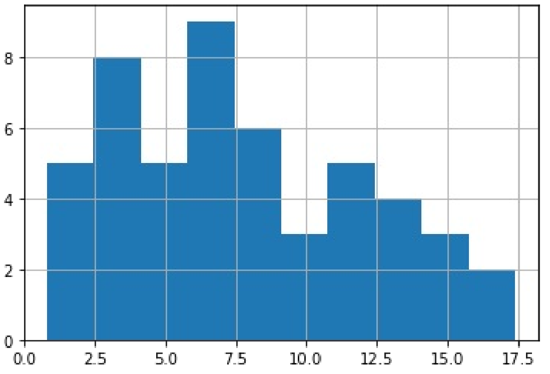
```
31    False
32    False
33    False
34    False
35    False
36    False
37    False
38    False
39    False
40    False
41    False
42    False
43    False
44    False
45    False
46    False
47    False
48    False
49    False
dtype: bool
```

In [15]:

```
d['Murder'].hist()
```

Out[15]:

```
<AxesSubplot:>
```

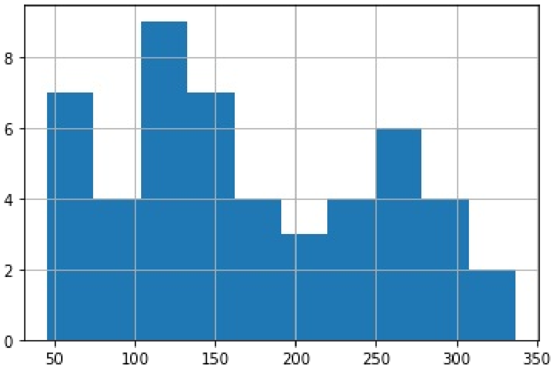


In [16]:

```
d['Assault'].hist()
```

Out[16]:

```
<AxesSubplot:>
```

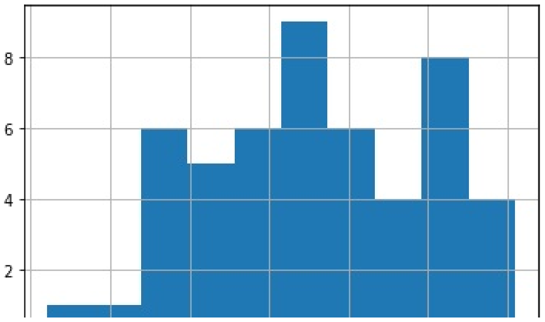


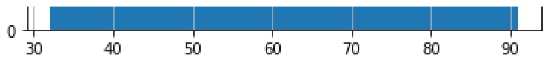
In [17]:

```
d['UrbanPop'].hist()
```

Out[17]:

```
<AxesSubplot:>
```





Normalise the Data

In [ ]:

```

```

In [ ]:

```

```

In [18]:

```
# Normalization function
def norm_func(i):
    x = (i-i.min())/(i.max()-i.min())
    return (x)
```

In [19]:

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

In [20]:

```
df_norm.head()
```

Out[20]:

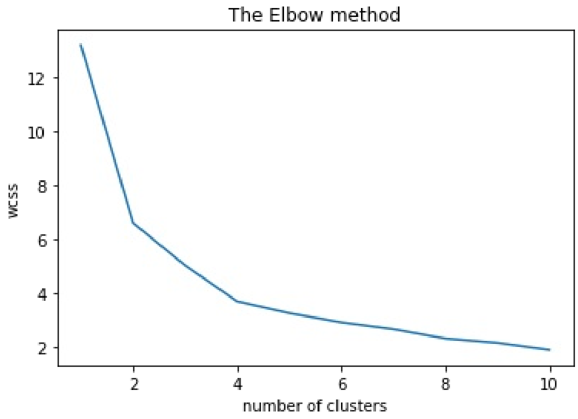
	Murder	Assault	UrbanPop	Rape
0	0.746988	0.654110	0.440678	0.359173
1	0.554217	0.746575	0.271186	0.961240
2	0.439759	0.852740	0.813559	0.612403
3	0.481928	0.496575	0.305085	0.315245
4	0.493976	0.791096	1.000000	0.860465

WCSS - within cluster Sum of Square

In [21]:

```
# We are Using Elbow Graph to find optimum number of clusters (K value) from K values range
from sklearn.cluster import KMeans
wcss = []
for i in range (1,11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state= 42)
    kmeans.fit(df_norm)
    wcss.append(kmeans.inertia_)
plt.plot(range(1,11),wcss)
plt.title('The Elbow method')
plt.xlabel('number of clusters')
plt.ylabel('wcss')
plt.show()
```

```
C:\Users\rajesh\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:881: UserWarning: KMeans is known to have
a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting
the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```



In [22]:

```
# by the elbow we can see the there are 4clusters
```

In [23]:

```
wcss
```

Out[23]:

```
[13.184122550256445,
 6.596893867946199,
 5.0184999914891115,
 3.683456153585915,
 3.249870851106594,
 2.903479372843045,
 2.6533726943439317,
 2.30474654408108,
 2.1433148484911446,
 1.8842960184808824]
```

K-means

In [24]:

```
# Build The Cluster Algorithm With K =4
kmeans = KMeans(n_clusters=4, init='k-means++', random_state=0)
y_means = kmeans.fit_predict(df_norm)
```

In [25]:

```
y_means
```

Out[25]:

```
array([2, 0, 0, 2, 0, 0, 1, 1, 0, 2, 1, 3, 0, 1, 3, 1, 3, 2, 3, 0, 1, 0,
       3, 2, 1, 3, 3, 0, 3, 1, 0, 0, 2, 3, 1, 1, 1, 1, 1, 2, 3, 2, 0, 1,
       3, 1, 1, 3, 3, 1])
```

In [26]:

```
df_norm['Cluster']=y_means
df_norm.head()
```

Out[26]:

	Murder	Assault	UrbanPop	Rape	Cluster
0	0.746988	0.654110	0.440678	0.359173	2
1	0.554217	0.746575	0.271186	0.961240	0
2	0.439759	0.852740	0.813559	0.612403	0
3	0.481928	0.496575	0.305085	0.315245	2
4	0.493976	0.791096	1.000000	0.860465	0

In [27]:

```
import sklearn.cluster as cluster
```

In [28]:

```
from sklearn import metrics
```

In [29]:

```
# Using The Silhoutee Score we can whether k=4 cluster are not
for i in range(3,13):
    labels=cluster.KMeans(n_clusters=i,init="k-means++",random_state=200).fit(df_norm).labels_
    print ("Silhouette score for k(clusters) = "+str(i)+" is "
           +str(metrics.silhouette_score(df_norm,labels,metric="euclidean",sample_size=1000,random_state=200)))
```

```
Silhouette score for k(clusters) = 3 is 0.5864466445785239
Silhouette score for k(clusters) = 4 is 0.7041330922958315
Silhouette score for k(clusters) = 5 is 0.5487050284097537
Silhouette score for k(clusters) = 6 is 0.4455962081559419
Silhouette score for k(clusters) = 7 is 0.3113422696814967
Silhouette score for k(clusters) = 8 is 0.3299440611325817
Silhouette score for k(clusters) = 9 is 0.3469562205025572
Silhouette score for k(clusters) = 10 is 0.2687864178584056
Silhouette score for k(clusters) = 11 is 0.2793593570987476
Silhouette score for k(clusters) = 12 is 0.25625741896429743
```

In [30]:

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

Out[30]:

```
array([3, 0, 0, 3, 0, 0, 2, 2, 0, 3, 2, 1, 0, 2, 1, 2, 1, 3, 1, 0, 2, 0,
       1, 3, 2, 1, 1, 0, 1, 2, 0, 0, 3, 1, 2, 2, 2, 2, 2, 3, 1, 3, 0, 2,
       1, 2, 2, 1, 1, 2])
```

In [31]:

```
km = pd.Series(model.labels_)
df_norm['kclust']= km
df_norm.iloc[:,:4].groupby(df_norm.kclust).mean()
```

Out[31]:

kclust	Murder	Assault	UrbanPop	Rape
0	0.612450	0.750000	0.754237	0.679802
1	0.168675	0.114858	0.340287	0.126019
2	0.304394	0.329371	0.705882	0.310990
3	0.791416	0.680223	0.368644	0.364664

Plot the Clusters

In [32]:

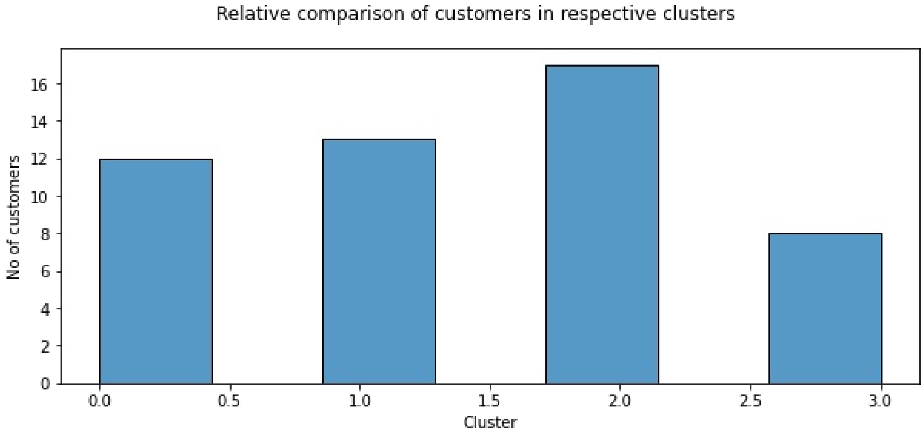
```
import seaborn as sns
```

In [33]:

```
plt.figure(figsize=(10,4))
sns.histplot (x='kclust', data=df_norm)
plt.xlabel('Cluster')
plt.ylabel('No of customers')
plt.suptitle('Relative comparison of customers in respective clusters')
```

Out[33]:

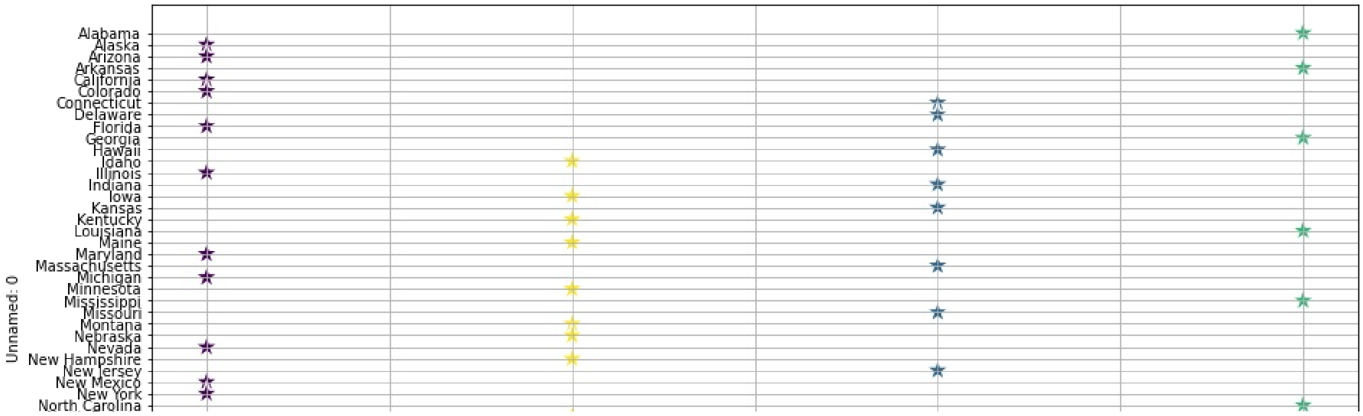
```
Text(0.5, 0.98, 'Relative comparison of customers in respective clusters')
```

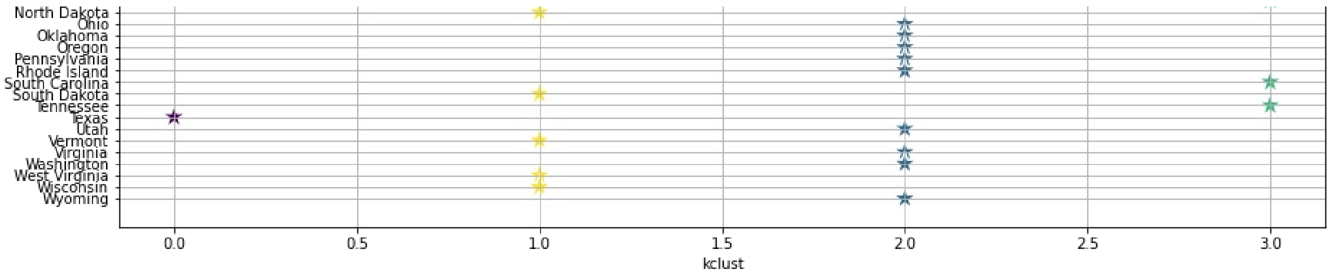


In [34]:

```
plt.figure(figsize=(15,8))
sns.scatterplot(df_norm['kclust'],data['Unnamed: 0'],c=kmeans.labels_,s=300,marker='*')
plt.grid()
plt.show();
```

```
C:\Users\rajesh\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable
s as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



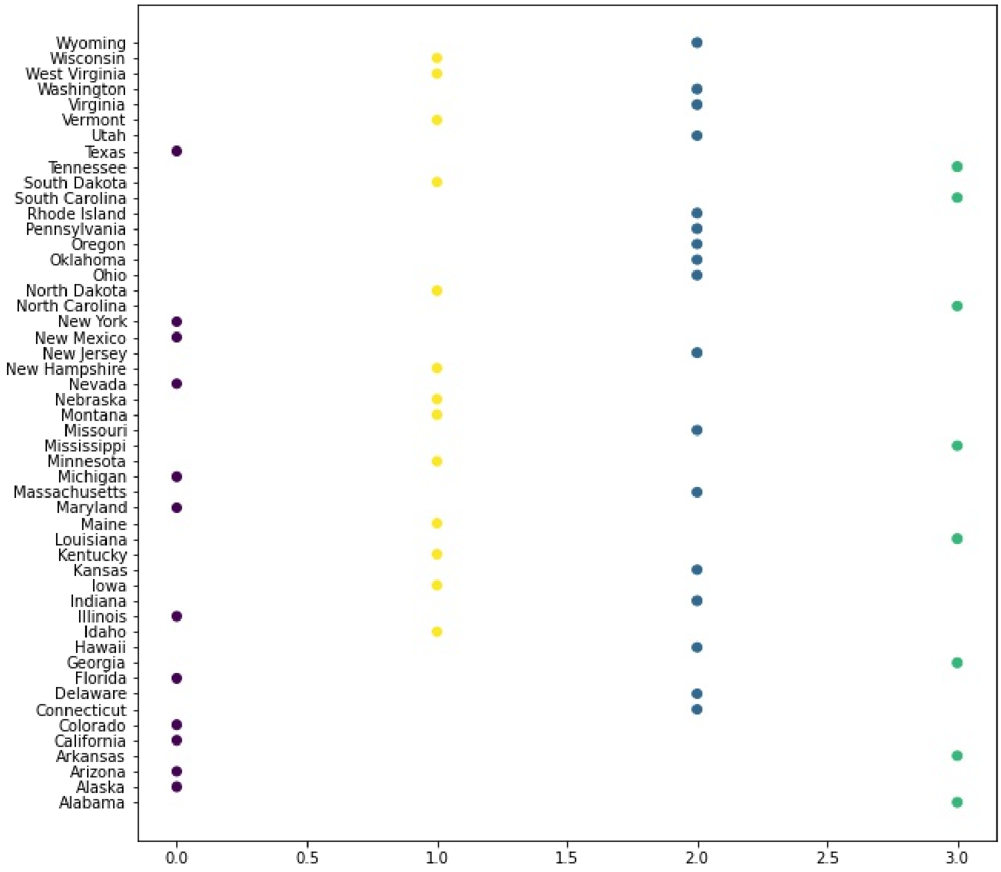


In [35]:

```
plt.figure(figsize=(10, 10))
plt.scatter(df_norm['kclust'],data['Unnamed: 0'], c=kmeans.labels_)
```

Out[35]:

```
<matplotlib.collections.PathCollection at 0x293f01ec490>
```



- We Can See The Are 4Clusters Are Formed By Given Data

Hierachical - Clustering

In [36]:

```
df = df_norm.drop("Cluster",axis=1)
```

In [37]:

```
df.head()
```

Out[37]:

	Murder	Assault	UrbanPop	Rape	kclust
0	0.746988	0.654110	0.440678	0.359173	3
1	0.554217	0.746575	0.271186	0.961240	0
2	0.439759	0.852740	0.813559	0.612403	0
3	0.481928	0.496575	0.305085	0.315245	3
4	0.493976	0.791096	1.000000	0.860465	0

DenDrogram

In [38]:

```
import scipy.cluster.hierarchy as sch
```

In [39]:

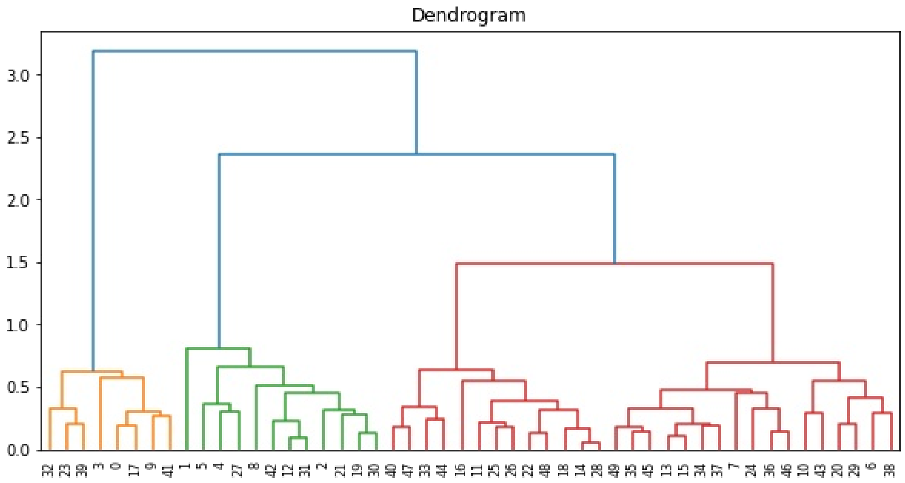
```
from sklearn.cluster import AgglomerativeClustering
```

In [40]:

```
# create the dendrogram
plt.figure(figsize=(10,5))
dendrogram = sch.dendrogram(sch.linkage(df,method='complete'))
plt.title('Dendrogram')
```

Out[40]:

```
Text(0.5, 1.0, 'Dendrogram')
```



Train Model

In [41]:

```
hc = AgglomerativeClustering(n_clusters=4, affinity = 'euclidean',linkage = 'complete')
```

In [42]:

```
hc
```

Out[42]:

```
AgglomerativeClustering(linkage='complete', n_clusters=4)
```

In [43]:

```
y_hc = hc.fit_predict(df)
```

In [44]:

```
y_hc
```

Out[44]:

```
array([3, 0, 0, 3, 0, 0, 2, 2, 0, 3, 2, 1, 0, 2, 1, 2, 1, 3, 1, 0, 2, 0,
       1, 3, 2, 1, 1, 0, 1, 2, 0, 0, 3, 1, 2, 2, 2, 2, 2, 3, 1, 3, 0, 2,
       1, 2, 2, 1, 1, 2], dtype=int64)
```

In [45]:

```
cluster = df.drop("kclust",axis=1)
```

In [46]:

```
cluster.head()
```

Out[46]:

	Murder	Assault	UrbanPop	Rape
0	0.746988	0.654110	0.440678	0.359173
1	0.554217	0.746575	0.271186	0.961240
2	0.439759	0.852740	0.813559	0.612403
3	0.481928	0.496575	0.305085	0.315245
4	0.493976	0.791096	1.000000	0.860465

In [47]:

```
y=pd.DataFrame(hc.fit_predict(cluster),columns=['clustersid'])
y['clustersid'].value_counts()
```

Out[47]:

```
1    20
3    12
2    10
0     8
Name: clustersid, dtype: int64
```

In [48]:

```
cluster['clustersid']=hc.labels_
df.head(10)
```

Out[48]:

	Murder	Assault	UrbanPop	Rape	kclust
0	0.746988	0.654110	0.440678	0.359173	3
1	0.554217	0.746575	0.271186	0.961240	0
2	0.439759	0.852740	0.813559	0.612403	0
3	0.481928	0.496575	0.305085	0.315245	3
4	0.493976	0.791096	1.000000	0.860465	0
5	0.427711	0.544521	0.779661	0.811370	0
6	0.150602	0.222603	0.762712	0.098191	2
7	0.307229	0.660959	0.677966	0.219638	2
8	0.879518	0.993151	0.813559	0.635659	0
9	1.000000	0.568493	0.474576	0.478036	3

In [49]:

```
cluster.head(2)
```

Out[49]:

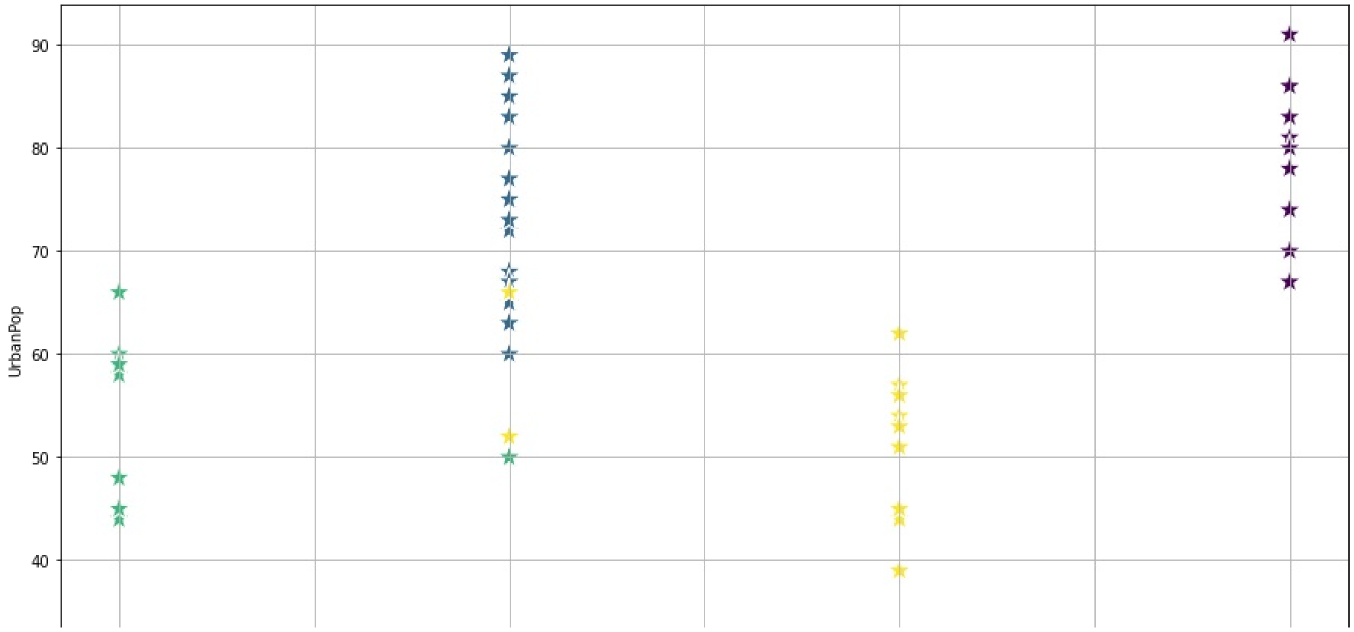
	Murder	Assault	UrbanPop	Rape	clustersid
0	0.746988	0.654110	0.440678	0.359173	0
1	0.554217	0.746575	0.271186	0.961240	0

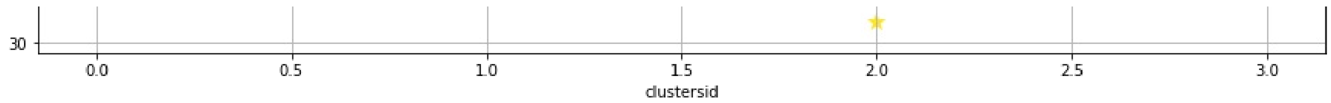
Plot Clusters

In [50]:

```
plt.figure(figsize=(15,8))
sns.scatterplot(cluster['clustersid'],data['UrbanPop'],c=kmeans.labels_,s=300,marker='*')
plt.grid()
plt.show();
```

```
C:\Users\rajesh\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable
s as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



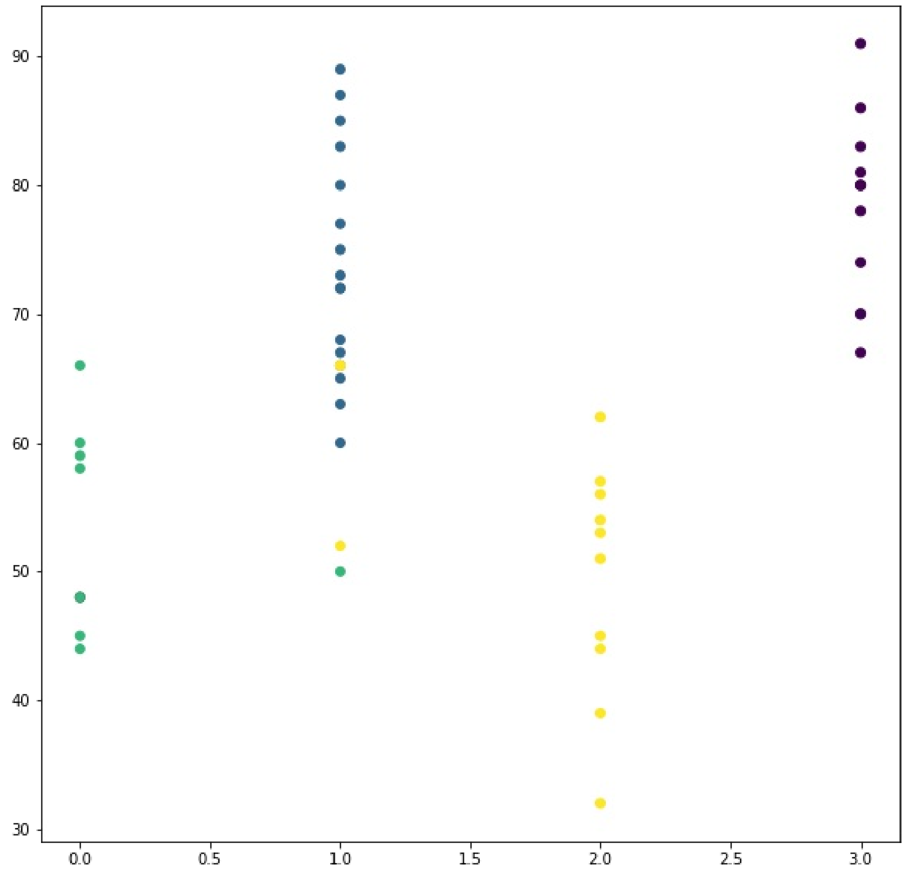


In [51]:

```
plt.figure(figsize=(10, 10))
plt.scatter(cluster['clustersid'],data['UrbanPop'], c=kmeans.labels_)
```

Out[51]:

```
<matplotlib.collections.PathCollection at 0x293f02bc670>
```



DB-SCAN

- Density-Based Spatial Clustering Applications with Noise (DB-scan)

In [52]:

```
import pandas as pd
from sklearn.cluster import DBSCAN
```

In [53]:

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

In [54]:

```
d.head()
```

Out[54]:

	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 [55]:

```
crime = d.drop("Unnamed: 0",axis = True)
```

In [56]:

```
crime.head(2)
```

Out[56]:

	Murder	Assault	UrbanPop	Rape
0	13.2	236	58	21.2
1	10.0	263	48	44.5

In [57]:

```
# normalize the data
def data(i):
    x = (i-i.min())/(i.max()-i.min())
    return (x)
```

In [58]:

```
#data =data.crime(d.iloc[:,:])
data = data(crime.iloc[:,:])
```

In [59]:

```
data.head()
```

Out[59]:

	Murder	Assault	UrbanPop	Rape
0	0.746988	0.654110	0.440678	0.359173
1	0.554217	0.746575	0.271186	0.961240
2	0.439759	0.852740	0.813559	0.612403
3	0.481928	0.496575	0.305085	0.315245
4	0.493976	0.791096	1.000000	0.860465

In [60]:

```
from sklearn.preprocessing import StandardScaler
```

In [61]:

```
Crime_n=StandardScaler().fit_transform(crime)
Crime_n
```

Out[61]:

```
array([[ 1.25517927,  0.79078716, -0.52619514, -0.00345116],
       [ 0.51301858,  1.11805959, -1.22406668,  2.50942392],
       [ 0.07236067,  1.49381682,  1.00912225,  1.05346626],
       [ 0.23470832,  0.23321191, -1.08449238, -0.18679398],
       [ 0.28109336,  1.2756352 ,  1.77678094,  2.08881393],
       [ 0.02597562,  0.40290872,  0.86954794,  1.88390137],
       [-1.04088037, -0.73648418,  0.79976079, -1.09272319],
       [-0.43787481,  0.81502956,  0.45082502, -0.58583422],
       [ 1.76541475,  1.99078607,  1.00912225,  1.1505301 ],
       [ 2.22926518,  0.48775713, -0.38662083,  0.49265293],
       [-0.57702994, -1.51224105,  1.21848371, -0.11129987],
       [-1.20322802, -0.61527217, -0.80534376, -0.75839217],
       [ 0.60578867,  0.94836277,  1.21848371,  0.29852525],
       [-0.13637203, -0.70012057, -0.03768506, -0.0250209 ],
       [-1.29599811, -1.39102904, -0.5959823 , -1.07115345],
       [-0.41468229, -0.67587817,  0.03210209, -0.34856705],
       [ 0.44344101, -0.74860538, -0.94491807, -0.53190987],
       [ 1.76541475,  0.94836277,  0.03210209,  0.10439756],
       [-1.31919063, -1.06375661, -1.01470522, -1.44862395],
       [ 0.81452136,  1.56654403,  0.10188925,  0.70835037],
       [-0.78576263, -0.26375734,  1.35805802, -0.53190987],
       [ 1.00006153,  1.02108998,  0.59039932,  1.49564599],
       [-1.1800355 , -1.19708982,  0.03210209, -0.68289807],
       [ 1.9277624 ,  1.06957478, -1.5032153 , -0.44563089],
       [ 0.28109336,  0.0877575 ,  0.31125071,  0.75148985],
       [-0.41468229, -0.74860538, -0.87513091, -0.521125  ],
       [-0.80895515, -0.83345379, -0.24704653, -0.51034012],
       [ 1.02325405,  0.98472638,  1.0789094 ,  2.671197  ],
       [-1.31919063, -1.37890783, -0.66576945, -1.26528114],
       [-0.08998698, -0.14254532,  1.63720664, -0.26228808],
       [ 0.83771388,  1.38472601,  0.31125071,  1.17209984],
       [ 0.76813632,  1.00896878,  1.42784517,  0.52500755],
       [ 1.20879423,  2.01502847, -1.43342815, -0.55347961],
       [-1.62069341, -1.52436225, -1.5032153 , -1.50254831],
       [-0.11317951, -0.61527217,  0.66018648,  0.01811858],
       [-0.27552716, -0.23951493,  0.1716764 , -0.13286962],
       [-0.66980002, -0.14254532,  0.10188925,  0.87012344],
       [-0.34510472, -0.78496898,  0.45082502, -0.68289807],
       [-1.01768785,  0.03927269,  1.49763233, -1.39469959],
       [ 1.53348953,  1.3119988 , -1.22406668,  0.13675217],
       [-0.92491776, -1.027393  , -1.43342815, -0.90938037],
       [ 1.25517927,  0.20896951, -0.45640799,  0.61128652],
       [ 1.13921666,  0.36654512,  1.00912225,  0.46029832],
       [-1.06407289, -0.61527217,  1.00912225,  0.17989166],
       [-1.29599811, -1.48799864, -2.34066115, -1.08193832],
       [ 0.16513075, -0.17890893, -0.17725937, -0.05737552],
       [-0.87853272, -0.31224214,  0.52061217,  0.53579242],
       [-0.48425985, -1.08799901, -1.85215107, -1.28685088],
       [-1.20322802, -1.42739264,  0.03210209, -1.1250778 ],
       [-0.22914211, -0.11830292, -0.38662083, -0.60740397]])
```

In [62]:

```
dbscan=DBSCAN(eps=1,min_samples=4)
dbscan.fit(Crime_n)
```

Out[62]:

```
DBSCAN(eps=1, min_samples=4)
```

In [63]:

```
dbscan.labels_
```

Out[63]:

```
array([ 0, -1, -1, -1, -1, -1,  1, -1, -1, -1, -1,  1, -1,  1,  1,  1,  1,
        0,  1, -1,  1, -1,  1, -1,  1,  1,  1, -1,  1,  1, -1, -1, -1,  1,
        1,  1,  1,  1,  1,  0,  1,  0, -1,  1,  1,  1,  1,  1,  1,  1],
      dtype=int64)
```

In [64]:

```
crime['clusters']=dbscan.labels_
crime.head(10)
```

Out[64]:

	Murder	Assault	UrbanPop	Rape	clusters
0	13.2	236	58	21.2	0
1	10.0	263	48	44.5	-1
2	8.1	294	80	31.0	-1
3	8.8	190	50	19.5	-1
4	9.0	276	91	40.6	-1
5	7.9	204	78	38.7	-1
6	3.3	110	77	11.1	1
7	5.9	238	72	15.8	-1
8	15.4	335	80	31.9	-1
9	17.4	211	60	25.8	-1

In [65]:

```
crime.groupby('clusters').agg(['mean']).reset_index()
```

Out[65]:

	clusters	Murder	Assault	UrbanPop	Rape
		mean	mean	mean	mean
0	-1	11.005556	247.166667	70.666667	28.766667
1	0	14.050000	238.000000	57.750000	23.200000
2	1	4.825000	112.035714	63.357143	16.107143

In [66]:

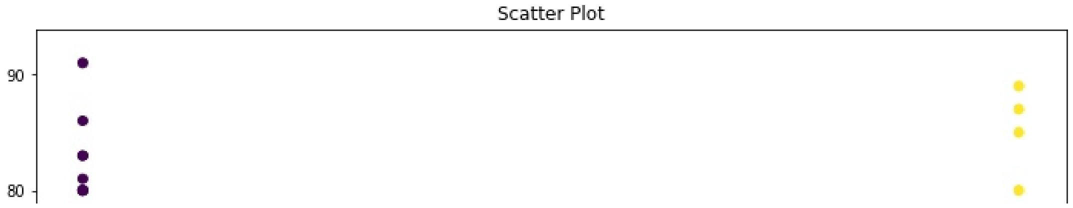
```
import matplotlib.pyplot as plt
%matplotlib inline
```

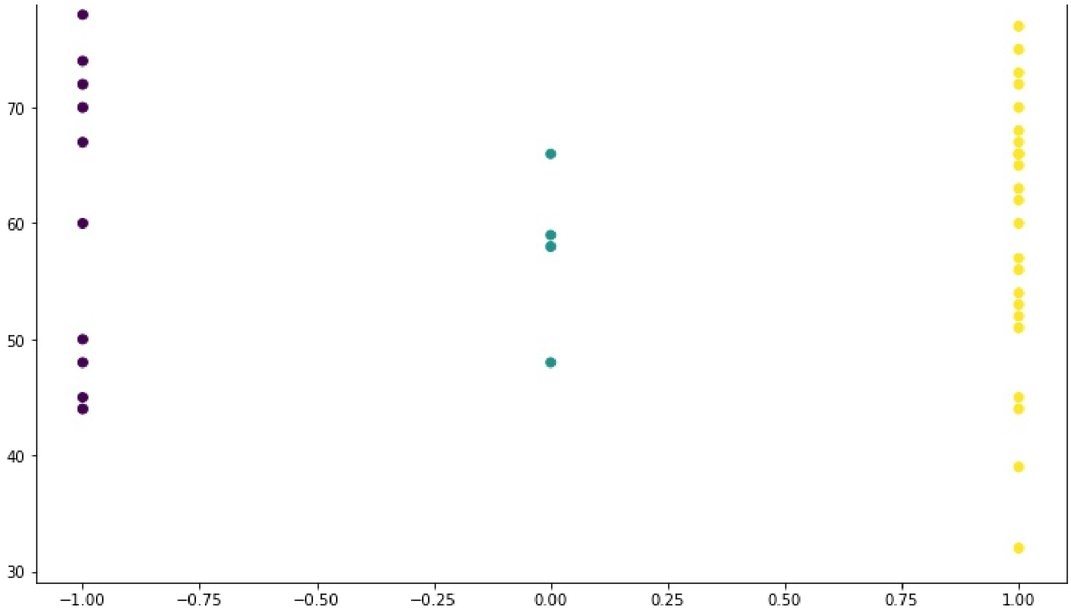
In [67]:

```
plt.figure(figsize=(12,9))
plt.title('Scatter Plot')
plt.scatter(crime['clusters'],crime['UrbanPop'], c=dbscan.labels_)
```

Out[67]:

```
<matplotlib.collections.PathCollection at 0x293effe93a0>
```





In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js