In [44]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder,StandardScaler,MinMaxScaler
from sklearn.cluster import KMeans,AgglomerativeClustering,DBSCAN
from sklearn.metrics import silhouette_score
from scipy.cluster.hierarchy import linkage
import scipy.cluster.hierarchy as sch
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
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

	Unnamed: 0	Murder	Assault	UrbanPop	Rape
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

In [3]:
```
data.shape
```

Out[3]:
```
(50, 5)
```

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

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

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

Out[5]:
```
Unnamed: 0     object
Murder        float64
Assault         int64
UrbanPop        int64
Rape          float64
dtype: object
```

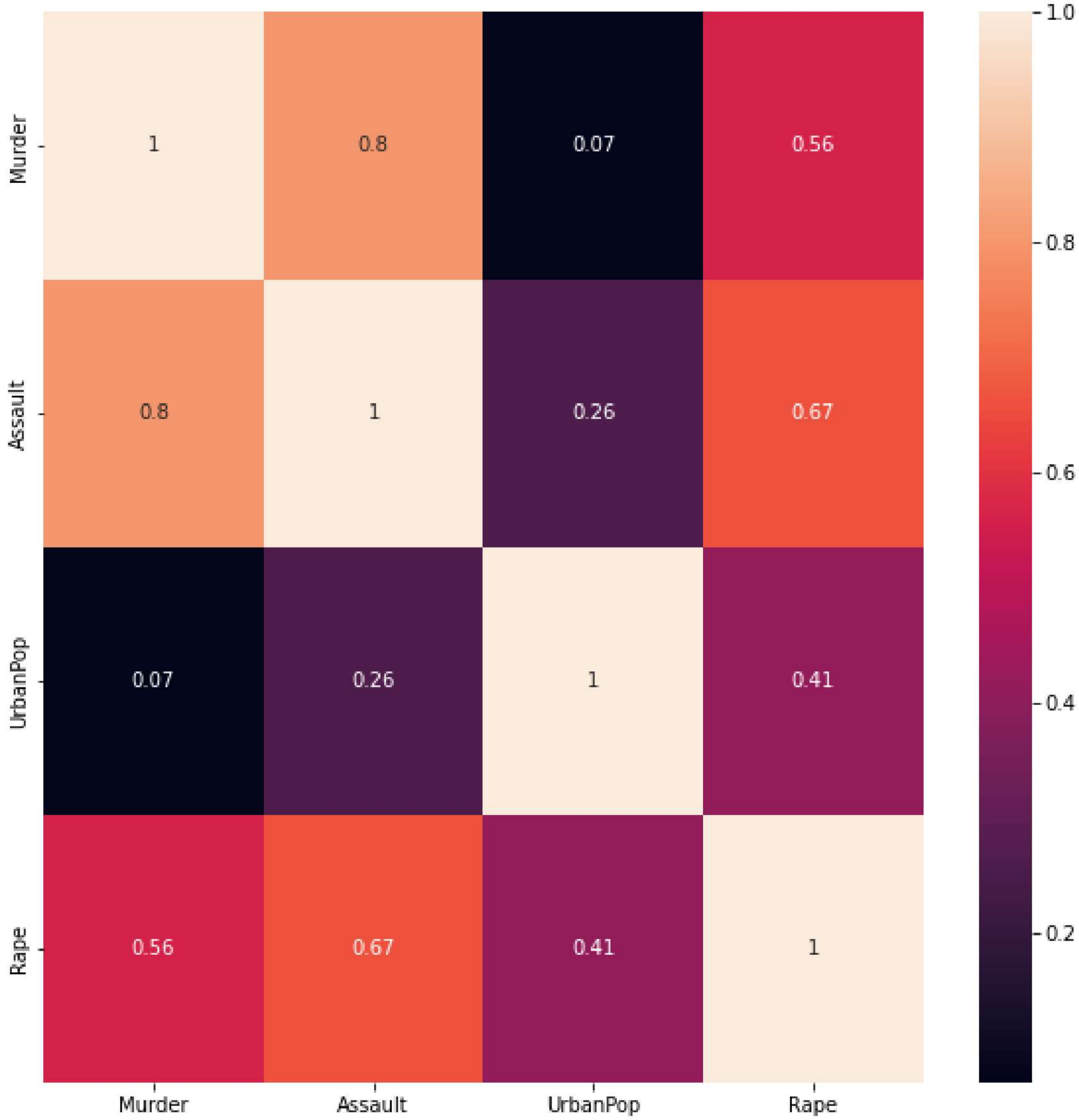
In [12]:
```
data.rename({'Unnamed: 0': 'location'},axis=1,inplace=True)
```

In [13]:
```
data.describe(include = 'all')
```

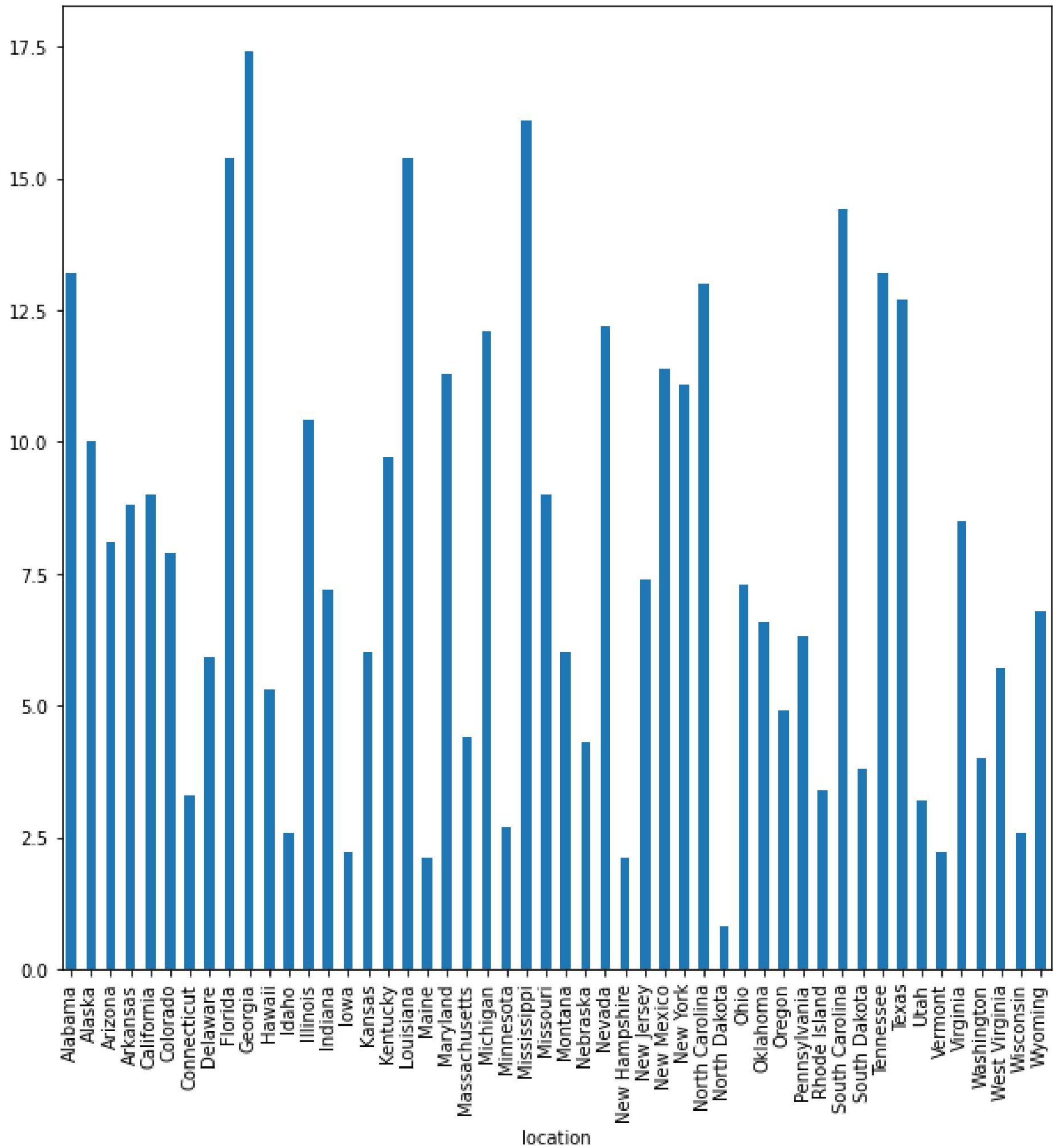
Out[13]:

	location	Murder	Assault	UrbanPop	Rape
count	50	50.00000	50.000000	50.000000	50.000000
unique	50	NaN	NaN	NaN	NaN
top	Alabama	NaN	NaN	NaN	NaN
freq	1	NaN	NaN	NaN	NaN
mean	NaN	7.78800	170.760000	65.540000	21.232000
std	NaN	4.35551	83.337661	14.474763	9.366385
min	NaN	0.80000	45.000000	32.000000	7.300000
25%	NaN	4.07500	109.000000	54.500000	15.075000
50%	NaN	7.25000	159.000000	66.000000	20.100000
75%	NaN	11.25000	249.000000	77.750000	26.175000
max	NaN	17.40000	337.000000	91.000000	46.000000

In [14]:
```
plt.figure(figsize=(10,10))
sns.heatmap(data.corr(),annot=True)
plt.show()
```

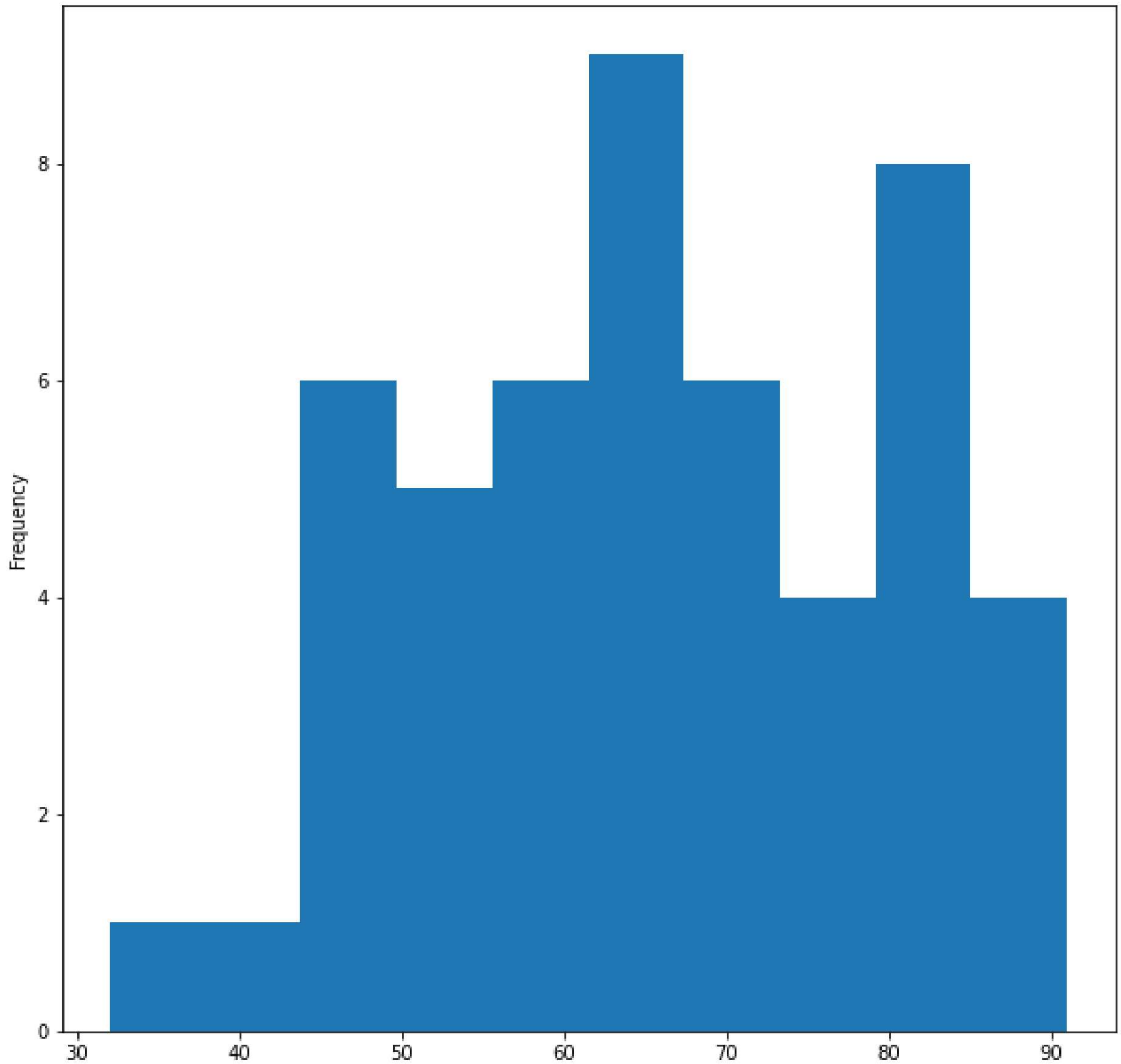


In [15]:
```
plt.figure(figsize=(10,10))
data.groupby(['location'])['Murder'].mean().plot(kind='bar')
plt.show()
```



In [18]:
```
plt.figure(figsize=(10,10))
data.groupby(['location'])['UrbanPop'].mean().plot(kind='hist')
```

Out[18]: <AxesSubplot:ylabel='Frequency'>



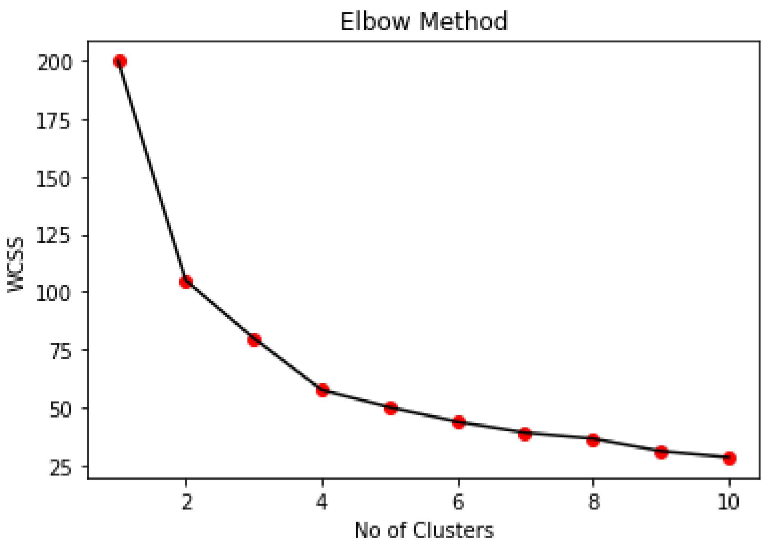
In [20]:
```
x = data.drop('location',axis = 1)
```

In [21]:
```
std = StandardScaler()
x_scaled = std.fit_transform(x)
x_scaled
```

Out[21]:
```
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 [23]:
```
wcss = []
for i in range(1,11):
    kmeans = KMeans(n_clusters=i,random_state=12)
    kmeans.fit(x_scaled)
    wcss.append(kmeans.inertia_)
```

In [24]:
```
plt.plot(range(1,11),wcss,color = 'black')
plt.scatter(range(1,11),wcss,color = 'red')
plt.title('Elbow Method')
plt.ylabel('WCSS')
plt.xlabel('No of Clusters')
plt.show()
```



In [25]:
```
crime_kmeans = KMeans(n_clusters=3, random_state=12)
crime_kmeans.fit(x_scaled)
```

Out[25]:
```
KMeans(n_clusters=3, random_state=12)
```

In [27]:
```
labels=crime_kmeans.labels_
labels
```

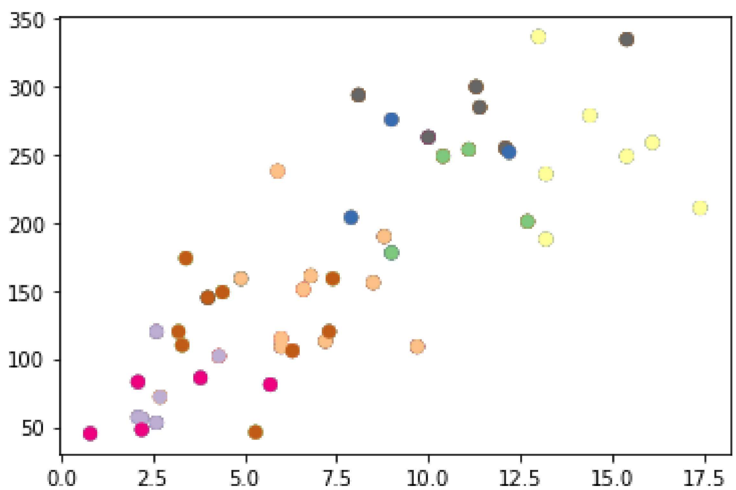
Out[27]:
```
array([1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 2, 1, 0, 2, 0, 2, 1, 2, 1, 0, 1,
       2, 1, 1, 2, 0, 1, 2, 0, 1, 1, 1, 2, 0, 0, 0, 0, 0, 1, 2, 1, 1, 0,
       2, 0, 0, 2, 2, 0])
```

In [28]:
```
labels=pd.DataFrame(data=labels)
labels.columns=['kmeans']
labels.head()
```

Out[28]:

	kmeans
0	1
1	1
2	1
3	0
4	1

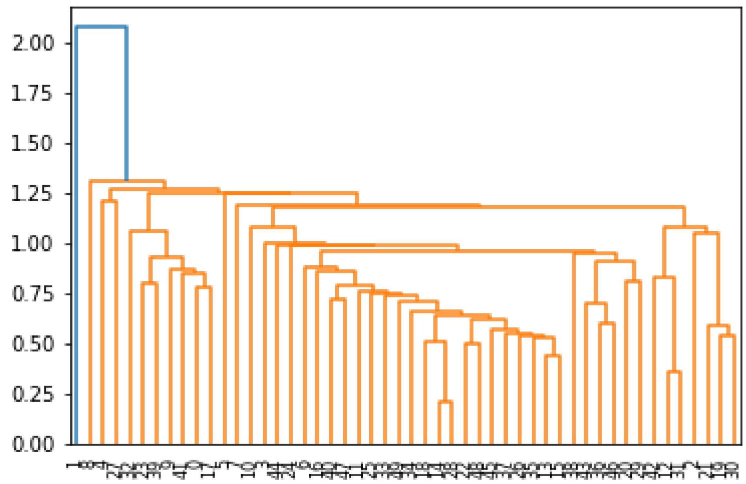
In [36]:
```
for i in range(2,10):
    model = KMeans(n_clusters=i, max_iter=600, algorithm = 'auto',init="k-means+
    model.fit(x_scaled)
    pred=model.predict(x_scaled)
    plt.scatter(data.iloc[:,1],data.iloc[:,2],c=pred,cmap=plt.cm.Accent)
```



In [37]:
```
crime_kmeans.cluster_centers_
```

Out[37]:
```
array([[-0.47137512, -0.37688855,  0.44307089, -0.27367211],
       [ 1.01513667,  1.02412028,  0.19959126,  0.85556386],
       [-0.98483178, -1.14153431, -0.99725843, -1.01543161]])
```

In [38]:
```
dendrogram = sch.dendrogram(sch.linkage(x_scaled,method='single'))
```



In [40]:
```
ıcr = AgglomerativeClustering(n_clusters=4,affinity='euclidean',linkage='single')
```

In [41]:
```
hcr
```

Out[41]:
```
AgglomerativeClustering(linkage='single', n_clusters=4)
```

In [42]:
```
y_pred_hcr = hcr.fit_predict(x_scaled)
y_pred_hcr
```

Out[42]:
```
array([0, 3, 0, 0, 1, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0], dtype=int64)
```

In [43]:
```
labels['hierarchical'] = y_pred_hcr
labels.head()
```

Out[43]:

	kmeans	hierarchical
0	1	0
1	1	3
2	1	0
3	0	0
4	1	1

In [47]:
```
db = DBSCAN(min_samples=2,eps = 0.2)
crime_db =db.fit_predict(x_scaled)
crime_db
```

Out[47]:
```
array([-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
       -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
       -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1],
      dtype=int64)
```

In [50]:
```
labels['dbscan']=crime_db
labels.head()
```

Out[50]:

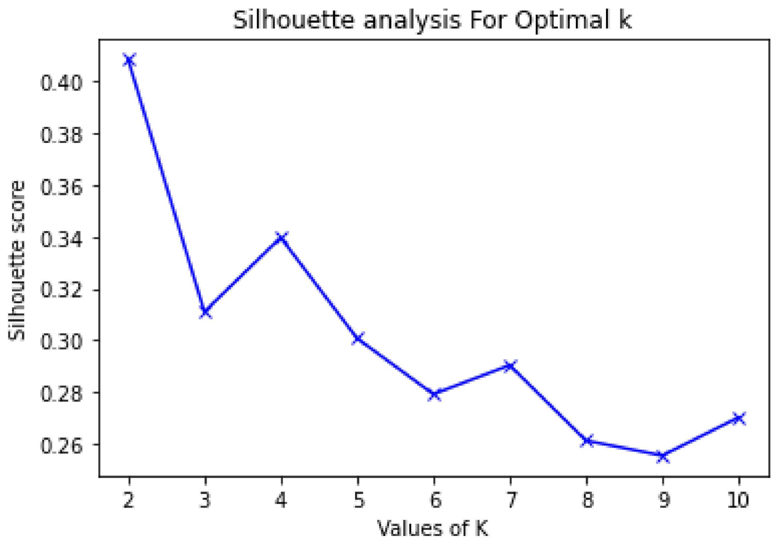
	kmeans	hierarchical	dbscan
0	1	0	-1
1	1	3	-1
2	1	0	-1
3	0	0	-1
4	1	1	-1

In [51]:
```
silhouette_avg=[]
for i in range(2,11):
    kmean = KMeans(n_clusters=i,random_state=12)
    kmean.fit(x_scaled)
    label= kmean.labels_
    silhouette_avg.append(silhouette_score(x_scaled,labels=label))
```

In [52]:
```
silhouette_avg
```

Out[52]:
```
[0.4084890326217641,
 0.3110602770365059,
 0.33968891433344395,
 0.300836812659398,
 0.279269531176561,
 0.29046780399358935,
 0.26135356563109335,
 0.25558061036269225,
 0.2703381408949321]
```

In [53]:
```
plt.plot(range(2,11),silhouette_avg,'bx-')
plt.xlabel('Values of K')
plt.ylabel('Silhouette score')
plt.title('Silhouette analysis For Optimal k')
plt.show()
```



In [ ]: