Assignment-07-Clustering-Hierarchical (Airlines)

In [1]:

```
# Import Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scipy.cluster.hierarchy as sch
from sklearn.cluster import AgglomerativeClustering
from sklearn.preprocessing import normalize
```

In [2]:

```
# Import dataset
airline=pd.read_csv("C:/Users/LENOVO/Documents/Custom Office Templates/EastWestAirlines.csv
airline
```

Out[2]:

	ID#	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus_trans
0	1	28143	0	1	1	1	174	1
1	2	19244	0	1	1	1	215	2
2	3	41354	0	1	1	1	4123	4
3	4	14776	0	1	1	1	500	1
4	5	97752	0	4	1	1	43300	26
...	...	...	...	...	...	...	...	...
3994	4017	18476	0	1	1	1	8525	4
3995	4018	64385	0	1	1	1	981	5
3996	4019	73597	0	3	1	1	25447	8
3997	4020	54899	0	1	1	1	500	1
3998	4021	3016	0	1	1	1	0	0

3999 rows × 12 columns

In [3]:

```
airline.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3999 entries, 0 to 3998
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   ID#                3999 non-null   int64
 1   Balance            3999 non-null   int64
 2   Qual_miles         3999 non-null   int64
 3   cc1_miles          3999 non-null   int64
 4   cc2_miles          3999 non-null   int64
 5   cc3_miles          3999 non-null   int64
 6   Bonus_miles        3999 non-null   int64
 7   Bonus_trans        3999 non-null   int64
 8   Flight_miles_12mo  3999 non-null   int64
 9   Flight_trans_12    3999 non-null   int64
 10  Days_since_enroll  3999 non-null   int64
 11  Award?             3999 non-null   int64
dtypes: int64(12)
memory usage: 375.0 KB
```

In [4]:

```
airline2=airline.drop(['ID#'],axis=1)
airline2
```

Out[4]:

	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus_trans	Flight
0	28143	0	1	1	1	174	1	
1	19244	0	1	1	1	215	2	
2	41354	0	1	1	1	4123	4	
3	14776	0	1	1	1	500	1	
4	97752	0	4	1	1	43300	26	
...	...	...	...	...	...	...	...	
3994	18476	0	1	1	1	8525	4	
3995	64385	0	1	1	1	981	5	
3996	73597	0	3	1	1	25447	8	
3997	54899	0	1	1	1	500	1	
3998	3016	0	1	1	1	0	0	

3999 rows × 11 columns

In [5]:

```
# Normalize hetrogenous numarical data
airline2_norm=pd.DataFrame(normalize(airline2),columns=airline2.columns)
airline2_norm
```

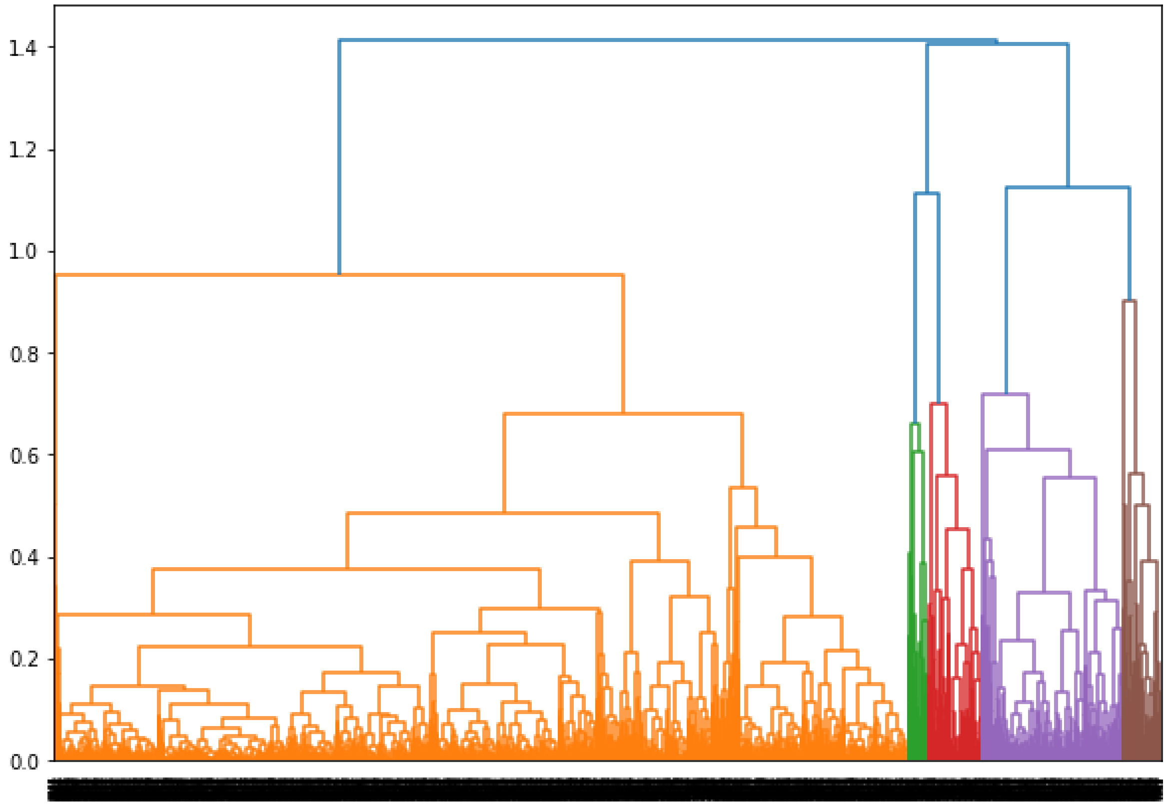
Out[5]:

	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus_trans	Fligh
0	0.970414	0.0	0.000034	0.000034	0.000034	0.006000	0.000034	
1	0.940209	0.0	0.000049	0.000049	0.000049	0.010504	0.000098	
2	0.981113	0.0	0.000024	0.000024	0.000024	0.097817	0.000095	
3	0.904428	0.0	0.000061	0.000061	0.000061	0.030605	0.000061	
4	0.912226	0.0	0.000037	0.000009	0.000009	0.404078	0.000243	
...	...	...	...	...	...	...	...	
3994	0.905810	0.0	0.000049	0.000049	0.000049	0.417949	0.000196	
3995	0.999649	0.0	0.000016	0.000016	0.000016	0.015231	0.000078	
3996	0.944948	0.0	0.000039	0.000013	0.000013	0.326726	0.000103	
3997	0.999592	0.0	0.000018	0.000018	0.000018	0.009104	0.000018	
3998	0.907271	0.0	0.000301	0.000301	0.000301	0.000000	0.000000	

3999 rows × 11 columns

In [6]:

```
# Create Dendograms
plt.figure(figsize=(10,7))
dendograms=sch.dendrogram(sch.linkage(airline2_norm,'complete'))
```



In [7]:

```
# Create Clusters (y)
hclusters=AgglomerativeClustering(n_clusters=5,affinity='euclidean',linkage='ward')
hclusters
```

Out[7]:

```
AgglomerativeClustering(n_clusters=5)
```

In [8]:

```
y=pd.DataFrame(hclusters.fit_predict(airline2_norm),columns=['clustersid'])
y['clustersid'].value_counts()
```

Out[8]:

```
2    1547
4    1191
3     579
1     453
0     229
Name: clustersid, dtype: int64
```

In [9]:

```
# Adding clusters to dataset
airline2['clustersid']=hclusters.labels_
airline2
```

Out[9]:

	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus_trans	Flight
0	28143	0	1	1	1	174	1	
1	19244	0	1	1	1	215	2	
2	41354	0	1	1	1	4123	4	
3	14776	0	1	1	1	500	1	
4	97752	0	4	1	1	43300	26	
...	...	...	...	...	...	...	...	
3994	18476	0	1	1	1	8525	4	
3995	64385	0	1	1	1	981	5	
3996	73597	0	3	1	1	25447	8	
3997	54899	0	1	1	1	500	1	
3998	3016	0	1	1	1	0	0	

3999 rows × 12 columns

In [10]:

```
airline2.groupby('clustersid').agg(['mean']).reset_index()
```

Out[10]:

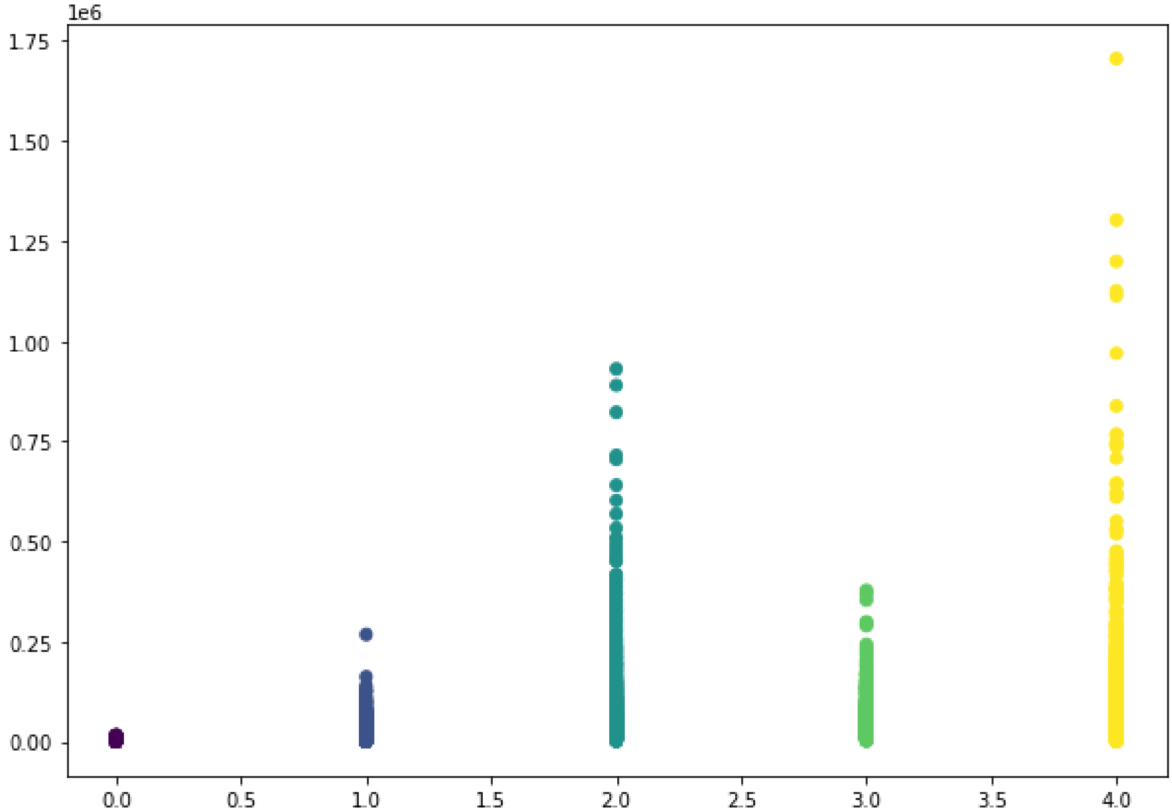
	clustersid	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus
		mean	mean	mean	mean	mean	mean	mean
0	0	5524.222707	8.755459	1.000000	1.000000	1.000000	584.532751	2.4
1	1	31066.514349	111.415011	3.200883	1.026490	1.070640	40266.935982	17.2
2	2	81201.080802	136.521008	2.115061	1.013575	1.000646	16350.149968	13.5
3	3	69569.894646	97.257340	3.326425	1.032815	1.022453	35743.675302	17.7
4	4	94957.590260	215.220823	1.141058	1.005038	1.002519	3524.928631	5.6

In [11]:

```
# Plot Clusters
plt.figure(figsize=(10,7))
plt.scatter(airline2['clustersid'],airline2['Balance'],c=hclusters.labels_)
```

Out[11]:

```
<matplotlib.collections.PathCollection at 0xe10387be20>
```



In [ ]:

```

```

Assignment-07-K-Means Clustering (Airlines)

In [12]:

```
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
```

In [13]:

```
# import dataset
airline1=pd.read_csv("C:/Users/LENOVO/Documents/Custom Office Templates/EastWestAirlines.cs
airline1
```

Out[13]:

	ID#	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus_trans
0	1	28143	0	1	1	1	174	1
1	2	19244	0	1	1	1	215	2
2	3	41354	0	1	1	1	4123	4
3	4	14776	0	1	1	1	500	1
4	5	97752	0	4	1	1	43300	26
...	...	...	...	...	...	...	...	...
3994	4017	18476	0	1	1	1	8525	4
3995	4018	64385	0	1	1	1	981	5
3996	4019	73597	0	3	1	1	25447	8
3997	4020	54899	0	1	1	1	500	1
3998	4021	3016	0	1	1	1	0	0

3999 rows × 12 columns

In [14]:

```
airline2=airline1.drop(['ID#'],axis=1)
airline2
```

Out[14]:

	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus_trans	Flight_
0	28143	0	1	1	1	174	1	
1	19244	0	1	1	1	215	2	
2	41354	0	1	1	1	4123	4	
3	14776	0	1	1	1	500	1	
4	97752	0	4	1	1	43300	26	
...	...	...	...	...	...	...	...	
3994	18476	0	1	1	1	8525	4	
3995	64385	0	1	1	1	981	5	
3996	73597	0	3	1	1	25447	8	
3997	54899	0	1	1	1	500	1	
3998	3016	0	1	1	1	0	0	

3999 rows × 11 columns

In [15]:

```
airline2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3999 entries, 0 to 3998
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Balance            3999 non-null   int64
 1   Qual_miles         3999 non-null   int64
 2   cc1_miles          3999 non-null   int64
 3   cc2_miles          3999 non-null   int64
 4   cc3_miles          3999 non-null   int64
 5   Bonus_miles        3999 non-null   int64
 6   Bonus_trans        3999 non-null   int64
 7   Flight_miles_12mo  3999 non-null   int64
 8   Flight_trans_12    3999 non-null   int64
 9   Days_since_enroll  3999 non-null   int64
 10  Award?             3999 non-null   int64
dtypes: int64(11)
memory usage: 343.8 KB
```

In [16]:

```
# Normalize hetrogenous numerical data by using Standard Scaler
airline2_norm=StandardScaler().fit_transform(airline2)
```

In [17]:

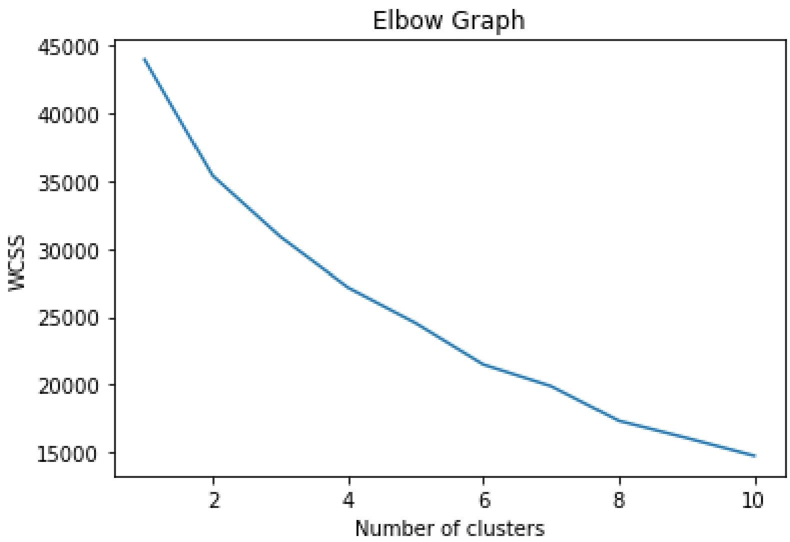
```
# Use Elbow Graph to find optimum number of clusters (K value) from K values range
# The K-means algorithm aims to choose centroids that minimise the inertia, or within-clust
# random state can be anything from 0 to 42, but the same number to be used everytime, so t
```

In [18]:

```
# within-cluster sum-of-squares criterion
wcss=[]
for i in range (1,11):
    kmeans=KMeans(n_clusters=i, random_state=2)
    kmeans.fit(airline2_norm)
    wcss.append(kmeans.inertia_)
```

In [19]:

```
# Plot K values range vs WCSS to get Elbow graph for choosing K (no. of clusters)
plt.plot(range(1,11),wcss)
plt.title('Elbow Graph')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



Build Cluster algorithm using K=4

In [20]:

```
# cluster algorithm using K=4
clusters4=KMeans(4,random_state=30).fit(airline2_norm)
clusters4
```

Out[20]:

```
KMeans(n_clusters=4, random_state=30)
```

In [21]:

```
clusters4.labels_
```

Out[21]:

```
array([0, 0, 0, ..., 3, 0, 0])
```

In [22]:

```
# Assign clusters to the data set
airline3=airline2.copy()
airline3['clusters4id']=clusters4.labels_
airline3
```

Out[22]:

	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus_trans	Flight
0	28143	0	1	1	1	174	1	
1	19244	0	1	1	1	215	2	
2	41354	0	1	1	1	4123	4	
3	14776	0	1	1	1	500	1	
4	97752	0	4	1	1	43300	26	
...	...	...	...	...	...	...	...	
3994	18476	0	1	1	1	8525	4	
3995	64385	0	1	1	1	981	5	
3996	73597	0	3	1	1	25447	8	
3997	54899	0	1	1	1	500	1	
3998	3016	0	1	1	1	0	0	

3999 rows × 12 columns

In [23]:

```
# Compute the centroids for K=4 clusters with 11 variables
clusters4.cluster_centers_
```

Out[23]:

```
array([[-0.29584258, -0.0603903 , -0.60911906,  0.03222463, -0.06075301,
        -0.51562007, -0.48778976, -0.18468409, -0.19727389, -0.2065444 ,
        -0.34933297],
       [ 0.63971926, -0.08443292,  1.0220844 , -0.09824189, 15.64629931,
         3.17969131,  1.71461374,  0.03329269,  0.05969539,  0.23987261,
         0.33752735],
       [ 1.19916278,  0.8413837 ,  0.07934291,  0.15576844, -0.06276658,
         0.61091878,  1.63802866,  3.57547132,  3.86140846,  0.28565421,
         0.91563614],
       [ 0.43059508,  0.0158422 ,  1.18872871, -0.08236624, -0.05476264,
         0.91116803,  0.74463543, -0.08026444, -0.09129375,  0.37198921,
         0.57588937]])
```

In [24]:

```
# Group data by Clusters (K=4)
airline3.groupby('clusters4id').agg(['mean']).reset_index()
```

Out[24]:

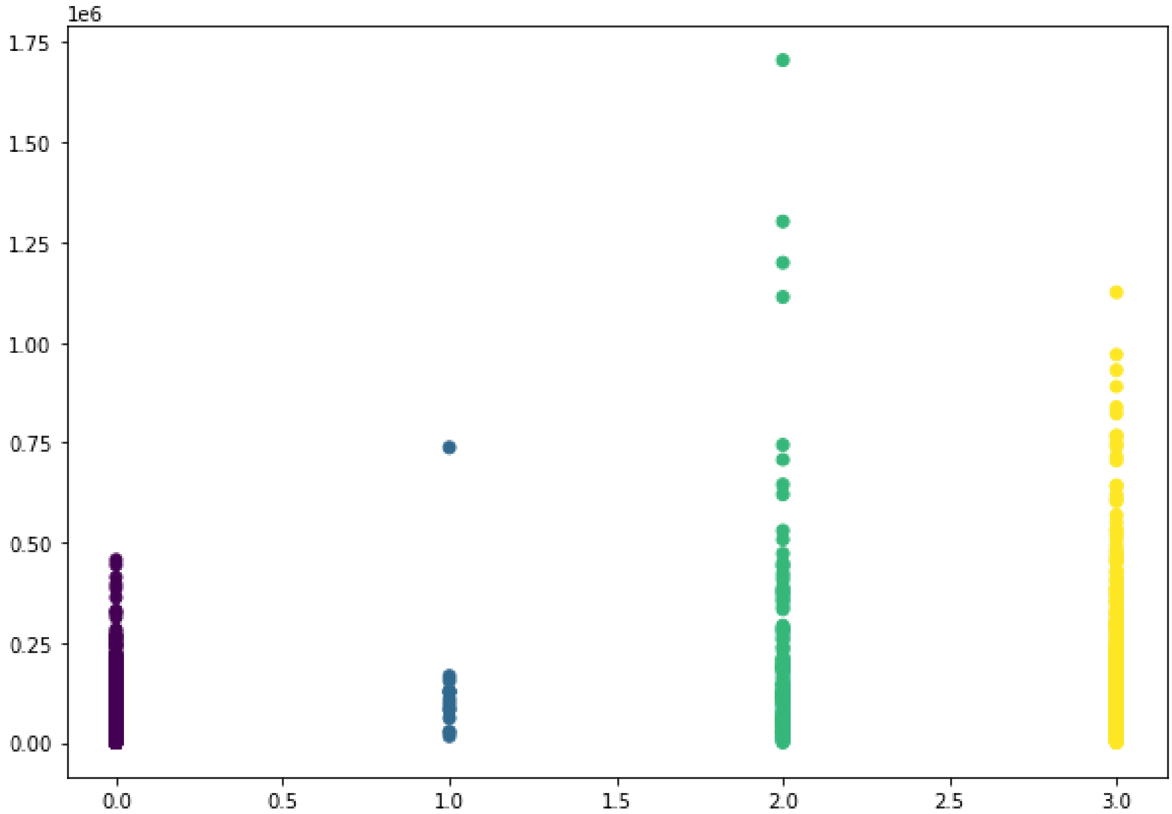
	clusters4id	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bon
		mean	mean	mean	mean	mean	mean	mea
0	0	43828.396152	97.283863	1.223007	1.019238	1.000393	4707.805654	
1	1	138061.400000	78.800000	3.466667	1.000000	4.066667	93927.866667	2
2	2	194432.643750	794.981250	2.168750	1.037500	1.000000	31897.281250	2
3	3	117087.423649	156.736883	3.697729	1.002349	1.001566	39200.451057	1

In [25]:

```
# Plot Clusters
plt.figure(figsize=(10,7))
plt.scatter(airline3['clusters4id'],airline3['Balance'], c=clusters4.labels_)
```

Out[25]:

<matplotlib.collections.PathCollection at 0xe10e856cd0>



Build Cluster algorithm using K=5

In [26]:

```
# cluster algorithm using K=5
clusters5=KMeans(5,random_state=30).fit(airline2_norm)
clusters5
```

Out[26]:

```
KMeans(n_clusters=5, random_state=30)
```

In [27]:

```
clusters5.labels_
```

Out[27]:

```
array([0, 0, 0, ..., 3, 0, 0])
```

In [28]:

```
# Assign clusters to the data set
airline4=airline2.copy()
airline4['clusters5id']=clusters5.labels_
airline4
```

Out[28]:

	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus_trans	Flight_
0	28143	0	1	1	1	174	1	
1	19244	0	1	1	1	215	2	
2	41354	0	1	1	1	4123	4	
3	14776	0	1	1	1	500	1	
4	97752	0	4	1	1	43300	26	
...	...	...	...	...	...	...	...	
3994	18476	0	1	1	1	8525	4	
3995	64385	0	1	1	1	981	5	
3996	73597	0	3	1	1	25447	8	
3997	54899	0	1	1	1	500	1	
3998	3016	0	1	1	1	0	0	

3999 rows × 12 columns

In [29]:

```
# Compute the centroids for K=5 clusters with 11 variables
clusters5.cluster_centers_
```

Out[29]:

```
array([[-2.97441370e-01, -6.26013958e-02, -6.09944285e-01,
        -9.82418871e-02, -6.07159307e-02, -5.21330789e-01,
        -5.01454441e-01, -1.87180704e-01, -1.99831577e-01,
        -2.10475283e-01, -3.52420208e-01],
       [ 1.14970340e+00,  1.06407580e+00,  1.02900923e-01,
        -9.82418871e-02, -6.27665798e-02,  5.78252911e-01,
         1.52034538e+00,  3.44900129e+00,  3.66768718e+00,
         2.64756357e-01,  8.99553448e-01],
       [ 6.39719256e-01, -8.44329231e-02,  1.02208440e+00,
        -9.82418871e-02,  1.56462993e+01,  3.17969131e+00,
         1.71461374e+00,  3.32926913e-02,  5.96953922e-02,
         2.39872612e-01,  3.37527346e-01],
       [ 4.24748143e-01, -1.21396375e-02,  1.19282031e+00,
        -9.82418871e-02, -5.47249449e-02,  9.11488076e-01,
         7.40508649e-01, -9.38523027e-02, -1.02849450e-01,
         3.77190968e-01,  5.65958814e-01],
       [-4.68896637e-02, -1.56235600e-01, -6.68227273e-01,
         9.03825361e+00, -6.27665798e-02, -1.01665326e-01,
         6.17851143e-01,  8.75493989e-02,  2.20346809e-01,
        -7.24639805e-02,  5.17838824e-02]])
```

In [30]:

```
# Group data by Clusters (K=5)
airline4.groupby('clusters5id').agg(['mean']).reset_index()
```

Out[30]:

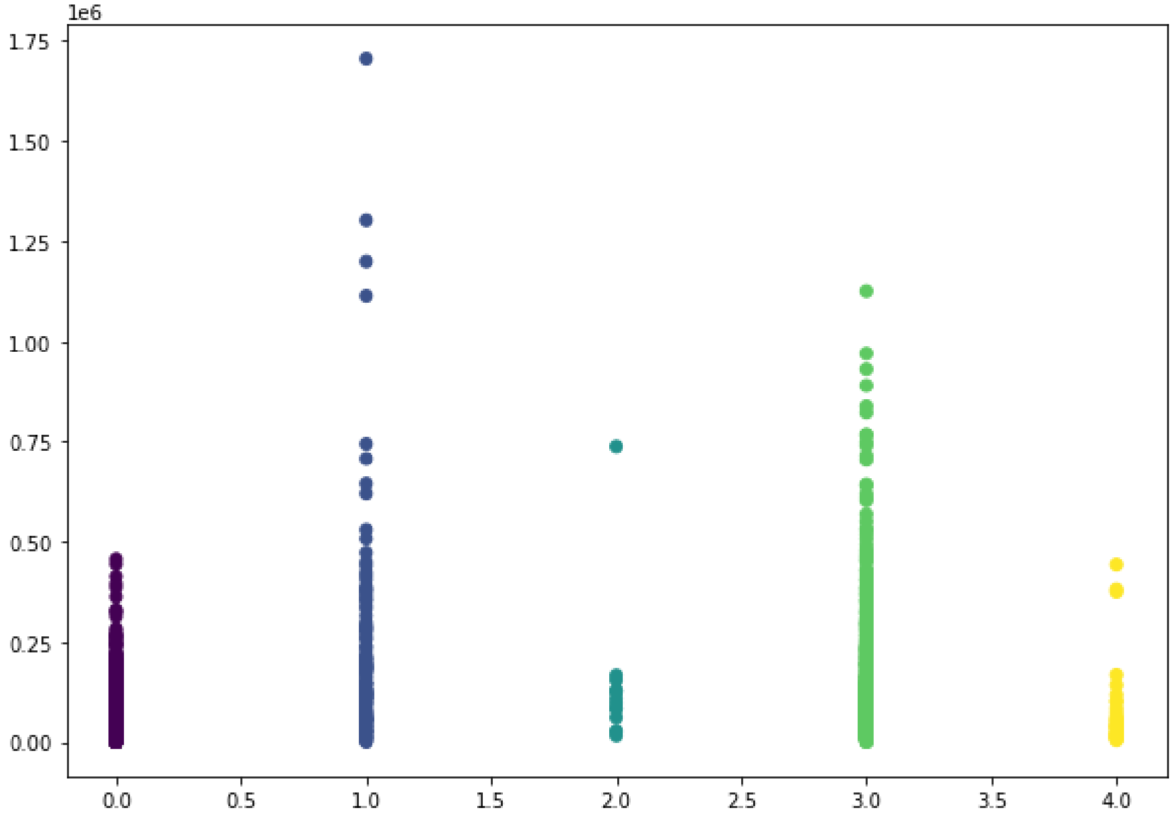
	clusters5id	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bon
		mean	mean	mean	mean	mean	mean	mea
0	0	43669.876000	95.611600	1.221200	1.000000	1.000400	4568.151200	
1	1	189448.964497	967.248521	2.201183	1.000000	1.000000	31108.467456	2
2	2	138061.400000	78.800000	3.466667	1.000000	4.066667	93927.866667	2
3	3	116436.737421	134.935535	3.702830	1.000000	1.001572	39185.495283	1
4	4	68876.581395	23.255814	1.139535	2.348837	1.000000	14689.837209	1

In [31]:

```
# Plot Clusters
plt.figure(figsize=(10,7))
plt.scatter(airline4['clusters5id'],airline4['Balance'], c=clusters5.labels_)
```

Out[31]:

```
<matplotlib.collections.PathCollection at 0xe10e8bb400>
```



In [ ]:

```

```

Assignment-07-DBSCAN Clustering (Airlines)

In [32]:

```
from sklearn.cluster import DBSCAN
```

In [33]:

```
# import dataset
airline=pd.read_csv("C:/Users/LENOVO/Documents/Custom Office Templates/EastWestAirlines.csv
airline
```

Out[33]:

	ID#	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus_trans
0	1	28143	0	1	1	1	174	1
1	2	19244	0	1	1	1	215	2
2	3	41354	0	1	1	1	4123	4
3	4	14776	0	1	1	1	500	1
4	5	97752	0	4	1	1	43300	26
...	...	...	...	...	...	...	...	...
3994	4017	18476	0	1	1	1	8525	4
3995	4018	64385	0	1	1	1	981	5
3996	4019	73597	0	3	1	1	25447	8
3997	4020	54899	0	1	1	1	500	1
3998	4021	3016	0	1	1	1	0	0

3999 rows × 12 columns

In [34]:

```
airline.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3999 entries, 0 to 3998
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   ID#                3999 non-null   int64
 1   Balance            3999 non-null   int64
 2   Qual_miles         3999 non-null   int64
 3   cc1_miles          3999 non-null   int64
 4   cc2_miles          3999 non-null   int64
 5   cc3_miles          3999 non-null   int64
 6   Bonus_miles        3999 non-null   int64
 7   Bonus_trans        3999 non-null   int64
 8   Flight_miles_12mo  3999 non-null   int64
 9   Flight_trans_12    3999 non-null   int64
 10  Days_since_enroll  3999 non-null   int64
 11  Award?             3999 non-null   int64
dtypes: int64(12)
memory usage: 375.0 KB
```

In [35]:

```
airline2=airline.drop(['ID#'],axis=1)
airline2
```

Out[35]:

	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus_trans	Flig
0	28143	0	1	1	1	174	1	
1	19244	0	1	1	1	215	2	
2	41354	0	1	1	1	4123	4	
3	14776	0	1	1	1	500	1	
4	97752	0	4	1	1	43300	26	
...	...	...	...	...	...	...	...	
3994	18476	0	1	1	1	8525	4	
3995	64385	0	1	1	1	981	5	
3996	73597	0	3	1	1	25447	8	
3997	54899	0	1	1	1	500	1	
3998	3016	0	1	1	1	0	0	

3999 rows × 11 columns

In [36]:

```
# Normalize hetrogenous numerical data using Standard Scaler fit transform to dataset
airline2_norm=StandardScaler().fit_transform(airline2)
airline2_norm
```

Out[36]:

```
array([[-4.51140783e-01, -1.86298687e-01, -7.69578406e-01, ...,
        -3.62167870e-01,  1.39545434e+00, -7.66919299e-01],
       [-5.39456874e-01, -1.86298687e-01, -7.69578406e-01, ...,
        -3.62167870e-01,  1.37995704e+00, -7.66919299e-01],
       [-3.20031232e-01, -1.86298687e-01, -7.69578406e-01, ...,
        -3.62167870e-01,  1.41192021e+00, -7.66919299e-01],
       ...,
       [-4.29480975e-05, -1.86298687e-01,  6.83121167e-01, ...,
        -3.62167870e-01, -1.31560393e+00,  1.30391816e+00],
       [-1.85606976e-01, -1.86298687e-01, -7.69578406e-01, ...,
        -9.85033311e-02, -1.31608822e+00, -7.66919299e-01],
       [-7.00507951e-01, -1.86298687e-01, -7.69578406e-01, ...,
        -3.62167870e-01, -1.31754109e+00, -7.66919299e-01]])
```

In [37]:

```
# DBSCAN Clustering
dbscan=DBSCAN(eps=1,min_samples=11)
dbscan.fit(airline2_norm)
```

Out[37]:

```
DBSCAN(eps=1, min_samples=11)
```

In [38]:

```
# Noisy samples are given the label -1.
dbscan.labels_
```

Out[38]:

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

In [39]:

```
# Adding Clusters to dataset
airline2['clusters']=dbscan.labels_
airline2
```

Out[39]:

	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus_trans	Flight_
0	28143	0	1	1	1	174	1	
1	19244	0	1	1	1	215	2	
2	41354	0	1	1	1	4123	4	
3	14776	0	1	1	1	500	1	
4	97752	0	4	1	1	43300	26	
...	...	...	...	...	...	...	...	
3994	18476	0	1	1	1	8525	4	
3995	64385	0	1	1	1	981	5	
3996	73597	0	3	1	1	25447	8	
3997	54899	0	1	1	1	500	1	
3998	3016	0	1	1	1	0	0	

3999 rows × 12 columns

In [40]:

```
airline2.groupby('clusters').agg(['mean']).reset_index()
```

Out[40]:

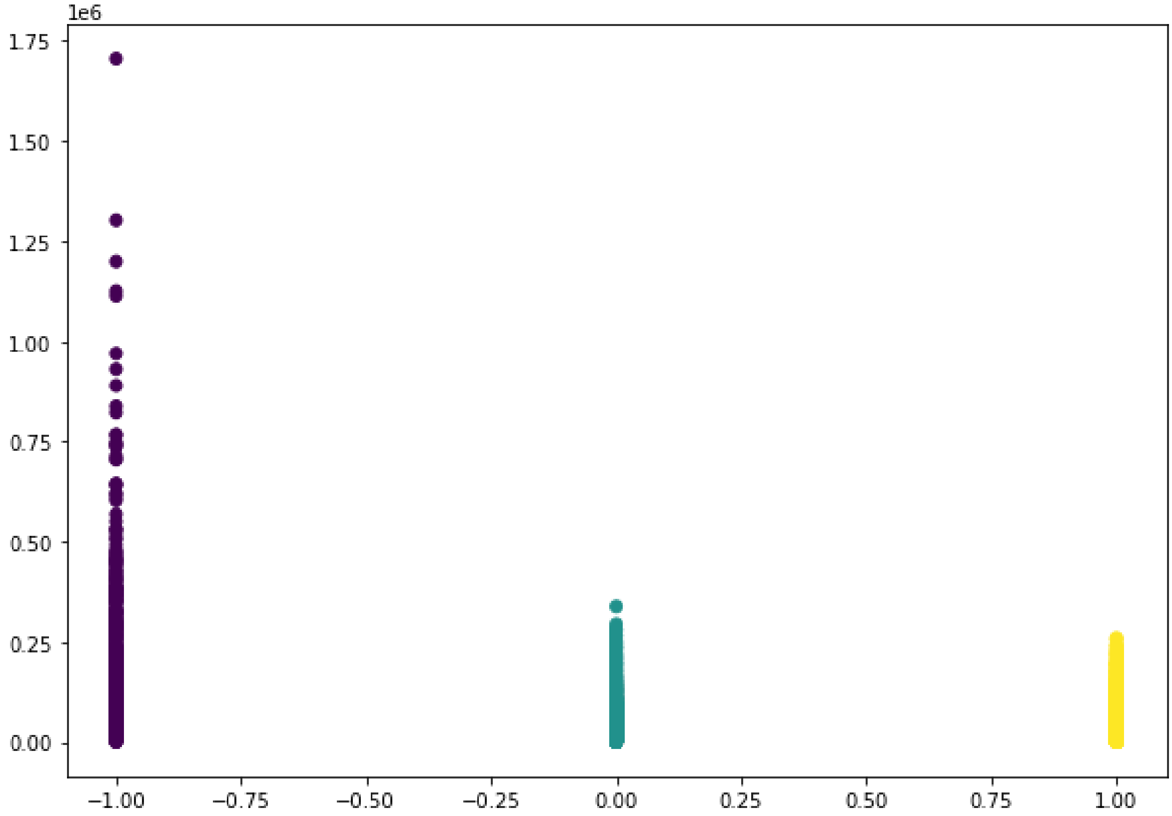
	clusters	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus_
		mean	mean	mean	mean	mean	mean	mean
0	-1	167974.051576	808.249284	2.598854	1.083095	1.070201	36208.904011	20.6
1	0	52023.848218	2.256929	1.652002	1.000000	1.000000	8893.520458	8.3
2	1	57233.087549	6.834630	2.594358	1.000000	1.000000	22444.993191	12.7

In [41]:

```
# plot clusters
plt.figure(figsize=(10,7))
plt.scatter(airline2['clusters'],airline2['Balance'],c=dbscan.labels_)
```

Out[41]:

```
<matplotlib.collections.PathCollection at 0xe10e3ec2e0>
```



In [ ]:

Assignment-07-Clustering-Hierarchical (Crimes)

In [42]:

```
crime1=pd.read_csv("C:/Users/LENOVO/Documents/Custom Office Templates/crime_data.csv")
crime1
```

Out[42]:

	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

	Unnamed: 0	Murder	Assault	UrbanPop	Rape
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 [43]:

```
crime1.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 [44]:

```
crime2=crime1.drop(['Unnamed: 0'],axis=1)
crime2
```

Out[44]:

	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
5	7.9	204	78	38.7
6	3.3	110	77	11.1
7	5.9	238	72	15.8
8	15.4	335	80	31.9
9	17.4	211	60	25.8
10	5.3	46	83	20.2
11	2.6	120	54	14.2
12	10.4	249	83	24.0
13	7.2	113	65	21.0
14	2.2	56	57	11.3
15	6.0	115	66	18.0
16	9.7	109	52	16.3
17	15.4	249	66	22.2
18	2.1	83	51	7.8
19	11.3	300	67	27.8
20	4.4	149	85	16.3
21	12.1	255	74	35.1
22	2.7	72	66	14.9
23	16.1	259	44	17.1
24	9.0	178	70	28.2
25	6.0	109	53	16.4
26	4.3	102	62	16.5
27	12.2	252	81	46.0
28	2.1	57	56	9.5
29	7.4	159	89	18.8
30	11.4	285	70	32.1
31	11.1	254	86	26.1
32	13.0	337	45	16.1
33	0.8	45	44	7.3

	Murder	Assault	UrbanPop	Rape
34	7.3	120	75	21.4
35	6.6	151	68	20.0
36	4.9	159	67	29.3
37	6.3	106	72	14.9
38	3.4	174	87	8.3
39	14.4	279	48	22.5
40	3.8	86	45	12.8
41	13.2	188	59	26.9
42	12.7	201	80	25.5
43	3.2	120	80	22.9
44	2.2	48	32	11.2
45	8.5	156	63	20.7
46	4.0	145	73	26.2
47	5.7	81	39	9.3
48	2.6	53	66	10.8
49	6.8	161	60	15.6

In [49]:

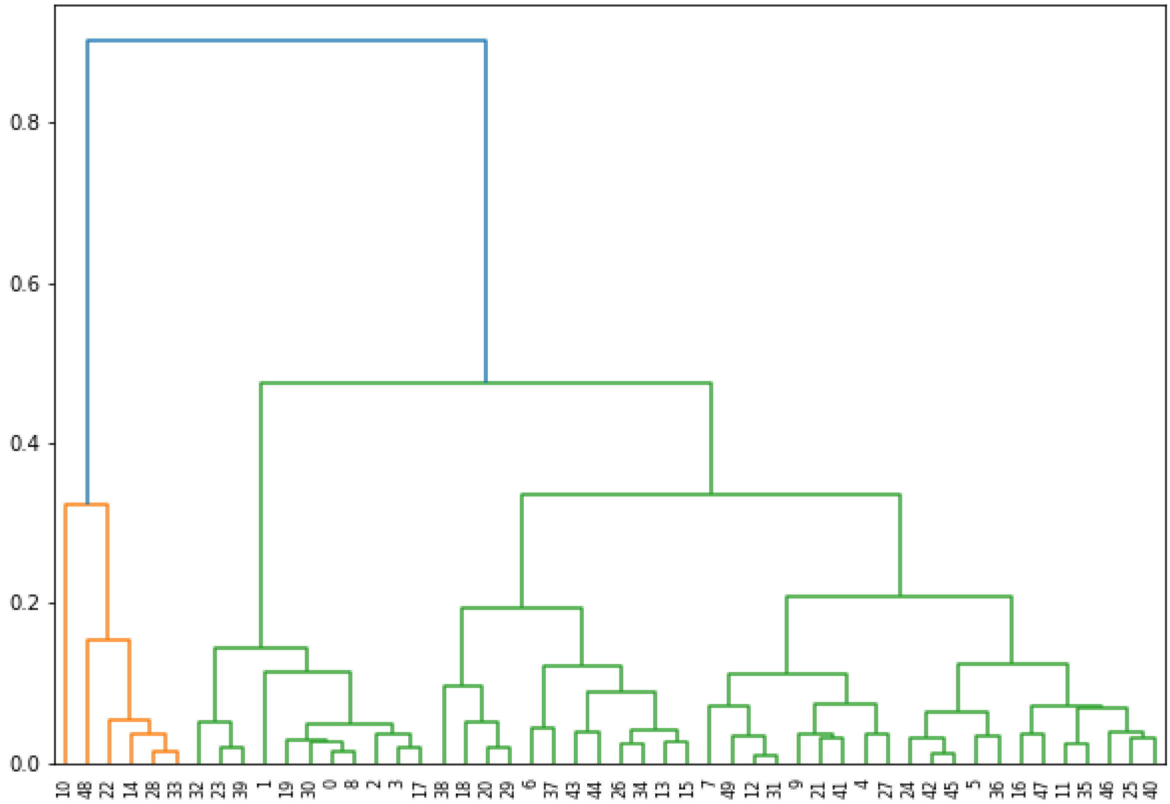
```
# Normalize hetrogenous numerical data
crime2_norm=pd.DataFrame(normalize(crime2),columns=crime2.columns)
crime2_norm
```

Out[49]:

	Murder	Assault	UrbanPop	Rape
0	0.054031	0.966016	0.237411	0.086778
1	0.036872	0.969739	0.176987	0.164081
2	0.026439	0.959624	0.261122	0.101185
3	0.044528	0.961392	0.252998	0.098669
4	0.030657	0.940134	0.309972	0.138295
5	0.035594	0.919142	0.351437	0.174367
6	0.024486	0.816202	0.571341	0.082362
7	0.023674	0.954965	0.288897	0.063397
8	0.044478	0.967547	0.231056	0.092134
9	0.078534	0.952332	0.270805	0.116446

In [50]:

```
# Create Dendograms
plt.figure(figsize=(10,7))
dendograms=sch.dendrogram(sch.linkage(crime2_norm,'complete'))
```



In [51]:

```
# Create Clusters (y)
hclusters=AgglomerativeClustering(n_clusters=3,affinity='euclidean',linkage='ward')
hclusters
```

Out[51]:

```
AgglomerativeClustering(n_clusters=3)
```

In [52]:

```
y=pd.DataFrame(hclusters.fit_predict(crime2_norm),columns=['clustersid'])
y['clustersid'].value_counts()
```

Out[52]:

```
0    24
2    20
1     6
Name: clustersid, dtype: int64
```

In [53]:

```
# Adding clusters to dataset
crime2['clustersid']=hclusters.labels_
crime2
```

Out[53]:

	Murder	Assault	UrbanPop	Rape	clustersid
0	13.2	236	58	21.2	2
1	10.0	263	48	44.5	2
2	8.1	294	80	31.0	2
3	8.8	190	50	19.5	2
4	9.0	276	91	40.6	2
5	7.9	204	78	38.7	0
6	3.3	110	77	11.1	0
7	5.9	238	72	15.8	2
8	15.4	335	80	31.9	2
9	17.4	211	60	25.8	2
10	5.3	46	83	20.2	1
11	2.6	120	54	14.2	0
12	10.4	249	83	24.0	2
13	7.2	113	65	21.0	0
14	2.2	56	57	11.3	1
15	6.0	115	66	18.0	0
16	9.7	109	52	16.3	0
17	15.4	249	66	22.2	2
18	2.1	83	51	7.8	0
19	11.3	300	67	27.8	2
20	4.4	149	85	16.3	0
21	12.1	255	74	35.1	2
22	2.7	72	66	14.9	1
23	16.1	259	44	17.1	2
24	9.0	178	70	28.2	0
25	6.0	109	53	16.4	0
26	4.3	102	62	16.5	0
27	12.2	252	81	46.0	2
28	2.1	57	56	9.5	1
29	7.4	159	89	18.8	0
30	11.4	285	70	32.1	2
31	11.1	254	86	26.1	2
32	13.0	337	45	16.1	2

	Murder	Assault	UrbanPop	Rape	clustersid
33	0.8	45	44	7.3	1
34	7.3	120	75	21.4	0
35	6.6	151	68	20.0	0
36	4.9	159	67	29.3	0
37	6.3	106	72	14.9	0
38	3.4	174	87	8.3	0
39	14.4	279	48	22.5	2
40	3.8	86	45	12.8	0
41	13.2	188	59	26.9	2
42	12.7	201	80	25.5	0
43	3.2	120	80	22.9	0
44	2.2	48	32	11.2	0
45	8.5	156	63	20.7	0
46	4.0	145	73	26.2	0
47	5.7	81	39	9.3	0
48	2.6	53	66	10.8	1
49	6.8	161	60	15.6	2

In [54]:

```
crime2.groupby('clustersid').agg(['mean']).reset_index()
```

Out[54]:

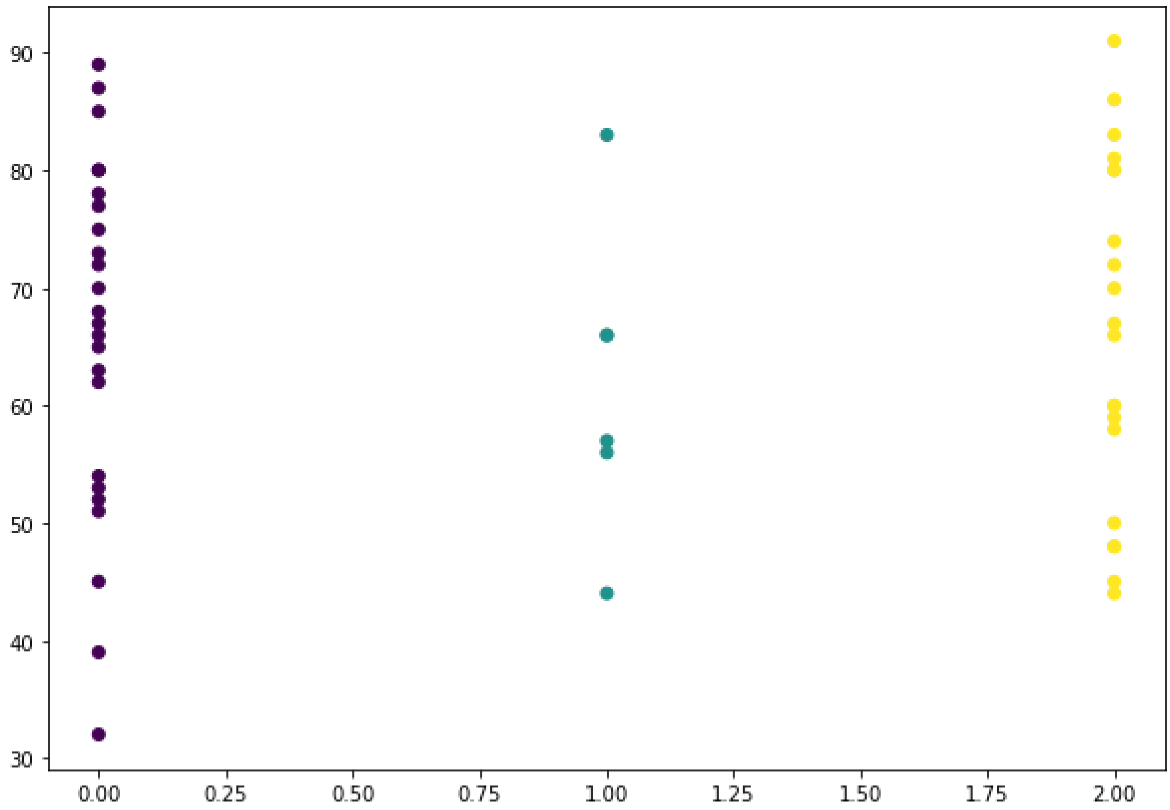
	clustersid	Murder	Assault	UrbanPop	Rape
		mean	mean	mean	mean
0	0	5.770833	129.083333	65.958333	18.575000
1	1	2.616667	54.833333	62.000000	12.333333
2	2	11.760000	255.550000	66.100000	27.090000

In [55]:

```
# Plot Clusters
plt.figure(figsize=(10,7))
plt.scatter(crime2['clustersid'],crime2['UrbanPop'],c=hclusters.labels_)
```

Out[55]:

<matplotlib.collections.PathCollection at 0xe103830d90>



In [ ]:

```

```

Assignment-07-K-Means Clustering (Crimes)

In [56]:

```
crime4=pd.read_csv("C:/Users/LENOVO/Documents/Custom Office Templates/crime_data.csv")
crime4
```

Out[56]:

	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

	Unnamed: 0	Murder	Assault	UrbanPop	Rape
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 [57]:

```
crime5=crime4.drop(['Unnamed: 0'],axis=1)
crime5
```

Out[57]:

	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
5	7.9	204	78	38.7
6	3.3	110	77	11.1
7	5.9	238	72	15.8
8	15.4	335	80	31.9
9	17.4	211	60	25.8
10	5.3	46	83	20.2
11	2.6	120	54	14.2
12	10.4	249	83	24.0
13	7.2	113	65	21.0
14	2.2	56	57	11.3
15	6.0	115	66	18.0
16	9.7	109	52	16.3
17	15.4	249	66	22.2
18	2.1	83	51	7.8
19	11.3	300	67	27.8
20	4.4	149	85	16.3
21	12.1	255	74	35.1
22	2.7	72	66	14.9
23	16.1	259	44	17.1
24	9.0	178	70	28.2
25	6.0	109	53	16.4
26	4.3	102	62	16.5
27	12.2	252	81	46.0
28	2.1	57	56	9.5
29	7.4	159	89	18.8
30	11.4	285	70	32.1
31	11.1	254	86	26.1
32	13.0	337	45	16.1
33	0.8	45	44	7.3

	Murder	Assault	UrbanPop	Rape
34	7.3	120	75	21.4
35	6.6	151	68	20.0
36	4.9	159	67	29.3
37	6.3	106	72	14.9
38	3.4	174	87	8.3
39	14.4	279	48	22.5
40	3.8	86	45	12.8
41	13.2	188	59	26.9
42	12.7	201	80	25.5
43	3.2	120	80	22.9
44	2.2	48	32	11.2
45	8.5	156	63	20.7
46	4.0	145	73	26.2
47	5.7	81	39	9.3
48	2.6	53	66	10.8
49	6.8	161	60	15.6

In [58]:

```
crime5.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Murder    50 non-null     float64
 1   Assault   50 non-null     int64  
 2   UrbanPop  50 non-null     int64  
 3   Rape      50 non-null     float64
dtypes: float64(2), int64(2)
memory usage: 1.7 KB
```

In [59]:

```
# Normalize hetrogenous numerical data by using Standard Scaler
crime5_norm=StandardScaler().fit_transform(crime5)
```

In [60]:

```
# Use Elbow Graph to find optimum number of clusters (K value) from K values range
# The K-means algorithm aims to choose centroids that minimise the inertia, or within-clust
# random state can be anything from 0 to 42, but the same number to be used everytime, so t
```

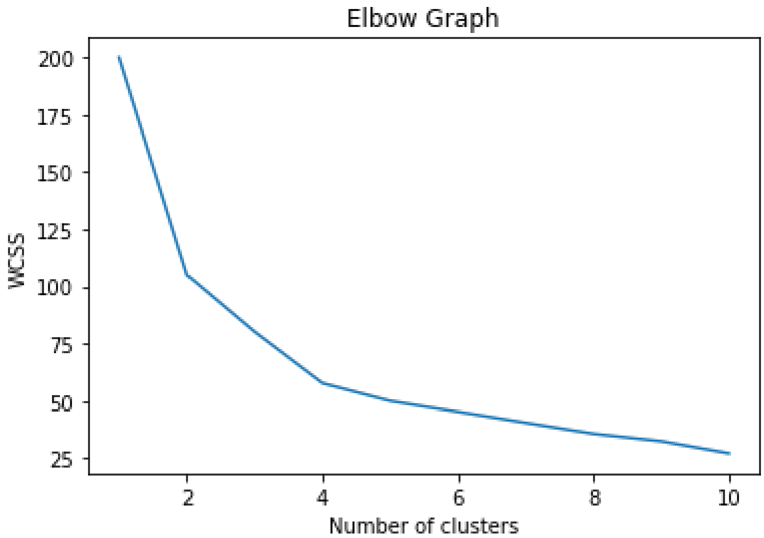
In [61]:

```
# within-cluster sum-of-squares criterion
wcss=[]
for i in range (1,11):
    kmeans=KMeans(n_clusters=i, random_state=2)
    kmeans.fit(crime5_norm)
    wcss.append(kmeans.inertia_)
```

```
C:\Users\LENOVO\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 th
e environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

In [62]:

```
# Plot K values range vs WCSS to get Elbow graph for choosing K (no. of clusters)
plt.plot(range(1,11),wcss)
plt.title('Elbow Graph')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



Building Cluster algorithm using K=4

In [63]:

```
# cluster algorithm using K=4
clusters1=KMeans(4,random_state=30).fit(crime5_norm)
clusters1
```

Out[63]:

```
KMeans(n_clusters=4, random_state=30)
```

In [64]:

```
clusters1.labels_
```

Out[64]:

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

In [65]:

```
# Assign clusters to the data set
crime6=crime5.copy()
crime6['clustersid']=clusters1.labels_
crime6
```

Out[65]:

	Murder	Assault	UrbanPop	Rape	clustersid
0	13.2	236	58	21.2	0
1	10.0	263	48	44.5	3
2	8.1	294	80	31.0	3
3	8.8	190	50	19.5	0
4	9.0	276	91	40.6	3
5	7.9	204	78	38.7	3
6	3.3	110	77	11.1	1
7	5.9	238	72	15.8	1
8	15.4	335	80	31.9	3
9	17.4	211	60	25.8	0
10	5.3	46	83	20.2	1
11	2.6	120	54	14.2	2
12	10.4	249	83	24.0	3
13	7.2	113	65	21.0	1
14	2.2	56	57	11.3	2
15	6.0	115	66	18.0	1
16	9.7	109	52	16.3	2
17	15.4	249	66	22.2	0
18	2.1	83	51	7.8	2
19	11.3	300	67	27.8	3
20	4.4	149	85	16.3	1
21	12.1	255	74	35.1	3
22	2.7	72	66	14.9	2
23	16.1	259	44	17.1	0
24	9.0	178	70	28.2	3
25	6.0	109	53	16.4	2
26	4.3	102	62	16.5	2
27	12.2	252	81	46.0	3
28	2.1	57	56	9.5	2
29	7.4	159	89	18.8	1
30	11.4	285	70	32.1	3
31	11.1	254	86	26.1	3

	Murder	Assault	UrbanPop	Rape	clustersid
32	13.0	337	45	16.1	0
33	0.8	45	44	7.3	2
34	7.3	120	75	21.4	1
35	6.6	151	68	20.0	1
36	4.9	159	67	29.3	1
37	6.3	106	72	14.9	1
38	3.4	174	87	8.3	1
39	14.4	279	48	22.5	0
40	3.8	86	45	12.8	2
41	13.2	188	59	26.9	0
42	12.7	201	80	25.5	3
43	3.2	120	80	22.9	1
44	2.2	48	32	11.2	2
45	8.5	156	63	20.7	1
46	4.0	145	73	26.2	1
47	5.7	81	39	9.3	2
48	2.6	53	66	10.8	2
49	6.8	161	60	15.6	1

In [66]:

```
# Compute the centroids for k=4 clusters with 4 variables
clusters1.cluster_centers_
```

Out[66]:

```
array([[ 1.42622412,  0.88321132, -0.82279055,  0.01946669],
       [-0.49440658, -0.3864845 ,  0.58167593, -0.26431024],
       [-0.97130281, -1.11783581, -0.93954982, -0.97657842],
       [ 0.70212683,  1.04999438,  0.72997363,  1.28990383]])
```

In [67]:

```
# Group data by Clusters (K=4)
crime6.groupby('clustersid').agg(['mean']).reset_index()
```

Out[67]:

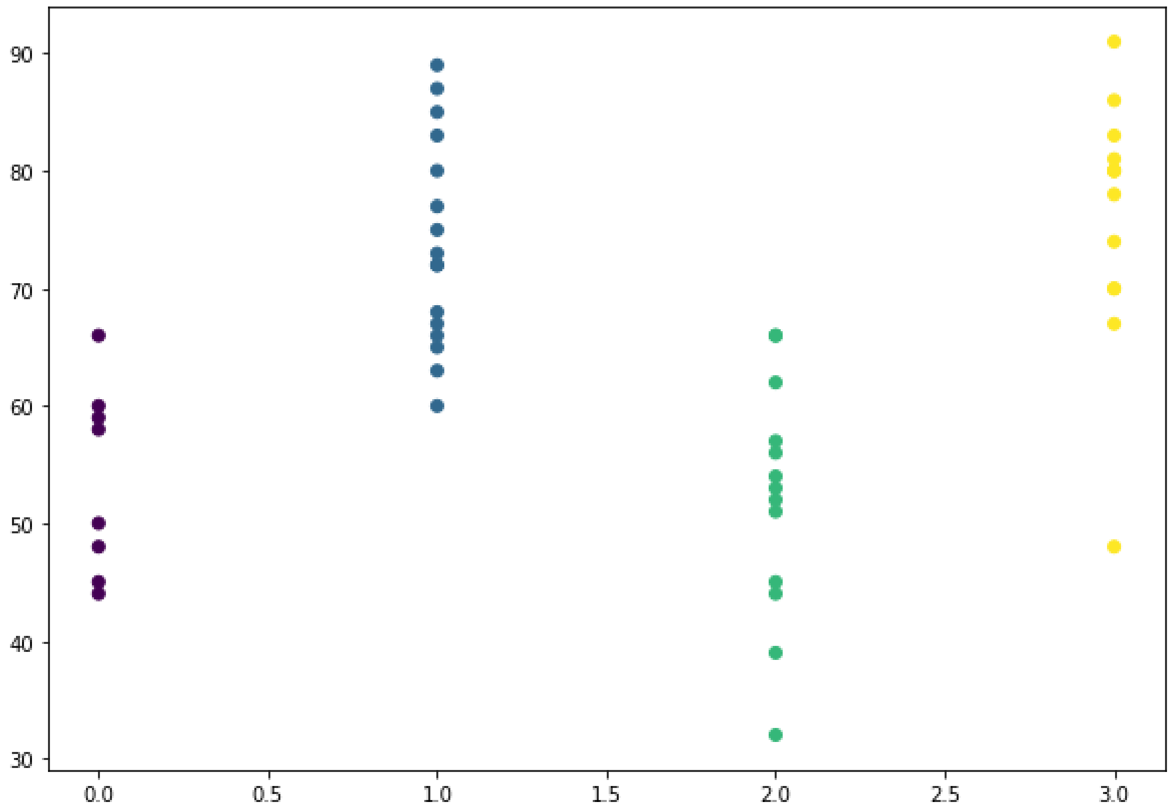
	clustersid	Murder	Assault	UrbanPop	Rape
		mean	mean	mean	mean
0	0	13.937500	243.625000	53.750000	21.412500
1	1	5.656250	138.875000	73.875000	18.781250
2	2	3.600000	78.538462	52.076923	12.176923
3	3	10.815385	257.384615	76.000000	33.192308

In [68]:

```
# Plot Clusters
plt.figure(figsize=(10,7))
plt.scatter(crime6['clustersid'],crime6['UrbanPop'],c=clusters1.labels_)
```

Out[68]:

```
<matplotlib.collections.PathCollection at 0xe10e2f8070>
```



In [ ]:

Assignment-07-DBSCAN Clustering (Crimes)

In [69]:

```
from sklearn.cluster import DBSCAN
```

In [70]:

```
# Import Dataset
crime=pd.read_csv("C:/Users/LENOVO/Documents/Custom Office Templates/crime_data.csv")
crime
```

Out[70]:

	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 [71]:

```
crime.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 [72]:

```
crime.drop(['Unnamed: 0'], axis=1, inplace=True)
crime
```

Out[72]:

	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
5	7.9	204	78	38.7
6	3.3	110	77	11.1
7	5.9	238	72	15.8
8	15.4	335	80	31.9
9	17.4	211	60	25.8
10	5.3	46	83	20.2
11	2.6	120	54	14.2
12	10.4	249	83	24.0
13	7.2	113	65	21.0
14	2.2	56	57	11.3
15	6.0	115	66	18.0
16	9.7	109	52	16.3
17	15.4	249	66	22.2
18	2.1	83	51	7.8
19	11.3	300	67	27.8
20	4.4	149	85	16.3
21	12.1	255	74	35.1
22	2.7	72	66	14.9
23	16.1	259	44	17.1
24	9.0	178	70	28.2
25	6.0	109	53	16.4
26	4.3	102	62	16.5
27	12.2	252	81	46.0
28	2.1	57	56	9.5
29	7.4	159	89	18.8
30	11.4	285	70	32.1
31	11.1	254	86	26.1
32	13.0	337	45	16.1
33	0.8	45	44	7.3

	Murder	Assault	UrbanPop	Rape
34	7.3	120	75	21.4
35	6.6	151	68	20.0
36	4.9	159	67	29.3
37	6.3	106	72	14.9
38	3.4	174	87	8.3
39	14.4	279	48	22.5
40	3.8	86	45	12.8
41	13.2	188	59	26.9
42	12.7	201	80	25.5
43	3.2	120	80	22.9
44	2.2	48	32	11.2
45	8.5	156	63	20.7
46	4.0	145	73	26.2
47	5.7	81	39	9.3
48	2.6	53	66	10.8
49	6.8	161	60	15.6

In [73]:

```
# Normalize hetrogenous numerical data using Standard Scaler fit transform to dataset
crime_norm=StandardScaler().fit_transform(crime)
crime_norm
```

Out[73]:

```
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 [74]:

```
# DBSCAN Clustering
dbscan=DBSCAN(eps=1,min_samples=4)
dbscan.fit(crime_norm)
```

Out[74]:

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

In [75]:

```
# Noisy samples are given the label -1.
dbscan.labels_
```

Out[75]:

```
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 [76]:

```
# Adding Clusters to dataset
crime['clusters']=dbscan.labels_
crime
```

Out[76]:

	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
10	5.3	46	83	20.2	-1
11	2.6	120	54	14.2	1
12	10.4	249	83	24.0	-1
13	7.2	113	65	21.0	1
14	2.2	56	57	11.3	1
15	6.0	115	66	18.0	1
16	9.7	109	52	16.3	1
17	15.4	249	66	22.2	0
18	2.1	83	51	7.8	1
19	11.3	300	67	27.8	-1
20	4.4	149	85	16.3	1
21	12.1	255	74	35.1	-1
22	2.7	72	66	14.9	1
23	16.1	259	44	17.1	-1
24	9.0	178	70	28.2	1
25	6.0	109	53	16.4	1
26	4.3	102	62	16.5	1
27	12.2	252	81	46.0	-1
28	2.1	57	56	9.5	1
29	7.4	159	89	18.8	1
30	11.4	285	70	32.1	-1
31	11.1	254	86	26.1	-1
32	13.0	337	45	16.1	-1

	Murder	Assault	UrbanPop	Rape	clusters
33	0.8	45	44	7.3	1
34	7.3	120	75	21.4	1
35	6.6	151	68	20.0	1
36	4.9	159	67	29.3	1
37	6.3	106	72	14.9	1
38	3.4	174	87	8.3	1
39	14.4	279	48	22.5	0
40	3.8	86	45	12.8	1
41	13.2	188	59	26.9	0
42	12.7	201	80	25.5	-1
43	3.2	120	80	22.9	1
44	2.2	48	32	11.2	1
45	8.5	156	63	20.7	1
46	4.0	145	73	26.2	1
47	5.7	81	39	9.3	1
48	2.6	53	66	10.8	1
49	6.8	161	60	15.6	1

In [77]:

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

Out[77]:

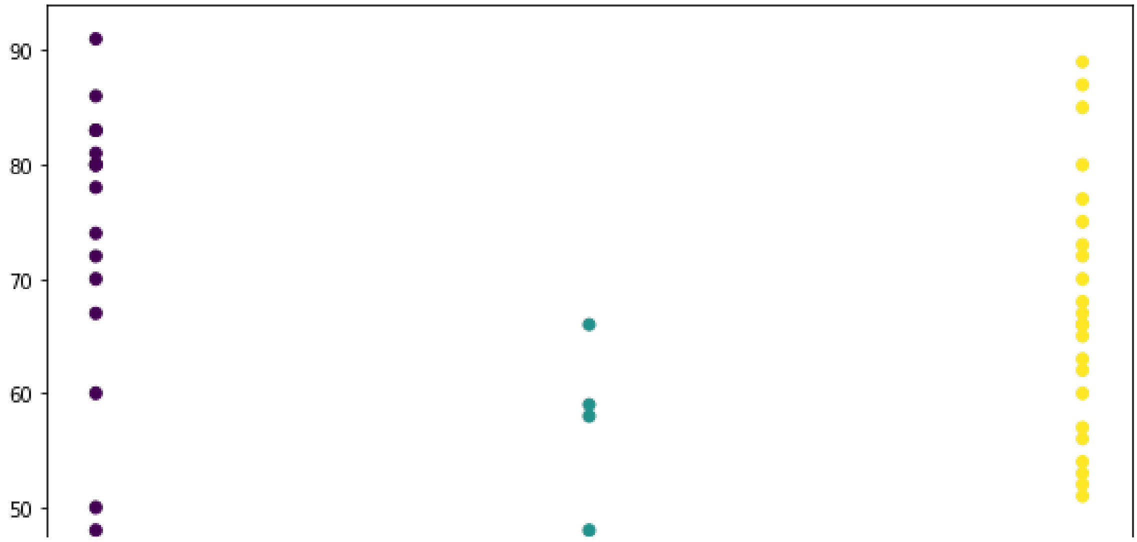
	clusters	Murder	Assault	UrbanPop	Rape
		mean	mean	mean	mean
0	-1	11.005556	247.166667	70.666667	28.766667
1	0	14.050000	238.000000	57.750000	23.200000
2	1	4.825000	112.035714	63.357143	16.107143

In [78]:

```
# plot clusters
plt.figure(figsize=(10,7))
plt.scatter(crime['clusters'],crime['UrbanPop'],c=dbscan.labels_)
```

Out[78]:

```
<matplotlib.collections.PathCollection at 0xe10e34b820>
```



In [ ]: