CRIME DATASET

In [ ]:
```
#Import the libraries
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn import preprocessing
import seaborn as sns
```

Problem : Perform Clustering for the crime data and identify the number of clusters formed and draw inferences

In [2]:
```
# import the crime data dataset
crime=pd.read_csv("C:\\Users\\nishi\\Desktop\\Assignments\\Clustering\\crime_data
```

In [3]:
```
crime
```

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
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 [4]:
```
crime1=crime.copy()
```

In [5]:
```
crime1.columns = ['City','Murder' , 'Assault', 'Urbanpop','Rape']
```

In [6]:
```
crime1
```

Out[6]:

	City	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

	City	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 [7]:
```
crime1.loc[:,'Total'] = crime1.sum(numeric_only=True, axis=1)
```

In [8]:
```
crime1.head()
```

Out[8]:

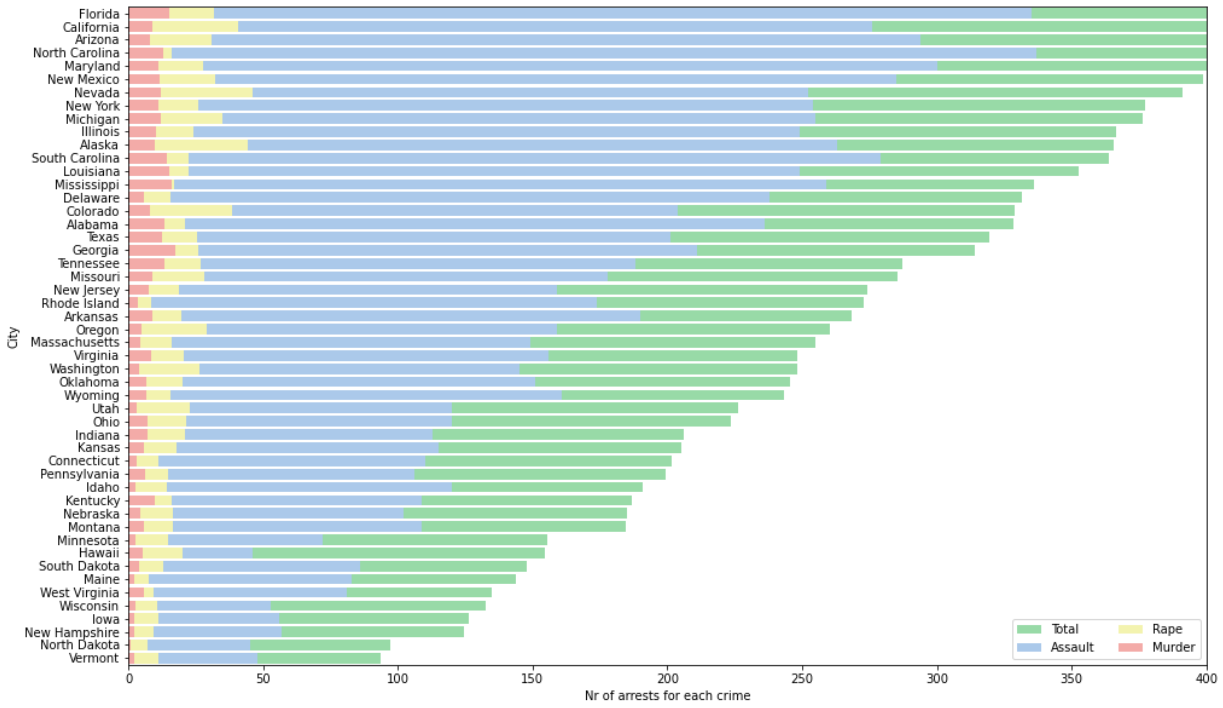
	City	Murder	Assault	Urbanpop	Rape	Total
0	Alabama	13.2	236	58	21.2	328.4
1	Alaska	10.0	263	48	44.5	365.5
2	Arizona	8.1	294	80	31.0	413.1
3	Arkansas	8.8	190	50	19.5	268.3
4	California	9.0	276	91	40.6	416.6

In [9]:
```
crime1.describe()
```

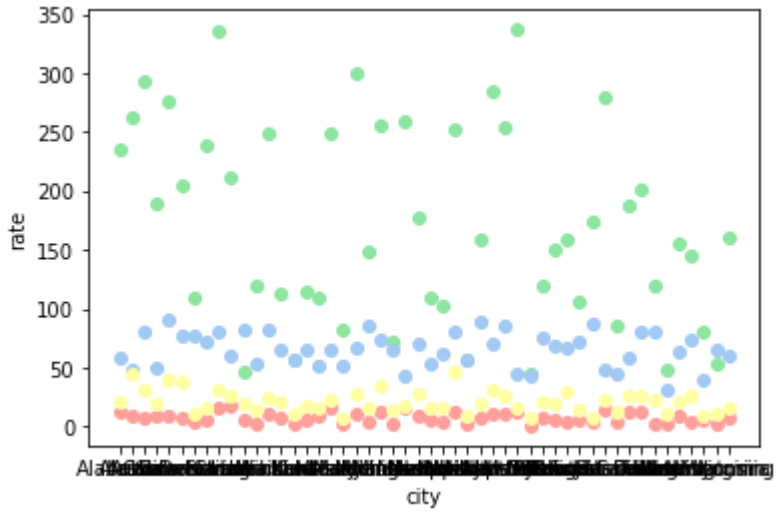
Out[9]:

	Murder	Assault	Urbanpop	Rape	Total
count	50.00000	50.000000	50.000000	50.000000	50.000000
mean	7.78800	170.760000	65.540000	21.232000	265.320000
std	4.35551	83.337661	14.474763	9.366385	98.350844
min	0.80000	45.000000	32.000000	7.300000	93.400000
25%	4.07500	109.000000	54.500000	15.075000	187.950000
50%	7.25000	159.000000	66.000000	20.100000	257.450000
75%	11.25000	249.000000	77.750000	26.175000	348.500000
max	17.40000	337.000000	91.000000	46.000000	462.300000

In [10]:
```
f, ax = plt.subplots(figsize=(16, 10))

stats = crime1.sort_values("Total", ascending=False)

sns.set_color_codes("pastel")

sns.barplot(x="Total", y="City", data=stats,
            label="Total", color="g")

sns.barplot(x="Assault", y="City", data=stats,
            label="Assault", color="b")

sns.barplot(x="Rape", y="City", data=stats,
            label="Rape", color="y")

sns.barplot(x="Murder", y="City", data=stats,
            label="Murder", color="r")

ax.legend(ncol=2, loc="lower right", frameon=True)
ax.set(xlim=(0, 400), ylabel="City",
       xlabel="Nr of arrests for each crime");
```



In [11]:
```
plt.scatter(crime1.City, crime1.Murder, color='r')
plt.scatter(crime1.City, crime1.Assault, color='g')
plt.scatter(crime1.City, crime1.Urbanpop, color='b')
plt.scatter(crime1.City, crime1.Rape, color='y')
plt.xlabel('city')
plt.ylabel('rate')
plt.show()
```



Finding out the optimal number of clusters

In [12]:
```
X = crime1[['Murder', 'Assault', 'Rape', 'Urbanpop']]
```

In [13]:
```
crime1_norm = preprocessing.scale(X)
```

In [14]:
```
crime1_norm = pd.DataFrame(crime1_norm) #standardize the data to normal distribut
```

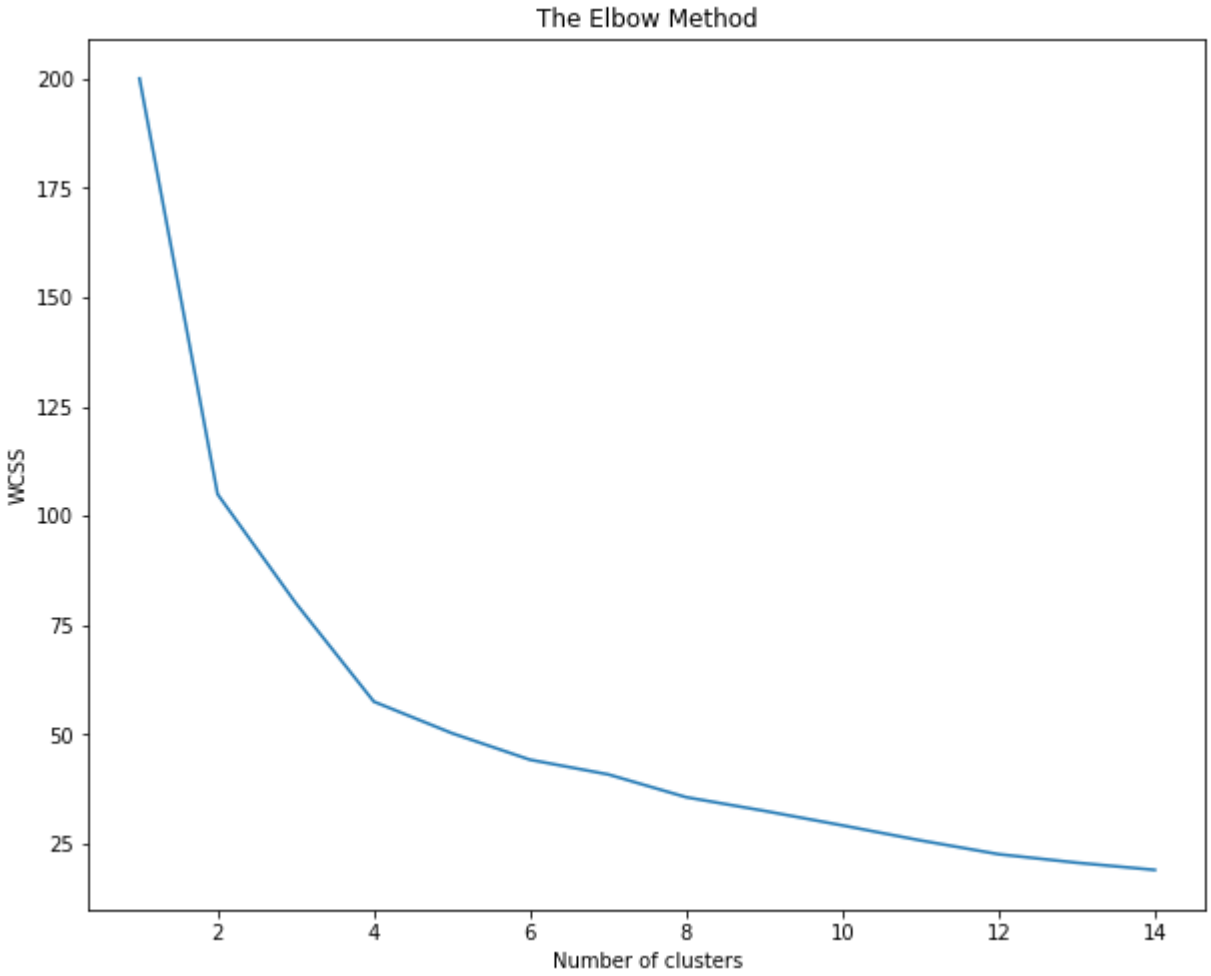
In [15]:
```
crime1_norm.head()
```

Out[15]:

	0	1	2	3
0	1.255179	0.790787	-0.003451	-0.526195
1	0.513019	1.118060	2.509424	-1.224067
2	0.072361	1.493817	1.053466	1.009122
3	0.234708	0.233212	-0.186794	-1.084492
4	0.281093	1.275635	2.088814	1.776781

In [16]:
```
plt.figure(figsize=(10, 8))
wcss = []
for i in range(1, 15):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans.fit(crime1_norm)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 15), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```

```
C:\Users\nishi\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1036: 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(
```



The scree plot levels off at k=4 and let's use it to determine the clusters

Analysing the data

In [17]:
```
kmeans = KMeans(n_clusters = 4, init = 'k-means++', random_state = 42)
y_kmeans = kmeans.fit_predict(crime1_norm)
```

In [18]:
```
y_kmeans
```

Out[18]:
```
array([1, 2, 2, 1, 2, 2, 0, 0, 2, 1, 0, 3, 2, 0, 3, 0, 3, 1, 3, 2, 0, 2,
       3, 1, 2, 3, 3, 2, 3, 0, 2, 2, 1, 3, 0, 0, 0, 0, 0, 1, 3, 1, 2, 0,
       3, 0, 0, 3, 3, 0])
```

In [19]:
```
y_kmeans1=y_kmeans+1
cluster = list(y_kmeans1)
```

In [20]:
```
crime1['cluster'] = cluster
```

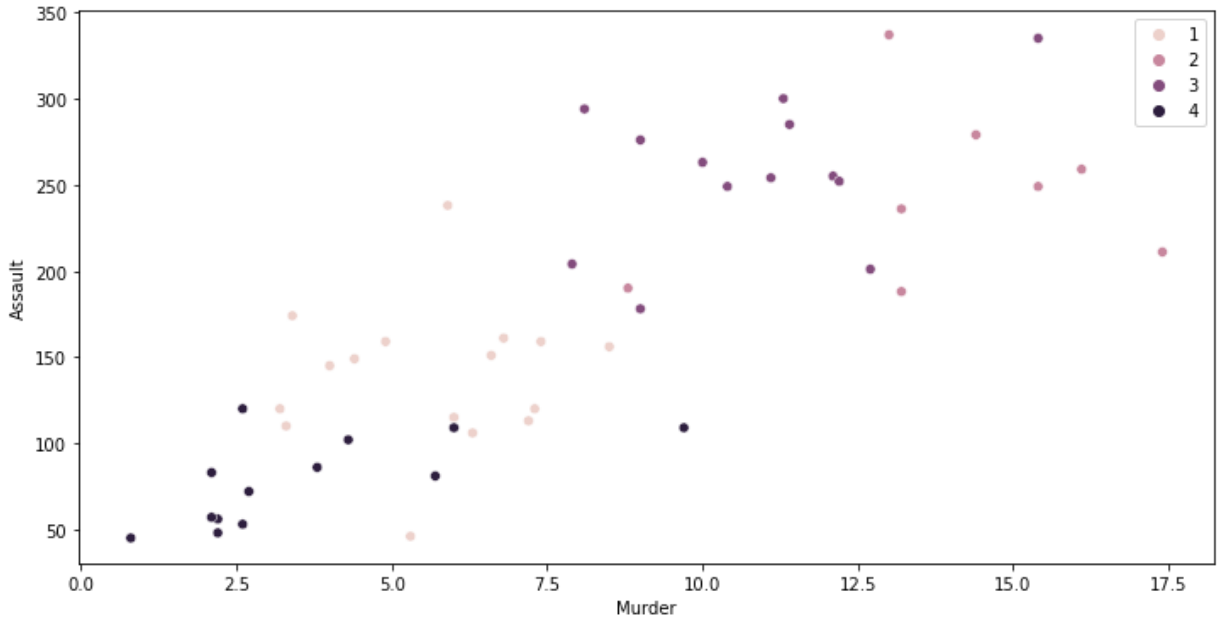
In [21]:
```
kmeans_mean_cluster = pd.DataFrame(round(crime1.groupby('cluster').mean(),1))
kmeans_mean_cluster
```

Out[21]:

cluster	Murder	Assault	Urbanpop	Rape	Total
1	5.7	138.9	73.9	18.8	237.2
2	13.9	243.6	53.8	21.4	332.7
3	10.8	257.4	76.0	33.2	377.4
4	3.6	78.5	52.1	12.2	146.4

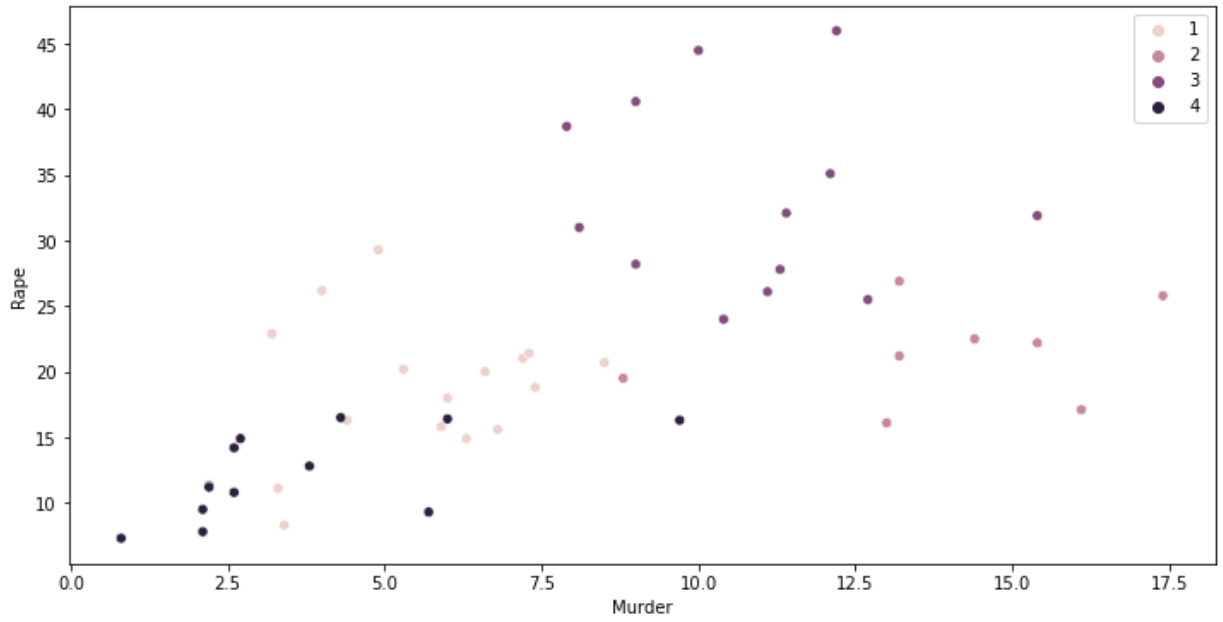
In [22]:
```
plt.figure(figsize=(12,6))
sns.scatterplot(x=crime1['Murder'], y = crime1['Assault'],hue=y_kmeans1)
```

Out[22]:
```
<AxesSubplot:xlabel='Murder', ylabel='Assault'>
```



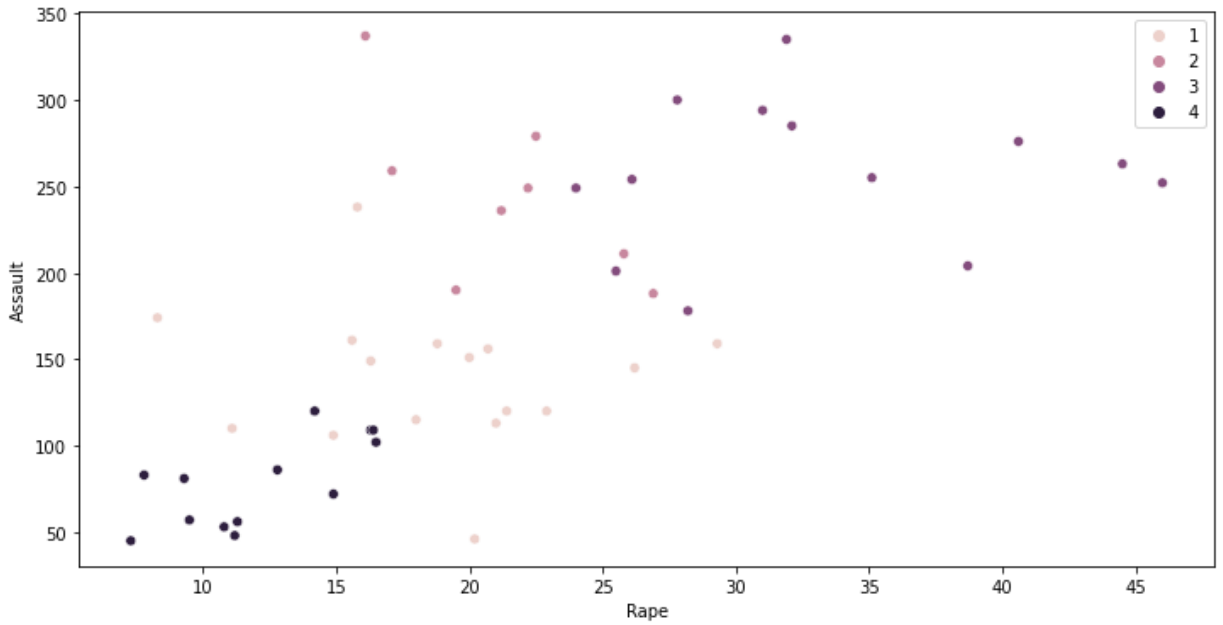
In [23]:
```
plt.figure(figsize=(12,6))
sns.scatterplot(x=crime1['Murder'], y = crime1['Rape'],hue=y_kmeans1)
```

Out[23]: <AxesSubplot:xlabel='Murder', ylabel='Rape'>



In [24]:
```
plt.figure(figsize=(12,6))
sns.scatterplot(x=crime1['Rape'], y = crime1['Assault'],hue=y_kmeans1)
```

Out[24]: <AxesSubplot:xlabel='Rape', ylabel='Assault'>



In [25]:
```
stats = crime1.sort_values("Total", ascending=True)
crime1_total= pd.DataFrame(stats)
```

In [26]:
```
crime1_total.head()
```

Out[26]:

	City	Murder	Assault	Urbanpop	Rape	Total	cluster
44	Vermont	2.2	48	32	11.2	93.4	4
33	North Dakota	0.8	45	44	7.3	97.1	4
28	New Hampshire	2.1	57	56	9.5	124.6	4
14	Iowa	2.2	56	57	11.3	126.5	4
48	Wisconsin	2.6	53	66	10.8	132.4	4

Conclusion drawn from the Crime Dataset

a) Analysing Murder and Assault variables shows a clearer connection between them. Higher the murder rates in a city higher the assaults and vice versa.

b) Comparing with murders and assaults, there is much more spread among the clusters when comparing murders and rapes. Some correlation is visible, but lower murder rates in a city seem to indicate lower number of rapes and vice versa.

c) As with murder and assault, also rates of rape and assault show clearer correlations.

EASTWEST AIRLINES DATASET

In [1]:
```
# Import the Libraries
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn import preprocessing
import seaborn as sns
import scipy.cluster.hierarchy as sch
from sklearn.cluster import AgglomerativeClustering
```

PROBLEM: Perform clustering (Both hierarchical and K means clustering) for the airlines data to obtain optimum number of clusters

In [2]:
```
# Import the datset
airlines1=pd.read_csv("C:\\Users\\nishi\\Desktop\\Assignments\\Clustering\\EastWe
```

In [3]:
```
airlines1
```

Out[3]:

	ID#	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus_trans	Flig
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 [4]:
```
airlines2=airlines1.copy()
```

In [5]:
```
airlines2.head()
```

Out[5]:

	ID#	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus_trans	Flight_r
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	

In [6]:
```
airlines2_norm = preprocessing.scale(airlines2)
```

In [7]:
```
airlines2_norm = pd.DataFrame(airlines2_norm) #standardize the data to normal dis
```

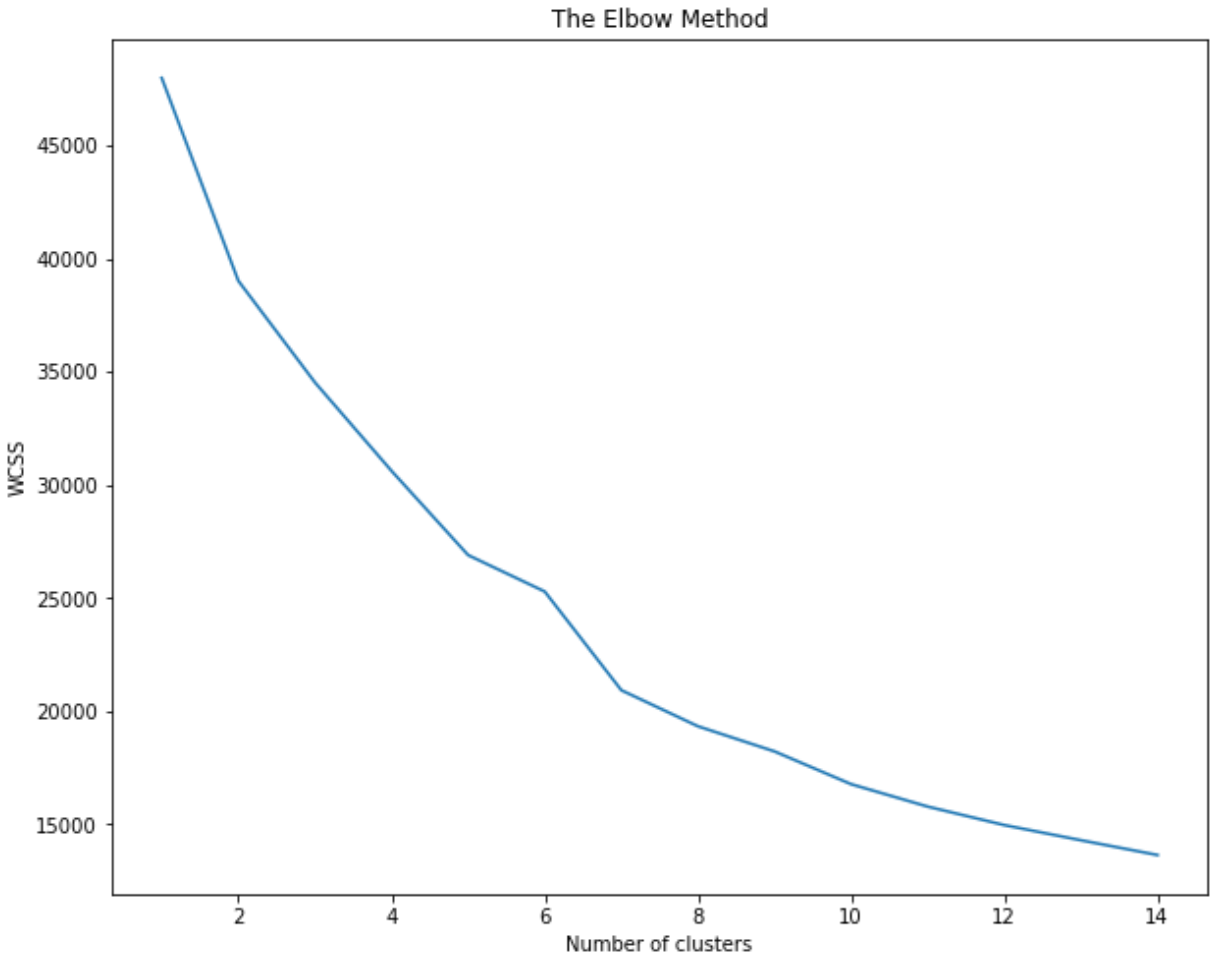
In [8]:
```
airlines2_norm.head()
```

Out[8]:

	0	1	2	3	4	5	6	7	8
0	-1.735125	-0.451141	-0.186299	-0.769578	-0.098242	-0.062767	-0.702786	-1.104065	-0.328603
1	-1.734263	-0.539457	-0.186299	-0.769578	-0.098242	-0.062767	-0.701088	-0.999926	-0.328603
2	-1.733402	-0.320031	-0.186299	-0.769578	-0.098242	-0.062767	-0.539253	-0.791649	-0.328603
3	-1.732540	-0.583799	-0.186299	-0.769578	-0.098242	-0.062767	-0.689286	-1.104065	-0.328603
4	-1.731679	0.239678	-0.186299	1.409471	-0.098242	-0.062767	1.083121	1.499394	1.154932

FINDING OUT THE OPTIMAL NUMBER OF CLUSTERS

In [9]:
```
plt.figure(figsize=(10, 8))
wcss = []
for i in range(1, 15):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans.fit(airlines2_norm)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 15), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```

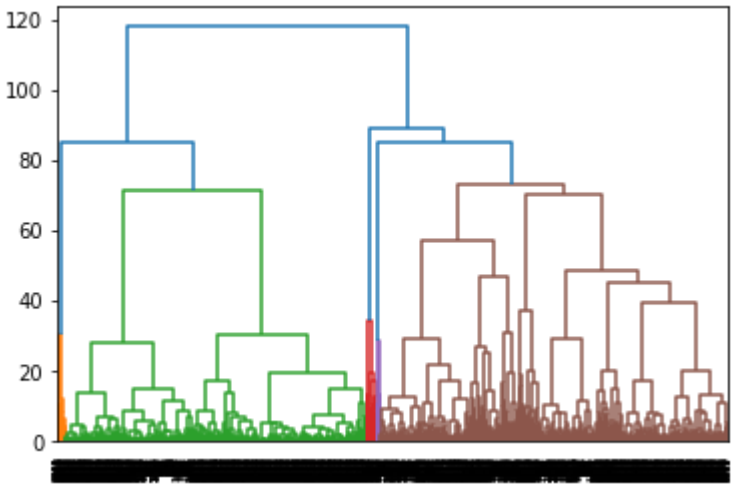


As seen from the elbow graph, the slope changes at 2. However, since splting the dataset into 2 groups would not be very beneficial, we further evaluate clusters for higher values of k.

H Clustering

Euclidean distance, Ward

In [10]:
```
dendrogram = sch.dendrogram(sch.linkage(airlines2_norm, method='ward'))
```



From the Ward method, we see that as the height increases the clusters get grouped together

We cut the tree at height 85 to obtain 3 clusters and then assigned each cluster with its respective observations.

In [11]:
```
X = airlines2_norm.values
```

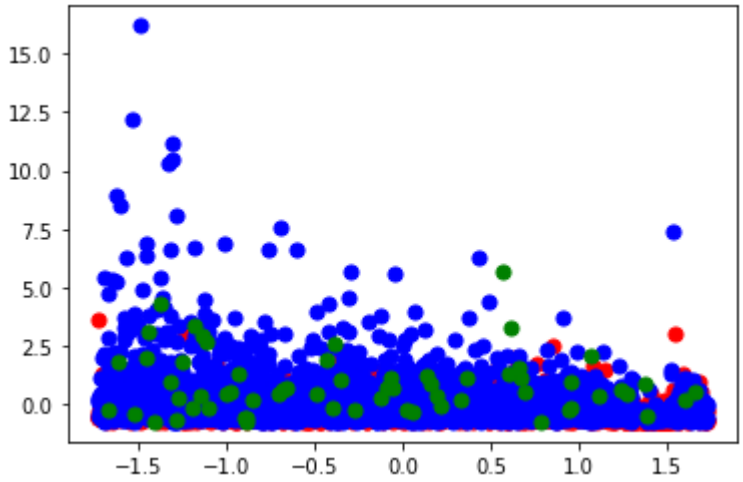
In [12]:
```
model = AgglomerativeClustering(n_clusters=3, affinity='euclidean', linkage='ward
```

In [13]:
```
h_cluster = model.fit(X)
```

In [14]:
```
labels = model.labels_
```

In [15]:
```
plt.scatter(X[labels==0, 0], X[labels==0, 1], s=50, marker='o', color='red')
plt.scatter(X[labels==1, 0], X[labels==1, 1], s=50, marker='o', color='blue')
plt.scatter(X[labels==2, 0], X[labels==2, 1], s=50, marker='o', color='green')
```

Out[15]:
```
<matplotlib.collections.PathCollection at 0x1e65e0b0940>
```



K Means

In [17]:
```
kmeans = KMeans(n_clusters = 3, init = 'k-means++', random_state = 42)
k_means = kmeans.fit_predict(airlines2_norm)
```

In [18]:
```
k_means
```

Out[18]:
```
array([0, 0, 0, ..., 0, 0, 0])
```

In [20]:
```
k_means1=k_means+1
k_cluster = list(k_means1)
```

In [22]:
```
airlines2['k_cluster'] = k_cluster
```

In [23]:
```
kmeans_mean_cluster = pd.DataFrame(round(airlines2.groupby('k_cluster').mean(),1)
```

In [24]:
```
kmeans_mean_cluster
```

Out[24]:

k_cluster	ID#	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus_trar
1	2327.1	42243.6	91.1	1.2	1.0	1.0	4896.4	7
2	1445.6	119557.7	165.6	3.6	1.0	1.0	38921.2	18
3	1753.1	189304.2	788.7	2.2	1.0	1.0	31780.5	27

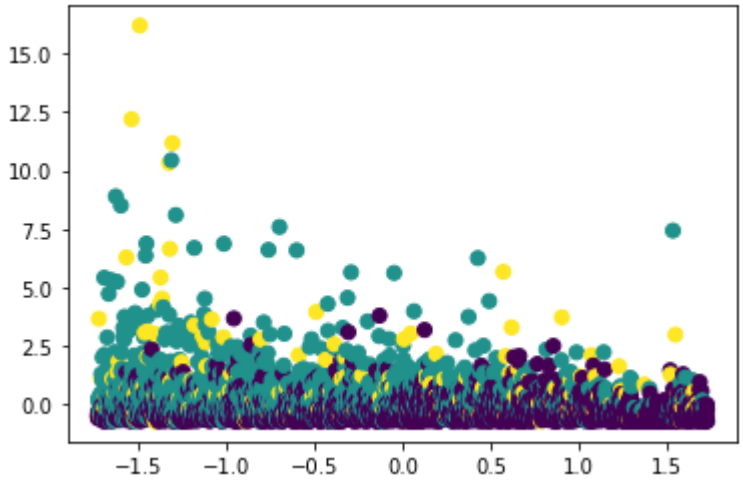
In [25]:
```
pd.DataFrame(round(airlines2.groupby('k_cluster').count(),1))
```

Out[25]:

k_cluster	ID#	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus_trans
1	2525	2525	2525	2525	2525	2525	2525	2525
2	1310	1310	1310	1310	1310	1310	1310	1310
3	164	164	164	164	164	164	164	164

In [26]:
```
plt.scatter(X[:, 0], X[:, 1], c=k_means, s=50, cmap='viridis')
```

Out[26]:
```
<matplotlib.collections.PathCollection at 0x1e65e3aae80>
```



CONCLUSION

From the above data generated from K-Means clustering, we can see Cluster-1 has around 63% total travelers and cluster 2 has 33% of the travelers. We will target cluster 1 & 2. Cluster 1 contains less frequent or first time travellers, by giving them discount provided they travel more than twice or thrice and introduce more offer if they register or take the membership.