In [1]:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
```

In [3]:

```
crime_rates=pd.read_csv("C:\\Users\\Admin\\Downloads\\assignment 4\\crime_data.csv", in
```

In [4]:

```
crime_rates.head()
```

Out[4]:

	Murder	Assault	UrbanPop	Rape
Alabama	13.2	236	58	21.2
Alaska	10.0	263	48	44.5
Arizona	8.1	294	80	31.0
Arkansas	8.8	190	50	19.5
California	9.0	276	91	40.6

In [5]:

```
crime_rates.describe()
```

Out[5]:

	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 [6]:

```
#standardize the data to normal distribution
from sklearn import preprocessing
crime_rates_standardized = preprocessing.scale(crime_rates)
print(crime_rates_standardized)
crime_rates_standardized = pd.DataFrame(crime_rates_standardized)
```

```
[[ 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]]
```

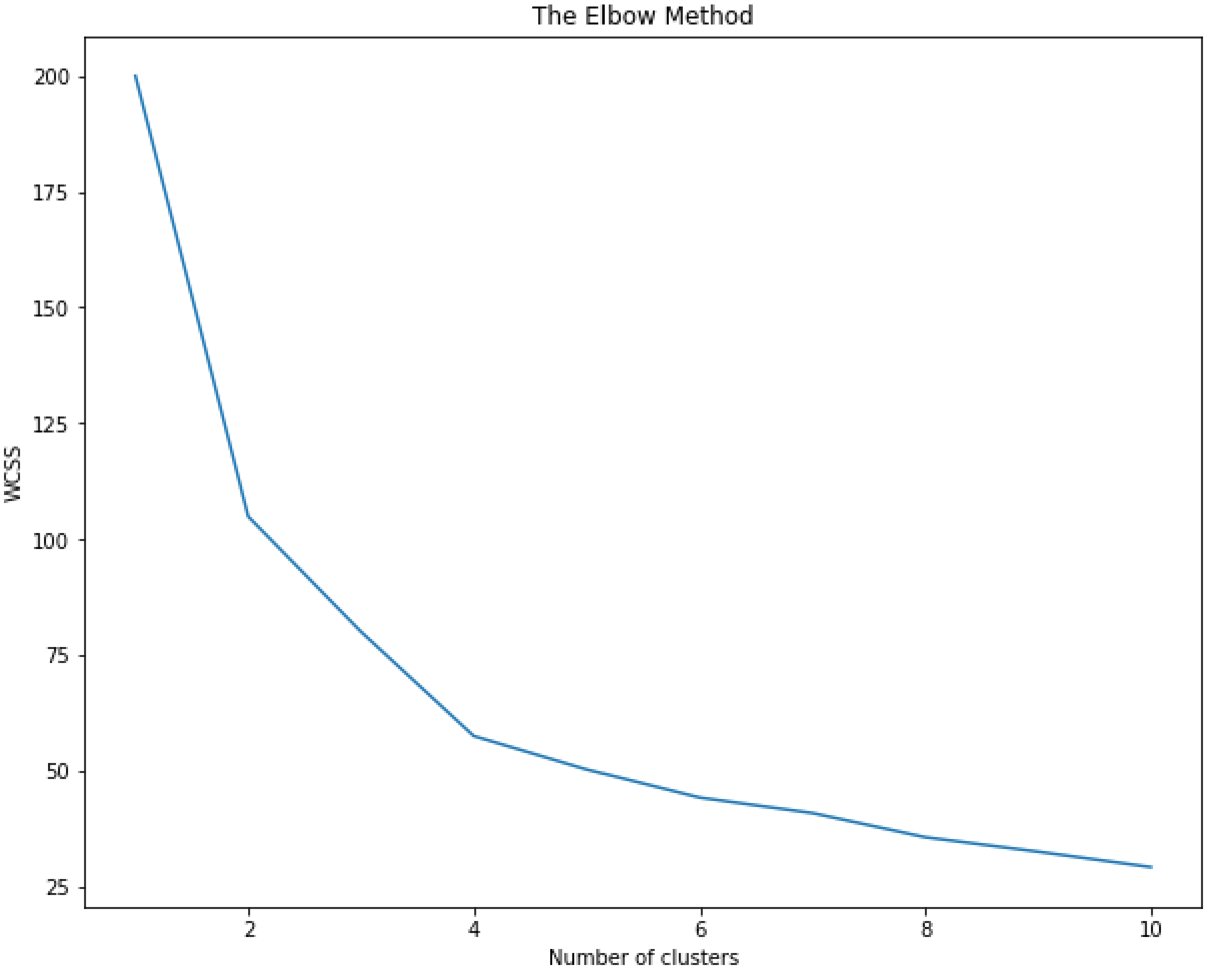
In [8]:

```python
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
%matplotlib inline

plt.figure(figsize=(10, 8))
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans.fit(crime_rates_standardized)
    wcss.append(kmeans.inertia_) #criterion based on which K-means clustering works
plt.plot(range(1, 11), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```

```
C:\Users\Admin\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 th
an available threads. You can avoid it by setting the environment variable OMP_NUM_THREA
DS=1.
  warnings.warn(
```



In [9]:

```
# Fitting K-Means to the dataset
kmeans = KMeans(n_clusters = 4, init = 'k-means++', random_state = 42)
y_kmeans = kmeans.fit_predict(crime_rates_standardized)

y_kmeans
```

Out[9]:

```
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 [10]:

```
#beginning of  the cluster numbering with 1 instead of 0
y_kmeans1=y_kmeans+1

# New list called cluster
cluster = list(y_kmeans1)
# Adding cluster to our data set
crime_rates['cluster'] = cluster
```

In [11]:

```
#Mean of clusters 1 to 4
kmeans_mean_cluster = pd.DataFrame(round(crime_rates.groupby('cluster').mean(),1))
kmeans_mean_cluster
```

Out[11]:

	Murder	Assault	UrbanPop	Rape
cluster				
1	5.7	138.9	73.9	18.8
2	13.9	243.6	53.8	21.4
3	10.8	257.4	76.0	33.2
4	3.6	78.5	52.1	12.2

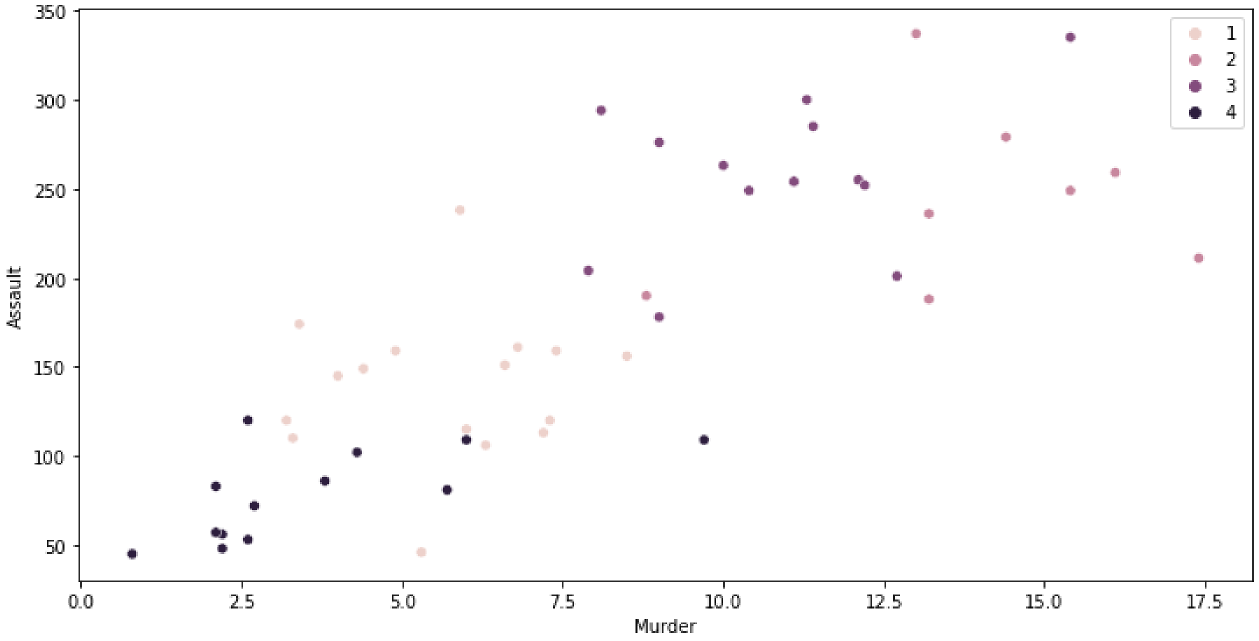
In [12]:

```
import seaborn as sns

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

Out[12]:

```
<AxesSubplot:xlabel='Murder', ylabel='Assault'>
```



In [13]:

```
crime_rates[crime_rates['cluster']==1]
```

Out[13]:

	Murder	Assault	UrbanPop	Rape	cluster
Connecticut	3.3	110	77	11.1	1
Delaware	5.9	238	72	15.8	1
Hawaii	5.3	46	83	20.2	1
Indiana	7.2	113	65	21.0	1
Kansas	6.0	115	66	18.0	1
Massachusetts	4.4	149	85	16.3	1
New Jersey	7.4	159	89	18.8	1
Ohio	7.3	120	75	21.4	1

	Murder	Assault	UrbanPop	Rape	cluster
Oklahoma	6.6	151	68	20.0	1
Oregon	4.9	159	67	29.3	1
Pennsylvania	6.3	106	72	14.9	1
Rhode Island	3.4	174	87	8.3	1
Utah	3.2	120	80	22.9	1
Virginia	8.5	156	63	20.7	1
Washington	4.0	145	73	26.2	1
Wyoming	6.8	161	60	15.6	1

In [15]:

```
from IPython.display import HTML
import base64

df = crime_rates
def create_download_link( df, title = "Download CSV file", filename = "data.csv"):
    csv = df.to_csv()
    b64 = base64.b64encode(csv.encode())
    payload = b64.decode()
    html = '<a download="{filename}" href="data:text/csv;base64,{payload}" target="_bla
    html = html.format(payload=payload,title=title,filename=filename)
    return HTML(html)

create_download_link(df)
```

Out[15]: Download CSV file

In [16]:

```
%%HTML

<div class='tableauPlaceholder' id='viz1558006161579' style='position: relative'>
<noscript>
<a href='#'>
<img alt=' ' src='https:&#47;&#47;public.tableau.com&#47;static&#47;images&#47;US&#47;U
</noscript>

<object class='tableauViz'  style='display:none;'>
<param name='host_url' value='https%3A%2F%2Fpublic.tableau.com%2F' />
<param name='embed_code_version' value='3' />
<param name='site_root' value='' />
<param name='name' value='USCrimeRatesClusters&#47;Dashboard' />
<param name='tabs' value='no' />
<param name='toolbar' value='yes' />
<param name='static_image' value='https:&#47;&#47;public.tableau.com&#47;static&#47;ima
<param name='display_static_image' value='yes' />
<param name='display_spinner' value='yes' />
<param name='display_overlay' value='yes' />
<param name='display_count' value='yes' />
</object>
</div>
<script type='text/javascript'>                    var divElement = document.getElement
```

In [ ]: