Importing Libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import pandas as pd
```

Importing Dataset

In [12]:
```
df=pd.read_csv("Amazon.com Clusturing.csv")
x=df.iloc[:,3:5].values
```

In [13]:
```
print(x)
```

```
[[ 306555      44]
 [ 306555      91]
 [ 326992       7]
 [ 326992      87]
 [ 347429      45]
 [ 347429      86]
 [ 367866       7]
 [ 367866     106]
 [ 388303       4]
 [ 388303      81]
 [ 388303      16]
 [ 388303     111]
 [ 408740      17]
 [ 408740      87]
 [ 408740      15]
 [ 408740      89]
 [ 429177      40]
 [ 429177      74]
 [ 470051      33]
 [ 470051     110]
 [ 490488      40]
 [ 490488      82]
 [ 510925       6]
 [ 510925      82]
 [ 572236      16]
 [ 572236      92]
 [ 572236      36]
 [ 572236      69]
 [ 592673      35]
 [ 592673      98]
 [ 613110       5]
 [ 613110      82]
 [ 674421       5]
 [ 674421     104]
 [ 674421      16]
 [ 674421      91]
 [ 694858      20]
 [ 694858      82]
 [ 756169      30]
 [ 756169      84]
 [ 776606      40]
 [ 776606     104]
 [ 797043      41]
 [ 797043      69]
 [ 797043      32]
 [ 797043      73]
 [ 817480      62]
 [ 817480      53]
 [ 817480      48]
 [ 817480      48]
 [ 858354      59]
 [ 858354      68]
 [ 878791      61]
 [ 878791      68]
 [ 878791      51]
 [ 878791      46]
 [ 899228      56]
 [ 899228      52]
 [ 940102      58]
```

```
[ 940102      52]
[ 940102      63]
[ 940102      62]
[ 960539      59]
[ 960539      67]
[ 980976      58]
[ 980976      67]
[ 980976      56]
[ 980976      54]
[ 980976      67]
[ 980976      53]
[1001413      62]
[1001413      48]
[1021850      55]
[1021850      63]
[1103598      53]
[1103598      61]
[1103598      60]
[1103598      54]
[1103598      59]
[1103598      48]
[1103598      58]
[1103598      62]
[1103598      46]
[1103598      50]
[1103598      64]
[1103598      52]
[1164909      65]
[1164909      62]
[1185346      68]
[1185346      52]
[1205783      62]
[1205783      46]
[1226220      55]
[1226220      45]
[1226220      48]
[1226220      59]
[1226220      53]
[1226220      56]
[1246657      48]
[1246657      55]
[1267094      46]
[1267094      54]
[1267094      67]
[1267094      62]
[1267094      63]
[1267094      48]
[1287531      56]
[1287531      52]
[1287531      49]
[1287531      54]
[1287531      59]
[1287531      61]
[1307968      48]
[1307968      52]
[1328405      54]
[1328405      56]
[1328405      49]
[1328405      67]
```

```
[1369279     49]
[1369279     64]
[1369279     63]
[1369279     45]
[1410153     65]
[1410153    102]
[1430590     33]
[1430590     87]
[1451027     40]
[1451027    107]
[1451027     13]
[1451027     84]
[1451027     11]
[1451027     84]
[1471464     39]
[1471464     80]
[1491901      6]
[1491901     99]
[1491901      8]
[1491901     82]
[1512338     12]
[1512338     81]
[1532775      6]
[1532775    105]
[1553212     45]
[1553212     98]
[1573649     14]
[1573649    109]
[1573649     41]
[1573649     83]
[1594086     25]
[1594086    101]
[1594086     20]
[1594086     99]
[1594086     23]
[1594086     86]
[1594086     18]
[1594086    100]
[1594086      2]
[1594086     88]
[1594086      2]
[1594086     82]
[1614523     40]
[1614523     93]
[1655397      6]
[1655397    105]
[1737145     30]
[1737145     84]
[1757582     23]
[1757582    107]
[1778019     31]
[1778019     71]
[1778019     15]
[1778019     84]
[1778019     12]
[1778019    104]
[1798456     15]
[1798456     97]
[1798456     17]
```

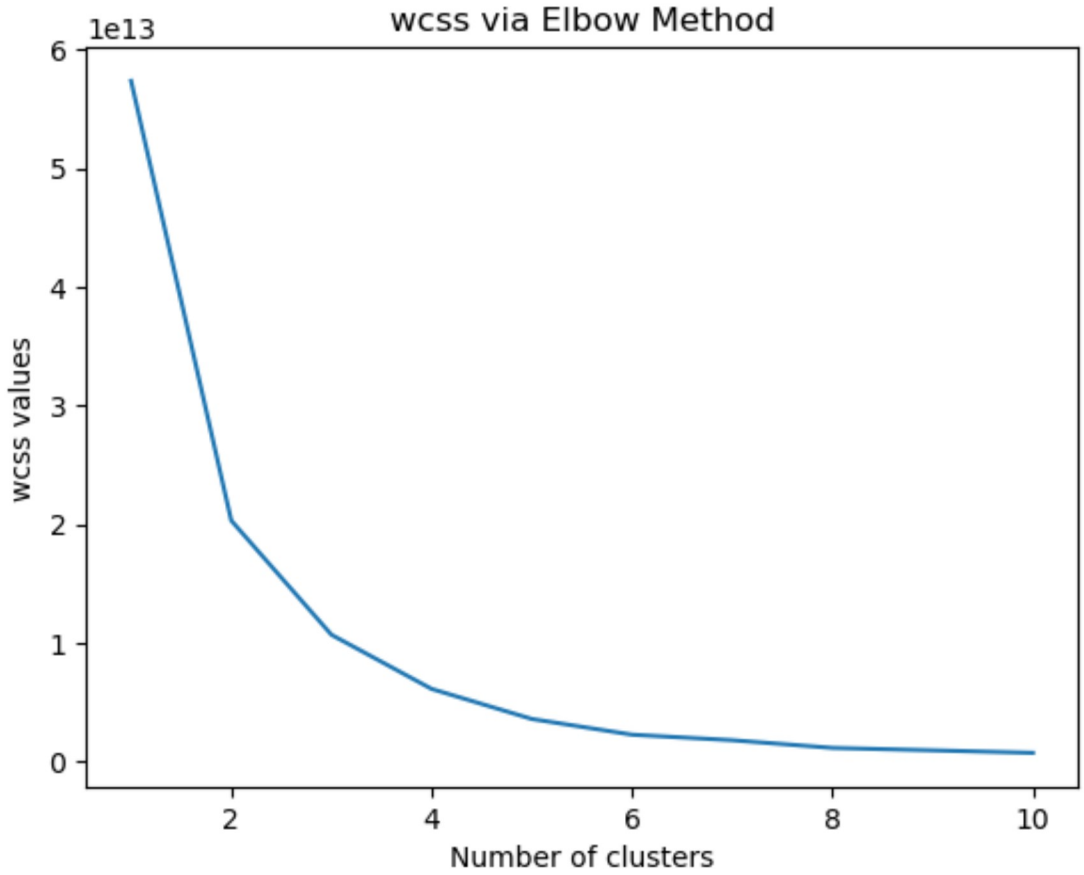
```
[1798456      78]
[1900641      16]
[1900641     101]
[1982389      36]
[1982389      97]
[2002826      17]
[2002826      99]
[2023263      44]
[2023263     109]
[2064137      27]
[2064137      77]
[2105011      20]
[2105011      96]
[2105011      26]
[2105011      78]
[2309381       9]
[2309381     102]
[2452440      18]
[2452440      89]
[2575062      32]
[2575062      83]
[2799869      21]
[2799869      93]]
```

Optimal number of clusters via Elbow Method

In [15]:
```
from sklearn.cluster import KMeans
wcss=[]
for i in range(1,11):
    kmeans=KMeans(n_clusters=i,init='k-means++',random_state=21)
    kmeans.fit(x)
    wcss.append(kmeans.inertia_)
plt.plot(range(1,11),wcss)
plt.title('wcss via Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('wcss values')
```

```
C:\Users\Lenovo\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446: UserWa
rning: KMeans is known to have a memory leak on Windows with MKL, when there are le
ss chunks than available threads. You can avoid it by setting the environment varia
ble OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\Lenovo\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446: UserWa
rning: KMeans is known to have a memory leak on Windows with MKL, when there are le
ss chunks than available threads. You can avoid it by setting the environment varia
ble OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\Lenovo\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446: UserWa
rning: KMeans is known to have a memory leak on Windows with MKL, when there are le
ss chunks than available threads. You can avoid it by setting the environment varia
ble OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\Lenovo\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446: UserWa
rning: KMeans is known to have a memory leak on Windows with MKL, when there are le
ss chunks than available threads. You can avoid it by setting the environment varia
ble OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\Lenovo\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446: UserWa
rning: KMeans is known to have a memory leak on Windows with MKL, when there are le
ss chunks than available threads. You can avoid it by setting the environment varia
ble OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\Lenovo\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446: UserWa
rning: KMeans is known to have a memory leak on Windows with MKL, when there are le
ss chunks than available threads. You can avoid it by setting the environment varia
ble OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\Lenovo\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446: UserWa
rning: KMeans is known to have a memory leak on Windows with MKL, when there are le
ss chunks than available threads. You can avoid it by setting the environment varia
ble OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\Lenovo\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446: UserWa
rning: KMeans is known to have a memory leak on Windows with MKL, when there are le
ss chunks than available threads. You can avoid it by setting the environment varia
ble OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\Lenovo\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446: UserWa
rning: KMeans is known to have a memory leak on Windows with MKL, when there are le
ss chunks than available threads. You can avoid it by setting the environment varia
ble OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\Lenovo\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446: UserWa
rning: KMeans is known to have a memory leak on Windows with MKL, when there are le
ss chunks than available threads. You can avoid it by setting the environment varia
ble OMP_NUM_THREADS=1.
  warnings.warn(
```

Out[15]:
```
Text(0, 0.5, 'wcss values')
```



K Means Model Training on Training Set

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

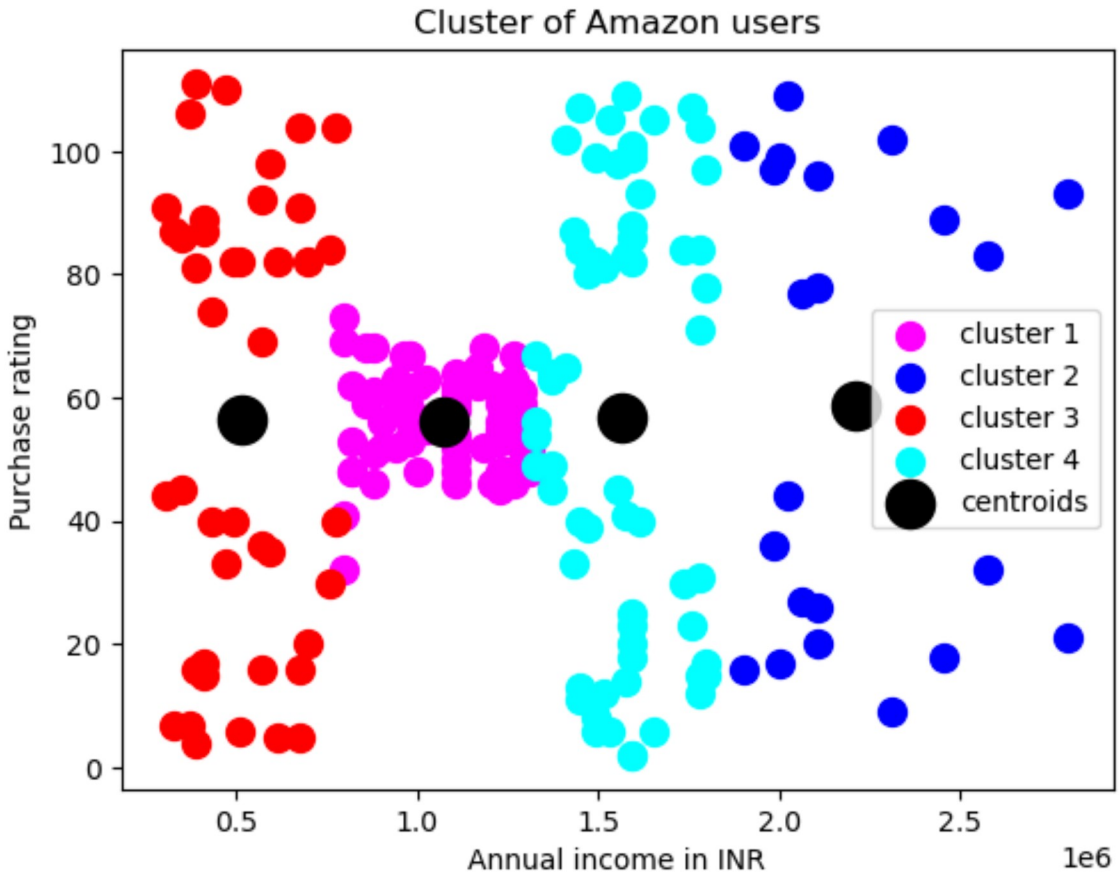
```
C:\Users\Lenovo\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446: UserWa
rning: KMeans is known to have a memory leak on Windows with MKL, when there are le
ss chunks than available threads. You can avoid it by setting the environment varia
ble OMP_NUM_THREADS=1.
  warnings.warn(
```

In [18]:
```
print(y_means)
```

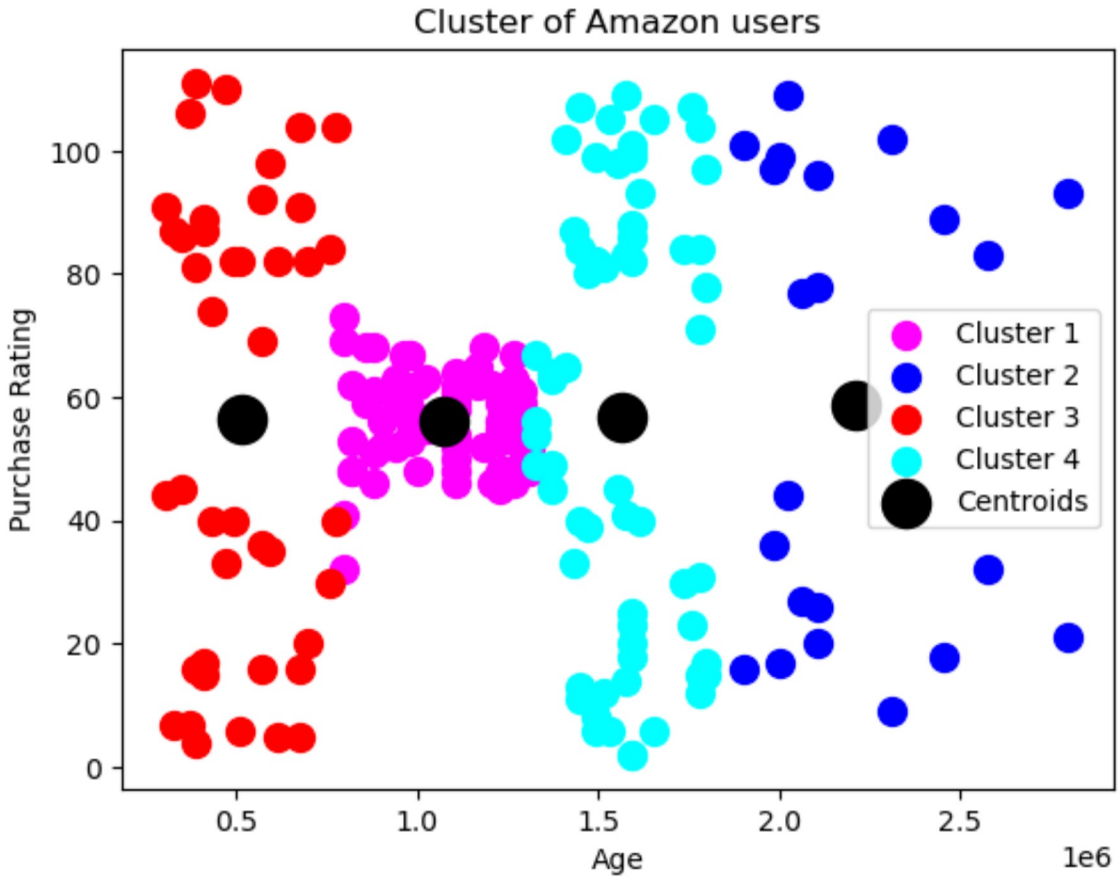
```
[2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
```

Visualising Clusters

In [20]:
```
plt.scatter(x[y_means==0, 0], x[y_means==0, 1], s=100, c='magenta', label='cluster
plt.scatter(x[y_means==1, 0], x[y_means==1, 1], s=100, c='blue', label='cluster 2')
plt.scatter(x[y_means==2, 0], x[y_means==2, 1], s=100, c='red', label='cluster 3')
plt.scatter(x[y_means==3, 0], x[y_means==3, 1], s=100, c='cyan', label='cluster 4')
plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1], s=300, c='b
plt.title('Cluster of Amazon users')
plt.xlabel('Annual income in INR')
plt.ylabel('Purchase rating')
plt.legend()
plt.show()
```



In [23]:
```
plt.scatter(x[y_means == 0, 0], x[y_means == 0, 1], s = 100, c = 'magenta', label =
plt.scatter(x[y_means == 1, 0], x[y_means == 1, 1], s = 100, c = 'blue', label = 'C
plt.scatter(x[y_means == 2, 0], x[y_means == 2, 1], s = 100, c = 'red', label = 'Cl
plt.scatter(x[y_means == 3, 0], x[y_means == 3, 1], s = 100, c = 'cyan', label = 'C
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 300,
plt.title('Cluster of Amazon users')
plt.xlabel('Age')
plt.ylabel('Purchase Rating')
plt.legend()
plt.show()
```


Cluster of Amazon users
Purchase Rating
Age
1e6
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Centroids
0
20
40
60
80
100
0.5
1.0
1.5
2.0
2.5

In [ ]: