In [1]:
```
from sklearn.cluster import KMeans
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from matplotlib import pyplot as plt
%matplotlib inline
```

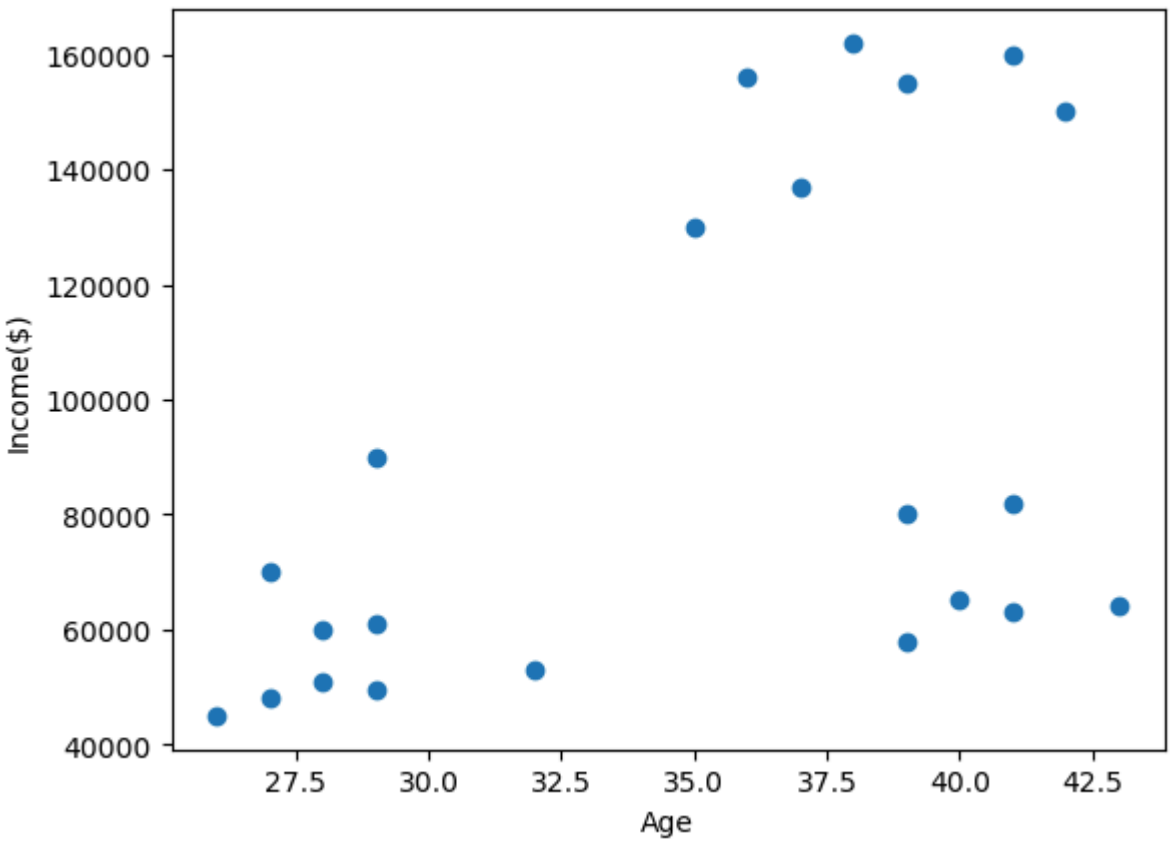
In [2]:
```
df = pd.read_csv("income.csv")
df.head()
```

Out[2]:

	Name	Age	Income($)
0	Rob	27	70000
1	Michael	29	90000
2	Mohan	29	61000
3	Ismail	28	60000
4	Kory	42	150000

In [3]:
```
plt.scatter(df.Age,df['Income($)'])
plt.xlabel('Age')
plt.ylabel('Income($)')
```

Out[3]: Text(0, 0.5, 'Income($)')



In [4]:
```
km = KMeans(n_clusters=3)
y_predicted = km.fit_predict(df[['Age','Income($)']])
y_predicted
```

```
C:\Users\shikh\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_ini
t` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Users\shikh\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a me
mory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the env
ironment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

Out[4]: array([0, 0, 2, 2, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 0, 0, 2])

In [5]:
```
df['cluster']=y_predicted
df.head()
```

Out[5]:

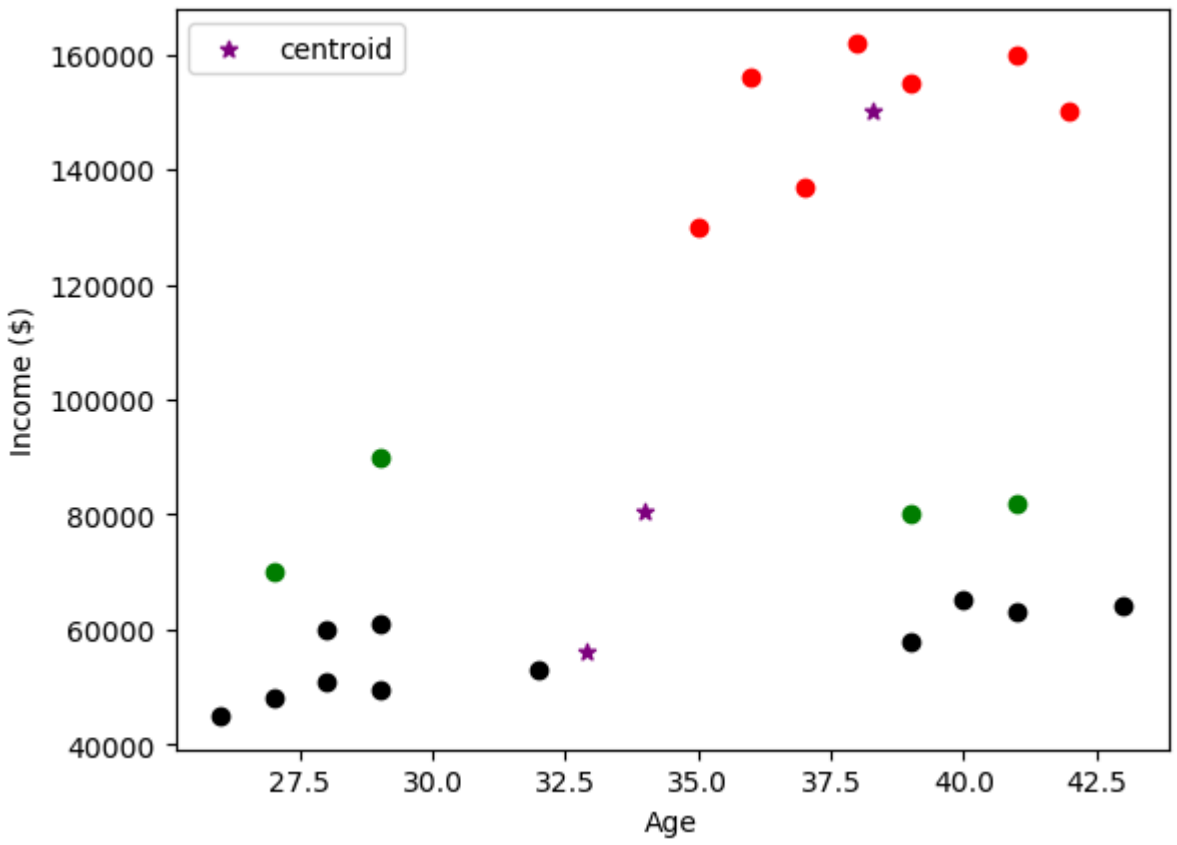
	Name	Age	Income($)	cluster
0	Rob	27	70000	0
1	Michael	29	90000	0
2	Mohan	29	61000	2
3	Ismail	28	60000	2
4	Kory	42	150000	1

In [6]:
```
km.cluster_centers_
```

Out[6]:
```
array([[3.40000000e+01, 8.05000000e+04],
       [3.82857143e+01, 1.50000000e+05],
       [3.29090909e+01, 5.61363636e+04]])
```

In [7]:
```
df1 = df[df.cluster==0]
df2 = df[df.cluster==1]
df3 = df[df.cluster==2]
plt.scatter(df1.Age,df1['Income($)'],color='green')
plt.scatter(df2.Age,df2['Income($)'],color='red')
plt.scatter(df3.Age,df3['Income($)'],color='black')
plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1],color='purple',marker='*',label='centroid')
plt.xlabel('Age')
plt.ylabel('Income ($)')
plt.legend()
```

Out[7]:
```
<matplotlib.legend.Legend at 0x1a7a325c790>
```



Preprocessing using min max scaler

In [8]:
```
scaler = MinMaxScaler()

scaler.fit(df[['Income($)']])
df['Income($)'] = scaler.transform(df[['Income($)']])

scaler.fit(df[['Age']])
df['Age'] = scaler.transform(df[['Age']])
```

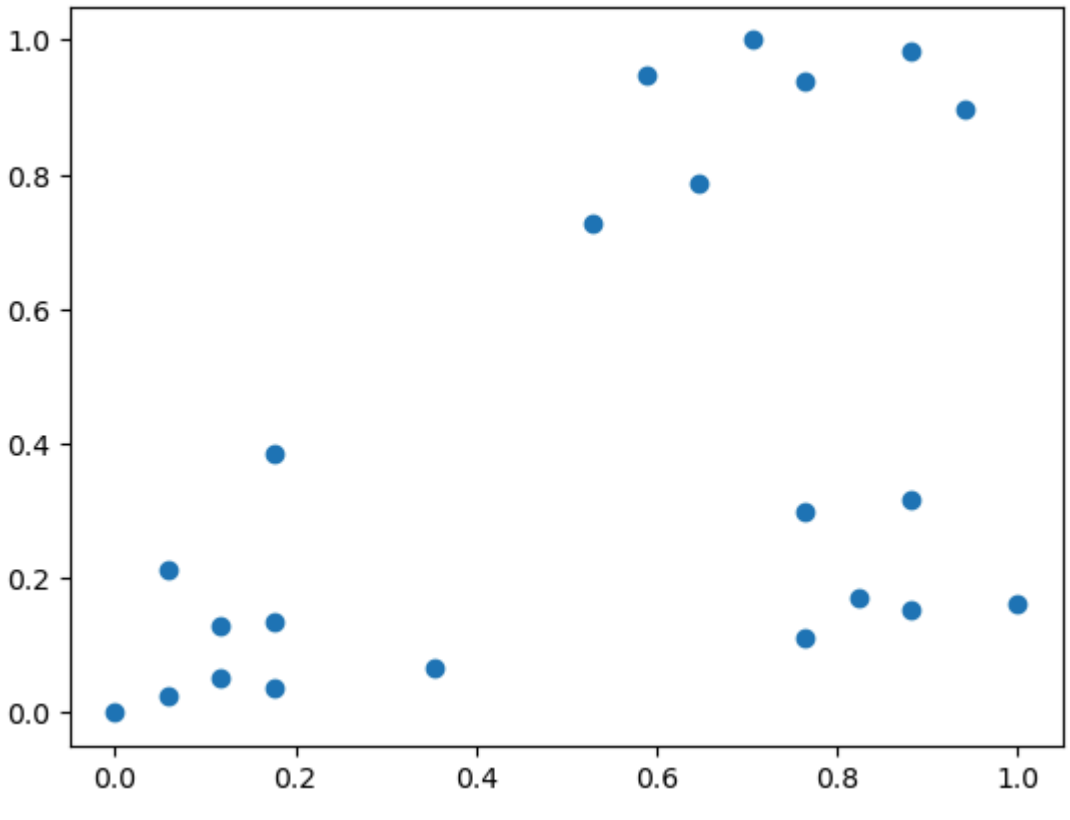
In [9]:
```
df.head()
```

Out[9]:

	Name	Age	Income($)	cluster
0	Rob	0.058824	0.213675	0
1	Michael	0.176471	0.384615	0
2	Mohan	0.176471	0.136752	2
3	Ismail	0.117647	0.128205	2
4	Kory	0.941176	0.897436	1

In [10]:
```
1 plt.scatter(df.Age,df['Income($)'])
```

Out[10]:
```
<matplotlib.collections.PathCollection at 0x1a7a420ba90>
```



In [11]:
```
1 km = KMeans(n_clusters=3)
2 y_predicted = km.fit_predict(df[['Age','Income($)']])
3 y_predicted
```

```
C:\Users\shikh\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Users\shikh\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: 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(
```

Out[11]:
```
array([0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 2, 2, 2, 2, 2, 2])
```

In [12]:
```
1 df['cluster']=y_predicted
2 df.head()
```

Out[12]:

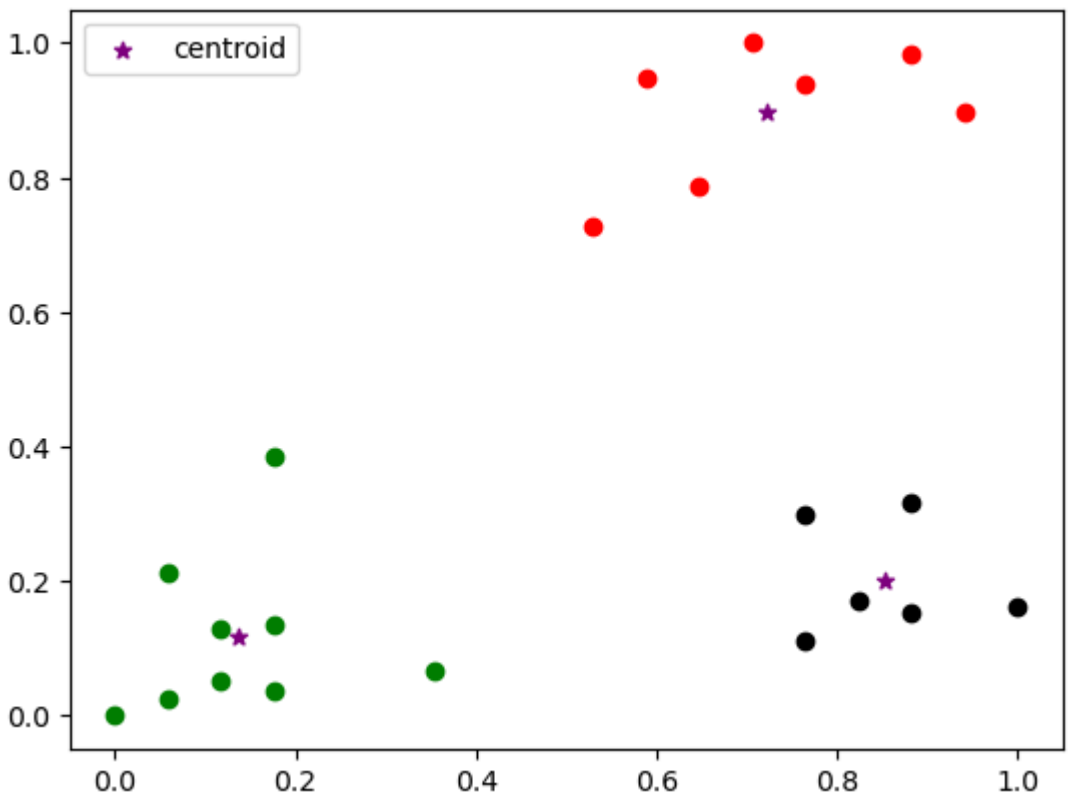
	Name	Age	Income($)	cluster
0	Rob	0.058824	0.213675	0
1	Michael	0.176471	0.384615	0
2	Mohan	0.176471	0.136752	0
3	Ismail	0.117647	0.128205	0
4	Kory	0.941176	0.897436	1

In [13]:
```
1 km.cluster_centers_
```

Out[13]:
```
array([[0.1372549 , 0.11633428],
       [0.72268908, 0.8974359 ],
       [0.85294118, 0.2022792 ]])
```

In [14]:
```
df1 = df[df.cluster==0]
df2 = df[df.cluster==1]
df3 = df[df.cluster==2]
plt.scatter(df1.Age,df1['Income($)'],color='green')
plt.scatter(df2.Age,df2['Income($)'],color='red')
plt.scatter(df3.Age,df3['Income($)'],color='black')
plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1],color='purple',marker='*',label='centroid')
plt.legend()
```

Out[14]: <matplotlib.legend.Legend at 0x1a7a41ca590>



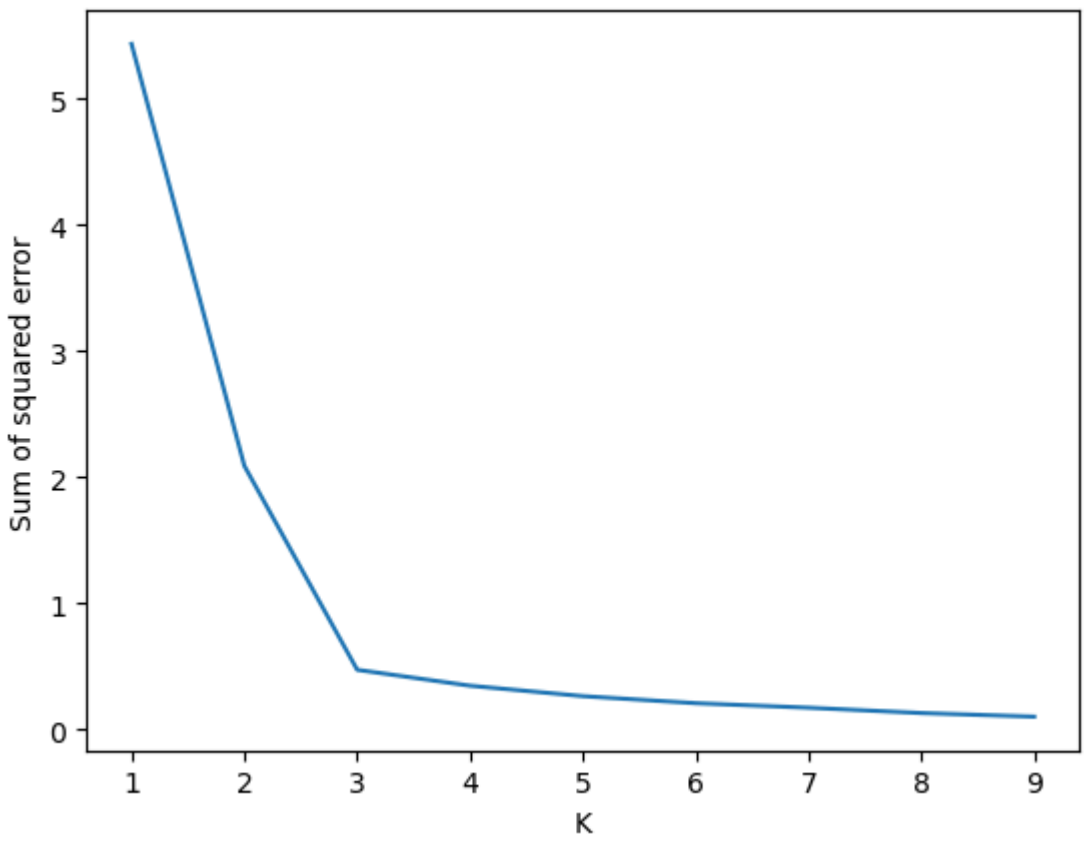
Elbow Plot

To find the perfect value of k

In [ ]:
```
sse = []
k_rng = range(1,10)
for k in k_rng:
    km = KMeans(n_clusters=k)
    km.fit(df[['Age','Income($)']])
    sse.append(km.inertia_)
```

In [16]:
```
plt.xlabel('K')
plt.ylabel('Sum of squared error')
plt.plot(k_rng,sse)
```

Out[16]: [<matplotlib.lines.Line2D at 0x1a7a380aa90>]



DBSCAN

In [17]:
```
df = pd.read_csv("income.csv")
df.head()
```

Out[17]:

	Name	Age	Income($)
0	Rob	27	70000
1	Michael	29	90000
2	Mohan	29	61000
3	Ismail	28	60000
4	Kory	42	150000

In [19]:
```
features = df[['Age', 'Income($)']].values
features
```

Out[19]:
```
array([[    27,  70000],
       [    29,  90000],
       [    29,  61000],
       [    28,  60000],
       [    42, 150000],
       [    39, 155000],
       [    41, 160000],
       [    38, 162000],
       [    36, 156000],
       [    35, 130000],
       [    37, 137000],
       [    26,  45000],
       [    27,  48000],
       [    28,  51000],
       [    29,  49500],
       [    32,  53000],
       [    40,  65000],
       [    41,  63000],
       [    43,  64000],
       [    39,  80000],
       [    41,  82000],
       [    39,  58000]], dtype=int64)
```

In [21]:
```
from sklearn.cluster import DBSCAN
dbscan = DBSCAN(eps=10000, min_samples=2)  # Adjust epsilon and min_samples according to your dataset
clusters = dbscan.fit_predict(features)
```

In [22]:
```
plt.scatter(features[:, 0], features[:, 1], c=clusters, cmap='viridis')
plt.xlabel("Age")
plt.ylabel("Income")
plt.title("DBSCAN Clustering")
plt.show()
```