K-Means Clustering

Importing the libraries

In [28]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Importing the dataset

In [47]:

```
dataset = pd.read_csv(r"C:\Users\Admin\Downloads\Synthetic_Online_Retail.csv
```

In [48]:

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 6 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   TransactionID       1000 non-null   int64  
 1   CustomerSegment     1000 non-null   object 
 2   ProductCategory     1000 non-null   object 
 3   PurchaseAmount ($)  1000 non-null   float64
 4   Quantity            1000 non-null   int64  
 5   PurchaseDate        1000 non-null   object 
dtypes: float64(1), int64(2), object(3)
memory usage: 47.0+ KB
```

In [49]:

```
dataset.shape
```

Out[49]:

```
(1000, 6)
```

In [50]:

```
X = dataset.iloc[:,[3,4]].values
```

In [51]:

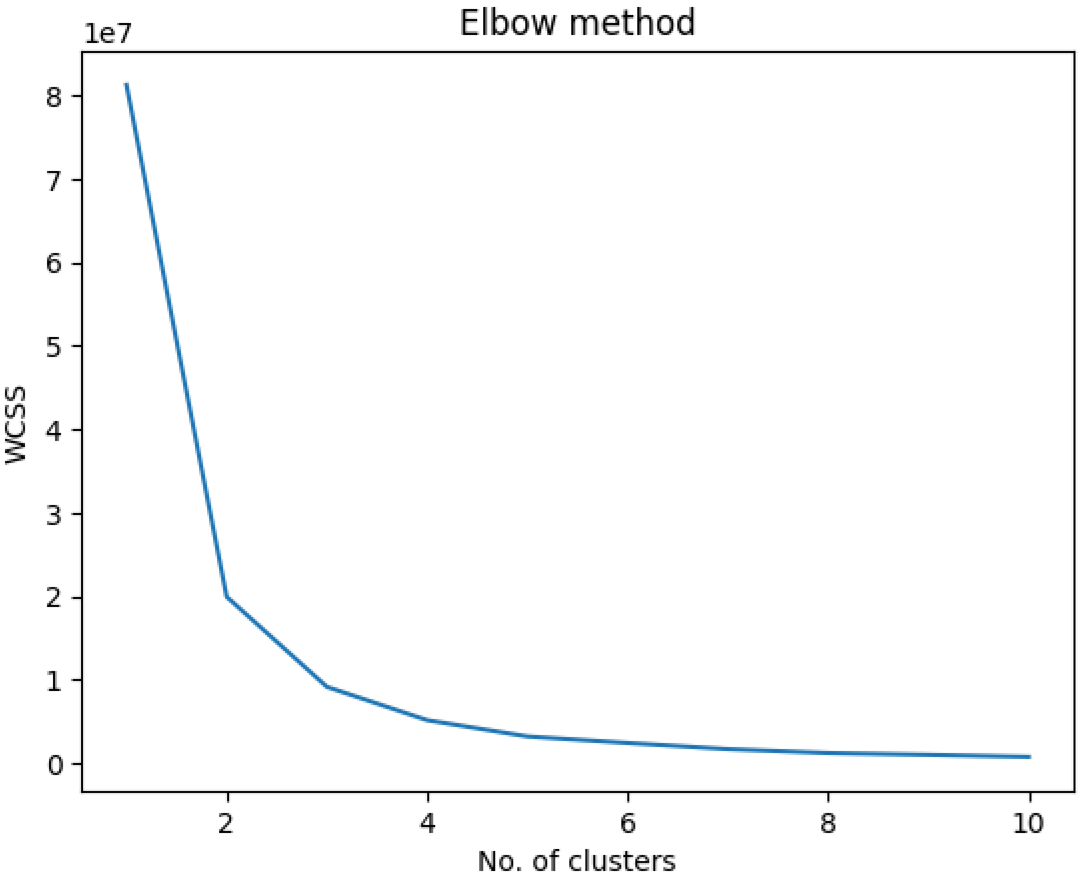
```
print(X)
```

```
[[525.95   4.  ]
 [285.85   9.  ]
 [301.59   7.  ]
 ...
 [954.78   8.  ]
 [918.93   5.  ]
 [ 51.76   2.  ]]
```

Using the elbow method to find the optimal number of clusters

In [52]:

```
 from sklearn.cluster import KMeans
 wcss = []
 for i in range(1,11):
   kmeans = KMeans(n_clusters= i, init='k-means++',random_state= 42)
   kmeans.fit(X)
   wcss.append(kmeans.inertia_)
plt.plot(range(1,11),wcss)
plt.title("Elbow method")
plt.xlabel("No. of clusters")
plt.ylabel("WCSS")
plt.show()
```



Training the K-Means model on the dataset

In [53]:

```
kmeans = KMeans(n_clusters= 5, init='k-means++',random_state= 42)
kmeans.fit(X)
```

Out[53]:

```
▾ KMeans
KMeans(n_clusters=5, random_state=42)
```

Statistics from the initialization run with the lowest SSE are available as attributes of

kmeans after calling .fit()

```
In [54]: # The lowest SSE value
         kmeans.inertia_
```

```
Out[54]: 3218151.3009848516
```

```
In [55]: # Final locations of the centroid
         kmeans.cluster_centers_
```

```
Out[55]: array([[671.96708738,   5.09223301],
                [275.85811594,   4.99033816],
                [882.22143519,   4.92592593],
                [ 91.40374302,   4.81005587],
                [469.47682292,   4.71875   ]])
```

```
In [56]: # The number of iterations required to converge
         kmeans.n_iter_
```

```
Out[56]: 11
```

```
In [57]: #Finally, the cluster assignments are stored as a one-dimensional NumPy arra
         kmeans.labels_
```

```
Out[57]: array([4, 1, 1, 1, 0, 0, 2, 2, 4, 3, 3, 1, 2, 4, 1, 4, 0, 0, 3, 2, 4, 3,
               2, 3, 2, 2, 0, 0, 1, 2, 4, 0, 2, 2, 2, 1, 2, 0, 2, 0, 1, 4, 4, 2,
               1, 0, 1, 2, 4, 3, 3, 3, 4, 3, 3, 2, 2, 4, 2, 3, 0, 0, 4, 4, 4, 4,
               0, 3, 1, 1, 1, 4, 3, 2, 1, 3, 0, 3, 0, 0, 1, 3, 1, 3, 3, 1, 1, 4,
               0, 0, 3, 3, 2, 4, 2, 4, 0, 4, 1, 4, 4, 1, 1, 2, 2, 1, 1, 1, 0, 1,
               2, 1, 4, 0, 2, 4, 4, 2, 3, 2, 3, 2, 0, 1, 2, 3, 1, 4, 1, 0, 3, 3,
               3, 1, 4, 0, 2, 2, 1, 4, 1, 1, 4, 3, 1, 2, 4, 4, 3, 1, 2, 3, 4, 3,
               4, 3, 2, 2, 2, 1, 2, 4, 4, 0, 3, 3, 0, 1, 0, 4, 2, 2, 2, 1, 1, 4,
               3, 3, 4, 4, 2, 1, 1, 3, 2, 2, 2, 2, 4, 4, 1, 0, 4, 4, 2, 2, 2, 0,
               0, 3, 3, 0, 0, 3, 2, 3, 1, 0, 3, 3, 1, 0, 4, 0, 0, 3, 2, 3, 2, 4,
               2, 3, 4, 3, 3, 0, 1, 0, 3, 1, 2, 4, 0, 0, 2, 1, 0, 2, 0, 4, 0, 0,
               0, 3, 1, 2, 1, 0, 4, 4, 3, 1, 2, 2, 0, 4, 0, 0, 4, 3, 0, 0, 2, 3,
               0, 0, 4, 3, 4, 1, 1, 0, 0, 0, 2, 1, 1, 0, 1, 1, 1, 3, 2, 1, 4, 3,
               0, 1, 0, 4, 1, 4, 3, 4, 1, 2, 3, 0, 1, 1, 4, 2, 1, 3, 1, 3, 3, 0,
               3, 3, 0, 1, 3, 1, 0, 1, 4, 3, 0, 1, 0, 2, 0, 1, 1, 1, 3, 1, 2, 4,
               4, 0, 0, 3, 2, 3, 0, 0, 0, 0, 3, 1, 0, 4, 2, 1, 1, 1, 1, 4, 0, 0,
               2, 2, 2, 3, 0, 1, 3, 0, 1, 3, 3, 3, 2, 3, 1, 4, 1, 2, 3, 4, 0, 2,
               0, 2, 3, 4, 4, 2, 3, 3, 1, 3, 1, 2, 4, 0, 0, 1, 3, 1, 1, 0, 4, 3,
               4, 2, 2, 2, 2, 1, 0, 3, 0, 0, 3, 2, 2, 1, 2, 3, 0, 4, 4, 2, 2, 2,
               1, 2, 3, 2, 1, 3, 0, 1, 1, 1, 3, 3, 3, 4, 3, 0, 1, 2, 2, 1, 2, 2,
               2, 0, 3, 0, 0, 0, 1, 2, 2, 1, 3, 1, 2, 0, 0, 4, 0, 1, 4, 2, 2, 2,
               2, 4, 4, 0, 2, 3, 1, 3, 0, 1, 1, 3, 1, 2, 1, 0, 0, 2, 3, 2, 0, 2,
               2, 1, 3, 1, 4, 3, 3, 4, 3, 3, 4, 3, 4, 4, 4, 2, 4, 4, 4, 2, 2, 2,
               0, 3, 1, 3, 2, 0, 0, 1, 2, 1, 3, 4, 0, 4, 2, 3, 4, 4, 1, 2, 0, 0,
               2, 0, 1, 0, 1, 1, 3, 0, 0, 1, 0, 2, 3, 1, 0, 1, 0, 3, 4, 0, 0, 2,
               2, 0, 0, 3, 4, 3, 2, 1, 4, 1, 2, 2, 4, 2, 1, 4, 2, 1, 2, 3, 3, 2,
               3, 2, 2, 3, 4, 1, 4, 3, 1, 1, 4, 2, 2, 0, 4, 2, 1, 0, 2, 1, 0, 2,
               2, 0, 1, 1, 3, 0, 3, 2, 3, 3, 4, 0, 3, 4, 4, 1, 3, 2, 4, 3, 2, 2,
               0, 4, 2, 2, 0, 4, 0, 0, 1, 4, 2, 0, 4, 0, 3, 4, 2, 1, 1, 2, 1, 1,
               2, 0, 0, 2, 3, 0, 4, 2, 4, 0, 4, 4, 0, 0, 0, 4, 4, 0, 2, 0, 2, 2,
               4, 0, 2, 0, 2, 0, 4, 1, 4, 2, 0, 4, 2, 0, 4, 0, 0, 1, 4, 4, 4, 4,
               4, 1, 4, 3, 0, 2, 0, 1, 3, 1, 4, 1, 4, 1, 2, 4, 1, 2, 1, 3, 0, 0,
               0, 1, 4, 1, 3, 2, 0, 1, 0, 1, 2, 3, 3, 4, 0, 0, 3, 1, 4, 1, 2, 3,
               4, 0, 1, 0, 2, 1, 2, 4, 1, 4, 1, 0, 2, 2, 2, 2, 0, 1, 4, 3, 0, 4,
               3, 3, 4, 0, 4, 0, 4, 0, 1, 3, 0, 4, 3, 4, 3, 4, 4, 0, 2, 1, 2, 4,
               0, 0, 0, 4, 4, 3, 3, 1, 1, 4, 3, 0, 2, 2, 4, 0, 2, 4, 1, 2, 4, 4,
               3, 3, 2, 0, 1, 1, 4, 0, 1, 2, 3, 1, 0, 1, 2, 0, 2, 3, 3, 1, 4, 0,
               2, 0, 2, 0, 1, 1, 0, 1, 3, 4, 2, 3, 2, 2, 4, 3, 3, 0, 0, 2, 0, 3,
               3, 0, 1, 1, 4, 3, 1, 4, 1, 3, 0, 2, 1, 0, 4, 4, 2, 0, 2, 3, 3, 2,
               4, 1, 0, 2, 2, 1, 2, 3, 3, 0, 3, 1, 0, 1, 1, 1, 4, 4, 2, 0, 4, 2,
               3, 1, 2, 4, 2, 4, 2, 4, 4, 4, 0, 4, 2, 1, 3, 4, 0, 2, 1, 0, 1, 0,
               2, 1, 4, 2, 2, 4, 4, 1, 4, 2, 0, 1, 1, 1, 3, 3, 4, 3, 3, 4, 3, 3,
               1, 1, 4, 3, 0, 2, 0, 4, 2, 0, 3, 4, 2, 1, 3, 0, 2, 3, 4, 1, 0, 1,
               1, 2, 4, 0, 4, 1, 0, 3, 4, 1, 0, 2, 2, 4, 1, 4, 2, 1, 2, 1, 2, 2,
               2, 1, 3, 4, 1, 3, 4, 4, 0, 0, 3, 2, 1, 4, 4, 2, 3, 4, 0, 1, 0, 2,
               2, 3, 1, 1, 4, 1, 1, 2, 2, 3], dtype=int32)
```

Creating Output labels for Generating Graph

```
In [58]: y_kmeans = kmeans.fit_predict(X)
```

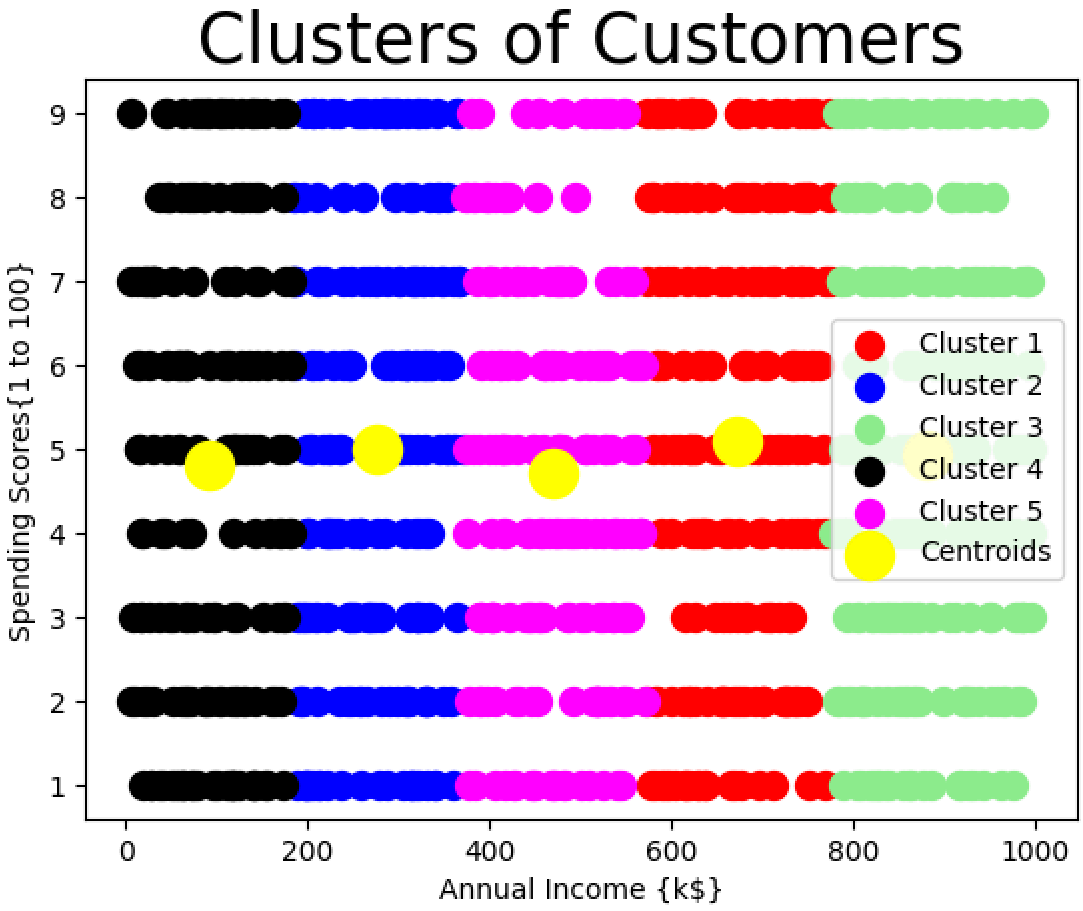
```
In [59]: print(y_kmeans)
```

```
[4 1 1 1 0 0 2 2 4 3 3 1 2 4 1 4 0 0 3 2 4 3 2 3 2 2 0 0 1 2 4 0 2 2 2 1 2
 0 2 0 1 4 4 2 1 0 1 2 4 3 3 3 4 3 3 2 2 4 2 3 0 0 4 4 4 4 0 3 1 1 1 4 3 2
 1 3 0 3 0 0 1 3 1 3 3 1 1 4 0 0 3 3 2 4 2 4 0 4 1 4 4 1 1 2 2 1 1 1 0 1 2
 1 4 0 2 4 4 2 3 2 3 2 0 1 2 3 1 4 1 0 3 3 3 1 4 0 2 2 1 4 1 1 4 3 1 2 4 4
 3 1 2 3 4 3 4 3 2 2 2 1 2 4 4 0 3 3 0 1 0 4 2 2 2 1 1 4 3 3 4 4 2 1 1 3 2
 2 2 2 4 4 1 0 4 4 2 2 2 0 0 3 3 0 0 3 2 3 1 0 3 3 1 0 4 0 0 3 2 3 2 4 2 3
 4 3 3 0 1 0 3 1 2 4 0 0 2 1 0 2 0 4 0 0 0 3 1 2 1 0 4 4 3 1 2 2 0 4 0 0 4
 3 0 0 2 3 0 0 4 3 4 1 1 0 0 0 2 1 1 0 1 1 1 3 2 1 4 3 0 1 0 4 1 4 3 4 1 2
 3 0 1 1 4 2 1 3 1 3 3 0 3 3 0 1 3 1 0 1 4 3 0 1 0 2 0 1 1 1 3 1 2 4 4 0 0
 3 2 3 0 0 0 0 3 1 0 4 2 1 1 1 1 4 0 0 2 2 2 3 0 1 3 0 1 3 3 3 2 3 1 4 1 2
 3 4 0 2 0 2 3 4 4 2 3 3 1 3 1 2 4 0 0 1 3 1 1 0 4 3 4 2 2 2 2 1 0 3 0 0 3
 2 2 1 2 3 0 4 4 2 2 2 1 2 3 2 1 3 0 1 1 1 3 3 3 4 3 0 1 2 2 1 2 2 2 0 3 0
 0 0 1 2 2 1 3 1 2 0 0 4 0 1 4 2 2 2 2 4 4 0 2 3 1 3 0 1 1 3 1 2 1 0 0 2 3
 2 0 2 2 1 3 1 4 3 3 4 3 3 4 3 4 4 4 2 4 4 4 2 2 2 0 3 1 3 2 0 0 1 2 1 3 4
 0 4 2 3 4 4 1 2 0 0 2 0 1 0 1 1 3 0 0 1 0 2 3 1 0 1 0 3 4 0 0 2 2 0 0 3 4
 3 2 1 4 1 2 2 4 2 1 4 2 1 2 3 3 2 3 2 2 3 4 1 4 3 1 1 4 2 2 0 4 2 1 0 2 1
 0 2 2 0 1 1 3 0 3 2 3 3 4 0 3 4 4 1 3 2 4 3 2 2 0 4 2 2 0 4 0 0 1 4 2 0 4
 0 3 4 2 1 1 2 1 1 2 0 0 2 3 0 4 2 4 0 4 4 0 0 0 4 4 0 2 0 2 2 4 0 2 0 2 0
 4 1 4 2 0 4 2 0 4 0 0 1 4 4 4 4 4 1 4 3 0 2 0 1 3 1 4 1 4 1 2 4 1 2 1 3 0
 0 0 1 4 1 3 2 0 1 0 1 2 3 3 4 0 0 3 1 4 1 2 3 4 0 1 0 2 1 2 4 1 4 1 0 2 2
 2 2 0 1 4 3 0 4 3 3 4 0 4 0 4 0 1 3 0 4 3 4 3 4 4 0 2 1 2 4 0 0 0 4 4 3 3
 1 1 4 3 0 2 2 4 0 2 4 1 2 4 4 3 3 2 0 1 1 4 0 1 2 3 1 0 1 2 0 2 3 3 1 4 0
 2 0 2 0 1 1 0 1 3 4 2 3 2 2 4 3 3 0 0 2 0 3 3 0 1 1 4 3 1 4 1 3 0 2 1 0 4
 4 2 0 2 3 3 2 4 1 0 2 2 1 2 3 3 0 3 1 0 1 1 1 4 4 2 0 4 2 3 1 2 4 2 4 2 4
 4 4 0 4 2 1 3 4 0 2 1 0 1 0 2 1 4 2 2 4 4 1 4 2 0 1 1 1 3 3 4 3 3 4 3 3 1
 1 4 3 0 2 0 4 2 0 3 4 2 1 3 0 2 3 4 1 0 1 1 2 4 0 4 1 0 3 4 1 0 2 2 4 1 4
 2 1 2 1 2 2 2 1 3 4 1 3 4 4 0 0 3 2 1 4 4 2 3 4 0 1 0 2 2 3 1 1 4 1 1 2 2
 3]
```

Visualising the clusters

In [60]:

```
plt.scatter(X[y_kmeans == 0,0],X[y_kmeans == 0,1],s=100, c = 'red', label ="
plt.scatter(X[y_kmeans == 1,0],X[y_kmeans == 1,1],s=100, c = 'blue', label =
plt.scatter(X[y_kmeans == 2,0],X[y_kmeans == 2,1],s=100, c = 'lightgreen', l
plt.scatter(X[y_kmeans == 3,0],X[y_kmeans == 3,1],s=100, c = 'black', label
plt.scatter(X[y_kmeans == 4,0],X[y_kmeans == 4,1],s=100, c = 'magenta', labe
plt.scatter(kmeans.cluster_centers_[:,0],kmeans.cluster_centers_[:,1],s = 30
plt.title("Clusters of Customers",size = 25)
plt.xlabel("Annual Income {k$}")
plt.ylabel("Spending Scores{1 to 100}")
plt.legend()
plt.show()
```



Internal Evaluation of Cluster

DB Score (lower is better)

In [61]:

```
from sklearn.metrics import davies_bouldin_score
davies_bouldin_score(X,y_kmeans)
```

Out[61]:

```
np.float64(0.497928242247099975)
```

External Evaluation

Homogenity Score (higher is better)

In [62]:

```
y_pred = kmeans.predict(X)
```

In [63]:

```
from sklearn.metrics.cluster import homogeneity_score
homogeneity_score(y_kmeans,y_pred)
```

Out[63]:

```
np.float64(1.0)
```

In [ ]:

This notebook was converted with convert.ploomber.io