In [1]:
```
# import the library
import pandas as pd
import numpy as np
```

Read and understand the data

In [2]:
```
# load the data
df = pd.read_csv('Mall_Customers.csv')
```

In [3]:
```
# first five rows
df.head()
```

Out[3]:

	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

In [4]:
```
# data types
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   CustomerID              200 non-null    int64 
 1   Genre                   200 non-null    object
 2   Age                     200 non-null    int64 
 3   Annual Income (k$)      200 non-null    int64 
 4   Spending Score (1-100)  200 non-null    int64 
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
```

Visualise the data

In [5]:
```
# import library
import matplotlib.pyplot as plt
plt.scatter(df['Annual Income (k$)'] , df['Spending Score (1-100)'])
plt.xlabel('Annual Income')
plt.ylabel('Spending Score')
plt.title('Unlabelled Mall customer data')
plt.show()
```

Define the value of X

In [6]:
```
# define X , unsupervised learning no labels , no Y
X = df[['Annual Income (k$)','Spending Score (1-100)']]
X.head()
```

Out[6]:

	Annual Income (k$)	Spending Score (1-100)
0	15	39
1	15	81
2	16	6
3	16	77
4	17	40

Build the model

In [7]:
```
# import the library to build the model
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=5 , random_state=42)
```

Train the Model

In [8]:
```
kmeans.fit(X)
```

Out[8]:
```
KMeans(n_clusters=5, random_state=42)
```

Predict

In [9]:
```
pred = kmeans.predict(X)
```

In [10]:
```
df['clusters'] = pred
```

In [11]:
```
df.head()
```

Out[11]:

	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)	clusters
0	1	Male	19	15	39	2
1	2	Male	21	15	81	3
2	3	Female	20	16	6	2
3	4	Female	23	16	77	3
4	5	Female	31	17	40	2

In [12]:
```
## Show the value for cluster 2 for first ten rows
df[df['clusters']==2].head(10)
```

Out[12]:

	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)	clusters
0	1	Male	19	15	39	2
2	3	Female	20	16	6	2
4	5	Female	31	17	40	2
6	7	Female	35	18	6	2
8	9	Male	64	19	3	2
10	11	Male	67	19	14	2
12	13	Female	58	20	15	2
14	15	Male	37	20	13	2
16	17	Female	35	21	35	2
18	19	Male	52	23	29	2

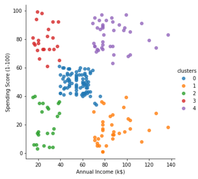
In [13]:
```
## Show the value for cluster 2 and only Annual Income (k$)
df[df['clusters']==2]['Annual Income (k$)'].head(10)
```

Out[13]:
```
0     15
2     16
4     17
6     18
8     19
10    19
12    20
14    20
16    21
18    23
Name: Annual Income (k$), dtype: int64
```

In [14]:
```
## Show the value for cluster 2 and only Spending Score (1-100)
df[df['clusters']==2]['Spending Score (1-100)'].head(10)
```

Out[14]:
```
0     39
2      6
4     40
6      6
8      3
10    14
12    15
14    13
16    35
18    29
Name: Spending Score (1-100), dtype: int64
```

In [15]:
```
# seaborn is a datavisualisation library
# visualising the cluster
import seaborn as sns
sns.lmplot(data=df , x='Annual Income (k$)'  , y='Spending Score (1-100)' , hue='clusters',fit_reg=False)
plt.show()
```



In [16]:
```
## Cordinates of centroids
kmeans.cluster_centers_
```

Out[16]:
```
array([[55.2962963 , 49.51851852],
       [88.2       , 17.11428571],
       [26.30434783, 20.91304348],
       [25.72727273, 79.36363636],
       [86.53846154, 82.12820513]])
```

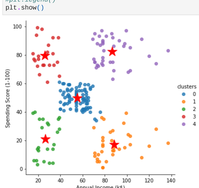
In [17]:
```
# X_coordinates of centroid
kmeans.cluster_centers_[:,0]
```

Out[17]:
```
array([55.2962963 , 88.2       , 26.30434783, 25.72727273, 86.53846154])
```

In [18]:
```
# Y_coordinates of centroid
kmeans.cluster_centers_[:,1]
```

Out[18]:
```
array([49.51851852, 17.11428571, 20.91304348, 79.36363636, 82.12820513])
```

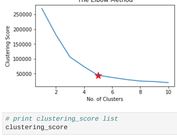
In [19]:
```
# visualise with centroids
sns.lmplot(data=df , x='Annual Income (k$)'  , y='Spending Score (1-100)' , hue='clusters',fit_reg=False)
plt.scatter(kmeans.cluster_centers_[:,0],kmeans.cluster_centers_[:,1], c='red',marker='*',s=400 , label='centroid')
#plt.legend()
plt.show()
```



In [20]:
```
clustering_score = [] # empty list using square brackets
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'random', random_state = 42)
    kmeans.fit(X)
    clustering_score.append(kmeans.inertia_) # inertia_ = Sum of squared distances of samples to their closest cluster center.


plt.figure(figsize=(5,3))
plt.plot(range(1, 11), clustering_score)
plt.scatter(5,clustering_score[4], s = 200, c = 'red', marker='*')
plt.title('The Elbow Method')
plt.xlabel('No. of Clusters')
plt.ylabel('Clustering Score')
plt.show()
```

```
C:\Users\win10\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(
```



In [21]:
```
# print clustering_score list
clustering_score
```

Out[21]:
```
[269981.28,
 181363.59595959596,
 106348.37306211119,
 73679.78903948834,
 44448.45544793371,
 37265.86520484346,
 30273.394312070042,
 25095.703209097548,
 23287.318947718948,
 19710.039271660018]
```

In [22]:
```
# print the first value
clustering_score[0]
```

Out[22]:
```
269981.28
```

In [23]:
```
# print the fifth value
clustering_score[4]
```

Out[23]:
```
44448.45544793371
```

In [ ]:
```

```