In [28]:
```
import pandas as pd
import numpy as np
```

In [29]:
```
df = pd.read_csv("shopping_data.csv")
```

In [30]:
```
df.head()
```

Out[30]:

	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 [31]:
```
df = df.drop(["CustomerID", "Genre"],axis=1)
```

In [54]:
```
df.head()
```

In [33]:
```
df.isnull().sum()
```

Out[33]:
```
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

In [34]:
```
from sklearn.preprocessing import MinMaxScaler

mn = MinMaxScaler()
df_sc = mn.fit_transform(df)
```

In [35]:
```
df_sc_df = pd.DataFrame(df_sc, columns=df.columns, index=df.index)
```

In [55]:
```
df_sc_df.head()
```

In [37]:
```
from sklearn.cluster import KMeans
```

In [38]:
```
km = KMeans(n_clusters=4)
```

In [56]:
```
km.fit(df_sc_df)
```

In [53]:
```
km.labels_
```

In [43]:
```
df["cluster_Nos"] = km.labels_
```

In [52]:
```
df.head(10)
```

In [46]:
```
## evaluate the clustering
#K-Means: Inertia
#Inertia measures how well a dataset was clustered by K-Means.
#It is calculated by measuring the distance between each data point and its centroid,
#squaring this distance, and summing these squares across one cluster.
#A good model is one with low inertia AND a low number of clusters ( K ).
km.inertia_
```

Out[46]:
```
12.65028767622991
```

Inertia is the sum of squared distance of samples to their closest cluster center. We would like this number to be as small as possible. But, if we choose K that is equal to the number of samples we will get inertia=0.

In [45]:
```
from sklearn.metrics import silhouette_score
```

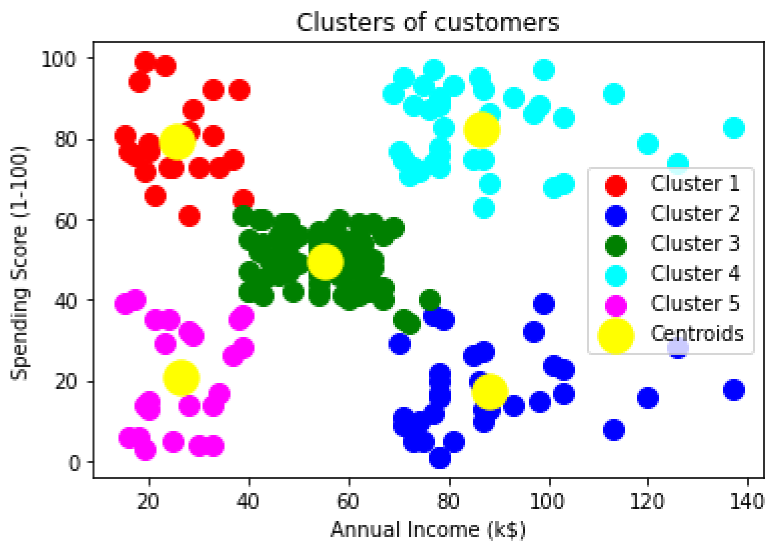
In [19]:
```
#Mean distance between the observation and all other data points in the same cluster.
#mean intra-cluster distance
silhouette_score(df_sc_df, km.labels_ )
```

Out[19]:
```
0.392319202055722
```

In [24]:
```
# X contains two features Annual income and spending score
X = df.iloc[:, [1, 2]].values
m=X.shape[0]
n=X.shape[1]
n_iter=100
K=5
```

In [25]:
```
#Kmeans ++ algo for training and predicting the model with given dataset x
#no of clusters=5 to form 5 clusters of customers based on their spending scores
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters = 5, init = 'k-means++', random_state = 1)
y_kmeans = kmeans.fit_predict(X)
# Visualising the clusters
plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 100, c = 'red', label = 'Cluster 1')
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 100, c = 'blue', label = 'Cluster 2')
plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 100, c = 'green', label = 'Cluster 3')
plt.scatter(X[y_kmeans == 3, 0], X[y_kmeans == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4')
plt.scatter(X[y_kmeans == 4, 0], X[y_kmeans == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 300, c = 'yellow', label = 'Centr
plt.title('Clusters of customers')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
```

```
C:\Users\solun\Anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1332: UserWarning: KMeans is known to hav
e a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by set
ting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```



In [51]:
```
 X[y_kmeans == 3,0]
```

The output image is clearly showing the five different clusters with different colors. The clusters are formed between two parameters of the dataset; Annual income of customer and Spending. We can change the colors and labels as per the requirement or choice. We can also observe some points from the above patterns, which are given below: Cluster5 shows the customers with average salary and average spending so we can categorize these customers as careful Cluster2 shows the customer has a high income but low spending, so we can categorize them as careful. Cluster3 shows the low income and also low spending so they can be categorized as sensible. Cluster1 shows the customers with low income with very high spending so they can be categorized as careless. Cluster4 shows the customers with high income and high spending so they can be categorized as target, and these customers can be the most profitable customers for the mall owner.

In [50]:
```
#using Elbow Method
#In cluster analysis, the elbow method is a heuristic used in
#determining the number of clusters in a data set.
from sklearn.cluster import KMeans
wcss = []
for i in range(1, 30):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 30), wcss)plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```