Customer classification

Introduction

Given the large number of customers and the large amount of purchasing and sales data, it is difficult to find a clear and understandable classification of customers.

In this part we will try to analyze customer behavior in four separate analyzes using the K-means cleustering model.Objective is to understand customer behavior according to different axes of analysis:

Analysis 1: The choice of sales in relation to product categories or general sales activities.
Analysis 2: Using sales volume and market membership.
Analysis 3: Relationship report with purchasing volume from suppliers.
Analysis 4: The volume of purchases made by members of the company who specialize in delivering products to customers.

Cluster analysis

Cluster analysis, also called data segmentation, has a variety of goals. All relate to grouping or segmenting a collection of objects into subsets or clusters, such that those wothin each cluster are more closely related to one another than objects assigned to different clusters. The clustering model starts works like the following :

- The creation of cluster centers.
- Each cluster center is replaced by the coordinate-wise average of all data points that are closest to it.
- It alternates the previous steps until convergence.

K-means clustering

The K-means alghorithm is one of the most popular iterative descent clustering methods. It is intended for sitations in which all variables are of the quantitative type, and squared Euclidean distance :

$$d(x_i, x_{i'}) = \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 = ||x_i - x_{i'}||^2$$

is chosen as the dissimilarity measure. Note that weighted Euclidean distance can be used by redefining the x_{ij} values. The within point scatter can be written as :

$$W(C) = \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \sum_{C(i')=k} ||x_i - x_{i'}||^2 = \sum_{k=1}^{K} N_k \sum_{C(i)=k} ||x_i - \bar{x}_k||^2$$

where $\bar{x}_k = (\bar{x}_{1k}, \bar{x}_{2k}, \ldots\ldots, \bar{x}_{pk})$ is the mean vector associated with the $k-th$ cluster, and $N_k = \sum_{i=1}^{N} I(C(i) = k)$. Thus, the criterion is minimized by each cluster the

average dissimilarity of the observations from the cluster mean, as defined by the points in taht cluster, is minimized.

An iterative descent alghorithm for solving : \

1 - For a given cluster assignment C, the total cluster variance is minimized with the respect to $\{m_1, m_2, \ldots, m_K\}$ yielding the means of the currently assigned clusters.

2 - Given a current set of means $\{m_1, m_2, \ldots, m_K\}$, is minimized by assigning each observation to the closest (current) cluster mean. That is :

$$C(i) = \underset{1 \leq k \leq K}{\arg\min} ||x_i - m_k||^2$$

3 - Steps 1 and 2 are iterated until the assignments do not change.

K-means clustering with analyse N°1

In analysis number 1 we will classify the customers on the basis of the sales volume made by the company and also by the categories which can be used either for the identification of the types of products, or globally to know the field of sales activity.

The data structure is as follows:

- Clients coded in 6 int to 6 digits.
- Sales in floats
- Categories in int from 1 to 502

The stages of setting up the classification are:

- Import data: Customer code, sales, categories and category names.
- Aggregation of data by customer code to have sales volumes by unique customers.
- Search for the optimal number of clusters with The elbow method
- Verification of the optimal number of clusters with The silhouette coefficient
- Application of K-means classification
- Data smoothing and prediction with the model
- Displaying classification results

With the Elbow method we have a significant decrease in the Within-Cluster Sum of Squares WCSS in cluster 2. After cluster 2 decrease is not important. We conclude that we have 2 clusters.

The silhouette coefficient graphic shows a pick in cluster 2. We concluded that in cluster 2, we reach the maximum coefficient value.

Centroids lecture : The cluster figure shows two importants categories of customers :

- The first one : Who spent 327270,5 and have activity categories corresponding to 'HYGIENE GENERALE INSECTICIDES - RAMPANT - AEROSOL' (Key number 15).

- The second one : Who spent 11105,94 and have activity categories corresponding to 'MATERIEL MANUEL ET CHARIOT - MATERIEL SOL MANUEL - MANCHE - TELESCOPIQUE' (category number 29).

General lecture :

- The first customer category spent between 1 and 180000 euros and all activities are present, except 500, 501 and 502 which are 'VAISSELLE A USAGE UNIQUE GOBELETS - GOBELET BOISSON CHAUDE' for category number 500, 'HYGIENE ET SOINS ADULTES - INCONTINENCE MODERE A FORTE - PROTECTION RESPIRANTE' for category number 501 and 'HYGIENE ET SOINS ADULTES - INCONTINENCE MODERE A FORTE' for category number 502.
- The second customer category spent more than 180000 euros and less than 700000 euros. The product categories include keys from 0 to 100.

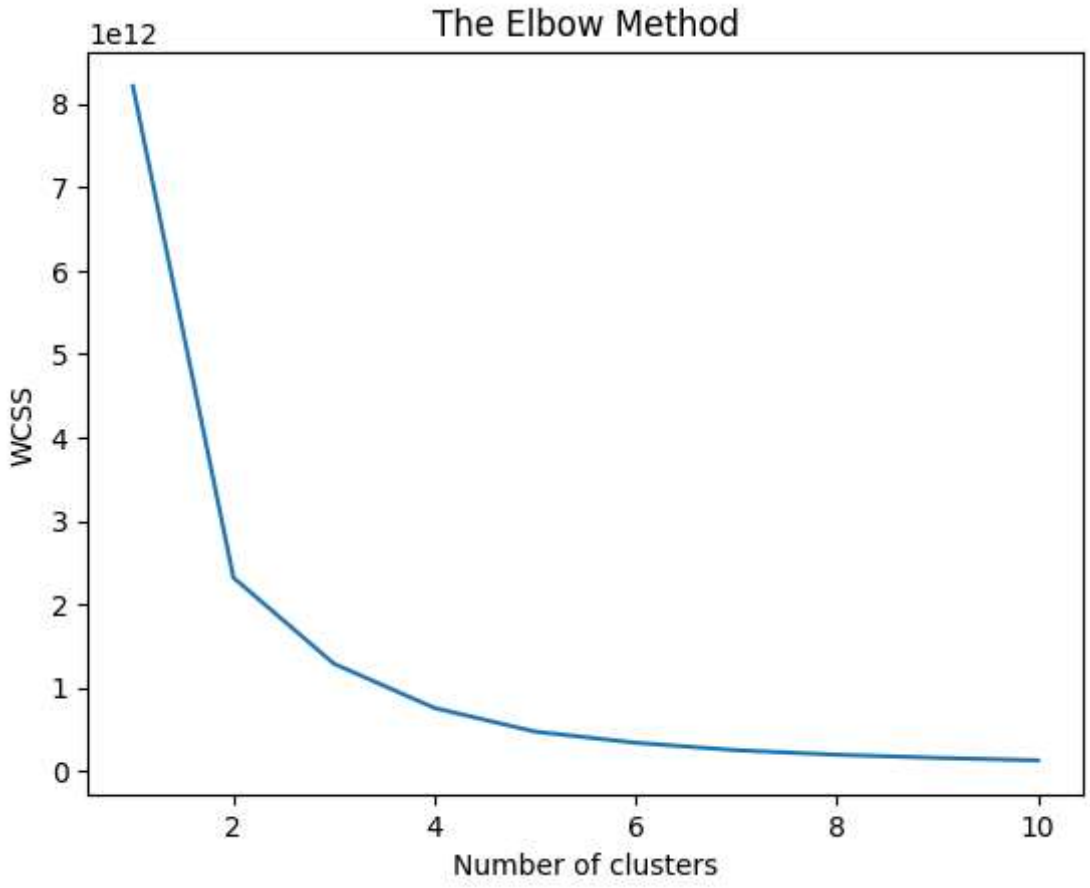
In [5]:
```
# Import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import silhouette_score

# Import dataset
dataset = pd.read_excel('C:/Autres/transfert/Financial Markets/Data modelling/Da
                        sheet_name = 'Base_KPI')

data_customers = dataset[['Customer code','Customer name','Sales amount','Catego
data1_Kmeans = data_customers[['Customer code','Sales amount','Category key']]

# Group data by customer key
data1_Kmeans = data1_Kmeans.groupby('Customer code', as_index=False, sort=False)

# Transform data : Sales amount and Category key into float and int
data1_Kmeans['Sales amount'] = data1_Kmeans['Sales amount'].astype(float)
data1_Kmeans['Category key'] = data1_Kmeans['Category key'].astype(int)
data_Kmeans = data1_Kmeans[['Sales amount','Category key']]

print(data_Kmeans)
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_16000\3266444764.py:11: FutureWarning:
Passing a set as an indexer is deprecated and will raise in a future version. Use
a list instead.
  data_customers = dataset[{'Customer code','Customer name','Sales amount','Categ
ory key','Category name'}]
C:\Users\DELL\AppData\Local\Temp\ipykernel_16000\3266444764.py:12: FutureWarning:
Passing a set as an indexer is deprecated and will raise in a future version. Use
a list instead.
  data1_Kmeans = data_customers[{'Customer code','Sales amount','Category key'}]
```

```
      Category key  Sales amount
0                1      29757.33
1                1      72455.27
2                1      92792.05
3                1     226418.13
4                1      15253.52
...            ...           ...
4264             1        732.38
4265             1        175.67
4266             1        187.56
4267             1          0.00
4268             1       1196.80

[4269 rows x 2 columns]
```

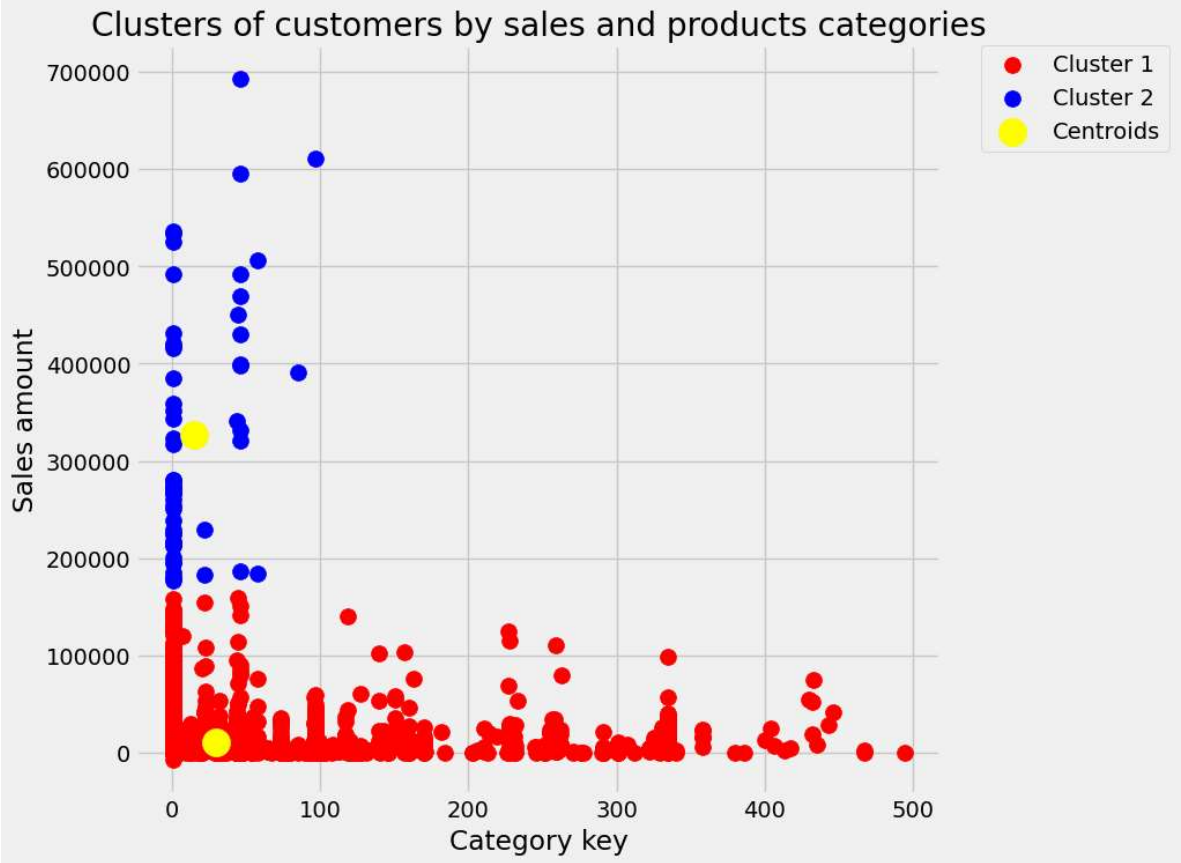
```
C:\Users\DELL\AppData\Local\Temp\ipykernel_16000\3266444764.py:20: FutureWarning:
Passing a set as an indexer is deprecated and will raise in a future version. Use
a list instead.
  data_Kmeans = data1_Kmeans[{'Sales amount','Category key'}]
```

In [6]:
```
import warnings
warnings.filterwarnings("ignore")

# Find the number of clusters by The elbow method (Min WCSS)
x = data_Kmeans.values
from sklearn.cluster import KMeans
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters= i, init='k-means++', random_state = None)
    kmeans.fit(x)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



In [7]:
```
# Find the number of clusters by The silhouette coefficient

silhouette_coefficients = []

for k in range(2, 11):
    kmeans = KMeans(n_clusters= k, init='k-means++', random_state = None)
    kmeans.fit(x)
    score = silhouette_score(x, kmeans.labels_)
    silhouette_coefficients.append(score)

plt.style.use("fivethirtyeight")
plt.plot(range(2, 11), silhouette_coefficients)
plt.xticks(range(2, 11))
plt.xlabel("Number of Clusters")
plt.ylabel("Silhouette Coefficient")
plt.show()
```

In [8]:
```
# Train the K-Means model on the dataset with 2 clusters
kmeans = KMeans(n_clusters=2 , init ='k-means++', random_state=None)
y_kmeans = kmeans.fit_predict(x)

# Visualize the clusters
plt.style.use("fivethirtyeight")
plt.figure(figsize=(8, 8))
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(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 30
plt.title('Clusters of customers by sales and products categories')
plt.ylabel('Sales amount')
plt.xlabel('Category key')
plt.legend()
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.0)
plt.show()
```



In [9]:
```
# Final locations of the centroid

print('Final locations of the centroid')
print(kmeans.cluster_centers_)
print('\n')
# The number of iterations required to converge
print('The number of iterations required to converge')
print(kmeans.n_iter_)
```

```
Final locations of the centroid
[[2.99368021e+01 1.11059416e+04]
 [1.55500000e+01 3.26727054e+05]]


The number of iterations required to converge
4
```

K-means clustering with analyse N°2

In the same way, analysis number 2 will be carried out on the basis of two axes of analysis which are: sales and affected markets.

We will create a table with customer codes, sales and market codes. Then we carry out an aggregation by customer code to have the total sales made by unique customers and also the market which includes this transaction.

The rest of the steps are the same as before but the results will be different.

Graphiquement the elbow method et the silhouette coefficient indiquent nous avons deux classes de clients.

Le résultat de la classification en deux lectures est le suivant :

Centroids lecture : We have two customer clusters:

- The first one, which spent 11105.94 euros and related to the 249 market key, which is ICF NOVEDIS.
- The second one, which spent 326727.05 euros and related to the Ramsay2 (générale de santé) , corresponds to 226 market keys.

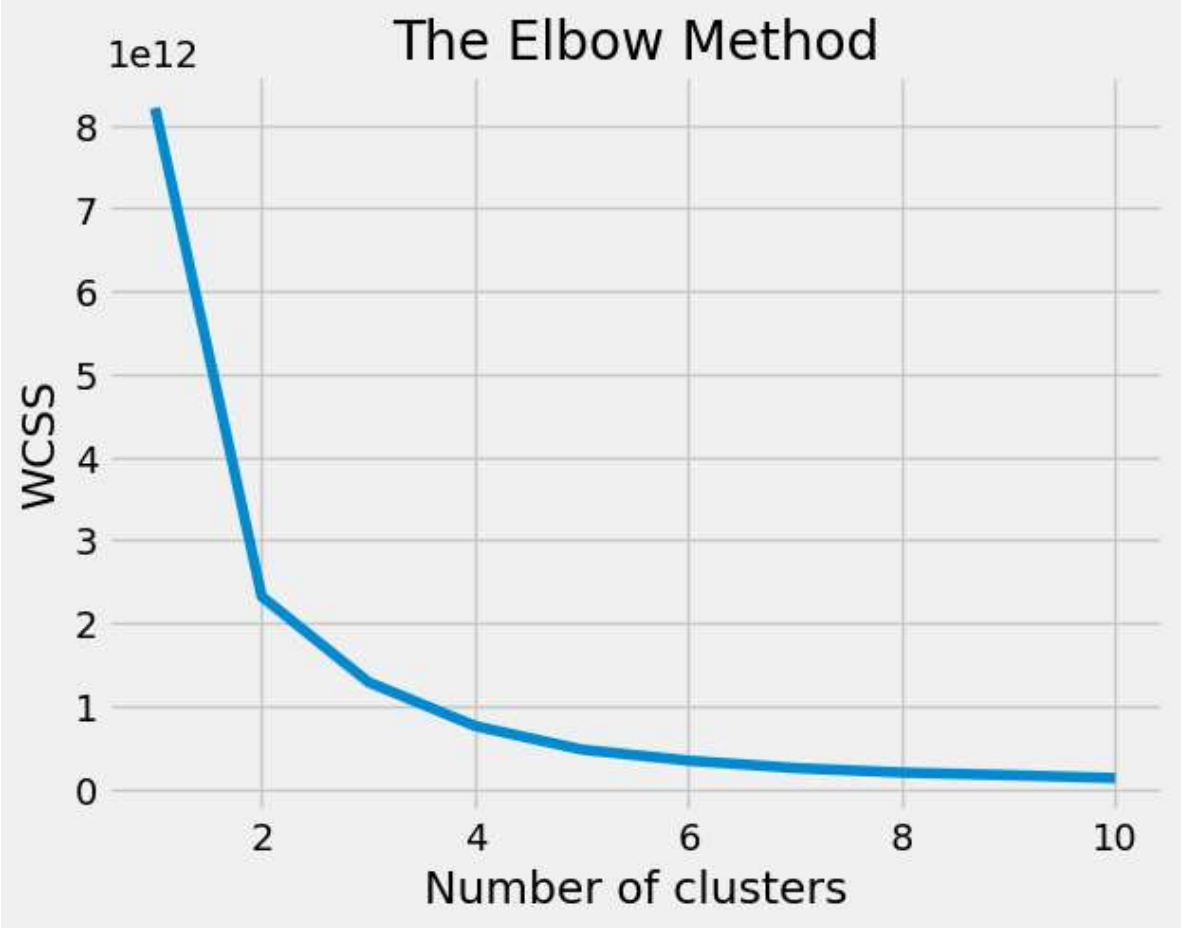
General lecture :
The first cluster can be identified by 4 sub-categories :
1 - Expenses less than 100000 euros and related to the market from 0 to 133.
2 - Expenses less than 150000 euros and related to market key 180 and 181, which are GSF and PENAUILLE markets.
3 - Expenses less than 100000 euros and related to markets keys between 200 and 250.
4 - Expenses less than 180000 euros and related to markets keys greater than 250.

The second cluster can be identified by 2 categories :
1 - Expenses between 150000 and 625000 euros and related to GSF and PENAUILLE markets keys (180 and 181).
2 - Expenses between 180000 and 700000 euros and related to RESIDENCE NATURE market key 262.

In [10]:
```
# Create dataset with Customers keys , Sales amount and Market keys
data_customers_a2 = dataset[{'Customer code','Customer name','Sales amount','Mar
                             'Market name'}]

data1_Kmeans_a2 = data_customers_a2[{'Customer code','Sales amount','Market code

# Group data by customers keys
data1_Kmeans_a2 = data1_Kmeans_a2.groupby('Customer code', as_index=False, sort=

# Transform data : Sales amount and market key into float and int

data1_Kmeans_a2['Sales amount'] = data1_Kmeans_a2['Sales amount'].astype(float)
data1_Kmeans_a2['Market code'] = data1_Kmeans_a2['Market code'].astype(int)
data_Kmeans_a2 = data1_Kmeans_a2[{'Sales amount','Market code'}]

print(data_Kmeans_a2)
```

```
      Market code  Sales amount
0             245      29757.33
1             207      72455.27
2             257      92792.05
3             106     226418.13
4             257      15253.52
...           ...           ...
4264          207        732.38
4265          106        175.67
4266          206        187.56
4267          180          0.00
4268          306       1196.80

[4269 rows x 2 columns]
```

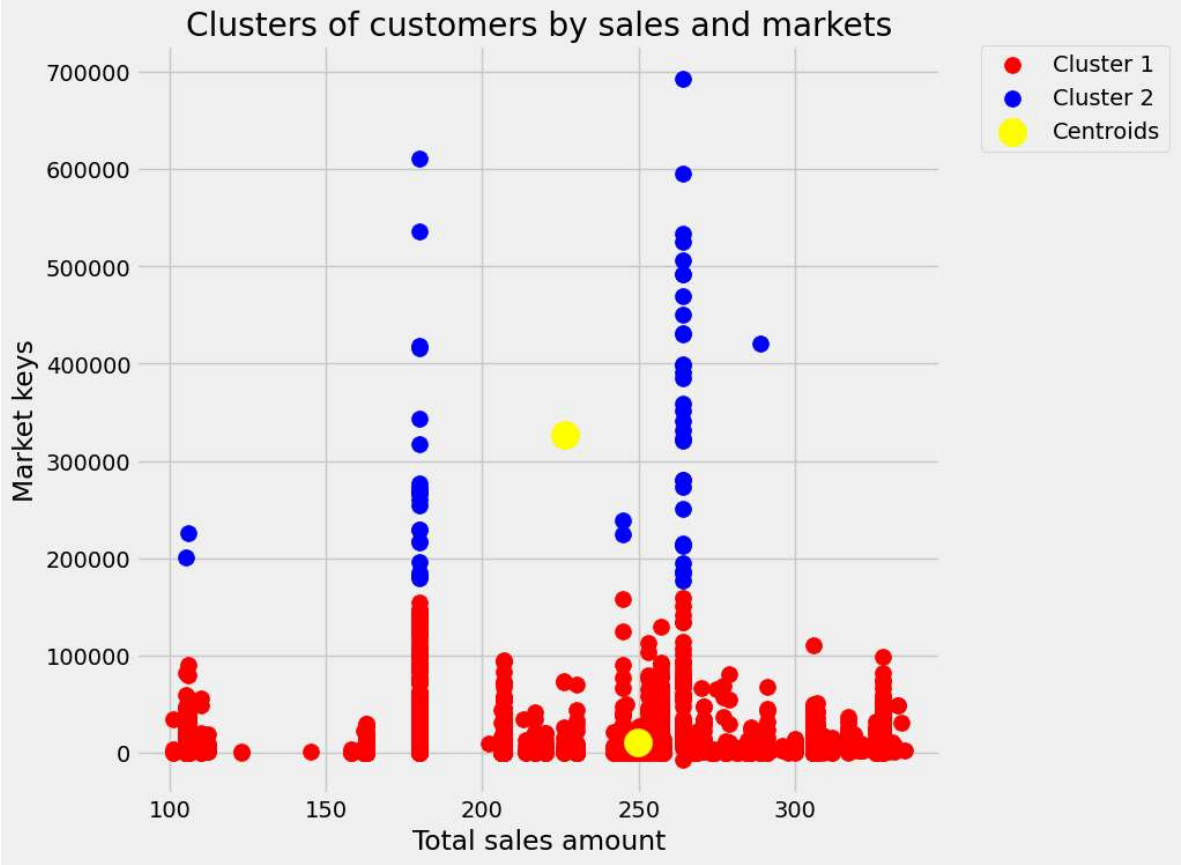
In [11]:
```
# Find the number of clusters by The elbow method (Min WCSS)

x_a2 = data_Kmeans_a2.values
from sklearn.cluster import KMeans
wcss_a2 = []
for i in range(1, 11):
    kmeans_a2 = KMeans(n_clusters= i, init='k-means++', random_state = None)
    kmeans_a2.fit(x_a2)
    wcss_a2.append(kmeans_a2.inertia_)
plt.plot(range(1, 11), wcss_a2)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



In [12]:
```
# Find the number of clusters by The silhouette coefficient
from sklearn.cluster import KMeans

silhouette_coefficients_a2 = []

for k in range(2, 11):
    kmeans_a2 = KMeans(n_clusters= k, init='k-means++', random_state = None)
    kmeans_a2.fit(x_a2)
    score_a2 = silhouette_score(x_a2, kmeans_a2.labels_)
    silhouette_coefficients_a2.append(score_a2)

plt.style.use("fivethirtyeight")
plt.plot(range(2, 11), silhouette_coefficients_a2)
plt.xticks(range(2, 11))
plt.xlabel("Number of Clusters")
```

```
plt.ylabel("Silhouette Coefficient")
plt.show()
```

In [13]:
```
# train the K-Means model on the dataset
kmeans_a2 = KMeans(n_clusters=2 , init ='k-means++', random_state=None)
y_kmeans_a2 = kmeans_a2.fit_predict(x_a2)

# Visualize the clusters

plt.style.use("fivethirtyeight")
plt.figure(figsize=(8, 8))
plt.scatter(x_a2[y_kmeans_a2 == 0, 0], x_a2[y_kmeans_a2 == 0, 1], s = 100, c = '
plt.scatter(x_a2[y_kmeans_a2 == 1, 0], x_a2[y_kmeans_a2 == 1, 1], s = 100, c = '
plt.scatter(kmeans_a2.cluster_centers_[:, 0], kmeans_a2.cluster_centers_[:, 1],
plt.title('Clusters of customers by sales and markets')
plt.ylabel('Market keys')
plt.xlabel('Total sales amount')
plt.legend()
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.0)
plt.show()
```



In [ ]:
```
# Final locations of the centroid
print('Final locations of the centroid')
print(kmeans_a2.cluster_centers_)
print('\n')
# The number of iterations required to converge
print('The number of iterations required to converge')
print(kmeans_a2.n_iter_)
```

K-means clustering with analyse N°3

Analysis number 3 will be based on the purchasing side. For this reason this time we consider the volume of purchases made by delivery members from suppliers for each customer.

The data structure is as follows:

- Customers in code and name
- Total purchases in euros
- Supplier in code with names

Before applying the model, we aggregate the data by customer code: That is to say we will have the total purchases by suppliers and by unique customers. Following the graphical analysis, the optimal number of clusters is 2.

There are two customer categories:

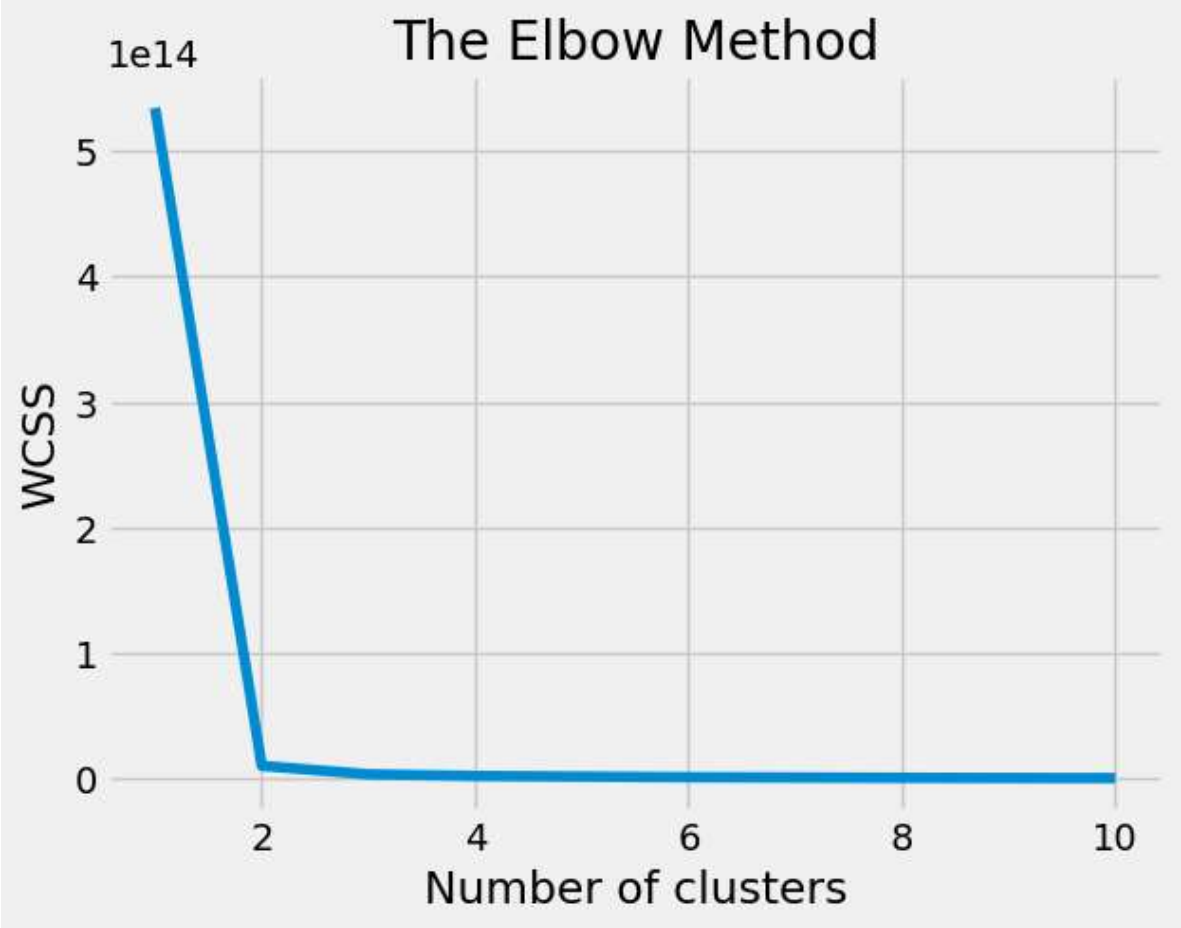
- The first one: which generally has purchases equal to 10000 euros from suppliers having key numbers 0 and 1. Those keys include 56 non-coded suppliers.

- The second one, which spent 900,000 euros on purchasing and includes probably all suppliers.

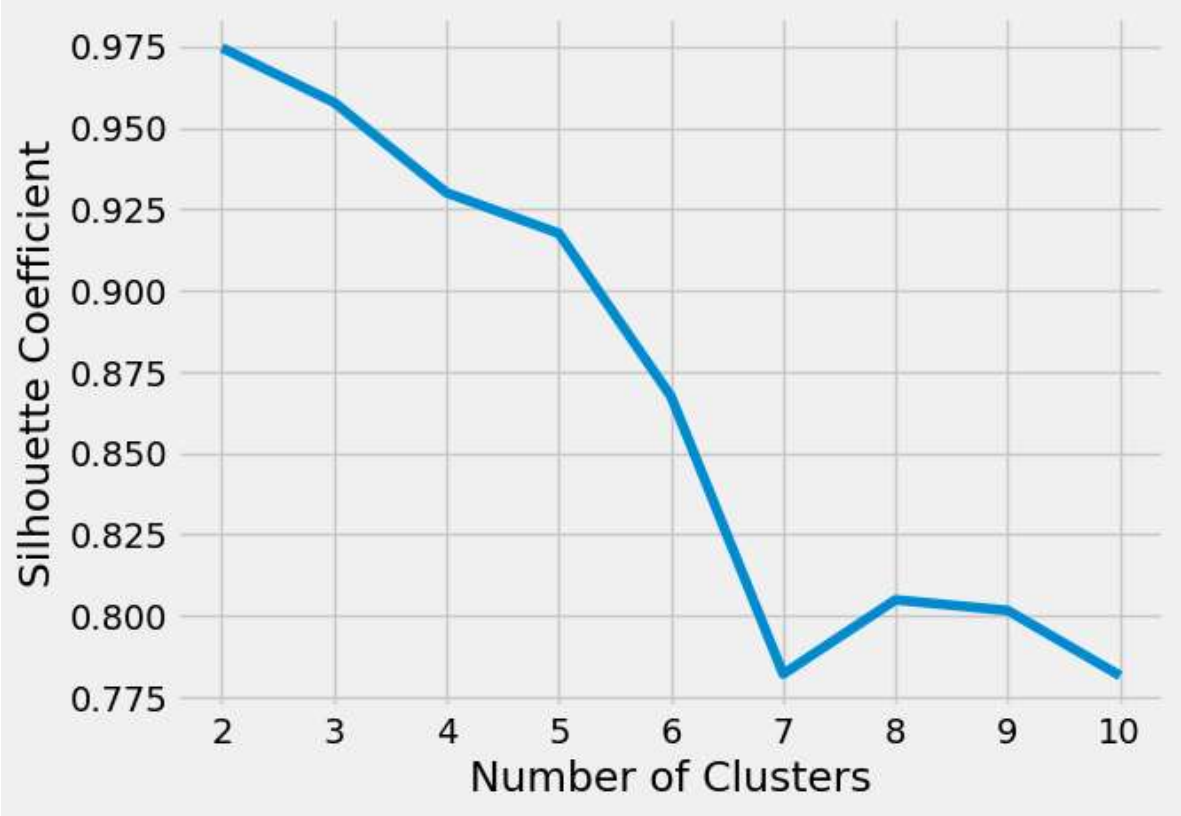
In [14]:
```
# Create dataset with : Customer code , purchasing amount and supplier

data_customers_a3 = dataset[{'Customer code','Customer name','Purchasing amount'
data1_Kmeans_a3 = data_customers_a3[{'Customer code','Purchasing amount','Suppli

# Group date by customer key
data1_Kmeans_a3 = data1_Kmeans_a3.groupby('Customer code', as_index=False, sort=

# Transform data : Purchasing amount and supplier key into float and int

data1_Kmeans_a3['Purchasing amount'] = data1_Kmeans_a3['Purchasing amount'].asty
data1_Kmeans_a3['Supplier key'] = data1_Kmeans_a3['Supplier key'].astype(int)
data_Kmeans_a3 = data1_Kmeans_a3[{'Purchasing amount','Supplier key'}]

print(data_Kmeans_a3)
```

```
      Supplier key  Purchasing amount
0                0        18578.60100
1                1        27671.01000
2                1        39779.34000
3                1       175562.01000
4                1        10188.29064
...            ...                ...
4264        999999            0.00000
4265        999999          140.58000
4266        999999            0.00000
4267        999999            0.00000
4268        999999          544.00000

[4269 rows x 2 columns]
```

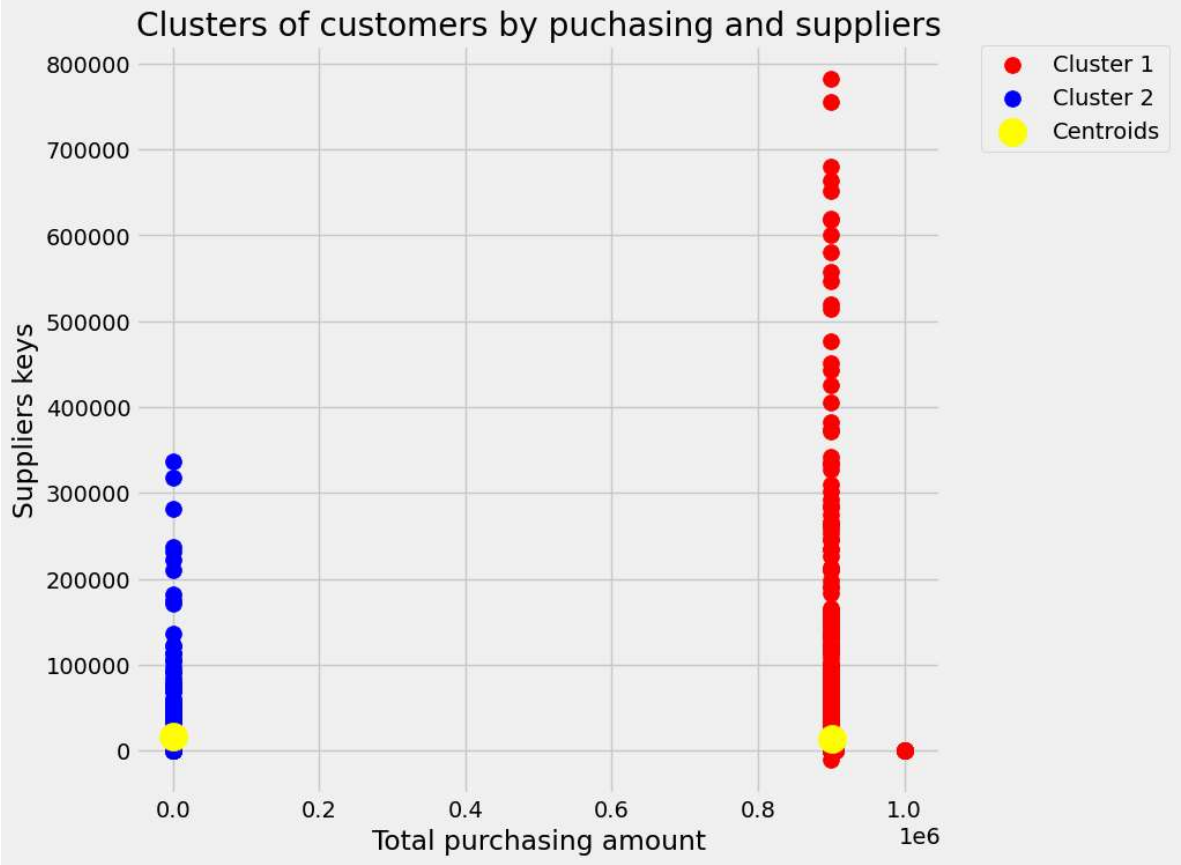
In [15]:
```
# Find the number of clusters by The elbow method

x_a3 = data_Kmeans_a3.values
from sklearn.cluster import KMeans
wcss_a3 = []
for i in range(1, 11):
    kmeans_a3 = KMeans(n_clusters= i, init='k-means++', random_state = None)
    kmeans_a3.fit(x_a3)
    wcss_a3.append(kmeans_a3.inertia_)
plt.plot(range(1, 11), wcss_a3)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



In [16]:
```
# Find the number of clusters by The silhouette coefficient

silhouette_coefficients_a3 = []
for k in range(2, 11):
    kmeans_a3 = KMeans(n_clusters= k, init='k-means++', random_state = None)
    kmeans_a3.fit(x_a3)
    score_a3 = silhouette_score(x_a3, kmeans_a3.labels_)
    silhouette_coefficients_a3.append(score_a3)

plt.style.use("fivethirtyeight")
plt.plot(range(2, 11), silhouette_coefficients_a3)
plt.xticks(range(2, 11))
plt.xlabel("Number of Clusters")
plt.ylabel("Silhouette Coefficient")
plt.show()
```



In [17]:
```
# train the K-Means model on the dataset
kmeans_a3 = KMeans(n_clusters=2 , init ='k-means++', random_state=None)
y_kmeans_a3 = kmeans_a3.fit_predict(x_a3)

# Visualize the clusters

plt.style.use("fivethirtyeight")
plt.figure(figsize=(8, 8))
plt.scatter(x_a3[y_kmeans_a3 == 0, 0], x_a3[y_kmeans_a3 == 0, 1], s = 100, c = '
plt.scatter(x_a3[y_kmeans_a3 == 1, 0], x_a3[y_kmeans_a3 == 1, 1], s = 100, c = '
plt.scatter(kmeans_a3.cluster_centers_[:, 0], kmeans_a3.cluster_centers_[:, 1],
plt.title('Clusters of customers by puchasing and suppliers')
plt.ylabel('Suppliers keys')
plt.xlabel('Total purchasing amount')
plt.legend()
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.0)
plt.show()
```



In [18]:
```
# Final locations of the centroid
print('Final locations of the centroid')
print(kmeans_a3.cluster_centers_)
print('\n')

# The number of iterations required to converge
print('The number of iterations required to converge')
print(kmeans_a3.n_iter_)
```

```
Final locations of the centroid
[[9.00835566e+05 1.40786832e+04]
 [9.98738974e-01 1.64288896e+04]]


The number of iterations required to converge
2
```

K-means clustering with analyse N°4

In part number 4 of our analysis, we will not be able to use product codes because it is a difficult task given the large number of articles in our database and the difficulty of coding them. For this reason we will only consider purchases with members. This choice is logical given that members purchase products from the parent company, then sell them to customers.

Graphically, the optimal number of clusters is two. Reading the location of the centroids indicates that the center of the first cluster is identified by a purchase amount of 9904.82 euros and delivery by 'HYGIENAZUR' (Key 6) and 'HYCODIS' (Key 7).

The second cluster of customers brings together expenses of around 374,384.99 euros, and delivery by 'DESLANDES' and 'DIFCO'.

Overall, a class of customers which was delivered by members who purchased less than 200,000 euros and a second class where members spent between 200,000 and 800,000 euros. The members in question are: 'API', 'Toussaint 57', 'Toussaint 59', 'BARTHOLUS', 'FCH', 'SODIPEC', 'SODIPREN', 'DESLANDES', 'Toussaint 67', 'TLD PRO', 'DIFCO', 'API' and 'Toussaint Dijon'.

We do not consider the rest whose code is between 7 and 10 which are 'HYGIENAZUR', 'SERIMCO', 'SEPRODO' and 'Plateforme because their purchasing volume is not considerable.

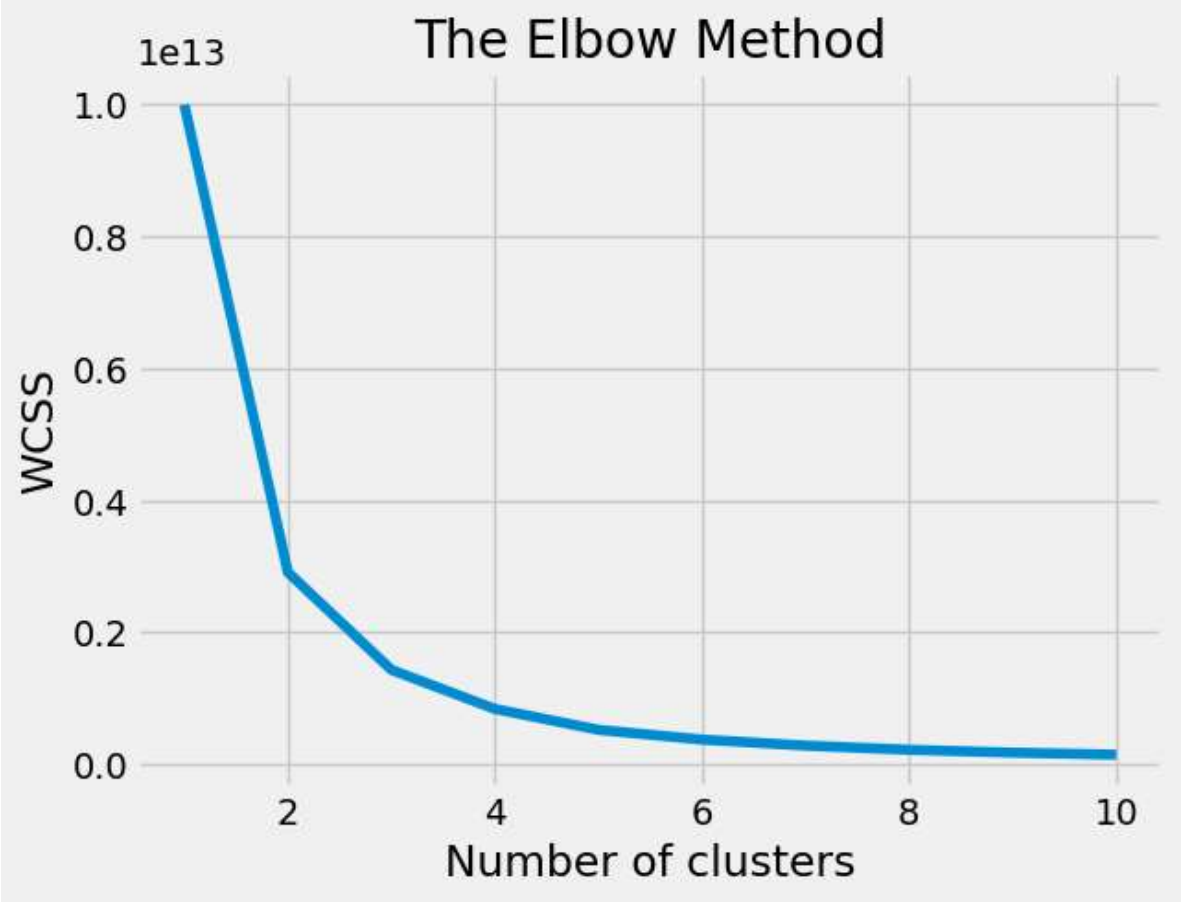
In [19]:
```
# Create dataset with : Customer code , purchasing amount and delevery members

data_customers_a4 = dataset[{'Customer code','Customer name','Purchasing amount'
data1_Kmeans_a4 = data_customers_a4[{'Customer code','Purchasing amount','Entity

# Group data by customer key
data1_Kmeans_a4 = data1_Kmeans_a4.groupby('Customer code', as_index=False, sort=
print(data1_Kmeans_a4)
```

```
     Customer code  Purchasing amount  Entity key
0           515634        18578.60100           1
1            13208        27671.01000          14
2            33624        39779.34000          15
3            30062       175562.01000           2
4             8000        10188.29064           5
...            ...                ...         ...
4264         37838            0.00000          15
4265          2808          140.58000          15
4266         20729            0.00000          15
4267        525375            0.00000          16
4268        517737          544.00000           2

[4269 rows x 3 columns]
```

In [20]:
```
# Transform data : Purchasing amount and members key into float and int

data1_Kmeans_a4['Purchasing amount'] = data1_Kmeans_a4['Purchasing amount'].asty
data1_Kmeans_a4['Entity key'] = data1_Kmeans_a4['Entity key'].astype(int)
data_Kmeans_a4 = data1_Kmeans_a4[{'Purchasing amount','Entity key'}]
print(data_Kmeans_a4)
```

```
      Entity key  Purchasing amount
0              1        18578.60100
1             14        27671.01000
2             15        39779.34000
3              2       175562.01000
4              5        10188.29064
...          ...                ...
4264          15            0.00000
4265          15          140.58000
4266          15            0.00000
4267          16            0.00000
4268           2          544.00000

[4269 rows x 2 columns]
```

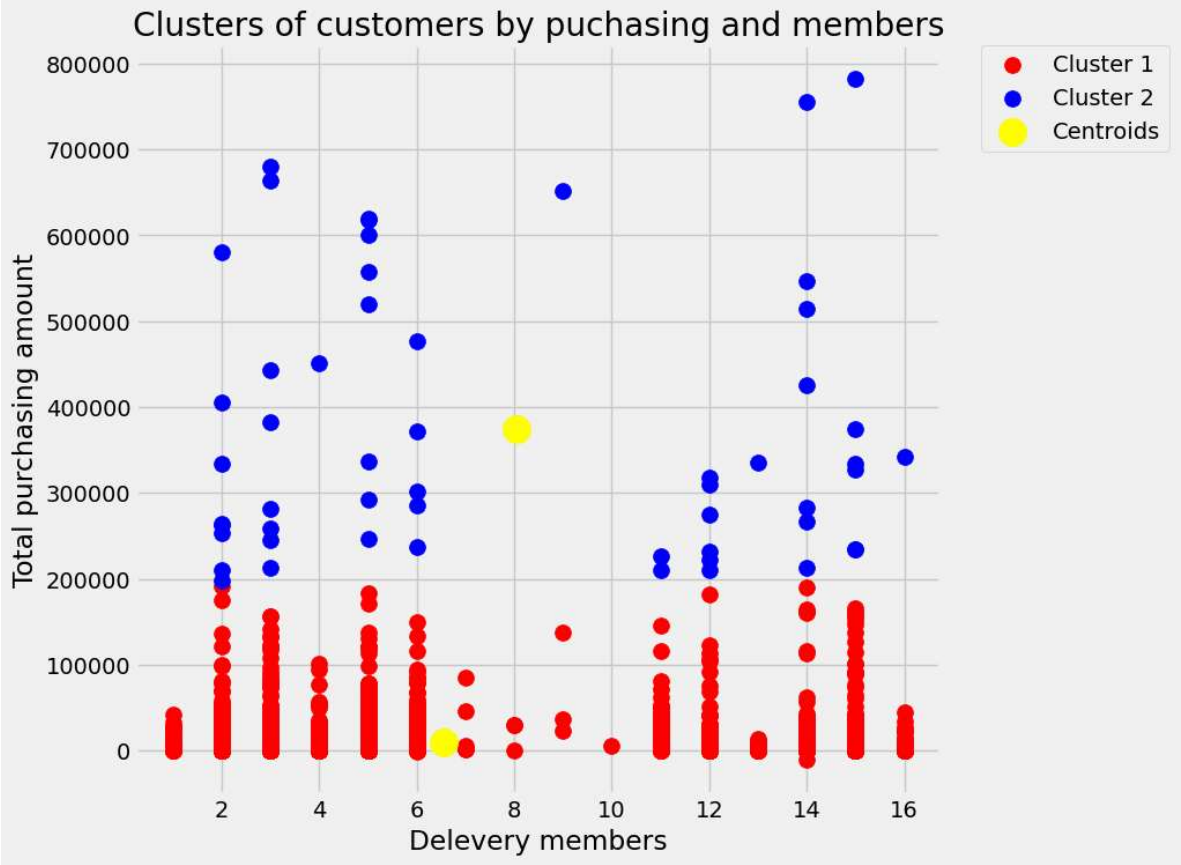
In [21]:
```
# Find the number of clusters by The elbow method (Min WCSS)

x_a4 = data_Kmeans_a4.values
from sklearn.cluster import KMeans
wcss_a4 = []
for i in range(1, 11):
    kmeans_a4 = KMeans(n_clusters= i, init='k-means++', random_state = None)
    kmeans_a4.fit(x_a4)
    wcss_a4.append(kmeans_a4.inertia_)
plt.plot(range(1, 11), wcss_a4)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



In [22]:
```
# Find the number of clusters by The silhouette coefficient

silhouette_coefficients_a4 = []
```

```
for k in range(2, 11):
    kmeans_a4 = KMeans(n_clusters= k, init='k-means++', random_state = None)
    kmeans_a4.fit(x_a4)
    score_a4 = silhouette_score(x_a4, kmeans_a4.labels_)
    silhouette_coefficients_a4.append(score_a4)

plt.style.use("fivethirtyeight")
plt.plot(range(2, 11), silhouette_coefficients_a4)
plt.xticks(range(2, 11))
plt.xlabel("Number of Clusters")
plt.ylabel("Silhouette Coefficient")
plt.show()
```

In [24]:
```
# train the K-Means model on the dataset
kmeans_a4 = KMeans(n_clusters=2 , init ='k-means++', random_state=None)
y_kmeans_a4 = kmeans_a4.fit_predict(x_a4)

# Visualize the clusters

plt.style.use("fivethirtyeight")
plt.figure(figsize=(8, 8))
plt.scatter(x_a4[y_kmeans_a4 == 0, 0], x_a4[y_kmeans_a4 == 0, 1], s = 100, c = '
plt.scatter(x_a4[y_kmeans_a4 == 1, 0], x_a4[y_kmeans_a4 == 1, 1], s = 100, c = '
plt.scatter(kmeans_a4.cluster_centers_[:, 0], kmeans_a4.cluster_centers_[:, 1],
plt.title('Clusters of customers by puchasing and members')
plt.xlabel('Delevery members')
plt.ylabel('Total purchasing amount')
plt.legend()
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.0)
plt.show()
```



In [26]:
```
# Final locations of the centroid
print('Final locations of the centroid')
print(kmeans_a4.cluster_centers_)
print('\n')

# The number of iterations required to converge
print('The number of iterations required to converge')
print(kmeans_a4.n_iter_)
```

```
Final locations of the centroid
[[6.54567023e+00 9.90482145e+03]
 [8.03703704e+00 3.74384997e+05]]


The number of iterations required to converge
2
```