Import data

In [2]:

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn import preprocessing
import seaborn as sns
import scipy.cluster.hierarchy as sch
from sklearn.cluster import AgglomerativeClustering
```

Problem

Perform clustering (hierarchical,K means clustering and DBSCAN) for the airlines data to obtain optimum number of clusters. Draw the inferences from the clusters obtained.

Import data

In [3]:

```
airlines_data = pd.read_csv('EastWestAirlines.csv')
airlines_data
```

Out[3]:

	ID#	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus_trans	Flight_miles_12mo	Flight_trans_12	Days_since_enroll	Awar
0	1	28143	0	1	1	1	174	1	0	0	7000	
1	2	19244	0	1	1	1	215	2	0	0	6968	
2	3	41354	0	1	1	1	4123	4	0	0	7034	
3	4	14776	0	1	1	1	500	1	0	0	6952	
4	5	97752	0	4	1	1	43300	26	2077	4	6935	
...	...	...	...	...	...	...	...	...	...	...	...	
3994	4017	18476	0	1	1	1	8525	4	200	1	1403	

	ID#	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus_trans	Flight_miles_12mo	Flight_trans_12	Days_since_enroll	Awar
3995	4018	64385	0	1	1	1	981	5	0	0	1395	
3996	4019	73597	0	3	1	1	25447	8	0	0	1402	
3997	4020	54899	0	1	1	1	500	1	500	1	1401	
3998	4021	3016	0	1	1	1	0	0	0	0	1398	

Data understanding

In [4]:

```
airlines_data.shape
```

Out[4]:

```
(3999, 12)
```

In [5]:

```
airlines_data.isna().sum()
```

Out[5]:

```
ID#                  0
Balance              0
Qual_miles           0
cc1_miles            0
cc2_miles            0
cc3_miles            0
Bonus_miles          0
Bonus_trans          0
Flight_miles_12mo    0
Flight_trans_12      0
Days_since_enroll    0
Award?               0
dtype: int64
```

In [6]:

```
airlines_data.dtypes
```

Out[6]:

```
ID#                  int64
Balance              int64
Qual_miles           int64
cc1_miles            int64
cc2_miles            int64
```

```
cc3_miles            int64
Bonus_miles          int64
Bonus_trans          int64
Flight_miles_12mo    int64
Flight_trans_12      int64
Days_since_enroll    int64
Award?               int64
dtype: object
```

In [7]:

```
airlines_data.describe()
```

Out[7]:

	ID#	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus_trans	Flight_miles_12mo	Flight_trans_12
count	3999.000000	3.999000e+03	3999.000000	3999.000000	3999.000000	3999.000000	3999.000000	3999.00000	3999.000000	3999.000000
mean	2014.819455	7.360133e+04	144.114529	2.059515	1.014504	1.012253	17144.846212	11.60190	460.055764	1.373593
std	1160.764358	1.007757e+05	773.663804	1.376919	0.147650	0.195241	24150.967826	9.60381	1400.209171	3.793172
min	1.000000	0.000000e+00	0.000000	1.000000	1.000000	1.000000	0.000000	0.00000	0.000000	0.000000
25%	1010.500000	1.852750e+04	0.000000	1.000000	1.000000	1.000000	1250.000000	3.00000	0.000000	0.000000
50%	2016.000000	4.309700e+04	0.000000	1.000000	1.000000	1.000000	7171.000000	12.00000	0.000000	0.000000
75%	3020.500000	9.240400e+04	0.000000	3.000000	1.000000	1.000000	23800.500000	17.00000	311.000000	1.000000
max	4021.000000	1.704838e+06	11148.000000	5.000000	3.000000	5.000000	263685.000000	86.00000	30817.000000	53.000000

Data preprocessing

In [8]:

```
#standardize the data to normal distribution
airline_norm = preprocessing.scale(airlines_data)
```

In [9]:

```
airline_norm = pd.DataFrame(airline_norm)
```

In [10]:

```
airline_norm.head()
```

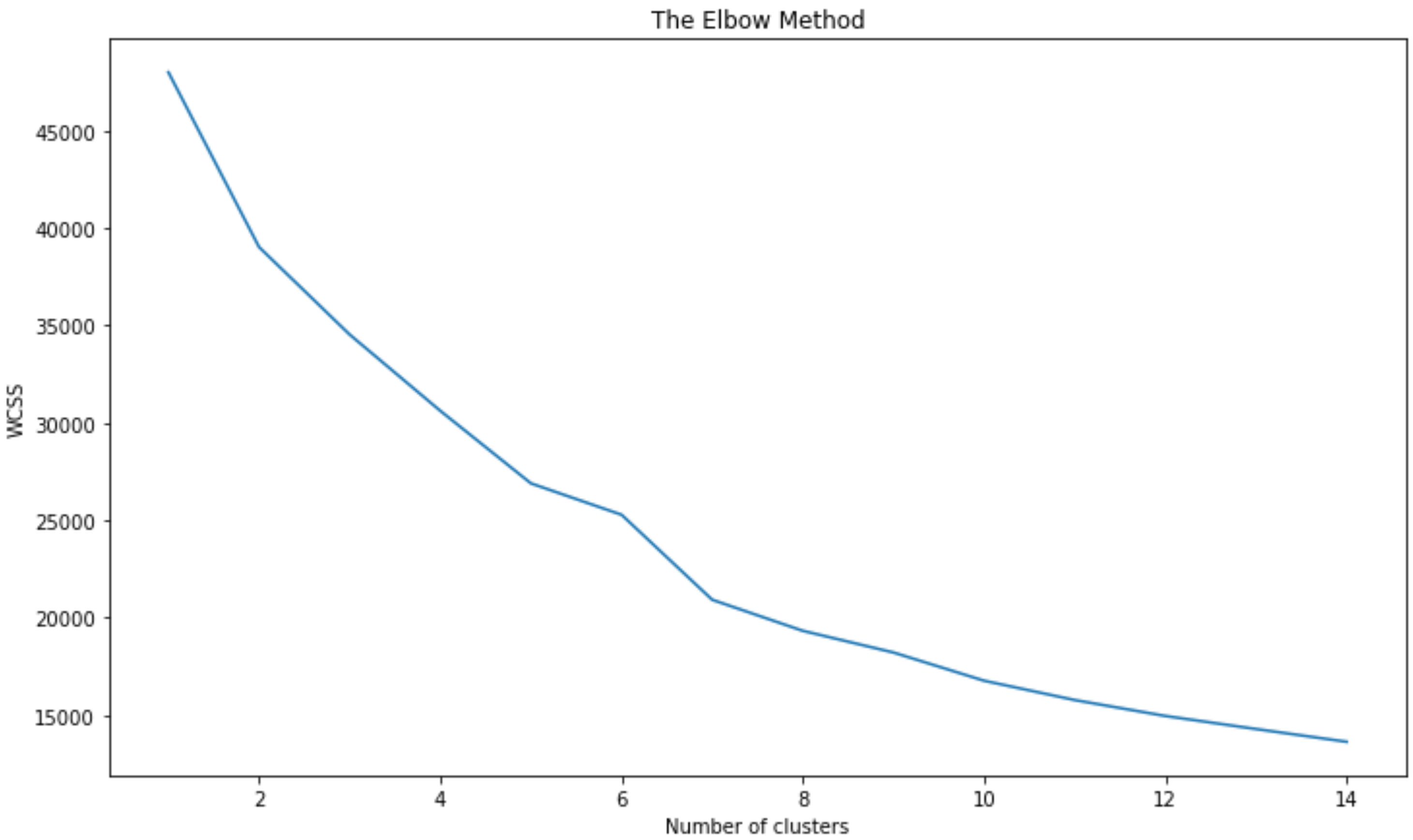
Out[10]:

	0	1	2	3	4	5	6	7	8	9	10	11
0	-1.735125	-0.451141	-0.186299	-0.769578	-0.098242	-0.062767	-0.702786	-1.104065	-0.328603	-0.362168	1.395454	-0.766919
1	-1.734263	-0.539457	-0.186299	-0.769578	-0.098242	-0.062767	-0.701088	-0.999926	-0.328603	-0.362168	1.379957	-0.766919
2	-1.733402	-0.320031	-0.186299	-0.769578	-0.098242	-0.062767	-0.539253	-0.791649	-0.328603	-0.362168	1.411920	-0.766919
3	-1.732540	-0.583799	-0.186299	-0.769578	-0.098242	-0.062767	-0.689286	-1.104065	-0.328603	-0.362168	1.372208	-0.766919
4	-1.731679	0.239678	-0.186299	1.409471	-0.098242	-0.062767	1.083121	1.499394	1.154932	0.692490	1.363975	1.303918

Finding out the optimal number of clusters

In [11]:

```
plt.figure(figsize=(12, 7))
wcss = []
for i in range(1, 15):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans.fit(airline_norm)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 15), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```

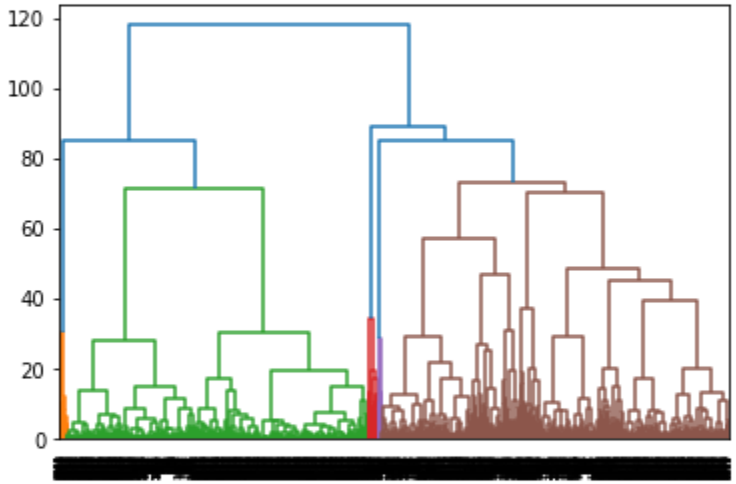


As seen from the elbow graph, the slope changes at 2. However, since spltting the dataset into 2 groups would not be very beneficial, we further evaluate clusters for higher values of k.

Hierarchical clustering

Euclidean distance & Ward

In [12]:
```
dendrogram = sch.dendrogram(sch.linkage(airline_norm, method='ward'))
```



From the Ward method, we see that as the height increases the clusters get grouped together

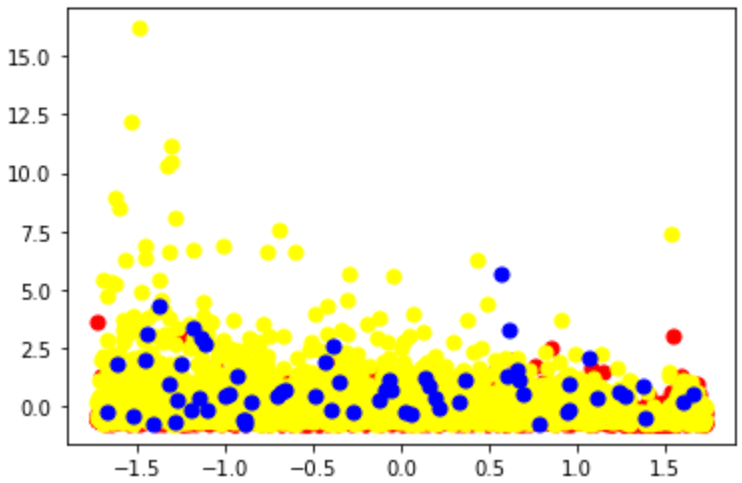
We decided to cut the tree at height 85 to obtain 3 clusters and then assigned each cluster with its respective observations

In [13]:
```
X = airline_norm.values
```

In [14]:
```
model = AgglomerativeClustering(n_clusters=3, affinity='euclidean', linkage='ward')
h_cluster = model.fit(X)
```

In [15]:
```
labels = model.labels_
```

In [16]:
```
plt.scatter(X[labels==0, 0], X[labels==0, 1], s=50, marker='o', color='red')
plt.scatter(X[labels==1, 0], X[labels==1, 1], s=50, marker='o', color='yellow')
plt.scatter(X[labels==2, 0], X[labels==2, 1], s=50, marker='o', color='blue')
plt.show()
```



K - means

k-means clustering is a method of vector quantization, originally from signal processing, that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean (cluster centers or cluster centroid), serving as a prototype of the cluster.

In [17]:
```
kmeans = KMeans(n_clusters = 3, init = 'k-means++', random_state = 42)
k_mean = kmeans.fit_predict(airline_norm)
```

In [18]:
```
k_mean
```

Out[18]:
```
array([0, 0, 0, ..., 0, 0, 0])
```

In [19]:
```
k_mean1=k_mean+1
k_cluster = list(k_mean1)
```

In [20]:
```
airlines_data['k_cluster'] = k_cluster
```

In [21]:

```
kmeans_cluster = pd.DataFrame(round(airlines_data.groupby('k_cluster').mean(),1))
kmeans_cluster
```

Out[21]:

k_cluster	ID#	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus_trans	Flight_miles_12mo	Flight_trans_12	Days_since_enroll
1	2327.1	42243.6	91.1	1.2	1.0	1.0	4896.4	7.0	194.4	0.6	3549.8
2	1445.6	119557.7	165.6	3.6	1.0	1.0	38921.2	18.6	351.2	1.1	5147.4
3	1753.1	189304.2	788.7	2.2	1.0	1.0	31780.5	27.1	5420.4	15.8	4657.0

In [22]:

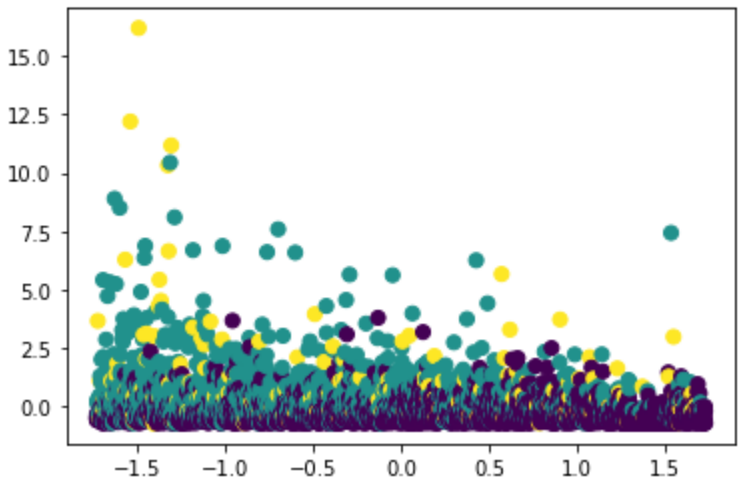
```
pd.DataFrame(round(airlines_data.groupby('k_cluster').count(),1))
```

Out[22]:

k_cluster	ID#	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus_trans	Flight_miles_12mo	Flight_trans_12	Days_since_enroll	A
1	2525	2525	2525	2525	2525	2525	2525	2525	2525	2525	2525	
2	1310	1310	1310	1310	1310	1310	1310	1310	1310	1310	1310	
3	164	164	164	164	164	164	164	164	164	164	164	

In [23]:

```
plt.scatter(X[:, 0], X[:, 1], c=k_mean, s=50, cmap='viridis')
plt.show()
```



From the above data generated from K-Means clustering, we can see Cluster-1 has around 63% total travelers and cluster 2 has 33% of the travelers. We will target cluster 1 & 2. Cluster 1 contains less frequent or first time travellers, by giving them discount provided they travel more than twice or thrice and introduce more offer if they register or take the membership.

In [ ]: