INTRODUCTION

Clustering involves the grouping of data points or items based on their shared similarities, which are typically quantified using distance measures that calculate the extent of separation between two data points. Two commonly used distance measures are the Euclidean and Manhattan distances.

The most prevalent types of clustering algorithms are:

K-Means Clustering: This is an exclusive clustering technique, where each data point belongs to only one cluster. With K-Means, the data points are grouped into a predetermined number, K, of clusters.

Hierarchical Clustering: This method organizes data points or items into groups in a hierarchical fashion, creating a tree-like structure of clusters. Unlike K-Means, hierarchical clustering does not require specifying the number of clusters in advance, offering more flexibility in clustering analysis.

In this analysis we will use both KMean and Hierarchical clustering to

1. Segregate the customers into clusters/group
2. provide recommendations to a retail manager based on the identified customer segments.

Analysis Scope: The analysis will focus on a 2D space for CustomerID and Spending score. This is an illustration, and similar approaches can be applied to different features.

Analysis on different features follow the same approach, however, for categorical features we need to convert to numerical values before using the machine learning algorithm.

Dataset

The customer dataset is a learning dataset from kaggle. It comprises of five features(CUstomerID, Gender, Age, Annual Income and Spending Score) with 200 rows. Spending score ranges from 0 to 100 and is assigned to a customers based on criteria such as purchase quantity and amount.

KMEAN clustering

Step by step in performing a KMean Clustering using python

1. Perform data preprocessing on the dataset
2. Extract the needed features for the analysis
3. Select the number of centroids using the elbow analysis
4. Build the model using the KMeans algorithm
5. Visualize your result of the clusters

In [28]:

```
# importing libraries
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn.cluster import KMeans


import warnings
warnings.filterwarnings('ignore')


plt.rcParams['figure.figsize'] =(16,8)
```

In [29]:

```
#loading data
customer_data = pd.read_csv("Mall_Customers.csv")
customer_data.head()
```

Out[29]:

	CustomerID	Gender	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 [30]:

```
customer_data.shape
```

Out[30]:

```
(200, 5)
```

In [31]:

```
customer_data.columns
```

Out[31]:

```
Index(['CustomerID', 'Gender', 'Age', 'Annual Income (k$)',
       'Spending Score (1-100)'],
      dtype='object')
```

Exploratory Data Analysis

In [32]:

```
customer_data.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   Gender                  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
```

In [33]:

```
customer_data.describe()
```

Out[33]:

	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)
count	200.000000	200.000000	200.000000	200.000000
mean	100.500000	38.850000	60.560000	50.200000
std	57.879185	13.969007	26.264721	25.823522
min	1.000000	18.000000	15.000000	1.000000
25%	50.750000	28.750000	41.500000	34.750000
50%	100.500000	36.000000	61.500000	50.000000
75%	150.250000	49.000000	78.000000	73.000000
max	200.000000	70.000000	137.000000	99.000000

In [34]:

```
customer_data.isnull().sum()
```

Out[34]:

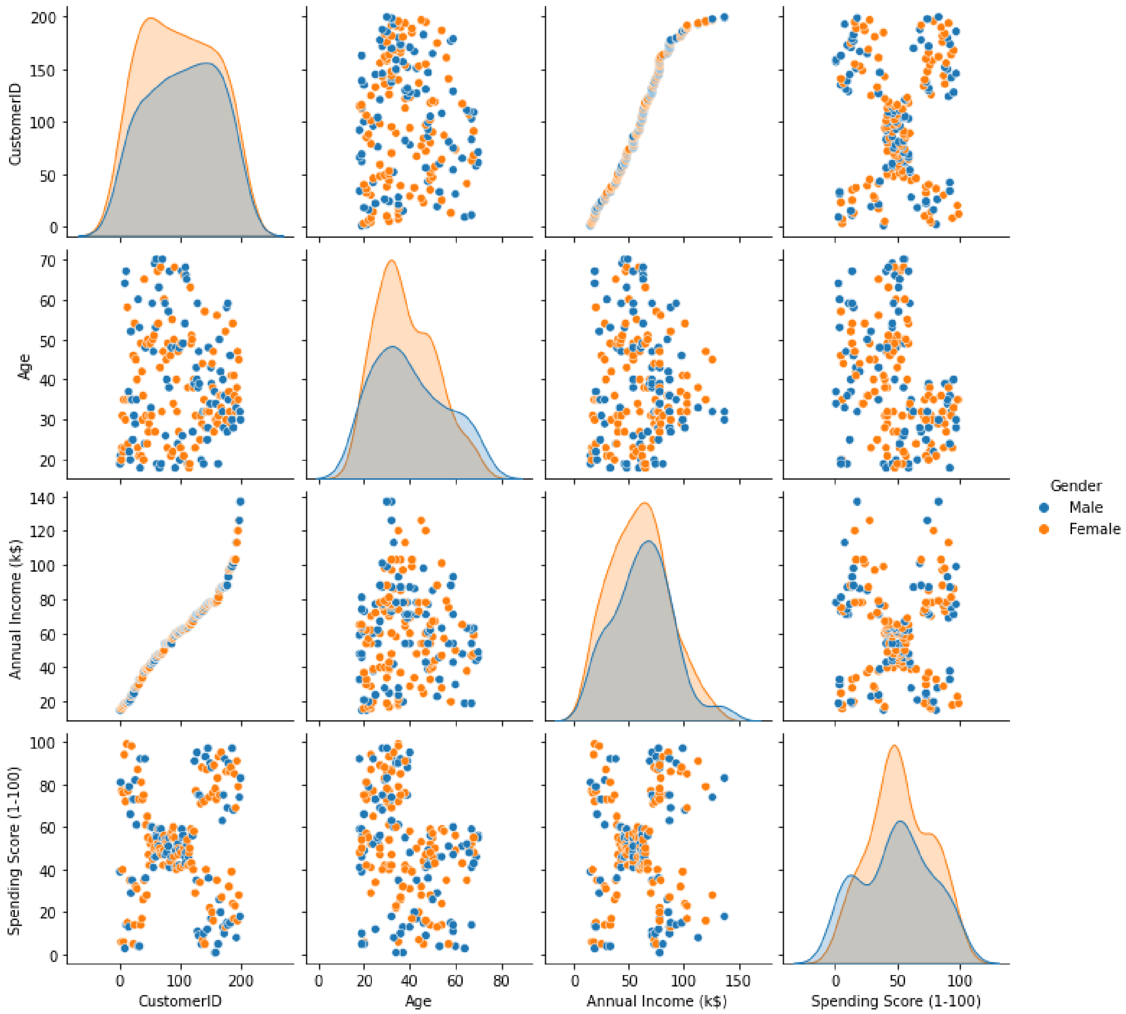
```
CustomerID                0
Gender                    0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

In [35]:

```
sns.pairplot(customer_data, hue="Gender")
```

Out[35]:

```
<seaborn.axisgrid.PairGrid at 0x1cbaa30dbe0>
```

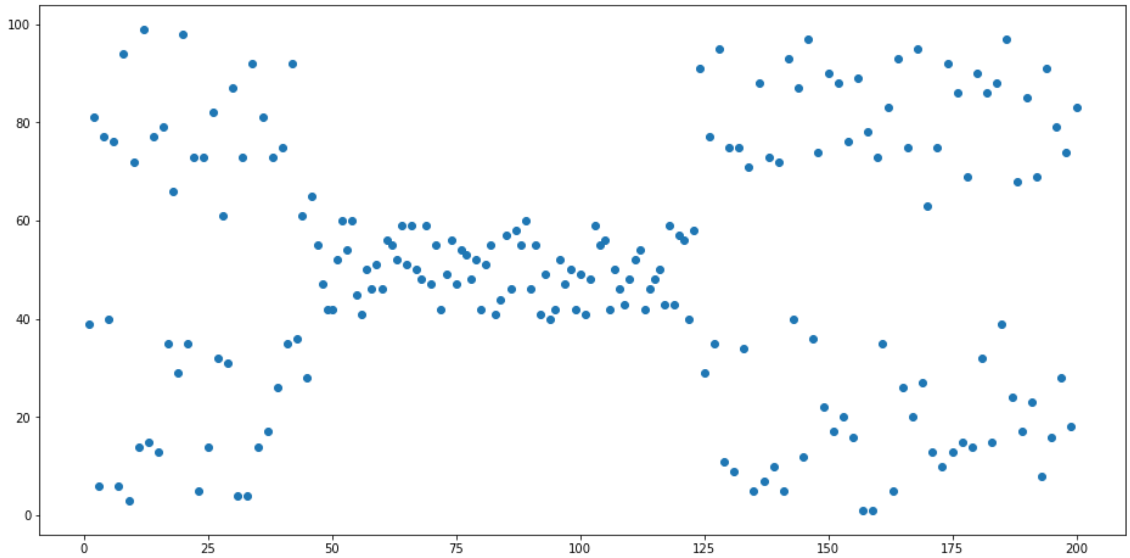


Extracting the needed features for the analysis

We segregate the customers into clusters based on CustomerIDs and Spending score.

In [36]:

```
plt.scatter(customer_data['CustomerID'], customer_data['Spending Score (1-100)']);
```



In [37]:

```
#extracting the ID and Spending Score
new_customer_data = customer_data.iloc[:, [0,4]]
new_customer_data
```

Out[37]:

	CustomerID	Spending Score (1-100)
0	1	39
1	2	81
2	3	6
3	4	77
4	5	40
...	...	...
195	196	79
196	197	28
197	198	74
198	199	18
199	200	83

200 rows × 2 columns

Computing the number of centroids

Elbow Analysis

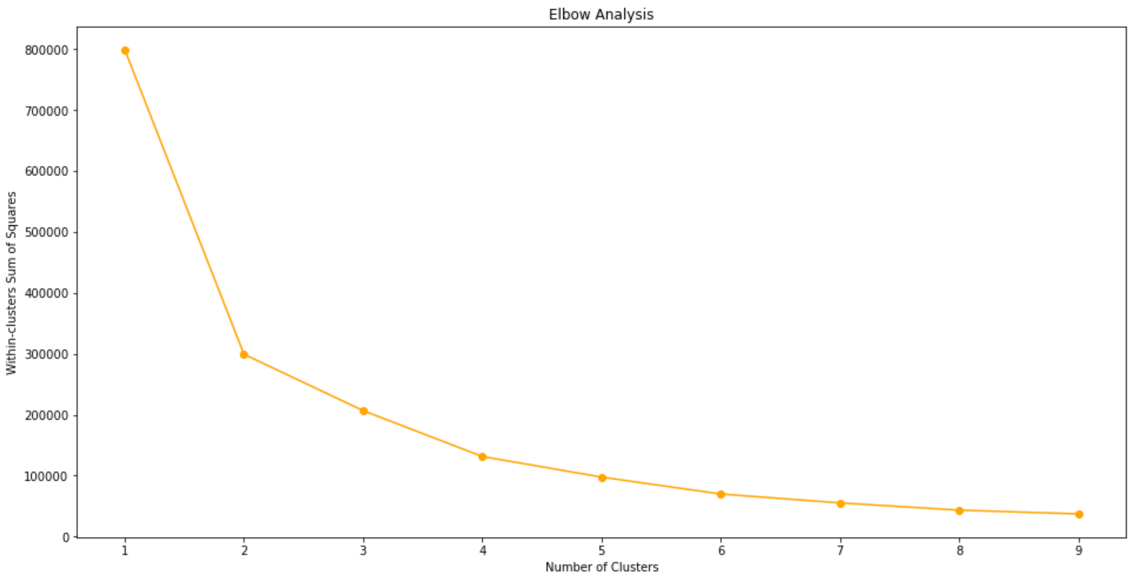
Using the elbow analysis, we can find the number of k centroids in the kmean clustering.

In [38]:

```
wcss = []
for n in range(1, 10):
    Km = KMeans(n_clusters=n, random_state=2)
    Km.fit(new_customer_data)
    wcss.append(Km.inertia_);
```

In [39]:

```
#ploting the elbow
plt.plot(np.arange(1,10), wcss, marker='o', color='orange', )
plt.title('Elbow Analysis')
plt.xlabel('Number of Clusters')
plt.ylabel('Within-clusters Sum of Squares');
```



Building the K Mean model

From the elbow analysis, the number of clusters to be used for the KMean clustering is 4.

In [40]:

```
#Fitting the Kmean
Kmeans = KMeans(n_clusters=4, random_state=2)
Kmeans.fit(new_customer_data)
```

Out[40]:

```
▾                 KMeans
KMeans(n_clusters=4, random_state=2)
```

Computing the centroids

In [41]:

```
#Computing the centroids of the dataset
centroids = Kmeans.cluster_centers_


labels = Kmeans.labels_
```

In [42]:

```
#predicting the clusters for each customer
pred = Kmeans.fit_predict(new_customer_data)
pred
```

Out[42]:

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 0, 2, 3, 2, 3, 2,
       3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2,
       3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2,
       3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2,
       3, 2])
```

Visualizing the Clusters

In [43]:

```
#converting dataset to numpy array
data_array = np.array(new_customer_data)


h = 0.02
x_min, x_max = data_array[:, 0].min() - 1, data_array[:, 0].max() + 1
y_min, y_max = data_array[:, 1].min() - 1, data_array[:, 1].max() + 1
x, y = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
Z = Kmeans.predict(np.c_[x.ravel(), y.ravel()])
```

In [44]:

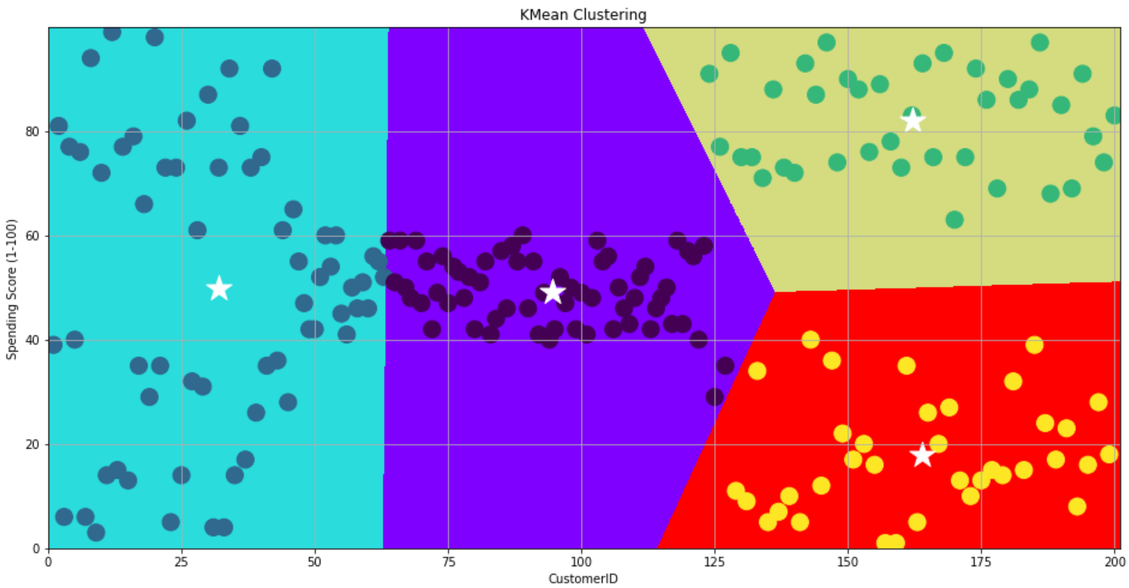
```
# Plotting the clusters

Z = Z.reshape(x.shape)
plt.imshow(Z , interpolation='nearest',
           extent=(x.min(), x.max(), y.min(), y.max()),
           cmap = 'rainbow', aspect = 'auto', origin='lower')

plt.scatter(new_customer_data['CustomerID'], new_customer_data['Spending Score (1-100)']

plt.scatter(centroids[:, 0], centroids[:, 1], s = 500, marker='*', color='white');


plt.grid()
plt.title('KMean Clustering')
plt.xlabel('CustomerID')
plt.ylabel('Spending Score (1-100)');
```



HIERARCHICAL CLUSTERING

This clustering approach aims to arrange data points or items in a hierarchical structure. There are two primary hierarchical clustering methods: Agglomerative and Divisive. Data points within the same clusters are considered similar based on a distance measure, and clusters are merged using a linkage method. Common linkage methods include the single linkage method, compound linkage method, ward linkage method, among others.

In this analysis, we have utilized the Euclidean distance measure and specifically chosen the Ward linkage method. The rationale behind selecting the Ward linkage method is to minimize the variance within clusters and achieve balanced cluster size. This method helps ensure that the resulting clusters are homogeneous and have relatively equal sizes.

AGGLOMERATIVE CLUSTERING

In agglomerative hierarchical clustering, the algorithm starts with a bottom-up approach, treating each data point as a separate cluster, and then successively merges the clusters using a linkage method until a single cluster containing all data points is formed.

Step by Step

1. Preprocess the dataset
2. Select the features needed for the analysis.
3. Construct a dendrogram to aid in selecting the number of clusters
4. Build the model.
5. Visualize the clusters

In [45]:

```
new_customer_data_hier = new_customer_data
new_customer_data_hier
```

Out[45]:

	CustomerID	Spending Score (1-100)
0	1	39
1	2	81
2	3	6
3	4	77
4	5	40
...	...	...
195	196	79
196	197	28
197	198	74
198	199	18
199	200	83

200 rows × 2 columns

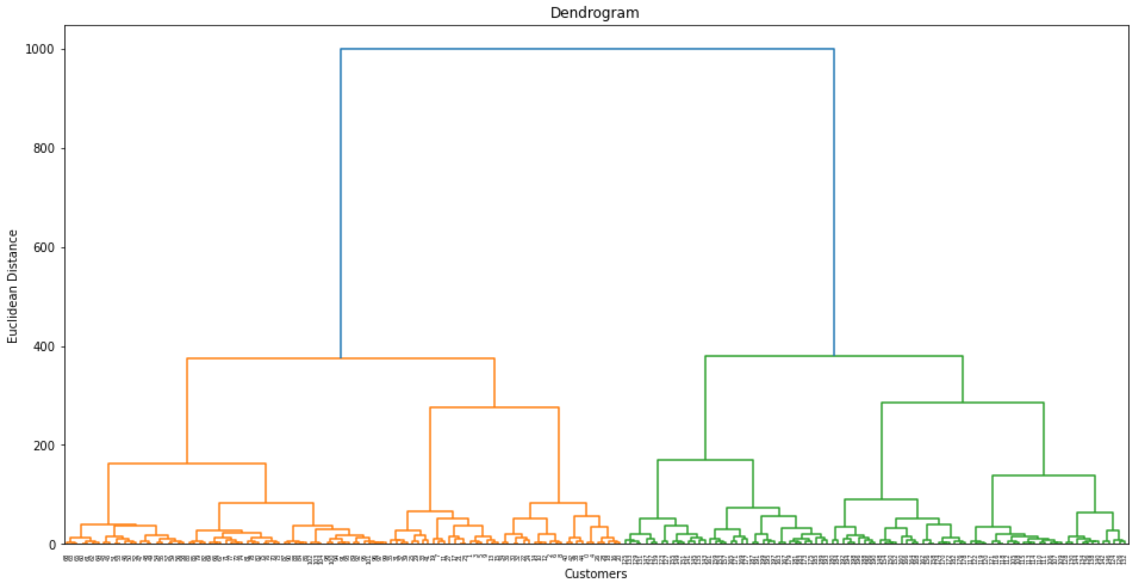
Constructing the dendrogram

In [46]:

```
from sklearn.cluster import AgglomerativeClustering
from scipy.cluster.hierarchy import dendrogram, linkage
```

In [47]:

```
dendro = dendrogram(linkage(new_customer_data_hier, method='ward'))
plt.xlabel('Customers')
plt.ylabel('Euclidean Distance')
plt.title('Dendrogram')
plt.show();
```



Building the Agglomerative model

Based on the dendrogram we consider **4** clusters to build the model

In [48]:

```
Agglom = AgglomerativeClustering(n_clusters = 4, affinity='euclidean', linkage='ward')
Agglom.fit(new_customer_data_hier)
```

Out[48]:

```
AgglomerativeClustering
AgglomerativeClustering(affinity='euclidean', n_clusters=4)
```

In [49]:

```
y_pred= Agglom.fit_predict(new_customer_data_hier)
y_pred
```

Out[49]:

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3,
       1, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2,
       0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2,
       0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2,
       0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2,
       0, 2], dtype=int64)
```

Visualizing the Clusters

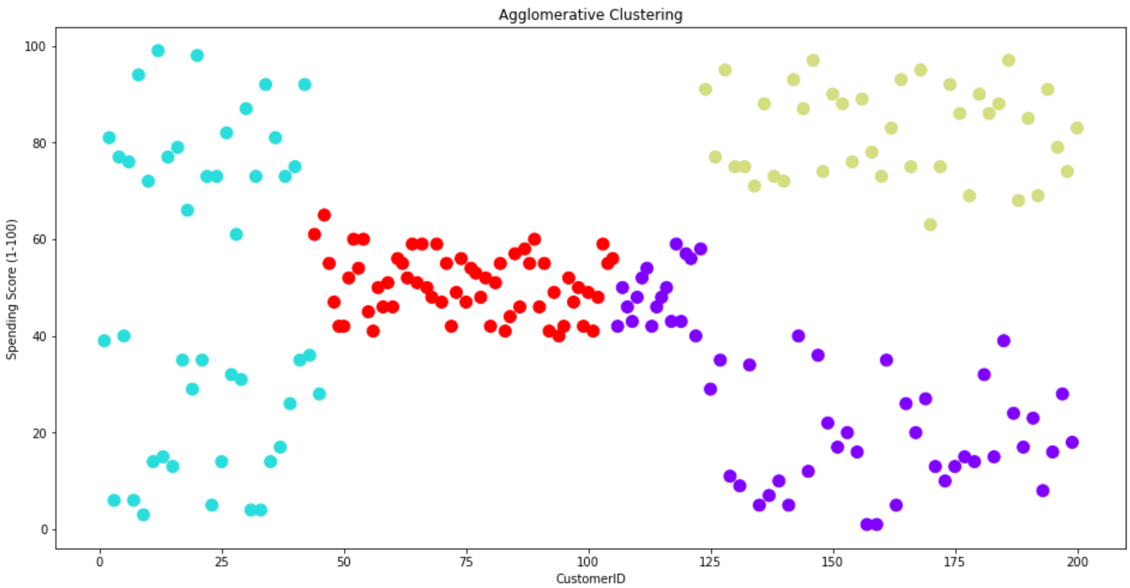
In [50]:

```
cluster_labels = Agglom.labels_



plt.scatter(new_customer_data_hier['CustomerID'], new_customer_data_hier['Spending Score
             c=cluster_labels, cmap='rainbow', s=100)

plt.title('Agglomerative Clustering')
plt.xlabel('CustomerID')
plt.ylabel('Spending Score (1-100)');
```



Extracting the customer segments

In [51]:

```
### Extracting the customer segement
new_customer_data_hier['y_pred'] = y_pred

segment_1 = new_customer_data_hier[new_customer_data_hier['y_pred']==0]
segment_1.head()
```

Out[51]:

	CustomerID	Spending Score (1-100)	y_pred
105	106	42	0
106	107	50	0
107	108	46	0
108	109	43	0
109	110	48	0

In [52]:

```
segment_2 = new_customer_data_hier[new_customer_data_hier['y_pred']==1]
segment_2.head()
```

Out[52]:

	CustomerID	Spending Score (1-100)	y_pred
0	1	39	1
1	2	81	1
2	3	6	1
3	4	77	1
4	5	40	1

In [53]:

```
segment_3 = new_customer_data_hier[new_customer_data_hier['y_pred']==2]
segment_3.head()
```

Out[53]:

	CustomerID	Spending Score (1-100)	y_pred
123	124	91	2
125	126	77	2
127	128	95	2
129	130	75	2
131	132	75	2

In [54]:

```
segment_4 = new_customer_data_hier[new_customer_data_hier['y_pred']==3]
segment_4.head()
```

Out[54]:

	CustomerID	Spending Score (1-100)	y_pred
43	44	61	3
45	46	65	3
46	47	55	3
47	48	47	3
48	49	42	3

RESULTS

From the analysis, it's evident that both K-Means clustering and Hierarchical clustering produce similar results in terms of segregating the dataset. The customer dataset has been effectively divided into four clusters based on the attributes of CustomerID and Spending score.

RECOMMENDATION

We recommend that the retail company implement strategies tailored to each customer segments through;

1. Customized Product recommendation: Send personalized emails with product recommendations based on specific segments need. This ensures that customers receive offers and products that are relevant to their interests.
2. Personalized Marketing Campaigns: Tailor advertising messages to resonate with the values and interests of specific customer segments.This approach enhances engagement and increases the likelihood of conversions.

CONCLUSION

We performed both KMean clustering and Hierarchical clustering using the Agglomerative Clustering to segregate the customer dataset. We believe implementing the above recommendations will help the retail store increase productivity and gain competitive advantage.

In [ ]:

```

```