Aim: Implementation of Clustering Algorithm using, 1) K-means
2)Hierarchical(complete)

Theory: Clustering or cluster analysis is a machine learning technique, which groups the unlabelled dataset. It can be defined as "A way of grouping the data points into different clusters, consisting of similar data points. The objects with the possible similarities remain in a group that has less or no similarities with another group."

It does it by finding some similar patterns in the unlabelled dataset such as shape, size, color, behavior, etc., and divides them as per the presence and absence of those similar patterns.

It is an unsupervised learning method; hence no supervision is provided to the algorithm, and it deals with the unlabeled dataset.

After applying this clustering technique, each cluster or group is provided with a cluster-ID. ML system can use this id to simplify the processing of large and complex datasets.

1] Hierarchical Clustering: Hierarchical clustering can be used as an alternative for the partitioned clustering as there is no requirement of pre-specifying the number of clusters to be created. In this technique, the dataset is divided into clusters to create a tree-like structure, which is also called a dendrogram. The observations or any number of clusters can be selected by cutting the tree at the correct level. The most common example of this method is the Agglomerative Hierarchical algorithm.

2] K-Means algorithm: The k-means algorithm is one of the most popular clustering algorithms. It classifies the dataset by dividing the samples into different clusters of equal variances. The number of clusters must be specified in this algorithm. It is fast with fewer computations required, with the linear complexity of O(n).

3] Elbow Method: In cluster analysis, the elbow method is a heuristic used in determining the number of clusters in a data set. The method consists of plotting the explained variation as a function of the number of clusters and picking the elbow of the curve as the number of clusters to use.

Code:

In [ ]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans, AgglomerativeClustering, DBSCAN
import seaborn as sns
```

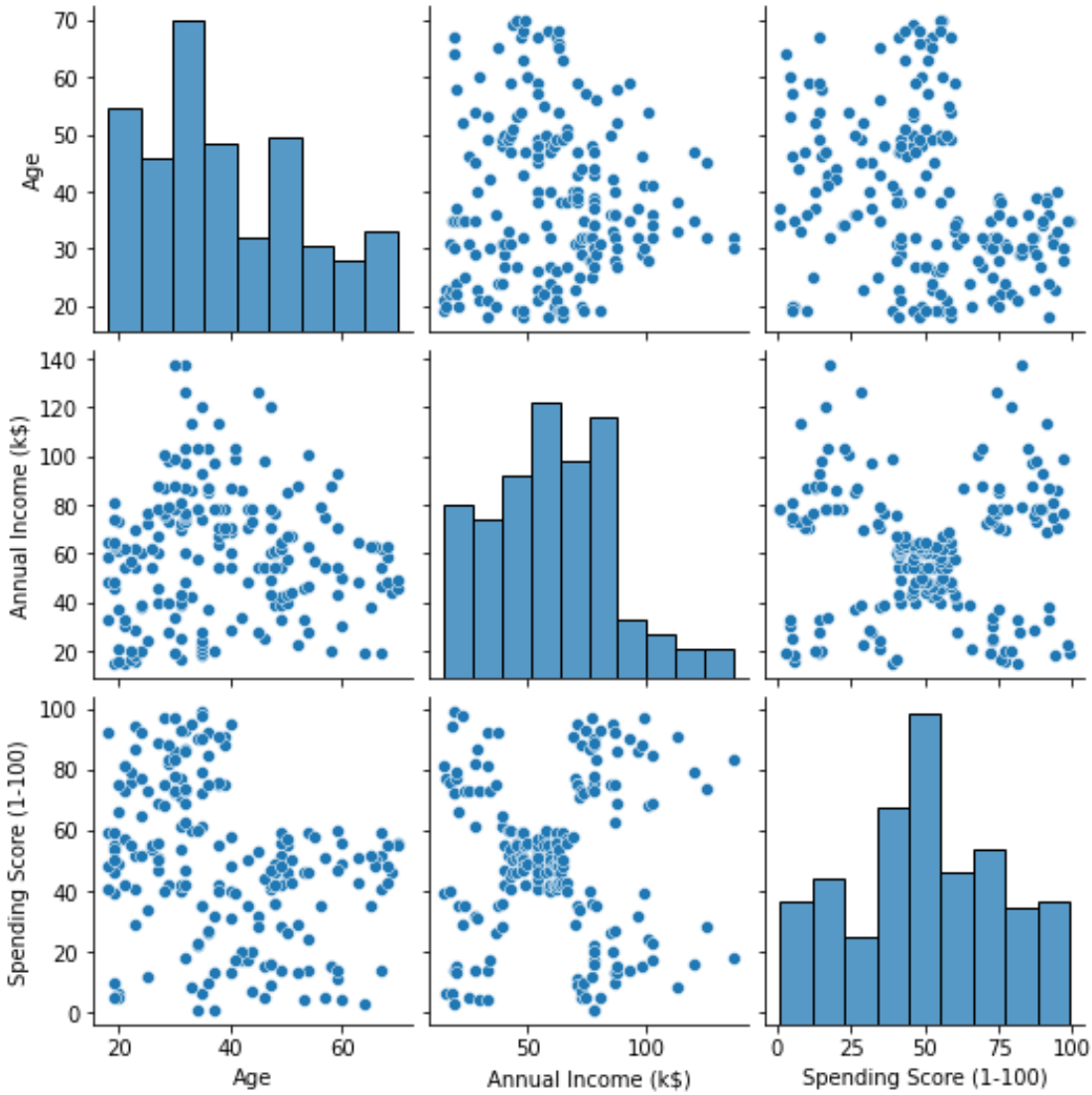
In [ ]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

dataset = pd.read_csv('/content/Mall_Customers.csv')
x = dataset.iloc[:, [3, 4]].values
dataset.head()
sns.pairplot(dataset[['Age','Annual Income (k$)','Spending Score (1-100)']])
```

Out[ ]:

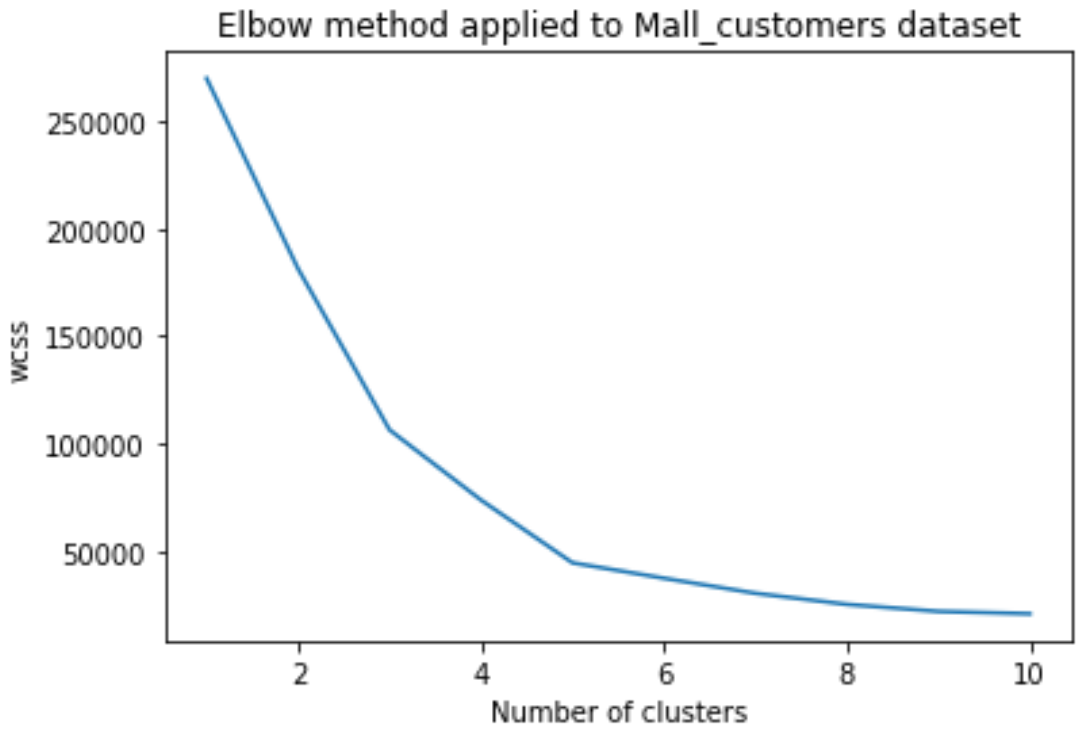
<seaborn.axisgrid.PairGrid at 0x7ff922901a10>



In [ ]:

```
from sklearn.cluster import KMeans
wcss = []

#Iterating over 1, 2, 3, ---- 10 clusters
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', max_iter = 300, n_init = 10,
random_state = 0)
    kmeans.fit(x)
    wcss.append(kmeans.inertia_) # intertia_ is an attribute that has the wcss number
plt.plot(range(1,11), wcss)
plt.title("Elbow method applied to Mall_customers dataset")
plt.xlabel("Number of clusters")
plt.ylabel("wcss")
plt.show()
```



In [ ]:

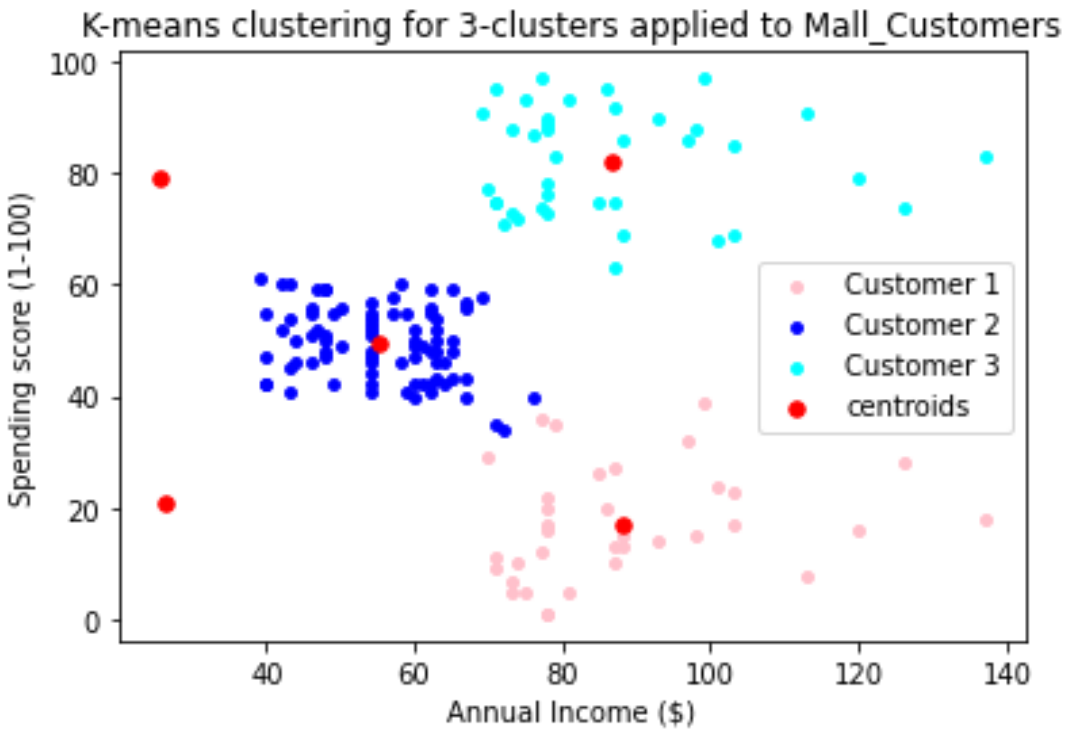
```
# Apply kmeans to the dataset
kmeans = KMeans(n_clusters = 5, max_iter = 300, init = 'k-means++', n_init = 10,
random_state = 0)
y_kmeans = kmeans.fit_predict(x) #predict which cluster each point belongs to
y_kmeans
```

Out[ ]:

```
array([4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3,
       4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 1,
       4, 3, 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, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 0, 2, 1, 2, 0, 2, 0, 2,
       1, 2, 0, 2, 0, 2, 0, 2, 0, 2, 1, 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=int32)
```

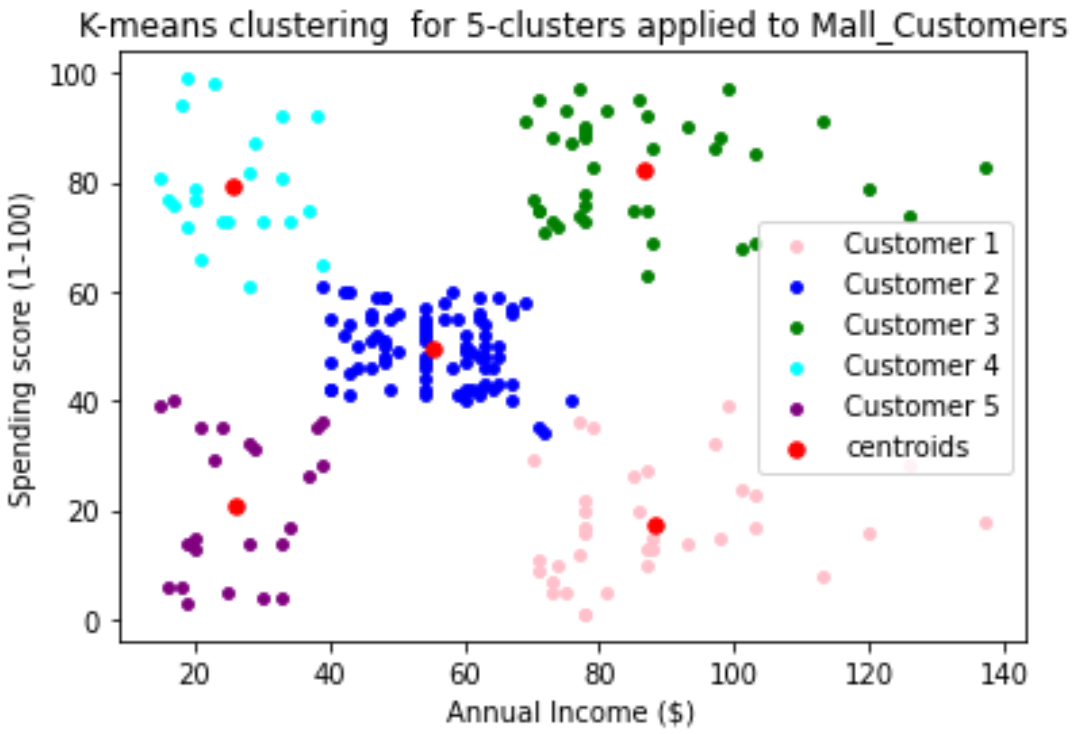
In [ ]:

```
plt.figure(2)
plt.scatter(x[y_kmeans == 0, 0], x[y_kmeans == 0, 1], s = 15, c = 'pink', label = 'Customer 1')
plt.scatter(x[y_kmeans == 1, 0], x[y_kmeans == 1, 1], s = 15, c = 'blue', label = 'Customer 2')
plt.scatter(x[y_kmeans == 2, 0], x[y_kmeans == 2, 1], s = 15, c = 'cyan', label = 'Customer 3')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 30, c = 'red', label = 'centroids' )
plt.title("K-means clustering for 3-clusters applied to Mall_Customers")
plt.xlabel('Annual Income ($)')
plt.ylabel('Spending score (1-100)')
plt.legend()
plt.show()
```



In [ ]:

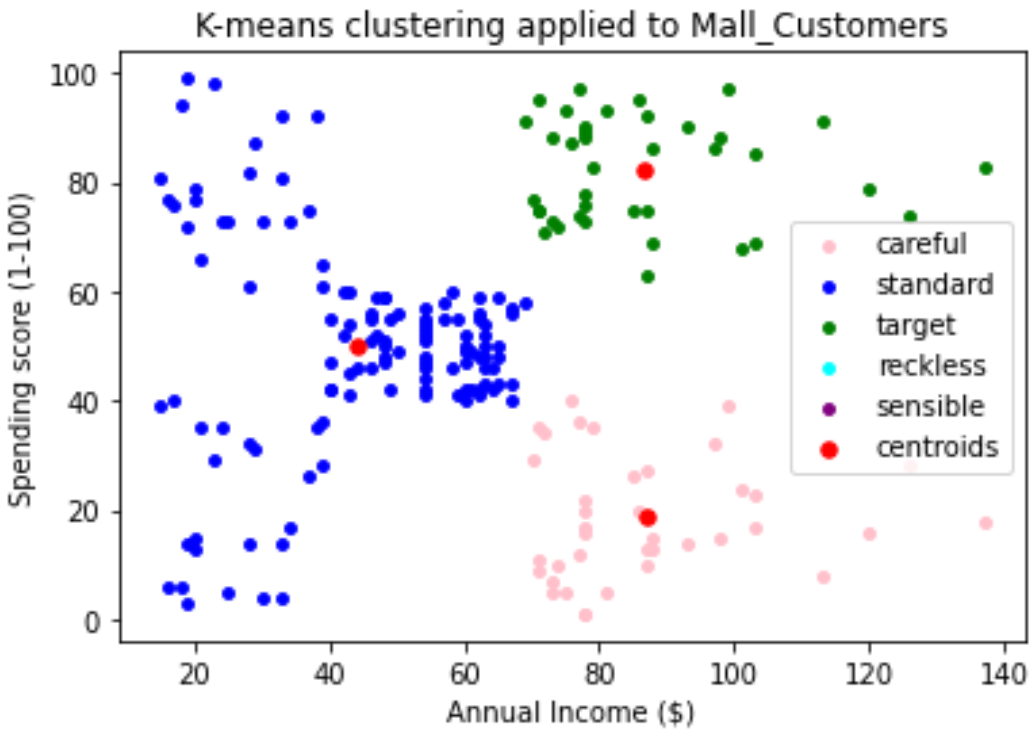
```
plt.figure(3)
plt.scatter(x[y_kmeans == 0, 0], x[y_kmeans == 0, 1], s = 15, c = 'pink', label = 'Customer 1')
plt.scatter(x[y_kmeans == 1, 0], x[y_kmeans == 1, 1], s = 15, c = 'blue', label = 'Customer 2')
plt.scatter(x[y_kmeans == 2, 0], x[y_kmeans == 2, 1], s = 15, c = 'green', label = 'Customer 3')
plt.scatter(x[y_kmeans == 3, 0], x[y_kmeans == 3, 1], s = 15, c = 'cyan', label = 'Customer 4')
plt.scatter(x[y_kmeans == 4, 0], x[y_kmeans == 4, 1], s = 15, c = 'purple', label = 'Customer 5')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 30, c = 'red', label = 'centroids' )
plt.title("K-means clustering  for 5-clusters applied to Mall_Customers")
plt.xlabel('Annual Income ($)')
plt.ylabel('Spending score (1-100)')
plt.legend()
plt.show()
```



In [ ]:

#Now that each group is identified, we can name each category.

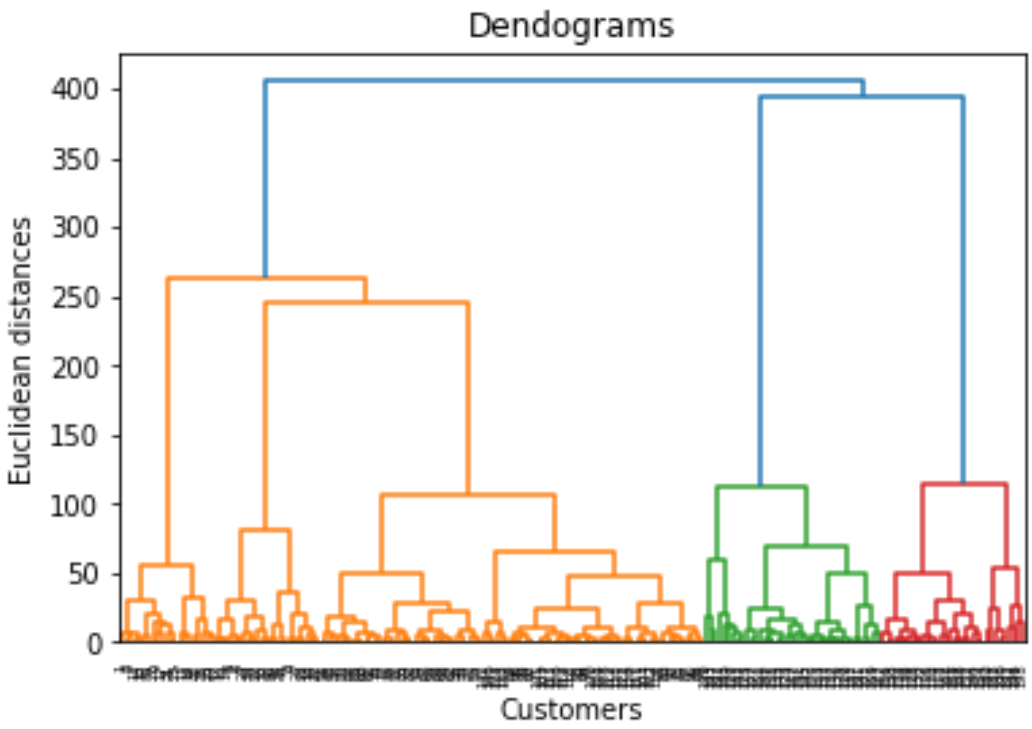
```
plt.figure(3)
plt.scatter(x[y_kmeans == 0, 0], x[y_kmeans == 0, 1], s = 15, c = 'pink', label = 'careful')
plt.scatter(x[y_kmeans == 1, 0], x[y_kmeans == 1, 1], s = 15, c = 'blue', label =
'standard')
plt.scatter(x[y_kmeans == 2, 0], x[y_kmeans == 2, 1], s = 15, c = 'green', label = 'target')
plt.scatter(x[y_kmeans == 3, 0], x[y_kmeans == 3, 1], s = 15, c = 'cyan', label = 'reckless')
plt.scatter(x[y_kmeans == 4, 0], x[y_kmeans == 4, 1], s = 15, c = 'purple', label =
'sensible')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 30, c = 'red',
label = 'centroids' )
plt.title("K-means clustering applied to Mall_Customers")
plt.xlabel('Annual Income ($)')
plt.ylabel('Spending score (1-100)')
plt.legend()
plt.show()
```



In [ ]:

```
x1 = dataset.iloc[:, [3, 4]].values
# y is not available, as we do not know the patterns we expect. It's our job to find patterns from x.

#Use dendrogram to find the optimal number of clusters
import scipy.cluster.hierarchy as sch
dendrogram_1 = sch.dendrogram(sch.linkage(x1, method = 'ward'))
plt.title('Dendograms')
plt.xlabel('Customers')
plt.ylabel('Euclidean distances')
plt.show()
# ward method tries to minimize variance between clusters
```

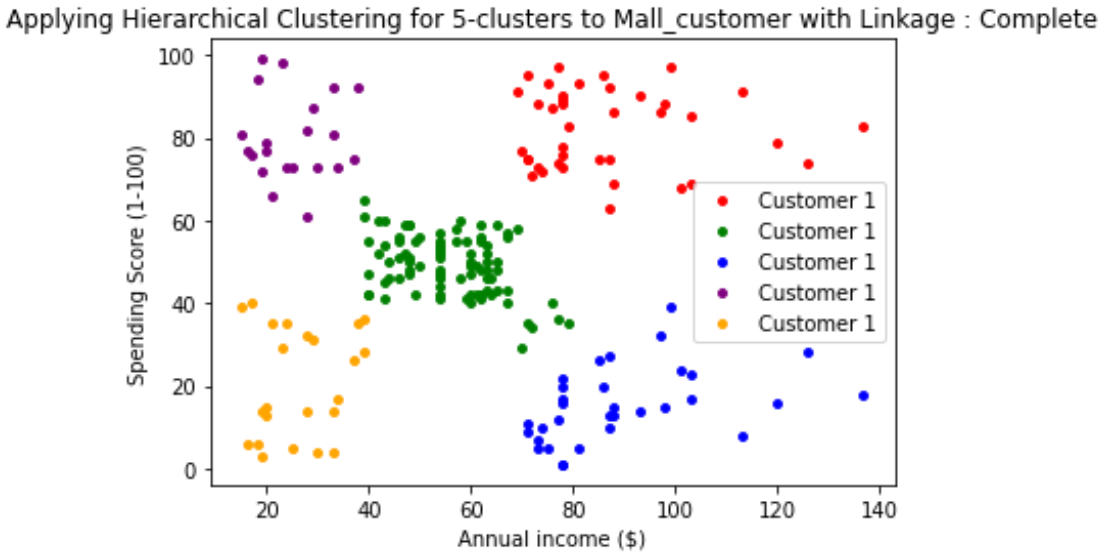


In [ ]:

```
# Fitting HC to the mall dataset

hc = AgglomerativeClustering(n_clusters = 5, affinity = 'euclidean', linkage = 'complete')
y_hc = hc.fit_predict(x)
plt.scatter(x[y_hc == 0, 0], x[y_hc == 0, 1], s = 15, c = 'red', label = 'Customer 1')
plt.scatter(x[y_hc == 1, 0], x[y_hc == 1, 1], s = 15, c = 'green', label = 'Customer 1')
plt.scatter(x[y_hc == 2, 0], x[y_hc == 2, 1], s = 15, c = 'blue', label = 'Customer 1')
plt.scatter(x[y_hc == 3, 0], x[y_hc == 3, 1], s = 15, c = 'purple', label = 'Customer 1')
plt.scatter(x[y_hc == 4, 0], x[y_hc == 4, 1], s = 15, c = 'orange', label = 'Customer 1')
plt.title('Applying Hierarchical Clustering for 5-clusters to Mall_customer with Linkage :
Complete')
plt.xlabel('Annual income ($)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()

#The above segment of code is only for visualizing 2D data, not higher-dimension
```


Applying Hierarchical Clustering for 5-clusters to Mall_customer with Linkage : Complete
Spending Score (1-100)
Annual income ($)
Customer 1
Customer 1
Customer 1
Customer 1
Customer 1
0
20
40
60
80
100
20
40
60
80
100
120
140