Name: Meith Navlakha
SAP: 60004190068
DWM Practical

Experiment 4

AIM: Implementation of Clustering Algorithm

1. KMeans
2. Hierarchical (Single/Average/Complete)

THEORY:

KMeans:

K-Means Clustering is an unsupervised learning algorithm that is used to solve the clustering problems in machine learning or data science. It is an iterative algorithm that tries to partition the dataset into K pre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to only one group, making the intra-cluster data points as similar as possible while also keeping the clusters as different (far) as possible.

It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

Hierarchical Clustering:

A Hierarchical clustering method works via grouping data into a tree of clusters. The endpoint is a set of clusters, where each cluster is distinct from each other cluster, and the objects within each cluster are broadly similar to each other. Hierarchical clustering typically works by sequentially merging similar clusters, this known as agglomerative hierarchical clustering. In theory, it can also be done by initially grouping all the observations into one cluster, and then successively splitting these clusters.

Syntax: linkage{'ward', 'complete', 'average', 'single'}, default='ward'

The linkage criterion determines which distance to use between sets of observation. The algorithm will merge the pairs of cluster that minimize this criterion.

- **'ward'** minimizes the variance of the clusters being merged.
- **'average'** uses the average of the distances of each observation of the two sets.
- **'complete'** or 'maximum' linkage uses the maximum distances between all observations of the two sets.
- **'single'** uses the minimum of the distances between all observations of the two sets.

CODE:

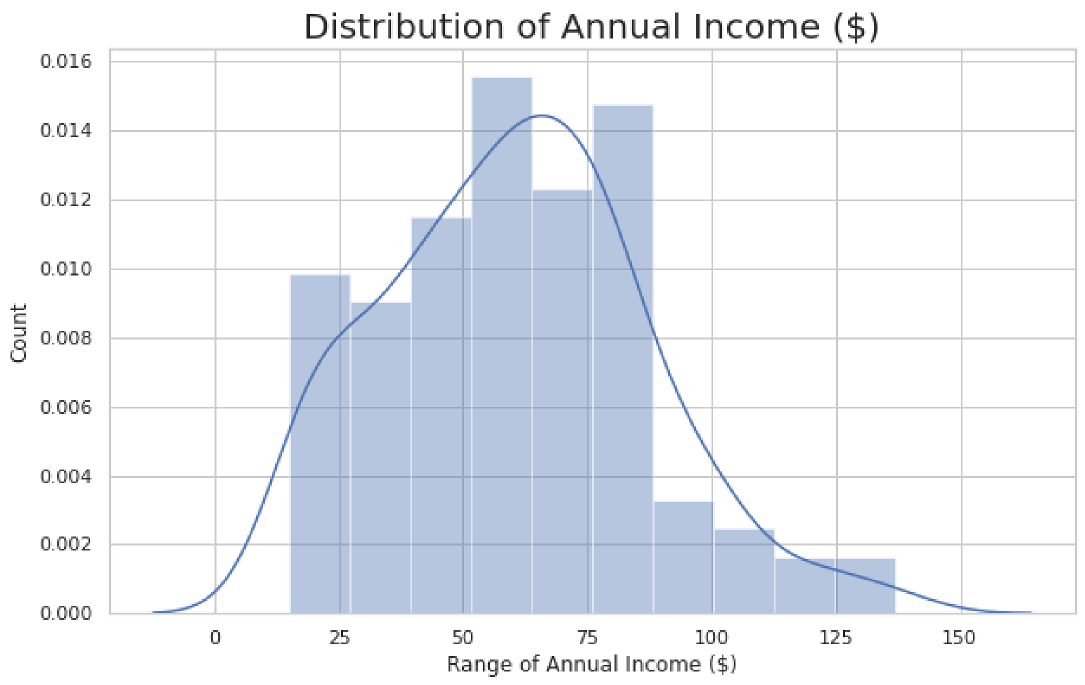
Part A: Program using inbuilt functions.

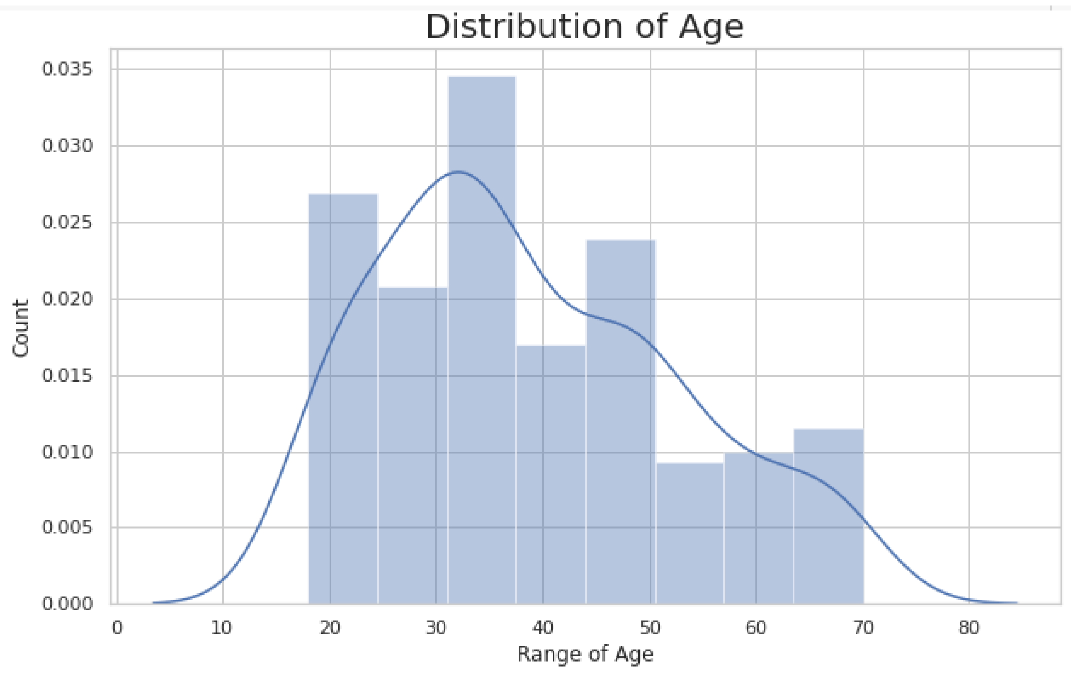
i) KMeans

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from mpl_toolkits.mplot3d import Axes3D
%matplotlib inline
from sklearn.cluster import KMeans

ds = pd.read_csv('Mall_Customers.csv')
plt.figure(figsize=(10, 6))
sns.set(style = 'whitegrid')
sns.distplot(ds['Annual Income (k$)'])
plt.title('Distribution of Annual Income ($)', fontsize = 20)
plt.xlabel('Range of Annual Income ($)')
plt.ylabel('Count')
plt.figure(figsize=(10, 6))
# sns.set(style = 'whitegrid')
sns.distplot(ds['Age'])
plt.title('Distribution of Age', fontsize = 20)
plt.xlabel('Range of Age')
plt.ylabel('Count')
genders = ds.Genre.value_counts()
# sns.set_style("darkgrid")
plt.figure(figsize=(10,4))
sns.barplot(x=genders.index, y=genders.values)
plt.show()
df1=ds[["CustomerID","Genre","Age","Annual Income (k$)","Spending Score (1-100)"]]
X=df1[["Annual Income (k$)","Spending Score (1-100)"]]
plt.figure(figsize=(10,6))
sns.scatterplot(x = 'Annual Income (k$)',y = 'Spending Score (1-100)',  data = X  ,s = 60 )
plt.xlabel('Annual Income ($)')
plt.ylabel('Spending Score (1-100)')
plt.title('Spending Score (1-100) vs Annual Income ($)')
plt.show()
list1=[]
for i in range(1,11):
    km=KMeans(n_clusters=i)
    km.fit(X)
    list1.append(km.inertia_)
km1=KMeans(n_clusters=5)
km1.fit(X)
y=km1.predict(X)
df1["label"] = y
df1.head()
plt.figure(figsize=(10,6))
sns.scatterplot(x = 'Annual Income (k$)',y = 'Spending Score (1-100)',hue="label",
                 palette=['yellow','orange','brown','red','violet'], legend='full',data = df1  ,s = 60
)
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.title('Spending Score (1-100) vs Annual Income (k$)')
plt.show()
cust1=df1[df1["label"]==1]
print('Number of customer in 1st group=', len(cust1))
print('They are -', cust1["CustomerID"].values)
print("--------------------------------------------")
cust2=df1[df1["label"]==2]
print('Number of customer in 2nd group=', len(cust2))
print('They are -', cust2["CustomerID"].values)
print("--------------------------------------------")
cust3=df1[df1["label"]==0]
print('Number of customer in 3rd group=', len(cust3))
print('They are -', cust3["CustomerID"].values)
print("--------------------------------------------")
cust4=df1[df1["label"]==3]
print('Number of customer in 4th group=', len(cust4))
print('They are -', cust4["CustomerID"].values)
print("--------------------------------------------")
cust5=df1[df1["label"]==4]
```

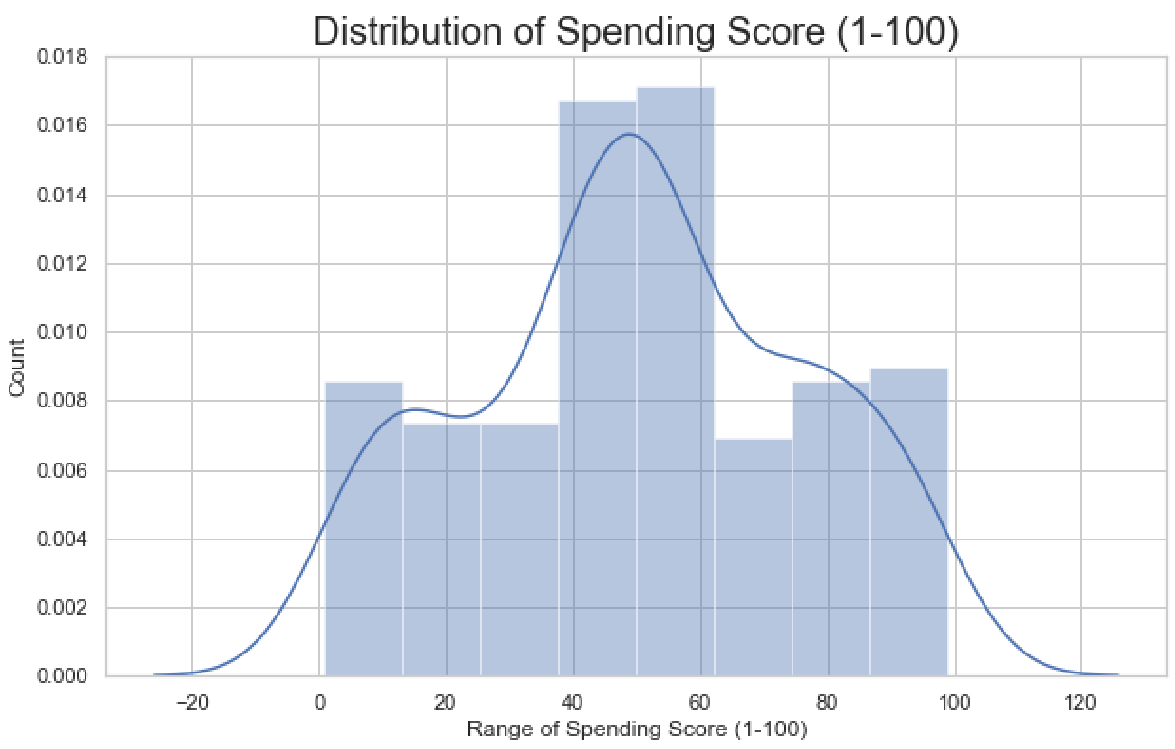
```
print('Number of customer in 5th group=', len(cust5))
print('They are -', cust5["CustomerID"].values)
print("--------------------------------------------")
```

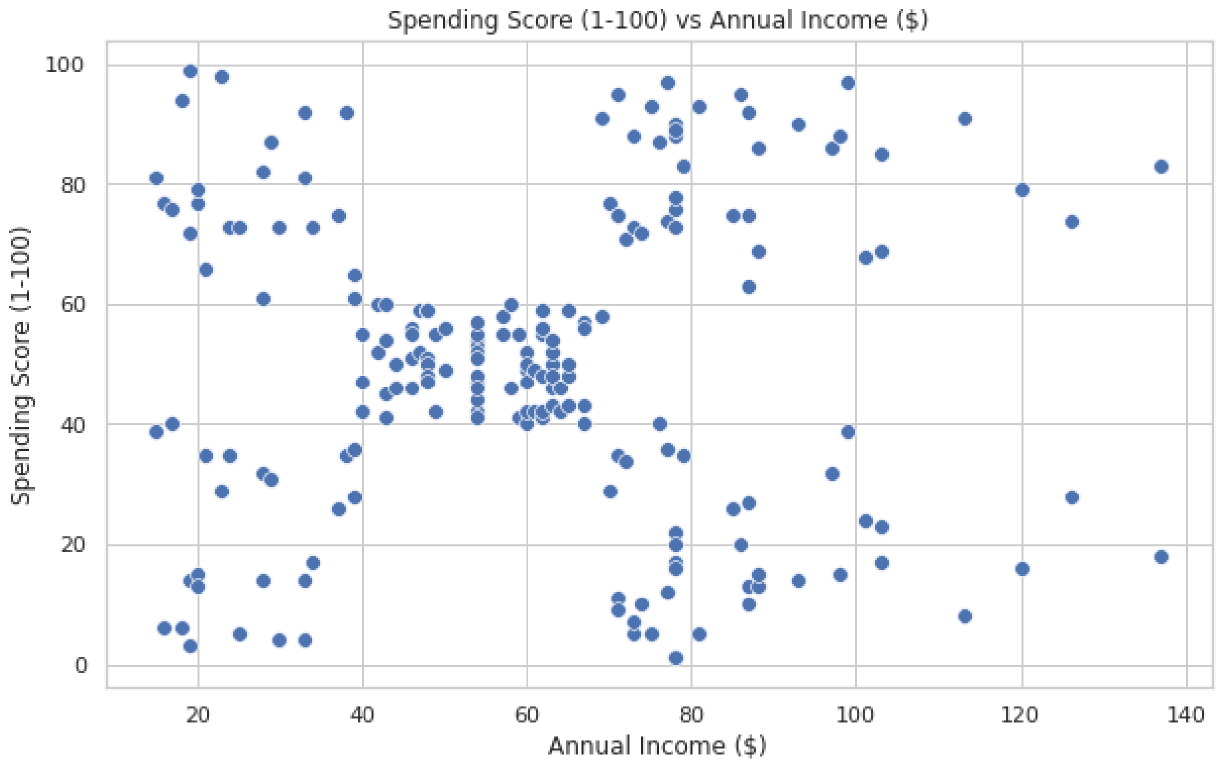
OUTPUT:



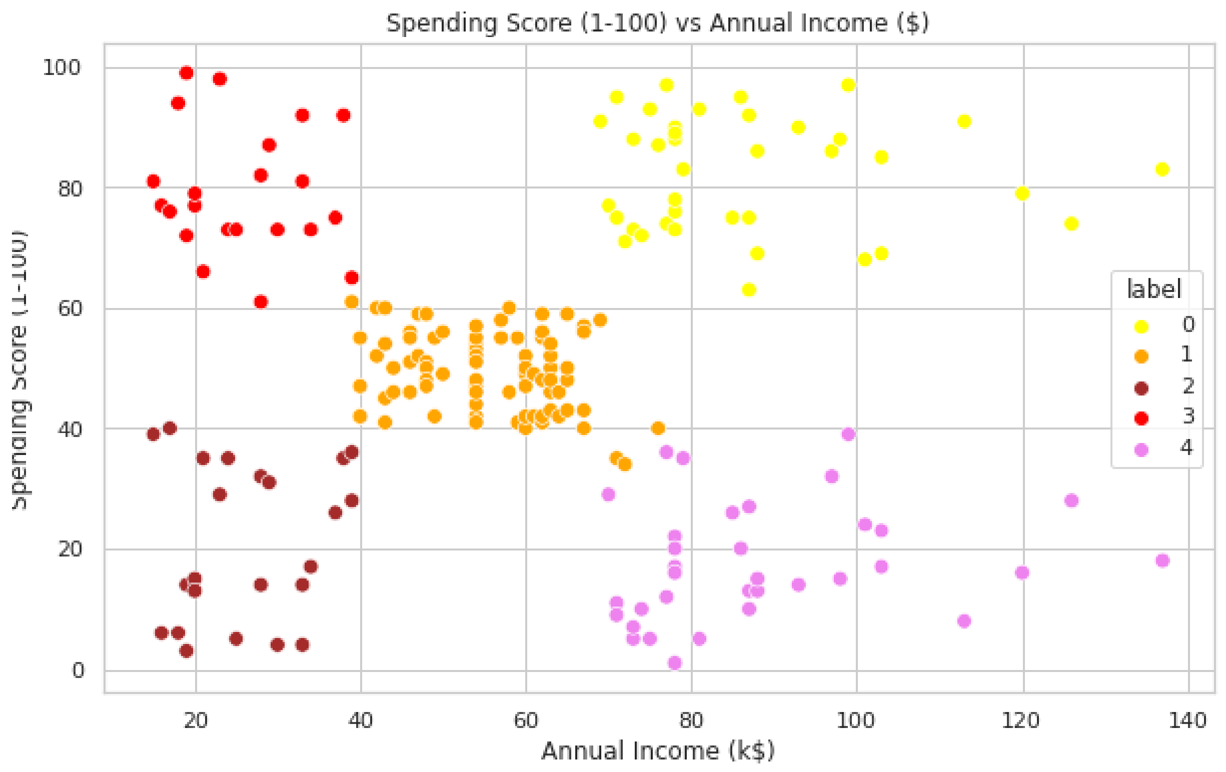


Name: Meith Navlakha
SAP: 60004190068
DWM Practical





Name: Meith Navlakha
SAP: 60004190068
DWM Practical



```
Number of customer in 1st group= 81
They are - [ 44  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63
  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81
  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99
 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117
 118 119 120 121 122 123 127 133 143]
--------------------------------------------
Number of customer in 2nd group= 23
They are - [ 1  3  5  7  9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45]
--------------------------------------------
Number of customer in 3rd group= 39
They are - [124 126 128 130 132 134 136 138 140 142 144 146 148 150 152 154 156 158
 160 162 164 166 168 170 172 174 176 178 180 182 184 186 188 190 192 194
 196 198 200]
--------------------------------------------
Number of customer in 4th group= 22
They are - [ 2  4  6  8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 46]
--------------------------------------------
Number of customer in 5th group= 35
They are - [125 129 131 135 137 139 141 145 147 149 151 153 155 157 159 161 163 165
 167 169 171 173 175 177 179 181 183 185 187 189 191 193 195 197 199]
--------------------------------------------
```

ii) Hierarchical Clustering

Code:

```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import scipy.cluster.hierarchy as sch
from sklearn.cluster import AgglomerativeClustering

# Importing the dataset
dataset = pd.read_csv('Mall_Customers.csv')
X = dataset.iloc[:, [3, 4]].values
dendrogram = sch.dendrogram(sch.linkage(X, method = 'single'))
plt.title('Dendrogram')
plt.xlabel('Customers')
plt.ylabel('Euclidean distances')
plt.show()
hc = AgglomerativeClustering(n_clusters = 5, affinity = 'euclidean', linkage = 'single')
y_hc = hc.fit_predict(X)
print(y_hc)
# Visualising the clusters
plt.scatter(X[y_hc == 0, 0], X[y_hc == 0, 1], s = 100, c = 'red', label = 'Cluster 1')
plt.scatter(X[y_hc == 1, 0], X[y_hc == 1, 1], s = 100, c = 'blue', label = 'Cluster 2')
plt.scatter(X[y_hc == 2, 0], X[y_hc == 2, 1], s = 100, c = 'yellow', label = 'Cluster 3')
plt.scatter(X[y_hc == 3, 0], X[y_hc == 3, 1], s = 100, c = 'green', label = 'Cluster 4')
plt.scatter(X[y_hc == 4, 0], X[y_hc == 4, 1], s = 100, c = 'pink', label = 'Cluster 5')
plt.title('Clusters of customers')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
dendrogram1 = sch.dendrogram(sch.linkage(X, method = 'single'))
plt.title('Dendrogram')
plt.xlabel('Customers')
plt.ylabel('Euclidean distances')
plt.show()
```

```
hc1 = AgglomerativeClustering(n_clusters = 5, affinity = 'euclidean', linkage = 'average')
y_hc1 = hc1.fit_predict(X)
print(y_hc1)
# Visualising the clusters
plt.scatter(X[y_hc1 == 0, 0], X[y_hc1 == 0, 1], s = 100, c = 'red', label = 'Clust1')
plt.scatter(X[y_hc1== 1, 0], X[y_hc1 == 1, 1], s = 100, c = 'violet', label = 'Clust2')
plt.scatter(X[y_hc1== 2, 0], X[y_hc1 == 2, 1], s = 100, c = 'yellow', label = 'Clust3')
plt.scatter(X[y_hc1== 3, 0], X[y_hc1 == 3, 1], s = 100, c = 'cyan', label = 'Clust4')
plt.scatter(X[y_hc1 == 4, 0], X[y_hc1 == 4, 1], s = 100, c = 'pink', label = 'Clust5')
plt.title('Clusters of customers')
plt.xlabel('Annual Income ($)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
dendrogram2 = sch.dendrogram(sch.linkage(X, method = 'complete'))
plt.title('Dendrogram')
plt.xlabel('Customers')
plt.ylabel('Euclidean distances')
plt.show()
hc2 = AgglomerativeClustering(n_clusters = 5, affinity = 'euclidean', linkage =
'complete')
y_hc2 = hc2.fit_predict(X)
print(y_hc2)
# Visualising the clusters
plt.scatter(X[y_hc2 == 0, 0], X[y_hc2 == 0, 1], s = 100, c = 'red', label = 'Clust1')
plt.scatter(X[y_hc2 == 1, 0], X[y_hc2 == 1, 1], s = 100, c = 'violet', label = 'Clust2')
plt.scatter(X[y_hc2 == 2, 0], X[y_hc2 == 2, 1], s = 100, c = 'green', label = 'Clust3')
plt.scatter(X[y_hc2 == 3, 0], X[y_hc2 == 3, 1], s = 100, c = 'cyan', label = 'Clust4')
plt.scatter(X[y_hc2 == 4, 0], X[y_hc2 == 4, 1], s = 100, c = 'orange', label = 'Clust5')
plt.title('Clusters of customers')
plt.xlabel('Annual Income ($)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
```

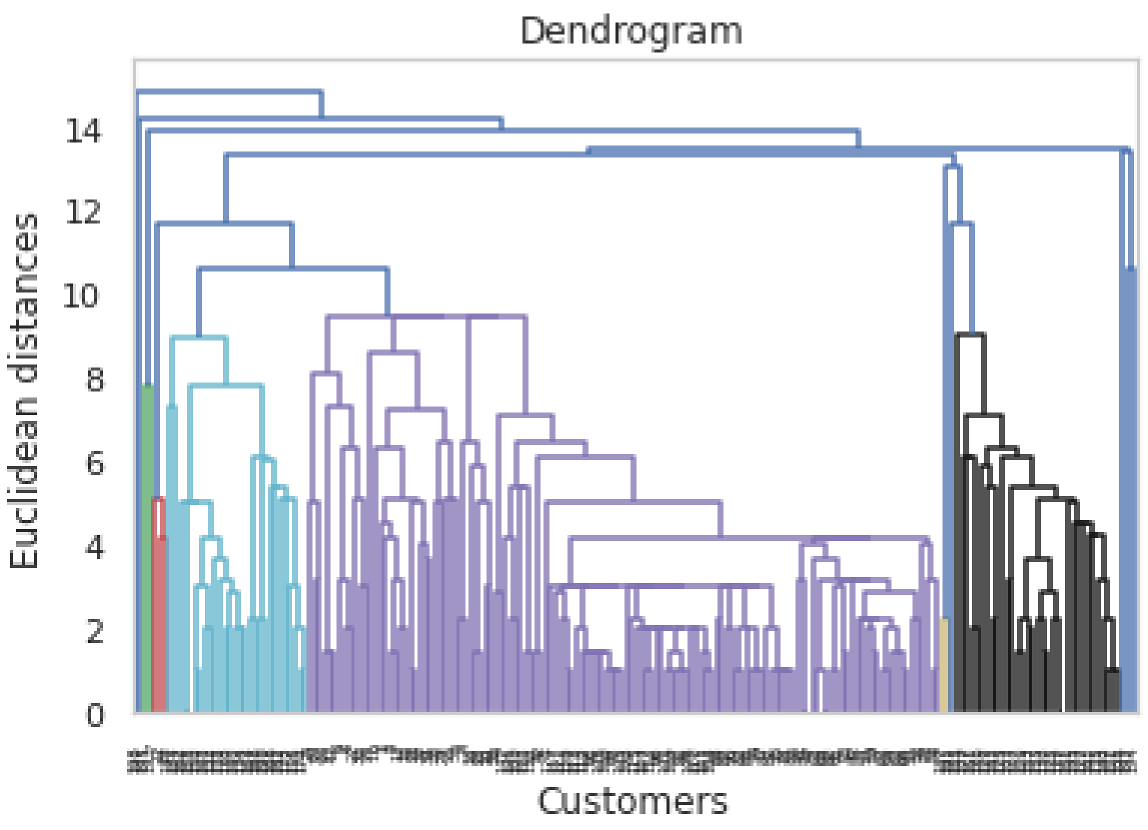
Name: Meith Navlakha

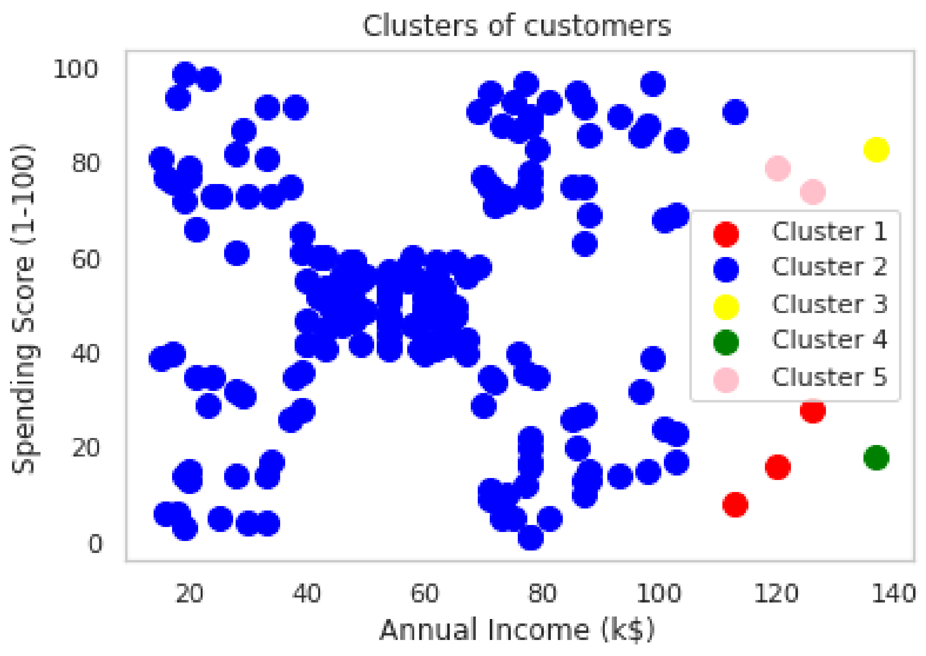
SAP: 60004190068

DWM Practical

Output:

Dendrogram -- Single



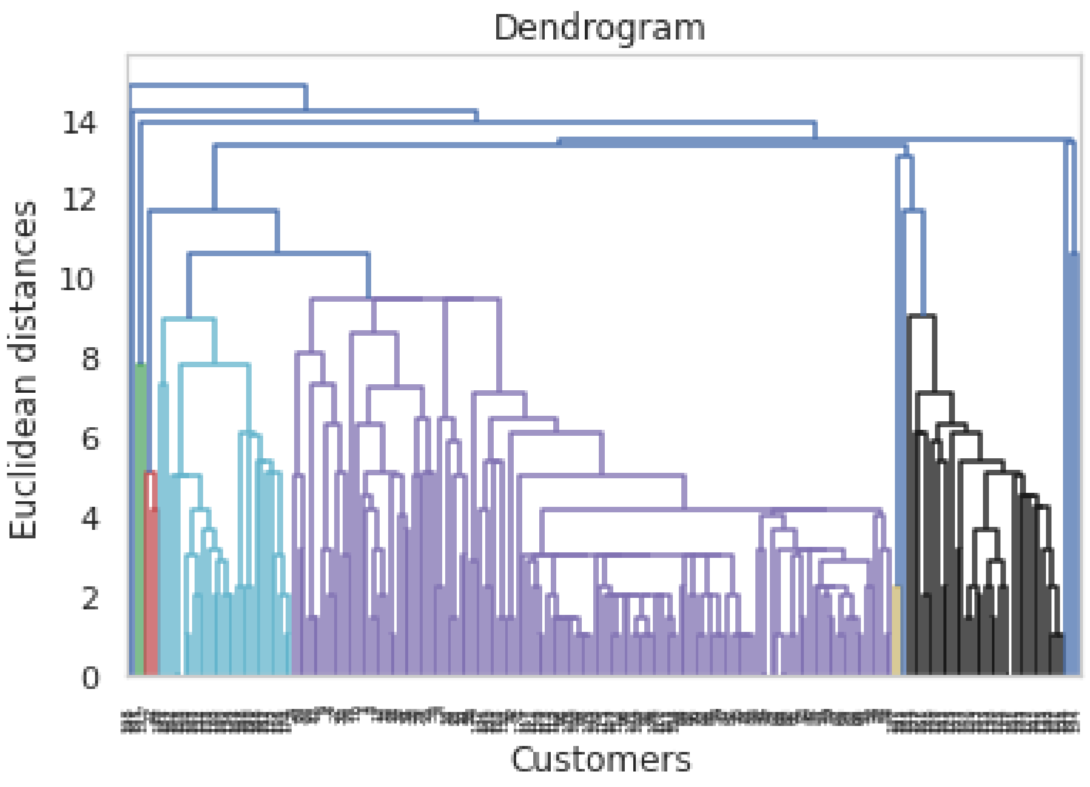


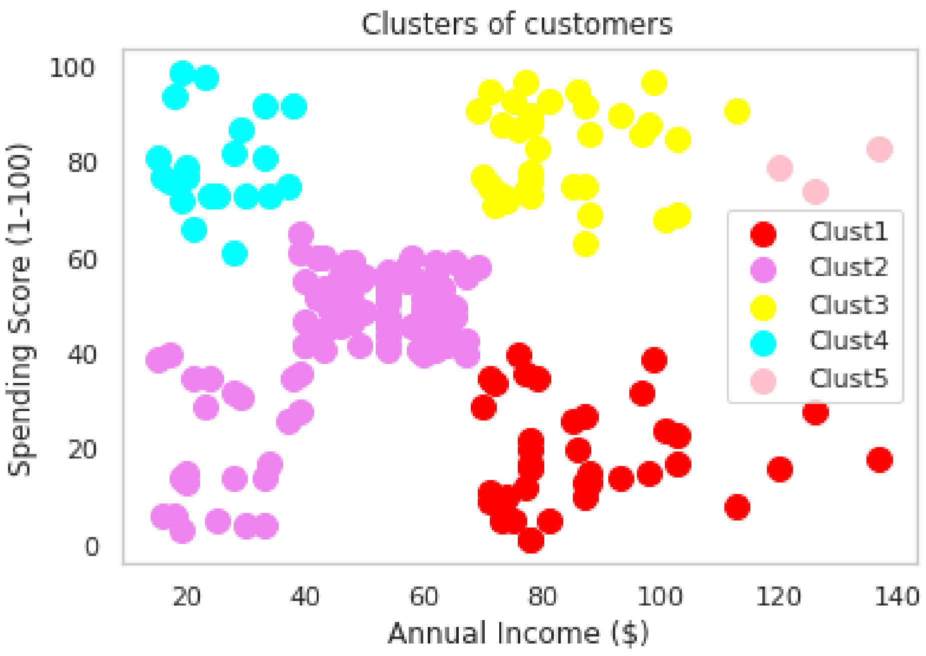
Name: Meith Navlakha

SAP: 60004190068

DWM Practical

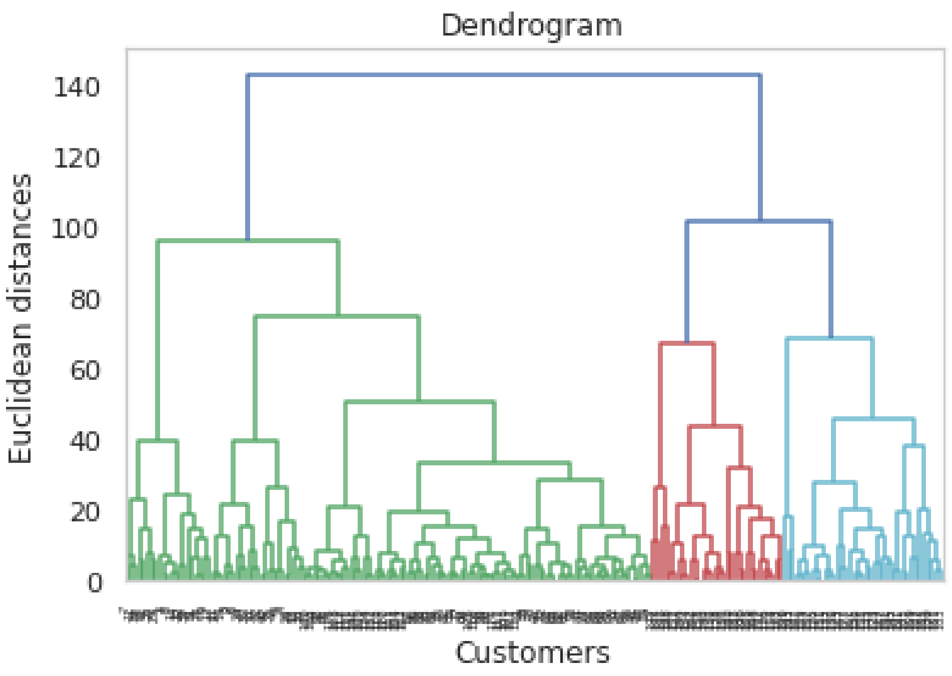
Dendrogram -- Average

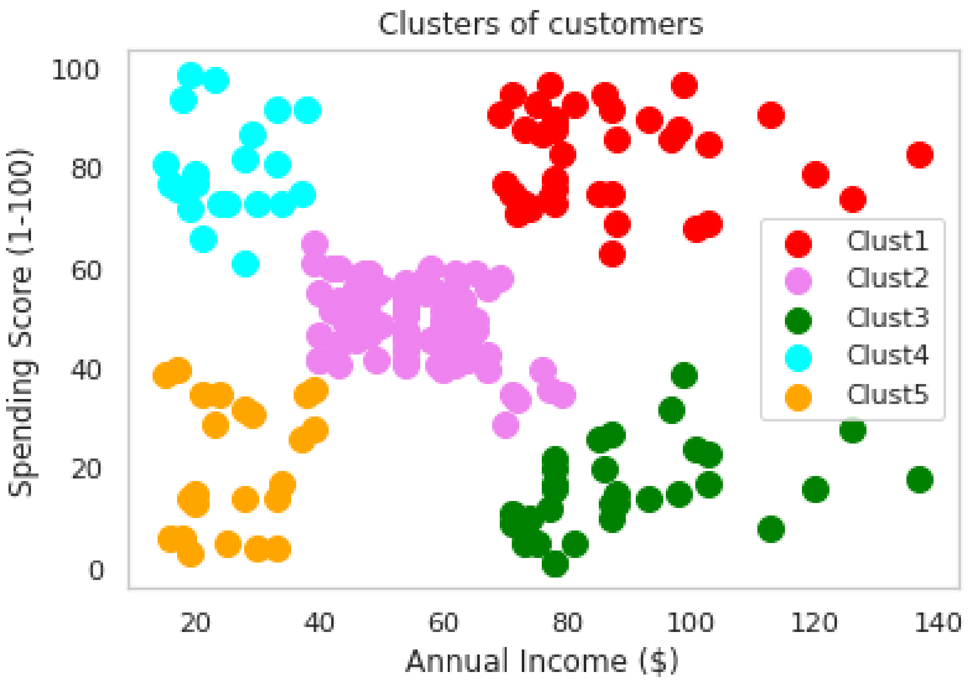




Name: Meith Navlakha
SAP: 60004190068
DWM Practical

Dendrogram -- Complete





Name: Meith Navlakha
SAP: 60004190068
DWM Practical

PART B: Program the algorithm from scratch.

i) KMeans

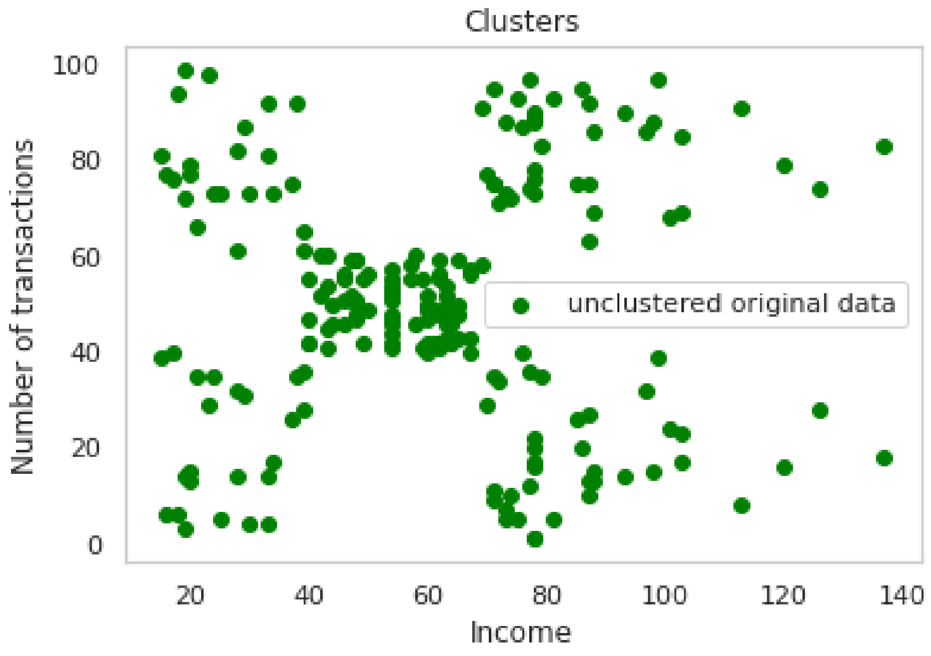
Code:

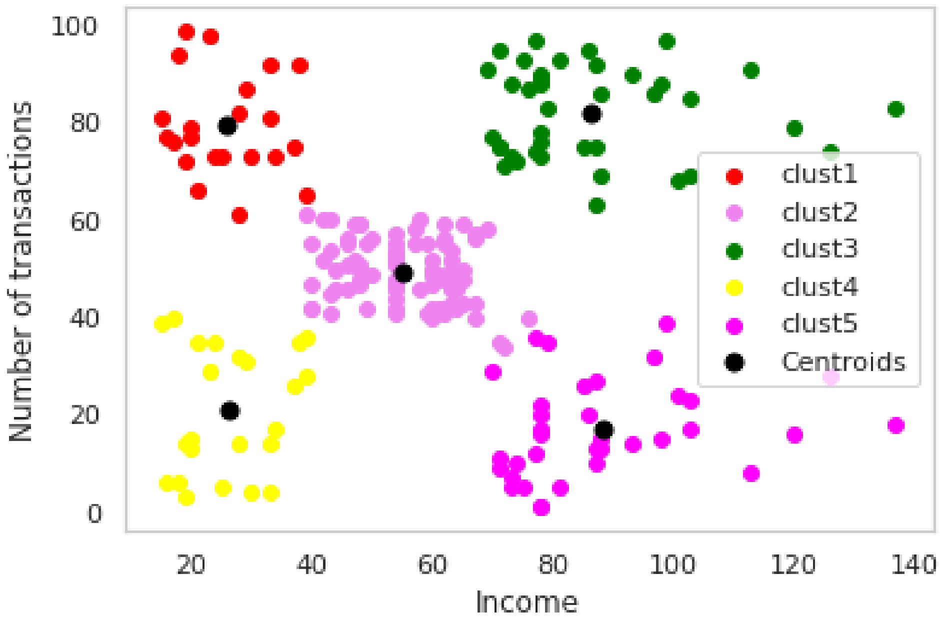
```
import pandas as pd
from matplotlib import pyplot as plt
import numpy as np
from sklearn.cluster import KMeans
import random as rd

dataset=pd.read_csv('Mall_Customers.csv')
X = dataset.iloc[:, [3, 4]].values
m=X.shape[0] # Training
n=X.shape[1] # n=2
n_iter=100
K=5 # number of clusters

Centroids=np.array([]).reshape(n,0)
for i in range(K):
   rand=rd.randint(0,m-1)
   Centroids=np.c_[Centroids,X[rand]]
Output={}
EuclidianDistance=np.array([]).reshape(m,0)
for k in range(K):
   tempDist=np.sum((X-Centroids[:,k])**2,axis=1)
   EuclidianDistance=np.c_[EuclidianDistance,tempDist]
C=np.argmin(EuclidianDistance,axis=1)+1
Y={}
for k in range(K):
   Y[k+1]=np.array([]).reshape(2,0)
for i in range(m):
   Y[C[i]]=np.c_[Y[C[i]],X[i]]
for k in range(K):
   Y[k+1]=Y[k+1].T
for k in range(K):
    Centroids[:,k]=np.mean(Y[k+1],axis=0)
```

```
for i in range(n_iter):
    #step 2.a
     EuclidianDistance=np.array([]).reshape(m,0)
     for k in range(K):
        tempDist=np.sum((X-Centroids[:,k])**2,axis=1)
        EuclidianDistance=np.c_[EuclidianDistance,tempDist]
     C=np.argmin(EuclidianDistance,axis=1)+1
    #step 2.b
     Y={}
     for k in range(K):
        Y[k+1]=np.array([]).reshape(2,0)
     for i in range(m):
        Y[C[i]]=np.c_[Y[C[i]],X[i]]

     for k in range(K):
        Y[k+1]=Y[k+1].T

     for k in range(K):
        Centroids[:,k]=np.mean(Y[k+1],axis=0)

     Output=Y

plt.scatter(X[:,0],X[:,1],c='green',label='unclustered original data')
plt.xlabel('Income')
plt.ylabel('Number of transactions')
plt.legend()
plt.title('Clusters')
plt.show()
color=['red','violet','green','yellow','magenta']
labels=['clust1','clust2','clust3','clust4','clust5']
for k in range(K):
    plt.scatter(Output[k+1][:,0],Output[k+1][:,1],c=color[k],label=labels[k])
plt.scatter(Centroids[0,:],Centroids[1,:],s=50,c='black',label='Centroids')
plt.xlabel('Income')
plt.ylabel('Number of transactions')
plt.legend()
plt.show()
```

Name: Meith Navlakha
SAP: 60004190068
DWM Practical

Output:





ii) Hierarchical Clustering

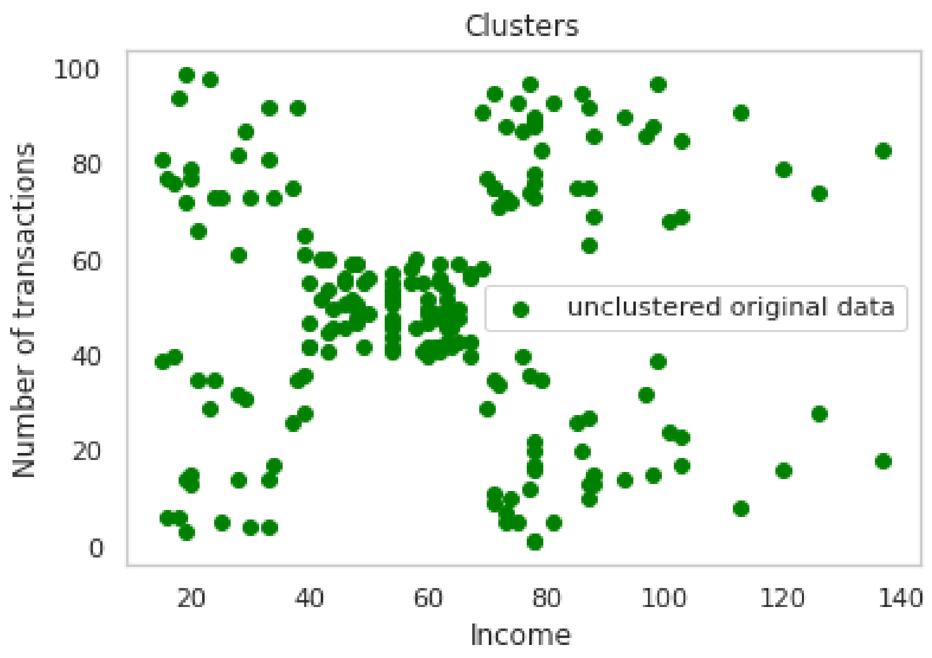
CODE:

```
import numpy as np
import pandas as pd
from scipy.cluster.hierarchy import dendrogram, linkage
from scipy.spatial.distance import cdist
from matplotlib import pyplot as plt
from scipy.spatial import distance
import math
%matplotlib inline
# plt.figure(figsize=(15, 6))
dataset = pd.read_csv('Mall_Customers.csv')
X = dataset.iloc[:, [3, 4]].values
# plt.scatter(dataset.iloc[:,3].values,dataset.iloc[:,4].values)

def hierarchicalScratch(k, linkage):
    clusters = [[i] for i in range(len(X))]
    matrix = [
        [math.sqrt(np.sum(np.square(np.subtract(X[i], X[j])))) for j in range(len(X))]
        for i in range(len(X))
    ]
    while len(clusters) != k:
        a = -1
        b = -1
        min_dist = 0
        if linkage == "ward":
            centers = [[0 for i in range(len(X[0]))] for j in range(len(clusters))]
            for i in range(len(clusters)):
                for point in clusters[i]:
                    for j in range(len(X[point])):
                        centers[i][j] += X[point][j]
                for j in range(len(X[0])):
                    centers[i][j] /= len(clusters[i])
            for i in range(len(clusters)):
                for j in range(i + 1, len(clusters)):
                    dist = (
                    len(clusters[i]) * len(clusters[j]) / (len(clusters[i]) + len(clusters[j]))
                ) * (np.sum(np.square(np.subtract(centers[i], centers[j]))))
                if a == -1 or dist < min_dist:
                    min_dist = dist
                    a = i
                    b = j
    else:
        for i in range(len(clusters)):
            for j in range(i + 1, len(clusters)):
                dist = 0
                if linkage == "Single":
                    dist = 999999999
                for p1 in clusters[i]:
                    for p2 in clusters[j]:
                        if linkage == "Complete":
                            dist = max(dist, matrix[p1][p2])
                        elif linkage == "Single":
                            dist = min(dist, matrix[p1][p2])
                        else:
                            dist += matrix[p1][p2]
                if linkage == "Average":
                    dist /= len(clusters[i]) * len(clusters[j])
                if a == -1 or dist < min_dist:
                    min_dist = dist
                    a = i
                    b = j
    clusters[a].extend(clusters.pop(b))
labels = [-1 for _ in range(len(X))]
for i in range(k):
    for p in clusters[i]:
        labels[p] = i
title = "Hierarchichal "+linkage
```

Name: Meith Navlakha
SAP: 60004190068
DWM Practical

```
plt.figure(figsize=(8, 8))
plt.title(title)
plt.scatter(X[:, 0], X[:, 1], c=labels, cmap="rainbow", marker=".")

hierarchicalScratch(5, "Complete")
hierarchicalScratch(5, "Average")
hierarchicalScratch(5, "Single")
plt.show()
```

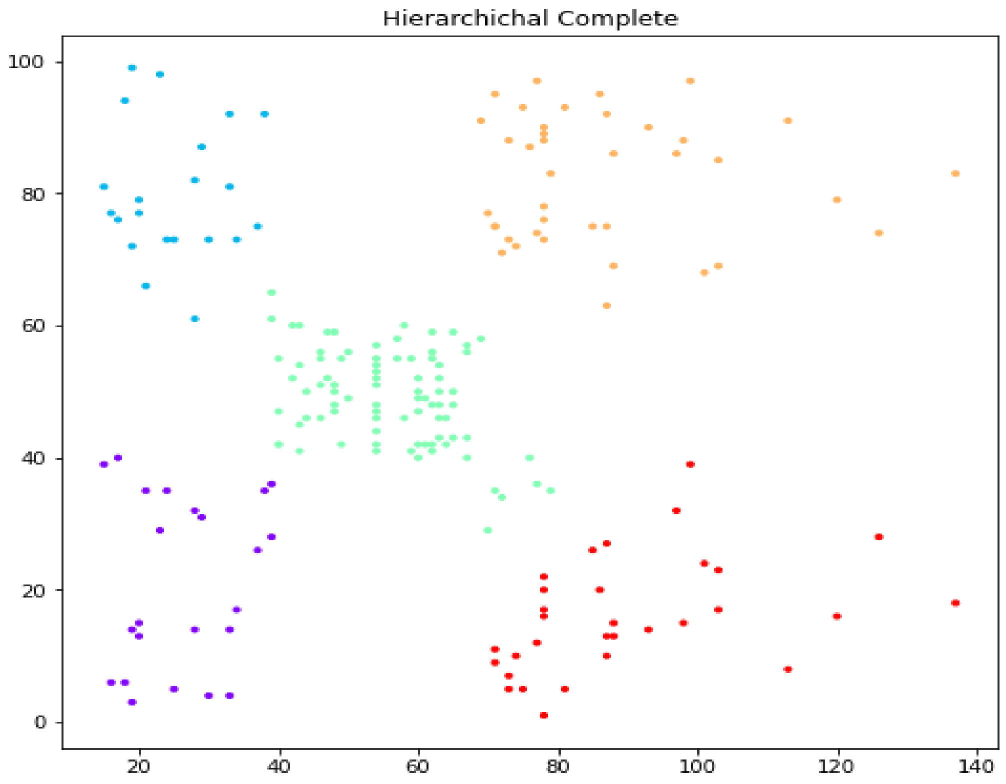
OUTPUT:

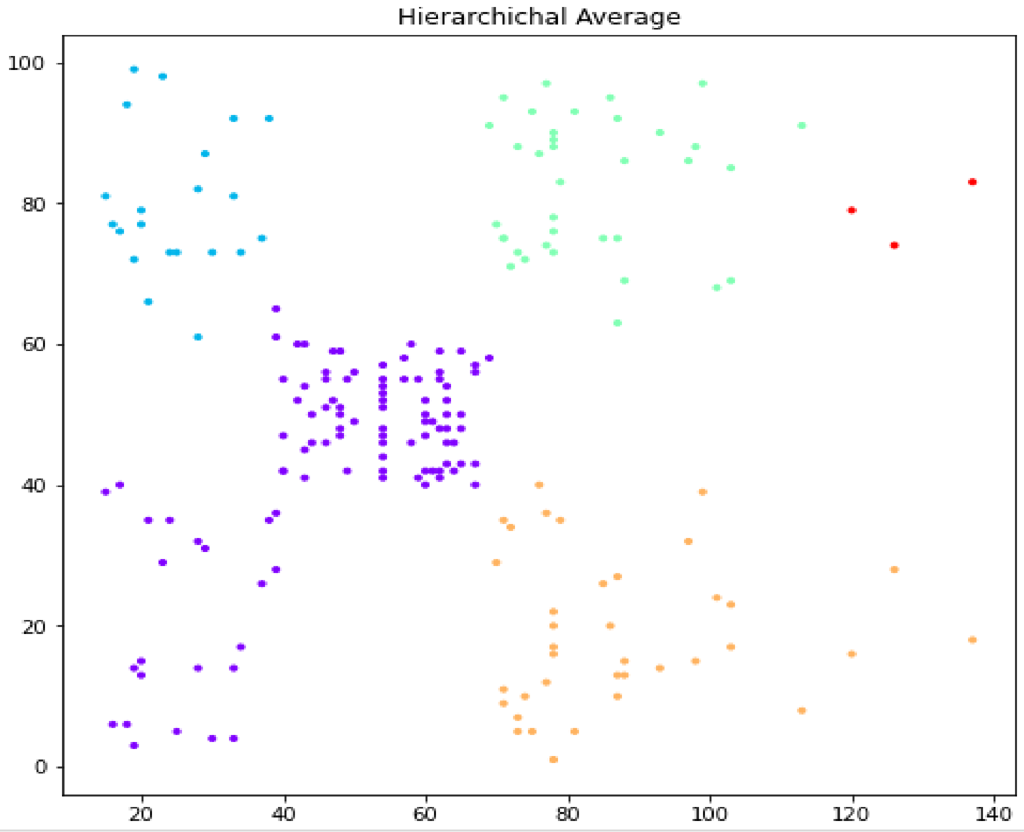


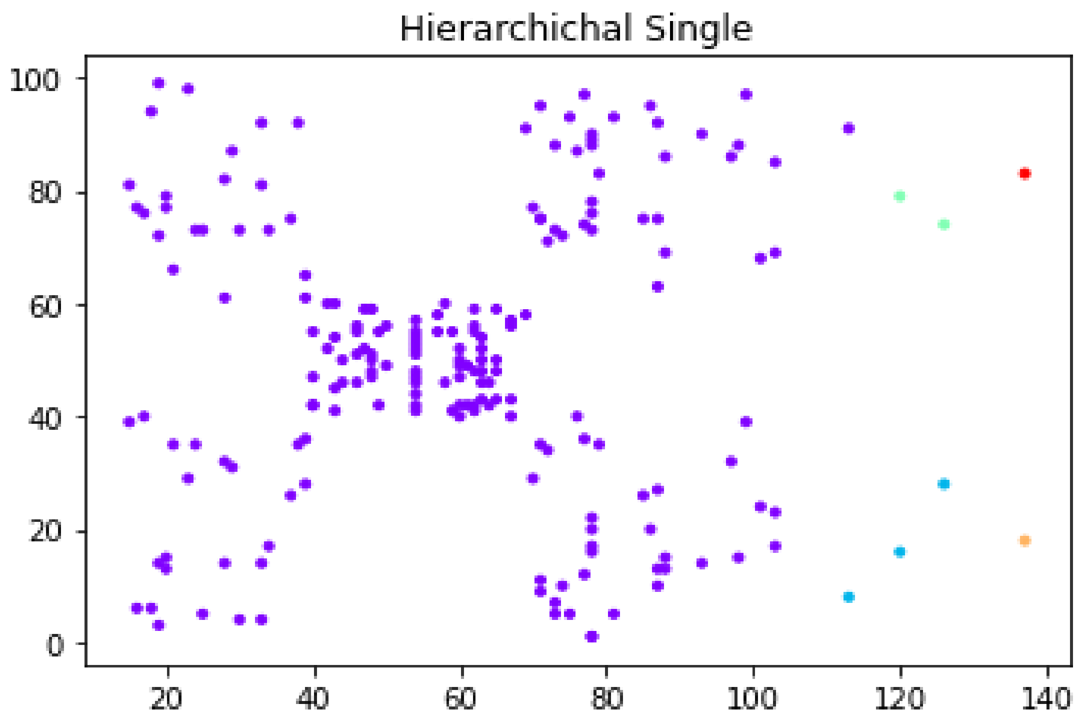
Name: Meith Navlakha

SAP: 60004190068

DWM Practical







Part C: Find optimum no. of cluster using elbow method.

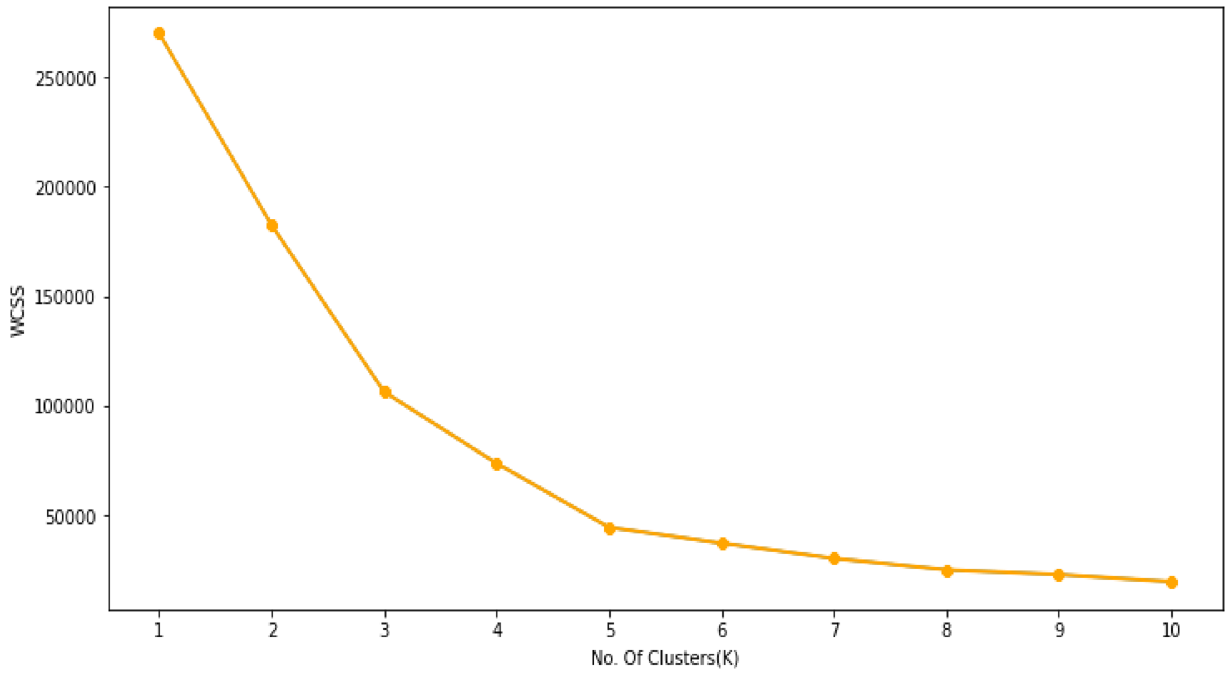
KMeans

Code:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from mpl_toolkits.mplot3d import Axes3D
%matplotlib inline

ds = pd.read_csv('Mall_Customers.csv')
df1=ds[["CustomerID","Genre","Age","Annual Income (k$)","Spending Score (1-100)"]]
X=df1[["Annual Income (k$)","Spending Score (1-100)"]]
from sklearn.cluster import KMeans
wcss=[]
for i in range(1,11):
   km=KMeans(n_clusters=i)
```

```
    km.fit(X)
    wcss.append(km.inertia_)

plt.figure(figsize=(12,6))
plt.plot(range(1,11),wcss)
plt.plot(range(1,11),wcss, linewidth=2, color="orange", marker ="8")
plt.xlabel("No. Of Clusters(K)")
plt.xticks(np.arange(1,11,1))
plt.ylabel("WCSS")
plt.show()
```

Output:



From the Elbow Curve, it is evident that for number of clusters 4, 5 and 6 we see a substantial decrease but the max decrease it at K=5, thus K=5 is taken as the elbow point.

Name: Meith Navlakha
SAP: 60004190068
DWM Practical

CONCLUSION:

In this experiment I implement KMeans Algorithm and Hierarchical clustering for Single, Complete and Average for Mall Customer dataset. In Part A, I used the inbuilt scikit learn for implementing KMeans and Hierarchical clustering and in Part B, I coded the function scratch. Finally, at the end I used the elbow method to find the optimum number of clusters which came out be K=5.