In [28]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
```

In [29]:
```
df=pd.read_csv("Mall_Customers.csv")
df
```

Out[29]:

	CustomerID	Genre	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
...	...	...	...	...	...
195	196	Female	35	120	79
196	197	Female	45	126	28
197	198	Male	32	126	74
198	199	Male	32	137	18
199	200	Male	30	137	83

200 rows × 5 columns

In [30]:
```
df.shape
```

Out[30]: (200, 5)

In [31]:
```
df.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   Genre                   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 [32]:
```
X= df.iloc[:,3:5].values
X
```

```
Out[32]: array([[ 15,  39],
                [ 15,  81],
                [ 16,   6],
                [ 16,  77],
                [ 17,  40],
                [ 17,  76],
                [ 18,   6],
                [ 18,  94],
                [ 19,   3],
                [ 19,  72],
                [ 19,  14],
                [ 19,  99],
                [ 20,  15],
                [ 20,  77],
                [ 20,  13],
                [ 20,  79],
                [ 21,  35],
                [ 21,  66],
                [ 23,  29],
                [ 23,  98],
                [ 24,  35],
                [ 24,  73],
                [ 25,   5],
                [ 25,  73],
                [ 28,  14],
                [ 28,  82],
                [ 28,  32],
                [ 28,  61],
                [ 29,  31],
                [ 29,  87],
                [ 30,   4],
                [ 30,  73],
                [ 33,   4],
                [ 33,  92],
                [ 33,  14],
                [ 33,  81],
                [ 34,  17],
                [ 34,  73],
                [ 37,  26],
                [ 37,  75],
                [ 38,  35],
                [ 38,  92],
                [ 39,  36],
                [ 39,  61],
                [ 39,  28],
                [ 39,  65],
                [ 40,  55],
                [ 40,  47],
                [ 40,  42],
                [ 40,  42],
                [ 42,  52],
```

```
       [ 78,  76],
       [ 78,  16],
       [ 78,  89],
       [ 78,   1],
       [ 78,  78],
       [ 78,   1],
       [ 78,  73],
       [ 79,  35],
       [ 79,  83],
       [ 81,   5],
       [ 81,  93],
       [ 85,  26],
       [ 85,  75],
       [ 86,  20],
       [ 86,  95],
       [ 87,  27],
       [ 87,  63],
       [ 87,  13],
       [ 87,  75],
       [ 87,  10],
       [ 87,  92],
       [ 88,  13],
       [ 88,  86],
       [ 88,  15],
       [ 88,  69],
       [ 93,  14],
       [ 93,  90],
       [ 97,  32],
       [ 97,  86],
       [ 98,  15],
       [ 98,  88],
       [ 99,  39],
       [ 99,  97],
       [101,  24],
       [101,  68],
       [103,  17],
       [103,  85],
       [103,  23],
       [103,  69],
       [113,   8],
       [113,  91],
       [120,  16],
       [120,  79],
       [126,  28],
       [126,  74],
       [137,  18],
       [137,  83]], dtype=int64)
```

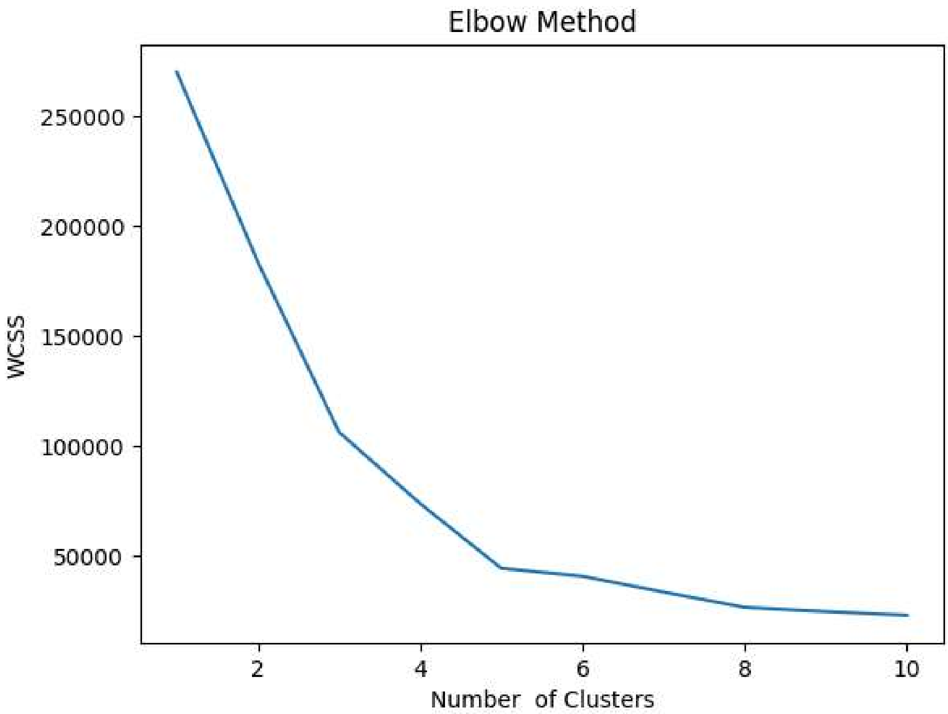
In [33]:

```
WCSS=[] # in your code defines an empty list named WCSS.
```

In [34]:
```
for i in range (1,11):
    kmeans =KMeans(n_clusters=i, init='k-means++',max_iter= 300,
    #Create a KMeans model with i clusters, k-means++ initializat
    #and a fixed random seed for reproducibility.
    kmeans.fit(X)
    #Fit the KMeans model to the data X.
    WCSS.append(kmeans.inertia_)
    #Append the within-cluster sum of squares (WCSS) or "inertia"
    #WCSS list for analysis of model performance.
```

In [35]:
```
plt.plot(range(1,11),WCSS)
#Plots the WCSS values against the number of clusters (1 to 10).
plt.title("Elbow Method")
#Sets the title of the plot to "Elbow Method".
plt.xlabel('Number  of Clusters')
# Labels the x-axis as "Number of Clusters".
plt.ylabel('WCSS')
#Labels the y-axis as "WCSS" (Within-Cluster Sum of Squares).
```

Out[35]: Text(0, 0.5, 'WCSS')



In [36]:
```
kmeans =KMeans(n_clusters=5, init ='k-means++', max_iter=300 ,ran
#Creates a KMeans clustering model with 5 clusters, using the 'k-
#running for a maximum of 300 iterations
#, and a random seed of 42 for reproducibility.
y_kmeans= kmeans.fit_predict(X)
#Fits the KMeans model to the data X and
#returns the cluster labels (predictions) for each data point in
```

In [37]:
```
y_kmeans
```

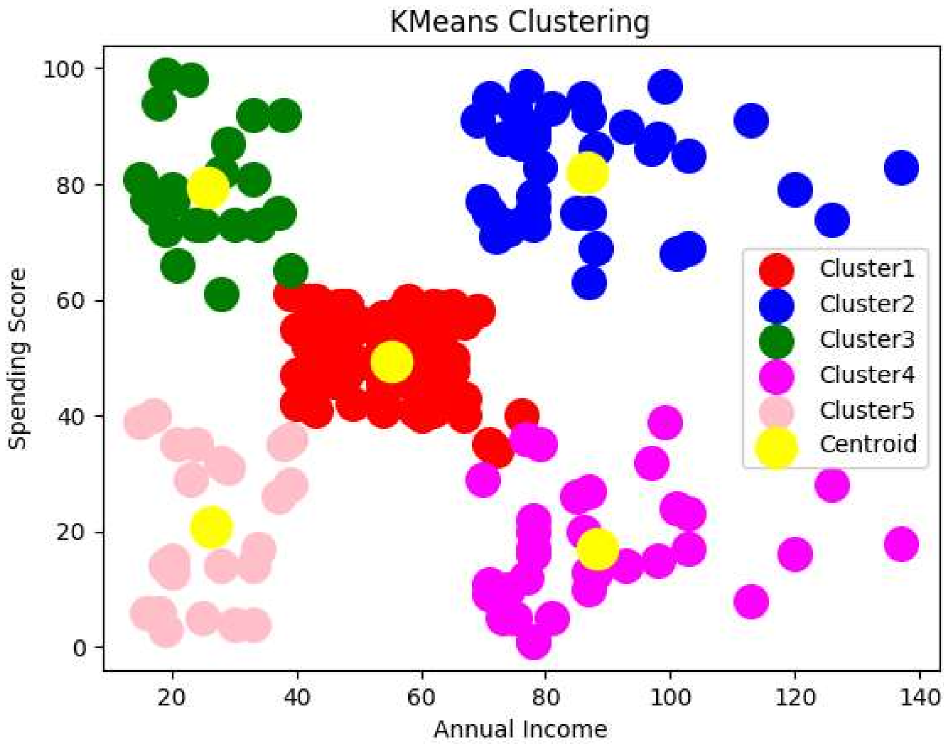
Out[37]:
```
array([4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4,
2, 4, 2,
       4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4,
2, 4, 0,
       4, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 3, 1, 0, 1, 3,
1, 3, 1,
       0, 1, 3, 1, 3, 1, 3, 1, 3, 1, 0, 1, 3, 1, 3, 1, 3, 1, 3,
1, 3, 1,
       3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3,
1, 3, 1,
       3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3,
1, 3, 1,
       3, 1])
```

In [38]:
```
y_kmeans.shape
```

Out[38]:
```
(200,)
```

In [39]:
```
plt.scatter(X[y_kmeans==0,0], X[y_kmeans==0,1], s=200, c='red', l
plt.scatter(X[y_kmeans==1,0], X[y_kmeans==1,1], s=200, c='blue',
plt.scatter(X[y_kmeans==2,0], X[y_kmeans==2,1], s=200, c='green',
plt.scatter(X[y_kmeans==3,0], X[y_kmeans==3,1], s=200, c='magenta
plt.scatter(X[y_kmeans==4,0], X[y_kmeans==4,1], s=200, c='pink',
#Plots data points from Cluster 1 with red color and size 200.
plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_
plt.title("KMeans Clustering")
#Sets the plot's title to "KMeans Clustering".
plt.xlabel("Annual Income")
#Labels the x-axis as "Annual Income".
plt.ylabel('Spending Score')
#Labels the y-axis as "Spending Score".
plt.legend()
#Adds a legend to differentiate the clusters.
```

```
plt.show()
#Displays the plot.
```



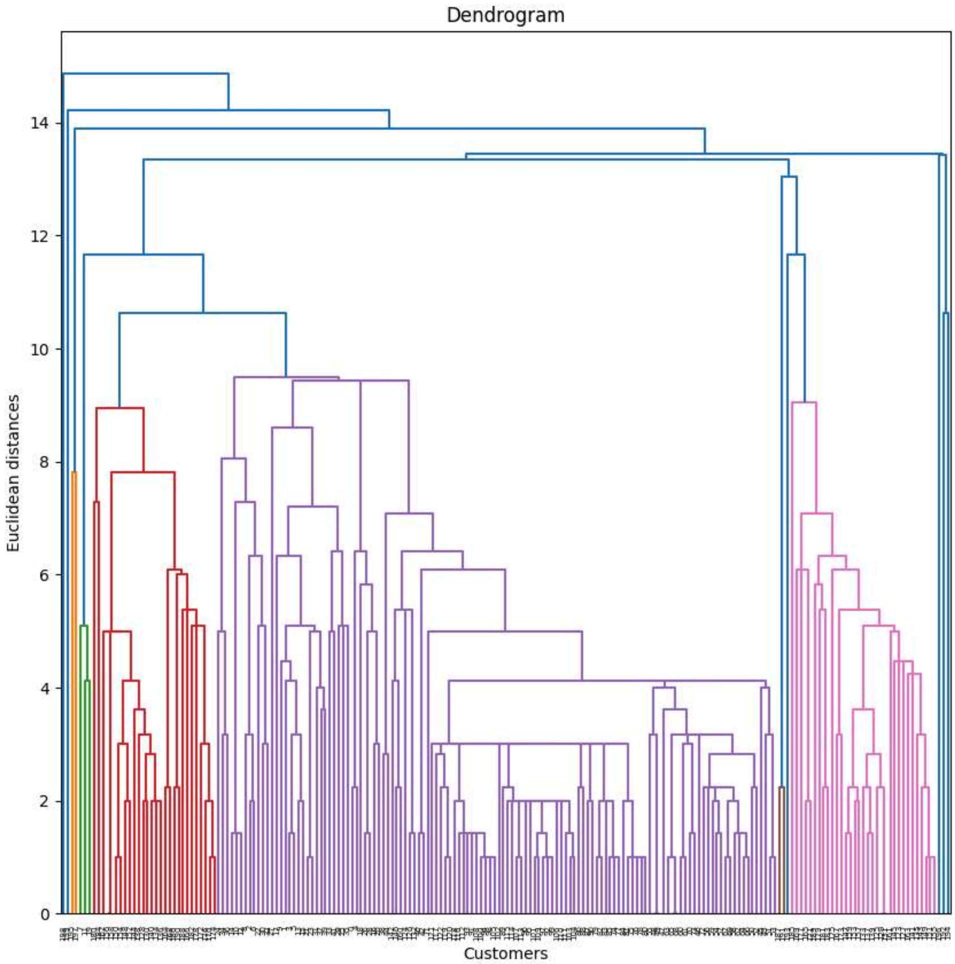
In [40]:
```
import matplotlib.cm as cm
from sklearn.metrics import silhouette_samples, silhouette_score
```

In [41]:
```
range_n_clusters =[2,3,4,5,6]
for n_clusters in range_n_clusters:
    #Initialize the clusterer with n_clusters value and a random
    #seed of 10 for reproducibility
    clusterer =KMeans(n_clusters, random_state=10)
    cluster_labels=clusterer.fit_predict(X)
    #The silhouette score gives the average value forall the samp
    #This gives a perspective into the density andseparation of t
    silhouette_avg=silhouette_score(X , cluster_labels)
    print(" For n_clusters= ", n_clusters, "The average silhouett
```

```
 For n_clusters=  2 The average silhouette score is : 0.3774913479
961559
 For n_clusters=  3 The average silhouette score is : 0.4676135815
8775435
 For n_clusters=  4 The average silhouette score is : 0.4937945814
354117
 For n_clusters=  5 The average silhouette score is : 0.5539319974
44648
 For n_clusters=  6 The average silhouette score is : 0.5379675585
622219
```

In [42]:

```
import  scipy.cluster.hierarchy as sch
plt.figure(figsize=(10,10))
#Creates a new figure for the plot with a size of 10x10 inches.
dendrogram= sch.dendrogram(sch.linkage(X, method='single'))
#Generates a dendrogram by computing hierarchical clustering usin
plt.title('Dendrogram')
#Adds the title 'Dendrogram' to the plot.
plt.xlabel('Customers')
#Labels the x-axis as 'Customers'.
plt.ylabel("Euclidean distances")
#Labels the y-axis as 'Euclidean distances'.
plt.show()
#Displays the dendrogram plot.
```

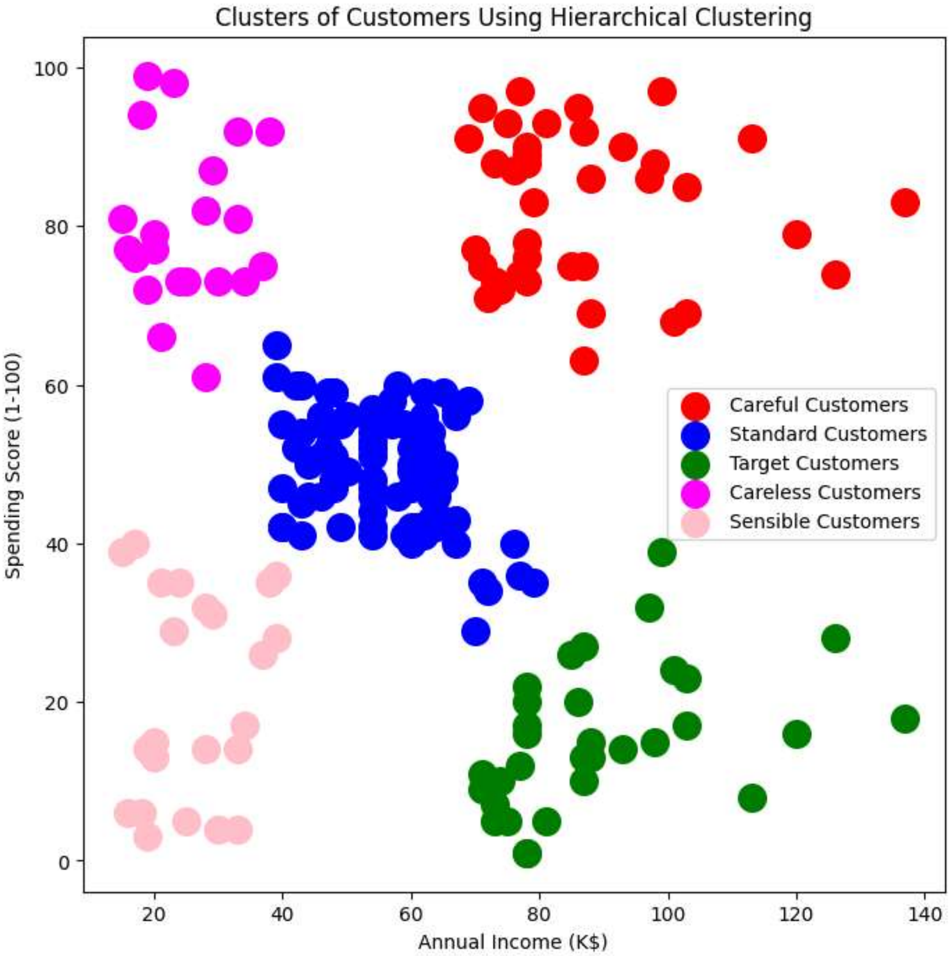


In [45]:
```
from sklearn.cluster import AgglomerativeClustering

hc = AgglomerativeClustering(n_clusters=5, linkage='complete')
#Initializes an Agglomerative Clustering model with 5 clusters us
y_hc = hc.fit_predict(X)
#Fits the clustering model to the data X and predicts the cluster
```

In [46]:
```
# Visualizing the clusters
plt.figure(figsize=(8, 8))
plt.scatter(X[y_hc == 0, 0], X[y_hc == 0, 1], s=200, c='red', lab
plt.scatter(X[y_hc == 1, 0], X[y_hc == 1, 1], s=200, c='blue', la
plt.scatter(X[y_hc == 2, 0], X[y_hc == 2, 1], s=200, c='green', l
plt.scatter(X[y_hc == 3, 0], X[y_hc == 3, 1], s=200, c='magenta',
plt.scatter(X[y_hc == 4, 0], X[y_hc == 4, 1], s=200, c='pink', la
#: Plots a scatter plot for the cluster where y_hc equals 4,
#with pink color, larger marker size, and labeled as "Sensible Cu
plt.title("Clusters of Customers Using Hierarchical Clustering")
#Sets the plot's title to "Clusters of Customers Using Hierarchic
plt.xlabel('Annual Income (K$)')
#Labels the x-axis as 'Annual Income (K$)'.
plt.ylabel('Spending Score (1-100)')
```

```
# Labels the y-axis as 'Spending Score (1-100)'.
plt.legend()
#Displays a legend showing the label for the plotted cluster.
plt.show()
#Displays the scatter plot.
```



In [ ]: