Mall Customer Segmentation KMeans | Heirarchical

In [1]:
```
import pandas as pd
import numpy as np

import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

import scipy.stats as stats

from sklearn.cluster import KMeans

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import silhouette_samples, silhouette_score


import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
df = pd.read_csv(r"https://raw.githubusercontent.com/NelakurthiSudheer/Mall-Customers-Segmentation/main/Dataset/Mall_Customers.cs
```

In [3]:
```
df
```

Out[3]:

	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
...	...	...	...	...	...
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 [4]:
```
df.head()
```

Out[4]:

	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 [5]:
```
df.columns
```

Out[5]:
```
Index(['CustomerID', 'Gender', 'Age', 'Annual Income (k$)',
       'Spending Score (1-100)'],
      dtype='object')
```

In [6]:
```
#data information....
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   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 [7]:
```
# null values.....
df.isnull().sum()
```

Out[7]:
```
CustomerID                0
Gender                    0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

In [8]:
```
# statistical analysis.....
```

In [9]:
```
df.describe().T
```

Out[9]:

	count	mean	std	min	25%	50%	75%	max
CustomerID	200.0	100.50	57.879185	1.0	50.75	100.5	150.25	200.0
Age	200.0	38.85	13.969007	18.0	28.75	36.0	49.00	70.0
Annual Income (k$)	200.0	60.56	26.264721	15.0	41.50	61.5	78.00	137.0
Spending Score (1-100)	200.0	50.20	25.823522	1.0	34.75	50.0	73.00	99.0

In [10]:
```
df.shape
```

Out[10]:
```
(200, 5)
```

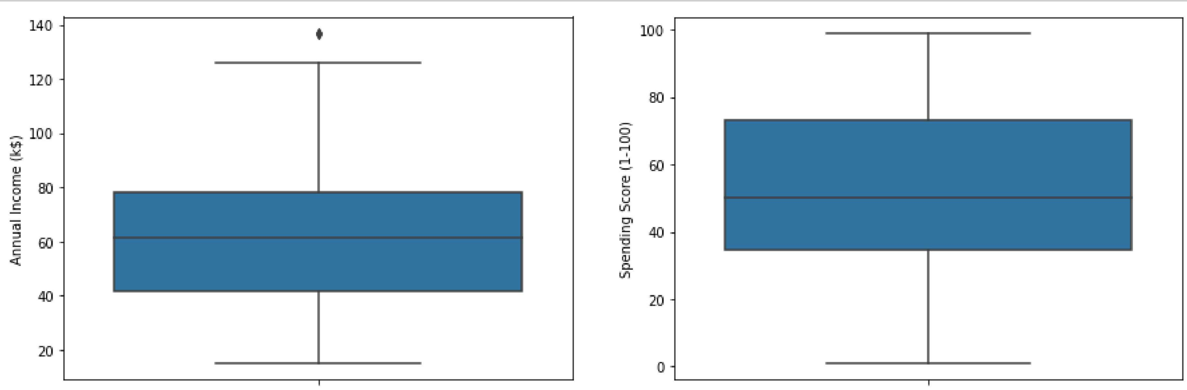
checking outliers.....

In [11]:
```
plt.figure(figsize=(15,5))

plt.subplot(1,2,1)
sns.boxplot(data=df, y="Annual Income (k$)")

plt.subplot(1,2,2)
sns.boxplot(data=df, y="Spending Score (1-100)")

plt.show()
```



segregation the data.....

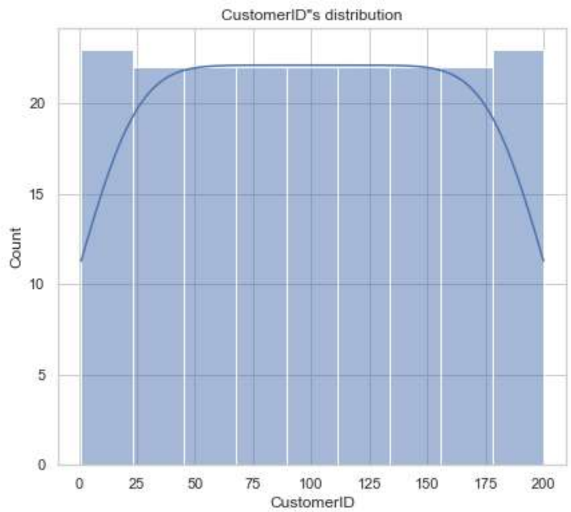
In [12]:
```
#numeric_col
numerical_col = [i for i in df.columns if df[i].dtype!='O']
numerical_col
```

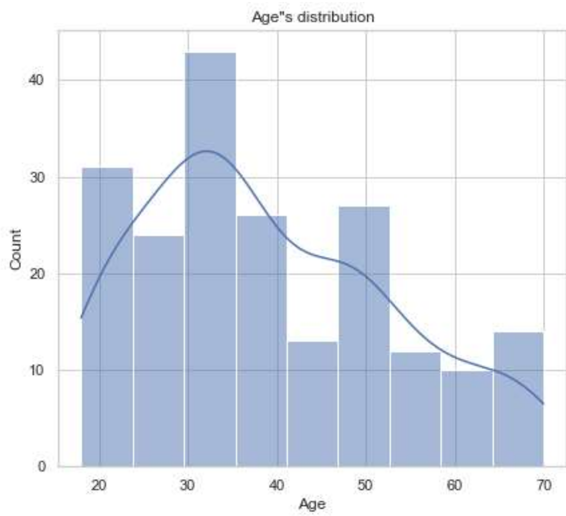
Out[12]: ['CustomerID', 'Age', 'Annual Income (k$)', 'Spending Score (1-100)']

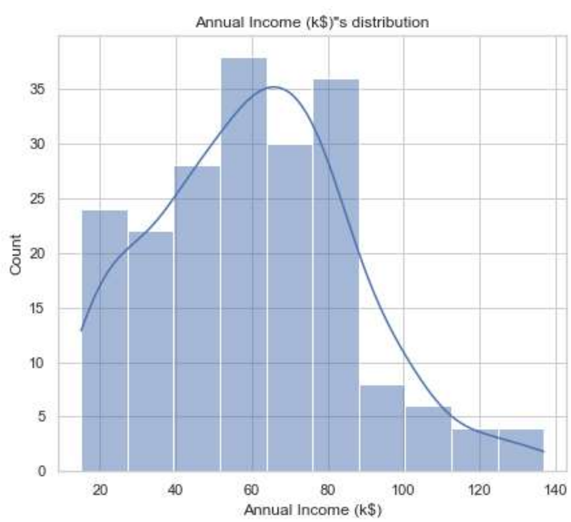
In [13]:
```
#categorical col
cat_col = [i for i in df.columns if df[i].dtype=='O']
cat_col
```

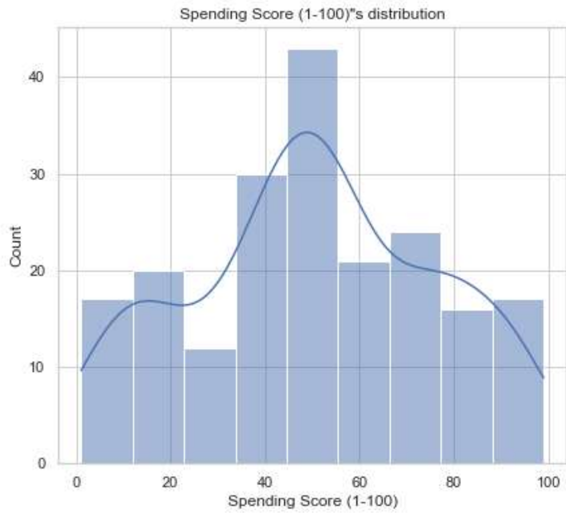
Out[13]: ['Gender']

In [14]:
```
# for numerical plot...
for x in numerical_col:
    sns.set(style = 'whitegrid')
    plt.figure(figsize=(15,6))
    plt.subplot(121)
    sns.histplot(df,x=x,kde=True)
    plt.title(f'{x}"s distribution')
    plt.show()
```







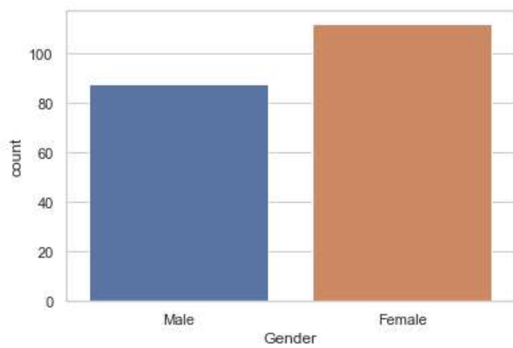


In [15]:
```
df['Gender'].value_counts()
```

Out[15]:
```
Female    112
Male       88
Name: Gender, dtype: int64
```

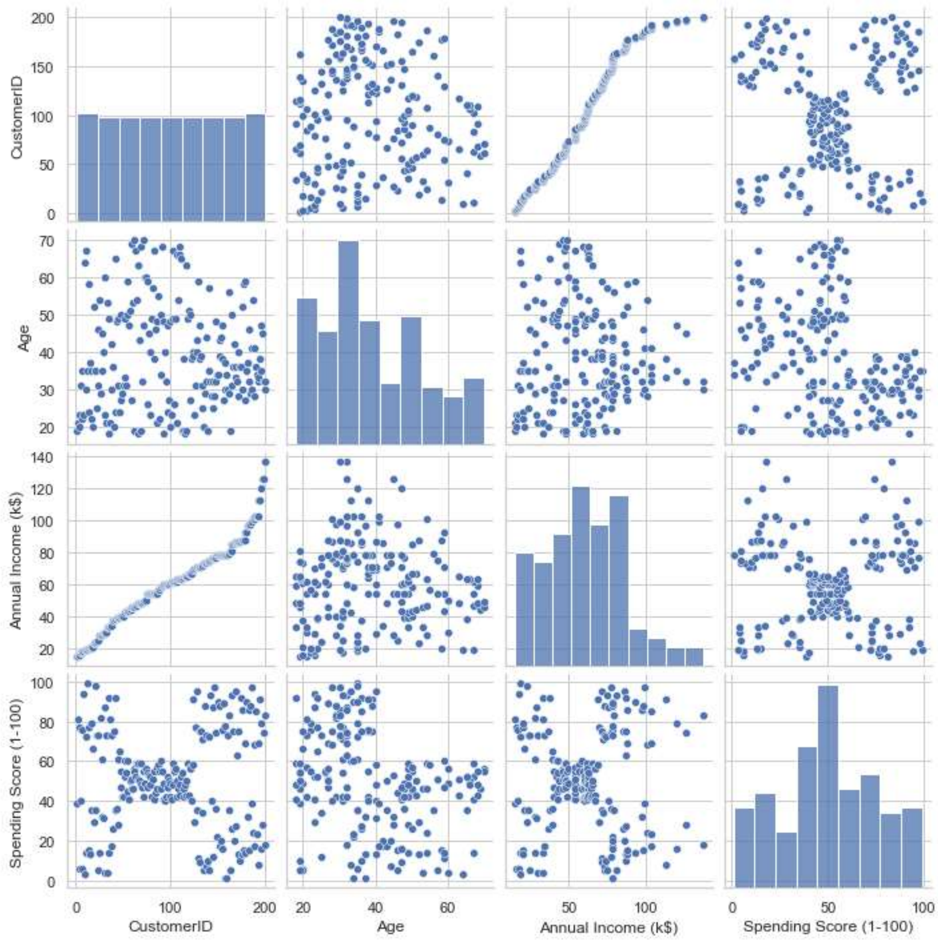
In [16]:
```
#for categorical plot....
sns.countplot(x='Gender',data=df)
```

Out[16]: <AxesSubplot:xlabel='Gender', ylabel='count'>



In [17]:
```
sns.pairplot(df)
```

Out[17]: <seaborn.axisgrid.PairGrid at 0x18e915af100>

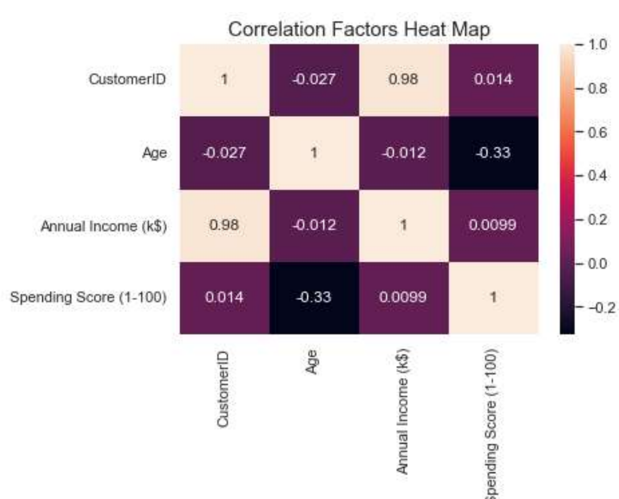


checking correlation......

In [18]:
```
#plt.figure(figsize=(10, 7))
#matrix = np.triu(df.corr())
#sns.heatmap(df.corr(), annot=True,linewidth=.8, mask=matrix, cmap="rocket");
```

In [19]:
```
## Correlation coeffecients heatmap
sns.heatmap(df.corr(), annot=True).set_title('Correlation Factors Heat Map', size='15')
```

Out[19]: Text(0.5, 1.0, 'Correlation Factors Heat Map')



In [20]:
```
# check multicol...
```

In [21]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

In [22]:
```
vif_data = pd.DataFrame()
vif_data['vif'] = [variance_inflation_factor(df[numerical_col].values,i) for i in range(len(numerical_col))]
vif_data['features'] = df[numerical_col].columns
vif_data
```

Out[22]:

	vif	features
0	74.814679	CustomerID
1	5.652358	Age
2	106.712605	Annual Income (k$)
3	3.668468	Spending Score (1-100)

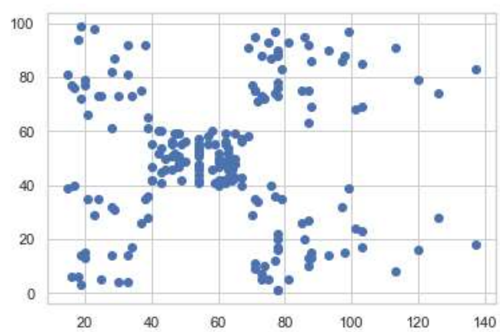
we will use Annual Income and Spending Score for clustering customers. Let's look how our plot is seen without clustering.

Clustering the data for 'Annual Income (k), 'Spending Score (1-100)'

we will use Annual Income and Spending Score for clustering customers. Let's look how our plot is seen without clustering.

In [23]:
```
plt.scatter(df['Annual Income (k$)'] , df['Spending Score (1-100)'])
```

Out[23]: <matplotlib.collections.PathCollection at 0x18e8c0bae80>



K-Means :

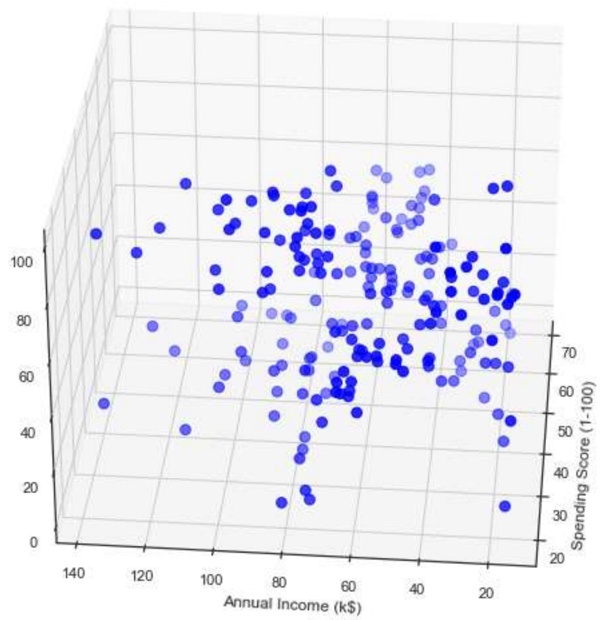
K-means clustering is a type of unsupervised learning which is used when you have unlabeled data. By using this algorithm you

will try to find groups in the data. "k" value represent number of groups.

In [24]:
```
from sklearn.cluster import KMeans
```

In [25]:
```
from mpl_toolkits.mplot3d import Axes3D

sns.set_style("white")
fig = plt.figure(figsize=(20,10))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(df.Age, df["Annual Income (k$)"], df["Spending Score (1-100)"], c='blue', s=60)
ax.view_init(30, 185)
plt.xlabel("Spending Score (1-100)")
plt.ylabel("Annual Income (k$)")
plt.show()
```



Clustering Analysis

In [26]:
```
x = df.iloc[:, [3, 4]].values

# let's check the shape of x
print(x.shape)
```

```
(200, 2)
```

In [ ]:

In [28]:
```
scaler = StandardScaler()

x_scaled = scaler.fit_transform(x)
x_scaled
```

```
       [-0.78476346, -0.12422899],
       [-0.78476346, -0.3183368 ],
       [-0.78476346, -0.3183368 ],
       [-0.70842461,  0.06987881],
       [-0.70842461,  0.38045129],
       [-0.67025518,  0.14752193],
       [-0.67025518,  0.38045129],
       [-0.67025518, -0.20187212],
       [-0.67025518, -0.35715836],
       [-0.63208575, -0.00776431],
       [-0.63208575, -0.16305055],
       [-0.55574689,  0.03105725],
       [-0.55574689, -0.16305055],
       [-0.55574689,  0.22516505],
       [-0.55574689,  0.18634349],
       [-0.51757746,  0.06987881],
       [-0.51757746,  0.34162973],
       [-0.47940803,  0.03105725],
       [-0.47940803,  0.34162973],
       [-0.47940803, -0.00776431],
```

Elbow-Method using WCSS(Within Cluster Sum of Squares):

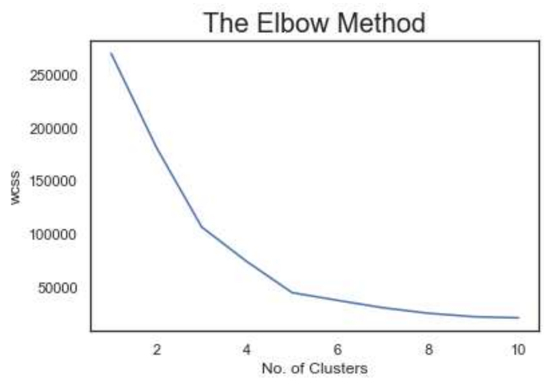
Now we will try to find what "k" value we should use. We will find out it with "elbow method".

let's say from 1 to 11) and for each value, we are calculating the sum of squared distances from each point to its assigned center.

In [29]:
```
from sklearn.cluster import KMeans

wcss = []
for i in range(1, 11):
    km = KMeans(n_clusters = i, init = 'k-means++', max_iter = 300, n_init = 10, random_state = 0)
    km.fit(x)
    wcss.append(km.inertia_)

plt.plot(range(1, 11), wcss)
plt.title('The Elbow Method', fontsize = 20)
plt.xlabel('No. of Clusters')
plt.ylabel('wcss')
plt.show()
```

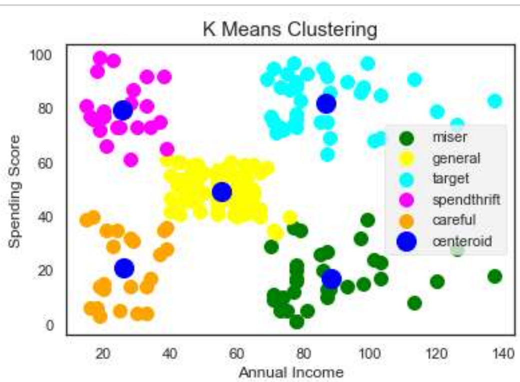


Inertia: It is defined as the mean squared distance between each instance and its closest centroid. Logically, as per the definition lower the inertia better the model.

In [30]:
```
km = KMeans(n_clusters = 5, init = 'k-means++', max_iter = 300, n_init = 10, random_state = 0)
y_means = km.fit_predict(x)

plt.scatter(x[y_means == 0, 0], x[y_means == 0, 1], s = 100, c = 'green', label = 'miser')
plt.scatter(x[y_means == 1, 0], x[y_means == 1, 1], s = 100, c = 'yellow', label = 'general')
plt.scatter(x[y_means == 2, 0], x[y_means == 2, 1], s = 100, c = 'cyan', label = 'target')
plt.scatter(x[y_means == 3, 0], x[y_means == 3, 1], s = 100, c = 'magenta', label = 'spendthrift')
plt.scatter(x[y_means == 4, 0], x[y_means == 4, 1], s = 100, c = 'orange', label = 'careful')
plt.scatter(km.cluster_centers_[:,0], km.cluster_centers_[:, 1], s = 200, c = 'blue' , label = 'centeroid')

plt.style.use('fivethirtyeight')
plt.title('K Means Clustering', fontsize = 15)
plt.xlabel('Annual Income')
plt.ylabel('Spending Score')
plt.legend()
plt.show()
```



This Clustering Analysis gives us a very clear insight about the different segments of the customers in the Mall. There are clearly Five segments of Customers namely Miser, General, Target, Spendthrift, Careful based on their Annual Income and Spending Score which are reportedly the best factors/attributes to determine the segments of a customer in a Mall.

Silhouette Coefficient Method:

The silhouette coefficient of a data measures how well data are assigned to its own cluster and how far they are from other clusters.

.A silhouette close to 1 means the data points are in an appropriate cluster .A silhouette coefficient close to −1 implies out data is in the wrong cluster.

Silhouette Coefficient = (x-y)/ max(x,y)

In [31]:
```
KMean= KMeans(n_clusters=5)
KMean.fit(x_scaled)
label=KMean.predict(x_scaled)

print("Silhouette Score(n=5):",silhouette_score(x_scaled, label))
```

```
Silhouette Score(n=5): 0.5546571631111091
```

In [32]:
```
print(KMean.cluster_centers_)
```

```
[[-0.20091257 -0.02645617]
 [-1.30751869 -1.13696536]
 [-1.32954532  1.13217788]
 [ 0.99158305  1.23950275]
 [ 1.05500302 -1.28443907]]
```

In [33]:
```
print(KMean.labels_)
```

```
[1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1
 2 1 2 1 2 1 0 1 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 0 0 0 0 3 4 3 0 3 4 3 4 3 0 3 4 3 4 3 4 3 4 3 0 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 3 4 3 4
 3 4 3 4 3 4 3 4 3 4 3 4 3]
```

In [34]:
```
kmeans = KMeans(n_clusters = 5, init = 'k-means++', random_state = 42)
```

In [35]:
```
y_kmeans = kmeans.fit_predict(x)
```

In [36]:
```

y_kmeans
```

Out[36]:
```
array([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, 0,
       2, 3, 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, 4, 1, 4, 0, 4, 1, 4, 1, 4,
       0, 4, 1, 4, 1, 4, 1, 4, 1, 4, 0, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4,
       1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4,
       1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4,
       1, 4])
```

In [37]:
```
#Add cluster results columns to the dataset dataframe

df["cluster"] = KMean.labels_
df.head()
```

Out[37]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)	cluster
0	1	Male	19	15	39	1
1	2	Male	21	15	81	2
2	3	Female	20	16	6	1
3	4	Female	23	16	77	2
4	5	Female	31	17	40	1

Visualising the clusters

In [ ]:

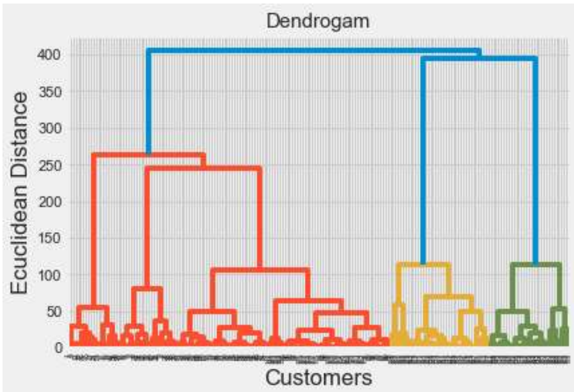
Hierarchical clustering

Hierarchical clustering, also known as hierarchical cluster analysis, is an algorithm that groups similar objects into groups called 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

In [38]:
```
# Using Dendrograms to find the no. of Optimal Clusters...
```

In [39]:
```
import scipy.cluster.hierarchy as sch

dendrogram = sch.dendrogram(sch.linkage(x, method = 'ward'))
plt.title('Dendrogam', fontsize = 15)
plt.xlabel('Customers')
plt.ylabel('Ecuclidean Distance')
plt.show()
```



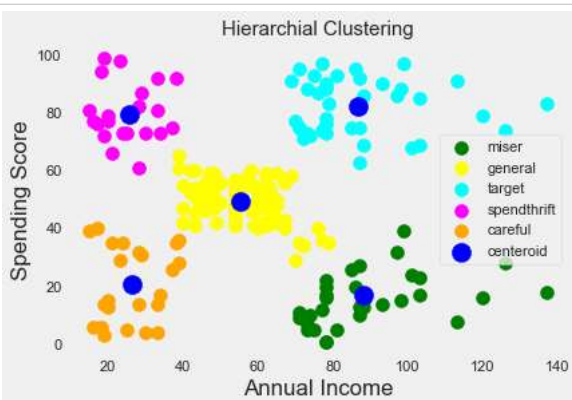
visualizing the Clusters of Hierarchial Clustering

In [40]:
```
from sklearn.cluster import AgglomerativeClustering

hc = AgglomerativeClustering(n_clusters = 5, affinity = 'euclidean', linkage = 'ward')
y_hc = hc.fit_predict(x)

plt.scatter(x[y_hc == 0, 0], x[y_hc == 0, 1], s = 100, c = 'green', label = 'miser')
plt.scatter(x[y_hc == 1, 0], x[y_hc == 1, 1], s = 100, c = 'yellow', label = 'general')
plt.scatter(x[y_hc == 2, 0], x[y_hc == 2, 1], s = 100, c = 'cyan', label = 'target')
plt.scatter(x[y_hc == 3, 0], x[y_hc == 3, 1], s = 100, c = 'magenta', label = 'spendthrift')
plt.scatter(x[y_hc == 4, 0], x[y_hc == 4, 1], s = 100, c = 'orange', label = 'careful')
plt.scatter(km.cluster_centers_[:,0], km.cluster_centers_[:, 1], s = 200, c = 'blue' , label = 'centeroid')

plt.style.use('fivethirtyeight')
plt.title('Hierarchial Clustering', fontsize = 15)
plt.xlabel('Annual Income')
plt.ylabel('Spending Score')
plt.legend()
plt.grid()
plt.show()
```



In [ ]: