# lab-11

In [1]:
```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.decomposition import PCA
import seaborn as sns
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import numpy as np
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
import warnings
warnings.filterwarnings('ignore')
import scipy.cluster.hierarchy as shc
from sklearn.cluster import MeanShift,AgglomerativeClustering
import statistics
```

## Step1. [Understand Data]

- Using Pandas, import “Mall_Customers.csv” file and print properties such as head, shape, columns, dtype, info and value_counts.
- For example: customers_data = pd.read_csv("Mall_Customers.csv")

In [2]:
```
import pandas as pd
data = pd.read_csv("Mall_Customers.csv")
data.head()
```

Out[2]:

| | 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 |

In [3]:
```
data.shape
```

Out[3]:
```
(200, 5)
```

In [4]:
```
data.columns
```

Out[4]:
```
Index(['CustomerID', 'Genre', 'Age', 'Annual Income (k$)',
       'Spending Score (1-100)'],
      dtype='object')
```

In [5]:
```
data.dtypes
```

Out[5]:
```
CustomerID                 int64
Genre                     object
Age                        int64
Annual Income (k$)         int64
Spending Score (1-100)     int64
dtype: object
```

In [6]:
```
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   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 [7]:
```
data.value_counts()
```

Out[7]:
```
CustomerID  Genre   Age  Annual Income (k$)  Spending Score (1-100)
1           Male    19   15                  39                        1
138         Male    32   73                  73                        1
128         Male    40   71                  95                        1
129         Male    59   71                  11                        1
130         Male    38   71                  75                        1
                                                                      ..
70          Female  32   48                  47                        1
71          Male    70   49                  55                        1
72          Female  47   49                  42                        1
73          Female  60   50                  49                        1
200         Male    30   137                 83                        1
Length: 200, dtype: int64
```

## Step2. [Label encode gender]

- Genre (ie., gender) is a string, so label encode into binary

In [8]:
```
data.drop(['CustomerID'],axis=1,inplace=True)
```

In [9]:
```
label_encoder = LabelEncoder()
data["Genre"] = label_encoder.fit_transform(data["Genre"])
data
```

Out[9]:

| | Genre | Age | Annual Income (k$) | Spending Score (1-100) |
|---|---|---|---|---|
| **0** | 1 | 19 | 15 | 39 |
| **1** | 1 | 21 | 15 | 81 |
| **2** | 0 | 20 | 16 | 6 |
| **3** | 0 | 23 | 16 | 77 |
| **4** | 0 | 31 | 17 | 40 |
| **...** | ... | ... | ... | ... |
| **195** | 0 | 35 | 120 | 79 |
| **196** | 0 | 45 | 126 | 28 |
| **197** | 1 | 32 | 126 | 74 |
| **198** | 1 | 32 | 137 | 18 |
| **199** | 1 | 30 | 137 | 83 |

200 rows × 4 columns

In [10]:
```
data.Genre.value_counts()
```

Out[10]:
```
0    112
1     88
Name: Genre, dtype: int64
```

In [11]:
```
df = data.copy()
```

## Step3. [ Check for variance]

- Use describe() on your data frame and check for variance. If variance is high for float columns, you need to normalize. Otherwise, ignore

In [12]:
```
data.describe(include='all')
```

Out[12]:

| | Genre | Age | Annual Income (k$) | Spending Score (1-100) |
|---|---|---|---|---|
| **count** | 200.000000 | 200.000000 | 200.000000 | 200.000000 |
| **mean** | 0.440000 | 38.850000 | 60.560000 | 50.200000 |
| **std** | 0.497633 | 13.969007 | 26.264721 | 25.823522 |
| **min** | 0.000000 | 18.000000 | 15.000000 | 1.000000 |
| **25%** | 0.000000 | 28.750000 | 41.500000 | 34.750000 |
| **50%** | 0.000000 | 36.000000 | 61.500000 | 50.000000 |
| **75%** | 1.000000 | 49.000000 | 78.000000 | 73.000000 |
| **max** | 1.000000 | 70.000000 | 137.000000 | 99.000000 |

In [13]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Genre                   200 non-null    int32
 1   Age                     200 non-null    int64
 2   Annual Income (k$)      200 non-null    int64
 3   Spending Score (1-100)  200 non-null    int64
dtypes: int32(1), int64(3)
memory usage: 5.6 KB
```

In [14]:
```
data.var()
```

Out[14]:
```
Genre                       0.247638
Age                       195.133166
Annual Income (k$)        689.835578
Spending Score (1-100)    666.854271
dtype: float64
```

In [15]:
```
data.corr()
```

Out[15]:

| | Genre | Age | Annual Income (k$) | Spending Score (1-100) |
|---|---|---|---|---|
| **Genre** | 1.000000 | 0.060867 | 0.056410 | -0.058109 |
| **Age** | 0.060867 | 1.000000 | -0.012398 | -0.327227 |
| **Annual Income (k$)** | 0.056410 | -0.012398 | 1.000000 | 0.009903 |
| **Spending Score (1-100)** | -0.058109 | -0.327227 | 0.009903 | 1.000000 |

## Step4. [Check skewness]

- Check if float columns are skewed. Use skew() on your data frame. If skew value is greater than 0.75, then you can perform log transformation on those skew columns.

In [16]:
```
data.skew()
```

Out[16]:
```
Genre                     0.243578
Age                       0.485569
Annual Income (k$)        0.321843
Spending Score (1-100)   -0.047220
dtype: float64
```

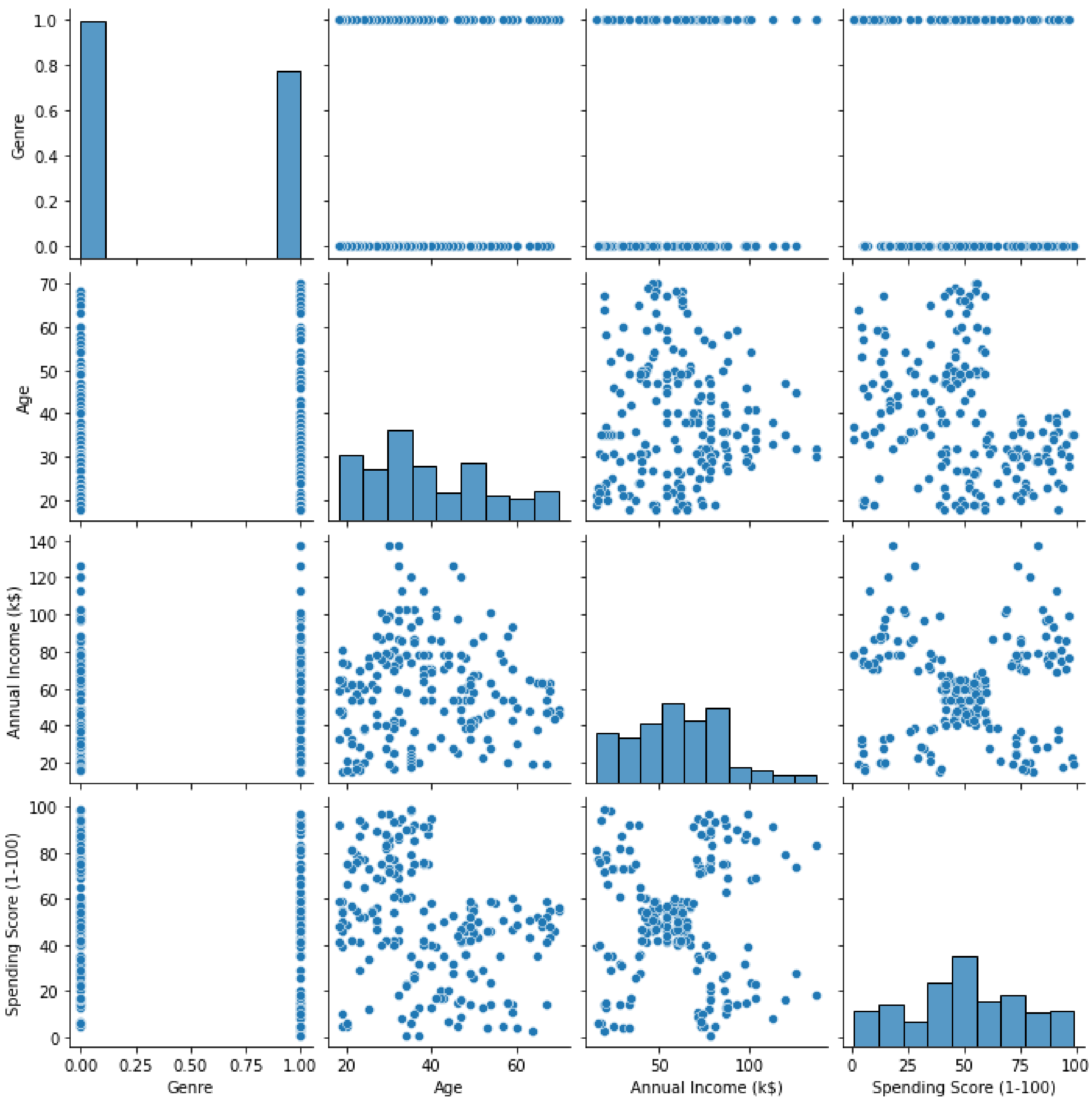
In [17]:
```
data.sort_values(by=["Genre","Age","Annual Income (k$)","Spending Score (1-100)"]
```

## Step5. [Pair plot]

- Draw pair plot and observe correlations

In [18]:
```
sns.pairplot(data)
```

Out[18]: <seaborn.axisgrid.PairGrid at 0x1f9ac65cd30>



## Step6. [Build KMeans]

- Create and fit KMeans (n_clusters variable can be set with any value)
- Print label_ and cluster_centers_ values

In [19]:
```
model = KMeans(n_clusters=5)
```

In [20]:
```
model.fit(data)
```

Out[20]:
```
KMeans(n_clusters=5)
```

In [21]:
```
model.labels_
```

Out[21]:
```
array([0, 0, 0, 0, 0, 0, 3, 0, 0, 3, 0, 3, 1, 1, 0, 0, 1, 0, 0, 0, 3, 3,
       1, 0, 1, 0, 0, 0, 3, 1, 0, 1, 1, 0, 1, 1, 2, 0, 2, 2, 2, 2, 0, 1,
       3, 2, 1, 2, 4, 3, 2, 1, 1, 0, 1, 2, 2, 2, 2, 4, 2, 2, 4, 2, 2, 2,
       2, 0, 0, 0, 1, 4, 0, 4, 4, 4, 1, 0, 4, 1, 1, 0, 0, 0, 3, 0, 0, 4,
       0, 0, 0, 0, 3, 0, 0, 0, 3, 1, 1, 0, 1, 0, 3, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 3, 3, 1, 1, 0, 0, 0, 3, 1, 0, 3, 1, 1, 0, 3, 1, 1, 0,
       0, 3, 2, 2, 0, 0, 1, 2, 1, 2, 3, 2, 2, 4, 3, 4, 3, 1, 1, 0, 2, 2,
       2, 2, 2, 0, 0, 2, 2, 2, 0, 0, 0, 3, 2, 2, 2, 4, 4, 2, 2, 2, 0, 0,
       2, 0, 0, 0, 1, 4, 4, 1, 0, 0, 4, 4, 4, 0, 4, 0, 0, 4, 4, 4, 3, 0,
       0, 0])
```

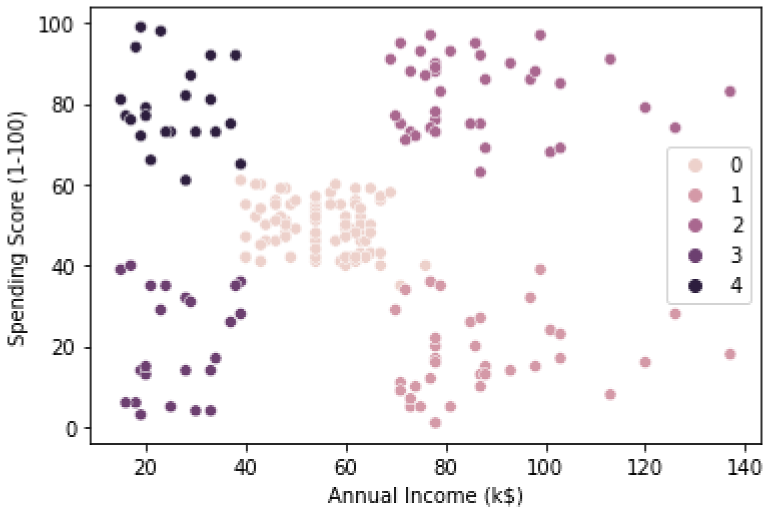
In [22]:
```
model.cluster_centers_
```

Out[22]:
```
array([[ 0.4125    , 42.9375    , 55.0875    , 49.7125    ],
       [ 0.52777778, 40.66666667, 87.75      , 17.58333333],
       [ 0.46153846, 32.69230769, 86.53846154, 82.12820513],
       [ 0.39130435, 45.2173913 , 26.30434783, 20.91304348],
       [ 0.40909091, 25.27272727, 25.72727273, 79.36363636]])
```

## Step7. [Scatter plot]

- Draw scatter plot between any two features with hue as “labels_”.

In [23]:
```
sns.scatterplot(data['Annual Income (k$)'],data['Spending Score (1-100)'],hue=moc
plt.show()
```




## Step8. [Cluster Analysis].

**Now, predict cluster labels for the same data. For example,**

```
kmeans2 = KMeans(n_clusters = 5, init='k-means++')
kmeans2.fit(customers_data)
pred = kmeans2.predict(customers_data)
```

**Now, add a new column for pred in a new dataframe, such as**

```
frame = pd.DataFrame(customers_data)
frame['cluster'] = pred
```

This will create a new column to frame. That means, you have added a cluster prediction column, whose values say the cluster number to which the row belongs to. That is, that customer belongs to that cluster number.

**Now, group customers based on cluster number. Remember, here we have 5 clusters from 0 to 4.**

**For each cluster group, print the following details.**

```
Average age: 45.21739130434783
Average annual income: 26.304347826086957
Deviation of the mean for annual income: 7.893811054517766
No of customers ie shape: (23, 5)
From those customers we have 9 male and 14 female
```

In [24]:
```
data.head()
```

Out[24]:

| | Genre | Age | Annual Income (k$) | Spending Score (1-100) |
|---|---|---|---|---|
| **70** | 1 | 70 | 49 | 55 |
| **60** | 1 | 70 | 46 | 56 |
| **57** | 1 | 69 | 44 | 46 |
| **108** | 1 | 68 | 63 | 43 |
| **102** | 1 | 67 | 62 | 59 |

In [25]:
```
x = data.iloc[:,[2,3]].values
```

In [26]:
```
kmeans2 = KMeans(n_clusters = 5, init='k-means++')
kmeans2.fit(x)
pred = kmeans2.predict(x)
```

In [27]:
```
frame = pd.DataFrame(x)
frame['cluster'] = pred
data['cluster'] = pred
```

In [28]:
```
frame.cluster.value_counts()
```

Out[28]:
```
2    81
0    39
3    35
1    23
4    22
Name: cluster, dtype: int64
```

In [29]:
```
frame
```

Out[29]:

| | 0 | 1 | cluster |
|---|---|---|---|
| **0** | 49 | 55 | 2 |
| **1** | 46 | 56 | 2 |
| **2** | 44 | 46 | 2 |
| **3** | 63 | 43 | 2 |
| **4** | 62 | 59 | 2 |
| **...** | ... | ... | ... |
| **195** | 37 | 75 | 4 |
| **196** | 16 | 6 | 1 |
| **197** | 65 | 50 | 2 |
| **198** | 63 | 54 | 2 |
| **199** | 65 | 48 | 2 |

200 rows × 3 columns

In [30]:
```
d_frameC0 = data[data['cluster'] == 0]
d_frameC1 = data[data['cluster'] == 1]
d_frameC2 = data[data['cluster'] == 2]
d_frameC3 = data[data['cluster'] == 3]
d_frameC4 = data[data['cluster'] == 4]
```

In [31]:
```
print("Average age for cluster 0 :",d_frameC0['Age'].mean())
print("Average Annual Income for cluster 0 :",d_frameC0['Annual Income (k$)'].mea
print("Deviation of the mean for annual income :",statistics.stdev(d_frameC0['Ann
print("No.of customers i.e shape of cluster 0 :",d_frameC0.shape)
print("From those customers we have",d_frameC0.Genre.value_counts()[1],"male and'
```

```
Average age for cluster 0 : 32.69230769230769
Average Annual Income for cluster 0 : 86.53846153846153
Deviation of the mean for annual income : 16.312484972924967
No.of customers i.e shape of cluster 0 : (39, 5)
From those customers we have 18 male and 21 female
```

In [32]:
```
print("Average age for cluster 1 :",d_frameC1.Age.mean())
print("Average Annual Income for cluster 1 :",d_frameC1['Annual Income (k$)'].mea
print("Deviation of the mean for annual income :",statistics.stdev(d_frameC1['Ann
print("No.of customers i.e shape of cluster 1 :",d_frameC1.shape)
print("From those customers we have",d_frameC1.Genre.value_counts()[1],"male and'
```

```
Average age for cluster 1 : 45.21739130434783
Average Annual Income for cluster 1 : 26.304347826086957
Deviation of the mean for annual income : 7.893811054517766
No.of customers i.e shape of cluster 1 : (23, 5)
From those customers we have 9 male and 14 female
```

In [33]:
```
print("Average age for cluster 2 :",d_frameC2.Age.mean())
print("Average Annual Income for cluster 2 :",d_frameC2['Annual Income (k$)'].mea
print("Deviation of the mean for annual income :",statistics.stdev(d_frameC2['Ann
print("No.of customers i.e shape of cluster 2 :",d_frameC2.shape)
print("From those customers we have",d_frameC2.Genre.value_counts()[1],"male and'
```

```
Average age for cluster 2 : 42.71604938271605
Average Annual Income for cluster 2 : 55.2962962962963
Deviation of the mean for annual income : 8.988109429190942
No.of customers i.e shape of cluster 2 : (81, 5)
From those customers we have 33 male and 48 female
```

In [34]:
```
print("Average age for cluster 3 :",d_frameC3.Age.mean())
print("Average Annual Income for cluster 3 :",d_frameC3['Annual Income (k$)'].mea
print("Deviation of the mean for annual income :",statistics.stdev(d_frameC3['Ann
print("No.of customers i.e shape of cluster 3 :",d_frameC3.shape)
print("From those customers we have",d_frameC3.Genre.value_counts()[1],"male and'
```

```
Average age for cluster 3 : 41.114285714285714
Average Annual Income for cluster 3 : 88.2
Deviation of the mean for annual income : 16.399067405334545
No.of customers i.e shape of cluster 3 : (35, 5)
From those customers we have 19 male and 16 female
```

In [35]:
```
print("Average age for cluster 4 :",d_frameC4.Age.mean())
print("Average Annual Income for cluster 4 :",d_frameC4['Annual Income (k$)'].mea
print("Deviation of the mean for annual income :",statistics.stdev(d_frameC4['Ann
print("No.of customers i.e shape of cluster 4 :",d_frameC4.shape)
print("From those customers we have",d_frameC4.Genre.value_counts()[1],"male and'
```

```
Average age for cluster 4 : 25.272727272727273
Average Annual Income for cluster 4 : 25.727272727272727
Deviation of the mean for annual income : 7.566730552584204
No.of customers i.e shape of cluster 4 : (22, 5)
From those customers we have 9 male and 13 female
```

## Step9. [Find the best number of clusters]

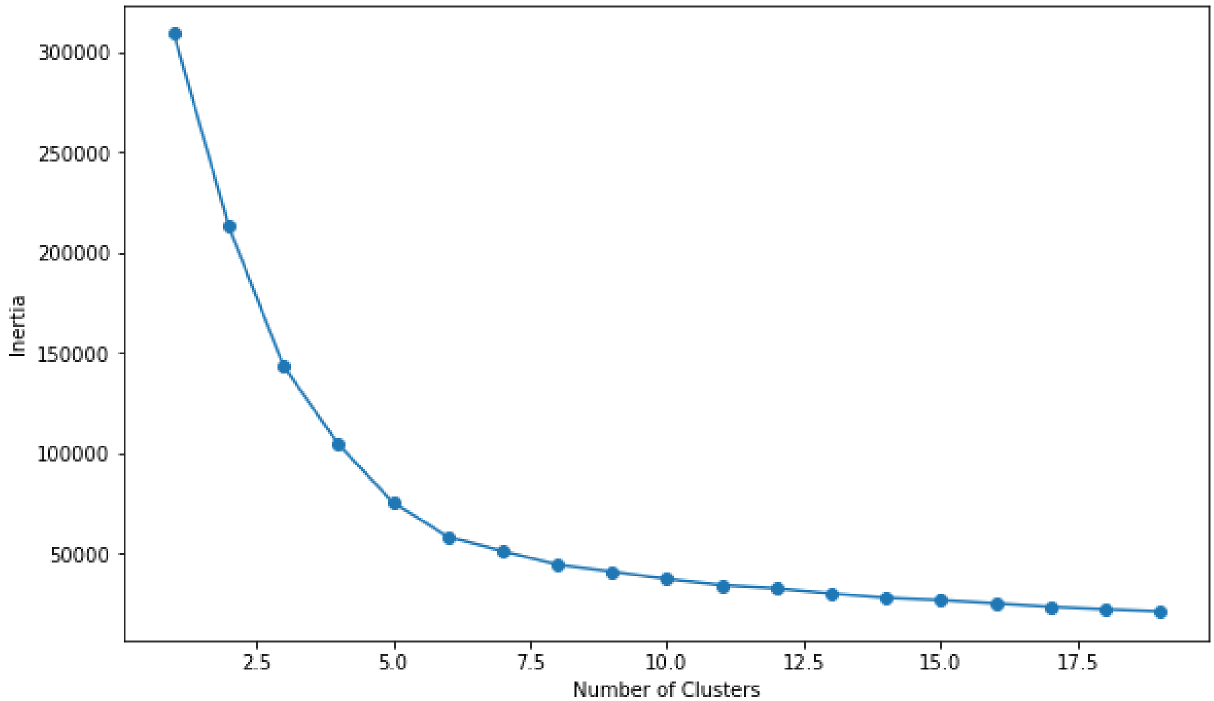
#### Compute inertia value as shown below

```
SSE = []
for clust in range(1,20):
   km = KMeans(n_clusters=clust, init="k-means++")
   km = km.fit(customers_data)
   SSE.append(km.inertia_)
```

Plot a line chart between cluster number and its inertia value. You will get graph as below. Can you identify the best number of clusters from this graph?

In [36]:
```
SSE = []
for cluster in range(1,20):
    km = KMeans(n_clusters=cluster, init="k-means++")
    km.fit(data)
    SSE.append(km.inertia_)
```

In [37]:
```
plt.figure(figsize=(10,6))
plt.plot(np.arange(1,20),SSE,"o-")
plt.xlabel("Number of Clusters")
plt.ylabel("Inertia")
plt.show()
```




## Step10. [Reduce Dimensions using PCA]

- Reduce 4 dimensions into 2 dimensions using PCA
- Create KMeans model, fit on the reduced dataset
- Print cluster_centers_ and labels_

In [38]:
```
y = data
```

In [39]:
```
pca = PCA(n_components=2)
principalComponents = pca.fit_transform(y)
PCA_components = pd.DataFrame(principalComponents)
```

In [40]:
```
PCA_components
```

Out[40]:

| | 0 | 1 |
|---|---|---|
| **0** | -8.937770 | -7.994355 |
| **1** | -9.925562 | -10.990670 |
| **2** | -18.762148 | -6.975124 |
| **3** | -9.703074 | 9.949852 |
| **4** | 2.444558 | -0.205293 |
| **...** | ... | ... |
| **195** | 9.052637 | -35.780641 |
| **196** | -57.422825 | -12.963479 |
| **197** | 6.206044 | 1.085767 |
| **198** | 8.162067 | -2.831962 |
| **199** | 4.825819 | 2.106376 |

200 rows × 2 columns

In [41]:
```
model1 = KMeans(n_clusters=5)
```

In [42]:
```
model1.fit(PCA_components)
```

Out[42]:
```
KMeans(n_clusters=5)
```

In [43]:
```
model1.cluster_centers_
```

Out[43]:
```
array([[ -4.4197535 ,  -3.08886972],
       [ 41.58214146,   1.76309289],
       [-10.12791659,  42.35498015],
       [  4.82223183, -46.69462455],
       [-44.3923333 ,  -9.92438654]])
```

In [44]:
```
model1.labels_
```

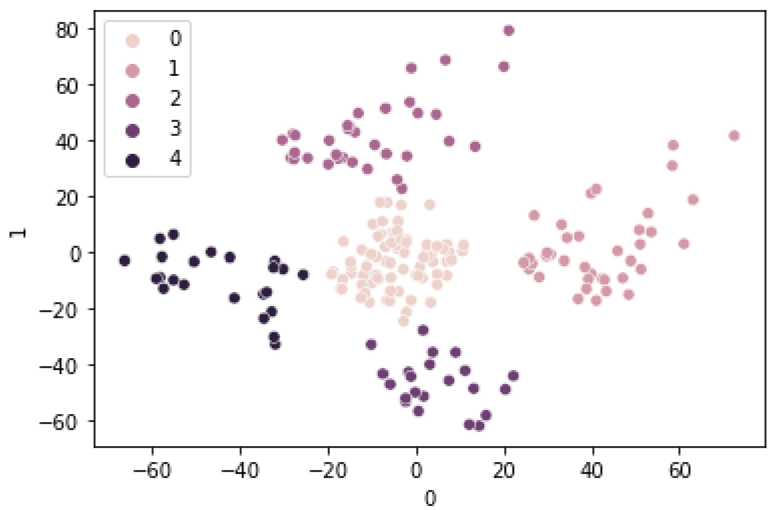
Out[44]:
```
array([0, 0, 0, 0, 0, 0, 4, 0, 0, 4, 0, 4, 2, 2, 0, 0, 2, 0, 0, 0, 4, 4,
       2, 0, 2, 0, 0, 0, 4, 2, 0, 2, 2, 0, 2, 2, 1, 0, 1, 1, 1, 1, 0, 2,
       4, 1, 2, 1, 3, 4, 1, 2, 2, 0, 2, 1, 1, 1, 1, 3, 1, 1, 3, 1, 1, 1,
       1, 0, 0, 0, 2, 3, 0, 3, 3, 3, 2, 0, 3, 2, 2, 0, 0, 0, 4, 0, 0, 3,
       0, 0, 0, 0, 4, 0, 0, 0, 4, 2, 2, 0, 2, 0, 4, 2, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 4, 4, 2, 2, 0, 0, 0, 4, 2, 0, 4, 2, 2, 0, 4, 2, 2, 0,
       0, 4, 1, 1, 0, 0, 2, 1, 2, 1, 4, 1, 1, 3, 4, 3, 4, 2, 2, 0, 1, 1,
       1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 4, 1, 1, 1, 3, 3, 1, 1, 1, 0, 0,
       1, 0, 0, 0, 0, 3, 3, 0, 0, 0, 3, 3, 3, 0, 3, 0, 0, 3, 3, 3, 4, 0,
       0, 0])
```

## Step11. [Scatter plot]

- Draw a scatter plot between the 2 reduced dimensions, with hue as label_
- Your scatter plot may look like below

In [45]:
```
sns.scatterplot(PCA_components[0],PCA_components[1],hue=model1.labels_)
```

Out[45]:
```
<AxesSubplot:xlabel='0', ylabel='1'>
```




## Step12. [ MeanShift clustering]

- Create MeanShift clustering model and fit on the reduced data of PCA and visualize clusters on the reduced data, as shown below.

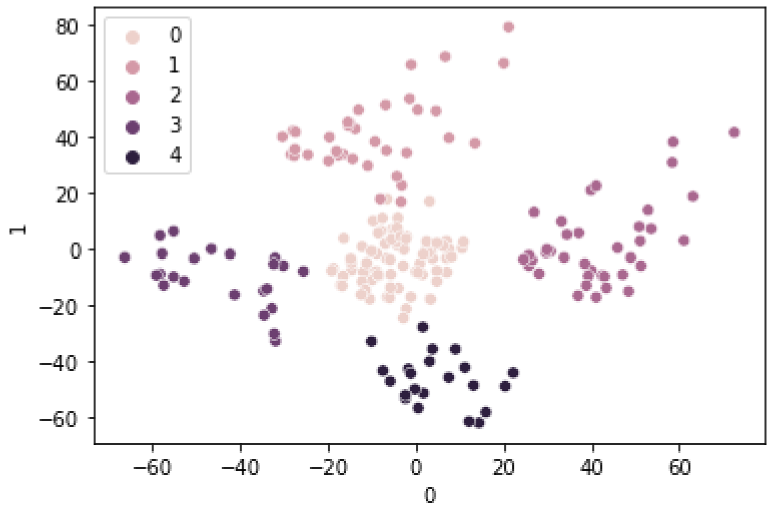
In [46]:
```
model2 = MeanShift(bandwidth=25)
model2.fit(PCA_components)
```

Out[46]:
```
MeanShift(bandwidth=25)
```

In [47]:
```
cluster_centers = model2.cluster_centers_
```

In [48]:
```
sns.scatterplot(PCA_components[0],PCA_components[1],hue=model.labels_)
```

Out[48]: <AxesSubplot:xlabel='0', ylabel='1'>




### Step13. [Predict hierarchical clusters using AgglomerativeClustering]

- Create 5 clusters, using AgglomerativeClustering class. Your dendrogram will look like as below.

In [49]:
```
model3 = AgglomerativeClustering(n_clusters=5,linkage='ward',compute_full_tree=Tı
model3.fit(df)
```

Out[49]: AgglomerativeClustering(compute_full_tree=True, n_clusters=5)

In [50]:
```
model3.labels_
```

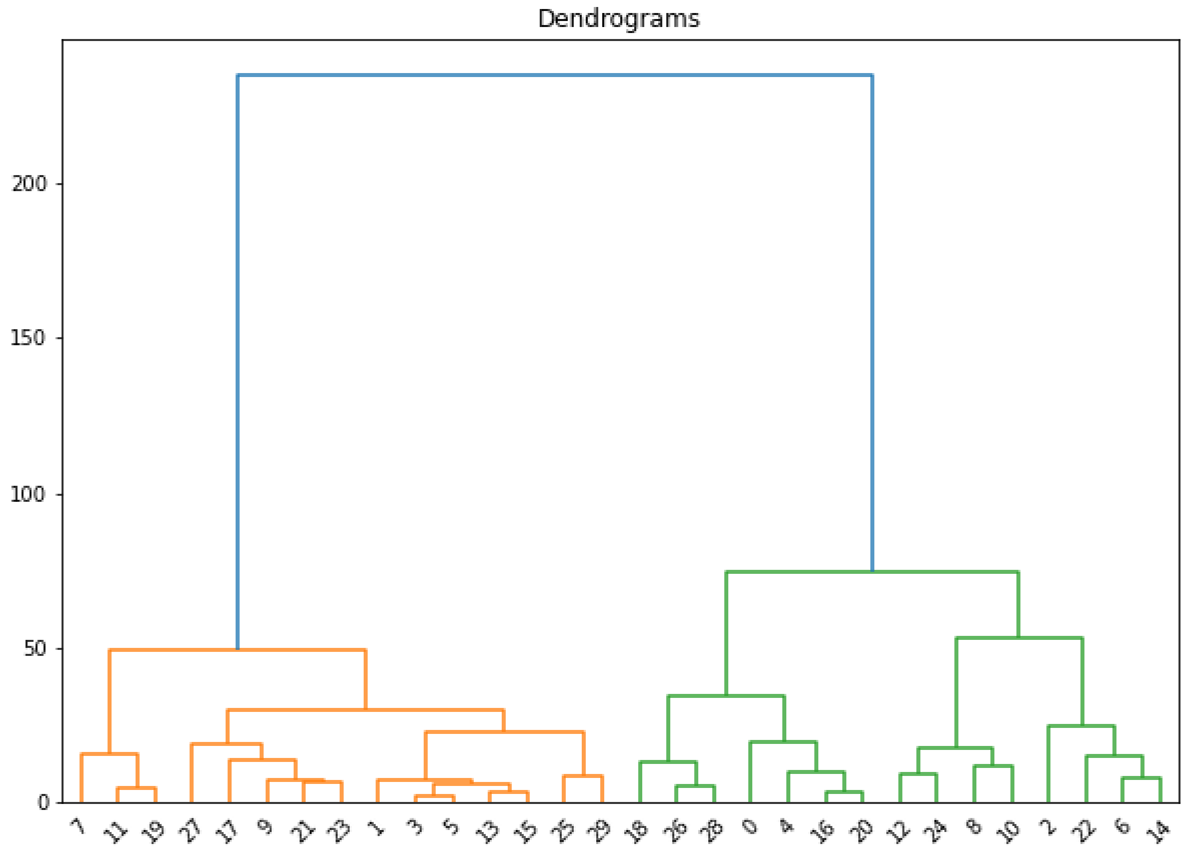
Out[50]:
```
array([4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3,
       4, 3, 4, 3, 4, 0, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 0,
       4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       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, 2, 0, 2, 1, 2, 1, 2, 1, 2,
       0, 2, 1, 2, 1, 2, 1, 2, 1, 2, 0, 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, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2,
       1, 2], dtype=int64)
```

In [51]:
```
frame = df.copy()
frame['cluster'] = model3.labels_
```

In [52]:
```
frame.value_counts()
```

Out[52]:
```
Genre  Age  Annual Income (k$)  Spending Score (1-100)  cluster
0      18   65                  48                      0          1
1      29   28                  82                      3          1
       24   60                  52                      0          1
       25   24                  73                      3          1
            77                  12                      1          1
                                                                  ..
0      41   99                  39                      1          1
            103                 17                      1          1
       42   34                  17                      4          1
       43   48                  50                      0          1
1      70   49                  55                      0          1
Length: 200, dtype: int64
```

In [55]:
```
plt.figure(figsize=(10,7))
plt.title("Dendrograms")
dend = shc.dendrogram(shc.linkage(frame[:30], method='ward'))
```




## Step14. [Visualize scatter plot with hue as agglomerativeclustering

## labels_]

- Visualize agglomerative clusters using the predicted label. Select any two features for X and Y with hue as labels_. Your scatter plot will look like below

In [54]:
```
sns.scatterplot(frame['Annual Income (k$)'],frame['Spending Score (1-100)'],hue=r
```

Out[54]: <AxesSubplot:xlabel='Annual Income (k$)', ylabel='Spending Score (1-100)'>