Labsheet 11: Shopping Mall Customer Segmentation using Clustering

Name: Ezhilarasan C

Roll No:225229151

Step 1 : Understand data

In [15]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
ds = pd.read_csv('Mall_Customers.csv')
```

In [3]:
```
ds.head()
```

Out[3]:

	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 [4]:
```
ds.shape
```

Out[4]:
```
(200, 5)
```

In [5]:
```
ds.size
```

Out[5]:
```
1000
```

In [6]:
```
ds.columns
```

Out[6]:
```
Index(['CustomerID', 'Genre', 'Age', 'Annual Income (k$)',
       'Spending Score (1-100)'],
      dtype='object')
```

In [7]:
```
ds.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 [8]:
```
ds.value_counts
```

Out[8]:
```
<bound method DataFrame.value_counts of      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 x 5 columns]>
```

Step 2: Label Encode gender

In [10]:
```
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()
ds['Genre']= label_encoder.fit_transform(ds['Genre'])
ds['Genre'].unique()
```

Out[10]:
```
array([1, 0])
```

Step 3: Check the variance

In [14]:
```
ds.describe()
```

Out[14]:

	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
count	200.000000	200.000000	200.000000	200.000000	200.000000
mean	100.500000	0.440000	38.850000	60.560000	50.200000
std	57.879185	0.497633	13.969007	26.264721	25.823522
min	1.000000	0.000000	18.000000	15.000000	1.000000
25%	50.750000	0.000000	28.750000	41.500000	34.750000
50%	100.500000	0.000000	36.000000	61.500000	50.000000
75%	150.250000	1.000000	49.000000	78.000000	73.000000
max	200.000000	1.000000	70.000000	137.000000	99.000000

In [16]:
```
ds.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    int64
 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(5)
memory usage: 7.9 KB
```

In [17]:
```
ds.corr()
```

Out[17]:

	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
CustomerID	1.000000	0.057400	-0.026763	0.977548	0.013835
Genre	0.057400	1.000000	0.060867	0.056410	-0.058109
Age	-0.026763	0.060867	1.000000	-0.012398	-0.327227
Annual Income (k$)	0.977548	0.056410	-0.012398	1.000000	0.009903
Spending Score (1-100)	0.013835	-0.058109	-0.327227	0.009903	1.000000

Step 4: Check Skewness

In [18]:
```
ds.skew()
```

Out[18]:
```
CustomerID                0.000000
Genre                     0.243578
Age                       0.485569
Annual Income (k$)        0.321843
Spending Score (1-100)   -0.047220
dtype: float64
```

In [19]:
```
ds.sort_values(by =['Genre','Age','Annual Income (k$)','Spending Score (1-100)'])
```

Out[19]:

	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
114	115	0	18	65	48
111	112	0	19	63	54
115	116	0	19	65	50
2	3	0	20	16	6
39	40	0	20	37	75
...	...	...	...	...	...
102	103	1	67	62	59
108	109	1	68	63	43
57	58	1	69	44	46
60	61	1	70	46	56
70	71	1	70	49	55

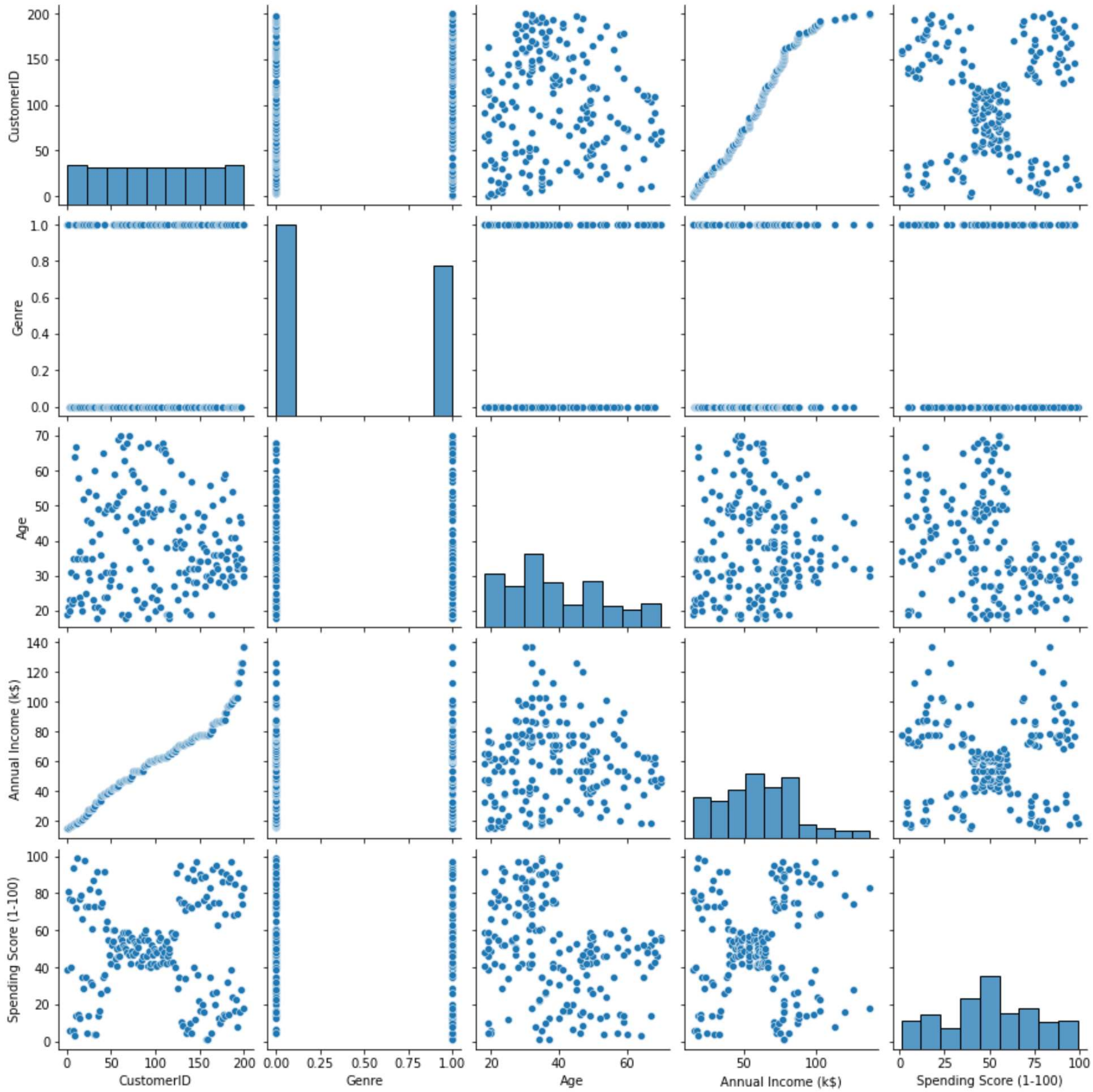
200 rows × 5 columns

Step 5: Pair plot

In [20]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [21]:
```
sns.pairplot(data=ds)
```

Out[21]: <seaborn.axisgrid.PairGrid at 0x7f8986b30ee0>



Step :6 Bulid KMeans

In [22]:
```
from sklearn.cluster import KMeans
```

In [23]:
```
ds.drop(['CustomerID'],axis=1, inplace=True)
```

In [24]:
```
KM = KMeans(n_clusters=5)
KM.fit(ds)
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value
of `n_init` explicitly to suppress the warning
  warnings.warn(
```

Out[24]:
```
KMeans(n_clusters=5)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [25]:
```
KM.labels_
```

Out[25]:
```
array([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, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 2,
       4, 4, 2, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 1, 3, 1, 4, 1, 3, 1, 3, 1,
       3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 4, 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], dtype=int32)
```

In [26]:
```
print(KM.cluster_centers_)
```

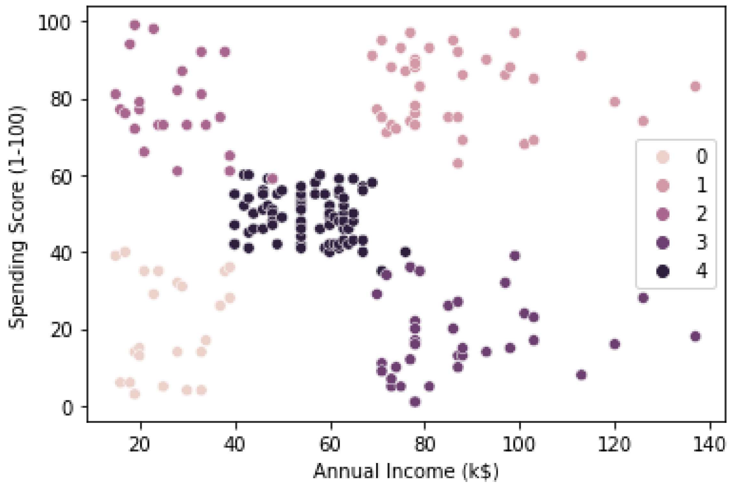
```
[[ 0.39130435 45.2173913  26.30434783 20.91304348]
 [ 0.46153846 32.69230769 86.53846154 82.12820513]
 [ 0.44       24.96       28.04       77.        ]
 [ 0.52777778 40.66666667 87.75       17.58333333]
 [ 0.4025974  43.72727273 55.48051948 49.32467532]]
```

Step 7: Scatter Plot

In [27]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [29]:
```
sns.scatterplot(ds['Annual Income (k$)'], ds['Spending Score (1-100)'], hue=KM.labels_
```

Out[29]:
```
<AxesSubplot:xlabel='Annual Income (k$)', ylabel='Spending Score (1-100)'>
```



Step : 8 Cluster Analysis

In [30]:
```
kmeans2 = KMeans(n_clusters = 5, init='k-means++')
kmeans2.fit(ds)
pred = kmeans2.predict(ds)
```

In [33]:
```
frame = pd.DataFrame(ds)
frame['cluster'] = pred
```

In [34]:
```
frame.cluster.value_counts()
```

Out[34]:
```
1    79
3    39
0    36
2    23
4    23
Name: cluster, dtype: int64
```

In [35]:

```
frame
```

Out[35]:

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

200 rows × 5 columns

In [36]:

```
C0 = ds[ds['cluster'] == 0]
C1 = ds[ds['cluster'] == 1]
C2 = ds[ds['cluster'] == 2]
C3 = ds[ds['cluster'] == 3]
C4 = ds[ds['cluster'] == 4]
```

In [37]:

```
import statistics as ss
print('Average Age : ',C0['Age'].mean())
print('Average Annual Income : ',C0['Annual Income (k$)'].mean())
print('Deviation of the mean for annual Income : ',ss.stdev(C0['Annual Income (k$)']))
print('No. of Customers ie shape :' ,C0.shape)
print('From those Customers We have',C0.Genre.value_counts()[1],'male and',C0.Genre.va
```

```
Average Age :  40.666666666666664
Average Annual Income :  87.75
Deviation of the mean for annual Income :  16.387059354433127
No. of Customers ie shape : (36, 5)
From those Customers We have 19 male and 17 female
```

In [38]:

```
import statistics as ss
print('Average Age : ',C1['Age'].mean())
print('Average Annual Income : ',C1['Annual Income (k$)'].mean())
print('Deviation of the mean for annual Income : ',ss.stdev(C0['Annual Income (k$)']))
print('No. of Customers ie shape :' ,C1.shape)
print('From those Customers We have',C1.Genre.value_counts()[1],'male and',C1.Genre.va
```

```
Average Age :  43.08860759493671
Average Annual Income :  55.29113924050633
Deviation of the mean for annual Income :  16.387059354433127
No. of Customers ie shape : (79, 5)
From those Customers We have 33 male and 46 female
```

In [39]:
```
import statistics as ss
print('Average Age : ',C2['Age'].mean())
print('Average Annual Income : ',C2['Annual Income (k$)'].mean())
print('Deviation of the mean for annual Income : ',ss.stdev(C2['Annual Income (k$)']))
print('No. of Customers ie shape :' ,C2.shape)
print('From those Customers We have',C2.Genre.value_counts()[1],'male and',C2.Genre.va
```

```
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 [40]:
```
import statistics as ss
print('Average Age : ',C3['Age'].mean())
print('Average Annual Income : ',C3['Annual Income (k$)'].mean())
print('Deviation of the mean for annual Income : ',ss.stdev(C3['Annual Income (k$)']))
print('No. of Customers ie shape :' ,C3.shape)
print('From those Customers We have',C3.Genre.value_counts()[1],'male and',C3.Genre.va
```

```
Average Age :  32.69230769230769
Average Annual Income :  86.53846153846153
Deviation of the mean for annual Income :  16.312484972924967
No. of Customers ie shape : (39, 5)
From those Customers We have 18 male and 21 female
```

In [41]:
```
import statistics as ss
print('Average Age : ',C4['Age'].mean())
print('Average Annual Income : ',C4['Annual Income (k$)'].mean())
print('Deviation of the mean for annual Income : ',ss.stdev(C4['Annual Income (k$)']))
print('No. of Customers ie shape :' ,C4.shape)
print('From those Customers We have',C4.Genre.value_counts()[1],'male and',C4.Genre.va
```

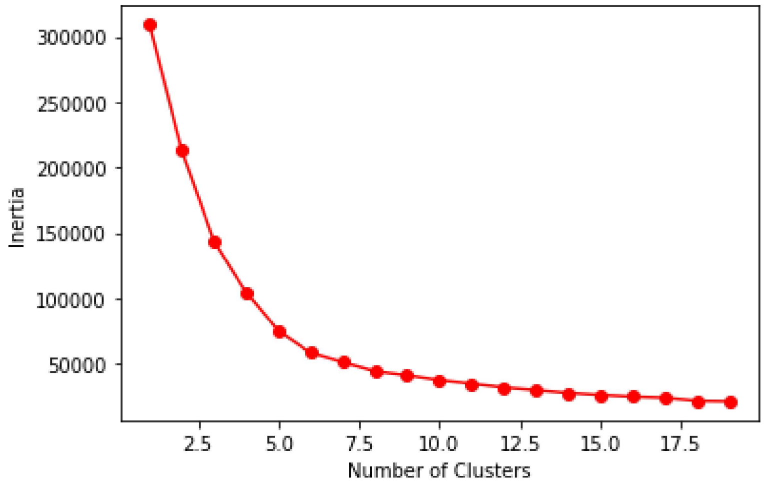
```
Average Age :  25.52173913043478
Average Annual Income :  26.304347826086957
Deviation of the mean for annual Income :  7.893811054517766
No. of Customers ie shape : (23, 5)
From those Customers We have 9 male and 14 female
```

Step 9: Find the best number

In [44]:
```
SSE = []
for clust in range(1,20):
  KM = KMeans(n_clusters= clust, init='k-means++')
  KM = KM.fit(ds)
  SSE.append(KM.inertia_)
```

In [45]:
```
plt.plot(np.arange(1,20), SSE,'ro-')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
```

Out[45]: Text(0, 0.5, 'Inertia')



Step 10 Reduce Dimensions using PCA

In [46]:
```
from sklearn.decomposition import PCA
```

In [47]:
```
pca = PCA(n_components=2)
_PCA = pca.fit_transform(ds)
PCA_Components = pd.DataFrame(_PCA)
```

In [48]:
```
PCA_Components
```

Out[48]:

	0	1
0	-30.783923	-34.018327
1	2.631040	-56.834422
2	-56.937768	-14.998902
3	-0.407293	-53.569628
4	-31.171684	-31.417704
...	...	...
195	57.346981	32.844848
196	17.733956	67.079578
197	57.281184	40.170501
198	18.391739	80.029752
199	71.086923	44.087025

200 rows × 2 columns

In [49]:
```
KM1 = KMeans(n_clusters=5)
KM1.fit(PCA_Components)
KM1.cluster_centers_
```

Out[49]:
```
array([[-11.60539958,  42.00674121],
       [ 41.48919069,   3.1814471 ],
       [ -4.32243398,  -3.21888597],
       [-44.02514268, -11.5008566 ],
       [  6.52377351, -46.53805847]])
```

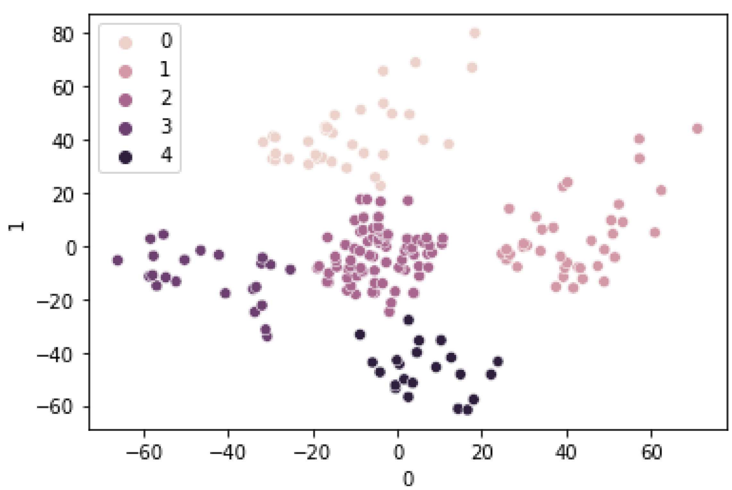
In [50]:
```
KM1.labels_
```

Out[50]:
```
array([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, 2,
       3, 4, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 1, 2, 1, 0, 1, 0, 1,
       2, 1, 0, 1, 0, 1, 0, 1, 0, 1, 2, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1,
       0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1,
       0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1,
       0, 1], dtype=int32)
```

Step 11 : Scatter Plot

In [51]:
```
sns.scatterplot(PCA_Components[0], PCA_Components[1], hue=KM1.labels_)
```

Out[51]: <AxesSubplot:xlabel='0', ylabel='1'>



Step 12 MeanShift clustering

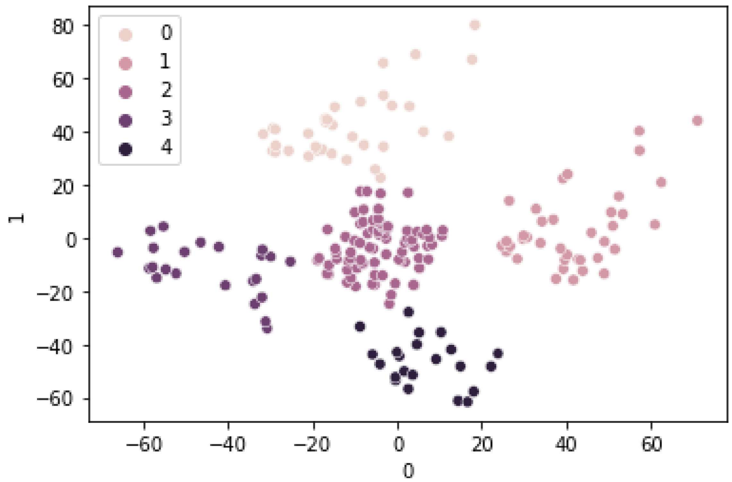
In [52]:
```
from sklearn.cluster import MeanShift, AgglomerativeClustering
```

In [53]:
```
MS = MeanShift(bandwidth = 50)
MS.fit(PCA_Components)
MS.cluster_centers_
```

Out[53]:
```
array([[ 0.48039133, -4.08943878]])
```

In [54]:
```
sns.scatterplot(PCA_Components[0], PCA_Components[1], hue=KM1.labels_)
```

Out[54]:
```
<AxesSubplot:xlabel='0', ylabel='1'>
```



Step 13 Predict hierarchical clusters using AgglomerativeClustering

In [63]:
```
As = AgglomerativeClustering(n_clusters = 5, linkage='ward',compute_full_tree=True)
As.fit(ds)
```

Out[63]:
```
AgglomerativeClustering(compute_full_tree=True, n_clusters=5)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [64]:
```
As.labels_
```

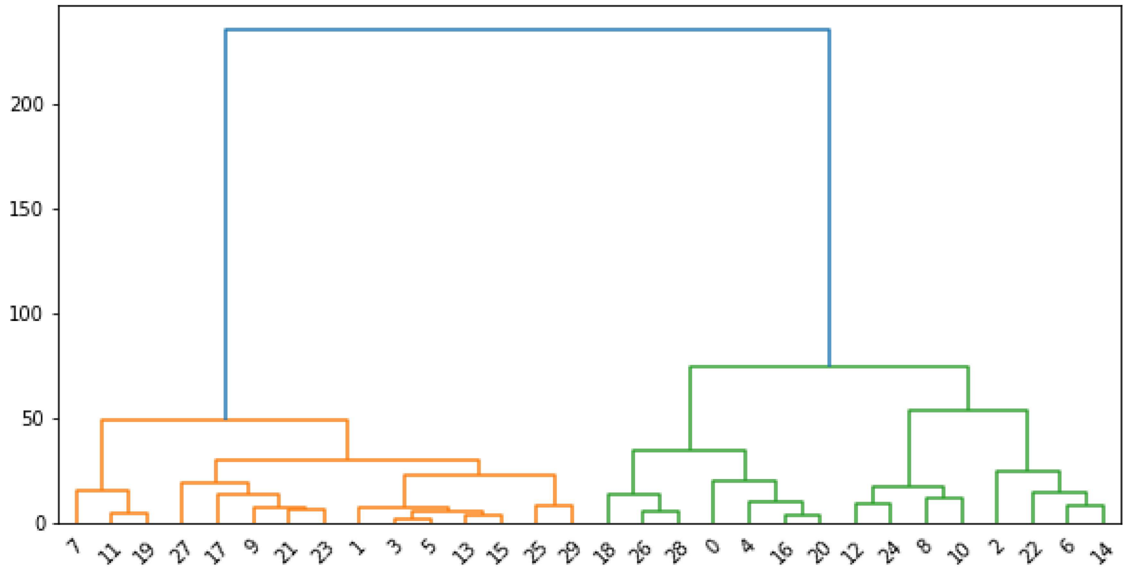
Out[64]:
```
array([4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3,
       4, 3, 4, 3, 4, 0, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 0,
       4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 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])
```

In [65]:
```
ds['Cluster'] = As.labels_
```

In [66]:
```
import scipy.cluster.hierarchy as sch
```

In [67]:
```
from scipy.cluster import hierarchy
```

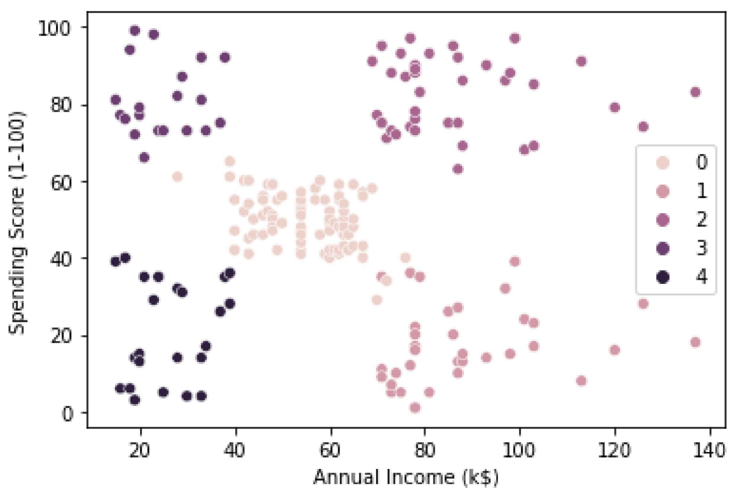
In [68]:
```
C = hierarchy.linkage(ds[:30], 'ward')
plt.figure(figsize=(10,5))
dn = hierarchy.dendrogram(C)
```



Step 14: Visualize scatter plot with hue as agglomerativeClustering Labels_

In [70]:
```
sns.scatterplot(ds['Annual Income (k$)'], ds['Spending Score (1-100)'], hue=As.labels_
```

Out[70]: <AxesSubplot:xlabel='Annual Income (k$)', ylabel='Spending Score (1-100)'>



In [ ]: