Name:P.Asha Belcilda

Rollno:2252229104

PML Lab11. Shopping Mall Customer Segmentation using Clustering

STEP -1 UNDERSTAND DATA

In [48]:
```
import pandas as pd
import numpy as np
```

In [49]:
```
df = pd.read_csv('Mall_Customers.csv')
```

In [50]:
```
# head
df.head()
```

Out[50]:

	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 [51]:
```
#shape
df.shape
```

Out[51]:
```
(200, 5)
```

In [52]:
```
#size
df.size
```

Out[52]:
```
1000
```

In [53]:
```
#columns
df.columns
```

Out[53]:
```
Index(['CustomerID', 'Genre', 'Age', 'Annual Income (k$)',
       'Spending Score (1-100)'],
      dtype='object')
```

In [54]:
```
#value_counts
df.Genre.value_counts()
```

Out[54]:
```
Female    112
Male       88
Name: Genre, dtype: int64
```

In [55]:
```
#info
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
CustomerID                200 non-null int64
Genre                     200 non-null object
Age                       200 non-null int64
Annual Income (k$)        200 non-null int64
Spending Score (1-100)    200 non-null int64
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
```

In [56]:
```
#dtypes
df.dtypes
```

Out[56]:
```
CustomerID                 int64
Genre                     object
Age                        int64
Annual Income (k$)         int64
Spending Score (1-100)     int64
dtype: object
```

STEP - 2 LABEL ENCODE GENDER

In [57]:
```
# Genre (ie., gender) is a string, so label encode into binary
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()
df['Genre']= label_encoder.fit_transform(df['Genre'])
df['Genre'].unique()
```

Out[57]:
```
array([1, 0], dtype=int64)
```

STEP - 3 CHECK FOR VARIANCE

In [58]:
```
df.describe()
```

Out[58]:

	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 [59]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
CustomerID                200 non-null int64
Genre                     200 non-null int64
Age                       200 non-null int64
Annual Income (k$)        200 non-null int64
Spending Score (1-100)    200 non-null int64
dtypes: int64(5)
memory usage: 7.9 KB
```

In [60]:
```
df.var()
```

Out[60]:
```
CustomerID                3350.000000
Genre                        0.247638
Age                        195.133166
Annual Income (k$)         689.835578
Spending Score (1-100)     666.854271
dtype: float64
```

In [61]:
```
df.corr()
```

Out[61]:

	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 [62]:
```
df.skew()
```

Out[62]:
```
CustomerID                0.000000
Genre                     0.243578
Age                       0.485569
Annual Income (k$)        0.321843
Spending Score (1-100)   -0.047220
dtype: float64
```

In [63]:
```
df.sort_values(by =['Genre','Age','Annual Income (k$)','Spending Score (1-100)'
```

Out[63]:

	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
31	32	0	21	30	73
35	36	0	21	33	81
84	85	0	21	54	57
105	106	0	21	62	42
5	6	0	22	17	76
87	88	0	22	57	55
3	4	0	23	16	77
7	8	0	23	18	94
29	30	0	23	29	87
78	79	0	23	54	52
100	101	0	23	62	41
124	125	0	23	70	29
13	14	0	24	20	77
45	46	0	24	39	65
132	133	0	25	72	34
47	48	0	27	40	47
58	59	0	27	46	51
97	98	0	27	60	50
155	156	0	27	78	89
142	143	0	28	76	40
48	49	0	29	40	42
135	136	0	29	73	88
161	162	0	29	79	83
183	184	0	29	98	88
9	10	0	30	19	72
...	...	...	...	...	...
130	131	1	47	71	9
42	43	1	48	39	36
85	86	1	48	54	46
92	93	1	48	60	49
98	99	1	48	61	42

	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
146	147	1	48	77	36
104	105	1	49	62	56
164	165	1	50	85	26
18	19	1	52	23	29
32	33	1	53	33	4
59	60	1	53	46	46
107	108	1	54	63	46
80	81	1	57	54	51
176	177	1	58	88	15
53	54	1	59	43	60
74	75	1	59	54	47
128	129	1	59	71	11
178	179	1	59	93	14
30	31	1	60	30	4
64	65	1	63	48	51
8	9	1	64	19	3
110	111	1	65	63	52
109	110	1	66	63	48
10	11	1	67	19	14
82	83	1	67	54	41
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

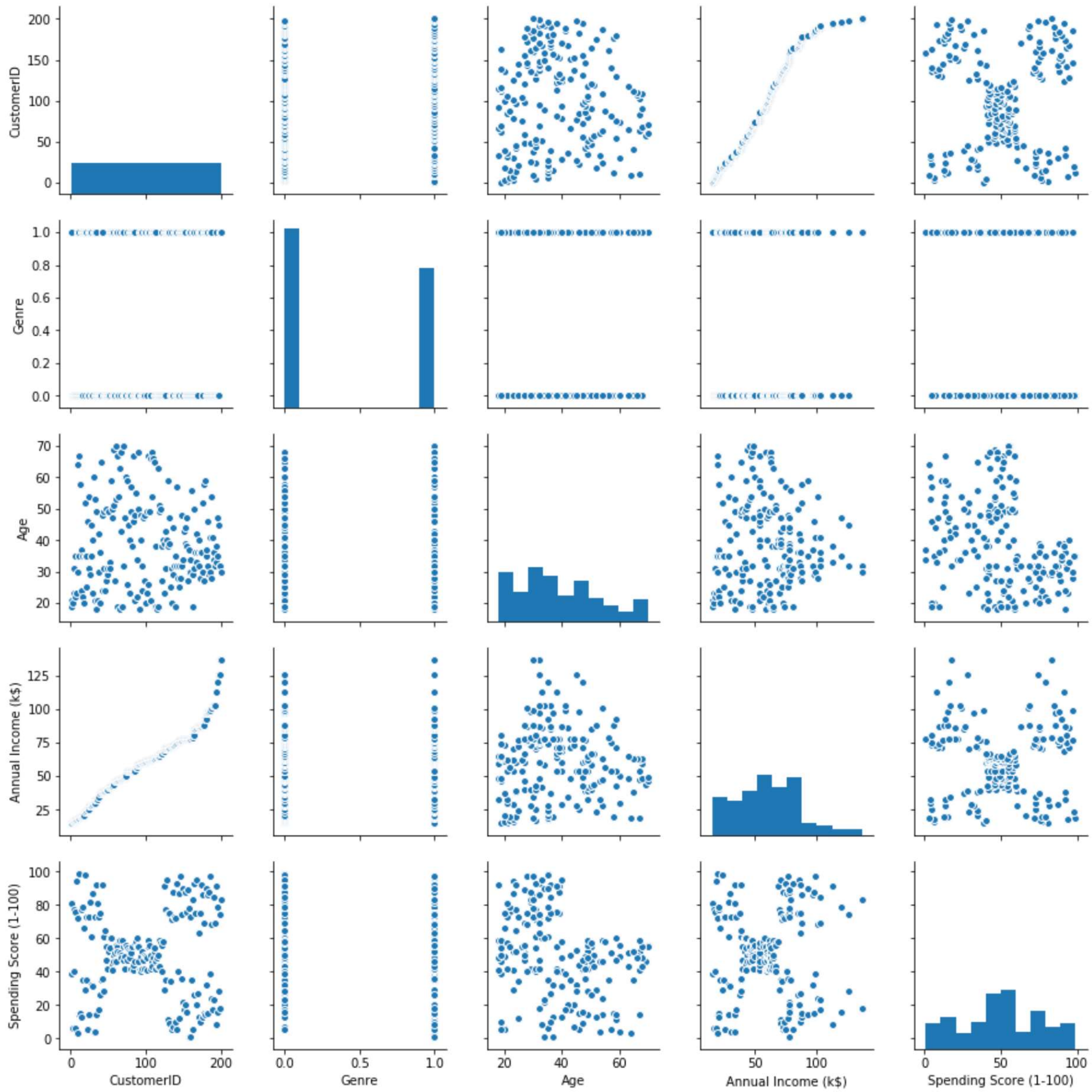
200 rows × 5 columns

STEP 5 PAIR PLOT

In [64]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [65]:
```
sns.pairplot(data=df)
```

Out[65]: <seaborn.axisgrid.PairGrid at 0x1215741c240>



STEP - 6 BUILD KMEANS

In [66]:
```
from sklearn.cluster import KMeans
```

In [67]:
```
df.drop(['CustomerID'],axis=1, inplace=True)
```

In [68]:
```
KM = KMeans(n_clusters=5)
```

In [69]:
```
KM.fit(df)
```

Out[69]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=5, n_init=10, n_jobs=1, precompute_distances='auto',
    random_state=None, tol=0.0001, verbose=0)
```

In [70]:
```
KM.labels_
```

Out[70]:
```
array([3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0,
       3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 1, 0, 1, 0,
       3, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0,
       1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0,
       0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1,
       1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 4, 2, 4, 1, 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, 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])
```

In [71]:
```
print(KM.cluster_centers_)
```

```
[[ 0.4        24.8        41.46       63.7       ]
 [ 0.43396226 53.50943396 54.73584906 48.47169811]
 [ 0.51351351 40.32432432 87.43243243 18.18918919]
 [ 0.38095238 44.14285714 25.14285714 19.52380952]
 [ 0.46153846 32.69230769 86.53846154 82.12820513]]
```

STEP - 7 SCATTER PLOT

In [72]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [98]:
```
sns.scatterplot(df['Annual Income (k$)'], df['Spending Score (1-100)'], hue=KM.
```

...

STEP - 8 CLUSTER ANALYSIS

In [99]:
```
kmeans2 = KMeans(n_clusters = 5, init='k-means++')
kmeans2.fit(df)
pred = kmeans2.predict(df)
```

In [100]:
```
frame = pd.DataFrame(df)
frame['cluster'] = pred
```

In [101]:
```
frame.cluster.value_counts()
```

Out[101]:
```
0    79
1    39
3    36
4    23
2    23
Name: cluster, dtype: int64
```

In [102]:
```
frame
```

Out[102]:

	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
5	0	22	17	76	4
6	0	35	18	6	2
7	0	23	18	94	4
8	1	64	19	3	2
9	0	30	19	72	4
10	1	67	19	14	2
11	0	35	19	99	4
12	0	58	20	15	2
13	0	24	20	77	4
14	1	37	20	13	2
15	1	22	20	79	4
16	0	35	21	35	2
17	1	20	21	66	4
18	1	52	23	29	2
19	0	35	23	98	4
20	1	35	24	35	2
21	1	25	24	73	4
22	0	46	25	5	2
23	1	31	25	73	4
24	0	54	28	14	2
25	1	29	28	82	4
26	0	45	28	32	2
27	1	35	28	61	4
28	0	40	29	31	2
29	0	23	29	87	4
...	...	...	...	...	...
170	1	40	87	13	3
171	1	28	87	75	1
172	1	36	87	10	3
173	1	36	87	92	1
174	0	52	88	13	3

	Genre	Age	Annual Income (k$)	Spending Score (1-100)	cluster
175	0	30	88	86	1
176	1	58	88	15	3
177	1	27	88	69	1
178	1	59	93	14	3
179	1	35	93	90	1
180	0	37	97	32	3
181	0	32	97	86	1
182	1	46	98	15	3
183	0	29	98	88	1
184	0	41	99	39	3
185	1	30	99	97	1
186	0	54	101	24	3
187	1	28	101	68	1
188	0	41	103	17	3
189	0	36	103	85	1
190	0	34	103	23	3
191	0	32	103	69	1
192	1	33	113	8	3
193	0	38	113	91	1
194	0	47	120	16	3
195	0	35	120	79	1
196	0	45	126	28	3
197	1	32	126	74	1
198	1	32	137	18	3
199	1	30	137	83	1

200 rows × 5 columns

In [78]:

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

In [79]:
```
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 (
print('No. of Customers ie shape :' ,C0.shape)
print('From those Customers We have',C0.Genre.value_counts()[1],'male and',C0.G
```

```
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 18
```

In [82]:
```
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(C1['Annual Income (
print('No. of Customers ie shape :' ,C1.shape)
print('From those Customers We have',C1.Genre.value_counts()[1],'male and',C1.G
```

```
Average Age :  43.72727272727273
Average Annual Income :  55.48051948051948
Deviation of the mean for annual Income :  8.742832236527411
No. of Customers ie shape : (77, 5)
From those Customers We have 31 male and 31
```

In [83]:
```
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 (
print('No. of Customers ie shape :' ,C2.shape)
print('From those Customers We have',C2.Genre.value_counts()[1],'male and',C2.G
```

```
Average Age :  24.96
Average Annual Income :  28.04
Deviation of the mean for annual Income :  9.654359982239457
No. of Customers ie shape : (25, 5)
From those Customers We have 11 male and 11
```

In [84]:
```
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 (
print('No. of Customers ie shape :' ,C3.shape)
print('From those Customers We have',C3.Genre.value_counts()[1],'male and',C3.G
```

```
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 19
```

In [85]:
```
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 (
print('No. of Customers ie shape :' ,C4.shape)
print('From those Customers We have',C4.Genre.value_counts()[1],'male and',C4.G
```

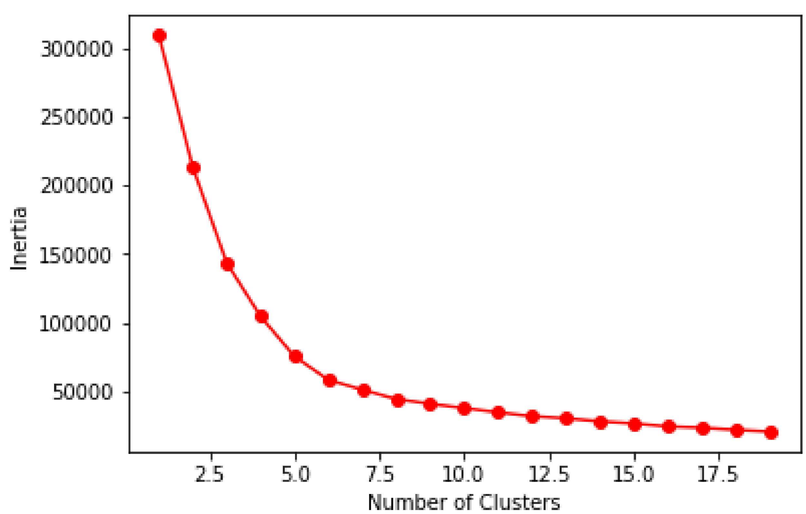
```
Average Age :  45.21739130434783
Average Annual Income :  26.304347826086957
Deviation of the mean for annual Income :  7.893811054517766
No. of Customers ie shape : (23, 5)
From those Customers We have 9 male and 9
```

STEP 9 FIND THE BEST NUMBER

In [89]:
```
SSE = []
for clust in range(1,20):
    KM = KMeans(n_clusters= clust, init='k-means++')
    KM = KM.fit(df)
    SSE.append(KM.inertia_)
```

In [90]:
```
plt.plot(np.arange(1,20), SSE,'ro-')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
```

Out[90]: Text(0,0.5,'Inertia')



STEP -10 REDUCE DIMESNSION USING PCA

In [91]:
```
from sklearn.decomposition import PCA
```

In [92]:
```
pca = PCA(n_components=2)
_PCA = pca.fit_transform(df)
PCA_Components = pd.DataFrame(_PCA)
```

In [93]:
```
PCA_Components
```

Out[93]:

	0	1
0	-31.532645	-33.381457
1	1.448933	-56.823775
2	-57.297507	-13.829755
3	-1.523021	-53.496952
4	-31.865811	-30.773356
5	-1.547168	-52.245811
6	-58.994143	-10.270642
7	13.106185	-61.451941
8	-66.309657	-4.034841
9	-5.082493	-47.330620
10	-58.162524	-9.852324
11	15.364576	-61.913645
12	-55.079684	-10.758648
13	0.599482	-50.105484
14	-52.670895	-12.327940
15	2.564571	-51.488747
16	-34.276510	-24.176811
17	-6.779627	-43.599196
18	-41.109021	-16.975647
19	16.884757	-58.086776
20	-32.542334	-21.728017
21	-0.448450	-44.456801
22	-57.831625	-2.582956
23	-1.011469	-42.870008
24	-50.486003	-4.182855
25	8.236144	-45.754996
26	-34.508834	-15.489814
27	-9.547990	-33.141834
28	-33.772184	-14.752709
29	13.918471	-48.531434
...	...	...
170	-14.465678	42.704595
171	37.067156	6.177396
172	-16.082444	43.882289
173	49.018418	-2.381003
174	-16.169540	45.060867

	0	1
175	45.983015	1.045569
176	-15.726045	44.705674
177	33.080405	10.248086
178	-13.817887	49.476792
179	51.092629	3.512747
180	6.944925	39.760603
181	50.806197	8.644245
182	-7.661899	51.320966
183	53.540012	7.946639
184	12.888126	37.958748
185	61.060325	3.817272
186	-0.315926	49.719493
187	39.613968	21.545123
188	-2.234632	53.627727
189	52.721933	14.616630
190	3.851888	49.344705
191	40.802395	23.125153
192	-2.064322	65.833817
193	62.878416	19.647586
194	5.660638	68.830266
195	57.985992	31.739157
196	19.020341	66.700768
197	58.062687	39.069193
198	19.927301	79.643964
199	71.935705	42.710124

200 rows × 2 columns

In [94]:
```
KM1 = KMeans(n_clusters=5)
KM1.fit(PCA_Components)
KM1.cluster_centers_
```

Out[94]:
```
array([[ 41.55727351,   2.38339005],
       [ -4.3467296 ,  -3.16043104],
       [  5.54186711, -46.6021966 ],
       [-44.3173493 , -10.5924271 ],
       [-10.7918789 ,  42.20815536]])
```

In [95]:
```
KM1.labels_
```

Out[95]:
```
array([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, 3, 1,
       3, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 4, 0, 4, 0,
       1, 0, 4, 0, 4, 0, 4, 0, 4, 0, 1, 0, 4, 0, 4, 0, 4, 0, 4, 0, 4, 0,
       4, 0, 4, 0, 4, 0, 4, 0, 4, 0, 4, 0, 4, 0, 4, 0, 4, 0, 4, 0, 4, 0,
       4, 0, 4, 0, 4, 0, 4, 0, 4, 0, 4, 0, 4, 0, 4, 0, 4, 0, 4, 0, 4, 0,
       4, 0])
```

STEP 11 SCATTER PLOT

In [97]:
```
sns.scatterplot(PCA_Components[0], PCA_Components[1], hue=KM1.labels_)
```

...

STEP 12 MEAN SHIFT CLUSTERING

In [103]:
```
from sklearn.cluster import MeanShift, AgglomerativeClustering
```

In [104]:
```
MS = MeanShift(bandwidth = 50)
MS.fit(PCA_Components)
MS.cluster_centers_
```

Out[104]:
```
array([[ 0.40694764, -4.10211689]])
```

In [105]:
```
sns.scatterplot(PCA_Components[0], PCA_Components[1], hue=KM1.labels_)
```

...

STEP 13 PREDICT HIERARCHICAL CLUSTERS USING AGGLOMERATIVE CLUSTERING

In [106]:
```
AC = AgglomerativeClustering(n_clusters = 5, linkage='ward',compute_full_tree=T
AC.fit(df)
```

Out[106]:
```
AgglomerativeClustering(affinity='euclidean', compute_full_tree=True,
                        connectivity=None, linkage='ward', memory=None, n_clusters=5,
                        pooling_func=<function mean at 0x0000012151ADC158>)
```

In [107]:
```
AC.labels_
```

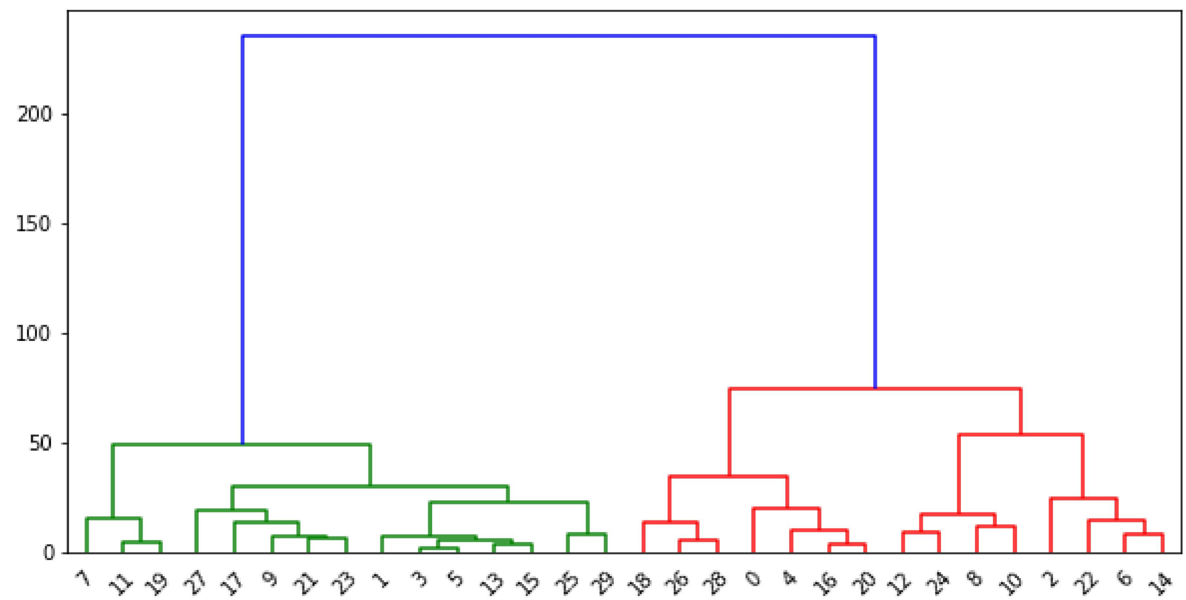
Out[107]:
```
array([4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3,
       4, 3, 4, 3, 4, 0, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 0,
       4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 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, 1, 0, 1, 0, 1, 2, 1, 2, 1,
       0, 1, 2, 1, 2, 1, 2, 1, 2, 1, 0, 1, 2, 1, 0, 1, 2, 1, 2, 1, 2, 1,
       2, 1, 2, 1, 2, 1, 0, 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], dtype=int64)
```

In [108]:
```
df['Cluster'] = AC.labels_
```

In [109]:
```
import scipy.cluster.hierarchy as sch
```

In [110]:
```
from scipy.cluster import hierarchy
```

In [111]:
```
Z = hierarchy.linkage(df[:30], 'ward')
plt.figure(figsize=(10,5))
dn = hierarchy.dendrogram(Z)
```



STEP 14 VISUALIZE SCATTER PLOT WITH HUE AS AGGLOMERATIVECLUSTERING LABELS_

In [112]:
```
sns.scatterplot(df['Annual Income (k$)'], df['Spending Score (1-100)'], hue=AC.
```

...

In [ ]:
```

```