SHOPPING MALL CUSTOMER SEGMENTATION USING CLUSTERING

HARI PRASATH S

225229110

STEP -1:UNDERSTAND DATA

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

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

In [3]:
```
# properties
df.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]:
```
df.shape
```

Out[4]: (200, 5)

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

Out[5]: 1000

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

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

In [8]:
```
df.CustomerID.value_counts()
```

Out[8]:
```
200    1
63     1
73     1
72     1
71     1
70     1
69     1
68     1
67     1
66     1
65     1
64     1
62     1
199    1
61     1
60     1
59     1
58     1
57     1
56     1
55     1
54     1
53     1
52     1
74     1
75     1
76     1
77     1
98     1
97     1
      ..
106    1
105    1
104    1
103    1
124    1
126    1
149    1
127    1
148    1
147    1
146    1
145    1
144    1
143    1
142    1
141    1
140    1
139    1
138    1
137    1
136    1
135    1
134    1
133    1
132    1
131    1
130    1
129    1
128    1
1      1
Name: CustomerID, Length: 200, dtype: int64
```

In [9]:
```
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 [10]:
```
df.dtypes
```

Out[10]:
```
CustomerID                 int64
Genre                     object
Age                        int64
Annual Income (k$)         int64
Spending Score (1-100)     int64
dtype: object
```

STEP - 2:LABEL ENCODE GENDER

In [11]:
```
# 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[11]:
```
array([1, 0], dtype=int64)
```

STEP - 3:CHECK FOR VARIANCE

In [12]:
```
df.describe()
```

Out[12]:

	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 [13]:
```
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 [14]:
```
df.var()
```

Out[14]:
```
CustomerID                3350.000000
Genre                        0.247638
Age                        195.133166
Annual Income (k$)         689.835578
Spending Score (1-100)     666.854271
dtype: float64
```

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

Out[15]:

	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 [16]:
```
df.skew()
```

Out[16]:
```
CustomerID                0.000000
Genre                     0.243578
Age                       0.485569
Annual Income (k$)        0.321843
Spending Score (1-100)   -0.047220
dtype: float64
```

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

Out[17]:

	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
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

	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
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

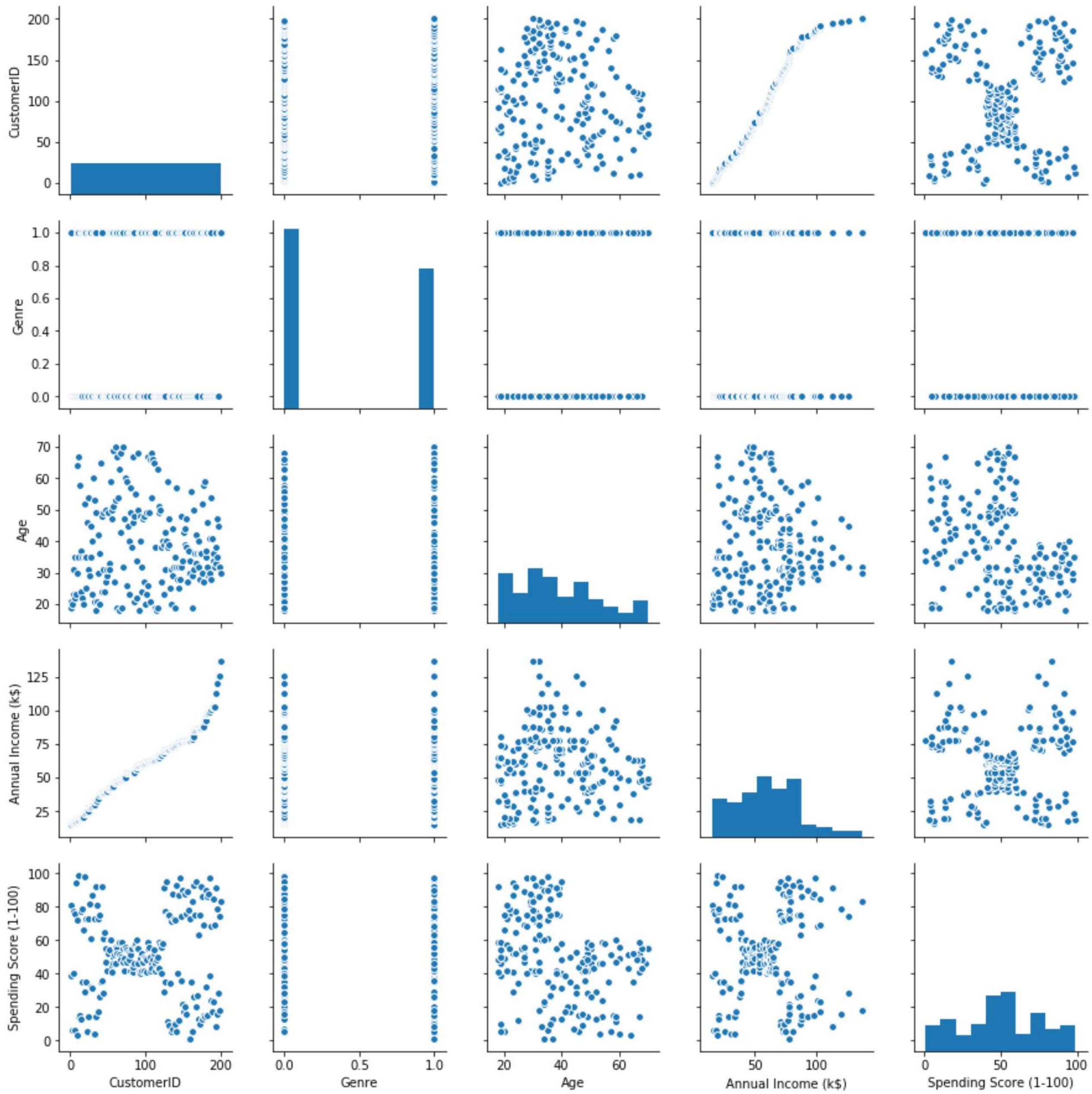
200 rows × 5 columns

STEP 5:PAIR PLOT

In [18]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [19]:
```
sns.pairplot(data=df)
```

Out[19]: <seaborn.axisgrid.PairGrid at 0x22d3fb07cf8>



STEP6:BUILD KMEANS

In [20]:
```
from sklearn.cluster import KMeans
```

In [21]:
```
df.drop(['CustomerID'],axis=1, inplace=True)
```

In [22]:
```
KM = KMeans(n_clusters=5)
```

In [23]:
```
KM.fit(df)
```

Out[23]:
```
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 [25]:
```
KM.labels_
```

Out[25]:
```
array([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, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 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, 4, 2, 0, 2, 4, 2, 4, 2,
       4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 0, 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])
```

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

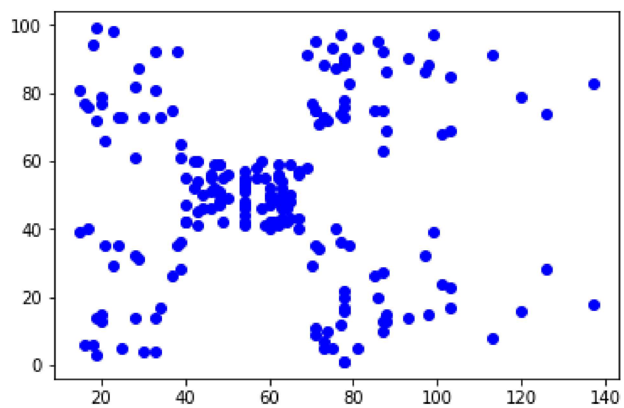
```
[[ 0.41772152 43.08860759 55.29113924 49.56962025]
 [ 0.39130435 25.52173913 26.30434783 78.56521739]
 [ 0.46153846 32.69230769 86.53846154 82.12820513]
 [ 0.39130435 45.2173913  26.30434783 20.91304348]
 [ 0.52777778 40.66666667 87.75       17.58333333]]
```

STEP - 7:SCATTER PLOT

In [29]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [33]:
```
plt.scatter(df['Annual Income (k$)'], df['Spending Score (1-100)'], color='blue')
```

Out[33]:
```
<matplotlib.collections.PathCollection at 0x22d4152ed68>
```



STEP8: CLUSTER ANALYSIS

In [34]:
```
kmeans2 = KMeans(n_clusters = 5, init='k-means++')
kmeans2.fit(df)
pred = kmeans2.predict(df)
```

In [35]:
```
frame = pd.DataFrame(df)
frame['cluster'] = pred
```

In [36]:
```
frame.cluster.value_counts()
```

Out[36]:
```
1    79
3    39
4    37
2    23
0    22
Name: cluster, dtype: int64
```

In [37]:
```
frame
```

Out[37]:

	Genre	Age	Annual Income (k$)	Spending Score (1-100)	cluster
0	1	19	15	39	2
1	1	21	15	81	0
2	0	20	16	6	2
3	0	23	16	77	0
4	0	31	17	40	2
5	0	22	17	76	0
6	0	35	18	6	2
7	0	23	18	94	0
8	1	64	19	3	2
9	0	30	19	72	0
10	1	67	19	14	2
11	0	35	19	99	0
12	0	58	20	15	2
13	0	24	20	77	0
14	1	37	20	13	2
15	1	22	20	79	0
16	0	35	21	35	2
17	1	20	21	66	0
18	1	52	23	29	2
19	0	35	23	98	0
20	1	35	24	35	2
21	1	25	24	73	0
22	0	46	25	5	2
23	1	31	25	73	0
24	0	54	28	14	2
25	1	29	28	82	0
26	0	45	28	32	2
27	1	35	28	61	0
28	0	40	29	31	2
29	0	23	29	87	0
...	...	...	...	...	...
170	1	40	87	13	4
171	1	28	87	75	3
172	1	36	87	10	4
173	1	36	87	92	3
174	0	52	88	13	4
175	0	30	88	86	3
176	1	58	88	15	4
177	1	27	88	69	3
178	1	59	93	14	4
179	1	35	93	90	3
180	0	37	97	32	4
181	0	32	97	86	3

	Genre	Age	Annual Income (k$)	Spending Score (1-100)	cluster
182	1	46	98	15	4
183	0	29	98	88	3
184	0	41	99	39	4
185	1	30	99	97	3
186	0	54	101	24	4
187	1	28	101	68	3
188	0	41	103	17	4
189	0	36	103	85	3
190	0	34	103	23	4
191	0	32	103	69	3
192	1	33	113	8	4
193	0	38	113	91	3
194	0	47	120	16	4
195	0	35	120	79	3
196	0	45	126	28	4
197	1	32	126	74	3
198	1	32	137	18	4
199	1	30	137	83	3

200 rows × 5 columns

In [38]:

```
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 [62]:

```
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.value_counts()[
```

```
Average Age :  25.272727272727273
Average Annual Income :  25.727272727272727
Deviation of the mean for annual Income :  7.566730552584204
No. of Customers ie shape : (22, 5)
From those Customers We have 9 male and 13
```

In [63]:

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

```
Average Age :  43.12658227848101
Average Annual Income :  54.822784810126585
Deviation of the mean for annual Income :  8.576592314850398
No. of Customers ie shape : (79, 5)
From those Customers We have 33 male and 46
```

In [64]:
```
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.value_counts()[(
```

```
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
```

In [65]:
```
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.value_counts()[(
```

```
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
```

In [66]:
```
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.value_counts()[(
```

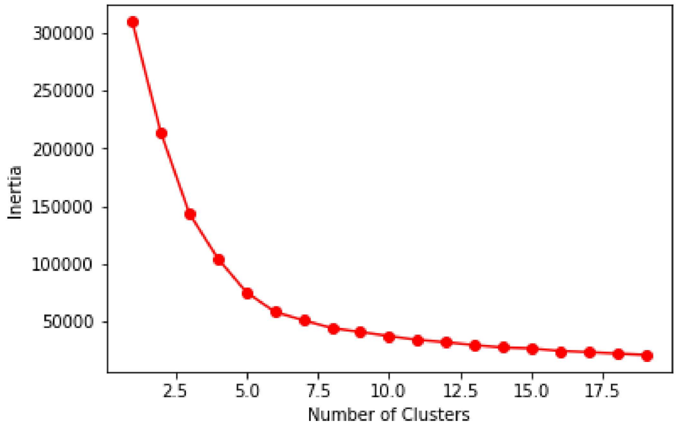
```
Average Age :  40.32432432432432
Average Annual Income :  87.43243243243244
Deviation of the mean for annual Income :  16.2729163891359
No. of Customers ie shape : (37, 5)
From those Customers We have 19 male and 18
```

STEP9:FIND THE BEST NUMBER

In [50]:
```
SSE = []
for clust in range(1,20):
    KM = KMeans(n_clusters= clust, init='k-means++')
    KM = KM.fit(df)
    SSE.append(KM.inertia_)
```

In [51]:
```
plt.plot(np.arange(1,20), SSE,'ro-')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
```

Out[51]:
```
Text(0,0.5,'Inertia')
```



STEP10: REDUCE DIMESNSION USING PCA

In [52]:
```
from sklearn.decomposition import PCA
```

In [53]:
```
pca = PCA(n_components=2)
_PCA = pca.fit_transform(df)
PCA_Components = pd.DataFrame(_PCA)
```

In [54]:
```
PCA_Components
```

Out[54]:

	0	1
0	-32.316178	-32.521082
1	-0.028294	-56.879048
2	-57.579285	-12.304252
3	-2.914984	-53.476855
4	-32.584904	-29.908105
5	-2.906245	-52.226341
6	-59.187952	-8.705258
7	11.511166	-61.812993
8	-66.351383	-2.284114
9	-6.316792	-47.223080
10	-58.354348	-8.312635
11	13.754747	-62.335528
12	-55.291653	-9.299255
13	-0.703762	-50.145748
14	-52.917051	-10.929002
15	1.226421	-51.579673
16	-34.825534	-23.254910
17	-7.914313	-43.449544
18	-41.477228	-15.881175
19	15.375106	-58.553349
20	-33.027034	-20.855003
21	-1.605072	-44.475406
22	-57.827891	-1.058005
23	-2.128618	-42.875920
24	-50.524114	-2.852672
25	7.047075	-46.003994
26	-34.834391	-14.572173
27	-10.415239	-32.931276
28	-34.076885	-13.855021
29	12.660417	-48.928244
...	...	...
170	-13.289127	43.112914
171	37.188580	5.239567
172	-14.874526	44.332790
173	48.917990	-3.628728
174	-14.935511	45.510862
175	45.972722	-0.124435
176	-14.502861	45.143662
177	33.307043	9.412147
178	-12.470090	49.858625
179	51.146692	2.203407
180	8.051848	39.601879
181	50.994770	7.337222

	0	1
182	-6.260526	51.537991
183	53.712060	6.567815
184	13.948677	37.643234
185	61.126660	2.242462
186	1.043562	49.741742
187	40.136642	20.522642
188	-0.770128	53.698087
189	53.065570	13.251648
190	5.208546	49.258224
191	41.365449	22.068893
192	-0.279370	65.887078
193	63.355555	18.006268
194	7.521927	68.673053
195	58.777689	30.215294
196	20.830597	66.190066
197	59.046297	37.535569
198	22.078862	79.096208
199	73.018819	40.803029

200 rows × 2 columns

In [55]:
```
KM1 = KMeans(n_clusters=5)
KM1.fit(PCA_Components)
KM1.cluster_centers_
```

Out[55]:
```
array([[ -4.45895813,  -3.07010572],
       [ 41.57978581,   1.32960205],
       [  4.33100055, -46.77811361],
       [ -9.62766645,  42.51895929],
       [-44.51819271,  -9.41877946]])
```

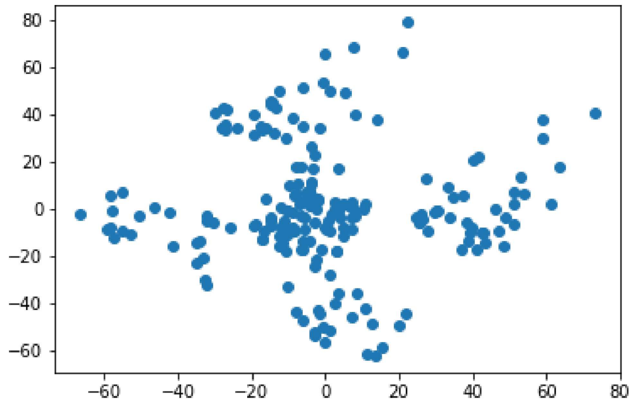
In [56]:
```
KM1.labels_
```

Out[56]:
```
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, 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, 0, 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])
```

STEP11: SCATTER PLOT

In [67]:
```
plt.scatter(PCA_Components[0], PCA_Components[1], data='KM1.labels_')
```

Out[67]:
```
<matplotlib.collections.PathCollection at 0x22d3fd5a550>
```



STEP12: MEAN SHIFT CLUSTERING

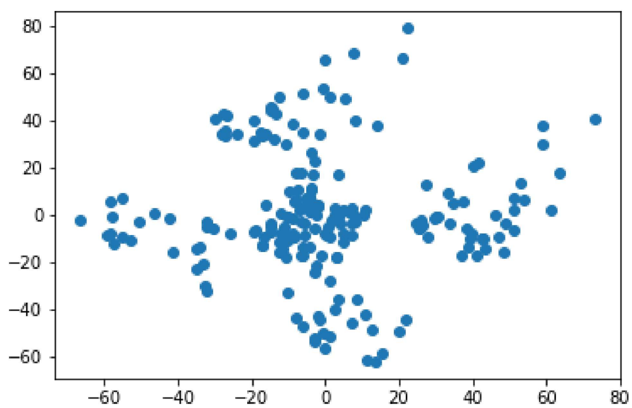
In [69]:
```
from sklearn.cluster import MeanShift, AgglomerativeClustering
```

In [70]:
```
MS = MeanShift(bandwidth = 50)
MS.fit(PCA_Components)
MS.cluster_centers_
```

Out[70]:
```
array([[ 0.29174595, -4.11523419]])
```

In [71]:
```
plt.scatter(PCA_Components[0], PCA_Components[1])
```

Out[71]:
```
<matplotlib.collections.PathCollection at 0x22d447075c0>
```



STEP13: PREDICT HIERARCHICAL CLUSTERS USING AGGLOMERATIVE CLUSTERING

In [72]:
```
AC = AgglomerativeClustering(n_clusters = 5, linkage='ward',compute_full_tree=True)
AC.fit(df)
```

Out[72]:
```
AgglomerativeClustering(affinity='euclidean', compute_full_tree=True,
            connectivity=None, linkage='ward', memory=None, n_clusters=5,
            pooling_func=<function mean at 0x0000022D3C28C6A8>)
```

In [73]:
```
AC.labels_
```

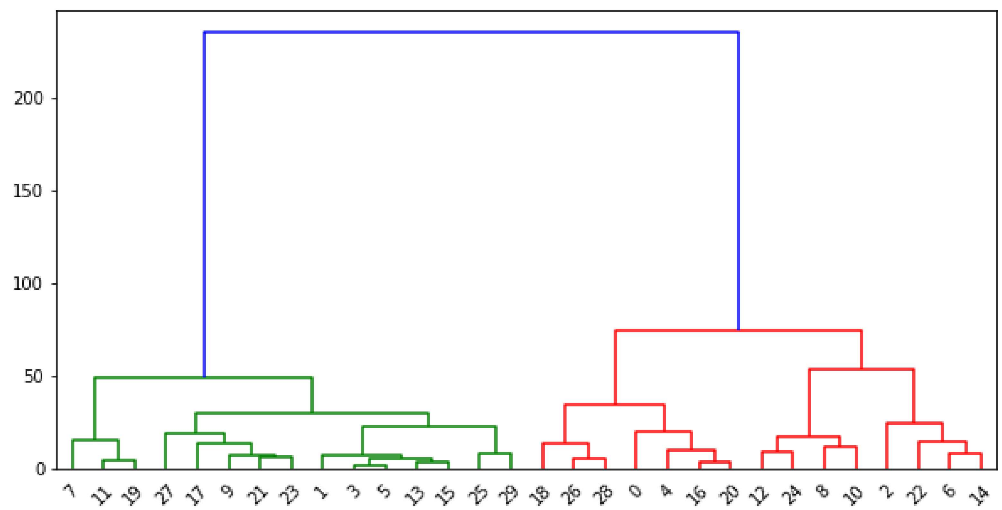
Out[73]:
```
array([4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3,
       4, 3, 4, 3, 4, 0, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 0,
       4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 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 [74]:
```
df['Cluster'] = AC.labels_
```

In [75]:
```
import scipy.cluster.hierarchy as sch
```

In [76]:
```
from scipy.cluster import hierarchy
```

In [78]:
```
Z = hierarchy.linkage(df[:30], 'ward')
plt.figure(figsize=(10,5))
dn = hierarchy.dendrogram(Z)
```



STEP14: VISUALIZE SCATTER PLOT WITH HUE AS AGGLOMERATIVECLUSTERING LABELS

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

In [ ]:
```

```