suriya s

In [ ]:

```
#step 1
```

In [2]:

```
import pandas as pd
```

In [3]:

```
df=pd.read_csv('Mall_Customers.csv')
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.columns
```

Out[5]:

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

In [6]:

```
df.dtypes
```

Out[6]:

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

In [7]:

```
df.Genre.value_counts()
```

Out[7]:

```
Female    112
Male       88
Name: Genre, dtype: int64
```

In [8]:

```
#step 2
```

In [9]:

```
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()
df['Genre']= label_encoder.fit_transform(df['Genre'])
df['Genre'].unique()
```

Out[9]:

```
array([1, 0])
```

In [10]:

```
#step 3
```

In [11]:

```
df.describe()
```

Out[11]:

	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 [12]:

```
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   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 [13]:

```
df.var()
```

Out[13]:

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

In [14]:
```
df.corr()
```

Out[14]:

	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

In [15]:
```
#step 4
```

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
...	...	...	...	...	...
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

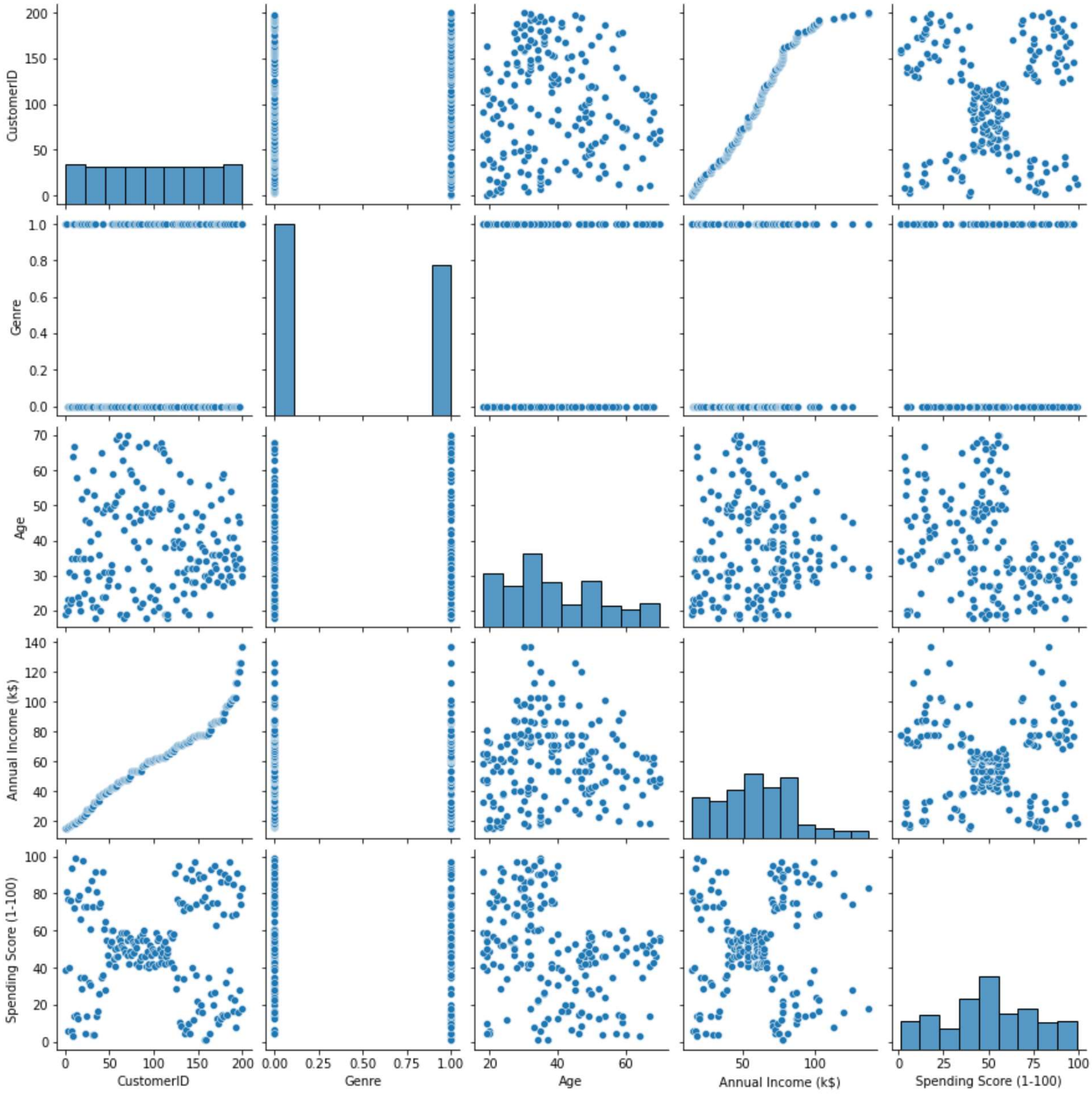
In [18]:
```
#step 5
```

In [19]:
```
import seaborn as sns
```

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

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

Out[21]:
```
<seaborn.axisgrid.PairGrid at 0x7facbba0ebe0>
```



In [22]:
```
#step 6
```

In [23]:
```
from sklearn.cluster import KMeans
```

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

In [25]:

```
KM.fit(df)
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWa
rning: 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[25]:

```
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 [26]:

```
KM.labels_
```

Out[26]:

```
array([3, 3, 1, 3, 3, 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, 1,
       1, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 4,
       4, 1, 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, 4, 0, 4, 0, 4, 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, 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], dtype=int32)
```

In [27]:

```
print(KM.cluster_centers_)
```

```
[[162.           0.46153846  32.69230769  86.53846154  82.12820513]
 [ 40.45238095   0.38095238  46.85714286  35.47619048  35.11904762]
 [164.           0.52777778  40.80555556  87.91666667  17.88888889]
 [ 21.41666667   0.41666667  25.25        24.91666667  76.04166667]
 [ 96.01694915   0.42372881  41.55932203  59.05084746  49.03389831]]
```

In [28]:

```
#step 7
```

In [29]:

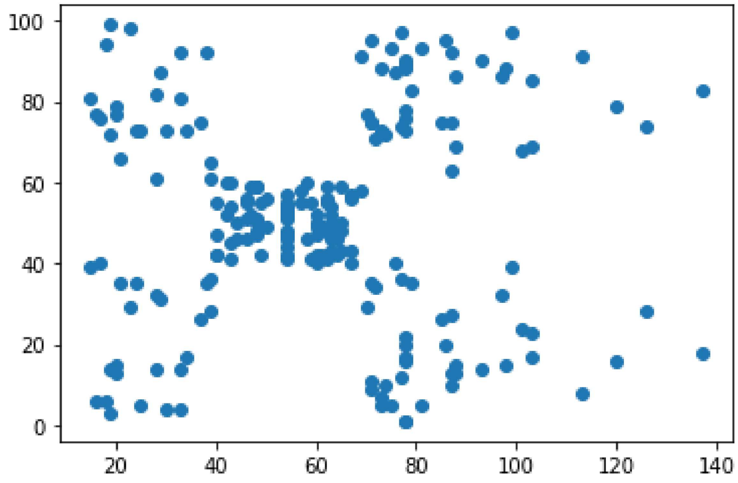
```
import warnings
warnings.filterwarnings('ignore')
```

In [30]:

```
plt.scatter(x='Annual Income (k$)', y='Spending Score (1-100)', data=df)
```

Out[30]:

```
<matplotlib.collections.PathCollection at 0x7facab33b160>
```



In [31]:

```
#step 8
```

In [31]:

```

```

In [32]:

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

In [33]:

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

In [34]:

```
frame.cluster.value_counts()
```

Out[34]:

```
0    68
4    39
1    38
2    30
3    25
Name: cluster, dtype: int64
```

In [35]:

```
frame
```

Out[35]:

	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)	cluster
0	1	1	19	15	39	2
1	2	1	21	15	81	3
2	3	0	20	16	6	2
3	4	0	23	16	77	3
4	5	0	31	17	40	2
...	...	...	...	...	...	...
195	196	0	35	120	79	4
196	197	0	45	126	28	1
197	198	1	32	126	74	4
198	199	1	32	137	18	1
199	200	1	30	137	83	4

200 rows × 6 columns

In [36]:

```
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 [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.Gen
```

```
Average Age :  43.911764705882355
Average Annual Income :  56.588235294117645
Deviation of the mean for annual Income :  7.109454067496337
No. of Customers ie shape : (68, 6)
From those Customers We have 30 male and 30
```

In [38]:

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

```
Average Age :  40.39473684210526
Average Annual Income :  87.0
Deviation of the mean for annual Income :  16.27134772404415
No. of Customers ie shape : (38, 6)
From those Customers We have 20 male and 20
```

In [39]:

```python
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.Gen
```

```
Average Age :  44.1
Average Annual Income :  29.766666666666666
Deviation of the mean for annual Income :  9.405366528755266
No. of Customers ie shape : (30, 6)
From those Customers We have 10 male and 10
```

In [40]:

```python
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.Gen
```

```
Average Age :  26.04
Average Annual Income :  27.6
Deviation of the mean for annual Income :  8.789197915623474
No. of Customers ie shape : (25, 6)
From those Customers We have 10 male and 10
```

In [41]:

```python
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.Gen
```

```
Average Age :  32.69230769230769
Average Annual Income :  86.53846153846153
Deviation of the mean for annual Income :  16.312484972924967
No. of Customers ie shape : (39, 6)
From those Customers We have 18 male and 18 female
```

In [42]:

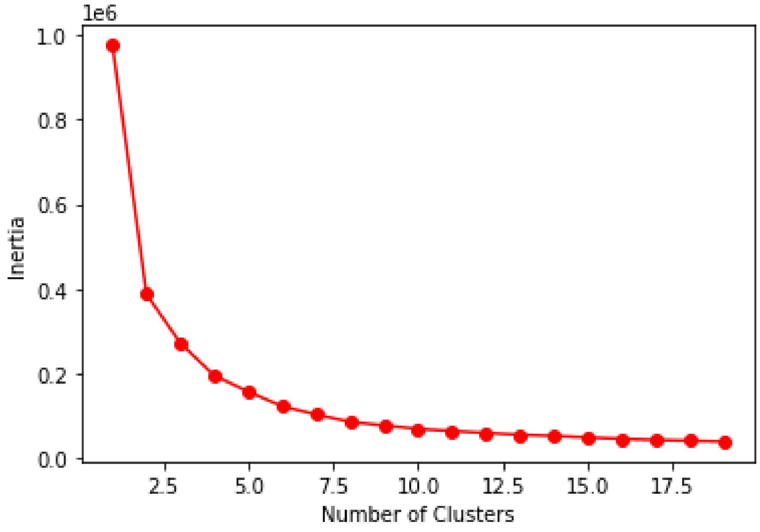
```python
#step 9
```

In [43]:
```
import numpy as np
```

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



In [46]:
```
#step 10
```

In [47]:
```
from sklearn.decomposition import PCA
```

In [48]:
```
pca = PCA(n_components=2)
_PCA = pca.fit_transform(df)
PCA_Components = pd.DataFrame(_PCA)
```

In [49]:

```
PCA_Components
```

Out[49]:

	0	1
0	-109.382564	5.447571
1	-108.197878	-34.971589
2	-107.375549	37.791205
3	-106.002925	-30.604532
4	-104.979021	7.267226
...	...	...
195	111.659330	-28.013701
196	114.612585	24.040106
197	115.918166	-23.781330
198	120.937023	30.877142
199	122.304562	-32.898226

200 rows × 2 columns

In [50]:

```
KM1 = KMeans(n_clusters=5)
KM1.fit(PCA_Components)
KM1.cluster_centers_
```

Out[50]:

```
array([[-73.21448121,  20.24695679],
       [ 68.90356677,  32.44217604],
       [ -8.45136839,   1.59152253],
       [ 67.0007653 , -32.02433556],
       [-82.21921226, -30.52721771]])
```

In [51]:

```
KM1.labels_
```

Out[51]:

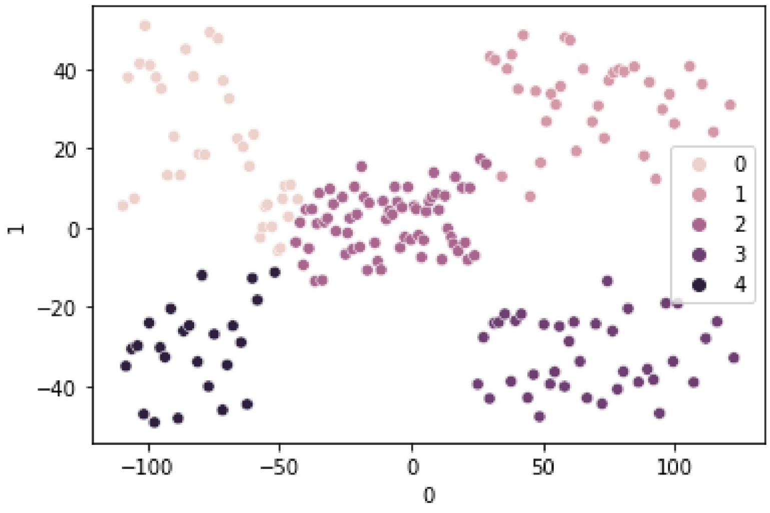
```
array([0, 4, 0, 4, 0, 4, 0, 4, 0, 4, 0, 4, 0, 4, 0, 4, 0, 4, 0, 4, 0, 4,
       0, 4, 0, 4, 0, 4, 0, 4, 0, 4, 0, 4, 0, 4, 0, 4, 0, 4, 0, 4, 0, 4,
       0, 4, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 2, 0, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
    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

In [52]:

```
#step 11
```

```
In [53]: sns.scatterplot(PCA_Components[0], PCA_Components[1], hue=KM1.labels_)
```

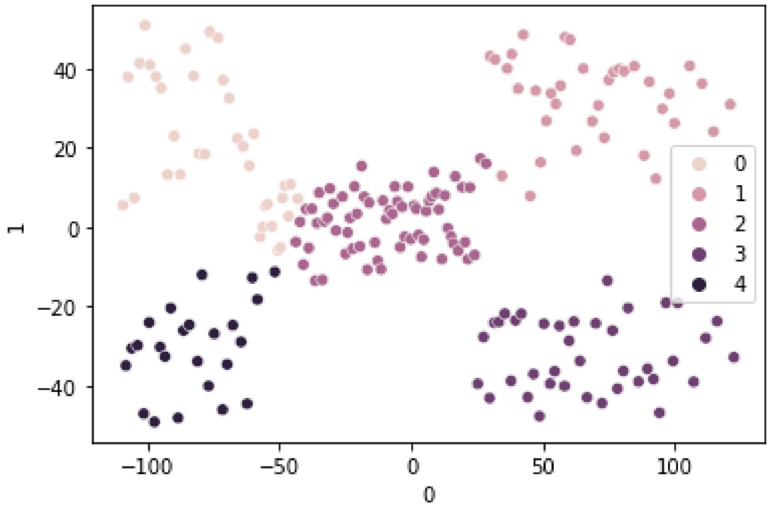
```
Out[53]: <AxesSubplot:xlabel='0', ylabel='1'>
```



```
In [54]: #step 12
```

```
In [55]: sns.scatterplot(PCA_Components[0], PCA_Components[1], hue=KM1.labels_)
```

```
Out[55]: <AxesSubplot:xlabel='0', ylabel='1'>
```



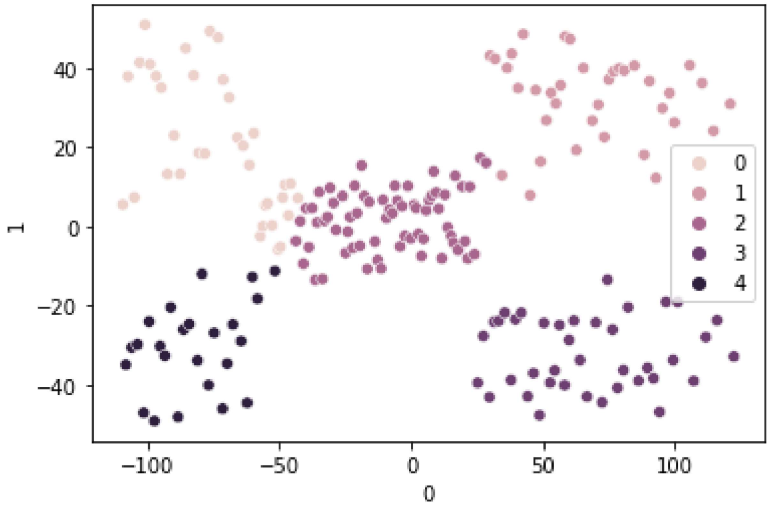
```
In [58]: from sklearn.cluster import MeanShift, AgglomerativeClustering
```

```
In [59]: MS = MeanShift(bandwidth = 50)
         MS.fit(PCA_Components)
         MS.cluster_centers_
```

```
Out[59]: array([[-18.11247356,   1.8729491 ],
                [ 60.80249243,  29.13280327]])
```

In [60]:
```
sns.scatterplot(PCA_Components[0], PCA_Components[1], hue=KM1.labels_)
```

Out[60]:
```
<AxesSubplot:xlabel='0', ylabel='1'>
```



In [61]:
```
#step 13
```

In [62]:
```
AC = AgglomerativeClustering(n_clusters = 5, linkage='ward',compute_full_tree=Tru
AC.fit(df)
```

Out[62]:
```
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 [63]:
```
AC.labels_
```

Out[63]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 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, 0, 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, 4, 2,
       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, 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, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1,
       3, 1])
```

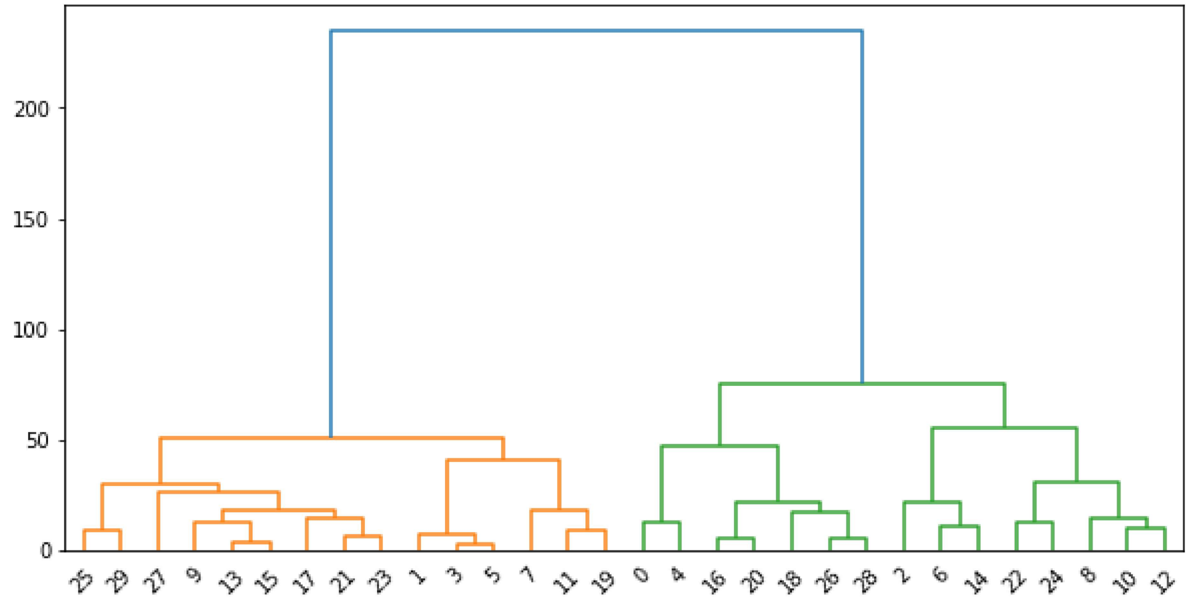
In [64]:
```
df['Cluster'] = AC.labels_
```

In [65]:
```
import scipy.cluster.hierarchy as sch
```

In [66]:
```
from scipy.cluster import hierarchy
```

In [67]:

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



In [68]:

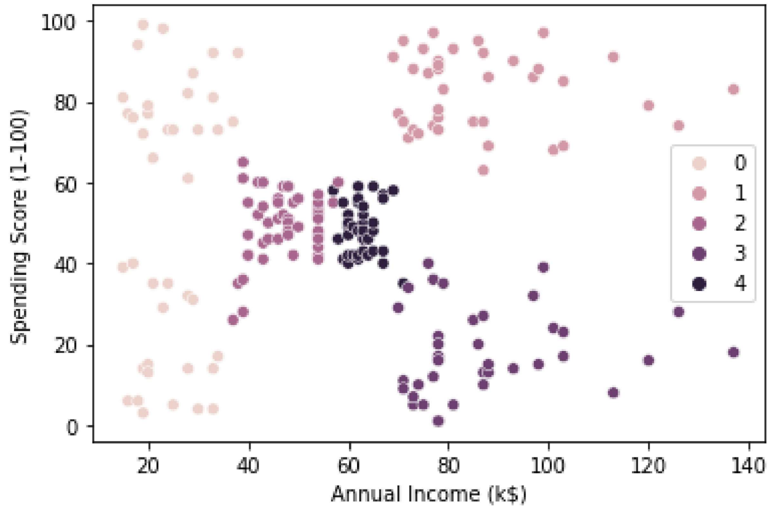
```
#step 14
```

In [69]:

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

Out[69]:

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



In [ ]: