PROGRAM 1

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
import scipy.cluster.hierarchy as shc
from sklearn.decomposition import PCA
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
```

```
data=pd.read_csv("Country-data.csv",index_col=None)
data
```

```
                 country  child_mort  exports  health  imports  income  \
0            Afghanistan        90.2     10.0    7.58     44.9    1610   

1                Albania        16.6     28.0    6.55     48.6    9930   

2                Algeria        27.3     38.4    4.17     31.4   12900   

3                 Angola       119.0     62.3    2.85     42.9    5900   

4    Antigua and Barbuda        10.3     45.5    6.03     58.9   19100   

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

162              Vanuatu        29.2     46.6    5.25     52.7    2950   

163            Venezuela        17.1     28.5    4.91     17.6   16500   

164              Vietnam        23.3     72.0    6.84     80.2    4490   

165                Yemen        56.3     30.0    5.18     34.4    4480   

166               Zambia        83.1     37.0    5.89     30.9    3280   


     inflation  life_expec  total_fer   gdpp
0         9.44        56.2       5.82    553
1         4.49        76.3       1.65   4090
2        16.10        76.5       2.89   4460
3        22.40        60.1       6.16   3530
4         1.44        76.8       2.13  12200
..         ...         ...        ...    ...
```

```
162       2.62        63.0       3.50   2970
163      45.90        75.4       2.47  13500
164      12.10        73.1       1.95   1310
165      23.60        67.5       4.67   1310
166      14.00        52.0       5.40   1460

[167 rows x 10 columns]
```

```
data.shape
```

```
(167, 10)
```

```
data1=data.drop(["country"],axis=1,index=None)
data1
```

```
     child_mort  exports  health  imports  income  inflation
life_expec  \
0          90.2     10.0    7.58     44.9    1610       9.44
56.2
1          16.6     28.0    6.55     48.6    9930       4.49
76.3
2          27.3     38.4    4.17     31.4   12900      16.10
76.5
3         119.0     62.3    2.85     42.9    5900      22.40
60.1
4          10.3     45.5    6.03     58.9   19100       1.44
76.8
..          ...      ...     ...      ...     ...        ...         .
..
162        29.2     46.6    5.25     52.7    2950       2.62
63.0
163        17.1     28.5    4.91     17.6   16500      45.90
75.4
164        23.3     72.0    6.84     80.2    4490      12.10
73.1
165        56.3     30.0    5.18     34.4    4480      23.60
67.5
166        83.1     37.0    5.89     30.9    3280      14.00
52.0

     total_fer   gdpp
0         5.82    553
1         1.65   4090
2         2.89   4460
3         6.16   3530
```

```
4          2.13  12200
..          ...    ...
162        3.50   2970
163        2.47  13500
164        1.95   1310
165        4.67   1310
166        5.40   1460

[167 rows x 9 columns]
```

```
st = StandardScaler()
col = ['child_mort', 'exports', 'health', 'imports', 'income',
'inflation', 'life_expec', 'total_fer', 'gdpp']
data1[col] = st.fit_transform(data1[col])
data1
```

```
     child_mort   exports    health   imports    income  inflation  \
0      1.291532 -1.138280  0.279088 -0.082455 -0.808245   0.157336
1     -0.538949 -0.479658 -0.097016  0.070837 -0.375369  -0.312347
2     -0.272833 -0.099122 -0.966073 -0.641762 -0.220844   0.789274
3      2.007808  0.775381 -1.448071 -0.165315 -0.585043   1.387054
4     -0.695634  0.160668 -0.286894  0.497568  0.101732  -0.601749
..          ...       ...       ...       ...       ...        ...
162   -0.225578  0.200917 -0.571711  0.240700 -0.738527  -0.489784
163   -0.526514 -0.461363 -0.695862 -1.213499 -0.033542   3.616865
164   -0.372315  1.130305  0.008877  1.380030 -0.658404   0.409732
165    0.448417 -0.406478 -0.597272 -0.517472 -0.658924   1.500916
166    1.114951 -0.150348 -0.338015 -0.662477 -0.721358   0.590015

     life_expec  total_fer      gdpp
0     -1.619092   1.902882 -0.679180
1      0.647866  -0.859973 -0.485623
2      0.670423  -0.038404 -0.465376
3     -1.179234   2.128151 -0.516268
4      0.704258  -0.541946 -0.041817
..          ...        ...       ...
162   -0.852161   0.365754 -0.546913
163    0.546361  -0.316678  0.029323
164    0.286958  -0.661206 -0.637754
165   -0.344633   1.140944 -0.637754
166   -2.092785   1.624609 -0.629546

[167 rows x 9 columns]
```

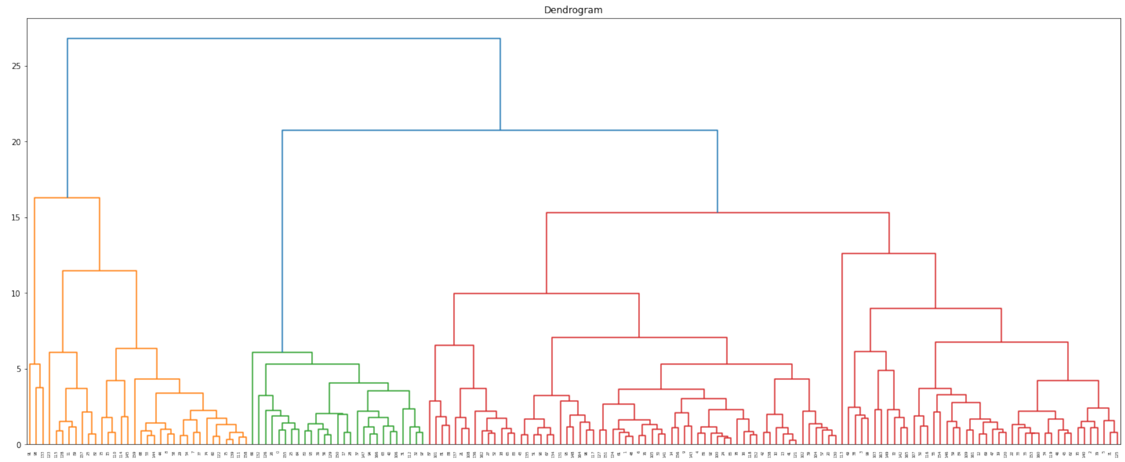
```
data1.columns
data1.set_axis(['child_mort', 'exports', 'health', 'imports',
'income', 'inflation', 'life_expec', 'total_fer',
```

```
'gdpp'],axis='columns',inplace=True)
print(data1.columns)
data1
```

```
Index(['child_mort', 'exports', 'health', 'imports', 'income',
'inflation',
       'life_expec', 'total_fer', 'gdpp'],
      dtype='object')
```

```
     child_mort   exports    health   imports    income  inflation  \
0      1.291532 -1.138280  0.279088 -0.082455 -0.808245   0.157336
1     -0.538949 -0.479658 -0.097016  0.070837 -0.375369  -0.312347
2     -0.272833 -0.099122 -0.966073 -0.641762 -0.220844   0.789274
3      2.007808  0.775381 -1.448071 -0.165315 -0.585043   1.387054
4     -0.695634  0.160668 -0.286894  0.497568  0.101732  -0.601749
..          ...       ...       ...       ...       ...        ...
162   -0.225578  0.200917 -0.571711  0.240700 -0.738527  -0.489784
163   -0.526514 -0.461363 -0.695862 -1.213499 -0.033542   3.616865
164   -0.372315  1.130305  0.008877  1.380030 -0.658404   0.409732
165    0.448417 -0.406478 -0.597272 -0.517472 -0.658924   1.500916
166    1.114951 -0.150348 -0.338015 -0.662477 -0.721358   0.590015

     life_expec  total_fer      gdpp
0     -1.619092   1.902882 -0.679180
1      0.647866  -0.859973 -0.485623
2      0.670423  -0.038404 -0.465376
3     -1.179234   2.128151 -0.516268
4      0.704258  -0.541946 -0.041817
..          ...        ...       ...
162   -0.852161   0.365754 -0.546913
163    0.546361  -0.316678  0.029323
164    0.286958  -0.661206 -0.637754
165   -0.344633   1.140944 -0.637754
166   -2.092785   1.624609 -0.629546

[167 rows x 9 columns]
```

```
plt.figure(figsize=(25,10))
plt.title("Dendrogram")
dendo=shc.dendrogram(shc.linkage(data1,method="ward"))
```




```
z=shc.ward(data1)
z


array([[4.10000000e+01, 1.21000000e+02, 2.70510920e-01,
2.00000000e+00],
       [7.50000000e+01, 1.39000000e+02, 3.45939626e-01,
2.00000000e+00],
       [2.40000000e+01, 8.50000000e+01, 4.23131474e-01,
2.00000000e+00],
       [1.11000000e+02, 1.58000000e+02, 4.94330379e-01,
2.00000000e+00],
       [6.00000000e+01, 1.22000000e+02, 5.44105161e-01,
2.00000000e+00],
       [1.00000000e+02, 1.69000000e+02, 5.58805736e-01,
3.00000000e+00],
       [1.00000000e+00, 4.80000000e+01, 5.59540632e-01,
2.00000000e+00],
       [5.30000000e+01, 1.44000000e+02, 5.68799380e-01,
2.00000000e+00],
       [2.90000000e+01, 5.40000000e+01, 5.90319960e-01,
2.00000000e+00],
       [2.00000000e+01, 1.30000000e+02, 6.04562360e-01,
2.00000000e+00],
       [6.00000000e+00, 7.60000000e+01, 6.13397046e-01,
2.00000000e+00],
       [5.60000000e+01, 1.29000000e+02, 6.27531275e-01,
2.00000000e+00],
       [4.30000000e+01, 1.35000000e+02, 6.39912838e-01,
2.00000000e+00],
       [5.10000000e+01, 9.00000000e+01, 6.49519933e-01,
2.00000000e+00],
       [1.18000000e+02, 1.52000000e+02, 6.51379582e-01,
2.00000000e+00],
       [6.70000000e+01, 1.34000000e+02, 6.67289482e-01,
2.00000000e+00],
```

```
       [8.00000000e+00, 5.80000000e+01, 6.74784770e-01,
2.00000000e+00],
       [9.40000000e+01, 1.66000000e+02, 6.81751502e-01,
2.00000000e+00],
       [1.30000000e+01, 1.67000000e+02, 6.83705937e-01,
3.00000000e+00],
       [2.30000000e+01, 8.20000000e+01, 6.88415143e-01,
2.00000000e+00],
       [1.20000000e+01, 6.90000000e+01, 7.05029503e-01,
2.00000000e+00],
       [3.00000000e+01, 1.41000000e+02, 7.08220976e-01,
2.00000000e+00],
       [2.70000000e+01, 5.20000000e+01, 7.22785225e-01,
2.00000000e+00],
       [1.53000000e+02, 1.60000000e+02, 7.25604863e-01,
2.00000000e+00],
       [6.50000000e+01, 8.30000000e+01, 7.37063496e-01,
2.00000000e+00],
       [4.50000000e+01, 6.20000000e+01, 7.48990755e-01,
2.00000000e+00],
       [4.00000000e+00, 8.60000000e+01, 7.50350440e-01,
2.00000000e+00],
       [3.40000000e+01, 1.19000000e+02, 7.53686516e-01,
2.00000000e+00],
       [9.20000000e+01, 1.72000000e+02, 7.60396964e-01,
4.00000000e+00],
       [3.50000000e+01, 1.90000000e+02, 7.61275522e-01,
3.00000000e+00],
       [1.50000000e+01, 1.10000000e+02, 7.77193159e-01,
2.00000000e+00],
       [1.70000000e+01, 2.80000000e+01, 7.91411283e-01,
2.00000000e+00],
       [4.20000000e+01, 1.38000000e+02, 7.94452442e-01,
2.00000000e+00],
       [1.68000000e+02, 1.70000000e+02, 7.99955525e-01,
4.00000000e+00],
       [7.10000000e+01, 1.25000000e+02, 8.15205386e-01,
2.00000000e+00],
       [3.20000000e+01, 9.70000000e+01, 8.16760749e-01,
2.00000000e+00],
       [7.00000000e+00, 7.70000000e+01, 8.18682770e-01,
2.00000000e+00],
       [1.90000000e+01, 1.20000000e+02, 8.20591059e-01,
2.00000000e+00],
       [6.10000000e+01, 1.73000000e+02, 8.21185399e-01,
3.00000000e+00],
       [4.00000000e+01, 1.06000000e+02, 8.36613603e-01,
2.00000000e+00],
       [1.60000000e+01, 1.81000000e+02, 8.53648471e-01,
3.00000000e+00],
```

```
        [8.00000000e+01, 9.30000000e+01, 8.87057526e-01,
2.00000000e+00],
        [1.15000000e+02, 1.28000000e+02, 9.04313322e-01,
2.00000000e+00],
        [1.62000000e+02, 1.89000000e+02, 9.07564284e-01,
3.00000000e+00],
        [6.80000000e+01, 1.74000000e+02, 9.15551148e-01,
3.00000000e+00],
        [1.27000000e+02, 1.51000000e+02, 9.37233421e-01,
2.00000000e+00],
        [4.60000000e+01, 1.92000000e+02, 9.43773301e-01,
3.00000000e+00],
        [5.70000000e+01, 1.76000000e+02, 9.52163338e-01,
3.00000000e+00],
        [0.00000000e+00, 1.55000000e+02, 9.55543804e-01,
2.00000000e+00],
        [9.60000000e+01, 1.17000000e+02, 9.68471114e-01,
2.00000000e+00],
        [1.09000000e+02, 1.61000000e+02, 9.71200850e-01,
2.00000000e+00],
        [1.24000000e+02, 2.05000000e+02, 9.84206687e-01,
4.00000000e+00],
        [1.05000000e+02, 1.88000000e+02, 9.85751419e-01,
3.00000000e+00],
        [2.50000000e+01, 6.40000000e+01, 9.94635310e-01,
2.00000000e+00],
        [4.40000000e+01, 1.83000000e+02, 9.96052768e-01,
3.00000000e+00],
        [4.70000000e+01, 2.04000000e+02, 1.01077157e+00,
3.00000000e+00],
        [3.60000000e+01, 1.78000000e+02, 1.01654785e+00,
3.00000000e+00],
        [2.10000000e+01, 1.08000000e+02, 1.03595667e+00,
2.00000000e+00],
        [1.80000000e+01, 1.91000000e+02, 1.06347159e+00,
3.00000000e+00],
        [7.00000000e+01, 1.40000000e+02, 1.08544089e+00,
2.00000000e+00],
        [1.40000000e+01, 1.56000000e+02, 1.09343951e+00,
2.00000000e+00],
        [1.04000000e+02, 2.14000000e+02, 1.09376027e+00,
4.00000000e+00],
        [1.00000000e+01, 1.85000000e+02, 1.09943050e+00,
4.00000000e+00],
        [1.80000000e+02, 1.82000000e+02, 1.10945738e+00,
4.00000000e+00],
        [5.90000000e+01, 8.40000000e+01, 1.10949338e+00,
2.00000000e+00],
        [3.30000000e+01, 1.96000000e+02, 1.11556205e+00,
4.00000000e+00],
```

```
       [2.00000000e+00, 7.90000000e+01, 1.11855708e+00,
 2.00000000e+00],
       [1.42000000e+02, 1.65000000e+02, 1.12502528e+00,
 2.00000000e+00],
       [1.93000000e+02, 1.95000000e+02, 1.13700193e+00,
 6.00000000e+00],
       [1.12000000e+02, 2.02000000e+02, 1.14286558e+00,
 3.00000000e+00],
       [9.50000000e+01, 1.48000000e+02, 1.14477708e+00,
 2.00000000e+00],
       [1.10000000e+01, 8.90000000e+01, 1.17036681e+00,
 2.00000000e+00],
       [1.47000000e+02, 1.84000000e+02, 1.18285462e+00,
 3.00000000e+00],
       [5.00000000e+01, 1.16000000e+02, 1.20166926e+00,
 2.00000000e+00],
       [9.00000000e+00, 1.43000000e+02, 1.20366239e+00,
 2.00000000e+00],
       [6.30000000e+01, 2.06000000e+02, 1.20842089e+00,
 3.00000000e+00],
       [8.10000000e+01, 8.80000000e+01, 1.24564805e+00,
 2.00000000e+00],
       [2.20000000e+01, 2.32000000e+02, 1.30059550e+00,
 5.00000000e+00],
       [1.77000000e+02, 2.19000000e+02, 1.30399955e+00,
 5.00000000e+00],
       [1.71000000e+02, 2.00000000e+02, 1.32009873e+00,
 6.00000000e+00],
       [3.90000000e+01, 2.28000000e+02, 1.35688254e+00,
 5.00000000e+00],
       [2.08000000e+02, 2.23000000e+02, 1.35863922e+00,
 5.00000000e+00],
       [2.15000000e+02, 2.20000000e+02, 1.41639134e+00,
 4.00000000e+00],
       [2.11000000e+02, 2.21000000e+02, 1.43556192e+00,
 6.00000000e+00],
       [1.87000000e+02, 2.22000000e+02, 1.48058758e+00,
 5.00000000e+00],
       [1.64000000e+02, 2.16000000e+02, 1.48805541e+00,
 3.00000000e+00],
       [2.09000000e+02, 2.38000000e+02, 1.51779486e+00,
 4.00000000e+00],
       [2.26000000e+02, 2.33000000e+02, 1.51873777e+00,
 4.00000000e+00],
       [1.46000000e+02, 2.31000000e+02, 1.52283309e+00,
 3.00000000e+00],
       [5.00000000e+00, 2.01000000e+02, 1.57904377e+00,
 3.00000000e+00],
       [2.18000000e+02, 2.45000000e+02, 1.60770822e+00,
 9.00000000e+00],
```

```
       [1.75000000e+02, 2.03000000e+02, 1.65191342e+00,
4.00000000e+00],
       [1.79000000e+02, 2.30000000e+02, 1.66293487e+00,
6.00000000e+00],
       [1.94000000e+02, 2.13000000e+02, 1.68498157e+00,
5.00000000e+00],
       [7.80000000e+01, 2.07000000e+02, 1.68772197e+00,
4.00000000e+00],
       [2.17000000e+02, 2.51000000e+02, 1.70123168e+00,
7.00000000e+00],
       [3.00000000e+00, 9.90000000e+01, 1.72000805e+00,
2.00000000e+00],
       [7.40000000e+01, 2.46000000e+02, 1.73092065e+00,
7.00000000e+00],
       [2.10000000e+02, 2.25000000e+02, 1.74033326e+00,
6.00000000e+00],
       [2.39000000e+02, 2.42000000e+02, 1.76631858e+00,
6.00000000e+00],
       [7.20000000e+01, 2.34000000e+02, 1.77796851e+00,
3.00000000e+00],
       [1.37000000e+02, 2.24000000e+02, 1.79373514e+00,
3.00000000e+00],
       [7.30000000e+01, 1.97000000e+02, 1.80862395e+00,
3.00000000e+00],
       [1.14000000e+02, 1.45000000e+02, 1.85026566e+00,
2.00000000e+00],
       [1.01000000e+02, 2.43000000e+02, 1.85083168e+00,
3.00000000e+00],
       [2.60000000e+01, 2.49000000e+02, 1.89427675e+00,
5.00000000e+00],
       [1.07000000e+02, 2.40000000e+02, 1.90522374e+00,
3.00000000e+00],
       [2.37000000e+02, 2.52000000e+02, 1.94331319e+00,
5.00000000e+00],
       [3.80000000e+01, 2.63000000e+02, 1.95023503e+00,
3.00000000e+00],
       [1.50000000e+02, 1.98000000e+02, 2.00471478e+00,
3.00000000e+00],
       [1.99000000e+02, 2.29000000e+02, 2.00988765e+00,
6.00000000e+00],
       [2.48000000e+02, 2.76000000e+02, 2.02521603e+00,
8.00000000e+00],
       [2.27000000e+02, 2.41000000e+02, 2.07686636e+00,
4.00000000e+00],
       [1.57000000e+02, 1.86000000e+02, 2.15169579e+00,
3.00000000e+00],
       [5.50000000e+01, 1.54000000e+02, 2.16839762e+00,
2.00000000e+00],
       [1.02000000e+02, 2.47000000e+02, 2.19445522e+00,
6.00000000e+00],
```

```
        [1.36000000e+02, 2.65000000e+02, 2.20462024e+00,
7.00000000e+00],
        [2.44000000e+02, 2.60000000e+02, 2.21005729e+00,
1.00000000e+01],
        [3.70000000e+01, 2.66000000e+02, 2.21613992e+00,
7.00000000e+00],
        [2.58000000e+02, 2.64000000e+02, 2.24332336e+00,
1.10000000e+01],
        [1.26000000e+02, 2.72000000e+02, 2.27477267e+00,
6.00000000e+00],
        [1.03000000e+02, 1.63000000e+02, 2.28261760e+00,
2.00000000e+00],
        [1.49000000e+02, 2.67000000e+02, 2.30914105e+00,
4.00000000e+00],
        [2.35000000e+02, 2.61000000e+02, 2.31758641e+00,
1.00000000e+01],
        [3.10000000e+01, 2.36000000e+02, 2.36095621e+00,
4.00000000e+00],
        [2.54000000e+02, 2.56000000e+02, 2.39613851e+00,
7.00000000e+00],
        [4.90000000e+01, 2.75000000e+02, 2.48059360e+00,
4.00000000e+00],
        [2.12000000e+02, 2.57000000e+02, 2.65948349e+00,
1.10000000e+01],
        [2.55000000e+02, 2.62000000e+02, 2.79272007e+00,
1.00000000e+01],
        [1.31000000e+02, 2.74000000e+02, 2.84833469e+00,
6.00000000e+00],
        [8.70000000e+01, 2.71000000e+02, 2.88139568e+00,
4.00000000e+00],
        [2.79000000e+02, 2.90000000e+02, 3.02460200e+00,
1.40000000e+01],
        [1.32000000e+02, 2.87000000e+02, 3.23163434e+00,
7.00000000e+00],
        [2.59000000e+02, 2.96000000e+02, 3.24808759e+00,
1.20000000e+01],
        [2.81000000e+02, 2.95000000e+02, 3.30495459e+00,
1.20000000e+01],
        [2.50000000e+02, 2.86000000e+02, 3.36490799e+00,
1.70000000e+01],
        [2.85000000e+02, 2.91000000e+02, 3.53621711e+00,
1.10000000e+01],
        [2.94000000e+02, 2.98000000e+02, 3.62524056e+00,
2.50000000e+01],
        [2.73000000e+02, 3.01000000e+02, 3.67862915e+00,
1.50000000e+01],
        [2.68000000e+02, 2.83000000e+02, 3.69733152e+00,
1.00000000e+01],
        [2.53000000e+02, 2.80000000e+02, 3.70074905e+00,
7.00000000e+00],
```

```
        [9.80000000e+01, 1.33000000e+02, 3.73565123e+00,
2.00000000e+00],
        [2.78000000e+02, 3.03000000e+02, 4.07106758e+00,
1.90000000e+01],
        [2.84000000e+02, 2.92000000e+02, 4.20247901e+00,
1.70000000e+01],
        [2.69000000e+02, 2.70000000e+02, 4.21976491e+00,
5.00000000e+00],
        [2.77000000e+02, 2.82000000e+02, 4.31212057e+00,
1.20000000e+01],
        [1.59000000e+02, 3.02000000e+02, 4.33781196e+00,
1.80000000e+01],
        [2.88000000e+02, 2.89000000e+02, 4.87087345e+00,
6.00000000e+00],
        [2.99000000e+02, 3.09000000e+02, 5.28072755e+00,
2.60000000e+01],
        [9.10000000e+01, 3.08000000e+02, 5.29571403e+00,
3.00000000e+00],
        [3.04000000e+02, 3.12000000e+02, 5.29619557e+00,
3.70000000e+01],
        [6.60000000e+01, 3.15000000e+02, 6.07272321e+00,
2.70000000e+01],
        [1.23000000e+02, 3.07000000e+02, 6.10132332e+00,
8.00000000e+00],
        [2.93000000e+02, 3.14000000e+02, 6.13217570e+00,
1.00000000e+01],
        [3.11000000e+02, 3.13000000e+02, 6.33677966e+00,
2.30000000e+01],
        [2.97000000e+02, 3.06000000e+02, 6.52270233e+00,
1.40000000e+01],
        [3.05000000e+02, 3.10000000e+02, 6.74860282e+00,
3.20000000e+01],
        [3.00000000e+02, 3.17000000e+02, 7.05787015e+00,
4.90000000e+01],
        [3.20000000e+02, 3.23000000e+02, 8.97090036e+00,
4.20000000e+01],
        [3.22000000e+02, 3.24000000e+02, 9.97794492e+00,
6.30000000e+01],
        [3.19000000e+02, 3.21000000e+02, 1.14814128e+01,
3.10000000e+01],
        [1.13000000e+02, 3.25000000e+02, 1.26203312e+01,
4.30000000e+01],
        [3.26000000e+02, 3.28000000e+02, 1.52947385e+01,
1.06000000e+02],
        [3.16000000e+02, 3.27000000e+02, 1.62896463e+01,
3.40000000e+01],
        [3.18000000e+02, 3.29000000e+02, 2.07270249e+01,
1.33000000e+02],
        [3.30000000e+02, 3.31000000e+02, 2.68161016e+01,
1.67000000e+02]])
```

```
data1["cluster_labels"]=shc.cut_tree(z,n_clusters=[3])
data1
```

```
     child_mort   exports    health   imports    income  inflation  \
0      1.291532 -1.138280  0.279088 -0.082455 -0.808245   0.157336
1     -0.538949 -0.479658 -0.097016  0.070837 -0.375369  -0.312347
2     -0.272833 -0.099122 -0.966073 -0.641762 -0.220844   0.789274
3      2.007808  0.775381 -1.448071 -0.165315 -0.585043   1.387054
4     -0.695634  0.160668 -0.286894  0.497568  0.101732  -0.601749
..          ...       ...       ...       ...       ...        ...
162   -0.225578  0.200917 -0.571711  0.240700 -0.738527  -0.489784
163   -0.526514 -0.461363 -0.695862 -1.213499 -0.033542   3.616865
164   -0.372315  1.130305  0.008877  1.380030 -0.658404   0.409732
165    0.448417 -0.406478 -0.597272 -0.517472 -0.658924   1.500916
166    1.114951 -0.150348 -0.338015 -0.662477 -0.721358   0.590015

     life_expec  total_fer      gdpp  cluster_labels
0     -1.619092   1.902882 -0.679180               0
1      0.647866  -0.859973 -0.485623               1
2      0.670423  -0.038404 -0.465376               1
3     -1.179234   2.128151 -0.516268               1
4      0.704258  -0.541946 -0.041817               1
..          ...        ...       ...             ...
162   -0.852161   0.365754 -0.546913               1
163    0.546361  -0.316678  0.029323               1
164    0.286958  -0.661206 -0.637754               1
165   -0.344633   1.140944 -0.637754               1
166   -2.092785   1.624609 -0.629546               0

[167 rows x 10 columns]
```

```
pCA=PCA(n_components=4)
Principalcomponents=pCA.fit_transform(data1)
principalDF = pd.DataFrame(data = Principalcomponents
             , columns = ['pc1', 'pc2',"pc3","pc4"])
principalDF
```

```
          pc1       pc2       pc3       pc4
0   -3.083119  0.099700 -0.679093  1.028106
1    0.404462 -0.555534 -0.412147 -1.157873
2   -0.273313 -0.483219  1.171500 -0.920294
3   -2.851405  1.570133  1.707855  0.835322
4    0.984471  0.179860 -0.297069 -0.855223
..        ...       ...       ...       ...
162 -0.820285  0.634664 -0.366362 -0.671870
```

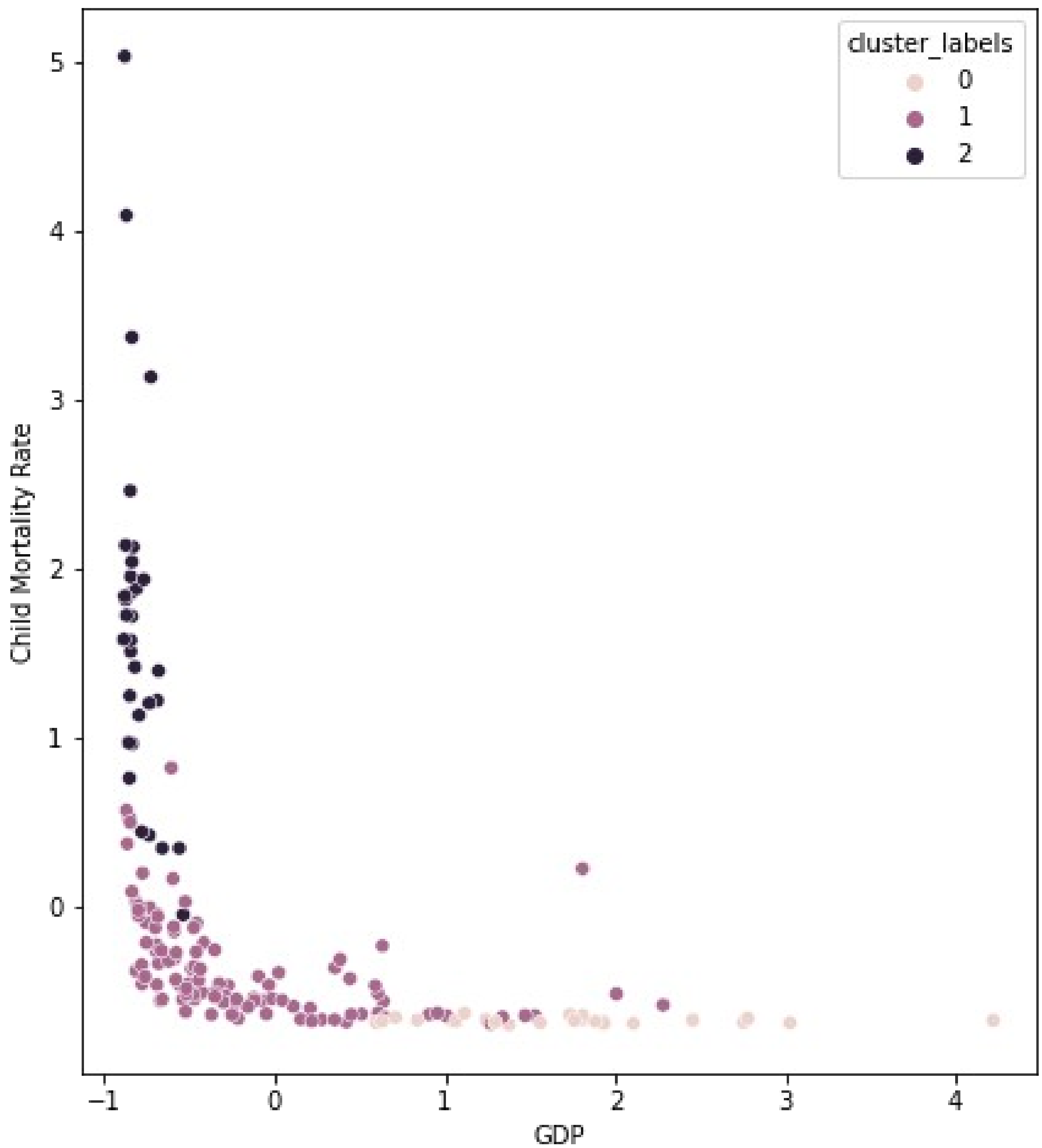
```
163 -0.494878 -1.326853  3.066054 -0.240219
164  0.456988  1.412565 -0.239677 -1.048228
165 -1.826078 -0.188016  1.174267  0.041781
166 -3.031061  0.468798  0.256446  0.800503

[167 rows x 4 columns]
```

```
final=pd.concat([data1,principalDF],axis=1)
final
```

```
     child_mort   exports    health   imports    income  inflation  \
0      1.291532 -1.138280  0.279088 -0.082455 -0.808245   0.157336
1     -0.538949 -0.479658 -0.097016  0.070837 -0.375369  -0.312347
2     -0.272833 -0.099122 -0.966073 -0.641762 -0.220844   0.789274
3      2.007808  0.775381 -1.448071 -0.165315 -0.585043   1.387054
4     -0.695634  0.160668 -0.286894  0.497568  0.101732  -0.601749
..          ...       ...       ...       ...       ...        ...
162   -0.225578  0.200917 -0.571711  0.240700 -0.738527  -0.489784
163   -0.526514 -0.461363 -0.695862 -1.213499 -0.033542   3.616865
164   -0.372315  1.130305  0.008877  1.380030 -0.658404   0.409732
165    0.448417 -0.406478 -0.597272 -0.517472 -0.658924   1.500916
166    1.114951 -0.150348 -0.338015 -0.662477 -0.721358   0.590015

     life_expec  total_fer      gdpp  cluster_labels       pc1
pc2  \
0     -1.619092   1.902882 -0.679180               0 -3.083119
0.099700
1      0.647866  -0.859973 -0.485623               1  0.404462 -
0.555534
2      0.670423  -0.038404 -0.465376               1 -0.273313 -
0.483219
3     -1.179234   2.128151 -0.516268               1 -2.851405
1.570133
4      0.704258  -0.541946 -0.041817               1  0.984471
0.179860
..          ...        ...       ...             ...       ...       .
..
162   -0.852161   0.365754 -0.546913               1 -0.820285
0.634664
163    0.546361  -0.316678  0.029323               1 -0.494878 -
1.326853
164    0.286958  -0.661206 -0.637754               1  0.456988
1.412565
165   -0.344633   1.140944 -0.637754               1 -1.826078 -
0.188016
166   -2.092785   1.624609 -0.629546               0 -3.031061
0.468798

          pc3       pc4
```

```
0   -0.679093  1.028106
1   -0.412147 -1.157873
2    1.171500 -0.920294
3    1.707855  0.835322
4   -0.297069 -0.855223
..        ...       ...
162 -0.366362 -0.671870
163  3.066054 -0.240219
164 -0.239677 -1.048228
165  1.174267  0.041781
166  0.256446  0.800503

[167 rows x 14 columns]
```

```
fig = plt.figure(figsize = (7,8))
sns.scatterplot(x='child_mort',y='gdpp',hue='cluster_labels',legend='f
ull',data=final)
plt.xlabel('GDP')
plt.ylabel('Child Mortality Rate')
plt.show()
```


cluster_labels
0
1
2
Child Mortality Rate
GDP
5
4
3
2
1
0
−1
0
1
2
3
4

PROGRAM 2

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
import scipy.cluster.hierarchy as shc
from sklearn.decomposition import PCA
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
```

```
df=pd.read_csv("Credit Card Customer Data.csv")
df
```

```
     Sl_No  Customer Key  Avg_Credit_Limit  Total_Credit_Cards  \
0        1         87073            100000                   2
1        2         38414             50000                   3
2        3         17341             50000                   7
3        4         40496             30000                   5
4        5         47437            100000                   6
..     ...           ...               ...                 ...
655    656         51108             99000                  10
656    657         60732             84000                  10
657    658         53834            145000                   8
658    659         80655            172000                  10
659    660         80150            167000                   9

     Total_visits_bank  Total_visits_online  Total_calls_made
0                    1                    1                 0
1                    0                   10                 9
2                    1                    3                 4
3                    1                    1                 4
4                    0                   12                 3
..                 ...                  ...               ...
655                  1                   10                 0
656                  1                   13                 2
657                  1                    9                 1
658                  1                   15                 0
659                  0                   12                 2

[660 rows x 7 columns]
```

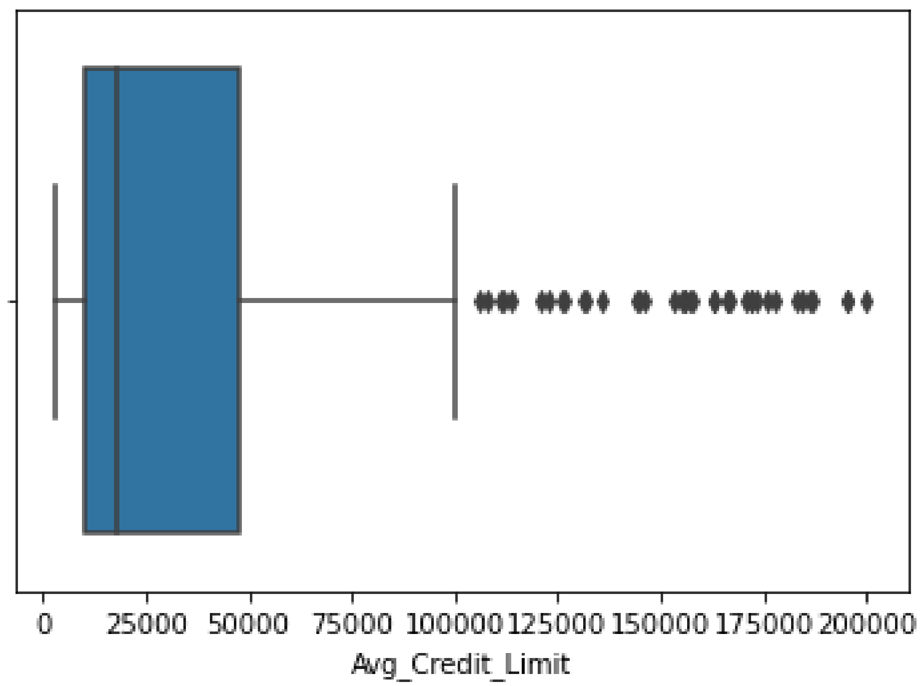
```
df.isna().sum()
```

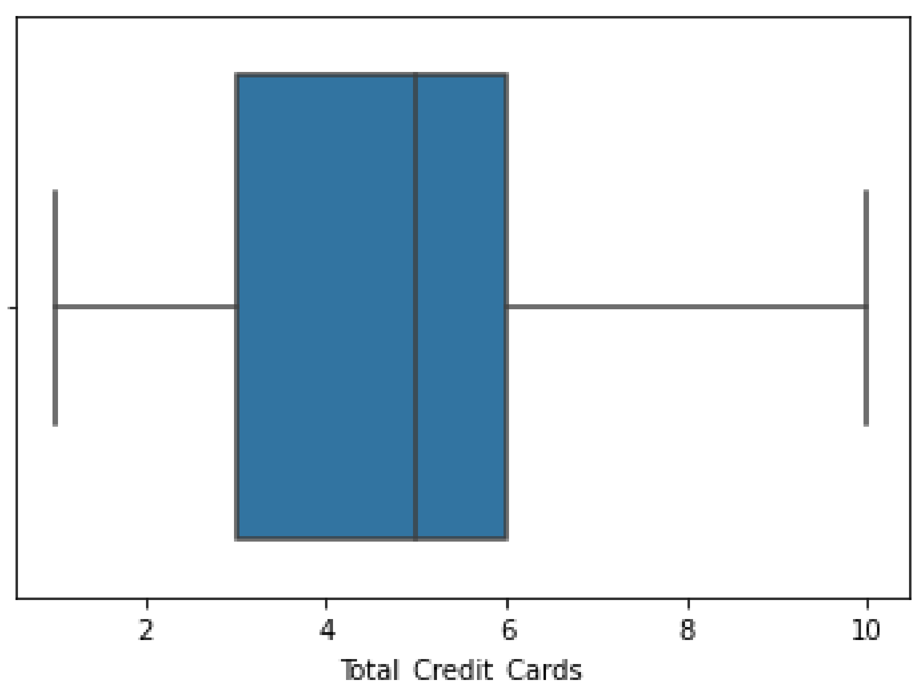
```
Sl_No                  0
Customer Key           0
Avg_Credit_Limit       0
Total_Credit_Cards     0
Total_visits_bank      0
Total_visits_online    0
Total_calls_made       0
dtype: int64
```

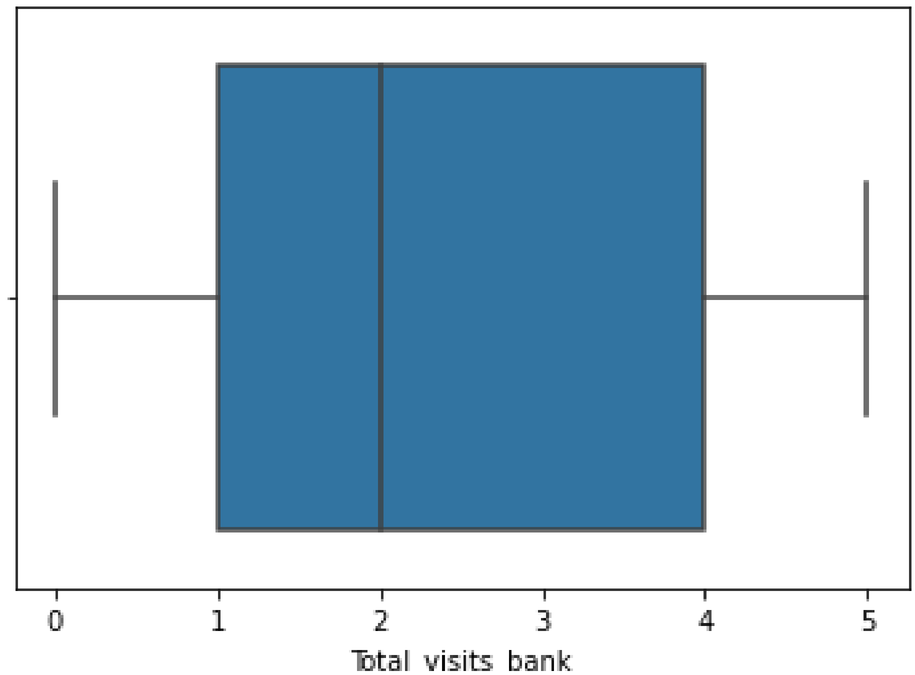
```
df=df.drop(['Sl_No','Customer Key'],axis=1)
df
```

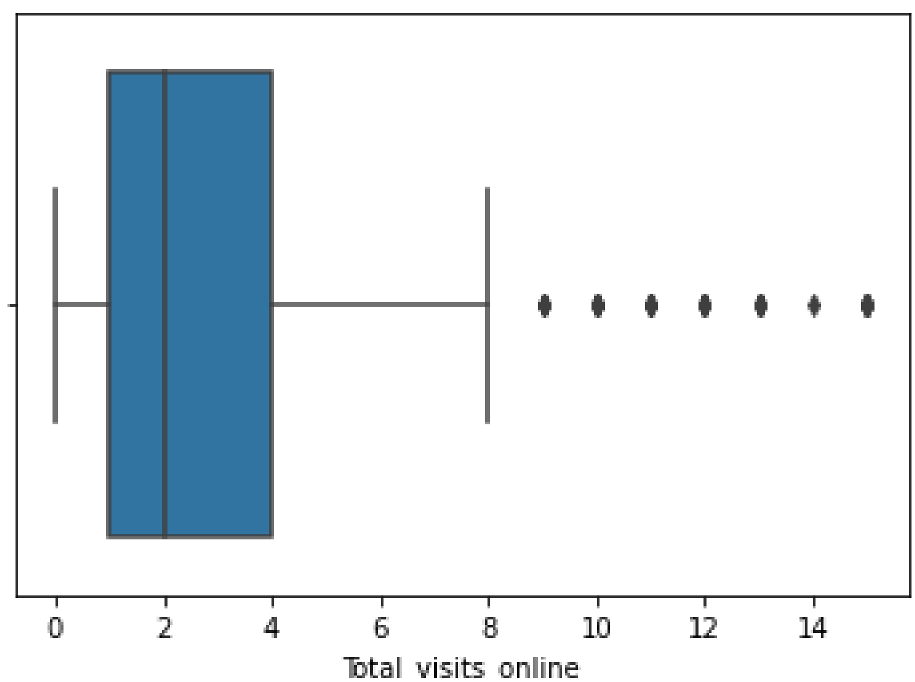
```
     Avg_Credit_Limit  Total_Credit_Cards  Total_visits_bank  \
0              100000                   2                  1
1               50000                   3                  0
2               50000                   7                  1
3               30000                   5                  1
4              100000                   6                  0
..                ...                 ...                ...
655             99000                  10                  1
656             84000                  10                  1
657            145000                   8                  1
658            172000                  10                  1
659            167000                   9                  0

     Total_visits_online  Total_calls_made
0                      1                 0
1                     10                 9
2                      3                 4
3                      1                 4
4                     12                 3
..                   ...               ...
655                   10                 0
656                   13                 2
657                    9                 1
658                   15                 0
659                   12                 2

[660 rows x 5 columns]
```

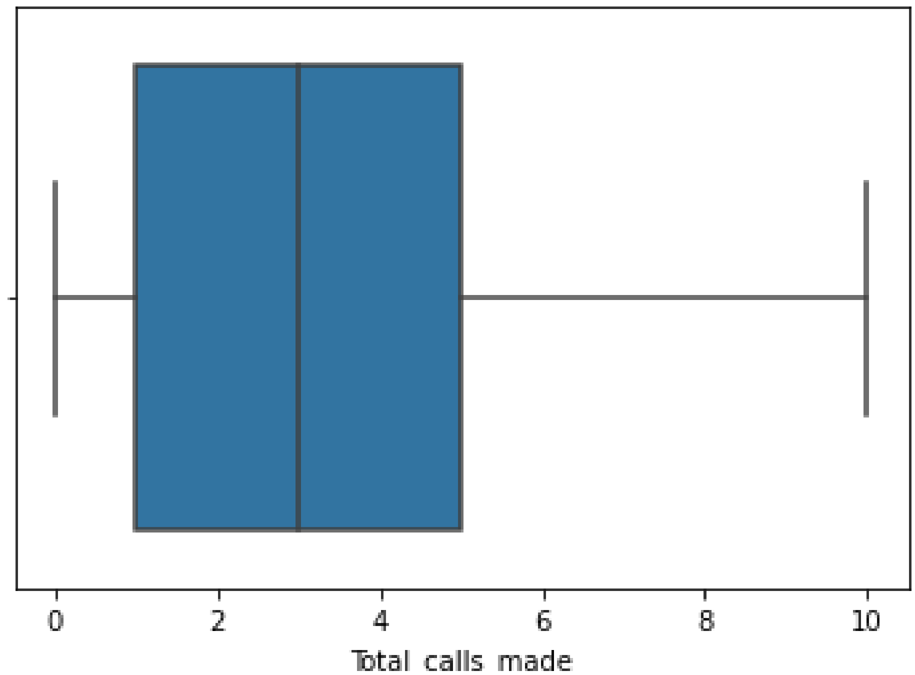
```
for col in df:
    sns.boxplot(x=col,data=df)
    plt.show()
```


0
25000
50000
75000
100000
125000
150000
175000
200000
Avg_Credit_Limit


2
4
6
8
10
Total_Credit_Cards

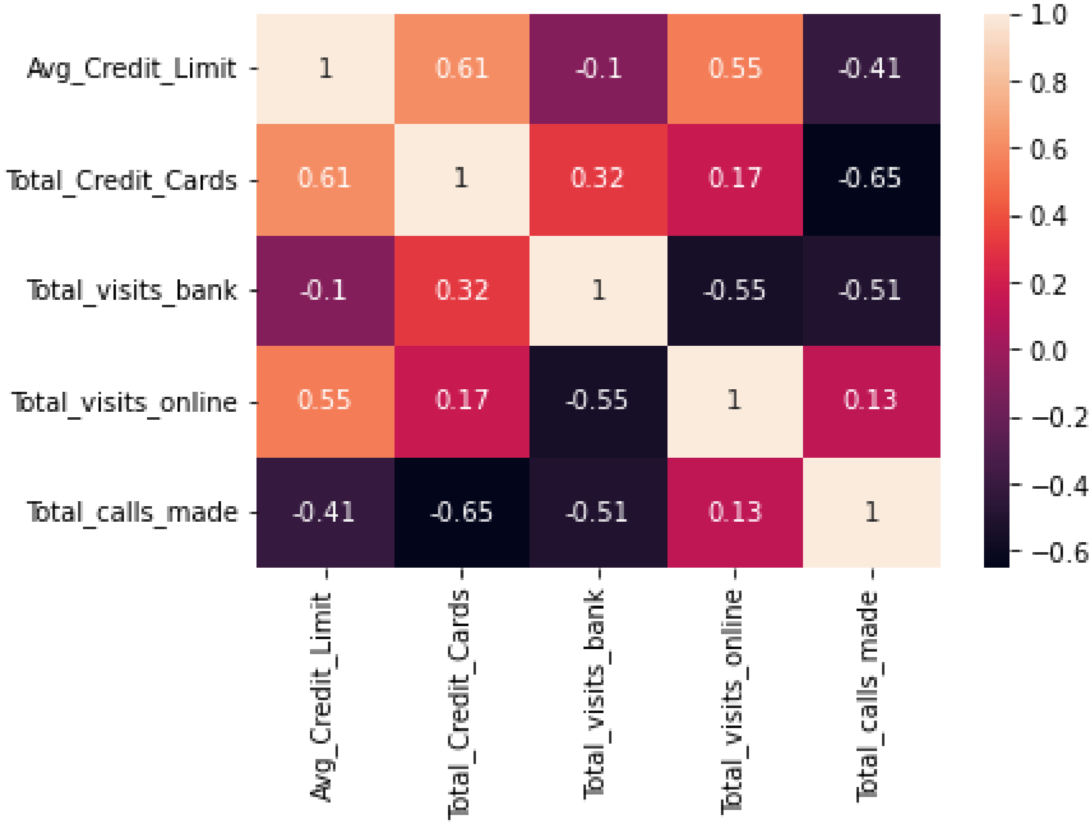

0
1
2
3
4
5
Total_visits_bank


0
2
4
6
8
10
12
14
Total_visits_online




```
sns.heatmap(data=df.corr(),annot=True)
```

```
<AxesSubplot:>
```



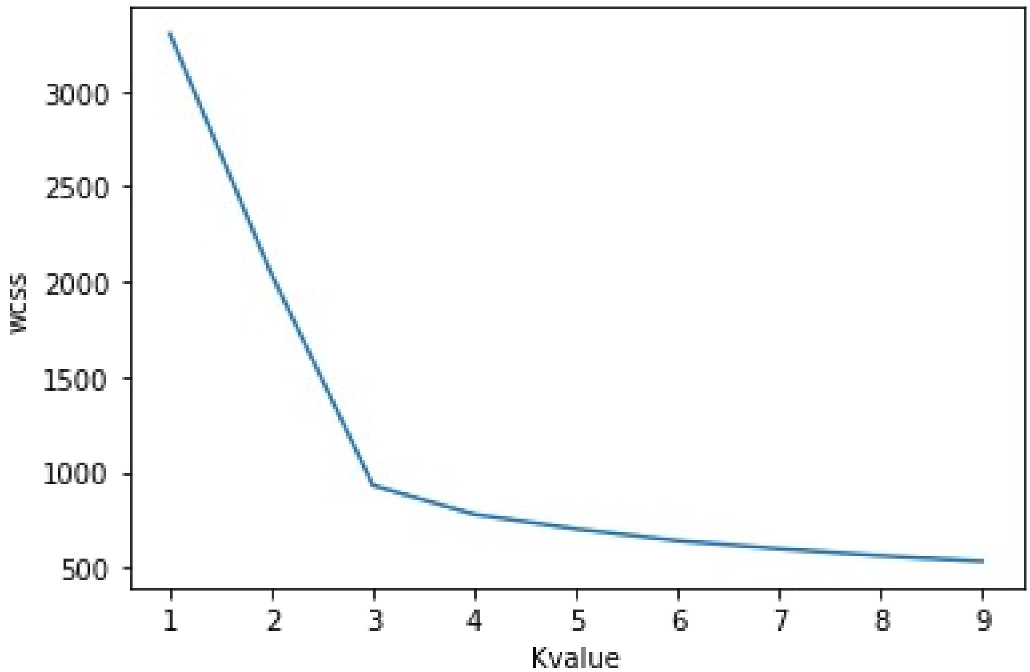

```
stnd=StandardScaler()
df.columns
names=['Avg_Credit_Limit', 'Total_Credit_Cards', 'Total_visits_bank',
       'Total_visits_online', 'Total_calls_made']
df[names]=stnd.fit_transform(df[names])
df
```

```
     Avg_Credit_Limit  Total_Credit_Cards  Total_visits_bank  \
0            1.740187           -1.249225          -0.860451
1            0.410293           -0.787585          -1.473731
2            0.410293            1.058973          -0.860451
3           -0.121665            0.135694          -0.860451
4            1.740187            0.597334          -1.473731
..                ...                 ...                ...
655          1.713589            2.443892          -0.860451
656          1.314621            2.443892          -0.860451
657          2.937092            1.520613          -0.860451
658          3.655235            2.443892          -0.860451
659          3.522245            1.982253          -1.473731

     Total_visits_online  Total_calls_made
```

```
0              -0.547490        -1.251537
1               2.520519         1.891859
2               0.134290         0.145528
3              -0.547490         0.145528
4               3.202298        -0.203739
..                   ...              ...
655             2.520519        -1.251537
656             3.543188        -0.553005
657             2.179629        -0.902271
658             4.224968        -1.251537
659             3.202298        -0.553005

[660 rows x 5 columns]
```

```python
wcss=[]
for i in range(1,10):
    Kmeans = KMeans(n_clusters = i,random_state = 42)
    Kmeans.fit(df)
    Kmeans.inertia_
    wcss.append(Kmeans.inertia_)
plt.plot(range(1, 10), wcss)
plt.xlabel('Kvalue')
plt.ylabel('wcss')
plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\cluster\
_kmeans.py:1036: UserWarning: KMeans is known to have a memory leak on
Windows with MKL, when there are less chunks than available threads.
You can avoid it by setting the environment variable
OMP_NUM_THREADS=3.
  warnings.warn(
```




```
slh_score=[]
for i in range(2,13):
    kme=KMeans(n_clusters=i)
    kme.fit(df)
    slh_score.append(silhouette_score(df,kme.labels_))

print(slh_score)
```

```
[0.4184249666322083, 0.5157182558881671, 0.35566706193741826,
0.2726413988287474, 0.25506309059220267, 0.24792558430237924,
0.24250287477977034, 0.22036530251707592, 0.20551995395799014,
0.21196880307031157, 0.20703017416956548]
```

```
kme1=KMeans(n_clusters=4)
kme1.fit(df)
x=kme1.predict(df)
x
```

```
array([3, 0, 3, 3, 2, 0, 2, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

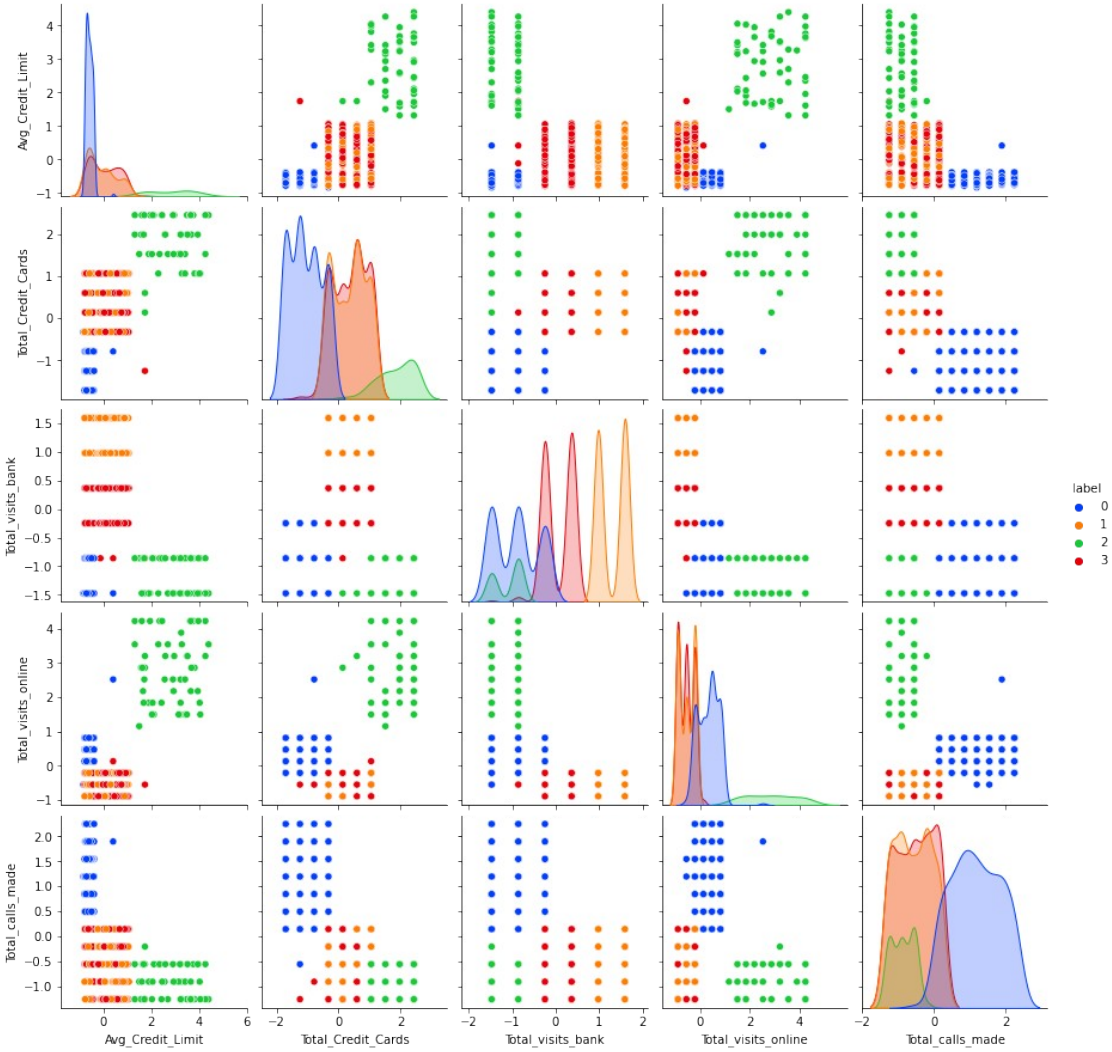
0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
0, 0, 0, 0, 0, 0, 0, 0, 3, 3, 1, 3, 1, 3, 1, 1, 1, 1, 3, 3, 3,
1,
3, 1, 3, 3, 1, 3, 1, 1, 3, 3, 1, 3, 3, 1, 3, 1, 3, 1, 1, 3, 1,
3,
1, 1, 1, 1, 1, 1, 1, 3, 3, 1, 1, 3, 1, 1, 1, 1, 1, 3, 3, 3, 3,
1,
1, 3, 3, 1, 3, 3, 3, 3, 1, 1, 1, 1, 1, 1, 1, 1, 3, 1, 1, 1, 3,
3,
1, 1, 1, 3, 3, 3, 3, 3, 1, 1, 1, 3, 1, 1, 3, 3, 1, 1, 1, 1, 1,
1,
3, 1, 3, 1, 1, 1, 1, 3, 3, 3, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3,
3,
3, 1, 1, 1, 1, 1, 3, 3, 3, 3, 1, 3, 1, 1, 1, 3, 3, 1, 3, 3, 1,
1,
1, 3, 1, 3, 1, 3, 3, 3, 3, 3, 1, 1, 3, 3, 1, 1, 3, 1, 1, 3, 1,
1,
1, 3, 3, 3, 3, 1, 3, 3, 3, 3, 3, 1, 3, 3, 3, 1, 1, 1, 3, 3, 3,
1,
3, 1, 3, 1, 3, 3, 1, 3, 3, 1, 1, 1, 1, 1, 3, 3, 3, 1, 3, 3, 3,
1,
3, 3, 3, 1, 1, 1, 1, 3, 1, 3, 3, 1, 1, 1, 3, 3, 1, 3, 3, 1, 1,
1,
1, 1, 3, 1, 3, 1, 1, 3, 1, 3, 3, 3, 3, 1, 1, 1, 3, 3, 3, 3, 3,
1,
1, 3, 3, 1, 1, 1, 3, 1, 1, 3, 3, 3, 3, 3, 1, 1, 1, 3, 3, 3, 1,
3,
1, 3, 3, 1, 3, 3, 3, 3, 3, 1, 1, 3, 1, 3, 1, 3, 1, 3, 3, 1, 3,
3,
3, 3, 3, 3, 3, 3, 1, 1, 1, 3, 1, 1, 3, 1, 1, 3, 3, 1, 3, 1, 3,
1,
1, 3, 3, 1, 3, 1, 1, 1, 1, 1, 3, 1, 3, 1, 1, 1, 1, 3, 1, 1, 3,
3,
1, 3, 3, 3, 3, 3, 1, 3, 1, 1, 1, 3, 1, 1, 3, 3, 3, 1, 3, 3, 1,
1,
3, 1, 1, 1, 1, 3, 1, 1, 1, 3, 3, 3, 3, 3, 1, 3, 3, 1, 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])
```

```
df["label"]=x
df
```

```
     Avg_Credit_Limit  Total_Credit_Cards  Total_visits_bank  \
0            1.740187           -1.249225          -0.860451
1            0.410293           -0.787585          -1.473731
2            0.410293            1.058973          -0.860451
3           -0.121665            0.135694          -0.860451
4            1.740187            0.597334          -1.473731
..                ...                 ...                ...
655          1.713589            2.443892          -0.860451
656          1.314621            2.443892          -0.860451
657          2.937092            1.520613          -0.860451
658          3.655235            2.443892          -0.860451
659          3.522245            1.982253          -1.473731

     Total_visits_online  Total_calls_made  label
0              -0.547490         -1.251537      3
1               2.520519          1.891859      0
2               0.134290          0.145528      3
3              -0.547490          0.145528      3
4               3.202298         -0.203739      2
..                   ...               ...    ...
655             2.520519         -1.251537      2
656             3.543188         -0.553005      2
657             2.179629         -0.902271      2
658             4.224968         -1.251537      2
659             3.202298         -0.553005      2

[660 rows x 6 columns]
```

```
sns.pairplot(df,hue='label',palette='bright')
plt.show()
```




```
centroid=kme.cluster_centers_
centroid
```

```
array([[ 0.64570368,  0.17014478, -0.01833461, -0.56784133, -
0.46438509],
       [ 3.06776829,  1.96218122, -1.15375852,  3.72104358, -
0.90227113],
       [-0.60400761, -0.50349924, -0.91941987,  0.29162368,
0.79704348],
       [-0.35395338,  0.8004551 ,  1.33100376, -0.52930891, -
0.55766177],
       [-0.58687726, -1.33632657, -1.03402048,  0.3336783 ,
0.35640533],
       [ 2.63072348,  1.77707934, -1.06487734,  2.06599885, -
0.85052798],
       [-0.62497902, -1.22259171, -0.32972743,  0.25229024,
1.39482608],
       [ 0.74818421,  0.59733368,  1.03617501, -0.54748969, -
```

```
0.65649118],
       [-0.30967215, -0.17417362,  1.29863153, -0.54748969, -
0.54343595],
       [-0.57382924, -1.14587261, -1.22658803,  0.39377321,
1.85536831],
       [-0.37861976,  0.20988615,  0.09232388, -0.56575165, -
1.00829838],
       [-0.35145347,  0.68058016,  0.04438893, -0.56984313, -
0.14648188]])
```

```
sch_score=silhouette_score(df,kme1.labels_)
sch_score
```

```
0.5392195186716479
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
import scipy.cluster.hierarchy as shc
from sklearn.decomposition import PCA
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
```

```
data=pd.read_csv("Mall_Customers.csv")
data
```

```
     CustomerID  Gender  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]
```

```
cormatrix=data.corr().abs()
cormatrix
```

```
                        CustomerID       Age  Annual Income (k$)  \
CustomerID                1.000000  0.026763            0.977548
Age                       0.026763  1.000000            0.012398
Annual Income (k$)        0.977548  0.012398            1.000000
```

```
Spending Score (1-100)    0.013835  0.327227            0.009903

                        Spending Score (1-100)
CustomerID                            0.013835
Age                                   0.327227
Annual Income (k$)                    0.009903
Spending Score (1-100)                1.000000
```

```
uppertri=cormatrix.where(np.triu(np.ones(cormatrix.shape),k=1).astype(
np.bool))
uppertri
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_2548\1562843034.py:1:
DeprecationWarning: `np.bool` is a deprecated alias for the builtin
`bool`. To silence this warning, use `bool` by itself. Doing this will
not modify any behavior and is safe. If you specifically wanted the
numpy scalar type, use `np.bool_` here.
Deprecated in NumPy 1.20; for more details and guidance:
https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations

uppertri=cormatrix.where(np.triu(np.ones(cormatrix.shape),k=1).astype(
np.bool))

                        CustomerID       Age  Annual Income (k$)  \
CustomerID                     NaN  0.026763            0.977548
Age                            NaN       NaN            0.012398
Annual Income (k$)             NaN       NaN                 NaN
Spending Score (1-100)         NaN       NaN                 NaN

                        Spending Score (1-100)
CustomerID                            0.013835
Age                                   0.327227
Annual Income (k$)                    0.009903
Spending Score (1-100)                     NaN
```

```
 to_drop = [column for column in uppertri.columns if
any(uppertri[column] <0.05)]
 to_drop
```

```
['Age', 'Annual Income (k$)', 'Spending Score (1-100)']
```

```
malldf=pd.get_dummies(data,columns=['Gender'])
malldf
```

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

     Gender_Female  Gender_Male
0                0            1
1                0            1
2                1            0
3                1            0
4                1            0
..             ...          ...
195              1            0
196              1            0
197              0            1
198              0            1
199              0            1

[200 rows x 6 columns]
```

```
from sklearn.cluster import DBSCAN
```

```
modeldbs= DBSCAN(eps=12.5,min_samples=4)
modeldbs
modeldbs.fit(malldf)
```

```
DBSCAN(eps=12.5, min_samples=4)
```

```
label=modeldbs.labels_
labels=pd.DataFrame(label,columns=['labels'])
labels
```

```
     labels
0        -1
1         0
2        -1
3         0
4        -1
..      ...
195      -1
```

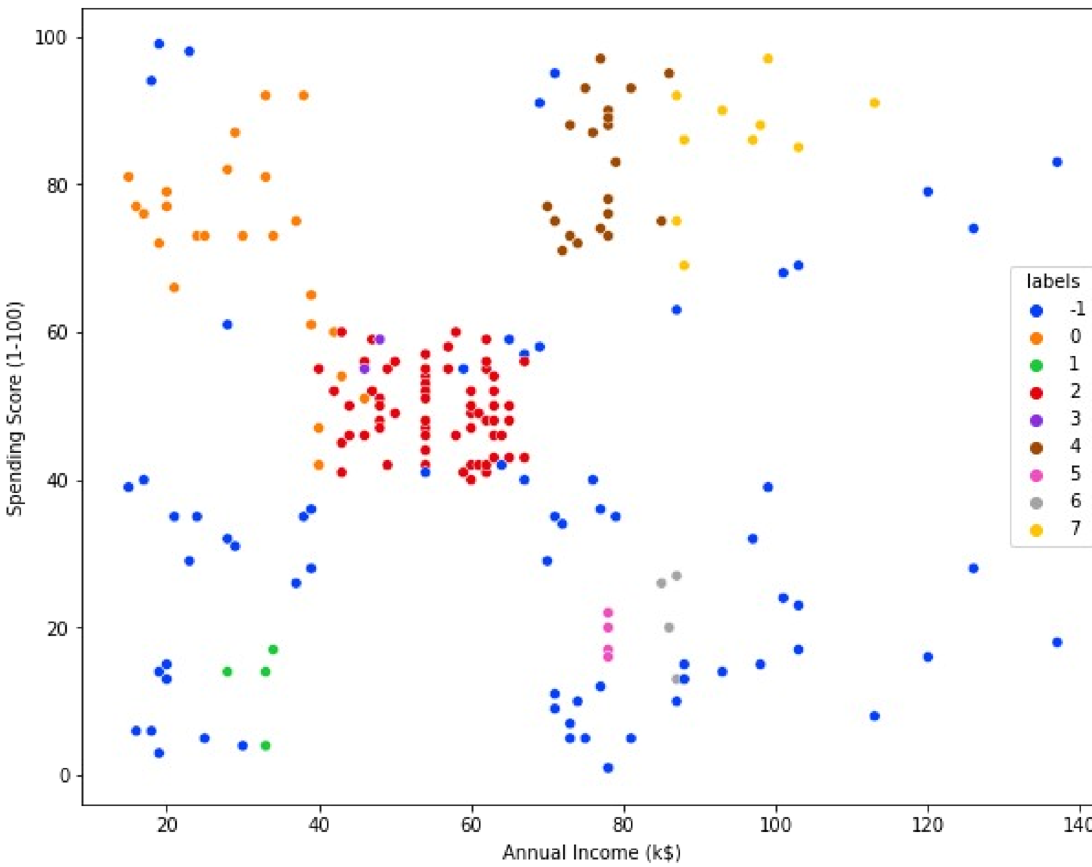
```
196      -1
197      -1
198      -1
199      -1

[200 rows x 1 columns]
```

```
finaldf=pd.concat([malldf,labels],axis=1)
finaldf
```

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

     Gender_Female  Gender_Male  labels
0                0            1      -1
1                0            1       0
2                1            0      -1
3                1            0       0
4                1            0      -1
..             ...          ...     ...
195              1            0      -1
196              1            0      -1
197              0            1      -1
198              0            1      -1
199              0            1      -1

[200 rows x 7 columns]
```

```
plt.figure(figsize=(10,8))
sns.scatterplot(x='Annual Income (k$)',y='Spending Score (1-
100)',data=finaldf,hue="labels",palette='bright')
```

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



labels
-1
0
1
2
3
4
5
6
7
Spending Score (1-100)
Annual Income (k$)
0
20
40
60
80
100
20
40
60
80
100
120
140

PROGRAM 4

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
import scipy.cluster.hierarchy as shc
from sklearn.decomposition import PCA
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
```

```
!pip install apyori
```

```
Defaulting to user installation because normal site-packages is not
writeable
Requirement already satisfied: apyori in c:\users\user\appdata\
roaming\python\python39\site-packages (1.1.2)
```

```
data=pd.read_csv("Groceries data.csv")
data
```

```
       Member_number        Date        itemDescription  year  month
day  \
0               1808  2015-07-21         tropical fruit  2015      7
21
1               2552  2015-05-01             whole milk  2015      5
1
2               2300  2015-09-19              pip fruit  2015      9
19
3               1187  2015-12-12       other vegetables  2015     12
12
4               3037  2015-01-02             whole milk  2015      1
2
...              ...         ...                    ...   ...    ...
...
38760           4471  2014-08-10         sliced cheese  2014      8
10
38761           2022  2014-02-23                  candy  2014      2
23
38762           1097  2014-04-16               cake bar  2014      4
```

```
16
38763           1510  2014-03-12  fruit/vegetable juice  2014      3
12
38764           1521  2014-12-26               cat food  2014     12
26

       day_of_week
0                1
1                4
2                5
3                5
4                4
...            ...
38760            6
38761            6
38762            2
38763            2
38764            4

[38765 rows x 7 columns]
```

```
data.loc[(data["Member_number"] == 1001 )]
```

```
       Member_number        Date     itemDescription  year  month  day
\
364             1001  2015-01-20         frankfurter  2015      1   20

5695            1001  2015-02-05         frankfurter  2015      2    5

6612            1001  2015-04-14                beef  2015      4   14

9391            1001  2014-07-02             sausage  2014      7    2

11046           1001  2014-12-12          whole milk  2014     12   12

16513           1001  2015-01-20                soda  2015      1   20

21844           1001  2015-02-05                curd  2015      2    5

22761           1001  2015-04-14         white bread  2015      4   14

25540           1001  2014-07-02          whole milk  2014      7    2

27195           1001  2014-12-12                soda  2014     12   12

32575           1001  2015-01-20  whipped/sour cream  2015      1   20
```

```
32727           1001  2014-07-02          rolls/buns  2014      7    2


       day_of_week
364              1
5695             3
6612             1
9391             2
11046            4
16513            1
21844            3
22761            1
25540            2
27195            4
32575            1
32727            2
```

```
value=1
data["item_count"]=value
data
```

```
       Member_number        Date        itemDescription  year  month
day  \
0               1808  2015-07-21         tropical fruit  2015      7
21
1               2552  2015-05-01             whole milk  2015      5
1
2               2300  2015-09-19              pip fruit  2015      9
19
3               1187  2015-12-12       other vegetables  2015     12
12
4               3037  2015-01-02             whole milk  2015      1
2
...              ...         ...                    ...   ...    ...
...
38760           4471  2014-08-10          sliced cheese  2014      8
10
38761           2022  2014-02-23                  candy  2014      2
23
38762           1097  2014-04-16               cake bar  2014      4
16
38763           1510  2014-03-12  fruit/vegetable juice  2014      3
12
38764           1521  2014-12-26               cat food  2014     12
26

       day_of_week  item_count
0                1           1
1                4           1
```

```
2                 5           1
3                 5           1
4                 4           1
...             ...         ...
38760             6           1
38761             6           1
38762             2           1
38763             2           1
38764             4           1

[38765 rows x 8 columns]
```

```
groc=data.groupby(["Member_number","itemDescription"])
["itemDescription"].count().unstack().fillna(0)
groc
```

```
itemDescription  Instant food products  UHT-milk  abrasive cleaner  \
Member_number
1000                               0.0       0.0               0.0
1001                               0.0       0.0               0.0
1002                               0.0       0.0               0.0
1003                               0.0       0.0               0.0
1004                               0.0       0.0               0.0
...                                ...       ...               ...
4996                               0.0       0.0               0.0
4997                               0.0       0.0               0.0
4998                               0.0       0.0               0.0
4999                               0.0       0.0               0.0
5000                               0.0       0.0               0.0

itemDescription  artif. sweetener  baby cosmetics  bags  baking powder
\
Member_number

1000                          0.0             0.0   0.0            0.0

1001                          0.0             0.0   0.0            0.0

1002                          0.0             0.0   0.0            0.0

1003                          0.0             0.0   0.0            0.0

1004                          0.0             0.0   0.0            0.0

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

4996                          0.0             0.0   0.0            0.0
```

```
4997                            0.0              0.0   0.0              0.0

4998                            0.0              0.0   0.0              0.0

4999                            0.0              0.0   0.0              0.0

5000                            0.0              0.0   0.0              0.0


itemDescription  bathroom cleaner  beef  berries  ...  turkey  vinegar
\
Member_number                                     ...

1000                            0.0   0.0      0.0  ...     0.0      0.0

1001                            0.0   1.0      0.0  ...     0.0      0.0

1002                            0.0   0.0      0.0  ...     0.0      0.0

1003                            0.0   0.0      0.0  ...     0.0      0.0

1004                            0.0   0.0      0.0  ...     0.0      0.0

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

4996                            0.0   0.0      0.0  ...     0.0      0.0

4997                            0.0   0.0      0.0  ...     0.0      0.0

4998                            0.0   0.0      0.0  ...     0.0      0.0

4999                            0.0   0.0      2.0  ...     0.0      0.0

5000                            0.0   0.0      0.0  ...     0.0      0.0


itemDescription  waffles  whipped/sour cream  whisky  white bread
white wine  \
Member_number

1000                 0.0                 0.0     0.0          0.0
0.0
1001                 0.0                 1.0     0.0          1.0
0.0
1002                 0.0                 0.0     0.0          0.0
0.0
1003                 0.0                 0.0     0.0          0.0
0.0
```

```
1004                    0.0                 0.0     0.0         0.0
0.0
...                     ...                 ...     ...         ...
...
4996                    0.0                 0.0     0.0         0.0
0.0
4997                    0.0                 0.0     0.0         0.0
1.0
4998                    0.0                 0.0     0.0         0.0
0.0
4999                    0.0                 1.0     0.0         0.0
0.0
5000                    0.0                 0.0     0.0         0.0
0.0

itemDescription  whole milk  yogurt  zwieback
Member_number
1000                    2.0     1.0       0.0
1001                    2.0     0.0       0.0
1002                    1.0     0.0       0.0
1003                    0.0     0.0       0.0
1004                    3.0     0.0       0.0
...                     ...     ...       ...
4996                    0.0     0.0       0.0
4997                    1.0     0.0       0.0
4998                    0.0     0.0       0.0
4999                    0.0     1.0       0.0
5000                    0.0     0.0       0.0

[3898 rows x 167 columns]
```

```python
def encode(x):
    if x <= 0:
        return 0
    if x >= 1:
        return 1
sets = groc.applymap(encode)
sets
```

```
itemDescription  Instant food products  UHT-milk  abrasive cleaner  \
Member_number
1000                                 0         0                 0
1001                                 0         0                 0
1002                                 0         0                 0
1003                                 0         0                 0
1004                                 0         0                 0
...                                ...       ...               ...
4996                                 0         0                 0
4997                                 0         0                 0
```

```
4998                                 0         0                  0
4999                                 0         0                  0
5000                                 0         0                  0

itemDescription  artif. sweetener  baby cosmetics  bags  baking powder
\
Member_number

1000                            0               0     0              0

1001                            0               0     0              0

1002                            0               0     0              0

1003                            0               0     0              0

1004                            0               0     0              0

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

4996                            0               0     0              0

4997                            0               0     0              0

4998                            0               0     0              0

4999                            0               0     0              0

5000                            0               0     0              0


itemDescription  bathroom cleaner  beef  berries  ...  turkey  vinegar
\
Member_number                                     ...

1000                            0     0        0  ...       0        0

1001                            0     1        0  ...       0        0

1002                            0     0        0  ...       0        0

1003                            0     0        0  ...       0        0

1004                            0     0        0  ...       0        0

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

4996                            0     0        0  ...       0        0
```

```
4997                                0     0        0  ...       0        0

4998                                0     0        0  ...       0        0

4999                                0     0        1  ...       0        0

5000                                0     0        0  ...       0        0


itemDescription  waffles  whipped/sour cream  whisky  white bread
white wine  \
Member_number

1000                   0                   0       0            0
0
1001                   0                   1       0            1
0
1002                   0                   0       0            0
0
1003                   0                   0       0            0
0
1004                   0                   0       0            0
0
...                  ...                 ...     ...          ...
...
4996                   0                   0       0            0
0
4997                   0                   0       0            0
1
4998                   0                   0       0            0
0
4999                   0                   1       0            0
0
5000                   0                   0       0            0
0

itemDescription  whole milk  yogurt  zwieback
Member_number
1000                      1       1         0
1001                      1       0         0
1002                      1       0         0
1003                      0       0         0
1004                      1       0         0
...                     ...     ...       ...
4996                      0       0         0
4997                      1       0         0
4998                      0       0         0
4999                      0       1         0
```

```
5000                          0        0          0

[3898 rows x 167 columns]
```