In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import skew
from sklearn.preprocessing import StandardScaler
from scipy.cluster import hierarchy as hi
from sklearn.cluster import AgglomerativeClustering
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings("ignore")
```

# load the dataset

In [2]:

```
df = pd.read_csv("Mall_Customers.csv")
df.head()
```

Out[2]:

| | 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 [3]:

```
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    object
 2   Age                     200 non-null    int64
 3   Annual Income (k$)      200 non-null    int64
 4   Spending Score (1-100)  200 non-null    int64
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
```

In [4]:

```
df.describe()
```

Out[4]:

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

In [5]:

```
df.corr().style.background_gradient()
```

Out[5]:

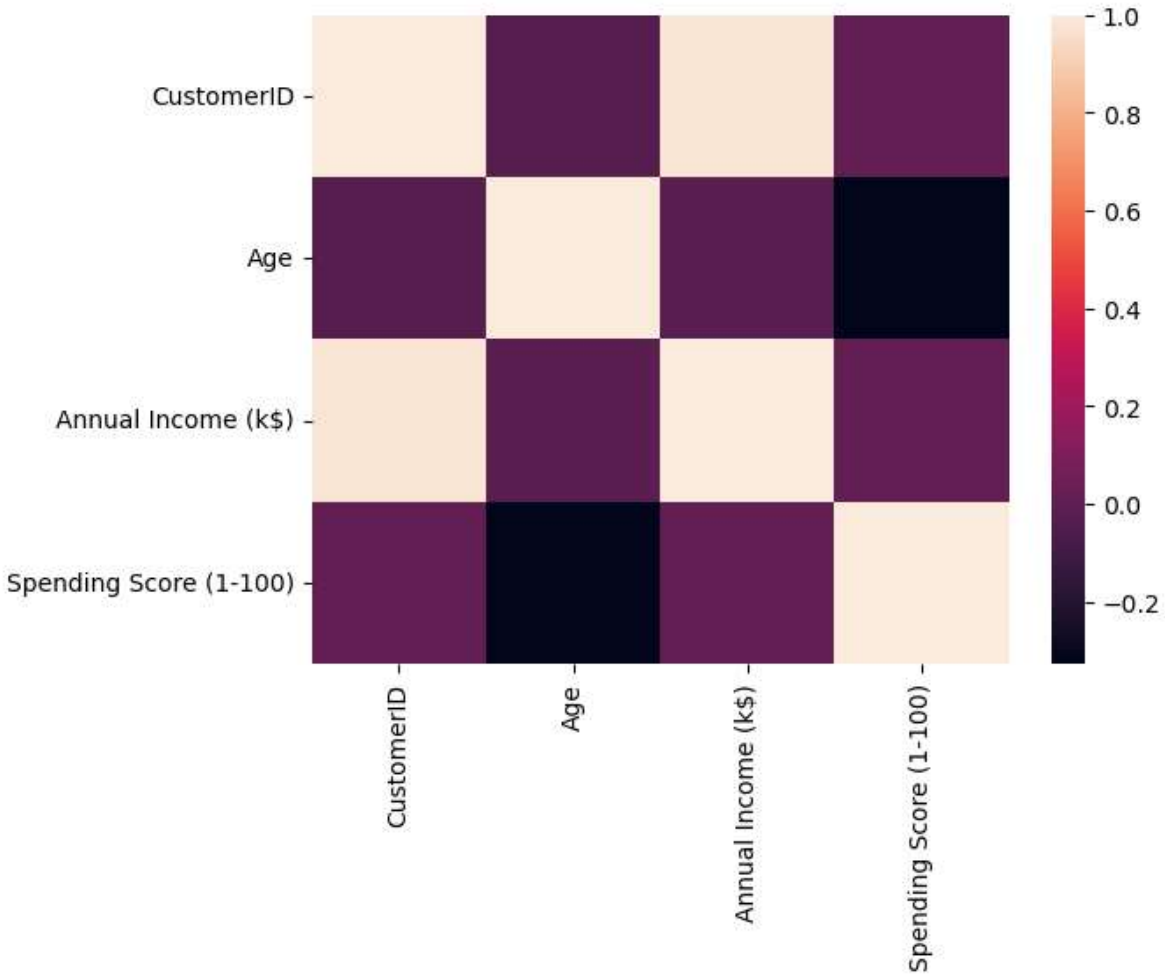
| | CustomerID | Age | Annual Income (k$) | Spending Score (1-100) |
|---|---|---|---|---|
| **CustomerID** | 1.000000 | -0.026763 | 0.977548 | 0.013835 |
| **Age** | -0.026763 | 1.000000 | -0.012398 | -0.327227 |
| **Annual Income (k$)** | 0.977548 | -0.012398 | 1.000000 | 0.009903 |
| **Spending Score (1-100)** | 0.013835 | -0.327227 | 0.009903 | 1.000000 |

In [6]:

```
sns.heatmap(df.corr())
```

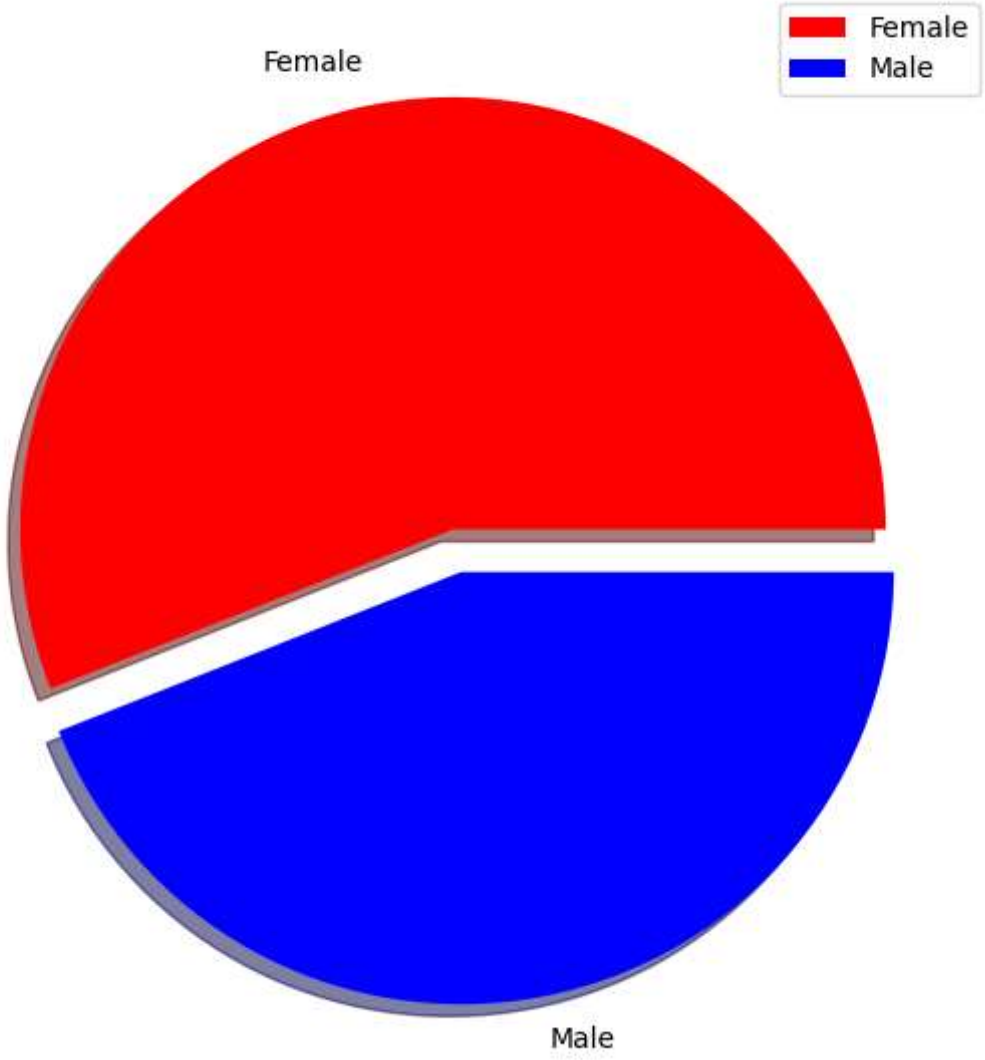
Out[6]:

```
<AxesSubplot:>
```



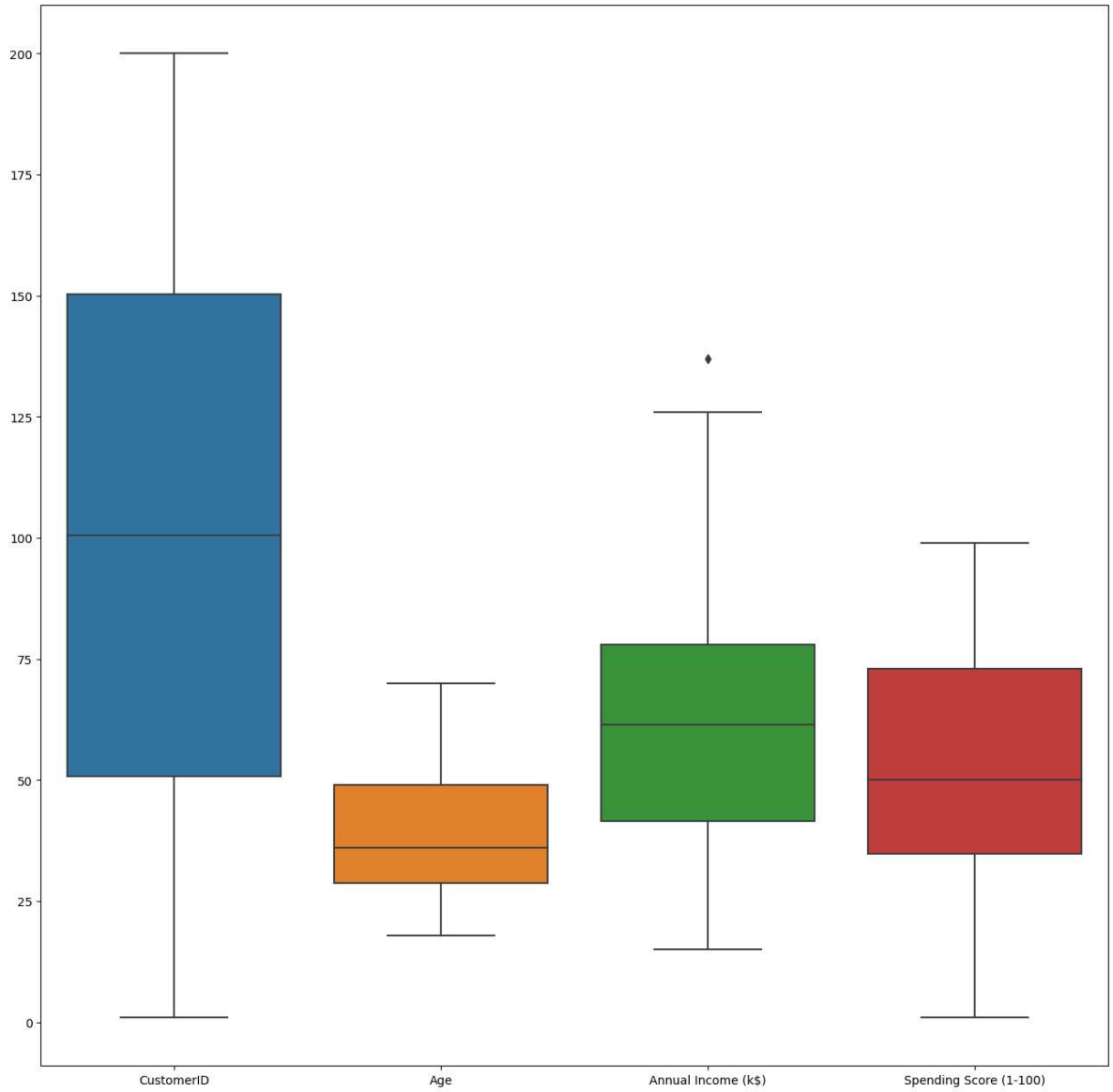
In [7]:

```
plt.figure(figsize=(7,7))
size=df['Genre'].value_counts()
label=['Female','Male']
color=['Red','Blue']
explode=[0,0.1]
plt.pie(size,explode=explode,labels=label,colors=color,shadow=True)
plt.legend()
plt.show()
```



In [8]:

```
plt.figure(figsize=(16,16))
sns.boxplot(data=df);
```

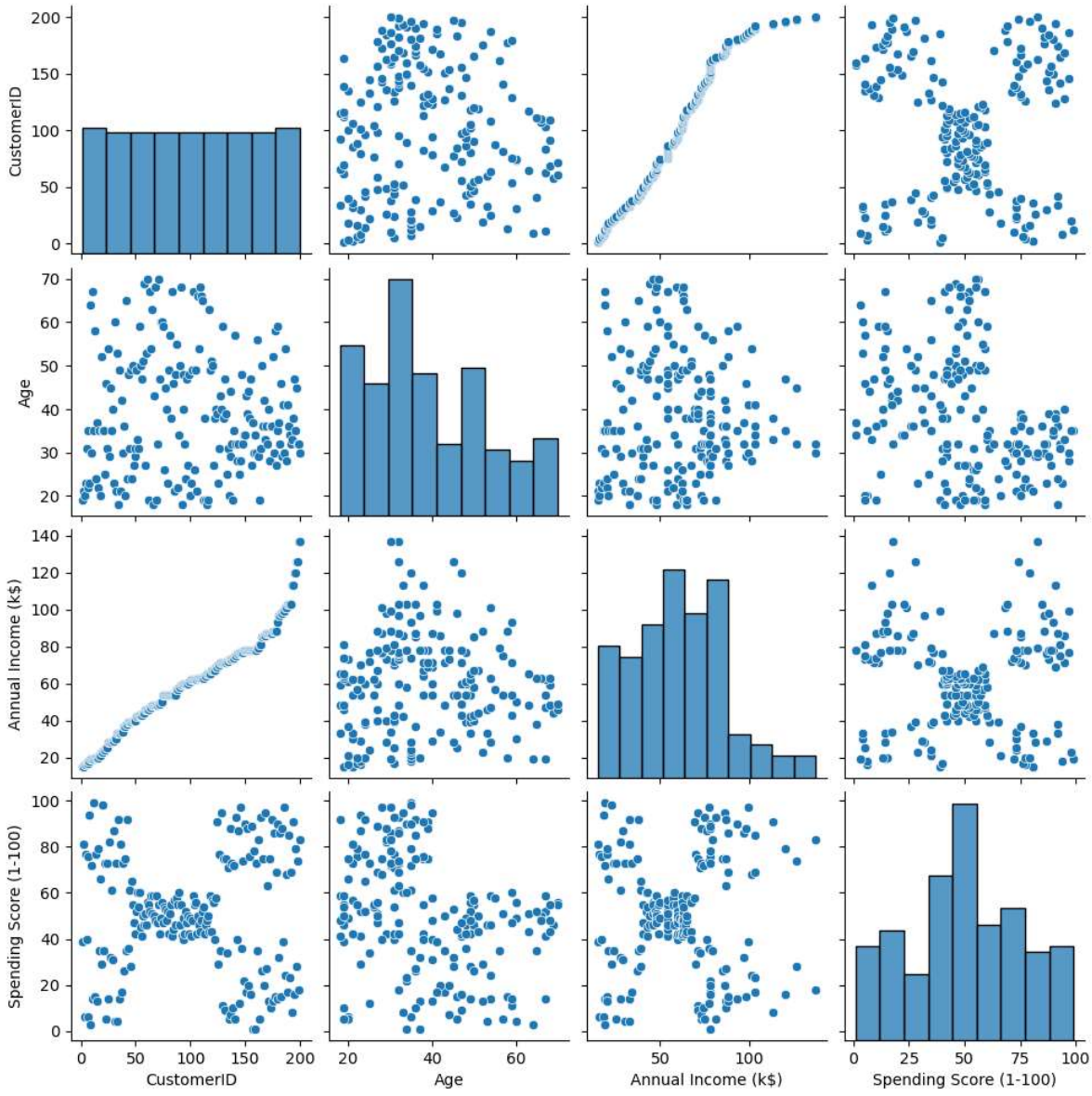


In [9]:

```
sns.pairplot(df)
```

Out[9]:

```
<seaborn.axisgrid.PairGrid at 0x19b380c6a30>
```




# handling a outliers

In [10]:

```
def whiskers(col):
    q1=np.quantile(col,0.25)
    q3=np.quantile(col,0.75)
    iqr=q3-q1
    uw=q3+1.5*iqr
    lw=q1-1.5*iqr
    return uw,lw
```

In [11]:

```
whiskers(df["Annual Income (k$)"])
```

Out[11]:

```
(132.75, -13.25)
```

In [12]:

```
a=df[df["Annual Income (k$)"]>132.75].index
a
```

Out[12]:

```
Int64Index([198, 199], dtype='int64')
```

In [13]:

```
df.loc[a,"Annual Income (k$)"]=132.75
```

# skew

In [14]:

```
col = df.select_dtypes("int64").columns
col
```

Out[14]:

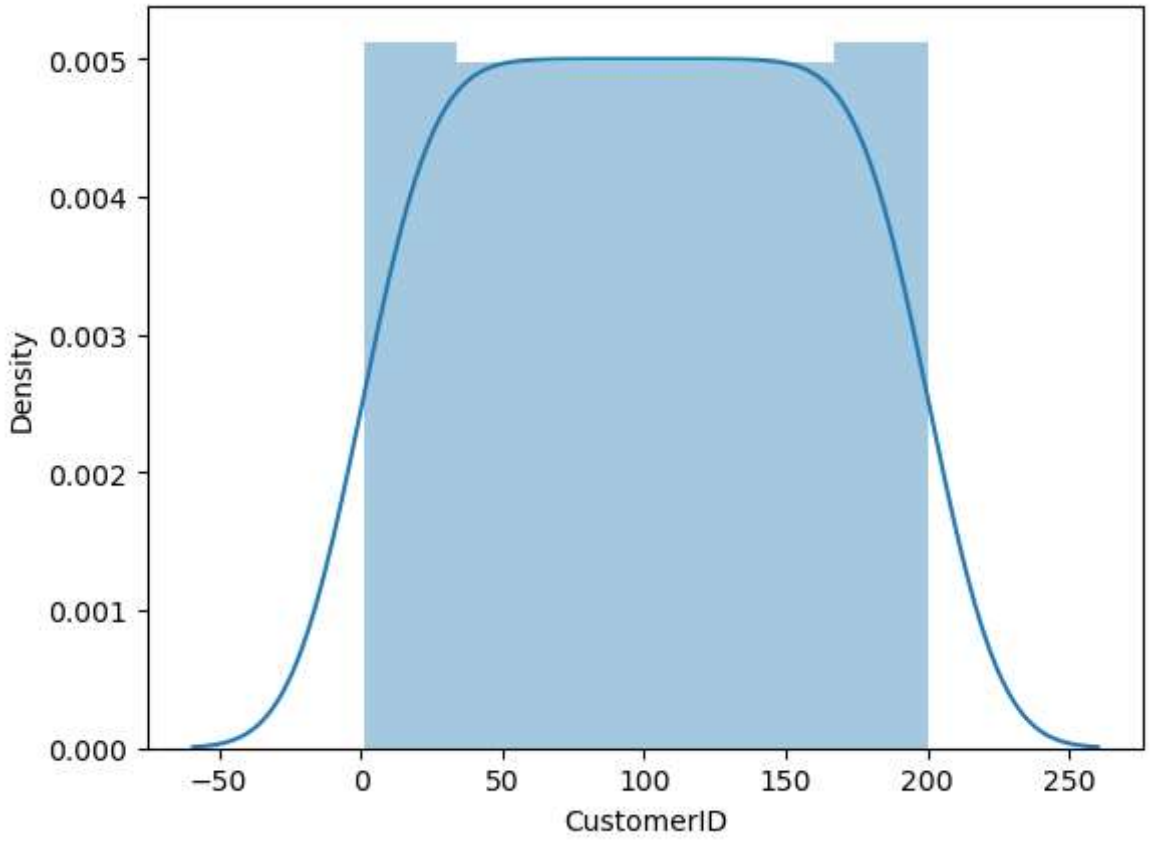
```
Index(['CustomerID', 'Age', 'Spending Score (1-100)'], dtype='object')
```

In [15]:

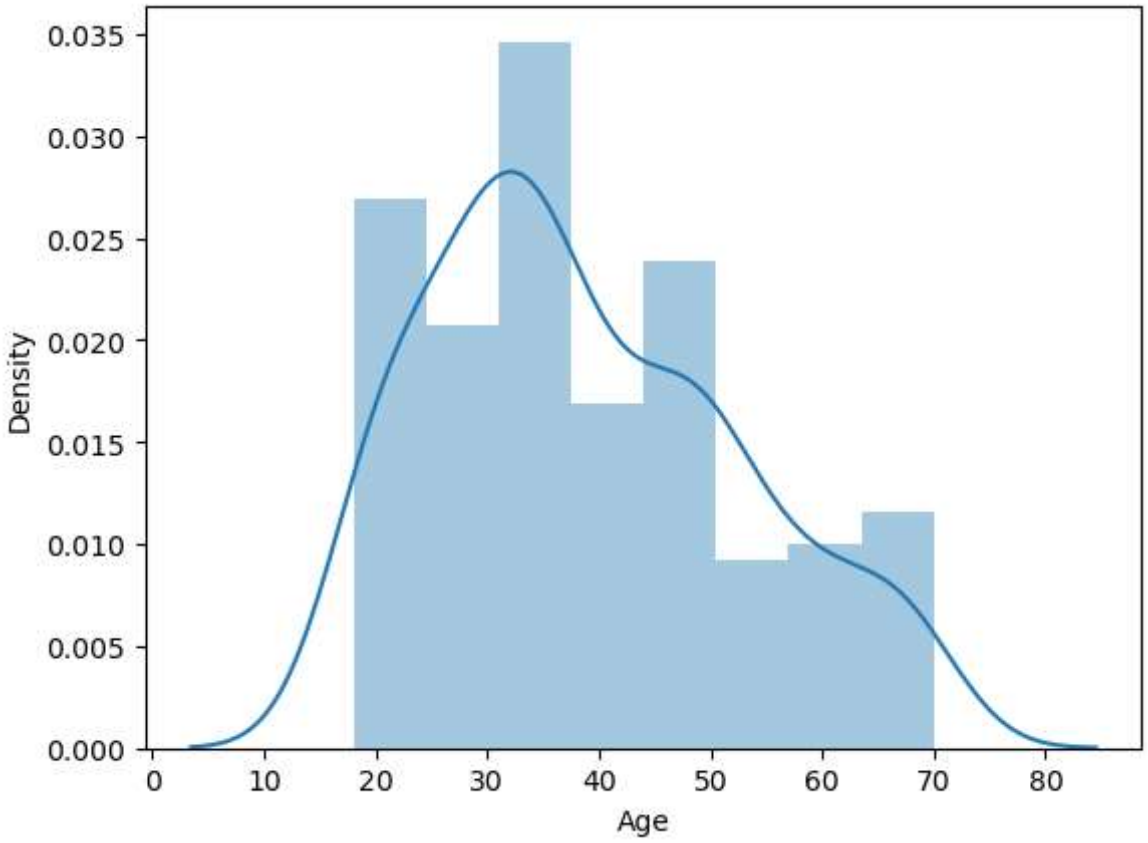
```
for i in df[col]:
    print(i)
    print(skew(df[i]))

    plt.figure()
    sns.distplot(df[i])
    plt.show()
```

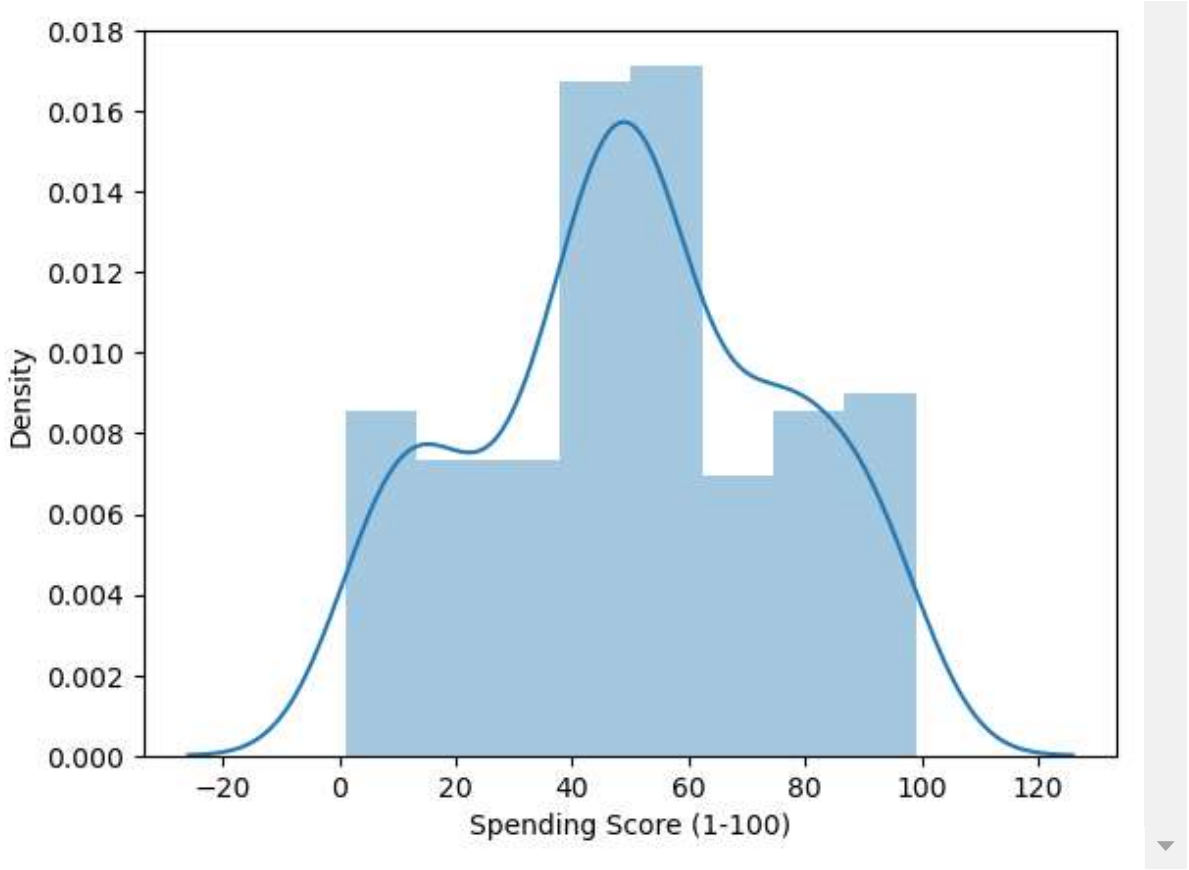
```
CustomerID
0.0
```




```
Age
0.48191947090957177
```




```
Spending Score (1-100)
-0.04686530945553505
```




In [16]:

```
x = df.iloc[:, [3,4]]
x
```

Out[16]:

| | Annual Income (k$) | Spending Score (1-100) |
|---|---|---|
| **0** | 15.00 | 39 |
| **1** | 15.00 | 81 |
| **2** | 16.00 | 6 |
| **3** | 16.00 | 77 |
| **4** | 17.00 | 40 |
| **...** | ... | ... |
| **195** | 120.00 | 79 |
| **196** | 126.00 | 28 |
| **197** | 126.00 | 74 |
| **198** | 132.75 | 18 |
| **199** | 132.75 | 83 |

200 rows × 2 columns

# standard scaler

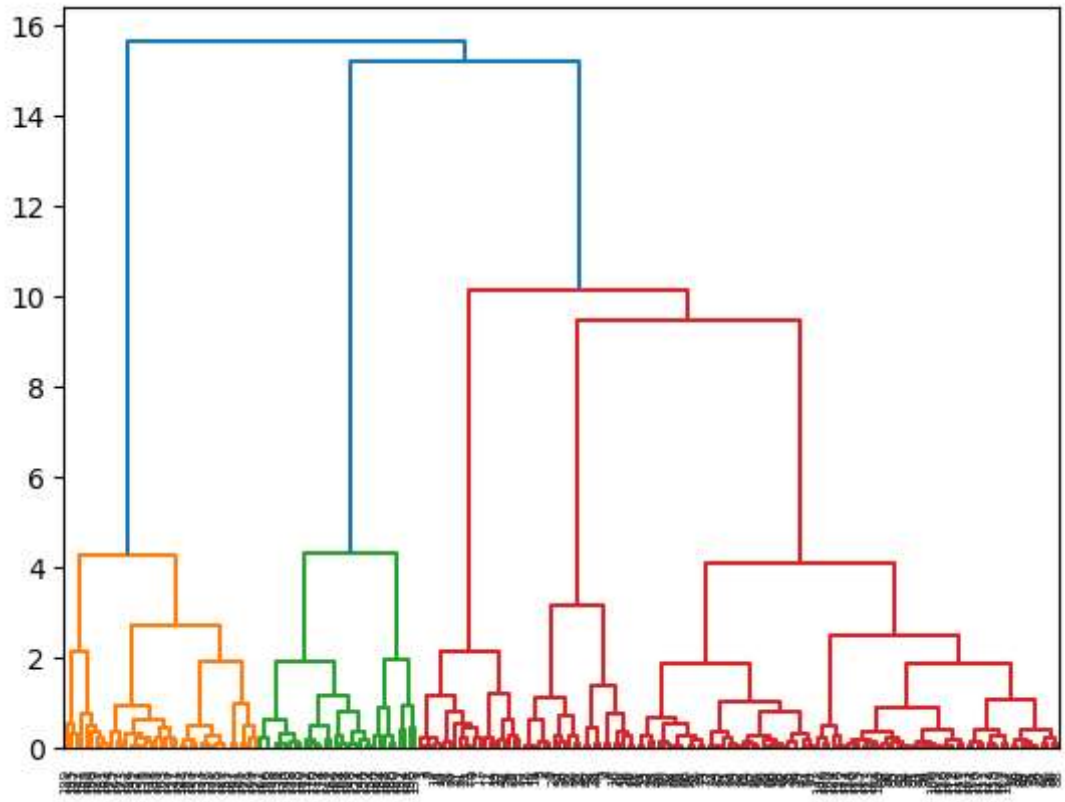
In [17]:

```
sc = StandardScaler()
x = sc.fit_transform(x)
```

# clustering

In [18]:

```
lk = hi.linkage(x, method="ward")
ddg = hi.dendrogram(lk)
```




In [19]:

```
hc = AgglomerativeClustering(n_clusters=5)
ylabel = hc.fit_predict(x)
```

In [20]:

```
df["target"]=ylabel
```

In [21]:

```
df["target"].value_counts()
```

Out[21]:

```
2    85
1    39
0    32
4    23
3    21
Name: target, dtype: int64
```

In [22]:

```
df.head()
```

Out[22]:

| | CustomerID | Genre | Age | Annual Income (k$) | Spending Score (1-100) | target |
|---|---|---|---|---|---|---|
| **0** | 1 | Male | 19 | 15.0 | 39 | 4 |
| **1** | 2 | Male | 21 | 15.0 | 81 | 3 |
| **2** | 3 | Female | 20 | 16.0 | 6 | 4 |
| **3** | 4 | Female | 23 | 16.0 | 77 | 3 |
| **4** | 5 | Female | 31 | 17.0 | 40 | 4 |

# classification

In [23]:

```
features = df.iloc[:, [3,4]]
y = df.iloc[:, -1]
```

In [24]:

```
xtrain, xtest, ytrain, ytest = train_test_split(x,y, test_size=0.3, random_state=1)
```

In [25]:

```
def pro(one):
    one.fit(xtrain, ytrain)
    ypred = one.predict(xtest)

    train = one.score(xtrain, ytrain)
    test = one.score(xtest, ytest)

    print(f"Training Accuracy : {train}\nTesting Accuracy : {test}\n\n")
    print(classification_report(ytest, ypred))
    return one
```

In [28]:

```
from sklearn.metrics import classification_report
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
```

In [30]:

```
svm=pro(SVC())
```

```
Training Accuracy : 0.9928571428571429
Testing Accuracy : 0.9833333333333333


              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00        11
           2       0.96      1.00      0.98        22
           3       1.00      1.00      1.00         8
           4       1.00      0.89      0.94         9

    accuracy                           0.98        60
   macro avg       0.99      0.98      0.98        60
weighted avg       0.98      0.98      0.98        60
```

In [ ]:

```

```