### Importing the libraries

In [78]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from fcmeans import FCM
from sklearn.cluster import KMeans
```

In [2]:

```
!pip install fuzzy-c-means
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/publi
c/simple/
Collecting fuzzy-c-means
  Downloading fuzzy_c_means-1.6.3-py3-none-any.whl (9.1 kB)
Requirement already satisfied: tabulate<0.9.0,>=0.8.9 in /usr/local/lib/python3.7/dist-pa
ckages (from fuzzy-c-means) (0.8.10)
Collecting typer<0.4.0,>=0.3.2
  Downloading typer-0.3.2-py3-none-any.whl (21 kB)
Requirement already satisfied: numpy<2.0.0,>=1.21.1 in /usr/local/lib/python3.7/dist-pack
ages (from fuzzy-c-means) (1.21.6)
Requirement already satisfied: pydantic<2.0.0,>=1.8.2 in /usr/local/lib/python3.7/dist-pa
ckages (from fuzzy-c-means) (1.9.2)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.7/dis
t-packages (from pydantic<2.0.0,>=1.8.2->fuzzy-c-means) (4.1.1)
Requirement already satisfied: click<7.2.0,>=7.1.1 in /usr/local/lib/python3.7/dist-packa
ges (from typer<0.4.0,>=0.3.2->fuzzy-c-means) (7.1.2)
Installing collected packages: typer, fuzzy-c-means
  Attempting uninstall: typer
    Found existing installation: typer 0.4.2
    Uninstalling typer-0.4.2:
      Successfully uninstalled typer-0.4.2
Successfully installed fuzzy-c-means-1.6.3 typer-0.3.2
```

### Load the Dataset

In [3]:

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

In [4]:

```
data.head()
```

Out[4]:

| | 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 |

In [5]:

```
data.corr()
```

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 |

### Univariate Analysis

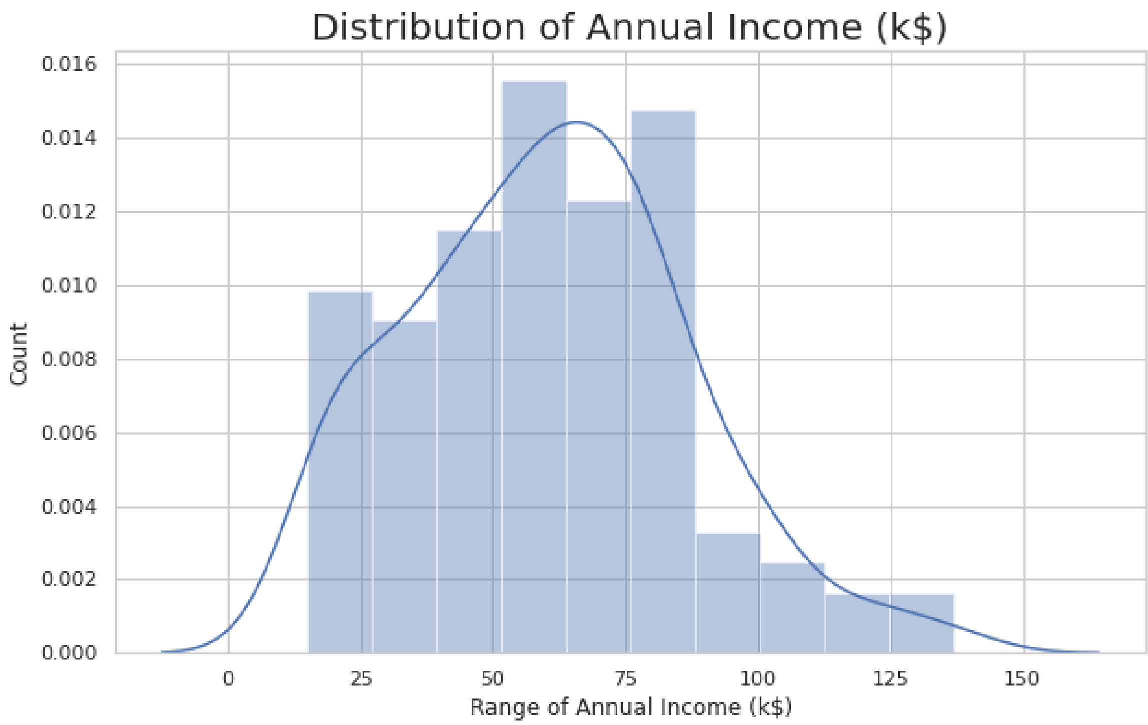
In [6]:

```
#Distribution of Annnual Income
plt.figure(figsize=(10, 6))
sns.set(style = 'whitegrid')
sns.distplot(data['Annual Income (k$)'])
plt.title('Distribution of Annual Income (k$)', fontsize = 20)
plt.xlabel('Range of Annual Income (k$)')
plt.ylabel('Count')
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `dis
tplot` is a deprecated function and will be removed in a future version. Please adapt you
r code to use either `displot` (a figure-level function with similar flexibility) or `his
tplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[6]:

```
Text(0, 0.5, 'Count')
```



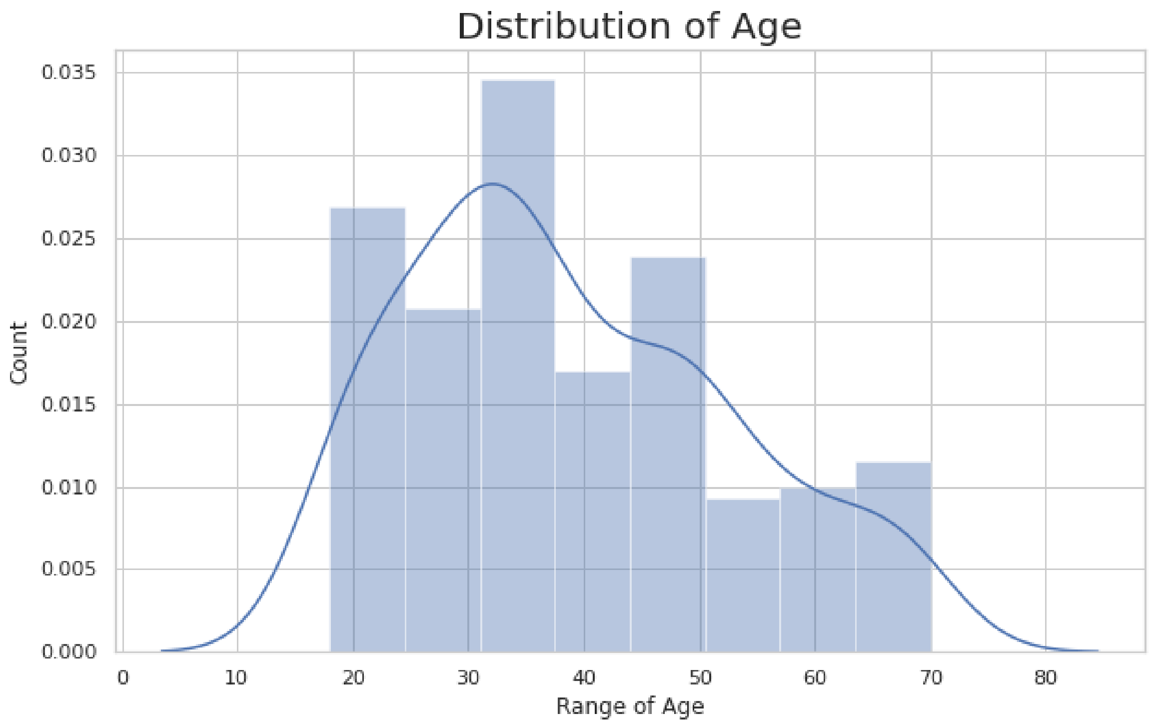

In [8]:

```
#Distribution of age
plt.figure(figsize=(10, 6))
sns.set(style = 'whitegrid')
sns.distplot(data['Age'])
plt.title('Distribution of Age', fontsize = 20)
plt.xlabel('Range of Age')
plt.ylabel('Count')
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `dis
tplot` is a deprecated function and will be removed in a future version. Please adapt you
r code to use either `displot` (a figure-level function with similar flexibility) or `his
tplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[8]:

Text(0, 0.5, 'Count')



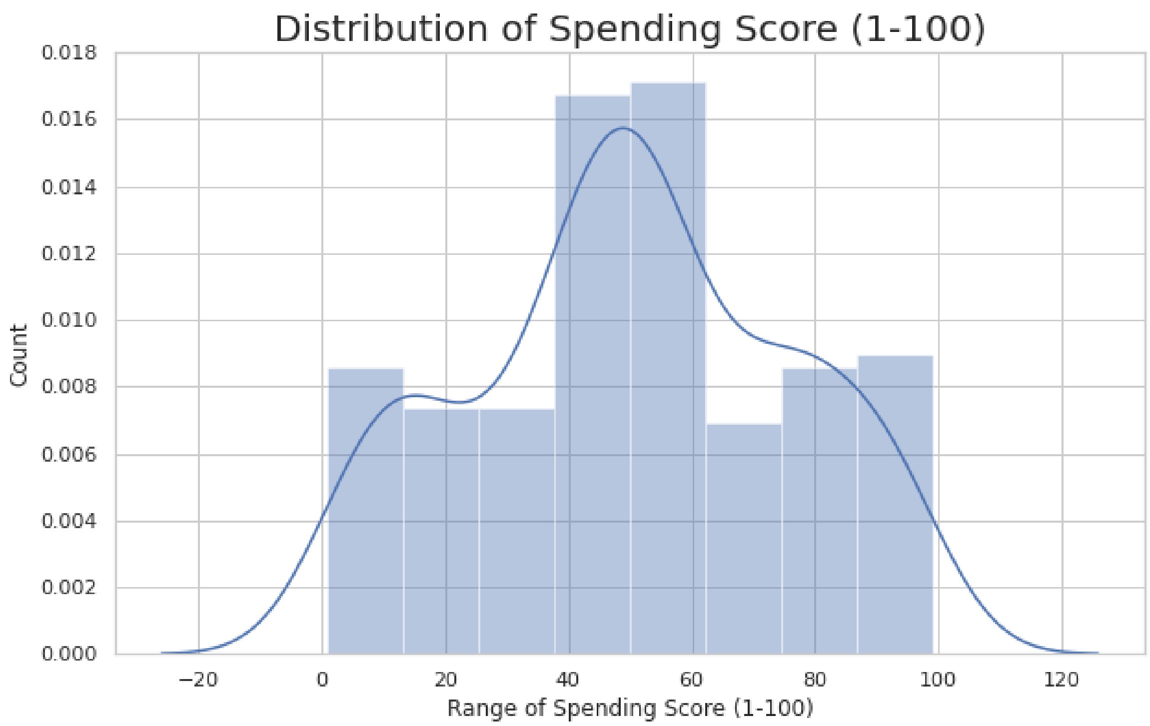

In [7]:

```
#Distribution of spending score
plt.figure(figsize=(10, 6))
sns.set(style = 'whitegrid')
sns.distplot(data['Spending Score (1-100)'])
plt.title('Distribution of Spending Score (1-100)', fontsize = 20)
plt.xlabel('Range of Spending Score (1-100)')
plt.ylabel('Count')
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `dis
tplot` is a deprecated function and will be removed in a future version. Please adapt you
r code to use either `displot` (a figure-level function with similar flexibility) or `his
tplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[7]:

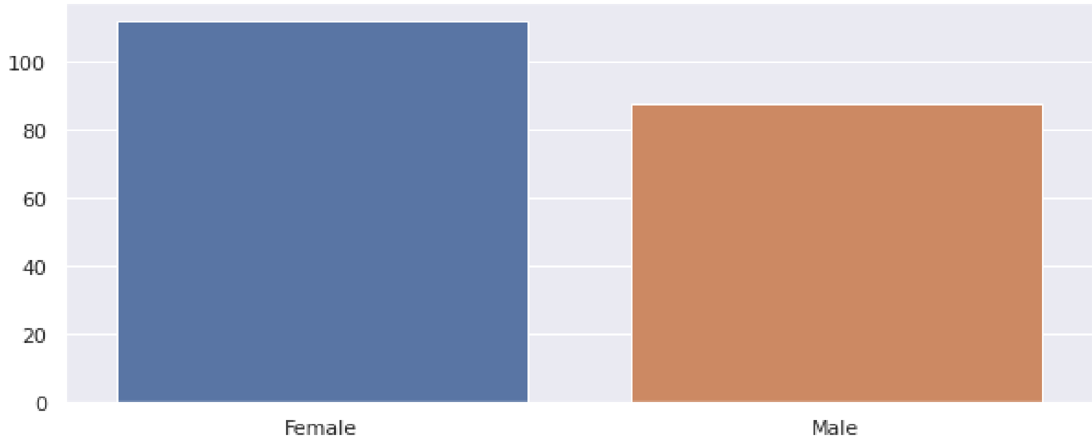
Text(0, 0.5, 'Count')



### Bivariate Analysis

In [10]:

```
genders = data.Gender.value_counts()
sns.set_style("darkgrid")
plt.figure(figsize=(10,4))
sns.barplot(x=genders.index, y=genders.values)
plt.show()
```



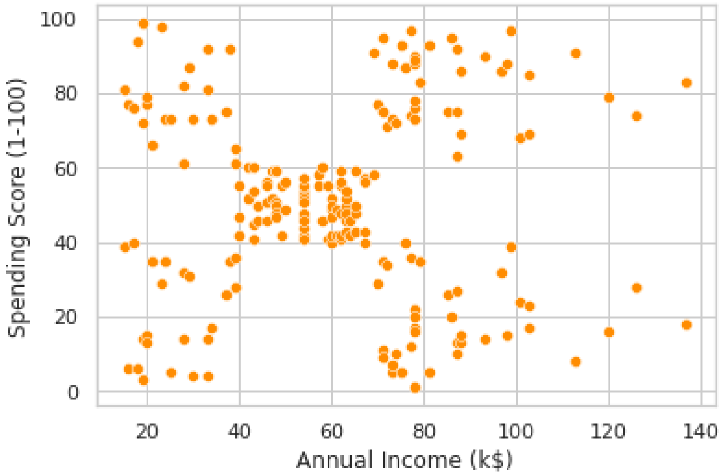

In [9]:

```
sns.scatterplot(data["Annual Income (k$)"],data['Spending Score (1-100)'],color='darkorange')
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[9]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f93d7664790>
```



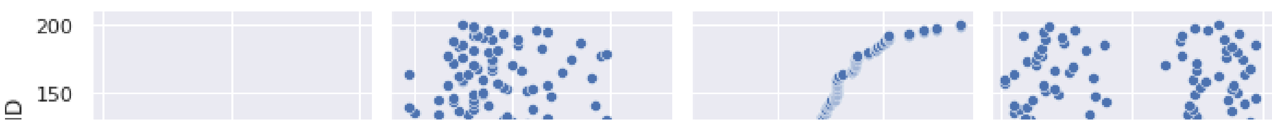

### Multivariate Analysis

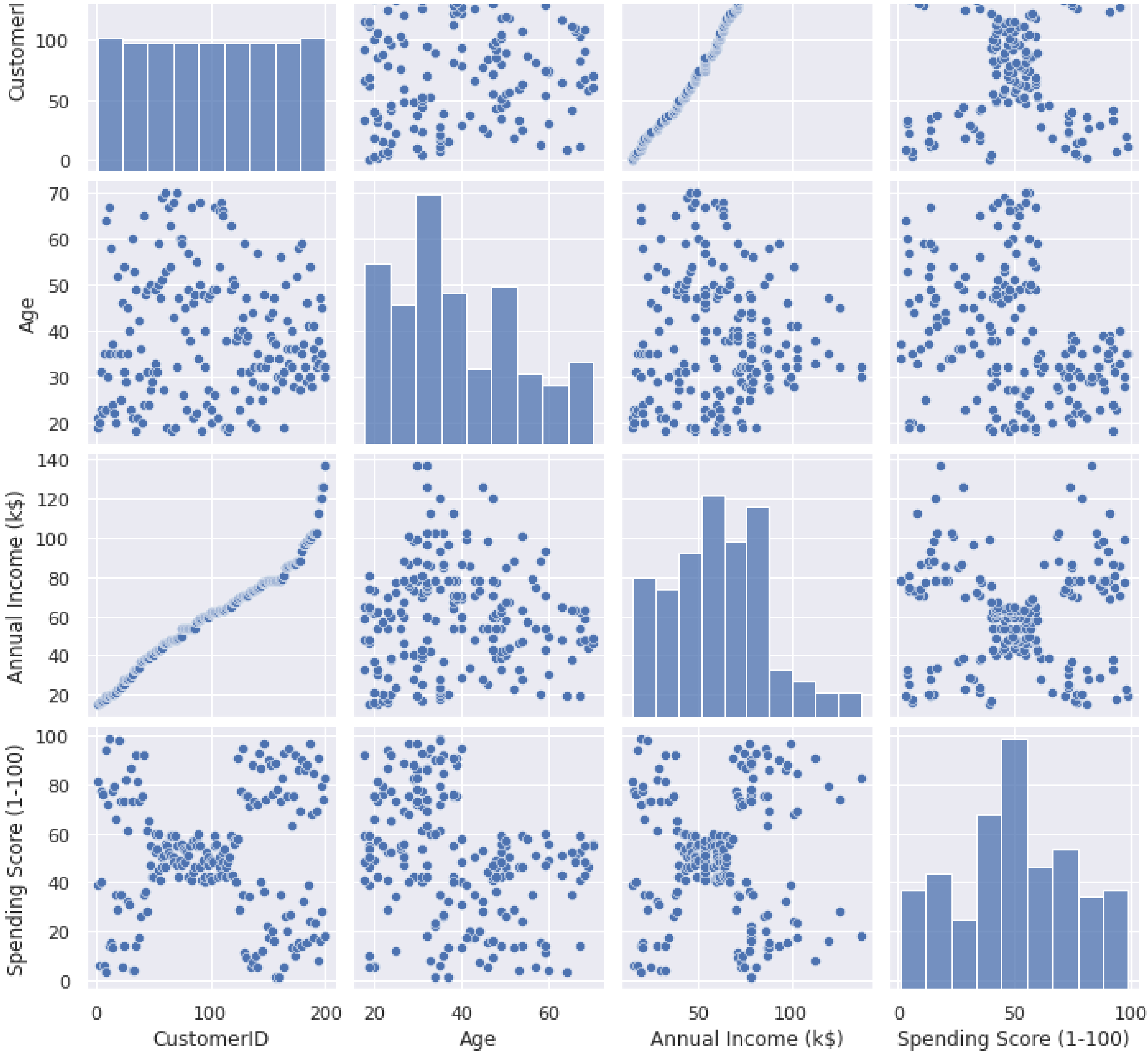
In [11]:

```
sns.pairplot(data)
```

Out[11]:

```
<seaborn.axisgrid.PairGrid at 0x7f93d750a690>
```






#### Descriptive Statistics

In [12]:

```
data.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   Gender                  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 [13]:

```
data.shape
```

Out[13]:

```
(200, 5)
```

In [14]:

```
data.describe().T
```

Out[14]:

| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| **CustomerID** | 200.0 | 100.50 | 57.879185 | 1.0 | 50.75 | 100.5 | 150.25 | 200.0 |
| **Age** | 200.0 | 38.85 | 13.969007 | 18.0 | 28.75 | 36.0 | 49.00 | 70.0 |
| **Annual Income (k$)** | 200.0 | 60.56 | 26.264721 | 15.0 | 41.50 | 61.5 | 78.00 | 137.0 |
| **Spending Score (1-100)** | 200.0 | 50.20 | 25.823522 | 1.0 | 34.75 | 50.0 | 73.00 | 99.0 |

In [15]:

```
data.isna().any()
```

Out[15]:

```
CustomerID                False
Gender                    False
Age                       False
Annual Income (k$)        False
Spending Score (1-100)    False
dtype: bool
```

In [79]:

```
data.mean()
```

Out[79]:

```
CustomerID                100.50
Gender                      0.44
Age                        38.85
Annual Income (k$)         60.56
Spending Score (1-100)     50.20
dtype: float64
```

In [16]:

```
data.median()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning: Dropping o
f nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a
future version this will raise TypeError.  Select only valid columns before calling the r
eduction.
  """Entry point for launching an IPython kernel.
```

Out[16]:

```
CustomerID                100.5
Age                        36.0
Annual Income (k$)         61.5
Spending Score (1-100)     50.0
dtype: float64
```

In [17]:

```
data.mode()
```

Out[17]:

| | CustomerID | Gender | Age | Annual Income (k$) | Spending Score (1-100) |
|---|---|---|---|---|---|
| **0** | 1 | Female | 32.0 | 54.0 | 42.0 |
| **1** | 2 | NaN | NaN | 78.0 | NaN |
| **2** | 3 | NaN | NaN | NaN | NaN |
| **3** | 4 | NaN | NaN | NaN | NaN |
| **4** | 5 | NaN | NaN | NaN | NaN |
| **...** | ... | ... | ... | ... | ... |
| **195** | 196 | NaN | NaN | NaN | NaN |

| | CustomerID | Gender | Age | Annual Income (k$) | Spending Score (1-100) |
|---|---|---|---|---|---|
| 196 | 197 | NaN | NaN | NaN | NaN |
| 197 | 198 | NaN | NaN | NaN | NaN |
| 198 | 199 | NaN | NaN | NaN | NaN |
| 199 | 200 | NaN | NaN | NaN | NaN |

200 rows × 5 columns

In [18]:

```
data.var()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning: Dropping o
f nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a
future version this will raise TypeError.  Select only valid columns before calling the r
eduction.
  """Entry point for launching an IPython kernel.
```

Out[18]:

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

In [19]:

```
data.std()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning: Dropping o
f nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a
future version this will raise TypeError.  Select only valid columns before calling the r
eduction.
  """Entry point for launching an IPython kernel.
```

Out[19]:

```
CustomerID                57.879185
Age                       13.969007
Annual Income (k$)        26.264721
Spending Score (1-100)    25.823522
dtype: float64
```

In [20]:

```
data.describe(include="all")
```

Out[20]:

| | CustomerID | Gender | Age | Annual Income (k$) | Spending Score (1-100) |
|---|---|---|---|---|---|
| count | 200.000000 | 200 | 200.000000 | 200.000000 | 200.000000 |
| unique | NaN | 2 | NaN | NaN | NaN |
| top | NaN | Female | NaN | NaN | NaN |
| freq | NaN | 112 | NaN | NaN | NaN |
| mean | 100.500000 | NaN | 38.850000 | 60.560000 | 50.200000 |
| std | 57.879185 | NaN | 13.969007 | 26.264721 | 25.823522 |
| min | 1.000000 | NaN | 18.000000 | 15.000000 | 1.000000 |
| 25% | 50.750000 | NaN | 28.750000 | 41.500000 | 34.750000 |
| 50% | 100.500000 | NaN | 36.000000 | 61.500000 | 50.000000 |
| 75% | 150.250000 | NaN | 49.000000 | 78.000000 | 73.000000 |
| max | 200.000000 | NaN | 70.000000 | 137.000000 | 99.000000 |

### Handling Missing Values

**Handling Missing Values**

In [21]:

```
data.isnull().sum()
```

Out[21]:

```
CustomerID                0
Gender                    0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

In [22]:

```
new_df = data.dropna(how='all')
new_df
```

Out[22]:

| | 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 × 5 columns

In [132]:

```
new_df.head()
```

Out[132]:

| | 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 |

**Finding the Outliers**

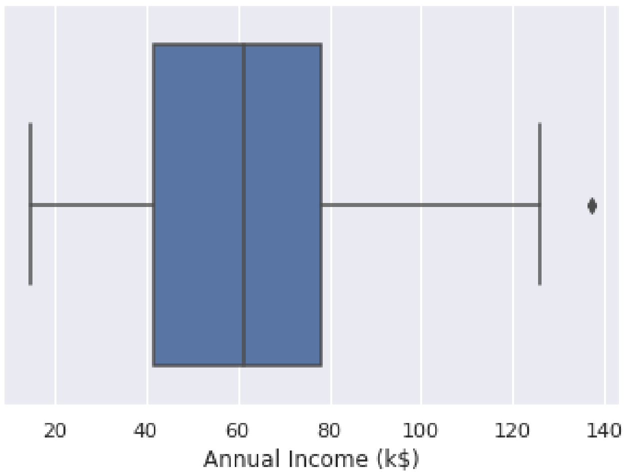
In [133]:

```
sns.boxplot(data['Annual Income (k$)'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[133]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f93758e5a90>

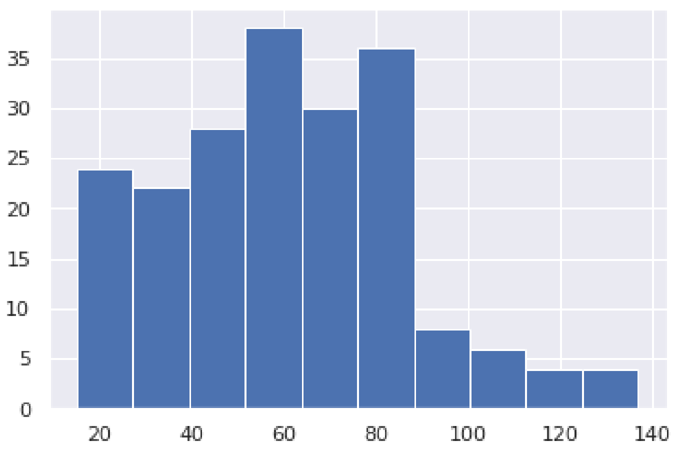



In [25]:

```
data['Annual Income (k$)'].hist()
```

Out[25]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f93d2d38150>




**Encoding**

In [26]:

```
from sklearn.preprocessing import LabelEncoder
from collections import Counter as count
```

In [27]:

```
le=LabelEncoder()
```

In [28]:

```
data['Gender'].unique()
```

Out[28]:

```
array(['Male', 'Female'], dtype=object)
```

In [29]:

```
data['Gender']=le.fit_transform(data['Gender'])
```

In [30]:

```
count(data['Gender'])
```

Out[30]:

```
Counter({1: 88, 0: 112})
```

#### Scale the data

In [74]:

```python
from sklearn.preprocessing import StandardScaler

s = StandardScaler()
s_xtrain = s.fit_transform(xtrain)
```

In [37]:

```python
s_xtrain
```

Out[37]:

```
array([[ 0.27949921, -0.90453403,  0.81961441,  0.20094935,  0.23917839],
       [ 0.12380999,  1.1055416 ,  1.95796776,  0.0470073 ,  0.04673601],
       [-0.13567203,  1.1055416 , -1.15353139, -0.06844923,  0.04673601],
       [ 1.10984169, -0.90453403, -0.47051938,  0.93217405,  1.70174048],
       [ 1.55961053, -0.90453403, -0.09106827,  1.97128284,  1.54778658],
       [-0.60273967,  1.1055416 , -1.53298251, -0.53027536,  0.31615534],
       [ 1.07524408, -0.90453403, -0.24284871,  0.89368854,  0.93197096],
       [ 0.1930052 , -0.90453403, -1.60887273,  0.12397832, -0.1072179 ],
       [-0.62003848, -0.90453403,  2.18563843, -0.53027536, -0.1072179 ],
       [-0.63733728, -0.90453403,  0.28838285, -0.53027536, -0.03024095],
       [ 0.29679801,  1.1055416 , -0.92586072,  0.20094935,  0.20068991],
       [-1.12170372, -0.90453403, -0.24284871, -0.95361598, -0.95396437],
       [ 1.61150694, -0.90453403,  0.4401633 ,  2.47159448, -0.87698742],
       [ 1.00604888, -0.90453403, -0.77408028,  0.66277547,  1.23987877],
       [-0.06647682,  1.1055416 , -1.45709229, -0.02996372, -0.06872942],
       [ 0.85035966, -0.90453403,  0.36427307,  0.62428996, -1.18489523],
       [-0.34325765, -0.90453403,  0.51605352, -0.29936229, -0.2611718 ],
       [ 1.09254289,  1.1055416 ,  0.21249263,  0.93217405, -1.18489523],
       [ 0.24490161, -0.90453403,  0.74372419,  0.12397832,  0.31615534],
       [-1.06980732,  1.1055416 , -1.15353139, -0.91513047,  1.58627505],
       [-0.39515405,  1.1055416 ,  1.35084597, -0.29936229,  0.00824753],
       [-0.37785525,  1.1055416 , -0.09106827, -0.29936229,  0.16220144],
       [-1.22549653,  1.1055416 ,  1.04728508, -1.10755802, -1.80071085],
       [ 1.16173809,  1.1055416 ,  0.06071218,  0.97065956, -1.45431456],
       [-0.94871571, -0.90453403, -0.77408028, -0.83815945, -0.33814875],
       [-1.34658814,  1.1055416 , -0.77408028, -1.29998557,  1.20139029],
       [-0.8622217 ,  1.1055416 ,  1.50262642, -0.72270291,  0.35464382],
       [ 0.03731599, -0.90453403, -1.38120206,  0.00852179, -0.33814875],
       [-1.48497856,  1.1055416 , -1.45709229, -1.56938415,  0.58557467],
       [ 1.57690933, -0.90453403,  0.59194374,  2.24068142, -1.33884913],
       [-0.8968193 ,  1.1055416 , -0.47051938, -0.76118842,  0.35464382],
       [-1.53687496,  1.1055416 , -0.16695849, -1.60786966, -1.45431456],
       [-0.36055645,  1.1055416 ,  2.10974821, -0.29936229, -0.37663723],
       [-0.32595885, -0.90453403, -1.38120206, -0.29936229,  0.23917839],
       [ 1.40392132, -0.90453403,  0.1366024 ,  1.43248569, -0.45361418],
       [-1.27739294, -0.90453403, -1.22942162, -1.26150006,  1.39383267],
       [-1.72716178, -0.90453403, -1.22942162, -1.7618117 ,  1.00894791],
       [-1.38118575,  1.1055416 , -0.62229983, -1.41544211,  0.85499401],
       [ 0.76386566, -0.90453403, -0.54640961,  0.58580445,  0.89348248],
       [ 1.47311652, -0.90453403,  0.1366024 ,  1.58642773, -1.30036066],
       [-1.13900252, -0.90453403, -0.69819005, -1.06907251,  0.85499401],
       [ 1.49041532, -0.90453403, -0.24284871,  1.58642773,  1.31685572],
       [ 1.43851892, -0.90453403,  1.12317531,  1.50945671, -1.03094132],
       [ 1.45581772,  1.1055416 , -0.8499705 ,  1.50945671,  0.66255162],
       [-1.00061211, -0.90453403, -1.15353139, -0.87664496,  0.5470862 ],
       [ 0.50438363, -0.90453403, -1.07764117,  0.3933769 , -0.64605656],
       [-0.10107443, -0.90453403, -0.92586072, -0.06844923, -0.03024095],
       [ 1.31742731,  1.1055416 , -0.31873894,  1.20157263,  1.5092981 ],
       [ 1.52501293, -0.90453403, -0.54640961,  1.58642773,  0.7010401 ],
       [-1.05250852,  1.1055416 ,  0.66783397, -0.87664496, -0.56907961],
       [ 0.45248723,  1.1055416 , -0.09106827,  0.35489139,  0.93197096],
       [ 0.48708483,  1.1055416 , -0.01517804,  0.35489139,  0.93197096],
```

```
[-0.41245286, -0.90453403,  0.74372419, -0.29936229, -0.33814875],
[ 0.98875008, -0.90453403,  1.27495575,  0.66277547, -0.60756809],
[ 1.2828297 ,  1.1055416 , -0.92586072,  1.00914507,  0.7010401 ],
[ 0.69467045, -0.90453403, -0.54640961,  0.54731894,  1.39383267],
[ 1.12714049, -0.90453403, -0.24284871,  0.97065956, -0.9154759 ],
[-1.57147257, -0.90453403,  1.4267362 , -1.60786966, -1.37733761],
[ 0.1584076 , -0.90453403, -0.09106827,  0.08549281, -0.33814875],
[-1.58877137, -0.90453403, -0.31873894, -1.64635517,  1.85569438],
[-1.39848455, -0.90453403,  0.51605352, -1.41544211, -1.76222237],
[ 0.05461479, -0.90453403,  2.03385798,  0.0470073 , -0.03024095],
[-0.30866005,  1.1055416 ,  0.66783397, -0.29936229, -0.18419485],
[ 0.74656685,  1.1055416 ,  0.66783397,  0.58580445, -0.56907961],
[-1.67526538, -0.90453403, -0.31873894, -1.68484068, -1.72373389],
[ 0.43518842,  1.1055416 ,  1.50262642,  0.35489139, -1.53129151],
[ 0.79846326,  1.1055416 , -0.39462916,  0.62428996,  1.5092981 ],
[ 0.90225607, -0.90453403, -0.92586072,  0.62428996,  1.47080962],
[ 0.78116446, -0.90453403, -0.39462916,  0.62428996, -1.10791828],
[ 0.02001718,  1.1055416 ,  0.74372419,  0.00852179,  0.20068991],
[-1.19089893, -0.90453403,  0.74372419, -1.10755802, -1.41582609],
[ 0.08921239,  1.1055416 ,  2.18563843,  0.0470073 , -0.29966028],
[-0.9141181 , -0.90453403,  0.74372419, -0.76118842,  0.04673601],
[ 0.53898123,  1.1055416 , -1.45709229,  0.43186241, -1.76222237],
[ 0.71196925,  1.1055416 , -1.07764117,  0.58580445, -1.49280304],
[ 0.72926805,  1.1055416 , -0.8499705 ,  0.58580445,  1.77871743],
[-1.70986298, -0.90453403, -0.62229983, -1.72332619, -0.41512571],
[ 0.52168243, -0.90453403, -0.62229983,  0.3933769 ,  0.77801705],
[-1.01791091, -0.90453403,  0.74372419, -0.87664496, -0.87698742],
[-0.11837323, -0.90453403,  0.59194374, -0.06844923, -0.14570637],
[ 1.2655309 ,  1.1055416 ,  1.4267362 ,  1.00914507, -1.37733761],
[-1.29469174, -0.90453403,  0.06071218, -1.26150006, -0.76152199],
[ 0.55628004, -0.90453403, -0.77408028,  0.43186241,  1.43232115],
[ 1.17903689,  1.1055416 , -0.8499705 ,  0.97065956,  0.93197096],
[ 1.33472611, -0.90453403, -0.16695849,  1.35551467, -0.72303352],
[-0.55084327, -0.90453403,  0.59194374, -0.49178985, -0.33814875],
[ 0.26220041, -0.90453403,  0.89550464,  0.20094935, -0.29966028],
[ 1.54231173,  1.1055416 , -0.47051938,  1.97128284, -1.64675694],
[ 1.62880574,  1.1055416 , -0.54640961,  2.47159448,  0.89348248],
[ 0.59087764,  1.1055416 , -0.54640961,  0.43186241,  0.85499401],
[-0.49894686,  1.1055416 ,  1.50262642, -0.29936229, -0.14570637],
[ 1.36932371,  1.1055416 ,  0.51605352,  1.39400018, -1.37733761],
[-0.15297083, -0.90453403, -0.54640961, -0.06844923, -0.33814875],
[ 0.83306086,  1.1055416 , -0.01517804,  0.62428996,  1.43232115],
[ 0.81576206,  1.1055416 ,  0.28838285,  0.62428996, -1.30036066],
[ 1.2136345 ,  1.1055416 , -0.24284871,  0.97065956,  1.58627505],
[-0.17026963, -0.90453403,  0.06071218, -0.06844923, -0.41512571],
[-1.46767976,  1.1055416 ,  0.97139486, -1.49241313, -0.83849894],
[-1.31199054,  1.1055416 , -0.31873894, -1.29998557,  0.39313229],
[-1.15630133, -0.90453403,  0.21249263, -1.06907251, -1.30036066],
[-0.79302649,  1.1055416 ,  2.26152865, -0.6842174 , -0.18419485],
[-1.24279533, -0.90453403, -1.38120206, -1.22301455,  0.85499401],
[-0.65463608,  1.1055416 , -1.60887273, -0.53027536,  0.31615534],
[ 0.64277404, -0.90453403,  1.35084597,  0.50883343, -1.76222237],
[-0.23946484, -0.90453403,  0.81961441, -0.14542025, -0.18419485],
[ 0.95415247,  1.1055416 , -0.39462916,  0.62428996, -1.91617627],
[-1.26009414,  1.1055416 ,  1.57851664, -1.22301455, -1.80071085],
[-0.29136124, -0.90453403,  1.19906553, -0.18390576,  0.27766686],
[-0.18756843,  1.1055416 ,  0.66783397, -0.06844923, -0.06872942],
[ 0.66007285,  1.1055416 , -0.54640961,  0.50883343,  1.62476353],
[ 0.40059082,  1.1055416 ,  0.28838285,  0.35489139, -0.60756809],
[ 0.86765846, -0.90453403, -0.09106827,  0.62428996,  0.97045943],
[-1.55417376, -0.90453403, -1.15353139, -1.60786966,  1.00894791],
[-0.44705046,  1.1055416 ,  0.06071218, -0.29936229, -0.1072179 ],
[ 1.35202491, -0.90453403, -0.54640961,  1.35551467,  1.35534419],
[ 1.38662251, -0.90453403, -0.77408028,  1.39400018,  1.43232115],
[-1.20819773,  1.1055416 , -1.60887273, -1.10755802,  1.58627505],
[-0.70653248, -0.90453403,  2.10974821, -0.56876087,  0.04673601],
[ 0.33139561, -0.90453403,  0.06071218,  0.27792037,  0.27766686],
[ 0.07191359,  1.1055416 ,  1.12317531,  0.0470073 , -0.18419485],
[-0.25676364, -0.90453403, -0.39462916, -0.14542025,  0.35464382],
[-0.8449229 , -0.90453403,  0.81961441, -0.72270291, -0.22268333],
[ 0.62547524, -0.90453403, -0.31873894,  0.47034792,  0.81650553],
[-0.04917802, -0.90453403, -1.22942162,  0.00852179, -0.37663723],
```

```
       [-1.50227736, -0.90453403, -0.31873894, -1.56938415, -0.60756809],
       [ 0.210304  , -0.90453403, -1.53298251,  0.12397832, -0.03024095],
       [ 1.05794528,  1.1055416 ,  0.81961441,  0.89368854, -0.95396437],
       [-1.08710612, -0.90453403,  1.95796776, -0.91513047, -0.60756809],
       [-1.77905819,  1.1055416 , -1.53298251, -1.80029721, -0.45361418],
       [-0.51624567, -0.90453403,  1.57851664, -0.45330434,  0.20068991],
       [-1.64066777,  1.1055416 ,  1.88207754, -1.64635517, -1.83919932],
       [ 1.59420813, -0.90453403, -0.31873894,  2.24068142,  1.08592486],
       [ 0.93685367, -0.90453403, -0.69819005,  0.62428996,  1.04743639],
       [ 0.91955487,  1.1055416 , -0.16695849,  0.62428996, -1.91617627],
       [ 0.34869442,  1.1055416 , -0.01517804,  0.27792037,  1.54778658],
       [ 0.1757064 ,  1.1055416 , -1.53298251,  0.08549281, -0.18419485],
       [-0.67193488,  1.1055416 ,  1.80618731, -0.53027536,  0.00824753],
       [-1.51957616,  1.1055416 , -1.30531184, -1.60786966,  1.08592486],
       [ 0.38329202, -0.90453403, -0.62229983,  0.31640588,  1.00894791],
       [-1.62336897, -0.90453403, -0.69819005, -1.64635517,  0.81650553]])
```

In [38]:

```
s_xtest=s.transform(xtest)
```

In [39]:

```
s_xtest
```

Out[39]:

```
array([[-7.58428889e-01,  1.10554160e+00,  1.04728508e+00,
        -6.07246381e-01, -1.84194850e-01],
       [-1.69256418e+00, -9.04534034e-01, -1.30531184e+00,
        -1.72332619e+00,  9.70459433e-01],
       [-1.43308215e+00,  1.10554160e+00, -3.18738938e-01,
        -1.45392762e+00, -6.07568087e-01],
       [ 1.64610454e+00,  1.10554160e+00, -5.46409608e-01,
         2.89493510e+00, -1.26187218e+00],
       [-8.79520500e-01, -9.04534034e-01, -6.22299831e-01,
        -7.22702913e-01,  1.23712959e-01],
       [-1.45038095e+00, -9.04534034e-01, -3.18738938e-01,
        -1.49241313e+00,  1.81720591e+00],
       [ 1.02334768e+00,  1.10554160e+00, -1.53298251e+00,
         7.39746496e-01, -1.76222237e+00],
       [-8.27624095e-01,  1.10554160e+00,  5.91943742e-01,
        -7.22702913e-01, -3.76637231e-01],
       [-5.85440873e-01, -9.04534034e-01, -5.46409608e-01,
        -5.30275359e-01, -1.45706374e-01],
       [-1.74446058e+00, -9.04534034e-01, -1.45709229e+00,
        -1.76181170e+00, -1.72373389e+00],
       [-8.37756251e-02,  1.10554160e+00,  6.67833965e-01,
        -2.99637191e-02, -3.38148754e-01],
       [-1.60607017e+00,  1.10554160e+00,  2.10974821e+00,
        -1.64635517e+00, -1.41582609e+00],
       [-4.81648063e-01,  1.10554160e+00, -1.00175095e+00,
        -2.99362295e-01,  1.23712959e-01],
       [ 6.77371647e-01, -9.04534034e-01, -8.49970501e-01,
         5.47318943e-01, -4.15125707e-01],
       [ 3.65993217e-01, -9.04534034e-01, -1.22942162e+00,
         3.16405878e-01, -8.38498944e-01],
       [-6.89233682e-01, -9.04534034e-01,  1.12317531e+00,
        -5.68760870e-01,  3.16155339e-01],
       [ 1.06511193e-01,  1.10554160e+00,  2.03385798e+00,
         4.70073025e-02, -1.07217898e-01],
       [-4.29751658e-01, -9.04534034e-01, -1.22942162e+00,
        -2.99362295e-01,  4.67360067e-02],
       [ 1.41108796e-01, -9.04534034e-01, -1.53298251e+00,
         4.70073025e-02,  1.23712959e-01],
       [ 1.42122012e+00,  1.10554160e+00, -6.98190054e-01,
         1.43248569e+00,  1.77871743e+00],
       [ 8.84957267e-01, -9.04534034e-01,  5.91943742e-01,
         6.24289964e-01, -1.33884913e+00],
       [ 4.69786027e-01,  1.10554160e+00,  5.91943742e-01,
         3.54891389e-01, -1.60826847e+00],
       [-7.23831286e-01,  1.10554160e+00, -1.53298251e+00,
```

```
 -6.07246381e-01,  1.62201435e-01],
[-2.74062443e-01, -9.04534034e-01, -1.30531184e+00,
 -1.83905762e-01,  1.62201435e-01],
[-1.45804185e-02,  1.10554160e+00,  2.10974821e+00,
  8.52179167e-03,  3.16155339e-01],
[ 3.14096813e-01, -9.04534034e-01, -9.10682680e-02,
  2.00949346e-01, -4.15125707e-01],
[ 5.73578837e-01, -9.04534034e-01,  3.64273072e-01,
  4.31862410e-01, -1.68524542e+00],
[-1.76175938e+00,  1.10554160e+00, -1.38120206e+00,
 -1.80029721e+00,  1.16290181e+00],
[-9.66014509e-01, -9.04534034e-01, -9.25860724e-01,
 -8.38159445e-01, -1.45706374e-01],
[ 1.19633570e+00,  1.10554160e+00, -2.42848715e-01,
  9.70659561e-01, -1.56977999e+00],
[ 9.71451275e-01, -9.04534034e-01, -6.98190054e-01,
  6.24289964e-01,  8.54994005e-01],
[-1.10440492e+00, -9.04534034e-01, -1.45709229e+00,
 -9.53615978e-01,  9.31970957e-01],
[-4.64349261e-01, -9.04534034e-01,  4.40163295e-01,
 -2.99362295e-01,  8.52244828e-02],
[-2.04867237e-01,  1.10554160e+00, -1.60887273e+00,
 -1.06934741e-01, -3.76637231e-01],
[-1.17360013e+00, -9.04534034e-01, -1.38120206e+00,
 -1.10755802e+00,  1.16290181e+00],
[ 1.30012851e+00,  1.10554160e+00,  1.50262642e+00,
  1.20157263e+00, -1.41582609e+00],
[ 4.17889622e-01,  1.10554160e+00,  6.07121786e-02,
  3.54891389e-01,  1.70174048e+00],
[ 1.14443929e+00,  1.10554160e+00, -5.46409608e-01,
  9.70659561e-01,  4.70109244e-01],
[-9.83313310e-01, -9.04534034e-01,  8.19614412e-01,
 -8.38159445e-01,  1.62201435e-01],
[ 1.23093330e+00, -9.04534034e-01,  9.71394858e-01,
  1.00914507e+00, -1.45431456e+00],
[ 1.50771413e+00, -9.04534034e-01, -3.94629161e-01,
  1.58642773e+00, -1.06942980e+00],
[-1.65796657e+00, -9.04534034e-01, -1.22942162e+00,
 -1.68484068e+00,  1.66325200e+00],
[-1.32928934e+00, -9.04534034e-01,  4.40163295e-01,
 -1.29998557e+00, -7.23033516e-01],
[ 6.08176440e-01,  1.10554160e+00, -1.53298251e+00,
  4.70347921e-01, -1.56977999e+00],
[-7.75727691e-01, -9.04534034e-01, -9.25860724e-01,
 -6.07246381e-01,  8.24753060e-03],
[-5.33544468e-01, -9.04534034e-01,  1.57851664e+00,
 -4.53304338e-01, -6.87294216e-02],
[ 2.71838311e-03,  1.10554160e+00, -1.00175095e+00,
  8.52179167e-03,  1.62201435e-01],
[ 1.66340334e+00,  1.10554160e+00, -6.98190054e-01,
  2.89493510e+00,  1.23987877e+00],
[-8.10325294e-01, -9.04534034e-01,  8.95504635e-01,
 -6.84217402e-01, -3.02409455e-02],
[ 2.27602804e-01, -9.04534034e-01,  1.80618731e+00,
  1.23978324e-01, -2.99660278e-01],
[-1.36388695e+00, -9.04534034e-01,  1.12317531e+00,
 -1.29998557e+00, -1.41582609e+00],
[-1.03520972e+00, -9.04534034e-01, -6.22299831e-01,
 -8.76644956e-01,  3.93132292e-01],
[-3.18792202e-02, -9.04534034e-01,  7.43724188e-01,
  8.52179167e-03, -1.07217898e-01],
[ 1.04064648e+00, -9.04534034e-01, -6.22299831e-01,
  7.39746496e-01,  1.62476353e+00],
[-1.41578335e+00,  1.10554160e+00, -1.07764117e+00,
 -1.45392762e+00,  8.54994005e-01],
[-7.41130087e-01,  1.10554160e+00,  2.33741888e+00,
 -6.07246381e-01,  2.00689911e-01],
[ 1.24823210e+00, -9.04534034e-01, -6.98190054e-01,
  1.00914507e+00,  1.35534419e+00],
[-5.68142071e-01,  1.10554160e+00,  2.33741888e+00,
 -4.91789848e-01,  1.62201435e-01],
[-2.22166038e-01, -9.04534034e-01,  2.18563843e+00,
 -1.06934741e-01,  1.62201435e-01]
```

```
        -1.06934741e-01,  1.62201435e-01],
       [-9.31416905e-01, -9.04534034e-01, -6.22299831e-01,
        -8.38159445e-01, -3.38148754e-01]])
```

### Split the data into dependent and independent variables

In [31]:

```
data.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   Gender                  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 [32]:

```
x=data.iloc[:,0:10]
y=data['Spending Score (1-100)']
```

In [33]:

```
x
```

Out[33]:

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

200 rows × 5 columns

In [34]:

```
y
```

Out[34]:

```
0      39
1      81
2       6
3      77
4      40
       ..
195    79
196    28
197    74
198    18
```

```
198     18
199     83
Name: Spending Score (1-100), Length: 200, dtype: int64
```

**Split bold text the data for training and testing**

---

In [124]:

```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.5,random_state=0)
```

**Clustering Algorithm**

In [40]:

```
#Importing KMeans from sklearn
from sklearn.cluster import KMeans
```

**Adding cluster data with the primary dataset**

In [41]:

```
df1=data[["CustomerID","Gender","Age","Annual Income (k$)","Spending Score (1-100)"]]
X=df1[["Annual Income (k$)","Spending Score (1-100)"]]
```

In [42]:

```
wcss=[]
for i in range(1,11):
    km=KMeans(n_clusters=i)
    km.fit(X)
    wcss.append(km.inertia_)
```

In [43]:

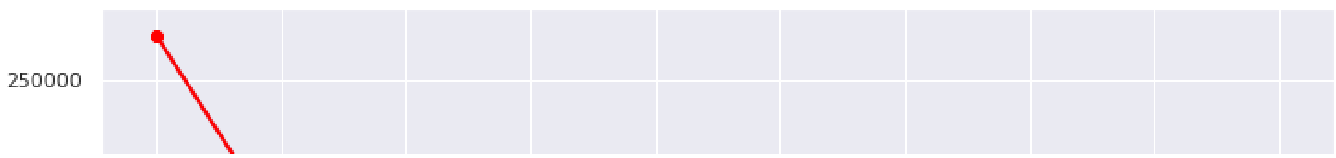
```
X.head()
```

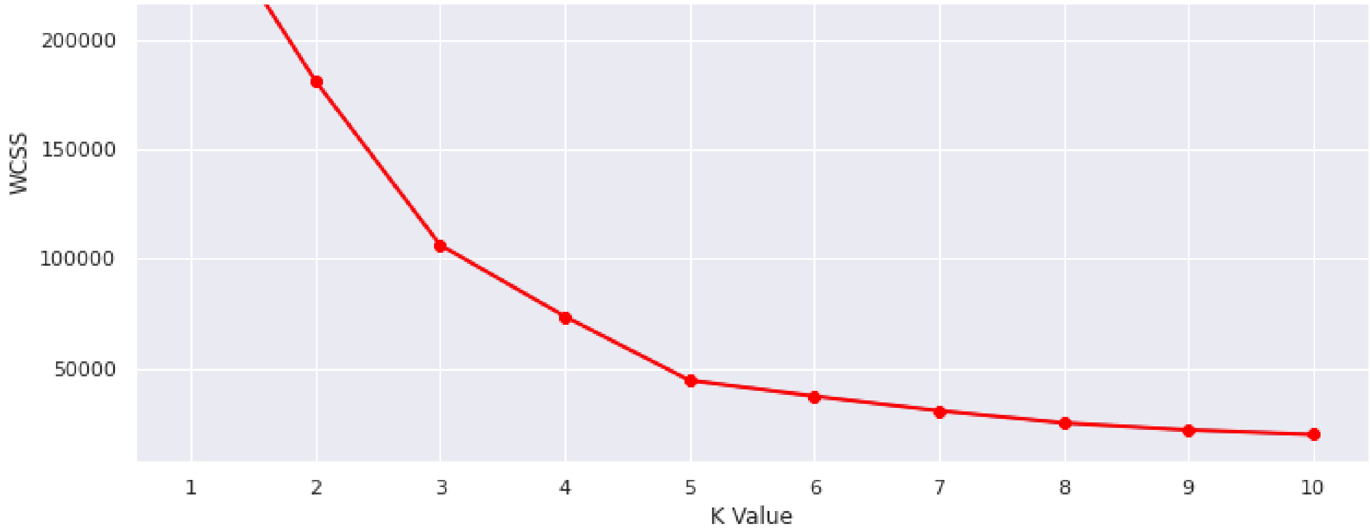
Out[43]:

| | Annual Income (k$) | Spending Score (1-100) |
|---|---|---|
| **0** | 15 | 39 |
| **1** | 15 | 81 |
| **2** | 16 | 6 |
| **3** | 16 | 77 |
| **4** | 17 | 40 |

In [44]:

```
#The elbow curve
plt.figure(figsize=(12,6))
plt.plot(range(1,11),wcss)
plt.plot(range(1,11),wcss, linewidth=2, color="red", marker ="8")
plt.xlabel("K Value")
plt.xticks(np.arange(1,11,1))
plt.ylabel("WCSS")
plt.show()
```






In [45]:

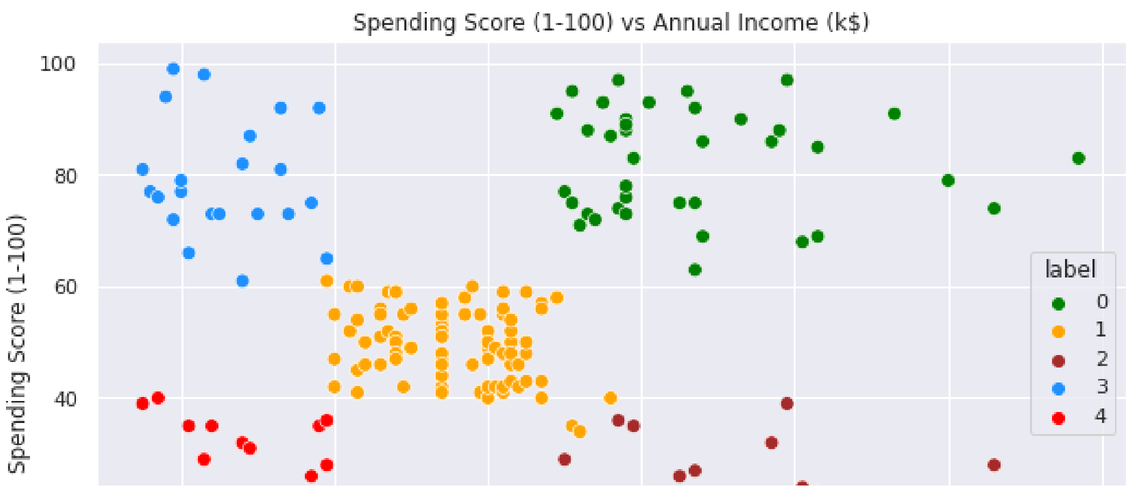
```
#Taking 5 clusters
km1=KMeans(n_clusters=5)
#Fitting the input data
km1.fit(X)
#predicting the labels of the input data
y=km1.predict(X)
#adding the labels to a column named label
df1["label"] = y
#The new dataframe with the clustering done
df1.head()
```

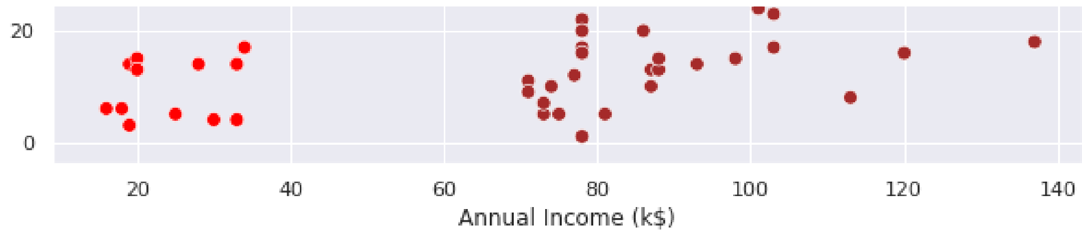
Out[45]:

| | CustomerID | Gender | Age | Annual Income (k$) | Spending Score (1-100) | label |
|---|---|---|---|---|---|---|
| **0** | 1 | 1 | 19 | 15 | 39 | 4 |
| **1** | 2 | 1 | 21 | 15 | 81 | 3 |
| **2** | 3 | 0 | 20 | 16 | 6 | 4 |
| **3** | 4 | 0 | 23 | 16 | 77 | 3 |
| **4** | 5 | 0 | 31 | 17 | 40 | 4 |

In [46]:

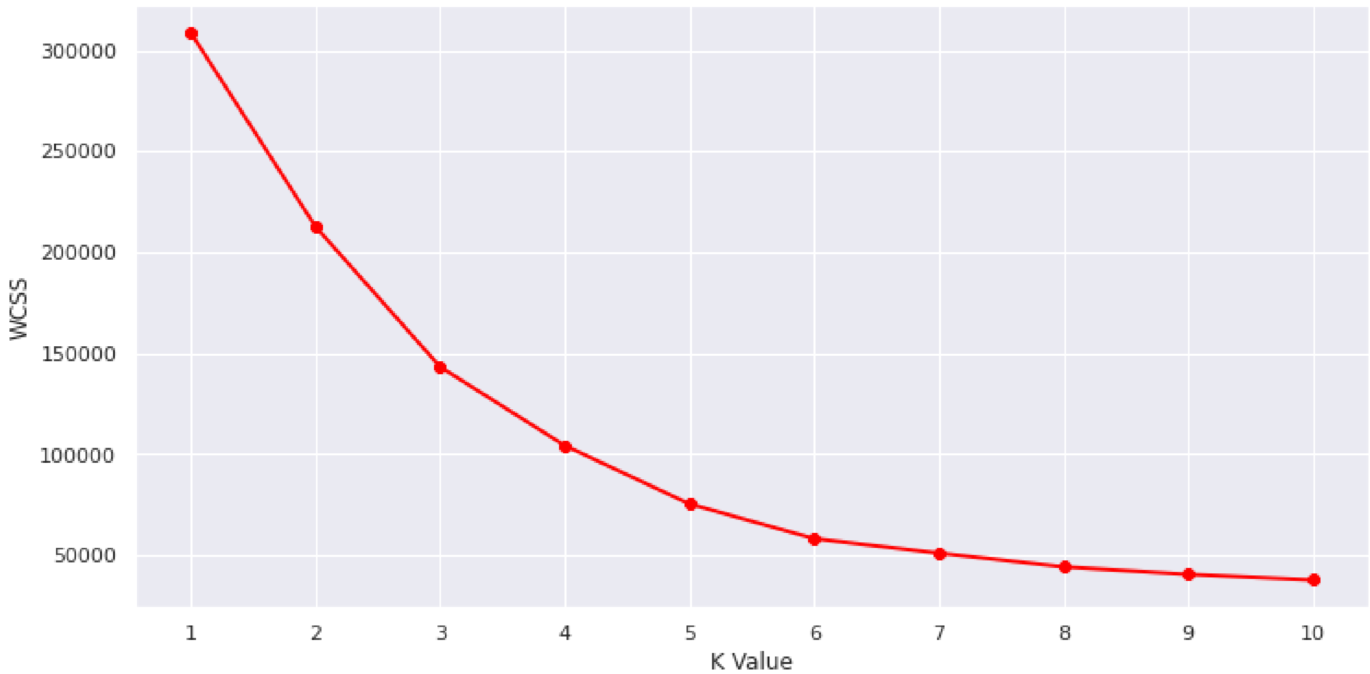
```
#Scatterplot of the clusters
plt.figure(figsize=(10,6))
sns.scatterplot(x = 'Annual Income (k$)',y = 'Spending Score (1-100)',hue="label",
                palette=['green','orange','brown','dodgerblue','red'], legend='full',da
ta = df1  ,s = 60 )
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.title('Spending Score (1-100) vs Annual Income (k$)')
plt.show()
```






In [47]:

```
#Taking the features
df2=df1[["CustomerID","Gender","Age","Annual Income (k$)","Spending Score (1-100)"]]
X2=df2[["Age","Annual Income (k$)","Spending Score (1-100)"]]
#Now we calculate the Within Cluster Sum of Squared Errors (WSS) for different values of
k.
wcss = []
for k in range(1,11):
    kmeans = KMeans(n_clusters=k, init="k-means++")
    kmeans.fit(X2)
    wcss.append(kmeans.inertia_)
plt.figure(figsize=(12,6))
plt.plot(range(1,11),wcss, linewidth=2, color="red", marker ="8")
plt.xlabel("K Value")
plt.xticks(np.arange(1,11,1))
plt.ylabel("WCSS")
plt.show()
```




In [48]:

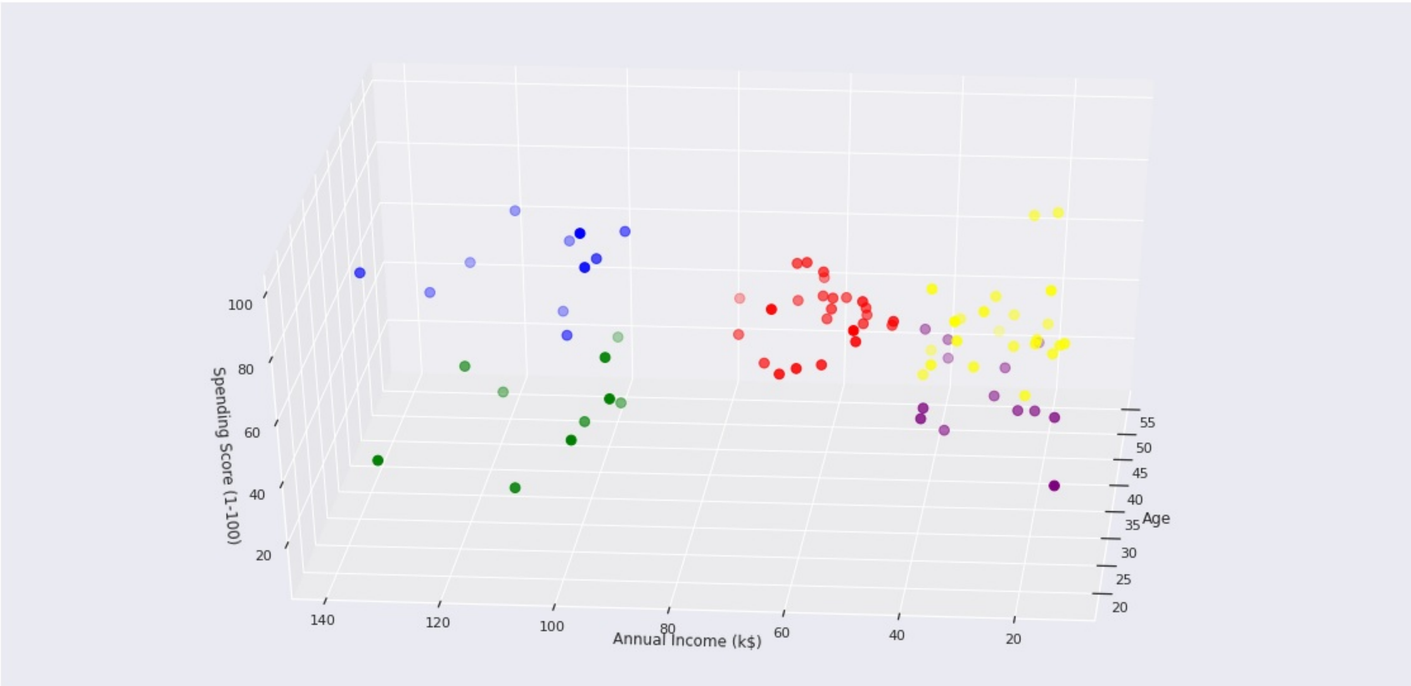
```
#We choose the k for which WSS starts to diminish
km2 = KMeans(n_clusters=5)
y2 = km.fit_predict(X2)
df2["label"] = y2
#The data with labels
df2.head()
```

Out[48]:

| | CustomerID | Gender | Age | Annual Income (k$) | Spending Score (1-100) | label |
|---|---|---|---|---|---|---|
| **0** | 1 | 1 | 19 | 15 | 39 | 0 |
| **1** | 2 | 1 | 21 | 15 | 81 | 4 |
| **2** | 3 | 0 | 20 | 16 | 6 | 7 |
| **3** | 4 | 0 | 23 | 16 | 77 | 4 |
| **4** | 5 | 0 | 31 | 17 | 40 | 0 |

In [49]:

```
#3D Plot as we did the clustering on the basis of 3 input features
fig = plt.figure(figsize=(20,10))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(df2.Age[df2.label == 0], df2["Annual Income (k$)"][df2.label == 0], df2["Spen
ding Score (1-100)"][df2.label == 0], c='purple', s=60)
ax.scatter(df2.Age[df2.label == 1], df2["Annual Income (k$)"][df2.label == 1], df2["Spen
ding Score (1-100)"][df2.label == 1], c='red', s=60)
ax.scatter(df2.Age[df2.label == 2], df2["Annual Income (k$)"][df2.label == 2], df2["Spen
ding Score (1-100)"][df2.label == 2], c='blue', s=60)
ax.scatter(df2.Age[df2.label == 3], df2["Annual Income (k$)"][df2.label == 3], df2["Spen
ding Score (1-100)"][df2.label == 3], c='green', s=60)
ax.scatter(df2.Age[df2.label == 4], df2["Annual Income (k$)"][df2.label == 4], df2["Spen
ding Score (1-100)"][df2.label == 4], c='yellow', s=60)
ax.view_init(35, 185)
plt.xlabel("Age")
plt.ylabel("Annual Income (k$)")
ax.set_zlabel('Spending Score (1-100)')
plt.show()
```




In [50]:

```
cust1=df2[df2["label"]==1]
print('Number of customer in 1st group=', len(cust1))
print('They are -', cust1["CustomerID"].values)
print("--------------------------------------------")
cust2=df2[df2["label"]==2]
print('Number of customer in 2nd group=', len(cust2))
print('They are -', cust2["CustomerID"].values)
print("--------------------------------------------")
cust3=df2[df2["label"]==0]
print('Number of customer in 3rd group=', len(cust3))
print('They are -', cust3["CustomerID"].values)
print("--------------------------------------------")
cust4=df2[df2["label"]==3]
print('Number of customer in 4th group=', len(cust4))
print('They are -', cust4["CustomerID"].values)
print("--------------------------------------------")
cust5=df2[df2["label"]==4]
print('Number of customer in 5th group=', len(cust5))
print('They are -', cust5["CustomerID"].values)
print("--------------------------------------------")
```

```
Number of customer in 1st group= 25
They are - [ 67  72  77  78  80  82  84  86  90  93  94  97  99 102 105 108 113 118
 119 120 122 123 127 147 161]
```

```
--------------------------------------------
Number of customer in 2nd group= 11
They are - [180 182 184 186 188 190 192 194 196 198 200]
--------------------------------------------
Number of customer in 3rd group= 13
They are - [ 1  5 17 19 21 27 29 39 43 45 49 50 56]
--------------------------------------------
Number of customer in 4th group= 10
They are - [181 183 185 187 189 191 193 195 197 199]
--------------------------------------------
Number of customer in 5th group= 23
They are - [ 2  4  6  8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46]
--------------------------------------------
```

#### Build the Model

In [128]:

```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
```

#### Train the dataset

In [129]:

```
classifier.fit(xtrain, ytrain)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818: Convergence
Warning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
```

Out[129]:

```
LogisticRegression(random_state=0)
```

In [130]:

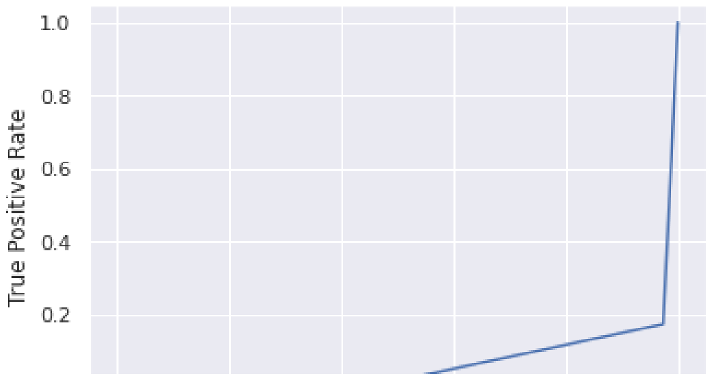
```
y_pred = classifier.predict(xtest)
```

#### Test the model

In [131]:

```
from sklearn.metrics import roc_curve
fpr, tpr,thresholds = roc_curve(ytest,  y_pred,pos_label=0)

#create ROC curve
plt.plot(fpr,tpr)
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```




#### Evaluation Metrics

In [59]:

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(ytest, y_pred)

print ("Confusion Matrix : \n", cm)
```

```
Confusion Matrix :
 [[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
```

In [60]:

```
from sklearn.metrics import accuracy_score
print ("Accuracy : ", accuracy_score(ytest, y_pred))
```

```
Accuracy :  0.0
```