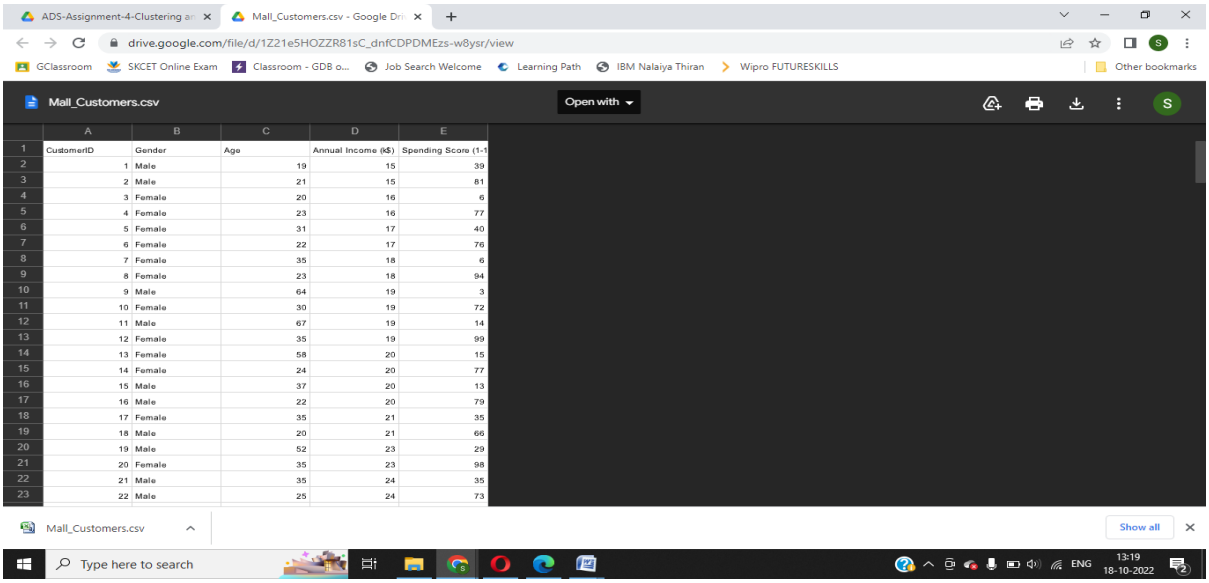
# ASSIGNMENT - 4
# CLUSTERING AND CLASSIFICATION

| | |
|---|---|
| Assignment Date | 15 October 2022 |
| Student Name | Mohana Sowdesh R |
| Student Roll Number | 727719EUCS091 |
| Maximum Marks | 2 Marks |

**Question-1:**

Download the dataset: Dataset

**Solution:**




**Question-2:**

Load the dataset into the tool

**Solution:**

```
In [3]: df = pd.read_csv("C://Users//Mohana Sowdesh//Downloads//Mall_Customers.csv")
        df.head()
```

Out[3]:

| | 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 |

**Question-3:**

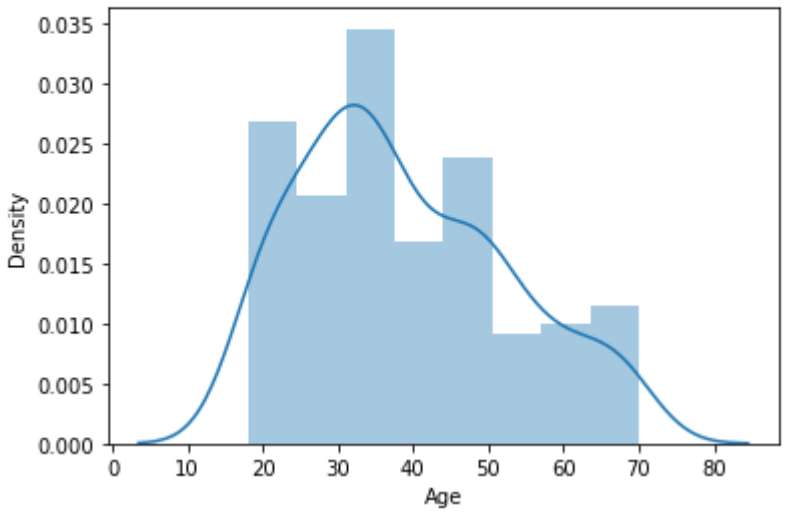
Perform Below Visualizations.

- Univariate analysis

**Solution:**

```
In [6]: sns.distplot(df.Age)
```

```
Out[6]: <AxesSubplot:xlabel='Age', ylabel='Density'>
```

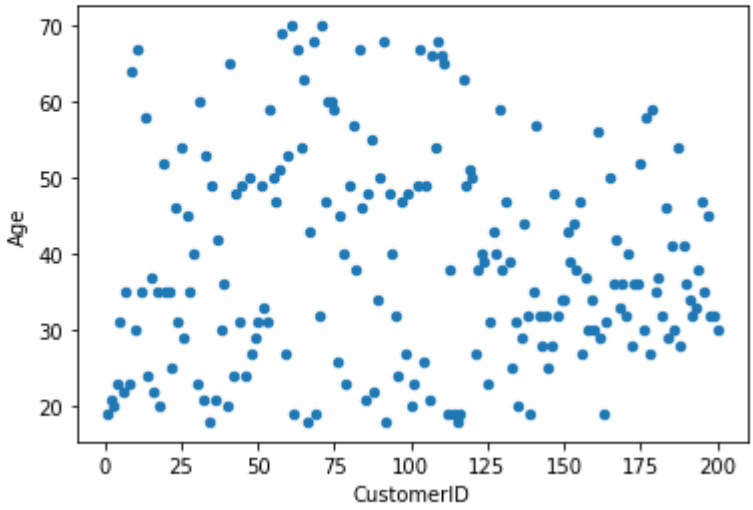



- Bi-variate analysis

**Solution:**

```
In [22]: df.plot.scatter(x='CustomerID',y='Age')
```

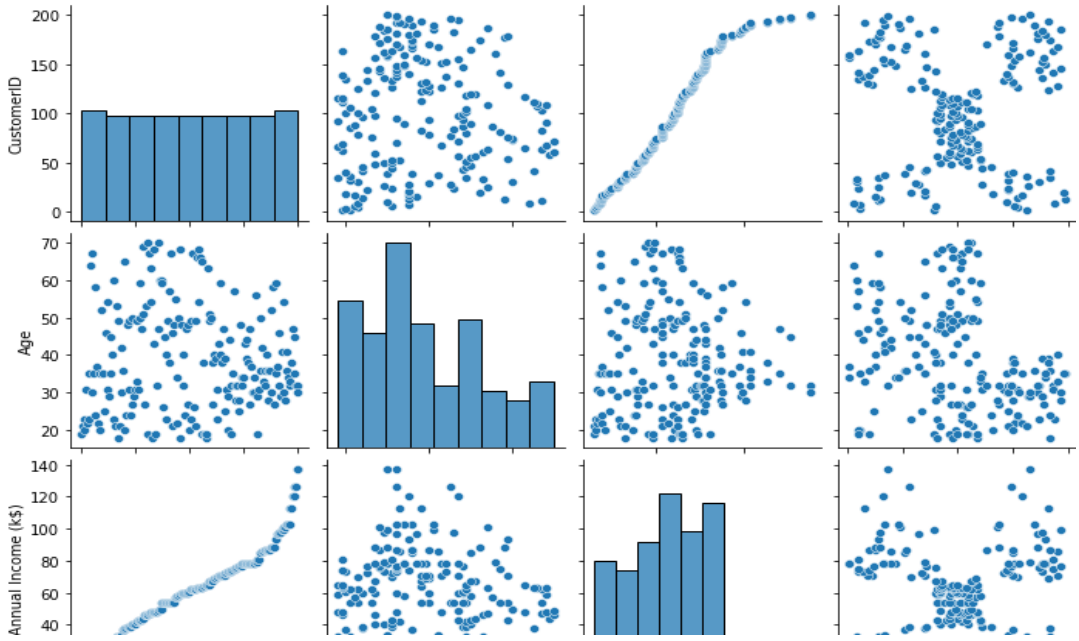
```
Out[22]: <AxesSubplot:xlabel='CustomerID', ylabel='Age'>
```



- Multi-variate analysis

**Solution:**

```
In [23]: sns.pairplot(df)
Out[23]: <seaborn.axisgrid.PairGrid at 0x233b3c62550>
```




**Question-4:**

Perform descriptive statistics on the dataset.

**Solution:**

```
In [24]: df['Age'].mean()
Out[24]: 38.85

In [25]: df['CustomerID'].median()
Out[25]: 100.5

In [26]: df['Age'].mode()
Out[26]: 0    32
         dtype: int64

In [27]: df.skew()
Out[27]: CustomerID                0.000000
         Age                       0.485569
         Annual Income (k$)        0.321843
         Spending Score (1-100)   -0.047220
         dtype: float64

In [28]: df.kurt()
Out[28]: CustomerID                -1.200000
         Age                       -0.671573
         Annual Income (k$)        -0.098487
         Spending Score (1-100)    -0.826629
         dtype: float64
```

```
In [29]: quantile = df['Age'].quantile(q=[0.75,0.25])
         quantile
```

```
Out[29]: 0.75    49.00
         0.25    28.75
         Name: Age, dtype: float64
```

```
In [30]: df.var()
```

```
Out[30]: CustomerID                 3350.000000
         Age                         195.133166
         Annual Income (k$)          689.835578
         Spending Score (1-100)      666.854271
         dtype: float64
```

```
In [31]: df.std()
```

```
Out[31]: CustomerID                 57.879185
         Age                        13.969007
         Annual Income (k$)         26.264721
         Spending Score (1-100)     25.823522
         dtype: float64
```

**Question-5:**

Check for Missing values and deal with them.

**Solution:**

```
In [32]: df.isna().any()
```

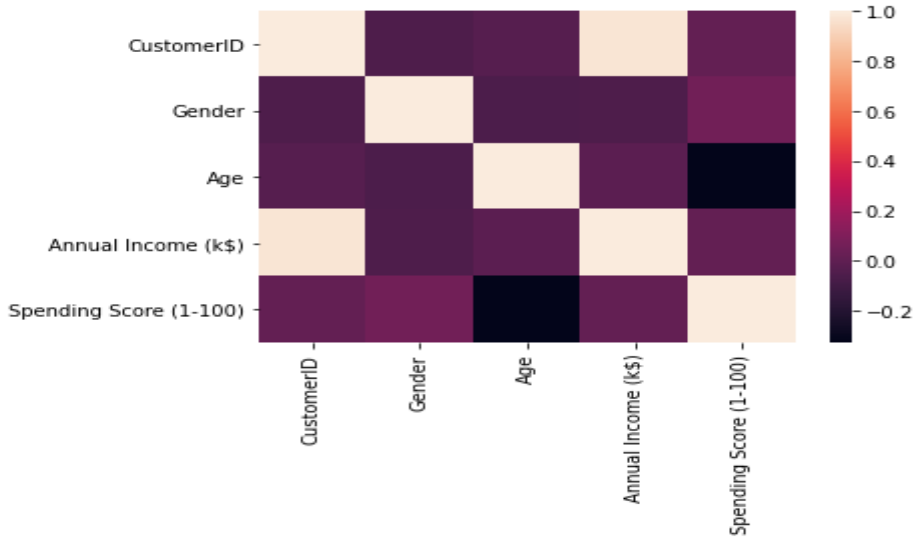
```
Out[32]: CustomerID                 False
         Gender                     False
         Age                        False
         Annual Income (k$)         False
         Spending Score (1-100)     False
         dtype: bool
```

```
In [33]: df['Age'].fillna(df['Age'].mean(),inplace=True)
         df
```

Out[33]:

| | 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 |

```
In [44]: sns.heatmap(df.corr())
Out[44]: <AxesSubplot:>
```

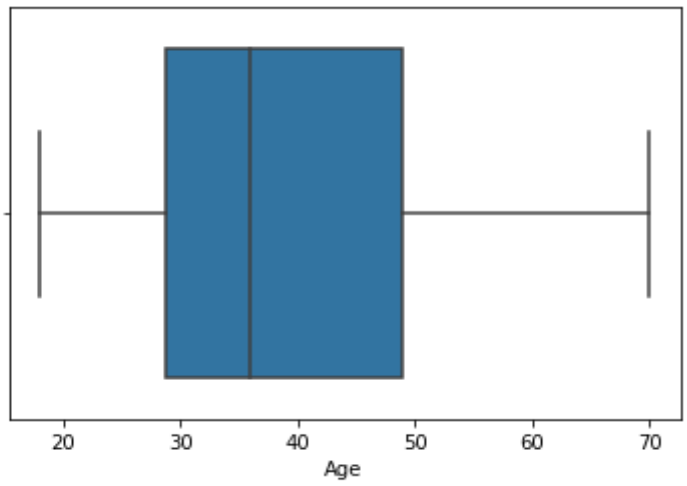



**Question-6:**

Find the outliers and replace the outliers

**Solution:**

```
In [34]: sns.boxplot(df['Age'])
Out[34]: <AxesSubplot:xlabel='Age'>
```




```
In [35]: Q1=df.Age.quantile(0.25)
         Q2=df.Age.quantile(0.75)
         IQR=Q2-Q1
         print(IQR)

         20.25
```

```
In [36]: df=df[~((df.Age<(Q1-1.5*IQR))|(df.Age>(Q2+1.5*IQR)))]
         df
```

Out[36]:

| | 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 |

**Question-7:**

Check for Categorical columns and perform encoding.

**Solution:**

```
In [38]: df['Gender'].replace({'Female':1,'Male':0},inplace=True)
         df.head()
```

Out[38]:

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

**Question-8:**

Scaling the data

**Solution:**

```
In [41]: from sklearn import preprocessing
         x = df.iloc[:, 1:3].values
         print ("\nOriginal data values : \n",  x)
```

```
Original data values :
 [[ 0 19]
 [ 0 21]
 [ 1 20]
 [ 1 23]
 [ 1 31]
 [ 1 22]
 [ 1 35]
 [ 1 23]
 [ 0 64]
 [ 1 30]
 [ 0 67]
 [ 1 35]
 [ 1 58]
 [ 1 24]
 [ 0 37]
 [ 0 22]
 [ 1 35]
```

```
In [42]: min_max_scaler = preprocessing.MinMaxScaler(feature_range =(0, 1))
         x_after_min_max_scaler = min_max_scaler.fit_transform(x)
         print ("\nAfter min max Scaling : \n", x_after_min_max_scaler)
```

```
After min max Scaling :
 [[0.         0.01923077]
 [0.         0.05769231]
 [1.         0.03846154]
 [1.         0.09615385]
 [1.         0.25      ]
 [1.         0.07692308]
 [1.         0.32692308]
 [1.         0.09615385]
 [0.         0.88461538]
 [1.         0.23076923]
 [0.         0.94230769]
 [1.         0.32692308]
 [1.         0.76923077]
 [1.         0.11538462]
 [0.         0.36538462]
 [0.         0.07692308]
```

In [43]:
```
Standardisation = preprocessing.StandardScaler()
x_after_Standardisation = Standardisation.fit_transform(x)
print ("\nAfter Standardisation : \n", x_after_Standardisation)
```

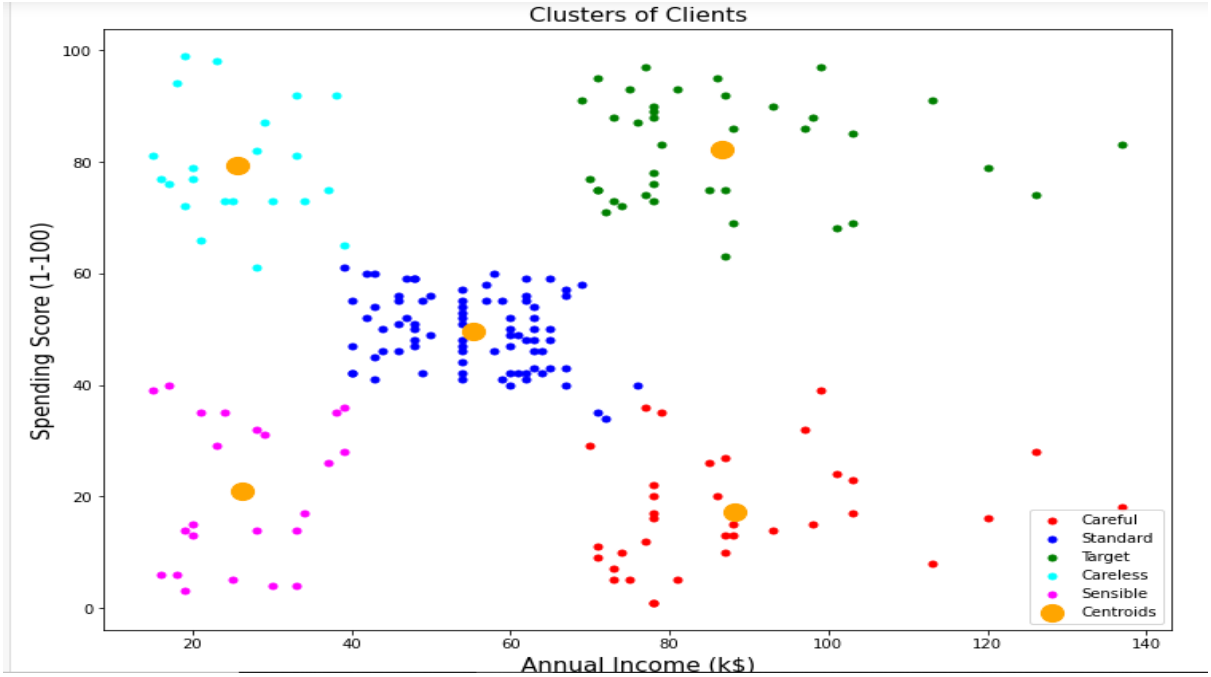
```
After Standardisation :
 [[-1.12815215 -1.42456879]
 [-1.12815215 -1.28103541]
 [ 0.88640526 -1.3528021 ]
 [ 0.88640526 -1.13750203]
 [ 0.88640526 -0.56336851]
 [ 0.88640526 -1.20926872]
 [ 0.88640526 -0.27630176]
 [ 0.88640526 -1.13750203]
 [-1.12815215  1.80493225]
 [ 0.88640526 -0.6351352 ]
 [-1.12815215  2.02023231]
 [ 0.88640526 -0.27630176]
 [ 0.88640526  1.37433211]
 [ 0.88640526 -1.06573534]
 [-1.12815215 -0.13276838]
 [-1.12815215 -1.20926872]
```

**Question-9:**

Perform any of the clustering algorithms

**Solution:**

In [52]:
```
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.cluster import KMeans
import scipy.cluster.hierarchy as sch
from sklearn.cluster import AgglomerativeClustering
target = df.iloc[:,[3,4]]

X = np.array(target)
kmeans = KMeans(n_clusters = 5, max_iter = 500, n_init = 10, random_state = 0)
kmeans_preds = kmeans.fit_predict(X)
point_size = 25
colors = ['red', 'blue', 'green', 'cyan', 'magenta']
labels = ['Careful', 'Standard', 'Target', 'Careless', 'Sensible']

plt.figure(figsize = (12,9))
for i in range(5):
    plt.scatter(X[kmeans_preds == i,0], X[kmeans_preds == i,1], s = point_size, c = colors[i], label = labels[i])

plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1], s = 200, c = 'orange', label = 'Centroids')
plt.title('Clusters of Clients',fontsize=15)
plt.xlabel('Annual Income (k$)',fontsize=15)
plt.ylabel('Spending Score (1-100)',fontsize=15)
plt.legend(loc = 'best')
plt.show()
```



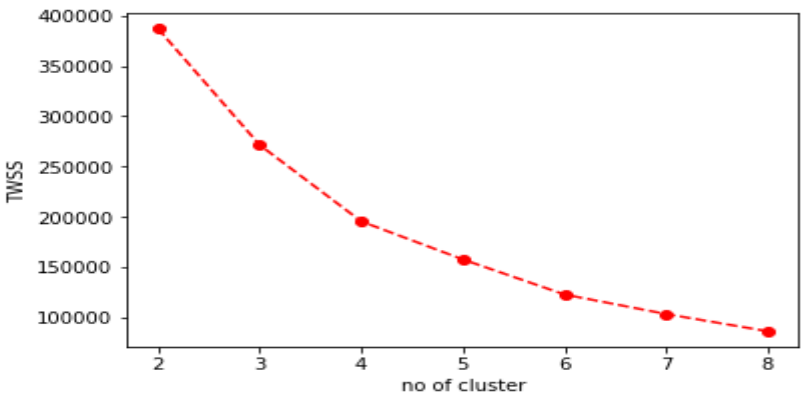

In [54]:
```
TWSS=[]
k=list(range(2,9))

for i in k:
    kmeans=KMeans(n_clusters=i,init='k-means++')
    kmeans.fit(df)
    TWSS.append(kmeans.inertia_)
```

In [55]:
```
TWSS
```

Out[55]:
```
[387065.7137713772,
 271396.5629660314,
 195393.50384615397,
 157534.71366300373,
 122670.55266775498,
 103254.37701808,
 86096.56850655384]
```

In [56]:
```
plt.plot(k,TWSS,'ro--')
plt.xlabel('no of cluster')
plt.ylabel('TWSS')
```

Out[56]:
```
Text(0, 0.5, 'TWSS')
```




```
In [58]: model=KMeans(n_clusters=4)

         model.fit(df)

Out[58]: KMeans(n_clusters=4)
```

**Question-10:**

Add the cluster data with the primary dataset

**Solution:**

```
In [59]: model.labels_

Out[59]: array([2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
                2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
                2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 1, 1, 2, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 1, 3, 1, 3, 0, 3, 0, 3,
                0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3,
                0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3,
                0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3,
                0, 3])
```

```
In [62]: mb=pd.Series(model.labels_)

In [63]: df.head(3)
```

Out[63]:

| | CustomerID | Gender | Age | Annual Income (k$) | Spending Score (1-100) |
|---|---|---|---|---|---|
| **0** | 1 | 0 | 19 | 15 | 39 |
| **1** | 2 | 0 | 21 | 15 | 81 |
| **2** | 3 | 1 | 20 | 16 | 6 |

```
In [64]: df['clust']=mb

In [65]: df.head(3)
```

Out[65]:

| | CustomerID | Gender | Age | Annual Income (k$) | Spending Score (1-100) | clust |
|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 19 | 15 | 39 | 2 |
| **1** | 2 | 0 | 21 | 15 | 81 | 2 |
| **2** | 3 | 1 | 20 | 16 | 6 | 2 |

```
In [66]: df.tail(3)
```

Out[66]:

| | CustomerID | Gender | Age | Annual Income (k$) | Spending Score (1-100) | clust |
|---|---|---|---|---|---|---|
| **197** | 198 | 0 | 32 | 126 | 74 | 3 |
| **198** | 199 | 0 | 32 | 137 | 18 | 0 |
| **199** | 200 | 0 | 30 | 137 | 83 | 3 |

**Question-11:**

Split the data into dependent and independent variables.

**Solution:**

```
In [67]: data_main= pd.get_dummies(df,columns=['Gender'])
         data_main
```

Out[67]:

| | CustomerID | Age | Annual Income (k$) | Spending Score (1-100) | clust | Gender_0 | Gender_1 |
|---|---|---|---|---|---|---|---|
| **0** | 1 | 19 | 15 | 39 | 2 | 1 | 0 |
| **1** | 2 | 21 | 15 | 81 | 2 | 1 | 0 |
| **2** | 3 | 20 | 16 | 6 | 2 | 0 | 1 |
| 3 | 4 | 23 | 16 | 77 | 2 | 0 | 1 |
| **4** | 5 | 31 | 17 | 40 | 2 | 0 | 1 |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **195** | 196 | 35 | 120 | 79 | 3 | 0 | 1 |
| **196** | 197 | 45 | 126 | 28 | 0 | 0 | 1 |
| **197** | 198 | 32 | 126 | 74 | 3 | 1 | 0 |
| **198** | 199 | 32 | 137 | 18 | 0 | 1 | 0 |
| **199** | 200 | 30 | 137 | 83 | 3 | 1 | 0 |

200 rows × 7 columns

```
In [68]: y = data_main['Age']
         y
```

```
Out[68]: 0      19
         1      21
         2      20
         3      23
         4      31
                ..
         195    35
         196    45
         197    32
         198    32
         199    30
         Name: Age, Length: 200, dtype: int64
```

```
In [69]: x = data_main.drop(columns='Age',axis=1)
         x.head()
```

Out[69]:

| | CustomerID | Annual Income (k$) | Spending Score (1-100) | clust | Gender_0 | Gender_1 |
|---|---|---|---|---|---|---|
| **0** | 1 | 15 | 39 | 2 | 1 | 0 |
| **1** | 2 | 15 | 81 | 2 | 1 | 0 |
| **2** | 3 | 16 | 6 | 2 | 0 | 1 |
| 3 | 4 | 16 | 77 | 2 | 0 | 1 |
| **4** | 5 | 17 | 40 | 2 | 0 | 1 |

**Question-12:**

Split the data into training and testing

**Solution:**

```
In [70]: from sklearn.model_selection import train_test_split
         x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

**Question-13:**

Build the Model

**Solution:**

```
In [75]: from sklearn.linear_model import LinearRegression
         regressor=LinearRegression()
         regressor.fit(x_train,y_train)
```

```
Out[75]: LinearRegression()
```

**Question-14:**

Train the Model

**Solution:**

```
In [71]: x_train
```

Out[71]:

| | CustomerID | Annual Income (k$) | Spending Score (1-100) | clust | Gender_0 | Gender_1 |
|---|---|---|---|---|---|---|
| **134** | 135 | 73 | 5 | 0 | 1 | 0 |
| **66** | 67 | 48 | 50 | 1 | 0 | 1 |
| **26** | 27 | 28 | 32 | 2 | 0 | 1 |
| **113** | 114 | 64 | 46 | 1 | 1 | 0 |
| **168** | 169 | 87 | 27 | 0 | 0 | 1 |
| **...** | ... | ... | ... | ... | ... | ... |
| **67** | 68 | 48 | 48 | 1 | 0 | 1 |
| **192** | 193 | 113 | 8 | 0 | 1 | 0 |
| **117** | 118 | 65 | 59 | 1 | 0 | 1 |
| **47** | 48 | 40 | 47 | 2 | 0 | 1 |
| **172** | 173 | 87 | 10 | 0 | 1 | 0 |

160 rows × 6 columns

```
In [72]: y_train
```

Out[72]:

```
134    20
66     43
26     45
113    19
168    36
```

**Question-15:**

Test the Model

**Solution:**

```
In [73]: x_test
```

Out[73]:

| | CustomerID | Annual Income (k$) | Spending Score (1-100) | clust | Gender_0 | Gender_1 |
|---|---|---|---|---|---|---|
| **18** | 19 | 23 | 29 | 2 | 1 | 0 |
| **170** | 171 | 87 | 13 | 0 | 1 | 0 |
| **107** | 108 | 63 | 46 | 1 | 1 | 0 |
| **98** | 99 | 61 | 42 | 1 | 1 | 0 |
| **177** | 178 | 88 | 69 | 3 | 1 | 0 |

```
In [74]: y_test
```

Out[74]:

```
18     52
170    40
107    54
98     48
177    27
182    46
5      22
146    48
12     58
```

**Question-16:**

Measure the performance using Evaluation Metrics.

**Solution:**

In [18]:
```
#Elbow method
from sklearn.cluster import KMeans
from sklearn import preprocessing

data_x = df.iloc[:, 2:4]
data_x.head()
x_array =  np.array(data_x)

scaler = preprocessing.MinMaxScaler()
x_scaled = scaler.fit_transform(x_array)
x_scaled
Sum_of_squared_distances =[]
K = range(1,15)
for k in K:
    km =KMeans(n_clusters =k)
    km =km.fit(x_scaled)
    Sum_of_squared_distances.append(km.inertia_)

plt.plot(K, Sum_of_squared_distances, 'bx-')
plt.xlabel('k')
plt.ylabel('SSE')
plt.title('Elbow Method For Optimal k')
plt.show()
```