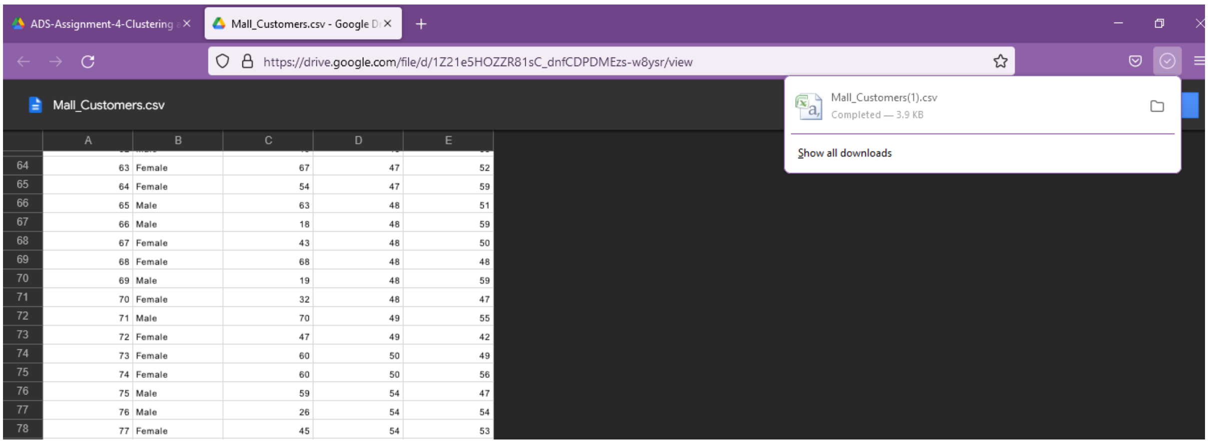
<u>Assignment - 4</u>
<u>Clustering And Classification</u>

Assignment Date	15 October 2022
Student Name	Logeshkumar R
Student Roll Number	727719EUCS074
Maximum Marks	2 Marks

Question-1:

Download the dataset: Dataset

Solution:



Question-2:

Load the dataset into the tool

Solution:

In [2]:
```python
d = pd.read_csv("E://Mall_Customers(1).csv")
d.head()
```

Out[2]:

	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

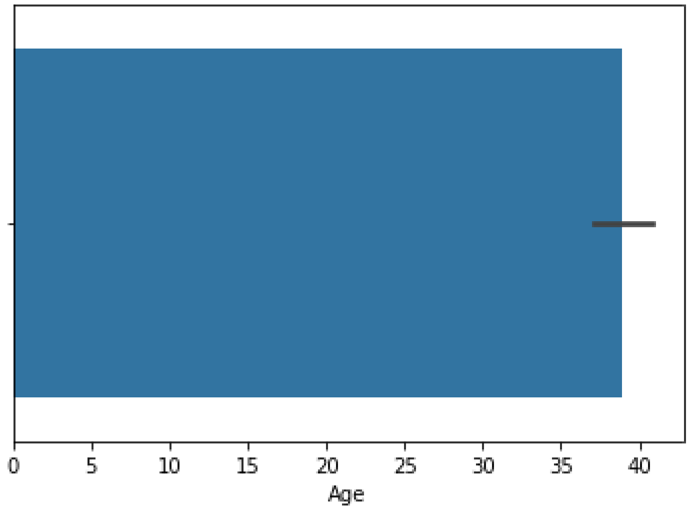
Question-3:

Perform Below Visualizations.

- Univariate analysis

Solution:

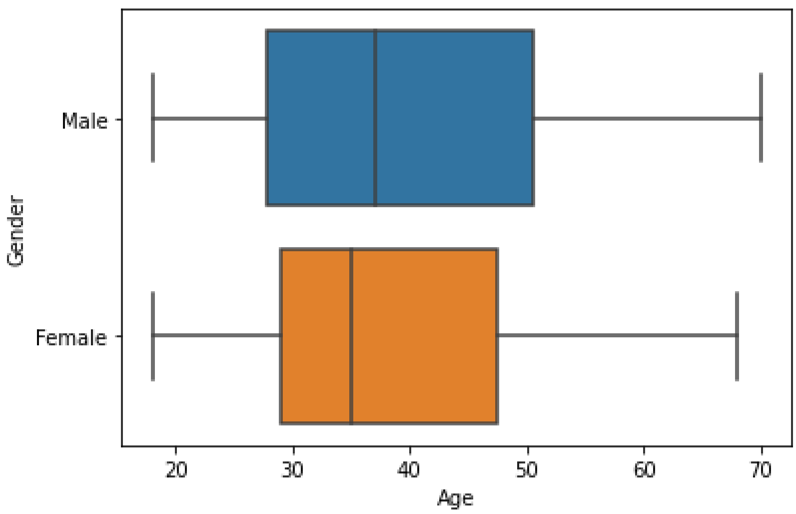
```
In [4]: sns.barplot(d.Age)
Out[4]: <AxesSubplot:xlabel='Age'>
```



- Bi-variate analysis

Solution:

```
In [14]: sns.boxplot(y=d.Gender,x=d.Age)
Out[14]: <AxesSubplot:xlabel='Age', ylabel='Gender'>
```

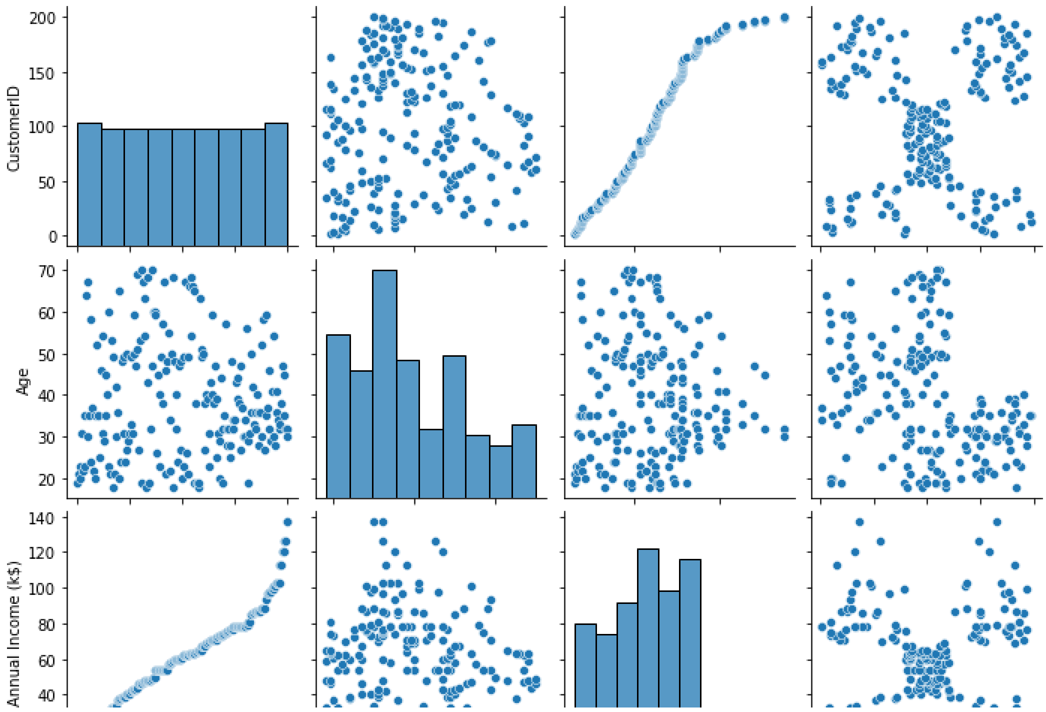


- Multi-variate analysis

Solution:

```
In [15]: sns.pairplot(d)
```

Out[15]: <seaborn.axisgrid.PairGrid at 0x1f19d20da30>



Question-4:

Perform descriptive statistics on the dataset.

Solution:

```
In [16]: d['CustomerID'].mean()
Out[16]: 100.5

In [18]: d['Age'].median()
Out[18]: 36.0

In [19]: d['Gender'].mode()
Out[19]: 0    Female
         dtype: object

In [20]: d.skew()
Out[20]: CustomerID                0.000000
         Age                       0.485569
         Annual Income (k$)        0.321843
         Spending Score (1-100)   -0.047220
         dtype: float64

In [21]: d.kurt()
Out[21]: CustomerID               -1.200000
         Age                      -0.671573
         Annual Income (k$)       -0.098487
         Spending Score (1-100)   -0.826629
         dtype: float64
```

```
In [22]: d.std()

Out[22]: 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 [23]: d.isna().any()

Out[23]: CustomerID                False
         Gender                    False
         Age                       False
         Annual Income (k$)        False
         Spending Score (1-100)    False
         dtype: bool
```

```
In [25]: d['Age'].fillna(d['Age'].mean(),inplace=True)
         d
```

Out[25]:

	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

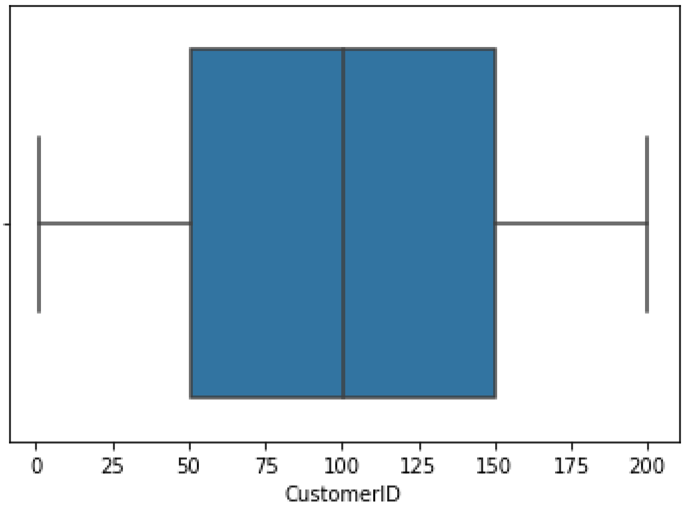
Question-6:

Find the outliers and replace the outliers

Solution:

```
In [26]: sns.boxplot(d['CustomerID'])
```

```
Out[26]: <AxesSubplot:xlabel='CustomerID'>
```




```
In [28]: Q1=d.CustomerID.quantile(0.25)
         Q2=d.CustomerID.quantile(0.75)
         IQR=Q2-Q1
         print(IQR)
```

```
99.5
```

```
In [30]: d=d[~((d.CustomerID<(Q1-1.5*IQR))|(d.CustomerID>(Q2+1.5*IQR)))]
         d
```

Out[30]:

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

Out[31]:

	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 [33]: from sklearn import preprocessing
         x = d.iloc[:, 2:4].values
         print ("\nOriginal data values : \n",  x)
```

```
[ 19  48]
[ 32  48]
[ 70  49]
[ 47  49]
[ 60  50]
[ 60  50]
[ 59  54]
[ 26  54]
[ 45  54]
[ 40  54]
[ 23  54]
[ 49  54]
[ 57  54]
[ 38  54]
[ 67  54]
[ 46  54]
[ 21  54]
```

In [34]:
```
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 :
 [[0.01923077 0.        ]
 [0.05769231 0.        ]
 [0.03846154 0.00819672]
 [0.09615385 0.00819672]
 [0.25       0.01639344]
 [0.07692308 0.01639344]
 [0.32692308 0.02459016]
 [0.09615385 0.02459016]
 [0.88461538 0.03278689]
 [0.23076923 0.03278689]
 [0.94230769 0.03278689]
 [0.32692308 0.03278689]
 [0.76923077 0.04098361]
 [0.11538462 0.04098361]
 [0.36538462 0.04098361]
 [0.07692308 0.04098361]
 [0.32692308 0.04918033]
```

In [35]:
```
Standardisation = preprocessing.StandardScaler()
x_after_Standardisation = Standardisation.fit_transform(x)
print ("\nAfter Standardisation : \n", x_after_Standardisation)
```

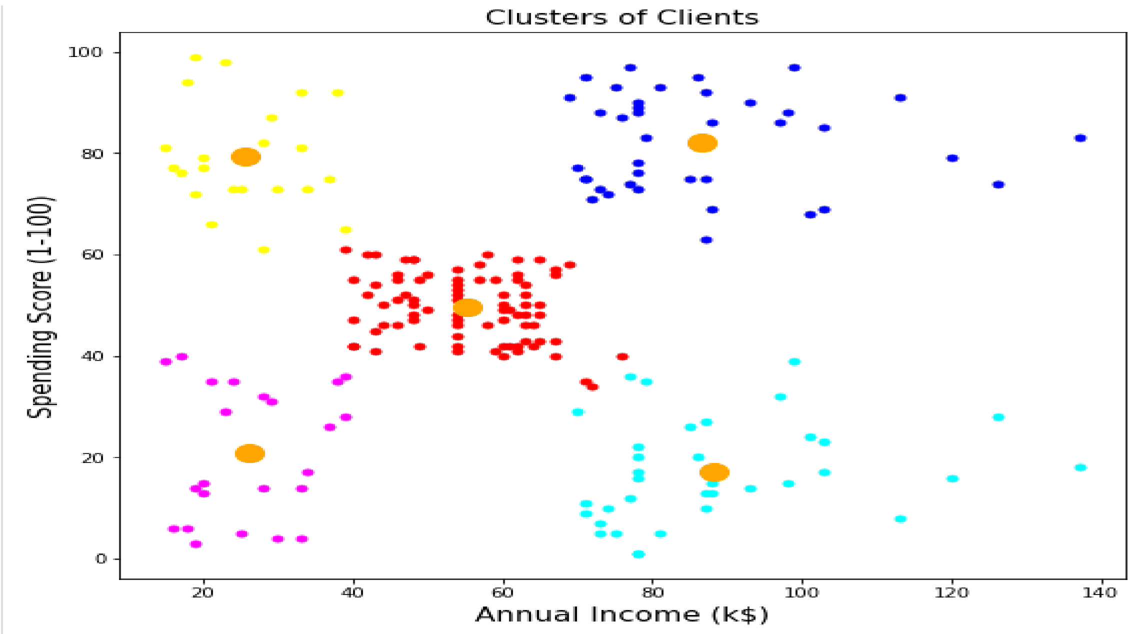
```
After Standardisation :
 [[-1.42456879 -1.73899919]
 [-1.28103541 -1.73899919]
 [-1.3528021  -1.70082976]
 [-1.13750203 -1.70082976]
 [-0.56336851 -1.66266033]
 [-1.20926872 -1.66266033]
 [-0.27630176 -1.62449091]
 [-1.13750203 -1.62449091]
 [ 1.80493225 -1.58632148]
 [-0.6351352  -1.58632148]
 [ 2.02023231 -1.58632148]
 [-0.27630176 -1.58632148]
 [ 1.37433211 -1.54815205]
 [-1.06573534 -1.54815205]
 [-0.13276838 -1.54815205]
```

Question-9:

Perform any of the clustering algorithms

Solution:

In [37]:
```
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 = d.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 = ['cyan', 'red', 'blue', 'yellow', 'magenta']
labels = ['Careful', 'Standard', 'Target', 'Careless', 'Sensible']
plt.figure(figsize = (9,8))

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.show()
```



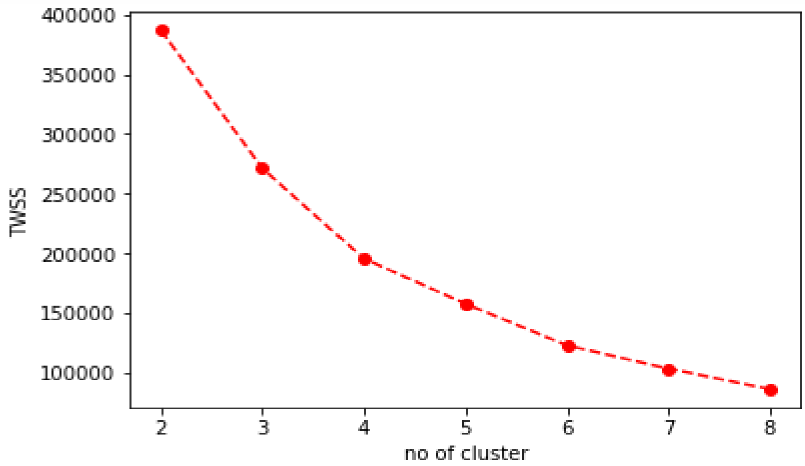
In [44]:
```python
TWSS=[]
k=list(range(2,9))

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

In [45]:
```python
TWSS
```

Out[45]:
```
[387065.7137713772,
 271384.50878286787,
 195401.19855991477,
 157620.97147979145,
 122637.55796110148,
 103233.09788480632,
 86028.09935619931]
```

In [46]:
```python
plt.plot(k,TWSS,'ro--')
plt.xlabel('no of cluster')
plt.ylabel('TWSS')
```

Out[46]:
```
Text(0, 0.5, 'TWSS')
```



In [48]:
```python
model=KMeans(n_clusters=4)
model.fit(d)
```

Out[48]:
```
KMeans(n_clusters=4)
```

Question-10:

Add the cluster data with the primary dataset

Solution:

```
In [49]: model.labels_

Out[49]: array([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, 0, 0, 1, 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, 2, 0, 2, 0, 2, 3, 2, 3, 2,
                3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2,
                3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2,
                3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2,
                3, 2])
```

```
In [50]: mb=pd.Series(model.labels_)
```

```
In [51]: d['clust']=mb
```

```
In [52]: d.head(3)
```

Out[52]:

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

```
In [53]: d.tail(3)
```

Out[53]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)	clust
197	198	0	32	126	74	2
198	199	0	32	137	18	3
199	200	0	30	137	83	2

Question-11:

Split the data into dependent and independent variables.

Solution:

In [54]:
```
dmf= pd.get_dummies(d,columns=['Gender'])
dmf
```

Out[54]:

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

200 rows × 7 columns

In [56]:
```
y = d['Age']
y
```

Out[56]:
```
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 [58]:
```
x = dmf.drop(columns='Age',axis=1)
x.head()
```

Out[58]:

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

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

```
Out[72]: 134    20
         66     43
         26     45
         113    19
         168    36
                ..
         67     68
         192    33
         117    49
         47     27
         172    36
         Name: Age, Length: 160, dtype: int64
```

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
182	183	98	15	0	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
         152    44
         61     19
         125    31
```

Question-16:

Measure the performance using Evaluation Metrics.

Solution:

In [5]:
```
from sklearn.cluster import KMeans
from sklearn import preprocessing

data_x = d.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()
```