Import libraries

In [4]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [ ]:

```
<h1 style="color:black;fontsize;30px;">Read the Dataset</h1>
```

In [5]:

```
df=pd.read_csv('Mall_Customers.csv')
```

In [6]:

```
df
```

Out[6]:

	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 [7]:

```
df.shape
```

Out[7]:

```
(200, 5)
```

In [8]:

```
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   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 [9]:

```
df.describe()
```

Out[9]:

	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 [10]:

```
df.duplicated().sum()
```

Out[10]:

```
0
```

checking null values

In [11]:

```
df.isnull().sum()
```

Out[11]:

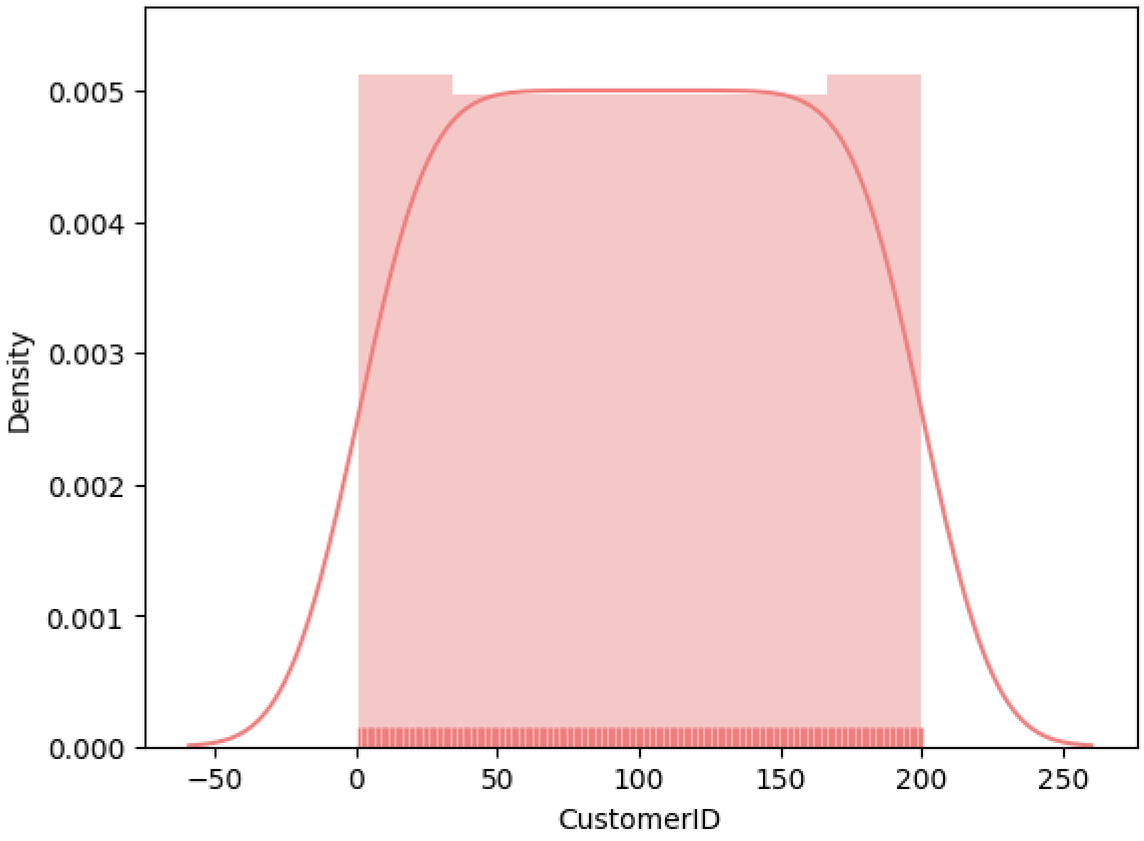
```
CustomerID                0
Gender                    0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

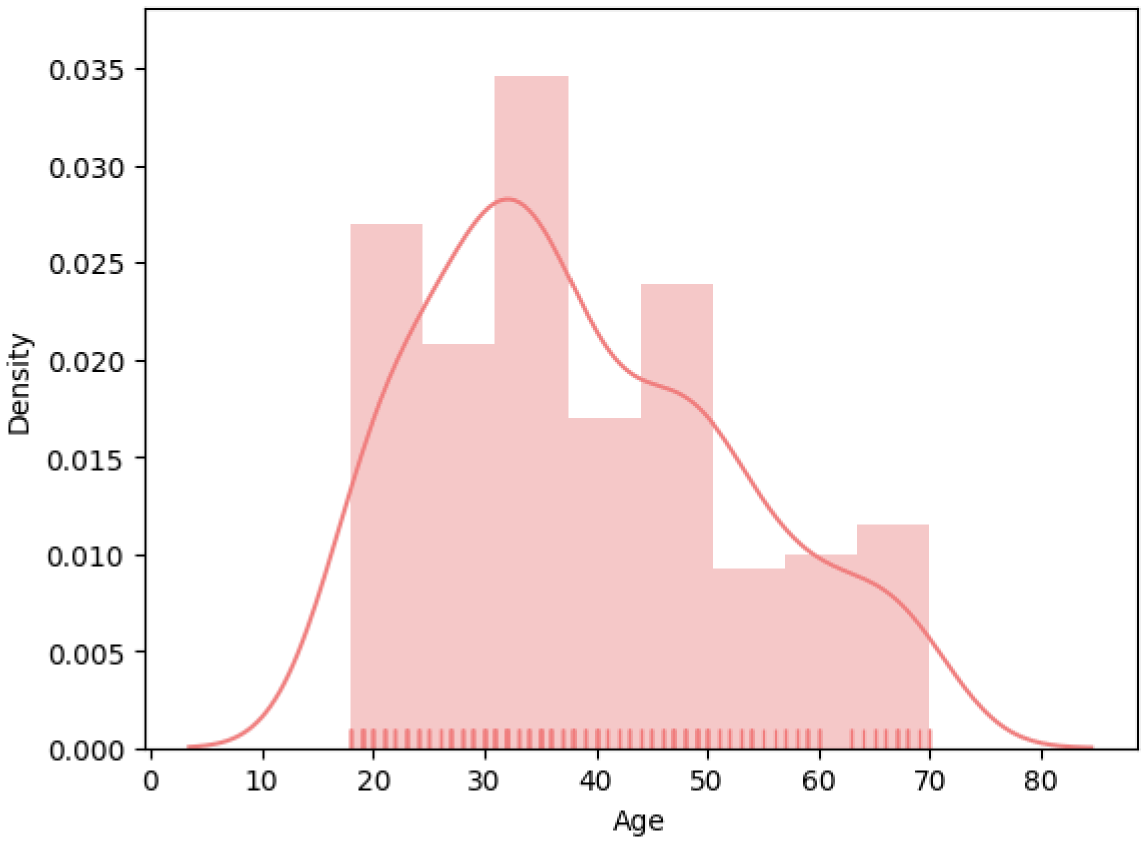
plotting each columns

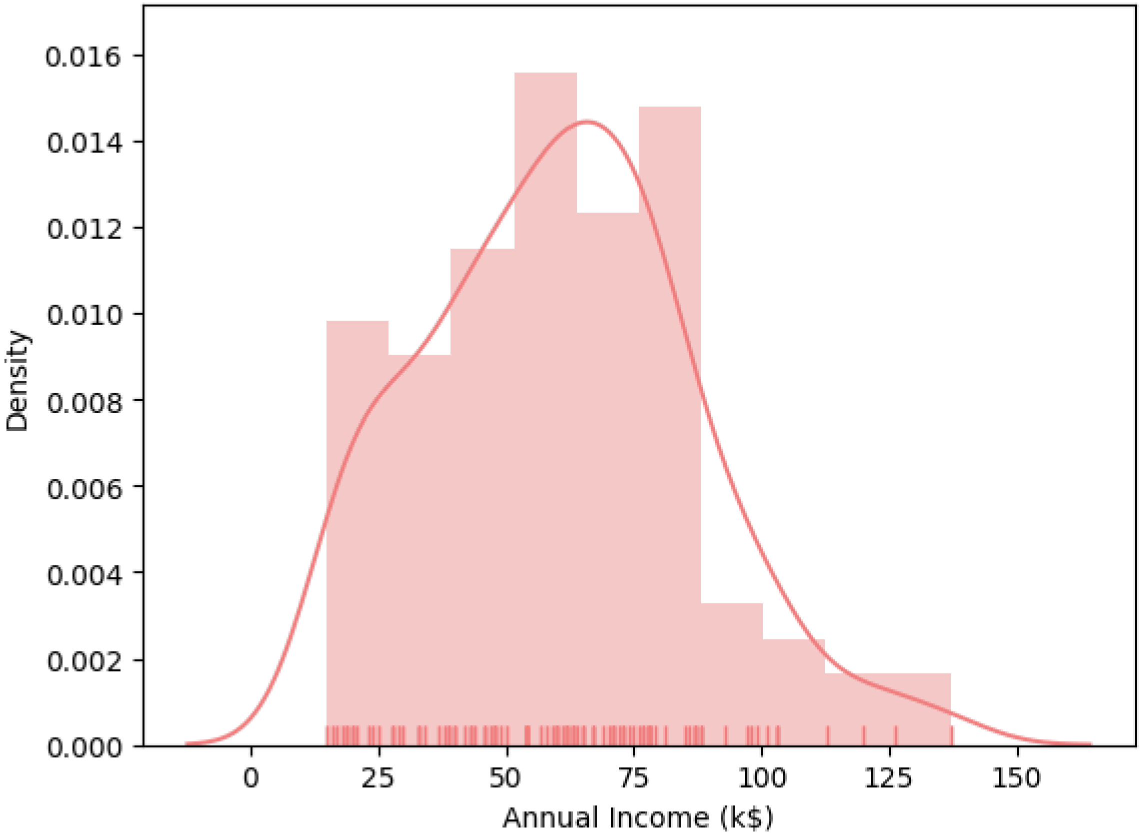
In [12]:

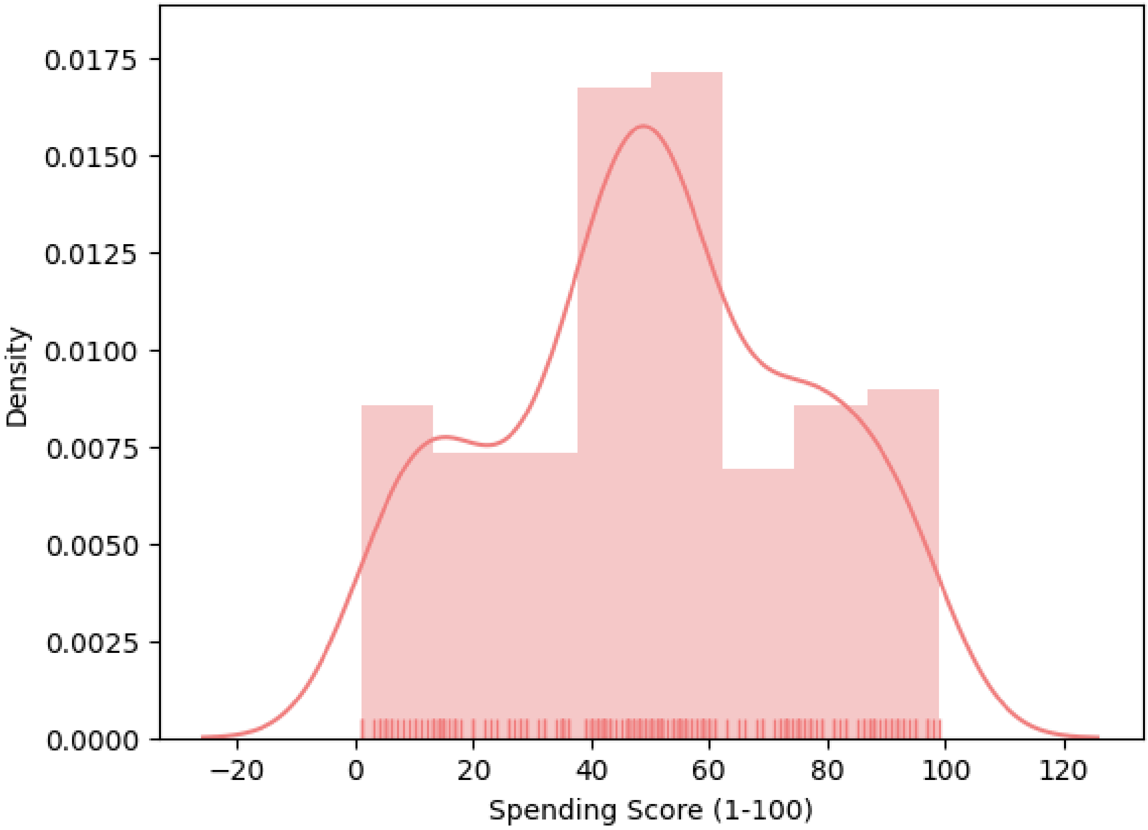
```
def distributionPlot(columnName):
    if not columnName == 'Gender':
        plt.figure()
        sns.distplot(df[columnName], color="lightcoral", rug=True);
for column in df.columns:
    distributionPlot(column)
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_1020\2872842887.py:4: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df[columnName], color="lightcoral", rug=True);
C:\Users\DELL\AppData\Local\Temp\ipykernel_1020\2872842887.py:4: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df[columnName], color="lightcoral", rug=True);
C:\Users\DELL\AppData\Local\Temp\ipykernel_1020\2872842887.py:4: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df[columnName], color="lightcoral", rug=True);
C:\Users\DELL\AppData\Local\Temp\ipykernel_1020\2872842887.py:4: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df[columnName], color="lightcoral", rug=True);
```


Density
0.005
0.004
0.003
0.002
0.001
0.000
−50
0
50
100
150
200
250
CustomerID


Density
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
10
20
30
40
50
60
70
80
Age





In [13]:
```
df['Gender']=df['Gender'].map({'Male':1,'Female':0})
df.head()
```

Out[13]:

	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

MinMax Normalization

In [14]:

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
df_scaled = pd.DataFrame(scaler.fit_transform(df), columns=df.columns)
df_scaled
```

Out[14]:

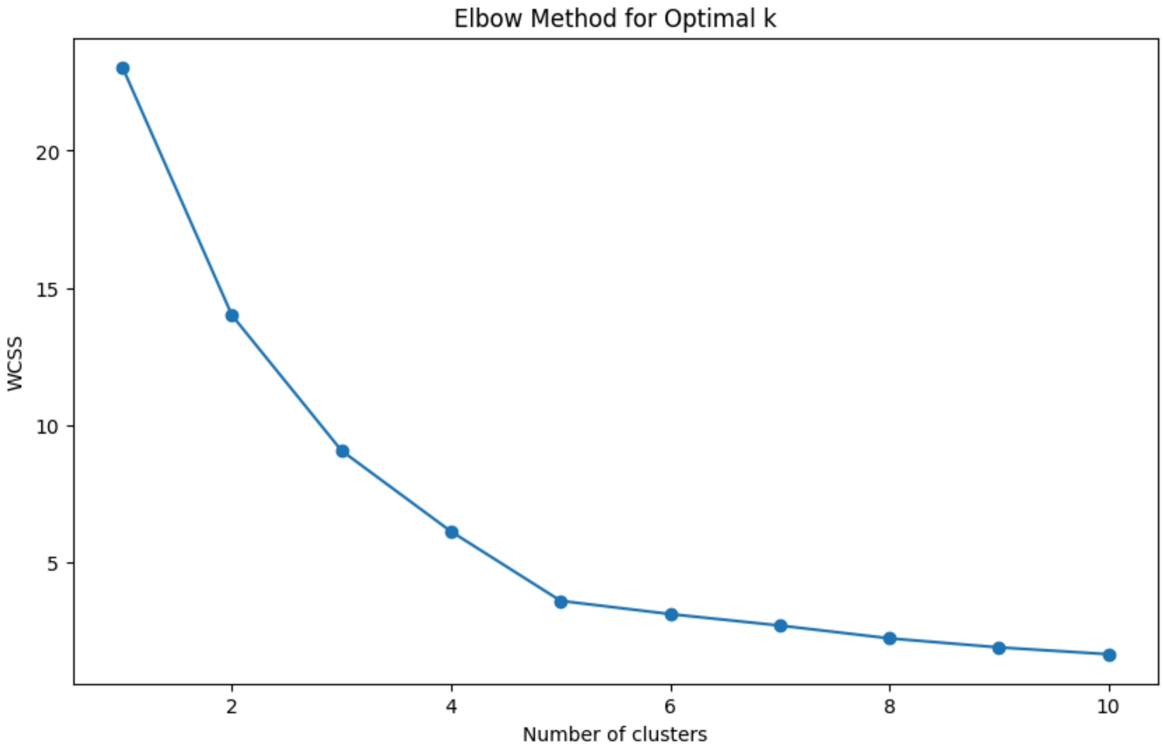
	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	0.000000	1.0	0.019231	0.000000	0.387755
1	0.005025	1.0	0.057692	0.000000	0.816327
2	0.010050	0.0	0.038462	0.008197	0.051020
3	0.015075	0.0	0.096154	0.008197	0.775510
4	0.020101	0.0	0.250000	0.016393	0.397959
...	...	...	...	...	...
195	0.979899	0.0	0.326923	0.860656	0.795918
196	0.984925	0.0	0.519231	0.909836	0.275510
197	0.989950	1.0	0.269231	0.909836	0.744898
198	0.994975	1.0	0.269231	1.000000	0.173469
199	1.000000	1.0	0.230769	1.000000	0.836735

200 rows × 5 columns

Elbow Method

In [15]:

```
from sklearn.cluster import KMeans

# Extract features
X = df_scaled[['Annual Income (k$)', 'Spending Score (1-100)']]

# Determine the optimal number of clusters using the elbow method
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, ran
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)

# Plot the WCSS values to visualize the elbow
plt.figure(figsize=(10, 6))
plt.plot(range(1, 11), wcss, marker='o')
plt.title('Elbow Method for Optimal k')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



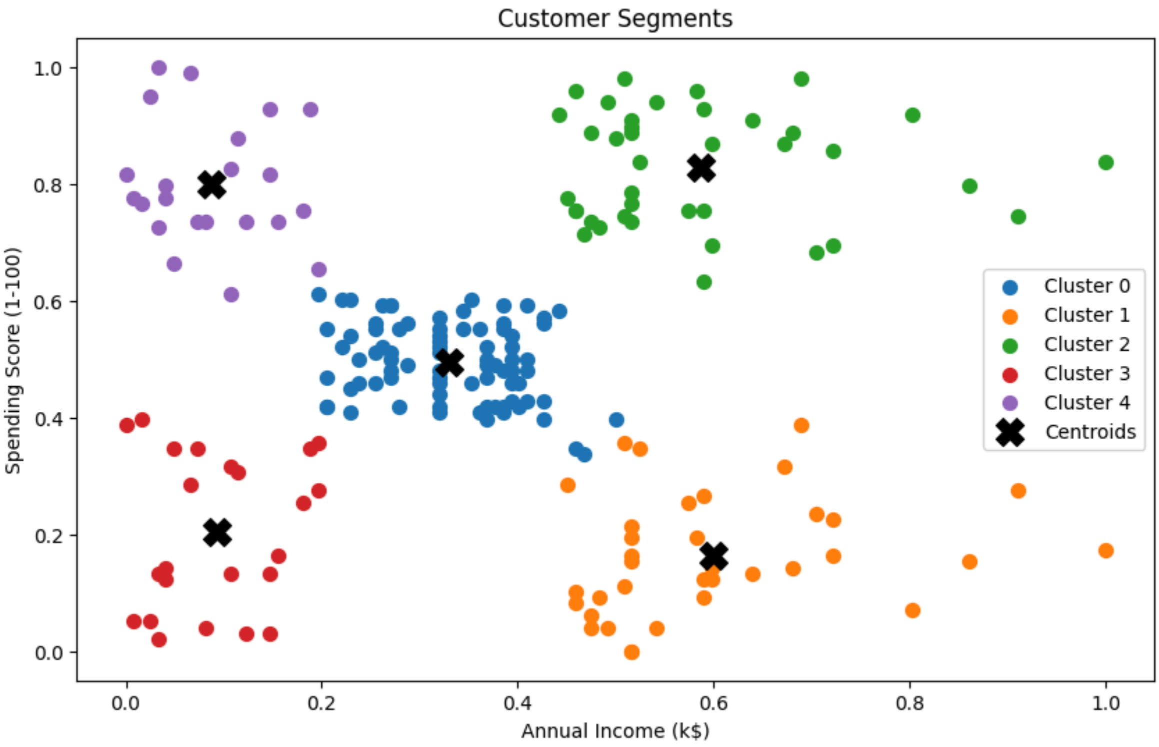
K-means clustering

In [16]:

```
 # Apply K-means clustering with the optimal number of clusters
kmeans = KMeans(n_clusters=5, init='k-means++', max_iter=300, n_init=10, random_
y_kmeans = kmeans.fit_predict(X)
```

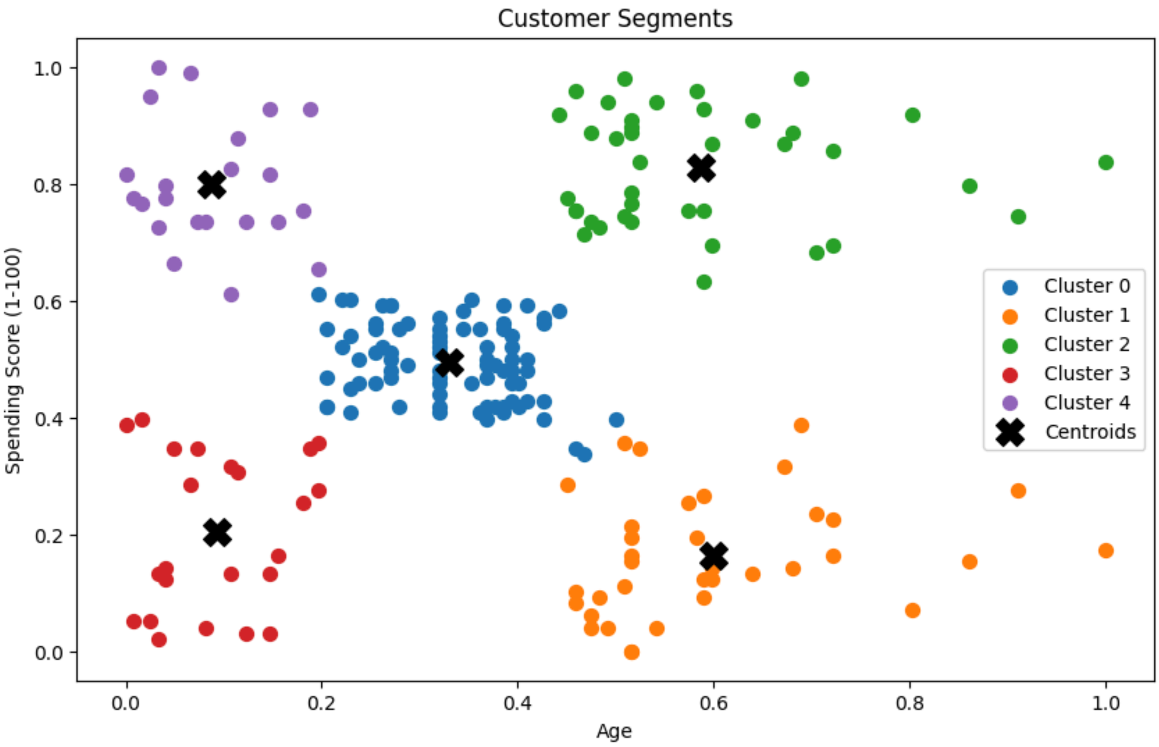
In [17]:

```
# Add the cluster labels to the original data
df_scaled['Cluster'] = y_kmeans
 # Visualize the clusters(Annual Income (k$) & Spending Score (1-100))
plt.figure(figsize=(10, 6))
for i in range(5):
    plt.scatter(df_scaled[df_scaled['Cluster'] == i]['Annual Income (k$)'],
                df_scaled[df_scaled['Cluster'] == i]['Spending Score (1-100)'],
                s=50, label=f'Cluster {i}')

plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s=200,
plt.title('Customer Segments')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
```



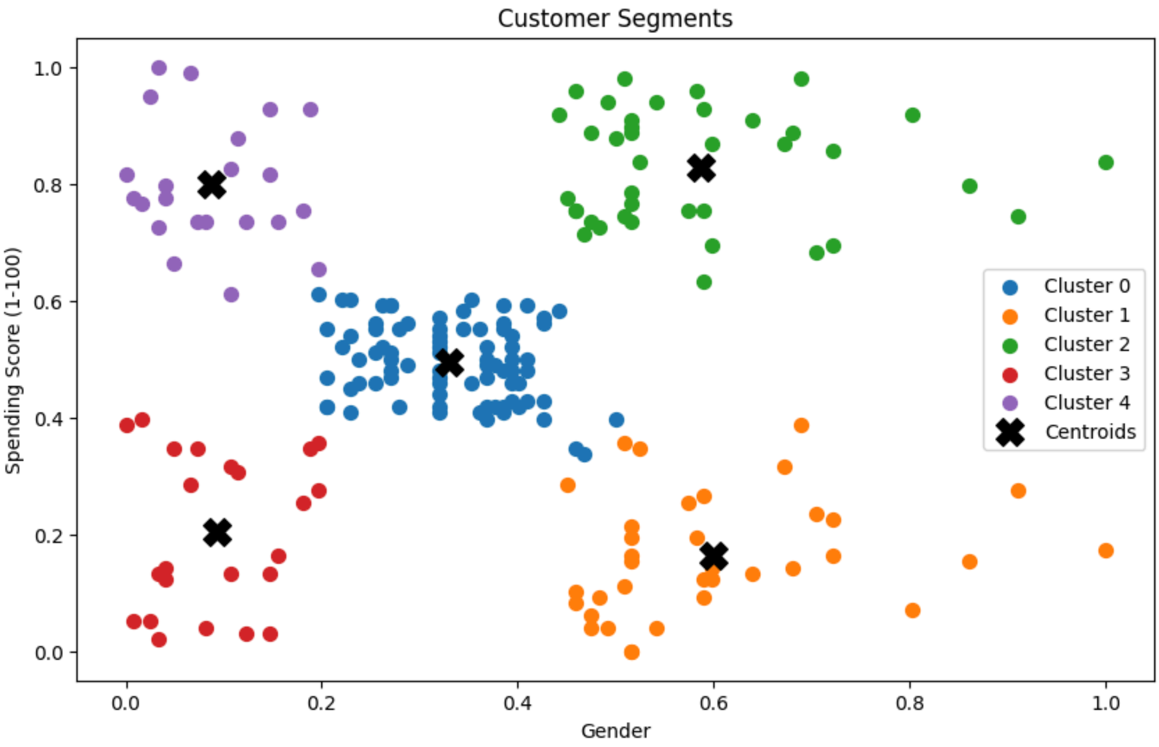
In [18]:

```
# Add the cluster labels to the original data
df_scaled['Cluster'] = y_kmeans
 # Visualize the clusters(Annual Income (k$) & Spending Score (1-100))
plt.figure(figsize=(10, 6))
for i in range(5):
    plt.scatter(df_scaled[df_scaled['Cluster'] == i]['Annual Income (k$)'],
                df_scaled[df_scaled['Cluster'] == i]['Spending Score (1-100)'],
                s=50, label=f'Cluster {i}')

plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s=200,
plt.title('Customer Segments')
plt.xlabel('Age')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
```



In [19]:

```python
# Add the cluster labels to the original data
df_scaled['Cluster'] = y_kmeans
 # Visualize the clusters(Annual Income (k$) & Spending Score (1-100))
plt.figure(figsize=(10, 6))
for i in range(5):
    plt.scatter(df_scaled[df_scaled['Cluster'] == i]['Annual Income (k$)'],
                df_scaled[df_scaled['Cluster'] == i]['Spending Score (1-100)'],
                s=50, label=f'Cluster {i}')

plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s=200,
plt.title('Customer Segments')
plt.xlabel('Gender')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
```


Customer Segments
Spending Score (1-100)
1.0
0.8
0.6
0.4
0.2
0.0
Gender
0.0
0.2
0.4
0.6
0.8
1.0
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Centroids

```
In [ ]:
```