In [68]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [69]:
```
data = pd.read_csv(r"C:\Users\DD\Desktop\ML PROJECTS\Unsupervised Learning\Mall_Customers.csv")
```

In [70]:
```
data
```

Out[70]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19.0	15	39.0
1	2	Male	21.0	15	81.0
2	3	Female	20.0	16	6.0
3	4	Female	23.0	16	77.0
4	5	Female	31.0	17	40.0
...	...	...	...	...	...
195	196	Female	NaN	120	79.0
196	197	Female	NaN	126	28.0
197	198	Male	NaN	126	74.0
198	199	Male	NaN	137	18.0
199	200	Male	30.0	137	83.0

200 rows × 5 columns

In [71]:
```
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                     190 non-null    float64
 3   Annual Income (k$)      200 non-null    int64  
 4   Spending Score (1-100)  187 non-null    float64
dtypes: float64(2), int64(2), object(1)
memory usage: 7.9+ KB
```

In [72]:
```
data.describe()
```

Out[72]:

	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)
count	200.000000	190.000000	200.000000	187.000000
mean	100.500000	39.047368	60.560000	50.379679
std	57.879185	14.240670	26.264721	25.267392
min	1.000000	18.000000	15.000000	1.000000
25%	50.750000	28.000000	41.500000	35.000000
50%	100.500000	36.000000	61.500000	50.000000
75%	150.250000	49.000000	78.000000	71.500000
max	200.000000	70.000000	137.000000	99.000000

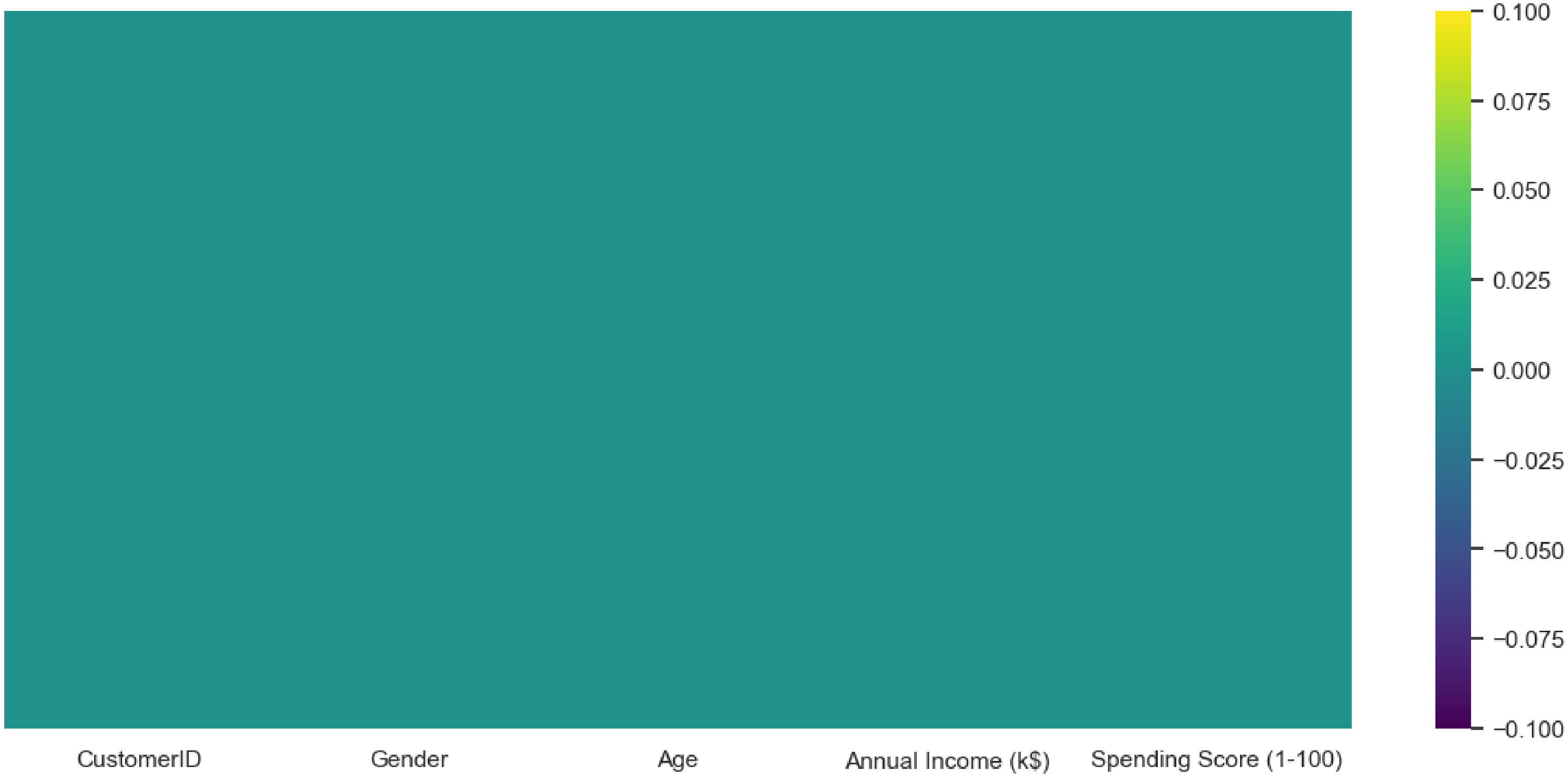
In [73]:
```
data.isnull().sum()
```

Out[73]:
```
CustomerID                 0
Gender                     0
Age                       10
Annual Income (k$)         0
Spending Score (1-100)    13
dtype: int64
```

In [74]:
```
data['Age'].fillna(data['Age'].median(),inplace=True)
data['Spending Score (1-100)'].fillna(data['Spending Score (1-100)'].median(),inplace=True)
```

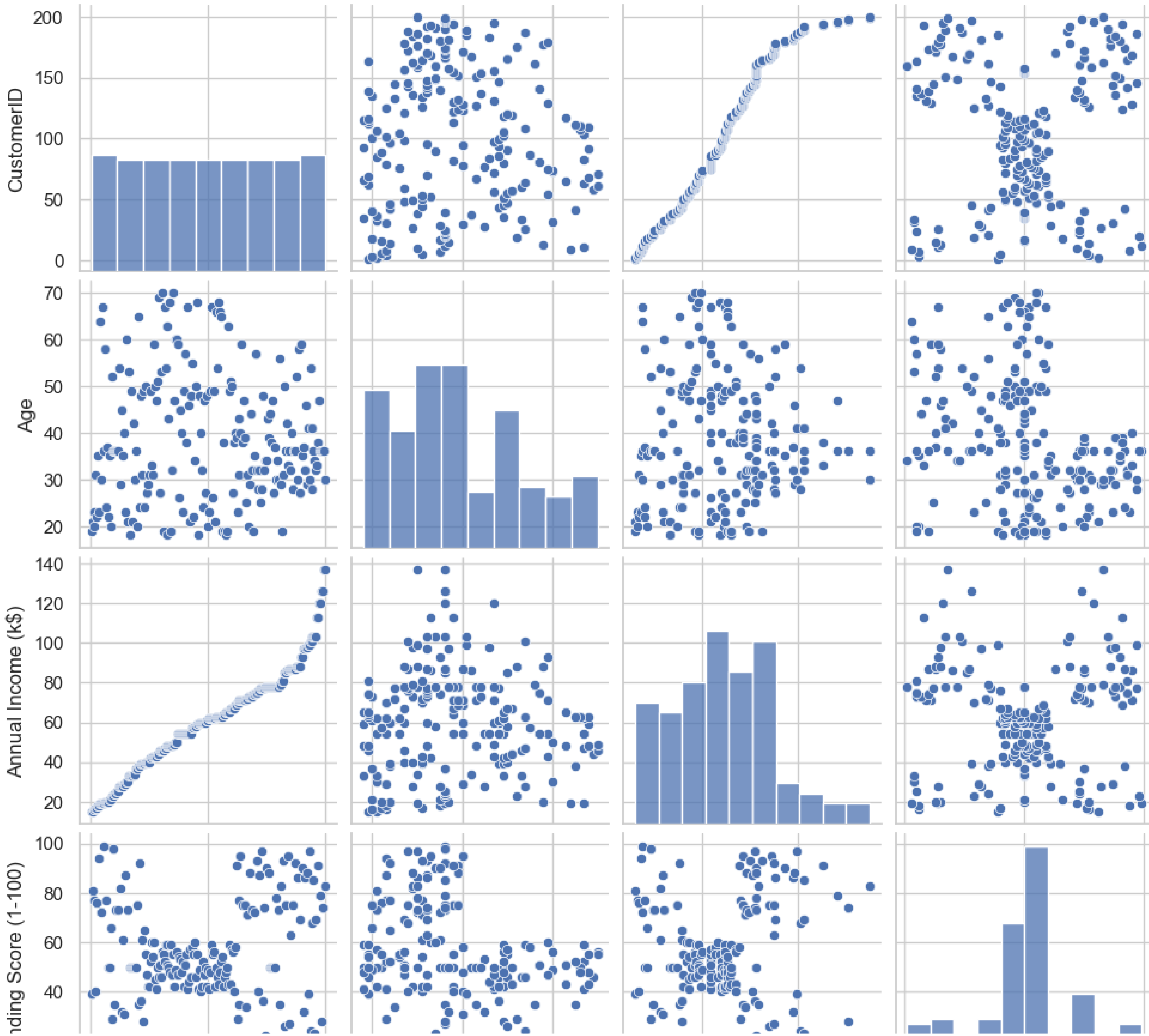
In [75]:
```
sns.heatmap(data.isnull(),yticklabels=False,cmap="viridis")
```

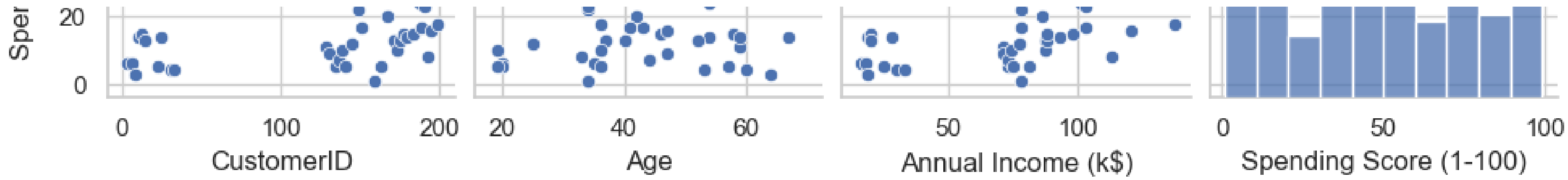
Out[75]: <Axes: >



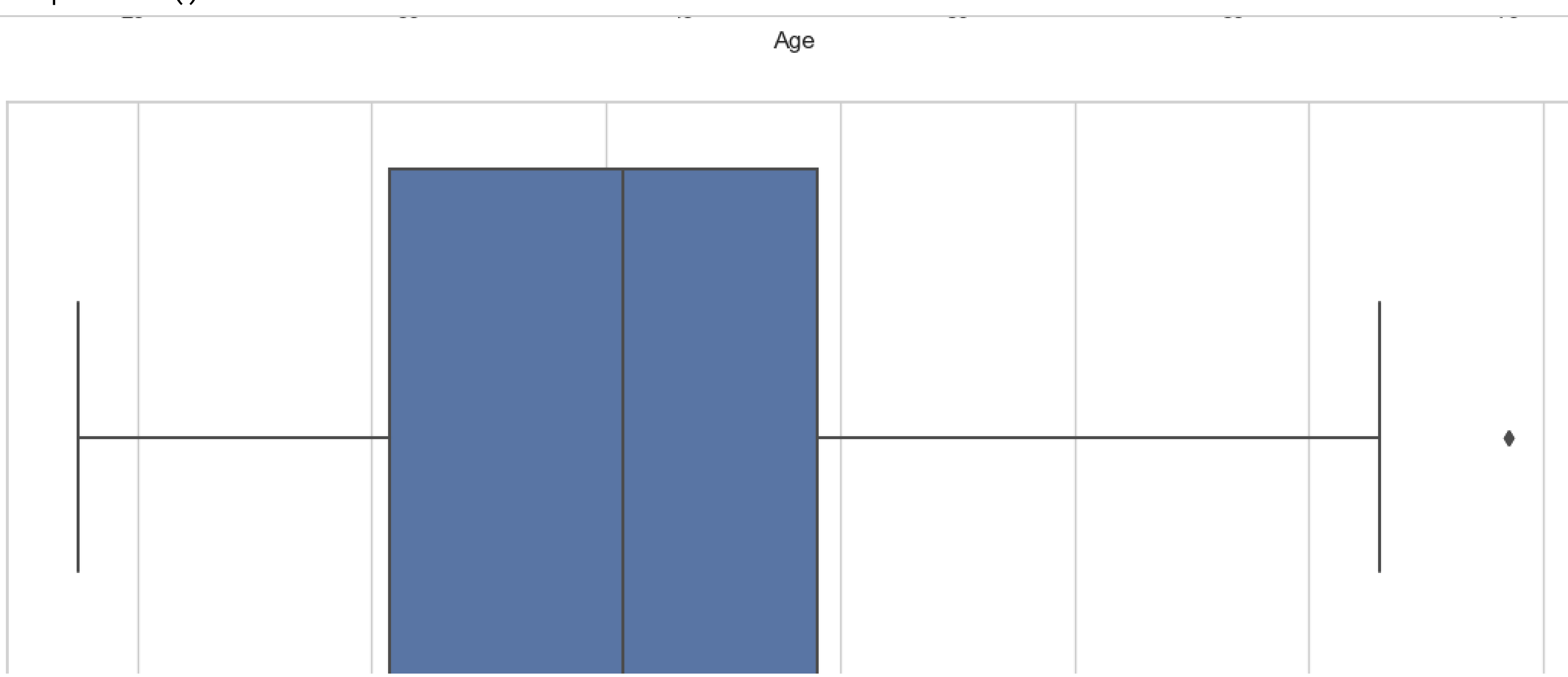
In [76]:
```
sns.pairplot(data)
```

Out[76]: <seaborn.axisgrid.PairGrid at 0x19b2c777690>


CustomerID
Age
Annual Income (k$)
nding Score (1-100)
200
150
100
50
0
70
60
50
40
30
20
140
120
100
80
60
40
20
100
80
60
40

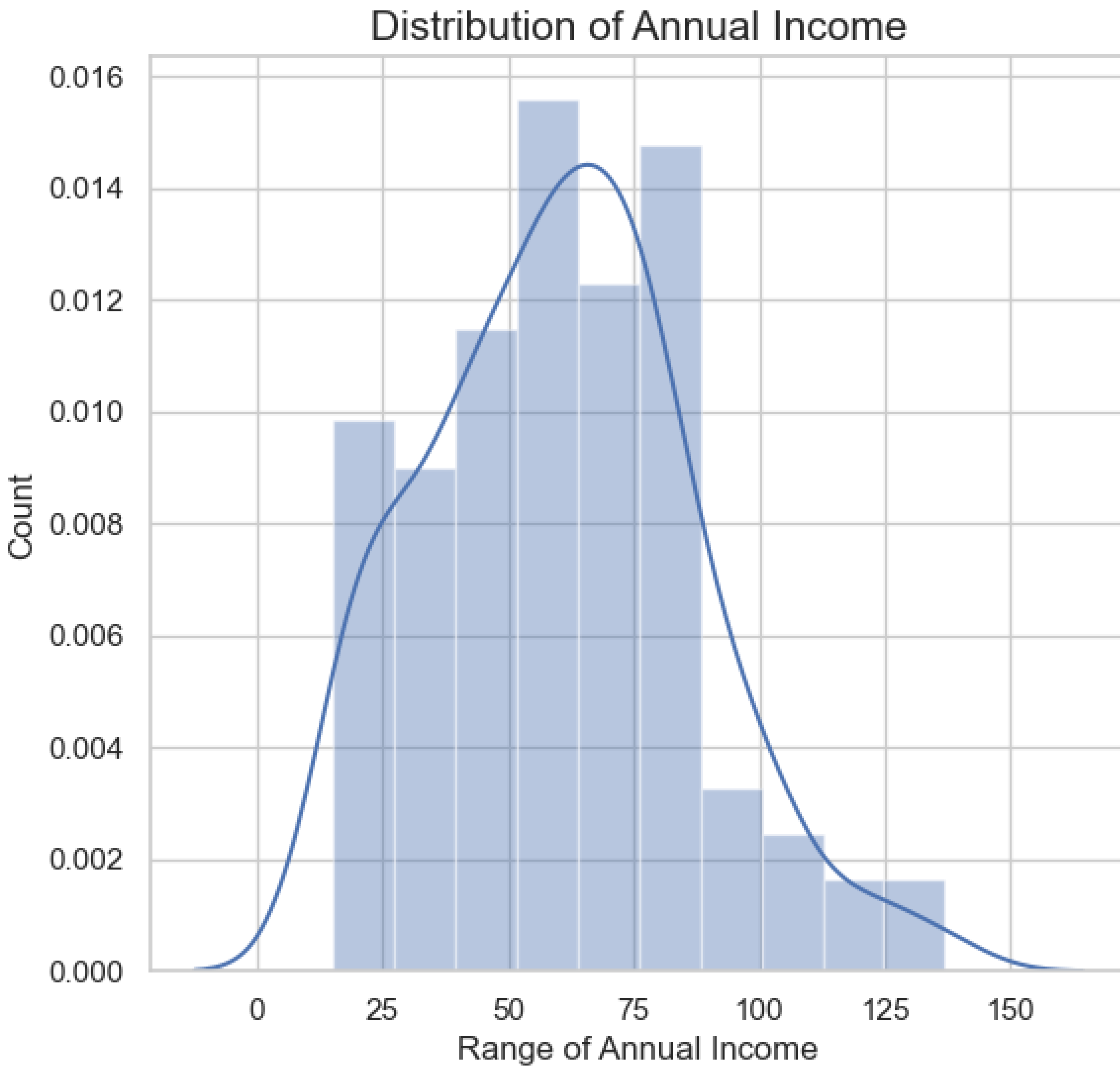


In [77]:
```
numeric_columns = data.select_dtypes(include=np.number).columns.tolist()

for i in numeric_columns:
    sns.boxplot(x=data[i])
    plt.show()
```

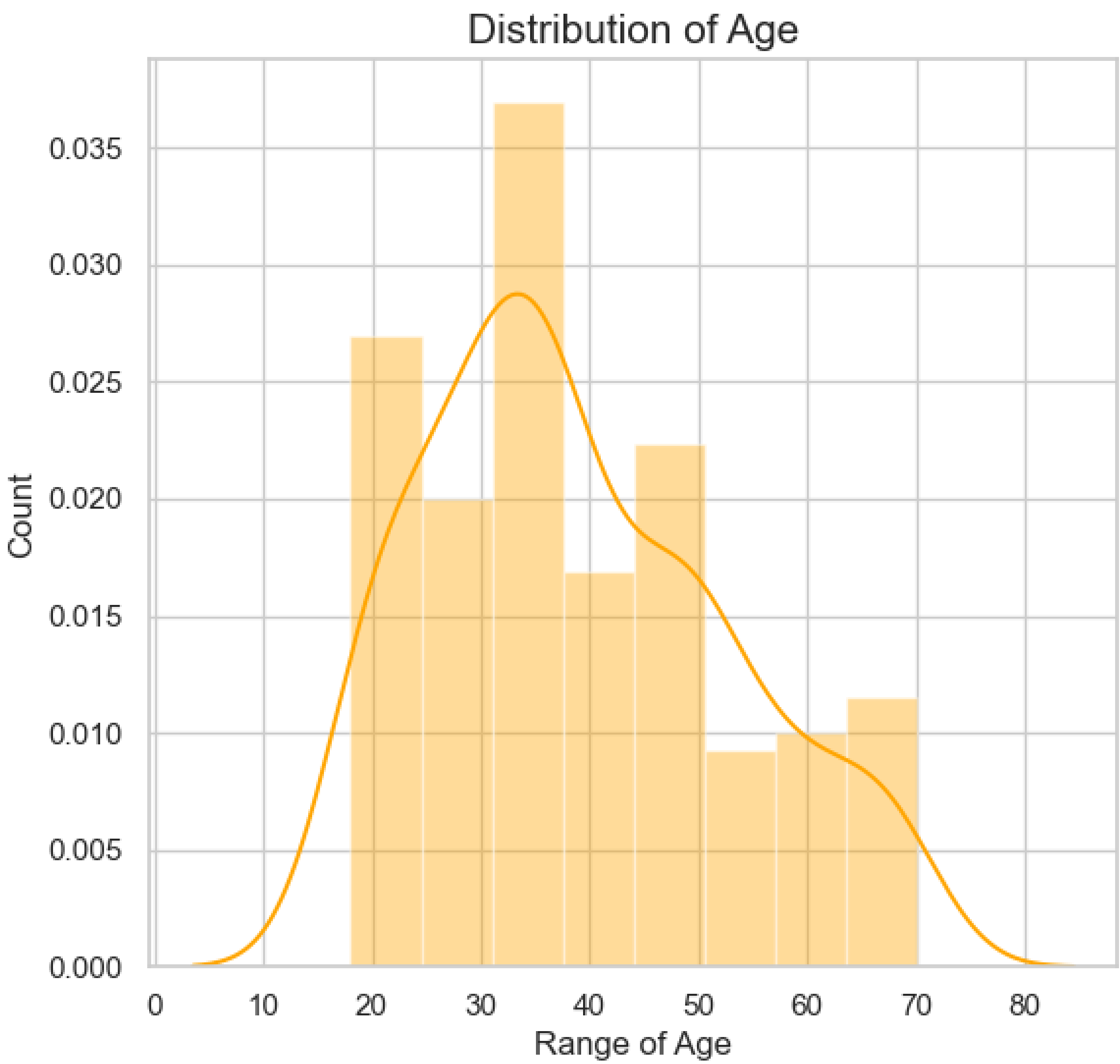


In [78]:
```
import warnings
warnings.filterwarnings('ignore')

plt.rcParams['figure.figsize'] = (14, 6)

plt.subplot(1, 2, 1)
sns.set(style = 'whitegrid')
sns.distplot(data['Annual Income (k$)'])
plt.title('Distribution of Annual Income', fontsize = 15)
plt.xlabel('Range of Annual Income')
plt.ylabel('Count')
```

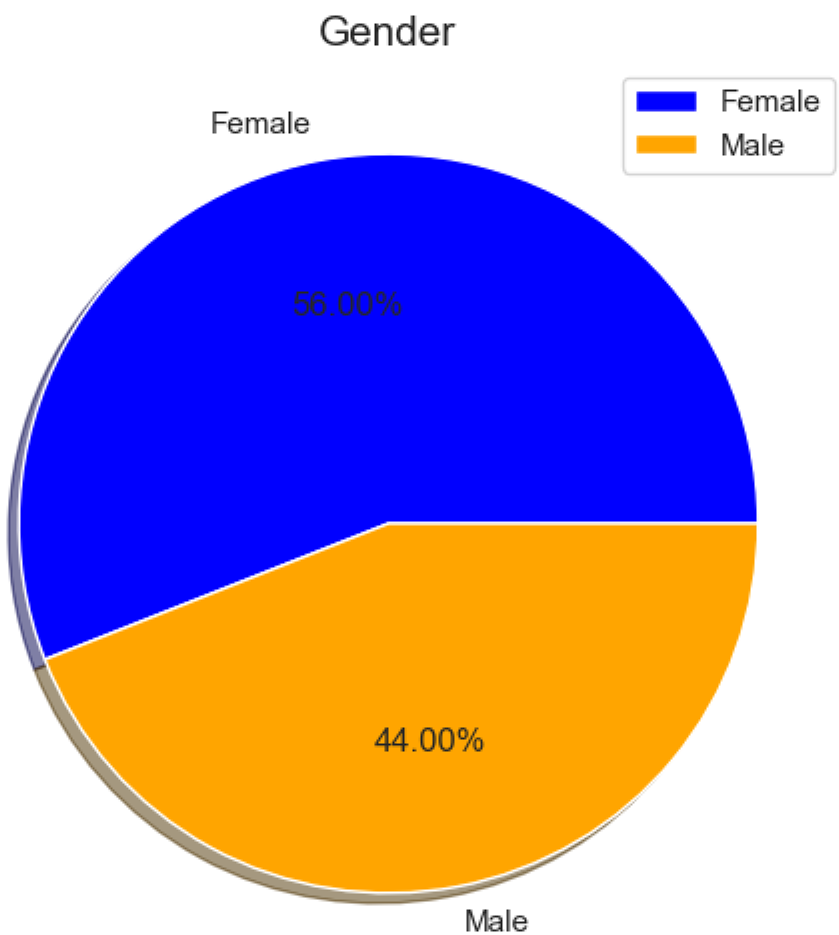
Out[78]:
```
Text(0, 0.5, 'Count')
```


Distribution of Annual Income
Count
0.000
0.002
0.004
0.006
0.008
0.010
0.012
0.014
0.016
0
25
50
75
100
125
150
Range of Annual Income

In [79]:
```
plt.subplot(1, 2, 2)
sns.set(style = 'whitegrid')
sns.distplot(data['Age'], color = 'orange')
plt.title('Distribution of Age', fontsize = 15)
plt.xlabel('Range of Age')
plt.ylabel('Count')
plt.show()
```

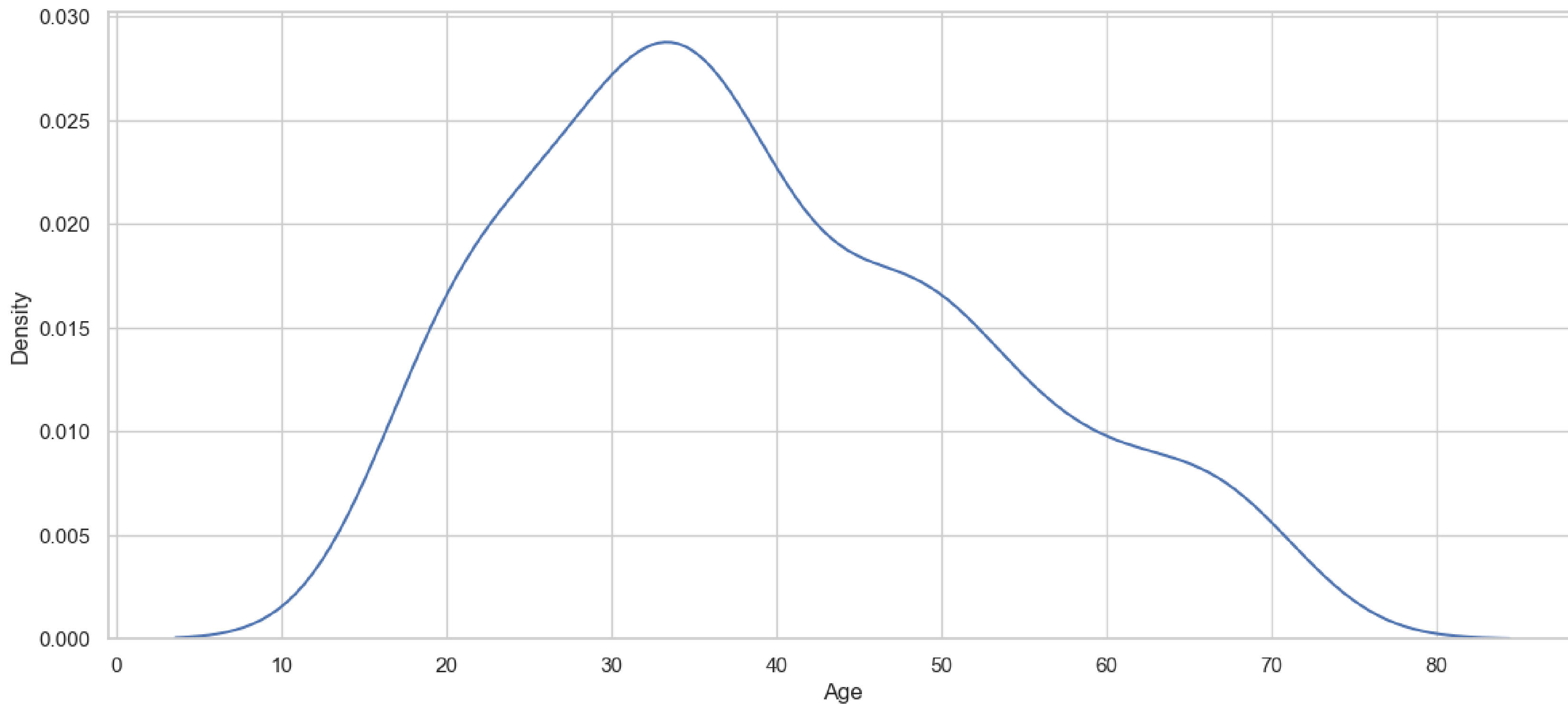


In [80]:
```
labels = ['Female', 'Male']
size = data['Gender'].value_counts()
colors = ['blue', 'orange']



plt.pie(size, colors = colors, labels = labels, shadow = True, autopct = '%.2f%%')
plt.title('Gender', fontsize = 15)
plt.axis('off')
plt.legend()
plt.show()
```

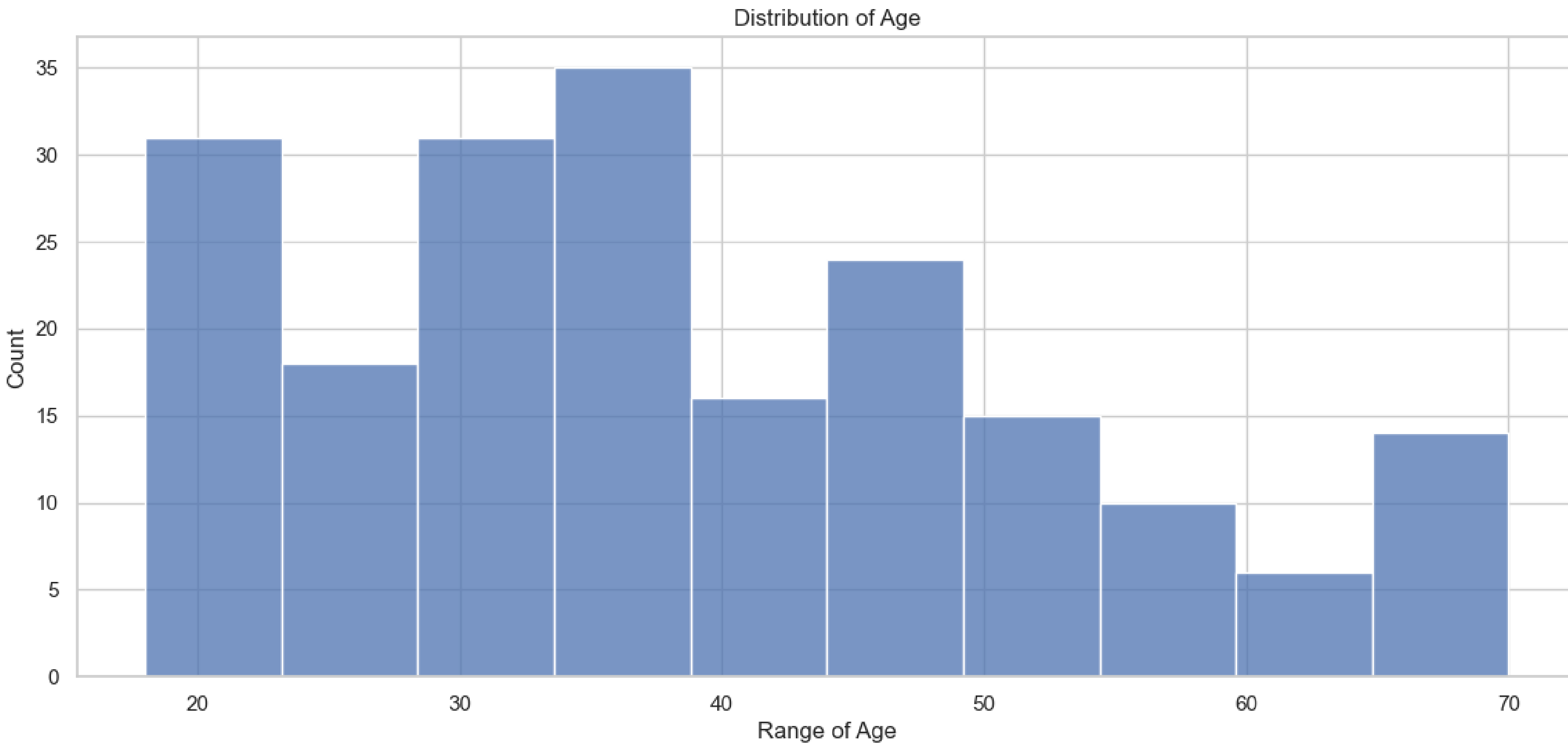


In [81]:
```
sns.kdeplot(x='Age',data=data)
```

Out[81]: <Axes: xlabel='Age', ylabel='Density'>

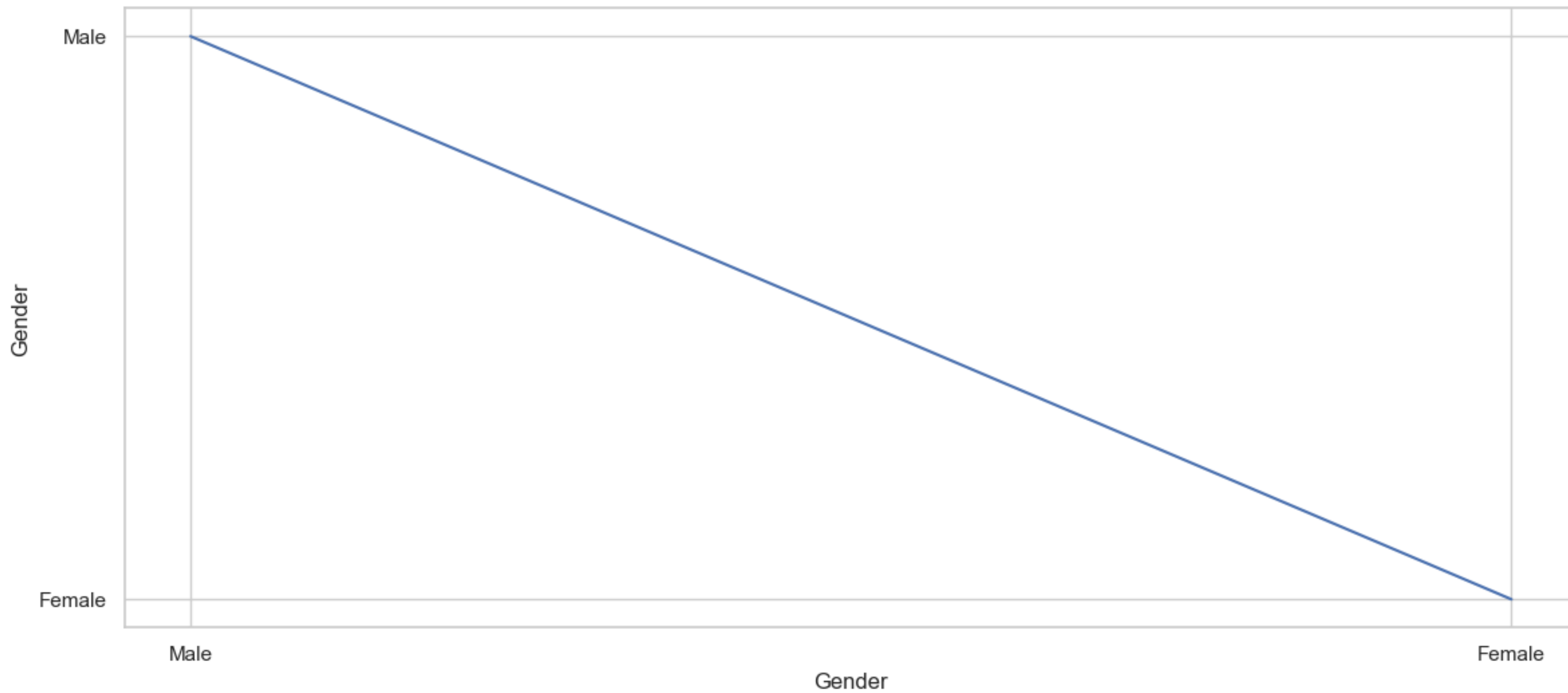


In [82]:
```
sns.histplot(data['Age'])
plt.title('Distribution of Age')
plt.xlabel('Range of Age')
plt.ylabel('Count')
plt.show()
```

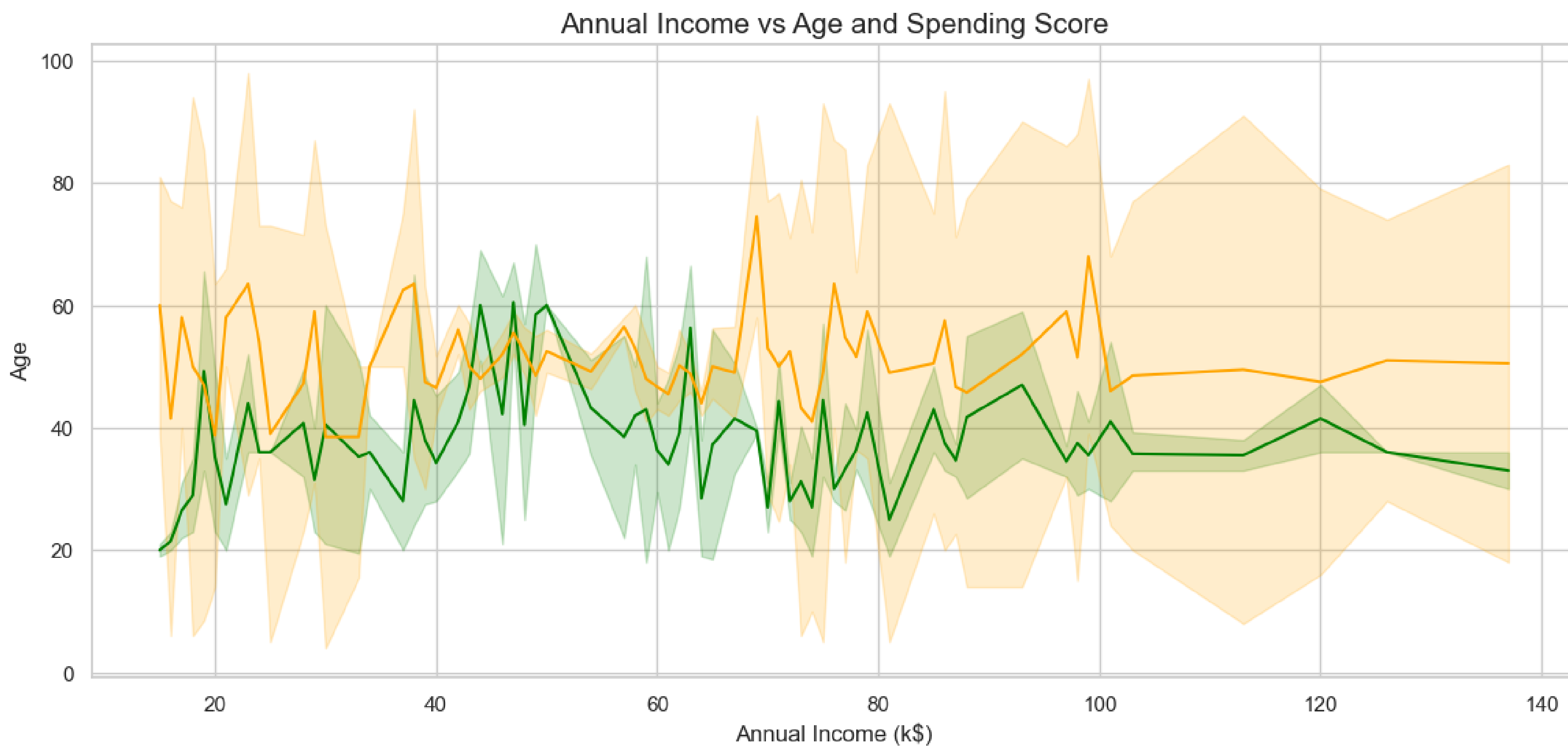


In [83]:
```
sns.lineplot(x='Gender',y='Gender',data=data)
```

Out[83]: <Axes: xlabel='Gender', ylabel='Gender'>

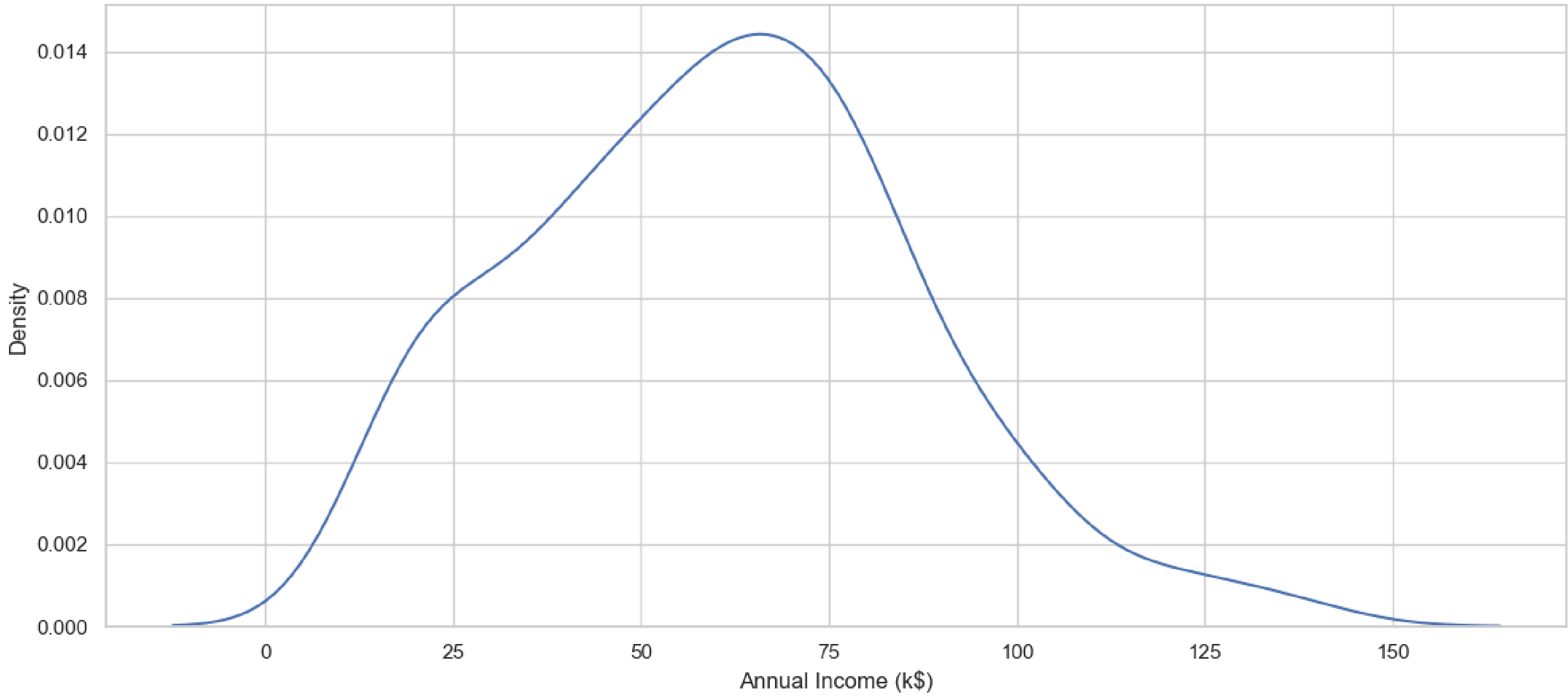


In [84]:
```
x = data['Annual Income (k$)']
y = data['Age']
z = data['Spending Score (1-100)']

sns.lineplot(x=x, y=y, color='green')
sns.lineplot(x=x, y=z, color='orange')
plt.title('Annual Income vs Age and Spending Score', fontsize=15)
plt.show()
```



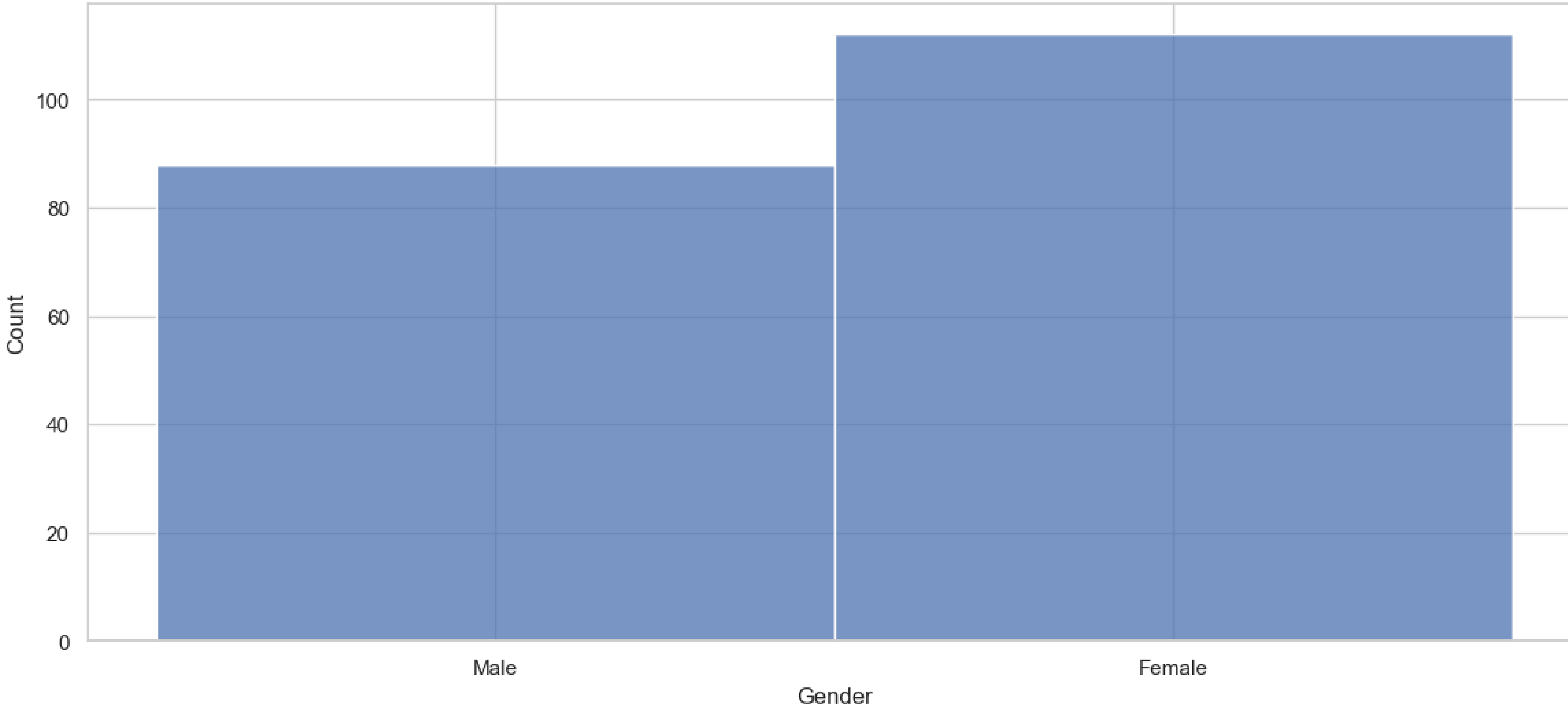
In [85]:
```
sns.kdeplot(x='Annual Income (k$)',data=data)
```

Out[85]:
```
<Axes: xlabel='Annual Income (k$)', ylabel='Density'>
```



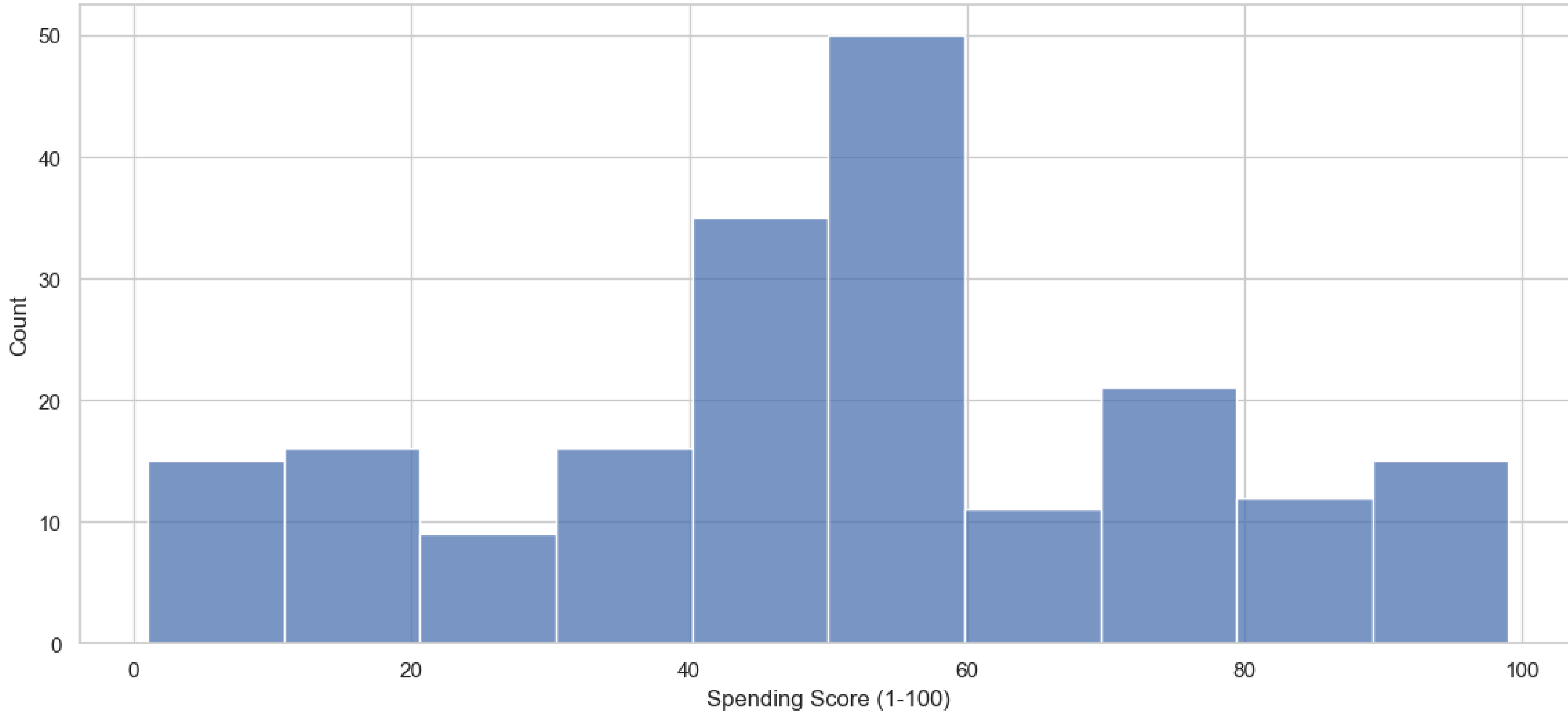
In [86]:
```
sns.histplot(x='Gender',data=data)
```

Out[86]: <Axes: xlabel='Gender', ylabel='Count'>



In [87]:
```
sns.histplot(x='Spending Score (1-100)',data=data)
```

Out[87]:
```
<Axes: xlabel='Spending Score (1-100)', ylabel='Count'>
```



In [88]:
```
sns.stripplot(x=data['Gender'], y=data['Age'], palette='Purples')
plt.title('Gender vs Age')
plt.show()
```

In [89]:
```
sns.violinplot(x=data['Gender'], y=data['Annual Income (k$)'], palette='rainbow')
plt.title('Gender vs Annual Income')
plt.show()
```

In [90]:
```
from scipy.stats import zscore
z_scores = zscore(data['Annual Income (k$)'])
z_score_outliers=(z_scores<-3)|(z_scores>3)
z_score_outlier_rows=data[z_score_outliers]
print("outliers detected by Z-score:",z_score_outlier_rows)
```

```
outliers detected by Z-score: Empty DataFrame
Columns: [CustomerID, Gender, Age, Annual Income (k$), Spending Score (1-100)]
Index: []
```

In [91]:
```
x=(z_scores>-3)&(z_scores<3)
df=data[x]
df
```

Out[91]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19.0	15	39.0
1	2	Male	21.0	15	81.0
2	3	Female	20.0	16	6.0
3	4	Female	23.0	16	77.0
4	5	Female	31.0	17	40.0
...	...	...	...	...	...
195	196	Female	36.0	120	79.0
196	197	Female	36.0	126	28.0
197	198	Male	36.0	126	74.0
198	199	Male	36.0	137	18.0
199	200	Male	30.0	137	83.0

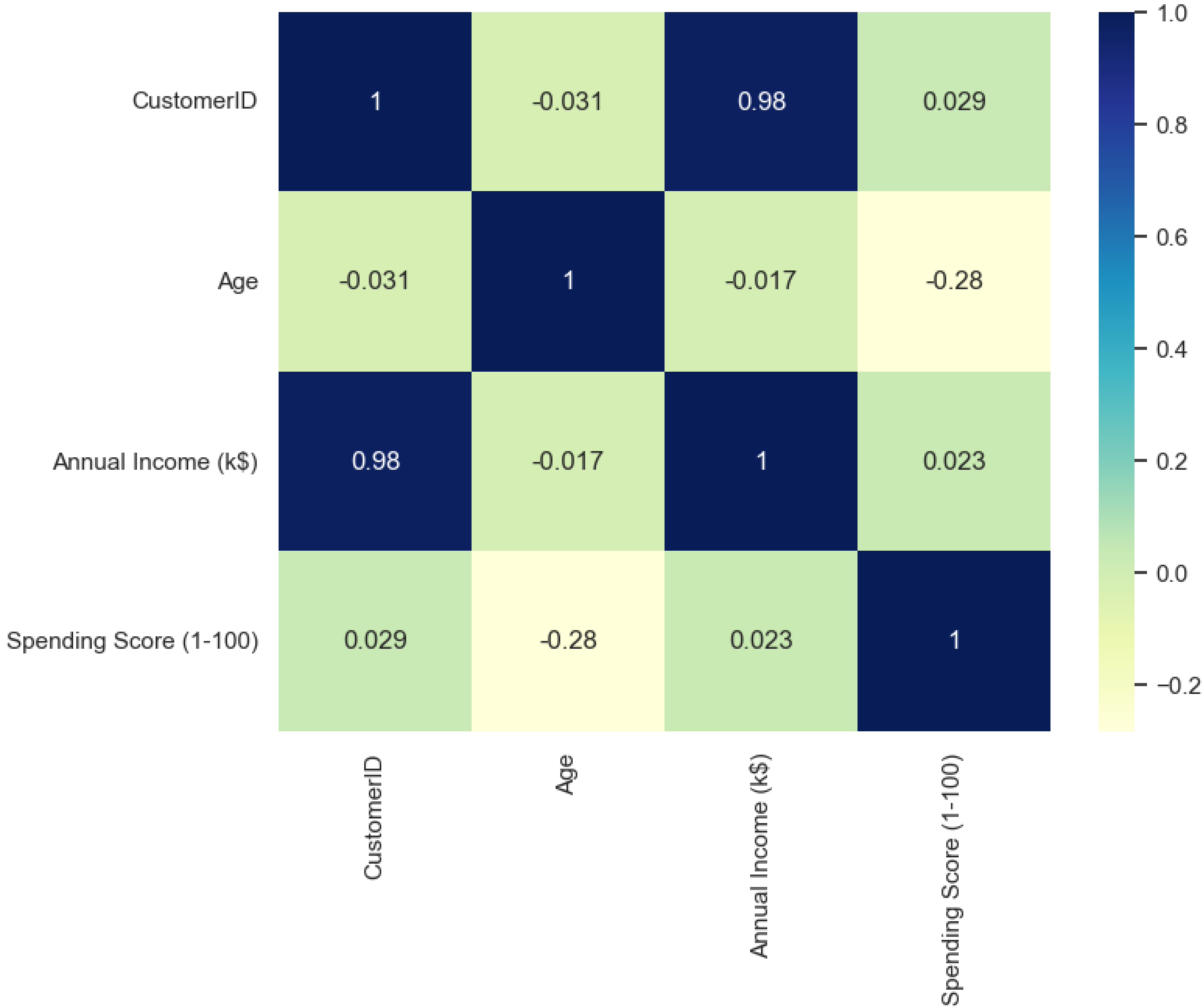
200 rows × 5 columns

K-Means Clustering

feature scaling

In [92]:
```
#set the 'Gender' column as the index of the DataFrame 'data'
data.set_index('Gender',inplace = True)
```

In [93]:
```
#Heatmap of the data
plt.figure(figsize = (8,6))
sns.heatmap(data.corr(),annot = True,cmap="YlGnBu")
plt.show()
```


CustomerID
Age
Annual Income (k$)
Spending Score (1-100)
1
-0.031
0.98
0.029
-0.031
1
-0.017
-0.28
0.98
-0.017
1
0.023
0.029
-0.28
0.023
1
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

In [94]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
```

In [95]:
```
#instantiate and fit 'StandardScaler' function
gender_scaler = scaler.fit_transform(data)
```

In [96]:
```
#new dataframe of the scaled features
gender_scaler = pd.DataFrame(gender_scaler)
gender_scaler.columns = ['CustomerID', 'Age', 'Annual Income (k$)', 'Spending Score (1-100)']

#to display top five rows
gender_scaler.head()
```

Out[96]:

	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)
0	-1.723412	-1.435484	-1.738999	-0.465996
1	-1.706091	-1.291178	-1.738999	1.257635
2	-1.688771	-1.363331	-1.700830	-1.820277
3	-1.671450	-1.146872	-1.700830	1.093479
4	-1.654129	-0.569648	-1.662660	-0.424957

In [97]:
```
#Importing libraries required for KMeans
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
```

In [98]:
```
#create a list for different values of k
n_clusters = [4, 5, 6, 7, 8]

for K in n_clusters :
    cluster = KMeans(n_clusters=K, random_state=10)
    predict = cluster.fit_predict(gender_scaler)

    score = silhouette_score(gender_scaler, predict, random_state=10)
    print("For n_clusters = {}, silhouette score is {})".format(K, score))
```

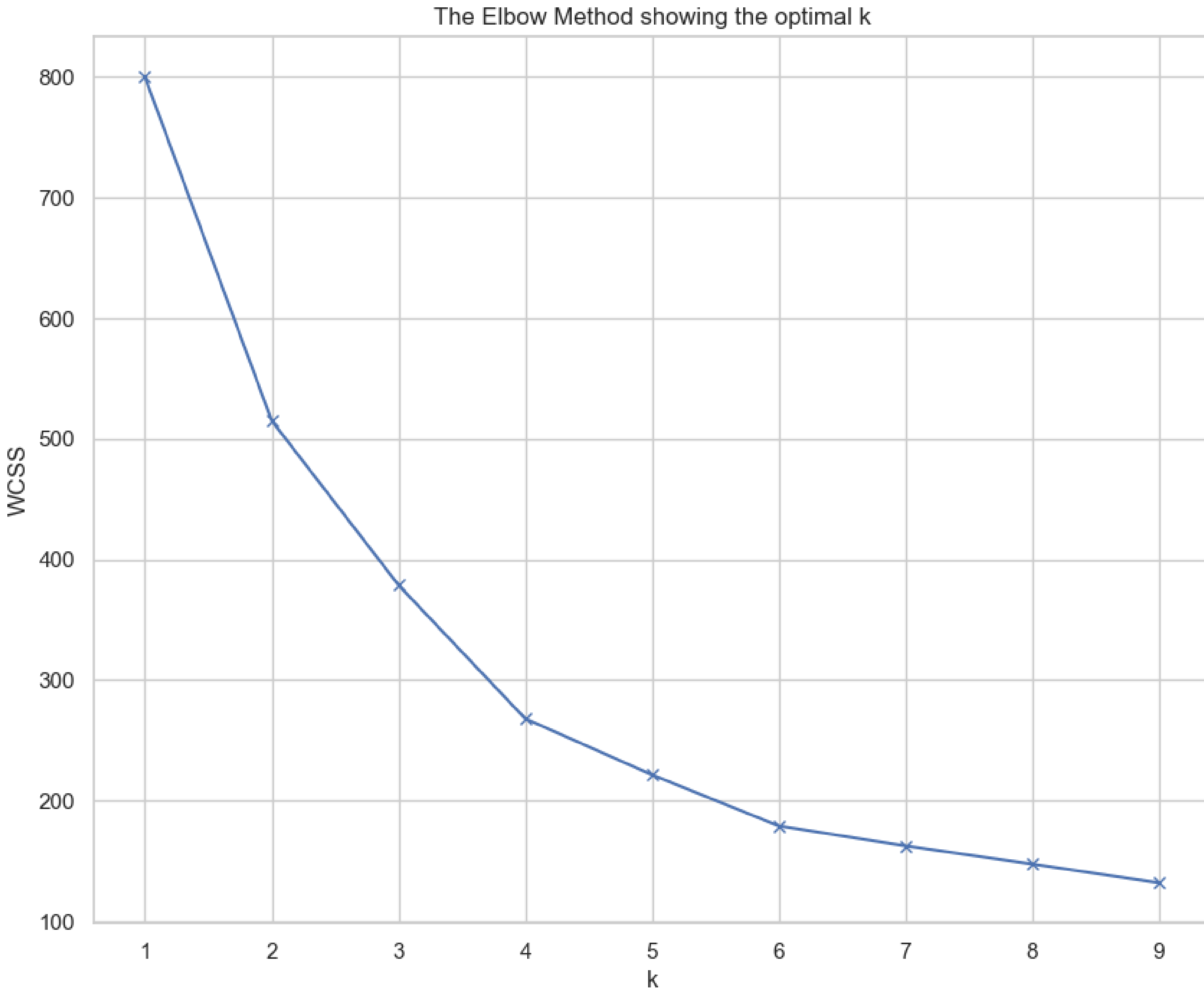
```
For n_clusters = 4, silhouette score is 0.3940736473040689)
For n_clusters = 5, silhouette score is 0.3783523335126997)
For n_clusters = 6, silhouette score is 0.396794439779569)
For n_clusters = 7, silhouette score is 0.3731111105185087)
For n_clusters = 8, silhouette score is 0.3628265445047377)
```

In [99]:
```
wcss = []                           #Create an empty list to store the WCSS values
K = range(1,10)                     #Set a range of values for the number of clusters (K)

# Loop through each value of K
for k in K:
    kmeanModel = KMeans(n_clusters=k)
    kmeanModel.fit(gender_scaler)
    wcss.append(kmeanModel.inertia_)
```

In [100]:
```
wcss
```

Out[100]:
```
[799.9999999999998,
 515.1159818402758,
 378.86248891036837,
 268.1336994823401,
 221.5841139517122,
 179.1720378318771,
 162.6897410058919,
 147.53858594961417,
 132.10461324044832]
```

In [101]:
```
#Elbow Method
plt.figure(figsize=(10,8))

# Plotting the WCSS values against the number of clusters (K)
plt.plot(K, wcss, 'bx-')                #'bx-' specifies blue color marker type xand line style '-'
plt.xlabel('k')
plt.ylabel('WCSS')
plt.title('The Elbow Method showing the optimal k')
plt.show()
```


The Elbow Method showing the optimal k
WCSS
800
700
600
500
400
300
200
100
1
2
3
4
5
6
7
8
9
k

K-Means Clustering with K= 6

In [102]:
```
#building a K-Means model for k=6
model = KMeans(n_clusters=6, random_state=10)

#fit the model
model.fit(gender_scaler)
```

Out[102]:
```
KMeans(n_clusters=6, random_state=10)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [103]:
```
data_output = data.copy(deep = True)
#add a column 'cluster' in the data giving cluster number corresponding to each observation
data_output['Cluster'] = model.labels_

#head to display top 5 rows
data_output.head()
```

Out[103]:

	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)	Cluster
Gender					
Male	1	19.0	15	39.0	4
Male	2	21.0	15	81.0	4
Female	3	20.0	16	6.0	1
Female	4	23.0	16	77.0	4
Female	5	31.0	17	40.0	4

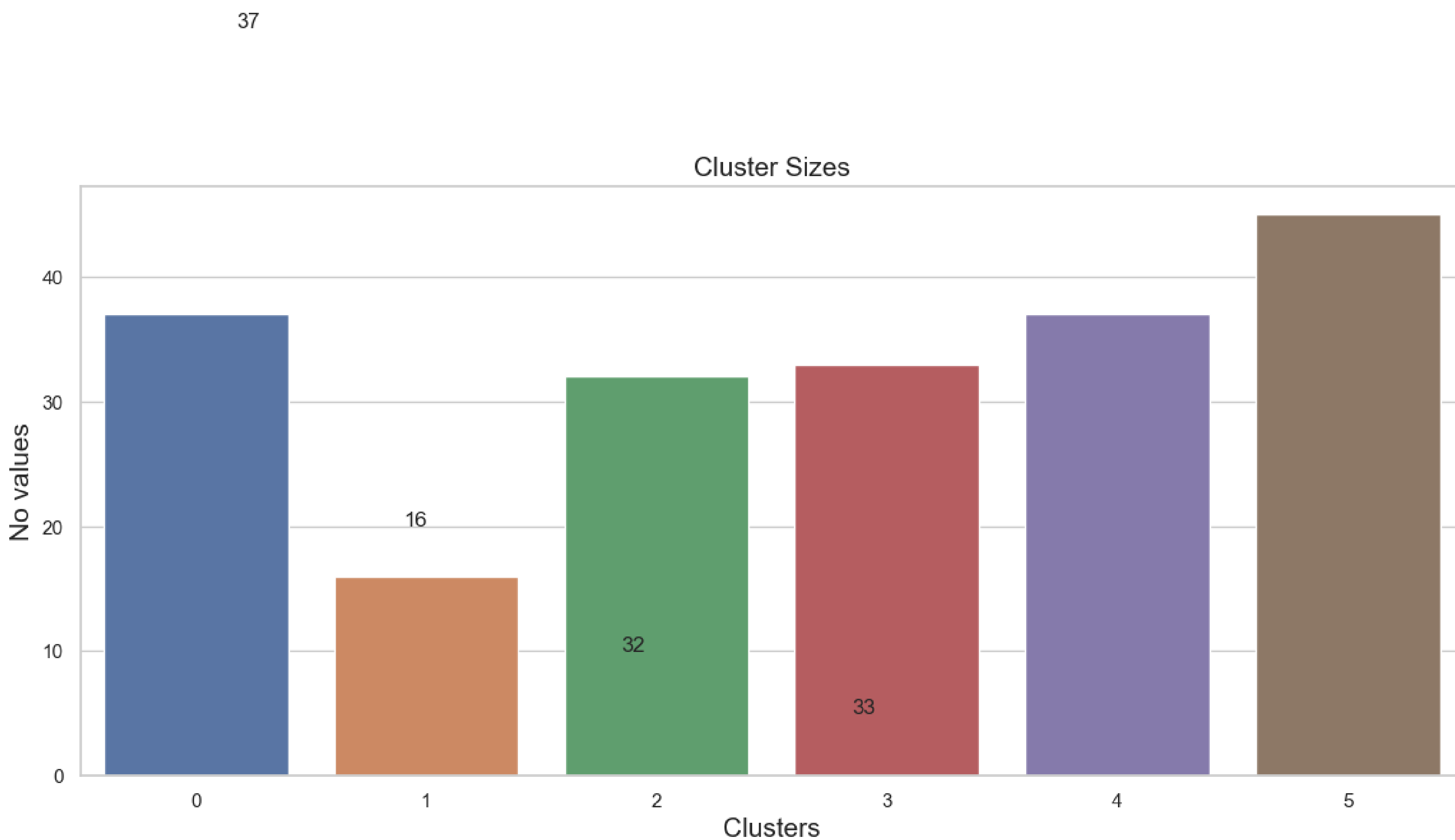
In [104]:
```
#Shape of data_output
data_output.shape
```

Out[104]:
```
(200, 5)
```

In [105]:
```
#'return_counts = True' gives the number observation in each cluster
np.unique(model.labels_, return_counts=True)
```

Out[105]:
```
(array([0, 1, 2, 3, 4, 5]), array([37, 16, 32, 33, 37, 45], dtype=int64))
```

In [106]:
```
sns.countplot(data=data_output, x='Cluster')
plt.title('Cluster Sizes', fontsize=15)
plt.xlabel('Clusters', fontsize=15)
plt.ylabel('No values', fontsize=15)
plt.text(x=0.18, y=60, s=np.unique(model.labels_, return_counts=True)[1][0])
plt.text(x=0.9, y=20, s=np.unique(model.labels_, return_counts=True)[1][1])
plt.text(x=1.85, y=10, s=np.unique(model.labels_, return_counts=True)[1][2])
plt.text(x=2.85, y=5, s=np.unique(model.labels_, return_counts=True)[1][3])

plt.show()
```



In [107]:
```
from sklearn.metrics import silhouette_score, calinski_harabasz_score,davies_bouldin_score
silhouette_avg = silhouette_score(gender_scaler, predict)
print(f"Silhouette Score: {silhouette_avg}")
```

```
Silhouette Score: 0.3628265445047377
```

In [108]:
```
calinski_harabasz_index = calinski_harabasz_score(gender_scaler, predict)
print(f"Calinski-Harabasz Index: {calinski_harabasz_index}")
```

```
Calinski-Harabasz Index: 120.90745409717387
```

In [109]:
```
davies_bouldin_index = davies_bouldin_score(gender_scaler, predict)
print(f"Davies-Bouldin Index: {davies_bouldin_index}")
```

```
Davies-Bouldin Index: 1.0158156985086655
```

Hierarchical clustering

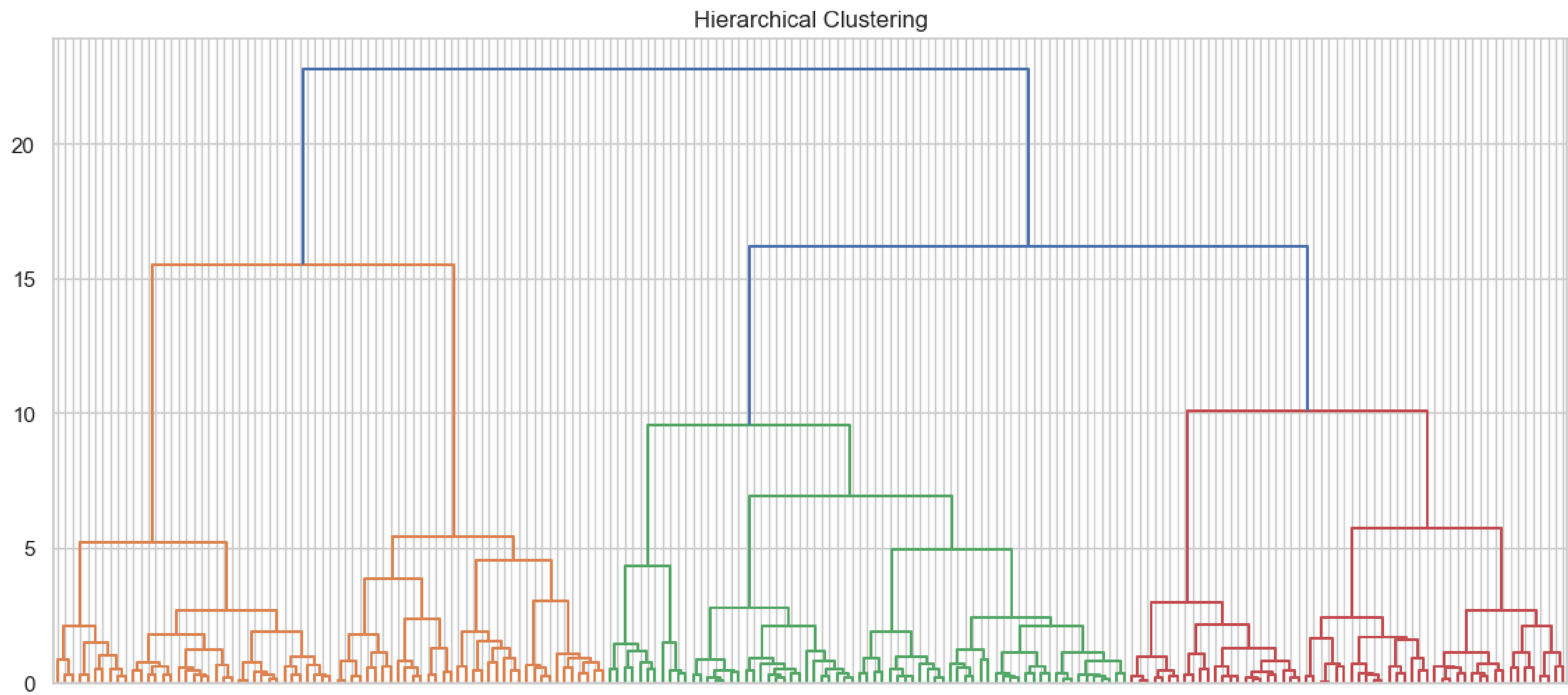
In [110]:
```
import scipy.cluster.hierarchy as sch
from sklearn.preprocessing import scale as s
from scipy.cluster.hierarchy import dendrogram, linkage
```

In [111]:
```python
Z = sch.linkage(gender_scaler,method='ward')
Z
```

Out[111]:
```
array([[2.10000000e+01, 2.30000000e+01, 5.15454685e-02, 2.00000000e+00],
       [1.29000000e+02, 1.31000000e+02, 8.00380307e-02, 2.00000000e+00],
       [6.50000000e+01, 6.80000000e+01, 8.89163869e-02, 2.00000000e+00],
       [1.53000000e+02, 1.56000000e+02, 8.89163869e-02, 2.00000000e+00],
       [1.06000000e+02, 1.09000000e+02, 9.71432333e-02, 2.00000000e+00],
       [3.00000000e+00, 5.00000000e+00, 9.77096554e-02, 2.00000000e+00],
       [1.14000000e+02, 1.15000000e+02, 1.10647206e-01, 2.00000000e+00],
       [9.20000000e+01, 9.60000000e+01, 1.29394423e-01, 2.00000000e+00],
       [1.33000000e+02, 1.37000000e+02, 1.34906716e-01, 2.00000000e+00],
       [1.17000000e+02, 1.19000000e+02, 1.37733249e-01, 2.00000000e+00],
       [4.80000000e+01, 4.90000000e+01, 1.45341780e-01, 2.00000000e+00],
       [1.13000000e+02, 2.06000000e+02, 1.57442484e-01, 3.00000000e+00],
       [8.40000000e+01, 8.70000000e+01, 1.66598355e-01, 2.00000000e+00],
       [8.30000000e+01, 8.50000000e+01, 1.69590671e-01, 2.00000000e+00],
       [1.10000000e+02, 2.04000000e+02, 1.72196666e-01, 3.00000000e+00],
       [1.00000000e+02, 1.05000000e+02, 1.73229899e-01, 2.00000000e+00],
       [4.60000000e+01, 5.00000000e+01, 1.76043924e-01, 2.00000000e+00],
       [7.20000000e+01, 7.40000000e+01, 1.90927504e-01, 2.00000000e+00],
       [6.20000000e+01, 6.70000000e+01, 2.02756283e-01, 2.00000000e+00],
```

In [112]:
```
den = sch.dendrogram(Z)
plt.tick_params(
    axis='x',
    which='both',
    bottom=False,
    top=False,
    labelbottom=False)
plt.title('Hierarchical Clustering')
```

Out[112]: Text(0.5, 1.0, 'Hierarchical Clustering')



In [113]:
```
from sklearn.cluster import AgglomerativeClustering
```

In [114]:
```
hc_model = AgglomerativeClustering(n_clusters = 2, affinity = 'euclidean', linkage ='ward')
```

In [115]:
```
y_cluster = hc_model.fit_predict(gender_scaler)
```

In [116]:
```
y_cluster
```

Out[116]:
```
array([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, 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, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 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], dtype=int64)
```

In [117]:
```
data_out = gender_scaler.copy(deep = True)
data_out['Cluster'] = hc_model.labels_
data_out.head()
```

Out[117]:

	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)	Cluster
0	-1.723412	-1.435484	-1.738999	-0.465996	0
1	-1.706091	-1.291178	-1.738999	1.257635	0
2	-1.688771	-1.363331	-1.700830	-1.820277	0
3	-1.671450	-1.146872	-1.700830	1.093479	0
4	-1.654129	-0.569648	-1.662660	-0.424957	0

In [152]:
```
silhouette_avg = silhouette_score(gender_scaler, y_cluster)
print(f"Silhouette Score: {silhouette_avg}")
```

```
Silhouette Score: 0.30765207464216293
```

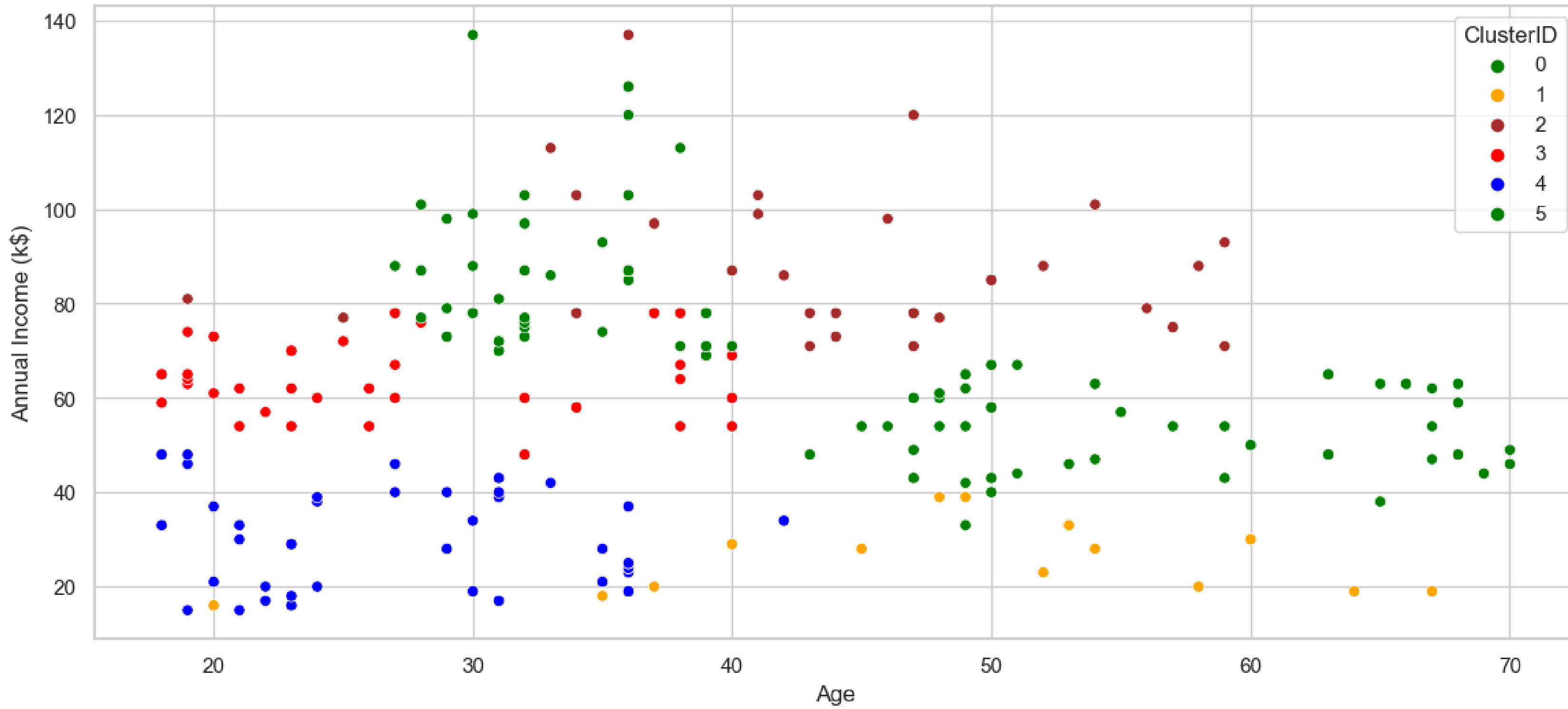
In [119]:
```
calinski_harabasz_index = calinski_harabasz_score(gender_scaler, y_cluster)
print(f"Calinski-Harabasz Index: {calinski_harabasz_index}")
```

```
Calinski-Harabasz Index: 95.12518004324022
```

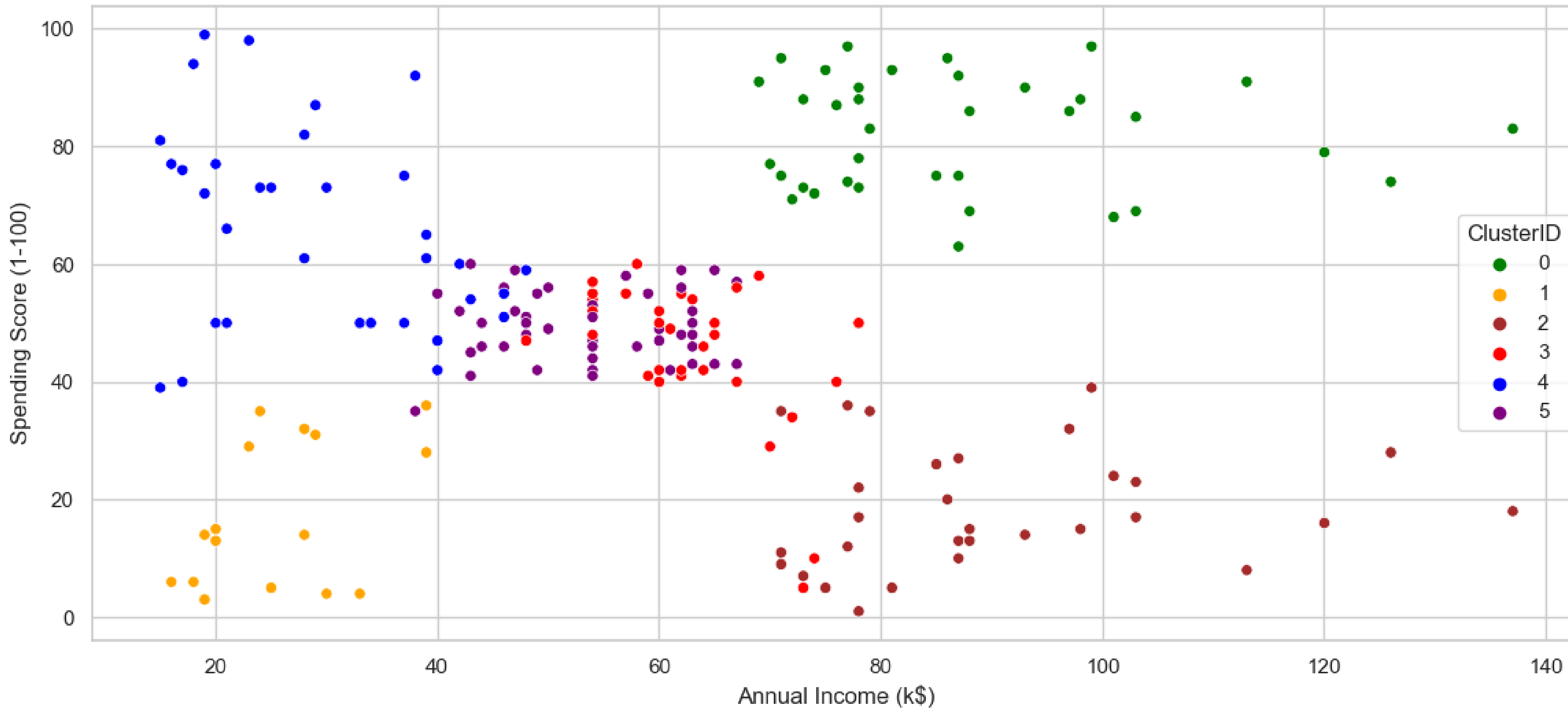
In [67]:
```
davies_bouldin_index = davies_bouldin_score(gender_scaler, y_cluster)
print(f"Davies-Bouldin Index: {davies_bouldin_index}")
```

```
Davies-Bouldin Index: 1.3122719763348933
```

In [124]:
```
#Scatter plot of child_mort in x-axis and income in y-axis with different colors indicating the cluster assig
sns.scatterplot(x='Age',y='Annual Income (k$)',hue='ClusterID',data=data_cluster,palette=['green','orange','br
plt.show()
```



In [164]:
```
#Scatter plot of child_mort in x-axis and income in y-axis with different colors indicating the cluster assign
sns.scatterplot(x='Annual Income (k$)',y='Spending Score (1-100)',hue='ClusterID',data=data_cluster,palette=['
plt.show()
```



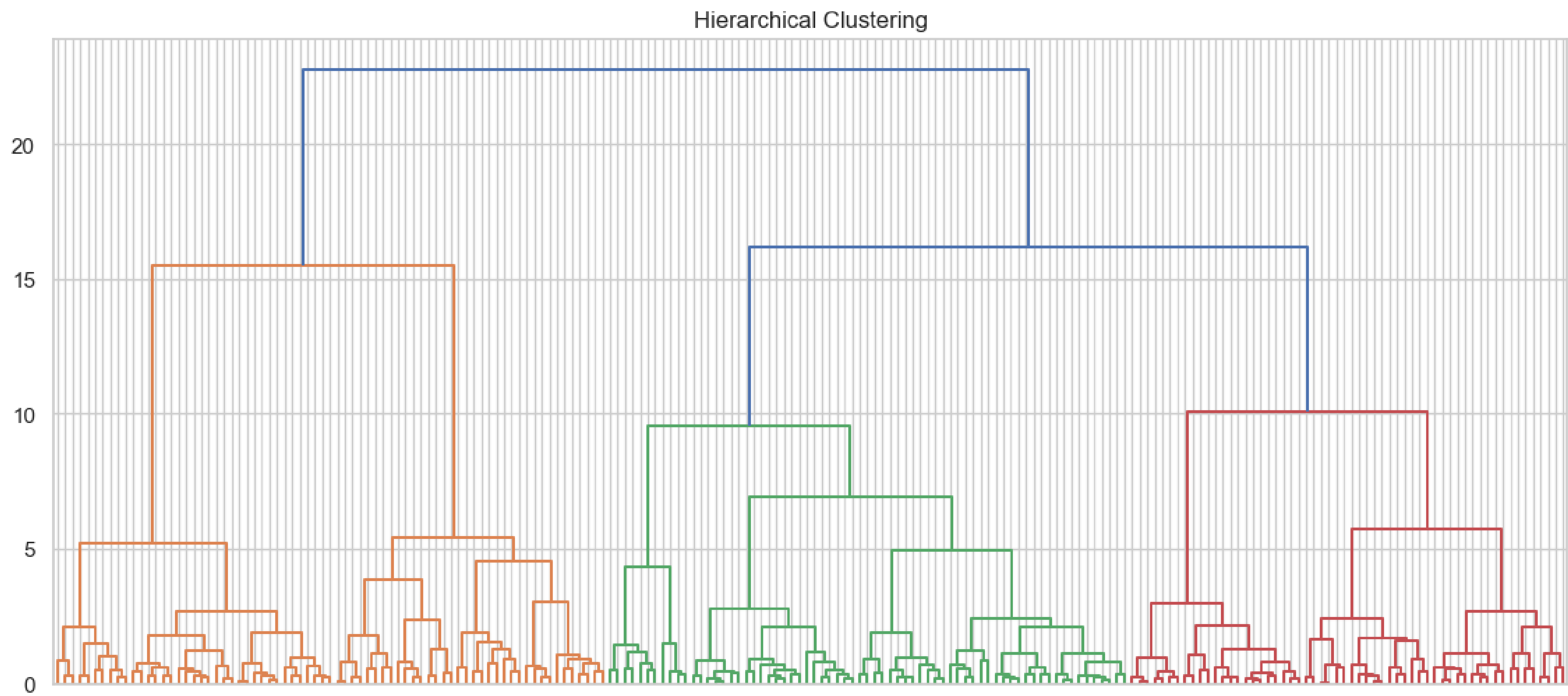
In [130]:
```
#Importing libraries required for Hierarchical clustering
import scipy.cluster.hierarchy as sch
from sklearn.preprocessing import scale as s
from scipy.cluster.hierarchy import dendrogram, linkage, cut_tree
```

In [131]:
```
Z = sch.linkage(gender_scaler, method='ward')       #Using linkage function to perform hierarchical clustering
Z
```

```
       [1.52000000e+02, 1.54000000e+02, 2.19213451e-01, 2.00000000e+00],
       [3.30000000e+01, 3.50000000e+01, 2.19213451e-01, 2.00000000e+00],
       [5.40000000e+01, 5.60000000e+01, 2.23534379e-01, 2.00000000e+00],
       [1.43000000e+02, 1.49000000e+02, 2.29368224e-01, 2.00000000e+00],
       [1.34000000e+02, 1.38000000e+02, 2.31446988e-01, 2.00000000e+00],
       [9.50000000e+01, 9.70000000e+01, 2.34075367e-01, 2.00000000e+00],
       [4.30000000e+01, 5.10000000e+01, 2.34138930e-01, 2.00000000e+00],
       [1.81000000e+02, 1.83000000e+02, 2.37166994e-01, 2.00000000e+00],
       [7.50000000e+01, 7.80000000e+01, 2.37257908e-01, 2.00000000e+00],
       [6.10000000e+01, 2.02000000e+02, 2.39864749e-01, 3.00000000e+00],
       [7.90000000e+01, 2.13000000e+02, 2.40782277e-01, 3.00000000e+00],
       [1.03000000e+02, 2.31000000e+02, 2.55020166e-01, 3.00000000e+00],
       [5.90000000e+01, 2.28000000e+02, 2.58799915e-01, 3.00000000e+00],
       [1.63000000e+02, 1.67000000e+02, 2.62266503e-01, 2.00000000e+00],
       [1.00000000e+00, 2.05000000e+02, 2.62796446e-01, 3.00000000e+00],
       [1.41000000e+02, 2.29000000e+02, 2.64924172e-01, 3.00000000e+00],
       [1.30000000e+02, 1.36000000e+02, 2.64988941e-01, 2.00000000e+00],
       [1.73000000e+02, 1.79000000e+02, 2.74210974e-01, 2.00000000e+00],
       [1.71000000e+02, 1.77000000e+02, 2.79452848e-01, 2.00000000e+00],
```

In [132]:
```
#Creating and Plotting a Dendrogram

den = sch.dendrogram(Z)          #Dendrogram plot based on the hierarchical clustering linkage matrix Z.
plt.tick_params(
    axis='x',
    which='both',
    bottom=False,
    top=False,
    labelbottom=False)
plt.title('Hierarchical Clustering')
```

Out[132]:
```
Text(0.5, 1.0, 'Hierarchical Clustering')
```



In [133]:
```
from sklearn.cluster import AgglomerativeClustering
```

In [134]:
```
hc_model = AgglomerativeClustering(n_clusters = 2, affinity = 'euclidean', linkage ='ward')
```

In [135]:
```
y_cluster = hc_model.fit_predict(gender_scaler)
```

In [136]:
```
data_clustered = gender_scaler.copy()
```

In [137]:
```
data_clustered["Cluster"] = y_cluster.astype('object')
```