In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df = pd.read_csv('Mall_Customers.csv')
```

In [3]:
```
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

In [4]:
```
df.tail()
```

Out[4]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
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 [5]:
```
df.isnull().sum()
```

Out[5]:
```
CustomerID                0
Gender                    0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

In [6]:
```
df.describe()
```

Out[6]:

	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 [7]:
```
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 [8]:
```
df = df.drop('CustomerID',axis=1)
```

In [9]:
```
df.head()
```

Out[9]:

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

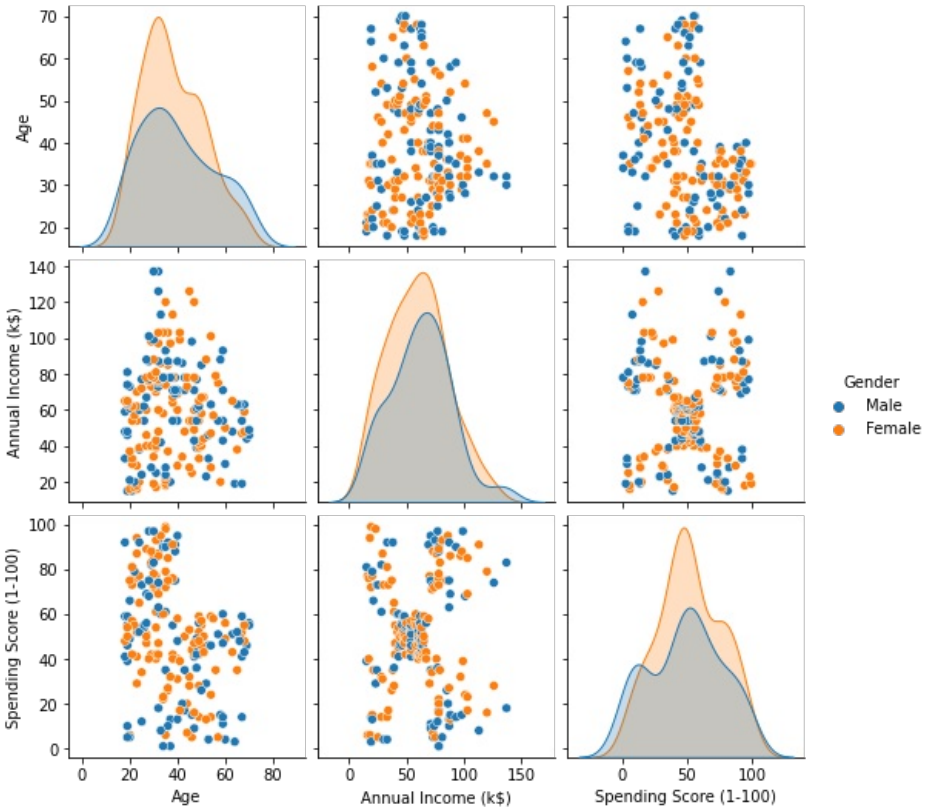
In [10]:
```
#check for special characters
for i in df:
    v=df[i].value_counts()
    print(v)
```

```
Female    112
Male       88
Name: Gender, dtype: int64
32    11
35     9
19     8
31     8
30     7
49     7
27     6
47     6
40     6
23     6
36     6
38     6
50     5
48     5
29     5
21     5
20     5
34     5
18     4
28     4
59     4
24     4
67     4
54     4
39     3
25     3
33     3
22     3
37     3
43     3
68     3
45     3
46     3
60     3
41     2
57     2
66     2
65     2
63     2
58     2
26     2
70     2
42     2
53     2
52     2
51     2
44     2
55     1
64     1
69     1
56     1
Name: Age, dtype: int64
54     12
78     12
60      6
87      6
62      6
       ..
61      2
```

```
126     2
59      2
58      2
15      2
Name: Annual Income (k$), Length: 64, dtype: int64
42    8
55    7
46    6
73    6
35    5
     ..
31    1
82    1
24    1
23    1
99    1
Name: Spending Score (1-100), Length: 84, dtype: int64
```

In [11]:
```
sns.pairplot(df,hue='Gender')
```

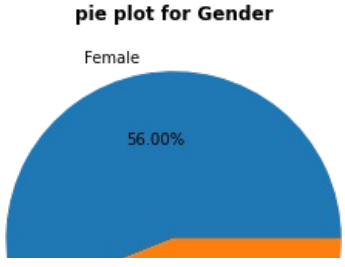
Out[11]:
```
<seaborn.axisgrid.PairGrid at 0x1b57e70f8e0>
```

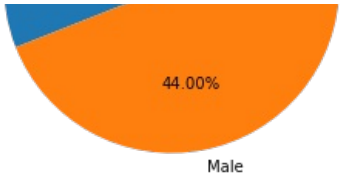


In [12]:
```
v1 = df['Gender'].value_counts()
v1
```

Out[12]:
```
Female    112
Male       88
Name: Gender, dtype: int64
```

In [13]:
```
label = v1.index
plt.figure(figsize=(10,5))
plt.pie(v1,labels=label, autopct='%.2f%%')
plt.title('pie plot for Gender',fontweight ='bold',size=12)
plt.show()
```





In [14]:
```
df.Age.min()
```

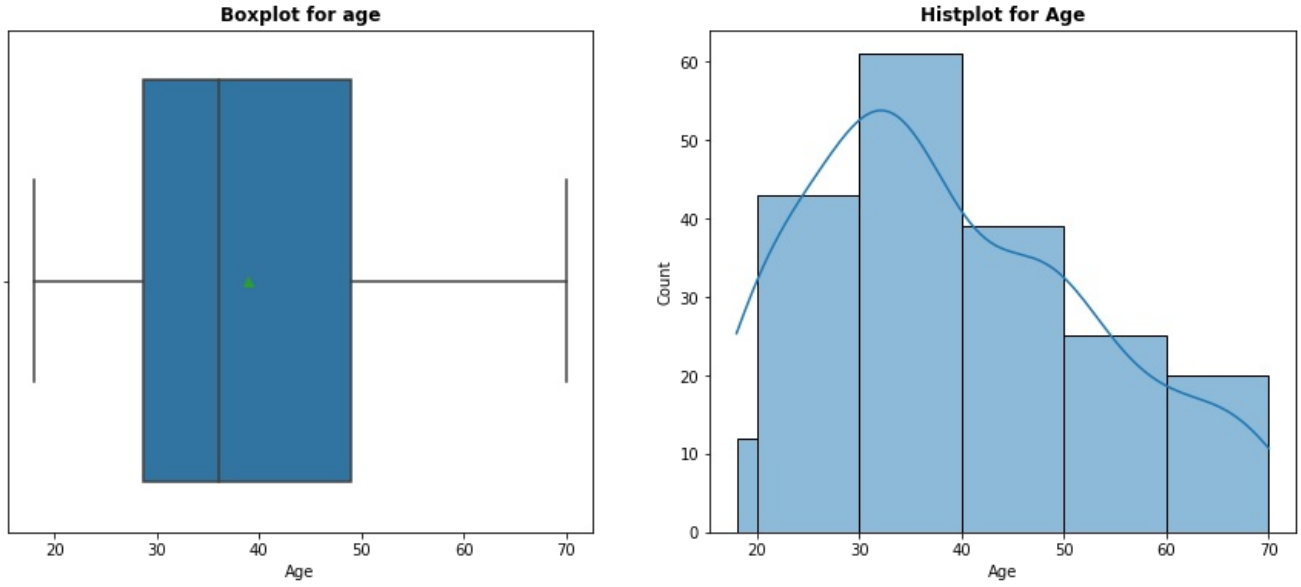
Out[14]: 18

In [15]:
```
df.Age.max()
```

Out[15]: 70

In [16]:
```
b1 =[18,20,30,40,50,60,70]
```

In [17]:
```
plt.subplots(1,2,figsize=(15,6))
plt.subplot(121)

sns.boxplot(x='Age',data=df,showmeans=True)
plt.title('Boxplot for age',fontweight='bold',size = 12)

plt.subplot(122)
ax = sns.histplot(data=df,x= 'Age',bins=b1,kde = True)


plt.title('Histplot for Age',fontweight = 'bold',size=12)
plt.show()
```



In [18]:
```
q = df['Age'].quantile(0.9)
m = df['Age'].mean()

print(f'90% of the data lies before Age {q}\n')
print(f'Mean of the data lies at {m}\n')
```

```
90% of the data lies before Age 59.099999999999994

Mean of the data lies at 38.85
```

In [19]:
```
df['Annual Income (k$)'].min()
```

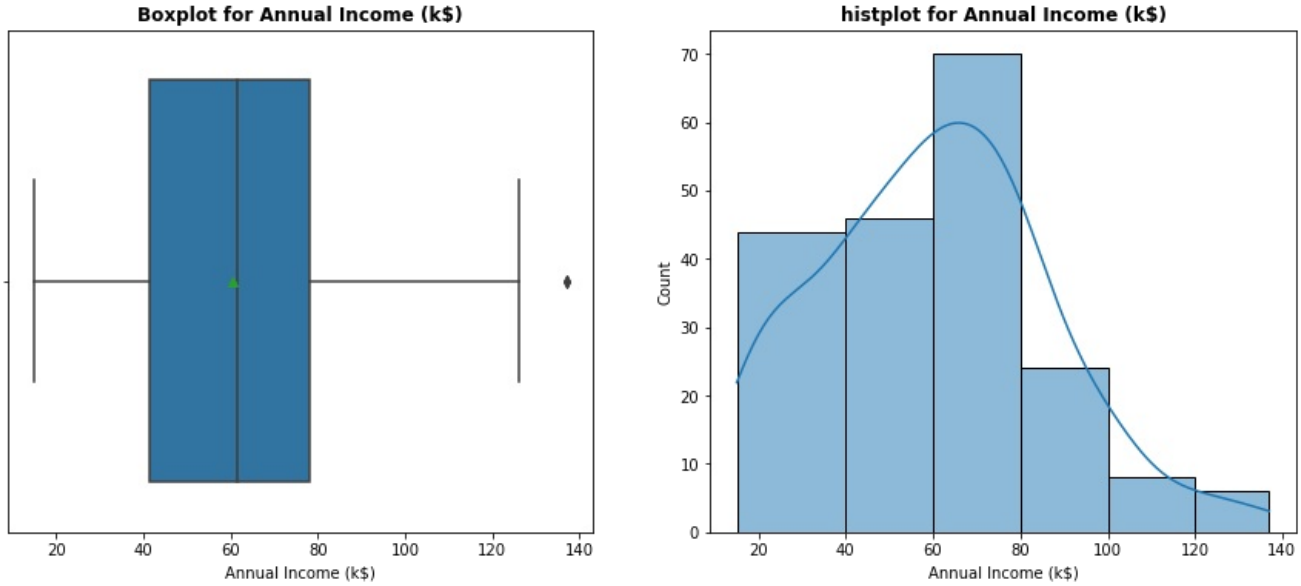
Out[19]: 15

In [20]:
```
df['Annual Income (k$)'].max()
```

Out[20]: 137

In [21]:
```
b2 =[15.20,40,60,80,100,120,137]
```

In [22]:
```
plt.subplots(1,2,figsize=(15,6))

plt.subplot(121)
sns.boxplot(x = 'Annual Income (k$)',data=df,showmeans=True)
plt.title('Boxplot for Annual Income (k$)',fontweight='bold',size= 12)

plt.subplot(122)
ax1 = sns.histplot(data=df,x='Annual Income (k$)',bins=b2,kde=True)
plt.title('histplot for Annual Income (k$)',fontweight='bold',size= 12)
plt.show()
```



In [23]:
```
q = df['Annual Income (k$)'].quantile(0.9)
m = df['Annual Income (k$)'].mean()

print(f'90% of the data lies before Age {q}\n')
print(f'Mean of the data lies at {m}\n')
```

```
90% of the data lies before Age 93.39999999999998

Mean of the data lies at 60.56
```

In [24]:
```
df['Spending Score (1-100)'].min()
```

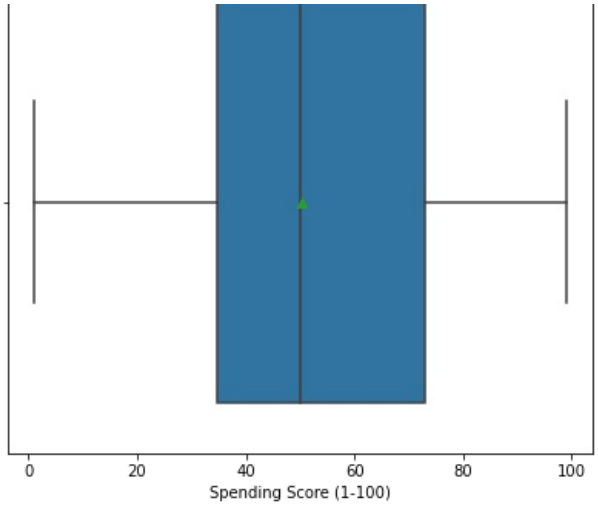
Out[24]: 1

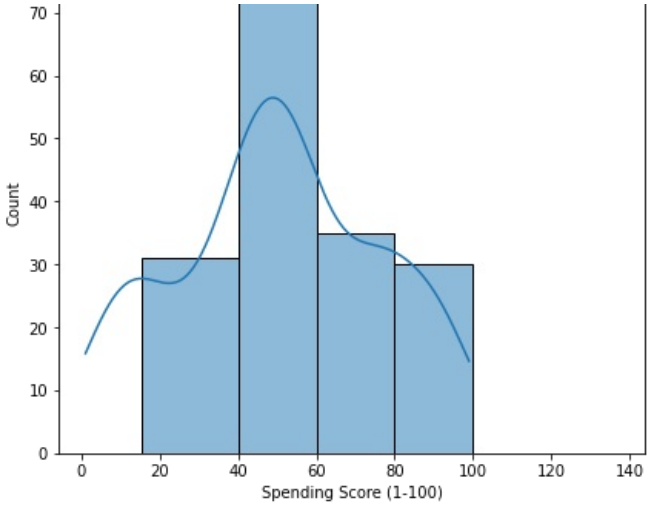
In [25]:
```
df['Spending Score (1-100)'].max()
```

Out[25]: 99

In [26]:
```
b3= [1,10,20,30,40,50,60,70,80,90,99]
```

In [27]:
```
plt.subplots(1,2,figsize=(15,6))

plt.subplot(121)
sns.boxplot(x = 'Spending Score (1-100)',data=df,showmeans=True)
plt.title('Boxplot for Spending Score (1-100)',fontweight='bold',size= 12)

plt.subplot(122)
ax1 = sns.histplot(data=df,x='Spending Score (1-100)',bins=b2,kde=True)
plt.title('histplot for Spending Score(1-100)',fontweight='bold',size= 12)
plt.show()
```

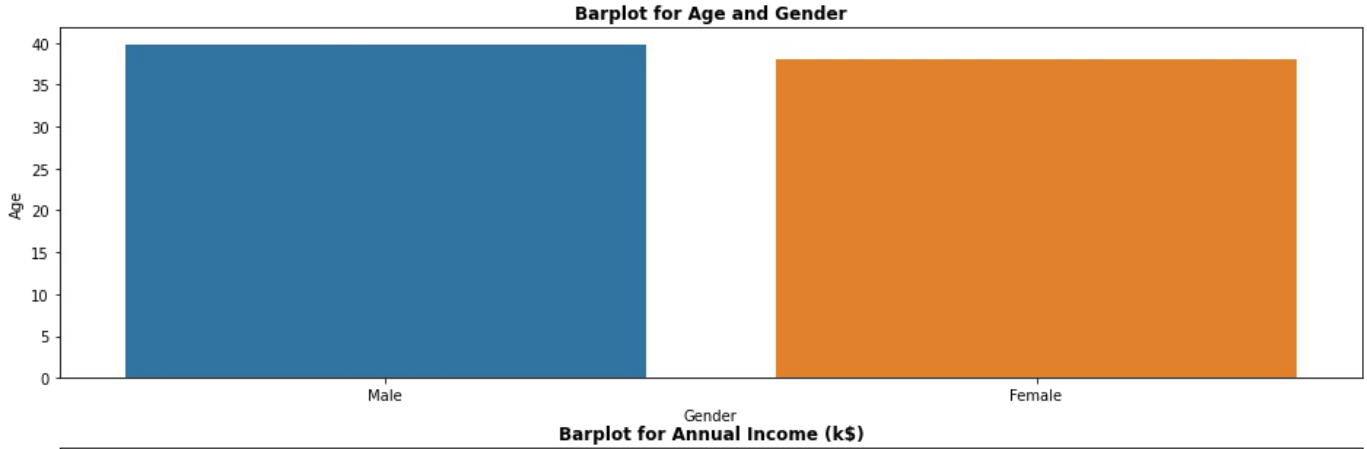


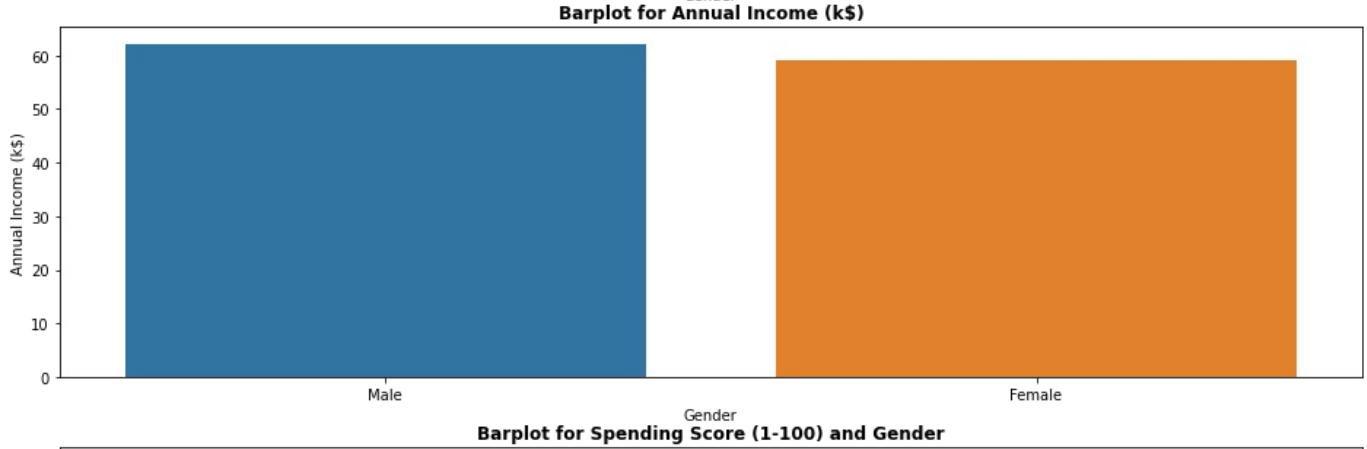


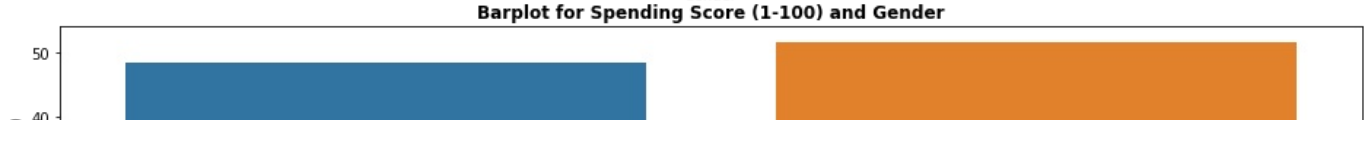
In [28]:
```
q = df['Spending Score (1-100)'].quantile(0.9)
m = df['Spending Score (1-100)'].mean()

print(f'90% of the data lies before Age {q}\n')
print(f'Mean of the data lies at {m}\n')
```

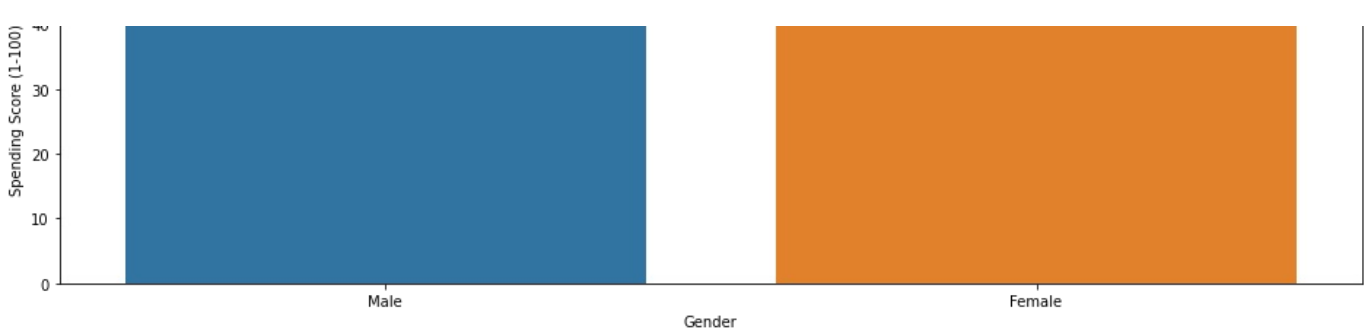
```
90% of the data lies before Age 87.1

Mean of the data lies at 50.2
```

In [41]:
```
plt.subplots(3,1,figsize=(16,15))
plt.subplot(311)
ax3 = sns.barplot(data=df,y='Age',x='Gender',ci=None)
plt.title('Barplot for Age and Gender',fontweight='bold',size=12)
plt.subplot(312)
ax3 = sns.barplot(data=df,y='Annual Income (k$)',x= 'Gender',ci=None)

plt.title('Barplot for Annual Income (k$)',fontweight='bold',size=12)
plt.subplot(313)
ax3 = sns.barplot(data=df,y='Spending Score (1-100)',x= 'Gender',ci=None)
plt.title('Barplot for Spending Score (1-100) and Gender',fontweight='bold',size=12)
plt.show()
```

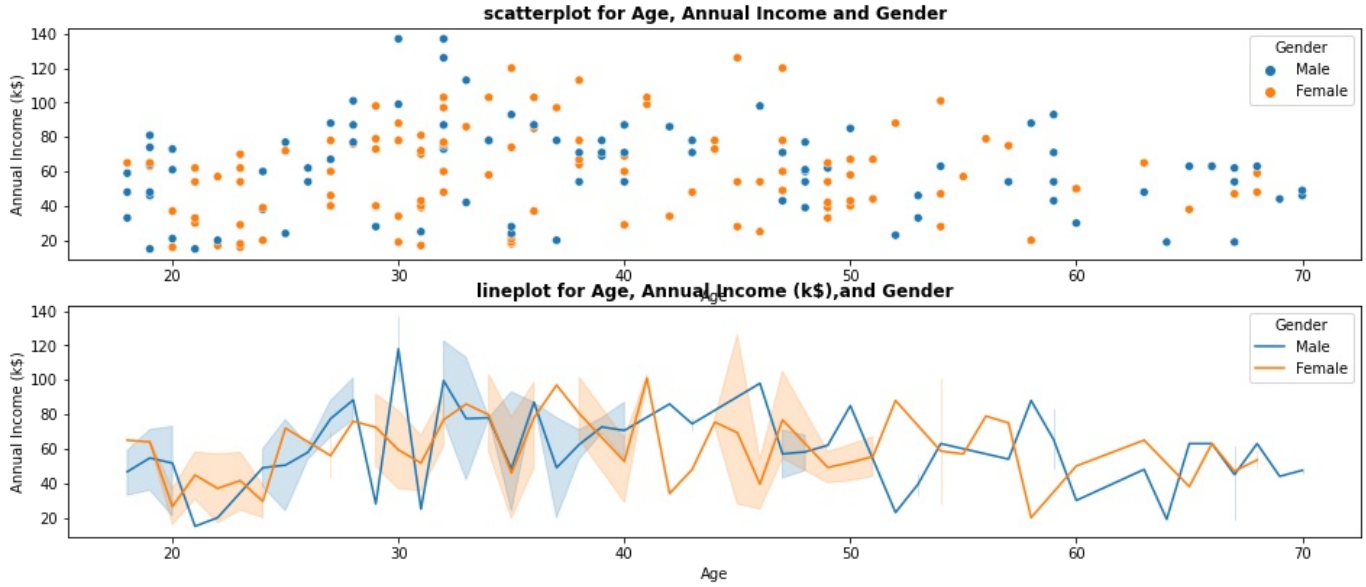






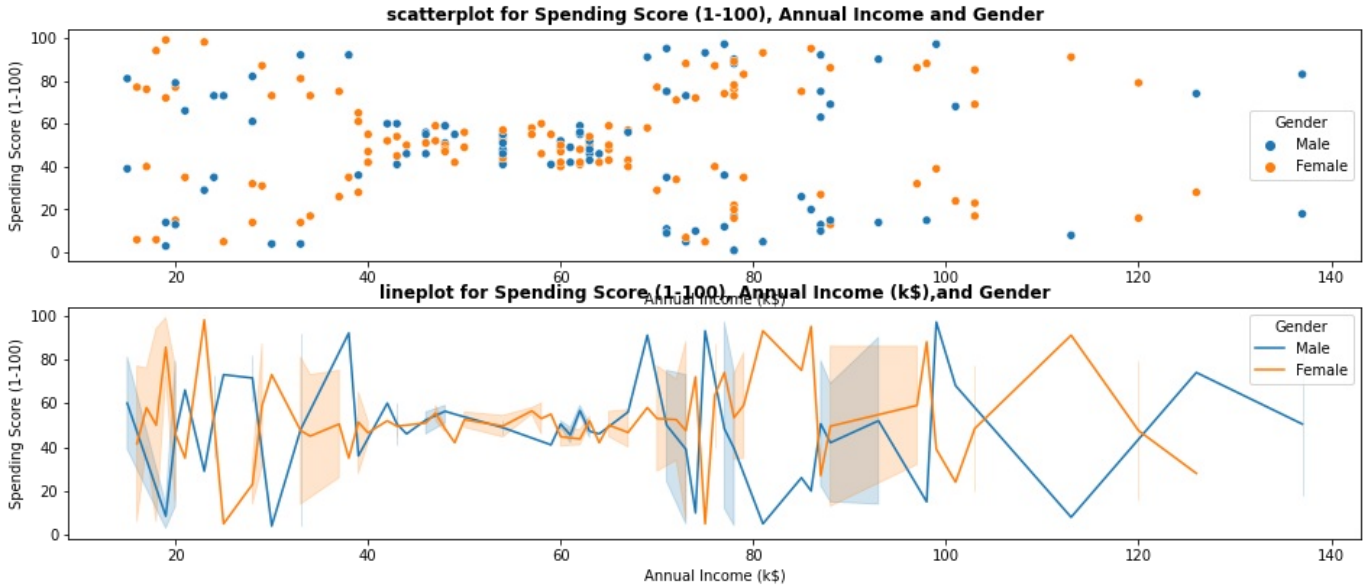


In [43]:
```
plt.subplots(2,1,figsize=(16,10))
plt.subplot(311)
sns.scatterplot(data=df,y='Annual Income (k$)',x='Age',hue = 'Gender')
plt.title('scatterplot for Age, Annual Income and Gender',fontweight='bold',size=12)
plt.subplot(312)
sns.lineplot(data=df,y='Annual Income (k$)',x= 'Age',hue = 'Gender')

plt.title('lineplot for Age, Annual Income (k$),and Gender',fontweight='bold',size=12)
plt.show()
```



In [44]:
```
plt.subplots(2,1,figsize=(16,10))
plt.subplot(311)
sns.scatterplot(data=df,y='Spending Score (1-100)',x='Age',hue = 'Gender')
plt.title('scatterplot for Age, Spending Score and Gender',fontweight='bold',size=12)
plt.subplot(312)
sns.lineplot(data=df,y='Spending Score (1-100)',x= 'Age',hue = 'Gender')

plt.title('lineplot for Age, Annual Income (k$),and Gender',fontweight='bold',size=12)
plt.show()
```

In [45]:
```
plt.subplots(2,1,figsize=(16,10))
plt.subplot(311)
sns.scatterplot(data=df,x='Annual Income (k$)',y='Spending Score (1-100)',hue = 'Gender')
plt.title('scatterplot for Spending Score (1-100), Annual Income and Gender',fontweight='bold',size=12)
plt.subplot(312)
sns.lineplot(data=df,x='Annual Income (k$)',y= 'Spending Score (1-100)',hue = 'Gender')

plt.title('lineplot for Spending Score (1-100), Annual Income (k$),and Gender',fontweight='bold',size=12)
plt.show()
```



In [46]:
```
df['Gender'] = df['Gender'].astype('category')
```

In [47]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column                  Non-Null Count  Dtype   
---  ------                  --------------  -----   
 0   Gender                  200 non-null    category
 1   Age                     200 non-null    int64   
 2   Annual Income (k$)      200 non-null    int64   
 3   Spending Score (1-100)  200 non-null    int64   
dtypes: category(1), int64(3)
memory usage: 5.1 KB
```

In [49]:
```
from sklearn.preprocessing import LabelEncoder
```

In [53]:
```
Label_Encoder = LabelEncoder()

df['Gender'] = Label_Encoder.fit_transform(df['Gender'])
```

In [54]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Gender                  200 non-null    int32
 1   Age                     200 non-null    int64
 2   Annual Income (k$)      200 non-null    int64
 3   Spending Score (1-100)  200 non-null    int64
dtypes: int32(1), int64(3)
memory usage: 5.6 KB
```

In [58]:
```
from sklearn.preprocessing import StandardScaler,MinMaxScaler
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings('ignore')
```

In [80]:
```
from sklearn.cluster import KMeans,AgglomerativeClustering,DBSCAN
```

In [81]:
```
scaler = StandardScaler()
```

In [81]:
```
scaler = StandardScaler()
numerics = ['int16','int32','int64']
df = df.select_dtypes(include=numerics)
scaled_df1 = pd.DataFrame(scaler.fit_transform(df.to_numpy()),columns=df.columns)
scaled_df1
```

Out[81]:

	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1.128152	-1.424569	-1.738999	-0.434801
1	1.128152	-1.281035	-1.738999	1.195704
2	-0.886405	-1.352802	-1.700830	-1.715913
3	-0.886405	-1.137502	-1.700830	1.040418
4	-0.886405	-0.563369	-1.662660	-0.395980
...	...	...	...	...
195	-0.886405	-0.276302	2.268791	1.118061
196	-0.886405	0.441365	2.497807	-0.861839
197	1.128152	-0.491602	2.497807	0.923953
198	1.128152	-0.491602	2.917671	-1.250054
199	1.128152	-0.635135	2.917671	1.273347

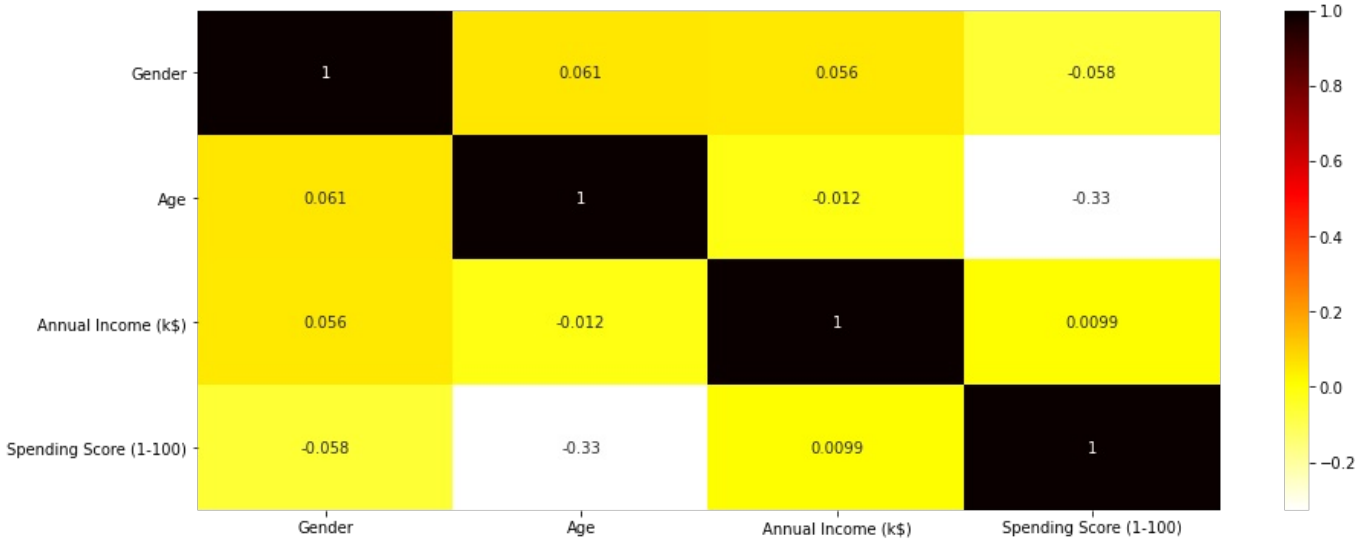
200 rows × 4 columns

In [82]:
```
v = scaled_df1.corr()
v
```

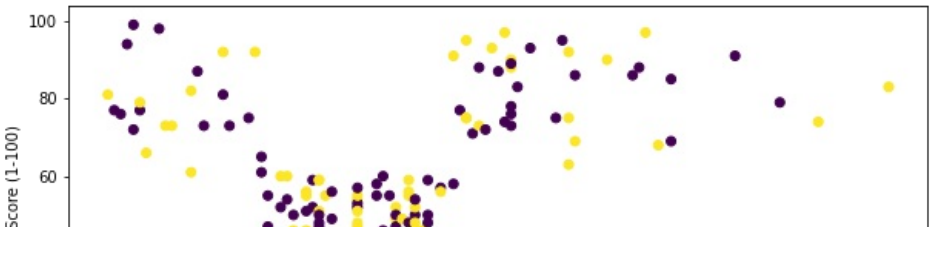
Out[82]:

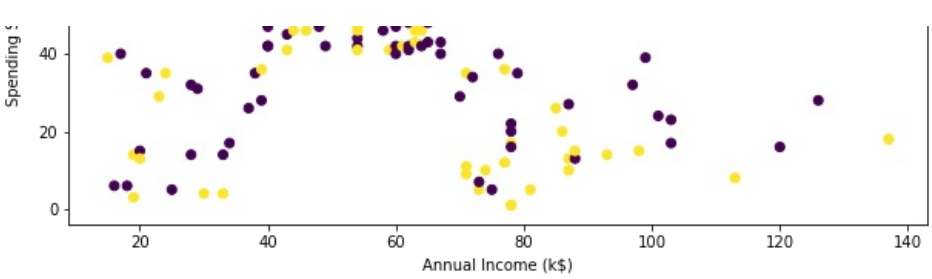
	Gender	Age	Annual Income (k$)	Spending Score (1-100)
Gender	1.000000	0.060867	0.056410	-0.058109
Age	0.060867	1.000000	-0.012398	-0.327227
Annual Income (k$)	0.056410	-0.012398	1.000000	0.009903
Spending Score (1-100)	-0.058109	-0.327227	0.009903	1.000000

In [83]:
```
plt.figure(figsize=(15,6))
sns.heatmap(v,annot=True,cmap='hot_r')
plt.show()
```



In [84]:
```
plt.figure(figsize=(10,5))
plt.scatter(df.iloc[:,2],df.iloc[:,3],c=df['Gender'])
plt.ylabel('Spending Score (1-100)')
plt.xlabel('Annual Income (k$)')
plt.show()
```

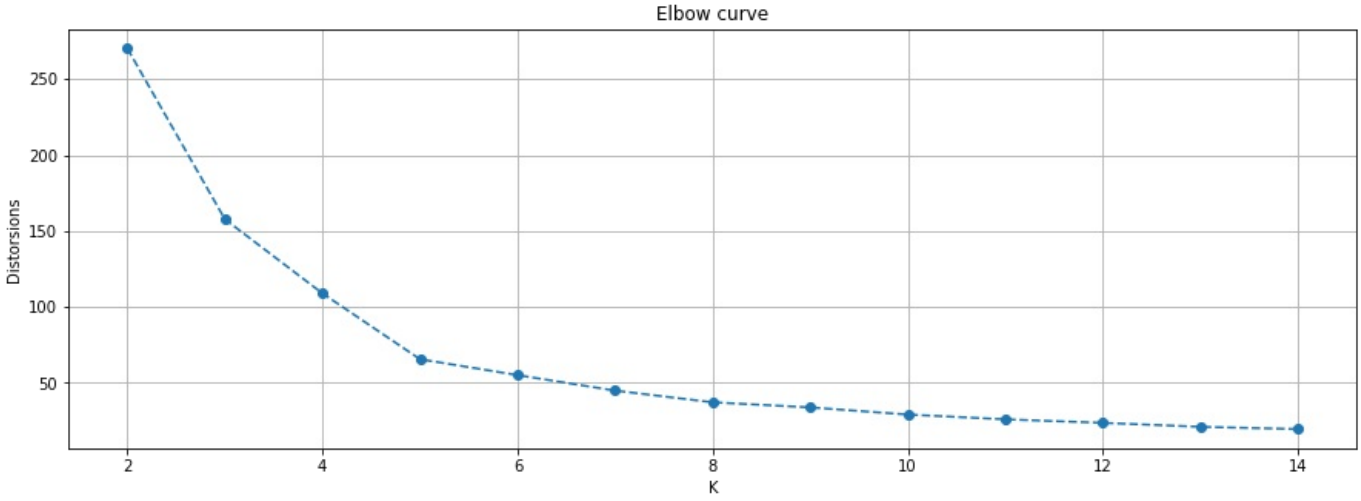




Model Building

K-means Clustering

In [86]:
```
distorsions = []
for k in range(2,15):
    kmeans = KMeans(n_clusters=k,init='k-means++',max_iter=300,n_init=10)
    kmeans.fit(scaled_df1.iloc[:,2:])
    distorsions.append(kmeans.inertia_)
fig = plt.figure(figsize=(15, 5))
plt.plot(range(2,15),distorsions,'o--')
plt.grid(True)
plt.xlabel('K')
plt.ylabel('Distorsions')
plt.title('Elbow curve');
```



In [88]:
```
from sklearn.metrics import silhouette_score
```

In [91]:
```
clusters_range = np.arange(2,10)
```

In [95]:
```
for cluster in clusters_range:
    kmeans = KMeans(n_clusters=cluster,init='k-means++',max_iter=300,n_init=10)
    kmeans.fit_predict(scaled_df1.iloc[:,2:])
    score = silhouette_score(scaled_df1.iloc[:,2:],kmeans.labels_, metric='euclidean')
    print(f'for cluster: {cluster} --> Silhouette Score: %.3f' % score)
```

```
for cluster: 2 --> Silhouette Score: 0.295
for cluster: 3 --> Silhouette Score: 0.467
for cluster: 4 --> Silhouette Score: 0.494
for cluster: 5 --> Silhouette Score: 0.555
for cluster: 6 --> Silhouette Score: 0.540
for cluster: 7 --> Silhouette Score: 0.453
for cluster: 8 --> Silhouette Score: 0.457
for cluster: 9 --> Silhouette Score: 0.447
```

In [96]:
```
model = KMeans(n_clusters=5)
```

In [97]:
```
labels = model.fit_predict(scaled_df1.iloc[:,2:])
labels
```

Out[97]:
```
array([3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3,
       3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 4,
       3, 1, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
```

```
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 0, 2, 0, 4, 0, 2, 0, 2, 0,
4, 0, 2, 0, 2, 0, 2, 0, 2, 0, 4, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2,
2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2,
2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2,
2, 0])
```

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js