Customer Segmentation (EDA) :

Problem Statement :

Understand the target customers for the strategic team to make decision related to marketing

Context :

Identify the most important shopping groups based on income, age and the mall shopping score.

Objective :

- Divide mall target market into approchable groups.
- Creates the subset of markets based on demographic behaviour criteria to better understand the target for marketing

Import Libraries :

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as py
import warnings
warnings.filterwarnings("ignore")
```

Import Dataset :

In [2]:
```
data = pd.read_csv(r"\Customer Segmentation\Customers.csv")
```

In [3]:
```
data
```

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
...	...	...	...	...	...
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

Explore the dataset :

In [4]:
```
data.head()
```

Out[4]:

	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 [5]:
```
data.tail()
```

Out[5]:

	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 [6]:
```
data.shape
```

Out[6]:
```
(200, 5)
```

In [7]:
```
data.columns
```

Out[7]:
```
Index(['CustomerID', 'Gender', 'Age', 'Annual Income (k$)',
       'Spending Score (1-100)'],
      dtype='object')
```

In [8]:
```
data.index
```

Out[8]:
```
RangeIndex(start=0, stop=200, step=1)
```

In [9]:
```
data.nunique()
```

Out[9]:
```
CustomerID                200
Gender                      2
Age                        51
Annual Income (k$)         64
Spending Score (1-100)     84
dtype: int64
```

In [10]:
```
data.count()
```

Out[10]:
```
CustomerID                200
Gender                    200
Age                       200
Annual Income (k$)        200
Spending Score (1-100)    200
dtype: int64
```

In [11]:
```
data.isna().sum()  ## To check null values
```

Out[11]:
```
CustomerID                0
Gender                    0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

In [12]:
```
data[data.duplicated()]  ## To check the duplicate values
```

Out[12]:

CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)

In [13]:
```
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                     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 [14]:
```
data.describe()
```

Out[14]:

	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 [15]:

```
data.select_dtypes("O")  ## Columns with Object data type
```

Out[15]:

	Gender
0	Male
1	Male
2	Female
3	Female
4	Female
...	...
195	Female
196	Female
197	Male
198	Male
199	Male

200 rows × 1 columns

In [16]:

```
data.select_dtypes("int")   ## Columns with interger data type
```

Out[16]:

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

200 rows × 4 columns

Univariate Analysis :

In [17]:

```
data.describe()
```

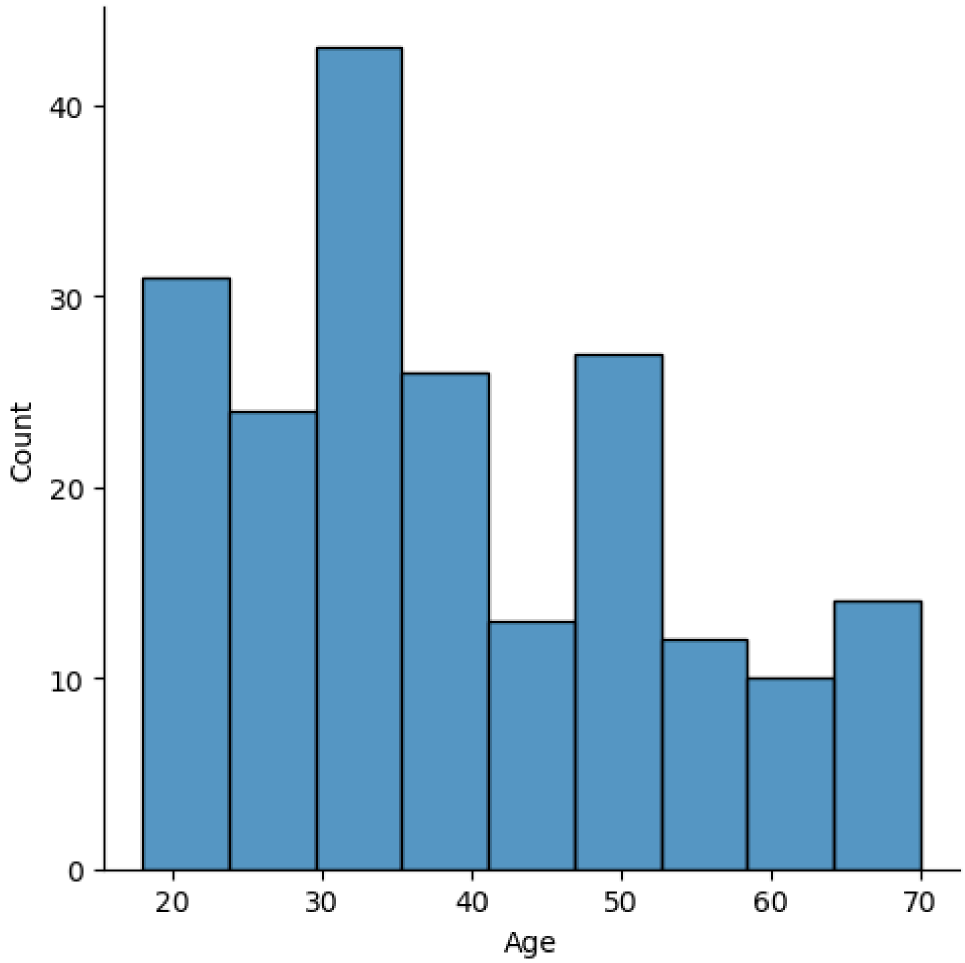
Out[17]:

	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

Age Distribution :

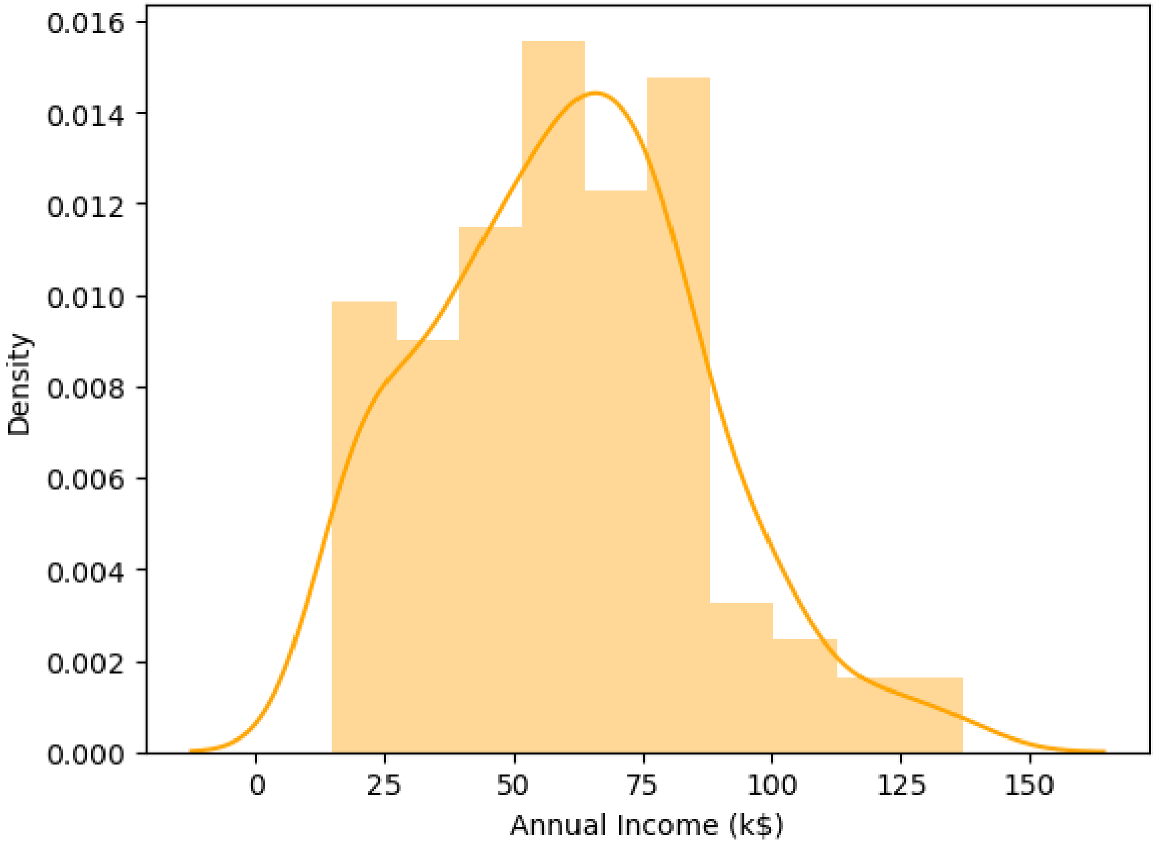
In [18]:

```
sns.displot(data["Age"])

py.show()
```



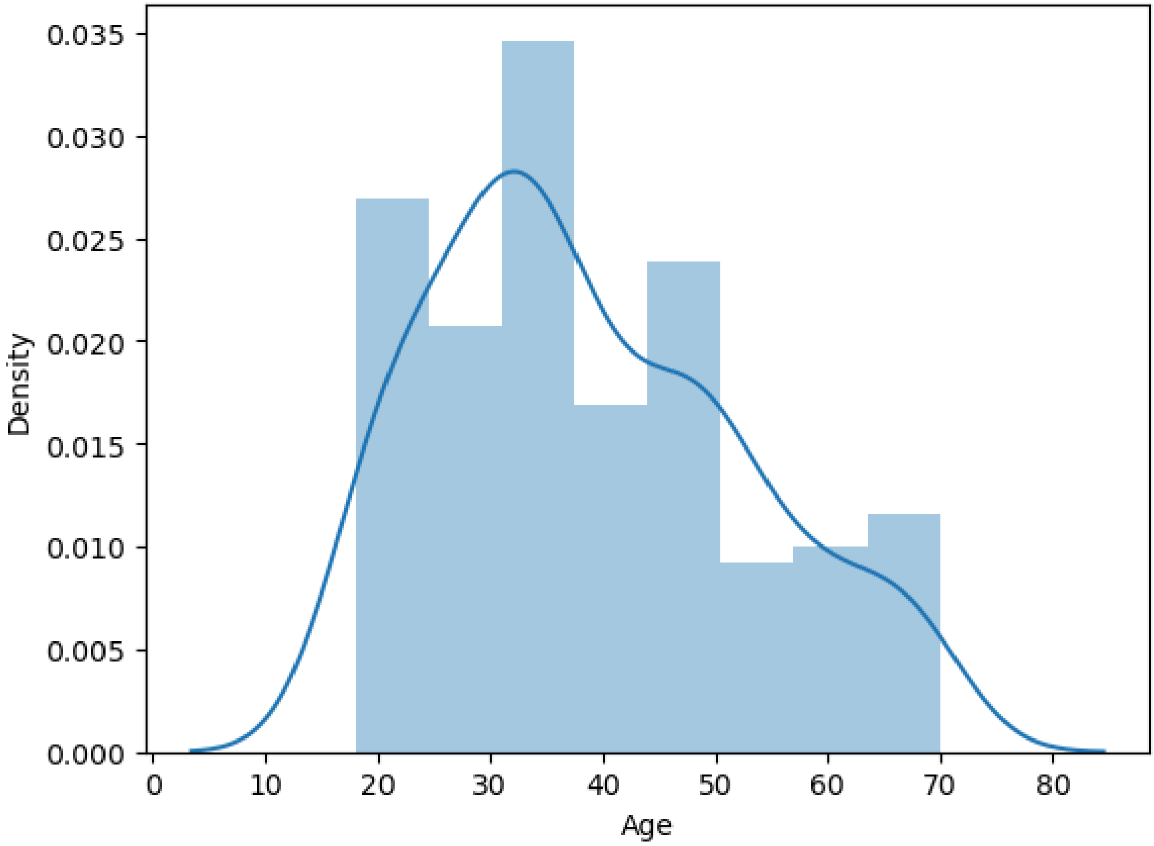
Annual Income Distribution :

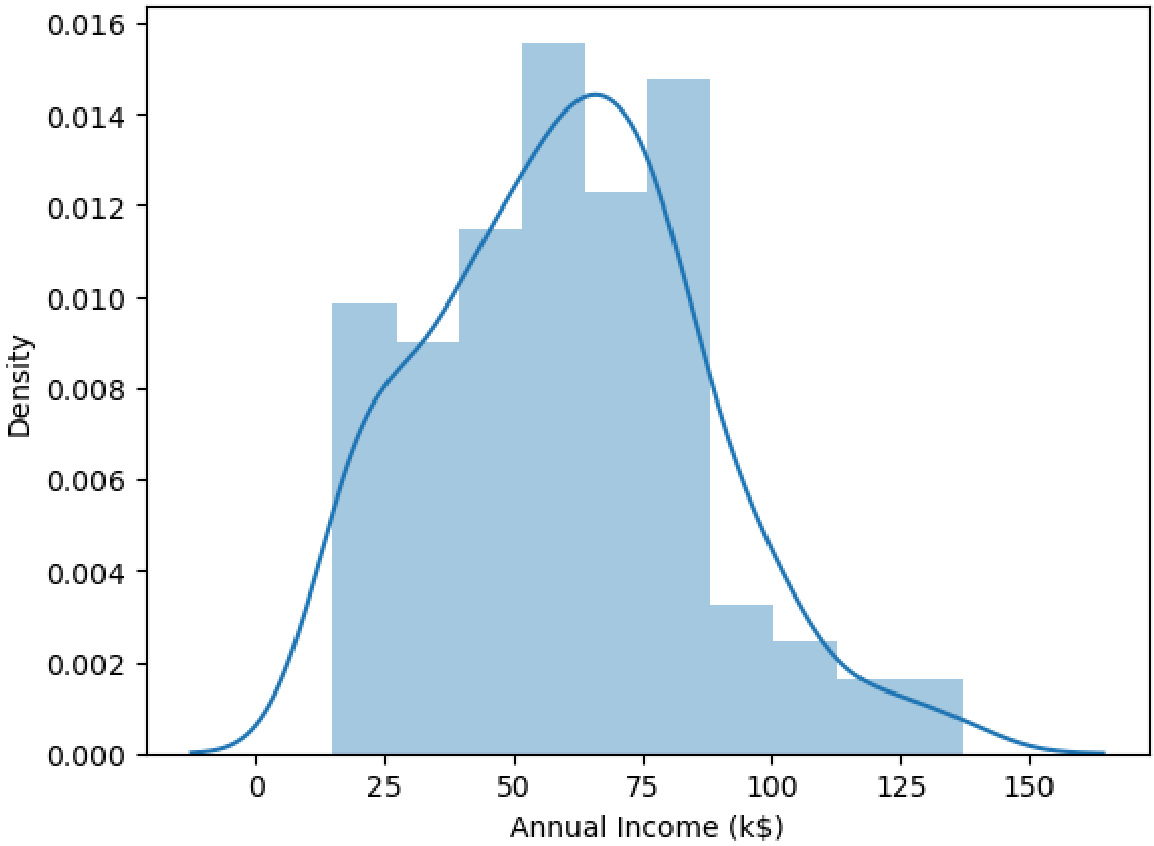
In [19]:
```
sns.distplot(data["Annual Income (k$)"], color = "orange")

py.show()
```

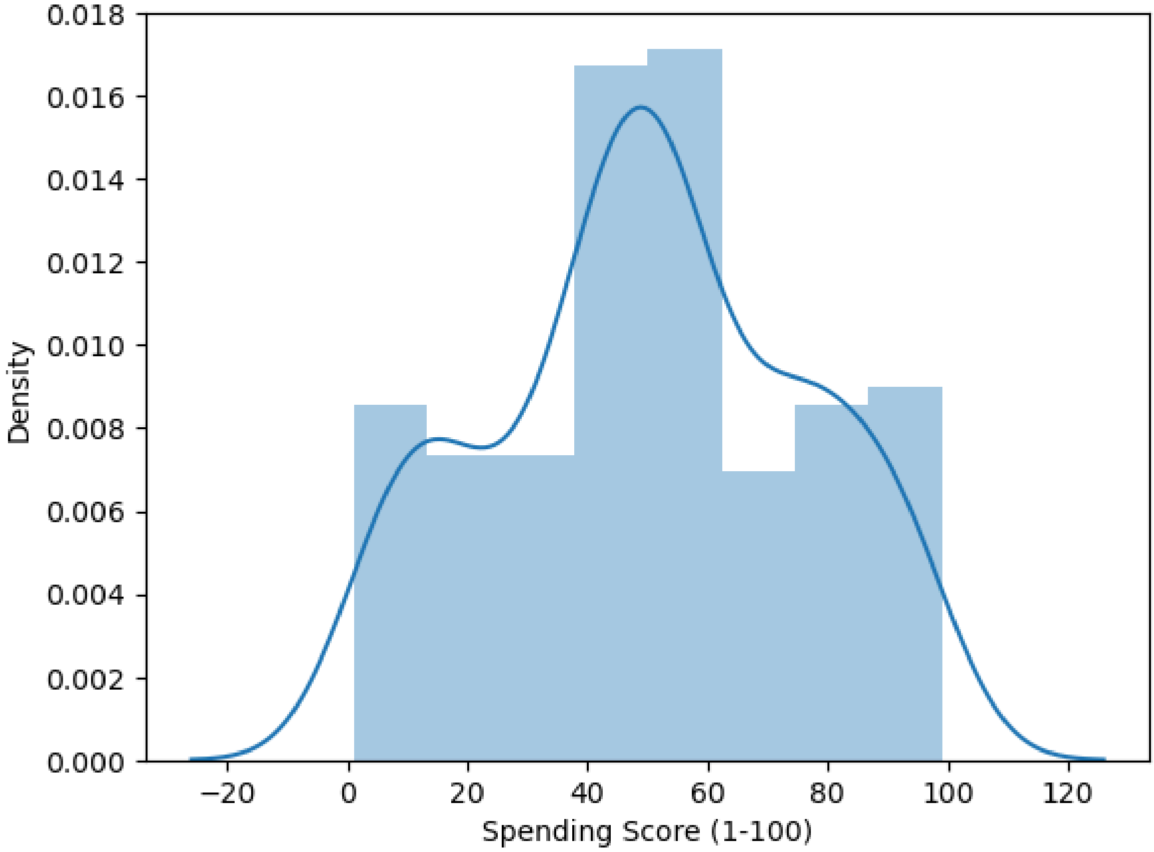


Age, Annual Income, Spending Score (1-100) Distribution :

In [20]:
```
columns = ["Age" , "Annual Income (k$)" , "Spending Score (1-100)"]

for i in columns:
    sns.distplot(data[i])
    py.show()
```

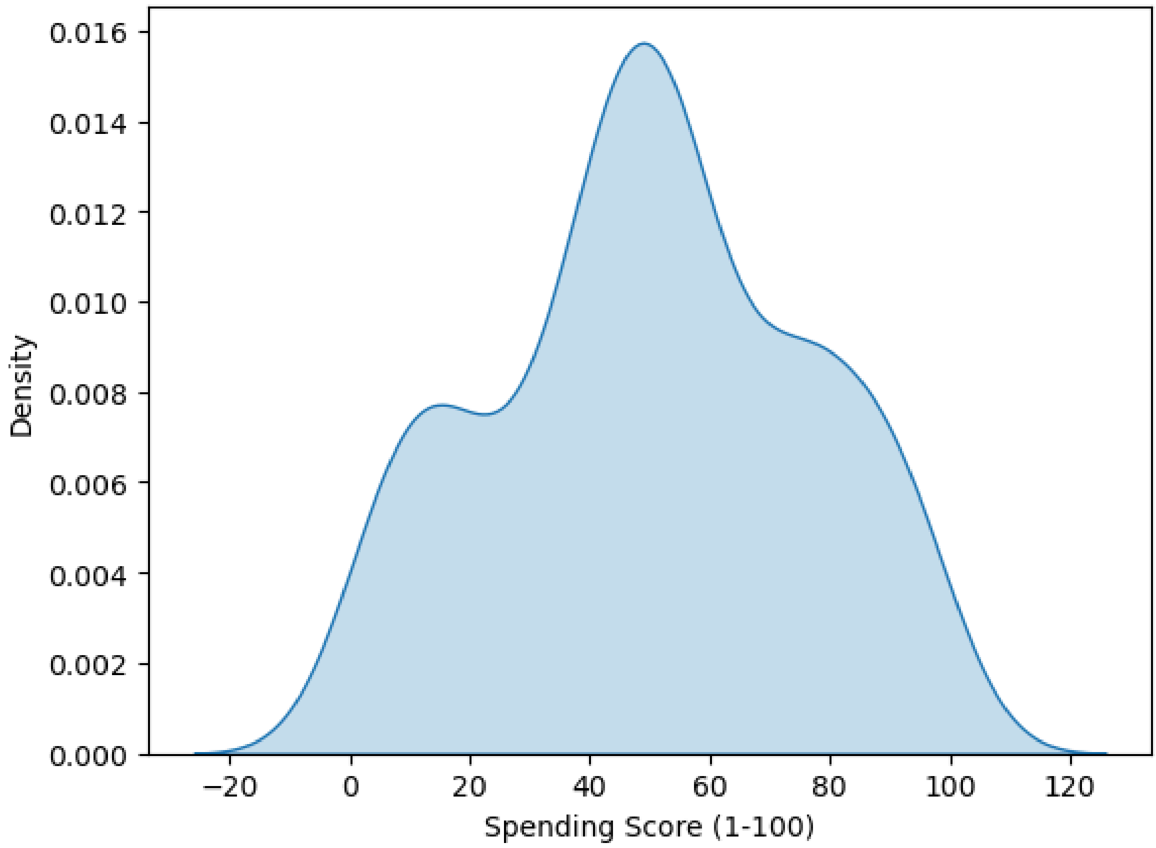




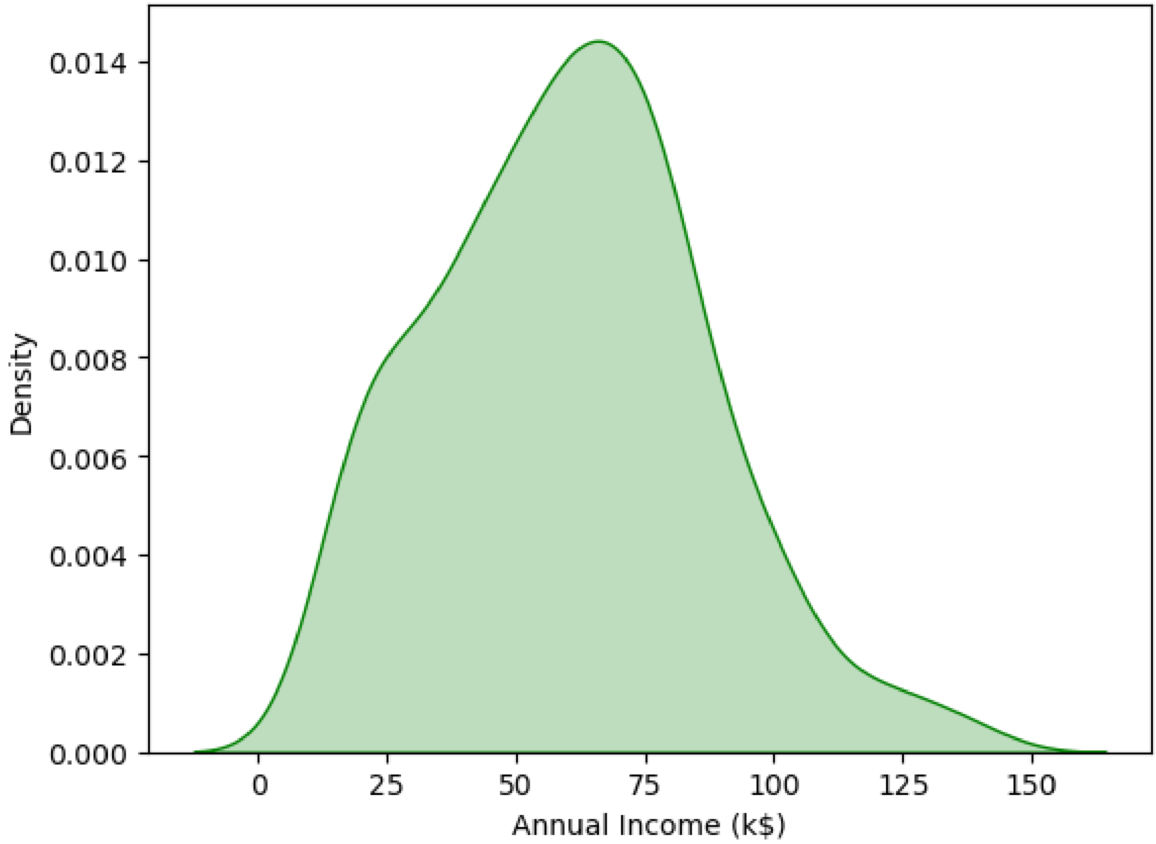


In [21]:

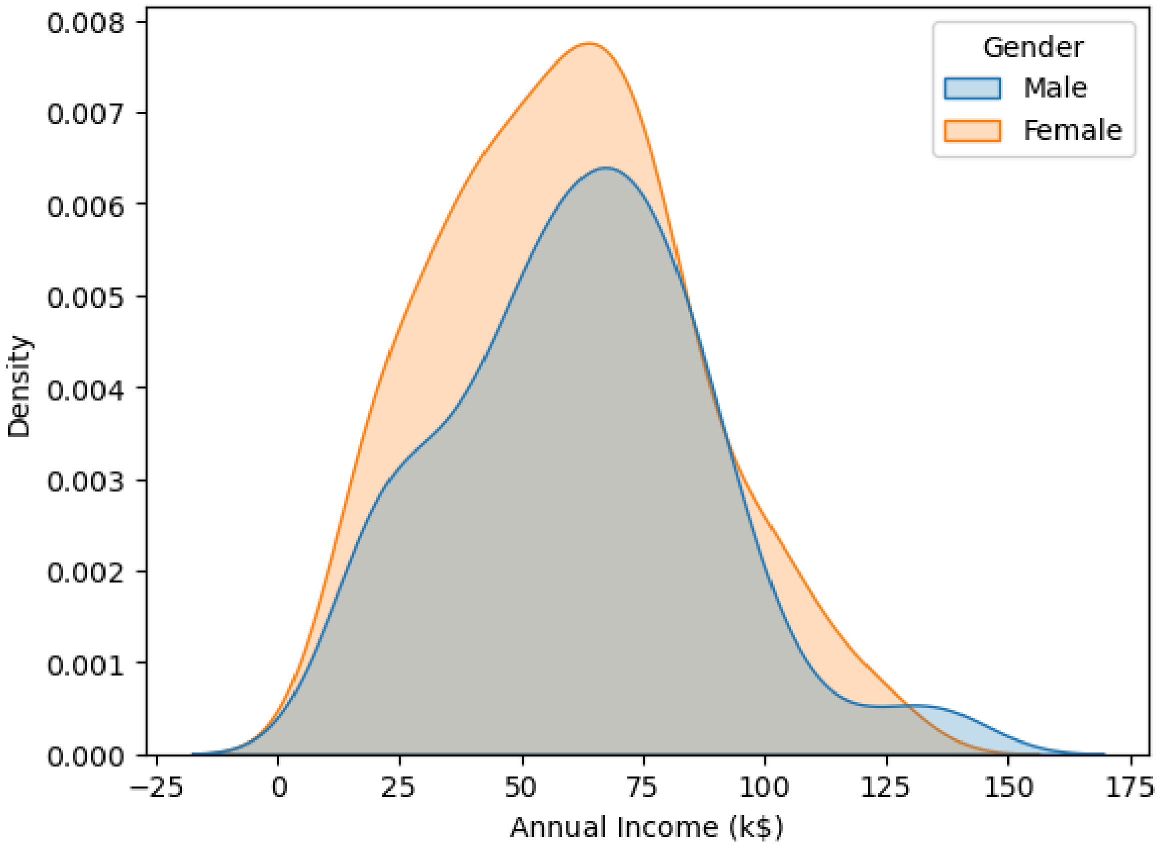
```
sns.kdeplot(data["Spending Score (1-100)"] , shade = True)
py.show()
```



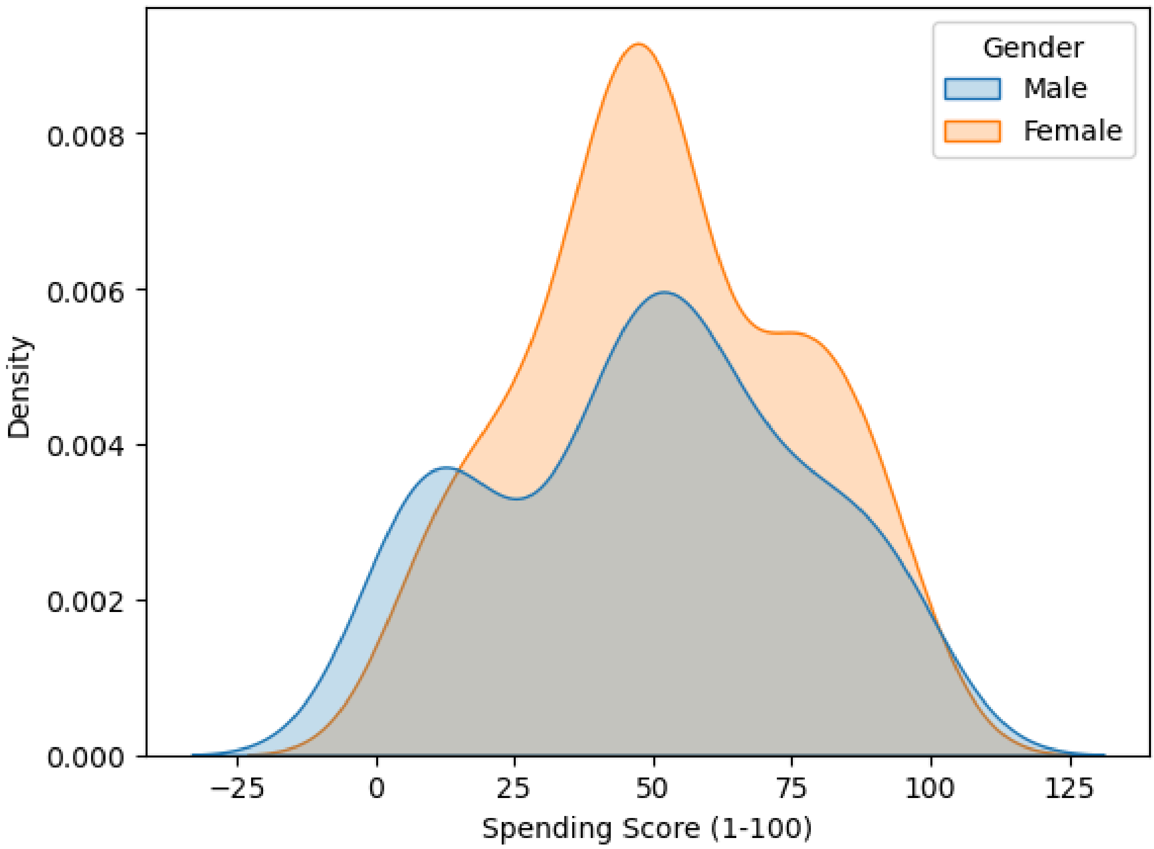
In [22]:
```
sns.kdeplot(data["Annual Income (k$)"] , color = "Green" , shade = True)
py.show()
```



In [23]:
```
sns.kdeplot(data["Annual Income (k$)"] , hue = data["Gender"] , shade = True)

py.show()
```



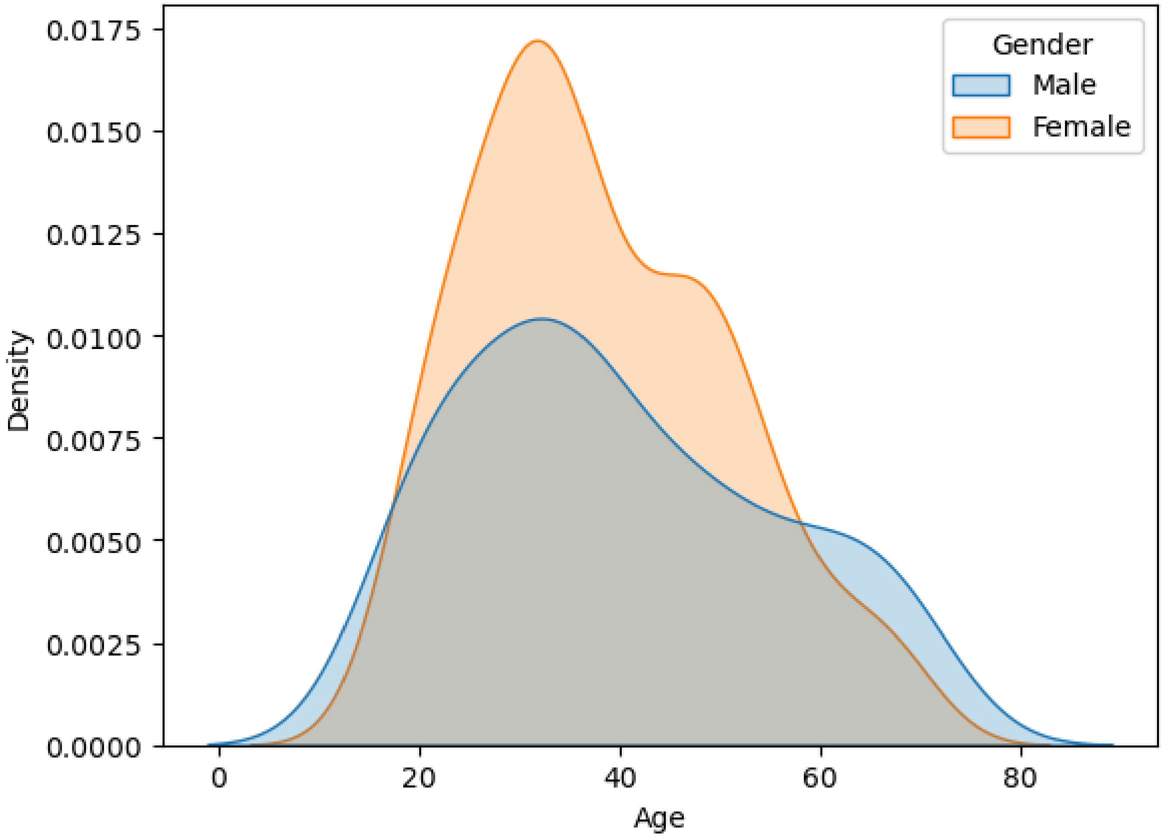
In [24]:
```
sns.kdeplot(data["Spending Score (1-100)"] , hue = data["Gender"] , shade = True)

py.show()
```

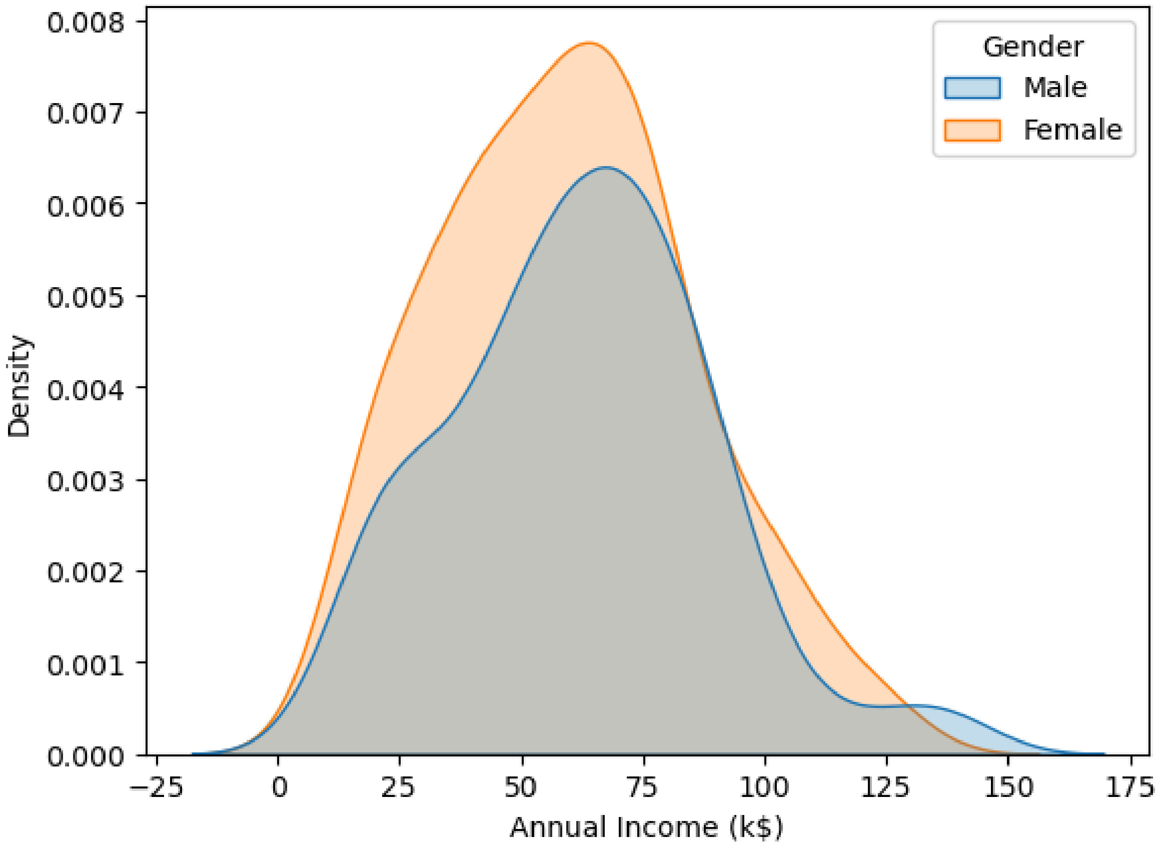


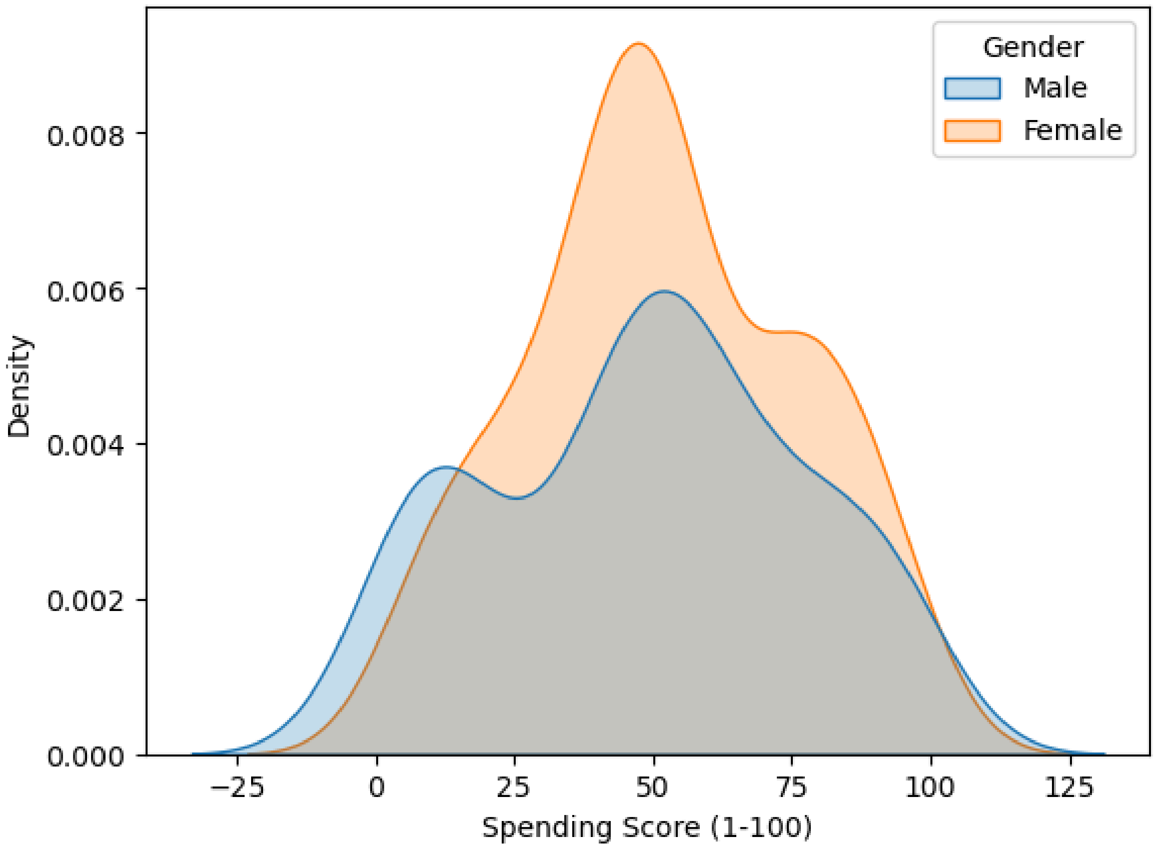
In [ ]:
```
###
```

In [25]:
```
columns = ["Age" , "Annual Income (k$)" , "Spending Score (1-100)"]
```

```python
for i in columns:
    sns.kdeplot(data[i] , hue = data["Gender"] , shade = True)
    py.show()
```

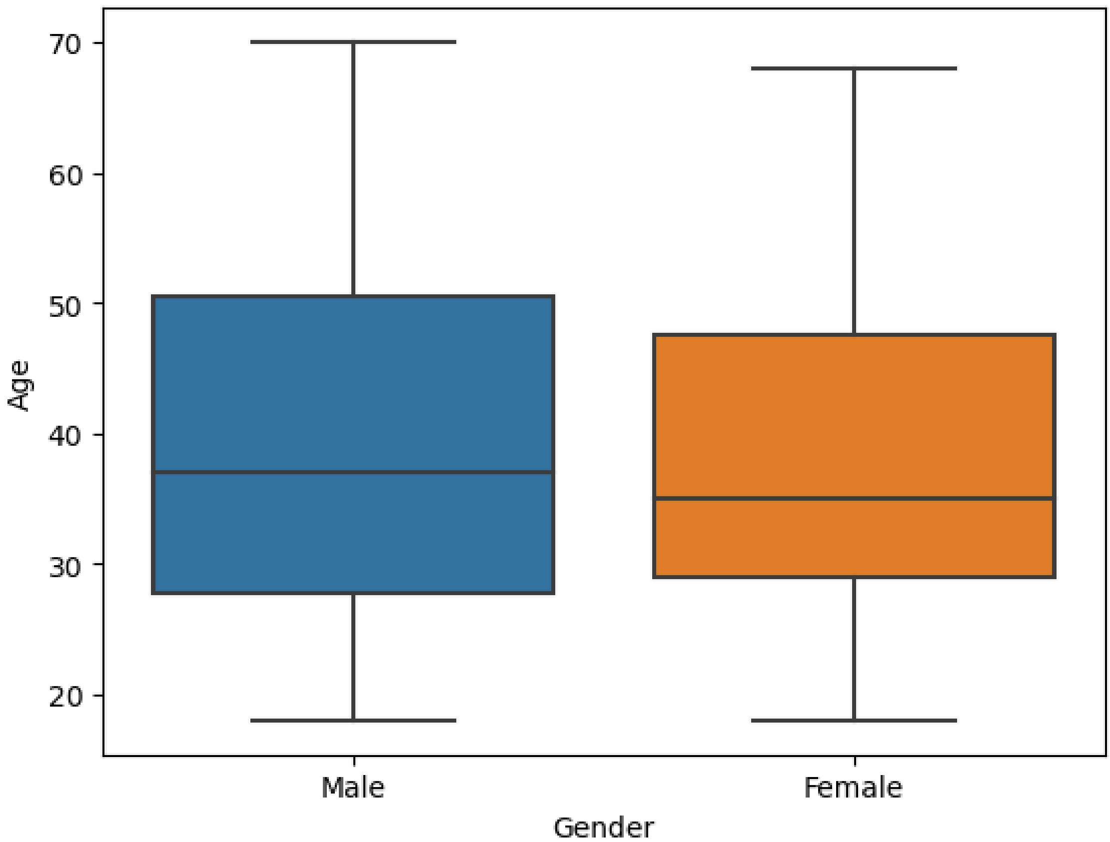


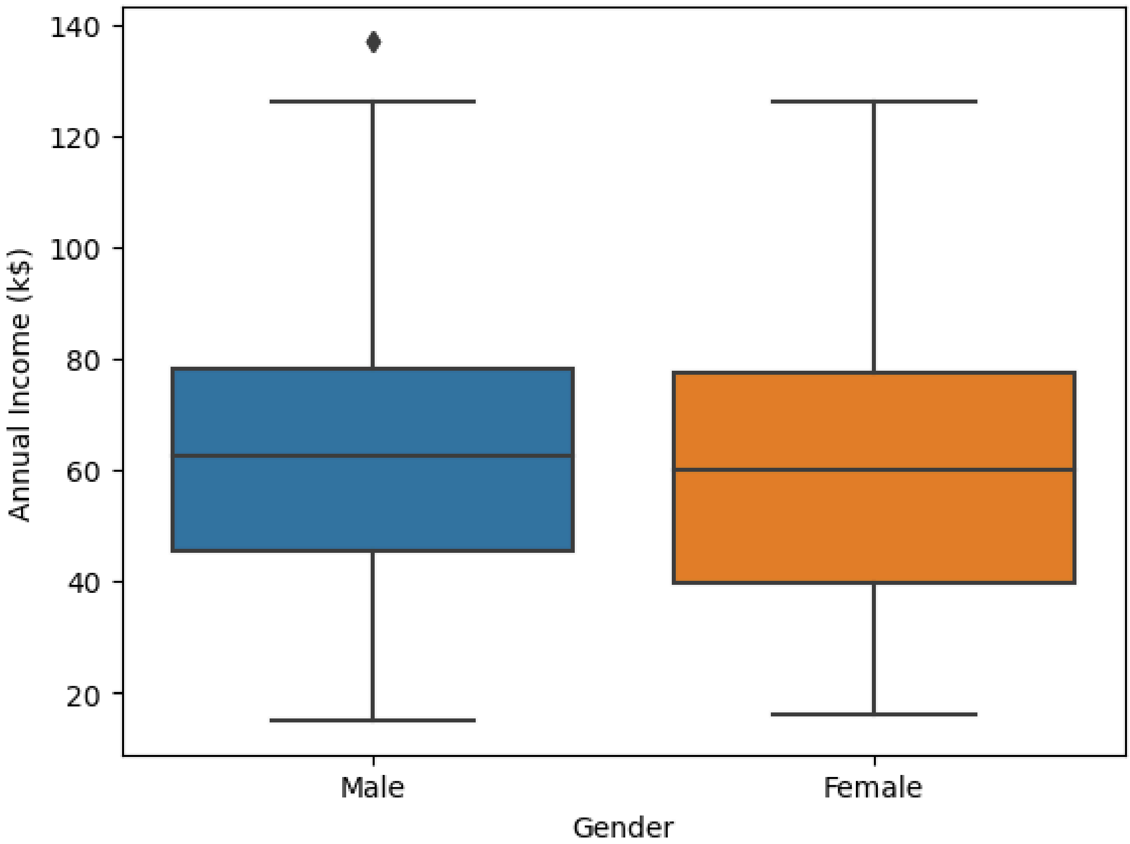


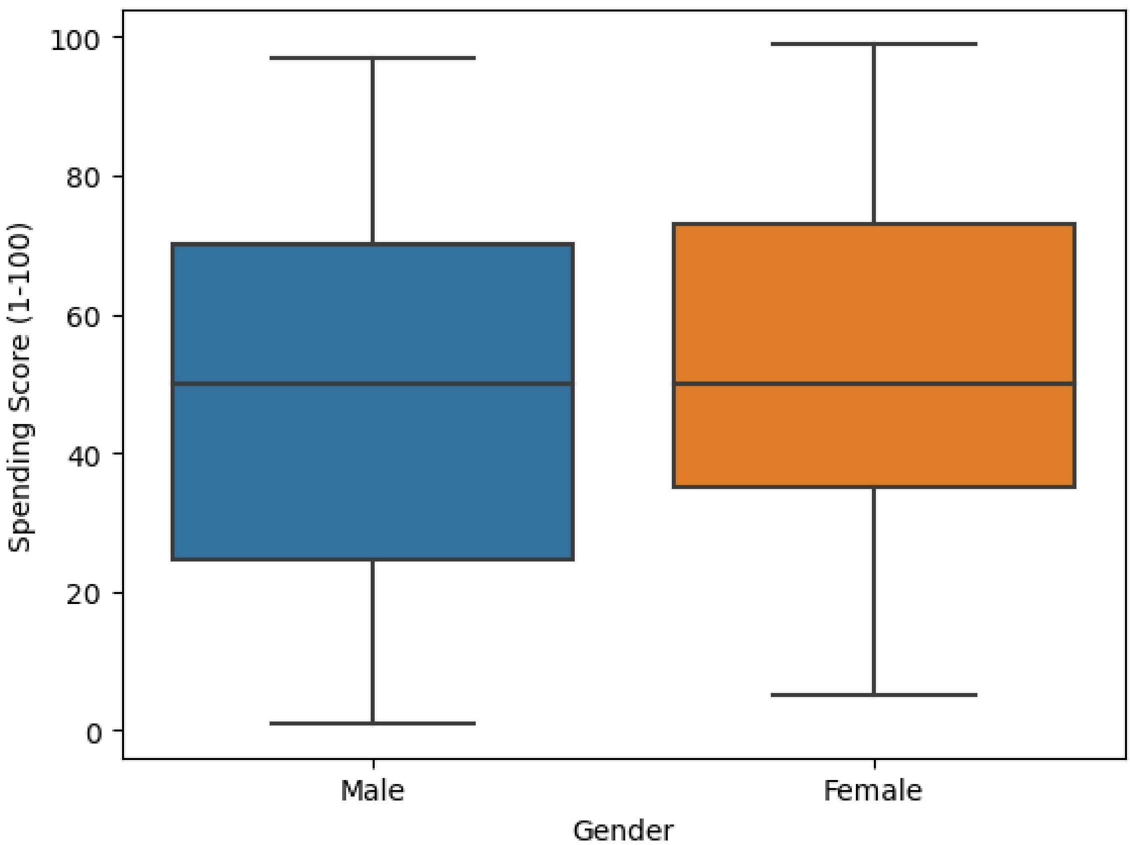


Check the outliers :

In [26]:
```
columns = ["Age" , "Annual Income (k$)" , "Spending Score (1-100)"]

for i in columns:
    sns.boxplot(data = data , x = "Gender" , y = i)
    py.show()
```







Gender Percentage :

In [27]:
```
data["Gender"].value_counts(normalize = True) * 100
```

Out[27]:
```
Female    56.0
Male      44.0
Name: Gender, dtype: float64
```

Conculsion :

- Female percentage more than Male. Also, the Spending Score and Annual Income is highest for Females.
- Females are spending more

Bivariate Analysis :

In [28]:
```
data.head()
```

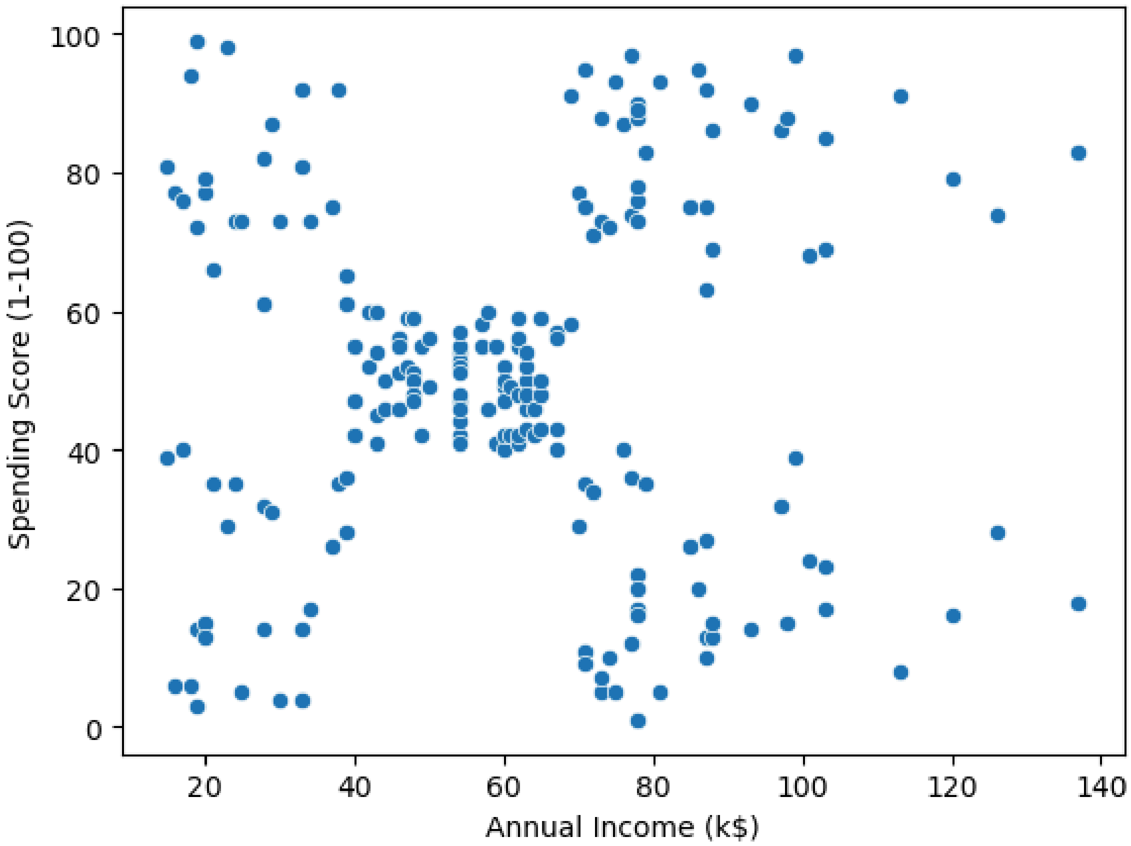
Out[28]:

	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

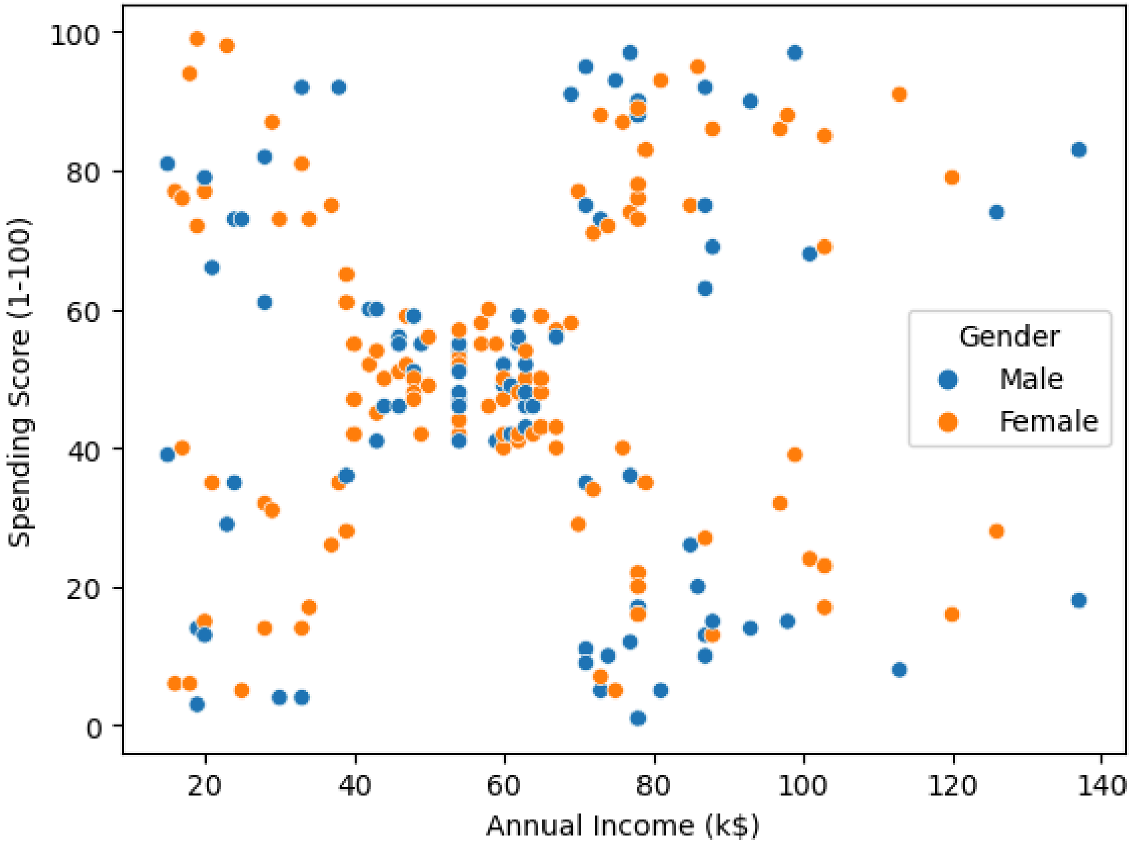
In [29]:
```
data.columns
```

Out[29]:
```
Index(['CustomerID', 'Gender', 'Age', 'Annual Income (k$)',
       'Spending Score (1-100)'],
      dtype='object')
```

In [30]:
```
sns.scatterplot(data = data , x = "Annual Income (k$)" , y = "Spending Score (1-10

py.show()
```



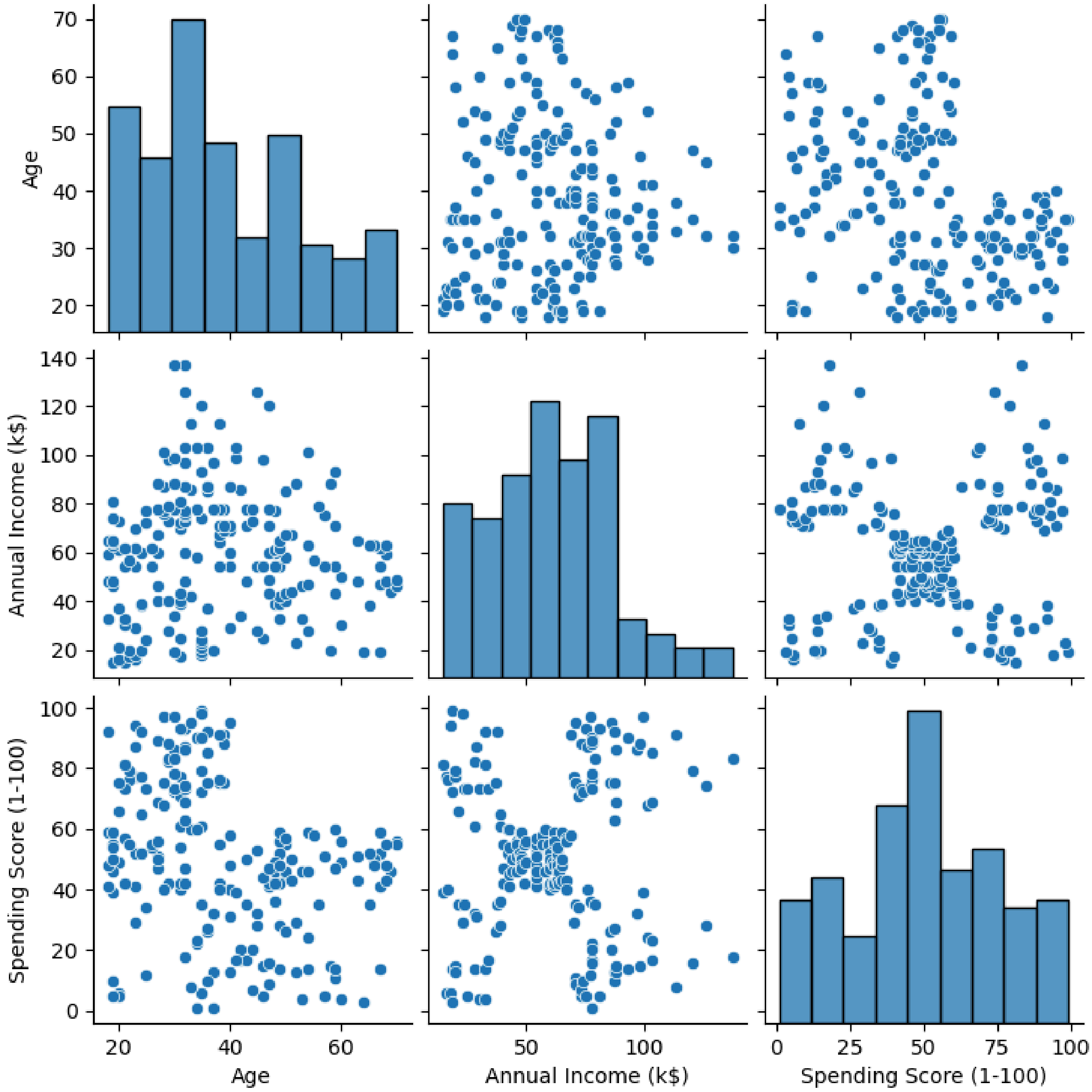
In [31]:
```
sns.scatterplot(data = data , x = "Annual Income (k$)" , y = "Spending Score (1-10

py.show()
```



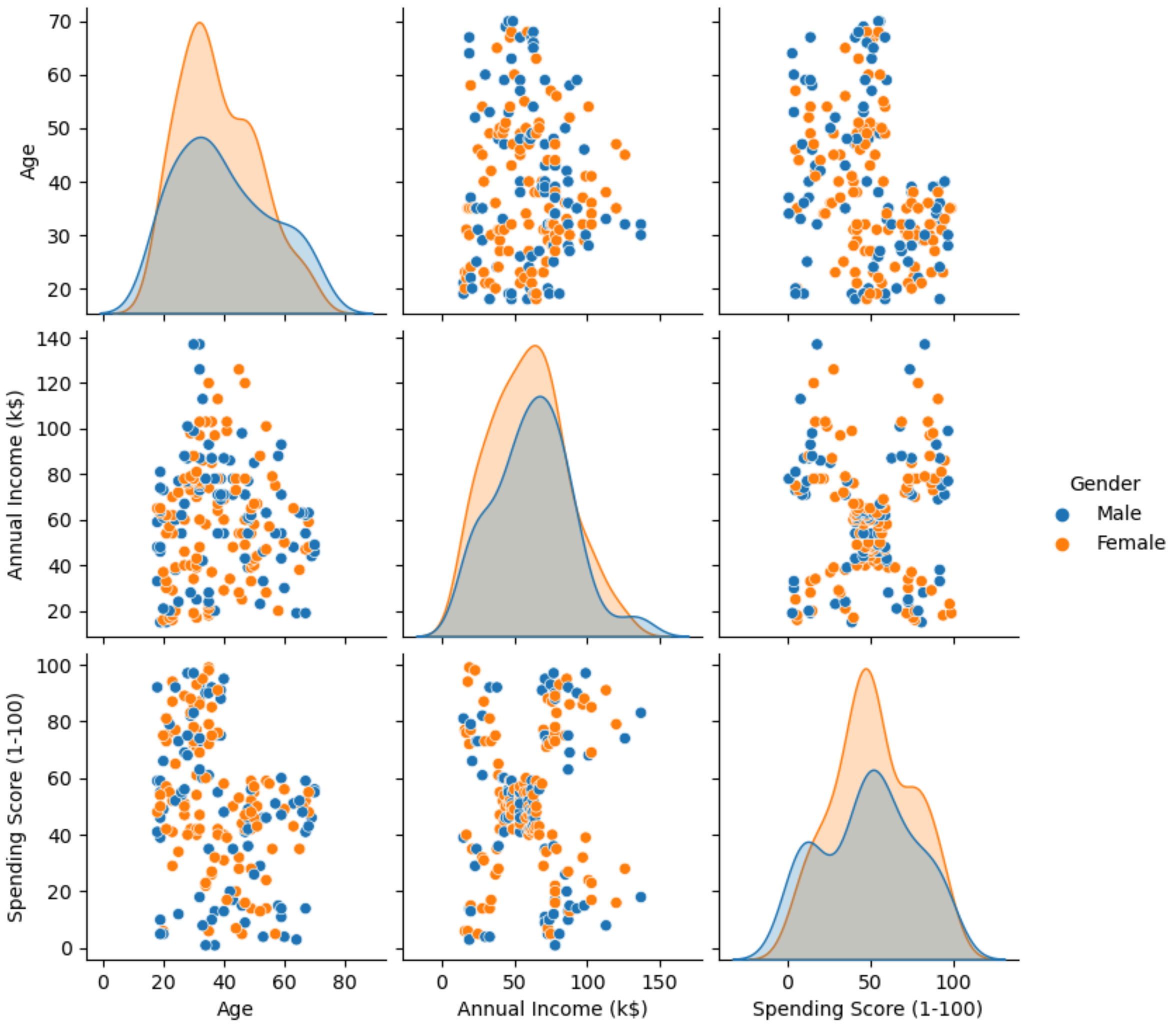
Conclusion :

Between 40 to 70 Annual Income and Spending Score is more

In [32]:
```
df = data.drop("CustomerID" , axis = 1)
sns.pairplot(df)

py.show()
```



In [33]:
```
df1 = data.drop("CustomerID" , axis = 1)
sns.pairplot(df1 , hue = "Gender")

py.show()
```



In [34]:
```
data.columns
```

Out[34]:
```
Index(['CustomerID', 'Gender', 'Age', 'Annual Income (k$)',
       'Spending Score (1-100)'],
      dtype='object')
```

The mean of Age, Income and Spending Score Gender-wise :

In [35]:
```
data.groupby("Gender")[['Age', 'Annual Income (k$)',
       'Spending Score (1-100)']].mean()

## Female are spending more
```

Out[35]:

	Age	Annual Income (k$)	Spending Score (1-100)
Gender			
Female	38.098214	59.250000	51.526786
Male	39.806818	62.227273	48.511364

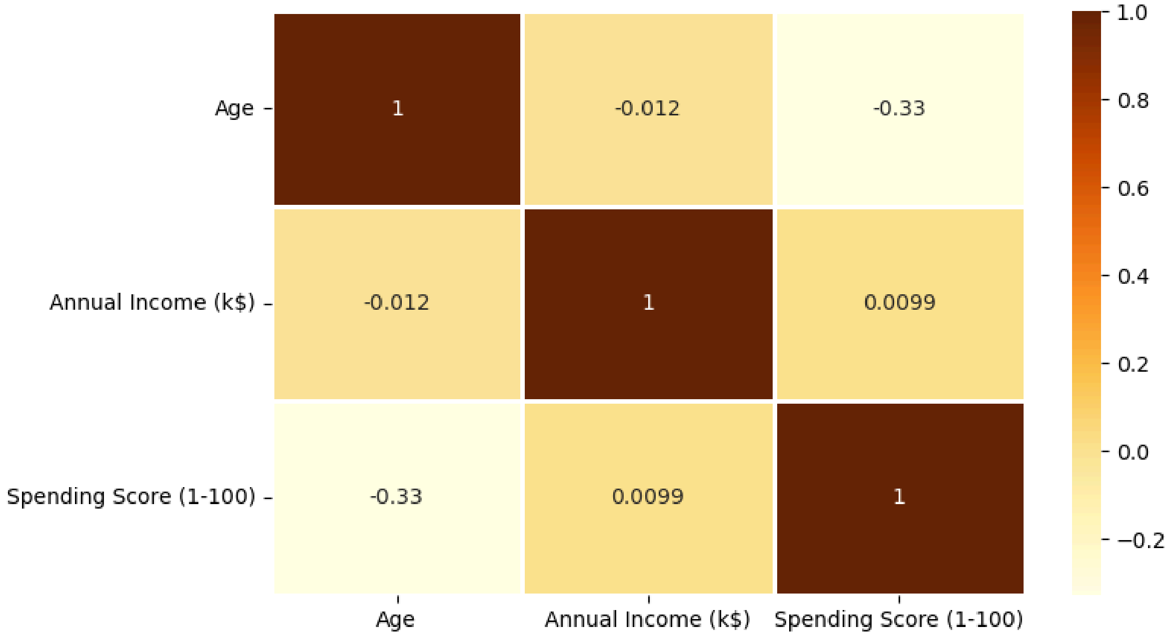
Correlation :

In [36]:
```
data.corr()
```

Out[36]:

	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)
CustomerID	1.000000	-0.026763	0.977548	0.013835
Age	-0.026763	1.000000	-0.012398	-0.327227
Annual Income (k$)	0.977548	-0.012398	1.000000	0.009903
Spending Score (1-100)	0.013835	-0.327227	0.009903	1.000000

In [37]:
```
py.figure(figsize = (8,5))
sns.heatmap(df.corr() , annot = True , cmap = "YlOrBr" , linewidth = 1 , linecolor
py.xticks(rotation = "0")

py.show()
```



In [ ]: