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

In [ ]:

```
df=pd.read_csv(r"/content/Mall_Customers (1) - Copy.csv")
df
```

Out[ ]:

	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 [ ]:

```
df.info()
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 [ ]:

```
df.describe()
```

Out[ ]:

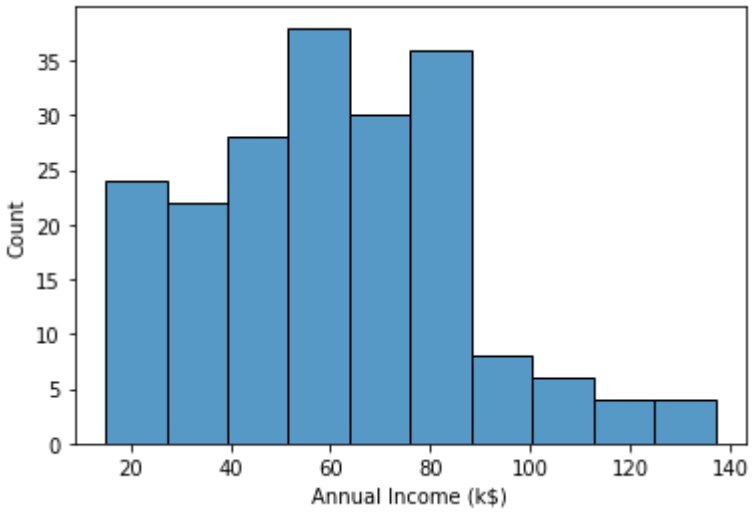
	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)
count	200.000000	200.000000	200.000000	200.000000
mean	100.500000	38.850000	60.560000	50.200000
std	57.879185	13.969007	26.264721	25.823522
min	1.000000	18.000000	15.000000	1.000000
25%	50.750000	28.750000	41.500000	34.750000
50%	100.500000	36.000000	61.500000	50.000000
75%	150.250000	49.000000	78.000000	73.000000
max	200.000000	70.000000	137.000000	99.000000

3.Perform the Visualization Univariate Analysis

In [ ]:

```
sns.histplot(df['Annual Income (k$)'])
```

Out[ ]:



In [ ]:

```
df.shape
```

Out[ ]:

```
(200, 5)
```

In [ ]:

```
from google.colab import drivedrive.mount('/content/drive')
```

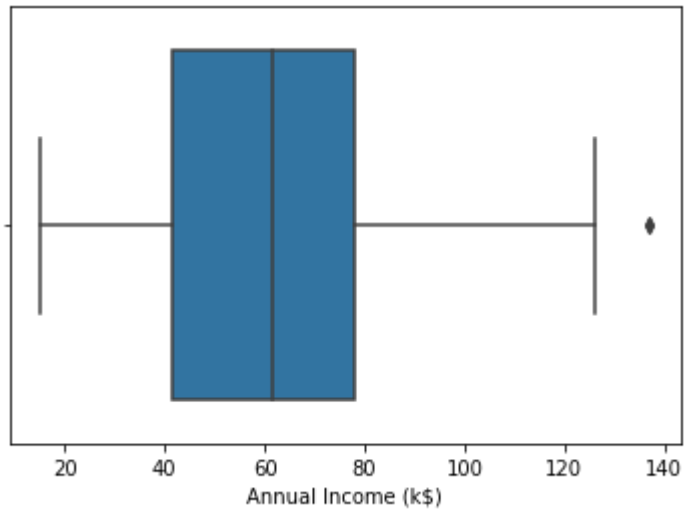
In [ ]:

```
sns.boxplot(df['Annual Income (k$)'])
```

/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

FutureWarning

Out[ ]:



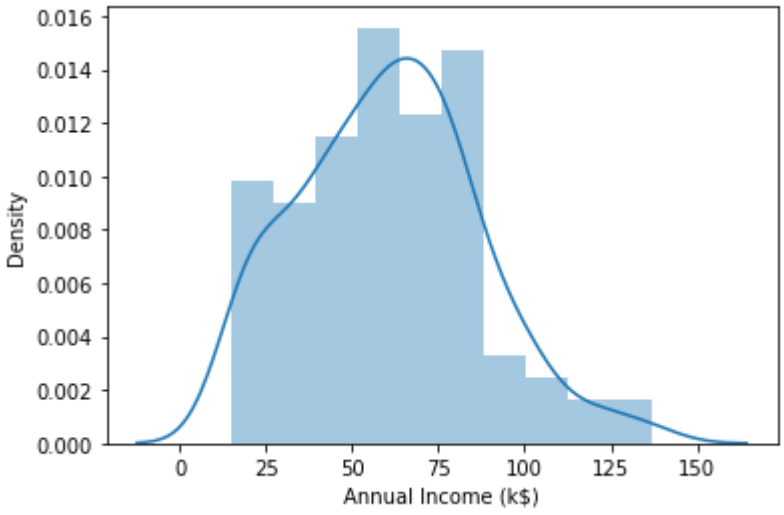
In [ ]:

```
sns.distplot(df['Annual Income (k$)'])
```

/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

warnings.warn(msg, FutureWarning)

Out[ ]:



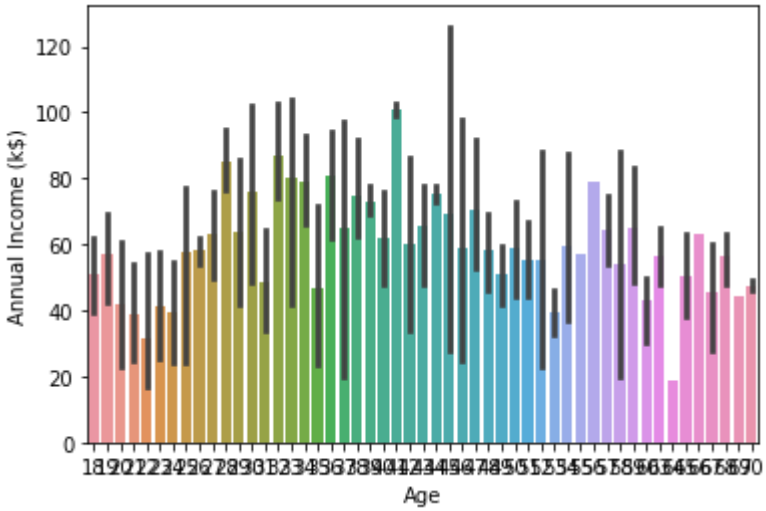
Bi-Variate Analysis

In [ ]:

```
sns.barplot(df['Age'],df['Annual Income (k$)'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[ ]:

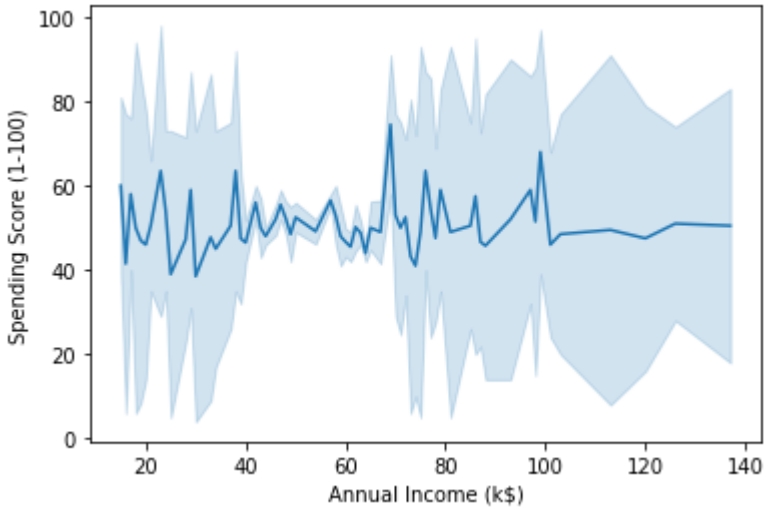


In [ ]:

```
sns.lineplot(df['Annual Income (k$)'], df['Spending Score (1-100)'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[ ]:

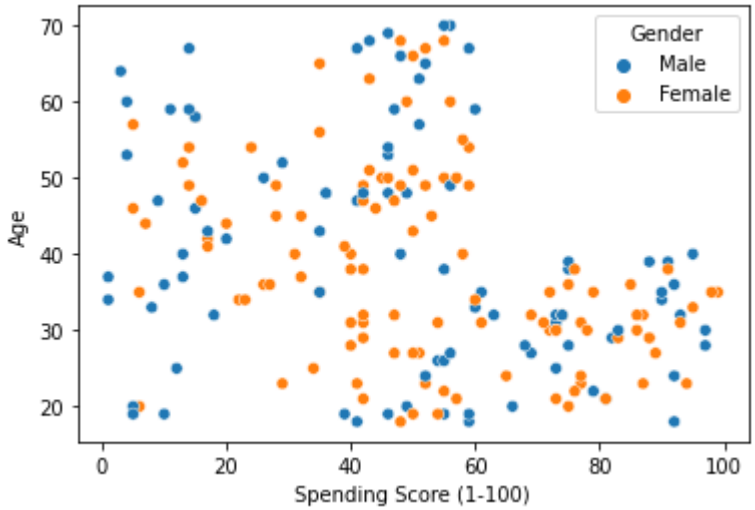


In [ ]:

```
sns.scatterplot(df['Spending Score (1-100)'],df['Age'],hue
=df['Gender'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: Fut
ureWarning: Pass the following variables as keyword args: x, y. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
 error or misinterpretation.
  FutureWarning
```

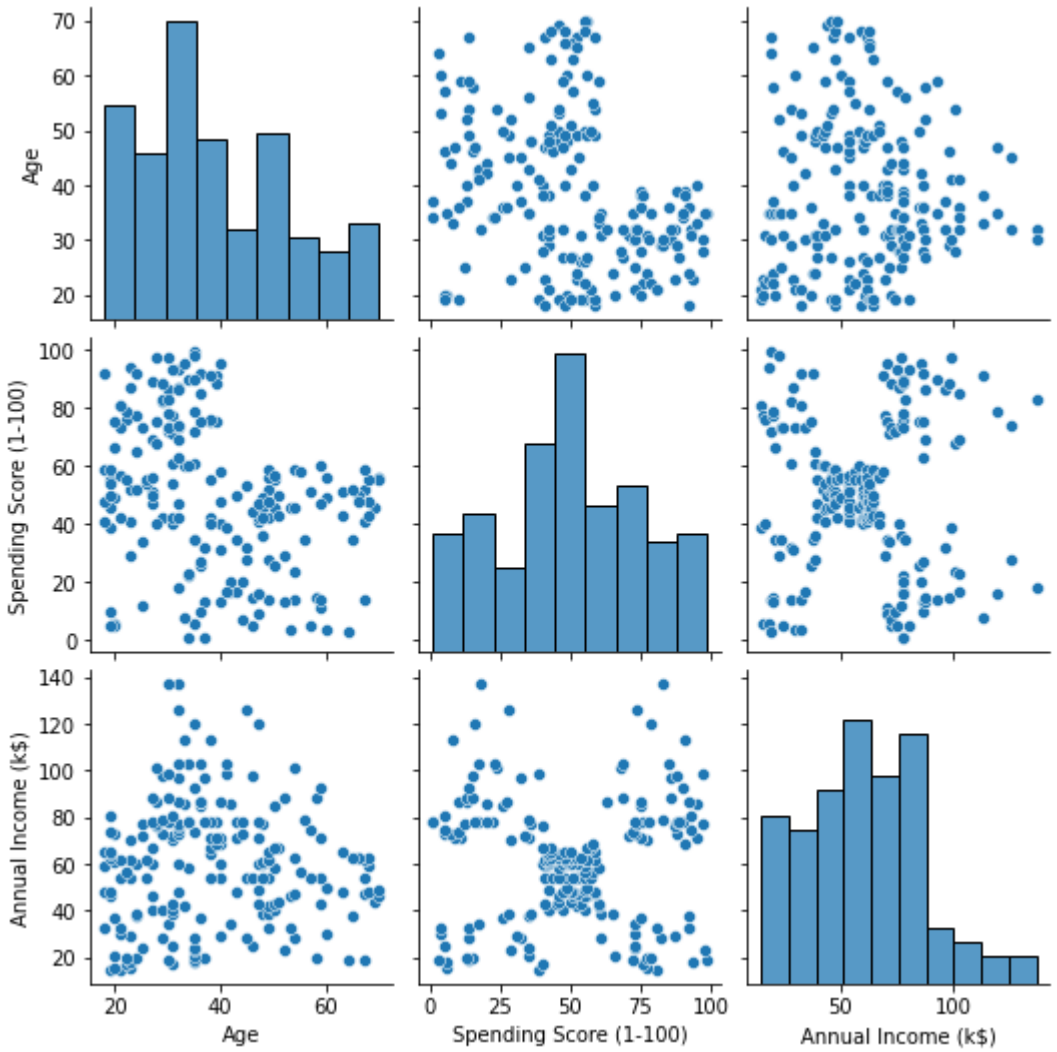
Out[ ]:



In [ ]:

```
sns.pairplot(data=df[["Age", "Gender","Spending Score (1-
100)","Annual Income (k$)"]])
```

Out[ ]:



In [ ]:

```
sns.heatmap(df.corr(),annot=True)
```

Out[ ]:

4. Perform descriptive statistics on the dataset

In [ ]:

```
df.describe()
```

Out[ ]:

	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 [ ]:

```
df.drop('CustomerID',axis=1,inplace=True)df.head()
```

Out[ ]:

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

	Gender	Age	Annual Income (k$)	Spending Score (1-100)
3	Female	23	16	77
4	Female	31	17	40

1. *Check for the missing values and deal with them*

In [ ]:

```
df.isnull().any()
```

Out[ ]:

```
Gender                    False
Age                       False
Annual Income (k$)        False
Spending Score (1-100)    False
dtype: bool
```

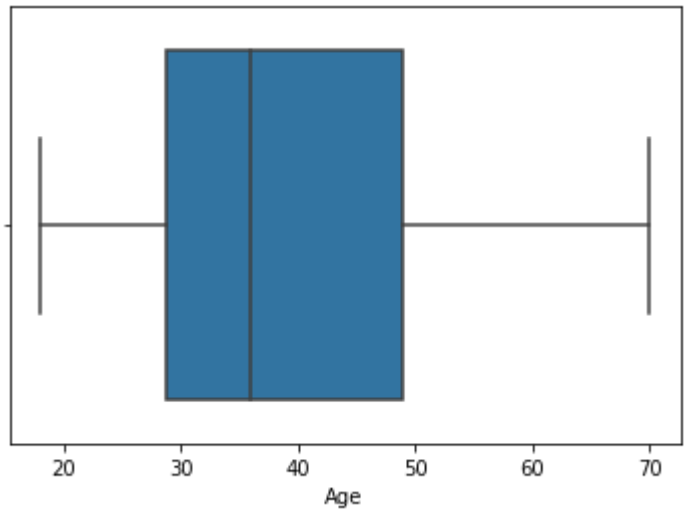
1. *Find the outliers and replace the outliers*

In [ ]:

```
sns.boxplot(df['Age'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[ ]:



1. *Check for categorical columns and perform encoding*

In [ ]:

```
from sklearn.preprocessing import LabelEncoderl_en = LabelEncoder()
```

In [ ]:

```
df['Gender'] = l_en.fit_transform(df['Gender'])df.head()
```

Out[ ]:

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

1. *Scaling the data*

In [ ]:

```
from sklearn.preprocessing import MinMaxScalerscaler =
MinMaxScaler()data_scaled = scaler.fit_transform(df)data_scaled[0:5]
```

Out[ ]:

```
array([[1.        , 0.01923077, 0.        , 0.3877551 ],
       [1.        , 0.05769231, 0.        , 0.81632653],
       [0.        , 0.03846154, 0.00819672, 0.05102041],
       [0.        , 0.09615385, 0.00819672, 0.7755102 ],
       [0.        , 0.25      , 0.01639344, 0.39795918]])
```

1. *Perform any of the clustering algorithms*

In [ ]:

```
from sklearn.cluster import KMeanskm = KMeans()res =
km.fit_predict(data_scaled)res
```

Out[ ]:

```
array([3, 3, 2, 2, 2, 2, 5, 2, 1, 2, 1, 2, 5, 2, 7, 3, 2, 3, 1, 2, 3,
 3,
       5, 3, 5, 3, 5, 3, 5, 2, 1, 2, 1, 3, 5, 2, 5, 2, 5, 2, 5, 3, 1,
 2,
       5, 2, 5, 2, 2, 2, 5, 3, 2, 1, 5, 1, 5, 1, 2, 1, 1, 3, 5, 5, 1,
 3,
       5, 5, 3, 2, 1, 5, 5, 5, 1, 3, 5, 3, 2, 5, 1, 3, 1, 5, 2, 1, 5,
 2,
       2, 5, 5, 3, 1, 5, 2, 3, 5, 2, 1, 3, 2, 5, 1, 3, 1, 2, 5, 1, 1,
 1,
       1, 2, 5, 3, 2, 2, 5, 5, 5, 5, 3, 0, 4, 6, 2, 4, 7, 6, 1, 6, 7,
 6,
       2, 4, 7, 4, 0, 6, 7, 4, 0, 6, 2, 4, 7, 6, 1, 4, 0, 6, 7, 6, 0,
 4,
       0, 4, 7, 4, 7, 4, 5, 4, 7, 4, 7, 4, 7, 4, 0, 6, 7, 6, 7, 6, 0,
 4,
```

7, 6, 7, 6, 0, 4, 7, 4, 0, 6, 0, 6, 0, 4, 0, 4, 7, 4, 0, 4, 0, 6,
7, 6], dtype=int32)

In [ ]:

```
data1 = pd.DataFrame(data_scaled, columns = df.columns)data1.head()
```

Out[ ]:

	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1.0	0.019231	0.000000	0.387755
1	1.0	0.057692	0.000000	0.816327
2	0.0	0.038462	0.008197	0.051020
3	0.0	0.096154	0.008197	0.775510
4	0.0	0.250000	0.016393	0.397959

In [ ]:

```
data1['kclus'] = pd.Series(res)data1.head()
```

Out[ ]:

	Gender	Age	Annual Income (k$)	Spending Score (1-100)	kclus
0	1.0	0.019231	0.000000	0.387755	3
1	1.0	0.057692	0.000000	0.816327	3
2	0.0	0.038462	0.008197	0.051020	2
3	0.0	0.096154	0.008197	0.775510	2
4	0.0	0.250000	0.016393	0.397959	2

In [ ]:

```
data1['kclus'].unique()
```

Out[ ]:

```
array([3, 2, 5, 1, 7, 0, 4, 6], dtype=int32)
```

In [ ]:

```
data1['kclus'].value_counts()
```

Out[ ]:

```
5    38
2    37
1    27
3    24
4    22
7    19
6    18
0    15
```

Name: kclus, dtype: int64

In [ ]:

```
ind = data1.iloc[:,0:4]ind.head()
```

Out[ ]:

	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1.0	0.019231	0.000000	0.387755
1	1.0	0.057692	0.000000	0.816327
2	0.0	0.038462	0.008197	0.051020
3	0.0	0.096154	0.008197	0.775510
4	0.0	0.250000	0.016393	0.397959

In [ ]:

```
dep = data1.iloc[:,4:]dep.head()
```

Out[ ]:

	kclus
0	3
1	3
2	2
3	2
4	2

Split the data into training and testing

In [ ]:

```
from sklearn.model_selection import
train_test_splitx_train,x_test,y_train,y_test =
train_test_split(ind,dep,test_size=0.3,random_state=1)x_train.head()
```

Out[ ]:

	Gender	Age	Annual Income (k$)	Spending Score (1-100)
116	0.0	0.865385	0.409836	0.428571
67	0.0	0.961538	0.270492	0.479592
78	0.0	0.096154	0.319672	0.520408
42	1.0	0.576923	0.196721	0.357143
17	1.0	0.038462	0.049180	0.663265

In [ ]:

```
x_test.head()
```

Out[ ]:

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

	Gender	Age	Annual Income (k$)	Spending Score (1-100)
58	0.0	0.173077	0.254098	0.510204
40	0.0	0.903846	0.188525	0.346939
34	0.0	0.596154	0.147541	0.132653
102	1.0	0.942308	0.385246	0.591837
184	0.0	0.442308	0.688525	0.387755

In [ ]:

```
y_train.head()
```

In [ ]:

```
y_test.head()
```