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

1. Load the Dataset

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

	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

```
df = df.drop(columns=['CustomerID'])
df.head()
```

	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

```
df.shape
```

```
(200, 4)
```

```
df['Spending Score (1-100)'].value_counts()
```

```
42    8
55    7
46    6
73    6
35    5
     ..
31    1
44    1
53    1
65    1
18    1
Name: Spending Score (1-100), Length: 84, dtype: int64
```

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

```
54     12
78     12
48      6
71      6
63      6
       ..
58      2
59      2
16      2
64      2
137     2
Name: Annual Income (k$), Length: 64, dtype: int64
```

2. Data PreProcessing including Visualizations

Mall_Customers.csv

1 to 10 of 200 entries Filter

CustomerID	Gender	Age	An
1	Male	19	15
2	Male	21	15
3	Female	20	16
4	Female	23	16
5	Female	31	17
6	Female	22	17
7	Female	35	18
8	Female	23	18
9	Male	64	19
10	Female	30	19

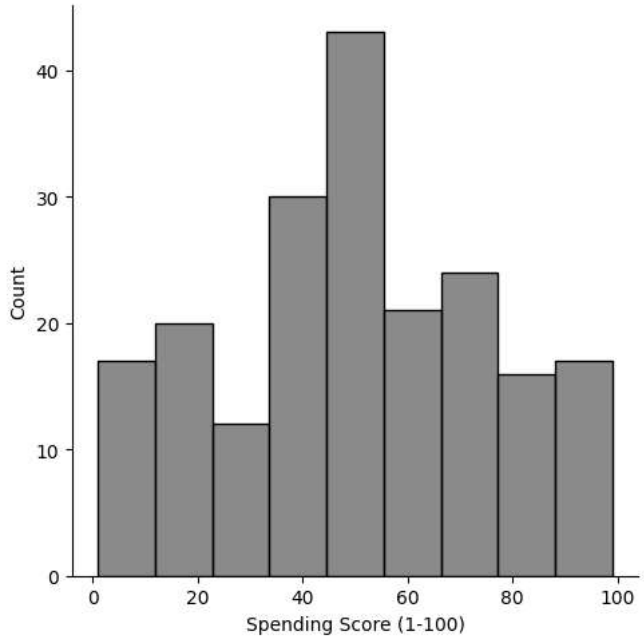
Show 10 per page

1 2 10 20

Univariate

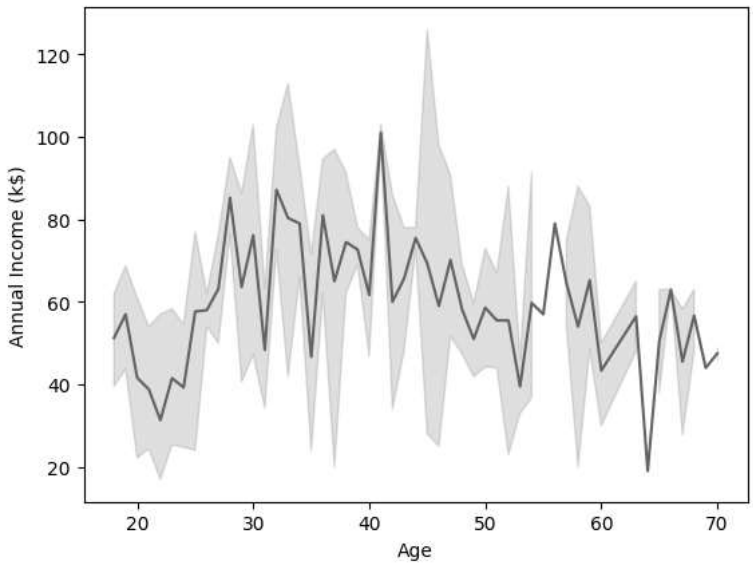
```
sns.displot(df['Spending Score (1-100)'])
```

```
<seaborn.axisgrid.FacetGrid at 0x7fb6e2905e70>
```

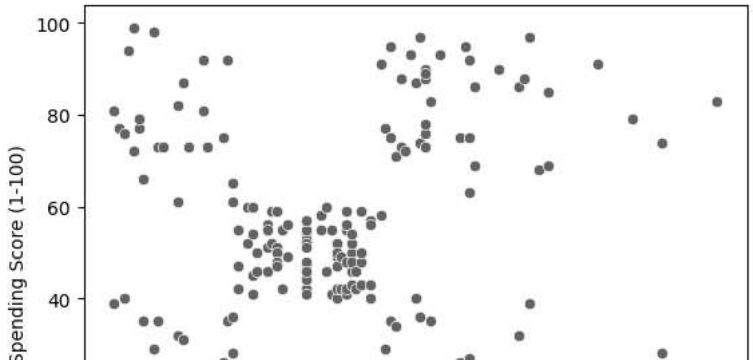


Bivariate

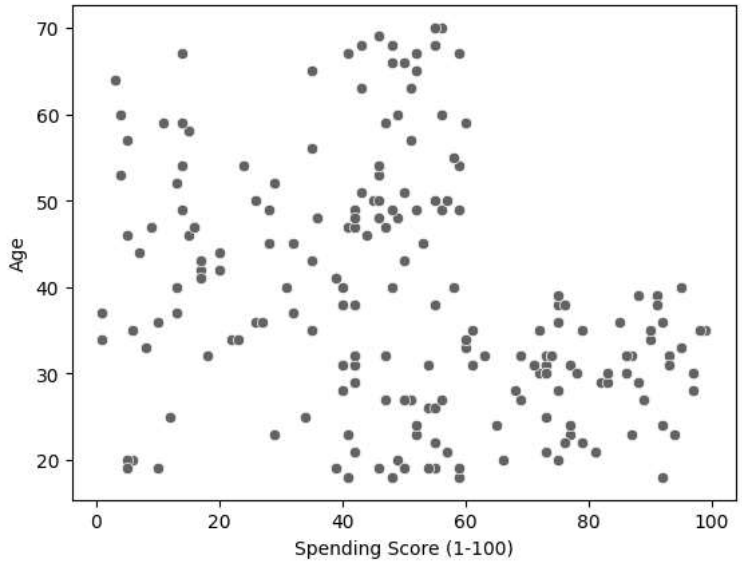
```
sns.lineplot(x='Age', y='Annual Income (k$)', data=df)
plt.show()
```




```
sns.scatterplot(x='Annual Income (k$)', y='Spending Score (1-100)', data=df)
plt.show()
```



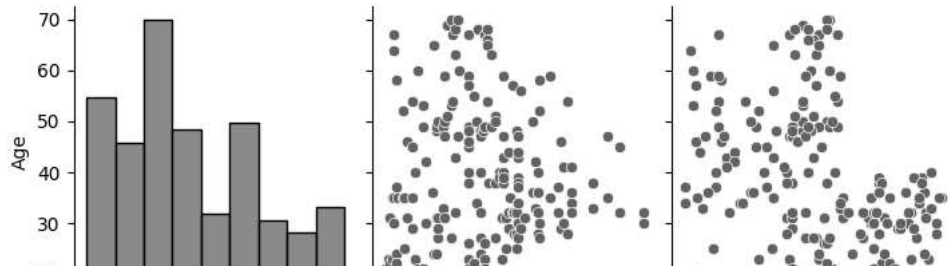

```
sns.scatterplot(x='Spending Score (1-100)',y='Age',data=df)
plt.show()
```



Multivariate

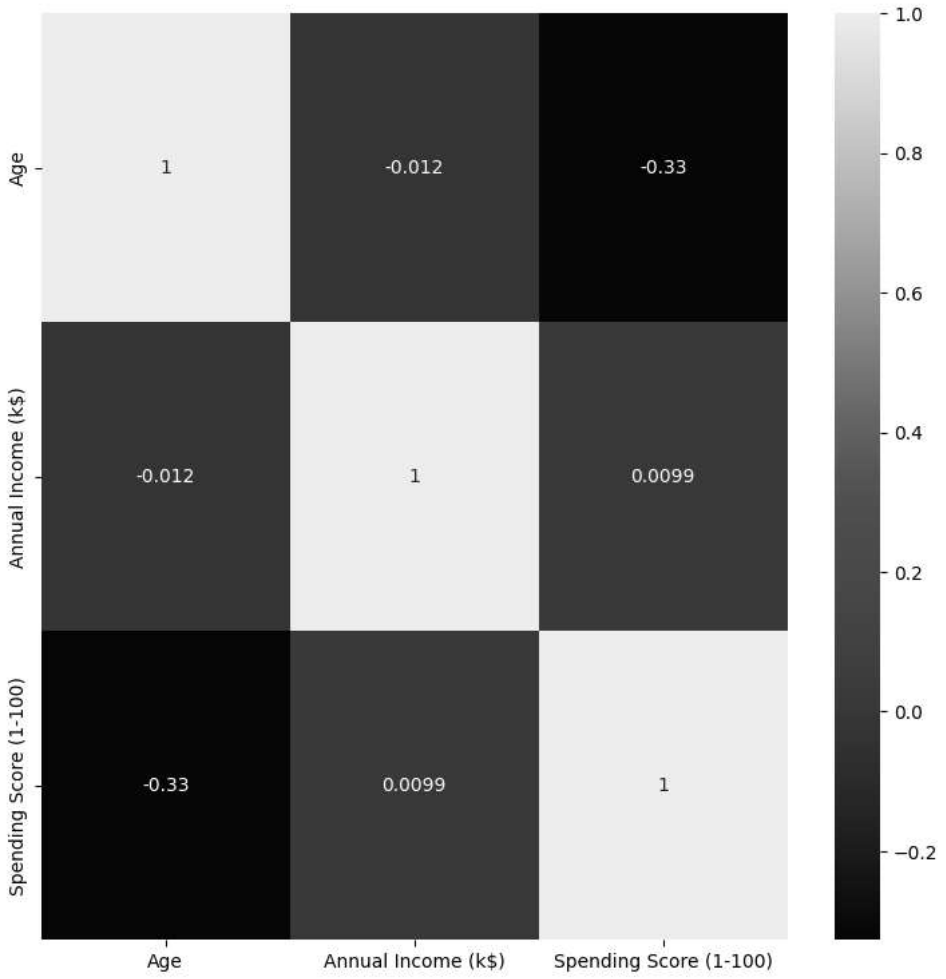
```
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x7fb6e25a9ab0>
```




```
plt.figure(figsize=(9,9))
sns.heatmap(df.corr(), annot=True)
```

```
<ipython-input-17-950e8f39b2a7>:2: FutureWarning: The default value of numeric_only in DataFrame.corr is de
  sns.heatmap(df.corr(), annot=True)
<Axes: >
```

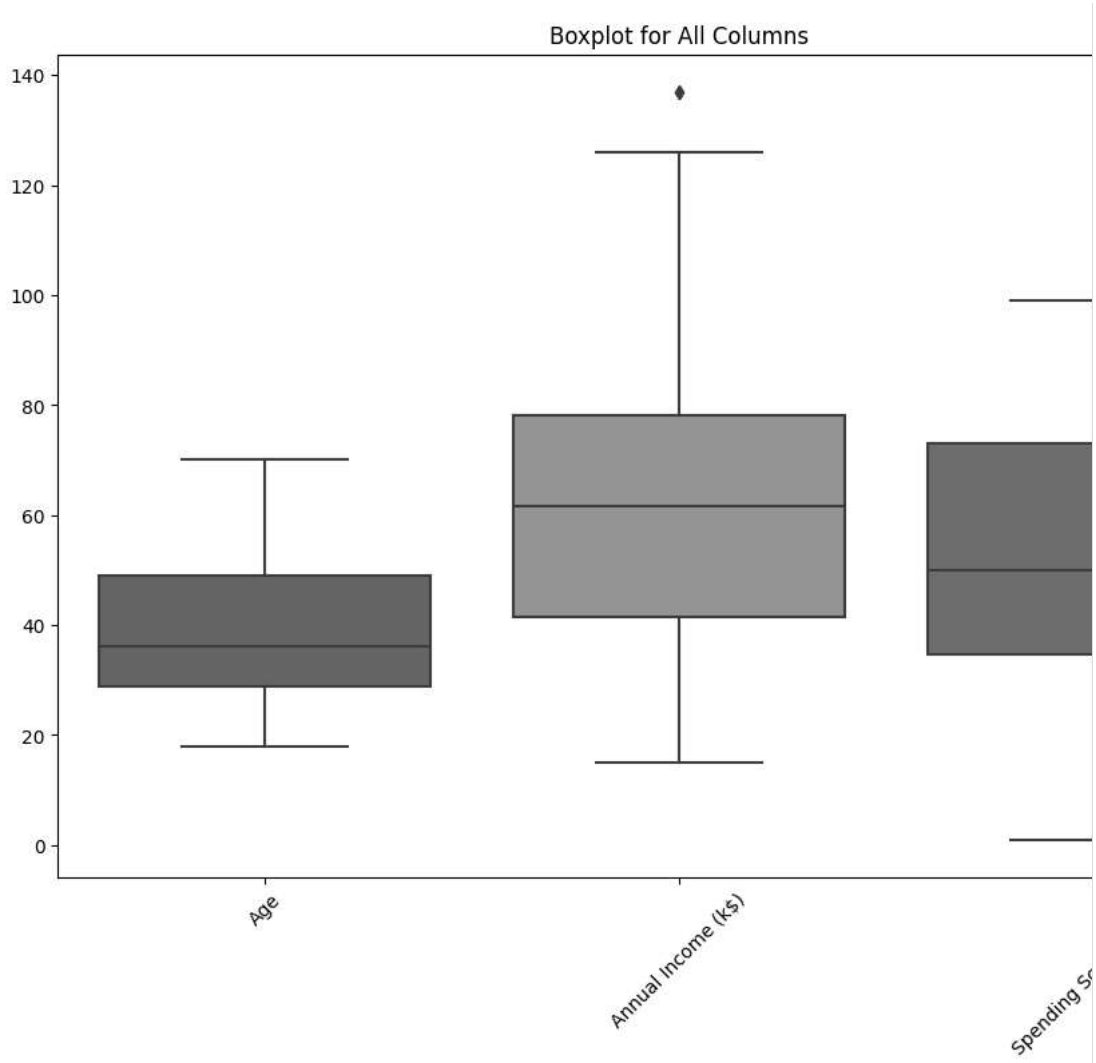


Preprocessing

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

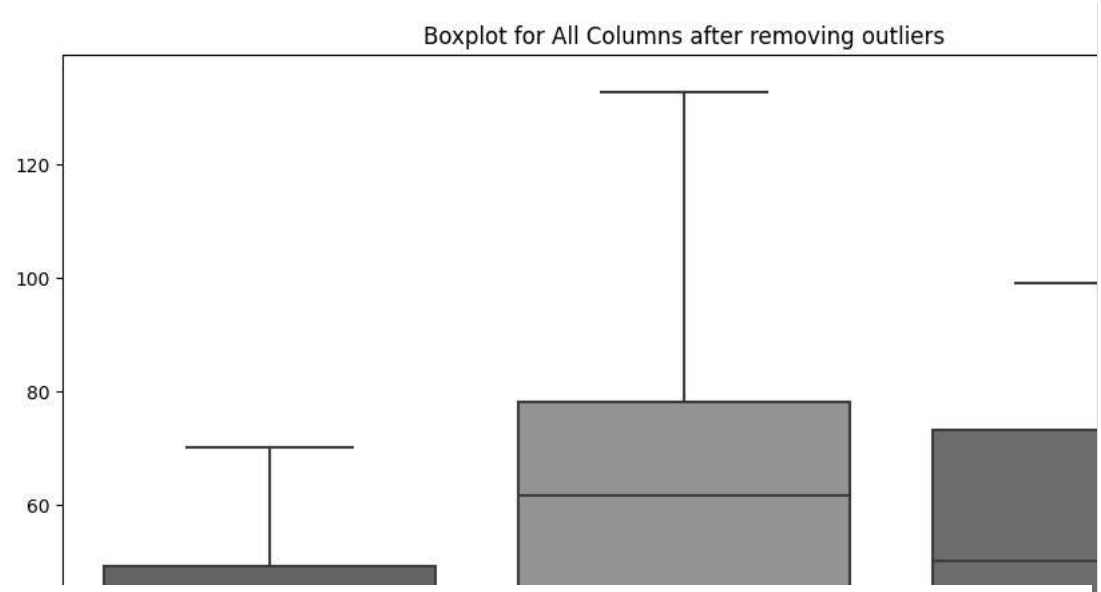
```
Gender                  0
Age                     0
Annual Income (k$)      0
Spending Score (1-100)  0
dtype: int64
```

```
plt.figure(figsize=(12, 8))
sns.boxplot(data=df)
plt.xticks(rotation=45)
plt.title('Boxplot for All Columns')
plt.show()
```




```
q1=df['Annual Income (k$)'].quantile(0.25)
q3=df['Annual Income (k$)'].quantile(0.75)
iqr=q3-q1
upperL=q3+1.5*iqr
lowerL=q1-1.5*iqr
df['Annual Income (k$)']=np.where(df['Annual Income (k$)']>upperL,upperL,np.where(df['Annual Income (k$)']<lower
```

```
plt.figure(figsize=(12, 8))
sns.boxplot(data=df)
plt.xticks(rotation=45)
plt.title('Boxplot for All Columns after removing outliers')
plt.show()
```




```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['Gender'] = le.fit_transform(df['Gender'])
df.head(10)
```

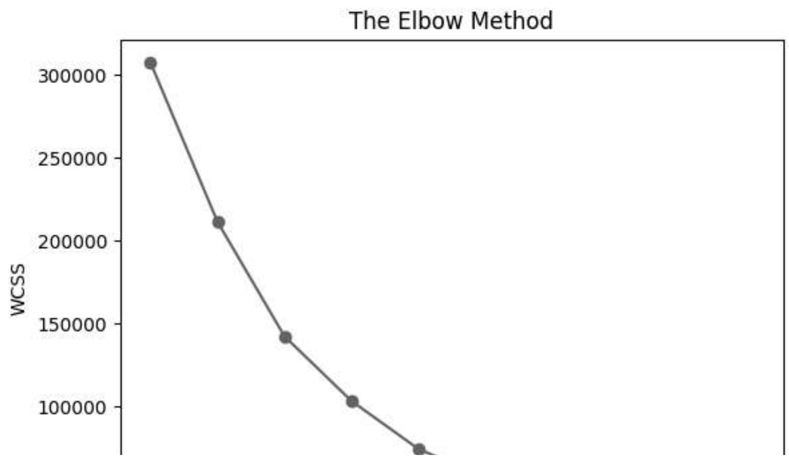
	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	19	15.0	39
1	1	21	15.0	81
2	0	20	16.0	6
3	0	23	16.0	77
4	0	31	17.0	40
5	0	22	17.0	76
6	0	35	18.0	6
7	0	23	18.0	94
8	1	64	19.0	3
9	0	30	19.0	72

3. ML Model Building, optimising

Clustering based by considering all columns -> Gender, Age, Annual Income, Spending Score and creating Clusters(Groups)

```
from sklearn.cluster import KMeans
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans.fit(df)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss, marker="o")
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of
  warnings.warn(
```




```
kmeans = KMeans(n_clusters=5,init = 'k-means++',random_state=0)
pred = kmeans.fit(df)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value o
  warnings.warn(
```

4. Test with random Observation

```
kmeans.predict([[1,23,17,40]])
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature na
  warnings.warn(
array([0], dtype=int32)
```

```
kmeans.predict([[0,25,10,23]])
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature na
  warnings.warn(
array([0], dtype=int32)
```

```
kmeans.predict([[0,20,20,10]])
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature na
  warnings.warn(
array([0], dtype=int32)
```