Business Problem

The market research team at AeroFit wants to identify the characteristics of the target audience for each type of treadmill offered by the company, to provide a better recommendation of the treadmills to the new customers. The team decides to investigate whether there are differences across the product with respect to customer characteristics.

```
!gdown https://drive.google.com/uc?id=1EQk9C7Fdpjknr0-xaTR-dWM1-vIBivu3
```

```
Downloading...
From: https://drive.google.com/uc?id=1EQk9C7Fdpjknr0-xaTR-dWM1-vIBivu3
To: /content/Aerofit.csv
100% 7.46k/7.46k [00:00<00:00, 24.2MB/s]
```

DATA INSPECTION

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

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

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
0	KP281	18	Male	14	Single	3	4	29562	112
1	KP281	19	Male	15	Single	2	3	31836	75
2	KP281	19	Female	14	Partnered	4	3	30699	66
3	KP281	19	Male	12	Single	3	3	32973	85
4	KP281	20	Male	13	Partnered	4	2	35247	47

Next steps: Generate code with df View recommended plots

```
df.tail()
```

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
175	KP781	40	Male	21	Single	6	5	83416	200
176	KP781	42	Male	18	Single	5	4	89641	200
177	KP781	45	Male	16	Single	5	5	90886	160
178	KP781	47	Male	18	Partnered	4	5	104581	120
179	KP781	48	Male	18	Partnered	4	5	95508	180

```
df.shape
```

```
(180, 9)
```

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

```
Product          0
Age              0
Gender           0
Education        0
MaritalStatus    0
Usage            0
Fitness          0
Income           0
Miles            0
dtype: int64
```

```
df.dtypes
```

```
Product          object
Age               int64
Gender           object
Education         int64
MaritalStatus    object
Usage             int64
Fitness           int64
Income            int64
Miles             int64
dtype: object
```

WE HAVE 3 CATEGORICAL DATA WHICH NEEDS TO BE CONVERTED INTO NUMERICAL,SO MACHINE UNDERSTANDS FOR FURTHER ANALYSAIS

```
df.select_dtypes(include='object').columns.tolist()
```

```
['Product', 'Gender', 'MaritalStatus']
```

```
df['Product'].unique()
```

```
array(['KP281', 'KP481', 'KP781'], dtype=object)
```

```
df['Gender'].unique()
```

```
array(['Male', 'Female'], dtype=object)
```

```
Start coding or generate with AI.
```

```
df['MaritalStatus'].unique()
```

```
array(['Single', 'Partnered'], dtype=object)
```

```
df["Gender"].replace({"Female":0,"Male":1},inplace=True)
df["MaritalStatus"].replace({"Single":0,"Partnered":1},inplace=True)
```

```
df.dtypes
```

```
Product          object
Age               int64
Gender            int64
Education         int64
MaritalStatus     int64
Usage             int64
Fitness           int64
Income            int64
Miles             int64
dtype: object
```

```
df
```

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
0	KP281	18	1	14	0	3	4	29562	112
1	KP281	19	1	15	0	2	3	31836	75
2	KP281	19	0	14	1	4	3	30699	66
3	KP281	19	1	12	0	3	3	32973	85
4	KP281	20	1	13	1	4	2	35247	47
...	...	...	...	...	...	...	...	...	...
175	KP781	40	1	21	0	6	5	83416	200
176	KP781	42	1	18	0	5	4	89641	200
177	KP781	45	1	16	0	5	5	90886	160
178	KP781	47	1	18	1	4	5	104581	120
179	KP781	48	1	18	1	4	5	95508	180

180 rows × 9 columns

Next steps: Generate code with df View recommended plots

```
gender_counts = df['Gender'].value_counts()
```

```
gender_counts
```

```
Gender
1    104
0     76
Name: count, dtype: int64
```

```
marital_status = df['MaritalStatus'].value_counts()
```

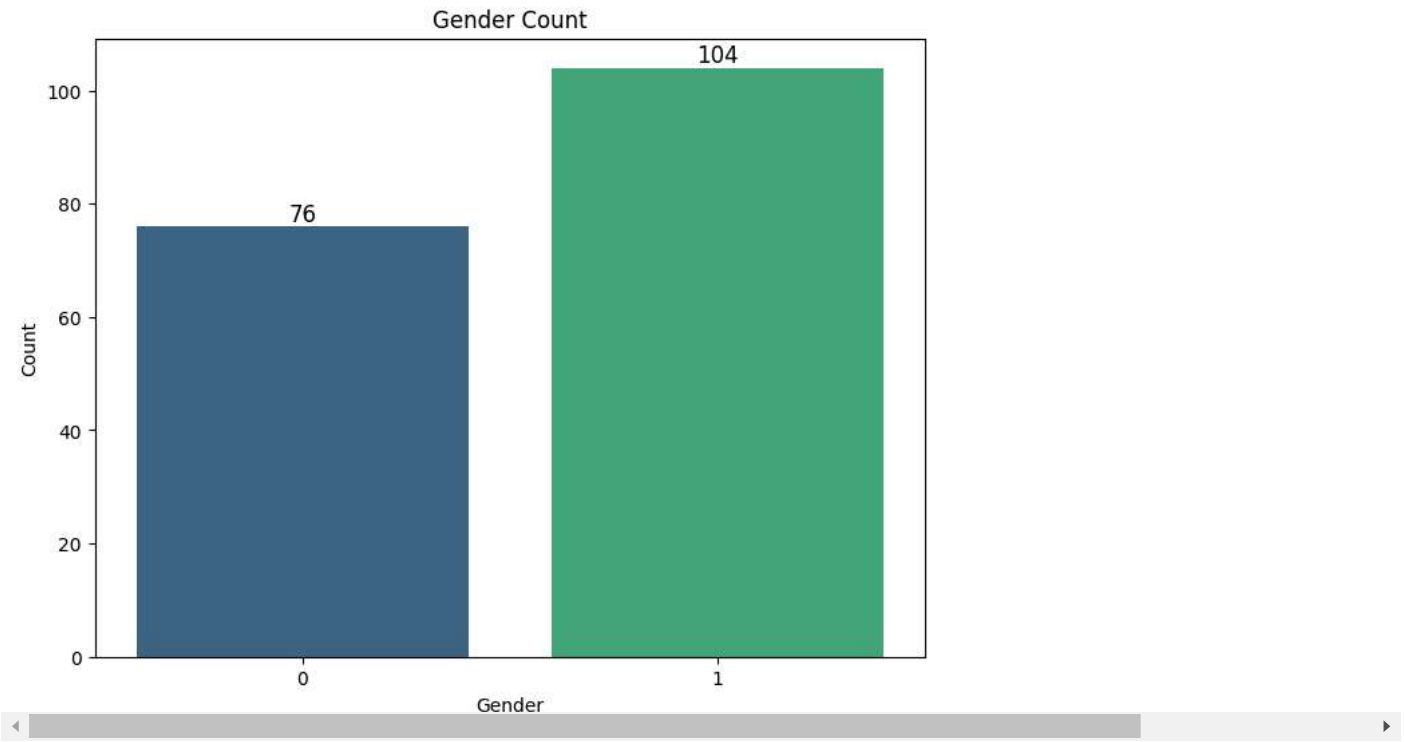
```
marital_status
```

```
MaritalStatus
1    107
0     73
Name: count, dtype: int64
```

Start coding or generate with AI.

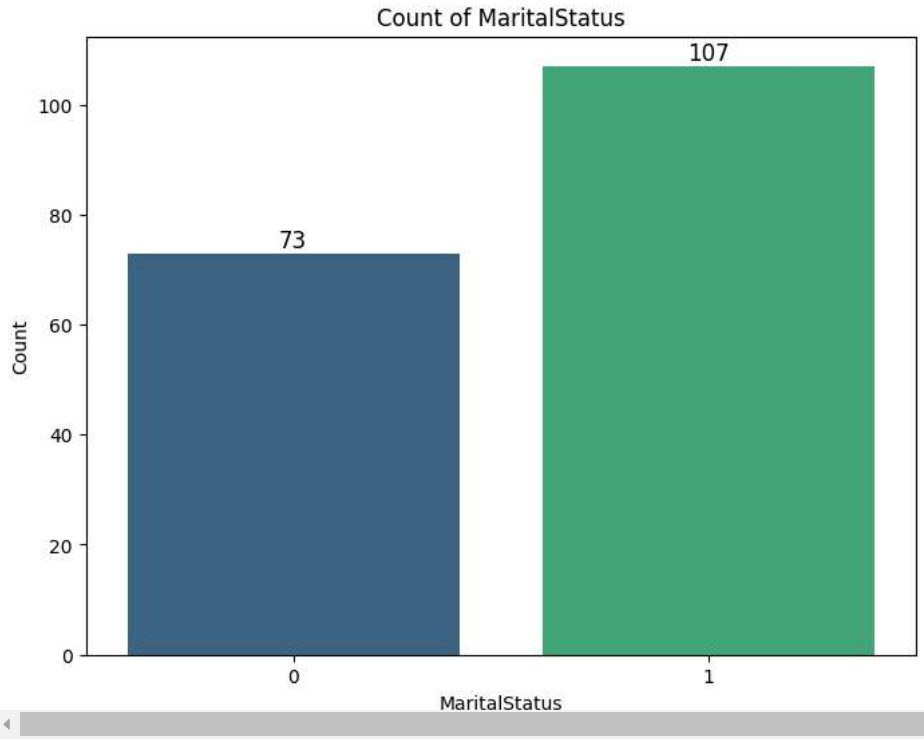
```
# Calculate the counts
gender_counts = df['Gender'].value_counts()
print(gender_counts)

# Create a bar plot
plt.figure(figsize=(8, 6))
ax2 = sns.barplot(x=gender_counts.index, y=gender_counts.values, palette='viridis', errorbar=None)

# Set plot title and labels
ax2.set_title('Gender Count')
ax2.set_xlabel('Gender')
ax2.set_ylabel('Count')

# Annotate the bars with the counts
for container in ax2.containers:
    ax2.bar_label(container, fontsize=12)

plt.show()
```

```
Gender
1    104
0     76
Name: count, dtype: int64
<ipython-input-19-2a9215400040>:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend

  ax2 = sns.barplot(x=gender_counts.index, y=gender_counts.values, palette='viridis', errorbar=None)
```




```
# Calculate the counts
marital_status = df['MaritalStatus'].value_counts()
print(marital_status)

# Create a bar plot
plt.figure(figsize=(8, 6))
ax2 = sns.barplot(x=marital_status.index, y=marital_status.values, palette='viridis', errorbar=None)

# Set plot title and labels
ax2.set_title('Count of MaritalStatus')
ax2.set_xlabel('MaritalStatus')
ax2.set_ylabel('Count')

# Annotate the bars with the counts
for container in ax2.containers:
    ax2.bar_label(container, fontsize=12)

plt.show()
```

```
MaritalStatus
1    107
0     73
Name: count, dtype: int64
<ipython-input-20-becf73cedeea>:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend

  ax2 = sns.barplot(x=marital_status.index, y=marital_status.values, palette='viridis', errorbar=None)
```



STATAISTICS OF THE DATA

```
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Age	180.0	28.788889	6.943498	18.0	24.00	26.0	33.00	50.0
Gender	180.0	0.577778	0.495291	0.0	0.00	1.0	1.00	1.0
Education	180.0	15.572222	1.617055	12.0	14.00	16.0	16.00	21.0
MaritalStatus	180.0	0.594444	0.492369	0.0	0.00	1.0	1.00	1.0
Usage	180.0	3.455556	1.084797	2.0	3.00	3.0	4.00	7.0
Fitness	180.0	3.311111	0.958869	1.0	3.00	3.0	4.00	5.0
Income	180.0	53719.577778	16506.684226	29562.0	44058.75	50596.5	58668.00	104581.0
Miles	180.0	103.194444	51.863605	21.0	66.00	94.0	114.75	360.0

Start coding or generate with AI.

SATISTICAL SUMMARY OF THE DATA IS AS FOLLOWS:

AVERAGE AGE OF POPULATION IS 28-29 AND MAXIMUM AGE IS 50 AND MINIMUM AGE OF THE POPULATION IS 18

AVERAGE INCOME LEVEL OF THE POPULATIO IS 53719.57 AND THE MAXIMUM IS 104581.0 THERFORE PEOPLE >= 50 ARE MAKING THE INCOME OF 104581.0

```
# Value counts for each column
gender_counts = df['Gender'].value_counts()
marital_status_counts = df['MaritalStatus'].value_counts()

# Unique values for each column
unique_genders = df['Gender'].unique()
unique_marital_statuses = df['MaritalStatus'].unique()

# Non-Graphical Analysis Report

# 1. Value Counts
print("Value Counts:")
print("Gender:")
print(gender_counts)
print("\nMarital Status:")
print(marital_status_counts)

# 2. Unique Attributes
print("\nUnique Attributes:")
print("Unique Genders:", unique_genders)
print("Number of Unique Genders:", len(unique_genders))
print("\nUnique Marital Statuses:", unique_marital_statuses)
print("Number of Unique Marital Statuses:", len(unique_marital_statuses))
```

```
Value Counts:
Gender:
Gender
1    104
0     76
Name: count, dtype: int64

Marital Status:
MaritalStatus
1    107
0     73
Name: count, dtype: int64

Unique Attributes:
Unique Genders: [1 0]
Number of Unique Genders: 2

Unique Marital Statuses: [0 1]
Number of Unique Marital Statuses: 2
```

Interpretation:

Gender Counts: The dataset has more 'male' entries (76) than 'Female' entries (104).

Marital Status Counts: There are more 'Parented' individuals (107) compared to 'Single' (73).

Unique Genders:The column 'Gender' contains 2 unique values: 'Male' and 'Female'.

Unique Marital Statuses: The column 'MaritalStatus' contains 2 unique values: 'Parented' and 'Single'.

```
Start coding or generate with AI.
```

UNIVARIATE AND BIVARIATE ANLYSIS

```
df
```

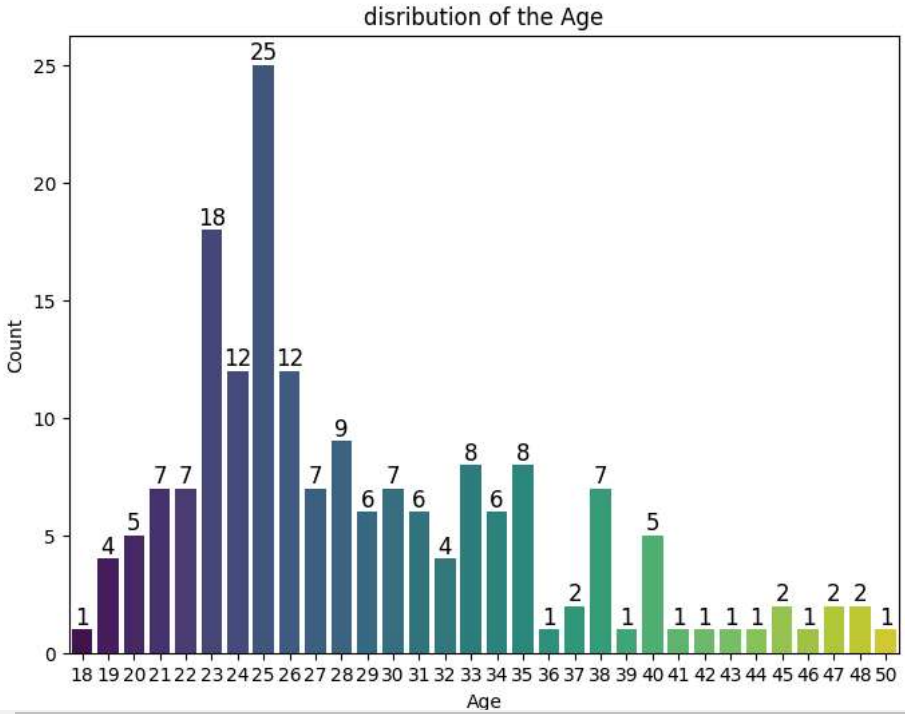
	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
0	KP281	18	1	14	0	3	4	29562	112
1	KP281	19	1	15	0	2	3	31836	75
2	KP281	19	0	14	1	4	3	30699	66
3	KP281	19	1	12	0	3	3	32973	85
4	KP281	20	1	13	1	4	2	35247	47
...	...	...	...	...	...	...	...	...	...
175	KP781	40	1	21	0	6	5	83416	200
176	KP781	42	1	18	0	5	4	89641	200
177	KP781	45	1	16	0	5	5	90886	160
178	KP781	47	1	18	1	4	5	104581	120
179	KP781	48	1	18	1	4	5	95508	180

180 rows × 9 columns

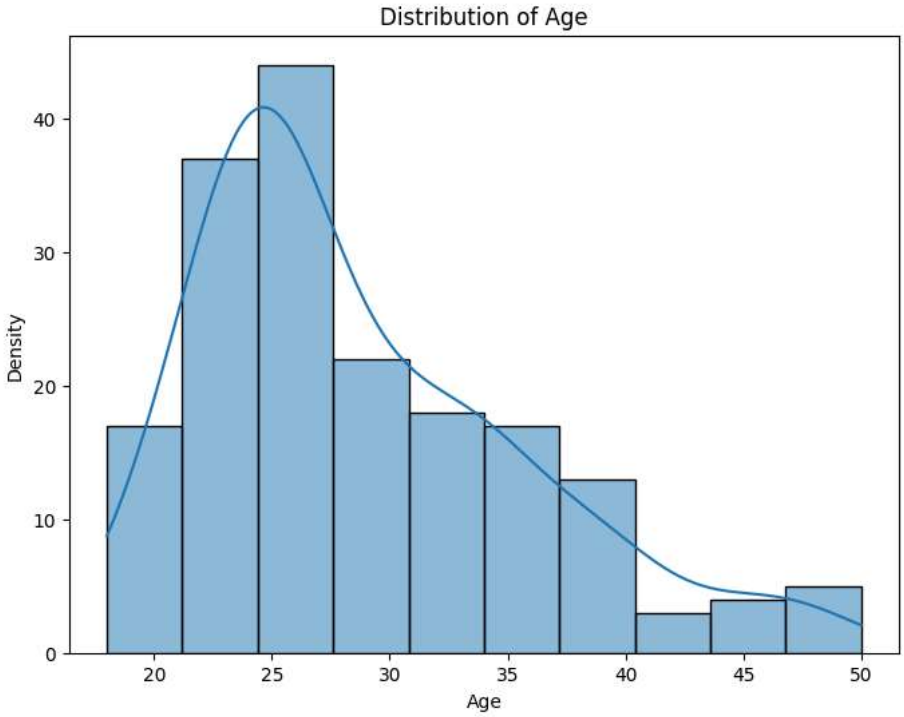
Next steps: Generate code with df View recommended plots

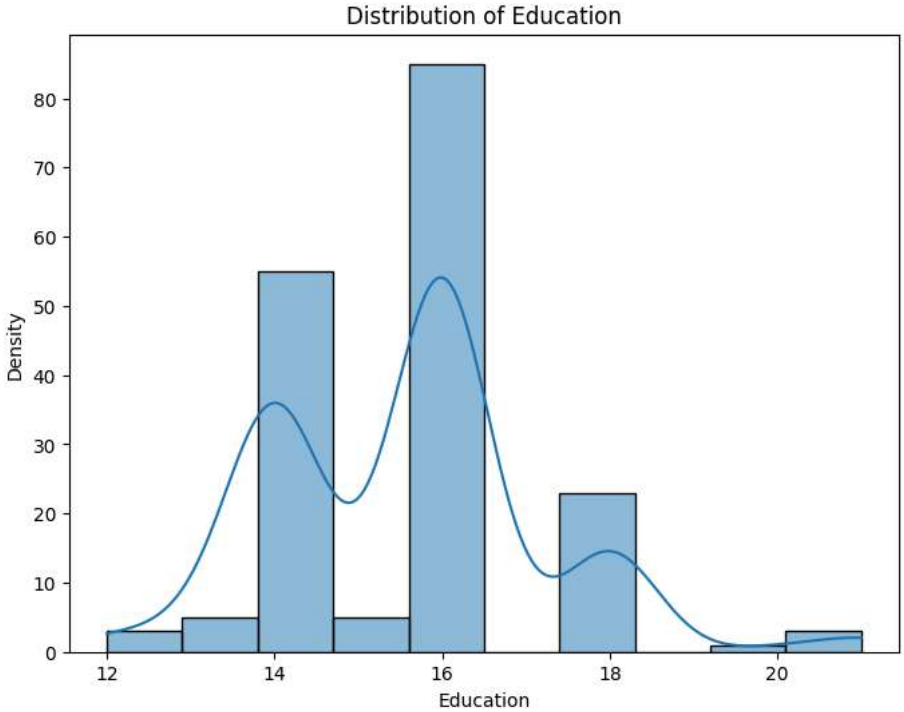
```
df["Age"].value_counts()
Age = df['Age'].value_counts()
print(Age)

# Create a bar plot
plt.figure(figsize=(8, 6))
ax2 = sns.barplot(x=Age.index, y=Age.values, palette='viridis', errorbar=None)

# Set plot title and labels
ax2.set_title('disribution of the Age')
ax2.set_xlabel('Age')
ax2.set_ylabel('Count')

# Annotate the bars with the counts
for container in ax2.containers:
    ax2.bar_label(container, fontsize=12)

plt.show()
```

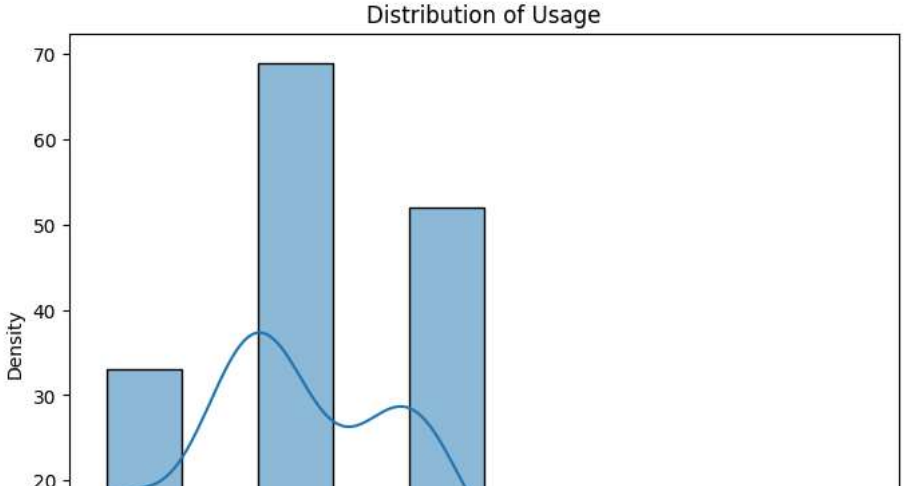
```
Age
25    25
23    18
24    12
26    12
28     9
35     8
33     8
30     7
38     7
21     7
22     7
27     7
31     6
34     6
29     6
20     5
40     5
32     4
19     4
48     2
37     2
45     2
47     2
46     1
50     1
18     1
44     1
43     1
41     1
39     1
36     1
42     1
Name: count, dtype: int64
<ipython-input-24-ef187f18e5cd>:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `leg

  ax2 = sns.barplot(x=Age.index, y=Age.values, palette='viridis', errorbar=None)
```

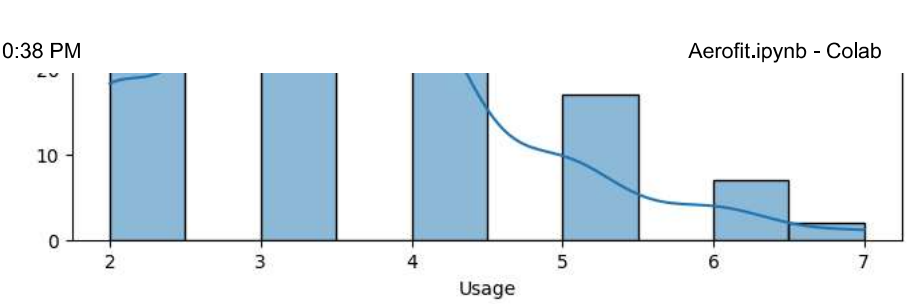


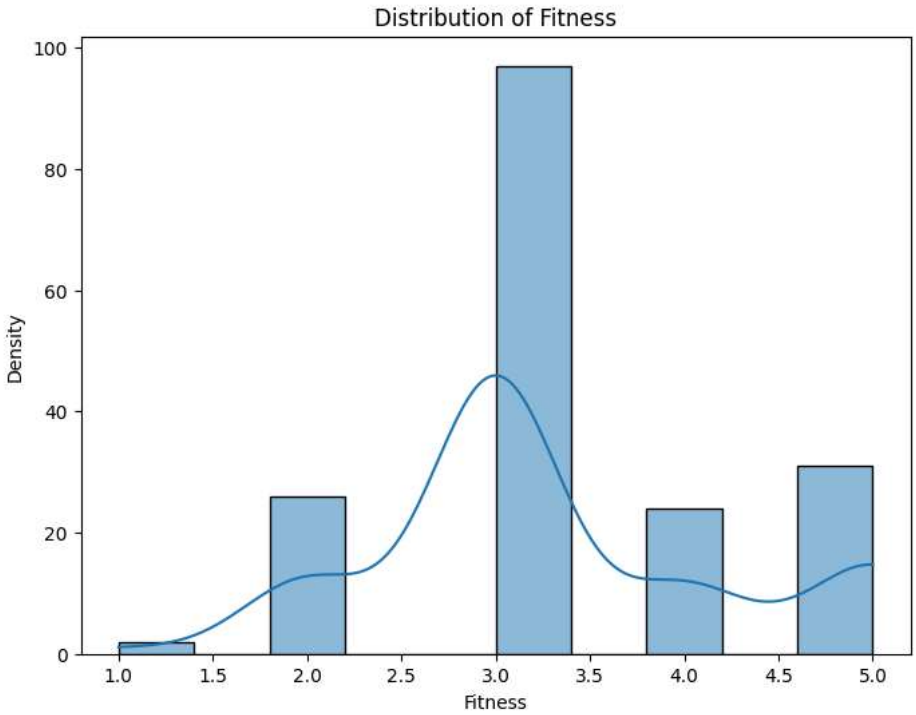
```
# Univariate Analysis for Continuous Variables
continuous_vars =df[ ['Age', 'Education', 'Usage', 'Fitness', 'Income', 'Miles']]

for var in continuous_vars:
    plt.figure(figsize=(8, 6))
    sns.histplot(df[var], kde=True, bins=10)
    plt.title(f'Distribution of {var}')
    plt.xlabel(var)
    plt.ylabel('Density')
    plt.show()
```

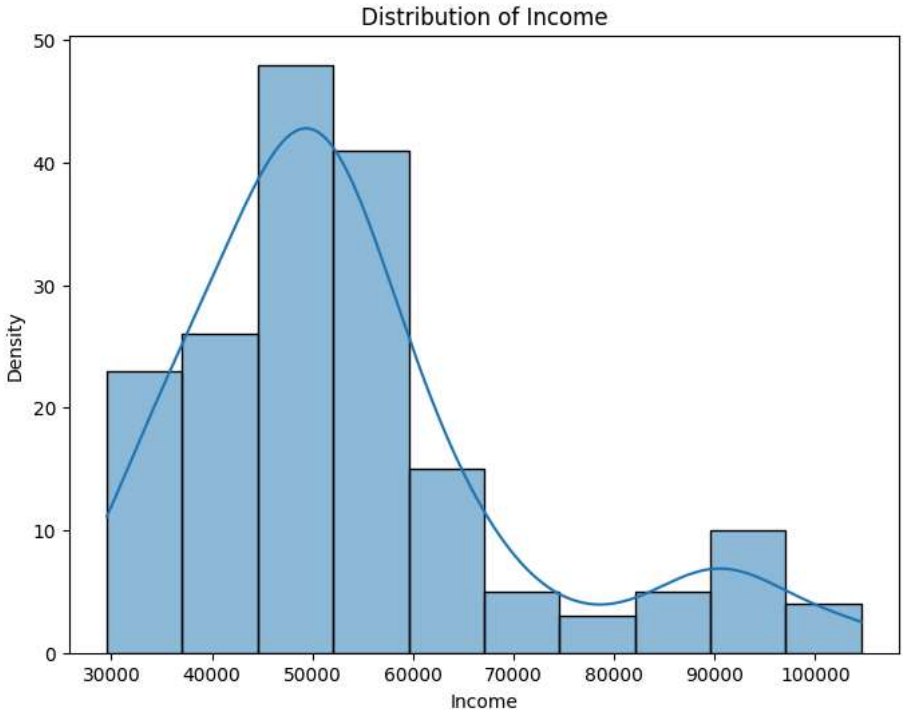

Distribution of Age
Density
Age
0
10
20
30
40
20
25
30
35
40
45
50

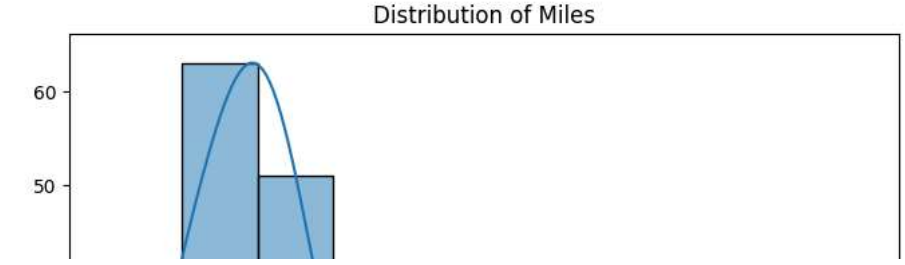

Distribution of Education
Density
Education
0
10
20
30
40
50
60
70
80
12
14
16
18
20

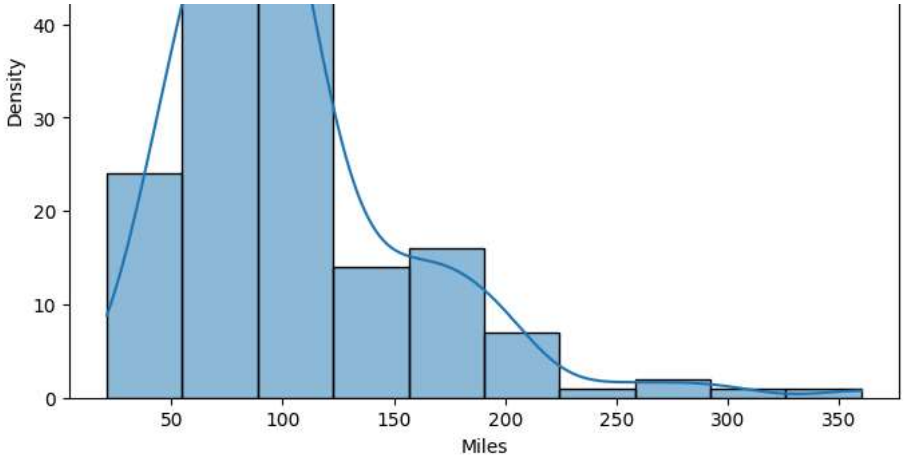

Distribution of Usage
Density
20
30
40
50
60
70


10
0
2
3
4
5
6
7
Usage

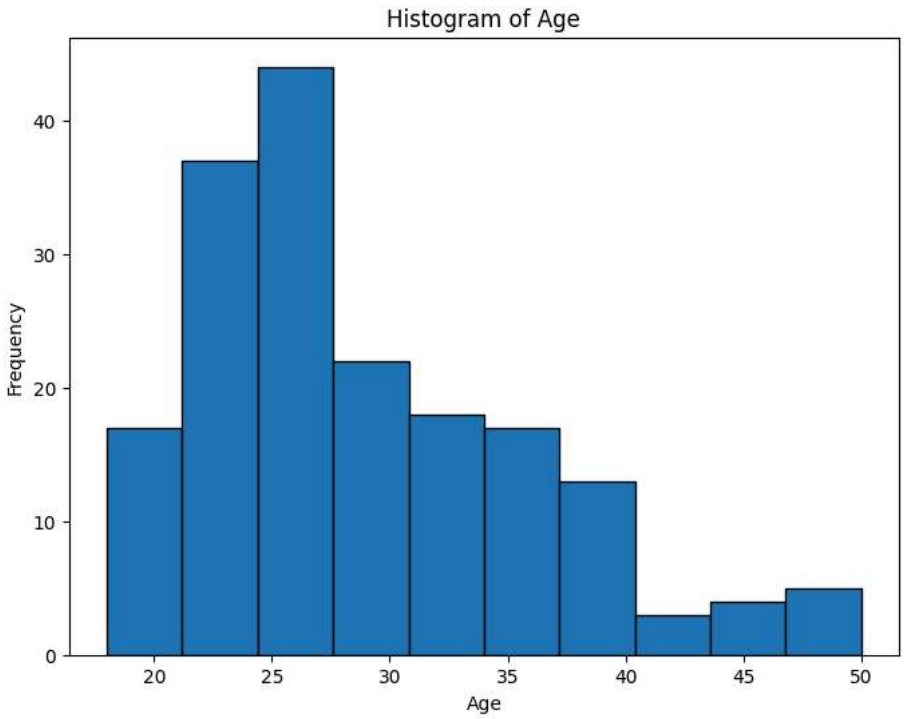

Distribution of Fitness
100
80
60
40
20
0
Density
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Fitness

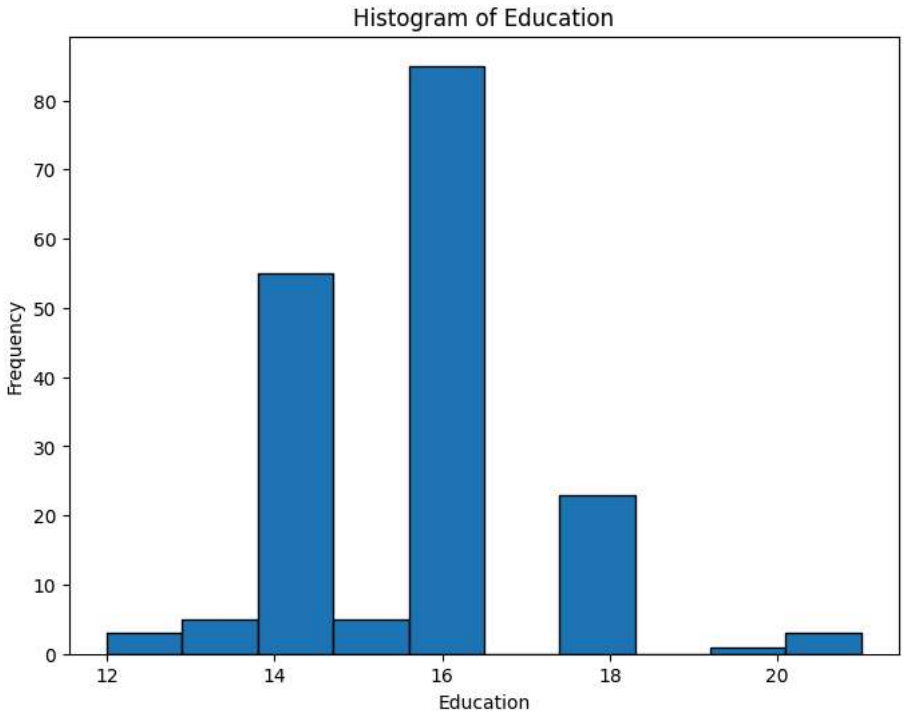

Distribution of Income
50
40
30
20
10
0
Density
30000
40000
50000
60000
70000
80000
90000
100000
Income

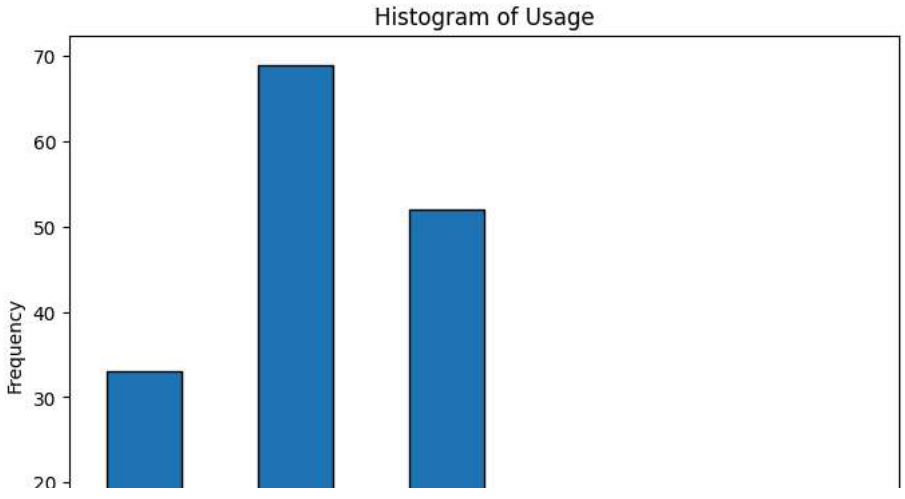

Distribution of Miles
60
50

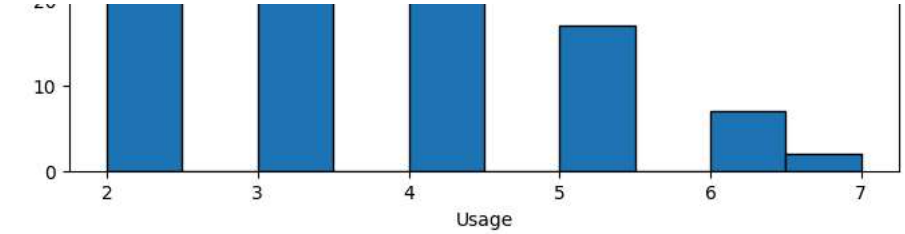


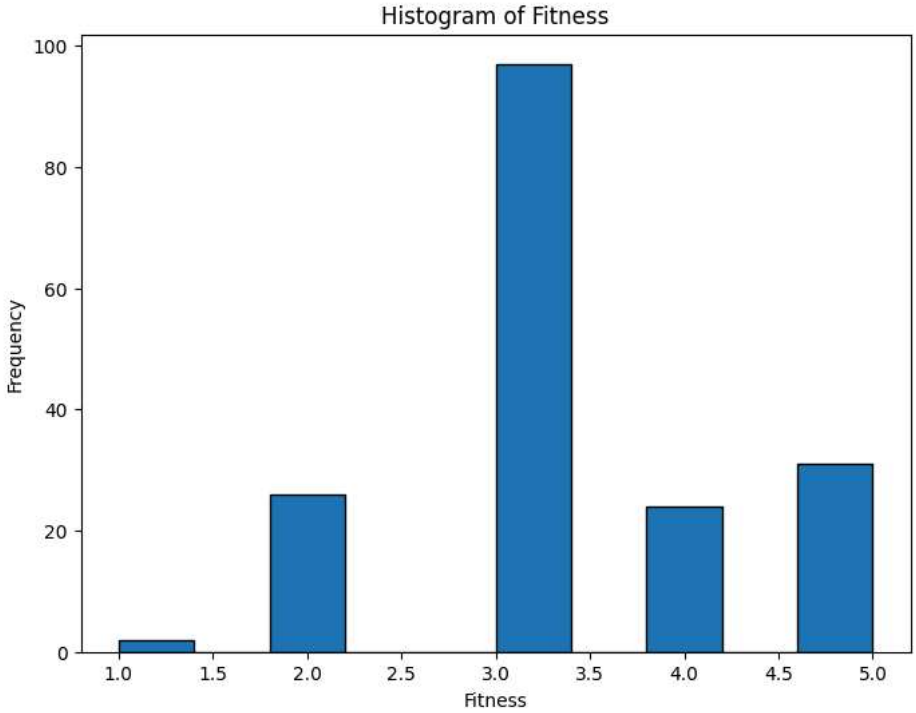

```
for var in continuous_vars:
    plt.figure(figsize=(8, 6))
    plt.hist(df[var], bins=10, edgecolor='black')
    plt.title(f'Histogram of {var}')
    plt.xlabel(var)
    plt.ylabel('Frequency')
    plt.show()
```

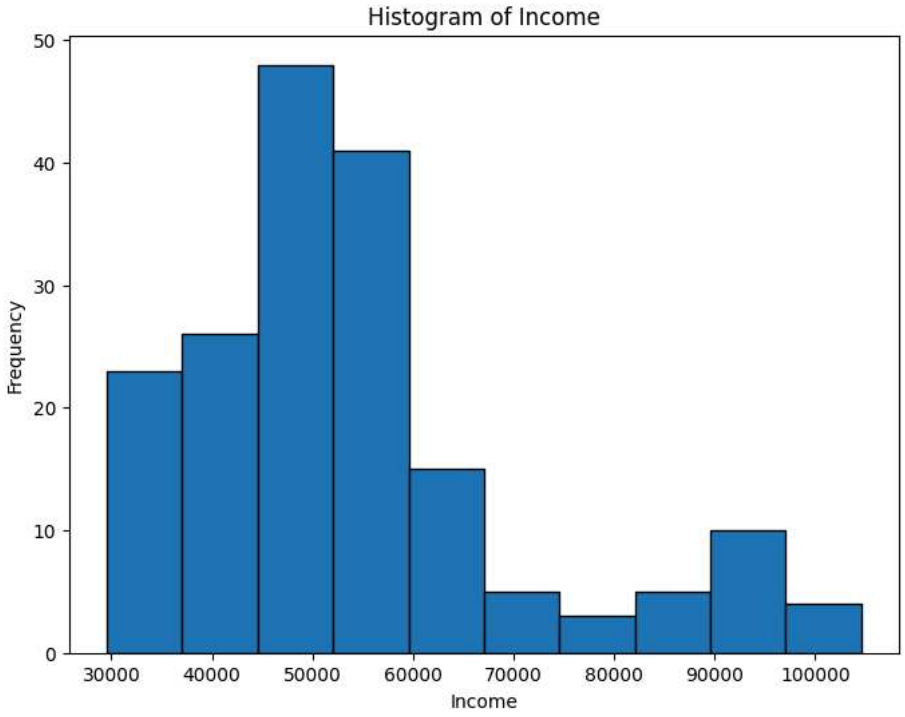

Histogram of Age
Frequency
Age
0
10
20
30
40
20
25
30
35
40
45
50

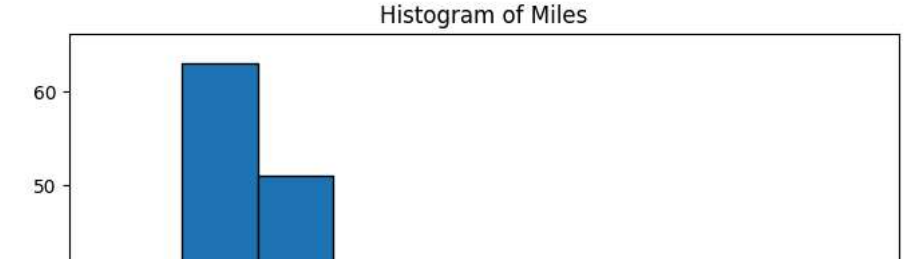

Histogram of Education
Frequency
Education
0
10
20
30
40
50
60
70
80
12
14
16
18
20

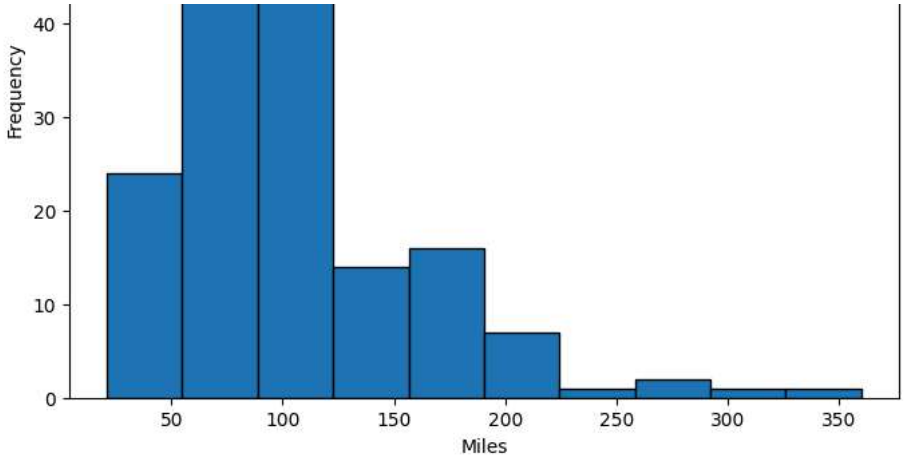

Histogram of Usage
Frequency
20
30
40
50
60
70


10
0
2
3
4
5
6
7
Usage

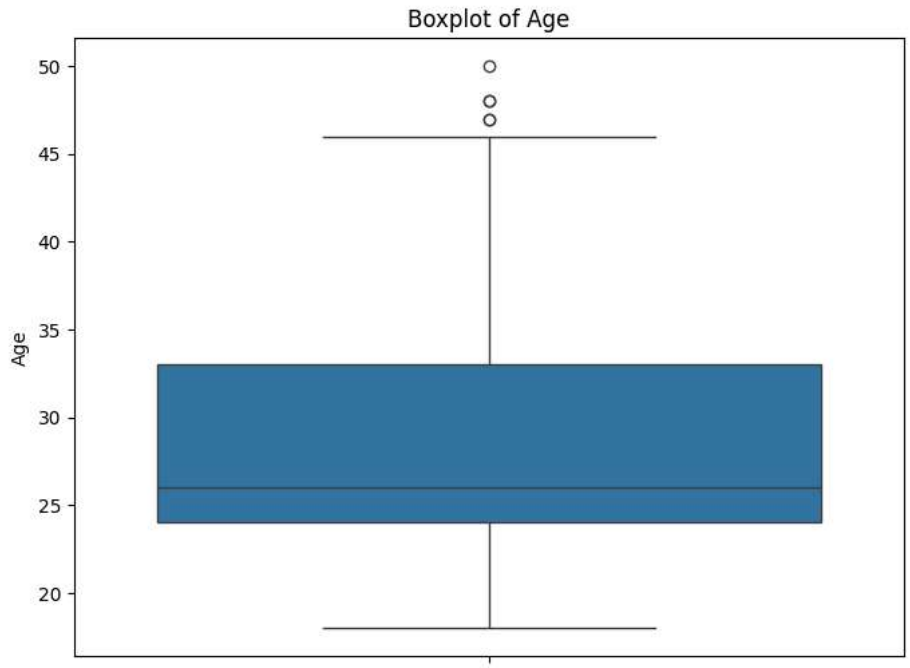

Histogram of Fitness
100
80
60
40
20
0
Frequency
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Fitness

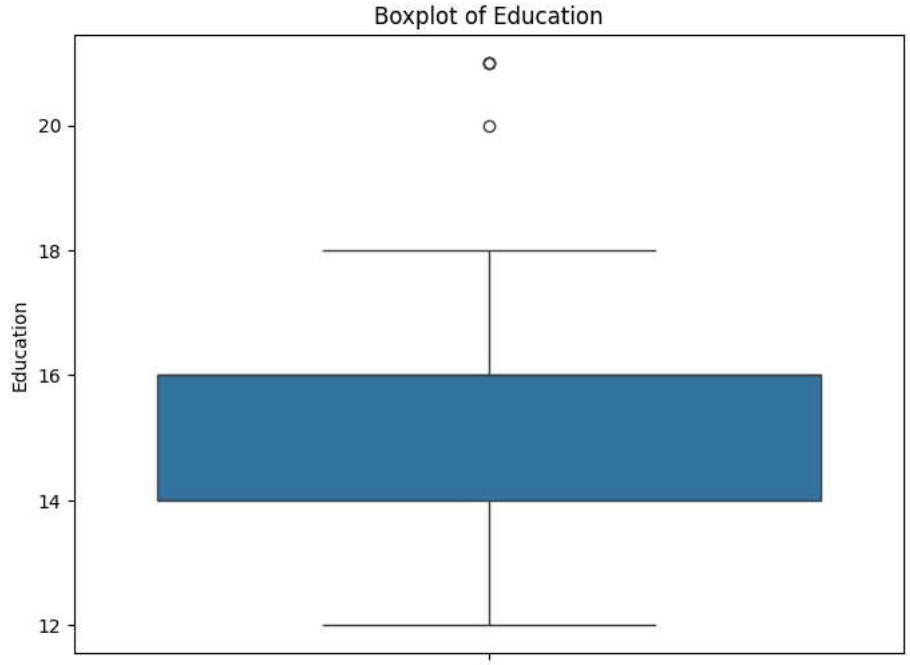

Histogram of Income
50
40
30
20
10
0
Frequency
30000
40000
50000
60000
70000
80000
90000
100000
Income

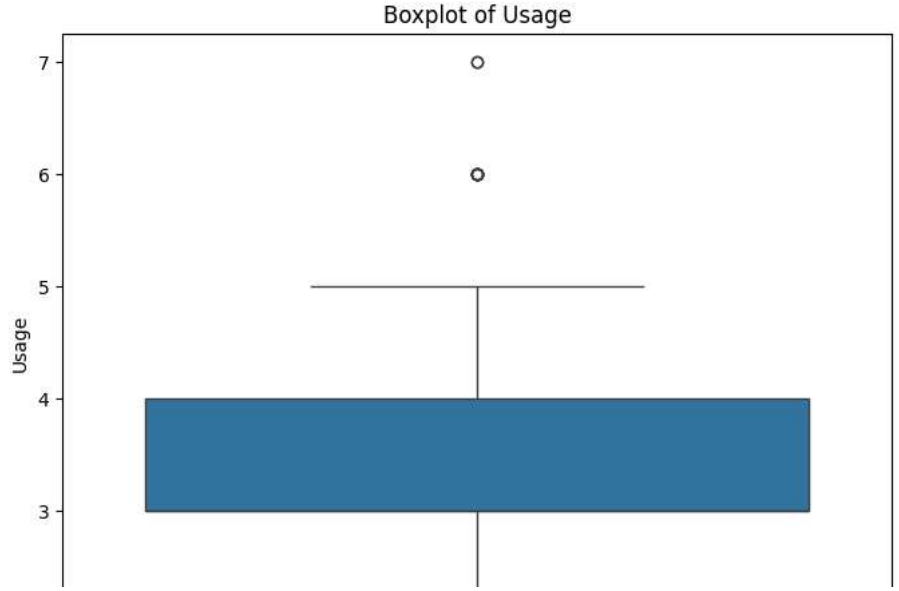

Histogram of Miles
60
50



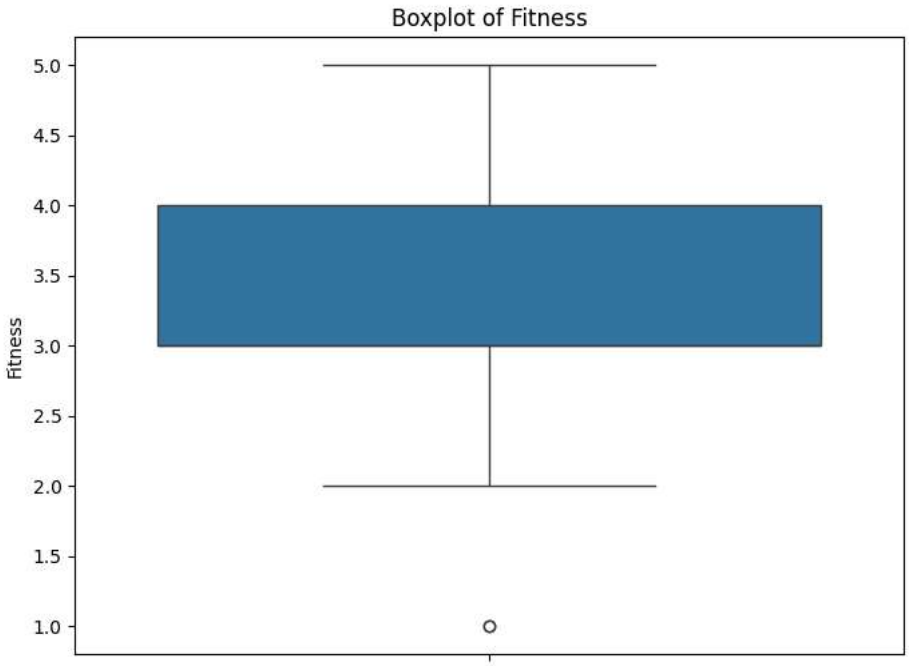

```
for var in continuous_vars:
    plt.figure(figsize=(8, 6))
    sns.boxplot(y=df[var])
    plt.title(f'Boxplot of {var}')
    plt.ylabel(var)
    plt.show()
```

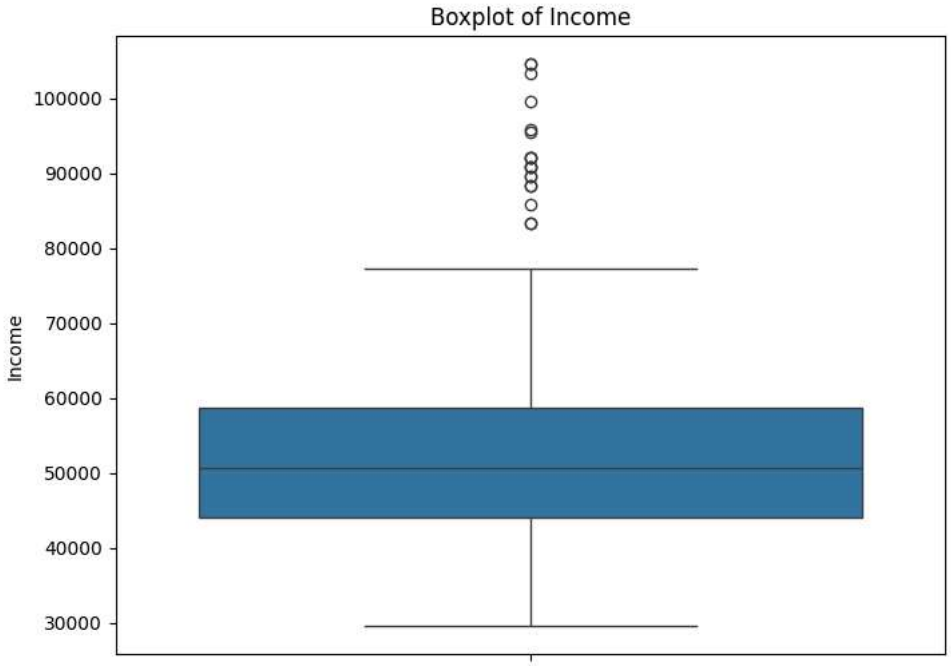

Boxplot of Age
Age
50
45
40
35
30
25
20

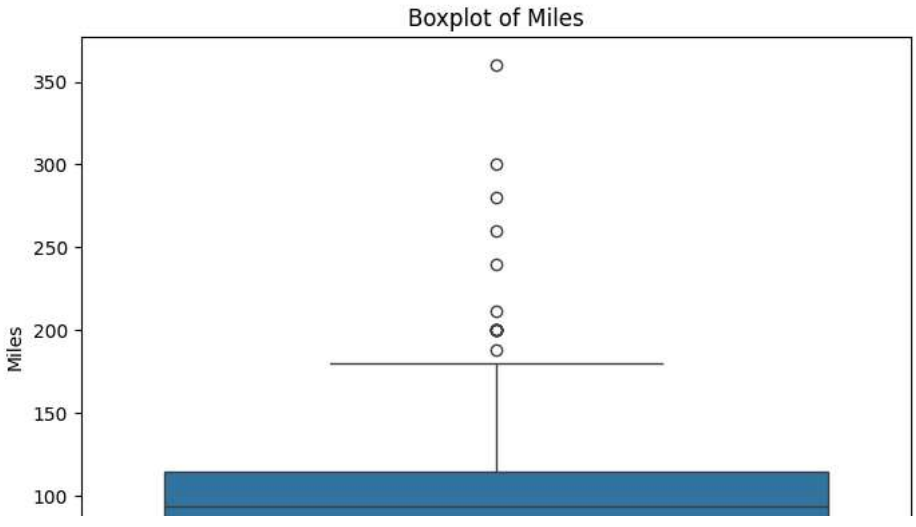

Boxplot of Education
Education
20
18
16
14
12

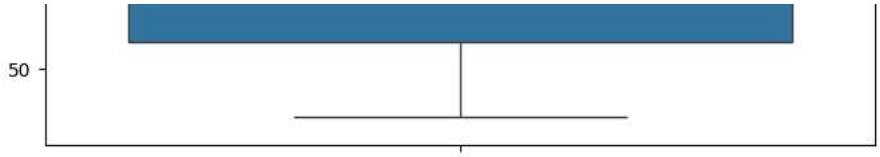

Boxplot of Usage
Usage
7
6
5
4
3


2


Boxplot of Fitness
Fitness
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0

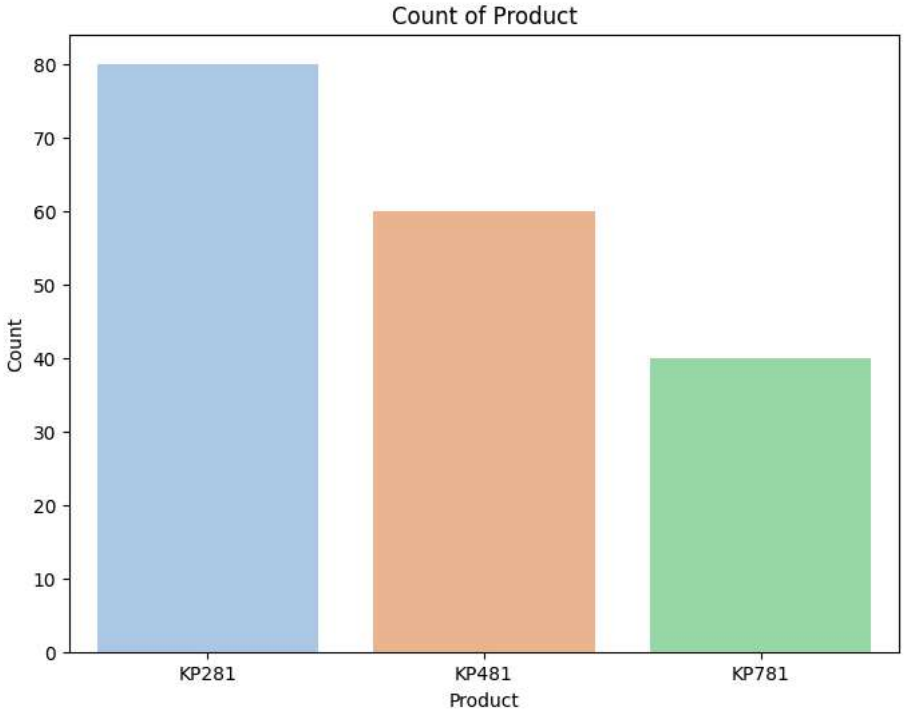

Boxplot of Income
Income
100000
90000
80000
70000
60000
50000
40000
30000


Boxplot of Miles
Miles
350
300
250
200
150
100

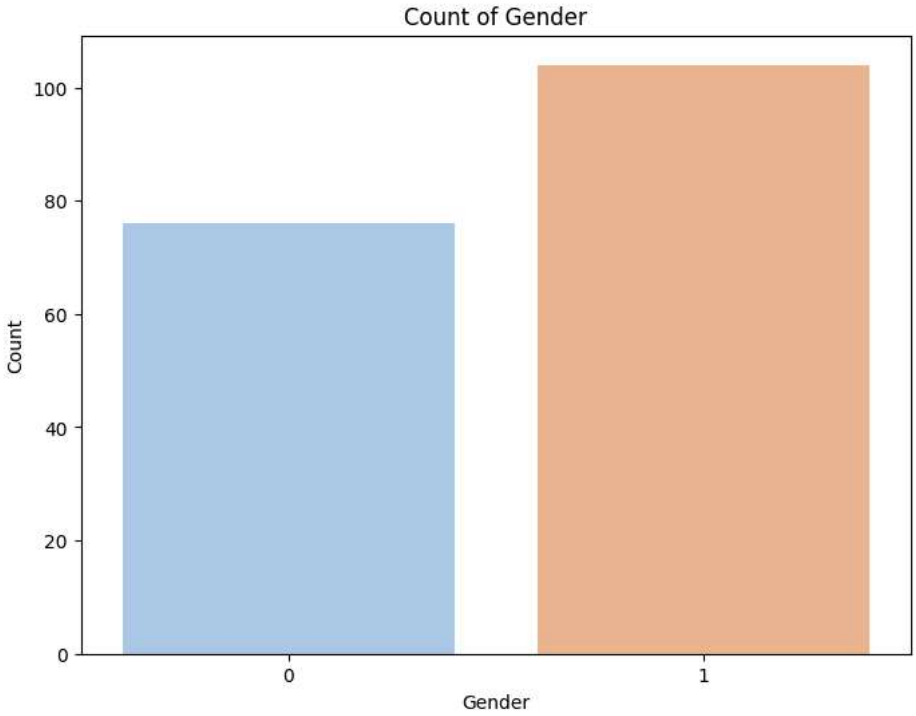



```
# Univariate Analysis for Categorical Variables
categorical_vars = df[['Product', 'Gender', 'MaritalStatus']]

for var in categorical_vars:
    plt.figure(figsize=(8, 6))
    sns.countplot(x=df[var],palette='pastel')
    plt.title(f'Count of {var}')
    plt.xlabel(var)
    plt.ylabel('Count')
    plt.show()
```

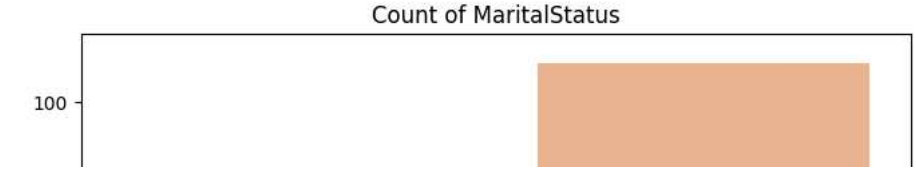
```
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `leg

  sns.countplot(x=df[var],palette='pastel')
```

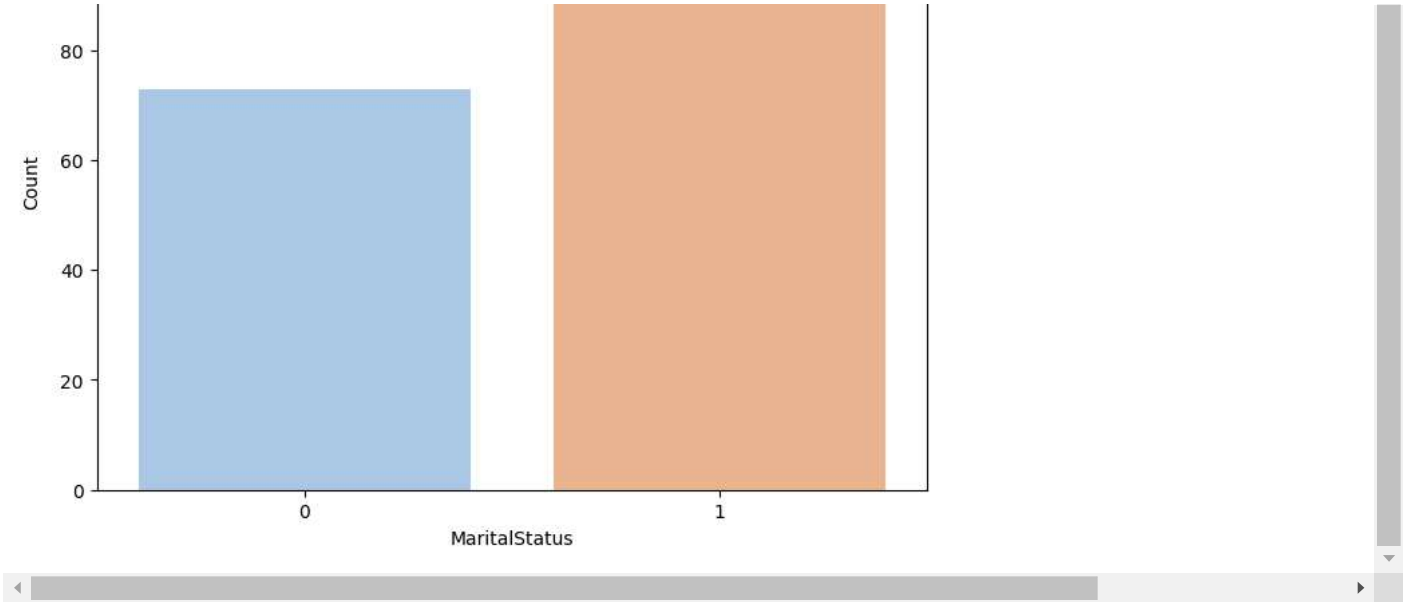



```
<ipython-input-28-10638b52232a>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `leg

  sns.countplot(x=df[var],palette='pastel')
```




```
<ipython-input-28-10638b52232a>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `leg

  sns.countplot(x=df[var],palette='pastel')
```

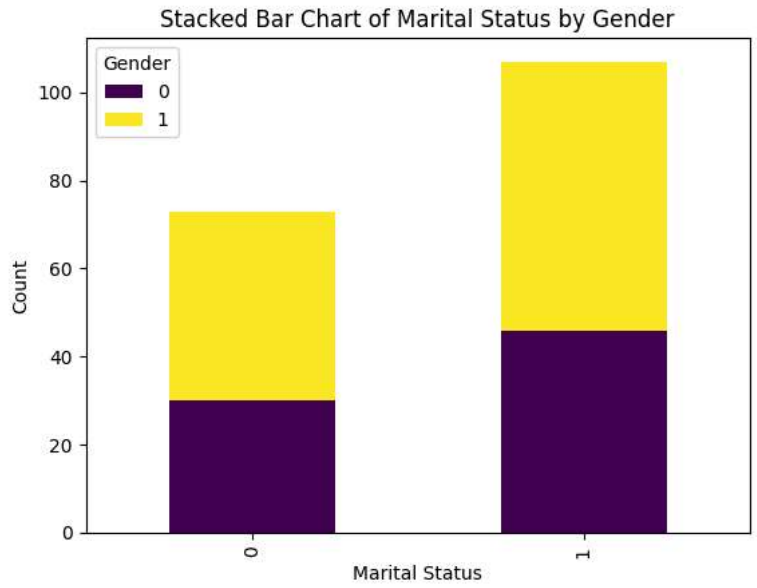





```
df.dtypes
```

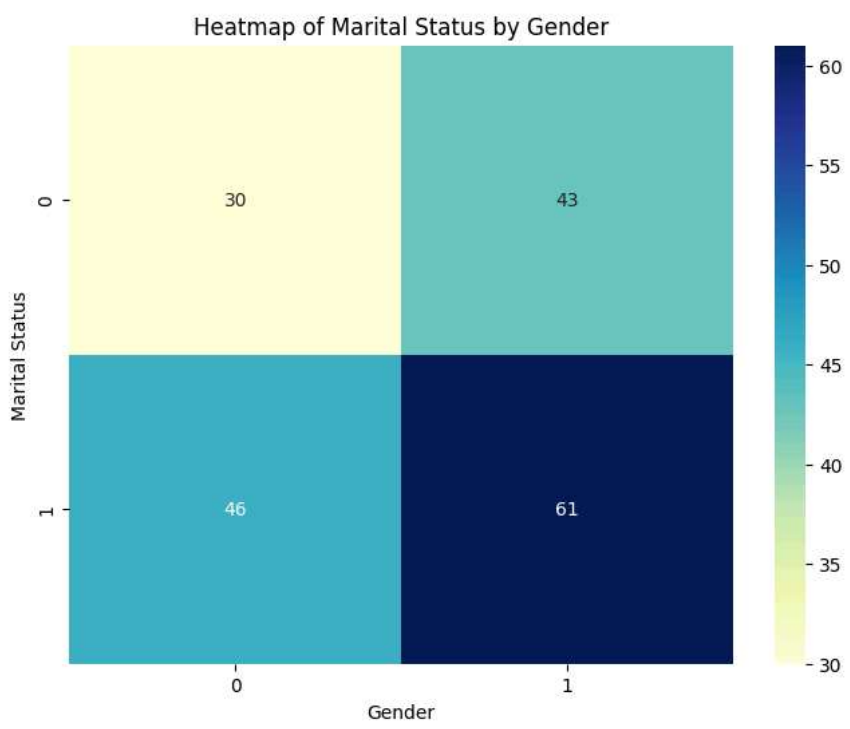
```
Product          object
Age               int64
Gender            int64
Education         int64
MaritalStatus     int64
Usage             int64
Fitness           int64
Income            int64
Miles             int64
dtype: object
```

```
# Cross-tabulation of Gender and MaritalStatus
crosstab = pd.crosstab(df['MaritalStatus'], df['Gender'])

# Plot stacked bar chart
crosstab.plot(kind='bar', stacked=True, colormap='viridis')
plt.title('Stacked Bar Chart of Marital Status by Gender')
plt.xlabel('Marital Status')
plt.ylabel('Count')
plt.show()
```



Heatmap for Categorical Variables

```
# Cross-tabulation of Gender and MaritalStatus
crosstab = pd.crosstab(df['MaritalStatus'], df['Gender'])

# Heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(crosstab, annot=True, fmt='d', cmap='YlGnBu')
plt.title('Heatmap of Marital Status by Gender')
plt.xlabel('Gender')
plt.ylabel('Marital Status')
plt.show()
```

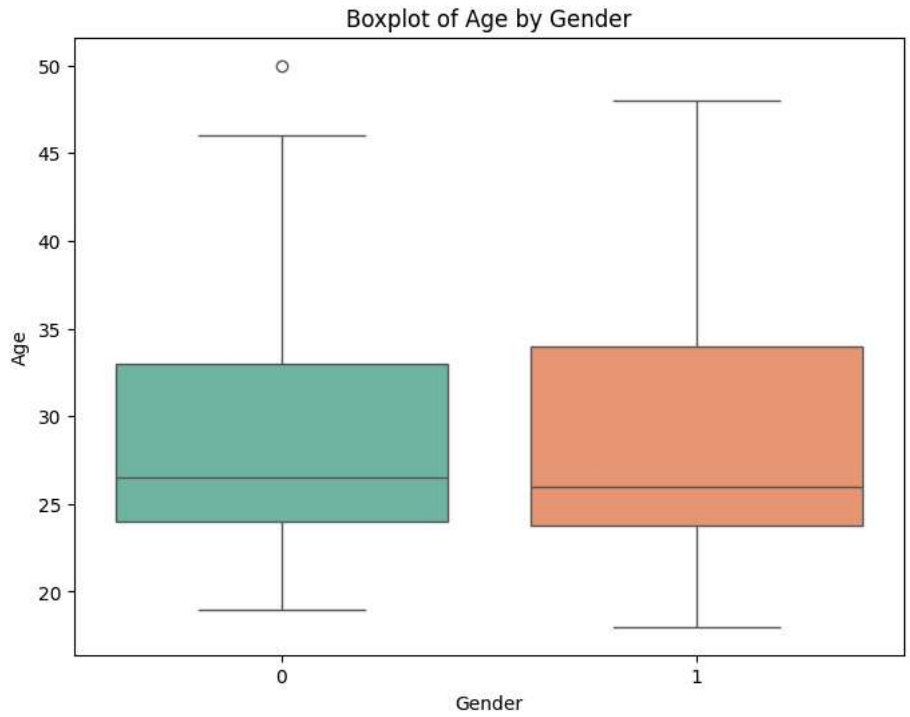



```
df.columns
```

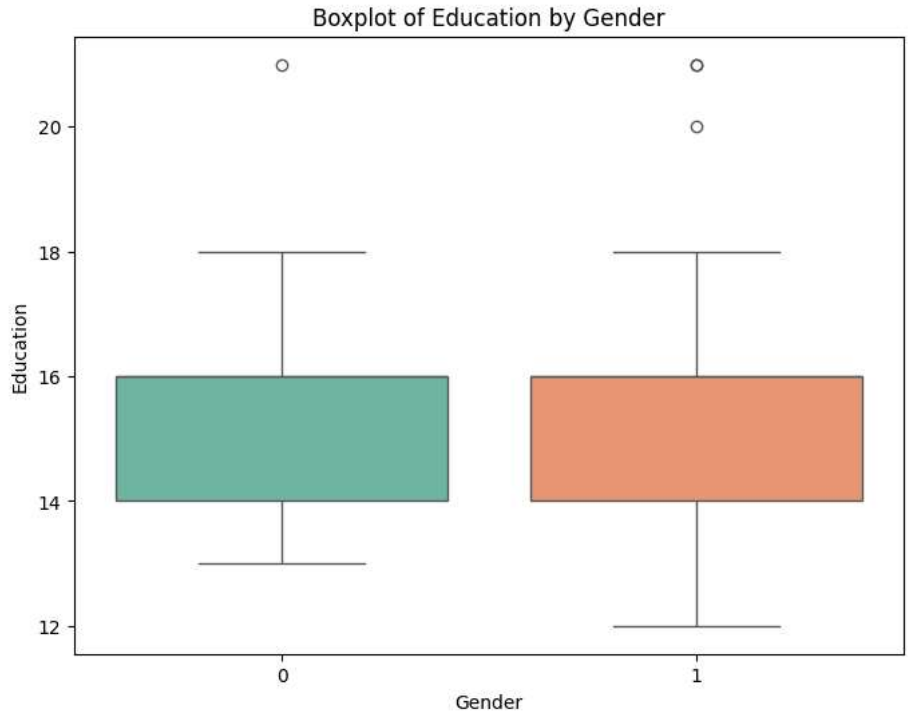
```
Index(['Product', 'Age', 'Gender', 'Education', 'MaritalStatus', 'Usage',
       'Fitness', 'Income', 'Miles'],
      dtype='object')
```

```
# Boxplot by Gender
for var in df[['Age', 'Education', 'Usage', 'Fitness', 'Income', 'Miles']]:
    plt.figure(figsize=(8, 6))
    sns.boxplot(x=df['Gender'], y=var, data=df, palette='Set2')
    plt.title(f'Boxplot of {var} by Gender')
    plt.xlabel('Gender')
    plt.ylabel(var)
    plt.show()

# Boxplot by MaritalStatus
for var in df[ ['Age', 'Education', 'Usage', 'Fitness', 'Income', 'Miles']]:
    plt.figure(figsize=(8, 6))
    sns.boxplot(x=df['MaritalStatus'], y=var, data=df, palette='husl')
    plt.title(f'Boxplot of {var} by Marital Status')
    plt.xlabel('Marital Status')
    plt.ylabel(var)
    plt.show()
```

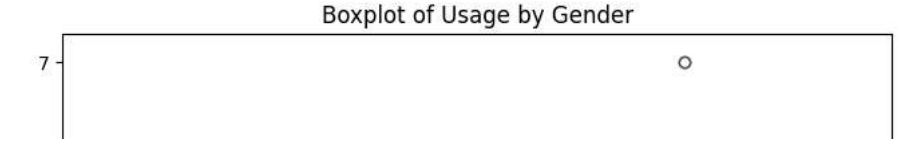
```
<ipython-input-33-4eb4b302ca23>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `leg

  sns.boxplot(x=df['Gender'], y=var, data=df, palette='Set2')
```

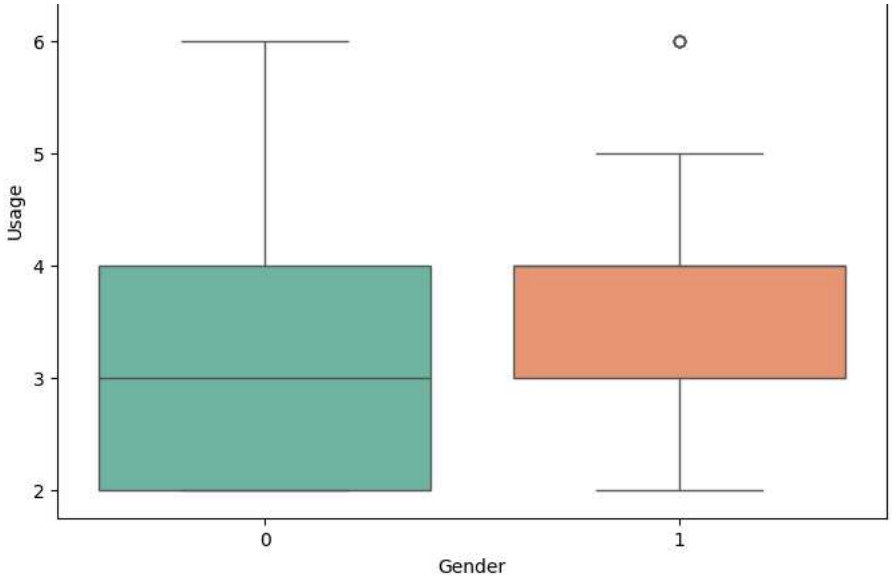



```
<ipython-input-33-4eb4b302ca23>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `leg

  sns.boxplot(x=df['Gender'], y=var, data=df, palette='Set2')
```

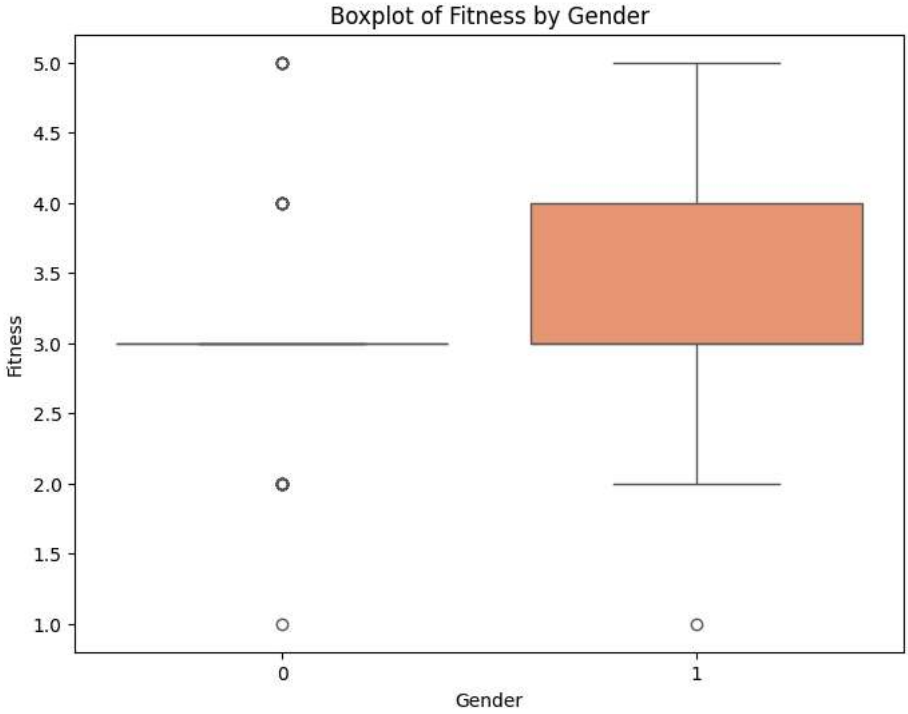



```
<ipython-input-33-4eb4b302ca23>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `leg

  sns.boxplot(x=df['Gender'], y=var, data=df, palette='Set2')
```

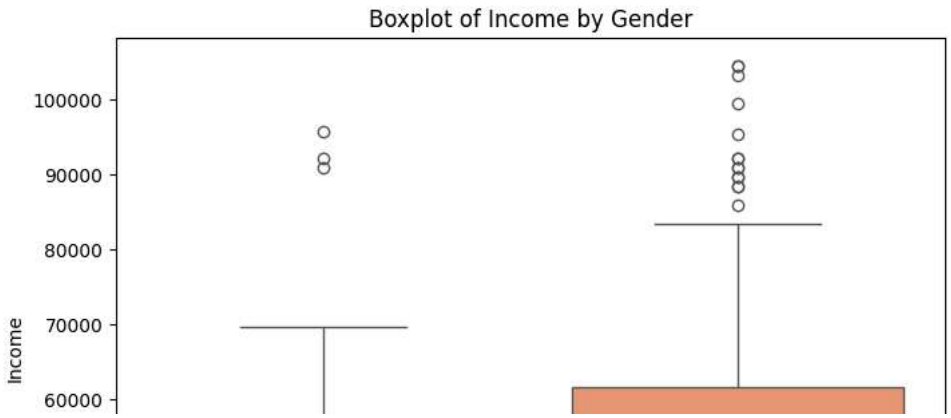


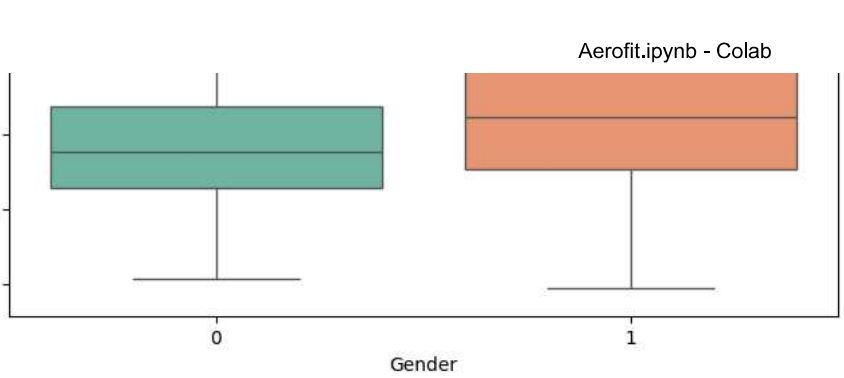



```
<ipython-input-33-4eb4b302ca23>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `leg

  sns.boxplot(x=df['Gender'], y=var, data=df, palette='Set2')
```

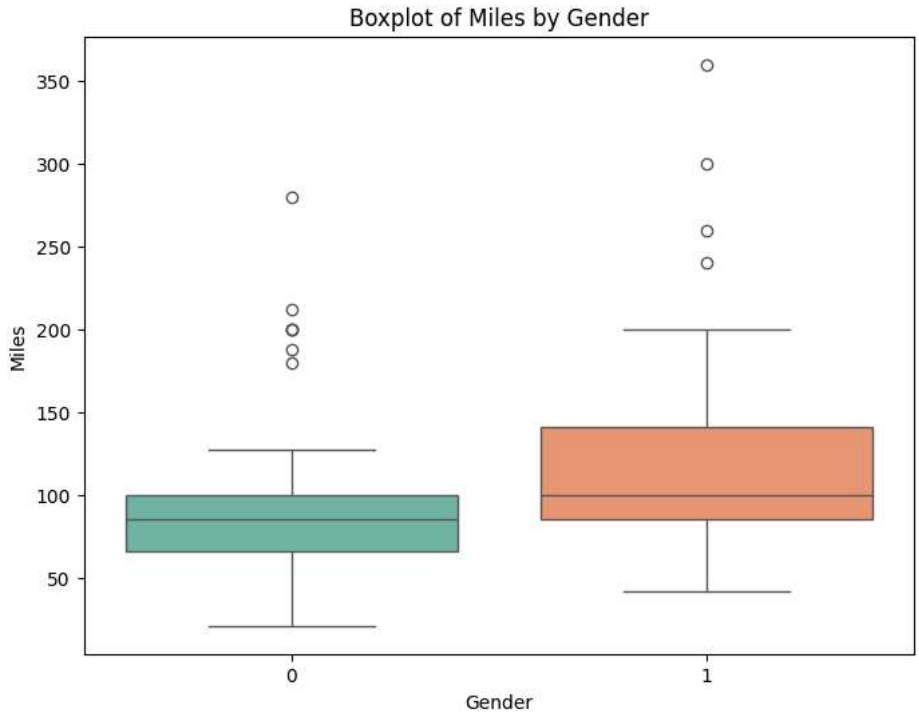



```
<ipython-input-33-4eb4b302ca23>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `leg

  sns.boxplot(x=df['Gender'], y=var, data=df, palette='Set2')
```

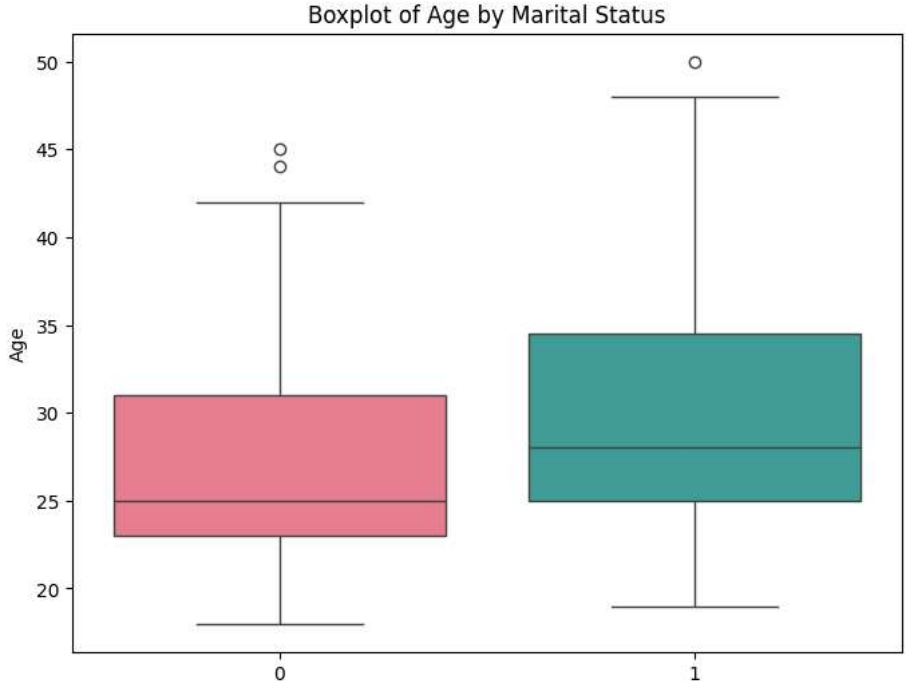





```
<ipython-input-33-4eb4b302ca23>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `leg

  sns.boxplot(x=df['Gender'], y=var, data=df, palette='Set2')
```

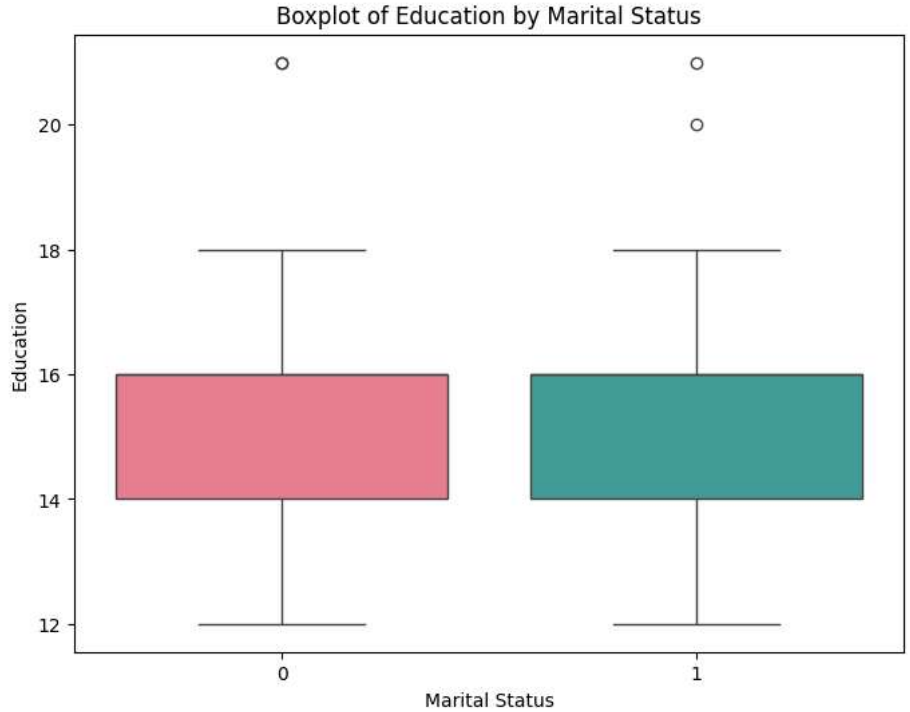



```
<ipython-input-33-4eb4b302ca23>:13: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `leg

  sns.boxplot(x=df['MaritalStatus'], y=var, data=df, palette='husl')
```

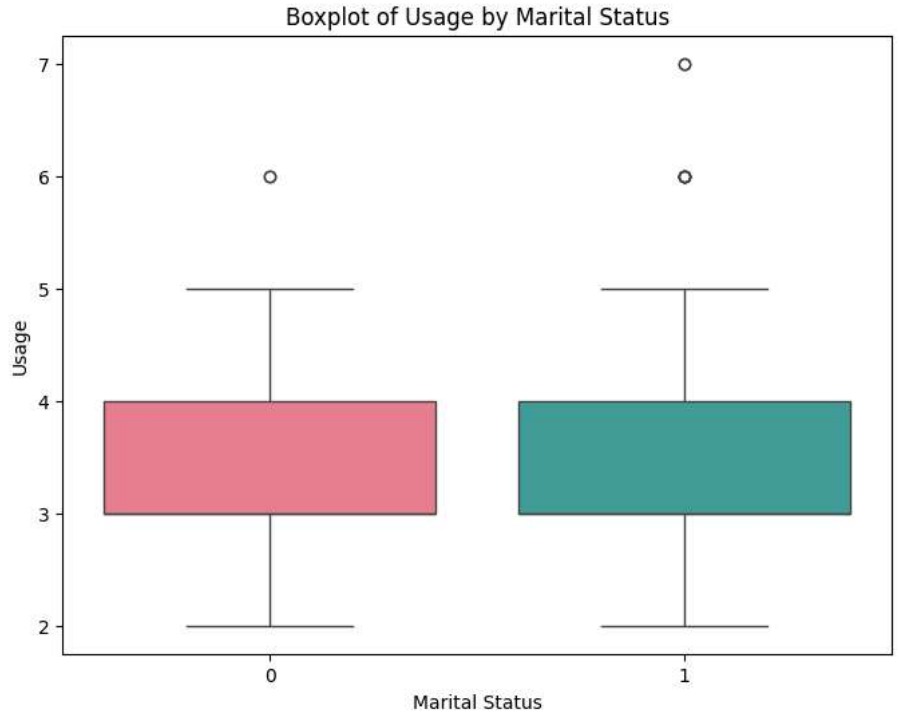


Marital Status

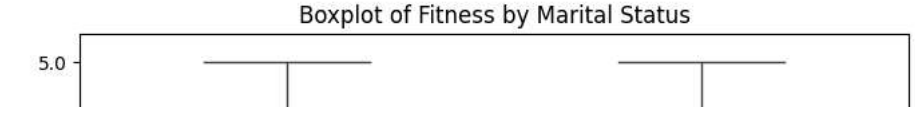
```
<ipython-input-33-4eb4b302ca23>:13: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `leg

  sns.boxplot(x=df['MaritalStatus'], y=var, data=df, palette='husl')
```

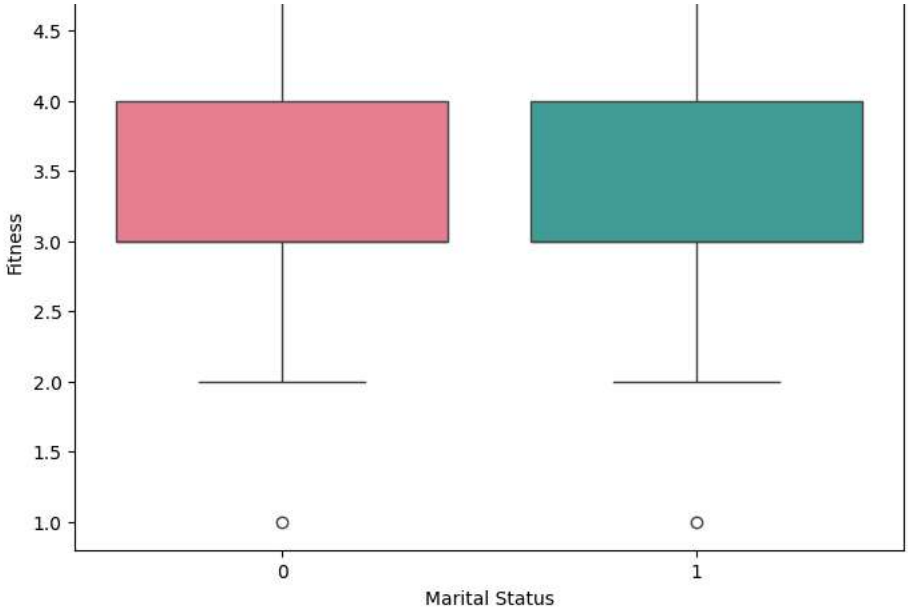



```
<ipython-input-33-4eb4b302ca23>:13: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `leg

  sns.boxplot(x=df['MaritalStatus'], y=var, data=df, palette='husl')
```

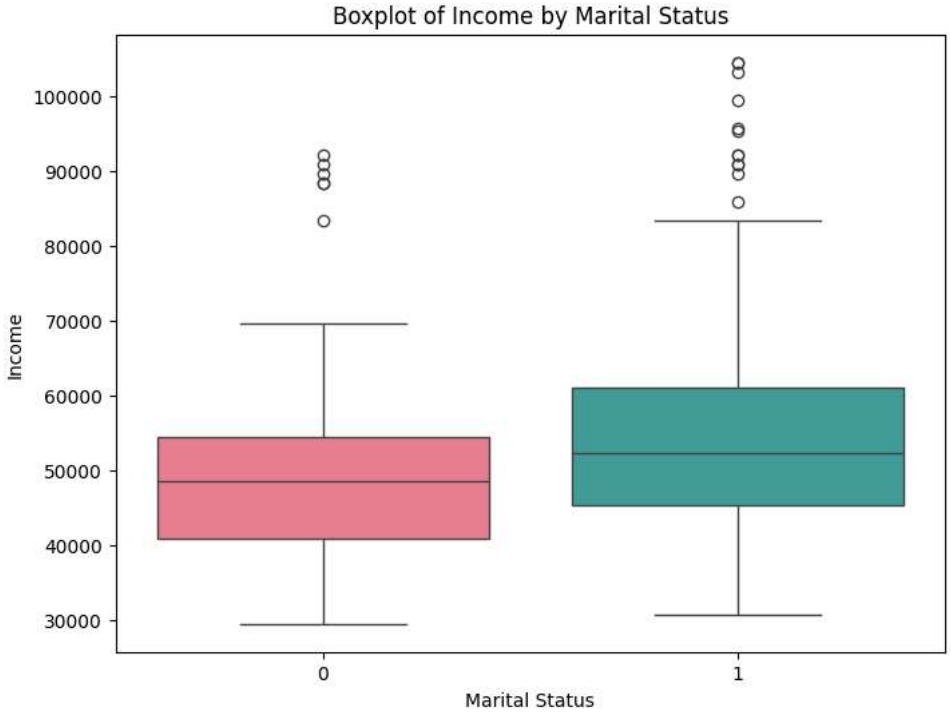



```
<ipython-input-33-4eb4b302ca23>:13: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `leg

  sns.boxplot(x=df['MaritalStatus'], y=var, data=df, palette='husl')
```

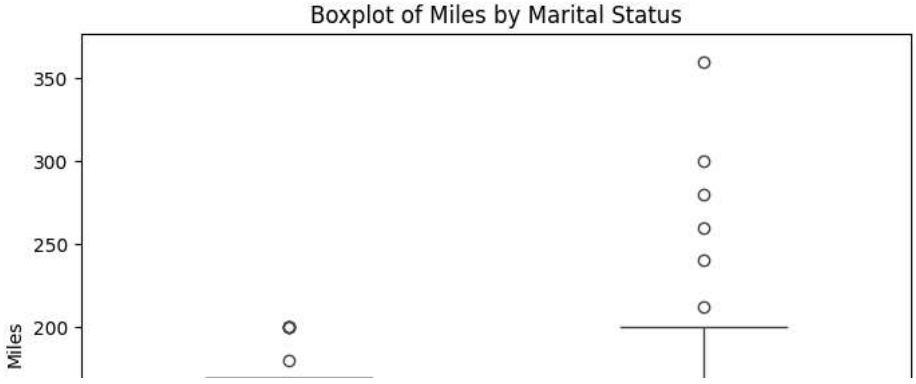


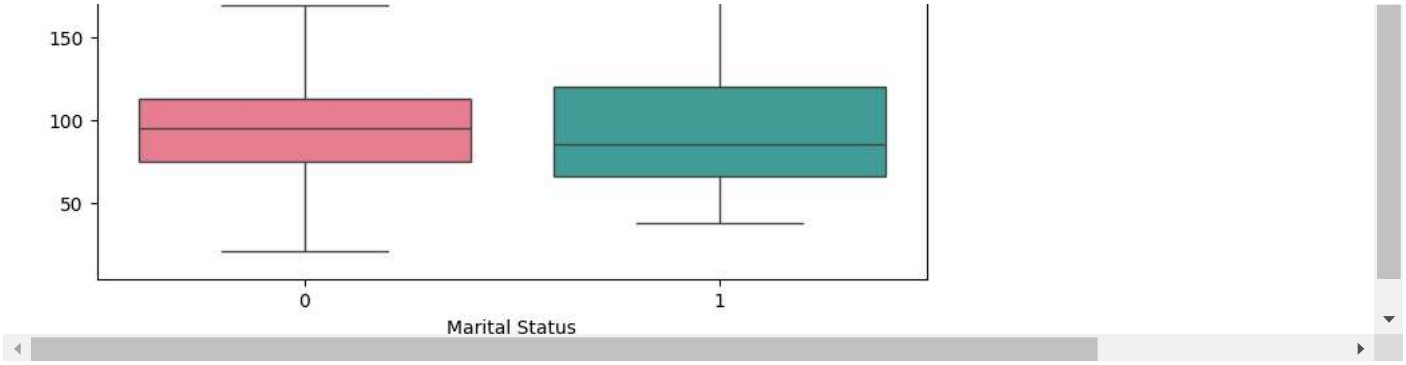



```
<ipython-input-33-4eb4b302ca23>:13: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `leg

  sns.boxplot(x=df['MaritalStatus'], y=var, data=df, palette='husl')
```

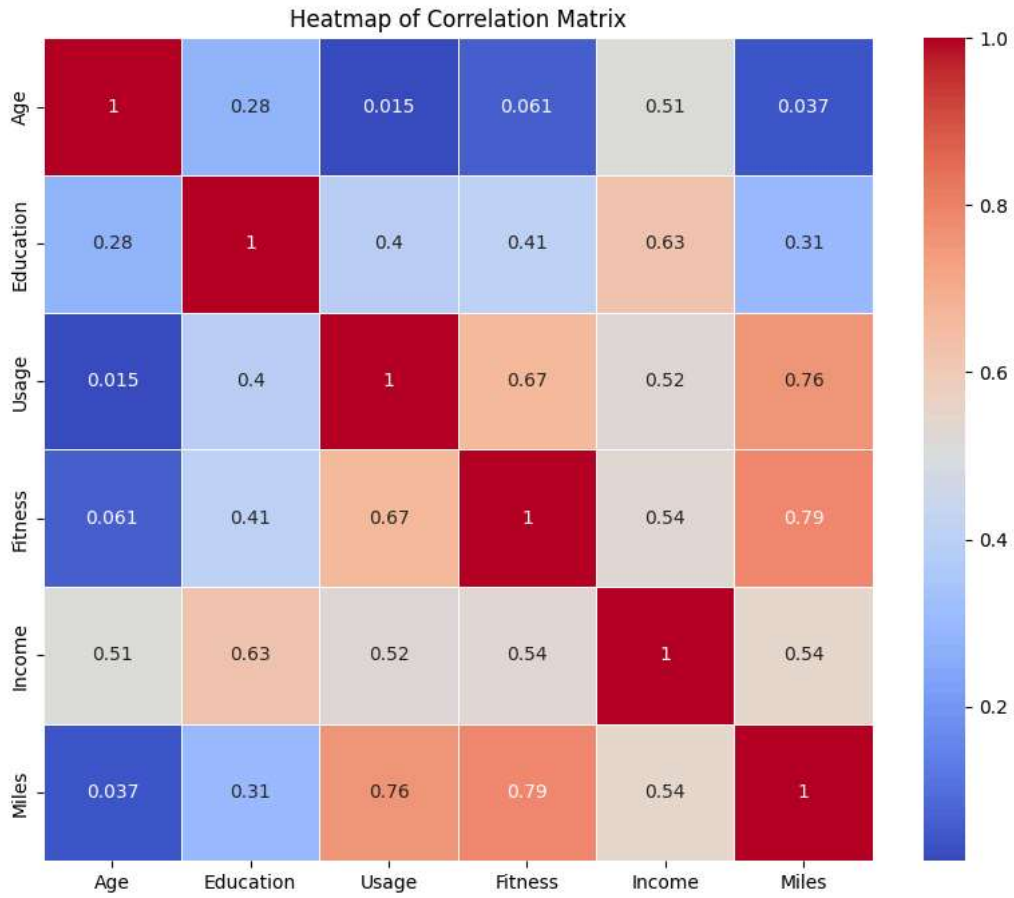



```
<ipython-input-33-4eb4b302ca23>:13: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `leg

  sns.boxplot(x=df['MaritalStatus'], y=var, data=df, palette='husl')
```






```
# Correlation matrix
df1=df[['Age', 'Education', 'Usage', 'Fitness', 'Income', 'Miles']]
corr = df1.corr()

# Plot heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(corr, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Heatmap of Correlation Matrix')
plt.show()
```



MILES,INCOME,FITNESS AND USAGE HAVE HIGHLY STRONG POSITIVE CORRELATION OF DATA

Start coding or generate with AI.

Start coding or generate with AI.

Missing Value & Outlier Detection

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

```
Product          0
Age              0
Gender           0
Education        0
MaritalStatus    0
Usage            0
Fitness          0
Income           0
Miles            0
dtype: int64
```

```
df.isna().sum()
```

```
Product          0
Age              0
Gender           0
Education        0
MaritalStatus    0
Usage            0
Fitness          0
Income           0
Miles            0
dtype: int64
```

Start coding or generate with AI.

THERE ARE NO MISSING DATA OR THE VALUES IN DATA PROVIDED BY COMPANY THERFORE IT DOSENOT CREATE ANY FURTHER HURDLES IN THE ANALYSIS OF THE DATA

Start coding or generate with AI.

```
df["Product"].value_counts()
```

```
Product
KP281    80
KP481    60
KP781    40
Name: count, dtype: int64
```

```
df["Product"].unique()
```

```
array(['KP281', 'KP481', 'KP781'], dtype=object)
```

```
df["Product"].replace({"KP281":0,"KP481":1,"KP781":2},inplace=True)
```

```
df
```

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
0	0	18	1	14	0	3	4	29562	112
1	0	19	1	15	0	2	3	31836	75
2	0	19	0	14	1	4	3	30699	66
3	0	19	1	12	0	3	3	32973	85
4	0	20	1	13	1	4	2	35247	47
...	...	...	...	...	...	...	...	...	...
175	2	40	1	21	0	6	5	83416	200
176	2	42	1	18	0	5	4	89641	200
177	2	45	1	16	0	5	5	90886	160
178	2	47	1	18	1	4	5	104581	120
179	2	48	1	18	1	4	5	95508	180

180 rows × 9 columns

Next steps: Generate code with df View recommended plots

```
# Z-Score method for outlier detection
from scipy import stats
z_scores = stats.zscore(df)
threshold = 3
outliers = df[(z_scores > threshold).any(axis=1)]
print("Outliers (Z-Score method):")
print(outliers)
```

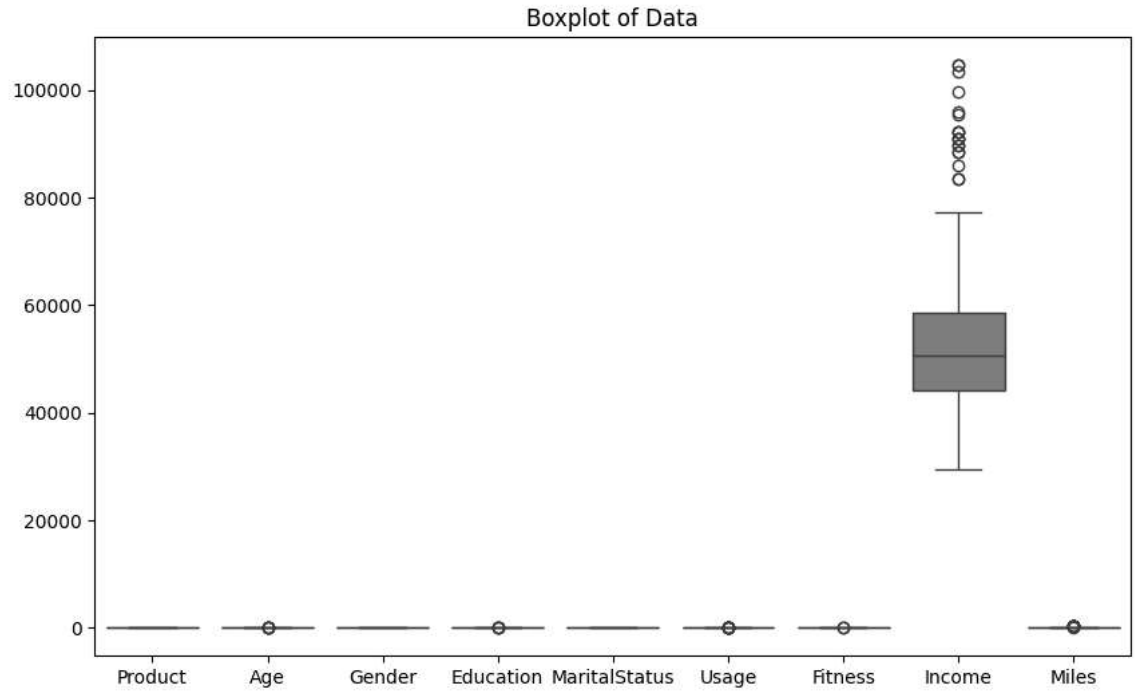
```
Outliers (Z-Score method):
     Product  Age  Gender  Education  MaritalStatus  Usage  Fitness  Income  \
79         0   50       0         16              1      3        3   64809
157        2   26       0         21              0      4        3   69721
161        2   27       1         21              1      4        4   90886
163        2   28       1         18              1      7        5   77191
166        2   29       1         14              1      7        5   85906
167        2   30       0         16              1      6        5   90886
168        2   30       1         18              1      5        4  103336
170        2   31       1         16              1      6        5   89641
173        2   35       1         16              1      4        5   92131
174        2   38       1         18              1      5        5  104581
175        2   40       1         21              0      6        5   83416
178        2   47       1         18              1      4        5  104581

     Miles
79      66
157    100
161    100
163    180
166    300
167    280
168    160
170    260
173    360
174    150
175    200
178    120
```

```
# IQR method for outlier detection
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
outliers = df[((df < lower_bound) | (df > upper_bound)).any(axis=1)]
print("Outliers (IQR method):")
print(outliers)
```

```
156        2   25       1         20              1      4        5   74701
157        2   26       0         21              0      4        3   69721
159        2   27       1         16              1      4        5   83416
160        2   27       1         18              0      4        3   88396
161        2   27       1         21              1      4        4   90886
162        2   28       0         18              1      6        5   92131
163        2   28       1         18              1      7        5   77191
164        2   28       1         18              0      6        5   88396
166        2   29       1         14              1      7        5   85906
167        2   30       0         16              1      6        5   90886
168        2   30       1         18              1      5        4  103336
169        2   30       1         18              1      5        5   99601
170        2   31       1         16              1      6        5   89641
171        2   33       0         18              1      4        5   95866
172        2   34       1         16              0      5        5   92131
173        2   35       1         16              1      4        5   92131
174        2   38       1         18              1      5        5  104581
175        2   40       1         21              0      6        5   83416
176        2   42       1         18              0      5        4   89641
177        2   45       1         16              0      5        5   90886
178        2   47       1         18              1      4        5  104581
```

```
155    240
156    170
157    100
159    160
160    100
161    100
162    180
163    180
164    150
166    300
167    280
168    160
169    150
170    260
171    200
172    150
173    360
174    150
175    200
176    200
177    160
178    120
179    180
```

```
# Create boxplot
plt.figure(figsize=(10, 6))
sns.boxplot(data=df)
plt.title('Boxplot of Data')
plt.show()
```



Start coding or generate with AI.

INCOME LEVEL HAVE MORE OUTLIERS APART FROM THAT NO OTHER DATA HAVE THAT MAJOR OUTLIERS WHERE PEOPLE ABOVE AGE OF >=35 TEND TO MAKE MORE INCOME

Start coding or generate with AI.

Start coding or generate with AI.

```
df.columns
```

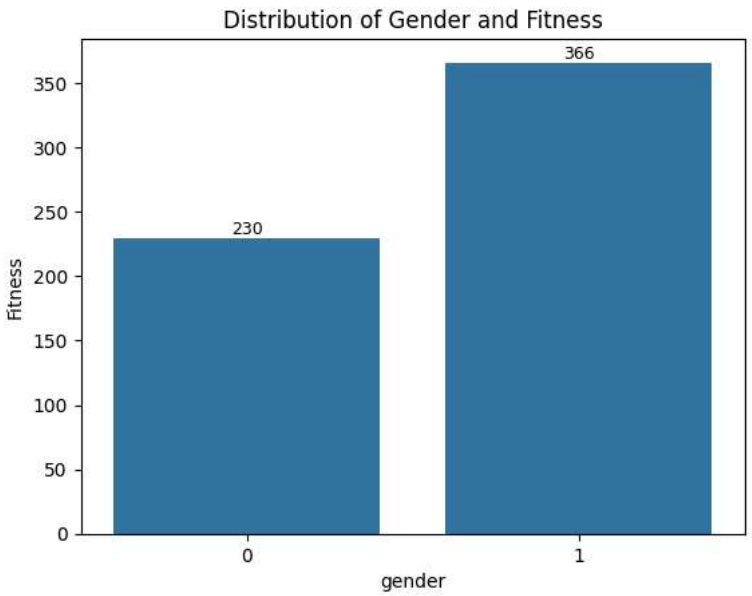
```
Index(['Product', 'Age', 'Gender', 'Education', 'MaritalStatus', 'Usage',
       'Fitness', 'Income', 'Miles'],
```

```
dtype='object')
```

```
df["Fitness"]
```

```
0      4
1      3
2      3
3      3
4      2
      ..
175    5
176    4
177    5
178    5
179    5
Name: Fitness, Length: 180, dtype: int64
```

```
ax2 = sns.barplot(data=df, x="Gender", y="Fitness", estimator=sum, errorbar=None)
ax2.set_title('Distribution of Gender and Fitness')
ax2.bar_label(ax2.containers[0], fontsize=9)
ax2.set_ylabel('Fitness')
ax2.set_xlabel('gender')
plt.show()
```



Double-click (or enter) to edit

MORE NUMBER OF MALES ARE SHOWING INTREST ON THE FİTNESS RATHER THAN THE FEAMLES

```
sns.countplot(data=df, x="Fitness", hue="Gender", palette="Set2")
plt.title('Distribution of Fitness by Gender')
plt.xlabel('Fitness Level')
plt.ylabel('Count')
plt.legend(title='Gender')
plt.show()
```