Business Case: Aerofit - Descriptive Statistics & Probability

```
!gdown 1ht_9NJnHs1mW8t7yAv3CcQ3jiJYxXPdP
```

```
Downloading...
From: https://drive.google.com/uc?id=1ht_9NJnHs1mW8t7yAv3CcQ3jiJYxXPdP
To: /content/aerofit_treadmill.csv
100% 7.28k/7.28k [00:00<00:00, 23.4MB/s]
```

Importing Libraries

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import norm
import math
```

Importing and Analyzing Data

```
df = pd.read_csv('/content/aerofit_treadmill.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

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 180 entries, 0 to 179
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Product        180 non-null    object
 1   Age            180 non-null    int64
 2   Gender         180 non-null    object
 3   Education      180 non-null    int64
 4   MaritalStatus  180 non-null    object
 5   Usage          180 non-null    int64
 6   Fitness        180 non-null    int64
 7   Income         180 non-null    int64
 8   Miles          180 non-null    int64
dtypes: int64(6), object(3)
memory usage: 12.8+ KB
```

```
df.shape
```

```
(180, 9)
```

```
df.describe()
```

	Age	Education	Usage	Fitness	Income	Miles
count	180.000000	180.000000	180.000000	180.000000	180.000000	180.000000
mean	28.788889	15.572222	3.455556	3.311111	53719.577778	103.194444
std	6.943498	1.617055	1.084797	0.958869	16506.684226	51.863605
min	18.000000	12.000000	2.000000	1.000000	29562.000000	21.000000
25%	24.000000	14.000000	3.000000	3.000000	44058.750000	66.000000
50%	26.000000	16.000000	3.000000	3.000000	50596.500000	94.000000
75%	33.000000	16.000000	4.000000	4.000000	58668.000000	114.750000
max	50.000000	21.000000	7.000000	5.000000	104581.000000	360.000000

2. Non-Graphical Analysis: Value counts and unique attributes.

```
# Value counts for unique attributes for 'Age'
age_counts = df['Age'].value_counts()
age_unique = df['Age'].unique()
print(age_counts)
print(age_unique)
```

```
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: Age, dtype: int64
[18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41
 43 44 46 47 50 45 48 42]
```

```
# Value counts and unique attributes for 'Gender'
gender_counts = df['Gender'].value_counts()
unique_genders = df['Gender'].unique()
print(gender_counts)
print(unique_genders)
```

```
Male      104
Female     76
Name: Gender, dtype: int64
['Male' 'Female']
```

```
# Value counts and unique attributes for 'MaritalStatus'
marital_counts = df['MaritalStatus'].value_counts()
unique_marital_status = df['MaritalStatus'].unique()
print(marital_counts)
print(unique_marital_status)
```

```
Partnered    107
Single        73
Name: MaritalStatus, dtype: int64
['Single' 'Partnered']
```

```
# Value counts and unique attributes for 'Product'
product_counts = df['Product'].value_counts()
unique_products = df['Product'].unique()
print(product_counts)
print(unique_products)
```

```
KP281    80
KP481    60
KP781    40
Name: Product, dtype: int64
['KP281' 'KP481' 'KP781']
```

```
# Value counts and unique attributes for 'Education'
education_counts = df['Education'].value_counts()
unique_education = df['Education'].unique()
print(education_counts)
print(unique_education)
```

```
16    85
14    55
18    23
15     5
13     5
12     3
21     3
20     1
Name: Education, dtype: int64
[14 15 12 13 16 18 20 21]
```

3. Visual Analysis - Univariate & Bivariate

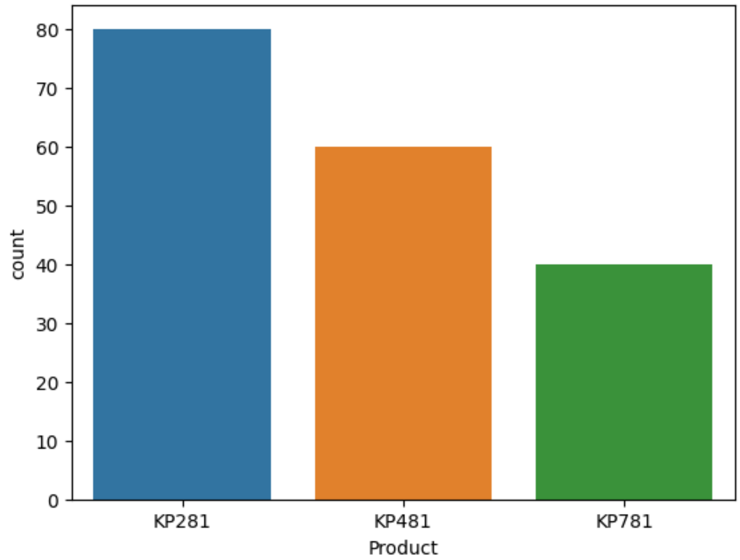
   For continuous variable(s): Distplot, countplot, histogram for univariate analysis

   For categorical variable(s): Boxplot

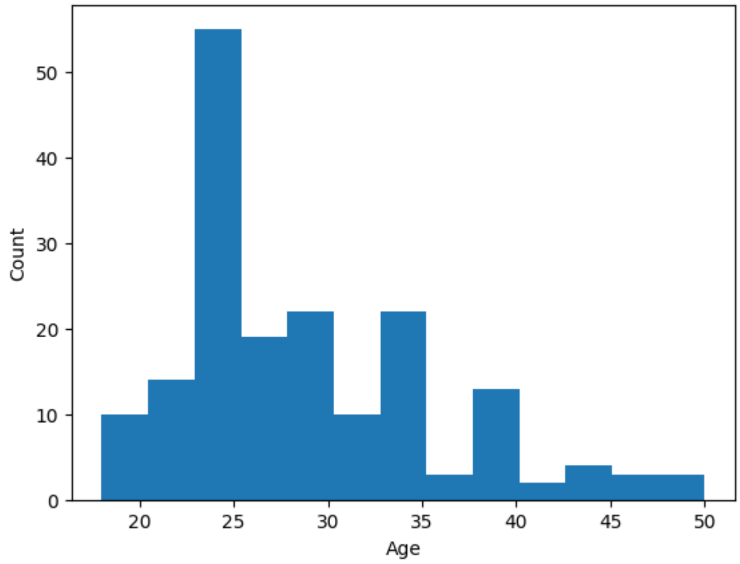
   For correlation: Heatmaps, Pairplots

```
# Univariate analysis on Product
sns.countplot(data=df, x=df['Product'])
#KP281 is being sold more.
```

```
<Axes: xlabel='Product', ylabel='count'>
```

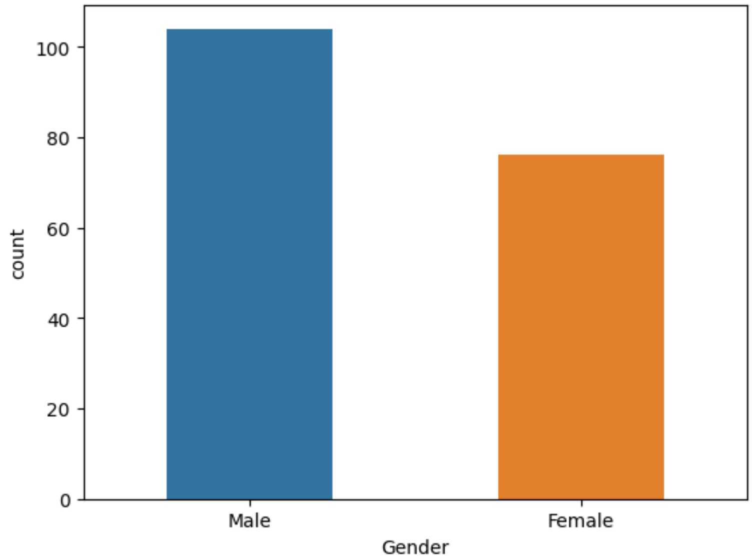



```
#Univariate Analysis on age
num_bins = int(math.sqrt(len(df)))
plt.hist(df['Age'], bins=num_bins)
plt.xlabel("Age")
plt.ylabel("Count")
plt.show()
#Most of the People lie in the age group(22.92307692, 25.38461538)
```

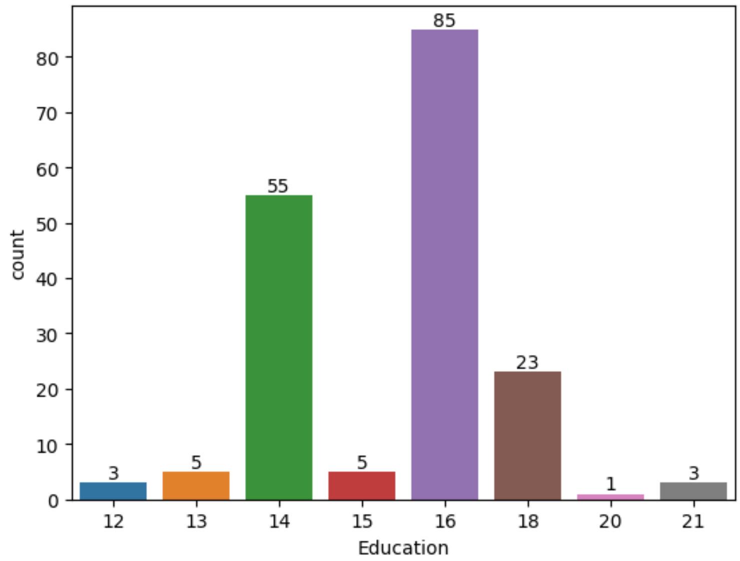


```
sns.countplot(data = df, x = 'Gender', width = 0.5)
```

```
<Axes: xlabel='Gender', ylabel='count'>
```



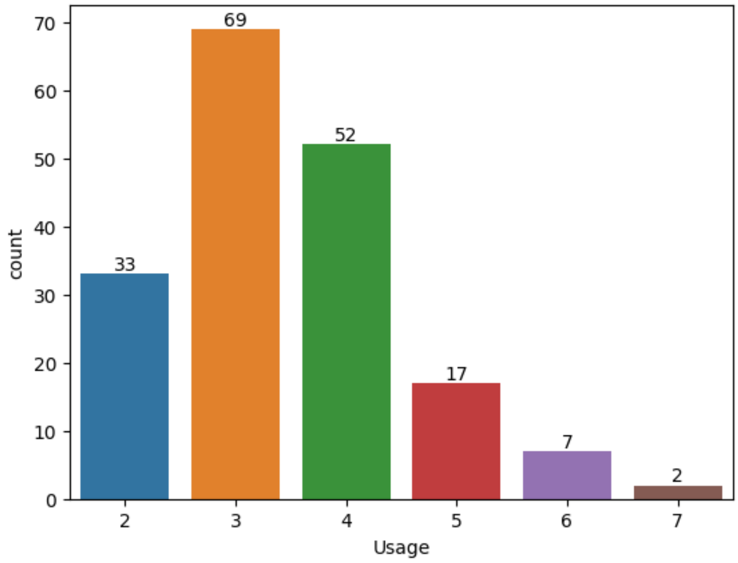

```
#Univariate Analysis on Education
ax = sns.countplot(x=df['Education'])
ax.bar_label(ax.containers[0])
plt.show()
#Most people have 16 years of Education.
```



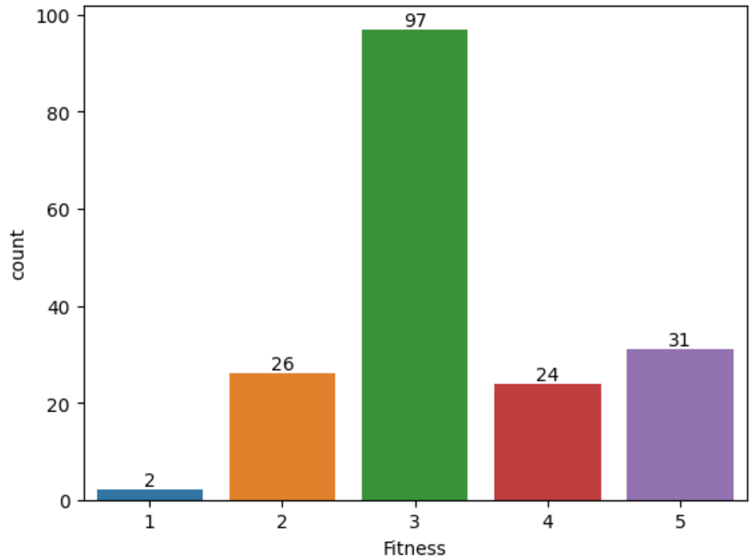

```
# Univariate analysis for MaritalStatus
sns.countplot(data=df, x = 'MaritalStatus', width = 0.25)
```

```
<Axes: xlabel='MaritalStatus', ylabel='count'>
```

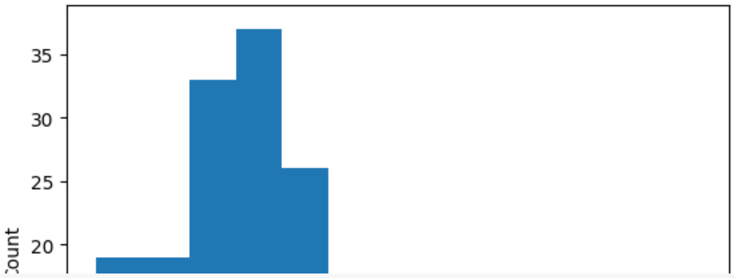
```
#Univariate Analysis on Usage
ax = sns.countplot(x=df['Usage'])
ax.bar_label(ax.containers[0])
plt.show()
#Most of the individuals are using treadmill 3 times a week.
```



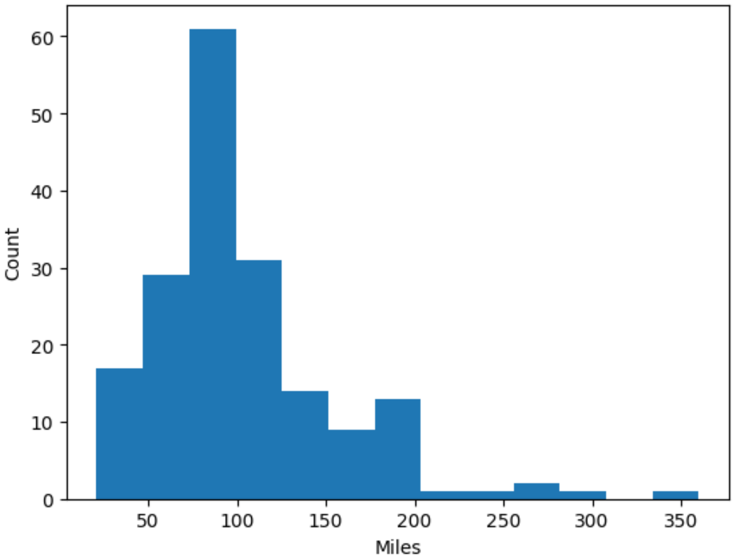

```
#Univariate Analysis on Fitness.
ax = sns.countplot(x=df['Fitness'])
ax.bar_label(ax.containers[0])
plt.show()
```




```
#Univariate Analysis on Income
plt.hist(df['Income'], bins=num_bins)
plt.xlabel("Income")
plt.ylabel("Count")
plt.show()
#Income lie in the range(46874.07692308,52644.76923077)
```

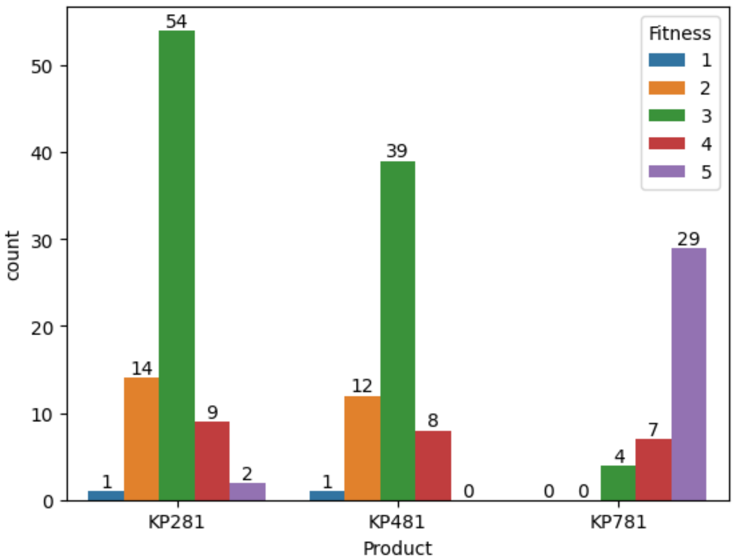



```
#Univariate analysis on miles.
plt.hist(df['Miles'], bins=num_bins)
plt.xlabel("Miles")
plt.ylabel("Count")
plt.show()
#(73.15384615,  99.23076923)
```



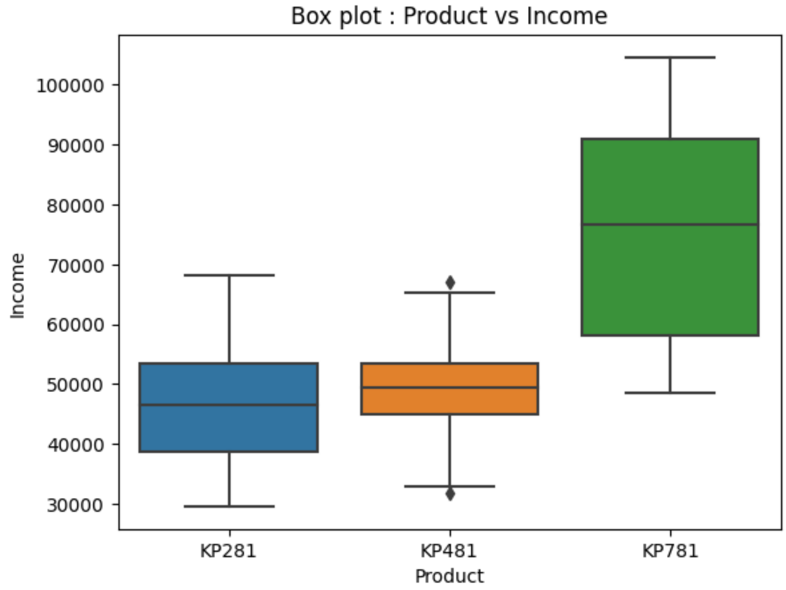
Bivariate Analysis

```
# Bivariate analysis on product via fitness
ax = sns.countplot(data = df, x = 'Product', hue='Fitness')

for container in ax.containers:
    ax.bar_label(container)
plt.show()
```



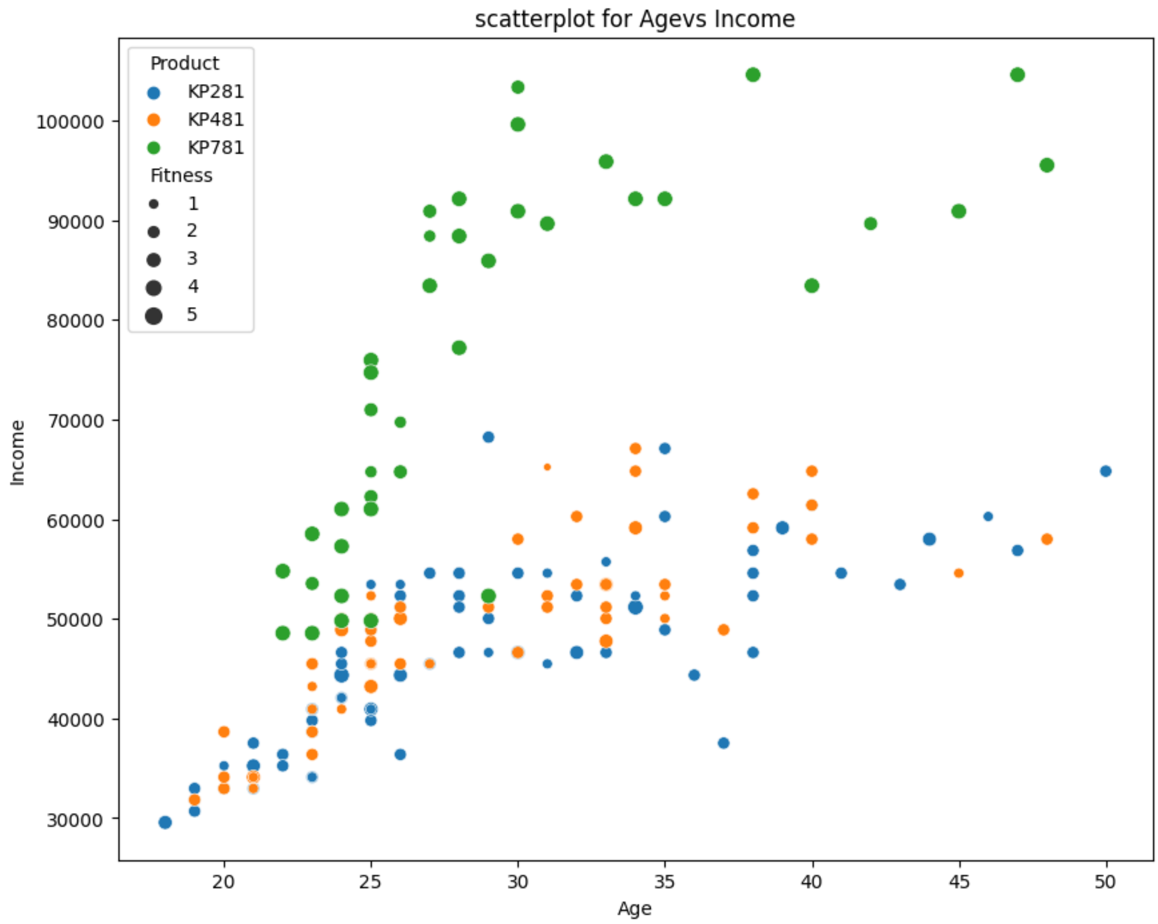

```
# Box plot for product vs income
sns.boxplot(data = df, x=df['Product'], y=df['Income'])
plt.title('Box plot : Product vs Income')
```

```
Text(0.5, 1.0, 'Box plot : Product vs Income')
```



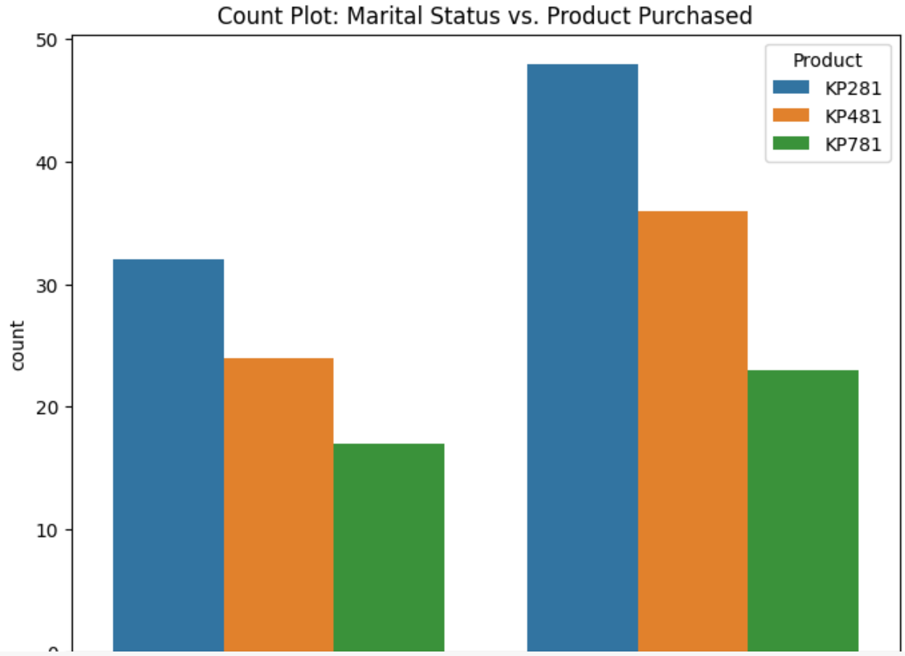

```
# Scatter plot for 'Age' vs. 'Income'
plt.figure(figsize = (10, 8))
sns.scatterplot(data = df, x = 'Age', y = 'Income', hue = 'Product', size = 'Fitness')
plt.title('scatterplot for Agevs Income')
```

```
Text(0.5, 1.0, 'scatterplot for Agevs Income')
```



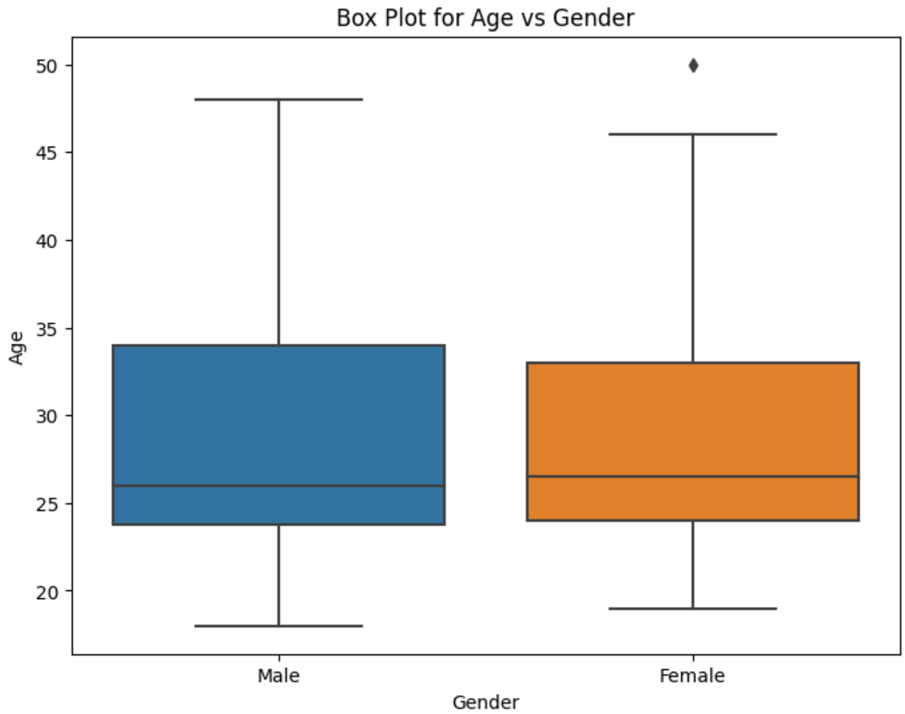

```
#Count Plot for Marital Status and Product Purchased

plt.figure(figsize=(8, 6))
sns.countplot(data=df, x='MaritalStatus', hue='Product')
plt.title('Count Plot: Marital Status vs. Product Purchased')
```

```
Text(0.5, 1.0, 'Count Plot: Marital Status vs. Product Purchased')
```



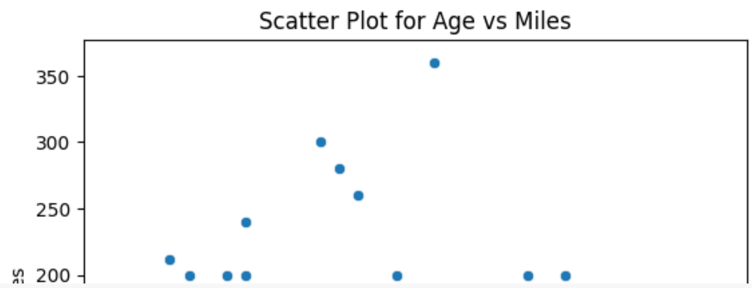

```
# Box plot for 'Age' vs. 'Gender'
plt.figure(figsize=(8, 6))
sns.boxplot(x='Gender', y='Age', data=df)
plt.title('Box Plot for Age vs Gender')
```

```
Text(0.5, 1.0, 'Box Plot for Age vs Gender')
```



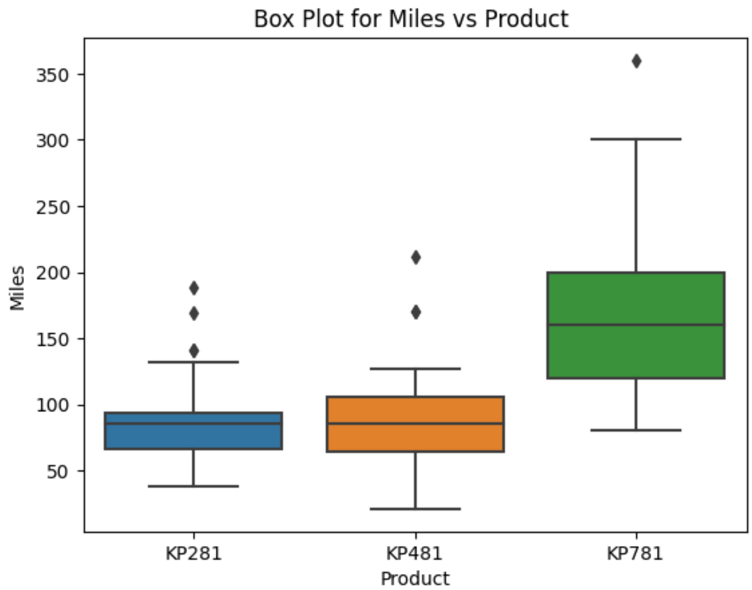

```
# scatterplot for Age vs Miles

sns.scatterplot(data = df, x = 'Age', y = 'Miles')
plt.title('Scatter Plot for Age vs Miles')
```

```
Text(0.5, 1.0, 'Scatter Plot for Age vs Miles')
```




```
# Box plot for 'Miles' vs. 'Product'
sns.boxplot(x='Product', y='Miles', data=df)
plt.title('Box Plot for Miles vs Product ')
```

```
Text(0.5, 1.0, 'Box Plot for Miles vs Product ')
```



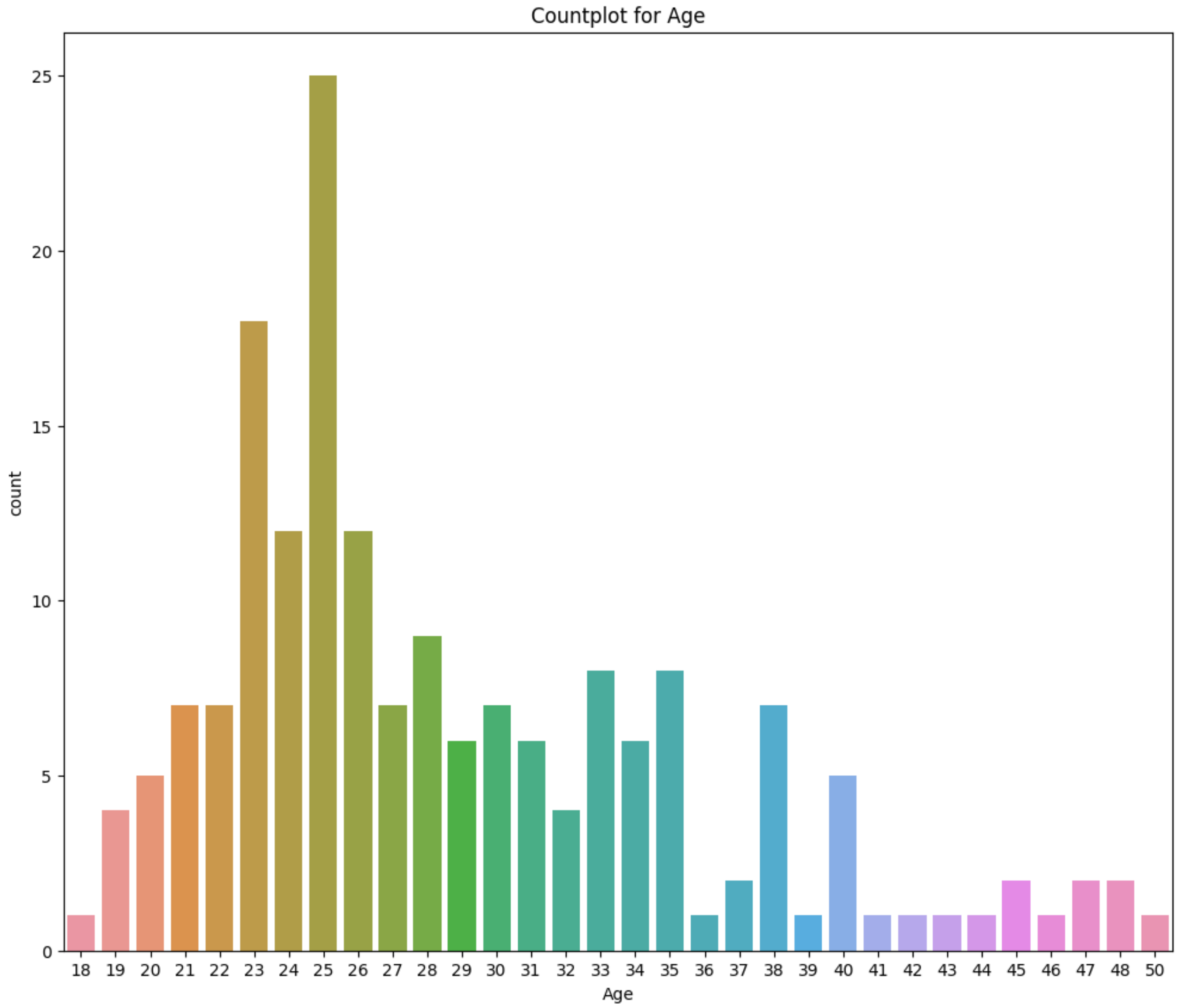
3. 1 For continuous variable(s): Distplot, countplot, histogram for univariate analysis

```
# Univariate Analysis for Continuous Variables
# Distplot for 'Age'
plt.figure(figsize=(8, 6))
sns.distplot(df['Age'])
plt.title('Distplot for Age Distribution')
```

```
<ipython-input-85-4b2d4b9abfb3>:4: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['Age'])
Text(0.5, 1.0, 'Distplot for Age Distribution')
```

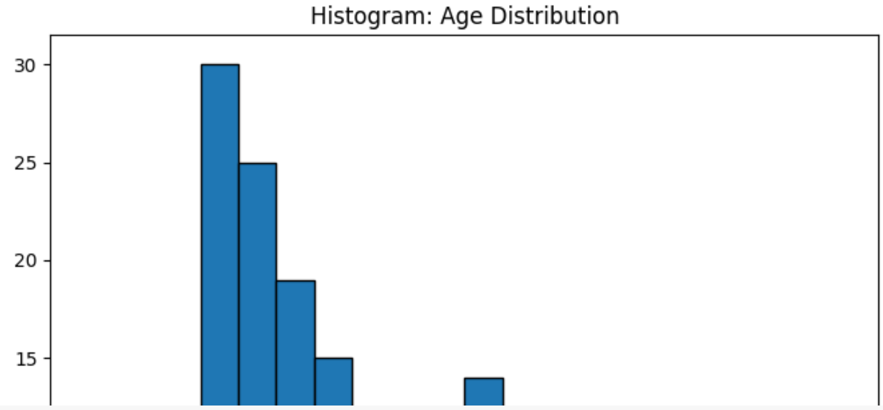
```
# countplot for Age
plt.figure(figsize = (12, 10))
sns.countplot(data = df, x = 'Age')
plt.title('Countplot for Age')
```

```
Text(0.5, 1.0, 'Countplot for Age')
```



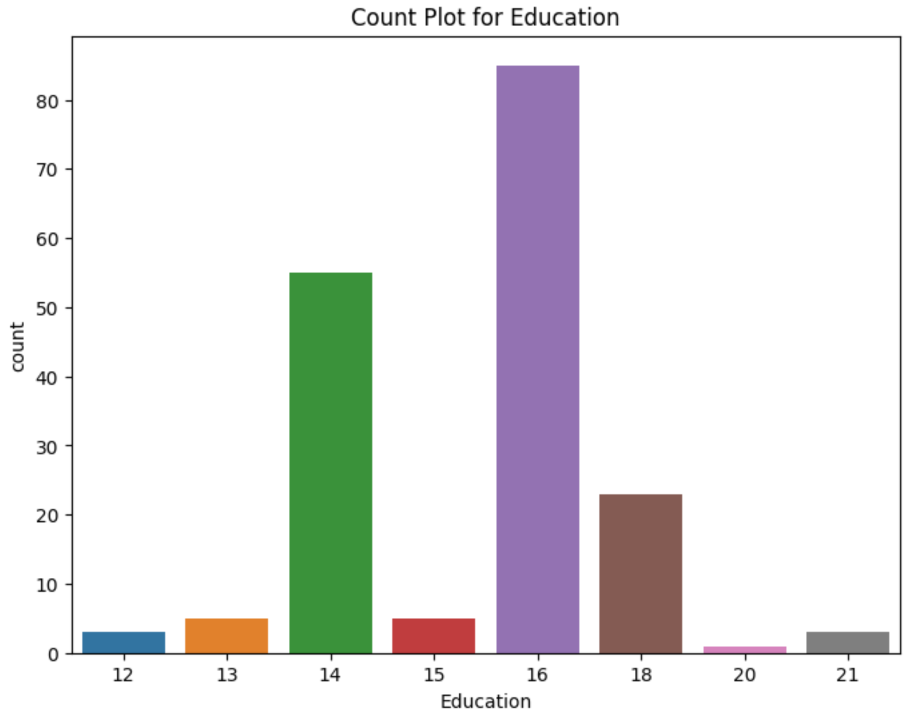

```
#histograph for Age distribution
plt.figure(figsize=(8, 6))
plt.hist(df['Age'], bins=20, edgecolor='black')
plt.title('Histogram: Age Distribution')
```

```
Text(0.5, 1.0, 'Histogram: Age Distribution')
```



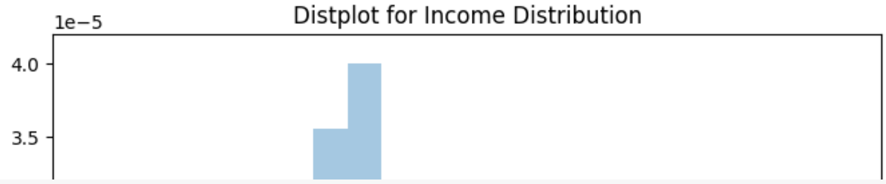

```
# Count Plot for 'Education'
plt.figure(figsize=(8, 6))
sns.countplot(x='Education', data=df)
plt.title('Count Plot: Education')
```

```
Text(0.5, 1.0, 'Count Plot for Education')
```



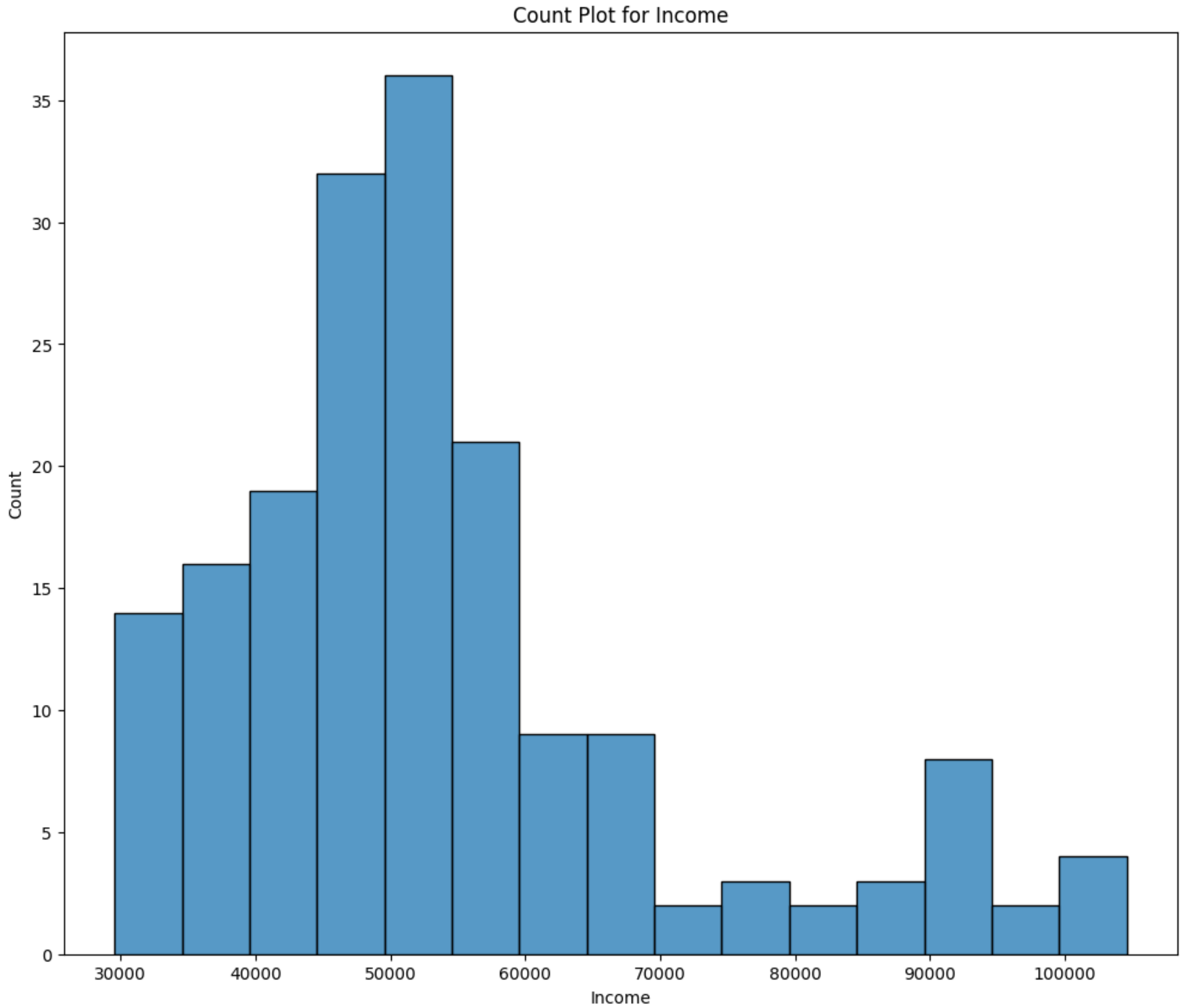

```
# Distplot for Income
plt.figure(figsize=(8, 6))
sns.distplot(df['Income'])
plt.title('Distplot for Income Distribution')
```

```
<ipython-input-98-5944320e4c4f>:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['Income'])
Text(0.5, 1.0, 'Distplot for Income Distribution')
```



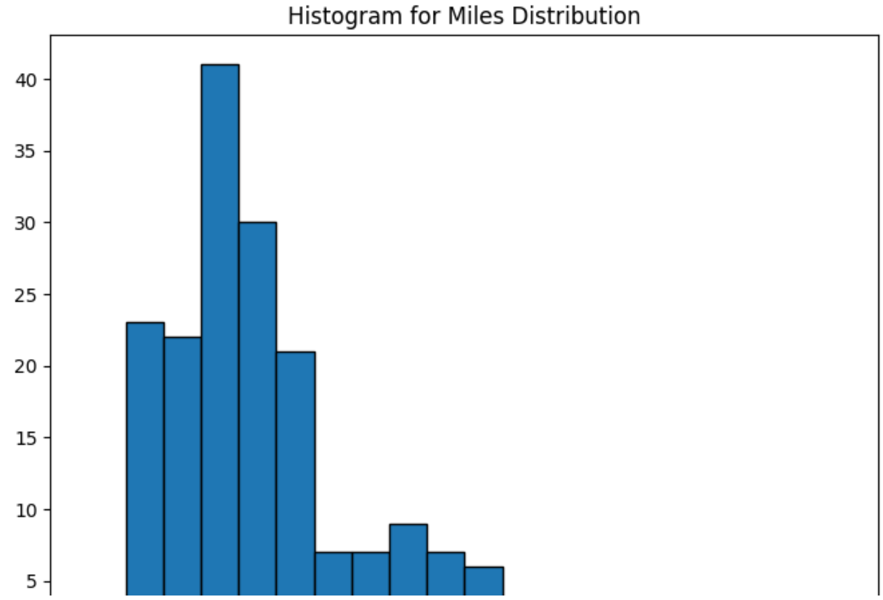

```
# Hist Plot for Income
plt.figure(figsize=(12, 10))
sns.histplot(x='Income', data=df)
plt.title('Hist Plot for Income')
```

```
Text(0.5, 1.0, 'Count Plot for Income')
```




```
# Histogram for 'Miles'
plt.figure(figsize=(8, 6))
plt.hist(df['Miles'], bins=20, edgecolor='black')
plt.title('Histogram for Miles Distribution')
```

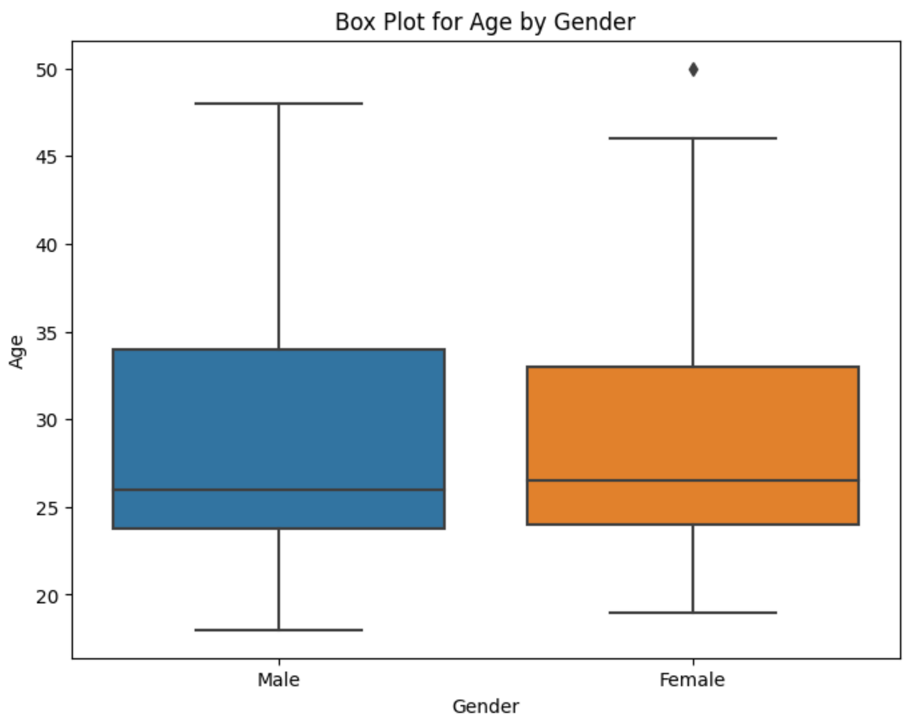
```
Text(0.5, 1.0, 'Histogram for Miles Distribution')
```



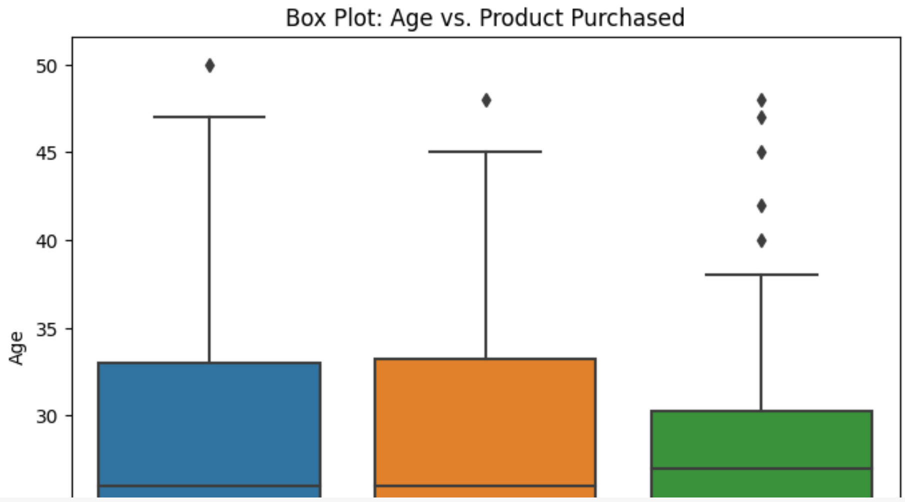
3.2. For categorical variable(s): Boxplot

```
# Boxplot for Gender
plt.figure(figsize=(8, 6))
sns.boxplot(x='Gender', y='Age', data=df)
plt.title('Box Plot for Age by Gender')
```

```
Text(0.5, 1.0, 'Box Plot for Age by Gender')
```

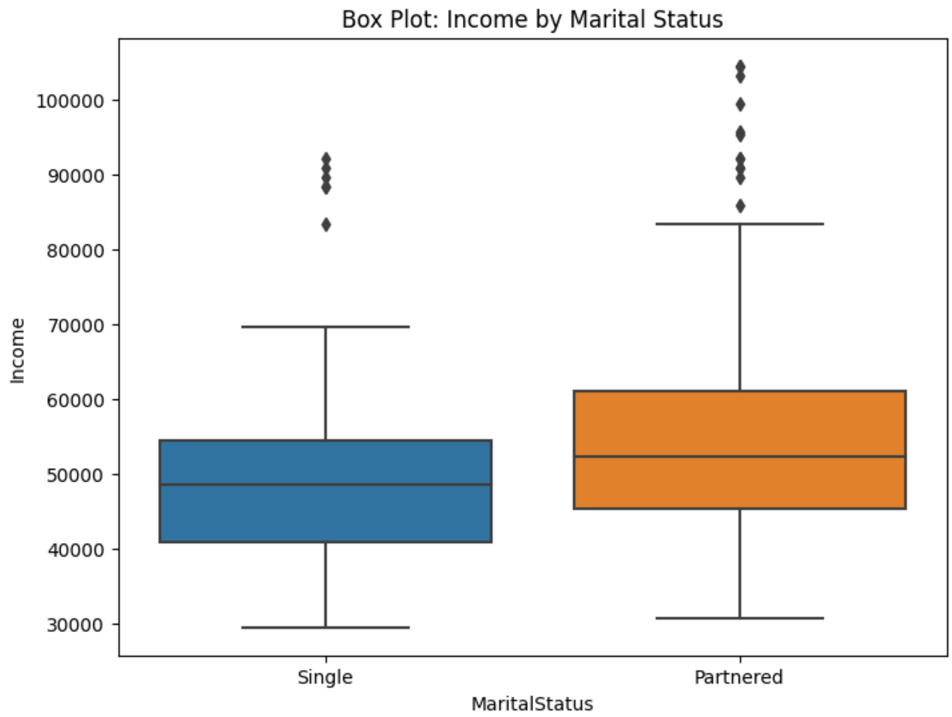



```
# box plot for age and product purchased
plt.figure(figsize=(8, 6))
sns.boxplot(data=df, x='Product', y='Age')
plt.title('Box Plot: Age vs. Product Purchased')
plt.xlabel('Product')
plt.ylabel('Age')
plt.show()
```



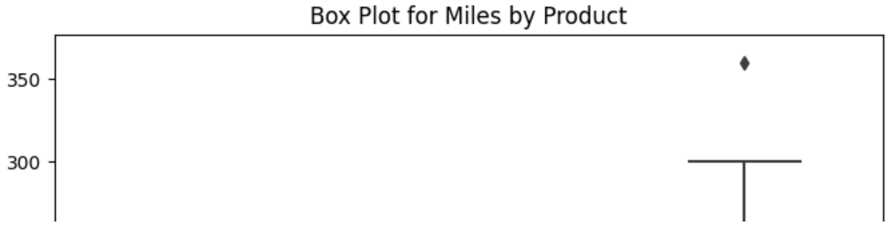

```
# Boxplot for 'MaritalStatus'
plt.figure(figsize=(8, 6))
sns.boxplot(x='MaritalStatus', y='Income', data=df)
plt.title('Box Plot for Income by Marital Status')
```

```
Text(0.5, 1.0, 'Box Plot: Income by Marital Status')
```



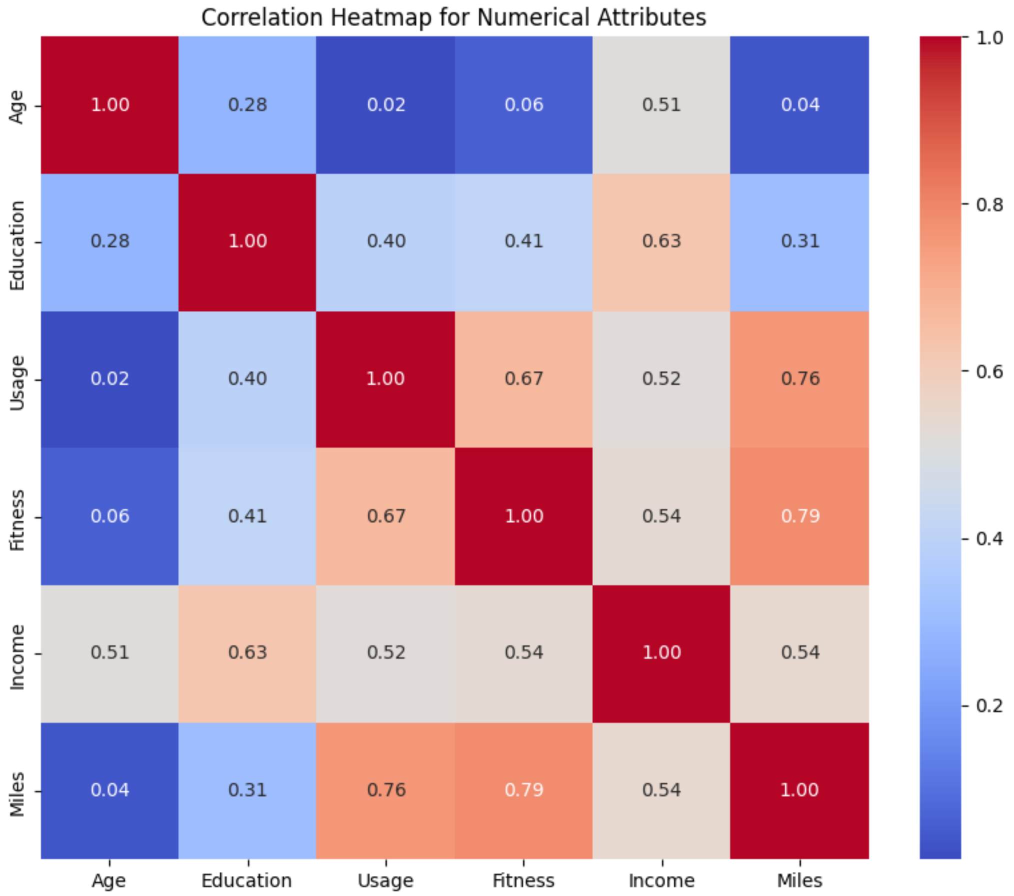

```
# Boxplot for 'Product'
plt.figure(figsize=(8, 6))
sns.boxplot(x='Product', y='Miles', data=df)
plt.title('Box Plot for Miles by Product')
```

```
Text(0.5, 1.0, 'Box Plot for Miles by Product')
```

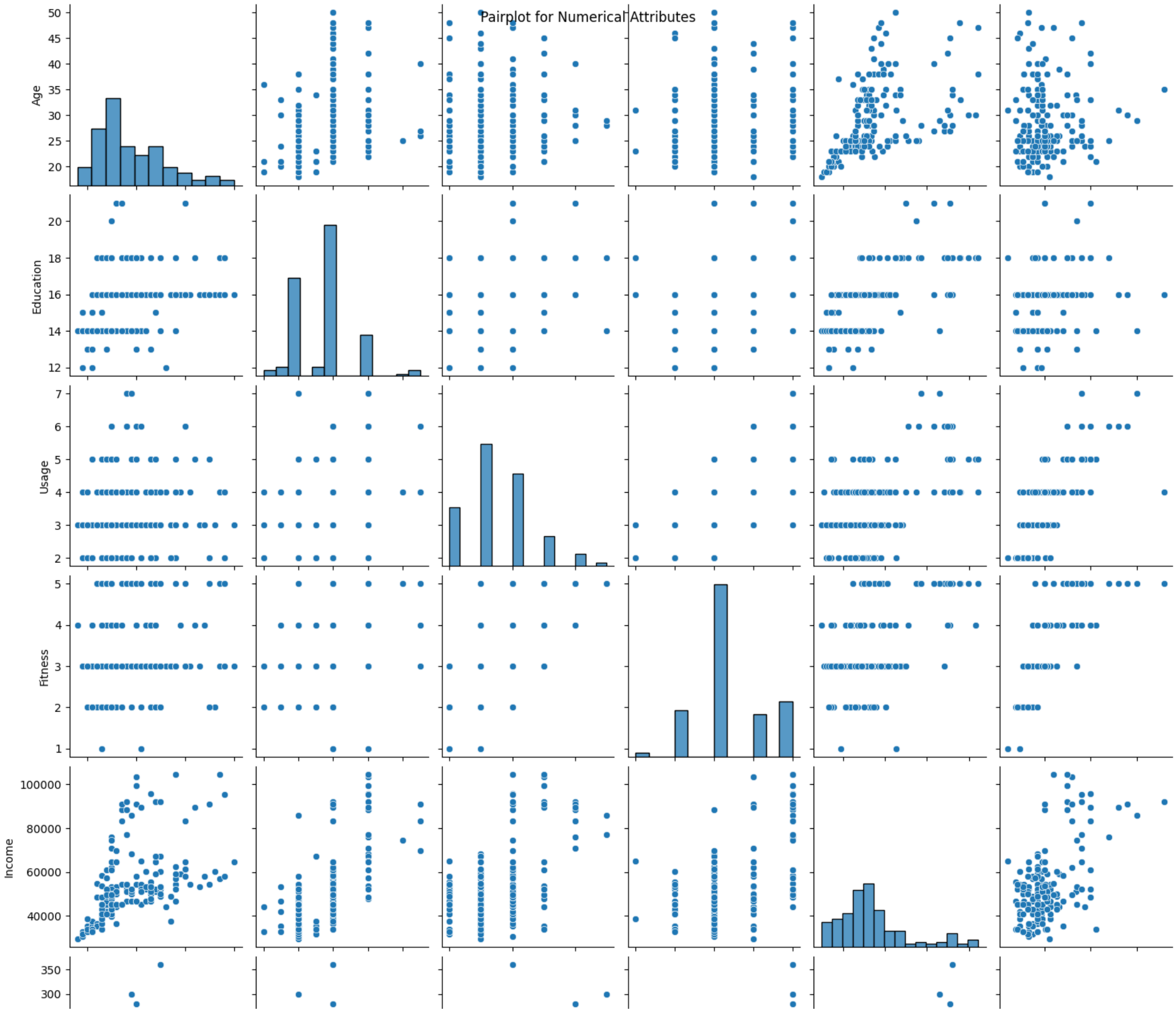


3.3 For correlation: Heatmaps, Pairplots

```
# Correlation heatmap for numerical attributes
numerical_corr_matrix = df.select_dtypes(include='number').corr()

plt.figure(figsize=(10, 8))
sns.heatmap(numerical_corr_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Heatmap for Numerical Attributes')
plt.show()
```




```
# Pairplot for numerical attributes
sns.pairplot(data = df)
plt.suptitle('Pairplot for Numerical Attributes')
plt.show()
```



4. Missing Value & Outlier Detection

```
# Finding missing values
missing_values = df.isnull().sum()
print("Missing Values:")
print(missing_values)

# As we can see that we have zero missing values
```

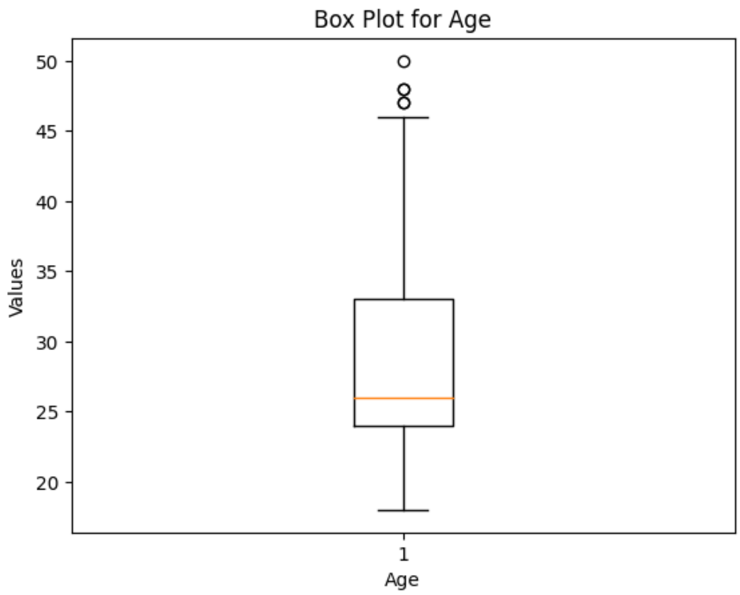
```
Missing Values:
Product          0
Age              0
Gender           0
Education        0
MaritalStatus    0
Usage            0
Fitness          0
Income           0
Miles            0
dtype: int64
```

```
# Outlier Detection using IQR method
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

# Finding outliers
outliers = ((df < lower_bound) | (df > upper_bound)).sum()
print("Outliers:")
print(outliers)

# here we can see that Income has the most outlier followed by Miles and Usage etc.
```

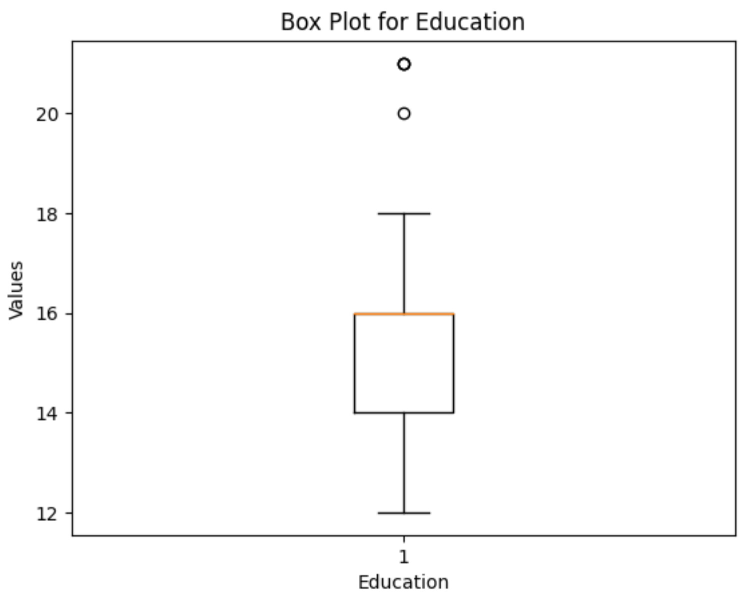
```
Outliers:
Age             5
Education       4
```

```
Fitness           2
Gender            0
Income           19
MaritalStatus     0
Miles            13
Product           0
Usage             9
dtype: int64
<ipython-input-108-0f79082a690c>:2: FutureWarning: The default value of numeric_only in DataFrame.quantile is deprecated. In a futu
  Q1 = df.quantile(0.25)
<ipython-input-108-0f79082a690c>:3: FutureWarning: The default value of numeric_only in DataFrame.quantile is deprecated. In a futu
  Q3 = df.quantile(0.75)
<ipython-input-108-0f79082a690c>:9: FutureWarning: Automatic reindexing on DataFrame vs Series comparisons is deprecated and will r
  outliers = ((df < lower_bound) | (df > upper_bound)).sum()
```

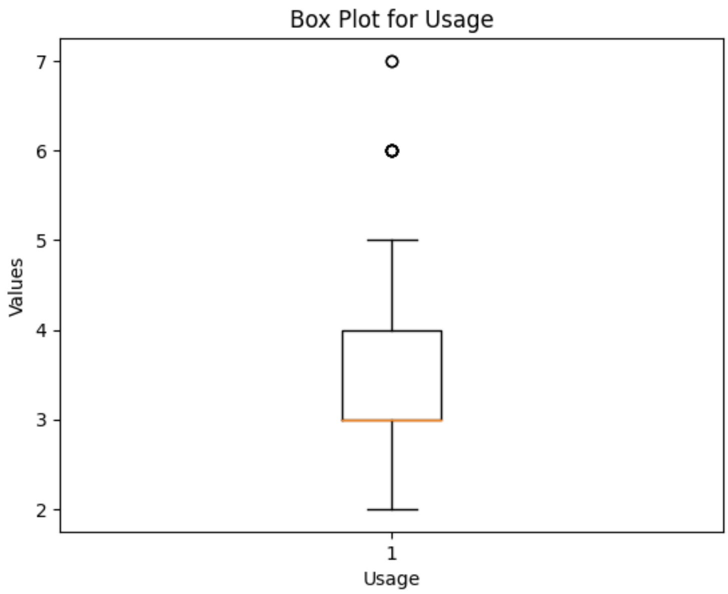
```
# Generate the box plot for the 'Age' column
plt.boxplot(df['Age'])
plt.xlabel('Age')
plt.ylabel('Values')
plt.title('Box Plot for Age')
plt.show()
```



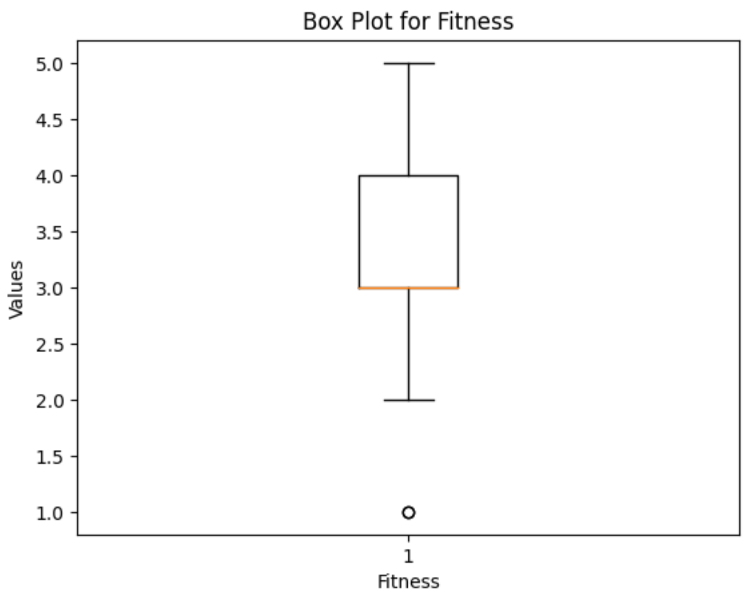

```
# Generate the box plot for the 'Education' column
plt.boxplot(df['Education'])
plt.xlabel('Education')
plt.ylabel('Values')
plt.title('Box Plot for Education')
plt.show()
```



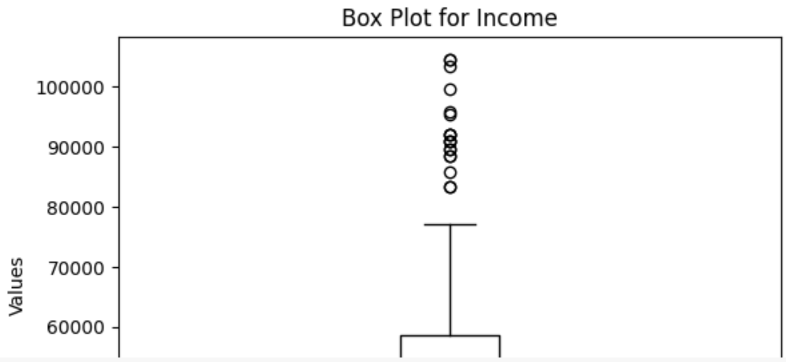
```
# Generate the box plot for the 'Usage' column
plt.boxplot(df['Usage'])
plt.xlabel('Usage')
plt.ylabel('Values')
plt.title('Box Plot for Usage')
plt.show()
```



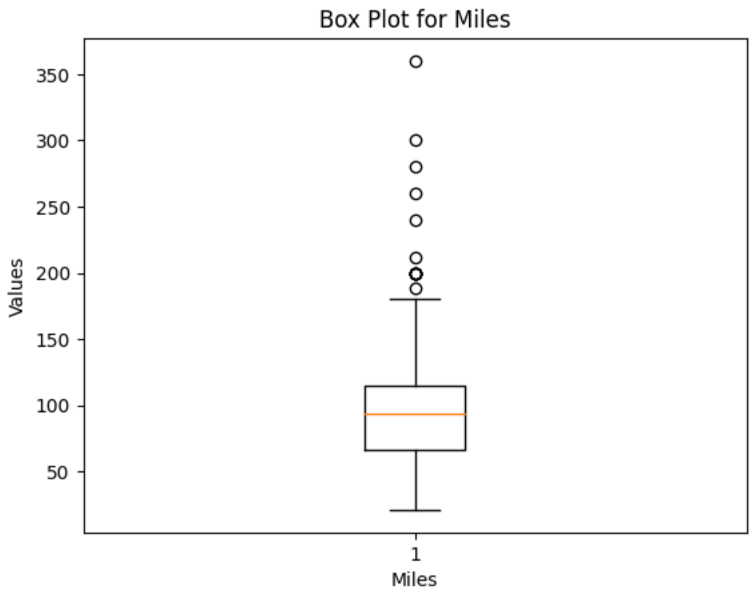

```
# Generate the box plot for the 'Fitness' column
plt.boxplot(df['Fitness'])
plt.xlabel('Fitness')
plt.ylabel('Values')
plt.title('Box Plot for Fitness')
plt.show()
```




```
# Generate the box plot for the 'Income' column
plt.boxplot(df['Income'])
plt.xlabel('Income')
plt.ylabel('Values')
plt.title('Box Plot for Income')
plt.show()
```

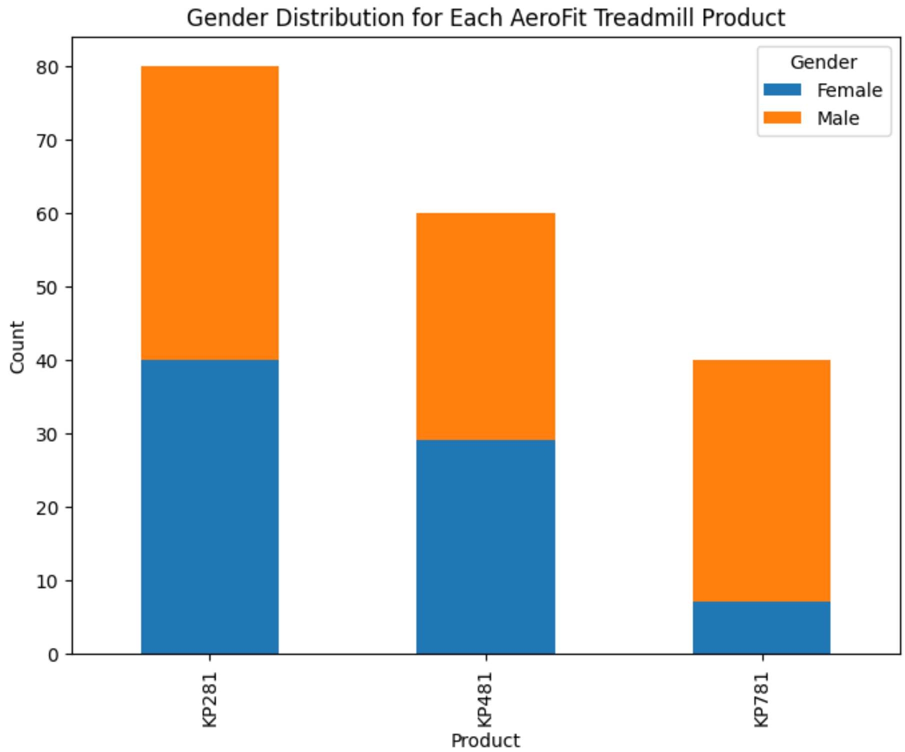


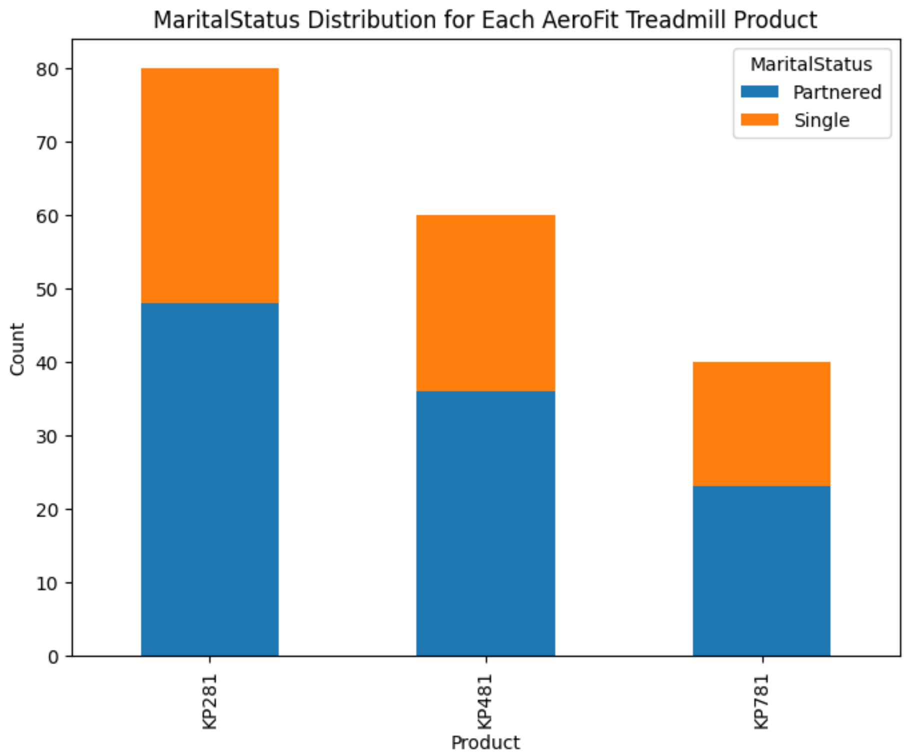

```
# Generate the box plot for the 'Miles' column
plt.boxplot(df['Miles'])
plt.xlabel('Miles')
plt.ylabel('Values')
plt.title('Box Plot for Miles')
plt.show()
```

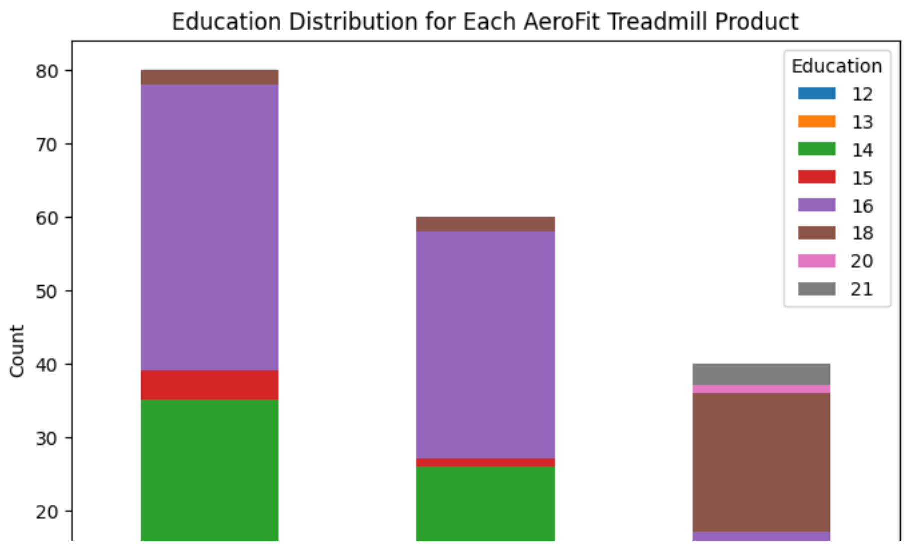


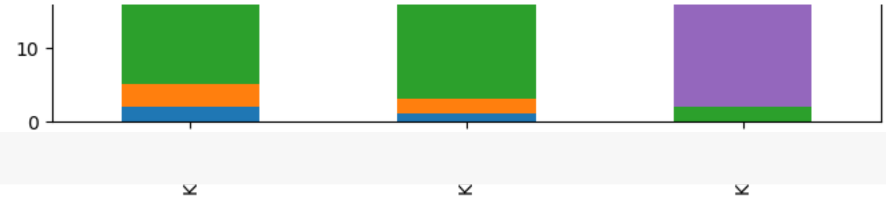
Perform descriptive analytics to create a customer profile for each AeroFit treadmill product by developing appropriate tables and charts.

```
# Customer profile for each AeroFit treadmill product
product_groups = df.groupby('Product')

# Gender distribution for each product
gender_counts = product_groups['Gender'].value_counts().unstack()
gender_counts.plot(kind='bar', stacked=True, figsize=(8, 6))
plt.title('Gender Distribution for Each AeroFit Treadmill Product')
plt.xlabel('Product')
plt.ylabel('Count')
plt.show()

# MaritalStatus distribution for each product
marital_counts = product_groups['MaritalStatus'].value_counts().unstack()
marital_counts.plot(kind='bar', stacked=True, figsize=(8, 6))
plt.title('MaritalStatus Distribution for Each AeroFit Treadmill Product')
plt.xlabel('Product')
plt.ylabel('Count')
plt.show()

# Education distribution for each product
education_counts = product_groups['Education'].value_counts().unstack()
education_counts.plot(kind='bar', stacked=True, figsize=(8, 6))
plt.title('Education Distribution for Each AeroFit Treadmill Product')
plt.xlabel('Product')
plt.ylabel('Count')
plt.show()
```


Gender Distribution for Each AeroFit Treadmill Product
Gender
Female
Male
Count
Product
KP281
KP481
KP781


MaritalStatus Distribution for Each AeroFit Treadmill Product
MaritalStatus
Partnered
Single
Count
Product
KP281
KP481
KP781


Education Distribution for Each AeroFit Treadmill Product
Education
12
13
14
15
16
18
20
21
Count



For each AeroFit treadmill product, construct two-way contingency tables and compute all conditional and marginal probabilities along with their insights/impact on the business

```
# Create two-way contingency tables and compute probabilities for each product
product_groups = df.groupby('Product')

for product, group in product_groups:
    # Two-way contingency table for Gender and MaritalStatus
    contingency_table = pd.crosstab(group['Gender'], group['MaritalStatus'], margins=True, margins_name='Total')
    print(f"\nProduct: {product}")
    print("Contingency Table for Gender vs. MaritalStatus:")
    print(contingency_table)

    # Conditional probabilities of MaritalStatus given Gender
    conditional_probs = contingency_table.div(contingency_table['Total'], axis=0)
    print("\nConditional Probabilities (MaritalStatus | Gender):")
    print(conditional_probs)

    # Marginal probabilities of Gender and MaritalStatus
    marginal_probs = contingency_table.div(contingency_table.loc['Total'], axis=1)
    print("\nMarginal Probabilities (Gender, MaritalStatus):")
    print(marginal_probs)
```

```
Product: KP281
Contingency Table for Gender vs. MaritalStatus:
MaritalStatus  Partnered  Single  Total
Gender
Female                27      13     40
Male                  21      19     40
Total                 48      32     80

Conditional Probabilities (MaritalStatus | Gender):
MaritalStatus  Partnered  Single  Total
Gender
Female             0.675   0.325    1.0
Male               0.525   0.475    1.0
Total              0.600   0.400    1.0

Marginal Probabilities (Gender, MaritalStatus):
MaritalStatus  Partnered   Single  Total
Gender
Female            0.5625  0.40625    0.5
Male              0.4375  0.59375    0.5
Total             1.0000  1.00000    1.0

Product: KP481
Contingency Table for Gender vs. MaritalStatus:
MaritalStatus  Partnered  Single  Total
Gender
Female                15      14     29
Male                  21      10     31
Total                 36      24     60

Conditional Probabilities (MaritalStatus | Gender):
MaritalStatus  Partnered    Single  Total
Gender
Female          0.517241  0.482759    1.0
Male            0.677419  0.322581    1.0
Total           0.600000  0.400000    1.0

Marginal Probabilities (Gender, MaritalStatus):
MaritalStatus  Partnered    Single     Total
Gender
Female          0.416667  0.583333  0.483333
Male            0.583333  0.416667  0.516667
Total           1.000000  1.000000  1.000000

Product: KP781
Contingency Table for Gender vs. MaritalStatus:
MaritalStatus  Partnered  Single  Total
Gender
Female                 4       3      7
Male                  19      14     33
Total                 23      17     40
```

```
Conditional Probabilities (MaritalStatus | Gender):
MaritalStatus  Partnered    Single  Total
Gender
Female          0.571429  0.428571    1.0
```

```
# Create the contingency table for Product and Gender
contingency_table = pd.crosstab(index=df['Product'], columns=df['Gender'], margins=True, margins_name='Total', normalize='index') * 100

# Display the marginal probabilities table
print("Marginal Probability Table (Percentage of Customers by Product and Gender):")
print(contingency_table)

# we can clearly see that 'Male' used more treadmill as compared to 'Female'
```

```
Marginal Probability Table (Percentage of Customers by Product and Gender):
Gender      Female       Male
Product
KP281    50.000000  50.000000
KP481    48.333333  51.666667
KP781    17.500000  82.500000
Total    42.222222  57.777778
```

5. Business Insights based on Non-Graphical and Visual Analysis

Comments on the range of attributes

Comments on the distribution of the variables and relationship between them

Comments for each univariate and bivariate plot

Comments on the range of attributes

```
# Age: #Most of the People lie in the age group(22.92307692, 25.38461538)
# Gender: AeroFit's customer base comprises both males and females.
# Education: The education levels of customers range from 12 to 21 years of schooling.
# Marital Status: Customers in the dataset are classified as either "Single" or "Partnered."
# Usage: The usage levels of customers' treadmill range from 2 to 6.
# Fitness: The fitness levels of customers range from 2 to 5.
# Income: The income levels of customers range from 29,562 to 104,581.
# Miles: The distance covered (in miles) on the treadmill ranges from a few miles to over 200 miles.
# Product: The dataset includes three different treadmill products: KP281, KP481, and KP781.
```

Comments on the distribution of the variables and relationship between them

Comments for each univariate and bivariate plot

```
# Age Distribution: The age distribution shows a roughly normal distribution, with a higher concentration between the age group (22.92307
# Gender Distribution: The dataset exhibits a balanced gender distribution, with a slightly higher number of male customers.
# Education Distribution: The education distribution shows a diverse range of educational backgrounds,
  # indicating that AeroFit's products are used by customers with different levels of education.
# MaritalStatus Distribution: The dataset contains a higher number of partnered customers compared to single customers.
# Usage and Fitness Relationship: There is a positive correlation between usage and fitness levels.
# Age and Income Relationship: There is a moderate positive correlation between age and income. As customers get older, their incomes ter
# Marital Status and Income Relationship: Partnered customers generally have higher incomes than single customers.
# Product and Miles Relationship: KP281 has the highest median miles, indicating it is being used more frequently compared to the other p
# Fitness and Miles Relationship: Customers with higher fitness levels tend to cover more miles on the treadmill.
```

6. Recommendations (10 Points) - Actionable items for business. No technical jargon. No complications. Simple action items that everyone can understand

```
# Personalized Workout Plans: Offer tailored workout plans to cater to individual fitness goals.
# Targeted Marketing for Age Groups: Develop targeted marketing for different age segments to attract diverse customers.
# Inclusive Marketing: Ensure marketing materials are inclusive, appealing to all genders and maritalstatuses.
# Affordable Pricing Options: Introduce flexible pricing to accommodate customers with varying budgets.
# Promote KP281: Highlight the popular KP281 treadmill to drive more sales.
# Promote KP781: Highlight this to because it gives the most income as compared to other treadmills.
# Partnered Customer Loyalty: Create loyalty programs for partnered customers to encourage repeat purchases.
# Customer Feedback for Improvements: Use customer feedback to enhance products and services.
# Engage on Social Media: Connect with customers on social platforms for community building.
# In-Store Demos: Allow in-store treadmill demos to encourage customer engagement.
# Responsive Customer Support: Provide excellent customer support for a positive experience.
```