BUSINESS CASE: AEROFIT- DESCRIPTIVE STATISTICS AND PROBABILITY

```
!pip install pandas
```

```
Requirement already satisfied: pandas in e:\rasa\lib\site-packages
(2.2.0)
Requirement already satisfied: numpy<2,>=1.23.2 in e:\rasa\lib\site-
packages (from pandas) (1.24.3)
Requirement already satisfied: python-dateutil>=2.8.2 in e:\rasa\lib\
site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in e:\rasa\lib\site-
packages (from pandas) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.7 in e:\rasa\lib\site-
packages (from pandas) (2023.3)
Requirement already satisfied: six>=1.5 in e:\rasa\lib\site-packages
(from python-dateutil>=2.8.2->pandas) (1.16.0)
```

```
import numpy as np
import pandas as pd
import math as m
```

Observations on Dataset

```
#importing the dataset
data = pd.read_csv('G:/dsml-scaler/probability and
stats/casestudy/aerofit_treadmill.csv')
```

```
data.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
```

```
print("Shape of the dataset:", data.shape)
```

```
Shape of the dataset: (180, 9)
```

```
print("\nData types of all attributes:")
print(data.dtypes)
```

```
Data types of all attributes:
Product          object
Age               int64
Gender           object
Education         int64
MaritalStatus    object
Usage             int64
Fitness           int64
Income            int64
Miles             int64
dtype: object
```

```
data.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
```

```
# Convert categorical attributes to 'category'
categorical_columns = ['Product', 'Gender', 'MaritalStatus']
for col in categorical_columns:
    data[col] = data[col].astype('category')

# Observations after converting categorical attributes to 'category'
print("\nData types after converting categorical attributes to
'category':")
print(data.dtypes)
```

```
Data types after converting categorical attributes to 'category':
Product          category
Age                 int64
Gender           category
Education           int64
MaritalStatus    category
Usage               int64
Fitness             int64
```

```
Income                 int64
Miles                  int64
dtype: object
```

```python
# Statistical summary
print("\nStatistical summary:")
print(data.describe(include='all'))
```

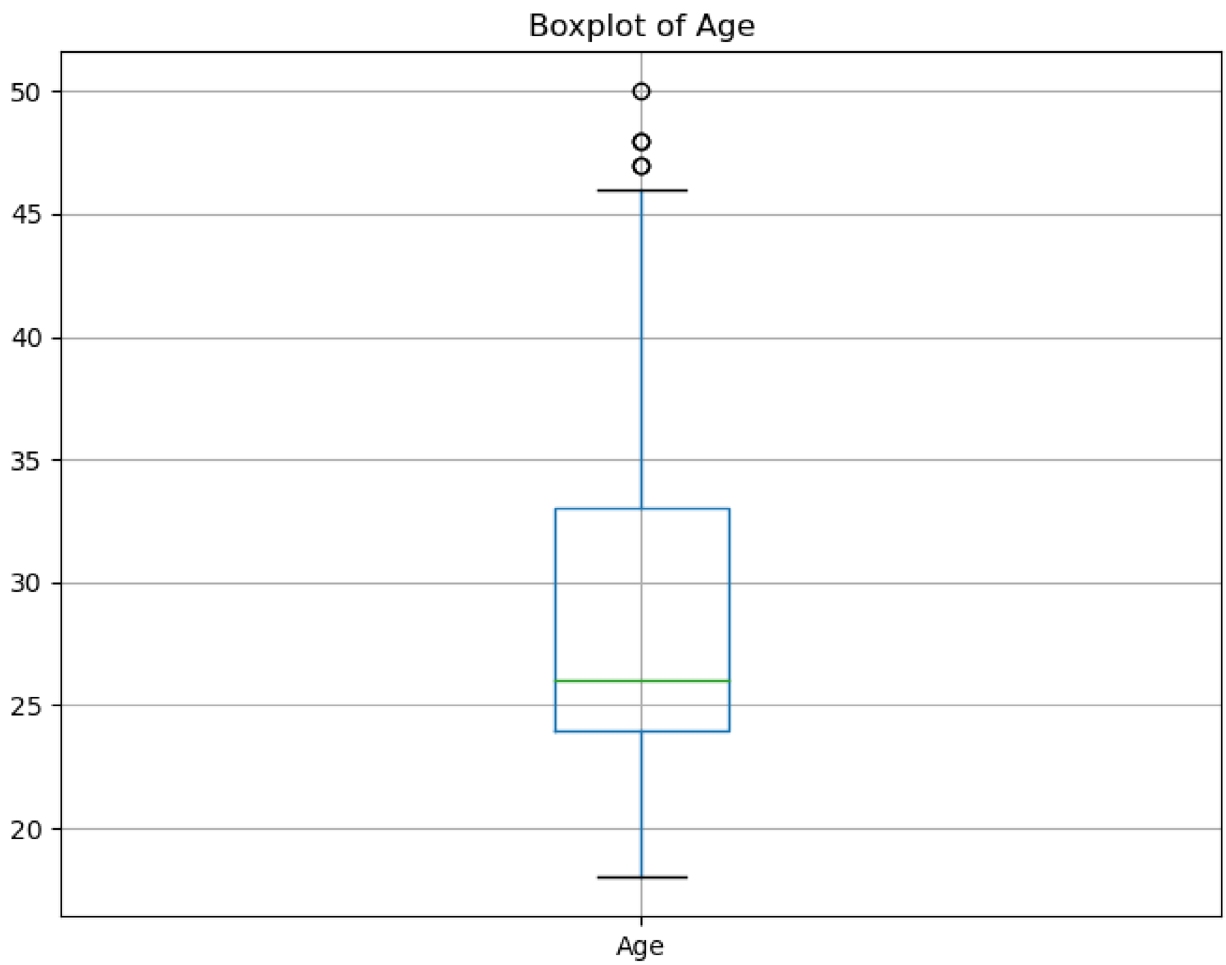
```
Statistical summary:
       Product         Age Gender   Education MaritalStatus
Usage  \
count      180  180.000000    180  180.000000           180
180.000000
unique       3         NaN      2         NaN             2
NaN
top      KP281         NaN   Male         NaN     Partnered
NaN
freq        80         NaN    104         NaN           107
NaN
mean       NaN   28.788889    NaN   15.572222           NaN
3.455556
std        NaN    6.943498    NaN    1.617055           NaN
1.084797
min        NaN   18.000000    NaN   12.000000           NaN
2.000000
25%        NaN   24.000000    NaN   14.000000           NaN
3.000000
50%        NaN   26.000000    NaN   16.000000           NaN
3.000000
75%        NaN   33.000000    NaN   16.000000           NaN
4.000000
max        NaN   50.000000    NaN   21.000000           NaN
7.000000

           Fitness         Income       Miles
count   180.000000     180.000000  180.000000
unique         NaN            NaN         NaN
top            NaN            NaN         NaN
freq           NaN            NaN         NaN
mean      3.311111   53719.577778  103.194444
std       0.958869   16506.684226   51.863605
min       1.000000   29562.000000   21.000000
25%       3.000000   44058.750000   66.000000
50%       3.000000   50596.500000   94.000000
75%       4.000000   58668.000000  114.750000
max       5.000000  104581.000000  360.000000
```

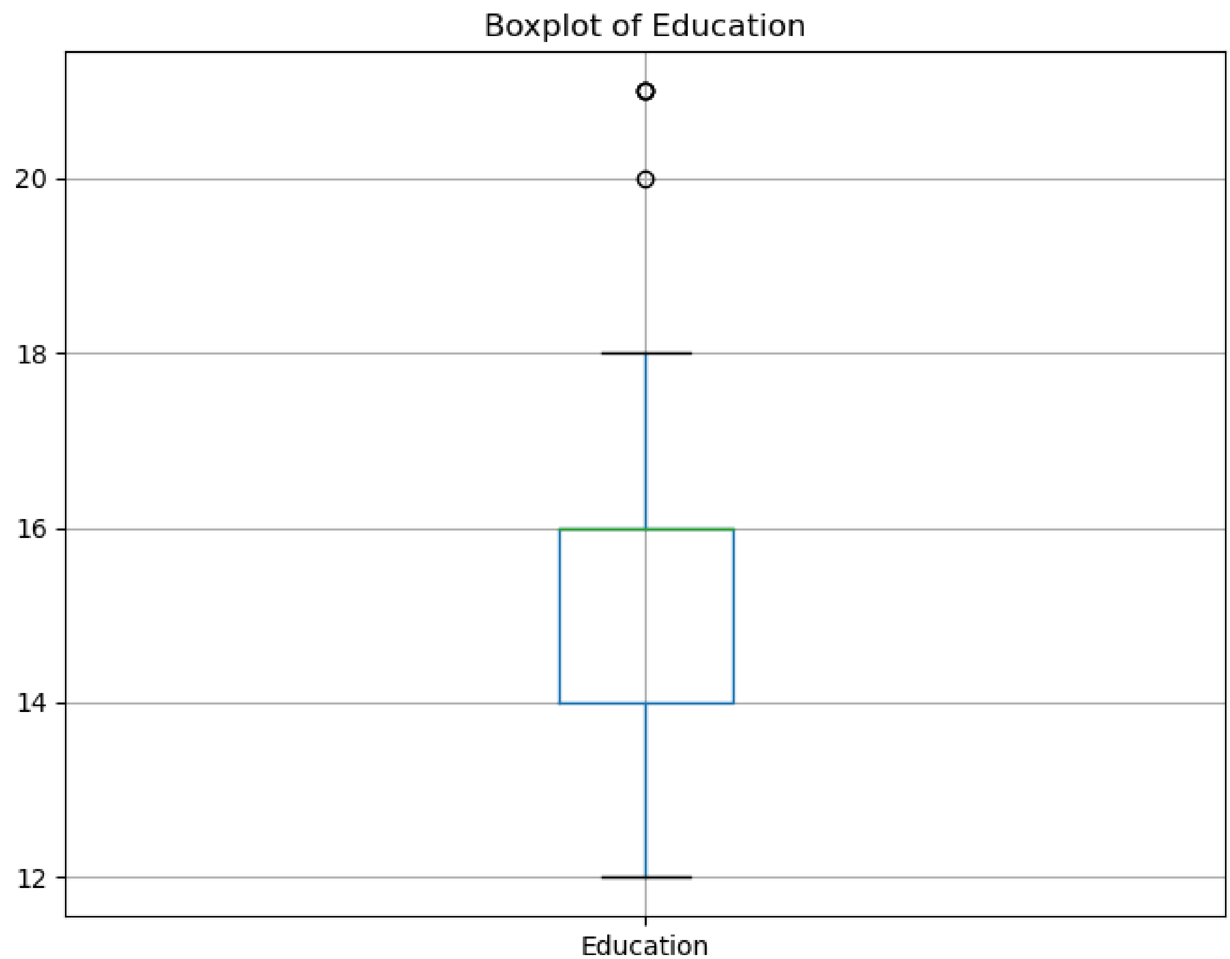
Missing Values and Outliers

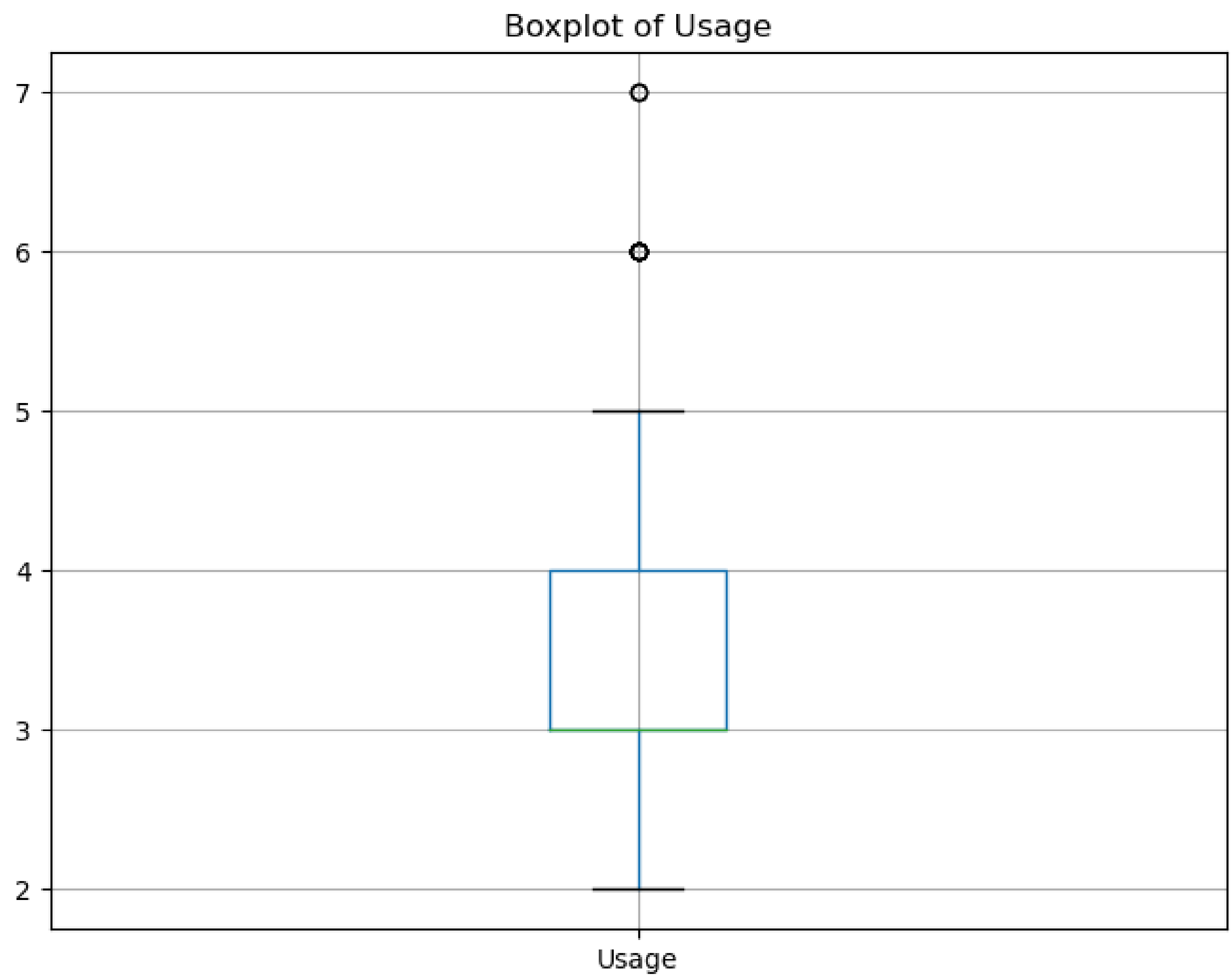
```
# Check for missing values
missing_values = data.isnull().sum()
print("Missing Values:")
print(missing_values)
```

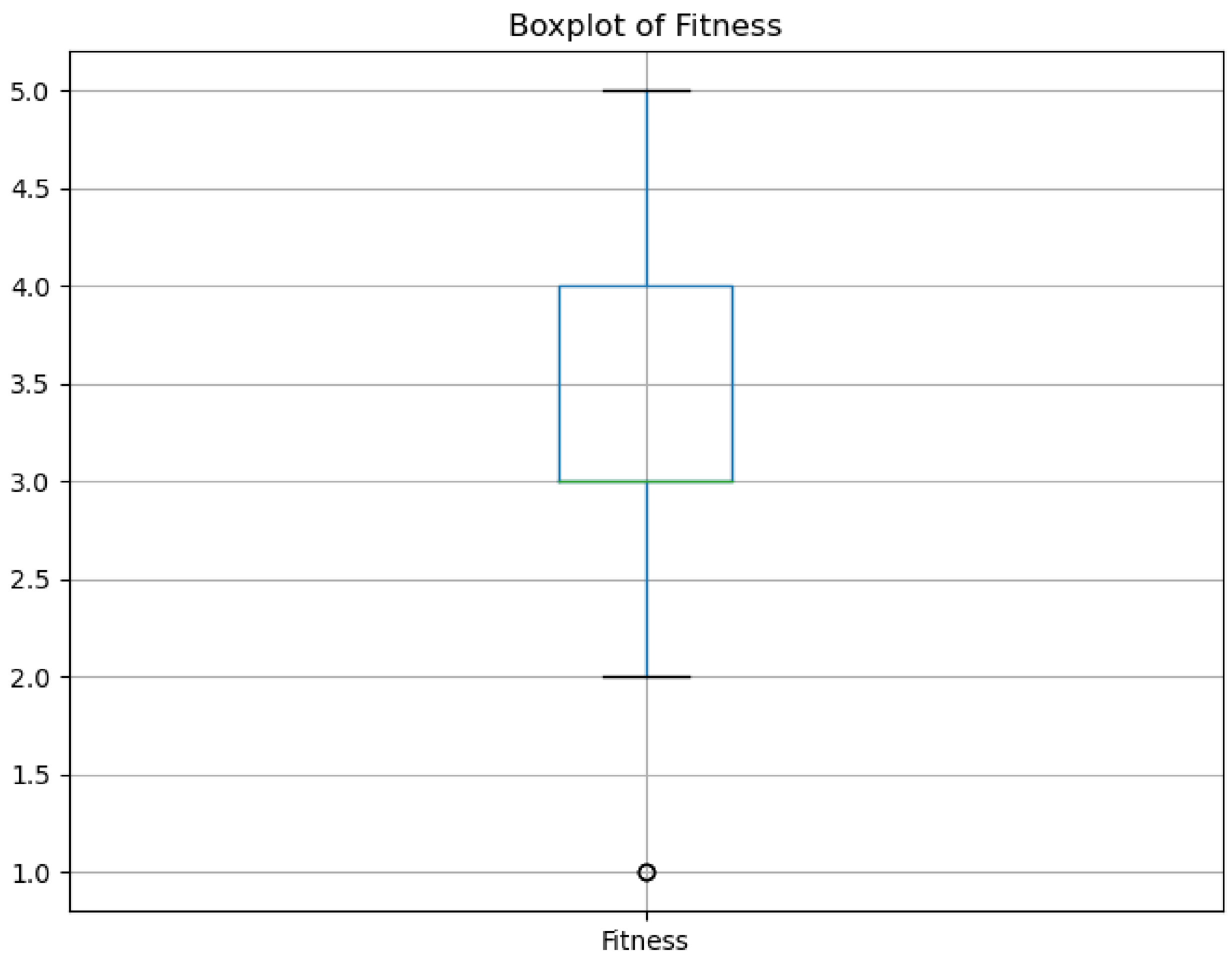
```
Missing Values:
Product          0
Age              0
Gender           0
Education        0
MaritalStatus    0
Usage            0
Fitness          0
Income           0
Miles            0
dtype: int64
```

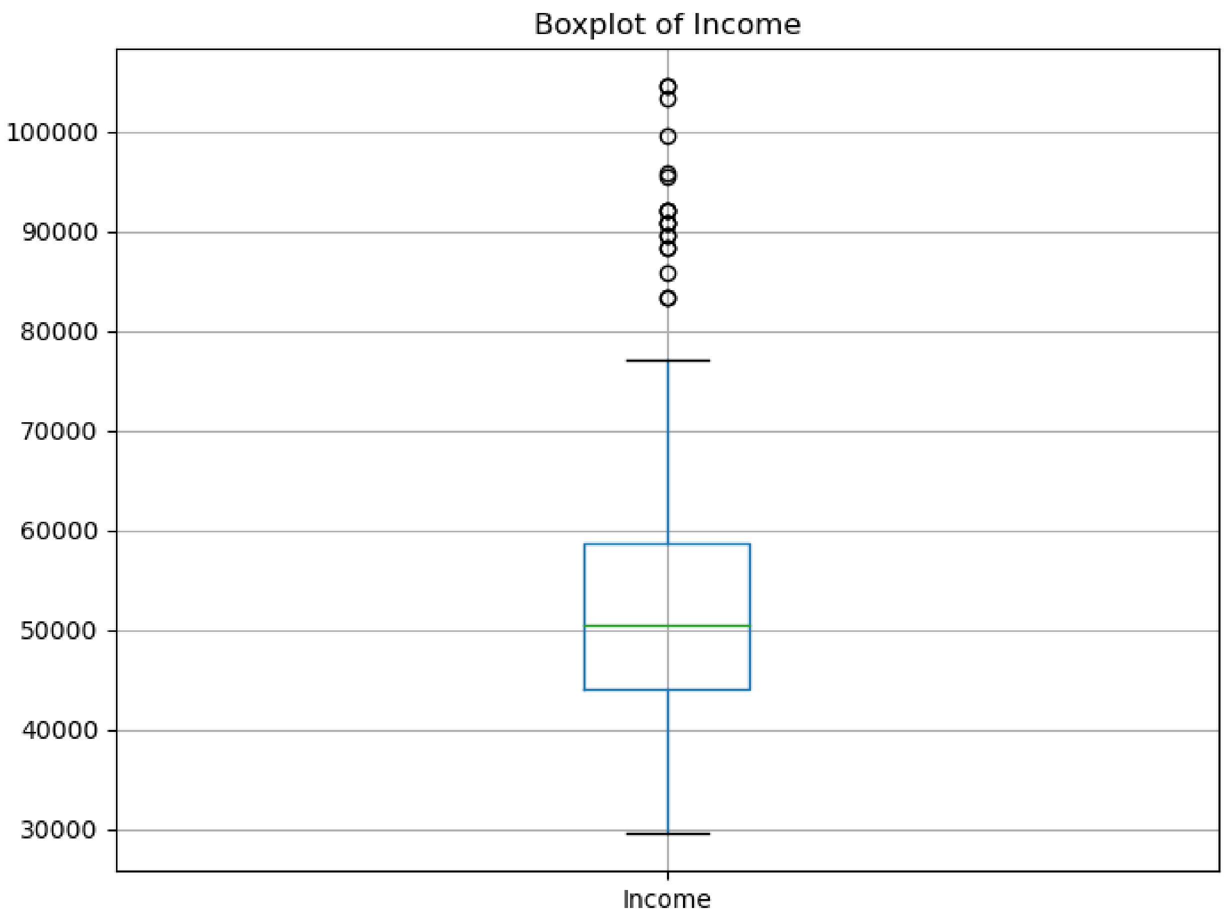
```
import matplotlib.pyplot as plt

# Plot boxplots for all numerical columns
numerical_columns = data.select_dtypes(include=['int64',
'float64']).columns
for col in numerical_columns:
    plt.figure(figsize=(8, 6))
    data.boxplot(column=[col])
    plt.title('Boxplot of ' + col)
    plt.show()
```

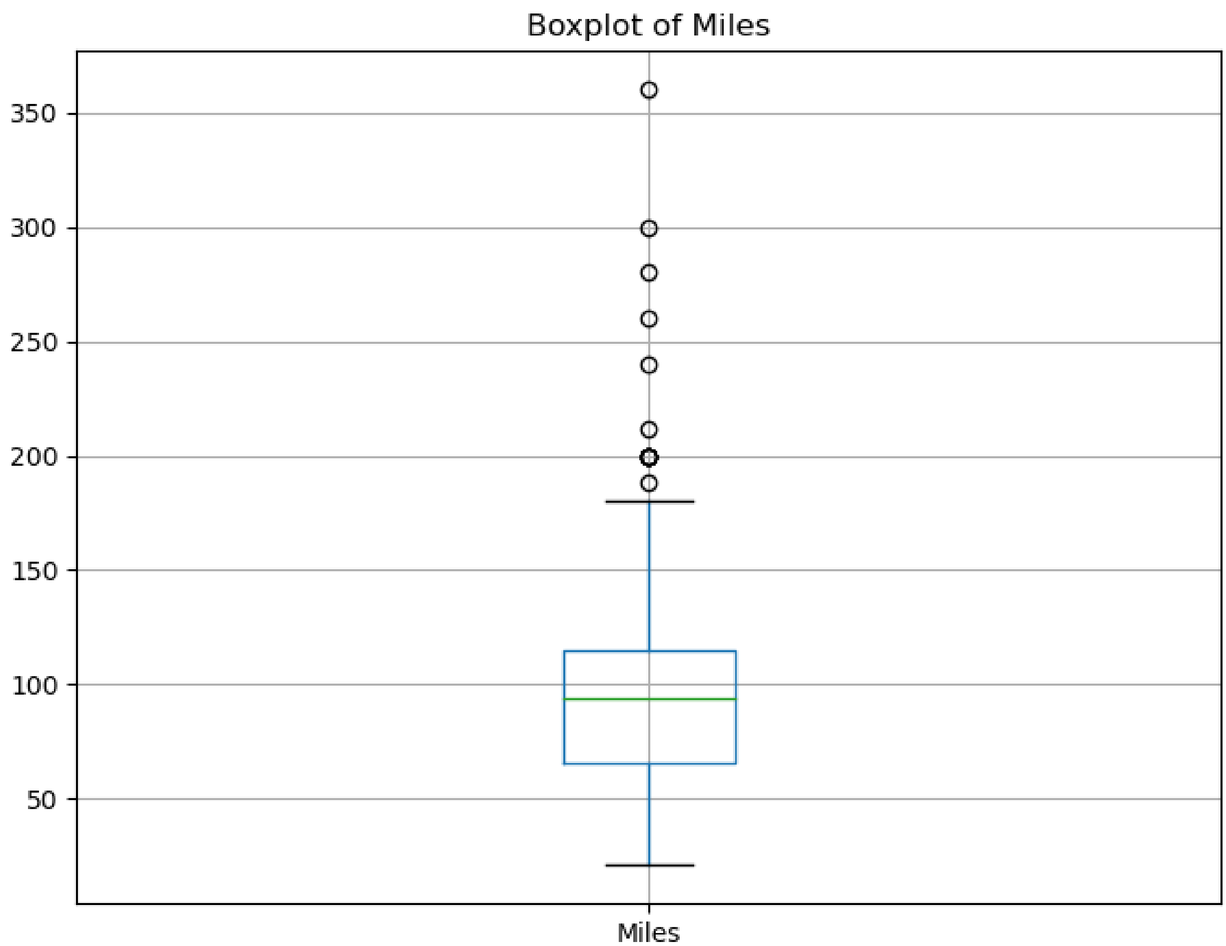

Boxplot of Age
50
45
40
35
30
25
20
Age


Boxplot of Education
20
18
16
14
12
Education


Boxplot of Usage
7
6
5
4
3
2
Usage


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


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




```
for col in numerical_columns:
    Q1 = data[col].quantile(0.25)
    Q3 = data[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = data[(data[col] < lower_bound) | (data[col] >
upper_bound)]
    print("Outliers in", col, ":", outliers)
    print("-----------------")

Outliers in Age :      Product  Age  Gender  Education MaritalStatus
Usage  Fitness  Income  \
78    KP281   47    Male          16      Partnered       4         3
56850
79    KP281   50  Female          16      Partnered       3         3
64809
139   KP481   48    Male          16      Partnered       2         3
57987
178   KP781   47    Male          18      Partnered       4         5
```

```
104581
179   KP781   48    Male         18     Partnered      4        5
95508

     Miles
78      94
79      66
139     64
178    120
179    180
-----------------
Outliers in Education :     Product  Age  Gender  Education
MaritalStatus  Usage  Fitness  Income  \
156   KP781   25    Male         20     Partnered      4        5
74701
157   KP781   26  Female         21        Single      4        3
69721
161   KP781   27    Male         21     Partnered      4        4
90886
175   KP781   40    Male         21        Single      6        5
83416

     Miles
156    170
157    100
161    100
175    200
-----------------
Outliers in Usage :     Product  Age  Gender  Education MaritalStatus
Usage  Fitness  Income  \
154   KP781   25    Male         18     Partnered      6        4
70966
155   KP781   25    Male         18     Partnered      6        5
75946
162   KP781   28  Female         18     Partnered      6        5
92131
163   KP781   28    Male         18     Partnered      7        5
77191
164   KP781   28    Male         18        Single      6        5
88396
166   KP781   29    Male         14     Partnered      7        5
85906
167   KP781   30  Female         16     Partnered      6        5
90886
170   KP781   31    Male         16     Partnered      6        5
89641
175   KP781   40    Male         21        Single      6        5
83416

     Miles
```

```
154    180
155    240
162    180
163    180
164    150
166    300
167    280
170    260
175    200
-----------------
Outliers in Fitness :     Product  Age  Gender  Education
MaritalStatus  Usage  Fitness  Income  \
14    KP281   23    Male         16     Partnered      3        1
38658
117   KP481   31  Female         18        Single      2        1
65220

     Miles
14      47
117     21
-----------------
Outliers in Income :     Product  Age  Gender  Education MaritalStatus
Usage  Fitness  Income  \
159   KP781   27    Male         16     Partnered      4        5
83416
160   KP781   27    Male         18        Single      4        3
88396
161   KP781   27    Male         21     Partnered      4        4
90886
162   KP781   28  Female         18     Partnered      6        5
92131
164   KP781   28    Male         18        Single      6        5
88396
166   KP781   29    Male         14     Partnered      7        5
85906
167   KP781   30  Female         16     Partnered      6        5
90886
168   KP781   30    Male         18     Partnered      5        4
103336
169   KP781   30    Male         18     Partnered      5        5
99601
170   KP781   31    Male         16     Partnered      6        5
89641
171   KP781   33  Female         18     Partnered      4        5
95866
172   KP781   34    Male         16        Single      5        5
92131
173   KP781   35    Male         16     Partnered      4        5
92131
```

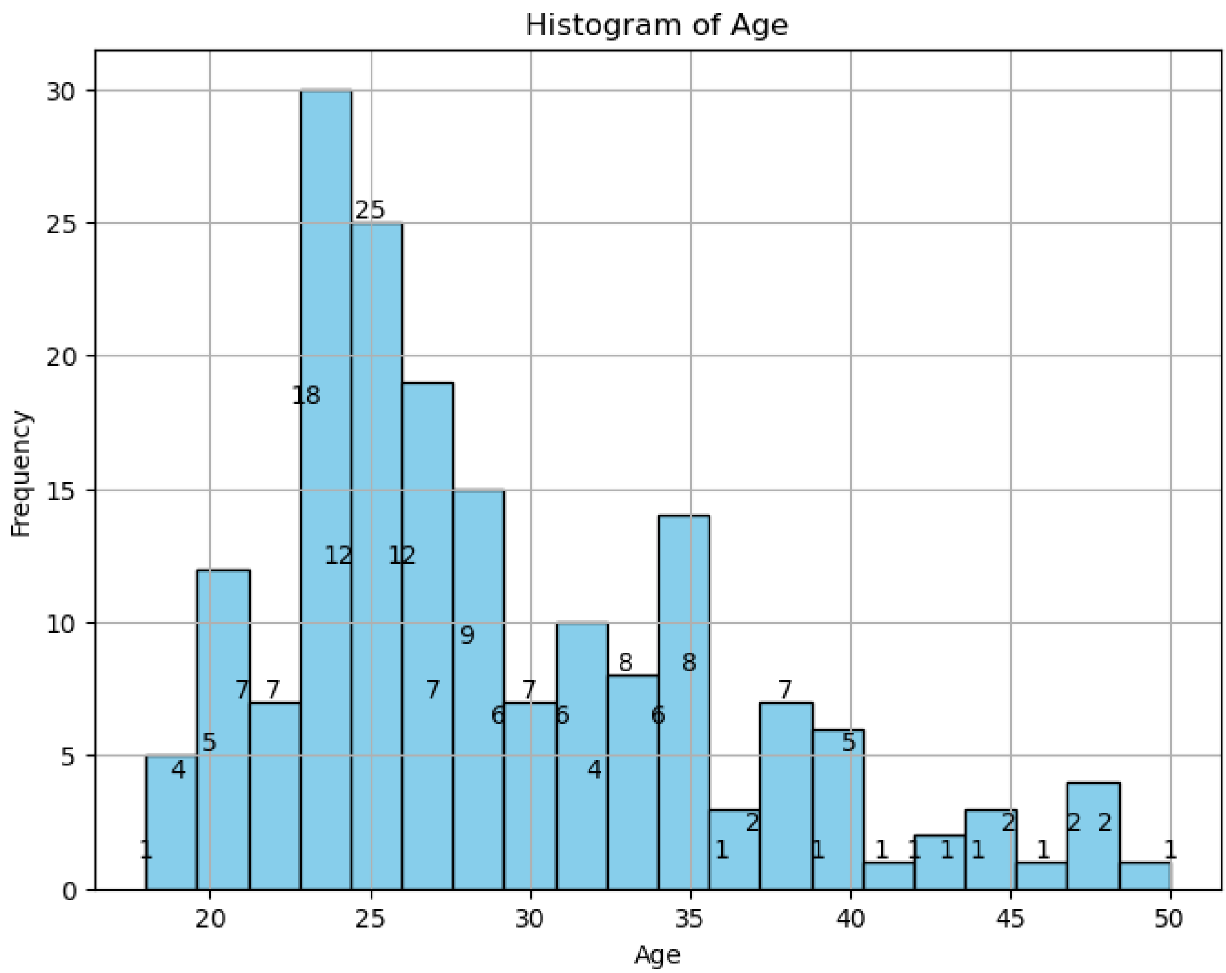
```
174   KP781   38    Male         18     Partnered      5        5
104581
175   KP781   40    Male         21        Single      6        5
83416
176   KP781   42    Male         18        Single      5        4
89641
177   KP781   45    Male         16        Single      5        5
90886
178   KP781   47    Male         18     Partnered      4        5
104581
179   KP781   48    Male         18     Partnered      4        5
95508

     Miles
159    160
160    100
161    100
162    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
-----------------
Outliers in Miles :     Product  Age  Gender  Education MaritalStatus
Usage  Fitness  Income  \
23    KP281   24  Female         16     Partnered      5        5
44343
84    KP481   21  Female         14     Partnered      5        4
34110
142   KP781   22    Male         18        Single      4        5
48556
148   KP781   24  Female         16        Single      5        5
52291
152   KP781   25  Female         18     Partnered      5        5
61006
155   KP781   25    Male         18     Partnered      6        5
75946
166   KP781   29    Male         14     Partnered      7        5
```

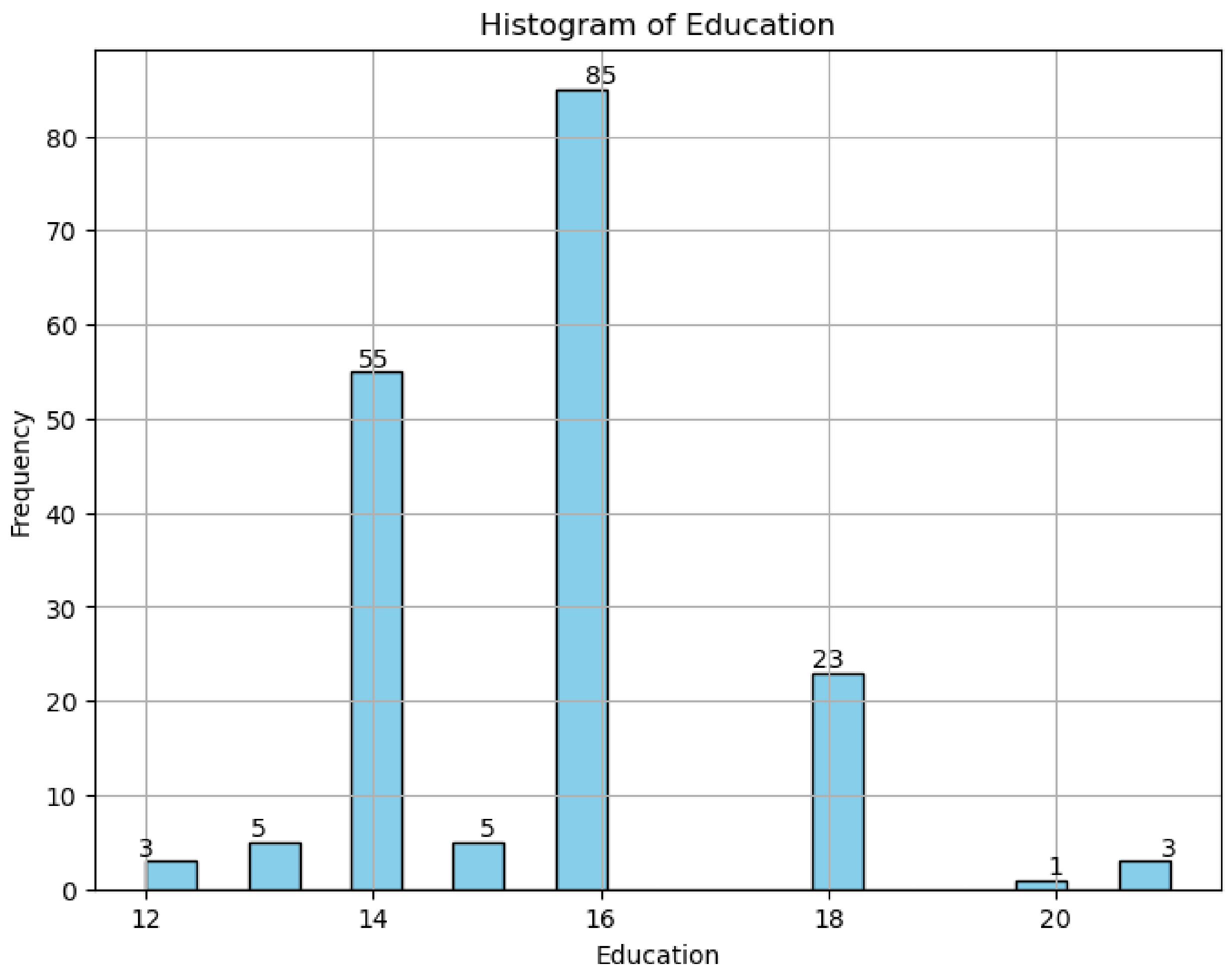
```
85906
167   KP781   30  Female          16      Partnered       6         5
90886
170   KP781   31    Male          16      Partnered       6         5
89641
171   KP781   33  Female          18      Partnered       4         5
95866
173   KP781   35    Male          16      Partnered       4         5
92131
175   KP781   40    Male          21         Single       6         5
83416
176   KP781   42    Male          18         Single       5         4
89641

     Miles
23     188
84     212
142    200
148    200
152    200
155    240
166    300
167    280
170    260
171    200
173    360
175    200
176    200
-----------------
```

Non Graphical Analysis

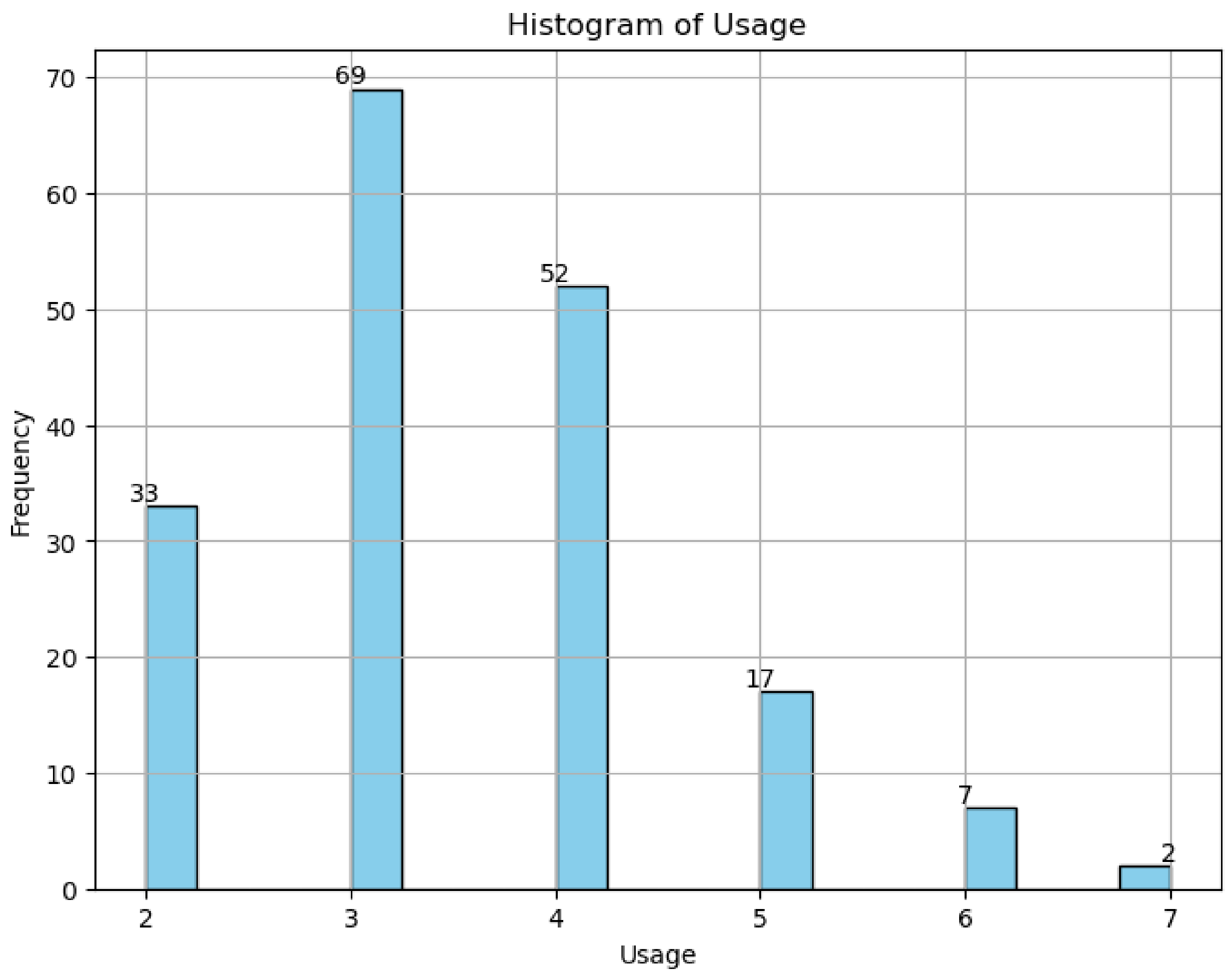
```
# Value counts and unique attributes for each column
non_numerical_columns =
data.select_dtypes(include=['category']).columns
for col in non_numerical_columns:
    print("Column:", col)
    print("Number of unique values:", data[col].nunique())
    print("Unique values:")
    print(data[col].unique())
    print("Value counts:")
    print(data[col].value_counts())
    print("\n")

Column: Product
Number of unique values: 3
Unique values:
['KP281', 'KP481', 'KP781']
Categories (3, object): ['KP281', 'KP481', 'KP781']
Value counts:
```

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


Column: Gender
Number of unique values: 2
Unique values:
['Male', 'Female']
Categories (2, object): ['Female', 'Male']
Value counts:
Gender
Male      104
Female     76
Name: count, dtype: int64


Column: MaritalStatus
Number of unique values: 2
Unique values:
['Single', 'Partnered']
Categories (2, object): ['Partnered', 'Single']
Value counts:
MaritalStatus
Partnered    107
Single        73
Name: count, dtype: int64



# Histograms with bins for numerical columns
numerical_columns = data.select_dtypes(include=['int64']).columns
for col in numerical_columns:
    plt.figure(figsize=(8, 6))
    plt.hist(data[col], bins=20, color='skyblue', edgecolor='black')
    # Add labels to the bars
    counts = data[col].value_counts()
    for i, count in enumerate(counts):
        plt.text(counts.index[i], count, str(count), ha='center',
va='bottom')
    plt.title(f'Histogram of {col}')
    plt.xlabel(col)
    plt.ylabel('Frequency')
    plt.grid(True)
    plt.show()
```

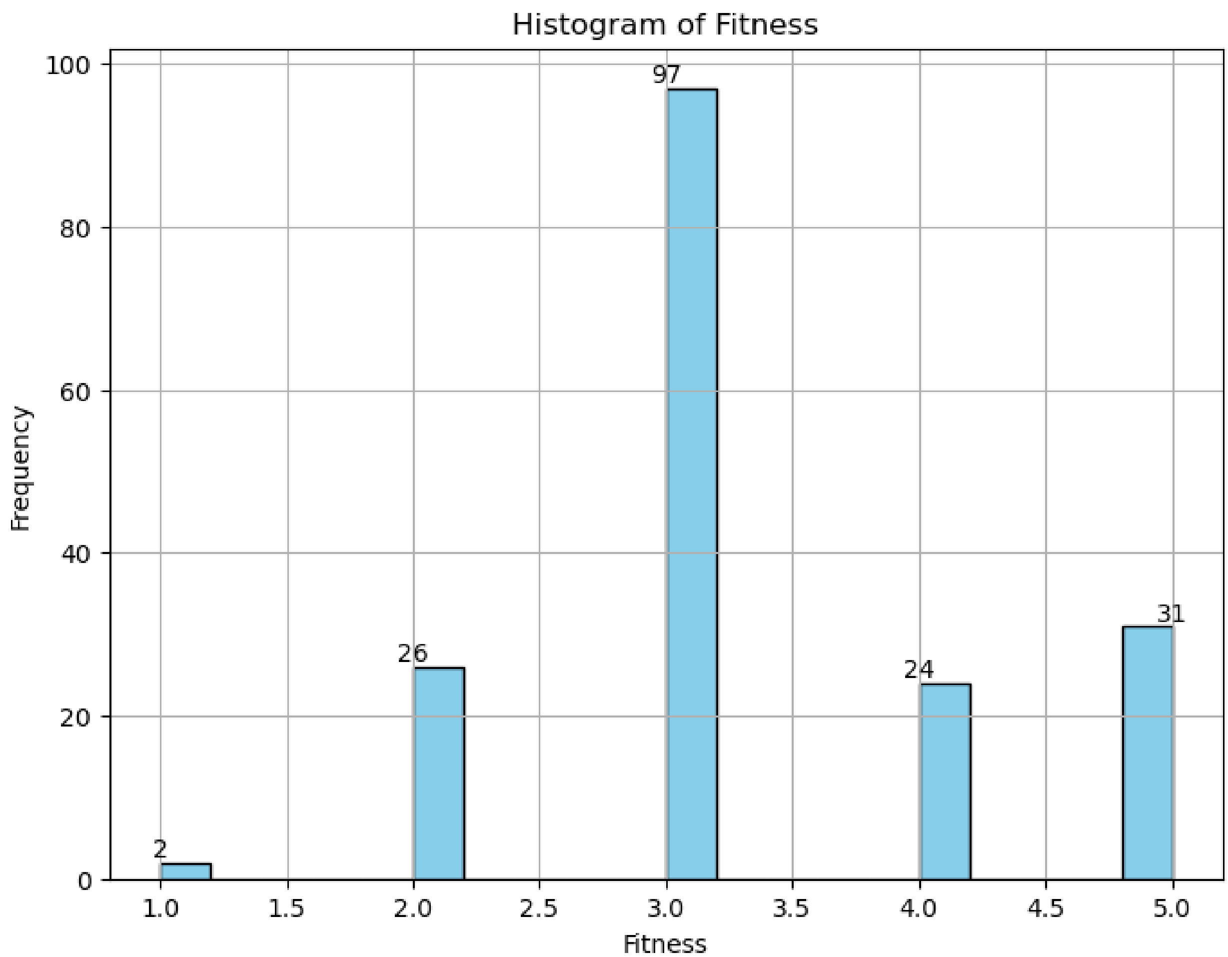

Histogram of Age
Frequency
Age
0
5
10
15
20
25
30
20
25
30
35
40
45
50
1
4
5
7
7
18
12
25
12
7
9
6
7
6
4
8
6
8
1
2
7
1
5
1
1
1
1
2
1
2
2
1

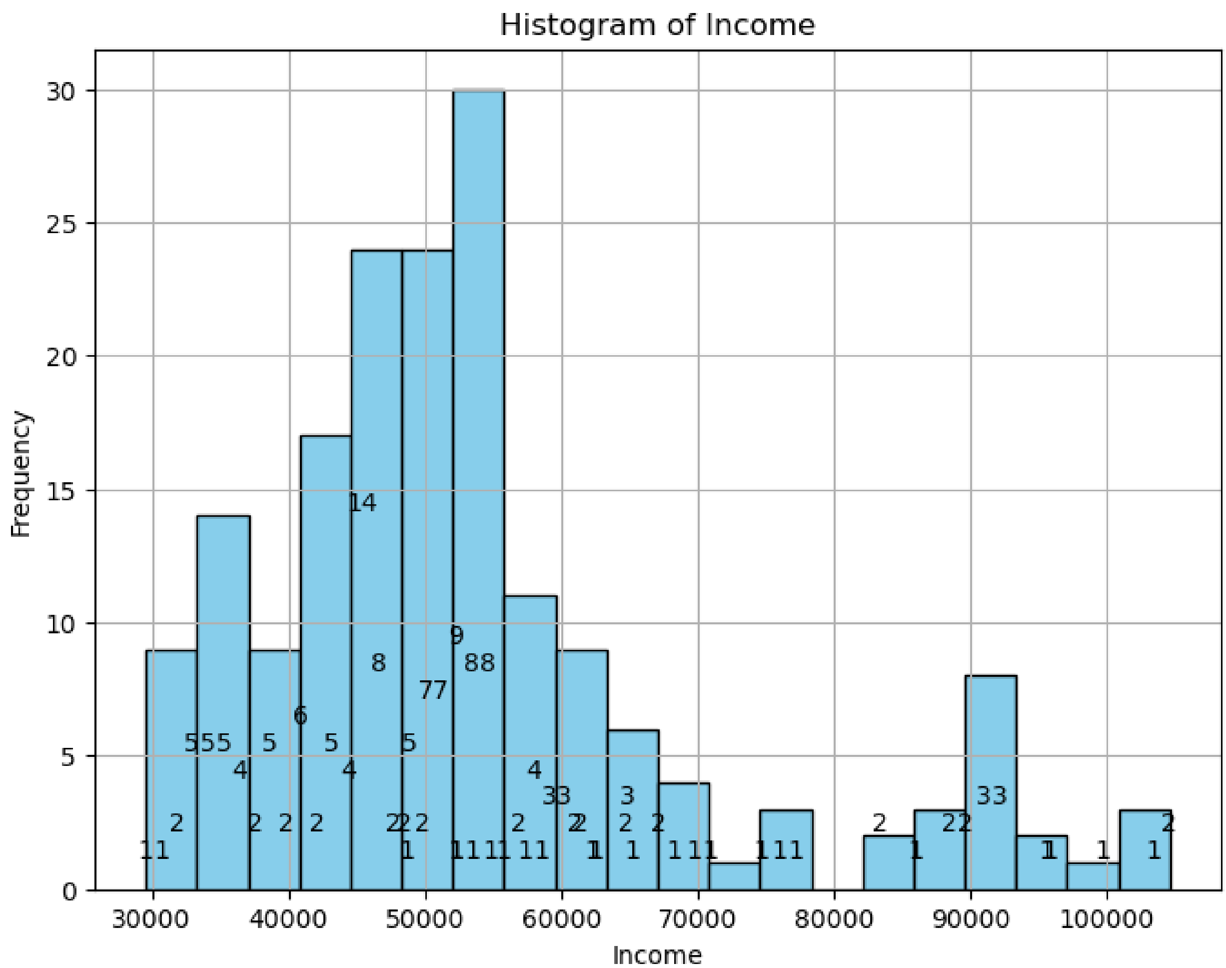


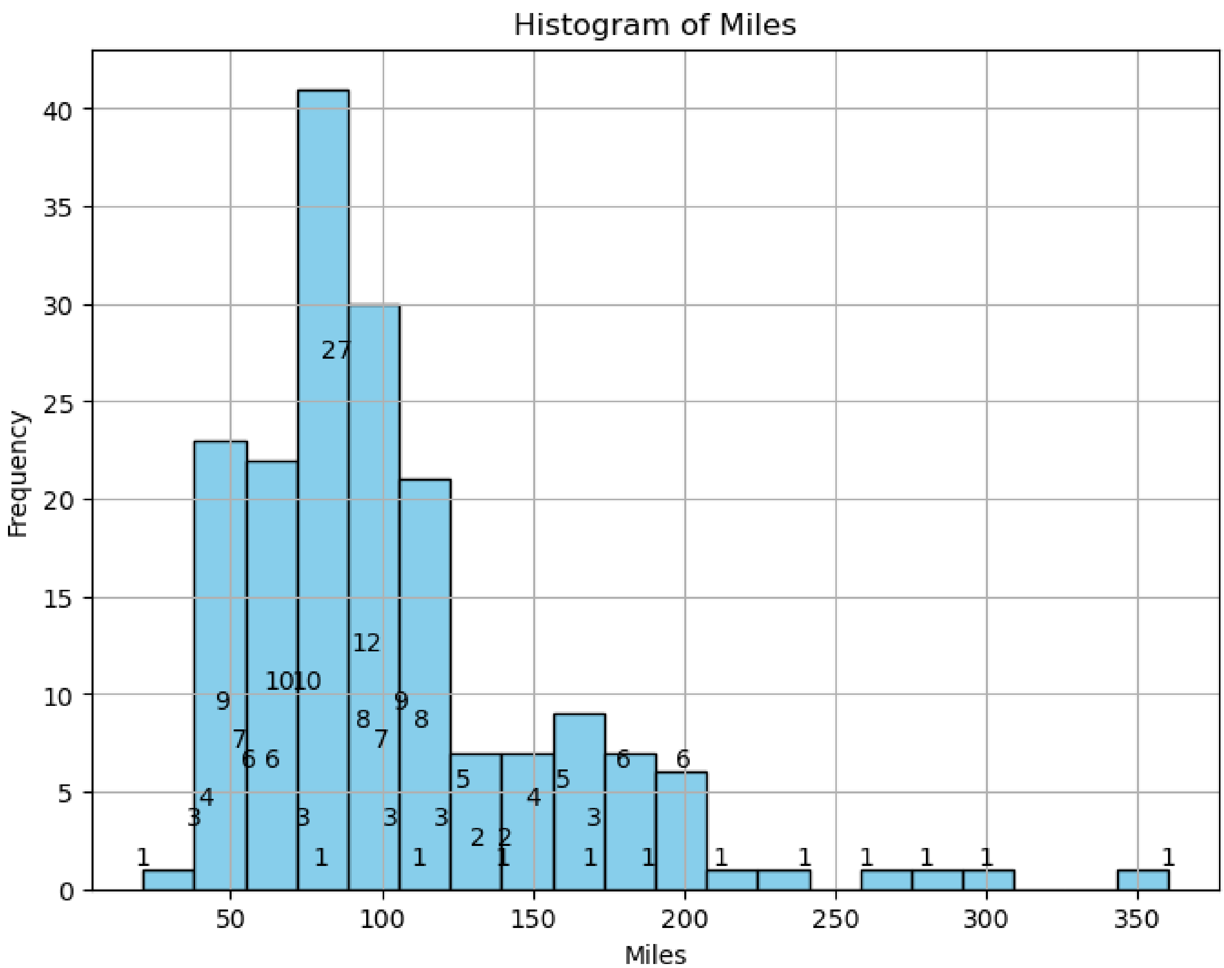
Histogram of Education
Frequency
Education
0
10
20
30
40
50
60
70
80
12
14
16
18
20
3
5
55
5
85
23
1
3



Histogram of Usage
Frequency
Usage
0
10
20
30
40
50
60
70
2
3
4
5
6
7
33
69
52
17
7
2


Histogram of Fitness
Frequency
Fitness
0
20
40
60
80
100
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
2
26
97
24
31


Histogram of Income
Frequency
Income
0
5
10
15
20
25
30
30000
40000
50000
60000
70000
80000
90000
100000




```
for col in numerical_columns:
    print("Column:", col)
    print("Value counts:")
    print(data[col].value_counts())
    print("\n")

Column: Age
Value counts:
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


Column: Education
Value counts:
Education
16    85
14    55
18    23
15     5
13     5
12     3
21     3
20     1
Name: count, dtype: int64


Column: Usage
Value counts:
Usage
3    69
4    52
2    33
5    17
6     7
7     2
Name: count, dtype: int64
```

```
Column: Fitness
Value counts:
Fitness
3    97
5    31
2    26
4    24
1     2
Name: count, dtype: int64


Column: Income
Value counts:
Income
45480    14
52302     9
46617     8
54576     8
53439     8
         ..
65220     1
55713     1
68220     1
30699     1
95508     1
Name: count, Length: 62, dtype: int64


Column: Miles
Value counts:
Miles
85     27
95     12
66     10
75     10
47      9
106     9
94      8
113     8
53      7
100     7
180     6
200     6
56      6
64      6
127     5
160     5
42      4
150     4
38      3
```

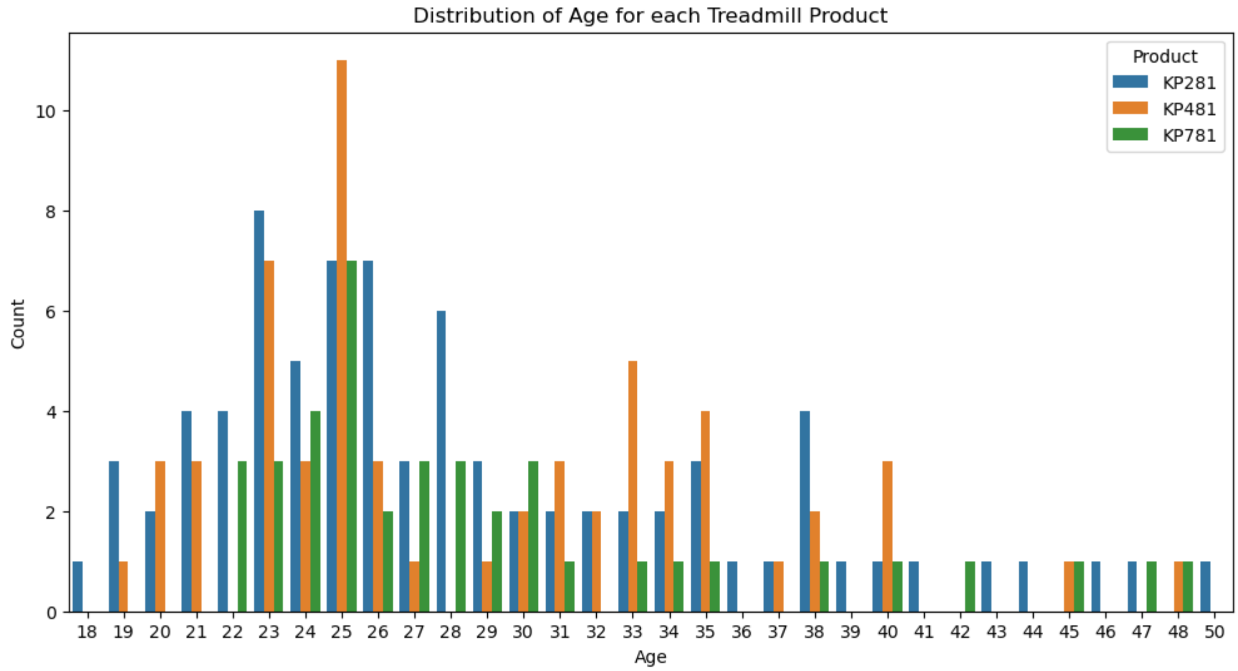
```
74     3
170    3
120    3
103    3
132    2
141    2
280    1
260    1
300    1
240    1
112    1
212    1
80     1
140    1
21     1
169    1
188    1
360    1
Name: count, dtype: int64
```

Graphical Aanalysis

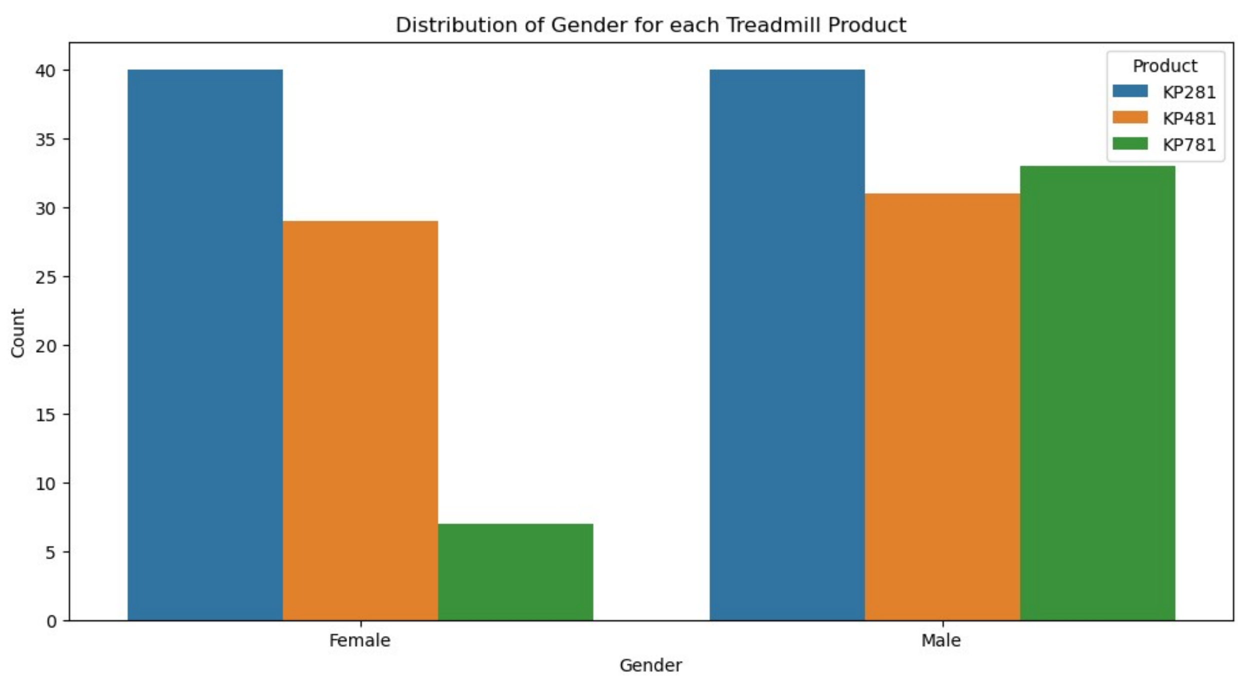
```
import seaborn as sns
```

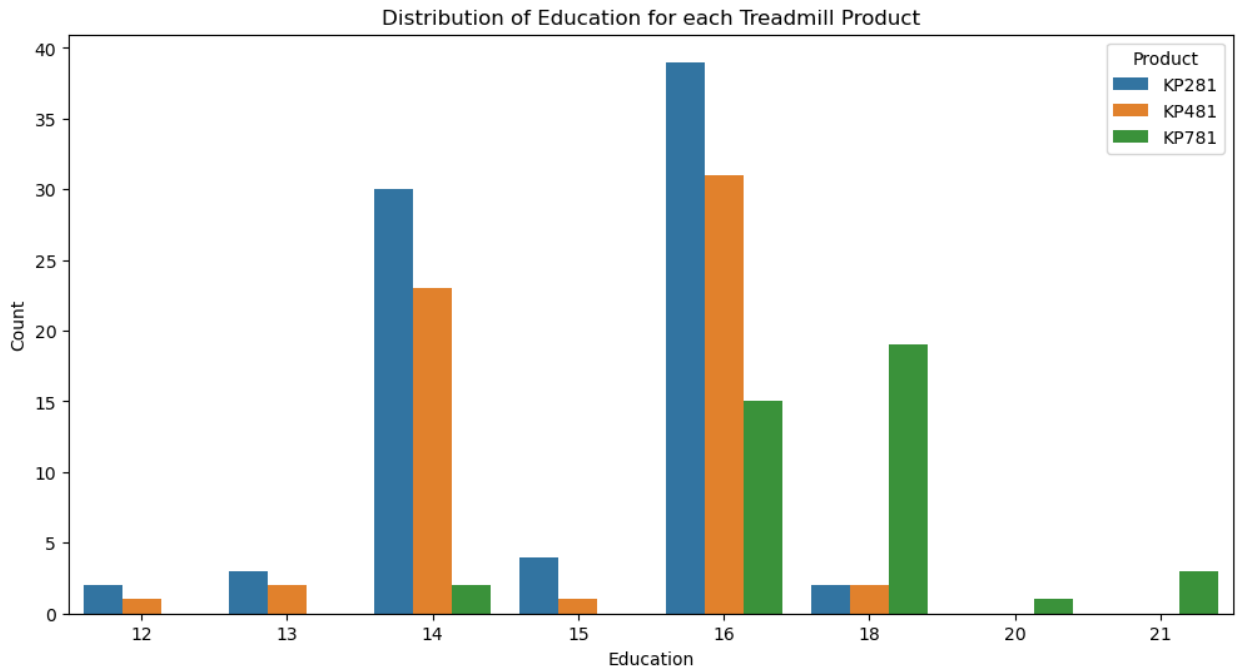
1. Demographic Characteristics Analysis:

```
# Define a function to plot demographic variables for each treadmill product
def plot_demographics(variable):
    plt.figure(figsize=(12, 6))
    sns.countplot(data=data, x=variable, hue='Product')
    plt.title(f'Distribution of {variable} for each Treadmill Product')
    plt.xlabel(variable)
    plt.ylabel('Count')
    plt.legend(title='Product')
    plt.show()

# Plot demographic variables for each treadmill product
demographic_variables = ['Age', 'Gender', 'Education', 'MaritalStatus']
for variable in demographic_variables:
    plot_demographics(variable)
```

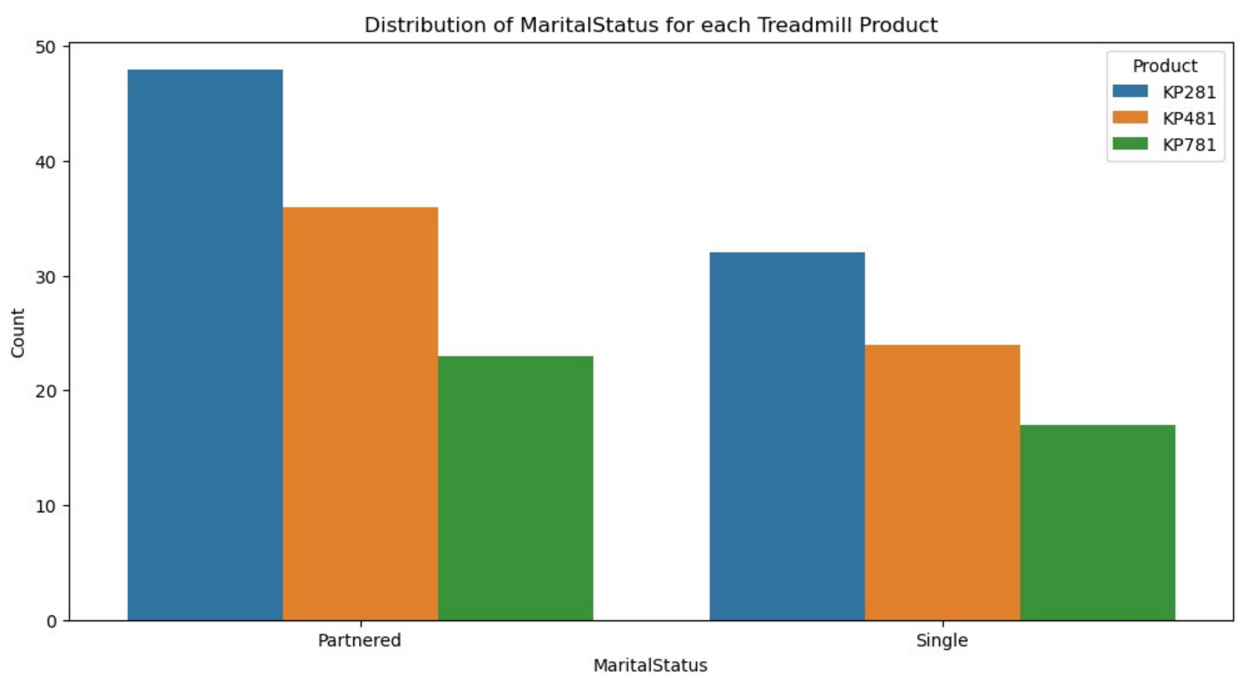



```
E:\rasa\Lib\site-packages\seaborn\categorical.py:641: FutureWarning:
The default of observed=False is deprecated and will be changed to
True in a future version of pandas. Pass observed=False to retain
current behavior or observed=True to adopt the future default and
silence this warning.
  grouped_vals = vals.groupby(grouper)
```





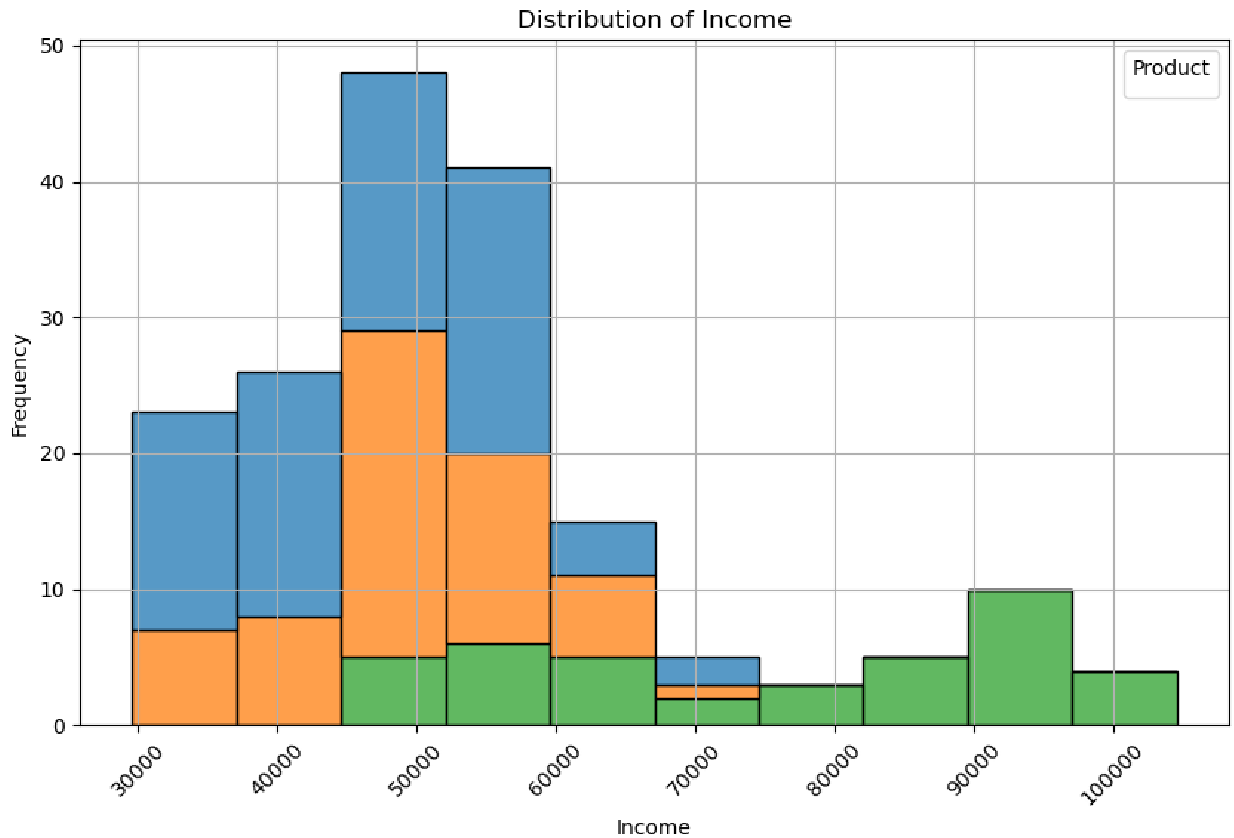

```
E:\rasa\Lib\site-packages\seaborn\categorical.py:641: FutureWarning:
The default of observed=False is deprecated and will be changed to
True in a future version of pandas. Pass observed=False to retain
current behavior or observed=True to adopt the future default and
silence this warning.
  grouped_vals = vals.groupby(grouper)
```



```python
# Define the number of bins and range for income
num_bins = 10
income_range = (data['Income'].min(), data['Income'].max())

# Plot the histogram of income with bins
plt.figure(figsize=(10, 6))
sns.histplot(data=data, x='Income', bins=num_bins,hue='Product',
multiple='stack')
plt.title('Distribution of Income')
plt.xlabel('Income')
plt.ylabel('Frequency')
plt.xticks(rotation=45)  # Rotate x-axis labels for better visibility
plt.legend(title='Product')
plt.grid(True)
plt.show()
```

```
E:\rasa\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
E:\rasa\Lib\site-packages\seaborn\_oldcore.py:1057: FutureWarning: The
default of observed=False is deprecated and will be changed to True in
a future version of pandas. Pass observed=False to retain current
behavior or observed=True to adopt the future default and silence this
warning.
  grouped_data = data.groupby(
E:\rasa\Lib\site-packages\seaborn\_oldcore.py:1075: FutureWarning:
When grouping with a length-1 list-like, you will need to pass a
length-1 tuple to get_group in a future version of pandas. Pass
`(name,)` instead of `name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
E:\rasa\Lib\site-packages\seaborn\_oldcore.py:1057: FutureWarning: The
default of observed=False is deprecated and will be changed to True in
a future version of pandas. Pass observed=False to retain current
behavior or observed=True to adopt the future default and silence this
warning.
  grouped_data = data.groupby(
E:\rasa\Lib\site-packages\seaborn\_oldcore.py:1075: FutureWarning:
When grouping with a length-1 list-like, you will need to pass a
length-1 tuple to get_group in a future version of pandas. Pass
`(name,)` instead of `name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
E:\rasa\Lib\site-packages\seaborn\_oldcore.py:1075: FutureWarning:
When grouping with a length-1 list-like, you will need to pass a
length-1 tuple to get_group in a future version of pandas. Pass
`(name,)` instead of `name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
E:\rasa\Lib\site-packages\seaborn\_oldcore.py:1075: FutureWarning:
When grouping with a length-1 list-like, you will need to pass a
length-1 tuple to get_group in a future version of pandas. Pass
```

```
`(name,)` instead of `name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
No artists with labels found to put in legend.  Note that artists
whose label start with an underscore are ignored when legend() is
called with no argument.
```



2.Usage Patterns Variation:

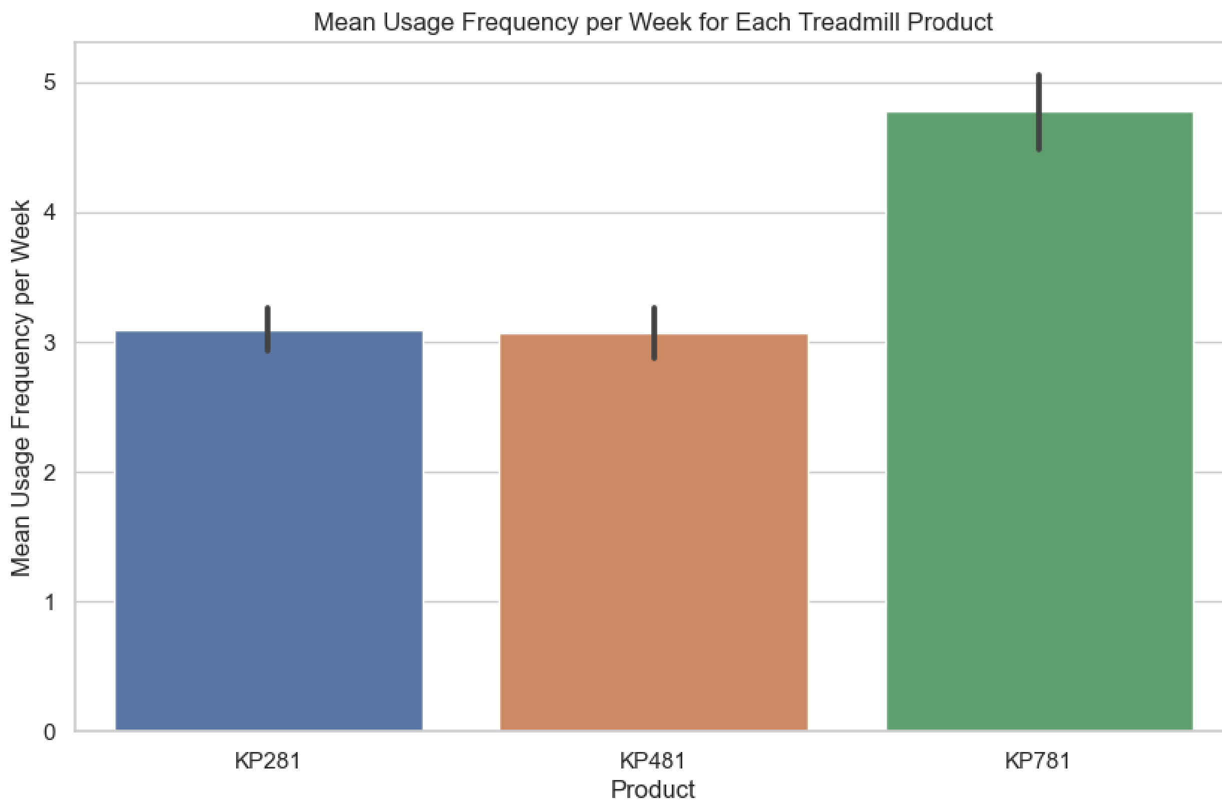
```
# Calculate mean usage frequency per week for each treadmill product
mean_usage_per_product = data.groupby('Product')['Usage'].mean()

# Print descriptive statistics
print("Descriptive Statistics for Usage Frequency:")
print(mean_usage_per_product.describe())

# Compute probability distribution of usage frequencies for each
product
usage_distribution = data.groupby('Product')
['Usage'].value_counts(normalize=True).unstack()

# Print probability distribution
print("\nProbability Distribution of Usage Frequency for Each
```

```
Product:")
print(usage_distribution)
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_14556\2159259940.py:2:
FutureWarning: The default of observed=False is deprecated and will be
changed to True in a future version of pandas. Pass observed=False to
retain current behavior or observed=True to adopt the future default
and silence this warning.
  mean_usage_per_product = data.groupby('Product')['Usage'].mean()
C:\Users\user\AppData\Local\Temp\ipykernel_14556\2159259940.py:9:
FutureWarning: The default of observed=False is deprecated and will be
changed to True in a future version of pandas. Pass observed=False to
retain current behavior or observed=True to adopt the future default
and silence this warning.
  usage_distribution = data.groupby('Product')
['Usage'].value_counts(normalize=True).unstack()

Descriptive Statistics for Usage Frequency:
count    3.000000
mean     3.643056
std      0.980348
min      3.066667
25%      3.077083
50%      3.087500
75%      3.931250
max      4.775000
Name: Usage, dtype: float64

Probability Distribution of Usage Frequency for Each Product:
Usage           2         3      4      5      6     7
Product
KP281    0.237500  0.462500  0.275  0.025  0.000  0.00
KP481    0.233333  0.516667  0.200  0.050  0.000  0.00
KP781    0.000000  0.025000  0.450  0.300  0.175  0.05
```

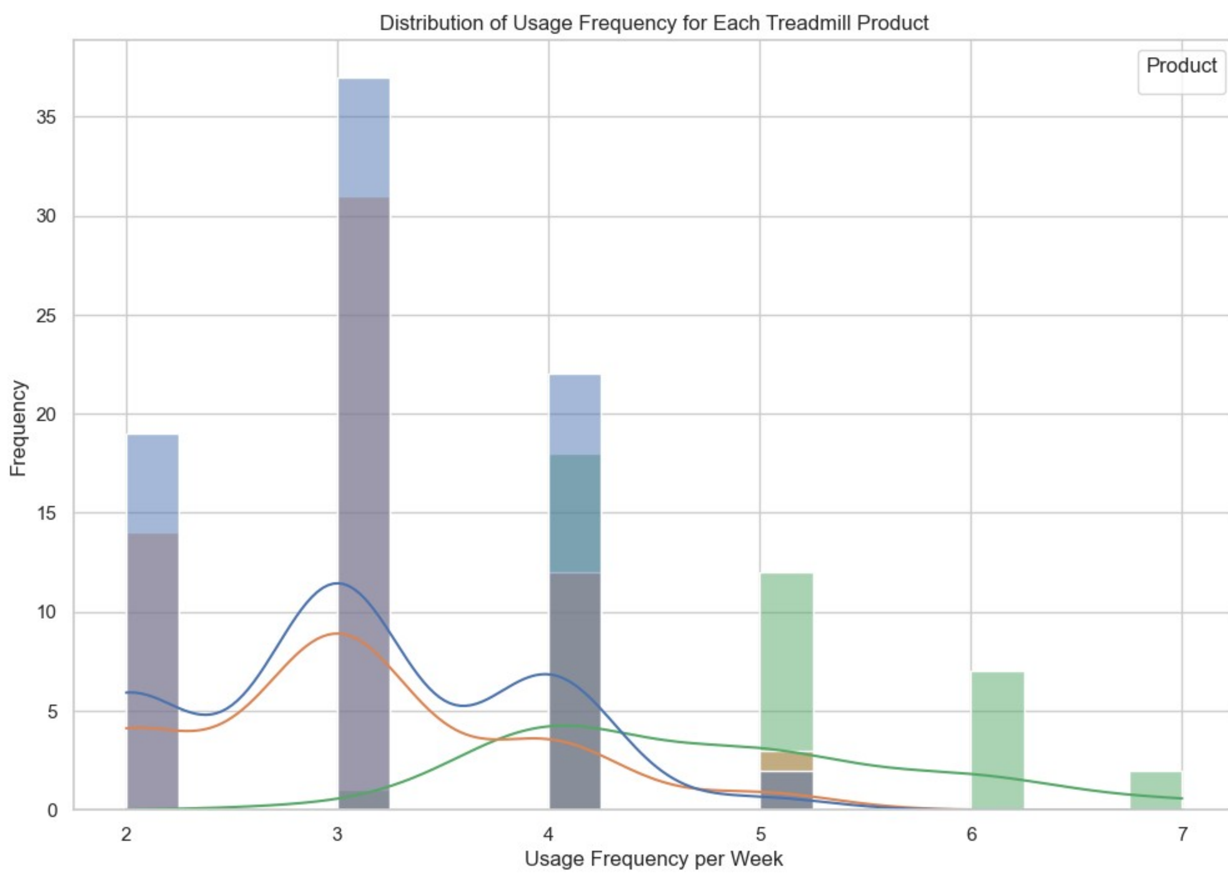
```
# Set the style of seaborn
sns.set(style="whitegrid")

# Create a bar plot of mean usage frequency for each treadmill product
plt.figure(figsize=(10, 6))
sns.barplot(x='Product', y='Usage', data=data, estimator=np.mean)
plt.title('Mean Usage Frequency per Week for Each Treadmill Product')
plt.xlabel('Product')
plt.ylabel('Mean Usage Frequency per Week')
plt.show()

# Create histograms or density plots of usage frequencies for each
treadmill product
plt.figure(figsize=(12, 8))
sns.histplot(data=data, x='Usage', hue='Product', kde=True, bins=20)
```

```
plt.title('Distribution of Usage Frequency for Each Treadmill
Product')
plt.xlabel('Usage Frequency per Week')
plt.ylabel('Frequency')
plt.legend(title='Product')
plt.show()

E:\rasa\Lib\site-packages\seaborn\categorical.py:641: FutureWarning:
The default of observed=False is deprecated and will be changed to
True in a future version of pandas. Pass observed=False to retain
current behavior or observed=True to adopt the future default and
silence this warning.
  grouped_vals = vals.groupby(grouper)
```




```
E:\rasa\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
E:\rasa\Lib\site-packages\seaborn\_oldcore.py:1057: FutureWarning: The
default of observed=False is deprecated and will be changed to True in
a future version of pandas. Pass observed=False to retain current
behavior or observed=True to adopt the future default and silence this
warning.
  grouped_data = data.groupby(
```

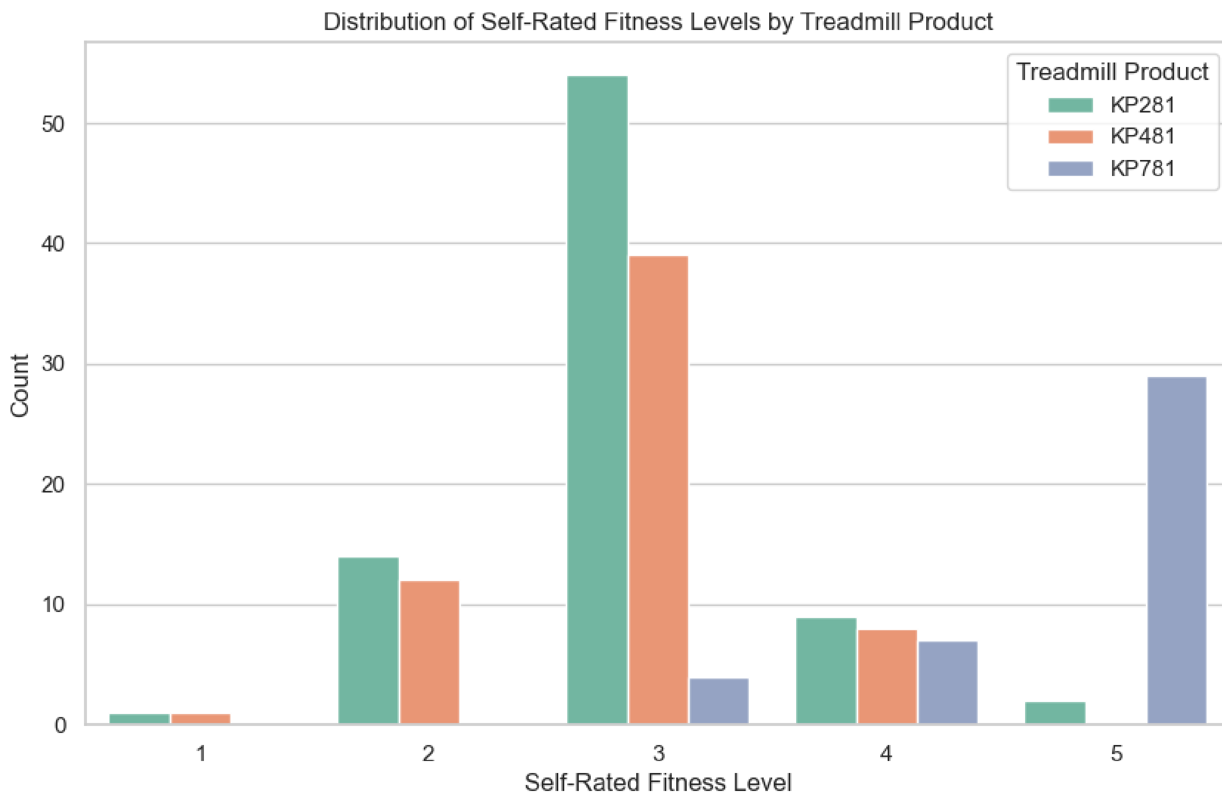
```
E:\rasa\Lib\site-packages\seaborn\_oldcore.py:1075: FutureWarning:
When grouping with a length-1 list-like, you will need to pass a
length-1 tuple to get_group in a future version of pandas. Pass
`(name,)` instead of `name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
E:\rasa\Lib\site-packages\seaborn\_oldcore.py:1075: FutureWarning:
When grouping with a length-1 list-like, you will need to pass a
length-1 tuple to get_group in a future version of pandas. Pass
`(name,)` instead of `name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
E:\rasa\Lib\site-packages\seaborn\_oldcore.py:1075: FutureWarning:
When grouping with a length-1 list-like, you will need to pass a
length-1 tuple to get_group in a future version of pandas. Pass
`(name,)` instead of `name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
No artists with labels found to put in legend.  Note that artists
whose label start with an underscore are ignored when legend() is
called with no argument.
```



3.Self-Rated Fitness Level Analysis:

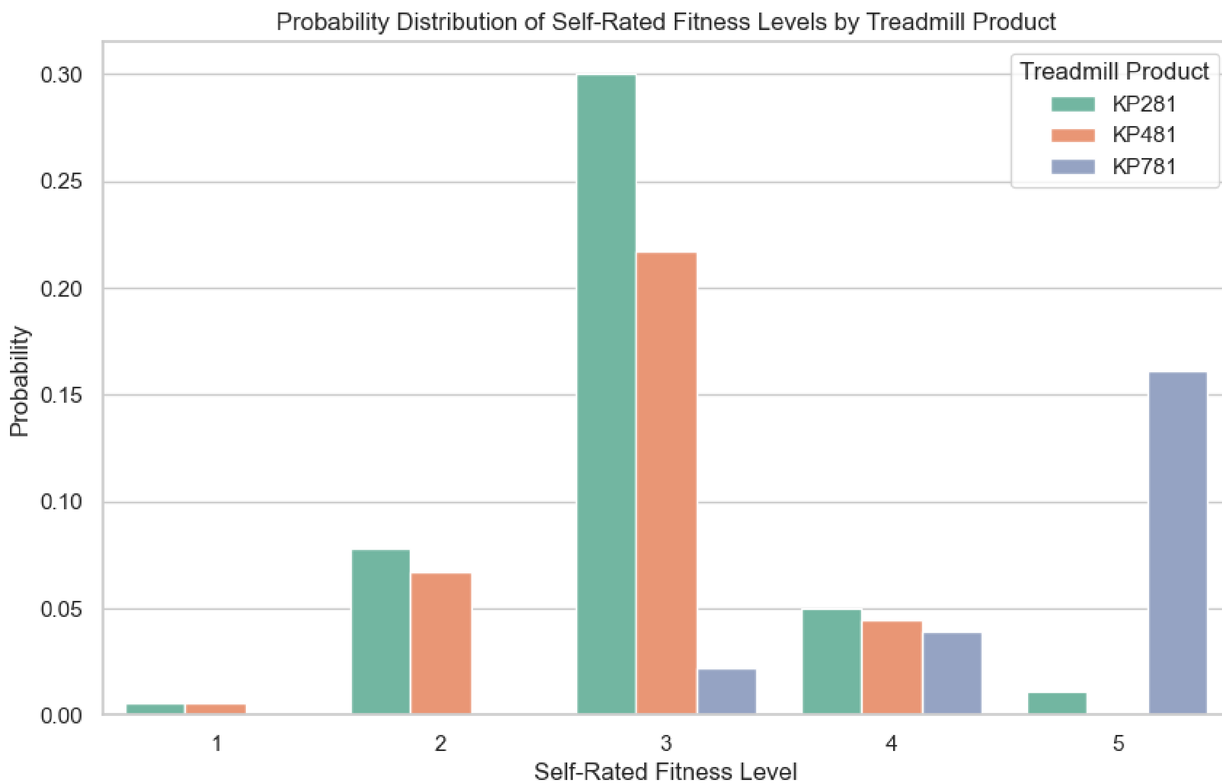
```
# Plotting the distribution of fitness levels for each treadmill
product using countplot
```

```
plt.figure(figsize=(10, 6))
sns.countplot(data=data, x='Fitness', hue='Product', palette='Set2')
plt.title('Distribution of Self-Rated Fitness Levels by Treadmill
Product')
plt.xlabel('Self-Rated Fitness Level')
plt.ylabel('Count')
plt.legend(title='Treadmill Product')
plt.show()
```




```
prob_df = data.groupby(['Product',
'Fitness']).size().div(len(data)).reset_index(name='Probability')

# Plotting the probability distribution of fitness levels for each
treadmill product
plt.figure(figsize=(10, 6))
sns.barplot(data=prob_df, x='Fitness', y='Probability', hue='Product',
palette='Set2')
plt.title('Probability Distribution of Self-Rated Fitness Levels by
Treadmill Product')
plt.xlabel('Self-Rated Fitness Level')
plt.ylabel('Probability')
plt.legend(title='Treadmill Product')
plt.show()
```

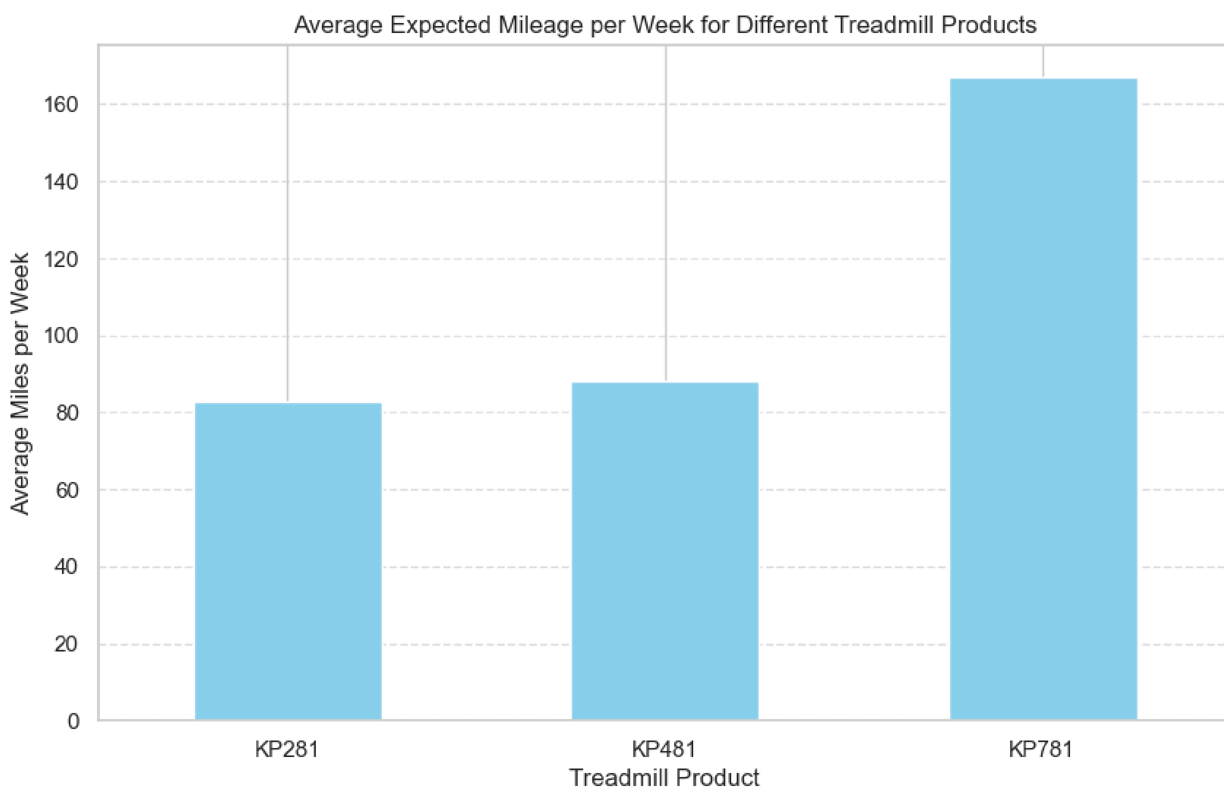
```
C:\Users\user\AppData\Local\Temp\ipykernel_14556\443691877.py:1: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future default and silence this warning.
  prob_df = data.groupby(['Product', 'Fitness']).size().div(len(data)).reset_index(name='Probability')
```



1. Expected Mileage Differences:

```
# Grouping the data by 'Product' and calculating the mean of 'Miles' for each product
average_mileage_per_product = data.groupby('Product')['Miles'].mean()

# Displaying the average expected mileage per week for each treadmill product
print("Average Expected Mileage per Week for Different Treadmill Products:")
print(average_mileage_per_product)

Average Expected Mileage per Week for Different Treadmill Products:
Product
KP281     82.787500
KP481     87.933333
KP781    166.900000
Name: Miles, dtype: float64
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_14556\2800426568.py:2: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future default and silence this warning.
  average_mileage_per_product = data.groupby('Product')['Miles'].mean()

# Plotting the average expected mileage per week for each treadmill product
plt.figure(figsize=(10, 6))
average_mileage_per_product.plot(kind='bar', color='skyblue')
plt.title('Average Expected Mileage per Week for Different Treadmill Products')
plt.xlabel('Treadmill Product')
plt.ylabel('Average Miles per Week')
plt.xticks(rotation=0)  # Rotate x-axis labels if necessary
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



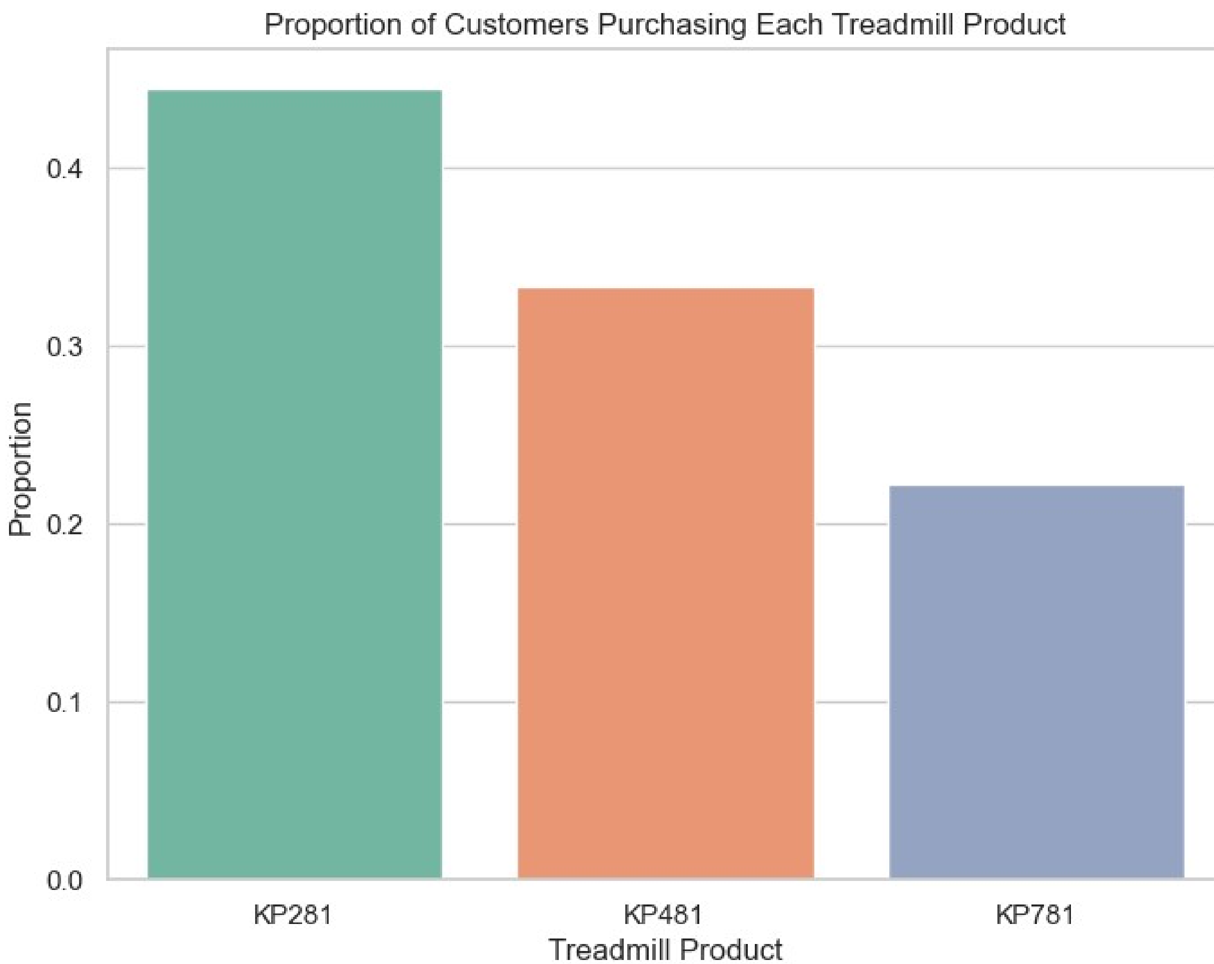
1. Customer Preferences Distribution:

```
# Calculate the proportion of customers purchasing each treadmill product
product_counts = data['Product'].value_counts(normalize=True)
product_counts
```

```
Product
KP281    0.444444
KP481    0.333333
KP781    0.222222
Name: proportion, dtype: float64
```

```python
# Visualize the distribution
plt.figure(figsize=(8, 6))
sns.barplot(x=product_counts.index, y=product_counts.values,
palette='Set2')
plt.title('Proportion of Customers Purchasing Each Treadmill Product')
plt.xlabel('Treadmill Product')
plt.ylabel('Proportion')
plt.show()
```

```
E:\rasa\Lib\site-packages\seaborn\categorical.py:641: FutureWarning:
The default of observed=False is deprecated and will be changed to
True in a future version of pandas. Pass observed=False to retain
current behavior or observed=True to adopt the future default and
silence this warning.
  grouped_vals = vals.groupby(grouper)
```



1. Influence of Demographic Characteristics:

```
# Create a crosstab to analyze the relationship between age group and
treadmill product preferences
age_treadmill_crosstab = pd.crosstab(index=data['Age'],
columns=data['Product'], normalize='index')

# Create a crosstab to analyze the relationship between gender and
treadmill product preferences
gender_treadmill_crosstab = pd.crosstab(index=data['Gender'],
columns=data['Product'], normalize='index')

# Create a crosstab to analyze the relationship between education
level and treadmill product preferences
education_treadmill_crosstab = pd.crosstab(index=data['Education'],
columns=data['Product'], normalize='index')

# Create a crosstab to analyze the relationship between marital status
and treadmill product preferences
marital_status_treadmill_crosstab =
pd.crosstab(index=data['MaritalStatus'], columns=data['Product'],
```

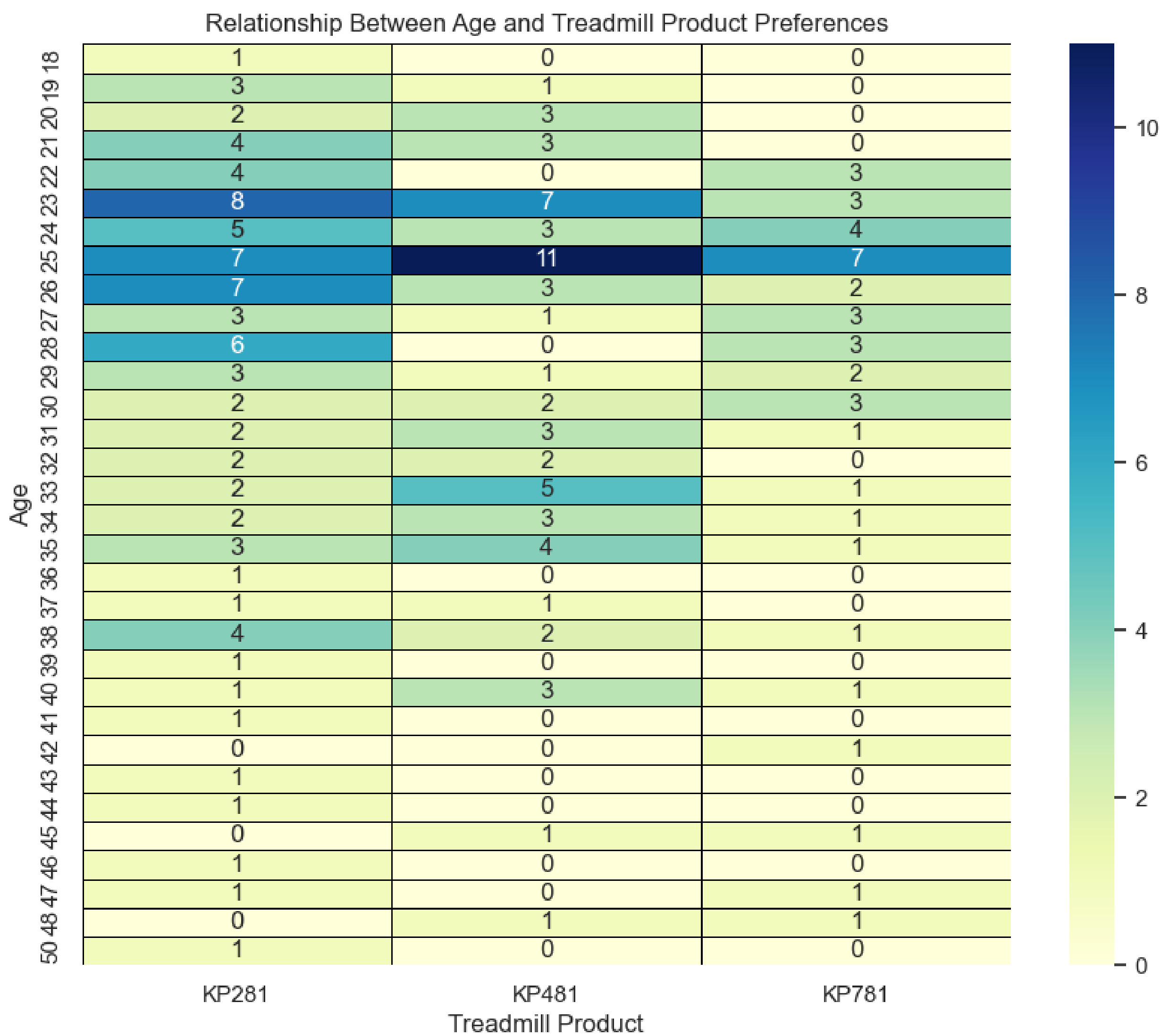
```
normalize='index')

# Create a crosstab to analyze the relationship between income level
and treadmill product preferences
income_treadmill_crosstab = pd.crosstab(index=data['Income'],
columns=data['Product'], normalize='index')

print("Relatiponship between age and product",age_treadmill_crosstab)
```

```
Relatiponship between age and product Product     KP281     KP481
KP781
Age
18       1.000000  0.000000  0.000000
19       0.750000  0.250000  0.000000
20       0.400000  0.600000  0.000000
21       0.571429  0.428571  0.000000
22       0.571429  0.000000  0.428571
23       0.444444  0.388889  0.166667
24       0.416667  0.250000  0.333333
25       0.280000  0.440000  0.280000
26       0.583333  0.250000  0.166667
27       0.428571  0.142857  0.428571
28       0.666667  0.000000  0.333333
29       0.500000  0.166667  0.333333
30       0.285714  0.285714  0.428571
31       0.333333  0.500000  0.166667
32       0.500000  0.500000  0.000000
33       0.250000  0.625000  0.125000
34       0.333333  0.500000  0.166667
35       0.375000  0.500000  0.125000
36       1.000000  0.000000  0.000000
37       0.500000  0.500000  0.000000
38       0.571429  0.285714  0.142857
39       1.000000  0.000000  0.000000
40       0.200000  0.600000  0.200000
41       1.000000  0.000000  0.000000
42       0.000000  0.000000  1.000000
43       1.000000  0.000000  0.000000
44       1.000000  0.000000  0.000000
45       0.000000  0.500000  0.500000
46       1.000000  0.000000  0.000000
47       0.500000  0.000000  0.500000
48       0.000000  0.500000  0.500000
50       1.000000  0.000000  0.000000
```

```
# Pivot the data to prepare for the heatmap
age_product_pivot = data.pivot_table(index='Age', columns='Product',
aggfunc='size', fill_value=0)

# Create the heatmap
```

```
plt.figure(figsize=(10, 8))
sns.heatmap(age_product_pivot, cmap='YlGnBu', annot=True, fmt='d',
linewidths=0.5, linecolor='black')
plt.title('Relationship Between Age and Treadmill Product
Preferences')
plt.xlabel('Treadmill Product')
plt.ylabel('Age')
plt.show()
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_14556\489202945.py:2:
FutureWarning: The default value of observed=False is deprecated and
will change to observed=True in a future version of pandas. Specify
observed=False to silence this warning and retain the current behavior
  age_product_pivot = data.pivot_table(index='Age', columns='Product',
aggfunc='size', fill_value=0)
```

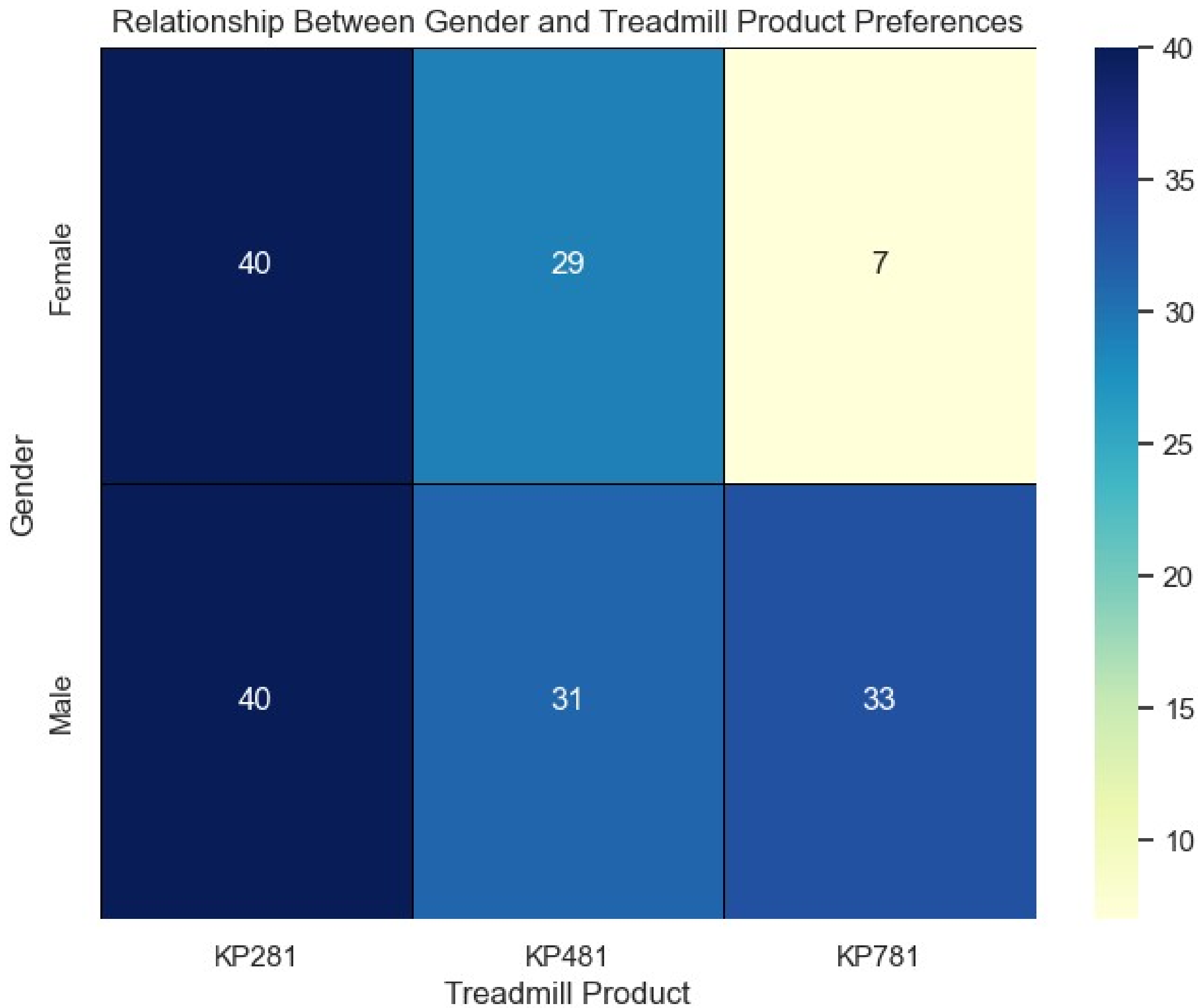


```
print("Relatiponship between gender and
product",gender_treadmill_crosstab)
```

```
Relatiponship between gender and product Product     KP281     KP481
KP781
Gender
Female   0.526316  0.381579  0.092105
Male     0.384615  0.298077  0.317308
```

```
# Pivot the data to prepare for the heatmap
gender_product_pivot = data.pivot_table(index='Gender',
columns='Product', aggfunc='size', fill_value=0)

# Create the heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(gender_product_pivot, cmap='YlGnBu', annot=True, fmt='d',
linewidths=0.5, linecolor='black')
plt.title('Relationship Between Gender and Treadmill Product
Preferences')
plt.xlabel('Treadmill Product')
plt.ylabel('Gender')
plt.show()
```

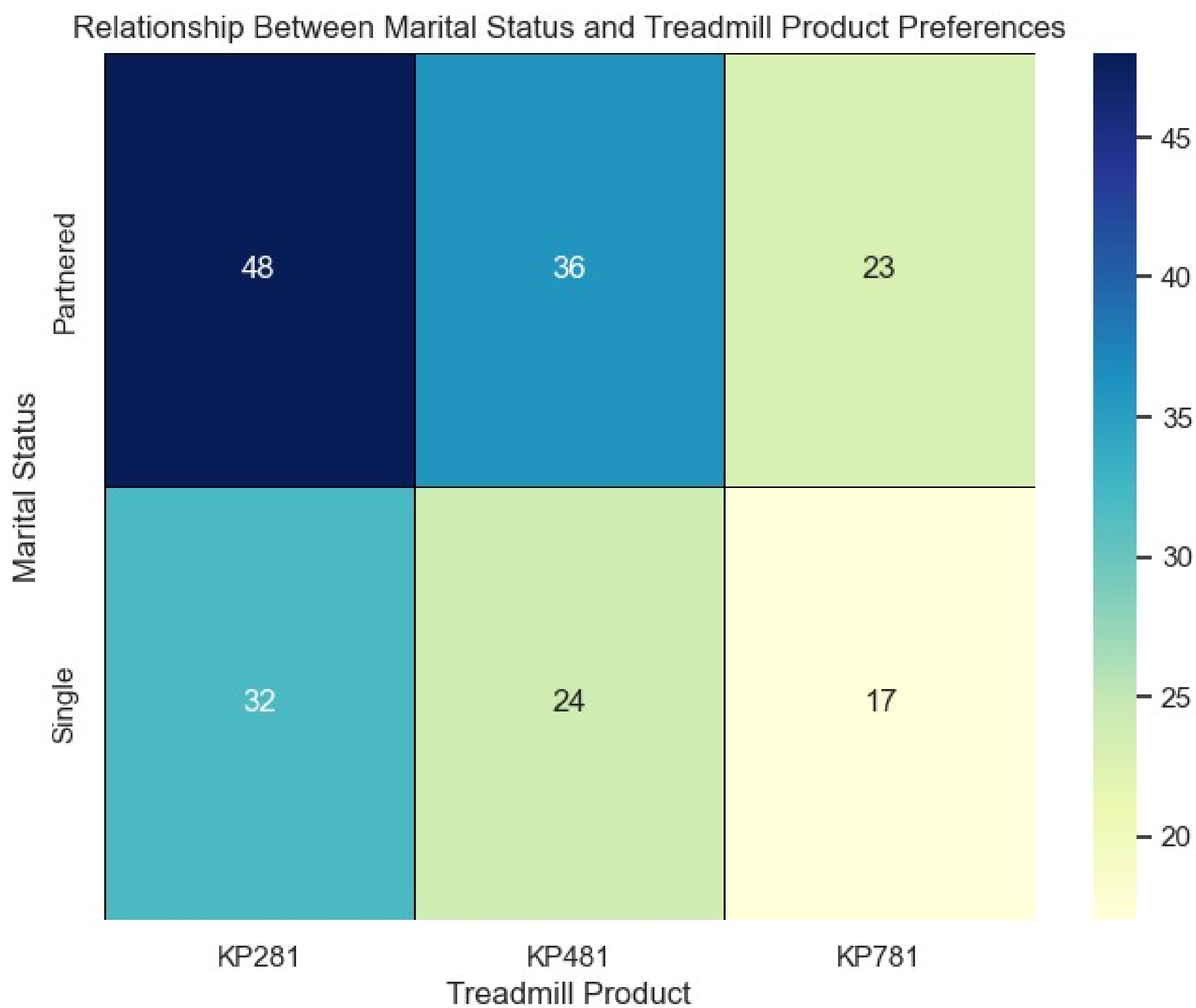
```
C:\Users\user\AppData\Local\Temp\ipykernel_14556\948853693.py:2:
FutureWarning: The default value of observed=False is deprecated and
will change to observed=True in a future version of pandas. Specify
observed=False to silence this warning and retain the current behavior
  gender_product_pivot = data.pivot_table(index='Gender',
columns='Product', aggfunc='size', fill_value=0)
```




```
print("Relatiponship between marital status and
product",marital_status_treadmill_crosstab)

Relatiponship between marital status and product Product
KP281     KP481     KP781
MaritalStatus
Partnered      0.448598  0.336449  0.214953
Single         0.438356  0.328767  0.232877

# Pivot the data to prepare for the heatmap
marital_product_pivot = data.pivot_table(index='MaritalStatus',
columns='Product', aggfunc='size', fill_value=0)

# Create the heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(marital_product_pivot, cmap='YlGnBu', annot=True, fmt='d',
linewidths=0.5, linecolor='black')
plt.title('Relationship Between Marital Status and Treadmill Product
```

```
Preferences')
plt.xlabel('Treadmill Product')
plt.ylabel('Marital Status')
plt.show()
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_14556\2445729968.py:2:
FutureWarning: The default value of observed=False is deprecated and
will change to observed=True in a future version of pandas. Specify
observed=False to silence this warning and retain the current behavior
  marital_product_pivot = data.pivot_table(index='MaritalStatus',
columns='Product', aggfunc='size', fill_value=0)
```



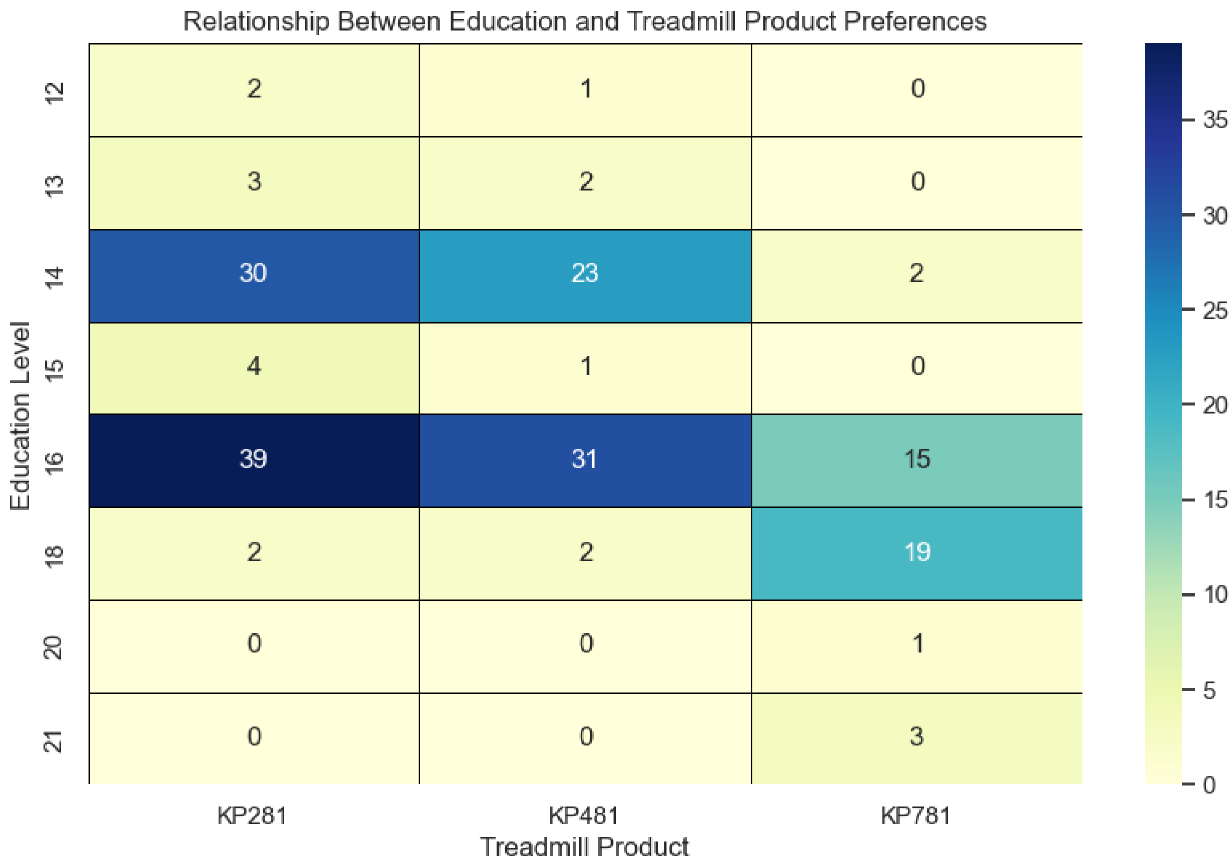

```
print("Relatiponship between education and
product",education_treadmill_crosstab)
```

```
Relatiponship between education and product Product       KP281
KP481     KP781
```

```
Education
12          0.666667  0.333333  0.000000
13          0.600000  0.400000  0.000000
14          0.545455  0.418182  0.036364
15          0.800000  0.200000  0.000000
16          0.458824  0.364706  0.176471
18          0.086957  0.086957  0.826087
20          0.000000  0.000000  1.000000
21          0.000000  0.000000  1.000000
```

```
# Pivot the data to prepare for the heatmap
education_product_pivot = data.pivot_table(index='Education',
columns='Product', aggfunc='size', fill_value=0)

# Create the heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(education_product_pivot, cmap='YlGnBu', annot=True,
fmt='d', linewidths=0.5, linecolor='black')
plt.title('Relationship Between Education and Treadmill Product
Preferences')
plt.xlabel('Treadmill Product')
plt.ylabel('Education Level')
plt.show()
```

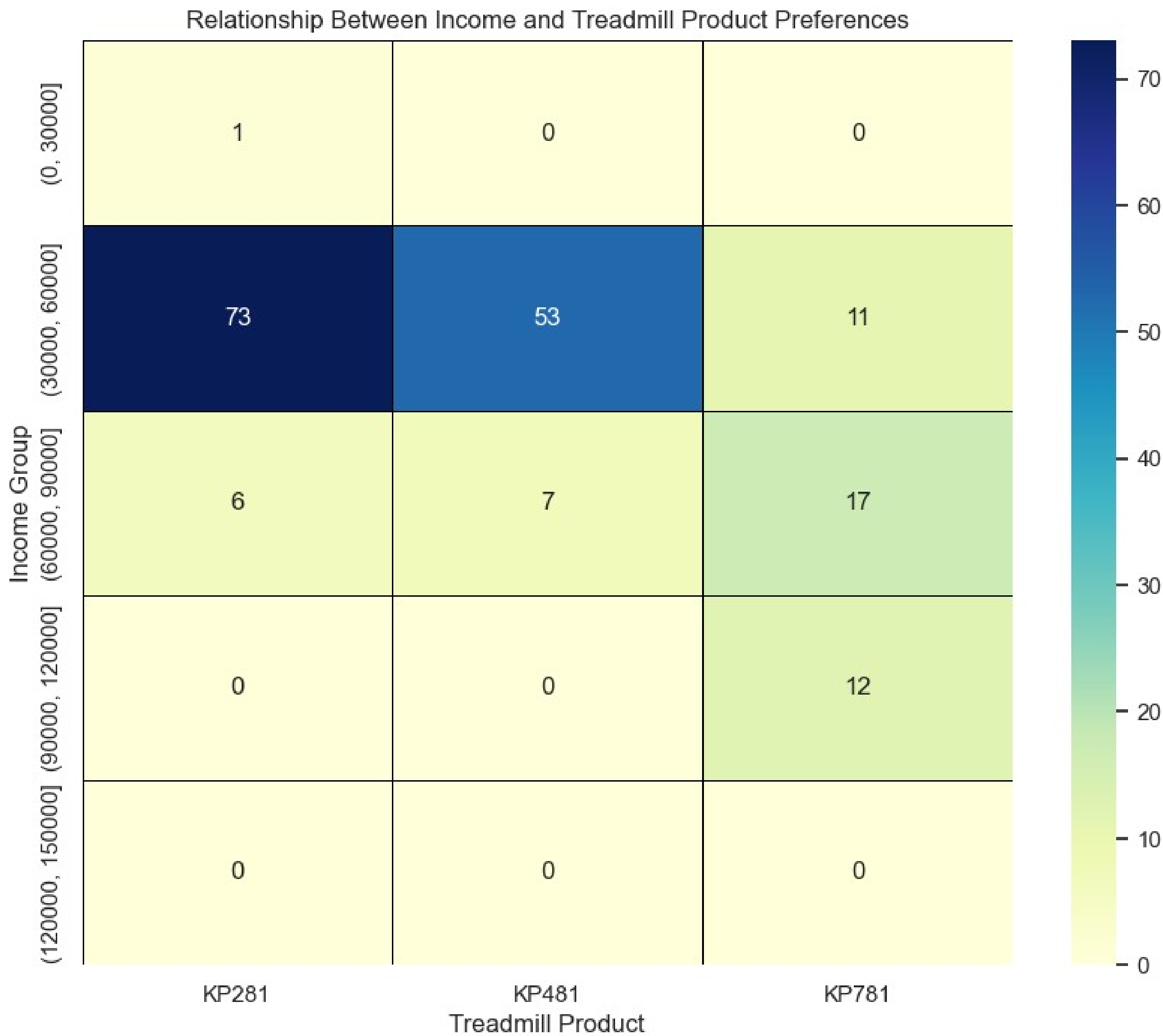
```
C:\Users\user\AppData\Local\Temp\ipykernel_14556\4109637894.py:2:
FutureWarning: The default value of observed=False is deprecated and
will change to observed=True in a future version of pandas. Specify
observed=False to silence this warning and retain the current behavior
  education_product_pivot = data.pivot_table(index='Education',
columns='Product', aggfunc='size', fill_value=0)
```




```
print("Relatiponship between income and
product",income_treadmill_crosstab)

Relatiponship between income and product Product  KP281  KP481  KP781
Income
29562      1.0    0.0    0.0
30699      1.0    0.0    0.0
31836      0.5    0.5    0.0
32973      0.6    0.4    0.0
34110      0.4    0.6    0.0
...        ...    ...    ...
95508      0.0    0.0    1.0
95866      0.0    0.0    1.0
99601      0.0    0.0    1.0
103336     0.0    0.0    1.0
104581     0.0    0.0    1.0

[62 rows x 3 columns]

# Define income bins
income_bins = [0, 30000, 60000, 90000, 120000, 150000]

# Create a new column for income bins
```

```python
data['Income Group'] = pd.cut(data['Income'], bins=income_bins)

# Pivot the data to prepare for the heatmap
income_product_pivot = data.pivot_table(index='Income Group',
columns='Product', aggfunc='size', fill_value=0)

# Create the heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(income_product_pivot, cmap='YlGnBu', annot=True, fmt='d',
linewidths=0.5, linecolor='black')
plt.title('Relationship Between Income and Treadmill Product
Preferences')
plt.xlabel('Treadmill Product')
plt.ylabel('Income Group')
plt.show()
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_14556\2005364524.py:8:
FutureWarning: The default value of observed=False is deprecated and
will change to observed=True in a future version of pandas. Specify
observed=False to silence this warning and retain the current behavior
  income_product_pivot = data.pivot_table(index='Income Group',
columns='Product', aggfunc='size', fill_value=0)
```



6.b Association Between Customer Characteristics and Product Preferences:

```
# Constructing contingency tables for age and product preferences
age_product_contingency = pd.crosstab(index=data['Age'],
columns=data['Product'])

# Computing conditional probabilities for age and product preferences
conditional_prob_age_product =
age_product_contingency.div(age_product_contingency.sum(axis=1),
axis=0)

# Constructing contingency tables for gender and product preferences
gender_product_contingency = pd.crosstab(index=data['Gender'],
columns=data['Product'])

# Computing conditional probabilities for gender and product
```

```
preferences
conditional_prob_gender_product =
gender_product_contingency.div(gender_product_contingency.sum(axis=1),
axis=0)

# Constructing contingency tables for education and product
preferences
education_product_contingency = pd.crosstab(index=data['Education'],
columns=data['Product'])

# Computing conditional probabilities for education and product
preferences
conditional_prob_education_product =
education_product_contingency.div(education_product_contingency.sum(ax
is=1), axis=0)

# Constructing contingency tables for marital status and product
preferences
marital_product_contingency = pd.crosstab(index=data['MaritalStatus'],
columns=data['Product'])

# Computing conditional probabilities for marital status and product
preferences
conditional_prob_marital_product =
marital_product_contingency.div(marital_product_contingency.sum(axis=1
), axis=0)

# Constructing contingency tables for income and product preferences
income_product_contingency = pd.crosstab(index=data['Income'],
columns=data['Product'])

# Computing conditional probabilities for income and product
preferences
conditional_prob_income_product =
income_product_contingency.div(income_product_contingency.sum(axis=1),
axis=0)

print("age_product_contingency table")
print("---------")
print(age_product_contingency)
print("age_product_conditional probability")
print("---------")
print(conditional_prob_age_product)
```

```
age_product_contingency table
---------
Product  KP281  KP481  KP781
Age
18           1      0      0
19           3      1      0
```

```
20            2     3     0
21            4     3     0
22            4     0     3
23            8     7     3
24            5     3     4
25            7    11     7
26            7     3     2
27            3     1     3
28            6     0     3
29            3     1     2
30            2     2     3
31            2     3     1
32            2     2     0
33            2     5     1
34            2     3     1
35            3     4     1
36            1     0     0
37            1     1     0
38            4     2     1
39            1     0     0
40            1     3     1
41            1     0     0
42            0     0     1
43            1     0     0
44            1     0     0
45            0     1     1
46            1     0     0
47            1     0     1
48            0     1     1
50            1     0     0
age_product_conditional probability
---------
Product     KP281     KP481     KP781
Age
18       1.000000  0.000000  0.000000
19       0.750000  0.250000  0.000000
20       0.400000  0.600000  0.000000
21       0.571429  0.428571  0.000000
22       0.571429  0.000000  0.428571
23       0.444444  0.388889  0.166667
24       0.416667  0.250000  0.333333
25       0.280000  0.440000  0.280000
26       0.583333  0.250000  0.166667
27       0.428571  0.142857  0.428571
28       0.666667  0.000000  0.333333
29       0.500000  0.166667  0.333333
30       0.285714  0.285714  0.428571
31       0.333333  0.500000  0.166667
32       0.500000  0.500000  0.000000
```

```
33        0.250000  0.625000  0.125000
34        0.333333  0.500000  0.166667
35        0.375000  0.500000  0.125000
36        1.000000  0.000000  0.000000
37        0.500000  0.500000  0.000000
38        0.571429  0.285714  0.142857
39        1.000000  0.000000  0.000000
40        0.200000  0.600000  0.200000
41        1.000000  0.000000  0.000000
42        0.000000  0.000000  1.000000
43        1.000000  0.000000  0.000000
44        1.000000  0.000000  0.000000
45        0.000000  0.500000  0.500000
46        1.000000  0.000000  0.000000
47        0.500000  0.000000  0.500000
48        0.000000  0.500000  0.500000
50        1.000000  0.000000  0.000000
```

```
print("gender_product_contingency table")
print("---------")
print(gender_product_contingency)
print("gender_product_conditional probability")
print("---------")
print(conditional_prob_gender_product)
```

```
gender_product_contingency table
---------
Product  KP281  KP481  KP781
Gender
Female      40     29      7
Male        40     31     33
gender_product_conditional probability
---------
Product     KP281     KP481     KP781
Gender
Female   0.526316  0.381579  0.092105
Male     0.384615  0.298077  0.317308
```

```
print("education_product_contingency table")
print("---------")
print(education_product_contingency )
print("education_product_conditional probability")
print("---------")
print(conditional_prob_education_product)
```

```
education_product_contingency table
---------
Product    KP281  KP481  KP781
Education
12             2      1      0
```

```
13                3      2      0
14               30     23      2
15                4      1      0
16               39     31     15
18                2      2     19
20                0      0      1
21                0      0      3
education_product_conditional probability
---------
Product       KP281     KP481     KP781
Education
12         0.666667  0.333333  0.000000
13         0.600000  0.400000  0.000000
14         0.545455  0.418182  0.036364
15         0.800000  0.200000  0.000000
16         0.458824  0.364706  0.176471
18         0.086957  0.086957  0.826087
20         0.000000  0.000000  1.000000
21         0.000000  0.000000  1.000000
```

```
print("marital_product_contingency table")
print("---------")
print(marital_product_contingency)
print("marital_product_conditional probability")
print("---------")
print(conditional_prob_marital_product)
```

```
marital_product_contingency table
---------
Product        KP281  KP481  KP781
MaritalStatus
Partnered         48     36     23
Single            32     24     17
marital_product_conditional probability
---------
Product           KP281     KP481     KP781
MaritalStatus
Partnered      0.448598  0.336449  0.214953
Single         0.438356  0.328767  0.232877
```

```
print("income_product_contingency table")
print("---------")
print(income_product_contingency)
print("income_product_conditional probability")
print("---------")
print(conditional_prob_income_product)
```

```
income_product_contingency table
---------
Product  KP281  KP481  KP781
```

```
Income
29562        1      0      0
30699        1      0      0
31836        1      1      0
32973        3      2      0
34110        2      3      0
...        ...    ...    ...
95508        0      0      1
95866        0      0      1
99601        0      0      1
103336       0      0      1
104581       0      0      2

[62 rows x 3 columns]
income_product_conditional probability
---------
Product  KP281  KP481  KP781
Income
29562      1.0    0.0    0.0
30699      1.0    0.0    0.0
31836      0.5    0.5    0.0
32973      0.6    0.4    0.0
34110      0.4    0.6    0.0
...        ...    ...    ...
95508      0.0    0.0    1.0
95866      0.0    0.0    1.0
99601      0.0    0.0    1.0
103336     0.0    0.0    1.0
104581     0.0    0.0    1.0

[62 rows x 3 columns]
```

8.Marginal Probabilities Calculations:

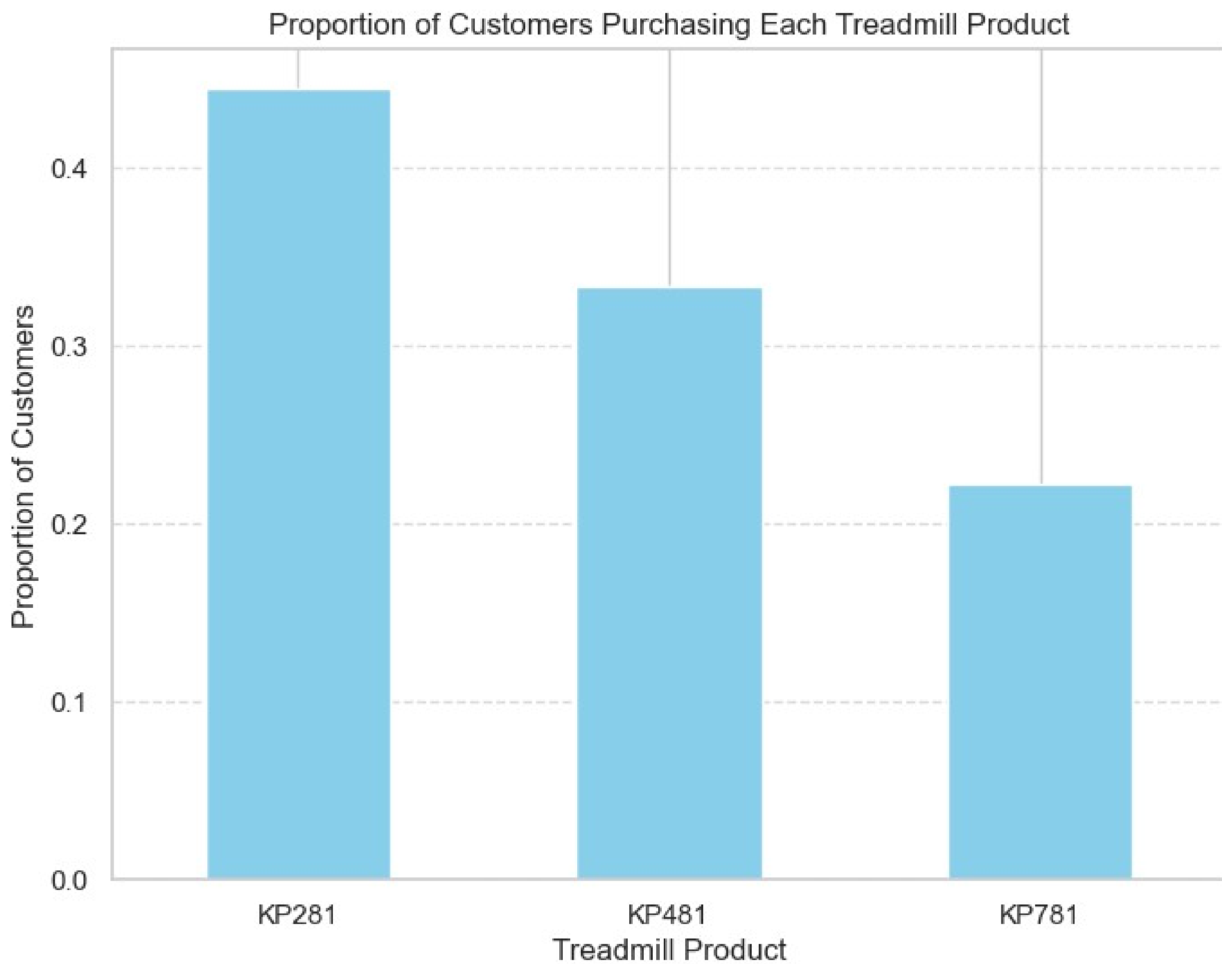
```
# 8.a Calculate marginal probabilities using crosstab
marginal_probs = pd.crosstab(index=data['MaritalStatus'],
columns=data['Product'], normalize='columns')

# Display marginal probabilities
print("Marginal probabilities of purchasing each treadmill product
within different customer segments:")
print(marginal_probs)
```

```
Marginal probabilities of purchasing each treadmill product within
different customer segments:
Product        KP281  KP481  KP781
MaritalStatus
Partnered        0.6    0.6  0.575
Single           0.4    0.4  0.425
```

```python
# 8.b Calculate marginal probability using value_counts
marginal_prob = data['Product'].value_counts(normalize=True)

# Display marginal probability
print("Marginal probability of purchasing each treadmill product:")
print(marginal_prob)
```

```
Marginal probability of purchasing each treadmill product:
Product
KP281    0.444444
KP481    0.333333
KP781    0.222222
Name: proportion, dtype: float64
```

```python
# Calculate proportion of customers purchasing each treadmill product
product_counts = data['Product'].value_counts(normalize=True)

# Visualize the distribution
plt.figure(figsize=(8, 6))
product_counts.plot(kind='bar', color='skyblue')
plt.title('Proportion of Customers Purchasing Each Treadmill Product')
plt.xlabel('Treadmill Product')
plt.ylabel('Proportion of Customers')
plt.xticks(rotation=0)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```


Proportion of Customers Purchasing Each Treadmill Product
Proportion of Customers
0.0
0.1
0.2
0.3
0.4
KP281
KP481
KP781
Treadmill Product