Problem statement:

The goal is to identify the characteristics of the target audience for each type of treadmill offered by AeroFit. This involves performing descriptive analytics to create customer profiles for each treadmill product and constructing contingency tables to compute conditional and marginal probabilities. The analysis aims to provide insights into customer behavior and preferences, facilitating better recommendations for new customers.

Shape of data:

```
print("Shape of Data:", df.shape)
```

```
Shape of Data: (180, 9)
```

Data types of attributes:

```
# Data types of attributes
print("\nData Types of Attributes:")
print(df.dtypes)
```

```
Data Types of Attributes:
Product          object
Age               int64
Gender           object
Education         int64
MaritalStatus    object
Usage             int64
Fitness           int64
Income            int64
Miles             int64
dtype: object
```

Conversion of categorical attributes:

```
# Convert categorical attributes to 'category' type if required
# For this dataset, we'll convert 'Gender' and 'MaritalStatus' to category type
df['Gender'] = df['Gender'].astype('category')
df['MaritalStatus'] = df['MaritalStatus'].astype('category')
print(df.dtypes)
```

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

Statistical Summary:

```
# Statistical Summary
print("\nStatistical Summary:")
print(df.describe())
```

```
Statistical Summary:
              Age   Education       Usage     Fitness         Income  \
count  180.000000  180.000000  180.000000  180.000000     180.000000
mean    28.788889   15.572222    3.455556    3.311111   53719.577778
std      6.943498    1.617055    1.084797    0.958869   16506.684226
min     18.000000   12.000000    2.000000    1.000000   29562.000000
25%     24.000000   14.000000    3.000000    3.000000   44058.750000
50%     26.000000   16.000000    3.000000    3.000000   50596.500000
75%     33.000000   16.000000    4.000000    4.000000   58668.000000
max     50.000000   21.000000    7.000000    5.000000  104581.000000

            Miles
count  180.000000
mean   103.194444
std     51.863605
min     21.000000
25%     66.000000
50%     94.000000
75%    114.750000
max    360.000000
```

```python
numeric_columns = df.select_dtypes(include=['number'])
print("\nDescriptive Statistics Analysis:")

# Mean
print("\nMean:")
print(numeric_columns.mean())

# Median
print("\nMedian:")
print(numeric_columns.median())

# Mode
print("\nMode:")
print(numeric_columns.mode().iloc[0])  # Only prints mode for the first row

# Standard Deviation
print("\nStandard Deviation:")
print(numeric_columns.std())

# Range
print("\nRange:")
print(numeric_columns.max() - numeric_columns.min())
```

```
Descriptive Statistics Analysis:

Mean:
Age             28.788889
Education       15.572222
Usage            3.455556
Fitness          3.311111
Income       53719.577778
Miles          103.194444
dtype: float64

Median:
Age             26.0
Education       16.0
Usage            3.0
Fitness          3.0
Income       50596.5
Miles           94.0
dtype: float64

Mode:
Age             25
Education       16
Usage            3
Fitness          3
Income       45480
Miles           85
Name: 0, dtype: int64

Standard Deviation:
Age              6.943498
Education        1.617055
Usage            1.084797
Fitness          0.958869
Income       16506.684226
Miles           51.863605
dtype: float64

Range:
Age             32
Education        9
Usage            5
Fitness          4
Income       75019
Miles          339
dtype: int64
```

Non-Graphical Analysis:

```
print("\nValue Counts")
for column in df.select_dtypes(include=['object', 'category']):
    print("\n", column, ":")
    print(df[column].value_counts())
```

```
Value Counts

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

 Gender :
Male      104
Female     76
Name: Gender, dtype: int64

 MaritalStatus :
Partnered    107
Single        73
Name: MaritalStatus, dtype: int64
```

Insights:

- The most popular treadmill model among customers is the KP281, with 80 purchases, followed by KP481 with 60 purchases, and KP781 with 40 purchases.
- There are more male customers (104) than female customers (76) who purchased treadmills from AeroFit.
- The majority of customers (107) are partnered, while 73 customers are single.

```python
print("\nUnique Attributes for Continuous Variables:")
for column in ['Age', 'Education', 'Usage', 'Income', 'Fitness', 'Miles']:
    print("\n", column, ":")
    print(df[column].unique())
```

```
Unique Attributes for Continuous Variables:

 Age :
[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]

 Education :
[14 15 12 13 16 18 20 21]

 Usage :
[3 2 4 5 6 7]

 Income :
[ 29562  31836  30699  32973  35247  37521  36384  38658  40932  34110
  39795  42069  44343  45480  46617  48891  53439  43206  52302  51165
  50028  54576  68220  55713  60261  67083  56850  59124  61398  57987
  64809  47754  65220  62535  48658  54781  48556  58516  53536  61006
  57271  52291  49801  62251  64741  70966  75946  74701  69721  83416
  88396  90886  92131  77191  52290  85906 103336  99601  89641  95866
 104581  95508]

 Fitness :
[4 3 2 1 5]

 Miles :
[112  75  66  85  47 141 103  94 113  38 188  56 132 169  64  53 106  95
 212  42 127  74 170  21 120 200 140 100  80 160 180 240 150 300 280 260
 360]
```

Insights:

- The age range of customers who purchased treadmills from AeroFit varies from 18 to 50 years.
- Customers' education levels range from 12 to 21 years of education.
- The average number of times customers plan to use the treadmill each week ranges from 2 to 7 times.
- The annual income of customers purchasing treadmills spans from $29,562 to $104,581.
- Customers' self-rated fitness levels range from 1 to 5, indicating a diverse range of fitness abilities among purchasers.
- Customers' expectations regarding the average number of miles to walk/run each week vary widely, from 21 to 360 miles.

```python
# Conditional probability: Probability of each gender given product purchased
print("\nConditional probability of each gender given product purchased:")
for product in df['Product'].unique():
    conditional_gender_probs = df[df['Product'] == product]['Gender'].value_counts(normalize=True) * 100
    print(f"\nFor {product}:")
    print(round(conditional_gender_probs, 2))
```

```
Conditional probability of each gender given product purchased:

For KP281:
Female    50.0
Male      50.0
Name: Gender, dtype: float64

For KP481:
Male      51.67
Female    48.33
Name: Gender, dtype: float64

For KP781:
Male      82.5
Female    17.5
Name: Gender, dtype: float64
```

Insights:

KP281:

- The probability of a customer being female or male given the purchase of the KP281 treadmill is equal at 50.0% each.
- This balanced distribution suggests that the KP281 model appeals to both genders equally, indicating a broad market appeal.

KP481:

- There's a slightly higher probability of a customer being male (51.67%) compared to female (48.33%) when purchasing the KP481 treadmill.
- While still relatively balanced, the KP481 model seems to attract slightly more male customers than female customers.

KP781:

- The probability of a customer being male (82.5%) is significantly higher than that of being female (17.5%) when purchasing the KP781 treadmill.
- This substantial difference suggests that the KP781 model may have features or marketing appeals that resonate more strongly with male customers, indicating potential gender-specific preferences or market segmentation opportunities.

```python
# Conditional probability: Probability of each marital status given product purchased
print("\nConditional probability of each marital status given product purchased:")
for product in df['Product'].unique():
    conditional_marital_probs = df[df['Product'] == product]['MaritalStatus'].value_counts(normalize=True) * 100
    print(f"\nFor {product}:")
    print(conditional_marital_probs)
```

```
Conditional probability of each marital status given product purchased:

For KP281:
Partnered    60.0
Single       40.0
Name: MaritalStatus, dtype: float64

For KP481:
Partnered    60.0
Single       40.0
Name: MaritalStatus, dtype: float64

For KP781:
Partnered    57.5
Single       42.5
Name: MaritalStatus, dtype: float64
```

KP281 and KP481:

- Both KP281 and KP481 have the same conditional probabilities for marital status.
- There is a balanced distribution between partnered and single customers, with each category representing 60.0% and 40.0%, respectively.
- This suggests that both treadmill models appeal similarly to customers regardless of their marital status.

KP781:

- The conditional probabilities for marital status differ slightly for KP781 compared to KP281 and KP481.
- While partnered customers still make up the majority at 57.5%, single customers account for a higher proportion at 42.5%.
- This indicates that the KP781 model may have features or marketing appeals that resonate slightly more with single customers, though partnered customers still constitute a significant portion of the purchasers.

```python
# Conditional probability: Probability of each age group given product purchased
print("\nConditional probability of each age group given product purchased:")
for product in df['Product'].unique():
    print(f"\nFor {product}:")
    age_group_probabilities = df[df['Product'] == product]['Age'].value_counts(normalize=True) * 100
    print(round(age_group_probabilities, 2))
```

Conditional probability of each age group given product purchased:

For KP281:
23   10.00
25   8.75
26   8.75
28   7.50
24   6.25
38   5.00
21   5.00
22   5.00
29   3.75
19   3.75
27   3.75
35   3.75
34   2.50
33   2.50
32   2.50
31   2.50
30   2.50
20   2.50
41   1.25
47   1.25
46   1.25
44   1.25
43   1.25
18   1.25
40   1.25
39   1.25
37   1.25
36   1.25
50   1.25
Name: Age, dtype: float64

```
For KP781:
25    17.5
24    10.0
22     7.5
27     7.5
28     7.5
30     7.5
23     7.5
26     5.0
29     5.0
40     2.5
47     2.5
45     2.5
42     2.5
31     2.5
38     2.5
35     2.5
34     2.5
33     2.5
48     2.5
Name: Age, dtype: float64

For KP481:
25    18.33
23    11.67
33     8.33
35     6.67
31     5.00
21     5.00
24     5.00
26     5.00
40     5.00
20     5.00
34     5.00
38     3.33
30     3.33
32     3.33
45     1.67
19     1.67
37     1.67
29     1.67
27     1.67
48     1.67
Name: Age, dtype: float64
```

Insights:

KP281:

- The highest conditional probability is for customers aged 23, representing 10.00% of purchases for this product.

- Customers aged between 20 and 29 years old make up the majority of purchasers, with probabilities ranging from 3.75% to 10.00%.
- This suggests that the KP281 treadmill model is particularly popular among younger adults, with a peak in purchases around the age of 23.

KP481:

- The highest conditional probability is for customers aged 25, representing 18.33% of purchases for this product.
- Customers aged between 20 and 40 years old are the primary purchasers, with probabilities ranging from 1.67% to 18.33%.
- This indicates a broader age range of purchasers for the KP481 treadmill model, with a significant proportion of customers in their mid-20s.

KP781:

- The highest conditional probability is for customers aged 25, representing 17.5% of purchases for this product.
- Customers aged between 22 and 31 years old are the primary purchasers, with probabilities ranging from 2.5% to 17.5%.
- Similar to KP481, there is a broader age range of purchasers for the KP781 treadmill model, with a concentration around the mid to late 20s.

```
# Conditional probability: Probability of each education level given product purchased
print("\nConditional probability of each education level given product purchased:")
for product in df['Product'].unique():
    print(f"\nFor {product}:")
    education_probabilities = df[df['Product'] == product]['Education'].value_counts(normalize=True) * 100
    print(round(education_probabilities, 2))
```

```
Conditional probability of each education level given product purchased:

For KP281:
16    48.75
14    37.50
15     5.00
13     3.75
12     2.50
18     2.50
Name: Education, dtype: float64

For KP481:
16    51.67
14    38.33
13     3.33
18     3.33
12     1.67
15     1.67
Name: Education, dtype: float64

For KP781:
18    47.5
16    37.5
21     7.5
14     5.0
20     2.5
Name: Education, dtype: float64
```

Insights:

KP281:

- The highest conditional probability is for customers with an education level of 16, representing 48.75% of purchases for this product.
- Customers with education levels of 14 and 15 also make up significant proportions of purchasers, at 37.50% and 5.00%, respectively.
- This suggests that the KP281 treadmill model is most popular among customers with higher education levels, particularly those with an education level of 16.

KP481:

- The highest conditional probability is for customers with an education level of 16, representing 51.67% of purchases for this product.
- Similar to KP281, customers with education levels of 14 also make up a significant proportion of purchasers, at 38.33%.
- This indicates that the education level distribution for KP481 is similar to KP281, with a higher concentration among customers with education level 16.

KP781:

- The highest conditional probability is for customers with an education level of 18, representing 47.5% of purchases for this product.
- Customers with education levels of 16 also make up a significant proportion of purchasers, at 37.5%.

- Unlike KP281 and KP481, KP781 has a higher proportion of purchasers with an education level of 18, indicating potential preferences for this product among individuals with higher education levels.

```
# Conditional probability: Probability of each usage frequency given product purchased
print("\nConditional probability of each usage frequency given product purchased:")
for product in df['Product'].unique():
    print(f"\nFor {product}:")
    usage_probabilities = df[df['Product'] == product]['Usage'].value_counts(normalize=True) * 100
    print(round(usage_probabilities, 2))
```

```
Conditional probability of each usage frequency given product purchased:

For KP281:
3    46.25
4    27.50
2    23.75
5     2.50
Name: Usage, dtype: float64

For KP481:
3    51.67
2    23.33
4    20.00
5     5.00
Name: Usage, dtype: float64

For KP781:
4    45.0
5    30.0
6    17.5
7     5.0
3     2.5
Name: Usage, dtype: float64
```

Insights:

KP281:

- The highest conditional probability is for customers who plan to use the treadmill 3 times per week, representing 46.25% of purchases for this product.
- Usage frequencies of 2 and 4 times per week also account for significant proportions of purchasers, at 23.75% and 27.50%, respectively.
- This suggests that the KP281 treadmill model is most popular among customers who plan to use it 3 times per week, with a substantial proportion also using it 4 times per week.

KP481:

- The highest conditional probability is for customers who plan to use the treadmill 3 times per week, representing 51.67% of purchases for this product.

- Usage frequencies of 2 and 4 times per week also account for significant proportions of purchasers, at 23.33% and 20.00%, respectively.
- Similar to KP281, KP481 is most popular among customers who plan to use it 3 times per week, with a substantial proportion also using it 4 times per week.

KP781:

- The highest conditional probability is for customers who plan to use the treadmill 4 times per week, representing 45.0% of purchases for this product.
- Usage frequencies of 5 and 6 times per week also account for significant proportions of purchasers, at 30.0% and 17.5%, respectively.
- This suggests that the KP781 treadmill model is most popular among customers who plan to use it 4 times per week, with a substantial proportion also using it 5 times per week.

```
# Conditional probability: Probability of each fitness level given product purchased
print("\nConditional probability of each fitness level given product purchased:")
for product in df['Product'].unique():
    print(f"\nFor {product}:")
    fitness_probabilities = df[df['Product'] == product]['Fitness'].value_counts(normalize=True) * 100
    print(round(fitness_probabilities, 2))
```

```
Conditional probability of each fitness level given product purchased:

For KP281:
3    67.50
2    17.50
4    11.25
5     2.50
1     1.25
Name: Fitness, dtype: float64

For KP481:
3    65.00
2    20.00
4    13.33
1     1.67
Name: Fitness, dtype: float64

For KP781:
5    72.5
4    17.5
3    10.0
Name: Fitness, dtype: float64
```

Insights:

KP281:

- The highest conditional probability is for customers with a fitness level of 3, representing 67.50% of purchases for this product.
- Fitness levels 2 and 4 also account for significant proportions of purchasers, at 17.50% and 11.25%, respectively.

- This suggests that the KP281 treadmill model is most popular among customers who rate their fitness level as moderate (level 3), with a smaller proportion of customers rating their fitness as lower (level 2) or higher (level 4).

KP481:

- The highest conditional probability is for customers with a fitness level of 3, representing 65.00% of purchases for this product.
- Fitness levels 2 and 4 also account for significant proportions of purchasers, at 20.00% and 13.33%, respectively.
- Similar to KP281, KP481 is most popular among customers who rate their fitness level as moderate (level 3), with smaller proportions at lower (level 2) and higher (level 4) fitness levels.

KP781:

- The highest conditional probability is for customers with a fitness level of 5, representing 72.5% of purchases for this product.
- Fitness levels 4 and 3 also account for smaller proportions of purchasers, at 17.5% and 10.0%, respectively.
- This indicates that the KP781 treadmill model is primarily popular among customers who rate their fitness level as high (level 5), with fewer customers at lower fitness levels.

```
# Conditional probability: Probability of each expected miles given product purchased
print("\nConditional probability of each expected miles given product purchased:")
for product in df['Product'].unique():
    print(f"\nFor {product}:")
    miles_probabilities = df[df['Product'] == product]['Miles'].value_counts(normalize=True) * 100
    print(round(miles_probabilities, 2))
```

```
Conditional probability of each expected miles given product purchased:

For KP281:
85     20.00
75     12.50
66     12.50
47     11.25
94     10.00
113    10.00
56      7.50
103     3.75
38      3.75
141     2.50
132     2.50
112     1.25
188     1.25
169     1.25
Name: Miles, dtype: float64

For KP481:
95     20.00
85     18.33
106    13.33
53     11.67
64     10.00
127     8.33
42      6.67
74      5.00
170     3.33
212     1.67
21      1.67
Name: Miles, dtype: float64

For KP781:
100    17.5
200    15.0
180    15.0
160    12.5
150    10.0
120     7.5
106     2.5
140     2.5
80      2.5
240     2.5
170     2.5
300     2.5
280     2.5
260     2.5
360     2.5
Name: Miles, dtype: float64
```

Insights:

KP281:

- The most common expected miles for customers purchasing the KP281 treadmill are 85 (20.00%), followed by 75 and 66 (12.50% each).

- There is a diverse range of expected miles, with several values contributing smaller proportions, such as 47, 94, 113, etc.
- This suggests that customers purchasing the KP281 treadmill have varying expectations regarding the distance they plan to walk/run each week, with a significant portion aiming for around 85 miles.

KP481:

- The most common expected miles for customers purchasing the KP481 treadmill are 95 (20.00%), followed closely by 85 (18.33%).
- Similar to KP281, there is a diverse range of expected miles, with several values contributing smaller proportions.
- This indicates that customers purchasing the KP481 treadmill also have varying expectations regarding the distance they plan to walk/run each week, with a significant portion aiming for around 95 or 85 miles.

KP781:

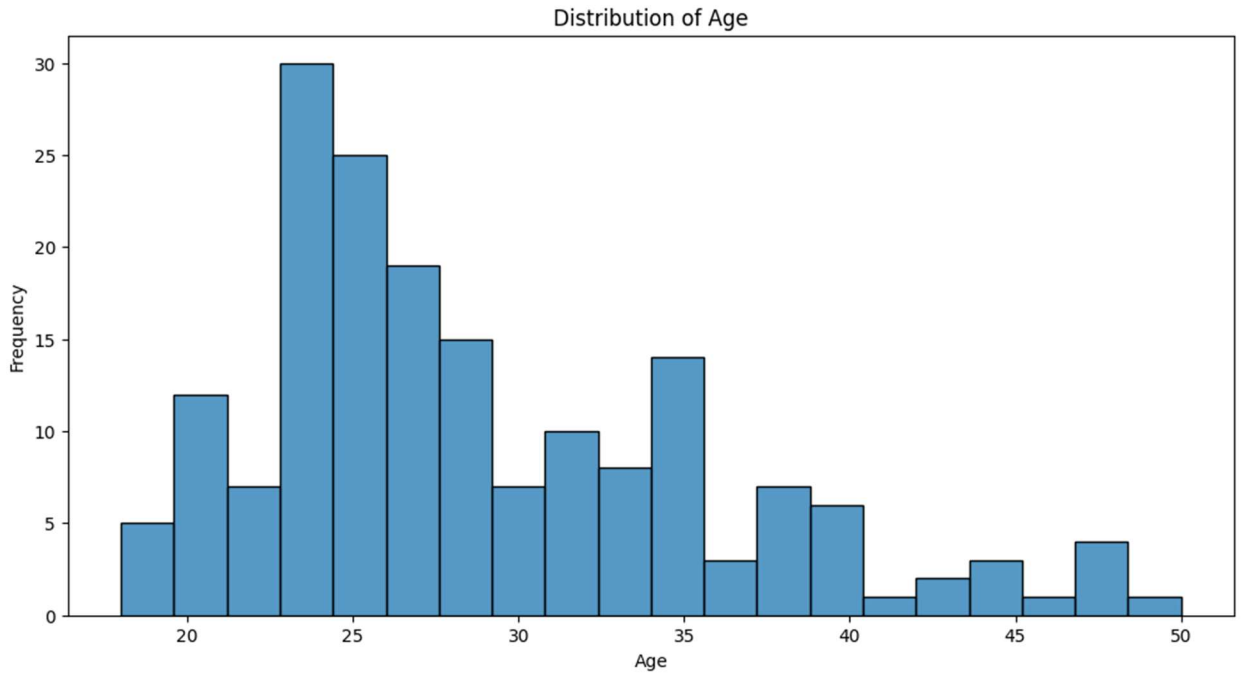
- The most common expected miles for customers purchasing the KP781 treadmill are 100 (17.5%), followed by 200 and 180 (15.0% each).
- Like the other models, there is a diverse range of expected miles, with several values contributing smaller proportions.
- This suggests that customers purchasing the KP781 treadmill typically have higher expectations for the distance they plan to walk/run each week, with a significant portion aiming for around 100, 200, or 180 miles.

Graphical Analysis:

Univariate analysis:

Distribution of Age:

```
plt.figure(figsize=(12, 6))
sns.histplot(df['Age'], kde=False, bins=20)
plt.title('Distribution of Age')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```

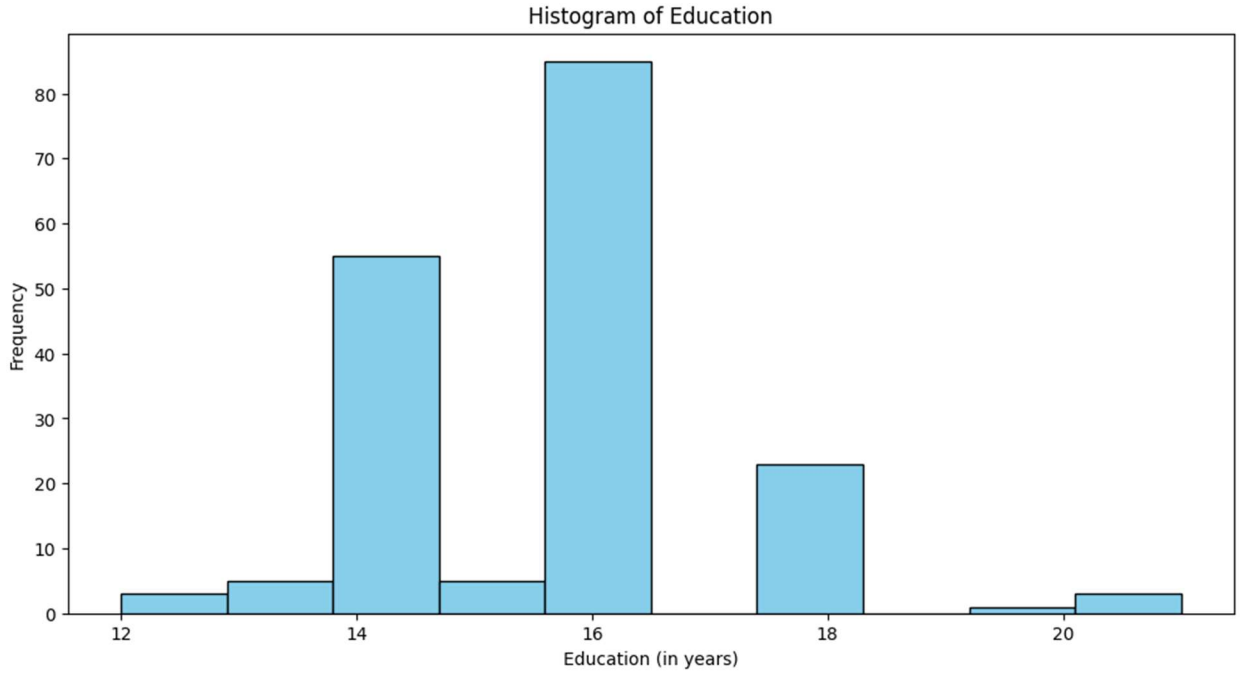


Insights:

- The age distribution of customers purchasing treadmills from AeroFit ranges from 18 to 50 years.

Distribution of Education in Years:

```
plt.figure(figsize=(12, 6))
plt.hist(df['Education'], bins=10, color='skyblue', edgecolor='black')
plt.title('Histogram of Education')
plt.xlabel('Education (in years)')
plt.ylabel('Frequency')
plt.show()
```

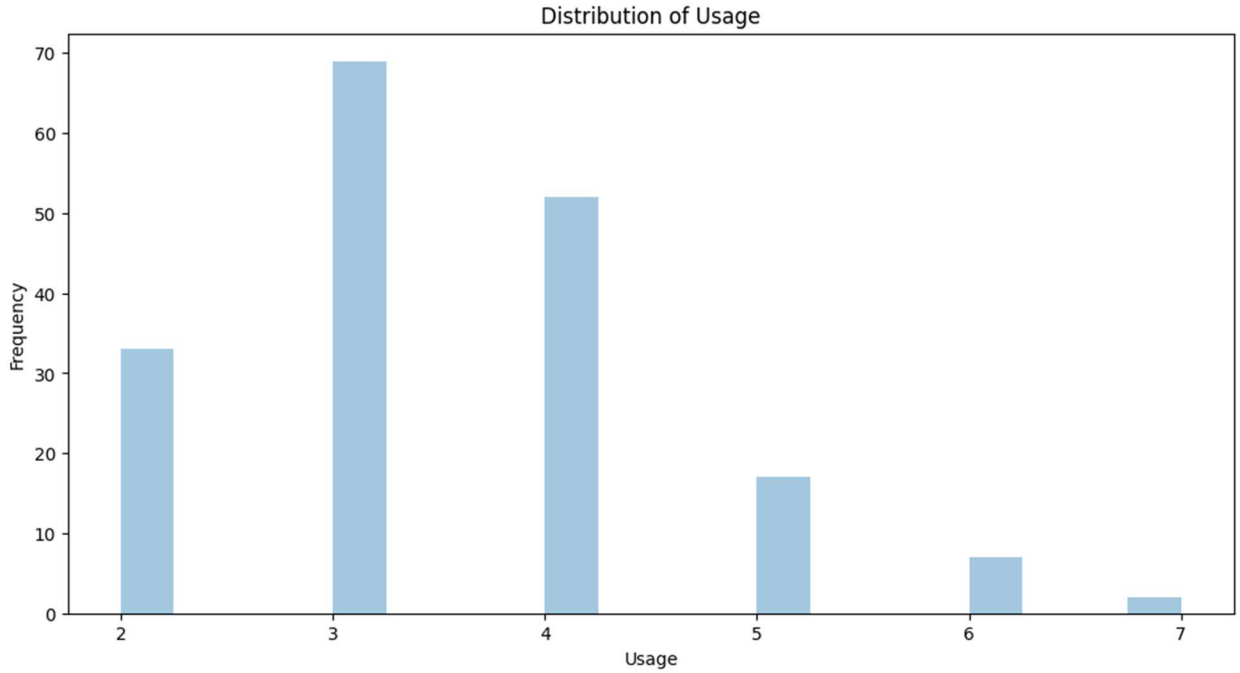


Insights:

- The education levels of customers purchasing AeroFit's products range from 12 to 21 years.
- The majority of customers have education levels of 14, 15, or 16 years, representing a significant portion of the customer base.
- Education levels of 14, 15, and 16 years are most prevalent among customers, indicating that a considerable portion of AeroFit's target market has completed high school or attained an equivalent level of education.
- Less common education levels include 12, 13, 18, 20, and 21 years, suggesting a smaller but still notable portion of customers with either lower or higher educational attainment.

Distribution of Usage:

```
plt.figure(figsize=(12, 6))
sns.distplot(df['Usage'], kde=False, bins=20)
plt.title('Distribution of Usage')
plt.xlabel('Usage')
plt.ylabel('Frequency')
plt.show()
```

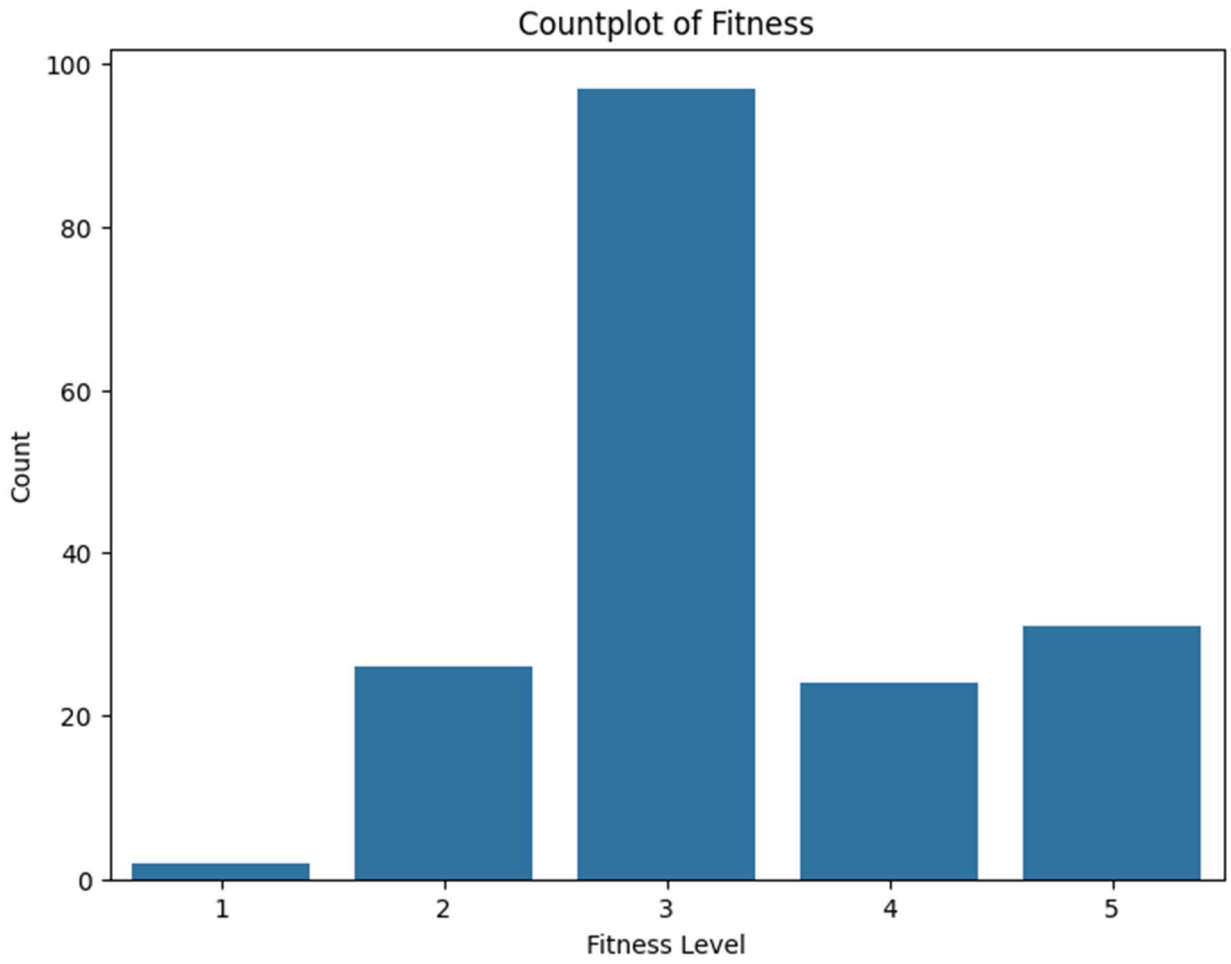


Insights:

- The usage frequency of AeroFit's products varies from 2 to 7 times per week, indicating different levels of engagement among customers.
- Most customers use the treadmill 3, 4, or 5 times per week, suggesting a common usage pattern among the majority of AeroFit's customer base.
- Usage frequencies of 3, 4, and 5 times per week are the most prevalent among customers, indicating that these are preferred routines for many individuals.
- Less common usage frequencies include 2, 6, and 7 times per week, suggesting that while some customers engage less frequently, there is still a subset of customers with more intensive usage habits.
- The predominance of 3, 4, and 5 times per week suggests consistent engagement with AeroFit's products among a significant portion of the customer base.

Distribution of Fitness:

```
plt.figure(figsize=(8, 6))
sns.countplot(x='Fitness', data=df)
plt.title('Countplot of Fitness')
plt.xlabel('Fitness Level')
plt.ylabel('Count')
plt.show()
```

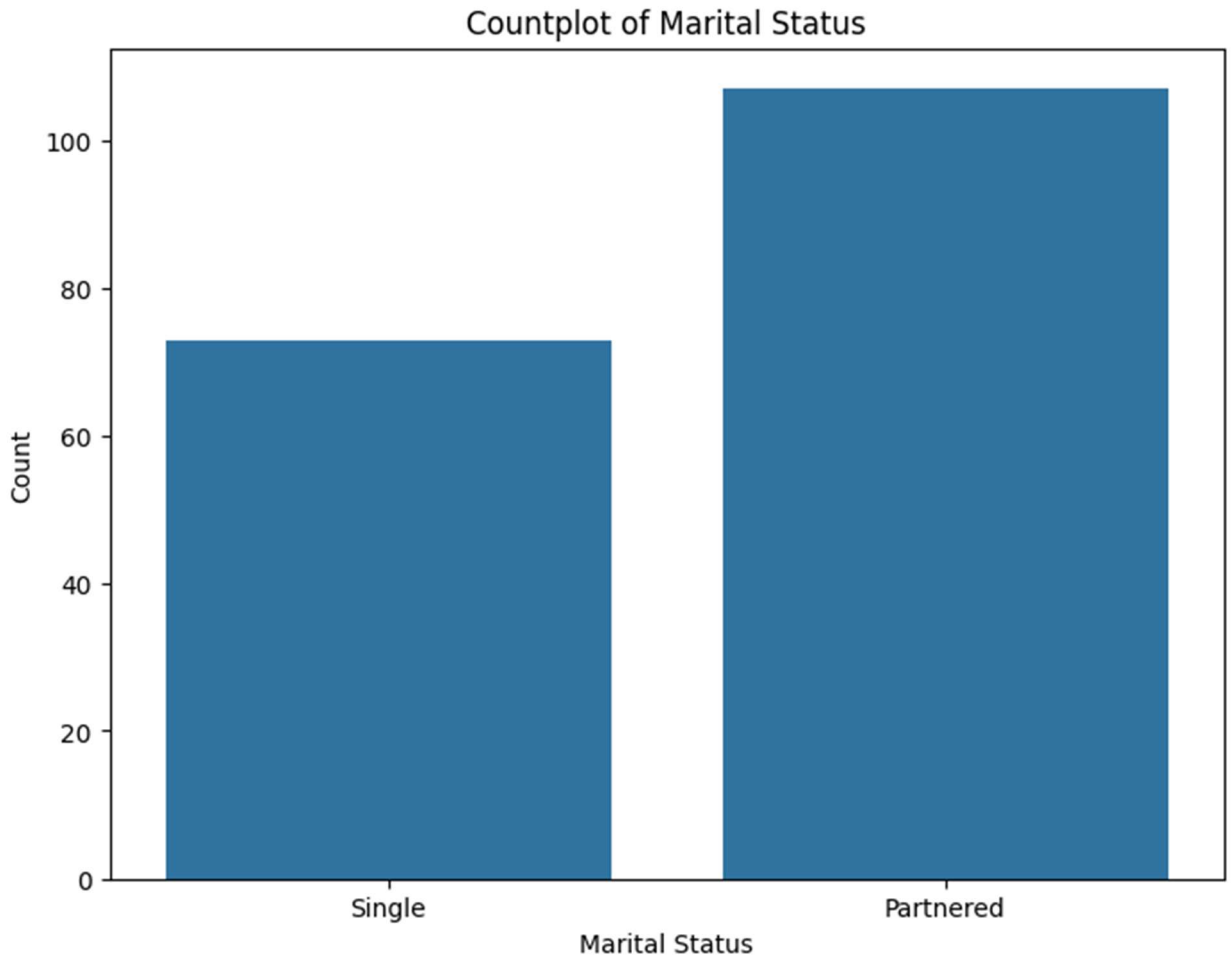


Insights:

- The fitness ratings provided by customers range from 1 to 5, indicating a diverse range of self-perceived fitness levels among AeroFit's clientele.
- The majority of customers rate their fitness levels as 3 or 4, suggesting that a significant portion of AeroFit's customer base considers themselves to be moderately fit.
- Ratings of 3 and 4 are the most prevalent among customers, indicating that many individuals perceive themselves to be in reasonably good shape.
- While moderate fitness levels are prevalent, there is also a notable portion of customers who rate their fitness as 2 or lower.
- Ratings of 5 indicate that some customers perceive themselves to be in excellent shape, reflecting a segment of the customer base with high fitness aspirations.

Distribution of Marital Status:

```
plt.figure(figsize=(8, 6))
sns.countplot(x='MaritalStatus', data=df)
plt.title('Countplot of Marital Status')
plt.xlabel('Marital Status')
plt.ylabel('Count')
plt.show()
```

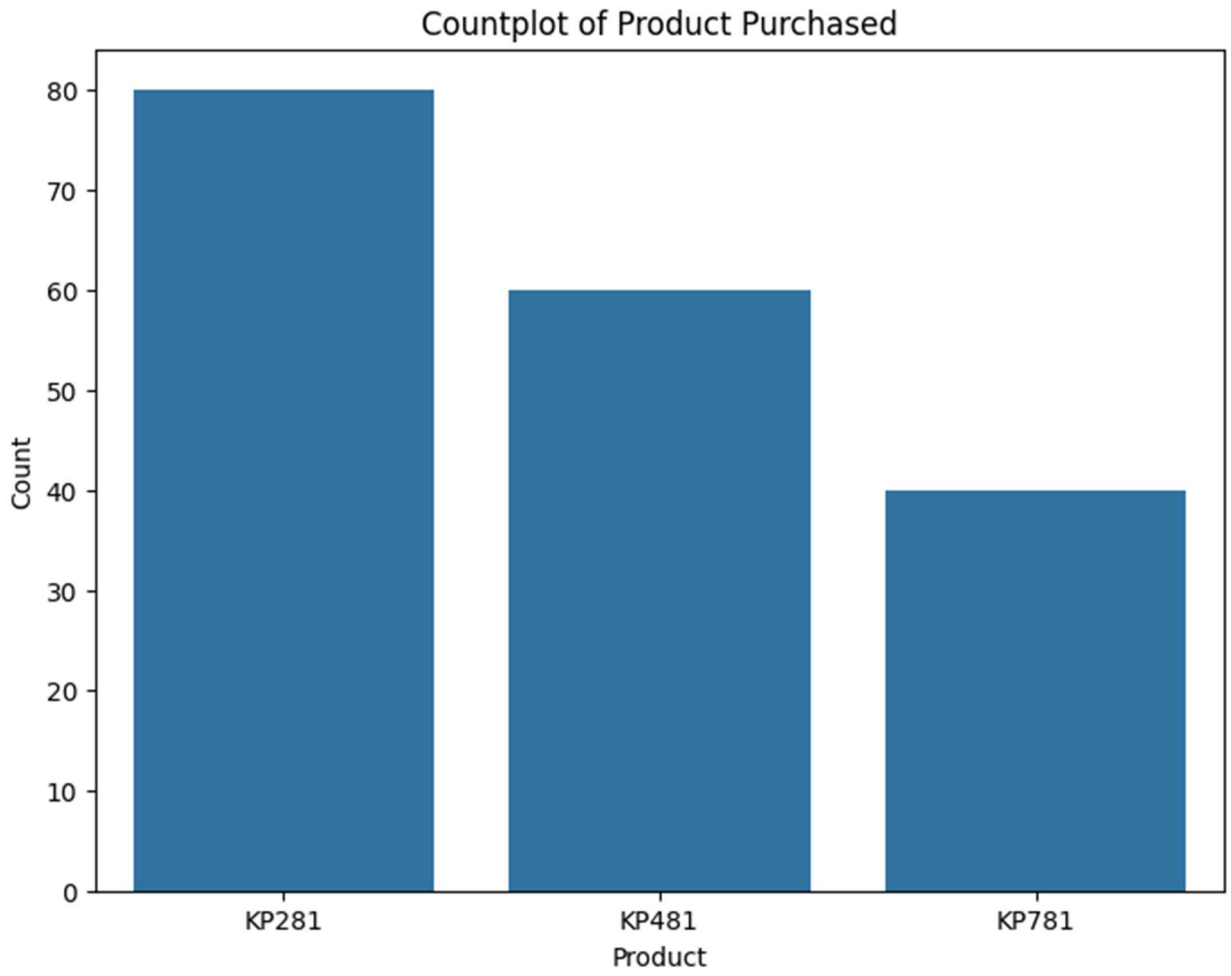


Insights:

- The majority of individuals in the dataset are partnered, with 107 out of 180.
- Single individuals represent a significant portion of the dataset, comprising 73 out of 180.
- This suggests that AeroFit's customer base includes both partnered and single individuals, indicating a diverse customer demographic.
- AeroFit may need to tailor marketing strategies and product offerings to cater to both partnered and single customers effectively.

Distribution of Products:

```
plt.figure(figsize=(8, 6))
sns.countplot(x='Product', data=df)
plt.title('Countplot of Product Purchased')
plt.xlabel('Product')
plt.ylabel('Count')
plt.show()
```



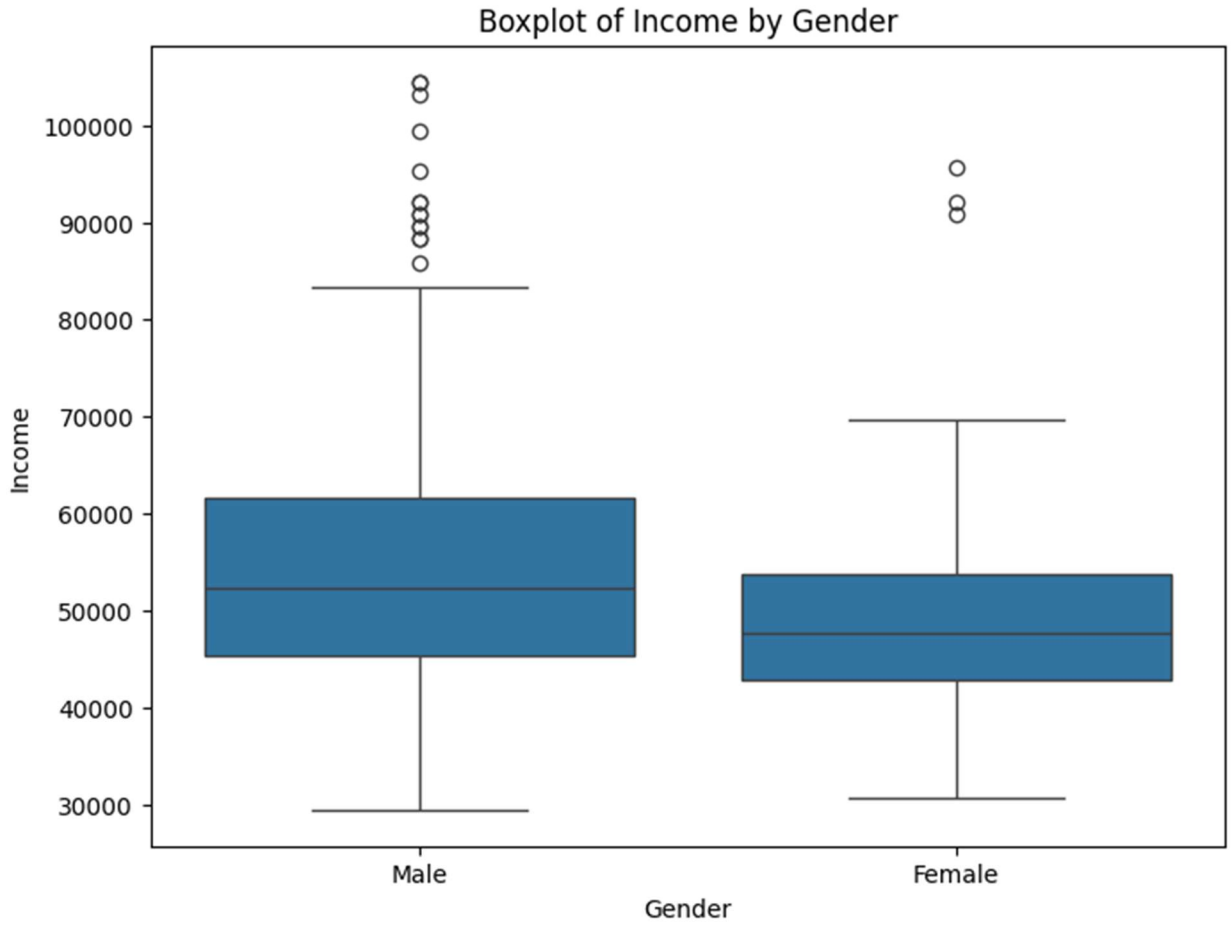
Insights:

- he product "KP281" has the highest sales, with 80 units sold.
- "KP481" follows closely behind with 60 units sold.
- "KP781" has the lowest sales among the three products, with 40 units sold.
- AeroFit should investigate the reasons behind the varying sales figures for each product and consider adjusting marketing strategies or product features accordingly.

Bivariate analysis:

Income by Gender:

```
plt.figure(figsize=(8, 6))
sns.boxplot(x='Gender', y='Income', data=df)
plt.title('Boxplot of Income by Gender')
plt.xlabel('Gender')
plt.ylabel('Income')
plt.show()
```

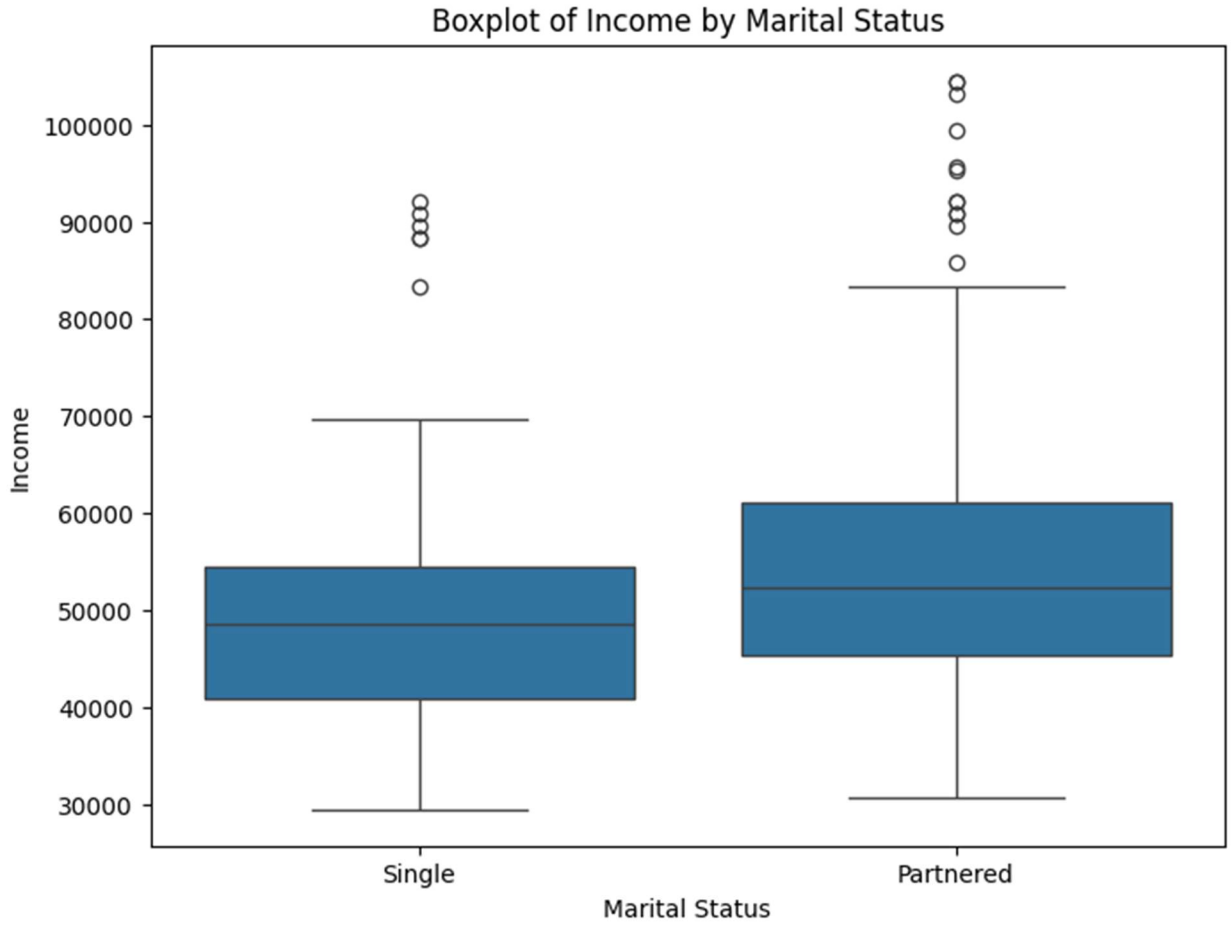


Insights:

- The dataset contains information on income distribution based on gender, with entries for both male and female individuals.
- The count column indicates the frequency of occurrence for each income level within each gender category.
- Both male and female individuals are represented across various income brackets, with differing counts at different income levels.
- There are some income levels where no individuals are represented for either gender, suggesting potential gaps in customer demographics or data collection.
- Analyzing income distribution by gender can help AeroFit understand the purchasing power and preferences of different customer segments, aiding in targeted marketing and product development efforts.

Income By Marital Status:

```
plt.figure(figsize=(8, 6))
sns.boxplot(x='MaritalStatus', y='Income', data=df)
plt.title('Boxplot of Income by Marital Status')
plt.xlabel('Marital Status')
plt.ylabel('Income')
plt.show()
```



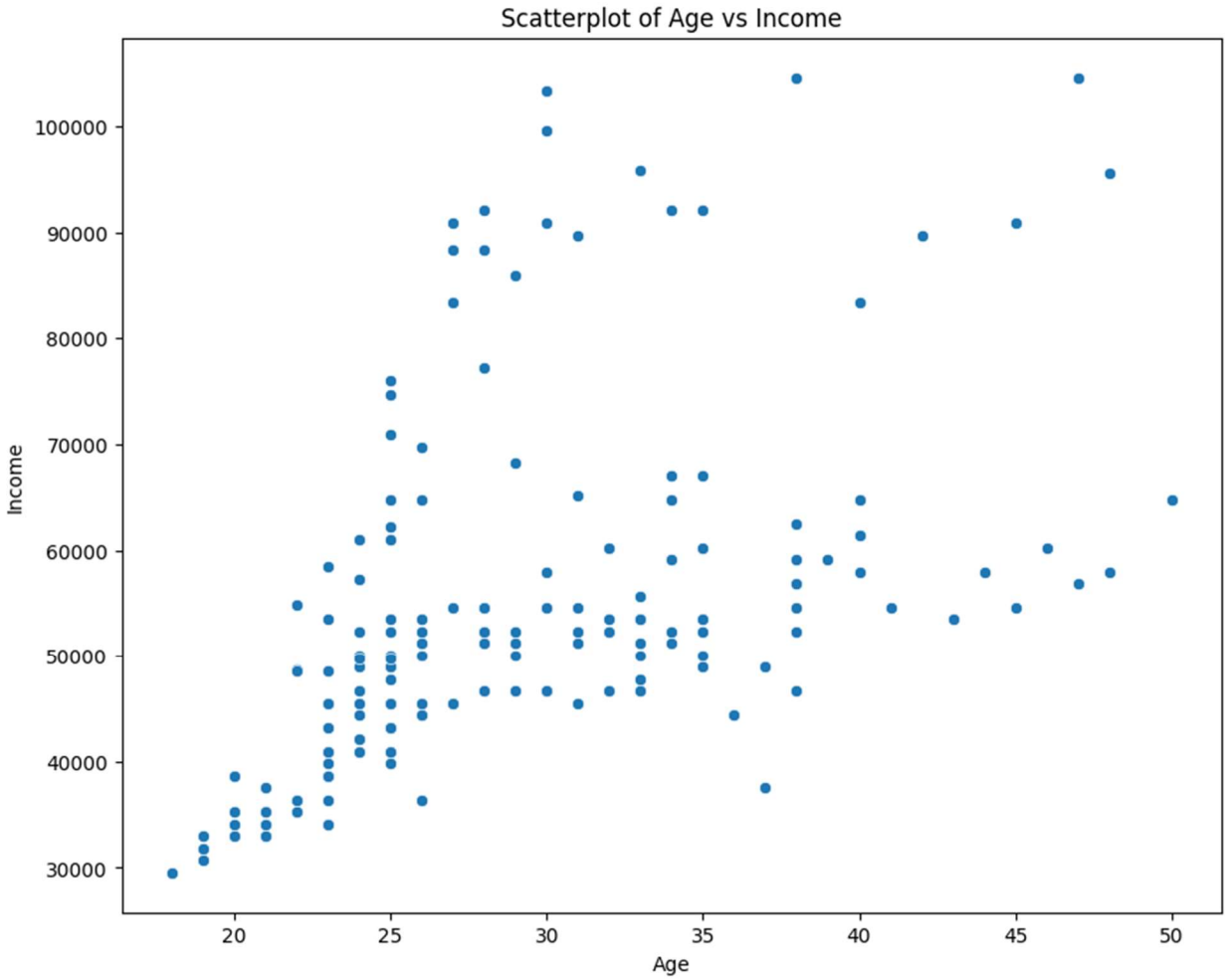
Insights:

- The dataset includes information on income distribution based on marital status, with entries for both partnered and single individuals.
- The count column indicates the frequency of occurrence for each income level within each marital status category.
- Partnered individuals appear to have a wider range of income levels compared to single individuals, as evidenced by the spread of counts across various income brackets.
- There are some income levels where no individuals are represented, suggesting potential gaps in customer demographics or data collection.

- Understanding income distribution by marital status can help AeroFit tailor marketing strategies and product offerings to different customer segments.

Age by Income:

```
plt.figure(figsize=(10, 8))
sns.scatterplot(x='Age', y='Income', data=df)
plt.title('Scatterplot of Age vs Income')
plt.xlabel('Age')
plt.ylabel('Income')
plt.show()
```



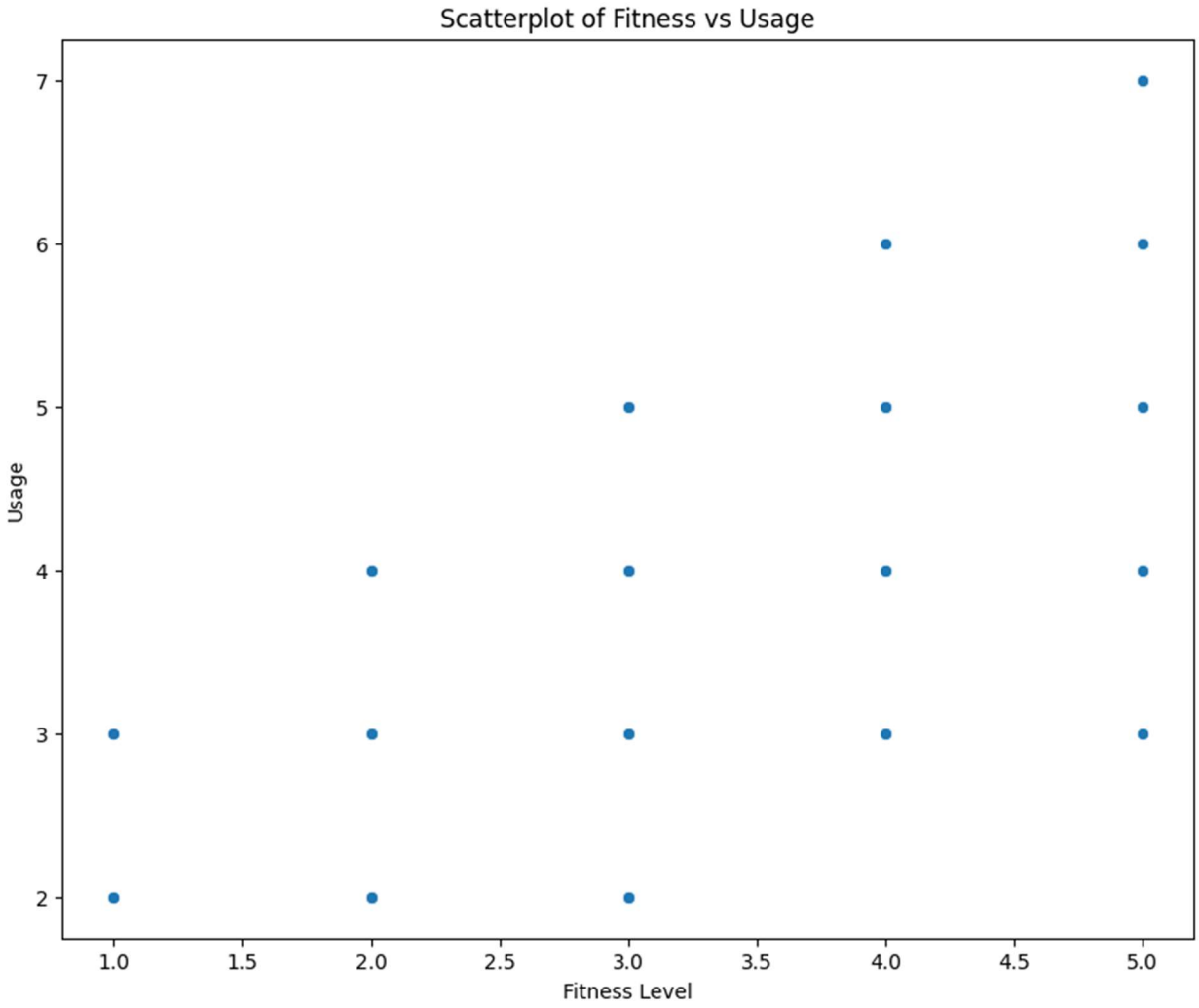
Insights:

- Customers of various ages and income levels are represented.
- Income distribution appears to be spread across different age groups, with no apparent concentration in any specific age range.

- There is variability in income within each age group, indicating diverse purchasing power among customers of similar ages.
- The dataset includes individuals as young as 18 and as old as 50, showcasing AeroFit's appeal to a wide age demographic.
- The count column represents the frequency of occurrence for each age-income pair, providing insight into the distribution of customers across different age and income brackets.

Fitness by Usage:

```
plt.figure(figsize=(10, 8))
sns.scatterplot(x='Fitness', y='Usage', data=df)
plt.title('Scatterplot of Fitness vs Usage')
plt.xlabel('Fitness Level')
plt.ylabel('Usage')
plt.show()
```

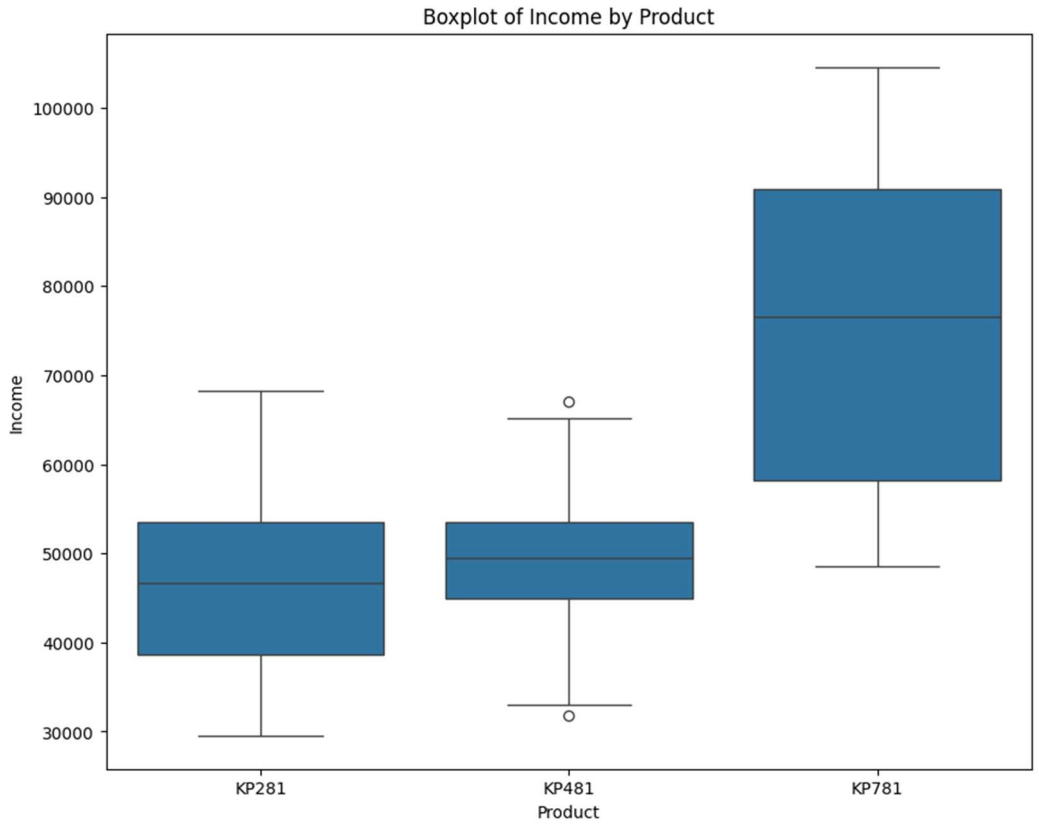


Insights:

- Customers with a fitness rating of 1 tend to use the treadmill for a short duration, with only 1 count each for usage frequencies of 2 and 3 times per week.
- The most common usage frequency is 2 times per week, with 14 counts.
- Usage frequencies of 3 and 4 times per week also have counts, indicating moderate usage among this group.
- The majority prefer to use the treadmill 3 times per week, with a count of 47.
- There are also significant counts for usage frequencies of 4 and 5 times per week, suggesting consistent usage among this group.
- Usage frequencies are more evenly distributed across 3, 4, and 5 times per week, with counts ranging from 6 to 10.
- Some variability in usage patterns is observed within this group.
- Usage frequencies span 3 to 7 times per week.
- The highest count is for usage 3 times per week, followed by counts for 4 and 5 times per week.
- A subset of customers within this group use the treadmill more frequently, with counts for 6 and 7 times per week.

Product by Income:

```
plt.figure(figsize=(10, 8))
sns.boxplot(x='Product', y='Income', data=df)
plt.title('Boxplot of Income by Product')
plt.xlabel('Product')
plt.ylabel('Income')
plt.show()
```



Insights:

For Product KP281:

- There is a wide range of incomes represented, ranging from $29,562 to $68,220.
- The income distribution appears to be relatively even across different income levels, with no clear dominant income range.

For Product KP481:

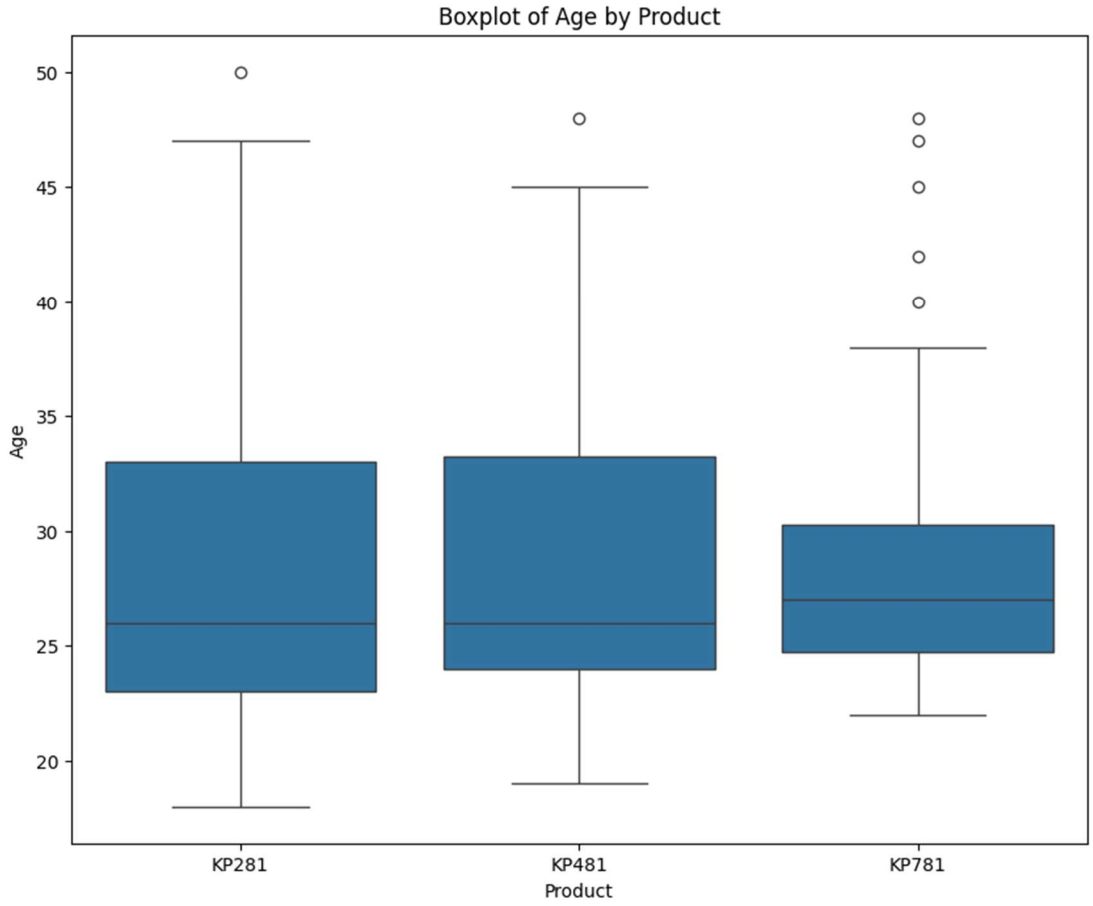
- The income distribution is also diverse, with incomes ranging from $29,562 to $104,581.
- Similar to KP281, there is no clear dominant income range for this product.

For Product KP781:

- The income distribution for this product is also varied, spanning from $47,754 to $104,581.
- Like the other products, there is no apparent concentration of counts in specific income ranges.

Product by Age:

```
plt.figure(figsize=(10, 8))
sns.boxplot(x='Product', y='Age', data=df)
plt.title('Boxplot of Age by Product')
plt.xlabel('Product')
plt.ylabel('Age')
plt.show()
```



Insights:

For Product KP281:

- There is a spread of ages, with the highest count at age 21 and 22 (4 counts each).
- The age distribution seems relatively even across the different age groups.

For Product KP481:

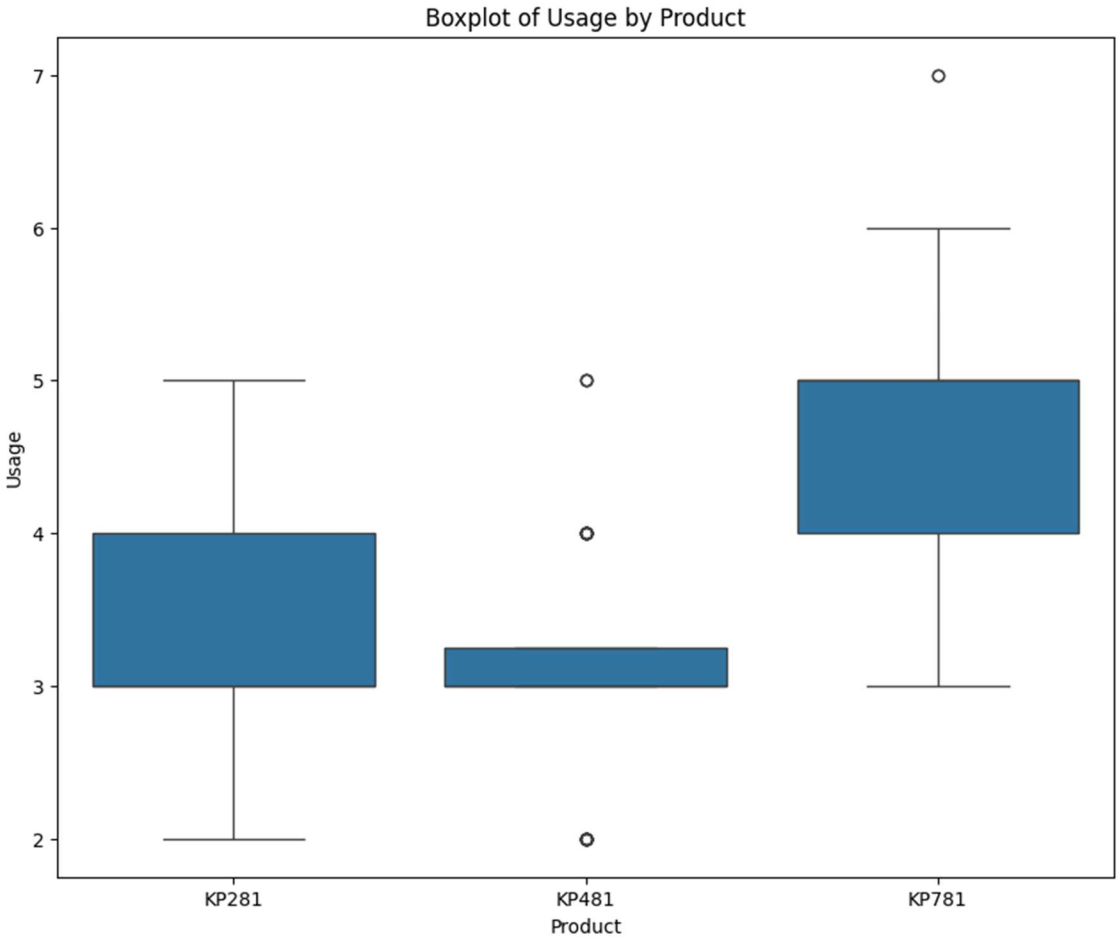
- The age distribution is also varied, with no clear dominant age group.
- The counts are spread across multiple age groups, with no age group having significantly higher counts than others.

For Product KP781:

- The age distribution is relatively uniform, with each age group having only 1 count.
- There is no clear pattern or dominant age group for this product.

Product by Usage:

```
plt.figure(figsize=(10, 8))
sns.boxplot(x='Product', y='Usage', data=df)
plt.title('Boxplot of Usage by Product')
plt.xlabel('Product')
plt.ylabel('Usage')
plt.show()
```



Insights:

For Product KP281:

- Most customers use the treadmill 3 times a week (37 counts), followed by 4 times a week (22 counts).
- Only a few customers use it 2 times a week (19 counts), and very few use it 5 times a week (2 counts).

For Product KP481:

- The usage frequency is more evenly distributed compared to KP281.

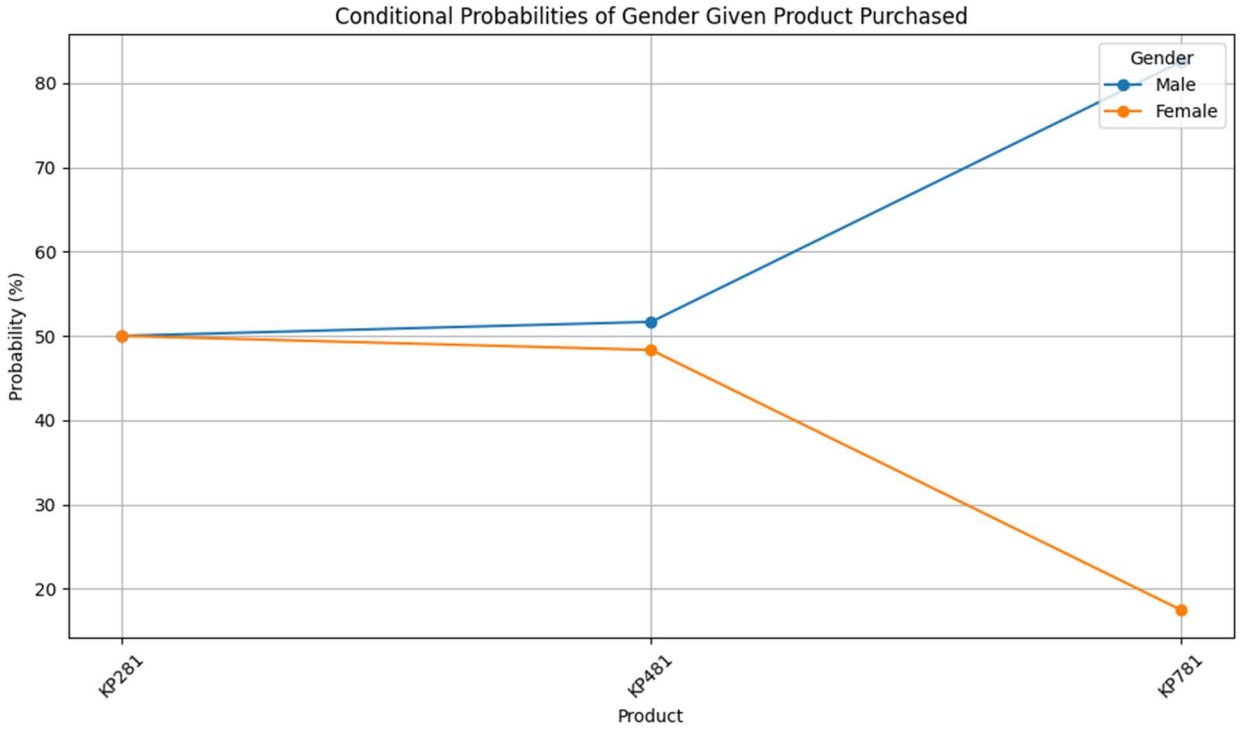
- The majority of customers use it 3 times a week (31 counts), followed by 4 times a week (12 counts).
- Fewer customers use it 2 times a week (14 counts), and very few use it 5 times a week (3 counts).

For Product KP781:

- Usage frequency varies widely for this product.
- The highest usage frequency is 4 times a week (18 counts), followed by 5 times a week (12 counts).
- Some customers also use it 6 times a week (7 counts), and very few use it 3 times a week (1 count) or 7 times a week (2 counts).

Conditional Probability:

```
# Conditional probability: Probability of each gender given product purchased
# Graphical analysis of conditional probabilities

# Set up the plot
plt.figure(figsize=(10, 6))

# Separate plots for each gender
for gender in df['Gender'].unique():
    gender_probs = []
    products = []
    for product in df['Product'].unique():
        prob = len(df[(df['Product'] == product) & (df['Gender'] == gender)]) / len(df[df['Product'] == product]) * 100
        gender_probs.append(prob)
        products.append(product)
    plt.plot(products, gender_probs, marker='o', linestyle='-', label=gender)

# Set plot title and labels
plt.title('Conditional Probabilities of Gender Given Product Purchased')
plt.xlabel('Product')
plt.ylabel('Probability (%)')

# Add legend outside the loop
plt.legend(title='Gender', loc='upper right')

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)

# Show grid
plt.grid(True)

# Show plot
plt.tight_layout()
plt.show()
```



Insights:

KP281:

- The probability of a customer being female or male given the purchase of the KP281 treadmill is equal at 50.0% each.
- This balanced distribution suggests that the KP281 model appeals to both genders equally, indicating a broad market appeal.

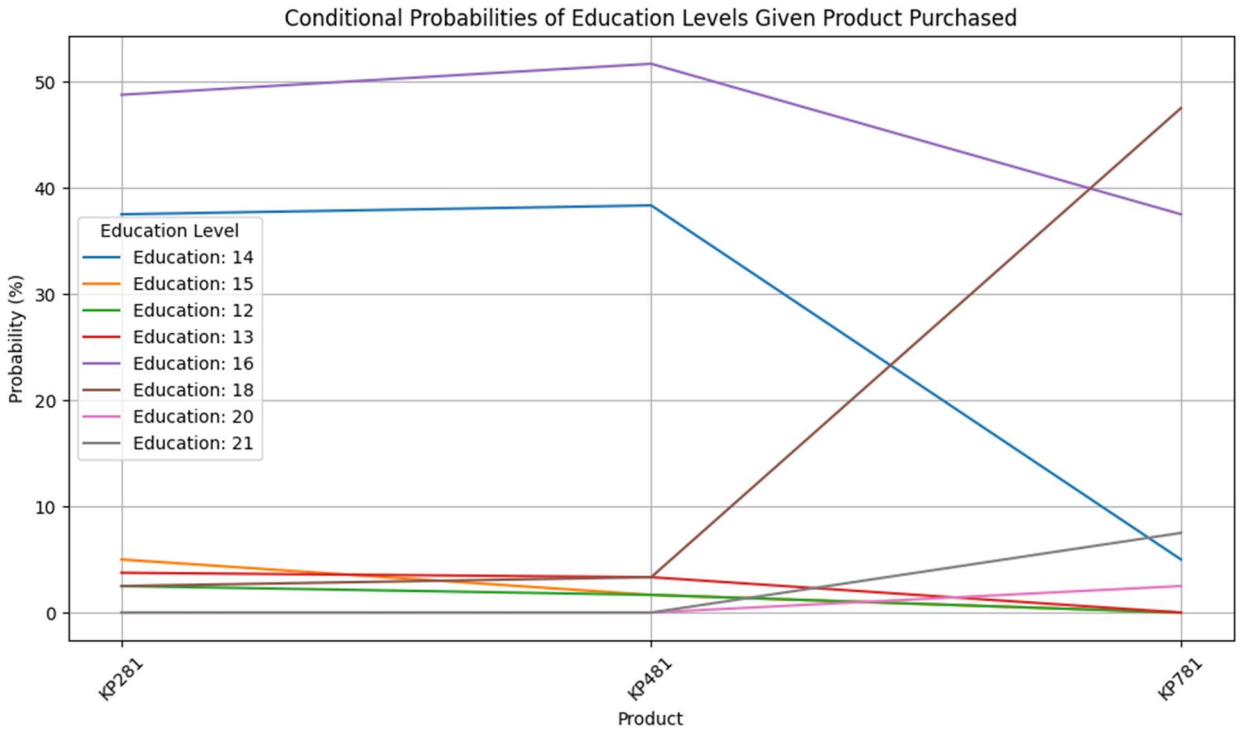
KP481:

- There's a slightly higher probability of a customer being male (51.67%) compared to female (48.33%) when purchasing the KP481 treadmill.
- While still relatively balanced, the KP481 model seems to attract slightly more male customers than female customers.

KP781:

- The probability of a customer being male (82.5%) is significantly higher than that of being female (17.5%) when purchasing the KP781 treadmill.
- This substantial difference suggests that the KP781 model may have features or marketing appeals that resonate more strongly with male customers, indicating potential gender-specific preferences or market segmentation opportunities.

```python
# Graphical analysis of conditional probabilities: Education
plt.figure(figsize=(10, 6))

# Separate plots for each education level
for education_level in df['Education'].unique():
    education_probs = []
    products = []
    for product in df['Product'].unique():
        prob = len(df[(df['Product'] == product) & (df['Education'] == education_level)]) / len(df[df['Product'] == product]) * 100
        education_probs.append(prob)
        products.append(product)
    plt.plot(products, education_probs, label=f'Education: {education_level}')

# Set plot title and labels
plt.title('Conditional Probabilities of Education Levels Given Product Purchased')
plt.xlabel('Product')
plt.ylabel('Probability (%)')

# Add legend
plt.legend(title='Education Level')

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)

# Show grid
plt.grid(True)

# Show plot
plt.tight_layout()
plt.show()
```



Insights:

KP281:

- The highest conditional probability is for customers with an education level of 16, representing 48.75% of purchases for this product.

- Customers with education levels of 14 and 15 also make up significant proportions of purchasers, at 37.50% and 5.00%, respectively.
- This suggests that the KP281 treadmill model is most popular among customers with higher education levels, particularly those with an education level of 16.

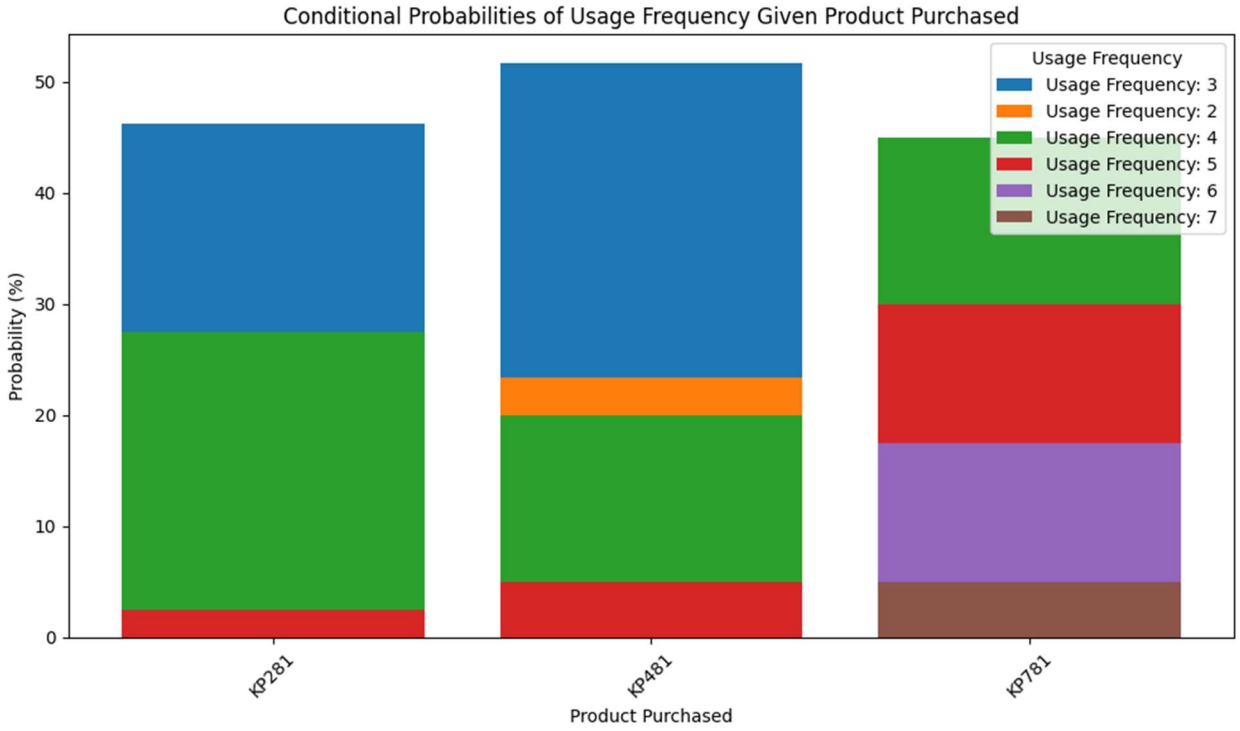
KP481:

- The highest conditional probability is for customers with an education level of 16, representing 51.67% of purchases for this product.
- Similar to KP281, customers with education levels of 14 also make up a significant proportion of purchasers, at 38.33%.
- This indicates that the education level distribution for KP481 is similar to KP281, with a higher concentration among customers with education level 16.

KP781:

- The highest conditional probability is for customers with an education level of 18, representing 47.5% of purchases for this product.
- Customers with education levels of 16 also make up a significant proportion of purchasers, at 37.5%.

```
# Graphical analysis of conditional probabilities: Usage
plt.figure(figsize=(10, 6))

# Separate plots for each usage frequency
for usage_freq in df['Usage'].unique():
    usage_probs = []
    products = []
    for product in df['Product'].unique():
        prob = len(df[(df['Product'] == product) & (df['Usage'] == usage_freq)]) / len(df[df['Product'] == product]) * 100
        usage_probs.append(prob)
        products.append(product)
    plt.bar(products, usage_probs, label=f'Usage Frequency: {usage_freq}')

# Set plot title and labels
plt.title('Conditional Probabilities of Usage Frequency Given Product Purchased')
plt.xlabel('Product Purchased')
plt.ylabel('Probability (%)')

# Add legend
plt.legend(title='Usage Frequency')

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)



# Show plot
plt.tight_layout()
plt.show()
```



Insights:

KP281:

- The highest conditional probability is for customers who plan to use the treadmill 3 times per week, representing 46.25% of purchases for this product.
- Usage frequencies of 2 and 4 times per week also account for significant proportions of purchasers, at 23.75% and 27.50%, respectively.
- This suggests that the KP281 treadmill model is most popular among customers who plan to use it 3 times per week, with a substantial proportion also using it 4 times per week.

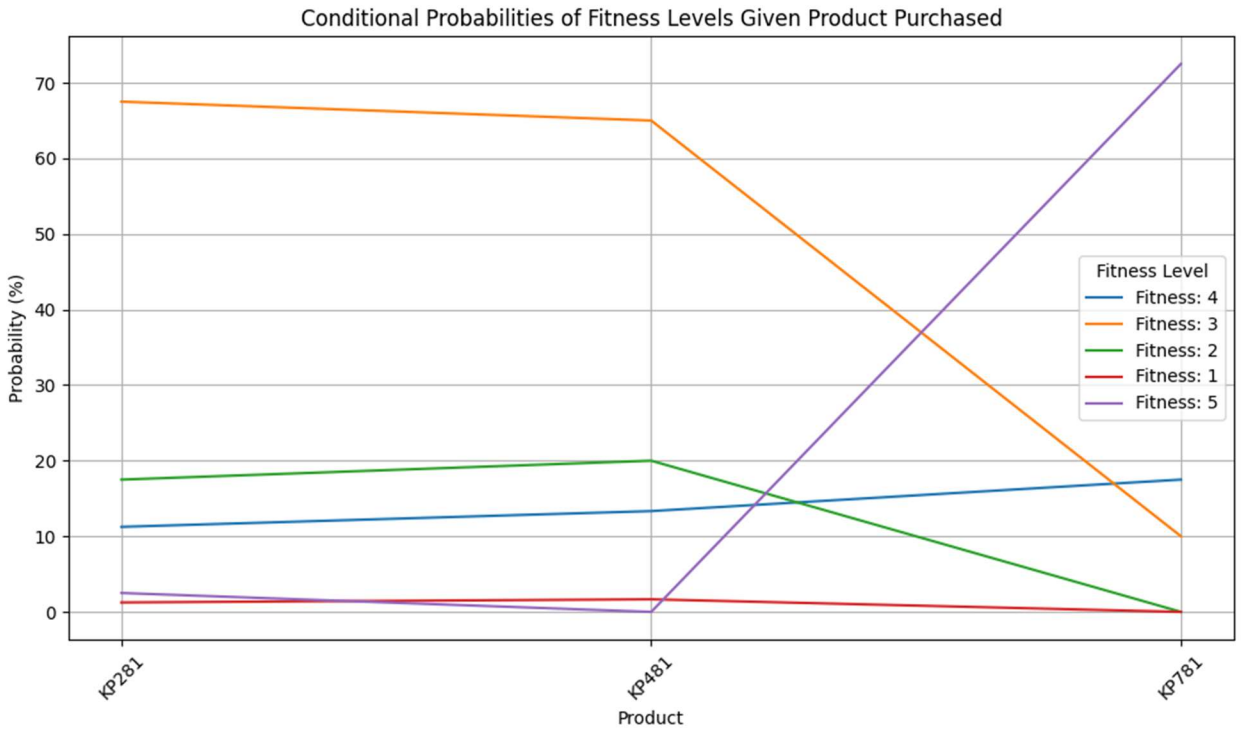
KP481:

- The highest conditional probability is for customers who plan to use the treadmill 3 times per week, representing 51.67% of purchases for this product.
- Usage frequencies of 2 and 4 times per week also account for significant proportions of purchasers, at 23.33% and 20.00%, respectively.
- Similar to KP281, KP481 is most popular among customers who plan to use it 3 times per week, with a substantial proportion also using it 4 times per week.

KP781:

- The highest conditional probability is for customers who plan to use the treadmill 4 times per week, representing 45.0% of purchases for this product.
- Usage frequencies of 5 and 6 times per week also account for significant proportions of purchasers, at 30.0% and 17.5%, respectively.
- This suggests that the KP781 treadmill model is most popular among customers who plan to use it 4 times per week, with a substantial proportion also using it 5 times per week.

```python
# Graphical analysis of conditional probabilities: Fitness
plt.figure(figsize=(10, 6))

# Separate plots for each fitness level
for fitness_level in df['Fitness'].unique():
    fitness_probs = []
    products = []
    for product in df['Product'].unique():
        prob = len(df[(df['Product'] == product) & (df['Fitness'] == fitness_level)]) / len(df[df['Product'] == product]) * 100
        fitness_probs.append(prob)
        products.append(product)
    plt.plot(products, fitness_probs, label=f'Fitness: {fitness_level}')

# Set plot title and labels
plt.title('Conditional Probabilities of Fitness Levels Given Product Purchased')
plt.xlabel('Product')
plt.ylabel('Probability (%)')

# Add legend
plt.legend(title='Fitness Level')

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)

# Show grid
plt.grid(True)

# Show plot
plt.tight_layout()
plt.show()
```



Insights:

KP281:

- The highest conditional probability is for customers with a fitness level of 3, representing 67.50% of purchases for this product.
- Fitness levels 2 and 4 also account for significant proportions of purchasers, at 17.50% and 11.25%, respectively.
- This suggests that the KP281 treadmill model is most popular among customers who rate their fitness level as moderate (level 3), with a smaller proportion of customers rating their fitness as lower (level 2) or higher (level 4).

KP481:

- The highest conditional probability is for customers with a fitness level of 3, representing 65.00% of purchases for this product.
- Fitness levels 2 and 4 also account for significant proportions of purchasers, at 20.00% and 13.33%, respectively.
- Similar to KP281, KP481 is most popular among customers who rate their fitness level as moderate (level 3), with smaller proportions at lower (level 2) and higher (level 4) fitness levels.

KP781:

- The highest conditional probability is for customers with a fitness level of 5, representing 72.5% of purchases for this product.
- Fitness levels 4 and 3 also account for smaller proportions of purchasers, at 17.5% and 10.0%, respectively.
- This indicates that the KP781 treadmill model is primarily popular among customers who rate their fitness level as high (level 5), with fewer customers at lower fitness levels.

Missing Value & Outlier Detection:

```python
# Step 1: Count Missing Values
missing_values = df.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
```

```python
# Step 2: Treat Missing Values
# Let's handle missing values in the 'Income' column by imputing with the median value
median_income = df['Income'].median()
df['Income'].fillna(median_income, inplace=True)
```

```python
# Verify if missing values have been treated
print("\nMissing Values After Treatment:")
print(df.isnull().sum())
```

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

```python
### Outlier Detection and Treatment ###

# Step 1: Detect Outliers (using IQR method)
numeric_columns = df.select_dtypes(include=[np.number])
Q1 = dfs.quantile(0.25, numeric_only=True)
Q3 = dfs.quantile(0.75, numeric_only=True)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Print outliers before treatment
outliers_before = ((dfs.select_dtypes(include=[np.number]) < lower_bound) | (dfs.select_dtypes(include=[np.number]) > upper_bound)).sum()
print("\nOutliers Before Treatment:")
print(outliers_before)
```

```
Outliers Before Treatment:
Age           5
Education     4
Usage         9
Fitness       2
Income       19
Miles        13
dtype: int64
```

```python
# Step 2: Treat Outliers (replace with upper/lower bound)
df_outlier_treated = numeric_columns.apply(lambda x: np.where(x < lower_bound[x.name], lower_bound[x.name], np.where(x > upper_bound[x.name], upper_bound[x.name], x)))

# Combine treated numerical columns with non-numerical columns
df_outlier_treated = pd.concat([dfs.select_dtypes(exclude=[np.number]), df_outlier_treated], axis=1)

# Verify if outliers have been treated
print("\nOutliers After Treatment:")
outliers_after = ((df_outlier_treated.select_dtypes(include=[np.number]) < lower_bound) | (df_outlier_treated.select_dtypes(include=[np.number]) > upper_bound)).sum()
print(outliers_after)
```

```
Outliers After Treatment:
Age          0
Education    0
Usage        0
Fitness      0
Income       0
Miles        0
dtype: int64
```

Recommendations:

Customized Product Offerings:

- Develop tailored treadmill packages for different customer segments based on marital status, ensuring options cater to both partnered and single individuals' preferences and needs.

Targeted Marketing Campaigns:

- Create targeted marketing messages and advertisements considering gender preferences. Tailor promotions to appeal to both male and female customers, highlighting features that resonate with their specific interests.

Segmented Product Development:

- Design treadmill models with features suited to different age groups identified through customer data analysis. Consider age-specific preferences and requirements to ensure product offerings align with varying customer demographics.

Education-Oriented Sales Approach:

- Offer educational resources and support to customers based on their education levels. Develop materials tailored to different educational backgrounds to ensure all customers feel empowered and informed when making purchasing decisions.

Usage-Focused Product Design:

- Incorporate usage frequency insights into product design and functionality. Develop treadmills with features that cater to different usage patterns, such as offering preset workout programs for various weekly usage frequencies.

Income-Based Pricing Strategies:

- Implement pricing strategies that accommodate varying income levels. Offer flexible pricing options and financing plans to make products accessible to customers across different income brackets.

Fitness-Centric Marketing Messages:

- Craft marketing campaigns that emphasize the fitness benefits of AeroFit treadmills, tailored to different fitness levels. Highlight how each treadmill model can help customers achieve their fitness goals, regardless of their current fitness level.

Mileage-Specific Product Features:

- Develop treadmill features designed to support customers' expected mileage goals. Incorporate features such as adjustable incline levels or cushioning systems to enhance comfort and performance during long-distance runs or walks.