In [7]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

Step 1: Import the Dataset and Initial Analysis

First, let's import the dataset and take a look at its structure and characteristics.

In [26]:
```
# Load the dataset
data = pd.read_csv('Aerofit_treadmill.csv')

# Display the 5 records
print(data.sample(5))

print("\n")

# Check the structure and data types
print(data.info())

print("\n")

# Display basic statistics
print(data.describe())

print("\n")

# Additional analysis for categorical values
categorical_columns = data.select_dtypes(include=['object', 'category']).

for column in categorical_columns:
    print(f"Value counts for {column}:\n", data[column].value_counts())
    print(f"Unique values for {column}:\n", data[column].unique())
    print("\n")
```

```
    Product  Age  Gender  Education MaritalStatus  Usage  Fitness  Income
\
108   KP481   26  Female         16     Partnered      4        3   45480
45    KP281   28  Female         16     Partnered      2        3   52302
29    KP281   25  Female         14     Partnered      2        2   53439
42    KP281   27    Male         16        Single      4        3   54576
66    KP281   36    Male         12        Single      4        3   44343

     Miles
108     85
45      66
29      47
42      85
66      94


<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
None


              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  


Value counts for Product:
 Product
KP281    80
KP481    60
KP781    40
Name: count, dtype: int64
Unique values for Product:
 ['KP281' 'KP481' 'KP781']


Value counts for Gender:
 Gender
Male      104
Female     76
Name: count, dtype: int64
Unique values for Gender:
 ['Male' 'Female']
```

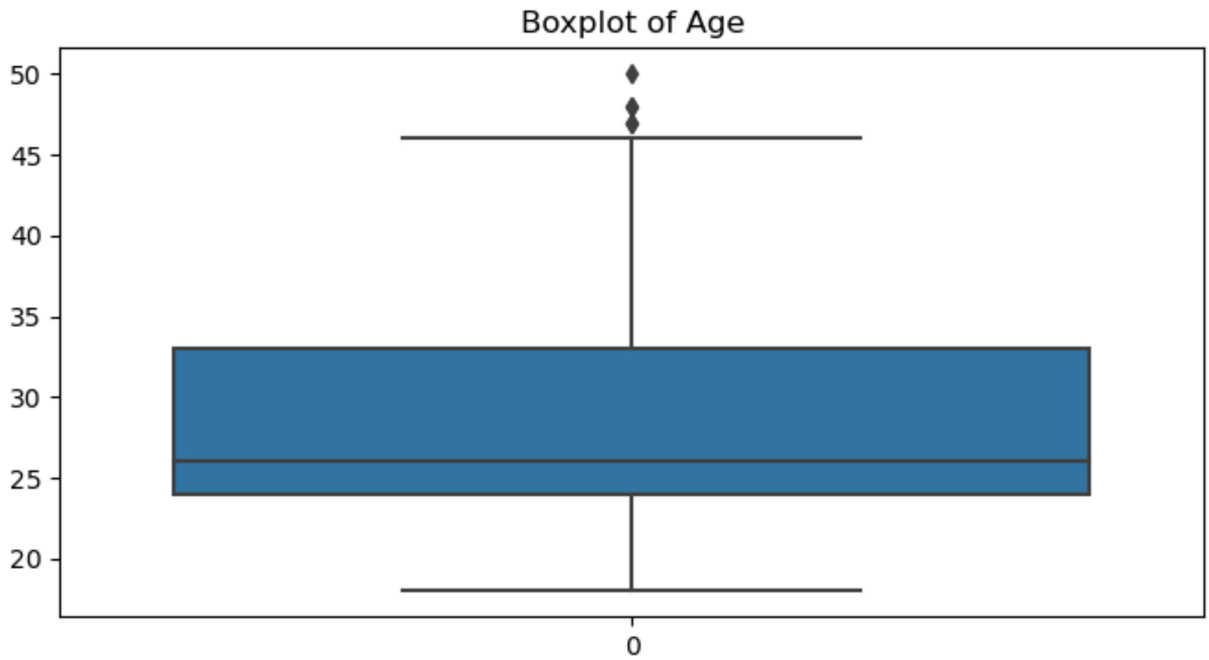
```
Value counts for MaritalStatus:
 MaritalStatus
Partnered    107
Single        73
Name: count, dtype: int64
Unique values for MaritalStatus:
 ['Single' 'Partnered']
```

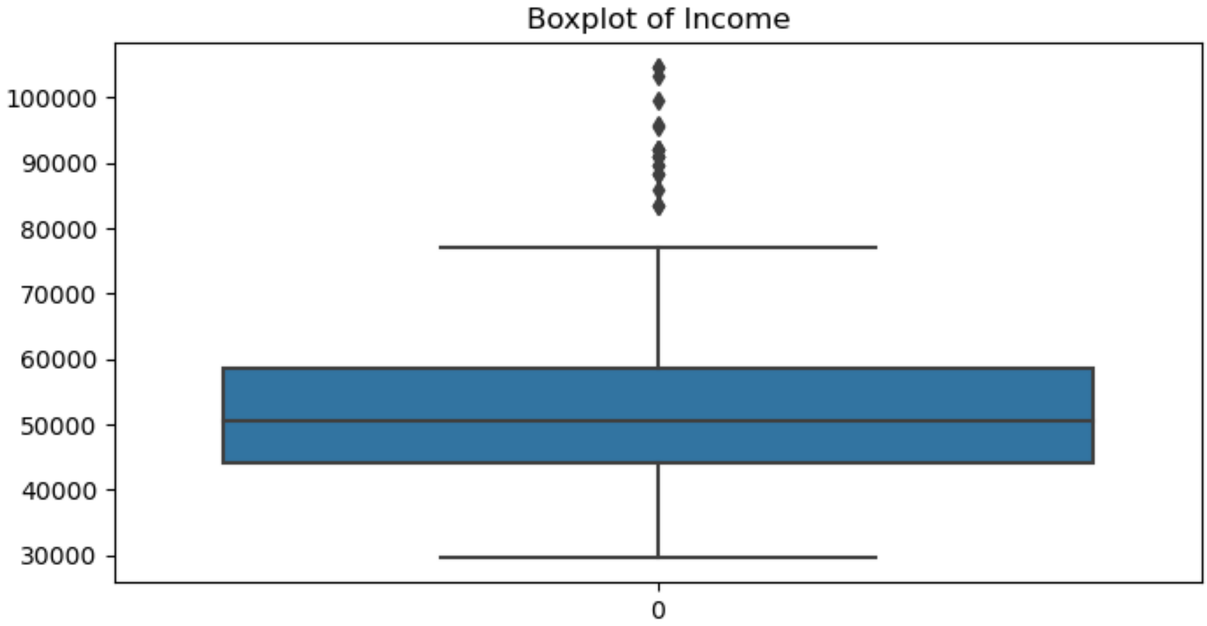
There is no missing data

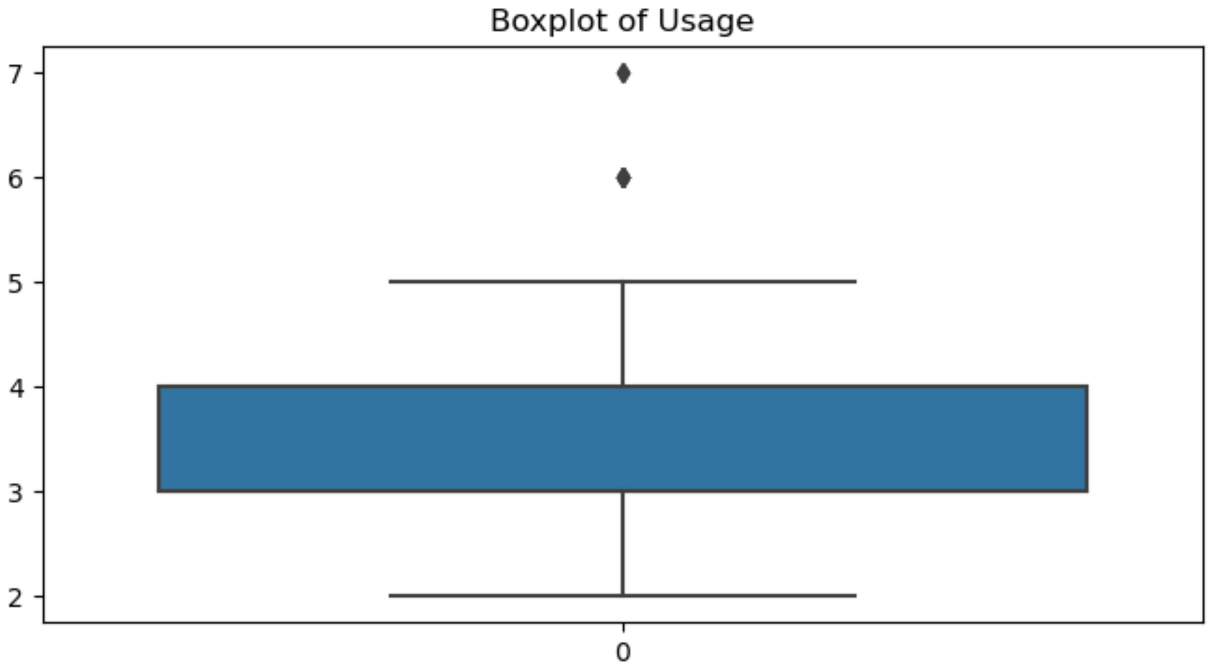
Step 2: Detect Outliers

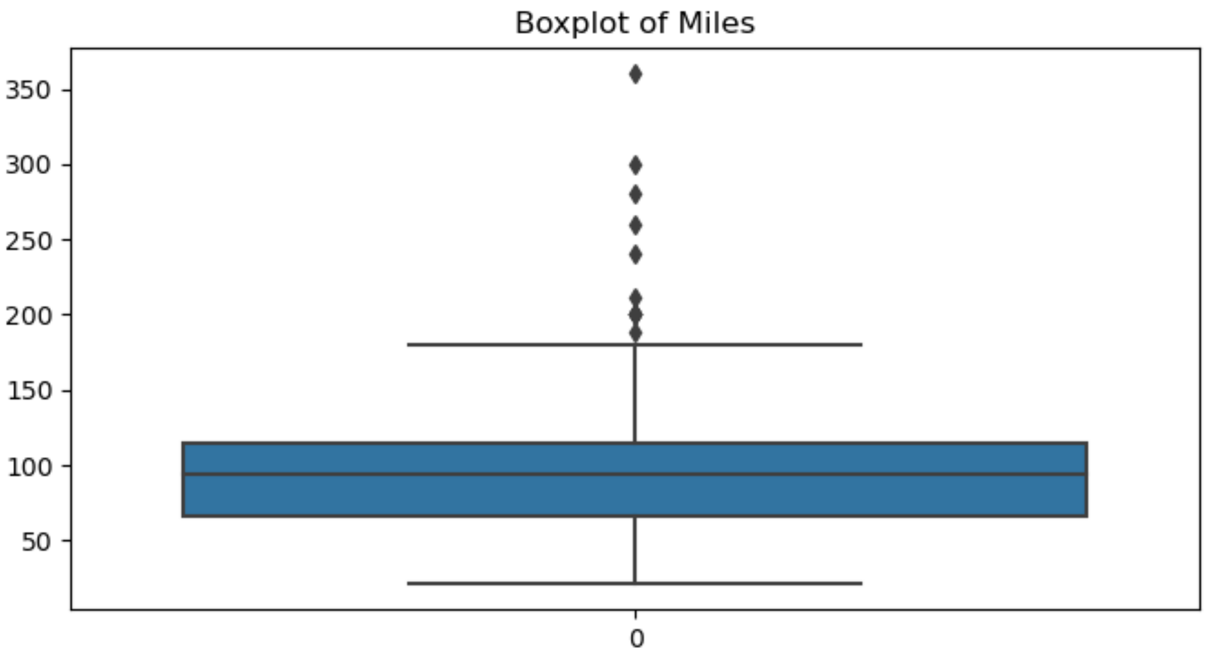
We'll use boxplots and the describe method to detect outliers, focusing on continuous variables like Age, Income, Usage, and Miles.

In [8]:
```
# Boxplots to detect outliers
for column in ['Age', 'Income', 'Usage', 'Miles']:
    plt.figure(figsize=(8, 4))
    sns.boxplot(data[column])
    plt.title(f'Boxplot of {column}')
    plt.show()

# Checking the difference between mean and median
for column in ['Age', 'Income', 'Usage', 'Miles']:
    print(f"{column}: Mean = {data[column].mean()}, Median = {data[column
```




Boxplot of Income
100000
90000
80000
70000
60000
50000
40000
30000
0


Boxplot of Usage
7
6
5
4
3
2
0




```
Age: Mean = 28.788888888888888, Median = 26.0
Income: Mean = 53719.57777777778, Median = 50596.5
Usage: Mean = 3.4555555555555557, Median = 3.0
Miles: Mean = 103.19444444444444, Median = 94.0
```

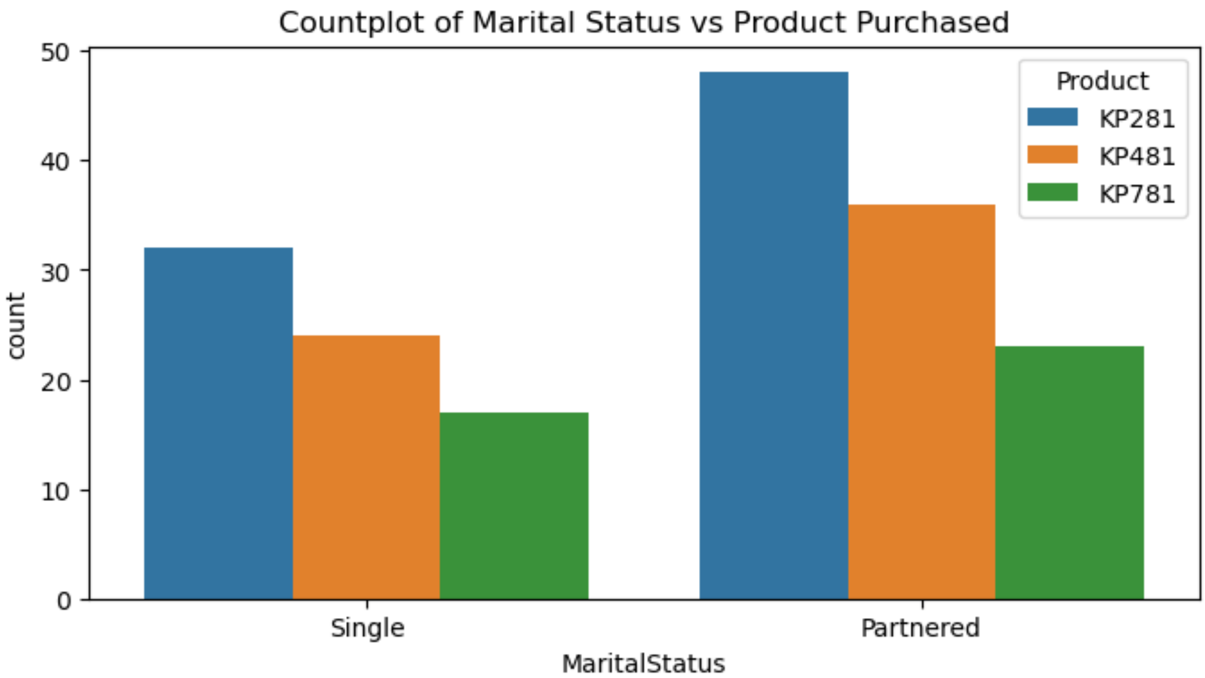
All the numerical columns have outliers in them, with maximum mean affected in usage because of outliers

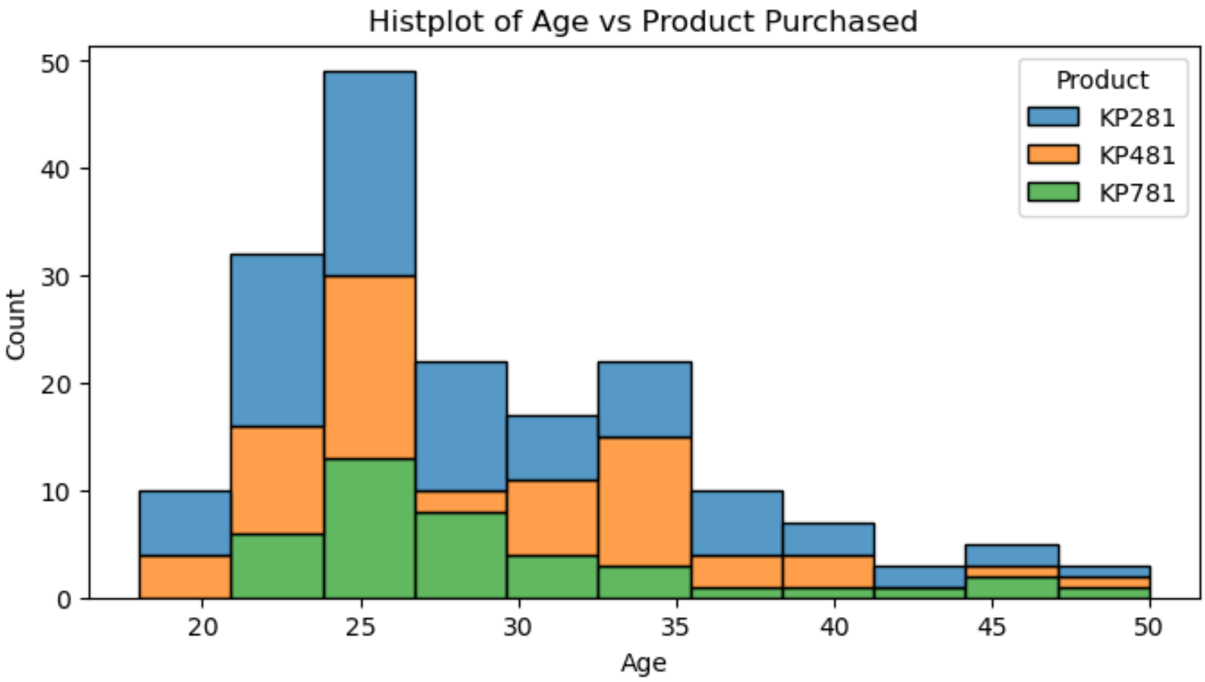
Step 3: Analyzing the Effect of Features on Product Purchased

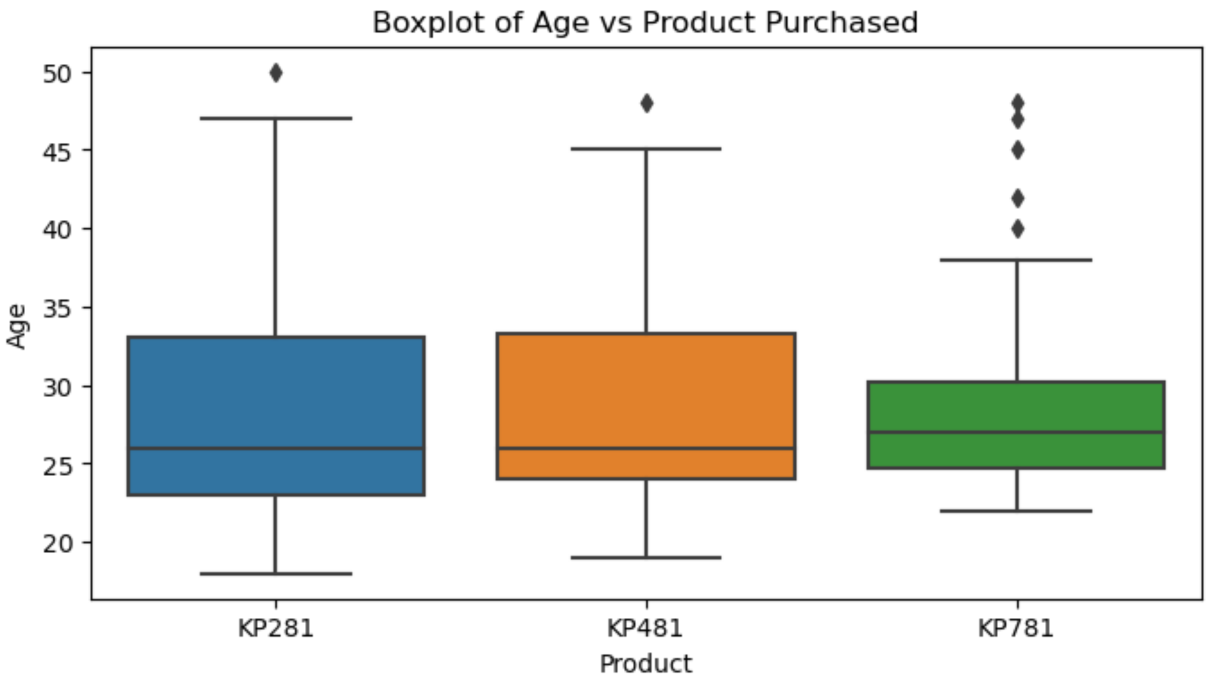
We'll examine if features like marital status and age have any effect on the product purchased using countplots, histplots, and boxplots.

In [9]:

```
# Countplot for Marital Status vs Product Purchased
plt.figure(figsize=(8, 4))
sns.countplot(x='MaritalStatus', hue='Product', data=data)
plt.title('Countplot of Marital Status vs Product Purchased')
plt.show()

# Histplot for Age vs Product Purchased
plt.figure(figsize=(8, 4))
sns.histplot(data=data, x='Age', hue='Product', multiple='stack')
plt.title('Histplot of Age vs Product Purchased')
plt.show()

# Boxplot for Age vs Product Purchased
plt.figure(figsize=(8, 4))
sns.boxplot(x='Product', y='Age', data=data)
plt.title('Boxplot of Age vs Product Purchased')
plt.show()
```


Countplot of Marital Status vs Product Purchased
Product
KP281
KP481
KP781
count
0
10
20
30
40
50
Single
Partnered
MaritalStatus


Histplot of Age vs Product Purchased
Product
KP281
KP481
KP781
Count
0
10
20
30
40
50
20
25
30
35
40
45
50
Age



KP281 is bought by both lower age and upper age limit people, with KP781 being bought by limited number of people between age 23 and 38 with maximum outliers.

Partnered couples have bought all 3 devices more than single people.

Count of products by Age is right skewed with most products bought by customers of age 25.

Step 4: Marginal Probability of Product Purchased

We'll calculate the marginal probability of customers purchasing each product using a crosstab.

In [10]:
```
# Crosstab for Product Purchased
product_counts = pd.crosstab(index=data['Product'], columns='count')
product_prob = product_counts / product_counts.sum()

print("Marginal Probability of Product Purchased:\n", product_prob)
```

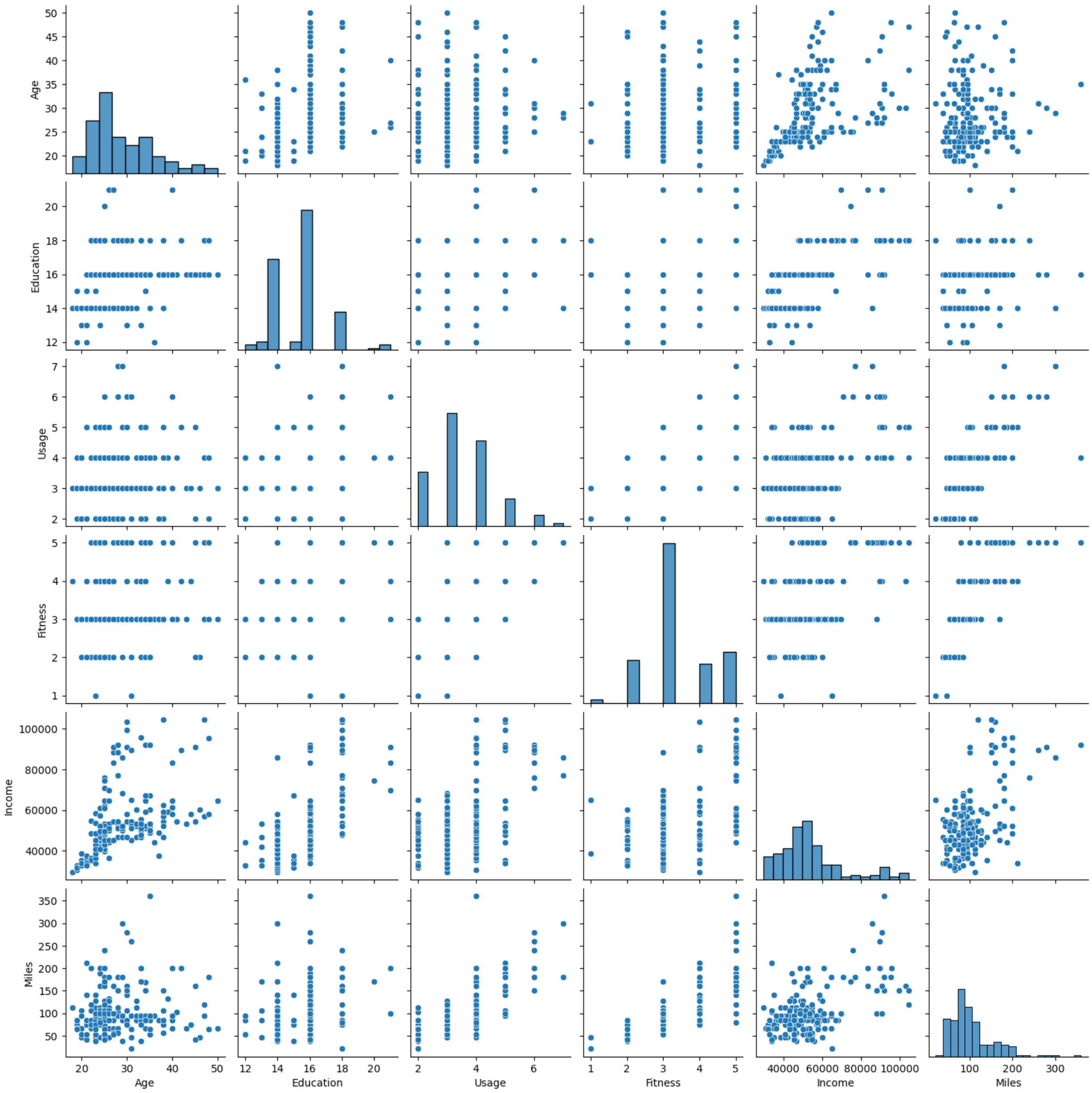
```
Marginal Probability of Product Purchased:
 col_0       count
Product
KP281    0.444444
KP481    0.333333
KP781    0.222222
```

Step 5: Correlation Analysis

We'll examine the correlation among different factors using a heatmap and pairplot.

In [15]:
```
# # Heatmap for Correlation
# plt.figure(figsize=(10, 8))
# sns.heatmap(data.corr(), annot=True, cmap='coolwarm')
# plt.title('Correlation Heatmap')
# plt.show()

# # Pairplot for visualizing relationships
# sns.pairplot(data)
# plt.show()

# Select only numeric columns for correlation analysis
numeric_data = data.select_dtypes(include=['float64', 'int64'])

# Heatmap for Correlation
plt.figure(figsize=(10, 8))
sns.heatmap(numeric_data.corr(), annot=True)
plt.title('Correlation Heatmap')
plt.show()

# Pairplot for visualizing relationships
sns.pairplot(numeric_data)
plt.show()
```


```
/Users/himkantnigam/anaconda3/lib/python3.11/site-packages/seaborn/axisgr
id.py:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



Step 6: Probability Analysis

We'll calculate the probability of specific events, such as the probability of a male customer buying a KP781 treadmill.

In [17]:

```
# Conditional probability: Probability of a male customer buying a KP781
male_kp781 = data[(data['Gender'] == 'Male') & (data['Product'] == 'KP781
total_males = data[data['Gender'] == 'Male'].shape[0]
prob_male_kp781 = male_kp781 / total_males

print(f"Probability of a male customer buying a KP781 treadmill: {prob_ma

# Conditional probability: Probability of a male customer buying a KP781
male_kp281 = data[(data['Gender'] == 'Male') & (data['Product'] == 'KP281
total_males = data[data['Gender'] == 'Male'].shape[0]
prob_male_kp281 = male_kp281 / total_males

print(f"Probability of a male customer buying a KP281 treadmill: {prob_ma

# Conditional probability: Probability of a male customer buying a KP781
male_kp481 = data[(data['Gender'] == 'Male') & (data['Product'] == 'KP481
total_males = data[data['Gender'] == 'Male'].shape[0]
prob_male_kp481 = male_kp481 / total_males

print(f"Probability of a male customer buying a KP481 treadmill: {prob_ma
```

```
Probability of a male customer buying a KP781 treadmill: 0.32
Probability of a male customer buying a KP281 treadmill: 0.38
Probability of a male customer buying a KP481 treadmill: 0.30
```

In [19]:

```
# Conditional probability: Probability of a male customer buying a KP781
male_kp781 = data[(data['Gender'] == 'Female') & (data['Product'] == 'KP7
total_males = data[data['Gender'] == 'Female'].shape[0]
prob_male_kp781 = male_kp781 / total_males

print(f"Probability of a Female customer buying a KP781 treadmill: {prob_

# Conditional probability: Probability of a male customer buying a KP781
male_kp281 = data[(data['Gender'] == 'Female') & (data['Product'] == 'KP2
total_males = data[data['Gender'] == 'Female'].shape[0]
prob_male_kp281 = male_kp281 / total_males

print(f"Probability of a Female customer buying a KP281 treadmill: {prob_

# Conditional probability: Probability of a male customer buying a KP781
male_kp481 = data[(data['Gender'] == 'Female') & (data['Product'] == 'KP4
total_males = data[data['Gender'] == 'Female'].shape[0]
prob_male_kp481 = male_kp481 / total_males

print(f"Probability of a Female customer buying a KP481 treadmill: {prob_
```

```
Probability of a Female customer buying a KP781 treadmill: 0.09
Probability of a Female customer buying a KP281 treadmill: 0.53
Probability of a Female customer buying a KP481 treadmill: 0.38
```

Males are High-Fitness Customers over Females since females prefer more of treadmill KP281 and it is very less likely that females opt for top treadmill KP781 which is a advanced product.

Step 7: Customer Profiling

We'll categorize users based on their purchasing behavior and demographic information.

In [13]:
```
# Example categorization based on income and fitness level
def categorize_income(income):
    if income < 30000:
        return 'Low'
    elif 30000 <= income < 60000:
        return 'Medium'
    else:
        return 'High'

data['IncomeCategory'] = data['Income'].apply(categorize_income)

# Profiling
profile_summary = data.groupby(['Product', 'IncomeCategory', 'Fitness']).
print("Customer Profiling Summary:\n", profile_summary)
```

```
Customer Profiling Summary:
 Fitness                  1   2   3  4   5
Product IncomeCategory
KP281   High             0   1   5  0   0
        Low              0   0   0  1   0
        Medium           1  13  49  8   2
KP481   High             1   0   6  0   0
        Medium           0  12  33  8   0
KP781   High             0   0   3  6  20
        Medium           0   0   1  1   9
```

Step 8: Recommendations and Actionable Insights

Recommendations:

1. **Target High-Income and High-Fitness Customers**: Marketing efforts should focus on high-income individuals with higher fitness levels, especially for the KP781 model.
2. **Custom Marketing for Different Age Groups**: Tailor marketing campaigns based on age groups. For example, older customers (above 45) may prefer products with lower intensity.
3. **Promote Usage Benefits**: Highlight the benefits of regular treadmill usage to increase the usage frequency among current users.
4. **Improve Product Features Awareness**: Educate customers on the advanced features of the KP781 to justify its higher price and enhance perceived value.
5. **Address Outliers in Income Data**: Investigate and validate any anomalies in income data to ensure accurate profiling and targeted marketing.

By implementing these strategies, AeroFit can better align its marketing efforts with customer preferences and purchasing behaviors, ultimately driving higher sales and customer satisfaction.