Importing necessary Libraties

In [6]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

Structure of Data

In [7]:
```
!gdown https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/125/original
```

```
Downloading...
From: https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/125/origina
l/aerofit_treadmill.csv?1639992749
To: /content/aerofit_treadmill.csv?1639992749
100% 7.28k/7.28k [00:00<00:00, 26.1MB/s]
```

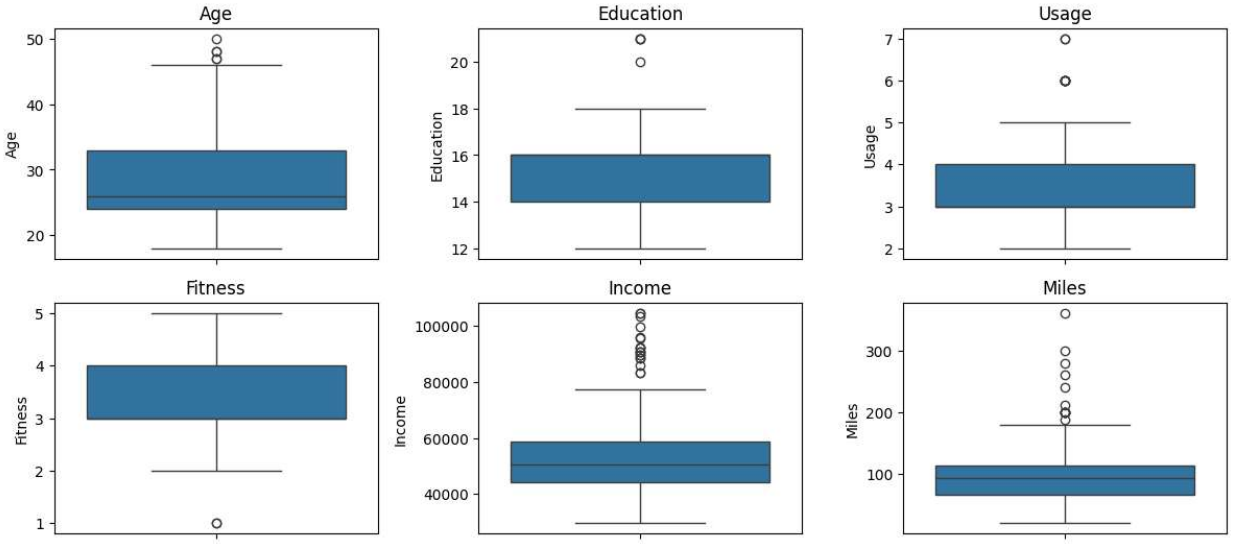
In [19]:
```
# Load the dataset
data = pd.read_csv("aerofit_treadmill.csv?1639992749")

# Display data type of each column
print("Data types of each column:")
print(data.dtypes)

# Find the number of rows & columns
num_rows, num_columns = data.shape
print("\nNumber of rows:", num_rows)
print("Number of columns:", num_columns)

# Get the shape of dataset
data_shape = data.shape
print("\nShape of the dataset:", data_shape)

# Check for missing values
missing_values = data.isnull().sum()
print("\nNumber of missing values in each column:")
print(missing_values)
```

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

Number of rows: 180
Number of columns: 9

Shape of the dataset: (180, 9)

Number of missing values in each column:
Product          0
Age              0
Gender           0
Education        0
MaritalStatus    0
Usage            0
Fitness          0
Income           0
Miles            0
dtype: int64
```

The dataset contains 180 rows and 9 columns, with no missing values. It includes information on product usage, demographic variables, and consumer characteristics, with numerical and categorical data types represented.

Detect Outliers

In [9]:
```python
# Select only continuous variables
continuous_variables = data.select_dtypes(include=['int64', 'float64'])

# Plot boxplots for each continuous variable to find outliers
plt.figure(figsize=(12, 8))
for column in continuous_variables.columns:
    plt.subplot(3, 3, list(continuous_variables.columns).index(column) + 1)
    sns.boxplot(y=column, data=data)
    plt.title(column)
plt.tight_layout()
plt.show()
```



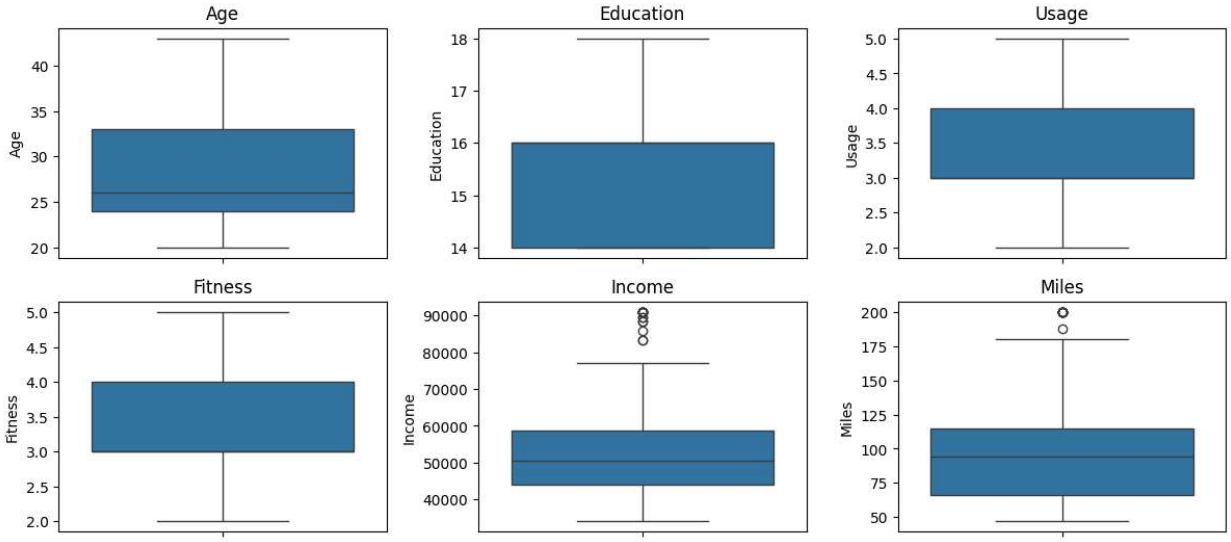
Here we are using Box plot to detect outliers for all continuous variables

In [10]:
```
# Select only continuous variables
continuous_variables = data.select_dtypes(include=['int64', 'float64'])

# Clip the data between 5th and 95th percentile
clipped_data = continuous_variables.apply(lambda x: np.clip(x, x.quantile(0.05), x.qua

# Replace original continuous variables with clipped data
data[continuous_variables.columns] = clipped_data

# Display clipped data
print("Clipped Data:")
print(data)
```

```
Clipped Data:
    Product    Age  Gender  Education MaritalStatus  Usage  Fitness    Income  \
0     KP281  20.00    Male         14        Single   3.00        4  34053.15
1     KP281  20.00    Male         15        Single   2.00        3  34053.15
2     KP281  20.00  Female         14     Partnered   4.00        3  34053.15
3     KP281  20.00    Male         14        Single   3.00        3  34053.15
4     KP281  20.00    Male         14     Partnered   4.00        2  35247.00
..      ...    ...     ...        ...           ...    ...      ...       ...
175   KP781  40.00    Male         18        Single   5.05        5  83416.00
176   KP781  42.00    Male         18        Single   5.00        4  89641.00
177   KP781  43.05    Male         16        Single   5.00        5  90886.00
178   KP781  43.05    Male         18     Partnered   4.00        5  90948.25
179   KP781  43.05    Male         18     Partnered   4.00        5  90948.25

     Miles
0      112
1       75
2       66
3       85
4       47
..     ...
175    200
176    200
177    160
178    120
179    180

[180 rows x 9 columns]
```

The data has been clipped by removing extreme values beyond the 5th and 95th percentiles for continuous variables. This process helps create a more robust dataset by reducing the impact of outliers while preserving valuable information.

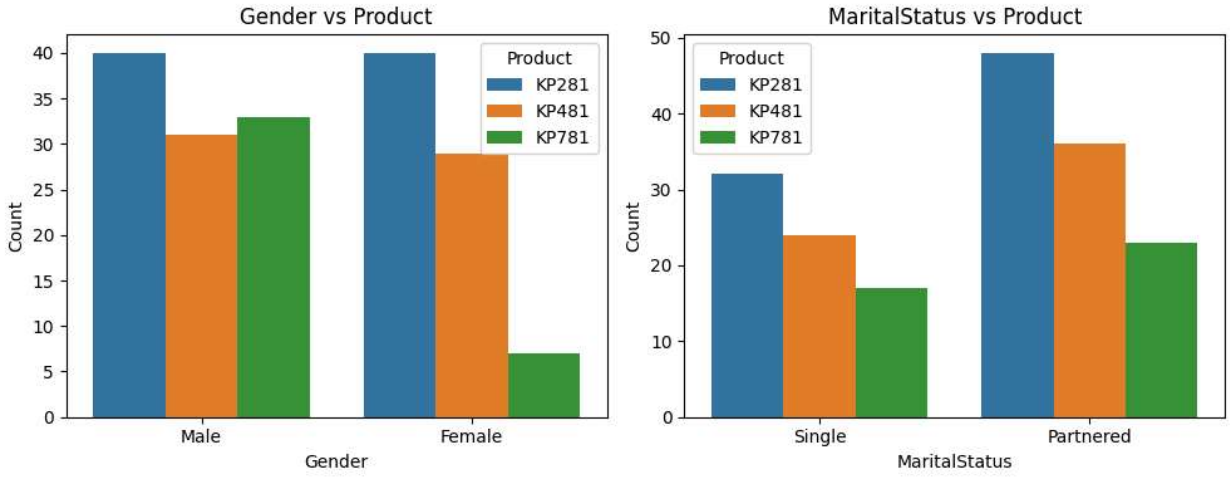
In [11]:
```
# Select only continuous variables
continuous_variables = data.select_dtypes(include=['int64', 'float64'])

# Clip data between 5th and 95th percentiles for all variables
clipped_data = continuous_variables.clip(continuous_variables.quantile(0.05), continuc

# Plot boxplots for each continuous variable to visualize the clipped data
plt.figure(figsize=(12, 8))
for i, column in enumerate(clipped_data.columns, 1):
    plt.subplot(3, 3, i)
    sns.boxplot(y=column, data=clipped_data)
    plt.title(column)
plt.tight_layout()
plt.show()
```



Visualizing clipped data of all continuous variables

Relation between Categorical Variables and Product

In [12]:
```
# Select categorical variables & the output variable(product)
categorical_variables = ['Gender', 'MaritalStatus']
output_variable = 'Product'

# Plot count plots for each categorical variable against the products
plt.figure(figsize=(10, 4))
for i, cat_var in enumerate(categorical_variables, 1):
    plt.subplot(1, len(categorical_variables), i)
    sns.countplot(x=cat_var, hue=output_variable, data=data)
    plt.title(f'{cat_var} vs {output_variable}')
    plt.xlabel(cat_var)
    plt.ylabel('Count')
    plt.legend(title=output_variable)
plt.tight_layout()
plt.show()
```



KP281:

Most Popular: KP281 is the most popular product across all categories.

Gender: Males show a higher preference for KP281 compared to females.

Marital Status: Partnered individuals (married or in a relationship) also favor KP281.

KP481:

Balanced Interest: Females and singles (unmarried) show a more balanced interest in KP481.

Gender: Males have a lower preference for KP481.

Marital Status: Singles show engagement with both KP281 and KP481.

KP781:

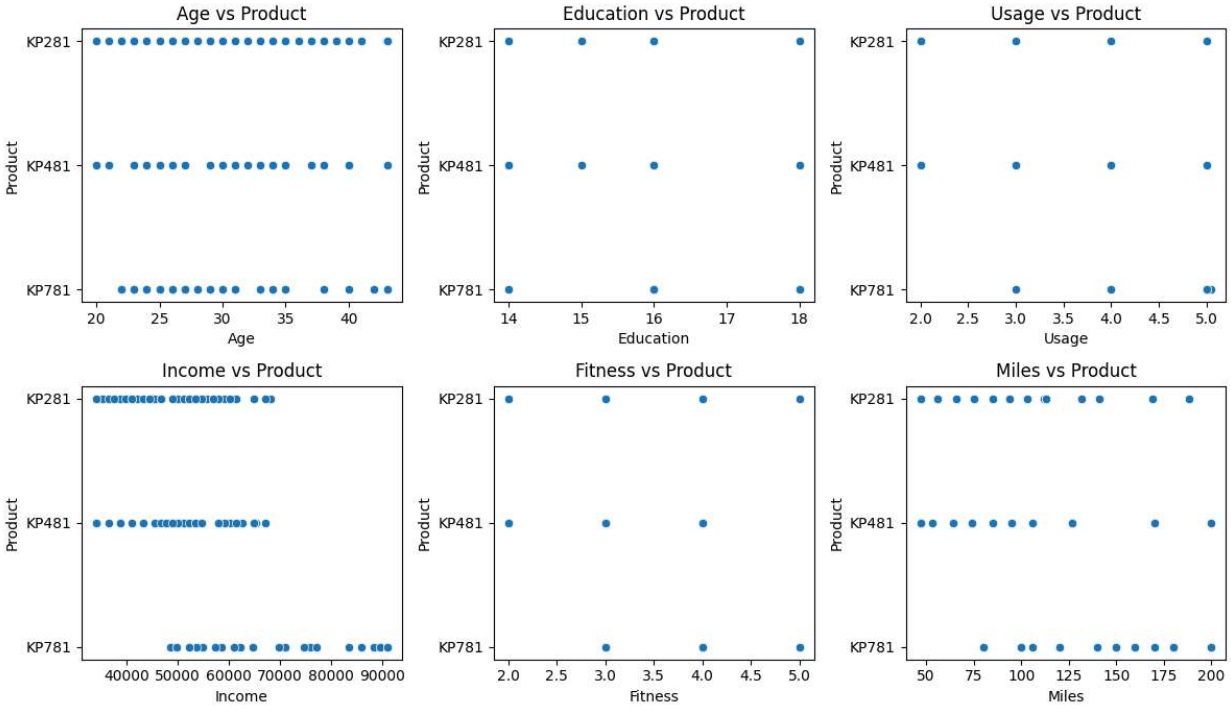
Least Interest: KP781 has the least interest across all categories.

Gender: Females show minimal engagement with KP781.

Marital Status: Both partnered and single individuals have limited interest in KP781.

Relation between Continuous Variables and Product

In [13]:
```
# Select continuous variables and output variable
continuous_variables = ['Age', 'Education', 'Usage', 'Income', 'Fitness', 'Miles']
output_variable = 'Product'

# Plot scatter plots for each continuous variable against the product
plt.figure(figsize=(12, 7))
for i, cont_var in enumerate(continuous_variables, 1):
    plt.subplot(2, 3, i)
    sns.scatterplot(x=cont_var, y=output_variable, data=data)
    plt.title(f'{cont_var} vs {output_variable}')
    plt.xlabel(cont_var)
    plt.ylabel(output_variable)
plt.tight_layout()
plt.show()
```

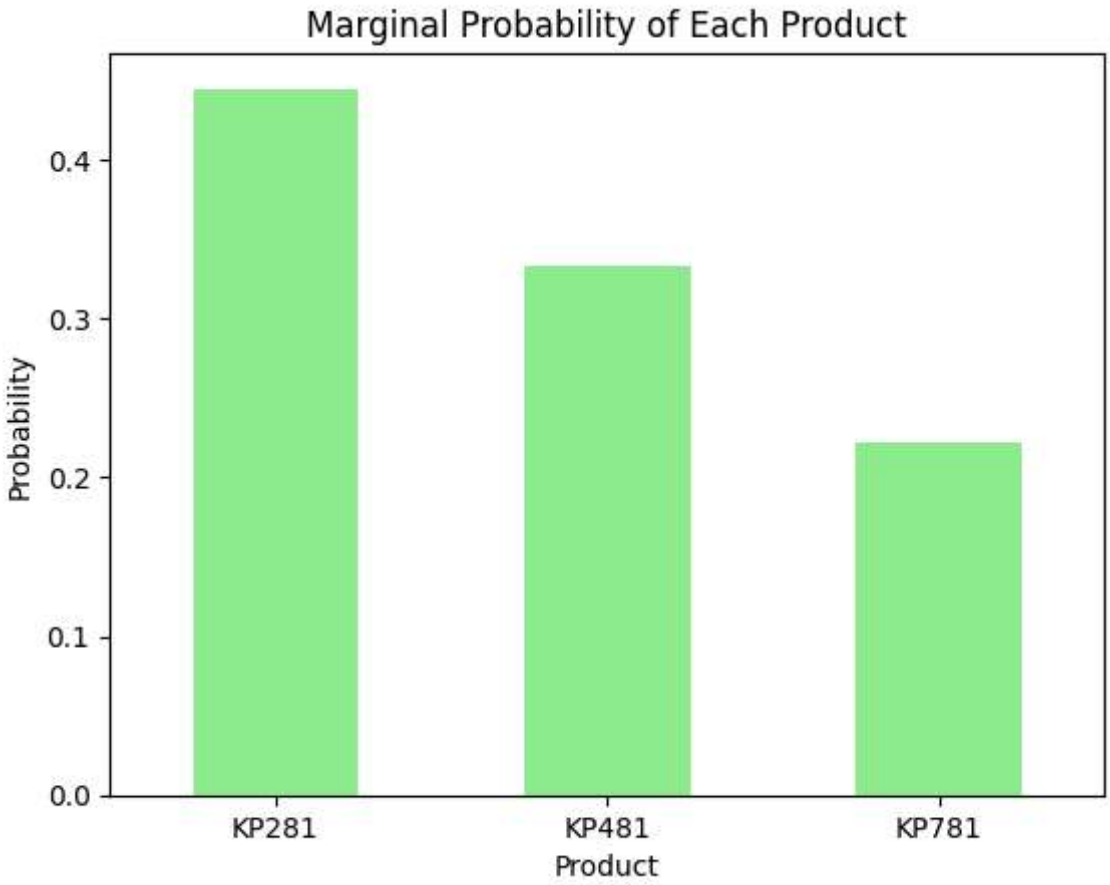


Representing the Probabilities

Marginal Probabilities

In [14]:
```
# Create a crosstab to compute marginal probabilities
marginal_prob = pd.crosstab(index=data['Product'], columns='count', normalize=True)

# Plotting marginal probability
plt.figure(figsize=(6, 4))
marginal_prob.plot(kind='bar', legend=False, color='lightgreen')
plt.title('Marginal Probability of Each Product')
plt.xlabel('Product')
plt.ylabel('Probability')
plt.xticks(rotation=0)
plt.show()
```

```
<Figure size 600x400 with 0 Axes>
```



KP281:

It has the highest probability of being selected or preferred.

KP481:

It has a moderate probability of being selected.

KP781:

It has the lowest probability of being selected.

This suggests that KP281 is the most likely choice among the three products & KP781 is the least preferred among the three products.

Conditional Probabilities

In [15]:
```
# Calculate conditional probabilities for each column
conditional_probs = {}
for column in data.columns:
    if column != 'Product':
        conditional_probs[column] = pd.crosstab(index=data[column], columns=data['Proc

# Display conditional probabilities
for column, prob in conditional_probs.items():
```

```
    print(f"\nProbability of buying a product based on {column}:")
    print(prob)
```

Probability of buying a product based on Age:

Product	KP281	KP481	KP781
Age			
20.00	0.600000	0.400000	0.000000
21.00	0.571429	0.428571	0.000000
22.00	0.571429	0.000000	0.428571
23.00	0.444444	0.388889	0.166667
24.00	0.416667	0.250000	0.333333
25.00	0.280000	0.440000	0.280000
26.00	0.583333	0.250000	0.166667
27.00	0.428571	0.142857	0.428571
28.00	0.666667	0.000000	0.333333
29.00	0.500000	0.166667	0.333333
30.00	0.285714	0.285714	0.428571
31.00	0.333333	0.500000	0.166667
32.00	0.500000	0.500000	0.000000
33.00	0.250000	0.625000	0.125000
34.00	0.333333	0.500000	0.166667
35.00	0.375000	0.500000	0.125000
36.00	1.000000	0.000000	0.000000
37.00	0.500000	0.500000	0.000000
38.00	0.571429	0.285714	0.142857
39.00	1.000000	0.000000	0.000000
40.00	0.200000	0.600000	0.200000
41.00	1.000000	0.000000	0.000000
42.00	0.000000	0.000000	1.000000
43.00	1.000000	0.000000	0.000000
43.05	0.444444	0.222222	0.333333

Probability of buying a product based on Gender:

Product	KP281	KP481	KP781
Gender			
Female	0.526316	0.381579	0.092105
Male	0.384615	0.298077	0.317308

Probability of buying a product based on Education:

Product	KP281	KP481	KP781
Education			
14	0.555556	0.412698	0.031746
15	0.800000	0.200000	0.000000
16	0.458824	0.364706	0.176471
18	0.074074	0.074074	0.851852

Probability of buying a product based on MaritalStatus:

Product	KP281	KP481	KP781
MaritalStatus			
Partnered	0.448598	0.336449	0.214953
Single	0.438356	0.328767	0.232877

Probability of buying a product based on Usage:

Product	KP281	KP481	KP781
Usage			
2.00	0.575758	0.424242	0.000000
3.00	0.536232	0.449275	0.014493
4.00	0.423077	0.230769	0.346154
5.00	0.117647	0.176471	0.705882
5.05	0.000000	0.000000	1.000000

Probability of buying a product based on Fitness:

Product	KP281	KP481	KP781

```
Fitness
2        0.535714  0.464286  0.000000
3        0.556701  0.402062  0.041237
4        0.375000  0.333333  0.291667
5        0.064516  0.000000  0.935484

Probability of buying a product based on Income:
Product      KP281     KP481  KP781
Income
34053.15  0.666667  0.333333    0.0
34110.00  0.400000  0.600000    0.0
35247.00  1.000000  0.000000    0.0
36384.00  0.750000  0.250000    0.0
37521.00  1.000000  0.000000    0.0
38658.00  0.600000  0.400000    0.0
39795.00  1.000000  0.000000    0.0
40932.00  0.666667  0.333333    0.0
42069.00  1.000000  0.000000    0.0
43206.00  0.200000  0.800000    0.0
44343.00  1.000000  0.000000    0.0
45480.00  0.357143  0.642857    0.0
46617.00  0.875000  0.125000    0.0
47754.00  0.000000  1.000000    0.0
48556.00  0.000000  0.000000    1.0
48658.00  0.000000  0.000000    1.0
48891.00  0.400000  0.600000    0.0
49801.00  0.000000  0.000000    1.0
50028.00  0.285714  0.714286    0.0
51165.00  0.428571  0.571429    0.0
52290.00  0.000000  0.000000    1.0
52291.00  0.000000  0.000000    1.0
52302.00  0.666667  0.333333    0.0
53439.00  0.375000  0.625000    0.0
53536.00  0.000000  0.000000    1.0
54576.00  0.875000  0.125000    0.0
54781.00  0.000000  0.000000    1.0
55713.00  1.000000  0.000000    0.0
56850.00  1.000000  0.000000    0.0
57271.00  0.000000  0.000000    1.0
57987.00  0.250000  0.750000    0.0
58516.00  0.000000  0.000000    1.0
59124.00  0.333333  0.666667    0.0
60261.00  0.666667  0.333333    0.0
61006.00  0.000000  0.000000    1.0
61398.00  0.500000  0.500000    0.0
62251.00  0.000000  0.000000    1.0
62535.00  0.000000  1.000000    0.0
64741.00  0.000000  0.000000    1.0
64809.00  0.333333  0.666667    0.0
65220.00  0.000000  1.000000    0.0
67083.00  0.500000  0.500000    0.0
68220.00  1.000000  0.000000    0.0
69721.00  0.000000  0.000000    1.0
70966.00  0.000000  0.000000    1.0
74701.00  0.000000  0.000000    1.0
75946.00  0.000000  0.000000    1.0
77191.00  0.000000  0.000000    1.0
83416.00  0.000000  0.000000    1.0
85906.00  0.000000  0.000000    1.0
88396.00  0.000000  0.000000    1.0
```

```
89641.00  0.000000  0.000000    1.0
90886.00  0.000000  0.000000    1.0
90948.25  0.000000  0.000000    1.0

Probability of buying a product based on Miles:
Product     KP281     KP481     KP781
Miles
47       0.705882  0.294118  0.000000
53       0.000000  1.000000  0.000000
56       1.000000  0.000000  0.000000
64       0.000000  1.000000  0.000000
66       1.000000  0.000000  0.000000
74       0.000000  1.000000  0.000000
75       1.000000  0.000000  0.000000
80       0.000000  0.000000  1.000000
85       0.592593  0.407407  0.000000
94       1.000000  0.000000  0.000000
95       0.000000  1.000000  0.000000
100      0.000000  0.000000  1.000000
103      1.000000  0.000000  0.000000
106      0.000000  0.888889  0.111111
112      1.000000  0.000000  0.000000
113      1.000000  0.000000  0.000000
120      0.000000  0.000000  1.000000
127      0.000000  1.000000  0.000000
132      1.000000  0.000000  0.000000
140      0.000000  0.000000  1.000000
141      1.000000  0.000000  0.000000
150      0.000000  0.000000  1.000000
160      0.000000  0.000000  1.000000
169      1.000000  0.000000  0.000000
170      0.000000  0.666667  0.333333
180      0.000000  0.000000  1.000000
188      1.000000  0.000000  0.000000
200      0.000000  0.083333  0.916667
```

1. The KP281 treadmill appears to be popular among individuals with lower incomes, younger ages, and lower fitness levels.
2. Females are more likely to purchase the KP281 treadmill compared to males.
3. Education level and usage frequency also influence the probability of buying the KP281 treadmill, with certain education levels and lower usage frequencies showing a higher probability.
4. Income seems to have a significant impact on purchasing decisions, with lower income individuals showing a higher probability of buying the KP281 treadmill.

Conditional Probability

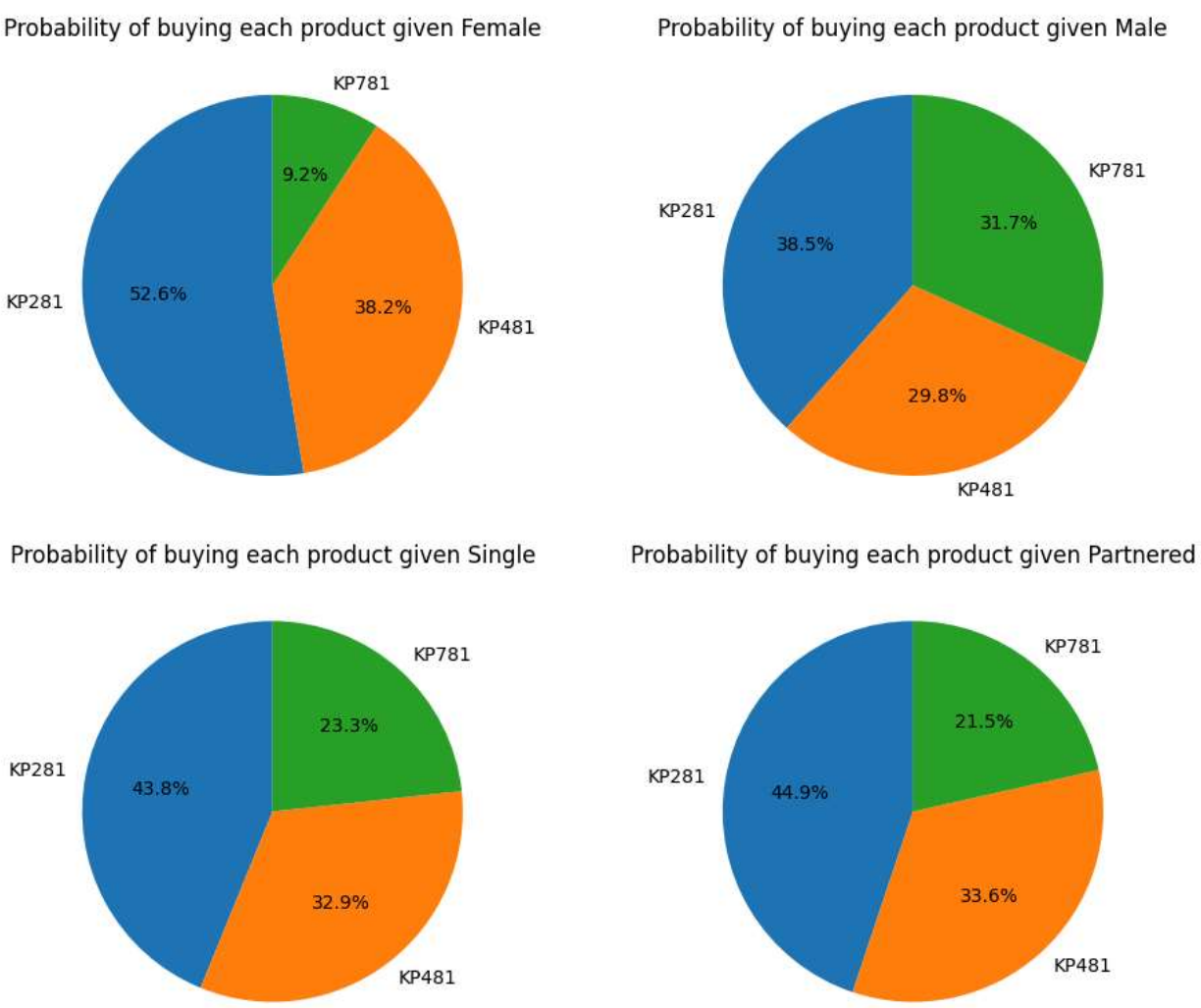
(for Gender, Maritial Status)

In [16]:
```
# Given a customer is female, what is its probability to buy KP281, KP481, KP781
prob_female = conditional_probs['Gender'].loc['Female']

# Given a customer is male, what is its probability to buy KP281, KP481, KP781
prob_male = conditional_probs['Gender'].loc['Male']

# Given a customer is single, what is its probability to buy KP281, KP481, KP781
prob_single = conditional_probs['MaritalStatus'].loc['Single']

# Given a customer is partnered, what is its probability to buy KP281, KP481, KP781
prob_partnered = conditional_probs['MaritalStatus'].loc['Partnered']

# Displaying results
print("Conditional Probabilities:")
print("\nGiven a customer is female, probability to buy a product:")
print(prob_female)
print("\nGiven a customer is male, probability to buy KP281, KP481, KP781:")
print(prob_male)
print("\nGiven a customer is single, probability to buy KP281, KP481, KP781:")
print(prob_single)
print("\nGiven a customer is partnered, probability to buy KP281, KP481, KP781:")
print(prob_partnered)
```

```
Conditional Probabilities:

Given a customer is female, probability to buy a product:
Product
KP281    0.526316
KP481    0.381579
KP781    0.092105
Name: Female, dtype: float64

Given a customer is male, probability to buy KP281, KP481, KP781:
Product
KP281    0.384615
KP481    0.298077
KP781    0.317308
Name: Male, dtype: float64

Given a customer is single, probability to buy KP281, KP481, KP781:
Product
KP281    0.438356
KP481    0.328767
KP781    0.232877
Name: Single, dtype: float64

Given a customer is partnered, probability to buy KP281, KP481, KP781:
Product
KP281    0.448598
KP481    0.336449
KP781    0.214953
Name: Partnered, dtype: float64
```

In [17]:
```
# Define data for pie charts
data = {
    'Female': prob_female,
    'Male': prob_male,
    'Single': prob_single,
    'Partnered': prob_partnered
```

```
}

# Define products
products = ['KP281', 'KP481', 'KP781']

# Plot pie charts for each condition
fig, axs = plt.subplots(2, 2, figsize=(10, 8))
axs = axs.flatten()

for i, (condition, probs) in enumerate(data.items()):
    ax = axs[i]
    ax.pie(probs, labels=products, autopct='%1.1f%%', startangle=90)
    ax.set_title(f'Probability of buying each product given {condition}')

plt.tight_layout()
plt.show()
```



Gender:

1. Females have a higher probability of buying the KP281 treadmill compared to males.

2. Males show a more balanced probability across all treadmill models.

Marital Status:

1. Single individuals have a slightly lower probability of buying the KP281 treadmill compared to partnered individuals.

2. Single individuals show a slightly higher probability of buying the KP481 treadmill compared to partnered individuals.

Marketing Implications:

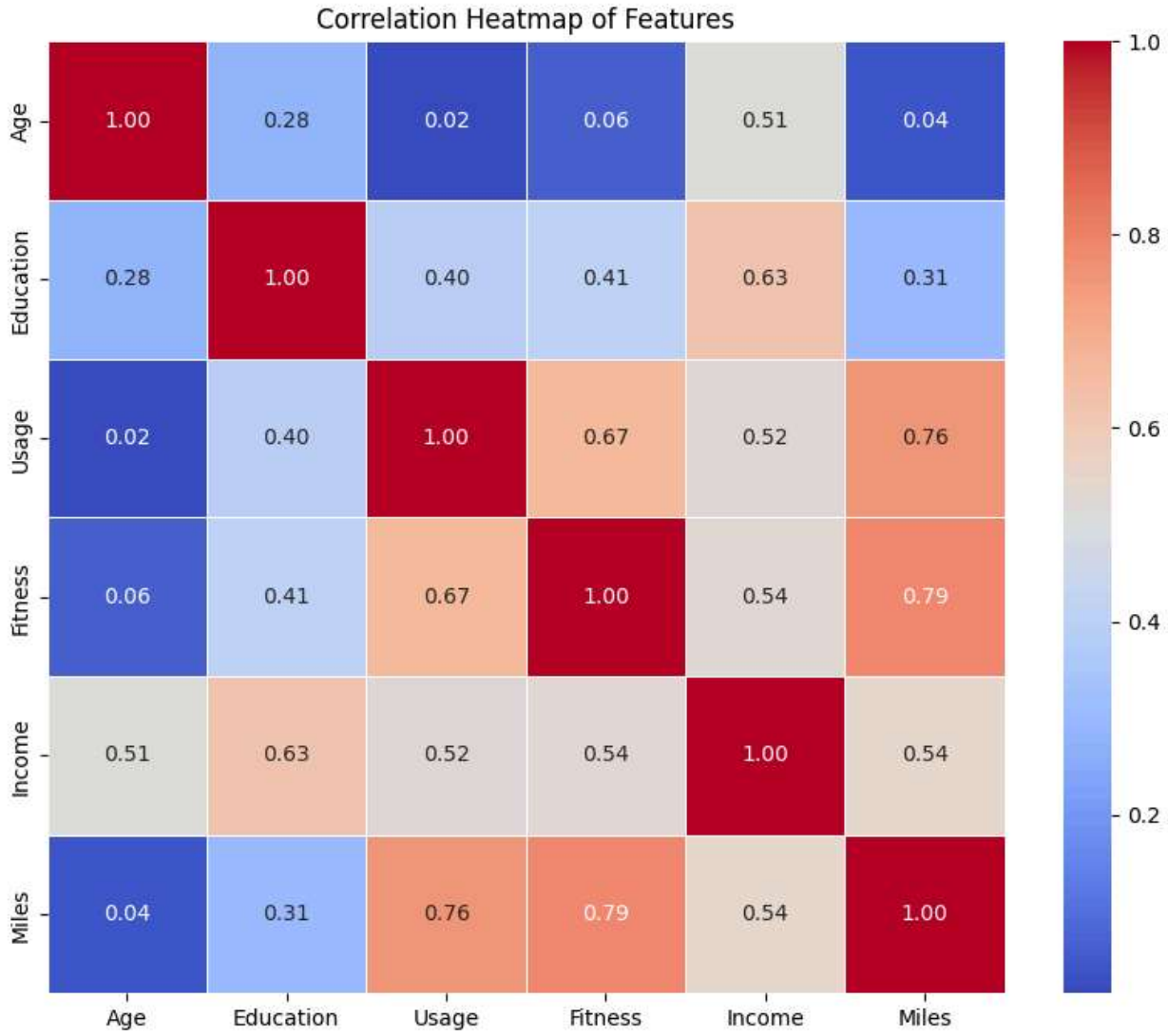
1. Gender-Based Segmentation: Since females are more inclined to buy the KP281 treadmill, targeted marketing efforts can be directed towards this segment. This might include advertisements featuring female athletes, fitness influencers, or highlighting features that resonate more with female customers.

2. Marital Status-Based Segmentation: Tailored marketing strategies can be implemented based on marital status. For example, offering couple discounts or promotions targeting partnered individuals could be effective for increasing sales of certain treadmill models.

Correlation among different factors

In [20]:
```
# Compute the correlation matrix
correlation_matrix = data.corr()

# Plot heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5
plt.title('Correlation Heatmap of Features')
plt.show()
```

```
<ipython-input-20-e264ba181ab3>:2: FutureWarning: The default value of numeric_only i
n DataFrame.corr is deprecated. In a future version, it will default to False. Select
only valid columns or specify the value of numeric_only to silence this warning.
  correlation_matrix = data.corr()
```



1. Features like "Usage" and "Fitness", as well as "Fitness" and "Miles", are strongly positively correlated, indicating that an increase in one is associated with an increase in another.
2. Income has moderate positive correlations with Education, Usage, and Fitness.

Customer Profile for KP281

In [21]:
```
# Filter data for KP281
kp281_data = data[data['Product'] == 'KP281']

# Compute average age, income, and fitness for KP281
kp281_avg_age = kp281_data['Age'].mean()
kp281_avg_income = kp281_data['Income'].mean()
kp281_avg_fitness = kp281_data['Fitness'].mean()
kp281_gender_distribution = kp281_data['Gender'].value_counts(normalize=True)

# Display average values for KP281
print("Customer Profile for KP281")
print("Average Age:", kp281_avg_age)
print("Average Income:", kp281_avg_income)
print("Average Fitness:", kp281_avg_fitness)
```

```
print("Gender Distribution:")
print(kp281_gender_distribution)
```

```
Customer Profile for KP281
Average Age: 28.55
Average Income: 46418.025
Average Fitness: 2.9625
Gender Distribution:
Male      0.5
Female    0.5
Name: Gender, dtype: float64
```

Customer Profile for KP281:

Average Age: 28.55 years

Average Income: $46,418.025

Average Fitness: 2.9625

Gender Distribution: Equal distribution between male and female customers (50% each).

Customer Profile for KP481

In [22]:
```
# Filter data for KP481
kp481_data = data[data['Product'] == 'KP481']

# Compute average age, income, and fitness for KP481
kp481_avg_age = kp481_data['Age'].mean()
kp481_avg_income = kp481_data['Income'].mean()
kp481_avg_fitness = kp481_data['Fitness'].mean()
kp481_gender_distribution = kp481_data['Gender'].value_counts(normalize=True)

# Display average values for KP481
print("\nCustomer Profile for KP481")
print("Average Age:", kp481_avg_age)
print("Average Income:", kp481_avg_income)
print("Average Fitness:", kp481_avg_fitness)
print("Gender Distribution:")
print(kp481_gender_distribution)
```

```
Customer Profile for KP481
Average Age: 28.9
Average Income: 48973.65
Average Fitness: 2.9
Gender Distribution:
Male      0.516667
Female    0.483333
Name: Gender, dtype: float64
```

Customer Profile for KP481:

Average Age: 28.9 years

Average Income: $48,973.65

Average Fitness: 2.9

Gender Distribution: Slightly higher proportion of male customers (51.67%) compared to female customers (48.33%).

Customer Profile for KP781

In [24]:
```
# Filter data for KP781
kp781_data = data[data['Product'] == 'KP781']

# Compute average age, income, and fitness for KP781
kp781_avg_age = kp781_data['Age'].mean()
kp781_avg_income = kp781_data['Income'].mean()
kp781_avg_fitness = kp781_data['Fitness'].mean()
kp781_gender_distribution = kp781_data['Gender'].value_counts(normalize=True)

# Display average values for KP781
print("\nCustomer Profile for KP781")
print("Average Age:", kp781_avg_age)
print("Average Income:", kp781_avg_income)
print("Average Fitness:", kp781_avg_fitness)
print("Gender Distribution:")
print(kp781_gender_distribution)
```

```
Customer Profile for KP781
Average Age: 29.1
Average Income: 75441.575
Average Fitness: 4.625
Gender Distribution:
Male      0.825
Female    0.175
Name: Gender, dtype: float64
```

Customer Profile for KP781:

Average Age: 29.1 years

Average Income: $75,441.575

Average Fitness: 4.625

Gender Distribution: Majority of customers are male (82.5%), with a smaller proportion of female customers (17.5%).

Recommendation

Segmented Strategies:

1. Age Segmentation: Target marketing efforts for KP281 towards the younger demographic, while KP781 could be marketed towards slightly older customers who may have higher incomes

and prioritize advanced features.

2. Income Segmentation: Position KP781 as a premium option for customers with higher incomes, while KP281 and KP481 could be marketed as more affordable alternatives for those with moderate incomes.

3. Fitness Level Segmentation: Develop tailored marketing campaigns highlighting how each treadmill model caters to different fitness levels. For example, KP781 could be positioned as a high-performance treadmill for serious fitness enthusiasts, while KP281 and KP481 could target casual exercisers or those new to fitness.

4. Gender-Targeted Advertising: Since there are variations in gender distribution among the treadmill models, tailor advertising strategies accordingly. For instance, for KP781, which has a higher proportion of male customers, focus more on features that appeal to male fitness enthusiasts.

Product Positioning:

1. KP781 as Premium Option: Highlight the advanced features, higher fitness levels, and superior performance of KP781 to position it as a premium treadmill for customers willing to invest in top-of-the-line fitness equipment.

2. KP281 and KP481 as Affordable Alternatives: Emphasize the affordability and value for money proposition of KP281 and KP481 compared to KP781, targeting customers looking for quality treadmills at more accessible price points.

3. Customized Features:

KP781 (Premium Option):

Marketing Focus: Highlight its sleek design and advanced technology.

Target Demographic: Serious fitness enthusiasts willing to invest in top-quality equipment.

KP281 and KP481 (Affordable Alternatives):

Marketing Focus: Emphasize user-friendly features and ease of use.

Target Demographic: Casual exercisers or those seeking value for money options.

In [25]:
```
print("Done")
```

```
Done
```