Aerofit - Business Case Study - Descriptive Statistics & Probability

The main objective of this case study is to provide insights about the characterstics of the targeted audience for each type of treadmill offered Aerofit, to provide a better recommendation of the treadmills to the new customers.

About the dataset

Aerofit is a leading brand in the field of fitness equipment. Aerofit provides a product range including machines such as treadmills, exercise bikes, gym equipment, and fitness accessories to cater to the needs of all categories of people.

The company collected the data on individuals who purchased a treadmill from the AeroFit stores during the prior three months. The dataset has the following features:

- **Product Purchased**: KP281, KP481, or KP781
- **Age**: In years
- **Gender**: Male/Female
- **Education**: In years
- **MaritalStatus**: Single or partnered
- **Usage**: The average number of times the customer plans to use the treadmill each week.
- **Income**: Annual income (in $)
- **Fitness**: Self-rated fitness on a 1-to-5 scale, where 1 is the poor shape and 5 is the excellent shape.
- **Miles**: The average number of miles the customer expects to walk/run each week

And below is the Product Portfolio :

- **KP281**: an entry-level treadmill that sells for $1,500
- **KP481**: for mid-level runners that sell for $1,750
- **KP781**: treadmill with advanced features that sell for $2,500

Business Problem:

Analyse the dataset and generate the insights on characteristics of the target audience for each type of treadmill offered by the company, to provide a better recommendation of the treadmills to the new customers. The team then decides investigate whether there are differences across the product with respect to customer characteristics

Objective:

1. Perform descriptive analytics to create a customer profile for each AeroFit treadmill product by developing appropriate tables and charts.
2. For each AeroFit treadmill product, construct two-way contingency tables and compute all conditional and marginal probabilities along with their insights/impact on the business.

1. Importing Libraries, Loading Dataset and Analyzing Basic Metrics

1.1 Importing Libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

1.2 Loading Dataset

In [2]:
```
data = pd.read_csv("aerofit_treadmill.csv")
data
```

Out[2]:

	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
...	...	...	...	...	...	...	...	...	...
175	KP781	40	Male	21	Single	6	5	83416	200
176	KP781	42	Male	18	Single	5	4	89641	200
177	KP781	45	Male	16	Single	5	5	90886	160
178	KP781	47	Male	18	Partnered	4	5	104581	120
179	KP781	48	Male	18	Partnered	4	5	95508	180

180 rows × 9 columns

1.3 Basic Metrics:

In [3]:
```
# Checking Shape of the data
data.shape
```

Out[3]:
```
(180, 9)
```

By looking into the shape of data, we have **180** rows and **9** columns

In [4]:
```
# Columns in the data
data.columns
```

Out[4]:
```
Index(['Product', 'Age', 'Gender', 'Education', 'MaritalStatus', 'Usage',
       'Fitness', 'Income', 'Miles'],
      dtype='object')
```

We have below 9 columns in the loaded data:

1. Product
2. Age
3. Gender
4. Education
5. MartialStatus
6. Usage
7. Fitness
8. Income
9. Miles

In [5]:
```
# Data Types of all Attributes
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
```

Below are the datatypes of all attributes:

---

1. "Product" is a string
2. "Age" is a number
3. "Gender" is a string
4. "Education" is a number
5. "MartialStatus" is a string
6. "Usage" is a number
7. "Fitness" is a number
8. "Income" is a number
9. "Miles" is a number

From above data types, we can convert "Product" into category

1.4 Converting Product to Category

In [6]:
```
data['Product'] = data['Product'].astype("category")
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    category
 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: category(1), int64(6), object(2)
memory usage: 11.7+ KB
```

Column "Product" is converted into Category

1.5 Checking for Null Values

In [7]:
```
data.isnull().sum()
```

Out[7]:
```
Product          0
Age              0
Gender           0
Education        0
MaritalStatus    0
Usage            0
Fitness          0
Income           0
Miles            0
dtype: int64
```

There are no null values in the dataset

1.6 Statistical Summary

In [8]:
```
data.describe()
```

Out[8]:

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

By observing the statistical summary, below are the insights:

1. **Age**: Minimum age of the customer in the dataset is 18 years, while Maximum age is 50 years. Average age if customers is 28.7 years
2. **Education**: Customers had minimum education for 12 years while maximum education of 21 years. Average years of education is around 15 years
3. **Usage**: Minimum number of times the customer plans to use the treadmill each week is 2 times where as Maximum number of times is 7 times. On an average customer uses 3.4 times in a week
4. **Fitness**: Few customers have self rated themselves as 1 (Minimum) for fitness whereas few have given 5 (Maximum) with average rating of around 3 rating
5. **Income**: Minimum salary of customer in dataset is 29562 whereas Maximum salary is 104581 with average salary of 53719
6. **Miles**: The minimum number of miles the customer expects to walk/run each week is 21 miles and maximum number is 360 miles. On an average customer plans to walk to 103 miles

---

In [9]:
```
data.describe(include = object)
```

Out[9]:

	Gender	MaritalStatus
count	180	180
unique	2	2
top	Male	Partnered
freq	104	107

While analysing the columns of non numeric values, below are our insights:

1. **Gender**: There are 2 unique values in Gender column. There are 104 times Male is repeated in the Gender column.
2. **MaritalStatus**: There are 2 unique values in MaritialStatus column. "Partnered" is repeated 107 times

---

2 Non Graphical Analysis - Value Counts and Unique Attributes

2.1 Value Counts

2.1.1 Value Count of Product field

In [10]:
```
data['Product'].value_counts()
```

Out[10]:
```
Product
KP281    80
KP481    60
KP781    40
Name: count, dtype: int64
```

By observing the value counts, we can see that product **KP281** is sold more compared to other two products and product **KP781** is least sold in the dataset

---

2.1.2 Value Counts of Age field

In [11]:
```
data['Age'].value_counts()[:5]
```

Out[11]:
```
Age
25    25
23    18
24    12
26    12
28     9
Name: count, dtype: int64
```

Top 5 customer ages from the dataset that bought the treadmills are 25, 23, 24, 26 and 28 ages

---

2.1.3 Value Counts of Gender field

In [12]:
```
data['Gender'].value_counts()
```

Out[12]:
```
Gender
Male      104
Female     76
Name: count, dtype: int64
```

From the dataset, 104 customers are **Male** and 76 customers are **Female**

2.1.4 Value Counts of Education field

In [13]:
```
data['Education'].value_counts()
```

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

---

85 customers have **16 years** of education, 55 customers have **14 years** of education and 23 customers have **18 years** of education

---

2.1.5 Value Counts of MaritalStatus field

In [14]:
```
data['MaritalStatus'].value_counts()
```

Out[14]:
```
MaritalStatus
Partnered    107
Single        73
Name: count, dtype: int64
```

---

107 customers are **Partnered** and 73 customers are **single**

2.1.6 Value Counts of Usage field

In [15]:
```
data['Usage'].value_counts()
```

Out[15]:
```
Usage
3    69
4    52
2    33
5    17
6     7
7     2
Name: count, dtype: int64
```

---

69 customers have said they will use treadmill atleast **3 times** a week, 52 customers have said **4 times** a week while 33 customers said **2 times** a week

---

2.1.7 Value Counts of Fitness field

In [16]:
```
data['Fitness'].value_counts()
```

Out[16]:
```
Fitness
3    97
5    31
2    26
4    24
1     2
Name: count, dtype: int64
```

---

97 customers have rated **3** on their fitness level, 31 customers rated **5** on their fitness level where as 26 customers rated **2**

2.2 Unique Attributes

In [17]:
```
data.columns
```

Out[17]:
```
Index(['Product', 'Age', 'Gender', 'Education', 'MaritalStatus', 'Usage',
       'Fitness', 'Income', 'Miles'],
      dtype='object')
```

2.2.1 Unique values in Product

In [18]:
```
data['Product'].unique()
```

Out[18]:
```
['KP281', 'KP481', 'KP781']
Categories (3, object): ['KP281', 'KP481', 'KP781']
```

2.2.2 Unique values in Age

In [19]:
```
data['Age'].unique()
```

Out[19]:
```
array([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])
```

2.2.3 Unique values in Gender

In [20]:
```
data['Gender'].unique()
```

Out[20]:
```
array(['Male', 'Female'], dtype=object)
```

2.2.4 Unique values in Education

In [21]:
```
data['Education'].unique()
```

Out[21]:
```
array([14, 15, 12, 13, 16, 18, 20, 21])
```

2.2.5 Unique values in MaritalStatus

In [22]:
```
data['MaritalStatus'].unique()
```

Out[22]:
```
array(['Single', 'Partnered'], dtype=object)
```

2.2.6 Unique values in Usage

In [23]:
```
data['Usage'].unique()
```

Out[23]:
```
array([3, 2, 4, 5, 6, 7])
```

2.2.7 Unique values in Fitness

In [24]:
```
data['Fitness'].unique()
```

Out[24]:
```
array([4, 3, 2, 1, 5])
```

Observations:

- There are 3 unique values in **Product** field - 'KP281', 'KP481', 'KP781'
- **Age** range of customers are from 18 to 50
- **Gender** field has both Male and Female values
- **Education** field has range from 12 to 21 years
- **Marital Status** field has both Partnered and Single values
- **Usage range** is from 2 to 7 times a week
- **Fitness ranges** from 1 to 5

---

3. Visual Analysis - Univariate & Bivariate

3.1 Univariate Analysis

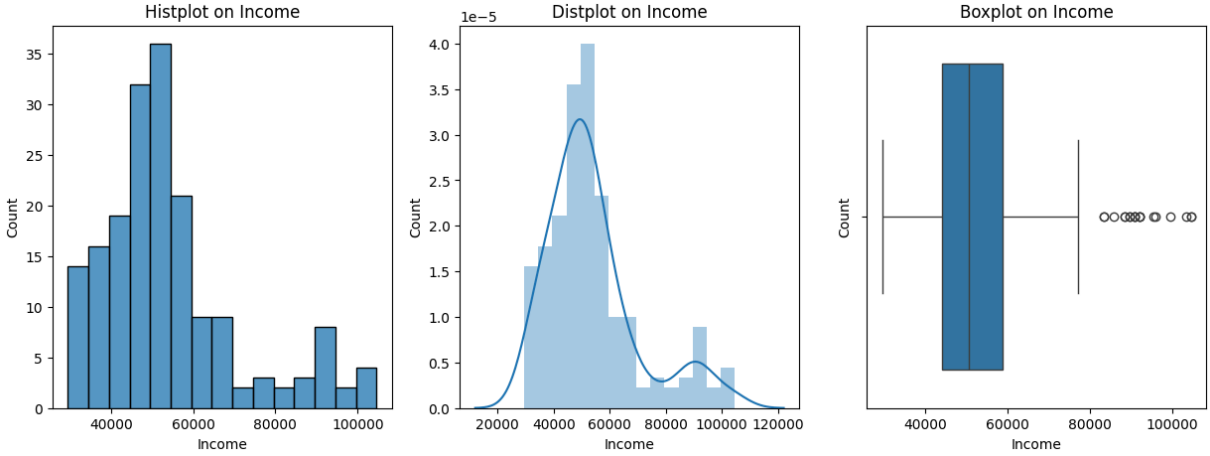
3.1.1 Plotting for Continuous Data

3.1.1 A Income

In [25]:
```
plt.figure(figsize = (15,5))

plt.subplot(1,3,1)
plt.xlabel("Income")
plt.ylabel("Count")
plt.title("Histplot on Income")
#plt.xticks(rotation = 90)
sns.histplot(data['Income'])
```

```
plt.subplot(1,3,2)
plt.xlabel("Income")
plt.ylabel("Count")
plt.title("Distplot on Income")
#plt.xticks(rotation = 90)
sns.distplot(data['Income'])

plt.subplot(1,3,3)
plt.xlabel("Income")
plt.ylabel("Count")
plt.title("Boxplot on Income")
#plt.xticks(rotation = 90)
sns.boxplot(data = data, x = 'Income')

plt.show()
```

```
/tmp/ipykernel_1681/2504577252.py:15: UserWarning:

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

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

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

  sns.distplot(data['Income'])
```



In [26]:
```
print("Mean of Income Column: ", np.mean(data['Income']))
print("Median of Income Column: ", np.median(data['Income']))
```

```
Mean of Income Column:  53719.57777777778
Median of Income Column:  50596.5
```

Observation:

We can identify that there is difference between mean and median in Income column which may be a sign of outliers in the data.

Computing Quantiles, Lowerbound and Upperbound

In [27]:
```
income_q1 = np.percentile(data['Income'], 25)
income_q3 = np.percentile(data['Income'], 75)
income_iqr = income_q3 - income_q1
income_upperbound = income_q3 + 1.5 * income_iqr
income_lowerbound = income_q1 - 1.5 * income_iqr
```

In [28]:
```
print("First Quantile of Income Column: ", income_q1)
print("Second Quantile (Median) of Income Column: ", np.percentile(data['Income'], 50))
print("Third Quantile of Income Column: ", income_q3)
print("IQR of Income Column: ", income_iqr)
print("Lowerbound of Income Column: ", income_lowerbound)
print("Upperbound of Income Column: ", income_upperbound)
```

```
First Quantile of Income Column:  44058.75
Second Quantile (Median) of Income Column:  50596.5
Third Quantile of Income Column:  58668.0
IQR of Income Column:  14609.25
Lowerbound of Income Column:  22144.875
Upperbound of Income Column:  80581.875
```

In [29]:
```
# Outliers in Income Field
data_income_outliers = data.loc[data['Income'] > income_upperbound]
data_income_outliers.head()
```

Out[29]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
159	KP781	27	Male	16	Partnered	4	5	83416	160
160	KP781	27	Male	18	Single	4	3	88396	100
161	KP781	27	Male	21	Partnered	4	4	90886	100
162	KP781	28	Female	18	Partnered	6	5	92131	180
164	KP781	28	Male	18	Single	6	5	88396	150

In [30]:
```
# % of outliers
print("Number of outliers in Income field: ", len(data_income_outliers))
print("% of outliers in Income Column: ",np.round(len(data_income_outliers)/ len(data['Income']), 4))
```

```
Number of outliers in Income field:  19
% of outliers in Income Column:  0.1056
```

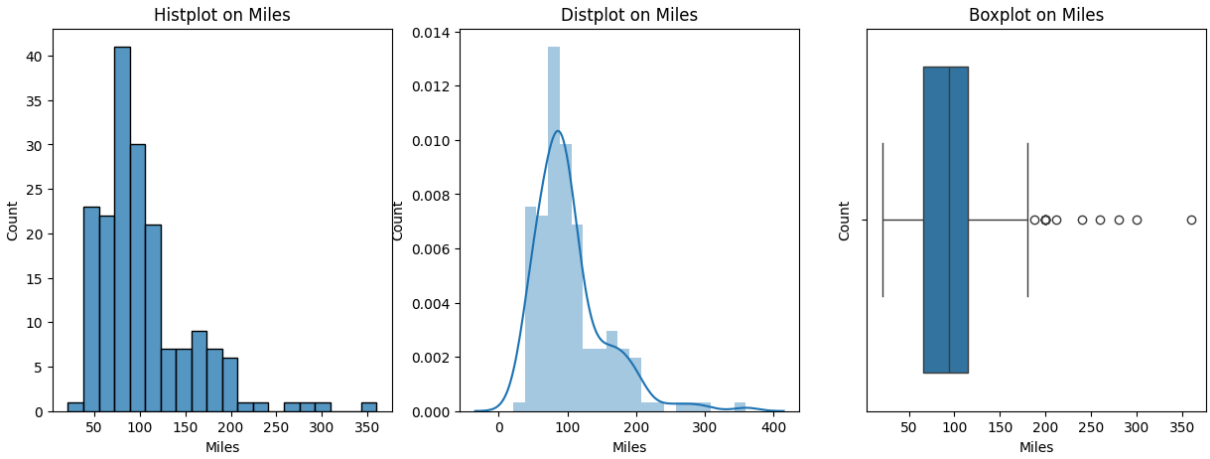
Insights from Income Column

1. Mean income of the customers is `Rs:/-53719` whereas Median is `Rs:/-50596.5` . We can identify that there is difference between mean and median in Income column which may be a sign of outliers in the data.
2. Lower bound of Income is `Rs:-22145` and Upper bound is `Rs:-80582` .
3. There are `19` outliers in Income column and this is around `11%` of data. We cannot drop these rows as we can loose information
4. There are around `60%` of data lies between `Rs:-44058` & `Rs:-58668`

3.1.1 B Miles

In [31]:
```
plt.figure(figsize = (15,5))

plt.subplot(1,3,1)
plt.xlabel("Miles")
plt.ylabel("Count")
plt.title("Histplot on Miles")
#plt.xticks(rotation = 90)
sns.histplot(data['Miles'])

plt.subplot(1,3,2)
plt.xlabel("Miles")
plt.ylabel("Count")
plt.title("Distplot on Miles")
#plt.xticks(rotation = 90)
sns.distplot(data['Miles'])

plt.subplot(1,3,3)
plt.xlabel("Miles")
plt.ylabel("Count")
plt.title("Boxplot on Miles")
#plt.xticks(rotation = 90)
sns.boxplot(data = data, x = 'Miles')

plt.show()
```

```
/tmp/ipykernel_1681/2644909888.py:15: UserWarning:

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

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

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

  sns.distplot(data['Miles'])
```



In [32]:
```
print("Mean of Miles Column: ", np.mean(data['Miles']))
print("Median of Miles Column: ", np.median(data['Miles']))
```

```
Mean of Miles Column:  103.19444444444444
Median of Miles Column:  94.0
```

Observation:

We can identify that there is difference between mean and median in Income column which may be a sign of outliers in the data.

Computing Quantiles, Lowerbound and Upperbound

In [33]:
```
miles_q1 = np.percentile(data['Miles'], 25)
miles_q3 = np.percentile(data['Miles'], 75)
miles_iqr = miles_q3 - miles_q1
miles_upperbound = miles_q3 + 1.5 * miles_iqr
miles_lowerbound = miles_q1 - 1.5 * miles_iqr
if miles_lowerbound < 0:
    miles_lowerbound = 0
```

In [34]:
```
print("First Quantile of Miles Column: ", miles_q1)
print("Second Quantile (Median) of Miles Column: ", np.percentile(data['Miles'], 50))
print("Third Quantile of Miles Column: ", miles_q3)
print("IQR of Miles Column: ",miles_iqr)
print("Lowerbound of Miles Column: ", miles_lowerbound)
print("Upperbound of Miles Column: ", miles_upperbound)
```

```
First Quantile of Miles Column:  66.0
Second Quantile (Median) of Miles Column:  94.0
Third Quantile of Miles Column:  114.75
IQR of Miles Column:  48.75
Lowerbound of Miles Column:  0
Upperbound of Miles Column:  187.875
```

In [35]:
```
# Outliers in Miles Field
data_miles_outliers = data.loc[data['Miles'] > miles_upperbound]
data_miles_outliers.head()
```

Out[35]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
23	KP281	24	Female	16	Partnered	5	5	44343	188
84	KP481	21	Female	14	Partnered	5	4	34110	212
142	KP781	22	Male	18	Single	4	5	48556	200
148	KP781	24	Female	16	Single	5	5	52291	200
152	KP781	25	Female	18	Partnered	5	5	61006	200

In [36]:
```
# % of outliers
print("Number of outliers in Miles field: ", len(data_miles_outliers))
print("% of outliers in Miles Column: ",np.round(len(data_miles_outliers)/ len(data['Miles']), 4))
```

```
Number of outliers in Miles field:  13
% of outliers in Miles Column:  0.0722
```

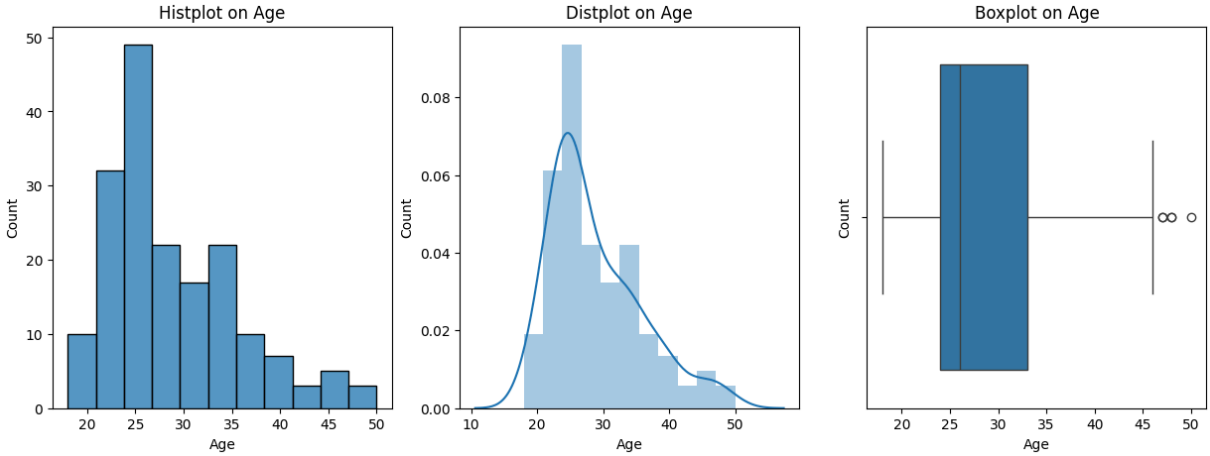
Insights from Miles Column

1. Mean miles is `103` whereas Median is `94`. We can identify that there is difference between mean and median in Miles column which may be a sign of outliers in the data.
2. Lower bound of Miles is `0` and Upper bound is `188`.
3. There are `13` outliers in Miles column and this is around `7%` of data. We cannot drop these rows as we can loose information
4. There are around `~60%` of data lies between `66` & `114`

3.1.1 C Age

In [37]:
```
plt.figure(figsize = (15,5))

plt.subplot(1,3,1)
plt.xlabel("Age")
plt.ylabel("Count")
plt.title("Histplot on Age")
#plt.xticks(rotation = 90)
sns.histplot(data['Age'])

plt.subplot(1,3,2)
plt.xlabel("Age")
plt.ylabel("Count")
plt.title("Distplot on Age")
#plt.xticks(rotation = 90)
sns.distplot(data['Age'])

plt.subplot(1,3,3)
plt.xlabel("Age")
plt.ylabel("Count")
plt.title("Boxplot on Age")
#plt.xticks(rotation = 90)
sns.boxplot(data = data, x = 'Age')

plt.show()
```

```
/tmp/ipykernel_1681/662075743.py:15: UserWarning:

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

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

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

  sns.distplot(data['Age'])
```



In [38]:
```
print("Mean of Age Column: ", np.mean(data['Age']))
print("Median of Age Column: ", np.median(data['Age']))
```

```
Mean of Age Column:  28.788888888888888
Median of Age Column:  26.0
```

Observation:

We can identify that there is difference between mean and median.

Computing Quantiles, Lowerbound and Upperbound

In [39]:
```
age_q1 = np.percentile(data['Age'], 25)
age_q3 = np.percentile(data['Age'], 75)
age_iqr = age_q3 - age_q1
age_upperbound = age_q3 + 1.5 * age_iqr
age_lowerbound = age_q1 - 1.5 * age_iqr
if age_lowerbound < 0:
    age_lowerbound = 0
```

In [40]:
```
print("First Quantile of Age Column: ", age_q1)
print("Second Quantile (Median) of Age Column: ", np.percentile(data['Age'], 50))
print("Third Quantile of Age Column: ", age_q3)
print("IQR of Age Column: ",age_iqr)
print("Lowerbound of Age Column: ", age_lowerbound)
print("Upperbound of Age Column: ", age_upperbound)
```

```
First Quantile of Age Column:  24.0
Second Quantile (Median) of Age Column:  26.0
Third Quantile of Age Column:  33.0
IQR of Age Column:  9.0
Lowerbound of Age Column:  10.5
Upperbound of Age Column:  46.5
```

In [41]:
```
# Outliers in Age Field
data_age_outliers = data.loc[data['Age'] > age_upperbound]
data_age_outliers.head()
```

Out[41]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
78	KP281	47	Male	16	Partnered	4	3	56850	94
79	KP281	50	Female	16	Partnered	3	3	64809	66
139	KP481	48	Male	16	Partnered	2	3	57987	64
178	KP781	47	Male	18	Partnered	4	5	104581	120
179	KP781	48	Male	18	Partnered	4	5	95508	180

In [42]:
```
# % of outliers
print("Number of outliers in Age field: ", len(data_age_outliers))
print("% of outliers in Age Column: ",np.round(len(data_age_outliers)/ len(data['Age']), 4))
```

```
Number of outliers in Age field:  5
% of outliers in Age Column:  0.0278
```

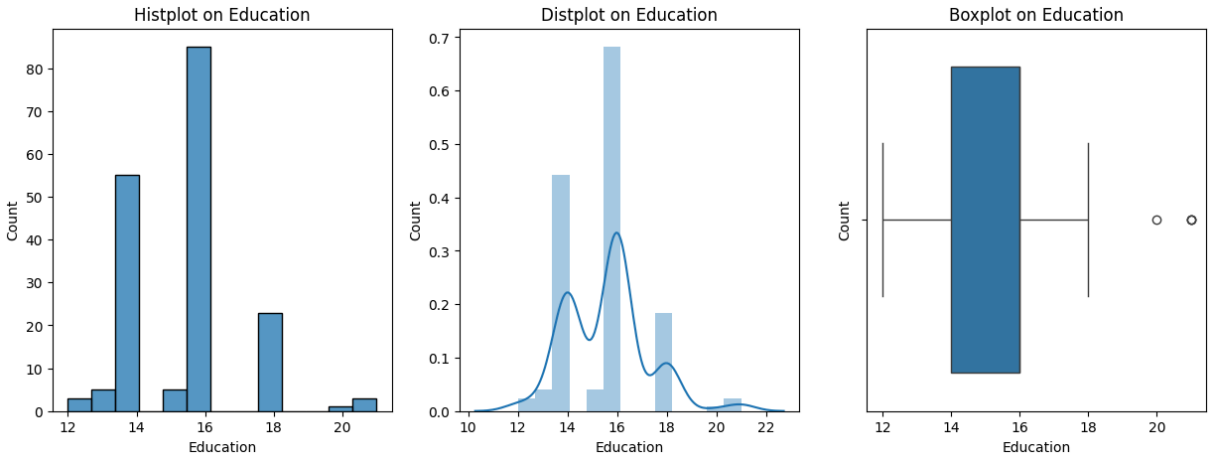
Insights from Age Column

1. Mean age is `28.7` whereas Median is `26`. We can identify that there is difference between mean and median.
2. Lower bound of Age is `10.5` and Upper bound is `46.5`.
3. There are `5` outliers in Age column and this is around `2%` of data.
4. There are around `~42%` of data lies between `24` and `33`

3.1.1 D Education

In [43]:
```
plt.figure(figsize = (15,5))

plt.subplot(1,3,1)
plt.xlabel("Education")
plt.ylabel("Count")
plt.title("Histplot on Education")
#plt.xticks(rotation = 90)
sns.histplot(data['Education'])

plt.subplot(1,3,2)
plt.xlabel("Education")
plt.ylabel("Count")
plt.title("Distplot on Education")
#plt.xticks(rotation = 90)
sns.distplot(data['Education'])

plt.subplot(1,3,3)
plt.xlabel("Education")
plt.ylabel("Count")
plt.title("Boxplot on Education")
#plt.xticks(rotation = 90)
sns.boxplot(data = data, x = 'Education')

plt.show()
```

```
/tmp/ipykernel_1681/66102555.py:15: UserWarning:

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

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

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

  sns.distplot(data['Education'])
```



In [44]:
```
print("Mean of Education Column: ", np.mean(data['Education']))
print("Median of Education Column: ", np.median(data['Education']))
```

```
Mean of Education Column:  15.572222222222223
Median of Education Column:  16.0
```

Computing Quantiles, Lowerbound and Upperbound

In [45]:
```
education_q1 = np.percentile(data['Education'], 25)
education_q3 = np.percentile(data['Education'], 75)
education_iqr = education_q3 - education_q1
education_upperbound = education_q3 + 1.5 * education_iqr
education_lowerbound = education_q1 - 1.5 * education_iqr
if education_lowerbound < 0:
    education_lowerbound = 0
```

In [46]:
```
print("First Quantile of Education Column: ", education_q1)
print("Second Quantile (Median) of Education Column: ", np.percentile(data['Education'], 50))
print("Third Quantile of Education Column: ", education_q3)
print("IQR of Education Column: ",education_iqr)
print("Lowerbound of Education Column: ", education_lowerbound)
print("Upperbound of Education Column: ", education_upperbound)
```

```
First Quantile of Education Column:  14.0
Second Quantile (Median) of Education Column:  16.0
Third Quantile of Education Column:  16.0
IQR of Education Column:  2.0
Lowerbound of Education Column:  11.0
Upperbound of Education Column:  19.0
```

In [47]:
```
# Outliers in Education Field
data_education_outliers = data.loc[data['Education'] > education_upperbound]
data_education_outliers.head()
```

Out[47]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
156	KP781	25	Male	20	Partnered	4	5	74701	170
157	KP781	26	Female	21	Single	4	3	69721	100
161	KP781	27	Male	21	Partnered	4	4	90886	100
175	KP781	40	Male	21	Single	6	5	83416	200

In [48]:
```
# % of outliers
print("Number of outliers in Education field: ", len(data_education_outliers))
print("% of outliers in Education Column: ",np.round(len(data_education_outliers)/ len(data['Education']
```

```
Number of outliers in Education field:  4
% of outliers in Education Column:  0.0222
```

In [49]:
```
len(data.loc[(data['Education'] >= education_q1) & (data['Education'] <= education_q3)])/len(data['Educa
```

Out[49]:
```
0.8055555555555556
```

Insights from Education Column

1. Mean education is `15.7` whereas Median is `16`. Mean and Median are almost same.
2. Lower bound of education is `11` and Upper bound is `19`.
3. There are `4` outliers in educcation column and this is around `2%` of data.
4. There are around `~80%` of data lies between `14` and `16`

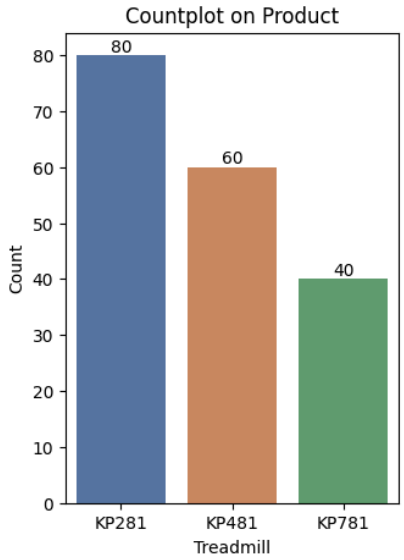
3.1.2 Plotting for Categorical Variables

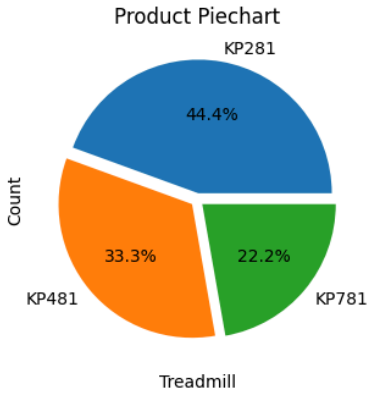
3.1.2 A Product

In [50]:
```
plt.figure(figsize = (12,5))


plt.subplot(1,3,1)
plt.xlabel("Treadmill")
plt.ylabel("Count")
plt.title("Countplot on Product")
#plt.xticks(rotation = 90)
ax = sns.countplot(data = data, x = 'Product', palette = 'deep')
for container in ax.containers:
```

```
    ax.bar_label(container)

plt.subplot(1,3,3)
plt.xlabel("Treadmill")
plt.ylabel("Count")
plt.title("Product Piechart")
#plt.xticks(rotation = 90)
plt.pie(data['Product'].value_counts(),labels = data['Product'].unique(), autopct = "%2.1f%%", explode =

plt.show()
```

```
/tmp/ipykernel_1681/897982621.py:9: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` va
riable to `hue` and set `legend=False` for the same effect.

  ax = sns.countplot(data = data, x = 'Product', palette = 'deep')
```





In [51]:
```
kp281 = data.loc[data['Product'] == 'KP281']
kp481 = data.loc[data['Product'] == 'KP481']
kp781 = data.loc[data['Product'] == 'KP781']
kp281_revenue = len(kp281) * 1500
kp481_revenue = len(kp481) * 1750
kp781_revenue = len(kp781) * 2500
print("=" * 50)
print("# of KP281 sold: ", len(kp281))
print("# of KP481 sold: ", len(kp481))
print("# of KP781 sold: ", len(kp781))
print("=" * 50)
print("Overall Revenue generated from KP281:", kp281_revenue)
print("Overall Revenue generated from KP481:", kp481_revenue)
print("Overall Revenue generated from KP781:", kp781_revenue)
```

```
==================================================
# of KP281 sold:  80
# of KP481 sold:  60
# of KP781 sold:  40
==================================================
Overall Revenue generated from KP281: 120000
Overall Revenue generated from KP481: 105000
Overall Revenue generated from KP781: 100000
```

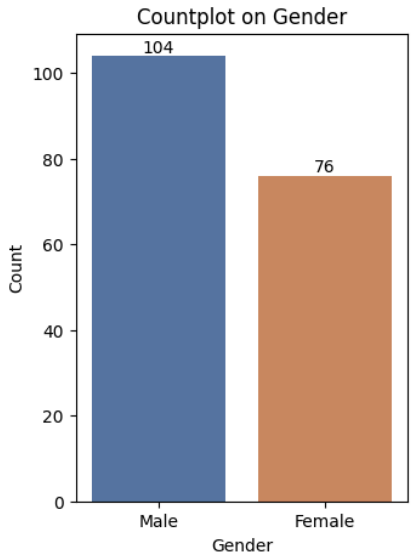
Insights on Product type

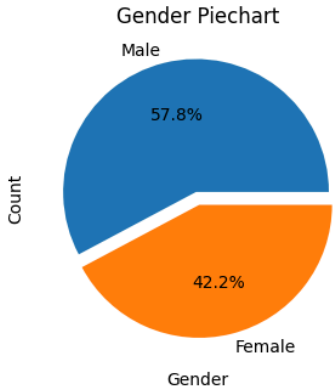
1. Among 3 treadmills, KP281 is more popular with `44.4%` share. Next is KP481 with `33.3%` and lastly KP781 with `22.2%` share
2. Revenue generated by KP281 is `$120000` and KP481 is `$105000` and KP781 is `$100000`

3.1.2 B Gender

In [52]:
```
plt.figure(figsize = (12,5))


plt.subplot(1,3,1)
plt.xlabel("Gender")
plt.ylabel("Count")
plt.title("Countplot on Gender")
#plt.xticks(rotation = 90)
ax = sns.countplot(data = data, x = 'Gender', palette = 'deep')
for container in ax.containers:
    ax.bar_label(container)

plt.subplot(1,3,3)
plt.xlabel("Gender")
plt.ylabel("Count")
plt.title("Gender Piechart")
#plt.xticks(rotation = 90)
plt.pie(data['Gender'].value_counts(),labels =data['Gender'].value_counts().index, autopct = "%2.1f%%",

plt.show()
```

```
/tmp/ipykernel_1681/443268861.py:9: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` va
riable to `hue` and set `legend=False` for the same effect.

  ax = sns.countplot(data = data, x = 'Gender', palette = 'deep')
```





In [53]:
```
male = data.loc[data['Gender'] == 'Male']
female = data.loc[data['Gender'] == 'Female']
print("=" * 50)
print("# of Male: ", len(male))
print("# of Female: ", len(female))
```

```
==================================================
# of Male:  104
# of Female:  76
```

Insights on Gender

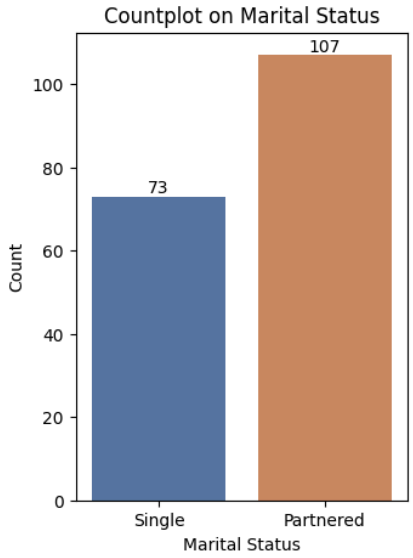
1. In the dataset, Male customers are `57.8%` and Female customers are `42.3%`

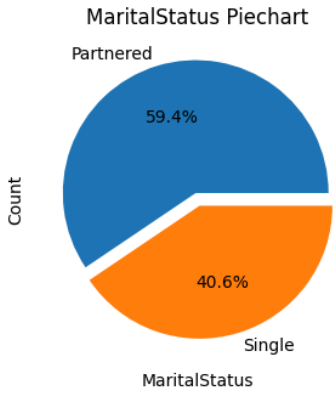
3.1.2 C Marital Status

In [54]:
```
plt.figure(figsize = (12,5))


plt.subplot(1,3,1)
plt.xlabel("Marital Status")
plt.ylabel("Count")
```

```
plt.title("Countplot on Marital Status")
#plt.xticks(rotation = 90)
ax = sns.countplot(data = data, x = 'MaritalStatus', palette = 'deep')
for container in ax.containers:
    ax.bar_label(container)

plt.subplot(1,3,3)
plt.xlabel("MaritalStatus")
plt.ylabel("Count")
plt.title("MaritalStatus Piechart")
#plt.xticks(rotation = 90)
plt.pie(data['MaritalStatus'].value_counts(),labels = data['MaritalStatus'].value_counts().index, autopc

plt.show()
```

```
/tmp/ipykernel_1681/2405296796.py:9: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` va
riable to `hue` and set `legend=False` for the same effect.

  ax = sns.countplot(data = data, x = 'MaritalStatus', palette = 'deep')
```





In [55]:
```
single = data.loc[data['MaritalStatus'] == 'Single']
partnered = data.loc[data['MaritalStatus'] == 'Partnered']
print("=" * 50)
print("# of Single: ", len(single))
print("# of Partnered: ", len(partnered))
```

```
==================================================
# of Single:  73
# of Partnered:  107
```

Insights on Marital Status

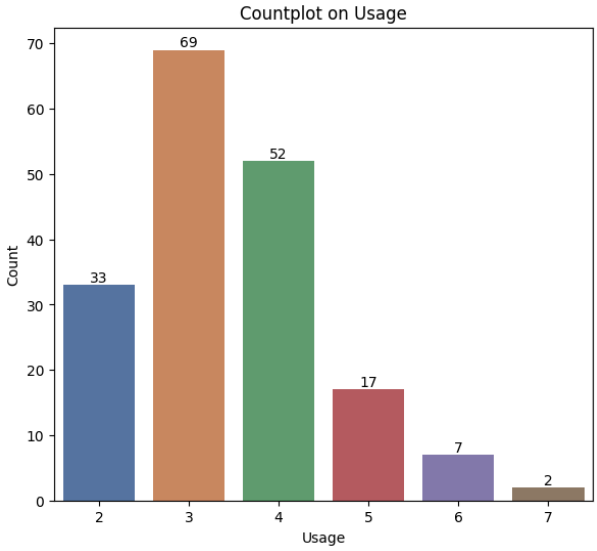
1. In the dataset, Single customers are `40.6%` and Partnered customers are `59.4%`

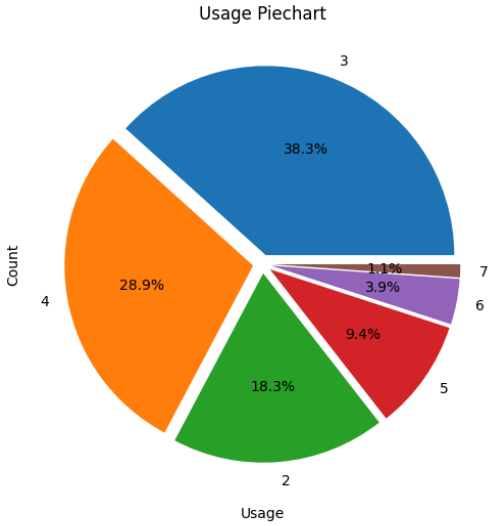
3.1.2 D Usage

In [56]:
```
plt.figure(figsize = (15,6))


plt.subplot(1,2,1)
plt.xlabel("Usage")
plt.ylabel("Count")
plt.title("Countplot on Usage")
#plt.xticks(rotation = 90)
ax = sns.countplot(data = data, x = 'Usage', palette = 'deep')
for container in ax.containers:
    ax.bar_label(container)

plt.subplot(1,2,2)
```

```
plt.xlabel("Usage")
plt.ylabel("Count")
plt.title("Usage Piechart")
#plt.xticks(rotation = 90)
plt.pie(data['Usage'].value_counts(),labels = data['Usage'].value_counts().index, autopct = "%2.1f%%", e

plt.show()
```

```
/tmp/ipykernel_1681/4196010939.py:9: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` va
riable to `hue` and set `legend=False` for the same effect.

  ax = sns.countplot(data = data, x = 'Usage', palette = 'deep')
```





Insights on Usage

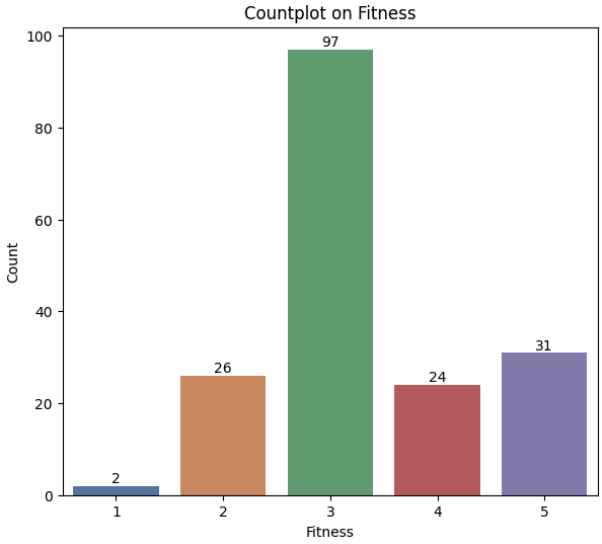
1. About `85%` of customers planning to use the treadmill atleast 2 to 4 times in a week and `15%` users are planning to use treadmill more than 4 times in a week

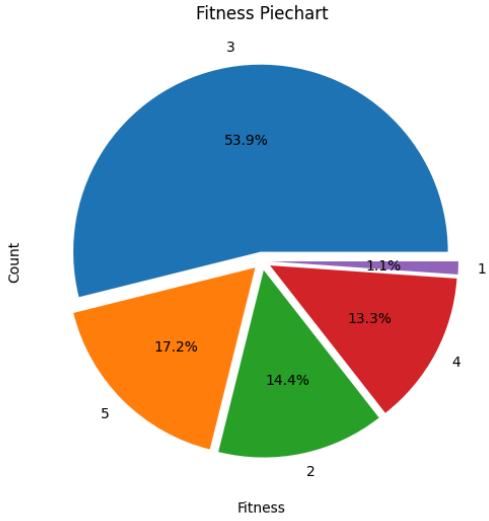
---

3.1.2 E Fitness

In [57]:
```
plt.figure(figsize = (15,6))


plt.subplot(1,2,1)
plt.xlabel("Fitness")
plt.ylabel("Count")
plt.title("Countplot on Fitness")
#plt.xticks(rotation = 90)
ax = sns.countplot(data = data, x = 'Fitness', palette = 'deep')
for container in ax.containers:
    ax.bar_label(container)

plt.subplot(1,2,2)
plt.xlabel("Fitness")
plt.ylabel("Count")
plt.title("Fitness Piechart")
#plt.xticks(rotation = 90)
plt.pie(data['Fitness'].value_counts(),labels = data['Fitness'].value_counts().index, autopct = "%2.1f%%

plt.show()
```

```
/tmp/ipykernel_1681/3948373169.py:9: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` va
riable to `hue` and set `legend=False` for the same effect.

  ax = sns.countplot(data = data, x = 'Fitness', palette = 'deep')
```





Insights on Fitness

1. About `54%` of users rated themselves as medium in the fitness and `1%` rated themselves as poor and `17%` as Excellent

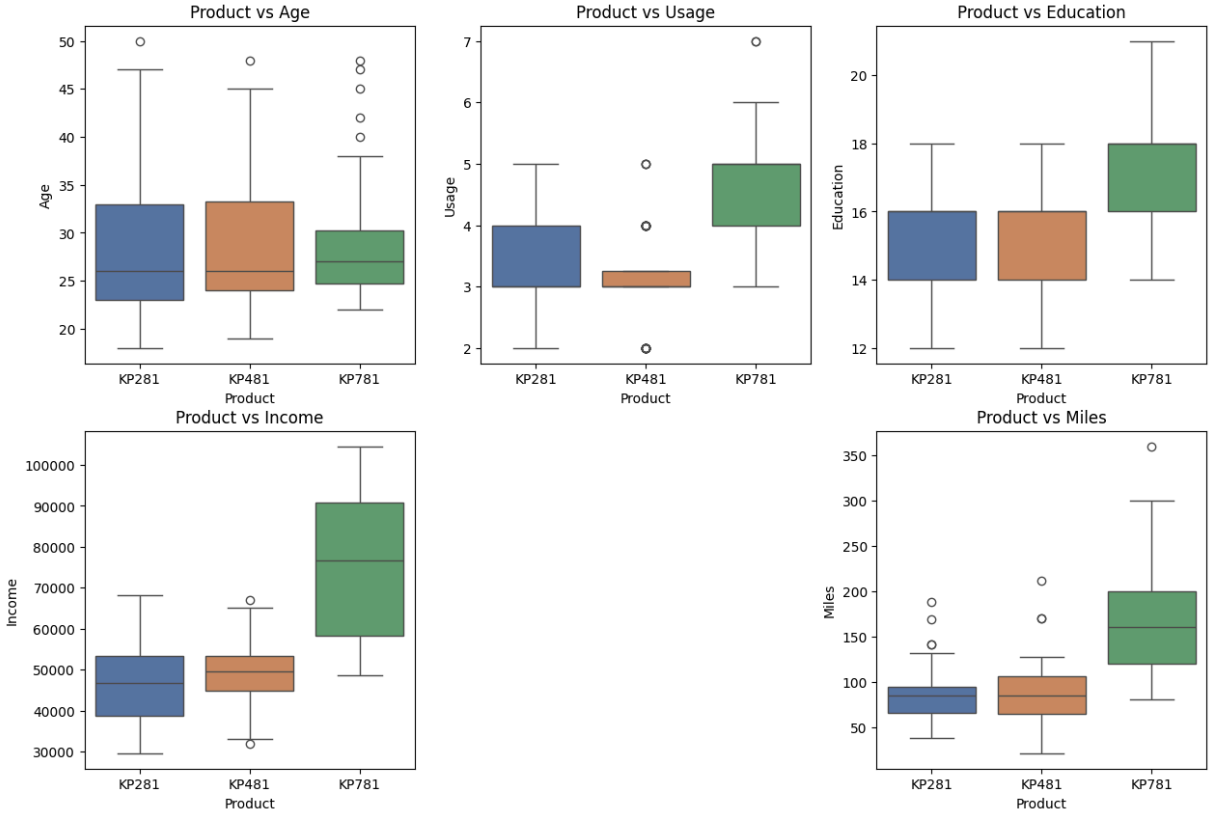
---

3.2 Bivariate Analysis

3.2.1 Product vs Age vs Usage vs Education vs Income vs Miles

In [58]:
```
plt.figure(figsize = (15,10))


plt.subplot(2,3,1)
plt.xlabel("Product")
plt.ylabel("Age")
plt.title("Product vs Age")
#plt.xticks(rotation = 90)
sns.boxplot(data = data, x = 'Product', y = 'Age',palette = 'deep')

plt.subplot(2,3,2)
plt.xlabel("Product")
plt.ylabel("Usage")
plt.title("Product vs Usage")
#plt.xticks(rotation = 90)
sns.boxplot(data = data, x = 'Product', y = 'Usage',palette = 'deep')

plt.subplot(2,3,3)
plt.xlabel("Product")
plt.ylabel("Education")
plt.title("Product vs Education")
#plt.xticks(rotation = 90)
sns.boxplot(data = data, x = 'Product', y = 'Education',palette = 'deep')

plt.subplot(2,3,4)
plt.xlabel("Product")
plt.ylabel("Income")
plt.title("Product vs Income")
#plt.xticks(rotation = 90)
sns.boxplot(data = data, x = 'Product', y = 'Income',palette = 'deep')


plt.subplot(2,3,6)
plt.xlabel("Product")
plt.ylabel("Miles")
plt.title("Product vs Miles")
#plt.xticks(rotation = 90)
sns.boxplot(data = data, x = 'Product', y = 'Miles',palette = 'deep')

plt.show()
```

```
/tmp/ipykernel_1681/1416416354.py:9: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` va
riable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(data = data, x = 'Product', y = 'Age',palette = 'deep')
/tmp/ipykernel_1681/1416416354.py:16: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` va
riable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(data = data, x = 'Product', y = 'Usage',palette = 'deep')
/tmp/ipykernel_1681/1416416354.py:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` va
riable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(data = data, x = 'Product', y = 'Education',palette = 'deep')
/tmp/ipykernel_1681/1416416354.py:30: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` va
riable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(data = data, x = 'Product', y = 'Income',palette = 'deep')
/tmp/ipykernel_1681/1416416354.py:38: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` va
riable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(data = data, x = 'Product', y = 'Miles',palette = 'deep')
```

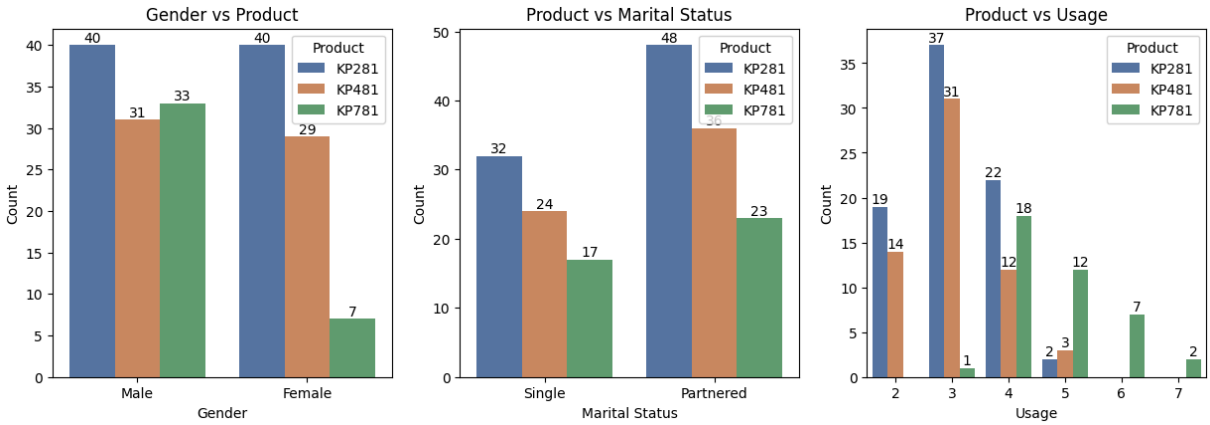


Insights on Product vs Age vs Usage vs Education vs Income vs Miles

```
KP781 is famous treadmill for :
- Age group between 25 to 30 years
- Customer usage is between 4 to 5 times a week
- Education is between 16 years to 18 years
- Income is above Rs:-60000
- Customer is planning for intence workouts - 125 miles per week to 200 miles per week
```

3.2.2 Product vs Gender vs Marital Status vs Usage

In [59]:
```
plt.figure(figsize = (15,10))


plt.subplot(2,3,1)
plt.xlabel("Gender")
plt.ylabel("Count")
plt.title("Gender vs Product")
#plt.xticks(rotation = 90)
ax = sns.countplot(data = data, x = 'Gender', hue = 'Product',palette = 'deep')
for container in ax.containers:
    ax.bar_label(container)

plt.subplot(2,3,2)
plt.xlabel("Marital Status")
plt.ylabel("Count")
plt.title("Product vs Marital Status")
#plt.xticks(rotation = 90)
ax = sns.countplot(data = data, x = 'MaritalStatus', hue = 'Product',palette = 'deep')
for container in ax.containers:
    ax.bar_label(container)

plt.subplot(2,3,3)
plt.xlabel("Usage")
plt.ylabel("Count")
plt.title("Product vs Usage")
#plt.xticks(rotation = 90)
ax = sns.countplot(data = data, x = 'Usage', hue = 'Product',palette = 'deep')
for container in ax.containers:
    ax.bar_label(container)

plt.show()
```



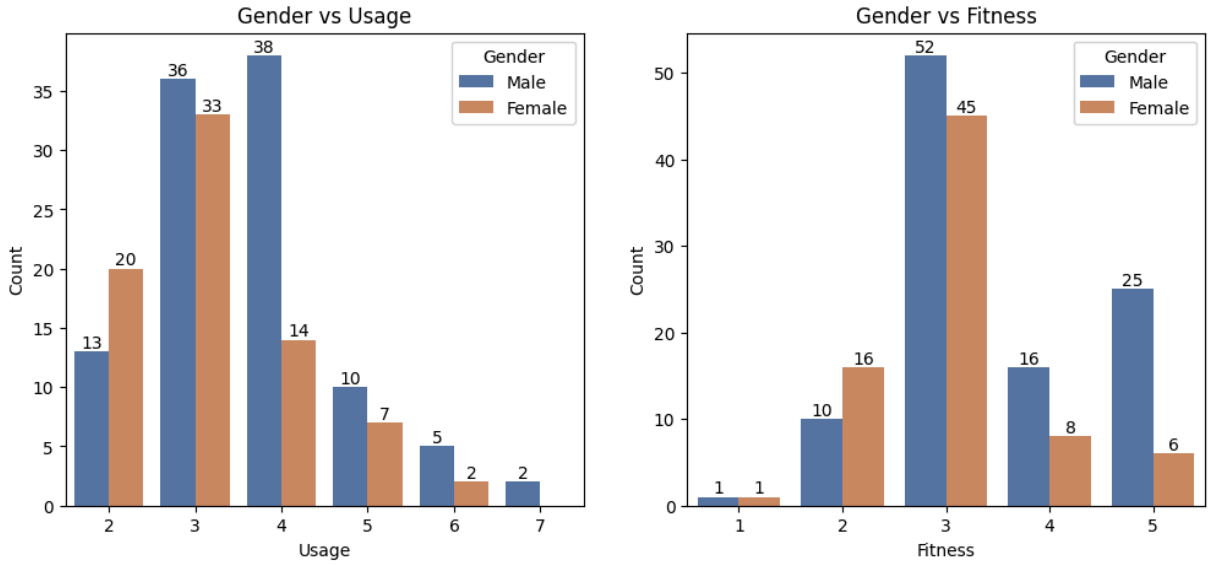
Insights on Product vs Gender vs Marital Status vs Usage

1. Product KP281 is famous among:
   - Male and Female
   - Single and Partnered
   - Usage upto 4 times in a week
2. Product KP781 is famous among:
   - Users who want to use treadmill more than 4 times in a week
3. Partnered customers are more likely to buy any treadmill compared to single customers

3.2.3 Gender vs Usage vs Fitness

In [60]:
```
plt.figure(figsize = (12,5))


plt.subplot(1,2,1)
plt.xlabel("Usage")
plt.ylabel("Count")
plt.title("Gender vs Usage")
#plt.xticks(rotation = 90)
ax = sns.countplot(data = data, x = 'Usage', hue = 'Gender',palette = 'deep')
for container in ax.containers:
    ax.bar_label(container)
```

```
plt.subplot(1,2,2)
plt.xlabel("Fitness")
plt.ylabel("Count")
plt.title("Gender vs Fitness")
#plt.xticks(rotation = 90)
ax = sns.countplot(data = data, x = 'Fitness', hue = 'Gender',palette = 'deep')
for container in ax.containers:
    ax.bar_label(container)


plt.show()
```

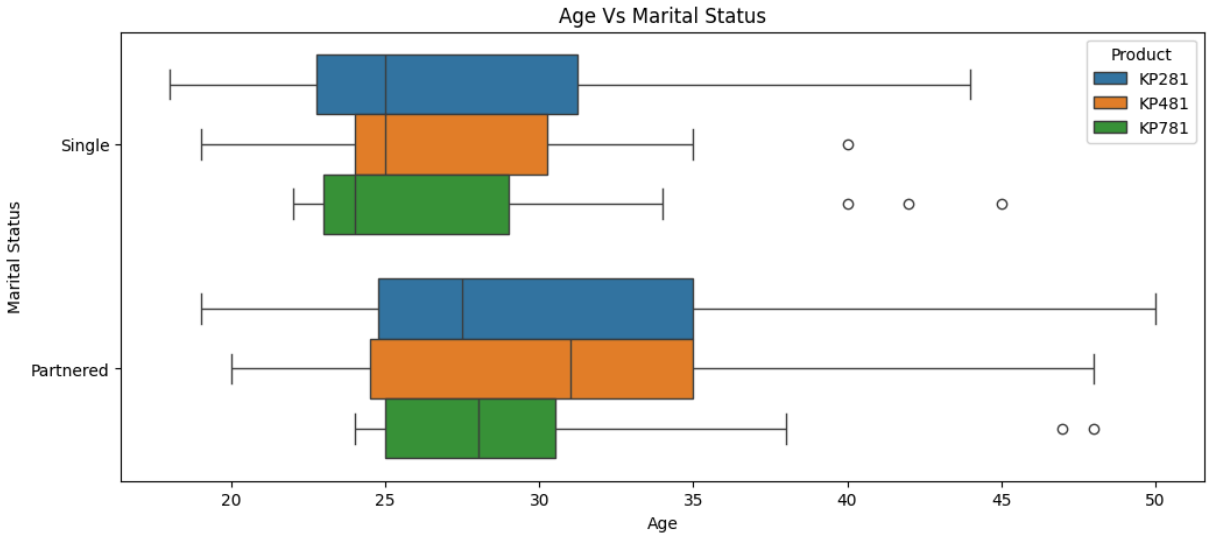


Insights on Gender vs Usage vs Fitness

1. Female users are more predominent for 2 times a week usage whereas male is predominent for rest of the usage
2. ~80% of Female users plan to use treadmill for 2 to 4 times a week and 70% of male users plan to use for 3 to 5 times a week
3. Except for fitness rating of 2, Male users are predominent for rest of the ratings
4. ~80% of Female users rated themselves between 2 to 3 whereas 90% Male users rated themselves between 3 to 5

3.2.4 Age vs MaritalStatus vs Product

In [61]:
```
plt.figure(figsize = (12,5))
plt.subplot(1,1,1)
plt.xlabel("Age")
plt.ylabel("Marital Status")
plt.title("Age Vs Marital Status")
#plt.xticks(rotation = 90)
sns.boxplot(data = data, x = 'Age', y = 'MaritalStatus', hue = 'Product')
plt.show()
```

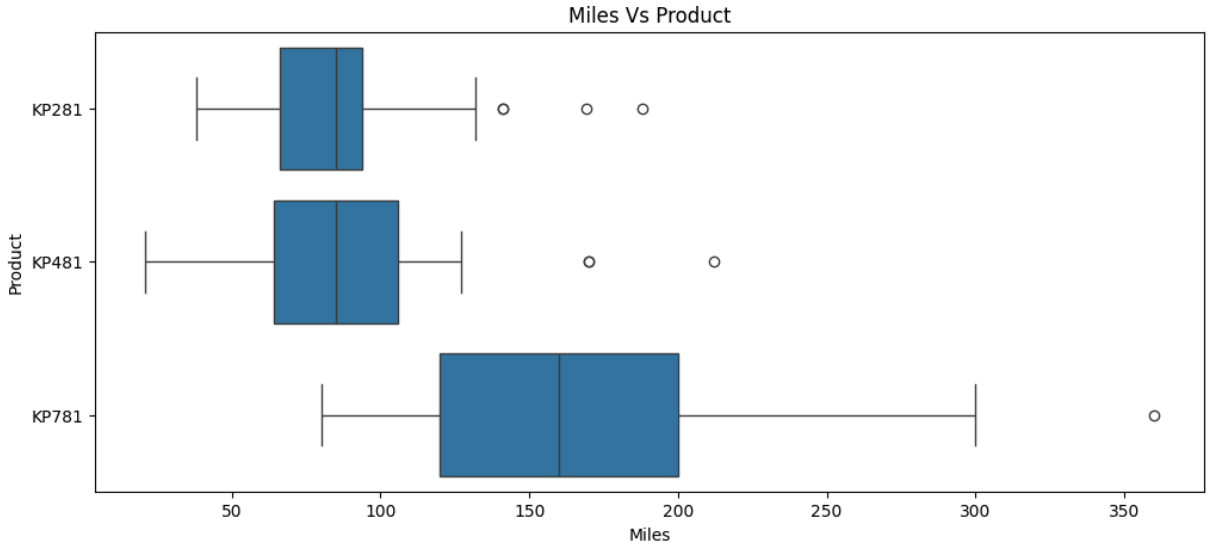


Insights on Age vs MaritalStatus vs Product

1. Median Age of single users who are using treadmill is less than Median Age of Partnered users
2. KP281 is more famous in both Single and Partnered users
3. Variability of KP281 is more compared to other two treadmills in both Single and Partnered users
4. Median Age of KP781 for single users is less than all other categories
5. Median Age of KP481 is higher compared to all other categories
6. Q1 to Q3 of KP281 and KP481 in partnered users are very similar

3.2.4 Miles vs Product

In [62]:
```
plt.figure(figsize = (12,5))
plt.subplot(1,1,1)
plt.xlabel("Miles")
plt.ylabel("Product")
plt.title("Miles Vs Product")
#plt.xticks(rotation = 90)
sns.boxplot(data = data, x = 'Miles', y = 'Product')
plt.show()
```

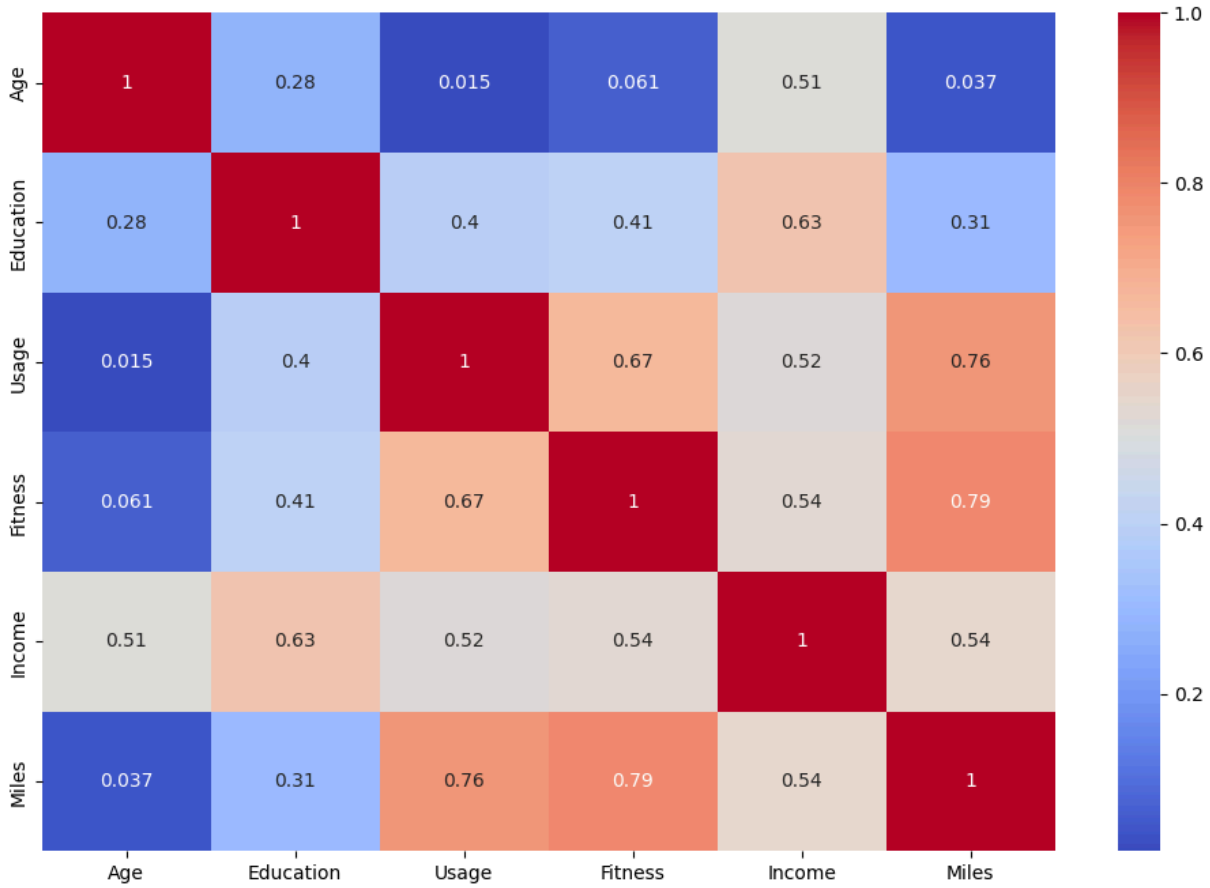


Insights:

1. KP281 is considered by customers who are planning to light workout with most users between 66 Miles to 94 Miles with mean of ~83 Miles
2. KP481 is considered by customers who are planning for medium workout with most users between 64 Miles to 106 Miles with mean of 88 Miles
3. KP781 is considered by customers who are planning for intence workout with most users between 120 Miles to 200 Miles with mean of 167 Miles

3.3 Heatmaps

In [63]:
```
data_copy = data[['Age','Education','Usage','Fitness','Income','Miles']].corr()
plt.figure(figsize=(12,8))
sns.heatmap(data_copy,annot = True, cmap="coolwarm")
plt.show()
```



Insights on Correlation:

1. Income and Education are correlated as higher education has chance of higher Income
2. Usage is highly correlated to Fitness rating and planned Miles
3. We can also see Usage and Age is very weakly correlated
4. Similarly Fitness and age are weakly correlated
5. Income is almost equally correlated to rest of the columns

4 Marginal Probability

4.1 Probability of Product and Gender

In [64]:
```
pd.crosstab(index = data['Product'], columns = data['Gender'], margins = True)
```

Out[64]:

Product	Female	Male	All
KP281	40	40	80
KP481	29	31	60
KP781	7	33	40
All	76	104	180

In [65]:
```
def prob_gender_product(data, gender):
    """Function to compute conditional probability of gender given product"""
    total = len(data)
```

```
    products = data['Product'].unique()
    condition_prob = {}
    for product in products:
        # Computing the length of Products
        product_total = len(data.loc[data['Product'] == product])
        # Computing the total of product and gender
        product_gender_total = len(data.loc[(data['Product'] == product) & (data['Gender'] == gender)])
        # Computing the probability of gender given product
        prob_gender_product = product_gender_total/product_total
        condition_prob[product] = prob_gender_product

    for product, prob in condition_prob.items():
        gender = gender
        print(f"Probability of {gender} customer buying {product} treadmill -> P({gender}|{product}): {n
```

In [66]:
```
def prob_product_gender(data, product):
    """Function to compute conditional probability of product given gender"""
    total = len(data)
    genders = data['Gender'].unique()
    condition_prob = {}
    for gender in genders:
        # Computing the length of gender
        gender_total = len(data.loc[data['Gender'] == gender])
        # Computing the total of product and gender
        product_gender_total = len(data.loc[(data['Product'] == product) & (data['Gender'] == gender)])
        # Computing the probability of gender given product
        prob_product_gender = product_gender_total/gender_total
        condition_prob[gender] = prob_product_gender

    for gender, prob in condition_prob.items():
        print(f"Probability of {product} given {gender} -> P({product}|{gender}): {np.round(prob,2)}")
```

In [67]:
```
for gender in data['Gender'].unique():
    prob_gender_product(data, gender)
```

```
Probability of Male customer buying KP281 treadmill -> P(Male|KP281): 0.5
Probability of Male customer buying KP481 treadmill -> P(Male|KP481): 0.52
Probability of Male customer buying KP781 treadmill -> P(Male|KP781): 0.82
Probability of Female customer buying KP281 treadmill -> P(Female|KP281): 0.5
Probability of Female customer buying KP481 treadmill -> P(Female|KP481): 0.48
Probability of Female customer buying KP781 treadmill -> P(Female|KP781): 0.18
```

In [68]:
```
for product in data['Product'].unique():
    prob_product_gender(data, product)
```

```
Probability of KP281 given Male -> P(KP281|Male): 0.38
Probability of KP281 given Female -> P(KP281|Female): 0.53
Probability of KP481 given Male -> P(KP481|Male): 0.3
Probability of KP481 given Female -> P(KP481|Female): 0.38
Probability of KP781 given Male -> P(KP781|Male): 0.32
Probability of KP781 given Female -> P(KP781|Female): 0.09
```

Insights on Probability of Product and Gender

> **1. Marginal Probabilities**:

- Probability of KP281 -> P(KP281) = **44.4%**
- Probability of KP481 -> P(KP481) = **33.3%**
- Probability of KP781 -> P(KP781) = **22.2%**
- Probability of Male -> P(Male) = **57.8%**
- Probability of Female -> P(Female) = **42.2%**

> **2. Condiitional Probabilities**:

- Probability of Male customer buying KP281 treadmill -> P(Male|KP281): **50%**
- Probability of Male customer buying KP481 treadmill -> P(Male|KP481): **52%**
- Probability of Male customer buying KP781 treadmill -> P(Male|KP781): **82%**
- Probability of Female customer buying KP281 treadmill -> P(Female|KP281): **50%**
- Probability of Female customer buying KP481 treadmill -> P(Female|KP481): **48%**

- Probability of Female customer buying KP781 treadmill -> P(Female|KP781): **18%**
- Probability of KP281 given Male -> P(KP281|Male): **38%**
- Probability of KP281 given Female -> P(KP281|Female): **53%**
- Probability of KP481 given Male -> P(KP481|Male): **30%**
- Probability of KP481 given Female -> P(KP481|Female): **38%**
- Probability of KP781 given Male -> P(KP781|Male): **32%**
- Probability of KP781 given Female -> P(KP781|Female): **9%**

4.2 Probability of Product and Marital Status

In [69]:
```
pd.crosstab(index = data['Product'], columns = data['MaritalStatus'], margins = True).round(3)
```

Out[69]:

MaritalStatus	Partnered	Single	All
Product			
KP281	48	32	80
KP481	36	24	60
KP781	23	17	40
All	107	73	180

In [70]:
```
def prob_maritalstatus_product(data, maritalstatus):
    """Function to compute conditional probability of marital status given product"""
    total = len(data)
    products = data['Product'].unique()
    condition_prob = {}
    for product in products:
        # Computing the length of Products
        product_total = len(data.loc[data['Product'] == product])
        # Computing the total of product and maritalstatus
        product_maritalstatus_total = len(data.loc[(data['Product'] == product) & (data['MaritalStatus']
        # Computing the probability of gender given product
        prob_maritalstatus_product = product_maritalstatus_total/product_total
        condition_prob[product] = prob_maritalstatus_product

    for product, prob in condition_prob.items():
        print(f"Probability of {maritalstatus} customer buying {product} treadmill -> P({marital}|{produ
```

In [71]:
```
def prob_product_maritalstatus(data, product):
    """Function to compute conditional probability of product given marital status"""
    total = len(data)
    maritalstatus = data['MaritalStatus'].unique()
    condition_prob = {}
    for marital in maritalstatus:
        # Computing the length of marital status
        marital_total = len(data.loc[data['MaritalStatus'] == marital])
        # Computing the total of product and marital status
        product_marital_total = len(data.loc[(data['Product'] == product) & (data['MaritalStatus'] == ma
        # Computing the probability of gender given product
        prob_product_marital = product_marital_total/marital_total
        condition_prob[marital] = prob_product_marital

    for marital, prob in condition_prob.items():
        print(f"Probability of {product} given {marital} -> P({product}|{marital}): {np.round(prob,2)}")
```

In [72]:
```
for marital in data['MaritalStatus'].unique():
    prob_maritalstatus_product(data, marital)
```

```
Probability of Single customer buying KP281 treadmill -> P(Single|KP281): 0.4
Probability of Single customer buying KP481 treadmill -> P(Single|KP481): 0.4
Probability of Single customer buying KP781 treadmill -> P(Single|KP781): 0.42
Probability of Partnered customer buying KP281 treadmill -> P(Partnered|KP281): 0.6
Probability of Partnered customer buying KP481 treadmill -> P(Partnered|KP481): 0.6
Probability of Partnered customer buying KP781 treadmill -> P(Partnered|KP781): 0.57
```

In [73]:
```
for product in data['Product'].unique():
    prob_product_maritalstatus(data, product)
```

```
Probability of KP281 given Single -> P(KP281|Single): 0.44
Probability of KP281 given Partnered -> P(KP281|Partnered): 0.45
Probability of KP481 given Single -> P(KP481|Single): 0.33
Probability of KP481 given Partnered -> P(KP481|Partnered): 0.34
Probability of KP781 given Single -> P(KP781|Single): 0.23
Probability of KP781 given Partnered -> P(KP781|Partnered): 0.21
```

In [ ]:

Insights on Probability of Product and Marital Status

> **1. Marginal Probabilities**:

- Probability of KP281 -> P(KP281) = **44.4%**
- Probability of KP481 -> P(KP481) = **33.3%**
- Probability of KP781 -> P(KP781) = **22.2%**
- Probability of Partnered -> P(Partnered) = **59.4%**
- Probability of Single -> P(Single) = **40.6%**

> **2. Condiitional Probabilities**:

- Probability of Single customer buying KP281 treadmill -> P(Single|KP281): **40%**
- Probability of Single customer buying KP481 treadmill -> P(Single|KP481): **40%**
- Probability of Single customer buying KP781 treadmill -> P(Single|KP781): **42%**
- Probability of Partnered customer buying KP281 treadmill -> P(Partnered|KP281): **60%**
- Probability of Partnered customer buying KP481 treadmill -> P(Partnered|KP481): **60%**
- Probability of Partnered customer buying KP781 treadmill -> P(Partnered|KP781): **58%**
- Probability of KP281 given Single -> P(KP281|Single): **44%**
- Probability of KP281 given Partnered -> P(KP281|Partnered): **45%**
- Probability of KP481 given Single -> P(KP481|Single): **33%**
- Probability of KP481 given Partnered -> P(KP481|Partnered): **34%**
- Probability of KP781 given Single -> P(KP781|Single): **23%**
- Probability of KP781 given Partnered -> P(KP781|Partnered): **21%**

4.3 Probability of Product and Age Group

In [74]:
```
data['Age'].describe()
```

Out[74]:
```
count    180.000000
mean      28.788889
std        6.943498
min       18.000000
25%       24.000000
50%       26.000000
75%       33.000000
max       50.000000
Name: Age, dtype: float64
```

In [75]:
```
age_bins = [17,30,40,51]
age_labels = ['Vicenarian','Tricenarian','Quadragenarian']
data['Age_Group'] = pd.cut(data['Age'], bins = age_bins, labels = age_labels)
```

In [76]:
```
pd.crosstab(index = data['Product'], columns = data['Age_Group'], margins = True).round(3)
```

Out[76]:

Age_Group	Vicenarian	Tricenarian	Quadragenarian	All
Product				
KP281	55	19	6	80
KP481	35	23	2	60
KP781	30	6	4	40
All	120	48	12	180

In [77]:
```
def prob_agegroup_product(data, agegroup):
    """Function to compute conditional probability of marital status given product"""
    total = len(data)
    products = data['Product'].unique()
    condition_prob = {}
    for product in products:
        # Computing the length of Products
        product_total = len(data.loc[data['Product'] == product])
        # Computing the total of product and maritalstatus
        product_age_total = len(data.loc[(data['Product'] == product) & (data['Age_Group'] == agegroup)])
        # Computing the probability of gender given product
        prob_age_product = product_age_total/product_total
        condition_prob[product] = prob_age_product

    for product, prob in condition_prob.items():
        print(f"Probability of {agegroup} customer buying {product} treadmill -> P({agegroup}|{product})
```

In [78]:
```
def prob_product_agegroup(data, product):
    """Function to compute conditional probability of product given marital status"""
    total = len(data)
    agegroup = data['Age_Group'].unique()
    condition_prob = {}
    for age in agegroup:
        # Computing the length of marital status
        age_total = len(data.loc[data['Age_Group'] == age])
        # Computing the total of product and marital status
        product_age_total = len(data.loc[(data['Product'] == product) & (data['Age_Group'] == age)])
        # Computing the probability of gender given product
        prob_product_age = product_age_total/age_total
        condition_prob[age] = prob_product_age

    for age, prob in condition_prob.items():
        print(f"Probability of {product} given {age} -> P({product}|{age}): {np.round(prob,2)}")
```

In [79]:
```
for age in data['Age_Group'].unique():
    prob_agegroup_product(data, age)
```

```
Probability of Vicenarian customer buying KP281 treadmill -> P(Vicenarian|KP281): 0.69
Probability of Vicenarian customer buying KP481 treadmill -> P(Vicenarian|KP481): 0.58
Probability of Vicenarian customer buying KP781 treadmill -> P(Vicenarian|KP781): 0.75
Probability of Tricenarian customer buying KP281 treadmill -> P(Tricenarian|KP281): 0.24
Probability of Tricenarian customer buying KP481 treadmill -> P(Tricenarian|KP481): 0.38
Probability of Tricenarian customer buying KP781 treadmill -> P(Tricenarian|KP781): 0.15
Probability of Quadragenarian customer buying KP281 treadmill -> P(Quadragenarian|KP281): 0.08
Probability of Quadragenarian customer buying KP481 treadmill -> P(Quadragenarian|KP481): 0.03
Probability of Quadragenarian customer buying KP781 treadmill -> P(Quadragenarian|KP781): 0.1
```

In [80]:
```
for product in data['Product'].unique():
    prob_product_agegroup(data, product)
```

```
Probability of KP281 given Vicenarian -> P(KP281|Vicenarian): 0.46
Probability of KP281 given Tricenarian -> P(KP281|Tricenarian): 0.4
Probability of KP281 given Quadragenarian -> P(KP281|Quadragenarian): 0.5
Probability of KP481 given Vicenarian -> P(KP481|Vicenarian): 0.29
Probability of KP481 given Tricenarian -> P(KP481|Tricenarian): 0.48
Probability of KP481 given Quadragenarian -> P(KP481|Quadragenarian): 0.17
Probability of KP781 given Vicenarian -> P(KP781|Vicenarian): 0.25
Probability of KP781 given Tricenarian -> P(KP781|Tricenarian): 0.12
Probability of KP781 given Quadragenarian -> P(KP781|Quadragenarian): 0.33
```

Insights on Probability of Product and Gender

> **1. Marginal Probabilities**:

- Probability of KP281 -> P(KP281) = **44.4%**
- Probability of KP481 -> P(KP481) = **33.3%**
- Probability of KP781 -> P(KP781) = **22.2%**
- Probability of Vicenarian -> P(Vicenarian) = **66.7%**
- Probability of Tricenarian -> P(Tricenarian) = **26.7%**
- Probability of Quadragenarian -> P(Quadragenarian) = **6.7%**

> **2. Condiitional Probabilities**:

- Probability of Vicenarian customer buying KP281 treadmill -> P(Vicenarian|KP281): **69%**
- Probability of Vicenarian customer buying KP481 treadmill -> P(Vicenarian|KP481): **58%**
- Probability of Vicenarian customer buying KP781 treadmill -> P(Vicenarian|KP781): **75%**
- Probability of Tricenarian customer buying KP281 treadmill -> P(Tricenarian|KP281): **24%**
- Probability of Tricenarian customer buying KP481 treadmill -> P(Tricenarian|KP481): **38%**
- Probability of Tricenarian customer buying KP781 treadmill -> P(Tricenarian|KP781): **15%**
- Probability of Quadragenarian customer buying KP281 treadmill -> P(Quadragenarian|KP281): **8%**
- Probability of Quadragenarian customer buying KP481 treadmill -> P(Quadragenarian|KP481): **3%**
- Probability of Quadragenarian customer buying KP781 treadmill -> P(Quadragenarian|KP781): **1%**
- Probability of KP281 given Vicenarian -> P(KP281|Vicenarian): **46%**
- Probability of KP281 given Tricenarian -> P(KP281|Tricenarian): **40%**
- Probability of KP281 given Quadragenarian -> P(KP281|Quadragenarian): **50%**
- Probability of KP481 given Vicenarian -> P(KP481|Vicenarian): **29%**
- Probability of KP481 given Tricenarian -> P(KP481|Tricenarian): **48%**
- Probability of KP481 given Quadragenarian -> P(KP481|Quadragenarian): **17%**
- Probability of KP781 given Vicenarian -> P(KP781|Vicenarian): **25%**
- Probability of KP781 given Tricenarian -> P(KP781|Tricenarian): **12%**
- Probability of KP781 given Quadragenarian -> P(KP781|Quadragenarian): **33%**

In [ ]:

4.4 Probability of Product and Income Group

In [81]:
```
data['Income'].describe()
```

Out[81]:
```
count       180.000000
mean      53719.577778
std       16506.684226
min       29562.000000
25%       44058.750000
50%       50596.500000
75%       58668.000000
max      104581.000000
Name: Income, dtype: float64
```

In [82]:
```
income_bins = [29560,50000,80000,104582]
income_labels = ['Low','Medium','High']
data['Income_Group'] = pd.cut(data['Income'], bins = income_bins, labels = income_labels)
```

In [83]:
```
pd.crosstab(index = data['Product'], columns = data['Income_Group'], margins = True, normalize = True).r
```

Out[83]:

Income_Group	Low	Medium	High	All
Product				
KP281	0.267	0.178	0.000	0.444
KP481	0.167	0.167	0.000	0.333
KP781	0.028	0.089	0.106	0.222
All	0.461	0.433	0.106	1.000

In [84]:
```
def prob_incomegroup_product(data, incomegroup):
    """Function to compute conditional probability of marital status given product"""
    total = len(data)
    products = data['Product'].unique()
    condition_prob = {}
    for product in products:
        # Computing the length of Products
        product_total = len(data.loc[data['Product'] == product])
        # Computing the total of product and maritalstatus
        product_income_total = len(data.loc[(data['Product'] == product) & (data['Income_Group'] == inco
        # Computing the probability of gender given product
        prob_income_product = product_income_total/product_total
        condition_prob[product] = prob_income_product
```

```
    for product, prob in condition_prob.items():
        print(f"Probability of {incomegroup} income group customer buying {product} treadmill -> P({inco
```

In [85]:
```
def prob_product_incomegroup(data, product):
    """Function to compute conditional probability of product given marital status"""
    total = len(data)
    incomegroup = data['Income_Group'].unique()
    condition_prob = {}
    for income in incomegroup:
        # Computing the length of marital status
        income_total = len(data.loc[data['Income_Group'] == income])
        # Computing the total of product and marital status
        product_income_total = len(data.loc[(data['Product'] == product) & (data['Income_Group'] == inco
        # Computing the probability of gender given product
        prob_product_income = product_income_total/income_total
        condition_prob[income] = prob_product_income

    for income, prob in condition_prob.items():
        print(f"Probability of {product} given {income} income group -> P({product}|{income}): {np.round
```

In [86]:
```
for income in data['Income_Group'].unique():
    prob_incomegroup_product(data, income)
```

```
Probability of Low income group customer buying KP281 treadmill -> P(Low|KP281): 0.6
Probability of Low income group customer buying KP481 treadmill -> P(Low|KP481): 0.5
Probability of Low income group customer buying KP781 treadmill -> P(Low|KP781): 0.12
Probability of Medium income group customer buying KP281 treadmill -> P(Medium|KP281): 0.4
Probability of Medium income group customer buying KP481 treadmill -> P(Medium|KP481): 0.5
Probability of Medium income group customer buying KP781 treadmill -> P(Medium|KP781): 0.4
Probability of High income group customer buying KP281 treadmill -> P(High|KP281): 0.0
Probability of High income group customer buying KP481 treadmill -> P(High|KP481): 0.0
Probability of High income group customer buying KP781 treadmill -> P(High|KP781): 0.48
```

In [87]:
```
for product in data['Product'].unique():
    prob_product_incomegroup(data, product)
```

```
Probability of KP281 given Low income group -> P(KP281|Low): 0.58
Probability of KP281 given Medium income group -> P(KP281|Medium): 0.41
Probability of KP281 given High income group -> P(KP281|High): 0.0
Probability of KP481 given Low income group -> P(KP481|Low): 0.36
Probability of KP481 given Medium income group -> P(KP481|Medium): 0.38
Probability of KP481 given High income group -> P(KP481|High): 0.0
Probability of KP781 given Low income group -> P(KP781|Low): 0.06
Probability of KP781 given Medium income group -> P(KP781|Medium): 0.21
Probability of KP781 given High income group -> P(KP781|High): 1.0
```

Insights on Probability of Product and Gender

> **1. Marginal Probabilities**:

- Probability of KP281 -> P(KP281) = **44.4%**
- Probability of KP481 -> P(KP481) = **33.3%**
- Probability of KP781 -> P(KP781) = **22.2%**
- Probability of Low -> P(Low) = **46.1%**
- Probability of Medium -> P(Medium) = **43.3%**
- Probability of High -> P(High) = **10.6%**

> **2. Condiitional Probabilities**:

- Probability of Low income group customer buying KP281 treadmill -> P(Low|KP281): **60%**
- Probability of Low income group customer buying KP481 treadmill -> P(Low|KP481): **50%**
- Probability of Low income group customer buying KP781 treadmill -> P(Low|KP781): **12%**
- Probability of Medium income group customer buying KP281 treadmill -> P(Medium|KP281): **40%**
- Probability of Medium income group customer buying KP481 treadmill -> P(Medium|KP481): **50%**
- Probability of Medium income group customer buying KP781 treadmill -> P(Medium|KP781): **40%**
- Probability of High income group customer buying KP281 treadmill -> P(High|KP281): **0%**
- Probability of High income group customer buying KP481 treadmill -> P(High|KP481): **0%**
- Probability of High income group customer buying KP781 treadmill -> P(High|KP781): **48%**
- robability of KP281 given Low income group -> P(KP281|Low): **58%**
- Probability of KP281 given Medium income group -> P(KP281|Medium): **41%**

- Probability of KP281 given High income group -> P(KP281|High): **0%**
- Probability of KP481 given Low income group -> P(KP481|Low): **36%**
- Probability of KP481 given Medium income group -> P(KP481|Medium): **38%**
- Probability of KP481 given High income group -> P(KP481|High): **0%**
- Probability of KP781 given Low income group -> P(KP781|Low): **6%**
- Probability of KP781 given Medium income group -> P(KP781|Medium): **21%**
- Probability of KP781 given High income group -> P(KP781|High): **100%**

4.5 Probability Questions

What is the probability of a male customer buying a KP781 treadmill?

There is **82%** chance that Male customer buying KP781

What is the probability of a female customer buying a KP781 treadmill?

There is **18%** chance that Male customer buying KP781

What is the probability of KP281 treadmill given a male customer

There is **38%** chance that KP281 treadmill given a male customer

What is the probability of KP281 treadmill given a female customer

There is **53%** chance that KP281 treadmill given a female customer

What is the probability of a single customer buying a KP781 treadmill?

There is **42%** chance that Single customer buying KP781

What is the probability of a married customer buying a KP781 treadmill?

There is **58%** chance that Married customer buying KP781

What is the probability of a KP781 given customer is a single?

There is **23%** chance that KP781 given single customer

6 Customer Profilling

Customer profile for KP281:

- Gender: Equally preferred by both male and female customers.
- Marital Status: Higher likelihood of purchase among married customers.
- Income Group: Popular among customers in the low to medium income bracket.
- Education Level: Favored by customers with 14 to 16 years of education.
- Frequency of Use: Preferred by customers who use a treadmill less than 4 times per week.
- Mileage: Ideal for those who average between 60 to 90 miles of treadmill usage.
- Overall: KP281 has a higher chance of being purchased compared to other treadmill models.

Customer profile for KP481:

- Gender: Equally preferred by both male and female customers.
- Marital Status: Higher likelihood of purchase among married customers.
- Income Group: Popular among customers in the medium income bracket.
- Education Level: Favored by customers with 14 to 16 years of education.
- Frequency of Use: Preferred by customers who use a treadmill less than 4 times per week.
- Mileage: Ideal for those who average between 64 to 106 miles of treadmill usage.
- KP481 is the next most popular product after the KP281 model.

Customer profile for KP781:

- Gender: More preferred by male customers compared to female customers.
- Age: Most preferred by customers below the age of 30 years.
- Income Group: Favored by high-income group customers.
- Education Level: Popular among customers with more than 16 years of education.
- Frequency of Use: Most preferred by users who use the treadmill more than 4 times per week.
- Mileage: Ideal for those who average between 80 to 200 miles of treadmill usage.

7 Insights

Age:

- Minimum age: 18 years
- Maximum age: 50 years
- Average age: 28.7 years
- Top 5 ages: 25, 23, 24, 26, and 28 years
- Median age: 26 years
- 42% of customers are aged between 24 and 33 years
- 2% of data contains age outliers, with the lower bound at 10.5 and the upper bound at 46.5

Education:

- Minimum education: 12 years
- Maximum education: 21 years
- Average education: 15 years
- Median education: 16 years
- 80% of customers have between 14 and 16 years of education
- 85 customers have 16 years of education, 55 have 14 years, and 23 have 18 years
- 2% of data contains education outliers, with the lower bound at 11 and the upper bound at 19

Frequency of Use:

- Minimum: 2 times per week
- Maximum: 7 times per week
- Average: 3.4 times per week
- 85% of customers plan to use the treadmill 2 to 4 times a week; 15% plan to use it more than 4 times a week
- 69 customers use the treadmill at least 3 times a week, 52 customers 4 times a week, and 33 customers 2 times a week

---

Fitness Level:

- Self-rated fitness (1 to 5):
- Minimum: 1
- Maximum: 5
- Average rating: 3
- 54% of users rated themselves as medium (3)
- 1% rated themselves as poor (1)
- 17% rated themselves as excellent (5)

Miles:

- Minimum: 21 miles per week
- Maximum: 360 miles per week
- Average: 103 miles per week
- Median: 94 miles
- 60% of customers plan to walk/run between 66 and 114 miles per week
- 7% of data contains mileage outliers, with the lower bound at 0 and the upper bound at 188

Income:

- Minimum: Rs. 29,562
- Maximum: Rs. 104,581
- Average: Rs. 53,719
- Median: Rs. 50,596.5
- 60% of data lies between Rs. 44,058 and Rs. 58,668
- 11% of data contains income outliers, with the lower bound at Rs. 22,145 and the upper bound at Rs. 80,582

Gender:

- Unique values: 2 (Male and Female)
- Male: 104 (57.8%)
- Female: 76 (42.3%)
- Male customers are predominant for higher frequency usage, while females are predominant for 2 times a week usage

---

Marital Status:

- Unique values: 2 (Single and Partnered)
- Partnered: 107 (59.4%)
- Single: 73 (40.6%)
- Partnered customers have a higher likelihood of purchasing any treadmill

---

Product Insights

Treadmill Popularity:

- KP281: 44.4% share, $120,000 revenue
- KP481: 33.3% share, $105,000 revenue
- KP781: 22.2% share, $100,000 revenue

Gender and Frequency of Use:

- Female users are more predominant for 2 times a week usage, whereas male users are predominant for higher frequency usage.
- About 80% of female users plan to use the treadmill 2 to 4 times a week, while 70% of male users plan to use it 3 to 5 times a week.

Gender and Fitness Rating:

- Male users are predominant for all fitness ratings except for a rating of 2.
- About 80% of female users rated themselves between 2 and 3, whereas 90% of male users rated themselves between 3 and 5.

Marital Status and Age:

- The median age of single users who use treadmills is less than the median age of partnered users.
- Partnered users are more likely to buy any treadmill compared to single users.

Product and Marital Status:

- KP281 is more popular among both single and partnered users.
- KP481 has higher median age compared to all other categories.
- The variability of KP281 is more compared to the other two treadmills in both single and partnered users.

Income and Education:

- Income and education are correlated, with higher education often leading to higher income.
- 85 customers have 16 years of education, 55 have 14 years, and 23 have 18 years.

Usage and Fitness:

- Usage is highly correlated with fitness rating and planned miles.
- 85% of customers plan to use the treadmill 2 to 4 times a week; 15% plan to use it more than 4 times a week.

Usage and Age:

- Usage and age are weakly correlated.
- The median age of KP781 users for single customers is less than all other categories.

Fitness and Age:

- Fitness and age are weakly correlated.
- 54% of users rated themselves as medium in fitness.

Income:

- Income is almost equally correlated with other columns.
- The mean income of customers is Rs. 53,719, while the median is Rs. 50,596.5, indicating potential outliers.

8 Recommendations

Product lineup options:

- Maintain KP281 as an entry-level model appealing to a broad customer base.
- Position KP481 as a mid-range option with balanced features.
- Market KP781 as a premium, high-end model and provides many features

Marketing campaigns targetting Marital Status:

- Partnered customers are more likely to purchase any treadmill model compared to single customers.
- Highlight the benefits of the treadmills for couples or families in marketing campaigns to appeal to this segment.

Consider Gender Preferences in Marketing:

- For lower frequency usage (2 times a week), target female customers.
- For higher frequency usage, target male customers.
- Tailor fitness programs and promotions to these preferences.

Age-Specific Promotions:

- For vicenarians, promote KP781's advanced features and benefits.
- For tricenarians, focus on KP481's balanced features and affordability.
- For quadragenarians, highlight the reliability and value of KP281.

Income-Specific Promotions:

- Create offers and payment plans for high-income customers interested in KP781.
- Provide budget-friendly options for KP281 and KP481 for low to medium income groups

Fitness Level Programs:

> - Offer personalized fitness programs based on self-rated fitness levels.
> - Highlight success stories and testimonials from customers who improved their fitness using your treadmills.

In [ ]: