In [ ]:

```

```

Aerofit Case Study

AeroFit is a fitness program that combines elements of aerobics and strength training to provide a comprehensive workout. It typically involves high-intensity cardio exercises interspersed with strength-building movements, all performed in a group setting or individually. The goal of AeroFit is to improve cardiovascular health, increase endurance, build strength, and promote overall fitness. It often incorporates music and choreographed routines to make the workout more enjoyable and engaging. AeroFit can be tailored to different fitness levels, making it accessible to a wide range of participants.

Problem Statement:

1. To provide better recomendation of threadmil to the new customers, we need to analyse the characteristics of existing customer and the threadmil offered to them by the company.
2. Analysis of customers based on the different threadmil to be performed inorder to find the relationship between the customers and the threadmil types.

1. Data Basic Analysis

In [ ]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
df = pd.read_csv("aerofit_treadmill_1.csv")
```

In [ ]:

```
df.shape
```

Out[ ]:

```
(180, 9)
```

In [ ]:

```
df.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
```

In [ ]:
```
df.isna().sum()
```

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

In [ ]:
```
df.nunique()
```

Out[ ]:
```
Product           3
Age              32
Gender            2
Education         8
MaritalStatus     2
Usage             6
Fitness           5
Income           62
Miles            37
dtype: int64
```

In [ ]:
```
df.dtypes
```

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

```
In [ ]: df.columns
```

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

```
In [ ]: df["Product"].unique()
```

```
Out[ ]: array(['KP281', 'KP481', 'KP781'], dtype=object)
```

```
In [ ]: df["Fitness"].unique()
```

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

```
In [ ]: print(df["Usage"].unique())
        print(df["Income"].max(), df["Income"].min(), df["Income"].mean())
        print(df["Education"].unique())
        print(df["MaritalStatus"].unique())
        print(df["Age"].unique())
```

```
[3 2 4 5 6 7]
104581 29562 53719.57777777778
[14 15 12 13 16 18 20 21]
['Single' 'Partnered']
[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]
```

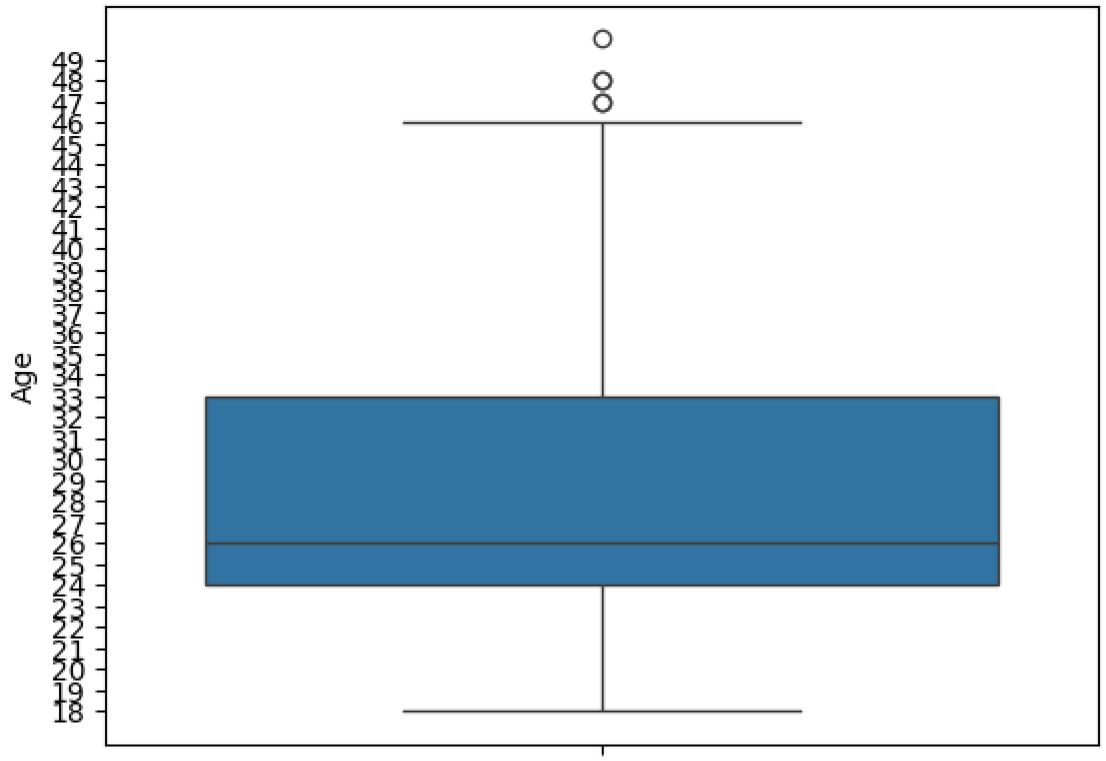
Inference:

- The dataset has 180 rows and 9 columns.

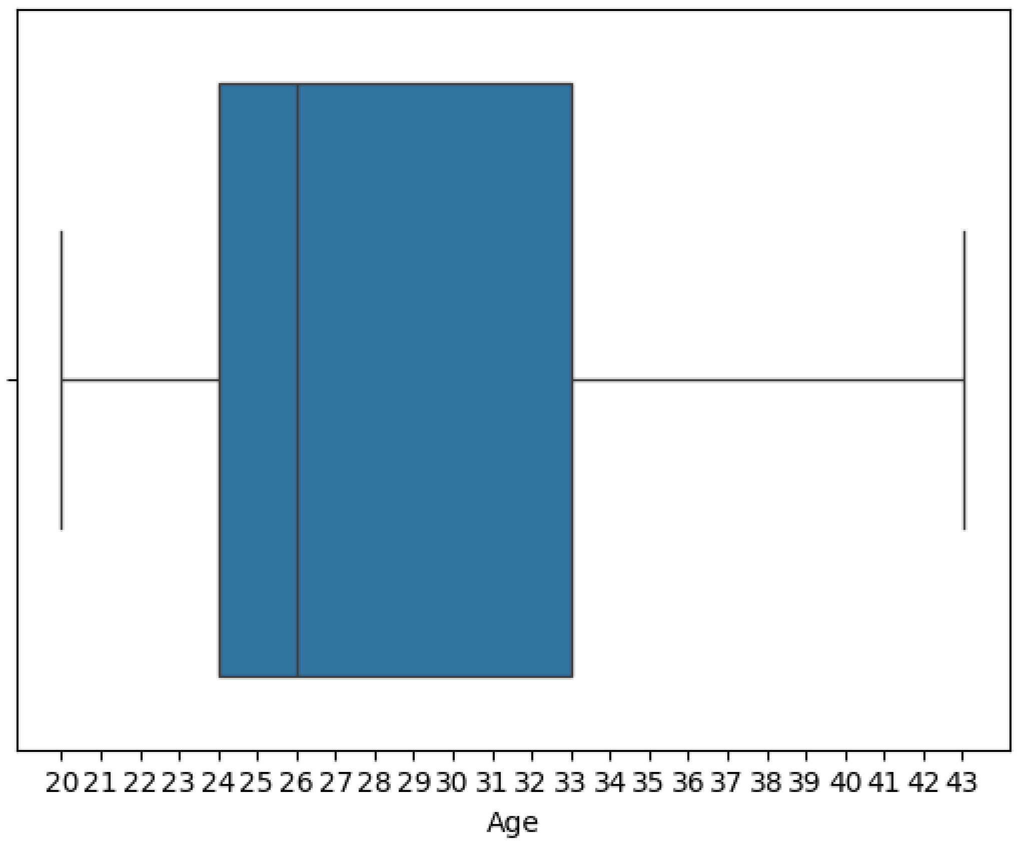
- Product, Gender and Marital Status are Categorical columns. Age, Education, Usage, Income, Fitness and Miles are Numerical Columns.
- The dataset has no null values.
- Product, Gender and Marital Status are Object data type. Rest all are integer data type.
- The Product Column has 3 unique values - KP281, KP481, KP781.
- Fitness column has 5 unique values ranging from 1 to 5 with 1 is Poor shape and 5 is excellent shape.
- Customers plans to use the threadmill 2,3,4,5,6 and 7 times per week.
- Income has the max value - 104581, min value - 29562 and the average value 53719.57777777778.
- The Education in years are 12,13,14,15,16,18,20,21
- Customers who belongs to both single and partnered are using the threadmil.
- The age of the customers using the threadmil are ranging from 18 to 50.

2. Detecting Outliers:

In [ ]:
```
sns.boxplot(df["Age"])
plt.yticks(range(df["Age"].min(), df["Age"].max(), 1))
plt.show()
```

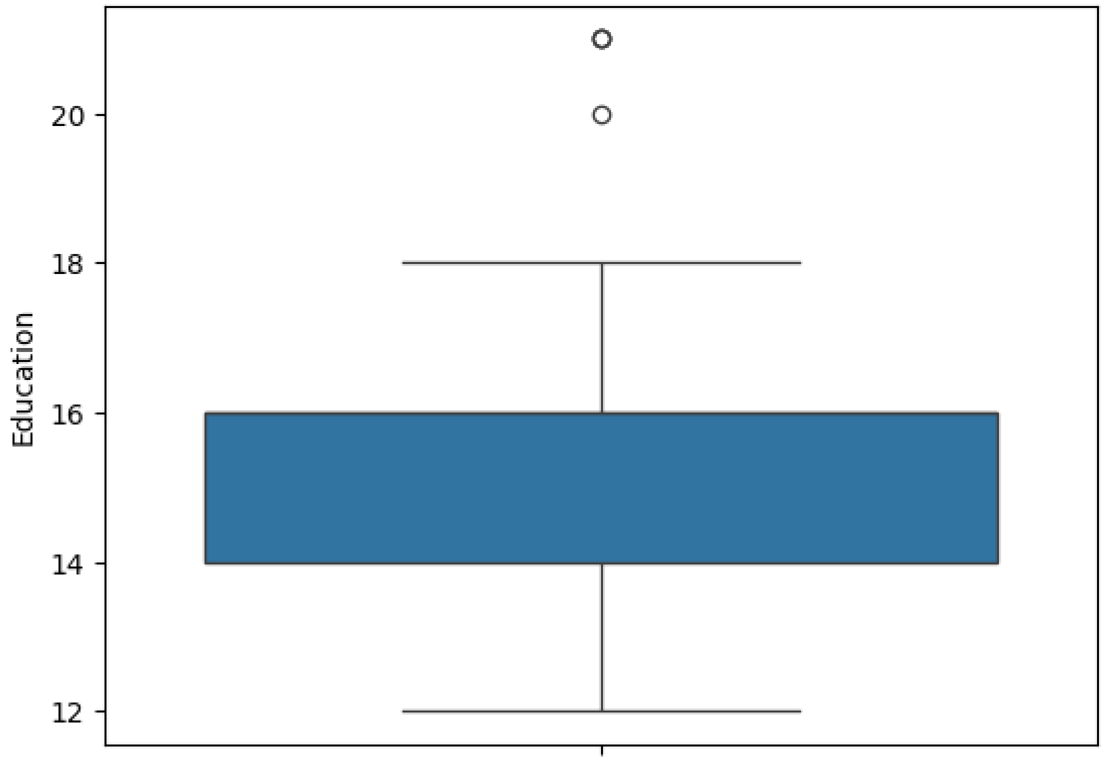


In [ ]:
```
df_clip = df.copy()
df_clip["Age"] = np.clip(df_clip["Age"], df_clip["Age"].quantile(0.05), df_clip["Age"].quantile(0.95))
df["Age"] = df_clip["Age"]
sns.boxplot(x = "Age", data = df)
plt.xticks(np.arange(df["Age"].min(), df["Age"].max(), 1))
plt.show()
```

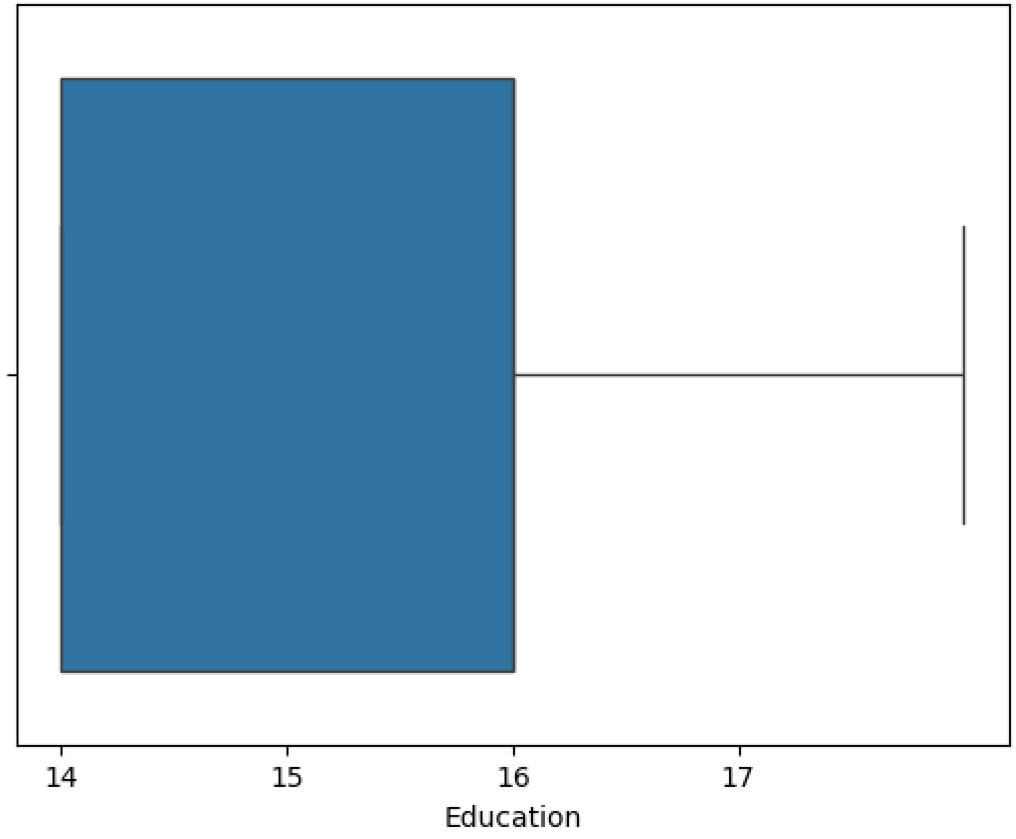


```
In [ ]: sns.boxplot(df["Education"])
```

```
Out[ ]: <Axes: ylabel='Education'>
```

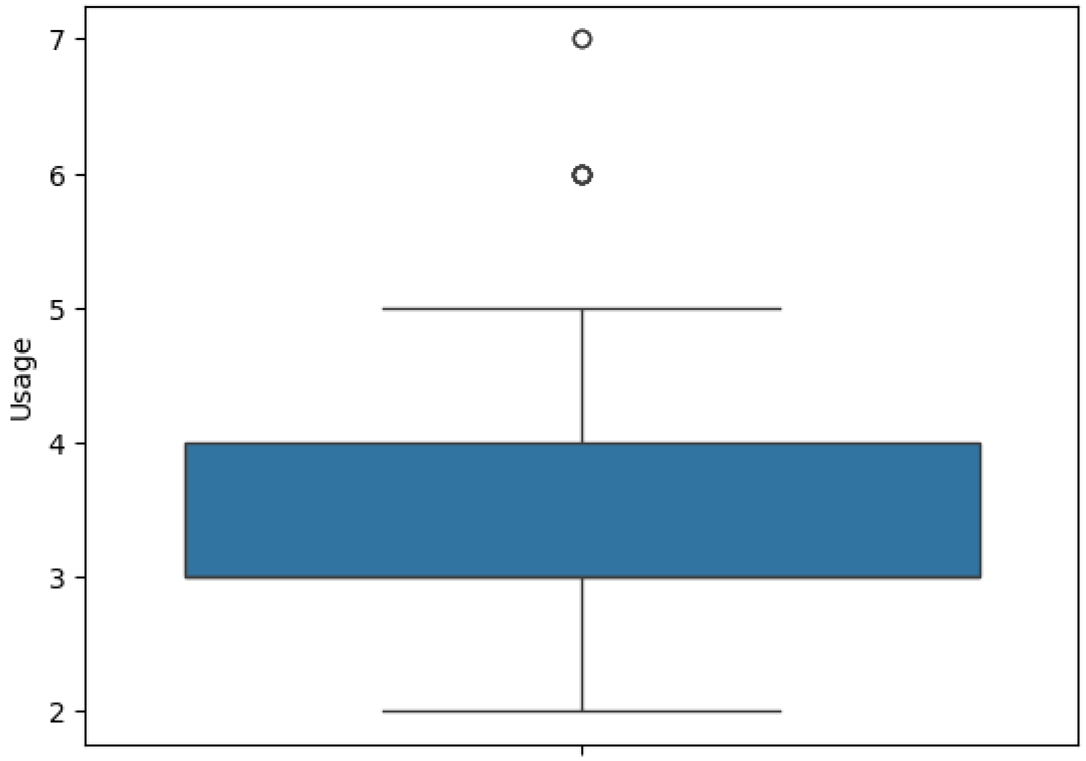


```
In [ ]: df_clip = df.copy()
        df_clip["Education"] = np.clip(df_clip["Education"], df_clip["Education"].quantile(0.05), df_clip["Education"].quantile
        df["Education"] = df_clip["Education"]
        sns.boxplot(x = "Education", data = df)
        plt.xticks(np.arange(df["Education"].min(), df["Education"].max(), 1))
        plt.show()
```

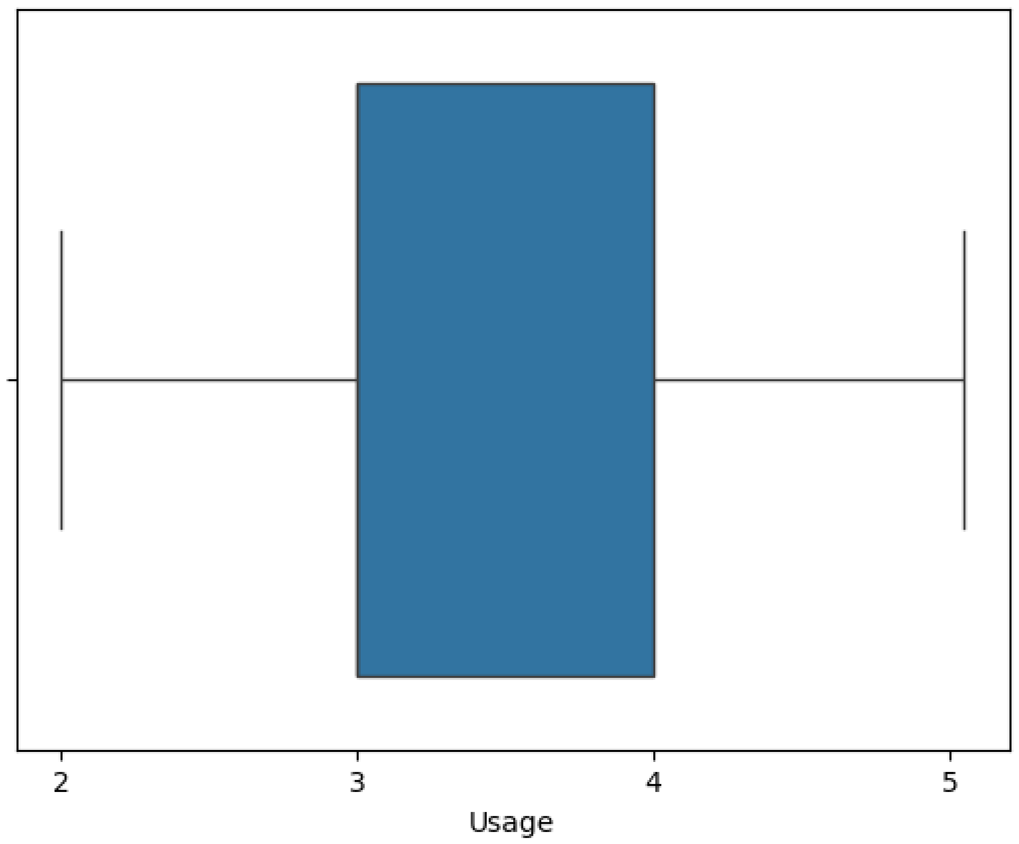


```
In [ ]: sns.boxplot(df["Usage"])
```

```
Out[ ]: <Axes: ylabel='Usage'>
```

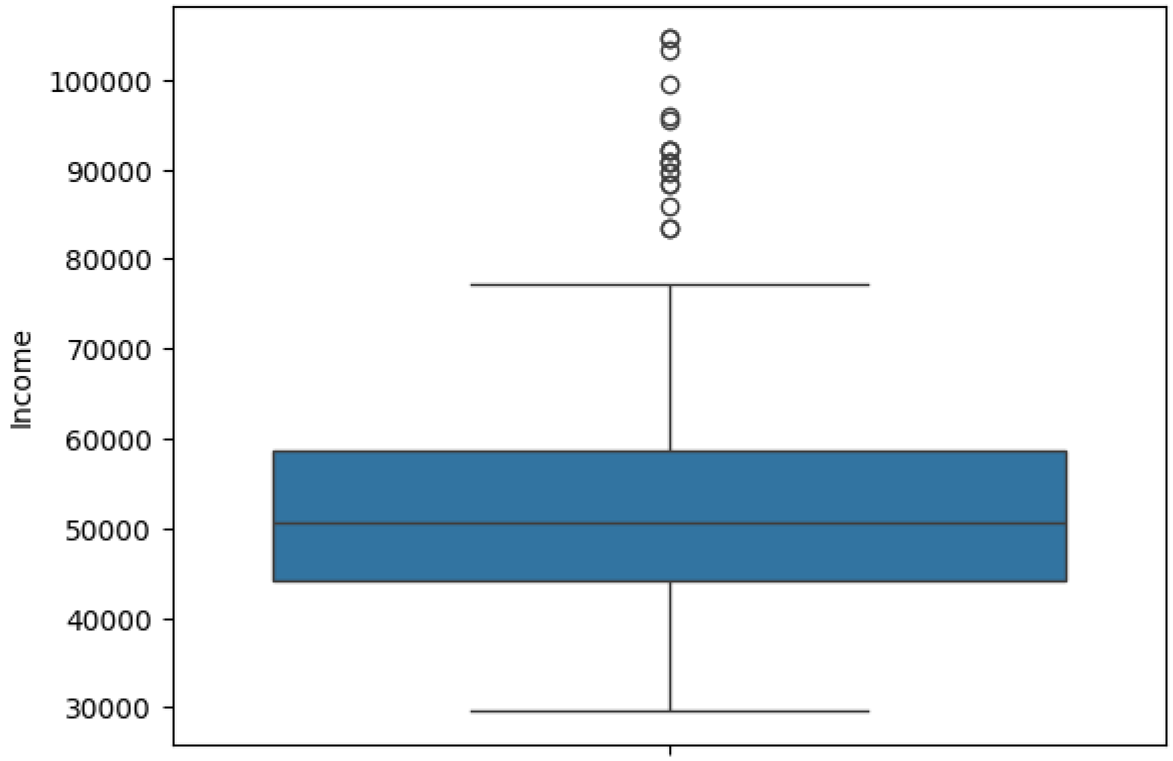


```
In [ ]: df_clip = df.copy()
        df_clip["Usage"] = np.clip(df_clip["Usage"], df_clip["Usage"].quantile(0.05), df_clip["Usage"].quantile(0.95))
        df["Usage"] = df_clip["Usage"]
        sns.boxplot(x = "Usage", data = df)
        plt.xticks(np.arange(df["Usage"].min(), df["Usage"].max(), 1))
        plt.show()
```

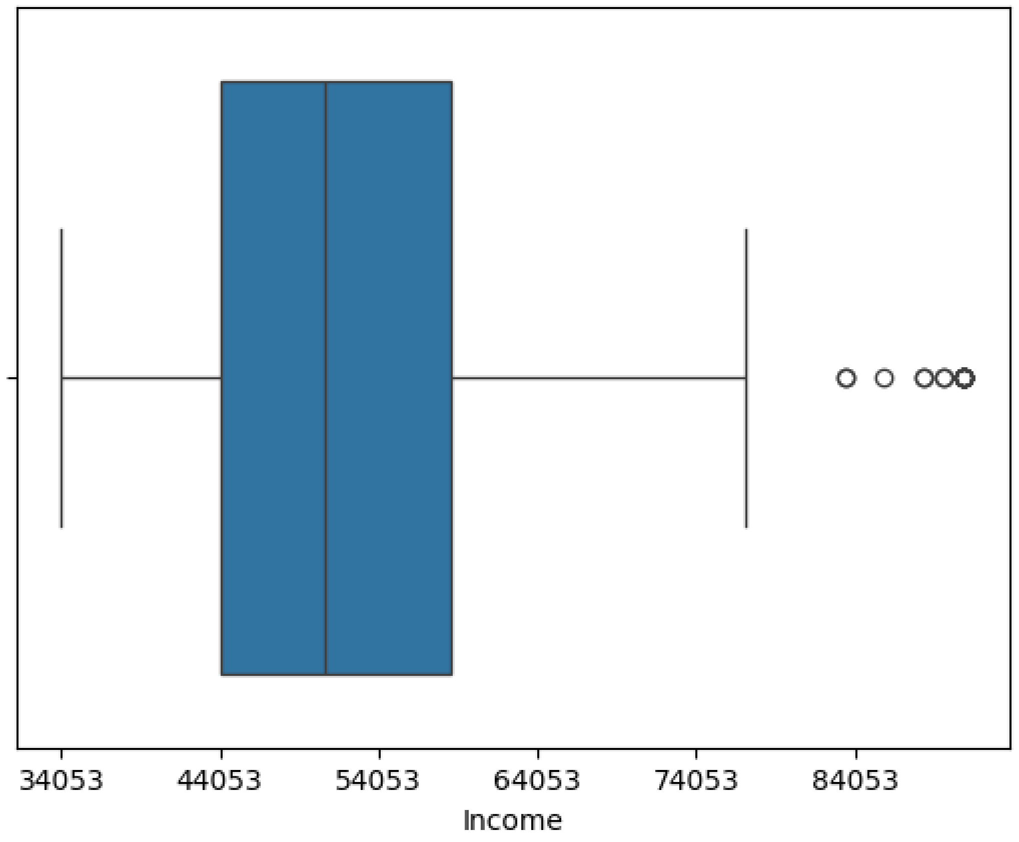


```
In [ ]: sns.boxplot(df["Income"])
```

```
Out[ ]: <Axes: ylabel='Income'>
```

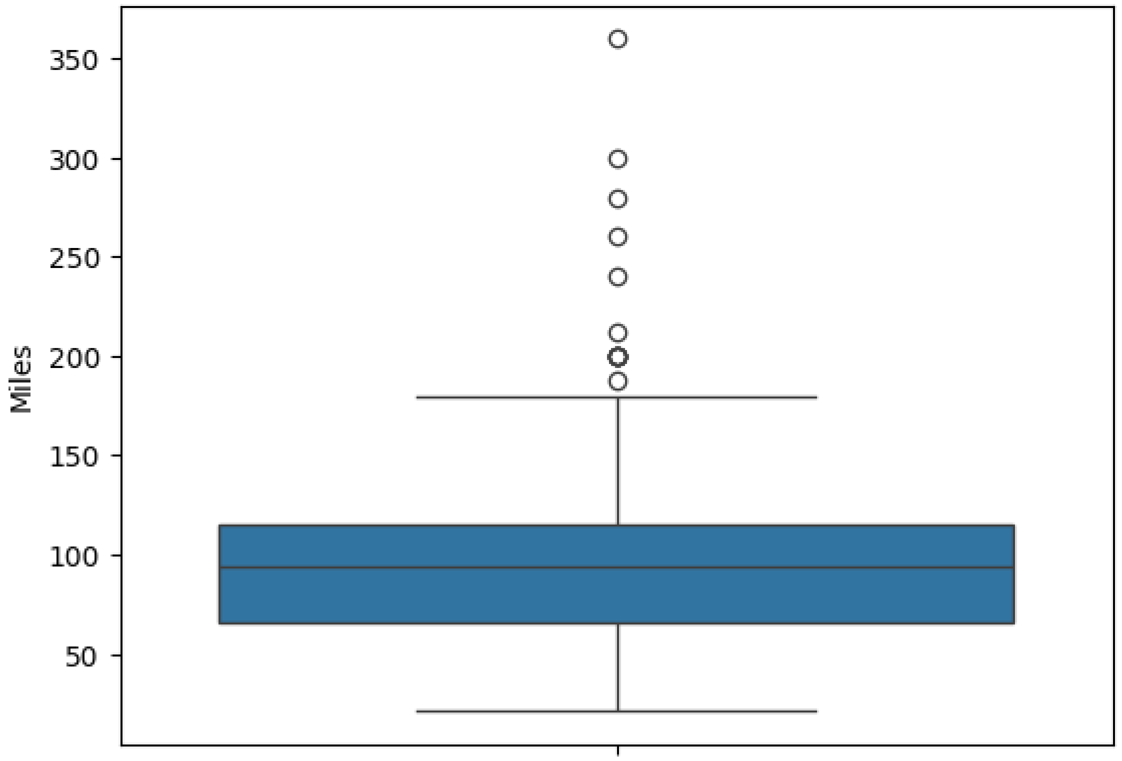


```
In [ ]: df_clip = df.copy()
        df_clip["Income"] = np.clip(df_clip["Income"], df_clip["Income"].quantile(0.05), df_clip["Income"].quantile(0.95))
        df["Income"] = df_clip["Income"]
        sns.boxplot(x = "Income", data = df)
        plt.xticks(np.arange(df["Income"].min(), df["Income"].max(), 10000))
        plt.show()
```

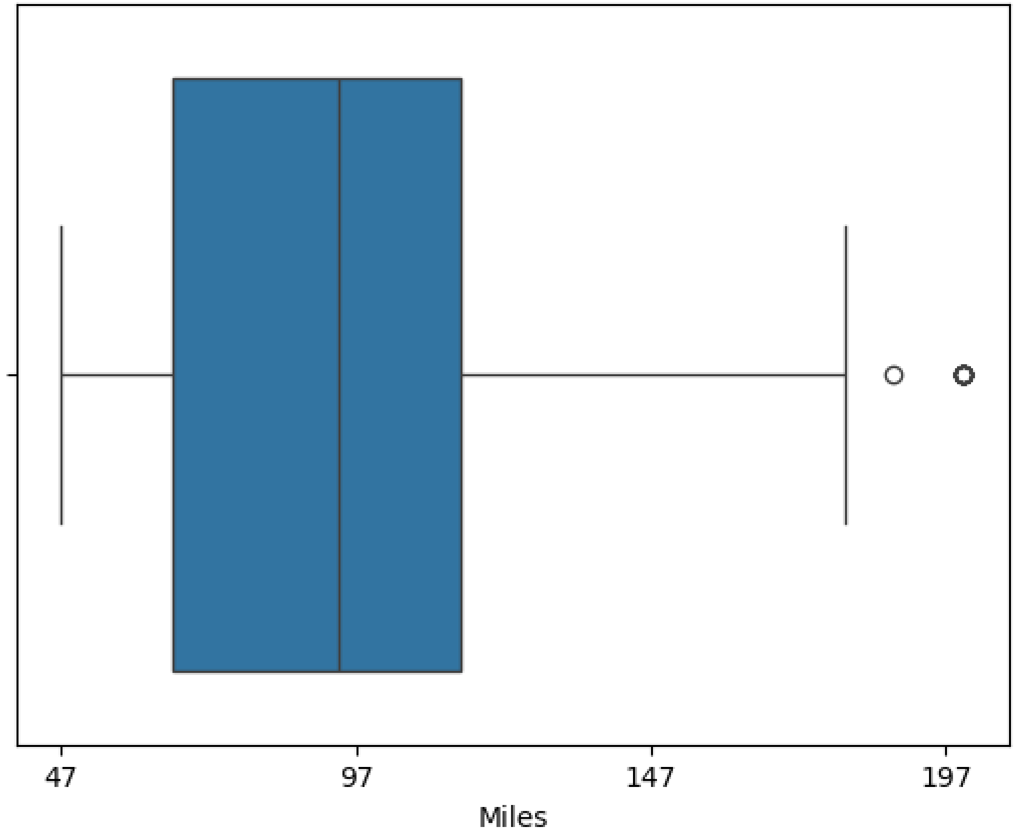


```
In [ ]: sns.boxplot(df["Miles"])
```

```
Out[ ]: <Axes: ylabel='Miles'>
```



```
In [ ]: df_clip = df.copy()
        df_clip["Miles"] = np.clip(df_clip["Miles"], df_clip["Miles"].quantile(0.05), df_clip["Miles"].quantile(0.95))
        df["Miles"] = df_clip["Miles"]
        sns.boxplot(x = "Miles", data = df)
        plt.xticks(np.arange(df["Miles"].min(), df["Miles"].max(), 50))
        plt.show()
```



Inference:

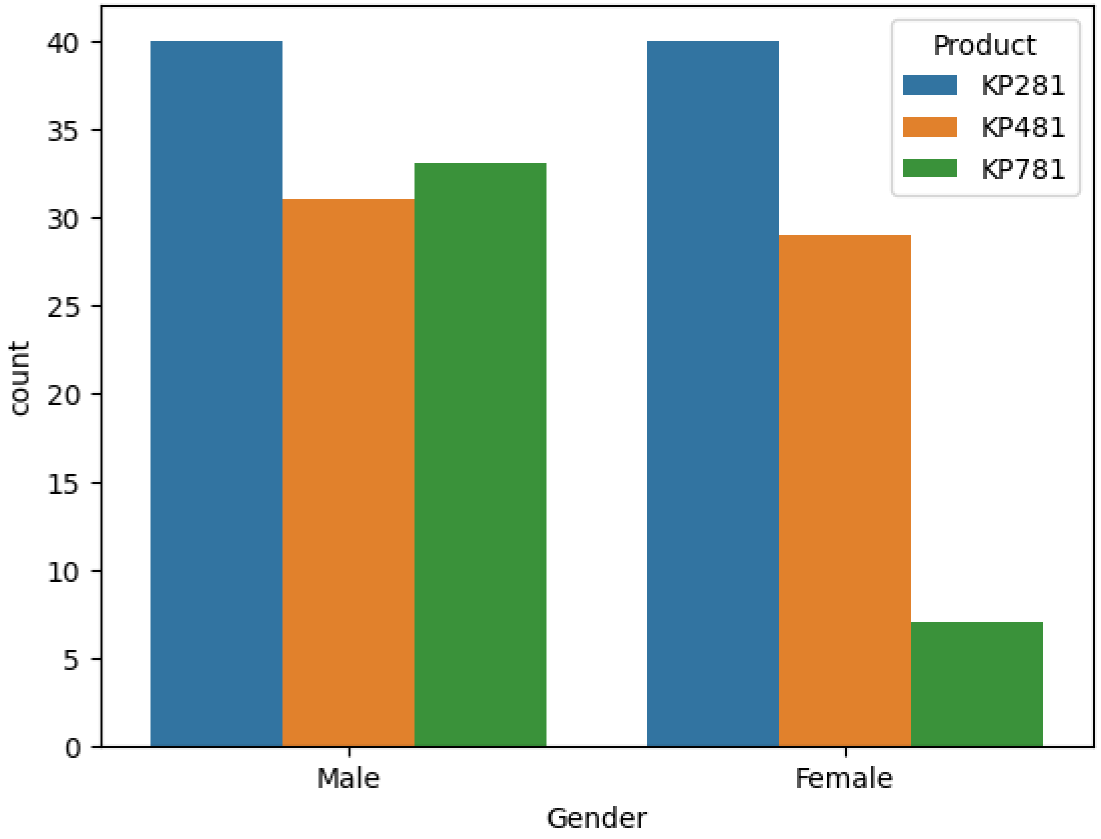
- Outliers in numerical data are data points that significantly differ from other data points in the dataset.
- Outliers can impact variance, mean, standard deviation of the data. Which can cause to misleading insights.
- So the Outliers were removed from the columns Age, Education, Usage, Income and Miles

3. Categorical Columns Vs Product

1. Gender Vs Product

```
In [ ]: sns.countplot(x= "Gender", hue = "Product", data = df)

Out[ ]: <Axes: xlabel='Gender', ylabel='count'>
```



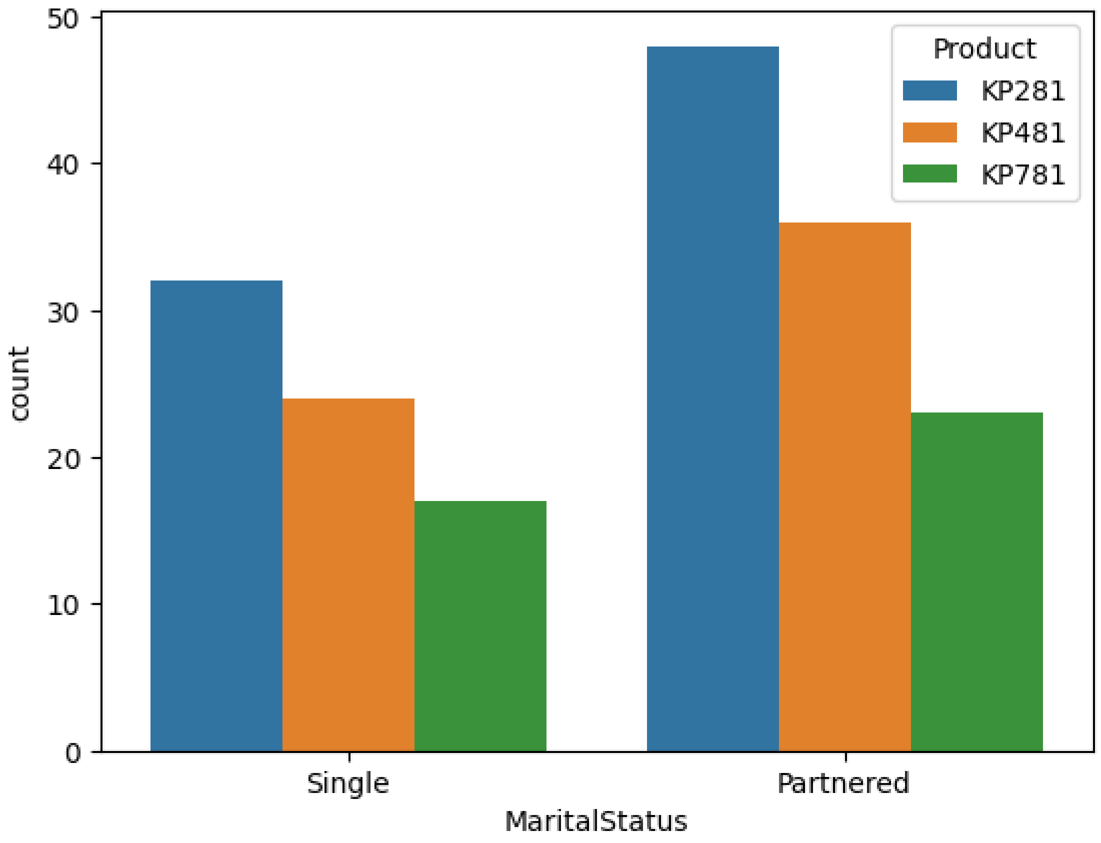
Inference:

1. Male customers shows intrest in all the 3 products compared to Female.
2. KP281 is the most popular product among both the Genders.
3. KP481 is the least bought product in Male Category.
4. KP781 is the lease bought product in Female Category.

2. Marital Status Vs Product

```
In [ ]: sns.countplot(x= "MaritalStatus", hue ="Product", data = df)
```

```
Out[ ]: <Axes: xlabel='MaritalStatus', ylabel='count'>
```



Inference:

1. Compared to the single customers the purchase of all the 3 products are high in partnered customer.
2. KP281 is the popular product in both the categories.

```
In [ ]: df_marital_Product = df[["MaritalStatus", "Product"]].value_counts().reset_index()
        df_marital_Product["Percentage"] = df_marital_Product[0]/len(df)
        df_marital_Product
```

Out[ ]:

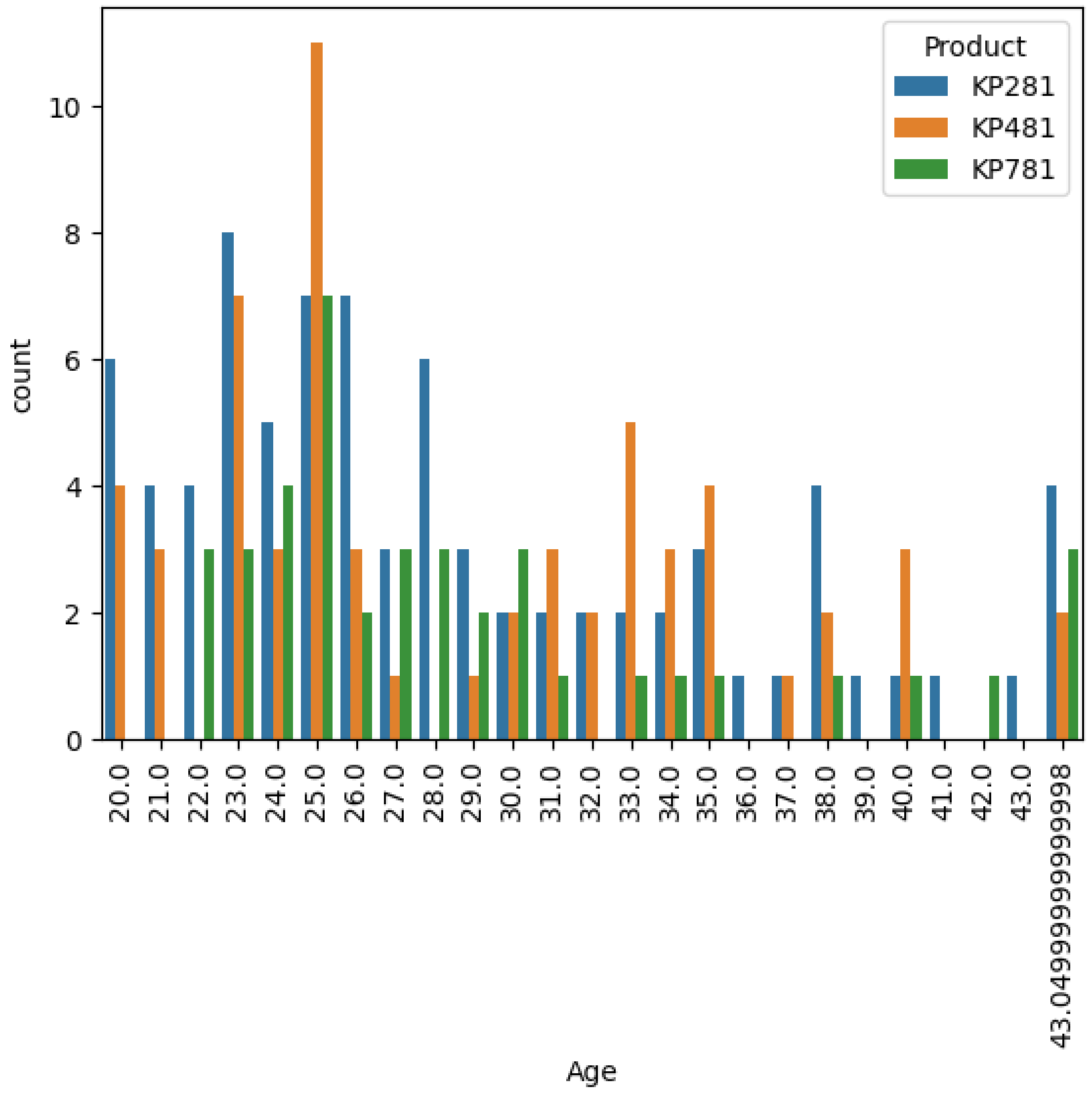
	MaritalStatus	Product	0	Percentage
0	Partnered	KP281	48	0.266667
1	Partnered	KP481	36	0.200000
2	Single	KP281	32	0.177778
3	Single	KP481	24	0.133333
4	Partnered	KP781	23	0.127778
5	Single	KP781	17	0.094444

Inference:

1. The Partnered Customers prefer KP281 26%, the second most prefered product is KP481 is 20%
2. The 17% of CUstomers and they are single prefer KP281.
3. The least bought by Partnered Customers and Single Customers are KP781

In [ ]:

```
sns.countplot(x= "Age", hue = "Product", data = df)
plt.xticks(rotation = 90)
plt.show()
```

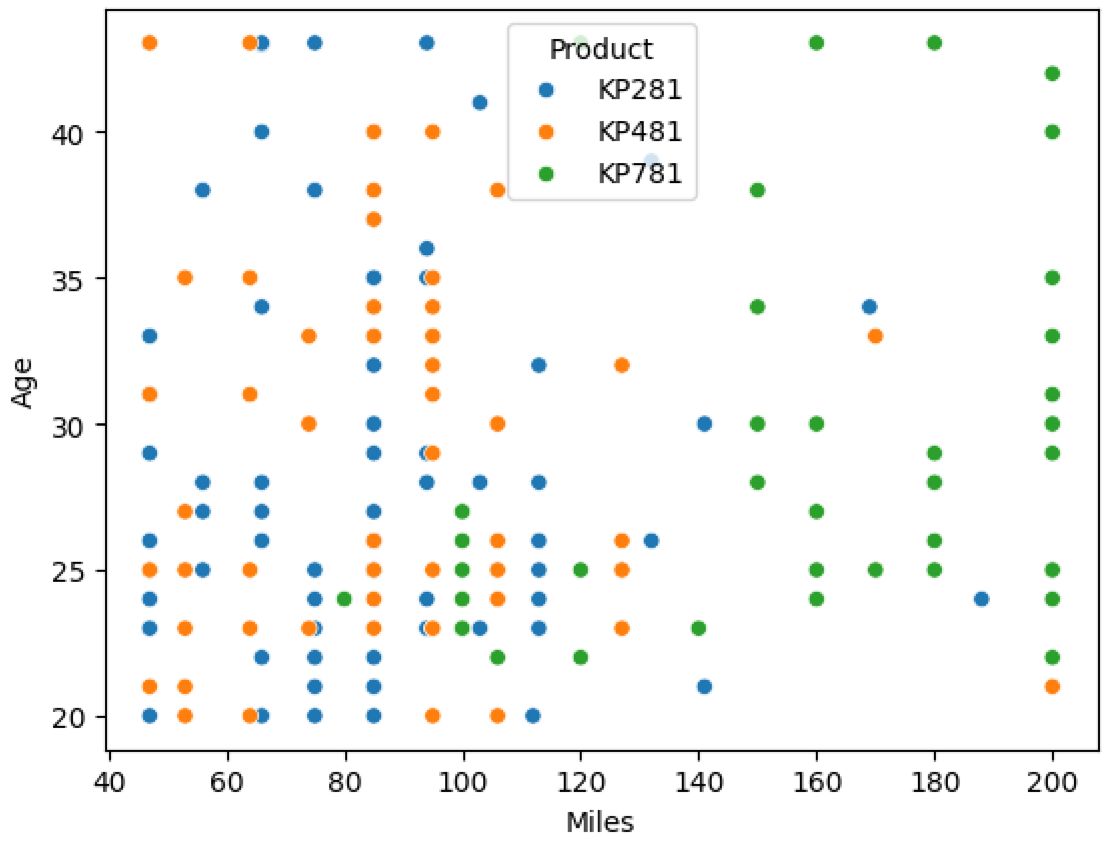


Inference:

1. The Customers of age 25, 23, 24 and 26 are very intrested in doing exercises in Threadmil
2. Customers of age 36, 39, 41, 42 amd 43 are the least intrested.

```
In [ ]: sns.scatterplot(x= "Miles", y ="Age", hue = "Product", data = df)
```

```
Out[ ]: <Axes: xlabel='Miles', ylabel='Age'>
```

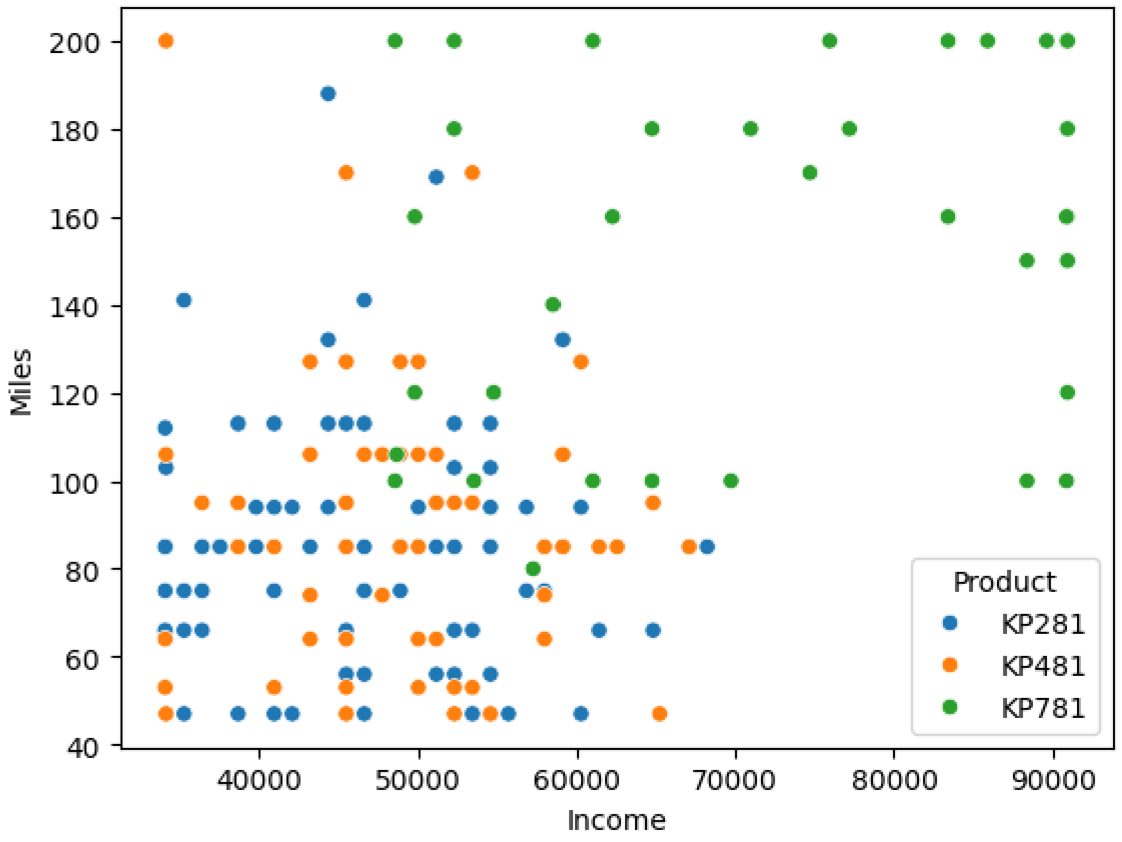


Inference:

1. Customers who prefer KP781 are the customers who runs in the range 120-200 Miles.
2. Customers who runs in the range 40 - 120 prefer KP481 and KP281

```
In [ ]: sns.scatterplot(x= "Income", y = "Miles", hue = "Product", data = df)
```

```
Out[ ]: <Axes: xlabel='Income', ylabel='Miles'>
```

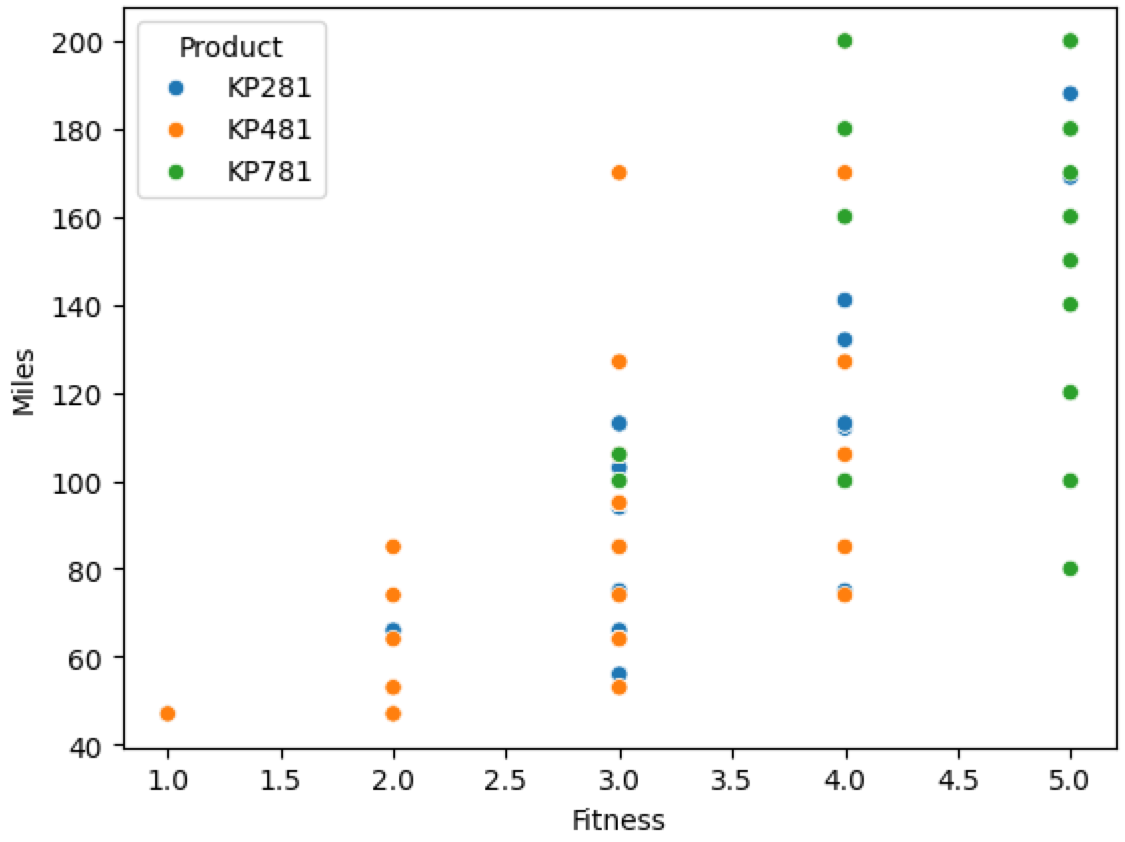


Inference:

1. Customers with the income more than 60000 and Miles above 120 prefer KP781.
2. Customers with Income 30000 to 60000 and Miles in the range 40 to 120 prefer KP281 and KP481.

```
In [ ]: sns.scatterplot(x= "Fitness", y = "Miles", hue = "Product", data = df)
```

```
Out[ ]: <Axes: xlabel='Fitness', ylabel='Miles'>
```

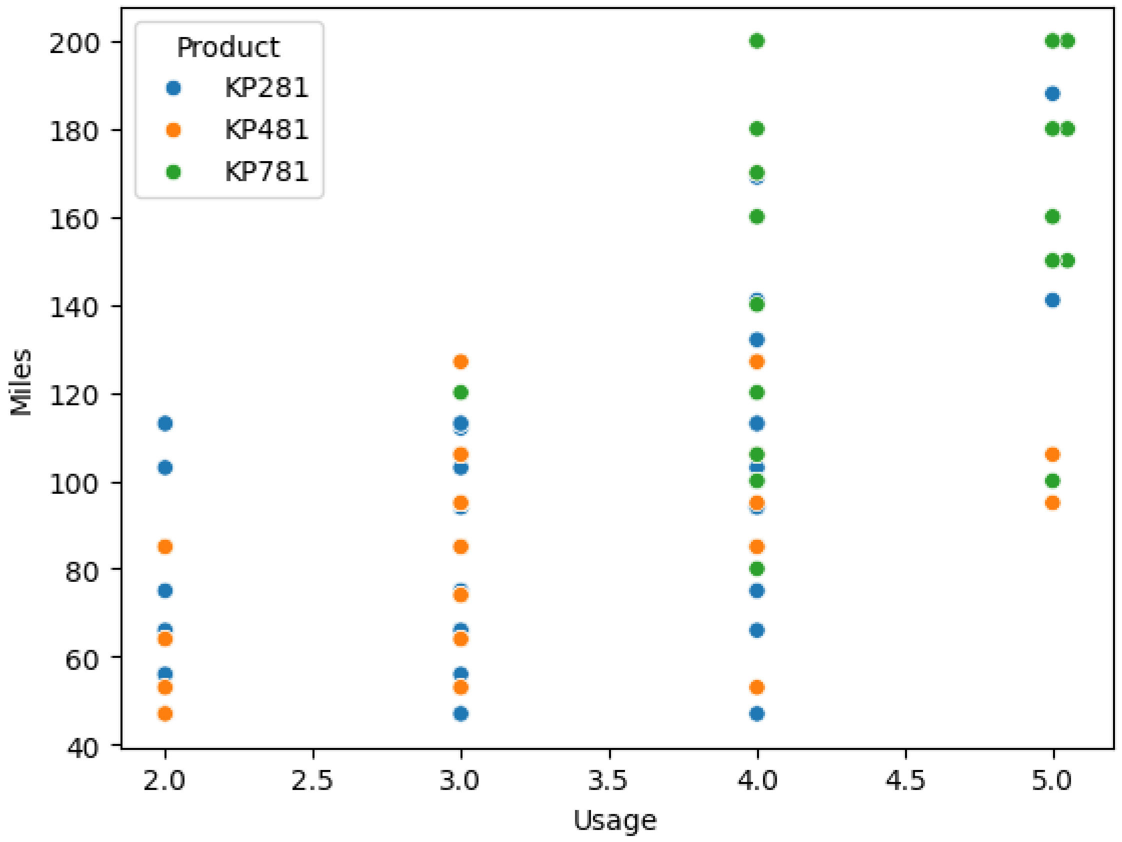


Inference:

The Customers with the Fitness level 3 and above prefer threadmil.

```
In [ ]: sns.scatterplot(x= "Usage", y = "Miles", hue = "Product", data = df)
```

```
Out[ ]: <Axes: xlabel='Usage', ylabel='Miles'>
```

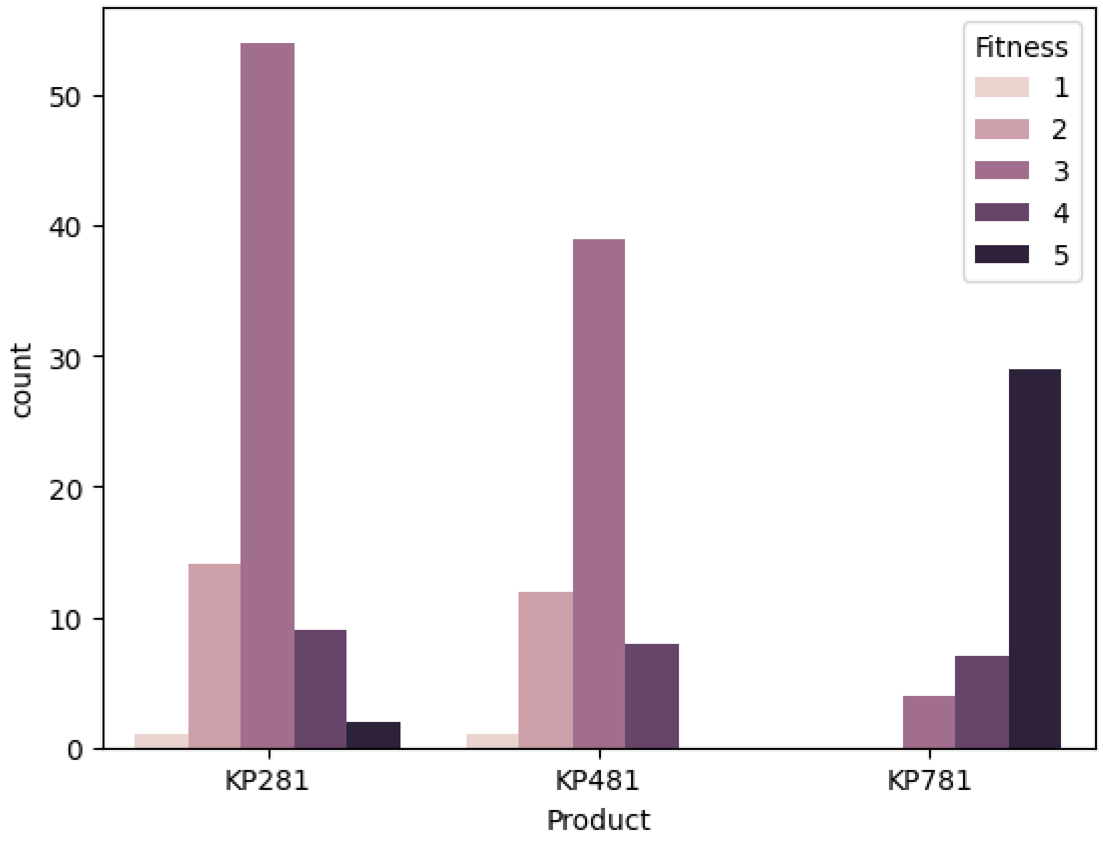


Inference:

The customers who do exercise 3 to 4 times a week prefer threadmil.

```
In [ ]: sns.countplot(x= "Product", hue = "Fitness", data = df)
```

```
Out[ ]: <Axes: xlabel='Product', ylabel='count'>
```

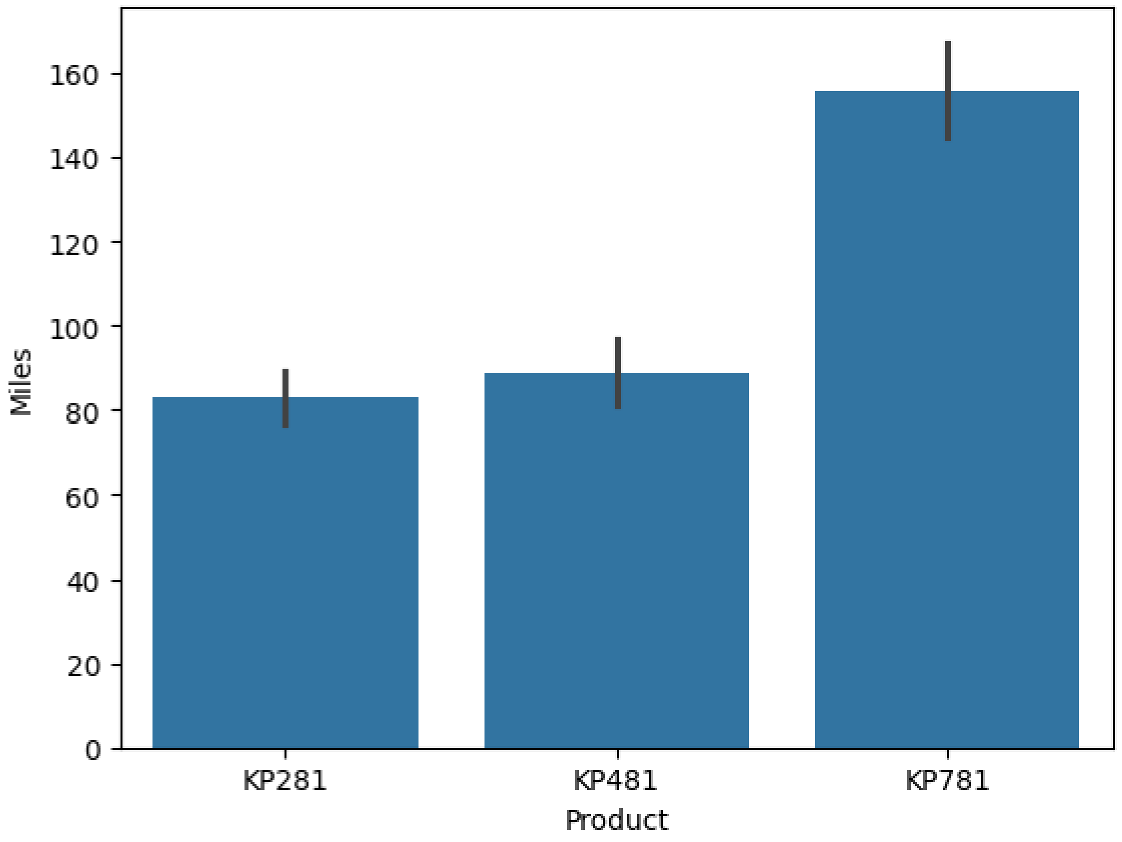


Inference:

1. The popular product in the Fitness level 5 is KP781.
2. The Popular product in the Fitness level 3 is KP281.
3. The Popular product in the Finess Level 4 is KP481.

```
In [ ]: sns.barplot(x= "Product", y = "Miles", data = df, estimator = np.mean)
```

```
Out[ ]: <Axes: xlabel='Product', ylabel='Miles'>
```



Inference:

1. The customer who buys KP281 uses the product for an average of 80 Miles/Week
2. The customer who buys KP481 uses the product for an average of 90 Miles/Week
3. The customer who buys KP281 uses the product for an average of 160 Miles/Week

4. Probability:

In [ ]:
```
table=pd.crosstab(index=df['Product'],columns='count')
marginal_prob=table/len(df)
print('marginal probability is',marginal_prob)
```

```
marginal probability is col_0       count
Product
KP281    0.444444
KP481    0.333333
KP781    0.222222
```

Inference:

1. KP281 is bought by 44% of the total customers.
2. Kp481 is bought by 33% and KP781 is by 22% of the overall customers.

Conditional Probability:

In [ ]:
```
df_gender_product = df[df["Gender"] == "Male"]["Product"].value_counts().reset_index()
sum_male = df_gender_product["Product"].sum()
df_gender_product["percentage"] = df_gender_product["Product"]/ sum_male
df_gender_product
```

Out[ ]:

	index	Product	percentage
0	KP281	40	0.384615
1	KP781	33	0.317308
2	KP481	31	0.298077

Inference:

1. Given that a customer is male, 38% chances are there, He may choose KP281.
2. similarly 31% are there for a male customer to choose KP781 and 29% chances for KP481.

In [ ]:
```
df_gender_product = df[df["Gender"] == "Female"]["Product"].value_counts().reset_index()
sum_female = df_gender_product["Product"].sum()
df_gender_product["percentage"] = df_gender_product["Product"]/ sum_female
df_gender_product
```

Out[ ]:

	index	Product	percentage
0	KP281	40	0.526316
1	KP481	29	0.381579
2	KP781	7	0.092105

Inference:

1. Given that a customer is female, 52% are there, she may buy KP281.
2. Similarly 38% chances for KP481 and 9% chances for KP781.

In [ ]:

```
df_gender_product = df[df["Gender"] == "Female"][["Product", "Fitness"]].value_counts().reset_index()
sum_female = df_gender_product[0].sum()
df_gender_product["Percentage"] = df_gender_product[0]/sum_female
df_gender_product.rename(columns = {0:"Count"}, inplace = True)
df_gender_product
```

Out[ ]:

	Product	Fitness	Count	Percentage
0	KP281	3	26	0.342105
1	KP481	3	18	0.236842
2	KP281	2	10	0.131579
3	KP481	2	6	0.078947
4	KP781	5	5	0.065789
5	KP481	4	4	0.052632
6	KP281	4	3	0.039474
7	KP281	5	1	0.013158
8	KP481	1	1	0.013158
9	KP781	3	1	0.013158
10	KP781	4	1	0.013158

In [ ]:

```
df_gender_product = df[df["Gender"] == "Male"][["Product", "Fitness"]].value_counts().reset_index()
sum_male = df_gender_product[0].sum()
df_gender_product["Percentage"] = df_gender_product[0]/sum_male
df_gender_product.rename(columns = {0:"Count"}, inplace = True)
df_gender_product
```

Out[ ]:

	Product	Fitness	Count	Percentage
0	KP281	3	28	0.269231
1	KP781	5	24	0.230769
2	KP481	3	21	0.201923
3	KP281	4	6	0.057692
4	KP481	2	6	0.057692
5	KP781	4	6	0.057692
6	KP281	2	4	0.038462
7	KP481	4	4	0.038462
8	KP781	3	3	0.028846
9	KP281	1	1	0.009615
10	KP281	5	1	0.009615

In [ ]:

```
df_marital_Product = pd.crosstab(df["Product"], df["MaritalStatus"]).reset_index()

sum_partnered = df_marital_Product["Partnered"].sum()
sum_single = df_marital_Product["Single"].sum()
df_marital_Product["Partnered_Percentage"] = df_marital_Product["Partnered"]/ sum_partnered
df_marital_Product["Single_Percentage"] = df_marital_Product["Single"]/ sum_single
df_marital_Product
```

Out[ ]:

MaritalStatus	Product	Partnered	Single	Partnered_Percentage	Single_Percentage
0	KP281	48	32	0.448598	0.438356
1	KP481	36	24	0.336449	0.328767
2	KP781	23	17	0.214953	0.232877

Inference:

1. Given that a customer is Partnered, 44% chances are there that they may buy 281, 33% for KP481, 21% for KP781.
2. Given that a customer is Single, 43% chances for customer to may buy KP281, 32% KP481 and 23% for KP781

5. Correlation:

In [ ]:
```
df.corr()
```

```
<ipython-input-53-2f6f6606aa2c>:1: FutureWarning: The default value of numeric_only in 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 t
his warning.
  df.corr()
```

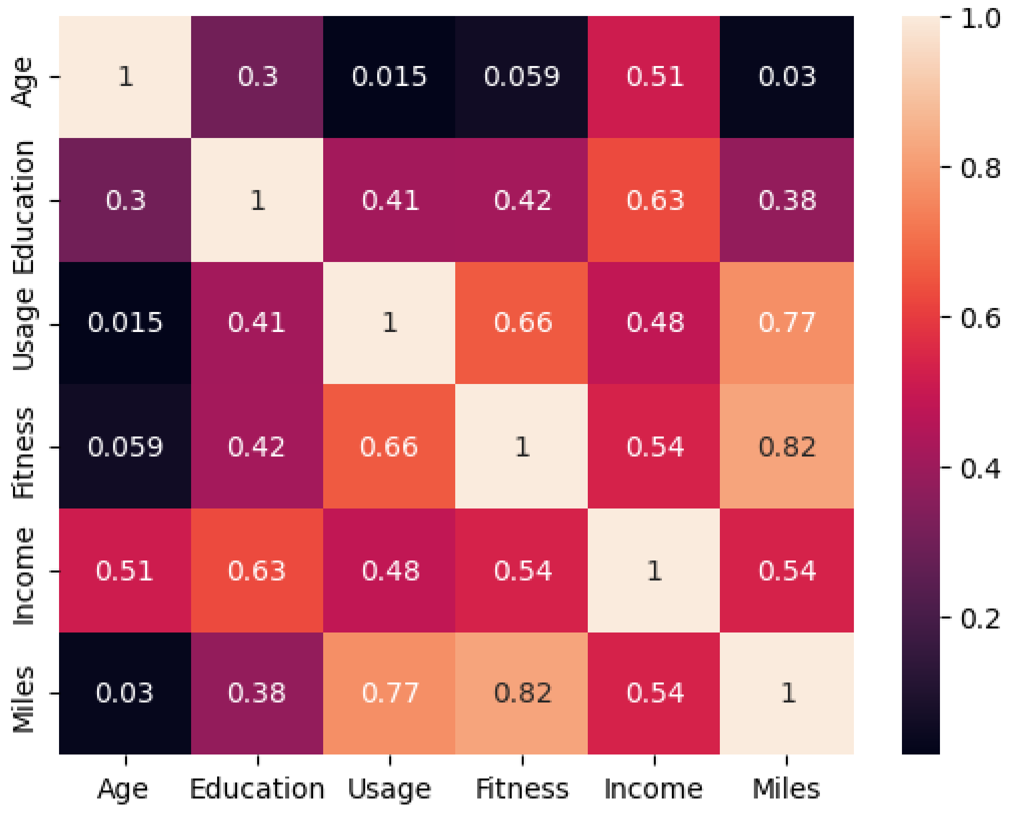
Out[ ]:

	Age	Education	Usage	Fitness	Income	Miles
Age	1.000000	0.301971	0.015394	0.059047	0.514362	0.029636
Education	0.301971	1.000000	0.413600	0.419020	0.628597	0.377294
Usage	0.015394	0.413600	1.000000	0.658169	0.481608	0.771030
Fitness	0.059047	0.419020	0.658169	1.000000	0.535945	0.822393
Income	0.514362	0.628597	0.481608	0.535945	1.000000	0.537297
Miles	0.029636	0.377294	0.771030	0.822393	0.537297	1.000000

In [ ]:
```
sns.heatmap(df.corr(), annot = True)
```

```
<ipython-input-56-fe43fffaf13b>:1: FutureWarning: The default value of numeric_only in 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 t
his warning.
  sns.heatmap(df.corr(), annot = True)
```

Out[ ]:
```
<Axes: >
```



	Age	Education	Usage	Fitness	Income	Miles
Age	1	0.3	0.015	0.059	0.51	0.03
Education	0.3	1	0.41	0.42	0.63	0.38
Usage	0.015	0.41	1	0.66	0.48	0.77
Fitness	0.059	0.42	0.66	1	0.54	0.82
Income	0.51	0.63	0.48	0.54	1	0.54
Miles	0.03	0.38	0.77	0.82	0.54	1

Inference

Based on the heatmap, Usage-Miles, Fitness-Miles, Income - Education, Fitness-Uage are highly correlated.

6.Customer Profiling

KP281

1. 44 % of total customers prefer this product
2. Both Male and Female customers prefer this.

3. Highly prefered by partnered customers. 26% of partnered and 17% single customers prefer this product.
4. Majority of the customers who buys this product are in the age group 25.
5. This product is prefered by medium income customers ranging from 30000 to 60000
6. Customer with Fitness Level 3 and 4 prefer this product.
7. Customer who buys this product runs Average of 80 Miles/week.

KP481

1. 33% of cutomers prefer this product
2. This product stands in second place based on the preference by both male and female customers
3. This product stands in the second place based on the preference by partnered and single customers. 20 % of partnered and 13 % of single customers prefer this product.
4. Mostly used by the customers of all age groups except 22, 28, 36, 39, 41,42 and 43. Customers of age 25 buys this product the most.
5. This product is prefered by medium income customers ranging from 30000 to 60000
6. This product is prefered by customers with Fitness level 2,3 and 4
7. Customer who buys this product runs Average of 85 Miles/week.

KP781

1. 22% of customers prefer this product.
2. This product stands in the last position based on the preference by both male and female customers
3. This product stands in the last place based on the preference by partnered and single customers. 12% of partnered and 9% of single customers prefer this product.
4. Customers who prefer this product the most belongs to the age group 23, 25, 26, 20 and 50.
5. Customers who prefers to longer Miles ranging from 120- 200 prefer this product.
6. Customers with Fitness level 3,4 and5 prefer this product and mostly by Fitness level 5.
7. This product is prefered by high income customers greater than 60000
8. Customer who buys this product runs Average of 160 Miles/week.

Recommendation

1. Customers who have good experience in Fitness prefer KP781 and high Income customers also.
2. Customers who are in entry level prefer KP281 and KP481, the Company can create a awareness regarding fitness in young people by conducting campaigns. By doing this Company can attract entry level customers in buying this product.
3. Company can create awareness regarding fitness and the features in different threadmil so that people who tries the threadmil can prefer to buy the product

In [ ]:
```
!jupyter
```

In [ ]:
```

```