```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
```

```
!gdown https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/125/original/aerofit_treadmill.csv
```

```
Downloading...
From: https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/125/original/aerofit_treadmill.csv
To: /content/aerofit_treadmill.csv
100% 7.28k/7.28k [00:00<00:00, 17.3MB/s]
```

```
customer = pd.read_csv('/content/aerofit_treadmill.csv')
```

importing the data set

```
customer
```

	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

```
customer['Product'].unique()
```

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

```
customer['Product'].value_counts()
```

Product	count
KP281	80
KP481	60
KP781	40

dtype: int64

```
customer.dtypes
```

	0
Product	object
Age	int64
Gender	object
Education	int64
MaritalStatus	object
Usage	int64
Fitness	int64
Income	int64
Miles	int64

dtype: object

Checking the data type of each column in the data set

```
customer.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
```

```
customer.describe(include = 'all')
```

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
count	180	180.000000	180	180.000000	180	180.000000	180.000000	180.000000	180.000000
unique	3	NaN	2	NaN	2	NaN	NaN	NaN	NaN
top	KP281	NaN	Male	NaN	Partnered	NaN	NaN	NaN	NaN
freq	80	NaN	104	NaN	107	NaN	NaN	NaN	NaN
mean	NaN	28.788889	NaN	15.572222	NaN	3.455556	3.311111	53719.577778	103.194444
std	NaN	6.943498	NaN	1.617055	NaN	1.084797	0.958869	16506.684226	51.863605
min	NaN	18.000000	NaN	12.000000	NaN	2.000000	1.000000	29562.000000	21.000000
25%	NaN	24.000000	NaN	14.000000	NaN	3.000000	3.000000	44058.750000	66.000000
50%	NaN	26.000000	NaN	16.000000	NaN	3.000000	3.000000	50596.500000	94.000000
75%	NaN	33.000000	NaN	16.000000	NaN	4.000000	4.000000	58668.000000	114.750000
max	NaN	50.000000	NaN	21.000000	NaN	7.000000	5.000000	104581.000000	360.000000

```
customer.shape
```

```
(180, 9)
```

the above is the shape of the data set

checking if there is any null values in each columns

```
customer.isna().sum()
```

	0
Product	0
Age	0
Gender	0
Education	0
MaritalStatus	0
Usage	0
Fitness	0
Income	0
Miles	0

dtype: int64

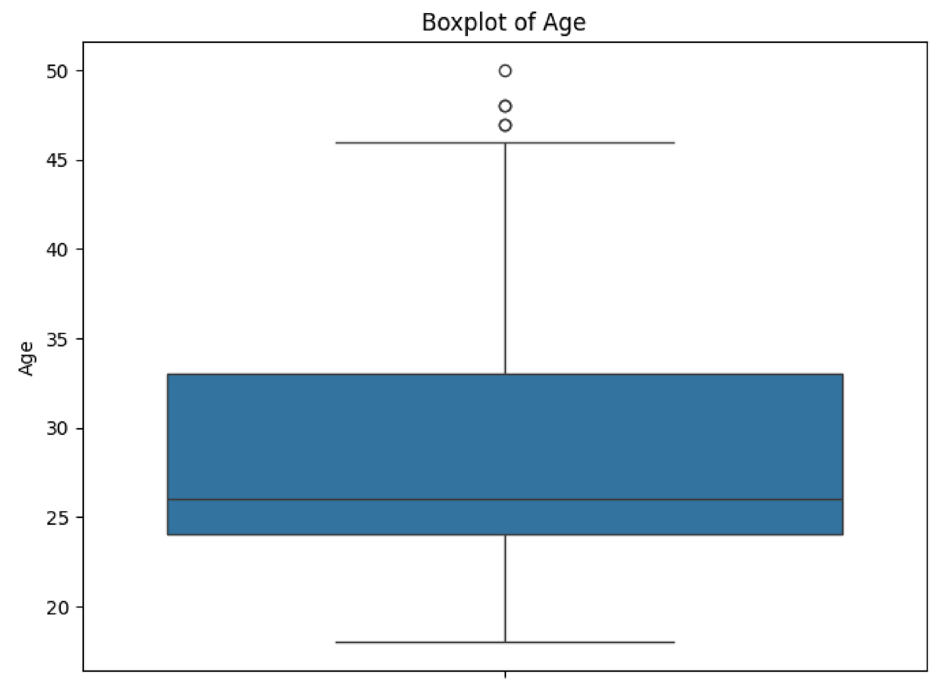
Observations:

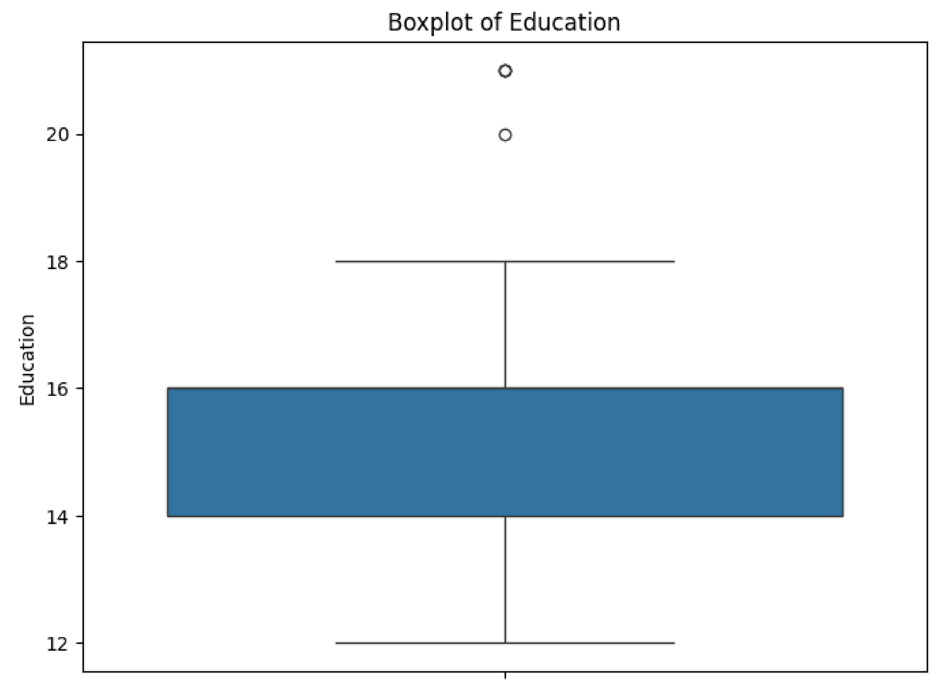
- There are no missing values in the data.(means there no null values)
- There are 3 unique products in the dataset.
- KP281 is the most frequent product.
- Minimum & Maximum age of the person is 20 & 43, mean is 28.641389 and 75% of persons have age less than or equal to 33.
- Most of the people are having 16 years of education i.e. 75% of people
- Out of 180 data points, 104's gender is Male and rest are the female.
- Standard deviation for Income & Miles is very high. These variables might have the outliers in it.

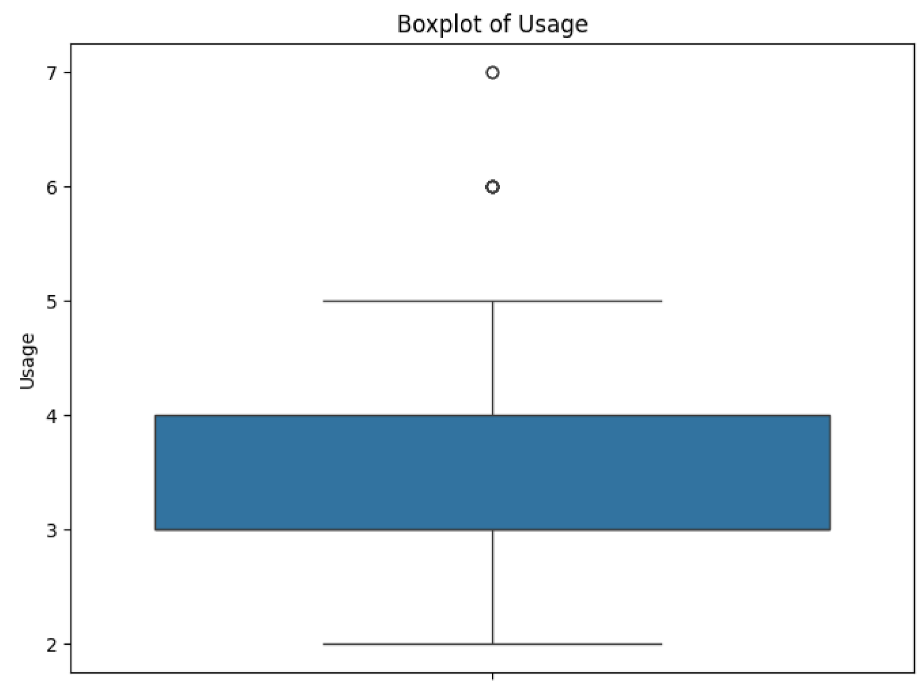
Detecting the outliers

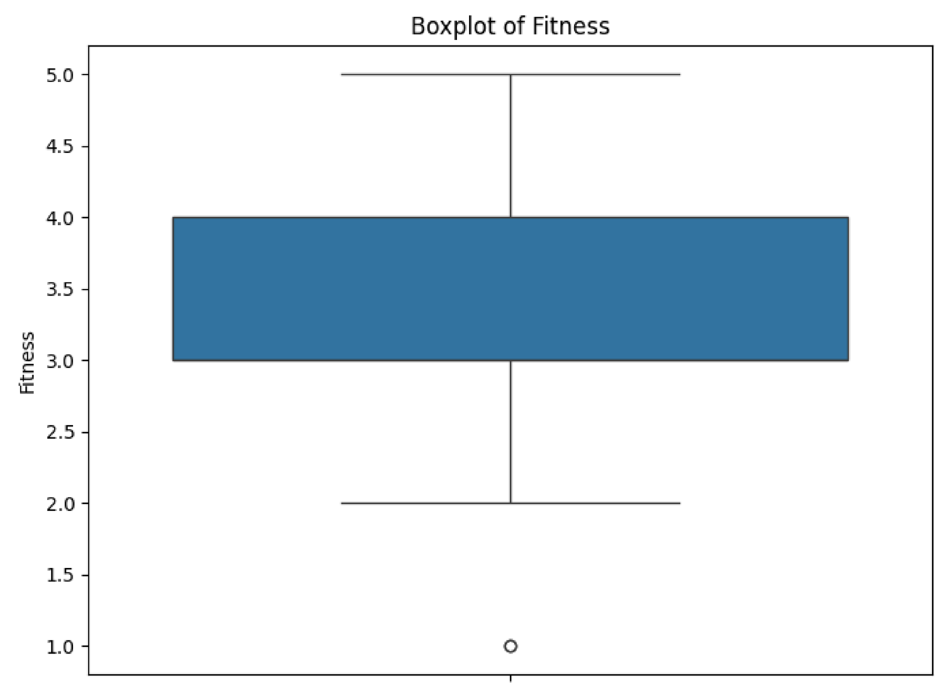
```
continuous_vars = customer.select_dtypes(include=['float64', 'int64'])
```

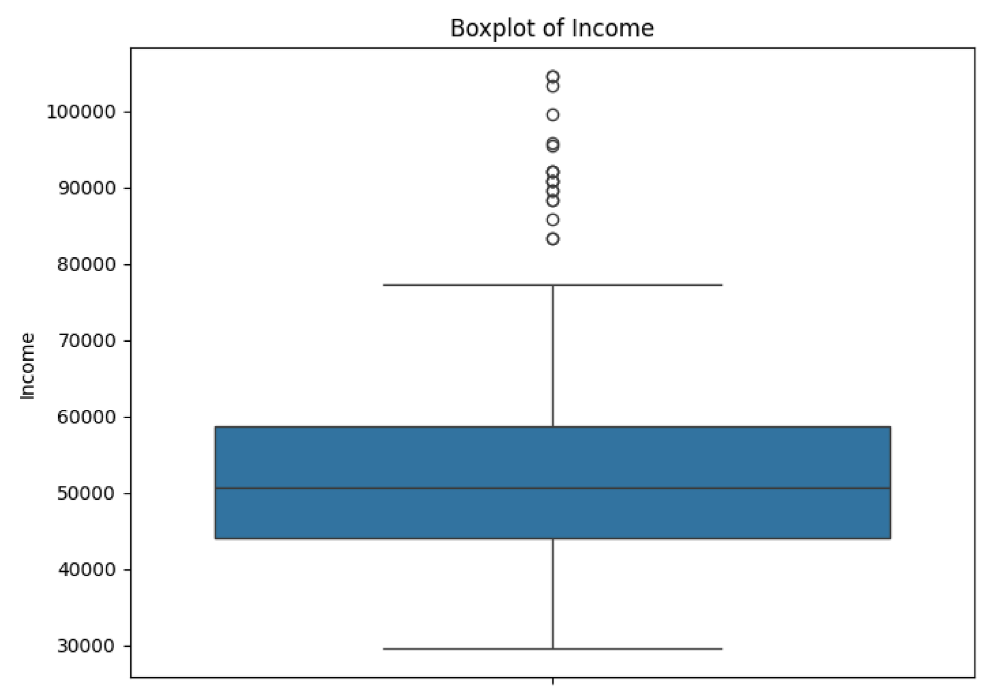
```
for column in continuous_vars.columns:
    plt.figure(figsize=(8, 6))
    sns.boxplot(data=continuous_vars[column])
    plt.title(f'Boxplot of {column}')
    plt.show()
```

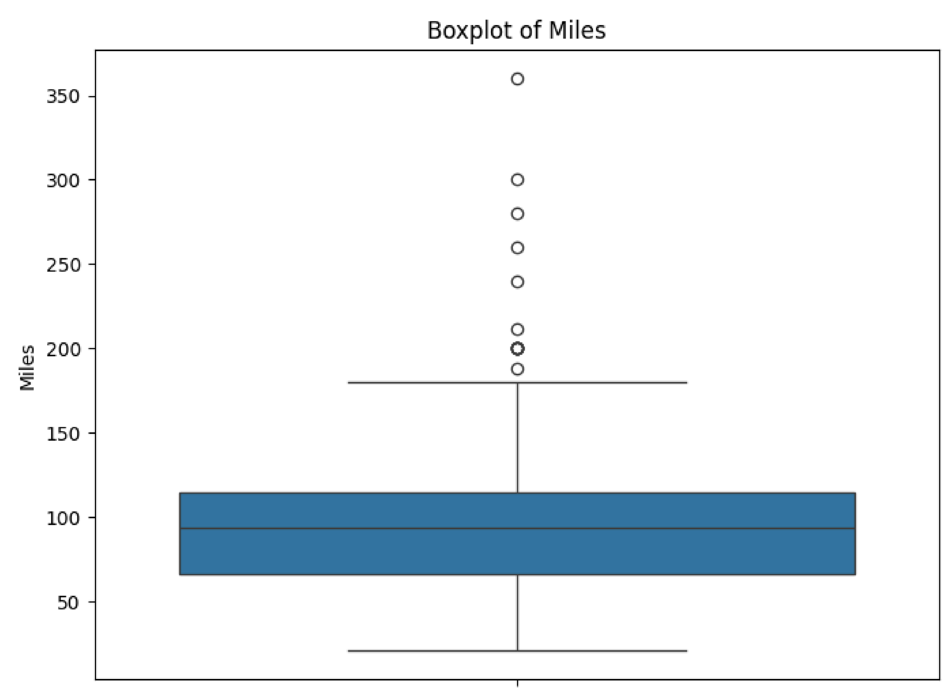












- boxplot is used here to show the outliers in the data set
- we see that Age, Education, Usage, Fitness have no outliers
- While Income and Miles are having more outliers.

```
customer
```

	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

```
continuous_vars = customer.select_dtypes(include=['int64', 'float64'])
```

```
# Remove/Clip the data between the 5th and 95th percentiles
for col in continuous_vars.columns:
    customer[col] = np.clip(customer[col], customer[col].quantile(0.05), customer[col].quantile(0.95))

# Display the updated dataset
print(customer)
```

```
    Product    Age  Gender  Education MaritalStatus  Usage  Fitness    Income  \
0     KP281  20.00    Male         14        Single   3.00        4  34053.15
```

```
1     KP281  20.00    Male         15       Single   2.00        3  34053.15
2     KP281  20.00  Female         14    Partnered   4.00        3  34053.15
3     KP281  20.00    Male         14       Single   3.00        3  34053.15
4     KP281  20.00    Male         14    Partnered   4.00        2  35247.00
..      ...    ...     ...        ...          ...    ...      ...       ...
175   KP781  40.00    Male         18       Single   5.05        5  83416.00
176   KP781  42.00    Male         18       Single   5.00        4  89641.00
177   KP781  43.05    Male         16       Single   5.00        5  90886.00
178   KP781  43.05    Male         18    Partnered   4.00        5  90948.25
179   KP781  43.05    Male         18    Partnered   4.00        5  90948.25

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

[180 rows x 9 columns]
```

clipping the data between the 5th and 9th percentile

```
customer
```

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
0	KP281	20.00	Male	14	Single	3.00	4	34053.15	112
1	KP281	20.00	Male	15	Single	2.00	3	34053.15	75
2	KP281	20.00	Female	14	Partnered	4.00	3	34053.15	66
3	KP281	20.00	Male	14	Single	3.00	3	34053.15	85
4	KP281	20.00	Male	14	Partnered	4.00	2	35247.00	47
...	...	...	...	...	...	...	...	...	...
175	KP781	40.00	Male	18	Single	5.05	5	83416.00	200
176	KP781	42.00	Male	18	Single	5.00	4	89641.00	200
177	KP781	43.05	Male	16	Single	5.00	5	90886.00	160
178	KP781	43.05	Male	18	Partnered	4.00	5	90948.25	120
179	KP781	43.05	Male	18	Partnered	4.00	5	90948.25	180

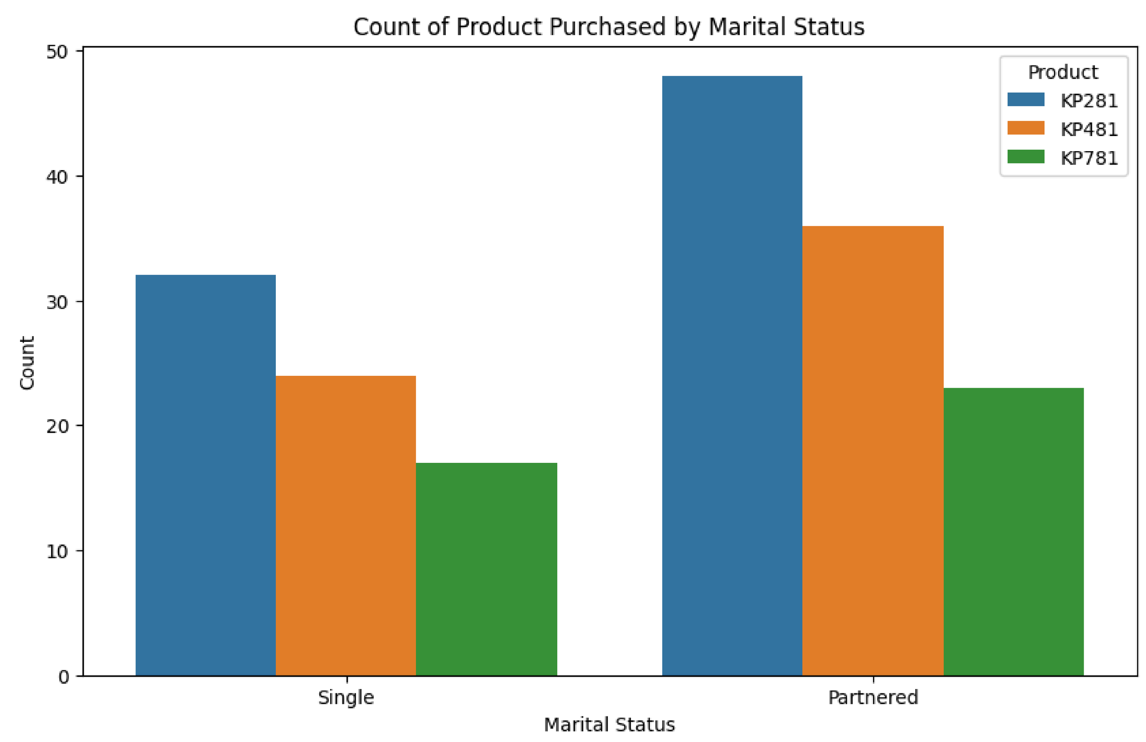
180 rows × 9 columns

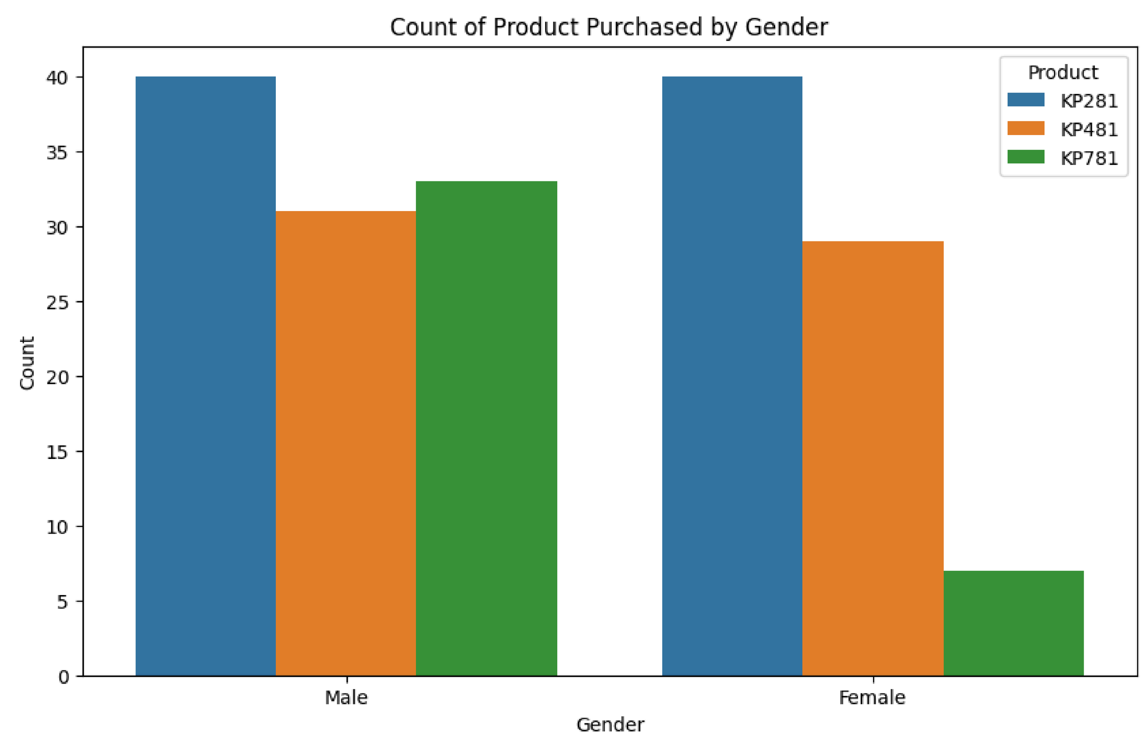
3. Check if features like marital status, Gender, and age have any effect on the product purchased

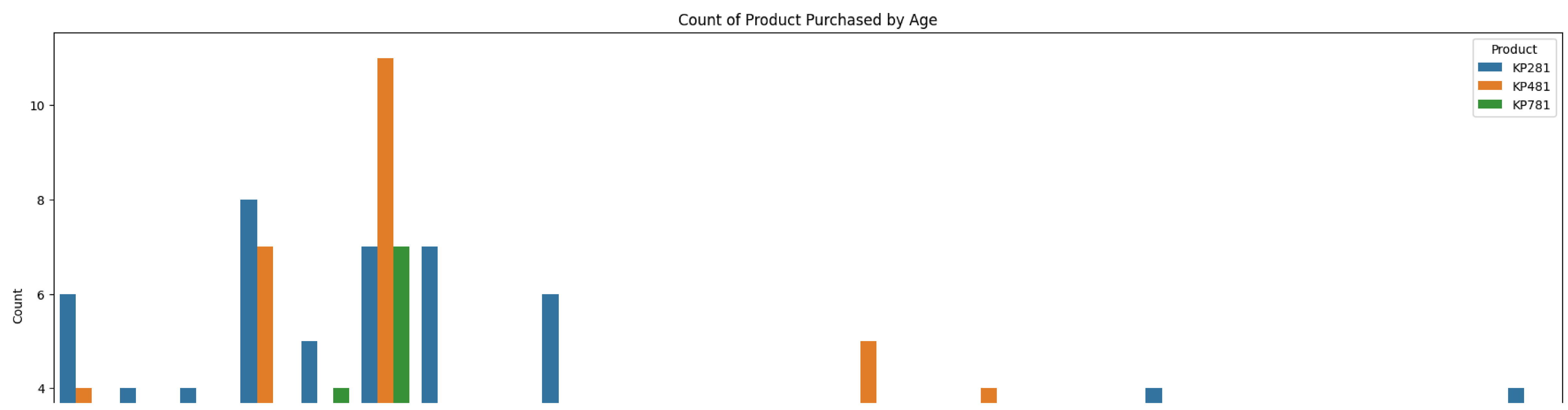
```
import pandas as pd

# Assuming your DataFrame is named df and the Usage column exists
filtered_df = customer[(customer['Usage'] > 2) & (customer['Usage'] < 3)]

# Display the filtered DataFrame
print(filtered_df)
```

```
Empty DataFrame
Columns: [Product, Age, Gender, Education, MaritalStatus, Usage, Fitness, Income, Miles]
Index: []
```

```
import seaborn as sns
import matplotlib.pyplot as plt

# Count plot for marital status vs. product purchased
plt.figure(figsize=(10, 6))
sns.countplot(x='MaritalStatus', hue='Product', data=customer)
plt.title('Count of Product Purchased by Marital Status')
plt.xlabel('Marital Status')
plt.ylabel('Count')
plt.show()

# Count plot for gender vs. product purchased
plt.figure(figsize=(10, 6))
sns.countplot(x='Gender', hue='Product', data=customer)
plt.title('Count of Product Purchased by Gender')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.show()

# Count plot for age vs. product purchased (you may need to adjust bins based on your data)
plt.figure(figsize=(22, 8))
sns.countplot(x='Age', hue='Product', data=customer)
plt.title('Count of Product Purchased by Age')
plt.xlabel('Age')
plt.ylabel('Count')
plt.show()
```



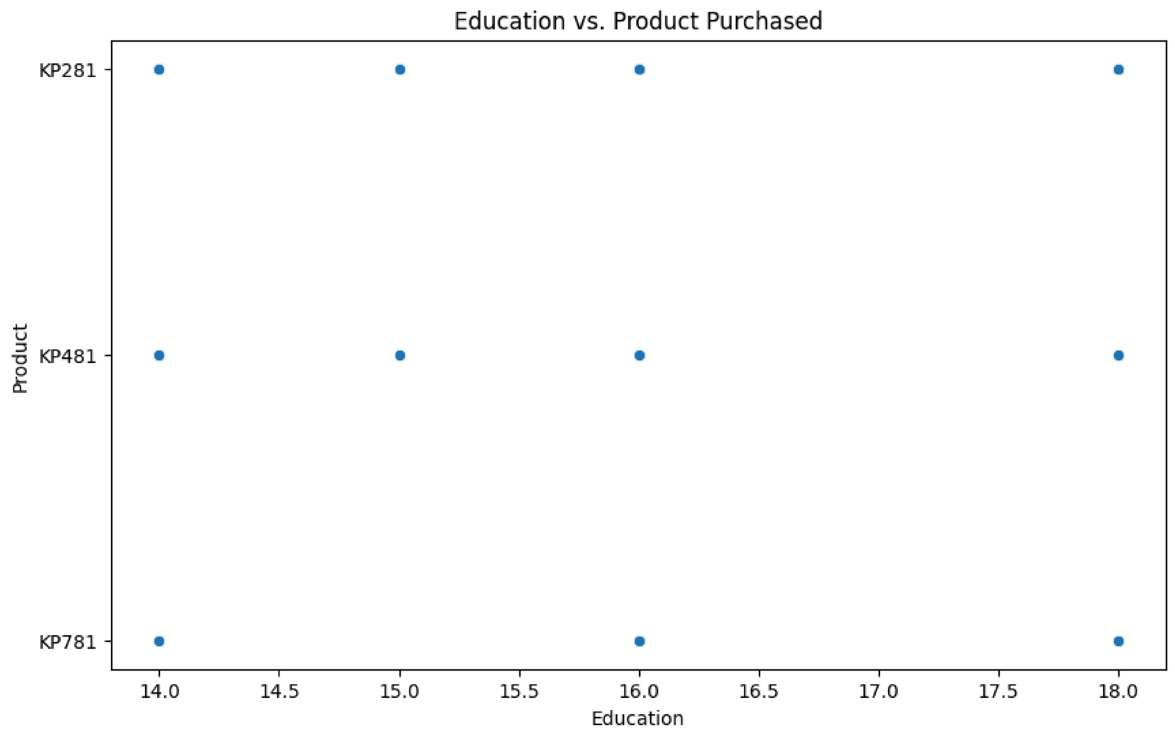


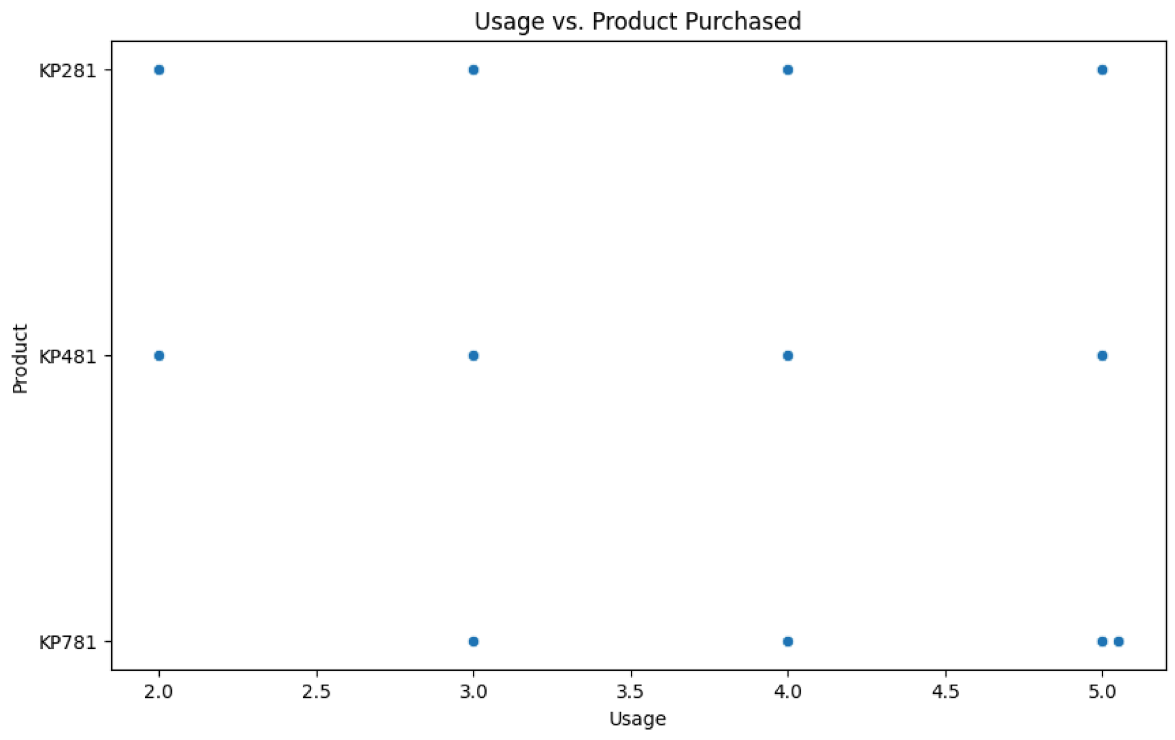


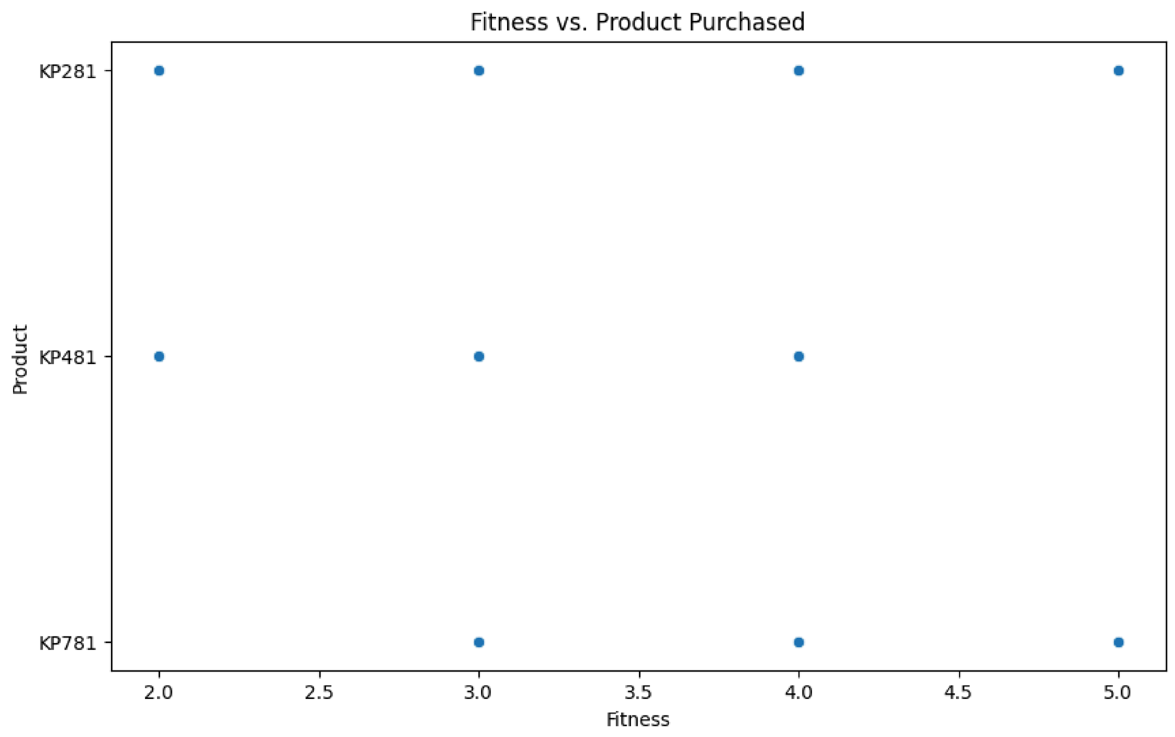
- most of the products are purchased by the maried couples.
- most of the products are purchased by male.
- We can clearly see that the majority of kp481 product purchases are made by individuals aged 25. And there are significant fluctuations in the ages of people purchasing the products.

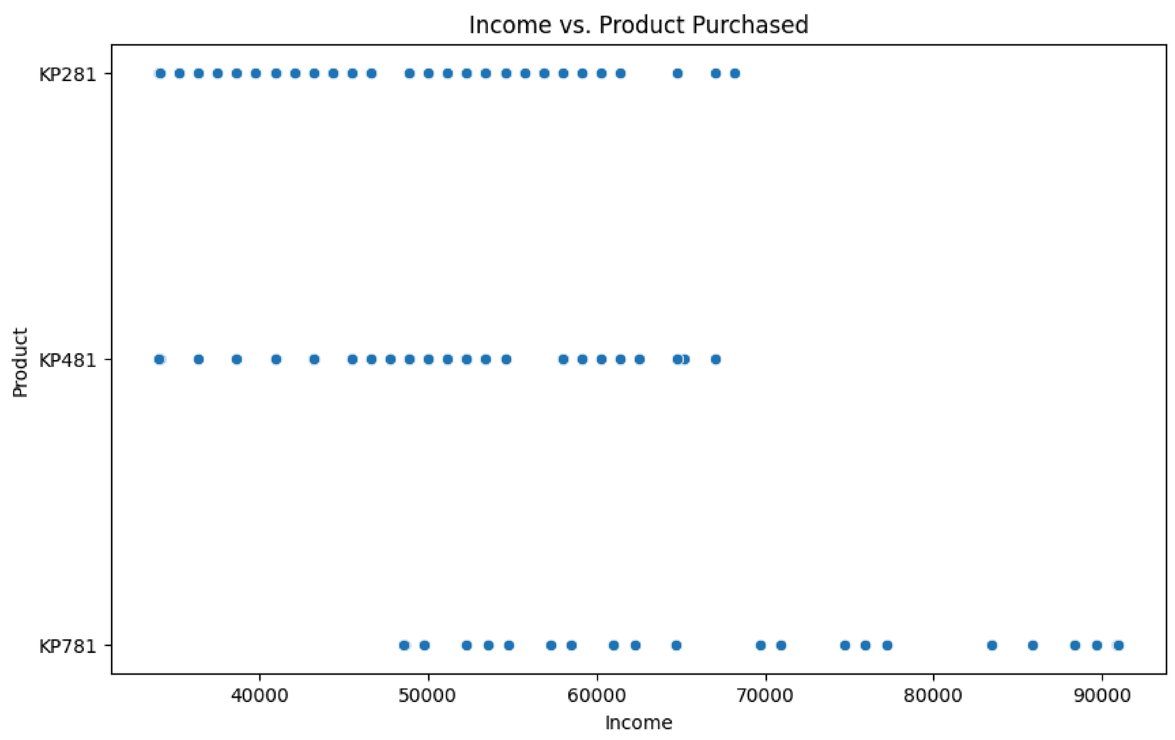
```
# Scatter plot for education vs. product purchased
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Education', y='Product', data=customer)
plt.title('Education vs. Product Purchased')
plt.xlabel('Education')
plt.ylabel('Product')
plt.show()

# Scatter plot for usage vs. product purchased
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Usage', y='Product', data=customer)
plt.title('Usage vs. Product Purchased')
plt.xlabel('Usage')
plt.ylabel('Product')
plt.show()

# Scatter plot for fitness vs. product purchased
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Fitness', y='Product', data=customer)
plt.title('Fitness vs. Product Purchased')
plt.xlabel('Fitness')
plt.ylabel('Product')
```

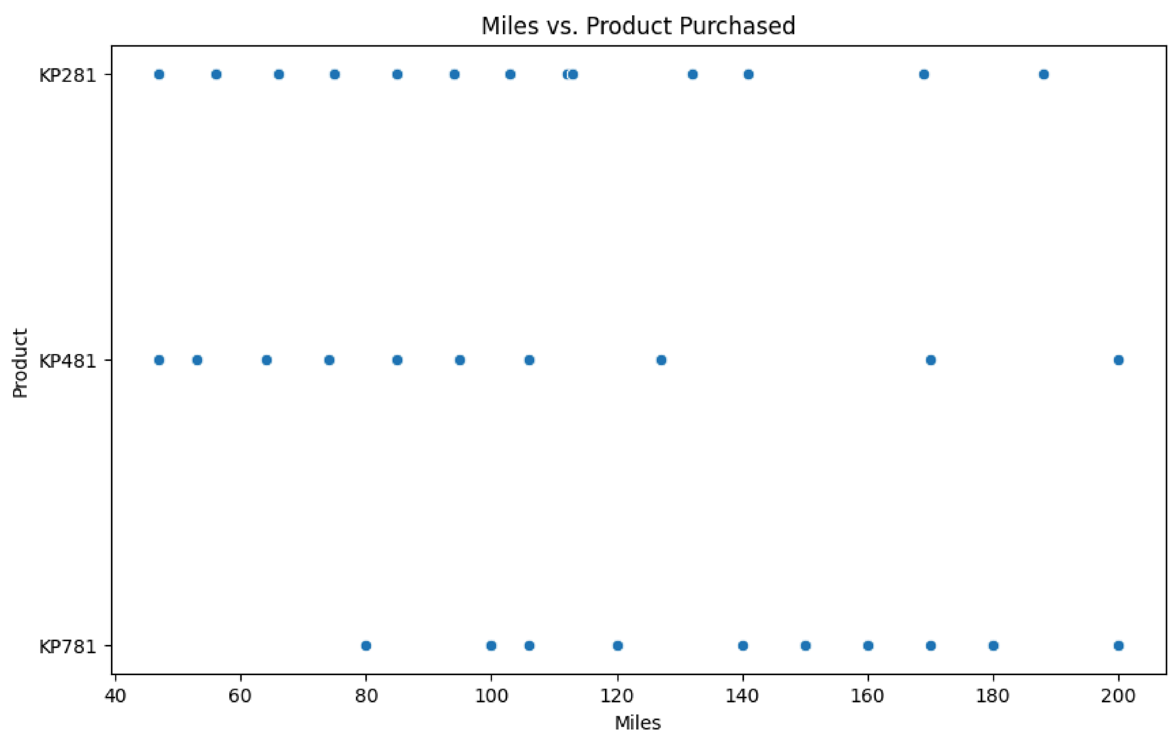
```
plt.show()

# Scatter plot for income vs. product purchased
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Income', y='Product', data=customer)
plt.title('Income vs. Product Purchased')
plt.xlabel('Income')
plt.ylabel('Product')
plt.show()

# Scatter plot for miles vs. product purchased
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Miles', y='Product', data=customer)
plt.title('Miles vs. Product Purchased')
plt.xlabel('Miles')
plt.ylabel('Product')
plt.show()
```











- We observe that individuals with less than or equal to 16 years of education make more purchases than those with education levels above 16 years.
- Customers who are planning to use the treadmill greater than or equal to 5 times a week, are more likely to purchase the KP781 product. While the other customers are likely to purchasing KP281 or KP481.
- The people who are fit (>=4.0) has higher chance of purchasing the KP781 and KP281 and the others have the higher chances of purchasing KP281 and KP481.
- The people with the salary 40k-70k have higher purchases of KP481 and KP281. The people with the salary above 70k have purchases only of KP781.
- The people who walk 120-200 miles have higher purchases of KP781 than the other. where has the people who walk less than 120 miles have higher purchases of KP481 and KP281.

```
# Marginal Probability
marginal_prob = pd.crosstab(index=customer['Product'], columns='count', normalize=True)

# Probability of Buying a Product Based on Each Column
prob_gender = pd.crosstab(index=customer['Product'], columns=customer['Gender'], normalize='index')
prob_marital_status = pd.crosstab(index=customer['Product'], columns=customer['MaritalStatus'], normalize='index')

# Conditional Probability
cond_prob_female = prob_gender.loc[:, 'Female']
cond_prob_KP481_given_female = prob_marital_status.loc['KP481', 'Single'] * cond_prob_female['KP481']

# Display results
print("Marginal Probability of Each Product:")
print(marginal_prob)
print("\nProbability of Buying a Product Based on Gender:")
print(prob_gender)
print("\nProbability of Buying a Product Based on Marital Status:")
print(prob_marital_status)
print("\nConditional Probability (given that a customer is female, what is the probability she'll purchase KP481):")
print(cond_prob_KP481_given_female)
```

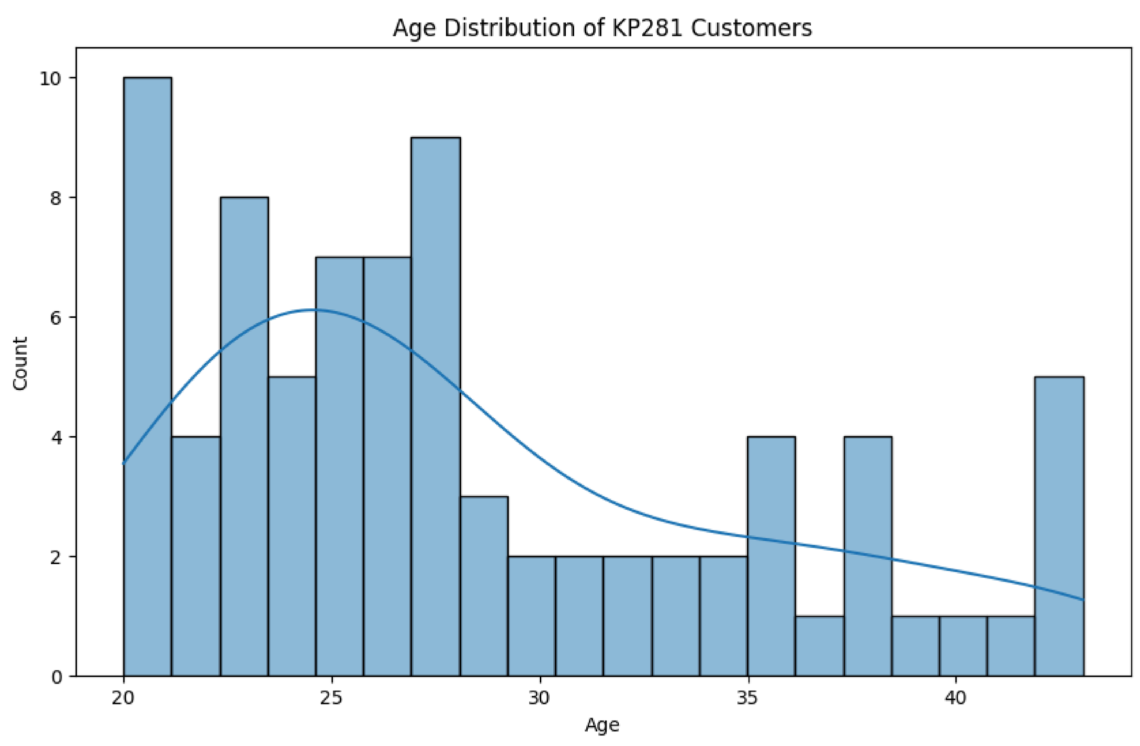
```
Marginal Probability of Each Product:
col_0       count
Product
KP281    0.444444
KP481    0.333333
KP781    0.222222
```

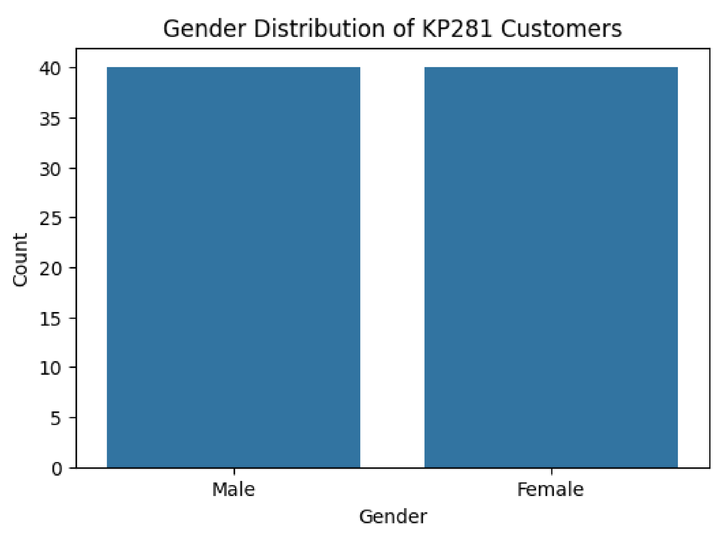
```
Probability of Buying a Product Based on Gender:
Gender     Female      Male
Product
KP281    0.500000  0.500000
KP481    0.483333  0.516667
KP781    0.175000  0.825000

Probability of Buying a Product Based on Marital Status:
MaritalStatus  Partnered  Single
Product
KP281              0.600   0.400
KP481              0.600   0.400
KP781              0.575   0.425

Conditional Probability (given that a customer is female, what is the probability she'll purchase KP481):
0.19333333333333336
```

```
import seaborn as sns
import matplotlib.pyplot as plt

# Select only numeric columns for correlation calculation
numeric_customer = customer.select_dtypes(include=['int64', 'float64'])

# Compute the correlation matrix
correlation_matrix = numeric_customer.corr()

# Visualize the correlation matrix using a heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=.5)
plt.title('Correlation Heatmap')
plt.show()
```

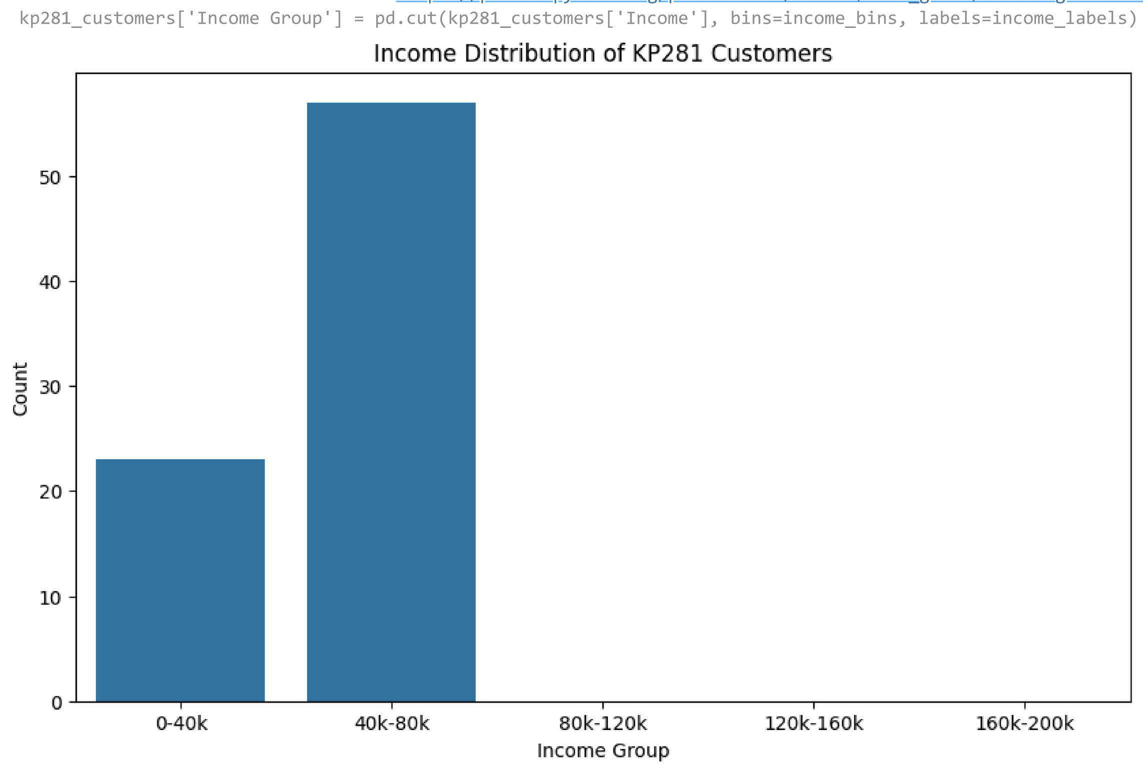
- The above heat map clearly shows that each of these variables has a perfect positive correlation with itself.
- A correlation coefficient of 0.83 indicates a strong positive correlation between Miles and Fitness. This means that as the number of miles a person runs or walks increases, their fitness level also tends to increase proportionally, and vice versa.
- Correlation coefficients of 0.03 (between Age and Miles) and 0.02 (between Age and Usage) indicate very weak or negligible correlations between these variables. This means that there is little to no linear relationship between the variables.

```
import matplotlib.pyplot as plt
import seaborn as sns

# Filter data for customers who purchased KP281
kp281_customers = customer[customer['Product'] == 'KP281']

# Customer profiling based on age
plt.figure(figsize=(10, 6))
sns.histplot(kp281_customers['Age'], bins=20, kde=True)
plt.title('Age Distribution of KP281 Customers')
plt.xlabel('Age')
plt.ylabel('Count')
plt.show()

# Customer profiling based on gender
plt.figure(figsize=(6, 4))
sns.countplot(x='Gender', data=kp281_customers)
plt.title('Gender Distribution of KP281 Customers')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.show()

# Customer profiling based on income group (you can define income groups based on your dataset)
income_bins = [0, 40000, 80000, 120000, 160000, 200000]
income_labels = ['0-40k', '40k-80k', '80k-120k', '120k-160k', '160k-200k']
kp281_customers['Income Group'] = pd.cut(kp281_customers['Income'], bins=income_bins, labels=income_labels)
plt.figure(figsize=(10, 6))
sns.countplot(x='Income Group', data=kp281_customers, order=income_labels)
plt.title('Income Distribution of KP281 Customers')
plt.xlabel('Income Group')
plt.ylabel('Count')
plt.show()
```






```
<ipython-input-21-5915fbf42aa9>:26: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  kp281_customers['Income Group'] = pd.cut(kp281_customers['Income'], bins=income_bins, labels=income_labels)
```



- We observe that customers between the ages of 20 and 28 make more purchases of KP281 compared to customers above the age of 28.
- We see that both females and males have an equal number of purchases of KP281.
- We observe that the majority of purchases are made by individuals with salaries ranging from 40k to 80k, and there are no orders made by individuals with salaries above 80k.

FINAL RECOMENDATION

1. For the female customers KP281 should be the first recomendation

2. Encourage customers who plan to use the treadmill at least 3-4 times a week to consider purchasing KP781. However, it's important to provide guidance on proper usage and fitness routines to ensure customer satisfaction and safety.

3. Target individuals with a fitness level of 3 or above, as they are more likely to benefit from and utilize the features of KP781 effectively. Consider providing additional resources or support to help customers improve their fitness levels if needed.

4. Since most customers who bought KP781 are male, consider targeting marketing efforts towards males. However, it's essential not to exclude females entirely, as they still represent a portion of the customer base.

HOWEVER THSESE ARE THE KEY RECOMENDATIONS

```
customer.to_csv('aerofit_cleaned.csv', index=False)
```

```
                                                    **THANK YOU**
```