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.

Business Problem

The market research team at AeroFit wants to identify the 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 decides to investigate whether there are differences across the product with respect to customer characteristics.

Importing libraries

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

DATASET:

```
data=pd.read_csv("https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/125/original/aerofit_treadmill.csv?1639992749")
data
```

	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

```
data.head() #Basic metrics of a dataset
```

	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

```
data.shape
```

```
(180, 9)
```

The Aerofit data has 180 rows and 9 columns

```
data.ndim
```

```
2
```

The Aerofit data is 2 dimensional.

```
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
```

Age, Education, Usage, fitness, Income, Miles are of **int** datatype. Product, Gender, maritialStatus are of **Object** type. Also, all values are non null.

```
data.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

MISSING VALUES

```
data.isnull().sum()
```

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

From the above values we can see that there are no null values.

UNIVARIANT PLOTS

```
ProductCount = data["Product"].value_counts()
GenderCount = data['Gender'].value_counts()
MStatusCount = data['MaritalStatus'].value_counts()
```

Double-click (or enter) to edit

```
ProductCount
```

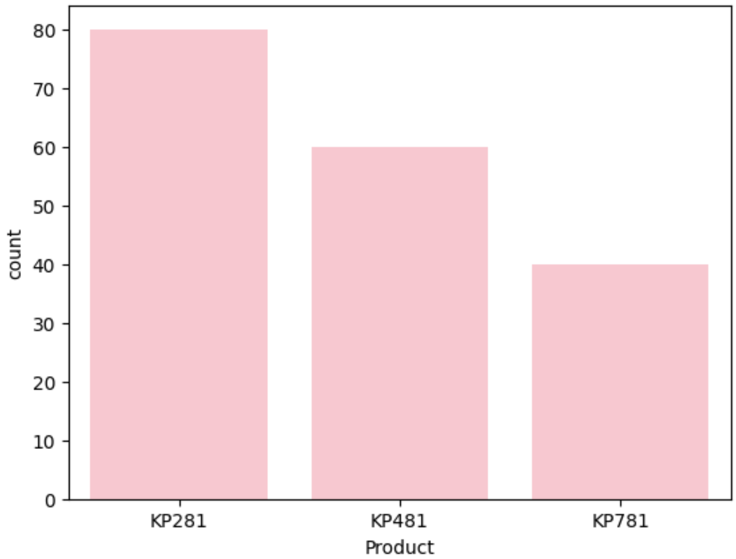
```
KP281    80
KP481    60
KP781    40
Name: Product, dtype: int64
```

```
GenderCount
```

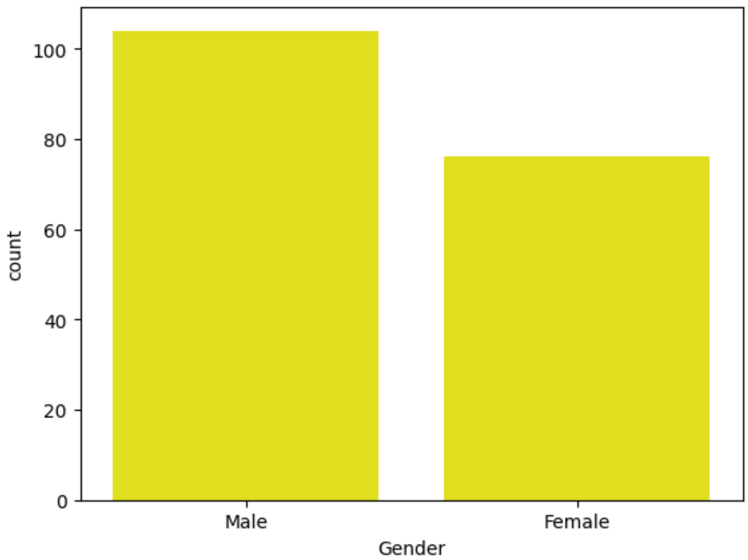
```
Male      104
Female     76
Name: Gender, dtype: int64
```

```
Partnered    107
Single        73
Name: MaritalStatus, dtype: int64
```

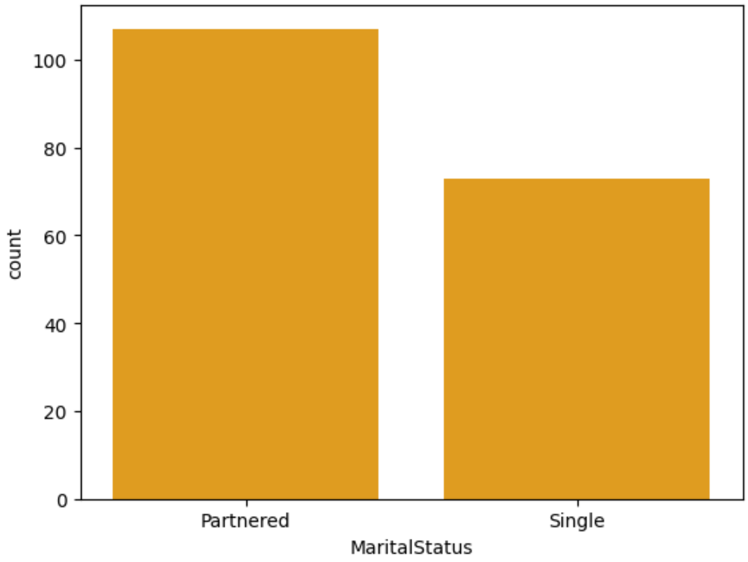
```
productplot= sns.countplot(x="Product", order=ProductCount.index,  data=data, color= "pink")
```




```
genderplot= sns.countplot(x="Gender", order=GenderCount.index, data=data, color= "yellow")
```



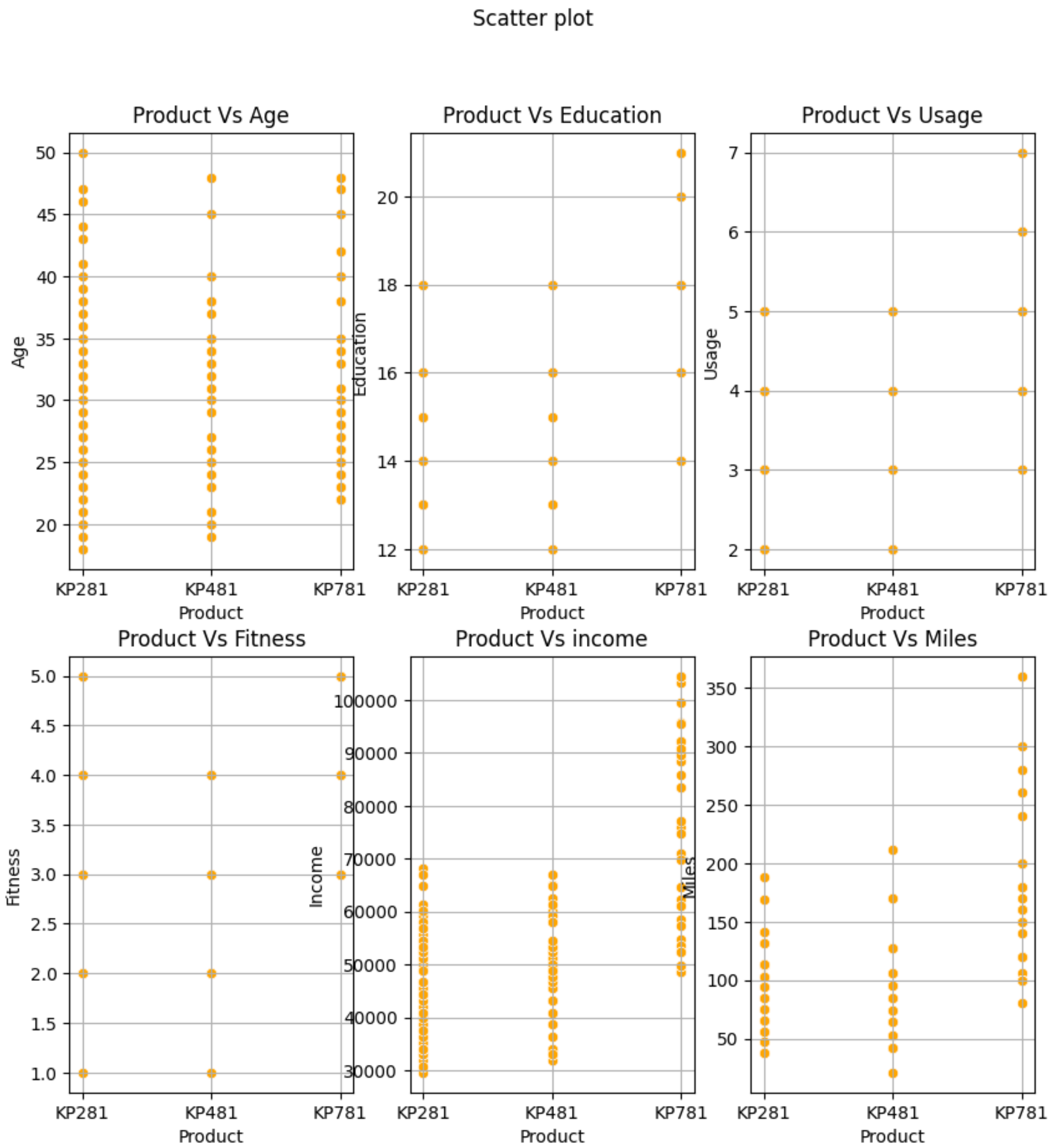

```
statusplot= sns.countplot(x="MaritalStatus", order=MStatusCount .index, data=data, color= "orange")
```



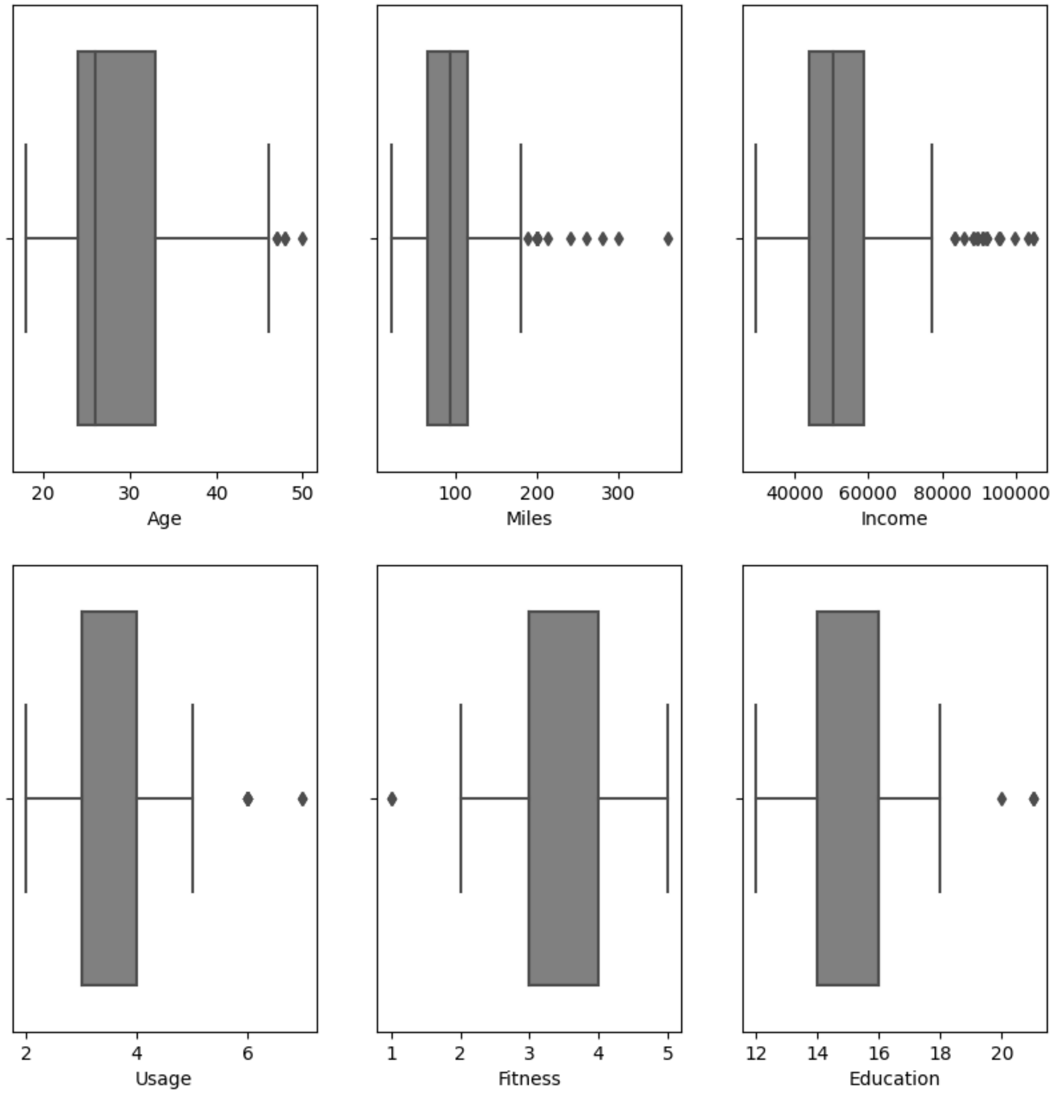
BIVARIANT PLOTS

PRODUCT COMPARISON USING SCATTER PLOTS

```
fig = plt.figure(figsize=(10, 10))

plt.subplot(2, 3, 1) # it divides figure into 1 row, 3 columns
sns.scatterplot(x="Product", y="Age", data=data, color='orange')
plt.title("Product Vs Age")
plt.grid()


plt.subplot(2, 3, 2)
sns.scatterplot(x="Product" ,y="Education", data=data, color='orange')
plt.title("Product Vs Education")
plt.grid()


plt.subplot(2, 3, 3)
sns.scatterplot(x="Product", y="Usage", data=data, color='orange')
plt.title("Product Vs Usage")
plt.grid()

plt.subplot(2, 3, 4)
sns.scatterplot(x="Product", y="Fitness", data=data, color='orange')
plt.title("Product Vs Fitness")
plt.grid()


plt.subplot(2, 3, 5)
sns.scatterplot(x="Product", y="Income", data=data, color='orange')
plt.title("Product Vs income")
plt.grid()

plt.subplot(2,3,6)
sns.scatterplot(x="Product", y="Miles",data= data, color= "orange")
plt.title("Product Vs Miles")
plt.grid()


fig.suptitle('Scatter plot ')
plt.show()
```




```
fig = plt.figure(figsize=(10,10))

plt.subplot(2,3,1)
sns.boxplot(data=data, x="Age", color= 'gray' )

plt.subplot(2,3,2)
sns.boxplot(data=data, x= "Miles", color= 'gray' )

plt.subplot(2,3,3)
sns.boxplot(data=data, x="Income", color= 'gray' )

plt.subplot(2,3,4)
sns.boxplot(data=data, x="Usage", color= 'gray' )

plt.subplot (2,3,5)
sns.boxplot(data=data, x="Fitness", color= 'gray' )

plt.subplot(2,3,6)
sns.boxplot(data=data, x="Education", color= 'gray' )


plt.show()
```



Age Distribution :

The boxplot for age provides insights into the central tendency, spread, and presence of outliers in the age distribution. Check for the median, interquartile range (IQR), and the presence of any extreme values. Miles Distribution :

The boxplot for miles gives information about the spread of the running distances customers are willing to cover. Look for differences in medians and variations in the spread among different mileage ranges. Income Distribution :

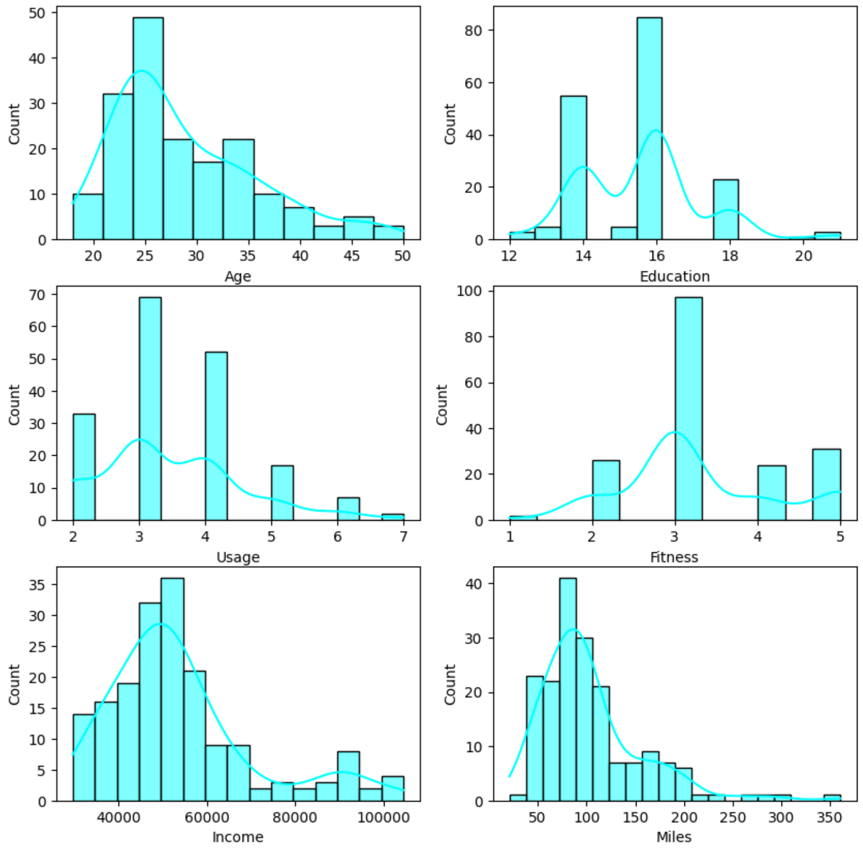
The boxplot for income visualizes the central tendency and variability in income levels. Identify any outliers and understand the income distribution among your customers. Usage Distribution:

The boxplot for usage provides insights into how frequently products are used. Look for differences in medians and variations in usage patterns among different groups. Fitness Distribution):

If uncommented, this subplot would visualize the distribution of fitness levels. Similar to other boxplots, it would show the central tendency, spread, and potential outliers in the fitness distribution. Education Distribution :

The boxplot for education levels gives insights into the central tendency and variability in the education distribution. Check for differences in medians and variations in education levels among different groups.

```
fig, axis = plt.subplots(3,2, figsize=(10,10))

sns.histplot(data=data, x="Age", kde=True, color='cyan',ax=axis[0,0])
sns.histplot(data=data, x="Education", kde=True, color='cyan', ax=axis[0,1])
sns.histplot(data=data, x="Usage", kde=True, color='cyan', ax=axis[1,0])
sns.histplot(data=data, x="Fitness", kde=True, color='cyan', ax=axis[1,1])
sns.histplot(data=data, x="Income", kde=True, color='cyan',ax=axis[2,0])
sns.histplot(data=data, x="Miles", kde=True, color='cyan', ax=axis[2,1])
plt.show()
```



This code is creating a 3x2 grid of histograms using Seaborn to visualize the distribution of different features in your dataset. Let's look at potential insights for each subplot:

Age Distribution (axis[0,0]): The histogram for age shows the frequency distribution of ages in your dataset. Check for peaks, gaps, or patterns to understand the age distribution of your target audience.

Education Distribution (axis[0,1]): This subplot visualizes the distribution of education levels. Look for the most common education levels and assess whether your dataset is diverse or concentrated in terms of education.

Usage Distribution (axis[1,0]): The histogram for usage provides insights into how frequently products are used. Peaks or specific patterns can indicate common usage patterns among your customers.

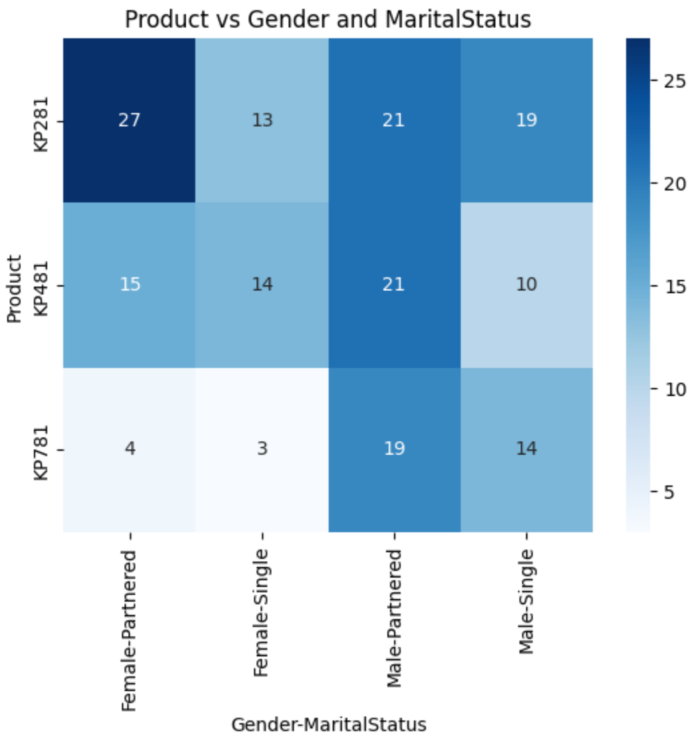
Fitness Distribution (axis[1,1]): This subplot shows the distribution of fitness levels. Identify the predominant fitness levels in your dataset, as this information could be relevant for fitness-related products or services.

Income Distribution (axis[2,0]): The histogram for income displays the income distribution of your dataset. Identify income ranges that are more common and assess the overall income diversity among your customers.

Miles Distribution (axis[2,1]): The histogram for miles indicates the distribution of the number of miles customers are willing to run. Look for peaks or patterns to understand the preferences or capabilities of your target audience in terms of running distances.

Correlation Using heatmap

```
data_pivot = data.pivot_table (data, index='Product', columns=['Gender', 'MaritalStatus'], aggfunc='size')

sns.heatmap(data_pivot, cmap='Blues', annot=True)

plt.title('Product vs Gender and MaritalStatus')
plt.show()
```



This code seems to be creating a heatmap using a pivot table from a dataset, visualizing the relationship between "Product," "Gender," and "MaritalStatus." The color intensity represents the size of each combination. Here are some potential insights:

Product Preferences: You can observe which products are more popular among different gender and marital status groups. Look for darker areas, indicating larger sizes, to identify the most popular combinations.

Market Segmentation: Check for clusters or patterns in the heatmap. Certain products might be preferred by specific gender-marital status demographics. This information can help in targeted marketing strategies.

Blank Spaces: If there are missing values in the heatmap, it suggests that certain combinations of Product, Gender, and MaritalStatus are not present in the dataset. This could be useful for understanding gaps in your data.

Outliers: Unusually large or small values might stand out. These could be outliers that deserve further investigation.

Marginial And Conditional Probability

Marginal probability is a fundamental concept in probability theory and statistics. It refers to the probability of a single event occurring, without considering any other events. In other words, it focuses on the probability of one variable or outcome, while ignoring the influence of other variables.

Conditional probability in probability theory that measures the likelihood of an event occurring given that another event has already occurred. It allows us to update our knowledge or beliefs about an event based on new information.

In mathematical terms, the conditional probability of event A given event B is denoted as P(A|B), and it is calculated as the probability of both events A and B occurring divided by the probability of event B occurring.

```
data["Product"].value_counts(normalize=True)
```

```
KP281    0.444444
KP481    0.333333
KP781    0.222222
Name: Product, dtype: float64
```

The product "KP281" has the highest normalized count with a relative frequency of approximately 44.44%. This suggests that "KP281" is the most prevalent product in your dataset. "KP281" not only is the dominant product but also has a significantly higher market share compared to others. "KP481" follows with a normalized count of around 33.33%, and "KP781" has the lowest market share at approximately 22.22%.

```
product_counts = data.groupby(['Gender', 'Product']).size().reset_index(name='Count')
gender_counts = data['Gender'].value_counts().reset_index()
gender_counts.columns = ['Gender', 'Total Count']

merged_data = pd.merge(product_counts, gender_counts, on='Gender')

merged_data['Probability'] = merged_data['Count'] / merged_data['Total Count']

male_probabilities = merged_data[merged_data['Gender'] == 'Male'][['Product', 'Probability']]

female_probabilities = merged_data[merged_data['Gender'] == 'Female'][['Product', 'Probability']]

print("Male probabilities:\n", male_probabilities)

print("Female probabilities:\n", female_probabilities)
```

```
Male probabilities:
   Product  Probability
3   KP281     0.384615
4   KP481     0.298077
5   KP781     0.317308
Female probabilities:
   Product  Probability
0   KP281     0.526316
1   KP481     0.381579
2   KP781     0.092105
```

Gender-Specific Probabilities: The code calculates probabilities for each product based on gender. For both males and females, it's determining the likelihood (probability) of each product being associated with that gender.

Product Preferences: Look at the probabilities for each product in the output. Higher probabilities indicate a stronger association between a product and a specific gender. This can provide insights into gender-based product preferences.

Gender Dominance for Products: Check if certain products have significantly higher probabilities for one gender over the other. This information can be useful for targeted marketing or product development tailored to specific gender preferences.

Overall Distribution: Consider the overall distribution of probabilities. Are there products that are universally popular regardless of gender? Or are there distinct gender preferences for different products?

** CONTINGENCY TABLES**

Contingency tables, also known as cross-tabulation tables or crosstabs, are a statistical tool used to analyze the relationship between two categorical variables. They provide a way to summarize and display the distribution of data across different categories.

In a contingency table, the rows represent one categorical variable, and the columns represent another categorical variable. The cells of the table contain the frequencies or counts of observations that fall into each combination of categories.

To create a contingency table in Python, you can use the crosstab function from the pandas library. This function takes two or more categorical variables as input and returns a table with the frequencies or counts of observations for each combination of categories.

Double-click (or enter) to edit

```
Crosstab=pd.crosstab(data.Product, data.Gender)
Crosstab2= pd.crosstab(data.MaritalStatus, data.Miles)
Crosstab4=pd.crosstab(data.Gender, data.Fitness)
```

```
Crosstab
```

Gender	Female	Male
Product		
KP281	40	40
KP481	29	31
KP781	7	33

```
Crosstab2
```

Miles	21	38	42	47	53	56	64	66	74	75	...	170	180	188	200	212	240	260	280	300	360
MaritalStatus																					

Crosstab4

Fitness	1	2	3	4	5
Gender					
Female	1	16	45	8	6
Male	1	10	52	16	25

Recommendations:

1.Age Group:

The majority of treadmill users fall within the age range of 20 to 30 years.

So, it would be beneficial to focus marketing efforts on this age group by highlighting features and benefits that appeal to their specific needs and preferences.

2.Income:

Customers with an income range of 50 to 60 thousand dollars are more likely to purchase treadmills.

This suggests that pricing strategies should be tailored to this income bracket, offering competitive pricing options and financing plans to make the product more accessible.

3. Marital Status:

Partnered individuals are more likely to buy treadmill products.

This indicates that marketing campaigns could emphasize the benefits of exercising together as a couple or family, promoting the treadmill as a shared activity.

4.Income and Product Choice:

Customers with higher incomes (income >= 60000) are more inclined to purchase the KP781 treadmill.