Aerofit Business Case Study

```
!wget    https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/125/original/aerofit_treadmill.csv?1639992749
```

```
--2024-10-20 12:43:28--  https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/125/original/aerofit_treadmill.csv?1639992749
Resolving d2beiqkhq929f0.cloudfront.net (d2beiqkhq929f0.cloudfront.net)... 18.64.229.91, 18.64.229.135, 18.64.229.71, ...
Connecting to d2beiqkhq929f0.cloudfront.net (d2beiqkhq929f0.cloudfront.net)|18.64.229.91|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 7279 (7.1K) [text/plain]
Saving to: ‘aerofit_treadmill.csv?1639992749’

aerofit_treadmill.c 100%[===================>]    7.11K  --.-KB/s    in 0s

2024-10-20 12:43:29 (123 MB/s) - ‘aerofit_treadmill.csv?1639992749’ saved [7279/7279]
```

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

```
df=pd.read_csv("aerofit_treadmill.csv?1639992749")
```

```
df.head()
```

	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

Next steps: Generate code with df | View recommended plots | New interactive sheet

#Checking the structure & characteristics of the dataset

```
#The data type of all columns in the table.
df.dtypes
```

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

dtype: object

```
#number of rows and columns given in the dataset
df.shape
```

```
(180, 9)
```

```
Start coding or ge_nerate with AI.
```

#deal with missing values

```
df.isna().sum()
```

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

dtype: int64

```
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
```

```
#Changing data types of usage and fitness columns
df["Usage"]=df["Usage"].astype("str")
df["Fitness"]=df["Fitness"].astype("str")
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    object
 6   Fitness         180 non-null    object
 7   Income          180 non-null    int64 
 8   Miles           180 non-null    int64 
dtypes: int64(4), object(5)
memory usage: 12.8+ KB
```

#Statastical Sunnary

```
#statastical summary of object types columns
df.describe(include="object")
```

	Product	Gender	MaritalStatus	Usage	Fitness
count	180	180	180	180	180
unique	3	2	2	6	5
top	KP281	Male	Partnered	3	3
freq	80	104	107	69	97

```
df.groupby("Fitness").count()
```

Fitness	Product	Age	Gender	Education	MaritalStatus	Usage	Income	Miles
1	2	2	2	2	2	2	2	2
2	26	26	26	26	26	26	26	26
3	97	97	97	97	97	97	97	97
4	24	24	24	24	24	24	24	24
5	31	31	31	31	31	31	31	31

```
df.groupby("MaritalStatus").count()
```

MaritalStatus	Product	Age	Gender	Education	Usage	Fitness	Income	Miles
Partnered	107	107	107	107	107	107	107	107
Single	73	73	73	73	73	73	73	73

```
df.groupby("Gender").count()
```

Gender	Product	Age	Education	MaritalStatus	Usage	Fitness	Income	Miles
Female	76	76	76	76	76	76	76	76
Male	104	104	104	104	104	104	104	104

```
df.groupby("Product").count()
```

Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
KP281	80	80	80	80	80	80	80	80
KP481	60	60	60	60	60	60	60	60
KP781	40	40	40	40	40	40	40	40

```
#statastical summary of numerical datatype
df.describe()
```

	Age	Education	Income	Miles
count	180.000000	180.000000	180.000000	180.000000
mean	28.788889	15.572222	53719.577778	103.194444
std	6.943498	1.617055	16506.684226	51.863605
min	18.000000	12.000000	29562.000000	21.000000
25%	24.000000	14.000000	44058.750000	66.000000
50%	26.000000	16.000000	50596.500000	94.000000
75%	33.000000	16.000000	58668.000000	114.750000
max	50.000000	21.000000	104581.000000	360.000000

```
#detect duplicates
df.duplicated().value_counts()
```

	count
False	180

dtype: int64

```
Start coding or ge_nerate with AI.
```

#Non graphical analysis valu counts and unique attributes

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

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

```
round(df["Product"].value_counts(normalize=True)*100,2)
```

Product	proportion
KP281	44.44
KP481	33.33
KP781	22.22

dtype: float64

```
df["Age"].unique()
```

```
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])
```

```
df["Gender"].unique()
```

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

```
round(df["Gender"].value_counts(normalize=True)*100,2)
```

Gender	proportion
Male	57.78
Female	42.22

dtype: float64

```
df["MaritalStatus"].unique()
```

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

```
round(df["MaritalStatus"].value_counts(normalize=True)*100,2)
```

MaritalStatus	proportion
Partnered	59.44
Single	40.56

dtype: float64

```
Start coding or ge_nerate with AI.
```

#Detecting outliers

#clipping data between 5th percentile and 95th percentile

```
#Income column


q1=np.percentile(df["Income"],25)
q3=np.percentile(df["Income"],75)
IQR = q3-q1
IQR
```

```
14609.25
```

```
upper_bound = (q3 + (1.5*IQR))
lower_bound = (q1 - (1.5*IQR))
median = df["Income"].median()
print("upper  bound","=",upper_bound)
print("lower  bound","=",lower_bound)
print("median","=",median)
```

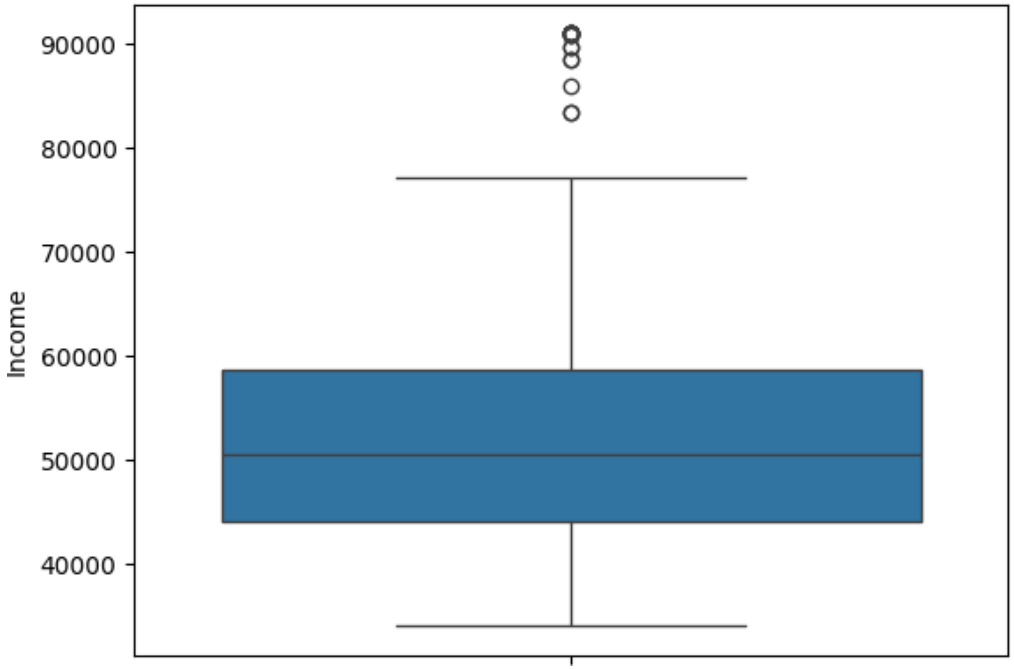
```
upper bound = 80581.875
lower bound = 22144.875
median = 50596.5
```

```
round((len(df.loc[df["Income"]>upper_bound])/len(df))*100,2)
```

```
10.56
```

```
#10.56% of values in Income column are outliers


#clipping data between 5th percentile and 95th percentile


d1= df["Income"].quantile(0.05)
d2= df["Income"].quantile(0.95)
df["Income"].clip(d1,d2,inplace=True)

sns.boxplot(data=df,y="Income")
plt.show()
```

```
<ipython-input-11-945fe1a7a68a>:3: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignme
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting value

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].met


  df["Income"].clip(d1,d2,inplace=True)
<ipython-input-11-945fe1a7a68a>:3: FutureWarning: Setting an item of incompatible dtype is deprecated and will raise an error in a future
  df["Income"].clip(d1,d2,inplace=True)
```




```
#Miles column


mq1=np.percentile(df["Miles"],25)
mq3=np.percentile(df["Miles"],75)
IQR = mq3-mq1
IQR
```

```
48.75
```

```
upper_bound = (mq3 + (1.5*IQR))
lower_bound = (mq1 - (1.5*IQR))
median = df["Miles"].median()
print("upper  bound","=",upper_bound)
print("lower  bound","=",lower_bound)
print("median","=",median)
```

```
upper bound = 187.875
lower bound = -7.125
median = 94.0
```

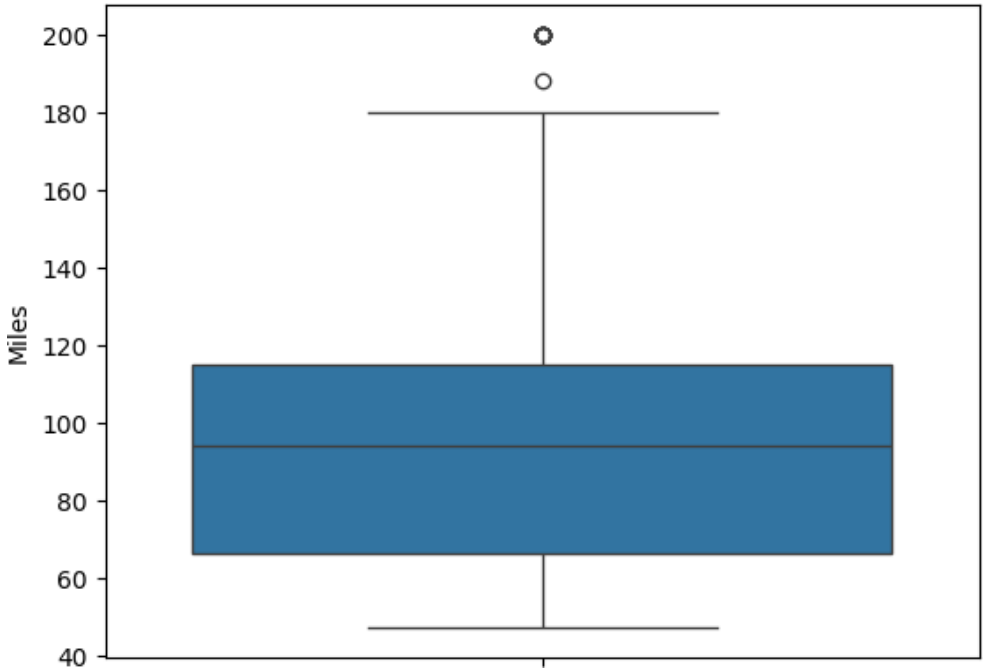
```
round((len(df.loc[df["Miles"]>upper_bound])/len(df))*100,2)
```

```
7.22
```

```
#7.22% valus in "Miles" column are outliers.


#clipping data between 5th percentile and 95th percentile


d1= df["Miles"].quantile(0.05)
d2= df["Miles"].quantile(0.95)
df["Miles"].clip(d1,d2,inplace=True)

sns.boxplot(data=df,y="Miles")
plt.show()
```

```
<ipython-input-53-e36ba10b9753>:3: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignme
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting value

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].met
```

```
df["Miles"].clip(d1,d2,inplace=True)
```




```
#Age column
```

```
aq1=np.percentile(df["Age"],25)
aq3=np.percentile(df["Age"],75)
IQR = aq3-aq1
IQR
```

```
9.0
```

```
upper_bound = (aq3 + (1.5*IQR))
lower_bound = (aq1 - (1.5*IQR))
median = df["Age"].median()
```

```
print("upper  bound","=",upper_bound)
print("lower  bound","=",lower_bound)
print("median","=",median)
```

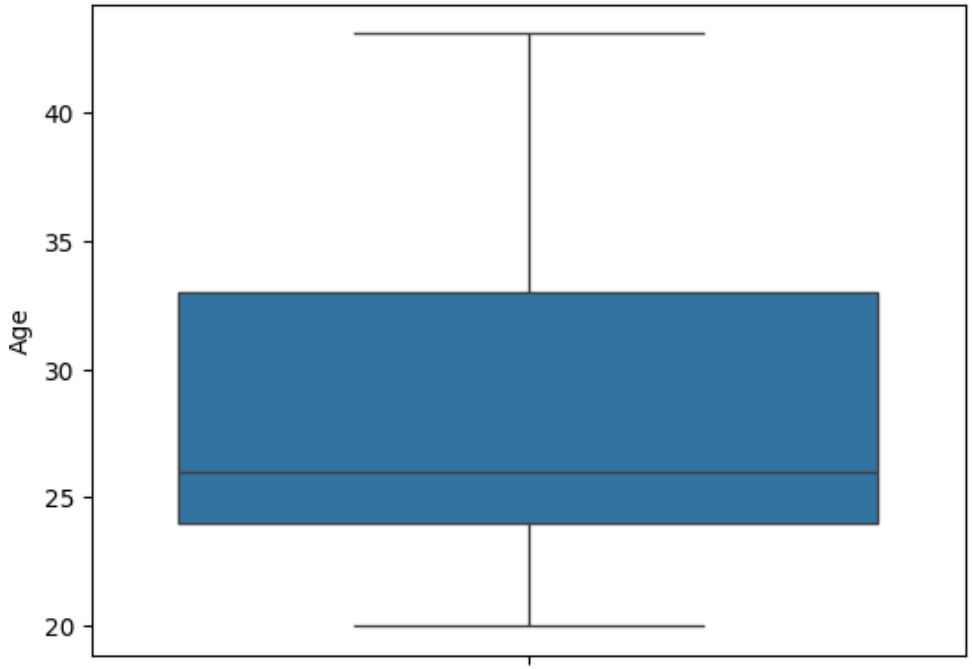
```
upper bound = 46.5
lower bound = 10.5
median = 26.0
```

```
round((len(df.loc[df["Age"]>upper_bound])/len(df))*100,2)
```

```
2.78
```

```
#clipping data between 5th percentile and 95th percentile


d1= df["Age"].quantile(0.05)
d2= df["Age"].quantile(0.95)
df["Age"].clip(d1,d2,inplace=True)

sns.boxplot(data=df,y="Age")
plt.show()
```

```
<ipython-input-18-442026faa94c>:3: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignme
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting value

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].met


  df["Age"].clip(d1,d2,inplace=True)
<ipython-input-18-442026faa94c>:3: FutureWarning: Setting an item of incompatible dtype is deprecated and will raise an error in a future
  df["Age"].clip(d1,d2,inplace=True)
```




```
#Education


eq1=np.percentile(df["Education"],25)
eq3=np.percentile(df["Education"],75)
IQR = eq3-eq1
IQR
```

```
2.0
```

```
upper_bound = (eq3 + (1.5*IQR))
lower_bound = (eq1 - (1.5*IQR))
median = df["Education"].median()
print("upper  bound","=",upper_bound)
print("lower  bound","=",lower_bound)
print("median","=",median)
```

```
upper bound = 19.0
lower bound = 11.0
median = 16.0
```

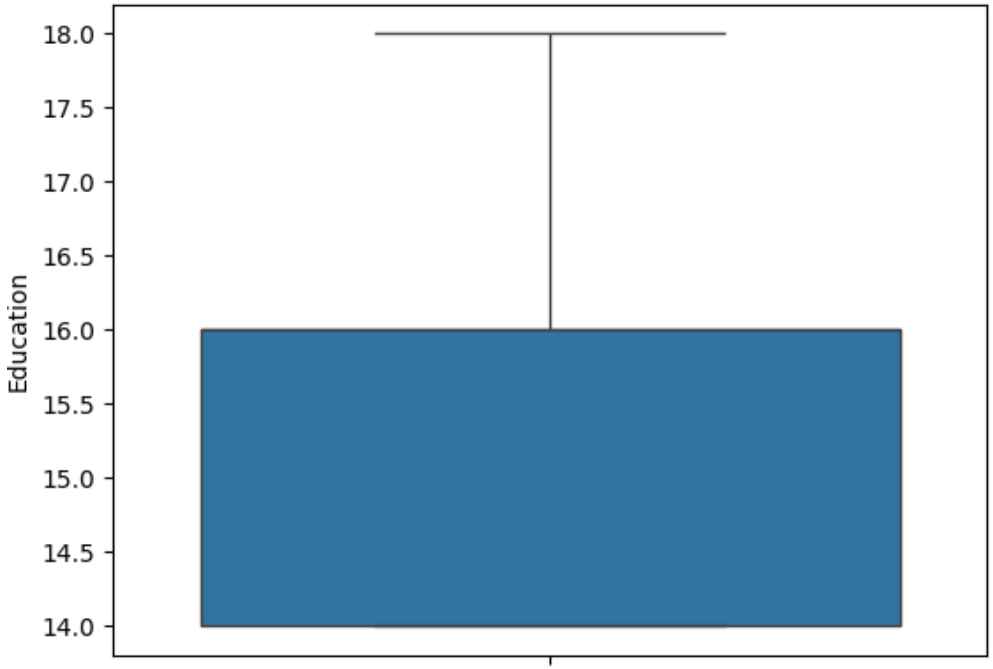
```
round((len(df.loc[df["Education"]>upper_bound])/len(df))*100,2)
```

```
2.22
```

```
#clipping data between 5th percentile and 95th percentile
d1= df["Education"].quantile(0.05)
d2= df["Education"].quantile(0.95)
df["Education"].clip(d1,d2,inplace=True)

sns.boxplot(data=df,y="Education")
plt.show()
```

```
<ipython-input-22-03338189569a>:4: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignme
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting value

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].met
```

```
df["Education"].clip(d1,d2,inplace=True)
```



#Adding new columns and categorizing values in Age , Education , Income and Miles for better analysis.

```
Start coding or ge_nerate with AI.
```

```
#bining age values in category
bin_range1=[17,25,35,45,float("inf")]
bin_labels1=["Young Adults","Adults","Middle Aged Adults","Elder"]
df["age_group"]=pd.cut(df["Age"],bins=bin_range1,labels=bin_labels1)
```

```
#bining Education values in category
bin_range2=[0,12,15,float("inf")]
bin labels2=["Primary Education","Secondary Education","Higher Secondary Education"]
```

```
df["edu_group"]=pd.cut(df["Education"],bins=bin_range2,labels=bin_labels2)


#bining income values in category
bin_range3=[0,40000,60000,80000,float("inf")]
bin_labels3=["Low Income","Moderate Income","High Income","Very High Income"]
df["inc_group"]=pd.cut(df["Income"],bins=bin_range3,labels=bin_labels3)


#bining miles values in category
bin_range4=[0,50,100,200,float("inf")]
bin_labels4=["Light Activity","Moderate Activity","Active Lifestyle","Fitness Enthusiast"]
df["miles_group"]=pd.cut(df["Miles"],bins=bin_range4,labels=bin_labels4)


df.head()
```

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles	age_group	edu_group	inc_group	miles_group
0	KP281	20.0	Male	14	Single	3	4	34053.15	112	Young Adults	Secondary Education	Low Income	Active Lifestyle
1	KP281	20.0	Male	15	Single	2	3	34053.15	75	Young Adults	Secondary Education	Low Income	Moderate Activity
2	KP281	20.0	Female	14	Partnered	4	3	34053.15	66	Young Adults	Secondary Education	Low Income	Moderate Activity
3	KP281	20.0	Male	14	Single	3	3	34053.15	85	Young Adults	Secondary Education	Low Income	Moderate Activity
4	KP281	20.0	Male	14	Partnered	4	2	35247.00	47	Young Adults	Secondary Education	Low Income	Light Activity

Next steps: Generate code with df | View recommended plots | New interactive sheet

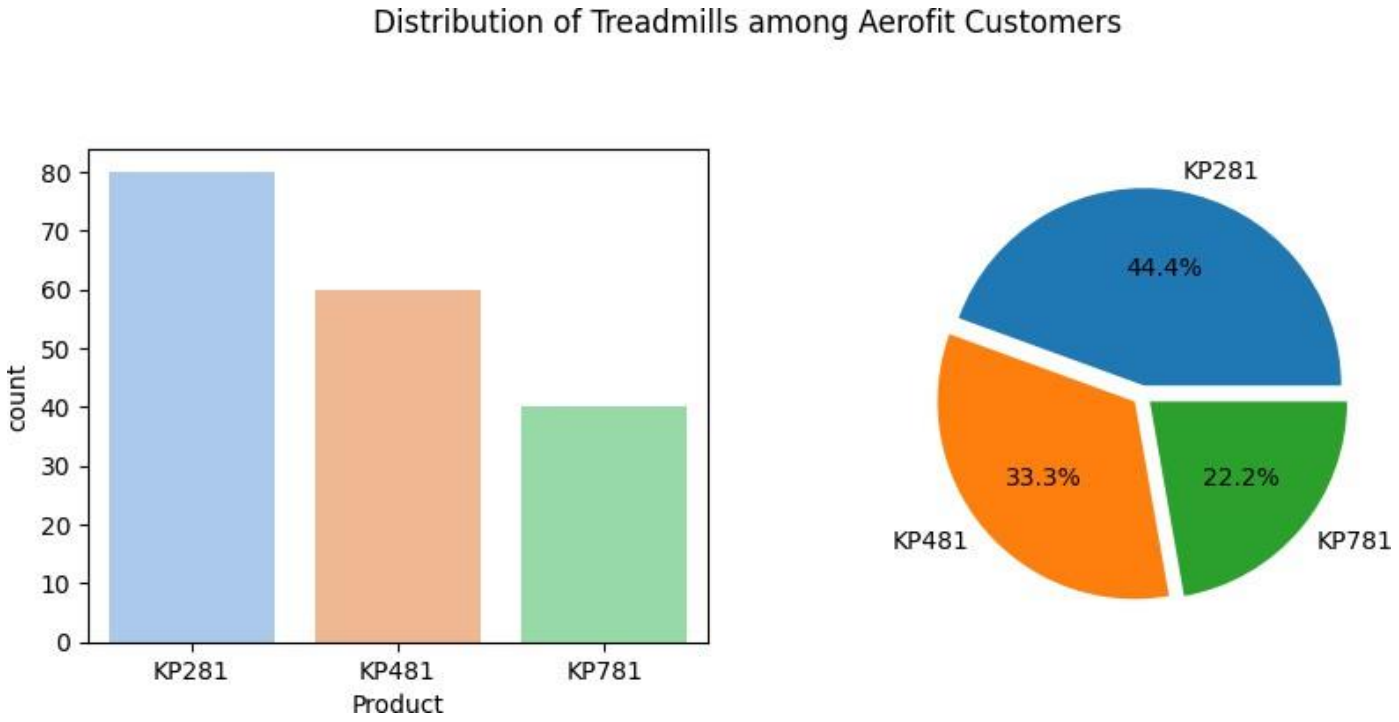
#Univariate Analysis

#Categorical columns

```
#Distribution of Treadmills among Aerofit Customers
```

```
plt.figure(figsize=(10,8))
plt.subplot(2,2,1)
sns.countplot(data=df,x=df["Product"],palette="pastel")
plt.subplot(2,2,2)
plt.pie(df["Product"].value_counts(),labels=df["Product"].unique(),
        explode=(0.05,0.05,0.05),autopct="%1.1f%%")
plt.suptitle("Distribution of Treadmills among Aerofit Customers")
plt.show()
```

```
<ipython-input-28-435a5452dc6a>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=

  sns.countplot(data=df,x=df["Product"],palette="pastel")
```

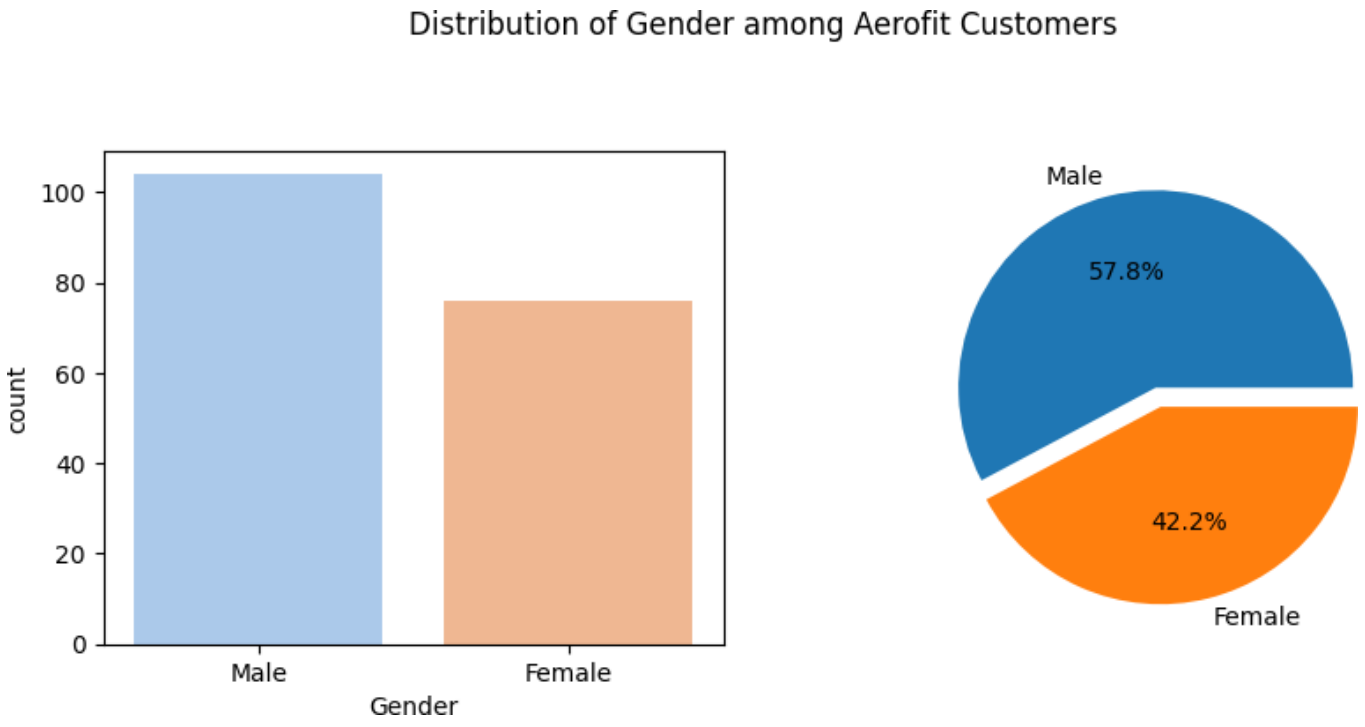


*44.4% users prefer KP281 Treadmill while 33.3% users prefer KP481 Treadmill and only 22.2% users favor KP781 Treadmill.

Start coding or ge_nerate with AI.

```
#Distribution of Gender among Aerofit Customers.


plt.figure(figsize=(10,8))
plt.subplot(2,2,1)
sns.countplot(data=df,x=df["Gender"],palette="pastel")
plt.subplot(2,2,2)
plt.pie(df["Gender"].value_counts(),labels=df["Gender"].unique(),
        explode=(0.05,0.05),autopct="%1.1f%%")
plt.suptitle("Distribution of Gender among Aerofit Customers")
plt.show()
```

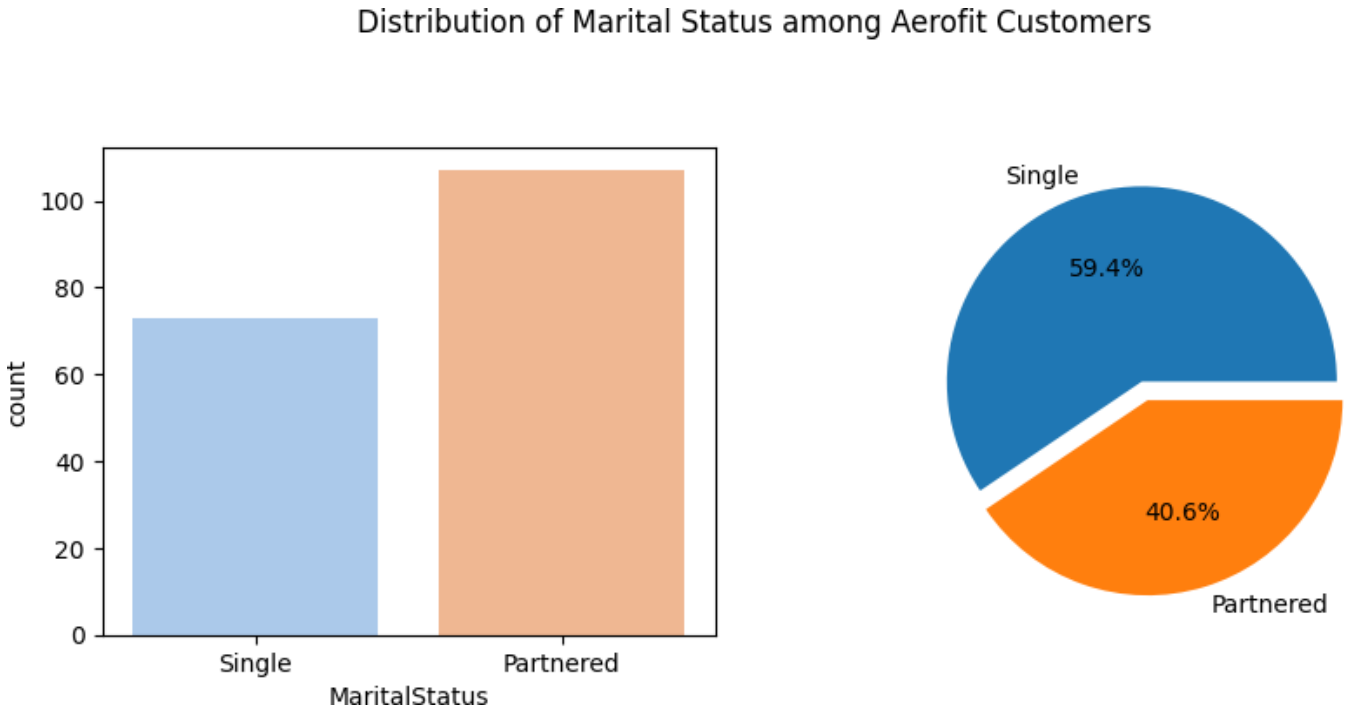
```
<ipython-input-29-6c8f4b7e51ff>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=

  sns.countplot(data=df,x=df["Gender"],palette="pastel")
```



*Aerofit has 57.8% male customers and 42.2% female customers.

```
Start coding or ge_nerate with AI.


#Distribution of Marital Status among aerofit customers.


plt.figure(figsize=(10,8))
plt.subplot(2,2,1)
sns.countplot(data=df,x=df["MaritalStatus"],palette="pastel")
plt.subplot(2,2,2)
plt.pie(df["MaritalStatus"].value_counts(),labels=df["MaritalStatus"].unique(),
        explode=(0.05,0.05),autopct="%1.1f%%")
plt.suptitle("Distribution of Marital Status among Aerofit Customers")
plt.show()
```

```
<ipython-input-30-98d363f24566>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=

  sns.countplot(data=df,x=df["MaritalStatus"],palette="pastel")
```



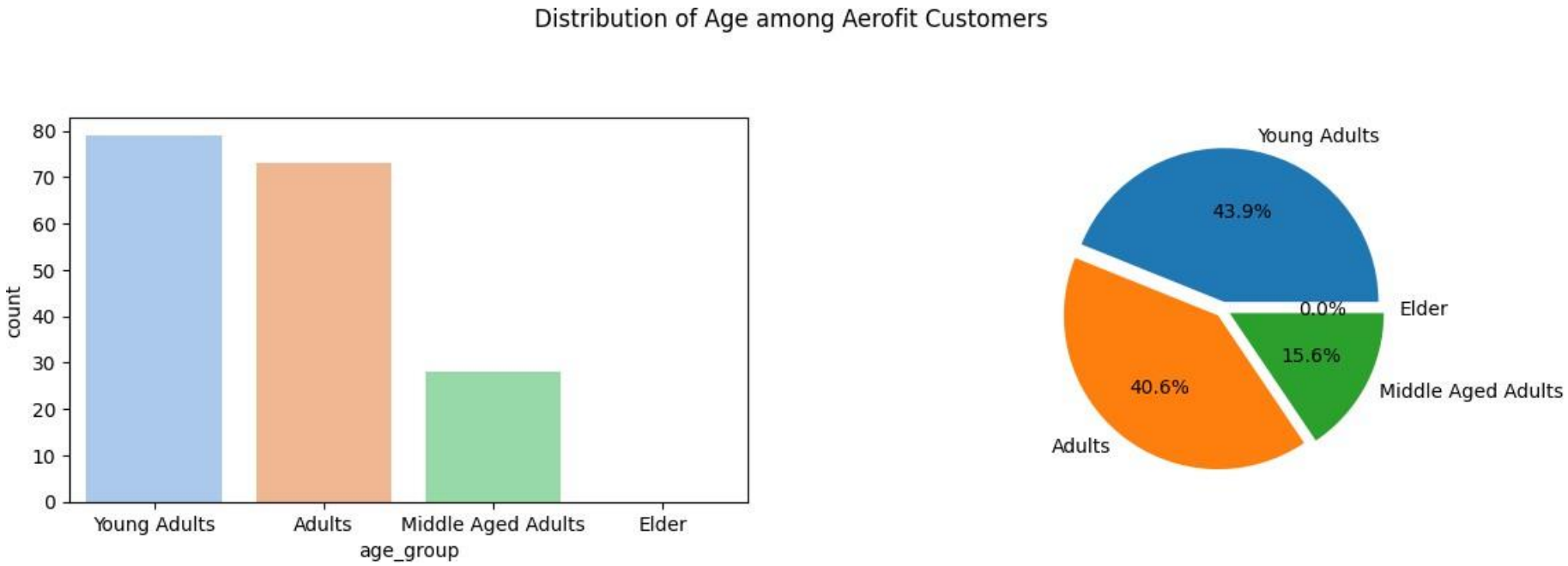
*59.4% Aerofit treadmill users are single while remaining 40.6% users are married.

```
Start coding or ge_nerate with AI.
```

```
#Distribution of age among Aerofit Customers
```

```
plt.figure(figsize=(14,8))
plt.subplot(2,2,1)
sns.countplot(data=df,x=df["age_group"],palette="pastel")
plt.xlabel="Age group"
plt.ylabel="Number of users"

plt.subplot(2,2,2)
plt.pie(df["age_group"].value_counts(),
        labels=df["age_group"].value_counts().index,
        explode=(0.05,) * len(df["age_group"].value_counts()),
        autopct="%1.1f%%")
plt.suptitle("Distribution of Age among Aerofit Customers")
plt.show()
```

```
<ipython-input-31-a31a004ecc95>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=

  sns.countplot(data=df,x=df["age_group"],palette="pastel")
```

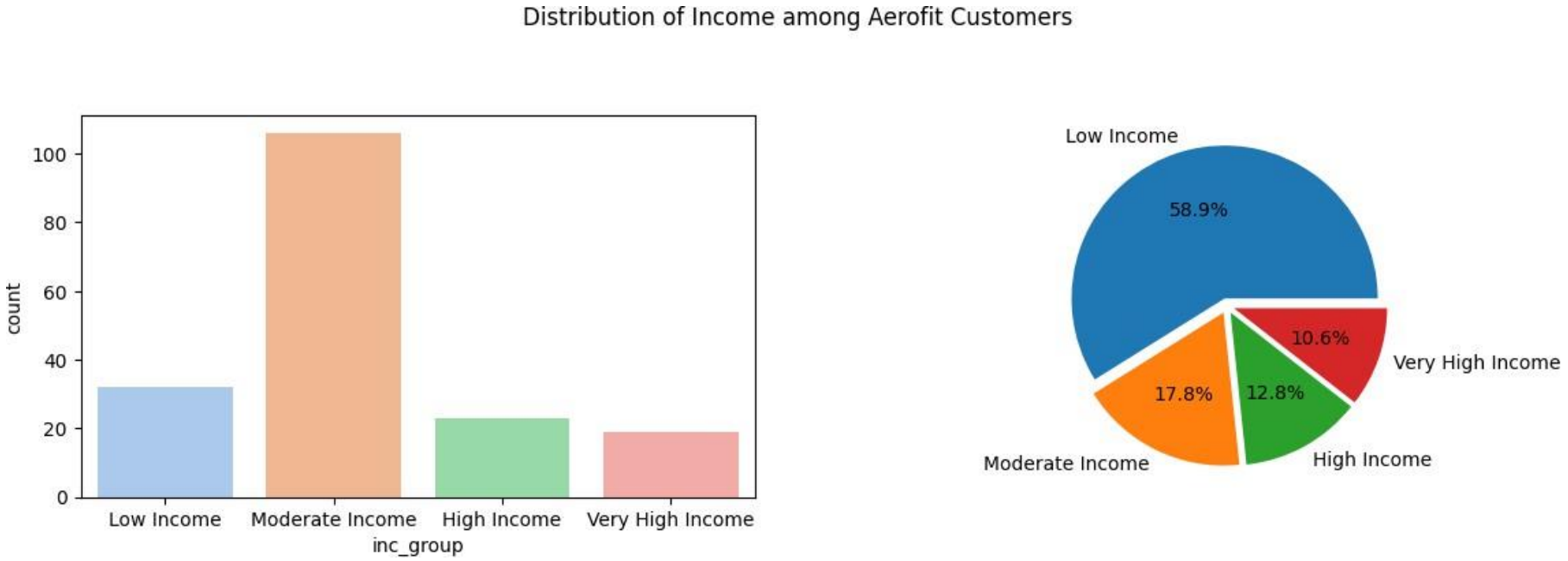


*Most of the aerofit customers falls under "Young Adults" age group.

```
#Distribution of age among Aerofit Customers


plt.figure(figsize=(14,8))
plt.subplot(2,2,1)
sns.countplot(data=df,x=df["inc_group"],palette="pastel")
plt.xlabel="Income group"
plt.ylabel="Number of users"

plt.subplot(2,2,2)
plt.pie(df["inc_group"].value_counts(), labels=df["inc_group"].unique(), explode=(0.05,0.05,0.05,0.05), autopct="%1.1f%%")
```

```
plt.suptitle("Distribution of Income among Aerofit Customers")
plt.show()
```

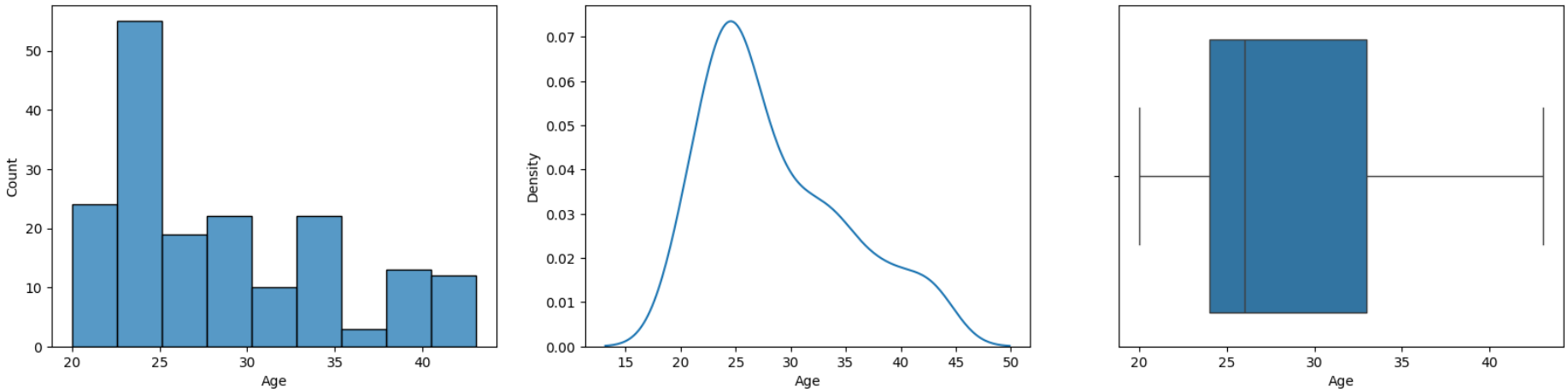
```
<ipython-input-32-4be012b17d2c>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=

  sns.countplot(data=df,x=df["inc_group"],palette="pastel")
```



#Numeric Columns

```
#Distribution of Age


plt.figure(figsize=(20,10))
#histogram
plt.subplot(2,3,1)
sns.histplot(data=df,x="Age")
```

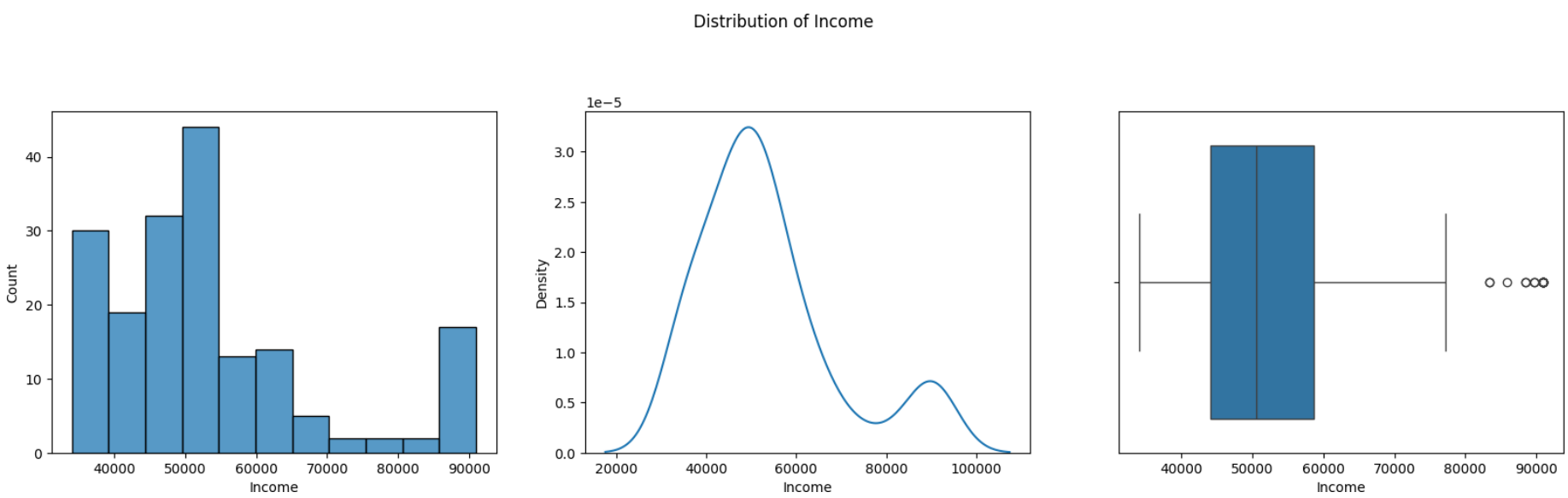
```
#kde plot
plt.subplot(2,3,2)
sns.kdeplot(data=df,x="Age")

#boxplot
plt.subplot(2,3,3)
sns.boxplot(data=df,x="Age")
plt.suptitle("Distribution of age")
plt.show()
```



*Majority of Aerofit customers belong to age group 18-30 and there is also a high probability of them buying Aerofit Treadmill. *There are few customers in age group of above 40 and having low probability of them buying Aerofit Treadmill.

```
#Distribution of Income


plt.figure(figsize=(20,10))
#histogram
plt.subplot(2,3,1)
sns.histplot(data=df,x="Income")
```

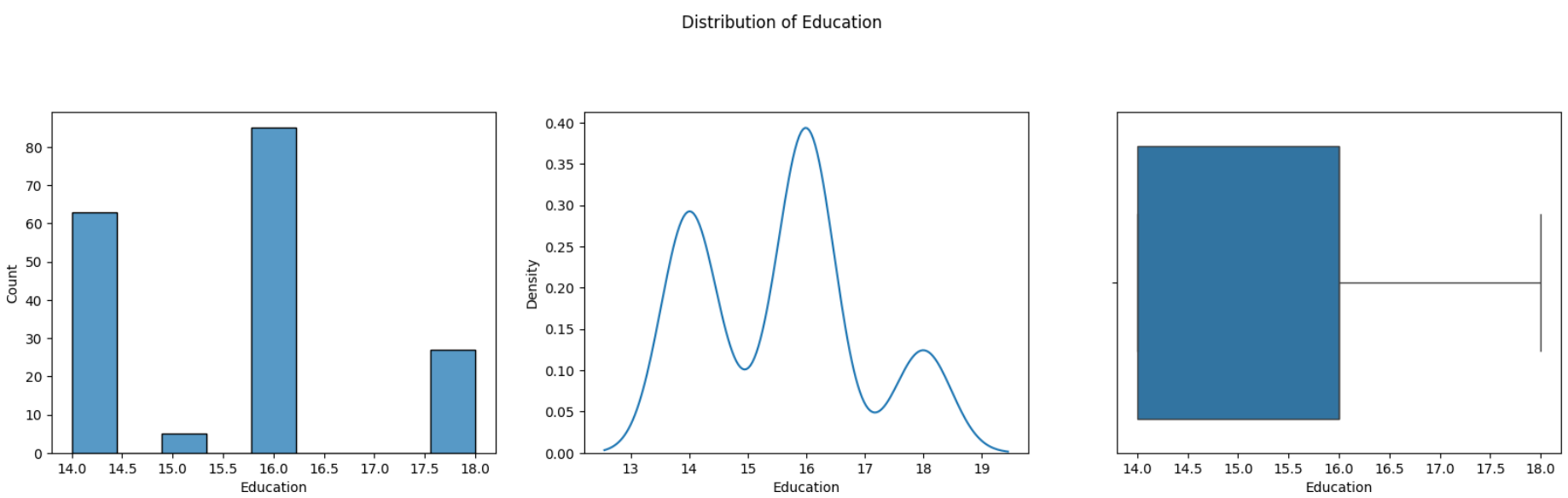
```
#kde plot
plt.subplot(2,3,2)
sns.kdeplot(data=df,x="Income")

#boxplot
plt.subplot(2,3,3)
sns.boxplot(data=df,x="Income")
plt.suptitle("Distribution of Income")
plt.show()
```



*Majority of aerofit users are in the income range of 40000 to 60000 and there is a high probability of them buying Aerofit treadmill. *Noted point is that high income people of 80000 and above having low probability of them buying Aerofit treadmill.

```
#Distribution of Education


plt.figure(figsize=(20,10))
#histogram
plt.subplot(2,3,1)
sns.histplot(data=df,x="Education")


#kde plot
```

```
plt.subplot(2,3,2)
sns.kdeplot(data=df,x="Education")

#boxplot
plt.subplot(2,3,3)
sns.boxplot(data=df,x="Education")
plt.suptitle("Distribution of Education")
plt.show()
```



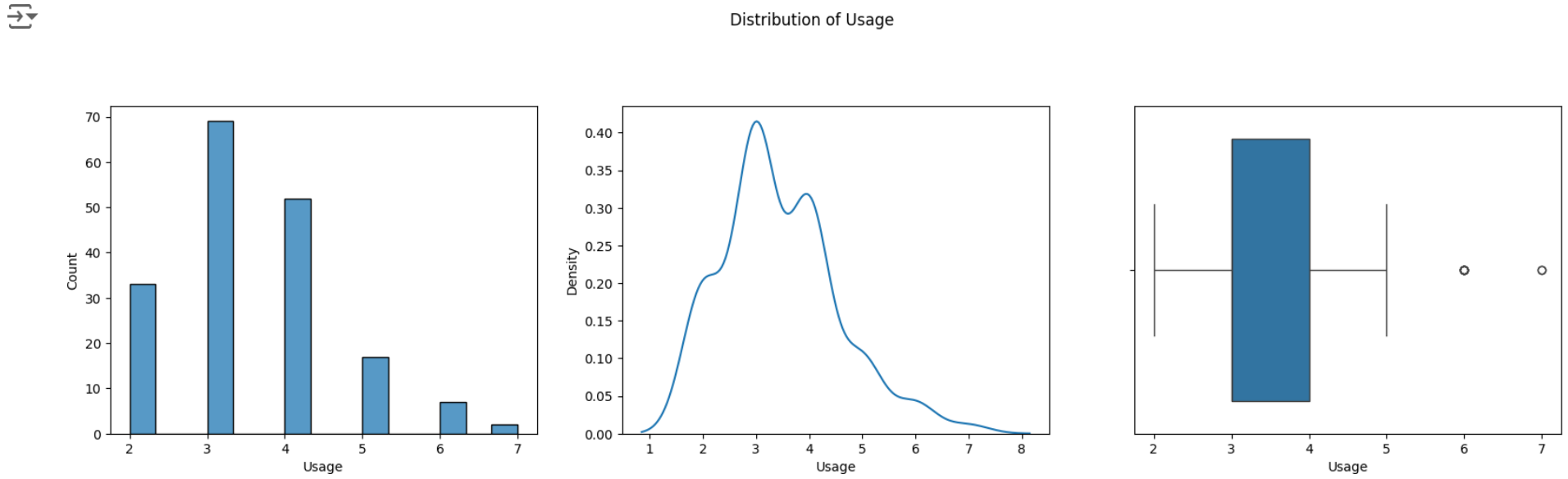
*Customers with 16 years of education(Higher Education) are having high probability of buying Aerofit treadmill and customers having 18 and above years of education have low probability of buying Aerofit treadmill.

```
Start coding or ge_nerate with AI.
```

```
df["Usage"]=df["Usage"].astype("int")
df["Fitness"]=df["Fitness"].astype("int")
```

```
#distribution of Usage
```

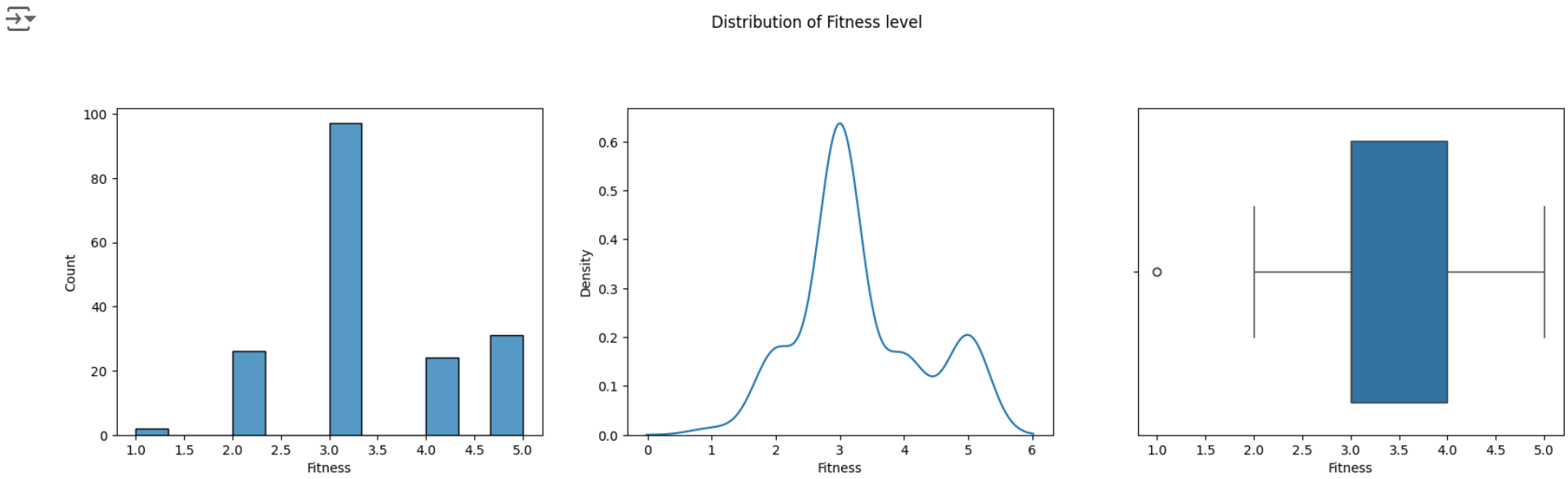
```
plt.figure(figsize=(20,10))
#histogram
plt.subplot(2,3,1)
sns.histplot(data=df,x="Usage")


#kde plot
plt.subplot(2,3,2)
sns.kdeplot(data=df,x="Usage")

#boxplot
plt.subplot(2,3,3)
sns.boxplot(data=df,x="Usage")
plt.suptitle("Distribution of Usage")
plt.show()
```



*Majority of customers use treadmills three times a week and they have higher probability of buying Aerofit Treadmill.

```
#Distribution of Fitness Level


plt.figure(figsize=(20,10))
#histogram
```

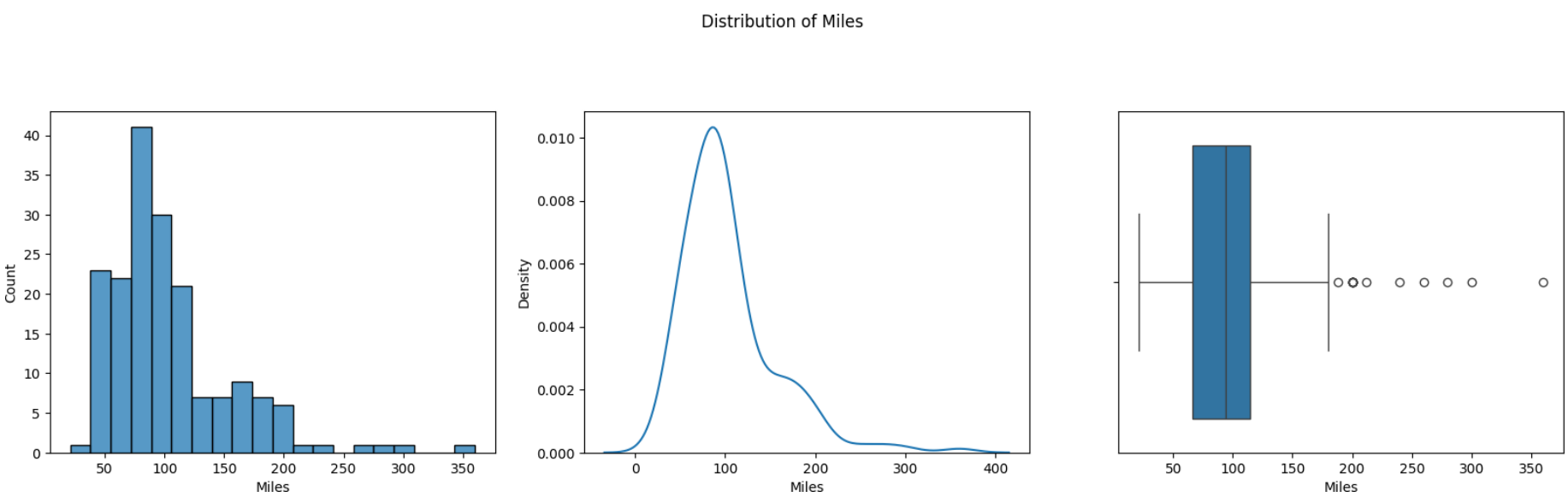
```
plt.subplot(2,3,1)
sns.histplot(data=df,x="Fitness")


#kde plot
plt.subplot(2,3,2)
sns.kdeplot(data=df,x="Fitness")

#boxplot
plt.subplot(2,3,3)
sns.boxplot(data=df,x="Fitness")
plt.suptitle("Distribution of Fitness level")
plt.show()
```



*Majority of Aerofit customers have fitness level 3 and they have high probability of buying Aerofit treadmill.

```
#Distribution of miles


plt.figure(figsize=(20,10))
#histogram
plt.subplot(2,3,1)
sns.histplot(data=df,x="Miles")
```

```
#kde plot
plt.subplot(2,3,2)
sns.kdeplot(data=df,x="Miles")

#boxplot
plt.subplot(2,3,3)
sns.boxplot(data=df,x="Miles")
plt.suptitle("Distribution of Miles")
plt.show()
```



*Customers who run 80-100 miles per week prefer Aerofit treadmill and who run above 200 miles prefer jogging over treadmill.

```
Start coding or ge_nerate with AI.
```

Bivariate Analysis

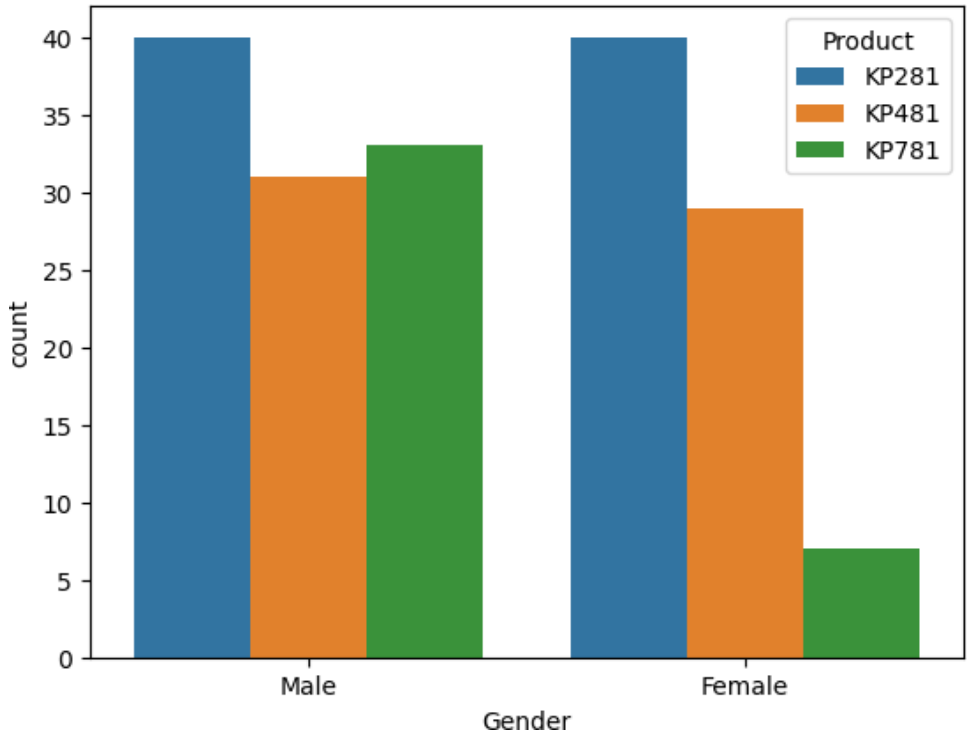
```
#Distribution of Gender across each treadmill.
```

```
df_gender = df.groupby(["Product","Gender"]).size().unstack()
df_gender
```

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

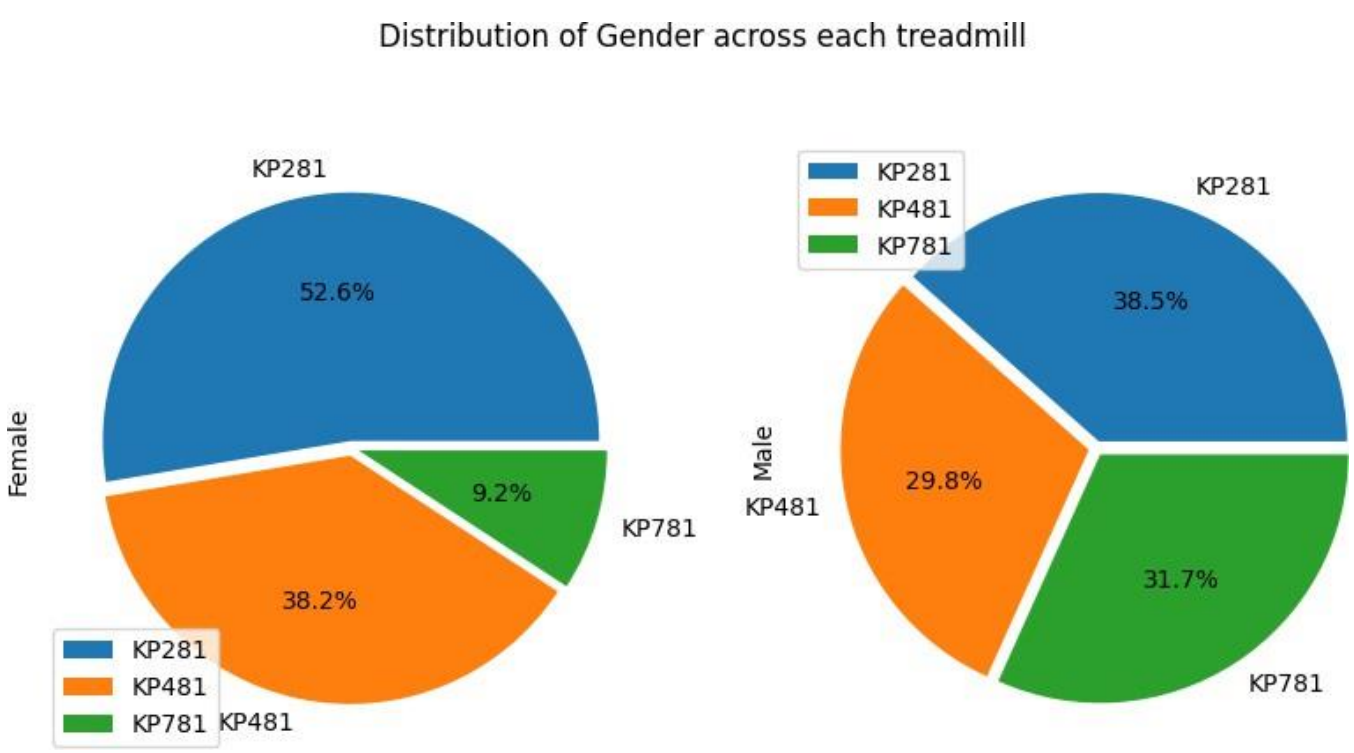
Next steps: Generate code with df_gender | View recommended plots | New interactive sheet

```
sns.countplot(data=df,x="Gender",hue="Product")
plt.show()
```



*Both Male and Female customers prefer KP281 treadmill.

```
df_gender.plot(kind="pie",subplots=True,explode=(0.03,0.03,0.03),figsize=(10,5),autopct="%1.1f%%")
plt.suptitle("Distribution of Gender across each treadmill")
plt.show()
```

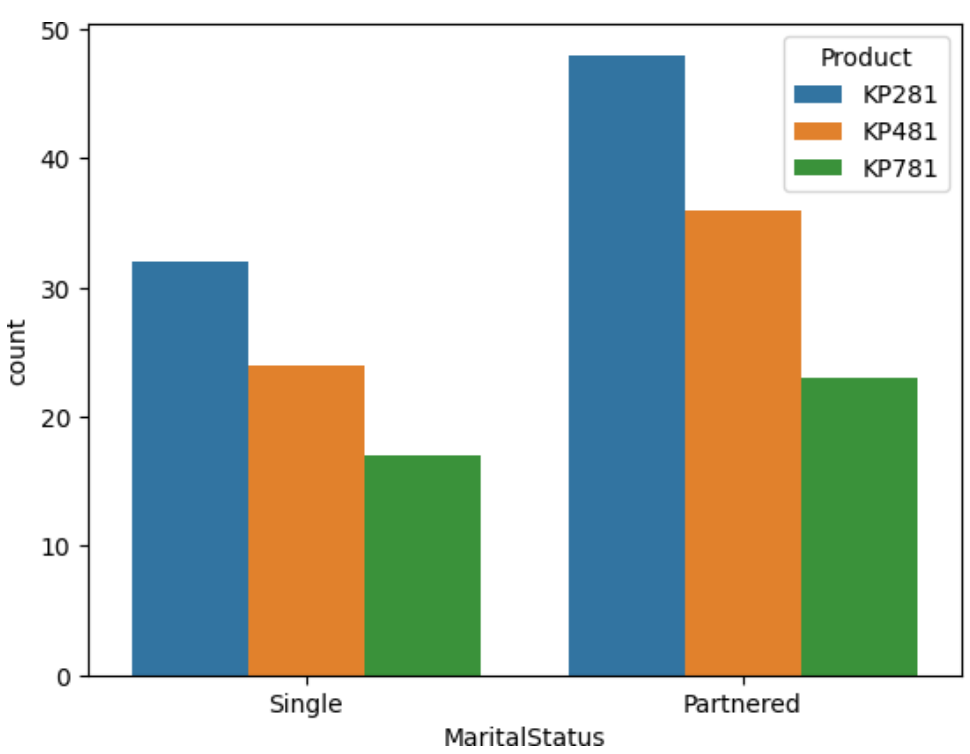



```
#Distribution of Marital Status among customers who purchased each treadmil.


df_maritalstatus = df.groupby(["Product","MaritalStatus"]).size().unstack()
df_maritalstatus
```

MaritalStatus	Partnered	Single
Product		
KP281	48	32
KP481	36	24
KP781	23	17

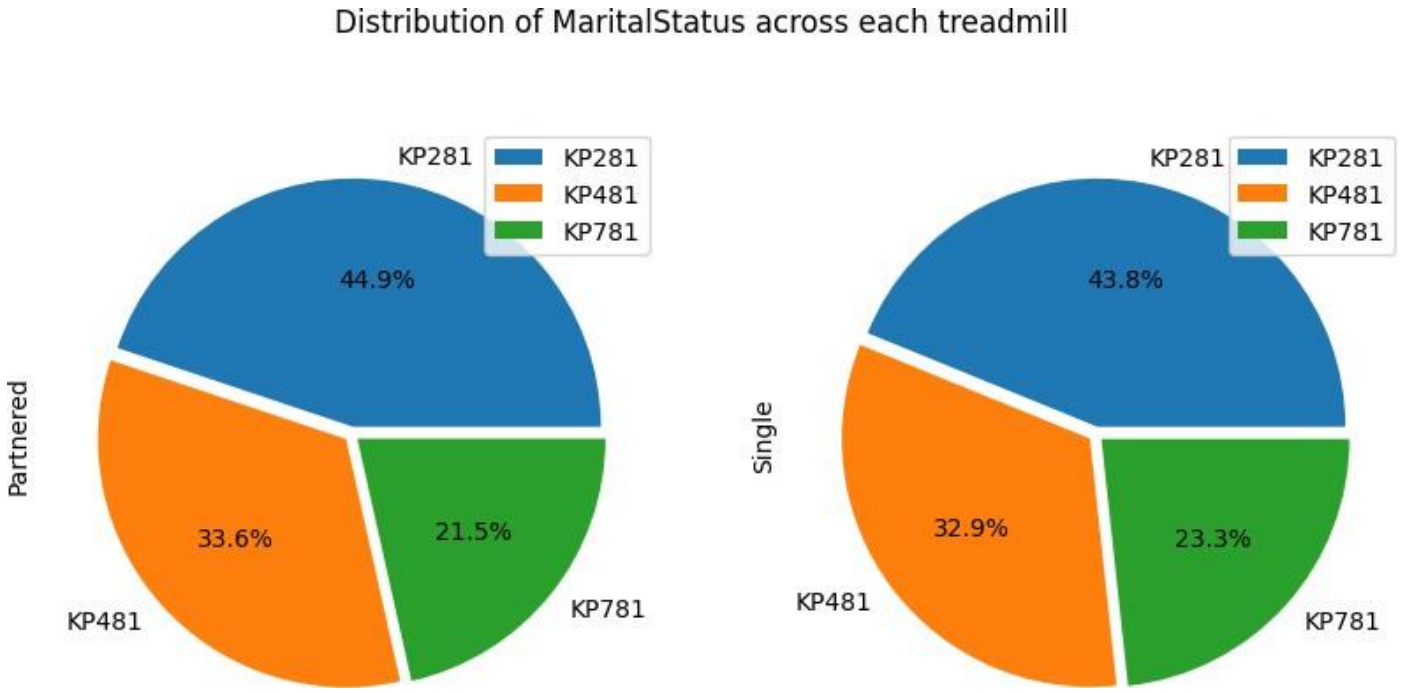
Next steps: Generate code with df_maritalstatus | View recommended plots | New interactive sheet

```
sns.countplot(data=df,x="MaritalStatus",hue="Product")
plt.show()
```

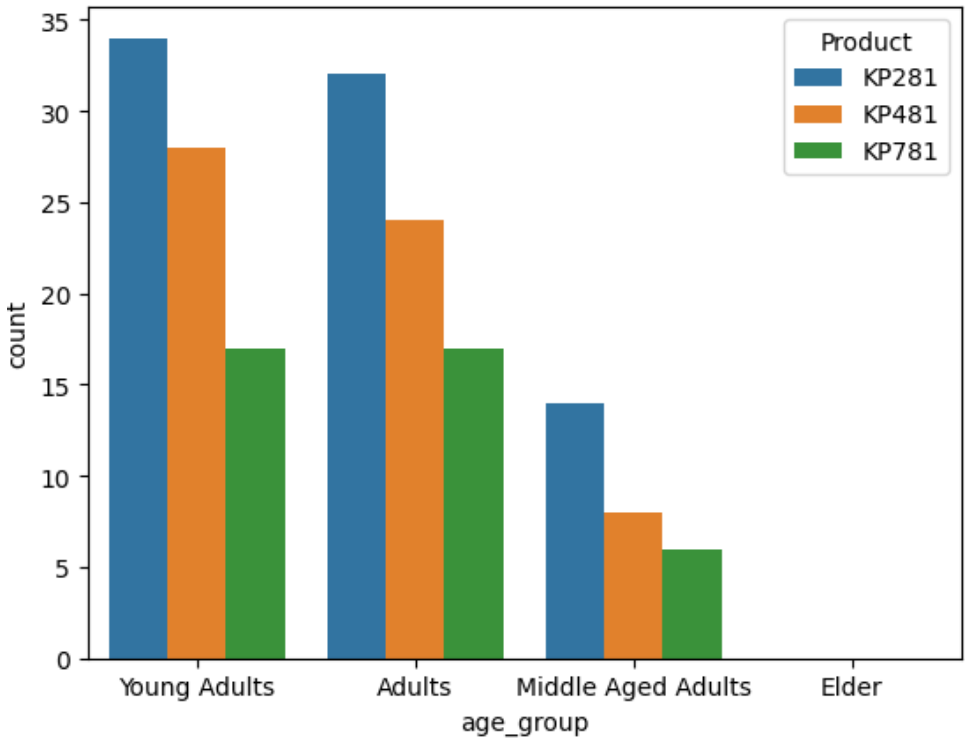


*Married customers have high frequency of purchasing aerofit treadmills compared to single customers.

```
df_maritalstatus.plot(kind="pie",subplots=True,explode=(0.03,0.03,0.03),figsize=(10,5),autopct="%1.1f%%")
plt.suptitle("Distribution of MaritalStatus across each treadmill")
plt.show()
```




```
#Distribution of Agegroup for each treadmil


sns.countplot(data=df,x="age_group",hue="Product")
plt.show()
```



*Majority of young adults and adults prefer KP281 treadmill over other two treadmillls

```
Start coding or ge_nerate with AI.
```

```
#Distribution of Income group across each treadmill
```

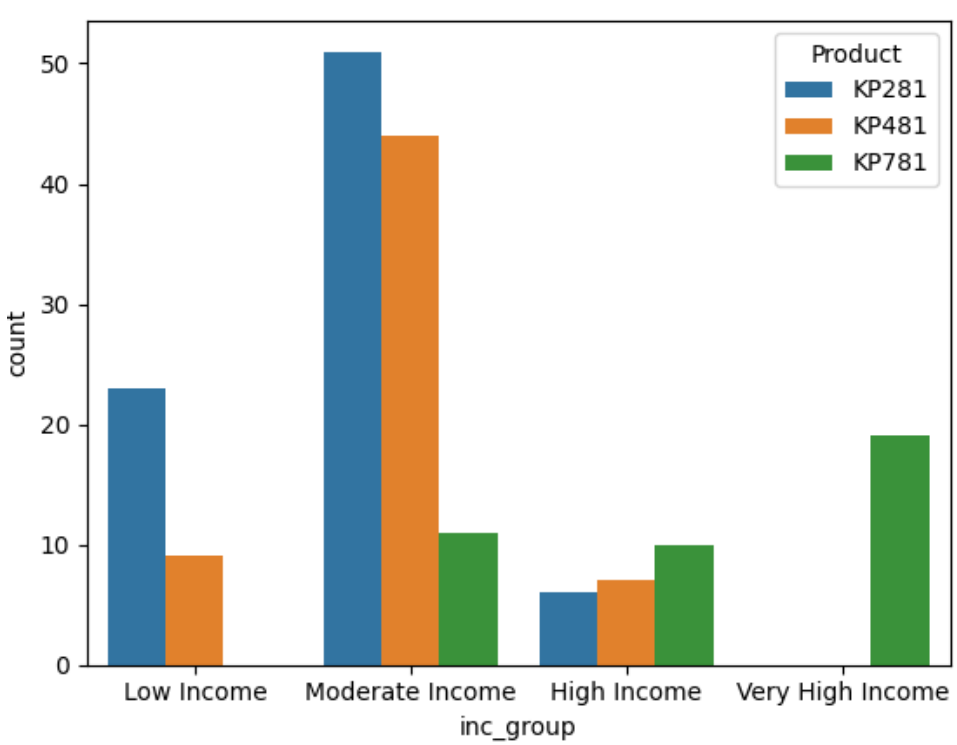
```
df_incomegroup = df.groupby(["Product","inc_group"]).size().unstack()
df_incomegroup
```

```
<ipython-input-58-6084d838e4c3>:1: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future ver
  df_incomegroup = df.groupby(["Product","inc_group"]).size().unstack()
```

Product	Low Income	Moderate Income	High Income	Very High Income
KP281	23	51	6	0
KP481	9	44	7	0
KP781	0	11	10	19

Next steps: Generate code with `df_incomegroup` | View recommended plots | New interactive sheet

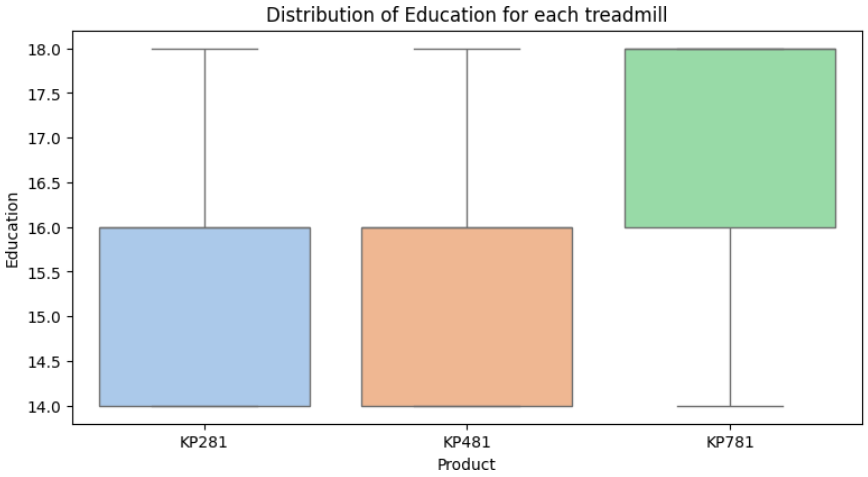
```
sns.countplot(data=df,x="inc_group",hue="Product")
plt.show()
```

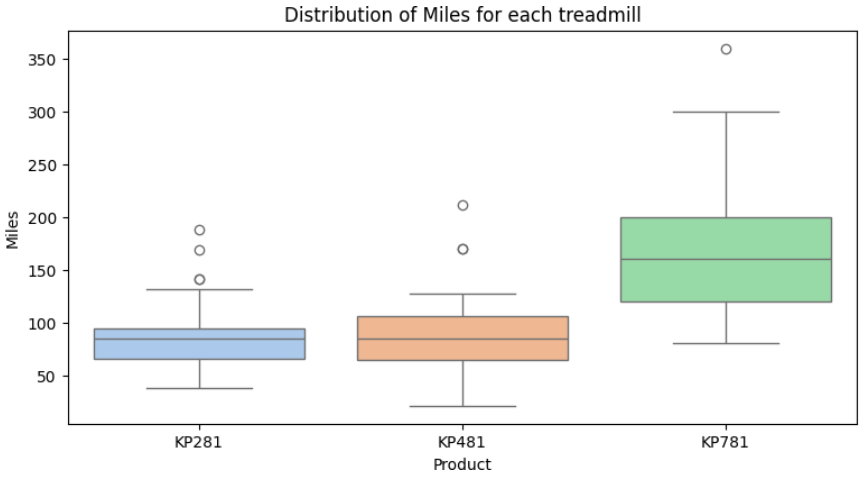


*Most of the customers belong to moderate income ,and these customers prefer both KP281 and KP481 treadmill with difference of few customers.

```
#Distribution of Miles and Education for each treadmill.


columns = ["Education","Miles"]
plt.figure(figsize=(20,10))
for i,col in enumerate (columns,1):
  plt.subplot(2,2,i)
  sns.boxplot(data=df,x="Product",y=col,palette="pastel")
  plt.title(f'Distribution of {col} for each treadmill')
plt.show()
```

```
<ipython-input-63-51b1c4994abf>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=

  sns.boxplot(data=df,x="Product",y=col,palette="pastel")
<ipython-input-63-51b1c4994abf>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=

  sns.boxplot(data=df,x="Product",y=col,palette="pastel")
```





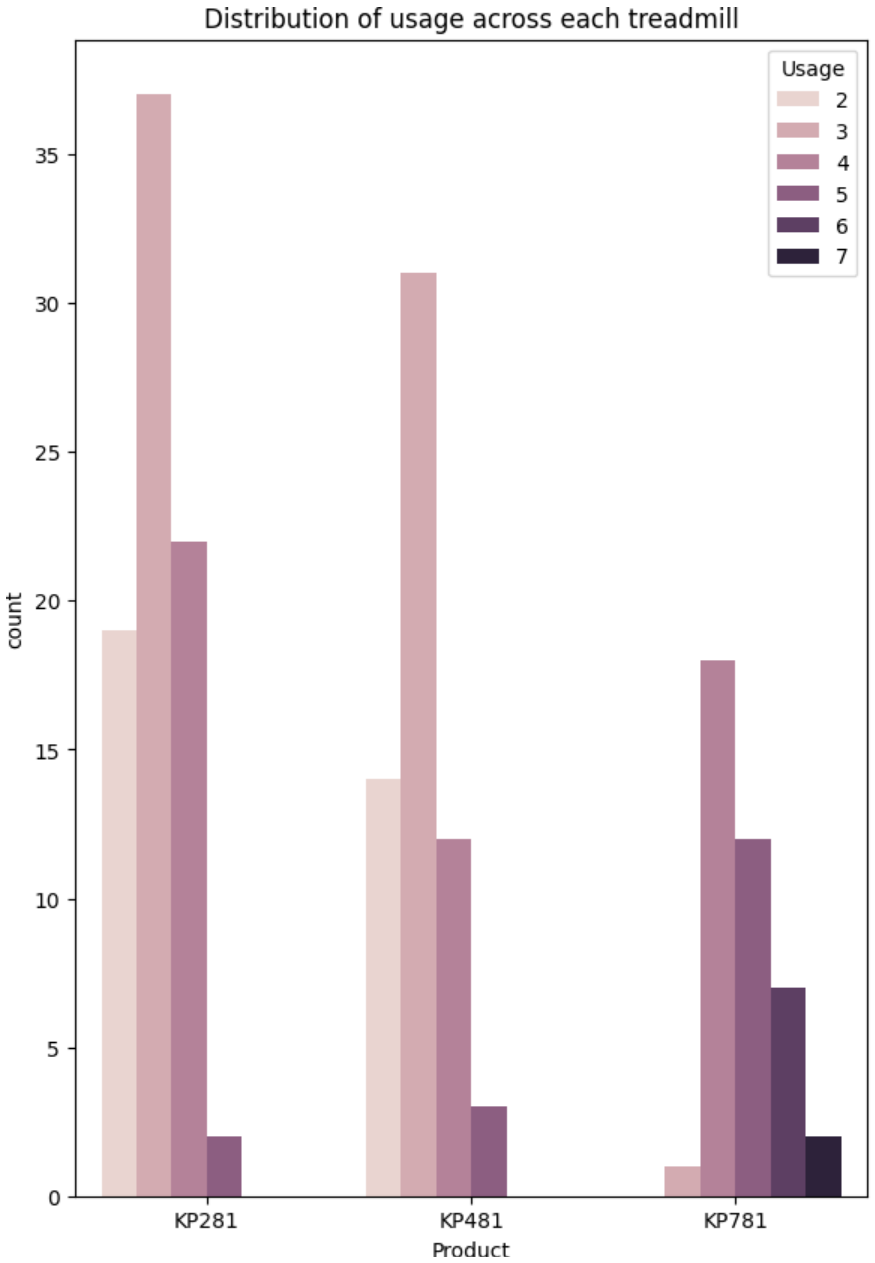
*Customers with 14-16 years of education prefers KP281 and KP481 treadmill,while majority of customerswith 16-18 years of education prefer KP781 treadmill.

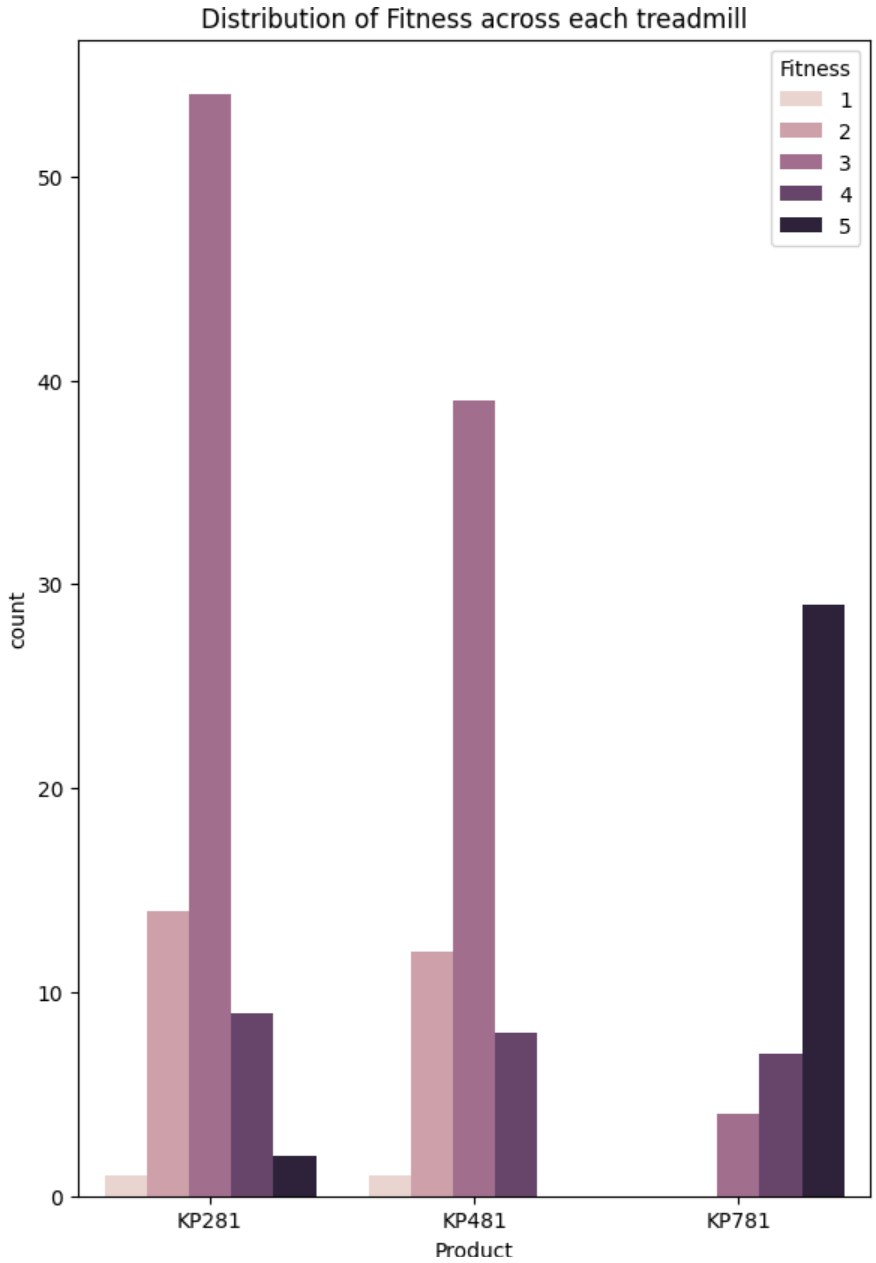
*Customers who run 60-100 miles per week prefers KP281 treadmill,who runs 60-120 miles per week prefers KP481 treadmill and who runs 120-200 miles per week prefers KP781

```
Start coding or ge_nerate with AI.
```

```
#Distribution of usage and fitness across each treadmill.
```

```
plt.figure(figsize=(15,10))
#Usage column
plt.subplot(1,2,1)
sns.countplot(data=df,x="Product",hue="Usage")
plt.title("Distribution of usage across each treadmill")

#Fitness column
plt.subplot(1,2,2)
sns.countplot(data=df,x="Product",hue="Fitness")
plt.title("Distribution of Fitness across each treadmill")
plt.show()
```





*Customers who use Treadmill 3 times a week prefers KP281 and KP481 treadmill,while who use 4-5 times a week prefers KP781 treadmill.

*Customers with fitness level 3 highly prefer KP281 and KP481 treadmill and who have fitness level 5 prefers KP781.

```
Start coding or ge_nerate with AI.
```

Conditional and marginal probabilities

```
#Impact of Gender on purchasing the Treadmill
```

```
round(pd.crosstab(index=df["Product"],columns=df["Gender"],margins=True,margins_name="Total",normalize=True),2)
```

Gender	Female	Male	Total
Product			
KP281	0.22	0.22	0.44
KP481	0.16	0.17	0.33
KP781	0.04	0.18	0.22
Total	0.42	0.58	1.00

```
#Impact of Marital Status on purchasing the Treadmill
```

```
round(pd.crosstab(index=df["Product"],columns=df["MaritalStatus"],margins=True,margins_name="Total",normalize=True),2)
```

MaritalStatus	Partnered	Single	Total
Product			
KP281	0.27	0.18	0.44
KP481	0.20	0.13	0.33
KP781	0.13	0.09	0.22
Total	0.59	0.41	1.00

```
#Impact of age group on purchasing the Treadmill
```

```
round(pd.crosstab(index=df["Product"],columns=df["age_group"],margins=True,margins_name="Total",normalize=True),2)
```

age_group	Young Adults	Adults	Middle Aged Adults	Total
Product				
KP281	0.19	0.18	0.08	0.44
KP481	0.16	0.13	0.04	0.33
KP781	0.09	0.09	0.03	0.22
Total	0.44	0.41	0.16	1.00

```
#Impact of Income group on purchasing the Treadmill
```

```
round(pd.crosstab(index=df["Product"],columns=df["inc_group"],margins=True,margins_name="Total",normalize=True),2)
```

inc_group	Low Income	Moderate Income	High Income	Very High Income	Total
Product					
KP281	0.13	0.28	0.03	0.00	0.44
KP481	0.05	0.24	0.04	0.00	0.33
KP781	0.00	0.06	0.06	0.11	0.22
Total	0.18	0.59	0.13	0.11	1.00

```
Start coding or ge_nerate with AI.
```

What is the probability that a customer purchased a perticular treadmill product(KP281,KP481,KP781)given that they use treadmill 3 times in a week?

```
total = len(df)
products=["KP281","KP481","KP781"]
usage=3
#calculating probability for each product and fitness level
probabilities={}

for product in products:
```

```
  #calculating no.of customes who purchased specific product
  total_usage = len(df.loc[df["Usage"]==usage])
  #calculating no. of customers who purchased specific product and who uses treadmill 3 times in a week.
  total_product_miles=len(df.loc[(df["Product"]==product)&(df["Usage"]==usage)])
  #calculating conditional probability
  conditional_probability=total_product_miles/total_usage
  #storing conditional probability in dictionary
  probabilities[product]=round(conditional_probability,2)

for product,probability in probabilities.items():
  print(f'Probability that a customer purchased a {product} given that they use treadmill 3 times in a week:',probabilities[product])
```

```
Probability that a customer purchased a KP281 given that they use treadmill 3 times in a week: 0.54
Probability that a customer purchased a KP481 given that they use treadmill 3 times in a week: 0.45
Probability that a customer purchased a KP781 given that they use treadmill 3 times in a week: 0.01
```

What is the probability that a customer purchased a perticular treadmill product(KP281,KP481,KP781)given that they runs 80 miles per week?

```
total = len(df)
products=["KP281","KP481","KP781"]
miles=80
#calculating probability for each product and fitness level
probabilities={}

for product in products:
  #calculating no.of customes who purchased specific product
  total_usage = len(df.loc[df["Miles"]==miles])
  #calculating no. of customers who purchased specific product and who runs 80 miles per week
  total_product_miles=len(df.loc[(df["Product"]==product)&(df["Miles"]==miles)])
  #calculating conditional probability
  conditional_probability=total_product_miles/total_usage
  #storing conditional probability in dictionary
  probabilities[product]=round(conditional_probability,2)

for product,probability in probabilities.items():
  print(f'Probability that a customer purchased a {product} given that they runs 80 miles per week:',probabilities[product])
```

```
Probability that a customer purchased a KP281 given that they runs 80 miles per week: 0.0
Probability that a customer purchased a KP481 given that they runs 80 miles per week: 0.0
Probability that a customer purchased a KP781 given that they runs 80 miles per week: 1.0
```

What is the probability that a customer has fitness level =4 given that they purchased a perticular treadmill product(KP281,KP481,KP781)?

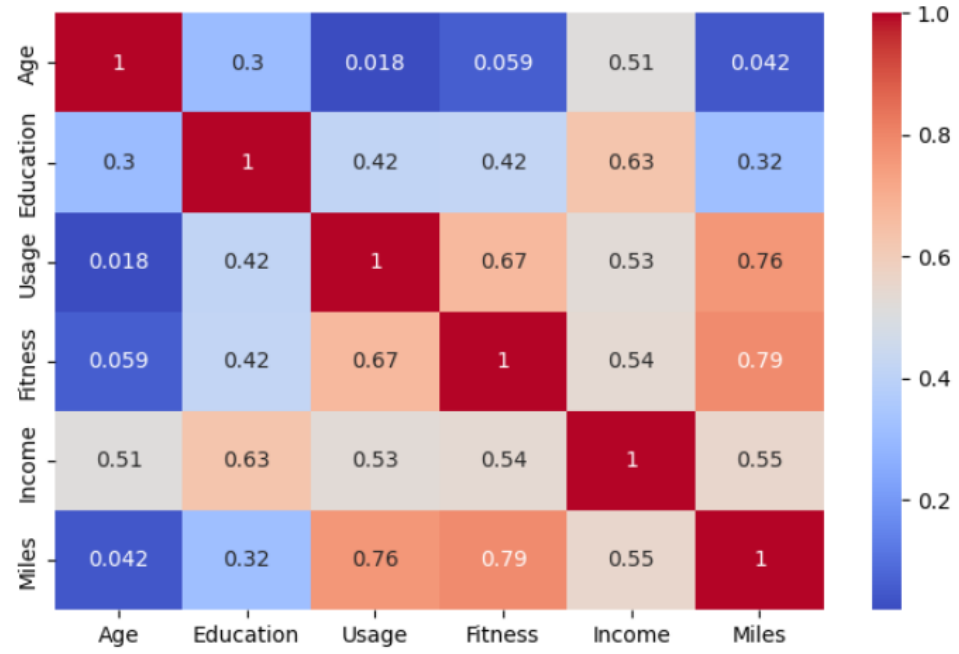
```
total = len(df)
products=["KP281","KP481","KP781"]
fitness_level=4
#calculating probability for each product and fitness level
probabilities={}

for product in products:
  #calculating no.of customes who purchased specific product
  total_usage = len(df.loc[df["Product"]==product])
  #calculating no. of customers who purchased specific product and who have fitness level = 4
  total_product_miles=len(df.loc[(df["Product"]==product)&(df["Fitness"]==fitness_level)])
  #calculating conditional probability
  conditional_probability=total_product_miles/total_usage
  #storing conditional probability in dictionary
  probabilities[product]=round(conditional_probability,2)

for product,probability in probabilities.items():
  print(f'Probability that a customer has fitness_level {fitness_level} given that they purchased a {product}:',probabilities[product])
```

```
Probability that a customer has fitness_level 4 given that they purchased a KP281: 0.11
Probability that a customer has fitness_level 4 given that they purchased a KP481: 0.13
Probability that a customer has fitness_level 4 given that they purchased a KP781: 0.17
```

Checking correlation among different factors

Heatmap

```
[81] import seaborn as sns
     import matplotlib.pyplot as plt

     # Select only numeric columns
     df_numeric = df.select_dtypes(include='number')

     # Plot heatmap of the correlation matrix
     plt.figure(figsize=(8,5))
     sns.heatmap(df_numeric.corr(), annot=True, cmap="coolwarm")
     plt.show()
```



Customer Profiling

#Potential buyers for KP281 Treadmill:

1.Gender:Male and Female

2.Marital Status : Both Parented or single

3.Age : 18-29 as majority of youngsters purchasing KP281 is high

4.Income : 29000-50000 USD as probability of customers belong to low income or moderate income group is high

5.Education : 14-16 years

6.Fitness level : 3

7.Usage : 3 times per week

8.Miles : Runs 60-100 miles per week

#Potential buyers for KP481 Treadmill:

1.Gender:Male and Female

2.Marital Status : Both Parented or single

3.Age : 18-39

4.Income : 29000-80000 USD as probability of customers belong to high income is high

5.Education : 14-16 years

6.Fitness level : 3

7.Usage : 3 times per week

8.Miles : Runs 80-120 miles per week

#Potential buyers for KP781 Treadmill:

1.Gender:Mostly males,low demales

2.Marital Status : Mostly Parented, low singles

3.Age : 18-39

4.Income : 80000 USD and above

5.Education : 16-18 years

6.Fitness level : 5

7.Usage : 3 times per week

8.Miles : Runs 120-200 miles per week

Insights

1.Aerofit has 57.8% male customers and 42.2 female customers.

2.Among the users 44.4% prefer KP281,while 33.3% prefer KP481 and 22.2% prefer KP781 treadmill.

3.KP281 is more affordable and is preferred chice by majority of customers,KP481 is fit for mid level runners,KP781 has advanced features and having high cost.

4.52.6% females uses KP281 treadmill.

5.38.5% males uses KP281 treadmill.

6.59.4% users are single and remaining 40.6% are married.

7.Married customers have higher frequency of purchasing treadmill than singles.

8.Most of the customers falls under "Young Adults" age group.

9.Customers from low income groups highly prefer KP281 treadmill.

10.Customers with fitness level 3 prefer KP281 and KP481 and who having high fitness level that 5 prefers KP781.

11.Customers who runs 60-100 miles per week pefer KP281 treadmill, Customers who runs 60-120 miles per week pefer KP481 treadmill,Customers who runs 120-200 miles per week pefer KP781 treadmill.

12.customes who use treadmill 3 times a week prefers KP281 and KP481.customes who use treadmill 4-5 times a week prefers KP781.

Recommendations

1.Focus on advertisement and promotions that appeal women.

2.Provide special offers and discounts for customers loooking for cost effective options.

3.Provide convinient EMI payment options for KP781 treadmill.This will allow low and moderate income group customers to spread the cost over sevaral months.

4.Offer personalised assistance to help users who are little high in age.

5.Showcase female friendly features and benefits of treadmill to attract more female customers.

6.Focus on marketing efforts for mid level runners.

7.Lets engage with fitness communities online to showcase KP281's appeal to beginners.

8.Focus on budget friendly nature.

9.Hihlight key features of KP281 to attract more customers.

10.Appoint some famous sport male and female celebrity as brand ambassador for Aerofit.