AEROFIT: Descriptive Stats & Probability

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

```
Downloading...
From: https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/125/original/aerofit_treadmill.csv?1639992749
To: /content/aerofit_treadmill.csv?1639992749
100% 7.28k/7.28k [00:00<00:00, 23.7MB/s]
```

```
# 1) Defining Problem Statement and Analysing basic metrics:
```

Problem Statement:

The problem at hand is to analyze and understand the factors influencing the purchase decision of a treadmill product (KP281, KP481, or KP781) based on various customer attributes. The goal is to uncover patterns and insights that can aid in optimizing marketing strategies and product offerings by using the following datas.

Product Purchased: All three products (KP281, KP481, KP781) are **represented** in the dataset, indicating a diverse range of offerings.

Age: The age range provides the understanding to find the target age group to the selective products.

Gender: Understanding the gender distribution helps to promote the various products cater to the preferences of different genders.

Education: The range of education years gives an indication of the educational background of the customer base.

MaritalStatus: The distribution between Single and Partnered customers can impact marketing messages and product features.

Usage: The distribution of usage patterns helps in understanding how frequently customers using the product in a week, influencing marketing and product development.

Income: Understanding income distribution provides insights into the purchasing capacity of the customer base, affecting pricing and promotional strategies.

Fitness: The distribution of self-rated fitness levels helps in identifying the fitness-conscious segment of customers.

Miles: Understanding the expected miles reveals the intensity of expected product usage, guiding product design and feature decisions.

```
import pandas as pd
import numpy as np
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

```
#shape of the data
df.shape
```

```
(180, 9)
```

```
#size of the data
df.size
```

```
1620
```

```
#info of the data
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
```

```
#statistical summary
df.describe()
```

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

```
#data types:
df.dtypes
```

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

```
#Coverting intp categorical data:

Categories = ['Product','Gender','MaritalStatus']
df_categories= df[Categories].astype('category')
df_categories.dtypes
```

```
Product          category
Gender           category
MaritalStatus    category
dtype: object
```

```
# Chcking the NULL value:
df.isna().sum()
```

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

```
Miles            0
dtype: int64
```

```
# Probability of the Genders with respect to the Products:

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

```
Product  Gender
KP281    Female    50.00
         Male      50.00
KP481    Male      51.67
         Female    48.33
KP781    Male      82.50
         Female    17.50
Name: Gender, dtype: float64
```

```
# Probability of the MaritalStatus with respect to the Products:

df.groupby('Product')['MaritalStatus'].value_counts(normalize = True)*100
```

```
Product  MaritalStatus
KP281    Partnered        60.0
         Single           40.0
KP481    Partnered        60.0
         Single           40.0
KP781    Partnered        57.5
         Single           42.5
Name: MaritalStatus, dtype: float64
```

```
# Probability of customers have purchased KP281, KP481, or KP781:

marginal_prob_table = pd.crosstab(index=df['Product'], columns='Count', normalize=True)* 100
round(marginal_prob_table,2)
```

col_0	Count
Product	
KP281	44.44
KP481	33.33
KP781	22.22

```
# 2) Non-Graphical Analysis: Value counts and unique attributes
```

```
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

```
# Probability of customers have purchased KP281, KP481, or KP781:

a = round(df['Product'].value_counts(normalize = True),2)*100
b = df['Product'].nunique()
```

```
Total_product_counts:

 KP281    44.0
KP481    33.0
KP781    22.0
Name: Product, dtype: float64

unique_products = 3
```

```
# Probability of Genders of customers have purchased KP281, KP481, or KP781:

a = round(df.groupby('Product')['Gender'].value_counts(normalize = True),2)*100
b = df['Gender'].nunique()

print('Total_Gender_counts:\n\n',a)
print('\nunique_Gender =', b)
```

```
Total_Gender_counts:

 Product  Gender
KP281    Female    50.0
         Male      50.0
KP481    Male      52.0
         Female    48.0
KP781    Male      82.0
         Female    18.0
Name: Gender, dtype: float64

unique_Gender = 2
```

```
# Probability of MaritalStatus of customers have purchased KP281, KP481, or KP781:

a = df.groupby('Product')['MaritalStatus'].value_counts(normalize = True)*100
b = df['MaritalStatus'].nunique()

print('Total_MaritalStatus_counts:\n\n',a)
print('\nunique_MaritalStatus =', b)
```

```
Total_MaritalStatus_counts:

 Product  MaritalStatus
KP281    Partnered        60.0
         Single           40.0
KP481    Partnered        60.0
         Single           40.0
KP781    Partnered        57.5
         Single           42.5
Name: MaritalStatus, dtype: float64

unique_MaritalStatus = 2
```

```
# Probability of Education of customers have purchased KP281, KP481, or KP781:

a = round(df.groupby('Product')['Education'].value_counts(normalize = True),2)*100
b = df['Education'].nunique()

print('Total_Education_counts:\n\n',a)
print('\nunique_Education =', b)
```

```
Total_Education_counts:

 Product  Education
KP281    16           49.0
         14           38.0
         15            5.0
         13            4.0
         12            2.0
         18            2.0
KP481    16           52.0
         14           38.0
         13            3.0
         18            3.0
         12            2.0
         15            2.0
KP781    18           48.0
         16           38.0
         21            8.0
         14            5.0
         20            2.0
Name: Education, dtype: float64

unique_Education = 8
```

```
# Probability of Income of customers have purchased KP281, KP481, or KP781:

a = df.groupby('Product')['Income'].value_counts(normalize = True)*100
b = df['Income'].nunique()

print('Total_Income_counts:\n\n',a)
print('\nunique_Income =', b)
```

```
Total_Income_counts:

 Product  Income
```

```
KP281    46617     8.75
         54576     8.75
         52302     7.50
         35247     6.25
         45480     6.25
                   ...
KP781    85906     2.50
         95508     2.50
         95866     2.50
         99601     2.50
         103336    2.50
Name: Income, Length: 83, dtype: float64

unique_Income = 62
```

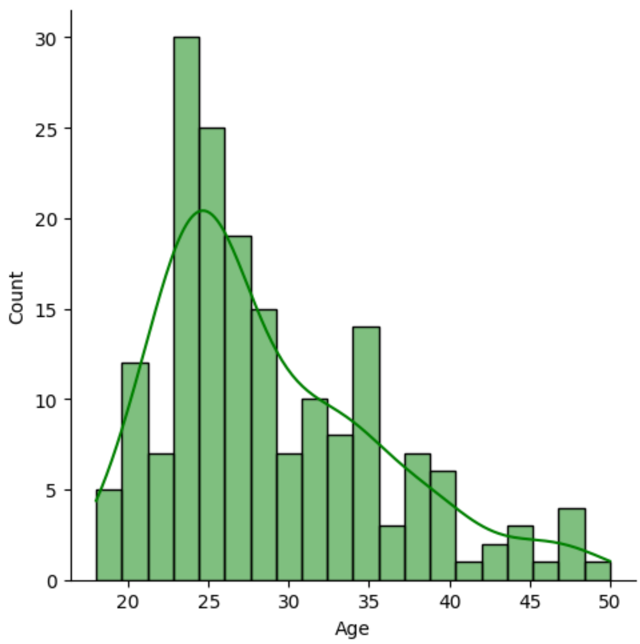
3) ***Visual Analysis - Univariate & Bivariate***

- For continuous variable(s): Distplot, countplot, histogram for univariate analysis
- For categorical variable(s): Boxplot
- For correlation: Heatmaps, Pairplots

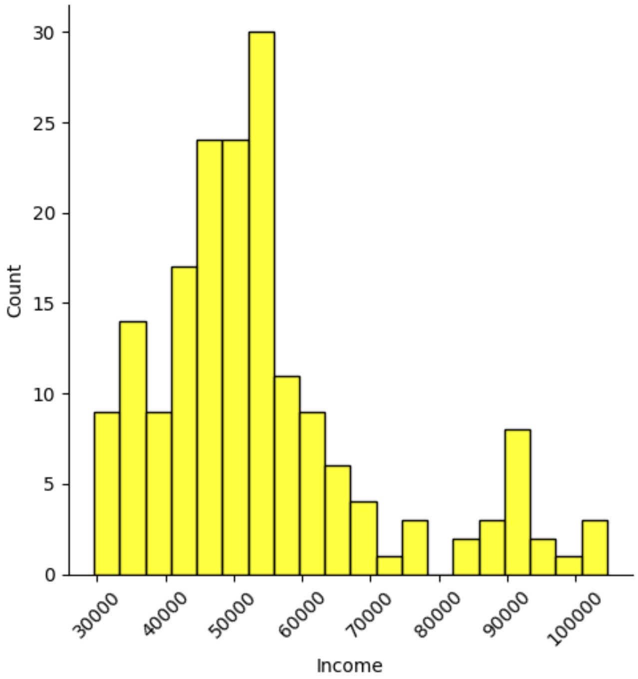
```
df.head()
```

	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

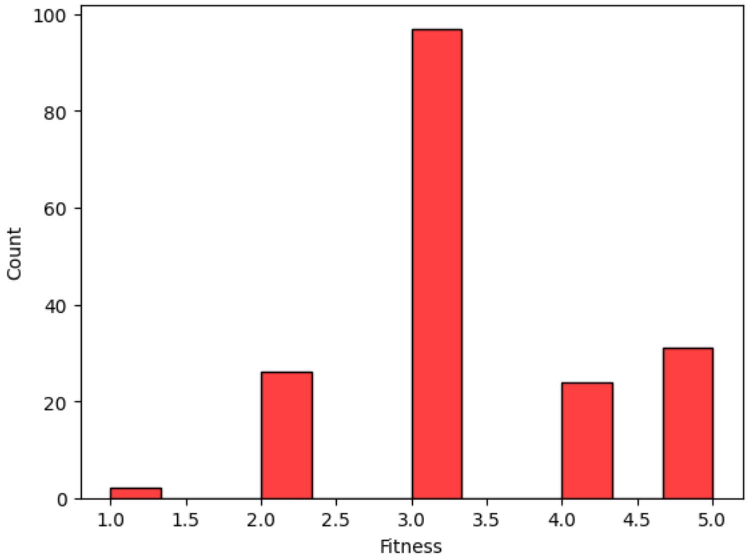
```
# Age Distributional Count:

sns.displot(data =df,x='Age',kde =True,bins = 20,color = 'green')
plt.show()
```



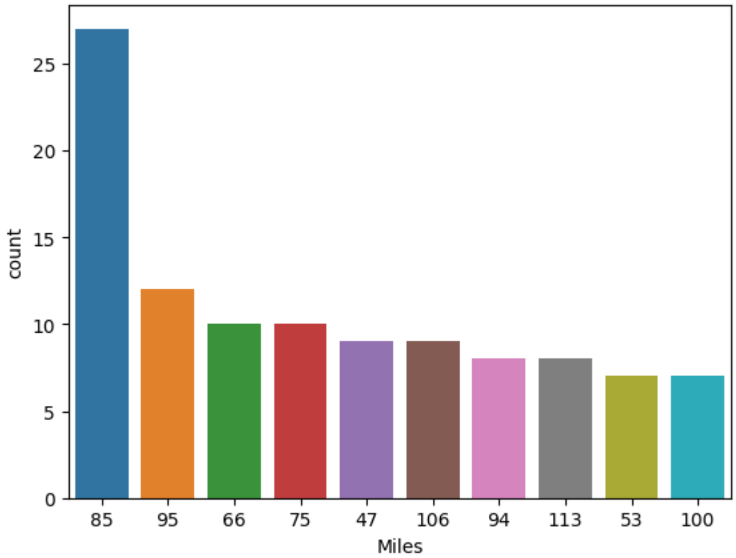

```
# Income Distributional Count:

sns.displot(data =df,x='Income',bins = 20,color = 'yellow')
plt.xticks(rotation = 45)
plt.show()
```



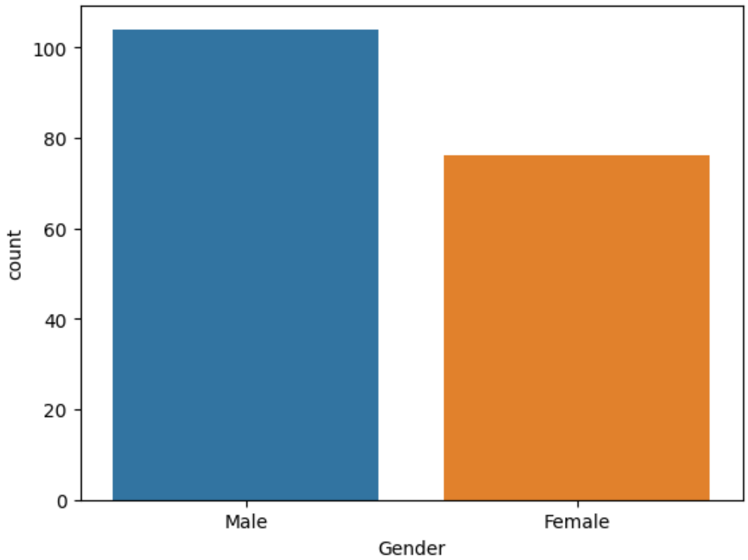

```
# Fitness Distributional Count:

sns.histplot(data =df,x='Fitness',color = 'red')
plt.show()
```



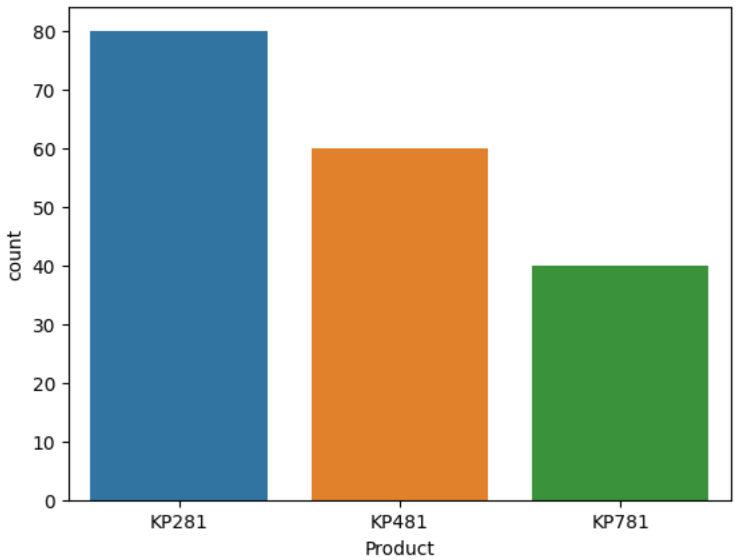

```
# Miles Distributional Count:

sns.countplot(data =df,x='Miles',order = df['Miles'].value_counts().index[:10])
plt.show()
```



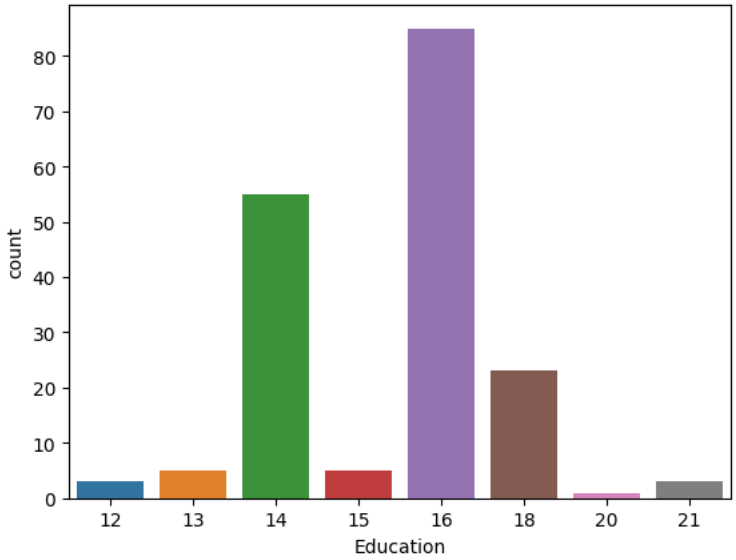

```
# Gender Distributional Count:

sns.countplot(data =df,x='Gender')
plt.show()
```



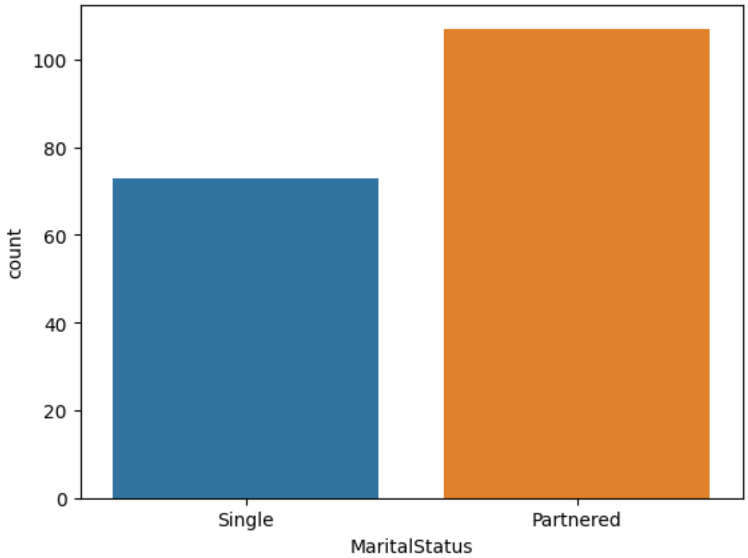

```
# Products Distributional Count:

sns.countplot(data =df,x='Product')
plt.show()
```



```
# Education Distributional Count:

sns.countplot(data =df,x='Education')
plt.show()
```



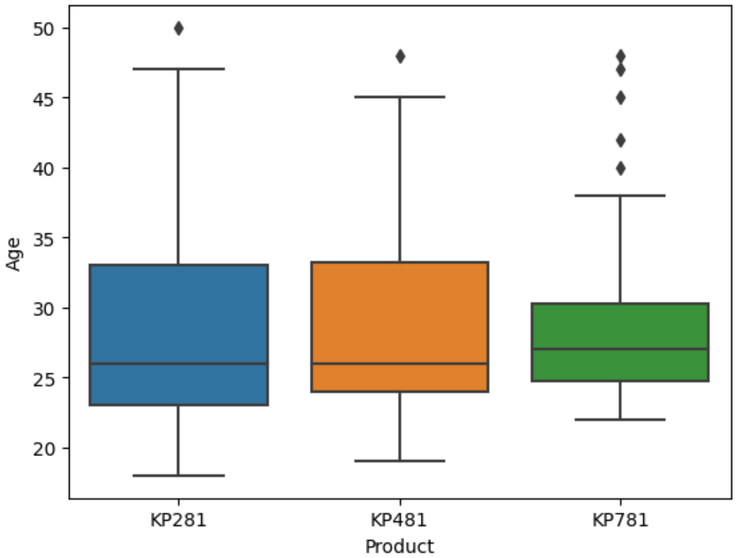

```
# MaritalStatus Distributional Count:

sns.countplot(data =df,x='MaritalStatus')
plt.show()
```



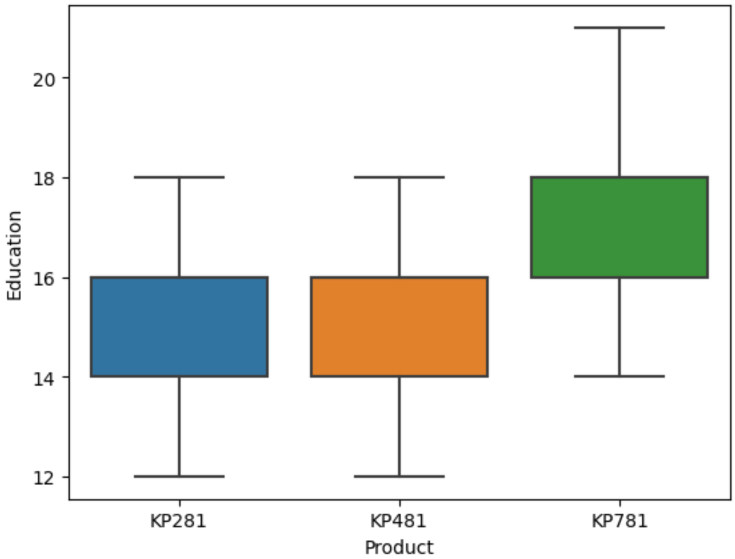

```
df.head(3)
```

	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

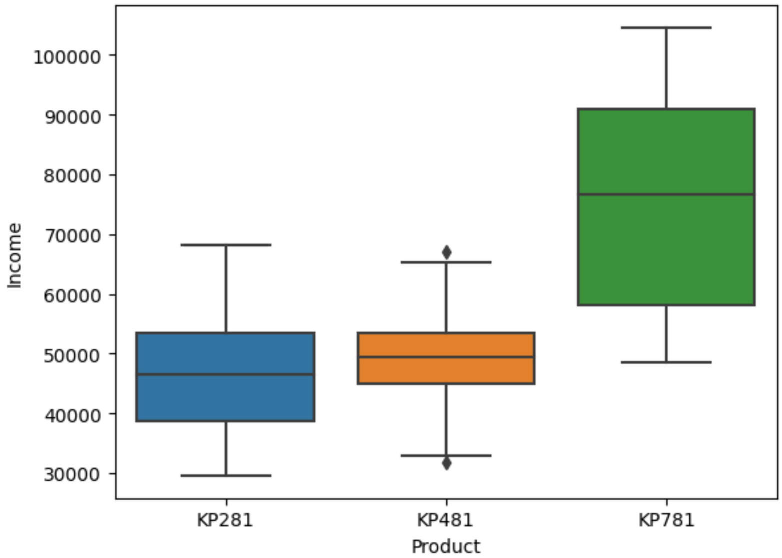
```
# Product distribution with respect to the Customer's Age:

sns.boxplot(data=df,x ='Product',y= 'Age')
plt.show()
```



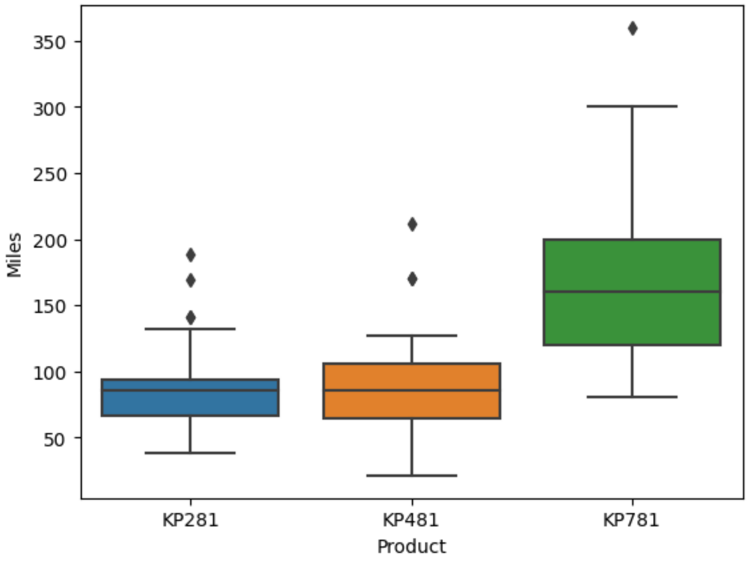

```
# Product distribution with respect to the Customer's Education:

sns.boxplot(data=df,x ='Product',y= 'Education')
plt.show()
```



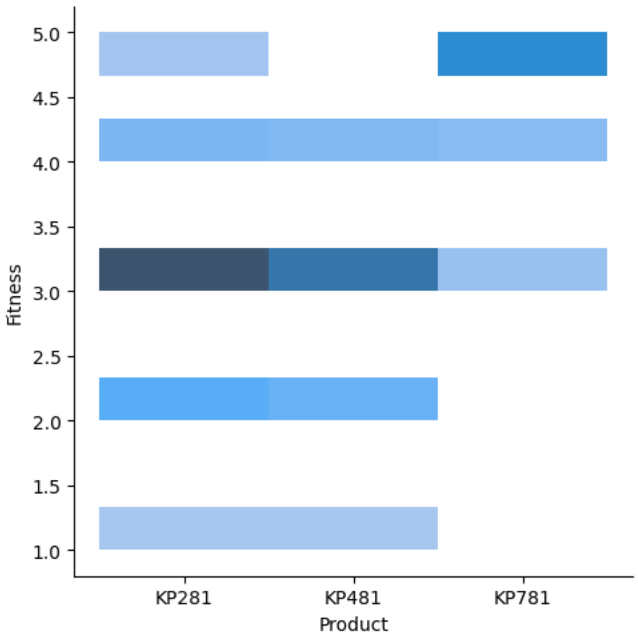

```
# Product distribution with respect to the Customer's Income:

sns.boxplot(data=df,x ='Product',y= 'Income')
plt.show()
```



```
# Product distribution with respect to the Customer's Miles:

sns.boxplot(data=df,x ='Product',y= 'Miles')
plt.show()
```

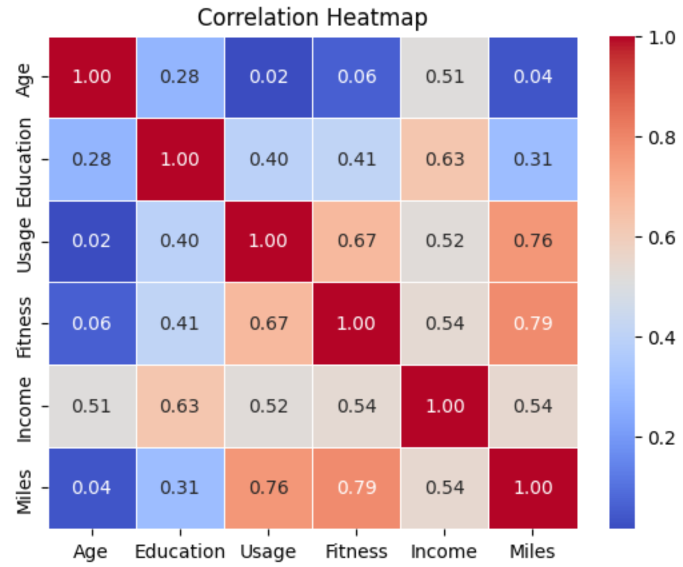



```
# Product distribution with respect to the Customer's Fitness:

sns.displot(data=df,x ='Product',y= 'Fitness')
plt.show()
```



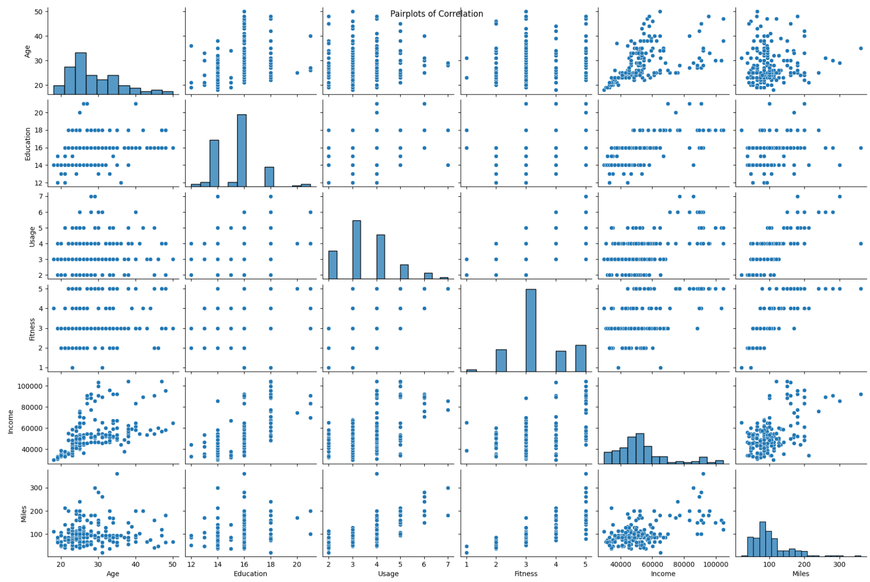

```
# Correlation by using HeatMap:

correlation_matrix = df.corr()

sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)
plt.title('Correlation Heatmap')
plt.show()
```

```
<ipython-input-122-54dcbbd3e82c>:1: FutureWarning: The default value of numeric_only
  correlation_matrix = df.corr()
```



	Age	Education	Usage	Fitness	Income	Miles
Age	1.00	0.28	0.02	0.06	0.51	0.04
Education	0.28	1.00	0.40	0.41	0.63	0.31
Usage	0.02	0.40	1.00	0.67	0.52	0.76
Fitness	0.06	0.41	0.67	1.00	0.54	0.79
Income	0.51	0.63	0.52	0.54	1.00	0.54
Miles	0.04	0.31	0.76	0.79	0.54	1.00

```
# Correlation by using Pairplot:

sns.pairplot(df, height=2,aspect = 1.5)
plt.suptitle('Pairplots of Correlation')

plt.show()
```



```
# 4) Missing Value & Outlier Detection:
```

```
# Missing Values from the given data:

Missing_value = df.isna().sum()
Missing_value
```

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

```
# Outlier Detection for the dataset:

Q3 = np.percentile(df[['Age','Usage','Fitness','Income','Miles']],75)
Q1 = np.percentile(df[['Age','Usage','Fitness','Income','Miles']],25)

IQR = Q3 - Q1
print('IQR =',IQR)

Upper = Q3 + 1.5*IQR
Lower = Q1 - 1.5*IQR
print('Upper =',Upper)
print('Lower =',Lower)
```

```
IQR = 111.75
Upper = 282.375
Lower = -164.625
```

```
# Outlier Detection for the Age in the dataset:

Q3 = np.percentile(df['Age'],75)
Q1 = np.percentile(df['Age'],25)

IQR = Q3 - Q1
print('IQR =',IQR)

Upper = Q3 + 1.5*IQR
Lower = Q1 - 1.5*IQR
print('Upper =',Upper)
print('Lower =',Lower)
```

```
IQR = 9.0
Upper = 46.5
Lower = 10.5
```

```
# Outlier Detection for the Usage in the dataset:

Q3 = np.percentile(df['Usage'],75)
Q1 = np.percentile(df['Usage'],25)

IQR = Q3 - Q1
print('IQR =',IQR)

Upper = Q3 + 1.5*IQR
Lower = Q1 - 1.5*IQR
print('Upper =',Upper)
print('Lower =',Lower)
```

```
IQR = 1.0
Upper = 5.5
Lower = 1.5
```

```
# Outlier Detection for the Fitness in the dataset:

Q3 = np.percentile(df['Fitness'],75)
Q1 = np.percentile(df['Fitness'],25)

IQR = Q3 - Q1
print('IQR =',IQR)

Upper = Q3 + 1.5*IQR
Lower = Q1 - 1.5*IQR
print('Upper =',Upper)
print('Lower =',Lower)
```

```
IQR = 1.0
Upper = 5.5
Lower = 1.5
```

```
# Outlier Detection for the Income in the dataset:

Q3 = np.percentile(df['Income'],75)
Q1 = np.percentile(df['Income'],25)

IQR = Q3 - Q1
print('IQR =',IQR)

Upper = Q3 + 1.5*IQR
Lower = Q1 - 1.5*IQR
print('Upper =',Upper)
print('Lower =',Lower)
```

```
IQR = 14609.25
Upper = 80581.875
Lower = 22144.875
```

```
# Outlier Detection for the Miles in the dataset:

Q3 = np.percentile(df['Miles'],75)
Q1 = np.percentile(df['Miles'],25)

IQR = Q3 - Q1
print('IQR =',IQR)

Upper = Q3 + 1.5*IQR
Lower = Q1 - 1.5*IQR
print('Upper =',Upper)
print('Lower =',Lower)
```

```
IQR = 48.75
Upper = 187.875
Lower = -7.125
```

```
# Conditional Probability:

df.head()
```

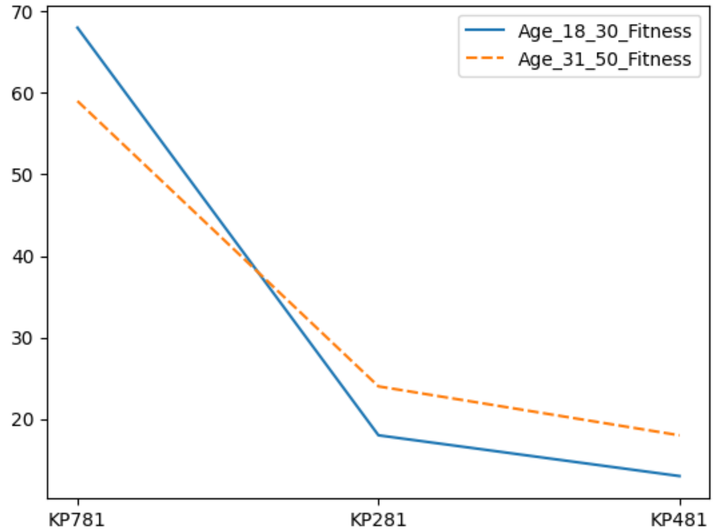
	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
0	KP281	18	Male	14	Single	3	4	29562	112
1	KP281	19	Male	15	Single	2	3	31836	75
2	KP281	19	Female	14	Partnered	4	3	30699	66
3	KP281	19	Male	12	Single	3	3	32973	85
4	KP281	20	Male	13	Partnered	4	2	35247	47

```
# Probability of the Age and the Fitness of the customers purchased the products:

Age_18_30_Fitness = df[(df['Age'] >=18) & (df['Age']<=30) & (df['Fitness']>=4)]['Product'].value_counts(normalize = True)*100
Age_31_50_Fitness = df[(df['Age'] >=31) & (df['Age']<=50) & (df['Fitness']>=4)]['Product'].value_counts(normalize = True)*100

data = pd.DataFrame({'Age_18_30_Fitness':round(df[(df['Age'] >=18) & (df['Age']<=30) & (df['Fitness']>=4)]['Product'].value_counts(norma
data
```

	Age_18_30_Fitness	Age_31_50_Fitness
KP781	68.0	59.0
KP281	18.0	24.0
KP481	13.0	18.0

```
sns.lineplot(data)
```

<Axes: >

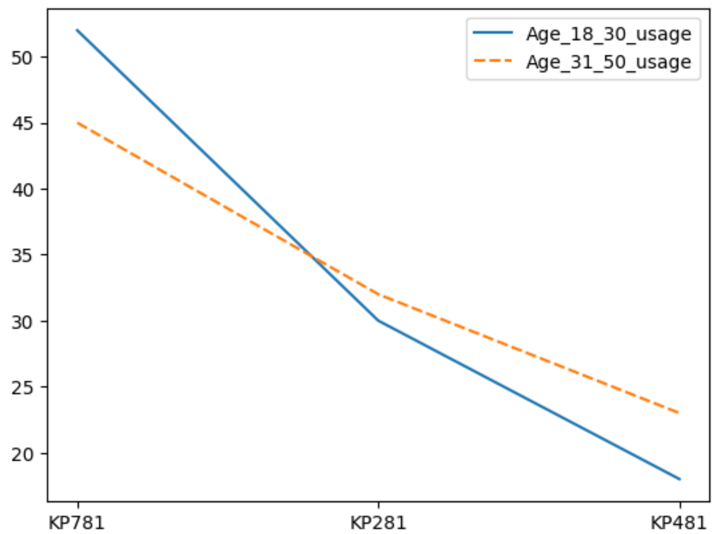



```
# Probability of the Age and the Usage of the customers purchased the products:

Age_18_30_usage = round(df[(df['Age'] >=18) & (df['Age']<=30) &(df['Usage']>= 4)]['Product'].value_counts(normalize = True),2)*100
Age_31_50_usage = round(df[(df['Age'] >=31) & (df['Age']<=50) &(df['Usage']>= 4)]['Product'].value_counts(normalize = True),2)*100

data = pd.DataFrame({'Age_18_30_usage': round(df[(df['Age'] >=18) & (df['Age']<=30) &(df['Usage']>= 4)]['Product'].value_counts(normali:
data
```

	Age_18_30_usage	Age_31_50_usage
KP781	52.0	45.0
KP281	30.0	32.0
KP481	18.0	23.0

```
sns.lineplot(data)
```

<Axes: >

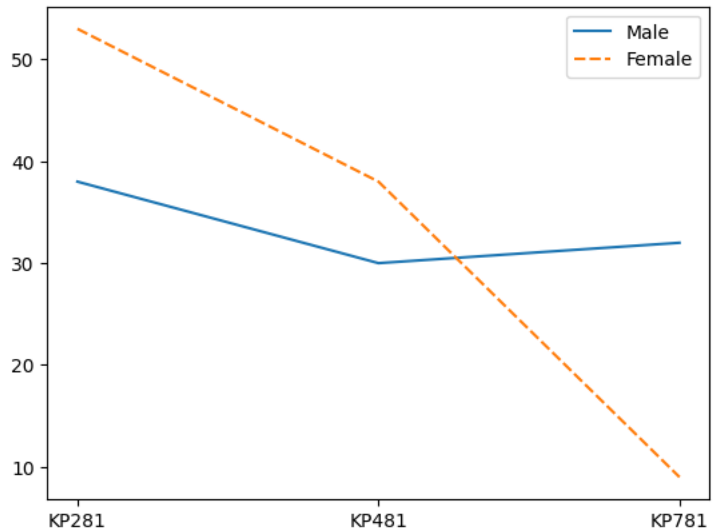


```
# Probability of Genders of the customers purchased the products:

Male = df[df['Gender'] == 'Male']['Product'].value_counts(normalize = True)*100
Female = df[df['Gender'] == 'Female']['Product'].value_counts(normalize = True)*100

data = pd.DataFrame({'Male': round(df[df['Gender'] == 'Male']['Product'].value_counts(normalize = True),2)*100, 'Female' : round(df[df[
data
```

	Male	Female
KP281	38.0	53.0
KP481	30.0	38.0
KP781	32.0	9.0

```
sns.lineplot(data)
```

<Axes: >

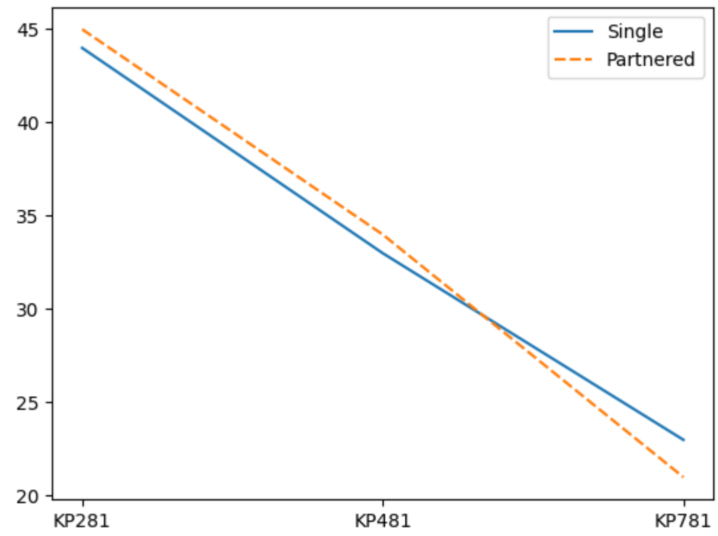



```
# Probability of MaritalStatus of the customers purchased the products:

Single = df[df['MaritalStatus'] == 'Single']['Product'].value_counts(normalize = True)*100
Partnered = df[df['MaritalStatus'] == 'Partnered']['Product'].value_counts(normalize = True)*100

data = pd.DataFrame({'Single': round(df[df['MaritalStatus'] == 'Single']['Product'].value_counts(normalize = True),2)*100, 'Partnered'
data
```

	Single	Partnered
KP281	44.0	45.0
KP481	33.0	34.0
KP781	23.0	21.0

```
sns.lineplot(data)
```

```
<Axes: >
```




```
df.head(3)
```

	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

Business Insights based on Non-Graphical and Visual Analysis :

1) Range of Attributes:

Product Purchased: All three products (KP281, KP481, KP781) are **represented** in the dataset, indicating a diverse range of offerings.

Age: The age range provides the understanding to find the target age group to the selective products.

Gender: Understanding the gender distribution helps to promote the various products cater to the preferences of different genders.

Education: The range of education years gives an indication of the educational background of the customer base.

MaritalStatus: The distribution between Single and Partnered customers can impact marketing messages and product features.

2) Distribution of the variables and relationship between them:

Usage: The distribution of usage patterns helps in understanding how frequently customers using the product in a week, influencing marketing and product development.

Income: Understanding income distribution provides insights into the purchasing capacity of the customer base, affecting pricing and promotional strategies.

Fitness: The distribution of self-rated fitness levels helps in identifying the fitness-conscious segment of customers.

Miles: Understanding the expected miles reveals the intensity of expected product usage, guiding product design and feature decisions.

3) Univariate Plots:

Age Distributional Count:

- The age categories of the customer starts from 18 and ends with 50.
- In the overall the Highest average age is between the 24 and 26.

Income Distributional Count:

- The Income of the Customer purchased the products are starts from the min 30000 and ends with the max more than 1lakh.
- The Hightest Count of Customers Income who purchased the products where income in between 50k to 60k.

Fitness Distributional Count:

- The Highest number of Customer purchased the products having the fitness ratings of 3 to 3.5.
- This helps the marketing team to promote more and create the needs and demands to the average fitness customers.

Miles Distributional Count:

- The Highest number of Customer purchased the products having the expected miles of 85 per week.
- This indictes to promote the products to the other customers.

MaritalStatus Distributional Count:

- In the purchased customer list, Partnered status is seems to higher when compared to the Single.
- This would help the marketing team to promote more to the Partnered when compared to the Single.

Bivariate Plots:

Product distribution with respect to the Customer's Age:

- The average age category for Aerofit products falls within the range of 25 to 27.
- The distribution of KP281 appears to be higher compared to the other two products.
- This suggests that KP781 attracts customers starting from the age of 23. For customers below this age, the marketing team should focus on promoting KP281 and KP481.

Product distribution with respect to the Customer's Education:

- KP281 and KP481 have educational counts ranging from 14 to 16.
- KP781 attracts customers with a good educational background.

Product distribution with respect to the Customer's Income:

- KP781 attracts customers with incomes crossing 70k, which is higher compared to the average incomes of the other two products, ranging between 45k to 50k.

Product distribution with respect to the Customer's Mil