aerofit-3

November 22, 2023

AEROFIT CASE STUDY

Problem Statement: The problem at hand involves understanding the factors that influence customers' decisions to purchase specific treadmill products (KP281, KP481, or KP781). The dataset provides information on various customer attributes, and the goal is to gain insights that can inform marketing strategies, product development, and customer engagement.

Basic Metrics for Analysis:

Product Distribution:

Analyze the distribution of purchases among the three treadmill products (KP281, KP481, KP781). Calculate the percentage of customers for each product.

Age Distribution:

Explore the age distribution of customers. Identify the average age, the age range with the highest number of customers, and any notable patterns.

Gender Distribution:

Examine the gender distribution among customers. Calculate the percentage of male and female customers.

Education Levels:

Investigate the distribution of education levels among customers. Identify the most common education level and any correlations with product purchases.

Marital Status:

Analyze the distribution of marital status (single or partnered) among customers. Explore whether marital status influences product choices.

Usage Plans:

Study the distribution of the average number of times customers plan to use the treadmill each week. Identify patterns and preferences in usage frequency.

Income Levels:

Explore the distribution of annual incomes among customers. Identify income brackets and analyze their correlation with product purchases.

Fitness Ratings:

Examine the self-rated fitness levels of customers on a scale of 1 to 5. Identify the most common fitness level and explore if fitness ratings correlate with product choices.

Expected Weekly Mileage:

Analyze the distribution of the average number of miles customers expect to walk/run each week. Identify patterns and preferences in expected weekly mileage.

[ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

[ ]:
```
#importing the data
data=pd.read_csv("/content/aerofit_treadmill.csv")
```

[ ]:
```
data.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

[ ]:
```
df=data.copy()
```

[ ]:
```
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 data
print(f"The shape of the give data is :{df.shape}")
```

```
The shape of the give data is :(180, 9)
```

[ ]:
```
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
```

[ ]:
```
# data types of all the attributes
data_types = df.dtypes
print("Data types :")
print(data_types)
```

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

[ ]:
```
#conversion of categorical attributes to 'category'
catergorical_attributes = ["Gender","Product","MaritalStatus"]
for category in catergorical_attributes:
    df[category] = df[category].astype('category')
```

With the help of **astype("category")** we can convert the required attributes into the categorical attributes. Here we have converted the **Gender,Product,Marital Status** into categorical attributes

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

[ ]:
```
Product          category
Age                 int64
Gender           category
Education           int64
MaritalStatus    category
Usage               int64
Fitness             int64
```

```
Income              int64
Miles               int64
dtype: object
```

[ ]:
```
#statistical summary
df.describe()
```

[ ]:
```
                Age   Education       Usage     Fitness         Income  \
count  180.000000  180.000000  180.000000  180.000000     180.000000
mean    28.788889   15.572222    3.455556    3.311111   53719.577778
std      6.943498    1.617055    1.084797    0.958869   16506.684226
min     18.000000   12.000000    2.000000    1.000000   29562.000000
25%     24.000000   14.000000    3.000000    3.000000   44058.750000
50%     26.000000   16.000000    3.000000    3.000000   50596.500000
75%     33.000000   16.000000    4.000000    4.000000   58668.000000
max     50.000000   21.000000    7.000000    5.000000  104581.000000

            Miles
count  180.000000
mean   103.194444
std     51.863605
min     21.000000
25%     66.000000
50%     94.000000
75%    114.750000
max    360.000000
```

With the help of **describe()** we can come to know some basic statistical data like count,mean,std,min,max etc for numerical data . For the given data attributes like **Age,Education,Usage ,Fitness, Income,Miles** we got some statistical information.

[ ]:
```
categorical_summary = data[catergorical_attributes].describe()
categorical_summary
```

[ ]:
```
       Gender Product MaritalStatus
count     180     180           180
unique      2       3             2
top      Male   KP281     Partnered
freq      104      80           107
```

For the **categorical data type** we will come to know the **count,unique , top,freq** as the statistical information. The above table describes the statistical data for the categorical attributes Gender,Product,Marital Status.

[ ]:
```
#Non-Graphical Analysis: Value counts and unique attributes
df.Product.value_counts()
```

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

For the Product attribute by using value_counts we came to know that **KP281 is 80,KP481 is 60 and KP781 is 40 out of 180**.By this we came to know that The customers are showing much interest in KP281 when compared to KP781.

[ ]:
```
df.Gender.value_counts()
```

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

By using the value_counts for **Gender attribute** we can conclude that out of 180 the male count is 104 and female count is 76.

[ ]:
```
df.MaritalStatus.value_counts()
```

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

By using the value_counts for **Marital Status** attribute we can say that out of 180 the Partnered count is 107 and Single count is 73.

[ ]:
```
#unique attributes
age_unique=df.Age.nunique()
print(f"Unique count of Age:{age_unique}")
```

```
Unique count of Age:32
```

For the **Age** attribute totally we arer having **32 unique values**

[ ]:
```
edu_unique=df.Education.nunique()
print(f"Unique count of Education:{edu_unique}")
```

```
Unique count of Education:8
```

For the **Education** attribute totally we arer having **8 unique values**

[ ]:
```
usg_unique=df.Usage.nunique()
print(f"Unique count of Usage:{usg_unique}")
```

```
Unique count of Usage:6
```

For the **Usage** attribute totally we arer having 6 unique values

[ ]:
```
fit_unique=df.Fitness.nunique()
print(f"Unique count of Fitness:{fit_unique}")
```

```
Unique count of Fitness:5
```

For the **Fitness** attribute totally we arer having **5 unique values** . The fitness** range is from 1-5**

[ ]:
```
inc_unique=df.Income.nunique()
print(f"Unique count of Income:{inc_unique}")
```

```
Unique count of Income:62
```

For the **Income** attribute totally we arer having **62 unique values**.The minimun income is 29562 and the maximum income is 104581

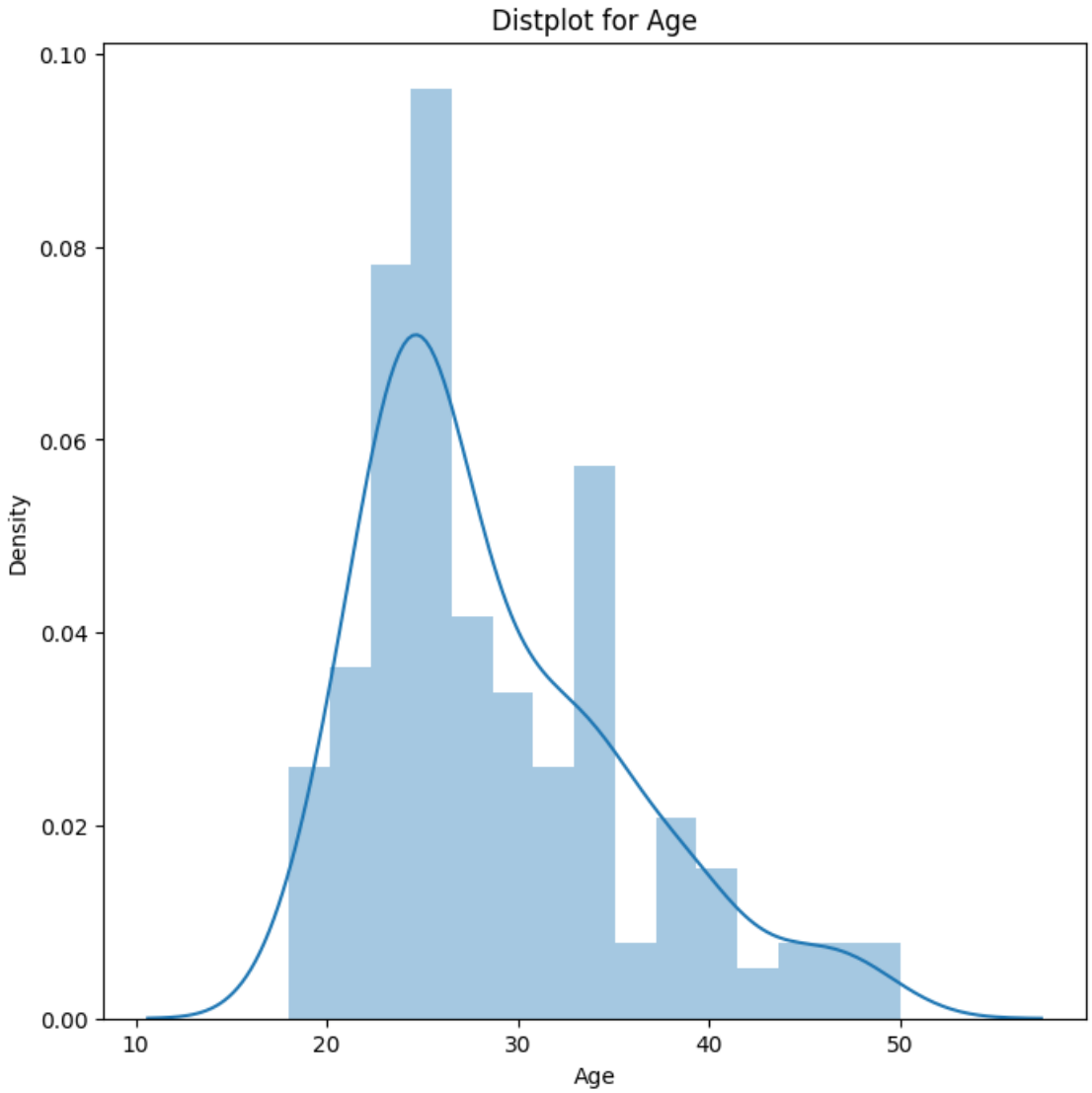
[ ]:
```
mil_unique=df.Miles.nunique()
print(f"Unique count of Miles:{mil_unique}")
```

```
Unique count of Miles:37
```

For the **Miles** attribute totally we arer having **37 unique values**.The minimun mile is 21 and the maximum mile is 360.

[ ]:
```
#For continuous variable(s): Distplot, countplot, histogram for univariate␣
↪analysis (10 Points)
```

[ ]:
```
#Distplot
plt.figure(figsize=(8, 8))
sns.distplot(df.Age,bins=15,kde=True)
plt.title('Distplot for Age')
plt.show()
```



The **distplot** function is designed to display the distribution of a univariate set of observations.

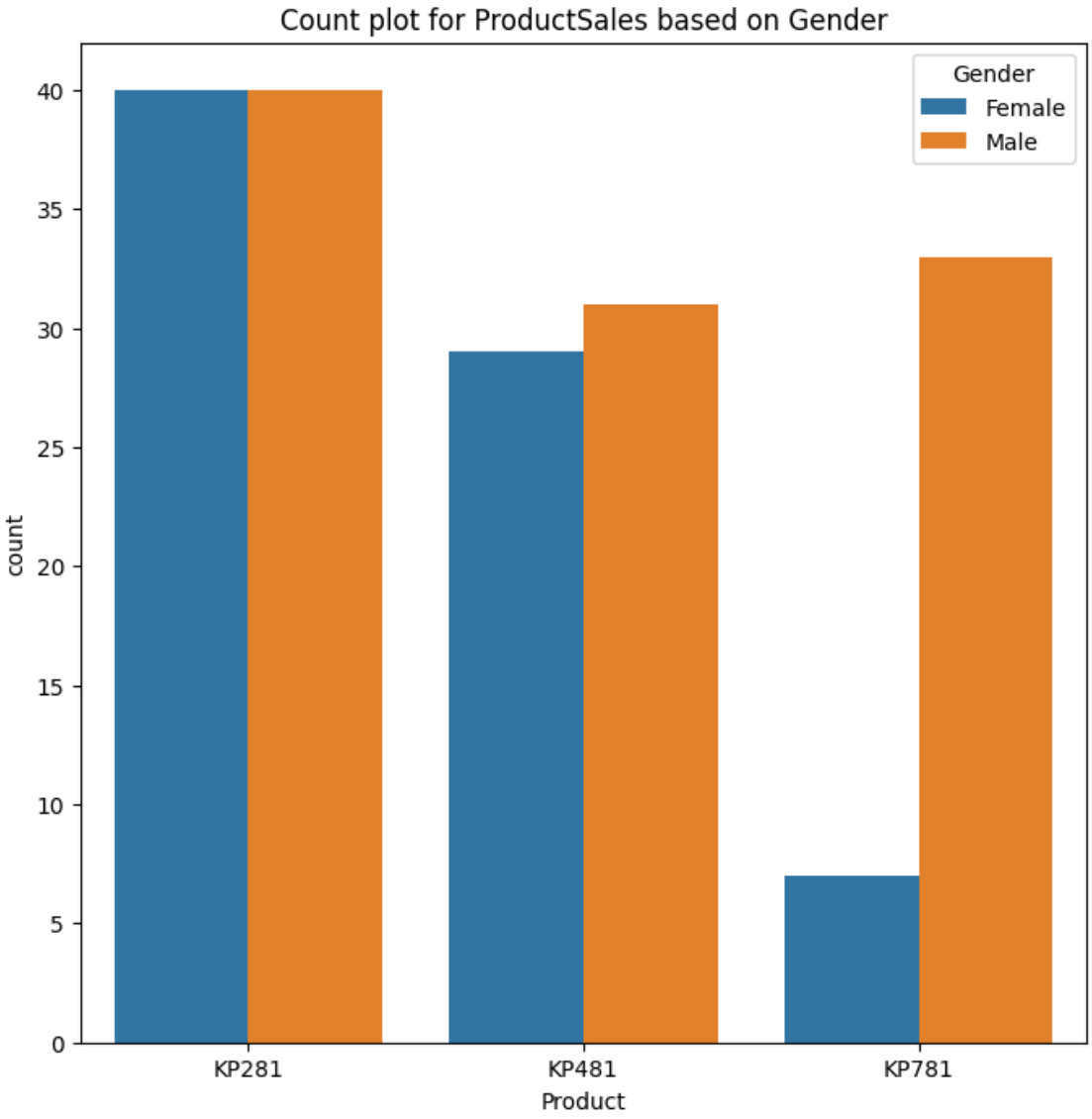
Here we created distplot for Age attribute. from this we say that age between 20 and 30 are maximum.

[ ]:
```
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

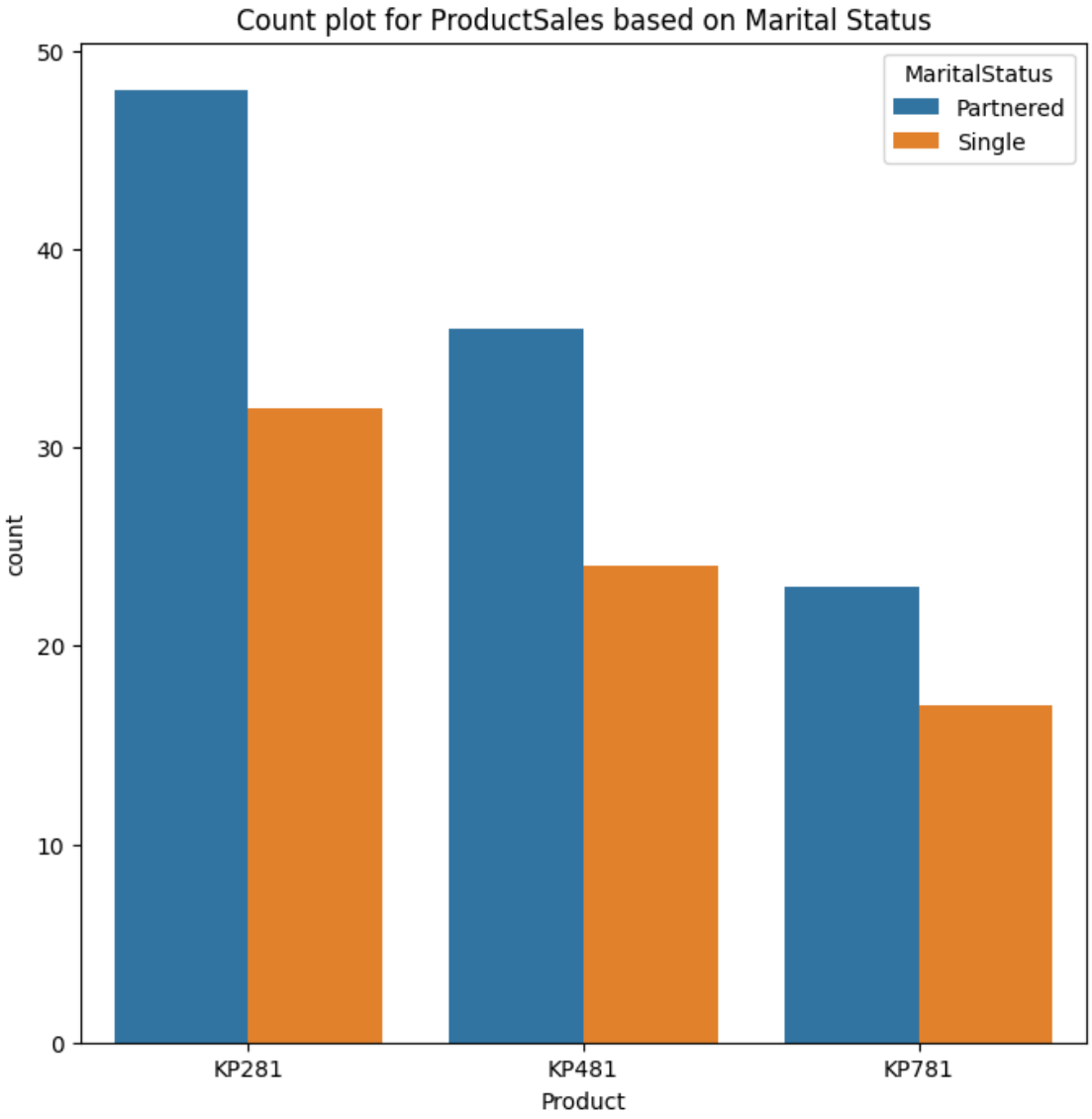
```
[ ]: #countplot
plt.figure(figsize=(8, 8))
sns.countplot(df,x="Product",hue="Gender")
plt.title("Count plot for ProductSales based on Gender")
plt.show()
```



A **countplot** is a type of categorical plot in seaborn that is specifically designed for counting the occurrences of each category in a categorical variable. It provides a simple way to visualize the distribution of categorical data by displaying the count of observations in each category as bars.

Here the countplot represents the ProductSales based on Gender.From this we can say both Males and Females are equal for KP281, where as in KP481 and KP781 the females count is less when compared to men.

```
[ ]: plt.figure(figsize=(8, 8))
     sns.countplot(df,x="Product",hue="MaritalStatus")
     plt.title("Count plot for ProductSales based on Marital Status")
     plt.show()
```

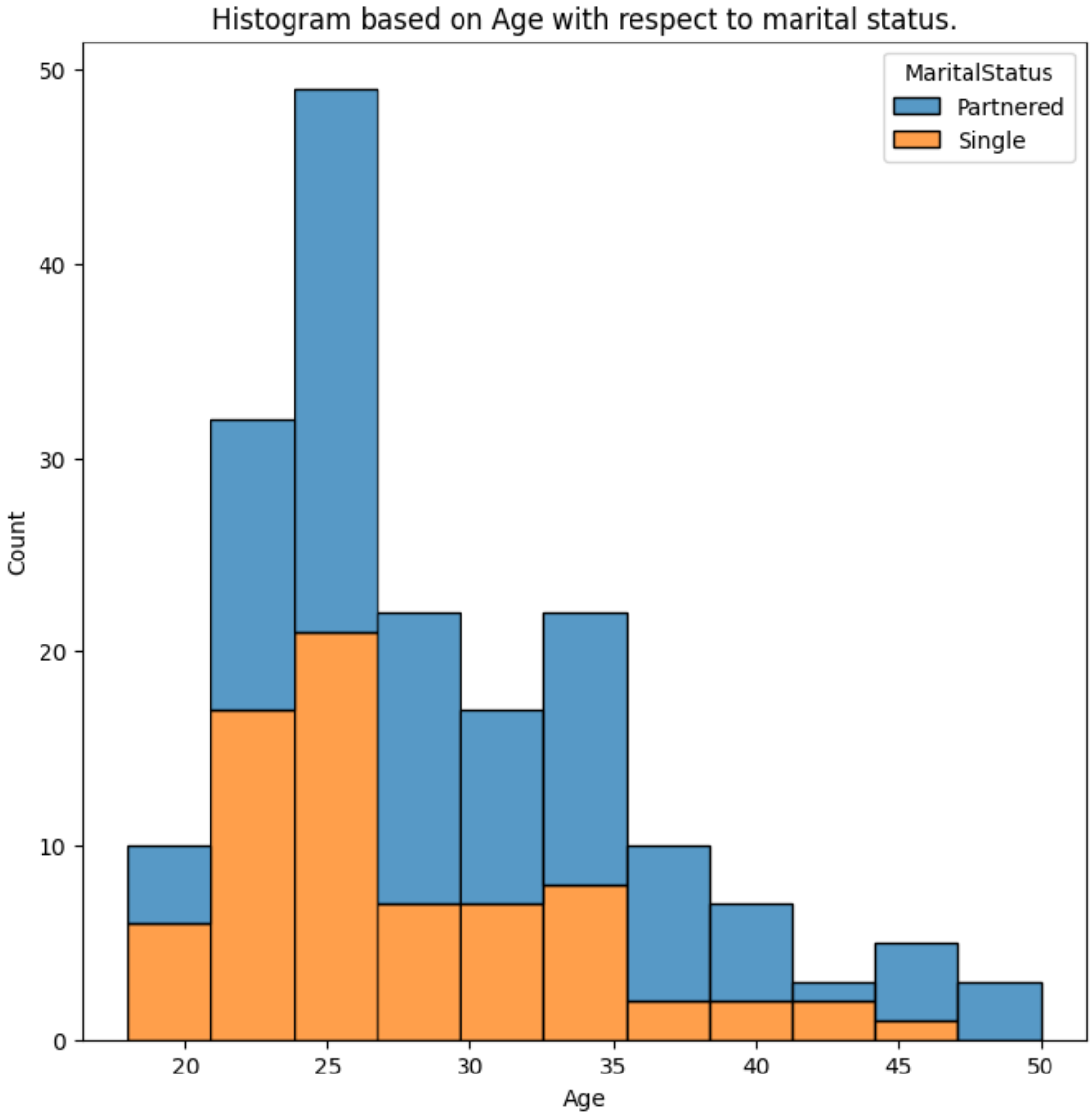


Here the countplot represents the ProductSales based on Marital Status .From this we can say that **Singles are purchasing less when compared to Partnered** in all KP281, KP481 and KP781 . And even Partnered as also purchasing very less KP781 when comapared to other KP481 and KP281.

A **histogram** is a graphical representation of the distribution of a dataset. It is commonly used to visualize the underlying frequency distribution of a **continuous variable**. In a histogram, the data range is divided into intervals (bins), and the height of each bar represents the frequency or

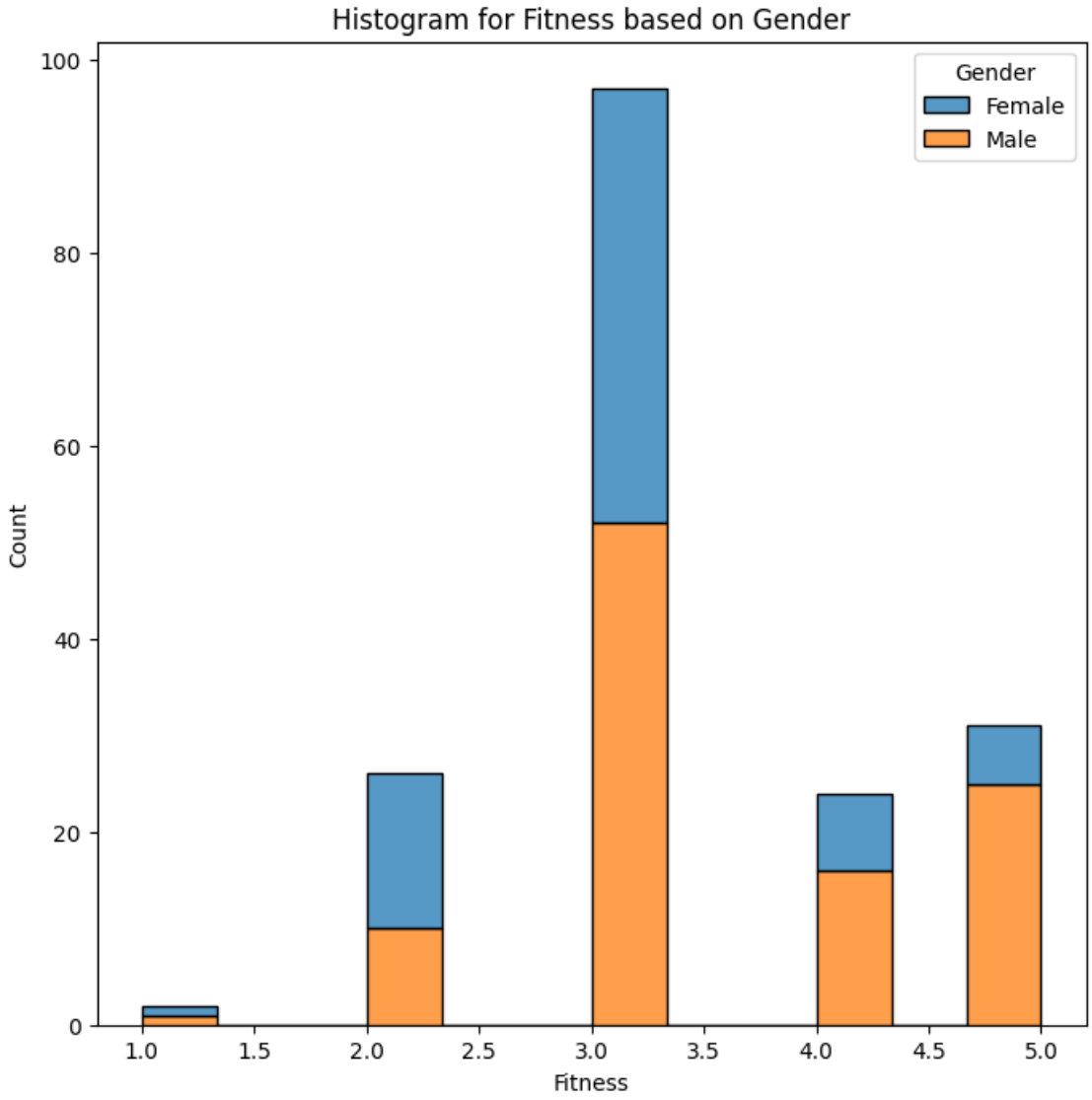
count of data points falling within that interval.

```
[ ]: #Histogram
plt.figure(figsize=(8, 8))
sns.histplot(data=df,x="Age",hue="MaritalStatus",multiple="stack")
plt.title("Histogram based on Age with respect to marital status.")
plt.show()
```



Here we are represting the plot with respect to age, because age is a continuous attribute and based on Marital Status.With the help of stack we have represented the data of both single and partnered in a single graph.

[ ]:
```
plt.figure(figsize=(8, 8))
sns.histplot(data=df,x="Fitness",hue="Gender",multiple="stack")
plt.title("Histogram for Fitness based on Gender")
plt.show()
```



Here we are represting the plot with respect to Fitness, because Fitness is a continuous attribute and based on Gender.With the help of stack we have represented the data of both Male and Female in a single graph.From the above graph we can describe that overall Male are performing better in Fitness when comapared to Women.

[ ]:
```
df.info()
```

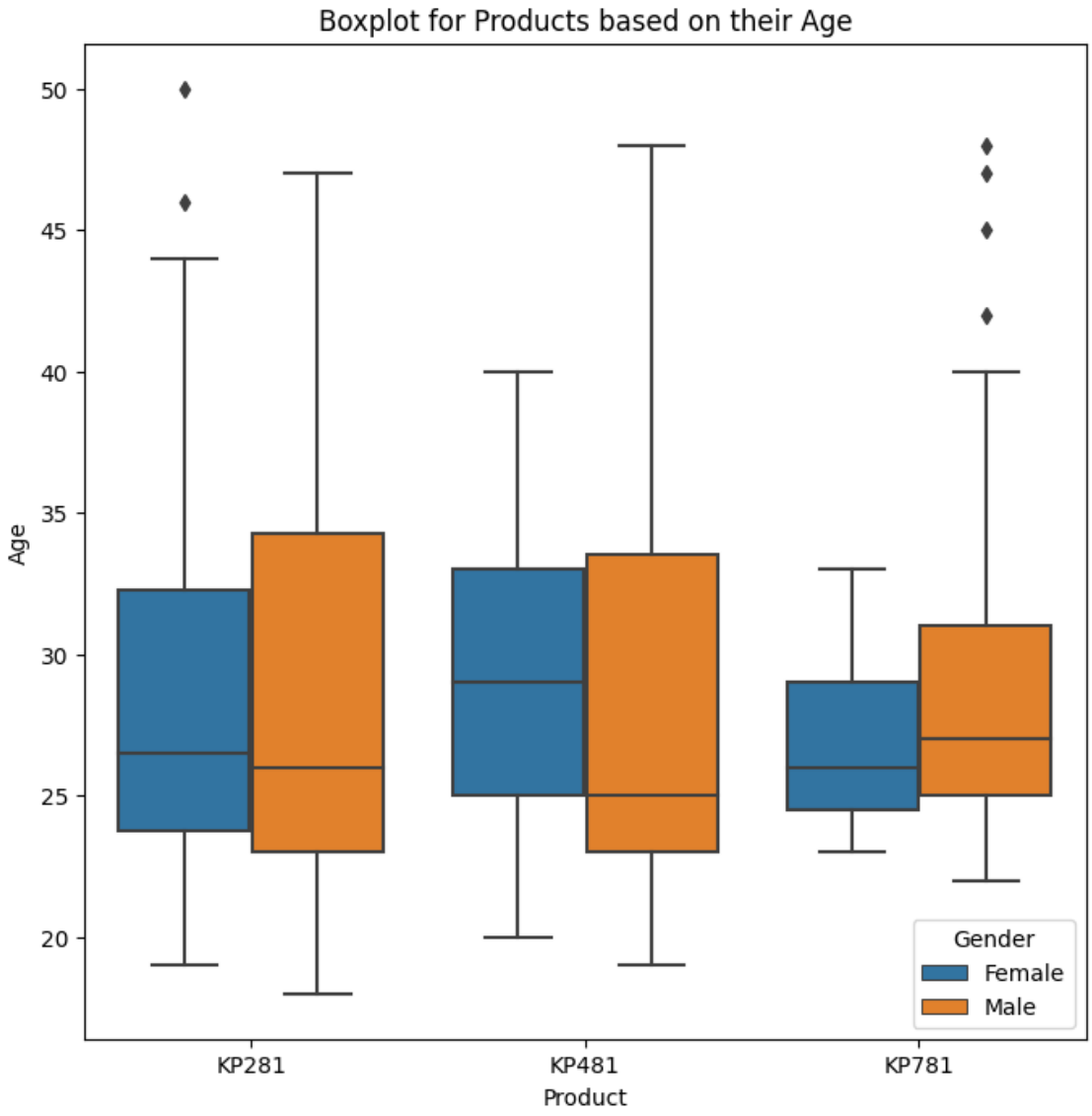
```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 180 entries, 0 to 179
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Product        180 non-null    category
 1   Age            180 non-null    int64
 2   Gender         180 non-null    category
 3   Education      180 non-null    int64
 4   MaritalStatus  180 non-null    category
 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: category(3), int64(6)
memory usage: 9.5 KB
```

[ ]:
```
#For categorical variable(s): Boxplot (10 Points)
```

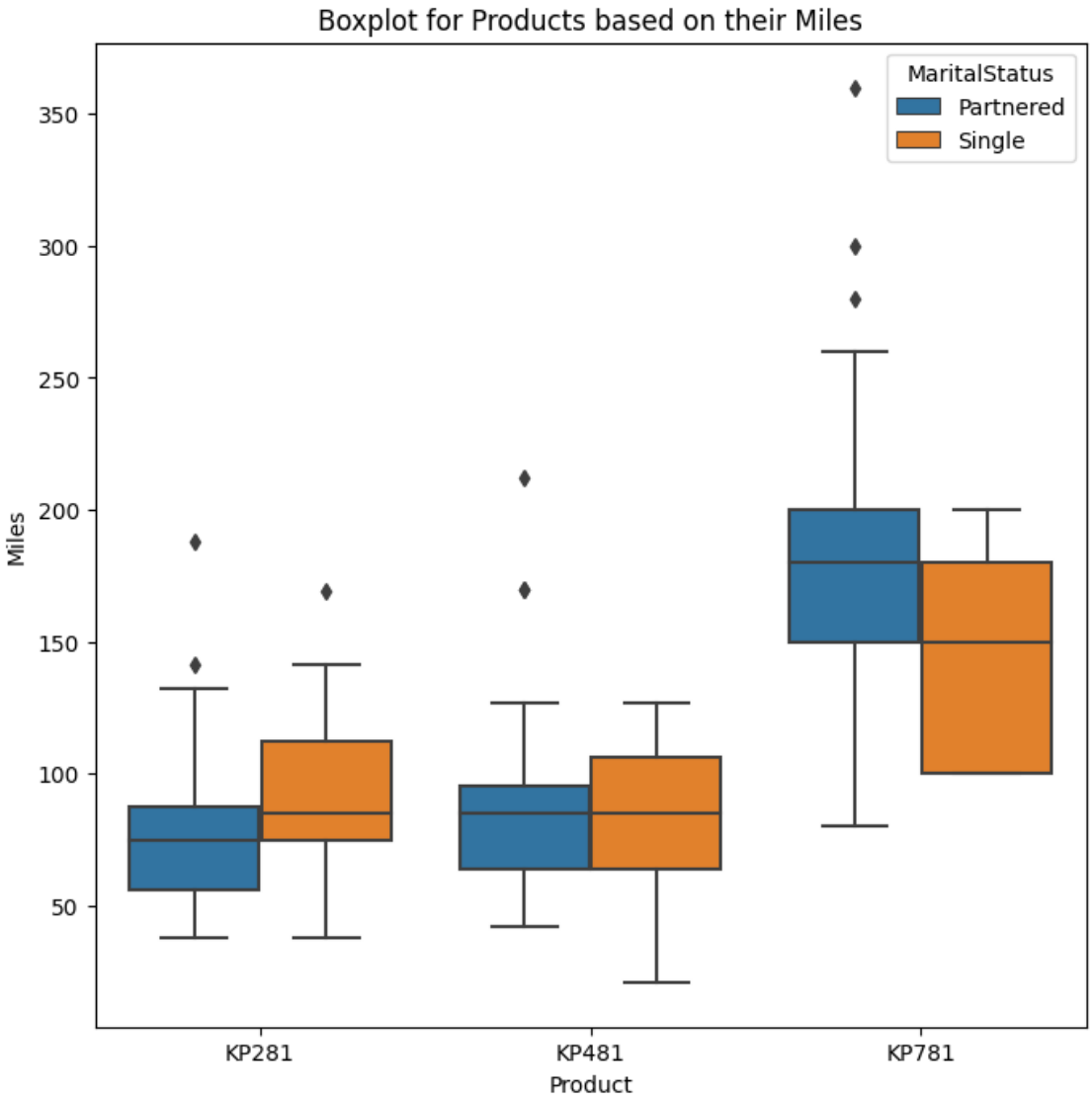
A **Box plot**, also known as a box-and-whisker plot, is a graphical representation that displays the distribution and key statistical properties of a dataset. It provides a concise summary of the central tendency, spread, and presence of outliers in a univariate or grouped dataset.

[ ]:
```
plt.figure(figsize=(8, 8))
sns.boxplot(data=df,x="Product",y="Age",hue="Gender")
plt.title("Boxplot for Products based on their Age")
plt.show()
```



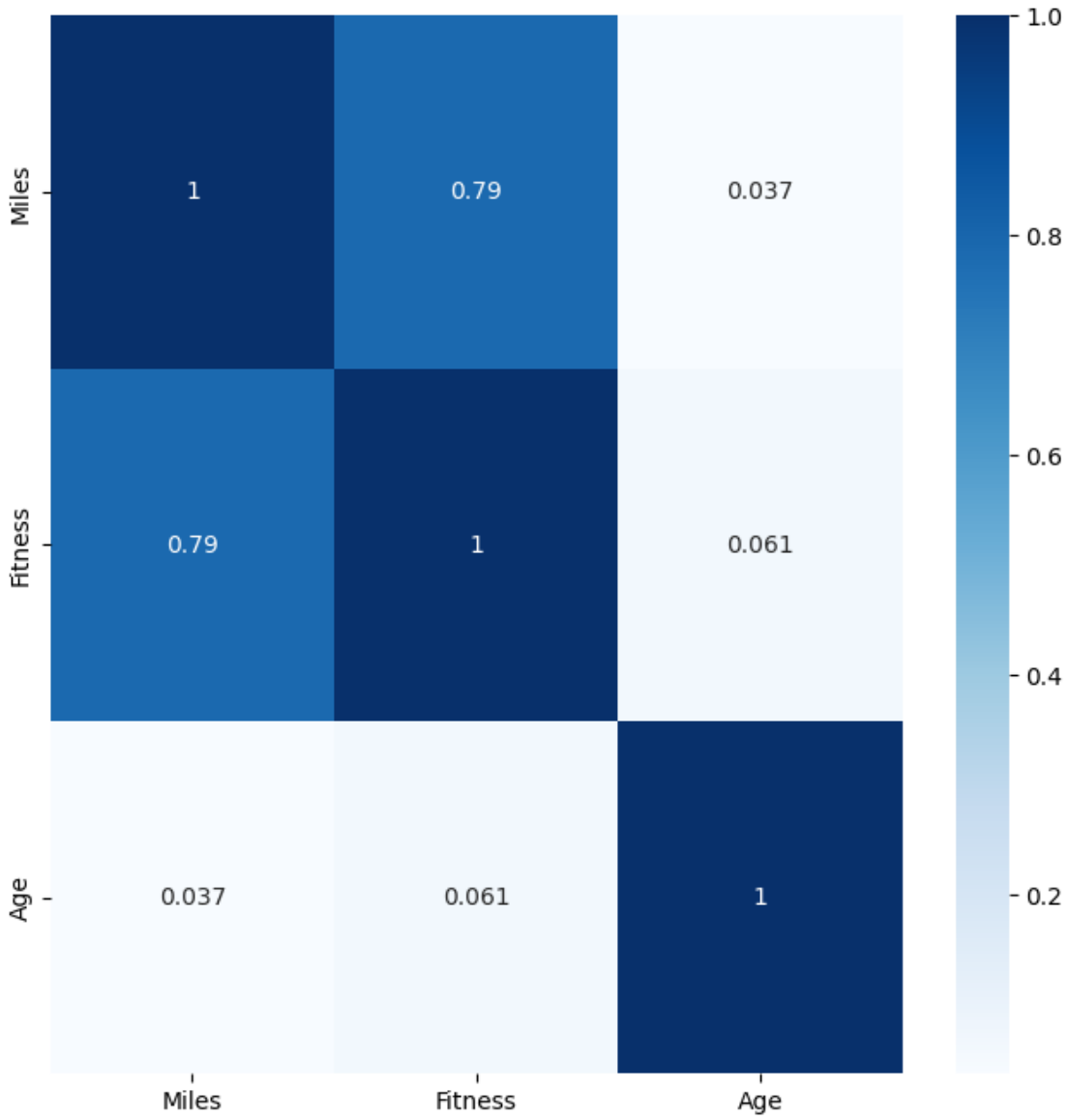
Here the box plot represents the product based on gender. In the above the plot we can see 3 products KP281,KP481 and KP781 for both Male and Female .From this we can say for KP781 for Male we are having more outliers when compared to other products.

[ ]:
```
plt.figure(figsize=(8, 8))
sns.boxplot(data=df,x="Product",y="Miles",hue="MaritalStatus")
plt.title("Boxplot for Products based on their Miles")
plt.show()
```



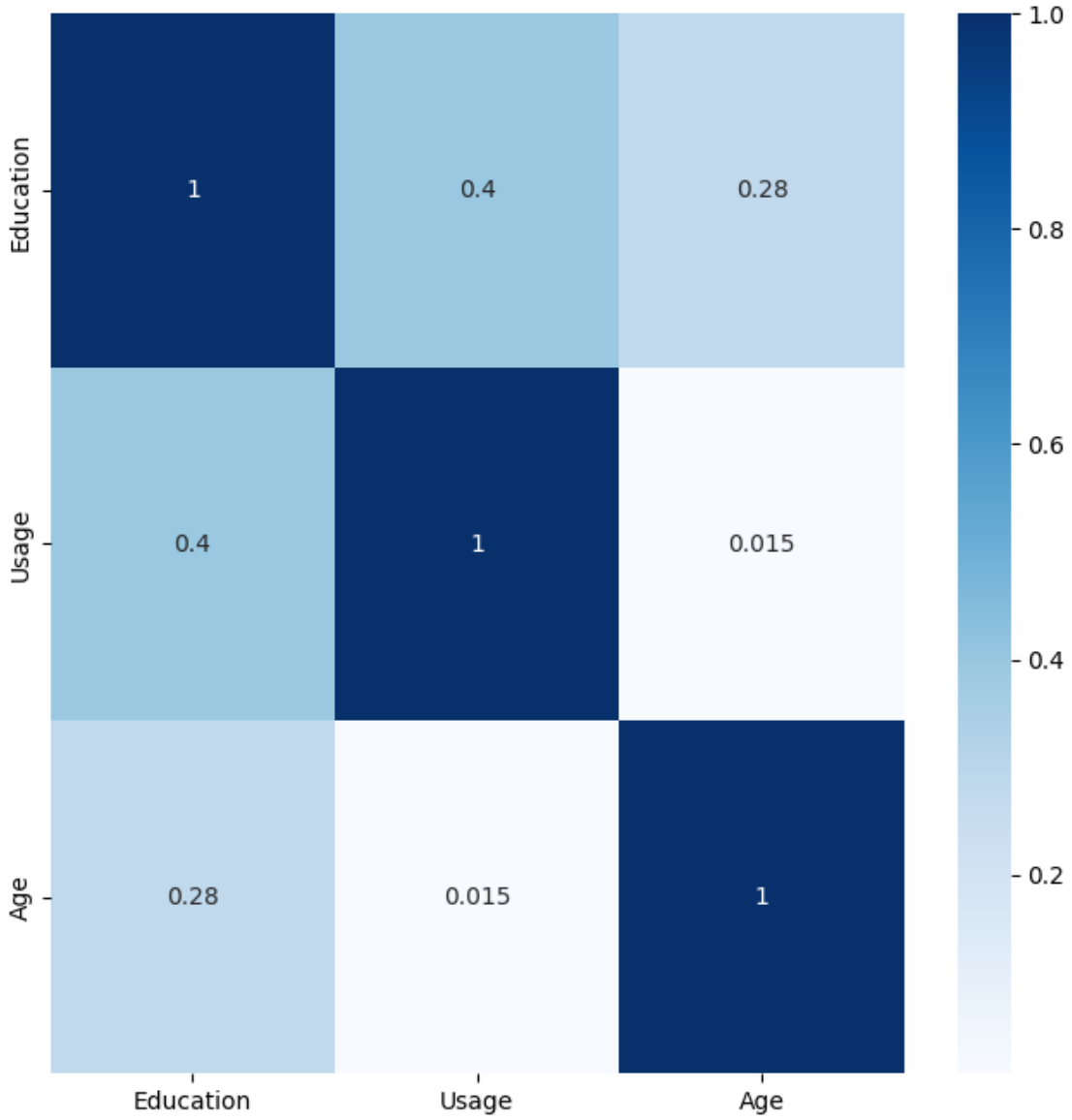
Here the box plot represents the product based on Marital Status. In the above the plot we can see 3 products KP281,KP481 and KP781 for both Single and Partnered .From this we can say for KP781 for Partnered we are having more outliers when compared to other .

[ ]:
```
#For correlation: Heatmaps, Pairplots(10 Points)
plt.figure(figsize=(8, 8))
heat=df[["Miles","Fitness","Age"]]
sns.heatmap(heat.corr(),cmap="Blues",annot=True)
plt.show()
```

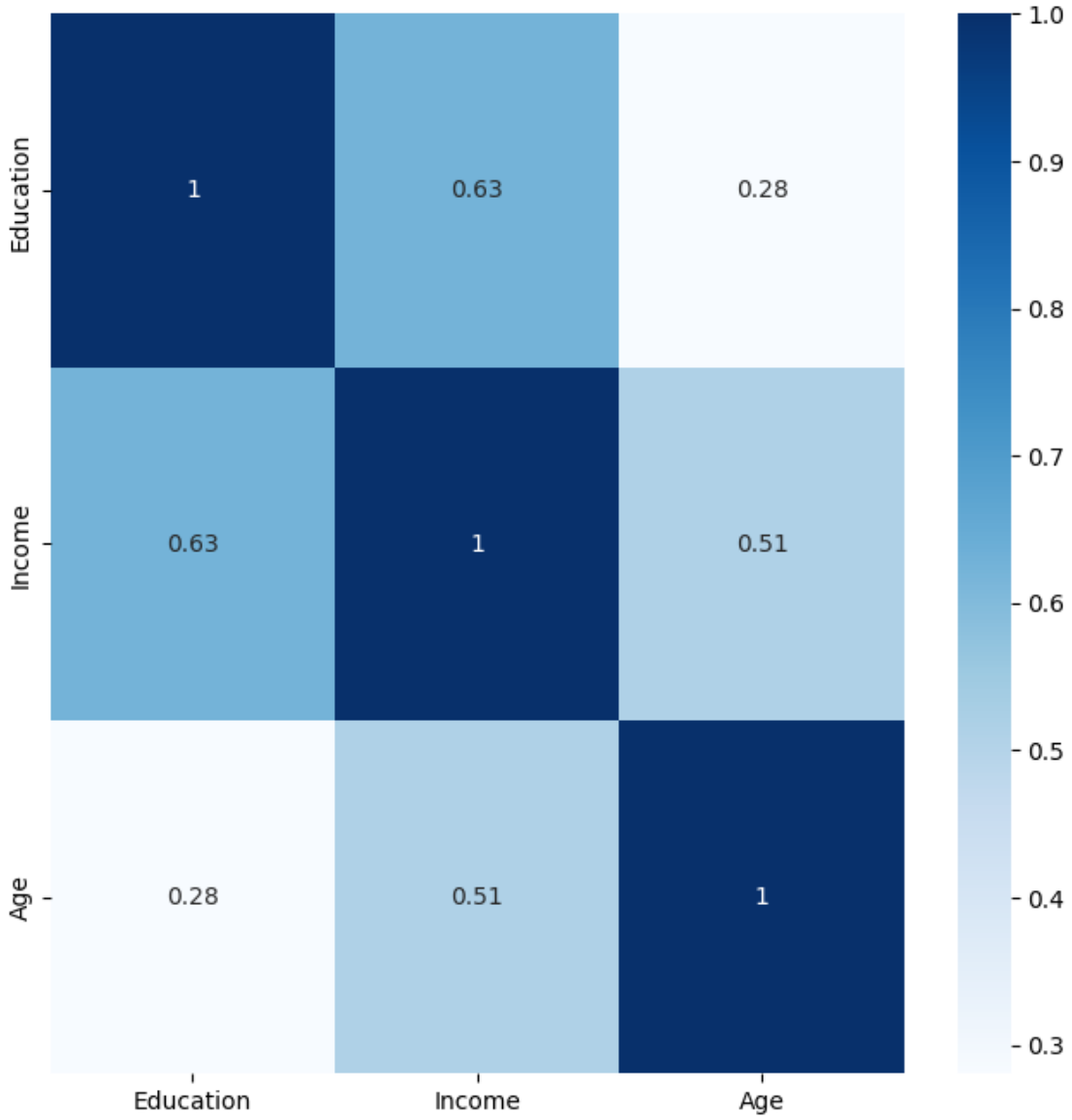


A **heatmap** is a graphical representation of data in a matrix format where values are represented as colors. It is particularly useful for visualizing the magnitude of relationships between two categorical variables or the correlation matrix of numerical variables.

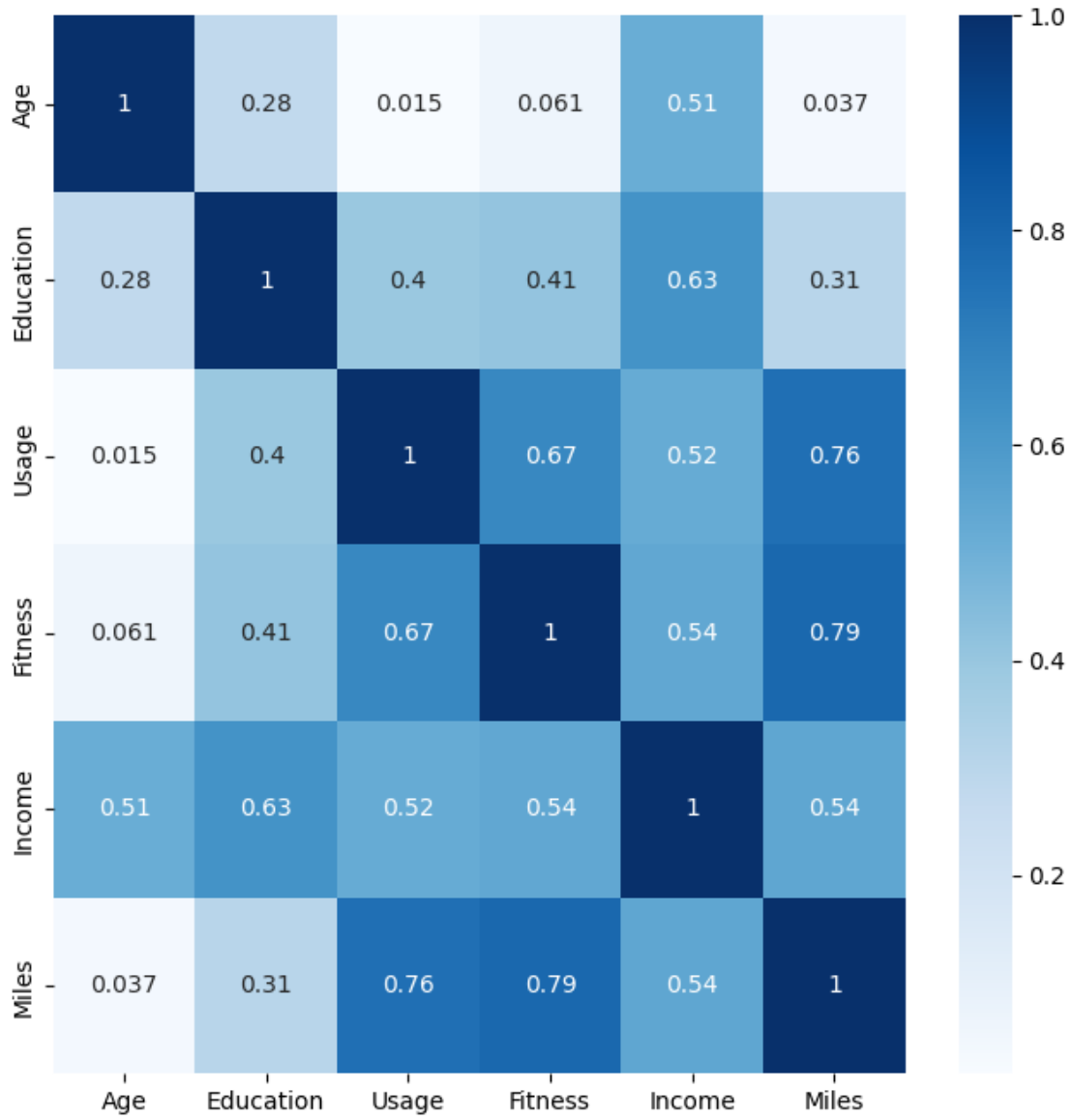
[ ]:
```
plt.figure(figsize=(8, 8))
heat=df[["Education","Usage","Age"]]
sns.heatmap(heat.corr(),cmap="Blues",annot=True)
plt.show()
```



[ ]:
```
plt.figure(figsize=(8, 8))
heat=df[["Education","Income","Age"]]
sns.heatmap(heat.corr(),cmap="Blues",annot=True)
plt.show()
```



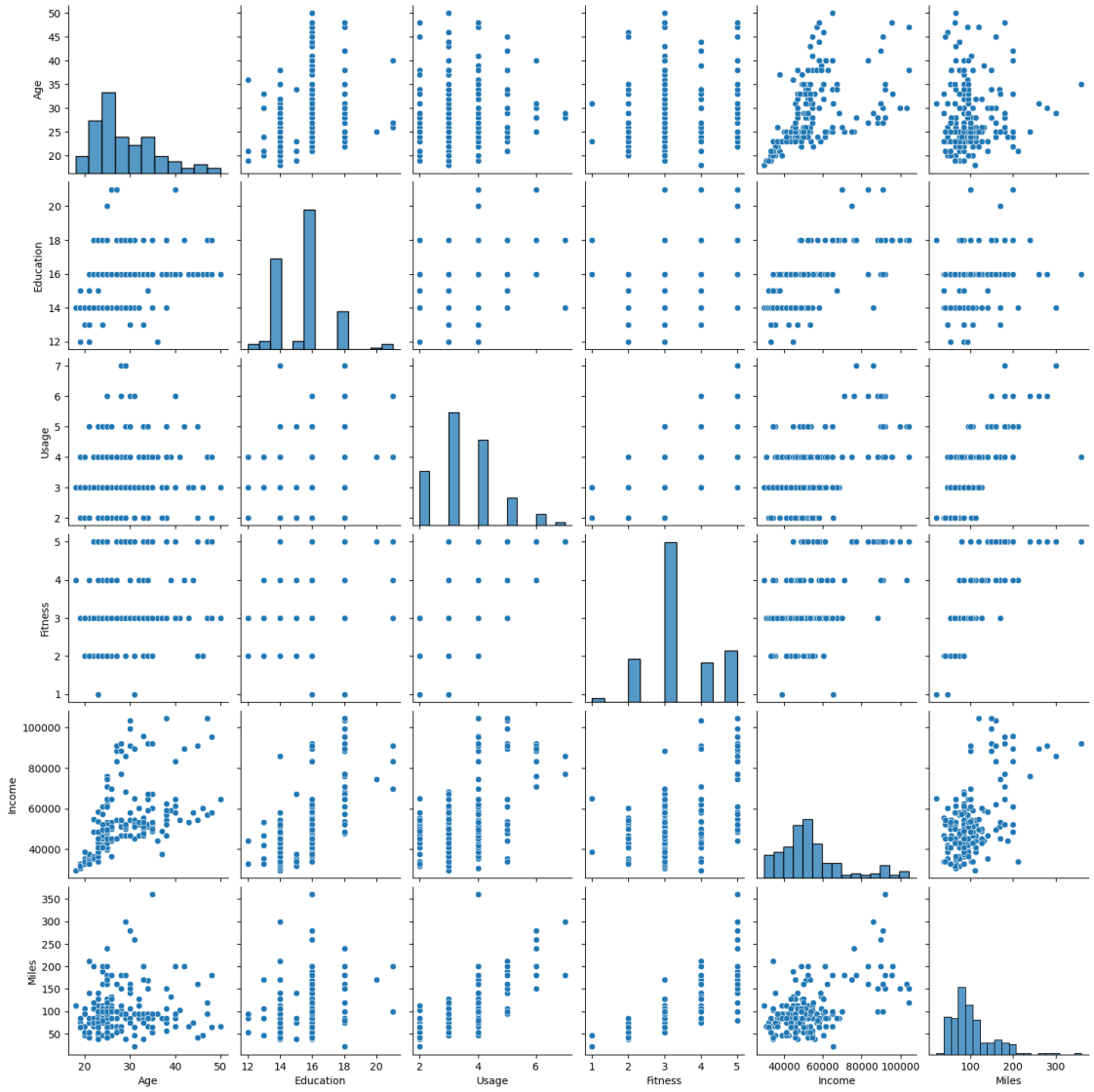
[ ]:
```
plt.figure(figsize=(8, 8))
sns.heatmap(df.corr(),cmap="Blues",annot=True)
plt.show()
```



This is the overall heatmap for the numerical data.By observing this matrix represenatation we will be getting some basic idea regarding the data.

[ ]:
```
#Pairplots
sns.pairplot(data=df)
```

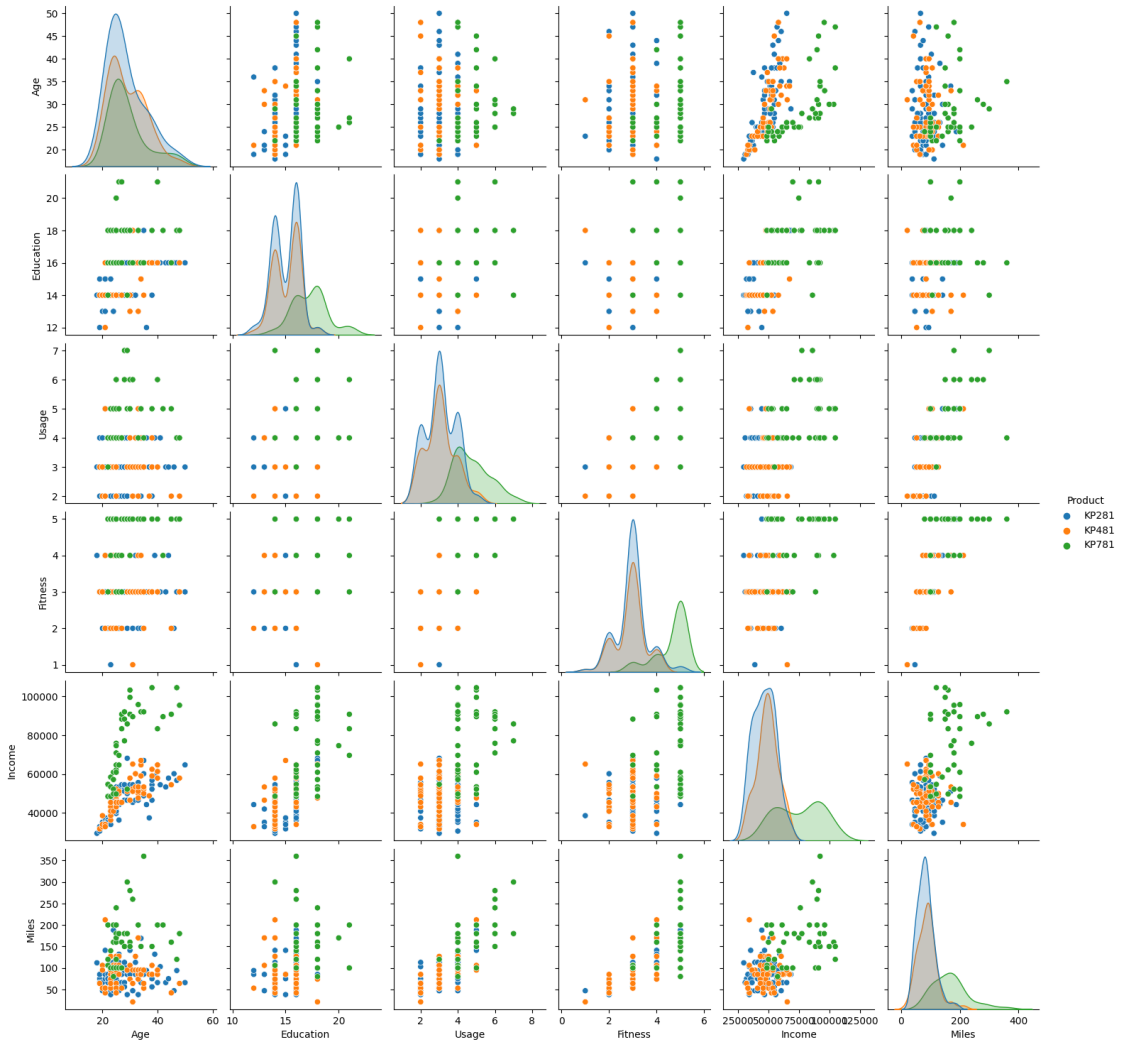
[ ]:
```
<seaborn.axisgrid.PairGrid at 0x7e5a9c3803d0>
```



A **pair plot** (or pairs plot) is a type of scatterplot matrix that provides a quick visual overview of the relationships between pairs of variables in a dataset. It is particularly useful for identifying patterns, correlations, and potential trends in multivariate data.

[ ]:
```
sns.pairplot(data=df,hue="Product")
```

[ ]: <seaborn.axisgrid.PairGrid at 0x7e5a9b1aa0e0>



[ ]:
```
#Missing Value
```

[ ]:
```
missing_values  = df.isnull().sum()
print("Missing Values in the given data:")
missing_values
```

```
Missing Values in the given data:
```

[ ]:
```
Product          0
Age              0
Gender           0
Education        0
MaritalStatus    0
Usage            0
Fitness          0
```

```
Income           0
Miles            0
dtype: int64
```

With the help of **isnull()** we can the null values.By observing this we can say that their are no null values or missing values in the given data.

```
[ ]: #Representing the marginal probability like - what percent of customers have␣
      ↪purchased KP281, KP481, or KP781 in a table (can use pandas.crosstab here)
```

```
[ ]: df[df.Gender=="Male"]["Product"].value_counts(normalize=True)
```

```
[ ]: KP281    0.384615
     KP781    0.317308
     KP481    0.298077
     Name: Product, dtype: float64
```

By observing this data we can say that probablity of men buying the Product threadmills are **0.38 for KP281 , 0.29 for KP481 and 0.31 for KP781**.

By this data we can say that the men were showing much intrest in buying the KP281 abd followed by KP781.

CROSS TAB: One of the most useful tools in Pandas for analyzing tabular data is the crosstab() function.

Cross tabulation (or crosstab) is an important tool for analyzing two categorical variables in a dataset. It provides a tabular summary of the frequency distribution of two variables, allowing us to see the relationship between them and identify any patterns or trends.

normalize: An optional parameter that specifies whether to normalize the frequency table by dividing the values by the grand total. If set to True, the function will normalize the table by dividing each value by the sum of all values

margins: the function will add a row and a column to the table that show the marginal totals.

```
[ ]: pd.crosstab(df.Product,df.Gender,normalize=True,margins=True)
```

```
[ ]: Gender     Female      Male       All
     Product
     KP281    0.222222  0.222222  0.444444
     KP481    0.161111  0.172222  0.333333
     KP781    0.038889  0.183333  0.222222
     All      0.422222  0.577778  1.000000
```

By observing the above table we can easily say that both male and female are equally purchasing KP281(0.22),where as for KP481 men(0.16) are purchasing more and even for KP781 men (0.18)are buying more.

And also the probability of buying KP281 is high and the probability of buying KP781 is low.

The probability of buying Male(0.57) is more than female (0.42)

[ ]:
```
pd.crosstab(df.Product,[df.Gender,df.MaritalStatus],margins=True)
```

[ ]:
```
Gender           Female              Male         All
MaritalStatus Partnered Single Partnered Single
Product
KP281                27     13        21     19   80
KP481                15     14        21     10   60
KP781                 4      3        19     14   40
All                  46     30        61     43  180
```

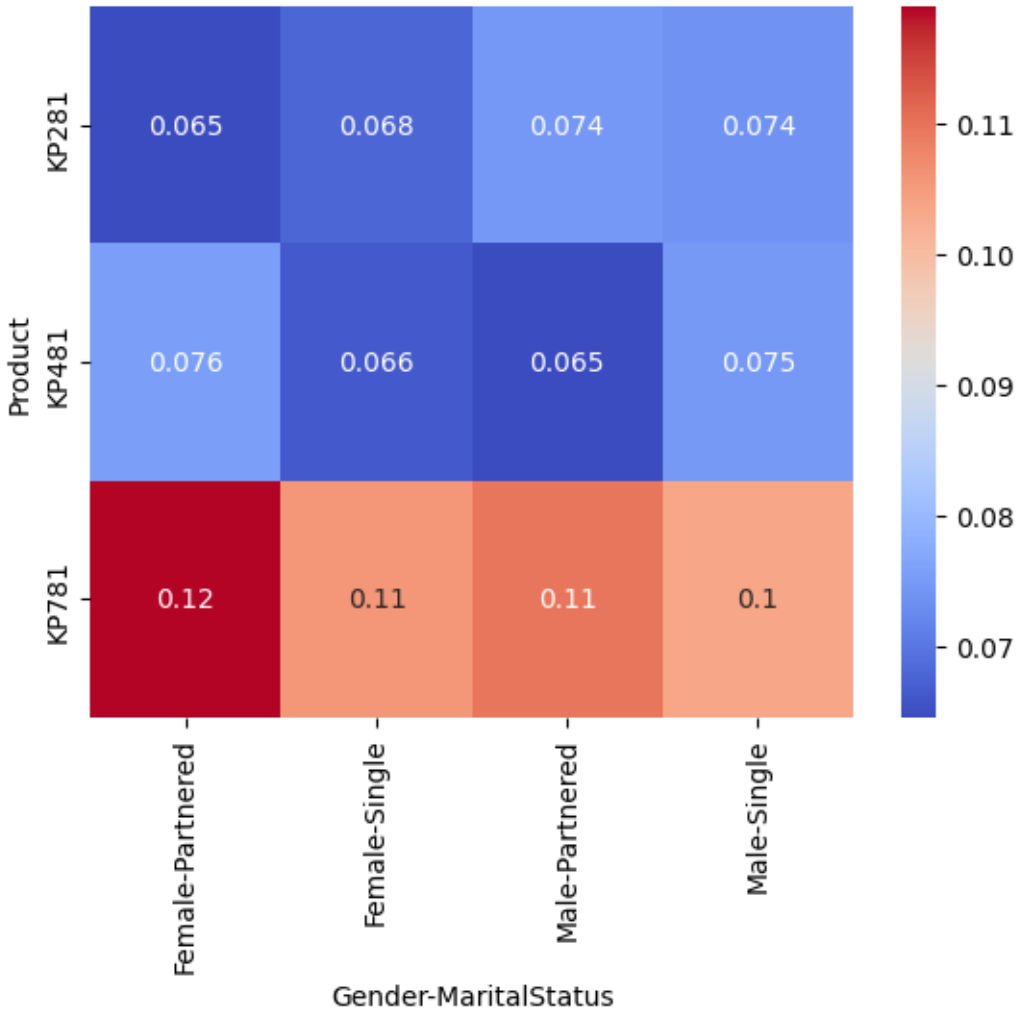
[ ]:
```
pd.crosstab(df.Product,[df.Gender,df.MaritalStatus],normalize=True)
```

[ ]:
```
Gender           Female                Male
MaritalStatus Partnered    Single Partnered    Single
Product
KP281          0.150000  0.072222  0.116667  0.105556
KP481          0.083333  0.077778  0.116667  0.055556
KP781          0.022222  0.016667  0.105556  0.077778
```

[ ]:
```
m=pd.crosstab(df.Product,[df.Gender,df.MaritalStatus],values=df.
↪Usage,aggfunc="mean",normalize=True)
```

[ ]:
```
sns.heatmap(m,cmap="coolwarm",annot=True)
```

[ ]:
```
<Axes: xlabel='Gender-MaritalStatus', ylabel='Product'>
```



The above heatmap represents the probability of buying the product with respect to Gender and Marital Status.

[ ]:
```
pd.crosstab(df.Product,df.Gender,normalize=True,margins=True)
```

[ ]:
```
Gender      Female      Male       All
Product
KP281     0.222222  0.222222  0.444444
KP481     0.161111  0.172222  0.333333
KP781     0.038889  0.183333  0.222222
All       0.422222  0.577778  1.000000
```

[ ]:
```
pd.crosstab(df.Product,df.MaritalStatus,normalize=True,margins=True)
```

[ ]:
MaritalStatus	Partnered	Single	All
Product			
KP281	0.266667	0.177778	0.444444
KP481	0.200000	0.133333	0.333333
KP781	0.127778	0.094444	0.222222
All	0.594444	0.405556	1.000000

[ ]:
```
#With all the above steps you can answer questions like: What is the␣
↪probability of a male customer buying a KP781 treadmill?
```

[ ]:
```
#What is the probability of a male customer buying a KP781 treadmill?
```

[ ]:
```
a=(len(df[(df.Gender=="Male") & (df.Product=="KP781")]))
b=(len(df[(df.Gender=="Male")]))
c=a/b
print(f"P(KP781|M)={c}")
```

```
P(KP781|M)=0.3173076923076923
```

[ ]:
```
#What is the probability of a female customer buying a KP781 treadmill?
```

[ ]:
```
a=(len(df[(df.Gender=="Female") & (df.Product=="KP781")]))
b=(len(df[(df.Gender=="Female")]))
c=a/b
print(f"The probability of female buying KP781 is :{c}")
```

```
The probability of female buying KP781 is :0.09210526315789473
```

[ ]:
```
a=(len(df[(df.MaritalStatus=="Partnered") & (df.Product=="KP781")]))
b=(len(df[(df.MaritalStatus=="Partnered")]))
c=a/b
print(f"The probability of partnered buying KP781 is : {c}")
```

```
The probability of partnered buying KP781 is : 0.21495327102803738
```

[ ]:
```
a=(len(df[(df.MaritalStatus=="Single") & (df.Product=="KP781")]))
b=(len(df[(df.MaritalStatus=="Single")]))
c=a/b
print(f"The probability of singles buying KP781 is :{c}")
```

```
The probability of singles buying KP781 is :0.2328767123287671
```

[ ]:
```
a=(len(df[(df.MaritalStatus=="Single") & ((df.Product=="KP781")|(df.
↪Product=="KP481"))]))
b=(len(df[(df.MaritalStatus=="Single")]))
c=a/b
print(f"The probability of single buying KP781 or KP481 is :{c}")
```

```
The probability of single buying KP781 or KP481 is :0.5616438356164384
```

```
[ ]: #Given that a customer is male, what is the probability that they have a␣
      ↪fitness level of 4 or 5
```

```
[ ]: a=(len(df[((df.Fitness==4) |(df.Fitness==5))&(df.Gender=="Male")]))
     b=(len(df[(df.Gender=="Male")]))
     c=a/b
     print(f"The probability of male for the fitness level 4 or 5 is :{c}")
```

```
The probability of male for the fitness level 4 or 5 is :0.3942307692307692
```

```
[ ]: #If a customer is under 30 years old, what is the probability that they␣
      ↪purchase product KP481
```

```
[ ]: a=(len(df[(df.Product=="KP481")&(df.Age<=30)]))
     b=(len(df[(df.Age<=30)]))
     c=a/b
     print(f"The probability of purchasing product KP481 age under 30 years are :
      ↪{c}")
```

```
The probability of purchasing product KP481 age under 30 years are
:0.2916666666666667
```

```
[ ]: #Given that a customer is over 40 years old, what is the probability that they␣
      ↪have an income over $50,000?
```

```
[ ]: a=(len(df[(df.Income>=50000)&(df.Age>=40)]))
     b=(len(df[(df.Age>=40)]))
     c=a/b
     print(f"The probability of having income more than $50000 more than age of 40␣
      ↪years are :{c}")
```

```
The probability of having income more than $50000 more than age of 40 years are
:1.0
```

```
[ ]: #What is the probability that a single customer plans to use the treadmill more␣
      ↪than 3 times a week?
```

```
[ ]: a=(len(df[(df.MaritalStatus=="Single")&(df.Usage>3)]))
     b=(len(df[(df.Usage>3)]))
     c=a/b
     print(f"The probability of Single and Usage of Threadmill is  :{c}")
```

```
The probability of Single and Usage of Threadmill is  :0.4230769230769231
```

```
[ ]: #If a customer has less than 12 years of education, what is the probability␣
      ↪that they purchase product KP281?
```

[ ]:
```
a=(len(df[(df.Product=="KP281")&(df.Education<=12)]))
b=(len(df[(df.Education<=12)]))
c=a/b
print(f"The probability of purchasing of KP281 of education less than 12   :
 ↪{c}")
```

```
The probability of purchasing of KP281 of education less than 12
:0.6666666666666666
```

[ ]:
```
#Given that a customer has an income between $50,000 and $70,000, what is the␣
 ↪probability that they are in excellent shape (fitness level 5)
```

[ ]:
```
a=(len(df[((df.Income>=50000)&(df.Income<=70000))&(df.Fitness==5)]))
b=(len(df[((df.Income>=50000)&(df.Income<=70000))]))
c=a/b
print(f"The probability of Excellent shape of customer who is earning between␣
 ↪50000 and 700000 :{c}")
```

```
The probability of Excellent shape of customer who is earning between 50000 and
700000 :0.10810810810810811
```

[ ]:
```
#If a customer rates their fitness level as 1, what is the probability that␣
 ↪they are single?
```

[ ]:
```
a=(len(df[(df.MaritalStatus=="Single")&(df.Fitness==1)]))
b=(len(df[(df.Fitness==1)]))
c=a/b
print(f"The probability of Single for fitnesss level 1 is :{c}")
```

```
The probability of Single for fitnesss level 1 is :0.5
```

[ ]:
```
#Given that a customer is in excellent shape (fitness level 5), what is the␣
 ↪probability that they plan to walk/run more than 30 miles a week?
```

[ ]:
```
a=(len(df[(df.Fitness==5)&(df.Miles>=(30*7))]))
b=(len(df[(df.Fitness==5)]))
c=a/b
print(f"The probability of fitnesss level 5 for 30 miles in a week :{c}")
```

```
The probability of fitnesss level 5 for 30 miles in a week :0.16129032258064516
```

Recommendations:

*From the overall data till now i have observed that the people are purchasing KP281 more than KP481 and KP781.

**The probability of female buying the product is KP781 is very low(0.03) when compared to female buying KP281(0.22) and KP481(0.16).

*The people with age less than 30 years are showing 30% intreset in buying the product.

*The single's usage of threadmill is 42% that means partners are usage of threadmill is mill is high ie is 48% the aerofit company need to concentarte on single to increase the purchases.

*The education level below 12 are showing much intreset in buying KP281 ie 66%

*Fitness of a person is good with usage of the product.So the aerofit threadmill can use to this point and explain how the fitness can be increased by using these products and can increase the sales.

*As the education level is high their is income is also good , so that peopele who are earning good income the aerofit can motivate them by creating awareness regarding fitness and how they can increase their fitness by usage of their product .

*The comapny nees to mainly concentrate on single sbuying the product Because singles are buying very less product.By creating awarness among the single's the comapnies sales will get increased.

*By creating awareness among singles and female reagrding the product so that the sales will get increased.

[ ]: