Aerofit Business Case Study (Collab Link: Aerofit)

This case study is about performing Descriptive Analytics for each treadmill product and to find the conditional and marginal probability of those products

Dataset:

```
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

Insights: We have 3 months data of customers who purchased treadmills from Aerofit. We have their details in this.

Shape of the Dataset:

```
df.shape
```

```
(180, 9)
```

Insights: There are 180 rows and 9 columns in our DataSet.

Data Types of the columns in the Dataset:

Data Types

```
df.dtypes
```

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

Data Cleaning:

Checking for Null or missing values:

```
df.isnull(  ).sum()

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

Insights: There are no missing values in the dataset and there are no merged values in a single cell. We got a clean data with us.

Checking for duplicated rows:

```
df.duplicated().sum()

0
```

Insights: There are no duplicated rows in our Dataset.

Exploratory Data Analytics

Product Column:

Unique values:

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

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

Insights: There are 3 types of Treadmills in the Dataset.

Value Counts:

```
[22] df['Product'].value_counts()

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

Insights: From the output we can see the value counts of the treadmills. These values define the sale count of each product. We can clearly see from the output that KP281 product has more sales compared to others.

Gender Column:

Unique values:

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

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

Value counts:

```
[26] df['Gender'].value_counts()

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

Marital Status Column:

Unique Values:

```
[28] df['MaritalStatus'].unique()

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

Value counts:

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

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

Descriptive statistics of our Dataset:

```
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

Detecting and Handling Outliers:

Age Column:

```
[ ] Age_Metrics = df['Age'].describe()
    print(Age_Metrics)
```

```
count    180.000000
mean      28.788889
std        6.943498
min       18.000000
25%       24.000000
50%       26.000000
75%       33.000000
max       50.000000
Name: Age, dtype: float64
```

IQR:

To get the Inter Quartile range we need to substract 75% percentile - 50% percentile

```
IQR = Age_Metrics['75%'] - Age_Metrics['25%']
print('Inter Quartile range is:',IQR)
```

```
Inter Quartile range is: 9.0
```

Finding Min and Max values to detect outliers:

We need to calculate the max and min to find out the outliers

```
max_value = Age_Metrics['75%'] + 1.5*IQR
min_value = Age_Metrics['25%'] - 1.5*IQR
print('Max value is:',max_value)
print('Min value is:',min_value)
```

```
Max value is: 46.5
Min value is: 10.5
```

Outliers percentage in Age column:

```
Outliers_count = ((df['Age'] < min_value) | (df['Age'] > max_value)).sum()
Outliers_percentage = round((Outliers_count/len(df['Age']))*100,1)
print('Outliers count is:',Outliers_count)
print('Outliers percentage is:',Outliers_percentage)
```

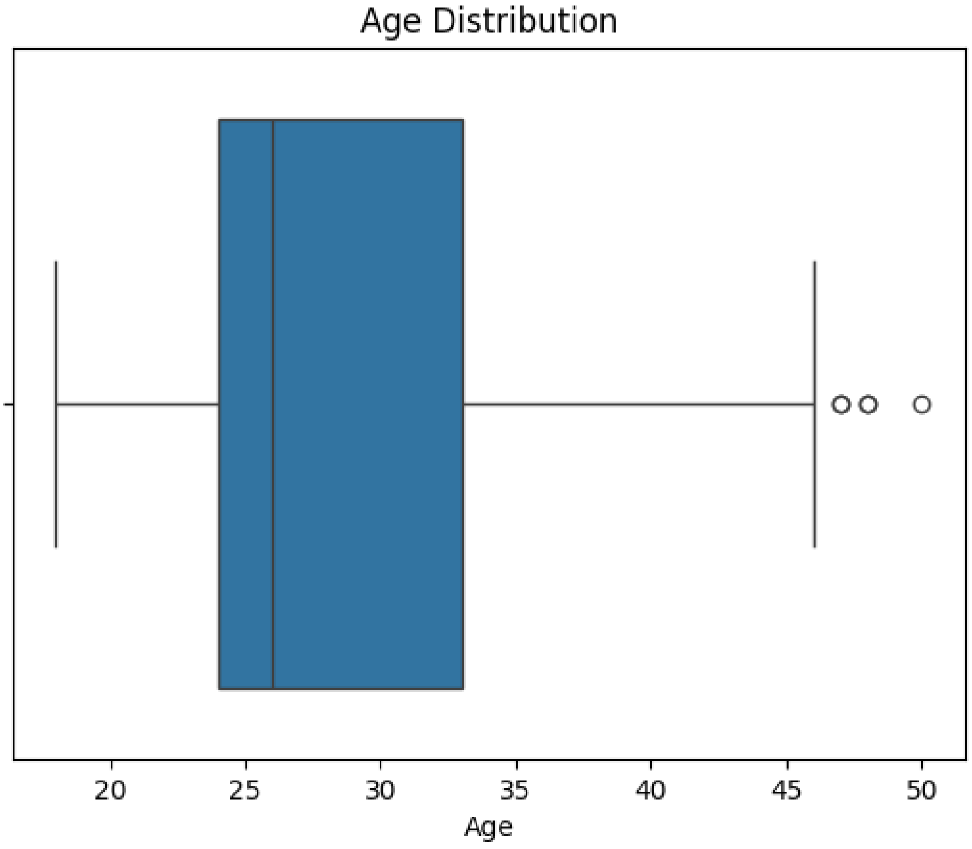
```
Outliers count is: 5
Outliers percentage is: 2.8
```

Visual Representation of the Age column:

Code:

```
sns.boxplot(x = df['Age'])
plt.title('Age Distribution')
plt.show()
```

Graph:



Insights:

From the grap and above calculations we can clearly see that there's 2.8% of outliers in the Age data.

1. Maximum age of our users is 46.
2. Minimum age of our users is 10.
3. Average age of our users is 28.

Handling the outliers using clip () method:

Getting 5 percentile and 95 percentile using numpy's percentile() function

```
min_clip, max_clip = np.percentile(df['Age'],[5,95])
print('5 percentile is:',min_clip)
print('95 percentile is:',max_clip)
```

```
5 percentile is: 20.0
95 percentile is: 43.04999999999998
```

Dataset after using clip () method:

Clipping the data for below 5 percentile and above 95 percentile.

```
[ ] df['Age'] = df['Age'].clip(lower = min_clip, upper = max_clip)
    df[(df['Age'] == min_clip) | (df['Age'] == max_clip)].head(10)
```

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
0	KP281	20.00	Male	14	Single	3	4	29562	112
1	KP281	20.00	Male	15	Single	2	3	31836	75
2	KP281	20.00	Female	14	Partnered	4	3	30699	66
3	KP281	20.00	Male	12	Single	3	3	32973	85
4	KP281	20.00	Male	13	Partnered	4	2	35247	47
5	KP281	20.00	Female	14	Partnered	3	3	32973	66
76	KP281	43.05	Female	16	Single	3	4	57987	75
77	KP281	43.05	Female	16	Partnered	3	2	60261	47
78	KP281	43.05	Male	16	Partnered	4	3	56850	94
79	KP281	43.05	Female	16	Partnered	3	3	64809	66

Income Column:

```
[48] Income_Metrics = df['Income'].describe()
     print(Income_Metrics)
```

```
count       180.000000
mean      53719.577778
std       16506.684226
min       29562.000000
25%       44058.750000
50%       50596.500000
75%       58668.000000
max      104581.000000
Name: Income, dtype: float64
```

IQR, MAX and MIN:

```
[49] Income_IQR = Income_Metrics['75%'] - Income_Metrics['25%']
     Income_Max_value = Income_Metrics['75%'] + 1.5*Income_IQR
     Income_Min_value = Income_Metrics['25%'] - 1.5*Income_IQR
     print('Income Inter Quartile range is:',Income_IQR)
     print('Income Max value is:',Income_Max_value)
     print('Income Min value is:',Income_Min_value)
```

```
Income Inter Quartile range is: 14609.25
Income Max value is: 80581.875
Income Min value is: 22144.875
```

Outliers count and percentage in Income Column:

```
Income_Outliers_count = ((df['Income'] < Income_Min_value) | (df['Income'] > Income_Max_value)).sum()
Income_Outliers_percentage = round((Income_Outliers_count/len(df['Income']))*100,1)
print('Income Outliers count is:',Income_Outliers_count)
print('Income Outliers percentage is:',Income_Outliers_percentage)
```

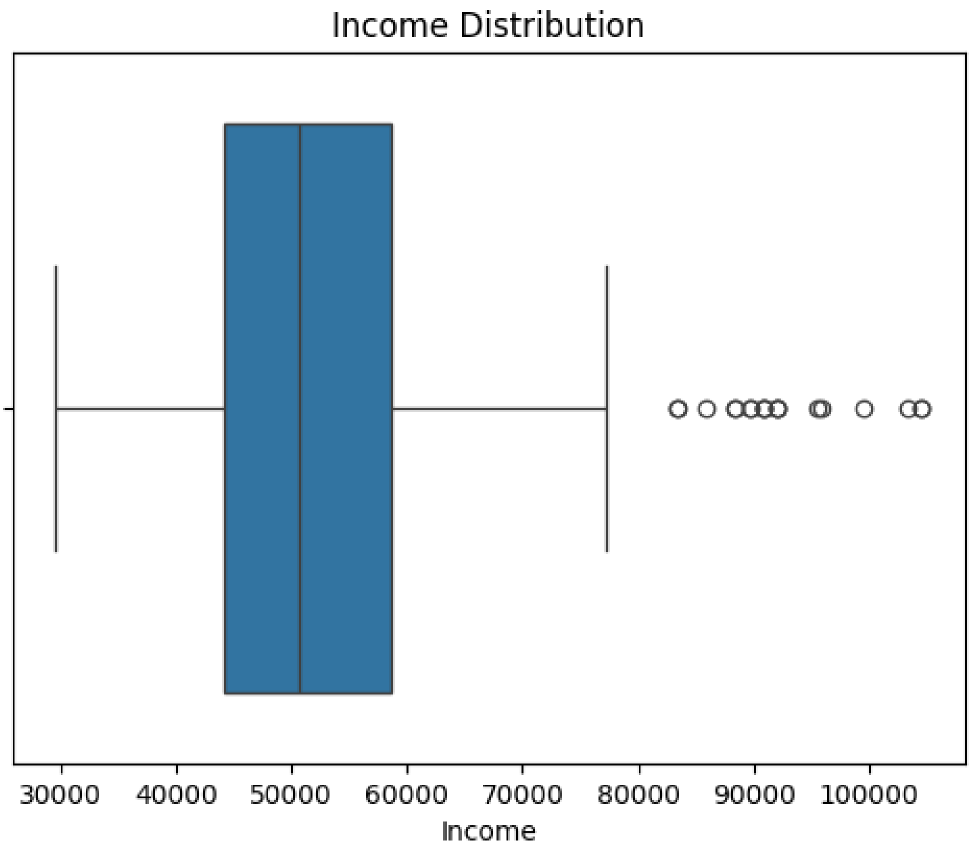
```
Income Outliers count is: 19
Income Outliers percentage is: 10.6
```

Visual representation of Income column:

Code:

```
[ ] sns.boxplot(x = df['Income'])
    plt.title('Income Distribution')
    plt.show()
```

Graph:



Handling Outliers:

```
[53] Income_min_clip, Income_max_clip = np.percentile(df['Income'],[5,95])
     print('5 percentile is:',Income_min_clip)
     print('95 percentile is:',Income_max_clip)
```

```
5 percentile is: 34053.15
95 percentile is: 90948.24999999999
```

Dataset after using clip () method:

```
df['Income'] = df['Income'].clip(lower = Income_min_clip, upper = Income_max_clip)
df[(df['Income'] == Income_min_clip) | (df['Income'] == Income_max_clip)].head()
```

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
0	KP281	20.0	Male	14	Single	3.0	4	34053.15	112
1	KP281	20.0	Male	15	Single	2.0	3	34053.15	75
2	KP281	20.0	Female	14	Partnered	4.0	3	34053.15	66
3	KP281	20.0	Male	14	Single	3.0	3	34053.15	85
5	KP281	20.0	Female	14	Partnered	3.0	3	34053.15	66

Mile’s column:

```
Miles_Metrics = df['Miles'].describe()
print(Miles_Metrics)
```

```
count    180.000000
mean     103.194444
std       51.863605
min       21.000000
25%       66.000000
50%       94.000000
75%      114.750000
max      360.000000
Name: Miles, dtype: float64
```

IQR, MAX, MIN:

```
Miles_IQR = Miles_Metrics['75%'] - Miles_Metrics['25%']
Miles_Max_value = Miles_Metrics['75%'] + 1.5*Miles_IQR
Miles_Min_value = Miles_Metrics['25%'] - 1.5*Miles_IQR
print('Miles Inter Quartile range is:',Miles_IQR)
print('Miles Max value is:',Miles_Max_value)
print('Miles Min value is:',Miles_Min_value)
```

```
Miles Inter Quartile range is: 48.75
Miles Max value is: 187.875
Miles Min value is: -7.125
```

Outliers count and percentage:

```
Miles_Outliers_count = ((df['Miles'] < Miles_Min_value) | (df['Miles'] > Miles_Max_value)).sum()
Miles_Outliers_percentage = round((Miles_Outliers_count/len(df['Miles']))*100,1)
print('Miles Outliers count is:',Miles_Outliers_count)
print('Miles Outliers percentage is:',Miles_Outliers_percentage)
```

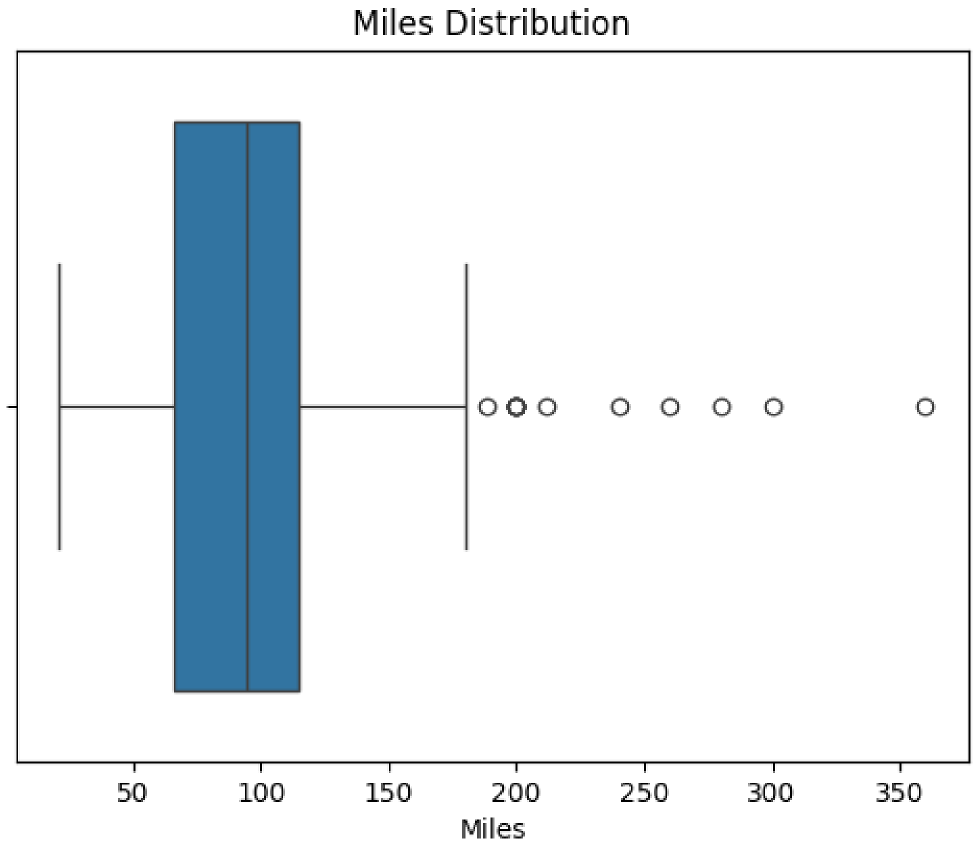
```
Miles Outliers count is: 13
Miles Outliers percentage is: 7.2
```

Visual representation of Miles column:

Code:

```
sns.boxplot(x = df['Miles'])
plt.title('Miles Distribution')
plt.show()
```

Graph:



Handling outliers:

```python
Miles_min_clip, Miles_max_clip = np.percentile(df['Miles'],[5,95])
print('5 percentile is:',Miles_min_clip)
print('95 percentile is:',Miles_max_clip)
```

```
5 percentile is: 47.0
95 percentile is: 200.0
```

Dataset after using clip () method:

```python
df['Miles'] = df['Miles'].clip(lower = Miles_min_clip, upper = Miles_max_clip)
df[(df['Miles'] == Miles_min_clip) | (df['Miles'] == Miles_max_clip)].head()
```

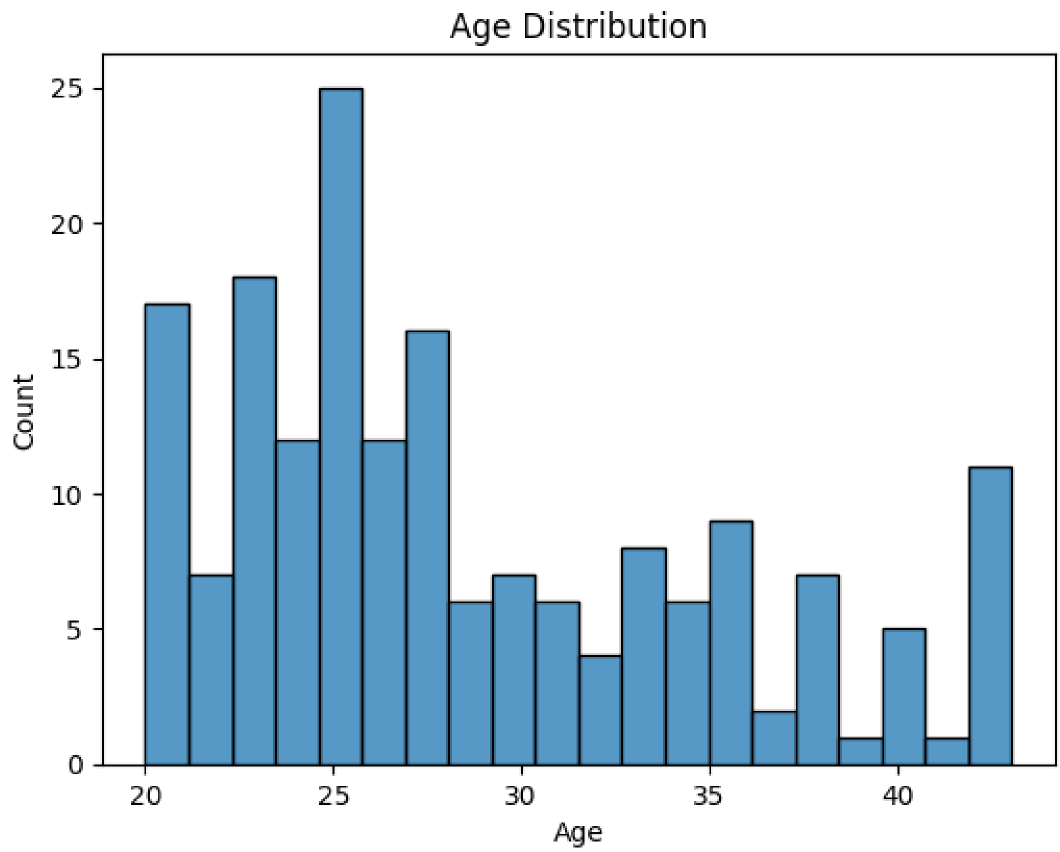
	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
4	KP281	20.0	Male	14	Partnered	4.0	2	35247.0	47
14	KP281	23.0	Male	16	Partnered	3.0	2	38658.0	47
19	KP281	23.0	Female	15	Partnered	2.0	2	34110.0	47
25	KP281	24.0	Male	14	Partnered	3.0	2	42069.0	47
29	KP281	25.0	Female	14	Partnered	2.0	2	53439.0	47

Age Distribution:

Code:

```
sns.histplot(x = df['Age'], bins = 20)
plt.title('Age Distribution')
plt.show()
```

Graph:



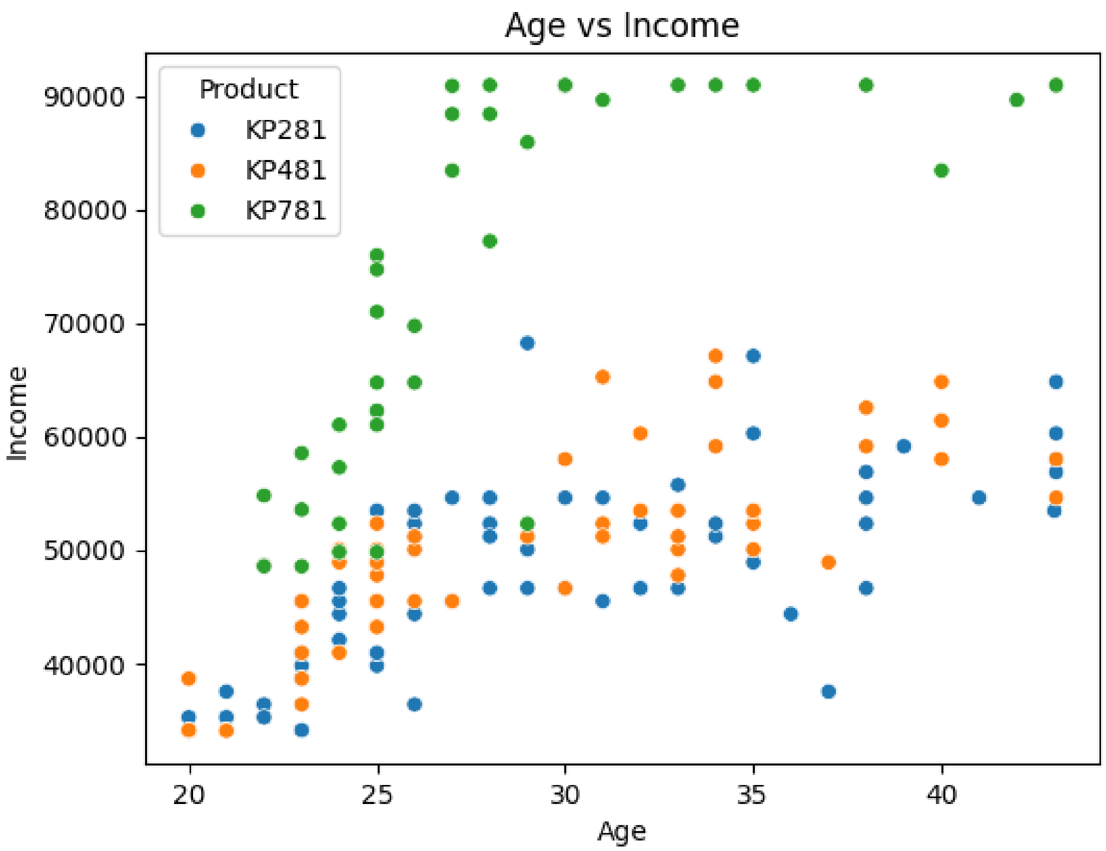
Insights: From the histogram we can clearly see that most our customers are around age 25.

Product sales relation with age and Income:

Code:

```
sns.scatterplot(x = df['Age'], y = df['Income'], hue = df['Product'])
plt.title('Age vs Income')
plt.show()
```

Graph:



Insights: From scatterplot we can see that people with high age are earning more and buying the most advanced product. Similarly, people with less age are earning less and buying the product with basic features.

Recommendation: From the histogram and scatter plot we can see that most of our customers are around age 25 and customers around that age are earning less compared to others. We can improve our sales by targeting these customers.

1. We can diversify our products by adding new features.

2. We can try to add advance features to our products within our target customers budget range to improve our sales and for better customer experience.

Product-wise Sales:

Code:

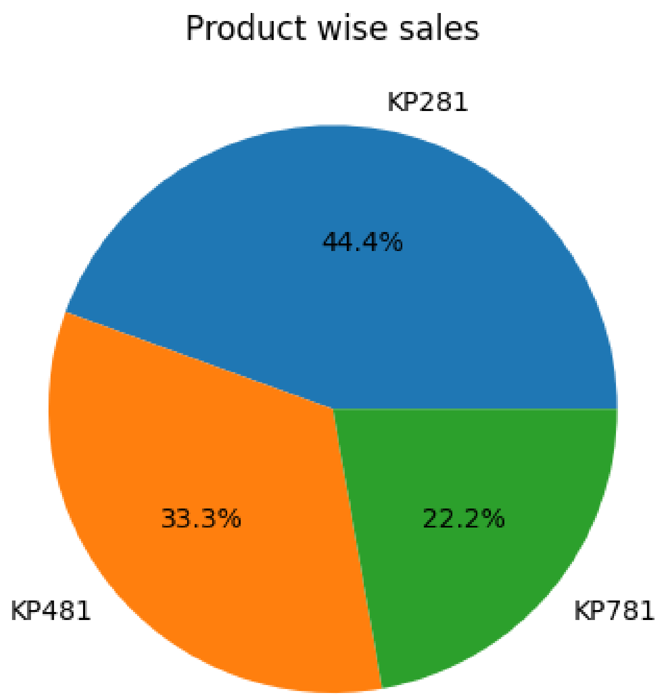
```
df['Product'].value_counts()

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

Graphical Representation:

Code:

```
[ ] plt.pie(x  = df['Product'].value_counts(), labels = df['Product'].unique(), autopct = '%.1f%%')
    plt.title('Product wise sales')
    plt.show()
```

Graph:



Insights: From the pie chart we can clearly see that KP281 is having more sales percentage compared to others.

Recommendation: From the previous chart we understood that most of our customers are around 25 and people with this age are buying KP281 as they are in low-income range. We need to concentrate on the majority to increase our sales even more.

Our customers are Partnered or Single:

Code:

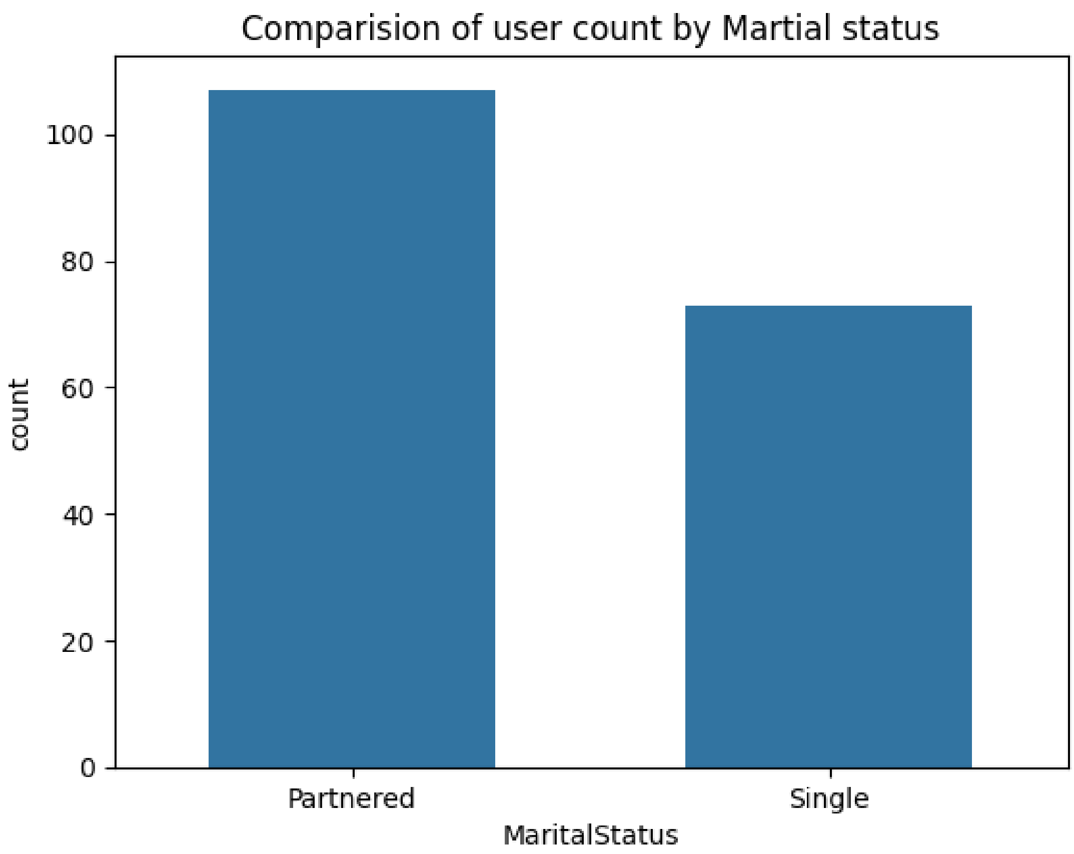
```
df['MaritalStatus'].value_counts()

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

Graphical Representation:

Code:

```
sns.barplot(x = df['MaritalStatus'].value_counts().index, y = df['MaritalStatus'].value_counts(), width = 0.6)
plt.title('Comparision of user count by Martial status')
plt.show()
```

Graph:



Insights: From the graph we can see that majority of our customers are partnered.

Recommendations: We can launch some features specifically for partnered people like games to increase the competitiveness between them. This will increase their fitness levels as well.

Majority of our customers are of which gender:

Code:

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

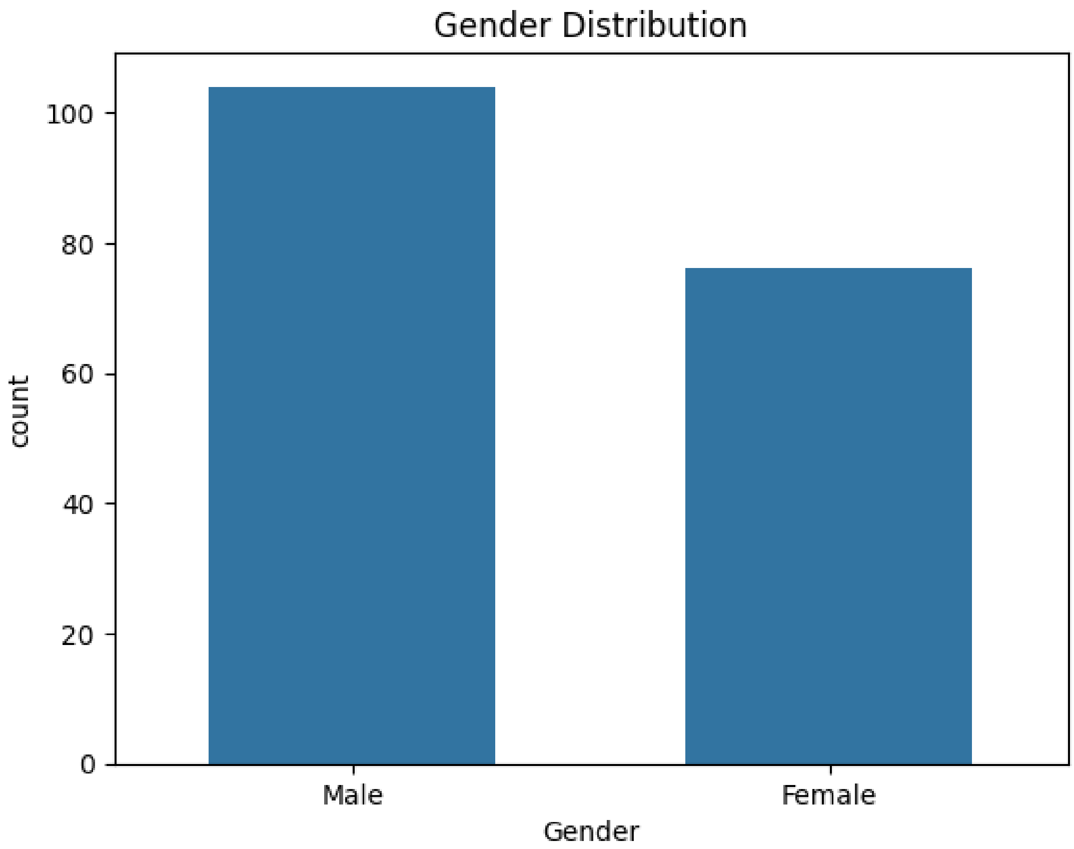
```
Gender
Male      104
Female     76
Name: count, dtype: int64
```

Visual Representation:

Code:

```
[68] sns.barplot(x = df['Gender'].value_counts().index, y = df['Gender'].value_counts(), width = 0.6)
     plt.title('Gender Distribution')
     plt.show()
```

Graph:



Insights: From the graph we can clearly see that most of our customers are Male.

Product sales grouped by Marital Status:

Code:

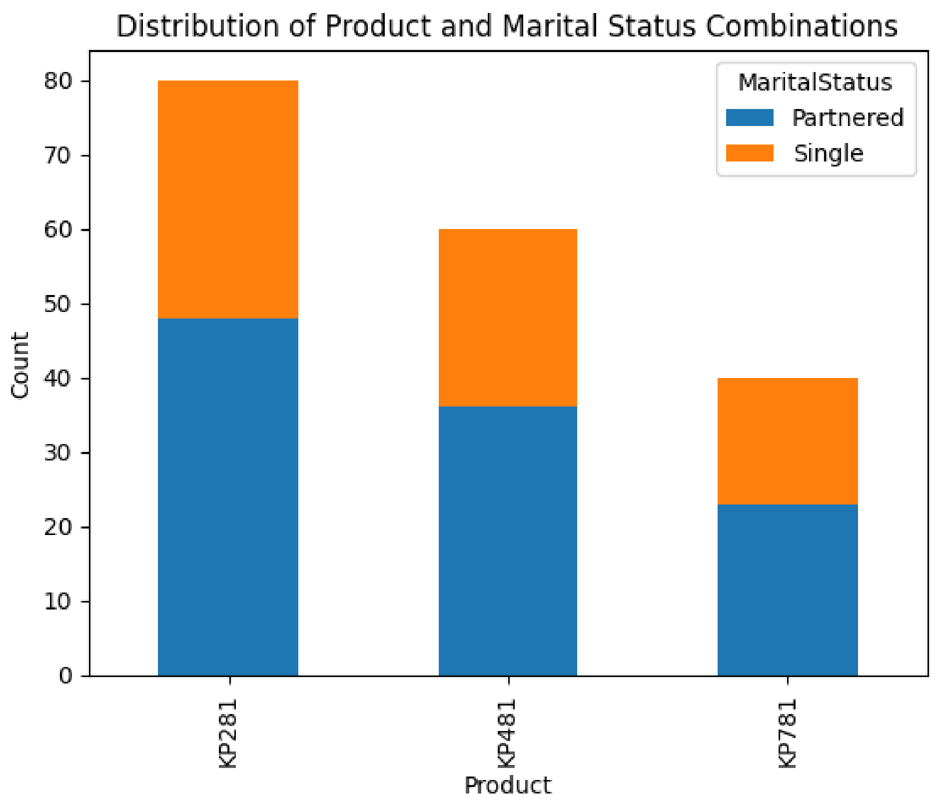
```
[82] df.head()
     grouped_counts = df.groupby(by = ['Product','MaritalStatus']).size()
     print(grouped_counts)

     Product  MaritalStatus
     KP281    Partnered        48
              Single           32
     KP481    Partnered        36
              Single           24
     KP781    Partnered        23
              Single           17
     dtype: int64
```

Graphical Representation:

Code:

```
grouped_counts = grouped_counts.unstack()
grouped_counts.plot(kind='bar', stacked=True)
plt.title('Distribution of Product and Marital Status Combinations')
plt.xlabel('Product')
plt.ylabel('Count')
plt.show()
```

Graph:



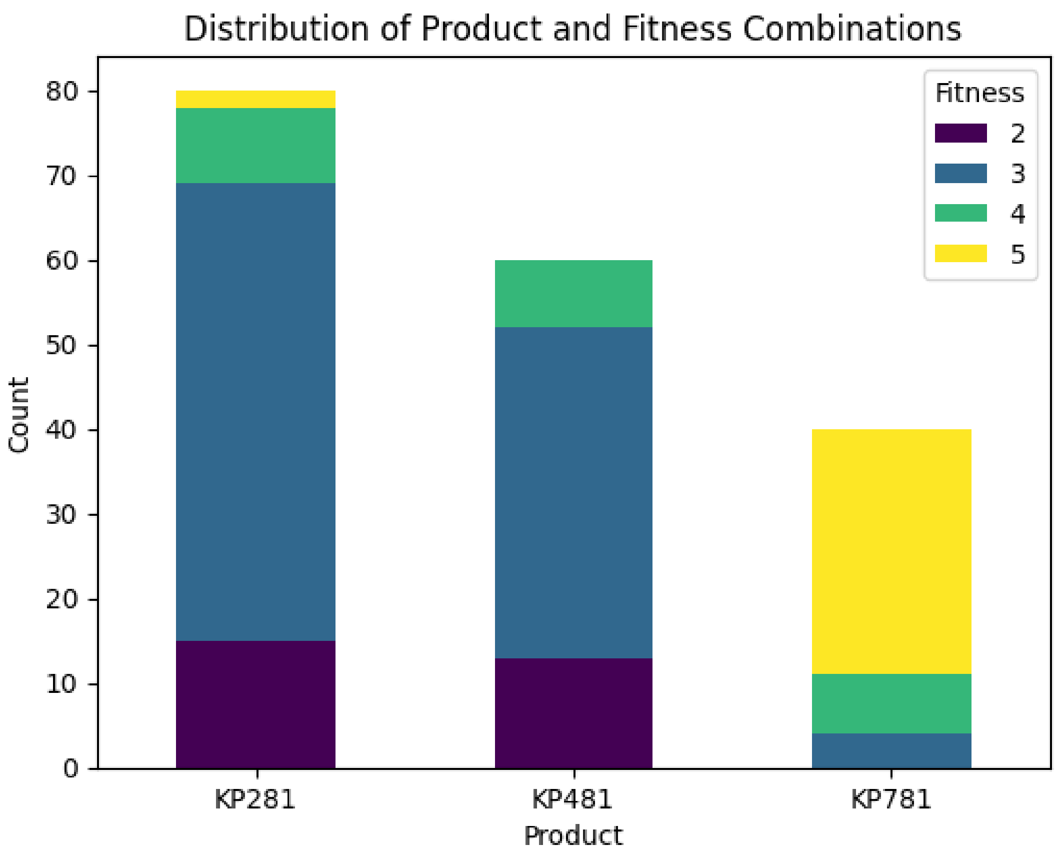
Insights: From the graph we can see that most of customers who bought KP281 are partnered.

Product sales grouped by Fitness:

Code:

```
df.groupby(by = ['Product','Fitness']).size().unstack().plot(kind='bar', stacked=True, colormap = 'viridis')
plt.title('Distribution of Product and Fitness Combinations')
plt.xlabel('Product')
plt.ylabel('Count')
plt.show()
```

Graph:



Insights: From the graph we can see that most the customers who bought KP281 are with low fitness level. People who bought KP781 are with high fitness levels.

Marginal Probability:

```
Product_probs = pd.crosstab(index=df['Product'], columns='count', normalize = True)
Product_Marginal_Percentage = Product_probs*100
print(Product_Marginal_Percentage)
```

```
col_0        count
Product
KP281    44.444444
KP481    33.333333
KP781    22.222222
```

Insights: If we randomly pick a product from the Dataset there's a 44% probability of that product being KP281.

```
[104] Gender_Product_PROb = pd.crosstab(index = df['Product'], columns = df['Gender'], margins = True)
      Gender_Product_PROb
```

Gender	Female	Male	All
Product			
KP281	40	40	80
KP481	29	31	60
KP781	7	33	40
All	76	104	180

Conditional Probability: (KP281 Product with Gender)

```
KP281_F = Gender_Product_PROb['Female']['KP281']/Gender_Product_PROb['Female']['All']
KP281_M = Gender_Product_PROb['Male']['KP281']/Gender_Product_PROb['Male']['All']
print('The Probability of a customer buying KP281 given they are Female is:',round((KP281_F*100),2))
print('The Probability of a customer buying KP281 given they are Male is:',round((KP281_M*100),2))
```

```
The Probability of a customer buying KP281 given they are Female is: 52.63
The Probability of a customer buying KP281 given they are Male is: 38.46
```

KP481 Product with Gender:

```
KP481_F = Gender_Product_PROb['Female']['KP481']/Gender_Product_PROb['Female']['All']
KP481_M = Gender_Product_PROb['Male']['KP481']/Gender_Product_PROb['Male']['All']
print('The Probability of a customer buying KP481 given they are Female is:',round((KP481_F*100),2))
print('The Probability of a customer buying KP481 given they are Male is:',round((KP481_M*100),2))
```

```
The Probability of a customer buying KP481 given they are Female is: 38.16
The Probability of a customer buying KP481 given they are Male is: 29.81
```

KP781 product with Gender"

```
[ ] KP781_F = Gender_Product_PROb['Female']['KP781']/Gender_Product_PROb['Female']['All']
    KP781_M = Gender_Product_PROb['Male']['KP781']/Gender_Product_PROb['Male']['All']
    print('The Probability of a customer buying KP781 given they are Female is:',round((KP781_F*100),2))
    print('The Probability of a customer buying KP781 given they are Male is:',round((KP781_M*100),2))
```

```
The Probability of a customer buying KP781 given they are Female is: 9.21
The Probability of a customer buying KP781 given they are Male is: 31.73
```

Product with Marital status:

```
[ ] Martial_Prob = pd.crosstab(index = df['Product'], columns = df['MaritalStatus'], margins = True)
    Martial_Prob
```

MaritalStatus	Partnered	Single	All
Product			
KP281	48	32	80
KP481	36	24	60
KP781	23	17	40
All	107	73	180

KP281 Product with Marital status:

```
KP281_P = Martial_Prob['Partnered']['KP281']/Martial_Prob['Partnered']['All']
KP281_S = Martial_Prob['Single']['KP281']/Martial_Prob['Single']['All']
print('The Probability of a customer buying KP281 given they are Partnered is:',round((KP281_P*100),2))
print('The Probability of a customer buying KP281 given they are Single is:',round((KP281_S*100),2))
```

```
The Probability of a customer buying KP281 given they are Partnered is: 44.86
The Probability of a customer buying KP281 given they are Single is: 43.84
```

KP481 product with Marital status:

```
KP481_P = Martial_Prob['Partnered']['KP481']/Martial_Prob['Partnered']['All']
KP481_S = Martial_Prob['Single']['KP481']/Martial_Prob['Single']['All']
print('The Probability of a customer buying KP481 given they are Partnered is:',round((KP481_P*100),2))
print('The Probability of a customer buying KP481 given they are Single is:',round((KP481_S*100),2))
```

```
The Probability of a customer buying KP481 given they are Partnered is: 33.64
The Probability of a customer buying KP481 given they are Single is: 32.88
```

KP781 product with Marital status:

```
[ ] KP781_P = Martial_Prob['Partnered']['KP781']/Martial_Prob['Partnered']['All']
    KP781_S = Martial_Prob['Single']['KP781']/Martial_Prob['Single']['All']
    print('The Probability of a customer buying KP781 given they are Partnered is:',round((KP781_P*100),2))
    print('The Probability of a customer buying KP781 given they are Single is:',round((KP781_S*100),2))
```

```
The Probability of a customer buying KP781 given they are Partnered is: 21.5
The Probability of a customer buying KP781 given they are Single is: 23.29
```

Correlation between columns in our Dataset:

```
[60] correlation_matrix = df[['Age','Fitness','Income','Miles','Education','Usage']].corr()
     correlation_matrix
```

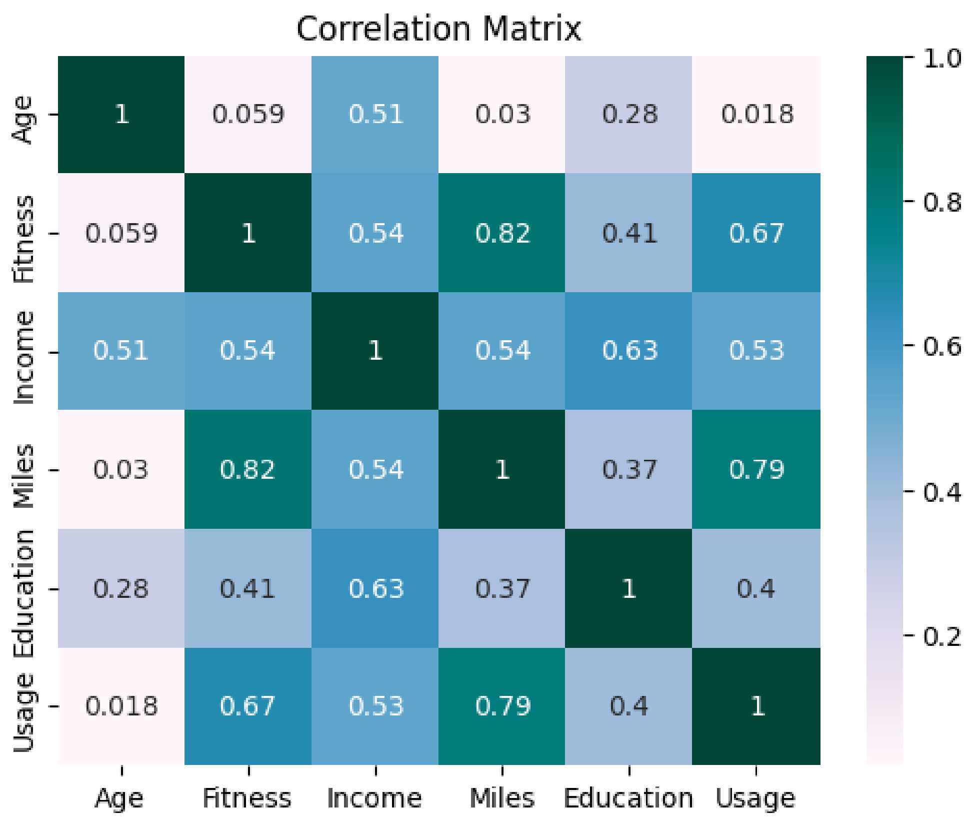
	Age	Fitness	Income	Miles	Education	Usage
Age	1.000000	0.059047	0.514362	0.029636	0.279533	0.018020
Fitness	0.059047	1.000000	0.535945	0.822393	0.410581	0.668606
Income	0.514362	0.535945	1.000000	0.537297	0.628908	0.527707
Miles	0.029636	0.822393	0.537297	1.000000	0.367262	0.786269
Education	0.279533	0.410581	0.628908	0.367262	1.000000	0.395155
Usage	0.018020	0.668606	0.527707	0.786269	0.395155	1.000000

Graphical Representation:

Code:

```
[61] sns.heatmap(correlation_matrix, annot = True, cmap = 'PuBuGn')
     plt.title('Correlation Matrix')
     plt.show()
```

Graph:



Insights: From the graph we see that correlation between Age and Income is high. This means with Age our customers income is also increasing.

We can also see that correlation between age and Fitness is weak. This means as increases Fitness decreasing. Same goes with Miles.

Relationship between Customer Demographics

Code:

```
sns.pairplot(df[['Age','Fitness','Income','Miles','Education','Usage']])
plt.show()
```

Graph:

Insights: From the pair plot we can see the distribution of different variables, we can see people with fitness level as 3 are higher.

2. Customers with income range around 60000 are higher compared to others.

3. People with less age are running more miles and earning less from the scatter plots.