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

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

```
--2024-06-13 17:42:14--  https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/125/original/aerofit_tread
Resolving d2beiqkhq929f0.cloudfront.net (d2beiqkhq929f0.cloudfront.net)... 108.157.172.183, 108.157.172.176, 108.157.1
Connecting to d2beiqkhq929f0.cloudfront.net (d2beiqkhq929f0.cloudfront.net)|108.157.172.183|: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

2024-06-13 17:42:14 (2.89 GB/s) - ‘aerofit_treadmill.csv?1639992749’ saved [7279/7279]
```

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

1. Data Analysis

```
df.sample(5)
```

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
19	KP281	23	Female	15	Partnered	2	2	34110	38
139	KP481	48	Male	16	Partnered	2	3	57987	64
130	KP481	35	Female	16	Single	3	2	50028	64
168	KP781	30	Male	18	Partnered	5	4	103336	160
91	KP481	23	Female	16	Partnered	3	2	43206	74

```
df.shape
```

```
(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
```

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

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

```
df.nunique()
```

```
Product           3
Age              32
```

```
Gender            2
Education         8
MaritalStatus     2
Usage             6
Fitness           5
Income           62
Miles            37
dtype: int64
```

```
df[df.duplicated()].count()
```

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

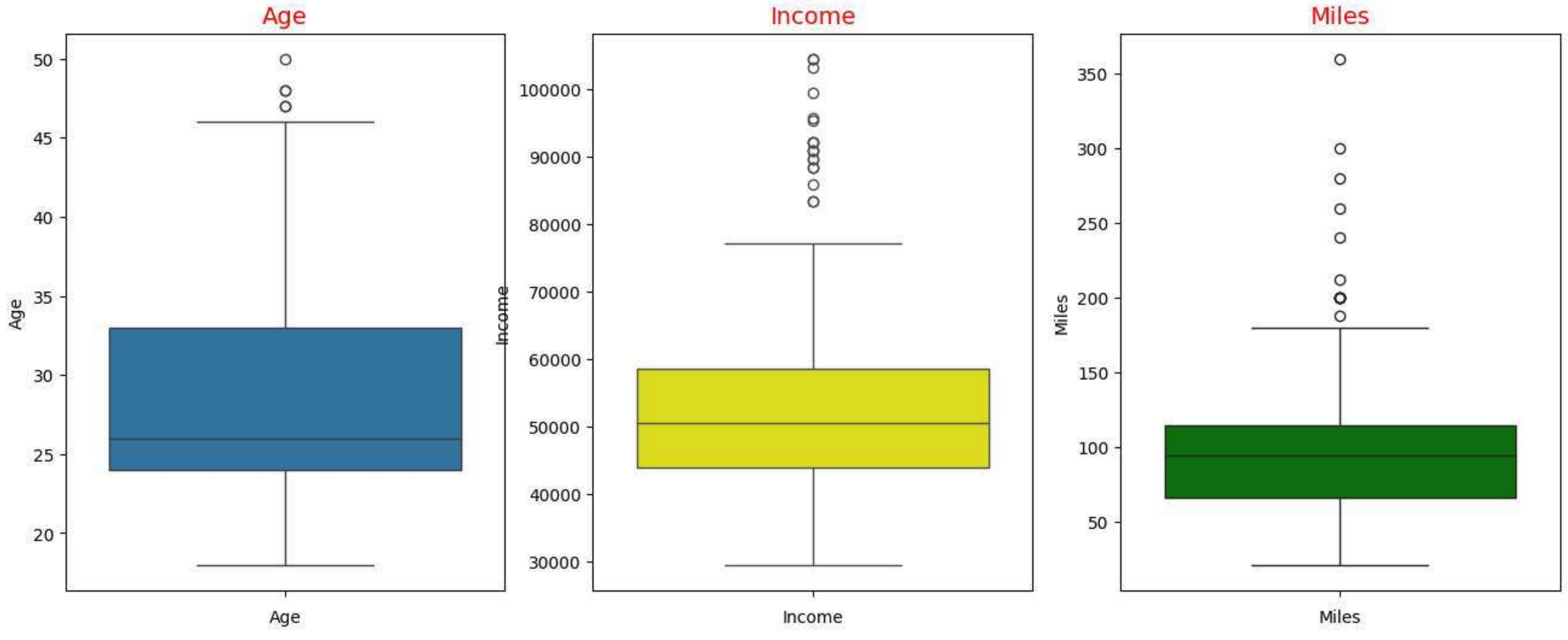
Observation of Data Frame:

1. The dataframe has 180 rows and 9 columns.
2. The dataframe contains 8 columns. The data types of columns 'Product', 'Gender', and 'MaritalStatus' are objects, while the remaining columns are of type int64.
3. The dataframe does not have missing or duplicate values.
4. Columns 'Product', 'Education', 'MaritalStatus', 'Usage', and 'Fitness' have 3, 2, 82, 6, and 5 unique values, respectively. Whereas columns like 'Age', 'Income', and 'Miles' have 32, 62, and 37 unique values, respectively.

2. Detect Outliers

A. Find the outliers for every continuous variable in the dataset

```
plt.figure(figsize=(16,6))

# Boxplot for column age
plt.subplot(1,3,1)
sns.boxplot(data=df["Age"])
plt.title("Age",fontsize=14, color="red")
plt.xlabel("Age")

# Boxplot for column income
plt.subplot(1,3,2)
plt.title("Income",fontsize=14, color="red")
sns.boxplot(data=df["Income"], color="yellow")
plt.xlabel("Income")

# Boxplot for column Miles
plt.subplot(1,3,3)
plt.title("Miles", fontsize=14, color="red")
sns.boxplot(data=df["Miles"], color="green")
plt.xlabel("Miles")
plt.show()
```



Observation:

Age:

1. The youngest customer is 18 years old.
2. The oldest customer is 50 years old.
3. 25% of the customers are 24 years old or younger.
4. 50% of the customers are 26 years old or younger.
5. 75% of the customers are 33 years old or younger.
6. There are outliers with ages between 46 and 50 years.

Income:

1. The minimum income is 29,562.
2. The maximum income is 104,581.
3. 25% of the incomes are 44,058 or lower.
4. 50% of the incomes are 50,596 or lower.
5. 75% of the incomes are 58,668 or lower.
6. There are 19 outliers where the income exceeds 80,000.

Miles covered by users per week:

1. The minimum miles recorded by users per week is 21.
2. The maximum miles recorded by users per week is 360.
3. 25% of users run 66 miles per week or less.
4. 50% of users run 94 miles per week or less.
5. 75% of users run 114 miles per week or less.
6. There are some outliers who run more than 180 miles per week.

B. Clipped Column miles.

```
# Clipping column miles
df["Miles"].max(), df["Miles"].min()
```

```
(360, 21)
```

```
miles_per=np.percentile(df["Miles"],[5,95])
miles_per
```

```
array([ 47., 200.])
```

```
clipped_miles=df["Miles"].clip(47,200)
```

```
clipped_miles
```

```
0      112
1       75
2       66
3       85
4       47
      ...
175    200
176    200
177    160
178    120
179    180
Name: Miles, Length: 180, dtype: int64
```

```
# clipping column age
Age_per=np.percentile(df["Age"],[5,95])
Age_per
```

```
array([20.  , 43.05])
```

```
clipped_age=np.clip(df["Age"],20,43)
clipped_age
```

```
0      20
1      20
2      20
3      20
4      20
```

```
          ..
175       40
176       42
177       43
178       43
179       43
Name: Age, Length: 180, dtype: int64
```

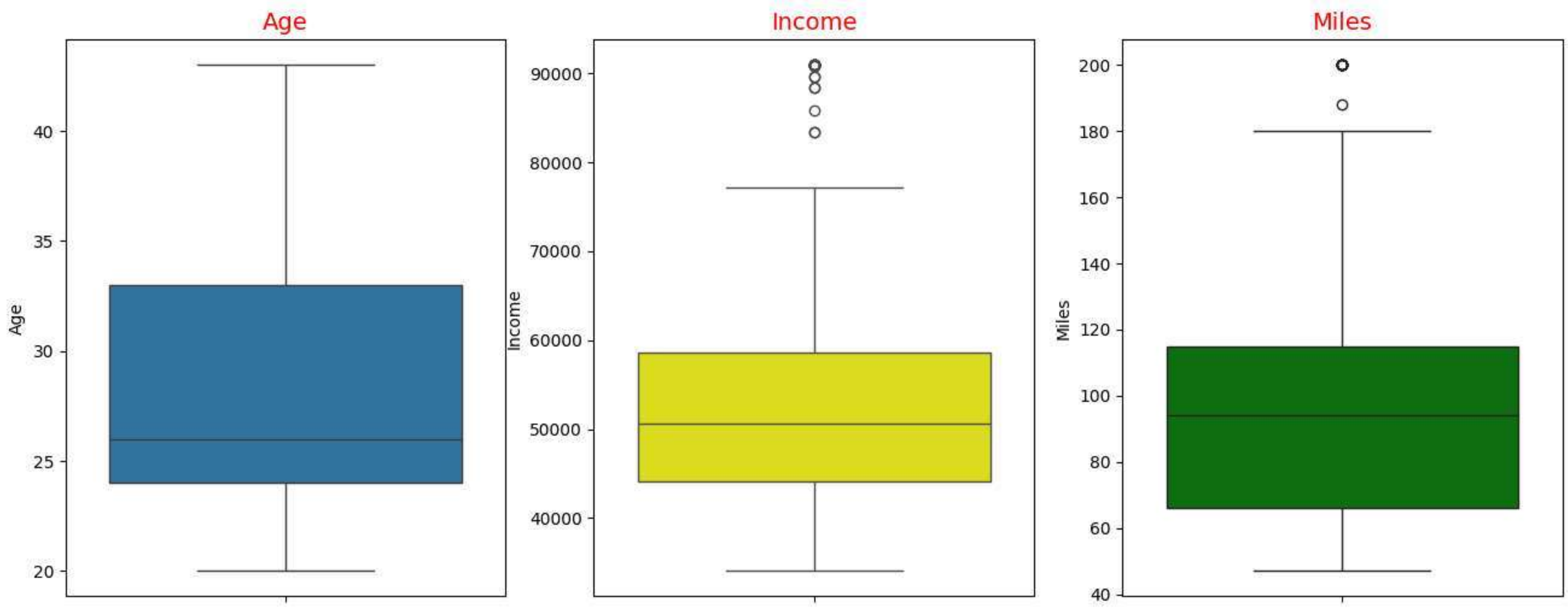
```
# Clipping column income
Income_per=np.percentile(df["Income"],[5,95])
Income_per
```

```
array([34053.15, 90948.25])
```

```
clipped_income=np.clip(df["Income"],34053.15, 90948.25)
clipped_income
```

```
0         34053.15
1         34053.15
2         34053.15
3         34053.15
4         35247.00
            ...
175       83416.00
176       89641.00
177       90886.00
178       90948.25
179       90948.25
Name: Income, Length: 180, dtype: float64
```

```
# plot for clipped columns

plt.figure(figsize=(16,6))

plt.subplot(1,3,1)
plt.title("Age",fontsize=14, color="red")
sns.boxplot(data=clipped_age)

plt.subplot(1,3,2)
plt.title("Income",fontsize=14, color="red")
sns.boxplot(data=clipped_income, color="yellow")

plt.subplot(1,3,3)
plt.title("Miles", fontsize=14, color="red")
sns.boxplot(data=clipped_miles, color="green")

plt.show()
```



Observations after clipping the data:

Age:

1. The youngest customer is 20 years old.
2. The oldest customer is 43 years old.
3. 25% of the customers are 24 years old or younger.
4. 50% of the customers are 26 years old or younger.
5. 75% of the customers are 33 years old or younger.

6. Currently, there are no outliers.

Income:

1. After using the clip() function, the new minimum income is $34,053 and the maximum income is 90,948.
2. The first quartile (Q1), second quartile (Q2), and third quartile (Q3) are the same, which is 44,058, 50,596, and 58,668, respectively.
3. Now, we have fewer outliers.

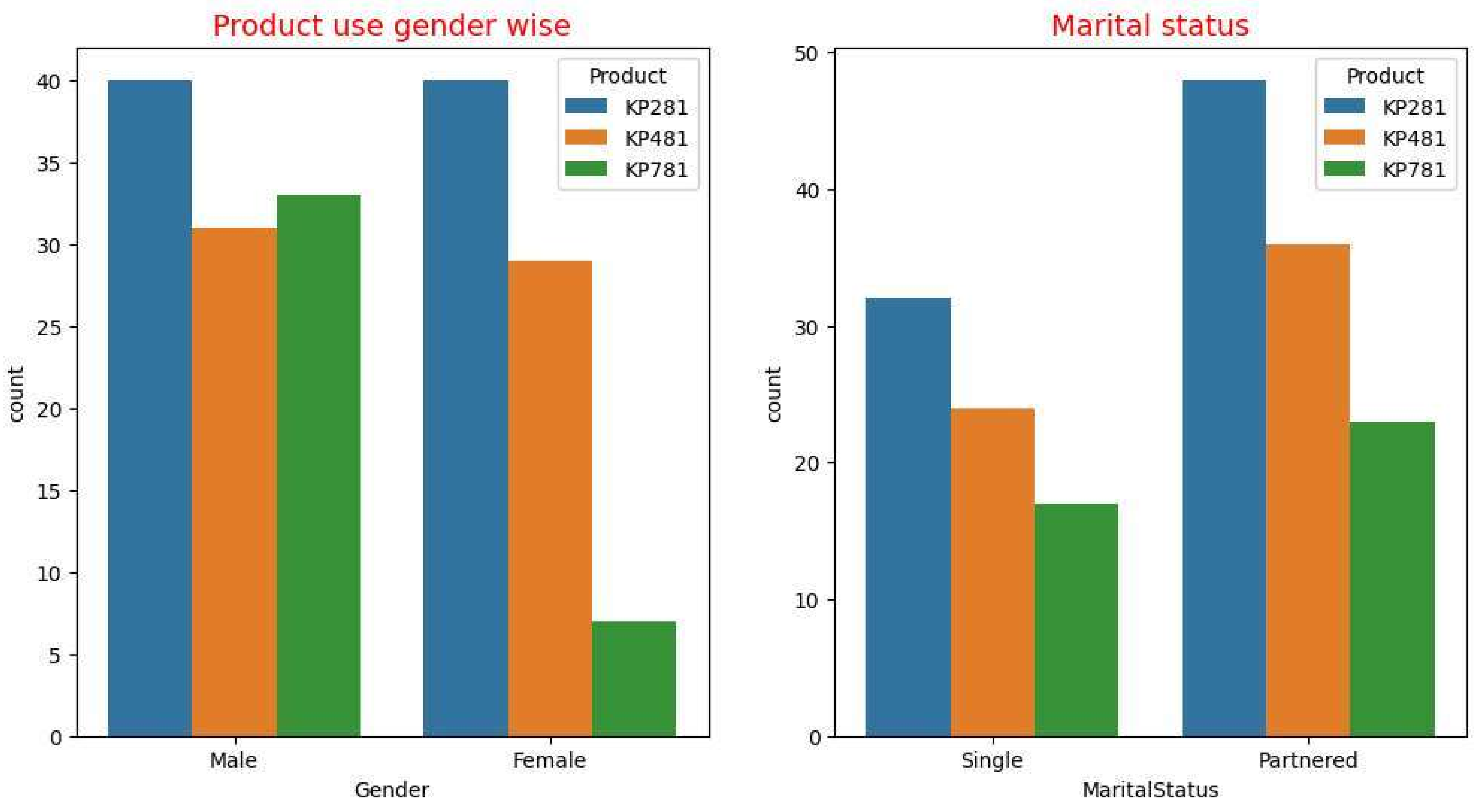
Miles covered by users per week:

1. After using the clip() function, the new minimum miles is 43 and the maximum miles is 200.
2. The first quartile (Q1), second quartile (Q2), and third quartile (Q3) are the same, which is 66, 94, and 114 miles, respectively.
3. Now, we have fewer outliers who run above 180 miles.

3. Check if features like marital status, Gender, and age have any effect on the product purchased

A. Find if there is any relationship between the categorical variables and the output variable in the data.

```
# Countplot of Gender
plt.figure(figsize=(12,6))

plt.subplot(1,2,1)
plt.title("Product use gender wise",fontsize=14, color="red")
sns.countplot(data=df, x=df["Gender"], hue=df["Product"])

# Countplot of Marital Status
plt.subplot(1,2,2)
plt.title("Marital status",fontsize=14, color="red")
sns.countplot(data=df, x=df["MaritalStatus"], hue=df["Product"])

plt.show()
```



Observation:

Usage of the product by gender:

1. With the help of the plot, we can observe that Product KP281 has an equal proportion of both genders.
2. Product KP481 has nearly equal numbers of male and female customers.
3. Product KP781 shows a significant difference between male and female customers, with 82% of customers being male.

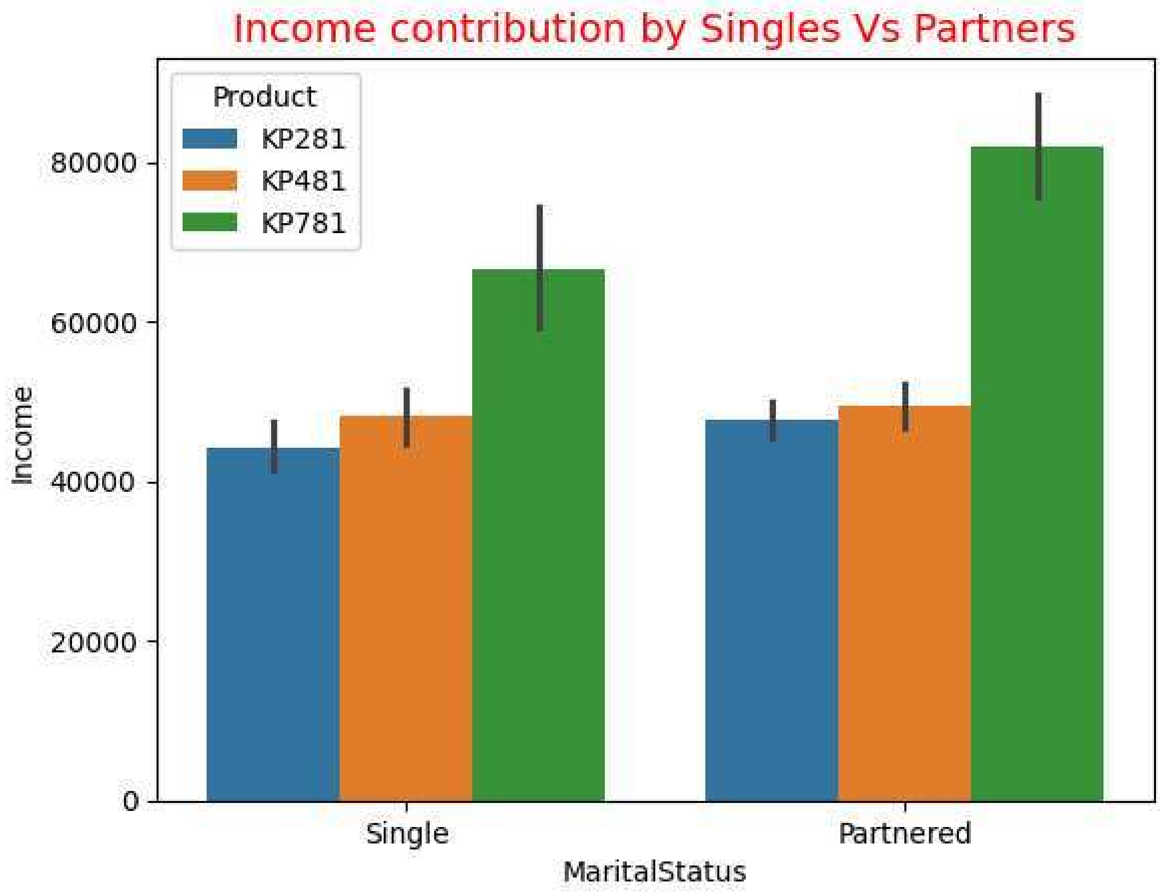
Product usage by marital status:

1. We observe that married individuals are the predominant customers in all three categories of products.

2. Products KP281 and KP481 exhibit a ratio of 60:40 between married and single customers.
3. Product KP481 has a ratio of 58:42 between married and single customers.

```
# Income contribution by Single vs Partners
sns.barplot(data=df, x=df["MaritalStatus"],y=df["Income"],hue=df["Product"])
plt.title("Income contribution by Singles Vs Partners",fontsize=14, color="red")
```

```
Text(0.5, 1.0, 'Income contribution by Singles Vs Partners')
```



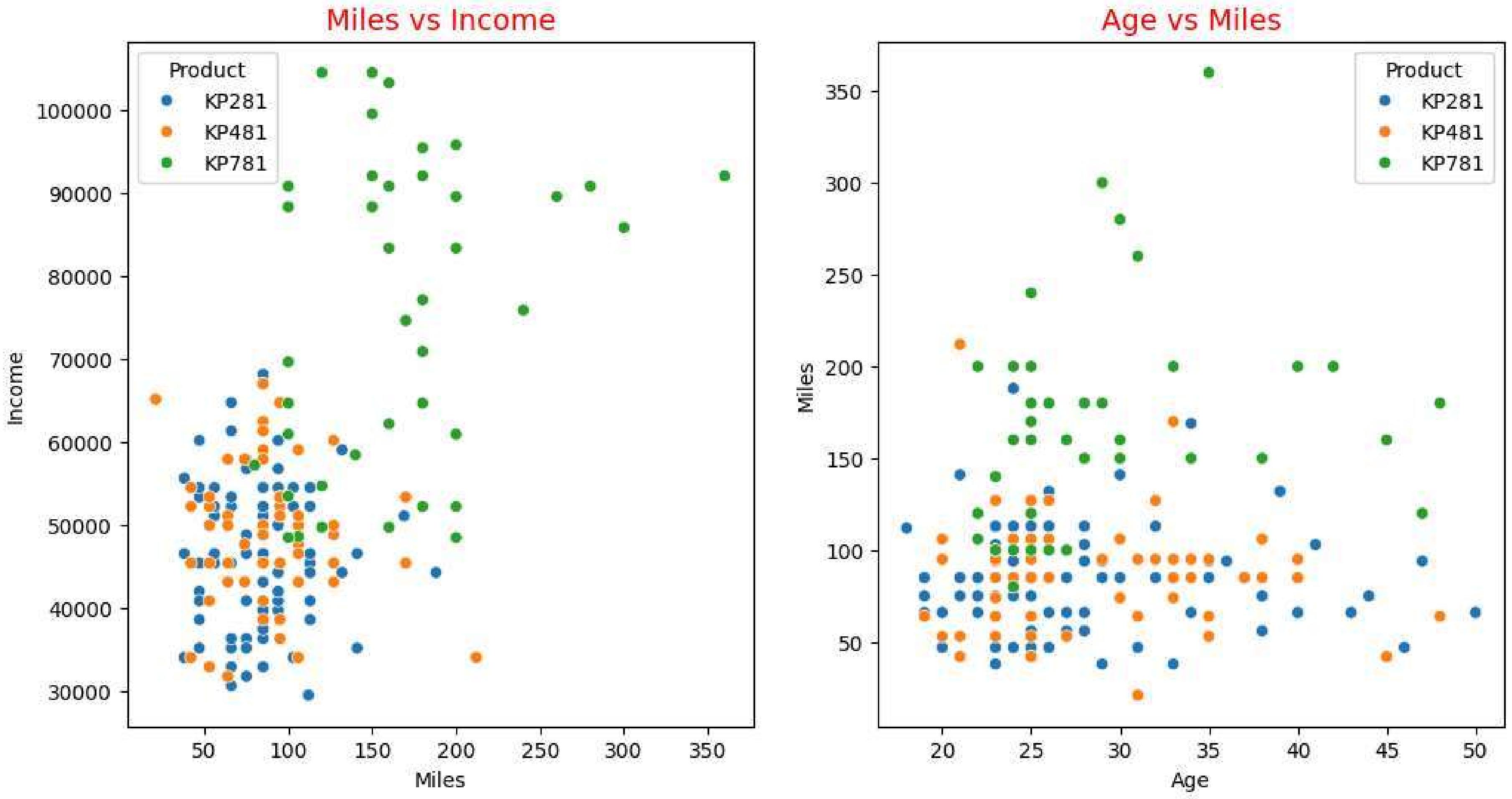
Observation:

As we know that married individuals are the primary customers for all three products, it's evident from this plot that they also contribute more to the income generated by each of the three products.

B. Relation between continus variable and output variable:

```
# Relation between Miles, Income and Product
plt.figure(figsize=(12,6))
plt.subplot(1,2,1)
sns.scatterplot(data=df, x=df["Miles"], y=df["Income"], hue=df["Product"])
plt.title("Miles vs Income",fontsize=14, color="red")

#Relation between Age, Miles and Product
plt.subplot(1,2,2)
sns.scatterplot(data=df, x=df["Age"], y=df["Miles"], hue=df["Product"])
plt.title("Age vs Miles",fontsize=14, color="red")
```

```
Text(0.5, 1.0, 'Age vs Miles')
```



Obesrvation:

MILES VS INCOME

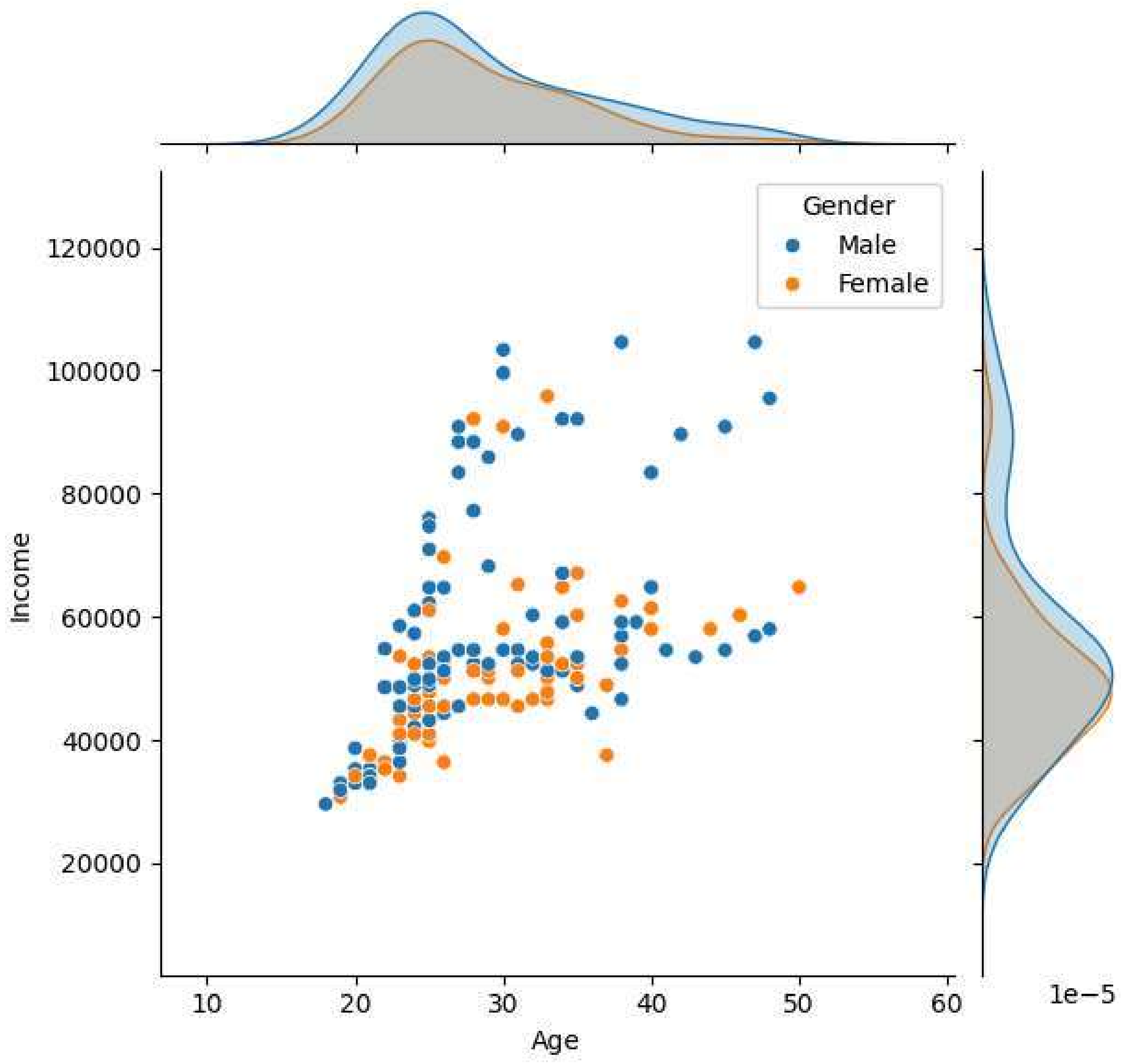
1. By examining the Income vs Miles plot, we can observe that most customers fall within the range of 40 to 180 miles and have an income between 30,000 to 70,000.
2. Additionally, the majority of these customers are users of Product KP281 or KP781. However, there are outliers who have more than 200 miles and an income exceeding 80,000.
3. Interestingly, both of these outliers are users of Product KP781.
4. KP281 contribute most to the sale and top by 38.30%. followed by KP781 31.21% and KP481 30.29%.

AGE VS MILES:

1. We can observe that the majority of customers belong to the age range of 20-40.
2. Most customers run between 40 to 150 miles per week, but we do have outliers in both age and miles.
3. It's evident that almost all customers who run more than 150 miles per week use Product KP781.

```
# Joint plot of Age and Income.

sns.jointplot(data=df, x=df["Age"], y=df["Income"], hue=df["Gender"])
```

```
<seaborn.axisgrid.JointGrid at 0x7bb026e11480>
```



AGE VS INCOME:

1. The majority of customers belong to the 20 to 40 age group and contribute between 30,000 to 70,000.
2. Both males and females contribute equally to this range. However, we do have some outliers who spend more than 80,000, and most of these outliers are male.

4 Representing the Probability

A. Find the marginal probability (what percent of customers have purchased KP281, KP481, or KP781)

```
cross_table=pd.crosstab(index=df["Product"], columns=df["Product"], margins=True)
cross_table.reset_index(inplace=True)
cross_table
```

Product	Product	KP281	KP481	KP781	All
0	KP281	80	0	0	80
1	KP481	0	60	0	60
2	KP781	0	0	40	40
3	All	80	60	40	180

Next steps: Generate code with `cross_table` View recommended plots

```
# Probability of KP281
P_KP281=cross_table["KP281"][0]/cross_table["All"][3]
P_KP481=cross_table["KP481"][1]/cross_table["All"][3]
P_KP781=cross_table["KP781"][2]/cross_table["All"][3]

print("% of Customer have Purchase KP281:", round(P_KP281*100,2))
print("% of Customer have Purchase KP481:", round(P_KP481*100,2))
print("% of Customer have Purchase KP781:", round(P_KP781*100,2))
```

```
% of Customer have Purchase KP281: 44.44
% of Customer have Purchase KP481: 33.33
% of Customer have Purchase KP781: 22.22
```

Observation: The sales contribution of each product is as follows:

1. KP281 contributes 44.4%.

2. KP481 contributes 33.3%.
3. KP781 contributes 22.2%.

B. Find the probability that the customer buys a product based on each column.

```
# The probability of sales based on gender
cross_table_g=pd.crosstab(index=df["Product"], columns=df["Gender"],margins=True)


cross_table_g.reset_index(inplace=True)
cross_table_g.columns.name=None
cross_table_g
```

	Product	Female	Male	All
0	KP281	40	40	80
1	KP481	29	31	60
2	KP781	7	33	40
3	All	76	104	180

Next steps: Generate code with `cross_table_g` View recommended plots

```
# Percentage of Female Customer and Male Customer

for i in range(4):
   print(f"Product: {cross_table_g['Product'][i]}")
   P_female = cross_table_g["Female"][i] / cross_table_g["All"][i]
   P_male = cross_table_g["Male"][i] / cross_table_g["All"][i]

   print(f"The proportion of Female customers : {round(P_female * 100)}%")
   print(f"The proportion of Male customers: {round(P_male * 100)}%")
   print()
```

```
Product: KP281
The proportion of Female customers : 50%
The proportion of Male customers: 50%

Product: KP481
The proportion of Female customers : 48%
The proportion of Male customers: 52%

Product: KP781
The proportion of Female customers : 18%
The proportion of Male customers: 82%

Product: All
The proportion of Female customers : 42%
The proportion of Male customers: 58%
```

```
# The probability of product sales based on fitness level
cross_tab_f=pd.crosstab(index=df["Product"], columns=df["Fitness"], margins=True)
cross_tab_f.reset_index(inplace=True)
cross_tab_f.columns.name=None
cross_tab_f
```

	Product	1	2	3	4	5	All
0	KP281	1	14	54	9	2	80
1	KP481	1	12	39	8	0	60
2	KP781	0	0	4	7	29	40
3	All	2	26	97	24	31	180

Next steps: Generate code with `cross_tab_f` View recommended plots

```
for i in range(3):
  print("Product:",cross_tab_f['Product'][i])

  for n in range(1,6):
    Level=cross_tab_f[n][i]/cross_tab_f["All"][i]
    print(f"The proportion of level {n} customers : {round(Level * 100)}%")
  print()
```

```
Product: KP281
The proportion of level 1 customers : 1%
The proportion of level 2 customers : 18%
The proportion of level 3 customers : 68%
The proportion of level 4 customers : 11%
The proportion of level 5 customers : 2%

Product: KP481
The proportion of level 1 customers : 2%
The proportion of level 2 customers : 20%
The proportion of level 3 customers : 65%
The proportion of level 4 customers : 13%
The proportion of level 5 customers : 0%

Product: KP781
The proportion of level 1 customers : 0%
The proportion of level 2 customers : 0%
The proportion of level 3 customers : 10%
The proportion of level 4 customers : 18%
The proportion of level 5 customers : 72%
```

```
# Per-week treadmill usage

cross_tab_u=pd.crosstab(df["Product"], df["Usage"], margins=True)
cross_tab_u.reset_index(inplace=True)
cross_tab_u.columns.name=None
cross_tab_u
```

	Product	2	3	4	5	6	7	All
0	KP281	19	37	22	2	0	0	80
1	KP481	14	31	12	3	0	0	60
2	KP781	0	1	18	12	7	2	40
3	All	33	69	52	17	7	2	180

Next steps: Generate code with cross_tab_u View recommended plots

```
for i in range(3):
    product = cross_tab_u['Product'][i]
    print("Product:", product)

    for day in range(2, 8):
        usage = cross_tab_u[day][i] / cross_tab_u["All"][i]
        print(f"The proportion of customers who usage treadmill {day} days per week: {round(usage * 100)}%")

    print()
```

```
Product: KP281
The proportion of customers who usage treadmill 2 days per week: 24%
The proportion of customers who usage treadmill 3 days per week: 46%
The proportion of customers who usage treadmill 4 days per week: 28%
The proportion of customers who usage treadmill 5 days per week: 2%
The proportion of customers who usage treadmill 6 days per week: 0%
The proportion of customers who usage treadmill 7 days per week: 0%

Product: KP481
The proportion of customers who usage treadmill 2 days per week: 23%
The proportion of customers who usage treadmill 3 days per week: 52%
The proportion of customers who usage treadmill 4 days per week: 20%
The proportion of customers who usage treadmill 5 days per week: 5%
The proportion of customers who usage treadmill 6 days per week: 0%
The proportion of customers who usage treadmill 7 days per week: 0%

Product: KP781
The proportion of customers who usage treadmill 2 days per week: 0%
The proportion of customers who usage treadmill 3 days per week: 2%
The proportion of customers who usage treadmill 4 days per week: 45%
The proportion of customers who usage treadmill 5 days per week: 30%
The proportion of customers who usage treadmill 6 days per week: 18%
The proportion of customers who usage treadmill 7 days per week: 5%
```

C. Find the Conditional Probability that an event occurs given that another event has occurred.

```
cross_table_g
```

	Product	Female	Male	All
0	KP281	40	40	80
1	KP481	29	31	60
2	KP781	7	33	40
3	All	76	104	180

Next steps: Generate code with cross_table_g View recommended plots

```
# Usage of each product given that the user is male or female
for i in range(3):
  print(f"Product: {cross_table_g['Product'][i]}")
  print("The probability of product usage given that the customer is female:", round(cross_table_g["Female"][i] / cross_
  print("The probability of product usage given that the customer is male:", round(cross_table_g["Male"][i] / cross_tabl
  print()
```

```
Product: KP281
The probability of product usage given that the customer is female: 53 %
The probability of product usage given that the customer is male: 38 %

Product: KP481
The probability of product usage given that the customer is female: 38 %
The probability of product usage given that the customer is male: 30 %

Product: KP781
The probability of product usage given that the customer is female: 9 %
The probability of product usage given that the customer is male: 32 %
```

```
# Cross table of Column Product and Marital status.
cross_tab_m=pd.crosstab(index=df["Product"], columns=df["MaritalStatus"], margins=True)
cross_tab_m.reset_index(inplace=True)
cross_tab_m.columns.name=None
cross_tab_m
```

	Product	Partnered	Single	All
0	KP281	48	32	80
1	KP481	36	24	60
2	KP781	23	17	40
3	All	107	73	180

Next steps: Generate code with cross_tab_m View recommended plots

```
# The conditional probability of product sales based on marital status

for i in range(3):
  print(f"Product: {cross_tab_m['Product'][i]}")
  print("Probability of Product usge given that customer is single :", round(cross_tab_m["Single"][i]/ cross_tab_m.iloc[3]
  print("Probability of Product usge given that customer is partner:", round(cross_tab_m["Partnered"][i] / cross_tab_m.ilo
  print()
```

```
Product: KP281
Probability of Product usge given that customer is single : 44 %
Probability of Product usge given that customer is partner: 45 %

Product: KP481
Probability of Product usge given that customer is single : 33 %
Probability of Product usge given that customer is partner: 34 %

Product: KP781
Probability of Product usge given that customer is single : 23 %
Probability of Product usge given that customer is partner: 21 %
```

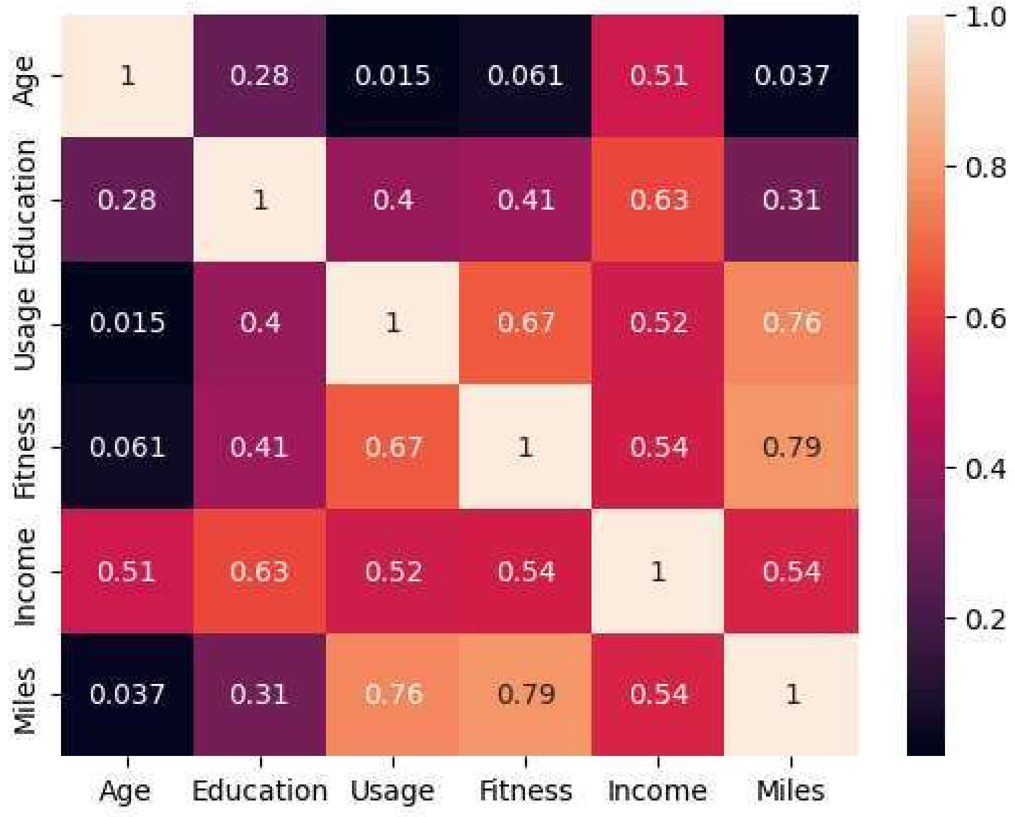
5. The correlation among different factors

```
df_corr=df[["Age","Education","Usage","Fitness","Income","Miles"]].corr()
df_corr*100
```

	Age	Education	Usage	Fitness	Income	Miles
Age	100.000000	28.049567	1.506447	6.110454	51.341369	3.661757
Education	28.049567	100.000000	39.515522	41.058079	62.582735	30.728428
Usage	1.506447	39.515522	100.000000	66.860557	51.953723	75.913048
Fitness	6.110454	41.058079	66.860557	100.000000	53.500532	78.570174
Income	51.341369	62.582735	51.953723	53.500532	100.000000	54.347326
Miles	3.661757	30.728428	75.913048	78.570174	54.347326	100.000000

```
sns.heatmap(df[["Age","Education","Usage","Fitness","Income","Miles"]].corr(), annot=True)
```

```
<Axes: >
```



Observation: Corelation between Columns

Upon analyzing the ubove plot we can say:

1. Age display high correlation with income, with education showing a secondary correlation.
2. Usage displays a significant correlation with miles, with fitness following closely. Furthermore, fitness is strongly correlated with miles, age, income, and education.
3. Income shows a pronounced correlation with education, with miles, fitness, and usage following suit. These correlations provide valuable insights into the interplay between various factors within the dataset.

6. Customer profiling and recommendation

Product KP281:

1. Market Share: 44.4%.
2. Target Age Group: 18-40.
3. Usage Preference: 53% for running ≤ 150 miles/week.
4. Income Range: 20,000 to 70,000 (In $) (51%).
5. Gender Preference: Female (53%).
6. Marital Status Preference: Single (44%), Partner (45%).
7. Treadmill Usage: 2 days: 24%, 3 days: 46%, 4 days: 28%, 5 days: 2%.
8. Fitness Level: Level 3 (68%).

Product KP481:

1. Market Share: 33.2%.
2. Target Age Group: 18-40.
3. Usage Preference: 38% for running ≤ 150 miles/week.
4. Income Range: 20,000 to 70,000 (In $) (38%).
5. Gender Preference: Female (38%).
6. Marital Status Preference: Single (33%), Partner (34%).

7. Treadmill Usage: 2 days: 23%, 3 days: 52%, 4 days: 20%, 5 days: 5%.
8. Fitness Level: Level 3 (65%).

Product KP781:

1. Market Share: 22.2%.
2. Target Age Group: 23-35.
3. Usage Preference: 84% for running 150 miles/week.
4. Income Range: > 70,000(In $) (100%).
5. Gender Preference: Male (32%).
6. Marital Status Preference: Single (23%), Partner (21%).
7. Treadmill Usage: 4 days: 45%, 5 days: 30%, 6 days: 18%, 7 days: 5%.
8. Fitness Level: Level 5 (72%).

Insights:

1. KP281 is the most popular product, which means a lot of people like it and want to buy it. To keep this popularity, companies should advertise it well and make it even better.

2. Many young people, between 18 and 40 years old, really like KP281 and KP481. This means companies should advertise these products in a way that appeals to young people's tastes and lifestyles.

3. KP781 is liked by people who have more money,($70,000 and more). By adding special features to make it even better, companies can increse profit on the product.

4. Knowing how often people use treadmills and how fit they are can help companies make better products and decide how to sell them. For example, KP781 users use treadmills a lot and like intense workouts, like running 150 miles (and more in some cases) a week. By using this information companies can focus on making treadmills to target those users.

5. Customizing ads and products based on whether someone is male or female, or if they're single or in a relationship, can make customers happier. For instance, KP281 is a bit more popular with single women, while KP781 is liked more by single men. This means companies can approch target users.

6. Looking at age, income, education, product use, and fitness levels can give companies insight into what customers prefer. For example, there's a strong correlation between income, treadmill usage, and fitness level. This suggests that people who work out more and are fitter are willing to spend more.

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
!pip install nbconvert
```

```
Requirement already satisfied: nbconvert in /usr/local/lib/python3.10/dist-packages (6.5.4)
Requirement already satisfied: lxml in /usr/local/lib/python3.10/dist-packages (from nbconvert) (4.9.4)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.10/dist-packages (from nbconvert) (4.12.3)
Requirement already satisfied: bleach in /usr/local/lib/python3.10/dist-packages (from nbconvert) (6.1.0)
Requirement already satisfied: defusedxml in /usr/local/lib/python3.10/dist-packages (from nbconvert) (0.7.1)
Requirement already satisfied: entrypoints>=0.2.2 in /usr/local/lib/python3.10/dist-packages (from nbconvert) (0.4)
Requirement already satisfied: jinja2>=3.0 in /usr/local/lib/python3.10/dist-packages (from nbconvert) (3.1.4)
Requirement already satisfied: jupyter-core>=4.7 in /usr/local/lib/python3.10/dist-packages (from nbconvert) (5.7.2)
Requirement already satisfied: jupyterlab-pygments in /usr/local/lib/python3.10/dist-packages (from nbconvert) (0.3.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from nbconvert) (2.1.5)
Requirement already satisfied: mistune<2,>=0.8.1 in /usr/local/lib/python3.10/dist-packages (from nbconvert) (0.8.4)
Requirement already satisfied: nbclient>=0.5.0 in /usr/local/lib/python3.10/dist-packages (from nbconvert) (0.10.0)
Requirement already satisfied: nbformat>=5.1 in /usr/local/lib/python3.10/dist-packages (from nbconvert) (5.10.4)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from nbconvert) (24.1)
Requirement already satisfied: pandocfilters>=1.4.1 in /usr/local/lib/python3.10/dist-packages (from nbconvert) (1.5.1
Requirement already satisfied: pygments>=2.4.1 in /usr/local/lib/python3.10/dist-packages (from nbconvert) (2.16.1)
Requirement already satisfied: tinycss2 in /usr/local/lib/python3.10/dist-packages (from nbconvert) (1.3.0)
Requirement already satisfied: traitlets>=5.0 in /usr/local/lib/python3.10/dist-packages (from nbconvert) (5.7.1)
Requirement already satisfied: platformdirs>=2.5 in /usr/local/lib/python3.10/dist-packages (from jupyter-core>=4.7->n
Requirement already satisfied: jupyter-client>=6.1.12 in /usr/local/lib/python3.10/dist-packages (from nbclient>=0.5.0
Requirement already satisfied: fastjsonschema>=2.15 in /usr/local/lib/python3.10/dist-packages (from nbformat>=5.1->nb
Requirement already satisfied: jsonschema>=2.6 in /usr/local/lib/python3.10/dist-packages (from nbformat>=5.1->nbconve
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist-packages (from beautifulsoup4->nbconver
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.10/dist-packages (from bleach->nbconvert) (1.16.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.10/dist-packages (from bleach->nbconvert) (0.5.1
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=2.6->nbforma
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /usr/local/lib/python3.10/dist-packages (from j
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=2.6->n
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=2.6->nbform
Requirement already satisfied: pyzmq>=13 in /usr/local/lib/python3.10/dist-packages (from jupyter-client>=6.1.12->nbcl
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.10/dist-packages (from jupyter-client>=6
Requirement already satisfied: tornado>=4.1 in /usr/local/lib/python3.10/dist-packages (from jupyter-client>=6.1.12->n
```