Business Case: Aerofit - Descriptive Statistics & Probability About Aerofit Aerofit is a leading brand in the field of fitness equipment. Aerofit provides a product range including machines such as treadmills, exercise bikes, gym equipment, and fitness accessories to cater to the needs of all categories of people.

Business Problem

The market research team at AeroFit wants to identify the characteristics of the target audience for each type of treadmill offered by the company, to provide a better recommendation of the treadmills to the new customers. The team decides to investigate whether there are differences across the product with respect to customer characteristics.

Product Portfolio:

The KP281 is an entry-level treadmill that sells for $1,500.

The KP481 is for mid-level runners that sell for $1,750.

The KP781 treadmill is having advanced features that sell for $2,500.

```
#Uploding dataset
from google.colab import files
uploaded = files.upload()
```

Choose Files aerofit_treadmill.csv

- **aerofit_treadmill.csv**(text/csv) - 7279 bytes, last modified: 7/11/2024 - 100% done

```
Saving aerofit_treadmill.csv to aerofit_treadmill.csv
```

```
df = pd.read_csv('aerofit_treadmill.csv')
df.head(10)
```

	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
5	KP281	20	Female	14	Partnered	3	3	32973	66
6	KP281	21	Female	14	Partnered	3	3	35247	75
7	KP281	21	Male	13	Single	3	3	32973	85
8	KP281	21	Male	15	Single	5	4	35247	141
9	KP281	21	Female	15	Partnered	2	3	37521	85

Next steps: Generate code with df View recommended plots

```
print(f"Number of rows: {df.shape[0]}\nNumber of columns: {df.shape[1]}")
```

```
Number of rows: 180
Number of columns: 9
```

```
df.ndim
```

```
2
```

```
df.columns
```

```
Index(['Product', 'Age', 'Gender', 'Education', 'MaritalStatus', 'Usage',
       'Fitness', 'Income', 'Miles'],
      dtype='object')
```

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

```
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.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

```
df.describe(include='object').T
```

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

```
df.describe(include="all")
```

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness
count	180	180.000000	180	180.000000	180	180.000000	180.000000
unique	3	NaN	2	NaN	2	NaN	NaN
top	KP281	NaN	Male	NaN	Partnered	NaN	NaN
freq	80	NaN	104	NaN	107	NaN	NaN
mean	NaN	28.788889	NaN	15.572222	NaN	3.455556	3.311111
std	NaN	6.943498	NaN	1.617055	NaN	1.084797	0.958869
min	NaN	18.000000	NaN	12.000000	NaN	2.000000	1.000000
25%	NaN	24.000000	NaN	14.000000	NaN	3.000000	3.000000
50%	NaN	26.000000	NaN	16.000000	NaN	3.000000	3.000000
75%	NaN	33.000000	NaN	16.000000	NaN	4.000000	4.000000
max	NaN	50.000000	NaN	21.000000	NaN	7.000000	5.000000

```
#finding unique values
df.nunique()
```

```
Product          3
Age             32
Gender           2
Education        8
MaritalStatus    2
Usage            6
Fitness          5
Income          62
Miles           37
dtype: int64
```

Double-click (or enter) to edit

```
# cheacking for missing values
df.isnull().sum()/len(df)*100
```

```
Product          0.0
Age              0.0
Gender           0.0
Education        0.0
MaritalStatus    0.0
Usage            0.0
Fitness          0.0
Income           0.0
Miles            0.0
dtype: float64
```

```
print('\nColumns with missing value:')
print(df.isnull().any())
```

```

Columns with missing value:
Product          False
Age              False
Gender           False
Education        False
MaritalStatus    False
Usage            False
Fitness          False
Income           False
Miles            False
dtype: bool
```

Observations:

Missing Values: There are no missing values in the dataset, ensuring a complete dataset for analysis.

Unique Products: There are 3 unique products in the dataset: KP281, KP481, and KP781. KP281 is the most frequent product, indicating it might be the most popular or best-selling product.

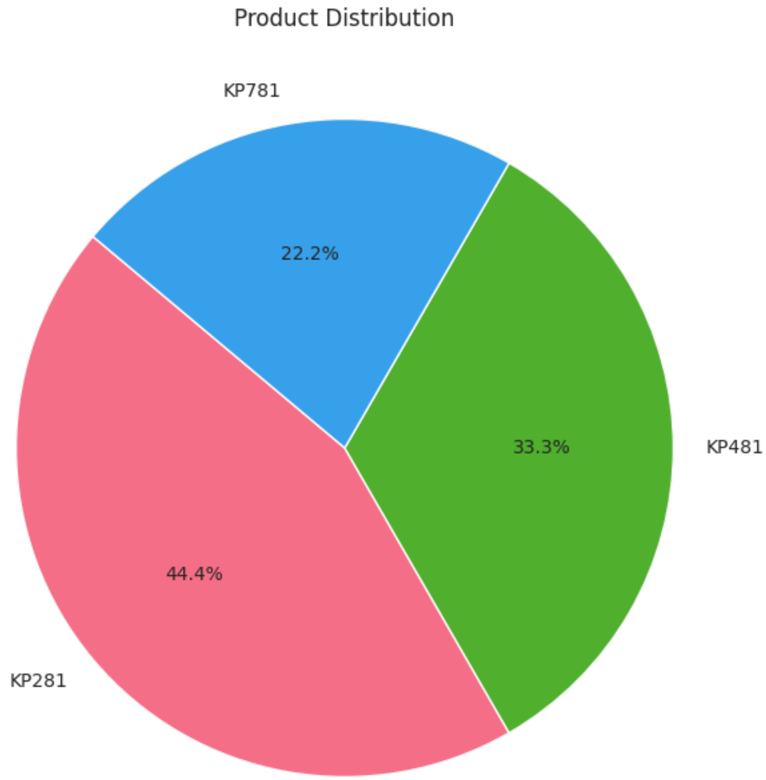
Age Distribution: The age of customers ranges from 18 to 50 years. The mean age is approximately 28.79 years. 75% of the customers are 33 years old or younger.

Education: Most of the customers have 16 years of education or less, with 75% of the customers having up to 16 years of education.

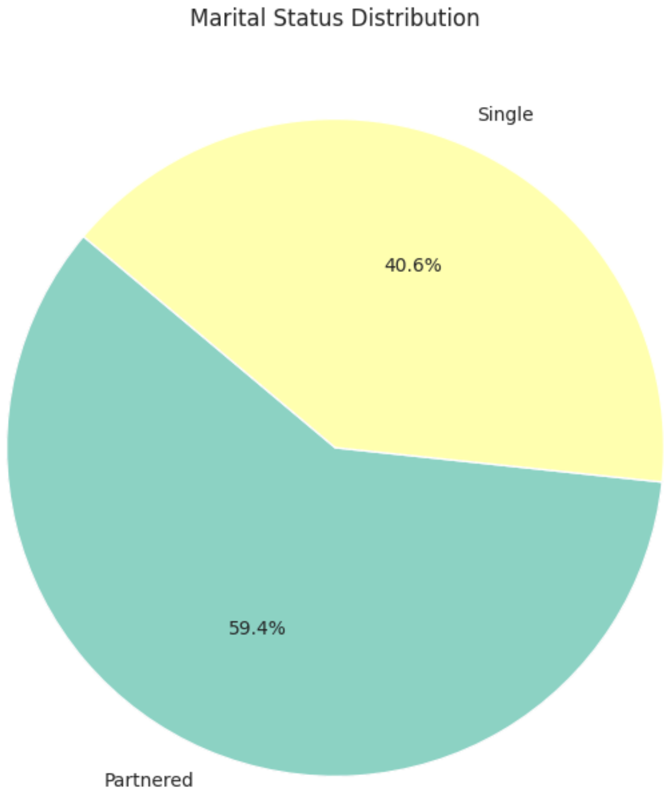
Gender Distribution: Out of 180 data points, 104 are males, and the rest are females.

Income and Miles: The standard deviation for Income and Miles is very high, suggesting the presence of outliers in these variables.

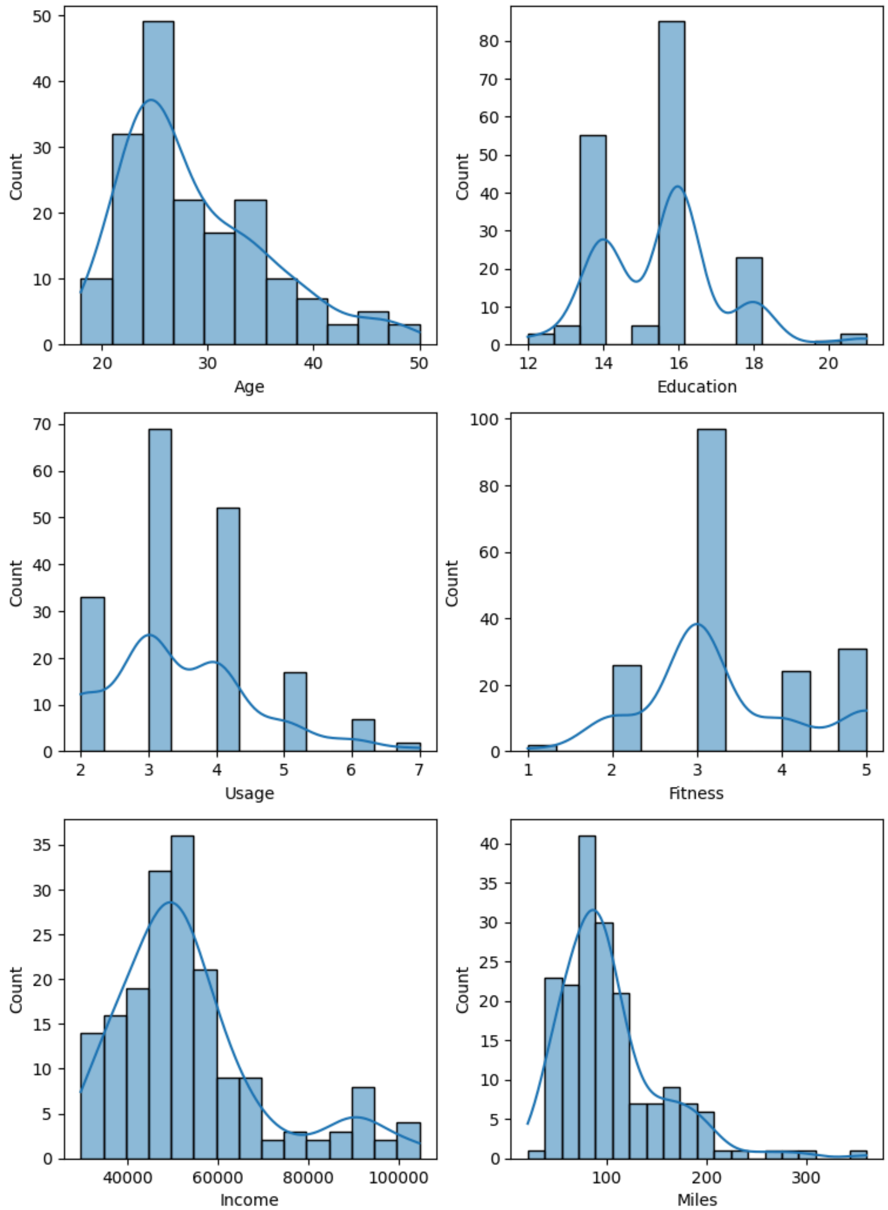
```
plt.figure(figsize=(8, 8))
product_counts = df['Product'].value_counts()
plt.pie(product_counts, labels=product_counts.index, autopct='%1.1f%%', startangle=140, colors=sns.color_palette("husl", len(product_counts)
plt.title('Product Distribution')
plt.show()
```




```
plt.figure(figsize=(8, 8))
marital_status_counts = df['MaritalStatus'].value_counts()
plt.pie(marital_status_counts, labels=marital_status_counts.index, autopct='%1.1f%%', startangle=140, colors=sns.color_palette("Set3", len(m
plt.title('Marital Status Distribution')
plt.show()
```




```
fig, axis = plt.subplots(nrows=3, ncols=2, figsize=(9, 9))
fig.subplots_adjust(top=1.2)
sns.histplot(data=df, x="Age", kde=True, ax=axis[0,0])
sns.histplot(data=df, x="Education", kde=True, ax=axis[0,1])
sns.histplot(data=df, x="Usage", kde=True, ax=axis[1,0])
sns.histplot(data=df, x="Fitness", kde=True, ax=axis[1,1])
sns.histplot(data=df, x="Income", kde=True, ax=axis[2,0])
sns.histplot(data=df, x="Miles", kde=True, ax=axis[2,1])
plt.show()
```



Observations:

Age: The majority of customers are in the 20-30 age range. There is a significant drop-off in the number of customers over the age of 35.

Education: Most customers have 16 years of education, suggesting a high level of education. There's a noticeable drop-off in education levels above 16 years.

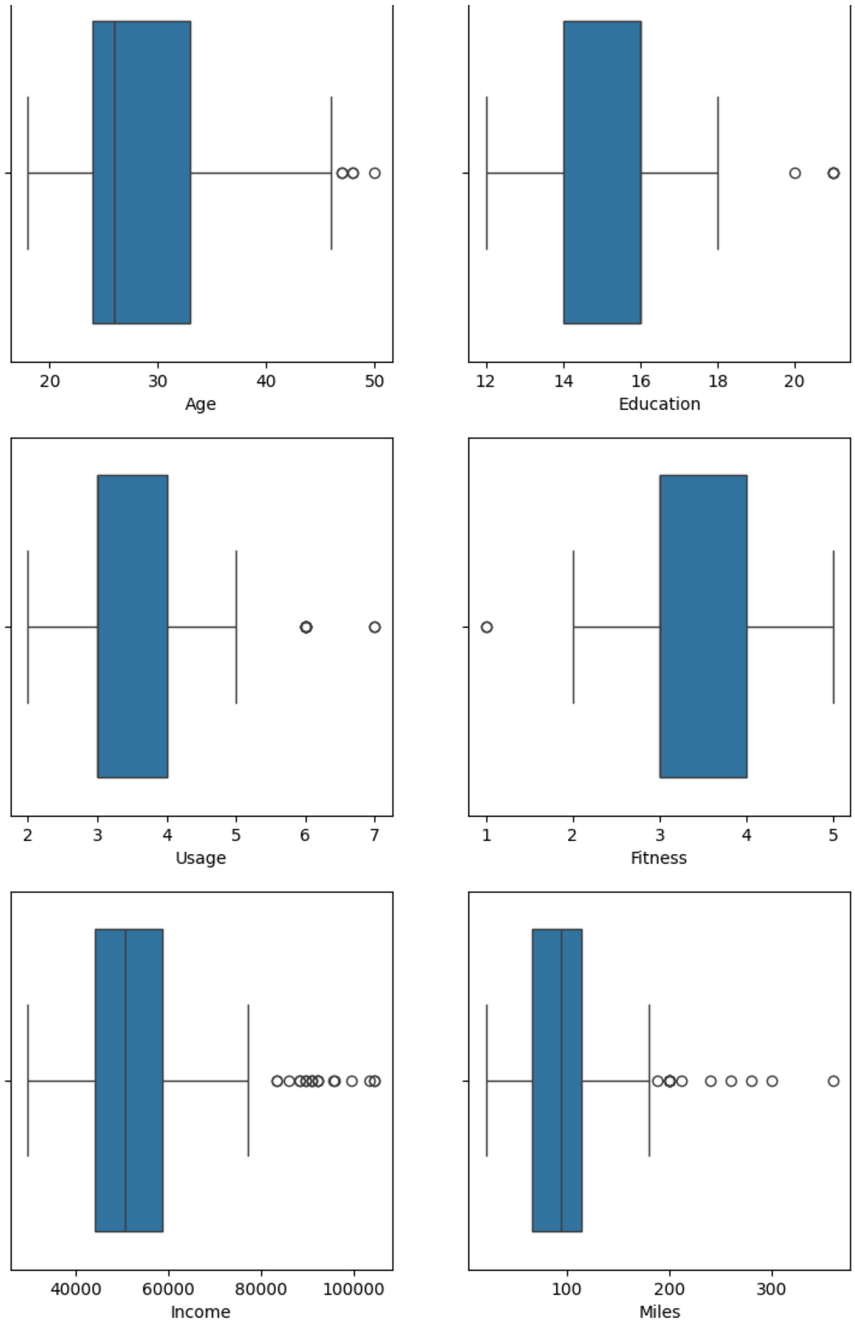
Usage: The usage frequency of the product is mostly concentrated around 3-4 times per week. Few customers use the product more than 5 times a week.

Fitness: Most customers rated their fitness level as 3 out of 5. There are fewer customers with fitness levels of 1, 2, 4, and 5.

Income: The income distribution shows a peak around the $40,000 to$60,000 range. There are outliers with incomes above $80,000, but they are less common.

Miles: Most customers log around 100-150 miles. There is a decrease in customers logging more than 200 miles.

```
fig, axis = plt.subplots(nrows=3, ncols=2, figsize=(9, 12))
fig.subplots_adjust(top=1.0)
sns.boxplot(data=df, x="Age", orient='h', ax=axis[0,0])
sns.boxplot(data=df, x="Education", orient='h', ax=axis[0,1])
sns.boxplot(data=df, x="Usage", orient='h', ax=axis[1,0])
sns.boxplot(data=df, x="Fitness", orient='h', ax=axis[1,1])
sns.boxplot(data=df, x="Income", orient='h', ax=axis[2,0])
sns.boxplot(data=df, x="Miles", orient='h', ax=axis[2,1])
plt.show()
```



Age:

The median age is around 30. The interquartile range (IQR) is roughly between 25 and 35. There are a few outliers above 45.

Education:

The median education level is around 16 years. The IQR is between approximately 14 and 17 years. There are outliers at 20 and above.

Usage:

The median usage is around 3 times. The IQR ranges between 2 and 4. There are a few outliers above 6.

Fitness:

The median fitness level is around 3. The IQR spans from 2 to 4. There are some lower outliers around 1.

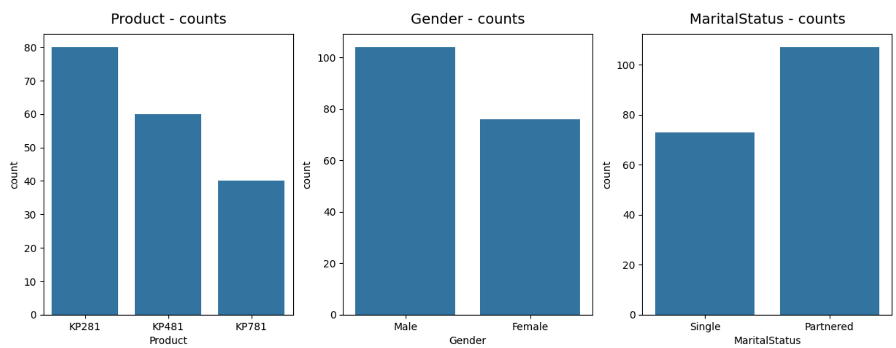
Income:

The median income is around $60,000.TheIQRisapproximatelybetween$50,000 and $70,000.Thereareseveraloutliersabove$80,000.

Miles:

The median miles are around 100. The IQR ranges from about 75 to 150 miles. There are several outliers above 200 miles. These box plots help in understanding the distribution, central tendency, and variability of the data for each variable.

```
fig, axs = plt.subplots(nrows=1, ncols=3, figsize=(15,5))
sns.countplot(data=df, x='Product', ax=axs[0])
sns.countplot(data=df, x='Gender', ax=axs[1])
sns.countplot(data=df, x='MaritalStatus', ax=axs[2])
axs[0].set_title("Product - counts", pad=10, fontsize=14)
axs[1].set_title("Gender - counts", pad=10, fontsize=14)
axs[2].set_title("MaritalStatus - counts", pad=10,
fontsize=14)
plt.show()
```



Product - counts:

KP281 has the highest count, approaching 80. KP481 has the second-highest count, around 60. KP781 has the lowest count, slightly above 40.

Gender - counts:

There are more males than females in the dataset. Male count is slightly above 100. Female count is slightly below 80.

MaritalStatus - counts:

The number of partnered individuals is higher than single individuals. Partnered count is slightly above 100. Single count is slightly below 80.

```
df1 = df[['Product', 'Gender', 'MaritalStatus']].melt()
df1.groupby(['variable', 'value'])[['value']].count() / len(df)
```

variable	value	value
Gender	Female	0.422222
	Male	0.577778
MaritalStatus	Partnered	0.594444
	Single	0.405556
Product	KP281	0.444444
	KP481	0.333333
	KP781	0.222222

Observations

Product 44.44% of the customers have purchased KP2821 product. 33.33% of the customers have purchased KP481 product. 22.22% of the customers have purchased KP781 product.

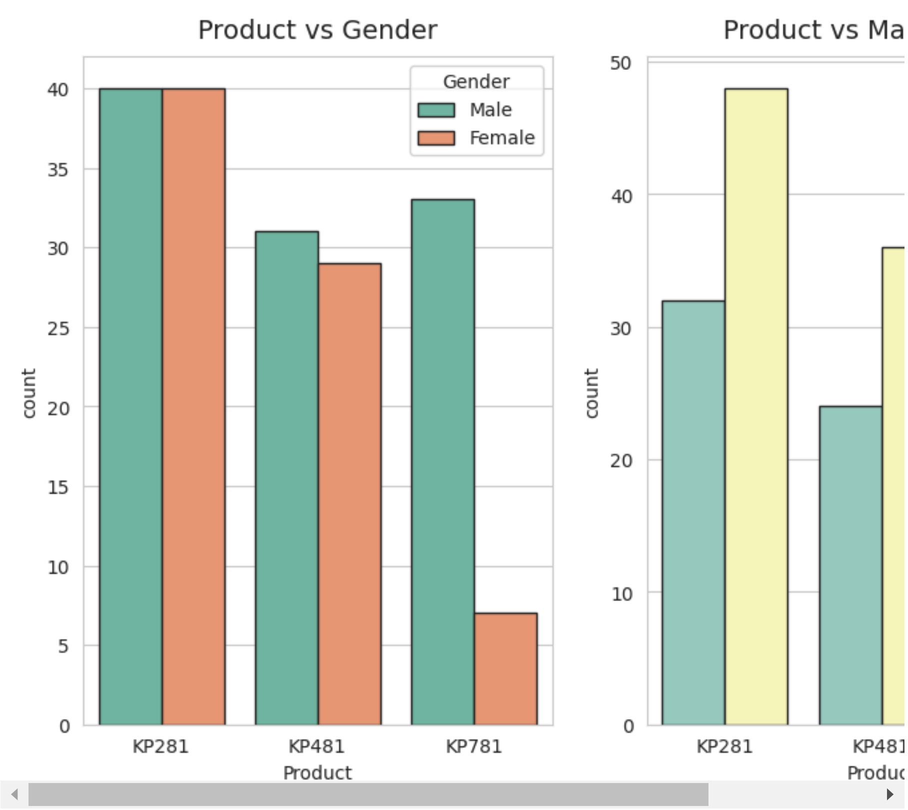
Gender

57.78% of the customers are Male.

MaritalStatus

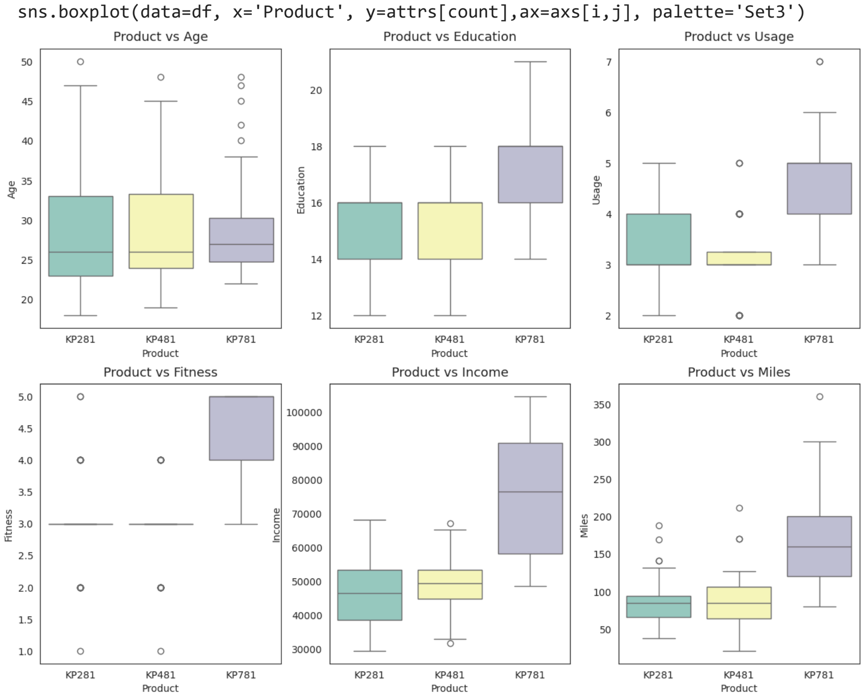
59.44% of the customers are Partnered.

```
sns.set_style(style='whitegrid')
fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(10, 6.5))
sns.countplot(data=df, x='Product', hue='Gender', edgecolor="0.15",palette='Set2', ax=axs[0])
sns.countplot(data=df, x='Product', hue='MaritalStatus',edgecolor="0.15", palette='Set3', ax=axs[1])
axs[0].set_title("Product vs Gender", pad=10, fontsize=14)
axs[1].set_title("Product vs MaritalStatus", pad=10, fontsize=14)
plt.show()
```



```
attrs = ['Age', 'Education', 'Usage', 'Fitness', 'Income','Miles']
sns.set_style("white")
fig, axs = plt.subplots(nrows=2, ncols=3, figsize=(15, 8))
fig.subplots_adjust(top=1.2)
count = 0
for i in range(2):
  for j in range(3):
    sns.boxplot(data=df, x='Product', y=attrs[count],ax=axs[i,j], palette='Set3')
    axs[i,j].set_title(f"Product vs {attrs[count]}",pad=8, fontsize=13)
    count += 1
```

```
  sns.boxplot(data=df, x='Product', y=attrs[count],ax=axs[i,j], palette='Set3')
<ipython-input-26-6a4436c601d6>:8: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.

  sns.boxplot(data=df, x='Product', y=attrs[count],ax=axs[i,j], palette='Set3')
<ipython-input-26-6a4436c601d6>:8: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.

  sns.boxplot(data=df, x='Product', y=attrs[count],ax=axs[i,j], palette='Set3')
<ipython-input-26-6a4436c601d6>:8: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.

  sns.boxplot(data=df, x='Product', y=attrs[count],ax=axs[i,j], palette='Set3')
<ipython-input-26-6a4436c601d6>:8: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.

  sns.boxplot(data=df, x='Product', y=attrs[count],ax=axs[i,j], palette='Set3')
<ipython-input-26-6a4436c601d6>:8: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.

  sns.boxplot(data=df, x='Product', y=attrs[count],ax=axs[i,j], palette='Set3')
```



Product vs Age:

KP281: The median age is around 25, with a range roughly from 20 to 45.

KP481: The median age is slightly above 25, with a range from about 20 to 40.

KP781: The median age is around 25, with a wider range up to 45, and several outliers around 50.

Product vs Education:

KP281: The median education level is around 16 years, with a range from about 13 to 18 years.

KP481: The median education level is slightly above 14 years, with a narrower range from about 12 to 16 years.

KP781: The median education level is around 18 years, with a range from about 14 to 20 years.

Product vs Usage:

KP281: The median usage is around 3, with a range from about 2 to 6.

KP481: The median usage is around 3, with a very narrow range and some outliers around 6.

KP781: The median usage is around 5, with a range from about 3 to 7 and a few outliers.

Product vs Fitness:

KP281 and KP481: The fitness levels are relatively constant around 3, with a few outliers.

KP781: The fitness level has a median around 4.5, with a range from about 3 to 5.

Product vs Income:

KP281: The median income is around $50, 000, witharangefromabout$30,000 to \$70,000.

KP481: The median income is around $50, 000, witharangefromabout$30,000 to \$70,000.

KP781: The median income is around $90, 000, witharangefromabout$70,000 to above \$100,000.

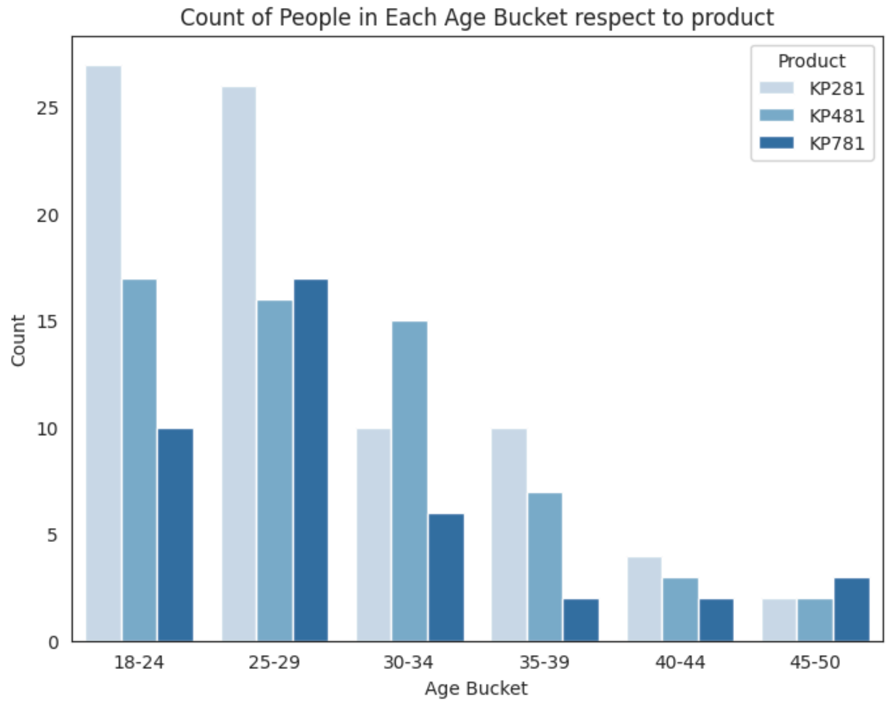
Product vs Miles:

KP281: The median miles is around 100, with a range from about 50 to 150.

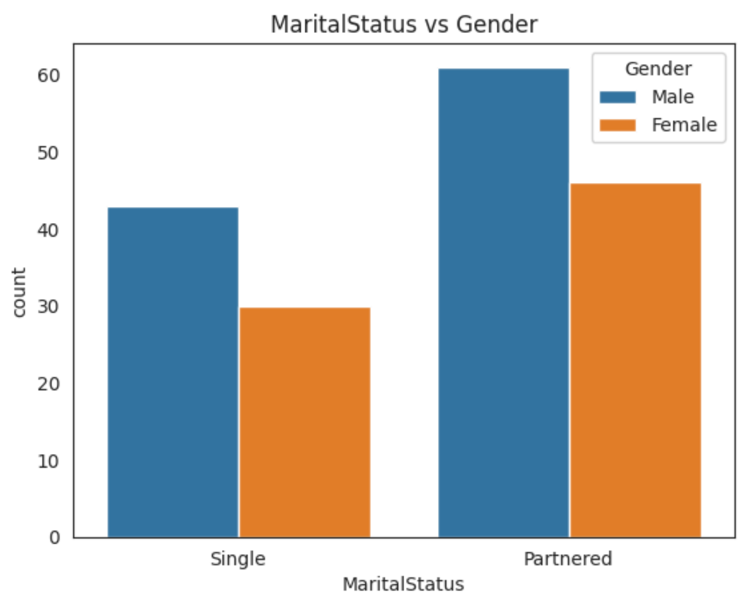
KP481: The median miles is around 100, with a range from about 50 to 150.

KP781: The median miles is around 150, with a range from about 100 to 250 and some outliers up to 350.

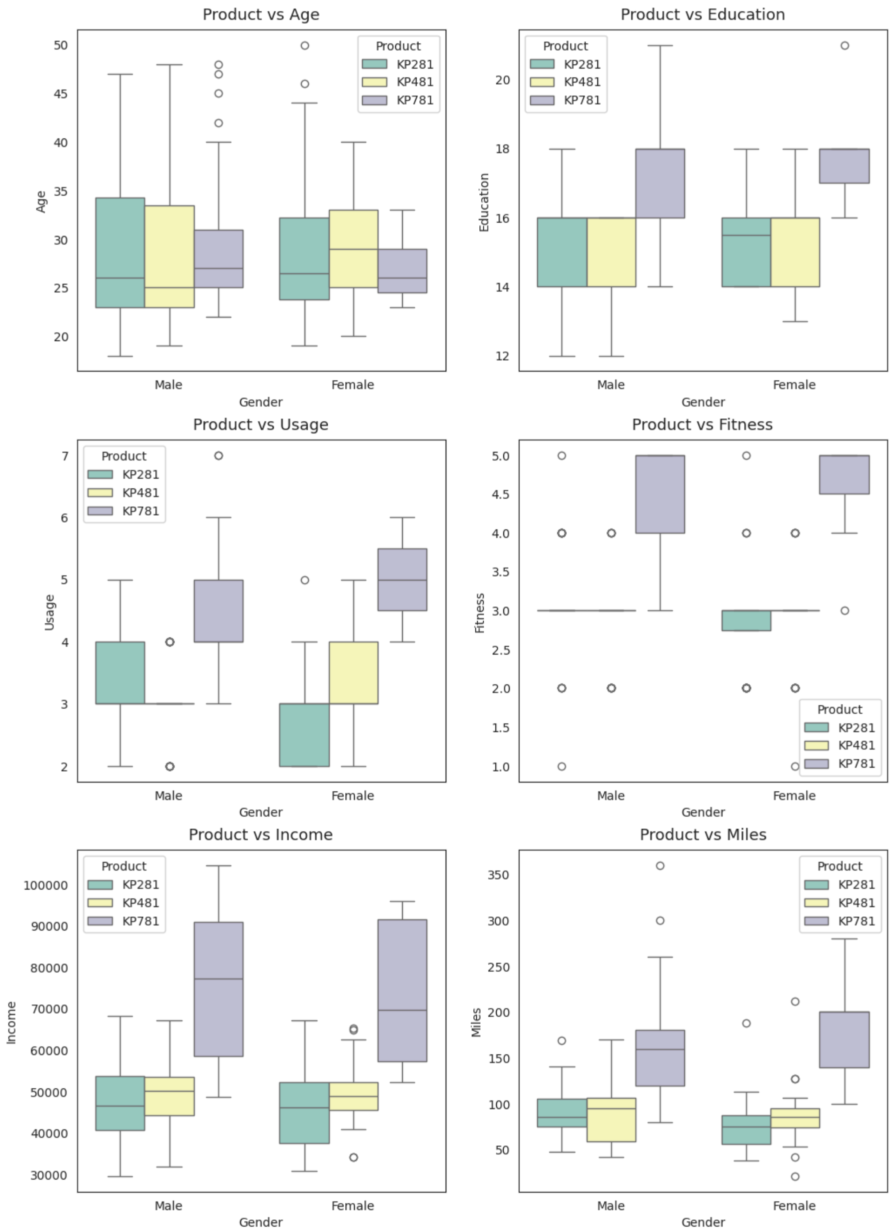
```
age_bins = [18, 25, 30, 35, 40, 45, 50]  # Define your desired age bins here
age_labels = ['18-24', '25-29', '30-34', '35-39', '40-44', '45-50']  # Corrected labels for the bins


# Create a new column 'AgeBucket' with the age bins
df['AgeBucket'] = pd.cut(df['Age'], bins=age_bins, labels=age_labels, right=False)

# Create a count plot based on the 'AgeBucket' column
plt.figure(figsize=(8, 6))  # Adjust the figure size as needed
sns.countplot(data=df, x='AgeBucket', palette='Blues',hue="Product")
# Set labels and title
plt.xlabel('Age Bucket')
plt.ylabel('Count')
plt.title('Count of People in Each Age Bucket respect to product')

# Show the plot
#plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.show()
```




```
sns.countplot(data=df, x='MaritalStatus', hue='Gender')
plt.title("MaritalStatus vs Gender")
plt.show()
```




```
attrs = ['Age', 'Education', 'Usage', 'Fitness', 'Income', 'Miles']
sns.set_style("white")
fig, axs = plt.subplots(nrows=3, ncols=2, figsize=(12, 15))
fig.subplots_adjust(top=1)
count = 0
for i in range(3):
 for j in range(2):
  sns.boxplot(data=df, x='Gender', y=attrs[count], hue='Product',ax=axs[i,j], palette='Set3')
  axs[i,j].set_title(f"Product vs {attrs[count]}", pad=8,fontsize=13)
  count += 1
```


Product vs Age
Age
Gender
Male
Female
Product
KP281
KP481
KP781
Product vs Education
Education
Product vs Usage
Usage
Product vs Fitness
Fitness
Product vs Income
Income
Product vs Miles
Miles

```
df['Product'].value_counts(normalize=True)
```

```
Product
KP281    0.444444
KP481    0.333333
KP781    0.222222
Name: proportion, dtype: float64
```

```
def p_prod_given_gender(gender, print_marginal=False):
  if gender is not "Female" and gender is not "Male":
    return "Invalid gender value."

  df1 = pd.crosstab(index=df['Gender'], columns=[df['Product']])
  p_781 = df1['KP781'][gender] / df1.loc[gender].sum()
  p_481 = df1['KP481'][gender] / df1.loc[gender].sum()
  p_281 = df1['KP281'][gender] / df1.loc[gender].sum()

  if print_marginal:
    print(f"P(Male): {df1.loc['Male'].sum()/len(df):.2f}")
    print(f"P(Female): {df1.loc['Female'].sum()/len(df):.2f}\n")

  print(f"P(KP781/{gender}): {p_781:.2f}")
  print(f"P(KP481/{gender}): {p_481:.2f}")
  print(f"P(KP281/{gender}): {p_281:.2f}\n")

p_prod_given_gender('Male', True)
p_prod_given_gender('Female')
```

```
P(Male): 0.58
P(Female): 0.42

P(KP781/Male): 0.32
P(KP481/Male): 0.30
P(KP281/Male): 0.38

P(KP781/Female): 0.09
P(KP481/Female): 0.38
P(KP281/Female): 0.53

<>:2: SyntaxWarning: "is not" with a literal. Did you mean "!="?
<>:2: SyntaxWarning: "is not" with a literal. Did you mean "!="?
<>:2: SyntaxWarning: "is not" with a literal. Did you mean "!="?
<>:2: SyntaxWarning: "is not" with a literal. Did you mean "!="?
<ipython-input-33-374c375a6b1f>:2: SyntaxWarning: "is not" with a literal. Did you mean "!="?
  if gender is not "Female" and gender is not "Male":
<ipython-input-33-374c375a6b1f>:2: SyntaxWarning: "is not" with a literal. Did you mean "!="?
  if gender is not "Female" and gender is not "Male":
```

```
def p_prod_given_mstatus(status, print_marginal=False):
    if status is not "Single" and status is not "Partnered":
        return "Invalid marital status value."

    df1 = pd.crosstab(index=df['MaritalStatus'], columns=[df['Product']])
    p_781 = df1['KP781'][status] / df1.loc[status].sum()
    p_481 = df1['KP481'][status] / df1.loc[status].sum()
    p_281 = df1['KP281'][status] / df1.loc[status].sum()

    if print_marginal:
        print(f"P(Single): {df1.loc['Single'].sum()/len(df):.2f}")
        print(f"P(Partnered): {df1.loc['Partnered'].sum()/len(df):.2f}\n")

    print(f"P(KP781/{status}): {p_781:.2f}")
    print(f"P(KP481/{status}): {p_481:.2f}")
    print(f"P(KP281/{status}): {p_281:.2f}\n")

p_prod_given_mstatus('Single', True)
p_prod_given_mstatus('Partnered')
```

```
P(Single): 0.41
P(Partnered): 0.59

P(KP781/Single): 0.23
P(KP481/Single): 0.33
P(KP281/Single): 0.44

P(KP781/Partnered): 0.21
P(KP481/Partnered): 0.34
P(KP281/Partnered): 0.45
```

```
<>:2: SyntaxWarning: "is not" with a literal. Did you mean "!="?
<>:2: SyntaxWarning: "is not" with a literal. Did you mean "!="?
<>:2: SyntaxWarning: "is not" with a literal. Did you mean "!="?
<>:2: SyntaxWarning: "is not" with a literal. Did you mean "!="?
<ipython-input-34-6c728dc64dfa>:2: SyntaxWarning: "is not" with a literal. Did you mean "!="?
  if status is not "Single" and status is not "Partnered":
<ipython-input-34-6c728dc64dfa>:2: SyntaxWarning: "is not" with a literal. Did you mean "!="?
  if status is not "Single" and status is not "Partnered":
```

```
plt.figure(figsize=(10, 8))
correlation_matrix = df.corr(numeric_only=True)
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Matrix Heatmap')
plt.show()
```

Age:

Positively correlated with Income (0.51).

Moderately correlated with Education (0.28).

Education:

Positively correlated with Income (0.63).

Moderately correlated with Usage (0.4), Fitness (0.41), and Miles (0.31).

Usage:

Strongly correlated with Fitness (0.67) and Miles (0.76).

Moderately correlated with Income (0.52).

Fitness:

Strongly correlated with Usage (0.67) and Miles (0.79).

Moderately correlated with Income (0.54).

Income:

Positively correlated with Education (0.63), Usage (0.52), and Fitness (0.54).

Miles:

Strongly correlated with Usage (0.76) and Fitness (0.79).

Moderately correlated with Income (0.54).

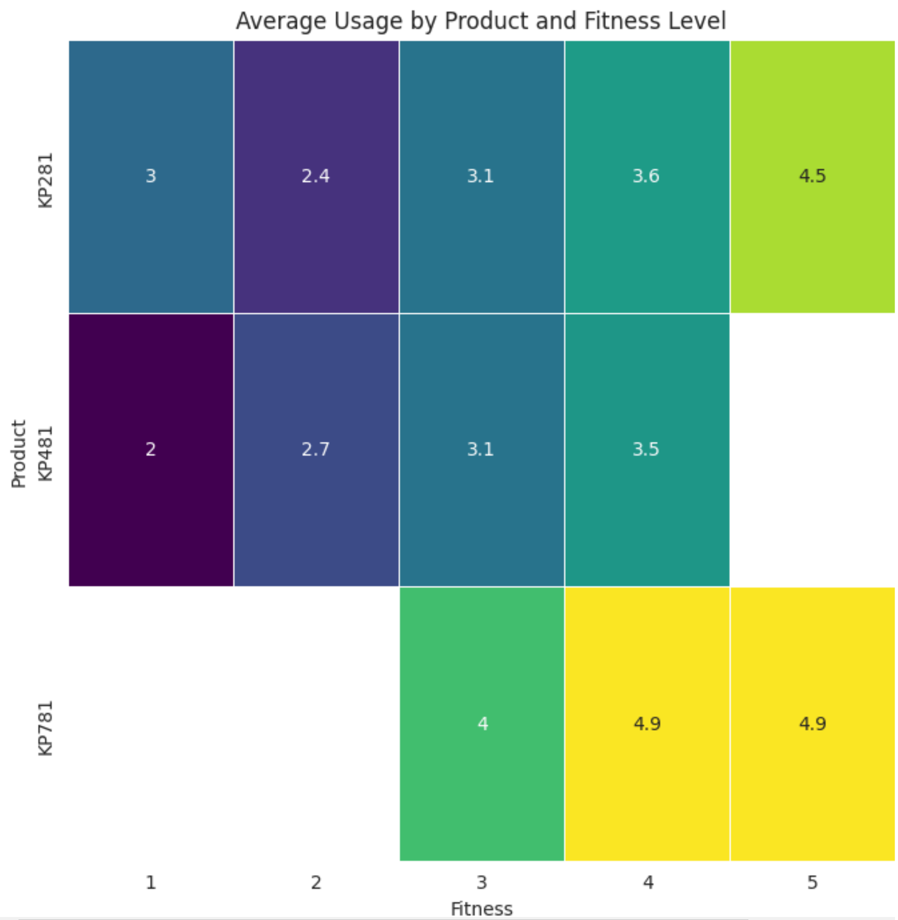
The heatmap highlights the strong positive correlations between:

Usage and Fitness Usage and Miles Fitness and Miles

It also shows moderate positive correlations between:

Age and Income Education and Income Usage and Income Fitness and Income Education and Fitness

```
# Creating a heatmap for Usage vs Fitness grouped by Product
plt.figure(figsize=(10, 8))
pivot_table = df.pivot_table(values='Usage', index='Product', columns='Fitness', aggfunc='mean')
sns.heatmap(pivot_table, annot=True, cmap='viridis', linewidths=0.5)
plt.title('Average Usage by Product and Fitness Level')
plt.show()
```



The heatmap visualizes the average usage of three products (KP281, KP481, and KP781) across different fitness levels (1 to 5). Here are some insights based on the provided heatmap:

KP281:

Shows an increasing trend in usage with higher fitness levels. Starts at an average usage of 3 at fitness level 1 and goes up to 4.5 at fitness level 5.

KP481:

Exhibits a more moderate increase in usage compared to KP281. Usage ranges from 2 at fitness level 1 to 3.5 at fitness level 5.

KP781:

Demonstrates the highest average usage across all fitness levels. Starts at 4 at fitness level 3 and remains consistent at 4.9 for fitness levels 4 and 5.

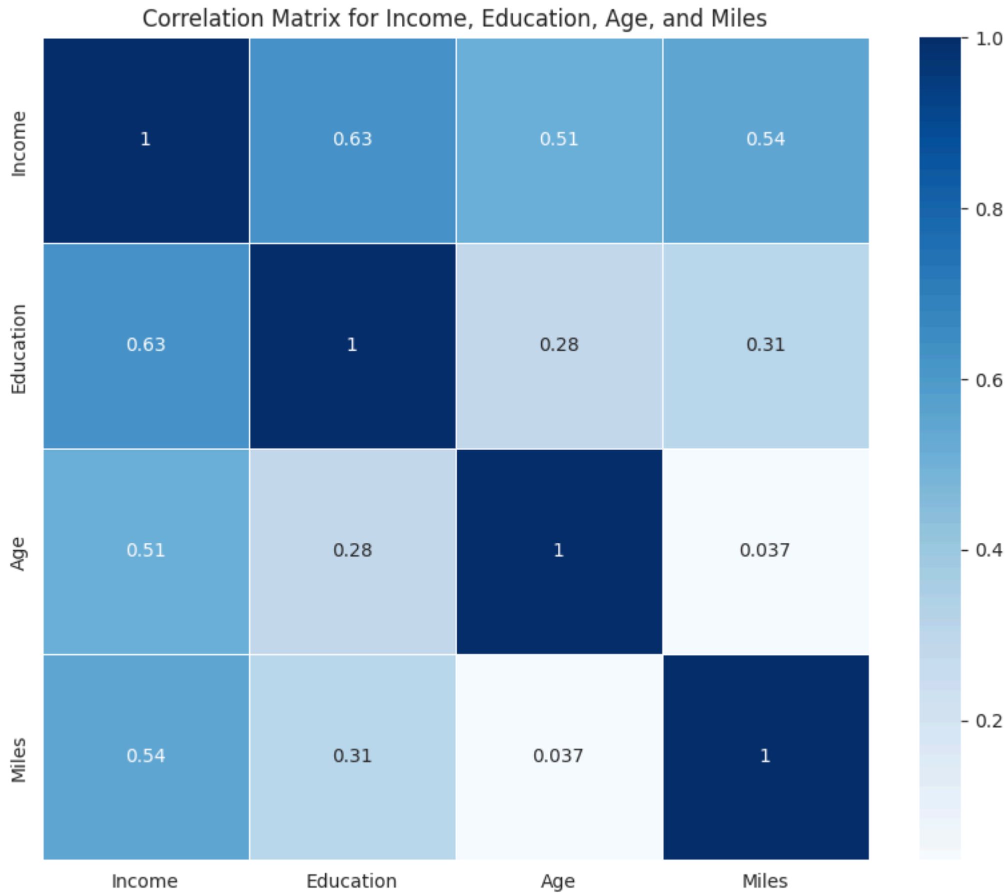
Fitness Levels:

Fitness levels 4 and 5 show the highest average usage across all products, indicating that individuals with higher fitness levels tend to use these products more frequently. Lower fitness levels (1 and 2) generally show lower average usage for all products.

Color Gradient:

The color gradient (from dark purple to bright yellow) effectively highlights the variation in average usage, with darker colors representing lower usage and brighter colors representing higher usage.

```
plt.figure(figsize=(10, 8))
selected_columns = df[['Income', 'Education', 'Age', 'Miles']]
correlation_matrix = selected_columns.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='Blues', linewidths=0.5)
plt.title('Correlation Matrix for Income, Education, Age, and Miles')
plt.show()
```



The correlation matrix visualizes the relationships between four variables: Income, Education, Age, and Miles. Here's an analysis of the insights derived from the matrix:

Income:

Positively correlated with Education (0.63): Higher education levels are associated with higher income. Moderately correlated with Age (0.51): Older individuals tend to have higher income, possibly due to career progression over time. Moderately correlated with Miles (0.54): Higher income individuals tend to travel more miles.

Education:

Positively correlated with Income (0.63): Higher education levels correspond to higher income. Weak correlation with Age (0.28): There is a slight tendency for higher education levels among older individuals. Weak correlation with Miles (0.31): Individuals with higher education levels tend to travel slightly more miles.

Age:

Moderately correlated with Income (0.51): Older individuals tend to have higher income. Weak correlation with Education (0.28): Slight tendency for higher education levels among older individuals. Very weak correlation with Miles (0.037): Age does not significantly affect the number of miles traveled.