Business Case : Aerofit Descriptive Statistics & Probability

About Aerofit:

Aerofi t 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.

INTRODUCTION:

AeroFit is popular for its reliable performance, easy maintenance, and features that support different fitness levels — from beginners to advanced users. AeroFit products often include various workout modes, heart-rate monitoring, and comfort-focused designs to help users achieve their fitness goals effectively. The company aims to promote a healthy lifestyle by making fitness accessible to everyone through safe, efficient, and innovative equipment.

PROBLEM STATEMENT:

AeroFit wants to understand the purchasing behavior of its customers to improve product recommendations, marketing strategies, and overall sales. Although the company offers multiple treadmill models with different features and price ranges, customers often choose products without clear patterns. This makes it difficult for AeroFit to target the right customer groups and design products that meet their needs. The problem is to analyze customer demographics (such as age, gender, income, fitness level, etc.) and identify the factors that influence the choice of a particular AeroFit treadmill model. By understanding these patterns, AeroFit can make better business decisions, create focused marketing campaigns, and improve customer satisfaction.

Analysis the Basic Metrics:

In [ ]:
```
# Importing  the libraries.

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

In [ ]:
```
# Load the Dataset.

df=pd.read_csv("/content/aerofit_treadmill.txt")
```

In [ ]:
```
# To Check the Dataset.

df
```

Out[ ]:

	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
...	...	...	...	...	...	...	...	...	...
175	KP781	40	Male	21	Single	6	5	83416	200
176	KP781	42	Male	18	Single	5	4	89641	200
177	KP781	45	Male	16	Single	5	5	90886	160
178	KP781	47	Male	18	Partnered	4	5	104581	120
179	KP781	48	Male	18	Partnered	4	5	95508	180

180 rows × 9 columns

In [ ]:
```
# To Check First Five Data.

df.head()
```

Out[ ]:

	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

In [ ]:
```
# To Check the Shape of Data - It means how many rows and columns are present in data.

df.shape
```

Out[ ]:
```
(180, 9)
```

In [ ]:
```
# To Check the Statistics.

df.describe().T
```

Out[ ]:

	count	mean	std	min	25%	50%	75%	max
Age	180.0	28.788889	6.943498	18.0	24.00	26.0	33.00	50.0
Education	180.0	15.572222	1.617055	12.0	14.00	16.0	16.00	21.0
Usage	180.0	3.455556	1.084797	2.0	3.00	3.0	4.00	7.0
Fitness	180.0	3.311111	0.958869	1.0	3.00	3.0	4.00	5.0
Income	180.0	53719.577778	16506.684226	29562.0	44058.75	50596.5	58668.00	104581.0
Miles	180.0	103.194444	51.863605	21.0	66.00	94.0	114.75	360.0

In [ ]:
```
#  To Check the Data Types of all the attributes.

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 Cleaning:

In [ ]:
```
# To Check the Missing Values.

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

Out[ ]:

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

dtype: int64

Non-Graphical Analysis:

- Non-graphical analysis helps in understanding the basic structure and distribution of data without using visualizations. We can perform this analysis using methods such as value_counts() and nunique() in Python (Pandas).

Value Counts:

- **Product:**

In [ ]:
```
# Identifies the most and least preferred product.

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

Out[ ]:

Product	count
KP281	80
KP481	60
KP781	40

dtype: int64

- **Gender:**

In [ ]:
```
# Shows count of Male and Female customers.

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

Out[ ]:

Gender	count
Male	104
Female	76

dtype: int64

- **Marital Status:**

In [ ]:
```
# Shows distribution of Single vs Married customers.

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

Out[ ]:

	count
MaritalStatus	
Partnered	107
Single	73

dtype: int64

- **Fitness:**

In [ ]:
```
# Shows how customers rate their fitness level (1–5 scale).

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

Out[ ]:

	count
Fitness	
3	97
5	31
2	26
4	24
1	2

dtype: int64

- The **value_counts()** function helps identify the most frequent occurrences of categorical data.

Unique Attributes:

- **Age:**

In [ ]:
```
# Tells how diverse the age distribution is:

df['Age'].nunique()
```

Out[ ]: 32

- **Education:**

In [ ]:
```
# Number of distinct education levels in the dataset.

df['Education'].nunique()
```

Out[ ]: 8

- **Miles:**

In [ ]:
```
# Number of distinct weekly miles values.

df['Miles'].nunique()
```

Out[ ]: 37

- **Usage:**

In [ ]:
```
# Number of distinct usage frequency levels.

df['Usage'].nunique()
```

Out[ ]: 6

- **Income:**

In [ ]:
```
# Number of distinct income values.

df['Income'].nunique()
```

Out[ ]: 62

- The **nunique()** function tells how many unique values each column contains.

Visual Analysis:

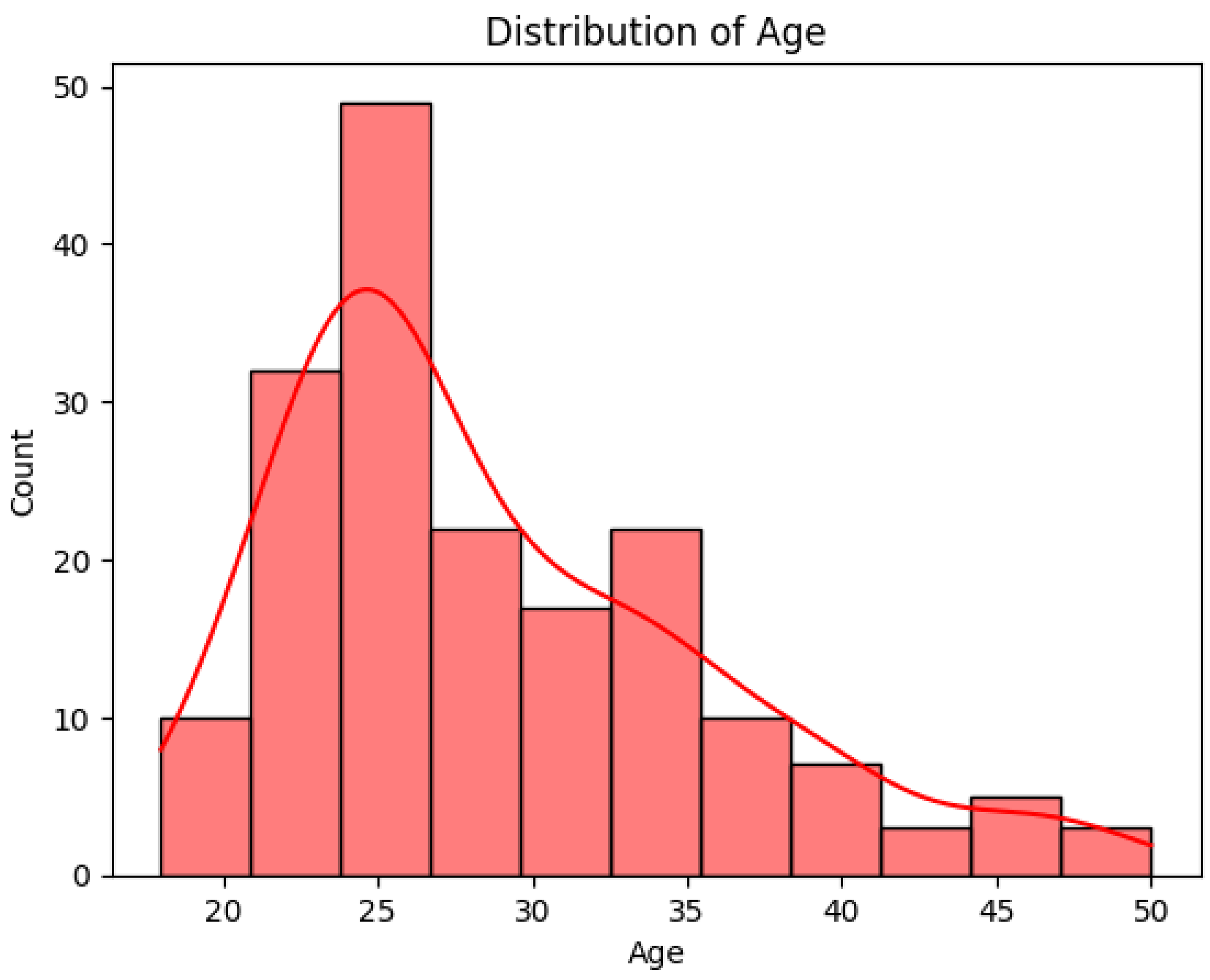
Univariate Analysis:

- Univariate = analyzing one variable at a time.

(a) For Continuous Variables.

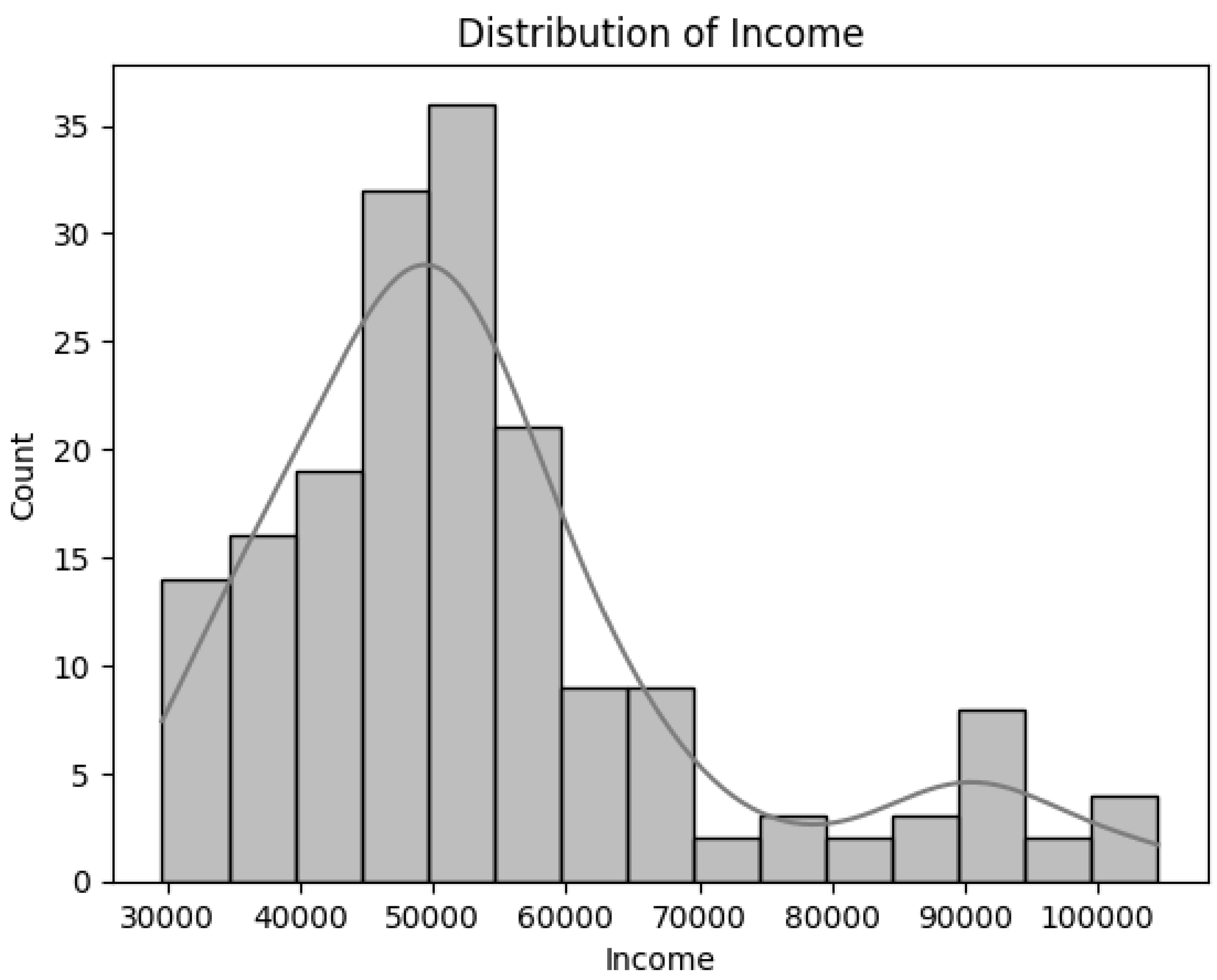
- **Age:**

In [ ]:
```
sns.histplot(df["Age"], kde=True, color='red')
plt.title("Distribution of Age")
plt.show()
```



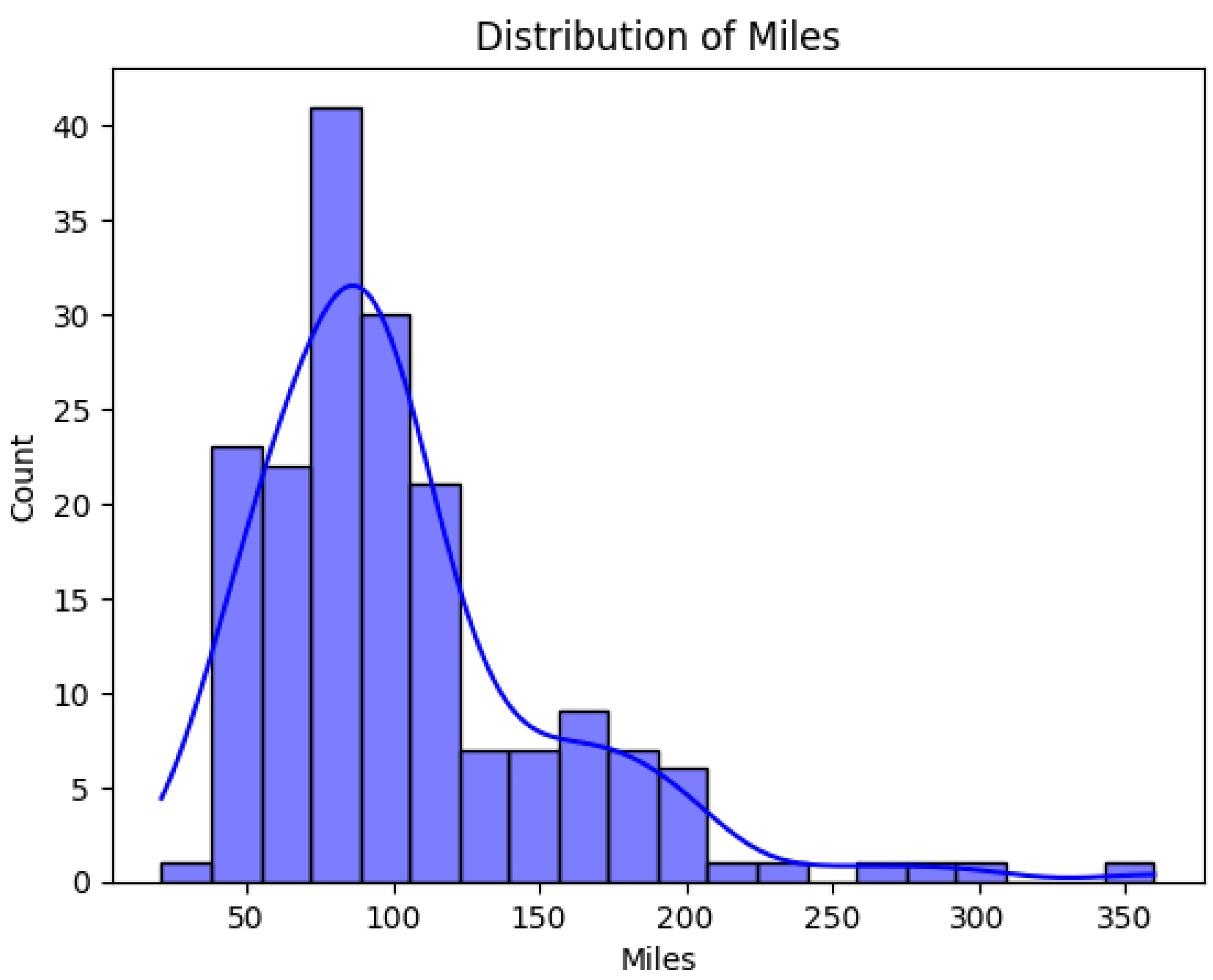
- **Income:**

In [ ]:
```
sns.histplot(df["Income"], kde=True,color='grey')
plt.title("Distribution of Income")
plt.show()
```



- **Miles:**

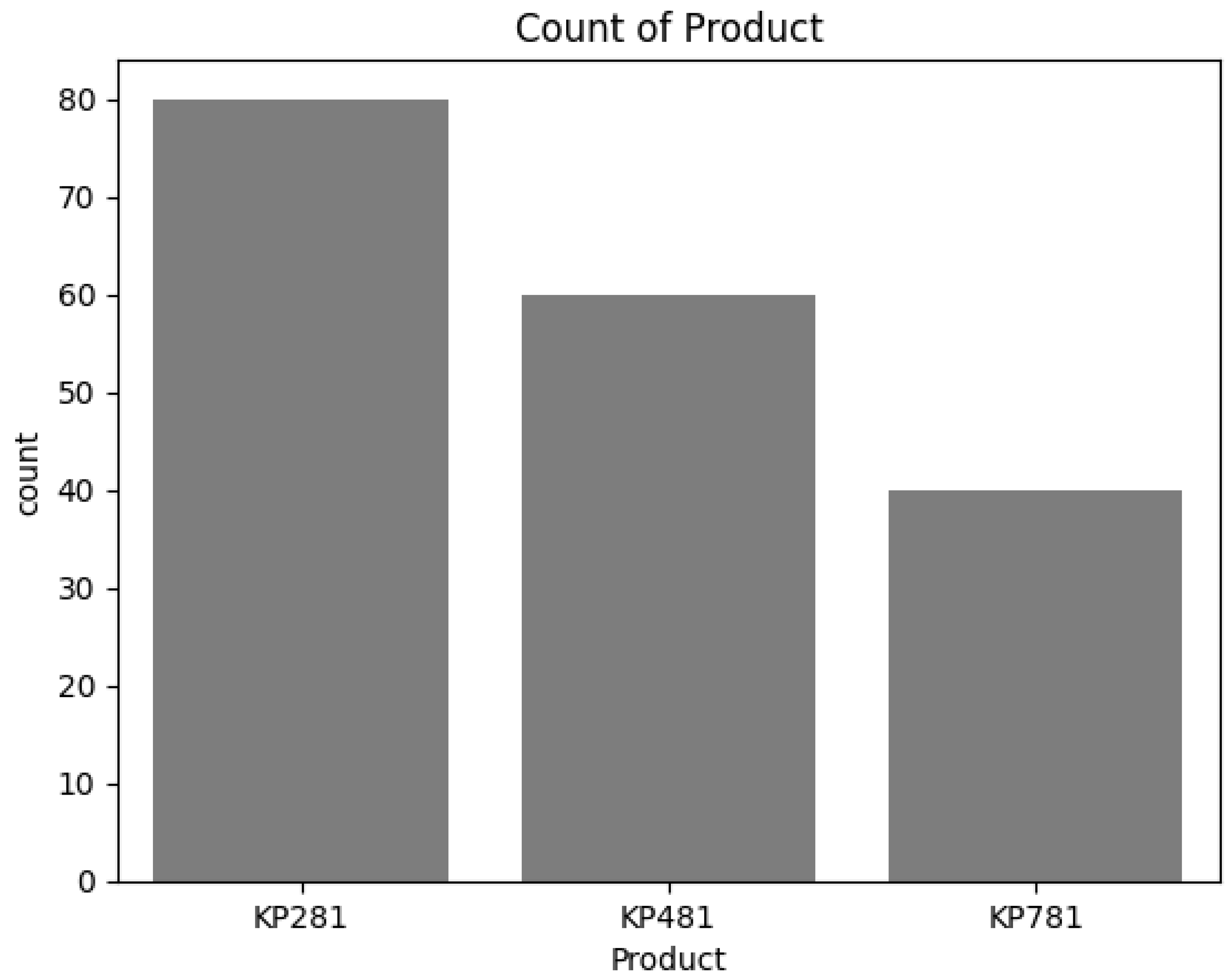
In [ ]:
```
sns.histplot(df["Miles"],kde=True,color='blue')
plt.title("Distribution of Miles")
plt.show()
```



(b) For Categorial Variables.

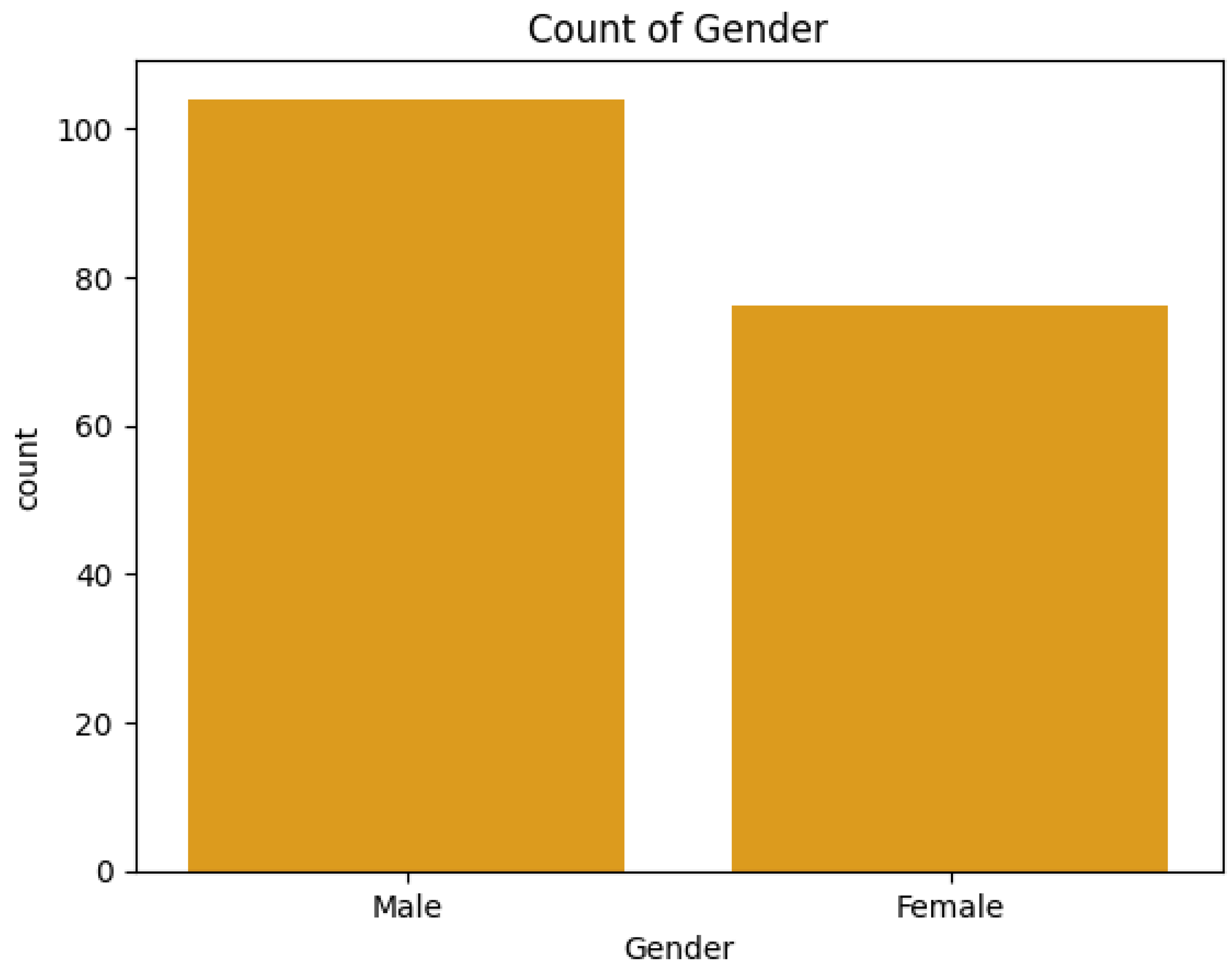
- **Product:**

In [ ]:
```
sns.countplot(data=df, x="Product", color='grey')
plt.title("Count of Product")
plt.show()
```



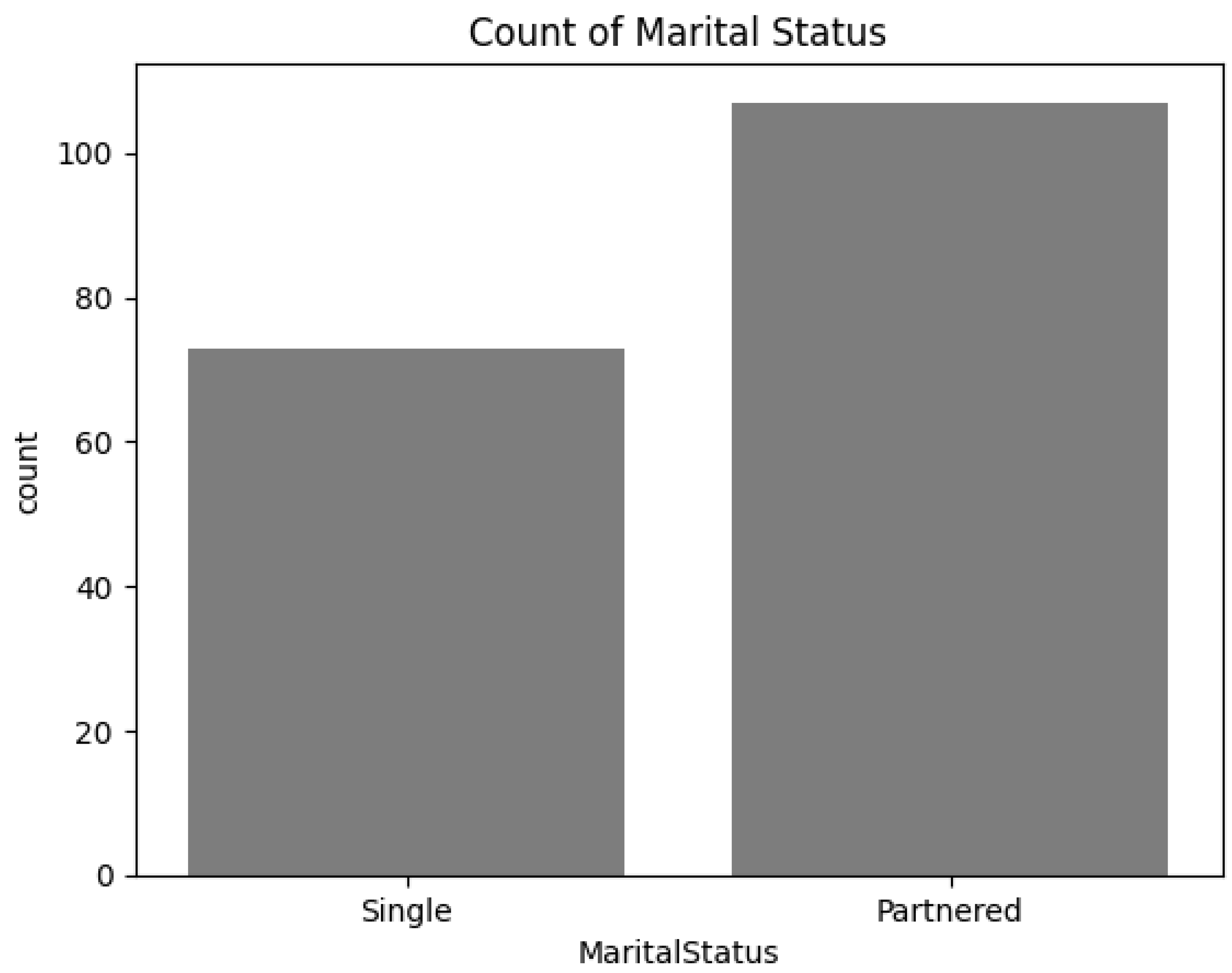
- **Gender:**

In [ ]:
```
sns.countplot(data=df, x="Gender", color='orange')
plt.title("Count of Gender")
plt.show()
```



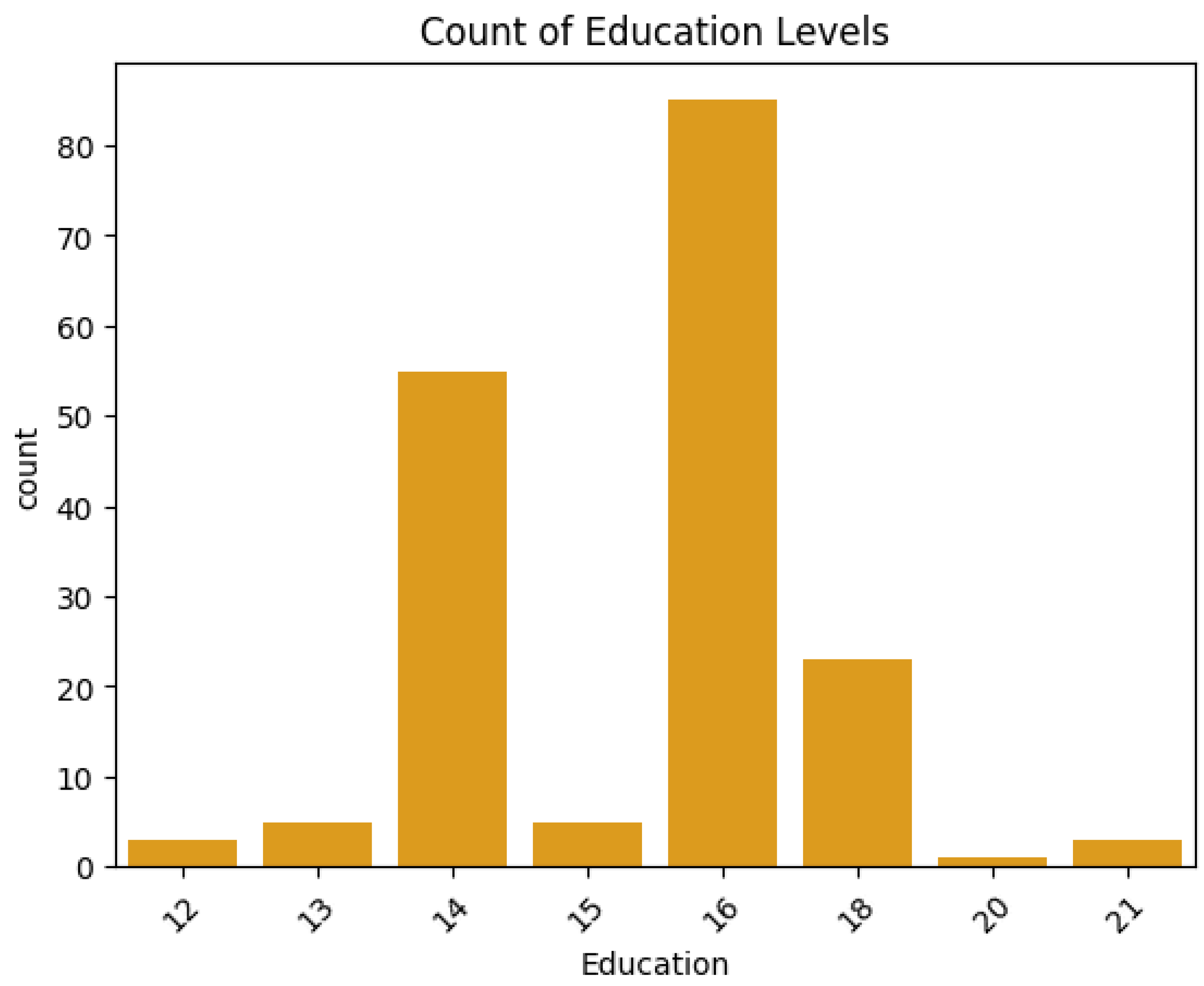
- **Marital Status:**

In [ ]:
```
sns.countplot(data=df, x="MaritalStatus",color='grey')
plt.title("Count of Marital Status")
plt.show()
```



- **Education:**

In [ ]:
```
sns.countplot(data=df, x="Education",color='orange')
plt.title("Count of Education Levels")
plt.xticks(rotation=45)
plt.show()
```



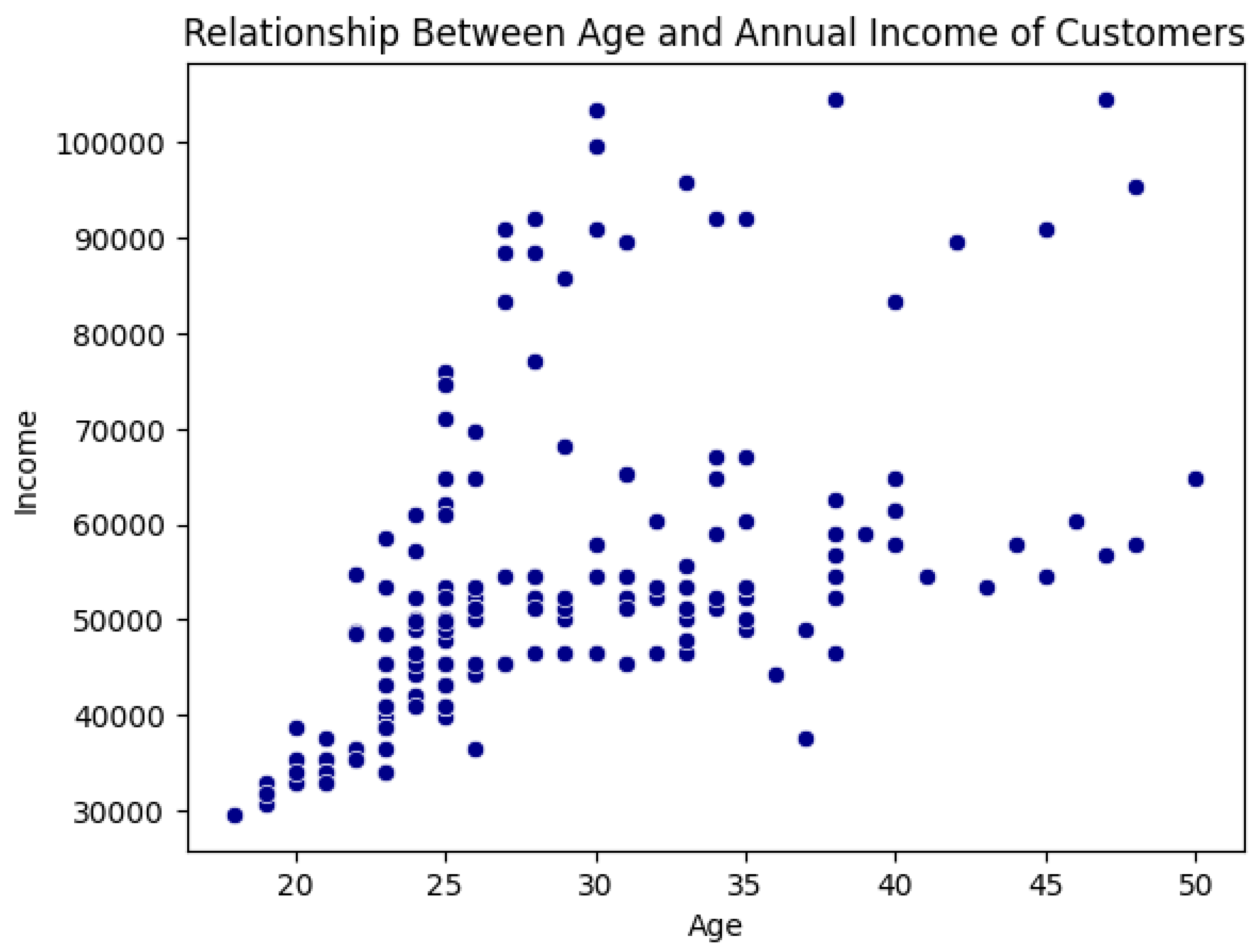
Bivariate Analysis:

- Bivariate = analyzing two variables together.

(a) Numerical vs Numerical:

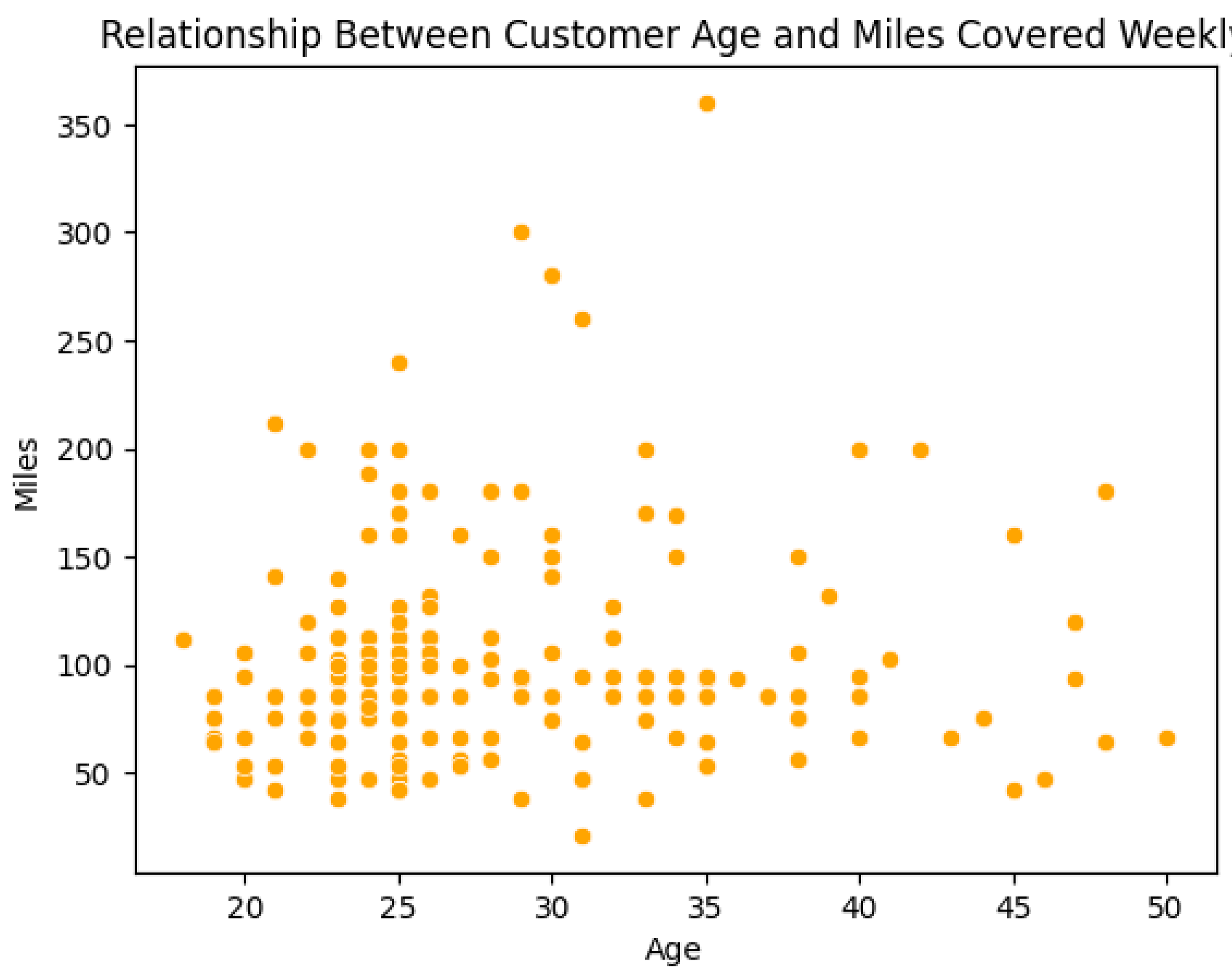
- **Age vs Income:**

In [ ]:
```
sns.scatterplot(data=df, x="Age", y="Income",color='darkblue')
plt.title("Relationship Between Age and Annual Income of Customers")
plt.show()
```



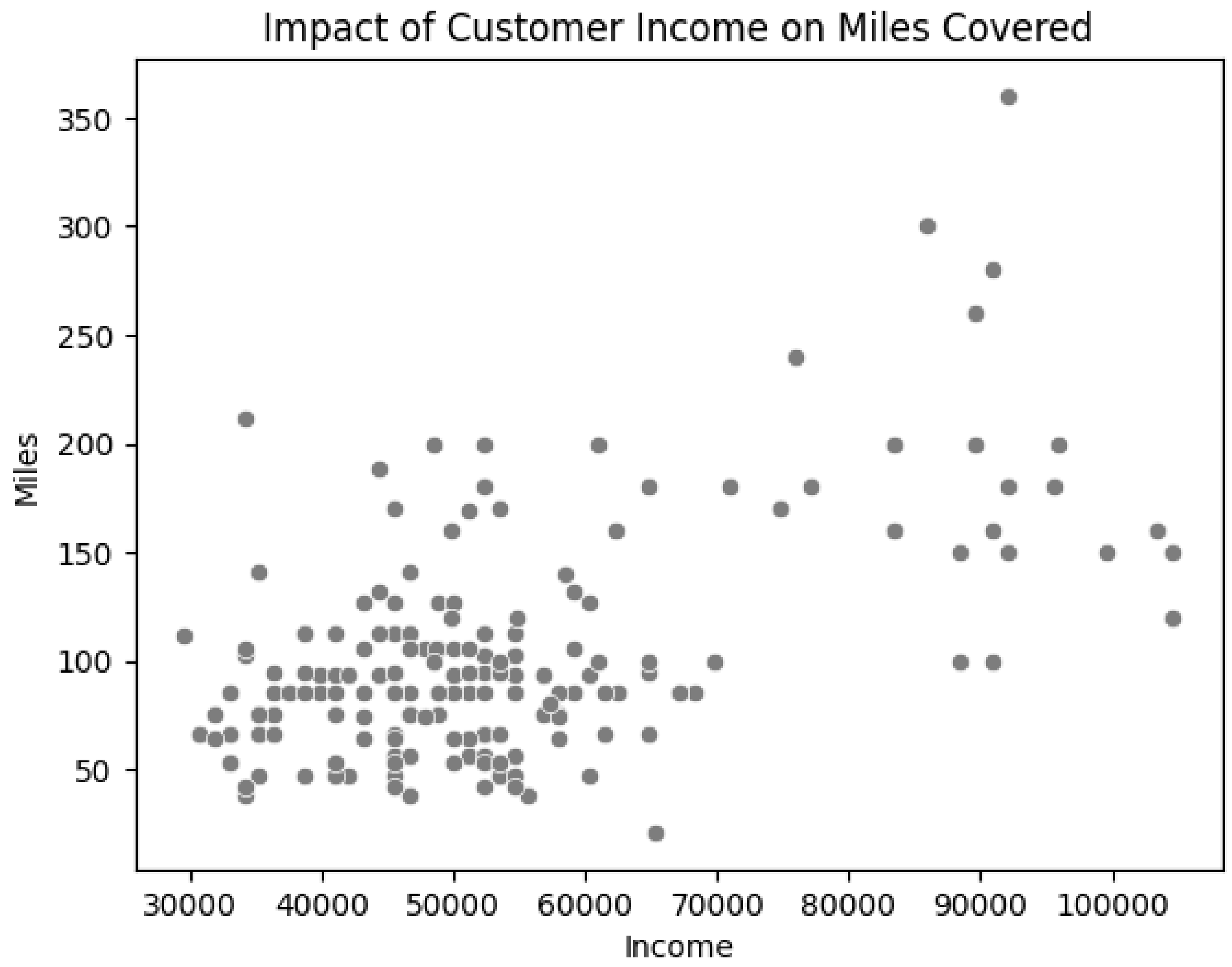
- **Age vs Miles:**

In [ ]:
```
sns.scatterplot(data=df, x="Age", y="Miles",color='orange')
plt.title("Relationship Between Customer Age and Miles Covered Weekly")
plt.show()
```



- **Income vs Miles:**

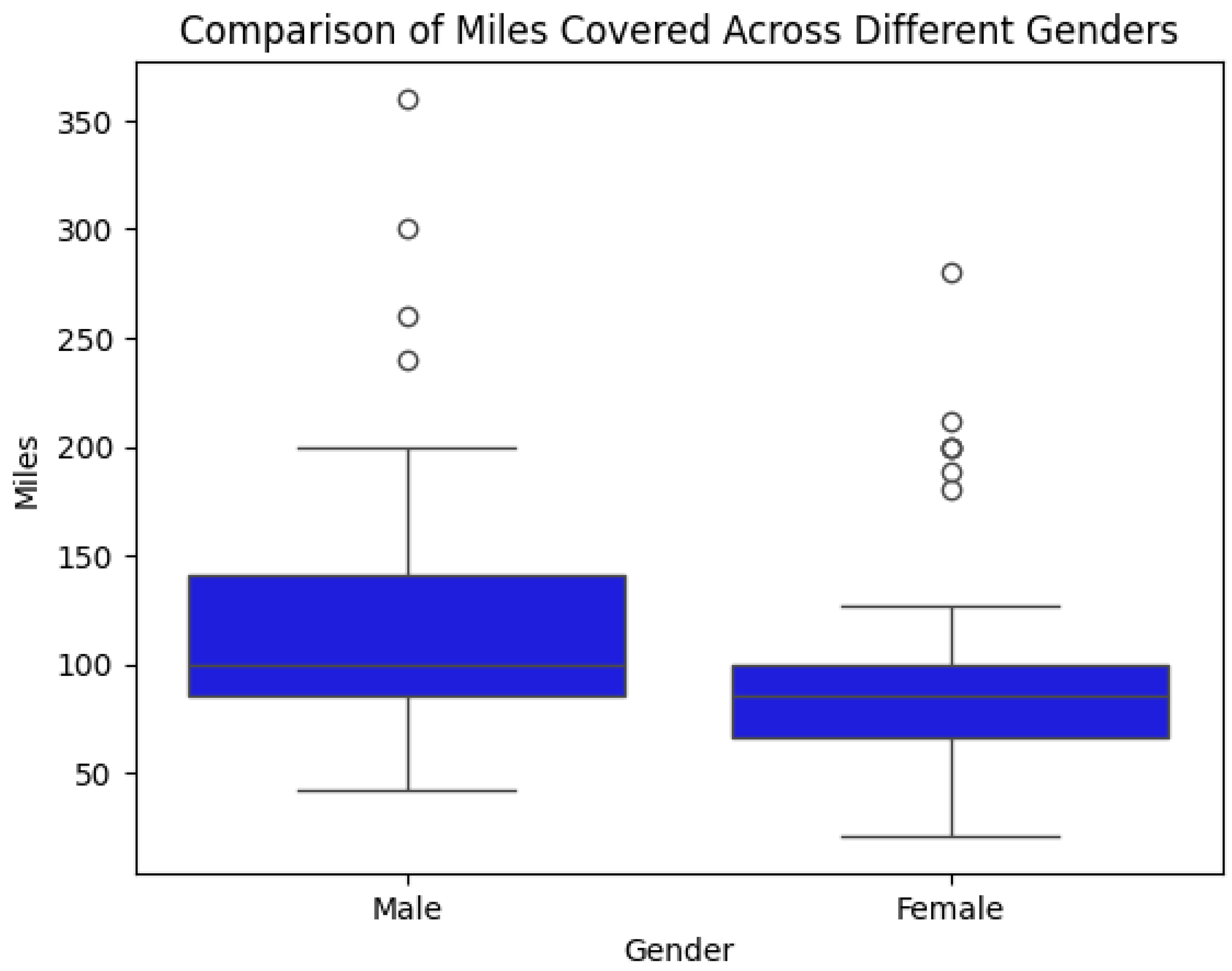
In [ ]:
```
sns.scatterplot(data=df, x="Income", y="Miles",color='grey')
plt.title("Impact of Customer Income on Miles Covered")
plt.show()
```



(b) Numerical vs Categorial:

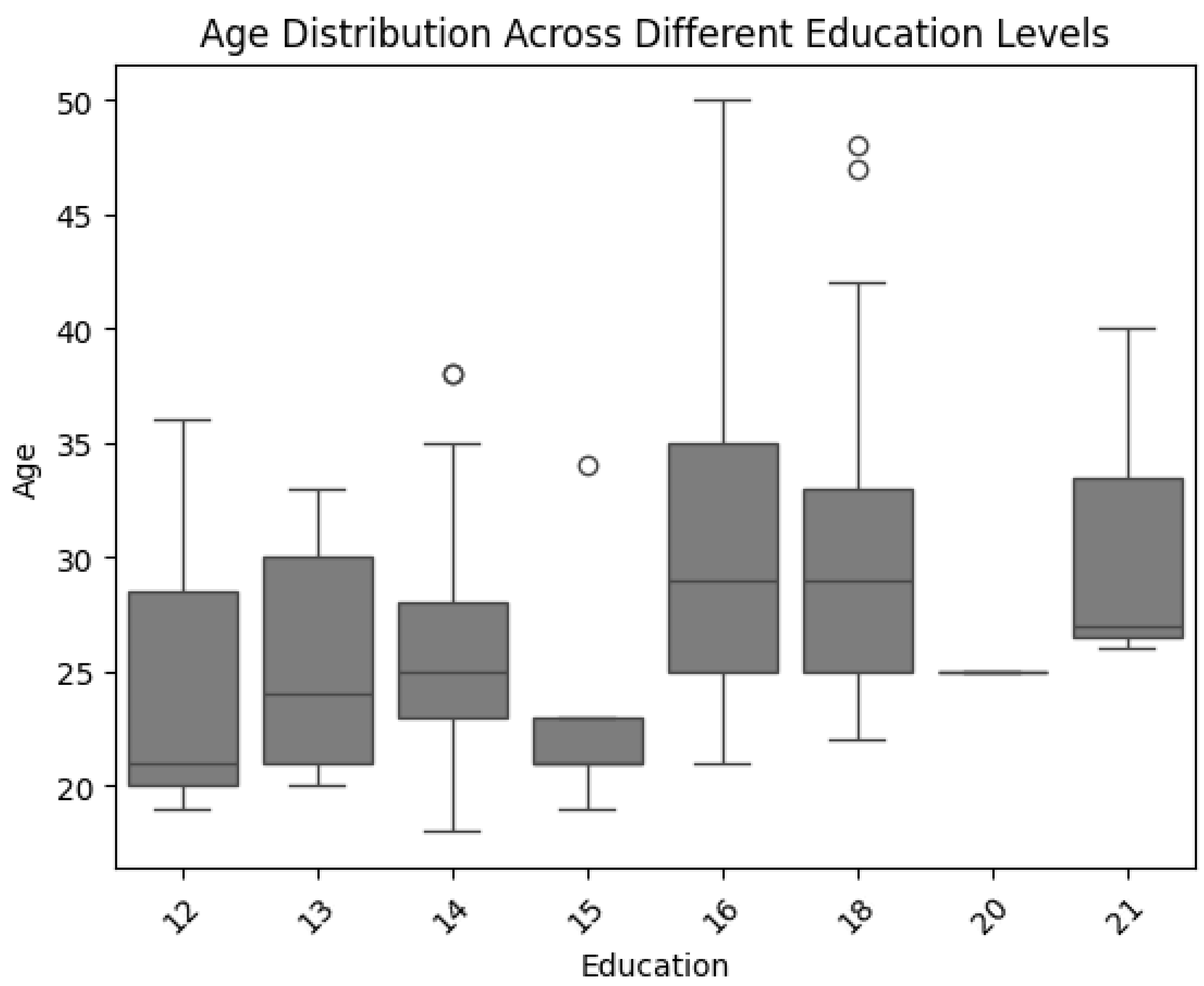
- **Gender vs Miles:**

In [ ]:
```
sns.boxplot(data=df, x="Gender", y="Miles", color='blue')
plt.title("Comparison of Miles Covered Across Different Genders")
plt.show()
```



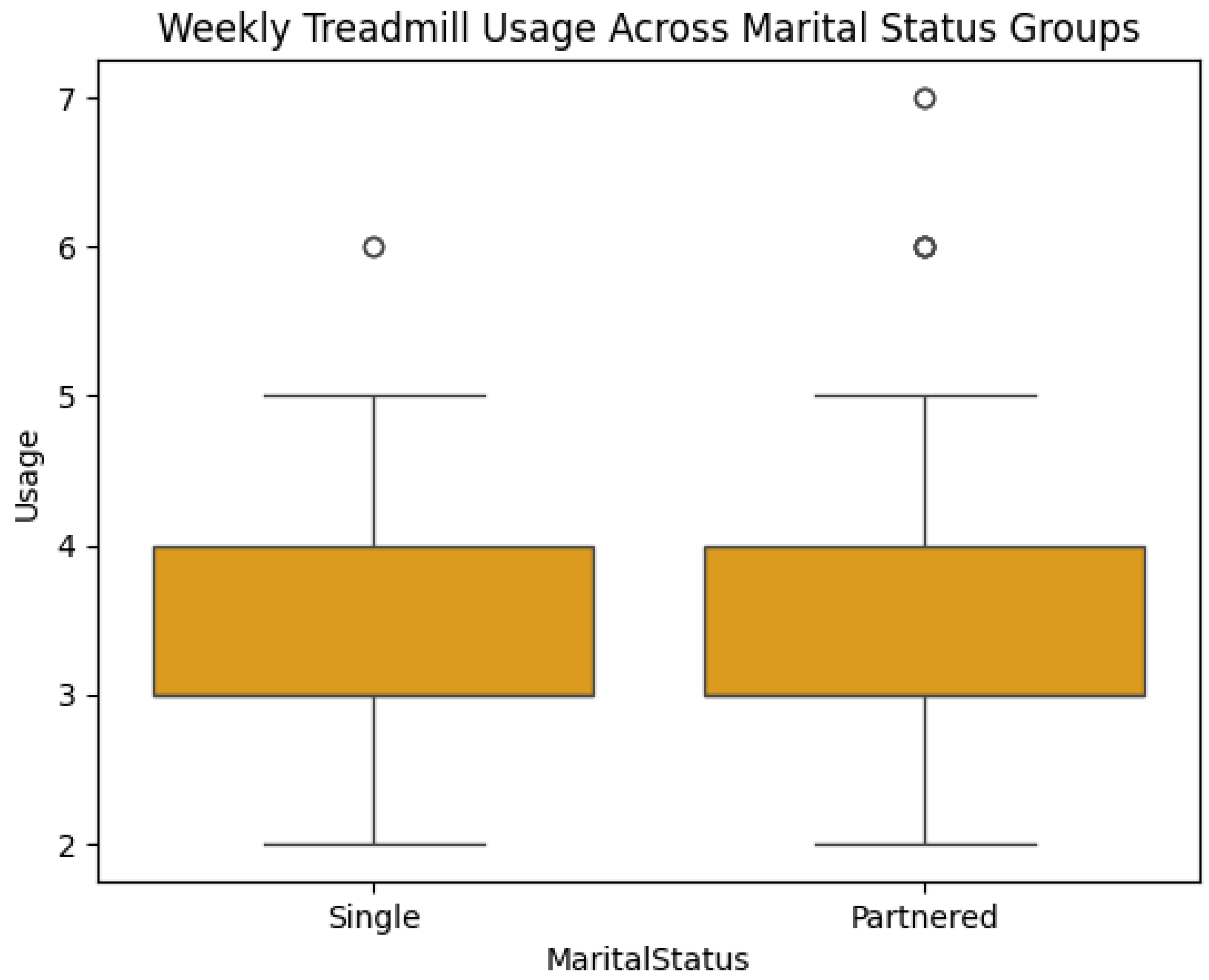
- **Education vs Age:**

In [ ]:
```
sns.boxplot(data=df, x="Education", y="Age", color='grey')
plt.title("Age Distribution Across Different Education Levels")
plt.xticks(rotation=45)
plt.show()
```



- **Marital Status vs Usage:**

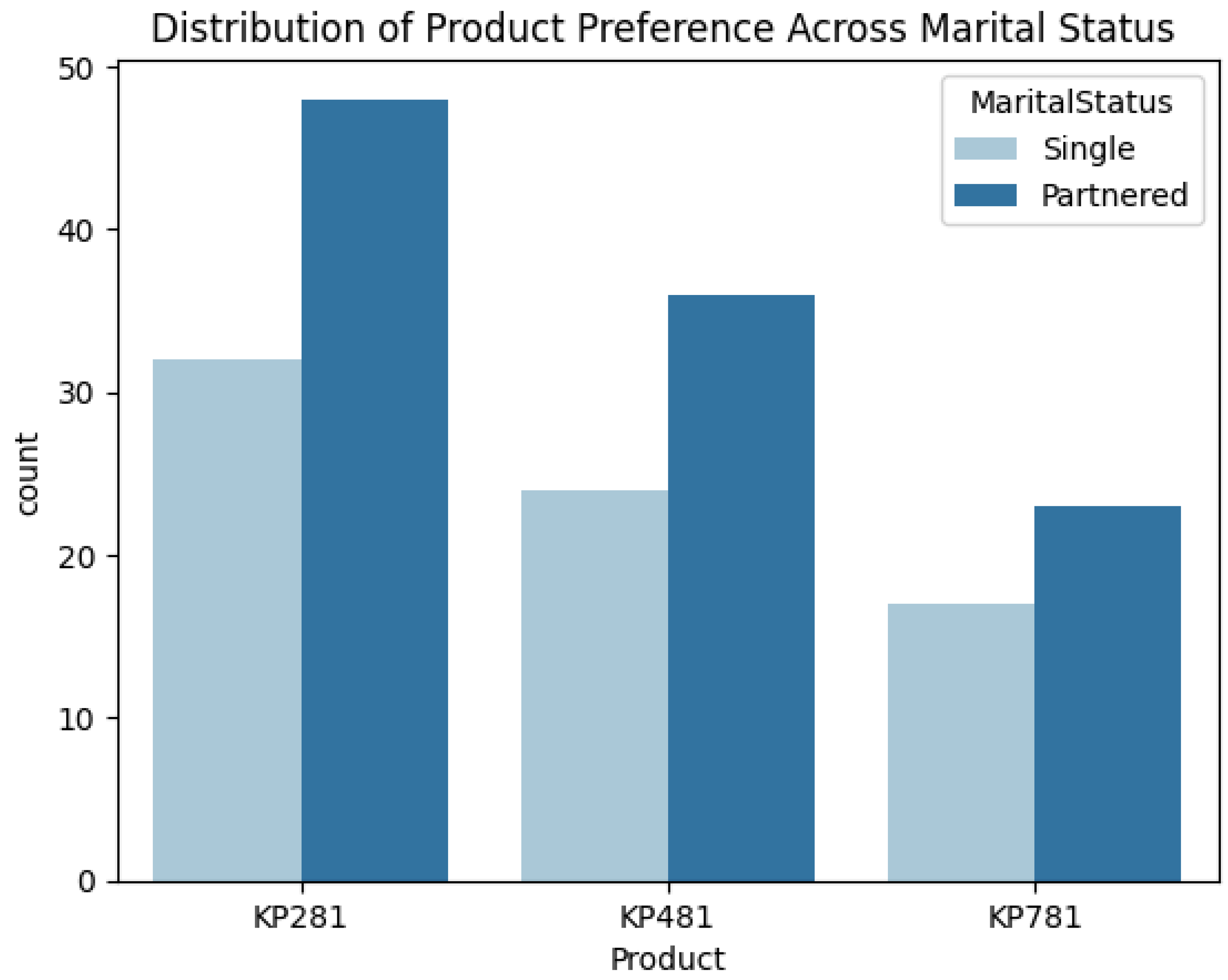
In [ ]:
```
sns.boxplot(data=df, x="MaritalStatus", y="Usage",color='orange')
plt.title("Weekly Treadmill Usage Across Marital Status Groups")
plt.show()
```



(c) Categorial vs Categorial:

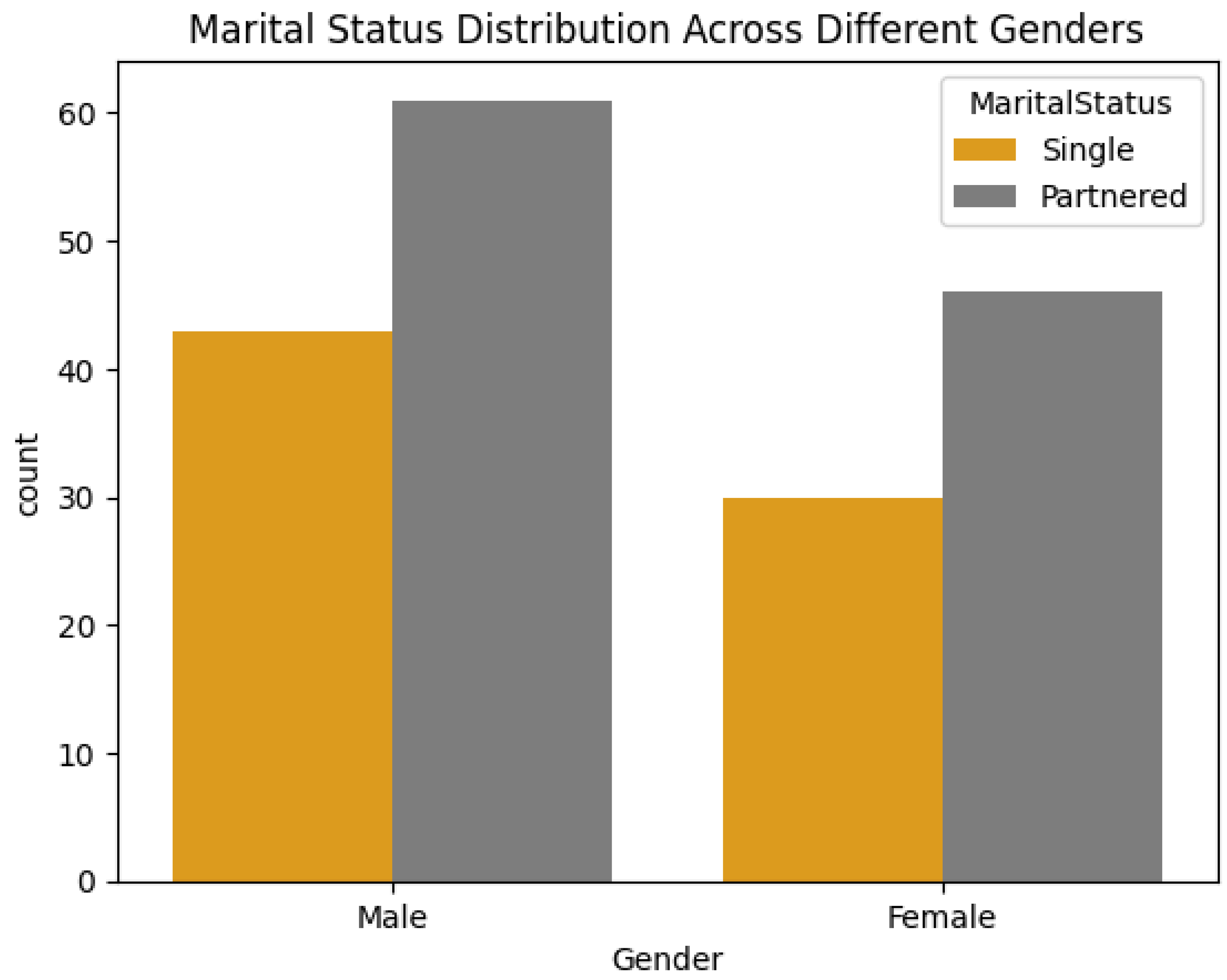
- **Product vs Marital Status:**

```
sns.countplot(data=df, x="Product", hue="MaritalStatus", palette='Paired')
plt.title("Distribution of Product Preference Across Marital Status ")
plt.show()
```



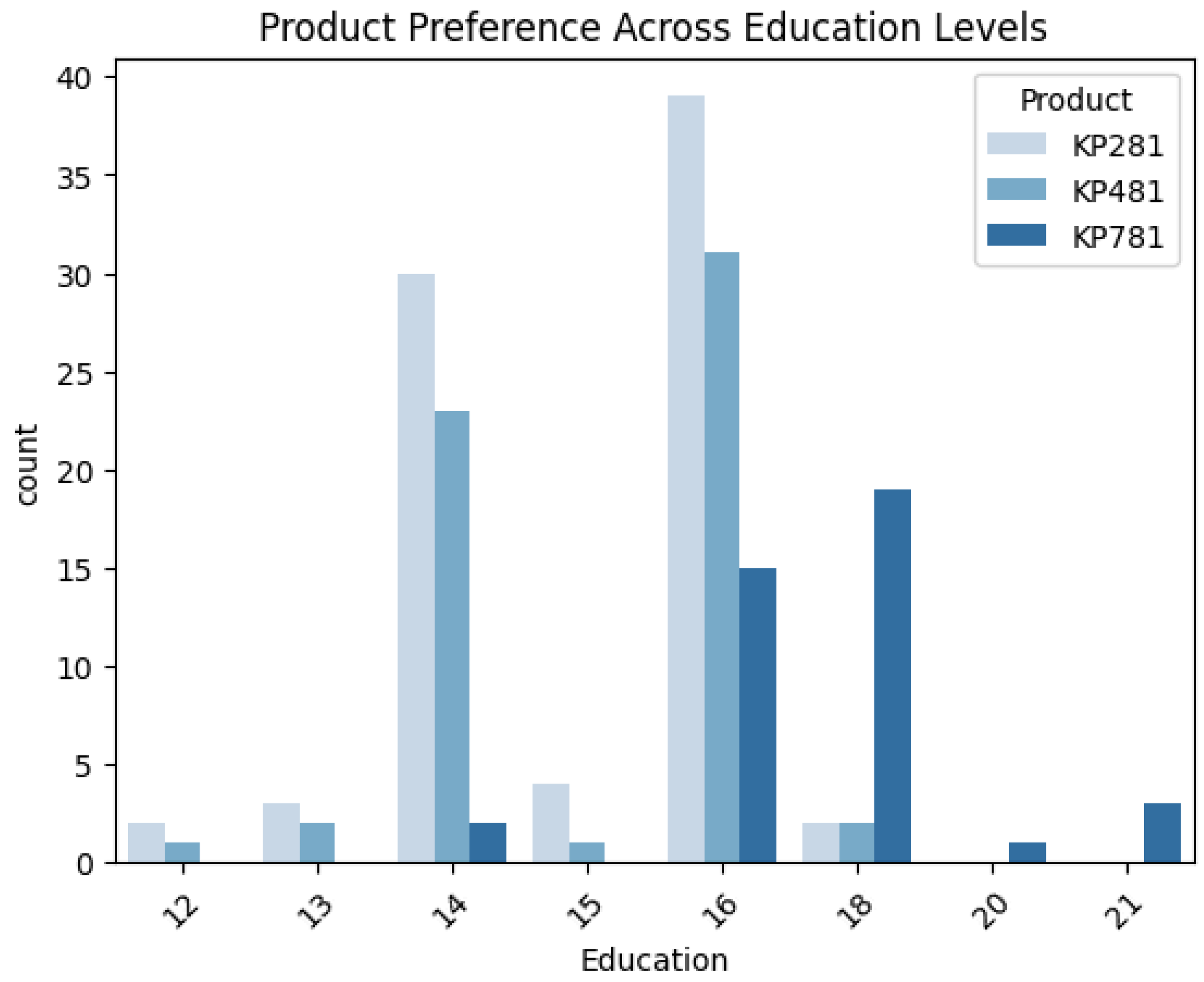
- **Gender vs Marital Status:**

```
sns.countplot(data=df, x="Gender", hue="MaritalStatus", palette=["orange", "grey"])
plt.title("Marital Status Distribution Across Different Genders")
plt.show()
```



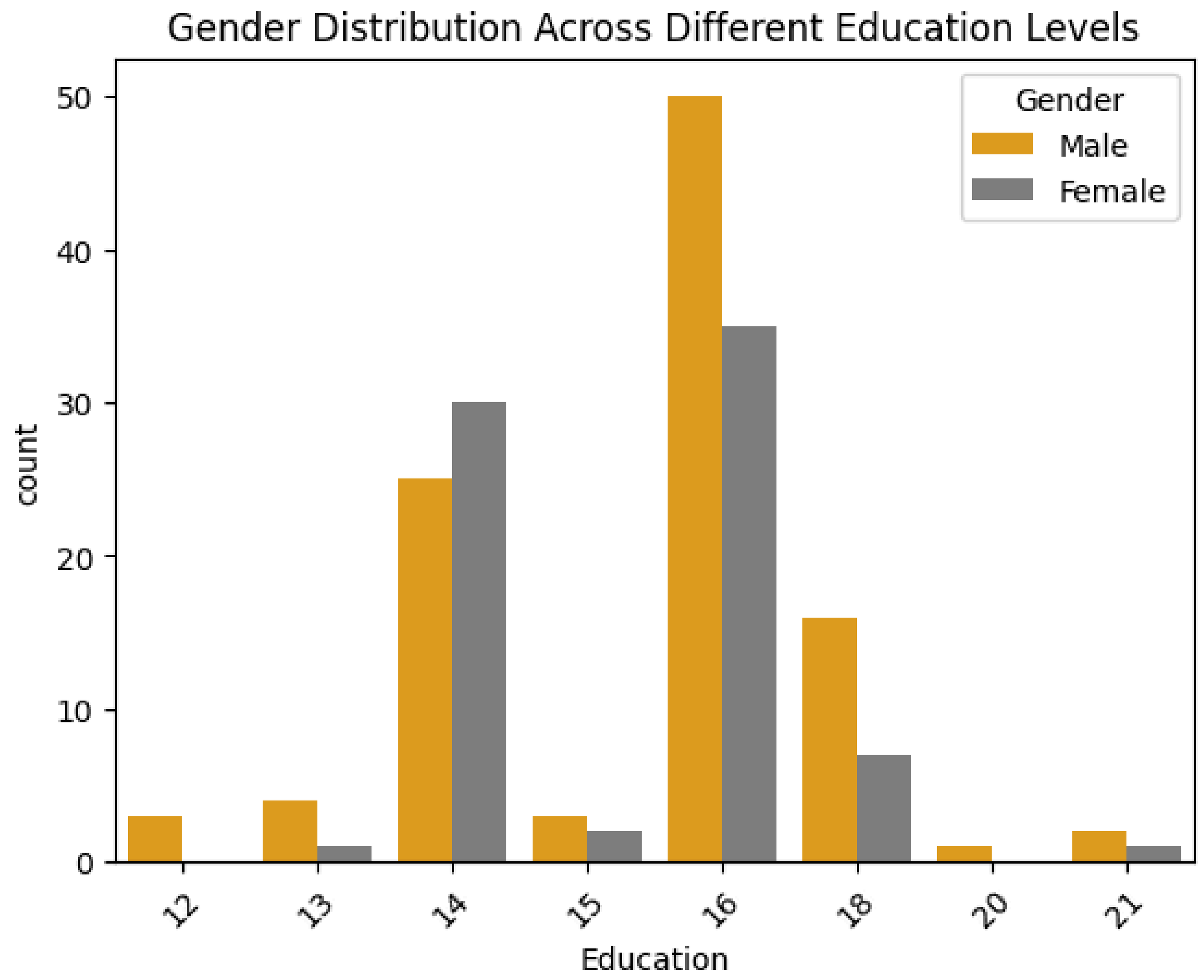
- **Education vs Product:**

```
sns.countplot(data=df, x='Education', hue="Product", palette='Blues')
plt.title("Product Preference Across Education Levels")
plt.xticks(rotation=45)
plt.show()
```



- **Education vs Gender:**

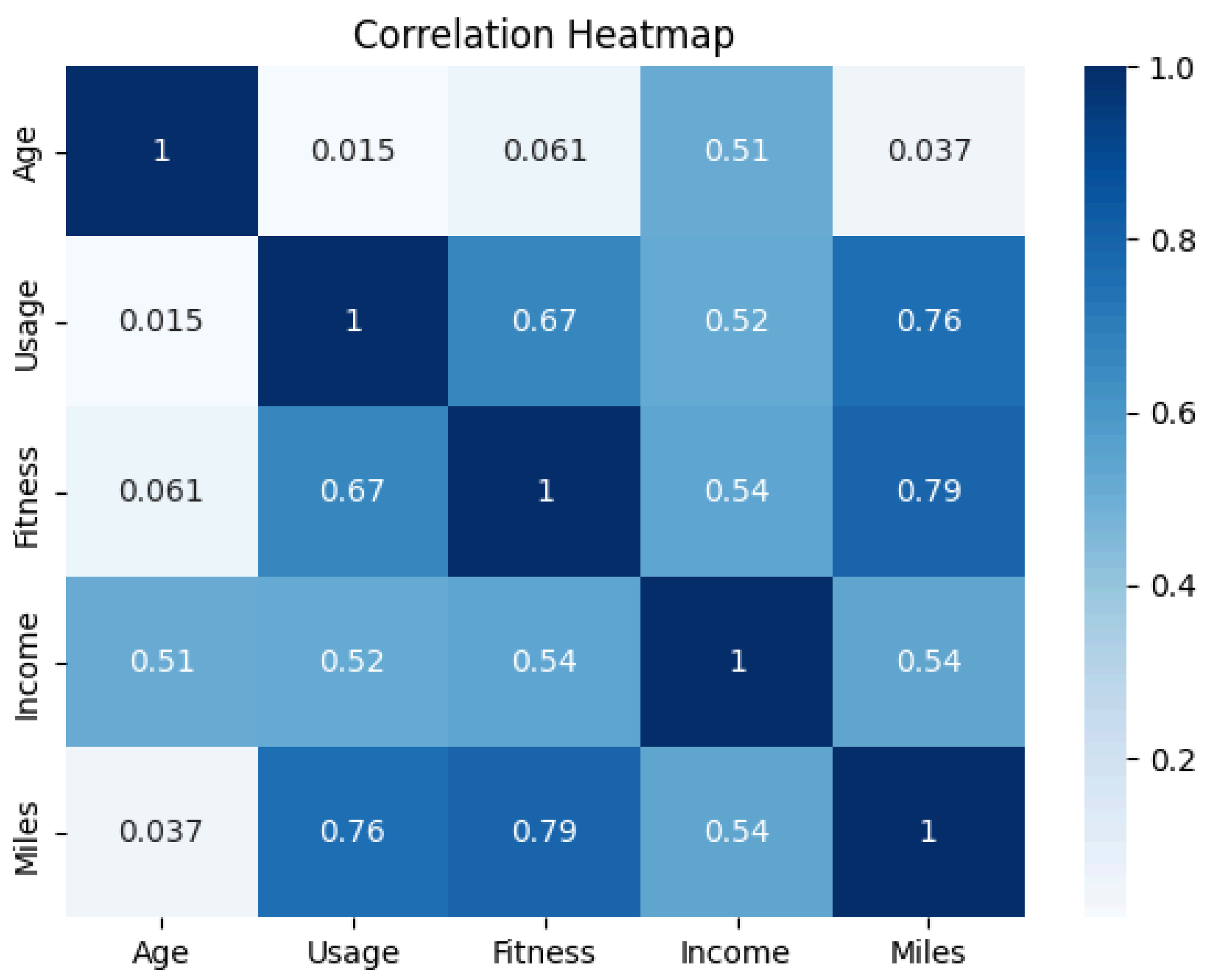
In [ ]:
```
sns.countplot(data=df, x="Education", hue="Gender", palette=["orange", "grey"])
plt.title("Gender Distribution Across Different Education Levels")
plt.xticks(rotation=45)
plt.show()
```



For Correlation:

- **Heatmap:**

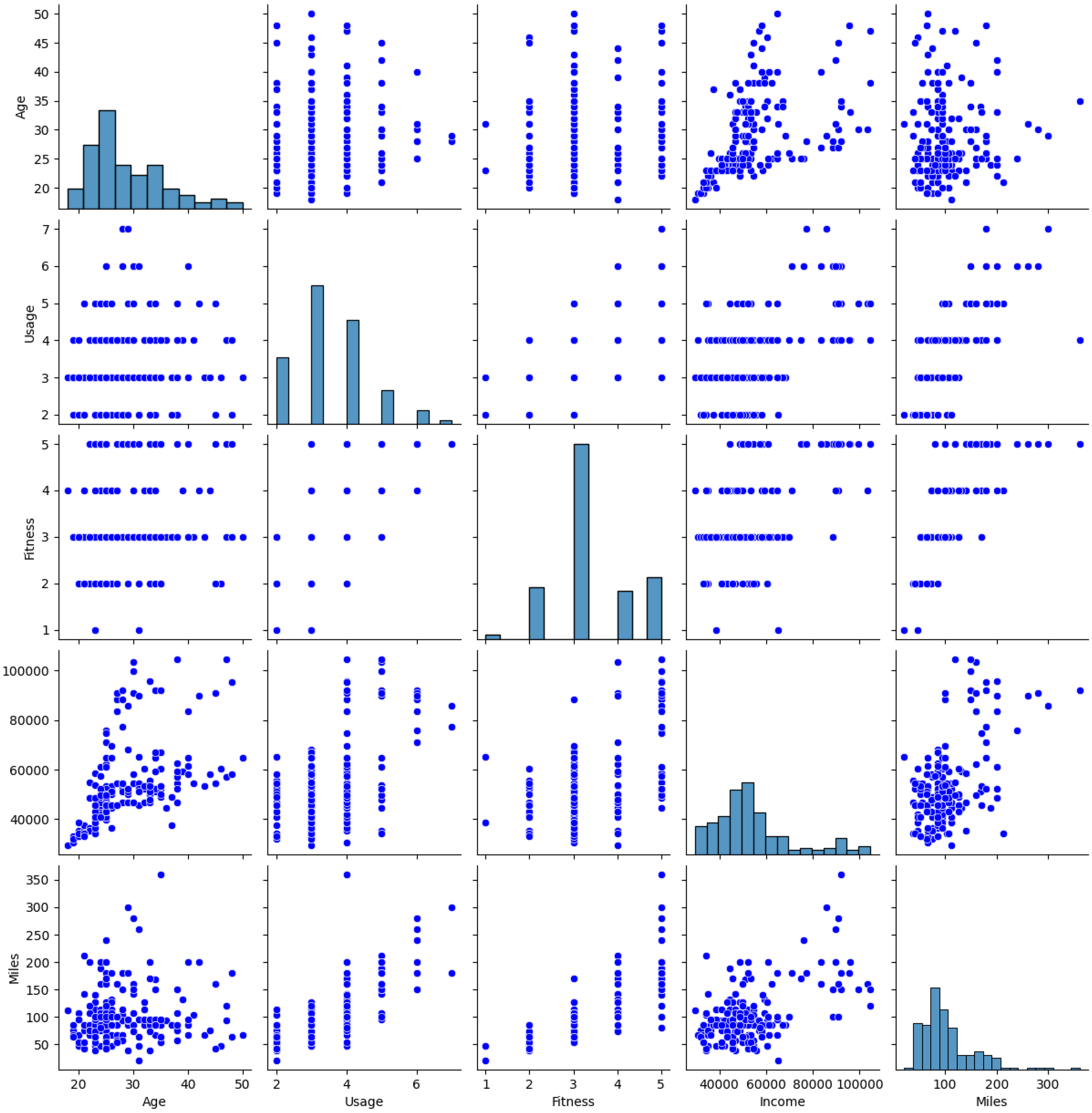
In [ ]:
```
numerical = ["Age", "Usage", "Fitness", "Income", "Miles"]
plt.figure(figsize=(7,5))
sns.heatmap(df[numerical].corr(), annot=True, cmap='Blues')
plt.title("Correlation Heatmap")
plt.show()
```



- **Pairplot:**

In [ ]:
```
numerical = ["Age", "Usage", "Fitness", "Income", "Miles"]

sns.pairplot(df[numerical],plot_kws={'color': 'blue'})
plt.show()
```



Outliers Check:

In [ ]:
```
df_num=df.describe()
df_num
```

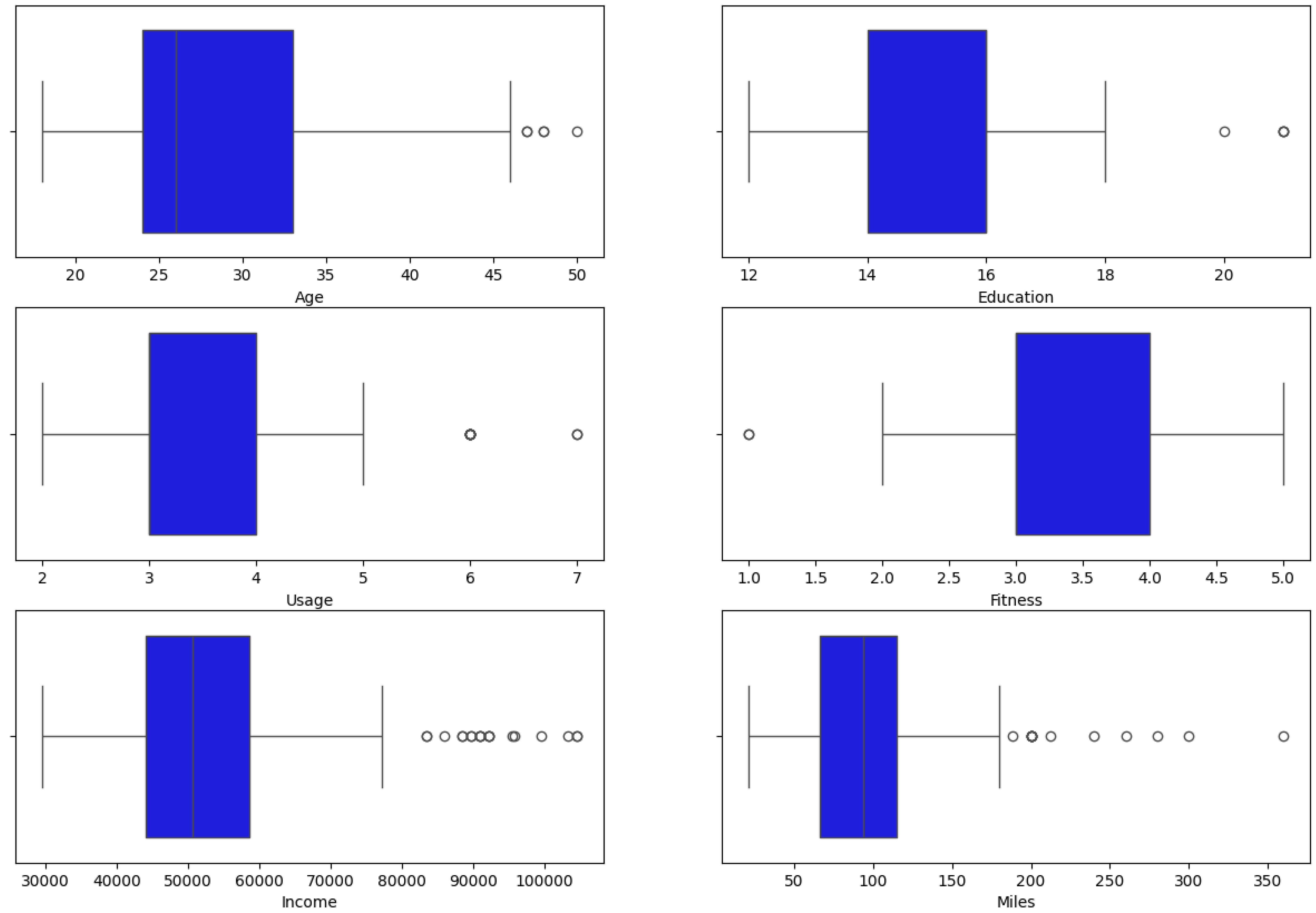
Out[ ]:

	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

In [ ]:
```
df_num.columns
```

Out[ ]:
```
Index(['Age', 'Education', 'Usage', 'Fitness', 'Income', 'Miles'], dtype='object')
```

In [ ]:
```
fig,axes=plt.subplots(nrows=3,ncols=2,figsize=(15,10))
axes=axes.flatten()
for i,j in enumerate(df_num.columns):
  sns.boxplot(x=df[j],ax=axes[i],color='blue')
```



Detect Outliers using boxplot, “describe” method by checking the difference between mean and median?

Income:

- **Detect Outliers Using describe():**

In [ ]:
```
df['Income'].describe()
```

Out[ ]:

	Income
count	180.000000
mean	53719.577778
std	16506.684226
min	29562.000000
25%	44058.750000
50%	50596.500000
75%	58668.000000
max	104581.000000

dtype: float64

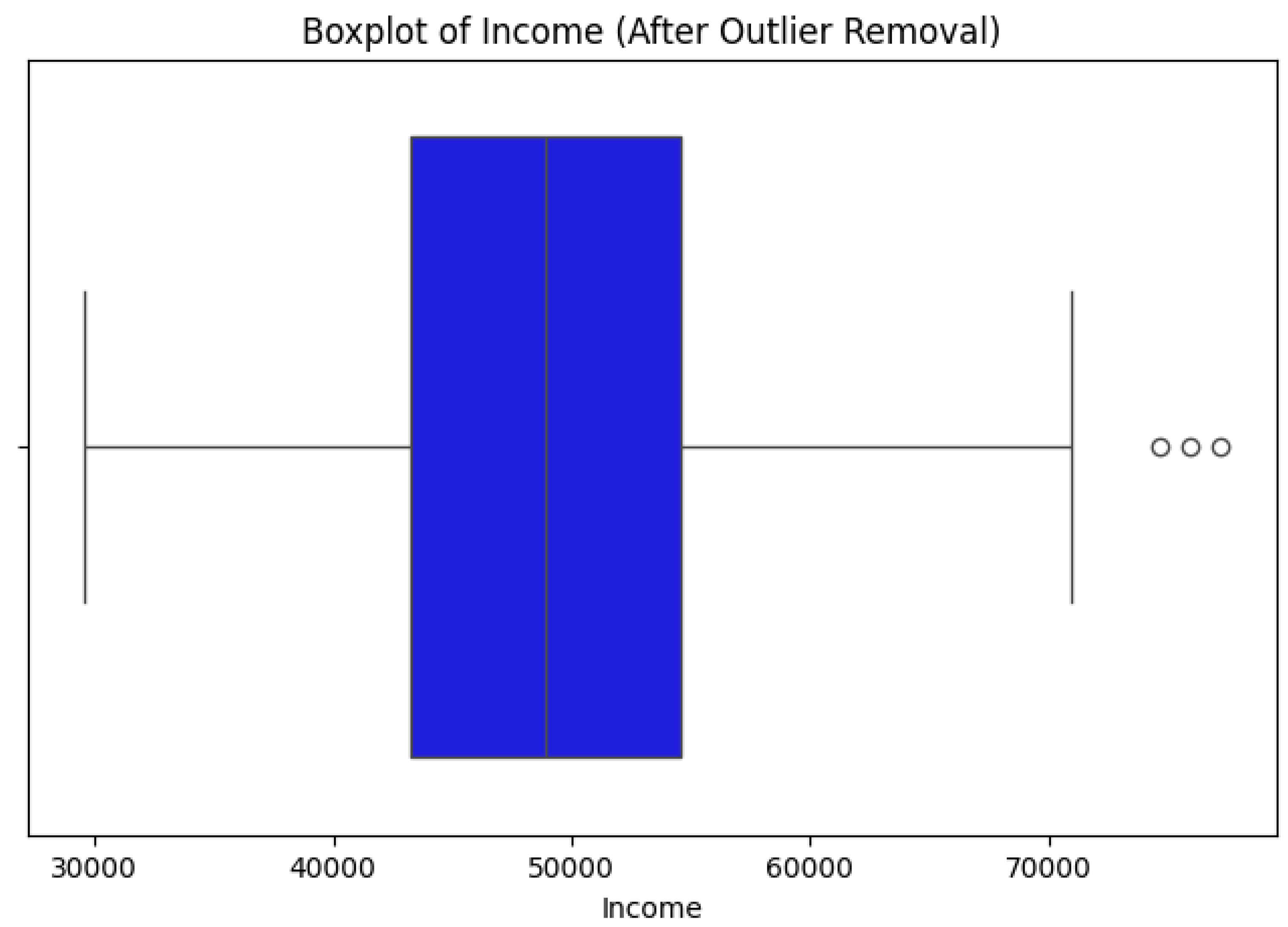
- **Detect Outlier:**

In [ ]:
```
Q1 = df['Income'].quantile(0.25)
Q3 = df['Income'].quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

df_income_clean = df[(df['Income'] >= lower_bound) & (df['Income'] <= upper_bound)]
```

In [ ]:
```
plt.figure(figsize=(8, 5))
sns.boxplot(x=df_income_clean['Income'], color='blue')
plt.title("Boxplot of Income (After Outlier Removal)")
plt.xlabel("Income")
plt.show()
```



- **Detect Outliers Using Mean vs Median Difference:**

In [ ]:
```
mean_val = df['Income'].mean()
median_val = df['Income'].median()

print("Mean:", mean_val)
print("Median:", median_val)
print("Difference:", abs(mean_val - median_val))
```

```
Mean: 53719.57777777778
Median: 50596.5
Difference: 3123.0777777777766
```

Miles:

- **Detect Outliers Using describe():**

In [ ]:
```
df['Miles'].describe()
```

Out[ ]:

	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

dtype: float64

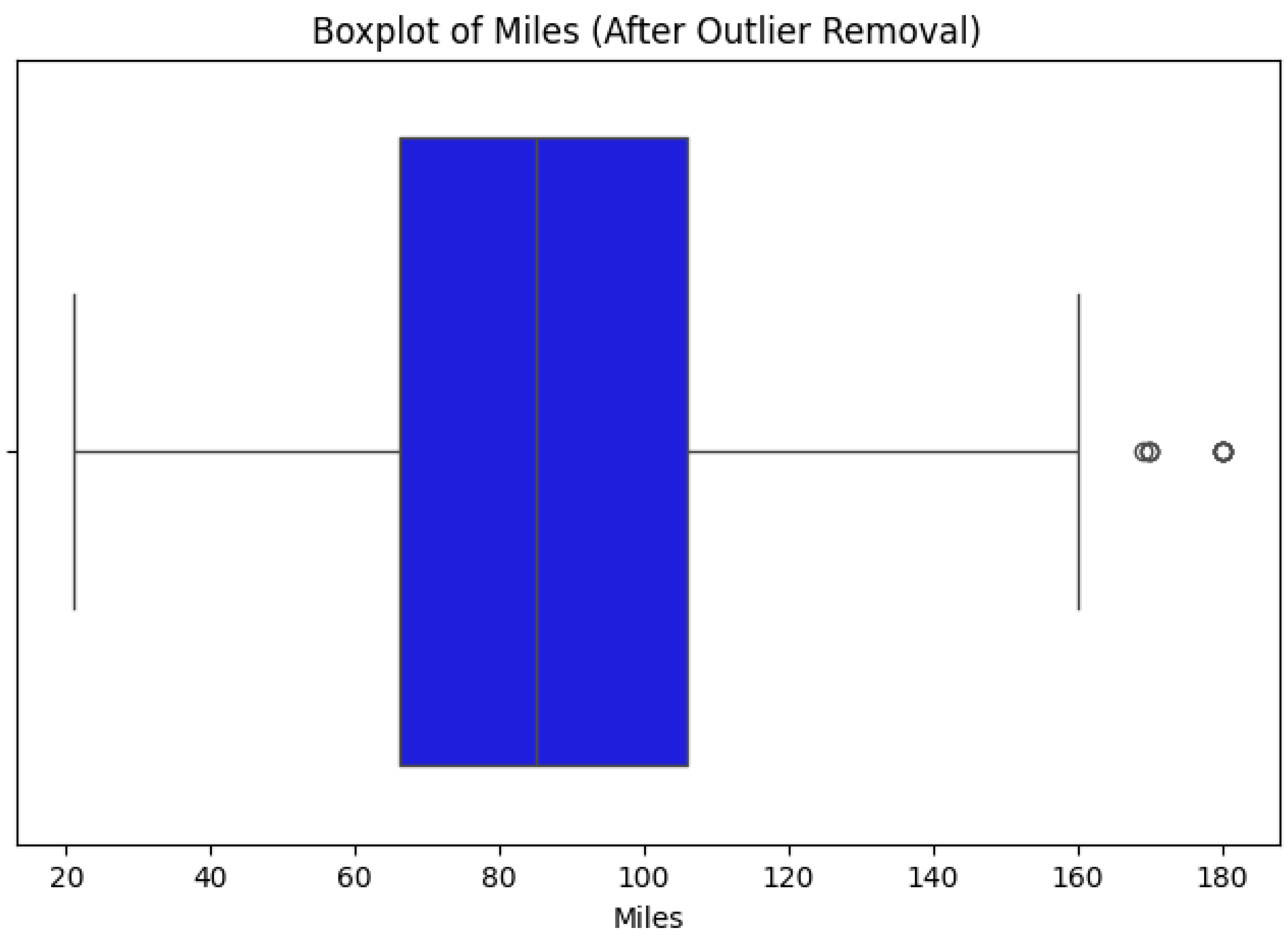
- **Detect Outlier:**

In [ ]:
```
Q1 = df['Miles'].quantile(0.25)
Q3 = df['Miles'].quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

df_miles_clean = df[(df['Miles'] >= lower_bound) & (df['Miles'] <= upper_bound)]
```

In [ ]:
```
plt.figure(figsize=(8, 5))
sns.boxplot(x=df_miles_clean['Miles'], color='blue')
plt.title("Boxplot of Miles (After Outlier Removal)")
plt.xlabel("Miles")
plt.show()
```



- **Detect Outliers Using Mean vs Median Difference:**

In [ ]:
```
mean_val = df['Miles'].mean()
median_val = df['Miles'].median()

print("Mean:", mean_val)
print("Median:", median_val)
print("Difference:", abs(mean_val - median_val))
```

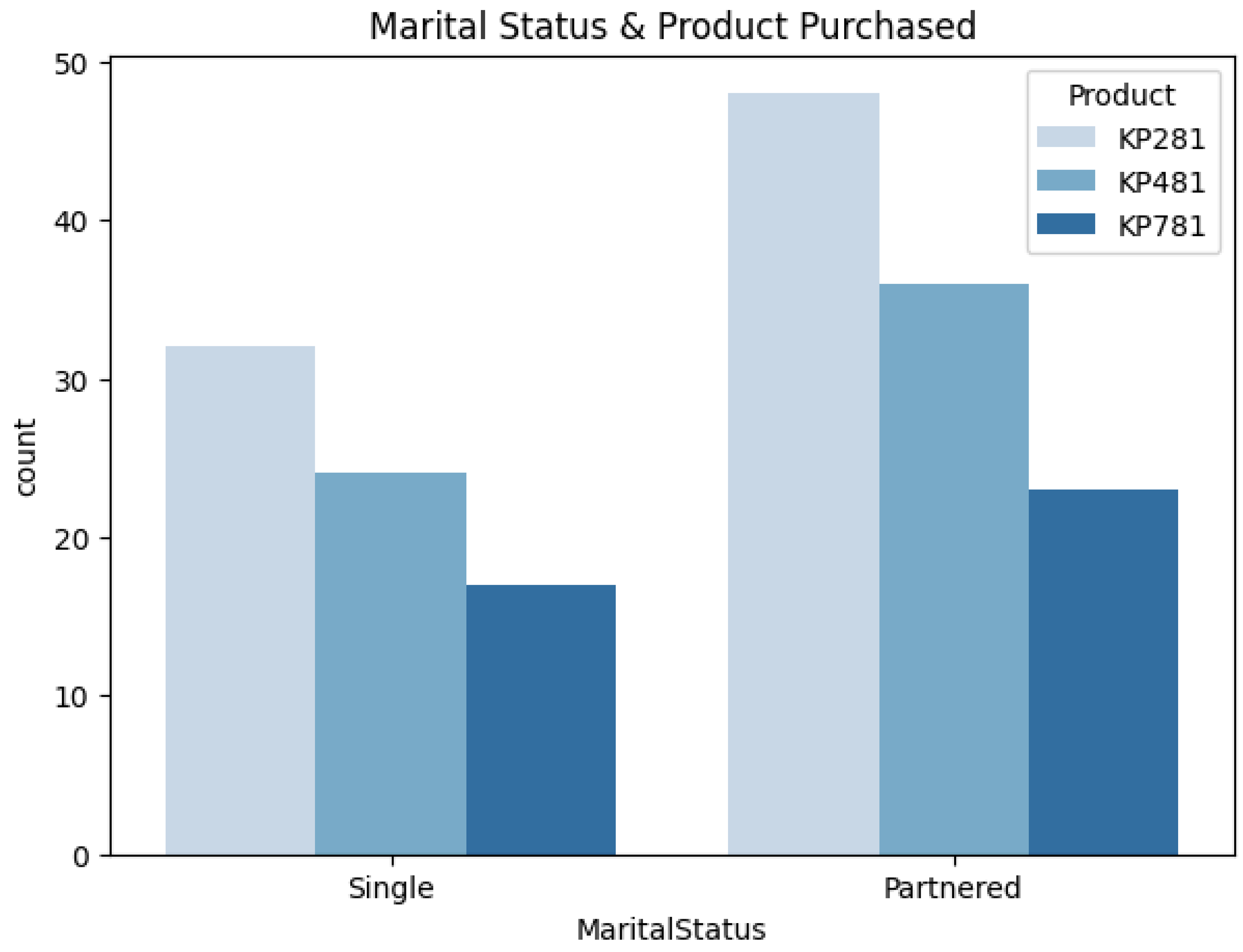
```
Mean: 103.19444444444444
Median: 94.0
Difference: 9.194444444444443
```

Queries Analysis:

1. Check if features like marital status, age have any effect on the product purchased?

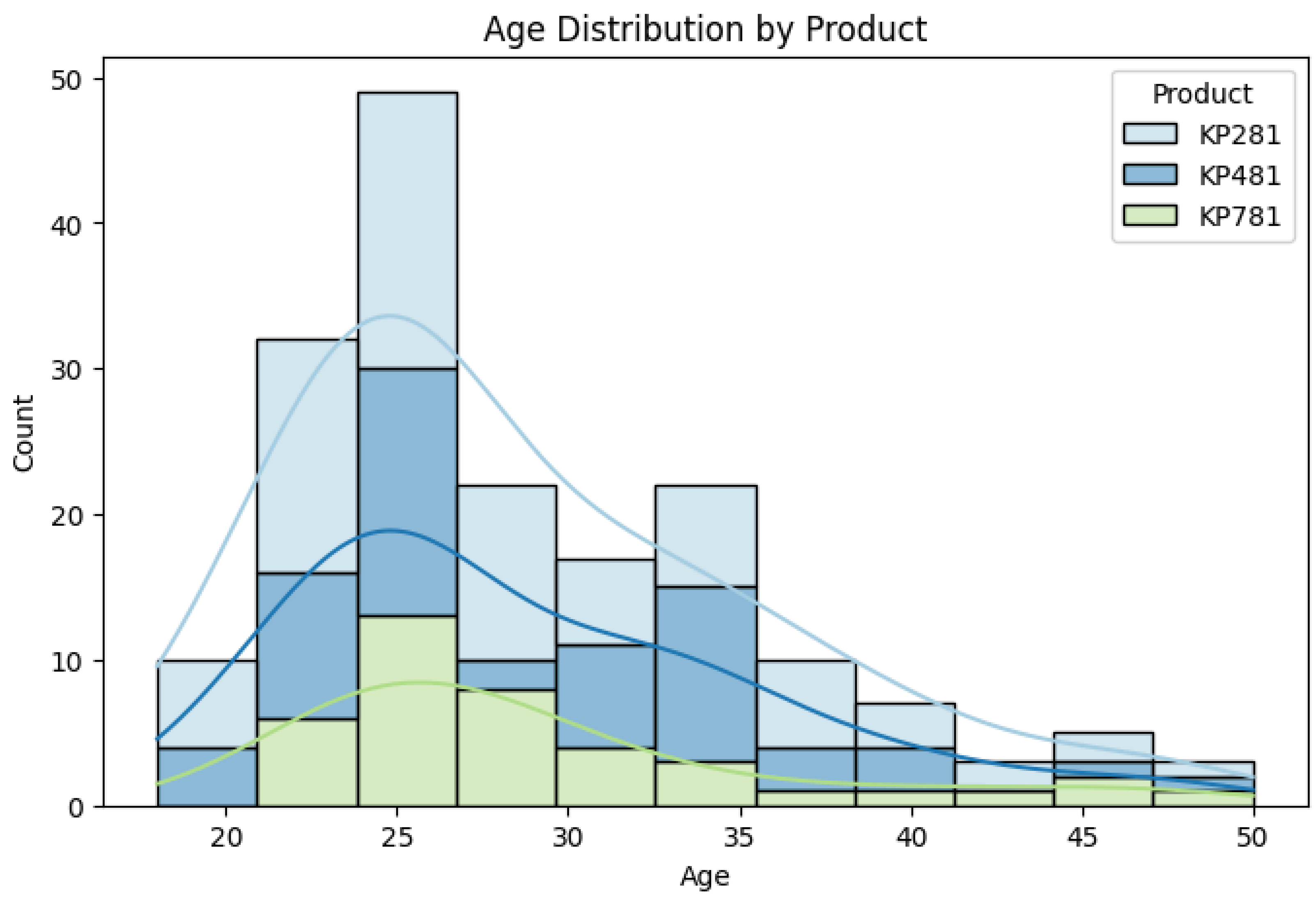
- **Does Marital Status affect Product Purchased?**

In [ ]:
```
plt.figure(figsize=(7,5))
sns.countplot(data=df, x='MaritalStatus', hue='Product', palette='Blues')
plt.title('Marital Status & Product Purchased')
plt.show()
```



- **Does Age affect Product Purchased?**

In [ ]:
```
plt.figure(figsize=(8,5))
sns.histplot(data=df, x='Age', hue='Product', kde=True, multiple='stack', palette='Paired')
plt.title('Age Distribution by Product')
plt.show()
```

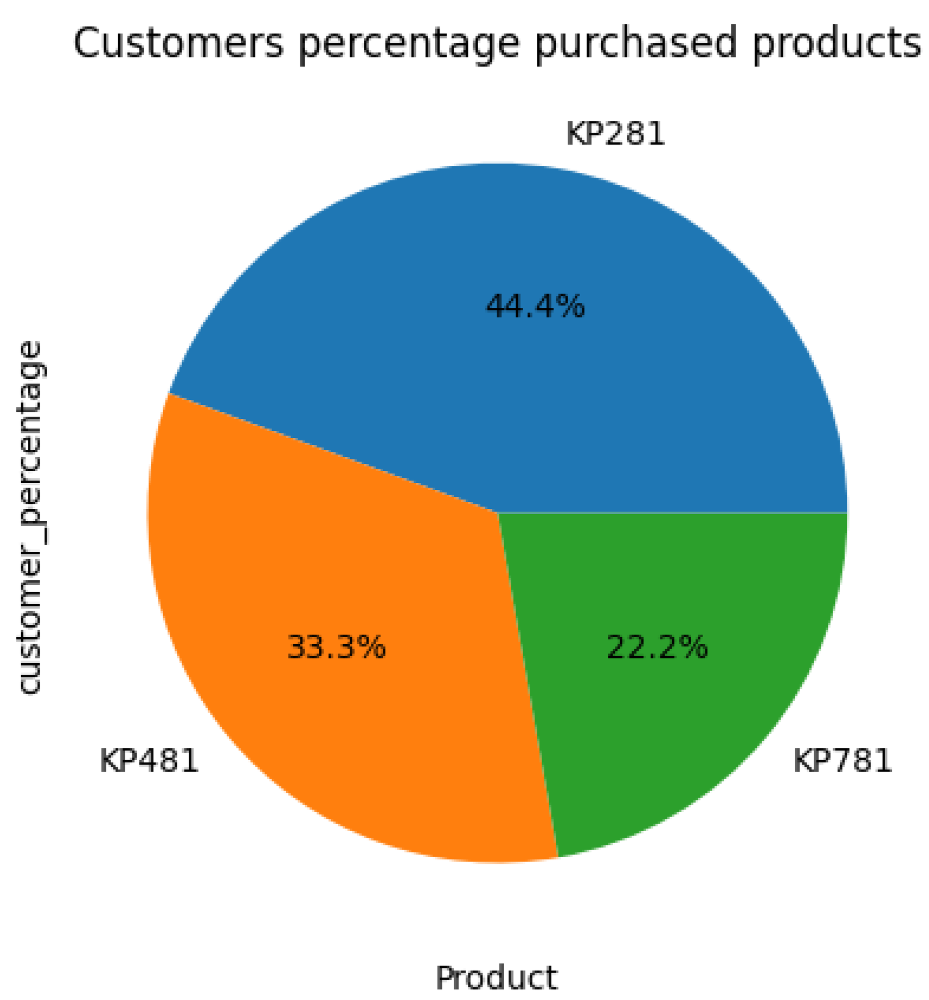


2. Representing the marginal probability like - what percent of customers have purchased KP281, KP481, or KP781 in a table (can use pandas.crosstab here)?

In [ ]:
```
counts = pd.crosstab(df['Product'], columns='Count')
percentages = pd.crosstab(df['Product'], columns='Percentage', normalize=True) * 100
marginal_probability = pd.concat([counts, percentages], axis=1)
marginal_probability['Percentage'] = marginal_probability['Percentage'].round(2)
print(marginal_probability)
```

```
col_0    Count  Percentage
Product                   
KP281       80       44.44
KP481       60       33.33
KP781       40       22.22
```

In [ ]:
```
marginal_probability['Percentage'].plot(kind='pie', autopct='%1.1f%%', labels=marginal_probability.index)
plt.xlabel('Product')
plt.ylabel('customer_percentage')
plt.title('Customers percentage purchased products')
plt.show()
```



3. With all the above steps you can answer questions like: What is the probability of a male customer buying a KP781 treadmill?

Step 1: Identify the Relevant Columns

- Gender
- Product Purchased

Step 2: Filter the Dataset for Male Customers are purchased product Kp781.

In [ ]:
```
male_data=df[df['Gender']=='Male']
total_males=len(male_data)
male_kp781 = male_data[male_data['Product'] == 'KP781']
male_kp781_count = len(male_kp781)
```

Step 3 : Probability of Male Customers are purchased product Kp781.

```
In [ ]: probability_of_males_KP781 = round(male_kp781_count / total_males,2)
        print("probability_of_males are purchased KP781 is :",probability_of_males_KP781)
```

```
probability_of_males are purchased KP781 is : 0.32
```

4. Customer Profiling - Categorization of users.

Customer Profiling: Customer profiling is the process of identifying and understanding different types of customers based on their characteristics, behavior, and product usage.

Categorization of users: Categorization of users means grouping customers into meaningful segments based on common features such as age, miles, usage level, and product preference.

Customer Profiling Code:

```
In [ ]: def customer_profile(row):
        # Home Fitness Beginner
            if row['Product'] == "KP281" and row['Usage'] <= 2:
                return "Home Fitness Beginner"
        # Weight Management User
            elif row['Miles'] >= 120:
                return "Weight Management User"
            else:
                return "General User"
        df['Customer_Profile'] = df.apply(customer_profile, axis=1)
        print(df['Customer_Profile'].value_counts())
```

```
Customer_Profile
General User              116
Weight Management User     45
Home Fitness Beginner      19
Name: count, dtype: int64
```

- **Insights:**

  • **General Users** show casual and irregular treadmill usage with low engagement.

  • **Weight Management Users** use the treadmill more intensely for weight loss goals.

  • **Home Fitness Beginners** are few, indicating limited adoption among new users.

Categorisation by Product:

```
In [ ]: df['Category'] = df['Product'].map({
            'KP281': 'Budget User',
            'KP781': 'Mid-Range User',
            'KP901': 'Premium User'
        })
        print(df[['Product','Category']].head())
```

```
  Product     Category
0   KP281  Budget User
1   KP281  Budget User
2   KP281  Budget User
3   KP281  Budget User
4   KP281  Budget User
```

Profile Summary:

```
In [ ]: profile_summary = df.groupby('Customer_Profile').agg({
            'Income':'mean',
            'Miles':'mean',
            'Usage':'mean',
            'Product':'count'
        })

        print(profile_summary)
```

```
                              Income       Miles     Usage  Product
Customer_Profile
General User            49056.775862   82.396552  3.206897      116
Home Fitness Beginner   46437.473684   59.842105  2.000000       19
Weight Management User  68813.911111  175.111111  4.711111       45
```

```
In [ ]: df.head()
```

Out[ ]:

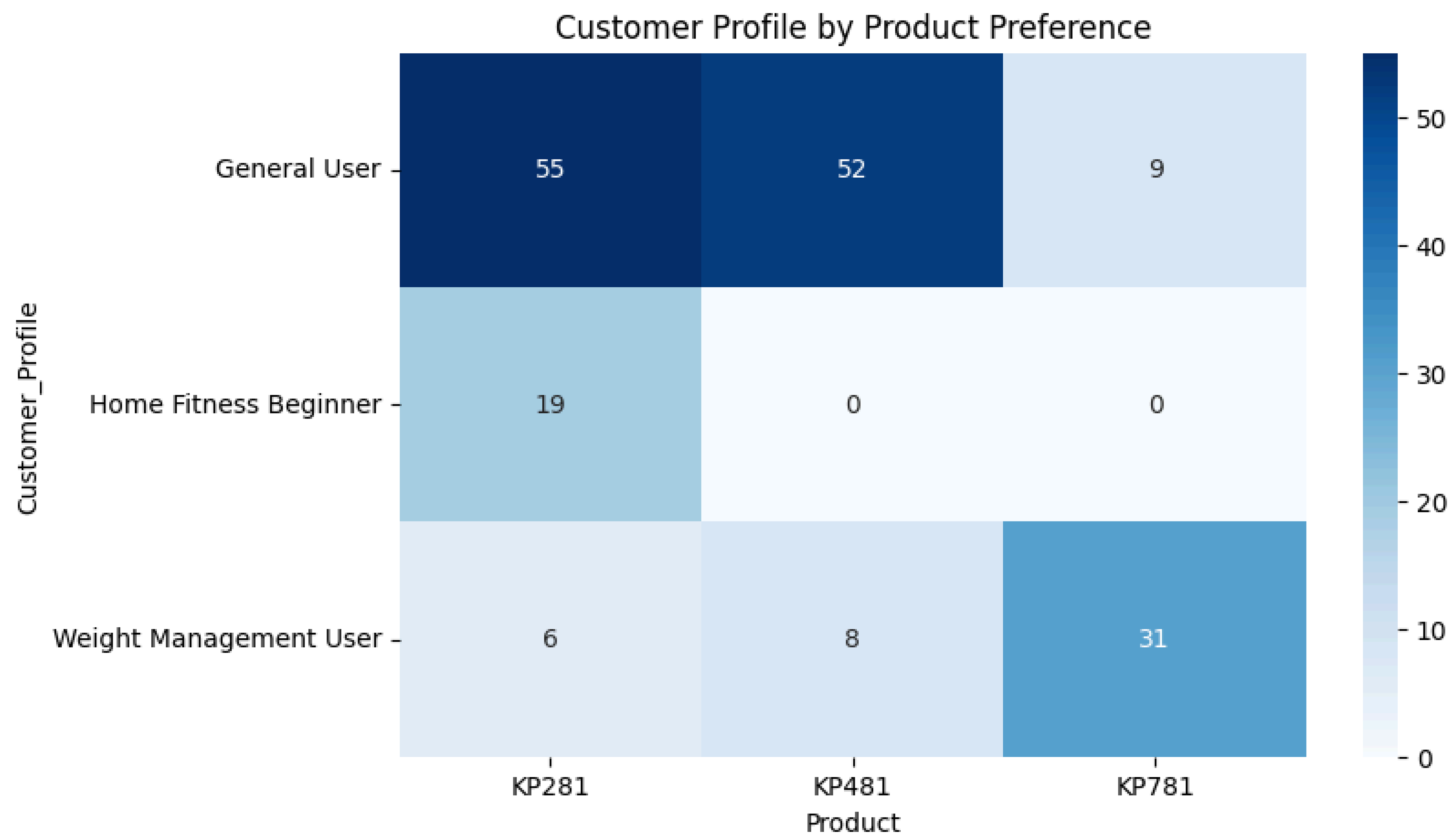
	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles	Age_Group	Income_Level	Customer_Profile	Category
0	KP281	18	Male	14	Single	3	4	29562	112	Young (17-30)	Low	General User	Budget User
1	KP281	19	Male	15	Single	2	3	31836	75	Young (17-30)	Mid	Home Fitness Beginner	Budget User
2	KP281	19	Female	14	Partnered	4	3	30699	66	Young (17-30)	Mid	General User	Budget User
3	KP281	19	Male	12	Single	3	3	32973	85	Young (17-30)	Mid	General User	Budget User
4	KP281	20	Male	13	Partnered	4	2	35247	47	Young (17-30)	Mid	General User	Budget User

Customer Profile Count:

```
In [ ]: plt.figure(figsize=(8,5))
        sns.countplot(data=df, x='Customer_Profile', color='blue')
        plt.title("Customer Profile Distribution")
        plt.xticks(rotation=45)
        plt.show()
```

Customer Profile vs Product:

In [ ]:
```
profile_product = pd.crosstab(df['Customer_Profile'], df['Product'])
plt.figure(figsize=(8,5))
sns.heatmap(profile_product, annot=True,cmap='Blues' )
plt.title("Customer Profile by Product Preference")
plt.show()
```



- **Insights:**

  • Home Fitness Beginners mostly choose KP281, reflecting price-sensitive buying.

  • General Users are spread across all models, showing purchases driven by initial interest.

  • Weight Management Users lean toward mid-range models, balancing cost and performance.

5. Probability- marginal, Conditional probability:

Marginal Probability : Marginal probability refers to the probability of a single event happening without any condition.

- What is the marginal probability that a customer runs more than 5 miles per week? Also, what is the conditional probability that a customer is high-income given that they run.

In [ ]:
```
total_customers = len(df)
miles_above_5 = len(df[df['Miles'] > 5])
marginal_prob_miles = miles_above_5 / total_customers
print("miles_above_5 :",miles_above_5)
print("Marginal Probability that a customer runs more than 5 miles per week:", marginal_prob_miles)
```

```
miles_above_5 : 180
Marginal Probability that a customer runs more than 5 miles per week: 1.0
```

Conditional Probability: Conditional probability is the probability of an event occurring given that another event has already happened.

- What is the conditional probability that a customer is high-income given that they run more than 5 miles per week?

In [ ]:
```
miles_group = df[df['Miles'] > 5]
high_income_miles = len(miles_group[miles_group['Income_Level'] == 'High'])
conditional_prob_income_given_miles = round(high_income_miles / miles_above_5,2)
print("high_income_miles is:",high_income_miles)
print("conditional_probablity that a customer runs more than 5 miles per week:",conditional_prob_income_given_miles)
```

```
high_income_miles is: 39
conditional_probablity that a customer runs more than 5 miles per week: 0.22
```

Business Insights:

1. Comments on the range of attributes:

- Attributes cover a wide range: young to older customers, low to high income, low to high miles.
- Both genders, all education levels, and all marital statuses are represented.
- Fitness levels vary from low to high, showing mixed customer maturity.
- Product range (KP281, KP481, KP781) covers budget, mid-range, and premium buyers.

2. Comments on the distribution & relationships:

- Miles and usage show a positive pattern—more usage leads to higher miles.
- Medium fitness customers form the largest segment.
- Higher-income customers prefer premium products like KP781.

- Older customers tend to buy more advanced models.

3. Comments for Each Univariate and Bivariate Plot:

Univariate (Single Variable):

- Age distribution highlights the most common buying age group.
- Gender countplot shows whether males or females dominate purchases.
- Education plot shows how awareness differs across qualification levels.
- Marital status plot reveals whether single or married customers buy more.
- Usage histogram shows how often customers use their treadmill weekly.

Bivariate Plots:

- Education vs Income shows higher education often relates to higher income.
- Marital Status vs Product shows purchasing patterns across family types.
- Usage vs Miles shows a strong positive relationship—more usage means more miles.
- Fitness vs Product shows fitter customers prefer higher-end models.
- Age vs Miles shows which age groups use the treadmill more consistently.

Recommendations:

⭐ Give special offers and EMI plans to attract higher-income buyers toward premium models.

🎯 Target middle-aged customers more, as they form the largest buying group.

👥 Provide couple/family discounts since many married customers purchase treadmills.

💡 Create beginner workout guides to support low-fitness customers and boost engagement.

🏷️ Run youth-focused promotions to increase sales of the basic model for younger buyers.

🎁 Introduce loyalty rewards for frequent users to encourage long-term brand connection.