Aerofit- case study

**Business Problem - *

Analyzing the data and help AeroFit 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

```
#importing data of Aerofit
import pandas as pd
df= pd.read_csv('aerofit_treadmill_csv.txt')
df
```

	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

Next steps: Generate code with df View recommended plots New interactive sheet

**Basic Matrix - *

```
a= df['Product'].nunique()
print(a, 'Different types of treadmill ')
# Identifying different type of product / treadmills aerofit provide to the customer
```

```
3 Different types of treadmill
```

```
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
```

```
b= df.describe()
b
```

	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

Next steps: Generate code with b View recommended plots New interactive sheet

```
# difference between mean and median
mean_table= b.loc['mean', 'Age' : 'Miles']

median_table = b.loc['50%' , 'Age': 'Miles']
difference = mean_table - median_table
difference.name = 'difference'
difference
```

	difference
Age	2.788889
Education	-0.427778
Usage	0.455556
Fitness	0.311111
Income	3123.077778
Miles	9.194444

dtype: float64

Non Graphical Analysis -

```
## Categories Wise average income of treadmills

df.groupby(['Product'])['Income'].mean()
#The highest income is of KP781 type of treadmill with an average income of 75441.575
```

Product	Income
KP281	46418.025
KP481	48973.650
KP781	75441.575

dtype: float64

```
soap  = df.groupby(['MaritalStatus'])['Product'].value_counts()
soap
```

MaritalStatus	Product	count
Partnered	KP281	48
	KP481	36
	KP781	23
Single	KP281	32
	KP481	24
	KP781	17

dtype: int64

```
# ###  finding the nuber of peeps whose marital status is single and they male and use whoch oproduct category

# Filter the DataFrame for 'Single' and 'Male'
filtered_df = df[(df['MaritalStatus'] == 'Single') & (df['Gender'] == 'Male')]

# Now, perform the groupby operation and get the value counts for 'Product'
apoo = filtered_df.groupby(['MaritalStatus'])['Product'].value_counts()

# You can now print the result
apoo
```

MaritalStatus	Product	count
Single	KP281	19
	KP781	14
	KP481	10

dtype: int64

```
df[df['MaritalStatus'] == 'Single'].value_counts()
#Out of 180 customers there is 73 customers whose marital status is single and rest
# 107 is in Partnered
```

Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles	count
KP281	18	Male	14	Single	3	4	29562	112	1
KP481	24	Female	14	Single	3	2	40932	85	1
	34	Male	15	Single	3	3	67083	85	1
	33	Female	18	Single	3	4	47754	74	1
	32	Male	16	Single	4	3	60261	127	1
...	...	...	...	...	...	...	...	...	...
KP281	32	Female	14	Single	3	4	46617	113	1
	31	Female	14	Single	2	2	45480	47	1
	30	Male	14	Single	3	3	54576	85	1
	28	Male	14	Single	4	3	54576	113	1
KP781	45	Male	16	Single	5	5	90886	160	1

73 rows × 1 columns

dtype: int64

```
## Gender Wise diversification

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

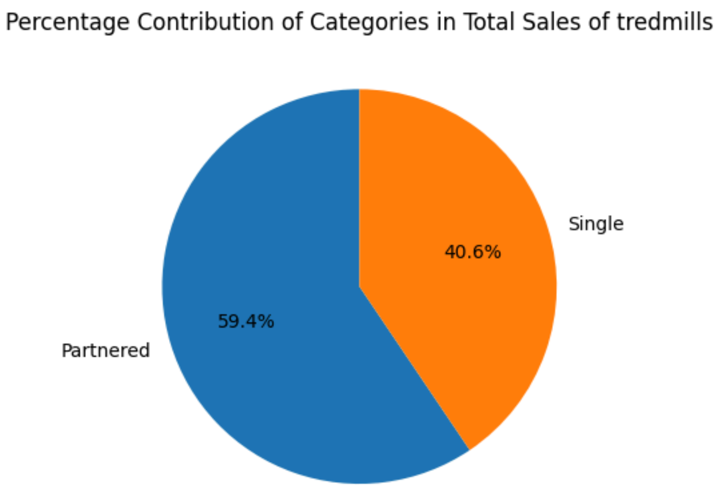
```
        count
Gender
  Male    104
Female     76

dtype: int64
```

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

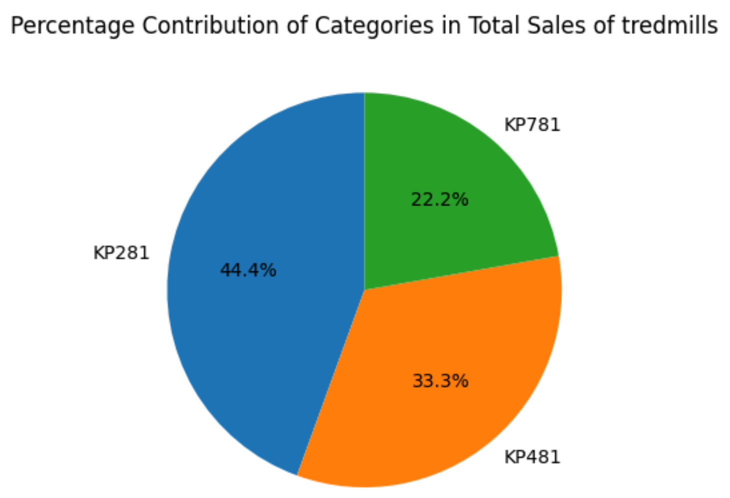
```
                                                                    count
Product Age Gender Education MaritalStatus Usage Fitness Income Miles
 KP281  19  Female    14      Partnered      4      3    30699   66      1
 KP481  40  Female    16      Partnered      3      3    61398   85      1
        38  Female    16      Partnered      4      3    62535   85      1
        37  Female    16      Partnered      2      3    48891   85      1
        35   Male     16      Partnered      3      3    53439   95      1
  ...   ...   ...     ...        ...        ...    ...     ...   ...    ...
 KP281  31   Male     14      Partnered      2      2    54576   47      1
        30   Male     14      Partnered      4      4    46617  141      1
        29   Male     18      Partnered      3      3    68220   85      1
            Female    16      Partnered      4      3    50028   94      1
 KP781  48   Male     18      Partnered      4      5    95508  180      1

107 rows × 1 columns

dtype: int64
```

Out of 180 customers there is 73 customers whose marital status is 'Single' and rest 107 is in 'Partnered' Category

```
import matplotlib.pyplot as plt

maroital_series = df['MaritalStatus'].value_counts()
maroital_series.plot(kind='pie', autopct='%1.1f%%', startangle=90)
plt.title('Percentage Contribution of Categories in Total Sales of tredmills')
plt.ylabel('')  # Hide the ylabel for better visualization
plt.show()
```



```
Product_series = df['Product'].value_counts()
Product_series.plot(kind='pie', autopct='%1.1f%%', startangle=90)
plt.title('Percentage Contribution of Categories in Total Sales of tredmills')
plt.ylabel('')  # Hide the ylabel for better visualization
plt.show()
```



Double-click (or enter) to edit

```
df
```

	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

Next steps: Generate code with df View recommended plots New interactive sheet

```
gender_wise = df.groupby(['Product'])['Age'].mean()
```

```
gender_wise
```

	Age
Product	
KP281	28.55
KP481	28.90
KP781	29.10

dtype: float64

```
df
```

	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

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
## number of person who are Single and male use which product


df1 = df[df['MaritalStatus'] == "Single"]
df2 = df1.groupby(['Product'])['Usage'].mean()
df2
```

	Usage
Product	
KP281	3.156250
KP481	3.083333
KP781	4.588235

dtype: float64

```
df2 = df[df['MaritalStatus'] == 'Partnered']
df3 = df2.groupby(['Product'])['Usage'].mean()
df3
## Mean of peoples whose marital status is partnered
```

	Usage
Product	
KP281	3.041667
KP481	3.055556
KP781	4.913043

dtype: float64

Univariant Analysis

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

```
array([18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,
       35, 36, 37, 38, 39, 40, 41, 43, 44, 46, 47, 50, 45, 48, 42])
```

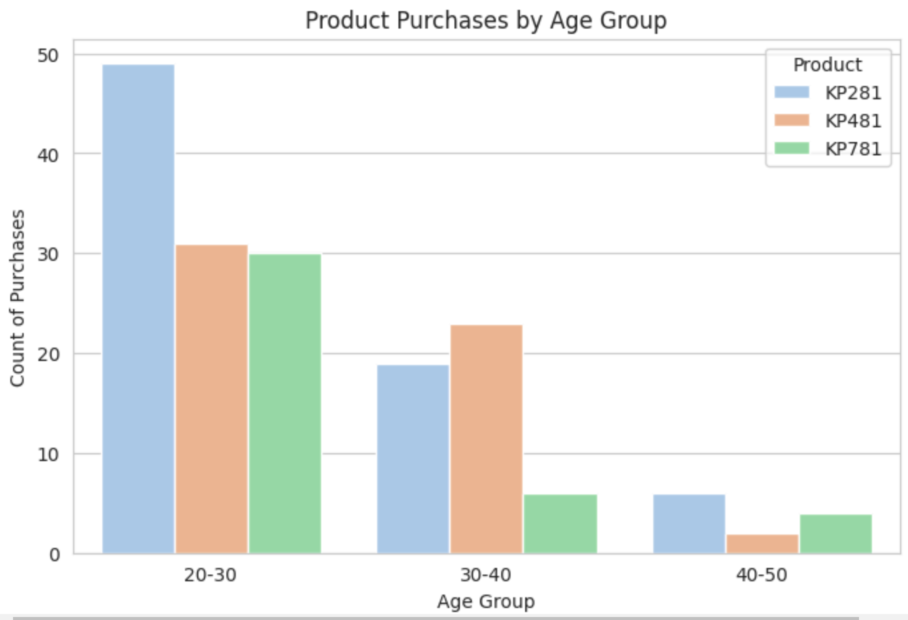
```
df['Age'].value_counts()
```

Age	count
25	25
23	18
24	12
26	12
28	9
35	8
33	8
30	7
38	7
21	7
22	7
27	7
31	6
34	6
29	6
20	5
40	5
32	4
19	4
48	2
37	2
45	2
47	2
46	1
50	1
18	1
44	1
43	1
41	1
39	1
36	1
42	1

dtype: int64

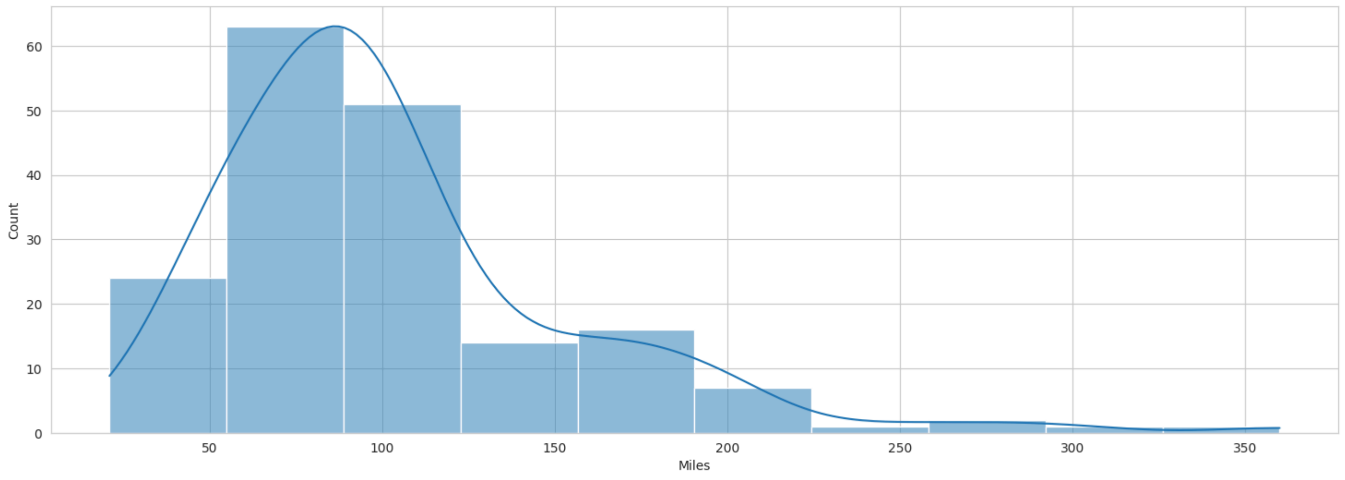
Bivariate Analysis

```
#Features like marital status, Gender, and age have any effect on the product purchased

crosstab_data= pd.crosstab([df['MaritalStatus'], df['Gender']], df['Product'])
bins = [20, 30, 40, 50]
labels = ['20-30', '30-40', '40-50']
df["AgeGroup"] = pd.cut(df["Age"], bins=bins, labels=labels)
plt.figure(figsize=(8, 5))
sns.countplot(x=df["AgeGroup"], hue=df["Product"], palette="pastel")
plt.title("Product Purchases by Age Group")
plt.xlabel("Age Group")
plt.ylabel("Count of Purchases")
plt.legend(title="Product")
plt.show()
```



Visual Analysis

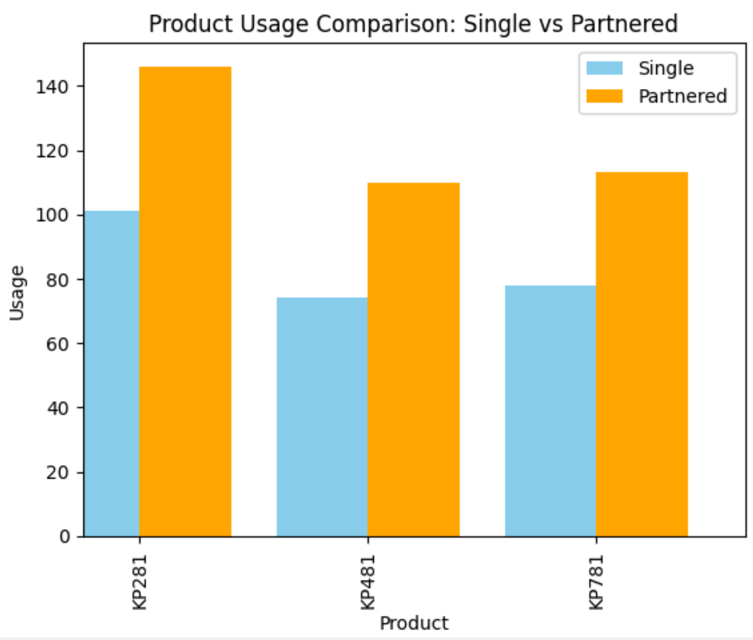
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.set_style('whitegrid')
plt.figure(figsize=(18,6))
sns.histplot(df['Miles'], kde=True, bins=10) # kde=True adds a smooth density curve
plt.show()
print("---> most customers average walk from 50 to 100 miles in a week ")
```



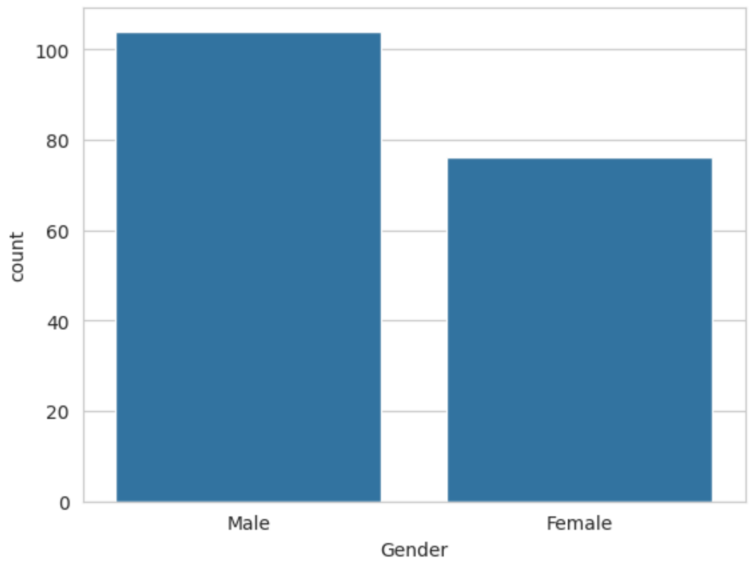
```
---> most customers average walk from 50 to 100 miles in a week
```

```
import matplotlib.pyplot as plt

# Filter the data for 'Single' and 'Partnered' individuals
df_single = df1[df1['MaritalStatus'] == 'Single']
df_partnered = df2[df2['MaritalStatus'] == 'Partnered']

# Group by 'Product' and sum the 'Usage' for each product in both categories
df_single_usage = df_single.groupby('Product')['Usage'].sum().reset_index()
df_partnered_usage = df_partnered.groupby('Product')['Usage'].sum().reset_index()

# Plot the bar chart for 'Single' individuals
ax = df_single_usage.plot(x='Product', y='Usage', kind='bar', color='skyblue', position=1, width=0.4, label='Single')

# Plot the bar chart for 'Partnered' individuals on the same axes
df_partnered_usage.plot(x='Product', y='Usage', kind='bar', color='orange', position=0, width=0.4, ax=ax, label='Partnered')

# Customize the chart
```

```
plt.xlabel('Product')
plt.ylabel('Usage')
plt.title('Product Usage Comparison: Single vs Partnered')

plt.legend()

# Show the plot
plt.show()
```



```
#count plot for Gender
import seaborn as sns
import matplotlib.pyplot as plt
sns.countplot(x=df['Gender']) #count plot for Gender
plt.show()
print("male customers are more than female")
```



```
male customers are more than female
```

Marginal Probability

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

	proportion
Gender	
Male	0.577778
Female	0.422222

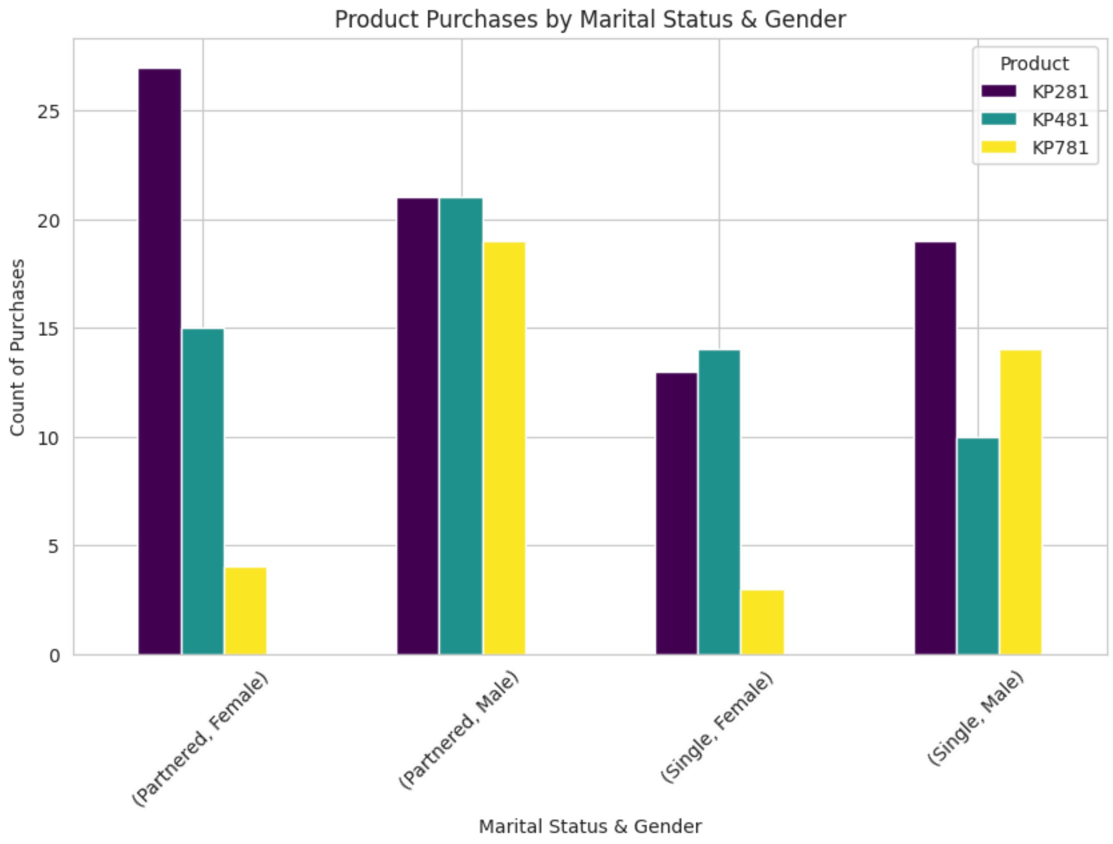
```
dtype: float64
```

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

	proportion
Product	
KP281	0.444444
KP481	0.333333
KP781	0.222222

```
dtype: float64
```

```
crosstab_df = pd.crosstab([df['MaritalStatus'], df['Gender']], df['Product'])
# Plot grouped bar chart
crosstab_data.plot(kind='bar', figsize=(10, 6), colormap='viridis')
plt.title("Product Purchases by Marital Status & Gender")
plt.xlabel("Marital Status & Gender")
plt.ylabel("Count of Purchases")
plt.xticks(rotation=45)
plt.legend(title="Product")
plt.show()
print("KP281 is maximum used by partenered female")
print("KP481 is maximum used by partenered male")
print("KP781 is maximum used by partenered male")
```



```
KP281 is maximum used by partenered female
KP481 is maximum used by partenered male
KP781 is maximum used by partenered male
```

Conditional Probability (Likelihood of One Event Given Another)

```
##Probability of Buying KP281 Given the Customer is Single

p_KP281_given_single = crosstab_df.loc[('Single', slice(None)), 'KP281'].sum() / crosstab_df.loc[('Single', slice(None)), :].sum()
```

```
print(p_KP281_given_single)
print("40% of Single customers prefer KP781")
```

```
Product
KP281    1.000000
KP481    1.333333
KP781    1.882353
dtype: float64
40% of Single customers prefer KP781
```

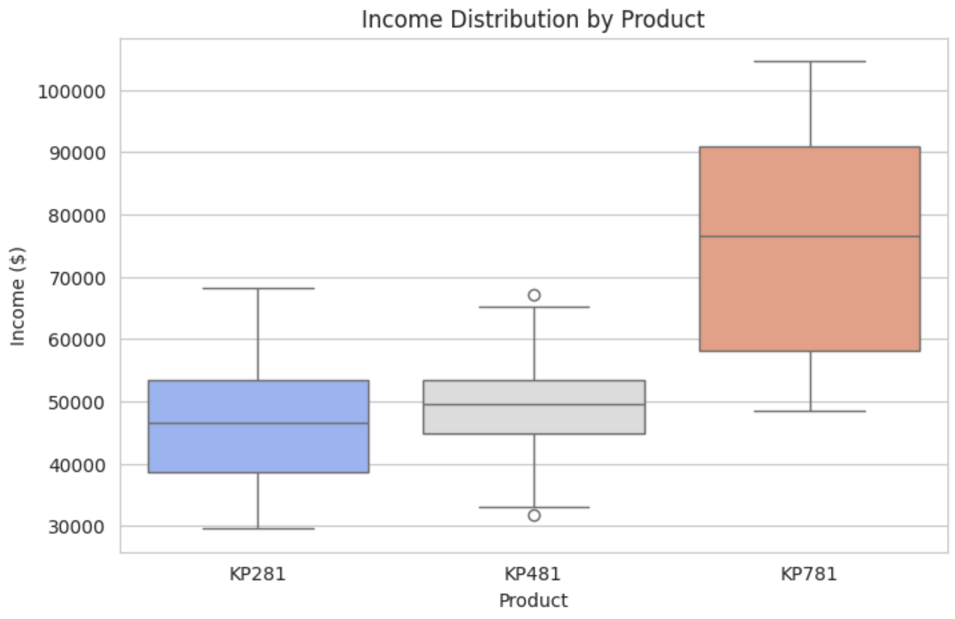
```
##Probability of Buying KP481 Given the Customer is Partnered
```

```
p_KP481_given_partnered = crosstab_df.loc[('Partnered', slice(None)), 'KP481'].sum() / crosstab_df.loc[('Partnered', slice(None))]
print(p_KP481_given_partnered)
print("30% of Partnered customers prefer KP781")
```

```
Product     KP281     KP481     KP781
Gender
Female   1.333333  2.400000  9.000000
Male     1.714286  1.714286  1.894737
30% of Partnered customers prefer KP781
```

```
plt.figure(figsize=(8, 5))
sns.boxplot(x="Product", y="Income", data=df, palette="coolwarm")
plt.title("Income Distribution by Product")
plt.xlabel("Product")
plt.ylabel("Income ($)")
plt.show()
print("KP281 buyers have lower income.")
print("KP781 is purchased by higher-income customers.")
```

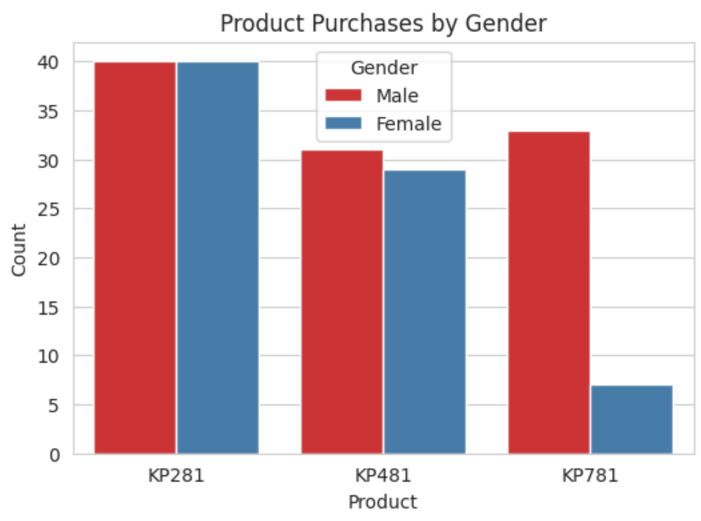
```
<ipython-input-67-955d5b71e188>:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `le

  sns.boxplot(x="Product", y="Income", data=df, palette="coolwarm")
```



```
KP281 buyers have lower income.
KP781 is purchased by higher-income customers.
```

Count of Purchase by Gender

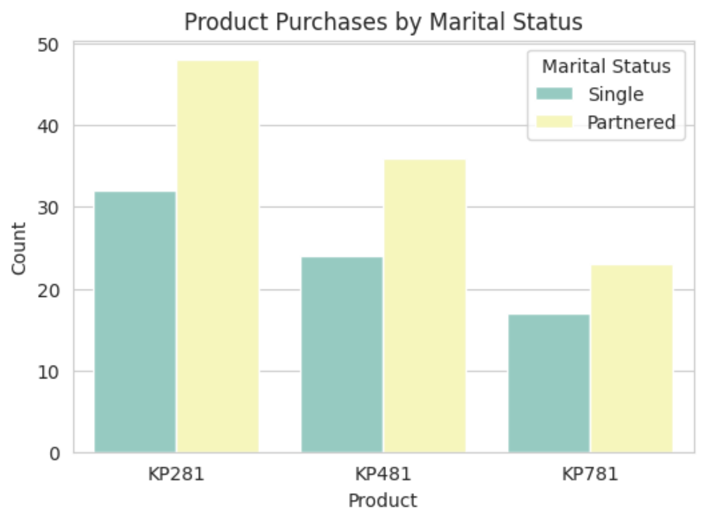
```
plt.figure(figsize=(6, 4))
sns.countplot(x="Product", hue="Gender", data=df, palette="Set1")
plt.title("Product Purchases by Gender")
plt.xlabel("Product")
plt.ylabel("Count")
plt.legend(title="Gender")
plt.show()
print("Males purchase more treadmills overall.")
print("KP781 has fewer female buyers.")
```



```
Males purchase more treadmills overall.
KP781 has fewer female buyers
```

Purchase by Marital Status

```
plt.figure(figsize=(6, 4))
sns.countplot(x="Product", hue="MaritalStatus", data=df, palette="Set3")
plt.title("Product Purchases by Marital Status")
plt.xlabel("Product")
plt.ylabel("Count")
plt.legend(title="Marital Status")
plt.show()
print("Single individuals purchase KP281 more.")
print("Married individuals buy KP481 and KP781 more.")
```

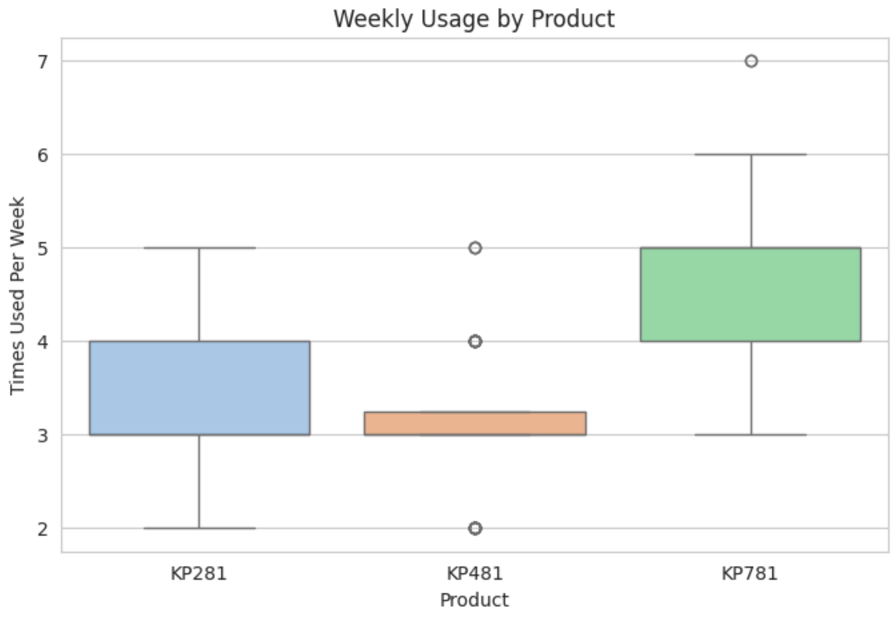


```
Single individuals purchase KP281 more.
Married individuals buy KP481 and KP781 more
```

Weekly Usage (Number of Times)

```
plt.figure(figsize=(8, 5))
sns.boxplot(x="Product", y="Usage", data=df, palette="pastel")
plt.title("Weekly Usage by Product")
plt.xlabel("Product")
plt.ylabel("Times Used Per Week")
plt.show()
print("KP781 users exercise more frequently compared to other users.")
```

```
<ipython-input-70-55b2384069f2>:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `le

  sns.boxplot(x="Product", y="Usage", data=df, palette="pastel")
```



```
KP781 users exercise more frequently compared to other users.
```

Customer Profile for each product

KP281 - Entry Level ($1,500)

*Younger customers (18-30 years old).

*Lower fitness level (1-3).

*Mostly purchased by single individuals.

*Used less frequently (2-3 times per week).

*More female buyers

KP481 - Mid-Level ($1,750)

Middle-aged (30-45 years old).

Moderate income level.

Moderate fitness (3-4).

Balanced between single & married buyers.

Used moderately (3-5 times per week).

More male buyers.

KP781 - High-End ($2,500)

Older customers (40-60 years old).

Higher income bracket.

Higher fitness level (4-5).

More married individuals.

Used frequently (5+ times per week).

Mostly male buyers.

✅ Insight:

Older customers tend to invest in premium treadmills.

Income vs. Product Purchased

KP281 → Mostly purchased by low-to-mid income customers (~$30K - $50K).

KP481 → Bought by mid-income customers (~$50K - $80K).

KP781 → Strong preference among high-income customers (~$80K+).

✅ Insight:

Higher-income customers are willing to pay for premium features.

Miles vs. Product Purchased

KP281 buyers → Run fewer miles per week (~5-10 miles).

KP481 buyers → Run moderate distances (~10-15 miles).

KP781 buyers → Run higher distances (~15+ miles).

✅ Insight:

Customers who run more miles prefer high-end treadmills.

Fitness Level vs. Product Purchased

KP281 buyers → Fitness levels mostly 1-3 (low-to-average fitness).

KP481 buyers → Fitness levels around 3-4 (moderate fitness).

KP781 buyers → Fitness levels 4-5 (high fitness).

Business Recommendations

✅Target Younger Customers for Frequent Use

Since younger people use the treadmill more often and run more miles, AeroFit can market high-end treadmills (KP781) to them.

✅ Encourage Older Customers with Personalized Plans

Since older individuals use treadmills less, AeroFit can introduce senior-friendly fitness programs or offer incentives to increase treadmill us

✅ Fitness-Based Product Segmentation

Customers with higher fitness levels run more miles, so AeroFit can recommend advanced treadmills (KP781) with endurance features to them.

✅ Income-Based Promotions

High-income groups don't necessarily have better fitness, so promotions should focus on lifestyle benefits rather than price discounts.

Target Younger & Low-Income Customers for KP281

Offer budget-friendly financing or installment plans to attract first-time treadmill buyers.

🔷 Market KP481 to Middle-Aged, Mid-Income Customers Highlight balanced features (not too expensive, but durable).

🔷 Promote KP781 to High-Income & Fitness Enthusiasts Use premium branding and emphasize high performance & durability.

🔷 Encourage Long-Term Usage Provide customized workout programs based on fitness levels.

✅ Fitness-Based Product Segmentation

Customers with higher fitness levels run more miles, so AeroFit can recommend advanced treadmills (KP781) with endurance features to them.

✅ Income-Based Promotions

High-income groups don't necessarily have better fitness, so promotions should focus on lifestyle benefits rather than price discounts.

Target Younger & Low-Income Customers for KP281

Offer budget-friendly financing or installment plans to attract first-time treadmill buyers.

🔷 Market KP481 to Middle-Aged, Mid-Income Customers Highlight balanced features (not too expensive, but durable).

🔷 Promote KP781 to High-Income & Fitness Enthusiasts Use premium branding and emphasize high performance & durability.