In [ ]:
```
!gdown https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/125/origi
```

```
Downloading...
From: https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/125/origi
nal/aerofit_treadmill.csv?1639992749
To: /content/aerofit_treadmill.csv?1639992749
100% 7.28k/7.28k [00:00<00:00, 19.3MB/s]
```

In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_csv("/content/aerofit_treadmill.csv?1639992749")
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 [ ]:
```
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 [ ]:
```
df.columns
```

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

In [ ]:
```
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
```

In [ ]:
```
df.shape
```

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

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

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.isnull().sum()
```

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

In [ ]:
```
product_counts = df['Product'].value_counts()
print("Value counts for 'Product Purchased' column:")
print(product_counts)
unique_products = df['Product'].unique()
print("\nUnique attributes for 'Product Purchased' column:")
print(unique_products)
```

```
Value counts for 'Product Purchased' column:
Product
KP281    80
KP481    60
KP781    40
Name: count, dtype: int64

Unique attributes for 'Product Purchased' column:
['KP281' 'KP481' 'KP781']
```

In [ ]:

```
product_counts = df['Age'].value_counts()
print("Value counts for 'Age' column:")
print(product_counts)
unique_products = df['Age'].unique()
print("\nUnique attributes for 'Age' column:")
print(unique_products)
```

```
Value counts for 'Age' column:
Age
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
Name: count, dtype: int64

Unique attributes for 'Age' column:
[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]
```

In [ ]:

```
product_counts = df['Gender'].value_counts()
print("Value counts for 'Gender' column:")
print(product_counts)
unique_products = df['Gender'].unique()
print("\nUnique attributes for 'Gender' column:")
print(unique_products)
```

```
Value counts for 'Gender' column:
Gender
Male      104
Female     76
Name: count, dtype: int64

Unique attributes for 'Gender' column:
['Male' 'Female']
```

In [ ]:
```
product_counts = df['MaritalStatus'].value_counts()
print("Value counts for 'Marital Status' column:")
print(product_counts)
unique_products = df['MaritalStatus'].unique()
print("\nUnique attributes for 'Marital Status' column:")
print(unique_products)
```

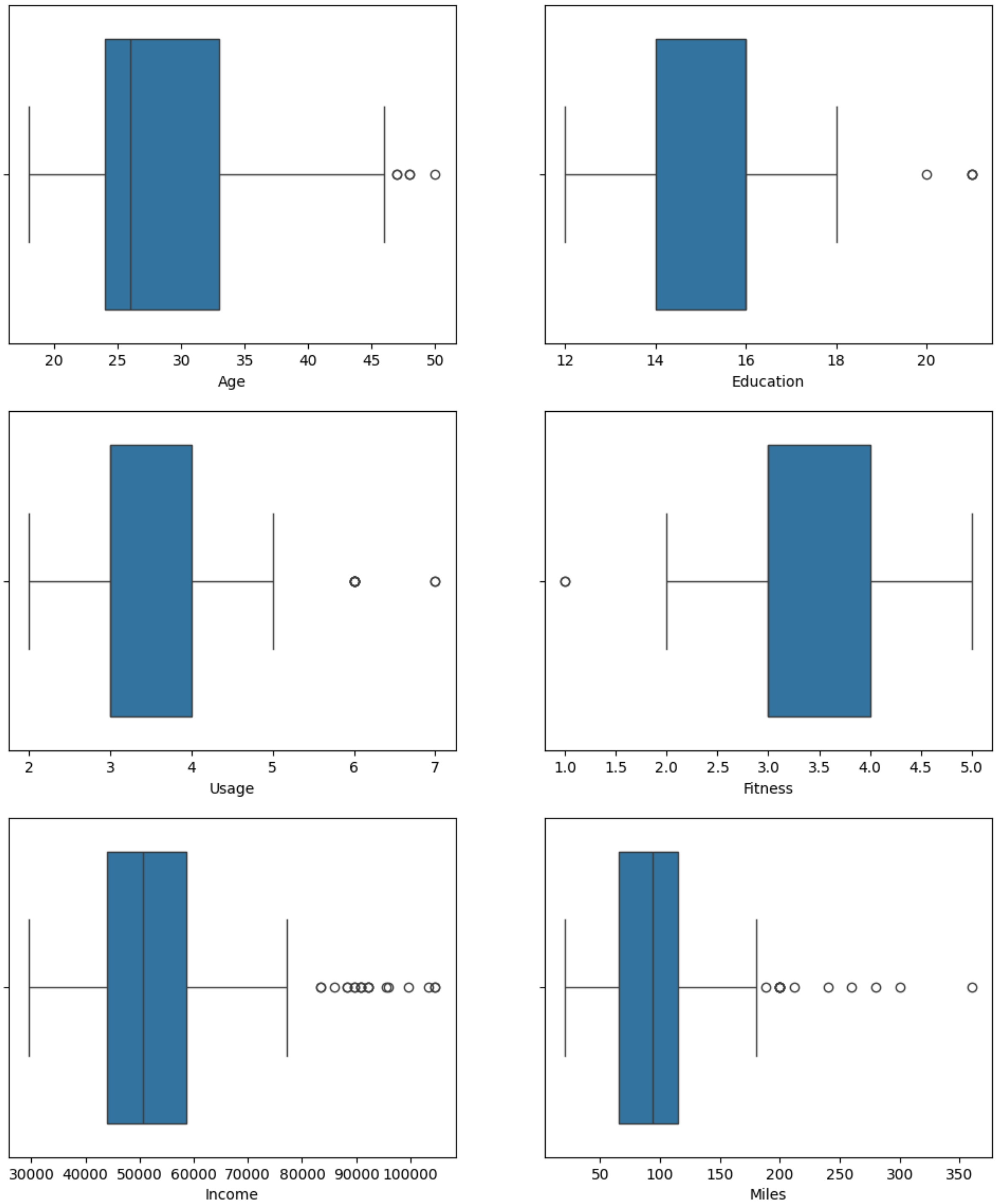
```
Value counts for 'Marital Status' column:
MaritalStatus
Partnered    107
Single        73
Name: count, dtype: int64

Unique attributes for 'Marital Status' column:
['Single' 'Partnered']
```

Outliers detection using BoxPlots

In [ ]:
```
fig, axis = plt.subplots(nrows=3, ncols=2, figsize=(12, 10))
fig.subplots_adjust(top=1.2)

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()
```



Obervation

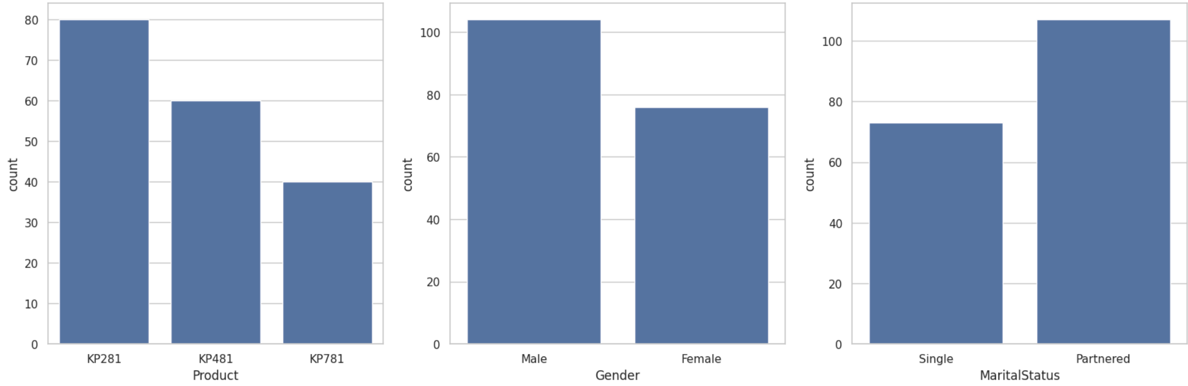
Even from the boxplots it is quite clear that:

Age, Education and Usage are having very few outliers. While Income and Miles are having more outliers.

Distribution of the data for the qualitative attributes

In [ ]:
```
fig, axs = plt.subplots(nrows=1, ncols=3, figsize=(20, 6))
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])
plt.show()
```

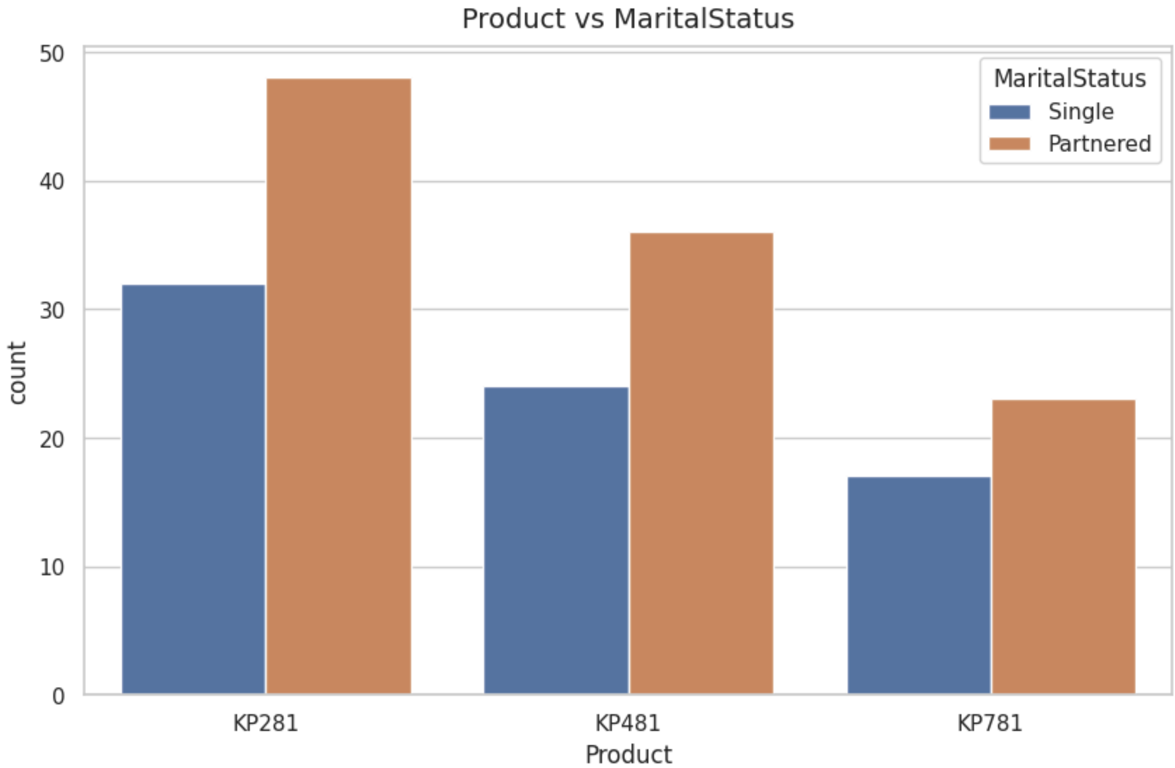


Obervations

KP281 is the most frequent product. Thare are more Males in the data than Females. More Partnered persons are there in the data.

Does Age or MaritalStatus have any effect on the product purchased.

In [ ]:
```
sns.set_style(style='whitegrid')
fig, axs = plt.subplots(nrows=1, ncols=1, figsize=(10, 6))
sns.countplot(data=df, x='Product', hue='MaritalStatus')
plt.title("Product vs MaritalStatus", pad=10, fontsize=14)
plt.show()
```



Product vs MaritalStatus

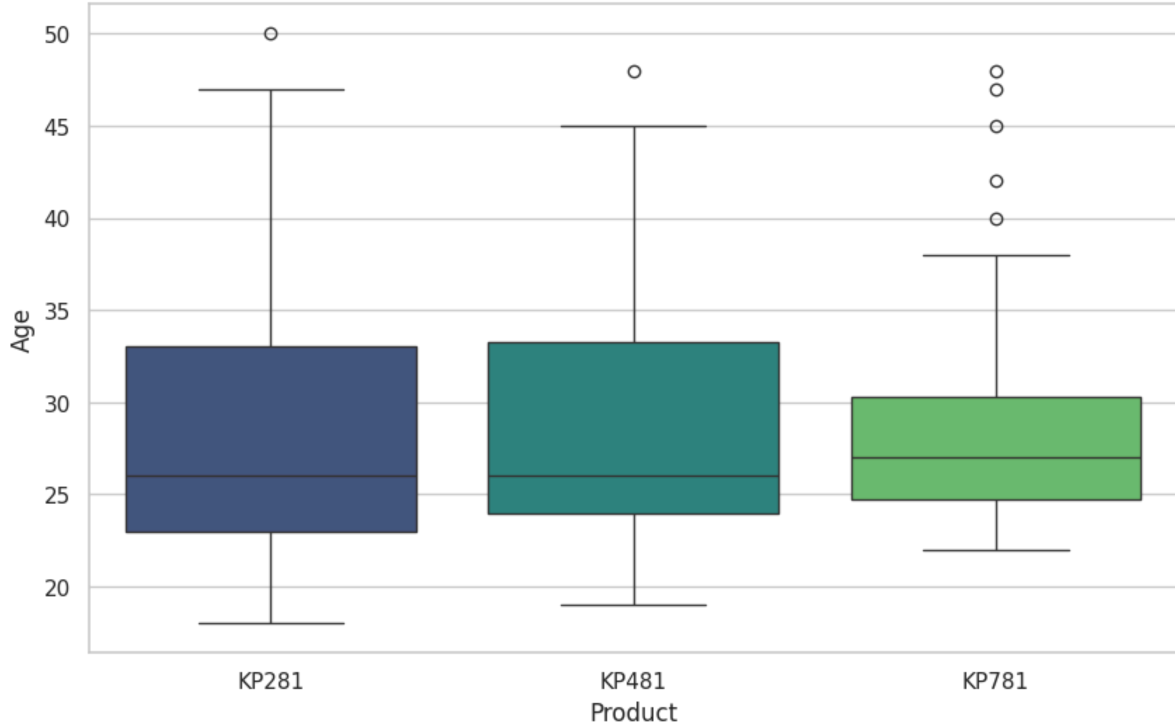
Customer who is Partnered, is more likely to purchase the product.

In [ ]:
```
plt.figure(figsize=(10, 6))
sns.boxplot(x='Product', y='Age', data=df, palette='viridis')
plt.show()
```

```
<ipython-input-53-e917bedd2c03>: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 `legend=False` for the same effect.

  sns.boxplot(x='Product', y='Age', data=df, palette='viridis')
```



Observation

The boxplot suggests that there are variations in the age distribution among different products purchased, indicating potential differences in the age demographics of customers for each product.

What percent of customers have purchased KP281, KP481, or KP781 products

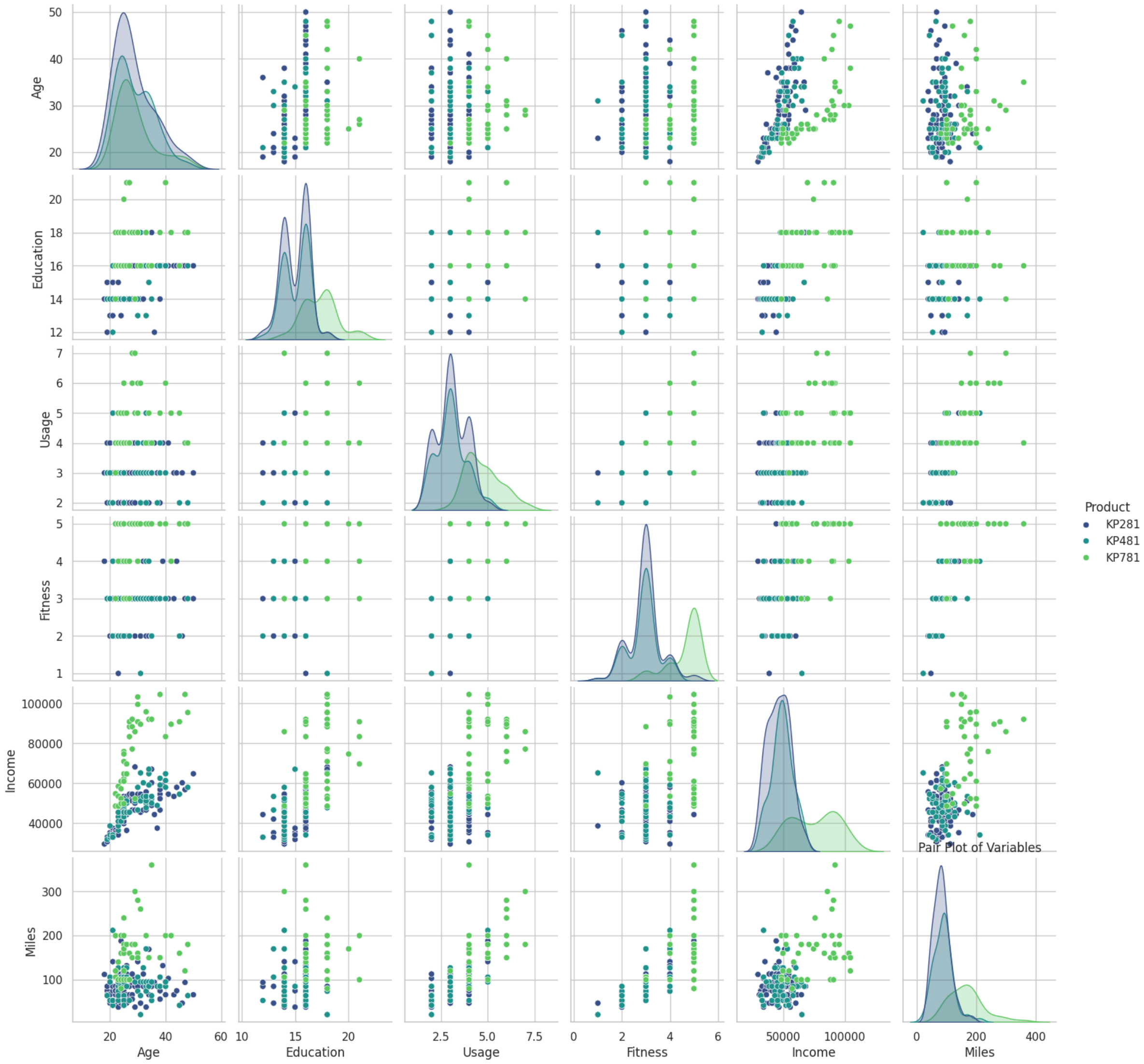
In [ ]:
```
contingency_table = pd.crosstab(index=df['Product'], columns='Count', normalize='co
contingency_table.columns = ['Percentage']
contingency_table = contingency_table.sort_values(by='Percentage', ascending=False)
print("Marginal Probability of Product Purchases:")
print(contingency_table)
```

```
Marginal Probability of Product Purchases:
         Percentage
Product
KP281     44.444444
KP481     33.333333
KP781     22.222222
```

Correlation among differet factors using pair plot

In [ ]:
```
# Create pair plot
plt.figure(figsize=(12,10))
sns.pairplot(df, diag_kind='kde', hue='Product', palette='viridis')
plt.title('Pair Plot of Variables')
plt.show()
```

```
<Figure size 1200x1000 with 0 Axes>
```



Observation

The pair plot showcases distinct groupings and trends among variables, implying potential differences in customer behavior and preferences across different products.

Probability of a male customer buying a KP781 treadmill

In [ ]:
```
male_customers_df = df[df['Gender'] == 'Male']
male_kp781_count = male_customers_df[male_customers_df['Product'] == 'KP781'].shape
total_male_customers = male_customers_df.shape[0]
probability_male_kp781 = male_kp781_count / total_male_customers
print("Probability of a male customer buying a KP781 treadmill:", probability_male_
```

```
Probability of a male customer buying a KP781 treadmill: 0.3173076923076923
```

Insights:

Popular Products: Product KP781 appears to be the most purchased among customers. Demographic Influence: Marital status and gender may have an influence on product preferences. Further analysis is needed to understand this relationship better. Age Distribution: The age distribution of customers varies, with a significant number of customers falling in the younger age groups. Channel Preference: Understanding customers' preferred channels for purchasing and communication is essential for effective marketing strategies. High-Value Customers: Identifying high-value customers through RFM analysis can help prioritize marketing efforts and retention strategies.

Recommendations:

Tailor marketing campaigns to target specific customer segments based on demographics, behaviors, and preferences.

Promote product KP781 aggressively due to its popularity, but also explore opportunities to upsell or cross-sell complementary products.

Engage with customers through their preferred channels, such as online, email, or social media, to increase brand visibility and engagement.

Implement strategies to retain high-value customers by offering personalized experiences, loyalty rewards, and exceptional customer service.

Use customer feedback and market insights to inform product development and innovation, focusing on meeting customer needs and preferences.

Monitor competitors' offerings and pricing strategies to stay competitive in the market and identify opportunities for differentiation.

Regularly analyze customer data and feedback to identify trends, opportunities, and areas for improvement in products and services.