

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

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

```
Download...
From: https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/125/original/aerofit_treadmill.csv?1639992749
To: /content/aerofit_treadmill.csv?1639992749
100% 7.28k/7.28k [00:00<00:00, 22.7MB/s]
```

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

```
df.head()
```

	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

```
df.tail()
```

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
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

```
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.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.dtypes
```



	0
Product	object
Age	int64
Gender	object
Education	int64
MaritalStatus	object
Usage	int64
Fitness	int64
Income	int64
Miles	int64

```
dtype: object
```

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



	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
count	180	180.000000	180	180.000000	180	180.000000	180.000000	180.000000	180.000000
unique	3	NaN	2	NaN	2	NaN	NaN	NaN	NaN
top	KP281	NaN	Male	NaN	Partnered	NaN	NaN	NaN	NaN
freq	80	NaN	104	NaN	107	NaN	NaN	NaN	NaN
mean	NaN	28.788889	NaN	15.572222	NaN	3.455556	3.311111	53719.577778	103.194444
std	NaN	6.943498	NaN	1.617055	NaN	1.084797	0.958869	16506.684226	51.863605
min	NaN	18.000000	NaN	12.000000	NaN	2.000000	1.000000	29562.000000	21.000000
25%	NaN	24.000000	NaN	14.000000	NaN	3.000000	3.000000	44058.750000	66.000000
50%	NaN	26.000000	NaN	16.000000	NaN	3.000000	3.000000	50596.500000	94.000000
75%	NaN	33.000000	NaN	16.000000	NaN	4.000000	4.000000	58668.000000	114.750000
max	NaN	50.000000	NaN	21.000000	NaN	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['Gender'].value_counts()
```

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

	count
Gender	
Male	104
Female	76

dtype: int64

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

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

	count
Product	
KP281	80
KP481	60
KP781	40

dtype: int64

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

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

	count
MaritalStatus	
Partnered	107
Single	73

dtype: int64

```
df.nunique()
```

```
df.nunique()
```

	0
Product	3
Age	32
Gender	2
Education	8
MaritalStatus	2
Usage	6
Fitness	5
Income	62
Miles	37

dtype: int64

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

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

array(['KP281', 'KP481', 'KP781'], dtype=object)

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

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


array(['Male', 'Female'], dtype=object)

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

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

array(['Single', 'Partnered'], dtype=object)

```
sns.distplot(df['Age'])
```

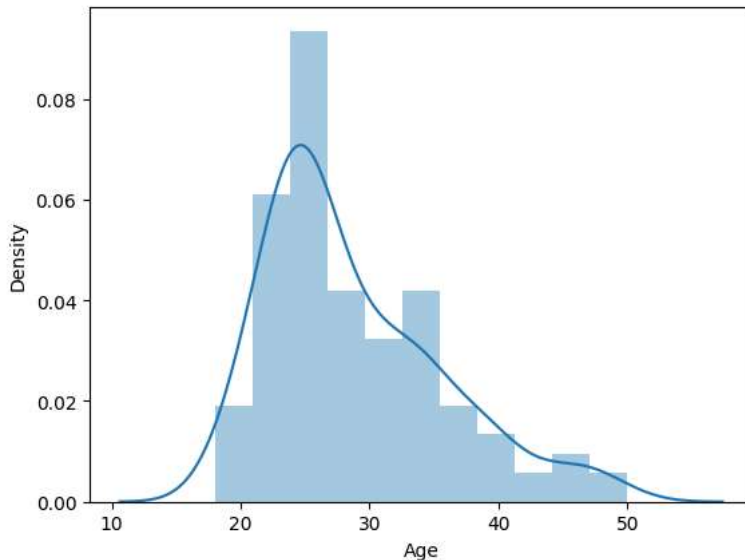
 /tmp/ipython-input-3255828239.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.


Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see <https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751>

```
sns.distplot(df['Age'])
<Axes: xlabel='Age', ylabel='Density'>
```



```
sns.distplot(df['Education'], kde = True)
```

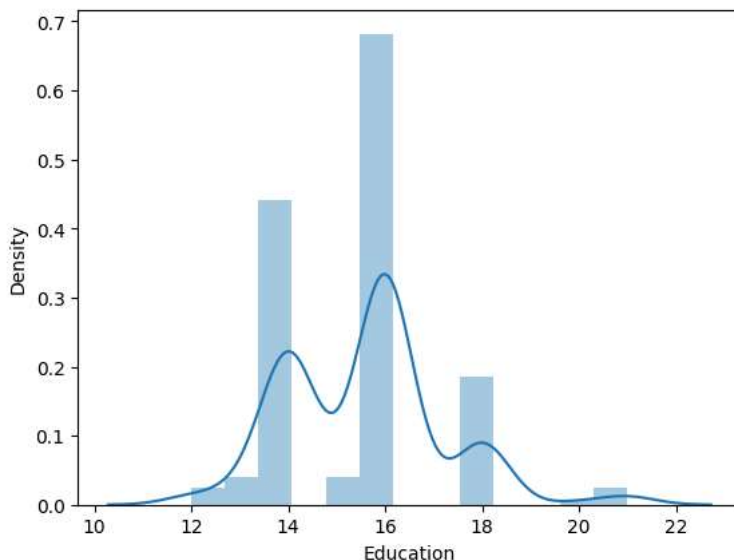
 /tmp/ipython-input-3299290954.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

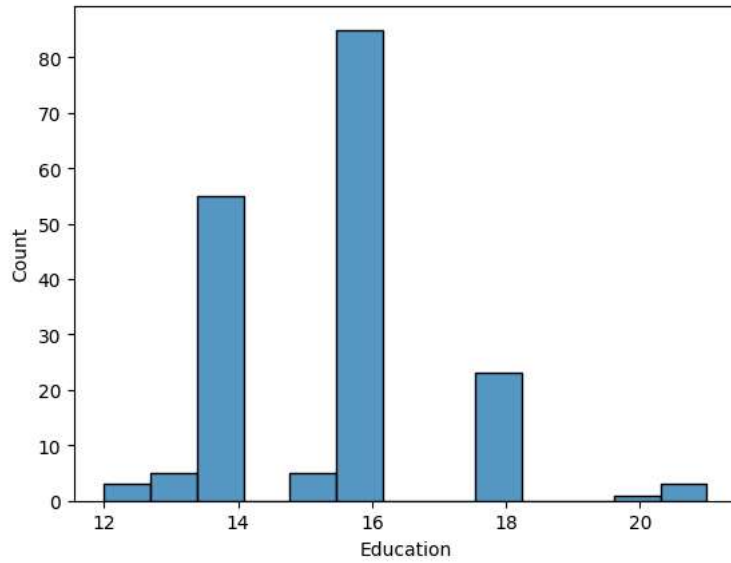
For a guide to updating your code to use the new functions, please see <https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751>

```
sns.distplot(df['Education'], kde = True)
<Axes: xlabel='Education', ylabel='Density'>
```




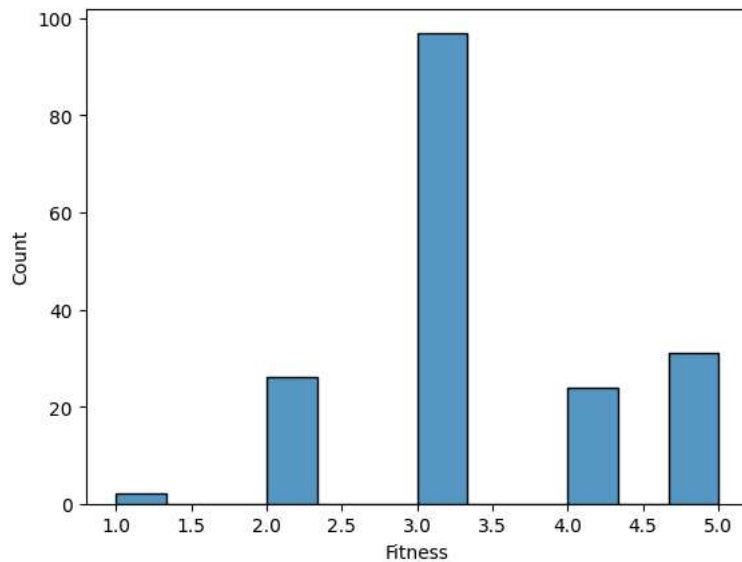
```
sns.histplot(df['Education'])
```

 <Axes: xlabel='Education', ylabel='Count'>



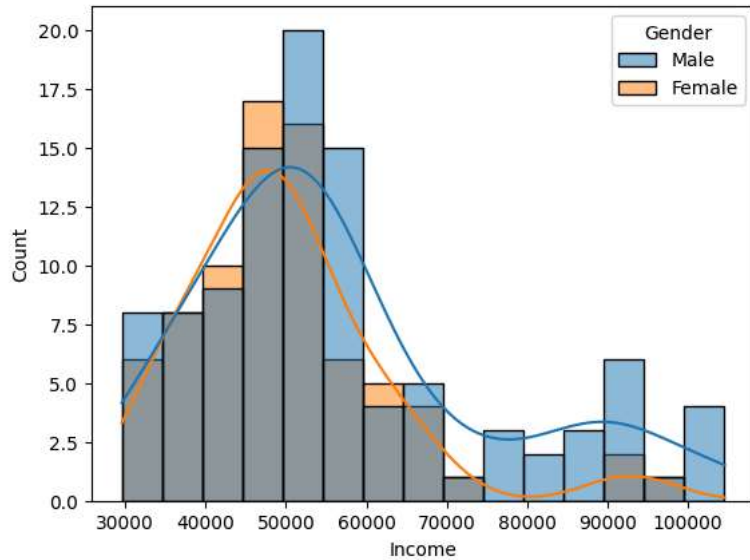
```
sns.histplot(df['Fitness'])
```

 <Axes: xlabel='Fitness', ylabel='Count'>



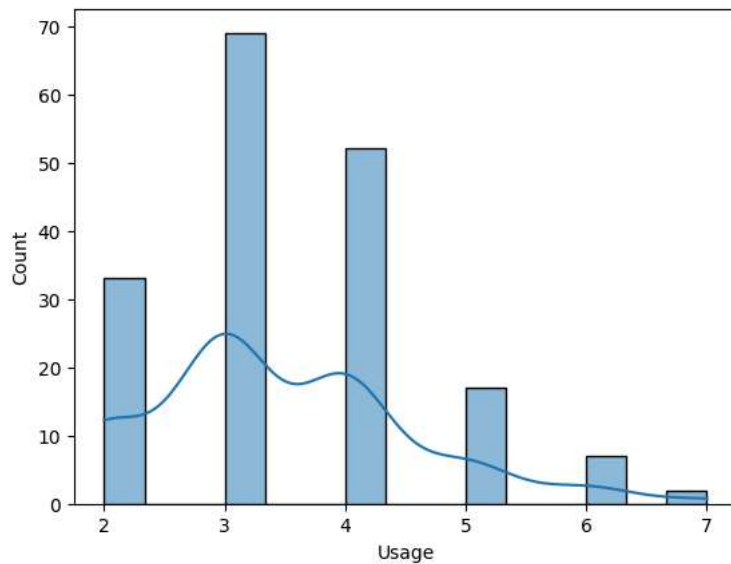
```
sns.histplot(x = 'Income',data=df,kde=True,hue='Gender')
```

<Axes: xlabel='Income', ylabel='Count'>



```
sns.histplot(df['Usage'],kde=True)
```

<Axes: xlabel='Usage', ylabel='Count'>



```
fig, axs=plt.subplots(nrows=1,ncols=3, figsize=(15,5))
sns.countplot(x='Gender',data=df,ax=axs[0], palette='coolwarm')
sns.countplot(x= 'MaritalStatus',data=df,ax=axs[1],palette='plasma')
sns.countplot(x='Product',data= df,ax=axs[2],palette='Set1')
```

```
axs[0].set_title('Product Count')
axs[1].set_title('Gender Count')
axs[2].set_title('MaritalStatus Count')
```

```
plt.show()
plt.tight_layout()
```

```
/tmp/ipython-input-3043894200.py: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 `legenc

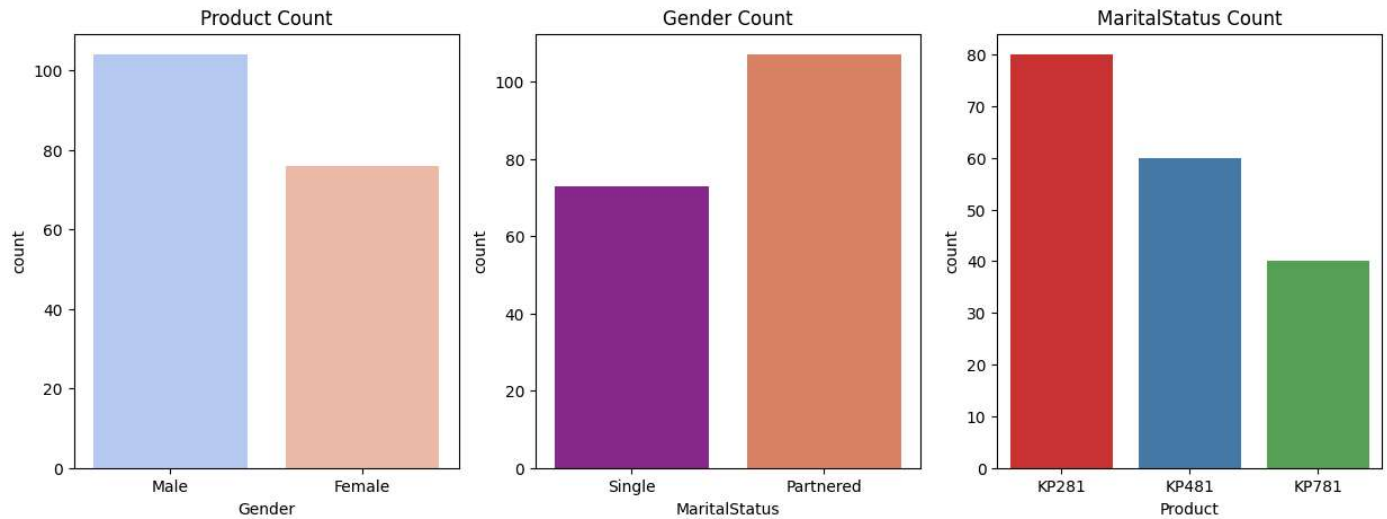
```
sns.countplot(x='Gender',data=df,ax=axes[0], palette='coolwarm')
/tmp/ipython-input-3043894200.py:3: FutureWarning:
```

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

```
sns.countplot(x= 'MaritalStatus',data=df,ax=axes[1],palette='plasma')
/tmp/ipython-input-3043894200.py:4: FutureWarning:
```

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

```
sns.countplot(x='Product',data= df,ax=axes[2],palette='Set1')
```



<Figure size 640x480 with 0 Axes>

- Observation
- Product Count (actually showing Gender distribution)

Male count is higher ( $\approx 105$ ).

Female count is lower ( $\approx 75$ ). 📌 This means your dataset has more male customers than female customers.

Gender Count (actually showing Marital Status distribution)

Partnered individuals are higher ( $\approx 107$ ).

Single individuals are fewer ( $\approx 73$ ). 📌 So, most customers are partnered rather than single.


Marital Status Count (actually showing Product distribution)

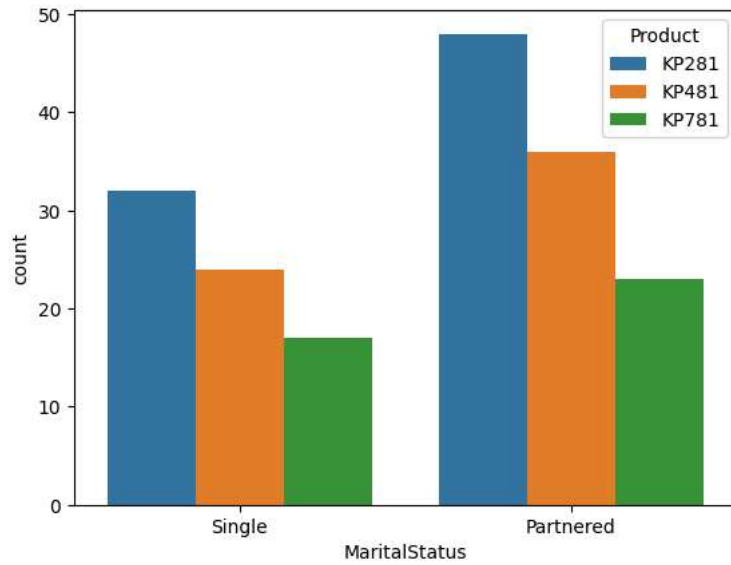
Product KP281 has the highest count ( $\approx 80$ ).

Product KP481 is next ( $\approx 60$ ).


Product KP781 is the lowest ( $\approx 40$ ). 📌 Meaning, KP281 is the most popular product, while KP781 is the least popular.

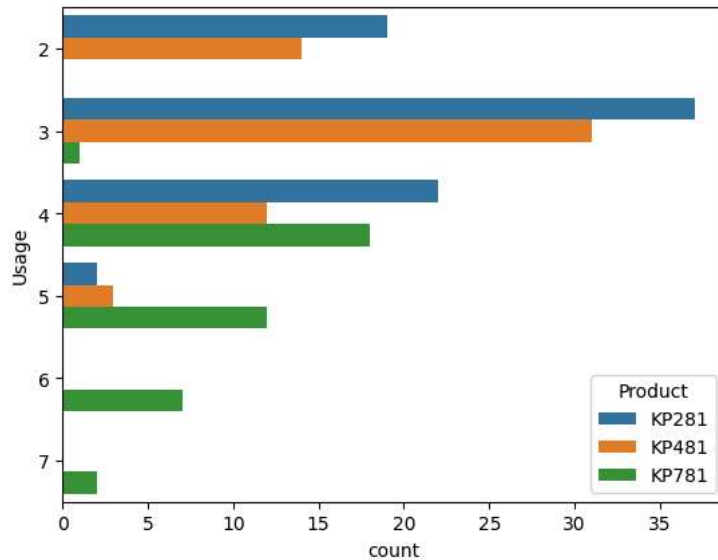
```
sns.countplot(data=df,x='MaritalStatus',hue='Product')
```

 <Axes: xlabel='MaritalStatus', ylabel='count'>



```
sns.countplot(data=df, y='Usage', hue='Product')
```

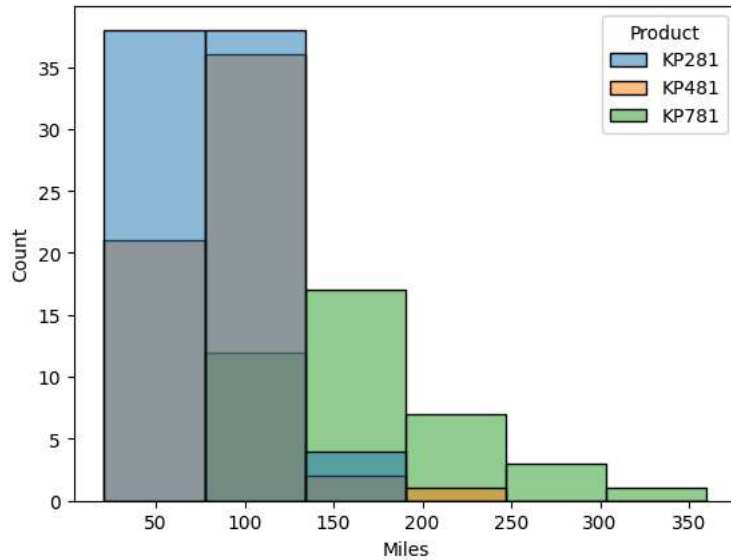
 <Axes: xlabel='count', ylabel='Usage'>



```
sns.histplot(data=df, x='Miles', hue='Product', bins=7)
```

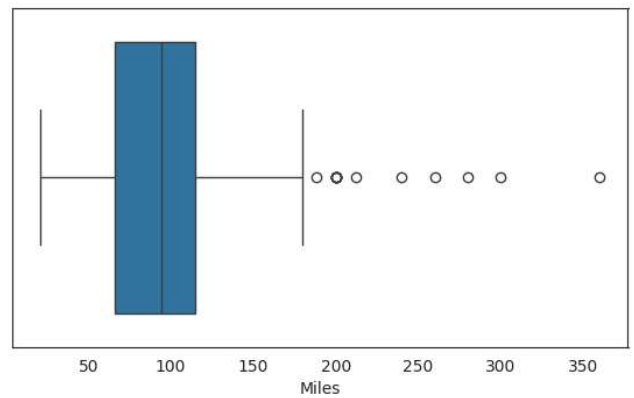
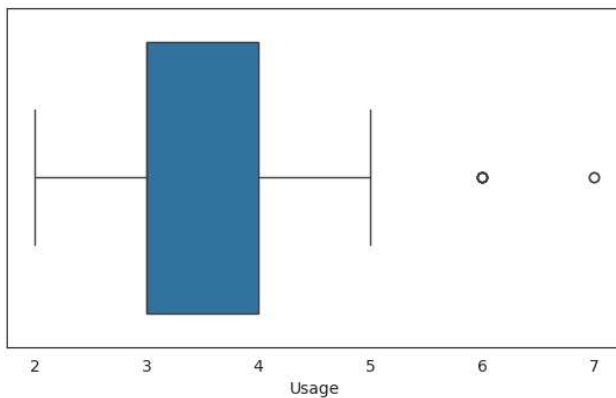
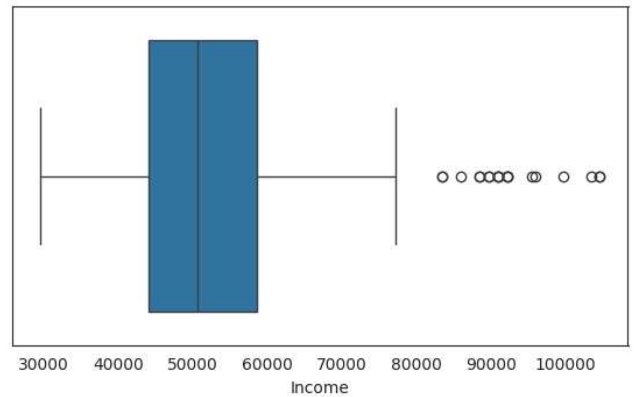
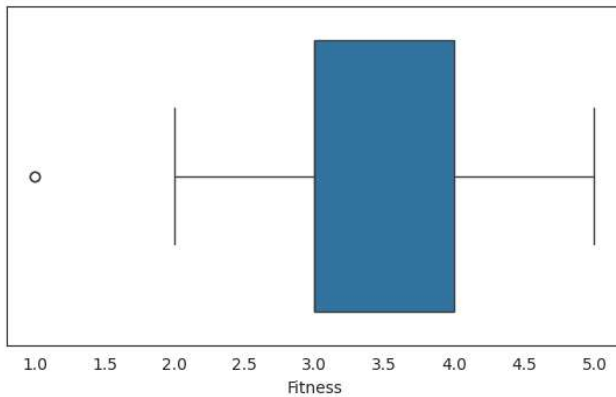
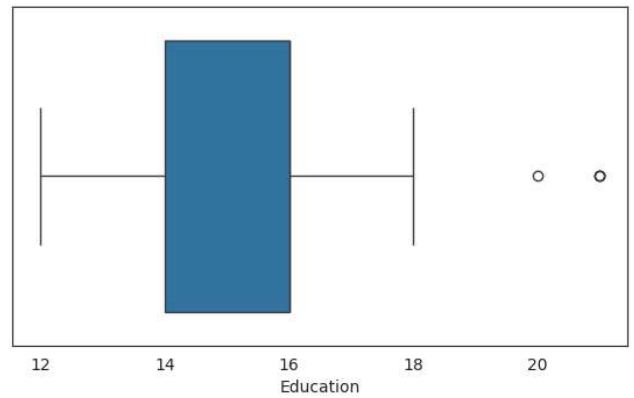
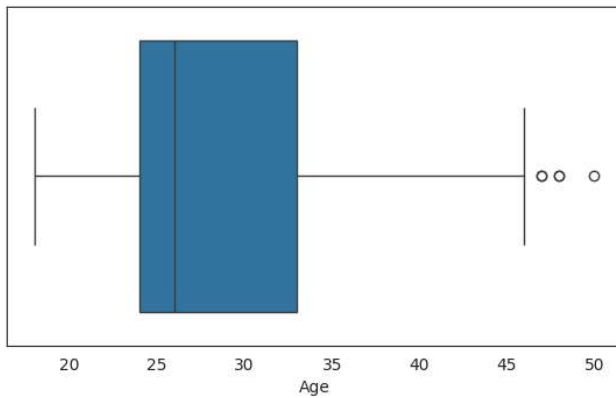


<Axes: xlabel='Miles', ylabel='Count'>



```
fig, axis = plt.subplots(nrows=3, ncols=2, figsize=(15,10))
fig.subplots_adjust(top=1.1)
```

```
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='Fitness', orient='h', ax=axis[1,0])
sns.boxplot(data=df, x='Income', orient='h', ax=axis[1,1])
sns.boxplot(data=df, x='Usage', orient='h', ax=axis[2,0])
sns.boxplot(data=df, x='Miles', orient='h', ax=axis[2,1])
plt.show()
```



#### Observations from the Boxplots Left Column (Usage & another variable)

Top-left (Variable ~20–50 range, maybe Age or similar)

Median  $\approx$  26–28.

Most values lie between 20 and 35.

A few outliers above 45–50. 📌 Data is slightly right-skewed with high-value outliers.

Middle-left (Usage 1–5)

Median  $\approx$  3.

Data is fairly symmetric.

One very low outlier ( $\approx$ 1). 📌 Most users fall in the 2–4 range.

Bottom-left (Usage 2–7)

Median  $\approx$  3–3.5.

Majority of data in 3–5.

Some high outliers above 6. 📌 A few heavy users compared to the majority.

Right Column (Miles)

Top-right (12–21 range, maybe Hours/Time)

Median  $\approx$  15.

Range 13–18 is common.

A couple of high outliers near 20. 📌 Most data tightly clustered around the center.

Middle-right (Miles 30,000–100,000)

Median  $\approx$  55,000.

IQR  $\approx$  45,000–65,000.

Many outliers above 80,000–100,000. 📌 Indicates a long right tail — some customers use extremely high miles.

Bottom-right (Miles 0–350)

Median  $\approx$  100.

IQR  $\approx$  50–150.

Several outliers above 200–350. 📌 Majority travel in shorter ranges, but a few travel much higher miles.

### ✅ Overall Insights

Usage variables: Mostly concentrated in the middle (2–5), with a few extreme users.

Miles: Highly skewed with many large outliers, meaning most users drive moderate distances but some travel exceptionally high.

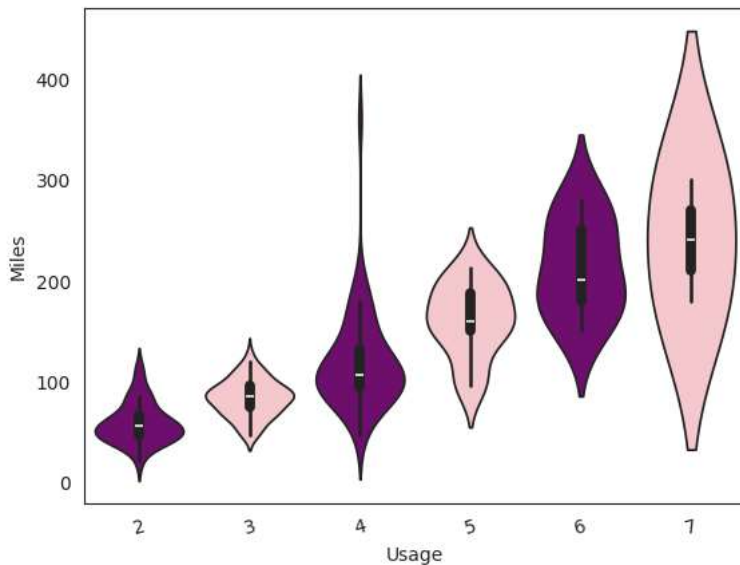
Age, Education and Usage are having very few outliers. While income and miles are having more outliers.

```
sns.violinplot(data=df, x='Usage', y='Miles', palette=['purple','pink'])
plt.xticks(rotation=15) # rotate x-axis labels
plt.show()
```


🔄 /tmp/ipython-input-2431158018.py:1: 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`

```
sns.violinplot(data=df, x='Usage', y='Miles', palette=['purple','pink'])
/tmp/ipython-input-2431158018.py:1: UserWarning:
The palette list has fewer values (2) than needed (6) and will cycle, which may produce an uninterpretable plot.
sns.violinplot(data=df, x='Usage', y='Miles', palette=['purple','pink'])
```

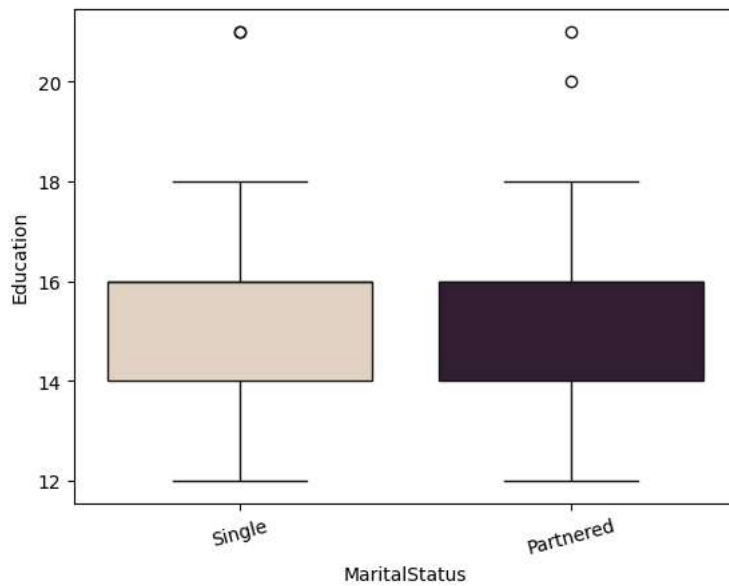


```
sns.boxplot(data=df, x='MaritalStatus', y='Education', palette='ch:s=.25')
plt.xticks(rotation=15) # rotate x-axis labels
```

 /tmp/ipython-input-2307146778.py:1: 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`

```
sns.boxplot(data=df, x='MaritalStatus', y='Education', palette='ch:s=.25')
([0, 1], [Text(0, 0, 'Single'), Text(1, 0, 'Partnered')])
```



```
sns.set_style(style='whitegrid')
fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(15,6.5))

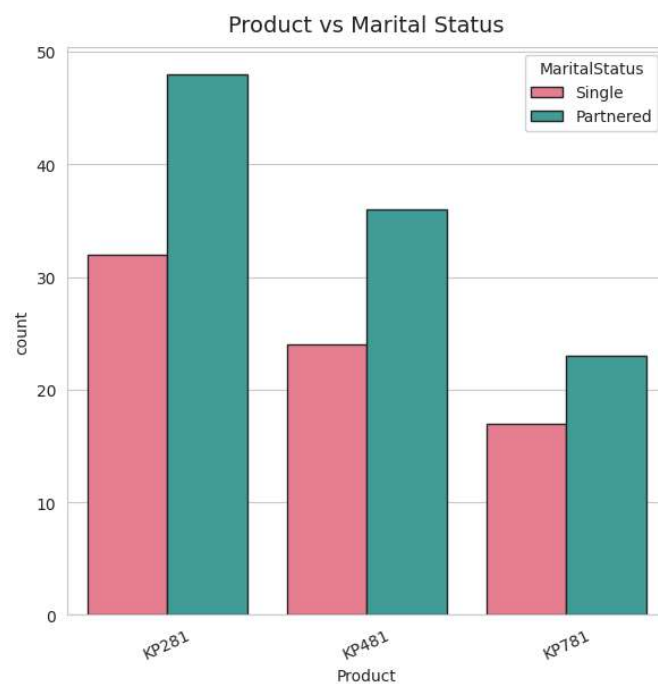
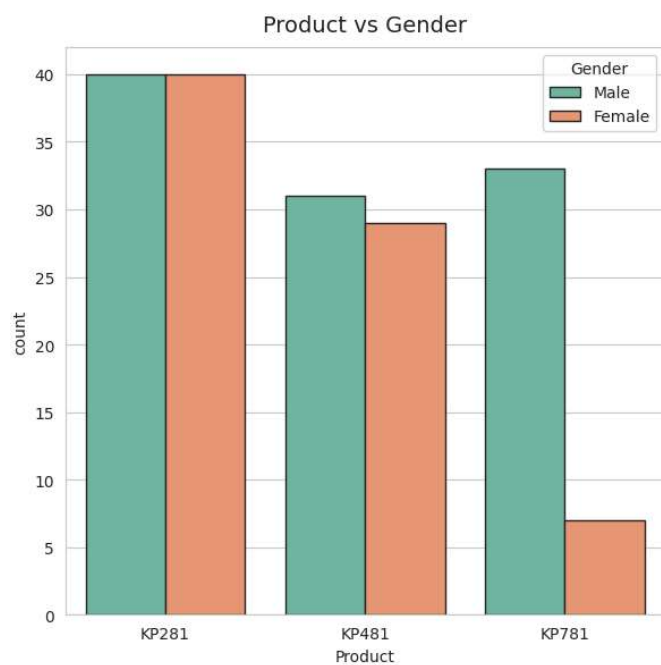
sns.countplot(data=df, x='Product', hue='Gender',
               edgecolor='0.15', ax=axs[0], palette="Set2")

sns.countplot(data=df, x='Product', hue='MaritalStatus',
               edgecolor='0.15', ax=axs[1], palette="husl")

axs[0].set_title('Product vs Gender', pad=10, fontsize=14)
axs[1].set_title('Product vs Marital Status', pad=10, fontsize=14)

axs[1].tick_params(axis='x', rotation=25)

plt.show()
```



## ✓ observation

Product vs Gender:

KP281 is equally popular among Male and Female.

KP481 is used by both genders almost equally, with males slightly higher.

KP781 is mostly chosen by Males, very few Females use it.

Product vs Marital Status:

Partnered customers dominate across all products.

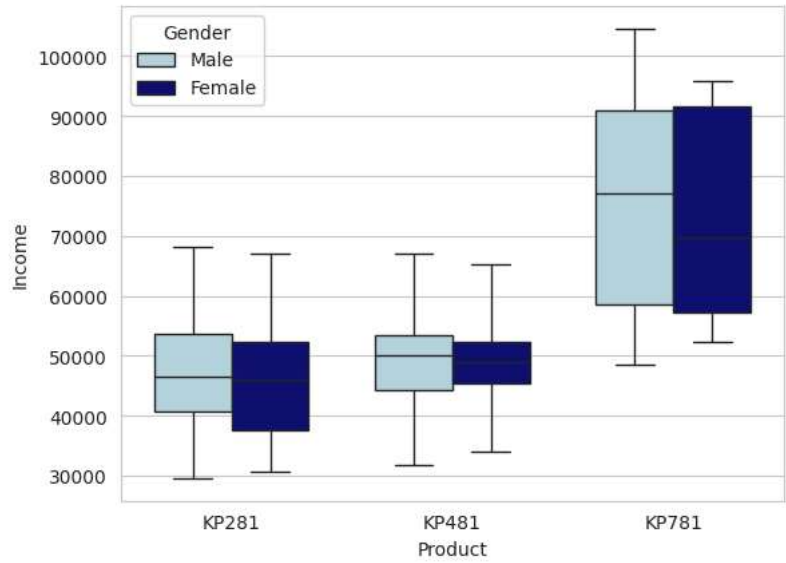
KP281 is the top choice for Partnered users, followed by KP481.

KP781 has the lowest adoption, especially among Singles.

👉 Overall: KP281 is the most popular product, balanced by gender, but more strongly favored by Partnered customers. KP781 is the least popular, especially among Females and Singles.

```
sns.boxplot(data=df, x='Product', y='Income',palette=["lightblue", "navy"] , hue='Gender', width=0.7, whis=5, notch=False,showcaps=True)
```

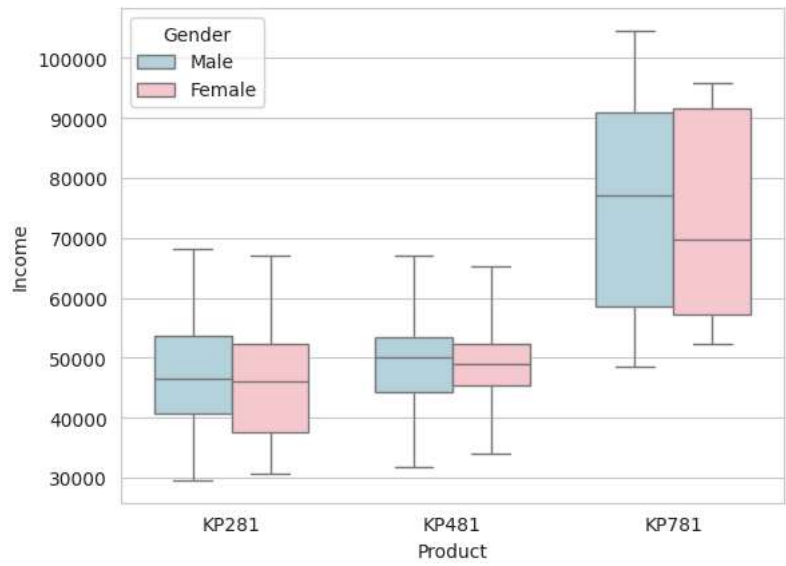
 <Axes: xlabel='Product', ylabel='Income'>




Start coding or [generate](#) with AI.

```
sns.boxplot(data=df, x='Product', y='Income',palette=["lightblue", "pink"] , hue='Gender', width=0.7, whis=5, notch=False,showcaps=True)
```

 <Axes: xlabel='Product', ylabel='Income'>




```
pd.crosstab(index=df.Product, columns=df.Gender, margins=True, normalize='index')
```



Gender	Female	Male
Product		
KP281	0.500000	0.500000
KP481	0.483333	0.516667
KP781	0.175000	0.825000
All	0.422222	0.577778

```
pd.crosstab(index=df.Product, columns=df.MaritalStatus, margins=True, normalize='index')
```



MaritalStatus	Partnered	Single
Product		
KP281	0.600000	0.400000