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

```
df = pd.read_csv("https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/125/original/aerofit_treadmill.csv?1639992749")
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

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

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

```
#Data Information
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 types of columns
df.dtypes
```

```
Product          object
Age               int64
Gender           object
Education         int64
MaritalStatus    object
Usage             int64
Fitness           int64
Income            int64
Miles             int64
dtype: object
```

```
df.describe()
```

```
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()/len(df)*100
```

```
Male      57.777778
Female    42.222222
Name: Gender, dtype: float64
```

```
df['Product'].value_counts()/len(df)*100
```

```
KP281    44.444444
KP481    33.333333
KP781    22.222222
Name: Product, dtype: float64
```

```
df['MaritalStatus'].value_counts()/len(df)*100
```

```
Partnered    59.444444
Single       40.555556
Name: MaritalStatus, dtype: float64
```

```
df.nunique()
```

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

```
#Unique product types
df['Product'].unique()
```

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

```
#Unique Gender type
df['Gender'].unique()
```

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

```
#Unique MaritalStatus type
df['MaritalStatus'].unique()
```

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

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

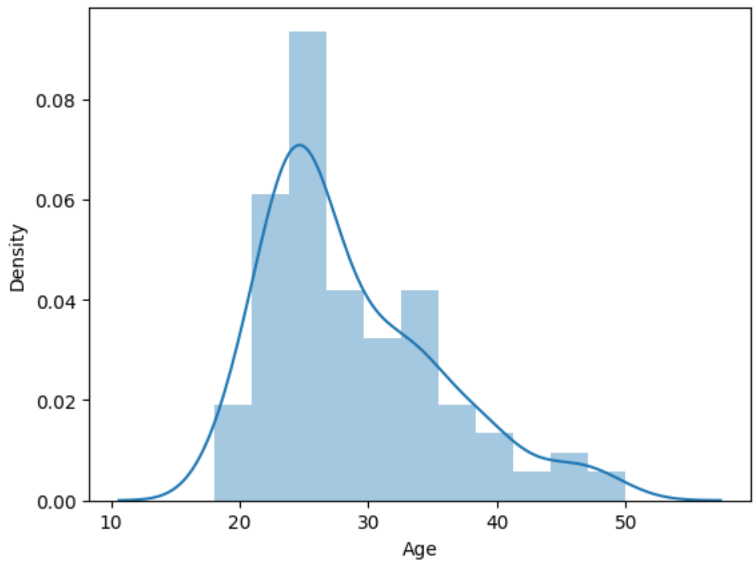
```
<ipython-input-18-0fafe04ea3f6>:1: UserWarning:

`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['Income'])
```

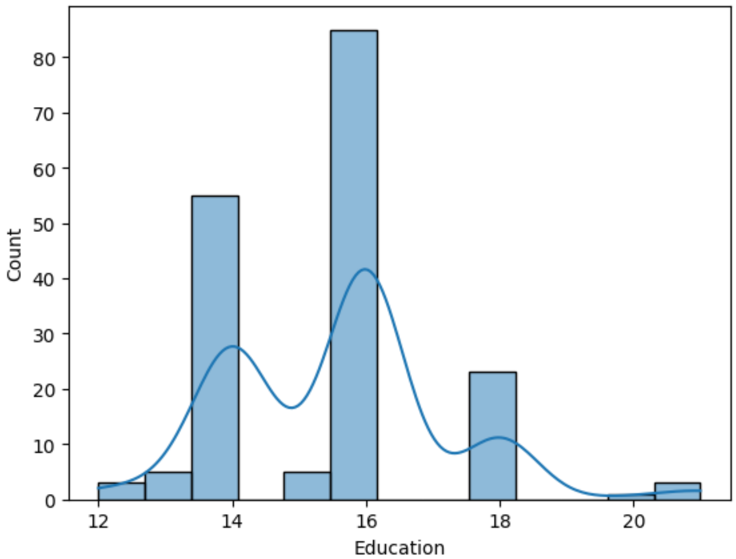
```
<ipython-input-19-332167f08bea>:1: UserWarning:

`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).
```

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

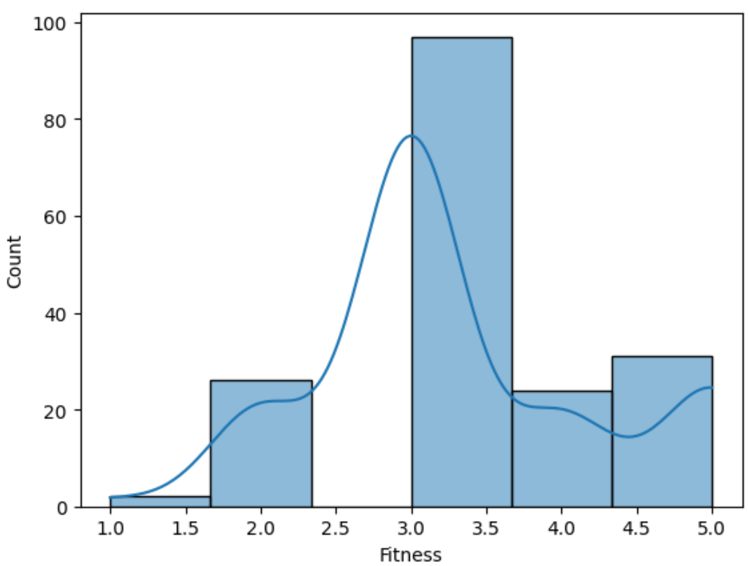
```
<Axes: xlabel='Education', ylabel='Count'>
```



Income

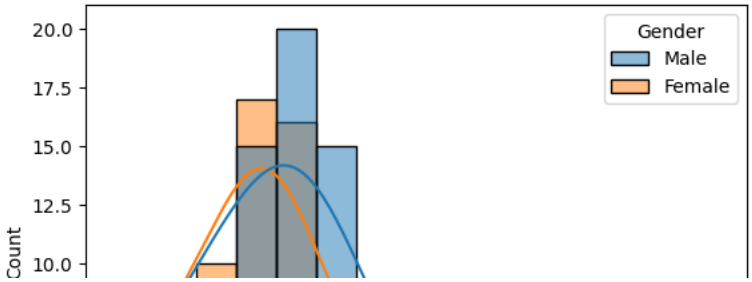
```
sns.histplot(df['Fitness'], kde=True,bins = 6)
```

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



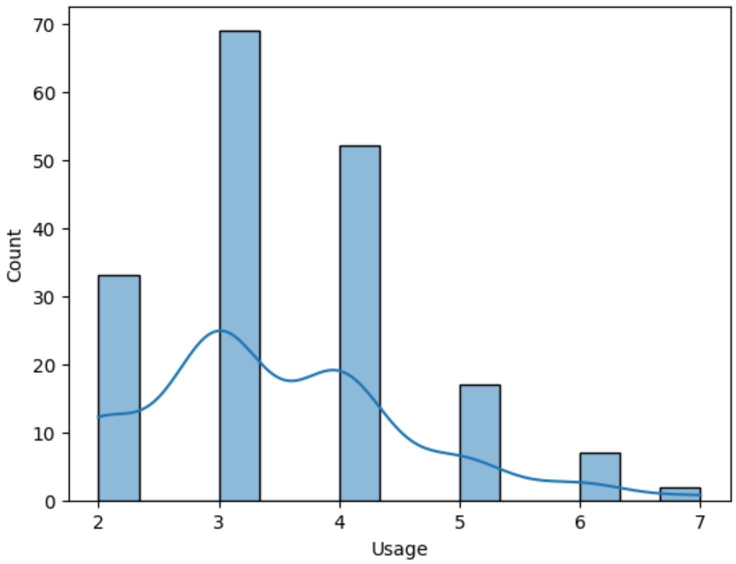

```
sns.histplot(x='Income',kde='True',data=df, hue='Gender')
```

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



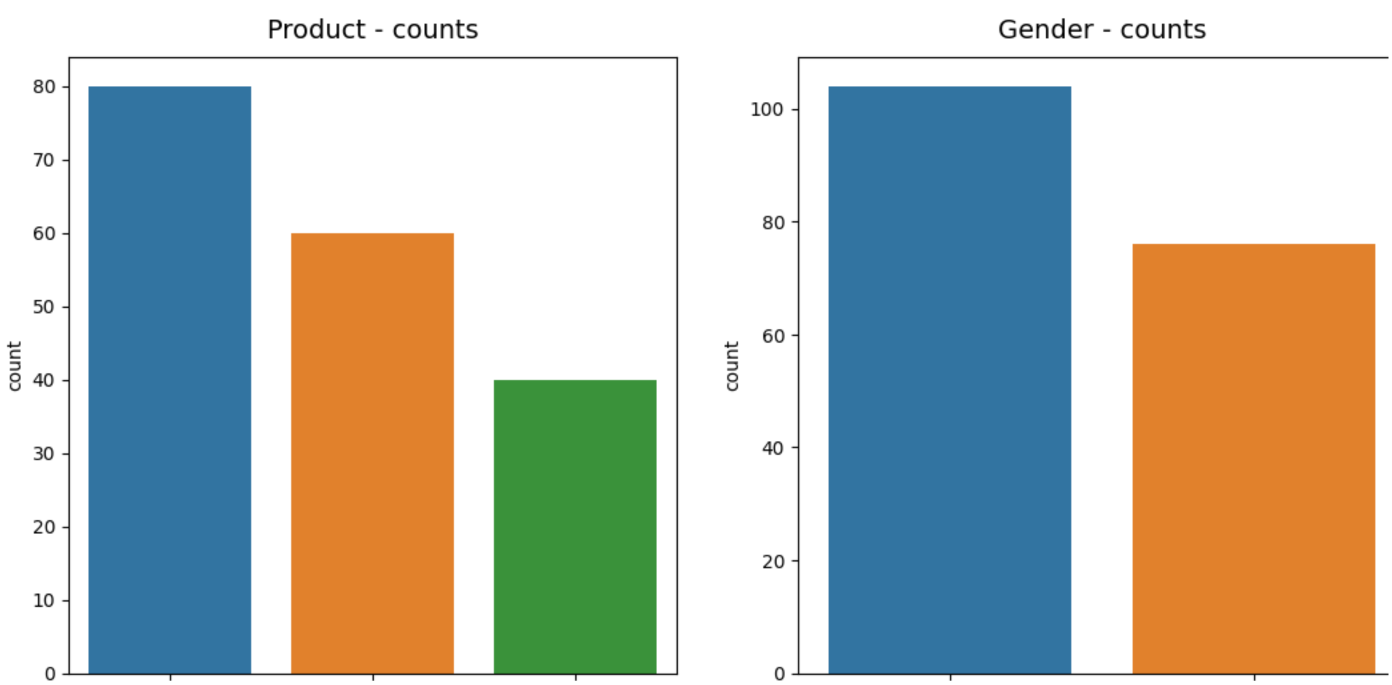

```
sns.histplot(df['Usage'],kde=True)
```

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




```
fig, axs = plt.subplots(nrows=1, ncols=3, figsize=(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])

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()
```

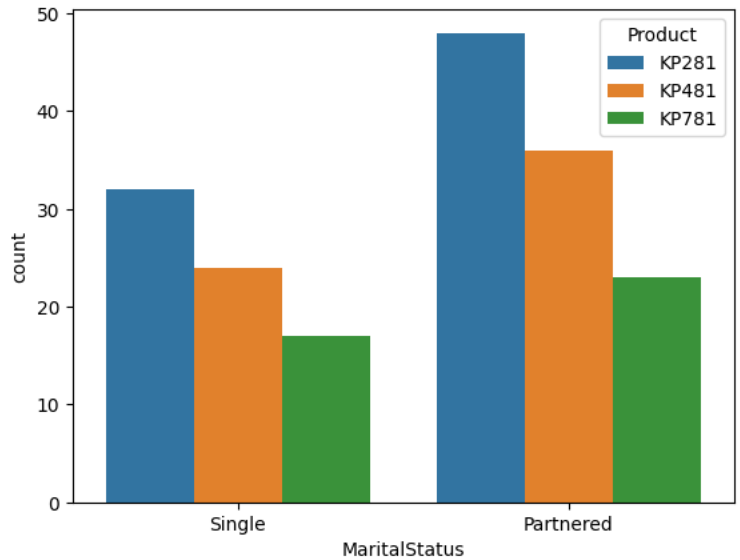


Observations:

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

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

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

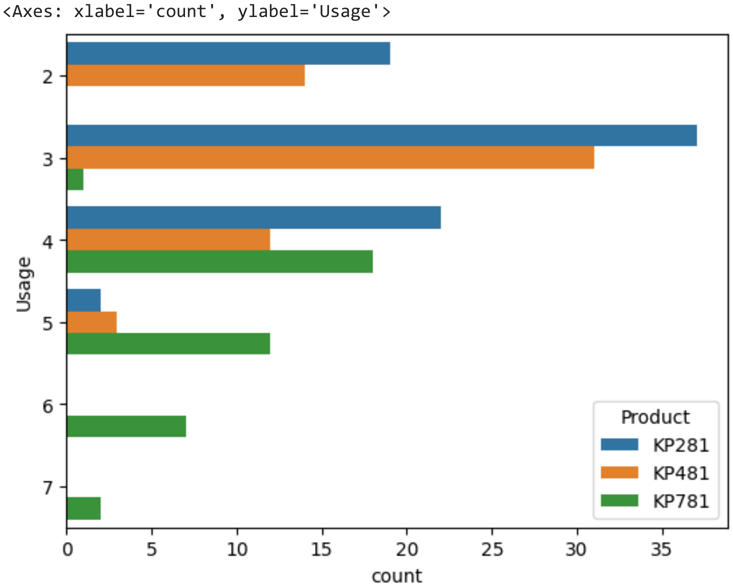


Observations:-

KP281 is the best product to pick both Partnered and Singles.

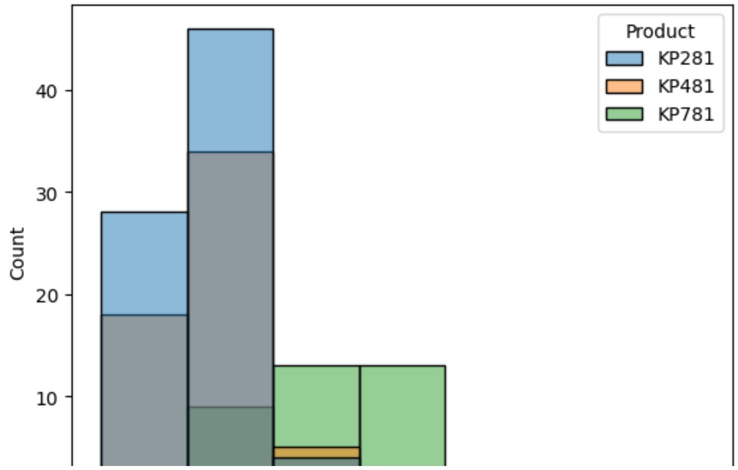
```
sns.countplot(data=df, y='Usage',hue='Product')
```

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



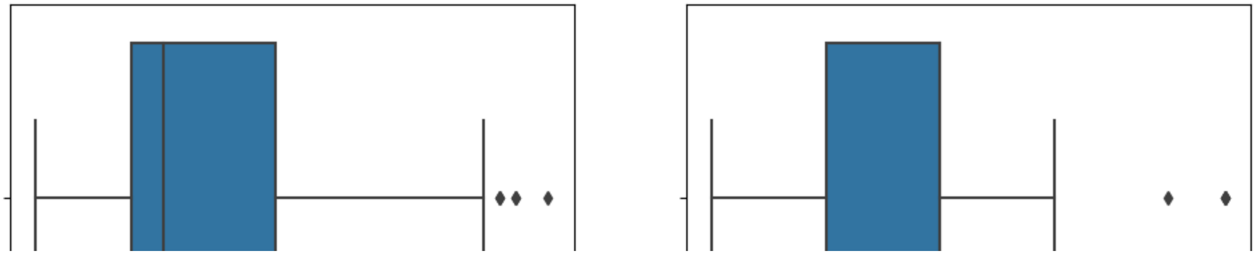

```
sns.histplot(data=df, x='Miles', hue='Product',bins= 7)
```

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




```
fig, axis = plt.subplots(nrows=3, ncols=2, figsize=(12, 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="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()
```



Observation:-

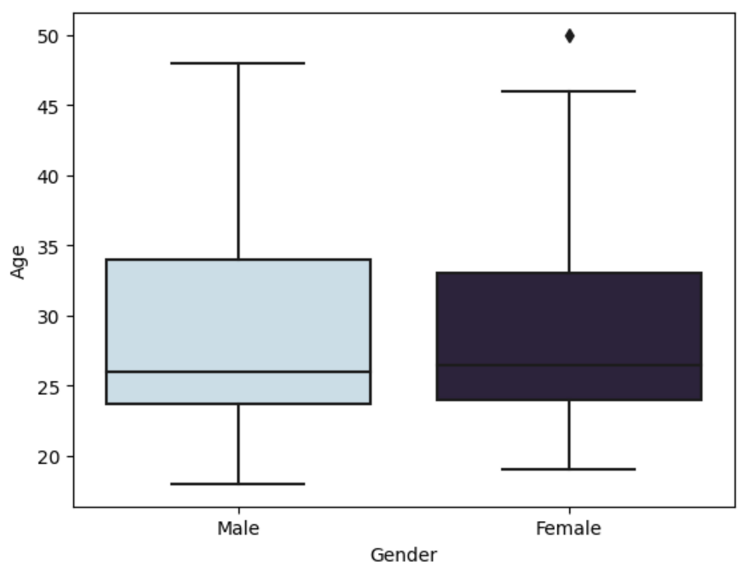
Even from the boxplots it is quite clear that:

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

Age Education

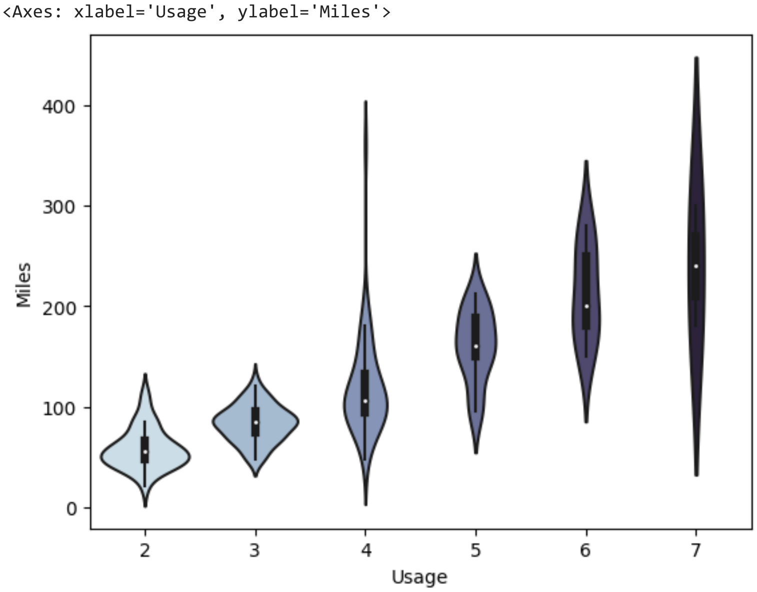
```
sns.boxplot(data=df, y = 'Age', x = 'Gender',palette='ch:s=.25,rot=-.25')
```

    <Axes: xlabel='Gender', ylabel='Age'>



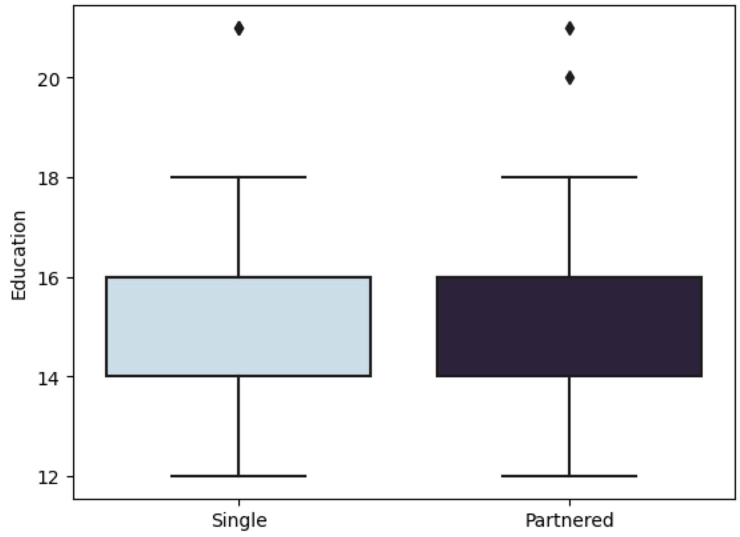
```
sns.violinplot(data=df, x='Usage', y = 'Miles',palette='ch:s=.25,rot=-.25')
```

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

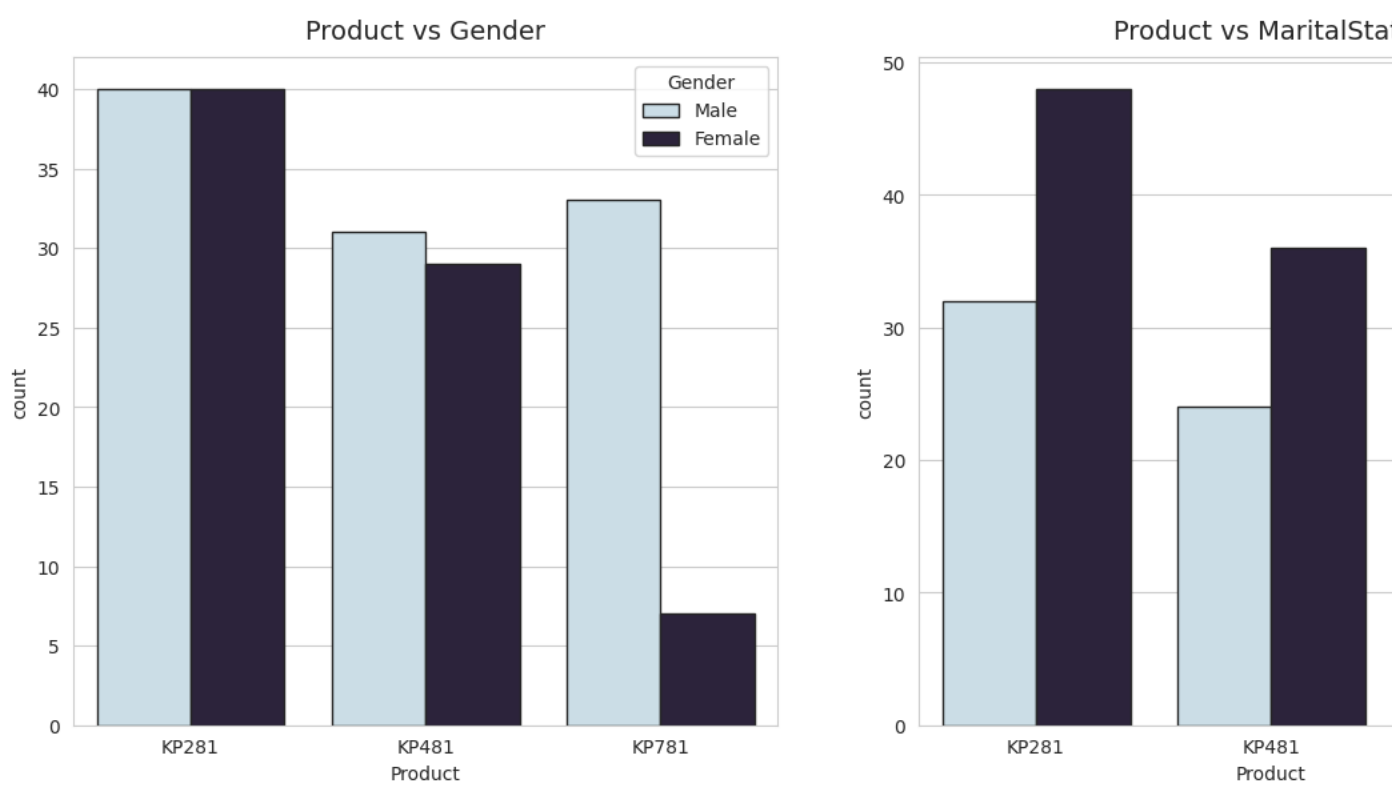



```
sns.boxplot(data=df, x='MaritalStatus', y='Education',palette='ch:s=.25,rot=-.25')
```

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




```
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", palette='ch:s=.25,rot=-.25', ax=axs[0])
sns.countplot(data=df, x='Product', hue='MaritalStatus', edgecolor="0.15", palette='ch:s=.25,rot=-.25', 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()
```



Observations:-

Product vs Gender:-

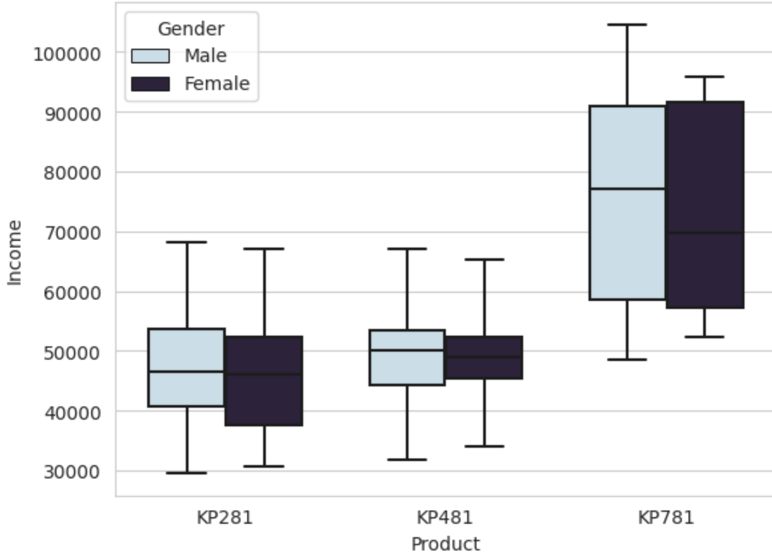
Equal number of males and females have purchased KP281 product and Almost same for the product KP481 Most of the Male customers have purchased the KP781 product.

Product vs MaritalStatus:-

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

```
sns.boxplot(data=df, x='Product', y='Income',palette='ch:s=.25,rot=-.25',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')
```

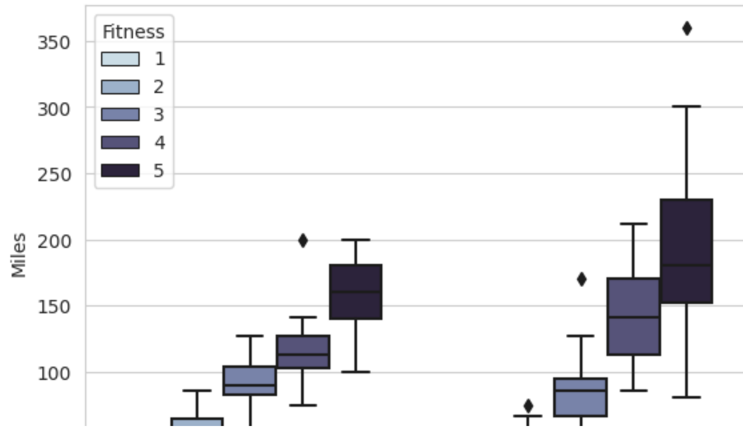
MaritalStatus	Partnered	Single
Product		
KP281	0.600000	0.400000
KP481	0.600000	0.400000
KP781	0.575000	0.425000
All	0.594444	0.405556

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

MaritalStatus	Partnered	Single
Product		
KP281	0.600000	0.400000
KP481	0.600000	0.400000
KP781	0.575000	0.425000
All	0.594444	0.405556

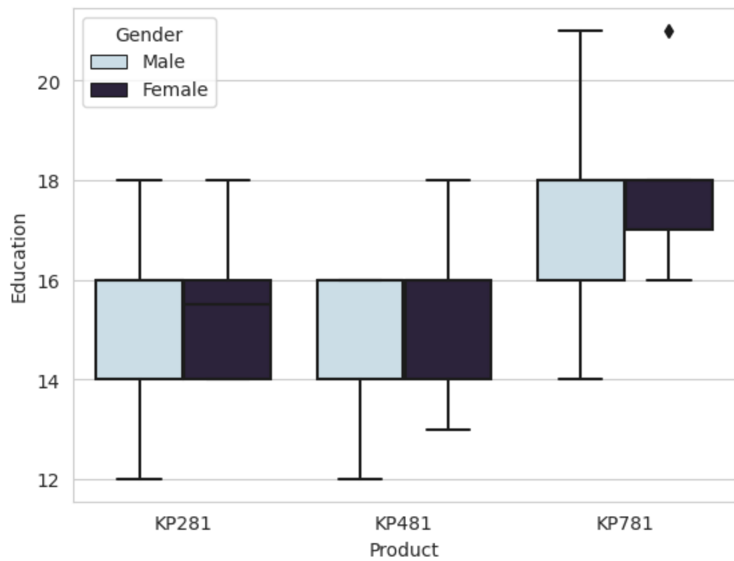
```
sns.boxplot(data=df, x='MaritalStatus', y='Miles',hue='Fitness',palette='ch:s=.25,rot=-.25')
```

```
<Axes: xlabel='MaritalStatus', ylabel='Miles'>
```

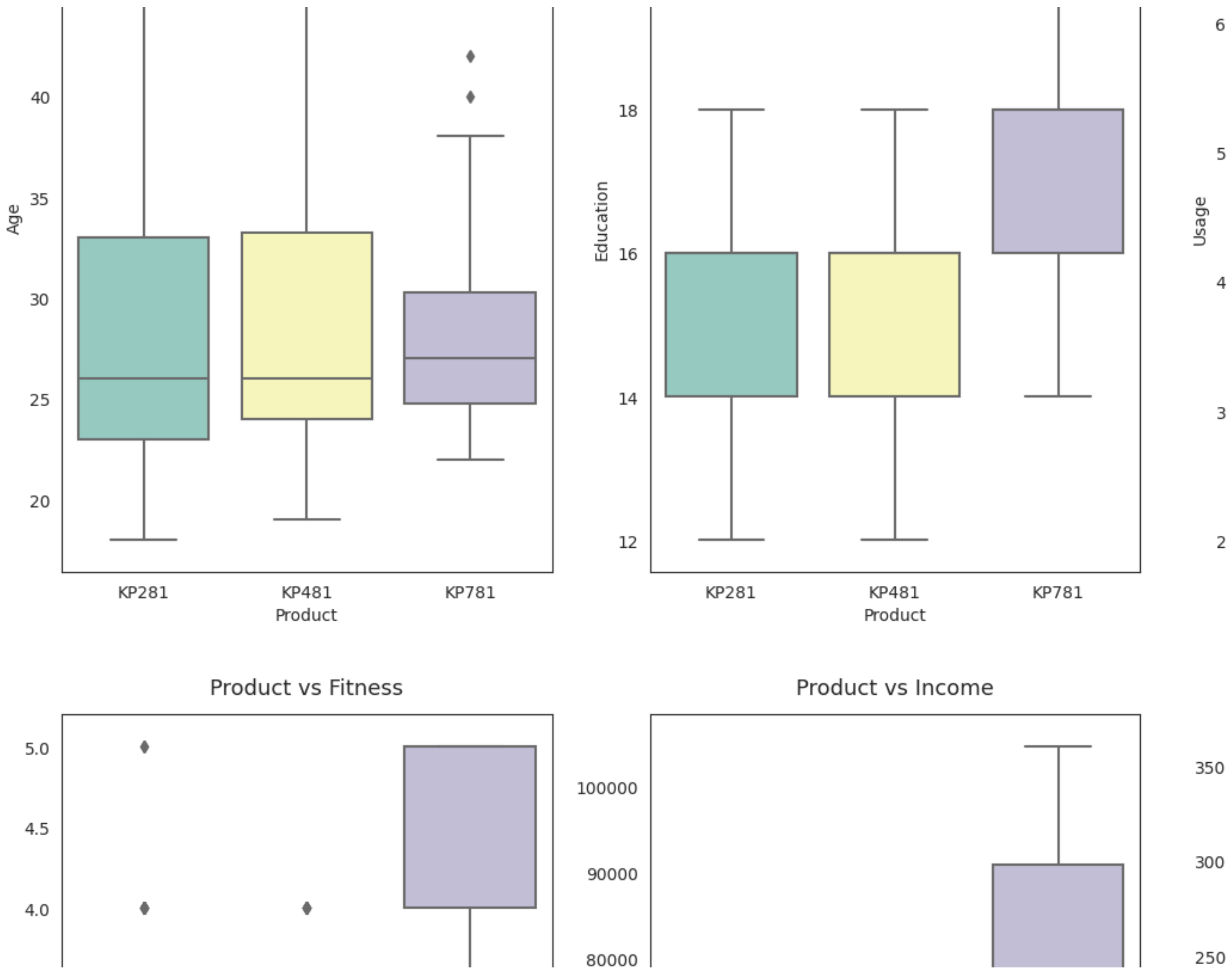



```
sns.boxplot(data=df, y='Education',x='Product',hue='Gender',palette='ch:s=.25,rot=-.25')
```

```
<Axes: xlabel='Product', ylabel='Education'>
```




```
attrs = ['Age', 'Education', 'Usage', 'Fitness', 'Income', 'Miles']
sns.set_style("white")
fig, axs = plt.subplots(nrows=2, ncols=3, figsize=(18, 12))
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=12, fontsize=13)
        count += 1
```



Observations:-

Product vs Age:-

```
Customers purchasing products KP281 & KP481 are having same Age median value.
Customers whose age lies between 25-30, are more likely to buy KP781 product
```

Product vs Education:-

```
Customers whose Education is greater than 16, have more chances to purchase the KP781 product.
While the customers with Education less than 16 have equal chances of purchasing KP281 or KP481.
```

Product vs Usage:-

```
Customers who are planning to use the treadmill greater than 4 times a week, are more likely to purchase the KP781 product.
While the other customers are likely to purchasing KP281 or KP481.
```

Product vs Fitness:-

```
The more the customer is fit (fitness >= 3), higher the chances of the customer to purchase the KP781 product.
```

Product vs Income:-

```
Higher the Income of the customer (Income >= 60000), higher the chances of the customer to purchase the KP781 product.
```

```
def p_prod_given_gender(gender, print_marginal=False):
    if gender != "Female" and gender != "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
```

```
def p_prod_given_mstatus(status, print_marginal=False):
    if status != "Single" and status != "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
```

Recommendations

1. KP281 is most frequent product. Hence Please make available in stock always.

2. Customer who is Partnered, is more likely to purchase the product. We need to do more survey why singles are not purchasing more and any discounts or compaine needed.
3. As per the data most of the customers are having 16 years of education and we need to enquiry why more than 16 education peoples are not purchasing.
4. P(KP781/Male): 0.32 > P(KP781/Female): 0.09. KP781 treadmill females are not purchasing more. We need to get the solution to sort out.