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

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

```
Downloading...
From: https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/125/original/aerofit_treadmill.csv
To: /content/aerofit_treadmill.csv
100% 7.28k/7.28k [00:00<00:00, 29.1MB/s]
```

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

```
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.keys()
```

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

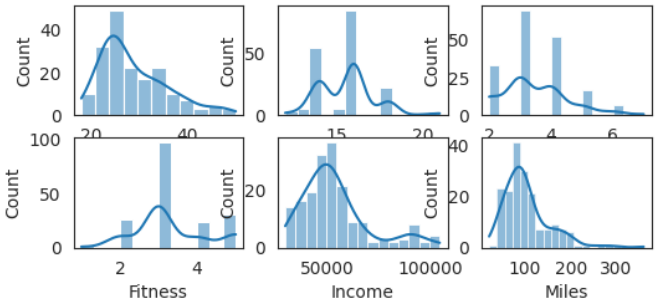
```
df.isnull().any()
```

```
Product          False
Age              False
Gender           False
Education        False
MaritalStatus    False
Usage            False
Fitness          False
Income           False
Miles            False
dtype: bool
```

There are no missing values in the data.

There are 3 unique products in the dataset.

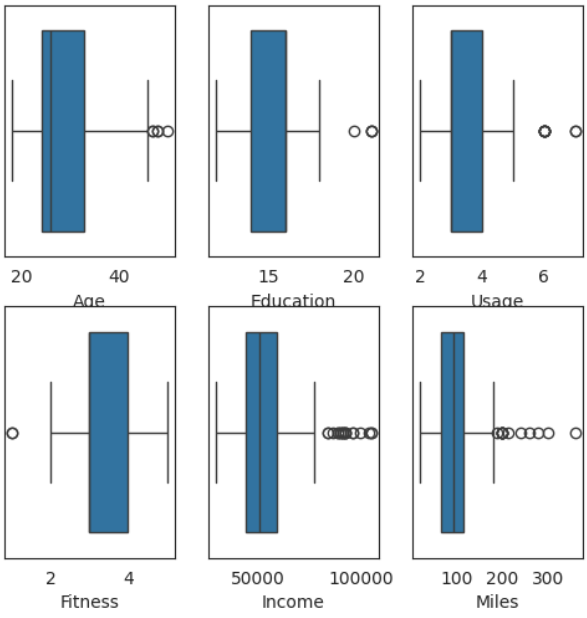
```
fig, axis = plt.subplots(nrows=2, ncols=3, figsize=(6, 5))
fig.subplots_adjust(top=0.5)
sns.histplot(data=df, x="Age", kde=True, ax=axis[0,0])
sns.histplot(data=df, x="Education", kde=True, ax=axis[0,1])
sns.histplot(data=df, x="Usage", kde=True, ax=axis[0,2])
sns.histplot(data=df, x="Fitness", kde=True, ax=axis[1,0])
sns.histplot(data=df, x="Income", kde=True, ax=axis[1,1])
sns.histplot(data=df, x="Miles", kde=True, ax=axis[1,2])
plt.show()
```



Above is the distribution of the data for the quanatative attributes:

Now lets check the outliers by using boxplots.

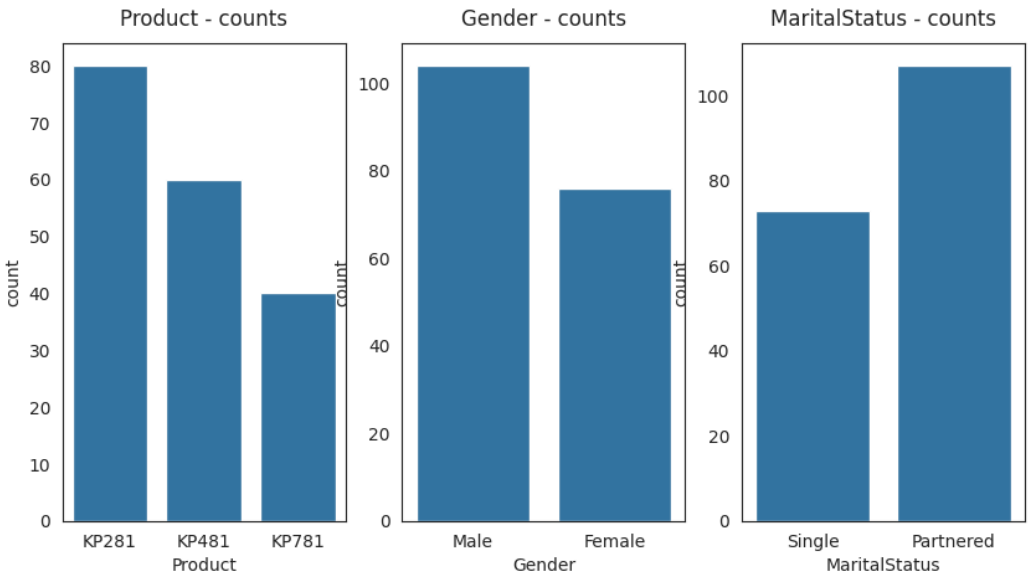
```
fig, axis = plt.subplots(nrows=2, ncols=3, figsize=(6, 5))
fig.subplots_adjust(top=1.0)
sns.boxplot(data=df, x="Age", orient='h', ax=axis[0,0])
sns.boxplot(data=df, x="Education", orient='h', ax=axis[0,1])
sns.boxplot(data=df, x="Usage", orient='h', ax=axis[0,2])
sns.boxplot(data=df, x="Fitness", orient='h', ax=axis[1,0])
sns.boxplot(data=df, x="Income", orient='h', ax=axis[1,1])
sns.boxplot(data=df, x="Miles", orient='h', ax=axis[1,2])
plt.show()
```



From above boxplots we can see that "Income" and "Miles" have more outliers than any other attributes.

Now, as we know Product, Gender and Marital Status are qulatative attributes. FOr that we need to see the distribbution of data using countplot.

```
fig, axs = plt.subplots(nrows=1, ncols=3, figsize=(10,5))
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=12)
axs[1].set_title("Gender - counts", pad=10, fontsize=12)
axs[2].set_title("MaritalStatus - counts", pad=10, fontsize=12)
plt.show()
```



From above obtained graphs we can say that quantity of product KP281 is high, there are more males than female in the data and there are more people with marital status as Partnered than single.

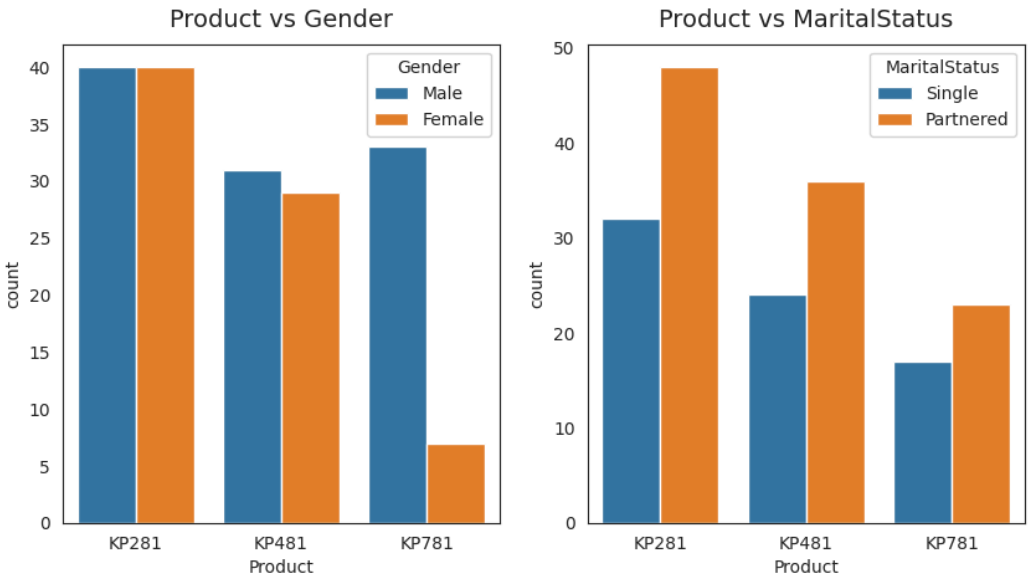
```
df1 = df[['Product', 'Gender', 'MaritalStatus']].melt() #melting down the data into small dataframe
df1.groupby(['variable', 'value'])[['value']].count() / len(df)
```

variable	value	value
Gender	Female	0.422222
	Male	0.577778
MaritalStatus	Partnered	0.594444
	Single	0.405556
Product	KP281	0.444444
	KP481	0.333333
	KP781	0.222222

- As we can see that 57.78% of the customers are Male.
- 59.44% of the customers are Partnered.
- 44.44% of the customers have purchased KP2821 product.
- 33.33% of the customers have purchased KP481 product.
- 22.22% of the customers have purchased KP781 product.

Now lets see if Gender and Marital Status has any affect on Product Purchase.

```
fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(10, 5.0))
sns.countplot(data=df, x='Product', hue='Gender', ax=axs[0])
sns.countplot(data=df, x='Product', hue='MaritalStatus', 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()
```

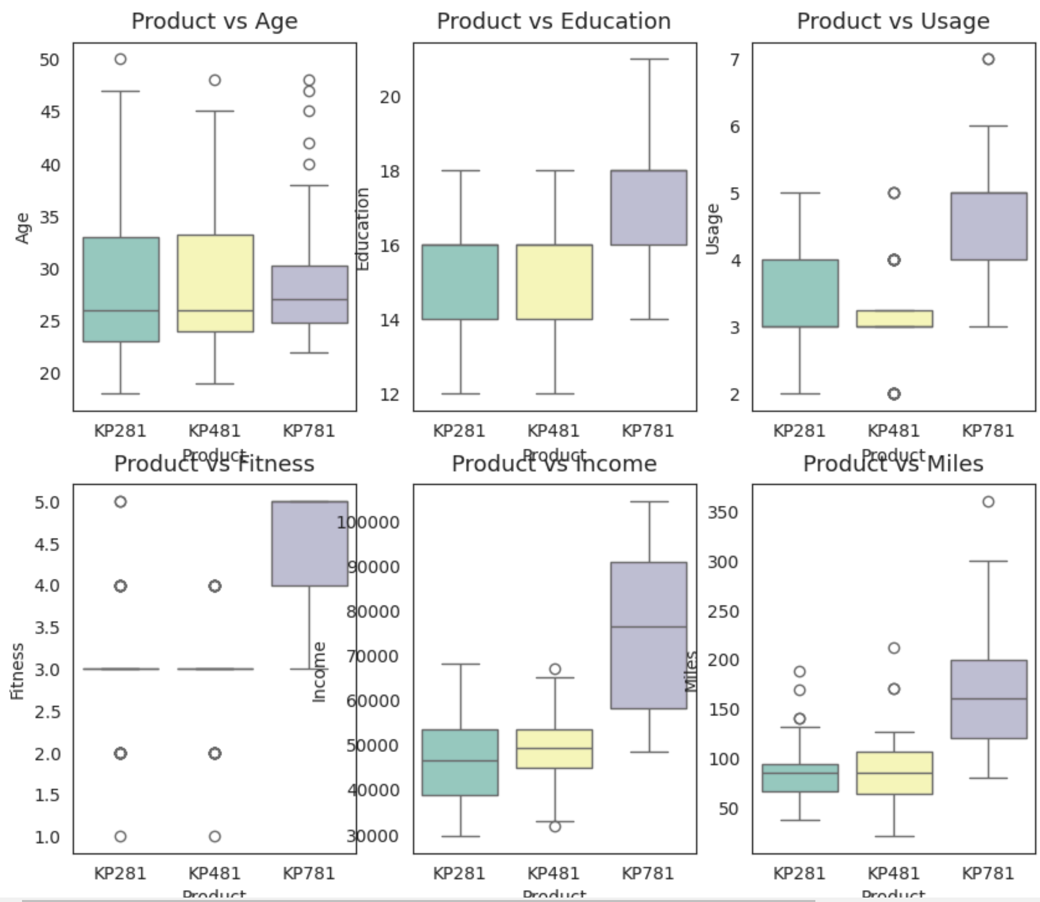


- Equal number of males and females have purchased KP281 product and Almost same for the product KP481
- Most of the Male customers have purchased the KP781 product.
- Customer who is Partnered, is more likely to purchase the product.

Lets see if quantative atributes have any affect on product

```
attrs = ['Age', 'Education', 'Usage', 'Fitness', 'Income', 'Miles']
sns.set_style("white")
fig, axs = plt.subplots(nrows=2, ncols=3, figsize=(10, 6))
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=8, fontsize=13)
    count += 1
```

```
<ipython-input-63-370316eb9267>:9: FutureWarning:

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

  sns.boxplot(data=df, x='Product', y=attrs[count], ax=axs[i,j], palette='Set3')
<ipython-input-63-370316eb9267>:9: FutureWarning:

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

  sns.boxplot(data=df, x='Product', y=attrs[count], ax=axs[i,j], palette='Set3')
<ipython-input-63-370316eb9267>:9: FutureWarning:

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

  sns.boxplot(data=df, x='Product', y=attrs[count], ax=axs[i,j], palette='Set3')
<ipython-input-63-370316eb9267>:9: FutureWarning:

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

  sns.boxplot(data=df, x='Product', y=attrs[count], ax=axs[i,j], palette='Set3')
```



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.

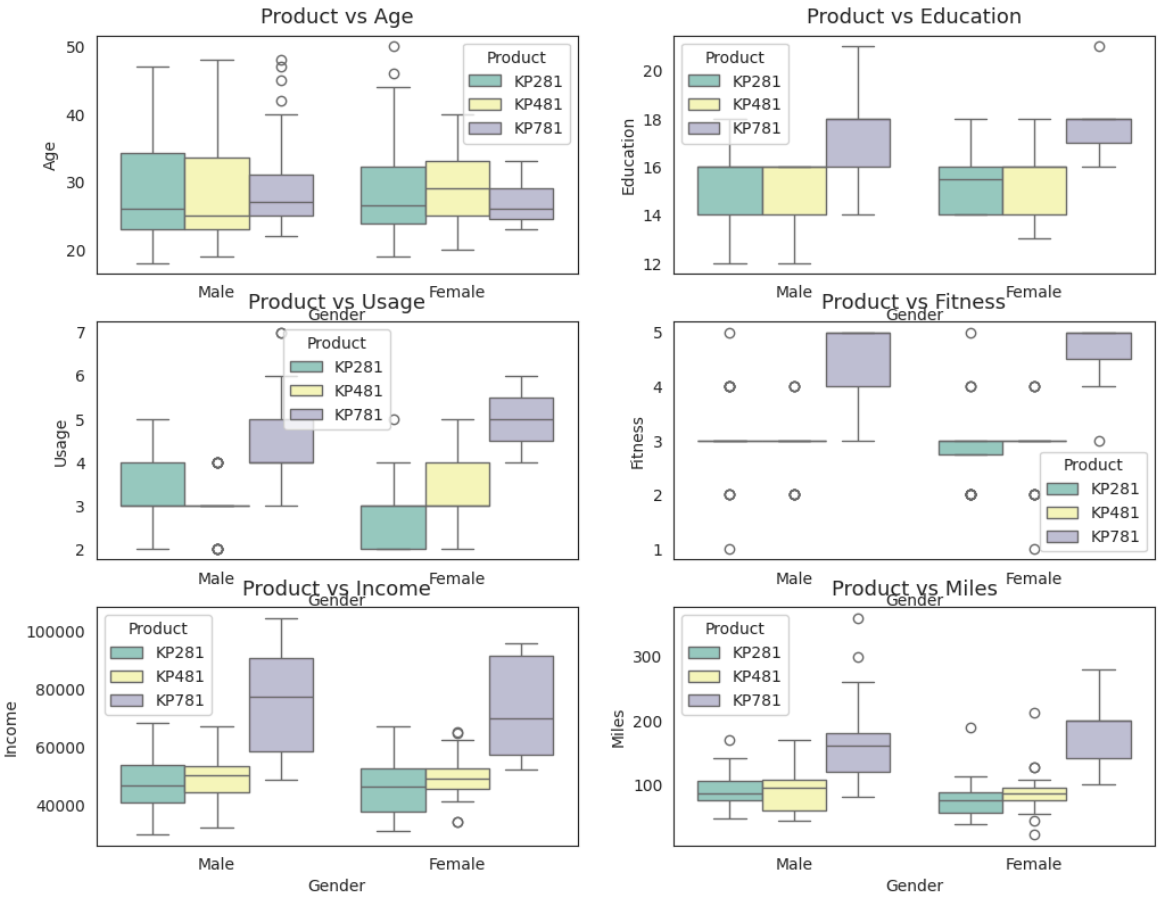
Product vs Income

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

Product vs Miles

- If the customer expects to walk/run greater than 120 Miles per week, it is more likely that the customer will buy KP781 product.

```
attrs = ['Age', 'Education', 'Usage', 'Fitness', 'Income', 'Miles']
sns.set_style("white")
fig, axs = plt.subplots(nrows=3, ncols=2, figsize=(12, 8))
fig.subplots_adjust(top=1)
count = 0
for i in range(3):
  for j in range(2):
    sns.boxplot(data=df, x='Gender', y=attrs[count], hue='Product', ax=axs[i,j], palette='Set3')
    axs[i,j].set_title(f"Product vs {attrs[count]}", pad=8, fontsize=13)
    count += 1
```



- Females planning to use treadmill 3-4 times a week, are more likely to buy KP481 product

Now Lets Calculate Marginal & Conditional Probabilities:

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

```
KP281    0.444444
KP481    0.333333
KP781    0.222222
Name: Product, dtype: float64
```

```
def p_prod_given_gender(gender, print_marginal=False):
  if gender is not "Female" and gender is not "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

<>:2: SyntaxWarning: "is not" with a literal. Did you mean "!="?
```