AEROFIT CASE STUDY

About the Company:

Aerofit is a company that manufactures and sells fitness equipment. The company has three products: KP281, KP481, and KP781. The company has collected data on customers who have purchased their products. The data includes the following columns:

- Product: The product that the customer purchased (KP281, KP781, KP481)
- Age : This shows the age of the customer who buys the product.
- Gender : This shows the gender of the customer who buys the product.
- Education : This shows the education level of the customer who buys the product.
- MaritalStatus : This shows the marital status of the customer who buys the product.
- Usage : This shows the number of times the customer uses the product in a week.
- Fitness : This shows the fitness level of the customer who buys the product.
- Income : This shows the income of the customer who buys the product.
- Miles : This shows the number of miles the customer expects to run.

BASIC INFORMATION

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

pd.read_csv("fitness.csv")
```

```
    Product  Age  Gender  Education MaritalStatus  Usage  Fitness
Income  \
0     KP281   18    Male         14        Single      3        4
29562
1     KP281   19    Male         15        Single      2        3
31836
2     KP281   19  Female         14     Partnered      4        3
30699
3     KP281   19    Male         12        Single      3        3
32973
4     KP281   20    Male         13     Partnered      4        2
35247
..      ...  ...     ...        ...           ...    ...      ...
...
175   KP781   40    Male         21        Single      6        5
83416
176   KP781   42    Male         18        Single      5        4
```

```
89641
177   KP781   45    Male          16        Single      5         5
90886
178   KP781   47    Male          18     Partnered      4         5
104581
179   KP781   48    Male          18     Partnered      4         5
95508

     Miles
0      112
1       75
2       66
3       85
4       47
..     ...
175    200
176    200
177    160
178    120
179    180

[180 rows x 9 columns]
```

```
df = pd.read_csv("fitness.csv")
df
```

```
    Product  Age  Gender  Education MaritalStatus  Usage  Fitness
Income  \
0     KP281   18    Male         14        Single      3         4
29562
1     KP281   19    Male         15        Single      2         3
31836
2     KP281   19  Female         14     Partnered      4         3
30699
3     KP281   19    Male         12        Single      3         3
32973
4     KP281   20    Male         13     Partnered      4         2
35247
..      ...  ...     ...        ...           ...    ...       ...
...
175   KP781   40    Male         21        Single      6         5
83416
176   KP781   42    Male         18        Single      5         4
89641
177   KP781   45    Male         16        Single      5         5
90886
178   KP781   47    Male         18     Partnered      4         5
104581
179   KP781   48    Male         18     Partnered      4         5
95508
```

```
     Miles
0      112
1       75
2       66
3       85
4       47
..     ...
175    200
176    200
177    160
178    120
179    180

[180 rows x 9 columns]
```

Shape of the Data(No of Rows and Columns)

```
np.shape(df)
```

```
(180, 9)
```

Description of the Data(mean, median, mode, standard deviation, min, max, etc of the given column)

```
df.describe()
```

```
              Age   Education       Usage     Fitness
Income  \
count  180.000000  180.000000  180.000000  180.000000     180.000000

mean    28.788889   15.572222    3.455556    3.311111   53719.577778

std      6.943498    1.617055    1.084797    0.958869   16506.684226

min     18.000000   12.000000    2.000000    1.000000   29562.000000

25%     24.000000   14.000000    3.000000    3.000000   44058.750000

50%     26.000000   16.000000    3.000000    3.000000   50596.500000

75%     33.000000   16.000000    4.000000    4.000000   58668.000000

max     50.000000   21.000000    7.000000    5.000000  104581.000000


            Miles
count  180.000000
mean   103.194444
```

```
std       51.863605
min       21.000000
25%       66.000000
50%       94.000000
75%      114.750000
max      360.000000
```

Information about the Data(The data types of the columns, the number of non-null values, the memory usage, etc.)

```
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
```

Changing the data type of the categorical columns to category

```
categorical_columns = ['Product', 'Gender', 'MaritalStatus', 
'Fitness']

for column in categorical_columns:
    df[column] = df[column].astype('category')

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    category
 1   Age            180 non-null    int64   
 2   Gender         180 non-null    category
```

```
 3   Education      180 non-null    int64
 4   MaritalStatus  180 non-null    category
 5   Usage          180 non-null    int64
 6   Fitness        180 non-null    category
 7   Income         180 non-null    int64
 8   Miles          180 non-null    int64
dtypes: category(4), int64(5)
memory usage: 8.4 KB
```

Things Done:

- The data was loaded
- The shape of the data was checked
- The data was described using the describe function
- The data types were checked. The info function was used to check the data types.
- The data types were changed to category for the categorical columns. Categorical Columns are those columns that describe characteristics or qualities of data points rather than numerical values.

Observations

- There are 180 rows and 9 columns
- There are no missing values
- The data types are correct
- The data is clean
- The data is ready for analysis

Questions:

- Why we changed the data type to category? Answer: Because the data is categorical and not numerical, which hence makes the code, efficient, faster and more readable.

Non Graphical Analysis

```
for col in df.columns:
    print("------------",col,"------------")
    print(df[col].value_counts())
    print()
```

```
------------ Product ------------
Product
KP281    80
KP481    60
KP781    40
Name: count, dtype: int64

------------ Age ------------
```

```
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

------------ Gender ------------
Gender
Male      104
Female     76
Name: count, dtype: int64

------------ Education ------------
Education
16    85
14    55
18    23
15     5
13     5
12     3
```

```
21     3
20     1
Name: count, dtype: int64

------------ MaritalStatus ------------
MaritalStatus
Partnered    107
Single        73
Name: count, dtype: int64

------------ Usage ------------
Usage
3    69
4    52
2    33
5    17
6     7
7     2
Name: count, dtype: int64

------------ Fitness ------------
Fitness
3    97
5    31
2    26
4    24
1     2
Name: count, dtype: int64

------------ Income ------------
Income
45480    14
52302     9
46617     8
54576     8
53439     8
         ..
65220     1
55713     1
68220     1
30699     1
95508     1
Name: count, Length: 62, dtype: int64

------------ Miles ------------
Miles
85     27
95     12
66     10
75     10
```

```
47      9
106     9
94      8
113     8
53      7
100     7
180     6
200     6
56      6
64      6
127     5
160     5
42      4
150     4
38      3
74      3
170     3
120     3
103     3
132     2
141     2
280     1
260     1
300     1
240     1
112     1
212     1
80      1
140     1
21      1
169     1
188     1
360     1
Name: count, dtype: int64
```

```
for col in categorical_columns:
    print("------------",col,"------------")
    print(df.groupby(col).size())
    print()
```

```
------------ Product ------------
Product
KP281    80
KP481    60
KP781    40
dtype: int64

------------ Gender ------------
Gender
```

```
Female     76
Male      104
dtype: int64

------------ MaritalStatus ------------
MaritalStatus
Partnered    107
Single        73
dtype: int64


C:\Users\HEMKESH\AppData\Local\Temp\ipykernel_21040\2634789354.py:3: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future default and silence this warning.
  print(df.groupby(col).size())
C:\Users\HEMKESH\AppData\Local\Temp\ipykernel_21040\2634789354.py:3: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future default and silence this warning.
  print(df.groupby(col).size())
C:\Users\HEMKESH\AppData\Local\Temp\ipykernel_21040\2634789354.py:3: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future default and silence this warning.
  print(df.groupby(col).size())
```

Things Done:

- The value counts of each column was checked. Looped on each column and counted the number of times each value appeared in the column.
- The value counts of each categorical column was checked.
- The value counts of each categorical column was checked using the groupby function.

Observations

```
## Product
    - There are 3 products
    - They are different types of treadmills.
    - The product with the most sales is *KP281*.
    - The product with the least sales is *KP781*.
    - The product with the second most sales is *KP481*.
## Gender
    - There is an higher number of the male purchasers.
    - Male purchasers are 1.5 times more than the female purchasers.
```

```
## MaritalStatus
    - There are more married purchasers than single purchasers.
    - Married purchasers are also 1.5 times more than the single 
purchasers.

continuous_columns =["Age", "Education", "Usage", "Fitness", "Income", 
"Miles"]
df.describe()
```

```
              Age   Education       Usage     Fitness  
Income  \
count  180.000000  180.000000  180.000000  180.000000     180.000000   

mean    28.788889   15.572222    3.455556    3.311111   53719.577778   

std      6.943498    1.617055    1.084797    0.958869   16506.684226   

min     18.000000   12.000000    2.000000    1.000000   29562.000000   

25%     24.000000   14.000000    3.000000    3.000000   44058.750000   

50%     26.000000   16.000000    3.000000    3.000000   50596.500000   

75%     33.000000   16.000000    4.000000    4.000000   58668.000000   

max     50.000000   21.000000    7.000000    5.000000  104581.000000   


            Miles  
count  180.000000  
mean   103.194444  
std     51.863605  
min     21.000000  
25%     66.000000  
50%     94.000000  
75%    114.750000  
max    360.000000  
```

Marginal Probability

```
pd.crosstab(index = [df["Product"],df['Gender']], columns = 
[df["Age"]], margins = True)
```

```
Age             18  19  20  21  22  23  24  25  26  27  ...  41  42  
43  44  \
Product Gender                                          ...           

KP281   Female   0   1   1   2   3   3   3   4   3   2  ...   0   0   
0   1   
        Male     1   2   1   2   1   5   2   3   4   1  ...   1   0   
1   0   
```

```
KP481   Female   0   0   1   1   0   3   2   5   2   0  ...   0   0
0   0
        Male     0   1   2   2   0   4   1   6   1   1  ...   0   0
0   0
KP781   Female   0   0   0   0   0   1   1   1   1   0  ...   0   0
0   0
        Male     0   0   0   0   3   2   3   6   1   3  ...   0   1
0   0
All              1   4   5   7   7  18  12  25  12   7  ...   1   1
1   1

Age             45  46  47  48  50  All
Product Gender
KP281   Female   0   1   0   0   1   40
        Male     0   0   1   0   0   40
KP481   Female   0   0   0   0   0   29
        Male     1   0   0   1   0   31
KP781   Female   0   0   0   0   0    7
        Male     1   0   1   1   0   33
All              2   1   2   2   1  180

[7 rows x 33 columns]
```

```
pd.crosstab(df["Product"],df['Gender'])
```

```
Gender   Female  Male
Product
KP281        40    40
KP481        29    31
KP781         7    33
```

Observations:

- The marginal probability of a customer buying the product KP281 is <u>0.5</u>.
- The marginal probability of a customer buying the product KP481 is <u>0.3</u>.
- The marginal probability of a customer buying the product KP781 is <u>0.2</u>.
- *Product and Gender*: The marginal probability table between Product and Gender shows the count of each gender for each product. This can help us understand the gender distribution for each product. For example, we can observe how many males and females bought each product.
- *Product and Age*: The marginal probability table between Product and Age shows the count of each age group for each product. This can help us understand the age distribution for each product. For example, we can observe how many customers of each age group bought each product.
- *Product and Marital Status*: The marginal probability table between Product and Marital Status shows the count of each marital status for each product. This can help us understand the marital status distribution for each product. For example, we can observe how many married and single customers bought each product.

Conditional Probability

CASE 1

- Event A: Selling A product with id KP281
- Event B: Selling a product to male customer
- P(B|A) = P(A and B)/P(A)
- P(A and B) = Number of events where A and B occur/Total number of events
- P(A) = Number of events where A occurs/Total number of events
- P(B|A) = Number of events where A and B occur/Number of events where A occurs

```
event_A_Condition = df['Product'] == 'KP281'
event_B_Condition = df['Gender'] == 'Male'

number_of_events_where_A_and_B = df[event_A_Condition &
event_B_Condition].shape[0]
number_of_events_where_A_occurs = df[event_A_Condition].shape[0]

probab_B_Given_A = int(number_of_events_where_A_and_B)/
int(number_of_events_where_A_occurs)
probab_B_Given_A

0.5
```

CASE 2

- Event A: Selling a product to female customer
- Event B: Selling a product to a customer of age greater than 23.
- P(B|A) = P(A and B)/P(A)
- P(A and B) = Number of events where A and B occur/Total number of events
- P(A) = Number of events where A occurs/Total number of events
- P(B|A) = Number of events where A and B occur/Number of events where A occurs

```
event_A_Condition = df["Gender"] == 'Female'
event_B_Condition = df["Age"] > 23

number_of_events_where_A_and_B = df[event_A_Condition &
event_B_Condition].shape[0]
number_of_events_where_A_occurs = df[event_A_Condition].shape[0]

probab_B_Given_A = int(number_of_events_where_A_and_B)/
int(number_of_events_where_A_occurs)
probab_B_Given_A

0.7894736842105263
```

CASE 3

- Event A: Selling A product with id KP781
- Event B: Selling a product to a married customer
- P(B|A) = P(A and B)/P(A)

- P(A and B) = Number of events where A and B occur/Total number of events
- P(A) = Number of events where A occurs/Total number of events
- P(B|A) = Number of events where A and B occur/Number of events where A occurs

```
event_A_Condition = df['Product'] == 'KP781'
event_B_Condition = df['MaritalStatus'] == 'Married'
number_of_events_where_A_and_B = df[event_A_Condition &
event_B_Condition].shape[0]
number_of_events_where_A_occurs = df[event_A_Condition].shape[0]
probab_B_Given_A = int(number_of_events_where_A_and_B)/
int(number_of_events_where_A_occurs)
probab_B_Given_A

0.0
```

CASE 4

- Event A: Selling a product to a customer with a fitness level equal to 3
- Event B: Selling a product to a female customer
- P(B|A) = P(A and B)/P(A)
- P(A and B) = Number of events where A and B occur/Total number of events
- P(A) = Number of events where A occurs/Total number of events
- P(B|A) = Number of events where A and B occur/Number of events where A occurs

```
event_A_Condition = df['Fitness'] == 3
event_B_Condition = df['Gender'] == 'Female'
number_of_events_where_A_and_B = df[event_A_Condition &
event_B_Condition].shape[0]
number_of_events_where_A_occurs = df[event_A_Condition].shape[0]
probab_B_Given_A = int(number_of_events_where_A_and_B)/
int(number_of_events_where_A_occurs)
probab_B_Given_A

0.4639175257731959
```

CASE 5

- Event A: Selling a product to a customer who uses the product more than 3 times a week
- Event B: Selling a product to a customer who runs more than 100 miles
- P(B|A) = P(A and B)/P(A)
- P(A and B) = Number of events where A and B occur/Total number of events
- P(A) = Number of events where A occurs/Total number of events
- P(B|A) = Number of events where A and B occur/Number of events where A occurs

```
event_A_Condition = df['Usage'] > 3
event_B_Condition = df['Miles'] > 100
number_of_events_where_A_and_B = df[event_A_Condition &
event_B_Condition].shape[0]
number_of_events_where_A_occurs = df[event_A_Condition].shape[0]
probab_B_Given_A = int(number_of_events_where_A_and_B)/
```

```
int(number_of_events_where_A_occurs)
probab_B_Given_A
```

```
0.6666666666666666
```

CASE 6

- Event A: Selling A product with id KP281
- Event B: Selling a product to a customer with an income above the median income
- P(B|A) = P(A and B)/P(A)
- P(A and B) = Number of events where A and B occur/Total number of events
- P(A) = Number of events where A occurs/Total number of events
- P(B|A) = Number of events where A and B occur/Number of events where A occurs

```
event_A_Condition = df['Product'] == 'KP281'
event_B_Condition = df['Income'] > df['Income'].median()
number_of_events_where_A_and_B = df[event_A_Condition &
event_B_Condition].shape[0]
number_of_events_where_A_occurs = df[event_A_Condition].shape[0]
probab_B_Given_A = int(number_of_events_where_A_and_B)/
int(number_of_events_where_A_occurs)
probab_B_Given_A
```

```
0.375
```

CASE 7

- Event A: Selling a product to a customer with an education level above the median education level
- Event B: Selling a product to a single customer
- P(B|A) = P(A and B)/P(A)
- P(A and B) = Number of events where A and B occur/Total number of events
- P(A) = Number of events where A occurs/Total number of events
- P(B|A) = Number of events where A and B occur/Number of events where A occurs

```
event_A_Condition = df['Education'] > df['Education'].median()
event_B_Condition = df['MaritalStatus'] == 'Single'
number_of_events_where_A_and_B = df[event_A_Condition &
event_B_Condition].shape[0]
number_of_events_where_A_occurs = df[event_A_Condition].shape[0]
probab_B_Given_A = int(number_of_events_where_A_and_B)/
int(number_of_events_where_A_occurs)
probab_B_Given_A
```

```
0.4074074074074074
```

Observations:

- If someone buys the most selling product (KP281), there is a 50% chance that a male would be buying that product.

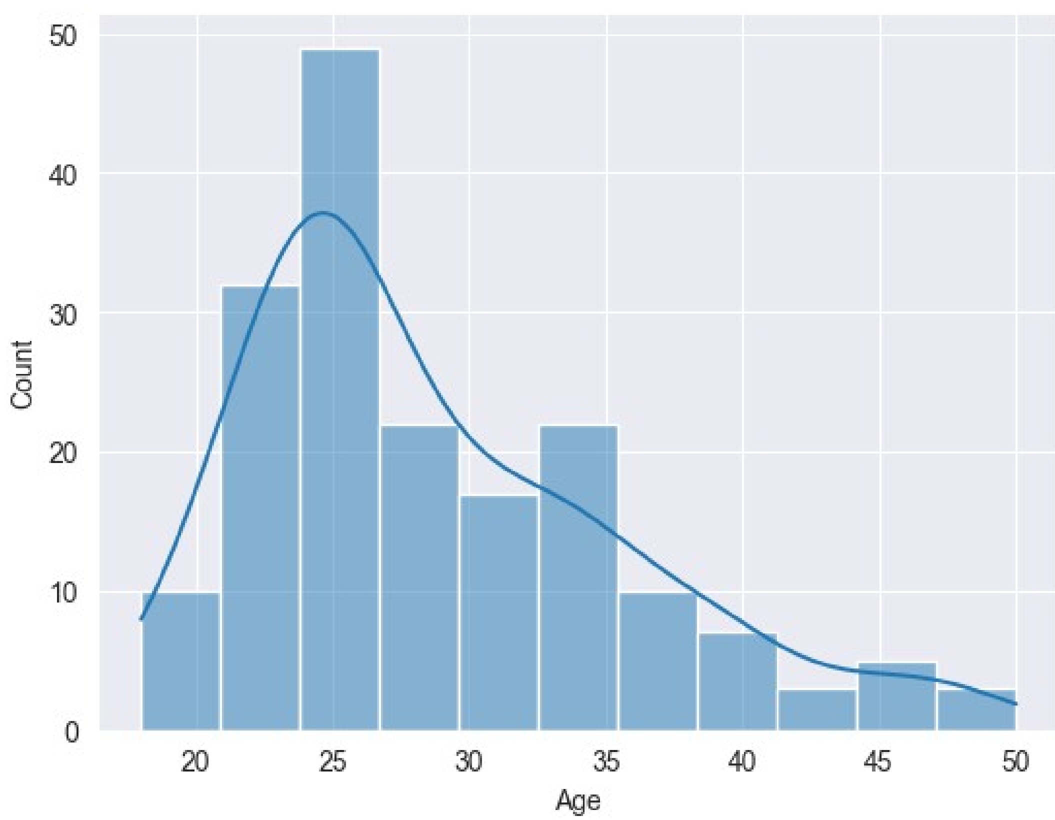
- If a female buys a product, there is a 78.94% chance that the person would be older than 23.
- The probability of a customer being married given that they bought the KP781 product is calculated.
- The probability of a customer being female given that they have a fitness level above 3 is calculated.
- The probability of a customer running more than 100 miles given that they use the product more than 3 times a week is calculated.
- The probability of a customer having an income above the median income given that they bought the KP281 product is calculated.
- The probability of a customer being single given that they have an education level above the median education level is calculated.

Graphical Analysis

Univariate Analysis

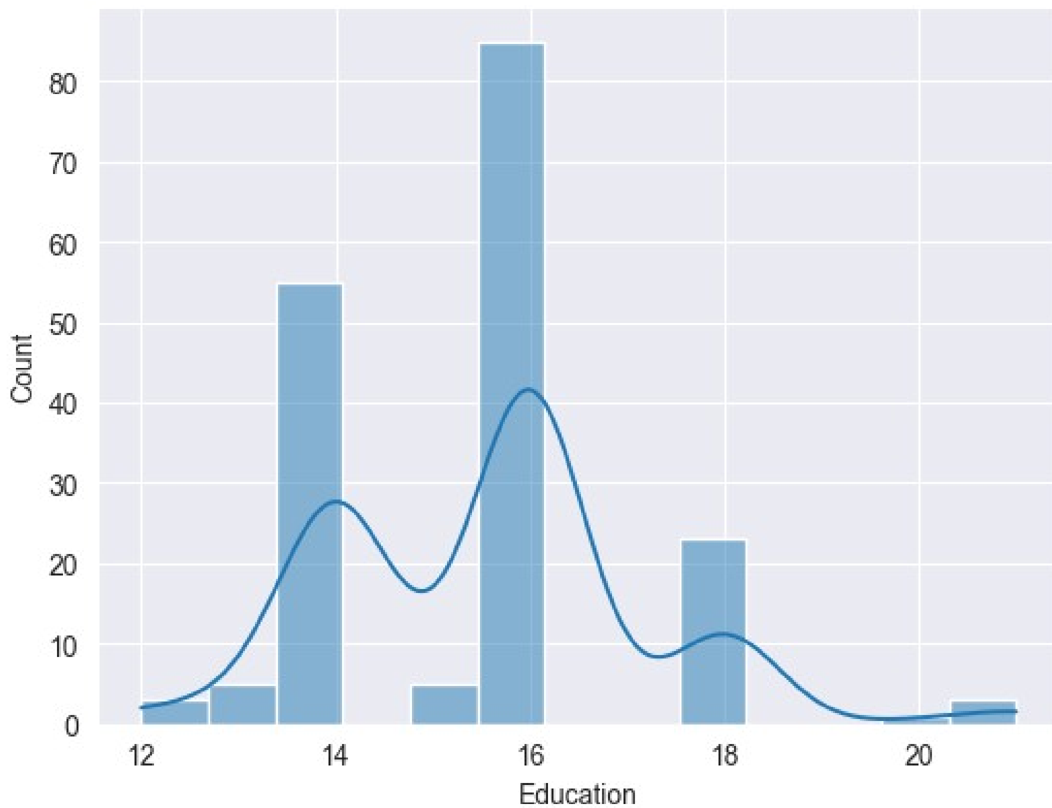
```
sns.histplot(df['Age'], kde = True)
```

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



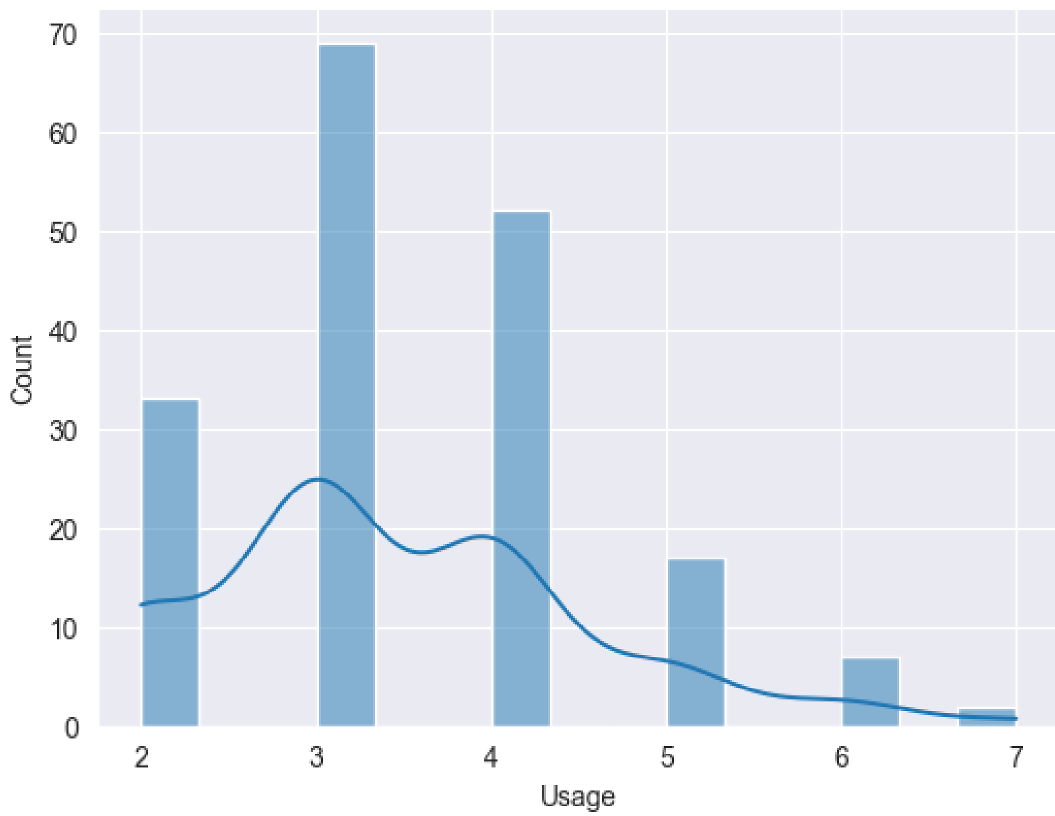
```python
sns.histplot(df['Education'], kde = True)
```

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



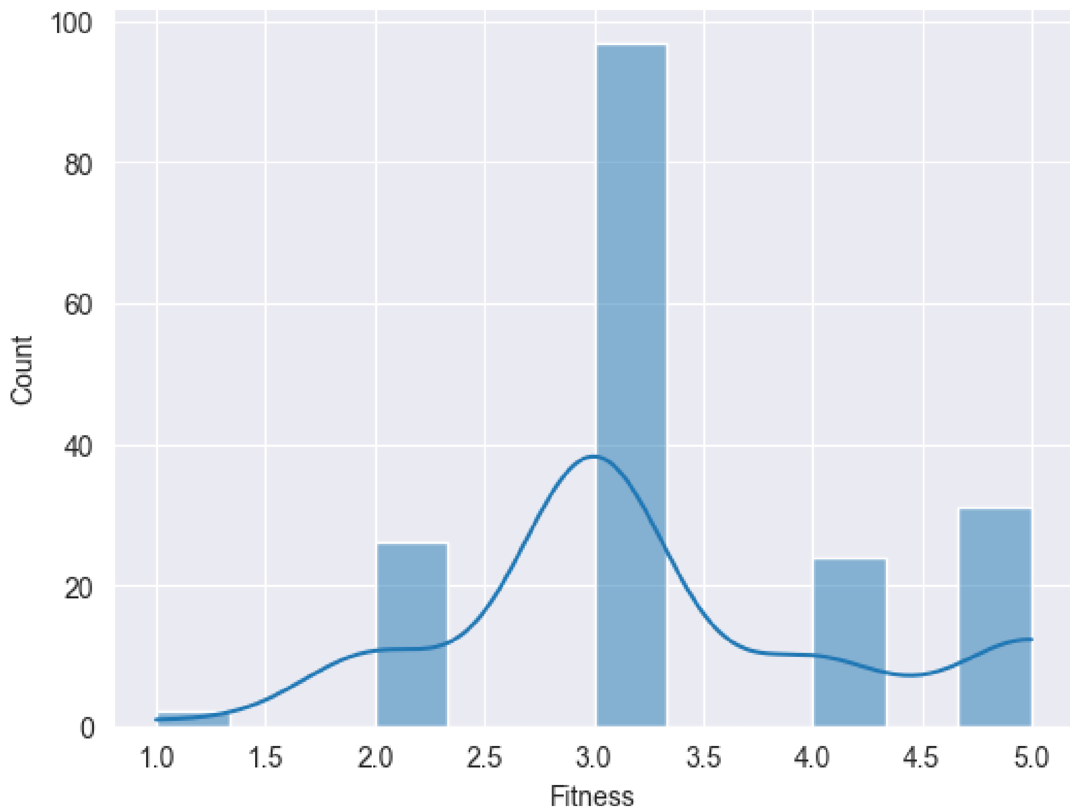

```python
sns.histplot(df['Usage'], kde = True)
```

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



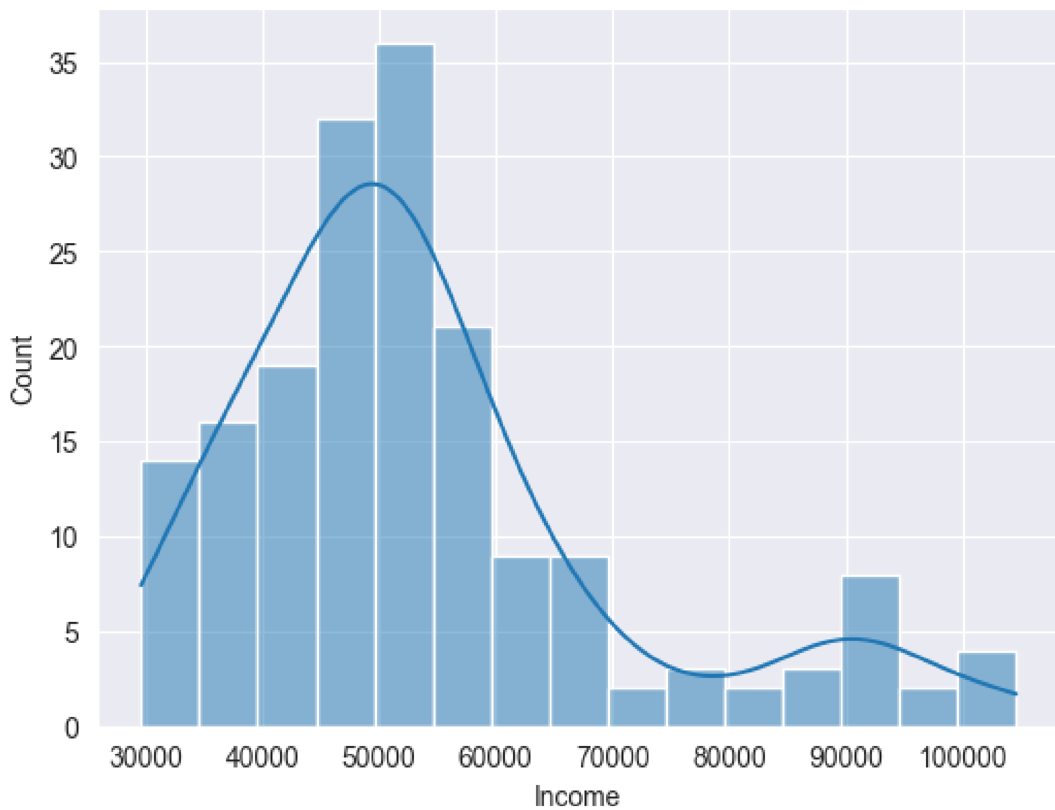

```
sns.histplot(df['Fitness'], kde = True)
```

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



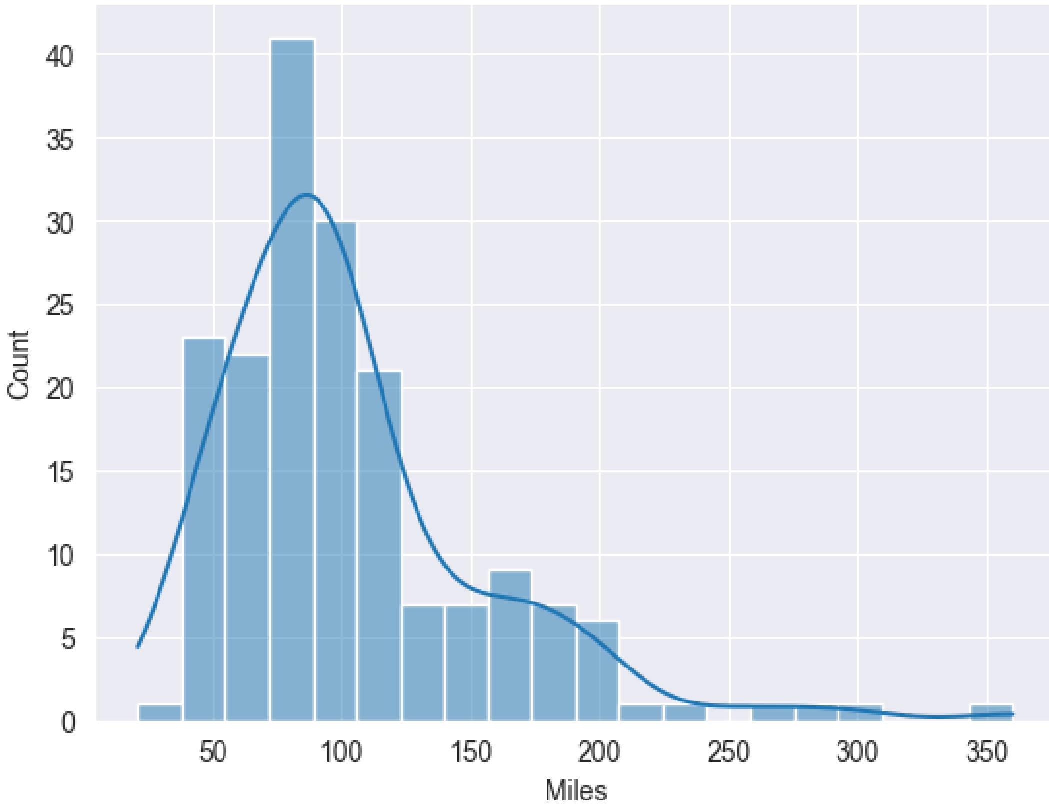

```
sns.histplot(df['Income'], kde = True)
```

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




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

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



Count plot of all the categorical columns Using histograms

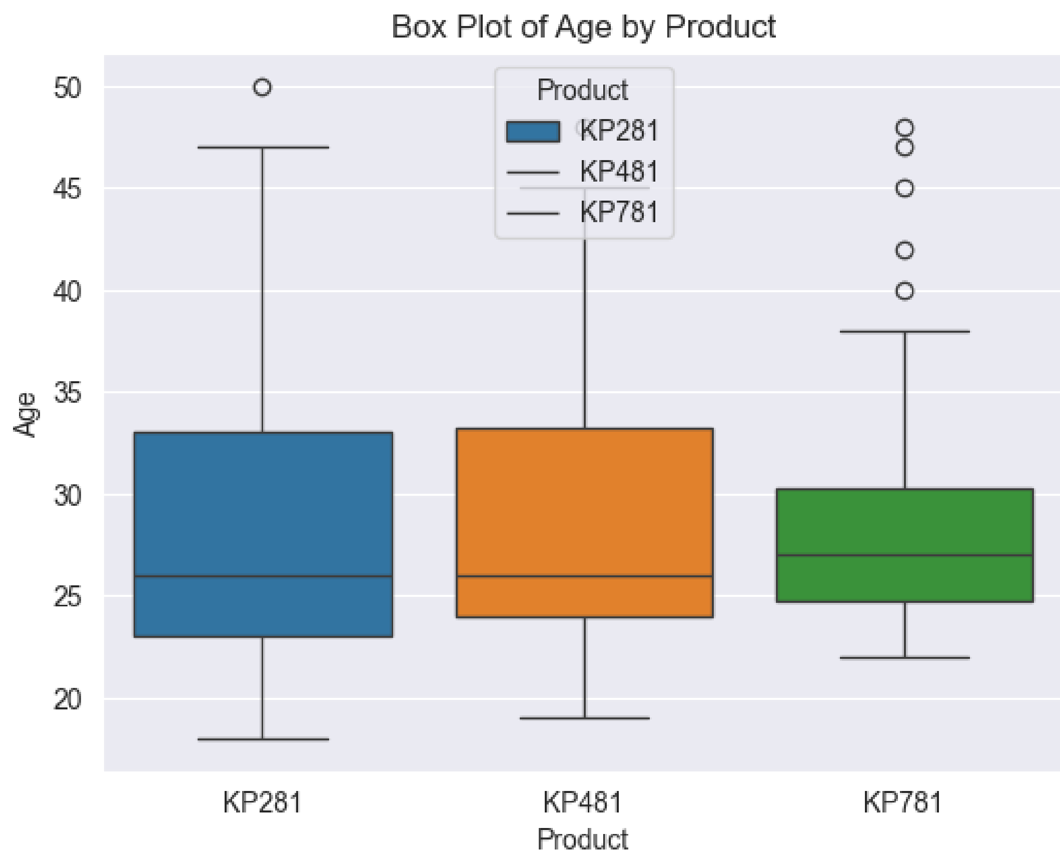
```
for col in categorical_columns:
    sns.countplot(x = col, data = df)
```

Bivariate Analysis

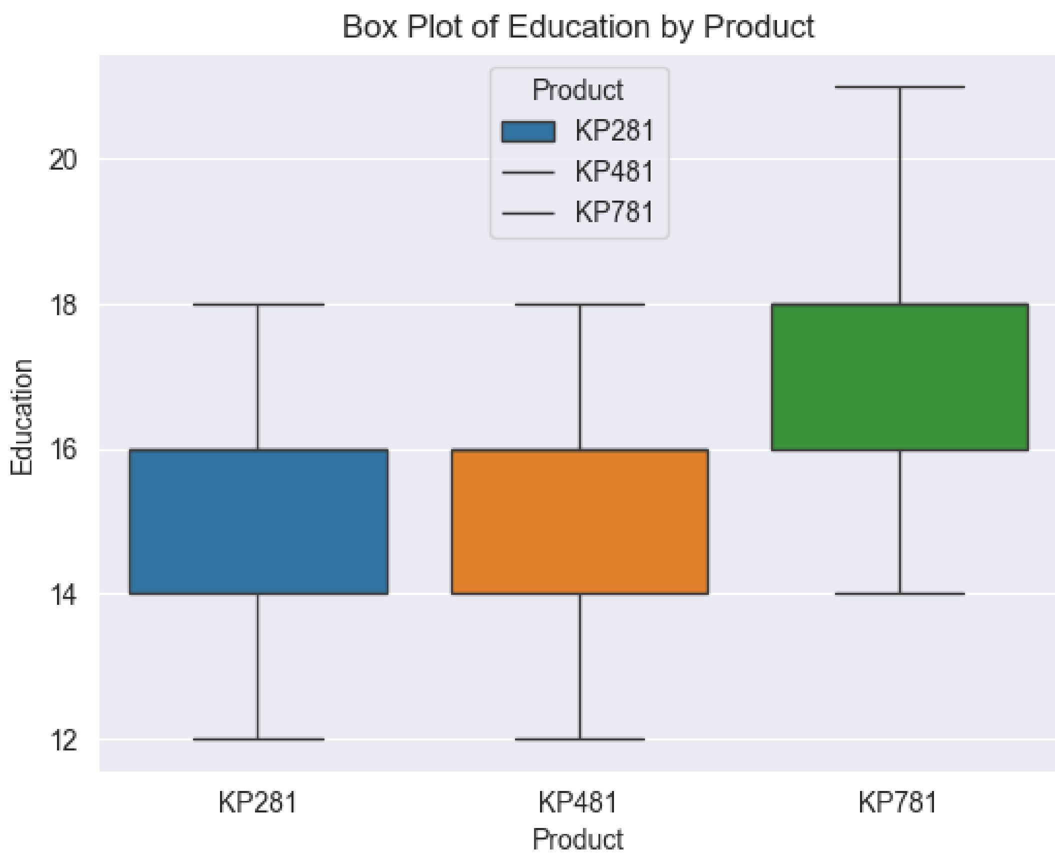
Box Plots for the columns Age, Education, Usage, Fitness, Income, Miles

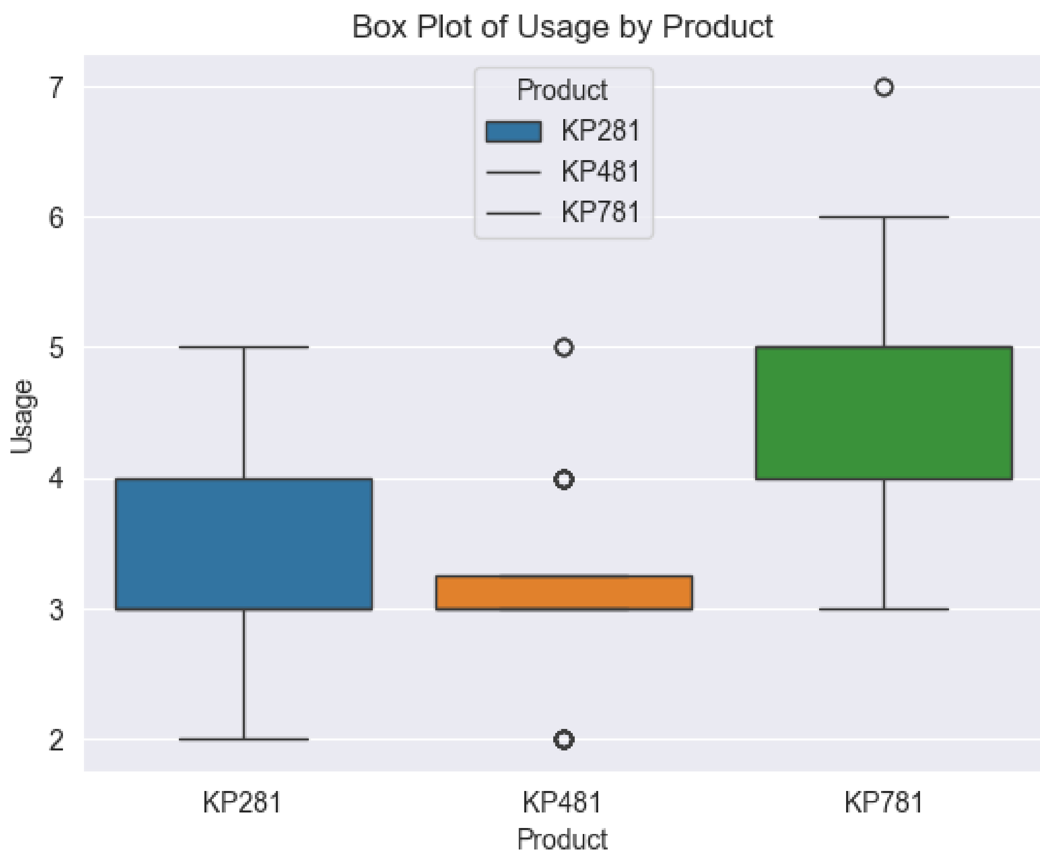
Here we are using box plots to compare the distribution of the columns Age, Education, Usage, Fitness, Income, Miles according to the product.

```
cols_for_box_plot = ['Age', 'Education', 'Usage', 'Fitness', 'Income', 
'Miles']
for col in cols_for_box_plot:
    sns.boxplot(x='Product', y=col, data=df, hue="Product")
    # Add appropriate plot titles and axis labels here
    plt.title(f'Box Plot of {col} by Product')
    plt.xlabel('Product')
    plt.ylabel(col)
    plt.legend(title='Product', loc='upper center', labels=['KP281', 
'KP481', 'KP781'])
    plt.show()
```

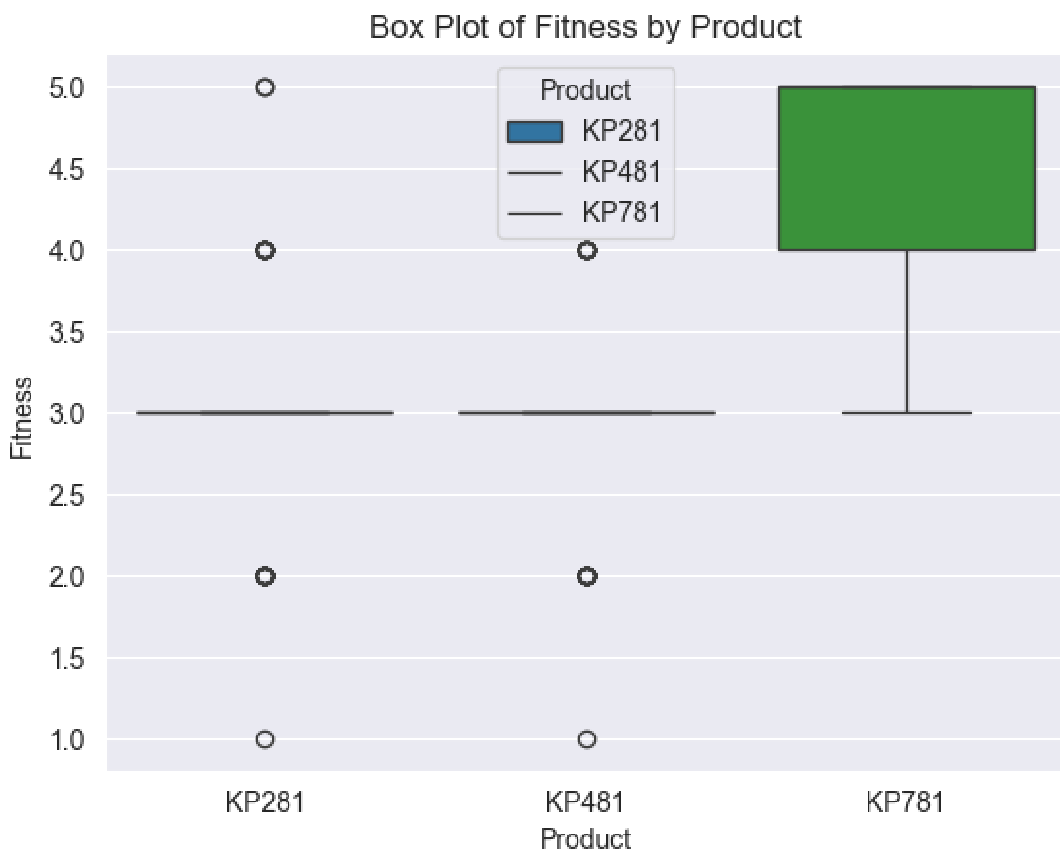


Box Plot of Age by Product
Product
KP281
KP481
KP781
50
45
40
35
30
25
20
Age
KP281
KP481
KP781
Product

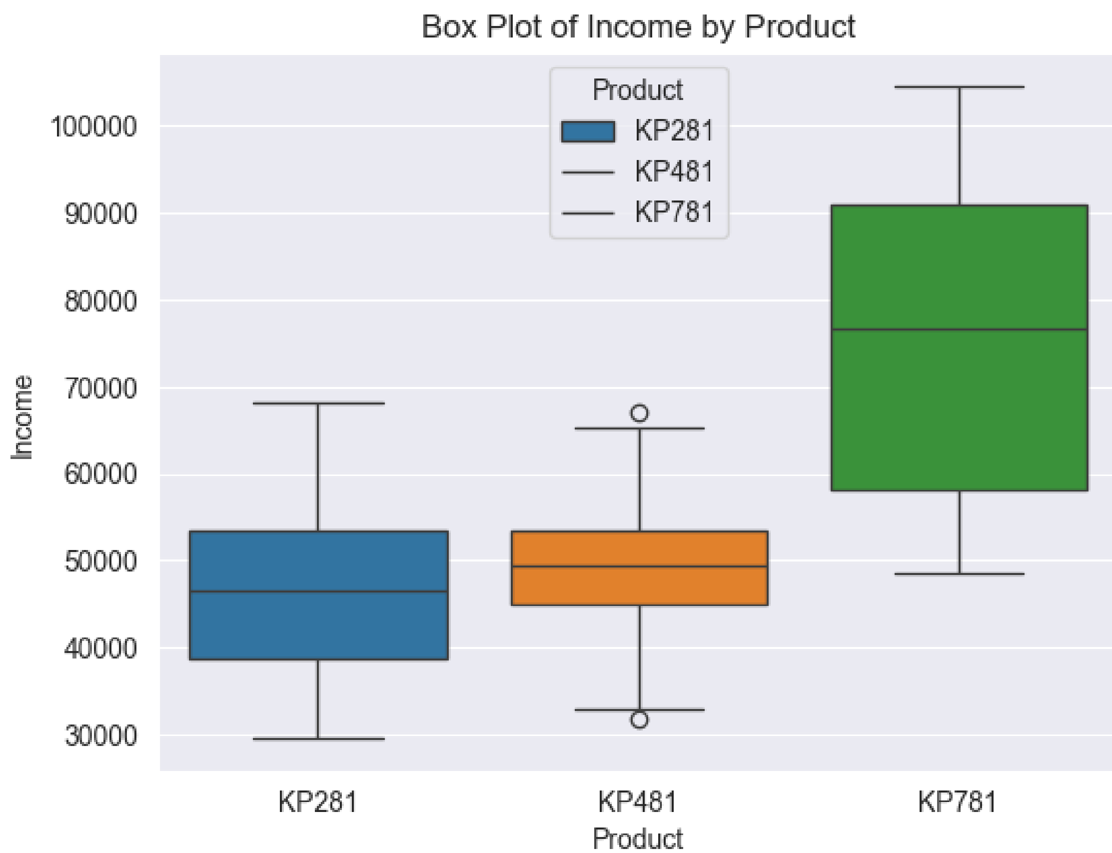

Box Plot of Education by Product
Product
KP281
KP481
KP781
20
18
16
14
12
Education
KP281
KP481
KP781
Product

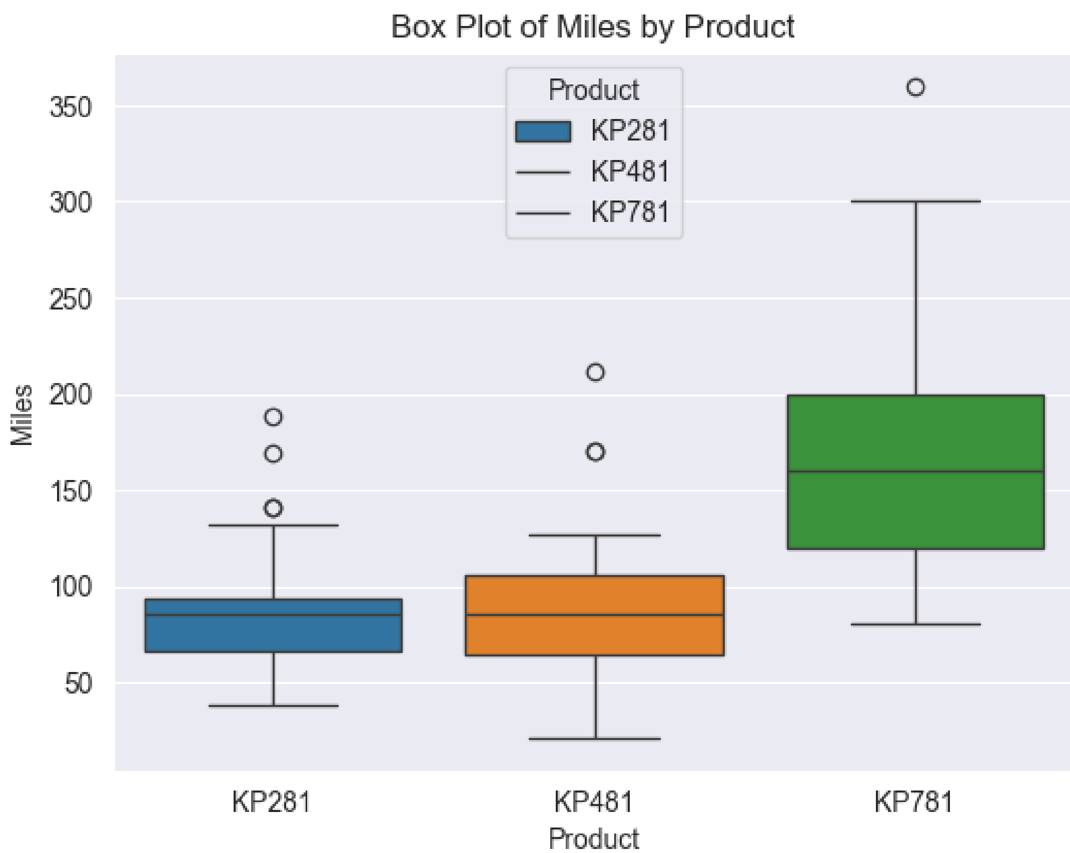


Box Plot of Usage by Product
Product
KP281
KP481
KP781
7
6
5
4
3
2
Usage
KP281
KP481
KP781
Product



Box Plot of Fitness by Product
Product
KP281
KP481
KP781
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
Fitness
KP281
KP481
KP781
Product

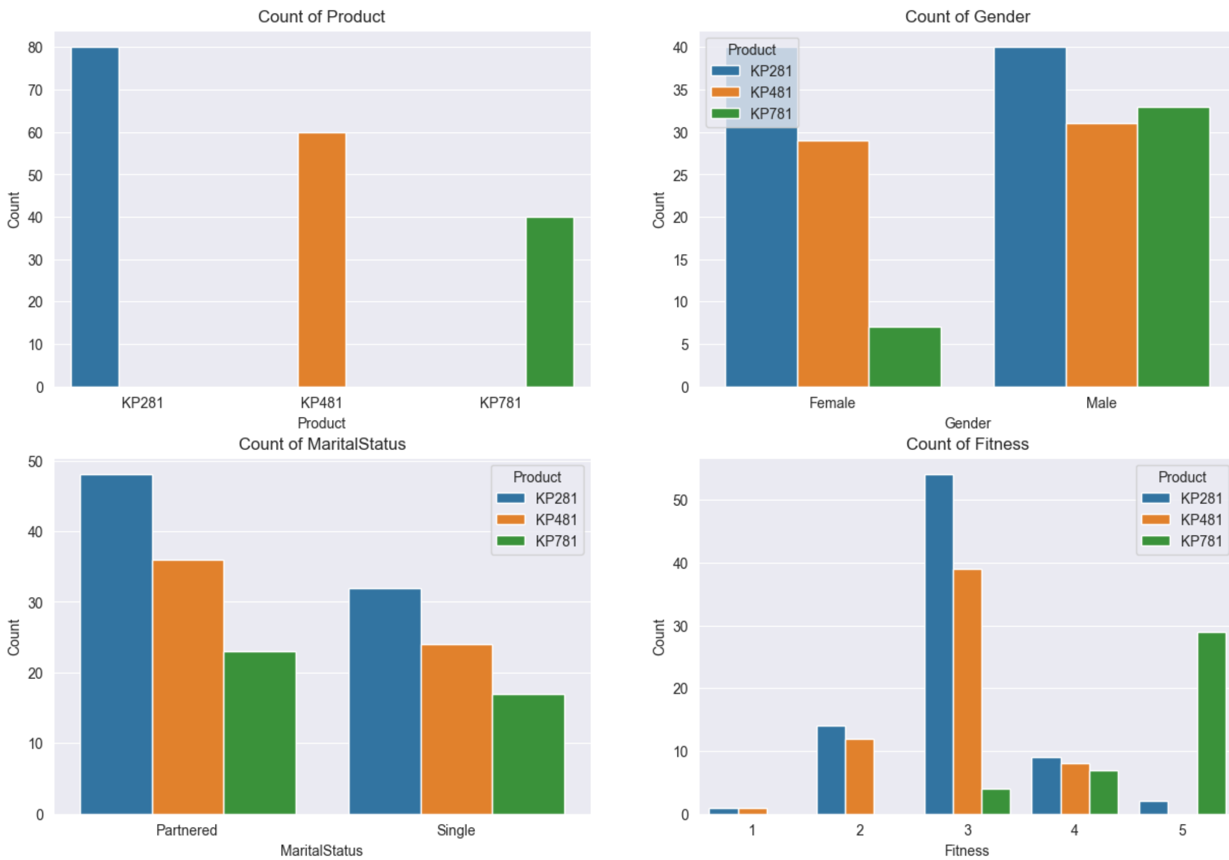

Box Plot of Income by Product
Product
KP281
KP481
KP781
100000
90000
80000
70000
60000
50000
40000
30000
Income
KP281
KP481
KP781
Product



SubPlots for Bivariate Analysis for the count of all the categorical columns

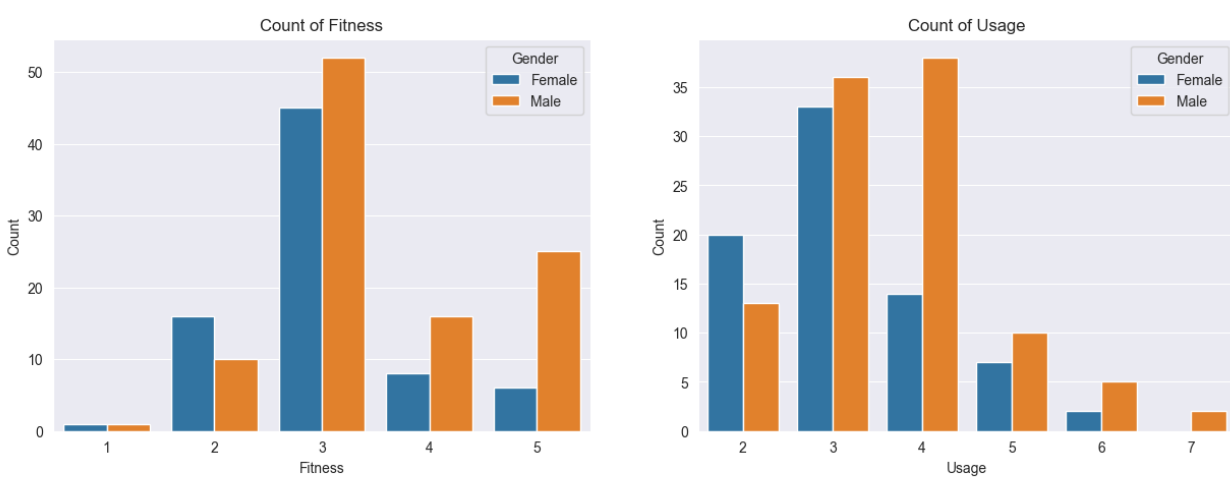
```
fig, axs = plt.subplots(2,2, figsize=(15,10))

for col in categorical_columns:
    sns.countplot(x = col, data = df, ax =
axs[categorical_columns.index(col) // 2,
categorical_columns.index(col) % 2], hue = 'Product', dodge = True)
    axs[categorical_columns.index(col) // 2,
categorical_columns.index(col) % 2].set_title(f'Count of {col}')
    axs[categorical_columns.index(col) // 2,
categorical_columns.index(col) % 2].set_xlabel(col)
    axs[categorical_columns.index(col) // 2,
categorical_columns.index(col) % 2].set_ylabel('Count')
```



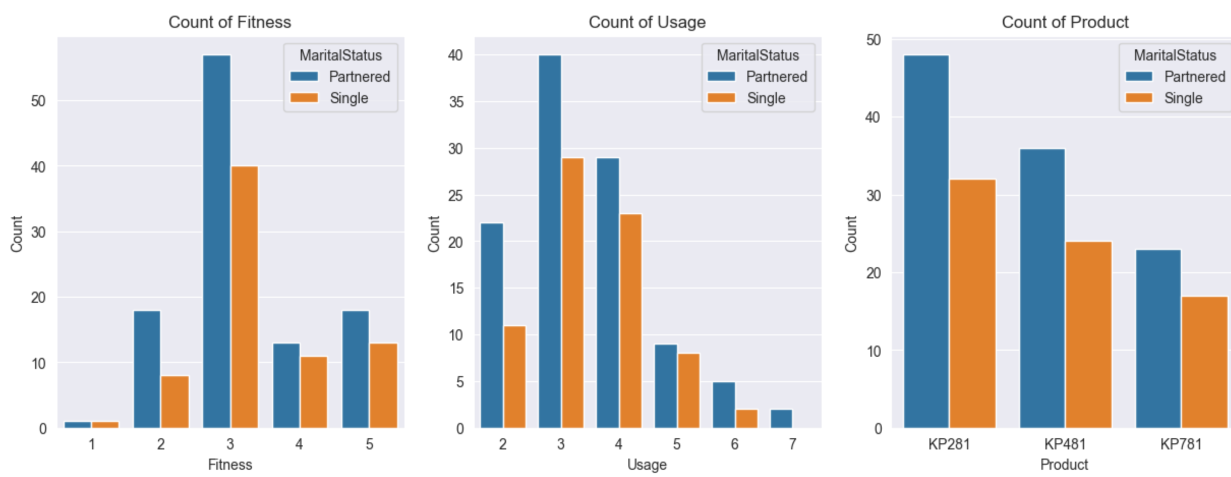
Subplots for the columns fitness, miles according to the gender.

```
fig, axs = plt.subplots(1,2, figsize=(15,5))
for col in ['Fitness', 'Usage']:
    sns.countplot(x = col, data = df, ax = axs[['Fitness',
'Usage'].index(col)], hue = "Gender")
    axs[['Fitness', 'Usage'].index(col)].set_title(f'Count of {col}')
    axs[['Fitness', 'Usage'].index(col)].set_xlabel(col)
    axs[['Fitness', 'Usage'].index(col)].set_ylabel('Count')
```



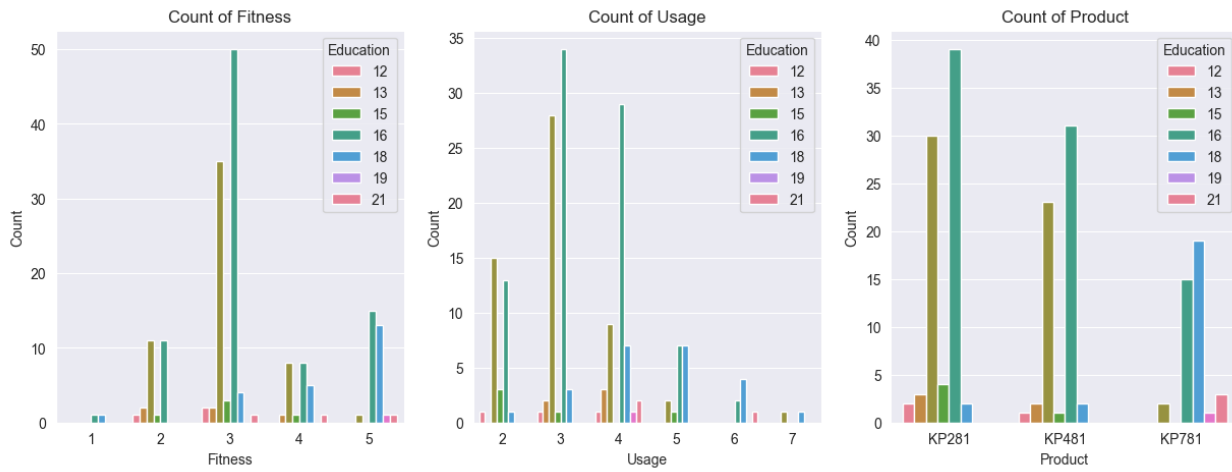
Subplots for the columns fitness, Usage and Product according to the MaritalStatus.

```
fig, axs = plt.subplots(1,3, figsize=(15,5))
for col in ['Fitness', 'Usage', 'Product']:
    sns.countplot(x = col, data = df, ax = axs[['Fitness', 'Usage',
'Product'].index(col)], hue = "MaritalStatus")
    axs[['Fitness', 'Usage', 'Product'].index(col)].set_title(f'Count
of {col}')
    axs[['Fitness', 'Usage', 'Product'].index(col)].set_xlabel(col)
    axs[['Fitness', 'Usage',
'Product'].index(col)].set_ylabel('Count')
```



Subplots for the columns fitness, Usage and Product according to the Education.

```
fig, axs = plt.subplots(1,3, figsize=(15,5))
for col in ['Fitness', 'Usage', 'Product']:
    sns.countplot(x = col, data = df, ax = axs[['Fitness', 'Usage',
'Product'].index(col)], hue = "Education", palette="husl")
    axs[['Fitness', 'Usage', 'Product'].index(col)].set_title(f'Count
of {col}')
    axs[['Fitness', 'Usage', 'Product'].index(col)].set_xlabel(col)
    axs[['Fitness', 'Usage',
'Product'].index(col)].set_ylabel('Count')
```



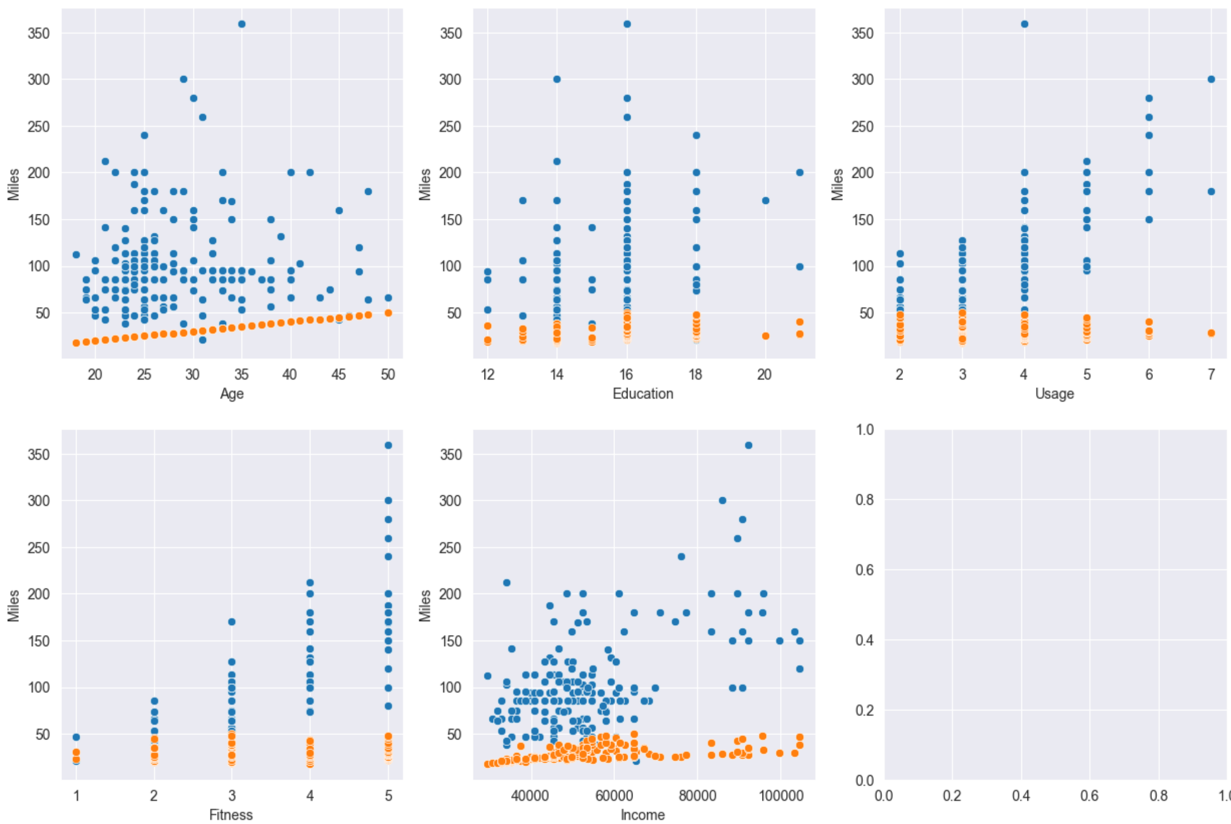
HeatMap for the correlation of all the columns in a dataframe

```
cols_for_heat_map = ['Age', 'Education', 'Usage', 'Fitness', 'Income',
'Miles']
```

```
<Axes: >
```

Scatter plots for the columns Age, Education, Usage, Fitness, Income with miles and age as subPlots

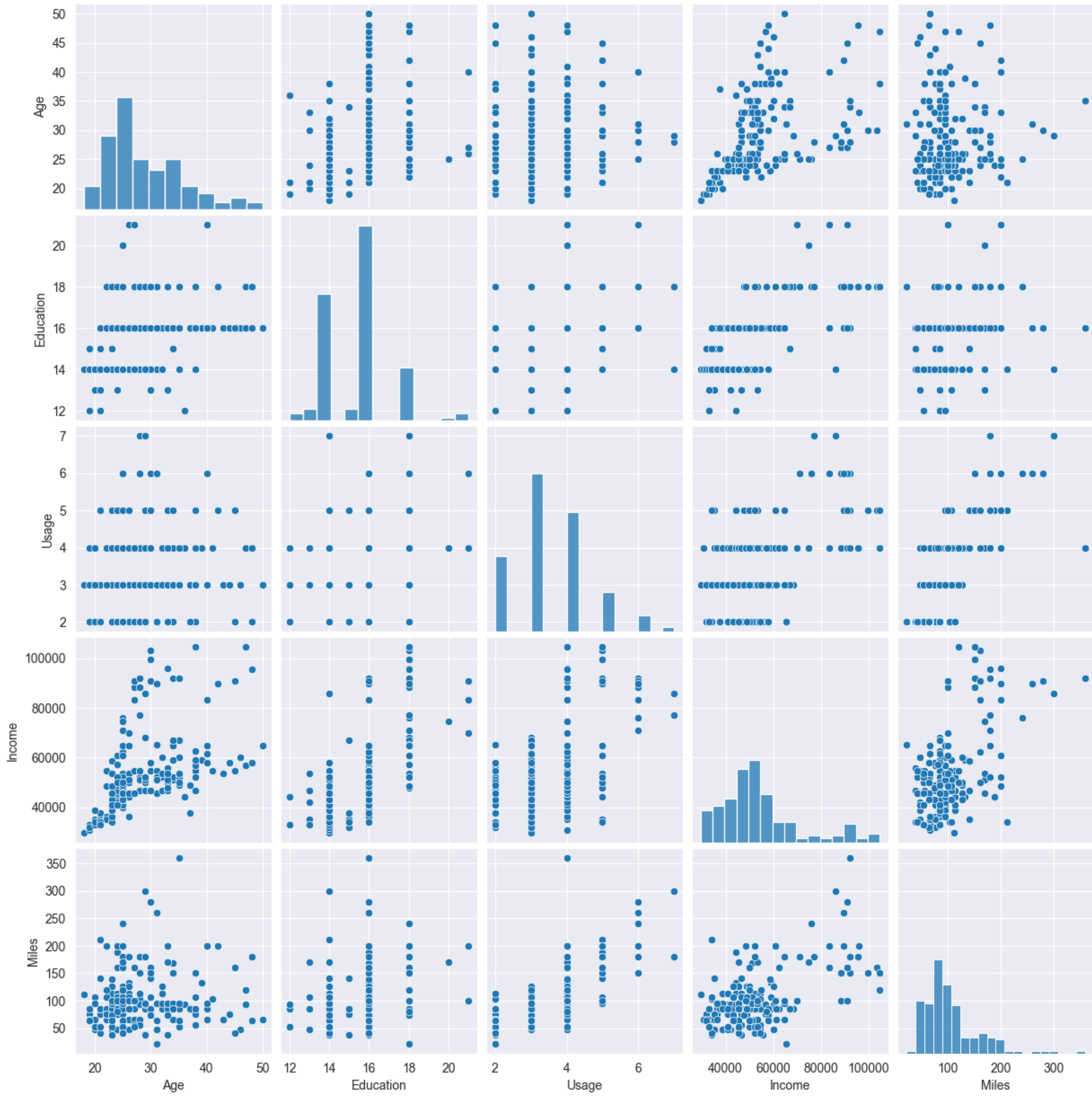
```
fig, axs = plt.subplots(2,3, figsize=(15,10))
for col in ["Age", "Education", "Usage", "Fitness", "Income"]:
    sns.scatterplot(x = col, y = 'Miles', data = df, ax =
axs[cols_for_heat_map.index(col) // 3, cols_for_heat_map.index(col) %
3])
```



Pairplot for the columns Age, Education, Usage, Fitness, Income, Miles with age as subPlots

```
sns.pairplot(df[cols_for_heat_map])
```

```
<seaborn.axisgrid.PairGrid at 0x1bd5a1f7430>
```



Checking for Missing values:

```
missing = df.isnull().sum()
missing

Product          0
Age              0
Gender           0
Education        0
MaritalStatus    0
Usage            0
Fitness          0
```

```
Income        0
Miles         0
dtype: int64
```

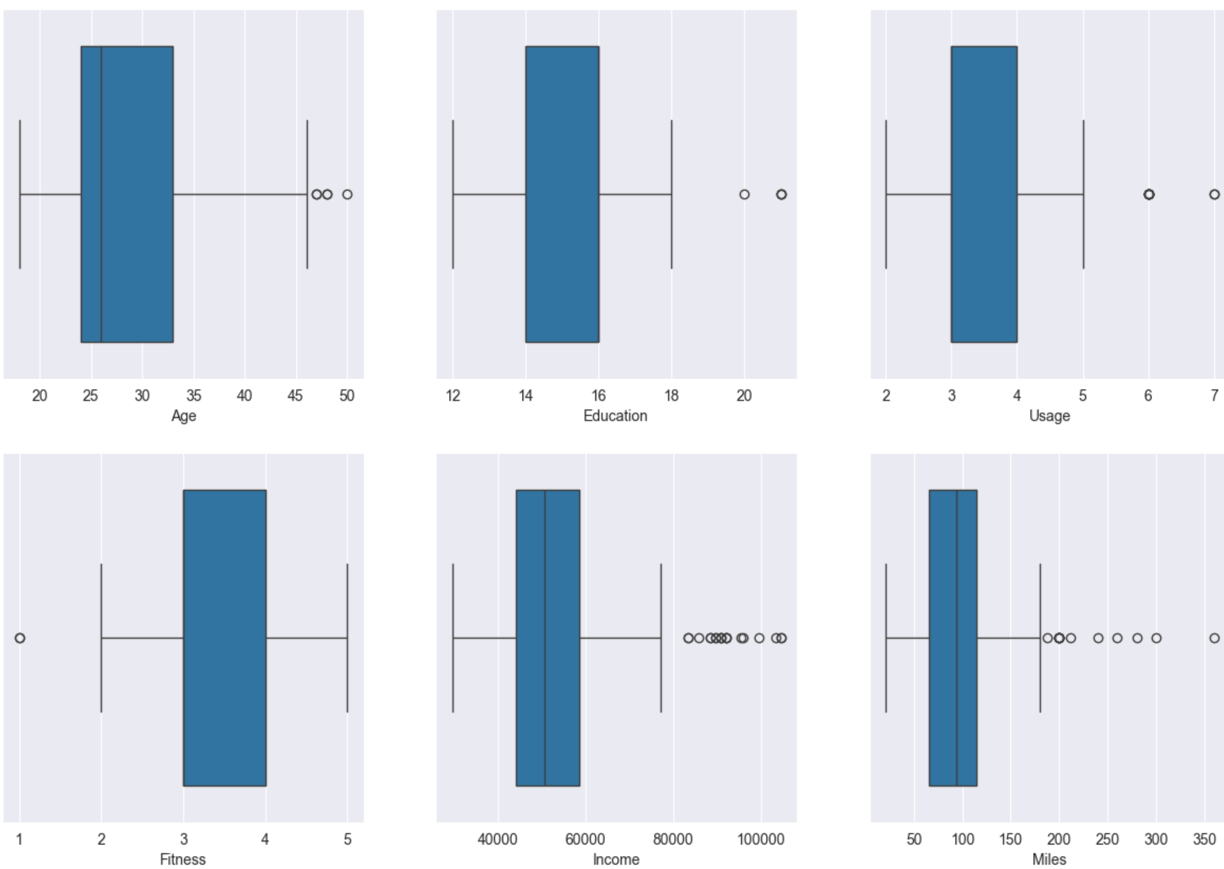
Observations:

- We can see that there are `no missing values` in the data.
- The data is `clean`.

Checking for Outliers:

Here I am using the box plots to get the outliers in each of the columns.

```
fig, axs = plt.subplots(2,3, figsize=(15,10))
for col in cols_for_box_plot:
    sns.boxplot(x = col, data = df, ax =
axs[cols_for_box_plot.index(col) // 3, cols_for_box_plot.index(col) %
3])
```




```
# Counting the number of outliers in the columns Age, Education, Usage

for col in ['Age', 'Education', 'Usage']:
```

```
    q1 = df[col].quantile(0.25)
    q3 = df[col].quantile(0.75)
    iqr = q3 - q1
    lower_bound = q1 - 1.5 * iqr
    upper_bound = q3 + 1.5 * iqr
    print(f'Number of outliers in {col} are:', df[(df[col] <
lower_bound) | (df[col] > upper_bound)].shape[0])
```

```
Number of outliers in Age are: 5
Number of outliers in Education are: 4
Number of outliers in Usage are: 9
```

Observations:

- We can see that there are outliers in the various column of the data.
- The outliers are present in the columns Age, Education, Usage, Fitness, Income, Miles.
- The number of outliers in the columns Age, Education, Usage are 5, 4, 9 respectively. Hence we could say that, that data would lead to incorrect analysis of the dataframe.

Customer Profiling According to the Product

```
df[df['Product'] == 'KP281'].describe()
```

	Age	Education	Usage	Income	Miles
count	80.000000	80.000000	80.000000	80.00000	80.000000
mean	28.550000	15.037500	3.087500	46418.02500	82.787500
std	7.221452	1.216383	0.782624	9075.78319	28.874102
min	18.000000	12.000000	2.000000	29562.00000	38.000000
25%	23.000000	14.000000	3.000000	38658.00000	66.000000
50%	26.000000	16.000000	3.000000	46617.00000	85.000000
75%	33.000000	16.000000	4.000000	53439.00000	94.000000
max	50.000000	18.000000	5.000000	68220.00000	188.000000

```
df[df['Product'] == 'KP481'].describe()
```

	Age	Education	Usage	Income	Miles
count	60.000000	60.000000	60.000000	60.000000	60.000000
mean	28.900000	15.116667	3.066667	48973.650000	87.933333
std	6.645248	1.222552	0.799717	8653.989388	33.263135
min	19.000000	12.000000	2.000000	31836.000000	21.000000
25%	24.000000	14.000000	3.000000	44911.500000	64.000000
50%	26.000000	16.000000	3.000000	49459.500000	85.000000
75%	33.250000	16.000000	3.250000	53439.000000	106.000000
max	48.000000	18.000000	5.000000	67083.000000	212.000000

```
df[df['Product'] == 'KP781'].describe()
```

	Age	Education	Usage	Income	Miles
count	40.000000	40.000000	40.000000	40.00000	40.000000

```
mean   29.100000  17.325000   4.775000   75441.57500  166.900000
std     6.971738   1.639066   0.946993   18505.83672   60.066544
min    22.000000  14.000000   3.000000   48556.00000   80.000000
25%    24.750000  16.000000   4.000000   58204.75000  120.000000
50%    27.000000  18.000000   5.000000   76568.50000  160.000000
75%    30.250000  18.000000   5.000000   90886.00000  200.000000
max    48.000000  21.000000   7.000000  104581.00000  360.000000
```

Observations:

- We can see that the customers who bought the product *KP281* are *younger* than the customers who bought the product KP481 and KP781.
- The customers who bought the product *KP281 have a higher education* than the customers who bought the product KP481 and KP781.
- The customers who bought the product KP281 use the product more than the customers who bought the product KP481 and KP781.
- The customers who bought the product KP281 have a higher fitness level than the customers who bought the product KP481 and KP781.
- The customers who bought the product KP281 have a higher income than the customers who bought the product KP481 and KP781.
- The customers who bought the product KP281 run more miles than the customers who bought the product KP481 and KP781.
- The customers who bought the product KP481 are older than the customers who bought the product KP281 and KP781.
- The customers who bought the product KP481 have a lower education than the customers who bought the product KP281 and KP781.
- The customers who bought the product KP481 use the product less than the customers who bought the product KP281 and KP781.
- The customers who bought the product KP481 have a lower fitness level than the customers who bought the product KP281 and KP781.
- The customer who bought the product KP781 are older than the customers who bought the product KP281 and KP481.
- The customers who bought the product KP781 have a lower education than the customers who bought the product KP281 and KP481.
- The customers who bought the product KP781 use the product less than the customers who bought the product KP281 and KP481.
- The customers who bought the product KP781 have a lower fitness level than the customers who bought the product KP281 and KP481.

Recommendations:

- The company should focus on marketing the KP281 product more as it has the highest sales and is popular among younger, more educated customers with a higher fitness level.

- The company should also consider improving the features of the KP481 and KP781 products to attract more customers. This could include adding more advanced features or offering discounts to attract a wider customer base.
- The company should also consider targeting more female and single customers as there is potential for growth in these segments.
- The company could also consider offering fitness programs or partnerships with fitness influencers to attract customers who are looking to improve their fitness level.
- The company should also consider conducting customer surveys to understand the needs and preferences of their customers better. This will help in improving their products and services and in turn, increase sales.
- The company should also consider offering financing options to attract customers with lower income levels.
- The company should also consider offering a wider range of products to cater to different customer needs and preferences.

Conclusion:

- Focus on marketing the KP281 product more as it has the highest sales and is popular among younger, more educated customers with a higher fitness level.
- Consider improving the features of the KP481 and KP781 products to attract more customers. This could include adding more advanced features or offering discounts to attract a wider customer base.
- Consider targeting more female and single customers as there is potential for growth in these segments.
- Consider offering fitness programs or partnerships with fitness influencers to attract customers who are looking to improve their fitness level.
- Conduct customer surveys to understand the needs and preferences of their customers better. This will help in improving their products and services and in turn, increase sales.
- Consider offering financing options to attract customers with lower income levels.
- Consider offering a wider range of products to cater to different customer needs and preferences.