Business Case: Aerofit - Descriptive Statistics & Probability by shahnaz

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

Input:

```
# read the data
data = pd.read_csv('aerofit_treadmill.csv')
data
```

Output:

	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

```
# first 5 rows
data.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

```
# last 5 rows
data.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

```
#check data types
data.dtypes
```

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

```
#check shape of data
data.shape
```

```
(180, 9)
```

```
# check data info
data.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
```

```
## Checking number of Rows and Columns:
print(f"Number of Rows:{data.shape[0]} \nNumber of
Columns:{data.shape[1]}")
```

```
Number of Rows:180
Number of Columns:9
```

```
#check is there any null values
data.isna().sum()
```

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

```
# If there is any null value then return True otherwise False
data.isnull().any()
```

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

```
# count unique values
data.nunique()
```

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

```
#  age calculation
data['Age'].describe().round()
```

```
count    180.0
mean      29.0
std        7.0
min       18.0
25%       24.0
50%       26.0
75%       33.0
max       50.0
Name: Age, dtype: float64
```

```
# count numbers of male and female

data['Gender'].value_counts()
Output:
Income 16506.684226
Miles 51.863605
dtype: float64
```

Observations:

1-There are no missing values in the data.

2-There are 3 unique products in the dataset.

3-KP281 is the most frequent product.

4- Minimum & Maximum age of the person is 18 & 50, mean is 28.79 and 75% of persons have age less than or equal to 33.

5-Most of the people are having 16 years of education i.e., 75% of persons are having education <= 16 years.

6- Out of 180 data points, 104's gender is Male and rest are the female.

7- Standard deviation for Income & Miles is very high. These variables might have the outliers in it.

statistical summary:

This function provides summary statistics (count, mean, std, min, 25%, 50%, 75%, max) for numerical columns in the DataFrame, allowing us to understand the central tendency,spread, and distribution of the data.

```
# for numerical columns
data.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Age	180.0	28.788889	6.943498	18.0	24.00	26.0	33.00	50.0
Education	180.0	15.572222	1.617055	12.0	14.00	16.0	16.00	21.0
Usage	180.0	3.455556	1.084797	2.0	3.00	3.0	4.00	7.0
Fitness	180.0	3.311111	0.958869	1.0	3.00	3.0	4.00	5.0
Income	180.0	53719.577778	16506.684226	29562.0	44058.75	50596.5	58668.00	104581.0
Miles	180.0	103.194444	51.863605	21.0	66.00	94.0	114.75	360.0

```python
# for categorical columns
data.describe(include = 'object').T
```

	count	unique	top	freq
Product	180	3	KP281	80
Gender	180	2	Male	104
MaritalStatus	180	2	Partnered	107

```python
# Get the count the  number of each type in the 'Product' column
data['Product'].value_counts()
```

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

Univariate Analysis:

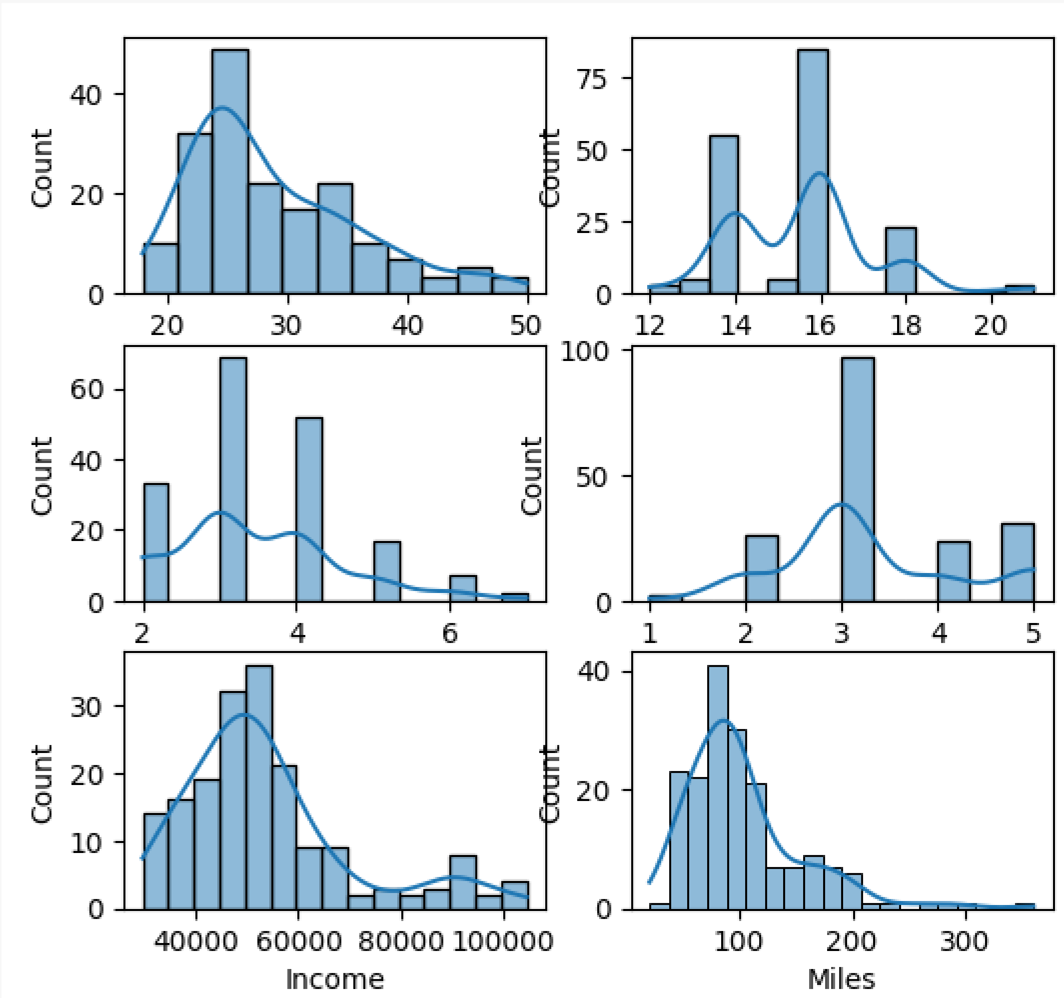
Understanding the distribution of the data for the quantitative attributes:

1. Age
2. Education
3. Usage
4. Fitness
5. Income
6. Miles

7. Histplot

```python
# Histplot for continuous variables
fig, axis = plt.subplots(nrows=3, ncols=2, figsize=(6, 4))
fig.subplots_adjust(top=1.2)

sns.histplot(data=data, x="Age", kde=True, ax=axis[0,0])

sns.histplot(data=data, x="Education", kde=True, ax=axis[0,1])

sns.histplot(data=data, x="Usage", kde=True, ax=axis[1,0])

sns.histplot(data=data, x="Fitness", kde=True, ax=axis[1,1])
```

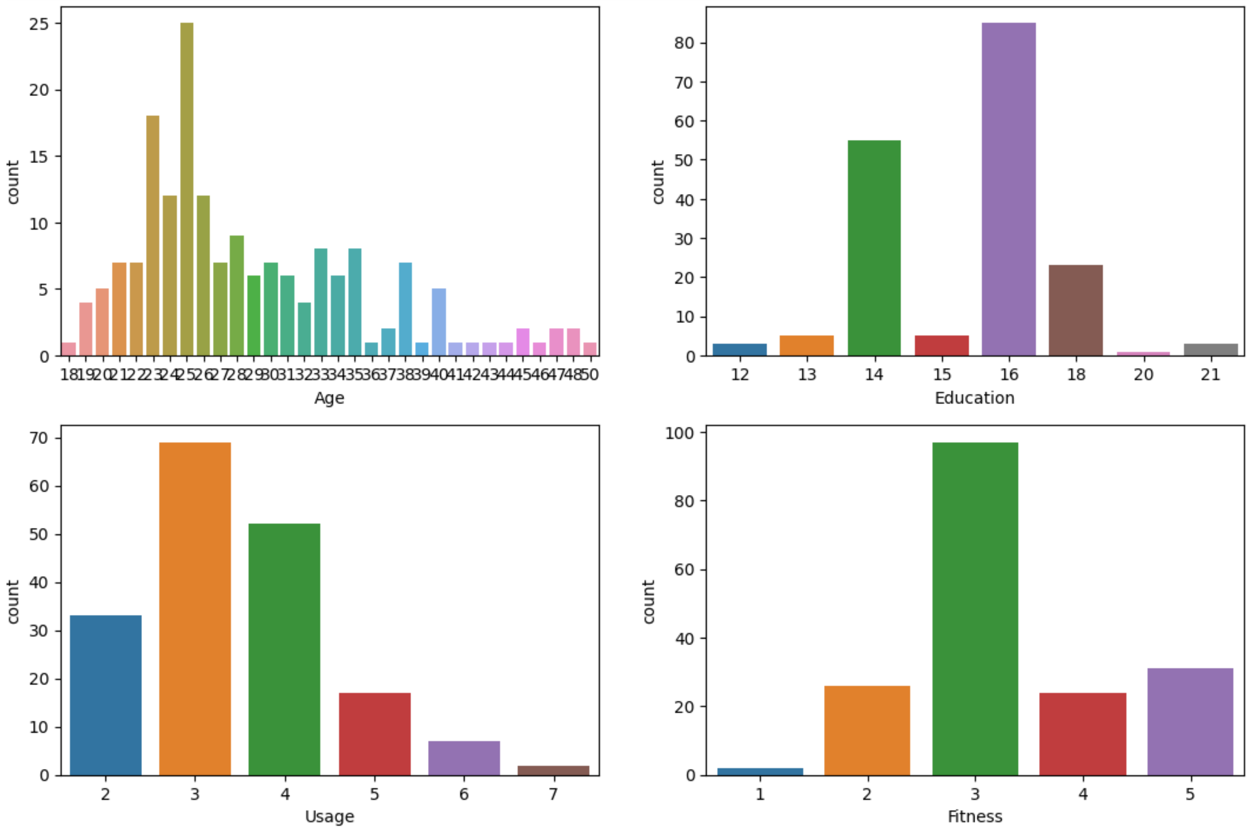
```
sns.histplot(data=data, x="Income", kde=True, ax=axis[2,0])

sns.histplot(data=data, x="Miles", kde=True, ax=axis[2,1])

plt.show()
```



Countplot

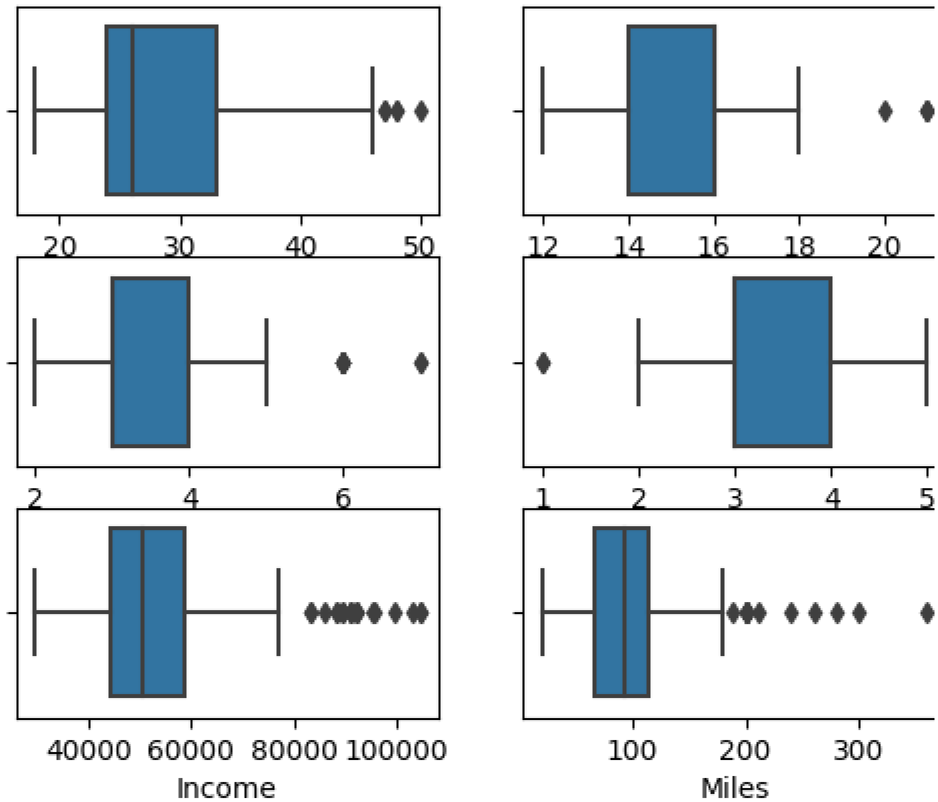
```
# countplot for continuous variables
fig, axis = plt.subplots(nrows=2, ncols=2, figsize=(13, 6))
fig.subplots_adjust(top = 1.2)
sns.countplot(data=data, x="Age", ax=axis[0,0])

sns.countplot(data=data, x="Education", ax=axis[0,1])

sns.countplot(data=data, x="Usage", ax=axis[1,0])
```

```
sns.countplot(data=data, x="Fitness", ax=axis[1,1])

plt.show()
```



Boxplot

```
# boxplot for continuous variables
fig, axis = plt.subplots(nrows=3, ncols=2, figsize=(6, 4))
fig.subplots_adjust(top=1.0)
sns.boxplot(data=data, x="Age", orient='h', ax=axis[0,0])
sns.boxplot(data=data, x="Education", orient='h', ax=axis[0,1])
sns.boxplot(data=data, x="Usage", orient='h', ax=axis[1,0])
sns.boxplot(data=data, x="Fitness", orient='h', ax=axis[1,1])
sns.boxplot(data=data, x="Income", orient='h', ax=axis[2,0])
sns.boxplot(data=data, x="Miles", orient='h', ax=axis[2,1])
plt.show()
```



Observations:

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.

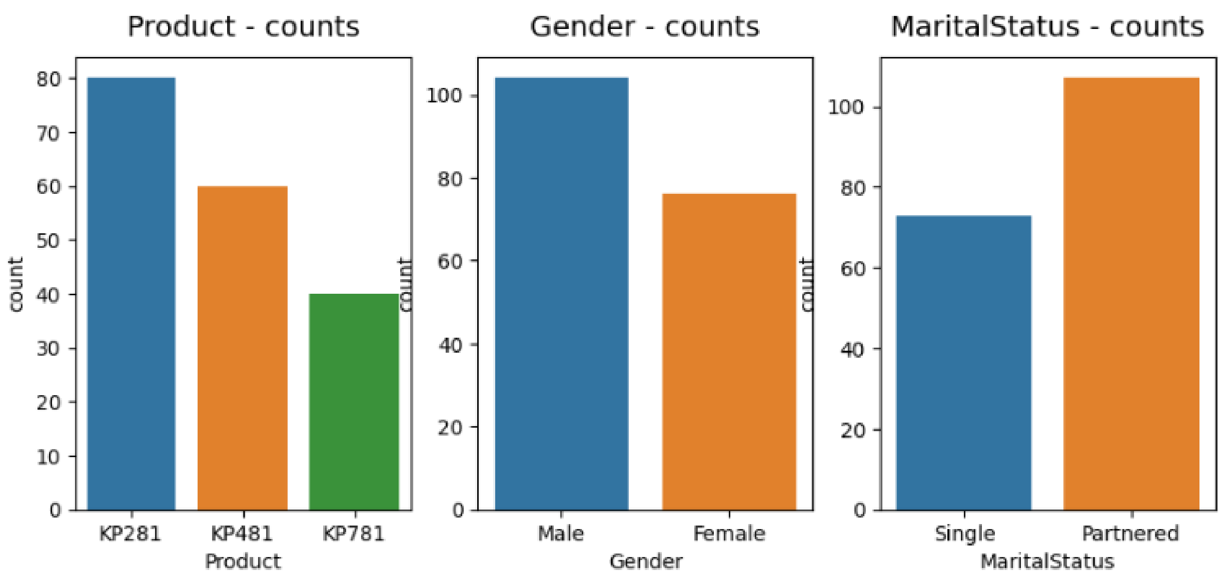
Understanding the distribution of the data for the qualitative attributes:

1. Product
2. Gender
3. MaritalStatus

Countplot for categorical variables

```
# countplot for categorical variables

fig, axis = plt.subplots(nrows=1, ncols=3, figsize=(10,4))
sns.countplot(data=data, x='Product', ax=axis[0])
sns.countplot(data=data, x='Gender', ax=axis[1])
sns.countplot(data=data, x='MaritalStatus', ax=axis[2])
axis[0].set_title("Product - counts", pad=10, fontsize=14)
```

```
axis[1].set_title("Gender - counts", pad=10, fontsize=14)
axis[2].set_title("MaritalStatus - counts", pad=10,
fontsize=14)
plt.show()
```



Observations

KP281 is the most frequent product.

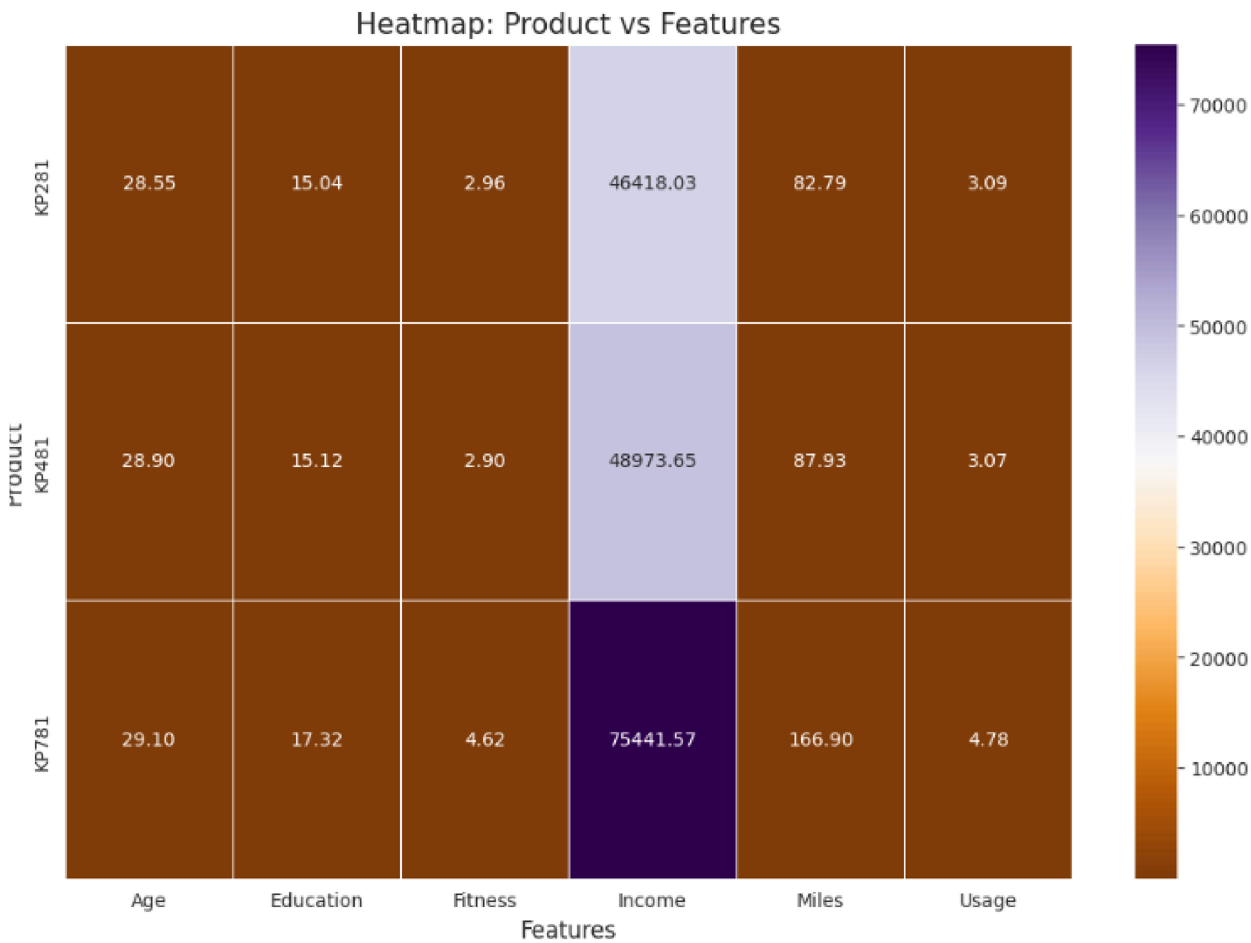
There are more Males in the data than Females.

More Partnered persons are there in the data.

Heatmap

```
# Heatmap for continuous variables

cols = ['Age', 'Education', 'Usage', 'Fitness', 'Income', 'Miles']

heatmap_data = data[['Product'] + cols]

heatmap_data_pivot = heatmap_data.pivot_table(index='Product',
values=cols)

plt.figure(figsize=(12, 8))
sns.set_style("white")
sns.heatmap(heatmap_data_pivot, cmap='PuOr', annot=True, fmt=".2f",
linewidths=.5)
```

```
plt.title("Heatmap: Product vs Features", fontsize=15)
plt.xlabel("Features", fontsize=12)
plt.ylabel("Product", fontsize=12)

plt.show()
```




```
# analysis with catg cols
data_1 = data[['Product', 'Gender', 'MaritalStatus']].melt()
data_1.groupby(['variable', 'value'])[['value']].count() / len(data)
```

		value
variable	**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

Observations

Product

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.

Gender

57.78% of the customers are Male.

MaritalStatus

59.44% of the customers are Partnered.

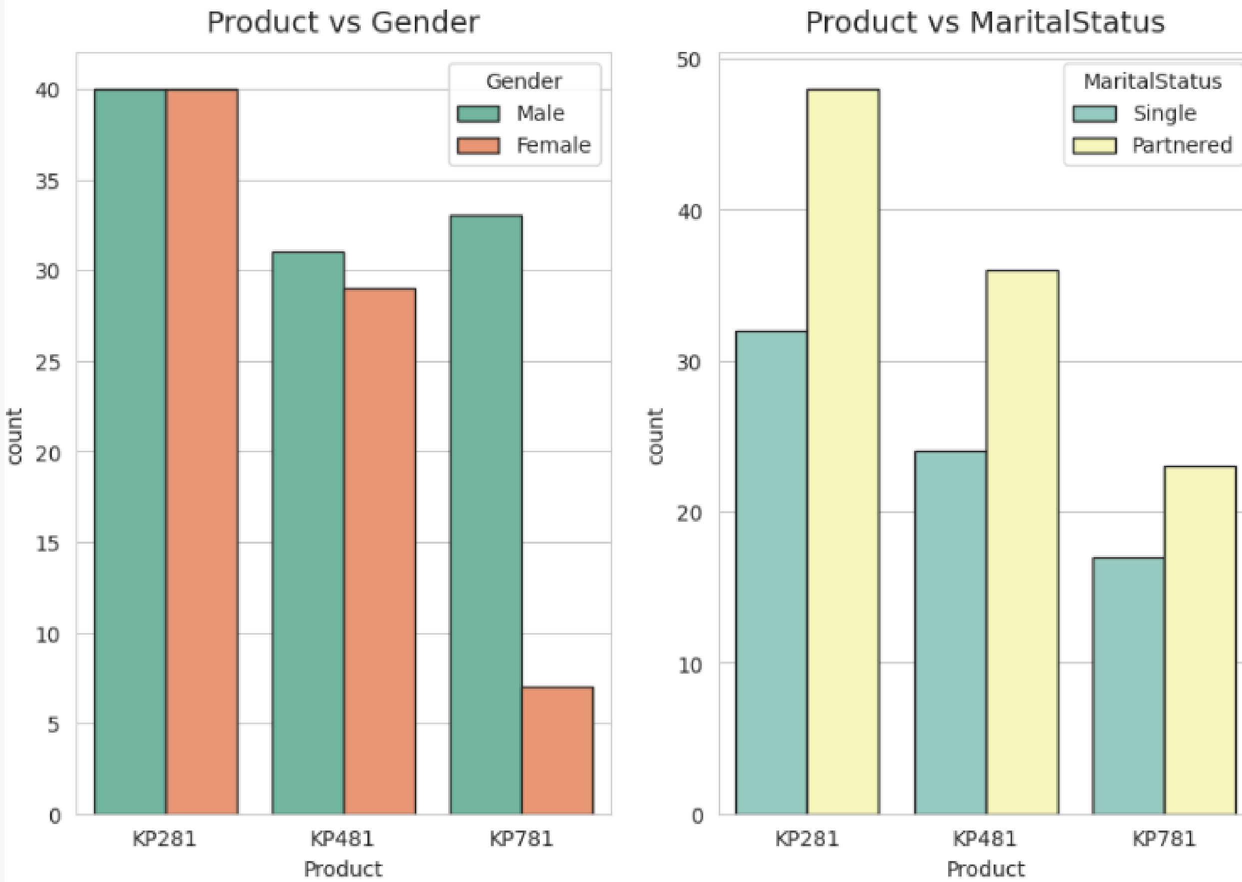
Bivariate Analysis

Checking if features - Gender or MaritalStatus have any effect on the product purchased.

Countplot

```
# countplot for categorical variables

sns.set_style(style='whitegrid')
fig, axis = plt.subplots(nrows=1, ncols=2, figsize=(10, 6.5))
sns.countplot(data=data, x='Product', hue='Gender', edgecolor="0.15",
palette='Set2', ax=axis[0])
sns.countplot(data=data, x='Product', hue='MaritalStatus',
edgecolor="0.15", palette='Set3', ax=axis[1])
axis[0].set_title("Product vs Gender",pad=10, fontsize=14)
```

```
axis[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.

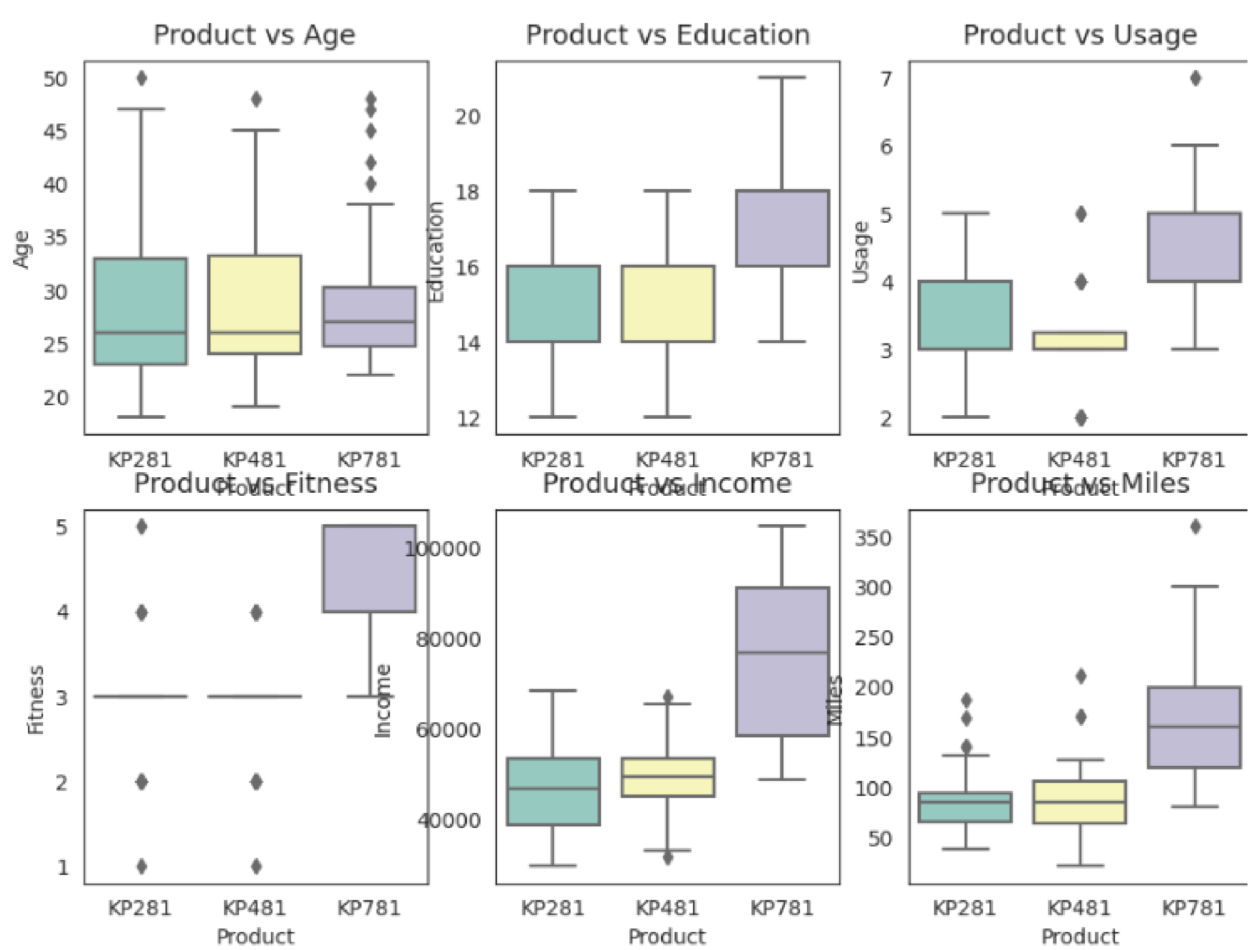
Product vs MaritalStatus

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

Checking if following features have any effect on the product purchased:

1. Age
2. Education
3. Usage
4. Fitness
5. Income
6. Miles

```
cols = ['Age', 'Education', 'Usage', 'Fitness', 'Income', 'Miles']
sns.set_style("white")
fig, axis = 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=data, x='Product', y=cols[count],
    ax=axis[i,j], palette='Set3')
    axis[i,j].set_title(f"Product vs {cols[count]}",
    pad=8, 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.

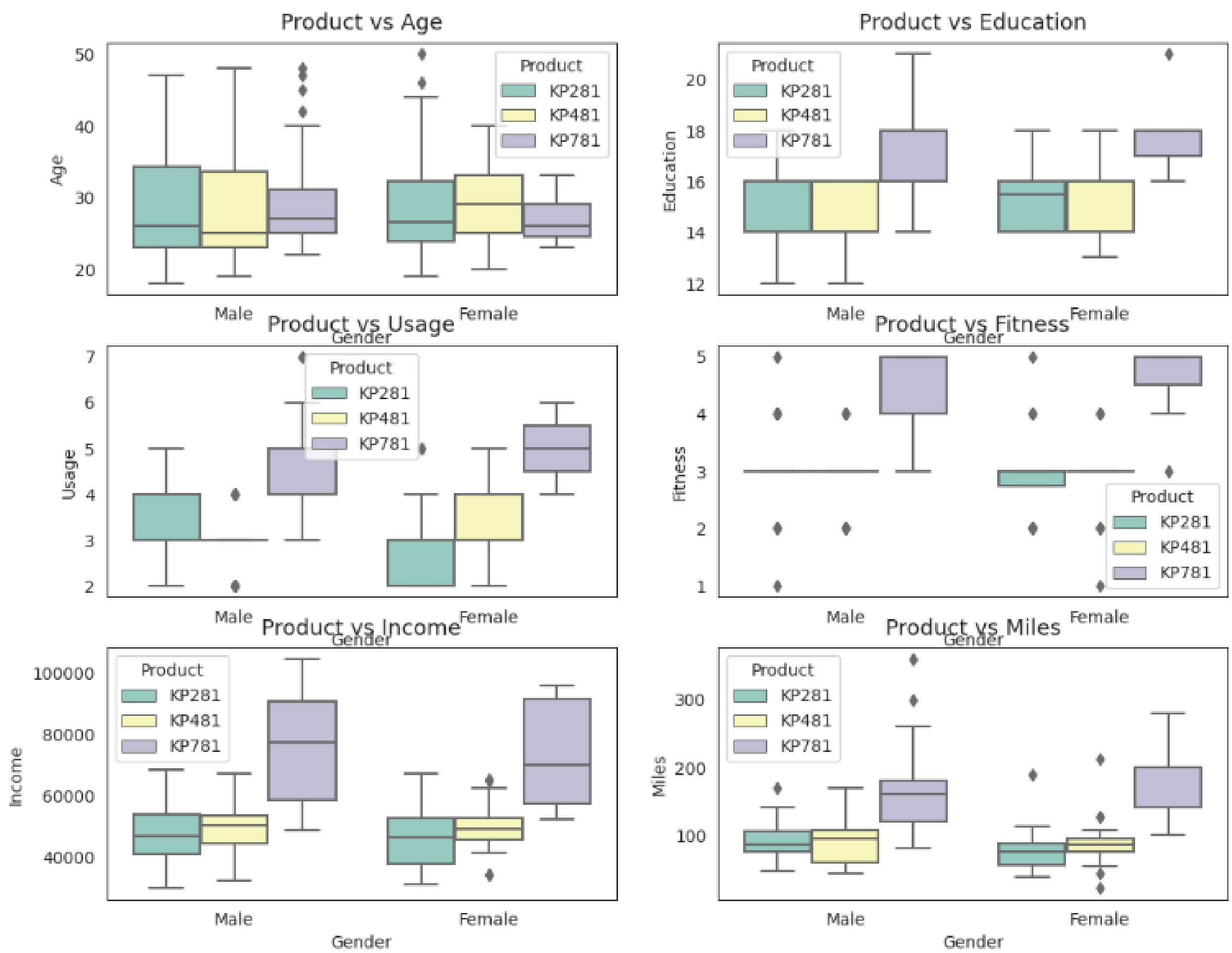
Product vs Miles

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

Multivariate Analysis

```
cols = ['Age', 'Education', 'Usage', 'Fitness', 'Income', 'Miles']
sns.set_style("white")
fig, axis = 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=data, x='Gender', y=cols[count], hue='Product',
    ax=axis[i,j], palette='Set3')
    axis[i,j].set_title(f"Product vs {cols[count]}", pad=8,
    fontsize=13)
```

```
    count += 1
```



Observations

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

---

Computing Marginal & Conditional Probabilities:

Marginal Probability

```
# calculate the marginal probability
data['Product'].value_counts(normalize = True)
```

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

Conditional Probabilities

```
# calculate the conditional probability
def p_prod_given_gender(gender, print_marginal=False):
  if gender is not "Female" and gender is not "Male":
    return "Invalid gender value."
  data_1 = pd.crosstab(index=data['Gender'], columns=[data['Product']])
  p_781 = data_1['KP781'][gender] / data_1.loc[gender].sum()
  p_481 = data_1['KP481'][gender] / data_1.loc[gender].sum()
  p_281 = data_1['KP281'][gender] / data_1.loc[gender].sum()
  if print_marginal:
    print(f"P(Male): {data_1.loc['Male'].sum()/len(data):.2f}")
    print(f"P(Female): {data_1.loc['Female'].sum()/len(data):.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
```

Outliers Detection

Find outliers of every columns

```
# Extract Age col
data['Age'].head(2)

# calculate basic operations
data['Age'].describe()
```

```
# calculate percentile
per_25_age = np.percentile(data['Age'], 25)
per_50_age = np.percentile(data['Age'], 50)
per_75_age = np.percentile(data['Age'], 75)
per_25_age, per_50_age, per_75_age
```

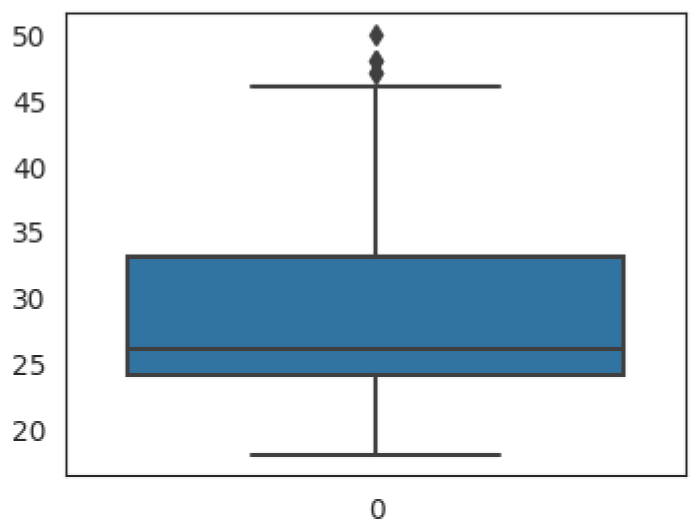
output:
(24.0, 26.0, 33.0)

```
# IQR -  Inter Quartile Range
IQR_age = per_75_age-per_25_age
IQR_age
```

Output: 9.0

Boxplot

```
# Boxplot by univariate variable to find outliers
plt.figure(figsize=(6,4))
sns.boxplot(data["Age"])
plt.show()
```




```
# upper value
Upper_age = per_75_age+1.5*IQR_age
Upper_age
```

output: 46.5

```
# number or outliers

Outliers_age = data[data['Age']>upper_age]
len(outliers_age)
```

output: 5

```
# percentage of outliers and round of 2 decimal points
round(5/180*100, 2)
```

output: 2.78

Observation

so number of outliers of age col. is 5 and percentage of outliers is 2.78

Find outliers of ‘Education’ column

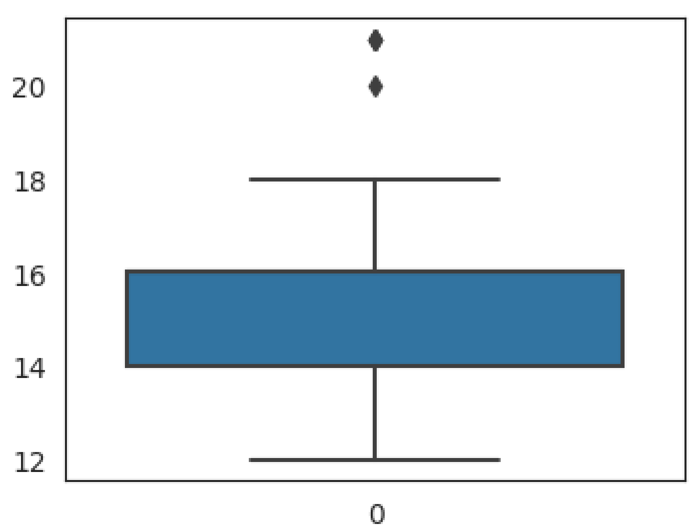
```
# calculate percentile
per_25_education = np.percentile(data['Education'], 25)
per_50_education = np.percentile(data['Education'], 50)
per_75_education = np.percentile(data['Education'], 75)
per_25_education, per_50_education, per_75_education
```

output: `(14.0, 16.0, 16.0)`

```
# IQR -  Inter Quartile Range
IQR_edu = per_75_education-per_25_education
IQR_edu
```

Output: 2.0

```
# boxplot for Education column
plt.figure(figsize=(6,4))
sns.boxplot(data["Education"])
plt.show()
```



```
# upper whisker
upper_edu = per_75_education+1.5*IQR_edu
upper_edu
```
output: 19.0

```
# number or outliers
outliers_edu = data[data['Education']>upper_edu]
len(outliers_edu)
```
output: 4

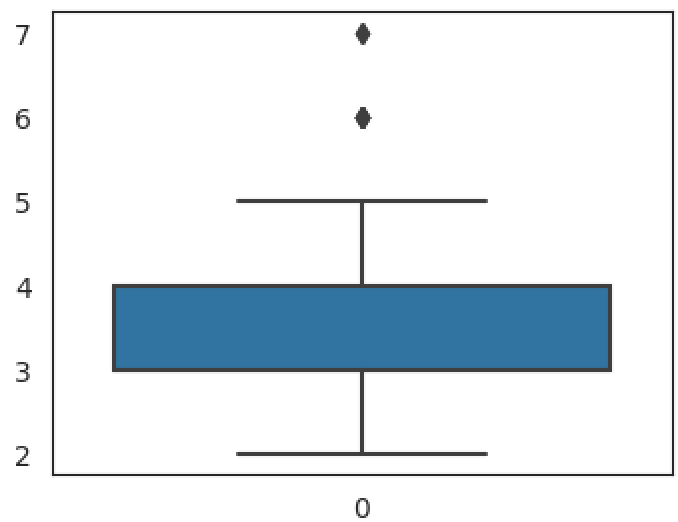
```
# percentage of outliers and round of 2 decimal points
round(4/180*100, 2)
```
output: 2.22

Observation

so number of outliers of education col. is 4 and percentage of outliers is 2.22

outliers of 'Usage' column

```
# boxplot for "usage col"
plt.figure(figsize=(6,4))
sns.boxplot(data["Usage"])
plt.show()
```




```
# calculate percentile
p_25_usage = np.percentile(data['Usage'], 25)
p_50_usage = np.percentile(data['Usage'], 50)
p_75_usage = np.percentile(data['Usage'], 75)
p_25_usage, p_50_usage, p_75_usage
```
output: (3.0, 3.0, 4.0)

```
# IQR -  Inter Quartile Range
iqr_usage = p_75_usage -  p_25_usage
iqr_usage
```
output: 1.0

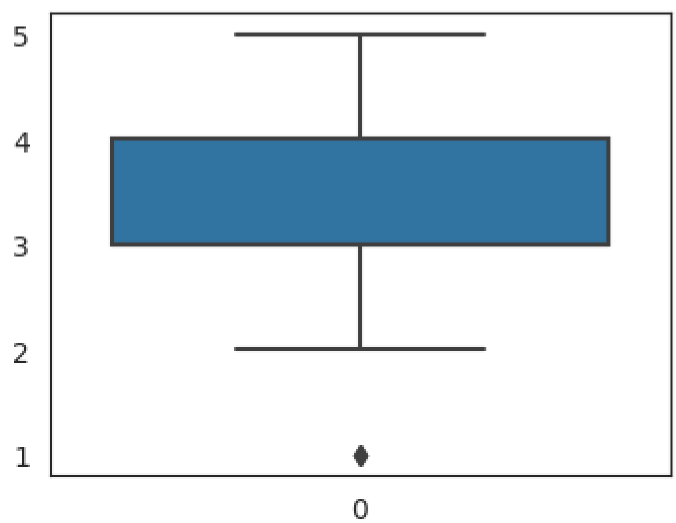
```
#  Determining the upper whisker
upper_usage = p_75_usage + 1.5*iqr_usage
upper_usage
```
output: 5.5

```
# number or outliers
outliers_usage = data[data['Usage']>upper_usage]
len(outliers_usage)
```
ouput: 9

```
# percentage of outliers and round of 2 decimal points
round(9/180*100, 2)
```
output: 5.0

find outliers of fitness column

```
# boxplot for fitness col
plt.figure(figsize=(4,3))
sns.boxplot(data["Fitness"])
plt.show()
```



```
# calculate percentile
p_25_fitness = np.percentile(data['Fitness'], 25)
p_50_fitness = np.percentile(data['Fitness'], 50)
p_75_fitness = np.percentile(data['Fitness'], 75)
p_25_fitness, p_50_fitness, p_75_fitness
```

output: (3.0, 3.0, 4.0)

```
# IQR -  Inter Quartile Range
iqr_fitness = p_75_fitness -  p_25_fitness
iqr_fitness
```

output: 1.0

```
#  Determining the upper whisker
upper_fitness = p_75_fitness + 1.5*iqr_fitness
upper_fitness
```

output: 5.5

```
# calculate lower whisker
lower =  p_25_fitness- 1.5 * iqr_fitness
lower
```

output: 1.5

```
# number or outliers for upper whisker
outliers_fitness = data[data['Fitness']>upper]
len(outliers_fitness)
```

output: 0

```
# number or outliers for lower whisker
outliers_fitness = data[data['Fitness']<lower]
len(outliers_fitness)
```

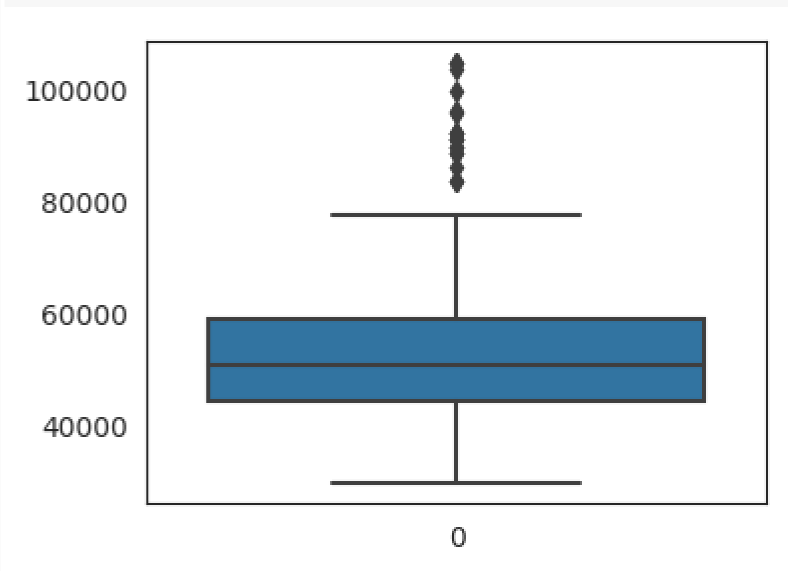
output : 2

```
# percentage of outliers and round of 2 decimal points
round(2/180*100, 2)
```

output: 1.11

outliers for Income column

```
# boxplot for Income column
plt.figure(figsize=(4,3))
sns.boxplot(data["Income"])
plt.show()
```




```
# calculate percentile
p_25_income = np.percentile(data['Income'], 25)
p_50_income = np.percentile(data['Income'], 50)
p_75_income = np.percentile(data['Income'], 75)
p_25_income, p_50_income, p_75_income
```

output: (44058.75, 50596.5, 58668.0)

```
# IQR -  Inter Quartile Range
iqr_income = p_75_income -  p_25_income
iqr_income
```

output: 14609.25

```
#  Determining the upper whisker
upper_income = p_75_income + 1.5*iqr_income
upper_income
```

output: 80581.875

```
# number or outliers for upper whisker
outliers_income = data[data['Income']>upper_income]
len(outliers_income)
```

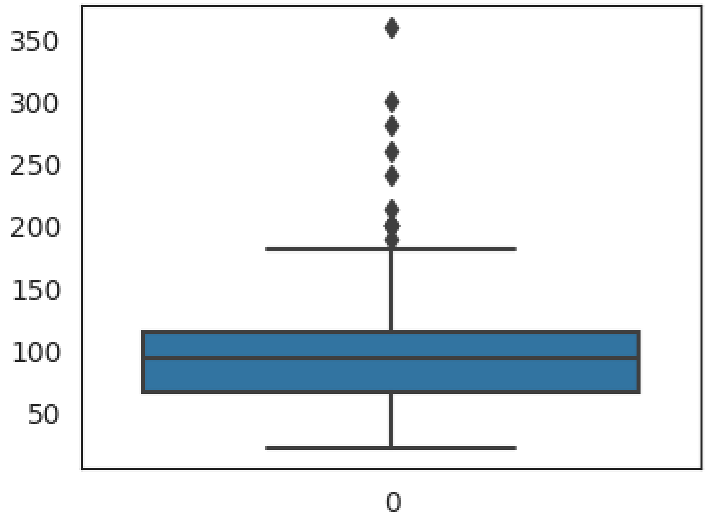
output: 19

```
# percentage of outliers and round of 2 decimal points
round(19/180*100, 2)
```

output: 10.56

outliers for Miles column

```
# boxplot for miles col
plt.figure(figsize=(4,3))
sns.boxplot(data["Miles"])
plt.show()
```




```
# calculate percentile
p_25_miles = np.percentile(data['Miles'], 25)
p_50_miles = np.percentile(data['Miles'], 50)
p_75_miles = np.percentile(data['Miles'], 75)
p_25_miles, p_50_miles, p_75_miles
```

output: **(66.0, 94.0, 114.75)**

```
# IQR -  Inter Quartile Range
iqr_miles = p_75_miles -  p_25_miles
iqr_miles
```

output: 48.75

```
#  Determining the upper whisker
upper_miles = p_75_miles + 1.5*iqr_miles
upper_miles
```

output: 187.875

```
# number or outliers for upper whisker
outliers_miles = data[data['Miles']>upper_miles]
len(outliers_miles)
```

output: 13

```
# percentage of outliers and round of 2 decimal points
round(13/180*100, 2)
```

output: 7.22

Recommendations

Actionable items for business. No technical jargon. No complications. Simple action items that everyone can understand.

1- Customer Health Check-In

2- Class Popularity Analysis

3- Equipment Maintenance Schedule

4- Staff Fitness Challenge

5- Promote Fitness Events

6- Healthy Snack Options

7- Fitness Goal Achievement Celebrations

8- Gym Cleanliness Campaign