```
#Import necessary libraries
```

ChatGPT

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

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
data = pd.read_csv("/content/drive/MyDrive/Colab_Notebooks/aerofit_treadmill.csv")
data
```

	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

1) Defining Problem Statement and Analysing basic metrics.

Problem Statement:

To identify the charecteristics of the target audience for each type of trademill available and to provide a better recommendation for the new customers.

1) Observations on shape of data, data types of all the attributes, conversion of categorical attributes to 'category' (If required), statistical summary

```
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

```
data.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

```
data.shape, data.size
```

```
((180, 9), 1620)
```

```
data.dtypes
```

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

```
# Conversion of categorical attributes to categories
data["Product"] = data["Product"].astype("category")
data["Gender"] = data["Gender"].astype("category")
data["MaritalStatus"] = data["MaritalStatus"].astype("category")
```

```
data.dtypes
```

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

```
data.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

2) Non-Graphical Analysis: Value counts and unique attributes.

```
for column in data.columns:
  value_count = data[column].value_counts()
  print(f'values in {column} :  {value_count}')
```

```
5    31
2    26
4    24
1     2
Name: Fitness, dtype: int64
values in Income :  45480    14
52302     9
46617     8
54576     8
53439     8
         ..
65220     1
55713     1
68220     1
30699     1
95508     1
Name: Income, Length: 62, dtype: int64
values in 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: Miles, dtype: int64
```

```
data.nunique()
```

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

```
data["Product"].unique()
```

```
['KP281', 'KP481', 'KP781']
Categories (3, object): ['KP281', 'KP481', 'KP781']
```

```
# Finding the no of unique values in each feature and also its values
for column in data.columns:
    unique_values = data[column].unique()
    num_unique = data[column].nunique()

    print(f'Unique values in "{column}" ({num_unique} unique values):', end = "")
    print(unique_values)
```

```
Unique values in "Product" (3 unique values):['KP281', 'KP481', 'KP781']
Categories (3, object): ['KP281', 'KP481', 'KP781']
Unique values in "Age" (32 unique values):[18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41
 43 44 46 47 50 45 48 42]
```

```
Unique values in "Gender" (2 unique values):['Male', 'Female']
Categories (2, object): ['Female', 'Male']
Unique values in "Education" (8 unique values):[14 15 12 13 16 18 20 21]
Unique values in "MaritalStatus" (2 unique values):['Single', 'Partnered']
Categories (2, object): ['Partnered', 'Single']
Unique values in "Usage" (6 unique values):[3 2 4 5 6 7]
Unique values in "Fitness" (5 unique values):[4 3 2 1 5]
Unique values in "Income" (62 unique values):[ 29562  31836  30699  32973  35247  37521  36384  38658  40932  34110
  39795  42069  44343  45480  46617  48891  53439  43206  52302  51165
  50028  54576  68220  55713  60261  67083  56850  59124  61398  57987
  64809  47754  65220  62535  48658  54781  48556  58516  53536  61006
  57271  52291  49801  62251  64741  70966  75946  74701  69721  83416
  88396  90886  92131  77191  52290  85906 103336  99601  89641  95866
 104581  95508]
Unique values in "Miles" (37 unique values):[112  75  66  85  47 141 103  94 113  38 188  56 132 169  64  53 106  95
 212  42 127  74 170  21 120 200 140 100  80 160 180 240 150 300 280 260
 360]
```

```
# Create a cross-tabulation (contingency table) to calculate marginal probabilities
cross_tab = pd.crosstab(index=data['Product'], columns='Count', normalize=True)

# Rename the 'Count' column to 'Marginal Probability'
cross_tab.rename(columns={'Count': 'Marginal Probability'}, inplace=True)

# Print the cross-tabulation
print(cross_tab)
```

```
col_0    Marginal Probability
Product
KP281                0.444444
KP481                0.333333
KP781                0.222222
```

```
# Create a crosstab to calculate probabilities (Conditional Probability)
crosstab = pd.crosstab(data['Gender'], data['Product'], normalize='index') * 100

# Print the crosstab
print(crosstab)
```

```
Product      KP281      KP481      KP781
Gender
Female   52.631579  38.157895   9.210526
Male     38.461538  29.807692  31.730769
```

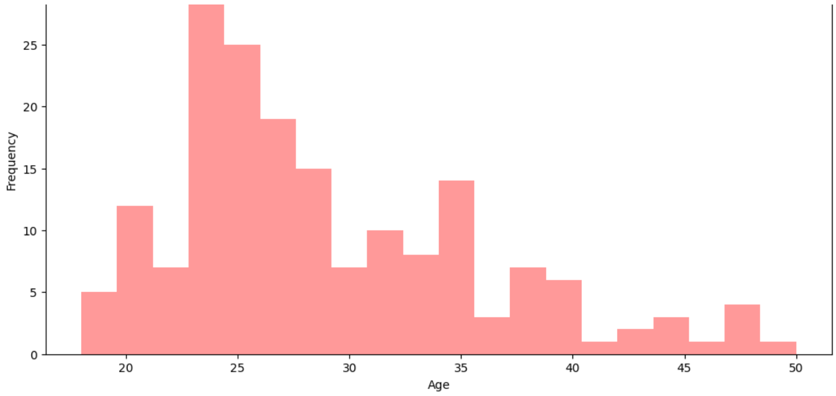
3) Visual Analysis - Univariate & Bivariate.

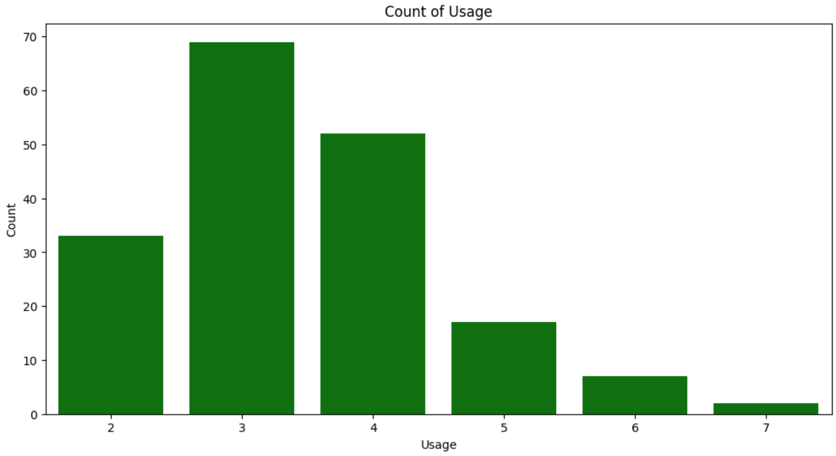
1) For continuous variable(s): Distplot, countplot, histogram for univariate analysis (10 Points)

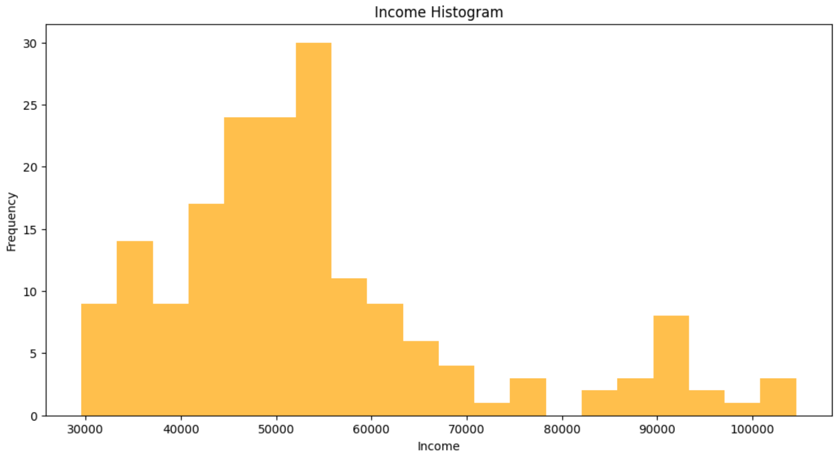
```
data.head()
```

	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

```
# Distplot (Distribution Plot)
plt.figure(figsize=(12, 6))
sns.distplot(data["Age"], kde=False, bins=20, color = "red")
plt.title("Distribution of Age")
plt.xlabel("Age")
plt.ylabel("Frequency")
plt.show()

# Countplot for Usage
plt.figure(figsize=(12, 6))
sns.countplot(data=data, x="Usage", color="green")
plt.title("Count of Usage")
plt.xlabel("Usage")
plt.ylabel("Count")
plt.show()


# Histogram
plt.figure(figsize=(12, 6))
plt.hist(data["Income"], bins=20, color="orange", alpha=0.7)
plt.title("Income Histogram")
plt.xlabel("Income")
plt.ylabel("Frequency")
plt.show()
```

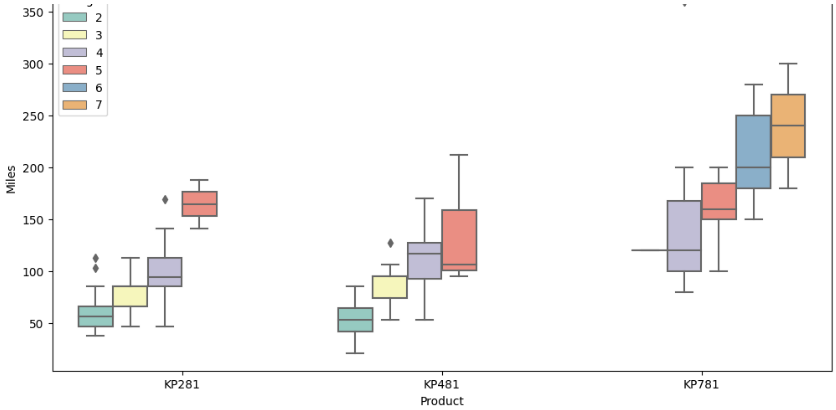

Frequency
Age

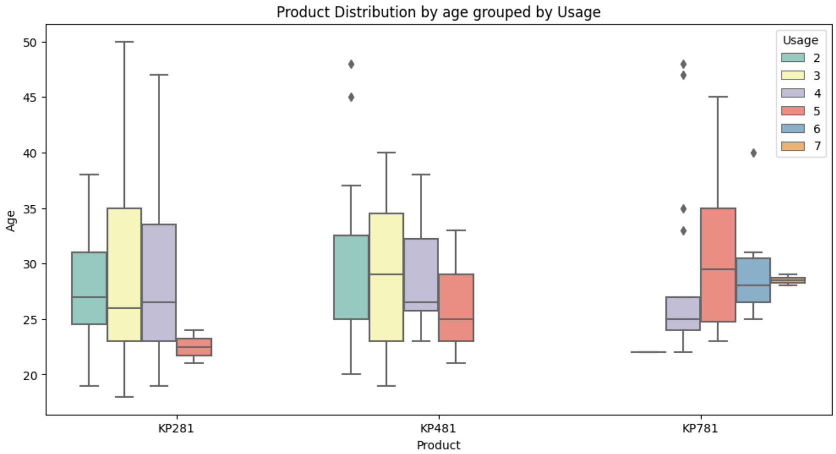

Count of Usage
Count
Usage

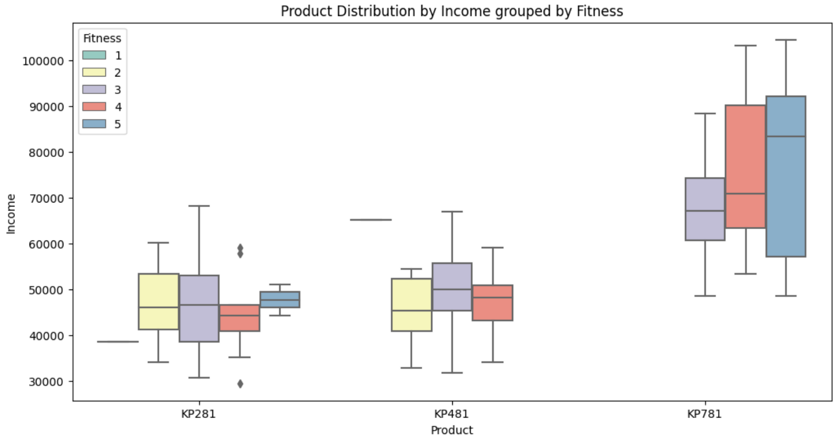

Income Histogram
Frequency
Income

2) For categorical variables: Boxplot

```
# Box plot for categorical variables
plt.figure(figsize=(12, 6))
sns.boxplot(data=data, x="Gender", y="Income", hue = "MaritalStatus", palette="Set3")
plt.title("Income Distribution by Gender grouped by MaritalStatus")
plt.xlabel("Gender")
plt.ylabel("Income")
plt.show()

plt.figure(figsize=(12, 6))
sns.boxplot(data=data, x="Product", y="Miles", hue = "Usage", palette="Set3")
plt.title("Product Distribution by miles grouped by Usage")
plt.xlabel("Product")
plt.ylabel("Miles")
plt.show()

plt.figure(figsize=(12, 6))
sns.boxplot(data=data, x="Product", y="Age", hue = "Usage", palette="Set3")
plt.title("Product Distribution by age grouped by Usage")
plt.xlabel("Product")
plt.ylabel("Age")
plt.show()

plt.figure(figsize=(12, 6))
sns.boxplot(data=data, x="Product", y="Income", hue = "Fitness", palette="Set3")
plt.title("Product Distribution by Income grouped by Fitness")
plt.xlabel("Product")
plt.ylabel("Income")
plt.show()
```

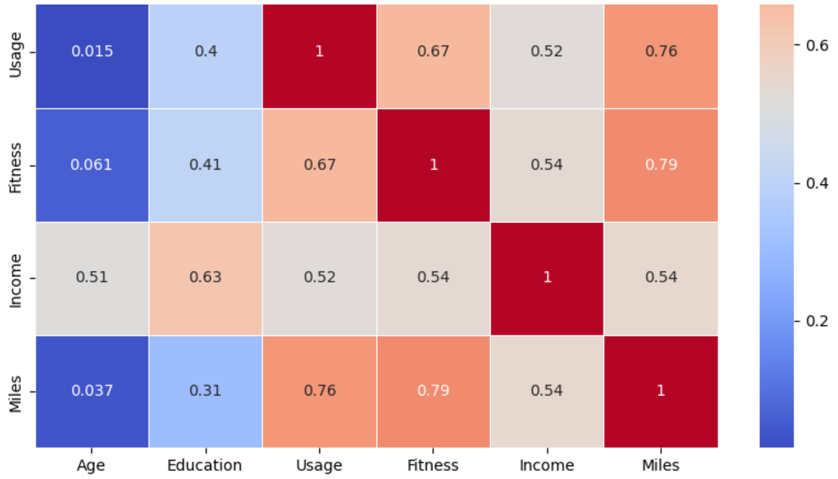

2
3
4
5
6
7
Miles
350
300
250
200
150
100
50
KP281
KP481
KP781
Product

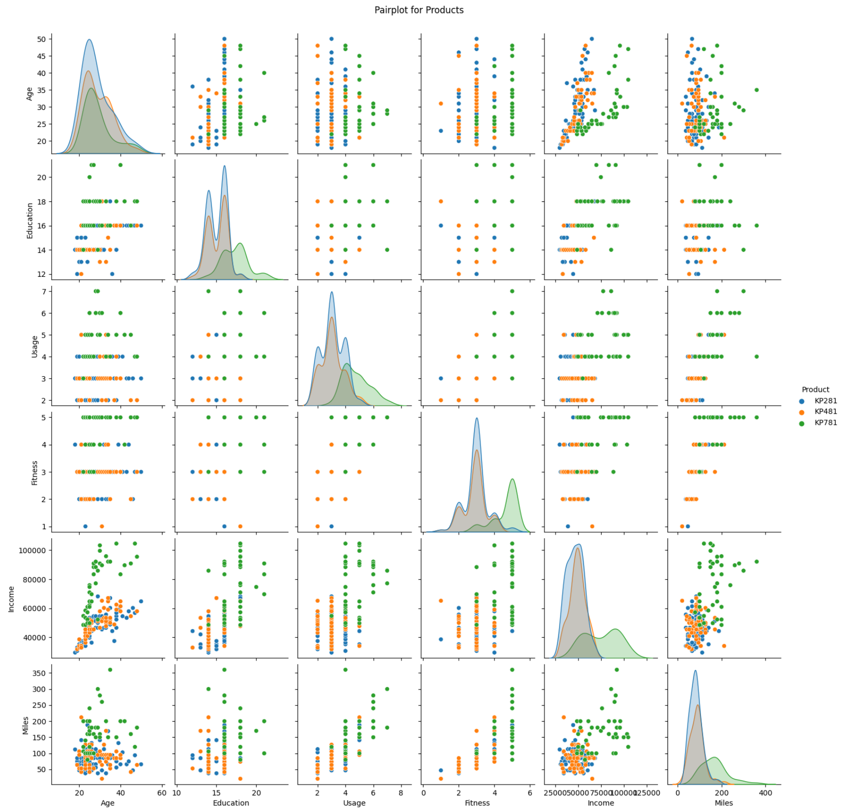

Product Distribution by age grouped by Usage
Usage
2
3
4
5
6
7
Age
50
45
40
35
30
25
20
KP281
KP481
KP781
Product


Product Distribution by Income grouped by Fitness
Fitness
1
2
3
4
5
Income
100000
90000
80000
70000
60000
50000
40000
30000
KP281
KP481
KP781
Product

3) For correlation: Heatmaps, Pairplots

```
# Correlation Heatmap
correlation_matrix = data.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", linewidths=.5)
plt.title("Correlation Heatmap")
plt.show()

# Pairplot
sns.pairplot(data, hue="Product", diag_kind="kde")
plt.suptitle("Pairplot for Products", y=1.02)
plt.show()
```


Usage
Fitness
Income
Miles
0.015 0.4 1 0.67 0.52 0.76
0.061 0.41 0.67 1 0.54 0.79
0.51 0.63 0.52 0.54 1 0.54
0.037 0.31 0.76 0.79 0.54 1
Age
Education
Usage
Fitness
Income
Miles
0.6
0.4
0.2


Pairplot for Products
Age
Education
Usage
Fitness
Income
Miles
Product
KP281
KP481
KP781