```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
import re
import plotly.express as px
import plotly.graph_objs as go
import plotly.figure_factory as ff
from textblob import TextBlob
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler


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

To explore and Visualise the data

```
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

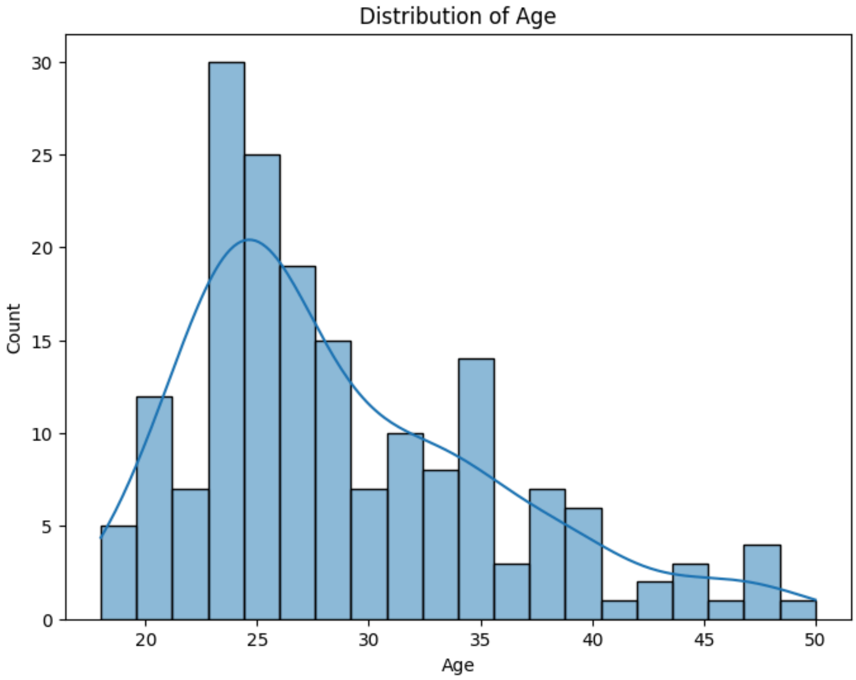
```
df.head()
```

	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(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
None
```

```
plt.figure(figsize=(8, 6))
sns.histplot(data=df, x='Age', bins=20, kde=True)
plt.title('Distribution of Age')
plt.xlabel('Age')
plt.ylabel('Count')
plt.show()
```




```
print(df.describe())
```

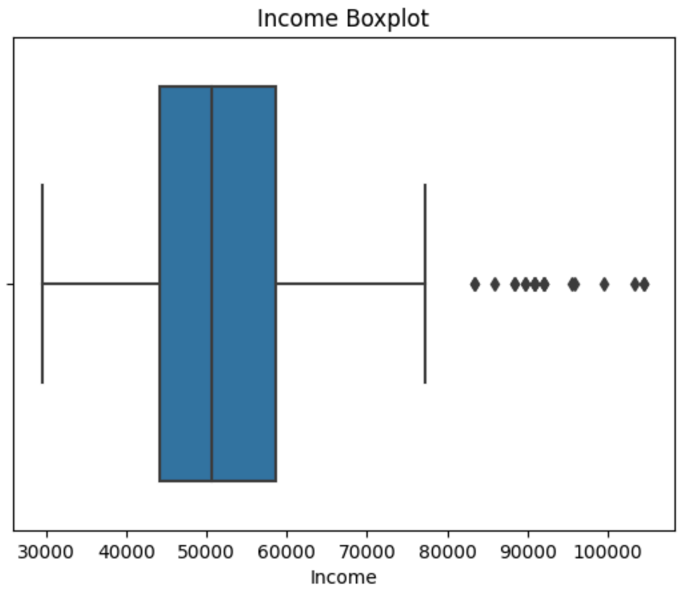
```
              Age   Education       Usage     Fitness         Income  \
count  180.000000  180.000000  180.000000  180.000000     180.000000
mean    28.788889   15.572222    3.455556    3.311111   53719.577778
std      6.943498    1.617055    1.084797    0.958869   16506.684226
min     18.000000   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
```

```
# Check for missing values
missing_values = df.isnull().sum()
print("Missing Values:")
print(missing_values)
```

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

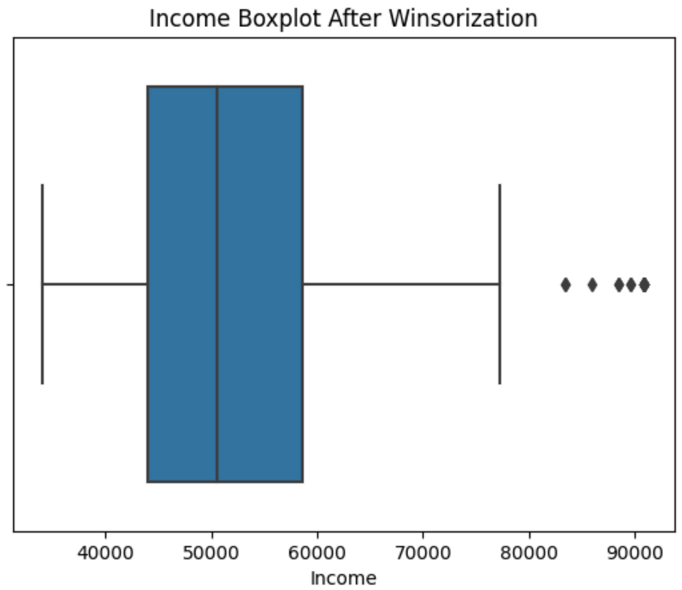
```
# Handling missing values
mean_age = df['Age'].mean()
df['Age'].fillna(mean_age, inplace=True)
```

```
sns.boxplot(x=df['Income'])
plt.title("Income Boxplot")
plt.show()
```




```
from scipy.stats.mstats import winsorize
df['Income'] = winsorize(df['Income'], limits=[0.05, 0.05])
```

```
sns.boxplot(x=df['Income'])
plt.title("Income Boxplot After Winsorization")
plt.show()
```



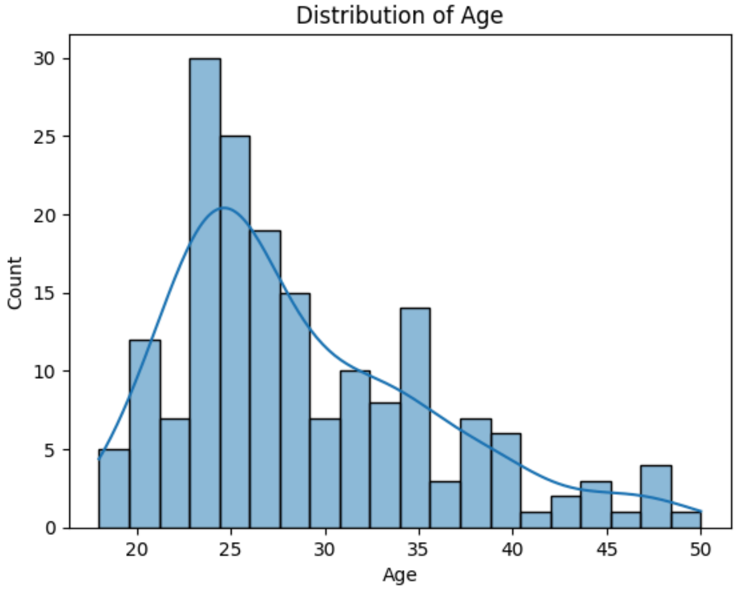

```
# Summary statistics for numerical attributes

numerical_attributes = ['Age', 'Usage', 'Income', 'Fitness', 'Miles']
summary_stats = df[numerical_attributes].describe()
print("Summary Statistics for Numerical Attributes:")
print(summary_stats)
```

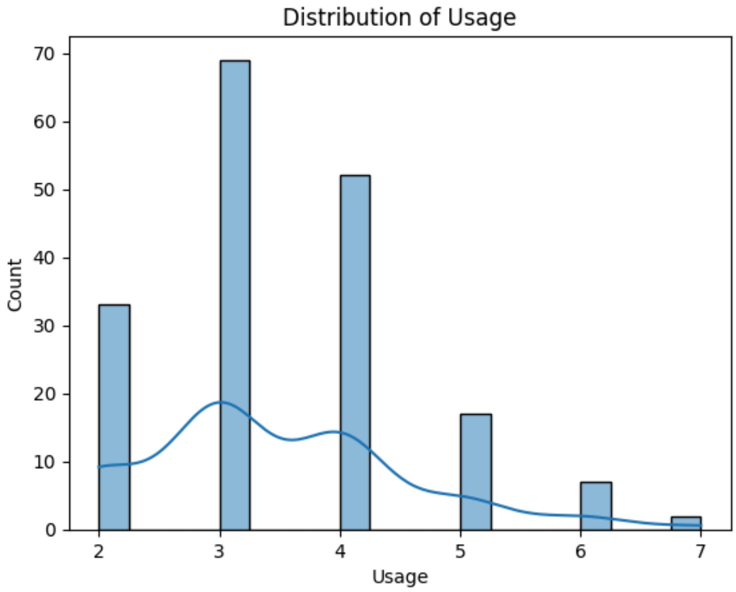
```
Summary Statistics for Numerical Attributes:
              Age       Usage        Income     Fitness       Miles
count  180.000000  180.000000    180.000000  180.000000  180.000000
mean    28.788889    3.455556  53476.800000    3.311111  103.194444
std      6.943498    1.084797  15452.495358    0.958869   51.863605
min     18.000000    2.000000  34110.000000    1.000000   21.000000
25%     24.000000    3.000000  44058.750000    3.000000   66.000000
50%     26.000000    3.000000  50596.500000    3.000000   94.000000
75%     33.000000    4.000000  58668.000000    4.000000  114.750000
max     50.000000    7.000000  90886.000000    5.000000  360.000000
```

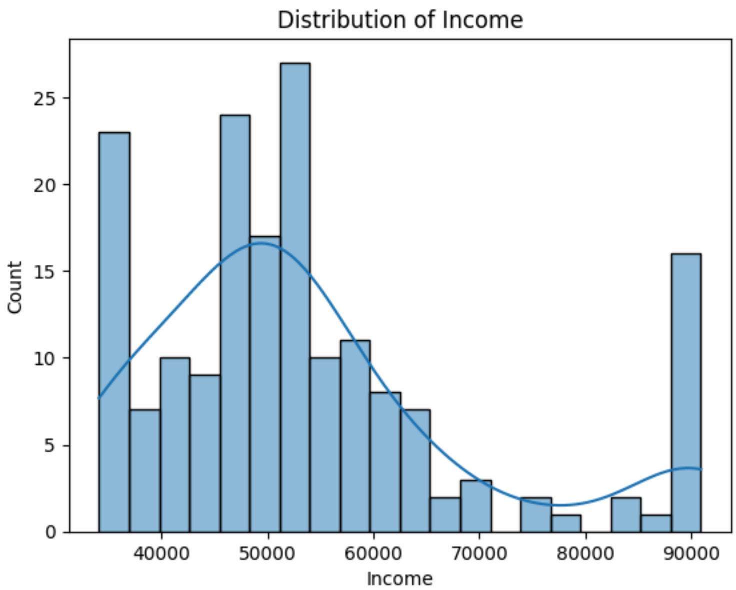
```
# Count of unique values for categorical attributes

categorical_attributes = ['Gender', 'Education', 'MaritalStatus', 'Product']
unique_counts = df[categorical_attributes].nunique()
print("\nCount of Unique Values for Categorical Attributes:")
print(unique_counts)
```

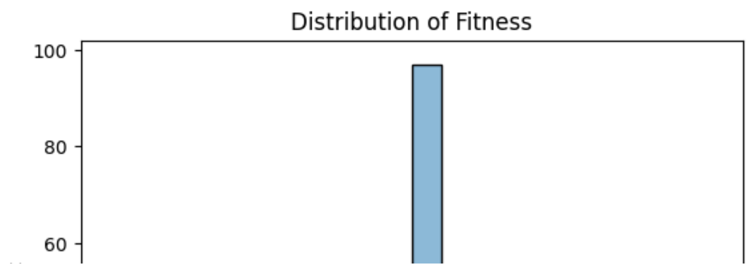
```
Count of Unique Values for Categorical Attributes:
Gender           2
Education        8
MaritalStatus    2
Product          3
dtype: int64
```

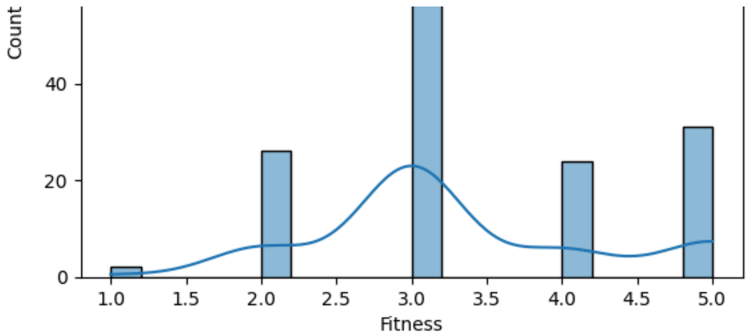
```
# Visualizations to explore attributes

for attribute in numerical_attributes:
    sns.histplot(data=df, x=attribute, bins=20, kde=True)
    plt.title(f'Distribution of {attribute}')
    plt.show()

for attribute in categorical_attributes:
    sns.countplot(data=df, x=attribute)
    plt.title(f'Count of Customers by {attribute}')
    plt.xticks(rotation=45)
    plt.show()
```

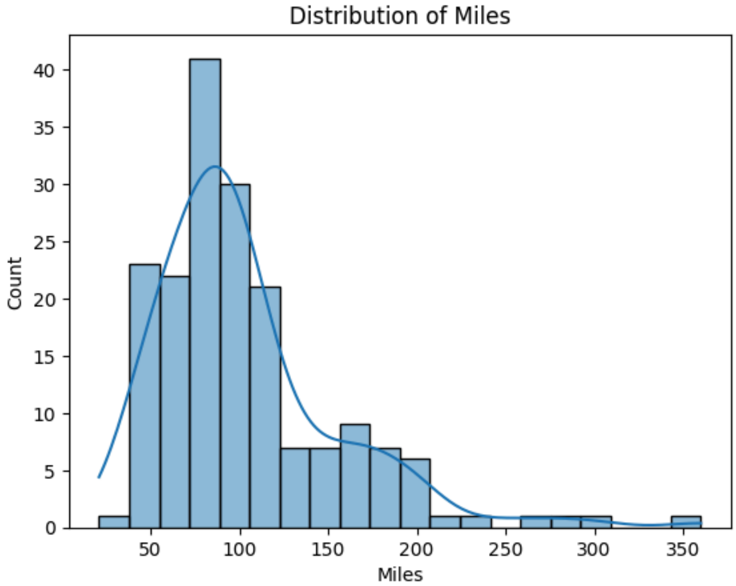

Distribution of Age
Count
Age
0
5
10
15
20
25
30
20
25
30
35
40
45
50

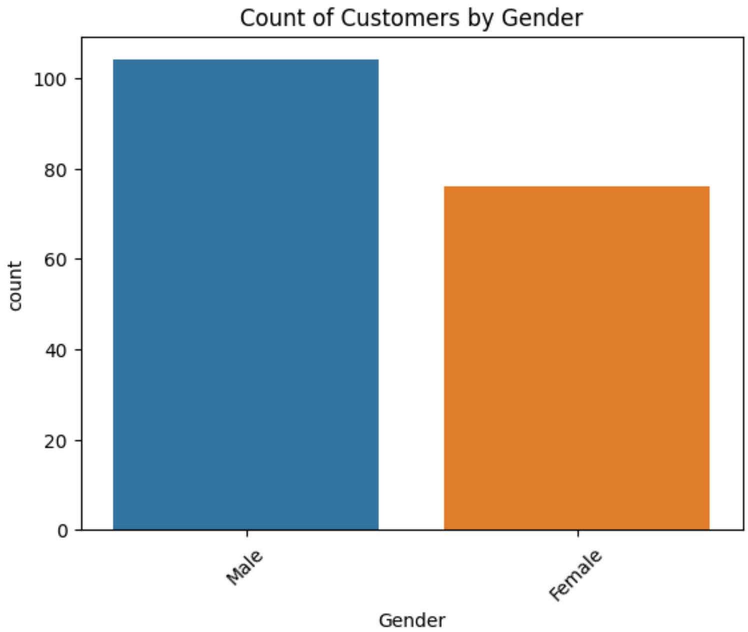

Distribution of Usage
Count
Usage
0
10
20
30
40
50
60
70
2
3
4
5
6
7

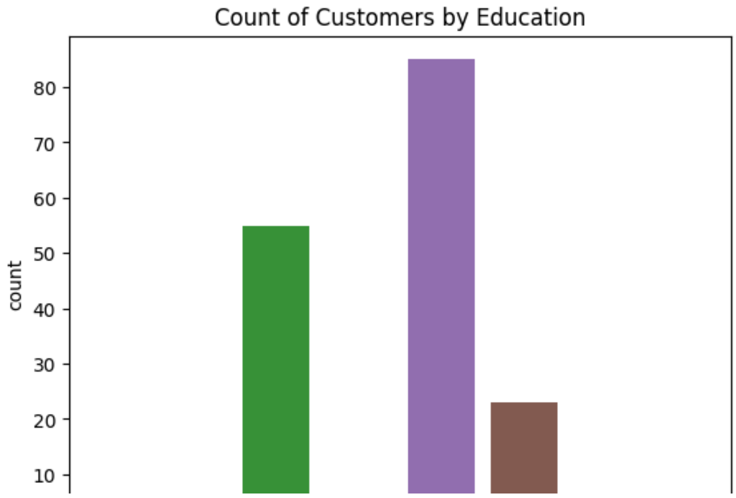

Distribution of Income
Count
Income
0
5
10
15
20
25
40000
50000
60000
70000
80000
90000

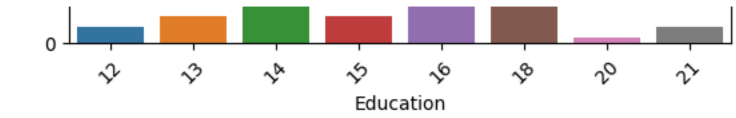

Distribution of Fitness
60
80
100

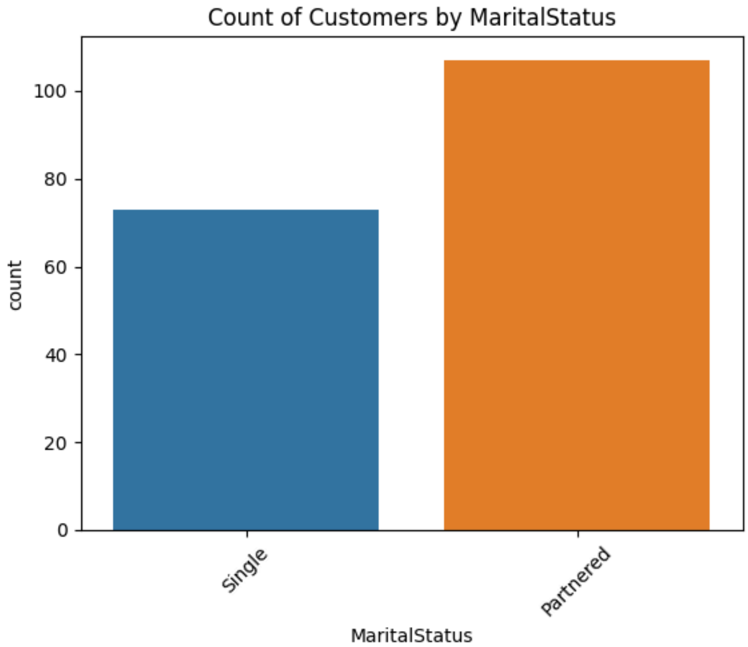

Count
40
20
0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Fitness

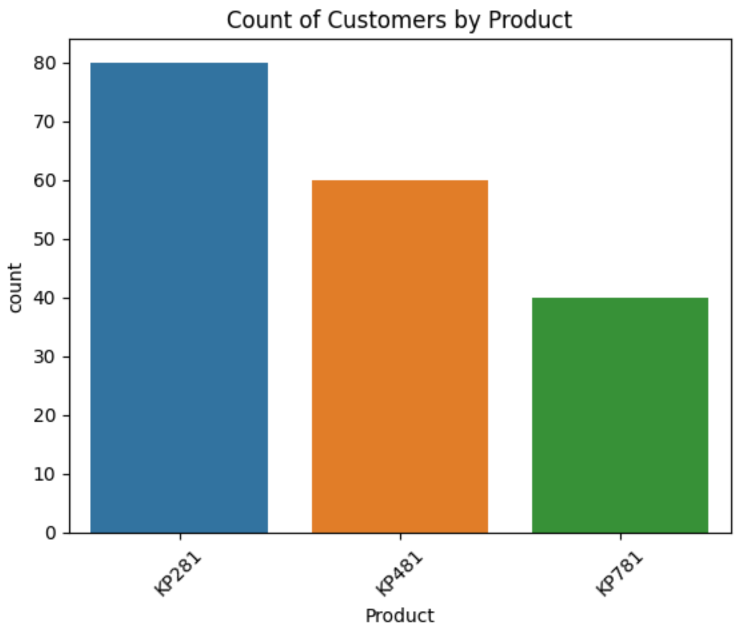

Distribution of Miles
Count
40
35
30
25
20
15
10
5
0
50
100
150
200
250
300
350
Miles


Count of Customers by Gender
count
100
80
60
40
20
0
Male
Female
Gender


Count of Customers by Education
count
80
70
60
50
40
30
20
10





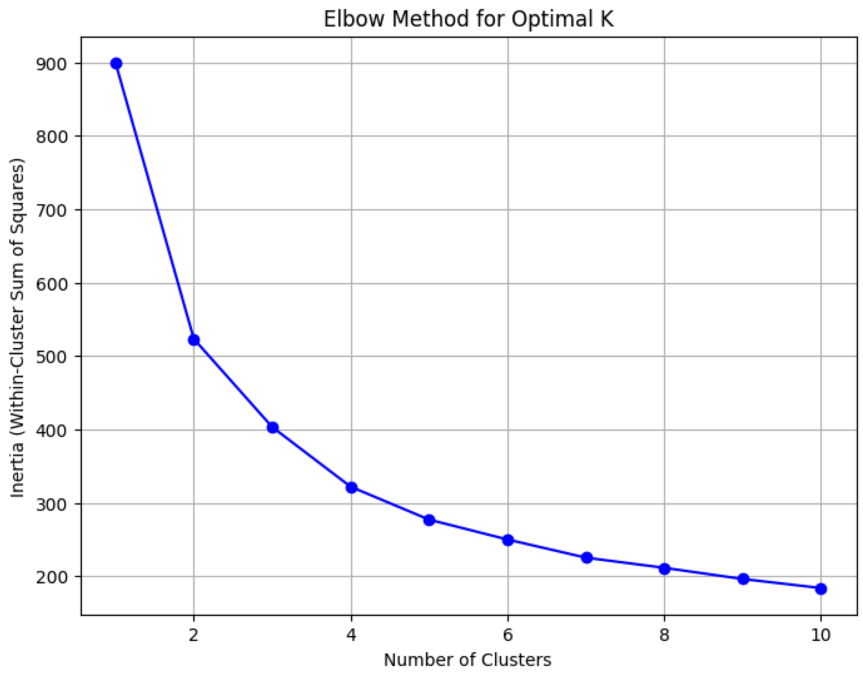



```
attributes_for_clustering = ['Age', 'Usage', 'Income', 'Fitness', 'Miles']
```

```
scaler = StandardScaler()
data_for_clustering = scaler.fit_transform(df[attributes_for_clustering])
```

```
inertia_values = []
k_values = range(1, 11)
for k in k_values:
    kmeans = KMeans(n_clusters=k, n_init=10, random_state=42)
    kmeans.fit(data_for_clustering)
    inertia_values.append(kmeans.inertia_)
```

```
plt.figure(figsize=(8, 6))
plt.plot(k_values, inertia_values, marker='o', linestyle='-', color='b')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia (Within-Cluster Sum of Squares)')
plt.title('Elbow Method for Optimal K')
plt.grid(True)
plt.show()
```




```
optimal_k = 3
kmeans = KMeans(n_clusters=optimal_k, n_init=10, random_state=42)
df['Cluster'] = kmeans.fit_predict(data_for_clustering)
for cluster in range(optimal_k):
    cluster_data = df[df['Cluster'] == cluster]
    plt.scatter(cluster_data['Age'], cluster_data['Income'], label=f'Cluster {cluster + 1}')

plt.xlabel('Age')
plt.ylabel('Income')
plt.title('Customer Clusters')
plt.legend()
plt.grid(True)
plt.show()
```

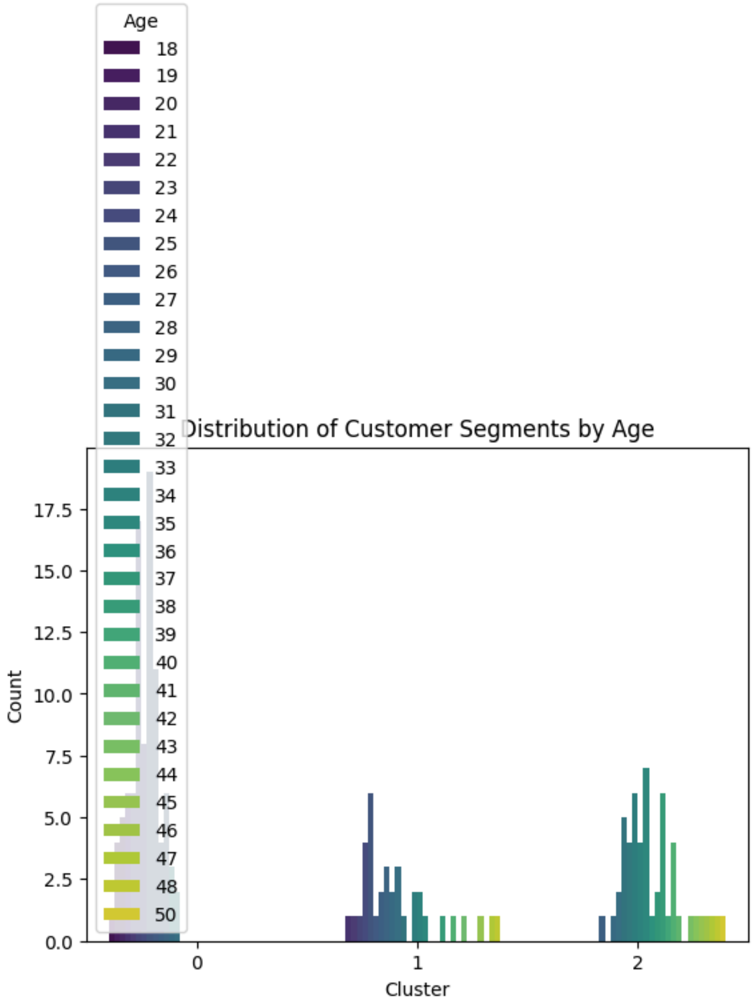

```
cluster_characteristics = df.groupby('Cluster')[attributes_for_clustering].mean()
print("Cluster Characteristics:")
print(cluster_characteristics)
```

```
Cluster Characteristics:
               Age     Usage        Income   Fitness       Miles
Cluster
0        24.119565  3.141304  44137.804348  2.978261   85.423913
1        30.111111  4.916667  73699.111111  4.777778  183.583333
2        36.134615  3.000000  55999.576923  2.884615   78.980769
```

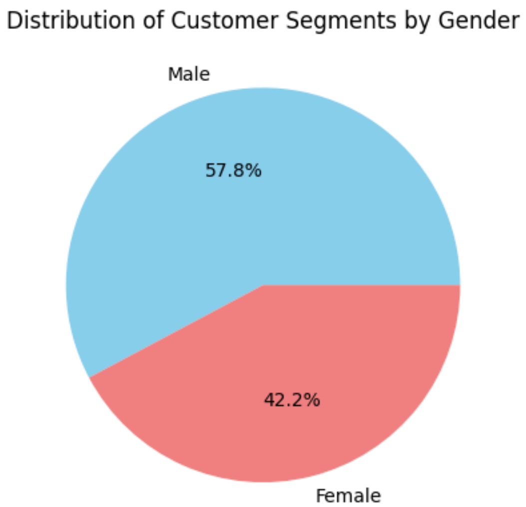
```
clustered_data = df.groupby('Cluster')

for cluster, data in clustered_data:
    print(f"Cluster {cluster}:")
    print("--------------------")
    print("Mean:")
    print(data.mean(numeric_only=True))
    print("\nMedian:")
    print(data.median(numeric_only=True))
    print("\nOther Metrics:")
    print(data.describe())

# to calculate the mean of 'Age' for each cluster
age_means = clustered_data['Age'].mean()
print("Mean Age for Each Cluster:")
print(age_means)
```

```
Cluster 0:
--------------------
Mean:
Age             24.119565
Education       14.902174
Usage            3.141304
Fitness          2.978261
Income       44137.804348
Miles           85.423913
Cluster          0.000000
dtype: float64

Median:
Age             24.0
Education       14.0
Usage            3.0
Fitness          3.0
Income       45480.0
Miles           85.0
Cluster          0.0
dtype: float64

Other Metrics:
             Age  Education      Usage    Fitness        Income       Miles  \
count  92.000000  92.000000  92.000000  92.000000     92.000000   92.000000
mean   24.119565  14.902174   3.141304   2.978261  44137.804348   85.423913
std     2.676074   1.383258   0.806315   0.695010   7423.448941   27.258725
min    18.000000  12.000000   2.000000   1.000000  34110.000000   38.000000
25%    23.000000  14.000000   3.000000   3.000000  38373.750000   65.500000
50%    24.000000  14.000000   3.000000   3.000000  45480.000000   85.000000
75%    26.000000  16.000000   4.000000   3.000000  49175.250000  106.000000
max    30.000000  21.000000   5.000000   5.000000  69721.000000  170.000000

       Cluster
count     92.0
mean       0.0
std        0.0
min        0.0
25%        0.0
50%        0.0
75%        0.0
max        0.0
Cluster 1:
--------------------
Mean:
Age             30.111111
Education       17.027778
Usage            4.916667
Fitness          4.777778
Income       73699.111111
Miles          183.583333
Cluster          1.000000
dtype: float64

Median:
Age             28.0
Education       17.0
Usage            5.0
```

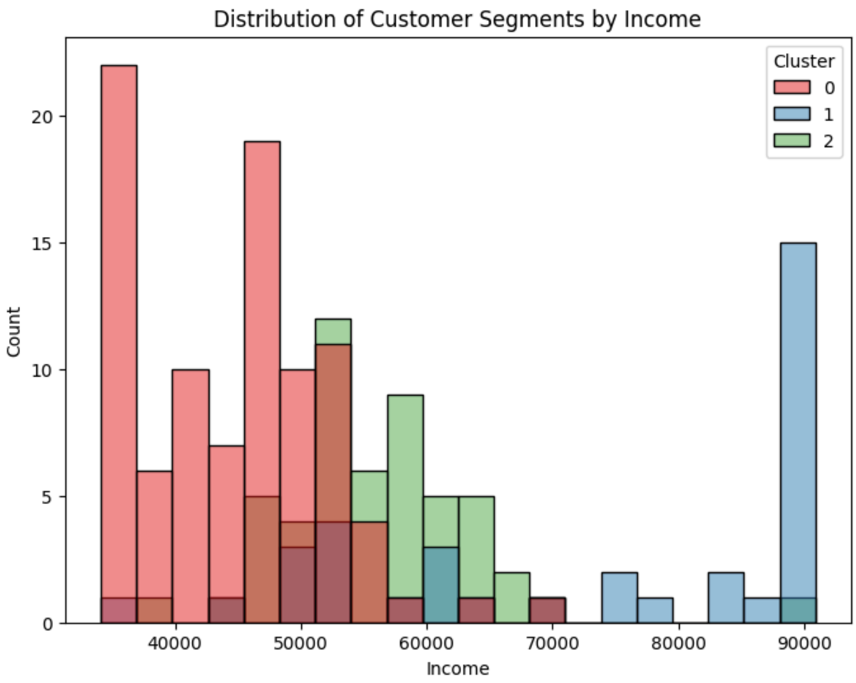
```
#bar chart to visualize the distribution of customer segments by age
sns.countplot(data=df, x='Cluster', hue='Age', palette='viridis')
plt.title("Distribution of Customer Segments by Age")
plt.xlabel("Cluster")
plt.ylabel("Count")
plt.show()
```




```
# Plot a pie chart to visualize the distribution of customer segments by gender
gender_counts = df['Gender'].value_counts()
plt.pie(gender_counts, labels=gender_counts.index, autopct='%1.1f%%', colors=['skyblue', 'lightcoral'])
plt.title("Distribution of Customer Segments by Gender")
plt.show()
```




```
# Plot a histogram to visualize the distribution of customer segments by income
plt.figure(figsize=(8, 6))
sns.histplot(data=df, x='Income', hue='Cluster', bins=20, palette='Set1')
plt.title("Distribution of Customer Segments by Income")
plt.xlabel("Income")
plt.ylabel("Count")
plt.show()
```



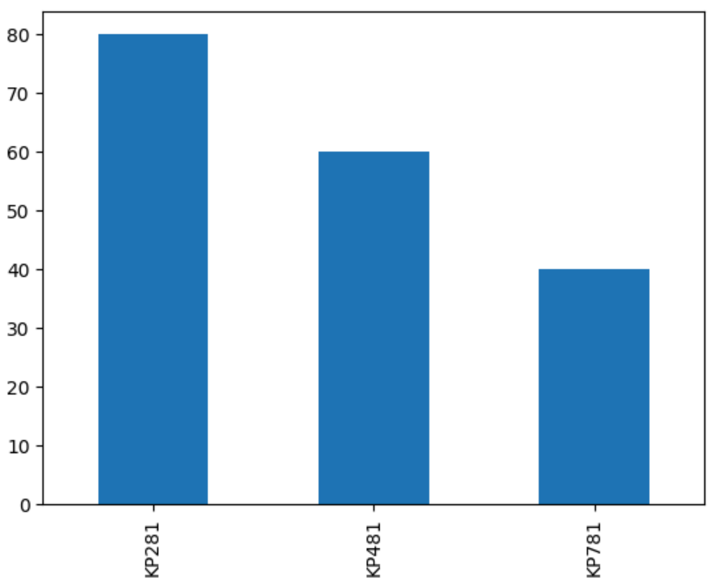
Observations

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.

checking the structure & characteristics of the dataset

```
df["Product"].value_counts().plot(kind="bar")
```

<Axes: >




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

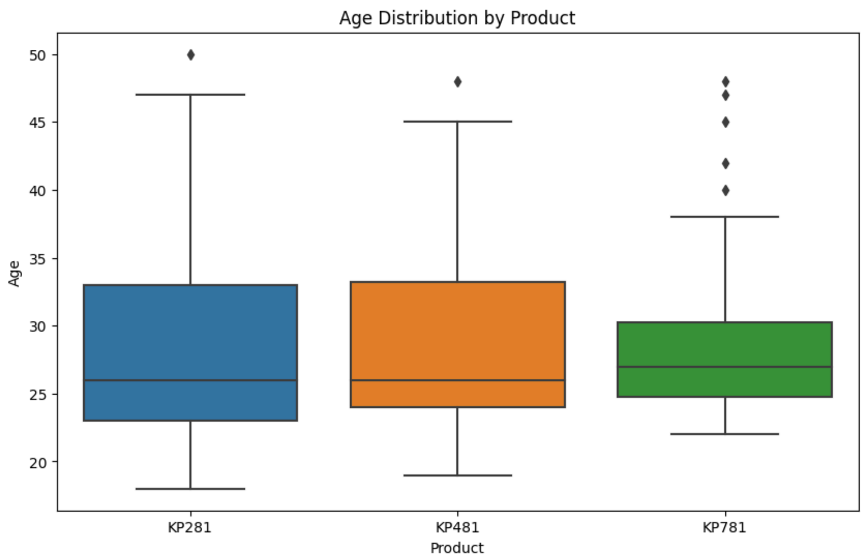
Product	KP281	KP481	KP781
Gender			
Female	40	29	7
Male	40	31	33

checking the difference between mean and median

```
kp281_df = df[df['Product'] == 'KP281']
kp481_df = df[df['Product'] == 'KP481']
kp781_df = df[df['Product'] == 'KP781']
```

```
kp281_profile = kp281_df.describe()
kp481_profile = kp481_df.describe()
kp781_profile = kp781_df.describe()
```

```
plt.figure(figsize=(10, 6))
sns.boxplot(data=df, x='Product', y='Age')
plt.title('Age Distribution by Product')
plt.xlabel('Product')
plt.ylabel('Age')
plt.show()
```




```
df['Product'].value_counts()
df['Gender'].value_counts()
df['MaritalStatus'].value_counts()
```

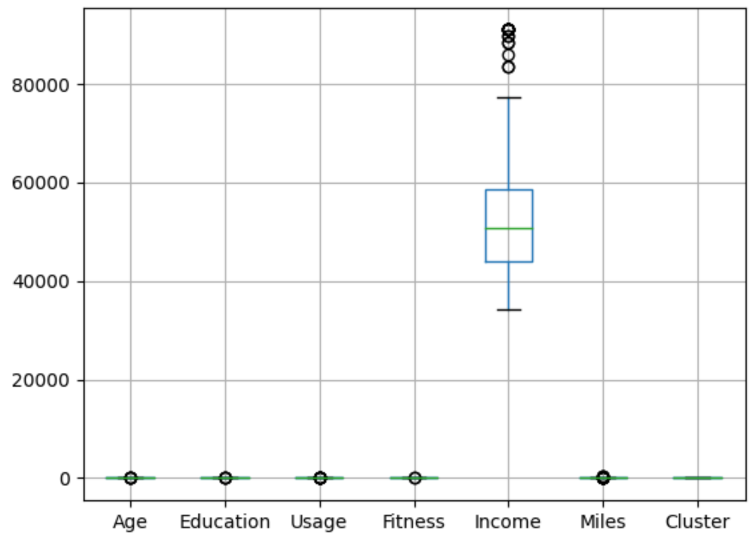
```
Partnered    107
Single        73
Name: MaritalStatus, dtype: int64
```

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

MaritalStatus	Partnered	Single
Product		
KP281	48	32
KP481	36	24
KP781	23	17

```
df.boxplot()
```

```
<Axes: >
```



Observations:

From the above boxplots it is quite clear that Age, Education and Usage are having very few outliers while Income and Miles are having more outliers.

```
# Calculate correlation matrix
df.corr(numeric_only=True)
```

	Age	Education	Usage	Fitness	Income	Miles	Cluster
Age	1.000000	0.280496	0.015064	0.061105	0.511992	0.036618	0.751567
Education	0.280496	1.000000	0.395155	0.410581	0.628884	0.307284	0.281891
Usage	0.015064	0.395155	1.000000	0.668606	0.527747	0.759130	0.030955
Fitness	0.061105	0.410581	0.668606	1.000000	0.535939	0.785702	0.056629
Income	0.511992	0.628884	0.527747	0.535939	1.000000	0.554514	0.412124
Miles	0.036618	0.307284	0.759130	0.785702	0.554514	1.000000	0.046590
Cluster	0.751567	0.281891	0.030955	0.056629	0.412124	0.046590	1.000000

```
contingency_kp281 = pd.crosstab(kp281_df['Gender'], kp281_df['MaritalStatus'])
```

```
conditional_prob_kp281 = kp281_df.groupby('Gender')['Fitness'].value_counts(normalize=True)
marginal_prob_gender_kp281 = kp281_df['Gender'].value_counts(normalize=True)
marginal_prob_fitness_kp281 = kp281_df['Fitness'].value_counts(normalize=True)
```

```
conditional_prob_kp281
```

```
Gender  Fitness
Female  3          0.650
        2          0.250
        4          0.075
        5          0.025
Male    3          0.700
        4          0.150
        2          0.100
        1          0.025
        5          0.025
Name: Fitness, dtype: float64
```

```
marginal_prob_gender_kp281
```

```
Male      0.5
Female    0.5
Name: Gender, dtype: float64
```

```
marginal_prob_fitness_kp281
```

```
3    0.6750
2    0.1750
4    0.1125
```

```
5    0.0250
1    0.0125
Name: Fitness, dtype: float64
```

```
conditional_prob_kp481 = kp481_df.groupby('Gender')['Fitness'].value_counts(normalize=True)
marginal_prob_gender_kp481 = kp481_df['Gender'].value_counts(normalize=True)
marginal_prob_fitness_kp481 = kp481_df['Fitness'].value_counts(normalize=True)
```

```
conditional_prob_kp481
```

```
Gender  Fitness
Female  3          0.620690
        2          0.206897
        4          0.137931
        1          0.034483
Male    3          0.677419
        2          0.193548
        4          0.129032
Name: Fitness, dtype: float64
```

```
marginal_prob_gender_kp481
```

```
Male      0.516667
Female    0.483333
Name: Gender, dtype: float64
```

```
marginal_prob_fitness_kp481
```

```
3    0.650000
2    0.200000
4    0.133333
1    0.016667
Name: Fitness, dtype: float64
```

```
conditional_prob_kp781 = kp781_df.groupby('Gender')['Fitness'].value_counts(normalize=True)
marginal_prob_gender_kp781 = kp781_df['Gender'].value_counts(normalize=True)
marginal_prob_fitness_kp781 = kp781_df['Fitness'].value_counts(normalize=True)
```

```
conditional_prob_kp781
```

```
Gender  Fitness
Female  5          0.714286
        3          0.142857
        4          0.142857
Male    5          0.727273
        4          0.181818
        3          0.090909
Name: Fitness, dtype: float64
```

```
marginal_prob_gender_kp781
```

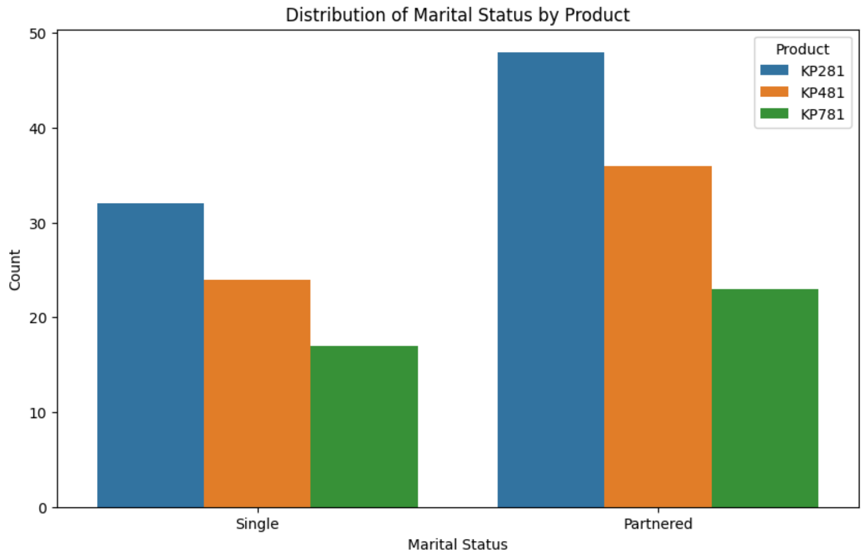
```
Male      0.825
Female    0.175
Name: Gender, dtype: float64
```

```
marginal_prob_fitness_kp781
```

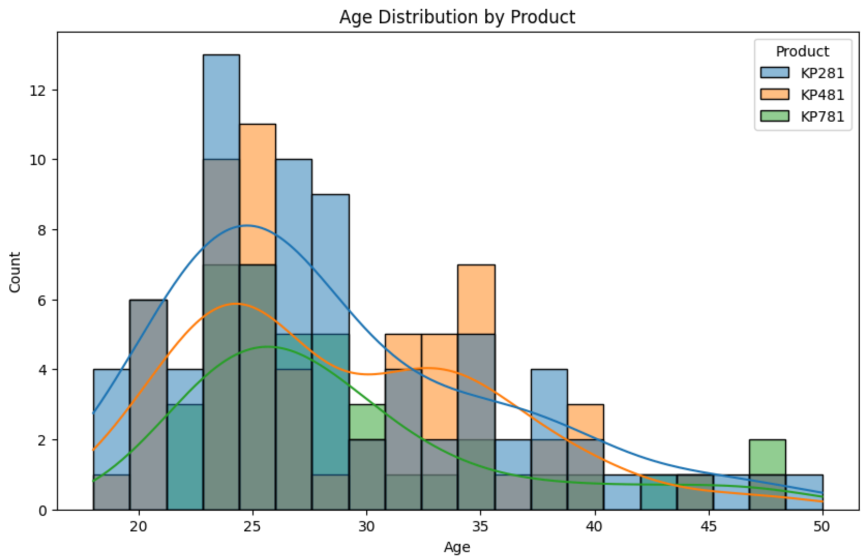
```
5    0.725
4    0.175
3    0.100
Name: Fitness, dtype: float64
```

Countplot to visualize the distribution of marital status by product

```
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='MaritalStatus', hue='Product')
plt.title('Distribution of Marital Status by Product')
plt.xlabel('Marital Status')
plt.ylabel('Count')
plt.show()
```




```
plt.figure(figsize=(10, 6))
sns.histplot(data=df, x='Age', hue='Product', bins=20, kde=True)
plt.title('Age Distribution by Product')
plt.xlabel('Age')
plt.ylabel('Count')
plt.show()
```



Representing the marginal probability like - what percent of customers have purchased KP281, KP481, or KP781 in a table

```
marginal_prob_table = pd.crosstab(index=df['Product'], columns='Count', normalize='all') * 100
marginal_prob_table.columns = ['Percentage']
marginal_prob_table.reset_index(inplace=True)
print(marginal_prob_table)
```

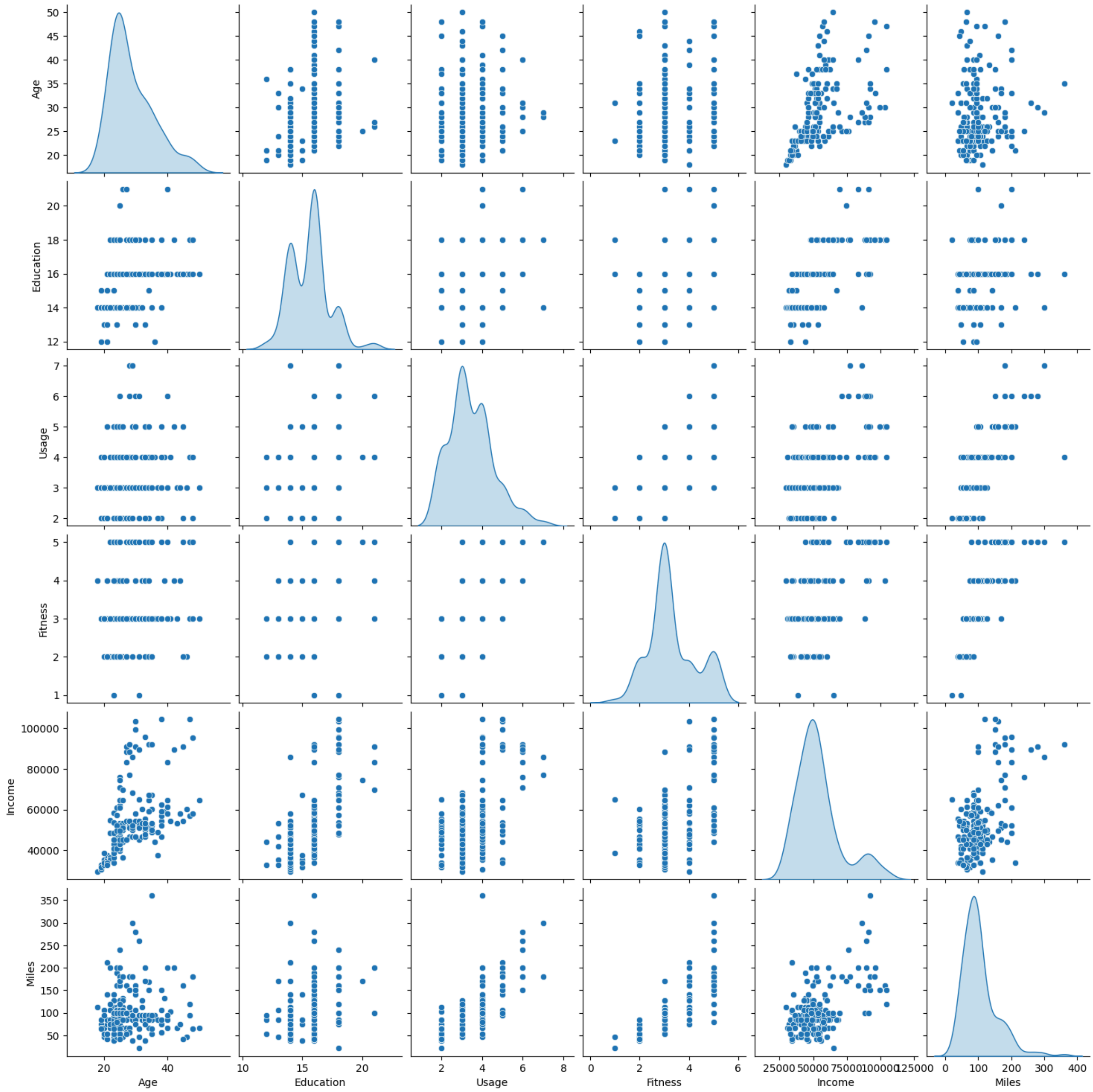
```
  Product  Percentage
0   KP281   44.444444
```

```
1    KP481    33.333333
2    KP781    22.222222
```

Check correlation among different factors using heat maps or pair plots

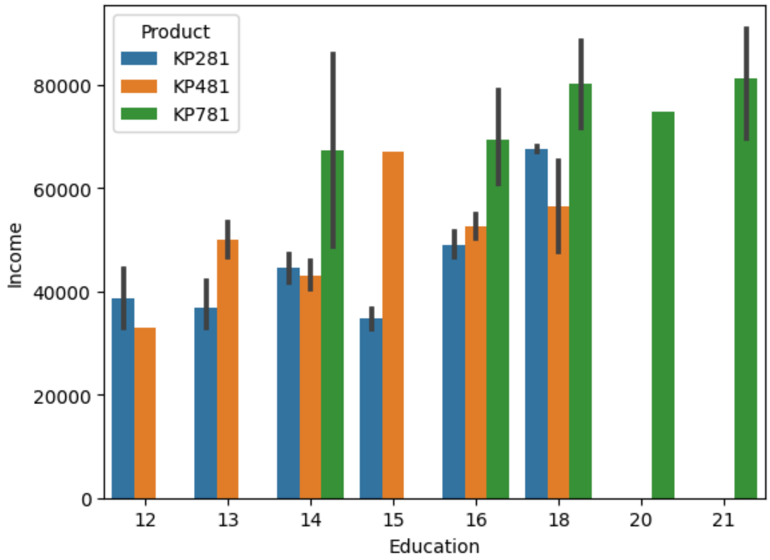
```
correlation_matrix = df.corr(numeric_only=True)


# Create a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Heatmap')
plt.show()
```


```
# Create a pair plot
sns.pairplot(df, diag_kind='kde')
plt.show()
```



Recommendation of Treadmill type based on factor Education, Income, Age.

```
sns.barplot(data = df, x = 'Education', y = 'Income', hue = 'Product')
plt.show()
```

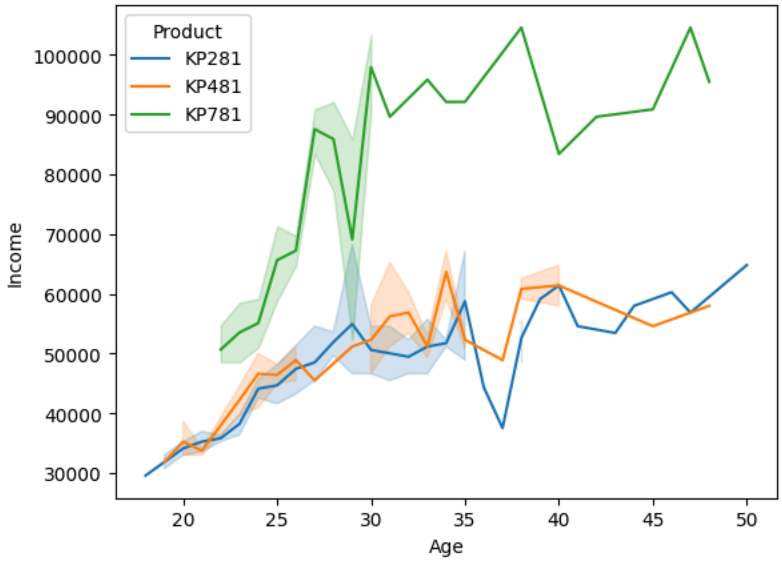


The probability of a male customer buying a KP781 treadmill

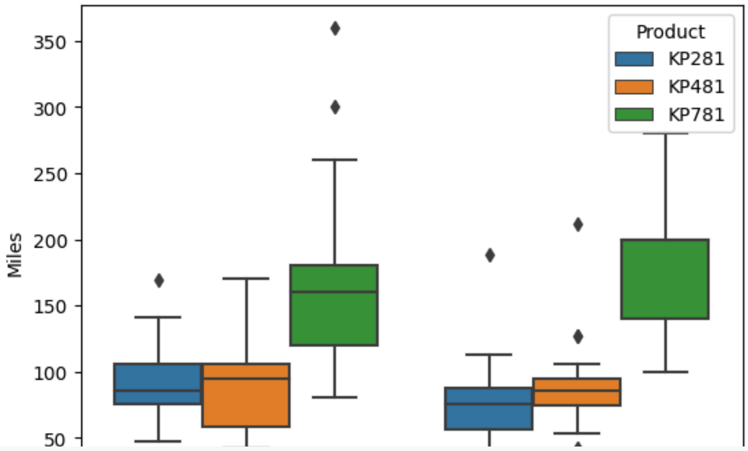
```
# Subset the DataFrame to include only male customers
male_customers = df[df['Gender'] == 'Male']

# Count the number of male customers who purchased KP781
male_kp781_customers = male_customers[male_customers['Product'] == 'KP781'].shape[0]

# Calculate the total number of male customers
total_male_customers = male_customers.shape[0]

# Calculate the probability
probability_male_kp781 = male_kp781_customers / total_male_customers

print("Probability of a male customer buying KP781 treadmill:", probability_male_kp781)
```

```
Probability of a male customer buying KP781 treadmill: 0.3173076923076923
```

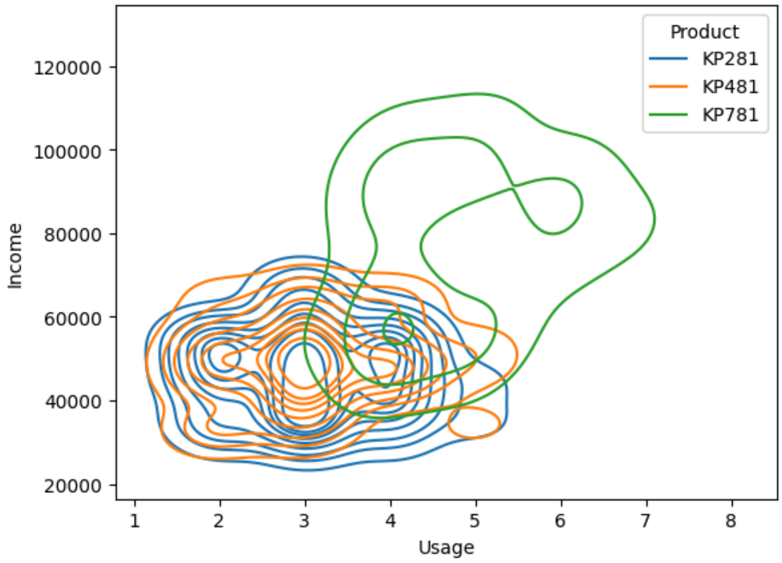
```
sns.lineplot(data = df, x = 'Age', y = 'Income', hue = 'Product')
plt.show()
```




```
sns.boxplot(data = df, x = 'Gender', y = 'Miles', hue = 'Product')
plt.show()
```




```
sns.kdeplot(data = df, x = 'Usage', y = 'Income', hue = 'Product')
plt.show()
```




```
pd.crosstab(df['Age'], [df['Income'], df['Product']])
```

Income	29562	30699	31836		32973		34110		35247	36384	...
Product	KP281	KP281	KP281	KP481	KP281	KP481	KP281	KP481	KP281	KP281	...
Age											
18	1	0	0	0	0	0	0	0	0	0	...
19	0	1	1	1	1	0	0	0	0	0	...
20	0	0	0	0	1	1	0	1	1	0	...
21	0	0	0	0	1	1	0	2	2	0	...
22	0	0	0	0	0	0	0	0	2	2	...
23	0	0	0	0	0	0	2	0	0	0	...
24	0	0	0	0	0	0	0	0	0	0	...
25	0	0	0	0	0	0	0	0	0	0	...
26	0	0	0	0	0	0	0	0	0	1	...
27	0	0	0	0	0	0	0	0	0	0	...
28	0	0	0	0	0	0	0	0	0	0	...
29	0	0	0	0	0	0	0	0	0	0	...
30	0	0	0	0	0	0	0	0	0	0	...
31	0	0	0	0	0	0	0	0	0	0	...
32	0	0	0	0	0	0	0	0	0	0	...
33	0	0	0	0	0	0	0	0	0	0	...
34	0	0	0	0	0	0	0	0	0	0	...