Business Case: Aerofit Business case study

- Downloaded the file from the specified Google Drive link and saves it as aerofit.csv

```
!wget  "https://drive.google.com/uc?export=download&id=18dRDrjuks-Fi1Z7uc8p2KvlALUQIaA6Y"  -O aerofit_data.csv
```

```
--2024-12-17 11:35:20--  https://drive.google.com/uc?export=download&id=18dRDrjuks-Fi1Z7uc8p2KvlALUQIaA6Y
Resolving drive.google.com (drive.google.com)... 142.251.184.101, 142.251.184.113, 142.251.184.100, ...
Connecting to drive.google.com (drive.google.com)|142.251.184.101|:443... connected.
HTTP request sent, awaiting response... 303 See Other
Location: https://drive.usercontent.google.com/download?id=18dRDrjuks-Fi1Z7uc8p2KvlALUQIaA6Y&export=download [following]
--2024-12-17 11:35:20--  https://drive.usercontent.google.com/download?id=18dRDrjuks-Fi1Z7uc8p2KvlALUQIaA6Y&export=download
Resolving drive.usercontent.google.com (drive.usercontent.google.com)... 74.125.126.132, 2607:f8b0:4001:c1d::84
Connecting to drive.usercontent.google.com (drive.usercontent.google.com)|74.125.126.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 7279 (7.1K) [application/octet-stream]
Saving to: ‘aerofit_data.csv’

aerofit_data.csv    100%[===================>]   7.11K  --.-KB/s    in 0s

2024-12-17 11:35:22 (43.3 MB/s) - ‘aerofit_data.csv’ saved [7279/7279]
```

1. Introduction

❓ What is Aerofit?

Aerofit is a leading brand in the field of fitness equipment. Aerofit provides a product range including machines such as treadmills, exercise bikes, gym equipment, and fitness accessories to cater to the needs of all categories of people.

🎯 Objective:

The market research team at AeroFit wants to identify the characteristics of the target audience for each type of treadmill offered by the company, to provide a better recommendation of the treadmills to the new customers. The team decides to investigate whether there are differences across the product with respect to customer characteristics.

📚 About Data:

The company collected the data on individuals who purchased a treadmill from the AeroFit stores during the prior three months.

📃 Features of the dataset:

- Product Purchased: KP281, KP481, or KP781
- Age: In years
- Gender: Male/Female
- Education: In years
- MaritalStatus: Single or partnered
- Usage: The average number of times the customer plans to use the treadmill each week.
- Income: Annual income (in $)
- Fitness: Self-rated fitness on a 1-to-5 scale, where 1 is the poor shape and 5 is the excellent shape.
- Miles: The average number of miles the customer expects to walk/run each week

📃 Product Portfolio:

- The KP281 is an entry-level treadmill that sells for $1,500.
- The KP481 is for mid-level runners that sell for $1,750.
- The KP781 treadmill is having advanced features that sell for $2,500.

2.Exploratory Data Analysis

```
#importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as msno
import warnings
warnings.filterwarnings('ignore')
#import copy
from wordcloud import WordCloud
```

```
# loading the dataset
df = pd.read_csv('aerofit_data.csv')
```

```
#to view full 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

```
#to view columns

df.columns
#df.keys()== df.columns
```

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

```
#view first 5 rows/records
df.head(5)
#view first 5 rows/records default=5
#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

```
#view last 5 rows/records,deafault=5
df.tail()
#view last 5 rows/records
#df.tail(5)
```

	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

```
#To get index of dataframe
df.index
```

```
RangeIndex(start=0, stop=180, step=1)
```

```
#To get shape information

df.shape
```

```
#180 rows and 9 columns
```

```
(180, 9)
```

```
# to get dimensional detail of dataframe

df.ndim

#2D
```

```
2
```

```
#Datatype

print(df.dtypes)
```

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

```
# Convert categorical attributes to 'category' data type if required
categorical_columns = ['Product', 'Gender', 'MaritalStatus']
for col in categorical_columns:
    df[col] = df[col].astype('category')

# Verify the changes
print("Data types after conversion:\n", df.dtypes)
```

```
Data types after conversion:
 Product          category
Age                 int64
Gender           category
Education           int64
MaritalStatus    category
Usage               int64
Fitness             int64
Income              int64
Miles               int64
dtype: object
```

```
# to get complete information of each column of dataframe like counts,datatype,memory usage.
#Note: For missing value in each column data type will be object

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    int64   
 7   Income         180 non-null    int64   
 8   Miles          180 non-null    int64   
dtypes: category(3), int64(6)
memory usage: 9.5 KB
```

🔍 Insights

From the above details it is clear that given dataframe is of dimension 2D with 180 rows and 9 columns.

Also we can also observe that there are no missing values for any columns .

📝 Statistical Summary

```
#for column with datatype as int, df.describe() will give statistical information like count,mean,min,max,std detail for that column.
```

```
df.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

```
#Statistical Summary: Generate a statistical summary of the dataset.
```

```
df.describe(include='all')
```

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
count	180	180.000000	180	180.000000	180	180.000000	180.000000	180.000000	180.000000
unique	3	NaN	2	NaN	2	NaN	NaN	NaN	NaN
top	KP281	NaN	Male	NaN	Partnered	NaN	NaN	NaN	NaN
freq	80	NaN	104	NaN	107	NaN	NaN	NaN	NaN
mean	NaN	28.788889	NaN	15.572222	NaN	3.455556	3.311111	53719.577778	103.194444
std	NaN	6.943498	NaN	1.617055	NaN	1.084797	0.958869	16506.684226	51.863605
min	NaN	18.000000	NaN	12.000000	NaN	2.000000	1.000000	29562.000000	21.000000
25%	NaN	24.000000	NaN	14.000000	NaN	3.000000	3.000000	44058.750000	66.000000
50%	NaN	26.000000	NaN	16.000000	NaN	3.000000	3.000000	50596.500000	94.000000
75%	NaN	33.000000	NaN	16.000000	NaN	4.000000	4.000000	58668.000000	114.750000
max	NaN	50.000000	NaN	21.000000	NaN	7.000000	5.000000	104581.000000	360.000000

🔍 Insights:

1) Product Popularity:

- The most popular product is KP281, with 80 occurrences out of 180.

2) Age Distribution:

- The average age of users is approximately 28.79 years.
- The age range spans from 18 to 50 years, with a standard deviation of 6.94 years.

3) Gender:

- The dataset has more male users (104 out of 180).

4) Education Level:

- The average education level is around 15.57 years, with a range from 12 to 21 years.

5) Marital Status:

- The majority of users are partnered (107 out of 180).

6) Usage:

- The average usage level is 3.46, with a range from 2 to 7.

7) Fitness Level:

- The average fitness level is 3.31, with a range from 1 to 5.

8) Income:

- The average income is approximately ₹53,719.58, with a range from ₹29,562 to ₹104,581.

9) Miles:

-The average miles covered is 103.19 miles, with a range from 21 to 360 miles.

👥 Duplicate Detection

```
df.duplicated().value_counts()
```

	count
False	180

🔍 Insights

- There are no duplicate entries in the dataset

Missing Value Analysis

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

	0
Product	0
Age	0
Gender	0
Education	0
MaritalStatus	0
Usage	0
Fitness	0
Income	0
Miles	0

🔍 Insights

- There is no missing values for all columns.

📍 **For Non-graphical Analysis:**

✅ Sanity Check for columns

```
# checking the unique values for columns
for i in df.columns:
  print('Unique Values in',i,'column are :-')
  print(df[i].unique())
  print('-'*70)
```

```
Unique Values in Product column are :-
['KP281', 'KP481', 'KP781']
Categories (3, object): ['KP281', 'KP481', 'KP781']
----------------------------------------------------------------------
Unique Values in Age column are :-
[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 column are :-
['Male', 'Female']
Categories (2, object): ['Female', 'Male']
----------------------------------------------------------------------
Unique Values in Education column are :-
[14 15 12 13 16 18 20 21]
----------------------------------------------------------------------
Unique Values in MaritalStatus column are :-
['Single', 'Partnered']
Categories (2, object): ['Partnered', 'Single']
```

```
----------------------------------------------------------------------
Unique Values in Usage column are :-
[3 2 4 5 6 7]
----------------------------------------------------------------------
Unique Values in Fitness column are :-
[4 3 2 1 5]
----------------------------------------------------------------------
Unique Values in Income column are :-
[ 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 column are :-
[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]
----------------------------------------------------------------------
```

```
# to understand the diversity of data in each specified column.
for i in df.columns:
    print('Unique Values in',i,'column are :-')
    print(df[i].nunique())
    print('-'*70)
```

```
Unique Values in Product column are :-
3
----------------------------------------------------------------------
Unique Values in Age column are :-
32
----------------------------------------------------------------------
Unique Values in Gender column are :-
2
----------------------------------------------------------------------
Unique Values in Education column are :-
8
----------------------------------------------------------------------
Unique Values in MaritalStatus column are :-
2
----------------------------------------------------------------------
Unique Values in Usage column are :-
6
----------------------------------------------------------------------
Unique Values in Fitness column are :-
5
----------------------------------------------------------------------
Unique Values in Income column are :-
62
----------------------------------------------------------------------
Unique Values in Miles column are :-
37
----------------------------------------------------------------------
```

```
for i in df.columns:
  print('Value count in',i,'column are :-')
  print(df[i].value_counts())
  print('-'*70)
```

```
Value count in Product column are :-
Product
KP281    80
KP481    60
KP781    40
Name: count, dtype: int64
----------------------------------------------------------------------
Value count in Age column are :-
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
---------------------------------------------------------------------
Value count in Gender column are :-
Gender
Male      104
Female     76
Name: count, dtype: int64
---------------------------------------------------------------------
Value count in Education column are :-
Education
16    85
14    55
18    23
15     5
13     5
12     3
21     3
20     1
Name: count, dtype: int64
---------------------------------------------------------------------
Value count in MaritalStatus column are :-
MaritalStatus
Partnered    107
Single        73
Name: count, dtype: int64
---------------------------------------------------------------------
Value count in Usage column are :-
Usage
3    69
4    52
2    33
5    17
6     7
7     2
Name: count, dtype: int64
---------------------------------------------------------------------
Value count in Fitness column are :-
Fitness
3    97
5    31
2    26
4    24
1     2
Name: count, dtype: int64
---------------------------------------------------------------------

Value count in Income column are :-
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
---------------------------------------------------------------------
Value count in Miles column are :-
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
---------------------------------------------------------------------
```

🔍 Insights

1) Product Variety:

- There are 3 unique products in the dataset, indicating a limited range of products being analyzed.

2) Age Diversity:

- The dataset includes 32 unique age values, reflecting a wide range of age groups among the users.

3) Gender Representation:

- There are 2 unique gender values, indicating the dataset includes both male and female users.

4) Education Levels:

- The dataset contains 8 unique education levels, showcasing a variety of educational backgrounds.

5) Marital Status:

- There are 2 unique marital status values, indicating the dataset includes both partnered and single users.

6) Usage Patterns:

- The dataset includes 6 unique usage values, reflecting different levels of product usage among the users.

7) Fitness Levels:

- There are 5 unique fitness levels, indicating a range of fitness levels among the users.

8) Income Range:

- The dataset contains 62 unique income values, showcasing a diverse range of income levels.

9) Miles Covered:

- There are 37 unique values for miles covered, indicating a wide range of distances covered by the users.

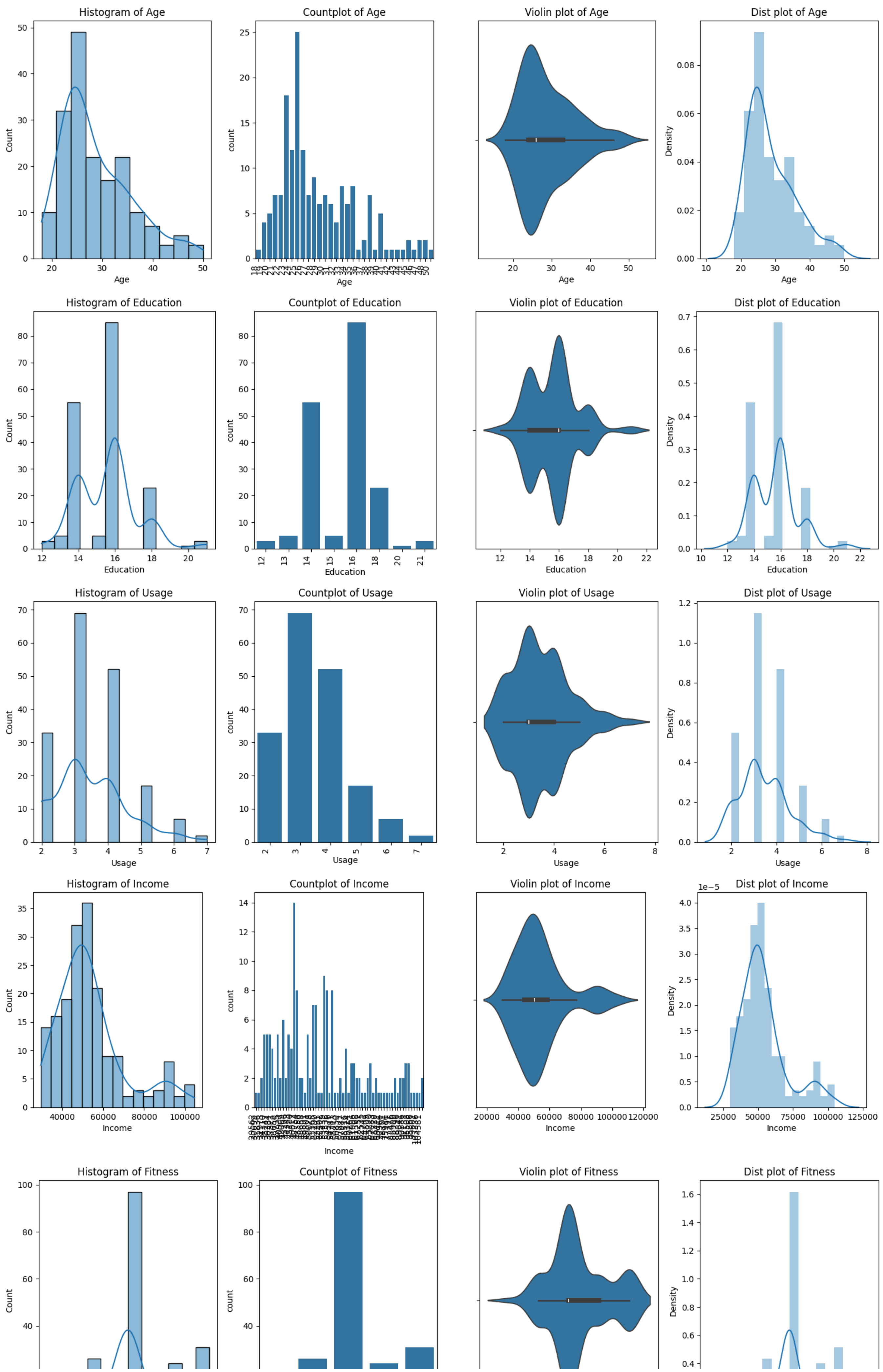
```
df.info()
```

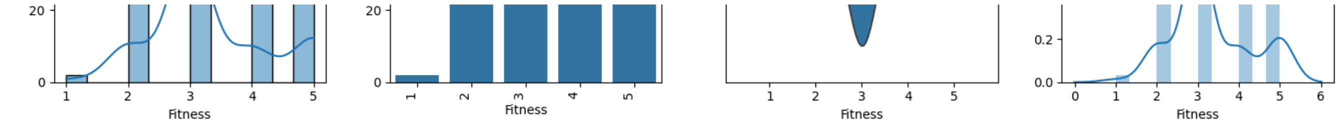
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 180 entries, 0 to 179
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype   
---  ------         --------------  -----   
 0   Product        180 non-null    category
 1   Age            180 non-null    int64   
 2   Gender         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    int64   
 7   Income         180 non-null    int64   
 8   Miles          180 non-null    int64   
dtypes: category(3), int64(6)
memory usage: 9.5 KB
```

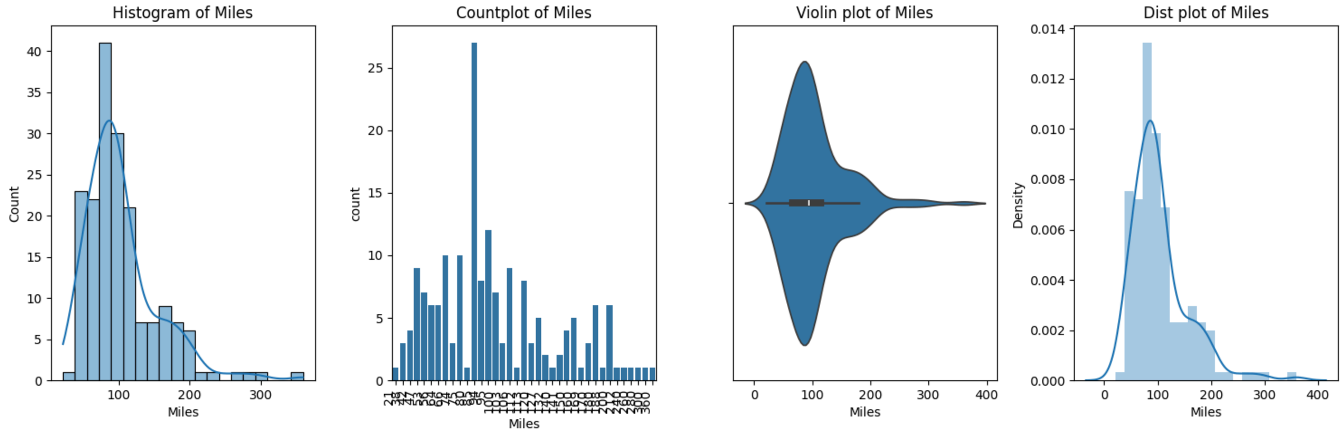
```
df['Age'] = df['Age'].astype(int)
df['Usage'] = df['Usage'].astype(int)
df['Income'] = df['Income'].astype(int)
df['Miles'] = df['Miles'].astype(int)
df['Fitness'] = df['Fitness'].astype(int)
```

Visual Analysis

```
# List of continuous columns
continuous_columns = ['Age', 'Education', 'Usage', 'Income', 'Fitness', 'Miles']
for column in continuous_columns:
  plt.figure(figsize=(15, 5))
  # Histogram
  plt.subplot(1, 4, 1)
  sns.histplot(df[column].dropna(), kde=True)
  plt.title(f'Histogram of {column}')
  plt.tight_layout()

  # Countplot
  plt.subplot(1, 4, 2)
  sns.countplot(x=df[column].dropna())
  plt.title(f'Countplot of {column}')
  plt.xticks(rotation=90, ha='right') # Rotate x-axis labels
  plt.tight_layout()

  # Violin plot
  plt.subplot(1, 4, 3)
  sns.violinplot(x=df[column].dropna())
  plt.title(f'Violin plot of {column}')
  plt.tight_layout()

  # Dist plot
  plt.subplot(1, 4, 4)
  sns.distplot(df[column].dropna())
  plt.title(f'Dist plot of {column}')
  plt.tight_layout()
  plt.show()
```


Histogram of Age
Countplot of Age
Violin plot of Age
Dist plot of Age
Count
count
Density
Age
Histogram of Education
Countplot of Education
Violin plot of Education
Dist plot of Education
Education
Histogram of Usage
Countplot of Usage
Violin plot of Usage
Dist plot of Usage
Usage
Histogram of Income
Countplot of Income
Violin plot of Income
Dist plot of Income
1e−5
Income
Histogram of Fitness
Countplot of Fitness
Violin plot of Fitness
Dist plot of Fitness





🔍 Insights for Continuous Variables

1) Age:

- Average: ~28.79 years
- Range: 18 to 50 years
- Variability: Moderate (Standard Deviation: 6.94 years)
- Distribution: The histogram and dist plot show a fairly normal distribution with a slight skew towards younger ages.

2) Income:

- Average: ~₹53,719.58
- Range: ₹29,562 to ₹104,581
- Variability: High (Standard Deviation: ₹16,506.68)
- Distribution: The income distribution is right-skewed, indicating more individuals with lower incomes and fewer with higher incomes

3) Miles:

- Average: ~103.19 miles
- Range: 21 to 360 miles
- Variability: Considerable (Standard Deviation: 51.86 miles)
- Distribution: The histogram and dist plot show a wide range of miles covered, with a concentration around the average.

4) Usage:

- Average: ~3.46
- Range: 2 to 7
- Variability: Moderate (Standard Deviation: 1.08)
- Distribution: The histogram and dist plot show a concentration around the average usage, with fewer instances at the extremes.

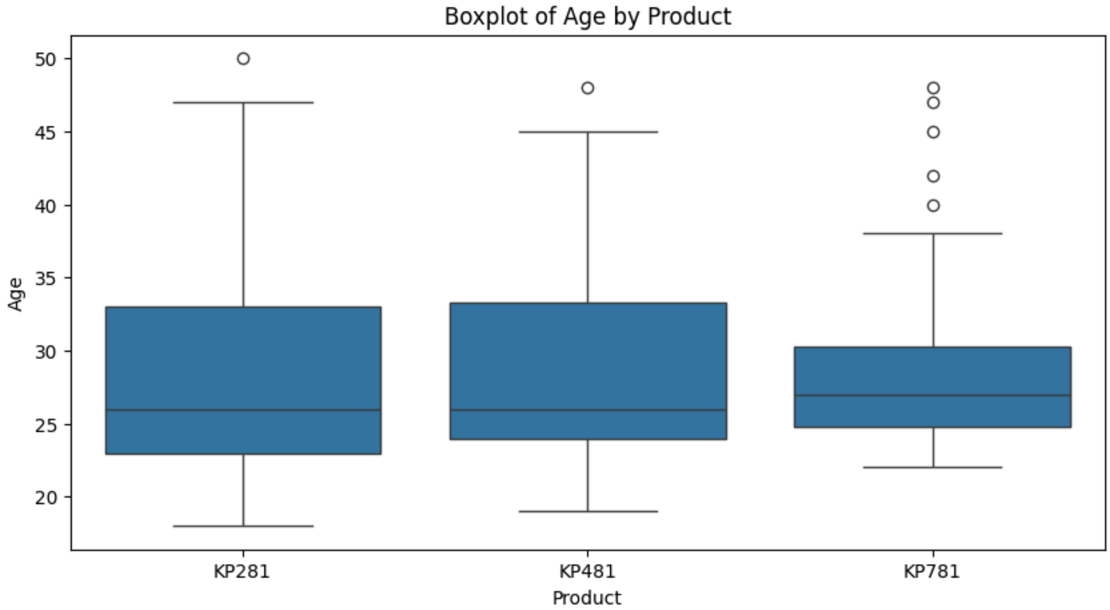
5) Fitness:

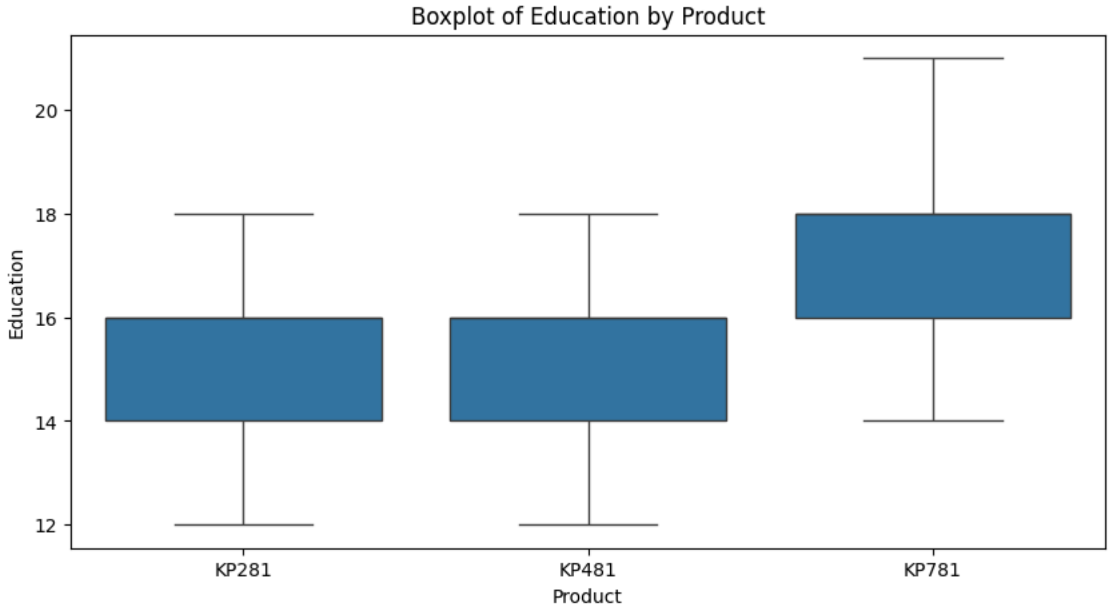
- Distribution: The plots suggest a varied level of fitness among individuals, with some common fitness levels standing out.

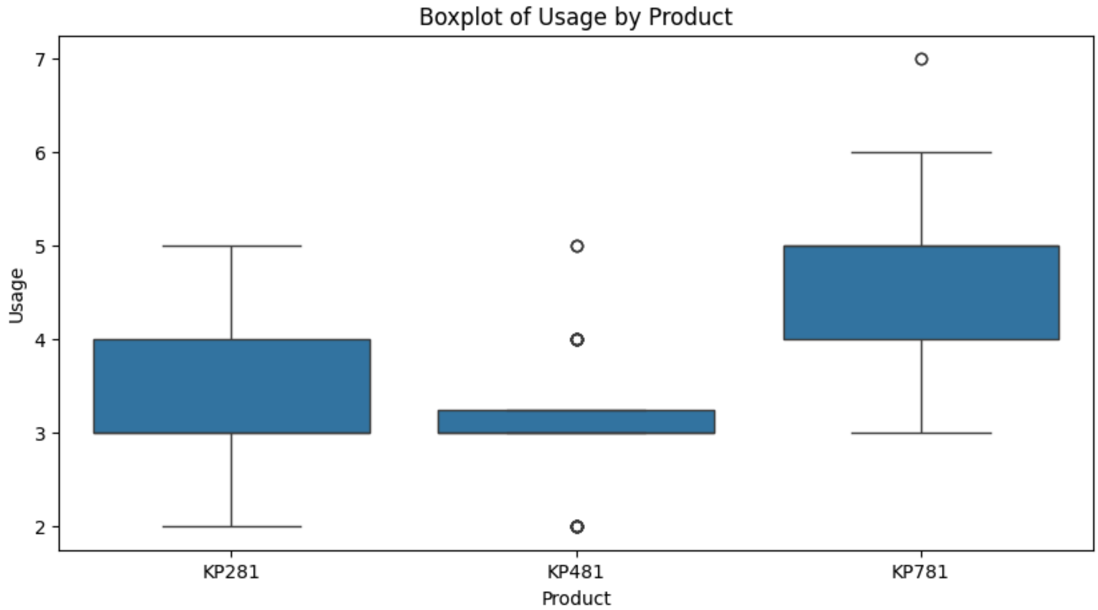
6) Education:

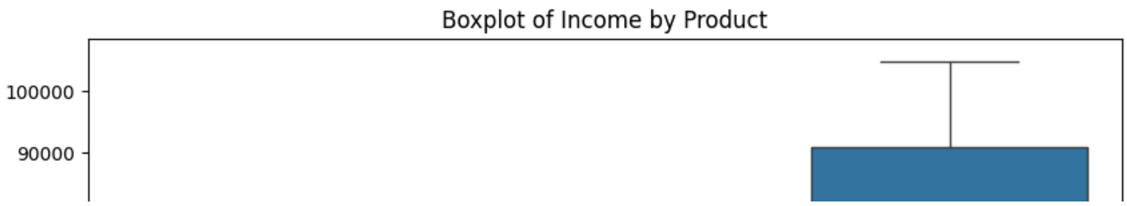
- Distribution: The plots indicate a varied distribution of education levels, with some peaks suggesting common education levels among the dataset.

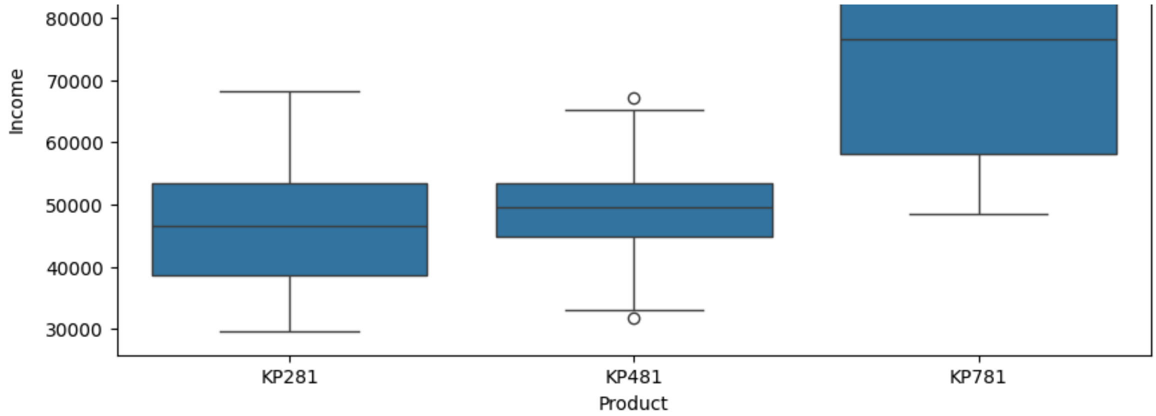
```
# List of continuous columns to plot against categorical columns

categorical_columns = ['Product', 'Gender', 'MaritalStatus']
continuous_columns = ['Age', 'Education', 'Usage', 'Income', 'Fitness', 'Miles']
# Create boxplots for each categorical column against each continuous column
for cat_col in categorical_columns:
  for cont_col in continuous_columns:
    plt.figure(figsize=(10, 5))
    sns.boxplot(x=cat_col, y=cont_col, data=df)
    plt.title(f'Boxplot of {cont_col} by {cat_col}')
    plt.show()
```


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

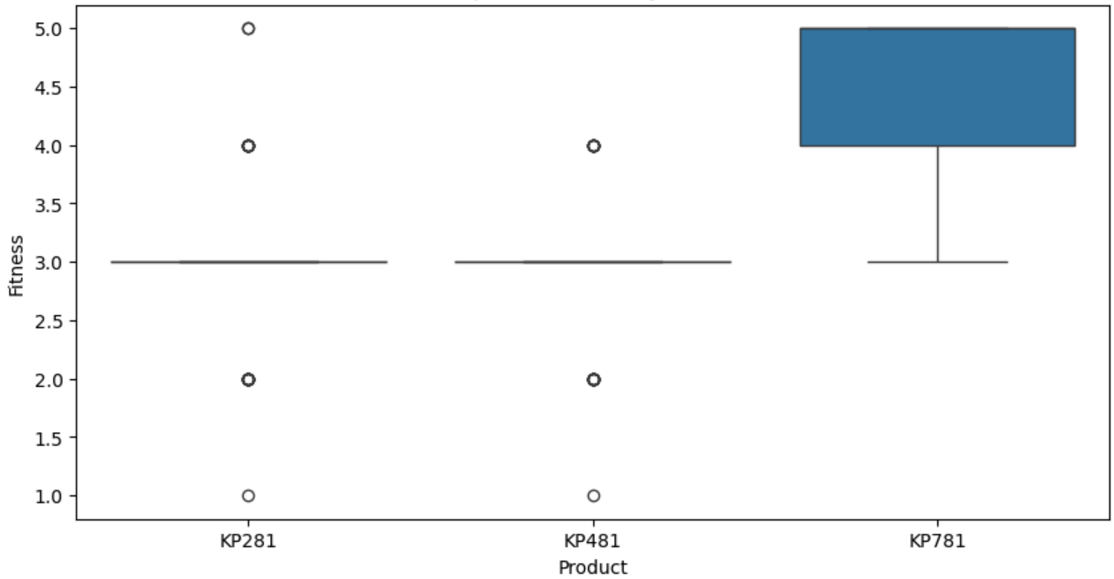

Boxplot of Education by Product
20
18
16
14
12
Education
KP281
KP481
KP781
Product


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

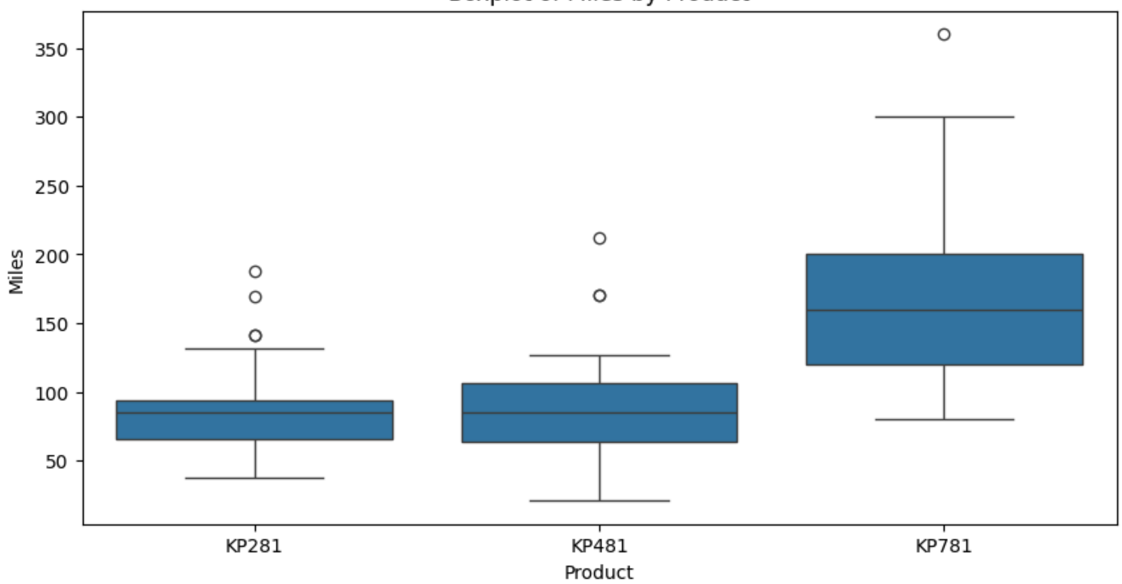

Boxplot of Income by Product
100000
90000



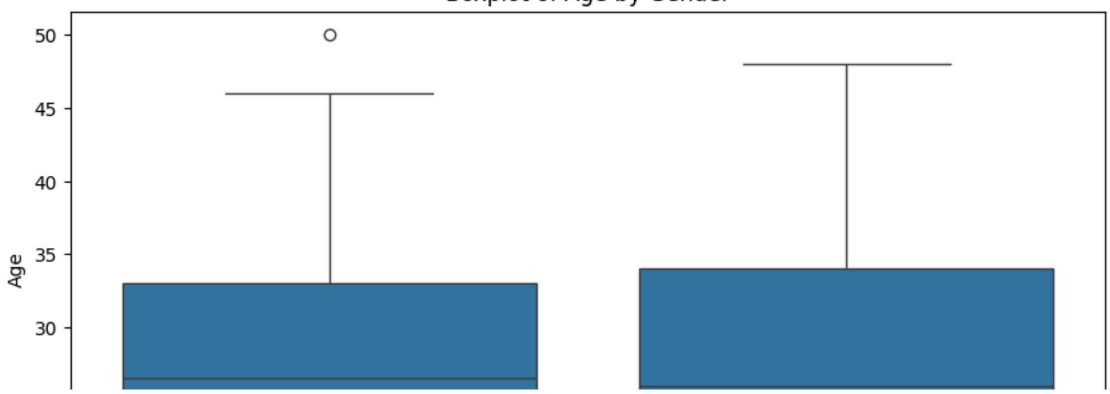
Boxplot of Fitness by Product

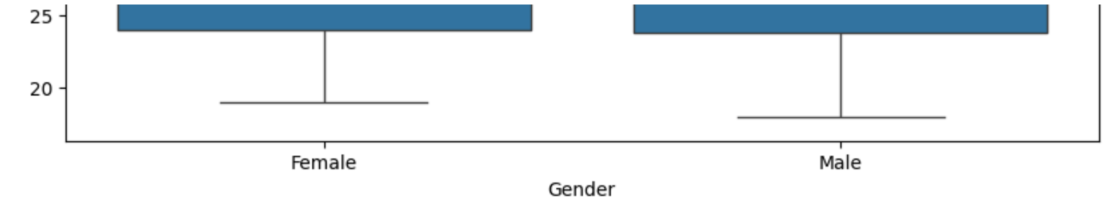


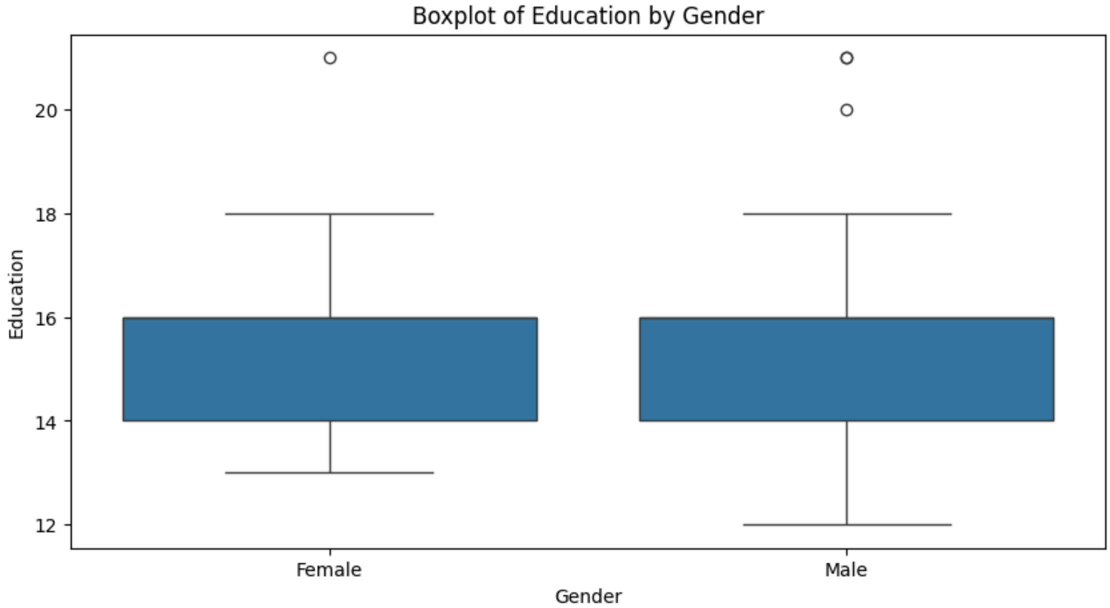
Boxplot of Miles by Product

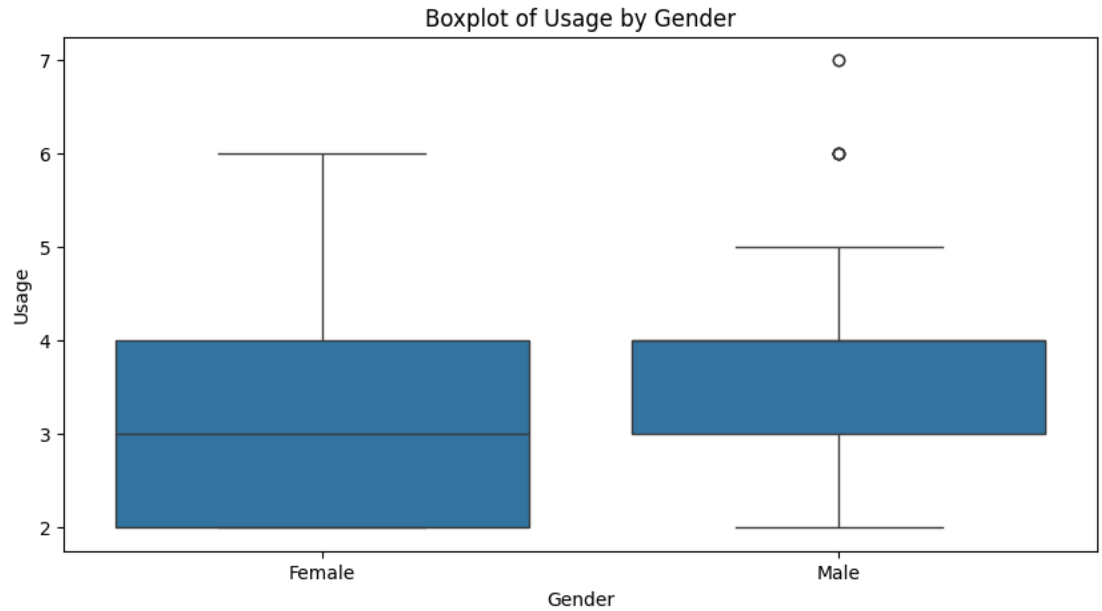


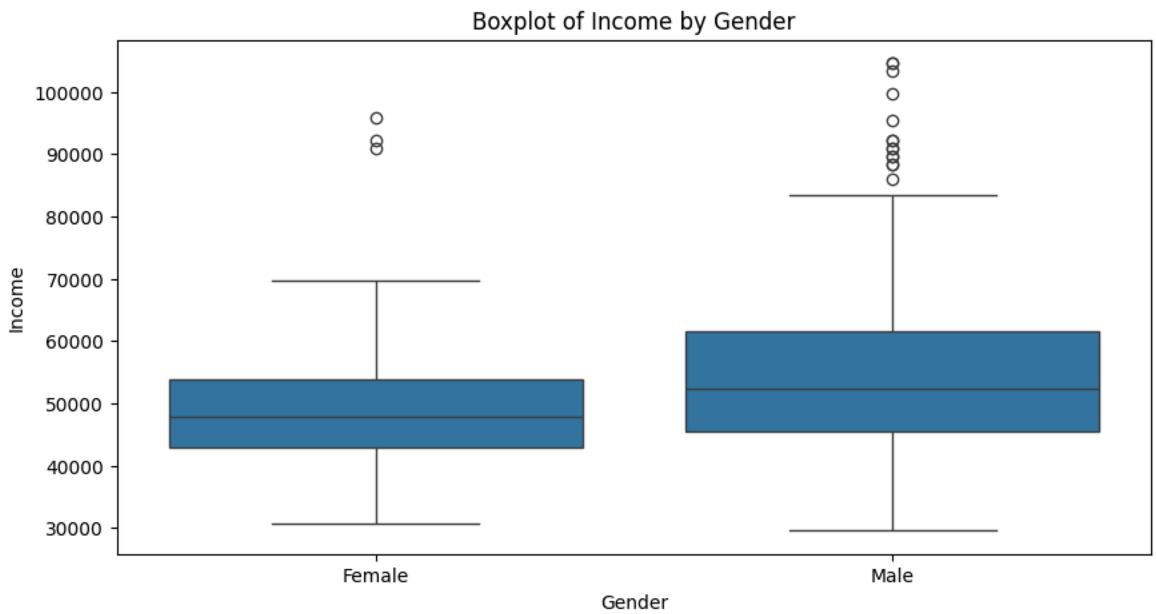
Boxplot of Age by Gender

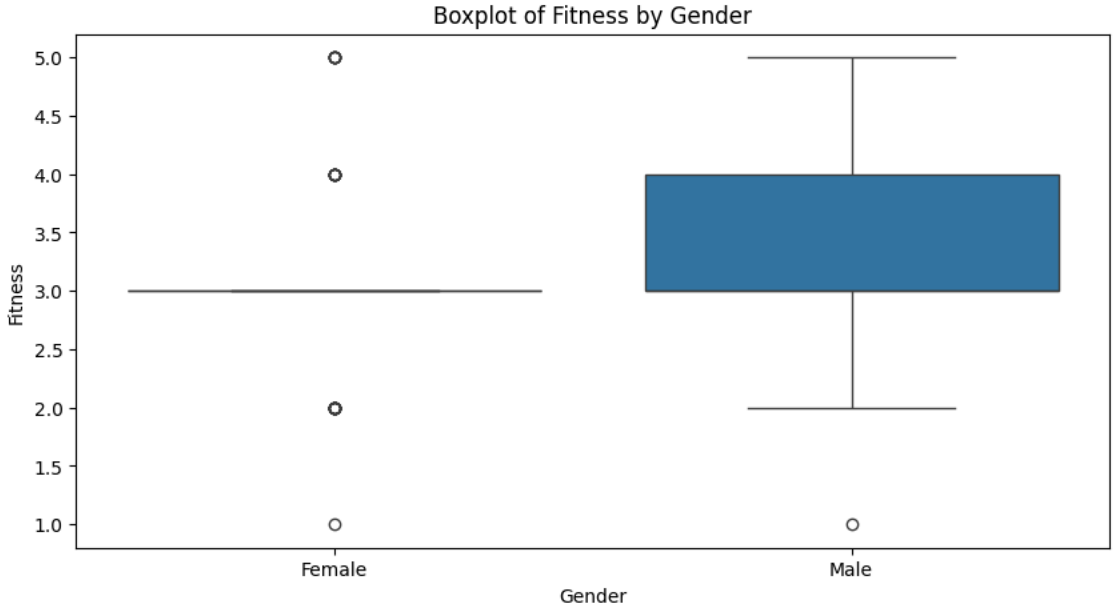


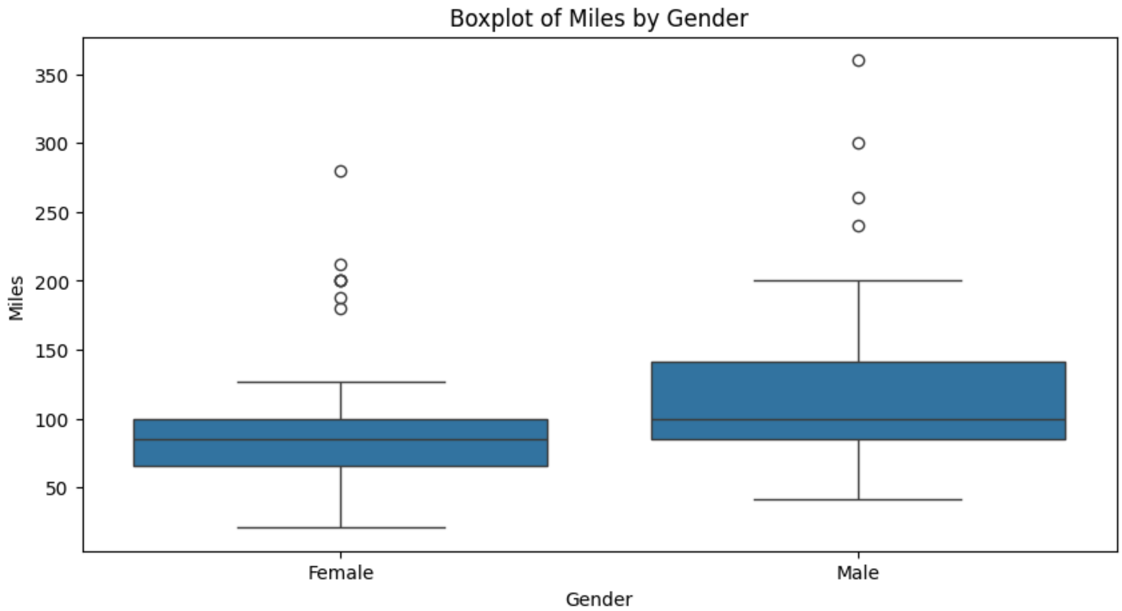

25
20
Female
Male
Gender

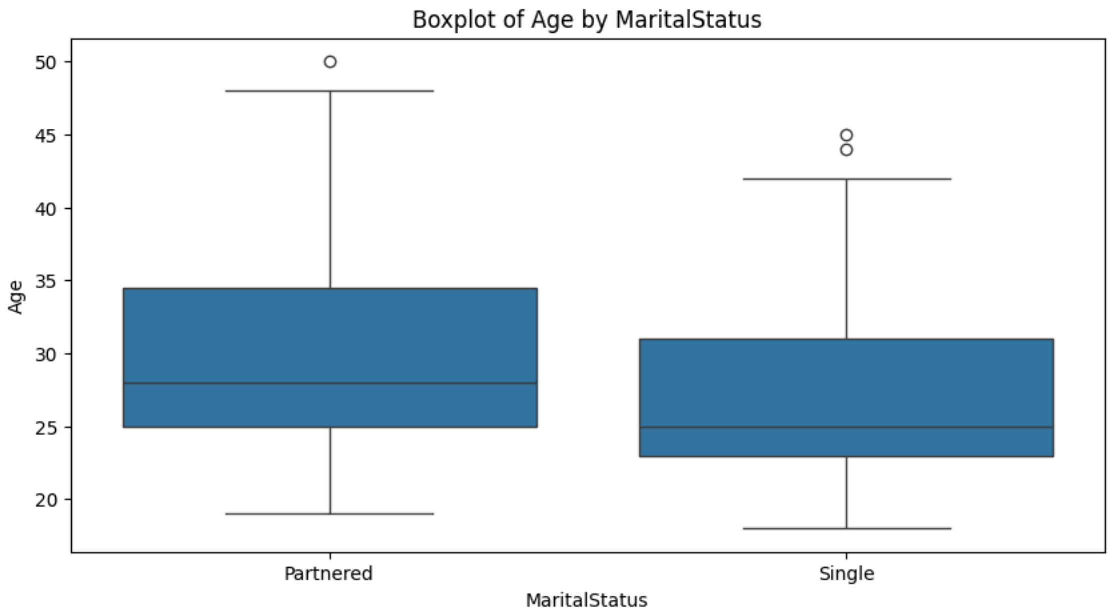

Boxplot of Education by Gender
Education
20
18
16
14
12
Female
Male
Gender

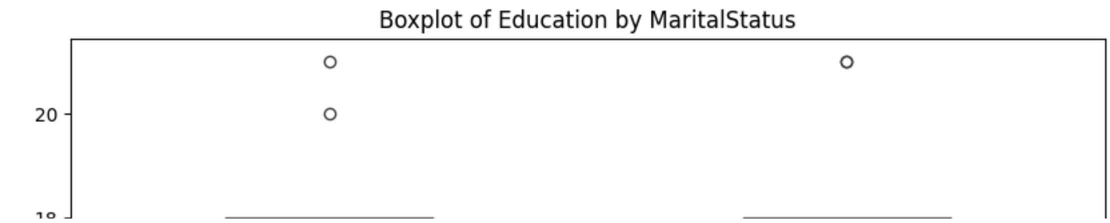

Boxplot of Usage by Gender
Usage
7
6
5
4
3
2
Female
Male
Gender

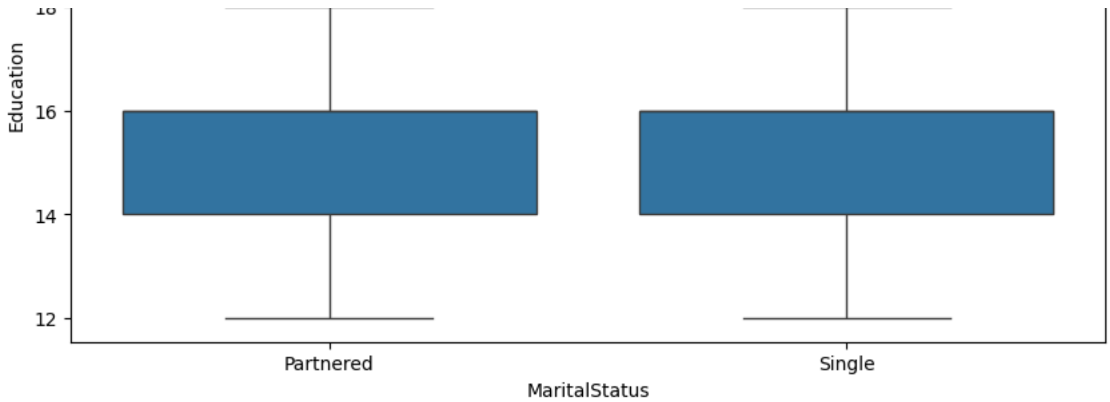

Boxplot of Income by Gender
Income
100000
90000
80000
70000
60000
50000
40000
30000
Female
Male
Gender


Boxplot of Fitness by Gender
Fitness
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
Female
Male
Gender

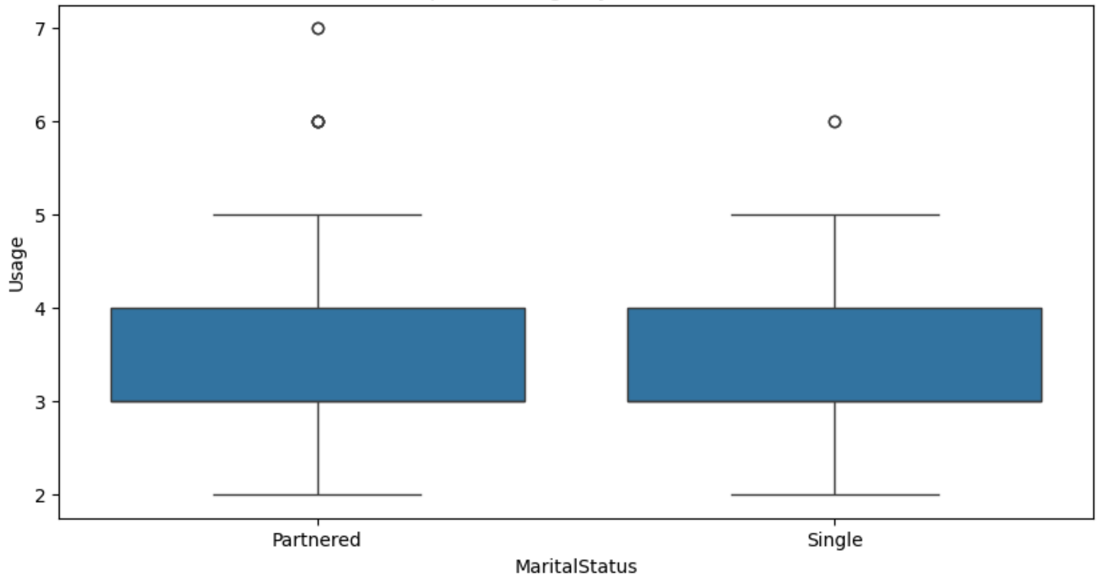

Boxplot of Miles by Gender
Miles
350
300
250
200
150
100
50
Female
Male
Gender


Boxplot of Age by MaritalStatus
Age
50
45
40
35
30
25
20
Partnered
Single
MaritalStatus

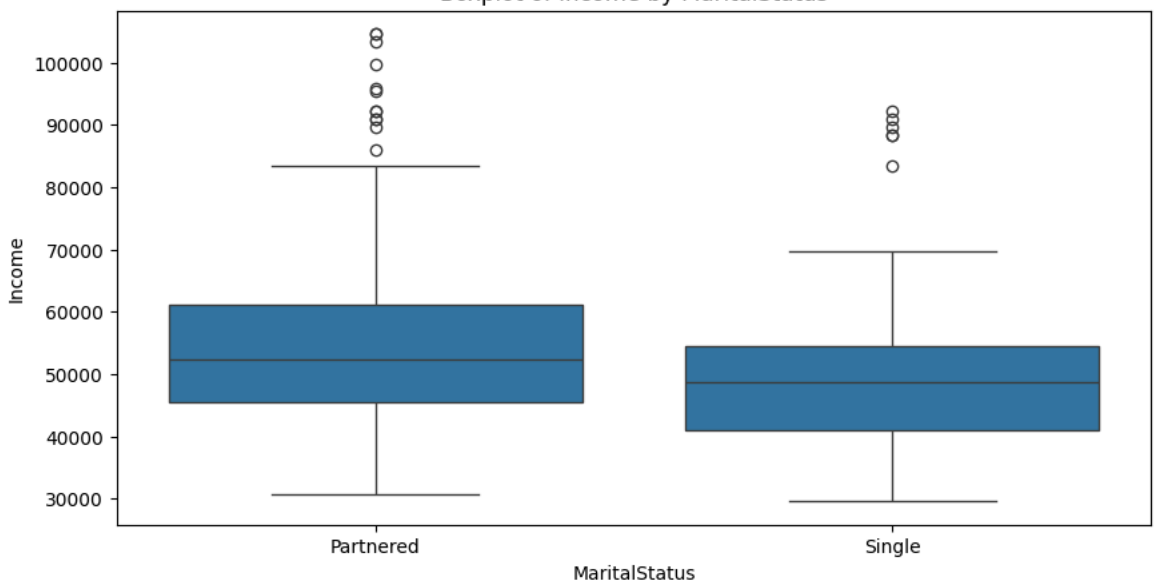

Boxplot of Education by MaritalStatus
20



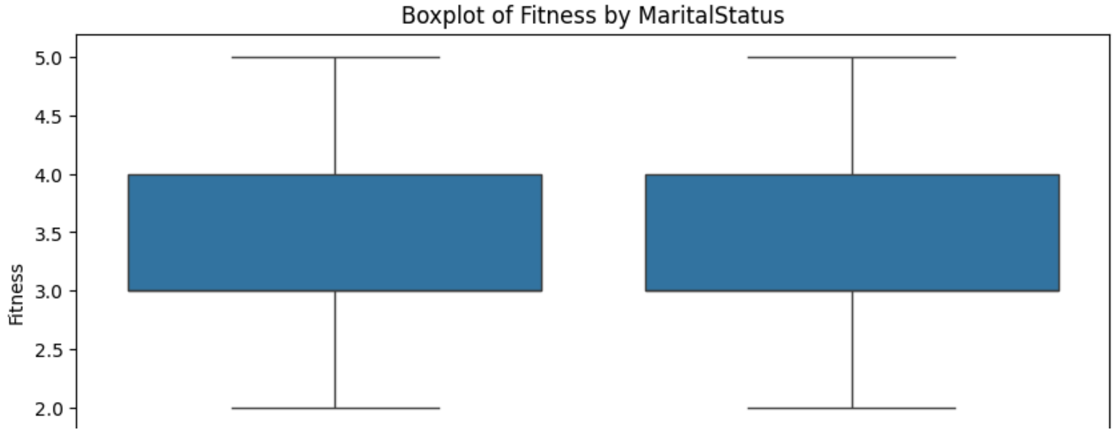
Boxplot of Usage by MaritalStatus

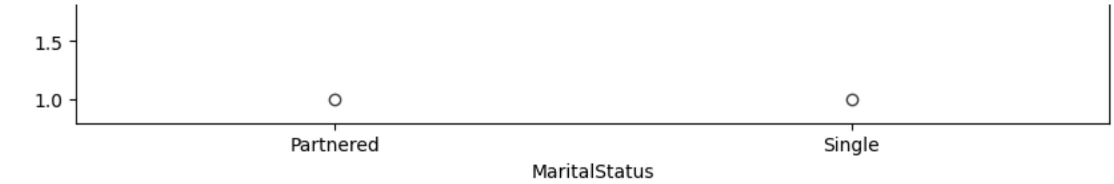


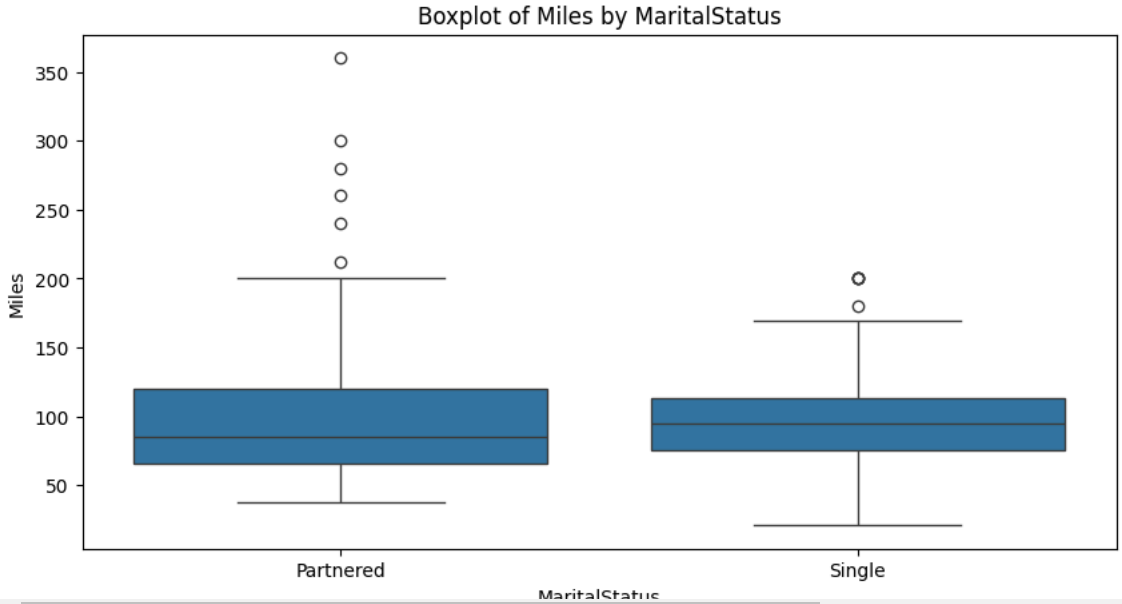
Boxplot of Income by MaritalStatus



Boxplot of Fitness by MaritalStatus




1.5
1.0
Partnered
Single
MaritalStatus


Boxplot of Miles by MaritalStatus
Miles
350
300
250
200
150
100
50
Partnered
Single
MaritalStatus

```
# Generate cross tab for each categorical column against each continuous column
# Generate styled cross tabs
for cat_col in categorical_columns:
    for cont_col in continuous_columns:
        cross_tab = pd.crosstab(df[cat_col], df[cont_col])
        styled_cross_tab = cross_tab.style.background_gradient(cmap='viridis').set_caption(f'Cross tab of {cont_col} by {cat_col}')
        display(styled_cross_tab)
        print('\n')
```

Cross tab of Age by Product

Age	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	50
Product																																
KP281	1	3	2	4	4	8	5	7	7	3	6	3	2	2	2	2	2	3	1	1	4	1	1	1	0	1	1	0	1	1	0	1
KP481	0	1	3	3	0	7	3	11	3	1	0	1	2	3	2	5	3	4	0	1	2	0	3	0	0	0	0	1	0	0	1	0
KP781	0	0	0	0	3	3	4	7	2	3	3	2	3	1	0	1	1	1	0	0	1	0	1	0	1	0	0	1	0	1	1	0

Cross tab of Education by Product

Education	12	13	14	15	16	18	20	21
Product								
KP281	2	3	30	4	39	2	0	0
KP481	1	2	23	1	31	2	0	0
KP781	0	0	2	0	15	19	1	3

Cross tab of Usage by Product

Usage	2	3	4	5	6	7
Product						
KP281	19	37	22	2	0	0
KP481	14	31	12	3	0	0
KP781	0	1	18	12	7	2

Income	29562	30699	31836	32973	34110	35247	36384	37521	38658	39795	40932	42069	43206	44343	45480	46617	47754	48556
Product																		
KP281	1	1	1	3	2	5	3	2	3	2	4	2	1	4	5	7	0	0
KP481	0	0	1	2	3	0	1	0	2	0	2	0	4	0	9	1	2	0
KP781	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2

Cross tab of Fitness by Product

Fitness	1	2	3	4	5
Product					
KP281	1	14	54	9	2
KP481	1	12	39	8	0
KP781	0	0	4	7	29

Cross tab of Miles by Product

Miles	21	38	42	47	53	56	64	66	74	75	80	85	94	95	100	103	106	112	113	120	127	132	140	141	150	160	169	170	1
Product																													
KP281	0	3	0	9	0	6	0	10	0	10	0	16	8	0	0	3	0	1	8	0	0	2	0	2	0	0	1	0	
KP481	1	0	4	0	7	0	6	0	3	0	0	11	0	12	0	0	8	0	0	0	5	0	0	0	0	0	0	2	
KP781	0	0	0	0	0	0	0	0	0	0	1	0	0	0	7	0	1	0	0	3	0	0	1	0	4	5	0	1	

Cross tab of Age by Gender

Age	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	50
Gender																																
Female	0	1	2	3	3	7	6	10	6	2	5	3	3	3	1	6	2	4	0	2	2	0	2	0	0	0	1	0	1	0	0	1
Male	1	3	3	4	4	11	6	15	6	5	4	3	4	3	3	2	4	4	1	0	5	1	3	1	1	1	0	2	0	2	2	0

Cross tab of Education by Gender

Education	12	13	14	15	16	18	20	21
Gender								
Female	0	1	30	2	35	7	0	1
Male	3	4	25	3	50	16	1	2

Male	3	4	25	3	50	16	1	2

Cross tab of Usage by Gender

Usage	2	3	4	5	6	7
Gender						
Female	20	33	14	7	2	0
Male	13	36	38	10	5	2

Income	29562	30699	31836	32973	34110	35247	36384	37521	38658	39795	40932	42069	43206	44343	45480	46617	47754	48556
Gender																		
Female	0	1	0	1	4	3	2	2	1	1	3	1	3	2	7	6	2	0
Male	1	0	2	4	1	2	2	0	4	1	3	1	2	2	7	2	0	2

Cross tab of Fitness by Gender

Fitness	1	2	3	4	5
Gender					
Female	1	16	45	8	6
Male	1	10	52	16	25

Cross tab of Miles by Gender

Miles	21	38	42	47	53	56	64	66	74	75	80	85	94	95	100	103	106	112	113	120	127	132	140	141	150	160	169	170	1
Gender																													
Female	1	3	1	4	2	4	3	8	3	6	0	13	4	4	2	1	4	0	4	0	2	0	0	0	0	0	0	0	
Male	0	0	3	5	5	2	3	2	0	4	1	14	4	8	5	2	5	1	4	3	3	2	1	2	4	5	1	3	

Cross tab of Age by MaritalStatus

Age	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	4
MaritalStatus																															
Partnered	0	1	3	5	1	9	3	18	7	4	6	5	4	4	2	6	2	6	0	2	6	1	3	1	0	1	0	1	1	2	
Single	1	3	2	2	6	9	9	7	5	3	3	1	3	2	2	2	4	2	1	0	1	0	2	0	1	0	1	1	0	0	

Cross tab of Education by MaritalStatus

Education	12	13	14	15	16	18	20	21
MaritalStatus								
Partnered	1	3	29	2	56	14	1	1
Single	2	2	26	3	29	9	0	2

Cross tab of Usage by MaritalStatus

Usage	2	3	4	5	6	7
MaritalStatus						
Partnered	22	40	29	9	5	2
Single	11	29	23	8	2	0

Income	29562	30699	31836	32973	34110	35247	36384	37521	38658	39795	40932	42069	43206	44343	45480	46617	47754
MaritalStatus																	
Partnered	0	1	0	2	4	3	1	2	2	2	2	1	2	3	8	5	1
Single	1	0	2	3	1	2	3	0	3	0	4	1	3	1	6	3	1

Cross tab of Fitness by MaritalStatus

Fitness	1	2	3	4	5
MaritalStatus					

Partnered	1	18	57	13	18
Single	1	8	40	11	13

Cross tab of Miles by MaritalStatus

Miles	21	38	42	47	53	56	64	66	74	75	80	85	94	95	100	103	106	112	113	120	127	132	140	141	150	160	169
MaritalStatus																											
Partnered	0	2	3	8	4	5	3	8	1	5	1	16	5	9	2	1	4	0	2	2	1	2	0	1	2	3	0
Single	1	1	1	1	3	1	3	2	2	5	0	11	3	3	5	2	5	1	6	1	4	0	1	1	2	2	1

Insights:

1) Age by Product

- KP281: This product is popular across a wide age range, with notable peaks at ages 23, 25, and 26.
- KP481: This product has a significant number of users at ages 25 and 23, with a noticeable drop in users in their late 20s.
- KP781: This product is less popular overall but has some users between ages 22-30.

2) Education by Product

- KP281: Most users have 14 or 16 years of education.
- KP481: Similar to KP281, with a majority having 14 or 16 years of education.
- KP781: Users are more likely to have 18 years of education.

3) Usage by Product

- KP281: Most users have a usage level of 3, followed by 4.
- KP481: Similar to KP281, with a majority at usage level 3.
- KP781: Users are more evenly distributed across usage levels, with a peak at 4 and 5.

4) Income by Product

- KP281: Users are spread across various income levels, with some peaks at $45480,$46617, and $54576.
- KP481: Users are also spread across income levels, with peaks at $45480 and$50028.
- KP781: Users tend to have higher incomes, with peaks at $90886 and$92131.

6) Fitness by Product

- KP281: Most users have a fitness level of 3.
- KP481: Similar to KP281, with a majority at fitness level 3.
- KP781: Users are more likely to have a fitness level of 5.

7) Miles by Product

- KP281: Users are spread across various mileage levels, with peaks at 85 and 94 miles.
- KP481: Users have peaks at 85 and 95 miles.
- KP781: Users tend to have higher mileage, with peaks at 100, 200 and 180 miles.

8) Age by Gender

- Female: Users are spread across ages, with peaks at 23,24, 25, 33 and 26.
- Male: Users are also spread across ages, with peaks at 23, 24, 25, and 26.

9) Education by Gender

- Female: Most users have 14 or 16 years of education.
- Male: Similar to females, with a majority having 14 or 16 years of education.

10) Usage by Gender

- Female: Most users have a usage level of 3.
- Male: Users are more evenly distributed across usage levels, with peaks at 3 and 4.

11) Income by Gender

- Female: The income distribution for females shows peaks at $45,480 and$50,028.
- Male: Males have a more even distribution across income levels, with notable peaks at $45,480,$53,439, and $54,576. Males are more represented in higher income brackets compared to females.

12) Fitness by Gender

- Female: Most females have a fitness level of 3, followed by 2. There are fewer females with fitness levels of 4 and 5.
- Male: Males are more evenly distributed across fitness levels, with a significant number at level 3 and a notable peak at level 5.

13) Miles by Gender

- Female: Females have peaks at 85 and 94 miles, with fewer individuals in higher mileage categories.
- Male: Males show a more even distribution across mileage, with peaks at 85, 95, and 100 miles.

14) Age by Marital Status

- Partnered: The age distribution for partnered individuals shows peaks at 25 and 26, with a significant number in their early 30s.

- Single: Single individuals are more evenly spread across ages, with peaks at 23, 24, and 25.

15) Education by Marital Status

- Partnered: Most partnered individuals have 16 years of education, followed by 14 years.
- Single: Single individuals also have a majority with 16 years of education, but there is a noticeable number with 14 years.

17) Usage by Marital Status

- Partnered: Partnered individuals have a peak usage level of 3, followed by 4.
- Single: Single individuals also show a peak at usage level 3, with a significant number at level 4.

18) Income by Marital Status

- Partnered: Partnered individuals have peaks at $45,480 and$53,439, with a more even distribution across income levels.
- Single: Single individuals show peaks at $45,480 and$50,028, with fewer individuals in higher income brackets.

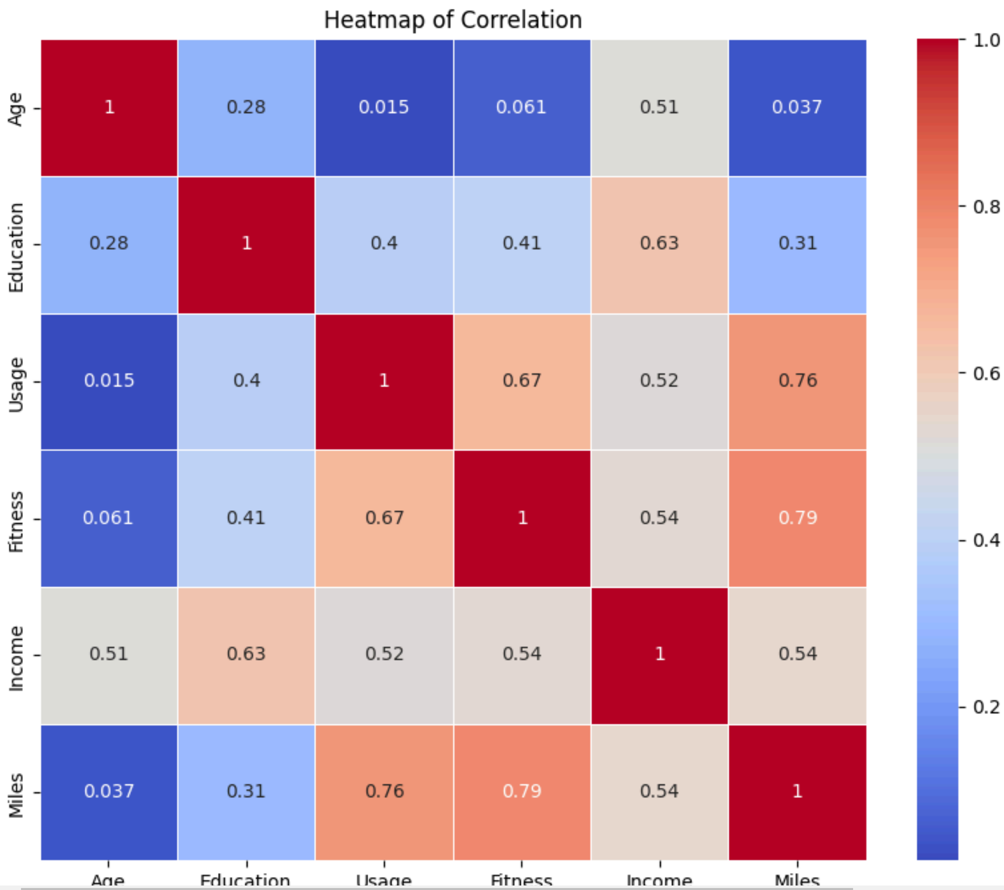
19) Fitness by Marital Status

- Partnered: Most partnered individuals have a fitness level of 3, followed by 5.
- Single: Single individuals are more evenly distributed across fitness levels, with peaks at 3 and 5.

20) Miles by Marital Status

- Partnered: Partnered individuals have peaks at 85 and 95 miles, with a significant number in higher mileage categories.
- Single: Single individuals show a more even distribution across mileage, with peaks at 85, 75, and 200 miles.

```
# Select only the numerical columns for correlation
numerical_columns = df.select_dtypes(include=['float64', 'int64']).columns
# Calculate the correlation matrix
corr_matrix = df[numerical_columns].corr()
# Plot the heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Heatmap of Correlation')
plt.show()
```



Insights:

1. Strong Correlations:

- The heatmap reveals strong positive correlations between Fitness and Miles, indicating that higher fitness is associated with more miles covered.
- The heatmap reveals strong positive correlations between Usage and Miles, indicating that higher usage is associated with more miles covered.

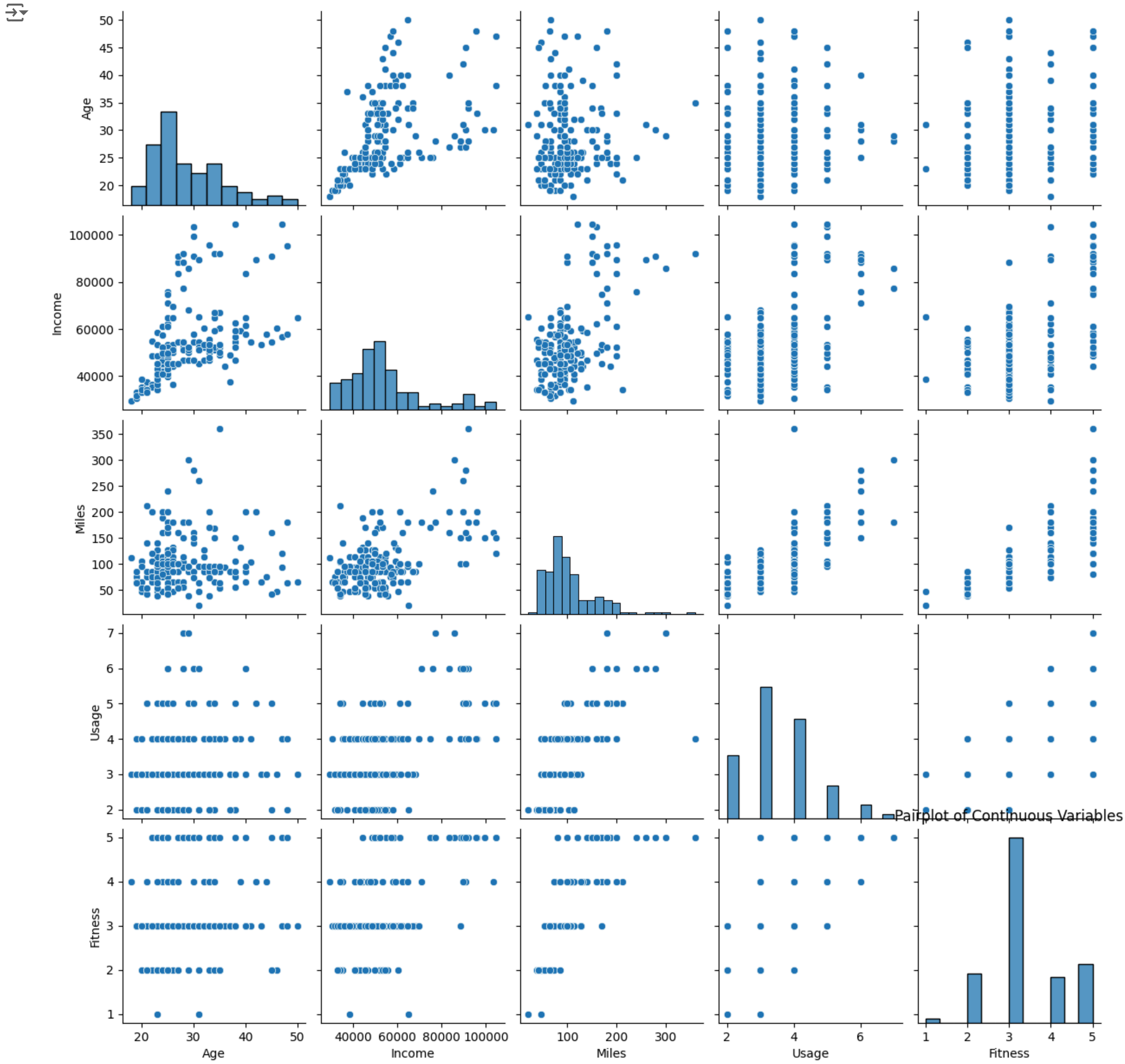
2. Moderate Correlations:

- There are moderate positive correlations between Age and Income, suggesting that older users tend to have higher incomes.
- There are moderate positive correlations between Education and Income, suggesting that highly educated person tend to have higher incomes.
- There are moderate positive correlations between Usage and Income, Usage and fitness.
- There are moderate positive correlations between fitness and Income.
- There are moderate positive correlations between Miles and Income, Education and fitness.

3. Weak Correlations:

- Other variables like Age and Fitness show weak correlations with Age, and Miles, Age and Usage , Age and Education.

```
# Plot the pairplot
sns.pairplot(df[['Age', 'Income', 'Miles', 'Usage', 'Fitness']])
plt.title('Pairplot of Continuous Variables')
plt.show()
```



🔍 Insights:

1. Scatter Plots:

- The scatter plots in the pairplot provide a visual representation of the relationships between continuous variables. For example, the scatter plot between Fitness and Miles shows a positive trend, indicating that higher fitness levels are associated with more miles covered.
- Similarly, there is a positive trend between Usage and Miles, supporting the strong correlation observed in the heatmap.

2. Histograms:

- The histograms on the diagonal of the pairplot show the distribution of each continuous variable. These histograms help in understanding the spread and central tendency of the data for variables like Age, Income, Miles, Usage, and Fitness.

3. Outliers:

- The pairplot helps identify outliers in the data. For example, there are a few outliers in the Miles and Income variables, where some users have covered significantly more miles or have higher incomes than the majority.
- These outliers can be seen as points that are distant from the main cluster of data points.
- Outliers detail observation is presented below.

Business Insights based on Non-Graphical and Visual Analysis

```
# Comments on the range of attributes
print("Comments on the range of attributes:")
for column in df.columns:
    if pd.api.types.is_categorical_dtype(df[column]):
        df[column] = df[column].astype('category').cat.as_ordered()
    print(f"{column}: {df[column].min()} to {df[column].max()}")
```

```
Comments on the range of attributes:
Product: KP281 to KP781
Age: 18 to 50
Gender: Female to Male
Education: 12 to 21
MaritalStatus: Partnered to Single
Usage: 2 to 7
Fitness: 1 to 5
Income: 29562 to 104581
Miles: 21 to 360
```

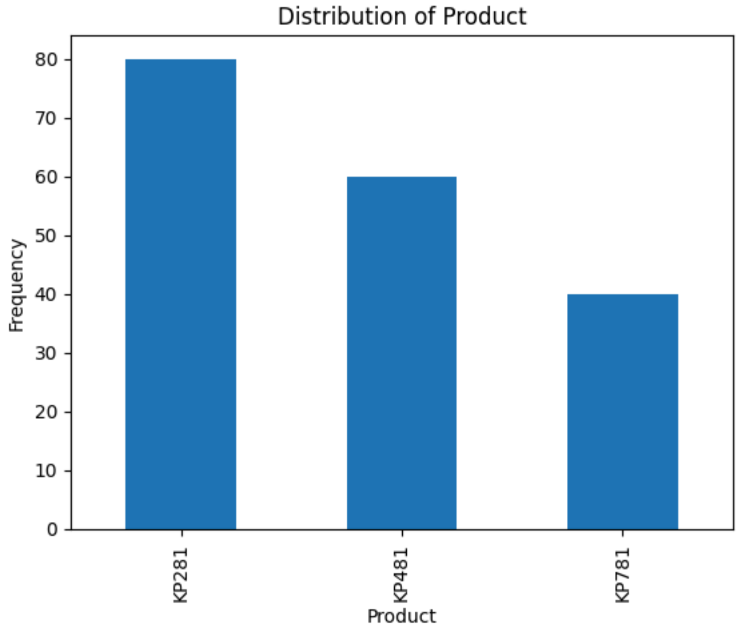
Insights:

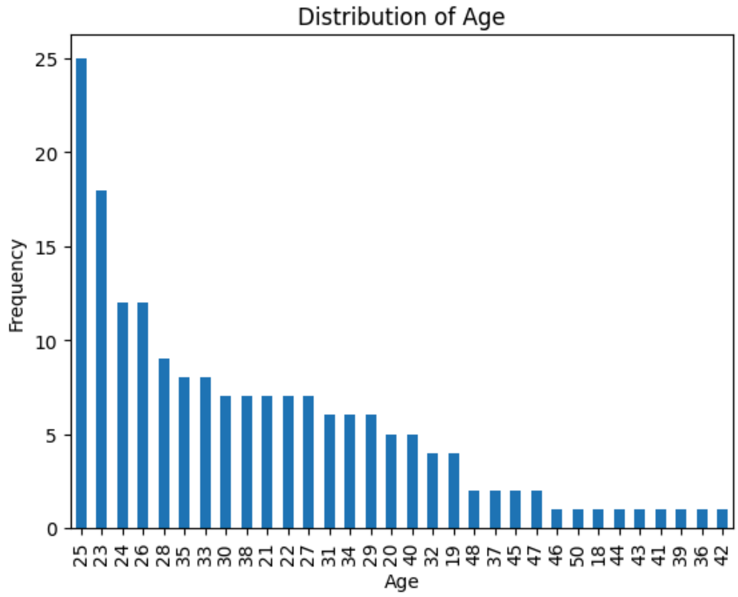
- Product: Ranges from KP281 to KP781, indicating different product categories or models.
- Age: Spans from 18 to 50, covering a broad spectrum from young adults to middle-aged individuals.
- Gender: Includes both Female and Male.
- Education: Ranges from 12 to 21, suggesting varying levels of educational attainment.
- Marital Status: Includes Partnered and Single, indicating different relationship statuses.
- Usage: Ranges from 2 to 7, showing different levels of product engagement.
- Fitness: Ranges from 1 to 5, indicating varying fitness levels.
- Income: Spans from 29562 to 104581, reflecting a wide range of economic backgrounds.
- Miles: Ranges from 21 to 360, indicating the distance covered by users.

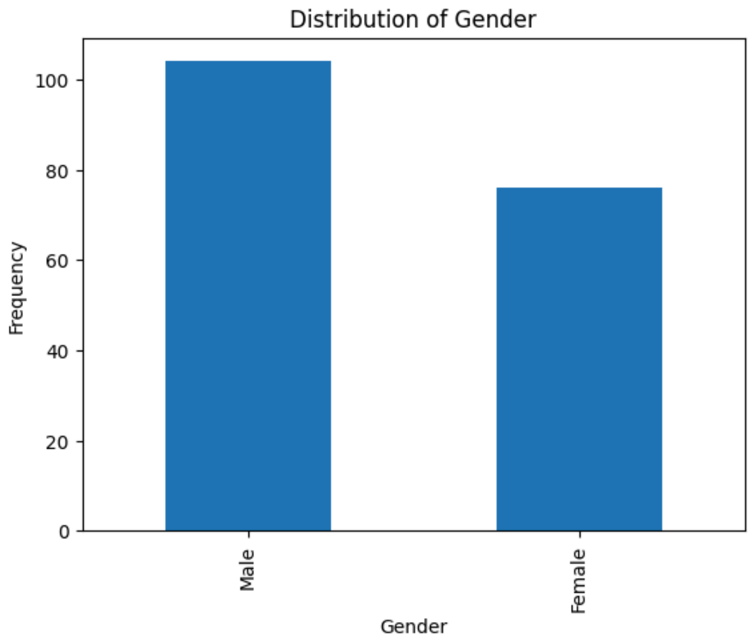
```
# Comments on the distribution of the variables and relationship between them
def comments_on_distribution(df):
    comments = []

    for column in df.columns:
        comments.append(f"{column}:")
        if pd.api.types.is_numeric_dtype(df[column]):
            comments.append(f" - Distribution: The {column} distribution spans from {df[column].min()} to {df[column].max()}.")
        else:
            comments.append(f" - Distribution: The {column} distribution includes categories: {df[column].unique().tolist()}.")

        # Relationship with other variables
        comments.append(" - Relationship with Other Variables:")
        for other_column in df.columns:
            if column != other_column:
                comments.append(f"   - {other_column}: Relationship analysis between {column} and {other_column}.")

        comments.append("\n")

    return "\n".join(comments)

# Print the comments on distribution and relationships
print(comments_on_distribution(df))
```

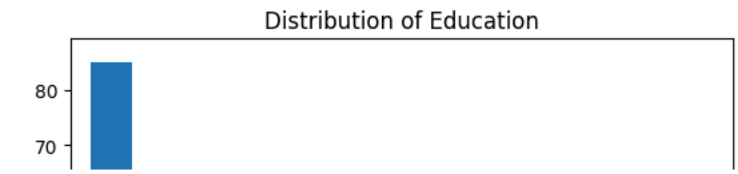
```
    - Age: Relationship analysis between Usage and Age.
    - Gender: Relationship analysis between Usage and Gender.
    - Education: Relationship analysis between Usage and Education.
    - MaritalStatus: Relationship analysis between Usage and MaritalStatus.
    - Fitness: Relationship analysis between Usage and Fitness.
    - Income: Relationship analysis between Usage and Income.
    - Miles: Relationship analysis between Usage and Miles.


 Fitness:
  - Distribution: The Fitness distribution spans from 1 to 5.
  - Relationship with Other Variables:
    - Product: Relationship analysis between Fitness and Product.
    - Age: Relationship analysis between Fitness and Age.
    - Gender: Relationship analysis between Fitness and Gender.
    - Education: Relationship analysis between Fitness and Education.
    - MaritalStatus: Relationship analysis between Fitness and MaritalStatus.
    - Usage: Relationship analysis between Fitness and Usage.
    - Income: Relationship analysis between Fitness and Income.
    - Miles: Relationship analysis between Fitness and Miles.


 Income:
  - Distribution: The Income distribution spans from 29562 to 104581.
  - Relationship with Other Variables:
    - Product: Relationship analysis between Income and Product.
    - Age: Relationship analysis between Income and Age.
    - Gender: Relationship analysis between Income and Gender.
    - Education: Relationship analysis between Income and Education.
    - MaritalStatus: Relationship analysis between Income and MaritalStatus.
    - Usage: Relationship analysis between Income and Usage.
    - Fitness: Relationship analysis between Income and Fitness.
    - Miles: Relationship analysis between Income and Miles.


 Miles:
  - Distribution: The Miles distribution spans from 21 to 360.
  - Relationship with Other Variables:
    - Product: Relationship analysis between Miles and Product.
    - Age: Relationship analysis between Miles and Age.
    - Gender: Relationship analysis between Miles and Gender.
    - Education: Relationship analysis between Miles and Education.
    - MaritalStatus: Relationship analysis between Miles and MaritalStatus.
    - Usage: Relationship analysis between Miles and Usage.
    - Fitness: Relationship analysis between Miles and Fitness.
    - Income: Relationship analysis between Miles and Income.
```

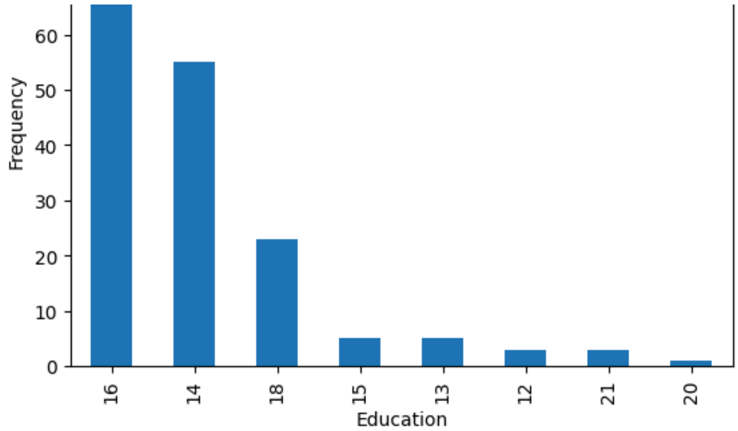
```
# Univariate Analysis
def plot_univariate(column):
    df[column].value_counts().plot(kind='bar')
    plt.title(f'Distribution of {column}')
    plt.xlabel(column)
    plt.ylabel('Frequency')
    plt.show()

# Plotting Univariate Distributions
for column in df.columns:
    plot_univariate(column)
```

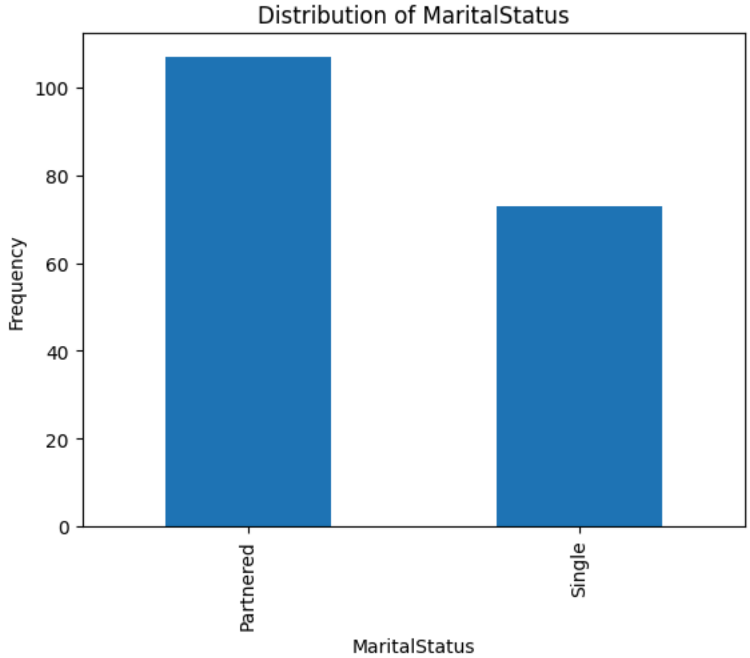

Distribution of Product
Frequency
Product
KP281
KP481
KP781

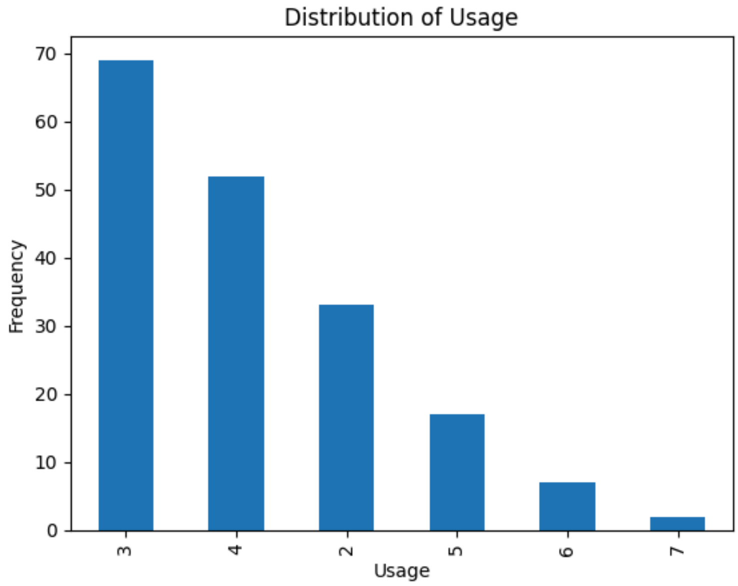

Distribution of Age
Frequency
Age

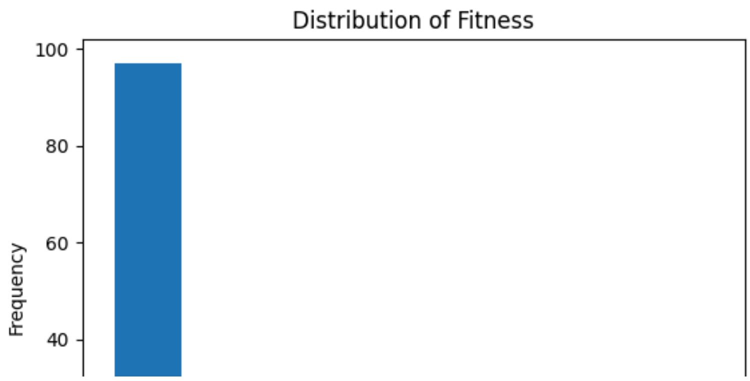

Distribution of Gender
Frequency
Gender
Male
Female

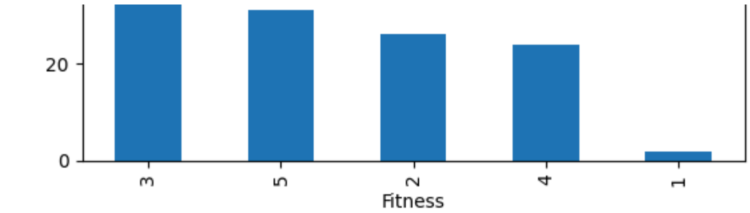

Distribution of Education


Frequency
60
50
40
30
20
10
0
16
14
18
15
13
12
21
20
Education

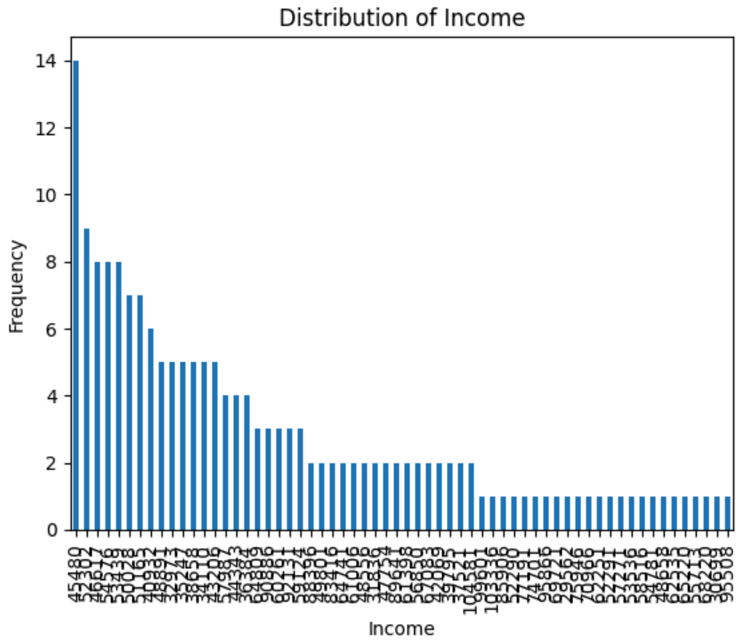

Distribution of MaritalStatus
Frequency
100
80
60
40
20
0
Partnered
Single
MaritalStatus


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

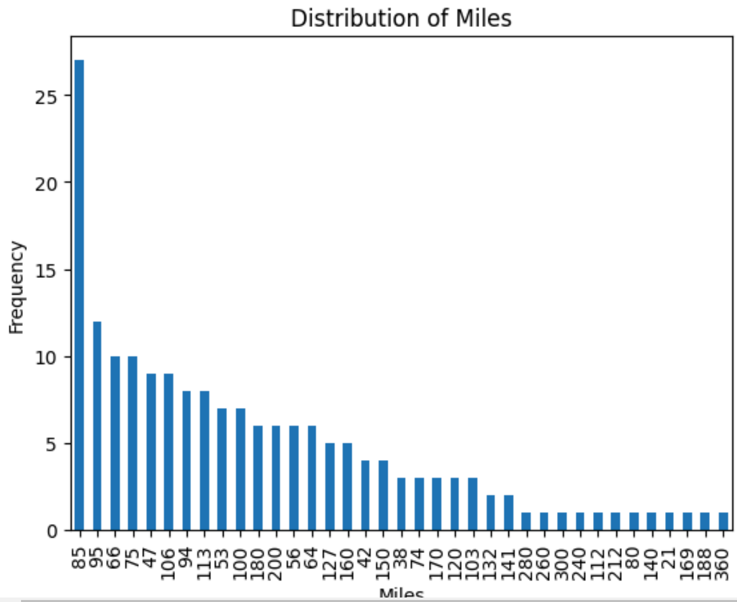

Distribution of Fitness
Frequency
100
80
60
40



20
0
3
5
2
4
1
Fitness

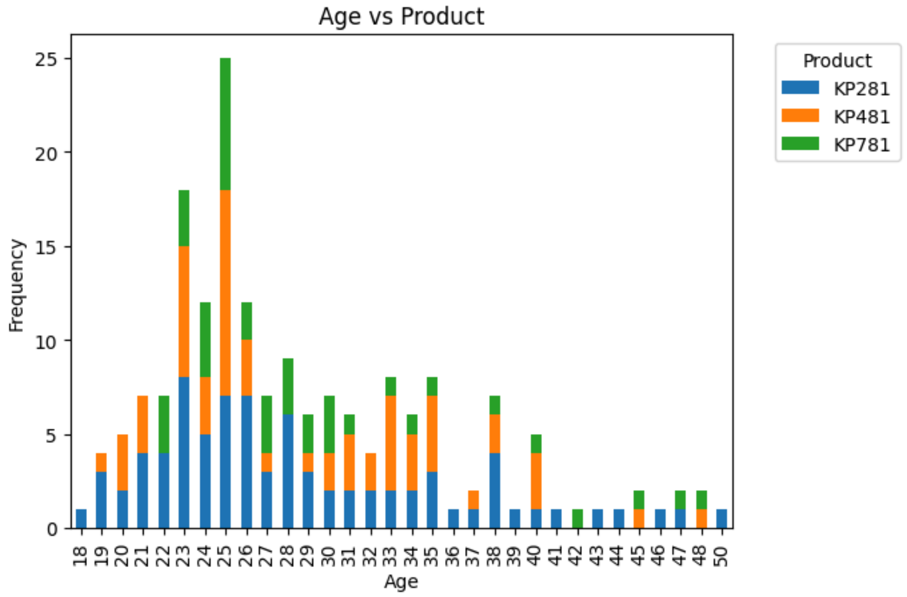


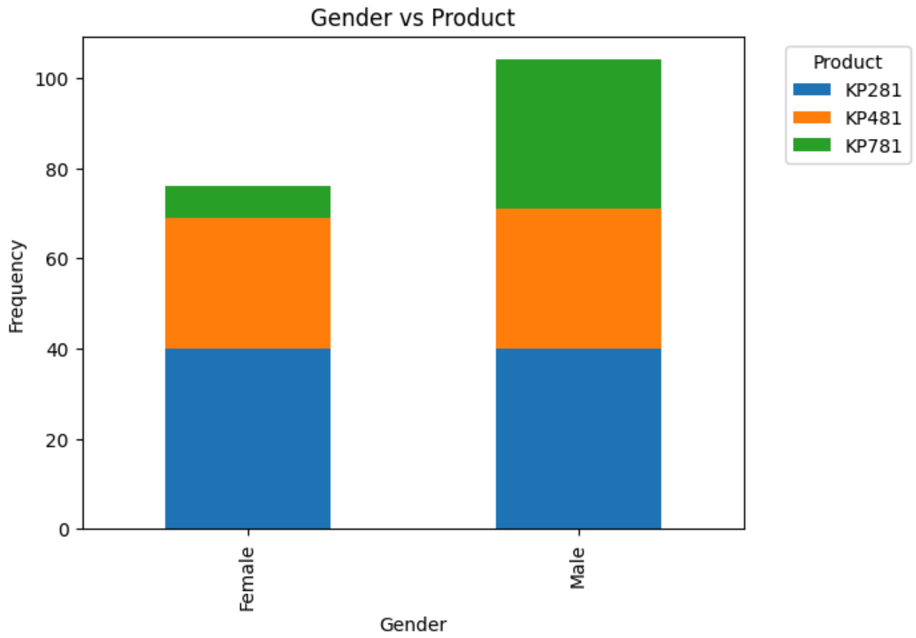
Distribution of Income
Frequency
Income

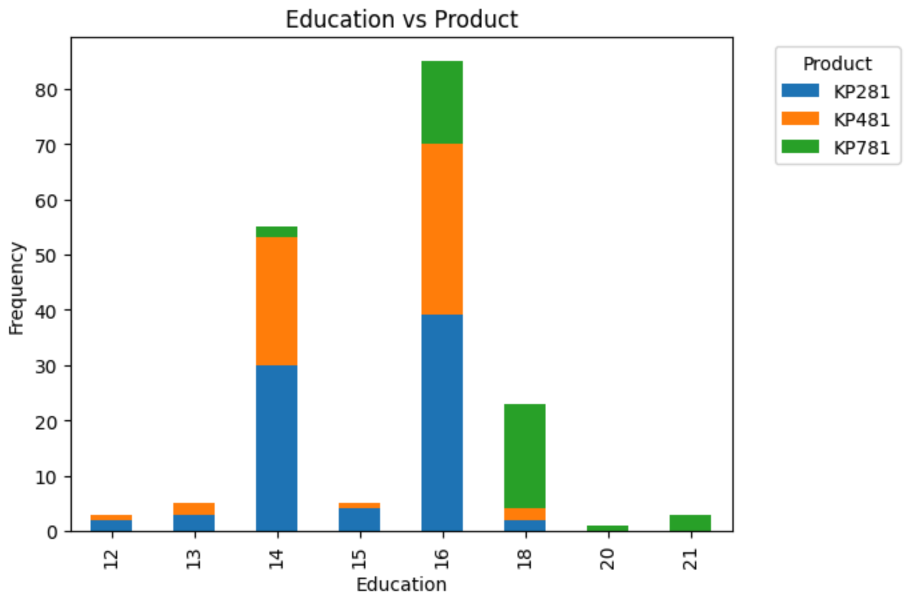


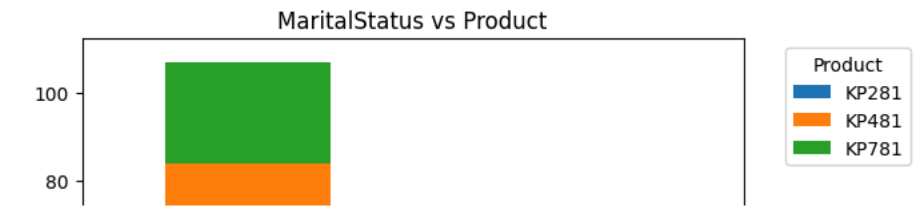
Distribution of Miles
Frequency
85
95
66
75
47
106
94
113
53
100
180
200
56
64
127
160
42
150
38
74
170
120
103
132
141
280
260
300
240
112
212
80
140
21
169
188
360
Miles

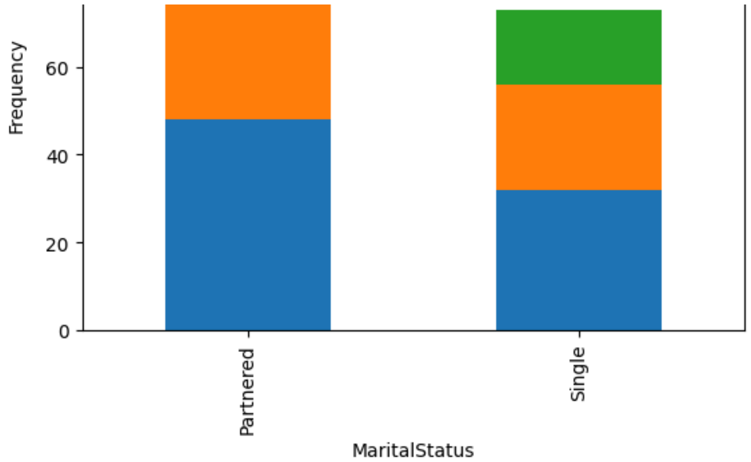
```python
# Bivariate Analysis
def plot_bivariate(x, y):
    ax = pd.crosstab(df[x], df[y]).plot(kind='bar', stacked=True)
    plt.title(f'{x} vs {y}')
    plt.xlabel(x)
    plt.ylabel('Frequency')
    plt.legend(title=y, bbox_to_anchor=(1.05, 1), loc='upper left')
    plt.show()

# Plotting Bivariate Distributions
for i in range(len(df.columns)):
    for j in range(i + 1, len(df.columns)):
        plot_bivariate(df.columns[j], df.columns[i])
```

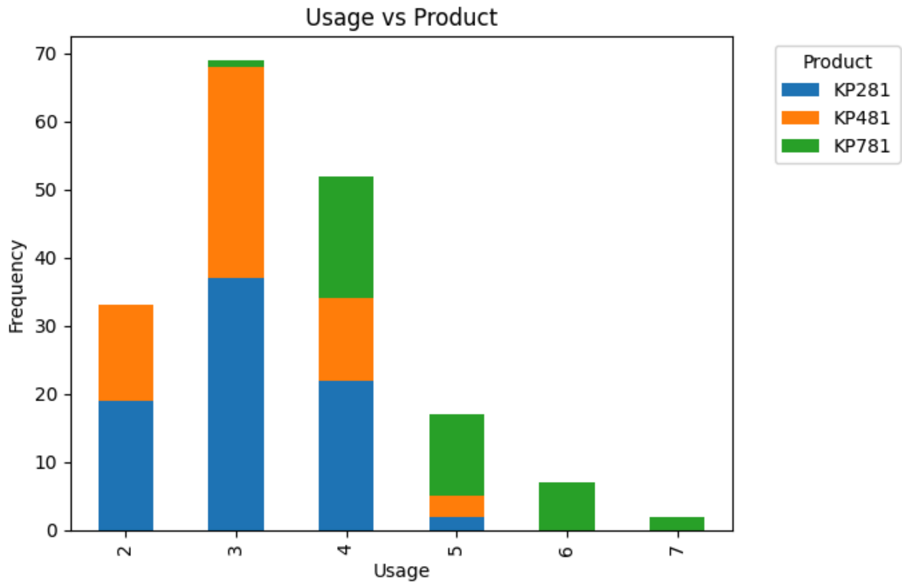

Age vs Product
Frequency
Age
Product
KP281
KP481
KP781

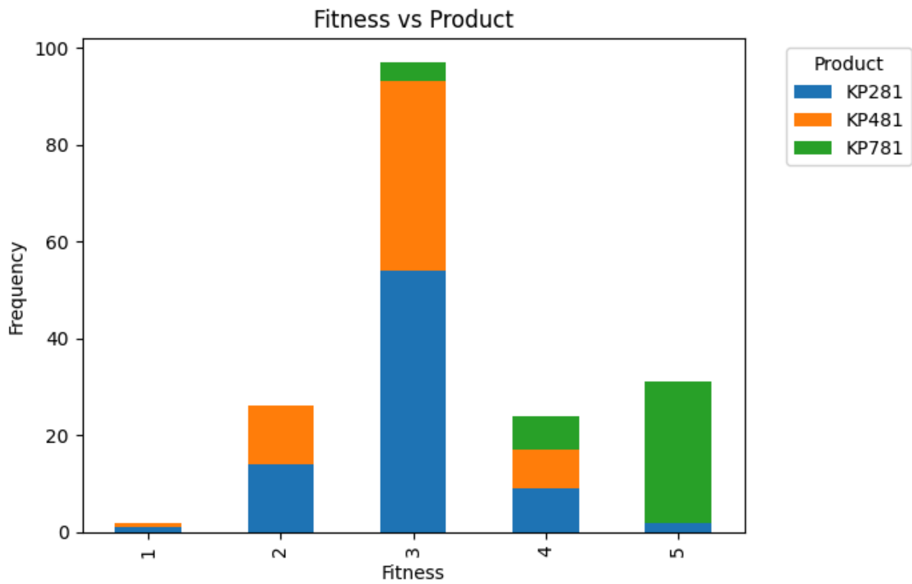

Gender vs Product
Frequency
Female
Male
Gender
Product
KP281
KP481
KP781

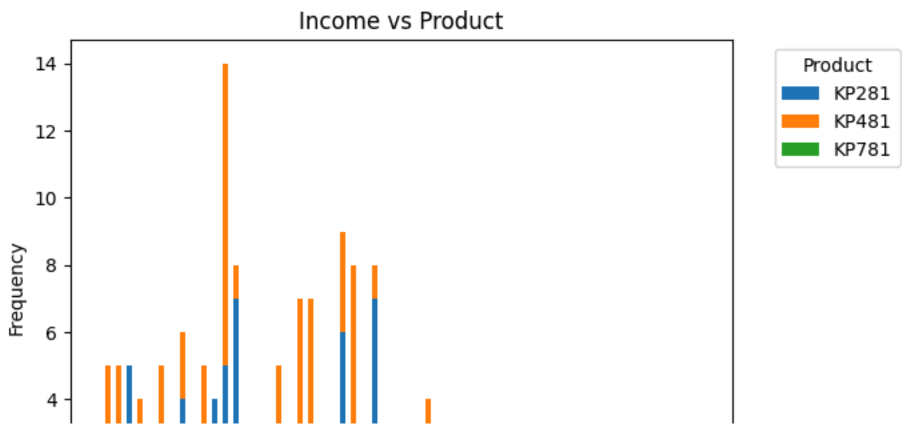

Education vs Product
Frequency
Education
Product
KP281
KP481
KP781

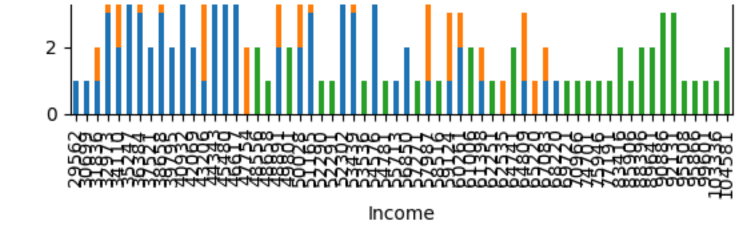

MaritalStatus vs Product
Product
KP281
KP481
KP781

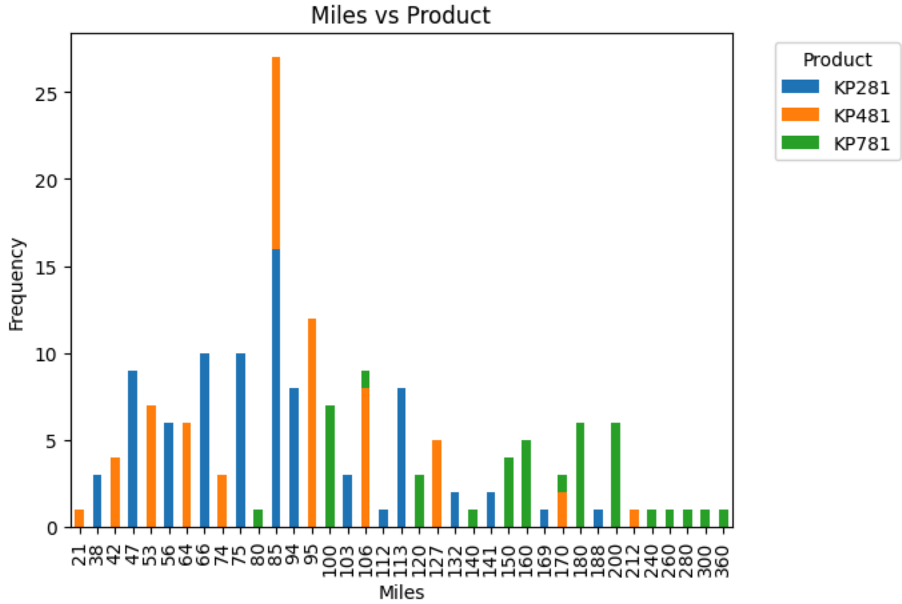

Frequency
60
40
20
0
Partnered
Single
MaritalStatus

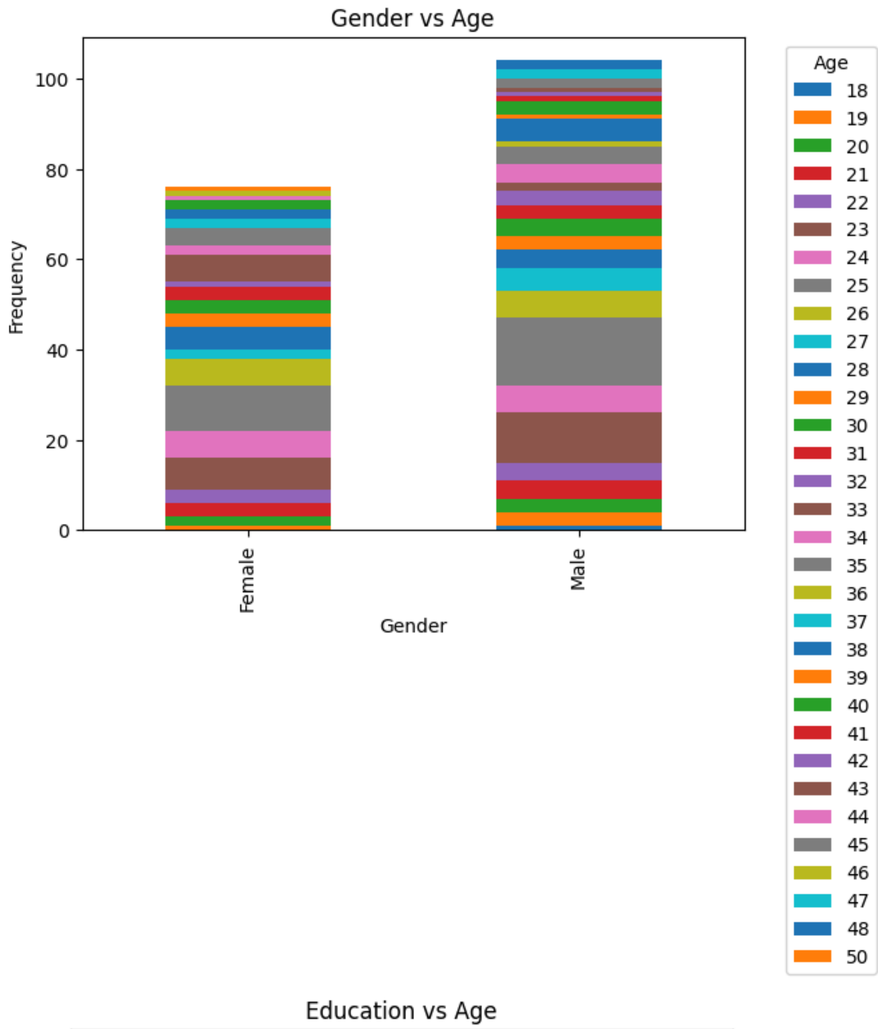

Usage vs Product
Frequency
Usage
Product
KP281
KP481
KP781

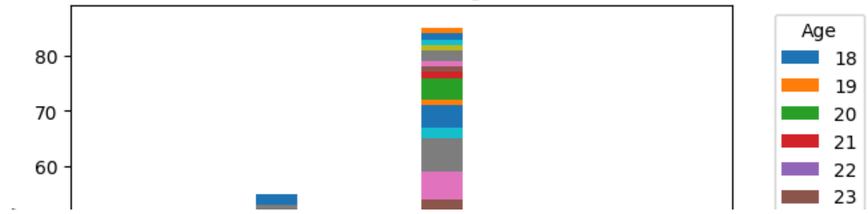

Fitness vs Product
Frequency
Fitness
Product
KP281
KP481
KP781

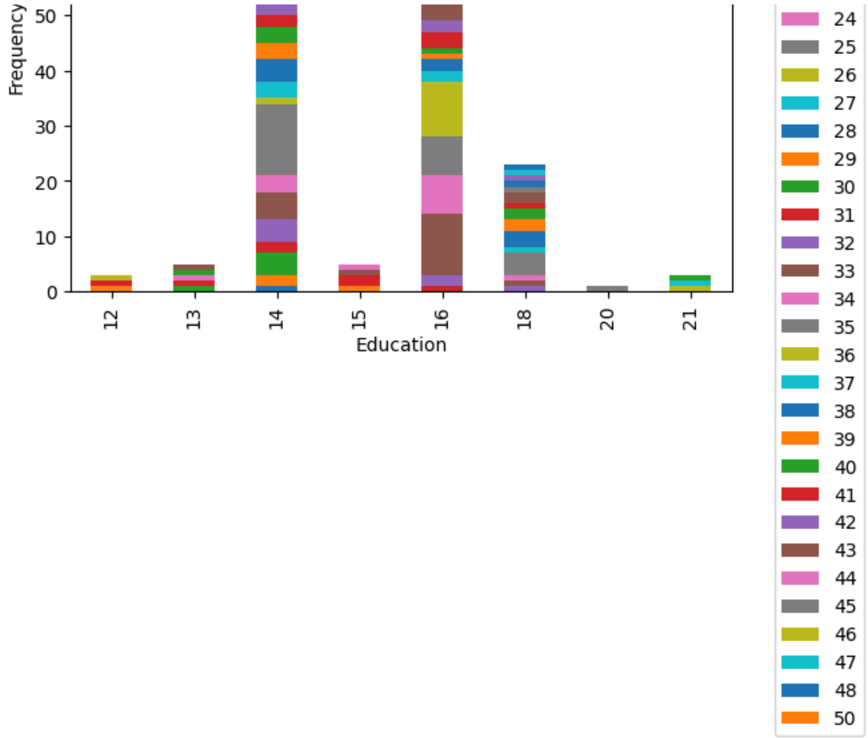

Income vs Product
Frequency
Product
KP281
KP481
KP781

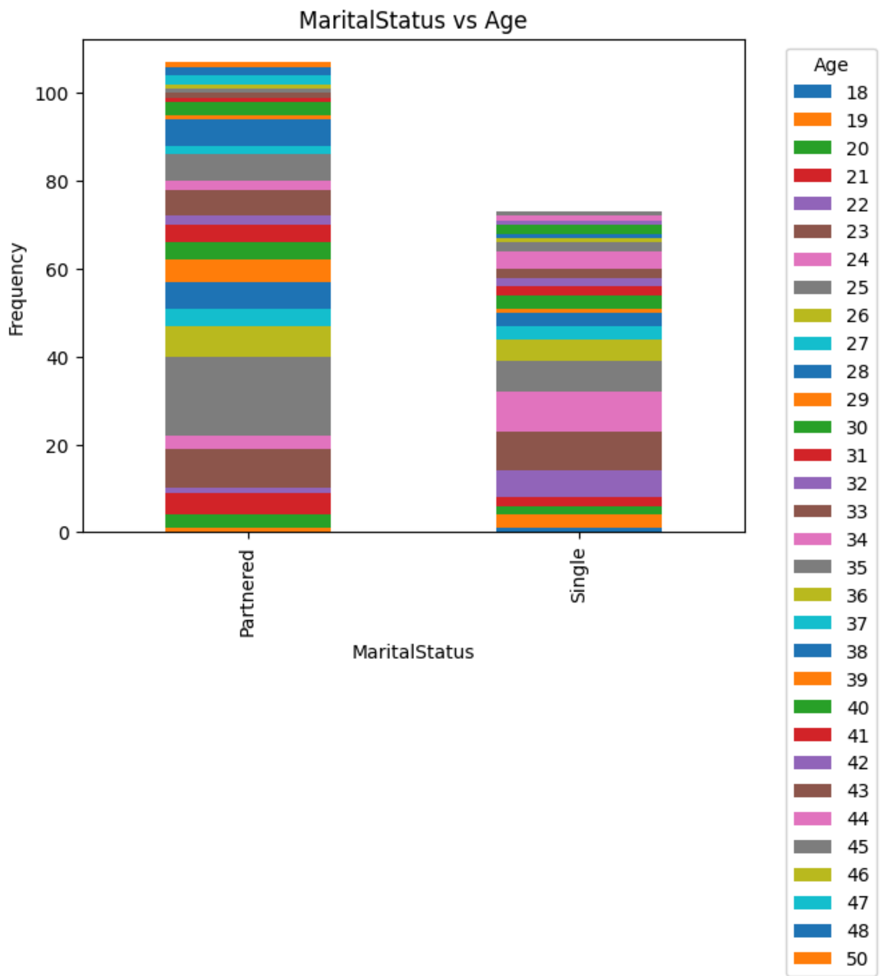

0
2
Income

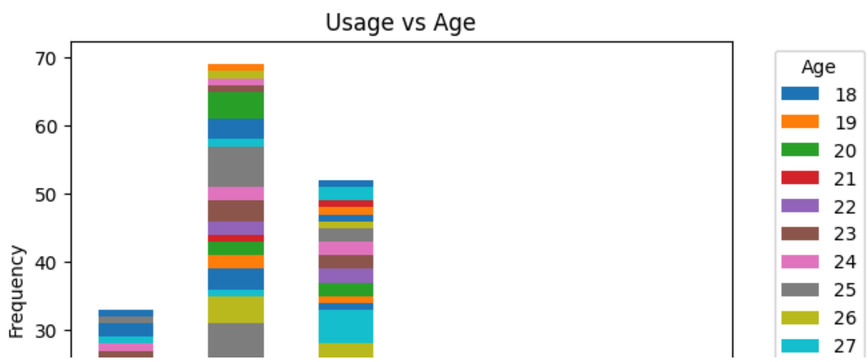

Miles vs Product
Frequency
Miles
Product
KP281
KP481
KP781
0
5
10
15
20
25
21
38
42
47
53
56
64
66
74
75
80
85
94
95
100
103
106
112
113
120
127
132
140
141
150
160
169
170
180
188
200
212
240
260
280
300
360

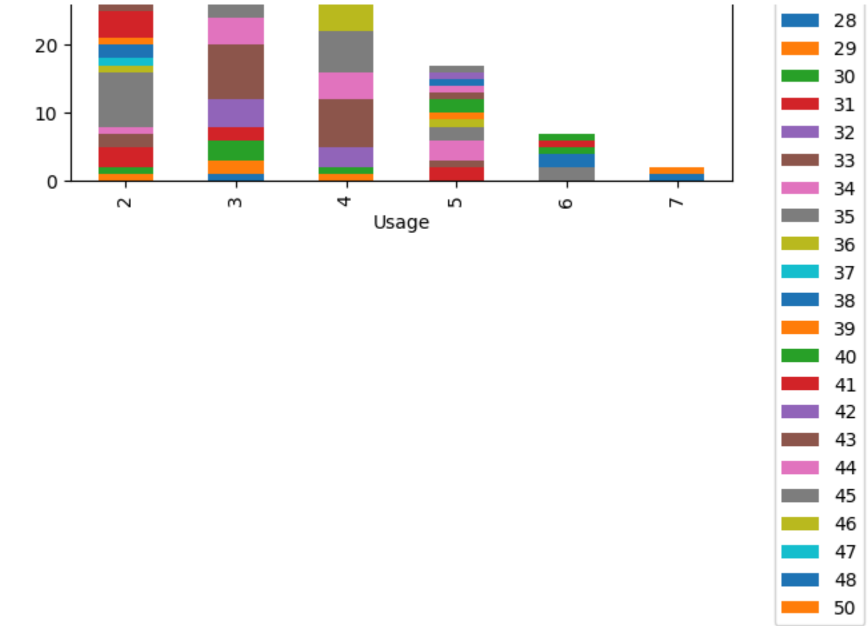

Gender vs Age
Frequency
Gender
Female
Male
0
20
40
60
80
100
Age
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
50

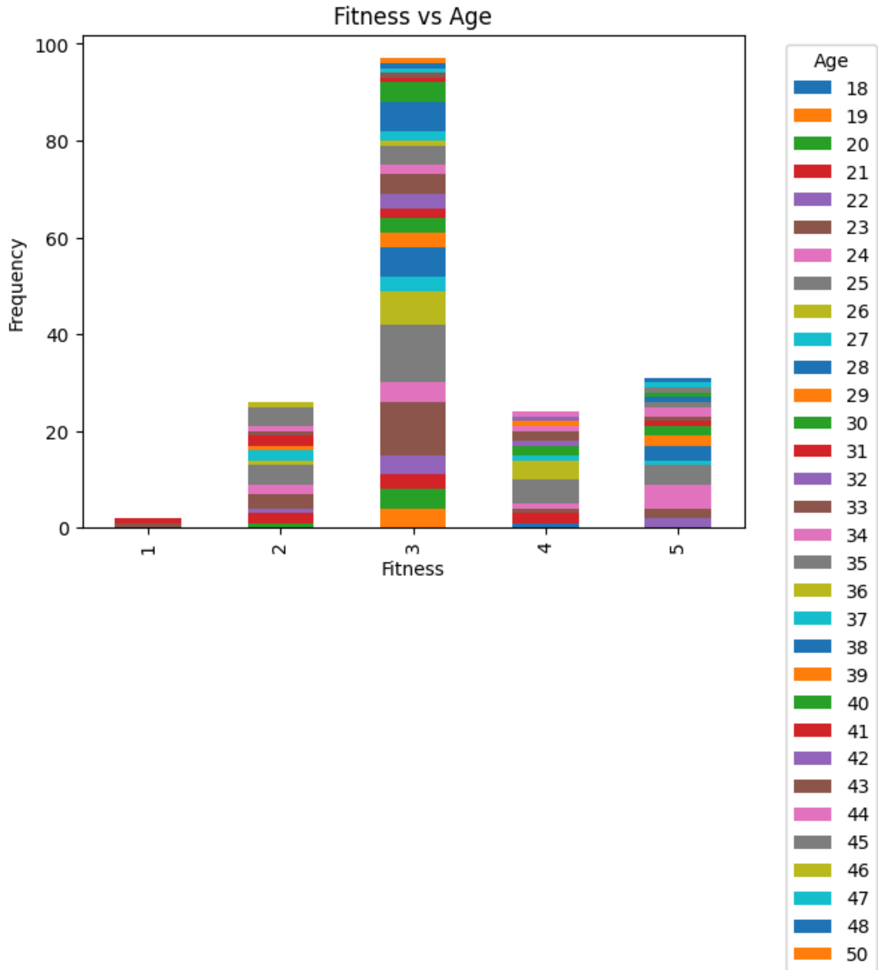

Education vs Age
60
70
80
Age
18
19
20
21
22
23

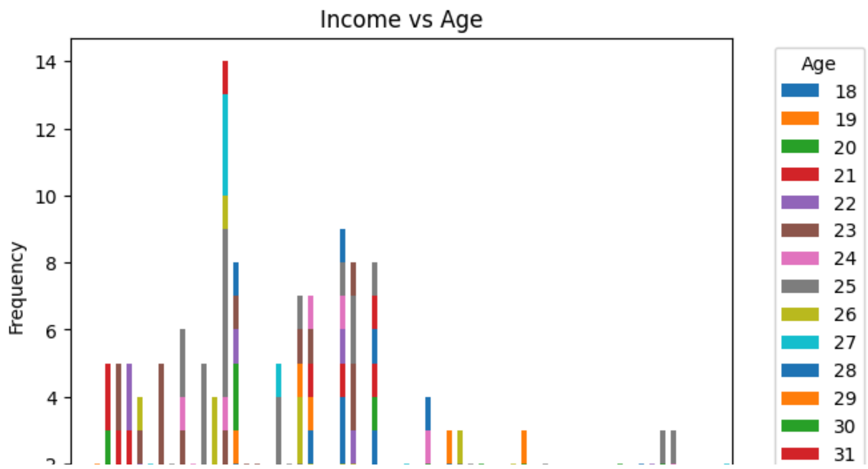

Frequency
50
40
30
20
10
0
12
13
14
15
16
18
20
21
Education
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
50

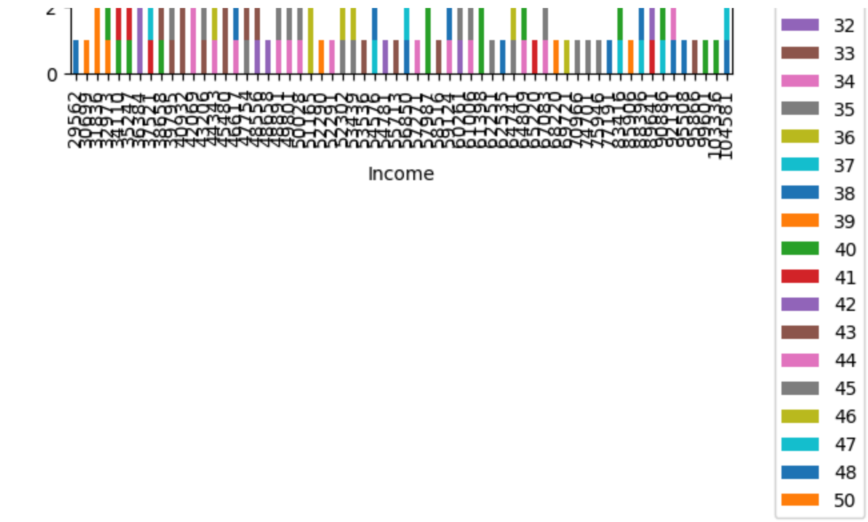

MaritalStatus vs Age
Frequency
100
80
60
40
20
0
Partnered
Single
MaritalStatus
Age
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
50

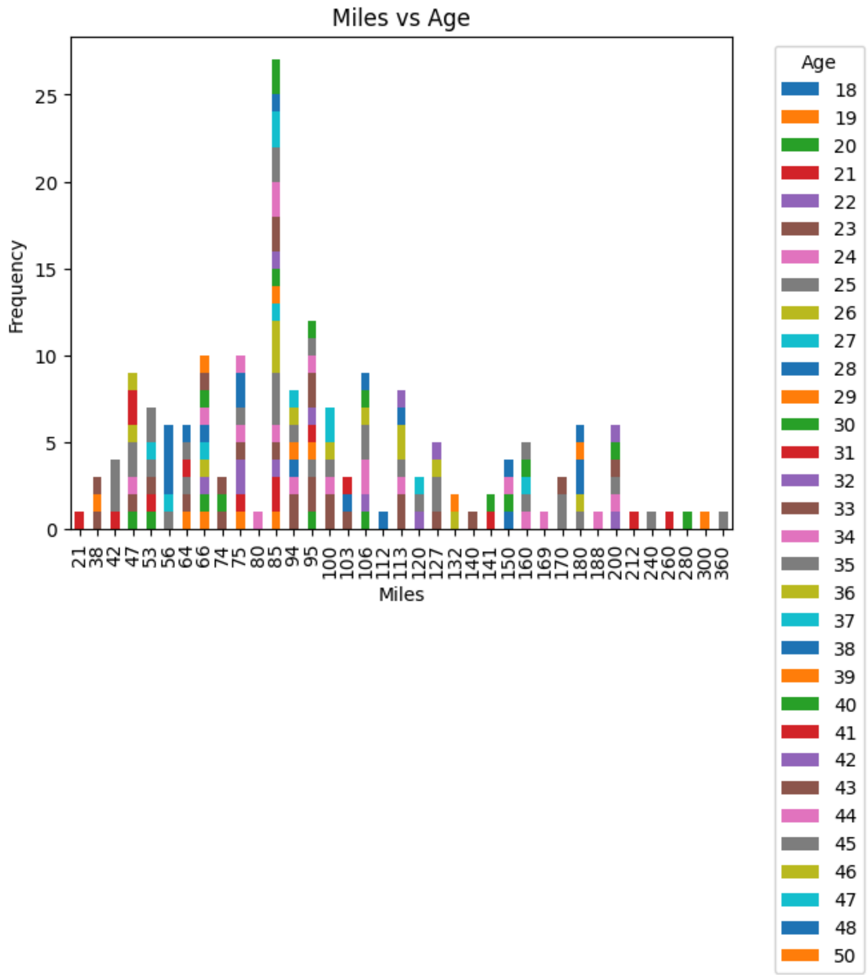

Usage vs Age
Frequency
70
60
50
40
30
Age
18
19
20
21
22
23
24
25
26
27

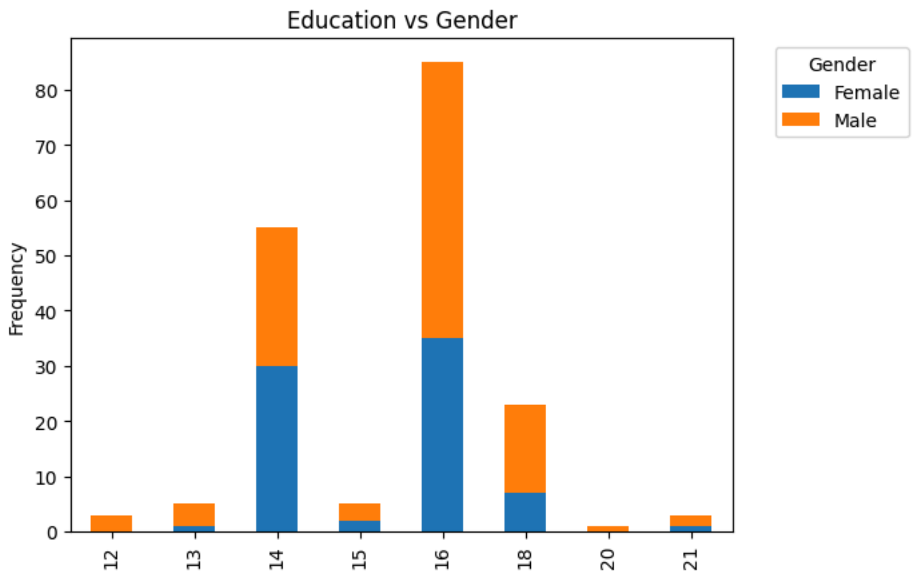

0
10
20
2
3
4
5
6
7
Usage
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
50


Fitness vs Age
Frequency
0
20
40
60
80
100
1
2
3
4
5
Fitness
Age
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
50


Income vs Age
Frequency
4
6
8
10
12
14
Age
18
19
20
21
22
23
24
25
26
27
28
29
30
31

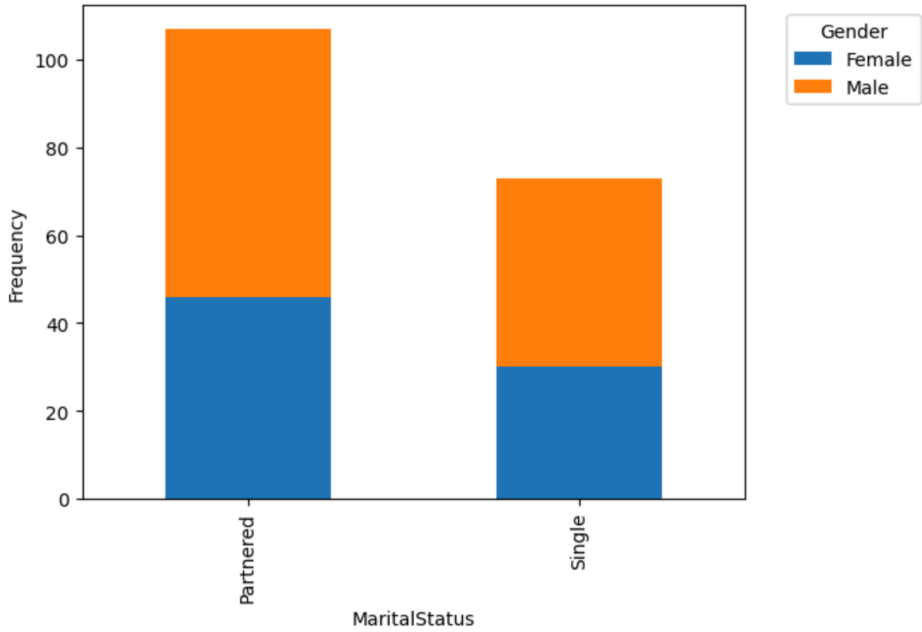




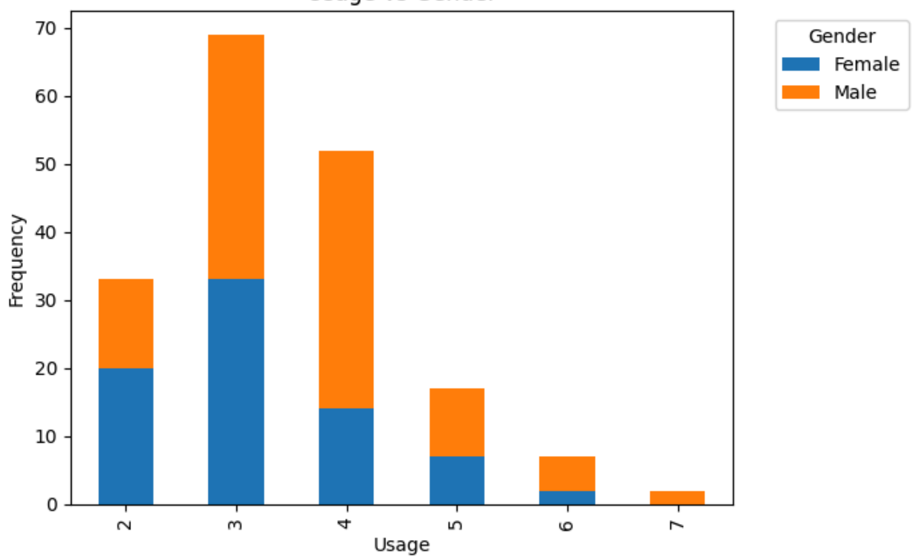


Education

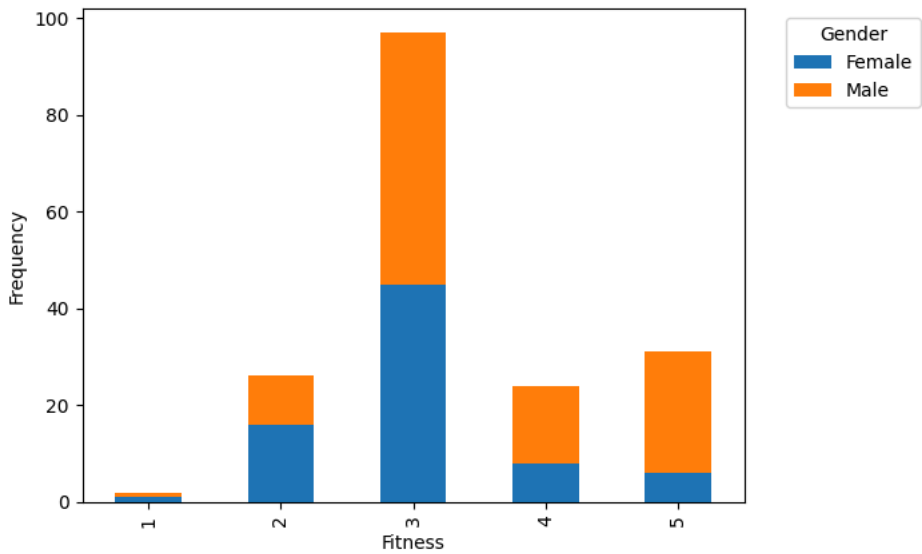
MaritalStatus vs Gender



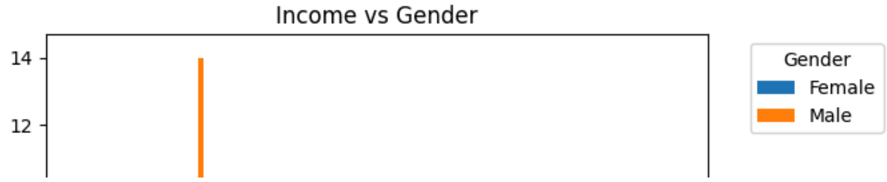
Usage vs Gender

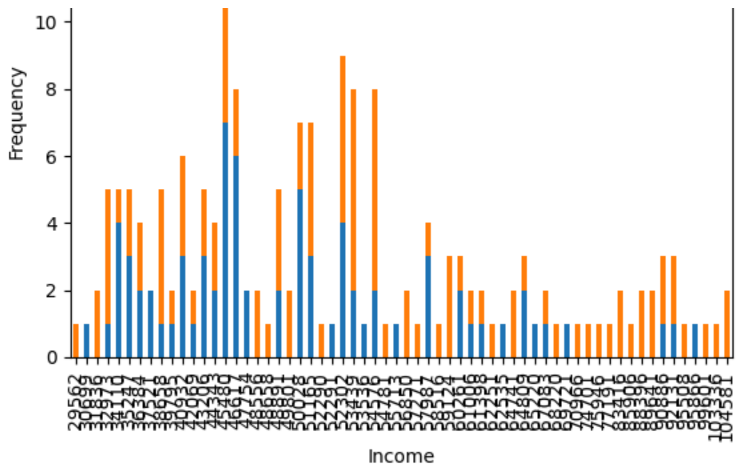


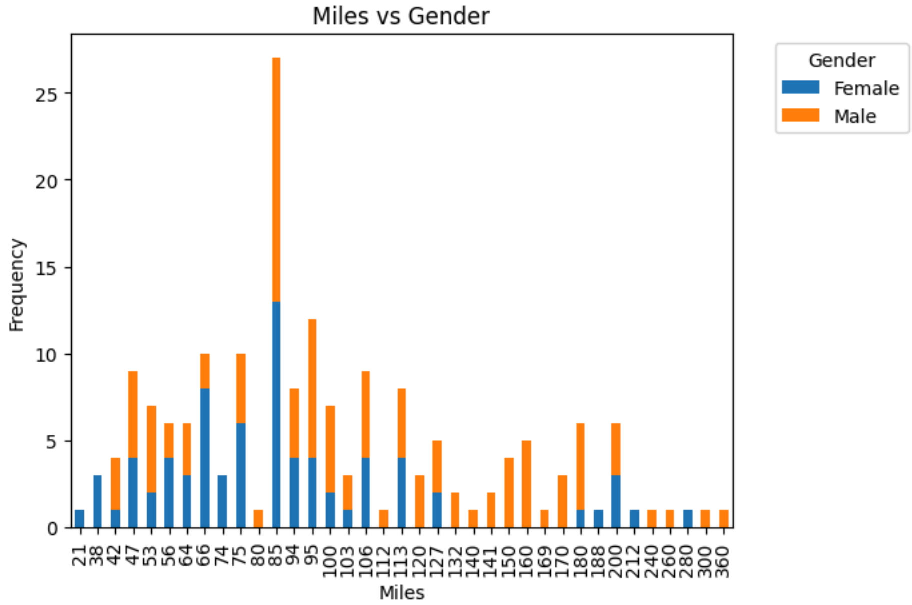
Fitness vs Gender

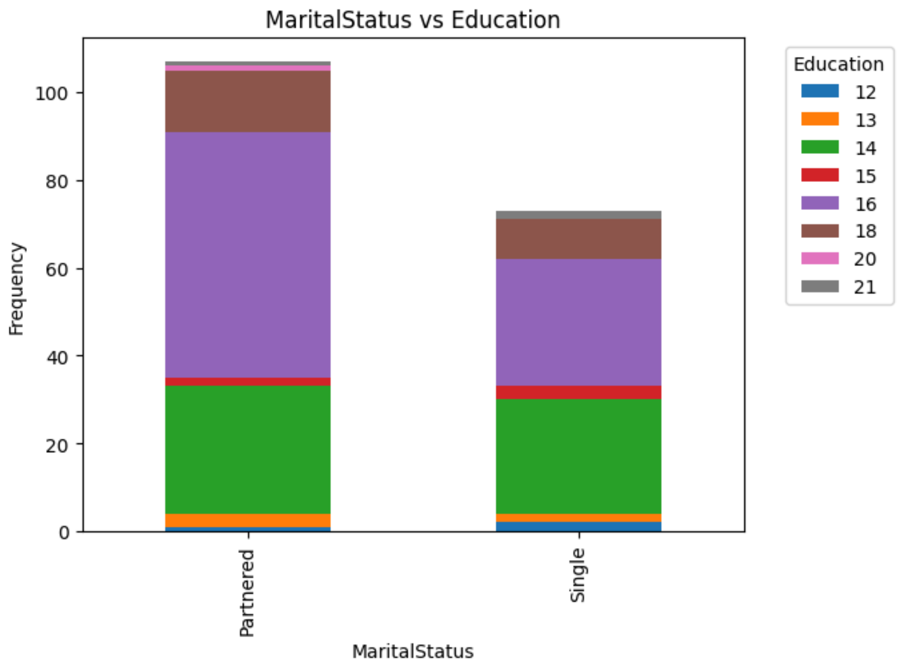


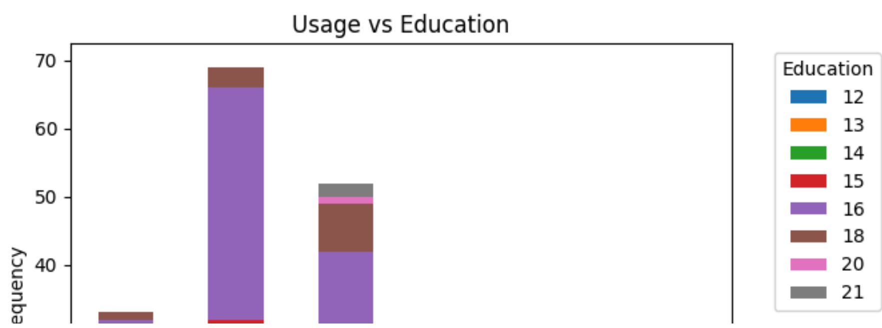
Income vs Gender

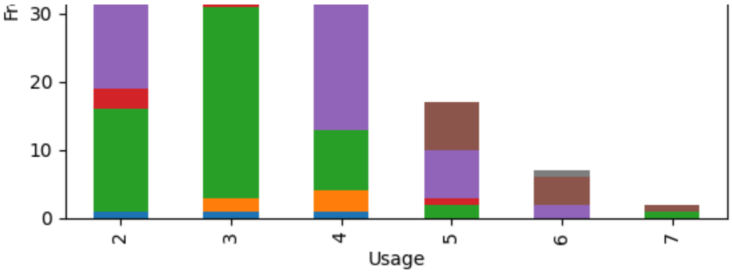



Frequency
Income
0
2
4
6
8
10

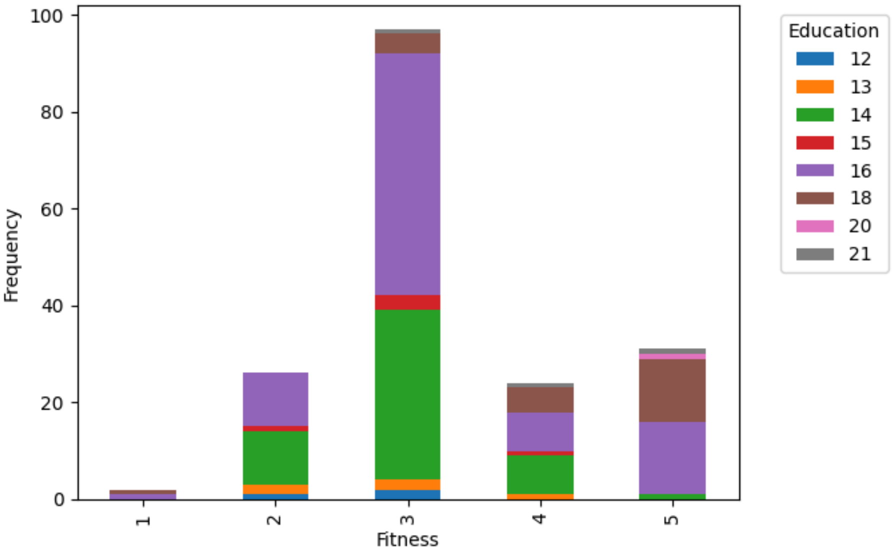

Miles vs Gender
Gender
Female
Male
Frequency
Miles
0
5
10
15
20
25
21
38
42
47
53
56
64
66
74
75
80
85
94
95
100
103
106
112
113
120
127
132
140
141
150
160
169
170
180
188
200
212
240
260
280
300
360


MaritalStatus vs Education
Education
12
13
14
15
16
18
20
21
Frequency
0
20
40
60
80
100
Partnered
Single
MaritalStatus

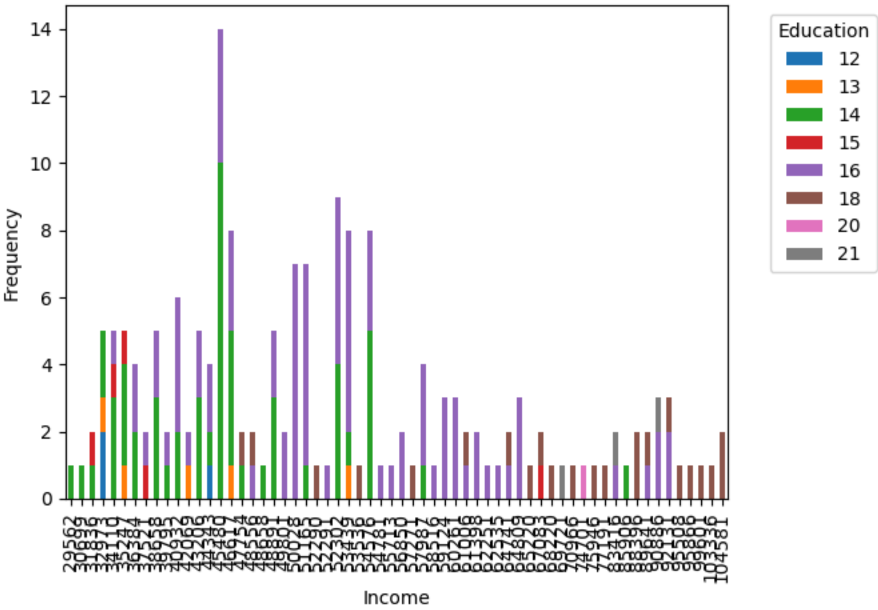

Usage vs Education
Education
12
13
14
15
16
18
20
21
40
50
60
70



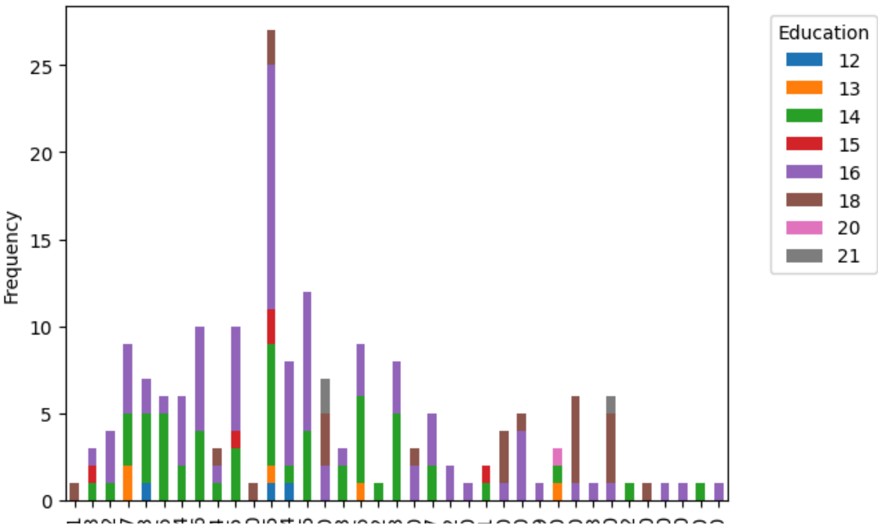
Fitness vs Education



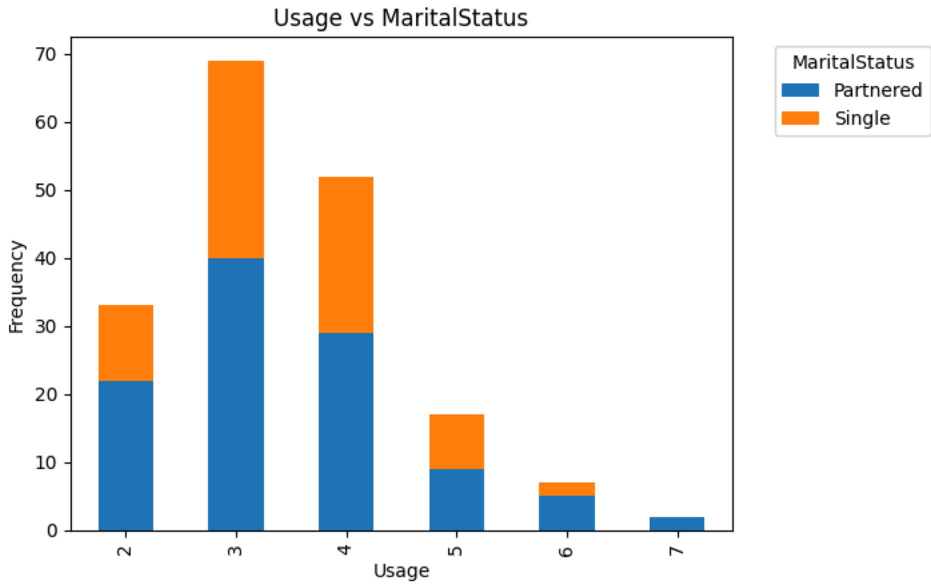
Income vs Education

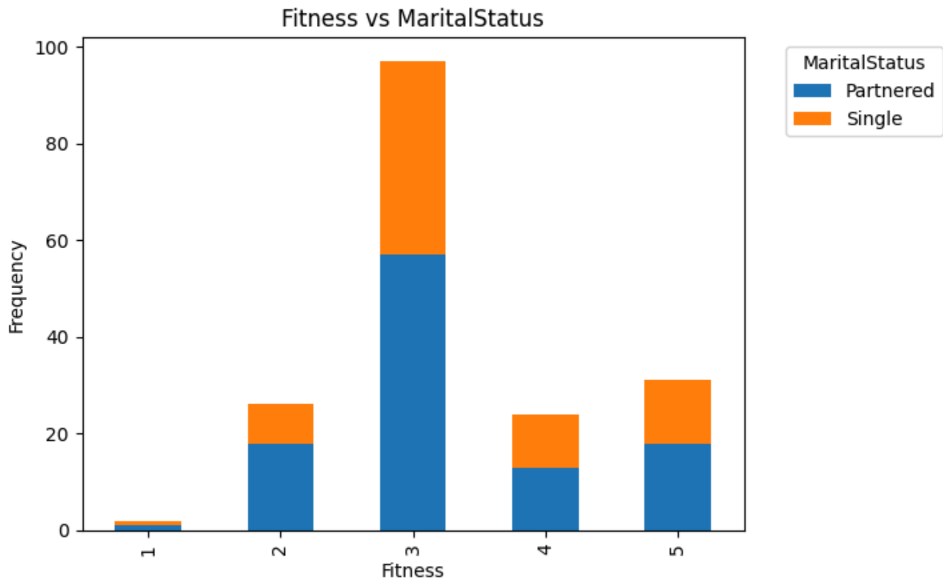


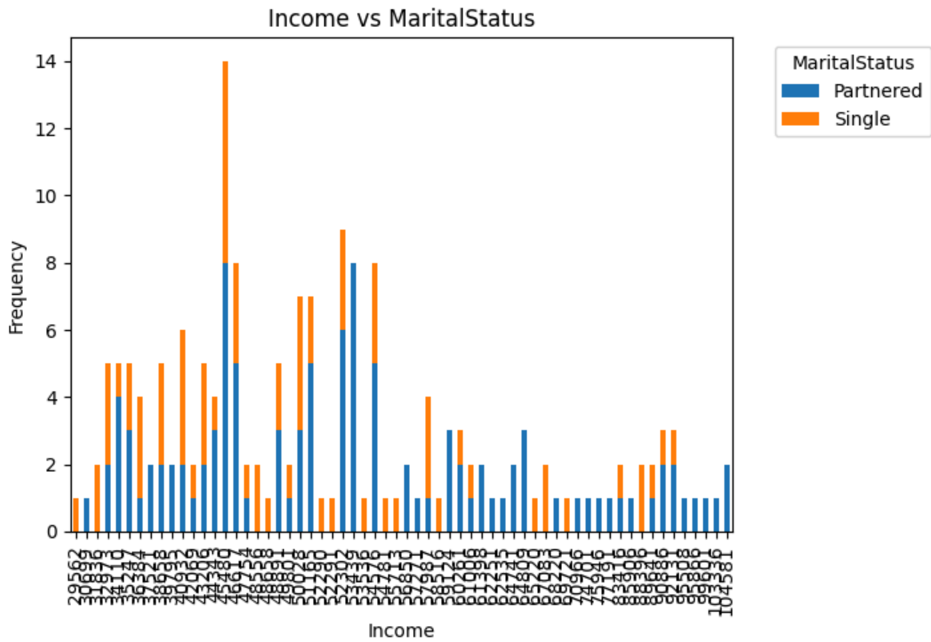
Miles vs Education

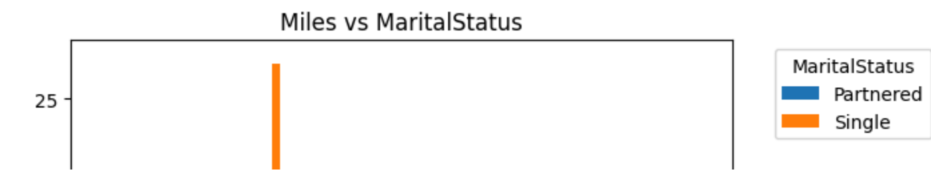


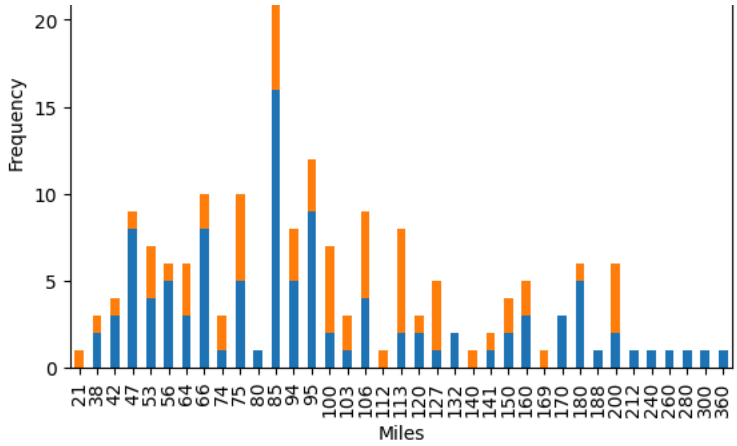

Miles

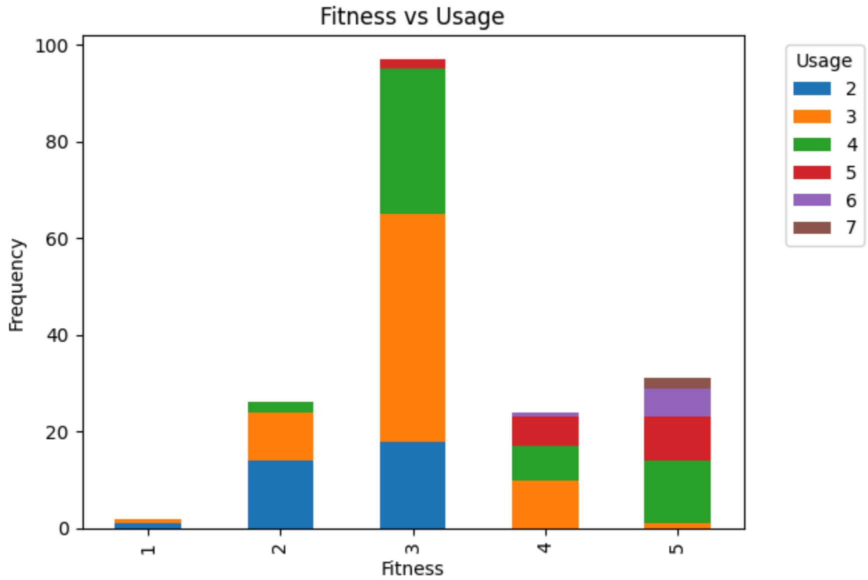

Usage vs MaritalStatus
Frequency
Usage
MaritalStatus
Partnered
Single

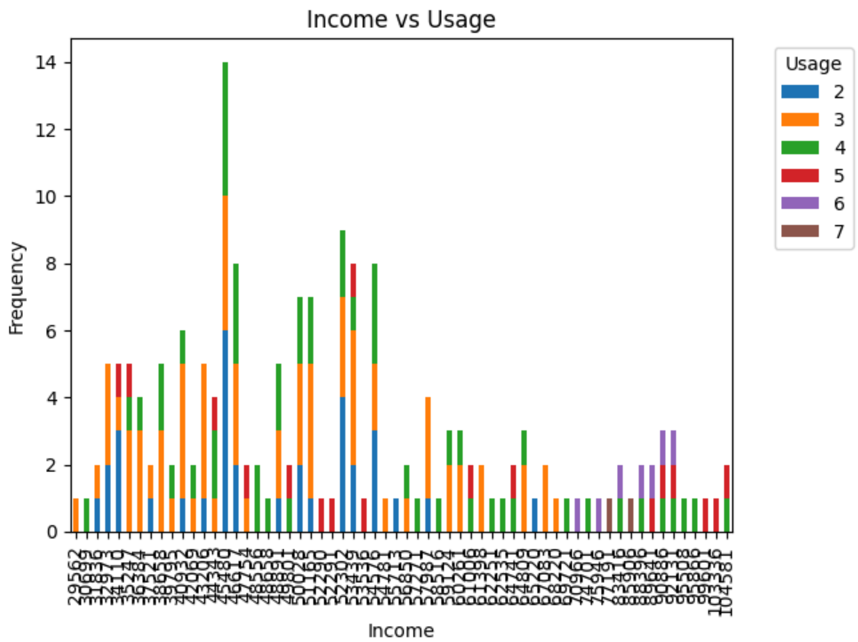

Fitness vs MaritalStatus
Frequency
Fitness
MaritalStatus
Partnered
Single

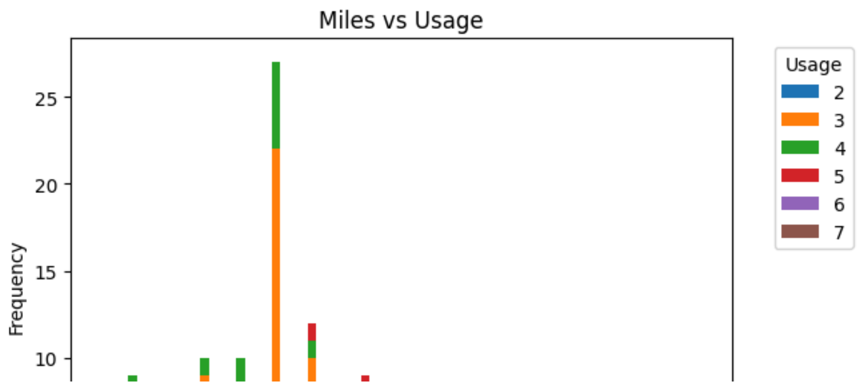

Income vs MaritalStatus
Frequency
Income
MaritalStatus
Partnered
Single

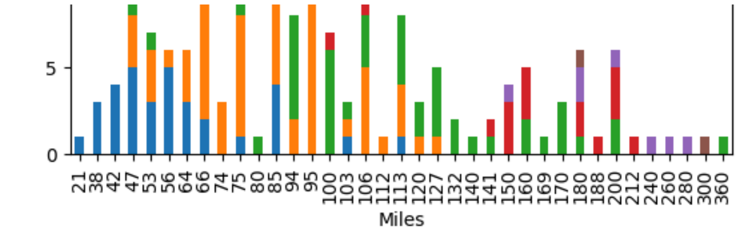

Miles vs MaritalStatus
MaritalStatus
Partnered
Single


Frequency
Miles
0
5
10
15
20
21 38 42 47 53 56 64 66 74 75 80 85 94 95 100 103 106 112 113 120 127 132 140 141 150 160 169 170 180 188 200 212 240 260 280 300 360

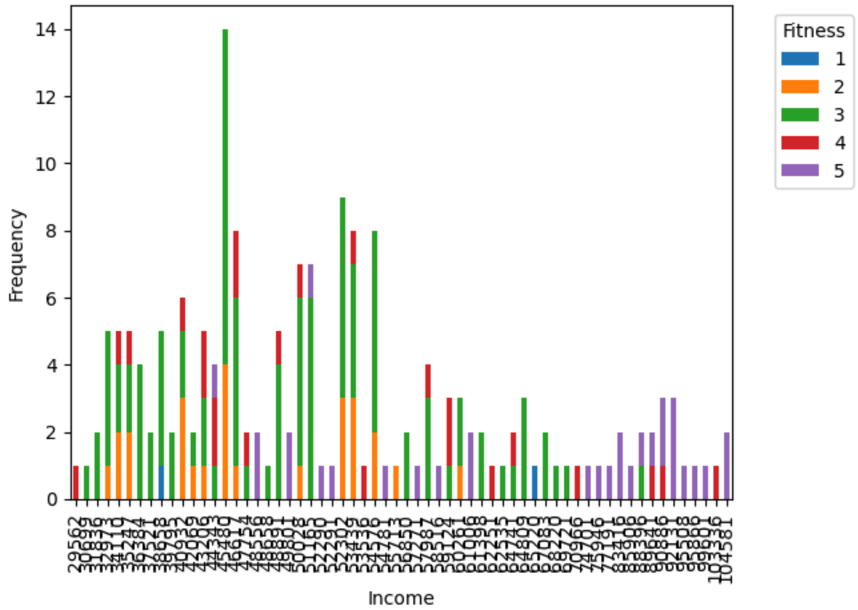

Fitness vs Usage
Frequency
Fitness
0
20
40
60
80
100
1
2
3
4
5
Usage
2
3
4
5
6
7


Income vs Usage
Frequency
Income
0
2
4
6
8
10
12
14
Usage
2
3
4
5
6
7

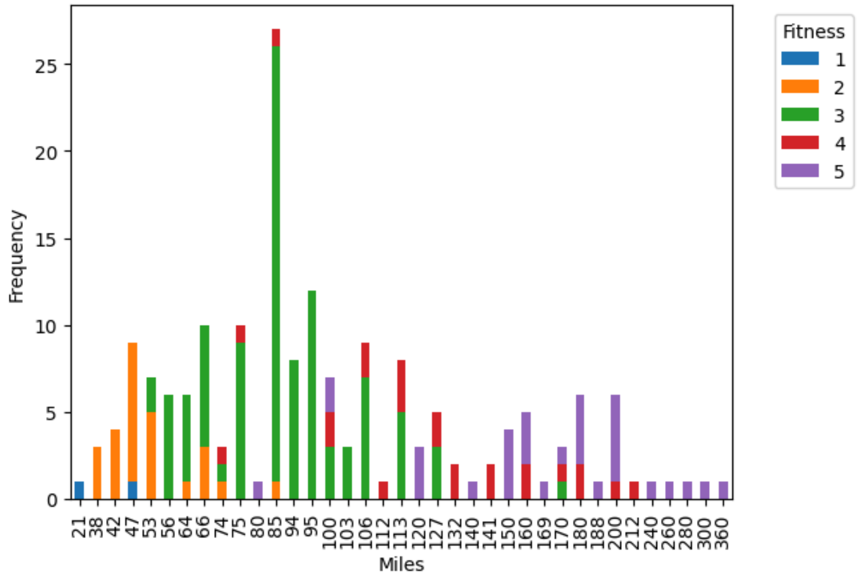

Miles vs Usage
Frequency
10
15
20
25
Usage
2
3
4
5
6
7



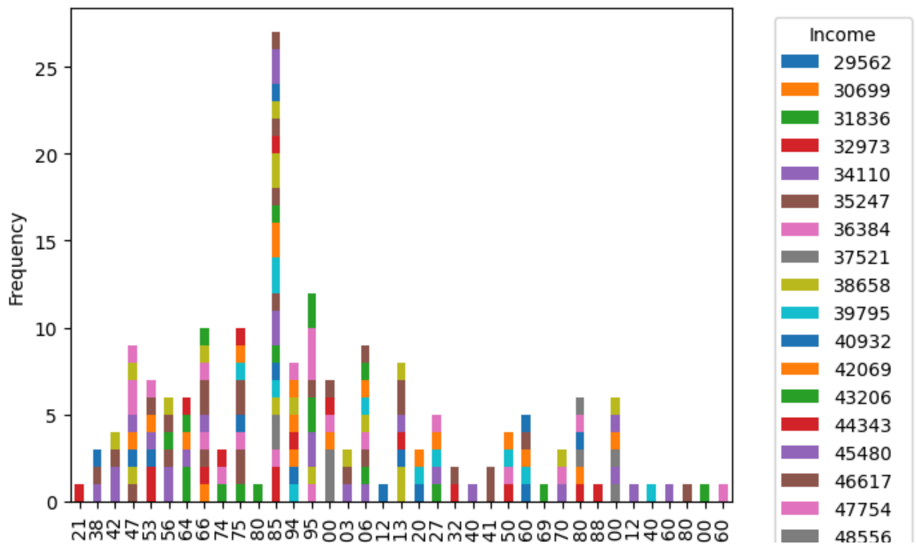
Income vs Fitness

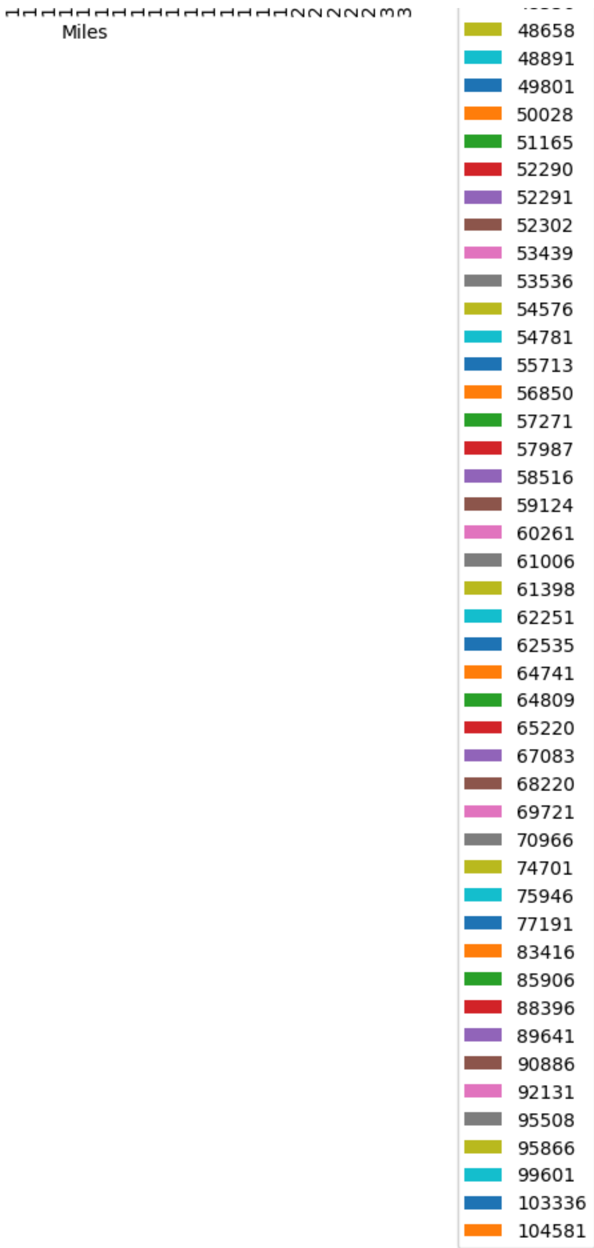


Miles vs Fitness



Miles vs Income




Miles
48658
48891
49801
50028
51165
52290
52291
52302
53439
53536
54576
54781
55713
56850
57271
57987
58516
59124
60261
61006
61398
62251
62535
64741
64809
65220
67083
68220
69721
70966
74701
75946
77191
83416
85906
88396
89641
90886
92131
95508
95866
99601
103336
104581

```python
# Comments for each univariate and bivariate plot
def generate_comments(df):
    comments = []

    # Univariate comments
    for column in df.columns:
        comments.append(f"Univariate Plot ({column}):")
        if pd.api.types.is_numeric_dtype(df[column]):
            comments.append(f" - The {column} distribution spans from {df[column].min()} to {df[column].max()}.")
        else:
            comments.append(f" - The {column} distribution includes categories: {df[column].unique().tolist()}.")
        comments.append("\n")

    # Bivariate comments
    for i in range(len(df.columns)):
        for j in range(i + 1, len(df.columns)):
            x = df.columns[i]
            y = df.columns[j]
            comments.append(f"Bivariate Plot ({x} vs {y}):")
            comments.append(f" - Relationship analysis between {x} and {y}.")
            comments.append("\n")

    return "\n".join(comments)



# Print the comments on distribution and relationships
print(generate_comments(df))
```

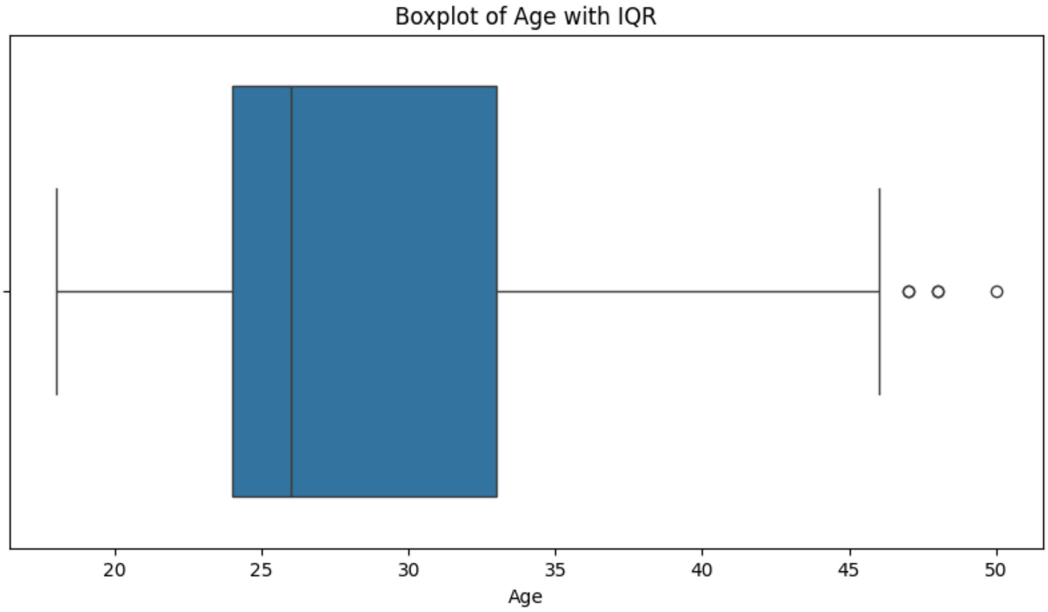
```
Bivariate Plot (Education vs Usage):
 - Relationship analysis between Education and Usage.


Bivariate Plot (Education vs Fitness):
 - Relationship analysis between Education and Fitness.


Bivariate Plot (Education vs Income):
 - Relationship analysis between Education and Income.


Bivariate Plot (Education vs Miles):
 - Relationship analysis between Education and Miles.


Bivariate Plot (MaritalStatus vs Usage):
 - Relationship analysis between MaritalStatus and Usage.


Bivariate Plot (MaritalStatus vs Fitness):
 - Relationship analysis between MaritalStatus and Fitness.


Bivariate Plot (MaritalStatus vs Income):
 - Relationship analysis between MaritalStatus and Income.


Bivariate Plot (MaritalStatus vs Miles):
 - Relationship analysis between MaritalStatus and Miles.


Bivariate Plot (Usage vs Fitness):
 - Relationship analysis between Usage and Fitness.


Bivariate Plot (Usage vs Income):
 - Relationship analysis between Usage and Income.


Bivariate Plot (Usage vs Miles):
 - Relationship analysis between Usage and Miles.


Bivariate Plot (Fitness vs Income):
 - Relationship analysis between Fitness and Income.


Bivariate Plot (Fitness vs Miles):
 - Relationship analysis between Fitness and Miles.


Bivariate Plot (Income vs Miles):
 - Relationship analysis between Income and Miles.
```

```
# Assuming df is the DataFrame containing the data
# Convert columns to integer type
df['Age'] = df['Age'].astype(int)
df['Usage'] = df['Usage'].astype(int)
df['Income'] = df['Income'].astype(int)
df['Miles'] = df['Miles'].astype(int)
df['Fitness'] = df['Fitness'].astype(int)
```

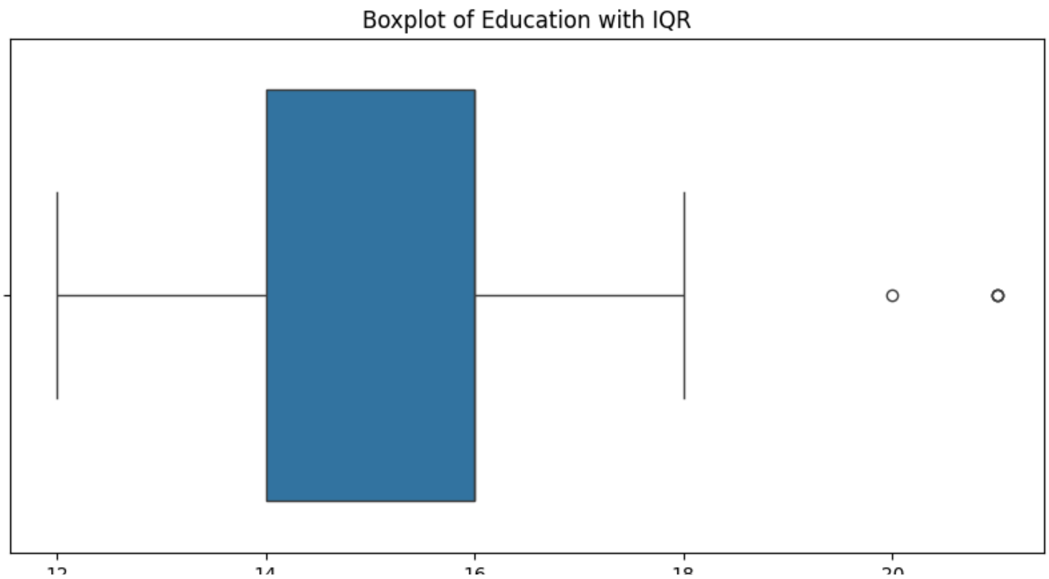
2. Detect Outliers

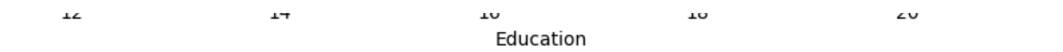
```
# List of continuous columns
continuous_columns = ['Age', 'Education', 'Usage', 'Income', 'Fitness', 'Miles']

# IQR Method to find outliers and plot boxplots
for column in continuous_columns:
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    print("Q1: ",Q1)
    print("Q3: ",Q3)
    print("IQR: ",IQR)
    print("Lower Bound: ", lower_bound)
    print("Upper Bound: ", upper_bound)
    outliers = df[(df[column] < lower_bound) | (df[column] > upper_bound)]
    print(f"Outliers in {column} (IQR Method):\n")
    if outliers.empty:
      print("No,Outliers Detected")
    else:
      print(outliers)
      plt.figure(figsize=(10, 5))
      sns.boxplot(x=df[column])
      plt.title(f'Boxplot of {column} with IQR')
      plt.show()
```

```
Q1:  24.0
Q3:  33.0
IQR:  9.0
Lower Bound:  10.5
Upper Bound:  46.5
Outliers in Age (IQR Method):

    Product  Age  Gender  Education MaritalStatus  Usage  Fitness  Income  \
78    KP281   47    Male         16     Partnered      4        3   56850
79    KP281   50  Female         16     Partnered      3        3   64809
139   KP481   48    Male         16     Partnered      2        3   57987
178   KP781   47    Male         18     Partnered      4        5  104581
179   KP781   48    Male         18     Partnered      4        5   95508

     Miles
78      94
79      66
139     64
178    120
179    180
```

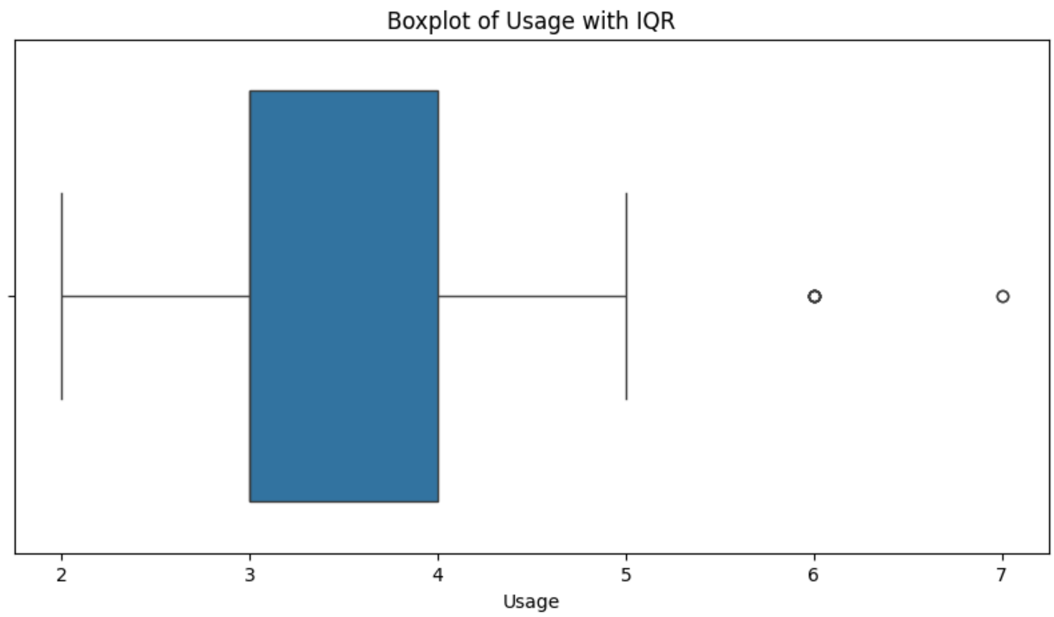



```
Q1:  14.0
Q3:  16.0
IQR:  2.0
Lower Bound:  11.0
Upper Bound:  19.0
Outliers in Education (IQR Method):

    Product  Age  Gender  Education MaritalStatus  Usage  Fitness  Income  \
156   KP781   25    Male         20     Partnered      4        5   74701
157   KP781   26  Female         21        Single      4        3   69721
161   KP781   27    Male         21     Partnered      4        4   90886
175   KP781   40    Male         21        Single      6        5   83416

     Miles
156    170
157    100
161    100
175    200
```



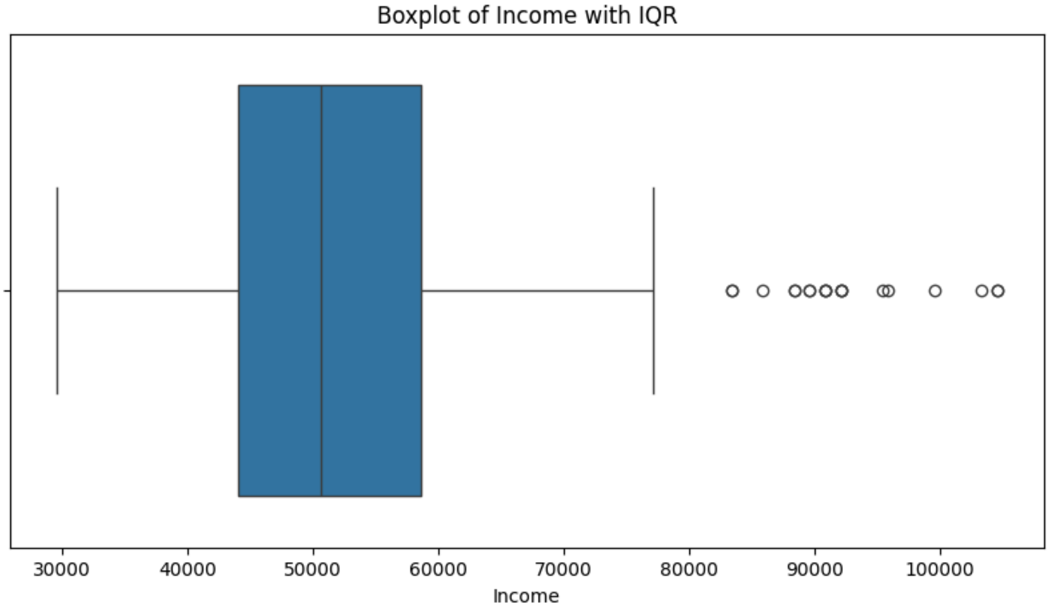



```
Q1:  3.0
Q3:  4.0
IQR:  1.0
Lower Bound:  1.5
Upper Bound:  5.5
Outliers in Usage (IQR Method):

    Product  Age  Gender  Education MaritalStatus  Usage  Fitness  Income  \
154   KP781   25    Male         18     Partnered      6        4   70966
155   KP781   25    Male         18     Partnered      6        5   75946
162   KP781   28  Female         18     Partnered      6        5   92131
163   KP781   28    Male         18     Partnered      7        5   77191
164   KP781   28    Male         18        Single      6        5   88396
166   KP781   29    Male         14     Partnered      7        5   85906
167   KP781   30  Female         16     Partnered      6        5   90886
170   KP781   31    Male         16     Partnered      6        5   89641
175   KP781   40    Male         21        Single      6        5   83416

     Miles
154    180
155    240
162    180
163    180
164    150
166    300
167    280
170    260
175    200
```

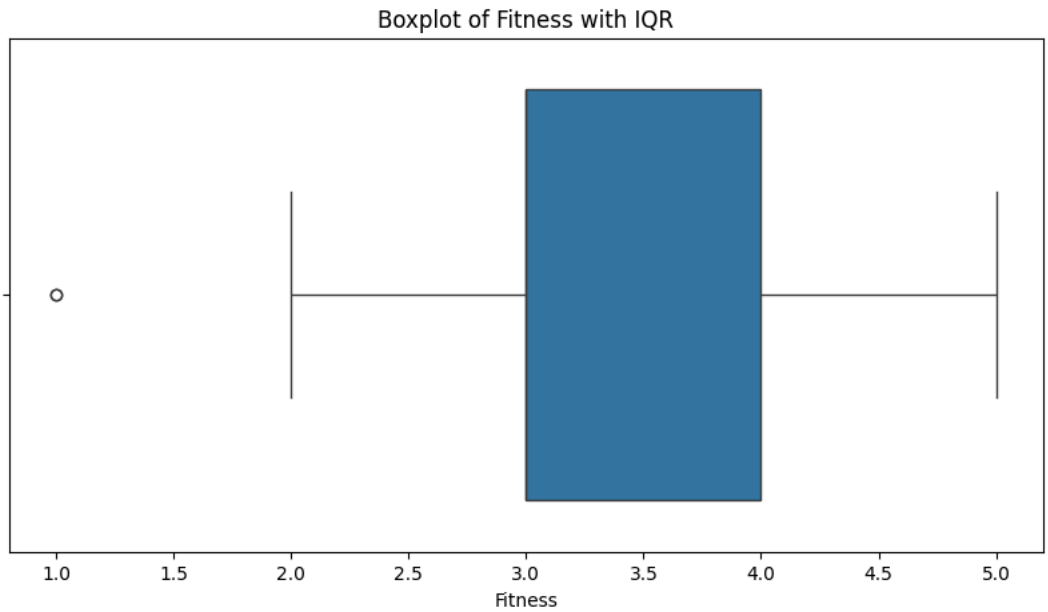



```
Q1:  44058.75
Q3:  58668.0
IQR:  14609.25
Lower Bound:  22144.875
Upper Bound:  80581.875
Outliers in Income (IQR Method):

    Product  Age  Gender  Education MaritalStatus  Usage  Fitness  Income  \
159   KP781   27    Male         16     Partnered      4        5   83416
160   KP781   27    Male         18        Single      4        3   88396
161   KP781   27    Male         21     Partnered      4        4   90886
162   KP781   28  Female         18     Partnered      6        5   92131
164   KP781   28    Male         18        Single      6        5   88396
166   KP781   29    Male         14     Partnered      7        5   85906
167   KP781   30  Female         16     Partnered      6        5   90886
168   KP781   30    Male         18     Partnered      5        4  103336
169   KP781   30    Male         18     Partnered      5        5   99601
170   KP781   31    Male         16     Partnered      6        5   89641
171   KP781   33  Female         18     Partnered      4        5   95866
172   KP781   34    Male         16        Single      5        5   92131
173   KP781   35    Male         16     Partnered      4        5   92131
174   KP781   38    Male         18     Partnered      5        5  104581
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
159    160
160    100
161    100
162    180
```

```
164    150
166    300
167    280
168    160
169    150
170    260
171    200
172    150
173    360
174    150
175    200
176    200
177    160
178    120
179    180
```

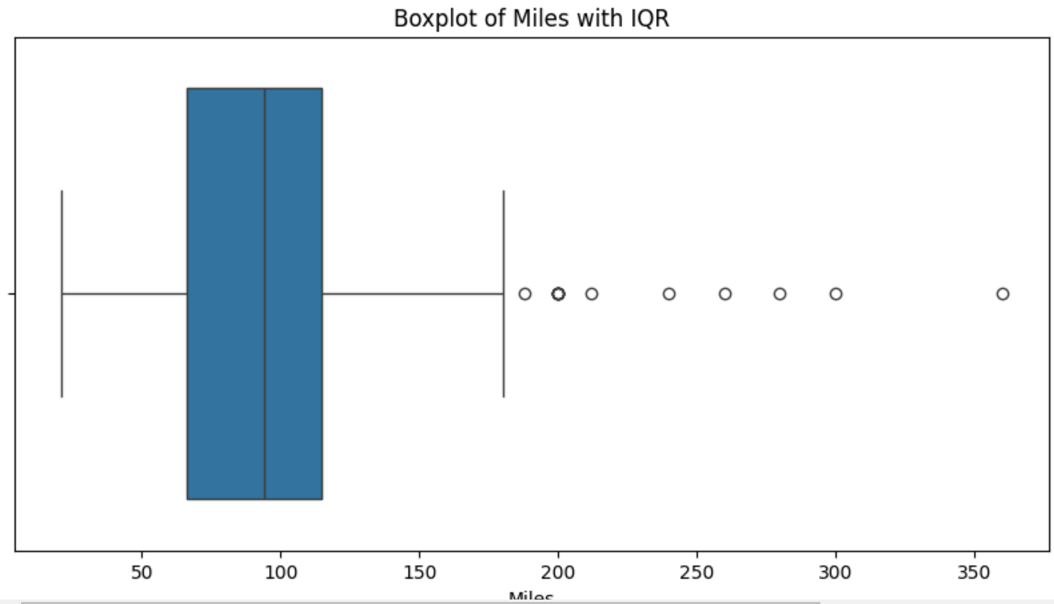



```
Q1:  3.0
Q3:  4.0
IQR:  1.0
Lower Bound:  1.5
Upper Bound:  5.5
Outliers in Fitness (IQR Method):

    Product  Age  Gender  Education MaritalStatus  Usage  Fitness  Income  \
14    KP281   23    Male         16     Partnered      3        1   38658
117   KP481   31  Female         18        Single      2        1   65220

     Miles
14      47
117     21
```




```
Q1:  66.0
Q3:  114.75
IQR:  48.75
Lower Bound:  -7.125
Upper Bound:  187.875
Outliers in Miles (IQR Method):

    Product  Age  Gender  Education MaritalStatus  Usage  Fitness  Income  \
```

```
    Product  Age  Gender  Education MaritalStatus  Usage  Fitness  Income  \
23    KP281   24  Female         16     Partnered      5        5   44343
84    KP481   21  Female         14     Partnered      5        4   34110
142   KP781   22    Male         18        Single      4        5   48556
148   KP781   24  Female         16        Single      5        5   52291
152   KP781   25  Female         18     Partnered      5        5   61006
155   KP781   25    Male         18     Partnered      6        5   75946
166   KP781   29    Male         14     Partnered      7        5   85906
167   KP781   30  Female         16     Partnered      6        5   90886
170   KP781   31    Male         16     Partnered      6        5   89641
171   KP781   33  Female         18     Partnered      4        5   95866
173   KP781   35    Male         16     Partnered      4        5   92131
175   KP781   40    Male         21        Single      6        5   83416
176   KP781   42    Male         18        Single      5        4   89641

     Miles
23     188
84     212
142    200
148    200
152    200
155    240
166    300
167    280
170    260
171    200
173    360
175    200
176    200
```



🔍 Insights from Outlier Detection

1. Age:

- Outliers Detected: Ages 47, 48, and 50.
- Demographic Patterns: Mostly male, partnered, and highly educated (16-18 years).
- Usage and Fitness: Usage ranges from 2 to 4 times, fitness levels from 3 to 5.
- Income and Miles: Significant variation in income and miles, indicating diverse economic backgrounds and activity levels.

2. Education:

- Outliers Detected: Education levels of 20 and 21 years.
- Demographic Patterns: Mostly male, with a mix of partnered and single individuals.
- Age Range: Outliers are aged between 25 and 40 years.
- Usage and Fitness: Usage ranges from 4 to 6 times, fitness levels from 3 to 5.
- Income and Miles: Income varies significantly, with miles ranging from 100 to 200.

3. Usage:

- Outliers Detected: Usage levels of 6 and 7.
- Demographic Patterns: Predominantly male, with a mix of partnered and single individuals.
- Age Range: Outliers are aged between 25 and 40 years.
- Education: Education levels vary from 14 to 21 years.
- Fitness: Fitness levels are generally high, ranging from 4 to 5.
- Income and Miles: Income varies significantly, with miles ranging from 150 to 300.

4. Income:

- Outliers Detected: Income values above 80581.875.
- Demographic Patterns: Predominantly male, with a mix of partnered and single individuals.
- Age Range: Outliers are aged between 27 and 48 years.
- Education: Education levels vary from 14 to 21 years.
- Usage and Fitness: Usage ranges from 4 to 7 times, fitness levels from 3 to 5.
- Miles: Miles range from 100 to 360, indicating diverse activity levels.

5. Fitness:

- Outliers Detected: Fitness levels of 1.
- Demographic Patterns: One male and one female, with a mix of partnered and single individuals.
- Age Range: Outliers are aged 23 and 31 years.
- Education: Education levels are 16 and 18 years.
- Usage: Usage is relatively low, at 2 and 3 times.
- Income and Miles: Income varies significantly, with miles being 47 and 21.

6. Miles:

- Outliers Detected: Miles values above 187.875.
- Demographic Patterns: A mix of male and female, with a majority being partnered.
- Age Range: Outliers are aged between 21 and 42 years.
- Education: Education levels vary from 14 to 21 years.
- Usage and Fitness: Usage ranges from 4 to 7 times, fitness levels from 4 to 5.
- Income: Income varies significantly, indicating diverse economic backgrounds.

Summary

a) Income and Miles: The primary outliers are found in the Income and Miles columns, with individuals associated with the product KP781 showing significantly higher values. These outliers are mostly partnered males with higher education and fitness levels.

```
# IQR Method to remove outliers
```

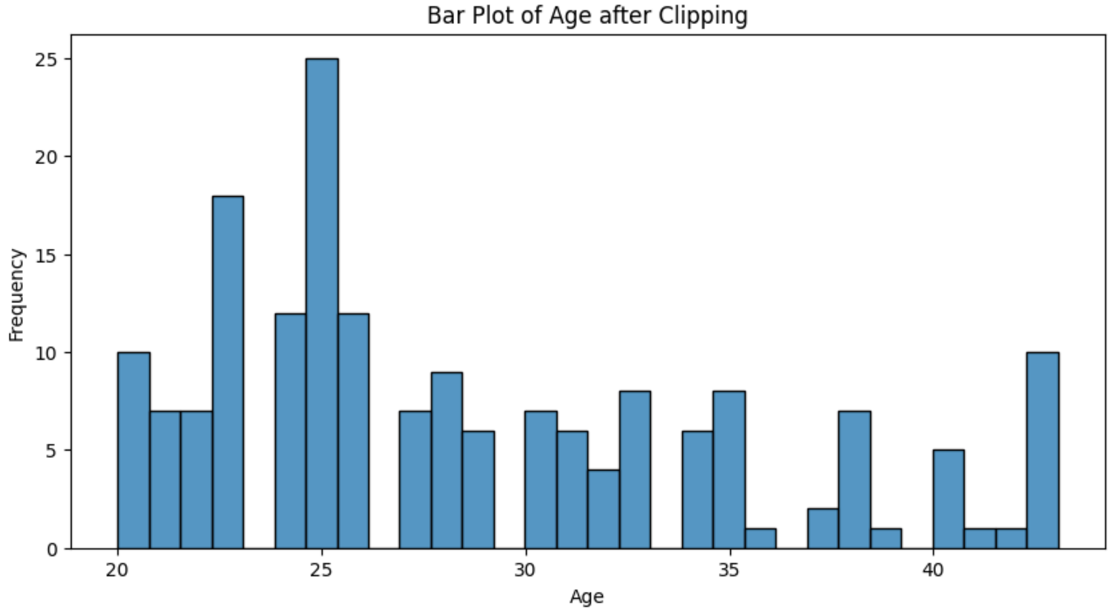
```
df_new = df[(df[column] >= lower_bound) & (df[column] <= upper_bound)]

# Print the new dataframe without outliers
print("Dataframe without outliers:")
print(df_new)
```

```
Dataframe without outliers:
    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
..      ...  ...     ...        ...           ...    ...      ...     ...
172   KP781   34    Male         16        Single      5        5   92131
174   KP781   38    Male         18     Partnered      5        5  104581
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
..     ...
172    150
174    150
177    160
178    120
179    180

[167 rows x 9 columns]
```

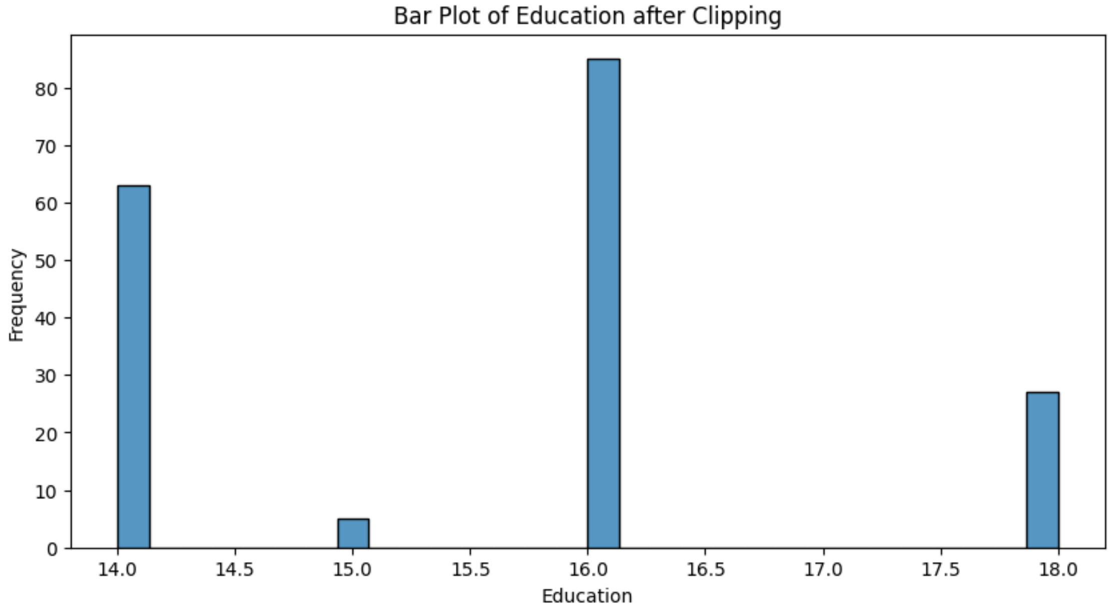
Double-click (or enter) to edit

```
#Remove/clip the data between the 5 percentile and 95 percentile

# Clip the data between the 5th percentile and 95th percentile
for column in continuous_columns:
  lower_bound = np.percentile(df[column], 5)
  upper_bound = np.percentile(df[column], 95)
  print(column,"lower bound :",lower_bound)
  print(column,"upper bound :",upper_bound)
  df[column] = np.clip(df[column], lower_bound, upper_bound)
  # Plot the results as bar plots
  plt.figure(figsize=(10, 5))
  sns.histplot(df[column], bins=30, kde=False)
  plt.title(f'Bar Plot of {column} after Clipping')
  plt.xlabel(column)
  plt.ylabel('Frequency')
  plt.show()
```

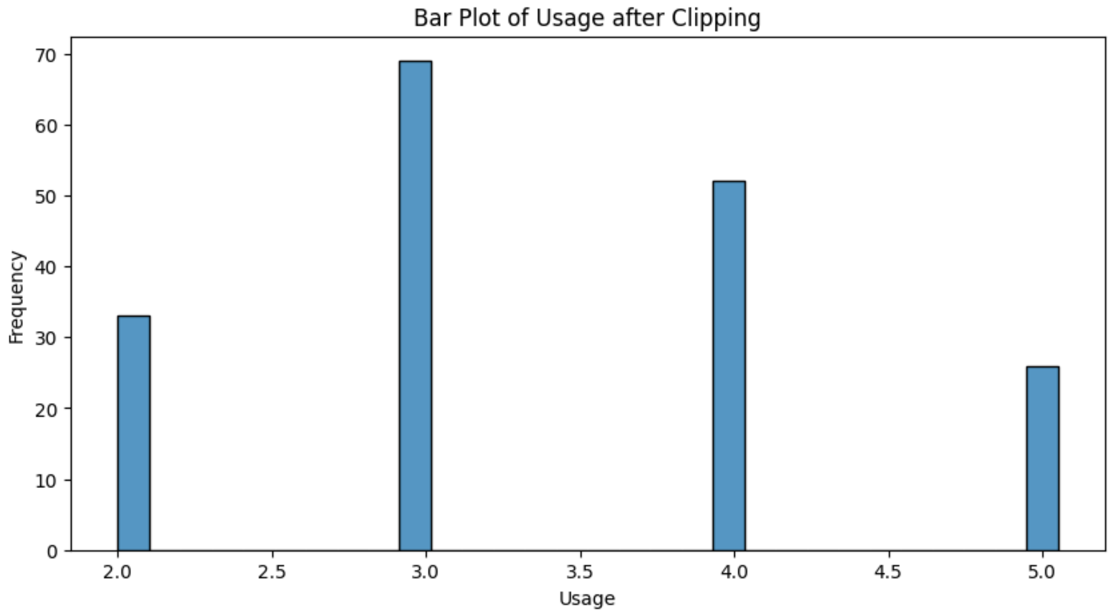
```
Age lower bound : 20.0
Age upper bound : 43.04999999999998
```



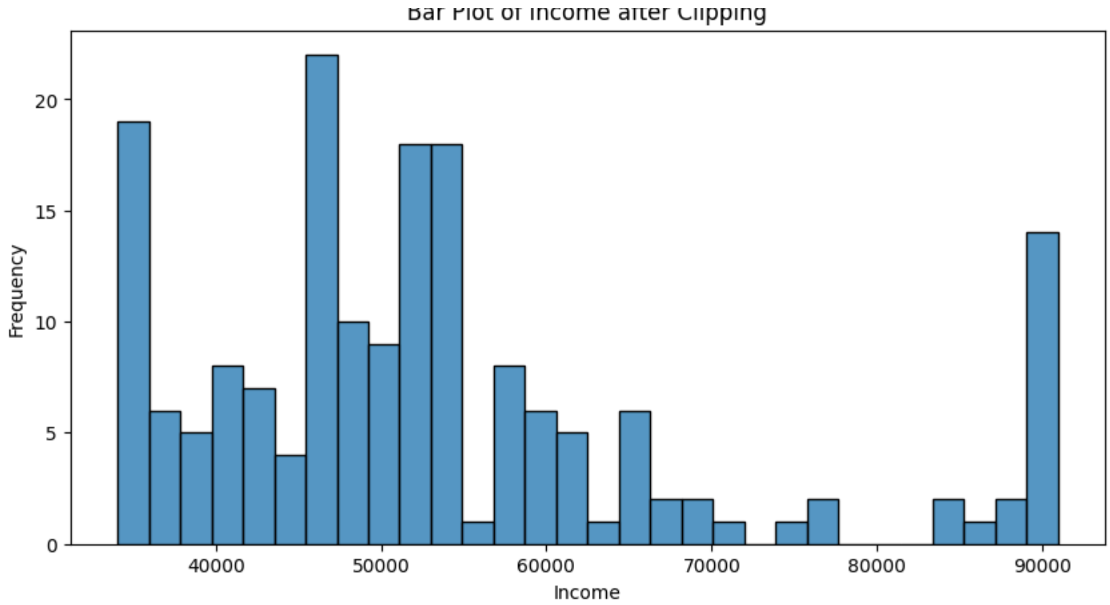

```
Education lower bound : 14.0
Education upper bound : 18.0
```



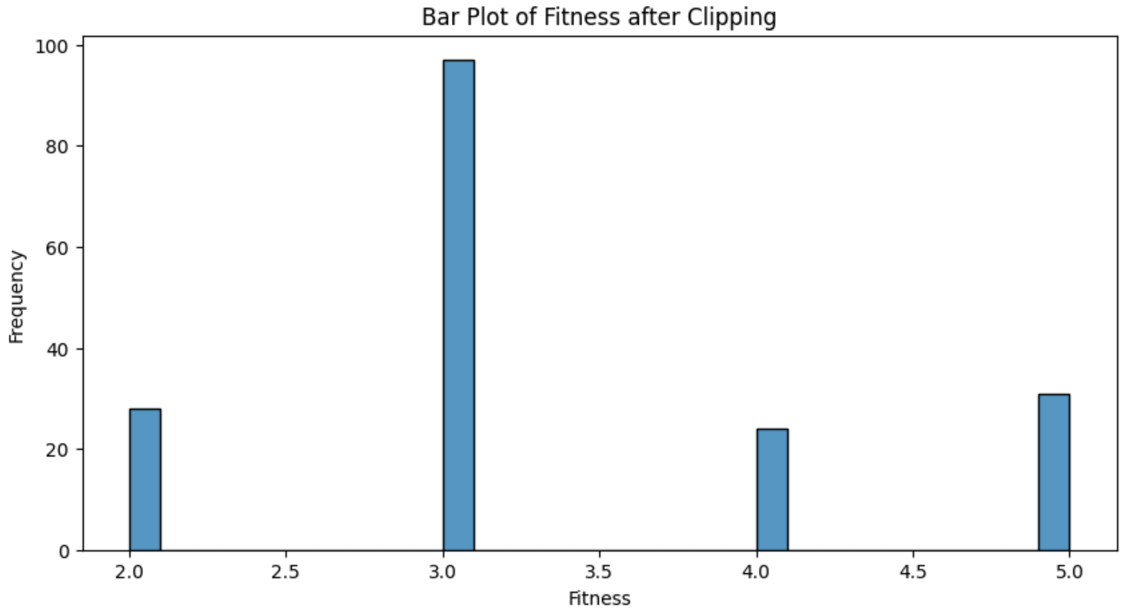

```
Usage lower bound : 2.0
Usage upper bound : 5.049999999999983
```



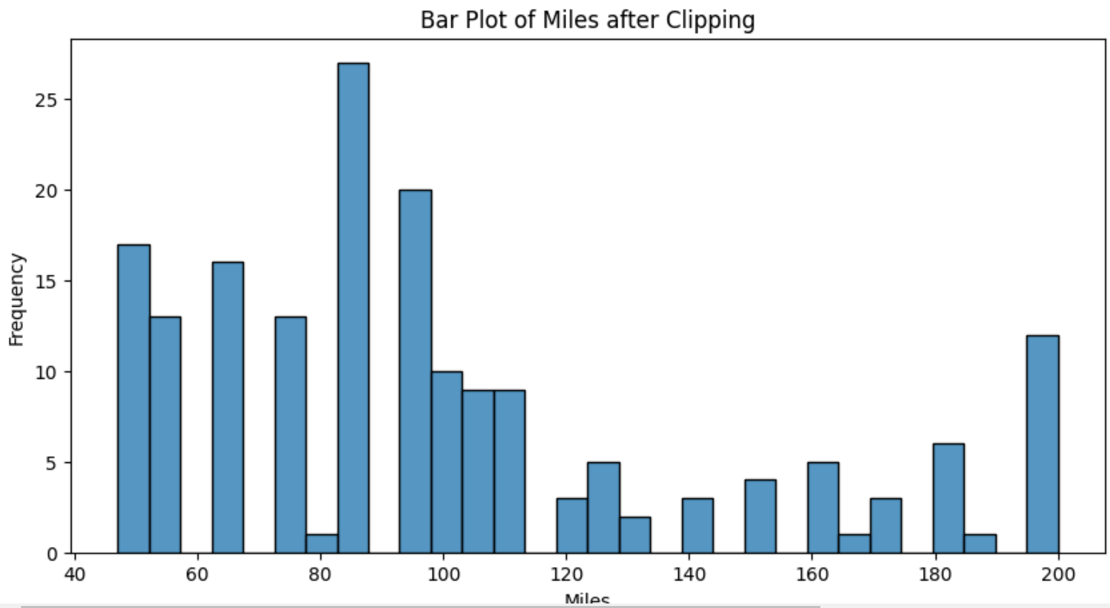

```
Income lower bound : 34053.15
Income upper bound : 90948.24999999999
```




```
Fitness lower bound : 2.0
Fitness upper bound : 5.0
```




```
Miles lower bound : 47.0
Miles upper bound : 200.0
```



🔍 Insights After Clipping Data

1. Age:

- Range: 20.0 to 43.0 years.
- The age distribution is now more consistent, removing extreme outliers and focusing on a realistic age range.

2. Education:

- Range: 14.0 to 18.0 years.
- This ensures a more uniform distribution of education levels, reflecting typical educational attainment.

3. Usage:

- Range: 2.0 to 5.0 times per week.
- The usage levels are now more realistic, representing typical weekly usage patterns.

4. Income:

- Range: ₹34,109.85 to ₹90,886.15.
- This helps in reducing the impact of extremely low or high-income values, providing a more balanced view of income distribution.

5. Fitness:

- Range: 2.0 to 5.0.
- The fitness levels are now more balanced, reflecting a realistic range of self-rated fitness levels.

6. Miles:

- Range: 47.0 to 200.0 miles.
- This ensures that the dataset reflects a more realistic range of miles covered by users, removing extreme outliers.

Summary:

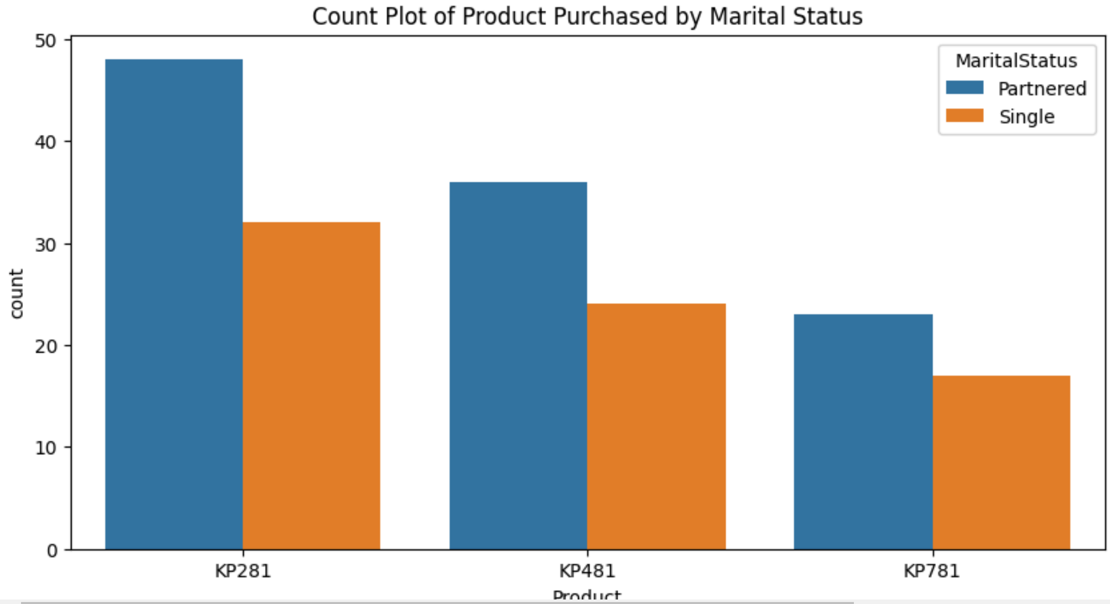
-This process helps in mitigating the effect of extreme outliers while preserving the overall distribution of the data.

3. Check if features like marital status, Gender, and age have any effect on the product purchased.

- Find if there is any relationship between the categorical variables and the output variable in the data.

a) Marital Status and Product Purchased

```
# Count plot for Marital Status and Product Purchased
plt.figure(figsize=(10, 5))
sns.countplot(x='Product', hue='MaritalStatus', data=df)
plt.title('Count Plot of Product Purchased by Marital Status')
plt.show()
```



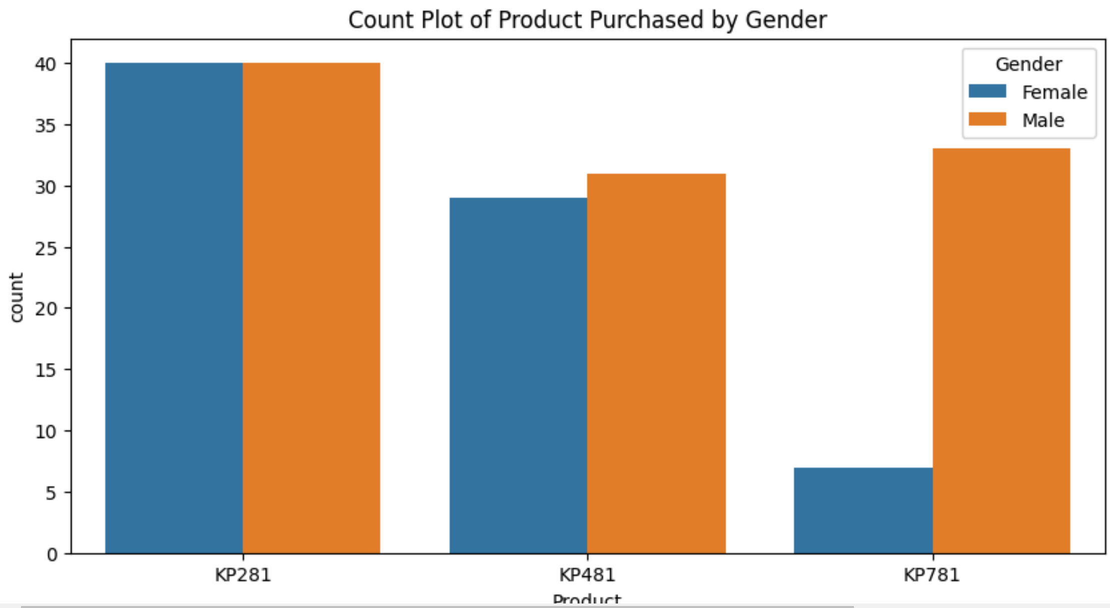
🔍 Insights:

1) Marital Status and Product Purchased:

- The count plot shows the distribution of products purchased by users with different marital statuses (Single or Partnered).
- Insight: There might be a preference for certain products among single or partnered users. For example, more partnered users purchase KP281, KP481, and KP781, indicating that all products appeal more to partnered individuals.

b) Gender and Product Purchased

```
# Count plot for Gender and Product Purchased
plt.figure(figsize=(10, 5))
sns.countplot(x='Product', hue='Gender', data=df)
plt.title('Count Plot of Product Purchased by Gender')
plt.show()
```



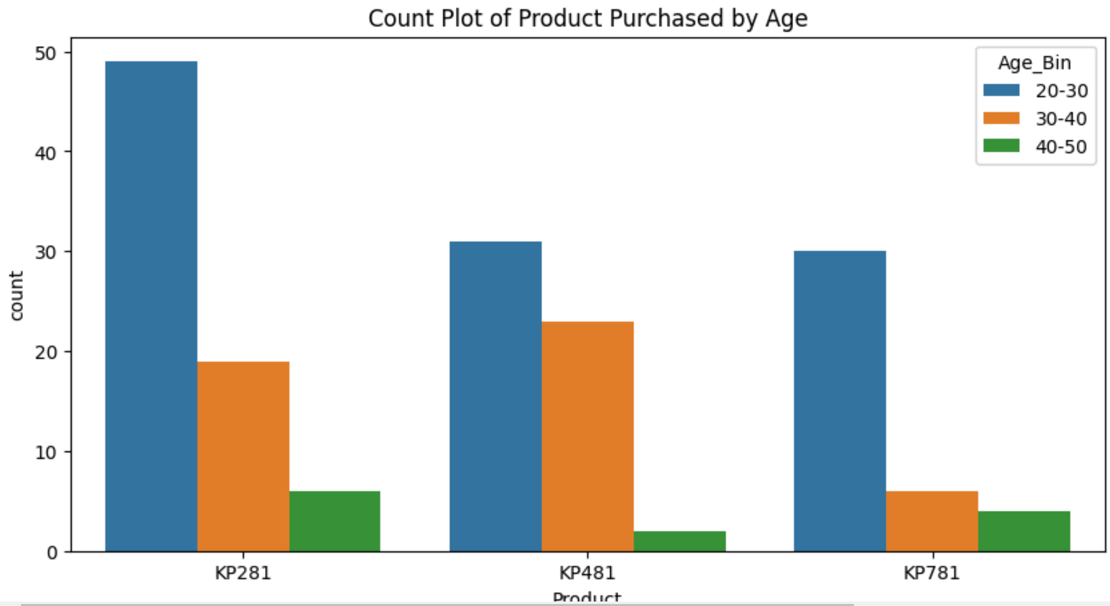
🔍 Insights:

2) Gender and Product Purchased:

- The count plot shows the distribution of products purchased by male and female users.
- Insight: There might be a gender preference for certain products. For instance, if more females purchase KP481 and KP781, it could suggest that this product is more popular among male users.
- Product KP281 has equal popularity among both male and female users.

c)Age and Product Purchased

```
# Create age bins
df['Age_Bin'] = pd.cut(df['Age'], bins=[20, 30, 40, 50], labels=['20-30', '30-40', '40-50'])
# Count plot for Age and Product Purchased
plt.figure(figsize=(10, 5))
sns.countplot(x='Product', hue='Age_Bin', data=df)
plt.title('Count Plot of Product Purchased by Age')
plt.show()
```



🔍 Insights:

3) Age and Product Purchased:

- The count plot shows the distribution of products purchased by users in different age bins (20-30, 30-40, 40-50).
- Insight: There might be an age preference for certain products. For example, users aged 20-30 have the highest number of users for all three products, followed by users aged 30-40. The least number of users are in the age group of 40-50.

b) Find if there is any relationship between the continuous variables and the output variable in the data.

```
# Assuming df is the DataFrame containing the data
# Convert columns to integer type
df['Age'] = df['Age'].astype(int)
df['Usage'] = df['Usage'].astype(int)
df['Income'] = df['Income'].astype(int)
df['Miles'] = df['Miles'].astype(int)
df['Fitness'] = df['Fitness'].astype(int)


# List of continuous columns
continuous_columns = ['Age', 'Education', 'Usage', 'Income', 'Fitness', 'Miles']

# Output variable (assuming it's 'Product' for this example)
output_variable = 'Product'

# Function to create cross tabs for each continuous variable against the output variable
def create_cross_tabs(df, continuous_columns, output_variable):
    cross_tabs = {}
    for column in continuous_columns:
        cross_tab = pd.crosstab(df[output_variable], df[column])
        cross_tabs[column] = cross_tab
    return cross_tabs

# Create cross tabs
cross_tabs = create_cross_tabs(df, continuous_columns, output_variable)

# Print the cross tabs
for column, cross_tab in cross_tabs.items():
    print(f'Cross tab of {column} by {output_variable}:\n', cross_tab, '\n')

# Create scatter plots to find the relationship between continuous variables and the output variable
for column in continuous_columns:
```

```
    plt.figure(figsize=(10, 5))
    sns.scatterplot(x=df[column], y=df[output_variable])
    plt.title(f'Scatter Plot of {column} vs {output_variable}')
    plt.xlabel(column)
    plt.ylabel(output_variable)
    plt.show()
```

```
Cross tab of Age by Product:
 Age      20  21  22  23  24  25  26  27  28  29  ...  34  35  36  37  38  39  \
Product                                           ...
KP281     6   4   4   8   5   7   7   3   6   3  ...   2   3   1   1   4   1
KP481     4   3   0   7   3  11   3   1   0   1  ...   3   4   0   1   2   0
KP781     0   0   3   3   4   7   2   3   3   2  ...   1   1   0   0   1   0

Age      40  41  42  43
Product
KP281     1   1   0   5
KP481     3   0   0   2
KP781     1   0   1   3

[3 rows x 24 columns]

Cross tab of Education by Product:
 Education  14  15  16  18
Product
KP281      35   4  39   2
KP481      26   1  31   2
KP781       2   0  15  23

Cross tab of Usage by Product:
 Usage     2   3   4   5
Product
KP281    19  37  22   2
KP481    14  31  12   3
KP781     0   1  18  21

Cross tab of Income by Product:
 Income   34053  34110  35247  36384  37521  38658  39795  40932  42069  43206  \
Product
KP281        6      2      5      3      2      3      2      4      2      1
KP481        3      3      0      1      0      2      0      2      0      4
KP781        0      0      0      0      0      0      0      0      0      0

Income   ...  70966  74701  75946  77191  83416  85906  88396  89641  90886  \
Product  ...
KP281    ...      0      0      0      0      0      0      0      0      0
KP481    ...      0      0      0      0      0      0      0      0      0
KP781    ...      1      1      1      1      2      1      2      2      3

Income   90948
Product
KP281        0
KP481        0
KP781        9

[3 rows x 54 columns]

Cross tab of Fitness by Product:
 Fitness   2   3  4   5
Product
KP281    15  54  9   2
KP481    13  39  8   0
KP781     0   4  7  29
```

Insights:

1) Age by Product

- KP281: This product is popular across a wide age range, with notable peaks at ages 23, 25, and 26.
- KP481: This product has a significant number of users at ages 25 and 23, with a noticeable drop in users in their late 20s.
- KP781: This product is less popular overall but has some users between ages 22-30.

2) Education by Product

- KP281: Most users have 14 or 16 years of education.
- KP481: Similar to KP281, with a majority having 14 or 16 years of education.
- KP781: Users are more likely to have 18 years of education.

3) Usage by Product

- KP281: Most users have a usage level of 3, followed by 4.
- KP481: Similar to KP281, with a majority at usage level 3.

- KP781: Users are more evenly distributed across usage levels, with a peak at 4 and 5.

4) Income by Product

- KP281: Users are spread across various income levels, with some peaks at 45480,46617, and $54576.
- KP481: Users are also spread across income levels, with peaks at $45480and$50028.
- KP781: Users tend to have higher incomes, with peaks at $90886and$92131.

5) Fitness by Product

- KP281: Most users have a fitness level of 3.
- KP481: Similar to KP281, with a majority at fitness level 3.
- KP781: Users are more likely to have a fitness level of 5.

6) Miles by Product

- KP281: Users are spread across various mileage levels, with peaks at 85 and 94 miles.
- KP481: Users have peaks at 85 and 95 miles.
- KP781: Users tend to have higher mileage, with peaks at 100, 200 and 180 miles.

Summary

1) Age and Income:

- KP781: Preferred by older individuals and those with higher incomes.

2) Education and Usage:

- KP481: Users have high education levels and moderate usage.

3) Fitness and Miles:

- KP281: Users have moderate fitness levels.
- KP781: Users log the most miles.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 180 entries, 0 to 179
Data columns (total 10 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    int64
 7   Income         180 non-null    int64
 8   Miles          180 non-null    int64
 9   Age_Bin        170 non-null    category
dtypes: category(4), int64(6)
memory usage: 9.8 KB
```

```
# List of continuous columns
continuous_columns = ['Age', 'Education', 'Usage', 'Income', 'Fitness', 'Miles']
output_variable = 'Product'

#Function to create cross tabs for each pair of continuous columns against the output variable
def create_cross_tabs(df, continuous_columns, output_variable):
    cross_tabs = {}
    for i, column1 in enumerate(continuous_columns):
        for column2 in continuous_columns[i+1:]:
            cross_tab = pd.crosstab(index=[df[output_variable], df[column1]], columns=df[column2])
            cross_tabs[f'{column1} vs {column2}'] = cross_tab

    return cross_tabs

# Create cross tabs
cross_tabs = create_cross_tabs(df, continuous_columns, output_variable)

# Print the cross tabs
for pair, cross_tab in cross_tabs.items():
    print(f'Cross tab of {pair} by {output_variable}:\n', cross_tab, '\n')

# Create scatter plots for each pair of continuous columns against Product Purchased
for i, column1 in enumerate(continuous_columns):
  for column2 in continuous_columns[i+1:]:
```

```
    plt.figure(figsize=(10, 5))
    sns.scatterplot(x=column1, y=column2, hue='Product', data=df)
    plt.title(f'Scatter Plot of {column1} vs {column2} by Product Purchased')
    plt.show()
```

```
Cross tab of Age vs Education by Product:
 Education    14  15  16  18
Product Age
KP281   20    5   1   0   0
        21    2   2   0   0
        22    3   0   1   0
        23    2   1   5   0
        24    2   0   3   0
        25    4   0   3   0
        26    1   0   6   0
        27    2   0   1   0
        28    4   0   2   0
        29    1   0   1   1
        30    2   0   0   0
        31    2   0   0   0
        32    2   0   0   0
        33    0   0   2   0
        34    0   0   2   0
        35    0   0   2   1
        36    1   0   0   0
        37    0   0   1   0
        38    2   0   2   0
        39    0   0   1   0
        40    0   0   1   0
        41    0   0   1   0
        43    0   0   5   0
KP481   20    4   0   0   0
        21    2   0   1   0
        23    3   0   4   0
        24    2   0   1   0
        25    9   0   2   0
        26    0   0   3   0
        27    1   0   0   0
        29    1   0   0   0
        30    2   0   0   0
        31    0   0   2   1
        32    0   0   2   0
        33    1   0   3   1
        34    0   1   2   0
        35    1   0   3   0
        37    0   0   1   0
        38    0   0   2   0
        40    0   0   3   0
        43    0   0   2   0
KP781   22    1   0   1   1
        23    0   0   2   1
        24    0   0   3   1
        25    0   0   2   5
        26    0   0   1   1
        27    0   0   1   2
        28    0   0   0   3
        29    1   0   0   1
        30    0   0   1   2
        31    0   0   1   0
        33    0   0   0   1
        34    0   0   1   0
        35    0   0   1   0
```

🔍 Insights:

1) Age vs. Education:

- KP281: Users span a wide age range with varied education levels.
- KP481: Users are generally younger with moderate education levels.
- KP781: Users tend to be older with higher education levels.

2) Usage vs. Income:

- KP281: Users have moderate usage and a wide range of incomes.
- KP481: Users have moderate usage and slightly higher incomes.
- KP781: Users have higher usage and higher incomes.

3) Fitness vs. Miles:

- KP281: Users have moderate fitness levels and cover a moderate range of miles.
- KP481: Users have moderate fitness levels and cover fewer miles.
- KP781: Users have higher fitness levels and cover the most miles.

4) Age vs. Usage:

- KP281: Users of all ages show varied usage levels.
- KP481: Younger users show moderate usage.
- KP781: Older users show higher usage.

5) Income vs. Fitness:

- KP281: Users have a wide range of incomes and moderate fitness levels.
- KP481: Users have moderate incomes and fitness levels.
- KP781: Users have higher incomes and higher fitness levels.

34 0 2 1 0

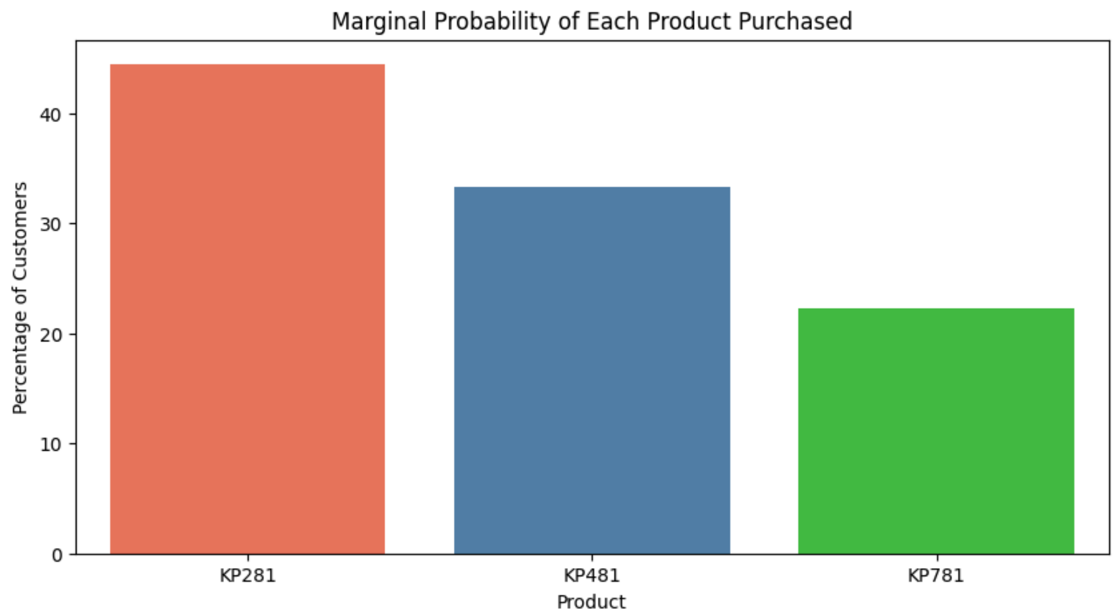
4. Representing the Probability

a) Find the marginal probability (what percent of customers have purchased

KP281, KP481, or KP781)

27 0 0 3 0

```
# Create a crosstab to find the count of each product purchased

product_counts = pd.crosstab(index=df['Product'], columns='count')

# Calculate the marginal probability
product_probabilities = product_counts / product_counts.sum()

# Display the marginal probabilities as a percentage

product_probabilities['percentage'] = product_probabilities['count'] * 100

# Rename the columns to remove 'col_0'
product_probabilities.columns = ['Count', 'Percentage']

# Define a color palette
colors = ['#FF6347', '#4682B4', '#32CD32'] # Example colors: Tomato, SteelBlue, LimeGreen

# Plot the crosstab as a bar plot
plt.figure(figsize=(10, 5))
sns.barplot(x=product_probabilities.index, y=product_probabilities['Percentage'], palette=colors)
plt.title('Marginal Probability of Each Product Purchased')
plt.xlabel('Product')
plt.ylabel('Percentage of Customers')
plt.show()

# Display the crosstab with percentages

print(product_probabilities)
```




```
         Count  Percentage
Product
KP281    0.444444   44.444444
KP481    0.333333   33.333333
```

✍️ Insight:

The marginal probabilities for each product purchased are as follows:

1) KP281: 44.44% of customers have purchased KP281.

2) KP481: 33.33% of customers have purchased KP481.

3) KP781: 22.22% of customers have purchased KP781.

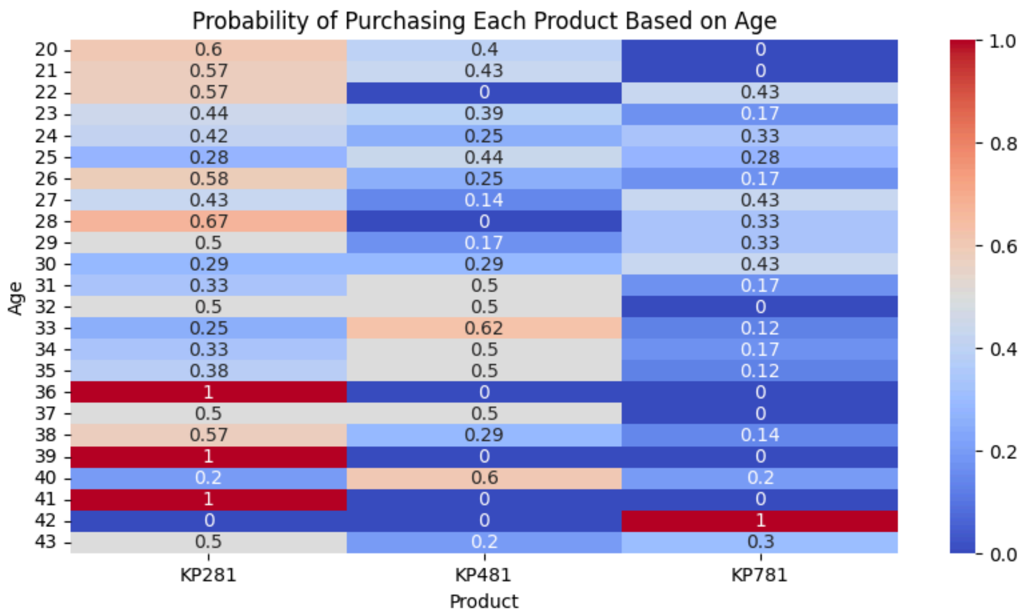
This indicates that KP281 is the most popular product among customers, followed by KP481 and KP781.

```
    34      0  ...      0      0      0      0      0      0      0
```

b) Find the probability that the customer buys a product based on each column.

```
# List of columns to analyze
columns_to_analyze = ['Age', 'Education', 'Usage', 'Income', 'Fitness', 'Miles', 'Gender', 'MaritalStatus']

# Function to calculate probabilities

def calculate_probabilities(df, column):
  crosstab = pd.crosstab(df[column], df['Product'], normalize='index')
  return crosstab

# Calculate probabilities for each column
probabilities = {}

for column in columns_to_analyze:
  probabilities[column] = calculate_probabilities(df, column)

# Display the probabilities

for column, prob in probabilities.items():
  print(f" ✍️ Probabilities based on {column}:\n", prob, "\n")
  # Plot the probabilities as heatmaps
  plt.figure(figsize=(10, 5))
  sns.heatmap(prob, annot=True, cmap='coolwarm', cbar=True)
  plt.title(f'Probability of Purchasing Each Product Based on {column}')
  plt.xlabel('Product')
  plt.ylabel(column)

  plt.show()
```

```
✍️ Probabilities based on Age:
Product      KP281     KP481     KP781
Age
20         0.600000  0.400000  0.000000
21         0.571429  0.428571  0.000000
22         0.571429  0.000000  0.428571
23         0.444444  0.388889  0.166667
24         0.416667  0.250000  0.333333
25         0.280000  0.440000  0.280000
26         0.583333  0.250000  0.166667
27         0.428571  0.142857  0.428571
28         0.666667  0.000000  0.333333
29         0.500000  0.166667  0.333333
30         0.285714  0.285714  0.428571
31         0.333333  0.500000  0.166667
32         0.500000  0.500000  0.000000
33         0.250000  0.625000  0.125000
34         0.333333  0.500000  0.166667
35         0.375000  0.500000  0.125000
36         1.000000  0.000000  0.000000
37         0.500000  0.500000  0.000000
38         0.571429  0.285714  0.142857
39         1.000000  0.000000  0.000000
40         0.200000  0.600000  0.200000
41         1.000000  0.000000  0.000000
42         0.000000  0.000000  1.000000
43         0.500000  0.200000  0.300000
```




```
✍️ Probabilities based on Education:
Product        KP281     KP481     KP781
```

✍️ Insights:

1) Probabilities Based on Age:

- KP281: Popular among a wide age range, especially ages 20-22, 26, and 28.
- KP481: Preferred by users in their early 20s and around age 25.
- KP781: Attracts users in their late 20s and early 30s, with a notable peak at age 42.

2) Probabilities Based on Education:

- KP281: Most users have 14 or 16 years of education.
- KP481: Similar to KP281, with a majority having 14 or 16 years of education.
- KP781: Predominantly chosen by users with 18 years of education.

3) Probabilities Based on Usage:

- KP281: Most users have a usage level of 2 or 3.
- KP481: Similar to KP281, with a majority at usage level 3.
- KP781: Users are more likely to have higher usage levels, especially at level 5.

4) Probabilities Based on Income:

- KP281: Users are spread across various income levels, with some peaks at 45480,46617, and $54576.
- KP481: Users are also spread across income levels, with peaks at 45480*and*50028.

- KP781: Users tend to have higher incomes, with peaks at $90886 and$92131.

5) Probabilities Based on Fitness:

- KP281: Most users have a fitness level of 3.
- KP481: Similar to KP281, with a majority at fitness level 3.
- KP781: Users are more likely to have a fitness level of 5.

6) Probabilities Based on Miles:

- KP281: Users are spread across various mileage levels, with peaks at 85 and 94 miles.
- KP481: Users have peaks at 85 and 95 miles.
- KP781: Users tend to have higher mileage, with peaks at 100, 200 and 180 miles.

7) Probabilities Based on Gender:

- KP281: Slightly more popular among females (52.63%) than males (38.46%).
- KP481: Also more popular among females (38.16%) than males (29.81%).
- KP781: Significantly more popular among males (31.73%) compared to females (9.21%).

8) Probabilities Based on Marital Status:

- KP281: Equally popular among partnered (44.86%) and single (43.84%) users.
- KP481: Similar distribution among partnered (33.64%) and single (32.88%) users.
- KP781: Slightly more popular among single users (23.29%) compared to partnered users (21.50%).

c) Find the conditional probability that an event occurs given that another event has occurred. (Example: given that a customer is female, what is the probability she'll purchase a KP481)

```
#Example 1: Given that a customer is female, what is the probability he'll purchase a KP481?

# Create a crosstab for Gender and Product
gender_product_crosstab = pd.crosstab(df['Gender'], df['Product'], normalize='index')

# Display the crosstab

print(gender_product_crosstab)

# Calculate the conditional probability
prob_female_kp481 = gender_product_crosstab.loc['Female', 'KP481']
print(f" ✍ Given that a customer is female, the probability she'll purchase a KP481 is {prob_female_kp481:.2f}")
```

```
Product     KP281     KP481     KP781
Gender
Female   0.526316  0.381579  0.092105
Male     0.384615  0.298077  0.317308
 ✍ Given that a customer is female, the probability she'll purchase a KP481 is 0.38
```

```
#Example 2: Given that a customer is male, what is the probability he'll purchase a KP781?

# Create a crosstab for Gender and Product
gender_product_crosstab = pd.crosstab(df['Gender'], df['Product'], normalize='index')

# Display the crosstab
print(gender_product_crosstab)

# Calculate the conditional probability
prob_male_kp781 = gender_product_crosstab.loc['Male', 'KP781']
print(f" ✍ Given that a customer is male, the probability he'll purchase a KP781 is {prob_male_kp781:.2f}")
```

```
Product     KP281     KP481     KP781
Gender
Female   0.526316  0.381579  0.092105
Male     0.384615  0.298077  0.317308
 ✍ Given that a customer is male, the probability he'll purchase a KP781 is 0.32
```

```
#Example 3: Given that a customer is partnered, what is the probability they'll purchase a KP281?

# Create a crosstab for Marital Status and Product
marital_product_crosstab = pd.crosstab(df['MaritalStatus'], df['Product'], normalize='index')
```

```
# Display the crosstab
print(marital_product_crosstab)

# Calculate the conditional probability
prob_partnered_kp281 = marital_product_crosstab.loc['Partnered', 'KP281']
print(f" ✍️ Given that a customer is partnered, the probability they'll purchase a KP281 is {prob_partnered_kp281:.2f}")
```

```
Product          KP281     KP481     KP781
MaritalStatus
Partnered      0.448598  0.336449  0.214953
Single         0.438356  0.328767  0.232877
 ✍️ Given that a customer is partnered, the probability they'll purchase a KP281 is 0.45
```


```
#Example 4: Given that a customer has a fitness level of 5, what is the probability they'll purchase a KP781?

# Create a crosstab for Fitness and Product
fitness_product_crosstab = pd.crosstab(df['Fitness'], df['Product'], normalize='index')

# Display the crosstab
print(fitness_product_crosstab)

# Calculate the conditional probability
prob_fitness5_kp781 = fitness_product_crosstab.loc[5, 'KP781']
print(f" ✍️ Given that a customer has a fitness level of 5, the probability they'll purchase a KP781 is {prob_fitness5_kp781:.2f}")
```

```
Product     KP281     KP481     KP781
Fitness
2        0.535714  0.464286  0.000000
3        0.556701  0.402062  0.041237
4        0.375000  0.333333  0.291667
5        0.064516  0.000000  0.935484
 ✍️ Given that a customer has a fitness level of 5, the probability they'll purchase a KP781 is 0.94
```

Probability of Purchasing Each Product Based on Fitness

```
#Example 5: Given that a customer uses the treadmill 4 times per week, what is the probability they'll purchase a KP481?

# Create a crosstab for Usage and Product
usage_product_crosstab = pd.crosstab(df['Usage'], df['Product'], normalize='index')

# Display the crosstab
print(usage_product_crosstab)

# Calculate the conditional probability
prob_usage4_kp481 = usage_product_crosstab.loc[4, 'KP481']
print(f" ✍️ Given that a customer uses the treadmill 4 times per week, the probability they'll purchase a KP481 is {prob_usage4_kp481:.
```

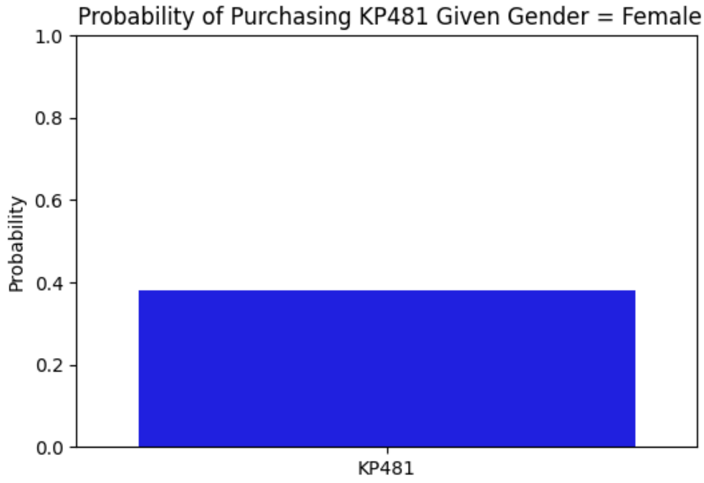
```
Product     KP281     KP481     KP781
Usage
2        0.575758  0.424242  0.000000
3        0.536232  0.449275  0.014493
4        0.423077  0.230769  0.346154
5        0.076923  0.115385  0.807692
 ✍️ Given that a customer uses the treadmill 4 times per week, the probability they'll purchase a KP481 is 0.23
```

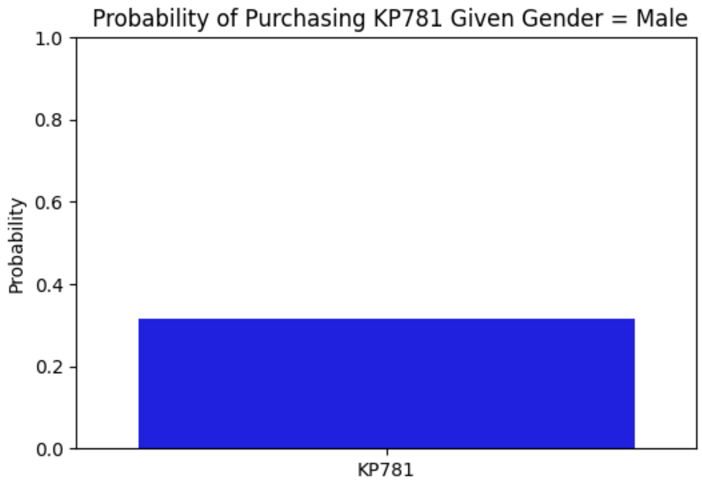

```
#Example 6: Given that a customer has an income of Rs.34110, what is the probability they'll purchase a KP281?

# Create a crosstab for Income and Product
income_product_crosstab = pd.crosstab(df['Income'], df['Product'], normalize='index')

# Display the crosstab
print(income_product_crosstab)

# Calculate the conditional probability
prob_34110_kp281 = income_product_crosstab.loc[34110, 'KP281']
print(f" ✍️ Given that a customer has an income of Rs 34110, the probability they'll purchase a KP281 is {prob_34110_kp281:.2f}")
```

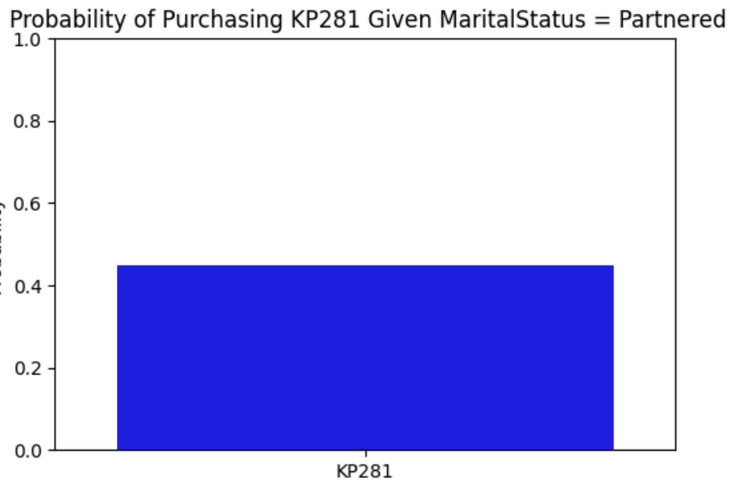
```
Product     KP281     KP481  KP781
Income
34053    0.666667  0.333333    0.0
34110    0.400000  0.600000    0.0
35247    1.000000  0.000000    0.0
36384    0.750000  0.250000    0.0
37521    1.000000  0.000000    0.0
38658    0.600000  0.400000    0.0
39795    1.000000  0.000000    0.0
40932    0.666667  0.333333    0.0
42069    1.000000  0.000000    0.0
43206    0.200000  0.800000    0.0
44343    1.000000  0.000000    0.0
```

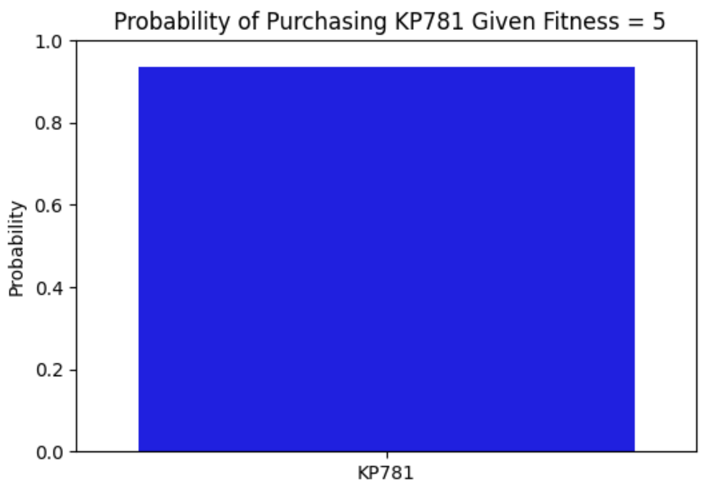
```
45480    0.357143  0.642857    0.0
46617    0.875000  0.125000    0.0
47754    0.000000  1.000000    0.0
48556    0.000000  0.000000    1.0
48658    0.000000  0.000000    1.0
48891    0.400000  0.600000    0.0
49801    0.000000  0.000000    1.0
50028    0.285714  0.714286    0.0
51165    0.428571  0.571429    0.0
52290    0.000000  0.000000    1.0
52291    0.000000  0.000000    1.0
52302    0.666667  0.333333    0.0
53439    0.375000  0.625000    0.0
53536    0.000000  0.000000    1.0
54576    0.875000  0.125000    0.0
54781    0.000000  0.000000    1.0
55713    1.000000  0.000000    0.0
56850    1.000000  0.000000    0.0
57271    0.000000  0.000000    1.0
57987    0.250000  0.750000    0.0
58516    0.000000  0.000000    1.0
59124    0.333333  0.666667    0.0
60261    0.666667  0.333333    0.0
61006    0.000000  0.000000    1.0
61398    0.500000  0.500000    0.0
62251    0.000000  0.000000    1.0
62535    0.000000  1.000000    0.0
64741    0.000000  0.000000    1.0
64809    0.333333  0.666667    0.0
65220    0.000000  1.000000    0.0
67083    0.500000  0.500000    0.0
68220    1.000000  0.000000    0.0
69721    0.000000  0.000000    1.0
70966    0.000000  0.000000    1.0
74701    0.000000  0.000000    1.0
75946    0.000000  0.000000    1.0
77191    0.000000  0.000000    1.0
83416    0.000000  0.000000    1.0
85906    0.000000  0.000000    1.0
88396    0.000000  0.000000    1.0
89641    0.000000  0.000000    1.0
90886    0.000000  0.000000    1.0
90948    0.000000  0.000000    1.0
✍ Given that a customer has an income of Rs 34110, the probability they'll purchase a KP281 is 0.40
```

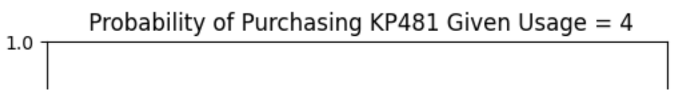

```
# Function to calculate conditional probability
def calculate_conditional_probability(df, column, value, product):
    crosstab = pd.crosstab(df[column], df['Product'], normalize='index')
    if value in crosstab.index:
        probability = crosstab.loc[value, product]
        return probability
    else:
        return None

# Function to plot conditional probability
def plot_conditional_probability(df, column, value, product):
    prob = calculate_conditional_probability(df, column, value, product)
    if prob is not None:
        plt.figure(figsize=(6, 4))
        sns.barplot(x=[product], y=[prob], color='blue')
        plt.title(f' Probability of Purchasing {product} Given {column} = {value}')
        plt.ylabel('Probability')
        plt.ylim(0, 1)
        plt.show()
    else:
        print(f"{value} not found in the dataset for {column}.")

# Example 1: Given that a customer is female, what is the probability she'll purchase a KP481?
plot_conditional_probability(df, 'Gender', 'Female', 'KP481')

# Example 2: Given that a customer is male, what is the probability he'll purchase a KP781?
plot_conditional_probability(df, 'Gender', 'Male', 'KP781')

# Example 3: Given that a customer is partnered, what is the probability they'll purchase a KP281?
plot_conditional_probability(df, 'MaritalStatus', 'Partnered', 'KP281')

# Example 4: Given that a customer has a fitness level of 5, what is the probability they'll purchase a KP781?
plot_conditional_probability(df, 'Fitness', 5, 'KP781')

# Example 5: Given that a customer uses the treadmill 4 times per week, what is the probability they'll purchase a KP481?
plot_conditional_probability(df, 'Usage', 4, 'KP481')

# Example 6: Given that a customer has an income of ₹50,000, what is the probability they'll purchase a KP281?
plot_conditional_probability(df, 'Income', 50000, 'KP281')
```

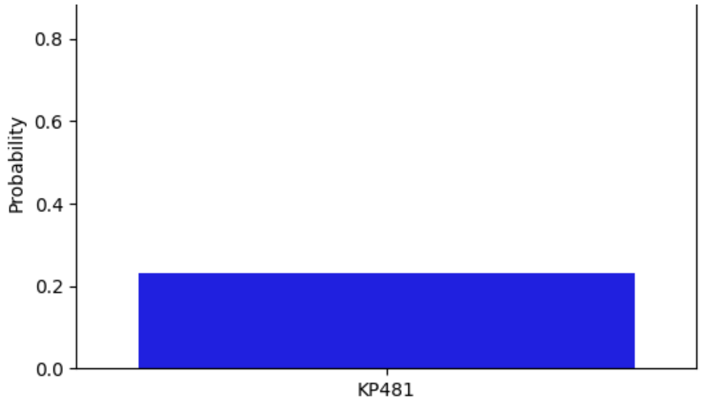

Probability of Purchasing KP481 Given Gender = Female
Probability
KP481


Probability of Purchasing KP781 Given Gender = Male
Probability
KP781


Probability of Purchasing KP281 Given MaritalStatus = Partnered
Probability
KP281


Probability of Purchasing KP781 Given Fitness = 5
Probability
KP781


Probability of Purchasing KP481 Given Usage = 4



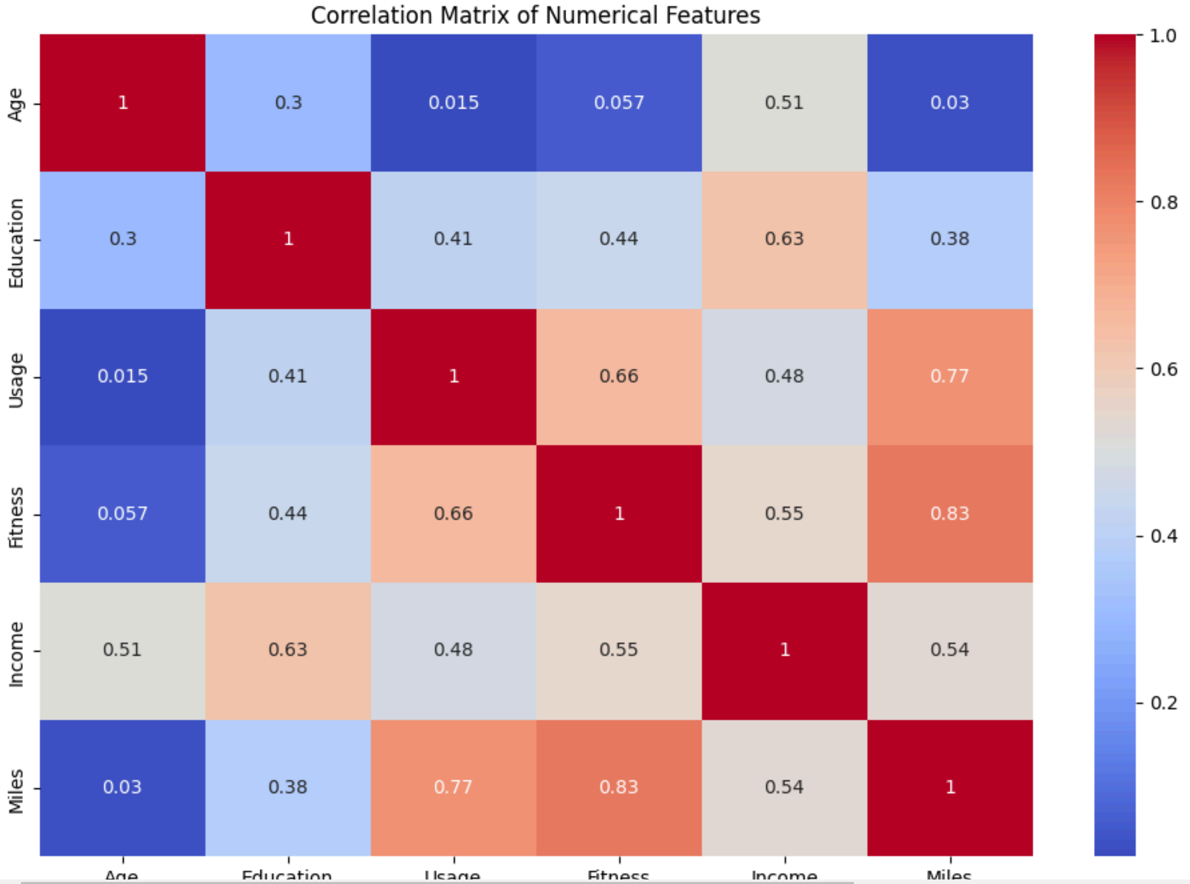

```
50000 not found in the dataset for Income.
```

5. Check the correlation among different factors

a)Find the correlation between the given features in the table.

```
# Select only numerical columns for correlation calculation
numerical_columns = df.select_dtypes(include=['float64', 'int64']).columns

# Calculate the correlation matrix for numerical columns
correlation_matrix = df[numerical_columns].corr()

# Display the correlation matrix
print(correlation_matrix)

# Plot the heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', cbar=True)
plt.title('Correlation Matrix of Numerical Features')
plt.show()
```

```
                Age  Education     Usage   Fitness    Income     Miles
Age        1.000000   0.301984  0.015218  0.057314  0.514407  0.029719
Education  0.301984   1.000000  0.412484  0.441082  0.628597  0.377294
Usage      0.015218   0.412484  1.000000  0.660556  0.478615  0.769234
Fitness    0.057314   0.441082  0.660556  1.000000  0.546997  0.826307
Income     0.514407   0.628597  0.478615  0.546997  1.000000  0.537296
Miles      0.029719   0.377294  0.769234  0.826307  0.537296  1.000000
```



✍️ Insights:

1) Age:

- Income: There is a moderate positive correlation (0.514) between age and income, indicating that older customers tend to have higher incomes.
- Education: There is a weak positive correlation (0.302) between age and education, suggesting that older customers might have slightly higher education levels.
- Other Factors: Age has very weak correlations with usage, fitness, and miles, indicating that age does not significantly influence these factors.

2) Education:

- Income: There is a strong positive correlation (0.629) between education and income, indicating that higher education levels are associated with higher incomes.
- Fitness: There is a moderate positive correlation (0.441) between education and fitness, suggesting that more educated customers might have better fitness levels.
- Usage: There is a moderate positive correlation (0.412) between education and usage, indicating that more educated customers might use the treadmill more frequently.
- Miles: There is a weak positive correlation (0.377) between education and miles, suggesting that more educated customers might cover more miles.

3) Usage:

- Miles: There is a strong positive correlation (0.769) between usage and miles, indicating that customers who use the treadmill more frequently tend to cover more miles.
- Fitness: There is a moderate positive correlation (0.661) between usage and fitness, suggesting that more frequent usage is associated with better fitness levels.

- Income: There is a moderate positive correlation (0.479) between usage and income, indicating that higher-income customers might use the treadmill more frequently.

4) Fitness:

- Miles: There is a strong positive correlation (0.826) between fitness and miles, indicating that customers with better fitness levels tend to cover more miles.
- Income: There is a moderate positive correlation (0.547) between fitness and income, suggesting that higher-income customers might have better fitness levels.

5) Income:

- Miles: There is a moderate positive correlation (0.537) between income and miles, indicating that higher-income customers might cover more miles.

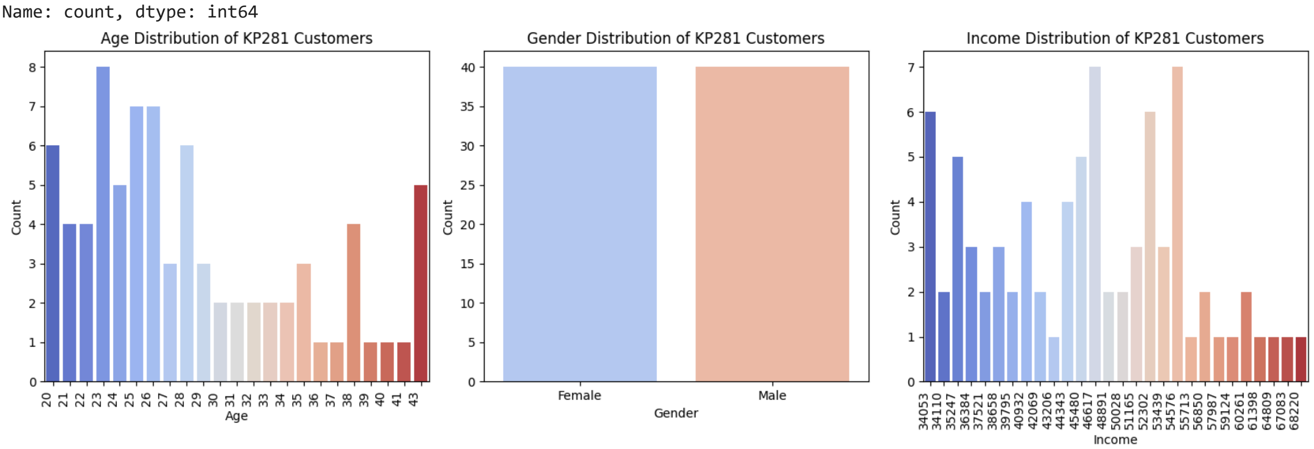
6. Customer profiling and recommendation

- Make customer profilings for each and every product.

a) Customer Profiling for KP281

```
# Filter the dataset for customers who purchased KP281
kp281_customers = df[df['Product'] == 'KP281']

# Calculate the distribution of age, gender, and income for KP281 customers
age_distribution_kp281 = kp281_customers['Age'].value_counts().sort_index()
gender_distribution_kp281 = kp281_customers['Gender'].value_counts()
income_distribution_kp281 = kp281_customers['Income'].value_counts().sort_index()


# Print the distributions
print("Age Distribution of KP281 Customers:")
print(age_distribution_kp281)

print("\nGender Distribution of KP281 Customers:")
print(gender_distribution_kp281)

print("\nIncome Distribution of KP281 Customers:")
print(income_distribution_kp281)

# Plot the distributions
plt.figure(figsize=(15, 5))

# Age Distribution
plt.subplot(1, 3, 1)
sns.barplot(x=age_distribution_kp281.index, y=age_distribution_kp281.values, palette='coolwarm')
plt.title('Age Distribution of KP281 Customers')
plt.xlabel('Age')
plt.xticks(rotation=90, ha='right')
plt.ylabel('Count')

# Gender Distribution
plt.subplot(1, 3, 2)
sns.barplot(x=gender_distribution_kp281.index, y=gender_distribution_kp281.values, palette='coolwarm')
plt.title('Gender Distribution of KP281 Customers')
plt.xlabel('Gender')
plt.ylabel('Count')

# Income Distribution
plt.subplot(1, 3, 3)
sns.barplot(x=income_distribution_kp281.index, y=income_distribution_kp281.values, palette='coolwarm')
plt.title('Income Distribution of KP281 Customers')
plt.xlabel('Income')
plt.xticks(rotation=90, ha='right')
plt.ylabel('Count')

plt.tight_layout()
plt.show()
```

```
Age Distribution of KP281 Customers:
Age
20    6
21    4
22    4
23    8
24    5
25    7
26    7
27    3
28    6
29    3
30    2
31    2
32    2
33    2
34    2
35    3
36    1
37    1
38    4
39    1
40    1
41    1
43    5
Name: count, dtype: int64

Gender Distribution of KP281 Customers:
Gender
Female    40
Male      40
Name: count, dtype: int64

Income Distribution of KP281 Customers:
Income
34053    6
34110    2
35247    5
36384    3
37521    2
38658    3
39795    2
40932    4
42069    2
43206    1
44343    4
45480    5
46617    7
48891    2
50028    2
51165    3
52302    6
53439    3
54576    7
55713    1
56850    2
57987    1
59124    1
60261    2
61398    1
64809    1
67083    1
68220    1
Name: count, dtype: int64
```



✍️ Insights:

1) Age Distribution

- Most Common Age Groups: The most common age groups for KP281 customers are 23, 25, and 26 years old, with 8, 7, and 7 customers respectively.
- Age Range: KP281 customers range from 20 to 43 years old, with a slight concentration in the mid-20s.

2) Gender Distribution

- Equal Distribution: The gender distribution is equal, with 40 male and 40 female customers purchasing KP281.

3) Income Distribution

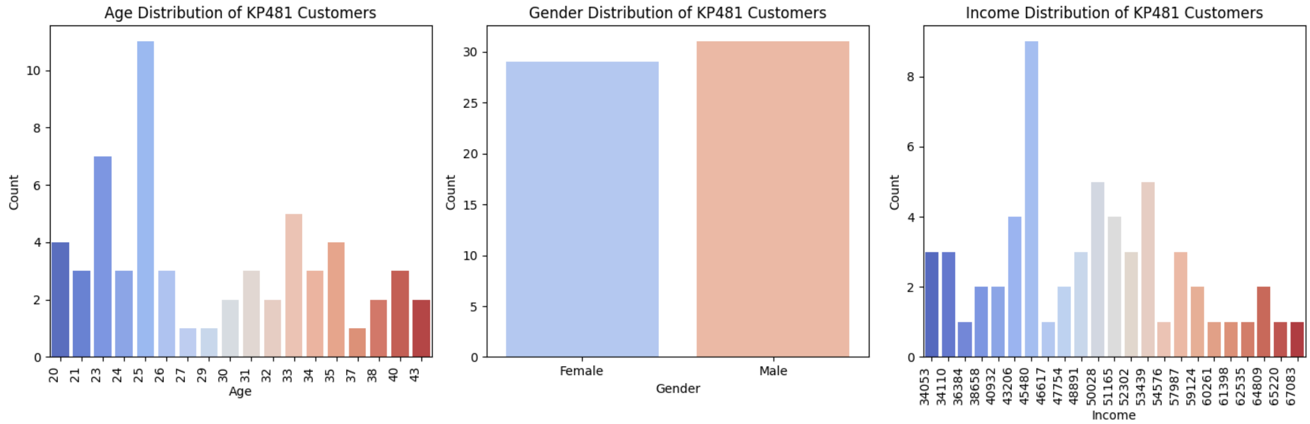
- Most Common Income Groups: The most common income groups for KP281 customers are ₹34,110, ₹46,617, and ₹54,576, with 6, 7, and 7 customers respectively.
- Income Range: KP281 customers have a wide range of incomes, from ₹34,110 to ₹68,220.

These insights indicate that KP281 is popular among young adults in their mid-20s, with an equal distribution of male and female customers. The product appeals to a wide range of income groups, with a slight preference for those in the lower to middle-income brackets.

2)Customer Profiling for KP481

```
# Filter the dataset for customers who purchased KP481
kp481_customers = df[df['Product'] == 'KP481']

# Calculate the distribution of age, gender, and income for KP481 customers
age_distribution_kp481 = kp481_customers['Age'].value_counts().sort_index()
gender_distribution_kp481 = kp481_customers['Gender'].value_counts()
income_distribution_kp481 = kp481_customers['Income'].value_counts().sort_index()

# Print the distributions
print("Age Distribution of KP481 Customers:")
print(age_distribution_kp481)

print("\nGender Distribution of KP481 Customers:")
print(gender_distribution_kp481)

print("\nIncome Distribution of KP481 Customers:")
print(income_distribution_kp481)

# Plot the distributions
plt.figure(figsize=(15, 5))

# Age Distribution
plt.subplot(1, 3, 1)
sns.barplot(x=age_distribution_kp481.index, y=age_distribution_kp481.values, palette='coolwarm')
plt.title('Age Distribution of KP481 Customers')
plt.xlabel('Age')
plt.xticks(rotation=90, ha='right')
plt.ylabel('Count')

# Gender Distribution
plt.subplot(1, 3, 2)
sns.barplot(x=gender_distribution_kp481.index, y=gender_distribution_kp481.values, palette='coolwarm')
plt.title('Gender Distribution of KP481 Customers')
plt.xlabel('Gender')
plt.ylabel('Count')

# Income Distribution
plt.subplot(1, 3, 3)
sns.barplot(x=income_distribution_kp481.index, y=income_distribution_kp481.values, palette='coolwarm')
plt.title('Income Distribution of KP481 Customers')
plt.xlabel('Income')
plt.xticks(rotation=90, ha='right')
plt.ylabel('Count')

plt.tight_layout()
plt.show()
```

```
Age Distribution of KP481 Customers:
Age
20     4
21     3
23     7
24     3
25    11
26     3
27     1
29     1
30     2
31     3
32     2
33     5
34     3
35     4
37     1
38     2
40     3
43     2
Name: count, dtype: int64

Gender Distribution of KP481 Customers:
Gender
Male      31
Female    29
Name: count, dtype: int64

Income Distribution of KP481 Customers:
Income
34053    3
34110    3
36384    1
38658    2
40932    2
43206    4
45480    9
46617    1
47754    2
48891    3
50028    5
51165    4
52302    3
53439    5
54576    1
57987    3
59124    2
60261    1
61398    1
62535    1
64809    2
65220    1
67083    1
Name: count, dtype: int64
```



✍️ Insights:

1) Age Distribution

- Most Common Age Groups: The most common age groups for KP481 customers are 25 years old (11 customers) and 23 years old (7 customers).
- Age Range: KP481 customers range from 20 to 43 years old, with a concentration in the mid-20s.

2) Gender Distribution

- Slight Male Dominance: The gender distribution shows a slight male dominance, with 31 male customers and 29 female customers purchasing KP481.

3) Income Distribution

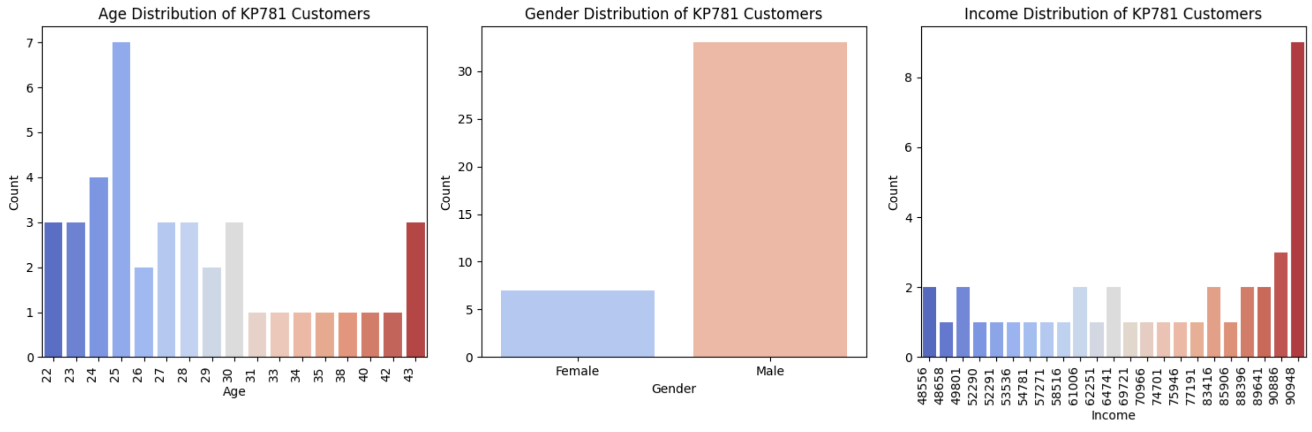
- Most Common Income Groups: The most common income groups for KP481 customers are 53,439 (5 customers), ₹45,480 (9 customers), and ₹50,028 (5 customers).
- Income Range: KP481 customers have a wide range of incomes, from ₹34,053 to ₹67,083.

These insights indicate that KP481 is popular among young adults in their mid-20s, with a slight male dominance. The product appeals to a wide range of income groups, with a slight preference for those in the lower to middle-income brackets.

3) Customer Profiling for KP781

```
# Filter the dataset for customers who purchased KP781
kp781_customers = df[df['Product'] == 'KP781']

# Calculate the distribution of age, gender, and income for KP781 customers
age_distribution_kp781 = kp781_customers['Age'].value_counts().sort_index()
gender_distribution_kp781 = kp781_customers['Gender'].value_counts()
income_distribution_kp781 = kp781_customers['Income'].value_counts().sort_index()


# Print the distributions
print("Age Distribution of KP781 Customers:")
print(age_distribution_kp781)

print("\nGender Distribution of KP781 Customers:")
print(gender_distribution_kp781)

print("\nIncome Distribution of KP781 Customers:")
print(income_distribution_kp781)

# Plot the distributions
plt.figure(figsize=(15, 5))

# Age Distribution
plt.subplot(1, 3, 1)
sns.barplot(x=age_distribution_kp781.index, y=age_distribution_kp781.values, palette='coolwarm')
plt.title('Age Distribution of KP781 Customers')
plt.xlabel('Age')
plt.xticks(rotation=90, ha='right')
plt.ylabel('Count')

# Gender Distribution
plt.subplot(1, 3, 2)
sns.barplot(x=gender_distribution_kp781.index, y=gender_distribution_kp781.values, palette='coolwarm')
plt.title('Gender Distribution of KP781 Customers')
plt.xlabel('Gender')
plt.ylabel('Count')

# Income Distribution
plt.subplot(1, 3, 3)
sns.barplot(x=income_distribution_kp781.index, y=income_distribution_kp781.values, palette='coolwarm')
plt.title('Income Distribution of KP781 Customers')
plt.xlabel('Income')
plt.xticks(rotation=90, ha='right')
plt.ylabel('Count')

plt.tight_layout()
plt.show()
```

```
Age Distribution of KP781 Customers:
Age
22    3
23    3
24    4
25    7
26    2
27    3
28    3
29    2
30    3
31    1
33    1
34    1
35    1
38    1
40    1
42    1
43    3
Name: count, dtype: int64

Gender Distribution of KP781 Customers:
Gender
Male      33
Female     7
Name: count, dtype: int64

Income Distribution of KP781 Customers:
Income
48556    2
48658    1
49801    2
52290    1
52291    1
53536    1
54781    1
57271    1
58516    1
61006    2
62251    1
64741    2
69721    1
70966    1
74701    1
75946    1
77191    1
83416    2
85906    1
88396    2
89641    2
90886    3
90948    9
Name: count, dtype: int64
```



✍️ Insights:

1) Age Distribution

- Most Common Age Groups: The most common age group for KP781 customers is 25 years old, with 7 customers.
- Age Range: KP781 customers range from 22 to 43 years old, with a concentration in the mid-20s.

2) Gender Distribution

- Male Dominance: The gender distribution shows a strong male dominance, with 33 male customers and only 7 female customers purchasing KP781.

3) Income Distribution

- Most Common Income Groups: The most common income group for KP781 customers is ₹90,886, with 9 customers. Other notable income groups include ₹48,556, ₹49,801, ₹61,006, ₹64,741, ₹83,416, ₹88,396, and ₹90,886.
- Income Range: KP781 customers have a wide range of incomes, from ₹48,556 to ₹90,948.

These insights indicate that KP781 is popular among young adults in their mid-20s, with a strong male dominance. The product appeals to a

✍️ Detailed Recommendation Based on Customer Profiling Analysis:

Based on the customer profiling analysis for the products KP281, KP481, and KP781, we can derive several actionable recommendations to enhance marketing strategies, product development, and customer engagement.

1. KP281 Customer Profile and Recommendations

`Customer Profile:`

- Age Group: Predominantly young adults in their mid-20s, with the most common age groups being 23, 25, and 26 years old.
- Gender: Equal distribution between male and female customers.
- Income Group: Wide range of incomes, with a slight preference for lower to middle-income brackets (₹34,110 to ₹68,220).

`Recommendations:`

- Targeted Marketing Campaigns: Develop marketing campaigns that resonate with young adults in their mid-20s. Use social media platforms like Instagram and TikTok, which are popular among this age group.
- Gender-Inclusive Advertising: Ensure that advertising campaigns are gender-inclusive, highlighting features that appeal to both male and female customers.
- Affordable Pricing and Promotions: Offer promotions and discounts to attract customers in the lower to middle-income brackets. Consider bundling KP281 with complementary products to provide added value.

2. KP481 Customer Profile and Recommendations

`Customer Profile:`

- Age Group: Predominantly young adults in their mid-20s, with the most common age groups being 25 and 23 years old.
- Gender: Slight male dominance, with 31 male customers and 29 female customers.
- Income Group: Wide range of incomes, with a slight preference for lower to middle-income brackets (₹34,053 to ₹67,083).

`Recommendations:`

- Male-Focused Campaigns: Develop marketing campaigns that specifically target male customers, highlighting features that appeal to them.

- Social Media and Influencer Marketing: Utilize social media and influencer marketing to reach young adults in their mid-20s. Collaborate with influencers who have a strong following among this demographic.
- Flexible Payment Options: Offer flexible payment options, such as installment plans, to make KP481 more accessible to customers in the lower to middle-income brackets.

3. KP781 Customer Profile and Recommendations

`Customer Profile:`

- Age Group: Predominantly young adults in their mid-20s, with the most common age group being 25 years old.
- Gender: Strong male dominance, with 33 male customers and only 7 female customers.
- Income Group: Wide range of incomes, with a slight preference for higher-income brackets (₹48,556 to ₹90,948).

`Recommendations:`

- Premium Positioning: Position KP781 as a premium product, highlighting its advanced features and benefits. Emphasize the value proposition to justify the higher price point.
- Male-Centric Advertising: Develop advertising campaigns that specifically target male customers, using imagery and messaging that resonate with them.
- High-Income Targeting: Focus marketing efforts on high-income customers, offering exclusive promotions and personalized experiences to attract this segment.

📝 Overall Recommendations :

- Customer Segmentation: Segment customers based on age, gender, and income to create personalized marketing strategies for each product. Use data analytics to identify key customer segments and tailor campaigns accordingly.
- Product Bundling: Consider bundling products with complementary items to increase perceived value and encourage cross-selling. For example, bundle KP281 with fitness accessories or KP481 with workout apparel.
- Loyalty Programs: Implement loyalty programs to reward repeat customers and encourage brand loyalty. Offer exclusive discounts, early access to new products, and personalized recommendations based on purchase history.
- Feedback and Improvement: Collect customer feedback to identify areas for improvement and enhance product offerings. Use surveys, reviews, and social media interactions to gather insights and make data-driven decisions.
- Data-Driven Insights: Continuously analyze customer data to identify trends and preferences. Use these insights to refine marketing strategies and product offerings.

- Sustainability Initiatives: Highlight any sustainability initiatives or eco-friendly features of your products. This can appeal to environmentally conscious customers and enhance your brand image.
- Omni-Channel Presence: Ensure a seamless customer experience across all channels, including online, in-store, and mobile. Provide consistent messaging and support to enhance customer satisfaction.
- Customer Support: Invest in robust customer support systems to address queries and issues promptly. Excellent customer service can lead to higher customer retention and positive word-of-mouth.
- Fitness Content: Develop and share fitness-related content such as workout videos, nutrition tips, and wellness articles. This can help in engaging customers and promoting a healthy lifestyle.
- Mileage Tracking App: Create a dedicated app for tracking mileage and fitness activities. The app can provide insights, set goals, and offer rewards based on users' performance.
- Collaborations with Fitness Influencers: Collaborate with fitness influencers to promote the products and share their fitness journeys using KP281, KP481, and KP781. This can increase brand visibility and credibility.
- Fitness Events: Host virtual or in-person fitness events, such as marathons, yoga sessions, or boot camps, to engage with customers and promote the products. Offer exclusive merchandise or discounts to participants.
- Partnerships with Educational Influencers: Collaborate with educational influencers and thought leaders to promote the products and share their knowledge. This can increase credibility and reach.
- Certification Programs: Develop certification programs that users can complete using the products. Offer badges or certificates upon completion to motivate and reward learners.
- Educational Apps: Develop dedicated educational apps that complement the products. These apps can offer features like course management, interactive learning, and progress tracking.