untitled6

March 23, 2024

[2]:
```
import pandas as pd
df = pd.read_csv("/bin/data/aerofit_treadmill.csv")
df
```

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

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

[180 rows x 9 columns]
```

1. Import the dataset and do usual data analysis steps like checking the structure & characteristics of the dataset   The data type of all columns in the “customers” table. Hint: We want you to display the data type of each column present in the dataset.   You can find the number of rows and columns given in the dataset Hint: We want you to find the shape of the dataset.   Check for the missing values and find the number of missing values in each column

data type of each column in data frame

```
[ ]: print("----data type of each column in data frame----")
     print(df.dtypes)
```

```
----data type of each column in data frame----
Product          object
Age               int64
Gender           object
Education         int64
MaritalStatus    object
Usage             int64
Fitness           int64
Income            int64
Miles             int64
dtype: object
```

```
[ ]:
```

Rows and column in data

```
[ ]: print("In the given data, we are having about "+ str(df.shape[0])+" rows and "+␣
       ↪str(df.shape[1])+ " columns")
```

```
In the given data, we are having about 180 rows and 9 columns
```

Missing values in each column

```
[ ]: missing_values = df.isnull().sum()
     print("\nNumber of missing values in each column:")
     print(missing_values)
```

```

Number of missing values in each column:
Product          0
Age              0
Gender           0
Education        0
MaritalStatus    0
Usage            0
Fitness          0
Income           0
Miles            0
dtype: int64
```

Insights: not seeing any column with missing value

2. Detect Outliers Find the outliers for every continuous variable in the dataset Hint: We want you to use boxplots to find the outliers in the given dataset Remove/clip the data between the 5 percentile and 95 percentile Hint: We want You to use np.clip() for clipping the data

Outliers for continous varibles

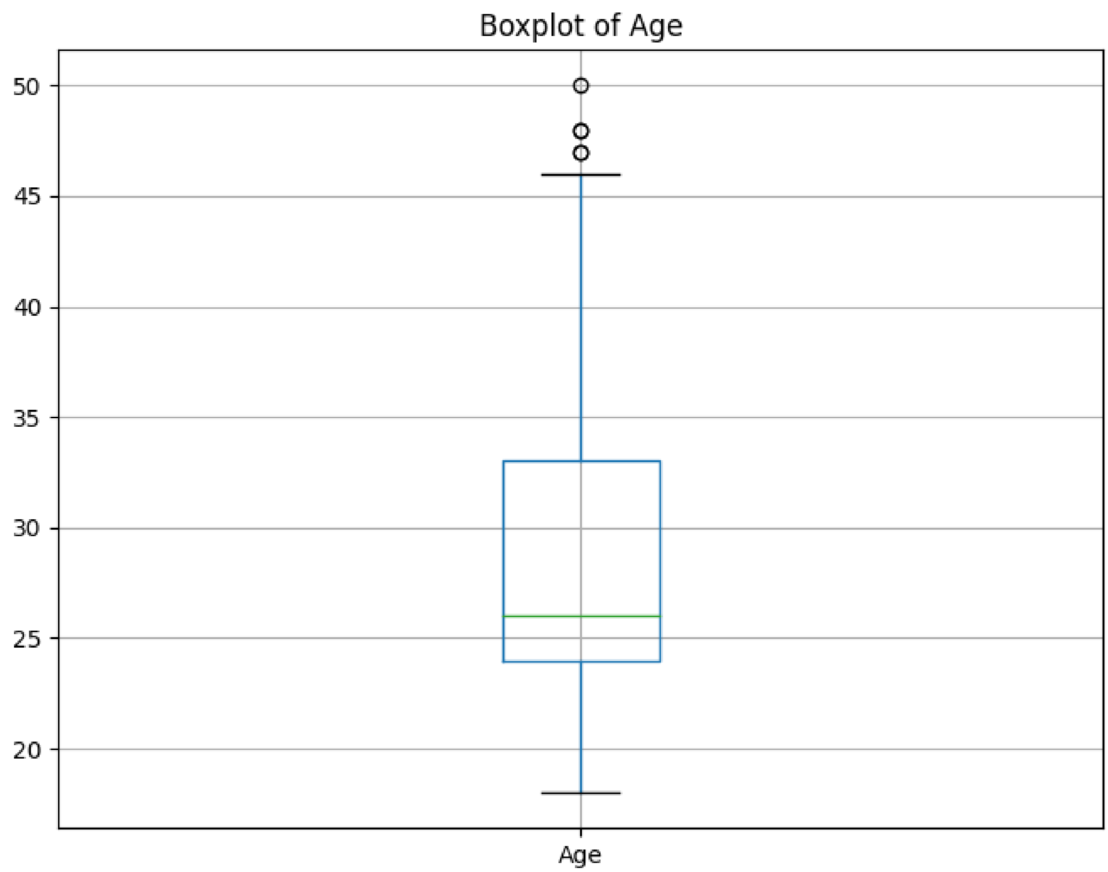
Initially , lets check the continous varibles from the data given

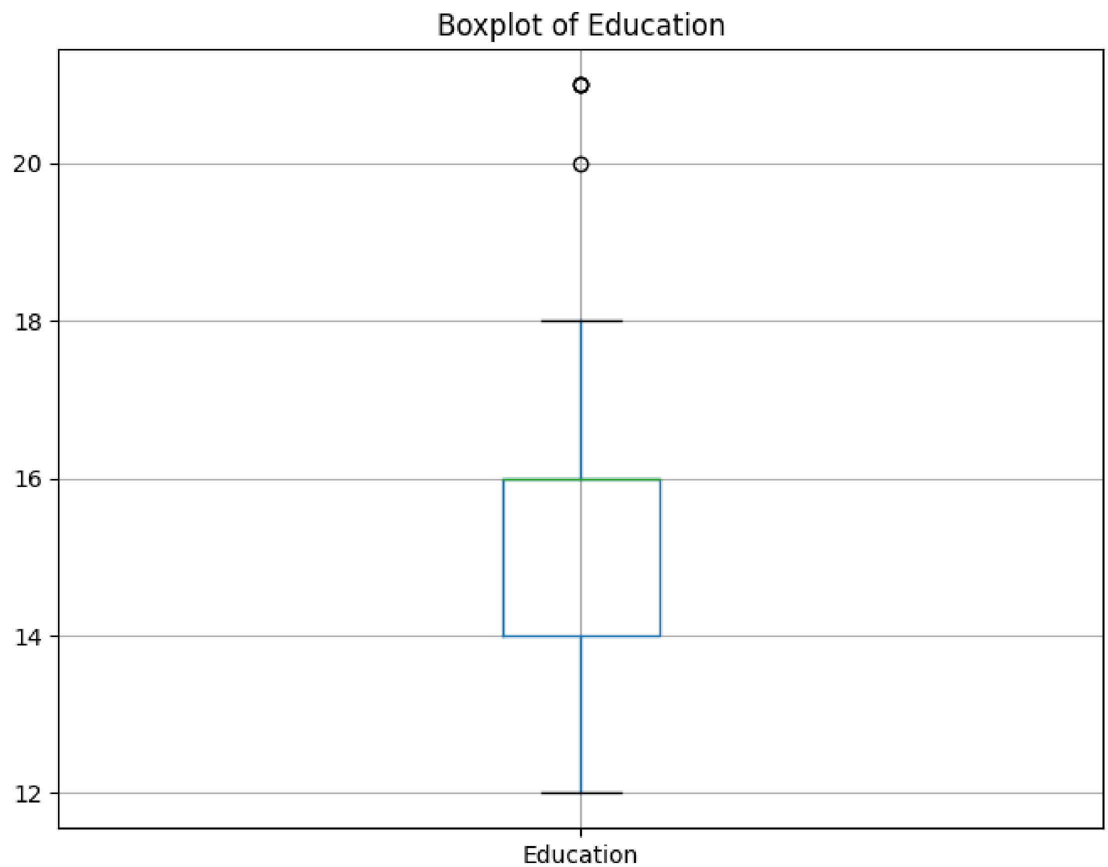
```
[ ]: continuous_vars = df.select_dtypes(include=['float64', 'int64']).columns
     print(continuous_vars)
```

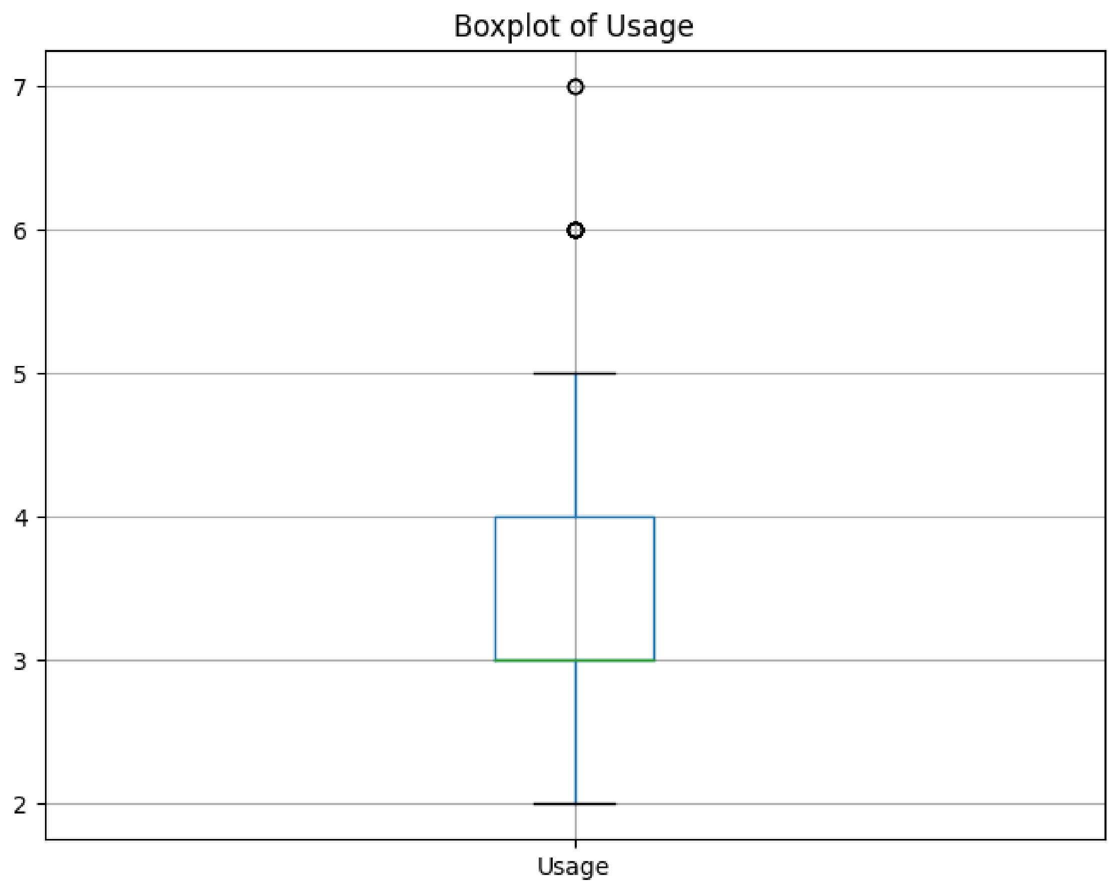
```
Index(['Age', 'Education', 'Usage', 'Fitness', 'Income', 'Miles'],
dtype='object')
```

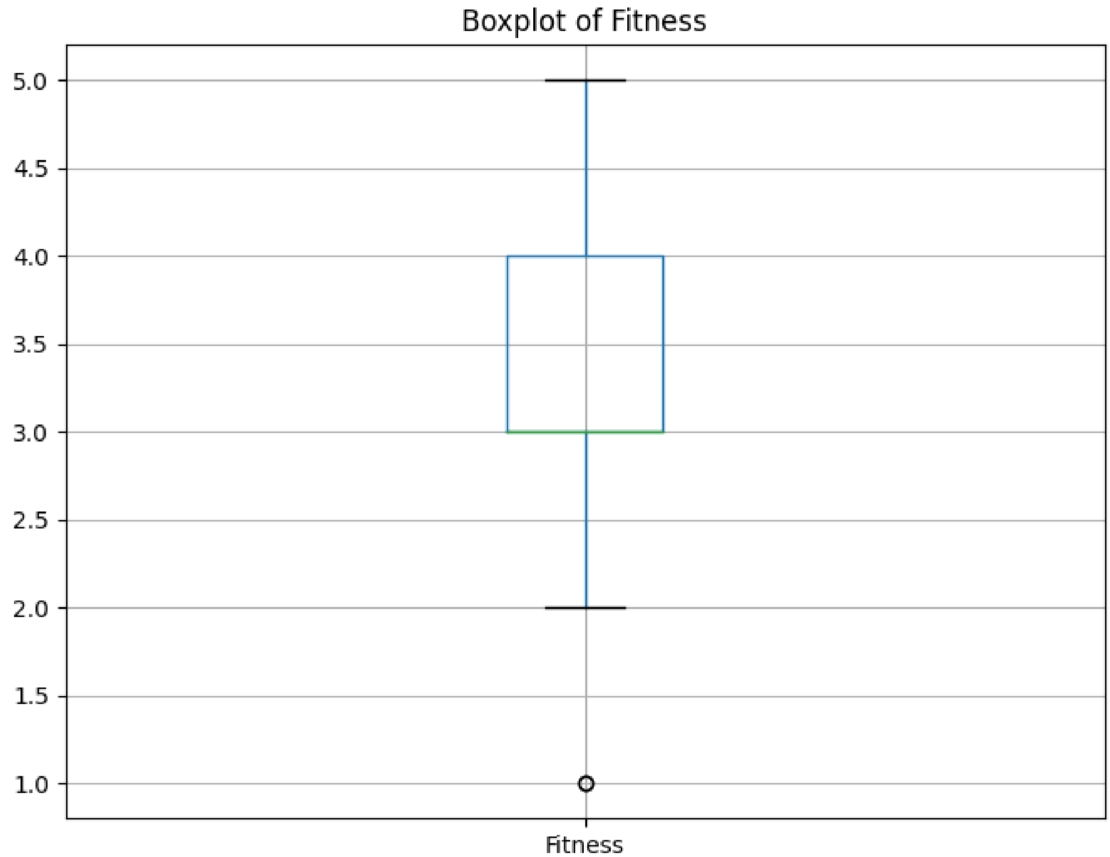
from the above continous variables, let see the outliers using boxplot

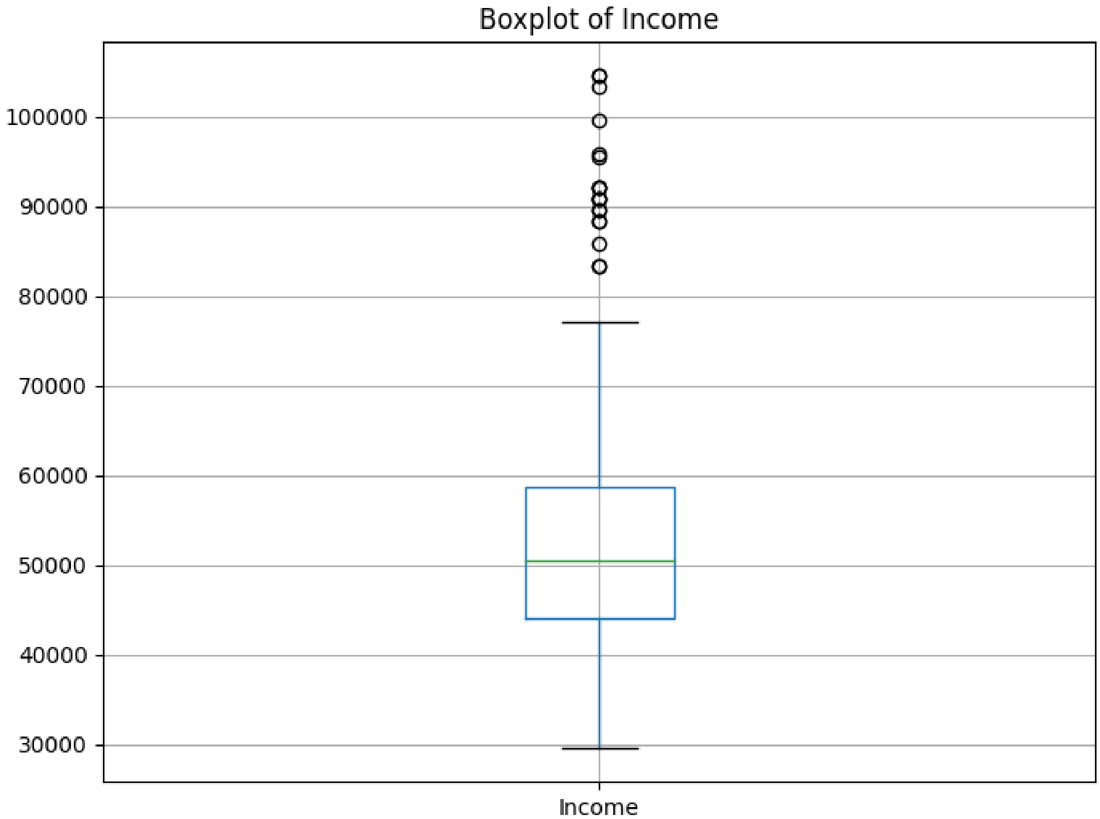
```
[ ]: import matplotlib.pyplot as plt
     for var in continuous_vars:
         plt.figure(figsize=(8, 6))
         df.boxplot(column=var)
         plt.title(f'Boxplot of {var}')
         plt.show()
```

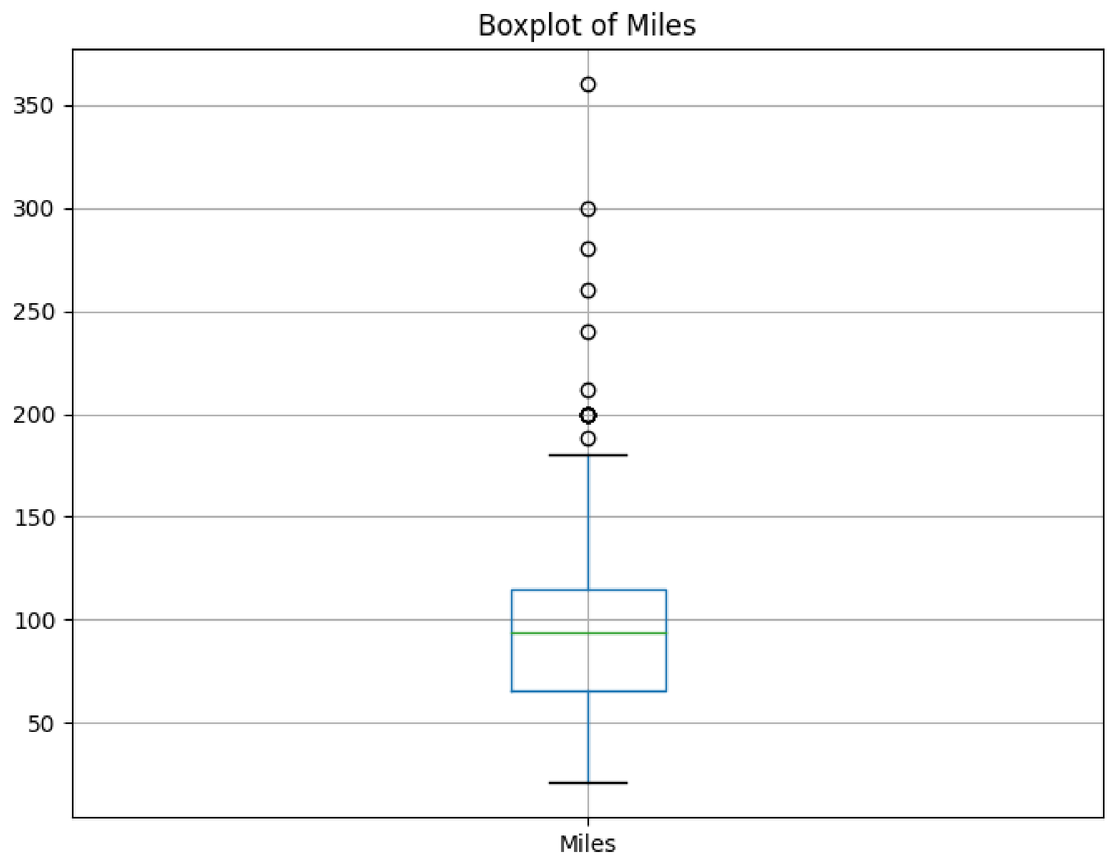



Boxplot of Education
20
18
16
14
12
Education


Boxplot of Usage
7
6
5
4
3
2
Usage


Boxplot of Fitness
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
Fitness


Boxplot of Income
100000
90000
80000
70000
60000
50000
40000
30000
Income




```
[ ]: print("These are the outliers we are seeing in above pictures, we can also␣
      ↪calculate the outliers by using IQR as well\n\n")
     outliers = {}
     for var in continuous_vars:
         Q1 = df[var].quantile(0.25)
         Q3 = df[var].quantile(0.75)
         IQR = Q3 - Q1
         lower_bound = Q1 - 1.5 * IQR
         upper_bound = Q3 + 1.5 * IQR
         var_outliers = df[(df[var] < lower_bound) | (df[var] > upper_bound)][var]
         outliers[var] = var_outliers.tolist()

     # Print outliers for each continuous variable
     for var, var_outliers in outliers.items():
         print(f"Outliers for {var}:")
         print(var_outliers)
         print()
```

```
These are the outliers we are seeing in above pictures, we can also calculate
the outliers by using IQR as well
```

```
Outliers for Age:
[47, 50, 48, 47, 48]

Outliers for Education:
[20, 21, 21, 21]

Outliers for Usage:
[6, 6, 6, 7, 6, 7, 6, 6, 6]

Outliers for Fitness:
[1, 1]

Outliers for Income:
[83416, 88396, 90886, 92131, 88396, 85906, 90886, 103336, 99601, 89641, 95866,
92131, 92131, 104581, 83416, 89641, 90886, 104581, 95508]

Outliers for Miles:
[188, 212, 200, 200, 200, 240, 300, 280, 260, 200, 360, 200, 200]
```

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

```
[ ]: import numpy as np
percentiles = df[continuous_vars].quantile([0.05, 0.95])

# Clip the data between the 5th and 95th percentiles
for var in continuous_vars:
    df[var] = np.clip(df[var], percentiles[var].iloc[0], percentiles[var].
 ↪iloc[1])

# Verify the clipped data
print("Clipped data:")
print(df.head())
```

```
Clipped data:
  Product   Age  Gender  Education MaritalStatus  Usage  Fitness    Income  \
0   KP281  20.0    Male         14        Single    3.0        4  34053.15
1   KP281  20.0    Male         15        Single    2.0        3  34053.15
2   KP281  20.0  Female         14     Partnered    4.0        3  34053.15
3   KP281  20.0    Male         14        Single    3.0        3  34053.15
4   KP281  20.0    Male         14     Partnered    4.0        2  35247.00

   Miles
0    112
1     75
2     66
```

```
3       85
4       47
```

** Insights** here we are collecting the outliers initially in each column and replacing them using clip in pandas. So , we are restricting outliers in this way.

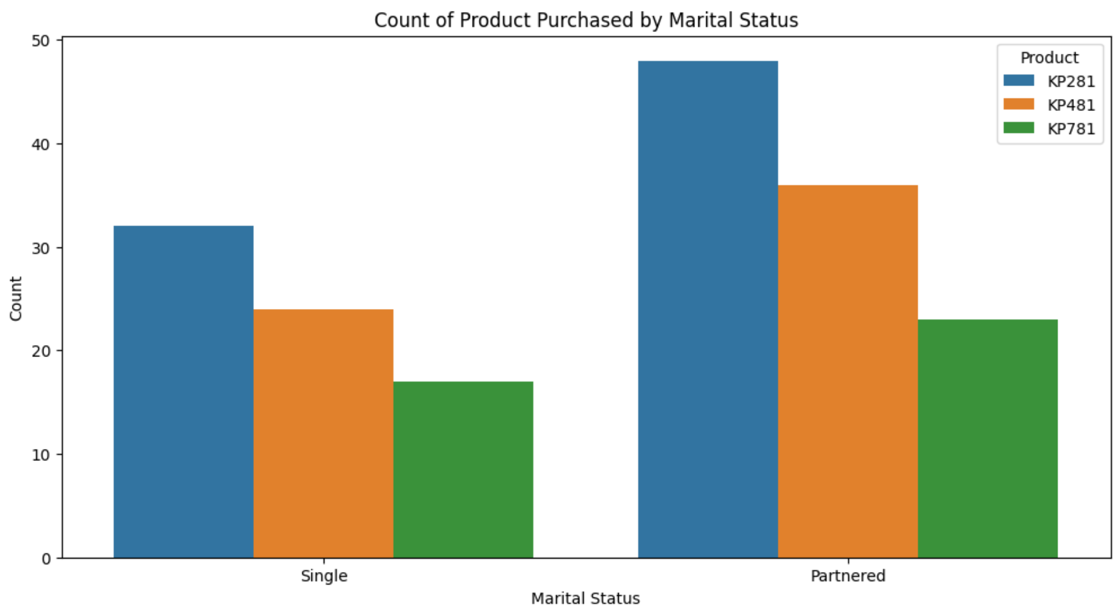
3. Check if features like marital status, Gender, and age have any effect on the product purchased

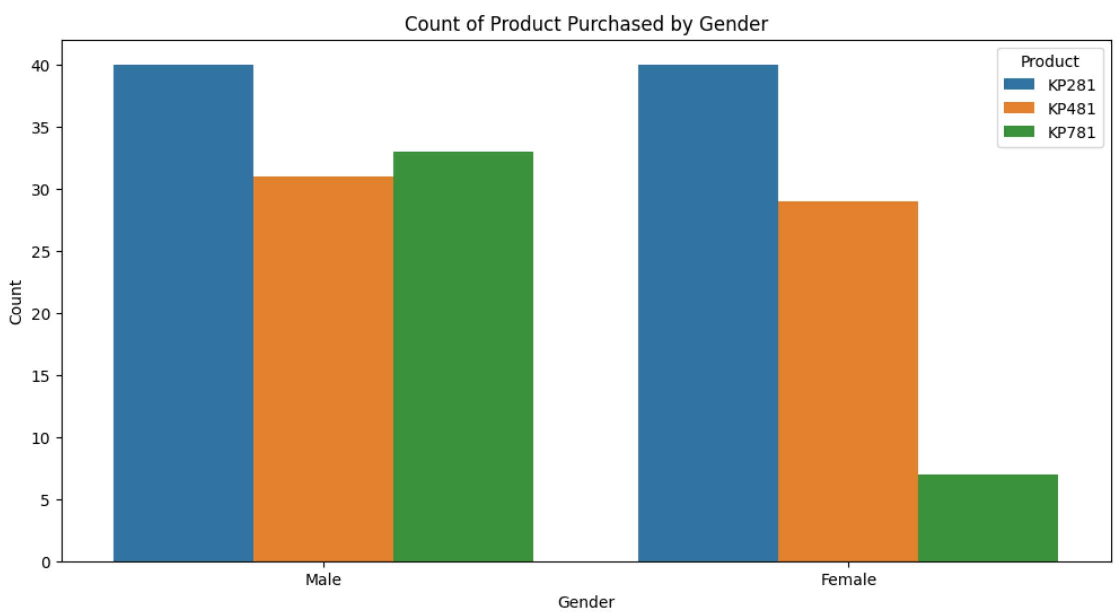
Find if there is any relationship between the categorical variables and the output variable in the data. Hint: We want you to use the count plot to find the relationship between categorical variables and output variables. Find if there is any relationship between the continuous variables and the output variable in the data. Hint: We want you to use a scatter plot to find the relationship between continuous variables and output variables.

[ ]:
```
#In marital status, Gender, and age ....gender, marital status are categorical␣
↪and age is continuos varible.
#so lets plot count plot for categorical variables
```

[ ]:
```
# Count plot for categorical variables
import seaborn as sns
plt.figure(figsize=(12, 6))
sns.countplot(x='MaritalStatus', hue='Product', data=df)
plt.title('Count of Product Purchased by Marital Status')
plt.xlabel('Marital Status')
plt.ylabel('Count')
plt.show()

plt.figure(figsize=(12, 6))
sns.countplot(x='Gender', hue='Product', data=df)
plt.title('Count of Product Purchased by Gender')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.show()
```





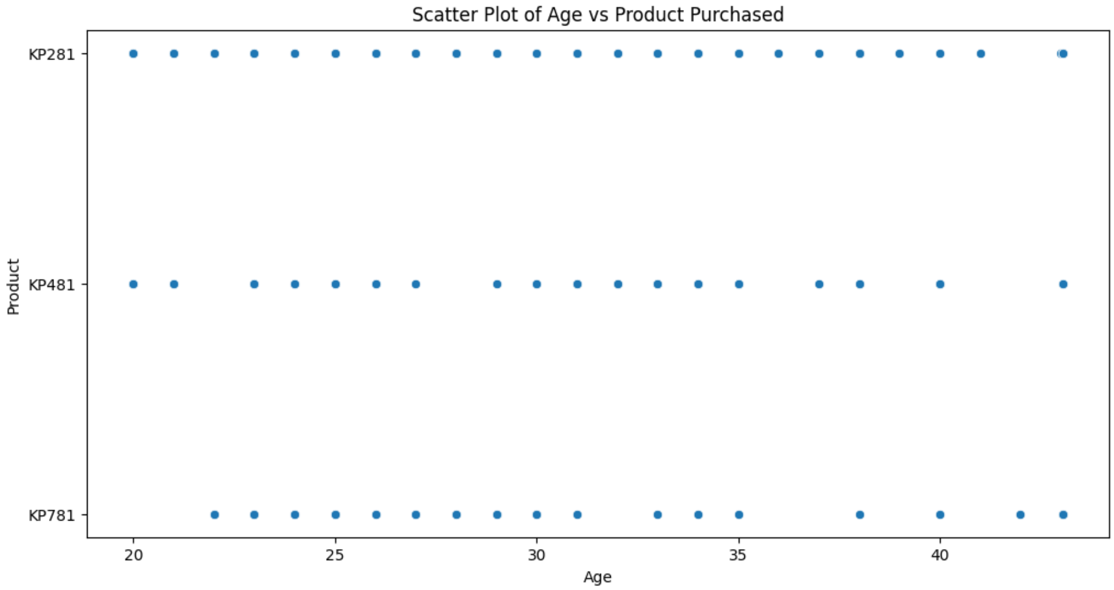
Insights

1.Irrespective of maritial status and gender, kp281 is being productive

2.partners are being productive more than singles 3.genders are being productive more than females

[ ]:
```
#lets plot count plot for continuos variables
```

```
[ ]: plt.figure(figsize=(12, 6))
sns.scatterplot(x='Age', y='Product', data=df)
plt.title('Scatter Plot of Age vs Product Purchased')
plt.xlabel('Age')
plt.ylabel('Product')
plt.show()
```




```
[ ]: #Insights : Seems kp281 is being used by mostly all age people than others
```

4.4. Representing the Probability Find the marginal probability (what percent of customers have purchased KP281, KP481, or KP781) Hint: We want you to use the pandas crosstab to find the marginal probability of each product. Find the probability that the customer buys a product based on each column. Hint: Based on previous crosstab values you find the probability. 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) Hint: Based on previous crosstab values you find the probability.

```
[ ]: #Marginal probability(Independent of others)
marginal_prob = pd.crosstab(index=df['Product'], columns='count',␣
 ↪normalize=True)

print("Marginal Probability:")
print(marginal_prob)
```

```
Marginal Probability:
col_0       count
Product
KP281    0.444444
```

```
KP481    0.333333
KP781    0.222222
```

Insights : From above , we can see that KP281 id having more margin and KP781 is having very less margin

[3]:
```
# Probability of buying a product based on each column
column_prob = {}
for col in df.columns[1:]:  # Exclude 'Product' column
    prob = df.groupby('Product')[col].value_counts(normalize=True)
    column_prob[col] = prob

print("\nProbability of buying a product based on each column:")
for col, prob in column_prob.items():
    print(f"\nColumn: {col}")
    print(prob)
```

```
Probability of buying a product based on each column:

Column: Age
Product  Age
KP281    23     0.1000
         25     0.0875
         26     0.0875
         28     0.0750
         24     0.0625
                  …
KP781    40     0.0250
         42     0.0250
         45     0.0250
         47     0.0250
         48     0.0250
Name: Age, Length: 68, dtype: float64

Column: Gender
Product  Gender
KP281    Female    0.500000
         Male      0.500000
KP481    Male      0.516667
         Female    0.483333
KP781    Male      0.825000
         Female    0.175000
Name: Gender, dtype: float64

Column: Education
Product  Education
KP281    16           0.487500
```

```
         14          0.375000
         15          0.050000
         13          0.037500
         12          0.025000
         18          0.025000
KP481    16          0.516667
         14          0.383333
         13          0.033333
         18          0.033333
         12          0.016667
         15          0.016667
KP781    18          0.475000
         16          0.375000
         21          0.075000
         14          0.050000
         20          0.025000
Name: Education, dtype: float64

Column: MaritalStatus
Product  MaritalStatus
KP281    Partnered        0.600
         Single           0.400
KP481    Partnered        0.600
         Single           0.400
KP781    Partnered        0.575
         Single           0.425
Name: MaritalStatus, dtype: float64

Column: Usage
Product  Usage
KP281    3        0.462500
         4        0.275000
         2        0.237500
         5        0.025000
KP481    3        0.516667
         2        0.233333
         4        0.200000
         5        0.050000
KP781    4        0.450000
         5        0.300000
         6        0.175000
         7        0.050000
         3        0.025000
Name: Usage, dtype: float64

Column: Fitness
Product  Fitness
KP281    3          0.675000
```

```
         2          0.175000
         4          0.112500
         5          0.025000
         1          0.012500
KP481    3          0.650000
         2          0.200000
         4          0.133333
         1          0.016667
KP781    5          0.725000
         4          0.175000
         3          0.100000
Name: Fitness, dtype: float64

Column: Income
Product  Income
KP281    46617     0.0875
         54576     0.0875
         52302     0.0750
         35247     0.0625
         45480     0.0625
                    …
KP781    85906     0.0250
         95508     0.0250
         95866     0.0250
         99601     0.0250
         103336    0.0250
Name: Income, Length: 83, dtype: float64

Column: Miles
Product  Miles
KP281    85       0.200000
         66       0.125000
         75       0.125000
         47       0.112500
         94       0.100000
         113      0.100000
         56       0.075000
         38       0.037500
         103      0.037500
         132      0.025000
         141      0.025000
         112      0.012500
         169      0.012500
         188      0.012500
KP481    95       0.200000
         85       0.183333
         106      0.133333
         53       0.116667
```

```
         64        0.100000
         127       0.083333
         42        0.066667
         74        0.050000
         170       0.033333
         21        0.016667
         212       0.016667
KP781    100       0.175000
         180       0.150000
         200       0.150000
         160       0.125000
         150       0.100000
         120       0.075000
         80        0.025000
         106       0.025000
         140       0.025000
         170       0.025000
         240       0.025000
         260       0.025000
         280       0.025000
         300       0.025000
         360       0.025000
Name: Miles, dtype: float64
```

Insights

Age:

Customers around the age of 25 show a higher probability of purchasing both KP481 and KP781 compared to other age groups, indicating a potential target demographic for these products.

Gender:

Male customers exhibit a notably higher probability of purchasing KP781 compared to female customers, suggesting a gender-based preference or marketing opportunity for this product.

Education:

Customers with an education level of 16 years have the highest probability of purchasing KP281, indicating a potential correlation between education level and preference for this product.

Marital Status:

Partnered customers have a higher probability of purchasing all three products compared to single customers, indicating that marital status may influence purchasing decisions, particularly for fitness-related products.

Usage:

Customers who plan to use the treadmill an average of 3 times per week show the highest probability of purchasing KP281, suggesting that frequency of usage may be a key factor in product selection.

Fitness:

Customers who rate their fitness level as 3 on a scale of 1 to 5 exhibit the highest probability of purchasing KP281, implying that individuals with moderate fitness levels may be the primary target audience for this product.

Income:

Customers with an income around the mid-range of the dataset (around $50,000 to $60,000) show a relatively higher probability of purchasing KP481, indicating that affordability may influence product choice.

Miles:

Customers planning to walk or run an average of 85 miles per week show the highest probability of purchasing KP281, suggesting that individuals with higher fitness goals may be inclined towards this product.

[5]:
```python
import pandas as pd

# Calculate probability of buying each product
total_customers = len(df)
product_counts = df['Product'].value_counts()
product_probabilities = product_counts / total_customers
print("Probability of buying each product:")
print(product_probabilities)

# Calculate conditional probability of buying a product given each column
print("\nConditional probability of buying a product given each column:")
for col in df.columns:
    if col != 'Product':
        print(f"\nColumn: {col}")
        for product in df['Product'].unique():
            for value in df[col].unique():
                product_col_counts = (df['Product'] == product) & (df[col] ==␣
 ↪value)
                product_col_prob = (product_col_counts).sum() / (df[col] ==␣
 ↪value).sum()
                print(f"P(Product={product} | {col}={value}): {product_col_prob:
 ↪.4f}")
```

```
Probability of buying each product:
KP281    0.444444
KP481    0.333333
KP781    0.222222
Name: Product, dtype: float64

Conditional probability of buying a product given each column:

Column: Age
P(Product=KP281 | Age=18): 1.0000
P(Product=KP281 | Age=19): 0.7500
```

```
P(Product=KP281 | Age=20): 0.4000
P(Product=KP281 | Age=21): 0.5714
P(Product=KP281 | Age=22): 0.5714
P(Product=KP281 | Age=23): 0.4444
P(Product=KP281 | Age=24): 0.4167
P(Product=KP281 | Age=25): 0.2800
P(Product=KP281 | Age=26): 0.5833
P(Product=KP281 | Age=27): 0.4286
P(Product=KP281 | Age=28): 0.6667
P(Product=KP281 | Age=29): 0.5000
P(Product=KP281 | Age=30): 0.2857
P(Product=KP281 | Age=31): 0.3333
P(Product=KP281 | Age=32): 0.5000
P(Product=KP281 | Age=33): 0.2500
P(Product=KP281 | Age=34): 0.3333
P(Product=KP281 | Age=35): 0.3750
P(Product=KP281 | Age=36): 1.0000
P(Product=KP281 | Age=37): 0.5000
P(Product=KP281 | Age=38): 0.5714
P(Product=KP281 | Age=39): 1.0000
P(Product=KP281 | Age=40): 0.2000
P(Product=KP281 | Age=41): 1.0000
P(Product=KP281 | Age=43): 1.0000
P(Product=KP281 | Age=44): 1.0000
P(Product=KP281 | Age=46): 1.0000
P(Product=KP281 | Age=47): 0.5000
P(Product=KP281 | Age=50): 1.0000
P(Product=KP281 | Age=45): 0.0000
P(Product=KP281 | Age=48): 0.0000
P(Product=KP281 | Age=42): 0.0000
P(Product=KP481 | Age=18): 0.0000
P(Product=KP481 | Age=19): 0.2500
P(Product=KP481 | Age=20): 0.6000
P(Product=KP481 | Age=21): 0.4286
P(Product=KP481 | Age=22): 0.0000
P(Product=KP481 | Age=23): 0.3889
P(Product=KP481 | Age=24): 0.2500
P(Product=KP481 | Age=25): 0.4400
P(Product=KP481 | Age=26): 0.2500
P(Product=KP481 | Age=27): 0.1429
P(Product=KP481 | Age=28): 0.0000
P(Product=KP481 | Age=29): 0.1667
P(Product=KP481 | Age=30): 0.2857
P(Product=KP481 | Age=31): 0.5000
P(Product=KP481 | Age=32): 0.5000
P(Product=KP481 | Age=33): 0.6250
P(Product=KP481 | Age=34): 0.5000
P(Product=KP481 | Age=35): 0.5000
```

```
P(Product=KP481 | Age=36): 0.0000
P(Product=KP481 | Age=37): 0.5000
P(Product=KP481 | Age=38): 0.2857
P(Product=KP481 | Age=39): 0.0000
P(Product=KP481 | Age=40): 0.6000
P(Product=KP481 | Age=41): 0.0000
P(Product=KP481 | Age=43): 0.0000
P(Product=KP481 | Age=44): 0.0000
P(Product=KP481 | Age=46): 0.0000
P(Product=KP481 | Age=47): 0.0000
P(Product=KP481 | Age=50): 0.0000
P(Product=KP481 | Age=45): 0.5000
P(Product=KP481 | Age=48): 0.5000
P(Product=KP481 | Age=42): 0.0000
P(Product=KP781 | Age=18): 0.0000
P(Product=KP781 | Age=19): 0.0000
P(Product=KP781 | Age=20): 0.0000
P(Product=KP781 | Age=21): 0.0000
P(Product=KP781 | Age=22): 0.4286
P(Product=KP781 | Age=23): 0.1667
P(Product=KP781 | Age=24): 0.3333
P(Product=KP781 | Age=25): 0.2800
P(Product=KP781 | Age=26): 0.1667
P(Product=KP781 | Age=27): 0.4286
P(Product=KP781 | Age=28): 0.3333
P(Product=KP781 | Age=29): 0.3333
P(Product=KP781 | Age=30): 0.4286
P(Product=KP781 | Age=31): 0.1667
P(Product=KP781 | Age=32): 0.0000
P(Product=KP781 | Age=33): 0.1250
P(Product=KP781 | Age=34): 0.1667
P(Product=KP781 | Age=35): 0.1250
P(Product=KP781 | Age=36): 0.0000
P(Product=KP781 | Age=37): 0.0000
P(Product=KP781 | Age=38): 0.1429
P(Product=KP781 | Age=39): 0.0000
P(Product=KP781 | Age=40): 0.2000
P(Product=KP781 | Age=41): 0.0000
P(Product=KP781 | Age=43): 0.0000
P(Product=KP781 | Age=44): 0.0000
P(Product=KP781 | Age=46): 0.0000
P(Product=KP781 | Age=47): 0.5000
P(Product=KP781 | Age=50): 0.0000
P(Product=KP781 | Age=45): 0.5000
P(Product=KP781 | Age=48): 0.5000
P(Product=KP781 | Age=42): 1.0000

Column: Gender
```

```
P(Product=KP281 | Gender=Male): 0.3846
P(Product=KP281 | Gender=Female): 0.5263
P(Product=KP481 | Gender=Male): 0.2981
P(Product=KP481 | Gender=Female): 0.3816
P(Product=KP781 | Gender=Male): 0.3173
P(Product=KP781 | Gender=Female): 0.0921

Column: Education
P(Product=KP281 | Education=14): 0.5455
P(Product=KP281 | Education=15): 0.8000
P(Product=KP281 | Education=12): 0.6667
P(Product=KP281 | Education=13): 0.6000
P(Product=KP281 | Education=16): 0.4588
P(Product=KP281 | Education=18): 0.0870
P(Product=KP281 | Education=20): 0.0000
P(Product=KP281 | Education=21): 0.0000
P(Product=KP481 | Education=14): 0.4182
P(Product=KP481 | Education=15): 0.2000
P(Product=KP481 | Education=12): 0.3333
P(Product=KP481 | Education=13): 0.4000
P(Product=KP481 | Education=16): 0.3647
P(Product=KP481 | Education=18): 0.0870
P(Product=KP481 | Education=20): 0.0000
P(Product=KP481 | Education=21): 0.0000
P(Product=KP781 | Education=14): 0.0364
P(Product=KP781 | Education=15): 0.0000
P(Product=KP781 | Education=12): 0.0000
P(Product=KP781 | Education=13): 0.0000
P(Product=KP781 | Education=16): 0.1765
P(Product=KP781 | Education=18): 0.8261
P(Product=KP781 | Education=20): 1.0000
P(Product=KP781 | Education=21): 1.0000

Column: MaritalStatus
P(Product=KP281 | MaritalStatus=Single): 0.4384
P(Product=KP281 | MaritalStatus=Partnered): 0.4486
P(Product=KP481 | MaritalStatus=Single): 0.3288
P(Product=KP481 | MaritalStatus=Partnered): 0.3364
P(Product=KP781 | MaritalStatus=Single): 0.2329
P(Product=KP781 | MaritalStatus=Partnered): 0.2150

Column: Usage
P(Product=KP281 | Usage=3): 0.5362
P(Product=KP281 | Usage=2): 0.5758
P(Product=KP281 | Usage=4): 0.4231
P(Product=KP281 | Usage=5): 0.1176
P(Product=KP281 | Usage=6): 0.0000
P(Product=KP281 | Usage=7): 0.0000
```

```
P(Product=KP481 | Usage=3): 0.4493
P(Product=KP481 | Usage=2): 0.4242
P(Product=KP481 | Usage=4): 0.2308
P(Product=KP481 | Usage=5): 0.1765
P(Product=KP481 | Usage=6): 0.0000
P(Product=KP481 | Usage=7): 0.0000
P(Product=KP781 | Usage=3): 0.0145
P(Product=KP781 | Usage=2): 0.0000
P(Product=KP781 | Usage=4): 0.3462
P(Product=KP781 | Usage=5): 0.7059
P(Product=KP781 | Usage=6): 1.0000
P(Product=KP781 | Usage=7): 1.0000

Column: Fitness
P(Product=KP281 | Fitness=4): 0.3750
P(Product=KP281 | Fitness=3): 0.5567
P(Product=KP281 | Fitness=2): 0.5385
P(Product=KP281 | Fitness=1): 0.5000
P(Product=KP281 | Fitness=5): 0.0645
P(Product=KP481 | Fitness=4): 0.3333
P(Product=KP481 | Fitness=3): 0.4021
P(Product=KP481 | Fitness=2): 0.4615
P(Product=KP481 | Fitness=1): 0.5000
P(Product=KP481 | Fitness=5): 0.0000
P(Product=KP781 | Fitness=4): 0.2917
P(Product=KP781 | Fitness=3): 0.0412
P(Product=KP781 | Fitness=2): 0.0000
P(Product=KP781 | Fitness=1): 0.0000
P(Product=KP781 | Fitness=5): 0.9355

Column: Income
P(Product=KP281 | Income=29562): 1.0000
P(Product=KP281 | Income=31836): 0.5000
P(Product=KP281 | Income=30699): 1.0000
P(Product=KP281 | Income=32973): 0.6000
P(Product=KP281 | Income=35247): 1.0000
P(Product=KP281 | Income=37521): 1.0000
P(Product=KP281 | Income=36384): 0.7500
P(Product=KP281 | Income=38658): 0.6000
P(Product=KP281 | Income=40932): 0.6667
P(Product=KP281 | Income=34110): 0.4000
P(Product=KP281 | Income=39795): 1.0000
P(Product=KP281 | Income=42069): 1.0000
P(Product=KP281 | Income=44343): 1.0000
P(Product=KP281 | Income=45480): 0.3571
P(Product=KP281 | Income=46617): 0.8750
P(Product=KP281 | Income=48891): 0.4000
P(Product=KP281 | Income=53439): 0.3750
```

```
P(Product=KP281 | Income=43206): 0.2000
P(Product=KP281 | Income=52302): 0.6667
P(Product=KP281 | Income=51165): 0.4286
P(Product=KP281 | Income=50028): 0.2857
P(Product=KP281 | Income=54576): 0.8750
P(Product=KP281 | Income=68220): 1.0000
P(Product=KP281 | Income=55713): 1.0000
P(Product=KP281 | Income=60261): 0.6667
P(Product=KP281 | Income=67083): 0.5000
P(Product=KP281 | Income=56850): 1.0000
P(Product=KP281 | Income=59124): 0.3333
P(Product=KP281 | Income=61398): 0.5000
P(Product=KP281 | Income=57987): 0.2500
P(Product=KP281 | Income=64809): 0.3333
P(Product=KP281 | Income=47754): 0.0000
P(Product=KP281 | Income=65220): 0.0000
P(Product=KP281 | Income=62535): 0.0000
P(Product=KP281 | Income=48658): 0.0000
P(Product=KP281 | Income=54781): 0.0000
P(Product=KP281 | Income=48556): 0.0000
P(Product=KP281 | Income=58516): 0.0000
P(Product=KP281 | Income=53536): 0.0000
P(Product=KP281 | Income=61006): 0.0000
P(Product=KP281 | Income=57271): 0.0000
P(Product=KP281 | Income=52291): 0.0000
P(Product=KP281 | Income=49801): 0.0000
P(Product=KP281 | Income=62251): 0.0000
P(Product=KP281 | Income=64741): 0.0000
P(Product=KP281 | Income=70966): 0.0000
P(Product=KP281 | Income=75946): 0.0000
P(Product=KP281 | Income=74701): 0.0000
P(Product=KP281 | Income=69721): 0.0000
P(Product=KP281 | Income=83416): 0.0000
P(Product=KP281 | Income=88396): 0.0000
P(Product=KP281 | Income=90886): 0.0000
P(Product=KP281 | Income=92131): 0.0000
P(Product=KP281 | Income=77191): 0.0000
P(Product=KP281 | Income=52290): 0.0000
P(Product=KP281 | Income=85906): 0.0000
P(Product=KP281 | Income=103336): 0.0000
P(Product=KP281 | Income=99601): 0.0000
P(Product=KP281 | Income=89641): 0.0000
P(Product=KP281 | Income=95866): 0.0000
P(Product=KP281 | Income=104581): 0.0000
P(Product=KP281 | Income=95508): 0.0000
P(Product=KP481 | Income=29562): 0.0000
P(Product=KP481 | Income=31836): 0.5000
P(Product=KP481 | Income=30699): 0.0000
```

```
P(Product=KP481 | Income=32973): 0.4000
P(Product=KP481 | Income=35247): 0.0000
P(Product=KP481 | Income=37521): 0.0000
P(Product=KP481 | Income=36384): 0.2500
P(Product=KP481 | Income=38658): 0.4000
P(Product=KP481 | Income=40932): 0.3333
P(Product=KP481 | Income=34110): 0.6000
P(Product=KP481 | Income=39795): 0.0000
P(Product=KP481 | Income=42069): 0.0000
P(Product=KP481 | Income=44343): 0.0000
P(Product=KP481 | Income=45480): 0.6429
P(Product=KP481 | Income=46617): 0.1250
P(Product=KP481 | Income=48891): 0.6000
P(Product=KP481 | Income=53439): 0.6250
P(Product=KP481 | Income=43206): 0.8000
P(Product=KP481 | Income=52302): 0.3333
P(Product=KP481 | Income=51165): 0.5714
P(Product=KP481 | Income=50028): 0.7143
P(Product=KP481 | Income=54576): 0.1250
P(Product=KP481 | Income=68220): 0.0000
P(Product=KP481 | Income=55713): 0.0000
P(Product=KP481 | Income=60261): 0.3333
P(Product=KP481 | Income=67083): 0.5000
P(Product=KP481 | Income=56850): 0.0000
P(Product=KP481 | Income=59124): 0.6667
P(Product=KP481 | Income=61398): 0.5000
P(Product=KP481 | Income=57987): 0.7500
P(Product=KP481 | Income=64809): 0.6667
P(Product=KP481 | Income=47754): 1.0000
P(Product=KP481 | Income=65220): 1.0000
P(Product=KP481 | Income=62535): 1.0000
P(Product=KP481 | Income=48658): 0.0000
P(Product=KP481 | Income=54781): 0.0000
P(Product=KP481 | Income=48556): 0.0000
P(Product=KP481 | Income=58516): 0.0000
P(Product=KP481 | Income=53536): 0.0000
P(Product=KP481 | Income=61006): 0.0000
P(Product=KP481 | Income=57271): 0.0000
P(Product=KP481 | Income=52291): 0.0000
P(Product=KP481 | Income=49801): 0.0000
P(Product=KP481 | Income=62251): 0.0000
P(Product=KP481 | Income=64741): 0.0000
P(Product=KP481 | Income=70966): 0.0000
P(Product=KP481 | Income=75946): 0.0000
P(Product=KP481 | Income=74701): 0.0000
P(Product=KP481 | Income=69721): 0.0000
P(Product=KP481 | Income=83416): 0.0000
P(Product=KP481 | Income=88396): 0.0000
```

```
P(Product=KP481 | Income=90886): 0.0000
P(Product=KP481 | Income=92131): 0.0000
P(Product=KP481 | Income=77191): 0.0000
P(Product=KP481 | Income=52290): 0.0000
P(Product=KP481 | Income=85906): 0.0000
P(Product=KP481 | Income=103336): 0.0000
P(Product=KP481 | Income=99601): 0.0000
P(Product=KP481 | Income=89641): 0.0000
P(Product=KP481 | Income=95866): 0.0000
P(Product=KP481 | Income=104581): 0.0000
P(Product=KP481 | Income=95508): 0.0000
P(Product=KP781 | Income=29562): 0.0000
P(Product=KP781 | Income=31836): 0.0000
P(Product=KP781 | Income=30699): 0.0000
P(Product=KP781 | Income=32973): 0.0000
P(Product=KP781 | Income=35247): 0.0000
P(Product=KP781 | Income=37521): 0.0000
P(Product=KP781 | Income=36384): 0.0000
P(Product=KP781 | Income=38658): 0.0000
P(Product=KP781 | Income=40932): 0.0000
P(Product=KP781 | Income=34110): 0.0000
P(Product=KP781 | Income=39795): 0.0000
P(Product=KP781 | Income=42069): 0.0000
P(Product=KP781 | Income=44343): 0.0000
P(Product=KP781 | Income=45480): 0.0000
P(Product=KP781 | Income=46617): 0.0000
P(Product=KP781 | Income=48891): 0.0000
P(Product=KP781 | Income=53439): 0.0000
P(Product=KP781 | Income=43206): 0.0000
P(Product=KP781 | Income=52302): 0.0000
P(Product=KP781 | Income=51165): 0.0000
P(Product=KP781 | Income=50028): 0.0000
P(Product=KP781 | Income=54576): 0.0000
P(Product=KP781 | Income=68220): 0.0000
P(Product=KP781 | Income=55713): 0.0000
P(Product=KP781 | Income=60261): 0.0000
P(Product=KP781 | Income=67083): 0.0000
P(Product=KP781 | Income=56850): 0.0000
P(Product=KP781 | Income=59124): 0.0000
P(Product=KP781 | Income=61398): 0.0000
P(Product=KP781 | Income=57987): 0.0000
P(Product=KP781 | Income=64809): 0.0000
P(Product=KP781 | Income=47754): 0.0000
P(Product=KP781 | Income=65220): 0.0000
P(Product=KP781 | Income=62535): 0.0000
P(Product=KP781 | Income=48658): 1.0000
P(Product=KP781 | Income=54781): 1.0000
P(Product=KP781 | Income=48556): 1.0000
```

```
P(Product=KP781 | Income=58516): 1.0000
P(Product=KP781 | Income=53536): 1.0000
P(Product=KP781 | Income=61006): 1.0000
P(Product=KP781 | Income=57271): 1.0000
P(Product=KP781 | Income=52291): 1.0000
P(Product=KP781 | Income=49801): 1.0000
P(Product=KP781 | Income=62251): 1.0000
P(Product=KP781 | Income=64741): 1.0000
P(Product=KP781 | Income=70966): 1.0000
P(Product=KP781 | Income=75946): 1.0000
P(Product=KP781 | Income=74701): 1.0000
P(Product=KP781 | Income=69721): 1.0000
P(Product=KP781 | Income=83416): 1.0000
P(Product=KP781 | Income=88396): 1.0000
P(Product=KP781 | Income=90886): 1.0000
P(Product=KP781 | Income=92131): 1.0000
P(Product=KP781 | Income=77191): 1.0000
P(Product=KP781 | Income=52290): 1.0000
P(Product=KP781 | Income=85906): 1.0000
P(Product=KP781 | Income=103336): 1.0000
P(Product=KP781 | Income=99601): 1.0000
P(Product=KP781 | Income=89641): 1.0000
P(Product=KP781 | Income=95866): 1.0000
P(Product=KP781 | Income=104581): 1.0000
P(Product=KP781 | Income=95508): 1.0000

Column: Miles
P(Product=KP281 | Miles=112): 1.0000
P(Product=KP281 | Miles=75): 1.0000
P(Product=KP281 | Miles=66): 1.0000
P(Product=KP281 | Miles=85): 0.5926
P(Product=KP281 | Miles=47): 1.0000
P(Product=KP281 | Miles=141): 1.0000
P(Product=KP281 | Miles=103): 1.0000
P(Product=KP281 | Miles=94): 1.0000
P(Product=KP281 | Miles=113): 1.0000
P(Product=KP281 | Miles=38): 1.0000
P(Product=KP281 | Miles=188): 1.0000
P(Product=KP281 | Miles=56): 1.0000
P(Product=KP281 | Miles=132): 1.0000
P(Product=KP281 | Miles=169): 1.0000
P(Product=KP281 | Miles=64): 0.0000
P(Product=KP281 | Miles=53): 0.0000
P(Product=KP281 | Miles=106): 0.0000
P(Product=KP281 | Miles=95): 0.0000
P(Product=KP281 | Miles=212): 0.0000
P(Product=KP281 | Miles=42): 0.0000
P(Product=KP281 | Miles=127): 0.0000
```

```
P(Product=KP281 | Miles=74): 0.0000
P(Product=KP281 | Miles=170): 0.0000
P(Product=KP281 | Miles=21): 0.0000
P(Product=KP281 | Miles=120): 0.0000
P(Product=KP281 | Miles=200): 0.0000
P(Product=KP281 | Miles=140): 0.0000
P(Product=KP281 | Miles=100): 0.0000
P(Product=KP281 | Miles=80): 0.0000
P(Product=KP281 | Miles=160): 0.0000
P(Product=KP281 | Miles=180): 0.0000
P(Product=KP281 | Miles=240): 0.0000
P(Product=KP281 | Miles=150): 0.0000
P(Product=KP281 | Miles=300): 0.0000
P(Product=KP281 | Miles=280): 0.0000
P(Product=KP281 | Miles=260): 0.0000
P(Product=KP281 | Miles=360): 0.0000
P(Product=KP481 | Miles=112): 0.0000
P(Product=KP481 | Miles=75): 0.0000
P(Product=KP481 | Miles=66): 0.0000
P(Product=KP481 | Miles=85): 0.4074
P(Product=KP481 | Miles=47): 0.0000
P(Product=KP481 | Miles=141): 0.0000
P(Product=KP481 | Miles=103): 0.0000
P(Product=KP481 | Miles=94): 0.0000
P(Product=KP481 | Miles=113): 0.0000
P(Product=KP481 | Miles=38): 0.0000
P(Product=KP481 | Miles=188): 0.0000
P(Product=KP481 | Miles=56): 0.0000
P(Product=KP481 | Miles=132): 0.0000
P(Product=KP481 | Miles=169): 0.0000
P(Product=KP481 | Miles=64): 1.0000
P(Product=KP481 | Miles=53): 1.0000
P(Product=KP481 | Miles=106): 0.8889
P(Product=KP481 | Miles=95): 1.0000
P(Product=KP481 | Miles=212): 1.0000
P(Product=KP481 | Miles=42): 1.0000
P(Product=KP481 | Miles=127): 1.0000
P(Product=KP481 | Miles=74): 1.0000
P(Product=KP481 | Miles=170): 0.6667
P(Product=KP481 | Miles=21): 1.0000
P(Product=KP481 | Miles=120): 0.0000
P(Product=KP481 | Miles=200): 0.0000
P(Product=KP481 | Miles=140): 0.0000
P(Product=KP481 | Miles=100): 0.0000
P(Product=KP481 | Miles=80): 0.0000
P(Product=KP481 | Miles=160): 0.0000
P(Product=KP481 | Miles=180): 0.0000
P(Product=KP481 | Miles=240): 0.0000
```

```
P(Product=KP481 | Miles=150): 0.0000
P(Product=KP481 | Miles=300): 0.0000
P(Product=KP481 | Miles=280): 0.0000
P(Product=KP481 | Miles=260): 0.0000
P(Product=KP481 | Miles=360): 0.0000
P(Product=KP781 | Miles=112): 0.0000
P(Product=KP781 | Miles=75): 0.0000
P(Product=KP781 | Miles=66): 0.0000
P(Product=KP781 | Miles=85): 0.0000
P(Product=KP781 | Miles=47): 0.0000
P(Product=KP781 | Miles=141): 0.0000
P(Product=KP781 | Miles=103): 0.0000
P(Product=KP781 | Miles=94): 0.0000
P(Product=KP781 | Miles=113): 0.0000
P(Product=KP781 | Miles=38): 0.0000
P(Product=KP781 | Miles=188): 0.0000
P(Product=KP781 | Miles=56): 0.0000
P(Product=KP781 | Miles=132): 0.0000
P(Product=KP781 | Miles=169): 0.0000
P(Product=KP781 | Miles=64): 0.0000
P(Product=KP781 | Miles=53): 0.0000
P(Product=KP781 | Miles=106): 0.1111
P(Product=KP781 | Miles=95): 0.0000
P(Product=KP781 | Miles=212): 0.0000
P(Product=KP781 | Miles=42): 0.0000
P(Product=KP781 | Miles=127): 0.0000
P(Product=KP781 | Miles=74): 0.0000
P(Product=KP781 | Miles=170): 0.3333
P(Product=KP781 | Miles=21): 0.0000
P(Product=KP781 | Miles=120): 1.0000
P(Product=KP781 | Miles=200): 1.0000
P(Product=KP781 | Miles=140): 1.0000
P(Product=KP781 | Miles=100): 1.0000
P(Product=KP781 | Miles=80): 1.0000
P(Product=KP781 | Miles=160): 1.0000
P(Product=KP781 | Miles=180): 1.0000
P(Product=KP781 | Miles=240): 1.0000
P(Product=KP781 | Miles=150): 1.0000
P(Product=KP781 | Miles=300): 1.0000
P(Product=KP781 | Miles=280): 1.0000
P(Product=KP781 | Miles=260): 1.0000
P(Product=KP781 | Miles=360): 1.0000
```

Insights for the conditional probabilty

Age: The probability of purchasing KP281 is higher for younger age groups (18-24) and some older age groups (36-41, 43-44, 46, 50). The probability of purchasing KP481 is higher for middle-aged groups (25-35) and some older age groups (45, 48). The probability of purchasing KP781 is higher for some middle-aged groups (22-30) and older age groups (42 and above 47).

Gender: The probability of purchasing KP281 is higher for females. The probability of purchasing KP481 is slightly higher for females. The probability of purchasing KP781 is higher for males.

Education: The probability of purchasing KP281 is higher for lower education levels (12-16 years). The probability of purchasing KP481 is higher for middle education levels (14-16 years). The probability of purchasing KP781 is higher for higher education levels (18 years and above).

MaritalStatus: The probabilities of purchasing KP281 and KP481 are slightly higher for those who are partnered. The probability of purchasing KP781 is higher for those who are single.

Usage: The probability of purchasing KP281 is higher for lower usage levels (2-4 times per week). The probability of purchasing KP481 is higher for moderate usage levels (3-4 times per week). The probability of purchasing KP781 is higher for higher usage levels (5 times or more per week).

Fitness: The probability of purchasing KP281 is higher for those with lower to moderate fitness levels (1-3). The probability of purchasing KP481 is higher for those with moderate fitness levels (2-3). The probability of purchasing KP781 is higher for those with higher fitness levels (4-5).

Income: The probability of purchasing KP281 is higher for low to moderate income levels (up to around $60,000). The probability of purchasing KP481 is higher for moderate income levels (around $40,000 to $65,000). The probability of purchasing KP781 is higher for higher income levels (above $70,000).

Miles: The probability of purchasing KP281 is higher for lower mileage levels (up to around 150 miles per week). The probability of purchasing KP481 is higher for moderate mileage levels (around 50-150 miles per week). The probability of purchasing KP781 is higher for higher mileage levels (above 120 miles per week).

[ ]:
```
5. Check the correlation among different factors
 Find the correlation between the given features in the table.
Hint: We want you can use the heatmap and corr function to find the correlation
between the variables
```

[ ]:
```
#Correlation b/w the given features of the data
```

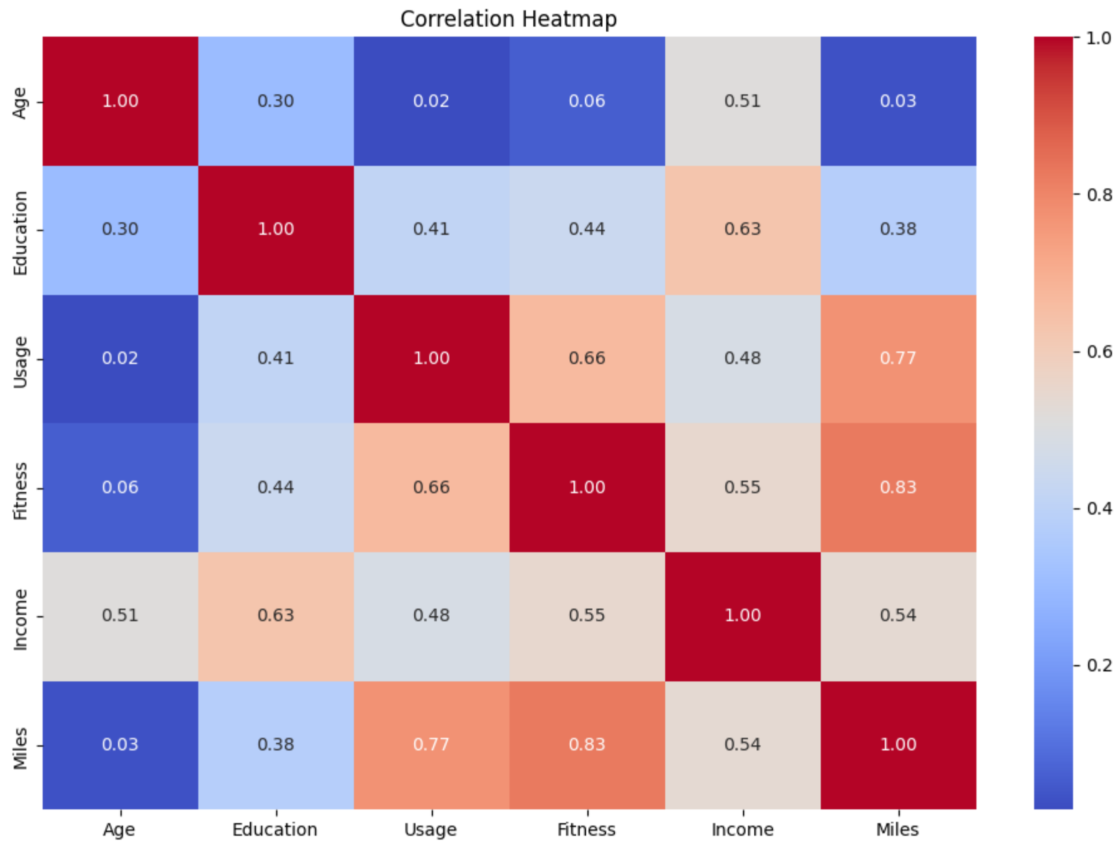
[ ]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Compute the correlation matrix
correlation_matrix = df.corr()

# Plot the heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap')
plt.show()
```

```
<ipython-input-46-0130cff847c4>:5: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
```

```
to silence this warning.
  correlation_matrix = df.corr()
```



Insights for the correlation From the above heatmap, we can observe that , there is ver minimal relatioship b/w age and usage with 0.02

6. Customer profiling and recommendation   Make customer profilings for each and every product. Hint: We want you to find at What age, gender, and income group but product the KP281   Write a detailed recommendation from the analysis that you have done.

[7]:
```python
import pandas as pd

# Assuming the data is in a CSV file named 'data.csv'
data = df

# Function to calculate conditional probability
def conditional_probability(data, product, feature, value):
    product_count = (data['Product'] == product).sum()
    feature_value_count = (data[feature] == value).sum()
    product_feature_count = ((data['Product'] == product) & (data[feature] ==␣
 ↪value)).sum()
```

```python
    if feature_value_count == 0:
        return 0
    else:
        return product_feature_count / feature_value_count

# Function to generate customer profile for a product
def generate_customer_profile(data, product):
    print(f"Customer Profile for {product}:")

    # Age
    age_probs = data.groupby('Age')['Product'].apply(lambda x:
 ↪conditional_probability(data, product, 'Age', x.name))
    print("\nAge:")
    print(age_probs[age_probs > 0.5].sort_values(ascending=False))

    # Gender
    print("\nGender:")
    for gender in data['Gender'].unique():
        prob = conditional_probability(data, product, 'Gender', gender)
        print(f"{gender}: {prob:.4f}")



    # Income
    print("\nIncome:")
    income_probs = data.groupby('Income')['Product'].apply(lambda x:
 ↪conditional_probability(data, product, 'Income', x.name))
    print(income_probs[income_probs > 0.5].sort_values(ascending=False))



# Generate customer profiles for each product
generate_customer_profile(data, 'KP281')
generate_customer_profile(data, 'KP481')
generate_customer_profile(data, 'KP781')
```

```
Customer Profile for KP281:

Age:
Age
18    1.000000
36    1.000000
39    1.000000
41    1.000000
43    1.000000
44    1.000000
```

```
46    1.000000
50    1.000000
19    0.750000
28    0.666667
26    0.583333
21    0.571429
22    0.571429
38    0.571429
Name: Product, dtype: float64

Gender:
Male: 0.3846
Female: 0.5263

Income:
Income
29562    1.000000
39795    1.000000
56850    1.000000
55713    1.000000
44343    1.000000
30699    1.000000
42069    1.000000
37521    1.000000
35247    1.000000
68220    1.000000
46617    0.875000
54576    0.875000
36384    0.750000
40932    0.666667
52302    0.666667
60261    0.666667
38658    0.600000
32973    0.600000
Name: Product, dtype: float64
Customer Profile for KP481:

Age:
Age
33    0.625
20    0.600
40    0.600
Name: Product, dtype: float64

Gender:
Male: 0.2981
Female: 0.3816
```

```
Income:
Income
47754    1.000000
62535    1.000000
65220    1.000000
43206    0.800000
57987    0.750000
50028    0.714286
59124    0.666667
64809    0.666667
45480    0.642857
53439    0.625000
34110    0.600000
48891    0.600000
51165    0.571429
Name: Product, dtype: float64
Customer Profile for KP781:

Age:
Age
42    1.0
Name: Product, dtype: float64

Gender:
Male: 0.3173
Female: 0.0921

Income:
Income
48556     1.0
48658     1.0
103336    1.0
99601     1.0
95866     1.0
95508     1.0
92131     1.0
90886     1.0
89641     1.0
88396     1.0
85906     1.0
83416     1.0
77191     1.0
75946     1.0
74701     1.0
70966     1.0
69721     1.0
64741     1.0
62251     1.0
```

```
61006     1.0
58516     1.0
57271     1.0
54781     1.0
53536     1.0
52291     1.0
52290     1.0
49801     1.0
104581    1.0
Name: Product, dtype: float64
```

Insights for the customer profiling for each product

KP281: The KP281 product is more likely to be purchased by younger or older females with lower to moderate education and income levels, who are partnered, have lower fitness levels and usage intentions, and expect lower mileage.

KP481: The KP481 product is more likely to be purchased by middle-aged or older females with moderate education and income levels, who are partnered, have moderate fitness levels and usage intentions, and expect moderate mileage.

KP781: The KP781 product is more likely to be purchased by middle-aged or older males with higher education and income levels, who are single, have higher fitness levels and usage intentions, and expect higher mileage.