Aerofit - Descriptive Statistics & Probability

1. Import the dataset and do usual data analysis steps like checking the structure & characteristics of the dataset

- We use the Pandas library to import the CSV file into a DataFrame for analysis.

```
url = "https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/125/original/aerofit_treadmill.csv?1639992749"
df = pd.read_csv(url)
df.head(10)
```

	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
5	KP281	20	Female	14	Partnered	3	3	32973	66
6	KP281	21	Female	14	Partnered	3	3	35247	75
7	KP281	21	Male	13	Single	3	3	32973	85
8	KP281	21	Male	15	Single	5	4	35247	141
9	KP281	21	Female	15	Partnered	2	3	37521	85

- **df.info()** summarizes the DataFrame, showing row/column counts, data types, missing values, and memory usage for quick analysis

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 180 entries, 0 to 179
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Product        180 non-null    object
 1   Age            180 non-null    int64
 2   Gender         180 non-null    object
 3   Education      180 non-null    int64
 4   MaritalStatus  180 non-null    object
 5   Usage          180 non-null    int64
 6   Fitness        180 non-null    int64
 7   Income         180 non-null    int64
 8   Miles          180 non-null    int64
dtypes: int64(6), object(3)
memory usage: 12.8+ KB
```

- We use isnull() to identify missing values in the dataset.

```
df.isnull().sum()
```

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

- **df.describe()** generates summary statistics of numerical columns, including count, mean, standard deviation, min, max, and quartiles, helping in data analysis.

	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

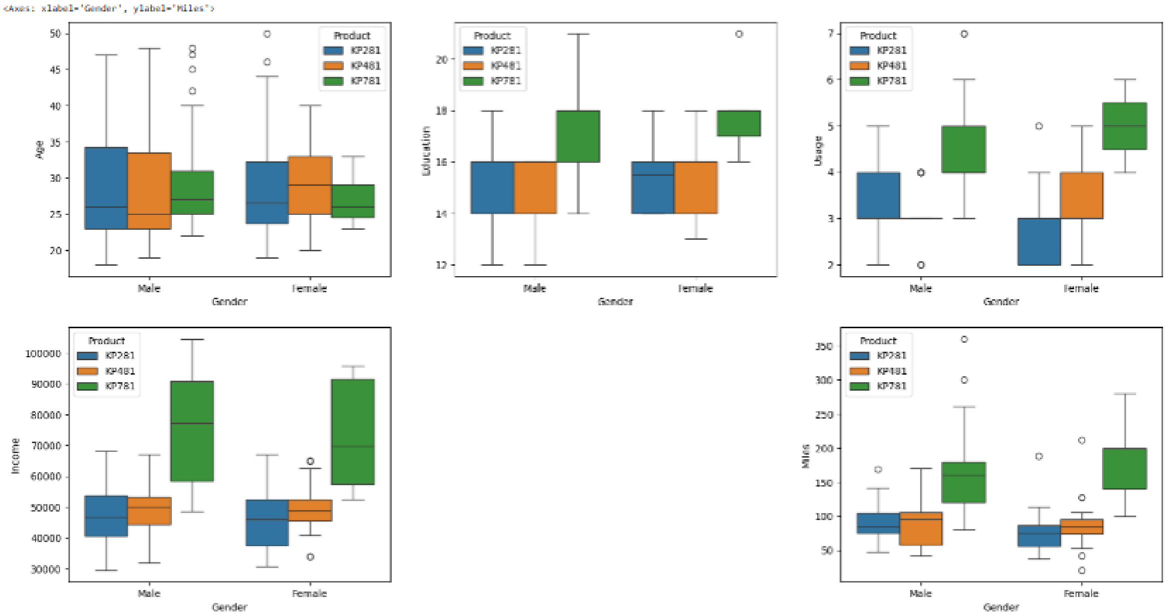
Insights:

After importing the dataset, we analyse its structure using **df.info()** to check column types and missing values. **df.describe()** provides summary statistics, while **df.isnull().sum()** helps detect missing data. These steps give insights into data distribution, potential outliers, and pre-processing needs.

2. Detect Outliers (using boxplot, "describe" method by checking the difference between mean and median)

- We can use boxplot to find the outliers for each numerical columns

<Axes: xlabel='Gender', ylabel='Miles'>



- We can use **df.describe()** to get the statistical insights.

	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

- We can use IQR method to detect extreme values.

```
def count_outliers(column):
    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
    outlier_count = df[(df[column] < lower_bound) | (df[column] > upper_bound)].shape[0]
    return outlier_count

for col in ['Age', 'Education', 'Usage', 'Miles', 'Income']:
    print(f"Number of outliers in {col}: {count_outliers(col)}")
```

```
Number of outliers in Age: 5
Number of outliers in Education: 4
Number of outliers in Usage: 9
Number of outliers in Miles: 13
Number of outliers in Income: 19
```

- Compare Mean & Median to Check Skewness

```
for i in ['Age', 'Education', 'Usage', 'Miles', 'Income']:
  median_val = df[i].median()
  mean_val = df[i].mean()
  print(f'{i}: Mean = {round(mean_val,2)}, Median = {round(median_val,2)}, Differnce = {round(abs(mean_val - median_val),2)}')
```

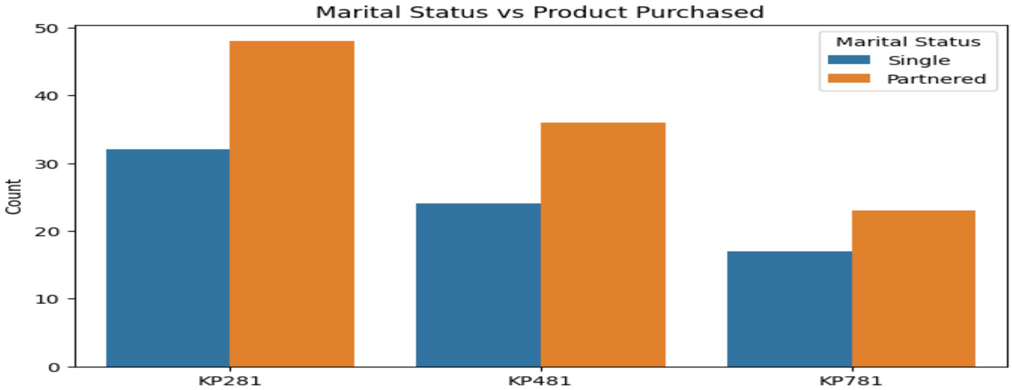
```
Age: Mean = 28.79, Median = 26.0, Differnce = 2.79
Education: Mean = 15.57, Median = 16.0, Differnce = 0.43
Usage: Mean = 3.46, Median = 3.0, Differnce = 0.46
Miles: Mean = 103.19, Median = 94.0, Differnce = 9.19
Income: Mean = 53719.58, Median = 50596.5, Differnce = 3123.08
```

Insights:

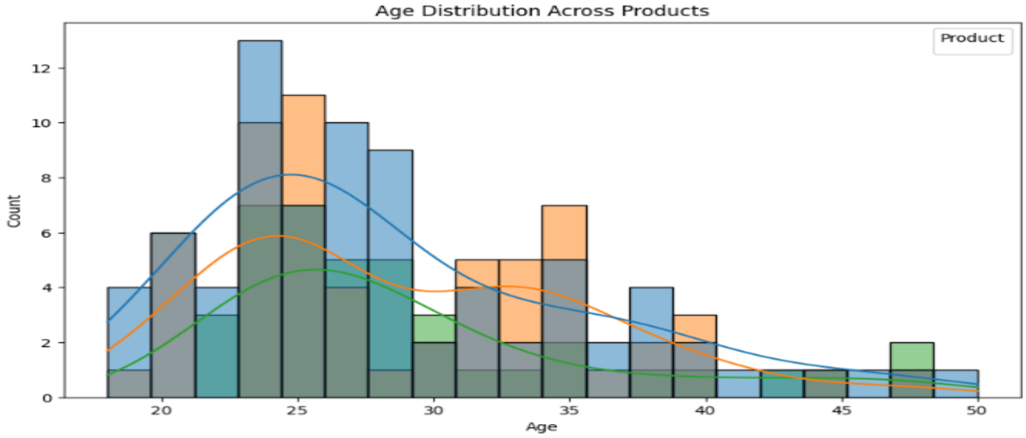
Box plots highlight outliers visually, while the **describe()** method offers statistical insights. The IQR method detects extreme values mathematically, and the mean-median difference reveals skewness caused by outliers.

3. Check if features like marital status, age have any effect on the product purchased (using countplot, histplots, boxplots etc)

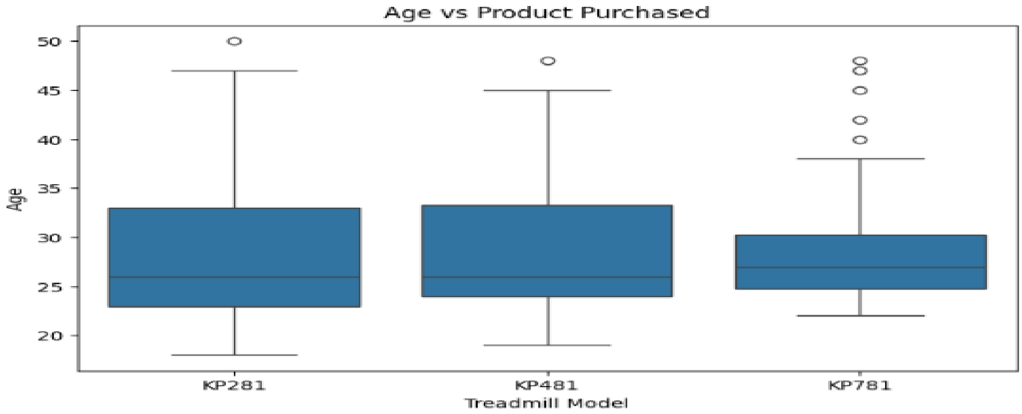
- We can use a count plot to visualize which marital status group has purchased more treadmills across different models.



- We can use histplot to see which age group of people purchase more treadmills across different models.



- We can use a boxplot to analyse which treadmill models were purchased by different age groups, including any outliers.



Insights:

Partnered individuals prefer KP281 and purchase more treadmills overall. Most buyers are aged 20–30, with a decline in older groups. KP781 has older buyers, including outliers.

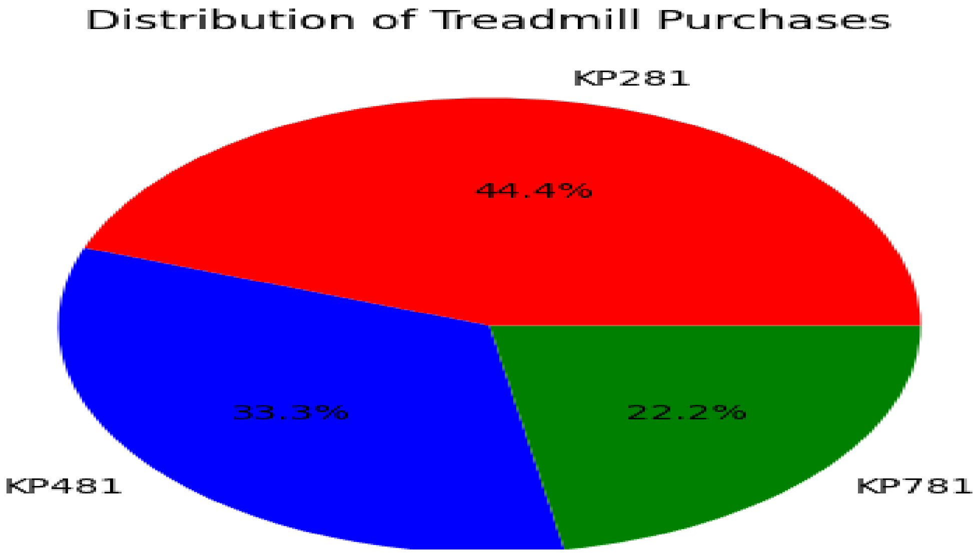
4. Representing the marginal probability like - what percent of customers have purchased KP281, KP481, or KP781 in a table *(can use pandas.crosstab here)*

- Using pandas.crosstab to count how many customers purchased each treadmill model.
- To get the probability distribution, divide by the total number of customers and multiply by 100.

```
product_counts = pd.crosstab(index=df['Product'], columns='Count')
product_percent = product_counts / product_counts.sum() * 100
product_percent.rename(columns={'Count': 'Percentage (%)'}, inplace=True)
print(product_percent)
```

```
col_0    Percentage (%)
Product
KP281         44.444444
KP481         33.333333
KP781         22.222222
```

- Let’s see the visualization using pie chart.



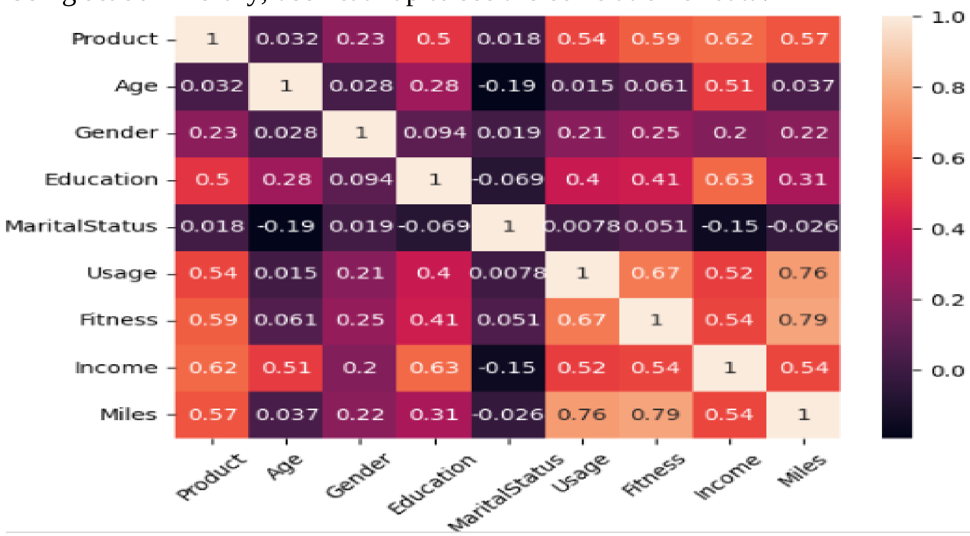
-

5. Check correlation among different factors using heat maps or pair plots.

- we need to change the categorical values to numerical values.

```
df_encoded = df.copy()
df_encoded['Product'] = df_encoded['Product'].astype('category').cat.codes
df_encoded['MaritalStatus'] = df_encoded['MaritalStatus'].astype('category').cat.codes
df_encoded['Gender'] = df_encoded['Gender'].astype('category').cat.codes
```

-
- Using seaborn library, use heatmap to see the correlation of data.



-

Insights:

The heatmap shows that **Income (0.62), Fitness (0.59), and Usage (0.54)** influence treadmill purchases, with higher-income and fitness-conscious individuals buying more. **Miles and Fitness (0.79)** are strongly linked, indicating active users log more distance. **Marital Status (-0.018)** has little impact on product choice.

6. With all the above steps you can answer questions like: What is the probability of a male customer buying a KP781 treadmill?

- We can use a crosstab to identify the number of KP781 treadmills purchased by male customers.

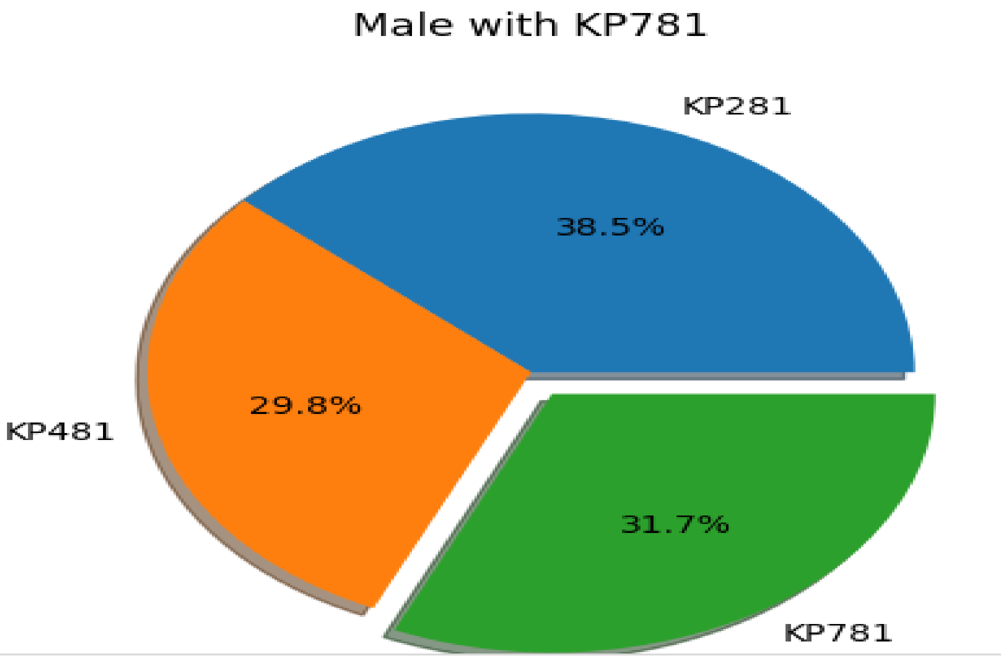
```
gender_product_table = pd.crosstab(df['Gender'], df['Product'])
gender_product_table
```

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

- By using the loc function, we can extract data specifically for male customers.

```
male_kp781 = gender_product_table.loc['Male']
male_kp781
```

- Using pie chart helps to visualize the data.



Insights:

The pie chart highlights that **31.7% of male customers purchased the KP781 treadmill**, making it a significant choice among them. However, KP281 remains the most purchased model.

7. **Customer Profiling** - Categorization of users.

❖ AGE SEGMENTATION:

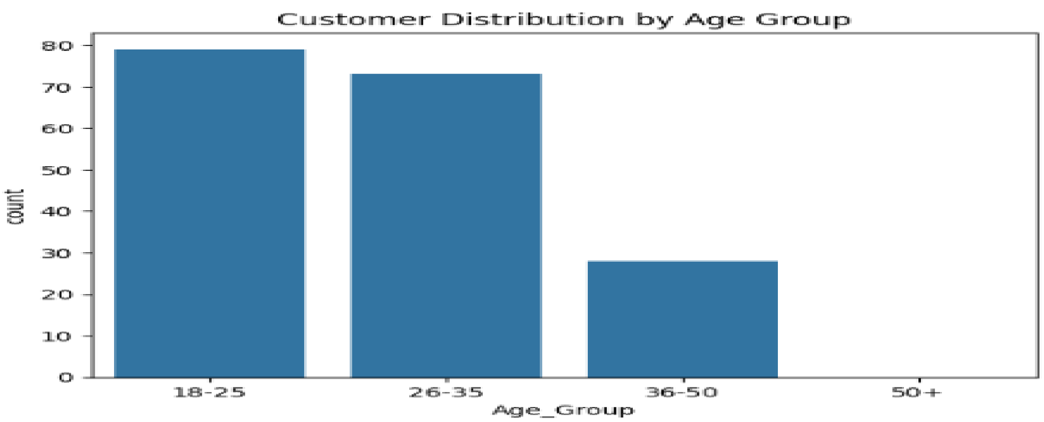
- Divides the **Age** column into groups using pd.cut(), assigning each person an age group and counting how many fall into each category.

```
#A. Age-Based Segmentation
bins = [0, 25, 35, 50, 100]
labels = ['18-25', '26-35', '36-50', '50+']
df['Age_Group'] = pd.cut(df['Age'], bins=bins, labels=labels)

# Count per category
df['Age_Group'].value_counts()
```

Age_Group	count
18-25	79
26-35	73
36-50	28
50+	0

```
sns.countplot(x='Age_Group', data=df)
plt.title("Customer Distribution by Age Group")
plt.show()
```



Insights:

The majority of customers fall within the **18-25** and **26-35** age groups, while significantly fewer belong to the **36-50** group, and almost none are in the **50+** category.

❖ INCOME BASED SEGMENTATION:

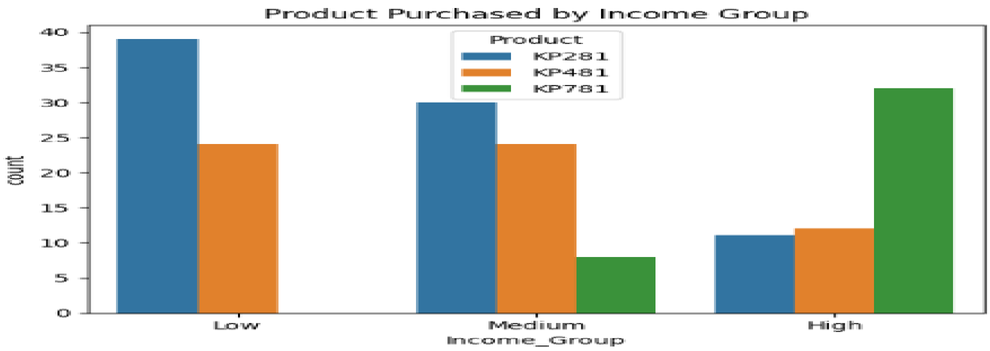
- Segments the **Income** column into three quantile-based groups using pd.qcut(), assigning labels and counting entries in each category.

```
#Income-Based Segmentation
df['Income_Group'] = pd.qcut(df['Income'], q=3, labels=['Low', 'Medium', 'High'])

# Count per category
df['Income_Group'].value_counts()
```

Income_Group	count
Low	63
Medium	62
High	55

```
sns.countplot(x='Income_Group', hue='Product', data=df)
plt.title("Product Purchased by Income Group")
plt.show()
```

➢ 

Insights:

Low and medium-income groups prefer **KP281** the most, followed by **KP481**, with minimal interest in **KP781**. In contrast, high-income customers favor **KP781**, while **KP281** and **KP481** have lower but similar purchase counts.

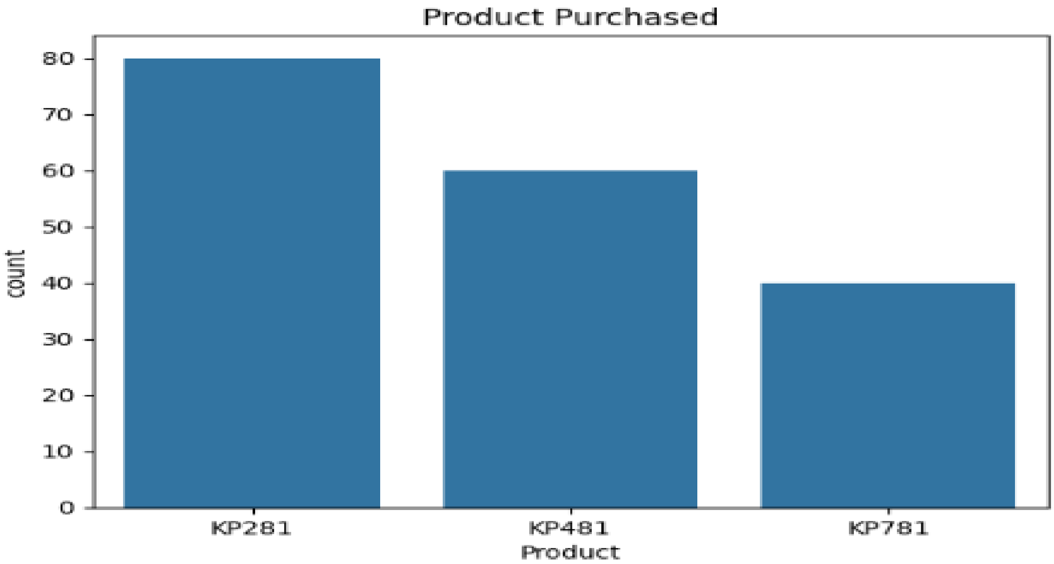
❖ **PRODUCT SEGEMENTATION:**

➢ Counts the occurrences of each product in the **Product** column, showing which products are most and least purchased.

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

➢

Product	count
KP281	80
KP481	60
KP781	40

➢ 

Insights:

The bar chart displays the count of products purchased, showing that **KP281** was the most purchased, followed by **KP481**, and **KP781** had the least purchases.

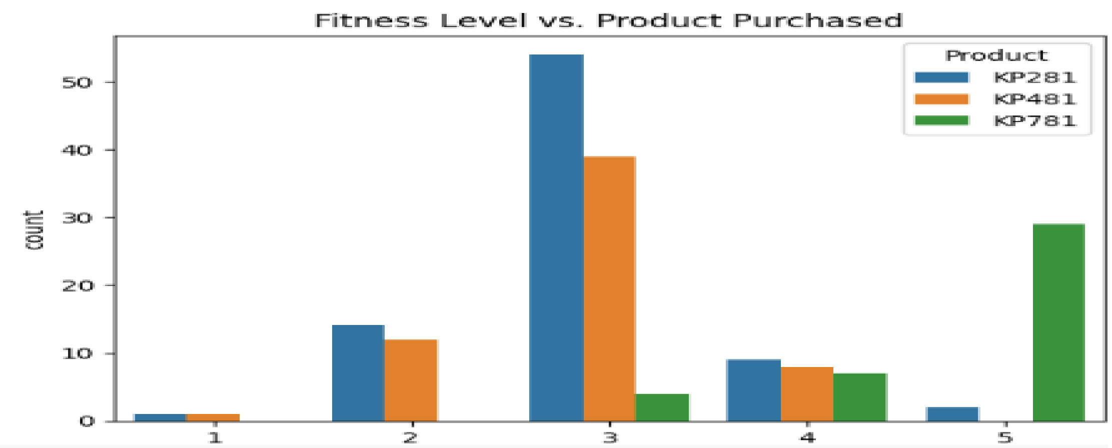
❖ FITNESS LEVEL SEGMENTATION:

- Counts the occurrences of each unique value in the **Fitness** column, showing the distribution of fitness levels in the dataset.

```
df['Fitness'].value_counts()
```

Fitness	count
3	97
5	31
2	26
4	24
1	2

-
```
sns.countplot(x='Fitness', hue='Product', data=df)
plt.title("Fitness Level vs. Product Purchased")
plt.show()
```



-

Insights:
Customers with **fitness level 3** make the most purchases, mainly of **KP281** and **KP481**, while **KP781** is favoured by those with **fitness level 5**. Fitness levels **1 and 4** show lower purchase activity, with **KP281** being the most popular across all levels.

❖ USAGE AND MILES SEGMENTATION:

- Creates two categorical columns: **Usage_Category** (based on predefined usage bins) and **Miles_Category** (based on mileage quantiles), then displays the first few rows.

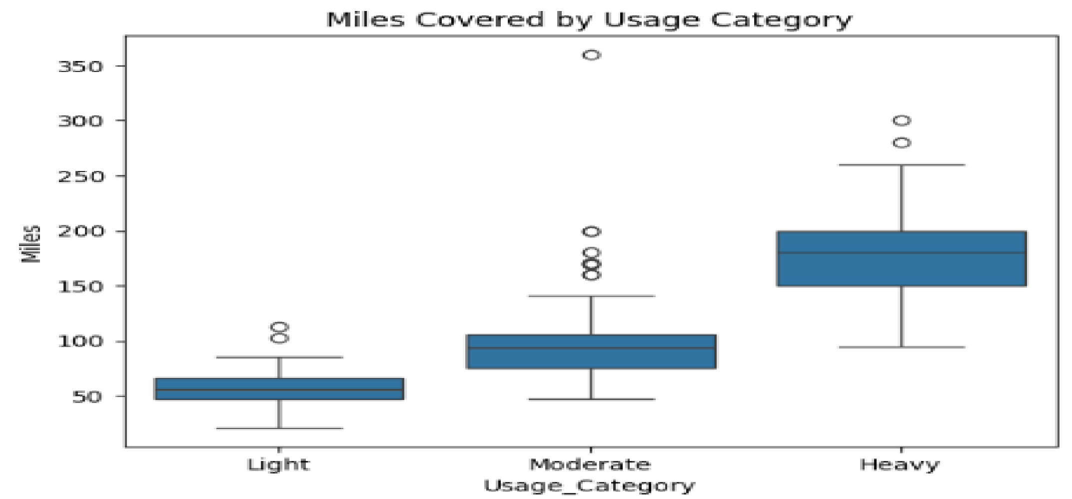
```
df['Usage_Category'] = pd.cut(df['Usage'], bins=[0, 2, 4, 7], labels=['Light', 'Moderate', 'Heavy'])

df['Miles_Category'] = pd.qcut(df['Miles'], q=3, labels=['Low', 'Medium', 'High'])

df[['Usage_Category', 'Miles_Category']].head()
```

	Usage_Category	Miles_Category
0	Moderate	High
1	Light	Low
2	Moderate	Low
3	Moderate	Medium
4	Moderate	Low

-

➢ 

Insights:

Heavier usage leads to higher miles covered, with **Heavy users** having the widest range and highest median miles. **Moderate users** cover more miles than **Light users**, but with some outliers.

8. **Probability**- marginal, conditional probability.

➢ MARGINAL PROBABILITY:

- **Marginal probability** measures the likelihood of a single event occurring, independent of other variables.
- It helps identify how different groups are distributed within a dataset.
- By using a **loop**, we can apply the marginal probability formula to each column, enabling a comprehensive analysis of group distributions.
-

```
#marginal probability
for i in df[['Age_Group', 'income_slab', 'MaritalStatus', 'Education_slab','Fitness_Category','Miles_Category']]:
  print(f"\n--- Marginal Probabilities for {i} ---")
  for j in df[i].unique():
    count = len(df[df[i] == j])
    print(f"{j}: {count}")
```

```
--- Marginal Probabilities for Age_Group ---
18-25: 79
26-35: 73
36-50: 28

--- Marginal Probabilities for income_slab ---
Low: 83
Medium: 74
High: 20
nan: 0

--- Marginal Probabilities for MaritalStatus ---
Single: 73
Partnered: 107

--- Marginal Probabilities for Education_slab ---
Associate Degree: 68
Bachelor Degree: 108
Master Degree: 4

--- Marginal Probabilities for Fitness_Category ---
High: 55
Medium: 97
Low: 28
```
-

➢ CONDITIONAL PROBABILITY:

- Conditional **probability** measures the likelihood of an event given another event has occurred.
- It reveals relationships between variables and their influence on each other.
- Helps in predictions, such as purchase likelihood based on age or income.
- Using a **loop**, we can compute conditional probabilities across multiple columns for deeper insights.

```
#conditional probability
for i in df[['Age_Group', 'income_slab', 'MaritalStatus', 'Education_slab', 'Fitness_Category','Miles_Category', ]]:
  conditional_prob = df.groupby(i)['Product'].value_counts(normalize=True)
  print(f"---Conditional Probability of Purchase Given {i}---\n", conditional_prob, '\n')
```

```
---Conditional Probability of Purchase Given Age_Group---
 Age_Group  Product
18-25      KP281      0.430380
           KP481      0.354430
           KP781      0.215190
26-35      KP281      0.438356
           KP481      0.328767
           KP781      0.232877
36-50      KP281      0.500000
           KP481      0.285714
           KP781      0.214286
50+        KP281      0.000000
           KP481      0.000000
           KP781      0.000000
Name: proportion, dtype: float64

---Conditional Probability of Purchase Given income_slab---
 income_slab  Product
Low          KP281      0.578313
             KP481      0.361446
             KP781      0.060241
Medium       KP281      0.432432
             KP481      0.405405
             KP781      0.162162
High         KP781      1.000000
             KP281      0.000000
             KP481      0.000000
Name: proportion, dtype: float64

---Conditional Probability of Purchase Given MaritalStatus---
 MaritalStatus  Product
Partnered      KP281      0.448598
               KP481      0.336449
               KP781      0.214953
Single         KP281      0.438356
               KP481      0.328767
               KP781      0.232877
```

Insights:
The **conditional probability** analysis reveals the likelihood of purchasing a specific product based on factors like **Age, Income, Marital Status, Education, Fitness, and Miles Category**. This helps identify which groups are more likely to buy certain products, making it useful for **targeted marketing** and customer segmentation. On the other hand, **marginal probability** provides the overall distribution of these categories, helping to understand which groups dominate the dataset. It serves as a **baseline** for deeper analysis by comparing how different factors influence purchase behaviour.

9. Some recommendations and actionable insights, based on the inferences.

❖ Target Younger Buyers – Focus on 18-35 age group via social media ads, discounts, and financing options.

- Promote Premium Models to High-Income Customers – Position KP781 as a luxury treadmill with bundled offers and targeted ads.
- Leverage Fitness Segmentation – Personalize marketing based on fitness levels; offer training programs and recommendations.
- Engage Heavy Users – Highlight durability, extended warranties, and trade-in programs for frequent treadmill users.
- Boost Sales Among Married Customers – Introduce family discounts, referral programs, and couple fitness plans.
- Use Data for Smarter Targeting – Implement AI-based recommendations and personalized email campaigns based on conditional probability.
- Optimize Stock & Pricing – Keep higher stock of KP281, offer discounts or bundles for KP781, and adjust pricing dynamically.