Aerofit Business case study

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

Structure:

Given data has 180 rows & 9 columns, viz ‘Product', 'Age', 'Gender', 'Education', 'MaritalStatus', 'Usage', 'Fitness', 'Income', 'Miles'.

```
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
```

a. Characteristics:

1. The data type of all columns in the “customers” table.

```
# 1. Data types of each column (part of dataset structure)
print("Data Types:\n", df.dtypes)
```

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

b. You can find the number of rows and columns given in the dataset

```
[59] df.shape
```

```
(180, 9)
```

c. Check for the missing values and find the number of missing values in each Column

```
[6] df.isnull().sum()
```

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

dtype: int64

2. Detect Outliers

a. Find the outliers for every continuous variable in the dataset.

```
# Detecting Outliers
num_col = df.select_dtypes(include=[np.number]).columns

def find_outliers(df, 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
    outliers = df[(df[column] < lower_bound) | (df[column] > upper_bound)][column]
    return outliers

for col in num_col:
    outliers = find_outliers(df, col)
    print(f"Outliers in '{col}':")
    print(outliers, "\n")
```

```
175    21
Name: Education, dtype: int64

Outliers in 'Usage':
154    6
155    6
162    6
163    7
164    6
166    7
167    6
170    6
175    6
Name: Usage, dtype: int64
```

```
Outliers in '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
Name: Miles, dtype: int64
```

```
Outliers in 'Fitness':
14     1
117    1
Name: Fitness, dtype: int64

Outliers in 'Income':
159     83416
160     88396
161     90886
162     92131
164     88396
166     85906
167     90886
168    103336
169     99601
170     89641
171     95866
172     92131
173     92131
174    104581
175     83416
176     89641
177     90886
178    104581
179     95508
Name: Income, dtype: int64
```

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

```
# To clip data between 5 percentile and 95 percentile
def clip_percentiles(df):
    for col in df.select_dtypes(include=[np.number]).columns:
        lower = np.percentile(df[col], 5)
        upper = np.percentile(df[col], 95)
        df[col] = np.clip(df[col], lower, upper)
    return df
df_clipped = clip_percentiles(df)
print(df_clipped)
```

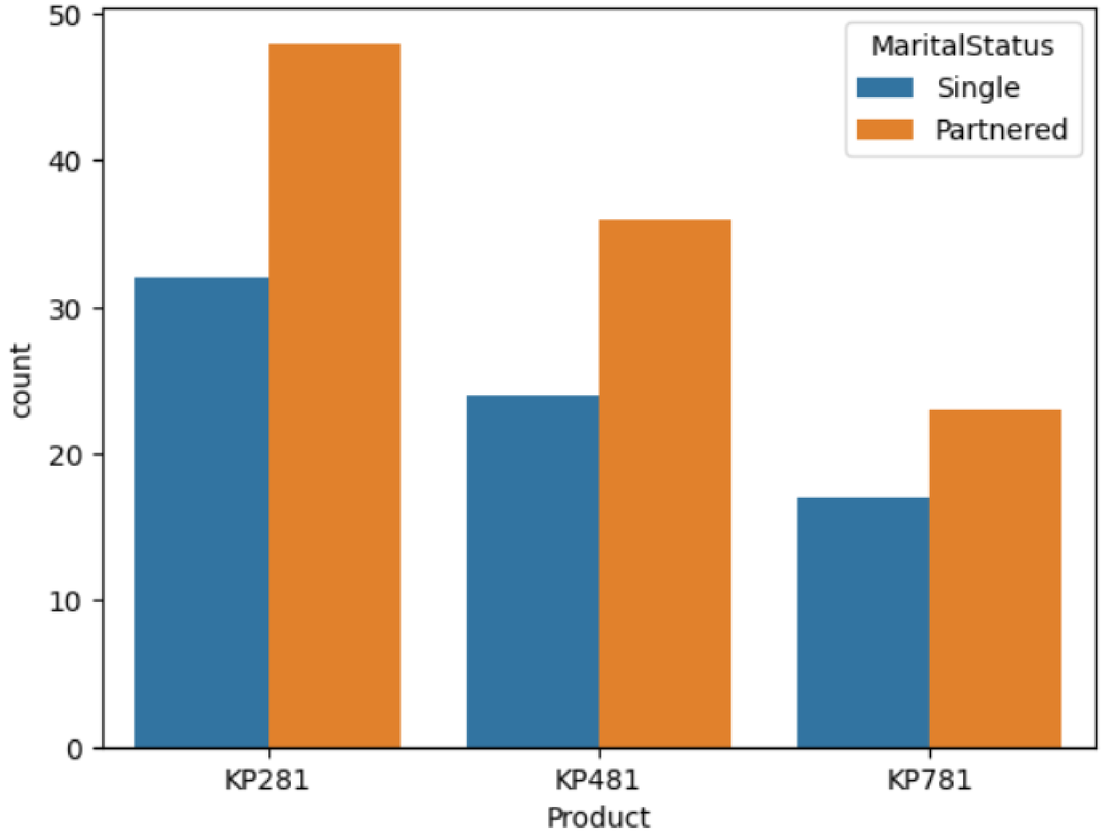
```
    Product    Age  Gender  Education MaritalStatus  Usage  Fitness    Income  \
0     KP281  20.00    Male         14        Single   3.00        4  34053.15
1     KP281  20.00    Male         15        Single   2.00        3  34053.15
2     KP281  20.00  Female         14     Partnered   4.00        3  34053.15
3     KP281  20.00    Male         14        Single   3.00        3  34053.15
4     KP281  20.00    Male         14     Partnered   4.00        2  35247.00
..      ...    ...     ...        ...           ...    ...      ...       ...
175   KP781  40.00    Male         18        Single   5.05        5  83416.00
176   KP781  42.00    Male         18        Single   5.00        4  89641.00
177   KP781  43.05    Male         16        Single   5.00        5  90886.00
178   KP781  43.05    Male         18     Partnered   4.00        5  90948.25
179   KP781  43.05    Male         18     Partnered   4.00        5  90948.25

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

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

- **Partnered people has purchased more product as compared to single people.**
- **Among all three products, KP281 was most purchased by partnered people.**

```
[14] sbn.countplot(data=df, x='Product', hue='MaritalStatus')
     plt.show()
```

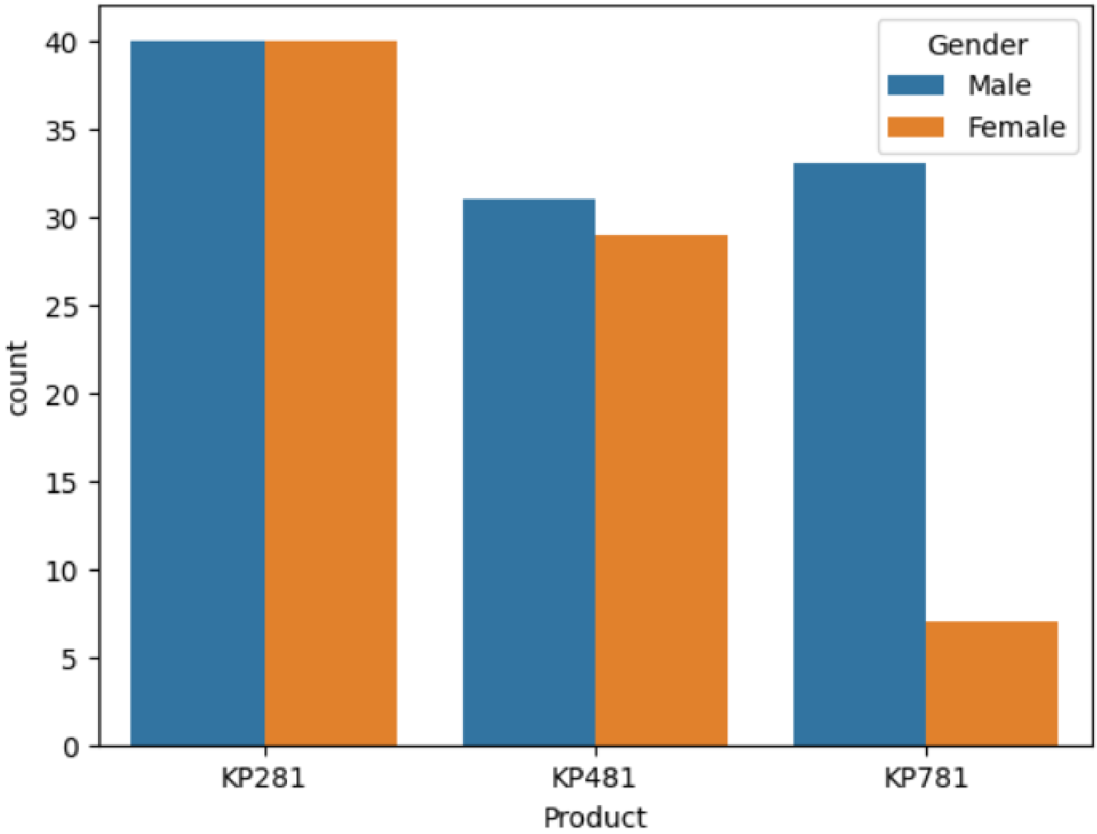
```
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning:
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning:
  data_subset = grouped_data.get_group(pd_key)
```



➢ All products were purchase by male people most but product KP281 was purchased by both equally.

```
[12] sbn.countplot(data=df, x='Product', hue='Gender')
     plt.show()
```

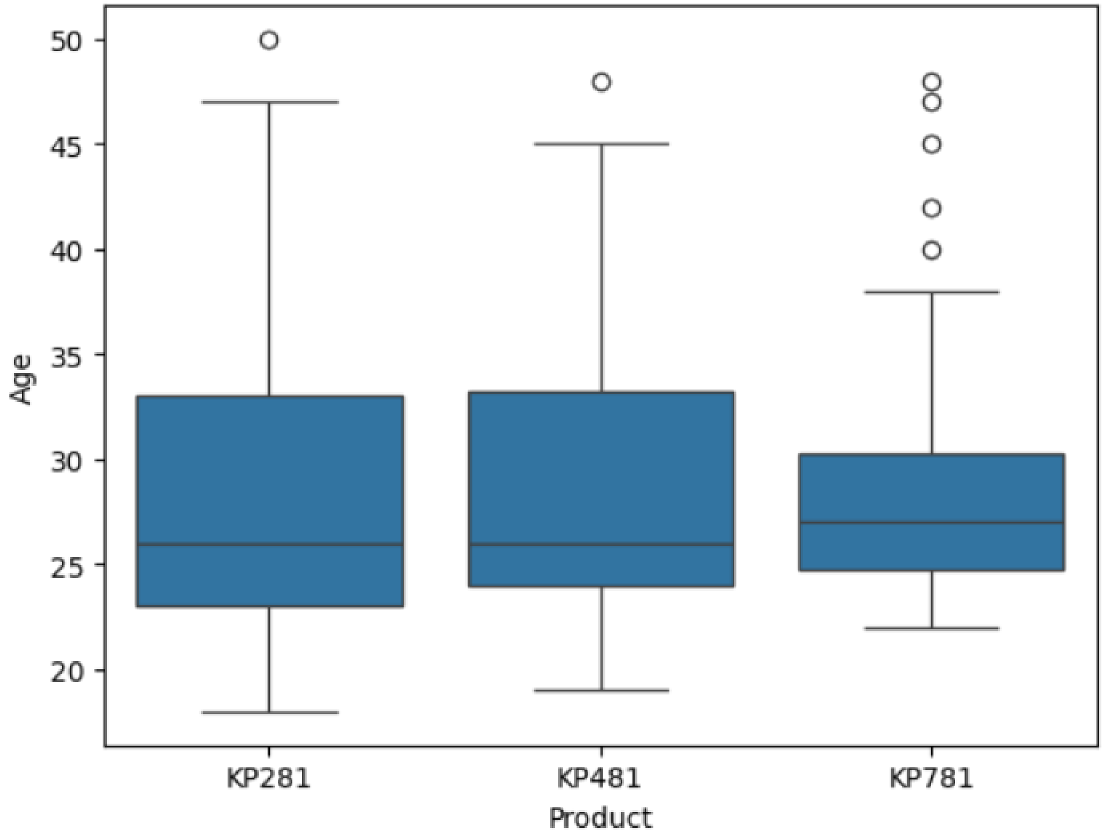
```
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning
  data_subset = grouped_data.get_group(pd_key)
```



➢ Product KP281 was most purchased, by people at median age 26 followed by Product KP481, by the people at median age 26 and least purchased product was KP781 by the people at median age 27.

```
[15] sbn.boxplot(data=df, x='Product', y='Age')
     plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:640: FutureWa
  positions = grouped.grouper.result_index.to_numpy(dtype=float)
```

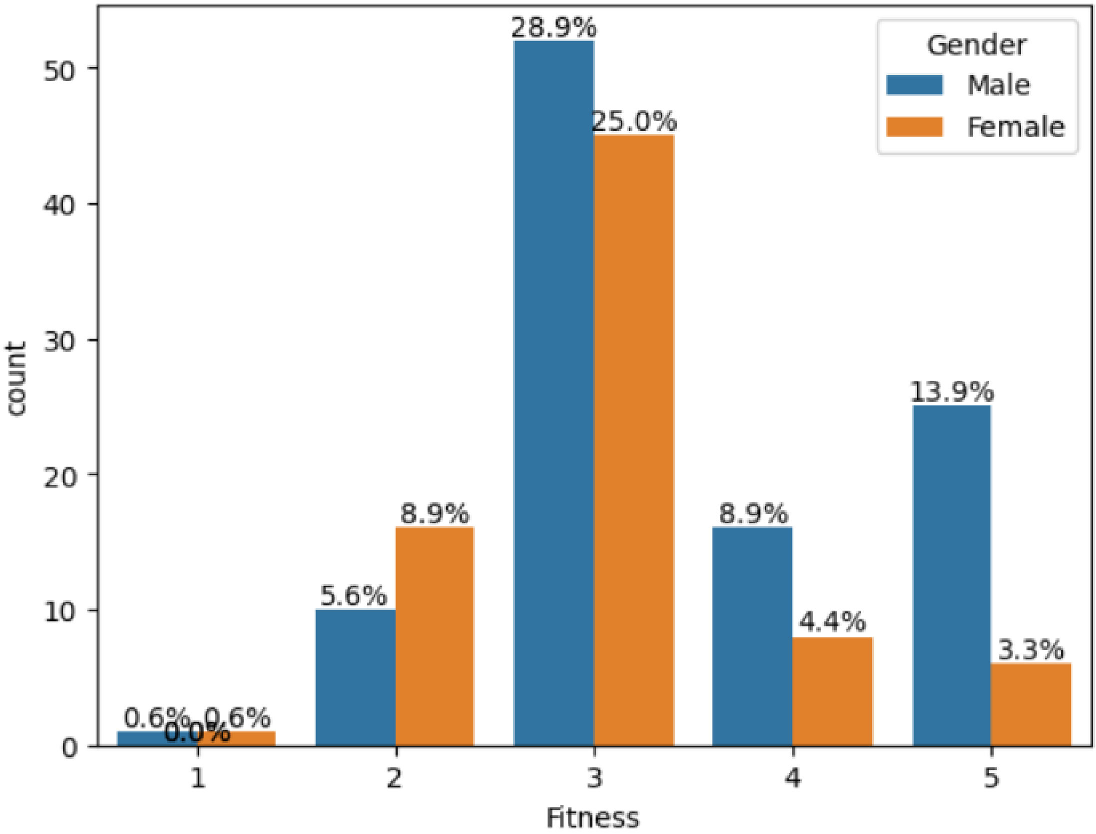


a. Find if there is any relationship between the categorical variables and the output variable in the data.

➢ Most of the Male & Female both has average Fitness score 3

```
[30] ax=sbn.countplot(data=df, x='Fitness', hue='Gender')
     for p in ax.patches:
         height = p.get_height()
         percentage = '{:.1f}%'.format(100 * height /len(df))
         x = p.get_x() + p.get_width() / 2
         ax.text(x, height, percentage, ha='center', va='bottom')
     plt.show()
```

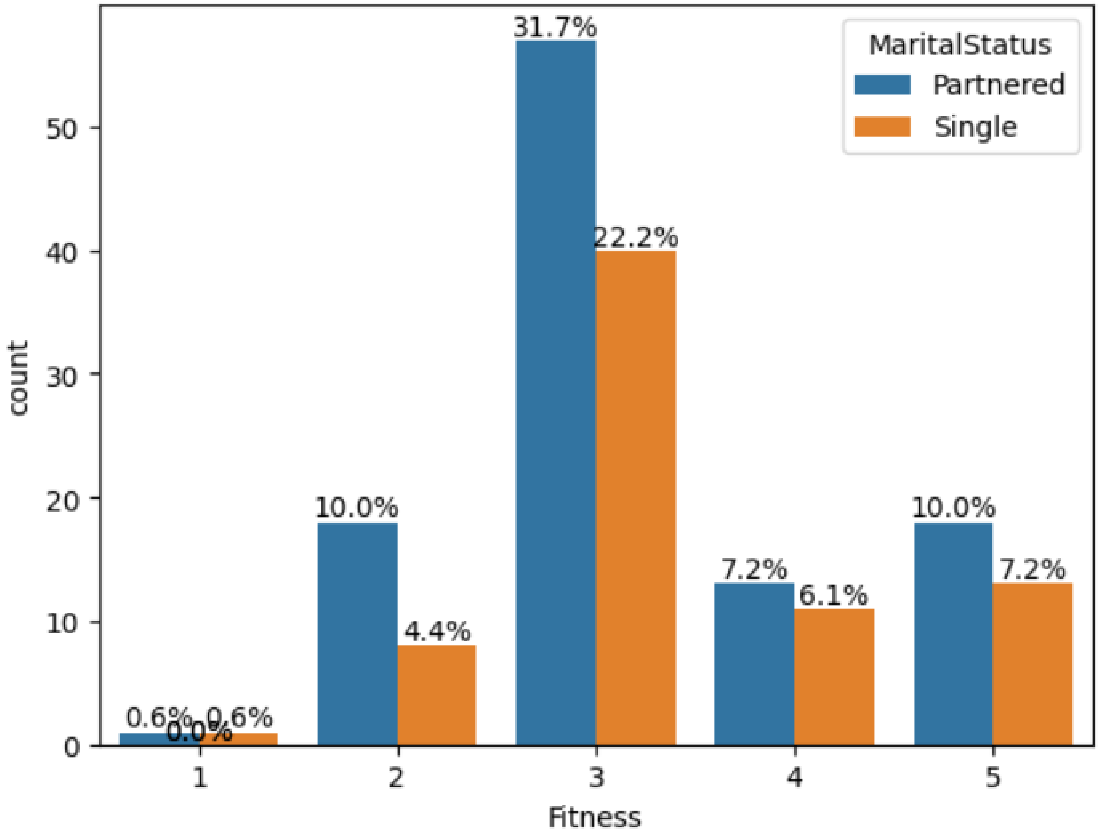
```
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning
  data_subset = grouped_data.get_group(pd_key)
```



➢ Partnered people are more average fit than single people.

```
[31] ax=sbn.countplot(data=df, x='Fitness', hue='MaritalStatus')
     for p in ax.patches:
         height = p.get_height()
         percentage = '{:.1f}%'.format(100 * height /len(df))
         x = p.get_x() + p.get_width() / 2
         ax.text(x, height, percentage, ha='center', va='bottom')
     plt.show()
```

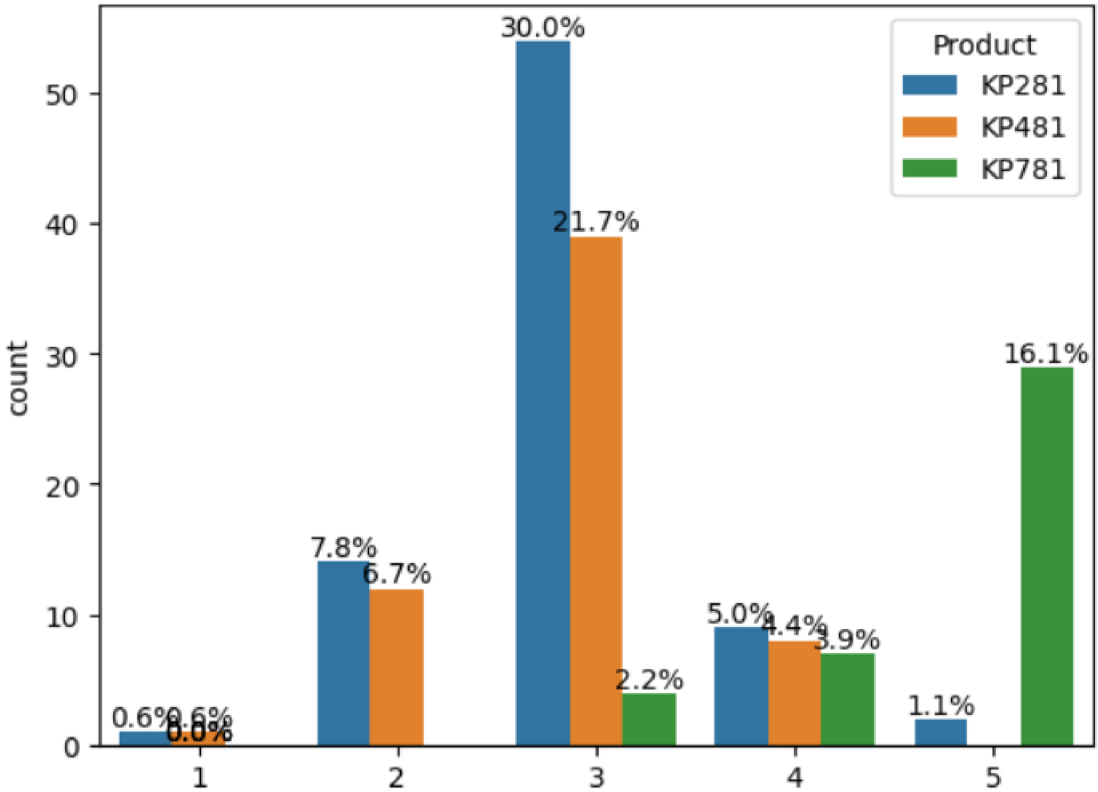
```
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning:
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning:
  data_subset = grouped_data.get_group(pd_key)
```



- People using product KP281 are more average fit, followed by people using product KP481 but people using KP781 are most fit.

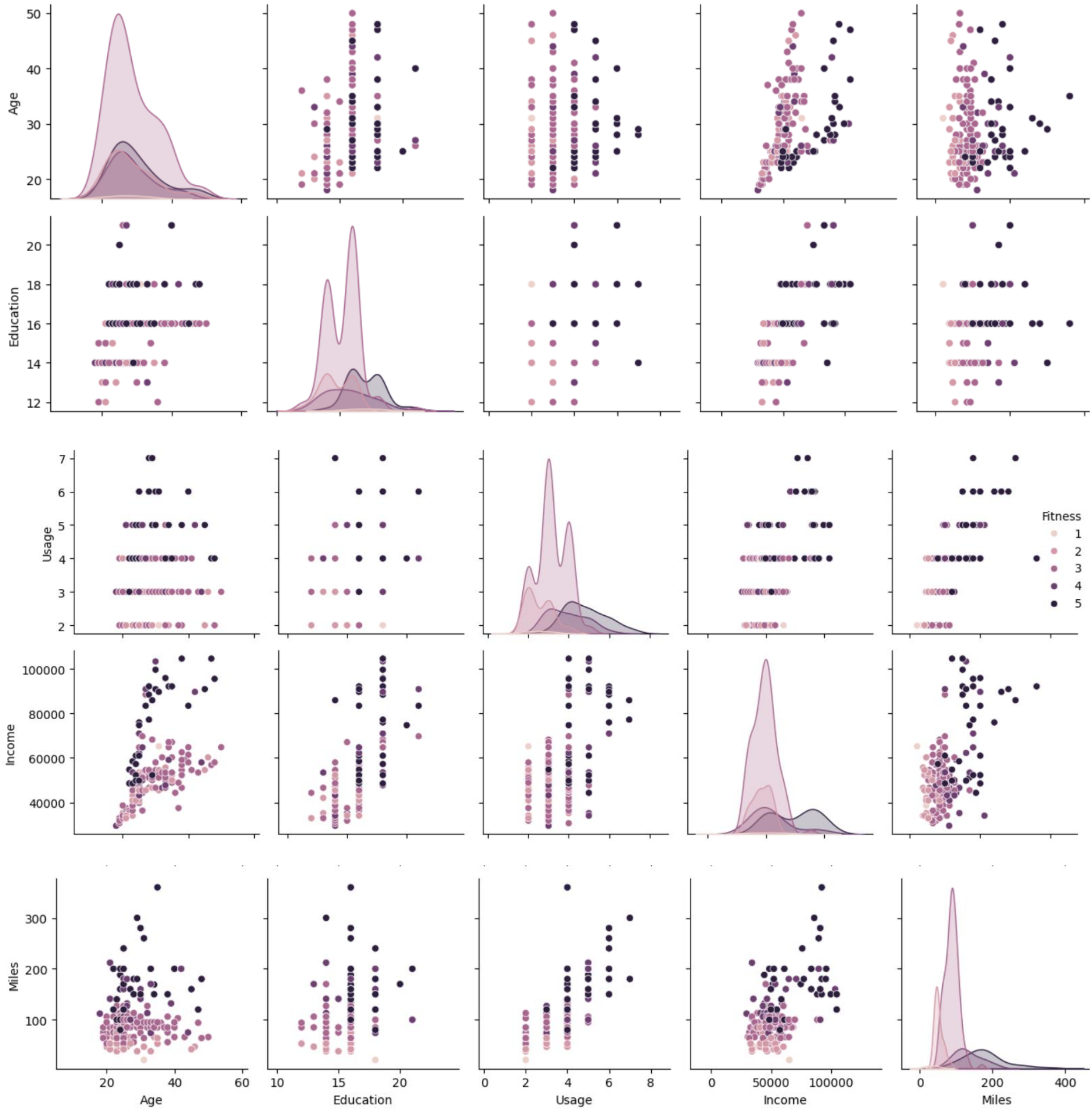
```
[32] ax=sbn.countplot(data=df, x='Fitness', hue='Product')
     for p in ax.patches:
         height = p.get_height()
         percentage = '{:.1f}%'.format(100 * height /len(df))
         x = p.get_x() + p.get_width() / 2
         ax.text(x, height, percentage, ha='center', va='bottom')
     plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning:
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning:
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning:
  data_subset = grouped_data.get_group(pd_key)
```



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

- Visualization showing relation between continuous variable & output variable.
- Almost closed to 100 people are average fit.
- People between age group 20-40 are likely to have more fit rating.
- People having income between 30K to 70K are more fit.
- People walking 50-200 miles per week are observed more fit.



4. Representing the Probability

a. Find the marginal probability (what percent of customers have purchased KP281, KP481, or KP781)

```
[27] product_percentage = pd.crosstab(df['Product'], df['Gender']).sum(axis=1).div(len(df)) * 100

     print(product_percentage)
```

```
Product
KP281    44.444444
KP481    33.333333
KP781    22.222222
dtype: float64
```

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

```
[32] # Probability for Gender
     probability = pd.crosstab(df['Product'], df['Gender'], normalize='columns') * 100
     probability
```

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

```
[33] # Probability for MaritalStatus
     probability = pd.crosstab(df['Product'], df['MaritalStatus'], normalize='columns') * 100
     probability
```

MaritalStatus	Partnered	Single
Product		
KP281	44.859813	43.835616
KP481	33.644860	32.876712
KP781	21.495327	23.287671

[35]
```python
# Probability for Education
probability = pd.crosstab(df['Product'], df['Education'], normalize='columns') * 100
probability
```

Education Product	12	13	14	15	16	18	20	21
KP281	66.666667	60.0	54.545455	80.0	45.882353	8.695652	0.0	0.0
KP481	33.333333	40.0	41.818182	20.0	36.470588	8.695652	0.0	0.0
KP781	0.000000	0.0	3.636364	0.0	17.647059	82.608696	100.0	100.0

[36]
```python
# Probability for Usage
probability = pd.crosstab(df['Product'], df['Usage'], normalize='columns') * 100
probability
```

Usage Product	2	3	4	5	6	7
KP281	57.575758	53.623188	42.307692	11.764706	0.0	0.0
KP481	42.424242	44.927536	23.076923	17.647059	0.0	0.0
KP781	0.000000	1.449275	34.615385	70.588235	100.0	100.0

```python
# Probability for Fitness
probability = pd.crosstab(df['Product'], df['Fitness'], normalize='columns') * 100
probability
```

Fitness Product	1	2	3	4	5
KP281	50.0	53.846154	55.670103	37.500000	6.451613
KP481	50.0	46.153846	40.206186	33.333333	0.000000
KP781	0.0	0.000000	4.123711	29.166667	93.548387

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)

For Gender:

```
[40] conditional_prob = pd.crosstab(df['Gender'], df['Product'], normalize='index') * 100

     print(conditional_prob)
```

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

```
Start coding or generate with AI.
```

For MaritalStatus:

```
[42] conditional_prob = pd.crosstab(df['MaritalStatus'], df['Product'], normalize='index') * 100

     print(conditional_prob)
```

```
Product            KP281      KP481      KP781
MaritalStatus
Partnered      44.859813  33.644860  21.495327
Single         43.835616  32.876712  23.287671
```

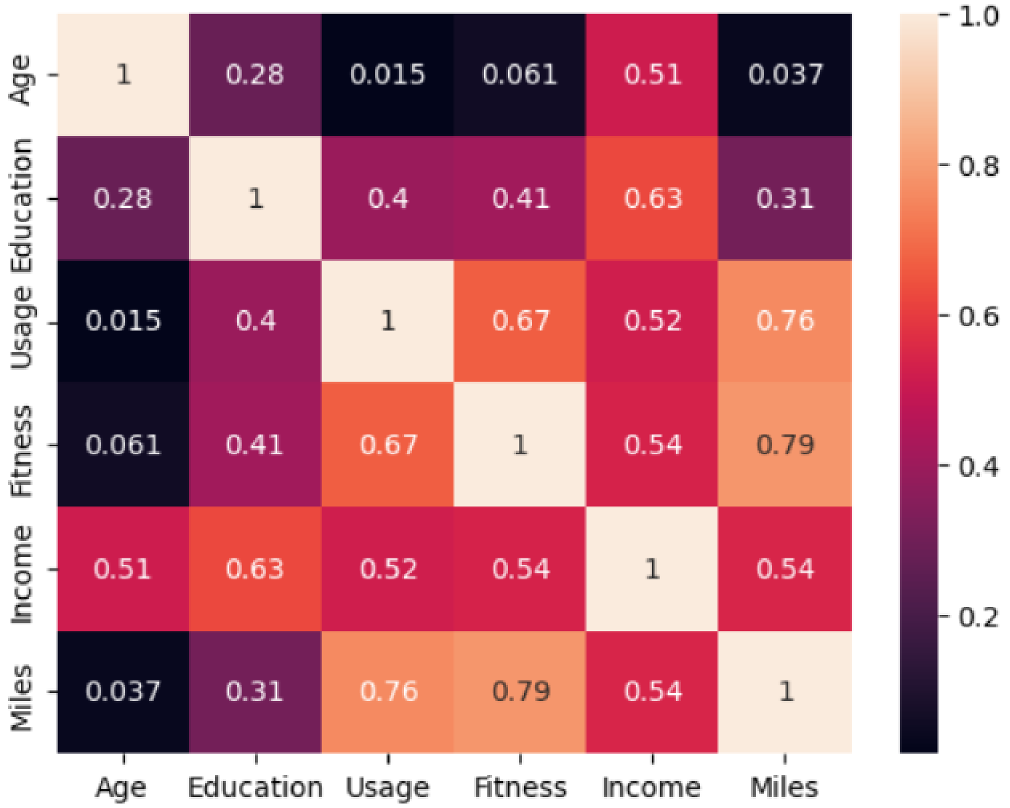
5. Check the correlation among different factors

Find the correlation between the given features in the table.

- Age & Income has strong positive correlation.
- Age has weak positive correlation with Usage, Fitness & Miles.

```
[56] sbn.heatmap( df.select_dtypes(include='number').corr(),annot=True)
```

```
<Axes: >
```



6. Customer profiling and recommendation

a. Make customer profilings for each and every product.

KP281:

- Male & Female bought equally
- People at median age 26 purchase most.
- People having income between 30K to 60K has purchased more.

KP481:

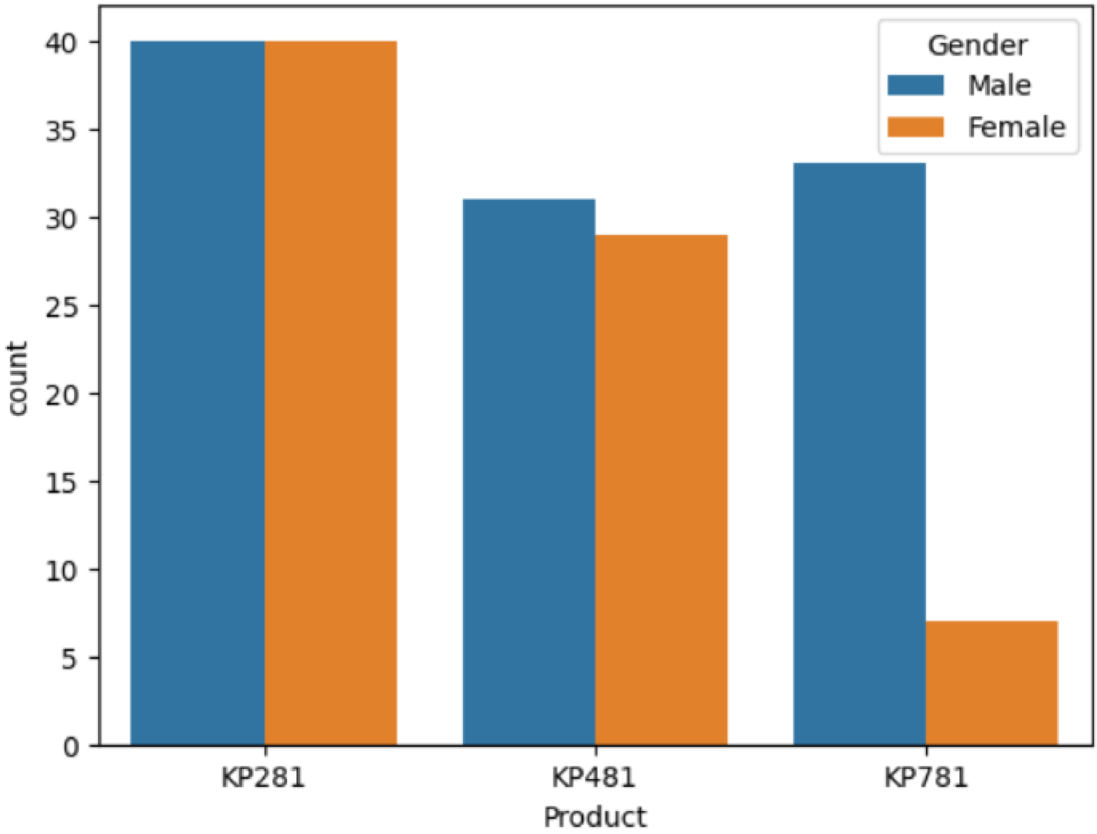
- Male people purchased more than Female.
- People at median age 26 purchase most.
- Very few purchased by people having income between 50K to 65K.

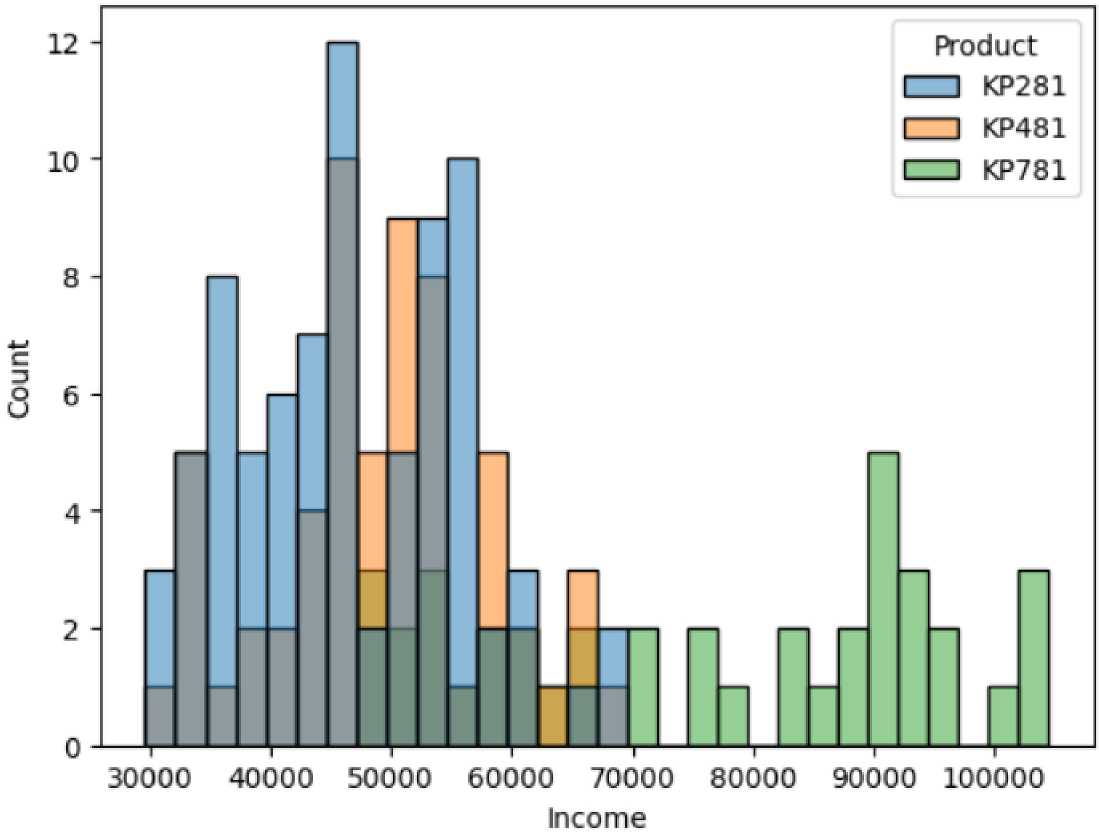
KP781:

- Male people purchase much more than female.
- People at median age 27 purchase most.
- Rich people, Income between 70K to 100K purchased more.

```
sbn.countplot(data=df, x='Product', hue='Gender')
plt.show()
```

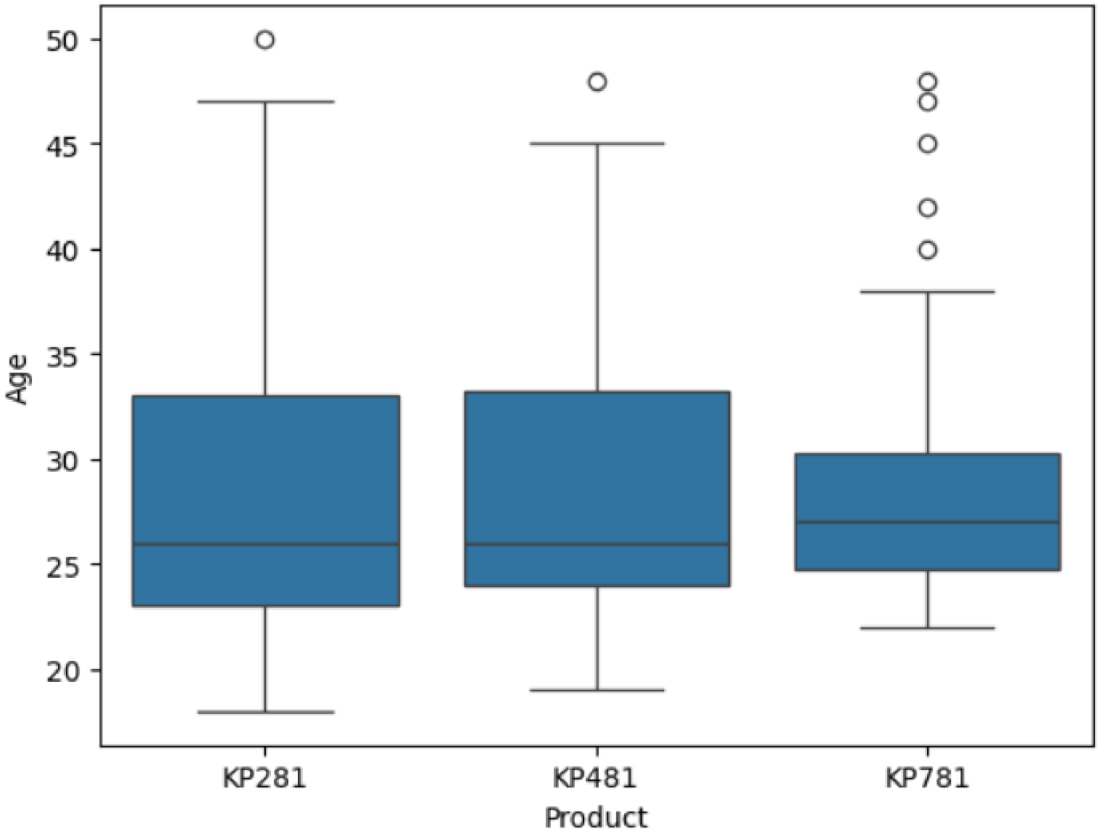
```
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning
  data_subset = grouped_data.get_group(pd_key)
```





```
sbn.boxplot(data=df, x='Product', y='Age')
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:640: FutureWar
  positions = grouped.grouper.result_index.to_numpy(dtype=float)
```



b. Write a detailed recommendation from the analysis that you have done.

- As the products were more purchased by Partnered & male people, therefor to increase sale more, some promotional offers must be given to them.
- Rich people most purchased KP781 which is most costly product, so some combo offer should be given to them to increase sale.
- From correlation matrix it found that, people at high age are less fit but are richer, therefor some offers must be given to them to increase.
- People almost purchased 50% which are average fit, so some guidance must be given to them to increase sale more.