```
!gdown 1hR_60jiBA_-NYy4RA9vKjYelxkwXcfLF -O 'Aerofit_treadmill.csv'
```

```
Downloading...
From: https://drive.google.com/uc?id=1hR_60jiBA_-NYy4RA9vKjYelxkwXcfLF
To: /content/Aerofit_treadmill.csv
100% 7.28k/7.28k [00:00<00:00, 50.4MB/s]
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
aft = pd.read_csv('Aerofit_treadmill.csv')
aft.head()
```

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitr
0	KP281	18	Male	14	Single	3	
1	KP281	19	Male	15	Single	2	
2	KP281	19	Female	14	Partnered	4	

```
aft.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
```

Saved successfully!

	Age	Education	Usage	Fitness	I
count	180.000000	180.000000	180.000000	180.000000	180.0
mean	28.788889	15.572222	3.455556	3.311111	53719.5
std	6.943498	1.617055	1.084797	0.958869	16506.6
min	18.000000	12.000000	2.000000	1.000000	29562.0
25%	24.000000	14.000000	3.000000	3.000000	44058.7
50%	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

```
aft.describe(include=object)
```

	Product	Gender	MaritalStatus
count	180	180	180
unique	3	2	2
top	KP281	Male	Partnered

```
aft.shape
```

```
(180, 9)
```

```
# Age

# 15-25 - Youth
# 25-35 - Middle Age Adults
# 35-55 - Older Adults

# Income

# 10k-30k                - Lower Middle
# 30k-50k                - Middle
# 50k-70k                - Upper Middle
# 70k-90k                - Wealthy
# 90k and 110k           - Very Wealthy

# Education

# 0-12                - Less Educated
# 12-16               - Moderately Educated
# 16-18               - Highly Educated
# 18-22               - Very Highly Educated
```

```
aft['AgeCategory'] = pd.cut(aft['Age'], bins = [15,25,35,55], labels = ['Youth','Middle Age Adults','Older Adults'])
aft['AgeCategory'] = aft['AgeCategory'].astype(str)
aft.describe(include=object)
```

	Product	Gender	MaritalStatus	AgeCategory
count	180	180	180	180
unique	3	2	2	3
top	KP281	Male	Partnered	Youth
freq	80	104	107	79

```
aft['IncomeCategory'] = pd.cut(aft['Income'], bins = [10000,30000,50000,70000,90000,110000], labels = ['Lower Middle','Middle','Upper Mic
aft['IncomeCategory'] = aft['IncomeCategory'].astype(str)
aft.describe(include=object)
```

	Product	Gender	MaritalStatus	AgeCategory	IncomeCategory
count	180	180	180	180	180
unique	3	2	2	3	5
top	KP281	Male	Partnered	Youth	Middle
			107	79	82

Saved successfully!

```
aft['EducationCategory'] = pd.cut(aft['Education'], bins = [0,12,16,18,22], labels = ['Less Educated','Moderately Educated','Highly Educa
aft['EducationCategory'] = aft['EducationCategory'].astype(str)
aft.describe(include=object)
```

	Product	Gender	MaritalStatus	AgeCategory	IncomeCategory	EducationCategory
count	180	180	180	180	180	180
unique	3	2	2	3	5	4
top	KP281	Male	Partnered	Youth	Middle	Moderately Educated
freq	80	104	107	79	82	150

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

```
KP281    80
KP481    60
KP781    40
Name: Product, dtype: int64
```

```
aft['Gender'].value_counts()
```

```
Male      104
Female     76
Name: Gender, dtype: int64
```

```
aft['MaritalStatus'].value_counts()
```

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

```
aft['AgeCategory'].value_counts()
```

```
Youth                79
Middle Age Adults    73
Older Adults         28
Name: AgeCategory, dtype: int64
```

```
aft['EducationCategory'].value_counts()
```

```
Moderately Educated     150
Highly Educated          23
Very Highly Educated      4
Less Educated             3
Name: EducationCategory, dtype: int64
```

```
aft['IncomeCategory'].value_counts()
```

```
Middle          82
Upper Middle    74
Very Wealthy    12
Wealthy         11
Lower Middle     1
Name: IncomeCategory, dtype: int64
```

```
aft['Usage'].value_counts()
```

```
3    69
4    52
2    33
5    17
6     7
7     2
Name: Usage, dtype: int64
```

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

```
3    97
5    31
2    26
4    24
1     2
Name: Fitness, dtype: int64
```

```
aft['Product'].unique()
```

Saved successfully!

```
'], dtype=object)
```

```
aft['Gender'].unique()
```

```
array(['Male', 'Female'], dtype=object)
```

```
aft['MaritalStatus'].unique()
```

```
array(['Single', 'Partnered'], dtype=object)
```

```
aft['AgeCategory'].unique()
```

```
array(['Youth', 'Middle Age Adults', 'Older Adults'], dtype=object)
```

```
aft['EducationCategory'].unique()
```

```
array(['Moderately Educated', 'Less Educated', 'Highly Educated',
       'Very Highly Educated'], dtype=object)
```

```
aft['IncomeCategory'].unique()
```

```
array(['Lower Middle', 'Middle', 'Upper Middle', 'Wealthy',
       'Very Wealthy'], dtype=object)
```

```
np.any(aft.isna().any(axis=1))

# No Missing Value
```

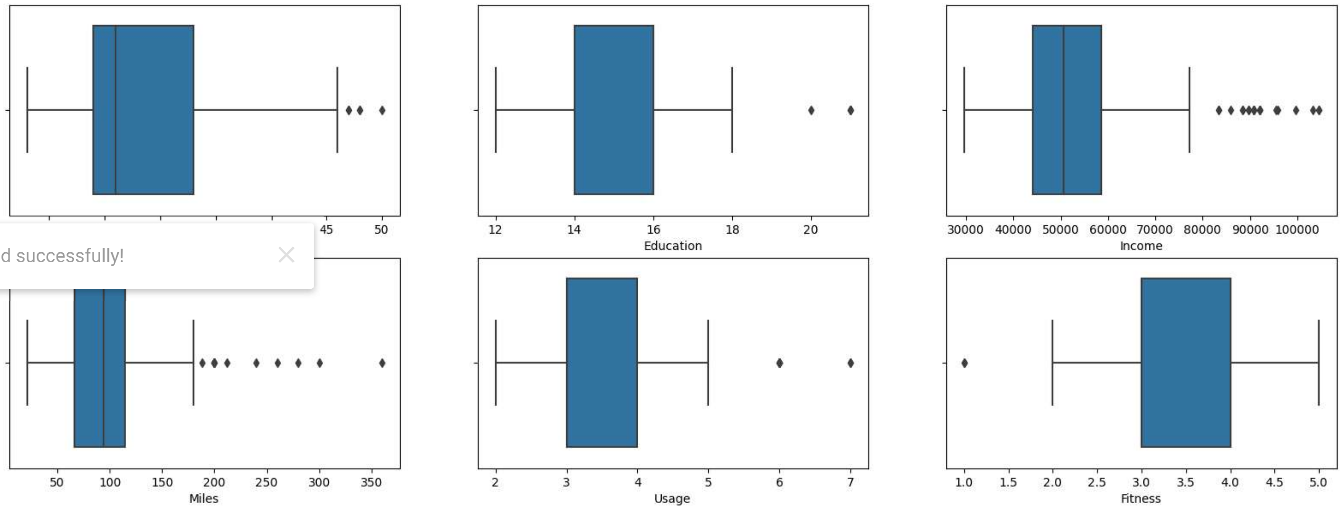
```
False
```

```
aft.isna().sum()
```

```
Product              0
Age                  0
Gender               0
Education            0
MaritalStatus        0
Usage                0
Fitness              0
Income               0
Miles                0
AgeCategory          0
IncomeCategory       0
EducationCategory    0
dtype: int64
```

```
#Outlier

fig = plt.figure(figsize=(20,7))

plt.subplot(2,3, 1)
sns.boxplot(data = aft , x ='Age' )

plt.subplot(2,3, 2)
sns.boxplot(data = aft , x = 'Education' )

plt.subplot(2,3, 3)
sns.boxplot(data = aft , x = 'Income' )

plt.subplot(2,3, 4)
sns.boxplot(data = aft , x = 'Miles' )

plt.subplot(2,3, 5)
sns.boxplot(data = aft , x = 'Usage' )

plt.subplot(2,3, 6)
sns.boxplot(data = aft , x = 'Fitness' )
```

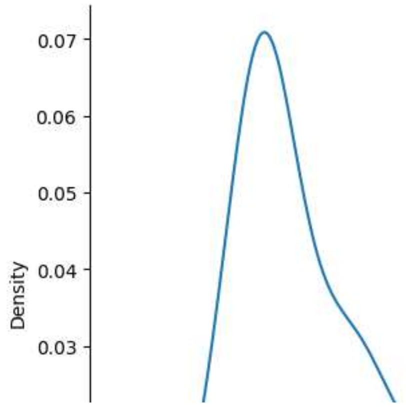
```
<Axes: xlabel='Fitness'>
```



Saved successfully!

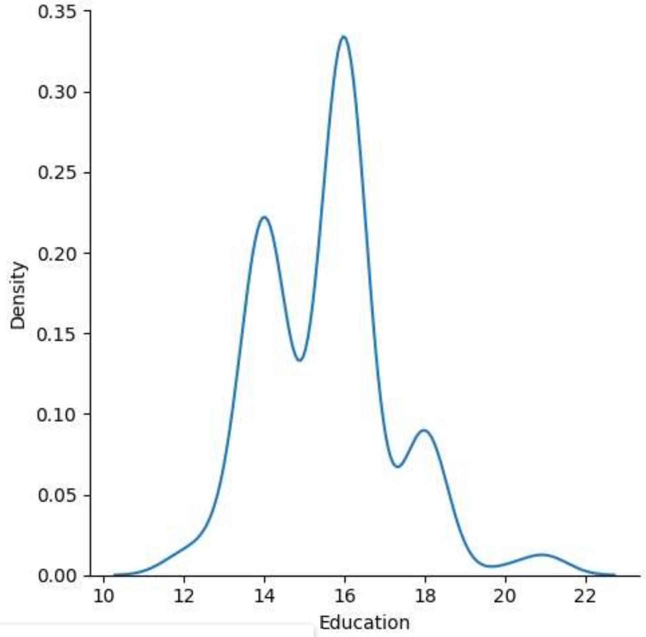
```
sns.displot(aft['Age'],kind='kde')
```

```
<seaborn.axisgrid.FacetGrid at 0x7fc5fa3a6080>
```



```
sns.displot(aft['Education'],kind='kde')
```

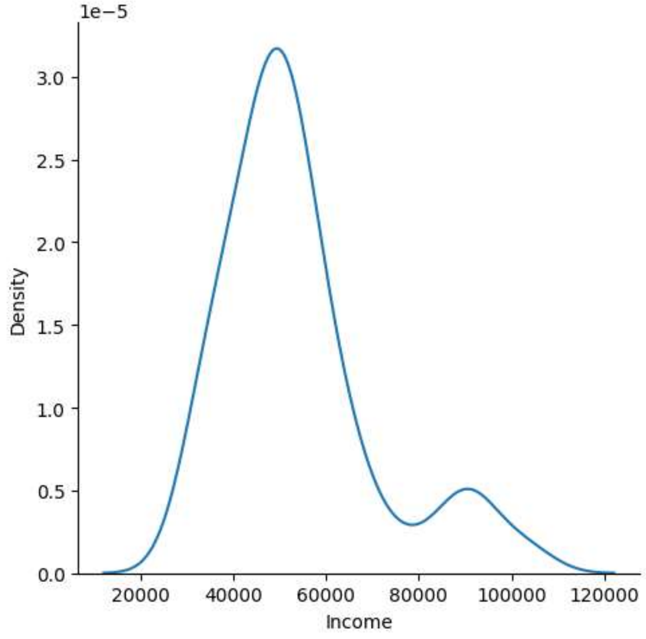
```
<seaborn.axisgrid.FacetGrid at 0x7fc5fa212710>
```



Saved successfully! ×

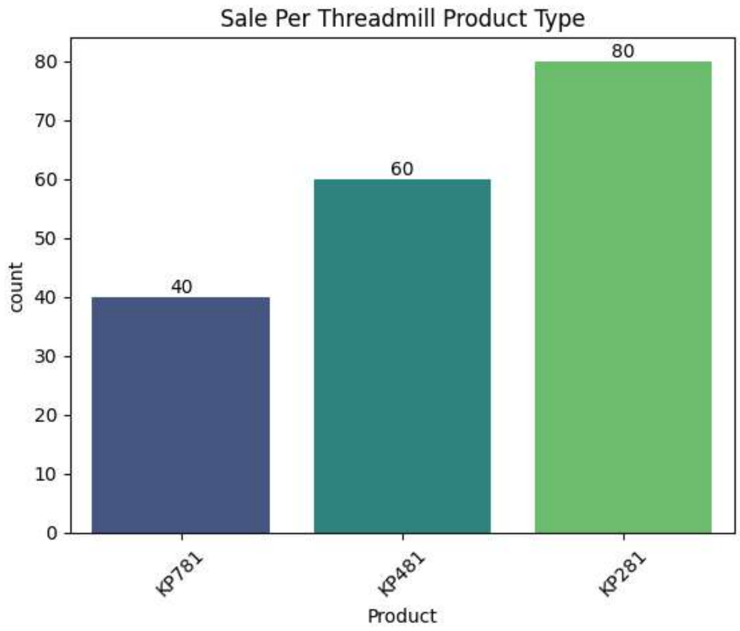
```
sns.displot(aft['Income'],kind='kde')
```

```
<seaborn.axisgrid.FacetGrid at 0x7fc638aef6a0>
```



```
ax = sns.countplot(data = aft,
              x = "Product",
             order = list(aft["Product"].value_counts().index)[::-1],
             palette= "viridis")
```

```
plt.xticks(rotation = 45, fontsize = 10)


for i in ax.containers:
    ax.bar_label(i, fontsize = 10)


plt.title("Sale Per Threadmill Product Type")

plt.show()
```

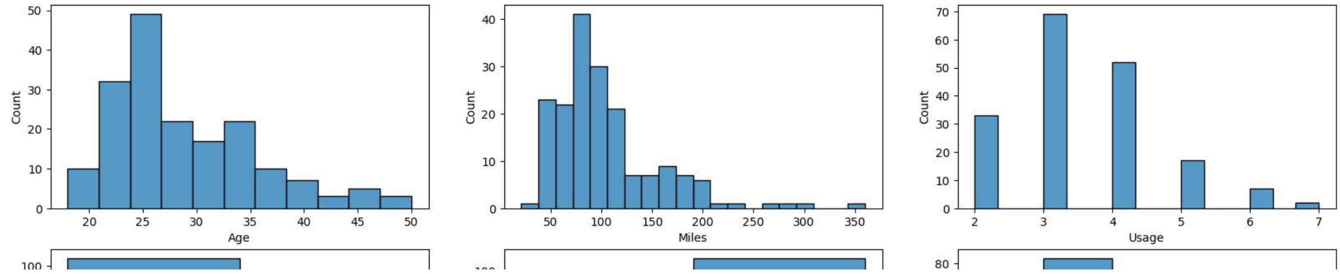


```
fig = plt.figure(figsize=(20,7))

plt.subplot(2,3, 1)
sns.histplot(data = aft , x ='Age' )

plt.subplot(2,3, 2)
                                          ' )
```

Saved successfully!

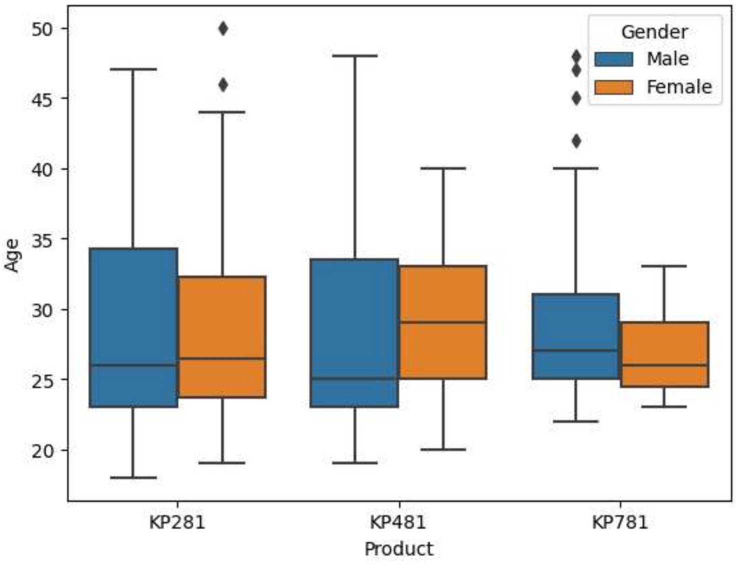
```
plt.subplot(2,3, 3)
sns.histplot(data = aft , x = 'Usage' )

plt.subplot(2,3, 4)
sns.histplot(data = aft , x = 'Gender' )

plt.subplot(2,3, 5)
sns.histplot(data = aft , x = 'MaritalStatus' )

plt.subplot(2,3, 6)
sns.histplot(data = aft , x = 'IncomeCategory' )
```

```
<Axes: xlabel='IncomeCategory', ylabel='Count'>
```

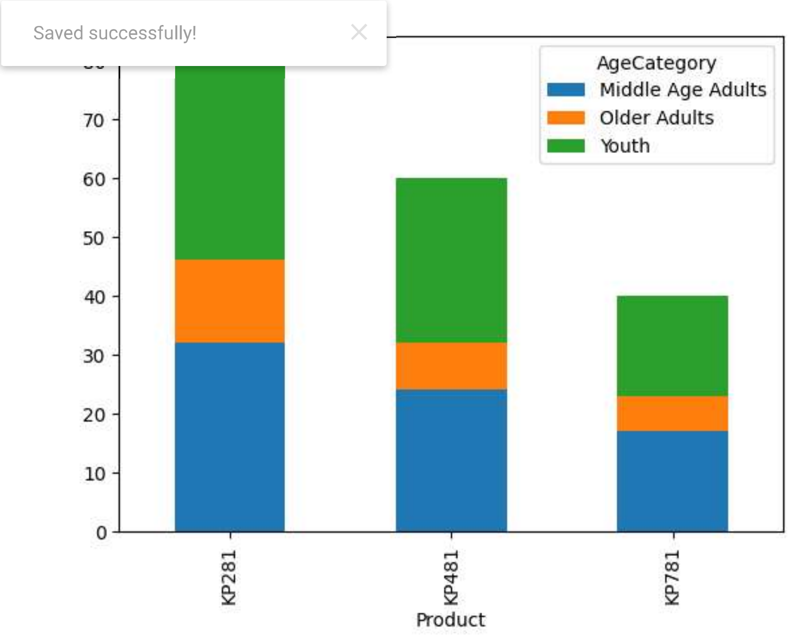


```
sns.boxplot(data=aft,
           x="Product",
           y="Age",
            hue = "Gender")
```

```
<Axes: xlabel='Product', ylabel='Age'>
```



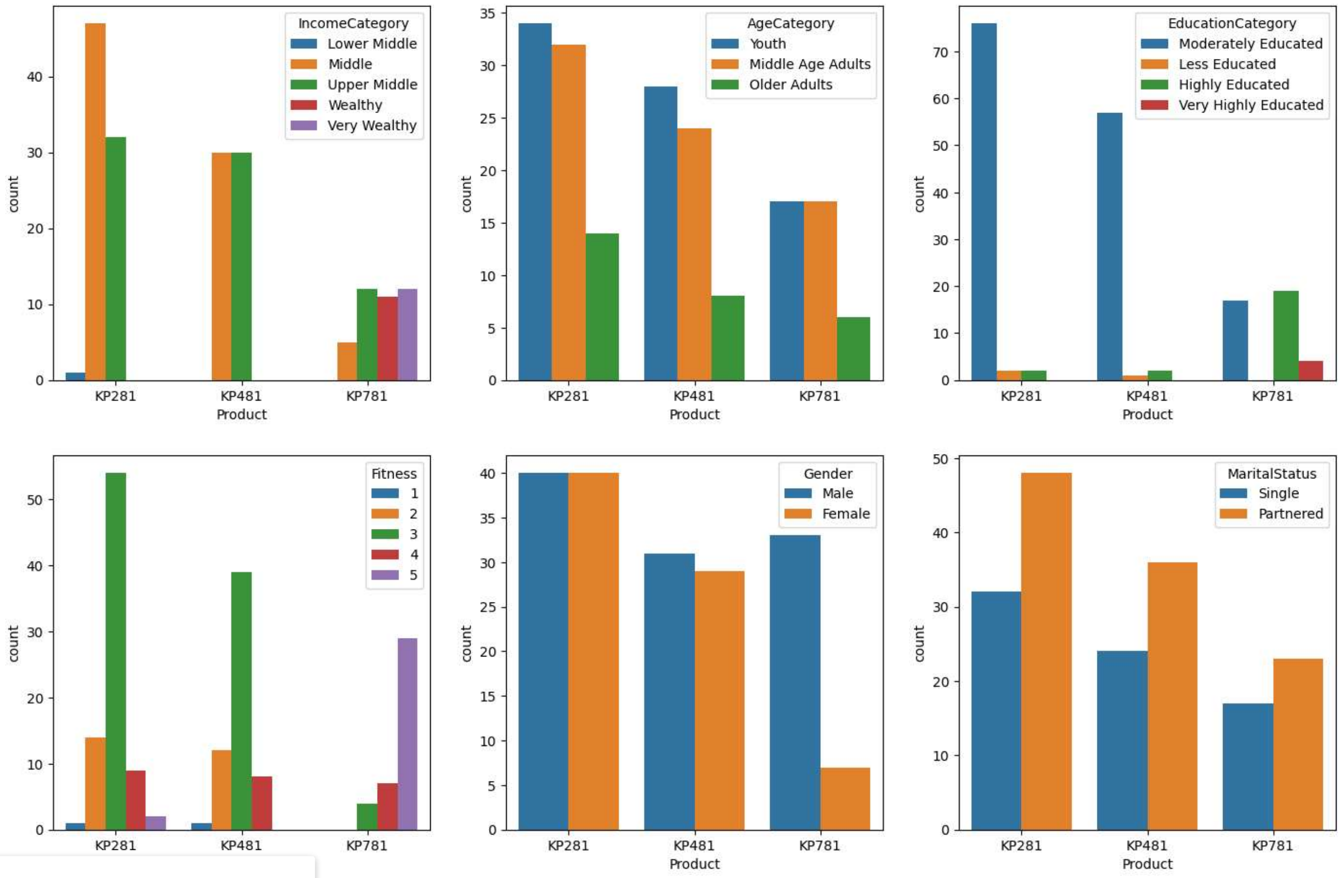
```
df_plot = pd.crosstab(index = aft["Product"], columns=aft["AgeCategory"])
df_plot.plot(kind="bar",stacked=True)
```


Saved successfully!

```
fig = plt.figure(figsize=(17,11))

plt.subplot(2,3, 1)
sns.countplot(data=aft, x="Product", hue = "IncomeCategory")

plt.subplot(2,3, 2)
sns.countplot(data=aft, x="Product", hue = "AgeCategory")

plt.subplot(2,3, 3)
```

```
sns.countplot(data=aft, x="Product", hue = "EducationCategory")

plt.subplot(2,3, 4)
sns.countplot(data=aft, x="Product", hue = "Fitness")

plt.subplot(2,3, 5)
sns.countplot(data=aft, x="Product", hue = "Gender")

plt.subplot(2,3, 6)
sns.countplot(data=aft, x="Product", hue = "MaritalStatus")
```

```
<Axes: xlabel='Product', ylabel='count'>
```



Saved successfully!

MARGINAL AND JOINT PROBABILITIES

```
np.round(pd.crosstab(index = aft["Product"], columns=aft["Gender"],margins=True,normalize = "all")*100,2)
```

Gender	Female	Male	All
Product			
KP281	22.22	22.22	44.44
KP481	16.11	17.22	33.33
KP781	3.89	18.33	22.22
All	42.22	57.78	100.00

```
np.round(pd.crosstab(index = aft["Product"], columns=aft["MaritalStatus"],margins=True,normalize = "all")*100,2)
```

MaritalStatus	Partnered	Single	All
Product			
KP281	26.67	17.78	44.44
KP481	20.00	13.33	33.33
KP781	12.78	9.44	22.22
All	59.44	40.56	100.00

```
np.round(pd.crosstab(index = aft["Product"], columns=aft["AgeCategory"],margins=True,normalize = "all")*100,2)
```

AgeCategory	Middle Age Adults	Older Adults	Youth	All
Product				
KP281	17.78	7.78	18.89	44.44
KP481	13.33	4.44	15.56	33.33
KP781	9.44	3.33	9.44	22.22
All	40.56	15.56	43.89	100.00

```
np.round(pd.crosstab(index = aft["Product"], columns=aft["IncomeCategory"],margins=True,normalize = "all")*100,2)
```

IncomeCategory	Lower Middle	Middle	Upper Middle	Very Wealthy	Wealthy	All
Product						
KP281	0.56	26.11	17.78	0.00	0.00	44.44
KP481	0.00	16.67	16.67	0.00	0.00	33.33
KP781	0.00	2.78	6.67	6.67	6.11	22.22
All	0.56	45.56	41.11	6.67	6.11	100.00

```
np.round(pd.crosstab(index = aft["Product"], columns=aft["EducationCategory"],margins=True,normalize = "all")*100,2)
```

EducationCategory	Highly Educated	Less Educated	Moderately Educated	Very Highly Educated	All
Product					
KP281	1.11	1.11	42.22	0.00	44.44
KP481	1.11	0.56	31.67	0.00	33.33
	10.56	0.00	9.44	2.22	22.22
All	12.78	1.67	83.33	2.22	100.00

Saved successfully! ×

```
np.round(pd.crosstab(index = aft["Product"], columns=aft["Fitness"],margins=True,normalize = "all")*100,2)
```

Fitness	1	2	3	4	5	All
Product						
KP281	0.56	7.78	30.00	5.00	1.11	44.44
KP481	0.56	6.67	21.67	4.44	0.00	33.33
KP781	0.00	0.00	2.22	3.89	16.11	22.22
All	1.11	14.44	53.89	13.33	17.22	100.00

```
np.round(pd.crosstab(index = aft["Product"], columns=aft["Usage"],margins=True,normalize = "all")*100,2)
```

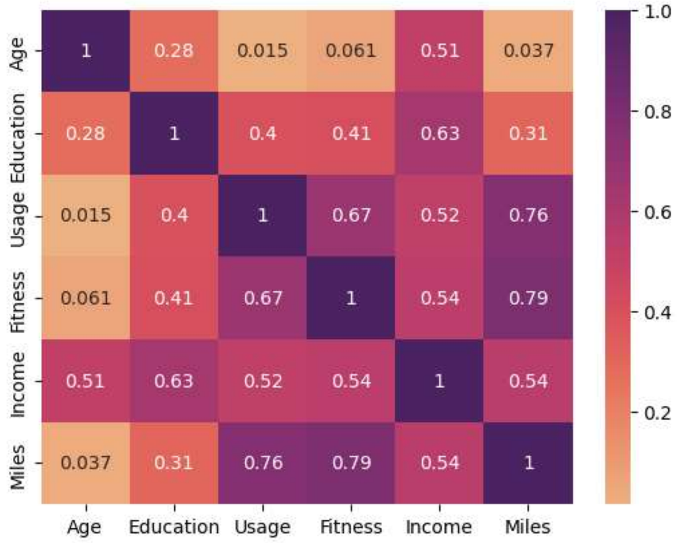
Usage	2	3	4	5	6	7	All
Product							
KP281	10.56	20.56	12.22	1.11	0.00	0.00	44.44
KP481	7.78	17.22	6.67	1.67	0.00	0.00	33.33
KP781	0.00	0.56	10.00	6.67	3.89	1.11	22.22
All	18.33	38.33	28.89	9.44	3.89	1.11	100.00

```
np.round(pd.crosstab(index = aft["Usage"], columns=aft["Fitness"],margins=True,normalize = "all")*100,2)
```

Fitness	1	2	3	4	5	All
Usage						
2	0.56	7.78	10.00	0.00	0.00	18.33
3	0.56	5.56	26.11	5.56	0.56	38.33
4	0.00	1.11	16.67	3.89	7.22	28.89
5	0.00	0.00	1.11	3.33	5.00	9.44
6	0.00	0.00	0.00	0.56	3.33	3.89
7	0.00	0.00	0.00	0.00	1.11	1.11

```
sns.heatmap(aft.corr(),annot=True, cmap="flare")
```

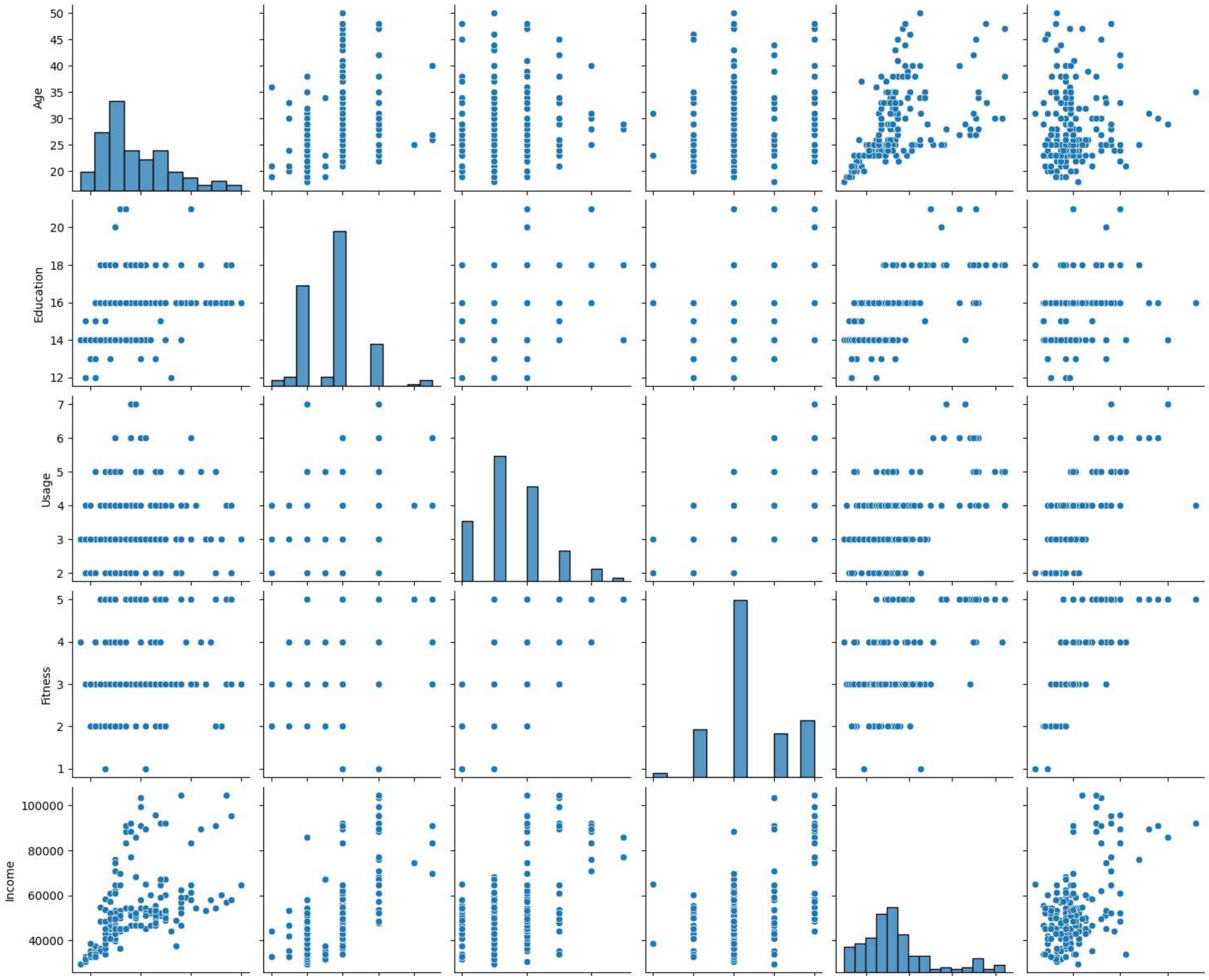
```
<ipython-input-45-1692e9f89b3e>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future ve
  sns.heatmap(aft.corr(),annot=True, cmap="flare")
<Axes: >
```



```
plt.figure(figsize=(8,8))
sns.pairplot(data=aft)
```

Saved successfully!

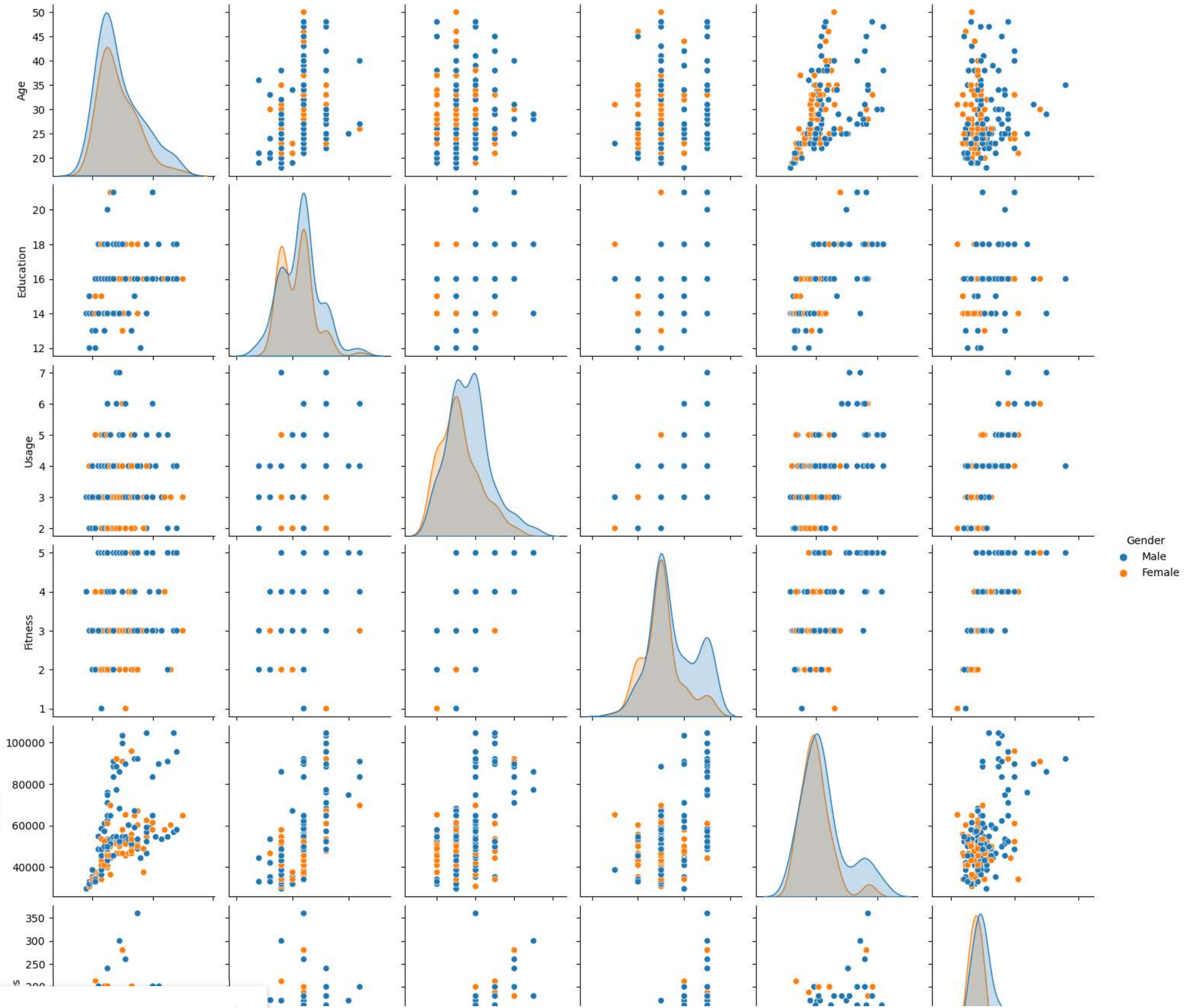
```
<seaborn.axisgrid.PairGrid at 0x7fc5f4c83580>
<Figure size 800x800 with 0 Axes>
```



Saved successfully!

```
r")
```

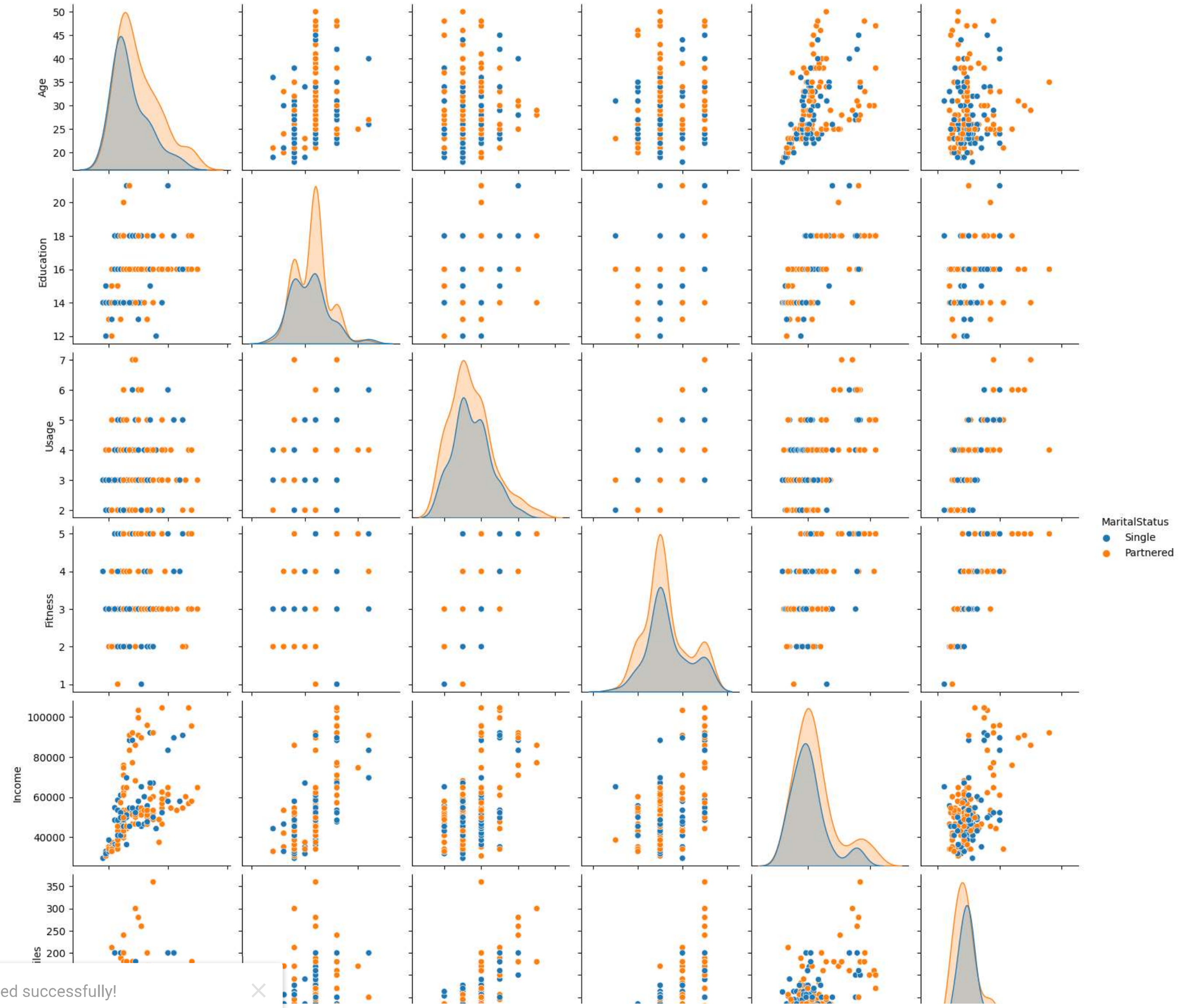
```
<seaborn.axisgrid.PairGrid at 0x7fc5f54b77c0>
<Figure size 800x800 with 0 Axes>
```



Saved successfully!

```
sns.pairplot(data=aft , hue = "MaritalStatus")
```

```
<seaborn.axisgrid.PairGrid at 0x7fc5f54b7f40>
<Figure size 800x800 with 0 Axes>
```



Saved successfully!

Conditional Probability of Parameters | given Treadmill Product type

P(Parameter|Product)

```
P_AgeCat_Given_Product = pd.crosstab(index = aft["Product"], columns=aft["AgeCategory"]).apply(lambda x: x/x.sum(),axis=1)
P_AgeCat_Given_Product["Total"] = P_AgeCat_Given_Product.apply(lambda x: x.sum(),axis=1)
P_AgeCat_Given_Product
```

AgeCategory	Middle Age Adults	Older Adults	Youth	Total
Product				
KP281	0.400	0.175000	0.425000	1.0
KP481	0.400	0.133333	0.466667	1.0
KP781	0.425	0.150000	0.425000	1.0

```
P_Gender_Given_Product = pd.crosstab(index = aft["Product"], columns=aft["Gender"]).apply(lambda x: x/x.sum(),axis=1)
P_Gender_Given_Product["Total"] = P_Gender_Given_Product.apply(lambda x: x.sum(),axis=1)
P_Gender_Given_Product
```

```
P_EduCat_Given_Product = pd.crosstab(index = aft["Product"], columns=aft["EducationCategory"]).apply(lambda x: x/x.sum(),axis=1)
P_EduCat_Given_Product["Total"] = P_EduCat_Given_Product.apply(lambda x: x.sum(),axis=1)
P_EduCat_Given_Product
```

EducationCategory	Highly Educated	Less Educated	Moderately Educated	Very Highly Educated	Total
Product					
KP281	0.025000	0.025000	0.950	0.0	1.0
KP481	0.033333	0.016667	0.950	0.0	1.0
KP781	0.475000	0.000000	0.425	0.1	1.0

```
P_MarStatus_Given_Product = pd.crosstab(index = aft["Product"], columns=aft["MaritalStatus"]).apply(lambda x: x/x.sum(),axis=1)
P_MarStatus_Given_Product["Total"] = P_MarStatus_Given_Product.apply(lambda x: x.sum(),axis=1)
P_MarStatus_Given_Product
```

MaritalStatus	Partnered	Single	Total
Product			
KP281	0.600	0.400	1.0
KP481	0.600	0.400	1.0
KP781	0.575	0.425	1.0

```
P_Usage_Given_Product = pd.crosstab(index = aft["Product"], columns=aft["Usage"]).apply(lambda x: x/x.sum(),axis=1)
P_Usage_Given_Product["Total"] = P_Usage_Given_Product.apply(lambda x: x.sum(),axis=1)
P_Usage_Given_Product
```

Usage	2	3	4	5	6	7	Total
Product							
KP281	0.237500	0.462500	0.275	0.025	0.000	0.00	1.0
KP481	0.233333	0.516667	0.200	0.050	0.000	0.00	1.0
KP781	0.000000	0.025000	0.450	0.300	0.175	0.05	1.0

```
P_Fitness_Given_Product = pd.crosstab(index = aft["Product"], columns=aft["Fitness"]).apply(lambda x: x/x.sum(),axis=1)
P_Fitness_Given_Product["Total"] = P_Fitness_Given_Product.apply(lambda x: x.sum(),axis=1)
P_Fitness_Given_Product
```

Saved successfully! ✕

Fitness	1	2	3	4	5	Total
Product						
KP281	0.012500	0.175	0.675	0.112500	0.025	1.0
KP481	0.016667	0.200	0.650	0.133333	0.000	1.0
KP781	0.000000	0.000	0.100	0.175000	0.725	1.0

```
P_IncomeCat_Given_Product = pd.crosstab(index = aft["Product"], columns=aft["IncomeCategory"]).apply(lambda x: x/x.sum(),axis=1)
P_IncomeCat_Given_Product["Total"] = P_IncomeCat_Given_Product.apply(lambda x: x.sum(),axis=1)
P_IncomeCat_Given_Product
```

IncomeCategory	Lower Middle	Middle	Upper Middle	Very Wealthy	Wealthy	Total
Product						
KP281	0.0125	0.5875	0.4	0.0	0.000	1.0
KP481	0.0000	0.5000	0.5	0.0	0.000	1.0
KP781	0.0000	0.1250	0.3	0.3	0.275	1.0

Conditional Probability of Treadmill Product | given other paramaters

P(Product|Parameter)

```
P_Product_Given_AgeCat = pd.crosstab(index = aft["AgeCategory"], columns=aft["Product"]).apply(lambda x: x/x.sum(),axis=1)
P_Product_Given_AgeCat["Total"] = P_Product_Given_AgeCat.apply(lambda x: x.sum(),axis=1)
P_Product_Given_AgeCat
```

Product AgeCategory	KP281	KP481	KP781	Total
Middle Age Adults	0.438356	0.328767	0.232877	1.0
Older Adults	0.500000	0.285714	0.214286	1.0
Youth	0.430380	0.354430	0.215190	1.0

```
P_Product_Given_Gender = pd.crosstab(index = aft["Gender"], columns=aft["Product"]).apply(lambda x: x/x.sum(),axis=1)
P_Product_Given_Gender["Total"] = P_Product_Given_Gender.apply(lambda x: x.sum(),axis=1)
P_Product_Given_Gender
```

Product Gender	KP281	KP481	KP781	Total
Female	0.526316	0.381579	0.092105	1.0
Male	0.384615	0.298077	0.317308	1.0

```
P_Product_Given_Educat = pd.crosstab(index = aft["EducationCategory"], columns=aft["Product"]).apply(lambda x: x/x.sum(),axis=1)
P_Product_Given_Educat["Total"] = P_Product_Given_Educat.apply(lambda x: x.sum(),axis=1)
P_Product_Given_Educat
```

Product EducationCategory	KP281	KP481	KP781	Total
Highly Educated	0.086957	0.086957	0.826087	1.0
Less Educated	0.666667	0.333333	0.000000	1.0
Moderately Educated	0.506667	0.380000	0.113333	1.0
Very Highly Educated	0.000000	0.000000	1.000000	1.0

```
P_Product_Given_MarStatus = pd.crosstab(index = aft["MaritalStatus"], columns=aft["Product"]).apply(lambda x: x/x.sum(),axis=1)
P_Product_Given_MarStatus["Total"] = P_Product_Given_MarStatus.apply(lambda x: x.sum(),axis=1)
P_Product_Given_MarStatus
```

Product MaritalStatus	KP281	KP481	KP781	Total
Partnered	0.448598	0.336449	0.214953	1.0

Saved successfully! ×

```
P_Product_Given_Usage = pd.crosstab(index = aft["Usage"], columns=aft["Product"]).apply(lambda x: x/x.sum(),axis=1)
P_Product_Given_Usage["Total"] = P_Product_Given_Usage.apply(lambda x: x.sum(),axis=1)
P_Product_Given_Usage
```

Product Usage	KP281	KP481	KP781	Total
2	0.575758	0.424242	0.000000	1.0
3	0.536232	0.449275	0.014493	1.0
4	0.423077	0.230769	0.346154	1.0
5	0.117647	0.176471	0.705882	1.0
6	0.000000	0.000000	1.000000	1.0
7	0.000000	0.000000	1.000000	1.0

```
P_Product_Given_Fitness = pd.crosstab(index = aft["Fitness"], columns=aft["Product"]).apply(lambda x: x/x.sum(),axis=1)
P_Product_Given_Fitness["Total"] = P_Product_Given_Fitness.apply(lambda x: x.sum(),axis=1)
P_Product_Given_Fitness
```

```
Product    KP281    KP481    KP781  Total

Fitness
```

```
P_Product_Given_IncomeCat = pd.crosstab(index = aft["IncomeCategory"], columns=aft["Product"]).apply(lambda x: x/x.sum(),axis=1)
P_Product_Given_IncomeCat["Total"] = P_Product_Given_IncomeCat.apply(lambda x: x.sum(),axis=1)
P_Product_Given_IncomeCat
```

```
       Product     KP281     KP481     KP781  Total
IncomeCategory
  Lower Middle  1.000000  0.000000  0.000000    1.0
        Middle  0.573171  0.365854  0.060976    1.0
  Upper Middle  0.432432  0.405405  0.162162    1.0
  Very Wealthy  0.000000  0.000000  1.000000    1.0
       Wealthy  0.000000  0.000000  1.000000    1.0
```

```
df = aft.groupby(["Product","Gender","MaritalStatus","IncomeCategory"])[["Product"]].count()
df.rename(columns={"Product":"ProductCount"},inplace=True)
df.reset_index()
df.sort_values(["ProductCount"],ascending = False)
```

```
                                                ProductCount
Product  Gender  MaritalStatus  IncomeCategory
 KP281   Female    Partnered        Middle                17
 KP481    Male     Partnered     Upper Middle             13
 KP281    Male     Partnered     Upper Middle             11
                    Single          Middle                11
                   Partnered        Middle                10
         Female    Partnered     Upper Middle             10
                    Single          Middle                 9
 KP481    Male     Partnered        Middle                 8
         Female    Partnered     Upper Middle              8
                    Single          Middle                 8
                                    Wealthy                7
                                  Very Wealthy             7
 KP481    Male      Single          Middle                 7
         Female    Partnered        Middle                 7
 KP281    Male      Single       Upper Middle              7
 KP481   Female     Single       Upper Middle              6
 KP281   Female     Single       Upper Middle              4
 KP781    Male     Partnered     Upper Middle              4
                    Single          Middle                 4
                                 Upper Middle              4
                                    Wealthy                4
 KP481    Male      Single       Upper Middle              3
 KP781   Female    Partnered     Very Wealthy              3
                    Single       Upper Middle              3
          Male      Single       Very Wealthy              2
         Female    Partnered     Upper Middle              1
          Male     Partnered        Middle                 1
 KP281    Male      Single       Lower Middle              1
```

Saved successfully! ×

```
df = aft[['Product']].value_counts().reset_index()
df.loc[0,0] = df.loc[0,0] * 1500
df.loc[1,0] = df.loc[1,0] * 1750
df.loc[2,0] = df.loc[2,0] * 2500
df.rename({0: 'Total Revenue'},axis=1,inplace = True)
```