In [374]:
```
#importing libraries for our purpose
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
#loading dataset
df=pd.read_csv('aerofit_treadmill.csv')
```

In [377]:
```
df.head(10)
```

Out[377]:

	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

Problem Statement :

1. Identifying the characteristics of the target audience for each type of treadmill offered by the company, to provide a better recommendation of the treadmills to the new customers.
2. Most popular treadmill
3. Average usage of treadmill per week
4. Average number of miles the customer expects to walk/run each week
5. Which treadmill is purchased by people with what kind of income range?

In [21]:
```
#Shape of dataset
print(f"Number of rows: {df.shape[0]}")
print(f"Number of columns: {df.shape[1]}")
```

```
Number of rows: 180
Number of columns: 9
```

In [10]:
```
#Columns present in dataset
df.columns
```

Out[10]:
```
Index(['Product', 'Age', 'Gender', 'Education', 'MaritalStatus', 'Usage',
       'Fitness', 'Income', 'Miles'],
      dtype='object')
```

In [14]:
```
#checking datatypes
df.dtypes
```

Out[14]:
```
Product          object
Age               int64
Gender           object
Education         int64
MaritalStatus    object
Usage             int64
Fitness           int64
Income            int64
Miles             int64
dtype: object
```

In [17]:
```
#Number of unique values in dataset
for i in df.columns:
    print(i,":",df[i].nunique())
```

```
Product : 3
Age : 32
Gender : 2
Education : 8
MaritalStatus : 2
Usage : 6
Fitness : 5
Income : 62
Miles : 37
```

In [29]:
```
#checking null values in every column of dataset
df.isnull().sum()
```

Out[29]:
```
Product          0
Age              0
Gender           0
Education        0
MaritalStatus    0
Usage            0
Fitness          0
Income           0
Miles            0
dtype: int64
```

In [372]:
```
#Perecntage of null values in every column of our data
df.isnull().sum()/len(df)*100
```

Out[372]:
```
Product          0.0
Age              0.0
Gender           0.0
Education        0.0
MaritalStatus    0.0
Usage            0.0
Fitness          0.0
Income           0.0
Miles            0.0
dtype: float64
```

No missing values in data

In [87]:
```
#Brief info about the dataset
df.describe()
```

Out[87]:

	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

Standard deviation of "Income" and "Miles" columns are high => more outliers

In [41]:
```
df.describe(include=object).T
```

Out[41]:

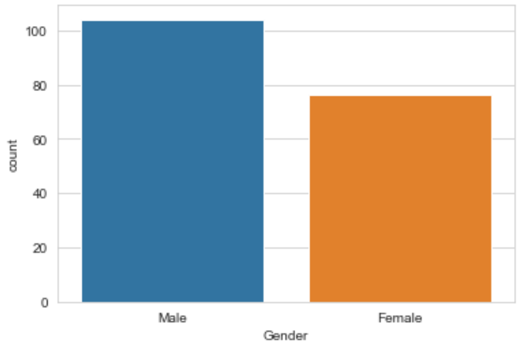
	count	unique	top	freq
Product	180	3	KP281	80
Gender	180	2	Male	104
MaritalStatus	180	2	Partnered	107

Univariate Analysis

In [42]:
```
#Gender-wise usage distribution of treadmill
df['Gender'].value_counts()
```

Out[42]:
```
Male      104
Female     76
Name: Gender, dtype: int64
```

In [45]:
```
sns.set_style(style='whitegrid')
sns.countplot(data=df,x='Gender')
plt.show()
```



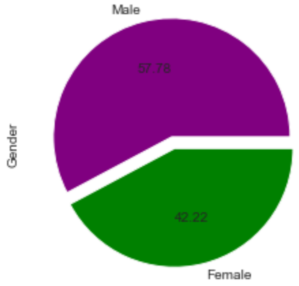
In [362]:
```
#Marginal Probability of Male and Female
df['Gender'].value_counts(normalize=True)*100
```

Out[362]:
```
Male      57.777778
Female    42.222222
Name: Gender, dtype: float64
```

Insight: Marginal Probability of Male and Female Customers

1. Probability of male customers using the product = 0.578
2. Probability of female customers using the product = 0.422

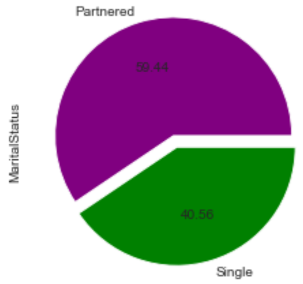
In [228]:
```
# Percentage of Male and Female using treadmill
df["Gender"].value_counts().plot.pie(explode=(0.05,0.05),colors=['purple','green'],autopct ="%.2f")
plt.show()
```



In [51]:
```
#Marital status wise usage of treadmill
df['MaritalStatus'].value_counts()
```

Out[51]:
```
Partnered    107
Single        73
Name: MaritalStatus, dtype: int64
```

In [229]:
```
# Percentage of Single and Partnered using treadmill
df["MaritalStatus"].value_counts().plot.pie(explode=(0.05,0.05),colors=['purple','green'],autopct ="%.2f")
plt.show()
```



In [53]:
```
df['MaritalStatus'].value_counts(normalize=True)*100
```

Out[53]:
```
Partnered    59.444444
Single       40.555556
Name: MaritalStatus, dtype: float64
```

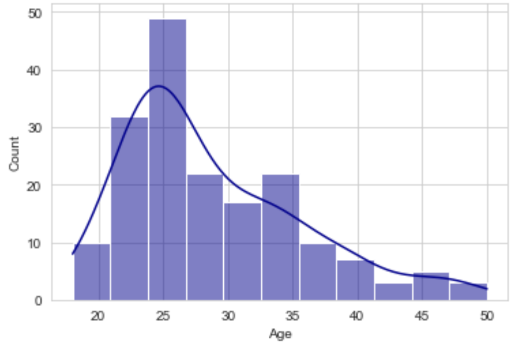
Insight: Marginal Probability of Male and Female Customers

1. Probability of male customers using the product = 0.578
2. Probability of female customers using the product = 0.422

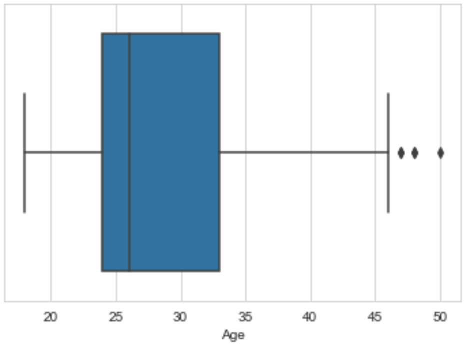
In [90]:
```
#Age-wise analysis of treadmill usage
df['Age'].describe()
```

Out[90]:
```
count    180.000000
mean      28.788889
std        6.943498
min       18.000000
25%       24.000000
50%       26.000000
75%       33.000000
max       50.000000
Name: Age, dtype: float64
```

In [230]:
```
sns.histplot(data=df,x='Age',kde=True,color ='darkblue')
plt.show()
```



In [91]:
```
sns.set_style(style='whitegrid')
sns.boxplot(data=df,x='Age')
plt.show()
```



In [260]:
```
# Determining age outliers
Q1=24
Q3=33
IQR = Q3-Q1
np.array(df[(df['Age']>(Q3+1.5*IQR)) | (df['Age']<(Q1-1.5*IQR))]['Age'])
```

Out[260]:
```
array([47, 50, 48, 47, 48], dtype=int64)
```

In [217]:
```
df['Age'].mode()
```

Out[217]:
```
0    25
dtype: int64
```

Insights :

1. Outliers in the 'Age' Column : array([47, 50, 48, 47, 48], dtype=int64)
2. Mean age = 28.78

3. Median age = 26
4. Modal age = 25
5. Minimum age = 18
6. Maximum age = 50
7. Standard deviation = 6.94

In [220]:
```
#Education Analysis
df['Education'].describe()
```

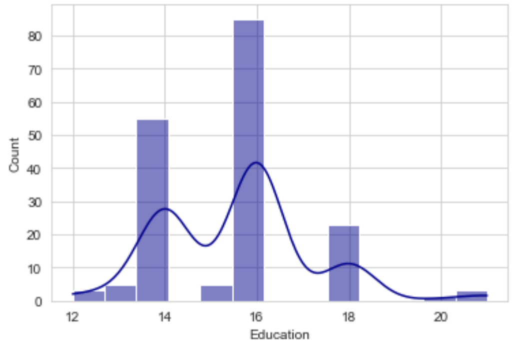
Out[220]:
```
count    180.000000
mean      15.572222
std        1.617055
min       12.000000
25%       14.000000
50%       16.000000
75%       16.000000
max       21.000000
Name: Education, dtype: float64
```

In [233]:
```
#Checking if values in education column are less than age or not to avoid any incorrect data values
df[df['Age']<df['Education']]
#Empty data set => 'Education' column does not have any incorrect values with respect to age
```

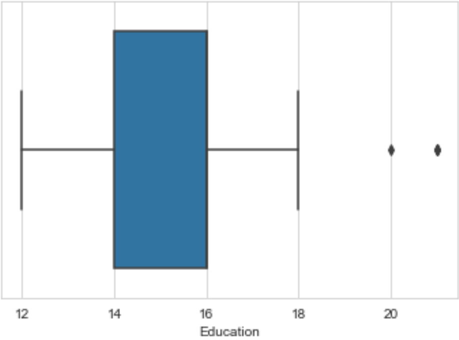
Out[233]:

Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles

In [226]:
```
sns.histplot(data=df,x='Education',kde=True,color ='darkblue')
plt.show()
```



In [223]:
```
sns.boxplot(data=df,x='Education')
plt.show()
```



In [261]:
```
# Determining outliers in "Education"
Q1_ed=14
Q3_ed=16
IQR_ed = Q3_ed-Q1_ed
np.array(df[(df['Education']>(Q3_ed+1.5*IQR_ed)) | (df['Education']<(Q1_ed-1.5*IQR_ed))]['Education'])
```

Out[261]:
```
array([20, 21, 21, 21], dtype=int64)
```

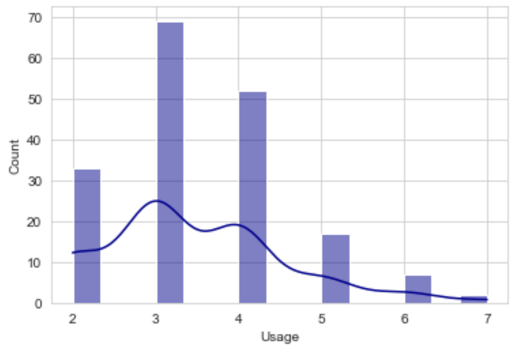
In [240]:
```
df['Education'].mode()
#Modal education age = 16 years
```

Out[240]:
```
0    16
dtype: int64
```

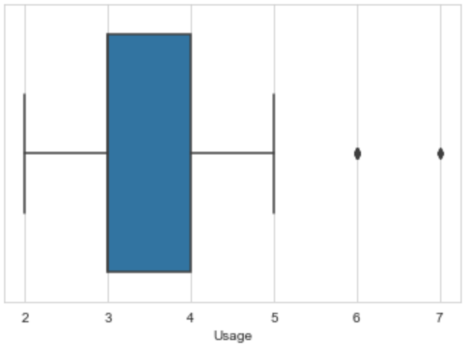
In [241]:
```
#Usage Analysis:
#Usage : The average number of times the customer plans to use the treadmill each week
df['Usage'].describe()
```

Out[241]:
```
count    180.000000
mean       3.455556
std        1.084797
min        2.000000
25%        3.000000
50%        3.000000
75%        4.000000
max        7.000000
Name: Usage, dtype: float64
```

In [243]:
```
sns.histplot(data=df,x='Usage',kde=True,color ='darkblue')
plt.show()
```



In [245]:
```
sns.boxplot(data=df,x='Usage')
plt.show()
```



In [262]:
```
# Determining outliers in "Usage"
Q1_usage=3
Q3_usage=4
IQR_usage = 1
np.array(df[(df['Usage']>(Q3_usage+1.5*IQR_usage)) | (df['Usage']<(Q1_usage-1.5*IQR_usage))]['Usage'])
```

Out[262]:
```
array([6, 6, 6, 7, 6, 7, 6, 6, 6], dtype=int64)
```

In [248]:
```
df['Usage'].mode()
#Modal usage per customer = 3 times per week
```

Out[248]:
```
0    3
dtype: int64
```

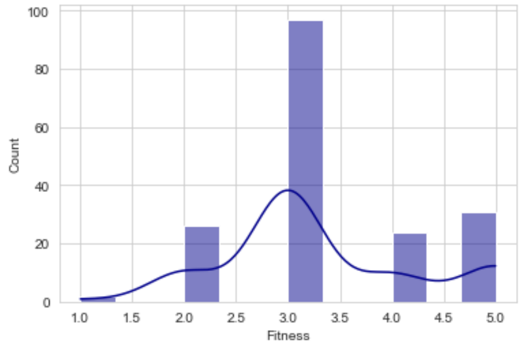
In [249]:
```
#Fitness analysis
#Fitness: Self-rated fitness on a 1-to-5 scale, where 1 is the poor shape and 5 is the excellent shape
df['Fitness'].describe()
```

Out[249]:
```
count    180.000000
mean       3.311111
std        0.958869
min        1.000000
25%        3.000000
50%        3.000000
75%        4.000000
max        5.000000
Name: Fitness, dtype: float64
```

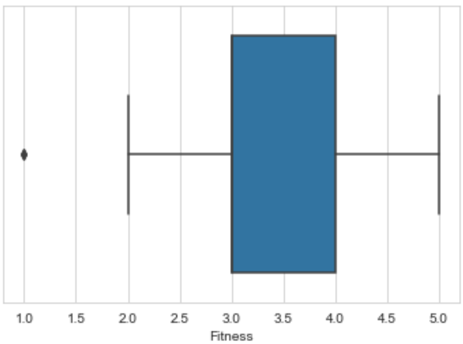
In [363]:
```
df['Fitness'].unique()
```

Out[363]:
```
array([4, 3, 2, 1, 5], dtype=int64)
```

In [256]:
```
sns.histplot(data=df,x='Fitness',kde=True,color ='darkblue')
plt.show()
```



In [257]:
```
sns.boxplot(data=df,x='Fitness')
plt.show()
```



In [263]:
```
# Determining outliers in "Fitness"
Q1_fitness=3
Q3_fitness=4
IQR_fitness = 1
np.array(df[(df['Fitness']>(Q3_fitness+1.5*IQR_fitness)) | (df['Fitness']<(Q1_fitness-1.5*IQR_fitness))]['Fitness'])
```

Out[263]:
```
array([1, 1], dtype=int64)
```

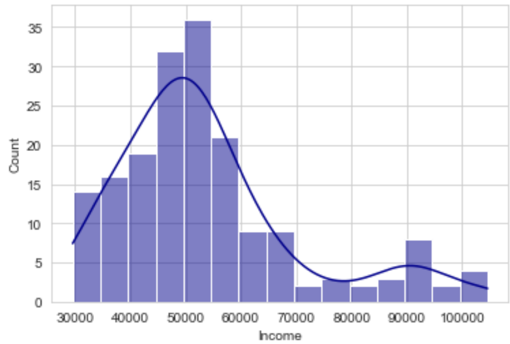
In [265]:
```
#Modal fitness rating
df['Fitness'].mode()
```

Out[265]:
```
0    3
dtype: int64
```

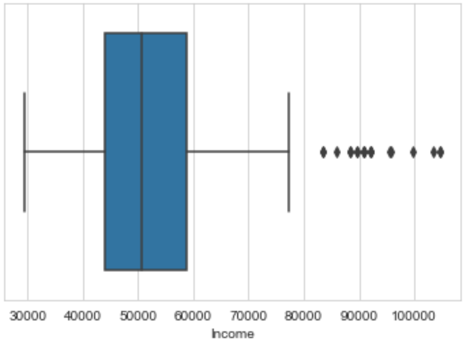
In [266]:
```
#Income Analysis: Annual Income
df['Income'].describe()
```

Out[266]:
```
count       180.000000
mean      53719.577778
std       16506.684226
min       29562.000000
25%       44058.750000
50%       50596.500000
75%       58668.000000
max      104581.000000
Name: Income, dtype: float64
```

In [267]:
```
sns.histplot(data=df,x='Income',kde=True,color ='darkblue')
plt.show()
```



In [269]:
```
sns.boxplot(data=df,x='Income')
plt.show()
```



In [280]:
```
# Determining outliers in "Income"
Q1_income=44058.750000
Q3_income=58668.000000
IQR_income = Q3_income-Q1_income
np.array(df[(df['Income']>(Q3_income+1.5*IQR_income)) | (df['Income']<(Q1_income-1.5*IQR_income))]['Income'])
```

Out[280]:
```
array([ 83416,  88396,  90886,  92131,  88396,  85906,  90886, 103336,
        99601,  89641,  95866,  92131,  92131, 104581,  83416,  89641,
        90886, 104581,  95508], dtype=int64)
```

In [281]:
```
#Number of Outliers in Income table
len(np.array(df[(df['Income']>(Q3_income+1.5*IQR_income)) | (df['Income']<(Q1_income-1.5*IQR_income))]['Income']))
```

Out[281]:
```
19
```

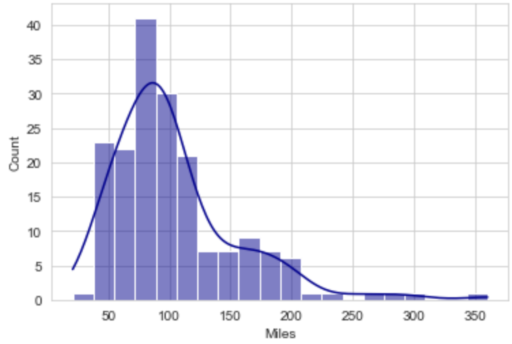
In [270]:
```
#Modal income = 45480
df['Income'].mode()
```

Out[270]:
```
0    45480
dtype: int64
```

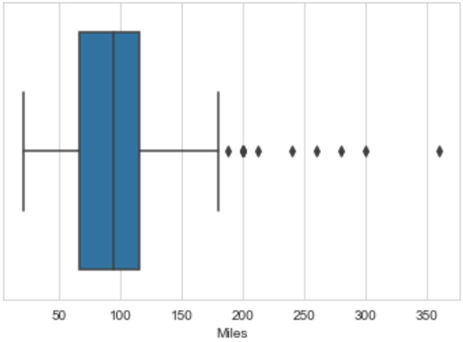
In [282]:
```
# Miles analysis
# Miles: The average number of miles the customer expects to walk/run each week
df['Miles'].describe()
```

Out[282]:
```
count    180.000000
mean     103.194444
std       51.863605
min       21.000000
25%       66.000000
50%       94.000000
75%      114.750000
max      360.000000
Name: Miles, dtype: float64
```

In [283]:
```
sns.histplot(data=df,x='Miles',kde=True,color ='darkblue')
plt.show()
```



In [284]:
```
sns.boxplot(data=df,x='Miles')
plt.show()
```



In [285]:
```
# Determining outliers in "Miles"
Q1_miles=66
Q3_miles=114.75
IQR_miles = Q3_miles-Q1_miles
np.array(df[(df['Miles']>(Q3_miles+1.5*IQR_miles)) | (df['Miles']<(Q1_miles-1.5*IQR_miles))]['Miles'])
```

Out[285]:
```
array([188, 212, 200, 200, 200, 240, 300, 280, 260, 200, 360, 200, 200],
      dtype=int64)
```

In [286]:
```
#Number of Outliers in Miles table
len(np.array(df[(df['Miles']>(Q3_miles+1.5*IQR_miles)) | (df['Miles']<(Q1_miles-1.5*IQR_miles))]['Miles']))
```

Out[286]:
```
13
```

In [288]:
```
#Modal number of miles customer expects to walk/run each week  = 85
df['Miles'].mode()
```

Out[288]:
```
0    85
dtype: int64
```

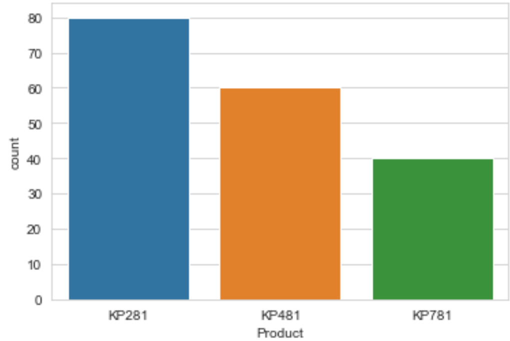
In [289]:
```
#Types of product analysis
df['Product'].unique()
#There are 3 types of products
```

Out[289]:
```
array(['KP281', 'KP481', 'KP781'], dtype=object)
```

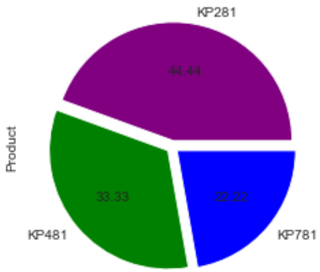
In [95]:
```
df['Product'].value_counts()
```

Out[95]:
```
KP281    80
KP481    60
KP781    40
Name: Product, dtype: int64
```

In [99]:
```
sns.set_style(style='whitegrid')
sns.countplot(data=df,x='Product')
plt.show()
```



In [291]:
```
#Marginal Probability of each type of product
df["Product"].value_counts().plot.pie(explode=(0.05,0.05,0.05),colors=['purple','green','blue'],autopct ="%.2f")
plt.show()
```



Bivariate Analysis

In [393]:
```
#Correlation between gender and different products
pd.crosstab(index=df["Gender"],columns=df["Product"],margins=True)
```

Out[393]:

Product Gender	KP281	KP481	KP781	All
Female	40	29	7	76
Male	40	31	33	104
All	80	60	40	180

In [396]:
```
#P(KP281 or KP481 or KP781 | Male or female)
pd.crosstab(index=df["Gender"],columns=df["Product"],margins=True,normalize='index')
```

Out[396]:

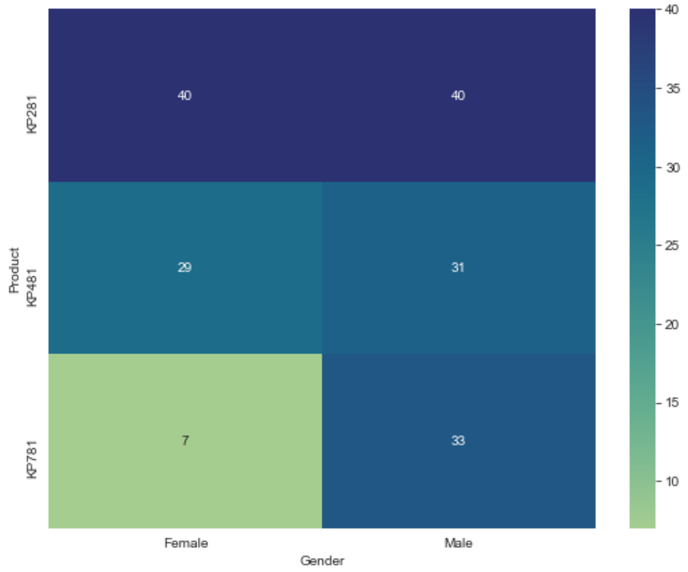
Product Gender	KP281	KP481	KP781
Female	0.526316	0.381579	0.092105
Male	0.384615	0.298077	0.317308
All	0.444444	0.333333	0.222222

In [397]:
```
#P(Male or female | KP281 or KP481 or KP781 )
pd.crosstab(index=df["Gender"],columns=df["Product"],margins=True,normalize='columns')
```

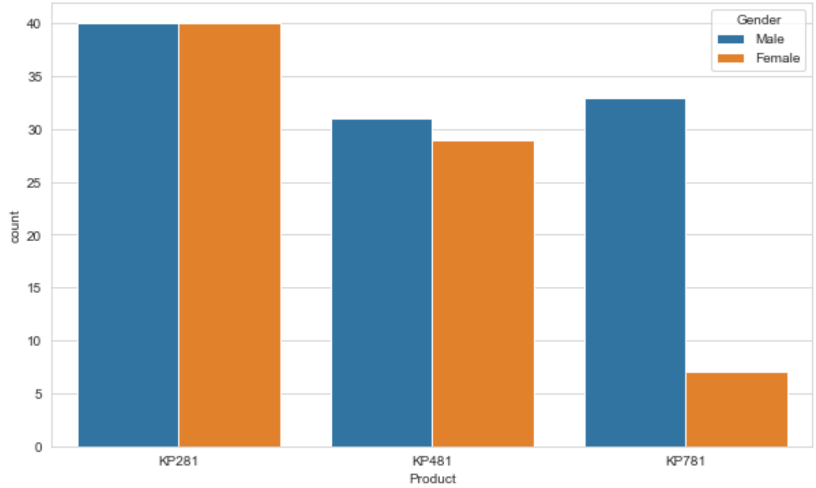
Out[397]:

Product Gender	KP281	KP481	KP781	All
Female	0.5	0.483333	0.175	0.422222
Male	0.5	0.516667	0.825	0.577778

In [147]:
```
#Heatmap for correlation between gender and different types of treadmills
plt.figure(figsize=(9,7))
sns.heatmap(pd.crosstab(df["Product"],df["Gender"]),cmap='crest',annot=True)
plt.show()
```



In [298]:
```
plt.figure(figsize=(10,6))
sns.countplot(data=df, x='Product', hue='Gender')
plt.show()
```



Insight :

1. The probability of female customers using 'KP781' is very low as compared to male customers
2. 52.6% of the female customers use KP281

In [379]:
```
#Correlation between marital status and different products
pd.crosstab(columns=df["Product"],index=df["MaritalStatus"],margins=True)
```

Out[379]:

Product	KP281	KP481	KP781	All
MaritalStatus				
Partnered	48	36	23	107
Single	32	24	17	73
All	80	60	40	180

In [384]:
```
#Conditional Probability: P(KP281 or KP481 or KP781 |Partnered or Single)
pd.crosstab(columns=df["Product"],index=df["MaritalStatus"],margins=True,normalize='index')
```

Out[384]:

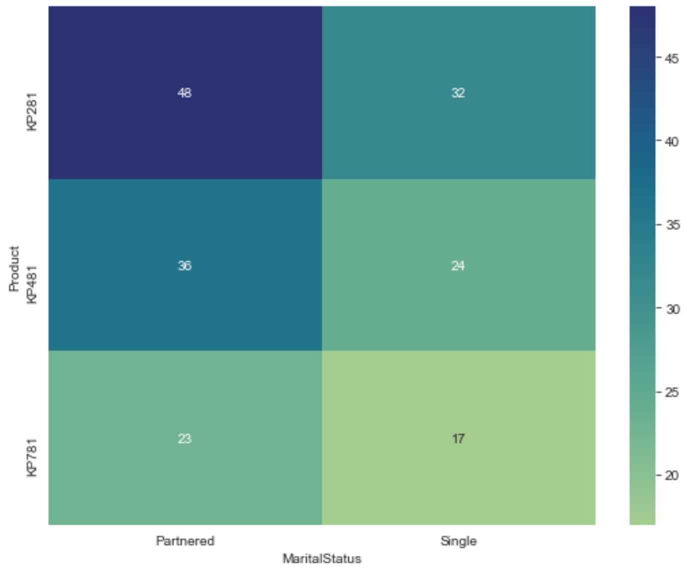
Product	KP281	KP481	KP781
MaritalStatus			
Partnered	0.448598	0.336449	0.214953
Single	0.438356	0.328767	0.232877
All	0.444444	0.333333	0.222222

In [398]:
```
#Conditional Probability: P(Partnered or Single|KP281 or KP481 or KP781 )
pd.crosstab(columns=df["Product"],index=df["MaritalStatus"],margins=True,normalize='columns')
```

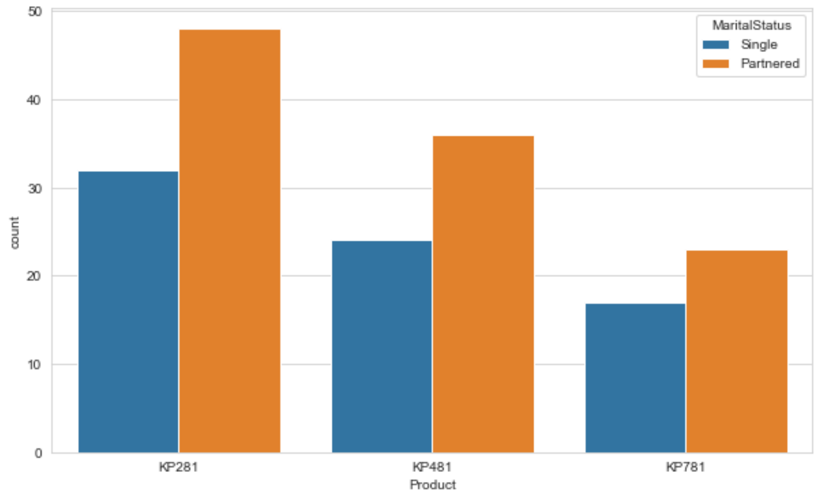
Out[398]:

Product	KP281	KP481	KP781	All
MaritalStatus				
Partnered	0.6	0.6	0.575	0.594444
Single	0.4	0.4	0.425	0.405556

In [149]:
```
#Heatmap for correlation between marital status and different types of treadmills
plt.figure(figsize=(9,7))
sns.heatmap(pd.crosstab(df["Product"],df["MaritalStatus"]),cmap='crest',annot=True)
plt.show()
```



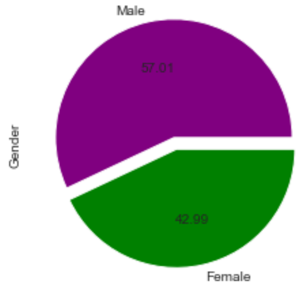
In [306]:
```
plt.figure(figsize=(10,6))
sns.countplot(data=df, x='Product', hue='MaritalStatus')
plt.show()
```



Insight :

1. Both partnered and single customers prefer KP281 over other treadmills
2. The probability of using KP281,KP481 & KP781 respectively is almost same among partnered and single customers

In [312]:
```
#Conditional Probability of male and female customers using treadmill under "Partnered" status
df[df["MaritalStatus"]=="Partnered"]["Gender"].value_counts().plot.pie(explode=(0.05,0.05),colors=['purple','green'],autopct ="%.2f")
plt.show()
```



Effect of remaining parameters on product purchased

In [402]:
```
#Conditional Probability: P(KP281 or KP481 or KP781 |Age)
pd.crosstab(columns=df["Product"],index=df["Age"],margins=True,normalize='index')
```

Out[402]:

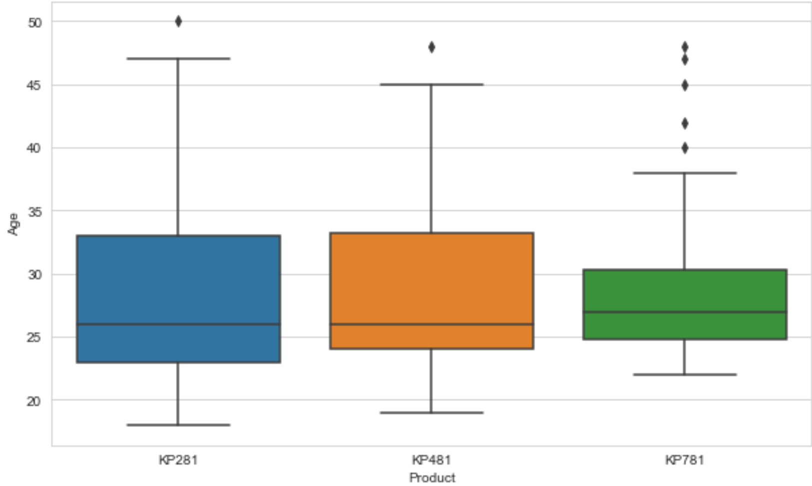
Product Age	KP281	KP481	KP781
18	1.000000	0.000000	0.000000
19	0.750000	0.250000	0.000000
20	0.400000	0.600000	0.000000
21	0.571429	0.428571	0.000000
22	0.571429	0.000000	0.428571
23	0.444444	0.388889	0.166667
24	0.416667	0.250000	0.333333
25	0.280000	0.440000	0.280000
26	0.583333	0.250000	0.166667
27	0.428571	0.142857	0.428571
28	0.666667	0.000000	0.333333
29	0.500000	0.166667	0.333333
30	0.285714	0.285714	0.428571
31	0.333333	0.500000	0.166667
32	0.500000	0.500000	0.000000
33	0.250000	0.625000	0.125000
34	0.333333	0.500000	0.166667
35	0.375000	0.500000	0.125000
36	1.000000	0.000000	0.000000
37	0.500000	0.500000	0.000000
38	0.571429	0.285714	0.142857
39	1.000000	0.000000	0.000000
40	0.200000	0.600000	0.200000
41	1.000000	0.000000	0.000000
42	0.000000	0.000000	1.000000
43	1.000000	0.000000	0.000000
44	1.000000	0.000000	0.000000
45	0.000000	0.500000	0.500000
46	1.000000	0.000000	0.000000
47	0.500000	0.000000	0.500000
48	0.000000	0.500000	0.500000
50	1.000000	0.000000	0.000000
All	0.444444	0.333333	0.222222

In [400]:
```
#Conditional Probability: P(Age | KP281 or KP481 or KP781 )
pd.crosstab(columns=df["Product"],index=df["Age"],margins=True,normalize='columns')
```

Out[400]:

Product	KP281	KP481	KP781	All
Age				
18	0.0125	0.000000	0.000	0.005556
19	0.0375	0.016667	0.000	0.022222
20	0.0250	0.050000	0.000	0.027778
21	0.0500	0.050000	0.000	0.038889
22	0.0500	0.000000	0.075	0.038889
23	0.1000	0.116667	0.075	0.100000
24	0.0625	0.050000	0.100	0.066667
25	0.0875	0.183333	0.175	0.138889
26	0.0875	0.050000	0.050	0.066667
27	0.0375	0.016667	0.075	0.038889
28	0.0750	0.000000	0.075	0.050000
29	0.0375	0.016667	0.050	0.033333
30	0.0250	0.033333	0.075	0.038889
31	0.0250	0.050000	0.025	0.033333
32	0.0250	0.033333	0.000	0.022222
33	0.0250	0.083333	0.025	0.044444
34	0.0250	0.050000	0.025	0.033333
35	0.0375	0.066667	0.025	0.044444
36	0.0125	0.000000	0.000	0.005556
37	0.0125	0.016667	0.000	0.011111
38	0.0500	0.033333	0.025	0.038889
39	0.0125	0.000000	0.000	0.005556
40	0.0125	0.050000	0.025	0.027778
41	0.0125	0.000000	0.000	0.005556
42	0.0000	0.000000	0.025	0.005556
43	0.0125	0.000000	0.000	0.005556
44	0.0125	0.000000	0.000	0.005556
45	0.0000	0.016667	0.025	0.011111
46	0.0125	0.000000	0.000	0.005556
47	0.0125	0.000000	0.025	0.011111
48	0.0000	0.016667	0.025	0.011111
50	0.0125	0.000000	0.000	0.005556

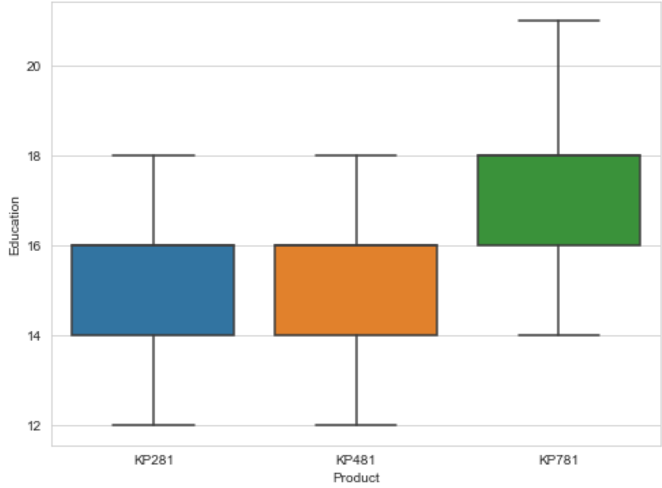
In [351]:
```
#Correlation between Age and Product
plt.figure(figsize=(10,6))
sns.boxplot(data=df, x='Product', y='Age')
plt.show()
```



Insights:

1. Median age of customers purchasing KP281 and KP481 is almost same
2. Customers of age range 25-30 prefer KP781

In [320]:
```
#Correlation between Education and Product
plt.figure(figsize=(8,6))
sns.boxplot(data=df, x='Product', y='Education')
plt.show()
```



Insights:

1. Median education of customers purchasing KP281 and KP481 is exactly same : 14-16 years
2. Customers with education > 16 years prefer KP781

In [405]:
```
#Correlation between Usage and Product
#Conditional Probability: P(KP281 or KP481 or KP781 |Usage)
pd.crosstab(columns=df["Product"],index=df["Usage"],margins=True,normalize='index')
```

Out[405]:

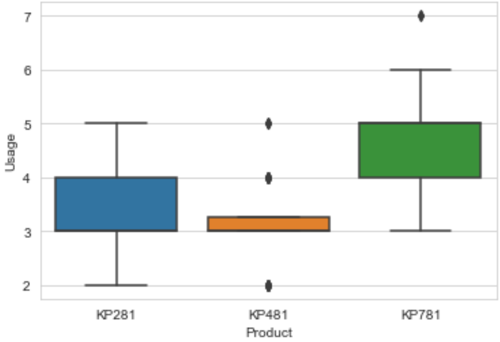
Product	KP281	KP481	KP781
Usage			
2	0.575758	0.424242	0.000000
3	0.536232	0.449275	0.014493
4	0.423077	0.230769	0.346154
5	0.117647	0.176471	0.705882
6	0.000000	0.000000	1.000000
7	0.000000	0.000000	1.000000
All	0.444444	0.333333	0.222222

In [404]:
```
#Conditional Probability: P(Usage | KP281 or KP481 or KP781)
pd.crosstab(columns=df["Product"],index=df["Usage"],margins=True,normalize='columns')
```

Out[404]:

Product	KP281	KP481	KP781	All
Usage				
2	0.2375	0.233333	0.000	0.183333
3	0.4625	0.516667	0.025	0.383333
4	0.2750	0.200000	0.450	0.288889
5	0.0250	0.050000	0.300	0.094444
6	0.0000	0.000000	0.175	0.038889
7	0.0000	0.000000	0.050	0.011111

In [322]:
```
plt.figure(figsize=(6,4))
sns.boxplot(data=df, x='Product', y='Usage')
plt.show()
```



Insights:

1. Customers using KP781 plan to use the treadmill for higher number of times(>4) as compared to KP281 & KP481
2. Customers using KP481 plan to use generally 3 times per week

In [406]:
```
#Correlation between Fitness and Product
#Conditional Probability: P(KP281 or KP481 or KP781 |Fitness)
pd.crosstab(columns=df["Product"],index=df["Fitness"],margins=True,normalize='index')
```

Out[406]:

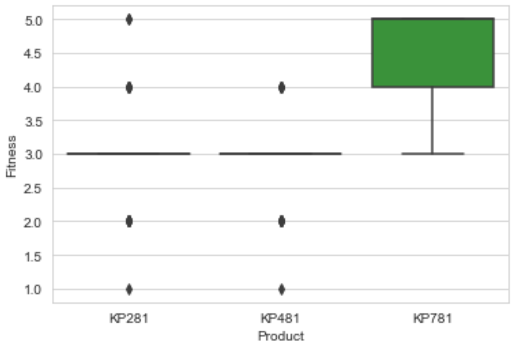
Product Fitness	KP281	KP481	KP781
1	0.500000	0.500000	0.000000
2	0.538462	0.461538	0.000000
3	0.556701	0.402062	0.041237
4	0.375000	0.333333	0.291667
5	0.064516	0.000000	0.935484
All	0.444444	0.333333	0.222222

In [407]:
```
#Conditional Probability: P(Fitness | KP281 or KP481 or KP781 )
pd.crosstab(columns=df["Product"],index=df["Fitness"],margins=True,normalize='columns')
```

Out[407]:

Product Fitness	KP281	KP481	KP781	All
1	0.0125	0.016667	0.000	0.011111
2	0.1750	0.200000	0.000	0.144444
3	0.6750	0.650000	0.100	0.538889
4	0.1125	0.133333	0.175	0.133333
5	0.0250	0.000000	0.725	0.172222

In [358]:
```
plt.figure(figsize=(6,4))
sns.boxplot(data=df, x='Product', y='Fitness')
plt.show()
```



Insight : Self rating of Fitness is high among customers using KP781(>=4 generally) which indicates that people using KP781 have higher fitness levels

In [408]:
```
#Correlation between Income and Product
#Binning income into three categories for analysis
bins=[-1.0,60000.0,90000.0,200000.0]
labels = ["lower","middle","higher"]
df["Income_category"]=pd.cut(df['Income'],labels=labels,bins=bins)
df.head()
```

Out[408]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles	Income_category
0	KP281	18	Male	14	Single	3	4	29562	112	lower
1	KP281	19	Male	15	Single	2	3	31836	75	lower
2	KP281	19	Female	14	Partnered	4	3	30699	66	lower
3	KP281	19	Male	12	Single	3	3	32973	85	lower
4	KP281	20	Male	13	Partnered	4	2	35247	47	lower

In [410]:
```
#Conditional Probability: P(KP281 or KP481 or KP781 |Income_category)
pd.crosstab(columns=df["Product"],index=df["Income_category"],margins=True,normalize='index')
```

Out[410]:

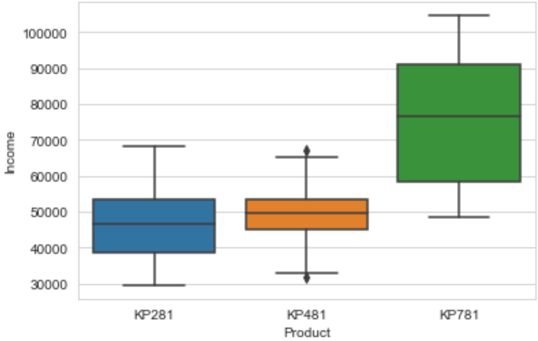
Product	KP281	KP481	KP781
Income_category			
lower	0.536232	0.384058	0.079710
middle	0.200000	0.233333	0.566667
higher	0.000000	0.000000	1.000000
All	0.444444	0.333333	0.222222

In [411]:
```
#Conditional Probability: P(Income_category | KP281 or KP481 or KP781 )
pd.crosstab(columns=df["Product"],index=df["Income_category"],margins=True,normalize='columns')
```

Out[411]:

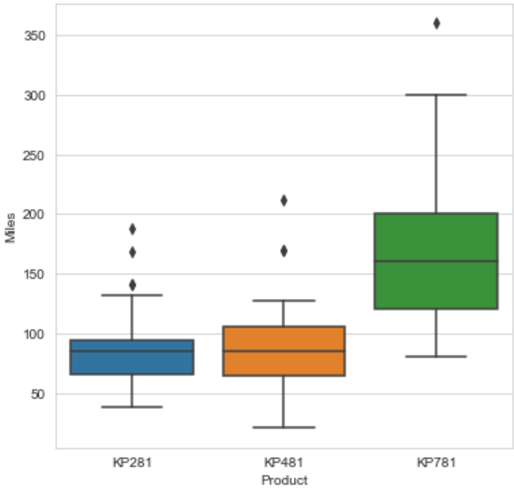
Product	KP281	KP481	KP781	All
Income_category				
lower	0.925	0.883333	0.275	0.766667
middle	0.075	0.116667	0.425	0.166667
higher	0.000	0.000000	0.300	0.066667

In [331]:
```
plt.figure(figsize=(6,4))
sns.boxplot(data=df, x='Product', y='Income')
plt.show()
```



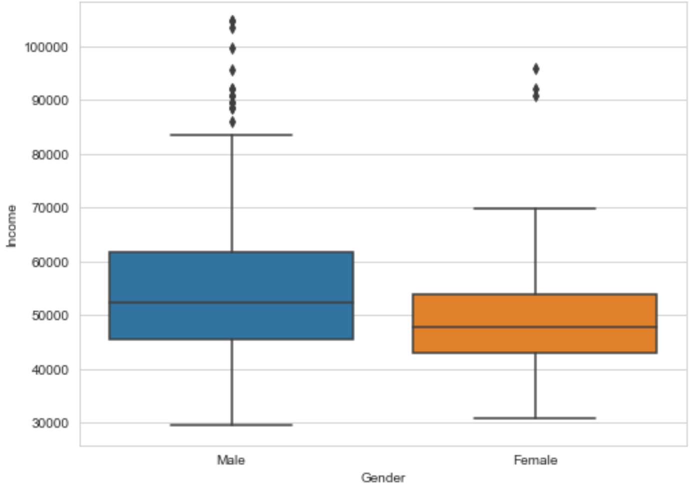
People with Income_category = high prefer 'KP781'

In [334]:
```
#Correlation between Miles and Product
plt.figure(figsize=(6,6))
sns.boxplot(data=df, x='Product', y='Miles')
plt.show()
```



Customers using KP781 plan to run more than 120 miles which is higher as compared to other products

In [422]:
```
#Correlation between income and gender
plt.figure(figsize=(8,6))
sns.boxplot(data=df, x='Gender', y='Income')
plt.show()
```



Male have higher median income than female

In [416]:
```
#Pairplots
sns.pairplot(df)
plt.show()
```

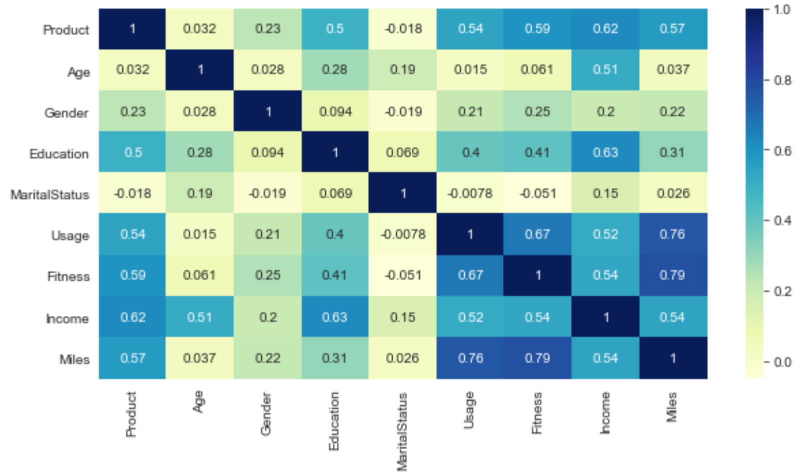

Age
Education
Usage
Fitness
Income
Miles

In [412]:
```
#Making all columns numerical for correlation computation
# Creating a copy of the dataframe
df_copy = df.drop(["Income_category"], axis = 1).copy()
df_copy['Gender'].replace(['Male', 'Female'], [1, 0], inplace=True)
df_copy['MaritalStatus'].replace(['Single', 'Partnered'], [0, 1], inplace=True)
df_copy['Product'].replace(['KP281', 'KP481', 'KP781'], [0, 1, 2], inplace=True)
df_copy.corr()
```

Out[412]:

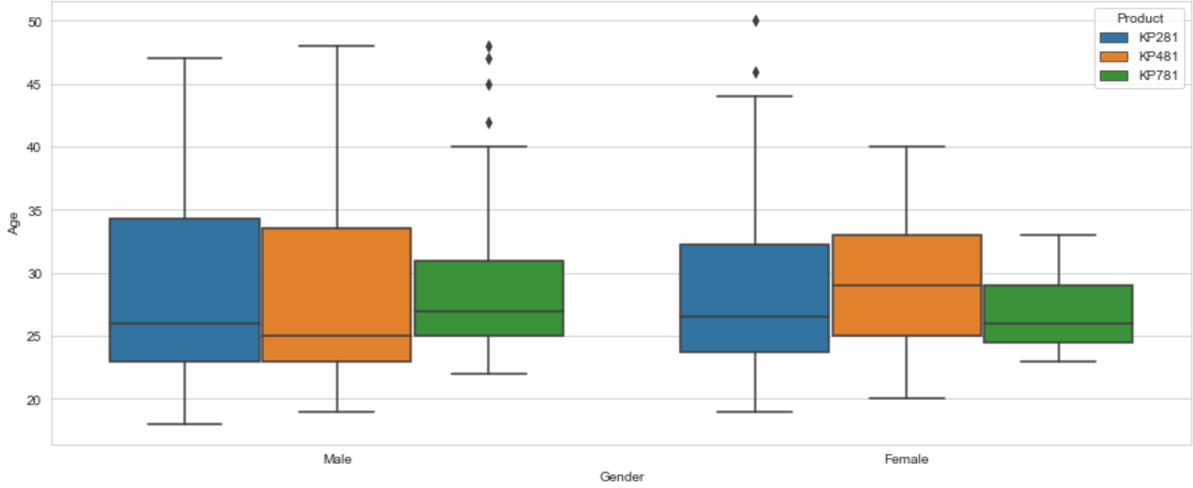
	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
Product	1.000000	0.032225	0.230653	0.495018	-0.017602	0.537447	0.594883	0.624168	0.571596
Age	0.032225	1.000000	0.027544	0.280496	0.192152	0.015064	0.061105	0.513414	0.036618
Gender	0.230653	0.027544	1.000000	0.094089	-0.018836	0.214424	0.254609	0.202053	0.217869
Education	0.495018	0.280496	0.094089	1.000000	0.068569	0.395155	0.410581	0.625827	0.307284
MaritalStatus	-0.017602	0.192152	-0.018836	0.068569	1.000000	-0.007786	-0.050751	0.150293	0.025639
Usage	0.537447	0.015064	0.214424	0.395155	-0.007786	1.000000	0.668606	0.519537	0.759130
Fitness	0.594883	0.061105	0.254609	0.410581	-0.050751	0.668606	1.000000	0.535005	0.785702
Income	0.624168	0.513414	0.202053	0.625827	0.150293	0.519537	0.535005	1.000000	0.543473
Miles	0.571596	0.036618	0.217869	0.307284	0.025639	0.759130	0.785702	0.543473	1.000000

In [414]:
```
# Correlation Plot above as a Heatmap -
plt.figure(figsize=(10,5))
sns.heatmap(df_copy.corr(), cmap="YlGnBu", annot=True)
plt.show()
```

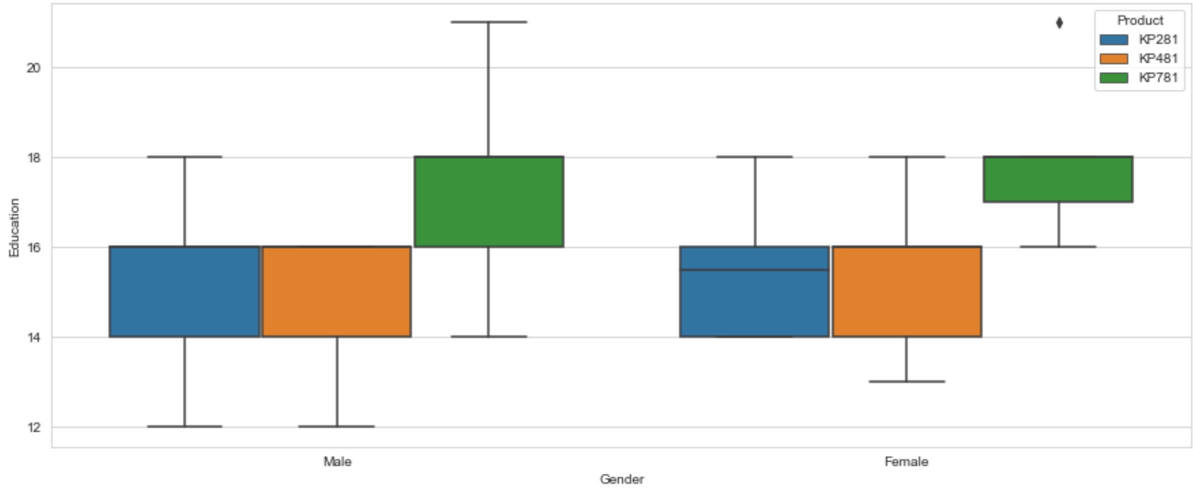


Multivariate Analysis

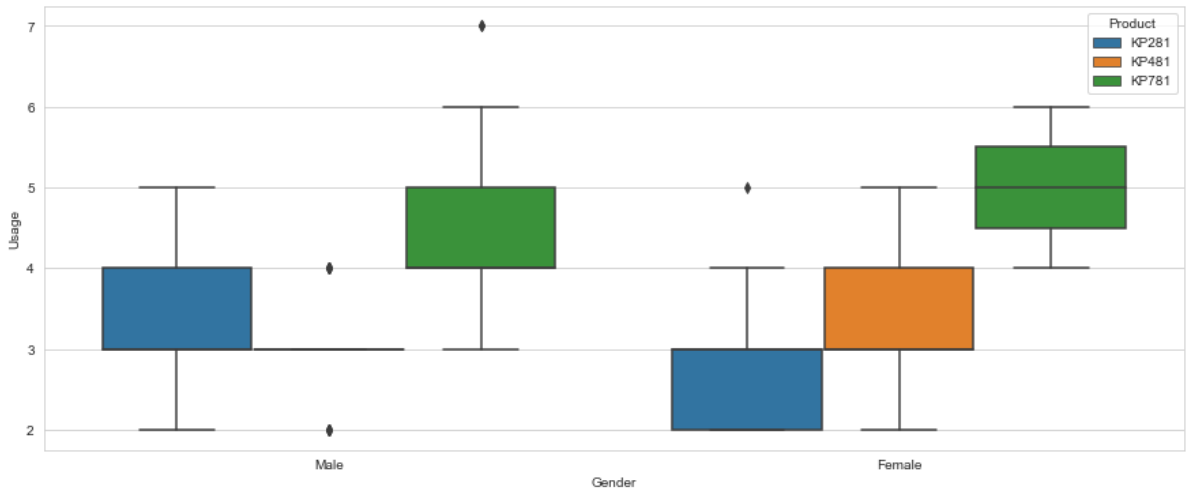
In [344]:
```
plt.figure(figsize=(15,6))
sns.boxplot(data=df,x='Gender',y='Age',hue='Product')
plt.show()
```



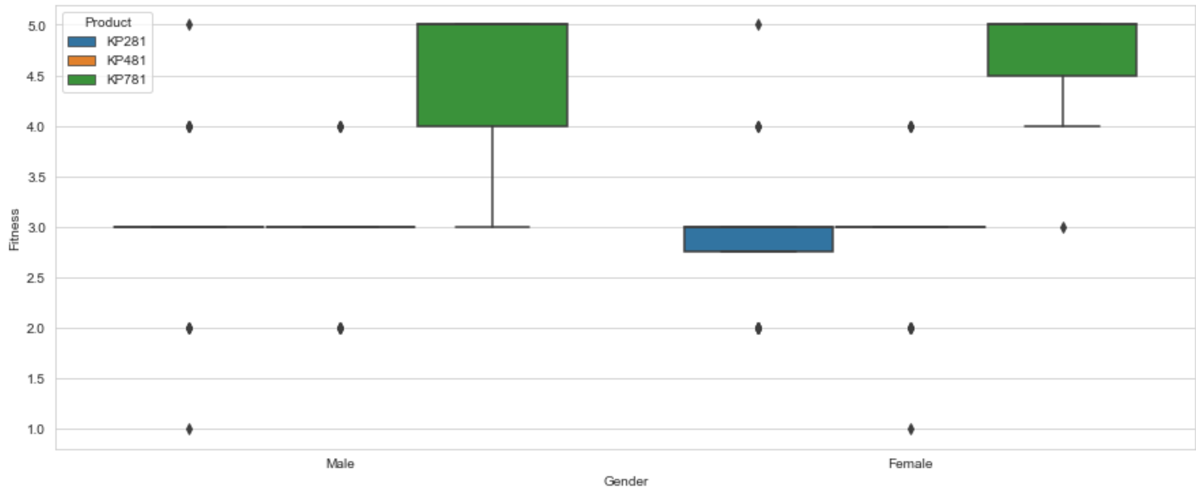
In [349]:
```
plt.figure(figsize=(15,6))
sns.boxplot(data=df,x='Gender',y='Education',hue='Product')
plt.show()
```



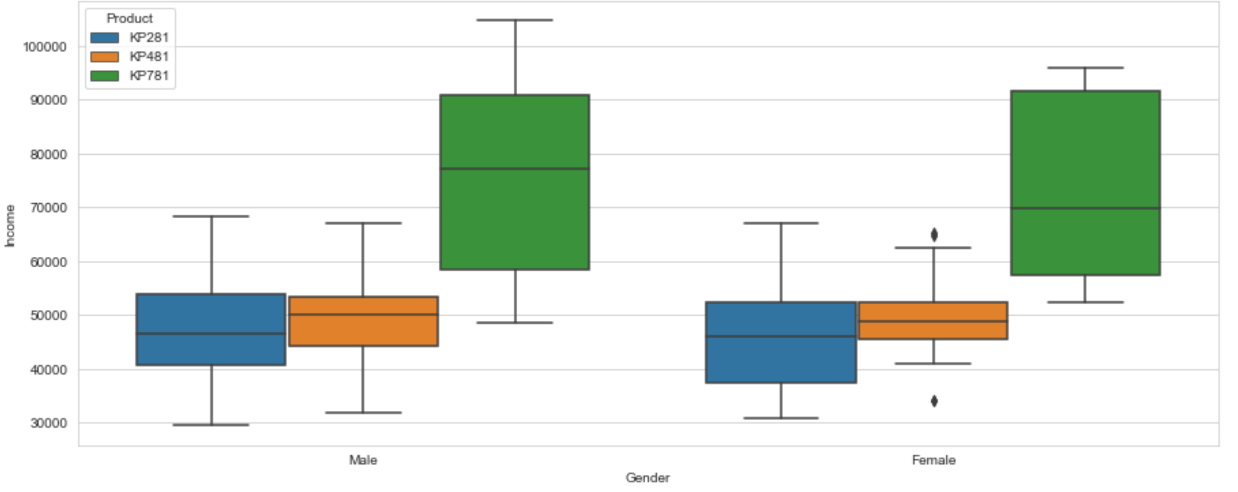
In [348]:
```
plt.figure(figsize=(15,6))
sns.boxplot(data=df,x='Gender',y='Usage',hue='Product')
plt.show()
```



In [347]:
```
plt.figure(figsize=(15,6))
sns.boxplot(data=df,x='Gender',y='Fitness',hue='Product')
plt.show()
```

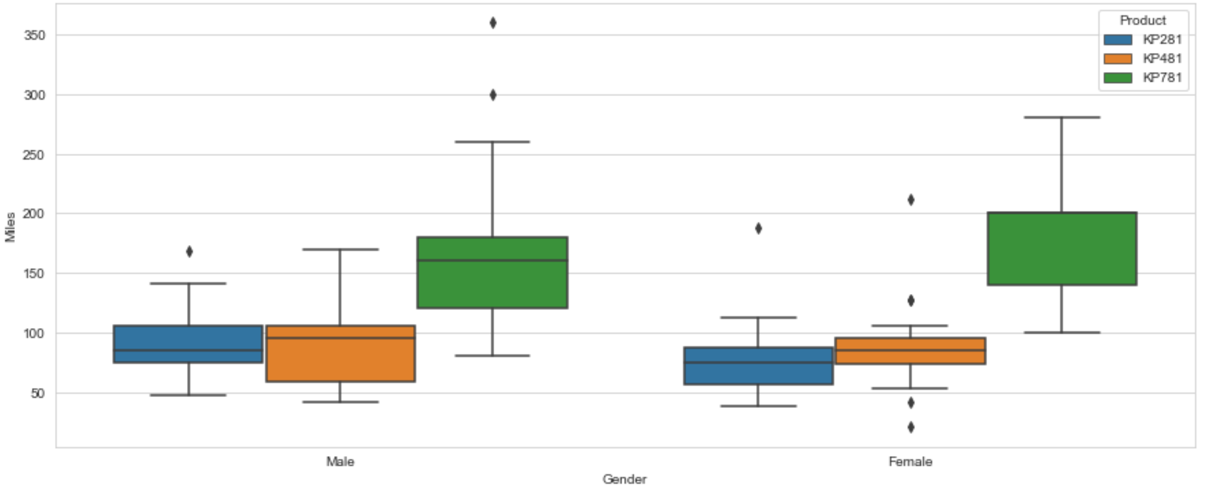


In [346]:
```
plt.figure(figsize=(15,6))
sns.boxplot(data=df,x='Gender',y='Income',hue='Product')
plt.show()
```



Insight: Male and Female Customers with higher income range prefer KP781

In [345]:
```
plt.figure(figsize=(15,6))
sns.boxplot(data=df,x='Gender',y='Miles',hue='Product')
plt.show()
```



Customer Profiling

In [417]:
```
#Case 1:
c1 = df.MaritalStatus == "Partnered"
c2 = df.Gender == "Female"
c3 = df.Education > 10
c4 = df.Fitness == 5
df[c1 & c2 & c3 & c4]
```

Out[417]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles	Income_category
23	KP281	24	Female	16	Partnered	5	5	44343	188	lower
152	KP781	25	Female	18	Partnered	5	5	61006	200	middle
162	KP781	28	Female	18	Partnered	6	5	92131	180	higher
167	KP781	30	Female	16	Partnered	6	5	90886	280	higher
171	KP781	33	Female	18	Partnered	4	5	95866	200	higher

In [419]:
```
df[c1 & c2 & c3 & c4]["Product"].value_counts(normalize = True)
```

Out[419]:
```
KP781    0.8
KP281    0.2
Name: Product, dtype: float64
```

In [420]:
```
#Case 2:
c1 = df.MaritalStatus == "Partnered"
c2 = df.Gender == "Female"
df[c1 & c2]["Product"].value_counts(normalize = True)
```

Out[420]:
```
KP281    0.586957
KP481    0.326087
KP781    0.086957
Name: Product, dtype: float64
```

Recommendations:

1. Increase the features of KP481 and increase the price little bit
2. Make KP481 as "decoy"
3. Give discounts for female customers on KP781(visible in multivariate analysis) because the probability of female customers using 'KP781' is very low as compared to male customers
4. People with higher income prefer KP781
5. Self rating of Fitness is high among customers using KP781(>=4 generally) which indicates that people using KP781 have higher fitness levels
6. Female customers under Partnered status with education>10 and Fitness=5, prefer KP781(80%)
7. Married females generally prefer KP281(58%)
8. Females planning to use treadmill 3-4 times a week, are more likely to buy KP481 product