Business Problem:

AEROFIT is a leading brand in the field of Fitness Equipments. It provides a product range including treadmills. Their Market Research team wants to identify 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.

Product Portfolio:

1. The KP281 is an entry-level treadmill that sells for $1,500.
2. The KP481 is for mid-level runners that sell for $1,750.
3. The KP781 treadmill is having advanced features that sell for $2,500.

Problem Statement:

1. To perform descriptive analytics and create a customer profile for each **AEROFIT** treadmill model
2. To construct two-way contingency tables and compute all conditional and marginal probabilities along with Insights and Recommendations (For each treadmill model)

https://stackoverflow.com/questions/10059594/a-simple-explanation-of-naive-bayes-classification

In [1]:

```
# Importing required packages for analysis
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
# Initial pandas & matplotlib setup
pd.options.display.max_rows = 20
pd.options.display.max_columns = 20
np.set_printoptions(precision=4,suppress=True)
sns.set_palette("muted")
```

In [3]:

```
import matplotlib_inline
matplotlib_inline.backend_inline.set_matplotlib_formats('svg')
```

In [4]:

```
# Importing the given dataset to pandas dataframe
df = pd.read_csv("./aerofit_treadmill.txt")
df.head(5)
```

Out[4]:

	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

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
4	KP281	20	Male	13	Partnered	4	2	35247	47

Insights:

1. As shown, dataset has data related to customers who bought treadmills from AeroFit in the past three months
2. For every customer, dataset has datapoints on Age, Gender, Education, Marital Status, Usage, Fitness, Income, Miles and Product purchased

Basic Metrics:

In [5]:
```
# To get the shape of the dataset
print(f"Number of Customers: {df.shape[0]}")
print(f"Total Features:  {df.shape[1]}")
```

```
Number of Customers: 180
Total Features:  9
```

In [6]:
```
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
```

In [7]:
```
# No duplicate records in the dataset
df.loc[df.duplicated() == True,:].sum().sum()
```

Out[7]:
```
0.0
```

Insights:

1. Looking at the data, there are no null values and no duplicates in any columns. There are 6 numerical columns and 3 categorical columns
2. Datatypes match the data.
   - All numerical columns like Age, Education, Usage, Income, Miles have int64 datatype
   - All categorical columns like Product, Gender, MaritalStatus have object datatype
3. Based on the further findings, more categorial columns could be created from numerical columns

In [8]:
```
# Statistical Summary of numerical columns
df.describe().round(2)
```

Out[8]:

	Age	Education	Usage	Fitness	Income	Miles
count	180.00	180.00	180.00	180.00	180.00	180.00
mean	28.79	15.57	3.46	3.31	53719.58	103.19
std	6.94	1.62	1.08	0.96	16506.68	51.86
min	18.00	12.00	2.00	1.00	29562.00	21.00
25%	24.00	14.00	3.00	3.00	44058.75	66.00
50%	26.00	16.00	3.00	3.00	50596.50	94.00
75%	33.00	16.00	4.00	4.00	58668.00	114.75
max	50.00	21.00	7.00	5.00	104581.00	360.00

Insights:

1. Age range of Customers 18 - 50 Years
2. Years of education varies from 12 to 21 Years
3. Customers plan to use the treadmill 2 to 7 times a week
4. Fitness levels of 25% of customers are under 3 on a scale of 1-5
5. 50% of customers expect to walk/run more than 94 miles a week !!
6. 75% of the Customers have an Yearly income of $60,000 or less
7. For all the Features, mean and median are almost very close. Age & Miles features are an exception

Non Graphical Analysis:

In [9]:

```
# Get the datatype & column name as a list of lists
columns_datatypes = df.dtypes.reset_index().to_dict("tight")["data"]
categorical_cols =[]
numerical_cols = []
# Segregate columns into two buckets based on the datatype
for column,datatype in columns_datatypes:
    if datatype == "int64":
        numerical_cols.append(column)
    else:
        categorical_cols.append(column)

print(f"Numerical Columns: {numerical_cols}; Count: {len(numerical_cols)}")
print(f"Categorical Columns: {categorical_cols}; Count: {len(categorical_cols)}")
```

```
Numerical Columns: ['Age', 'Education', 'Usage', 'Fitness', 'Income', 'Miles']; Count: 6
Categorical Columns: ['Product', 'Gender', 'MaritalStatus']; Count: 3
```

Unique Attributes:

In [10]:

```
# Get unique values & unique value counts of categorical variables
for column in categorical_cols:
    print(f"{column} :\n Unique Values: {df[column].unique()},\n Unique Value Counts: {df[column].nunique()}")
    print("--------------XXX-------------")
    print()
```

```python
# Get the unique values and counts for Fitness & Usage Columns as well
for column in ["Fitness","Usage"]:
    print(f"{column} :\n Unique Values: {df[column].unique()},\n Unique Value Counts: {df[column].nunique()}")
    print("--------------XXX-------------")
    print()
```

```
Product :
 Unique Values: ['KP281' 'KP481' 'KP781'],
 Unique Value Counts: 3
--------------XXX-------------

Gender :
 Unique Values: ['Male' 'Female'],
 Unique Value Counts: 2
--------------XXX-------------

MaritalStatus :
 Unique Values: ['Single' 'Partnered'],
 Unique Value Counts: 2
--------------XXX-------------

Fitness :
 Unique Values: [4 3 2 1 5],
 Unique Value Counts: 5
--------------XXX-------------

Usage :
 Unique Values: [3 2 4 5 6 7],
 Unique Value Counts: 6
--------------XXX-------------
```

Insights:

1. As expected, Product Column has three different types of treadmills
2. In Gender column, there are two unique values - Male, Female
3. In MaritalStatus column, there are two unique values - Single, Partnered
4. All Customers expect to use the treadmill from 2 to 7 times a week
5. Fitness levels range from 1 to 5

Value Counts:

In [11]:
```python
# Get the value counts for the categorical colums
print()
for column in categorical_cols:
    print(df.loc[:,column].value_counts())
    print()
    print("--------------XXX-------------")
    print()

# Get the value counts for Fitness & Usage columns
for column in ["Fitness","Usage"]:
    print(df.loc[:,column].value_counts())
    print()
```

```
    print("--------------XXX-------------")
```

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

--------------XXX-------------

Male      104
Female     76
Name: Gender, dtype: int64

--------------XXX-------------

Partnered    107
Single        73
Name: MaritalStatus, dtype: int64

--------------XXX-------------

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

--------------XXX-------------

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

--------------XXX-------------
```

Insights:

In the past three months:

1. Men seem to have bought more treadmills than Women
2. Majority of customers that bought treadmill are married
3. Customers who bought entry level treadmill are more when compared to the other two models
4. Almost 100 of 180 Customers feel that their Fitness level is 3 (average)
5. More than 100 Customers expect to use the treadmill atleast 3 to 4 times a week

In [12]:

```
entryLevel = df.loc[(df["Product"] == "KP281") == True ,:].copy()
print(f"Total Customers that bought KP281 model: {entryLevel.shape[0]}")
```

```
Total Customers that bought KP281 model: 80
```

In [13]:
```
midLevel = df.loc[(df["Product"] == "KP481") == True ,:].copy()
print(f"Total Customers that bought KP481 model: {midLevel.shape[0]}")
```

```
Total Customers that bought KP481 model: 60
```

In [14]:
```
advanceLevel = df.loc[(df["Product"] == "KP781") == True ,:].copy()
print(f"Total Customers that bought KP781 model: {advanceLevel.shape[0]}")
```

```
Total Customers that bought KP781 model: 40
```

Insights:

1. 45 % of Customers bought entry level model KP281
2. ~ 33 % bought mid Level model KP481
3. ~ 22 % bought Advance model KP781
4. Considering the prices of the three models, distribtuion of Customers makes sense. However, there could be more reasons.

In [15]:
```
# Get the value counts for the categorical colums
print("Entry Level Model KP281:")
print()
for column in categorical_cols:
    print(entryLevel.loc[:,column].value_counts(normalize=True).round(2)*100)
    print()
    print("--------------XXX-------------")
    print()

# Get the value counts for Fitness & Usage columns
for column in ["Fitness","Usage"]:
    print(entryLevel.loc[:,column].value_counts(normalize=True).round(2)*100)
    print()
    print("--------------XXX-------------")
    print()
```

```
Entry Level Model KP281:

KP281    100.0
Name: Product, dtype: float64

--------------XXX-------------

Male      50.0
Female    50.0
Name: Gender, dtype: float64

--------------XXX-------------

Partnered    60.0
Single       40.0
Name: MaritalStatus, dtype: float64

--------------XXX-------------

3    68.0
2    18.0
4    11.0
5     2.0
```

```
1     1.0
Name: Fitness, dtype: float64

--------------XXX-------------

3    46.0
4    28.0
2    24.0
5     2.0
Name: Usage, dtype: float64

--------------XXX-------------
```

Insights:

Regarding Customers that bought KP281 Model:

1. Equal Proportion of Men & Women
2. Partnered Customers proportion is more than Singles
3. ~ 70 percent have Fitness Level of 3, followed by ~11 percent at Fitness Level 2
4. About 50 percent expect to use it 3 times a week. Rest 50 percent either 2 or 4 times

In [16]:

```python
# Get the value counts for the categorical colums
print("Mid Level Model KP481:")
print()
for column in categorical_cols:
    print(midLevel.loc[:,column].value_counts(normalize=True).round(2)*100)
    print()
    print("--------------XXX-------------")
    print()

# Get the value counts for Fitness & Usage columns
for column in ["Fitness","Usage"]:
    print(midLevel.loc[:,column].value_counts(normalize=True).round(2)*100)
    print()
    print("--------------XXX-------------")
    print()
```

```
Mid Level Model KP481:

KP481    100.0
Name: Product, dtype: float64

--------------XXX-------------

Male      52.0
Female    48.0
Name: Gender, dtype: float64

--------------XXX-------------

Partnered    60.0
Single       40.0
Name: MaritalStatus, dtype: float64
```

```
--------------XXX-------------

3    65.0
2    20.0
4    13.0
1     2.0
Name: Fitness, dtype: float64

--------------XXX-------------

3    52.0
2    23.0
4    20.0
5     5.0
Name: Usage, dtype: float64

--------------XXX-------------
```

Insights:

Regarding Customers that bought KP481 Model:

1. Men preferred this model a little bit more than Women
2. Partnered Customers proportion is more than Singles like entrylevel model
3. 65 percent have Fitness Level of 3, followed by ~20 percent at Fitness Level 2
4. About 50 percent expect to use it 3 times a week. Rest 50 percent- either 2 or 4 times

In [17]:

```python
# Get the value counts for the categorical colums
print("Advance Level Model KP781:")
print()
for column in categorical_cols:
    print(advanceLevel.loc[:,column].value_counts(normalize=True).round(2)*100)
    print()
    print("--------------XXX-------------")
    print()

# Get the value counts for Fitness & Usage columns
for column in ["Fitness","Usage"]:
    print(advanceLevel.loc[:,column].value_counts(normalize=True).round(2)*100)
    print()
    print("--------------XXX-------------")
    print()
```

```
Advance Level Model KP781:

KP781    100.0
Name: Product, dtype: float64

--------------XXX-------------

Male      82.0
Female    18.0
Name: Gender, dtype: float64

--------------XXX-------------
```

```
Partnered    57.0
Single       42.0
Name: MaritalStatus, dtype: float64

--------------XXX-------------

5    72.0
4    18.0
3    10.0
Name: Fitness, dtype: float64

--------------XXX-------------

4    45.0
5    30.0
6    18.0
7     5.0
3     2.0
Name: Usage, dtype: float64

--------------XXX-------------
```

Insights:

Regarding Customers that bought KP781 Model:

1. Men preferred this model way more than Women
2. Partnered Customers proportion is more than Singles like other two models
3. More than 70 percent have Fitness Level of 5, followed by ~20 percent at Fitness Level 4
4. Almost 100 percent expect to use the treadmill more than 4 times a week

So Far, this is what can be inferred:

Apparently, Male Customers in better shape, and high weekly expected usage, seem to buy KP781 model. As KP781 is pricey, explored Income levels for these customers

For the other two models, Customers tend to have similar Fitness levels, Usage, MaritalStatus, and Gender proportions. However, given higher number of entry level customers, looks like entry level is preferred more so far. May be other features would give more information to profile these Customers

In [18]:

```
# Based on above findings, two categorical features can be created as follows:
df.loc[:,"FitnessLevels"] = df.loc[:,"Fitness"].apply(lambda x:"Fitness_Above_3" if int(x) > 3 else "Fitness_Below_3")
df.loc[:,"WeeklyUsageCount"] = df.loc[:,"Usage"].apply(lambda x:"Usage_Above_3" if int(x) > 3 else "Usage_Below_3")
print(df["FitnessLevels"].value_counts())
print()
print(df["WeeklyUsageCount"].value_counts())
#pd.crosstab(index=[df["Product"],],columns=[df["FitnessLevels"],],margins=True)
```

```
Fitness_Below_3    125
Fitness_Above_3     55
Name: FitnessLevels, dtype: int64

Usage_Below_3    102
```

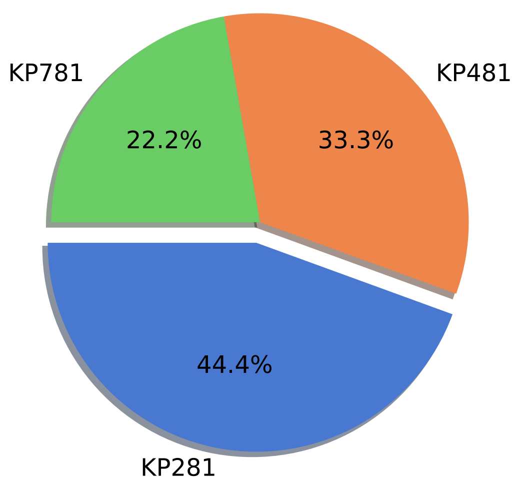
```
Usage_Above_3     78
Name: WeeklyUsageCount, dtype: int64
```

Visual Analysis:

Univariate Analysis:

In [19]:

```
# Pie Chart to show the distribution of the Customers for each treadmill model
sizes = df["Product"].value_counts().values
labels = ["KP281","KP481","KP781"]
plt.pie(sizes,explode=[0.1,0,0],labels=labels,shadow=True,autopct='%1.1f%%',startangle=180)
plt.show()
```

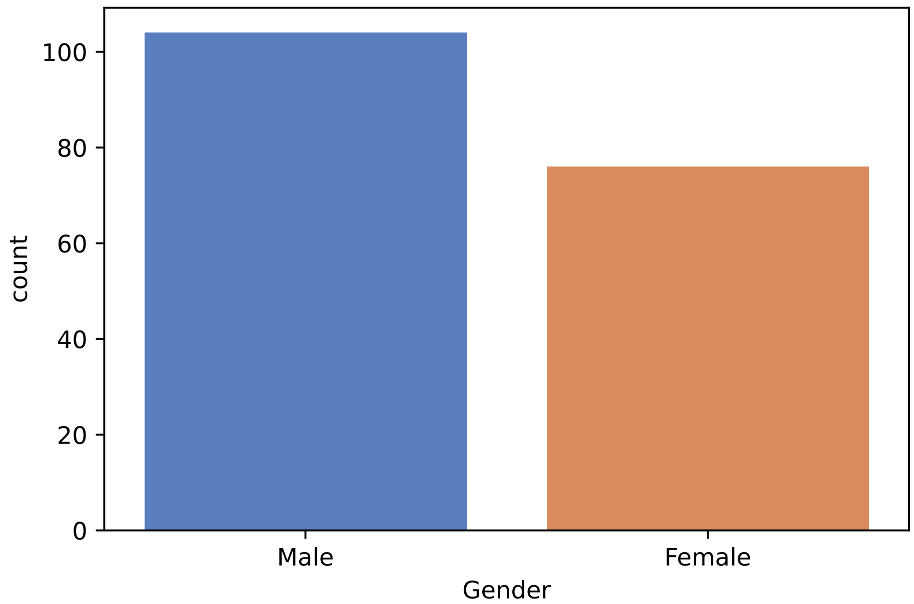


As we have seen in non-graphical analysis:

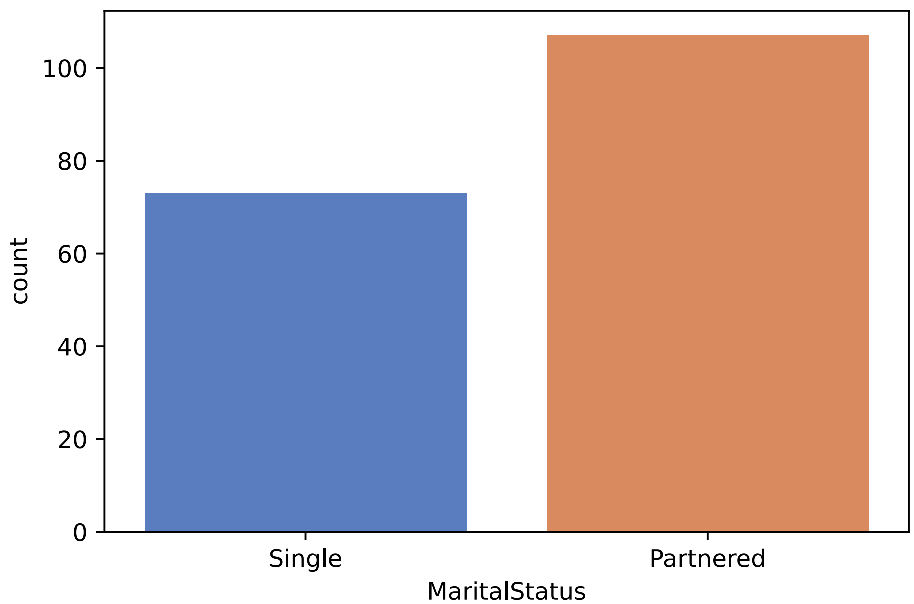
1. About 45 percent of the customers bought the entry level model KP281
2. Followed by 33.3 percent for mid level KP481 model and 22.2 percent for advanced model KP781

In [20]:

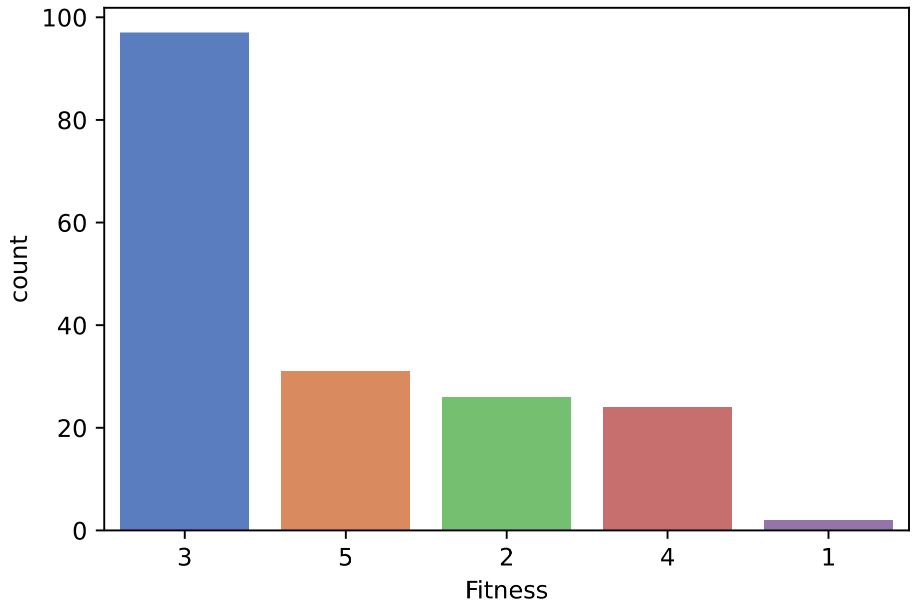
```
# Overall there are more Males than Females
sns.countplot(data=df,x="Gender")
plt.show()
```



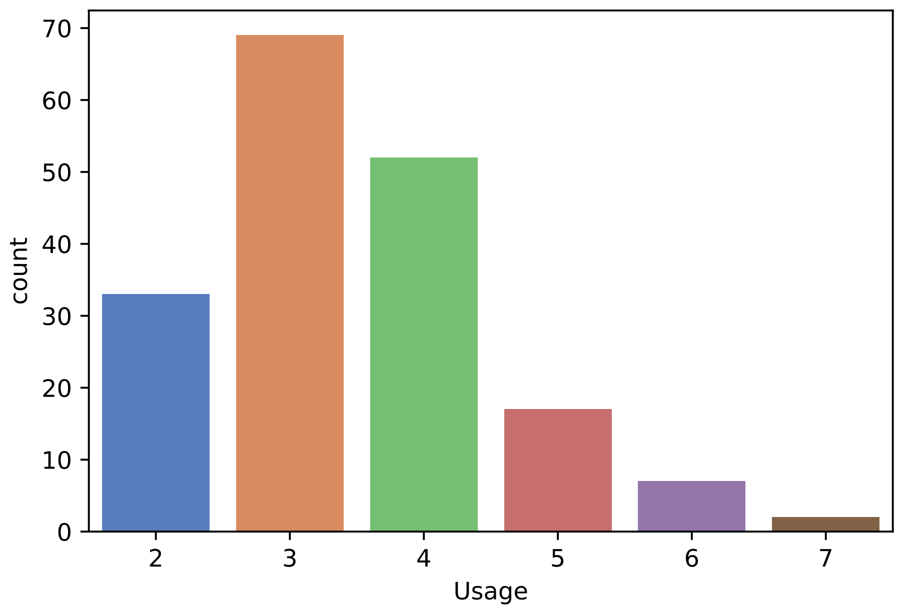
In [21]:
```
# There are more Married/Partnered Customers that bought one of the three treadmill Models
sns.countplot(data=df,x="MaritalStatus")
plt.show()
```



In [22]:
```
# There are more Customers with self rated Fitness level 3 with 1 being poor shape and 5 being in excellent shape
sns.countplot(data=df,x="Fitness",order=df["Fitness"].value_counts().index)
plt.show()
```



In [23]:
```
# Majority of Customers expect to use the treadmill 3 to 4 times a week
sns.countplot(data=df,x="Usage")
plt.show()
```

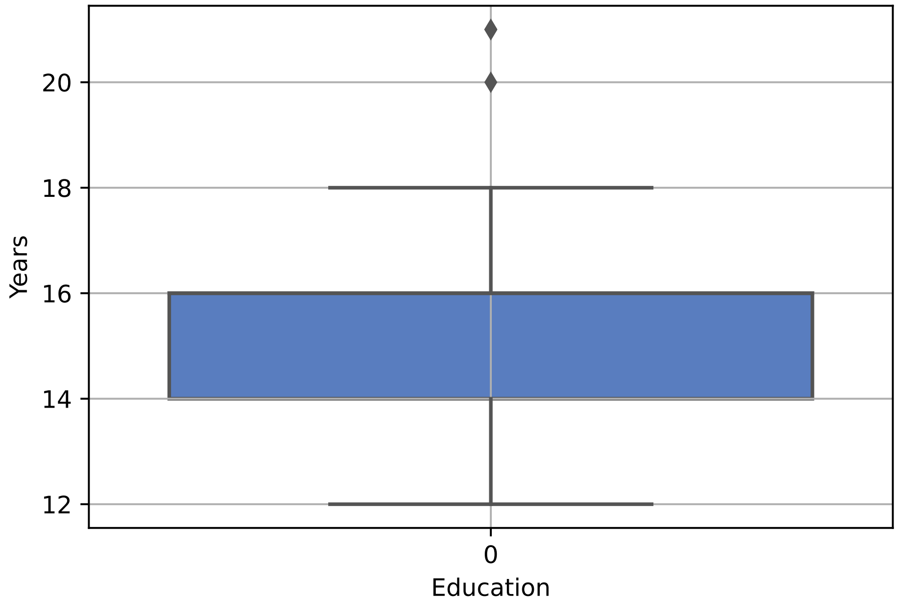


Insights for above Countplots:

1. Overall there are more Males than Females
2. There are more Married/Partnered Customers that bought one of the three treadmill Models
3. There are more Customers with self rated Fitness level 3 with 1 being poor shape and 5 being in excellent shape

4. Majority of Customers expect to use the treadmill 3 to 4 times a week

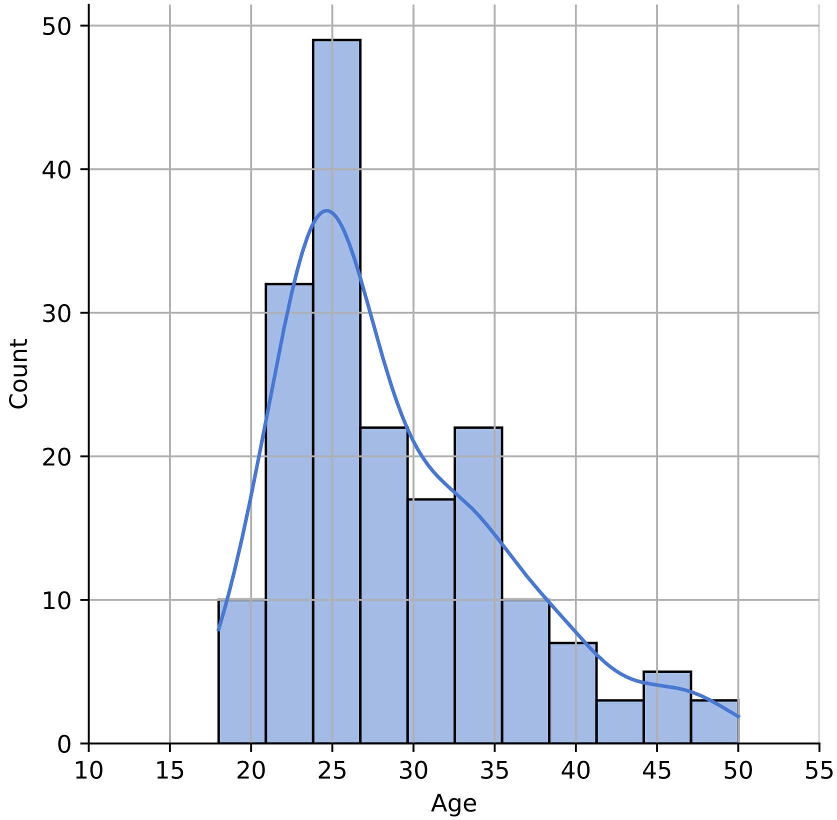
In [24]:
```
sns.boxplot(data=df["Education"],orient="v")
plt.ylabel("Years")
plt.xlabel("Education")
plt.grid(True)
plt.show()
```



Insights:

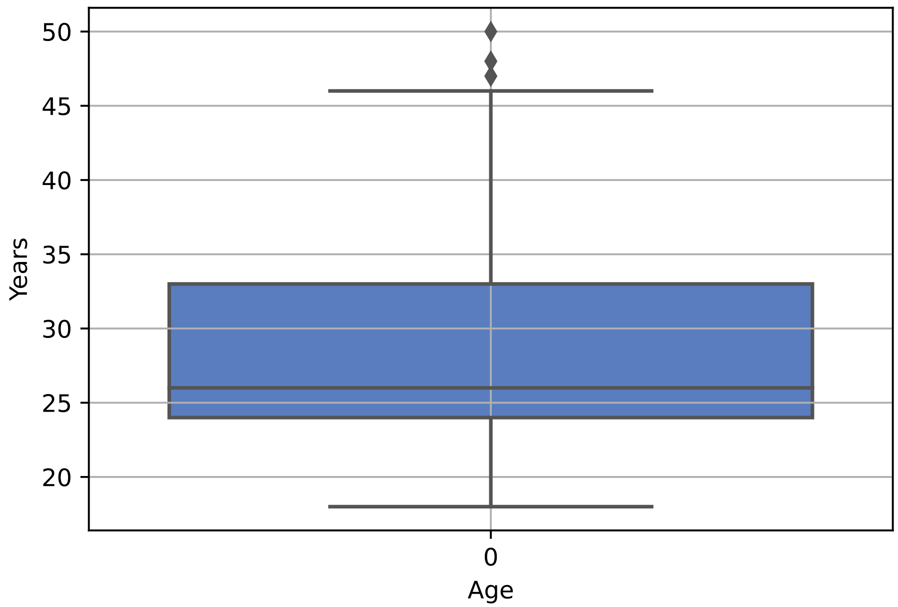
1. Abount 50% of Customers have 14 to 16 Years of education. Minium being 12 Years
2. There are couple of Customers with more than 20 Years of Education

In [25]:
```
sns.displot(df["Age"],kde=True,)
plt.xticks(ticks=np.arange(10,60,5))
plt.grid(True)
plt.show()
```



In [26]:

```
sns.boxplot(data=df["Age"],orient="v")
plt.grid(True)
plt.ylabel("Years")
plt.xlabel("Age")
plt.show()
```



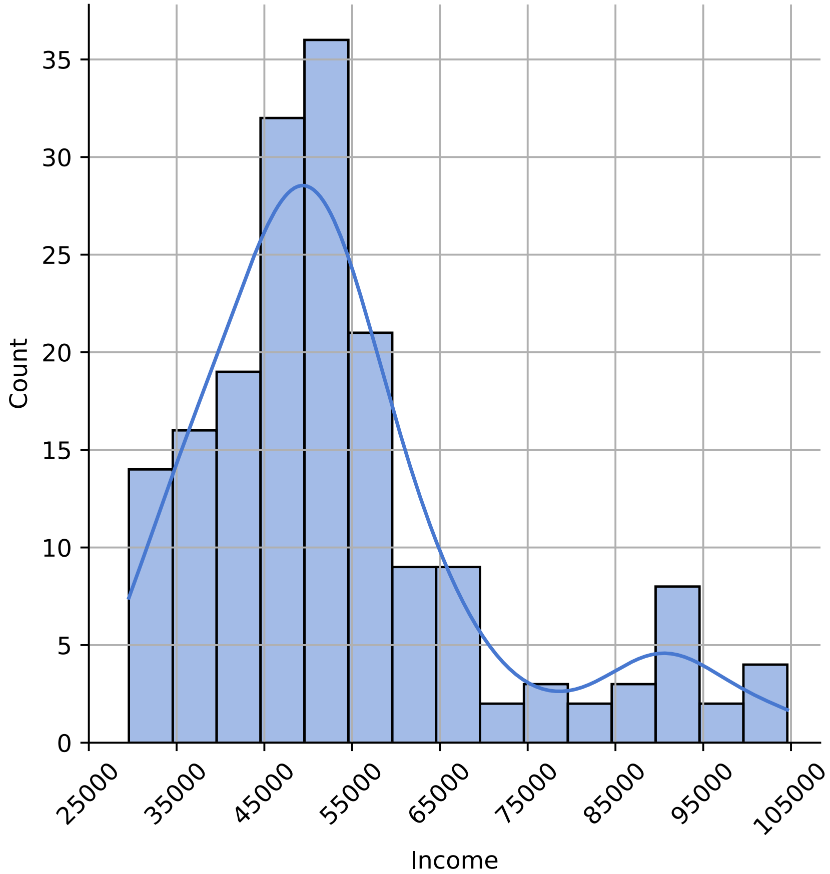
In [27]:
```python
# Quartile Calculations for Age
quartile1 = np.percentile(df["Age"],25)
quartile3 = np.percentile(df["Age"],75)
IQR = quartile3-quartile1
print(f" Quartile 1:  {quartile1}\n Quartile 3:  {quartile3}\n Minimum Age: {quartile1-1.5*IQR}\n Maximum Age: {quartile3+1.5*IQR}")
```

```
 Quartile 1:  24.0
 Quartile 3:  33.0
 Minimum Age: 10.5
 Maximum Age: 46.5
```

Insights:

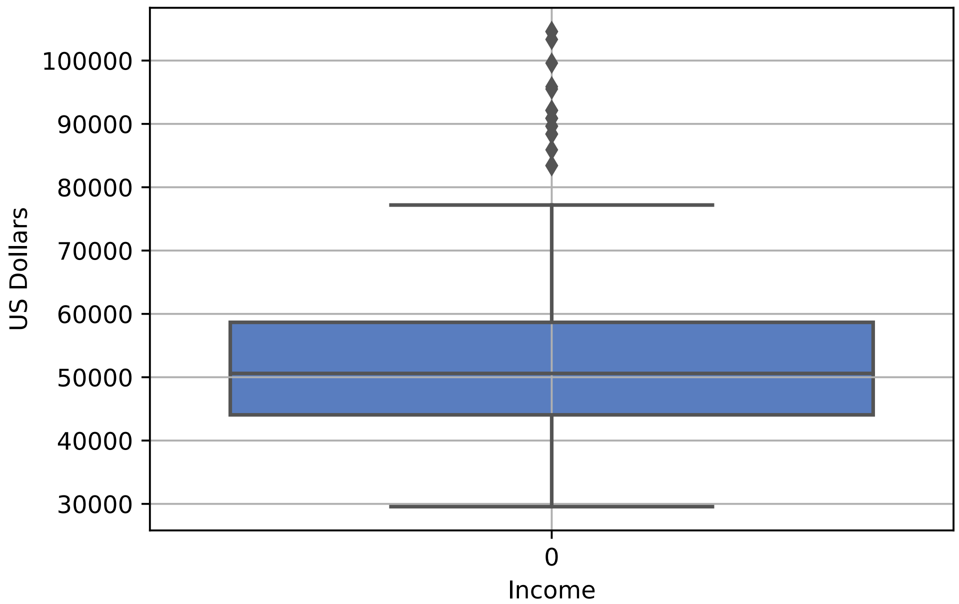
1. 75 percent of Customers that bought the treadmills are around 35 or younger
2. About 50 percent of Customers fall into Age range of 24 to 33 Years
3. There are Customers who are older than 46 years and those Customers could be considered outliers
4. Ages of Customers range from 18 to 50 Years

In [28]:
```python
sns.displot(df["Income"],kde=True)
plt.xticks(ticks=np.arange(25000,110000,10000),rotation=45)
plt.grid(True)
plt.show()
```



In [29]:

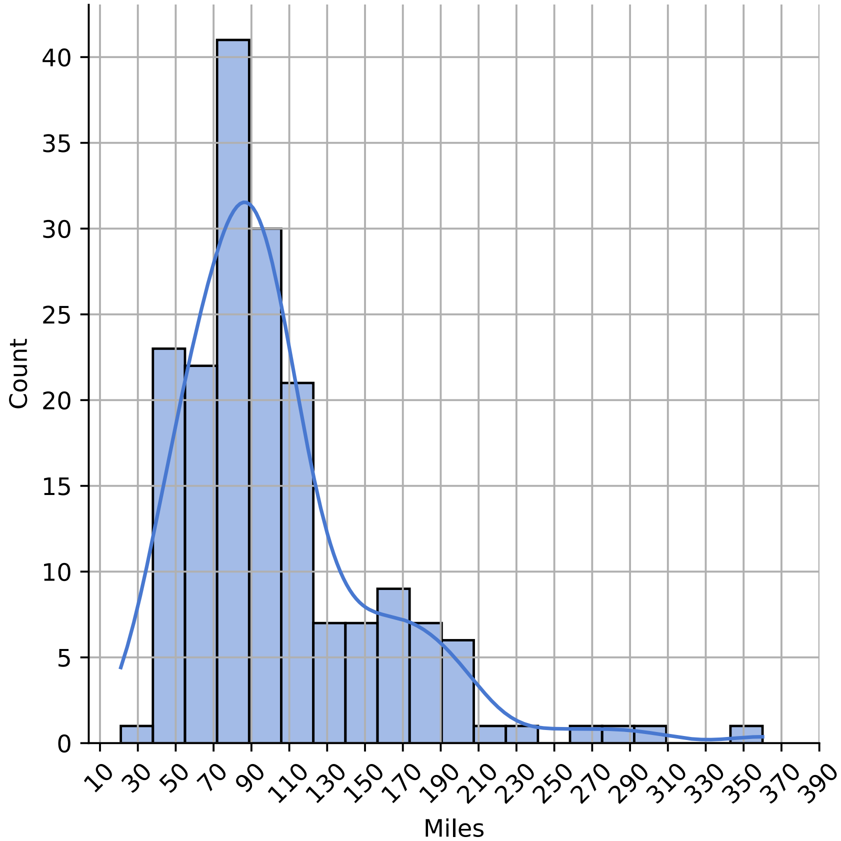
```
sns.boxplot(data=df["Income"],orient="v")
plt.grid(True)
plt.ylabel("US Dollars")
plt.xlabel("Income")
plt.show()
```



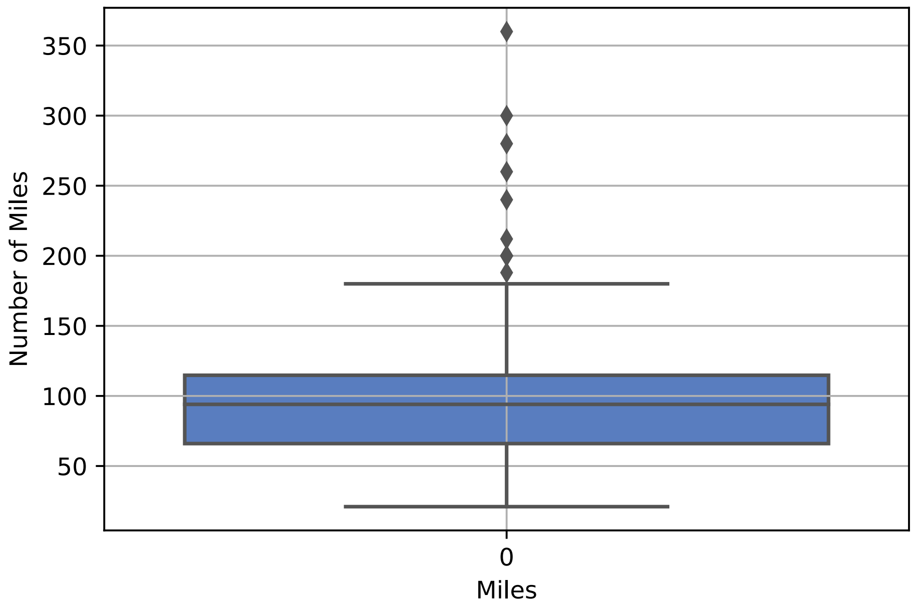
Insights:

1. Majority of Customers have an annual salary of 45,000 dollars to 55,000 dollars
2. Any Customers with salary on the higher side of 80,000 dollars can be considered to be outliers

In [30]:
```
sns.displot(df["Miles"],kde=True)
plt.xticks(ticks=np.arange(10,400,20),rotation=45)
plt.grid(True)
plt.show()
```



In [31]:
```
sns.boxplot(data=df["Miles"],orient="v")
plt.grid(True)
plt.ylabel("Number of Miles")
plt.xlabel("Miles")
plt.show()
```



In [32]:
```
# Calculation of quartiles for Miles
quartile1 = np.percentile(df["Miles"],25)
quartile3 = np.percentile(df["Miles"],75)
IQR = quartile3-quartile1
print(f"Quartile 1:  {quartile1}\nQuartile 3:  {quartile3}\nMinimum Miles: {quartile1-1.5*IQR}\nMaximum Miles: {quartile3+1.5*IQR}")
```

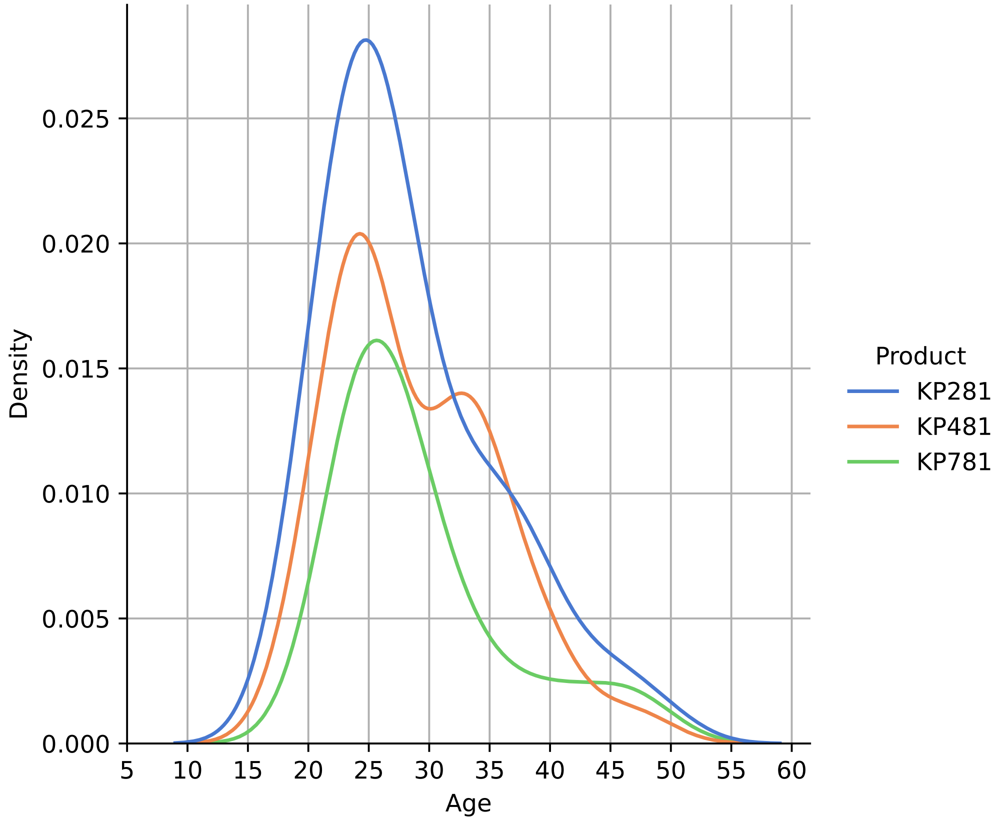
```
Quartile 1:  66.0
Quartile 3:  114.75
Minimum Miles: -7.125
Maximum Miles: 187.875
```

Insights:

1. About 50% of Customers expect to walk/run 66 miles to 115 miles per week
2. Customers with expected Miles above ~190 miles can be considered outliers. There are few such outliers

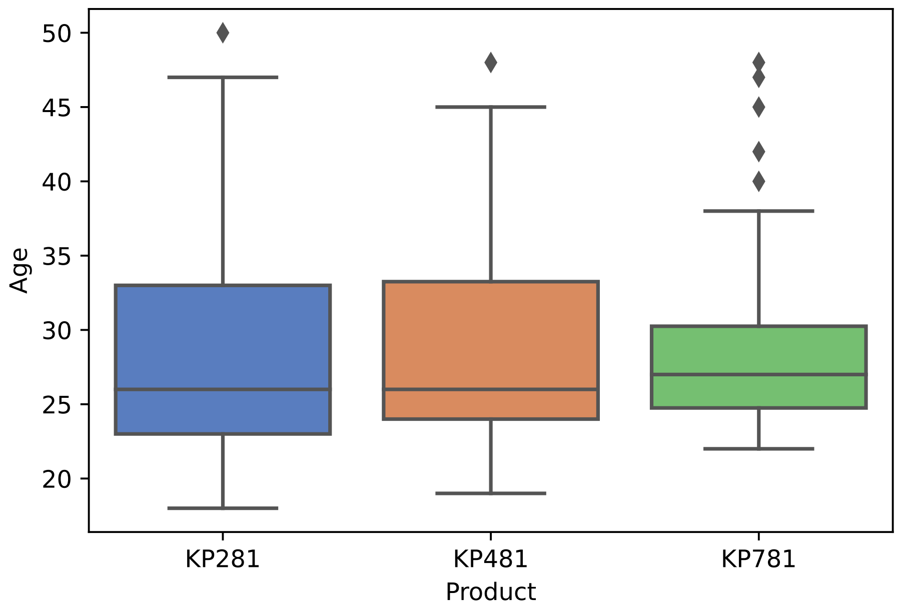
Bivariate Analysis

In [33]:
```
sns.displot(data=df,x="Age",hue="Product",kind="kde")
plt.xticks(ticks=np.arange(5,61,5))
plt.grid(True)
plt.show()
```



In [34]:

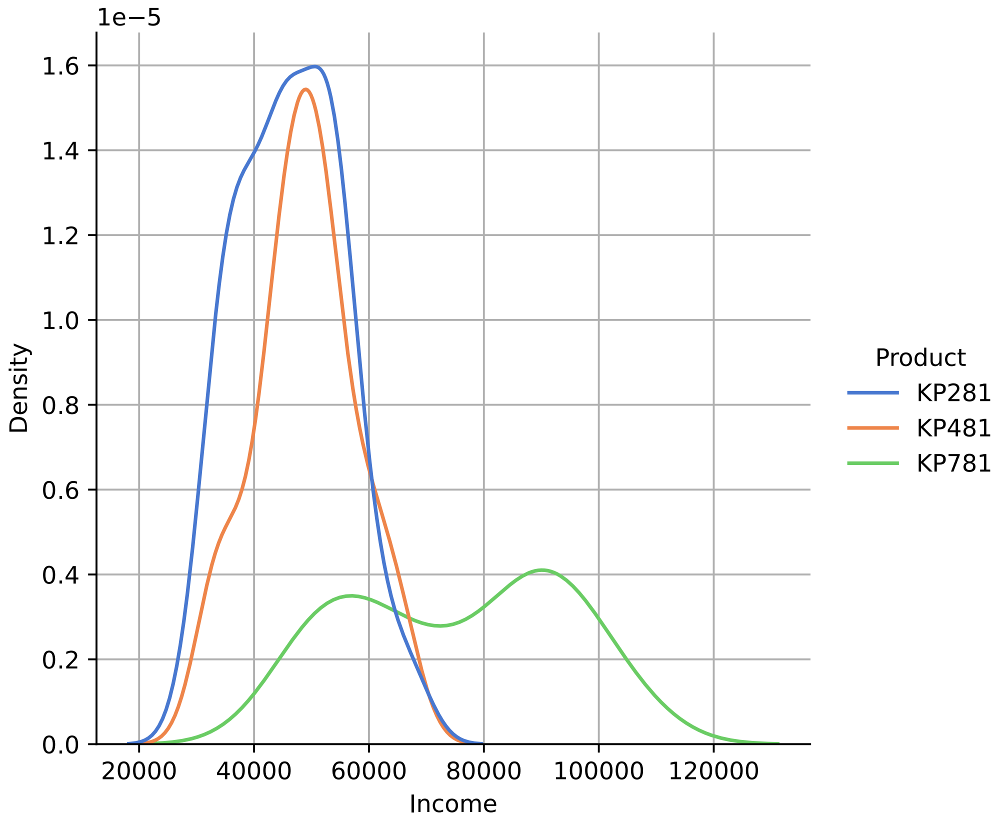
```
sns.boxplot(data=df,y="Age",x="Product")
plt.show()
```



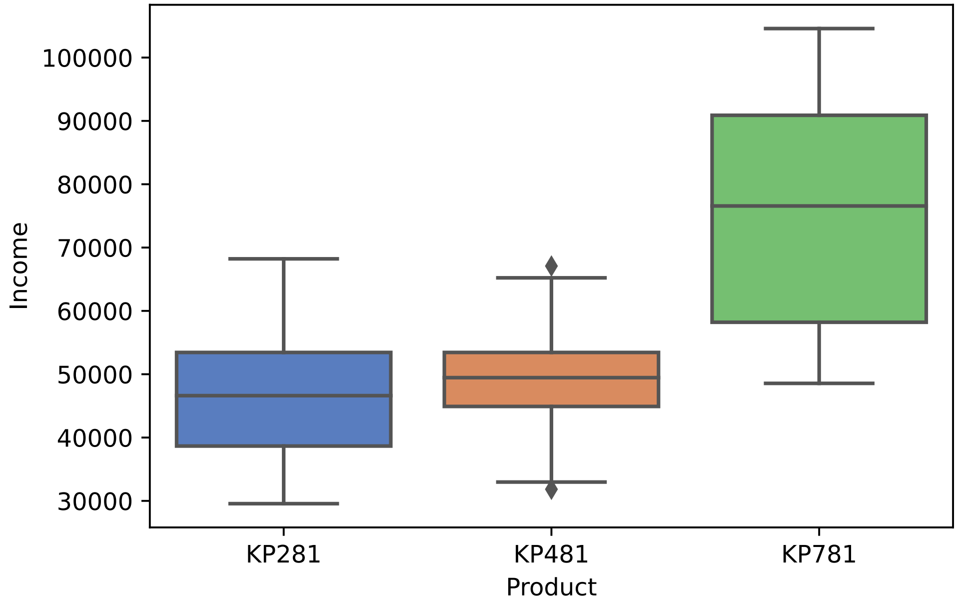
Insights:

1. For all three treadmill models, Age of the Customers is spread throught out. Therefore, it is not easy to say that Customers of specific age range, tend to buy a specific treadmill model. Typical Customers age range is 25 to 35 Years
2. Minimum & median age of Customers that preferred the advanced model, are higher than the other two models

In [35]:
```
sns.displot(data=df,x="Income",hue="Product",kind="kde")
plt.grid(True)
plt.show()
```



In [36]:
```
sns.boxplot(data=df,y="Income",x="Product")
plt.show()
```

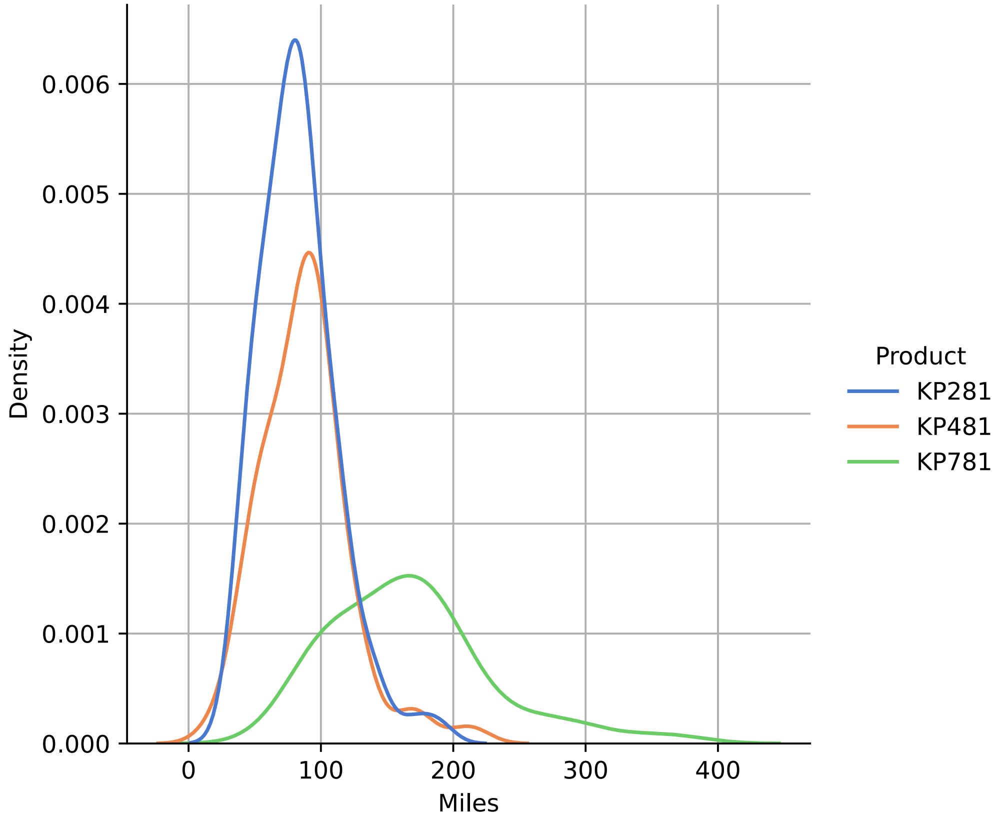


Insights:

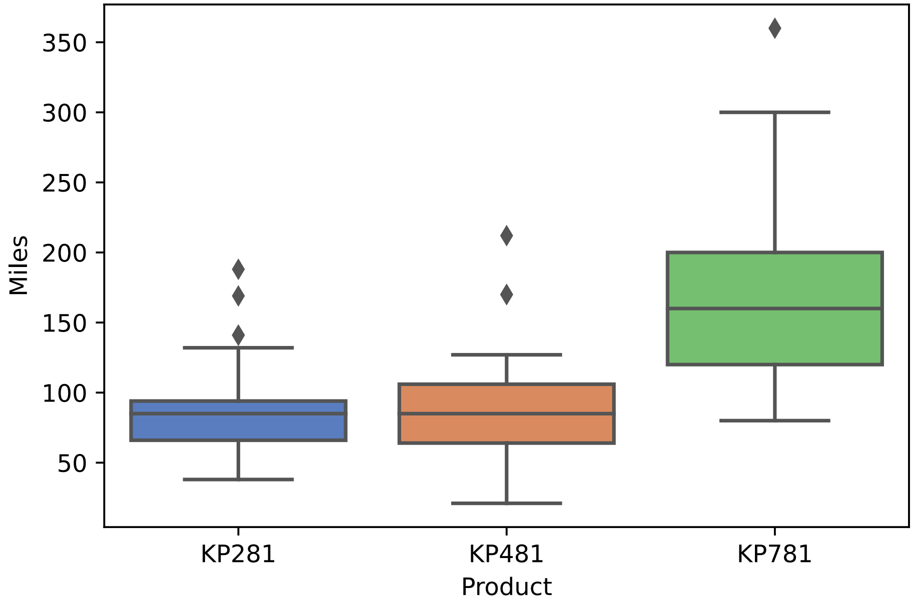
1. Minimum Income of the customers that bought advanced model is >= to the median income of customers that bought the other two models
2. Income could be used as a feature to segregate Customers, who purchased the advanced model, from Customers who did not

In [37]:

```
sns.displot(data=df,x="Miles",hue="Product",kind="kde")
plt.grid(True)
plt.show()
```



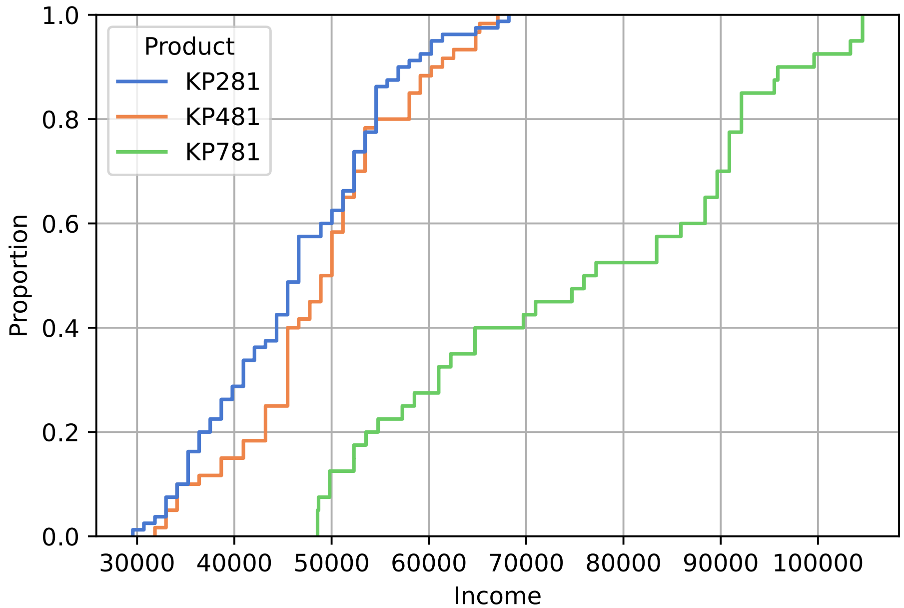
In [38]:
```
sns.boxplot(data=df,y="Miles",x="Product")
plt.show()
```



Insights:

1. Miles expected to walk/run per week could be another feature to segregate Advanced model Customers
2. No clear way to separate Customers of other two models using Miles

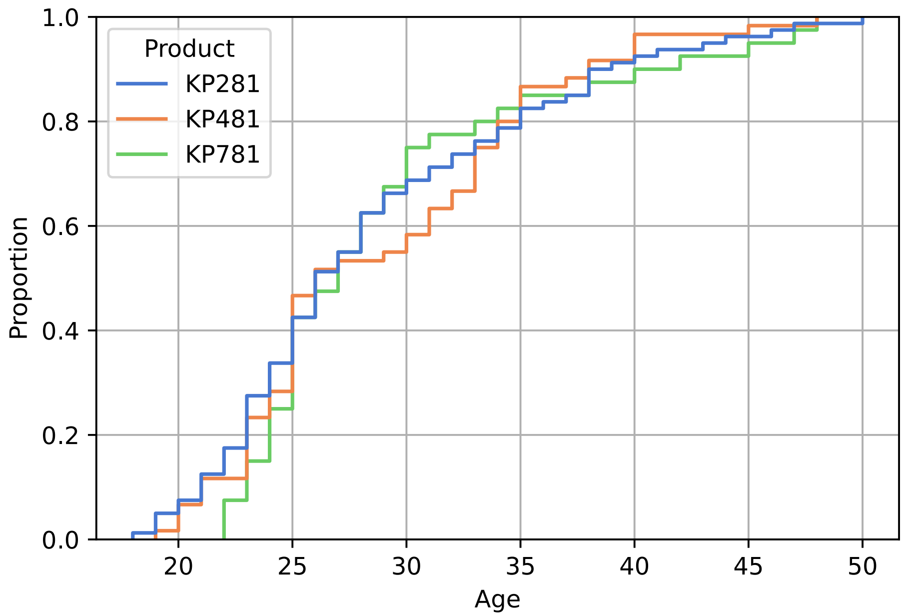
In [39]:
```
sns.ecdfplot(data=df,x="Income",hue="Product")
plt.grid(True)
```



Insights:

1. From the above plot, it is evident that 80 percent of Customers that bought the advanced model KP781, have an annual salary >= 55,000 dollars
2. Abought 90 percent of the Customers that bought the entry level and mid level models have an annual salary of 60,000 dollars or less

In [40]:
```
# Cumulative Density Function plot for Age
sns.ecdfplot(data=df,x="Age",hue="Product")
plt.grid(True)
```

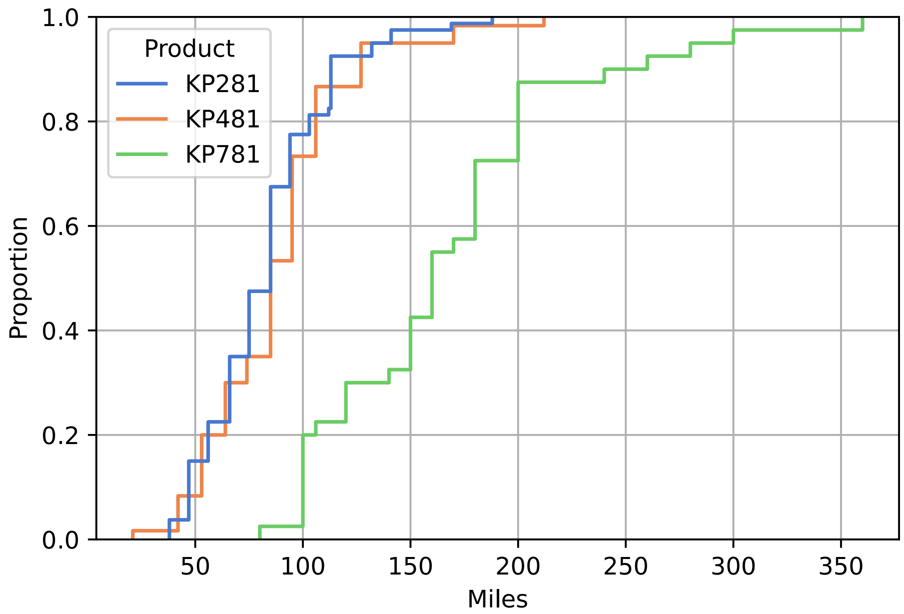


Insights:

1. When it comes to age of Customers, all three models are preferred by Age ranges from around 20 Years till 50 Years.
2. KP781 model is the model that Customers of age 30 to 35 Years preferred the most
3. Customers of age 35 or more preferred mid level model KP481

In [41]:

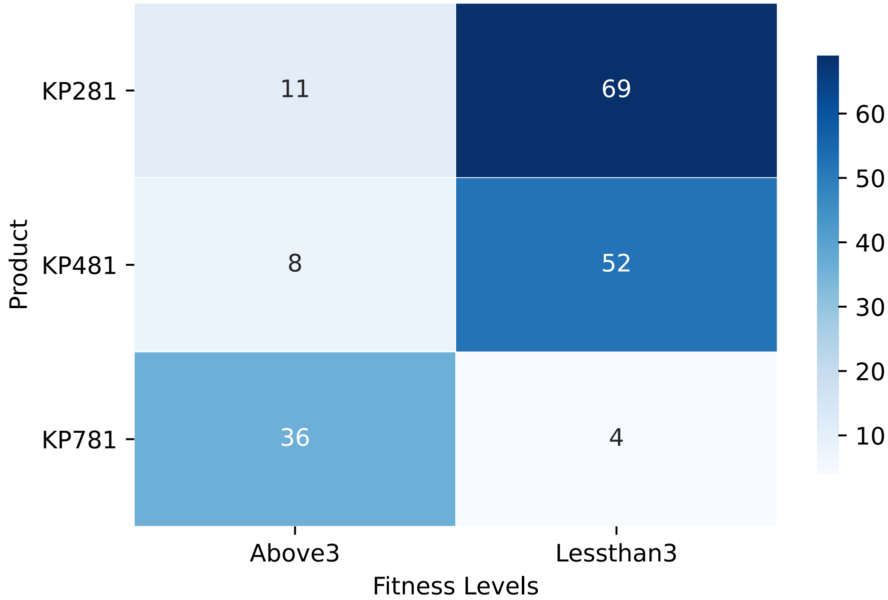
```
# Cumulative Density Function plot of Miles per week
sns.ecdfplot(data=df,x="Miles",hue="Product")
plt.grid(True)
```



Insights:

1. Around 60 percent of Advanced Model buyers, expect to run/walk more than 150 Miles a week. Minimum being ~75 Miles.
2. About 90% of Customers that bought entry level & mid level model expect to run/walk 125 miles or less.

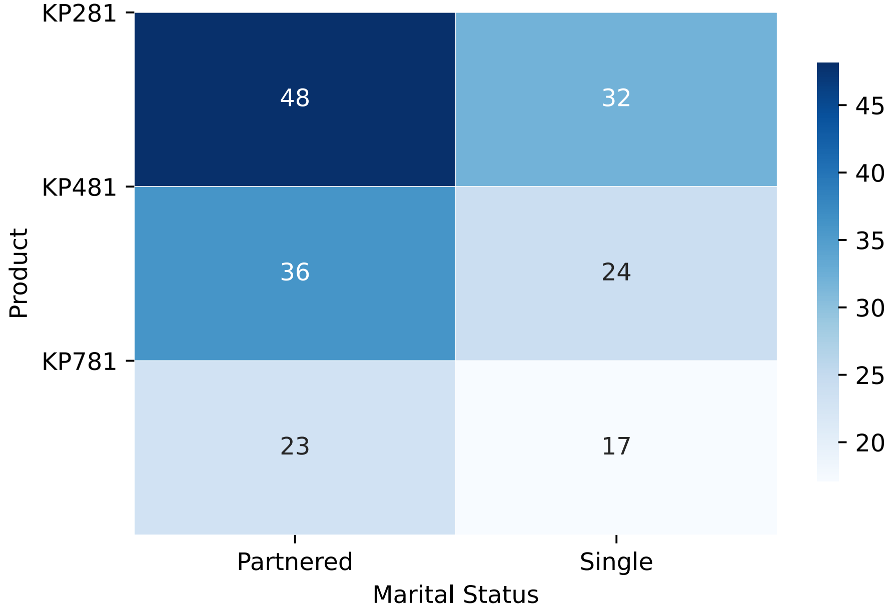
In [42]:
```
# Heatmap of Fitness Levels & Product
sns.heatmap(data=pd.crosstab(index=df["Product"]
            ,columns=df["FitnessLevels"]).values
            ,cbar_kws={"shrink": .8},linewidths=0.3,cmap="Blues",annot=True)
plt.yticks(ticks=[0.5,1.5,2.5],labels=["KP281","KP481","KP781"],rotation = 0)
plt.xticks(ticks=np.arange(1,3,1)-0.5,labels=["Above3","Lessthan3"])
plt.xlabel("Fitness Levels")
plt.ylabel("Product")
plt.show()
```



Insights:

1. Customers who bought entry level & mid level models rate themselves as Average Shape
2. Most of the Customers that bought the advanced model rated themselves as Excellent Shape

In [43]:
```
# Heatmap of Product and Marital Status
sns.heatmap(data=pd.crosstab(index=df["Product"]
            ,columns=df["MaritalStatus"]).values
            ,cbar_kws={"shrink": .8},linewidths=0.3,cmap="Blues",annot=True)
plt.yticks(ticks=[0,1,2],labels=["KP281","KP481","KP781"],rotation = 0)
plt.xticks(ticks=[0.5,1.5],labels=["Partnered","Single"])
plt.xlabel("Marital Status")
plt.ylabel("Product")
plt.show()
```

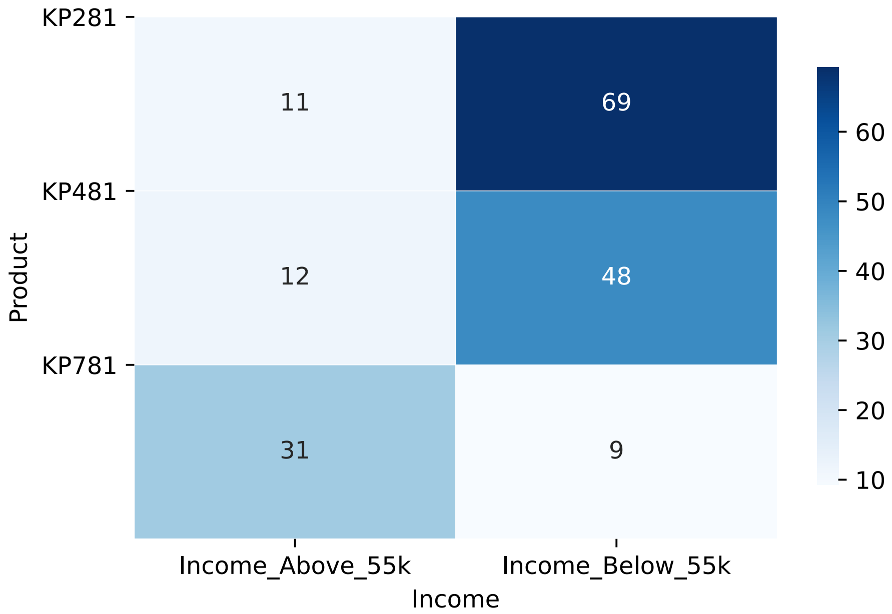


Insights:

1. Overall, Partnered Customers are more than Single Customers for very model

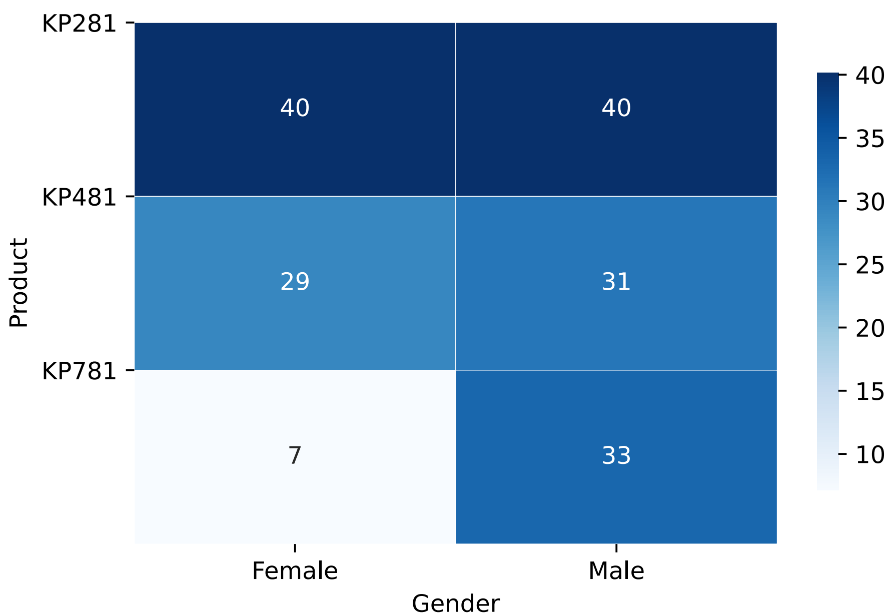
In [44]:
```
# Created a new categorical feature IncomeLevel
df.loc[:,"IncomeLevel"] = df.loc[:,"Income"].apply(lambda x:"Income_Above_55k" if int(x) > 55000 else "Income_Below_55k")
```

In [45]:
```
# Heatmap of Product and New feature Income Level
sns.heatmap(data=pd.crosstab(index=df["Product"]
            ,columns=df["IncomeLevel"]).values
            ,cbar_kws={"shrink": .8},linewidths=0.3,cmap="Blues",annot=True)
plt.yticks(ticks=[0,1,2],labels=["KP281","KP481","KP781"],rotation = 0)
plt.xticks(ticks=[0.5,1.5],labels=["Income_Above_55k","Income_Below_55k"])
plt.xlabel("Income")
plt.ylabel("Product")
plt.show()
```



In [46]:

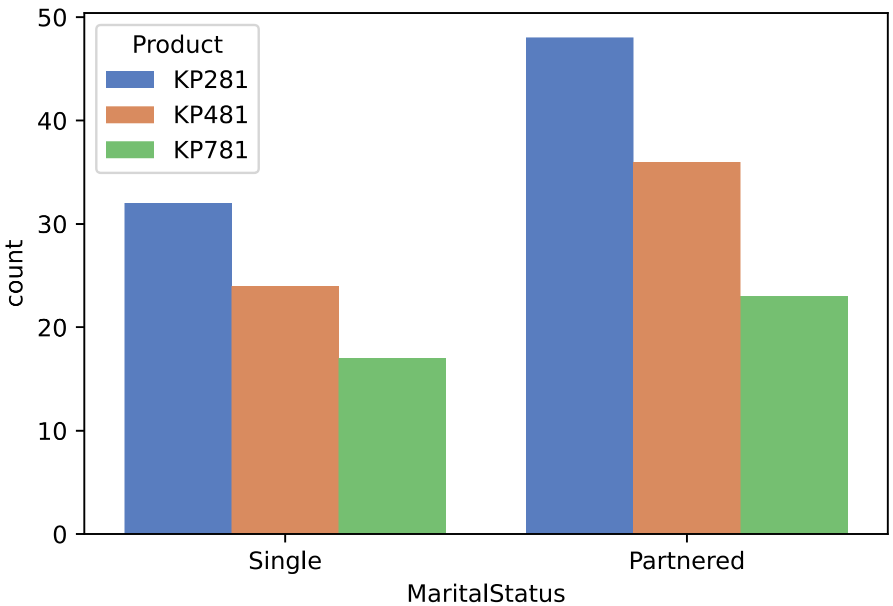
```
# Heatmap of Product and Marital Status
sns.heatmap(data=pd.crosstab(index=df["Product"]
            ,columns=df["Gender"]).values
            ,cbar_kws={"shrink": .8},linewidths=0.3,cmap="Blues",annot=True)
plt.yticks(ticks=[0,1,2],labels=["KP281","KP481","KP781"],rotation = 0)
plt.xticks(ticks=[0.5,1.5],labels=["Female","Male"])
plt.xlabel("Gender")
plt.ylabel("Product")
plt.show()
```



Insights:

1. Above three tables are 2-way contingency tables with respect to model & Features like Income, Gender, and Marital Status. We will explore more on the probabilities towards the end

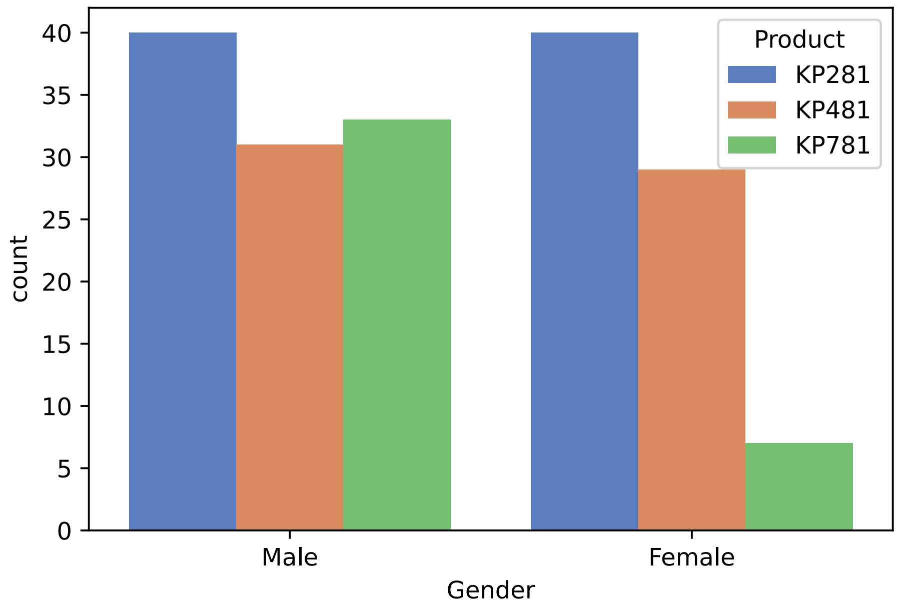
In [47]:
```
# Countplot between Product & MaritalStatus
sns.countplot(data=df,x="MaritalStatus",hue="Product")
plt.show()
```



Insights:

1. As seen earlier, Partnered Customers are more than Singles. Order of product preference remains the same in both marital status categories.

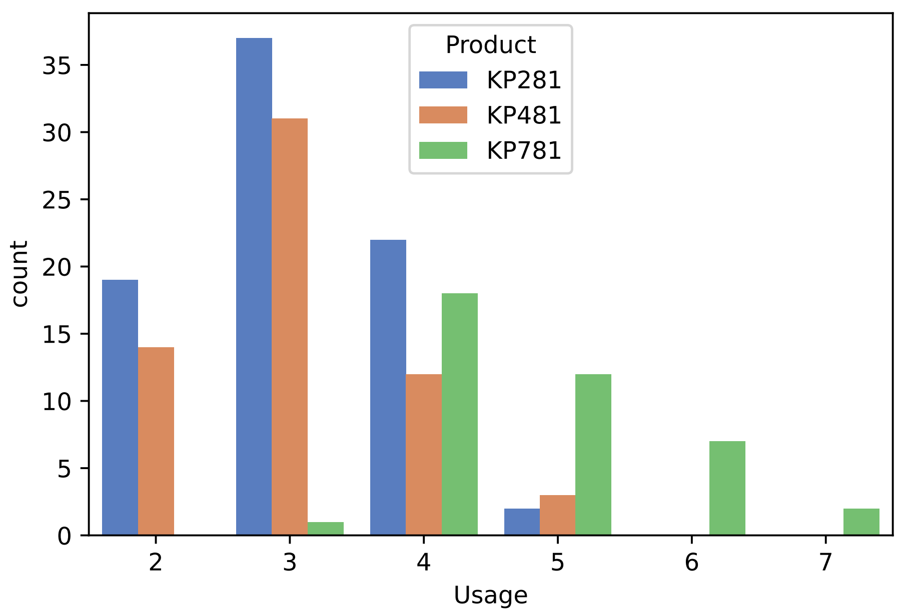
In [48]:
```
# Countplot between Product & Gender
sns.countplot(data=df,x="Gender",hue="Product")
plt.show()
```



Insights:

1. Advanced Model is not as preferred by Females as it is preferred by Males
2. For the other two models, Males & Females proportion is almost identical

In [49]:
```
# Countplot between Product & Usage
sns.countplot(data=df,x="Usage",hue="Product")
plt.show()
```



Insights:

1. Majority of Customers that purchased beginner or mid level model expect to use it for 3 times a week
2. Majority of Customers that purchased advanced model expect to use it for 4 or 5 times a week

In [50]:
```
def get_model_price(model):
    """
    Given the model, return the price of the model

    """
    if model == "KP281":
        return 1500
    elif model == "KP481":
        return 1750
    else:
        return 2500
```

In [51]:
```
df.loc[:,"TreadMillPrice"] = df.loc[:,"Product"].apply(get_model_price)
```

In [52]:
```
df.loc[:,"SpentPercentage"] = df.loc[:,"TreadMillPrice"] * 100/df.loc[:,"Income"]
df.head()
```

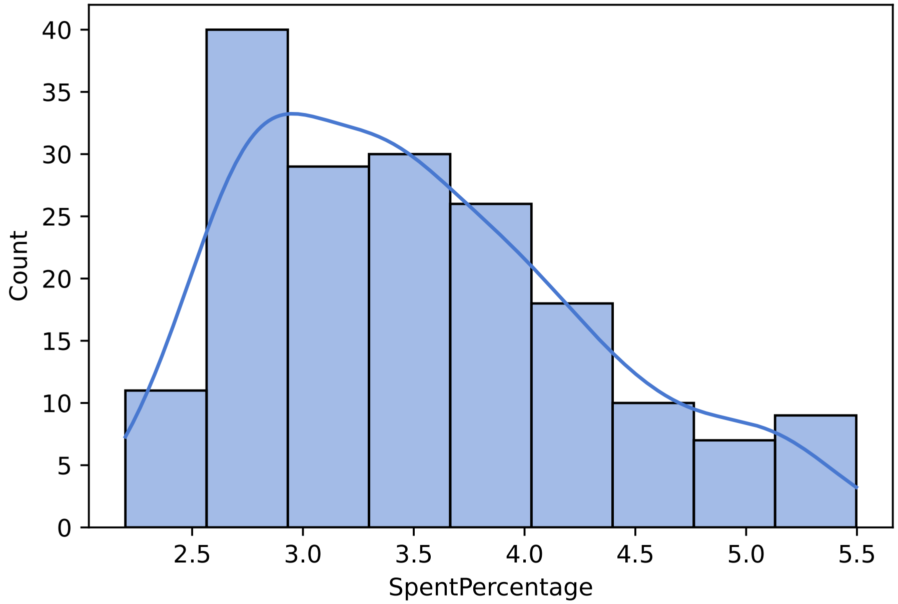
Out[52]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles	FitnessLevels	WeeklyUsageCount	IncomeLevel	TreadMillPrice	SpentPerc
0	KP281	18	Male	14	Single	3	4	29562	112	Fitness_Above_3	Usage_Below_3	Income_Below_55k	1500	5.
1	KP281	19	Male	15	Single	2	3	31836	75	Fitness_Below_3	Usage_Below_3	Income_Below_55k	1500	4
2	KP281	19	Female	14	Partnered	4	3	30699	66	Fitness_Below_3	Usage_Above_3	Income_Below_55k	1500	4.
3	KP281	19	Male	12	Single	3	3	32973	85	Fitness_Below_3	Usage_Below_3	Income_Below_55k	1500	4
4	KP281	20	Male	13	Partnered	4	2	35247	47	Fitness_Below_3	Usage_Above_3	Income_Below_55k	1500	4

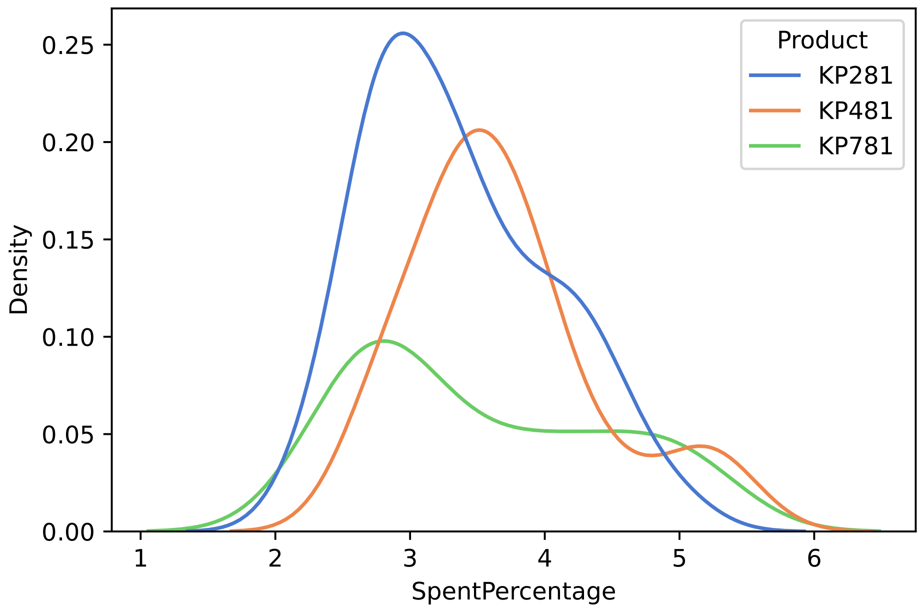
Insights:

1. Created a new feature called SpentPercentage - Model Cost as a percentage of Customer's annual income
2. As shown below, Customers prefer to spend 2.5 to 4 percent of annual income towards treadmill purchase

In [53]:
```
sns.histplot(data=df,x="SpentPercentage",kde=True,)
plt.show()
```

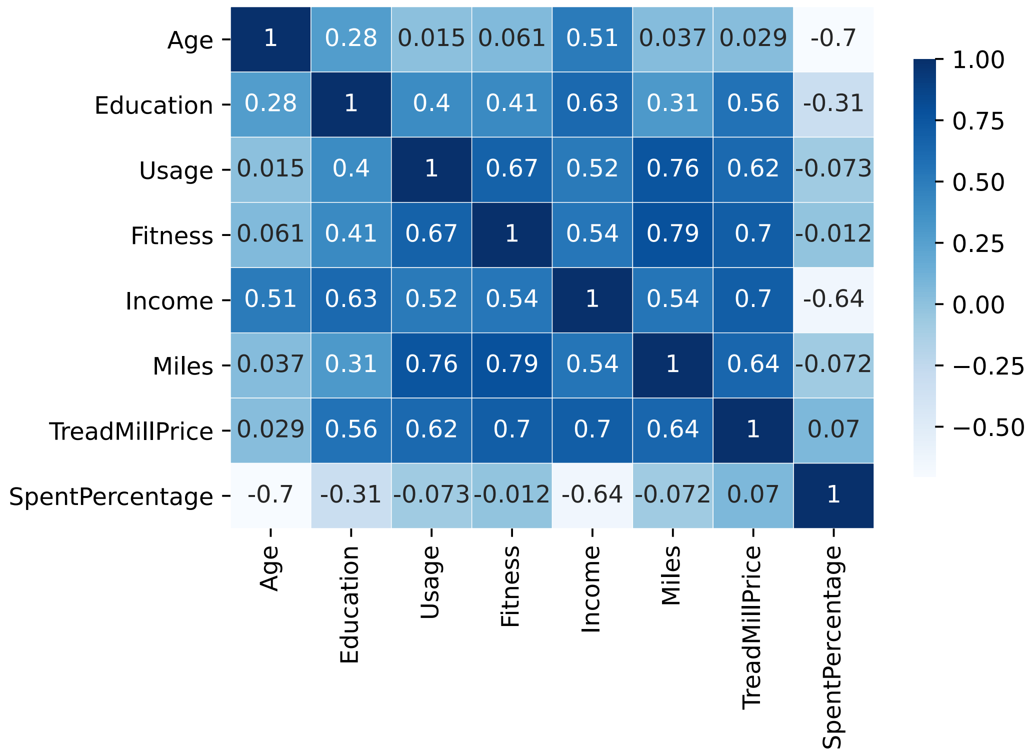


In [54]:
```
sns.kdeplot(data=df,x="SpentPercentage",hue="Product")
plt.show()
```



Correlation:

In [55]:
```
# Heatmap of correlation between different features:
sns.heatmap(data= df.corr()
            ,cbar_kws={"shrink": .8},linewidths=0.3,cmap="Blues",annot=True)
plt.yticks(rotation = 0)
plt.show()
```



	Age	Education	Usage	Fitness	Income	Miles	TreadMillPrice	SpentPercentage
Age	1	0.28	0.015	0.061	0.51	0.037	0.029	-0.7
Education	0.28	1	0.4	0.41	0.63	0.31	0.56	-0.31
Usage	0.015	0.4	1	0.67	0.52	0.76	0.62	-0.073
Fitness	0.061	0.41	0.67	1	0.54	0.79	0.7	-0.012
Income	0.51	0.63	0.52	0.54	1	0.54	0.7	-0.64
Miles	0.037	0.31	0.76	0.79	0.54	1	0.64	-0.072
TreadMillPrice	0.029	0.56	0.62	0.7	0.7	0.64	1	0.07
SpentPercentage	-0.7	-0.31	-0.073	-0.012	-0.64	-0.072	0.07	1

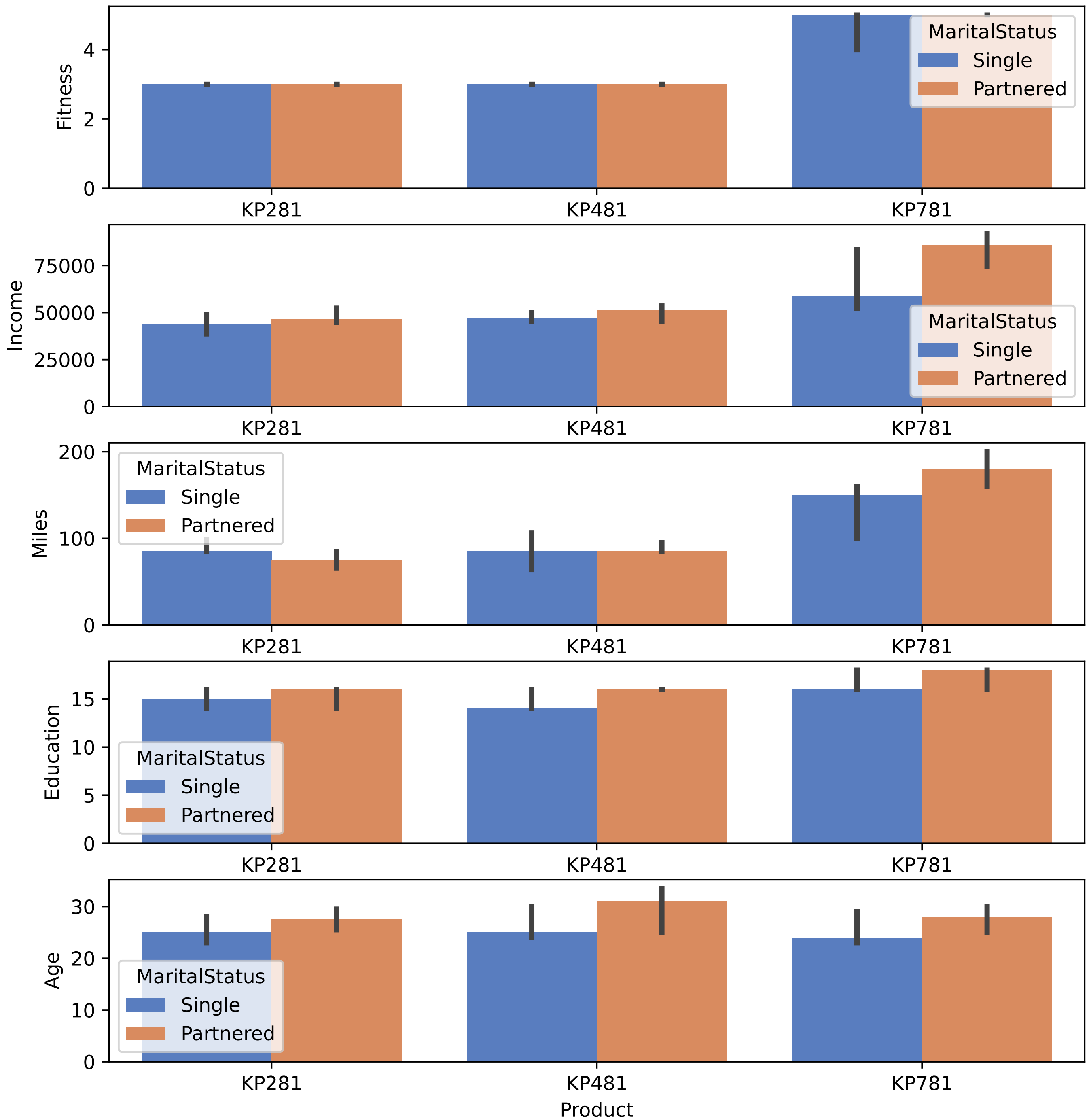
Insights:

1. Customer's age doesn't seem to have any correlation with Education, Usage, Fitness, Income, or Miles.
2. Education seem to have significant positive correlation with Income. Skipping Education in further analysis
3. Expected Miles, Fitness, and Usage are positively correlated -- Therefore, considering just the Fitness levels for further analysis

Multivariate Analysis:

In [56]:

```
fig,ax = plt.subplots(nrows=5,ncols=1,figsize=(9,10))
for i,col in enumerate(["Fitness","Income","Miles","Education","Age"]):
    for j,col2 in enumerate(["Product"]):
        sns.barplot(data=df,x=col2,y=col,ax=ax[i],hue="MaritalStatus",estimator=np.median)
plt.show()
```

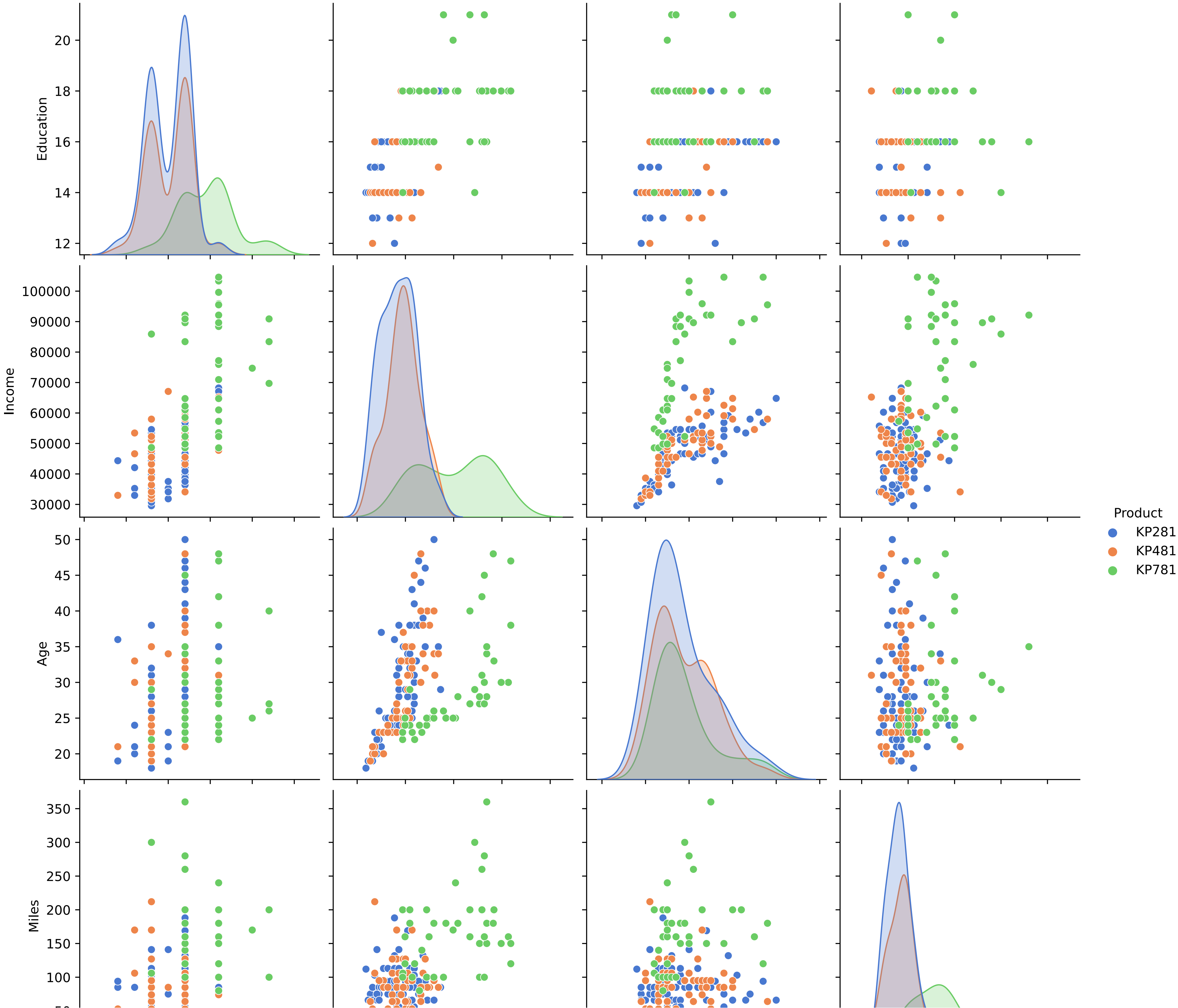


Insights:

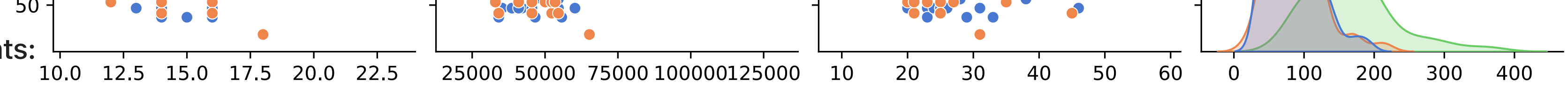
1. Customers that bought KP781 model have higher median income, Education, Miles, Fitness Levels
2. Gender, Marital Status features proportions are almost same in KP281, KP481 models

In [57]:

```
sns.pairplot(data= df[["Education","Income","Age","Miles","Product",]],hue="Product",height=3,kind="scatter")
plt.show()
```



Education
Income
Age
Miles
Product
KP281
KP481
KP781



Insights:

1. Clearly, KP781 model is bought by Customers with Higher income levels
2. KP281, KP481 models are clearly inseparable from one another

Contingency Table:

In [58]:
```python
df2 = pd.crosstab(index=df["Product"],columns=[df["Gender"]],margins=True,margins_name="All")
for i,column in enumerate(["MaritalStatus","IncomeLevel","FitnessLevels"]):
    temp = pd.crosstab(index=df["Product"],columns=df[column],margins=True,margins_name="All")
    df2 = df2.merge(temp,how="inner",on="Product")
df2.columns = ['Female', 'Male', 'All_x', 'Partnered', 'Single', 'All_y',
       'Income_Above_55k', 'Income_Below_55k', 'All_x', 'Fitness_Above_3',
       'Fitness_Below_3', 'All']
df2.drop(columns=["All_x","All_y"],inplace=True)
df2
```

```
<ipython-input-58-01553c7814f5>:4: FutureWarning: Passing 'suffixes' which cause duplicate columns {'All_x'} in the result is deprecated and will raise a MergeError in a future version.
  df2 = df2.merge(temp,how="inner",on="Product")
```

Out[58]:

Product	Female	Male	Partnered	Single	Income_Above_55k	Income_Below_55k	Fitness_Above_3	Fitness_Below_3	All
KP281	40	40	48	32	11	69	11	69	80
KP481	29	31	36	24	12	48	8	52	60
KP781	7	33	23	17	31	9	36	4	40
All	76	104	107	73	54	126	55	125	180

Comments:

1. Based on our earlier analysis, only Gender, Income, Marital Status, and Fitness levels are considered for creating the Contingency table
2. Below, calculated Mariginal Probabilities, Conditional Probabilities for models & Features

In [59]:
```python
counts = {}
customers = {}
for column in df2.columns:
    count = df2.loc["All",column]
    counts[column] = count

for index in df2.index:
    count = df2.loc[index,"All"]
    customers[index] = count
```

Marginal Probabilities:

In [60]:
```python
marginal_probabilities = {}
for key,val in customers.items():
    if key != "All":
```

```
        marginal_probabilities[f"P[{key}]"] = np.round(val/customers["All"],2)

for key,val in counts.items():
    if key != "All":
        marginal_probabilities[f"P[{key}]"] = np.round(val/counts["All"],2)

print("Marginal Probabilities of Features & Models:")
print()
for key in sorted(marginal_probabilities, key=lambda k: marginal_probabilities[k],reverse=True):
    print(f"{key} = {marginal_probabilities[key]}")
```

```
Marginal Probabilities of Features & Models:

P[Income_Below_55k] = 0.7
P[Fitness_Below_3] = 0.69
P[Partnered] = 0.59
P[Male] = 0.58
P[KP281] = 0.44
P[Female] = 0.42
P[Single] = 0.41
P[KP481] = 0.33
P[Fitness_Above_3] = 0.31
P[Income_Above_55k] = 0.3
P[KP781] = 0.22
```

Conditional Probabilities of Features, given the model:

In [61]:

```
cond_probs_features_given_model = {}
for index in df2.index:
    if index != "All":
        for key in counts.keys():
            if key != "All":
                probability = np.round(df2.loc[index,key]/customers[index],2)
                cond_probs_features_given_model[f"P[{key}|{index}]"] = probability

print("Conditional Probabilities of Features, given the model:")
print()
for key in sorted(cond_probs_features_given_model,key= lambda k: k.split("|")[1]):
    print(f"{key} = {cond_probs_features_given_model[key]}")
```

```
Conditional Probabilities of Features, given the model:

P[Female|KP281] = 0.5
P[Male|KP281] = 0.5
P[Partnered|KP281] = 0.6
P[Single|KP281] = 0.4
P[Income_Above_55k|KP281] = 0.14
P[Income_Below_55k|KP281] = 0.86
P[Fitness_Above_3|KP281] = 0.14
P[Fitness_Below_3|KP281] = 0.86
P[Female|KP481] = 0.48
P[Male|KP481] = 0.52
P[Partnered|KP481] = 0.6
P[Single|KP481] = 0.4
P[Income_Above_55k|KP481] = 0.2
P[Income_Below_55k|KP481] = 0.8
P[Fitness_Above_3|KP481] = 0.13
P[Fitness_Below_3|KP481] = 0.87
```

```
P[Female|KP781] = 0.18
P[Male|KP781] = 0.82
P[Partnered|KP781] = 0.57
P[Single|KP781] = 0.42
P[Income_Above_55k|KP781] = 0.78
P[Income_Below_55k|KP781] = 0.22
P[Fitness_Above_3|KP781] = 0.9
P[Fitness_Below_3|KP781] = 0.1
```

Customer Profile Insights:

1. **KP281**: 86 % of Customers rated themselves with Fitness Levels 3 or below; Also 86 % have an annual income 55000 USD or below
    - P [Fitness_Below_3 | KP281] = 0.86
    - P [Income_Below_55k | KP281] = 0.86
2. **KP481**: 87 % of Customers rated themselves with Fitness Levels 3 or below; Also 80 % have an annual income 55000 USD or below
    - P [Income_Below_55k | KP481] = 0.8
    - P [Fitness_Below_3 | KP481] = 0.87
3. **KP781**: 90 % of Customers rated themselves with Fitness Levels above 3; 82 % are male and 78 % have an annual income above 55000 USD
    - P [Income_Above_55k | KP781] = 0.78
    - P [Male | KP781] = 0.82
    - P [Fitness_Above_3 | KP781] = 0.9

Conditional Probabilities of Model, given the Feature:

In [62]:
```
cond_probs_model_given_feature = {}
for index in df2.index:
    if index != "All":
        for key in counts.keys():
            if key != "All":
                probability = np.round(df2.loc[index,key]/counts[key],2)
                cond_probs_model_given_feature[f"P[{index}|{key}]"] = probability
```

In [63]:
```
for key in sorted(cond_probs_model_given_feature,key= lambda k: k.split("|")[1],reverse=True):
    print(f"{key} = {cond_probs_model_given_feature[key]}")
```

```
P[KP281|Single] = 0.44
P[KP481|Single] = 0.33
P[KP781|Single] = 0.23
P[KP281|Partnered] = 0.45
P[KP481|Partnered] = 0.34
P[KP781|Partnered] = 0.21
P[KP281|Male] = 0.38
P[KP481|Male] = 0.3
P[KP781|Male] = 0.32
P[KP281|Income_Below_55k] = 0.55
P[KP481|Income_Below_55k] = 0.38
P[KP781|Income_Below_55k] = 0.07
P[KP281|Income_Above_55k] = 0.2
P[KP481|Income_Above_55k] = 0.22
P[KP781|Income_Above_55k] = 0.57
P[KP281|Fitness_Below_3] = 0.55
```

```
P[KP481|Fitness_Below_3] = 0.42
P[KP781|Fitness_Below_3] = 0.03
P[KP281|Fitness_Above_3] = 0.2
P[KP481|Fitness_Above_3] = 0.15
P[KP781|Fitness_Above_3] = 0.65
P[KP281|Female] = 0.53
P[KP481|Female] = 0.38
P[KP781|Female] = 0.09
```

Model Recommendations based on Customer Characteristics:

In [64]:
```
MaritalStatus=["Single","Partnered"]
Gender = ["Male","Female"]
Income = ["Income_Above_55k","Income_Below_55k"]
Fitness = ["Fitness_Above_3","Fitness_Below_3"]
```

In [66]:
```
# Using Naive Bayes approach, given the customer characteristics, this function returns recommended model
def get_model_given_multiple_features(*args):

    """
    Given one or more features, return the recommended treadmill model
    """
    if not args:
        return

    probabilities = {}
    for model in customers.keys():

        probability = 1
        if model != "All":
            probability *= marginal_probabilities[f"P[{model}]"]
            for feature in args:
                probability *= cond_probs_features_given_model[f"P[{feature}|{model}]"]
            probabilities[model] = np.round(probability,2)

    # Ignoring the denominator as it is same for all three models
    recommended_model = max(probabilities,key=lambda k:probabilities[k])
    return recommended_model, probabilities[recommended_model]
```

In [67]:
```
for m in MaritalStatus:
    for g in Gender:
        for i in Income:
            for f in Fitness:
                model, prob = get_model_given_multiple_features(m,g,i,f)
                if prob >= 0.05:
                    print(f"Features: [{m},{g},{i},{f}] :- Model: {model}, Probability: {prob}")
```

```
Features: [Single,Male,Income_Above_55k,Fitness_Above_3] :- Model: KP781, Probability: 0.05
Features: [Single,Male,Income_Below_55k,Fitness_Below_3] :- Model: KP281, Probability: 0.07
Features: [Single,Female,Income_Below_55k,Fitness_Below_3] :- Model: KP281, Probability: 0.07
Features: [Partnered,Male,Income_Above_55k,Fitness_Above_3] :- Model: KP781, Probability: 0.07
Features: [Partnered,Male,Income_Below_55k,Fitness_Below_3] :- Model: KP281, Probability: 0.1
Features: [Partnered,Female,Income_Below_55k,Fitness_Below_3] :- Model: KP281, Probability: 0.1
```

Insights:

1. Irrespective of Marital Status, Men with an income above 55000 dollars and Fitness levels above 3, we should recommend KP781
2. Irrespective of Marital Status, Men with an income below 55000 dollars and Fitness levels below 3, we should recommend KP481, KP281
3. As Marital Status didn't change the recommendation, dropping Marital Status in the further analysis

In [68]:

```
for g in Gender:
    for i in Income:
        for f in Fitness:
            model, prob = get_model_given_multiple_features(g,i,f)
            if prob > 0.1:
                print(f"Features: [{g},{i},{f}] :- Model: {model}, Probability: {prob}")
```

```
Features: [Male,Income_Above_55k,Fitness_Above_3] :- Model: KP781, Probability: 0.13
Features: [Male,Income_Below_55k,Fitness_Below_3] :- Model: KP281, Probability: 0.16
Features: [Female,Income_Below_55k,Fitness_Below_3] :- Model: KP281, Probability: 0.16
```

Insights:

1. Irrespective of Gender, For Customers with an income above 55000 dollars and Fitness levels above 3, recommended model is KP781
2. Irrespective of Gender, For Customers with an income below 55000 dollars and Fitness levels below 3, recommended models are KP281, KP481 as both seem to be equally likely
3. Dropping gender for further analysis

In [69]:

```
for i in Income:
    for f in Fitness:
        model, prob = get_model_given_multiple_features(i,f)
        if prob > 0.1:
            print(f"Features: [{i},{f}] :- Model: {model}, Probability: {prob}")
```

```
Features: [Income_Above_55k,Fitness_Above_3] :- Model: KP781, Probability: 0.15
Features: [Income_Below_55k,Fitness_Below_3] :- Model: KP281, Probability: 0.33
```

Insights:

1. For Customers with salary of 55000 dollars and more, and Fitness levels above 3, recommended model is KP781
2. For Customers with salary of below 55000 dollars, Fitness levels 3 or below, recommended model is KP281, KP781

Business Insights Summary:

KP281 & KP481 Target Customers:

1. Either Men/Women with Income below 55000 USD & Fitness Levels 3 or below

KP781 Target Customers:

1. Men with expected weekly miles of 150 or more
2. Fitness Levels above 3
3. Annual Income of 55000 dollars and more
4. Weekly Usage of 4 times or more

5. Education Levels 16 Years or more

Other Observations:

1. All 3 models are preferred across ages 20 - 50. KP781 is preferred most by Customers aged 30-35. KP481 is more preferred by Customers aged 35 or more. Distinctions between ages is not vivid enough for deeper analysis.
2. More Partnered Customers than Single Customers across all three models. Distinctions between Marital Status is not vivid enough for deeper analysis.

Recommendations:

1. 83 % of KP781 Customers are Men - recommend looking into why Women are not preferring this model
2. Customers seem comfortable spending upto 4 percent of their annual income on treadmills. To increase the sales of treadmills, consider By Now, Pay Later schemes. - This might encourage more Customers to prefer KP481, and KP781 models
3. Not all Customers use treadmill for running. Therefore, datapoints on type of workouts would provide more insights
4. Positive Correlation between features like Fitness levels, expected Miles, expected Usage is very high. Therefore, we do not have to collect data for all three features.
5. KP281, KP481 seem to be equally preferred considering Age,Gender,Income,Fitness Levels. Further data on model's features might help. Strongly looking into differentiating features for the two models to cater to different customer bases