Cardio Good Fitness Project

Objective

Explore the dataset to identify differences between the customers of each product. Also explore relationships between the different attributes of the customers.

Data Dictionary The data is about customers of the treadmill product(s) of a retail store called Cardio Good Fitness. It contains the following variables-

1. Product - The model no. of the treadmill
2. Age - Age of the customer in no of years
3. Gender - Gender of the customer
4. Education - Education of the customer in no. of years
5. Marital Status - Marital status of the customer
6. Usage - Avg. # times the customer wants to use the treadmill every week
7. Fitness - Self rated fitness score of the customer (5 - very fit, 1 - very unfit)
8. Income - Income of the customer
9. Miles- Miles that a customer expects to run

In [1]:
```
#import the necessary libraries
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
#read the csv file and store in dataframe
df = pd.read_csv('CardioGoodFitness.csv')
```

In [3]:
```
#look at a sample of data, first few rows
df.head()
```

Out[3]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
0	TM195	18	Male	14	Single	3	4	29562	112
1	TM195	19	Male	15	Single	2	3	31836	75
2	TM195	19	Female	14	Partnered	4	3	30699	66
3	TM195	19	Male	12	Single	3	3	32973	85
4	TM195	20	Male	13	Partnered	4	2	35247	47

In [4]:
```
#check columns
df.columns
```

Out[4]:
```
Index(['Product', 'Age', 'Gender', 'Education', 'MaritalStatus', 'Usage',
       'Fitness', 'Income', 'Miles'],
      dtype='object')
```

In [5]:
```
#check the shape of the data
df.shape
```

Out[5]:
```
(180, 9)
```

There are 180 rows and 9 columns in this data set

In [6]:
```
#check the datatypes to make sure the data is read in correctly
df.dtypes
```

Out[6]:
```
Product          object
Age               int64
Gender           object
Education         int64
MaritalStatus    object
Usage             int64
Fitness           int64
Income            int64
Miles             int64
dtype: object
```

Observations:

1. Product, Gender and MaritalStatus are object data type.
2. All the other variables are numerical and their python data types (int64) are ok.
3. It would be good to convert Product, Gender and MaritalStatus into category data type instead of object
4. Also to convert Fitness into category data type instead of int64
5. There are no float data types.

In [7]:
```
#changing the Gender and MaritalStatus data types to category instead of object
df.Gender = df.Gender.astype('category')
df.MaritalStatus = df.MaritalStatus.astype('category')
df.Product = df.Product.astype('category')
#changing the Fitness data type to category instead of int64
df.Fitness = df.Fitness.astype('category')
```

In [8]:
```
#checking for null/missing values
totalnull = df.isnull().sum().sort_values (ascending=False)
print(totalnull)
```

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

Observation:

There are no missing values in the data set.

Analyze the quantitative variables in the dataset

In [9]:
```
#check data, describe
df.describe()
```

Out[9]:

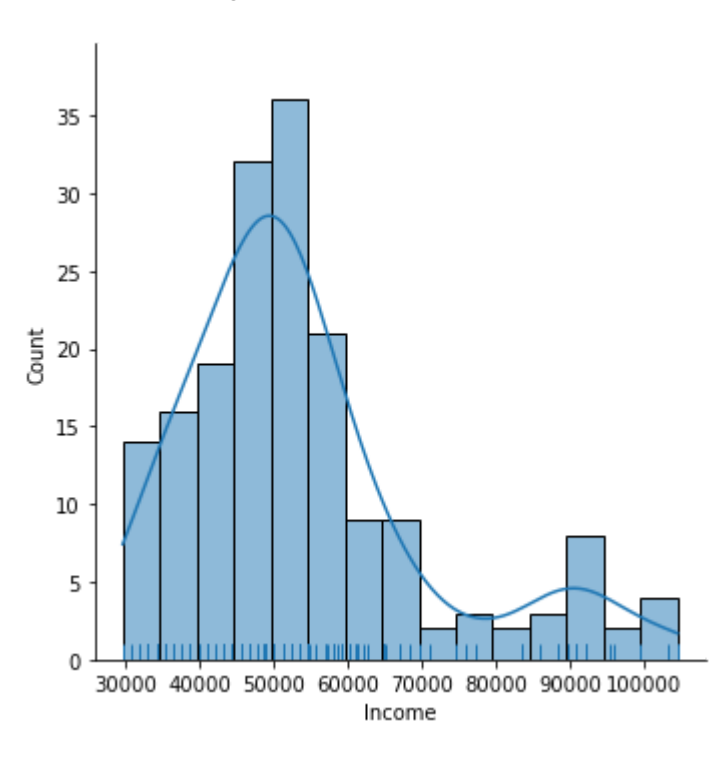
	Age	Education	Usage	Income	Miles
count	180.000000	180.000000	180.000000	180.000000	180.000000
mean	28.788889	15.572222	3.455556	53719.577778	103.194444
std	6.943498	1.617055	1.084797	16506.684226	51.863605
min	18.000000	12.000000	2.000000	29562.000000	21.000000
25%	24.000000	14.000000	3.000000	44058.750000	66.000000
50%	26.000000	16.000000	3.000000	50596.500000	94.000000
75%	33.000000	16.000000	4.000000	58668.000000	114.750000
max	50.000000	21.000000	7.000000	104581.000000	360.000000

Observations:

There are 180 total rows. The mean age is 28.7 and is slightly right skewed. The income is spread over a big range, ranging from 104,581 to 29,562. As expected, the standard deviation is also high for income. The miles is also spread over a big range, from 21 to 360. As expected, the standard deviation is also high for miles. The mean for miles is also right skewed.

In [10]:
```
#looking at Income distribution
sns.displot(df['Income'], kde=True, rug=True)
```

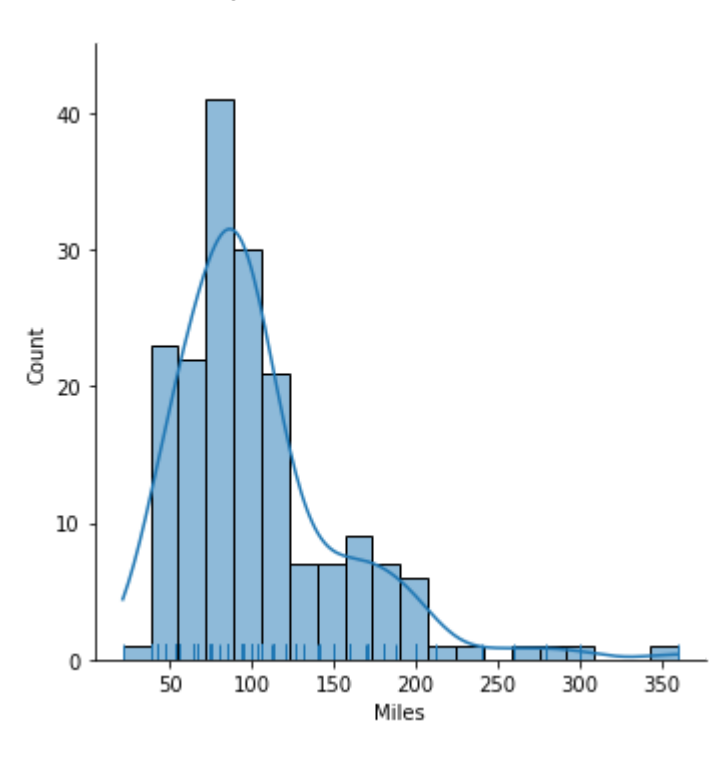
Out[10]:
```
<seaborn.axisgrid.FacetGrid at 0x2411b5c1fa0>
```



Observation: The income is spread over a big range, ranging from 29,562 to 104,581.

In [11]:
```
#looking at Miles distribution
sns.displot(df['Miles'], kde=True, rug=True)
```

Out[11]:
```
<seaborn.axisgrid.FacetGrid at 0x2411be23940>
```

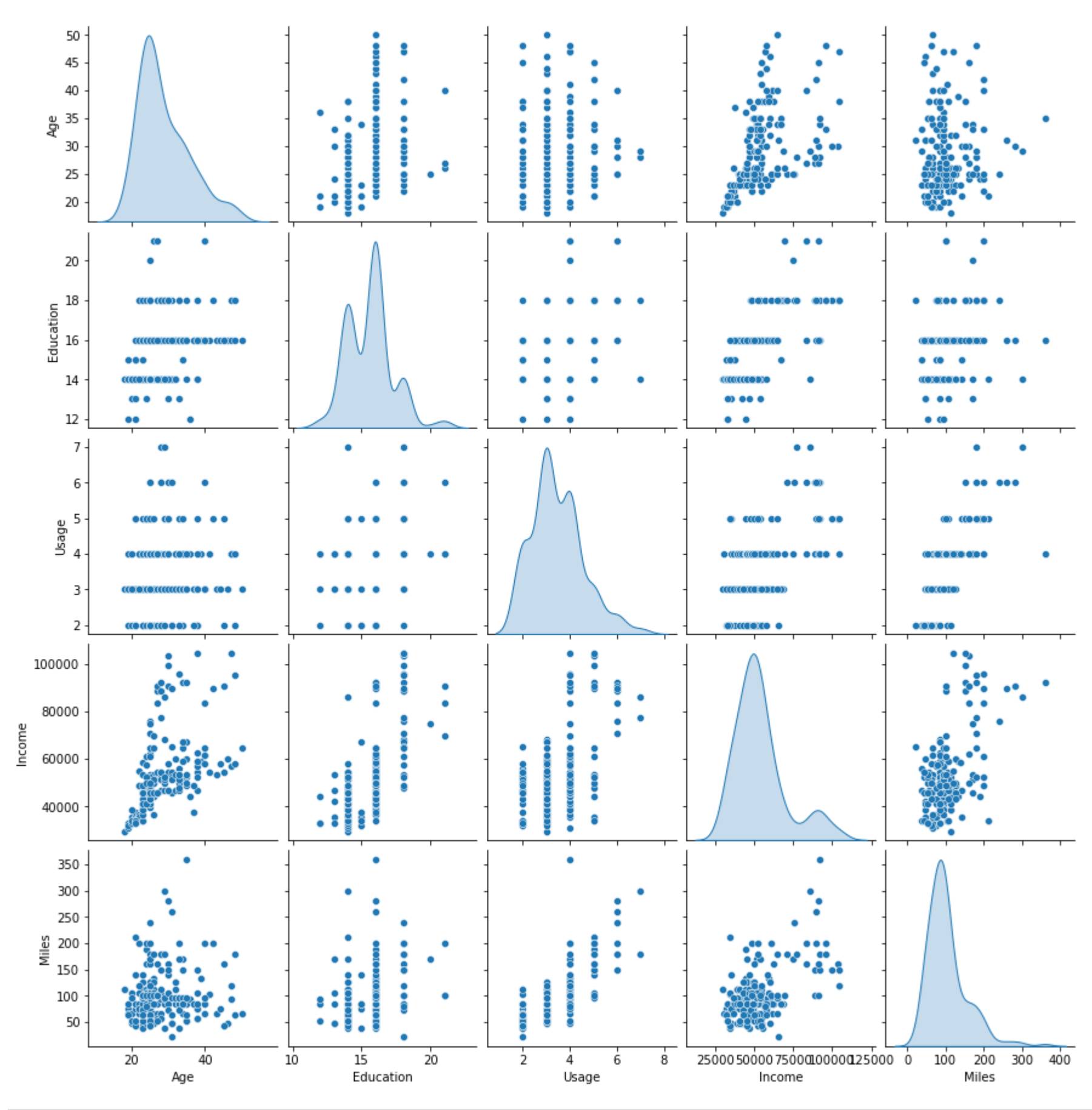


Observation: The miles is also spread over a big range, from 21 to 360. As expected, the standard deviation is also high for miles. The mean for miles is also right skewed.

Looking at the relationship between numerical variables using pair plots, correlation plots and heat map

In [12]:
```
sns.pairplot(df, diag_kind="kde")
```

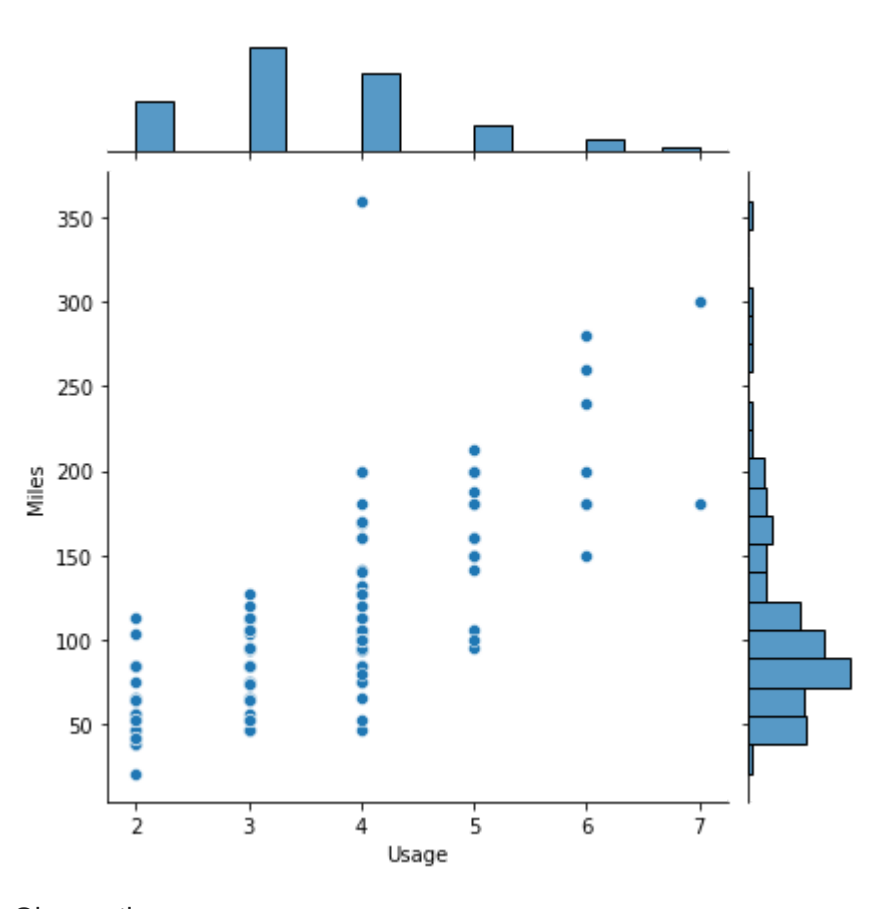
Out[12]:
```
<seaborn.axisgrid.PairGrid at 0x2411be23fa0>
```



In [13]:
```
correlation = df.corr()
plt.figure(figsize=(15, 7))
sns.heatmap(correlation, annot=True, vmin=-1, vmax=1, fmt=".2f", cmap="Spectral")
plt.show()
```

In [14]:
```
sns.jointplot(data=df, x="Usage", y="Miles")
```

Out[14]:
```
<seaborn.axisgrid.JointGrid at 0x2411cc3b700>
```



Observations:

There is a high correlation between miles and usage. There is a correlation between education and income There is some correlation between age and income. Interestingly, there is no correlation between age and usage or miles.

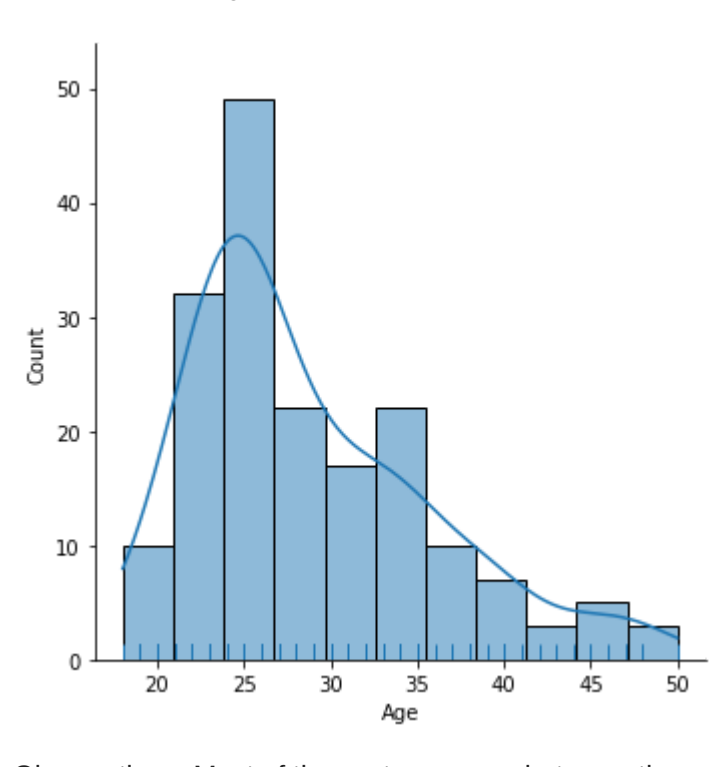
Explore the categorical features -

In [15]:
```
df.groupby('Age').size()
```

Out[15]:
```
Age
18     1
19     4
20     5
21     7
22     7
23    18
24    12
25    25
26    12
27     7
28     9
29     6
30     7
31     6
32     4
33     8
34     6
35     8
36     1
37     2
38     7
39     1
40     5
41     1
42     1
43     1
44     1
45     2
46     1
47     2
48     2
50     1
dtype: int64
```

In [16]:
```
sns.displot(df['Age'], kde=True, rug=True)
```

Out[16]:
```
<seaborn.axisgrid.FacetGrid at 0x2411cfd9880>
```



Observations: Most of the customers are between the ages 23 and 26 years. The age ranges from 18 to 50 years. This distribution is slightly right skewed.

In [17]:
```
df.groupby('Gender').size()
```

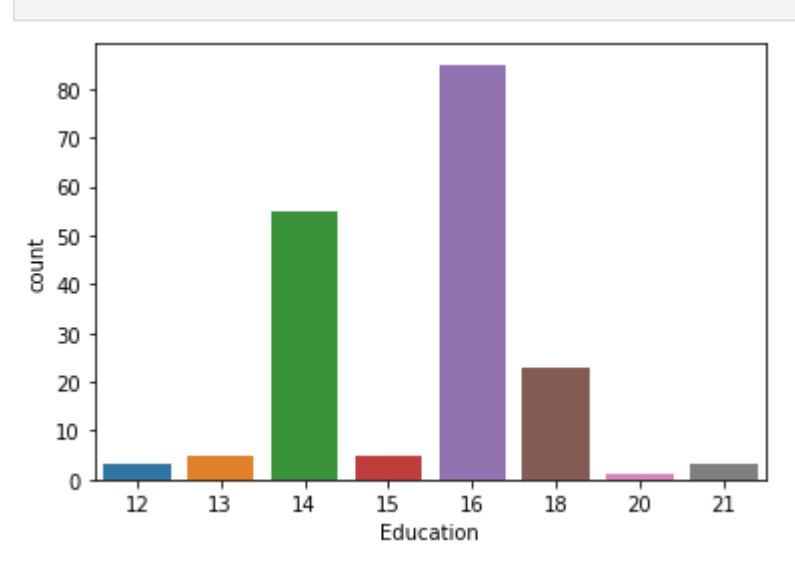
Out[17]:
```
Gender
Female     76
Male      104
dtype: int64
```

Observations: There are more male customers than female.

In [18]:
```
df.groupby('Education').size()
```

Out[18]:
```
Education
12     3
13     5
14    55
15     5
16    85
18    23
20     1
21     3
dtype: int64
```

In [19]:
```
sns.countplot(x='Education', data =df)
plt.show()
```



Observations: Most of the customers have 16 years of education. Then 14 years of education and then 18 years of education. The number of years of education range from 12 to 21 years.

In [20]:
```
df.groupby('MaritalStatus').size()
```

Out[20]:
```
MaritalStatus
Partnered    107
Single        73
dtype: int64
```

Observation: There are more customers who are partnered than single.

In [21]:
```
df.groupby('Fitness').size()
```

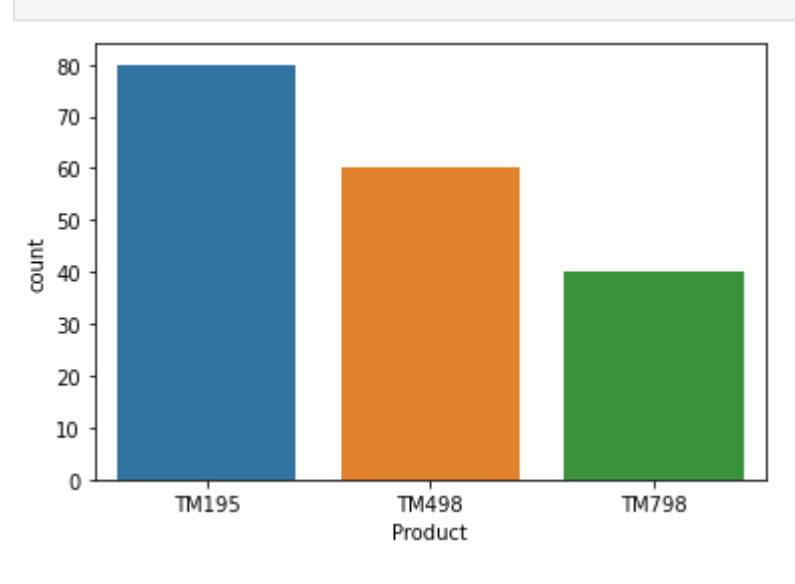
Out[21]:
```
Fitness
1     2
2    26
3    97
4    24
5    31
dtype: int64
```

Observation: Most of the customers rated their fitness in the middle at 3.

In [22]:
```
df.groupby('Product').size()
```

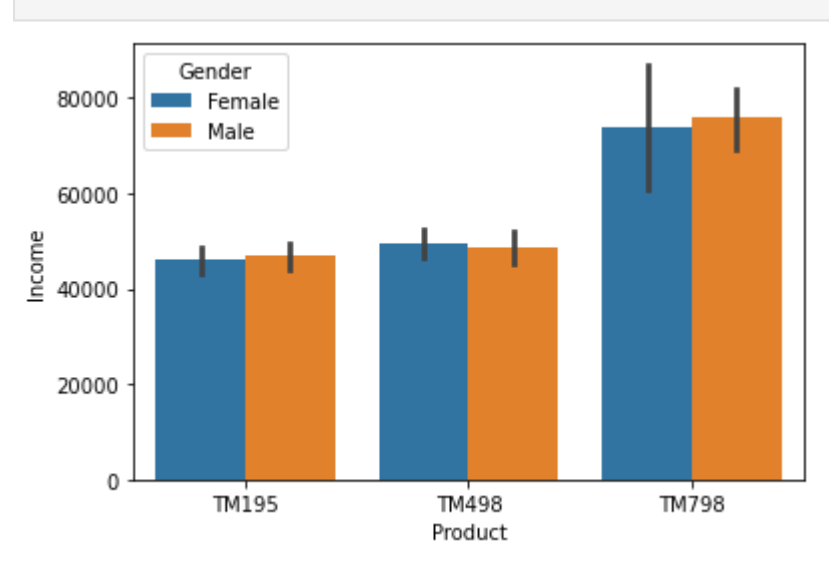
Out[22]:
```
Product
TM195    80
TM498    60
TM798    40
dtype: int64
```

In [23]:
```
sns.countplot(x='Product', data =df)
plt.show()
```



Observation: The most common product is TM195, then TM498 and finally TM798

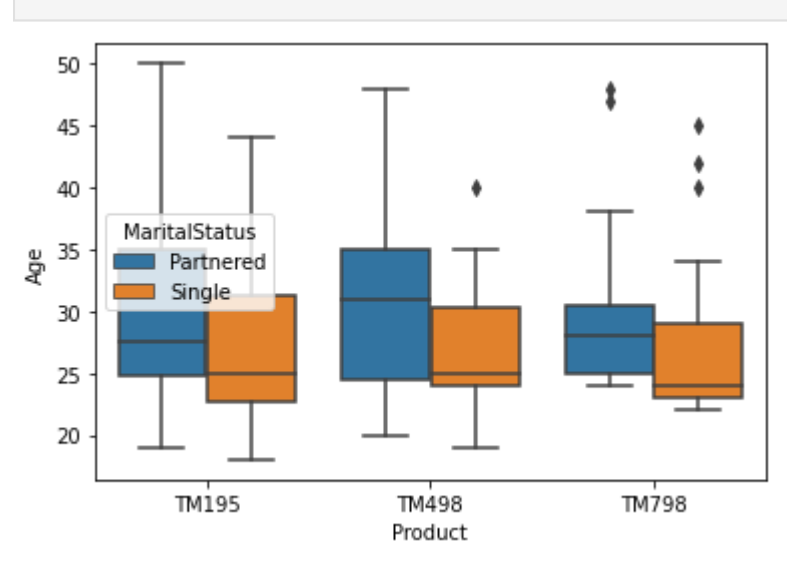
In [24]:
```
sns.barplot(data=df, x='Product', y='Income',hue='Gender')
plt.show()
```



Observation:

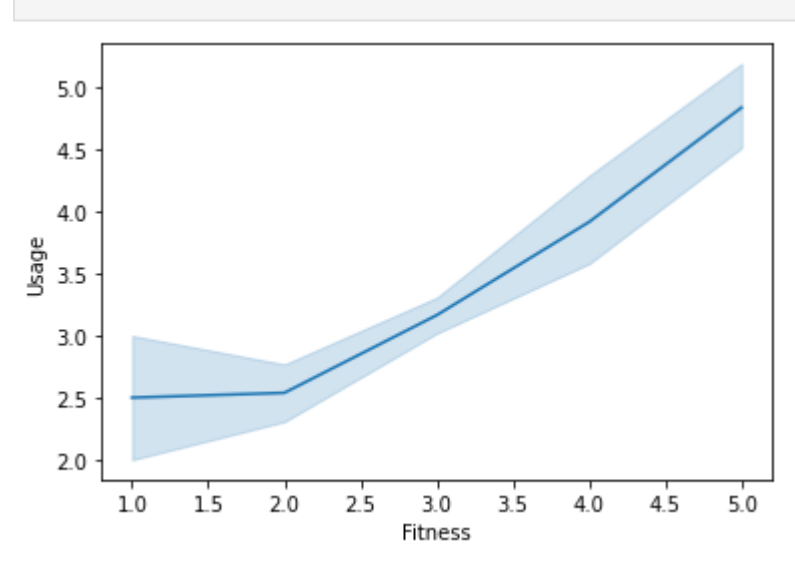
1. Customers with a higher income (>50,000) bought TM798
2. Customers with an income of less than 50,000 bought TM195 or TM498
3. The type of treadmill bought does not differ significantly between gender.

In [25]:
```
sns.boxplot(data=df, x='Product', y='Age', hue='MaritalStatus')
plt.show()
```



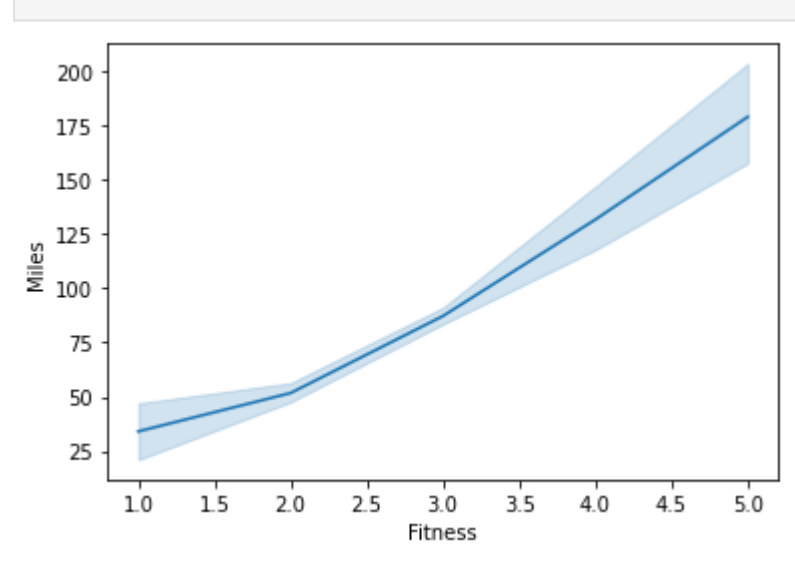
Observation: Overall more partnered people buy treadmills than single. Product TM798 was bought mainly by younger people.

In [26]:
```
sns.lineplot(data=df, x='Fitness', y='Usage')
plt.show()
```



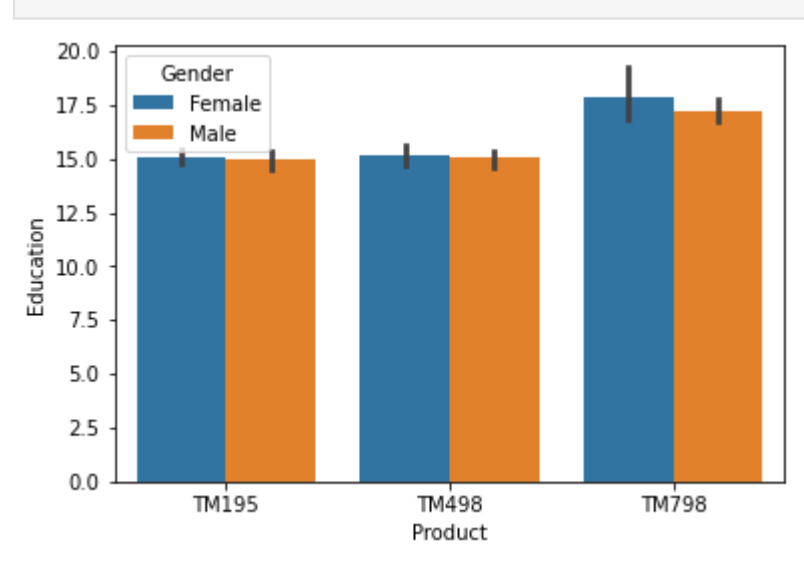
Obsrvation: There is a positive realtionship between fitness and usage. The higher the self rated fitness score, the higher the usage per week.

In [27]:
```
sns.lineplot(data=df, x='Fitness', y='Miles')
plt.show()
```



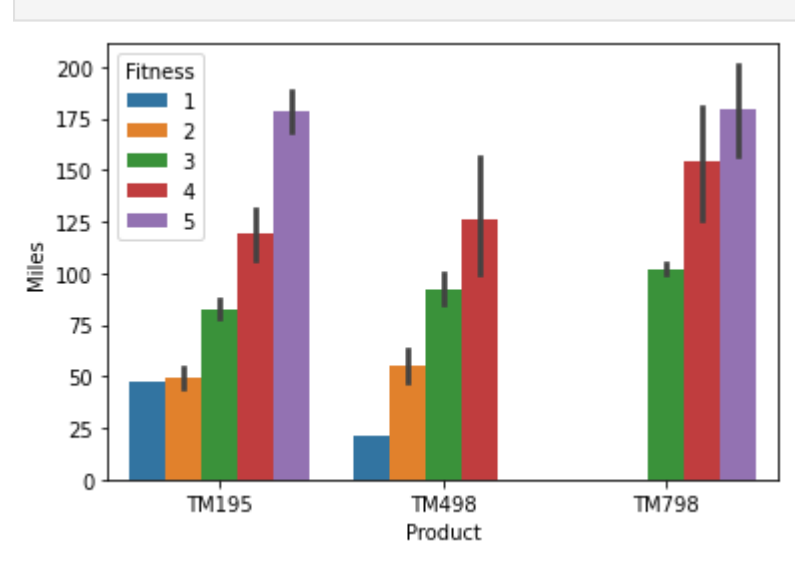
Observation: There is a positive realtionship between fitness and miles. The higher the self rated fitness score, the higher the number of miles the customer expects to run.

In [28]:
```
sns.barplot(data=df, x='Product', y='Education',hue='Gender')
plt.show()
```



Observation: Customers with higher education (>15 years) bought product TM798. There was not much difference between Gender.

In [29]:
```
sns.barplot(data=df, x='Product', y='Miles',hue='Fitness')
plt.show()
```



Observation: Customers who bought product TM498 did not rate their fitness as 5. There were customers who bought TM798 and TM195 and also rated their fitness at 5. Customers who bought TM798 had high to medium (5 to 3) fitness rating.

Conclusion:

1. The most common product is TM195, then TM498 and finally TM798
2. Customers with a higher income (>50,000) bought TM798
3. Customers with an income of less than 50,000 bought TM195 or TM498
4. The type of treadmill bought does not differ significantly between gender.
5. Overall more partnered people buy treadmills than single. Product TM798 was bought mainly by younger people.
6. There is a positive realtionship between fitness and usage. The higher the self rated fitness score, the higher the usage per week.
7. There is a positive realtionship between fitness and miles. The higher the self rated fitness score, the higher the number of miles the customer expects to run.
8. Customers with higher education (>15 years) bought product TM798. There was not much difference between Gender.
9. Customers who bought product TM498 did not rate their fitness as 5. There were customers who bought TM798 and TM195 and also rated their fitness at 5. Customers who bought TM798 had high to medium (5 to 3) fitness rating.

In [ ]: