In [1]:
```
# Importing all neccessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
# Importing Dataset
```

In [3]:
```
df = np.loadtxt(r'C:\Users\Lenovo\Desktop\Scaler\Case Studies\aerofit_treadmill.txt'
```

In [4]:
```
df
```

Out[4]:
```
array(['Product,Age,Gender,Education,MaritalStatus,Usage,Fitness,Income,Miles',
       'KP281,18,Male,14,Single,3,4,29562,112',
       'KP281,19,Male,15,Single,2,3,31836,75',
       'KP281,19,Female,14,Partnered,4,3,30699,66',
       'KP281,19,Male,12,Single,3,3,32973,85',
       'KP281,20,Male,13,Partnered,4,2,35247,47',
       'KP281,20,Female,14,Partnered,3,3,32973,66',
       'KP281,21,Female,14,Partnered,3,3,35247,75',
       'KP281,21,Male,13,Single,3,3,32973,85',
       'KP281,21,Male,15,Single,5,4,35247,141',
       'KP281,21,Female,15,Partnered,2,3,37521,85',
       'KP281,22,Male,14,Single,3,3,36384,85',
       'KP281,22,Female,14,Partnered,3,2,35247,66',
       'KP281,22,Female,16,Single,4,3,36384,75',
       'KP281,22,Female,14,Single,3,3,35247,75',
       'KP281,23,Male,16,Partnered,3,1,38658,47',
       'KP281,23,Male,16,Partnered,3,3,40932,75',
       'KP281,23,Female,14,Single,2,3,34110,103',
       'KP281,23,Male,16,Partnered,4,3,39795,94',
       'KP281,23,Female,16,Single,4,3,38658,113',
       'KP281,23,Female,15,Partnered,2,2,34110,38',
       'KP281,23,Male,14,Single,4,3,38658,113',
       'KP281,23,Male,16,Single,4,3,40932,94',
       'KP281,24,Female,16,Single,4,3,42069,94',
       'KP281,24,Female,16,Partnered,5,5,44343,188',
       'KP281,24,Male,14,Single,2,3,45480,113',
       'KP281,24,Male,13,Partnered,3,2,42069,47',
       'KP281,24,Female,16,Single,4,3,46617,75',
       'KP281,25,Female,14,Partnered,3,3,48891,75',
       'KP281,25,Male,14,Partnered,2,3,45480,56',
       'KP281,25,Female,14,Partnered,2,2,53439,47',
       'KP281,25,Female,14,Partnered,3,3,39795,85',
       'KP281,25,Male,16,Single,3,4,40932,113',
       'KP281,25,Female,16,Partnered,2,2,40932,47',
       'KP281,25,Male,16,Single,3,3,43206,85',
       'KP281,26,Female,14,Partnered,3,4,44343,113',
       'KP281,26,Female,16,Partnered,4,3,52302,113',
       'KP281,26,Male,16,Partnered,2,2,53439,47',
       'KP281,26,Male,16,Partnered,3,3,51165,85',
       'KP281,26,Female,16,Single,3,3,36384,66',
       'KP281,26,Male,16,Partnered,4,4,44343,132',
       'KP281,26,Male,16,Single,3,3,50028,85',
       'KP281,27,Female,14,Partnered,3,2,45480,66',
       'KP281,27,Male,16,Single,4,3,54576,85',
       'KP281,27,Female,14,Partnered,2,3,45480,56',
       'KP281,28,Female,14,Partnered,2,3,46617,56',
       'KP281,28,Female,16,Partnered,2,3,52302,66',
       'KP281,28,Male,14,Single,3,3,52302,103',
       'KP281,28,Female,14,Partnered,3,3,54576,94',
```

```
'KP281,28,Male,14,Single,4,3,54576,113',
'KP281,28,Female,16,Partnered,3,3,51165,56',
'KP281,29,Male,18,Partnered,3,3,68220,85',
'KP281,29,Female,14,Partnered,2,2,46617,38',
'KP281,29,Female,16,Partnered,4,3,50028,94',
'KP281,30,Male,14,Partnered,4,4,46617,141',
'KP281,30,Male,14,Single,3,3,54576,85',
'KP281,31,Male,14,Partnered,2,2,54576,47',
'KP281,31,Female,14,Single,2,2,45480,47',
'KP281,32,Female,14,Single,3,4,46617,113',
'KP281,32,Male,14,Partnered,4,3,52302,85',
'KP281,33,Female,16,Single,2,2,55713,38',
'KP281,33,Female,16,Partnered,3,3,46617,85',
'KP281,34,Male,16,Single,4,5,51165,169',
'KP281,34,Female,16,Single,2,2,52302,66',
'KP281,35,Male,16,Partnered,4,3,48891,85',
'KP281,35,Female,16,Partnered,3,3,60261,94',
'KP281,35,Female,18,Single,3,3,67083,85',
'KP281,36,Male,12,Single,4,3,44343,94',
'KP281,37,Female,16,Partnered,3,3,37521,85',
'KP281,38,Male,16,Partnered,3,3,46617,75',
'KP281,38,Female,14,Partnered,2,3,54576,56',
'KP281,38,Male,14,Single,2,3,52302,56',
'KP281,38,Male,16,Partnered,3,3,56850,75',
'KP281,39,Male,16,Partnered,4,4,59124,132',
'KP281,40,Male,16,Partnered,3,3,61398,66',
'KP281,41,Male,16,Partnered,4,3,54576,103',
'KP281,43,Male,16,Partnered,3,3,53439,66',
'KP281,44,Female,16,Single,3,4,57987,75',
'KP281,46,Female,16,Partnered,3,2,60261,47',
'KP281,47,Male,16,Partnered,4,3,56850,94',
'KP281,50,Female,16,Partnered,3,3,64809,66',
'KP481,19,Male,14,Single,3,3,31836,64',
'KP481,20,Male,14,Single,2,3,32973,53',
'KP481,20,Female,14,Partnered,3,3,34110,106',
'KP481,20,Male,14,Single,3,3,38658,95',
'KP481,21,Female,14,Partnered,5,4,34110,212',
'KP481,21,Male,16,Partnered,2,2,34110,42',
'KP481,21,Male,12,Partnered,2,2,32973,53',
'KP481,23,Male,14,Partnered,3,3,36384,95',
'KP481,23,Male,14,Partnered,3,3,38658,85',
'KP481,23,Female,16,Single,3,3,45480,95',
'KP481,23,Male,16,Partnered,4,3,45480,127',
'KP481,23,Female,16,Partnered,3,2,43206,74',
'KP481,23,Female,14,Single,3,2,40932,53',
'KP481,23,Male,16,Partnered,3,3,45480,64',
'KP481,24,Female,14,Single,3,2,40932,85',
'KP481,24,Male,14,Single,3,4,48891,106',
'KP481,24,Female,16,Single,3,3,50028,106',
'KP481,25,Female,14,Partnered,2,3,45480,85',
'KP481,25,Female,14,Single,3,4,43206,127',
'KP481,25,Male,16,Partnered,2,2,52302,42',
'KP481,25,Female,14,Partnered,5,3,47754,106',
'KP481,25,Male,14,Single,3,3,45480,95',
'KP481,25,Female,14,Single,2,3,43206,64',
'KP481,25,Male,14,Partnered,4,3,45480,170',
'KP481,25,Male,14,Partnered,3,4,43206,106',
'KP481,25,Male,16,Partnered,2,3,50028,53',
'KP481,25,Female,14,Single,2,2,45480,42',
'KP481,25,Male,14,Single,4,3,48891,127',
'KP481,26,Female,16,Partnered,4,3,45480,85',
'KP481,26,Female,16,Single,4,4,50028,127',
'KP481,26,Male,16,Single,4,3,51165,106',
'KP481,27,Male,14,Single,4,2,45480,53',
'KP481,29,Female,14,Partnered,3,3,51165,95',
'KP481,30,Female,14,Single,3,3,57987,74',
'KP481,30,Female,13,Single,4,3,46617,106',
'KP481,31,Male,16,Partnered,3,3,52302,95',
'KP481,31,Female,16,Partnered,2,3,51165,64',
```

```
       'KP481,31,Female,18,Single,2,1,65220,21',
       'KP481,32,Male,16,Single,4,3,60261,127',
       'KP481,32,Male,16,Partnered,3,3,53439,95',
       'KP481,33,Male,13,Partnered,4,4,53439,170',
       'KP481,33,Female,16,Partnered,2,3,50028,85',
       'KP481,33,Male,16,Partnered,3,3,51165,95',
       'KP481,33,Female,16,Partnered,5,3,53439,95',
       'KP481,33,Female,18,Single,3,4,47754,74',
       'KP481,34,Female,16,Partnered,4,3,64809,95',
       'KP481,34,Male,16,Partnered,3,4,59124,85',
       'KP481,34,Male,15,Single,3,3,67083,85',
       'KP481,35,Female,14,Partnered,3,2,52302,53',
       'KP481,35,Male,16,Partnered,3,2,53439,53',
       'KP481,35,Female,16,Single,3,2,50028,64',
       'KP481,35,Male,16,Partnered,3,3,53439,95',
       'KP481,37,Female,16,Partnered,2,3,48891,85',
       'KP481,38,Female,16,Partnered,4,3,62535,85',
       'KP481,38,Male,16,Partnered,3,3,59124,106',
       'KP481,40,Female,16,Partnered,3,3,61398,85',
       'KP481,40,Female,16,Single,3,3,57987,85',
       'KP481,40,Male,16,Partnered,3,3,64809,95',
       'KP481,45,Male,16,Partnered,2,2,54576,42',
       'KP481,48,Male,16,Partnered,2,3,57987,64',
       'KP781,22,Male,14,Single,4,3,48658,106',
       'KP781,22,Male,16,Single,3,5,54781,120',
       'KP781,22,Male,18,Single,4,5,48556,200',
       'KP781,23,Male,16,Single,4,5,58516,140',
       'KP781,23,Female,18,Single,5,4,53536,100',
       'KP781,23,Male,16,Single,4,5,48556,100',
       'KP781,24,Male,16,Single,4,5,61006,100',
       'KP781,24,Male,18,Partnered,4,5,57271,80',
       'KP781,24,Female,16,Single,5,5,52291,200',
       'KP781,24,Male,16,Single,5,5,49801,160',
       'KP781,25,Male,16,Partnered,4,5,49801,120',
       'KP781,25,Male,16,Partnered,4,4,62251,160',
       'KP781,25,Female,18,Partnered,5,5,61006,200',
       'KP781,25,Male,18,Partnered,4,3,64741,100',
       'KP781,25,Male,18,Partnered,6,4,70966,180',
       'KP781,25,Male,18,Partnered,6,5,75946,240',
       'KP781,25,Male,20,Partnered,4,5,74701,170',
       'KP781,26,Female,21,Single,4,3,69721,100',
       'KP781,26,Male,16,Partnered,5,4,64741,180',
       'KP781,27,Male,16,Partnered,4,5,83416,160',
       'KP781,27,Male,18,Single,4,3,88396,100',
       'KP781,27,Male,21,Partnered,4,4,90886,100',
       'KP781,28,Female,18,Partnered,6,5,92131,180',
       'KP781,28,Male,18,Partnered,7,5,77191,180',
       'KP781,28,Male,18,Single,6,5,88396,150',
       'KP781,29,Male,18,Single,5,5,52290,180',
       'KP781,29,Male,14,Partnered,7,5,85906,300',
       'KP781,30,Female,16,Partnered,6,5,90886,280',
       'KP781,30,Male,18,Partnered,5,4,103336,160',
       'KP781,30,Male,18,Partnered,5,5,99601,150',
       'KP781,31,Male,16,Partnered,6,5,89641,260',
       'KP781,33,Female,18,Partnered,4,5,95866,200',
       'KP781,34,Male,16,Single,5,5,92131,150',
       'KP781,35,Male,16,Partnered,4,5,92131,360',
       'KP781,38,Male,18,Partnered,5,5,104581,150',
       'KP781,40,Male,21,Single,6,5,83416,200',
       'KP781,42,Male,18,Single,5,4,89641,200',
       'KP781,45,Male,16,Single,5,5,90886,160',
       'KP781,47,Male,18,Partnered,4,5,104581,120',
       'KP781,48,Male,18,Partnered,4,5,95508,180'], dtype='<U69')
```

In [5]:

```
# Dimension of data
df.ndim
```

Out[5]: 1

In [6]:
```
# shape of data
```

In [7]:
```
df.shape
```

Out[7]: (181,)

In [8]:
```
# we will convert the text data set to pandas data frame
```

In [9]:
```
df1 = pd.read_csv(r'C:\Users\Lenovo\Desktop\Scaler\Case Studies\aerofit_treadmill.tx
```

In [10]:
```
df1
```

Out[10]:

	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
...	...	...	...	...	...	...	...	...	...
175	KP781	40	Male	21	Single	6	5	83416	200
176	KP781	42	Male	18	Single	5	4	89641	200
177	KP781	45	Male	16	Single	5	5	90886	160
178	KP781	47	Male	18	Partnered	4	5	104581	120
179	KP781	48	Male	18	Partnered	4	5	95508	180

180 rows × 9 columns

In [11]:
```
df1.head()
```

Out[11]:

	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

In [12]:
```
df1.tail()
```

Out[12]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
175	KP781	40	Male	21	Single	6	5	83416	200
176	KP781	42	Male	18	Single	5	4	89641	200
177	KP781	45	Male	16	Single	5	5	90886	160
178	KP781	47	Male	18	Partnered	4	5	104581	120
179	KP781	48	Male	18	Partnered	4	5	95508	180

In [13]:
```
# 180 rows and 9 columns
df1.shape
```

Out[13]:
```
(180, 9)
```

In [14]:
```
# length of data
len(df)
```

Out[14]:
```
181
```

In [15]:
```
# checking datatypes
df1.dtypes
```

Out[15]:
```
Product          object
Age               int64
Gender           object
Education         int64
MaritalStatus    object
Usage             int64
Fitness           int64
Income            int64
Miles             int64
dtype: object
```

In [16]:
```
# number of unique values in our data
df1.nunique()
```

Out[16]:
```
Product           3
Age              32
Gender            2
Education         8
MaritalStatus     2
Usage             6
Fitness           5
Income           62
Miles            37
dtype: int64
```

In [17]:
```
# other way to find unique values in our data
for i in df1.columns:
    print(i, ":", df1[i].nunique())
```

```
Product : 3
Age : 32
Gender : 2
Education : 8
MaritalStatus : 2
Usage : 6
```

```
Fitness : 5
Income : 62
Miles : 37
```

In [18]:
```
# checking null values in every columns in dataset
df1.isnull().sum()
```

Out[18]:
```
Product          0
Age              0
Gender           0
Education        0
MaritalStatus    0
Usage            0
Fitness          0
Income           0
Miles            0
dtype: int64
```

1.There are no null values in our data

In [19]:
```
#Checking null values in every columns in data % wise
round((df1.isnull().sum()/len(df1)*100),2)
```

Out[19]:
```
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
```

In [20]:
```
# statistical analysis of data
df1.describe()
```

Out[20]:

	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

In [21]:
```
df1.describe(include='all') # including all data types
```

Out[21]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Inco
count	180	180.000000	180	180.000000	180	180.000000	180.000000	180.000(
unique	3	NaN	2	NaN	2	NaN	NaN	N

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Inco
top	KP281	NaN	Male	NaN	Partnered	NaN	NaN	N
freq	80	NaN	104	NaN	107	NaN	NaN	N
mean	NaN	28.788889	NaN	15.572222	NaN	3.455556	3.311111	53719.577
std	NaN	6.943498	NaN	1.617055	NaN	1.084797	0.958869	16506.684
min	NaN	18.000000	NaN	12.000000	NaN	2.000000	1.000000	29562.000
25%	NaN	24.000000	NaN	14.000000	NaN	3.000000	3.000000	44058.750
50%	NaN	26.000000	NaN	16.000000	NaN	3.000000	3.000000	50596.500
75%	NaN	33.000000	NaN	16.000000	NaN	4.000000	4.000000	58668.000
max	NaN	50.000000	NaN	21.000000	NaN	7.000000	5.000000	104581.000

Detection of Outliers

BoxPlot Explanation

1.middle line = median value

2.lower to middle line = 25% quartile (Q1)

3.upper to middle line = 75% (Q3)

4.above dash bar = max

5.below dash bar = min

6.dot is outlier ----1.5*IQR

IQR (Interquartile Range) = Q3 - Q1

In [22]:
```
df2 = df1.select_dtypes(exclude=['object'])
```

In [23]:
```
df2
```

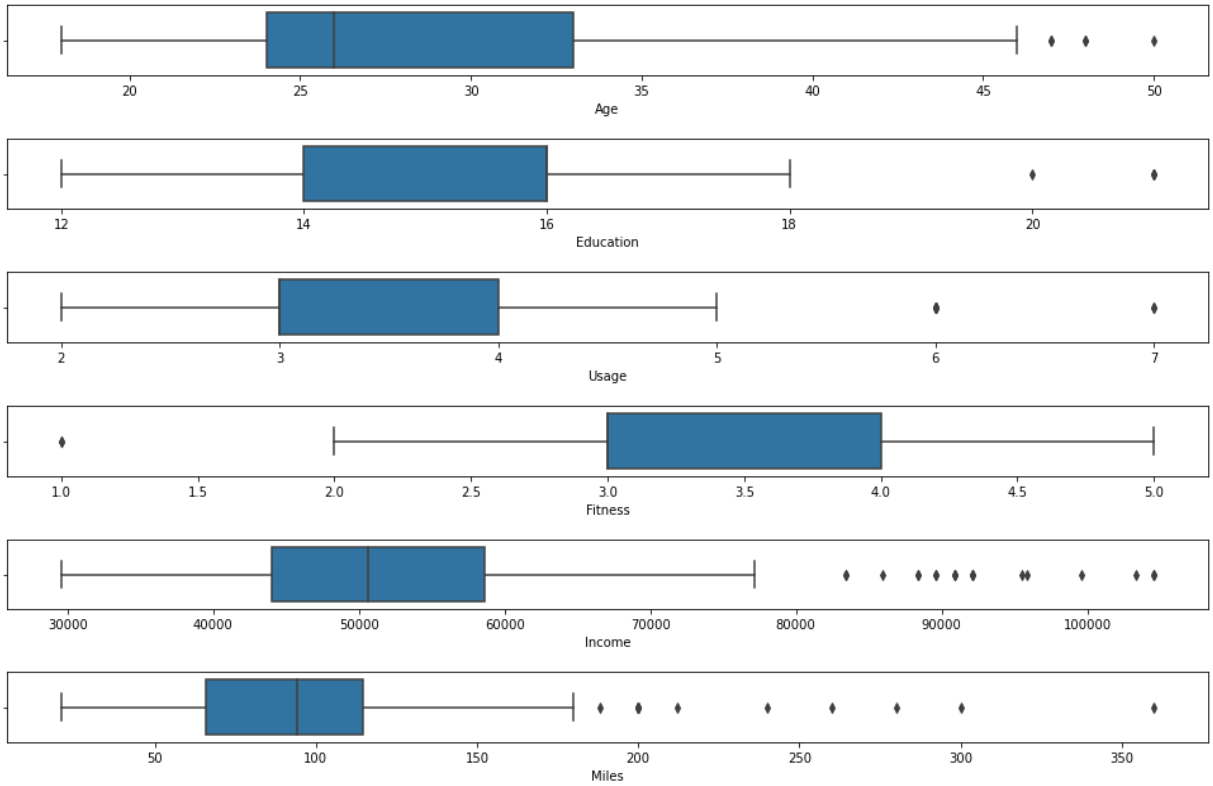
Out[23]:

	Age	Education	Usage	Fitness	Income	Miles
0	18	14	3	4	29562	112
1	19	15	2	3	31836	75
2	19	14	4	3	30699	66
3	19	12	3	3	32973	85
4	20	13	4	2	35247	47
...	...	...	...	...	...	...
175	40	21	6	5	83416	200

	Age	Education	Usage	Fitness	Income	Miles
176	42	18	5	4	89641	200
177	45	16	5	5	90886	160
178	47	18	4	5	104581	120
179	48	18	4	5	95508	180

180 rows × 6 columns

In [24]:
```
for column in df2:
    plt.figure(figsize=(17,1))
    sns.boxplot(data = df2, x=column)
```



The analysis of outliers as per box plot

1. Age : There outliers are only on upper bound and the values above 46 are outliers marked by dot

2. Education: There are outliers on upper bound and the values above 18 are outliers marked by dot

3. Usage: There are outliers on upper bound and the values above 5 are outliers marked by dot

4. Fitness: There are outliers on both upper and lower bound and -: values below 2 are outliers and values above 5 are outliers

5. Income: There are outliers on upper bound and values above 170 are outliers

Detection outliars using Statistical technique : IQR

Upper bound : Q3 + 1.5*IQR

Lower bound : Q1 - 1.5*IQR

In [25]:
```
for i in df2:
    Q1 = df2[i].quantile(0.25)
    Q3 = df2[i].quantile(0.75)
    IQR = Q3-Q1
    l1 = Q1-1.5*IQR
    u1 = Q3 + 1.5*IQR
    lower = df2[(df2[i]<(Q1-1.5*IQR))]
    upper = df2[(df2[i]>(Q3+1.5*IQR))]
    print("******",i,"******")
    print("First Quartile  :",Q1)
    print("Third Quartile  :",Q3)
    print("IQR range :",IQR)
    print("The outliers are values above : ",u1)
    print("The outliers are values below: ",l1)
    print("Outlier upper and lower bound are as : ")
    print("Total Values in upper bound are : ",upper[i].value_counts().sum())
    print("Total Values in lower bound are : ",lower[i].value_counts().sum())
    print()
```

```
****** Age ******
First Quartile  : 24.0
Third Quartile  : 33.0
IQR range : 9.0
The outliers are values above :  46.5
The outliers are values below:  10.5
Outlier upper and lower bound are as :
Total Values in upper bound are :  5
Total Values in lower bound are :  0

****** Education ******
First Quartile  : 14.0
Third Quartile  : 16.0
IQR range : 2.0
The outliers are values above :  19.0
The outliers are values below:  11.0
Outlier upper and lower bound are as :
Total Values in upper bound are :  4
Total Values in lower bound are :  0

****** Usage ******
First Quartile  : 3.0
Third Quartile  : 4.0
IQR range : 1.0
The outliers are values above :  5.5
The outliers are values below:  1.5
Outlier upper and lower bound are as :
Total Values in upper bound are :  9
Total Values in lower bound are :  0

****** Fitness ******
```

```
First Quartile  : 3.0
Third Quartile  : 4.0
IQR range : 1.0
The outliers are values above :  5.5
The outliers are values below:  1.5
Outlier upper and lower bound are as :
Total Values in upper bound are :  0
Total Values in lower bound are :  2

****** Income ******
First Quartile  : 44058.75
Third Quartile  : 58668.0
IQR range : 14609.25
The outliers are values above :  80581.875
The outliers are values below:  22144.875
Outlier upper and lower bound are as :
Total Values in upper bound are :  19
Total Values in lower bound are :  0

****** Miles ******
First Quartile  : 66.0
Third Quartile  : 114.75
IQR range : 48.75
The outliers are values above :  187.875
The outliers are values below:  -7.125
Outlier upper and lower bound are as :
Total Values in upper bound are :  13
Total Values in lower bound are :  0
```

In [26]:
```
# Checking columns
df1.columns
```

Out[26]:
```
Index(['Product', 'Age', 'Gender', 'Education', 'MaritalStatus', 'Usage',
       'Fitness', 'Income', 'Miles'],
      dtype='object')
```

In [27]:
```
df1['MaritalStatus'].unique()
```

Out[27]:
```
array(['Single', 'Partnered'], dtype=object)
```

In [28]:
```
df1['Product'].unique()
```

Out[28]:
```
array(['KP281', 'KP481', 'KP781'], dtype=object)
```

In [29]:
```
df_single = df1[df1["MaritalStatus"]=='Single']
```

In [30]:
```
df_single
```

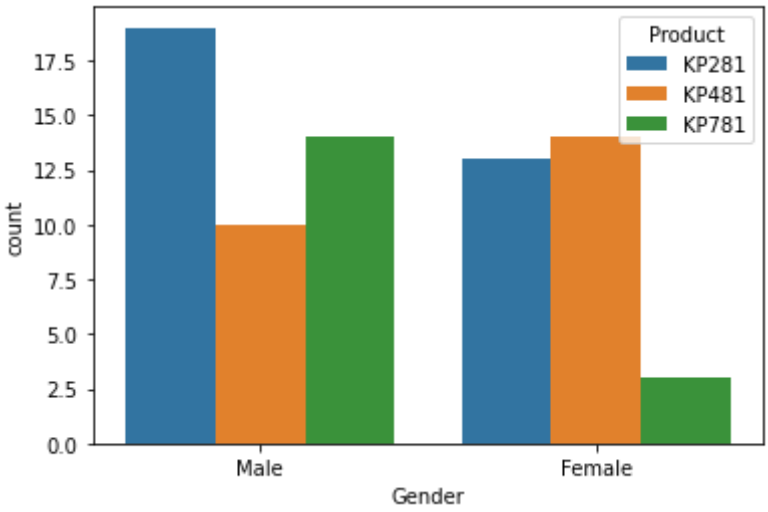
Out[30]:

	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
3	KP281	19	Male	12	Single	3	3	32973	85
7	KP281	21	Male	13	Single	3	3	32973	85
8	KP281	21	Male	15	Single	5	4	35247	141

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
...	...	...	...	...	...	...	...	...	...
165	KP781	29	Male	18	Single	5	5	52290	180
172	KP781	34	Male	16	Single	5	5	92131	150
175	KP781	40	Male	21	Single	6	5	83416	200
176	KP781	42	Male	18	Single	5	4	89641	200
177	KP781	45	Male	16	Single	5	5	90886	160

73 rows × 9 columns

In [31]:
```
sns.countplot(x='Gender', hue = "Product" , data = df_single)
plt.show()
```



In [32]:
```
df_single.loc[df_single["Product"]=="KP281",["Gender"]].value_counts()
```

Out[32]:
```
Gender
Male      19
Female    13
dtype: int64
```

In [33]:
```
df_single.loc[df_single["Product"]=="KP481",["Gender"]].value_counts()
```

Out[33]:
```
Gender
Female    14
Male      10
dtype: int64
```

In [34]:
```
df_single.loc[df_single["Product"]=="KP781",["Gender"]].value_counts()
```

Out[34]:
```
Gender
Male      14
Female     3
dtype: int64
```

In [35]:
```
df_single["Gender"].value_counts()
```

Out[35]:
```
Male      43
```

```
Female    30
Name: Gender, dtype: int64
```

The single male prefer to purchase treadmills more then a single female. ###### The single male prefer to buy an entry-level treadmill that sells for

1,500 more then other two

1,750 more then other two

####### The single female prefer to buy an mid-level runners that sell for

In [36]:
```
df_Partner = df1[df1["MaritalStatus"]=="Partnered"]
```

In [37]:
```
df_Partner
```

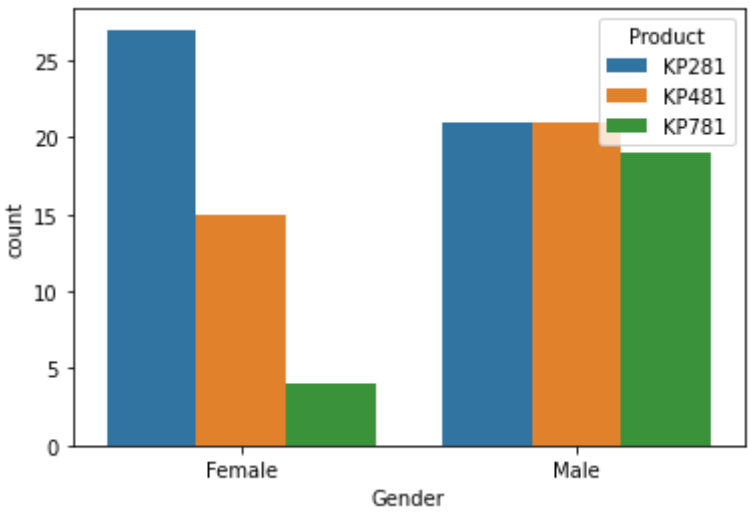
Out[37]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
2	KP281	19	Female	14	Partnered	4	3	30699	66
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
9	KP281	21	Female	15	Partnered	2	3	37521	85
...	...	...	...	...	...	...	...	...	...
171	KP781	33	Female	18	Partnered	4	5	95866	200
173	KP781	35	Male	16	Partnered	4	5	92131	360
174	KP781	38	Male	18	Partnered	5	5	104581	150
178	KP781	47	Male	18	Partnered	4	5	104581	120
179	KP781	48	Male	18	Partnered	4	5	95508	180

107 rows × 9 columns

In [38]:
```
sns.countplot(x='Gender', hue = "Product" , data = df_Partner)
```

Out[38]:
```
<AxesSubplot:xlabel='Gender', ylabel='count'>
```



In [39]:
```
df_Partner["Gender"].value_counts()
```

Out[39]:
```
Male      61
Female    46
Name: Gender, dtype: int64
```

In [40]:
```
df_Partner.loc[df_Partner["Product"]=="KP281",["Gender"]].value_counts()
```

Out[40]:
```
Gender
Female    27
Male      21
dtype: int64
```

In [41]:
```
df_Partner.loc[df_Partner["Product"]=="KP481",["Gender"]].value_counts()
```

Out[41]:
```
Gender
Male      21
Female    15
dtype: int64
```

In [42]:
```
df_Partner.loc[df_Partner["Product"]=="KP781",["Gender"]].value_counts()
```

Out[42]:
```
Gender
Male      19
Female     4
dtype: int64
```

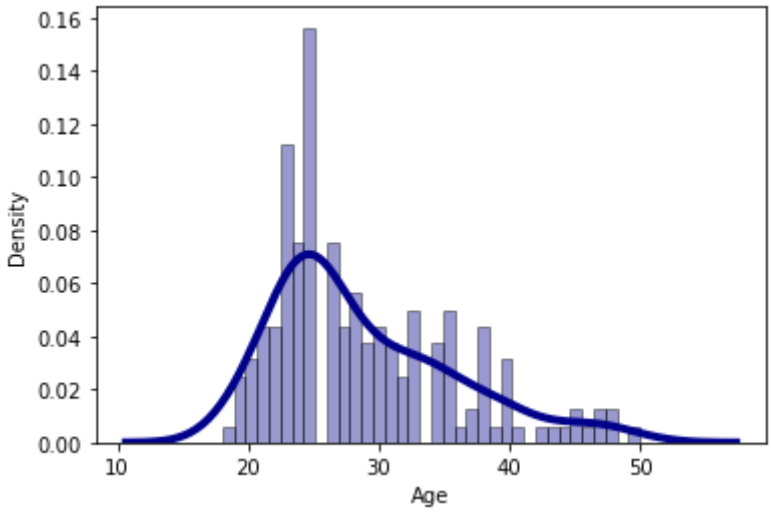
In [43]:
```
###### The married male prefer to purchase treadmills more then a single female.

###### The married male prefer to buy an entry-level treadmill that sells for $1,500

####### The married female prefer to buy an mentry-level treadmill that sells for $1
```

In [44]:
```
df1.columns
```

Out[44]:
```
Index(['Product', 'Age', 'Gender', 'Education', 'MaritalStatus', 'Usage',
       'Fitness', 'Income', 'Miles'],
      dtype='object')
```

In [45]:
```
sns.distplot(df1['Age'], hist=True, kde=True,
bins=int(36), color = 'darkblue',
hist_kws={'edgecolor':'black'},
kde_kws={'linewidth': 4})
plt.show()
```

```
C:\Users\Lenovo\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWar
ning: `distplot` is a deprecated function and will be removed in a future version. P
lease adapt your code to use either `displot` (a figure-level function with similar
flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [46]:
```
df1["Age"].unique()
```

Out[46]:
```
array([18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,
       35, 36, 37, 38, 39, 40, 41, 43, 44, 46, 47, 50, 45, 48, 42],
      dtype=int64)
```

In [47]:
```
df_new=df1.copy()
```

In [48]:
```
df_new
```

Out[48]:

	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
...	...	...	...	...	...	...	...	...	...
175	KP781	40	Male	21	Single	6	5	83416	200
176	KP781	42	Male	18	Single	5	4	89641	200
177	KP781	45	Male	16	Single	5	5	90886	160
178	KP781	47	Male	18	Partnered	4	5	104581	120
179	KP781	48	Male	18	Partnered	4	5	95508	180

180 rows × 9 columns

In [49]:
```
df_new['Age_bracket']=df_new['Age'].copy()
```

In [50]:
```
bins1 = [15,20,25,30,35,40,45,50,55]
labels1 = ['15-19','20-24','25-29','30-34','35-39','40-44','45-49','50-55']
df_new['Age_bracket'] = pd.cut(df_new['Age_bracket'],bins=bins1,labels=labels1)
df_new.head()
```

Out[50]:

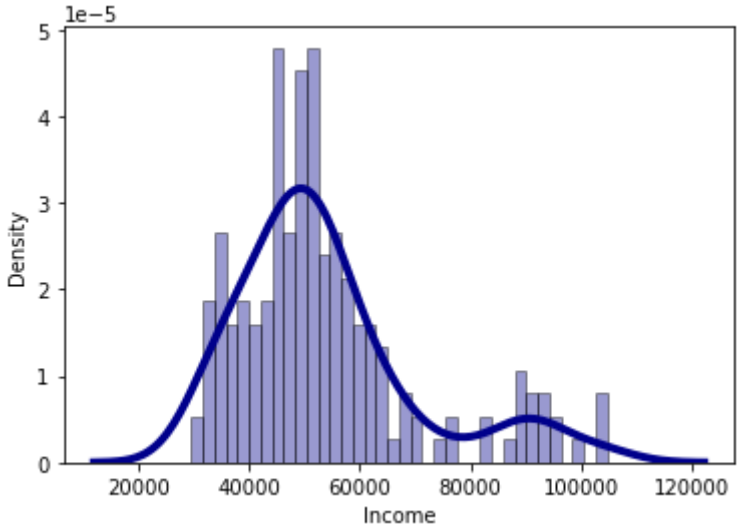
	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles	Age_bracket
0	KP281	18	Male	14	Single	3	4	29562	112	15-19
1	KP281	19	Male	15	Single	2	3	31836	75	15-19
2	KP281	19	Female	14	Partnered	4	3	30699	66	15-19
3	KP281	19	Male	12	Single	3	3	32973	85	15-19
4	KP281	20	Male	13	Partnered	4	2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

In [51]:
```
df1["Income"].unique()
```

Out[51]:
```
array([ 29562,  31836,  30699,  32973,  35247,  37521,  36384,  38658,
        40932,  34110,  39795,  42069,  44343,  45480,  46617,  48891,
        53439,  43206,  52302,  51165,  50028,  54576,  68220,  55713,
        60261,  67083,  56850,  59124,  61398,  57987,  64809,  47754,
        65220,  62535,  48658,  54781,  48556,  58516,  53536,  61006,
        57271,  52291,  49801,  62251,  64741,  70966,  75946,  74701,
        69721,  83416,  88396,  90886,  92131,  77191,  52290,  85906,
       103336,  99601,  89641,  95866, 104581,  95508], dtype=int64)
```

In [52]:
```
sns.distplot(df1['Income'], hist=True, kde=True,
bins=int(36), color = 'darkblue',
hist_kws={'edgecolor':'black'},
kde_kws={'linewidth': 4})
plt.show()
```

```
C:\Users\Lenovo\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWar
ning: `distplot` is a deprecated function and will be removed in a future version. P
lease adapt your code to use either `displot` (a figure-level function with similar
flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [53]:
```
df_new['Income_bracket']=df_new['Income'].copy()
```

In [54]:
```
bins1 = [20000,30000,50000,70000,90000,100000]
labels1 = ["20,000-29,999",'30,000-49,999','50,000-69,999','70,000-90,000','>90,000'
df_new['Income_bracket'] = pd.cut(df_new['Income_bracket'],bins=bins1,labels=labels1
df_new.head()
```

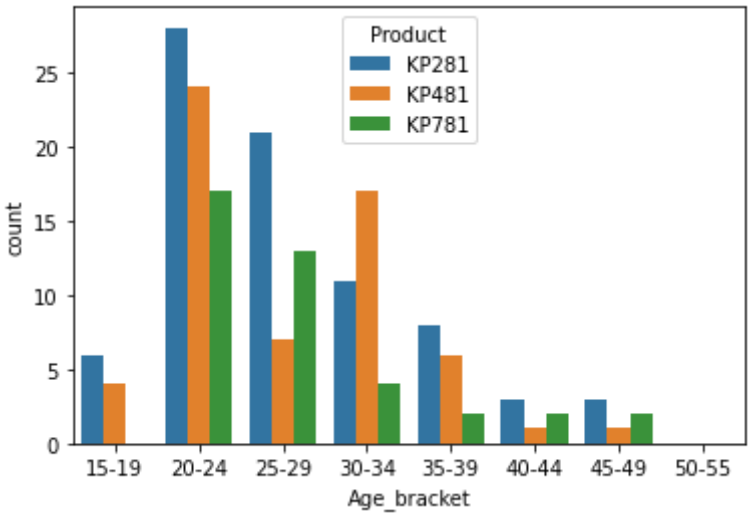
Out[54]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles	Age_bracket	Ir

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

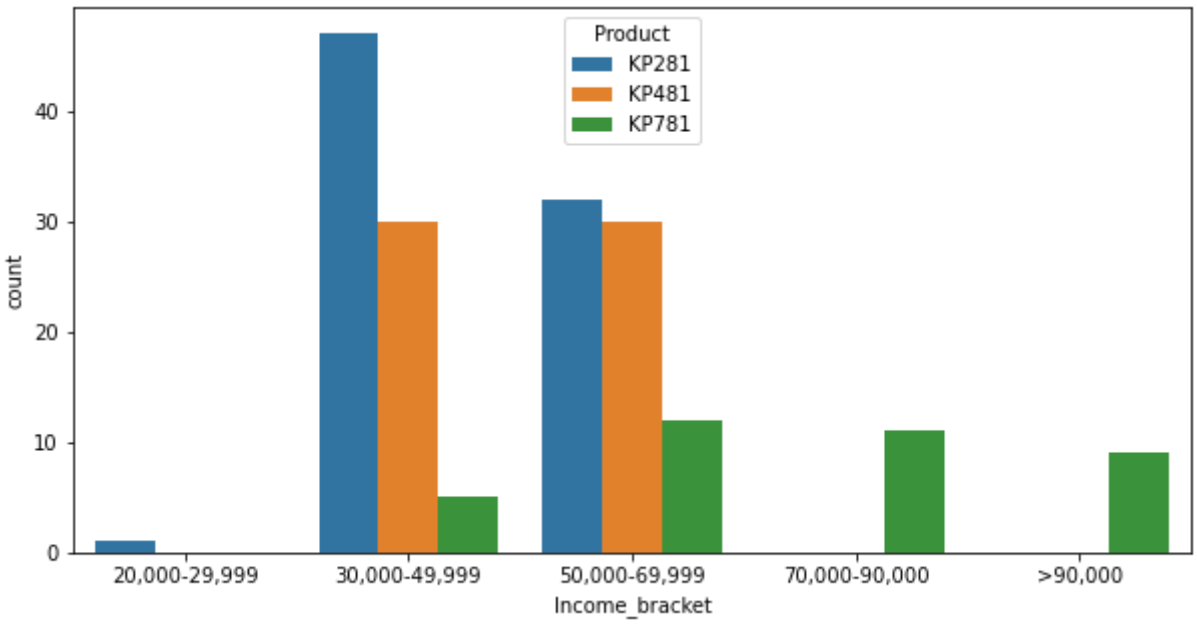
In [55]:
```
sns.countplot(x='Age_bracket', hue = "Product" , data = df_new)
```

Out[55]: <AxesSubplot:xlabel='Age_bracket', ylabel='count'>



In [56]:
```
plt.figure(figsize =(10,5))
sns.countplot(x='Income_bracket', hue = "Product" , data = df_new)
```

Out[56]: <AxesSubplot:xlabel='Income_bracket', ylabel='count'>



In [57]:
```
# The KP281 product is of interest to people who are of the age between 20 to 30 and
# The KP481 product is of interest to people to 20 to 35 and Income range 50K to 70K
```

```
# The KP781 product is of interest to young people of the 20 to 35 and high Income r
```

Marginal Probabilities Product Wise

Marginal probabilities with Product and Gender

In [58]:
```
df_marginal = pd.crosstab(df_new.Gender, df_new.Product, margins = True)
```

In [59]:
```
df_marginal
```

Out[59]:

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

In [60]:
```
df_marginal1 = pd.crosstab(df_new.Gender, df_new.Product, margins = True, normalize
```

In [61]:
```
df_marginal1
```

Out[61]:

Product	KP281	KP481	KP781	All
Gender				
Female	0.222222	0.161111	0.038889	0.422222
Male	0.222222	0.172222	0.183333	0.577778
All	0.444444	0.333333	0.222222	1.000000

Marginal probabilties with Product and Marital Status

In [62]:
```
df_marginal1 = pd.crosstab(df_new.MaritalStatus, df_new.Product, margins = True)
```

In [63]:
```
df_marginal1
```

Out[63]:

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

In [64]:
```
df_marginal1 = pd.crosstab(df_new.MaritalStatus, df_new.Product, margins = True, nor
```

In [65]:
```
df_marginal1
```

Out[65]:

Product	KP281	KP481	KP781	All
MaritalStatus				
Partnered	0.266667	0.200000	0.127778	0.594444
Single	0.177778	0.133333	0.094444	0.405556
All	0.444444	0.333333	0.222222	1.000000

Joint Probability

In [66]:
```
# Probability of product purchased given gender male or female
```

In [67]:
```
df_marginal1 = pd.crosstab(df_new.Gender, df_new.Product, margins= True)
```

In [68]:
```
df_marginal1
```

Out[68]:

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

In [69]:
```
df_marginal1 = pd.crosstab(df_new.Gender, df_new.Product, margins = True, normalize
```

In [70]:
```
df_marginal1
```

Out[70]:

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

In [71]:
```
# P(KP281|Female) = 0.526
```

In [72]:
```
df_marginal2 = pd.crosstab(df_new.MaritalStatus, df_new.Product, margins= True)
```

In [73]:
```
df_marginal2
```

Out[73]:

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

In [74]:
```
df_marginal2 = pd.crosstab(df_new.MaritalStatus, df_new.Product, margins= True, norm
```

In [75]:
```
df_marginal2
```

Out[75]:

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 [76]:
```
#Proability(KP281|Partnered) = 0.448
```

In [77]:
```
# Probability of gender male or female given Product
```

In [78]:
```
df_marginal3 = pd.crosstab(df_new.Gender, df_new.Product, margins= True)
```

In [79]:
```
df_marginal3
```

Out[79]:

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

In [80]:
```
df_marginal3 = pd.crosstab(df_new.Gender, df_new.Product, margins= True, normalize =
```

In [81]:
```
df_marginal3
```

Out[81]:

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

In [82]:
```
df_marginal4 = pd.crosstab(df_new.MaritalStatus, df_new.Product, margins= True)
```

In [83]:
```
df_marginal4
```

Out[83]:

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

In [84]:
```
df_marginal4 = pd.crosstab(df_new.MaritalStatus, df_new.Product, margins= True, norm
```

In [85]:
```
df_marginal4
```

Out[85]:

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

Analysis on marginal and conditonal probability

1.Marginal probability with both genders is high for KP281 treadmill i.e.,44%

2.Marginal probability with marital status partnerd is high overall i.e.,59.4%

3.The Males have high probability overall to buy treadmill

4.The Married people are more health conscious as per data as they have high probability to buy treadmill

5.The probability of buying KP281 treadmill which is beginers level is more for female is 52% ie.P(K281|Female)

6.The probability of buying K781 the most expensive treadmill is more for male is 31.7% and very less for female (~9%)

7.The probability of buying K781 the most expensive treadmill is more for single but the differnce is very minute i.e 1%

8.The probability of Female and buying any treadmill is 15% less then male

9.The probaility of being married and buying treadmill is ~ 20% more then single

In [86]:
```
# Correlation HeatMaps between features
```

In [90]:
```
df_new.columns
```

Out[90]:
```
Index(['Product', 'Age', 'Gender', 'Education', 'MaritalStatus', 'Usage',
       'Fitness', 'Income', 'Miles', 'Age_bracket', 'Income_bracket'],
      dtype='object')
```

In [91]:
```
df_new.corr()
```

Out[91]:

	Age	Education	Usage	Fitness	Income	Miles
Age	1.000000	0.280496	0.015064	0.061105	0.513414	0.036618
Education	0.280496	1.000000	0.395155	0.410581	0.625827	0.307284
Usage	0.015064	0.395155	1.000000	0.668606	0.519537	0.759130
Fitness	0.061105	0.410581	0.668606	1.000000	0.535005	0.785702
Income	0.513414	0.625827	0.519537	0.535005	1.000000	0.543473
Miles	0.036618	0.307284	0.759130	0.785702	0.543473	1.000000

Age is highly positive correlated with Income i.e. The people who with higher age generally have a higher income

Education is highly Positive correlated with Income i.e, Higher the education higher the income

Usage is highly Positive correlated with Fitness, Income, and Mileage i.e. people with higher income usually have a high usage and more fitness enthusiast

Fitness is highly Positive correlated with Usage, Income, Fitness, and Miles

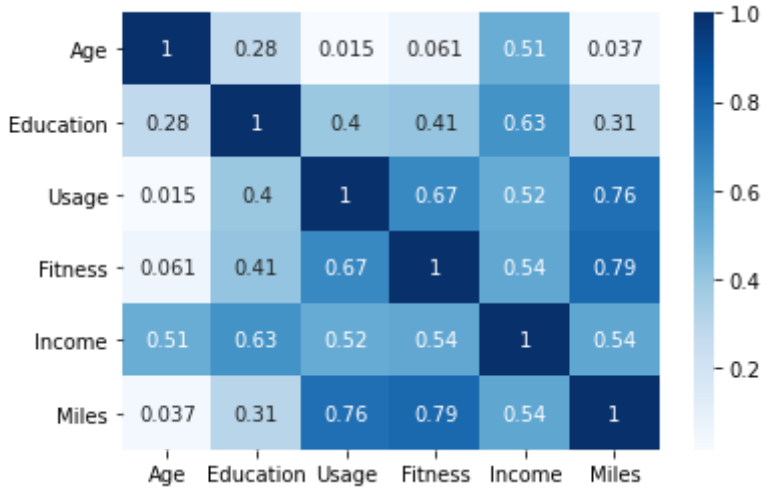
Income is highly correlated with Education

Miles is highly correlated with Usage and Fitness

People with higher income and lower age have a high probability to be fitness enthusiast and can be bet target market to sell Treadmill

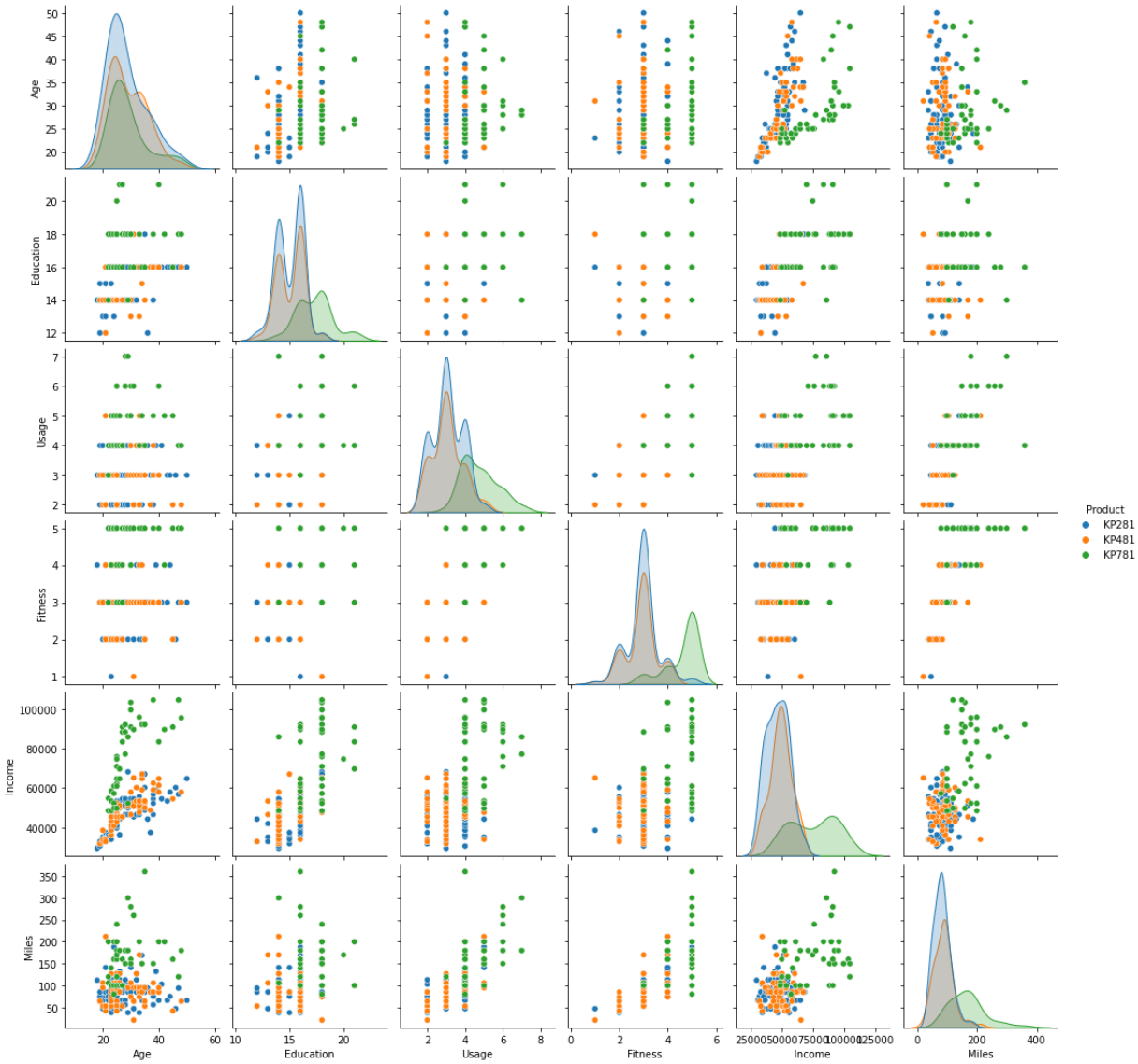
In [92]:
```
sns.heatmap(df_new.corr(), cmap="Blues", annot = True)
```

Out[92]: <AxesSubplot:>



In [93]:
```
sns.pairplot(data=df_new,hue="Product")
plt.plot
```

Out[93]:
```
<function matplotlib.pyplot.plot(*args, scalex=True, scaley=True, data=None, **kwarg
s)>
```



General Infrences from the case study analysis

1. The Income and age have the highest number of outliers in data i.e.170 and 46 respectively
2. The single male prefer to purchase treadmills more then a single female.
3. The single male prefer to buy an entry-level treadmill that sells for $1,500 more then other two
4. The single female prefer to buy an mid-level runners that sell for $1,750 more then other two
5. The KP281 product is of interest to people who are of the age between 20 to 30 and Income Range - 30K to 70K
6. The KP481 product is of interest to people to 20 to 35 and Income range 50K to 70K
7. The KP781 product is of interest to young people of the 20 to 35 and high Income range of 70K and above
8. The probability of buying KP281 treadmill which is beginers level is more for female is 52% ie.P(K281|Female)

9. The probability of buying K781 the most expensive treadmill is more for male is 31.7% and very less for female (~9%)
10. The probability of buying K781 the most expensive treadmill is more for single but the differnce is very minute i.e 1%
11. The probability of Female and buying any treadmill is 15% less then male
12. The probaility of being married and buying treadmill is ~ 20% more then single