In [40]:

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

Dataset Features

Product Purchased: KP281, KP481, or KP781
Age: In years
Gender: Male/Female
Education: In years
MaritalStatus: Single or partnered Usage: The average number of times the customer plans to use the treadmill each week.
Income: Annual income (in $) Fitness: Self-rated fitness on a 1-to-5 scale, where 1 is the poor shape and 5 is the excellent shape. Miles: The average number of miles the customer expects to walk/run each week

In [2]:

```
df = pd.read_csv('aerofit_treadmill.csv')
```

In [3]:

```
df
```

Out[3]:

	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

Data Analysis

In [4]:

```
#statitical summary
df.describe()
```

Out[4]:

	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 [36]:

```
#shape of data
df.shape
```

Out[36]:

```
(180, 9)
```

In [8]:

```
#Checking for missing values
df.isnull().sum()
```

Out[8]:

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

In [9]:

```
#dtypes
df.dtypes
```

Out[9]:

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

In [15]:

```
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
```

Non-Graphical Analysis (Value counts and unique attributes)

In [12]:

```
#How many products are there
df['Product'].unique()
```

Out[12]:

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

Analysis1

Prices($) : KP781 - 2500 , KP481 - 1750, KP281 - 1500

KP781 has been used by more no. of males, could be possible that males are earning more than females or males are more interested in being fit. The other products are not influenced by gender

In [35]:

```
df[['Product','Gender']].value_counts()
```

Out[35]:

```
Product  Gender
KP281    Female    40
         Male      40
KP781    Male      33
KP481    Male      31
         Female    29
KP781    Female     7
dtype: int64
```

Analysis2

Males use 3 to 4 times a treadmill in a week and females mostly used twice/thrice in a week

In [78]:

```
df.groupby(['Product','Gender'])['Usage'].value_counts()
```

Out[78]:

```
Product  Gender  Usage
KP281    Female  3        19
                 2        13
                 4         7
                 5         1
         Male    3        18
                 4        15
                 2         6
                 5         1
KP481    Female  3        14
                 2         7
                 4         5
                 5         3
         Male    3        17
                 2         7
                 4         7
KP781    Female  5         3
                 4         2
                 6         2
         Male    4        16
                 5         9
                 6         5
                 7         2
                 3         1
Name: Usage, dtype: int64
```

Analysis3

1. Most of the cusotmers have self rated themselves as 3
2. 31 of the customers have rated themselves being in excellent shape.

In [84]:

```

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

Out[84]:

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

In [86]:

```
1 df.groupby('Gender')['Fitness'].value_counts()
```

Out[86]:

```
Gender  Fitness
Female  3          45
        2          16
        4           8
        5           6
        1           1
Male    3          52
        5          25
        4          16
        2          10
        1           1
Name: Fitness, dtype: int64
```

In [88]:

```
1 df['Gender'].value_counts()
```

Out[88]:

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

In [83]:

```
1 df.groupby(['Product','Gender'])['Fitness'].value_counts()
```

Out[83]:

```
Product  Gender  Fitness
KP281    Female  3          26
                 2          10
                 4           3
                 5           1
         Male    3          28
                 4           6
                 2           4
                 1           1
                 5           1
KP481    Female  3          18
                 2           6
                 4           4
                 1           1
         Male    3          21
                 2           6
                 4           4
KP781    Female  5           5
                 3           1
                 4           1
         Male    5          24
                 4           6
                 3           3
Name: Fitness, dtype: int64
```

Analysis4 with Education

1. Customer who studied more than 18 years are only buying product KP781 and their income is also more
2. Customers who have bought KP781 product are also having expectation to cover high Miles(above 100) to walk/run each week

In [92]:

```
df['Education'].unique()
```

Out[92]:

```
array([14, 15, 12, 13, 16, 18, 20, 21], dtype=int64)
```

In [98]:

```
df.groupby(['Product','Education']).mean()
```

Out[98]:

		Age	Usage	Fitness	Income	Miles
Product	**Education**					
KP281	**12**	27.500000	3.500000	3.000000	38658.000000	89.500000
	13	21.666667	3.333333	2.333333	36763.000000	59.666667
	14	26.566667	2.800000	2.933333	44608.300000	80.200000
	15	21.000000	2.750000	3.000000	34678.500000	84.750000
	16	31.256410	3.307692	3.025641	49065.923077	85.897436
	18	32.000000	3.000000	3.000000	67651.500000	85.000000
KP481	**12**	21.000000	2.000000	2.000000	32973.000000	53.000000
	13	31.500000	4.000000	3.500000	50028.000000	138.000000
	14	24.478261	3.130435	2.956522	43156.565217	93.521739
	15	34.000000	3.000000	3.000000	67083.000000	85.000000
	16	31.903226	3.032258	2.870968	52668.774194	84.387097
	18	32.000000	2.500000	2.500000	56487.000000	47.500000
KP781	**14**	25.500000	5.500000	4.000000	67282.000000	203.000000
	16	27.866667	4.533333	4.866667	69389.000000	176.666667
	18	30.368421	4.947368	4.578947	80186.315789	160.526316
	20	25.000000	4.000000	5.000000	74701.000000	170.000000
	21	31.000000	4.666667	4.000000	81341.000000	133.333333

Analysis5 with Income

1. KP781 Product median income is more which mean people who are earning more are only buying KP781
2. Customers who studied for >=18years are earning above 75000$

In [99]:

```
df.groupby(['Product'])['Income'].median()
```

Out[99]:

```
Product
KP281    46617.0
KP481    49459.5
KP781    76568.5
Name: Income, dtype: float64
```

In [101]:

```
df.groupby(['Product'])['Income'].min()
```

Out[101]:

```
Product
KP281    29562
KP481    31836
KP781    48556
Name: Income, dtype: int64
```

In [102]:

```
df.groupby(['Product'])['Income'].max()
```

Out[102]:

```
Product
KP281     68220
KP481     67083
KP781    104581
Name: Income, dtype: int64
```

In [100]:

```
df.groupby(['Product','Gender'])['Income'].median()
```

Out[100]:

```
Product  Gender
KP281    Female    46048.5
         Male      46617.0
KP481    Female    48891.0
         Male      50028.0
KP781    Female    69721.0
         Male      77191.0
Name: Income, dtype: float64
```

In [103]:

```
df.groupby('Education')['Income'].median()
```

Out[103]:

```
Education
12    32973.0
13    42069.0
14    45480.0
15    35247.0
16    52302.0
18    75946.0
20    74701.0
21    83416.0
Name: Income, dtype: float64
```

In [108]:

```
df.groupby('Product')['MaritalStatus'].value_counts()
```

Out[108]:

```
Product  MaritalStatus
KP281    Partnered        48
         Single           32
KP481    Partnered        36
         Single           24
KP781    Partnered        23
         Single           17
Name: MaritalStatus, dtype: int64
```

Visual Analysis - Univariate & Bivariate

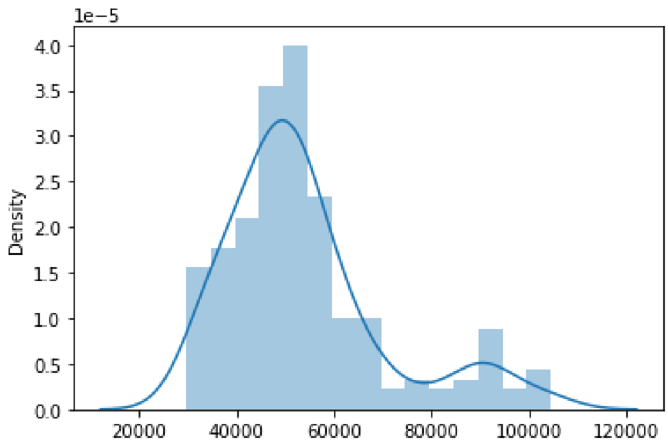
In [122]:

```
sns.distplot(x=df['Income'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please 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)
```

Out[122]:

```
<AxesSubplot:ylabel='Density'>
```

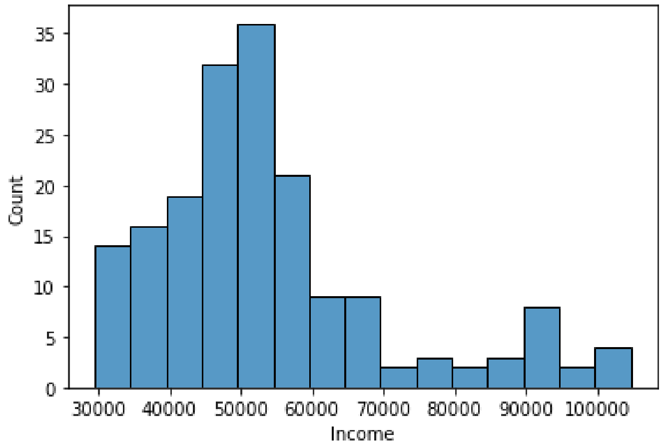


In [124]:

```
sns.histplot(x=df['Income'])
```

Out[124]:

```
<AxesSubplot:xlabel='Income', ylabel='Count'>
```



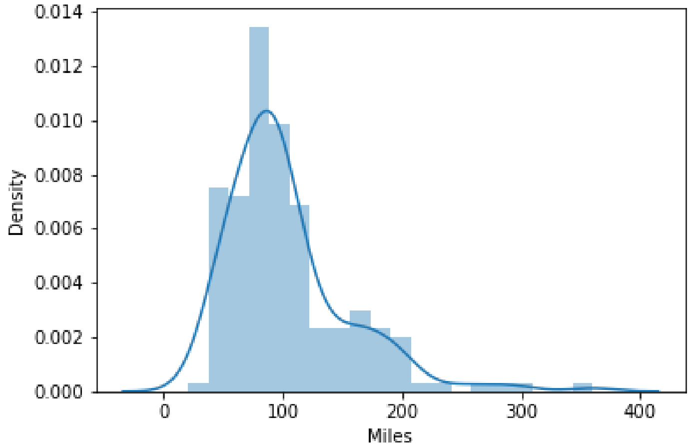
In [156]:

```
sns.distplot(df['Miles'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please 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)
```

Out[156]:

```
<AxesSubplot:xlabel='Miles', ylabel='Density'>
```

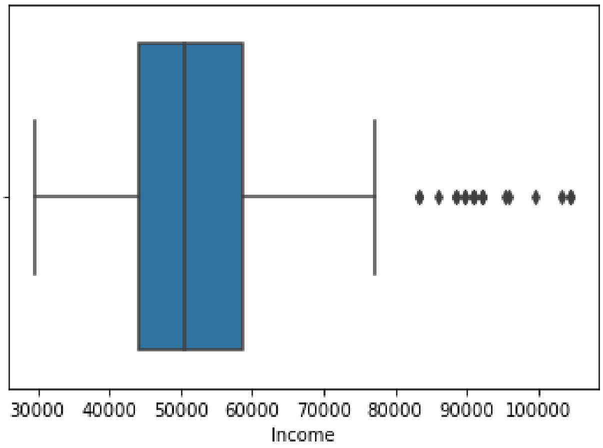


In [150]:

```
#Box plot (Outliers through visualization)
sns.boxplot(x='Income',data=df)
```

Out[150]:

```
<AxesSubplot:xlabel='Income'>
```

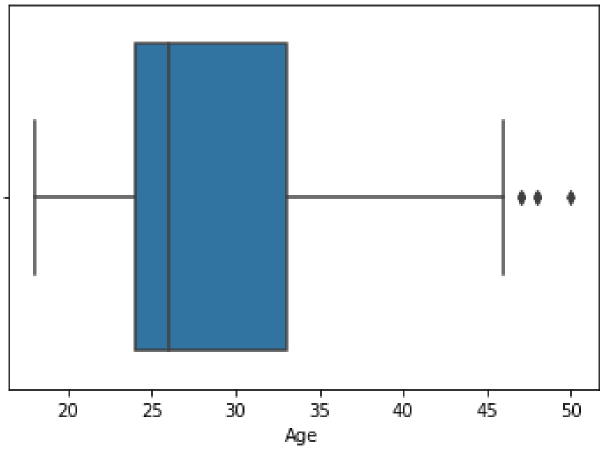


In [151]:

```
sns.boxplot(x='Age',data=df)
```

Out[151]:

```
<AxesSubplot:xlabel='Age'>
```

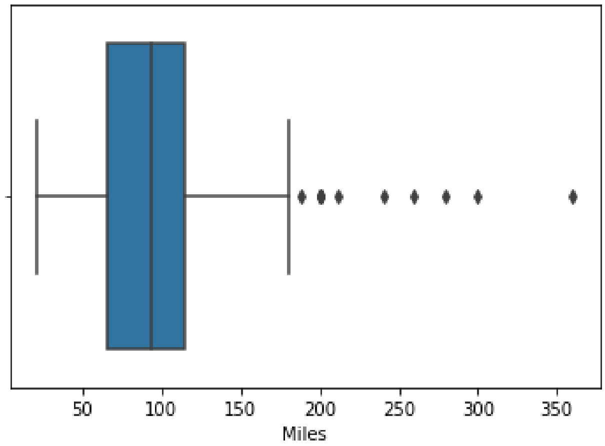


In [154]:

```
sns.boxplot(x='Miles',data=df)
```

Out[154]:

<AxesSubplot:xlabel='Miles'>



Analysis5

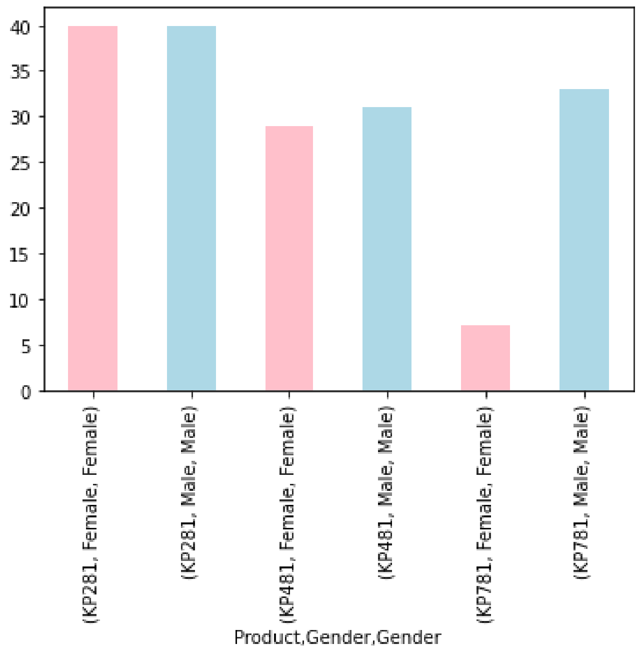
1. More males have bought product KP781 than females.
2. Partners usually buy more Products than single - could be possible that both of them are earning
3. The median Miles customer expects to walk/run each week is more(approximately 165) for product KP781. Customer who bought KP781 have set huge fitness expectations
4. Females who bought product KP781 has median salary 70000 and males bought product KP781 has median salary 75000

In [166]:

```
df.groupby(['Product','Gender'])['Gender'].value_counts().plot(kind='bar',color=['pink','lightblue'
```

Out[166]:

<AxesSubplot:xlabel='Product,Gender,Gender'>

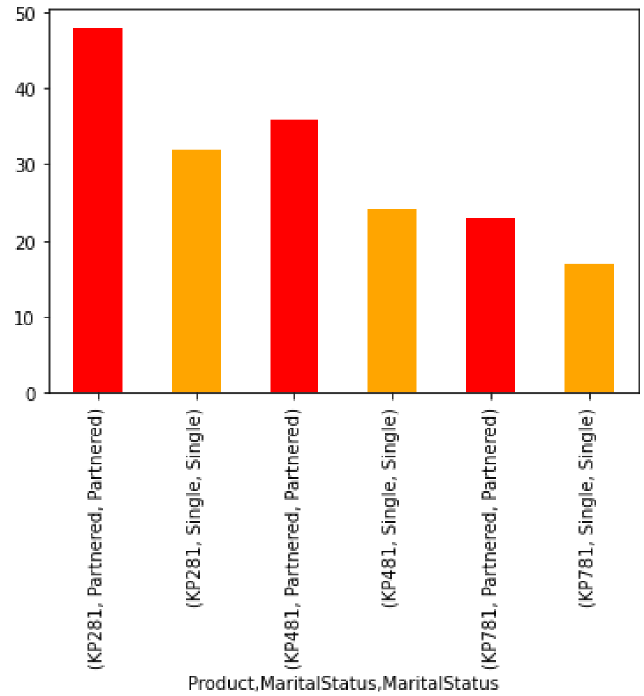


In [170]:

```
#The no of products bought by partnered are more than singles - could be possible that both of them
df.groupby(['Product','MaritalStatus'])['MaritalStatus'].value_counts().plot(kind='bar',color=['red
```

Out[170]:

<AxesSubplot:xlabel='Product,MaritalStatus,MaritalStatus'>

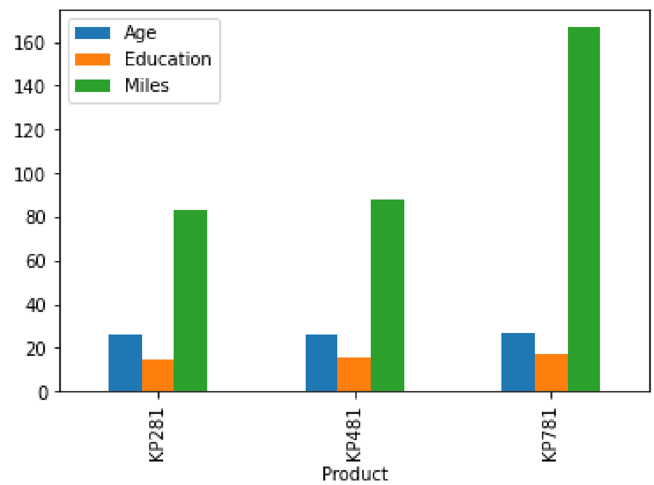


In [184]:

```
df.groupby('Product').aggregate({'Age':'median','Education':'mean','Miles':'mean'}).plot(kind='bar'
```

Out[184]:

<AxesSubplot:xlabel='Product'>

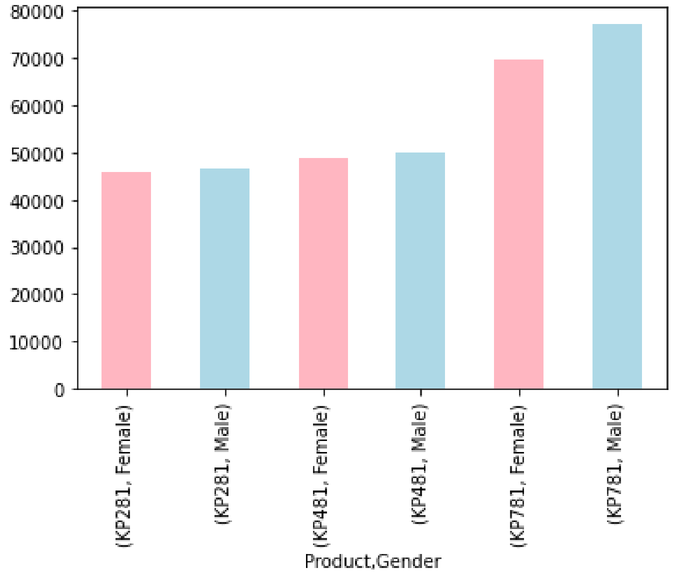


In [191]:

```
#Median income based on Product
df.groupby(['Product','Gender'])['Income'].median().plot(kind='bar',color=['lightpink','lightblue']
```

Out[191]:

<AxesSubplot:xlabel='Product,Gender'>



In [209]:

```
df.corr()
```

Out[209]:

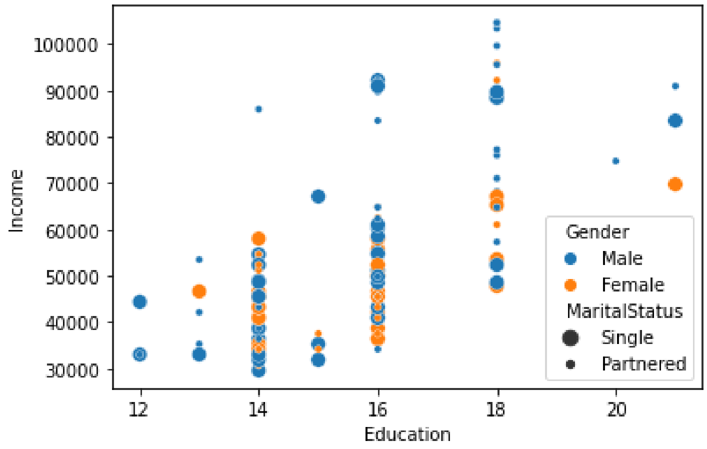
	Age	Education	Usage	Fitness	Income	Miles
Age	1.000000	0.280496	0.015064	0.061105	0.513414	0.036618
Education	0.280496	1.000000	0.395155	0.410581	0.625827	0.307284
Usage	0.015064	0.395155	1.000000	0.668606	0.519537	0.759130
Fitness	0.061105	0.410581	0.668606	1.000000	0.535005	0.785702
Income	0.513414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

In [223]:

```
sns.scatterplot(x='Education',y='Income',hue='Gender',size='MaritalStatus',data=df)
```

Out[223]:

<AxesSubplot:xlabel='Education', ylabel='Income'>

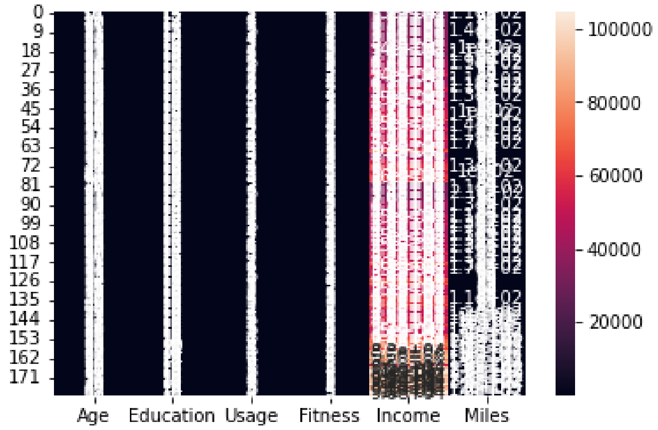


In [214]:

```
sns.heatmap(data=df[['Age','Education','Usage','Fitness','Income','Miles']],annot=True)
```

Out[214]:

<AxesSubplot:>



Outlier detection

In [206]:

```
q1 = df.quantile(0.25)
q3 = df.quantile(0.75)
IQR = q3-q1
outliers = df[((df<(q1-1.5*IQR)) | (df>(q3+1.5*IQR)))]
outliers
```

```
C:\Users\Shravanthi\AppData\Local\Temp\ipykernel_41316\845256912.py:4: FutureWarning: Aut
omatic reindexing on DataFrame vs Series comparisons is deprecated and will raise ValueEr
ror in a future version.  Do `left, right = left.align(right, axis=1, copy=False)` before
e.g. `left == right`
  outliers = df[((df<(q1-1.5*IQR)) | (df>(q3+1.5*IQR)))]
```

Out[206]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...
175	NaN	NaN	NaN	21.0	NaN	6.0	NaN	83416.0	200.0
176	NaN	NaN	NaN	NaN	NaN	NaN	NaN	89641.0	200.0
177	NaN	NaN	NaN	NaN	NaN	NaN	NaN	90886.0	NaN
178	NaN	47.0	NaN	NaN	NaN	NaN	NaN	104581.0	NaN
179	NaN	48.0	NaN	NaN	NaN	NaN	NaN	95508.0	NaN

180 rows × 9 columns

Making categories for income

In [286]:

```
def incomeCategories(x):
    if x<=df['Income'].quantile(0.25):
        return 'Low'
    elif x>df['Income'].quantile(0.25) and x<=df['Income'].quantile(0.75):
        return 'Sufficient'
    else:
        return 'High'
```

In [287]:

```
df['Income_Category'] = df['Income'].apply(incomeCategories)
```

In [253]:

```
q = pd.crosstab(index=df['Product'],columns=df['Income_Category'],margins=True)
q
```

Out[253]:

Income_Category	High	Low	Sufficient	All
Product				
KP281	7	34	39	80
KP481	9	15	36	60
KP781	30	0	10	40
All	46	49	85	180

In [289]:

```
p = pd.crosstab(index=df['Product'],columns=df['Income_Category'],margins=True)
p
```

Out[289]:

Income_Category	High	Low	Sufficient	All
Product				
KP281	7	30	43	80
KP481	9	15	36	60
KP781	29	0	11	40
All	45	45	90	180

```
#### Marginal Probability
1. Probability of buying KP281 = 80/180 = 0.44
2. Probability of buying KP481 = 60/180 = 0.33
3. Probability of buying KP781 = 40/180 = 0.22
```

In [290]:

```
#Probability of buying KP281
80/180
```

Out[290]:

0.4444444444444444

In [291]:

```
#Probability of buying KP481
60/180
```

Out[291]:

0.3333333333333333

In [292]:

```
#Probability of buying KP781
40/180
```

Out[292]:

0.2222222222222222

Join Probabality

1. Probability of Low Income_Category buying KP281
2. Probability of Sufficient Income_Category buying KP281
3. Probability of High Income_Category buying KP281
4. Probability of Low Income_Category buying KP481
5. Probability of Sufficient Income_Category buying KP481
6. Probability of High Income_Category buying KP481
7. Probability of Low Income_Category buying KP781
8. Probability of Sufficient Income_Category buying KP781
9. Probability of High Income_Category buying KP781

In [293]:

```
#P[KP281 intersection Low]
p['Low']['KP281']/p['All']['All']
```

Out[293]:

0.16666666666666666

In [294]:

```
#P[KP281 intersection Sufficient]
p['Sufficient']['KP281']/p['All']['All']
```

Out[294]:

0.2388888888888889

In [295]:

```
#P[KP281 intersection High]
p['High']['KP281']/p['All']['All']
```

Out[295]:

0.03888888888888889

In [296]:

```
#P[KP481 intersection Low]
p['Low']['KP481']/p['All']['All']
```

Out[296]:

0.08333333333333333

In [297]:

```
#P[KP481 intersection Sufficient]
p['Sufficient']['KP481']/p['All']['All']
```

Out[297]:

0.2

In [298]:

```
#P[KP481 intersection High]
p['High']['KP481']/p['All']['All']
```

Out[298]:

```
0.05
```

In [299]:

```
#P[KP781 intersection Low]
p['Low']['KP781']/p['All']['All']
```

Out[299]:

```
0.0
```

In [300]:

```
#P[KP781 intersection Sufficient]
p['Sufficient']['KP781']/p['All']['All']
```

Out[300]:

```
0.06111111111111111
```

In [301]:

```
#P[KP781 intersection Sufficient]
p['High']['KP781']/p['All']['All']
```

Out[301]:

```
0.16111111111111112
```

Conditional Probability

1. Probability of buying KP281 given income_category is Low
2. Probability of buying KP281 given income_category is Sufficient
3. Probability of buying KP281 given income_category is High
4. Probability of buying KP481 given income_category is Low
5. Probability of buying KP481 given income_category is Sufficient
6. Probability of buying KP481 given income_category is High
7. Probability of buying KP781 given income_category is Low
8. Probability of buying KP781 given income_category is Sufficient
9. Probability of buying KP781 given income_category is High

In [302]:

```
# P[KP281 | Low]
p['Low']['KP281']/p['Low']['All']
```

Out[302]:

```
0.6666666666666666
```

In [303]:

```
# P[KP281 | Sufficient]
p['Sufficient']['KP281']/p['Sufficient']['All']
```

Out[303]:

```
0.4777777777777778
```

In [304]:

```
# P[KP281 | High]
p['High']['KP281']/p['High']['All']
```

Out[304]:

```
0.15555555555555556
```

In [305]:

```
# P[KP481 | Low]
p['Low']['KP481']/p['Low']['All']
```

Out[305]:

```
0.3333333333333333
```

In [306]:

```
# P[KP481 | Sufficient]
p['Sufficient']['KP481']/p['Sufficient']['All']
```

Out[306]:

```
0.4
```

In [307]:

```
# P[KP481 | High]
p['High']['KP481']/p['High']['All']
```

Out[307]:

```
0.2
```

In [308]:

```
# P[KP781 | Low]
p['Low']['KP781']/p['Low']['All']
```

Out[308]:

```
0.0
```

In [309]:

```
# P[KP781 | Sufficient]
p['Sufficient']['KP781']/p['Sufficient']['All']
```

Out[309]:

```
0.12222222222222222
```

In [310]:

```
# P[KP781 | High]
p['High']['KP781']/p['High']['All']
```

Out[310]:

```
0.6444444444444445
```

Conditional Probability of age groups

In [316]:

```
def ageGroups(x):
    if x<=24:
        return 'Young Adult'
    elif x>24 and x<=45:
        return 'Middle age'
    else:
        return 'Old age'
```

In [317]:

```
df['Age_Group'] = df['Age'].apply(ageGroups)
```

In [329]:

```
age_grps = pd.crosstab(index=df['Product'],columns=df['Age_Group'],margins=True)
age_grps
```

Out[329]:

Age_Group Product	Middle age	Old age	Young Adult	All
KP281	50	3	27	80
KP481	42	1	17	60
KP781	28	2	10	40
All	120	6	54	180

In [330]:

```
#Prob[KP281|Young Adult]
age_grps['Young Adult']['KP281']/age_grps['Young Adult']['All']
```

Out[330]:

0.5

In [337]:

```
#Prob[KP281|Middle age]
age_grps['Middle age']['KP281']/age_grps['Middle age']['All']
```

Out[337]:

0.4166666666666667

In [336]:

```
#Prob[KP281|Old age]
age_grps['Old age']['KP281']/age_grps['Old age']['All']
```

Out[336]:

0.5

In [338]:

```
#Prob[KP481|Young Adult]
age_grps['Young Adult']['KP481']/age_grps['Young Adult']['All']
```

Out[338]:

```
0.3148148148148148
```

In [339]:

```
#Prob[KP481|Middle age]
age_grps['Middle age']['KP481']/age_grps['Middle age']['All']
```

Out[339]:

```
0.35
```

In [340]:

```
#Prob[KP481|Old age]
age_grps['Old age']['KP481']/age_grps['Old age']['All']
```

Out[340]:

```
0.16666666666666666
```

In [341]:

```
#Prob[KP781|Young Adult]
age_grps['Young Adult']['KP781']/age_grps['Young Adult']['All']
```

Out[341]:

```
0.18518518518518517
```

In [342]:

```
#Prob[KP481|Middle age]
age_grps['Middle age']['KP781']/age_grps['Middle age']['All']
```

Out[342]:

```
0.23333333333333334
```

In [343]:

```
#Prob[KP781|Old age]
age_grps['Old age']['KP781']/age_grps['Old age']['All']
```

Out[343]:

```
0.3333333333333333
```

Recommendations and Actionable Insights

1. More males buy product KP781 than females
2. The number of males and females buying KP281/KP481 are same
3. Customer with High Category Income buy product KP781
4. Middle age(24-45 age group) customers buy more products compared to other age groups so they are more focused on fitness/health
5. Males use treadmills 3 to 4 times a week
6. Females use 2 to 3 times a week
7. Most of the customers have rated(Self-rated fitness) themselves as 3
8. Customers who are buying KP781 have set more expectations - more miles to walk/run each week

9. Partners buy more products than Singles could be possible that both of them are earning
10. People who studies for 18 or more than 18 year mostly buy product KP781
11. Probability of buying KP481 product given his Income is Low is 0.67
12. Probability of buying KP781 product given his Income is High is 0.65
13. Probability of buying KP481 product given customer falls under young age group is 0.5
14. Probability of buying KP481 product given customer falls under old age group is 0.5