1.Defining Problem Statement and Analysing basic metrics

The market research team at AeroFit 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. The team decides to investigate whether there are differences across the product with respect to customer characteristics. Our aim is to :

- Perform descriptive analytics to create a customer profile for each AeroFit treadmill product by developing appropriate tables and charts.
- For each AeroFit treadmill product, construct two-way contingency tables and compute all conditional and marginal probabilities along with their insights/impact on the business.

Observations on shape of data, data types of all the attributes, conversion of categorical attributes to 'category' (If required), statistical summary

In [1]:

```
##importing all the required libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:

```
##reading the data
aerofit=pd.read_csv("treadmill.csv")
```

In [3]:

```
aerofit.shape
```

Out[3]:

```
(180, 9)
```

In [4]:

```
##determining the total number to rows, columns, and data type of each column,null value
aerofit.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
```

- Total number of rows are 180
- Total number of columns are 9
- No null values in each column
- Only 3 categorical attributes and remaining 6 are numerical

In [5]:

```
aerofit.head()
```

Out[5]:

	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

Statistical Summary

In [6]:

```
aerofit.describe()
```

Out[6]:

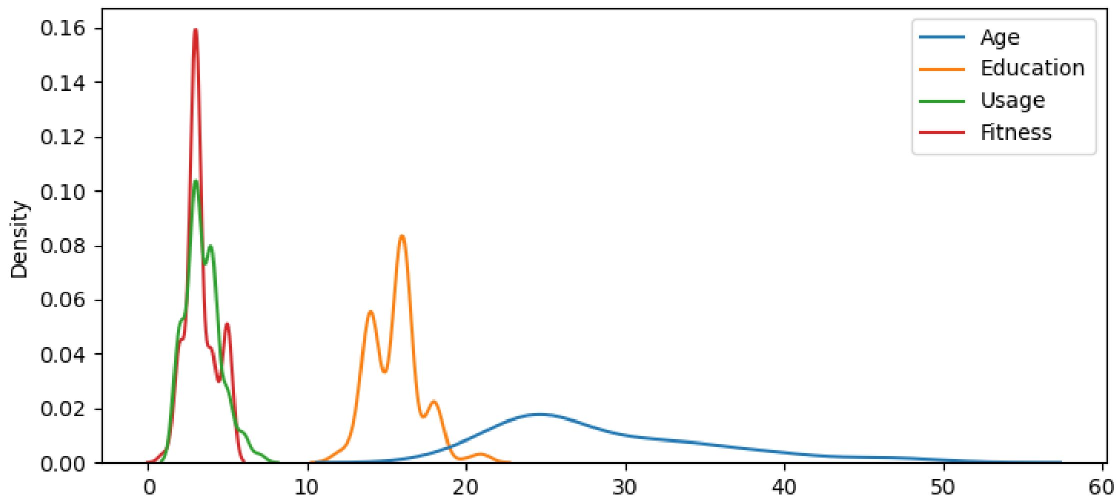
	Age	Education	Usage	Fitness	Income	Miles
count	180.000000	180.000000	180.000000	180.000000	180.000000	180.000000
mean	28.788889	15.572222	3.455556	3.311111	53719.577778	103.194444
std	6.943498	1.617055	1.084797	0.958869	16506.684226	51.863605
min	18.000000	12.000000	2.000000	1.000000	29562.000000	21.000000
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 [356]:

```
sns.kdeplot(aerofit[["Age","Education","Usage","Fitness"]])
```

Out[356]:

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

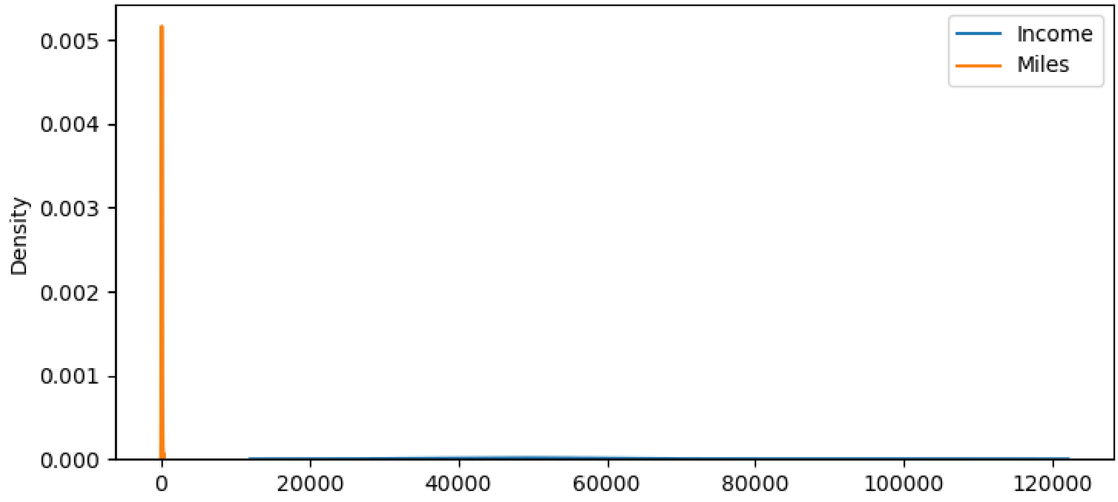


In [354]:

```
sns.kdeplot(aerofit[["Income","Miles"]])
```

Out[354]:

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



- Variance & Standard Deviation are least for Fitness and Usage , most for Age
- Thus Age distribution has a flat bell curve

In [334]:

```
aerofit[["Age","Education","Usage","Fitness","Income","Miles"]].mean()
```

Out[334]:

```
Age              28.788889
Education        15.572222
Usage             3.455556
Fitness           3.311111
Income        53719.577778
Miles           103.194444
dtype: float64
```

In [335]:

```
aerofit[["Age","Education","Usage","Fitness","Income","Miles"]].median()
```

Out[335]:

```
Age              26.0
Education        16.0
Usage             3.0
Fitness           3.0
Income        50596.5
Miles            94.0
dtype: float64
```

- From comparing the above two:
  - Age , Income,Miles has most outlier present, we will be calculating each of them later on

In [336]:

```
aerofit[["Age","Education","Usage","Fitness","Income","Miles"]].mode()
```

Out[336]:

	Age	Education	Usage	Fitness	Income	Miles
0	25	16	3	3	45480	85

- Most of the users are graduates/working professional with average earning and fitness

In [7]:

```
aerofit.describe(include='object')
```

Out[7]:

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

- Top product is KP281, most of the users are partnered males

In [325]:

```
aerofit.describe(include='all')
```

Out[325]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	
count	180	180.000000	180	180.000000	180	180.000000	180.000000	
unique	3	NaN	2	NaN	2	NaN	NaN	
top	KP281	NaN	Male	NaN	Partnered	NaN	NaN	
freq	80	NaN	104	NaN	107	NaN	NaN	
mean	NaN	28.788889	NaN	15.572222	NaN	3.455556	3.311111	5
std	NaN	6.943498	NaN	1.617055	NaN	1.084797	0.958869	1
min	NaN	18.000000	NaN	12.000000	NaN	2.000000	1.000000	2
25%	NaN	24.000000	NaN	14.000000	NaN	3.000000	3.000000	4
50%	NaN	26.000000	NaN	16.000000	NaN	3.000000	3.000000	5
75%	NaN	33.000000	NaN	16.000000	NaN	4.000000	4.000000	5
max	NaN	50.000000	NaN	21.000000	NaN	7.000000	5.000000	10

2.Non-Graphical Analysis: Value counts and unique attributes

In [361]:

```
aerofit["Product"].value_counts()
```

Out[361]:

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

- The KP281 treadmill which is an entry-level treadmill and is the cheapest in the segment of the three treadmills is sold the most
- KP781 has the least maket capture

In [9]:

```
aerofit["Age"].value_counts()
```

Out[9]:

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

- Maximum people are between 23 to 26 years of age
- Minimum people with age above 40

In [368]:

```
aerofit["Gender"].value_counts()
```

Out[368]:

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

- Most people are mamles

In [11]:

```
104/180
```

Out[11]:

```
0.5777777777777777
```

In [12]:

```
76/180
```

Out[12]:

```
0.4222222222222222
```

- Maximum people are males, 57.77 %
- Females percentage is 42.22 %

In [13]:

```
aerofit["Education"].value_counts()
```

Out[13]:

```
16    85
14    55
18    23
15     5
13     5
12     3
21     3
20     1
Name: Education, dtype: int64
```

- Most of the people are graduates
- In general people have atleast completed their secondary education

In [14]:

```
aerofit["MaritalStatus"].value_counts()
```

Out[14]:

```
Partnered    107
Single        73
Name: MaritalStatus, dtype: int64
```

- Most of the people are partnered

In [15]:

```
aerofit["Usage"].value_counts()
```

Out[15]:

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

- Most of the people use the treadmills thrice a week
- Count of people using treadmill for the 6 or 7 days is very less

In [16]:

```
aerofit["Fitness"].value_counts()
```

Out[16]:

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

- Most of the people are in average shape

In [371]:

```
aerofit["Income"].value_counts().head(10)
```

Out[371]:

```
45480    14
52302     9
46617     8
54576     8
53439     8
50028     7
51165     7
40932     6
48891     5
32973     5
Name: Income, dtype: int64
```

- Most of users have salary range between 32k dollars to 45k dollars

In [372]:

```
aerofit["Miles"].value_counts().head(10)
```

Out[372]:

```
85     27
95     12
66     10
75     10
47      9
106     9
94      8
113     8
53      7
100     7
Name: Miles, dtype: int64
```

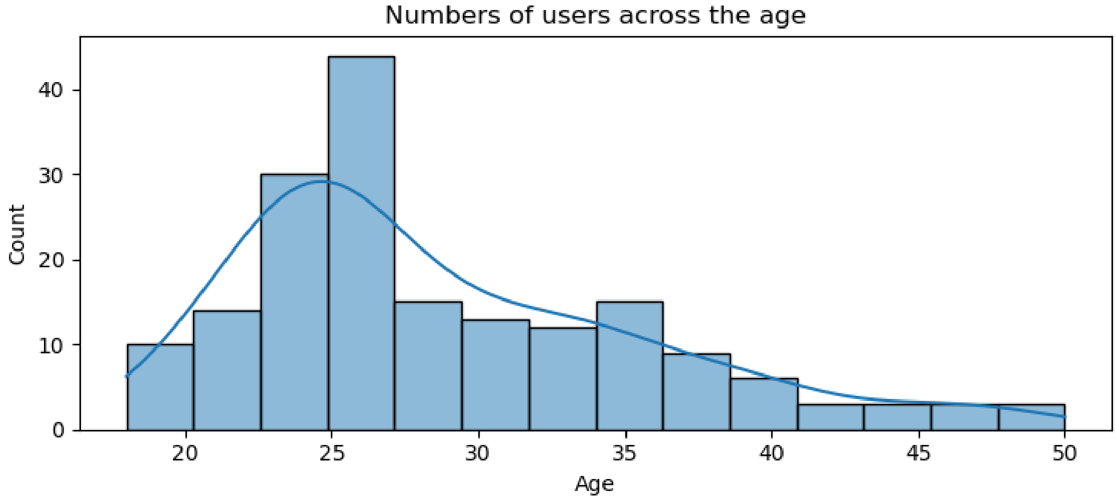
- Most of the users expect to run 85 miles per week

3.Visual Analysis - Univariate & Bivariate

Continous Univariate

In [374]:

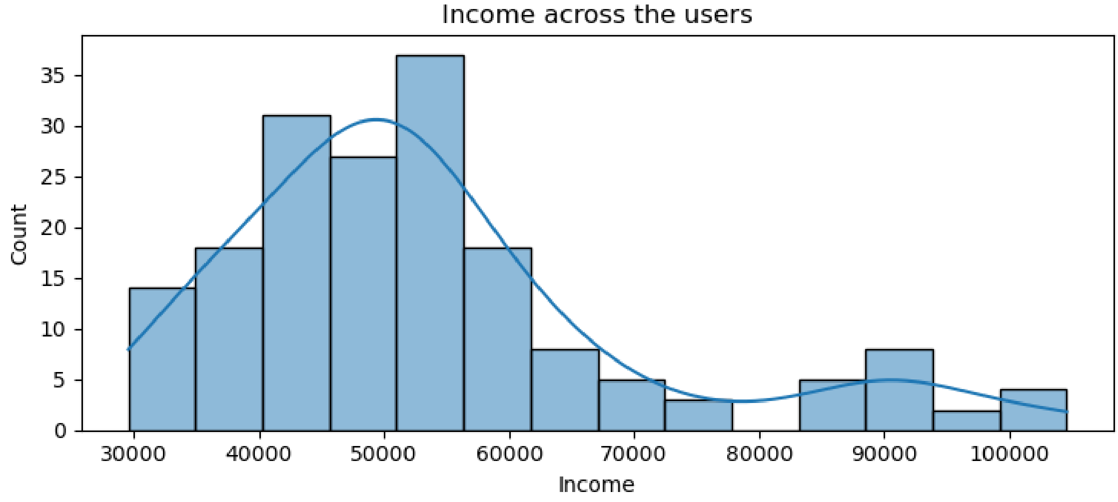
```
plt.rcParams["figure.figsize"] = [7.50, 3.50]
plt.rcParams["figure.autolayout"] = True
sns.histplot(data=aerofit["Age"],bins=14,kde=True)
plt.title("Numbers of users across the age")
plt.show()
```



Most of the users are in the age between 24 and 33

In [375]:

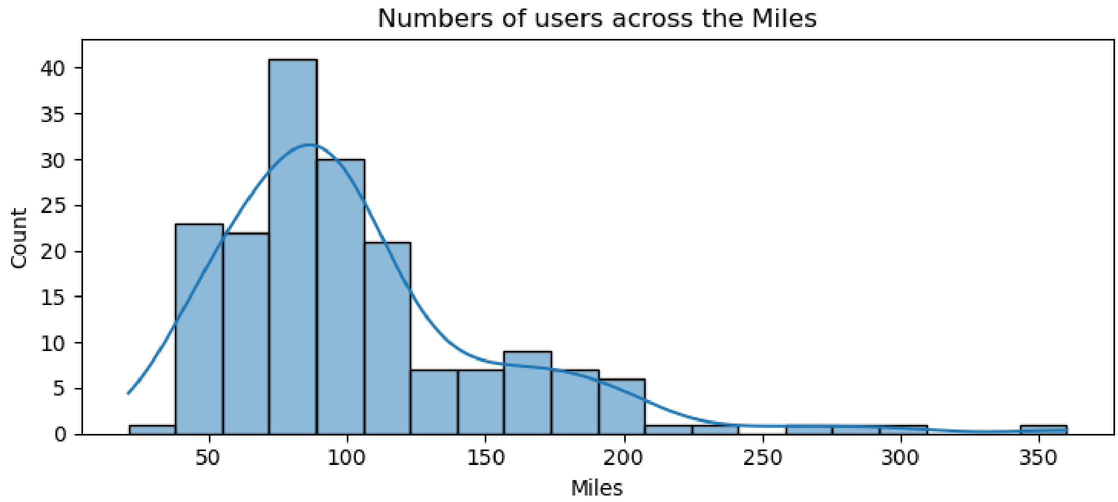
```
sns.histplot(data=aerofit["Income"],bins=14,kde=True,legend=True)
plt.title("Income across the users")
plt.show()
```



The distribution of income follows a uniform symmetric distribution between 30000 to 75000 , outliers are present

In [376]:

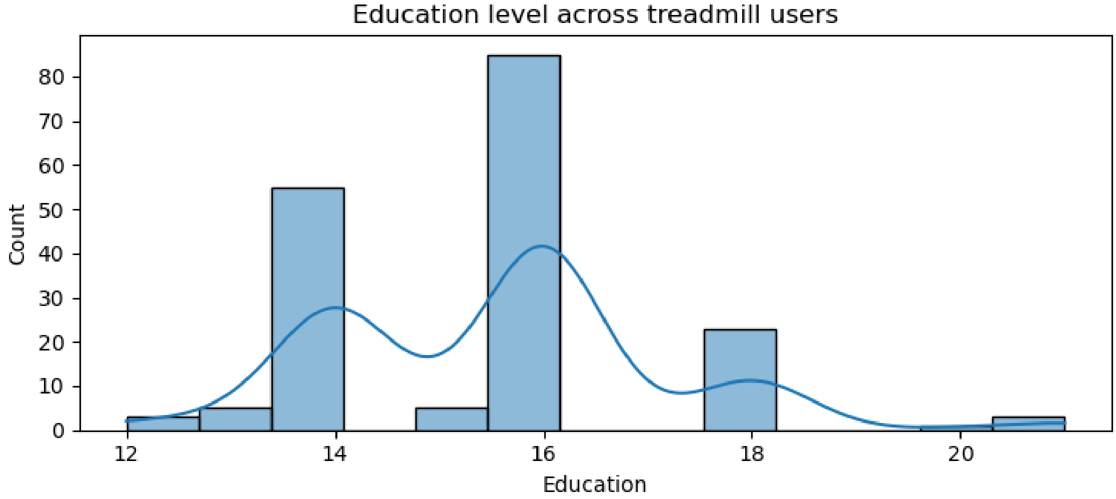
```
sns.histplot(data=aerofit["Miles"],kde=True)
plt.title("Numbers of users across the Miles")
plt.show()
```



Most of the users run between 50 to 150 miles and very less run for more than 150 miles

In [377]:

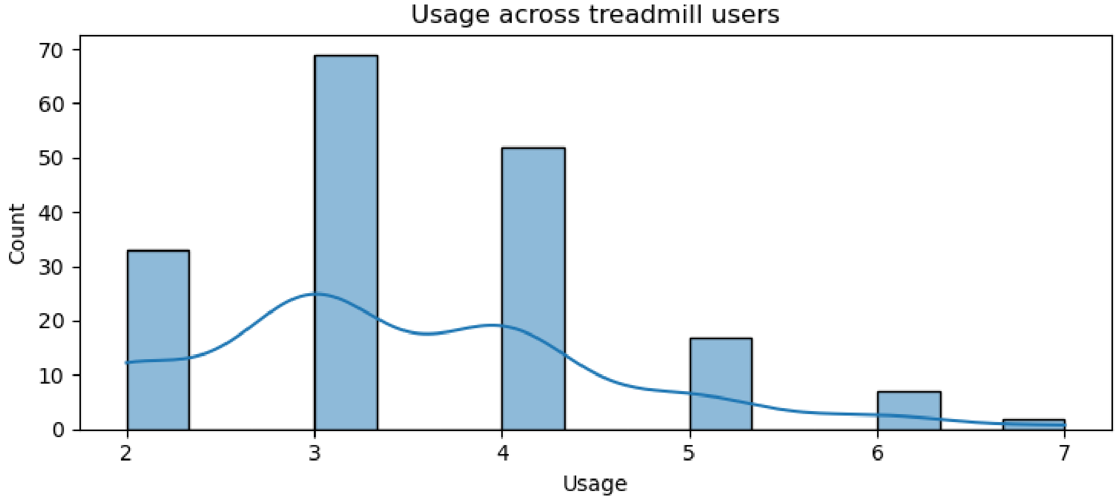
```
sns.histplot(data=aerofit["Education"],kde=True)
plt.title("Education level across treadmill users")
plt.show()
```



Education 16(Diploma) is them most commom education, most of the users have completed have completed their secondary education

In [378]:

```
sns.histplot(data=aerofit["Usage"],kde=True)
plt.title("Usage across treadmill users")
plt.show()
```

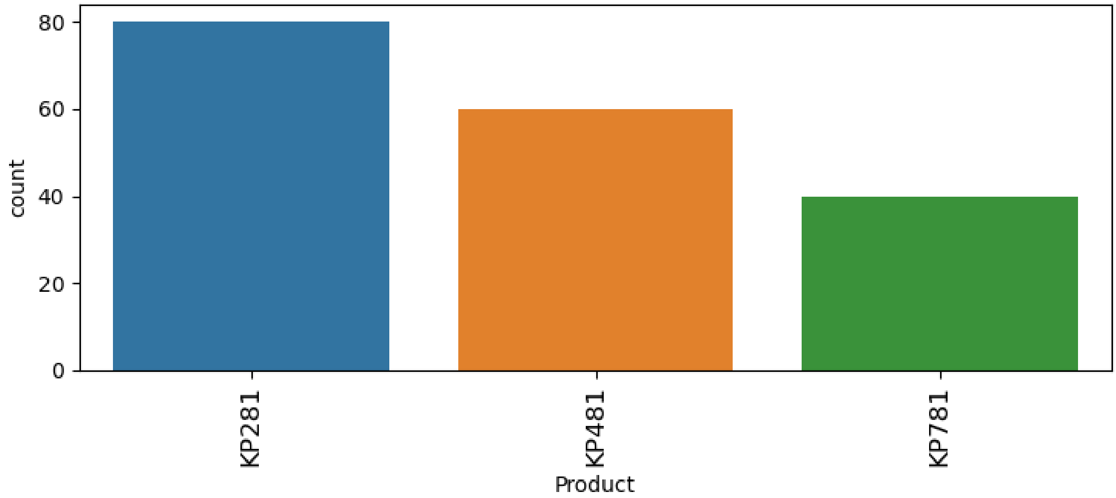


Most of the users spent 3 days per week on treadmill, very less users use it on a daily basis

Categorical Univariate

In [379]:

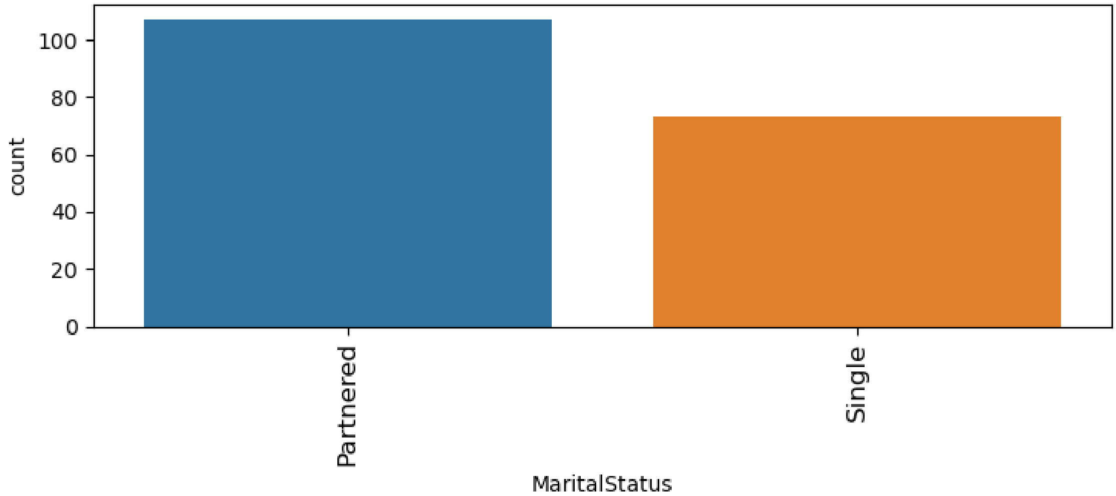
```
sns.countplot(data=aerofit,x="Product")
plt.xticks(rotation=90,fontsize=12) ## to avoid overlapping labels
plt.show()
```



- KP281, which the most basic and cheapest amongst the three, is most common
- KP781 , which is an advanced treadmill and costliest is used the least

In [380]:

```
sns.countplot(data=aerofit,x="MaritalStatus",order = aerofit['MaritalStatus'].value_cour
plt.xticks(rotation=90,fontsize=12) ## to avoid overlapping labels
plt.show()
```



Partnered users focus on fitness more than Single Users

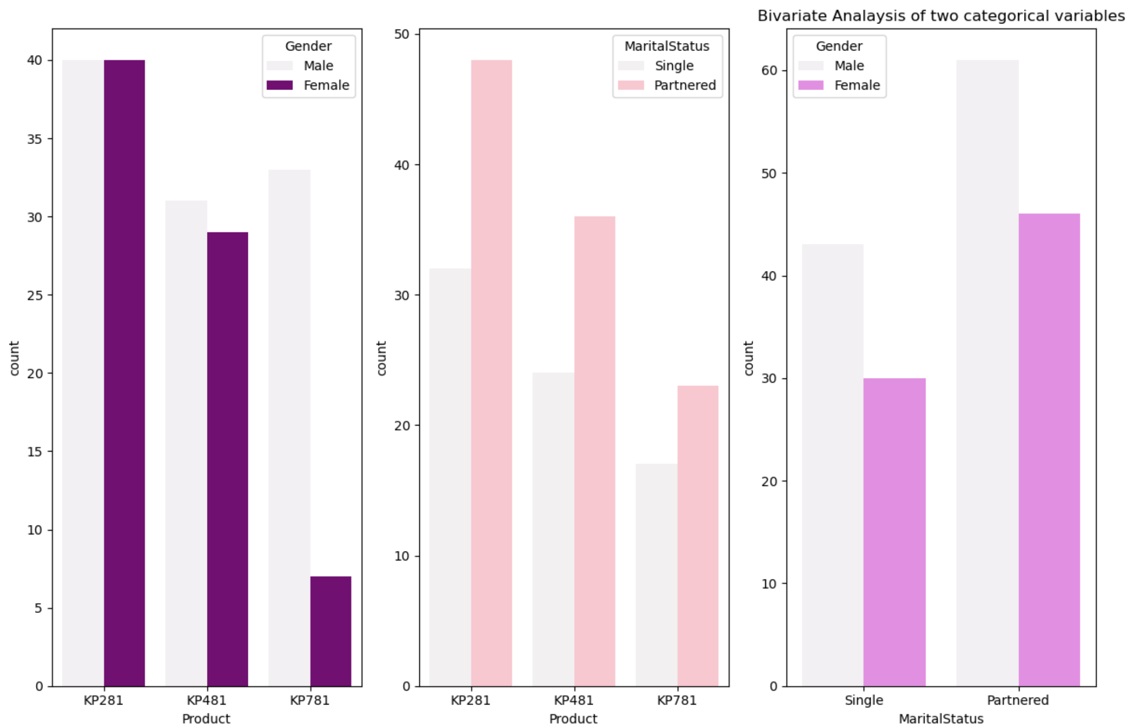
Bivariate Analysis

Bivariate Analaysis of two categorical variables

- Since there are 3 categorical variables , total 3 unique combinations are possible

In [381]:

```
plt.figure(figsize=(12,8))
plt.subplot(1,3,1) ##1 shows the position
sns.countplot(data=aerofit,x='Product',hue='Gender',color='purple')
plt.subplot(1,3,2)
sns.countplot(data=aerofit,x='Product',hue='MaritalStatus',color='pink')
plt.subplot(1,3,3)
sns.countplot(data=aerofit,x='MaritalStatus',hue='Gender',color='violet')
plt.title("Bivariate Analaysis of two categorical variables")
plt.show()
```

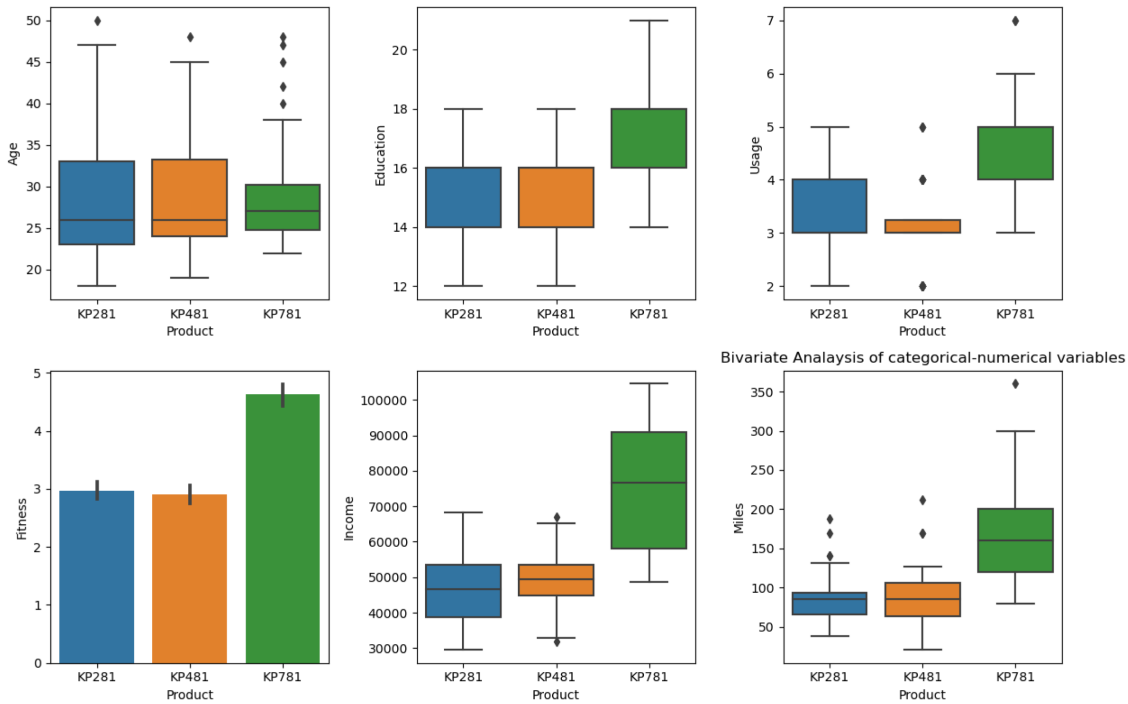


- Gender ratio of users for entry level treadmill , KP281, is 1:1, however the ratio of users is varying widely for advanced level treadmill KP781 , maximum users are male
- Most of the treadmill users are partnered , count of partnered males > count of partnered females

Bivariate analysis for Categorical-Numerical or Numerical-Categorical

In [382]:

```
plt.figure(figsize=(12,8))
plt.subplot(2,3,1) ##1 shows the position
sns.boxplot(data=aerofit,x='Product',y='Age')
plt.subplot(2,3,2)
sns.boxplot(data=aerofit,x='Product',y='Education')
plt.subplot(2,3,3)
sns.boxplot(data=aerofit,x='Product',y='Usage')
plt.subplot(2,3,4)
sns.barplot(data=aerofit,x='Product',y='Fitness')
plt.subplot(2,3,5)
sns.boxplot(data=aerofit,x='Product',y='Income')
plt.subplot(2,3,6)
sns.boxplot(data=aerofit,x='Product',y='Miles')
plt.title("Bivariate Analaysis of categorical-numerical variables")
plt.show()
```

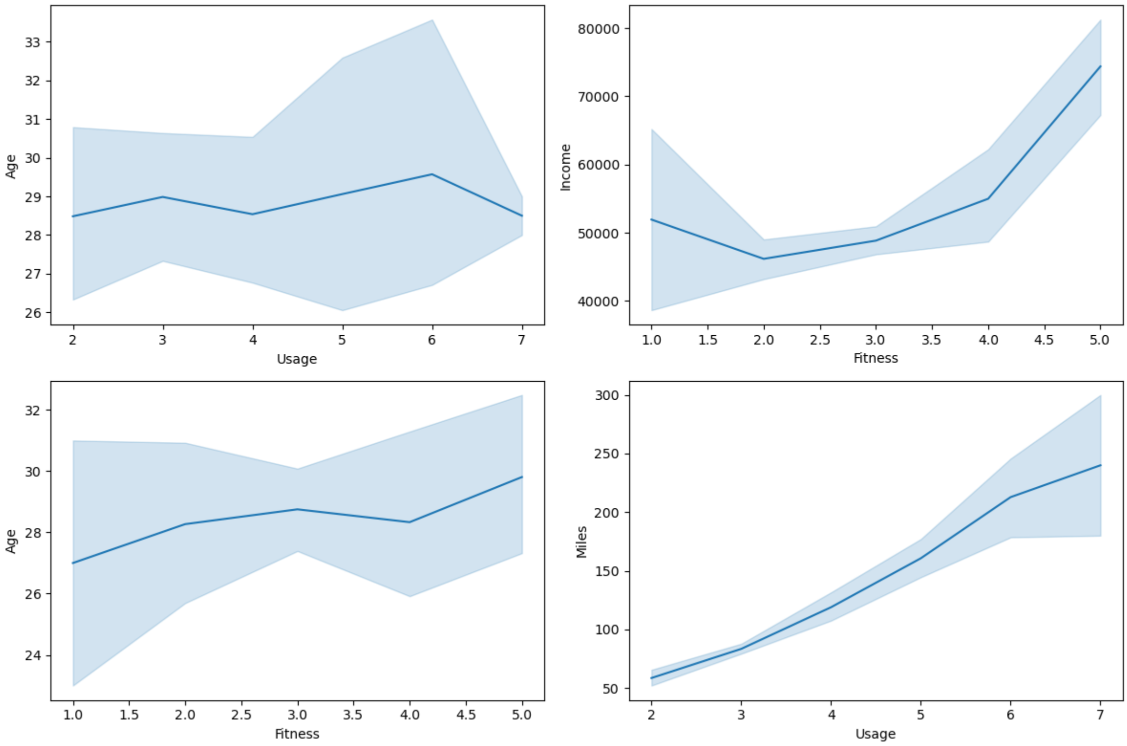


- Most of the users for KP281 and KP481 are aged from 24 to 33 years
- Most of the users for KP781 are aged from 25 to 30 years, this has more number of outliers
- User of KP781 are most qualified and educated and also have the higher income , usage and miles, fitness

Bivariate Analysis for Numerical-Numerical variables

In [383]:

```
plt.figure(figsize=(12,8))
plt.subplot(2,2,1) ##1 shows the position
sns.lineplot(data=aerofit,x='Usage',y='Age',sort=True)
plt.subplot(2,2,2)
sns.lineplot(data=aerofit,x='Fitness',y='Income')
plt.subplot(2,2,3)
sns.lineplot(data=aerofit,x='Fitness',y='Age')
plt.subplot(2,2,4)
sns.lineplot(data=aerofit,x='Usage',y='Miles')
plt.show()
```



Mutivariate Analysis

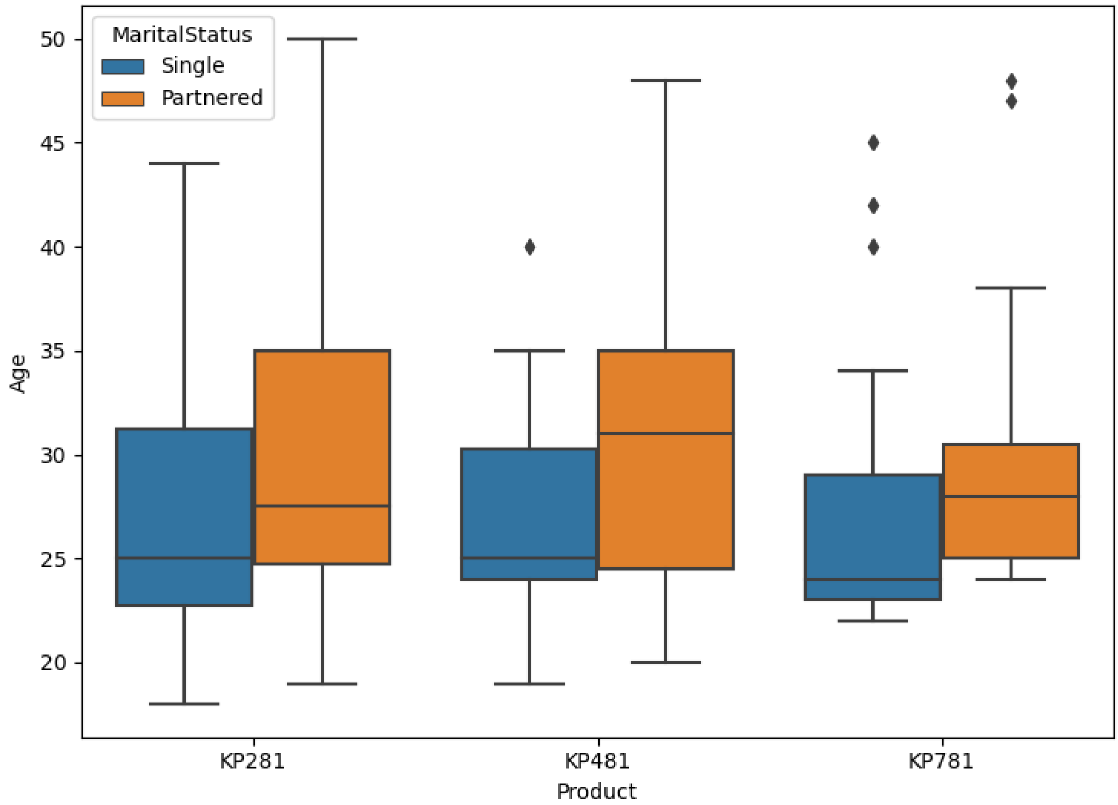
Effect of Marital state , Age on Product Puchased

In [397]:

```
plt.rcParams["figure.figsize"] = [7.50, 5.50]
plt.rcParams["figure.autolayout"] = True
sns.boxplot(data=aerofit,x='Product',y='Age',hue='MaritalStatus')
```

Out[397]:

```
<Axes: xlabel='Product', ylabel='Age'>
```



- Median age of partnered people is higher for KP481 than median age of Partnered people for KP281/781,50% of partnered people with age between use KP481.These people are more consistent
- Median age of single people is almost same across the product types

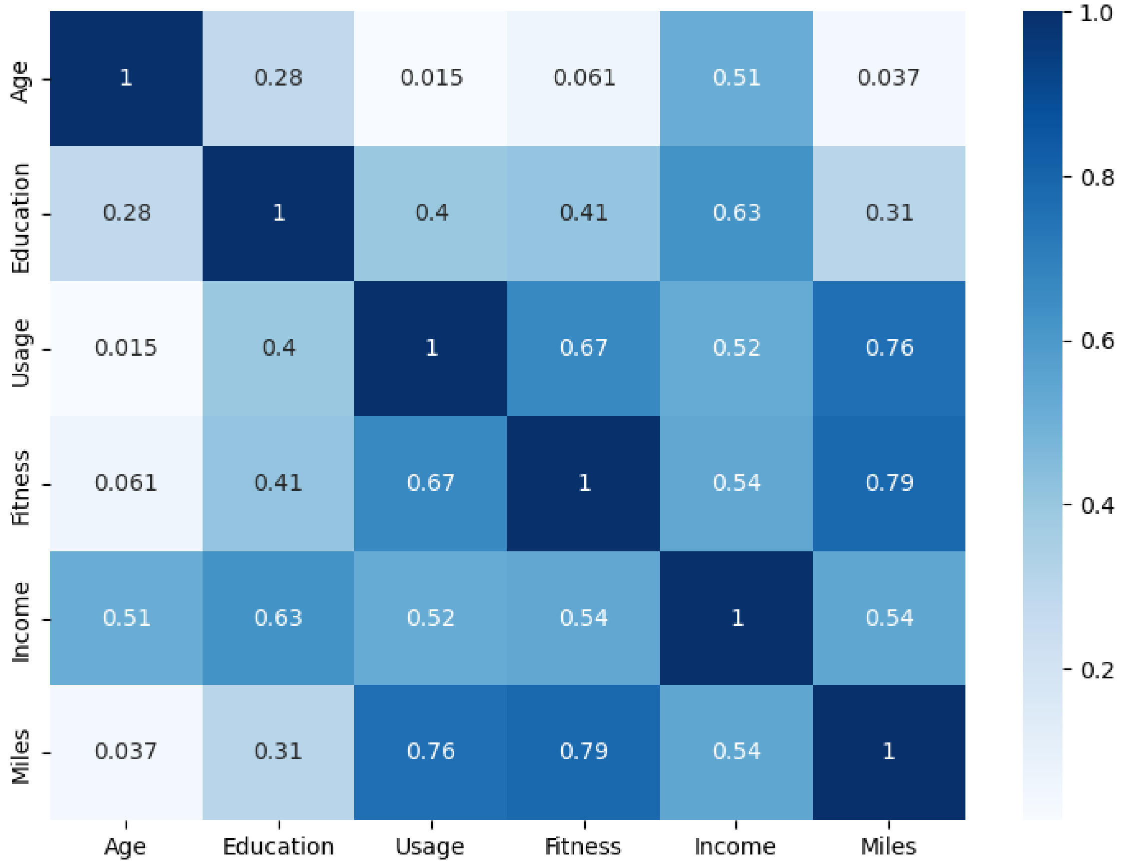
Correlation

In [400]:

```
plt.rcParams["figure.figsize"] = [7.50, 5.50]
plt.rcParams["figure.autolayout"] = True
sns.heatmap(aerofit[["Age","Education","Usage","Fitness","Income","Miles"]].corr(),annot
```

Out[400]:

```
<Axes: >
```



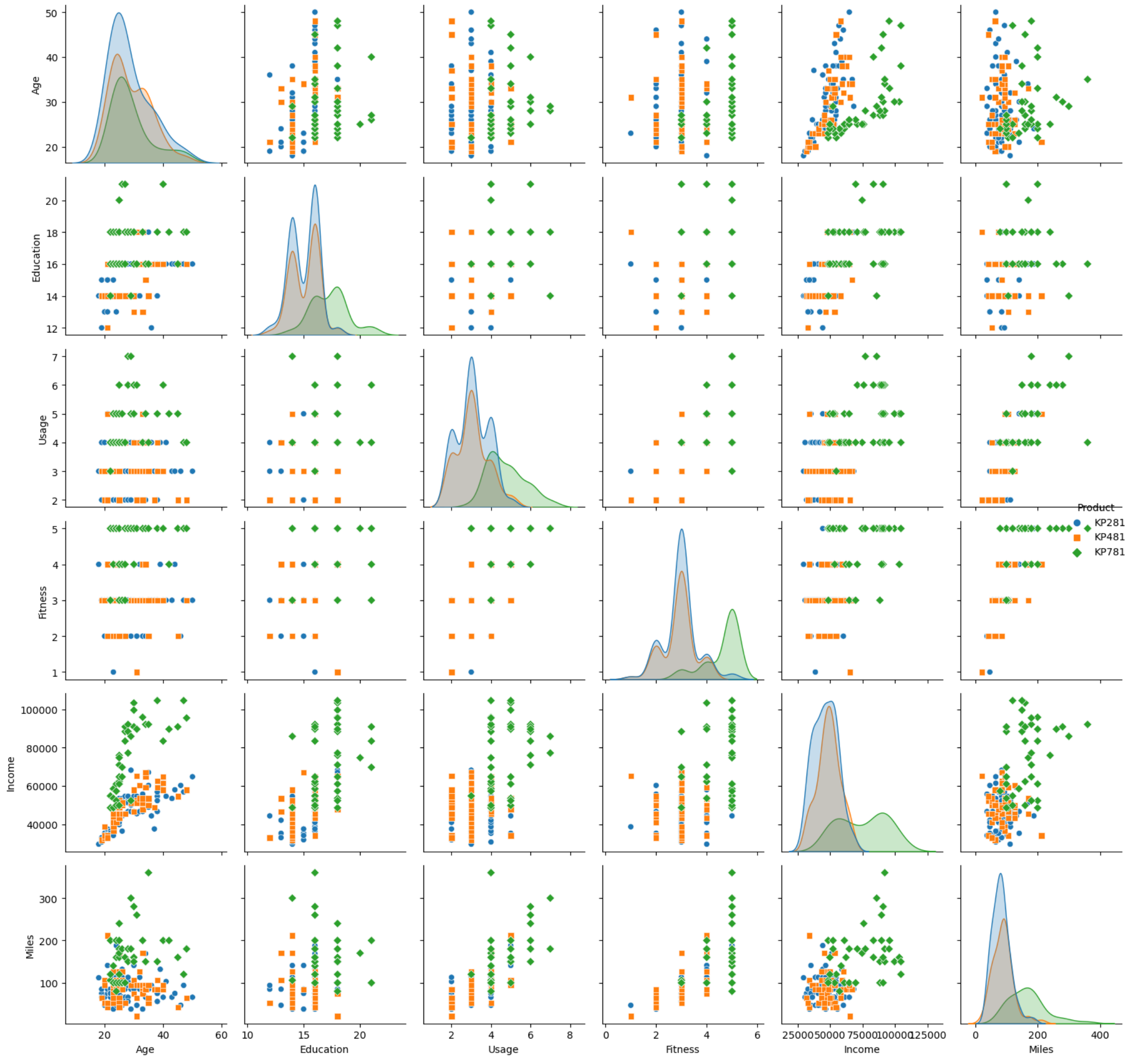
- Age is moderately correlated with Income, negligibly correlated with Education,Usage,Fitness,Miles
- Education is moderately correlated with Income,negligibly correlated with Age,Usage,Fitness,Miles
- Usage is highly correlated with Miles ,moderately correlated with Fitness,Income,negligibly correlated with Age,Education
- Fitness is highly correlated with Miles
- Income is moderaltey correlated with Age , Education, Usage,Fitness,Miles

In [387]:

```
plt.rcParams['figure.figsize']=(30,30)
sns.pairplot(data=aerofit,hue='Product',markers=["o", "s", "D"])
```

Out[387]:

```
<seaborn.axisgrid.PairGrid at 0x220ca6842e0>
```



4.Missing Value & Outlier Detection

- There are no missing values

Outliers Detection

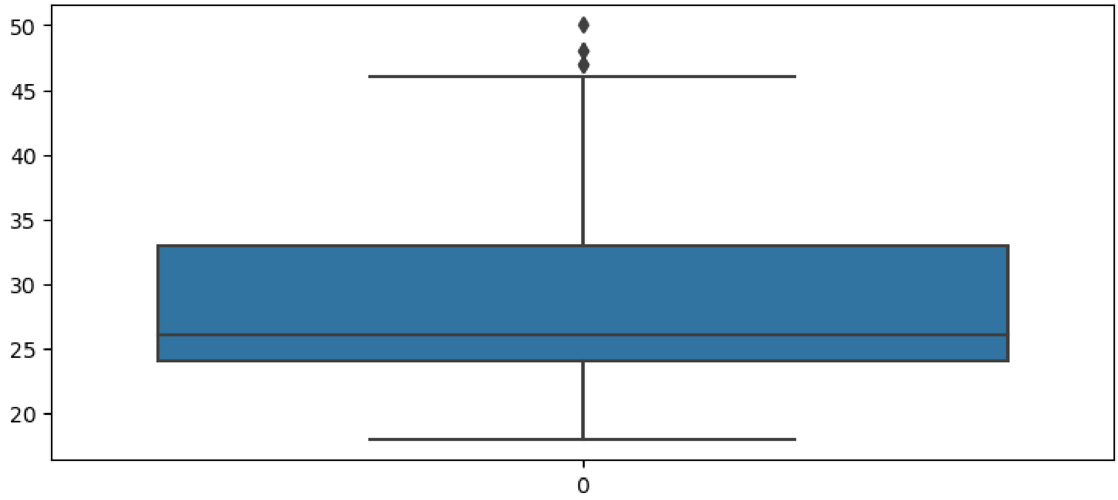
Calculation of outliers , upper/lower whisker , IQR for Age

In [385]:

```
plt.rcParams["figure.figsize"] = [7.50, 3.50]
plt.rcParams["figure.autolayout"] = True
sns.boxplot(data=aerofit["Age"])
```

Out[385]:

```
<Axes: >
```



In [224]:

```
p_25_age = np.percentile(aerofit['Age'],25) ##25th percentile
p_50_age = np.percentile(aerofit['Age'],50) ##50 percentile
p_75_age = np.percentile(aerofit['Age'],75) ##75 percentile
IQR_age = p_75_age - p_25_age
upper_age =  p_75_age + 1.5*IQR_age
lower_age = max(p_25_age - 1.5*IQR_age,0)
age_outliers=aerofit.loc[aerofit['Age'] > upper_age]["Age"]
percent_ouliers_age=(len(age_outliers)/len(aerofit['Age']))*100
print(f"Upper Whisker: {upper_age} \nLower Whisker: {lower_age}\nIQR: {IQR_age}\nOutlier
```

```
Upper Whisker: 46.5
Lower Whisker: 10.5
IQR: 9.0
Outlier in Age:[47 50 48 47 48]
Number of Outliers in Age:5
percent_ouliers_age:2.78
```

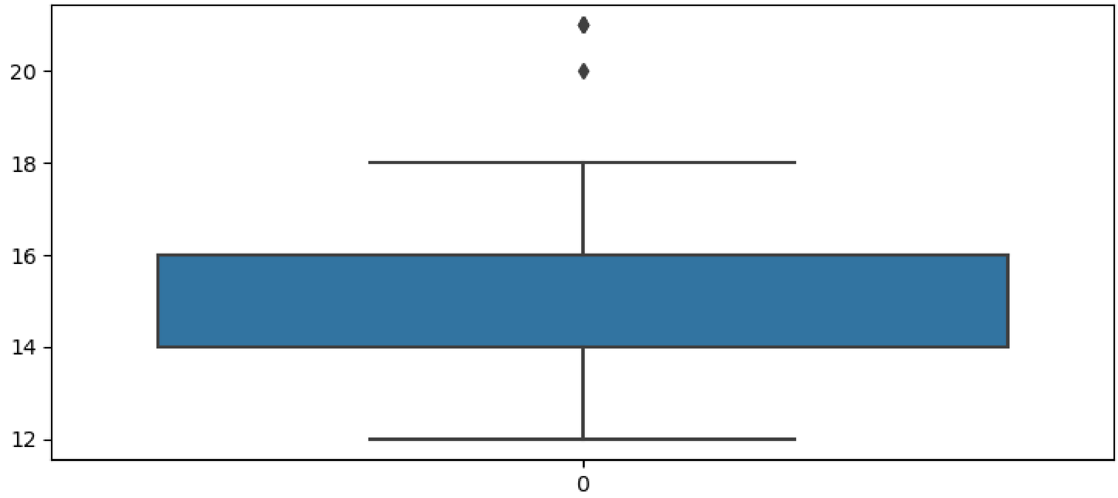
Calculation of outliers , upper/lower whisker , IQR for Education

In [386]:

```
plt.rcParams["figure.figsize"] = [7.50, 3.50]
plt.rcParams["figure.autolayout"] = True
sns.boxplot(data=aerofit["Education"])
```

Out[386]:

```
<Axes: >
```



In [231]:

```
p_25_Education = np.percentile(aerofit['Education'],25) ##25th percentile
p_50_Education = np.percentile(aerofit['Education'],50) ##50 percentile
p_75_Education = np.percentile(aerofit['Education'],75) ##75 percentile
IQR_Education = p_75_Education - p_25_Education
upper_Education =  p_75_Education + 1.5*IQR_Education
lower_Education = max(p_25_Education - 1.5*IQR_Education,0)
Education_outliers=aerofit.loc[aerofit['Education'] > upper_Education]["Education"]
percent_outliers_Education=(len(Education_outliers)/len(aerofit['Education']))*100
print(f"Upper Whisker: {upper_Education} \nLower Whisker: {lower_Education}\nIQR: {IQR_E
```

```
Upper Whisker: 19.0
Lower Whisker: 11.0
IQR: 2.0
Outlier in Education:[20 21 21 21]
Number of Outliers in Education:4
percent_ouliers_Education:2.22
```

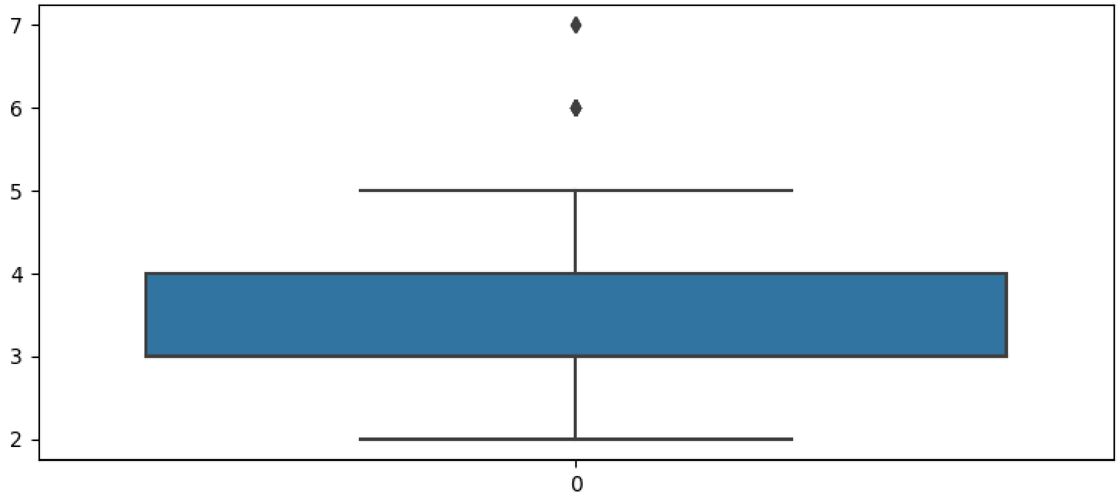
Calculation of outliers , upper/lower whisker , IQR for Usage

In [232]:

```
plt.rcParams["figure.figsize"] = [7.50, 3.50]
plt.rcParams["figure.autolayout"] = True
sns.boxplot(data=aerofit["Usage"])
```

Out[232]:

```
<Axes: >
```



In [240]:

```
p_25_Usage = np.percentile(aerofit['Usage'],25) ##25th percentile
p_50_Usage = np.percentile(aerofit['Usage'],50) ##50 percentile
p_75_Usage = np.percentile(aerofit['Usage'],75) ##75 percentile
IQR_Usage = p_75_Usage - p_25_Usage
upper_Usage =  p_75_Usage + 1.5*IQR_Usage
lower_Usage = max(p_25_Usage - 1.5*IQR_Usage,0)
Usage_outliers=aerofit.loc[(aerofit['Usage'] > upper_Usage) | (aerofit['Usage'] < lower_
percent_outliers_Usage=(len(Usage_outliers)/len(aerofit['Usage']))*100
print(f"Upper Whisker: {upper_Usage} \nLower Whisker: {lower_Usage}\nIQR: {IQR_Usage}\nO
```

```
Upper Whisker: 5.5
Lower Whisker: 1.5
IQR: 1.0
Outlier in Usage:[6 6 6 7 6 7 6 6 6]
Number of Outliers in Usage:9
percent_ouliers_Usage:5.0
```

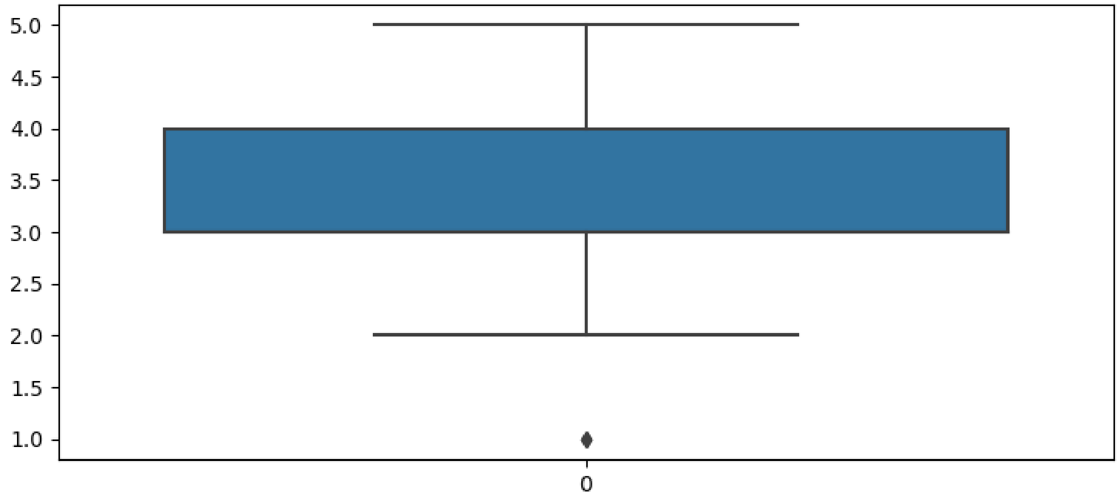
Calculation of outliers , upper/lower whisker , IQR for Fitness

In [235]:

```
plt.rcParams["figure.figsize"] = [7.50, 3.50]
plt.rcParams["figure.autolayout"] = True
sns.boxplot(data=aerofit["Fitness"])
```

Out[235]:

```
<Axes: >
```



In [239]:

```
p_25_Fitness = np.percentile(aerofit['Fitness'],25) ##25th percentile
p_50_Fitness = np.percentile(aerofit['Fitness'],50) ##50 percentile
p_75_Fitness = np.percentile(aerofit['Fitness'],75) ##75 percentile
IQR_Fitness = p_75_Fitness - p_25_Fitness
upper_Fitness =  p_75_Fitness + 1.5*IQR_Fitness
lower_Fitness = max(p_25_Fitness - 1.5*IQR_Fitness,0)
Fitness_outliers=aerofit.loc[(aerofit['Fitness'] > upper_Fitness) | (aerofit['Fitness']
percent_outliers_Fitness=(len(Fitness_outliers)/len(aerofit['Fitness']))*100
print(f"Upper Whisker: {upper_Fitness} \nLower Whisker: {lower_Fitness}\nIQR: {IQR_Fitne
```

```
Upper Whisker: 5.5
Lower Whisker: 1.5
IQR: 1.0
Outlier in Fitness:[1 1]
Number of Outliers in Usage:2
percent_ouliers_Usage:1.11
```

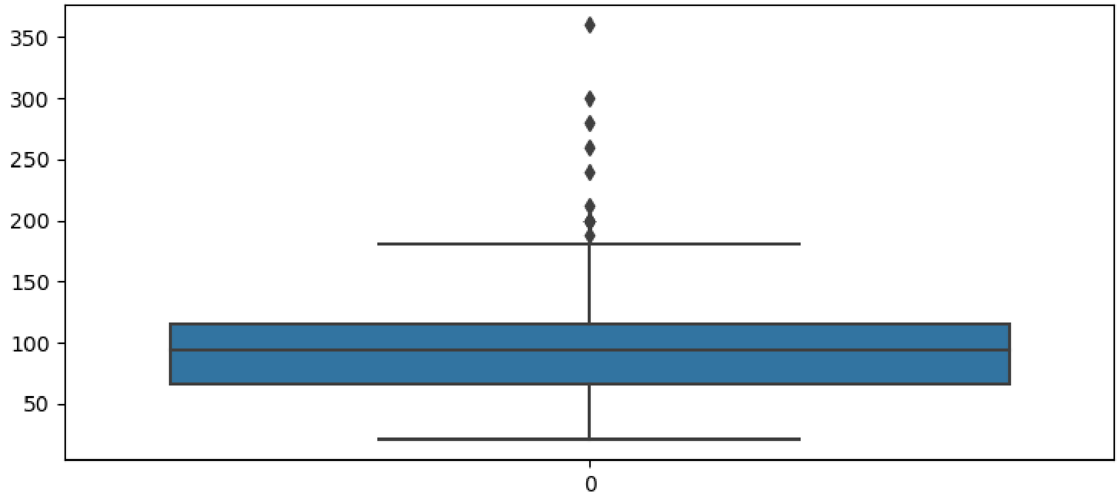
Calculation of outliers , upper/lower whisker , IQR for Miles

In [241]:

```
plt.rcParams["figure.figsize"] = [7.50, 3.50]
plt.rcParams["figure.autolayout"] = True
sns.boxplot(data=aerofit["Miles"])
```

Out[241]:

```
<Axes: >
```



In [248]:

```
p_25_Miles = np.percentile(aerofit['Miles'],25) ##25th percentile
p_50_Miles = np.percentile(aerofit['Miles'],50) ##50 percentile
p_75_Miles = np.percentile(aerofit['Miles'],75) ##75 percentile
IQR_Miles = p_75_Miles - p_25_Miles
upper_Miles =  p_75_Miles + 1.5*IQR_Miles
lower_Miles = max(p_25_Miles - 1.5*IQR_Miles,0)
Miles_outliers=aerofit.loc[(aerofit['Miles'] > upper_Miles) | (aerofit['Miles'] < lower_
percent_outliers_Miles=(len(Miles_outliers)/len(aerofit['Miles']))*100
print(f"Upper Whisker: {upper_Miles} \nLower Whisker: {lower_Miles}\nIQR: {IQR_Miles}\nO
```

```
Upper Whisker: 187.875
Lower Whisker: 0
IQR: 48.75
Outlier in Miles:[188 212 200 200 200 240 300 280 260 200 360 200 200]
Number of Outliers in Miles:13
percent_ouliers_Miles:7.22
```

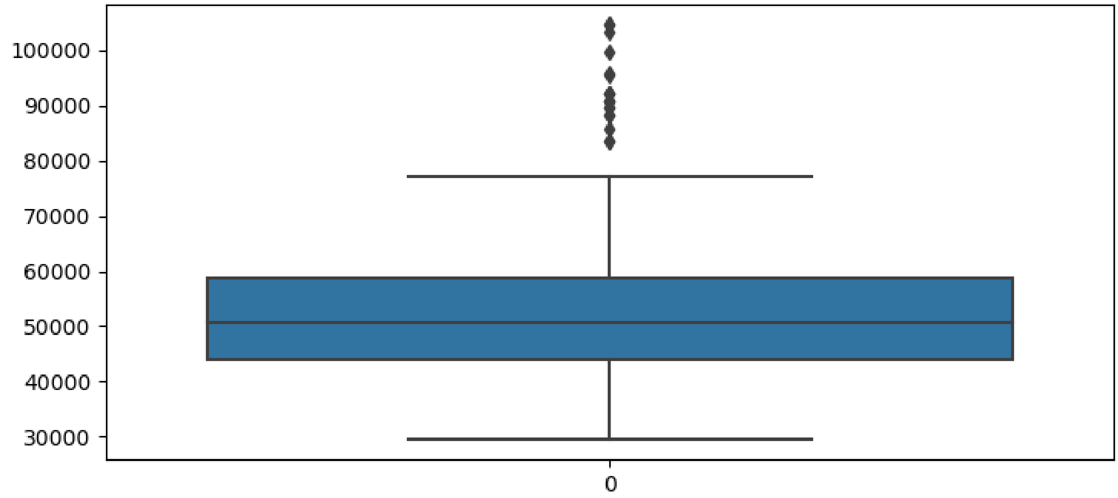
Calculation of outliers , upper/lower whisker , IQR for Income

In [244]:

```
plt.rcParams["figure.figsize"] = [7.50, 3.50]
plt.rcParams["figure.autolayout"] = True
sns.boxplot(data=aerofit["Income"])
```

Out[244]:

```
<Axes: >
```



In [249]:

```
p_25_Income = np.percentile(aerofit['Income'],25) ##25th percentile
p_50_Income = np.percentile(aerofit['Income'],50) ##50 percentile
p_75_Income = np.percentile(aerofit['Income'],75) ##75 percentile
IQR_Income = p_75_Income - p_25_Income
upper_Income =  p_75_Income + 1.5*IQR_Income
lower_Income = max(p_25_Income - 1.5*IQR_Income,0)
Income_outliers=aerofit.loc[(aerofit['Income'] > upper_Income) | (aerofit['Income'] < lo
percent_outliers_Income=(len(Income_outliers)/len(aerofit['Income']))*100
print(f"Upper Whisker: {upper_Income} \nLower Whisker: {lower_Income}\nIQR: {IQR_Income}
```

```
Upper Whisker: 80581.875
Lower Whisker: 22144.875
IQR: 14609.25
Outlier in Income:[ 83416  88396  90886  92131  88396  85906  90886 10333
6  99601  89641
  95866  92131  92131 104581  83416  89641  90886 104581  95508]
Number of Outliers in Income:19
percent_ouliers_Income:10.56
```

Calculation of Marginal Probablity

In [276]:

```
aerofit["Product"].value_counts()
```

Out[276]:

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

In [275]:

```
pd.crosstab(aerofit["Product"].value_counts().index,columns="percentage",values=(aerofit
```

Out[275]:

col_0	percentage
row_0	
KP281	44.444444
KP481	33.333333
KP781	22.222222

- KP281 is bought the most

Calculation of Conditional Probability

In [389]:

```
aerofit.head()
```

Out[389]:

	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

Probability of customer buying KP281, given that the cutomer is male

In [298]:

```
def p_prob_given_gender(gender, print_marginal=False):
    if gender!="Female" and gender!="Male":
        return "Invalid gender value."

    df = pd.crosstab(index=aerofit['Gender'], columns=[aerofit['Product']])
    p_781 = df['KP781'][gender] / df.loc[gender].sum() ## Probability of buying KP781 gi
    p_481 = df['KP481'][gender] / df.loc[gender].sum() ## Probability of buying KP481 gi
    p_281 = df['KP281'][gender] / df.loc[gender].sum() ## Probability of buying KP281 gi

    if print_marginal:
        print(f"P(Male): {df.loc['Male'].sum()/len(aerofit):.2f}")
        print(f"P(Female): {df.loc['Female'].sum()/len(aerofit):.2f}\n")

    print(f"P(KP781/{gender}): {p_781:.2f}")
    print(f"P(KP481/{gender}): {p_481:.2f}")
    print(f"P(KP281/{gender}): {p_281:.2f}\n")

p_prob_given_gender('Male', True)
p_prob_given_gender('Female')
```

```
P(Male): 0.58
P(Female): 0.42

P(KP781/Male): 0.32
P(KP481/Male): 0.30
P(KP281/Male): 0.38

P(KP781/Female): 0.09
P(KP481/Female): 0.38
P(KP281/Female): 0.53
```

- Amongst the treadmills, KP281 is slightly more preferrable in both males and females.
- Probability of female customer buying KP781 is very less compared to males , hence Males should be the target customer

Probability of each product given MaritalStatus

In [302]:

```
def p_prob_given_maritalstatus(MaritalStatus, print_marginal=False):
    if MaritalStatus!="Single" and MaritalStatus!="Partnered":
        return "Invalid Marital Status."

    df = pd.crosstab(index=aerofit['MaritalStatus'], columns=[aerofit['Product']])
    p_781 = df['KP781'][MaritalStatus] / df.loc[MaritalStatus].sum() ## Probability of b
    p_481 = df['KP481'][MaritalStatus] / df.loc[MaritalStatus].sum() ## Probability of b
    p_281 = df['KP281'][MaritalStatus] / df.loc[MaritalStatus].sum() ## Probability of b

    if print_marginal:
        print(f"P(Single): {df.loc['Single'].sum()/len(aerofit):.2f}")
        print(f"P(Partnered): {df.loc['Partnered'].sum()/len(aerofit):.2f}\n")

    print(f"P(KP781/{MaritalStatus}): {p_781:.2f}")
    print(f"P(KP481/{MaritalStatus}): {p_481:.2f}")
    print(f"P(KP281/{MaritalStatus}): {p_281:.2f}\n")

p_prob_given_maritalstatus('Single', True)
p_prob_given_maritalstatus('Partnered')
```

```
P(Single): 0.41
P(Partnered): 0.59

P(KP781/Single): 0.23
P(KP481/Single): 0.33
P(KP281/Single): 0.44

P(KP781/Partnered): 0.21
P(KP481/Partnered): 0.34
P(KP281/Partnered): 0.45
```

- Probalility of Partnered user buying KP281 is more than single user buying KP281
- However , probability of single users buying advanced treadmill KP781 is more than Partnered user buying KP781

In [291]:

```
aerofit.groupby(["Product","Gender"])["Age"].count()
```

Out[291]:

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

In [390]:

```
df1 = pd.crosstab(index=aerofit['Gender'], columns=[aerofit['Product']])
```

In [391]:

```
df1
```

Out[391]:

Product	KP281	KP481	KP781
Gender			
Female	40	29	7
Male	40	31	33

In [307]:

```
df = pd.crosstab(index=aerofit['Fitness'], columns=[aerofit['Product']],normalize=True,m
```

In [392]:

```
df
```

Out[392]:

Product	KP281	KP481	KP781	All
Fitness				
1	0.005556	0.005556	0.000000	0.011111
2	0.077778	0.066667	0.000000	0.144444
3	0.300000	0.216667	0.022222	0.538889
4	0.050000	0.044444	0.038889	0.133333
5	0.011111	0.000000	0.161111	0.172222
All	0.444444	0.333333	0.222222	1.000000

- Users with Fitness level 1 & 2 are very unlikely to buy KP781
- User with Fitness level 3 are morke likely to buy KP281, KP481.Users can be surveyed for their fitness level, these users with fitness level of 3 can be potential customers
- Users with fitness level 4 could be send more promotional offers to offer them buying KP781.Complementary Peronalised fitness plans or free e-consultation with Nutritionistcan be a good way to approach these customer
- Users with Fitness level 5 are more likely to buy KP781

Three Way Contingency Table

In [401]:

```
df_i=(aerofit["Income"]/100000).round(1) ##normalising the salary
```

In [402]:

```
pd.crosstab([aerofit.Product, aerofit.Fitness],df_i,normalize=True,margins=True)
```

Out[402]:

	Income	0.3	0.4	0.5	0.6	0.7	0.8	0.9	
Product	**Fitness**								
KP281	**1**	0.000000	0.005556	0.000000	0.000000	0.000000	0.000000	0.000000	0.0C
	2	0.005556	0.022222	0.038889	0.011111	0.000000	0.000000	0.000000	0.0C
	3	0.033333	0.088889	0.138889	0.027778	0.011111	0.000000	0.000000	0.0C
	4	0.005556	0.022222	0.011111	0.011111	0.000000	0.000000	0.000000	0.0C
	5	0.000000	0.005556	0.005556	0.000000	0.000000	0.000000	0.000000	0.0C
KP481	**1**	0.000000	0.000000	0.000000	0.000000	0.005556	0.000000	0.000000	0.0C
	2	0.011111	0.016667	0.038889	0.000000	0.000000	0.000000	0.000000	0.0C
	3	0.016667	0.022222	0.122222	0.050000	0.005556	0.000000	0.000000	0.0C
	4	0.005556	0.011111	0.022222	0.005556	0.000000	0.000000	0.000000	0.0C
KP781	**3**	0.000000	0.000000	0.005556	0.005556	0.005556	0.000000	0.005556	0.0C
	4	0.000000	0.000000	0.005556	0.011111	0.005556	0.000000	0.011111	0.0C
	5	0.000000	0.000000	0.038889	0.022222	0.005556	0.022222	0.044444	0.02
All		0.077778	0.194444	0.427778	0.144444	0.038889	0.022222	0.061111	0.03

- Highly earning people who are in excellent shape tend to buy KP781 more.
- Moderatley earning people who are in good/very good shape tend to buy KP481
- Low earning people who are in good shape tend to buy KP281