Aerofit Project

1.Problem Statement

The Goal of this project is to provide data driven insights so as to improve the descision making and to increase revenue and identify bottleneck by customer profiling.

In [61]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [62]:
```
df=pd.read_csv('C:/Users/rahul.kumar/Downloads/aerofit.csv')
```

In [63]:
```
df.head()
```

Out[63]:

	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 [64]:
```
df.shape
```

Out[64]:
```
(180, 9)
```

In [65]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 180 entries, 0 to 179
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   Product        180 non-null    object
 1   Age            180 non-null    int64 
 2   Gender         180 non-null    object
 3   Education      180 non-null    int64 
 4   MaritalStatus  180 non-null    object
 5   Usage          180 non-null    int64 
 6   Fitness        180 non-null    int64 
 7   Income         180 non-null    int64 
 8   Miles          180 non-null    int64 
dtypes: int64(6), object(3)
memory usage: 12.8+ KB
```

In [66]:
```
df.describe(include='all')
```

Out[66]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Incom
count	180	180.000000	180	180.000000	180	180.000000	180.000000	180.00000
unique	3	NaN	2	NaN	2	NaN	NaN	Na
top	KP281	NaN	Male	NaN	Partnered	NaN	NaN	Na
freq	80	NaN	104	NaN	107	NaN	NaN	Na
mean	NaN	28.788889	NaN	15.572222	NaN	3.455556	3.311111	53719.57777
std	NaN	6.943498	NaN	1.617055	NaN	1.084797	0.958869	16506.68422
min	NaN	18.000000	NaN	12.000000	NaN	2.000000	1.000000	29562.00000
25%	NaN	24.000000	NaN	14.000000	NaN	3.000000	3.000000	44058.75000
50%	NaN	26.000000	NaN	16.000000	NaN	3.000000	3.000000	50596.50000
75%	NaN	33.000000	NaN	16.000000	NaN	4.000000	4.000000	58668.00000
max	NaN	50.000000	NaN	21.000000	NaN	7.000000	5.000000	104581.00000

2. Non-Graphical Analysis

In [7]:
```
df.nunique()
```

Out[7]:
```
Product          3
Age             32
Gender           2
Education        8
MaritalStatus    2
Usage            6
Fitness          5
Income          62
Miles           37
dtype: int64
```

In [8]:
```
columns=list(df)
for i in columns:
    print(i)
    print(df[i].value_counts())
    print('.....
```

```
Product
KP281    80
KP481    60
KP781    40
Name: Product, dtype: int64
..........................................................................................
Age
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
..........................................................................................
Gender
Male      104
Female     76
Name: Gender, dtype: int64
..........................................................................................
Education
16    85
14    55
18    23
15     5
13     5
12     3
21     3
20     1
Name: Education, dtype: int64
..........................................................................................
MaritalStatus
Partnered    107
Single        73
```

```
Name: MaritalStatus, dtype: int64
..........................................................................................
Usage
3    69
4    52
2    33
5    17
6     7
7     2
Name: Usage, dtype: int64
..........................................................................................
Fitness
3    97
5    31
2    26
4    24
1     2
Name: Fitness, dtype: int64
..........................................................................................
Income
45480    14
52302     9
46617     8
54576     8
53439     8
         ..
65220     1
55713     1
68220     1
30699     1
95508     1
Name: Income, Length: 62, dtype: int64
..........................................................................................
Miles
85     27
95     12
66     10
75     10
47      9
106     9
94      8
113     8
53      7
100     7
180     6
200     6
56      6
64      6
127     5
160     5
42      4
150     4
38      3
74      3
170     3
120     3
103     3
132     2
141     2
280     1
```

```
260     1
300     1
240     1
112     1
212     1
80      1
140     1
21      1
169     1
188     1
360     1
Name: Miles, dtype: int64
```

In [9]:
```
Min_Age=df['Age'].min()
Min_Age
```

Out[9]:
```
18
```

In [10]:
```
Max_Age=df['Age'].max()
Max_Age
```

Out[10]:
```
50
```

In [11]:
```
Age_Range=df['Age'].max()-df['Age'].min()
Age_Range
```

Out[11]:
```
32
```

In [12]:
```
def func(x):
    if x<=34:
        return 'Young'
    else:
        return 'Middle Age'
```

In [13]:
```
df['Age_Category']=df['Age'].apply(func)
```

In [14]:
```
df['Age_Category'].value_counts()
```

Out[14]:
```
Young         144
Middle Age     36
Name: Age_Category, dtype: int64
```

In [15]:
```
df
```

Out[15]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles	Age_Category
0	KP281	18	Male	14	Single	3	4	29562	112	Young
1	KP281	19	Male	15	Single	2	3	31836	75	Young
2	KP281	19	Female	14	Partnered	4	3	30699	66	Young
3	KP281	19	Male	12	Single	3	3	32973	85	Young
4	KP281	20	Male	13	Partnered	4	2	35247	47	Young
...	...	...	...	...	...	...	...	...	...	...
175	KP781	40	Male	21	Single	6	5	83416	200	Middle Age
176	KP781	42	Male	18	Single	5	4	89641	200	Middle Age
177	KP781	45	Male	16	Single	5	5	90886	160	Middle Age
178	KP781	47	Male	18	Partnered	4	5	104581	120	Middle Age
179	KP781	48	Male	18	Partnered	4	5	95508	180	Middle Age

180 rows × 10 columns

In [16]:
```
Min_Income=df['Income'].min()
Min_Income
```

Out[16]:
```
29562
```

In [17]:
```
Max_Income=df['Income'].max()
Max_Income
```

Out[17]:
```
104581
```

In [18]:
```
Income_Range=Max_Income-Min_Income
Income_Range
```

Out[18]:
```
75019
```

In [19]:
```
def func_1(x):
    if x<=40000:
        return 'Low Income'
    elif x<=60000:
        return 'Midium Income'
    else:
        return 'High Income'
```

In [20]:
```
df['Income_Category']=df['Income'].apply(func_1)
```

In [21]:
```
df['Income_Category'].value_counts()
```

Out[21]:
```
Midium Income    106
High Income       42
Low Income        32
Name: Income_Category, dtype: int64
```

In [22]:
```
df
```

Out[22]:

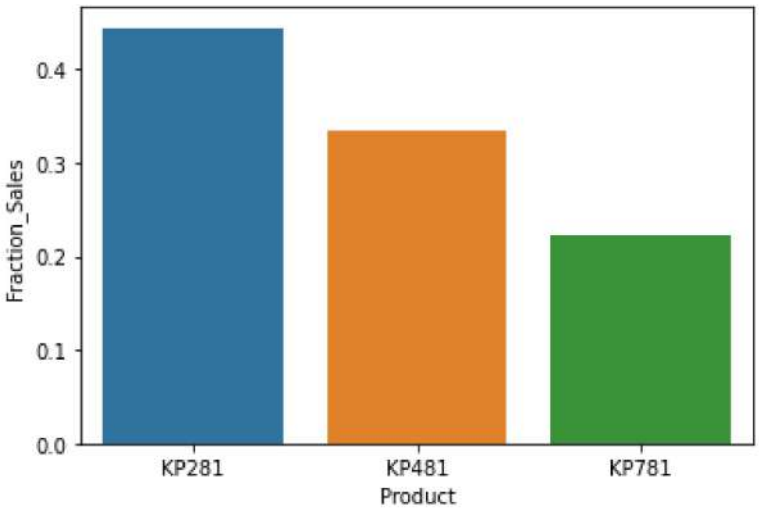
	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles	Age_Category
0	KP281	18	Male	14	Single	3	4	29562	112	Young
1	KP281	19	Male	15	Single	2	3	31836	75	Young
2	KP281	19	Female	14	Partnered	4	3	30699	66	Young
3	KP281	19	Male	12	Single	3	3	32973	85	Young
4	KP281	20	Male	13	Partnered	4	2	35247	47	Young
...	...	...	...	...	...	...	...	...	...	...
175	KP781	40	Male	21	Single	6	5	83416	200	Middle Age
176	KP781	42	Male	18	Single	5	4	89641	200	Middle Age
177	KP781	45	Male	16	Single	5	5	90886	160	Middle Age
178	KP781	47	Male	18	Partnered	4	5	104581	120	Middle Age
179	KP781	48	Male	18	Partnered	4	5	95508	180	Middle Age

180 rows × 11 columns

3.Visual Analysis

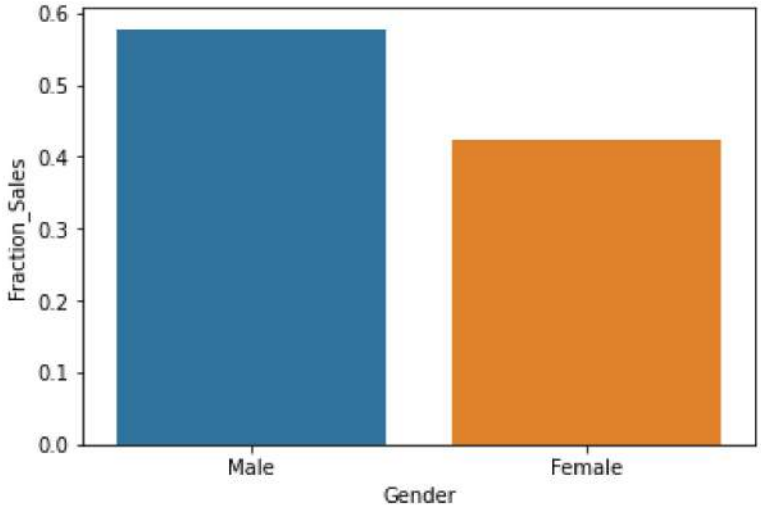
In [23]:
```
plot1 = sns.barplot(x=df['Product'].value_counts(normalize=True).index,y=df['Product']
plot1.set(xlabel ="Product", ylabel = "Fraction_Sales")
```

Out[23]:
```
[Text(0.5, 0, 'Product'), Text(0, 0.5, 'Fraction_Sales')]
```



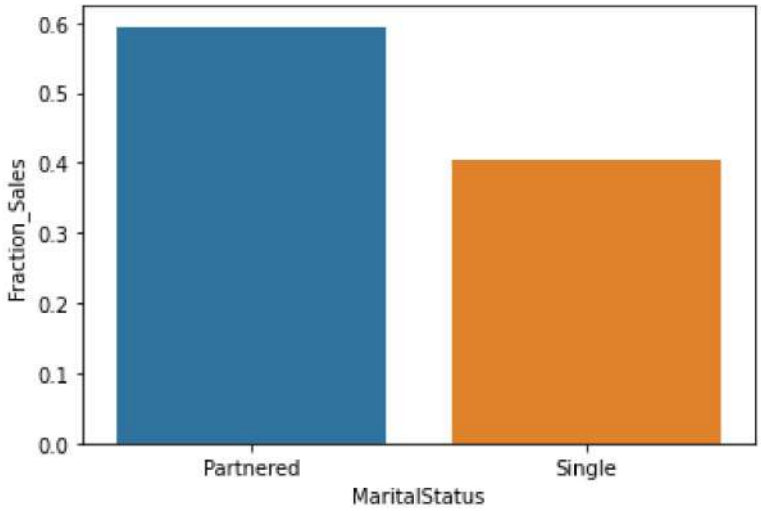
In [53]:
```
plot1 = sns.barplot(x=df['Gender'].value_counts(normalize=True).index,y=df['Gender'].v
plot1.set(xlabel ="Gender", ylabel = "Fraction_Sales")
```

Out[53]:
```
[Text(0.5, 0, 'Gender'), Text(0, 0.5, 'Fraction_Sales')]
```



In [54]:
```
plot1 = sns.barplot(x=df['MaritalStatus'].value_counts(normalize=True).index,y=df['Mar
plot1.set(xlabel ="MaritalStatus", ylabel = "Fraction_Sales")
```

Out[54]:
```
[Text(0.5, 0, 'MaritalStatus'), Text(0, 0.5, 'Fraction_Sales')]
```



In [55]:
```
df1 = df[['Product', 'Gender', 'MaritalStatus']].melt()
df1.groupby(['variable', 'value'])[['value']].count() / len(df)
```

Out[55]:

		value
variable	**value**	
Gender	**Female**	0.422222
	Male	0.577778
MaritalStatus	**Partnered**	0.594444
	Single	0.405556
Product	**KP281**	0.444444
	KP481	0.333333
	KP781	0.222222

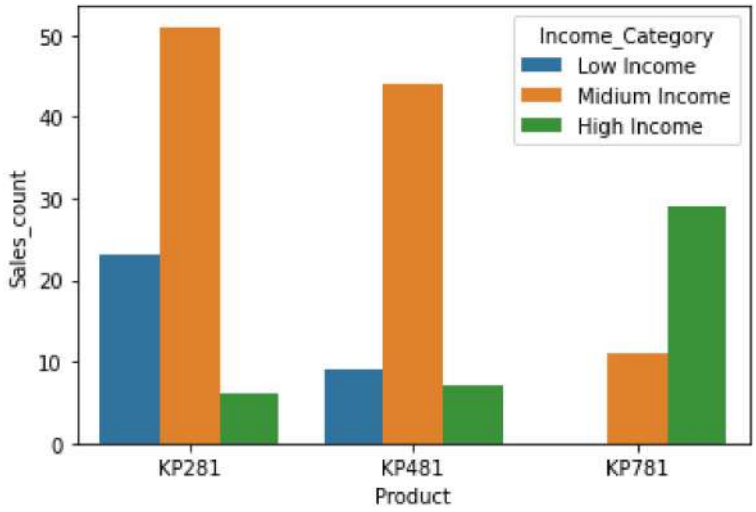
In [24]:
```
foo=df['Income_Category']
bar=df['Product']
pd.crosstab(foo,bar,margins=True)
```

Out[24]:

Income_Category	KP281	KP481	KP781	All
High Income	6	7	29	42
Low Income	23	9	0	32
Midium Income	51	44	11	106
All	80	60	40	180

In [25]:
```
plot1 = sns.countplot(x=df['Product'],hue=df['Income_Category'])
plot1.set(xlabel ="Product", ylabel = "Sales_count")
```

Out[25]:
```
[Text(0.5, 0, 'Product'), Text(0, 0.5, 'Sales_count')]
```



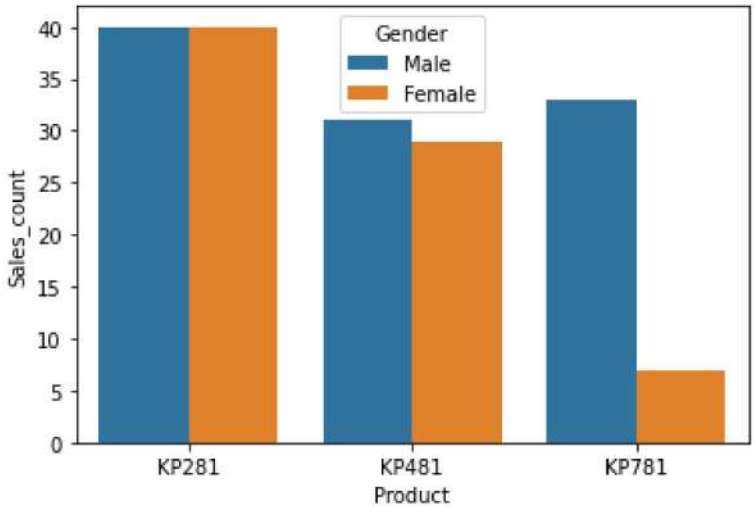
In [26]:
```
foo=df['Gender']
bar=df['Product']
pd.crosstab(foo,bar,margins=True)
```

Out[26]:

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

In [27]:
```
plot1 = sns.countplot(x=df['Product'],hue=df['Gender'])
plot1.set(xlabel ="Product", ylabel = "Sales_count")
```

Out[27]:
```
[Text(0.5, 0, 'Product'), Text(0, 0.5, 'Sales_count')]
```



In [57]:
```
def p_prod_given_gender(gender, print_marginal=False):
    if gender != "Female" and gender != "Male":
        return "Invalid gender value."

    df1 = pd.crosstab(index=df['Gender'], columns=[df['Product']])
    p_781 = df1['KP781'][gender] / df1.loc[gender].sum()
    p_481 = df1['KP481'][gender] / df1.loc[gender].sum()
    p_281 = df1['KP281'][gender] / df1.loc[gender].sum()

    if print_marginal:
        print(f"P(Male): {df1.loc['Male'].sum()/len(df):.2f}")
        print(f"P(Female): {df1.loc['Female'].sum()/len(df):.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_prod_given_gender('Male', True)
p_prod_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
```

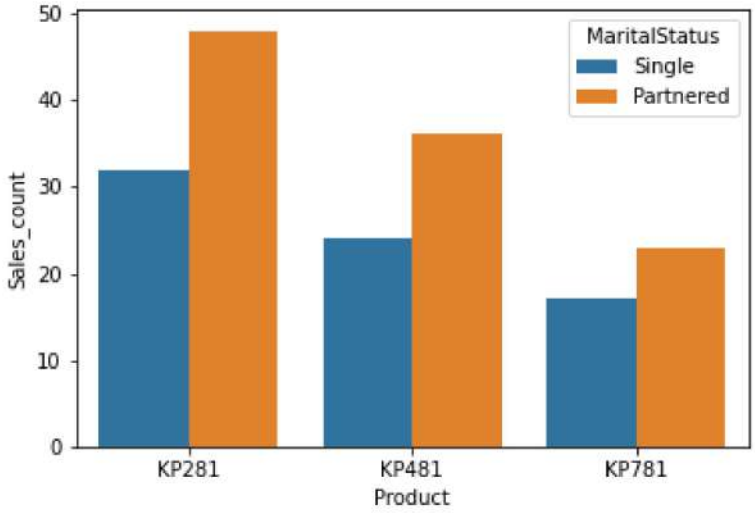
In [28]:
```
foo=df['MaritalStatus']
bar=df['Product']
pd.crosstab(foo,bar,margins=True)
```

Out[28]:

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

In [29]:
```
plot1 = sns.countplot(x=df['Product'],hue=df['MaritalStatus'])
plot1.set(xlabel ="Product", ylabel = "Sales_count")
```

Out[29]:
```
[Text(0.5, 0, 'Product'), Text(0, 0.5, 'Sales_count')]
```



In [58]:
```
def p_prod_given_mstatus(status, print_marginal=False):
    if status != "Single" and status != "Partnered":
        return "Invalid marital status value."

    df1 = pd.crosstab(index=df['MaritalStatus'], columns=[df['Product']])
    p_781 = df1['KP781'][status] / df1.loc[status].sum()
    p_481 = df1['KP481'][status] / df1.loc[status].sum()
    p_281 = df1['KP281'][status] / df1.loc[status].sum()

    if print_marginal:
        print(f"P(Single): {df1.loc['Single'].sum()/len(df):.2f}")
        print(f"P(Partnered): {df1.loc['Partnered'].sum()/len(df):.2f}\n")

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

p_prod_given_mstatus('Single', True)
p_prod_given_mstatus('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
```

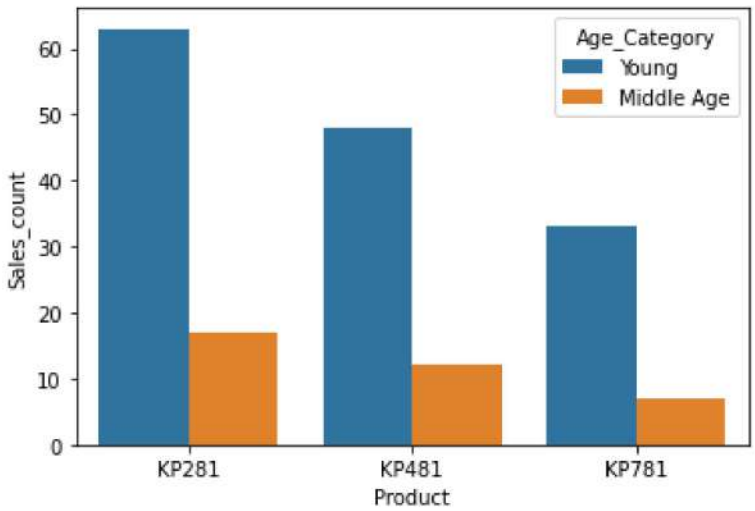
In [30]:
```
foo=df['Age_Category']
bar=df['Product']
pd.crosstab(foo,bar,margins=True)
```

Out[30]:

Age_Category	KP281	KP481	KP781	All
Middle Age	17	12	7	36
Young	63	48	33	144
All	80	60	40	180

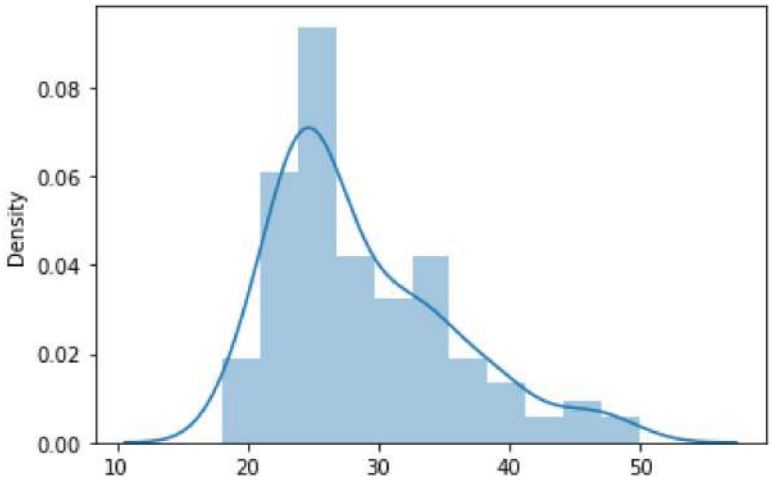
In [31]:
```
plot1 = sns.countplot(x=df['Product'],hue=df['Age_Category'])
plot1.set(xlabel ="Product", ylabel = "Sales_count")
```

Out[31]:
```
[Text(0.5, 0, 'Product'), Text(0, 0.5, 'Sales_count')]
```

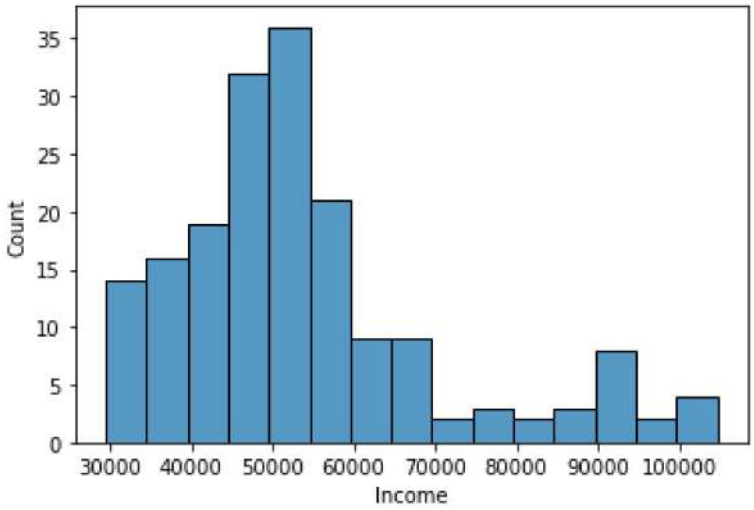


In [32]:
```
plot1=sns.distplot(x=df['Age'])
```

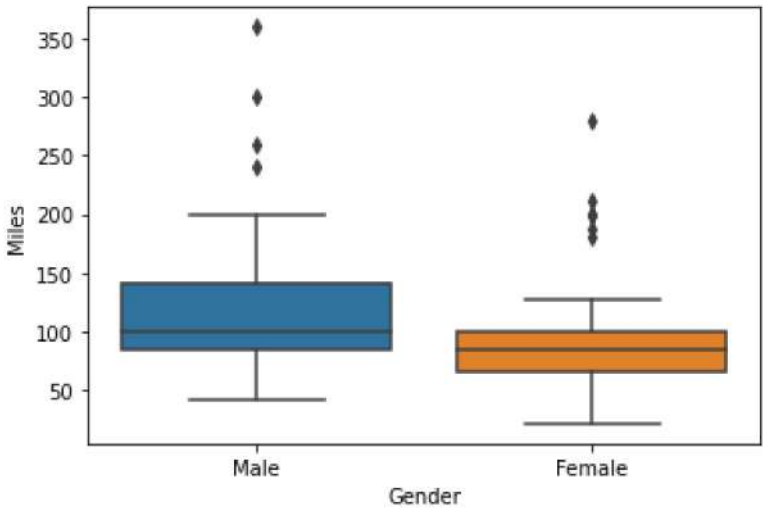
```
C:\Users\rahul.kumar\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



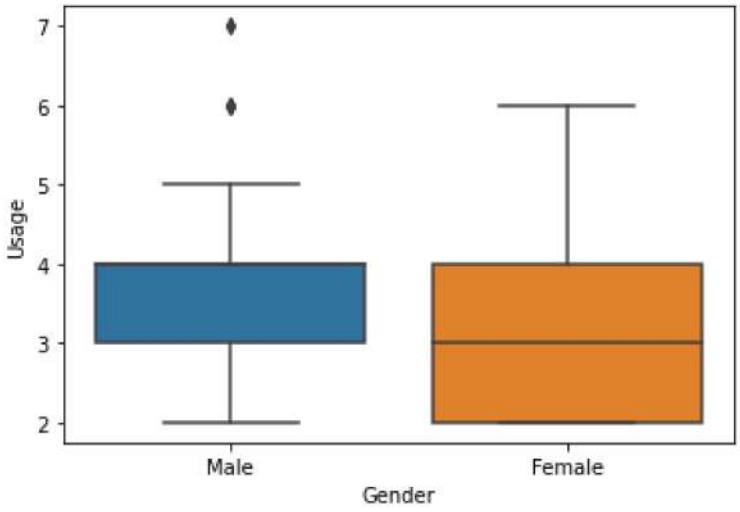
In [33]:
```
plot1=sns.histplot(x=df['Income'])
```



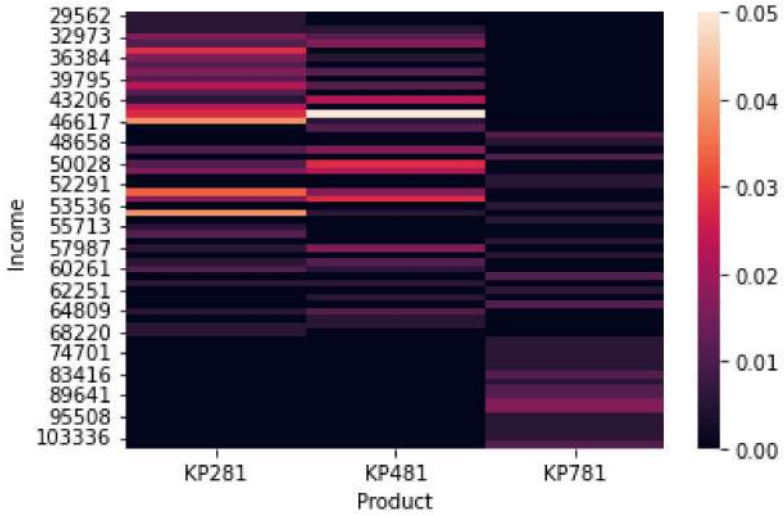
In [34]:
```
plot1=sns.boxplot(y=df['Miles'],x=df['Gender'])
```



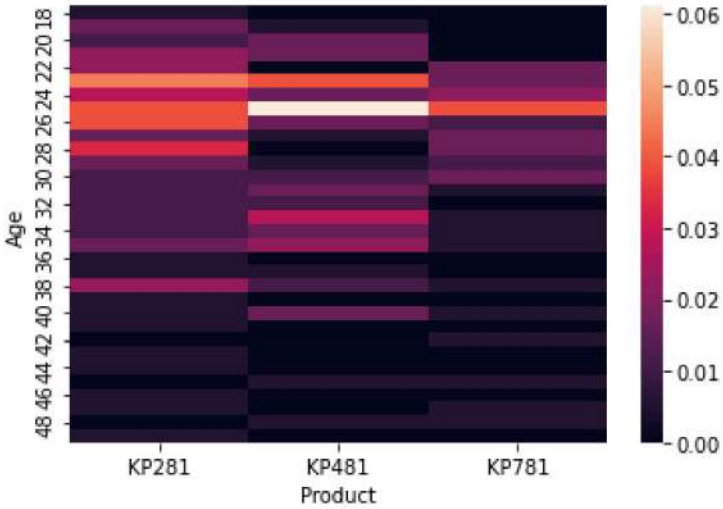
In [35]:
```
plot1=sns.boxplot(y=df['Usage'],x=df['Gender'])
```



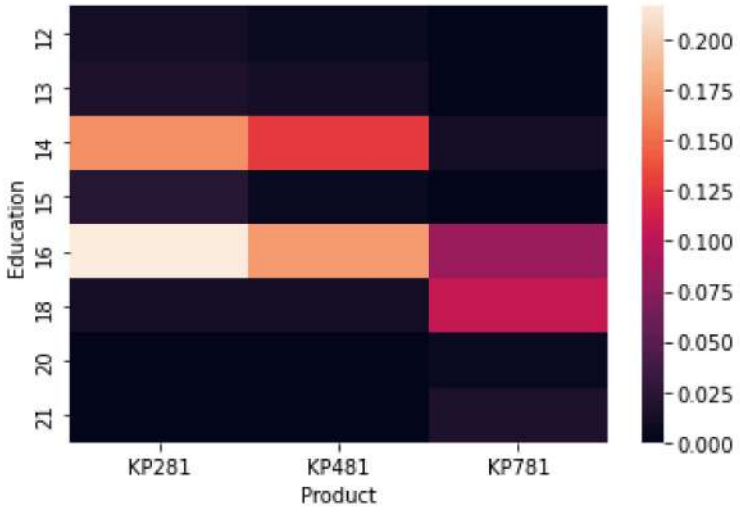
In [36]:
```
plot1=sns.heatmap(data=pd.crosstab(df['Income'],df['Product'],normalize=True))
```



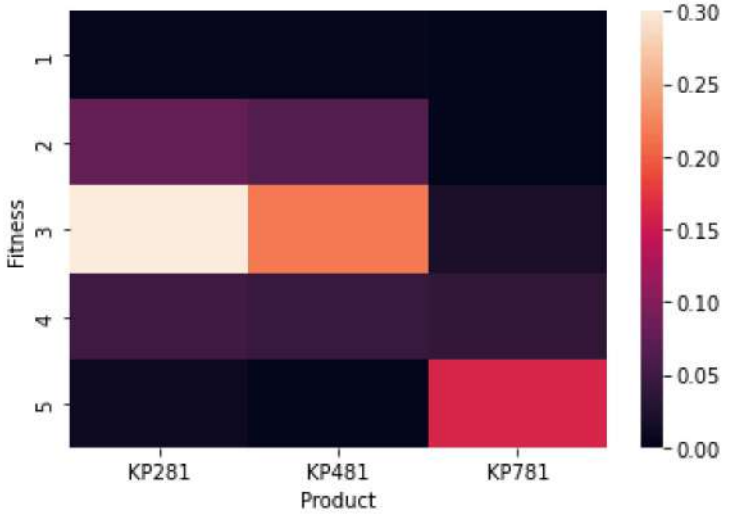
In [37]:
```
plot1=sns.heatmap(data=pd.crosstab(df['Age'],df['Product'],normalize=True))
```



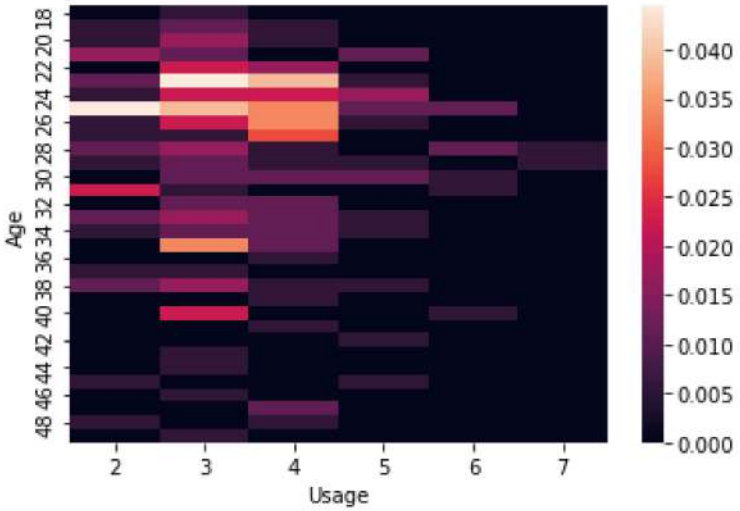
In [38]:
```
plot1=sns.heatmap(data=pd.crosstab(df['Education'],df['Product'],normalize=True))
```



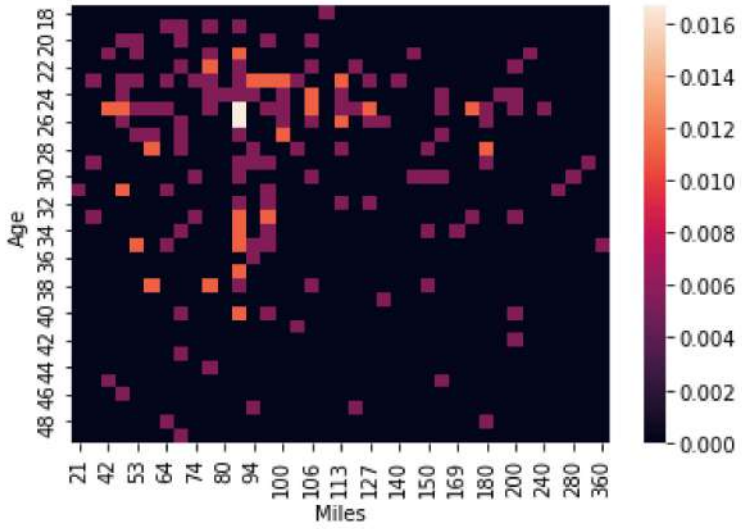
In [39]:
```
plot1=sns.heatmap(data=pd.crosstab(df['Fitness'],df['Product'],normalize=True))
```



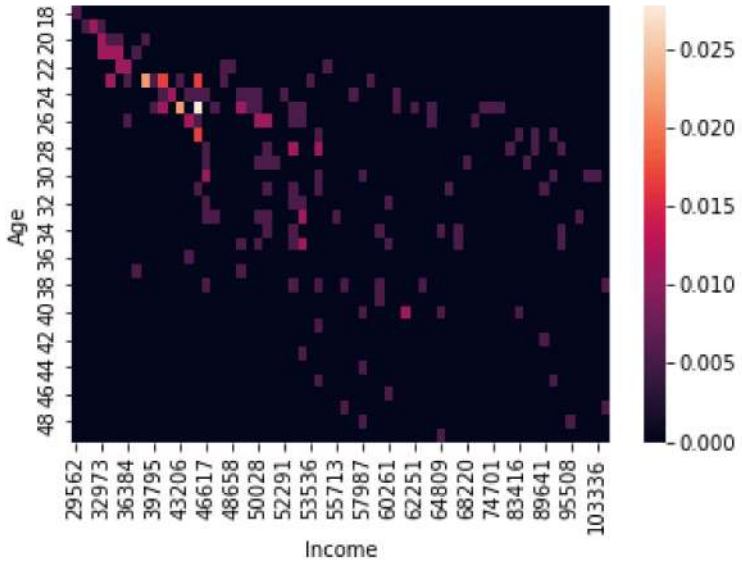
In [40]:
```
plot1=sns.heatmap(data=pd.crosstab(df['Age'],df['Usage'],normalize=True))
```



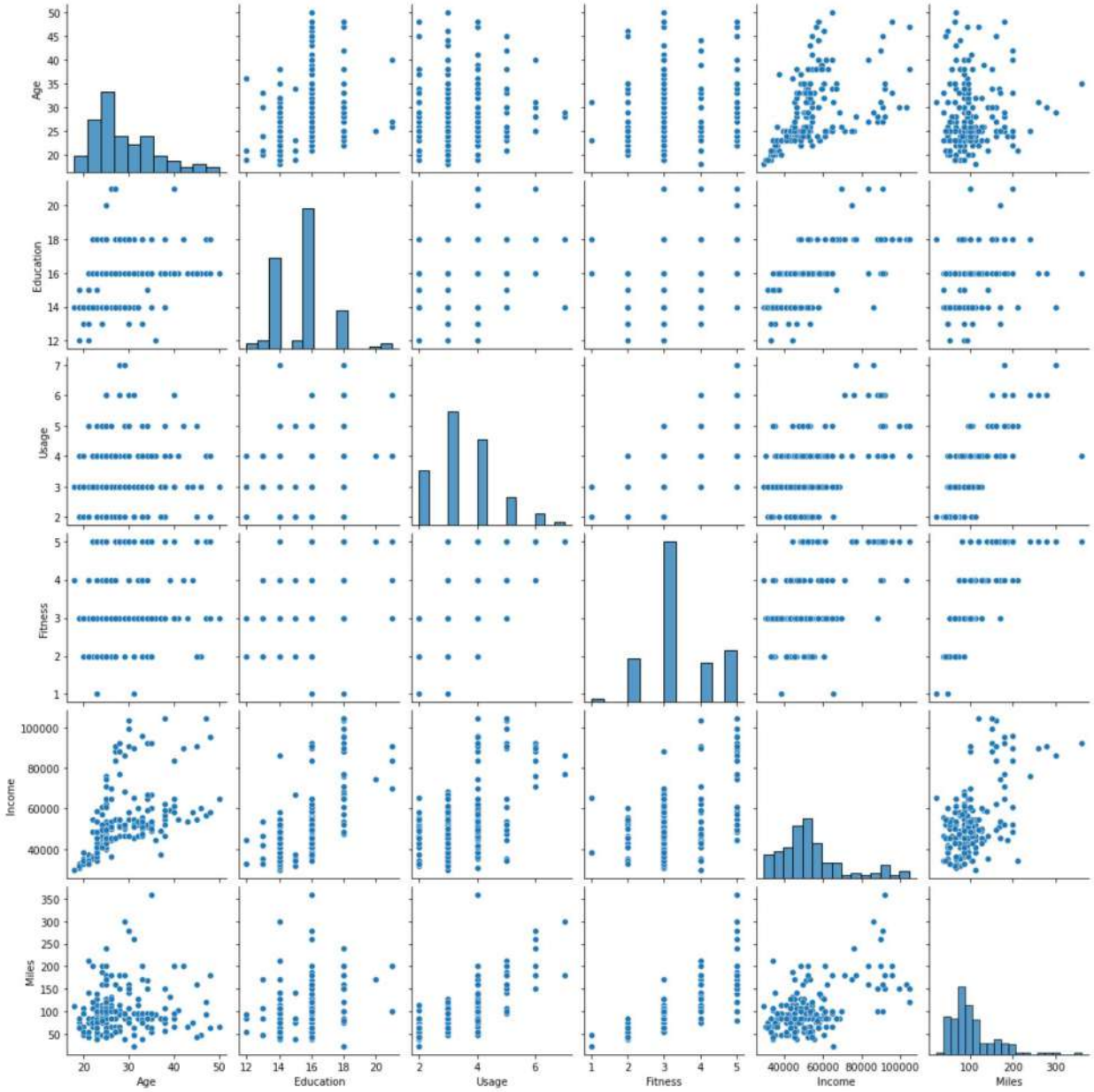
In [41]:
```
plot1=sns.heatmap(data=pd.crosstab(df['Age'],df['Miles'],normalize=True))
```



In [42]:
```
plot1=sns.heatmap(data=pd.crosstab(df['Age'],df['Income'],normalize=True))
```



In [43]:
```
plot1=sns.pairplot(data=df)
```



4.Missing Value & Outlier Detection

In [44]:
```
df.isna().sum()
```

Out[44]:
```
Product           0
Age               0
Gender            0
Education         0
MaritalStatus     0
Usage             0
Fitness           0
Income            0
Miles             0
Age_Category      0
Income_Category   0
dtype: int64
```

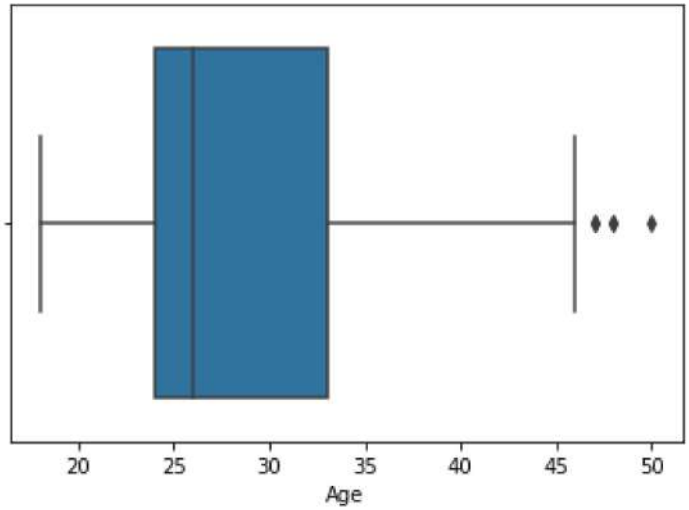
In [45]:
```
Age_25=df['Age'].quantile(0.25)
Age_75=df['Age'].quantile(0.75)
```

```
Age_IQR=Age_75-Age_25
LowerLimit_Age=max((Age_25-(Age_IQR)*1.5),df['Age'].min())
UpperLimit_Age=min((Age_75+(Age_IQR)*1.5),df['Age'].max())
Outliers_Age=df[(df['Age']<LowerLimit_Age) | (df['Age']>UpperLimit_Age)]['Age']
Outliers_Age.to_frame().sort_values(by=['Age']).reset_index(drop=True).rename(columns=
```

Out[45]:

	Outliers_Age
0	47
1	47
2	48
3	48
4	50

In [46]:
```
plot1=sns.boxplot(data=df,x=df['Age'])
```

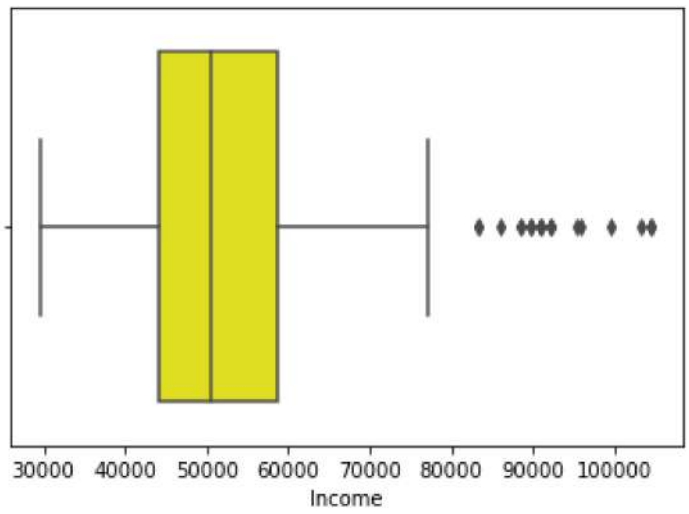


In [47]:
```
Income_25=df['Income'].quantile(0.25)
Income_75=df['Income'].quantile(0.75)
Income_IQR=Income_75-Income_25
LowerLimit_Income=max((Income_25-(Income_IQR)*1.5),df['Income'].min())
UpperLimit_Income=min((Income_75+(Income_IQR)*1.5),df['Income'].max())
Outliers_Income=df[(df['Income']<LowerLimit_Income) | (df['Income']>UpperLimit_Income)
Outliers_Income.to_frame().sort_values(by=['Income']).reset_index(drop=True).rename(co
```

Out[47]:

	Outliers_Income
0	83416
1	83416
2	85906
3	88396
4	88396
5	89641
6	89641
7	90886
8	90886
9	90886
10	92131
11	92131
12	92131
13	95508
14	95866
15	99601
16	103336
17	104581
18	104581

In [48]:
```
plot1=sns.boxplot(data=df,x=df['Income'],color='yellow')
```



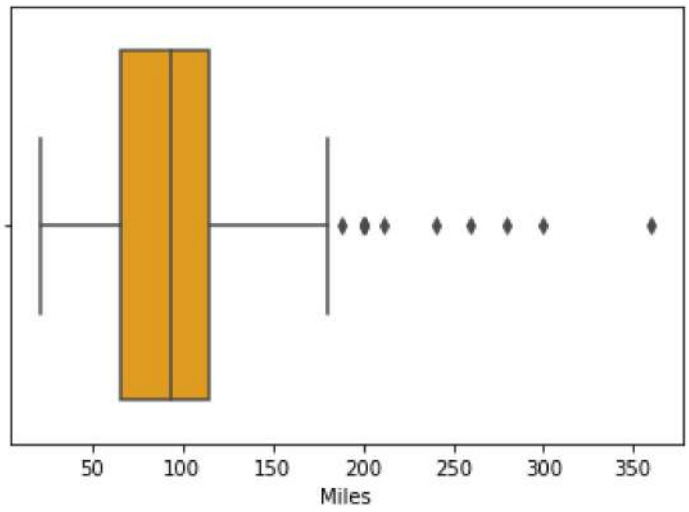
In [49]:
```
Miles_25=df['Miles'].quantile(0.25)
Miles_75=df['Miles'].quantile(0.75)
Miles_IQR=Miles_75-Miles_25
LowerLimit_Miles=max((Miles_25-(Miles_IQR)*1.5),df['Miles'].min())
```

```
UpperLimit_Miles=min((Miles_75+(Miles_IQR)*1.5),df['Miles'].max())
Outliers_Miles=df[(df['Miles']<LowerLimit_Miles) | (df['Miles']>UpperLimit_Miles)]['Mi
Outliers_Miles.to_frame().sort_values(by=['Miles']).reset_index(drop=True).rename(colu
```

Out[49]:

	Outliers_Miles
0	188
1	200
2	200
3	200
4	200
5	200
6	200
7	212
8	240
9	260
10	280
11	300
12	360

In [50]:
```
plot1=sns.boxplot(data=df,x=df['Miles'],color='orange')
```



5.Business Insights

1. Amongst the Age(in years) of range of 18 to 50. People around 24 years generally buy treadmill as they are having more energy and less responsibilities than Middle age people.
2. Most people are having income around 51000$ per Annum.
3. Amongst KP281,KP481 and KP781, KP281 is having maximum sales with 43% contiribution and next is KP481 with 33% contribution.

4. People with High Income prefers to buy KP781 and People with Medium-Less Income prefers to buy KP281 & KP481.
5. KP781 is less preferred by Females.
6. Young Age people tends to buy these equipments as compared to Miidle Age people.
7. Male population generally use treadmill more are compared to Females.
8. Usage of treadmill by Young aged people is more than Middle Aged People.
9. Age, Education and Income are the factors on which Sales depends
10. Customer who is Partnered, is more likely to purchase the product.

6.Recommendations

1. As most of the customers are around 24 years. we need to attract more customers having age more than 30 years by having casts in adds with age more than 30 years.
2. We need to add more variants of treadmills wich is affordable for people with income 50000$ per Annum.
3. We need to highlight KP781 in Top tier cities as it's for niche population.
4. As KP781 is less baught by Females. So, to aquire them we need to give special discounts to Female customers on KP781.
5. As Age, Education and Income are the factors on which Sales depends. So, we need to ask these from customers to pitch the variant which they are more likely to buy.

In [ ]: