```
import pandas as pd

data=pd.read_csv("https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/125/original/aerofit_treadmill.csv?1639992749")

# checking for null values & structure of the dataset will be analysed
data
```

	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.isnull().sum().sum()
```

```
0
```

```
data.duplicated().any()
```

```
False
```

```
data.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
```

```
data.shape
```

```
(180, 9)
```

```
data.describe()
```

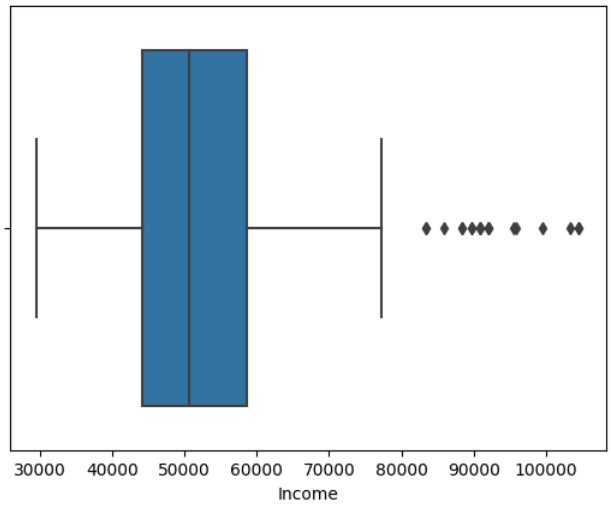
	Age	Education	Usage	Fitness	Income	Miles
count	180.000000	180.000000	180.000000	180.000000	180.000000	180.000000
mean	28.788889	15.572222	3.455556	3.311111	53719.577778	103.194444
std	6.943498	1.617055	1.084797	0.958869	16506.684226	51.863605
min	18.000000	12.000000	2.000000	1.000000	29562.000000	21.000000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

```
# outliers will be detected using interquartile range and visualized using boxplot
data.columns
```

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

```
data.head()
import seaborn as sns
sns.boxplot(x=data['Income'])
```

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




```
# finding inter quartile range
q3=data['Income'].quantile(0.75)
q1=data['Income'].quantile(0.25)
iqr=q3-q1
```

```
iqr
```

```
14609.25
```

```
upper_limit=data['Income'].quantile(0.75)+iqr
upper_limit
```

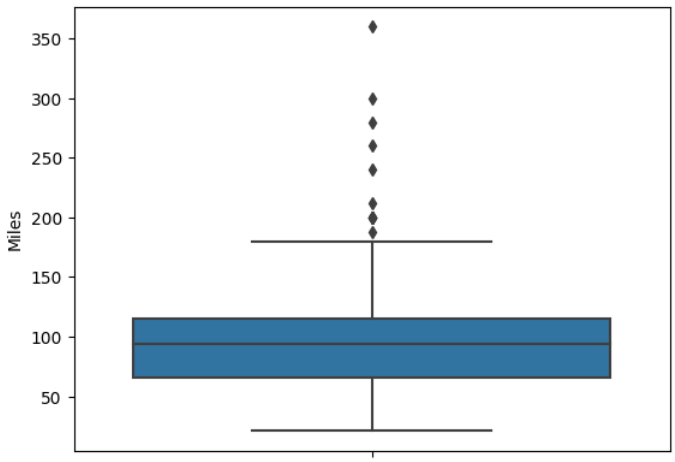
```
73277.25
```

```
data[data['Income']>upper_limit]
```

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
155	KP781	25	Male	18	Partnered	6	5	75946	240
156	KP781	25	Male	20	Partnered	4	5	74701	170
159	KP781	27	Male	16	Partnered	4	5	83416	160
160	KP781	27	Male	18	Single	4	3	88396	100
161	KP781	27	Male	21	Partnered	4	4	90886	100
162	KP781	28	Female	18	Partnered	6	5	92131	180
163	KP781	28	Male	18	Partnered	7	5	77191	180
164	KP781	28	Male	18	Single	6	5	88396	150
166	KP781	29	Male	14	Partnered	7	5	85906	300
167	KP781	30	Female	16	Partnered	6	5	90886	280
168	KP781	30	Male	18	Partnered	5	4	103336	160
169	KP781	30	Male	18	Partnered	5	5	99601	150
170	KP781	31	Male	16	Partnered	6	5	89641	260
171	KP781	33	Female	18	Partnered	4	5	95866	200
172	KP781	34	Male	16	Single	5	5	92131	150
173	KP781	35	Male	16	Partnered	4	5	92131	360
174	KP781	38	Male	18	Partnered	5	5	104581	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
178	KP781	47	Male	18	Partnered	4	5	104581	120
179	KP781	48	Male	18	Partnered	4	5	95508	180

```
# outliers detection using boxplot of column Miles
sns.boxplot(y=data['Miles'])
```

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




```
iqr1=data['Miles'].quantile(0.75)-data['Miles'].quantile(0.25)
upper_limit1=data['Miles'].quantile(0.75)+iqr1
```

```
66.0
```

```
iqr1
```

```
48.75
```

```
upper_limit1
```

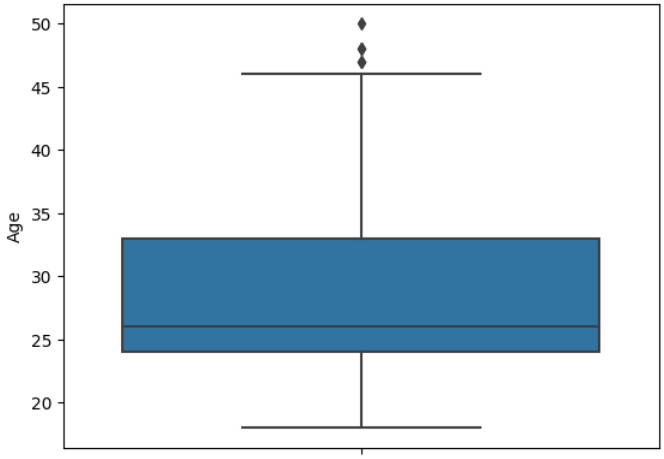
```
163.5
```

```
# the records greater than this upper limit are considered as an outliers
outliers1=data[data['Miles']>upper_limit1]
outliers1
```

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
23	KP281	24	Female	16	Partnered	5	5	44343	188
61	KP281	34	Male	16	Single	4	5	51165	169
84	KP481	21	Female	14	Partnered	5	4	34110	212
103	KP481	25	Male	14	Partnered	4	3	45480	170
120	KP481	33	Male	13	Partnered	4	4	53439	170
142	KP781	22	Male	18	Single	4	5	48556	200
148	KP781	24	Female	16	Single	5	5	52291	200
152	KP781	25	Female	18	Partnered	5	5	61006	200
154	KP781	25	Male	18	Partnered	6	4	70966	180
155	KP781	25	Male	18	Partnered	6	5	75946	240
156	KP781	25	Male	20	Partnered	4	5	74701	170
158	KP781	26	Male	16	Partnered	5	4	64741	180
162	KP781	28	Female	18	Partnered	6	5	92131	180
163	KP781	28	Male	18	Partnered	7	5	77191	180
165	KP781	29	Male	18	Single	5	5	52290	180
166	KP781	29	Male	14	Partnered	7	5	85906	300
167	KP781	30	Female	16	Partnered	6	5	90886	280
170	KP781	31	Male	16	Partnered	6	5	89641	260
171	KP781	33	Female	18	Partnered	4	5	95866	200
173	KP781	35	Male	16	Partnered	4	5	92131	360
175	KP781	40	Male	21	Single	6	5	83416	200
176	KP781	42	Male	18	Single	5	4	89641	200
179	KP781	48	Male	18	Partnered	4	5	95508	180

```
# outliers detection based on age
sns.boxplot(y=data['Age'])
```

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




```
iqr3=data['Age'].quantile(0.75)-data['Age'].quantile(0.25)
iqr3
```

```
9.0
```

```
upper_limit3=data['Age'].quantile(0.75)+iqr3
upper_limit3
```

```
42.0
```

```
# the records greater than that of upperlimit is considered as an outliers
outliers4=data[data['Age']>upper_limit3]
```

```
outliers4
```

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
75	KP281	43	Male	16	Partnered	3	3	53439	66
76	KP281	44	Female	16	Single	3	4	57987	75
77	KP281	46	Female	16	Partnered	3	2	60261	47
78	KP281	47	Male	16	Partnered	4	3	56850	94
79	KP281	50	Female	16	Partnered	3	3	64809	66
138	KP481	45	Male	16	Partnered	2	2	54576	42
139	KP481	48	Male	16	Partnered	2	3	57987	64
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

```
# there is no need to calculate lower_limit because below lowerlimit there are no o
```

```
data[data['Income']<upper_limit]
```

	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
...	...	...	...	...	...	...	...	...	...
153	KP781	25	Male	18	Partnered	4	3	64741	100
154	KP781	25	Male	18	Partnered	6	4	70966	180
157	KP781	26	Female	21	Single	4	3	69721	100
158	KP781	26	Male	16	Partnered	5	4	64741	180
165	KP781	29	Male	18	Single	5	5	52290	180

158 rows × 9 columns

```
# finding difference between mean and median
mean=(data['Income'].mean())
median=(data['Income'].median())
```

```
mean
```

```
53719.57777777778
```

```
median
```

```
50596.5
```

```
diff= mean-median
diff
# here we can observe clearly that mean is greater than median so thats why mean is shifted right towards the upper side
# here upperside contains outliers
```

```
3123.0777777777766
```

```
# check if features like martial status,age have any effect on the products purchased
data
```

	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

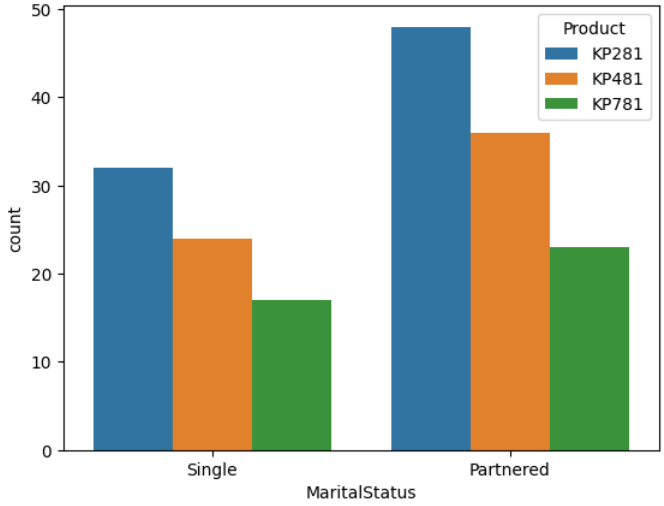
180 rows × 9 columns

```
# details of the countplot
pd.crosstab(data['MaritalStatus'],data['Product'],margins=True)
```

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

```
sns.countplot(x='MaritalStatus',hue='Product',data=data)
```

```
<Axes: xlabel='MaritalStatus', ylabel='count'>
```

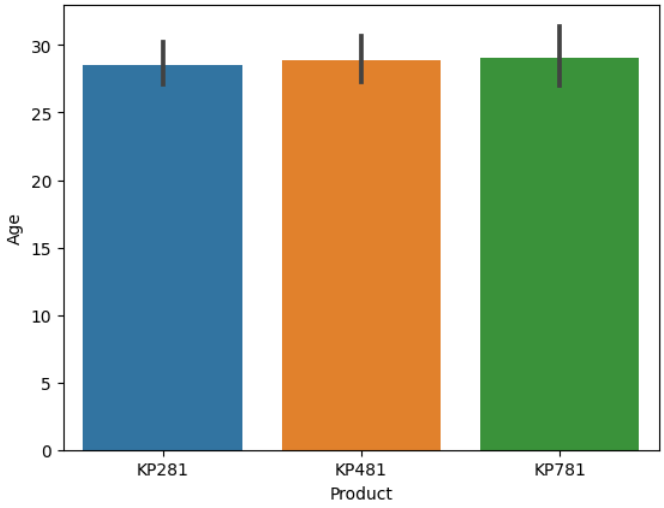



```
# analysis based on age  using barplot
data.columns
```

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

```
sns.barplot(x='Product',y='Age',data=data)
```

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

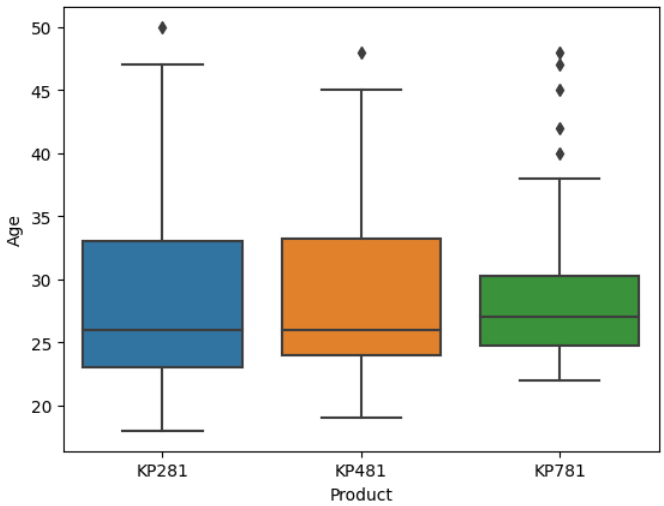



```
pd.crosstab(data['Age'],data['Product'],margins=True)
```

Product	KP281	KP481	KP781	All
Age				
18	1	0	0	1
19	3	1	0	4
20	2	3	0	5
21	4	3	0	7
22	4	0	3	7
23	8	7	3	18
24	5	3	4	12
25	7	11	7	25
26	7	3	2	12
27	3	1	3	7
28	6	0	3	9
29	3	1	2	6
30	2	2	3	7
31	2	3	1	6
32	2	2	0	4
33	2	5	1	8
34	2	3	1	6
35	3	4	1	8
36	1	0	0	1
37	1	1	0	2
38	4	2	1	7
39	1	0	0	1
40	1	3	1	5
41	1	0	0	1
42	0	0	1	1
43	1	0	0	1
44	1	0	0	1
45	0	1	1	2
46	1	0	0	1
47	1	0	1	2
48	0	1	1	2
50	1	0	0	1
All	80	60	40	180

```
# analysis of products purchased based on age using barplot
sns.boxplot(x='Product',y='Age',data=data)
```

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

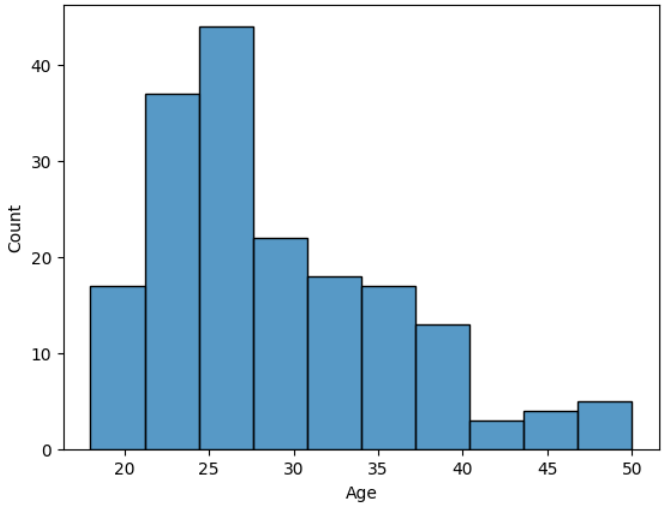


```
# for detailed understanding
p=data[data['Product']=='KP281']
p[['Product','Age']].value_counts()
# in above boxplot width of boxplot is more at the people age between 23 to 32 age people has buyed more no  of tredmills  of "kp281"
```

```
Product  Age
KP281    23     8
         25     7
         26     7
         28     6
         24     5
         38     4
         21     4
         22     4
         29     3
         19     3
         27     3
         35     3
         34     2
         33     2
         32     2
         31     2
         30     2
         20     2
         41     1
         47     1
         46     1
         44     1
         43     1
         18     1
         40     1
         39     1
         37     1
         36     1
         50     1
dtype: int64
```

```
data
# we are going to visualize using histplots
sns.histplot(data['Age'],bins=10)
```

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




```
# representing marginal probability like what percent of customers purchased "KP281,KP481,or KP781" in a table
```

```
data.columns
```

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

```
pd.crosstab(data['Product'],[data['Age'],data['Gender']],margins=True,normalize='index')
```

Age	18		19		20		21		22		23		...	40		41		42	43
Gender	Male	Female	Male	Female	Male	Female	Male	Female	Male	Female	Male	Female	...	Male		Male		Male	Male
Product																			
KP281	0.012500	0.012500	0.025000	0.012500	0.012500	0.025000	0.025000	0.037500	0.012500	0.037500			...	0.012500		0.012500		0.000000	0.012500
KP481	0.000000	0.000000	0.016667	0.016667	0.033333	0.016667	0.033333	0.000000	0.000000	0.050000			...	0.016667		0.000000		0.000000	0.000000
KP781	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.075000	0.025000			...	0.025000		0.000000		0.025000	0.000000
All	0.005556	0.005556	0.016667	0.011111	0.016667	0.016667	0.022222	0.016667	0.022222	0.038889			...	0.016667		0.005556		0.005556	0.005556

4 rows × 51 columns

```
# already with respect to information of product buyed by the customer like KP281,KP481, ETC
#we are going to find out the probabibilty that the person  age,gender,maritalstatus should be there
#age
```

```
pd.crosstab(data['Product'],[data['Age'],data['Gender'],data['MaritalStatus']],margins=True,normalize='index')
```

Age	18	19		20			21			22	...	41	42	43
Gender	Male	Female	Male	Female	Male		Female	Male		Female	...	Male	Male	Mal
MaritalStatus	Single	Partnered	Single	Partnered	Partnered	Single	Partnered	Partnered	Single	Partnered	...	Partnered	Single	Par
Product														
KP281	0.012500	0.012500	0.025000	0.012500	0.012500	0.000000	0.025000	0.000000	0.025000	0.012500	...	0.012500	0.000000	0.
KP481	0.000000	0.000000	0.016667	0.016667	0.000000	0.033333	0.016667	0.033333	0.000000	0.000000	...	0.000000	0.000000	0.
KP781	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.025000	0.
All	0.005556	0.005556	0.016667	0.011111	0.005556	0.011111	0.016667	0.011111	0.011111	0.005556	...	0.005556	0.005556	0.

4 rows × 77 columns

```
# conditional probability the person having age,gender and maritalstatus information  based on
#it what is the probability that the person is buying the product Kp281,kp481,kp781 etc
```

```
pd.crosstab(data['Product'],[data['Age'],data['Gender'],data['MaritalStatus']],margins=True,normalize='columns')
```

Age	18	19		20			21			22	...	42	43	44	4
Gender	Male	Female	Male	Female	Male		Female	Male		Female	...	Male	Male	Female	Ma
MaritalStatus	Single	Partnered	Single	Partnered	Partnered	Single	Partnered	Partnered	Single	Partnered	...	Single	Partnered	Single	Pa
Product															
KP281	1.0	1.0	0.666667	0.5	1.0	0.0	0.666667	0.0	1.0	1.0	...	0.0	1.0	1.0	
KP481	0.0	0.0	0.333333	0.5	0.0	1.0	0.333333	1.0	0.0	0.0	...	0.0	0.0	0.0	
KP781	0.0	0.0	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	...	1.0	0.0	0.0	

3 rows × 78 columns

```
# corelations using heatmaps
data.corr()
```

```
<ipython-input-25-2c9133e8ca02>:2: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will d
  data.corr()
```

	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

```
# heatmaps using pearson corelation concept
sns.heatmap(data.corr(),annot=True)
```

```
<ipython-input-27-6c71ac866e2e>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will d
  sns.heatmap(data.corr(),annot=True)
<Axes: >
```