```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import norm
```

```
df=pd.read_csv("/content/drive/MyDrive/dataset/Aerofit.csv")
```

```
df
```

	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

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

```
df.shape
```

```
(180, 9)
```

```
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
```

```
df.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

```
df.mean()
```

```
<ipython-input-44-c61f0c8f89b5>:1: FutureWarning: The default value of numeric_only in DataFrame.mean is deprecated. In a future ve
  df.mean()
Age             28.788889
Education       15.572222
Fitness          3.311111
Income       53719.577778
dtype: float64
```

```
df.median()
```

```
<ipython-input-46-6d467abf240d>:1: FutureWarning: The default value of numeric_only in DataFrame.median is deprecated. In a future
  df.median()
Age             26.0
Education       16.0
Fitness          3.0
Income       50596.5
dtype: float64
```

```
df.mean()-df.median()
```

```
<ipython-input-49-b31f8f474e96>:1: FutureWarning: The default value of numeric_only in DataFrame.mean is deprecated. In a future ve
  df.mean()-df.median()
<ipython-input-49-b31f8f474e96>:1: FutureWarning: The default value of numeric_only in DataFrame.median is deprecated. In a future
  df.mean()-df.median()
Age             2.788889
Education      -0.427778
Fitness         0.311111
Income       3123.077778
dtype: float64
```

```
df.describe(include="object")
```

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

```
df.describe(include="all")
```

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
count	180	180.000000	180	180.000000	180	180.000000	180.000000	180.000000	180.000000
unique	3	NaN	2	NaN	2	NaN	NaN	NaN	NaN
top	KP281	NaN	Male	NaN	Partnered	NaN	NaN	NaN	NaN
freq	80	NaN	104	NaN	107	NaN	NaN	NaN	NaN
mean	NaN	28.788889	NaN	15.572222	NaN	3.455556	3.311111	53719.577778	103.194444
std	NaN	6.943498	NaN	1.617055	NaN	1.084797	0.958869	16506.684226	51.863605
min	NaN	18.000000	NaN	12.000000	NaN	2.000000	1.000000	29562.000000	21.000000
25%	NaN	24.000000	NaN	14.000000	NaN	3.000000	3.000000	44058.750000	66.000000
50%	NaN	26.000000	NaN	16.000000	NaN	3.000000	3.000000	50596.500000	94.000000
75%	NaN	33.000000	NaN	16.000000	NaN	4.000000	4.000000	58668.000000	114.750000
max	NaN	50.000000	NaN	21.000000	NaN	7.000000	5.000000	104581.000000	360.000000

Non-Graphical Analysis: Value counts and unique attributes

```
df["Product"].unique()
```

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

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

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

```
df["Age"].unique()
```

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

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

```
df["Gender"].unique()
```

```
array(['Male', 'Female'], dtype=object)
```

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

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

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

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

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

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

```
df["MaritalStatus"].unique()
```

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

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

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

```
df["Usage"].unique()
```

```
array([3, 2, 4, 5, 6, 7])
```

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

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

```
df["Fitness"].unique()
```

```
array([4, 3, 2, 1, 5])
```

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

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

```
df["Income"].unique()
```

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

```
df["Income"].value_counts()
```

```
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
```

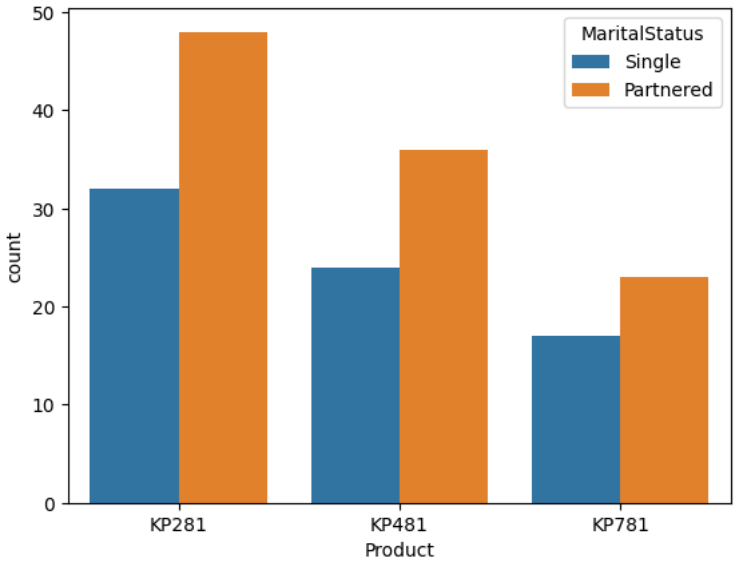
```
df["Miles"].unique()
```

```
array([112,  75,  66,  85,  47, 141, 103,  94, 113,  38, 188,  56, 132,
       169,  64,  53, 106,  95, 212,  42, 127,  74, 170,  21, 120, 200,
       140, 100,  80, 160, 180, 240, 150, 300, 280, 260, 360])
```

```
df["Miles"].value_counts()
```

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

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

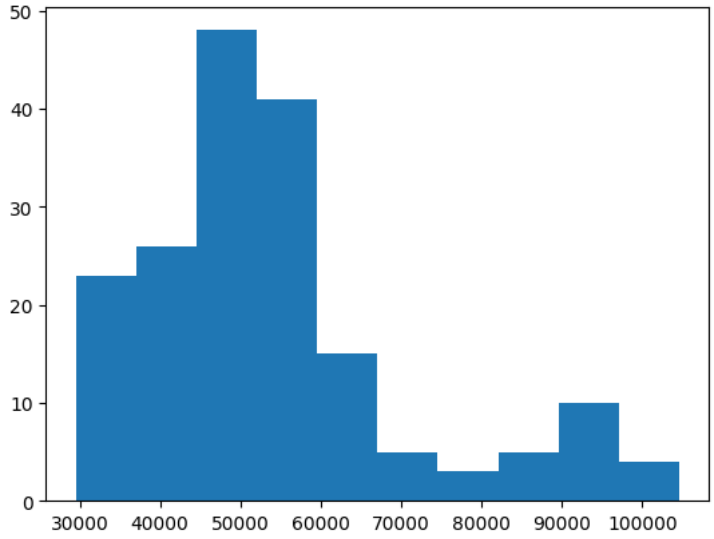



```
sns.boxplot(data=df,x="Age",y="Product")
```

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

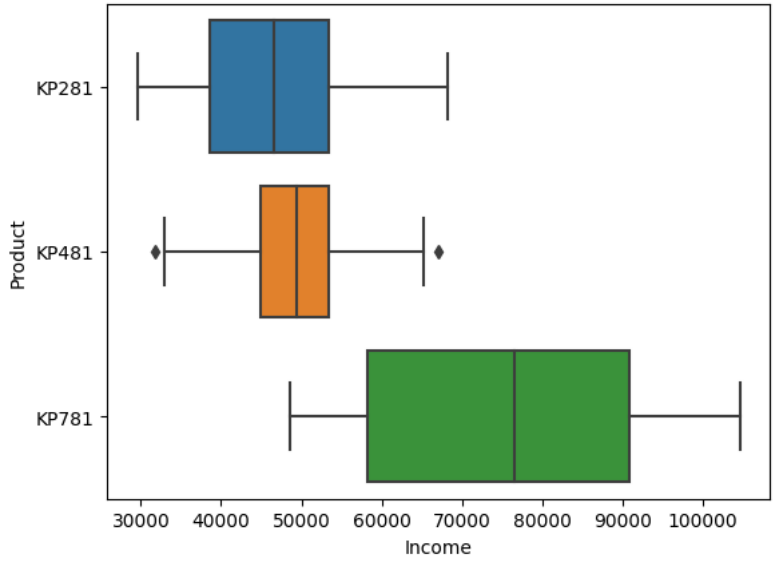

```
plt.hist(df["Income"])
```

```
(array([23., 26., 48., 41., 15.,  5.,  3.,  5., 10.,  4.]),
 array([ 29562. ,  37063.9,  44565.8,  52067.7,  59569.6,  67071.5,
         74573.4,  82075.3,  89577.2,  97079.1, 104581. ]),
 <BarContainer object of 10 artists>)
```




```
sns.boxplot(data=df,x='Income',y='Product')
```

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




```
sns.pairplot(data=df,hue='Product')
```

```
<seaborn.axisgrid.PairGrid at 0x79144f373fa0>
```


```
df.corr()
```

```
<ipython-input-32-2f6f6606aa2c>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a f
  df.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

```
sns.heatmap(df.corr(),annot=True,cmap="coolwarm")
```

```
<ipython-input-33-0dd8dea822e2>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a f
  sns.heatmap(df.corr(),annot=True,cmap="coolwarm")
<Axes: >
```


```
sns.distplot(df["Miles"],kde=True)
plt.show()
```

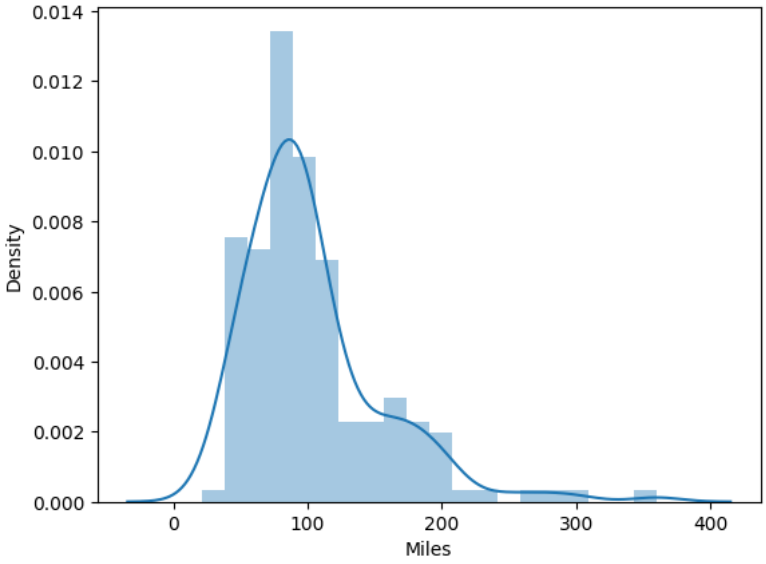
```
<ipython-input-34-de4135a45d97>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df["Miles"],kde=True)
```



probability

```
pd.crosstab(df.Gender,df.Product,margins=True)
```

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

```
prob_KP281=80/180
prob_KP281
```

```
0.4444444444444444
```

```
prob_KP481=60/180
prob_KP481
```

```
0.3333333333333333
```

```
prob_KP781=40/180
prob_KP781
```

```
0.2222222222222222
```

```
M_kp281=40/80
M_kp281
```

```
0.5
```

```
M_kp481=31/60
M_kp481
```

```
0.5166666666666667
```

```
M_kp781=33/40
M_kp781
```

```
0.825
```

```
FM_kp281=40/80
FM_kp281
```

```
0.5
```

```
FM_kp481=29/60
FM_kp481
```

```
0.48333333333333334
```

```
FM_kp781=7/40
FM_kp781
```

```
0.175
```

```
prob_Male=104/180
prob_Male
```

```
0.5777777777777777
```

```
prob_FMale=76/180
prob_FMale
```

```
0.4222222222222222
```

```
df.isna().sum()
```

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

Outlier Detection

```
df.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

```
u_25=df['Usage'].quantile(.25)
```

```
u_50=df['Usage'].quantile(.5)
```

```
u_75=df['Usage'].quantile(.75)
```

```
range=df['Usage'].max()-df['Usage'].min()
```

```
u_iq=u_75-u_25
```

```
u_lower_whisker=max(u_25-(1.5*u_iq),df['Usage'].min())
u_lower_whisker
```

```
2
```

```
u_upper_whisker=min(u_75+(1.5*u_iq),df['Usage'].max())
u_upper_whisker
```

```
5.5
```

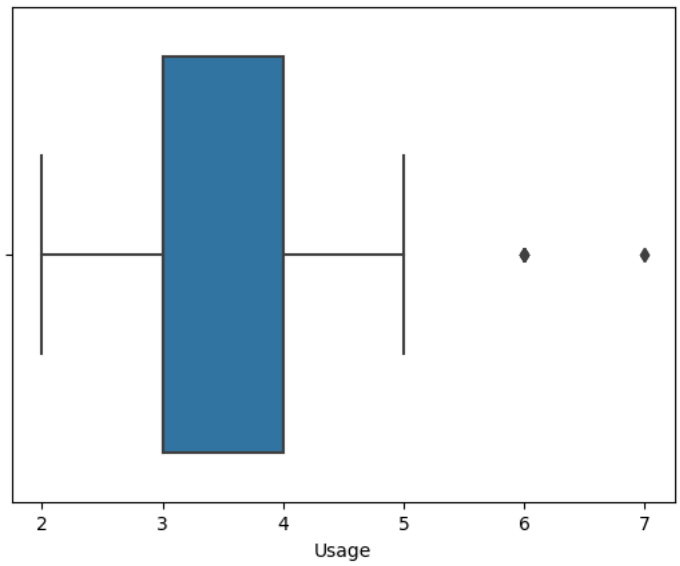
```
u_outlier=df.loc[df['Usage']>u_upper_whisker]
u_outlier
```

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
154	KP781	25	Male	18	Partnered	6	4	70966	180
155	KP781	25	Male	18	Partnered	6	5	75946	240
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
170	KP781	31	Male	16	Partnered	6	5	89641	260
175	KP781	40	Male	21	Single	6	5	83416	200

```
u_outlier.shape[0]/df.shape[0]
```

```
0.05
```

```
sns.boxplot(x=df['Usage'])
plt.show()
```



✓ 0s completed at 4:48 PM