What is Descriptive statistics, Predictive and Prescription?

Lets directly see example for clear understanding

Descriptive statistics

- How many Century does Virat Kholi have score in ODI cricket.

Predictive

- How many score may Virat Kholi will score in upcoming ODI with Australia.

Prescription

- What we can do so that Virat Kholi may score more century in ODI games

Descriptive statistics are methods used to summarize and describe the main features of a dataset.

Example of Descriptive statistics using Python

Cardio Good Fitness Case Study

The market research team at AdRight is assigned the task to identify the profile of the typical customer for each treadmill product offered by CardioGood Fitness. The market research team decides to investigate whether there are differences across the product lines with respect to customer characteristics. The team decides to collect data on individuals who purchased a treadmill at a CardioGoodFitness retail store during the prior three months.

The team identifies the following customer variables to study:

- product purchased, TM195, TM498, or TM798;
- gender;
- age, in years;
- education, in years;
- relationship status, single or partnered;
- annual household income ;
- average number of times the customer plans to use the treadmill each week;
- average number of miles the customer expects to walk/run each week;
- and self-rated fitness on an 1-to-5 scale, where 1 is poor shape and 5 is excellent shape.

Perform descriptive analytics to create a customer profile for each CardioGood Fitness treadmill product line.

In [1]:
```
#Import necessary library
import numpy as np
import pandas as pd
```

In [2]:
```
#import data
df=pd.read_csv(r'C:\Users\USER\Downloads\CardioGoodFitness.csv',encoding='latin1')
df.head()
```

Out[2]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
0	TM195	18	Male	14	Single	3	4	29562	112
1	TM195	19	Male	15	Single	2	3	31836	75
2	TM195	19	Female	14	Partnered	4	3	30699	66
3	TM195	19	Male	12	Single	3	3	32973	85
4	TM195	20	Male	13	Partnered	4	2	35247	47

In [3]:
```
df.describe()
```

Out[3]:

	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 [4]:

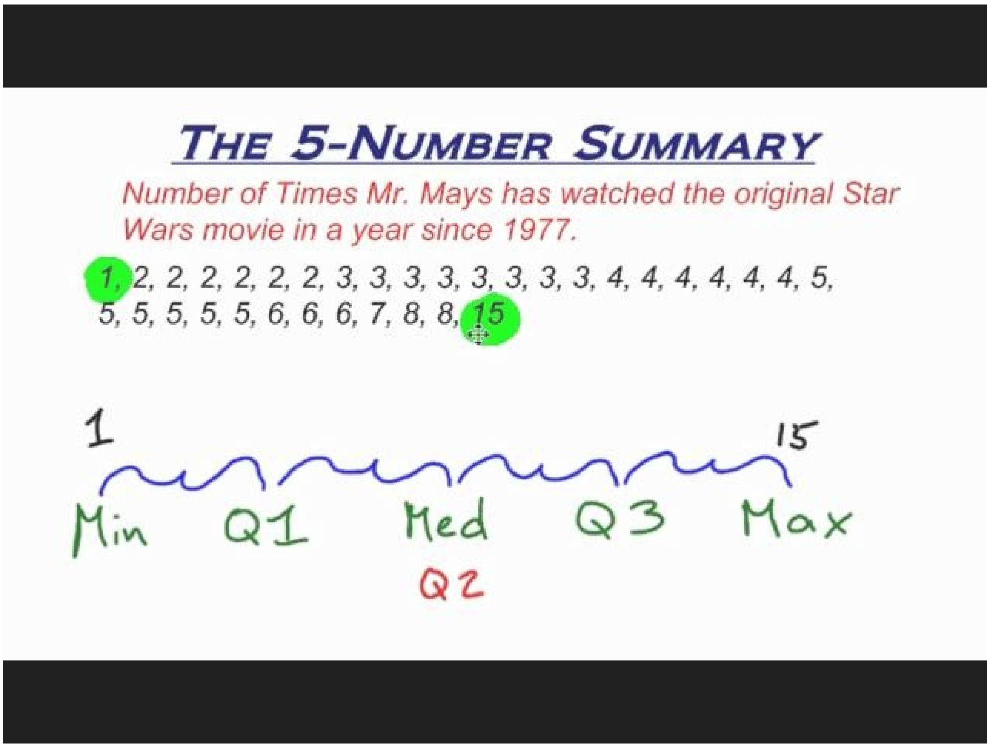
```
df.describe(include='all')
```

Out[4]:

	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	TM195	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

So suppose here, if someone asks us to create product according to age then how will we define age?

- According to 5 point summary, minimum is 18, so 25% of my data is approx age of 24s. 50 % of my data is approx age of 26. (According to above table)
- Another pictorial example of 5 point summary:



Note: Average age of person is 28.788(mean) but, age of average person is 26(median).

Now, if get confused think in this way_ Mean: add them all and divide how many there are, means (age1+age2+age3..)/180

Median: sort them from smallest to largest, pick of middle

Now lets look more above table, what we see: min is 18, mean is 28.78, median is 26 and max is 50, which shows that, people are more not average on oldest side than the youngest side means data are more pushed toward the right (Rightly skewed).

Skewness is measure with various things, one of measure of skewness is (mean-median), if mean-median=+ve, it correspondness right skewness.

In [5]:

```
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
```

Standard Devaition

Standard deviation is how spread typical object is from average.

Why we need to find out SD in Data analysis/Data Science?

- Firstly, it measure spreadness of data from mean. Secondly, we can find outliers. Outlier detection is a crucial aspect of data science, allowing data scientists to identify and address anomalies in the data and improve overall decision-making across various use cases. Such as :

Fraud Detection: Identifying unusual patterns in financial transactions can help detect fraudulent activities, such as credit card fraud or insider trading.

- Standard Deviation Method: The standard deviation measures a dataset's dispersion. Data points beyond a certain threshold (e.g., two or three standard deviations) from the mean are considered outliers.

In [6]:

```
#Example
l1=[2, 2.5, 1.25, 3.1, 1.75, 2.8] #suppose this list is our data
import numpy as np
standarddeviation1=np.std(l1)
print(standarddeviation1)
```

```
0.6322358912796886
```

In [7]:

```
#But above standard deviation is that deviation if we think our data as sample not population, for population w
#degree of freedom
standarddeviation2=np.std(l1,ddof=1)
print(standarddeviation2)
```

```
0.6925797186365384
```

HOW TO REMOVE OUTLIER USING SD

In [8]:

```
# Example:
# suppose we have a list
l1 = [10, 386, 479, 627, 20, 523, 482, 483, 542, 699, 535, 617, 577, 471, 615, 583, 441, 562, 563, 527, 453, 53
```

In [9]:

```
len(l1)      #length of list
```

Out[9]:

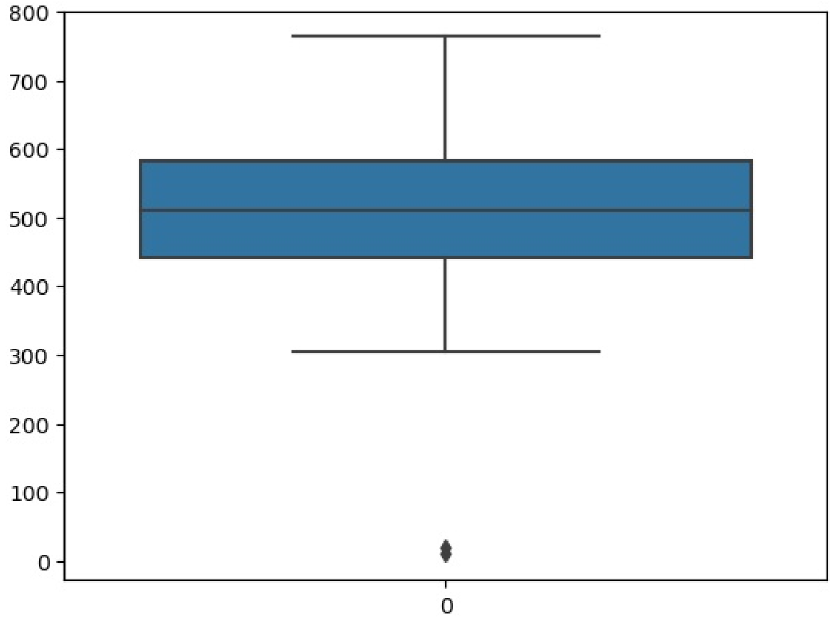
```
94
```

In [10]:

```
#lets draw curve from above list
import seaborn as sns
sns.boxplot(l1)
```

Out[10]:

```
<Axes: >
```



In [11]:
```
#Here, above we get simple boxplot because we have not provided any labels and style, for now we just want to o
# see below 100 there is small 2 traingle shaped dot, we have to remove it for effective data for better decisi
```

In [12]:
```
import numpy as np
l1_np=np.array(l1) #converting list into numpy
mean_l1_np=np.mean(l1_np)       #finding mean and standard deviation
std_l1_np=np.std(l1_np)
print(mean_l1_np)
print(std_l1_np)
```

```
509.531914893617
118.51857760182034
```

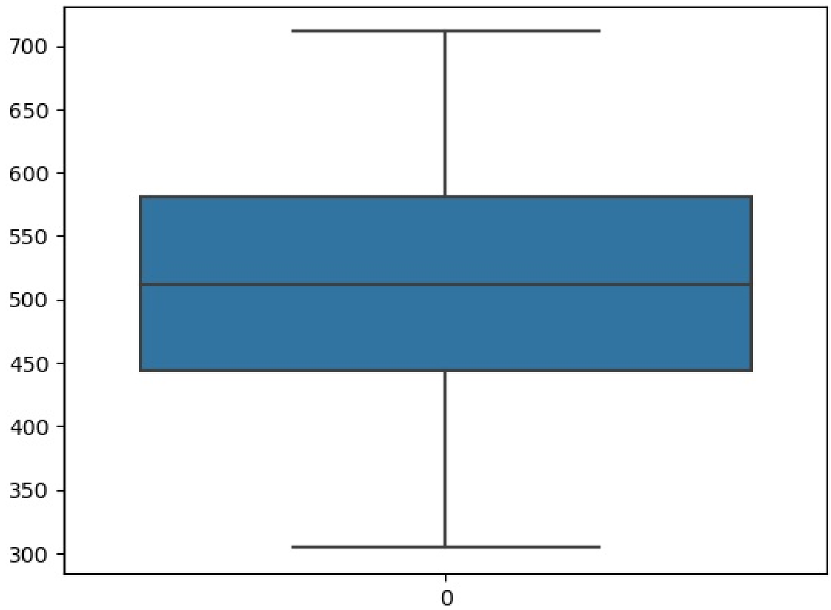
In [13]:
```
final_list = list(filter(lambda x: mean_l1_np - 2 * std_l1_np < x < mean_l1_np + 2 * std_l1_np, l1))
```

In [14]:
```
print(final_list)
```

```
[386, 479, 627, 523, 482, 483, 542, 699, 535, 617, 577, 471, 615, 583, 441, 562, 563, 527, 453, 530, 433, 541,
585, 704, 443, 569, 430, 637, 331, 511, 552, 496, 484, 566, 554, 472, 335, 440, 579, 341, 545, 615, 548, 604, 4
39, 556, 442, 461, 624, 611, 444, 578, 405, 487, 490, 496, 398, 512, 422, 455, 449, 432, 607, 679, 434, 597, 63
9, 565, 415, 486, 668, 414, 665, 557, 304, 404, 454, 689, 610, 483, 441, 657, 590, 492, 476, 437, 483, 529, 363
, 711, 543]
```

In [15]:
```
#now lets see boxplot
sns.boxplot(final_list)
```

Out[15]:
```
<Axes: >
```

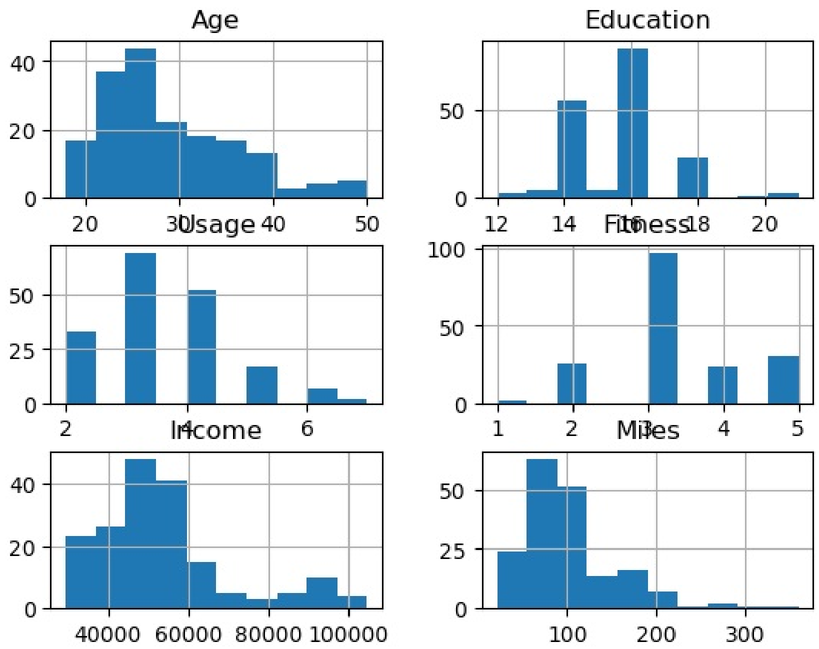


Understanding distribution and Histogram

In [16]:
```
import matplotlib.pyplot as plt
```

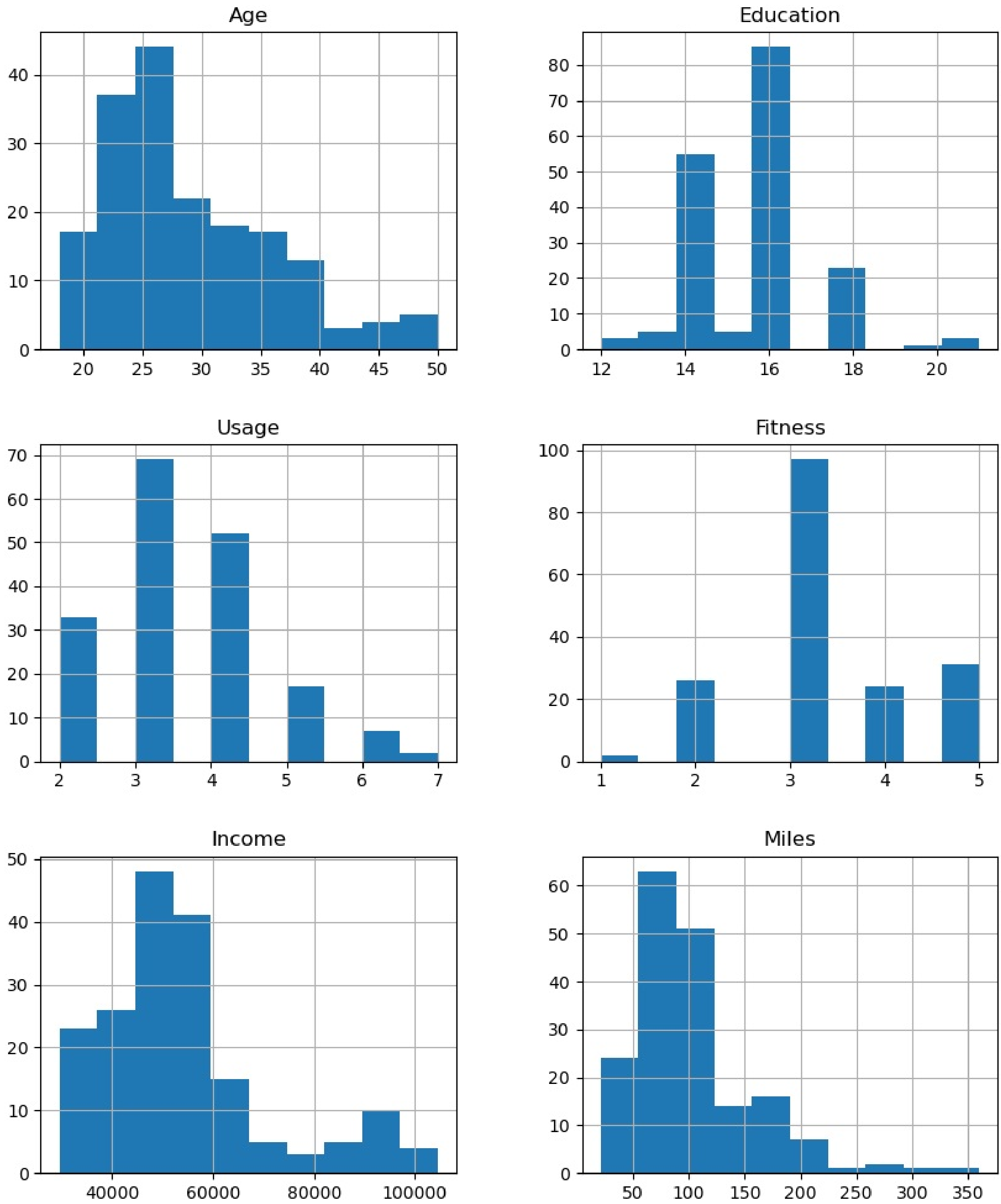
In [17]:
```
df.hist()         #it will give default figure and size because I have not kept fig size
```

Out[17]:
```
array([[<Axes: title={'center': 'Age'}>,
        <Axes: title={'center': 'Education'}>],
       [<Axes: title={'center': 'Usage'}>,
        <Axes: title={'center': 'Fitness'}>],
       [<Axes: title={'center': 'Income'}>,
        <Axes: title={'center': 'Miles'}>]], dtype=object)
```



In [18]:
```
#lets keep figsize
df.hist(figsize=(10,12))
```

Out[18]:
```
array([[<Axes: title={'center': 'Age'}>,
        <Axes: title={'center': 'Education'}>],
       [<Axes: title={'center': 'Usage'}>,
        <Axes: title={'center': 'Fitness'}>],
       [<Axes: title={'center': 'Income'}>,
        <Axes: title={'center': 'Miles'}>]], dtype=object)
```

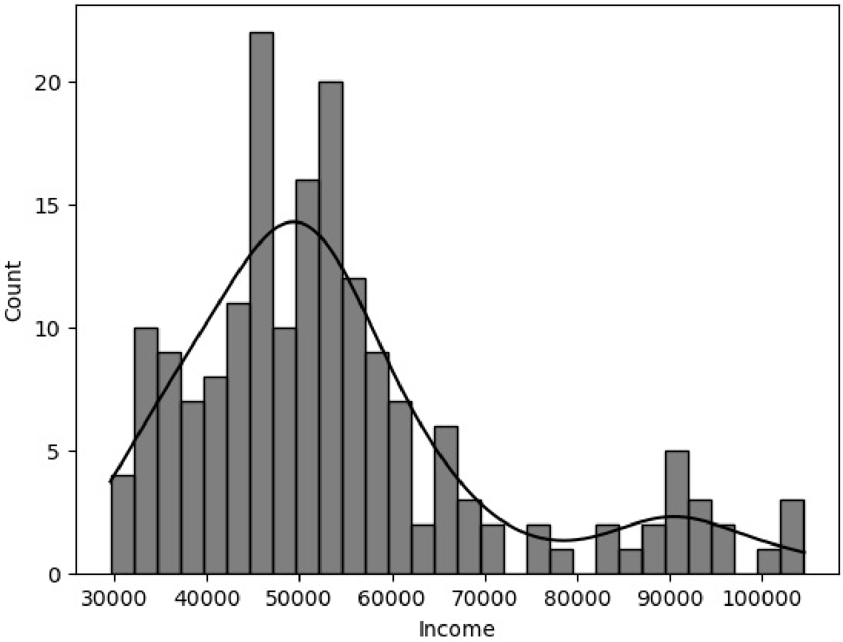

Age
0
10
20
30
40
20
25
30
35
40
45
50
Education
0
10
20
30
40
50
60
70
80
12
14
16
18
20
Usage
0
10
20
30
40
50
60
70
2
3
4
5
6
7
Fitness
0
20
40
60
80
100
1
2
3
4
5
Income
0
10
20
30
40
50
40000
60000
80000
100000
Miles
0
10
20
30
40
50
60
50
100
150
200
250
300
350

How we can read histogram?

In [19]:
```
#lets try to read income one: By seeing histogram, we can say it is rightly skewed histogram and it is unimodal
# And, atleast there is approx 47 people whose salary is in range 45000 to 50,000. Lets see income histogram mo
#suing sns than matplotlib

sns.histplot(df['Income'], kde=True, bins=30, color='black')
```

Out[19]:
```
<Axes: xlabel='Income', ylabel='Count'>
```



In [20]:
```
#lets find datapoints between 45000 and 50000 using pandas whether graph is approx or not
btw_45000_50000 =df[(df['Income'] > 45000) & (df['Income'] <= 50000)].count()        #used pandas
print(btw_45000_50000)
```

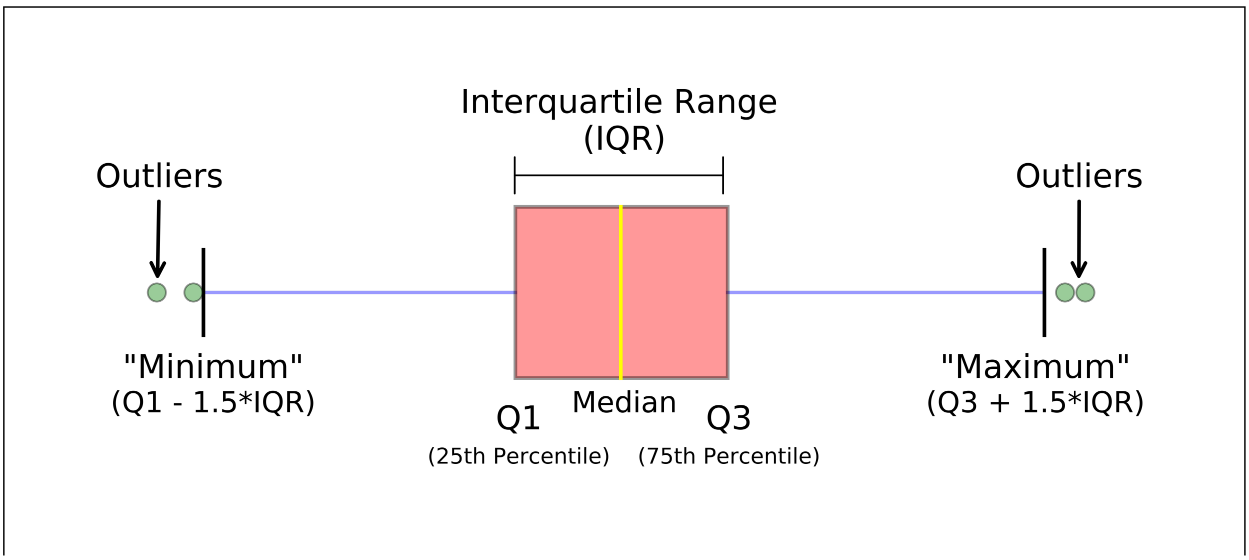
```
Product          34
Age              34
Gender           34
Education        34
MaritalStatus    34
Usage            34
Fitness          34
Income           34
Miles            34
dtype: int64
```

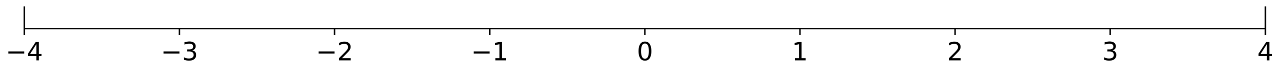
In [21]:
```
#this shows grouping by with all columns where income is between specefic values, but if we want all one group
btw_45000_50000 =df[(df['Income'] > 45000) & (df['Income'] <= 50000)]['Income'].count()        #used pandas
print('Income btw_45000_50000: ', btw_45000_50000)
```

```
Income btw_45000_50000:   34
```

Box Plot

A box and whisker plot—also called a box plot—displays the five-number summary of a set of data. The five-number summary is the minimum, first quartile, median, third quartile, and maximum





Summarizing Categorical data

CrossTabulation

In [22]:
```
pd.crosstab(df['Product'],df['Gender'])
```

Out[22]:

Gender	Female	Male
Product		
TM195	40	40
TM498	29	31
TM798	7	33

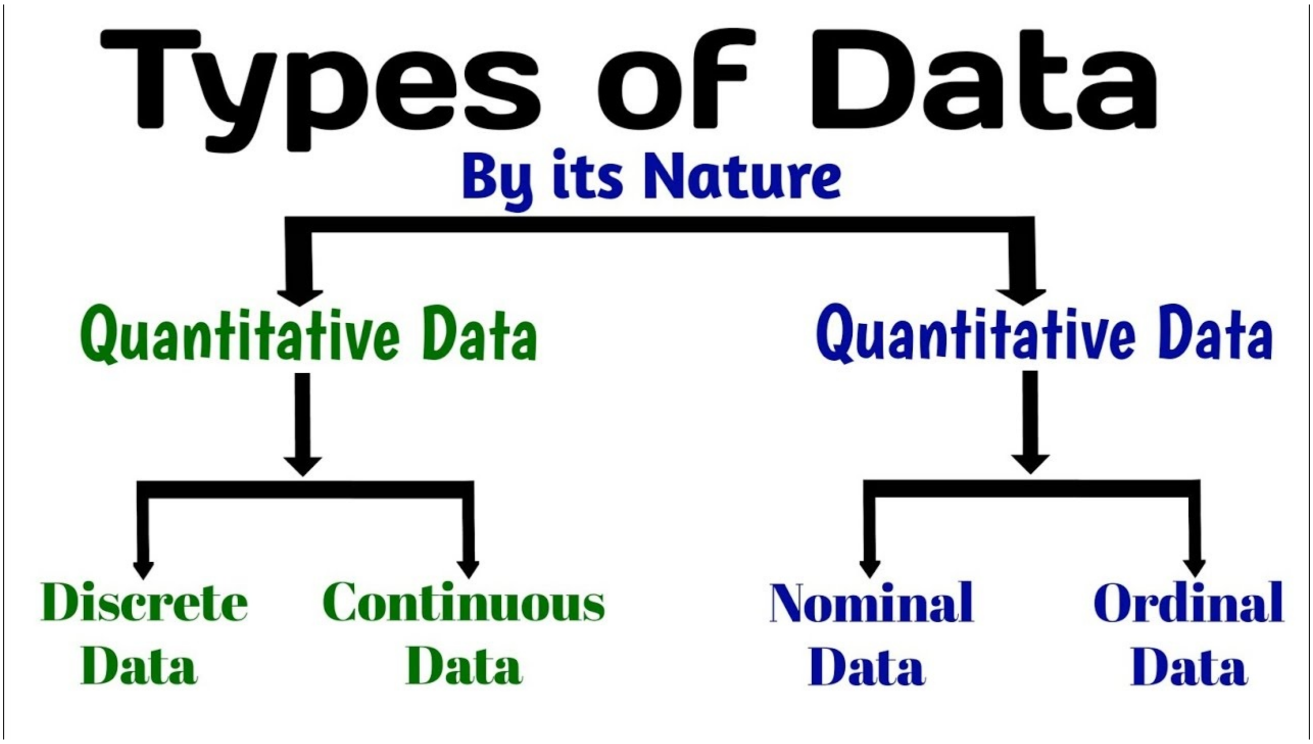
In [23]:
```
pd.crosstab(df['Product'],df['MaritalStatus'])
```

Out[23]:

MaritalStatus	Partnered	Single
Product		
TM195	48	32
TM498	36	24
TM798	23	17

How we can sue Categorical data and cross tab in stat and in data science?

- There is difference between male and female, some product may like by male more like TM798 and In simple terms, the data shows males and females differ in their purchases of some products but not others. Statistical tests help assess if those differences are significant or likely due to chance.
- For this we can use inferential statistics (chi square test & hypothesis)



Univariate Statistical plot and Univariate information

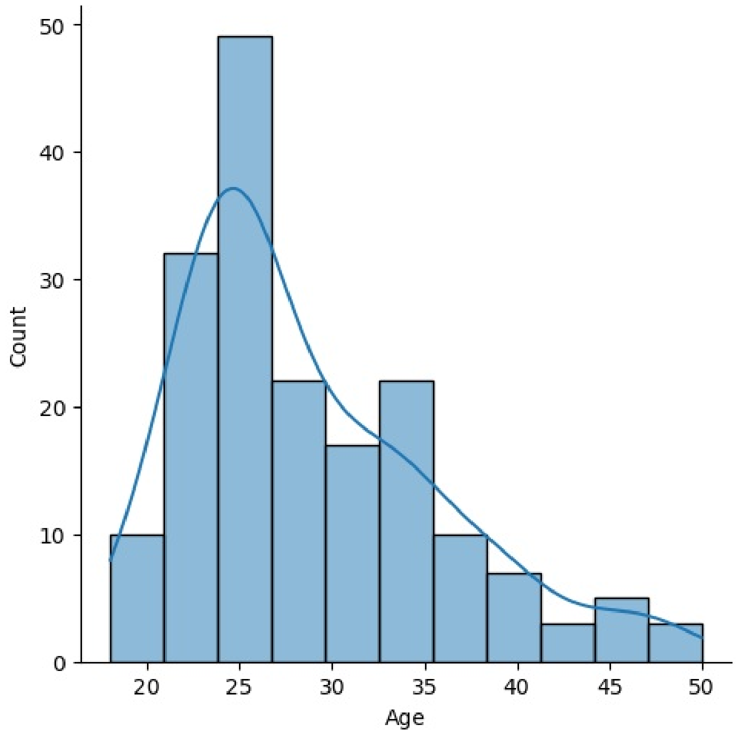
In [24]:
```
df['Age'].std()     #see standard deviation of age only
```

Out[24]:
```
6.943498135399795
```

In [25]:
```
#plot of age only
sns.displot(df['Age'],kde='True')
```

```
C:\Users\USER\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has changed t
o tight
  self._figure.tight_layout(*args, **kwargs)
```

Out[25]:
```
<seaborn.axisgrid.FacetGrid at 0x21545b41650>
```



Bi variate Statistics

Correlation: It is measure of linear relationship between variables. Its value lies between +1 to -1.

In [26]:
```
#find coorelation between variable from above df
#Correlation matrix
correlation_df=df.corr()
```

```
C:\Users\USER\AppData\Local\Temp\ipykernel_8772\3785941295.py:2: FutureWarning: The default value of numeric_on
ly in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or
specify the value of numeric_only to silence this warning.
  correlation_df=df.corr()
```

In [27]:
```
print(correlation_df)
```

```
                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
```

From above, just for reading: What relation between age and Education say?

- Age and education is positively correlated but relationship is very weak.

Heatmap (Correlation)

In [30]:
```
sns.heatmap(correlation_df,annot=True) #heatmap is drawn from above correlation table
```

Out[30]:
```
<Axes: >
```

Linear Regression

- It simply describe relationship, actually it does not give cause and effect
- It predict
- It prescript

$$Y_i = b_0 + \ b_1 X_{1i} + b_2 X_{2i} + b_n X_{ni} + u_i$$

$$\begin{aligned}
&Y_i &&= \text{dependent variable} \\
&b_0 &&= \text{Intercept} \\
&b_1 \dots b_n &&= \text{Coefficient of Regression} \\
&X_{1i} \dots X_{ni} &&= \text{independent variable} \\
&u_i &&= \text{disturbance error}
\end{aligned}$$

Example

Suppose, above x is income of person in some locality and y is profit for shopping mall of that locality, if new person come in locality, value of y can be different. This is a way linear regression predict for profit for shopping mall.

Now, lets look presciptive model of linear regression: In order to get different value of Y or according to my wish, to get this value of y, how value of X should be adjusted.

But, here now, we will only see descriptive relation or descriptive model for linear regression

In [33]:
```
#import linear regression model (Supervised learning)
```

```python
from sklearn import linear_model

reg=linear_model.LinearRegression()

y=df['Miles']       #Dependent variable (left side on above equation)
x=df[['Usage','Fitness']]

#train the model
reg.fit(x,y)
```

Out[33]:

```
▾ LinearRegression
LinearRegression()
```

In [34]:

```python
reg.coef_
```

Out[34]:

```
array([20.21486334, 27.20649954])
```

In [35]:

```python
reg.intercept_
```

Out[35]:

```
-56.74288178464859
```

In [36]:

```python
#Milespredicted=-56.743+20.21*Usage+27.20*Fitness          (Compare with above equation)
#Actually this is not good for prediction model. This is just for description.
```