In [1]:

```
import pandas as pd
```

In [2]:

```
df=pd.read_csv('D:/AnitaRJ/DATA SCIENCE/MScI_DataSci_Practicals/Practical6/stats.csv')
```

In [3]:

```
df
```

Out[3]:

	Name	Salary	Country
0	Dan	40000	USA
1	Elizabeth	32000	Brazil
2	Jon	45000	Italy
3	Maria	54000	USA
4	Mark	72000	USA
5	Bill	62000	Brazil
6	Jess	92000	Italy
7	Julia	55000	USA
8	Jeff	35000	Italy
9	Ben	48000	Brazil

Measure of Central Tendancy

In [4]:

```
# Mean Salary
mean1=df['Salary'].mean()
mean1
```

Out[4]:

```
53500.0
```

In [6]:

```
#Sum of Salaries
sum1=df['Salary'].sum()
sum1
```

Out[6]:

```
535000
```

In [7]:

```
#Maximum Salary
max1=df['Salary'].max()
max1
```

Out[7]:

```
92000
```

In [8]:

```
#Minimum Salary
min1=df['Salary'].min()
min1
```

Out[8]:

```
32000
```

In [9]:

```
#Total count

count1=df['Salary'].count()
count1
```

Out[9]:

```
10
```

In [10]:

```
#Median
median=df['Salary'].median()
median
```

Out[10]:

```
51000.0
```

In [12]:

```
#Mode
mode1=df['Salary'].mode()
mode1
```

Out[12]:

```
0    32000
1    35000
2    40000
3    45000
4    48000
5    54000
6    55000
7    62000
8    72000
9    92000
dtype: int64
```

In [16]:

```
countrywise_sum=df.groupby(['Country'])['Salary'].sum()
countrywise_sum
```

Out[16]:

```
Country
Brazil    142000
Italy     172000
USA       221000
Name: Salary, dtype: int64
```

In [14]:

```
countrywise_count=df.groupby(['Country']).count()
countrywise_count
```

Out[14]:

	Name	Salary
Country		
Brazil	3	3
Italy	3	3
USA	4	4

Measure of variability

In [17]:

```
#variance of salaries
var1=df['Salary'].var()
var1
```

Out[17]:

```
332055555.5555556
```

In [18]:

```
#standard deviation
std1=df['Salary'].std()
std1
```

Out[18]:

```
18222.391598128816
```

Measure of Symmetry

In [19]:

```
skew1=df.skew(axis=0, skipna=True)
skew1
```

Out[19]:

```
Salary    1.021551
dtype: float64
```

In [20]:

```
#The skewness is positive so x will have right side tail.
```

Covariance and Correlation

In [21]:

```
bw=pd.read_csv('D:/AnitaRJ/DATA SCIENCE/Anita_DSAI_Practicals/BirthWeight.csv')
bw.head()
```

Out[21]:

	Infant ID	Gestational Age (Weeks)	Birth Weight (Grams)
0	1	34.7	1895
1	2	36.0	2030
2	3	29.3	1440
3	4	40.1	2835
4	5	35.7	3090

In [22]:

```
bw.set_index('Infant ID', inplace=True)
bw.head()
```

Out[22]:

	Gestational Age (Weeks)	Birth Weight (Grams)
Infant ID		
1	34.7	1895
2	36.0	2030
3	29.3	1440
4	40.1	2835
5	35.7	3090

In [23]:

```
bw.cov()
```

Out[23]:

	Gestational Age (Weeks)	Birth Weight (Grams)
Gestational Age (Weeks)	9.963824	1798.025
Birth Weight (Grams)	1798.025000	485478.750

In [24]:

```
bw.corr(method="pearson")
```

Out[24]:

	Gestational Age (Weeks)	Birth Weight (Grams)
Gestational Age (Weeks)	1.000000	0.817519
Birth Weight (Grams)	0.817519	1.000000

In [25]:

```
#Covariance indicates that there is correlation exists between two
#Correlation coefficient of 0.818 indicates the relationship between two is positive and strong
```

In [1]:

```
# importing required libraries

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

In [2]:

```
pd.set_option("display.max_columns",None)  # to display all the columns
pd.options.display.float_format = "{:,.2f}".format # to display float value upto two decimals
```

Format : A data frame with 53940 rows and 10 variables

Description : A dataset containing the prices and other attributes of almost 54,000 diamonds.

The variables are as follows:

price: price in US dollars ($326--$18,823) carat: weight of the diamond (0.2--5.01) cut: quality of the cut (Fair, Good, Very Good, Premium, Ideal) colour: diamond colour, from J (worst) to D (best) clarity: a measurement of how clear the diamond is (IF (best), VVS1, VVS2,VS1, VS2, SI1, SI2, I1 (worst)) popularity: popularity of this specs (Good, Fair, Poor) x: length in mm (0--10.74) y: width in mm (0--58.9) z: depth in mm (0--31.8) depth: total depth percentage = z / mean(x, y) = 2 * z / (x + y) (43--79) table: width of top of diamond relative to widest point (43--95)

In [3]:

```
# reading data from csv file
xls = pd.read_csv('D:/AnitaRJ/DATA SCIENCE/Anita_DSAI_Practicals/diamonds.csv')
```

In [4]:

```
xls.head()
```

Out[4]:

	id	carat	cut	color	clarity	depth	table	price	x	y	z
0	1	0.23	Ideal	E	SI2	61.50	55.00	326	3.95	3.98	2.43
1	2	0.21	Premium	E	SI1	59.80	61.00	326	3.89	3.84	2.31
2	3	0.23	Good	E	VS1	56.90	65.00	327	4.05	4.07	2.31
3	4	0.29	Premium	I	VS2	62.40	58.00	334	4.20	4.23	2.63
4	5	0.31	Good	J	SI2	63.30	58.00	335	4.34	4.35	2.75

In [6]:

```
des_df = xls.drop(['id'],axis = 1) # drop id column
for col in des_df:   # drop all alpha-numeric columns
  if des_df[col].dtype == 'object':
    des_df = des_df.drop([col], axis = 1)

des_r = des_df.describe() # describe() gives us mean,min,max,median,1Q,3Q,std
des_r = des_r.rename(index={'50%':'median/50%'})
des_r
```

Out[6]:

	carat	depth	table	price	x	y	z
count	53,940.00	53,940.00	53,940.00	53,940.00	53,940.00	53,940.00	53,940.00
mean	0.80	61.75	57.46	3,932.80	5.73	5.73	3.54
std	0.47	1.43	2.23	3,989.44	1.12	1.14	0.71
min	0.20	43.00	43.00	326.00	0.00	0.00	0.00
25%	0.40	61.00	56.00	950.00	4.71	4.72	2.91
median/50%	0.70	61.80	57.00	2,401.00	5.70	5.71	3.53
75%	1.04	62.50	59.00	5,324.25	6.54	6.54	4.04
max	5.01	79.00	95.00	18,823.00	10.74	58.90	31.80

In [7]:

```
var_r = des_df.var() # calulating variance seperately

varlist = []
for col in des_df.columns: # converting result of var() from series to list
  if des_df[col].dtype == 'object':
    continue
  varlist.append(round(des_df[col],5))

df = pd.DataFrame([varlist],columns=des_r.columns, index=['var']) # putting results of variance into dataframe
mct = des_r.append(df) # adding var to describe result
mct
```

Out[7]:

	carat	depth	table	price	x	y	z
count	53,940.00	53,940.00	53,940.00	53,940.00	53,940.00	53,940.00	53,940.00
mean	0.80	61.75	57.46	3,932.80	5.73	5.73	3.54
std	0.47	1.43	2.23	3,989.44	1.12	1.14	0.71
min	0.20	43.00	43.00	326.00	0.00	0.00	0.00
25%	0.40	61.00	56.00	950.00	4.71	4.72	2.91
median/50%	0.70	61.80	57.00	2,401.00	5.70	5.71	3.53
75%	1.04	62.50	59.00	5,324.25	6.54	6.54	4.04
max	5.01	79.00	95.00	18,823.00	10.74	58.90	31.80
var	0 0.23 1 0.21 2 0.23 3 ...	0 61.50 1 59.80 2 56.90 3 ...	0 55.00 1 61.00 2 65.00 3 ...	0 326 1 326 2 327 3 ...	0 3.95 1 3.89 2 4.05 3 ...	0 3.98 1 3.84 2 4.07 3 ...	0 2.43 1 2.31 2 2.31 3 ...

In [ ]: