In [1]:

```
import pandas as pd
```

In [2]:

```
set={'Name':['Gautham','Siddhesh','Chandan','Manish','Shivam','Pratik','Jayesh','Bhagwa
    [100000,60000,70000,80000,90000,45000,50000,25000,75000,65000],'Country':['India','
                                                                          'SriLanka',
df=pd.DataFrame(set)
df
```

Out[2]:

	Name	Salary	Country
0	Gautham	100000	India
1	Siddhesh	60000	Pakistan
2	Chandan	70000	Afghanistan
3	Manish	80000	China
4	Shivam	90000	USA
5	Pratik	45000	Russia
6	Jayesh	50000	SriLanka
7	Bhagwan	25000	UAE
8	George	75000	Australia
9	Nitin	65000	England

In [3]:

```
df.Salary.mean()
```

Out[3]:

```
66000.0
```

In [4]:

```
df.Salary.median()
```

Out[4]:

```
67500.0
```

In [5]:

```
df.Salary.mode()
```

Out[5]:

```
0     25000
1     45000
2     50000
3     60000
4     65000
5     70000
6     75000
7     80000
8     90000
9    100000
dtype: int64
```

In [6]:

```
df.Salary.min()
```

Out[6]:

```
25000
```

In [7]:

```
df.Salary.max()
```

Out[7]:

```
100000
```

In [8]:

```
df.Salary.count()
```

Out[8]:

```
10
```

In [9]:

```
df.Salary.sum()
```

Out[9]:

```
660000
```

In [10]:

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

Out[10]:

	Salary
Country	
Afghanistan	70000
Australia	75000
China	80000
England	65000
India	100000
Pakistan	60000
Russia	45000
SriLanka	50000
UAE	25000
USA	90000

In [11]:

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

Out[11]:

	Name	Salary
Country		
Afghanistan	1	1
Australia	1	1
China	1	1
England	1	1
India	1	1
Pakistan	1	1
Russia	1	1
SriLanka	1	1
UAE	1	1
USA	1	1

In [12]:

```
var=df['Salary'].var()
var  #variance in salary
```

Out[12]:

```
493333333.3333333
```

In [13]:

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

Out[13]:

```
22211.108331943575
```

In [14]:

```
skew=df['Salary'].skew(skipna=True)
skew
```

Out[14]:

```
-0.32740191758018083
```

In [15]:

```
Data=pd.read_csv('BirthWeight.csv')
Data.head()
```

Out[15]:

	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 [16]:

```
Data.cov()
```

Out[16]:

	Infant ID	Gestational Age (Weeks)	Birth Weight (Grams)
Infant ID	25.500	7.825000	1333.750
Gestational Age (Weeks)	7.825	9.963824	1798.025
Birth Weight (Grams)	1333.750	1798.025000	485478.750

In [17]:

```
Data.corr(method='pearson')
```

Out[17]:

	Infant ID	Gestational Age (Weeks)	Birth Weight (Grams)
Infant ID	1.000000	0.490909	0.379070
Gestational Age (Weeks)	0.490909	1.000000	0.817519
Birth Weight (Grams)	0.379070	0.817519	1.000000

In [27]:

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

In [20]:

```
pd.set_option("display.max_columns",None)
pd.options.display.float_format="{:2f}".format
```

In [21]:

```
xls=pd.read_csv('diamonds.csv')
xls
```

Out[21]:

	id	carat	cut	color	clarity	depth	table	price	x	
0	1	0.230000	Ideal	E	SI2	61.500000	55.000000	326	3.950000	3.98000
1	2	0.210000	Premium	E	SI1	59.800000	61.000000	326	3.890000	3.84000
2	3	0.230000	Good	E	VS1	56.900000	65.000000	327	4.050000	4.07000
3	4	0.290000	Premium	I	VS2	62.400000	58.000000	334	4.200000	4.23000
4	5	0.310000	Good	J	SI2	63.300000	58.000000	335	4.340000	4.35000
...	...	...	...	...	...	...	...	...	...	.
53935	53936	0.720000	Ideal	D	SI1	60.800000	57.000000	2757	5.750000	5.76000
53936	53937	0.720000	Good	D	SI1	63.100000	55.000000	2757	5.690000	5.75000
53937	53938	0.700000	Very Good	D	SI1	62.800000	60.000000	2757	5.660000	5.68000
53938	53939	0.860000	Premium	H	SI2	61.000000	58.000000	2757	6.150000	6.12000
53939	53940	0.750000	Ideal	D	SI2	62.200000	55.000000	2757	5.830000	5.87000

53940 rows × 11 columns

In [22]:

```
xls.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 53940 entries, 0 to 53939
Data columns (total 11 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   id       53940 non-null  int64  
 1   carat    53940 non-null  float64
 2   cut      53940 non-null  object 
 3   color    53940 non-null  object 
 4   clarity  53940 non-null  object 
 5   depth    53940 non-null  float64
 6   table    53940 non-null  float64
 7   price    53940 non-null  int64  
 8   x        53940 non-null  float64
 9   y        53940 non-null  float64
 10  z        53940 non-null  float64
dtypes: float64(6), int64(2), object(3)
memory usage: 4.5+ MB
```

In [26]:

```
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()
des_r=des_r.rename(index={'50%':'median/50%'})
des_r
```

Out[26]:

	carat	depth	table	price	x	
count	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000	53940.0000
mean	0.797940	61.749405	57.457184	3932.799722	5.731157	5.7345
std	0.474011	1.432621	2.234491	3989.439738	1.121761	1.1421
min	0.200000	43.000000	43.000000	326.000000	0.000000	0.0000
25%	0.400000	61.000000	56.000000	950.000000	4.710000	4.7200
median/50%	0.700000	61.800000	57.000000	2401.000000	5.700000	5.7100
75%	1.040000	62.500000	59.000000	5324.250000	6.540000	6.5400
max	5.010000	79.000000	95.000000	18823.000000	10.740000	58.9000

In [ ]: