In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
my_dict={'Name':['dan','elizabeth','john','maria','mark','bill','jess','julia','j
         'Salary':[40000,32000,45000,54000,72000,62000,92000,55000,35000,48000],
         'Country':['USA','Brazil','Italy','USA','USA','Brazil','Italy','USA','Ita
df=pd.DataFrame(my_dict)
df
```

Out[2]:

	Name	Salary	Country
0	dan	40000	USA
1	elizabeth	32000	Brazil
2	john	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

In [3]:
```
df.to_csv('my_dict')
df
```

Out[3]:

	Name	Salary	Country
0	dan	40000	USA
1	elizabeth	32000	Brazil
2	john	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

In [4]:
```
df_csv=pd.read_csv('my_dict')
df_csv
```

Out[4]:

	Unnamed: 0	Name	Salary	Country
0	0	dan	40000	USA
1	1	elizabeth	32000	Brazil
2	2	john	45000	Italy
3	3	maria	54000	USA
4	4	mark	72000	USA
5	5	bill	62000	Brazil
6	6	jess	92000	Italy
7	7	julia	55000	USA
8	8	jeff	35000	Italy
9	9	ben	48000	Brazil

In [5]:
```
mean = df['Salary'].mean()
mean
```

Out[5]:
```
53500.0
```

In [6]:
```
median = df['Salary'].median()
median
```

Out[6]:
```
51000.0
```

In [7]:
```
mode = df['Salary'].mode()
mode
```

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

In [8]:
```
sum = df['Salary'].sum()
sum
```

Out[8]:
```
535000
```

In [9]:
```
max = df['Salary'].max()
max
```

Out[9]: 92000

In [10]:
```
min = df['Salary'].min()
min
```

Out[10]: 32000

In [11]:
```
count = df['Salary'].count()
count
```

Out[11]: 10

In [12]:
```
countrywise_sum = df.groupby(['Country'])['Salary'].sum()
countrywise_sum
```

Out[12]:
```
Country
Brazil    142000
Italy     172000
USA       221000
Name: Salary, dtype: int64
```

In [13]:
```
countrywise_count= df.groupby(['Country']).count()
countrywise_count
```

Out[13]:

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

In [14]:
```
#varience of sal
var1 = df['Salary'].var()
var1
```

Out[14]: 332055555.5555556

In [15]:
```
std1 = df['Salary'].std()
std1
```

Out[15]: 18222.391598128816

In [16]:
```
skew1= df.skew(axis = 0, skipna = True)
skew1
```

```
C:\Users\MSCIT\AppData\Local\Temp/ipykernel_6284/1353041505.py:1: FutureWarnin
g: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=Non
e') is deprecated; in a future version this will raise TypeError.  Select only
valid columns before calling the reduction.
  skew1= df.skew(axis = 0, skipna = True)
```

Out[16]:
```
Salary    1.021551
dtype: float64
```

In [17]:
```
# The skewnwss is positive so x will have right side tail.
```

Covarience and Correlation

In [19]:
```
bw = pd.read_csv('BirthWeight.csv')
bw.head()
```

Out[19]:

	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 [20]:
```
bw.set_index('Infant ID', inplace = True)
bw.head()
```

Out[20]:

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

In [21]:
```
bw.cov()
```

Out[21]:

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

In [23]:
```
bw.corr(method = 'pearson')
```

Out[23]:

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

In [24]:
```
#Covariance indicates that there is correlation exists between two
#Correlation coefficient of 0.818 indicates the relationship between two is posit
```

In [26]:
```
#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 [27]:
```
pd.set_option('display.max_columns',None) #to display all the columns
pd.options.display.float_format = '{:,.2f}'.format #to display float value upto
```

In [29]:
```
xls= pd.read_csv('diamonds.csv')
xls.head()
```

Out[29]:

	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 [31]:
```
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[31]:

	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 [32]:
```
var_r = des_df.var()

varlist = []
for col in des_df.columns:
    if des_df[col].dtype == 'object':
        continue
    varlist.append(round(des_df[col],5))

df = pd.DataFrame([varlist], columns = des_r.columns, index = ['var'])
mct = des_r.append(df)
mct
```

Out[32]:

	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 [ ]: