GOVERNAMENT POLYTECHNIC NAGAMANGALA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

5th semester Diploma

ARTIFICIAL INTELIGENCE and MACHINE LEARNING

(20CS51)

Assigment :02

NAME:**DEEKSHA .D**

ROLL NO:**158CS22012**

AIML(20CS51)

ASSIGMENT-02

1.Download any two datasets from the internet and perform the following operations.

a) Analyze the univariate dataset Ex-Mean,Mode,Median,Range,std,and Variance and perform Univariate tests for the dataset.

.b)Analyze the multivariate of the dataset Ex-co-variance,co-relation

c)Visualize the univariate and multivariate with various plots.

d)Push the code to your GitHub Repository

1.Download any two datasets from the internet and perform the following operations.

DATASET 01:"/content/archive (47).zip"

DESCRIPTION:

LAPTOP

STATUS

BRAND

MODEL

CPU

RAM

STORAGE

STORAGE TYPE

GPU

SCREEN TOUCH

FINAL PRICE

a) Analyze the univariate dataset Ex-Mean,Mode,Median,Range,std,and Variance and perform Univariate tests for the dataset.

- **MEAN**

```
import pandas as pd
path="/content/archive (47).zip"
df=pd.read_csv(path)
print(df)
```

:OUTPUT

```
      Laptop          Status     Brand    \
0     ASUS ExpertBook B1 B1502CBA-EJ0436X Intel Core...        New      Asus
1     Alurin Go Start Intel Celeron N4020/8GB/256GB ...        New    Alurin
2     ASUS ExpertBook B1 B1502CBA-EJ0424X Intel Core...        New      Asus
3     MSI Katana GF66 12UC-082XES Intel Core i7-1270...        New       MSI
4     HP 15S-FQ5085NS Intel Core i5-1235U/16GB/512GB...        New        HP
...                                                ...        ...      ...
2155  Razer Blade 17 FHD 360Hz Intel Core i7-11800H/...  Refurbished    Razer
2156  Razer Blade 17 FHD 360Hz Intel Core i7-11800H/...  Refurbished    Razer
2157  Razer Blade 17 FHD 360Hz Intel Core i7-11800H/...  Refurbished    Razer
2158  Razer Book 13 Intel Evo Core i7-1165G7/16GB/1T...  Refurbished    Razer
2159  Razer Book FHD+ Intel Evo Core i7-1165G7/16GB/...  Refurbished    Razer

          Model            CPU  RAM  Storage Storage type       GPU  \
0    ExpertBook    Intel Core i5    8     512          SSD       NaN
1            Go   Intel Celeron     8     256          SSD       NaN
2    ExpertBook    Intel Core i3    8     256          SSD       NaN
3        Katana    Intel Core i7   16    1000          SSD  RTX 3050
4           15S    Intel Core i5   16     512          SSD       NaN
...         ...              ...  ...     ...          ...       ...
```

2155	Blade	Intel Core i7	16	1000	SSD	RTX 3060
2156	Blade	Intel Core i7	16	1000	SSD	RTX 3070
2157	Blade	Intel Core i7	32	1000	SSD	RTX 3080
2158	Book	Intel Evo Core i7	16	1000	SSD	NaN
2159	Book	Intel Evo Core i7	16	256	SSD	NaN

	Screen	Touch	Final Price
0	15.6	No	1009.00
1	15.6	No	299.00
2	15.6	No	789.00
3	15.6	No	1199.00
4	15.6	No	669.01
...	...	...	...
2155	17.3	No	2699.99
2156	17.3	No	2899.99
2157	17.3	No	3399.99
2158	13.4	Yes	1899.99
2159	13.4	Yes	1699.99

[2160 rows x 12 columns]

.MODE

df.mode()

:OUTPUT

	Laptop	Status	Brand	Model	CPU	RAM	Storage	Storage type	GPU	Screen	Touch	Final Price
0	ASUS ROG Zephyrus M16 GU604VI-93D47PB1 Intel ...	New	Asus	15S	Intel Core i7	16.0	512.0	SSD	RTX 3050	15.6	No	999.0
1	ASUS BR1100FKA-BP1185XA Intel Celeron N4500/4G...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	ASUS Chromebook C433TA-AJ0336 Intel Core m3-81...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

3	ASUS Chromebook CR1 CR1100CKA-GJ0132 Intel Cel...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	ASUS Chromebook CR1100FKA-BP0024 Intel Celeron...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...
2155	Vant Edge Intel Core i5-10210U/16GB/500GB SSD/...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2156	Vant Edge Intel Core i7-10510U/16GB/500GB SSD/...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2157	Vant Edge Intel Core i7-10510U/16GB/500GB SSD/...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2158	Vant Moove3-14 Intel Core i5-1135G7/16GB/500GB...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2159	Vant Moove3-14 Intel Core i5-1135G7/8GB/500GB ...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

2160 rows × 12 columns

MEDIAN

df.median()

```
RAM            15.413889
Storage       596.294444
Screen         15.168112
Final Price  1312.638509
dtype: float64
```

dtype: float64

df.std(numeric_only=True)

output:

```
RAM            9.867815
Storage      361.220506
Screen         1.203329
Final Price  911.475417
```

dtype: float64

df.describe()

output:

	RAM	Storage	Screen	Final Price
count	2160.000000	2160.000000	2156.000000	2160.000000
mean	15.413889	596.294444	15.168112	1312.638509
std	9.867815	361.220506	1.203329	911.475417
min	4.000000	0.000000	10.100000	201.050000
25%	8.000000	256.000000	14.000000	661.082500
50%	16.000000	512.000000	15.600000	1031.945000
75%	16.000000	1000.000000	15.600000	1708.970000
max	128.000000	4000.000000	18.000000	7150.470000

```
import pandas as pd

path=("/content/archive (49).zip")

df=pd.read_csv(path)

numeric_columns = df.select_dtypes(include=['number'])

range_value = numeric_columns.max() - numeric_columns.min()

print(range_value)
```

output:

```
Open      563.450012
High      567.899986
Low       562.750000
Close     563.949997
dtype: float
```

- **Univariate tests for the dataset.**

one sample t test

```
import numpy as np
from scipy import stats
d=pd.read_csv('/content/archive (47).zip')
RAM=np.array([8,8,8,16,16,16,16,32,16,16])
t_stat,p_value= stats.ttest_1samp(RAM,popmean=7.2)
print( f"one sample t_test:{t_stat},p_value:{p_value}")
```

output:

one sample

t_test:3.6115755925730757,p_value:0.005645419042037466

chi square

```
import pandas as pd
from scipy.stats import chi2_contingency
data =pd.read_csv('/content/archive (47).zip')
cantingency_table = pd.crosstab(data['RAM'], data['Storage'])
chi2, p, dof, ex = chi2_contingency(cantingency_table)
print(f"Chi-square Test of Independence:
chi2={chi2},p={p},dof={dof},expected={ex}")
```

output:

Chi-square Test of Independence: chi2=3314.4100714961937,p=0.0,dof=88,expected=[[3.14814815e-02 4.40740741e-01 1.10185185e+00 2.10925926e+00

3.14814815e-02 1.41666667e+01 1.16481481e+00 2.96240741e+01

```
 1.79444444e+01 1.32222222e+00 3.14814815e-02 3.14814815e-02]
[1.38888889e-03 1.94444444e-02 4.86111111e-02 9.30555556e-02
 1.38888889e-03 6.25000000e-01 5.13888889e-02 1.30694444e+00
 7.91666667e-01 5.83333333e-02 1.38888889e-03 1.38888889e-03]
[3.78240741e-01 5.29537037e+00 1.32384259e+01 2.53421296e+01
 3.78240741e-01 1.70208333e+02 1.39949074e+01 3.55924537e+02
 2.15597222e+02 1.58861111e+01 3.78240741e-01 3.78240741e-01]
[6.94444444e-03 9.72222222e-02 2.43055556e-01  4.65277778e-01
 6.94444444e-03 3.12500000e+00 2.56944444e-01 6.53472222e+00
 3.95833333e+00 2.91666667e-01 6.94444444e-03 6.94444444e-03]
[4.29629630e-01 6.01481481e+00 1.50370370e+01 2.87851852e+01
 4.29629630e-01 1.93333333e+02 1.58962963e+01 4.04281481e+02
 2.44888889e+02 1.80444444e+01 4.29629630e-01  4.29629630e-01]
[1.39351852e-01 1.95092593e+00 4.87731481e+00 9.33657407e+00
 1.39351852e-01 6.27083333e+01 5.15601852e+00 1.31130093e+02
 7.94305556e+01 5.85277778e+00 1.39351852e-01 1.39351852e-01]
[9.25925926e-04 1.29629630e-02 3.24074074e-02 6.20370370e-02
 9.25925926e-04 4.16666667e-01 3.42592593e-02 8.71296296e-01
 5.27777778e-01 3.88888889e-02 9.25925926e-04 9.25925926e-04]
[1.15740741e-02 1.62037037e-01 4.05092593e-01 7.75462963e-01
 1.15740741e-02 5.20833333e+00 4.28240741e-01 1. 59722222e+00 4.86111111e-01 1.15740741e-02 1.15740741e-02]
[4.62962963e-04 6.48148148e-03 1.62037037e-02 3.10185185e-02
 4.62962963e-04 2.08333333e-01 1.71296296e-02 4.356481480-01
 2.63888889e-01 1.94444444e-02 4.62962963e-04 4.62962963e-04]]
08912037e+01
```

anova

```
import pandas as pd
from scipy.stats import f_oneway
d=pd.read_csv('/content/archive (47).zip')
```

```
groups = [d['Model'],d['Screen'], d['Storage']]
f_stat, p_value = f_oneway(*groups)
print(f"F-valuei:{f_stat},p-value:{p_value}")
```

output:

F-valuei:nan,p-value:nan

Wilcoxon

```
import pandas as pd
import numpy as np
from scipy.stats import wilcoxon
d=pd.read_csv('/content/archive (47).zip')
data={
        'before':[8,8,8,16,16,16,16,32,16,16],
         'after':[6,9,41,34,7,5,22,15,20,30]
}
df=pd.DataFrame(data)
stat,p_value=wilcoxon(df['before'],df['after'])
print(f"wilcoxon signed-rank statistics:{stat},p_value:{p_value}")
```

output:

wilcoxon signed-rank statistics:21.0,p_value:0.556640625

kruskal walls

```
import pandas as pd
```

```
from scipy.stats import kruskal
import numpy as np
d=pd.read_csv('/content/archive (47).zip')
d={
    'group1':np.random.normal(loc=0,scale=1,size=10),
    'group2':np.random.normal(loc=0,scale=1,size=10),
    'group3':np.random.normal(loc=0,scale=1,size=10)
}
f_stat,p_value=kruskal(d['group1'],d['group2'],d['group3'])
print(f"kruskal-walls H statistics :{f_stat},p-value:{p_value}")
```

output:

```
kruskal-walls H statistics :0.4670967741935499,p-value:0.7917192858157005
```

.b)Analyze the multivariate of the dataset Ex-co-variance,co-relation

df.var(numeric_only=True)

output:

```
RAM                97.373776
Storage        130480.254161
Screen              1.448000
Final Price    830787.435855
dtype: float64
```

df.corr(numeric_only=True)

output:

	RAM	Storage	Screen	Final Price
RAM	1.000000	0.751297	0.361404	0.724946
Storage	0.751297	1.000000	0.398025	0.695631
Screen	0.361404	0.398025	1.000000	0.268359
Final Price	0.724946	0.695631	0.268359	1.000000

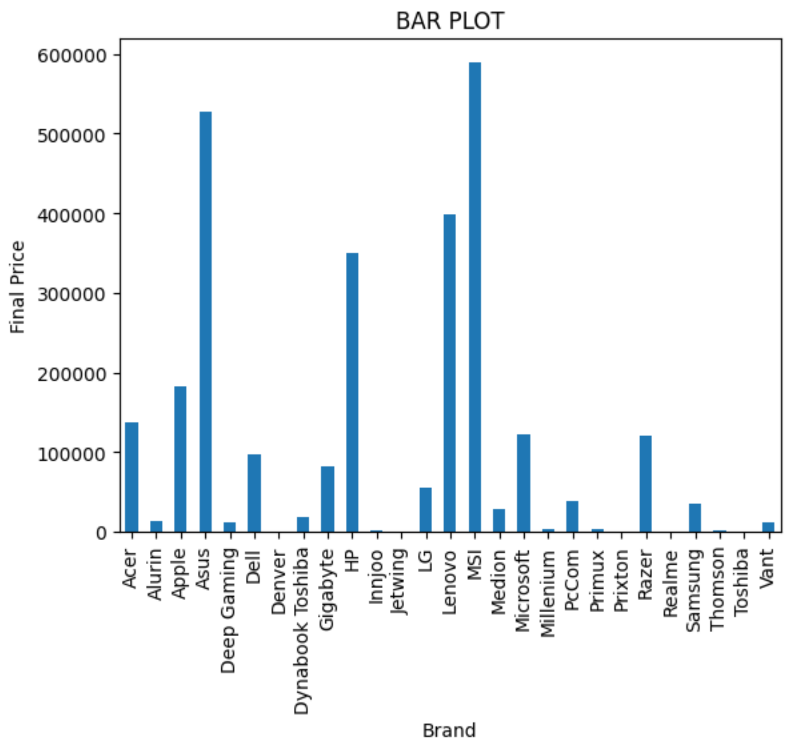
C)**Visualize the univariate and multivariate with various plots.**

various plots.

BAR PLOT

```
import pandas as pd

df=pd.read_csv('/content/archive (47).zip')

import matplotlib.pyplot as plt

df.groupby('Brand')['Final Price'].sum().plot(kind='bar')

plt.title("BAR PLOT")

plt.xlabel("Brand")

plt.ylabel("Final Price")

plt.show()
```

output:



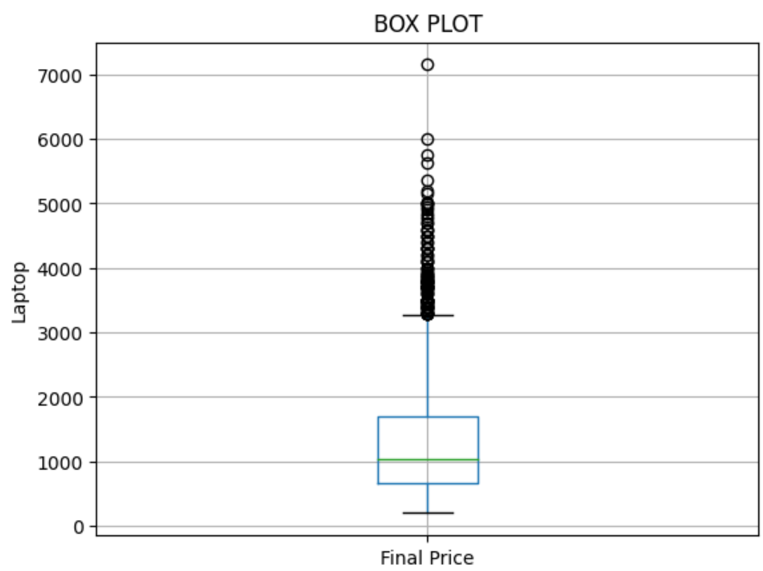
df.boxplot(column='Final Price')

plt.title("BOX PLOT")

plt.ylabel('Laptop')

plt.show()

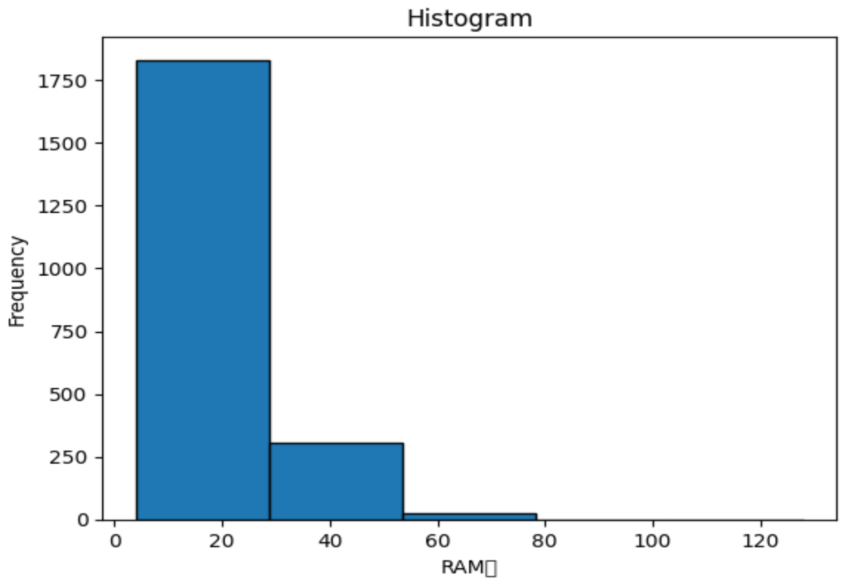
output:




```
plt.hist(df['RAM'],bins=5,edgecolor='black')
```

```
plt.title("Histogram")
plt.xlabel("RAM ")
plt.ylabel("Frequency")
plt.show()
```

output:



import pandas as pd

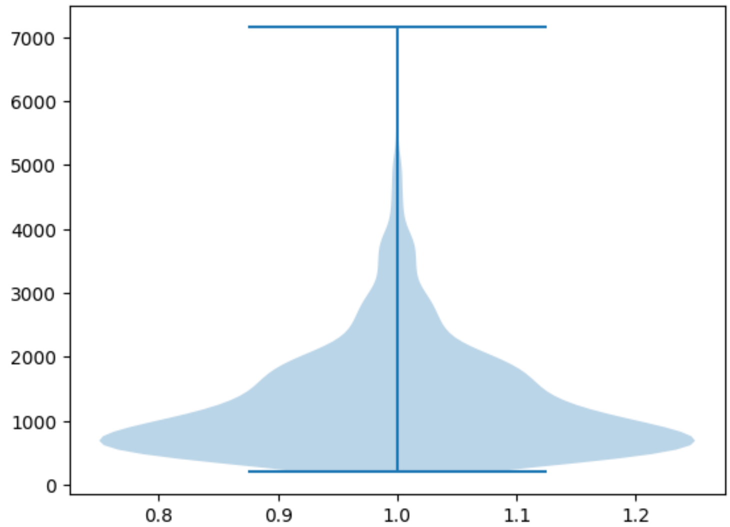
df=pd.read_csv('/content/archive (47).zip')

import matplotlib.pyplot as plt

plt.violinplot(df['Final Price'])

plt.show()

output:



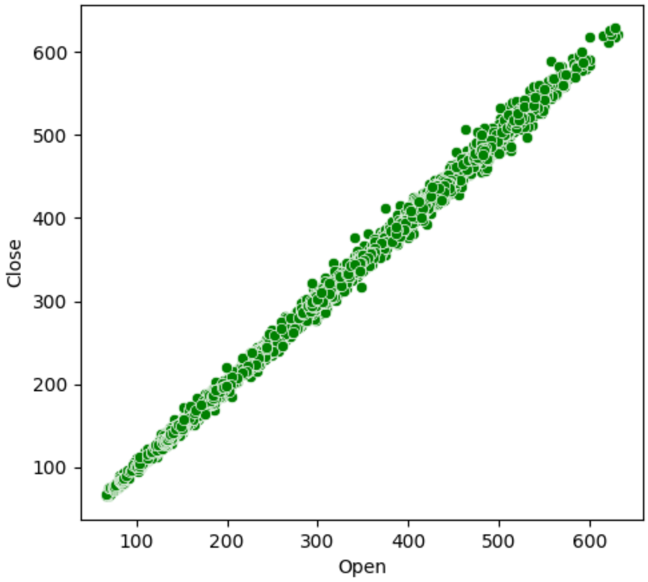
[redacted]

multivariate plots.

scatterplot

```
import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

d=pd.read_csv('/content/archive (49).zip')

plt.figure(figsize=(25,5))

plt.subplot(1,4,2)

sns.scatterplot(data=d, x='Open',y='Close',color='green')

plt.xlabel('Open')

plt.ylabel('Close')

plt.show()
```

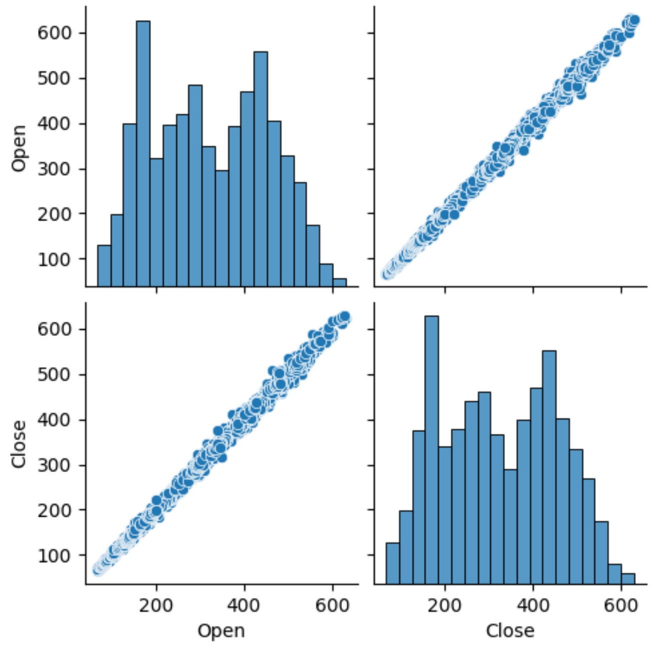
output:



pairplot

```
import pandas as pd
import seaborn as sns
df=pd.read_csv('/content/archive (49).zip')
import matplotlib.pyplot as plt
sns.pairplot(df,vars=['Open','Close'])
plt.show()
```

output:



KERNEL DENSITY PLOT

import pandas as pd

import seaborn as ssn

df=pd.read_csv('/content/archive (47).zip')

import matplotlib. pyplot as plt

ssn.kdeplot(df['Storage'])

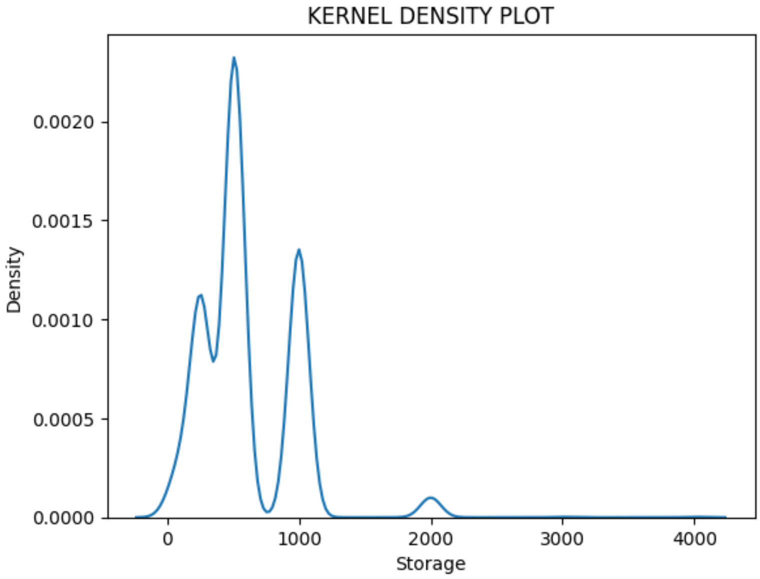
plt.title('KERNEL DENSITY PLOT')

plt.xlabel('Storage')

plt.ylabel('Density')

plt.show()

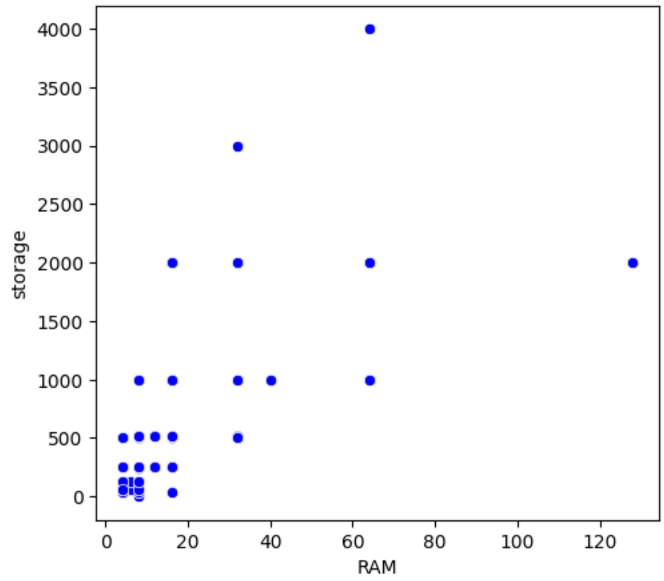
output:



scatterplot

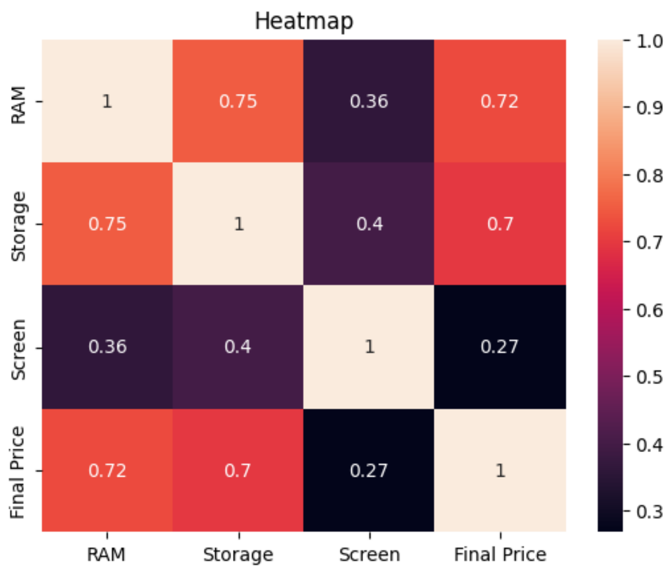
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
d=pd.read_csv('/content/archive (47).zip')
plt.figure(figsize=(25,5))
plt.subplot(1,4,2)
sns.scatterplot(data=d, x='RAM',y='Storage',color='blue')
plt.xlabel('RAM')
plt.ylabel('storage')
plt.show()
```

output:




```
import pandas as pd
import seaborn as sns
df=pd.read_csv('/content/archive (47).zip')
import matplotlib.pyplot as plt
corr_matrix = df.corr(numeric_only=True)
sns.heatmap(corr_matrix,annot=True)
plt.title("Heatmap")
plt.show()
```

output:



Perform any probability calculation

```
import pandas as pd
from scipy.stats import norm
#load the dataset
data=pd.read_csv('/content/archive (47).zip')
#calculate the mean &standard deviation of oncome
mean_Storage =data['Storage'].mean()
std_Storage=data['Storage'].std()
#define the value we're interested
value=10000
#calculate the probabality using the standard normal dustribution
z_score=(value-mean_Storage )/std_Storage
probabality=1-norm.cdf(z_score)
print(f'probablity  Storage greater than &{value}:{probabality:2%}')
```

output:

```
print(f'probablity  Storage greater than &{value}:{probabality:2%}')
```

Dataset:2=`content/archive (49).zip"`

- **MEAN**

```
import pandas as pd
path="/content/archive (49).zip"
df=pd.read_csv(path)
```

```
print(df)
```

output:

```
Open        High        Low        Close
0     153.988327  154.938126  150.386948  152.098587
1     148.803925  150.861862  140.730530  142.066193
2     143.461227  145.370758  138.959518  140.710739
3     139.503677  140.849243  132.874786  137.396286
4     136.080399  143.134735  134.230240  142.442169
...          ...         ...         ...         ...
3327  620.400024  620.849976  607.000000  612.099976
3328  614.000000  625.299988  610.150024  620.599976
3329  622.650024  623.599976  616.200012  621.650024
3330  621.000000  631.000000  617.700012  625.750000
3331  627.950012  634.450012  626.250000  629.250000

[3332 rows x 4 columns]
```

- **MODE**

```
df.mode()
```

Open	High	Low	Close	
0	262.187775	168.000000	435.0	433.200012
1	400.000000	172.199997	NaN	NaN
2	424.000000	175.000000	NaN	NaN
3	NaN	181.500000	NaN	NaN
4	NaN	269.113495	NaN	NaN
5	NaN	300.000000	NaN	NaN
6	NaN	395.000000	NaN	NaN
7	NaN	409.899994	NaN	NaN
8	NaN	441.450012	NaN	NaN
9	NaN	460.000000	NaN	NaN
10	NaN	475.000000	NaN	NaN

```
df.median()
Open     314.861069
High     320.000000
Low      309.291306
Close    314.787140
dtype: float64
```

```
df.std(numeric_only=True)
Open       131.120350
High       132.215823
Low        129.707901
Close      130.896514
dtype: float64

df.describe()
```

	Open	High	Low	Close
count	3332.000000	3332.000000	3332.000000	3332.000000
mean	323.548587	328.159113	318.401506	323.086718
std	131.120350	132.215823	129.707901	130.896514
min	66.500000	66.900002	63.500000	65.300003
25%	203.393403	206.420929	200.363395	202.943359
50%	314.861069	320.000000	309.291306	314.787140
75%	433.388901	437.924996	428.012497	432.812492
max	629.950012	634.799988	626.250000	629.250000

RANGE

```
import pandas as pd
path=("/content/archive (49).zip")
df=pd.read_csv(path)
numeric_columns = df.select_dtypes(include=['number'])
range_value = numeric_columns.max() - numeric_columns.min()
print(range_value)
Open        563.450012
High        567.899986
Low         562.750000
Close       563.949997
D type: float64
```

- **Univariate tests for the dataset.**

one sample t test

```
import numpy as np one sample
t_test:3.6115755925730757,p_value:0.005645419042037466
from scipy import stats
d=pd.read_csv('/content/archive (49).zip')
Open=np.array([3332.000000,323.548587,131,120350,66.500000])
t_stat,p_value= stats.ttest_1samp(RAM,popmean=7.2)
```

```
print( f"one sample t_test:{t_stat},p_value:{p_value}")
```

OUTPUT:

one sample t_test:3.6115755925730757,p_value:0.005645419042037466

chi_square

```
import pandas as pd
from scipy.stats import chi2_contingency
data =pd.read_csv('/content/archive (49).zip')
cantingency_table = pd.crosstab(data['High'], data['Low'])
chi2, p, dof, ex = chi2_contingency(cantingency_table)
print(f"Chi-square Test of Independence:
chi2={chi2},p={p},dof={dof},expected={ex}")
```

OUTPUT:

```
Chi-square Test of Independence:
chi2=8645688.488888884,p=2.493088990572916e-63,dof=8576100,expected=[[0
.00030012 0.00030012 0.00030012 ... 0.00030012 0.00030012 0.00030012]
 [0.00030012 0.00030012 0.00030012 ... 0.00030012 0.00030012 0.00030012]
 [0.00030012 0.00030012 0.00030012 ... 0.00030012 0.00030012 0.00030012]
 ...
 [0.00030012 0.00030012 0.00030012 ... 0.00030012 0.00030012 0.00030012]
 [0.00030012 0.00030012 0.00030012 ... 0.00030012 0.00030012 0.00030012]
 [0.00030012 0.00030012 0.00030012 ... 0.00030012 0.00030012 0.00030012]]
```

Anova

```
import pandas as pd
from scipy.stats import f_oneway
d=pd.read_csv('/content/archive (49).zip')
groups = [d['High'],d['Low'], d['Open']]
f_stat, p_value = f_oneway(*groups)
print(f"F-valuei:{f_stat},p-value:{p_value}")
```

output:

```
F-valuei:4.624901950203641,p-value:0.009825612325109255
```

Wilcoxon

```
import pandas as pd
import numpy as np
from scipy.stats import wilcoxon
d=pd.read_csv('/content/archive (49).zip')
data={
      'before':[3332.000000,323.548587,131,120350,66.500000],
       'after':[2002.44,434455,5433466,34566,3455]
}
```

```
df=pd.DataFrame(data)
stat,p_value=wilcoxon(df['before'],df['after'])
print(f"wilcoxon signed-rank statistics:{stat},p_value:{p_value}")
```

output

```
wilcoxon signed-rank statistics:4.0,p_value:0.4375
```

kruskal

```
import pandas as pd
from scipy.stats import kruskal
import numpy as np
d=pd.read_csv('/content/archive (49).zip')
d={
    'group1':np.random.normal(loc=0,scale=1,size=10),
    'group2':np.random.normal(loc=0,scale=1,size=10),
    'group3':np.random.normal(loc=0,scale=1,size=10)
}
f_stat,p_value=kruskal(d['group1'],d['group2'],d['group3'])
print(f"kruskal-walls H statistics :{f_stat},p-value:{p_value}")
```

output:

```
kruskal-walls H statistics :0.9599999999999937,p-value:0.6187833918061427
```

.b)Analyze the multivariate of the dataset Ex-co-variance,co-relation

df.var(numeric_only=True)

output:

```
Open       17192.546192
High       17481.023919
Low        16824.139578
Close      17133.897352
dtype: float64
```

```
df.corr(numeric_only=True)
```

output:

pen	High	Low	Close	
Open	1.000000	0.999391	0.999339	0.998726
High	0.999391	1.000000	0.999286	0.999548
Low	0.999339	0.999286	1.000000	0.999440
Close	0.998726	0.999548	0.999440	1.000000

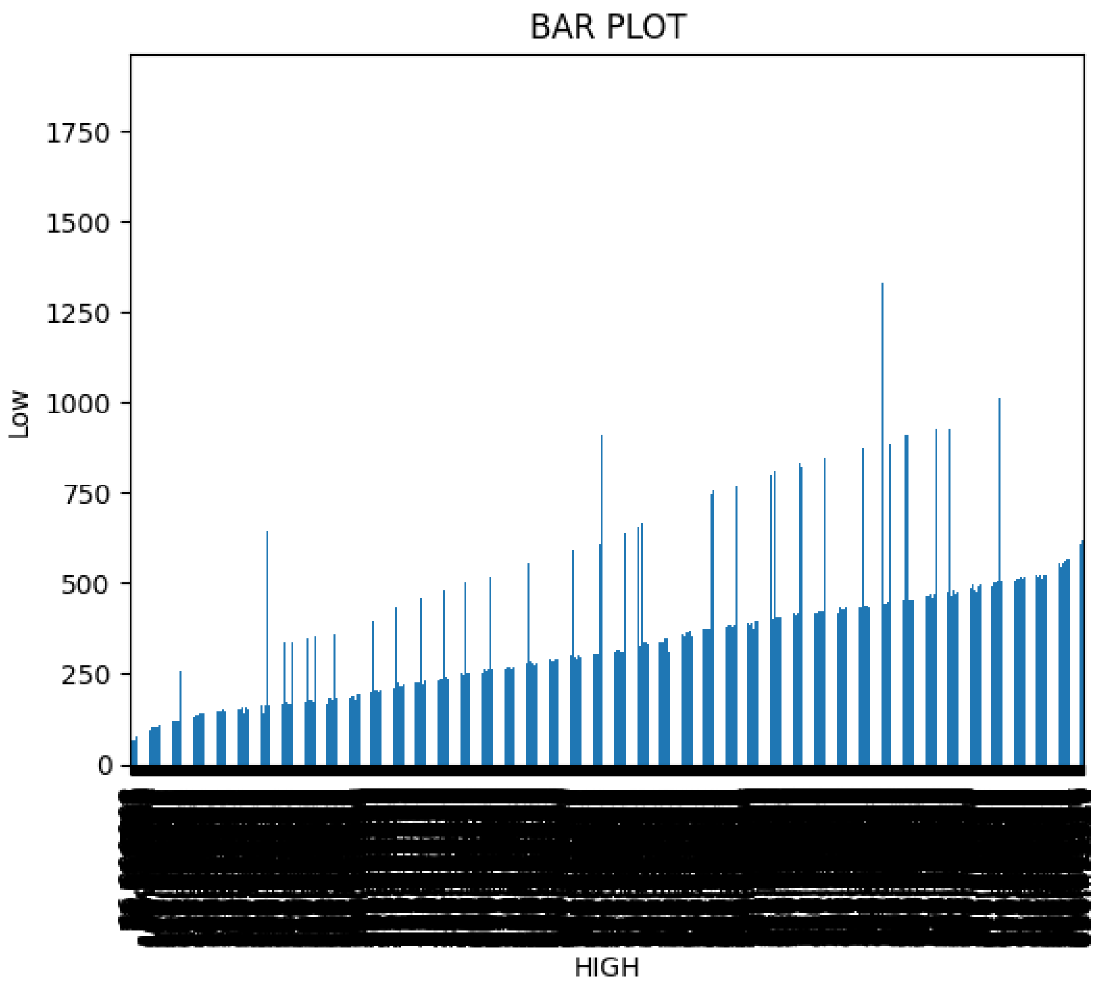
C)**Visualize the univariate and multivariate with various plots.**

various plots.

BAR PLOT

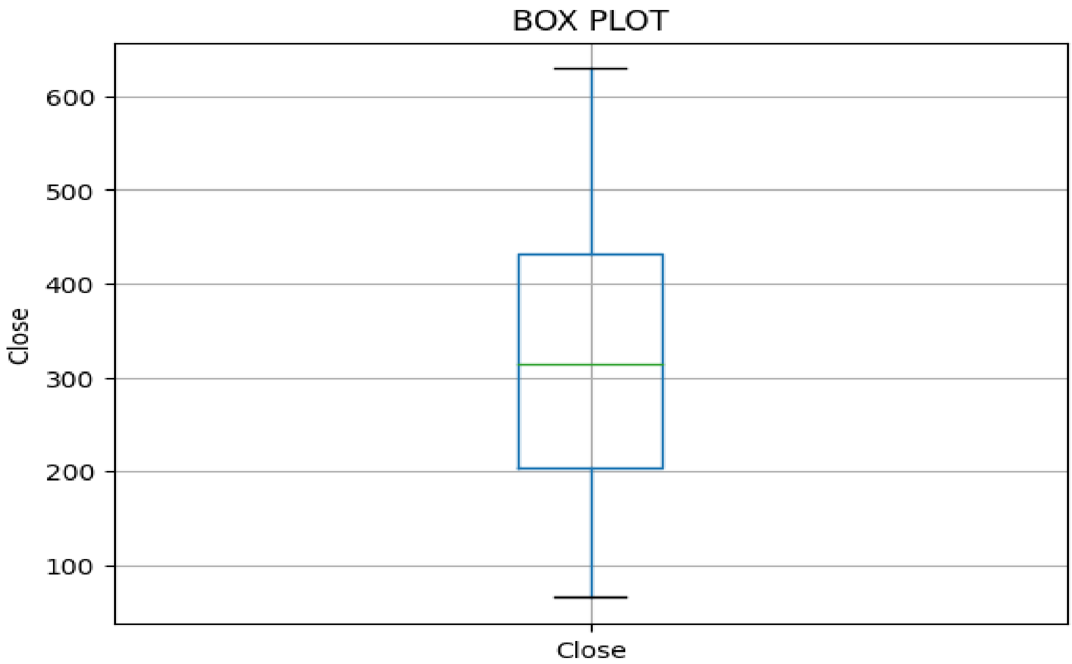
```
import pandas as pd
df=pd.read_csv('/content/archive (49).zip')
import matplotlib.pyplot as plt
df.groupby('High')['Low'].sum().plot(kind='bar')
plt.title("BAR PLOT")
plt.xlabel("HIGH")
plt.ylabel("Low")
plt.show()
```

output:



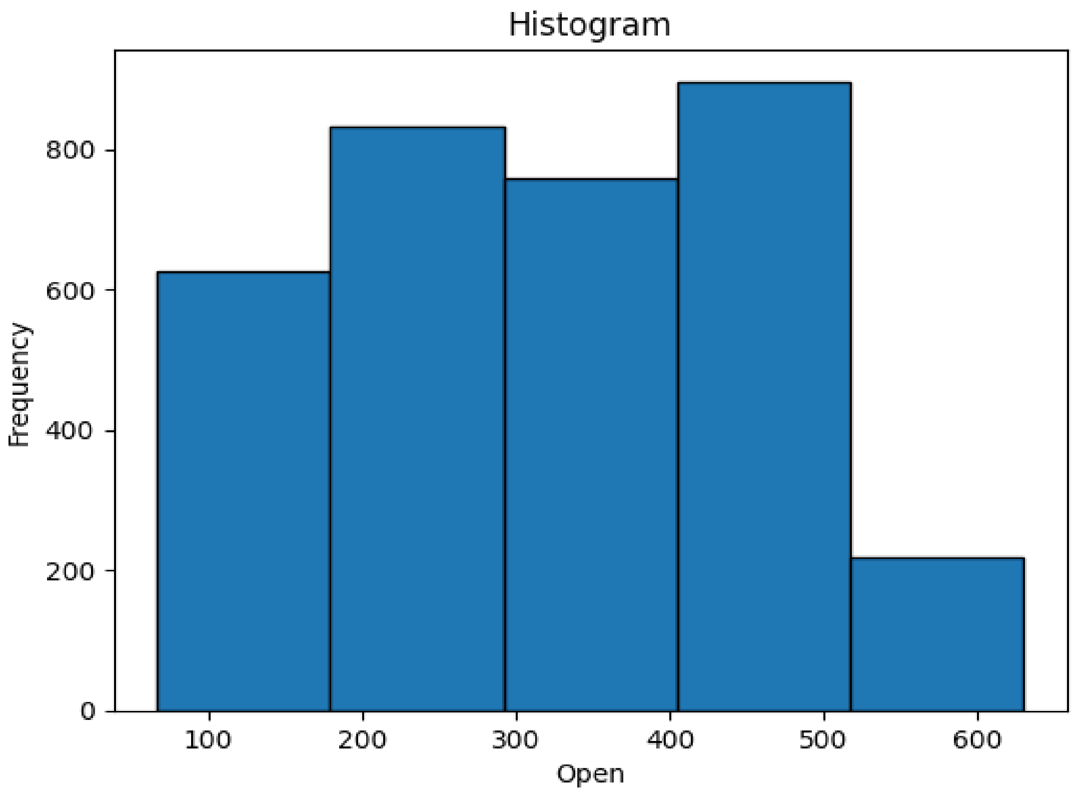

```
df.boxplot(column='Close')
plt.title("BOX PLOT")
plt.ylabel('Close')
plt.show()
```

output:



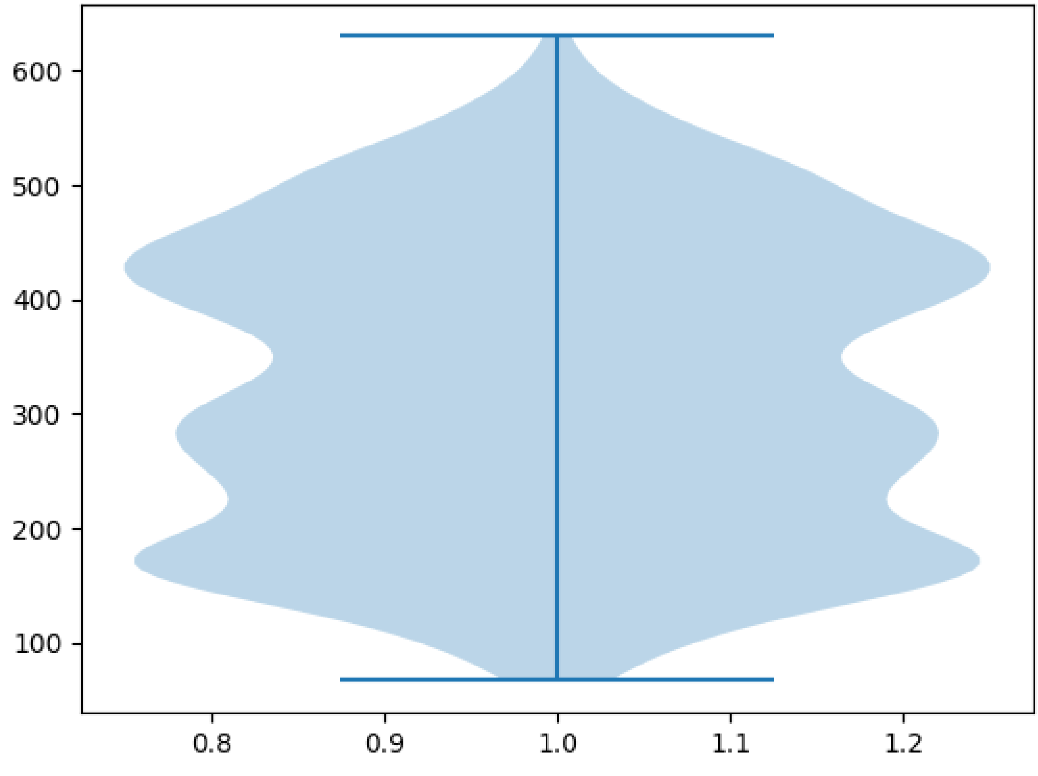

```
plt.hist(df['Open'],bins=5,edgecolor='black')
plt.title("Histogram")
plt.xlabel("Open")
plt.ylabel("Frequency")
plt.show()
```

output:



```python
import pandas as pd
df=pd.read_csv('/content/archive (49).zip')
import matplotlib.pyplot as plt
plt.violinplot(df['Open'])
plt.show()
```

output:



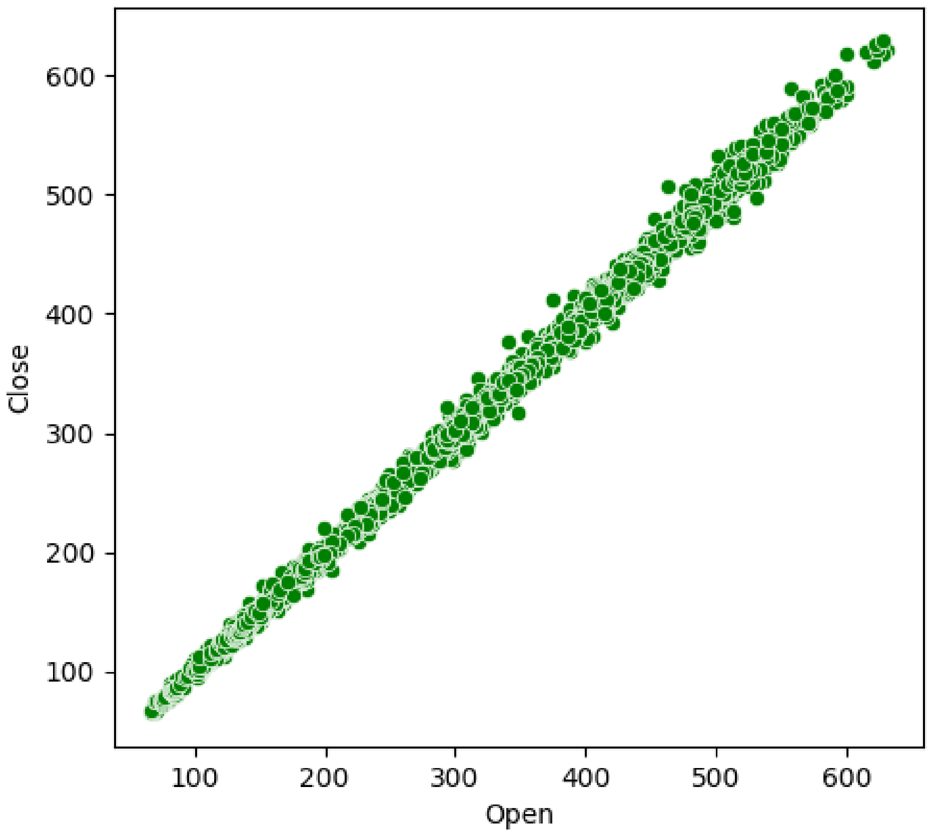
multivariate plots.

scatterplot

```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
d=pd.read_csv('/content/archive (49).zip')
```

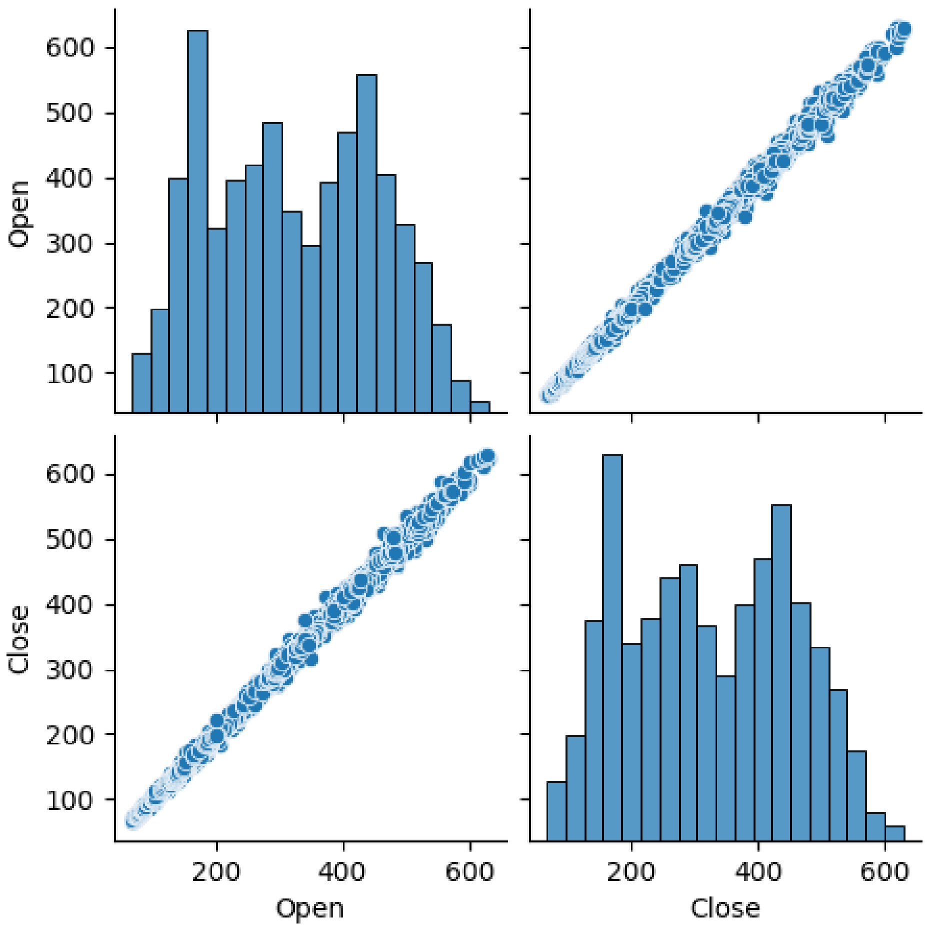
```
plt.figure(figsize=(25,5))
plt.subplot(1,4,2)
sns.scatterplot(data=d, x='Open',y='Close',color='green')
plt.xlabel('Open')
plt.ylabel('Close')
plt.show()
```

output:



```python
import pandas as pd
import seaborn as sns
df=pd.read_csv('/content/archive (49).zip')
import matplotlib.pyplot as plt
sns.pairplot(df,vars=['Open','Close'])
plt.show()
```

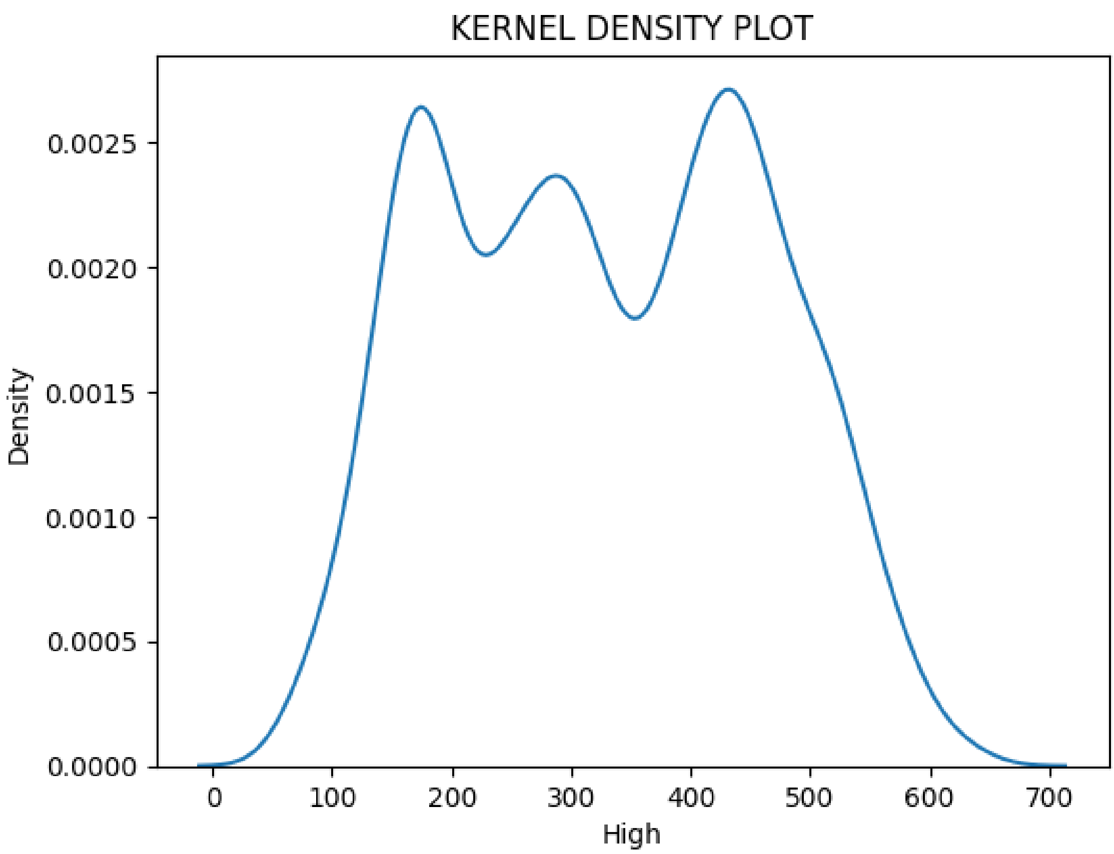
output:




```python
import pandas as pd
import seaborn as ssn
df=pd.read_csv('/content/archive (49).zip')
import matplotlib. pyplot as plt
```

```
ssn.kdeplot(df['High'])
plt.title('KERNEL DENSITY PLOT')
plt.xlabel('High')
plt.ylabel('Density')
plt.show()
```

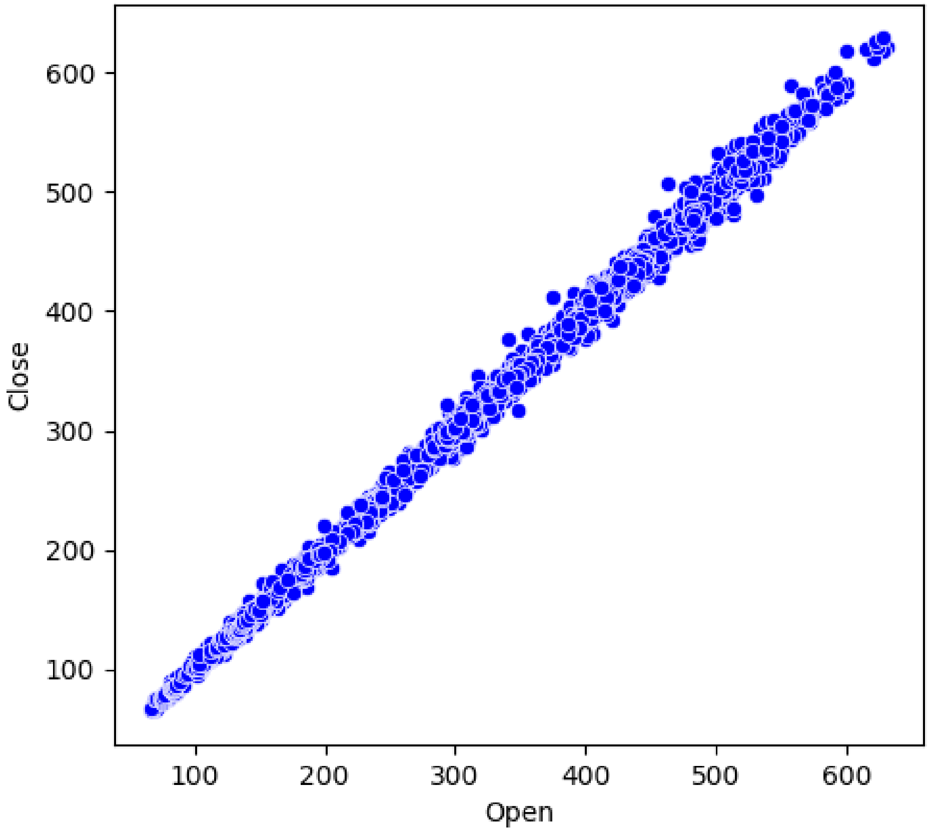
output:



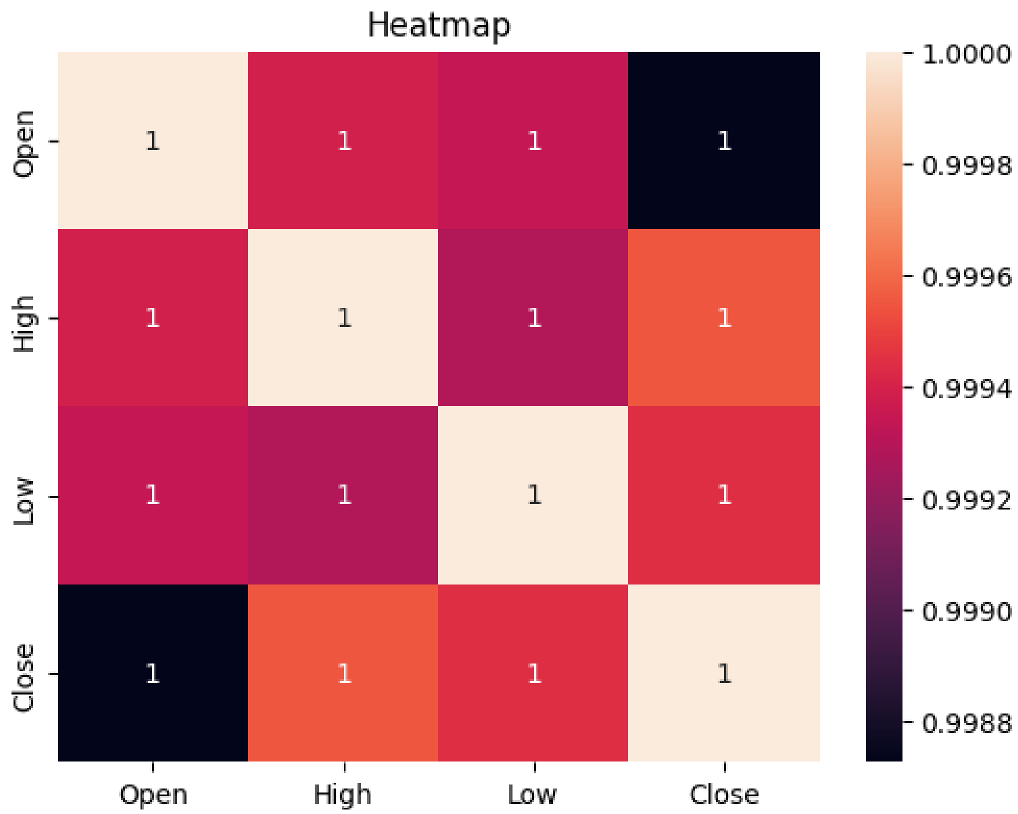

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
d=pd.read_csv('/content/archive (49).zip')
plt.figure(figsize=(25,5))
plt.subplot(1,4,2)
```

```
sns.scatterplot(data=d, x='Open',y='Close',color='blue')
plt.xlabel('Open')
plt.ylabel('Close')
plt.show()
```

output:




```
import pandas as pd
import seaborn as sns
df=pd.read_csv('/content/archive (49).zip')
import matplotlib.pyplot as plt
corr_matrix = df.corr(numeric_only=True)
sns.heatmap(corr_matrix,annot=True)
plt.title("Heatmap")
plt.show()
```
output:



Perform any probability calculation

```
import pandas as pd
from scipy.stats import norm
#load the dataset
data=pd.read_csv('/content/archive (49).zip')
#calculate the mean &standard deviation of oncome
mean_Open=data['Open'].mean()
std_Open=data['Open'].std()
#define the value we're interested
value=50000
#calculate the probabality using the standard normal dustribution
z_score=(value-mean_Open)/std_Open
probabality=1-norm.cdf(z_score)
print(f'probablity Open greater than &{value}:{probabality:2%}')
```

output:

```
probablity Open greater than &50000:0.000000%
```

[redacted]