INDEX

Name: Rayan Sanjai KS Roll No: 3122 01 A53

Std CSE Sub/Div: EDS School/College

Sl No.	Date	Title	Page No	Remarks
1.a	17/7/25	Line Plot	10	
1.b	17/7/25	Bar chart		
1.c	17/7/25	Pie chart		
1.d	17/7/25	Scatterplot		
1.e	17/7/25	Histogram		
2.	31/7/25	Data Analysis and Visulization	10	
3.	7/8/25	Data Preprocessing	10	
4.	7/8/25	Handling Missing and Inappropriate Data in a Data set	10	
5.	14/8/25	Outliers	10	
6.	14/8/25	Feature Scaling	10	
7.	28/8/25	EDA	10	
8.	18/9/25	Linear Regression	10	
9.		Logistic Regression	10	
10.		K-Mean clustering	10	
11.		KNN	10	
12.		T-test	10	
13.		Z-test	10	
14.		Anova test	10	

Complete

India Vs Srilanga
400
375
350
325
300
275
250
225
200
score
10
20
30
40
50
overs

Ex: 1(a) Lineplot Using matplotlib

Aim:

To obtain Lineplot for the data using matplotlib library.

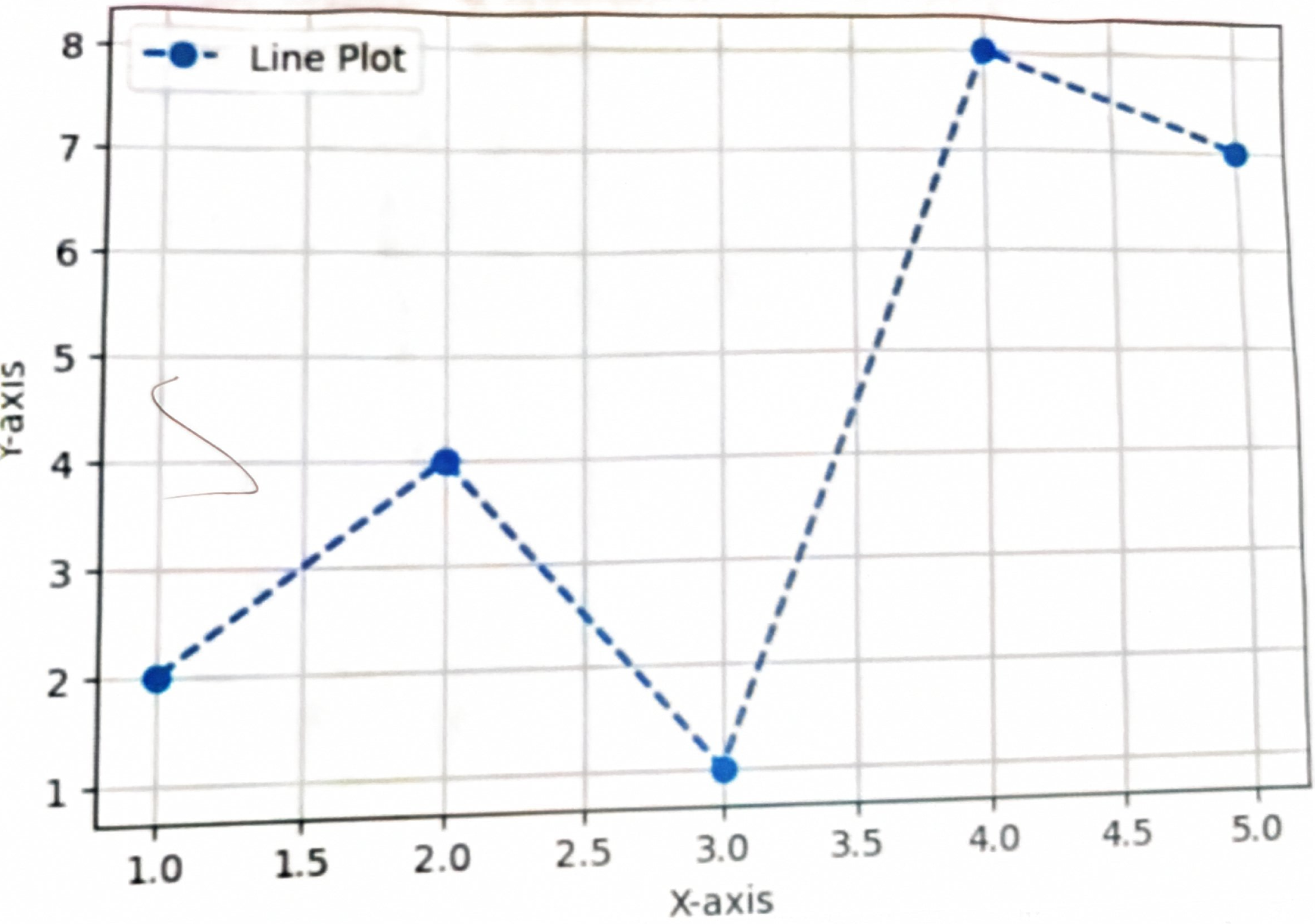
Code:

```
import matplotlib.pyplot as plt
overs = list(range(5,51,5))
India_Score =[400,360,270,365,276,298,350,245,345,299]
Srilanga_Score =[394,367,263,300,277,290,355,202,304,243]
plt.plot(overs, India_Score,'color'=='blue')
plt.plot(overs, Srilanga_Score)
plt.show()
plt.title("India Vs Srilanga")
plt.xlabel("overs")
plt.ylabel("score")
plt.legend()
plt.plot(overs, India_Score, color="blue", label="India")
plt.plot(overs, Srilanga_Score, color="green", label="Srilanga")
```

07/08/25

Result:

Lineplot using matplotlib library is obtained successfully.


Line Plot
8
7
6
5
4
3
2
1
Y-axis
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
X-axis

Ex: 1(b) ~~Bar chart~~ Line Plot Using matplotlib

Aim:

To obtain ~~Bar chart~~ Line Plot for the data using matplotlib library

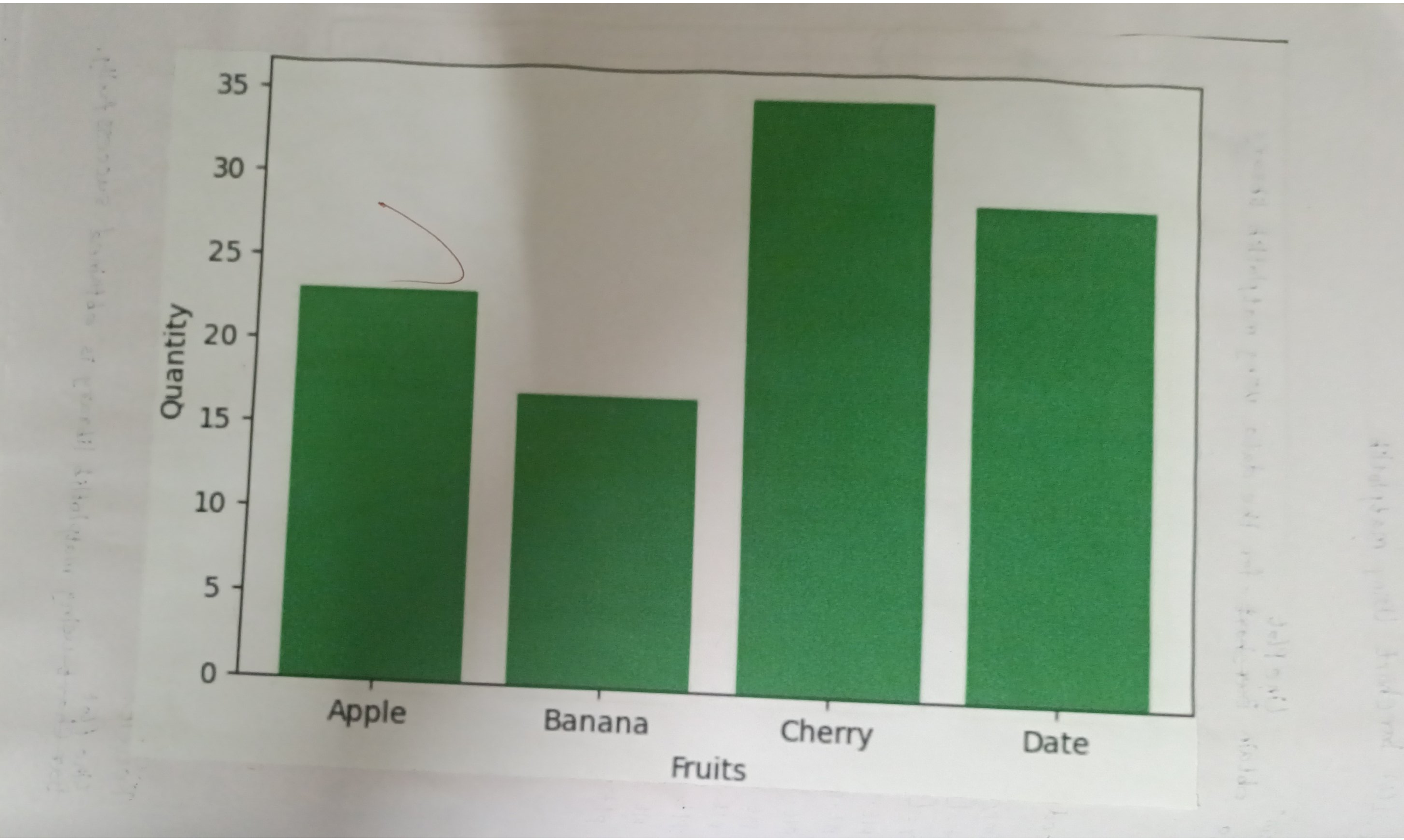
code:

```
import matplotlib.pyplot as plt
x = [1, 2, 3, 4, 5]
y = [2, 4, 1, 8, 7]
plt.figure (figsize = (6, 4))
plt.plot (x, y, color = 'blue', marker = 'o', linestyle = '--', label = 'Line Plot')
plt.title ("Line Plot")
plt.xlabel ("X-axis")
plt.ylabel ("Y-axis")
plt.legend ()
plt.grid (True)
plt.show()
```

07108

Result:

Line Plot
~~Bar Chart~~ using matplotlib library is obtained successfully.


35
30
25
20
15
10
5
0
Quantity
Apple
Banana
Cherry
Date
Fruits

Ex: 1(c) Bar Chart Using matplotlib

Aim:

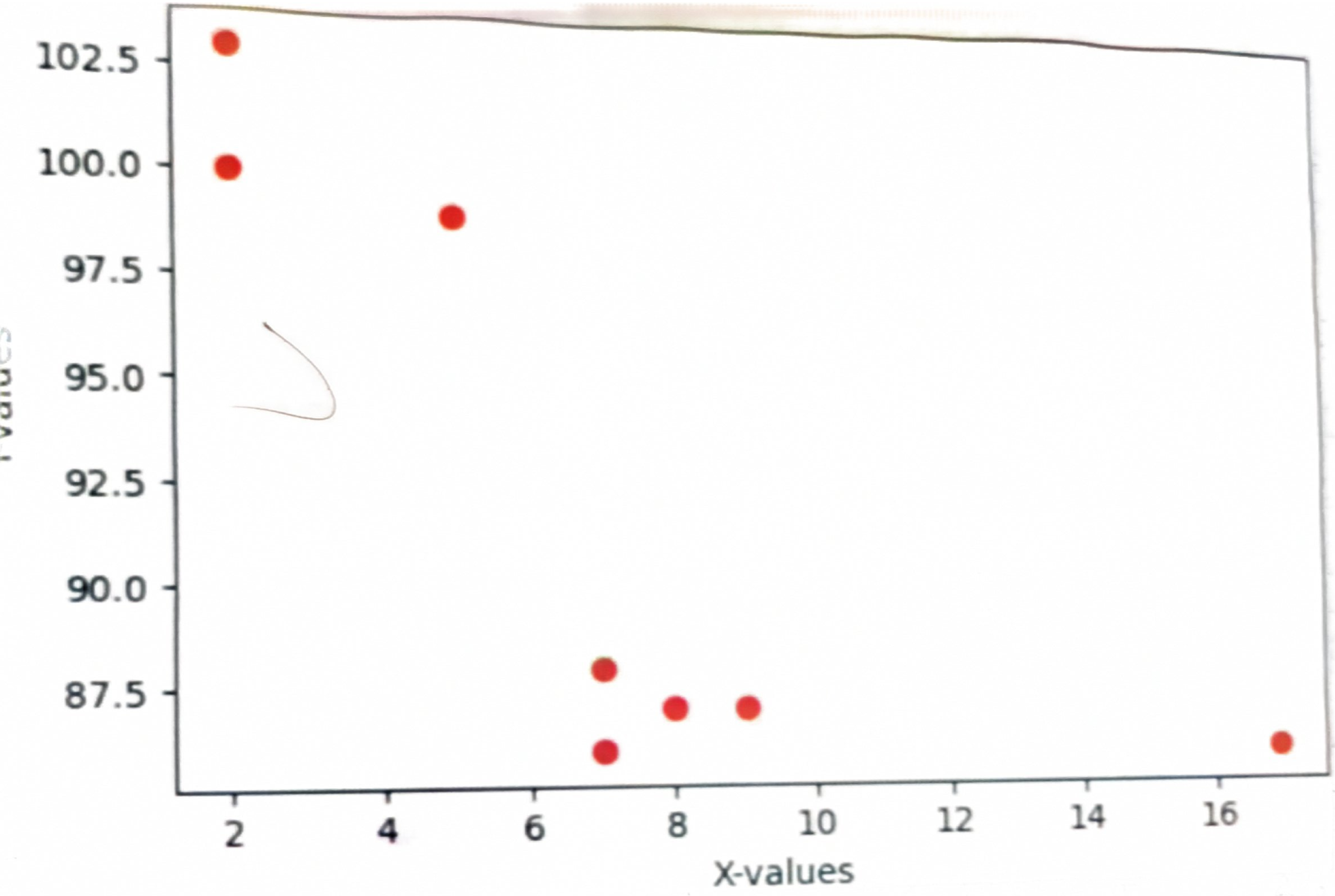
To obtain Bar chart for the data using matplotlib library

Code:

```
categories=['Apple','Banana','cherry','Date']
values =[23,17,35,29]
plt.figure(figsize =(6,4))
plt.bar (categories, values, color='green')
plt.title("Bar Chart")
plt.xlabel("Fruits")
plt.ylabel("Quantity")
plt.show()
```

Result:

Bar Chart using matplotlib library is obtained successfully.

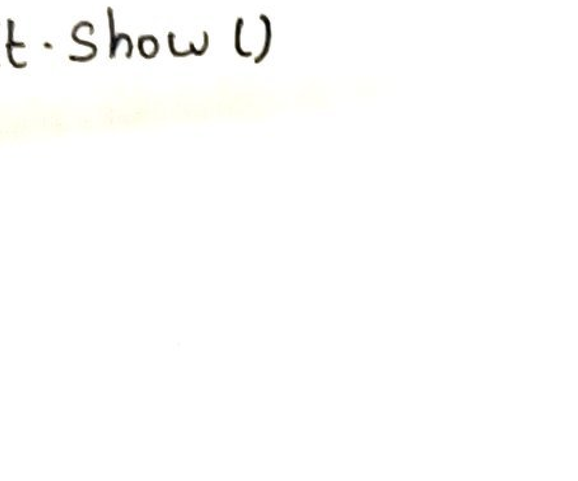

102.5
100.0
97.5
95.0
92.5
90.0
87.5
2
4
6
8
10
12
14
16
X-values

Ex: 1(d) Scatter Plot

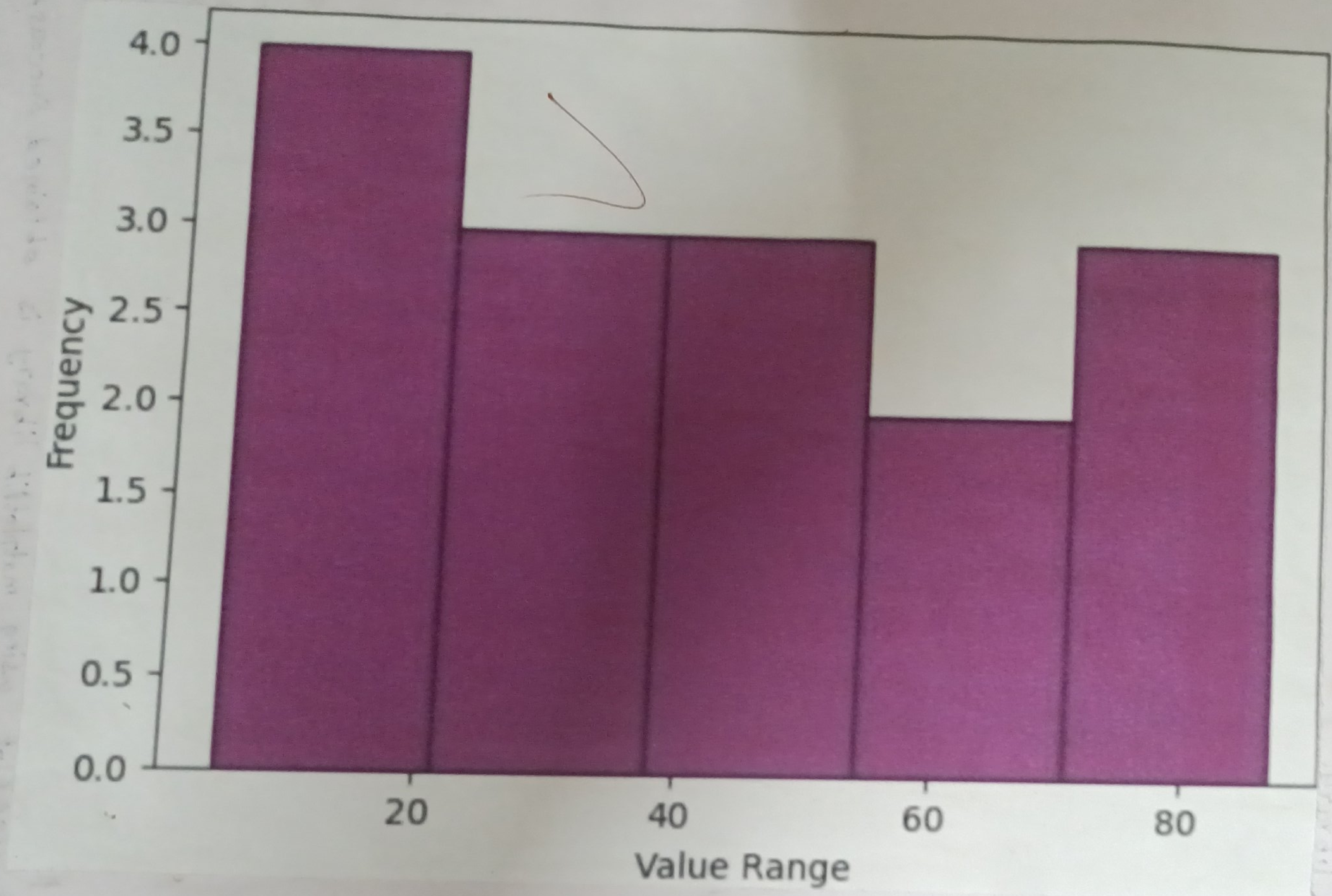
Aim:
To obtain scatter plot for data using matplotlib library.

Code:

```
x_scatter = [5, 7, 8, 7, 2, 17, 2, 9]
y_scatter = [99.86, 87, 88, 100, 86, 103, 87]
plt.figure(figsize=(6,4))
plt.scatter(x_scatter, y_scatter, color='red')
plt.title("Scatter Plot")
plt.xlabel("x-values")
plt.ylabel("y-values")
plt.show()
```



Result:
Scatter plot using matplotlib library is obtained successfully.


Frequency
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
20
40
60
80
Value Range

Ex: 1(e) Histogram

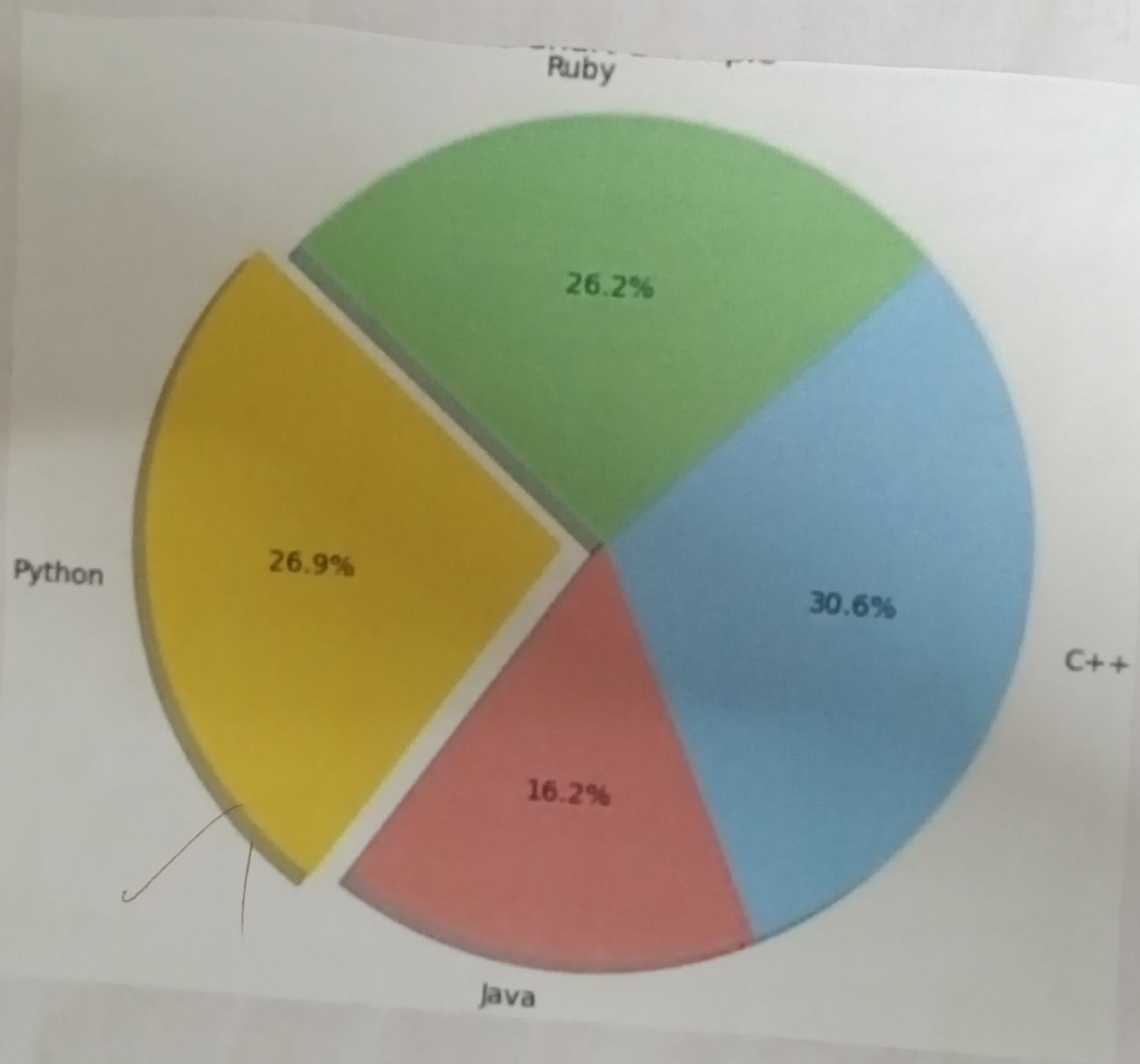
To obtain Histogram for data using matplotHib library

Code:

```
data = [22, 87, 5, 43, 56, 73, 55, 54, 11, 20, 51, 5, 79, 31, 27]
plt.figure (figsize =(6,4))
plt.hist (data, bins = 5, color='purple', edgecolor = 'black')
plt.title ("Histogram")
plt.xlabel ("Value Range")
plt.ylabel ("Frequency")
plt.show ()
```

Result:

Histogram using matplotlib library is obtained


Ruby
26.2%
Python
26.9%
30.6%
C++
16.2%
Java

3. (i) Pie chart

Aim:

To obtain pie chart for the data using matplotlib library.

Code:

```
labels = ['Python', 'Java', 'C++', 'Ruby']
sizes = [215, 130, 245, 210]
colors = ['gold', 'lightcoral', 'lightskyblue', 'lightgreen']
explode = (0, 1, 0, 0, 0)
plt.figure(figsize = (6,6))
plt.pie(sizes, explode = explode, labels = labels, colors, autopct = '%1.
1f %%', shadow = True, startangle = 140)
plt.title("Pie Chart")
plt.axis('equal')
plt.show()
```

10/10

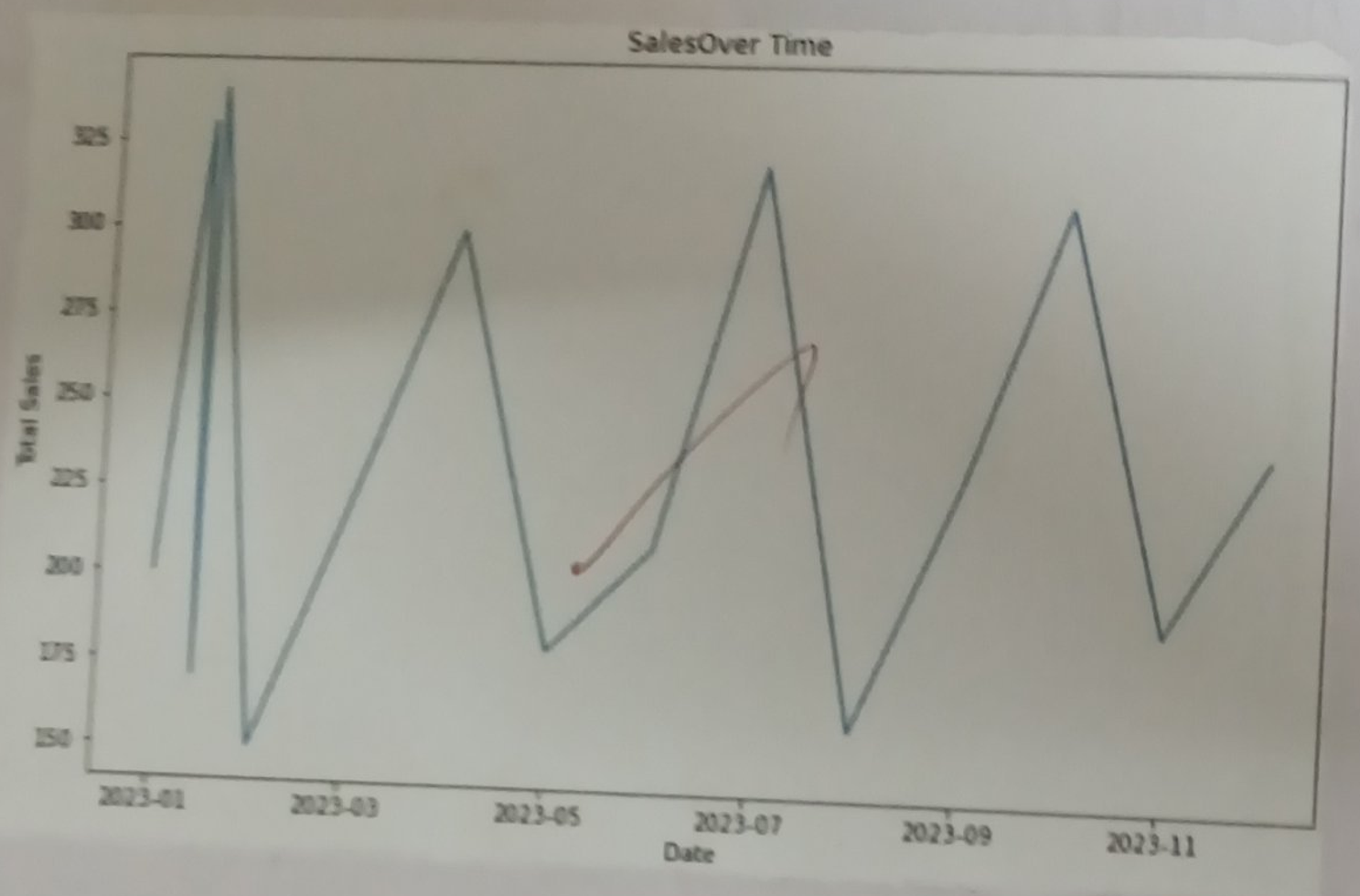
Result:

Pie chart using matplotlib library is obtained successfully.

```
Date Product Sales Quantity Region
0  01-01-2023  Product A     200        4  North
1  02-01-2023  Product B     150        3  South
2  03-01-2023  Product A     220        5  North
3  04-01-2023  Product C     300        6   East
4  05-01-2023  Product B     180        4   West
Date        0
Product     0
Sales       0
Quantity    0
Region      0
dtype: int64
```

Product	Product A	Product B	Product C
Region			
East	0	0	1600
North	1350	0	0
South	0	480	0
West	0	370	0

	Sales	Quantity
Sales	1.000000	0.944922
Quantity	0.944922	1.000000



Ex. 2 Data Analysis and Visualisation

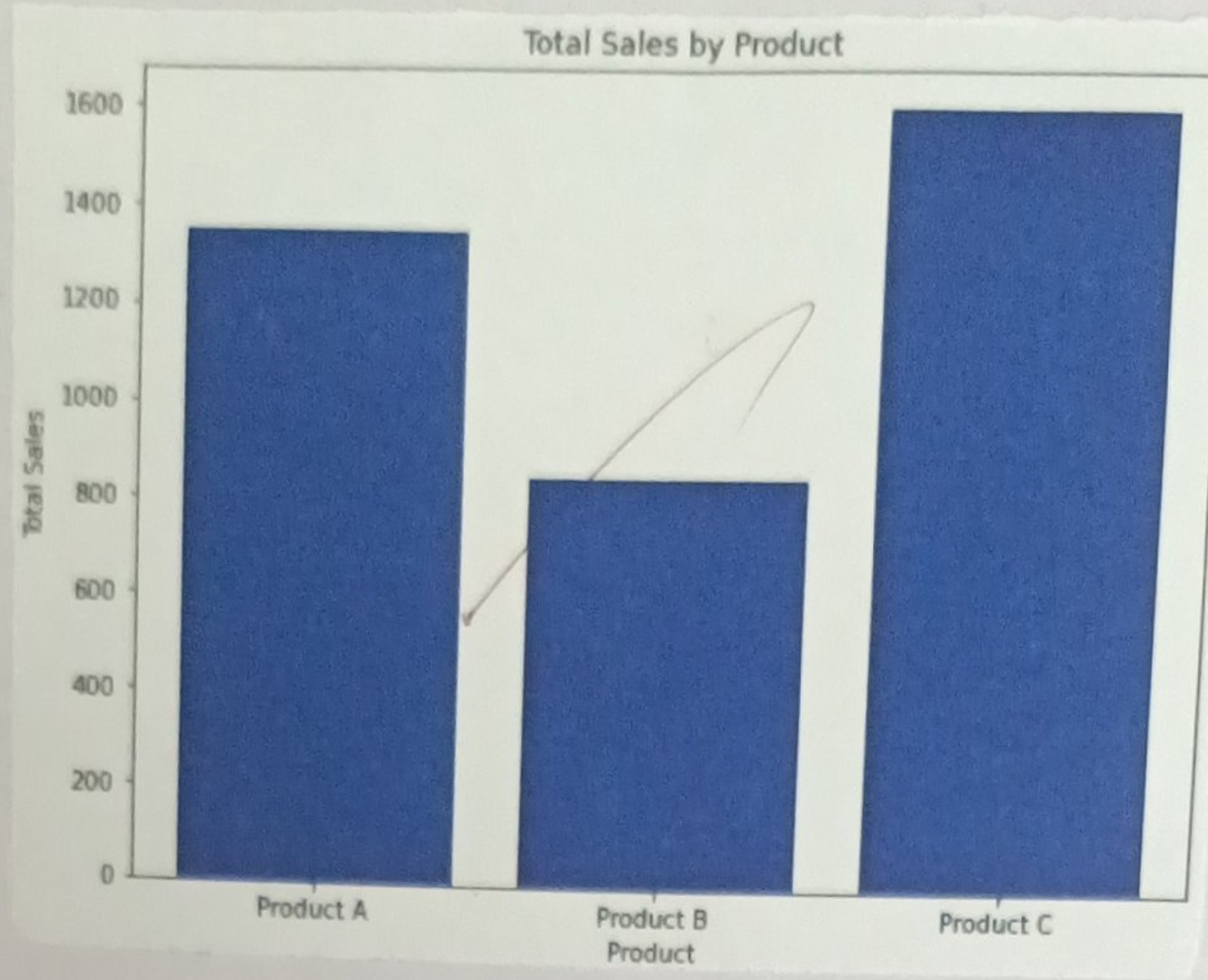
Description:

Upload and analysis the data set given in csv format and perform data preprocessing and visualization. Use sample data sales_data csv.

code:

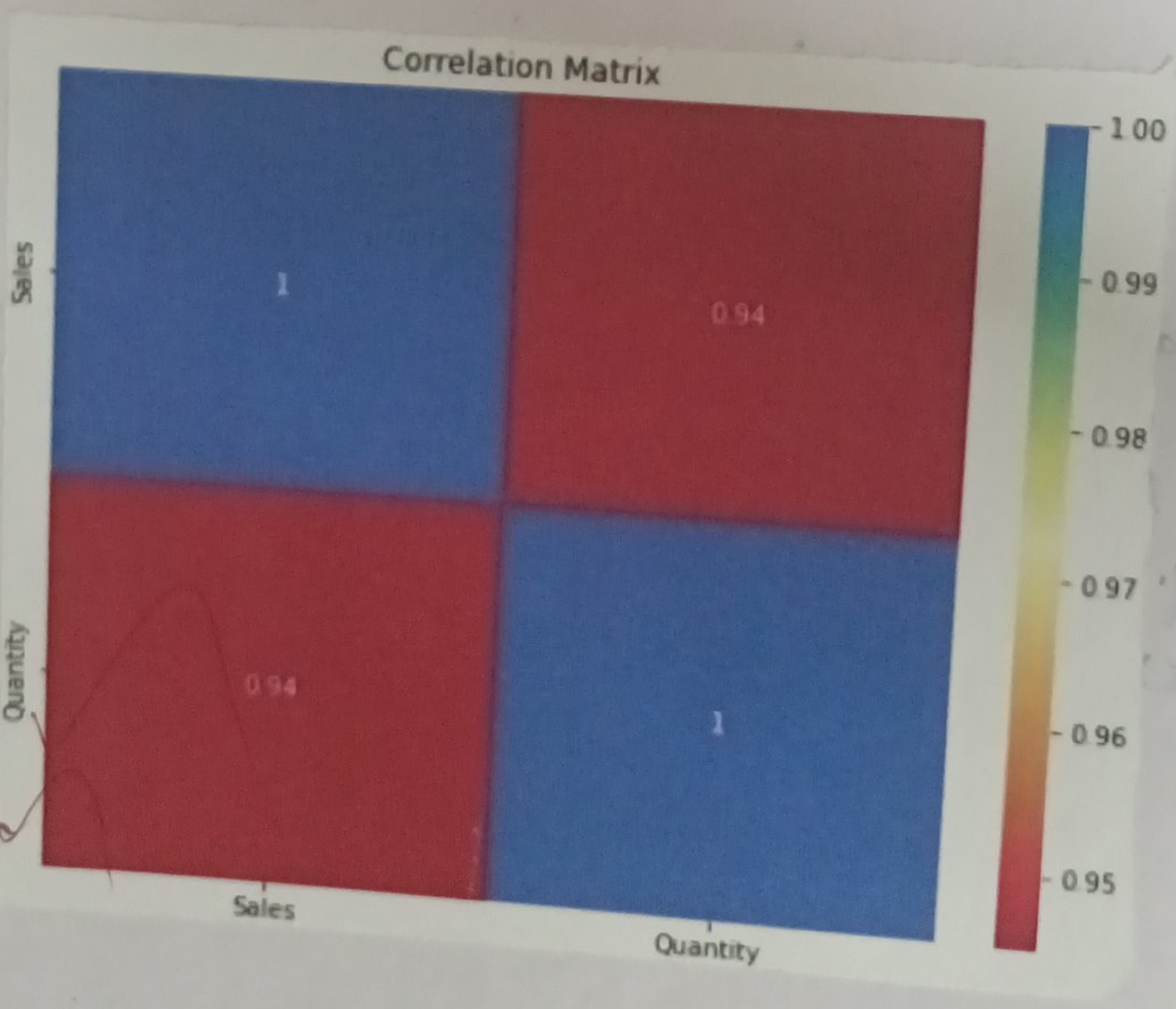
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
path ='sales_data.csv'
df = pd.read_csv(path)
print (df.head())
print (df.isnull().sum())
df['Sales'].fillna (df['Sales'].mean()/ inplace=True)
df.dropna (subset = ['Product', 'Quantity', 'Region'], inplace = True)
print(df.describe())
product_summary = df.groupby('Product').agg ({'Sales':'sum', 'Quantity':
                                                'sum'}).reset.index
print (Product_summary)

plt.figure (figsize=(10,6))
plt.bar(Product_summary['Product'], Product_summary ['sales'])
plt.xlabel ('Product')
plt.ylabel ('Total Sales')
plt.title ('Total sales by product')
plt.show()
```



	Sales	Quantity
count	16.000000	16.000000
mean	237.500000	5.375000
std	64.031242	1.746425
min	150.000000	3.000000
25%	187.500000	4.000000
50%	225.000000	5.500000
75%	302.500000	7.000000
max	340.000000	8.000000

	Product	Sales	Quantity
0	Product A	1350	33
1	Product B	850	17
2	Product C	1600	36



```
df['Date'] =pd.to_datetime(df['Date'])
sales_over_time=df.groupby('Date').agg({'Sales':'sum'}).reset_index()
plt.figure(figsize=(10,6))
plt.plot(sales_over_time['Data'],sales_over_time['Sales'])
plt.xlabel('Data')
plt.ylabel('Total Sales')
plt.title('Sales over time')
plt.show()

pivot_table = df.pivot_table(values='Sales',index='Region',column=
'product', aggfunc=np.sum,fill_value=0)

print(correlation_matrix)

import seaborn as sns
plt.figure[figsize=(8,6))
sns.heatmpap(correlation_matrix, annot=True, cmap='cool')
plt.title['Correlation Matrix')
plt.show()
```

10/10

07/10/25

Result:

The Data is analysed and performed Data performed Data preprocessing and visulisation.

Ex: 3 Data Preprocessing

Aim:

Experiment to understand the data preprocessing in Data Science.

code

```
#Rayvan Sanjai KS
#240701425
# 7/8/25
# Data Preprocessing

import numpy as np
import pandas as pd
df =pd.read_csv("Pre-process - datasample.csv")
df
```

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	NaN	Yes
5	France	35.0	58000.0	Yes
6	Spain	NaN	52000.0	No
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No
9	France	37.0	67000.0	Yes

df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Country    10 non-null     object
 1   Age        9 non-null      float64
 2   Salary     9 non-null      float64
 3   Purchased  10 non-null     object
dtypes: float64(2), object(2)
memory usage: 448.0+ bytes
```

```
df.Country.fillna(df.Country.mode()[0], inplace = True
df.Age.fillna(df.Age.median(), inplace = True)
df.Salary.fillna(round(df.Salary.mean()), inplace=True)
df
```

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	63778.0	Yes
5	France	35.0	58000.0	Yes
6	Spain	38.0	52000.0	No
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No
9	France	37.0	67000.0	Yes

```
Pd.get_dummies(df.Country)
```

	France	Germany	Spain
0	1	0	0
1	0	0	1
2	0	1	0
3	0	0	1
4	0	1	0
5	1	0	0
6	0	0	1
7	1	0	0
8	0	1	0
9	1	0	0

```
Updated_dataset = Pd.Concat([Pd.get_dummies(df.country), df.iloc[:,[1,2,3]]],
                            axis =1)

updated_dataset
```

	France	Germany	Spain	Age	Salary	Purchased
0		1	0	0	44.0	72000.0 No
1		0	0	1	27.0	48000.0 Yes
2		0	1	0	30.0	54000.0 No
3		0	0	1	38.0	61000.0 No
4		0	1	0	40.0	63778.0 Yes
5		1	0	0	35.0	58000.0 Yes
6		0	0	1	38.0	52000.0 No
7		1	0	0	48.0	79000.0 Yes
8		0	1	0	50.0	83000.0 No
9		1	0	0	37.0	67000.0 Yes

df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Country     10 non-null     object
 1   Age         10 non-null     float64
 2   Salary      10 non-null     float64
 3   Purchased   10 non-null     object
dtypes: float64(2), object(2)
memory usage: 448.0+ bytes
```

updated_dataset.Purchased.replace(['NO','yes'],[0,1],inplace=True)

updated_dataset

	France	Germany	Spain	Age	Salary	Purchased
0		Yes	No	No	44.0	72000.0 0
1		No	No	Yes	27.0	48000.0 1
2		No	Yes	No	30.0	54000.0 0
3		No	No	Yes	38.0	61000.0 0
4		No	Yes	No	40.0	63778.0 1
5		Yes	No	No	35.0	58000.0 1
6		No	No	Yes	38.0	52000.0 0
7		Yes	No	No	48.0	79000.0 1
8		No	Yes	No	50 0	83000.0 0
9		Yes	No	No	37.0	67000.0 1

Result: Data preprocessing in data science has been analyzed.

Handling Missing and Inappropriate Data in Dataset

Ex: 4

Aim:

code

```
#Rayvan Sanjai K.S
#240701425
#7.08.25
#Handling Missing and Inappropriate Data
import numpy as np
import pandas as pd
df = pd.read_csv("Hotel_Dataset.csv")
df
```

	CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill	NoOfPax	EstimatedSalary
0	1	20-25	4	Ibis	veg	1300	2	40000
1	2	30-35	5	LemonTree	Non-Veg	2000	3	59000
2	3	25-30	6	RedFox	Veg	1322	2	30000
3	4	20-25	-1	LemonTree	Veg	1234	2	120000
4	5	35+	3	Ibis	Vegetarian	989	2	45000
5	6	35+	3	Ibys	Non-Veg	1909	2	122220
6	7	35+	4	RedFox	Vegetarian	1000	-1	21122
7	8	20-25	7	LemonTree	Veg	2999	-10	345673
8	9	25-30	2	Ibis	Non-Veg	3456	3	-99999
9	10	30-35	5	RedFox	non-Veg	-6755	4	87777

```
df.duplicated()
```

```
0    False
1    False
2    False
3    False
4    False
5    False
6    False
7    False
8    False
9     True
10   False
dtype: bool
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11 entries, 0 to 10
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   CustomerID         11 non-null     int64
 1   Age_Group          11 non-null     objec
 2   Rating(1-5)        11 non-null     int64
 3   Hotel              11 non-null     objec
 4   FoodPreference     11 non-null     object
 5   Bill               11 non-null     int64
 6   NoOfPax            11 non-null     int64
 7   EstimatedSalary    11 non-null     int64
 8   Age_Group.1        11 non-null     object
dtypes: int64(5), object(4)
```

```
df.drop_duplicates(inplace=True)
df
```

	CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill	NoOfPax	Estimated Salary	Age_Group.1
0	1	20-25	4	Ibis	veg	1300	2	40000	20-25
1	2	30-35	5	LemonTree	Non-Veg	2000	3	59000	30-35
2	3	25-30	6	RedFox	Veg	1322	2	30000	25-30
3	4	20-25	-1	LemonTree	Veg	1234	2	120000	20-25
4	5	35+	3	Ibis	Vegetarian	989	2	45000	35+
5	6	35+	3	Ibys	Non-Veg	1909	2	122220	35+
6	7	35+	4	RedFox	Vegetarian	1000	-1	21122	35+
7	8	20-25	7	LemonTree	Veg	2999	-10	345673	20-25
8	9	25-30	2	Ibis	Non-Veg	3456	3	-99999	25-30
10	10	30-35	5	RedFox	non-Veg	-6755	4	87777	30-35

```
len(df)
index =np.array(list(range(0, len(df))))
df.set_index(index, inplace = True)
index
df
```

CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill	NoOfPax	EstimatedSalary	Age_Group
1	20-25	4	Ibis	veg	1300	2	40000	20-25
2	30-35	5	LemonTree	Non-Veg	2000	3	59000	30-35
3	25-30	6	RedFox	Veg	1322	2	30000	25-30
4	20-25	-1	LemonTree	Veg	1234	2	120000	20-25
5	35+	3	Ibis	Vegetarian	989	2	45000	35+
6	35+	3	Ibys	Non-Veg	1909	2	122220	35+
7	35+	4	RedFox	Vegetarian	1000	-1	21122	35+
8	20-25	7	LemonTree	Veg	2999	-10	345673	20-25
9	25-30	2	Ibis	Non-Veg	3456	3	-99999	25-30
9	25-30	2	Ibis	Non-Veg	3456	3	-99999	25-30
10	30-35	5	RedFox	non-Veg	-6755	4	87777	30-35

```
df.drop(['Age_Group.1'],axis=1,inplace = True)
df
```

	CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill	NoOfPax	EstimatedSalary	Age_Group.1
0	1	20-25	4	Ibis	veg	1300	2	40000	20-25
1	2	30-35	5	LemonTree	Non-Veg	2000	3	59000	30-35
2	3	25-30	6	RedFox	Veg	1322	2	30000	25-30
3	4	20-25	-1	LemonTree	Veg	1234	2	120000	20-25
4	5	35+	3	Ibis	Vegetarian	989	2	45000	35+
5	6	35+	3	Ibys	Non-Veg	1909	2	122220	35+
6	7	35+	4	RedFox	Vegetarian	1000	-1	21122	35+
7	8	20-25	7	LemonTree	Veg	2999	-10	345673	20-25
8	9	25-30	2	Ibis	Non-Veg	3456	3	-99999	25-30
9	10	30-35	5	RedFox	non-Veg	-6755	4	87777	30-35

```
df.CustomerID.loc[df.CustomerID<0]=np.nan
df.Bill.loc[df.Bill<0]=np.nan
df.EstimatedSalary.loc[df.EstimatedSalary<0]=np.nan
df
```

	CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill	NoOfPax	EstimatedSalary
0	1.0	20-25	4.0	Ibis	veg	1300.0	2.0	40000.0
1	2.0	30-35	5.0	LemonTree	Non-Veg	2000.0	3.0	59000.0
2	3.0	25-30	NaN	RedFox	Veg	1322.0	2.0	30000.0
3	4.0	20-25	NaN	LemonTree	Veg	1234.0	2.0	120000.0
4	5.0	35+	3.0	Ibis	Vegetarian	989.0	2.0	45000.0
5	6.0	35+	3.0	Ibys	Non-Veg	1909.0	2.0	122220.0
6	7.0	35+	4.0	RedFox	Vegetarian	1000.0	NaN	21122.0
7	8.0	20-25	NaN	LemonTree	Veg	2999.0	NaN	345673.0
8	9.0	25-30	2.0	Ibis	Non-Veg	3456.0	3.0	NaN
9	10.0	30-35	5.0	RedFox	non-Veg	NaN	4.0	87777.0

```
df['NoOfPax'].loc[(df['NoOfPax']<1)|(df['NoOfPax']>20)]=np.nan
df
```

	CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill	NoOfPax	EstimatedSalary
0	1.0	20-25	4.0	Ibis	veg	1300.0	2	40000.0
1	2.0	30-35	5.0	LemonTree	Non-Veg	2000.0	3	59000.0
2	3.0	25-30	NaN	RedFox	Veg	1322.0	2	30000.0
3	4.0	20-25	NaN	LemonTree	Veg	1234.0	2	120000.0
4	5.0	35+	3.0	Ibis	Vegetarian	989.0	2	45000.0
5	6.0	35+	3.0	Ibys	Non-Veg	1909.0	2	122220.0
6	7.0	35+	4.0	RedFox	Vegetarian	1000.0	-1	21122.0
7	8.0	20-25	NaN	LemonTree	Veg	2999.0	-10	345673.0
8	9.0	25-30	2.0	Ibis	Non-Veg	3456.0	3	NaN
9	10.0	30-35	5.0	RedFox	non-Veg	NaN	4	87777.0

```
df.Age_Group.unique()
df.Hotel.unique()
df.Hotel.replace(['Ibys'],'Ibis',inplace=True)
df.FoodPreference.unique
```

```
<bound method Series.unique of 0
1        Non-Veg
2            Veg
3            Veg
4     Vegetarian
5        Non-Veg
6     Vegetarian
7            Veg
8        Non-Veg
9        non-Veg
Name: FoodPreference, dtype: object>
```

```
df.FoodPreferrence.replace(['Vegetarian','Veg'],'Veg',inplace=True)
df.FoodPreferrence.replace(['non_Veg'],'Non_Veg',inplace=True)
df.EstimatedSalary.fillna(round(df.NoOfPax.medium()),inplace=True)
df.Bill.fillna(round(df.Bill.mean()),inplace=True)
df
```

Ex:5 Detect Outliers in a given data set

code:

```
#Rayvan Sanjai K.S
#240701425

import numpy as np
array =np.random.randint(1,100,16)
array

array.mean()
np.percentile(array,25)
np.percentile(array,50)
np.percentile(array,75)
np.percentile(array,100)
```

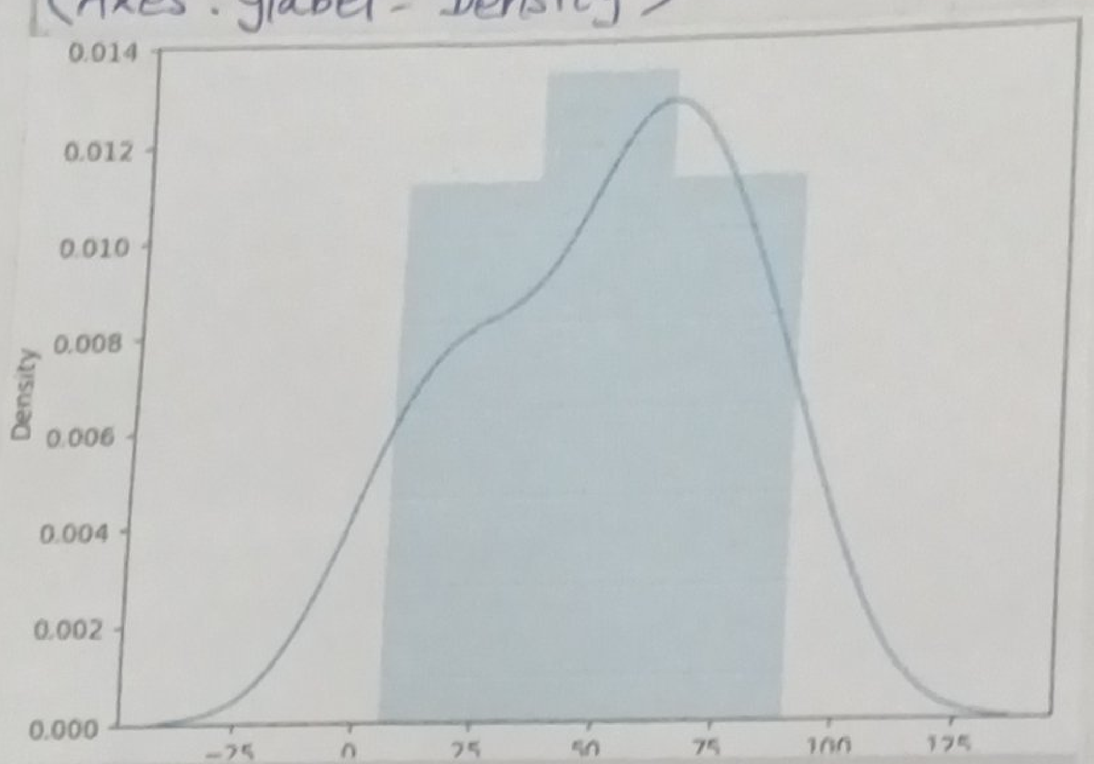
```
#outliers detection
def outDetection(array):
  sorted(array)
  Q1,Q3=np.percentile(array,[25,75])
  IQR=Q3-Q1
  lr=Q1-(1.5*IQR)
  ur=Q3+(1.5*IQR)
  return lr,ur
```

```
lr,ur=outDetection(array)
lr,ur

import seaborn as sns
%matplotlib inline
sns.displot(array)
```

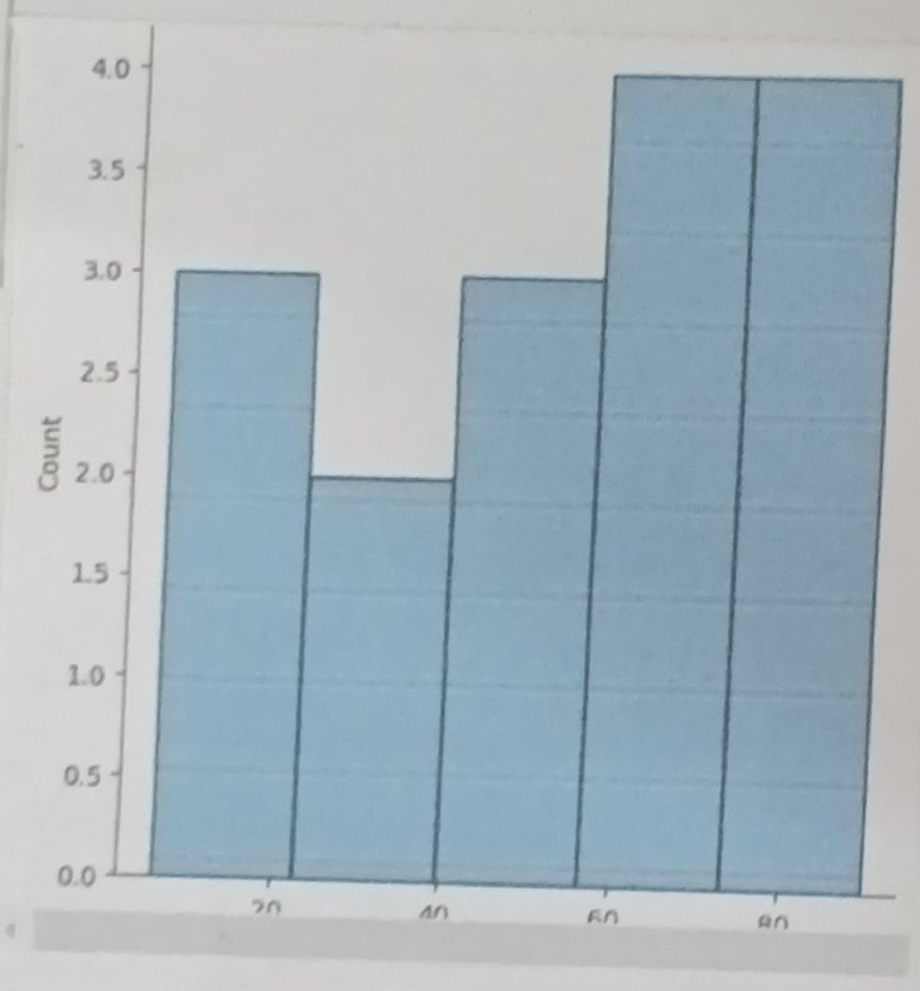

sns.distplot(array)

```
Sns.distplot(array)
```




```
new_array=array[(array>lr)&(array<ur)]
new_array
```

```
Sns.displot(new_array)
```




```
lr1,ur1=outDetection(new_array)
lr1,ur1

final_array=new_array[(new_array>lr1)&(new_array<ur1)]
final_array

Sns.distplot(final_array)

Sns.distplot(final_array)
<Axes:ylabel='Density'>
```

Ex:6 To understand feauture Scaling

Code:

#

#

```
import numpy as np
import pandas as pd
df=pd.read_csv(Pre-process_datasample.csv)
df
```

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	NaN	Yes
5	France	35.0	58000.0	Yes
6	Spain	NaN	52000.0	No
7	France	48.0	79000.0	Yes
8	NaN	50.0	83000.0	No
9	France	37.0	67000.0	Yes

```
df.head()
```

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	NaN	Yes

```
df.country.fillna(df.country.mode()[0],inplace=True)
features=df.iloc[:,:-1].values
label=df.iloc[:,-1].values
from sklearn.impute import SimpleImputer
age=SimpleImputer(strategy="mean",missing_values=np.nan)
Salary=SimpleImputer(strategy="mean",missing_values=np.nan)
age.fit(features[:,[1]])
```

SimpleImputer
SimpleImputer()

```
Salary.fit(features[:,[2]])
```

SimpleImputer
SimpleImputer()

SimpleImputer()

```
SimpleImputer()
```

```
features[:,[1]] = age.transform(features[:,[1]])
features[:,[2]] = Salary.transform(features[:,[2]])
features
```

```
array([['France', 44.0, 72000.0],
       ['Spain', 27.0, 48000.0],
       ['Germany', 30.0, 54000.0],
       ['Spain', 38.0, 61000.0],
       ['Germany', 40.0, 63777.77777777778],
       ['France', 35.0, 58000.0],
       ['Spain', 38.77777777777778, 52000.0],
       ['France', 48.0, 79000.0],
       ['France', 50.0, 83000.0],
       ['France', 37.0, 67000.0]], dtype=object)
```

```
from Sklearn.preprocessing import OneHotEncoder
oh = OneHotEncoder(sparse_output=false)
Country = oh.fit_transform(features[:,[0]])
Country
```

```
array([[1., 0., 0.],
       [0., 0., 1.],
       [0., 1., 0.],
       [0., 0., 1.],
       [0., 1., 0.],
       [1., 0., 0.],
       [0., 0., 1.],
       [1., 0., 0.],
       [1., 0., 0.],
       [1., 0., 0.]])
```

```
final_set = np.concatenate((Country, features[:,[1,2]]), axis=1)
final_set
```

```
array([[1.0, 0.0, 0.0, 44.0, 72000.0],
       [0.0, 0.0, 1.0, 27.0, 48000.0],
       [0.0, 1.0, 0.0, 30.0, 54000.0],
       [0.0, 0.0, 1.0, 38.0, 61000.0],
       [0.0, 1.0, 0.0, 40.0, 63777.77777777778],
       [1.0, 0.0, 0.0, 35.0, 58000.0],
       [0.0, 0.0, 1.0, 38.77777777777778, 52000.0],
       [1.0, 0.0, 0.0, 48.0, 79000.0],
       [1.0, 0.0, 0.0, 50.0, 83000.0],
       [1.0, 0.0, 0.0, 37.0, 67000.0]], dtype=object)
```

```
from Sklearn.preprocessing import StandardScaler
SC = StandardScaler()
SC.fit(final_set)
feat_Standard_Scaler = SC.transform(final_set)
feat_Standard_Scaler
```

```
array([[ 1.00000000e+00, -5.00000000e-01, -6.54653671e-01,
         7.58874362e-01,  7.49473254e-01],
       [-1.00000000e+00, -5.00000000e-01,  1.52752523e+00,
        -1.71150388e+00, -1.43817841e+00],
       [-1.00000000e+00,  2.00000000e+00, -6.54653671e-01,
        -1.27555478e+00, -8.91265492e-01],
       [-1.00000000e+00, -5.00000000e-01,  1.52752523e+00,
        -1.13023841e-01, -2.53200424e-01],
       [-1.00000000e+00,  2.00000000e+00, -6.54653671e-01,
         1.77608893e-01,  6.63219199e-16],
       [ 1.00000000e+00, -5.00000000e-01, -6.54653671e-01,
        -5.48972942e-01, -5.26656882e-01],
       [-1.00000000e+00, -5.00000000e-01,  1.52752523e+00,
         0.00000000e+00, -1.07356980e+00],
       [ 1.00000000e+00, -5.00000000e-01, -6.54653671e-01,
         1.34013983e+00,  1.38753832e+00],
       [ 1.00000000e+00, -5.00000000e-01, -6.54653671e-01,
         1.63077256e+00,  1.75214693e+00],
       [ 1.00000000e+00, -5.00000000e-01, -6.54653671e-01,
```

```
from sklearn.preprocessing import MinMaxScaler
mms = MinMaxScaler(feature_range=(0,1))
mms.fit(final_set)
feat_minmax_scaler=mms.transform(final_set)
feat_minmax_scaler
```

```
array([[1.        , 0.        , 0.        , 0.73913043, 0.68571429],
       [0.        , 0.        , 1.        , 0.        , 0.        ],
       [0.        , 1.        , 0.        , 0.13043478, 0.17142857],
       [0.        , 0.        , 1.        , 0.47826087, 0.37142857],
       [0.        , 1.        , 0.        , 0.56521739, 0.45079365],
       [1.        , 0.        , 0.        , 0.34782609, 0.28571429],
       [0.        , 0.        , 1.        , 0.51207729, 0.11428571],
       [1.        , 0.        , 0.        , 0.91304348, 0.88571429],
       [1.        , 0.        , 0.        , 1.        , 1.        ],
       [1.        , 0.        , 0.        , 0.43478261, 0.54285714]])
```

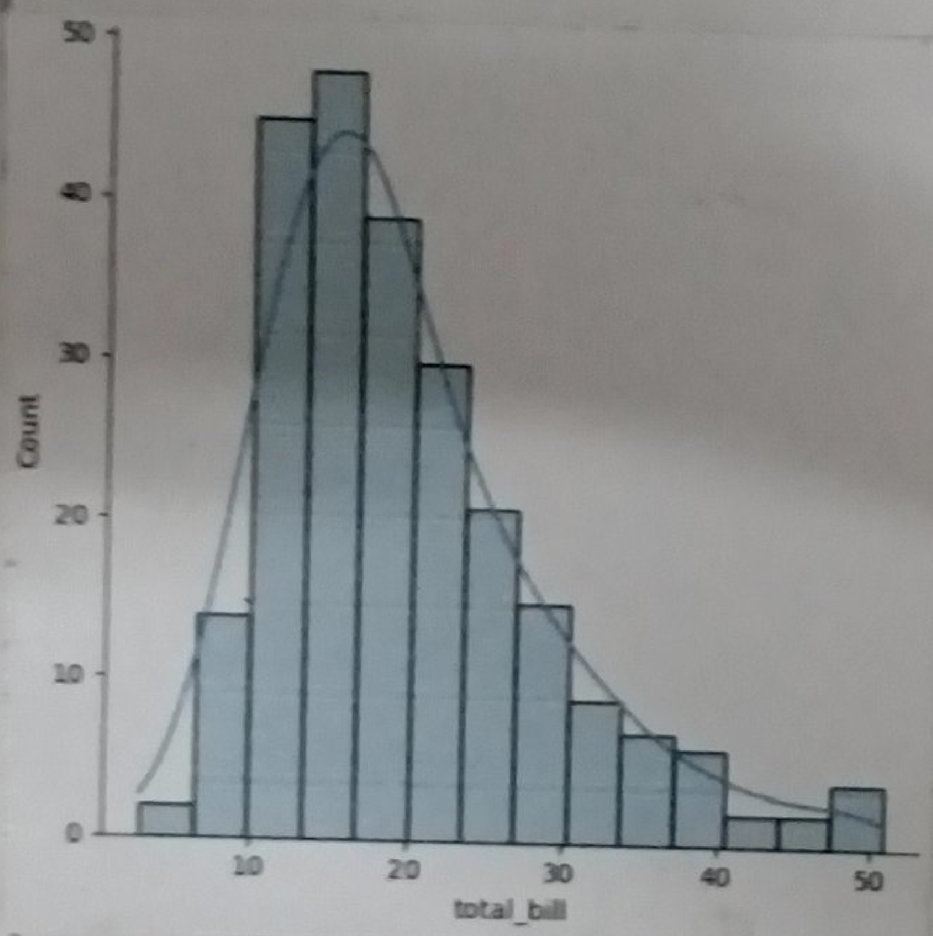
Exp: 7

Code:

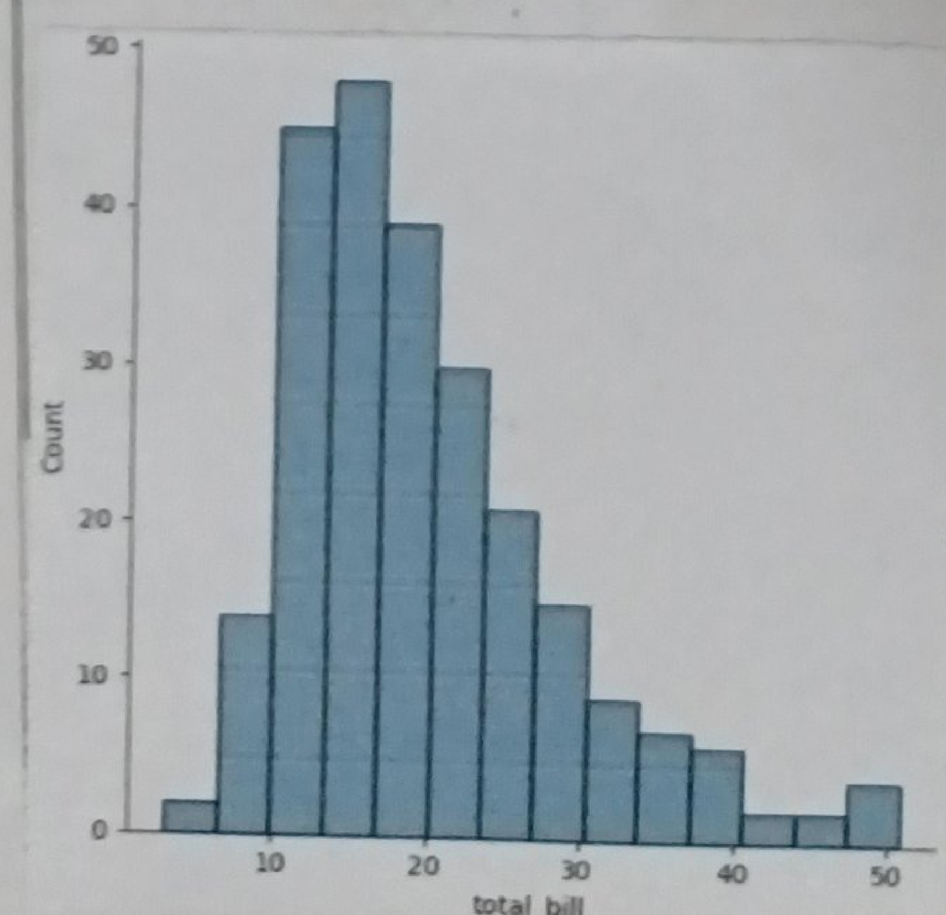
```
#
#
import seaborn as sns
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
tips = sns.load_dataset('tips')
tips.head()
```

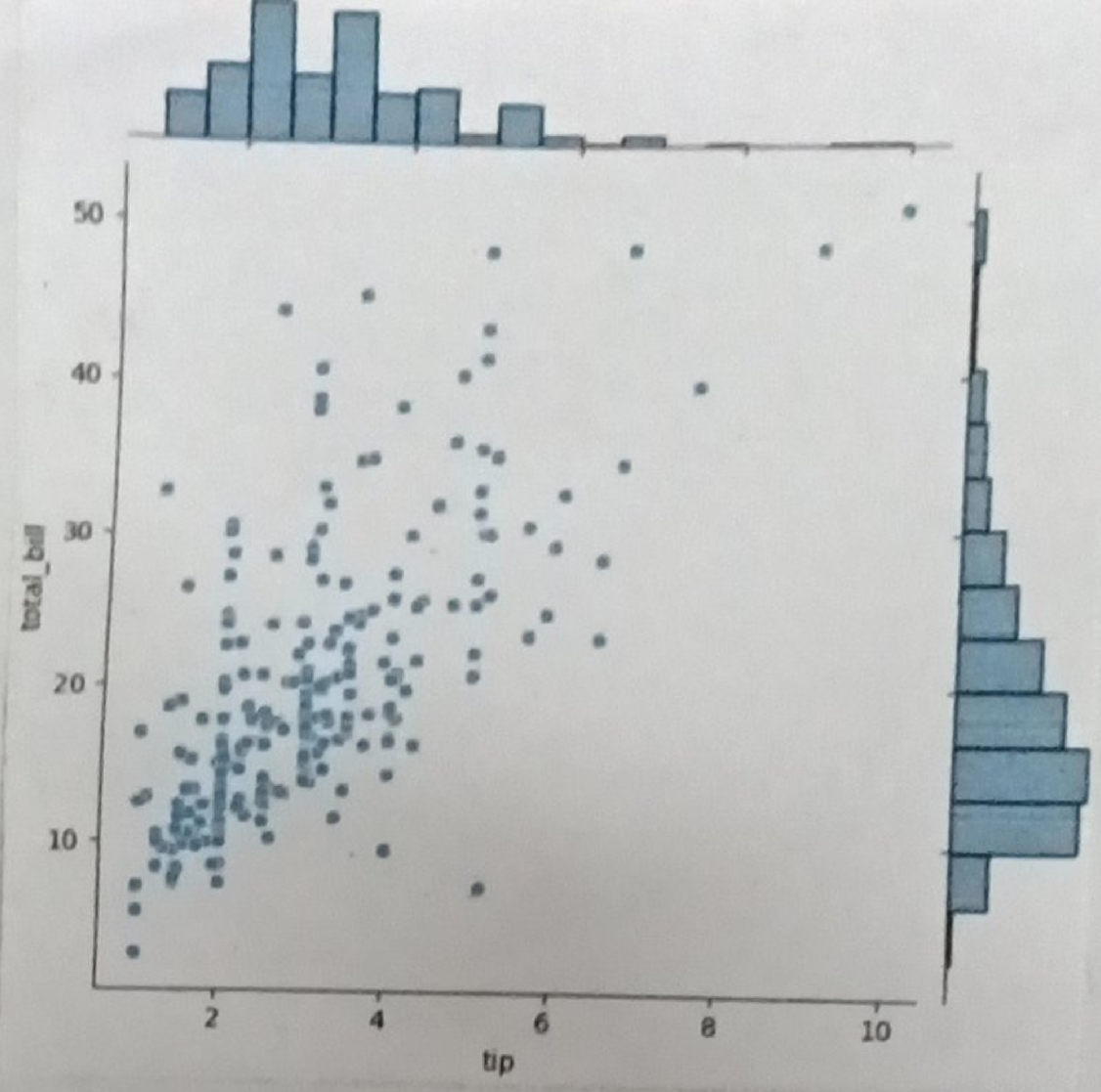
	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

```
sns.displot(tips.total_bill, kde=True)
```

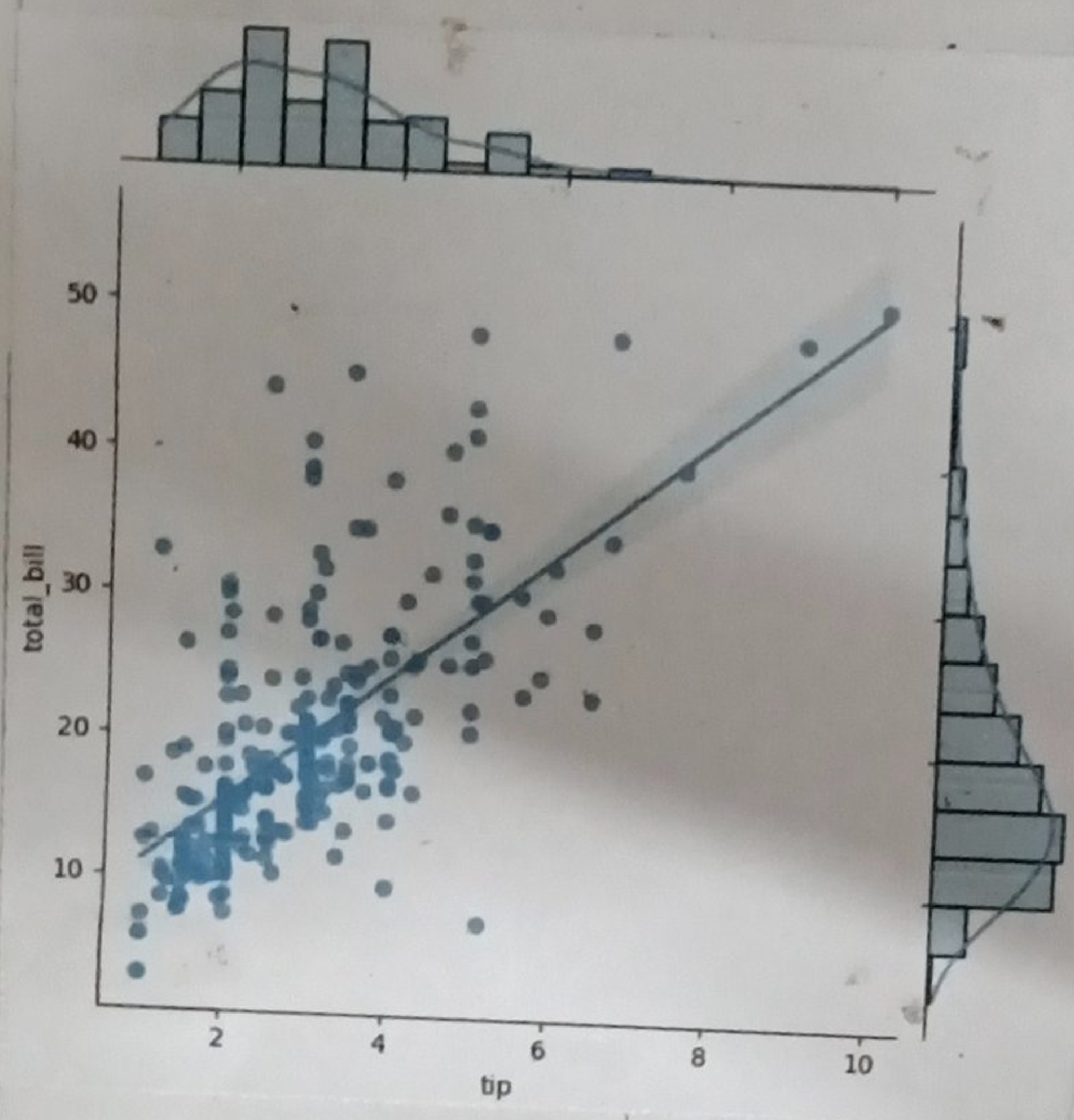



```
sns.displot(tips.total_bill, kde=False)
```

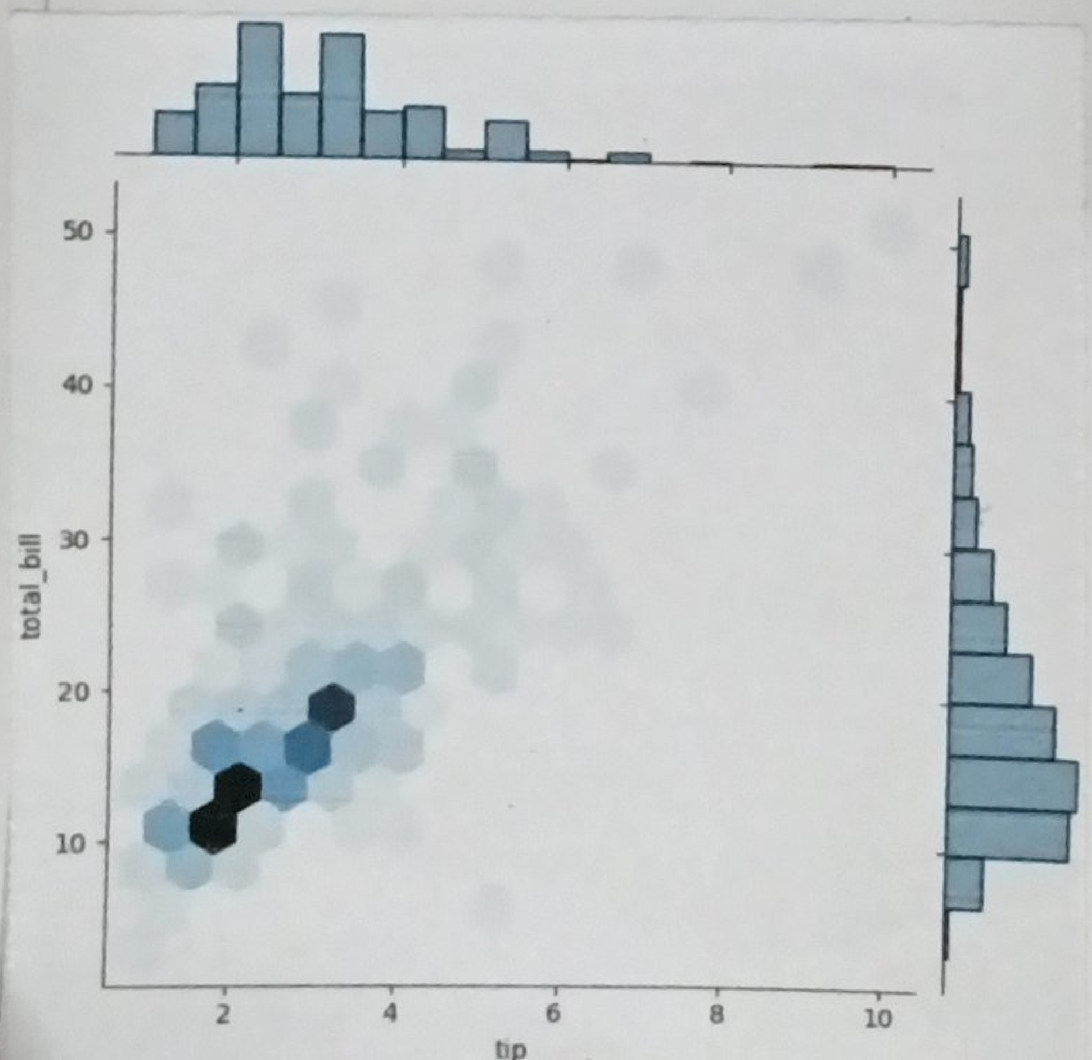


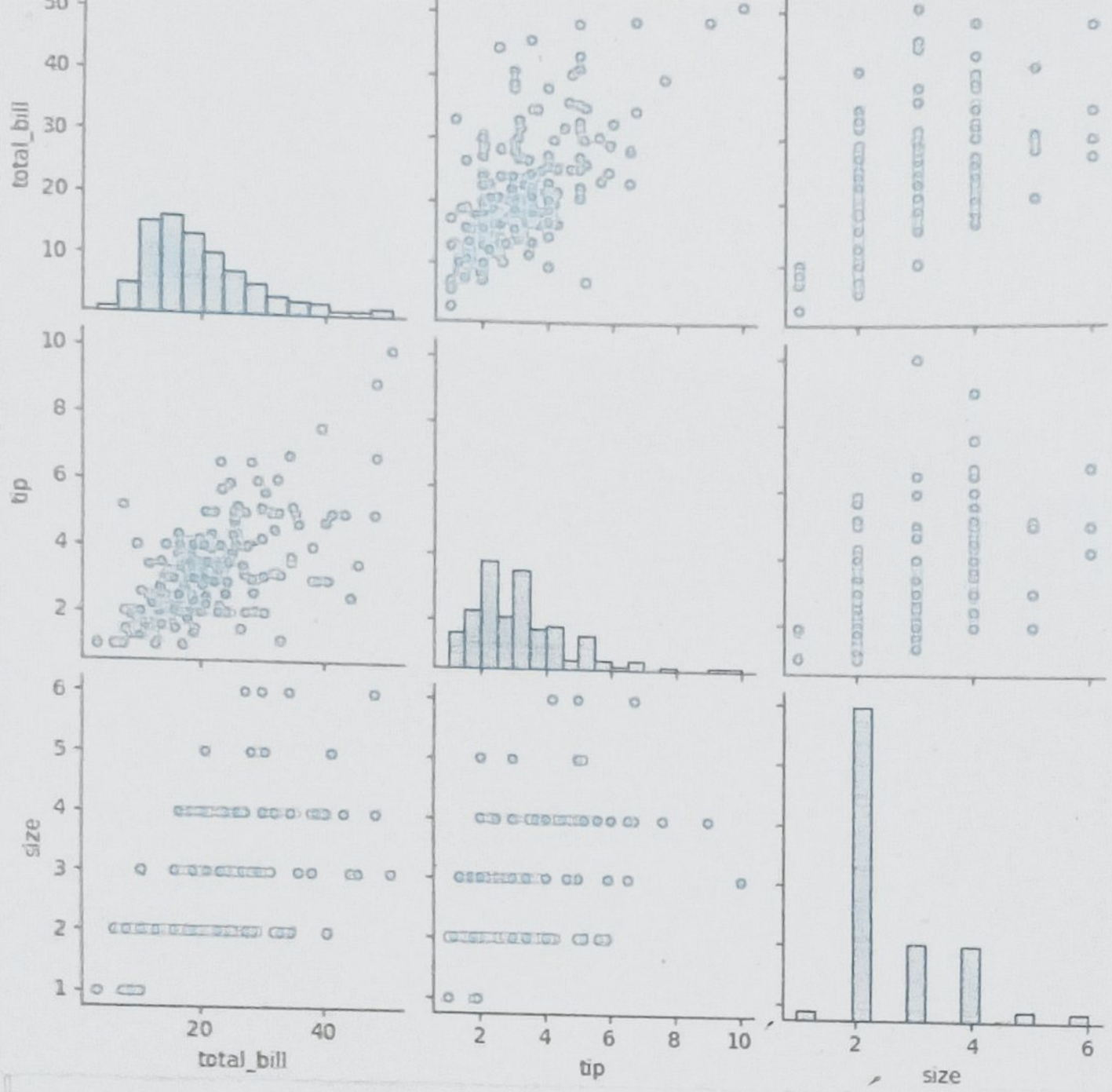


Sns.jointplot(x=tips.tip, y=tips.total_bill, kind="reg")



Sns.jointplot(x=tips.tip, y=tips.total_bill, kind="hex")



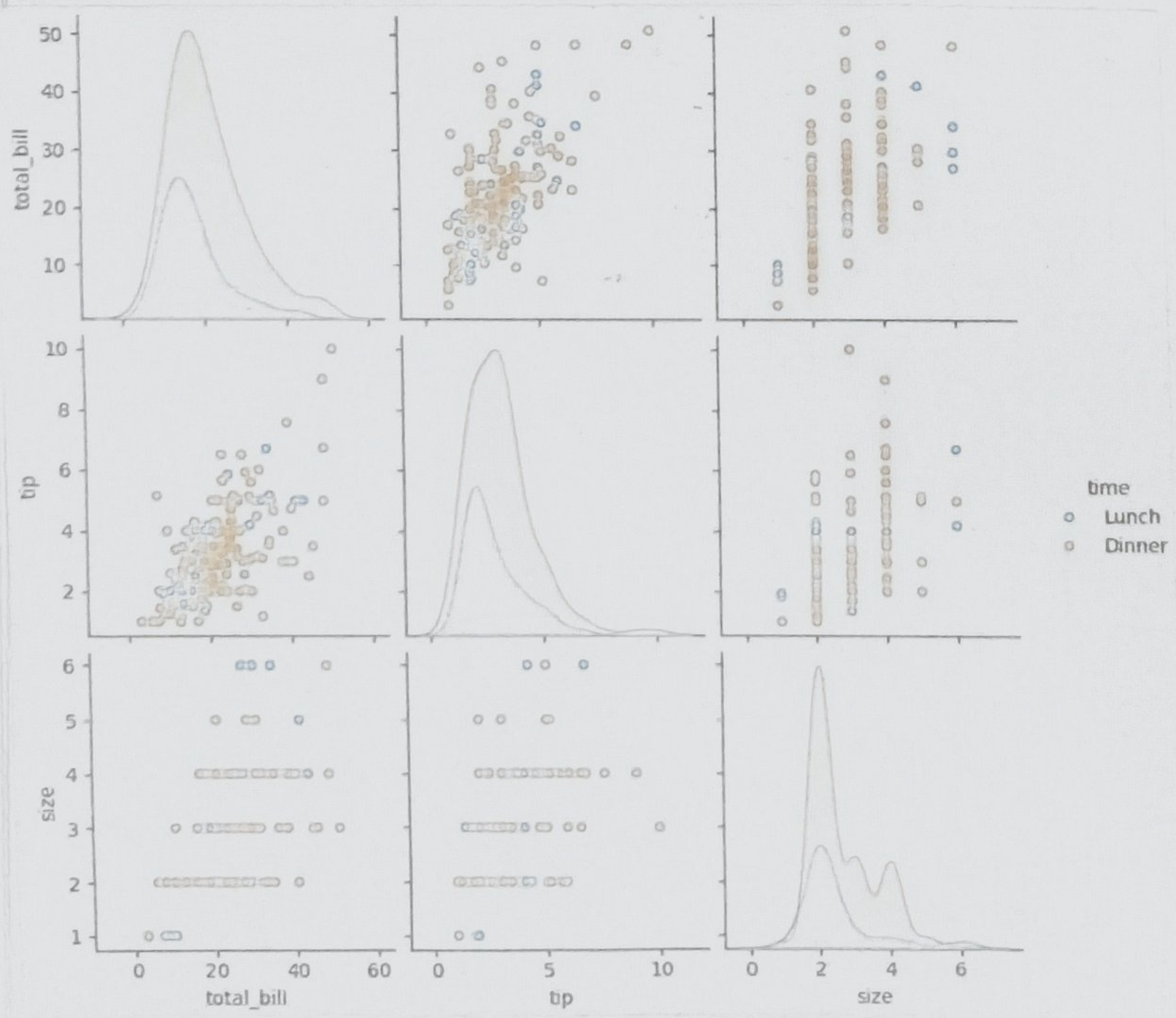



```
tips.time.value_counts()
```

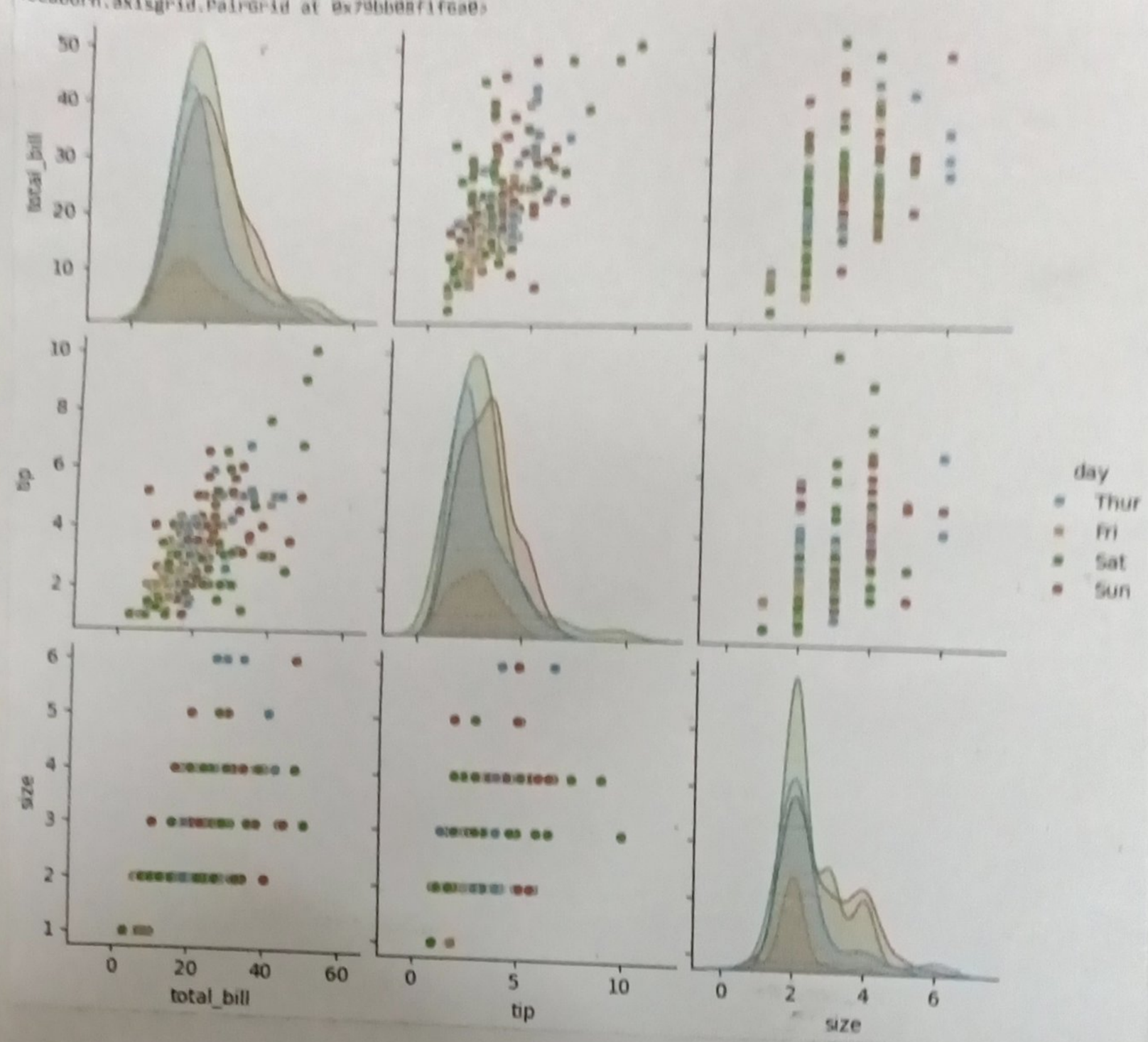
time	count
Dinner	176
Lunch	68

dtype: int64

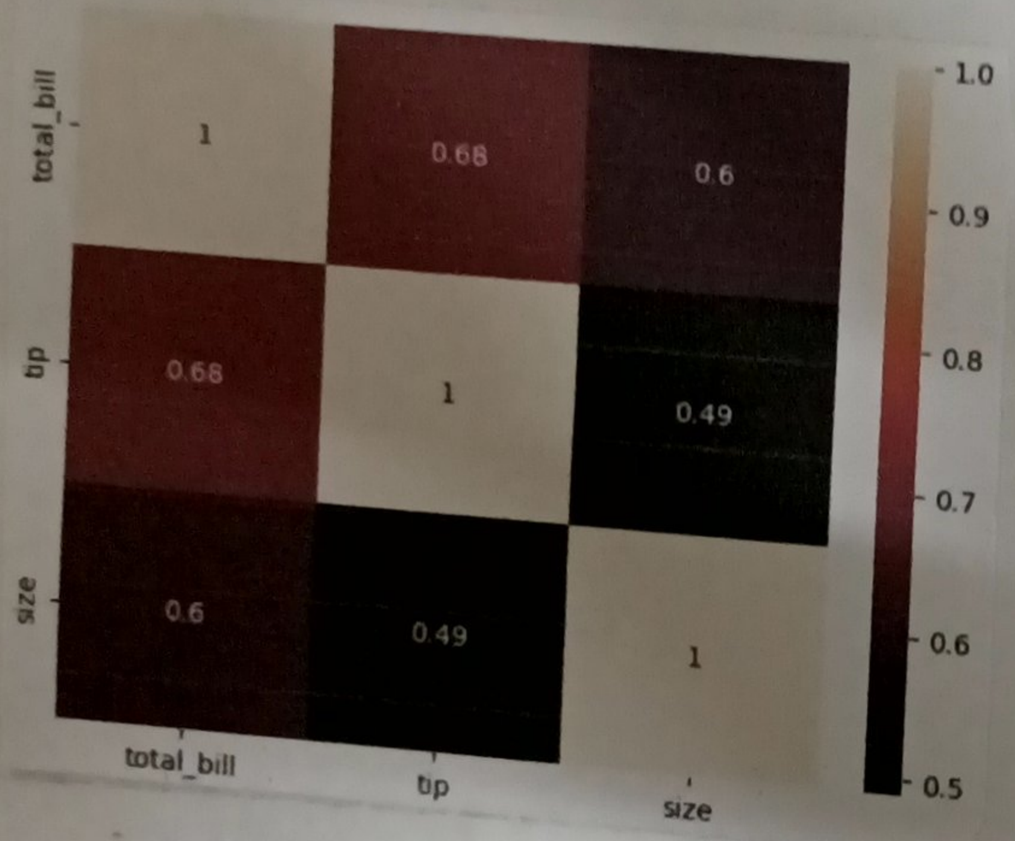
```
sns.pairplot(tips, hue='time')
```



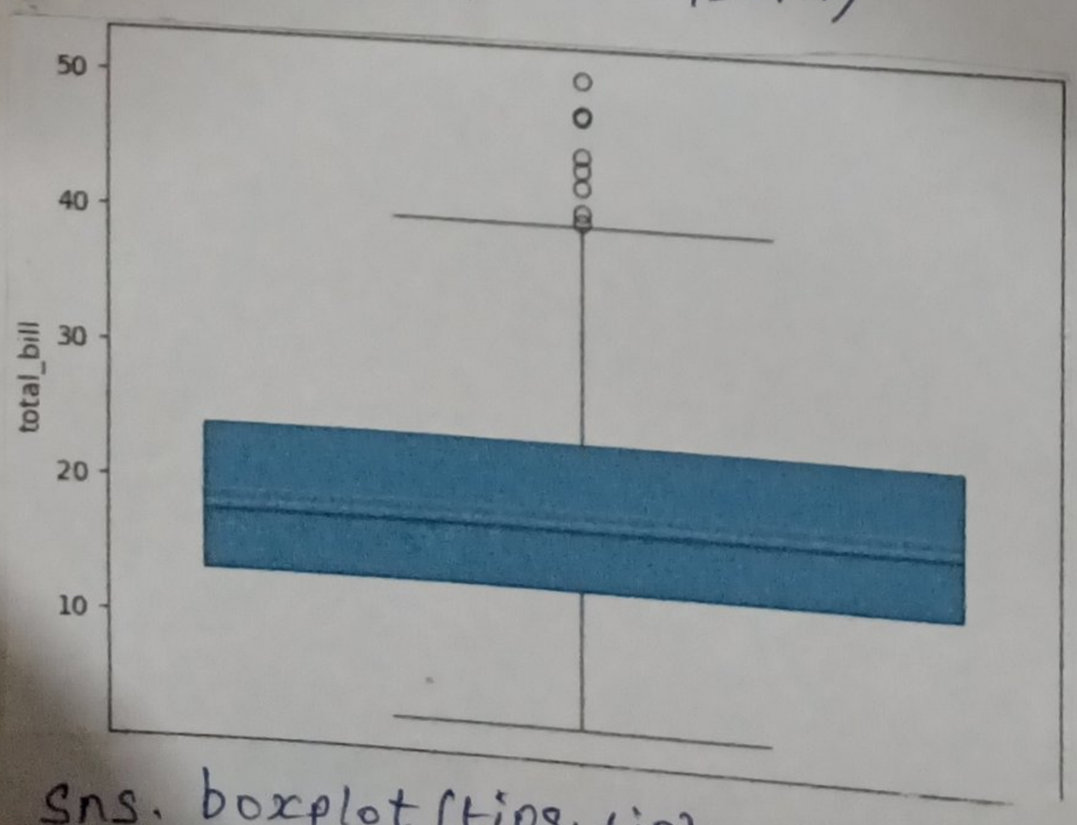

```
sns.pairplot(tips, hue='day')
```



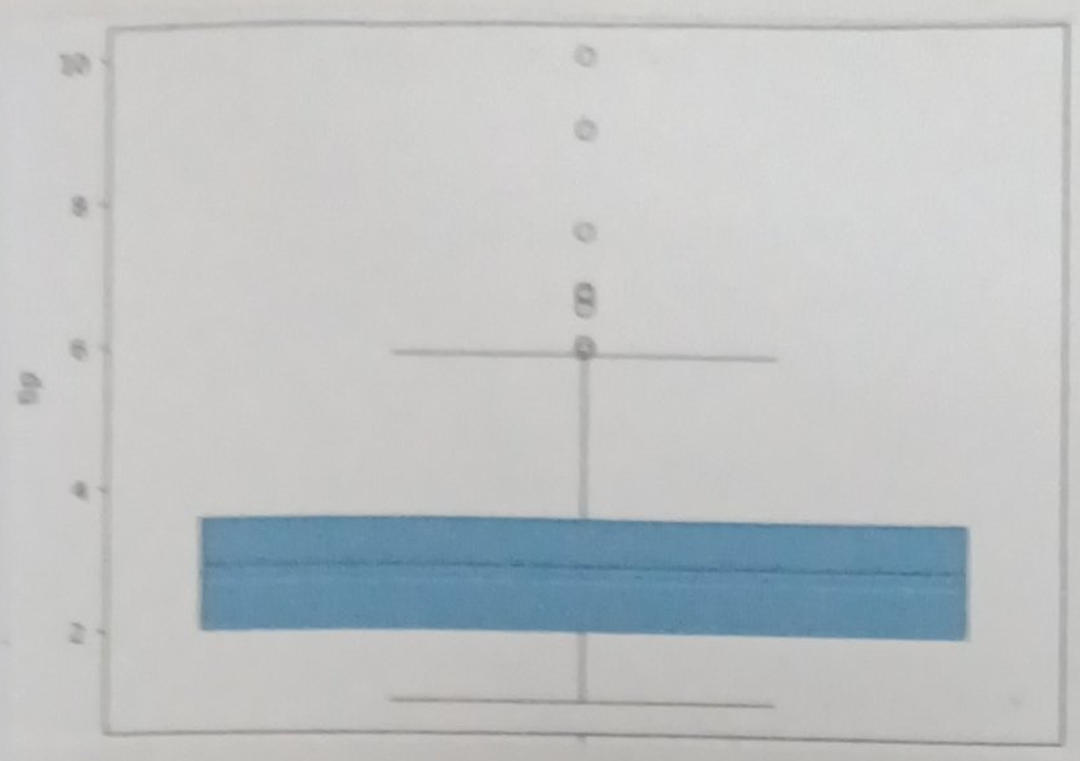
sns.heatmap(tips.corr(numeric_only=True), annot=True)



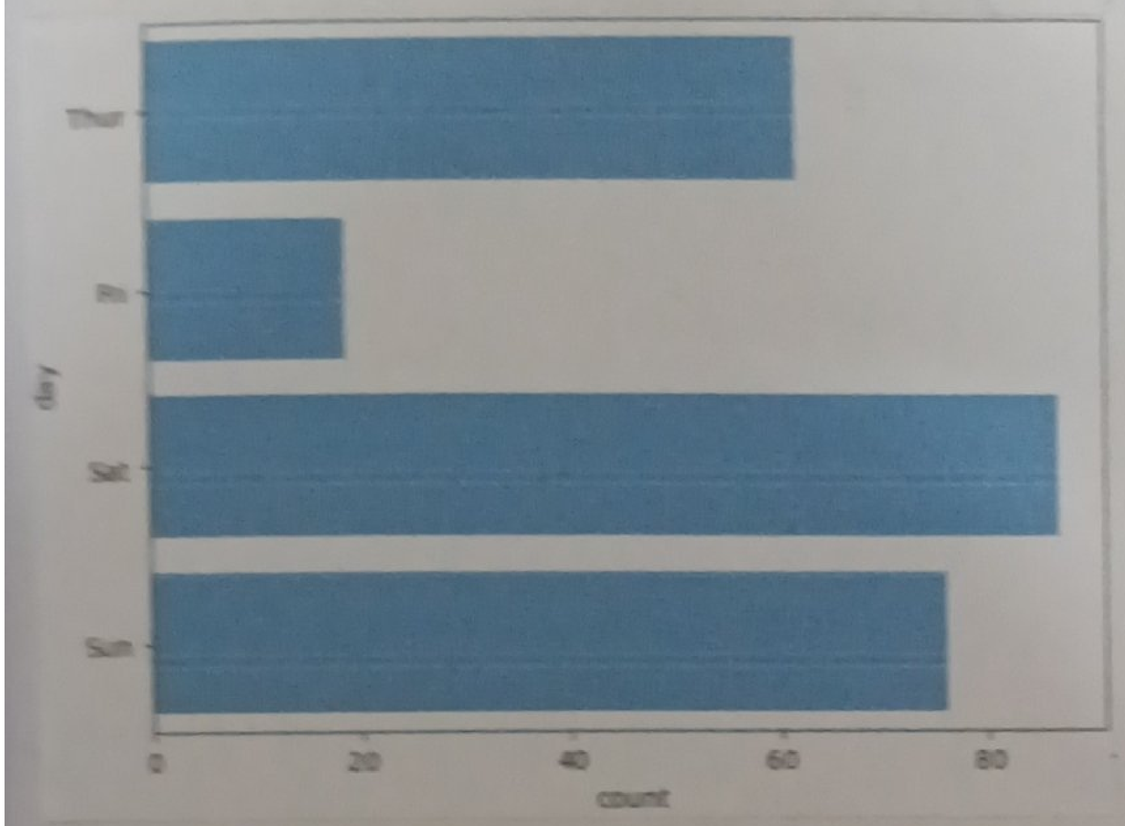
sns.boxplot(tips.total_bill)



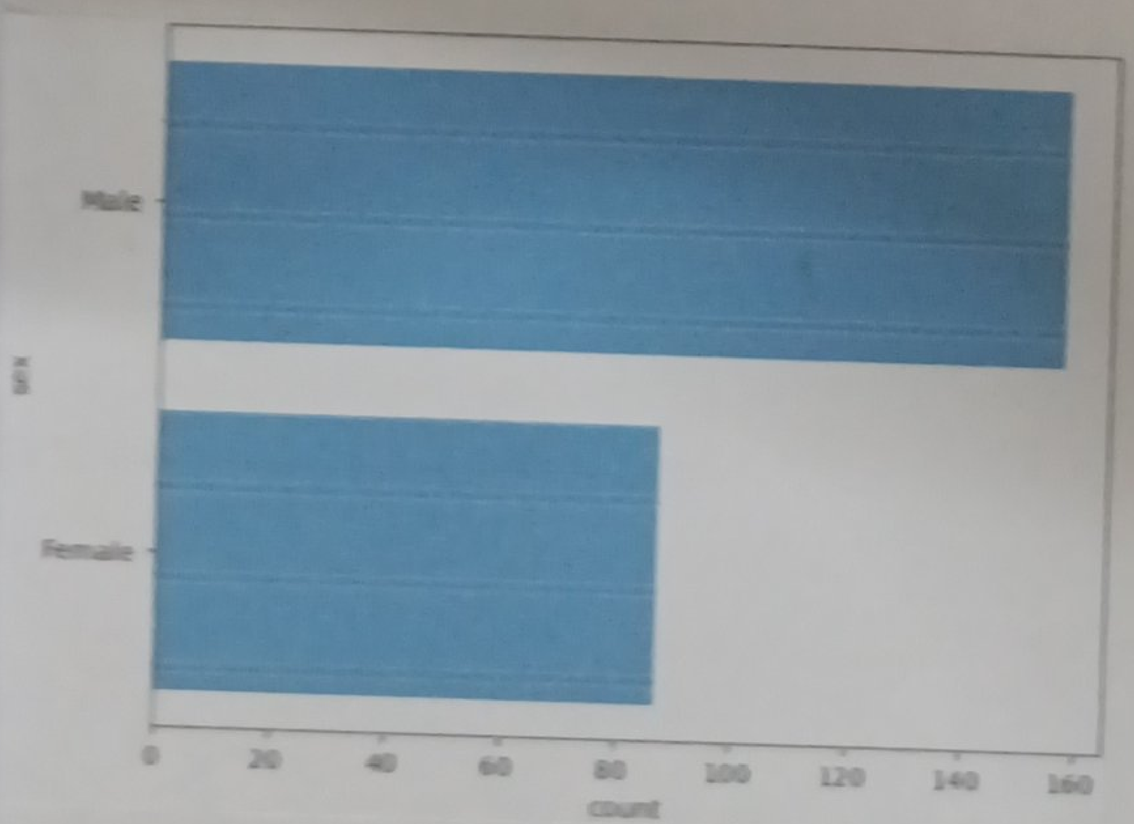
sns.boxplot(tips.tip)



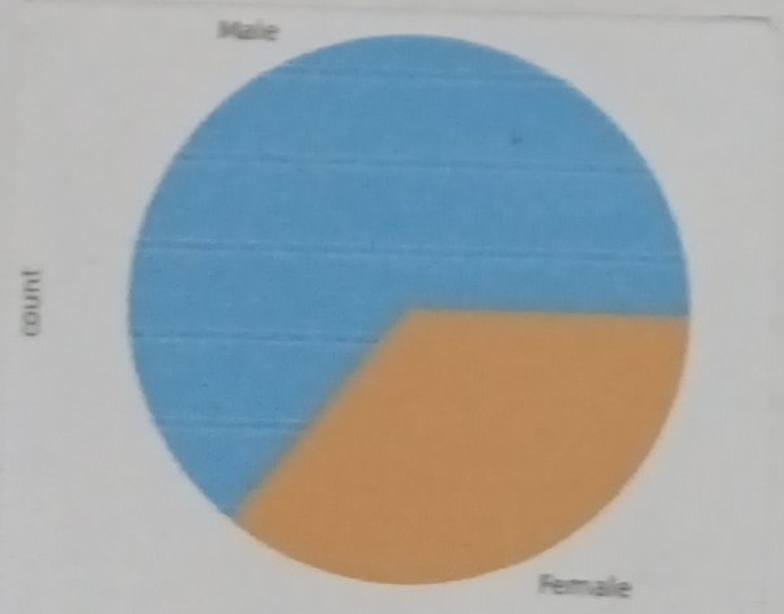
sns.countplot(tips.day)

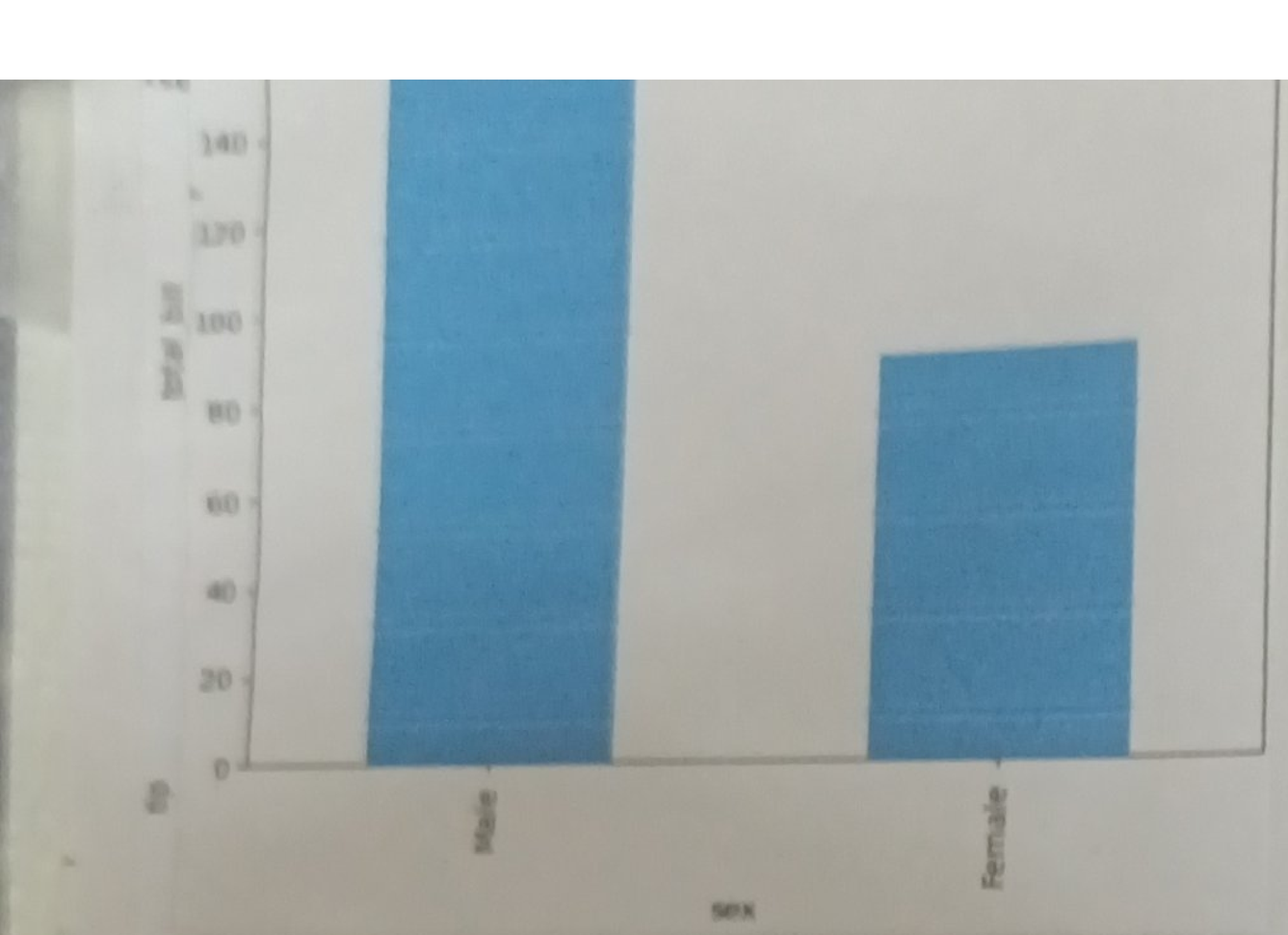


sns.countplot(tips.sex)

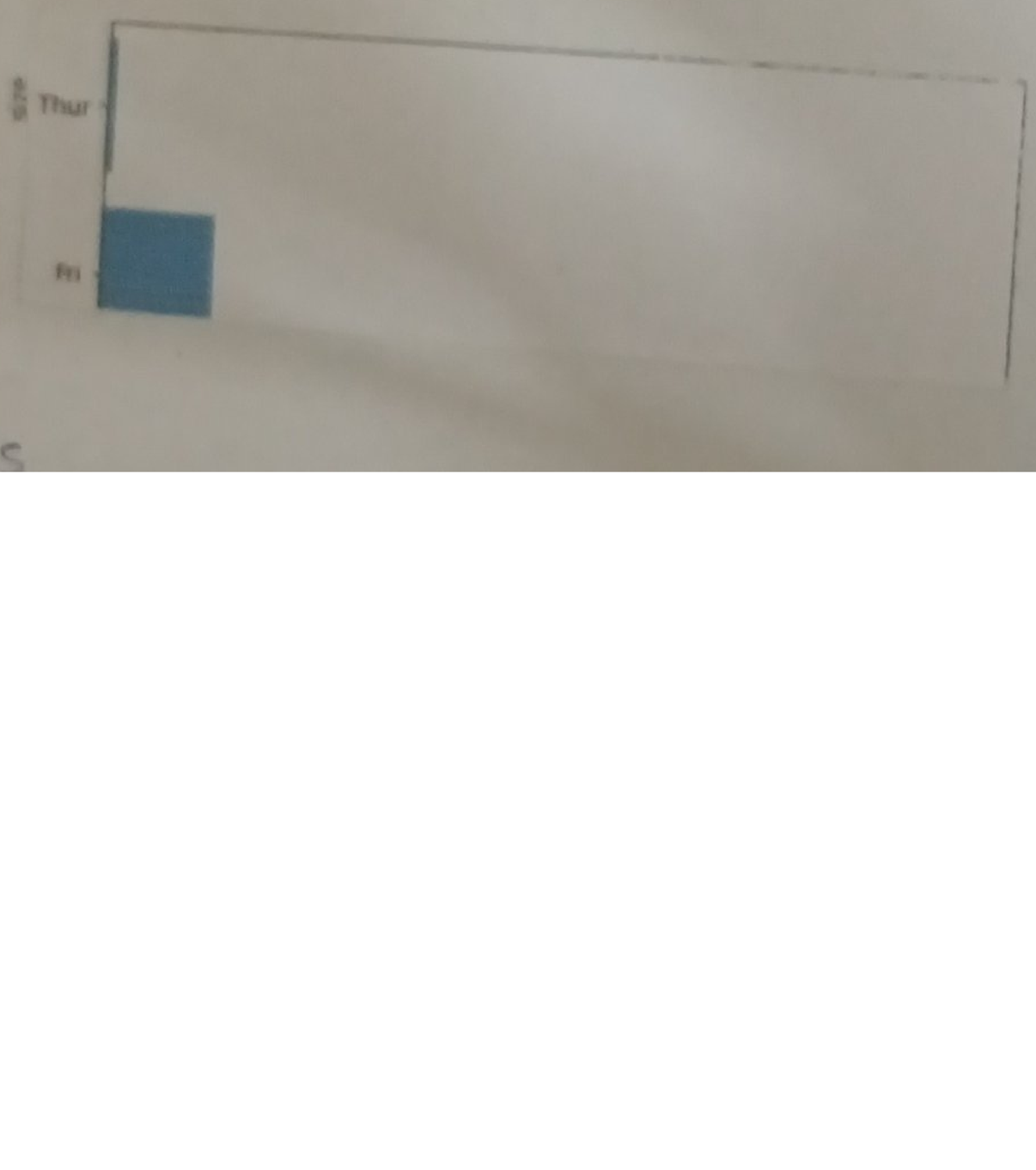


tips.sex.value_counts().plot(kind='pie')






```
sns.countplot(tips[tips.time=='Dinner']['day'])
```



Exp: 8

code:

```
#
#
import numpy as np
import pandas as pd
df=pd.read_csv('Salary_data.csv')
df
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   YearsExperience  30 non-null     float64
 1   Salary           30 non-null     int64  
dtypes: float64(1), int64(1)
memory usage: 612.0 bytes
```

```
df.dropna(inplace=True)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   YearsExperience  30 non-null     float64
 1   Salary           30 non-null     int64  
dtypes: float64(1), int64(1)
memory usage: 612.0 bytes
```

```
df.describe()
```

	YearsExperience	Salary
count	30.000000	30.000000
mean	5.313333	76003.000000
std	2.837888	27414.429785
min	1.100000	37731.000000
25%	3.200000	56720.750000
50%	4.700000	65237.000000
75%	7.700000	100544.750000
max	10.500000	122391.000000

```
features = df.iloc[:, 0].values
label = df.iloc[:, 1].values

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(features, label, test_size
= 0.2, random_state=0)


from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train, y_train)

LinearRegression()
LinearRegression()

model.score(X_train, y_train)

0.9411949620562126

model.score(X_test, y_test)

0.9749154407708353

model.coef_

array([9281.3884376])

model.intercept_

array(26816.19224403)

import pickle
pickle.dump(model, open('SalaryPred.model', 'wb'))
model = pickle.load(open('SalaryPred.model', 'rb'))

yr_of_exp = float(input("Enter Years of Experience: "))
yr_of_exp_NP = np.array([[yr_of_exp]])
Salary = model.predict(yr_of_exp_NP)

print("Estimated Salary for {} years of experience is {}: ".
format(yr_of_exp, Salary))
```

Exp : 9

code :

```
#
#
import numpy as np
import pandas as pd
df = pd.read_csv('Social_Network_Ads.csv')
df
```

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
...	...	...	...	...	...
395	15691863	Female	46	41000	1
396	15706071	Male	51	23000	1
397	15654296	Female	50	20000	1
398	15755018	Male	36	33000	0
399	15594041	Female	49	36000	1

```
df.head()
```

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

```
features = df.iloc[:, [2,3]].values
label = df.iloc[:,4].values
features
```

```
array([[    19,  19000],
       [    35,  20000],
       [    26,  43000],
       [    27,  57000],
       [    19,  76000],
       [    27,  58000],
       [    27,  84000],
       [    32, 150000],
       [    25,  33000],
       [    35,  65000],
       [    26,  80000],
       [    26,  52000],
       [    20,  86000],
       [    32,  18000],
       [    18,  82000],
       [    29,  80000],
       [    47,  25000],
       [    45,  26000],
       [    46,  28000],
```

```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0,
       1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0,
       1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
       0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1,
       1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1,
       0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0,
       1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1,
       0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1,
       1, 1, 0, 1], dtype=int64)
```

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

for i in range(1,401):
    x_train,x_test,y_train,y_test = train_test_split(features,label,test_
    model = LogisticRegression()
    model.fit(x_train,y_train)
    train_score=model.score(x_train,y_train)
    test_score=model.score(x_test,y_test)
    if test_score>train_score:
```

```
Test 0.6875 Train0.63125 Random State 3
Test 0.7375 Train0.61875 Random State 4
Test 0.6625 Train0.6375 Random State 5
Test 0.65 Train0.640625 Random State 6
Test 0.675 Train0.634375 Random State 7
Test 0.675 Train0.634375 Random State 8
Test 0.65 Train0.640625 Random State 10
Test 0.6625 Train0.6375 Random State 11
Test 0.7125 Train0.625 Random State 13
Test 0.675 Train0.634375 Random State 16
Test 0.7 Train0.628125 Random State 17
Test 0.7 Train0.628125 Random State 21
Test 0.65 Train0.640625 Random State 24
Test 0.6625 Train0.6375 Random State 25
Test 0.75 Train0.615625 Random State 26
Test 0.675 Train0.634375 Random State 27
Test 0.7 Train0.628125 Random State 28
Test 0.6875 Train0.63125 Random State 29
Test 0.6875 Train0.63125 Random State 31
```

```
x_train,x_test,y_train,y_test = train_test_split(feature,label,test_size=0.
finalModel=LogisticRegression()
finalModel.fit(x_train,y_train)


print(finalModel.score(x_train,y_train))
Print(finalModel.score(x_test,y_test))
```

```
from sklearn.metrics import classification_report
print(classification_report(label, finalModel.predict(features)))
```

	precision	recall	f1-score	support
0	0.85	0.93	0.89	257
1	0.84	0.71	0.77	143
accuracy			0.85	400
macro avg	0.85	0.82	0.83	400
weighted avg	0.85	0.85	0.85	400

Exp:10

Code:

```
#
#
import numpy as np
import pandas as pd
df=pd.read_csv('Iris.csv')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal.length  150 non-null    float64
 1   sepal.width   150 non-null    float64
 2   petal.length  150 non-null    float64
 3   petal.width   150 non-null    float64
 4   variety       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

```
df.variety.value_counts()
```

```
Setosa        50
Versicolor    50
Virginica     50
Name: variety, dtype: int64
```

```
df.head()
```

	sepal.length	sepal.width	petal.length	petal.width	variety
0	5.1	3.5	1.4	0.2	Setosa
1	4.9	3.0	1.4	0.2	Setosa
2	4.7	3.2	1.3	0.2	Setosa
3	4.6	3.1	1.5	0.2	Setosa
4	5.0	3.6	1.4	0.2	Setosa

```
features = df.iloc[:,:-1].values
label = df.iloc[:,4].values

from sklearn.model_selection import train_test_split
from sklearn.neighbours import KNeighborsClassifier

xtrain, xtest, ytrain, ytest = train_test_split(features, label, test_size=.
model_KNN = KNeighbourClassifier(n_neighbors=5)
model_KNN.fit(xtrain, ytrain)
```

```
print (model_KNN.score (xtrain_ytrain))
print (model_KNN. score (xtest,ytest))
```

```
0.9583333333333334
1.0
```

```
from sklearn.metrices import confusion_matrix
confusion_matrix (label, model_KNN.predict (features))
```

```
array([[50,  0,  0],
       [ 0, 47,  3],
       [ 0,  2, 48]], dtype=int64)
```

```
from sklearn.metrices import classification_report
print (classification_report (label, model_KNN- predict (features)))
```

```
              precision    recall  f1-score   support

      Setosa       1.00      1.00      1.00        50
  Versicolor       0.96      0.94      0.95        50
   Virginica       0.94      0.96      0.95        50

    accuracy                           0.97       150
   macro avg       0.97      0.97      0.97       150
weighted avg       0.97      0.97      0.97       150
```

T-test

Ex: 12

A Sample of 10 Students scored the following marks in an exam:
[72, 68, 75, 70, 74, 69, 71, 73, 70, 72] we want to test whether the average mark = 70 ($\mu_0 = 70$) at 5% significance level using python.

code:

```
import numpy as np
from scipy import stats
#Sample data
marks = np.array([72,68,75,70,74,69,71,73,70,72])
#Hypothesized mean
mu_0 = 70
#one-sample t-test
t_stat, p_value = stat.ttest_1samp(marks, mu_0)
print(f"T-statistic: {t_stat:.3f}")
print(f"P-value: {p_value:.4f}")
alpha = 0.05
if p_value < alpha:
    Print("Reject Null Hypothesis -> Mean is significantly from 70.")
else:
    print("Fail to Reject Null Hypothesis -> No Significant difference.")
```

OUTPUT:

T-Statistic : 1.993
P-value : 0.0774
Fail to Reject Null Hypothesis -> No significant difference.

Exp: 13 Z - test

A manufacturer claims that the average weight of packets is 50g. A random sample of 36 packets has an average weight of 51.2g with a known $\sigma = 3g$. At a 5% significance level, test the claim.

Code:

```
import numpy as np
import
from math import sqrt
from scipy.stats import norm
# Given data
x_bar = 51.2
mu_0 = 50
sigma = 3
n = 36

z_stat = (x_bar - mu_0) / (sigma / sqrt(n))

p_value = 2 * (1 - norm.cdf(abs(z_stat)))
print(f"z-statistic: {z_stat:.3f}")
print(f"p-value: {p_value:.4f}")
alpha = 0.05
if p_value < alpha:
    print("Reject Null Hypothesis -> Mean is significantly from 50g.")
else:
    print("Fail to Reject Null Hypothesis -> No Significant difference.")
```

OUTPUT:

```
Z_statistic : 2.400
P-value : 0.0164
Reject Null Hypothesis -> Mean is significantly different from 50g.
```

Ex:14 ANOVA TEST

Three fertilizers (A,B,C) were tested on crop yield (in kg). Is there a significant difference among fertilizers? (Use $\alpha = 0.05$)

Fertilizer	yields
A	20, 22, 23
B	19, 20, 18
C	25, 27, 26

Code:

```
import numpy as np
from spicy import stats

# Data

A = [20,22, 23]
B = [19,20, 18]
c = [25, 27, 26]

f_stat, p_value = stats. f_oneway (A,B, C)

print (f"F-statistic : {f_stat :. 3f}")
print (f"P_value : {p_value :. 4f}")

alpha = 0.05

if p_value < alpha:

    print ("Reject Null Hypothesis -> Means are significantly different")

else:
    print ("Fail to Reject Null Hypothesis -> No significant difference.")
```

OUTPUT:

F-statistic : 25.923
P-value : 0.0011
Reject Null Hypothesis -> Means are significantly different