

INDEX

Name: Rayyan Sanjai KS

sid: CSE Subject: FDS

Roll No: 387201103

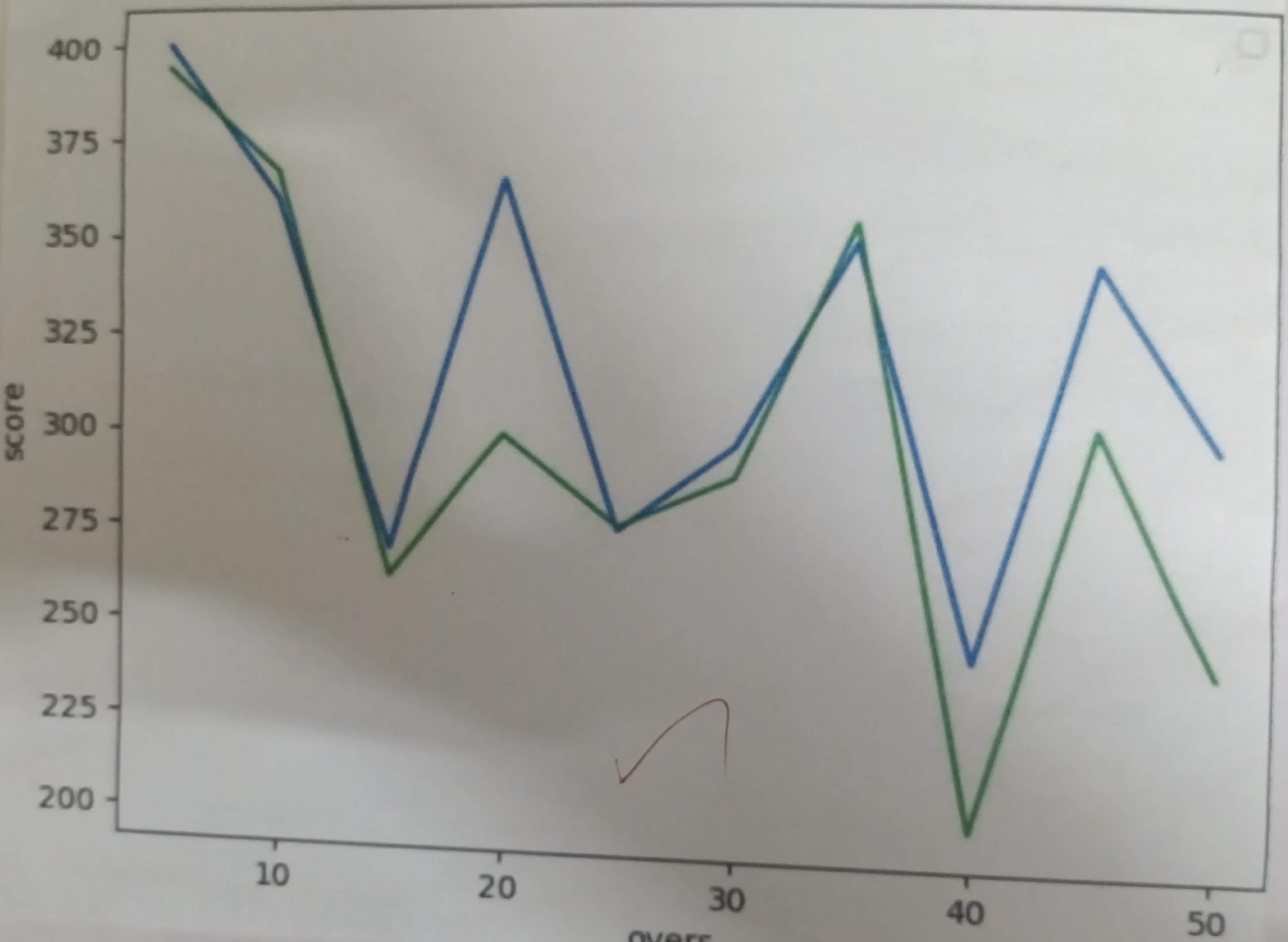
Schad/Collage

Sl No.	Date	Title	Page No	Remarks
1.a	17/7/25	Line Plot	?	
1.b	17/7/25	Bar chart	?	
1.c	17/7/25	Pie chart	10	
1.d	17/7/25	Scatterplot	10	M1 July 3rd lesson
1.e	17/7/25	histogram	10	
2.	31/7/25	Data Analysis and Visualization	10	✓ pr Topic
3.	7/8/25	Data Preprocessing	10	✓ pr Topic
4.	7/8/25	Handling Missing and Inappropriate Data in a Data Set	10	✓ pr Topic
5.	14/8/25	Outliers	10	✓ pr
6.	14/8/25	Feature Scaling	10	✓ pr
7.	28/8/25	EDA	10	✓ pr
8.	18/9/25	Linear Regression	10	✓ pr
9.		Logistic Regression	10	✓ pr
10.		K-Mean clustering	10	✓ pr
11.		KNN	10	✓ pr
12.		T-test	10	✓ pr
13.		Z-test	10	✓ pr
14.		Anova test	10	✓ pr

Completion

In Progress

India Vs Srilanga



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")
```

Q1
Topic 1081us

Result:

Lineplot using matplotlib library is obtained successfully.



Ex: 1(b) Line Plot

Barchart Using matplotlib

Aim: Line Plot

To obtain Barchart for the data using matplotlib library

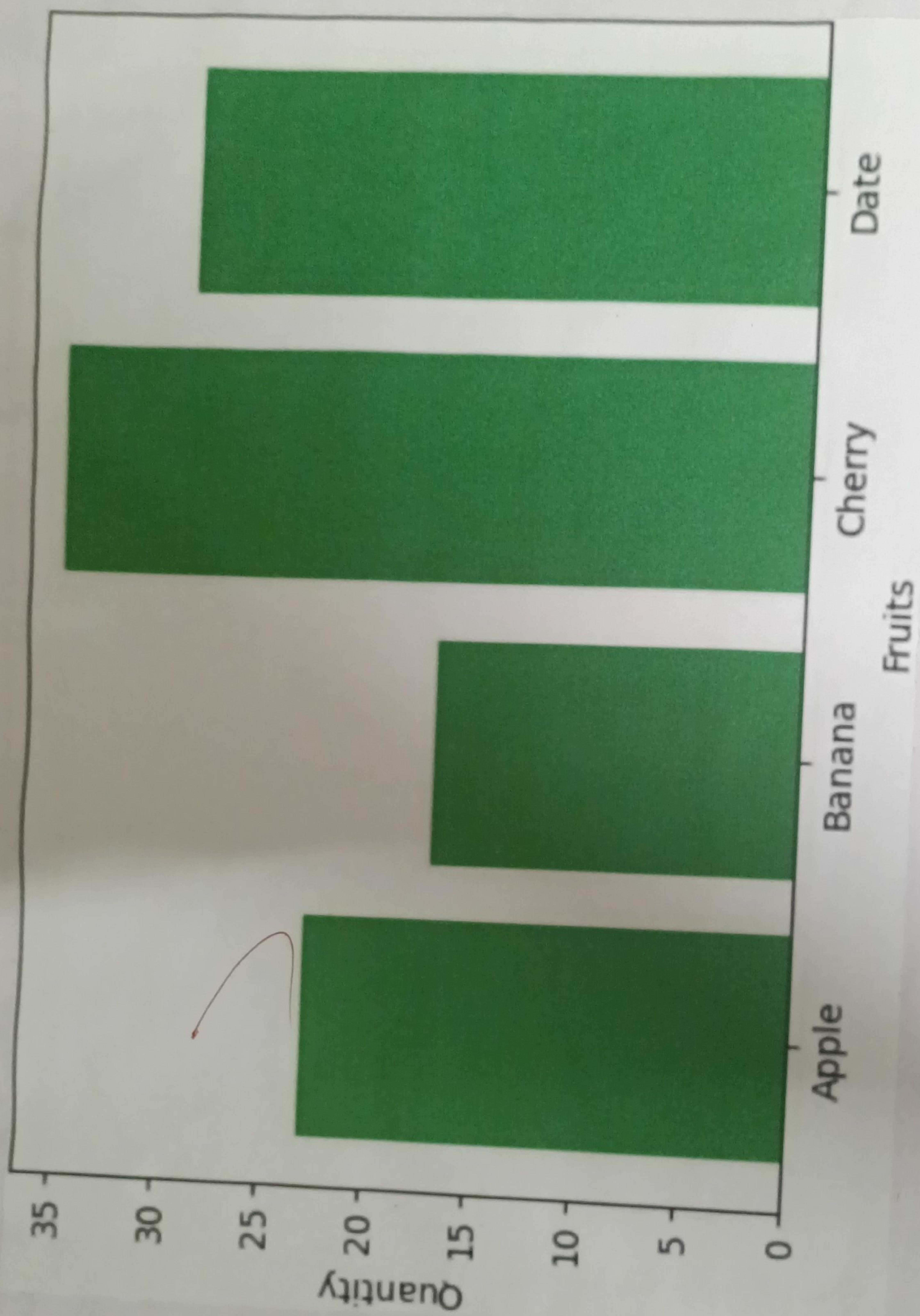
code:

```
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()
```

Dr. P.W.
070811

Result:

Line Plot
Barchart using matplotlib library is obtained successfully.



(Handwritten notes below chart)

What percentage is each fruit of total?

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()
```

Q. Ans
Portfolios

Result:

Bar Chart using matplotlib library is obtained successfully.



Ex: 1(d)

Scatter Plot

Aim:

To obtain scatter plot for data using matplotlib library.

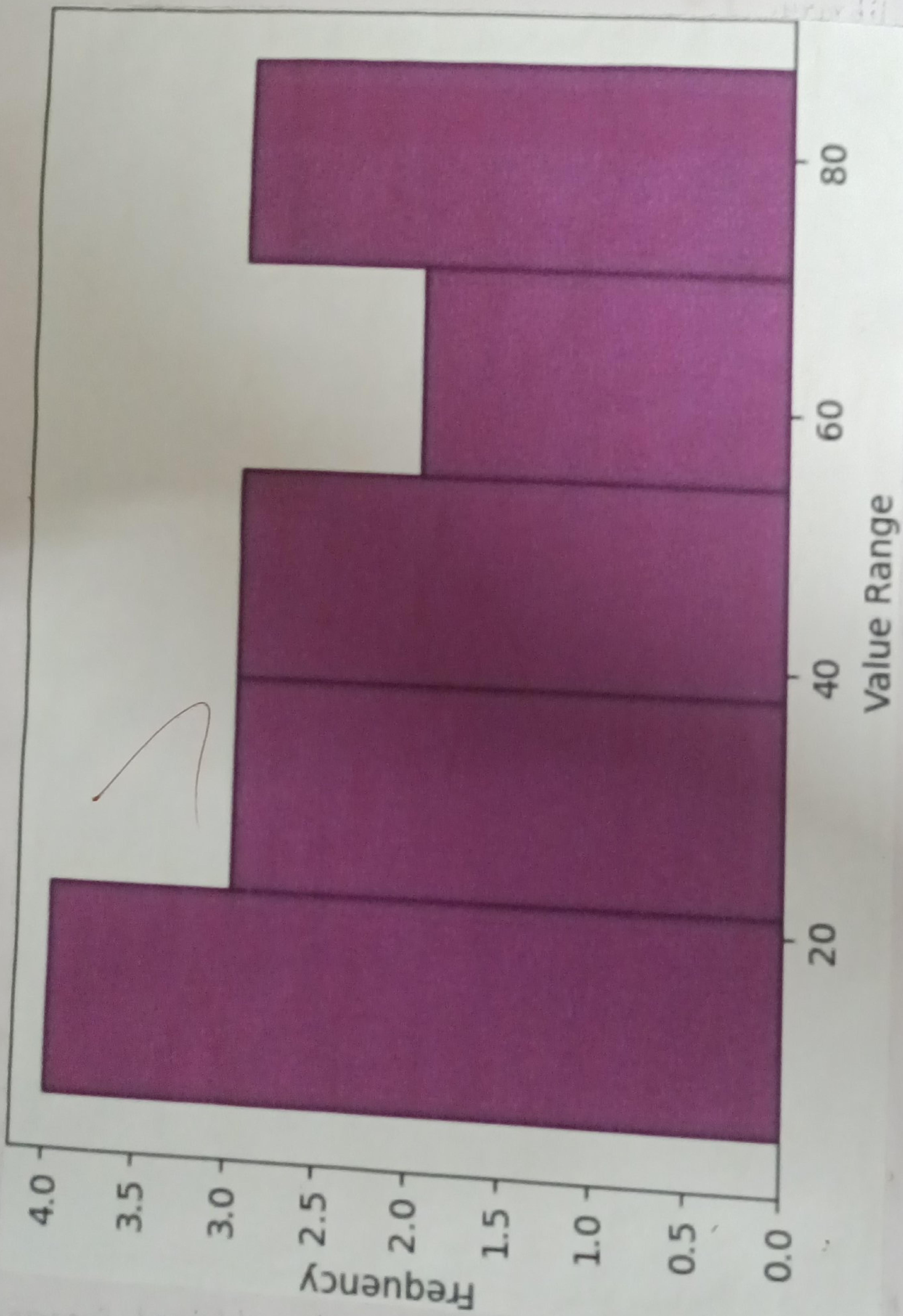
Code:

```
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()
```

Ans: Python

Result:

Scatter plot using matplotlib library is obtained successfully.



Ex: 1(e)

Histogram

To obtain Histogram for data using matplotlib library

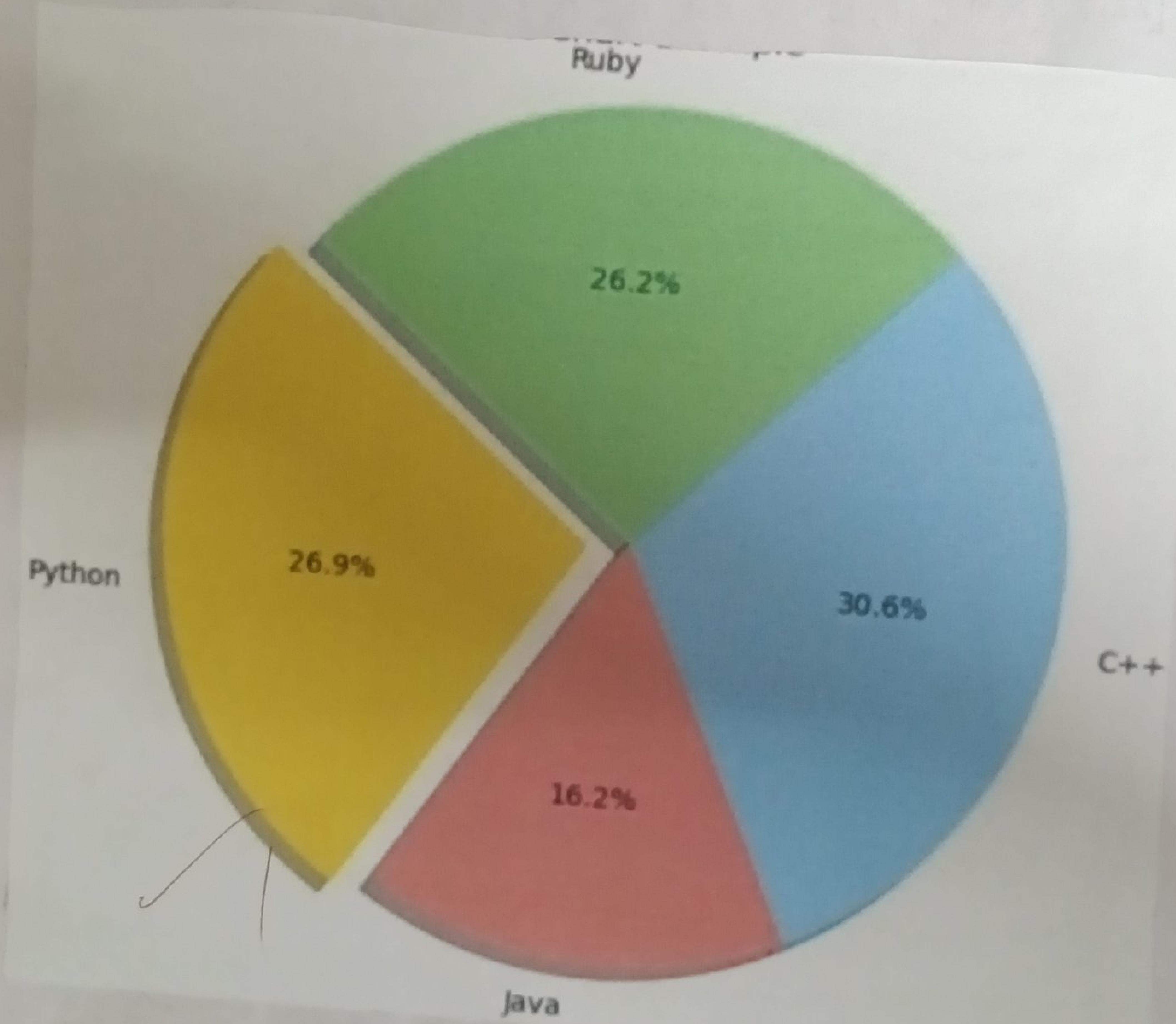
Code:

```
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()
```

a. *RV
09/08/15*

Result:

Histogram using matplotlib library is obtained



2. (1) Pie chart

Aim:

To obtain pie chart for the data using matplotlib library.

Code:

```
labels = ['Python', 'Java', 'C++', 'Ruby']
sizes = [210, 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=colors, autopct='%1.1f %%',
        shadow=True, startangle=140)
plt.title("Pie chart")
plt.axis('equal')
plt.show()
```



Dr. R. P. Patel

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

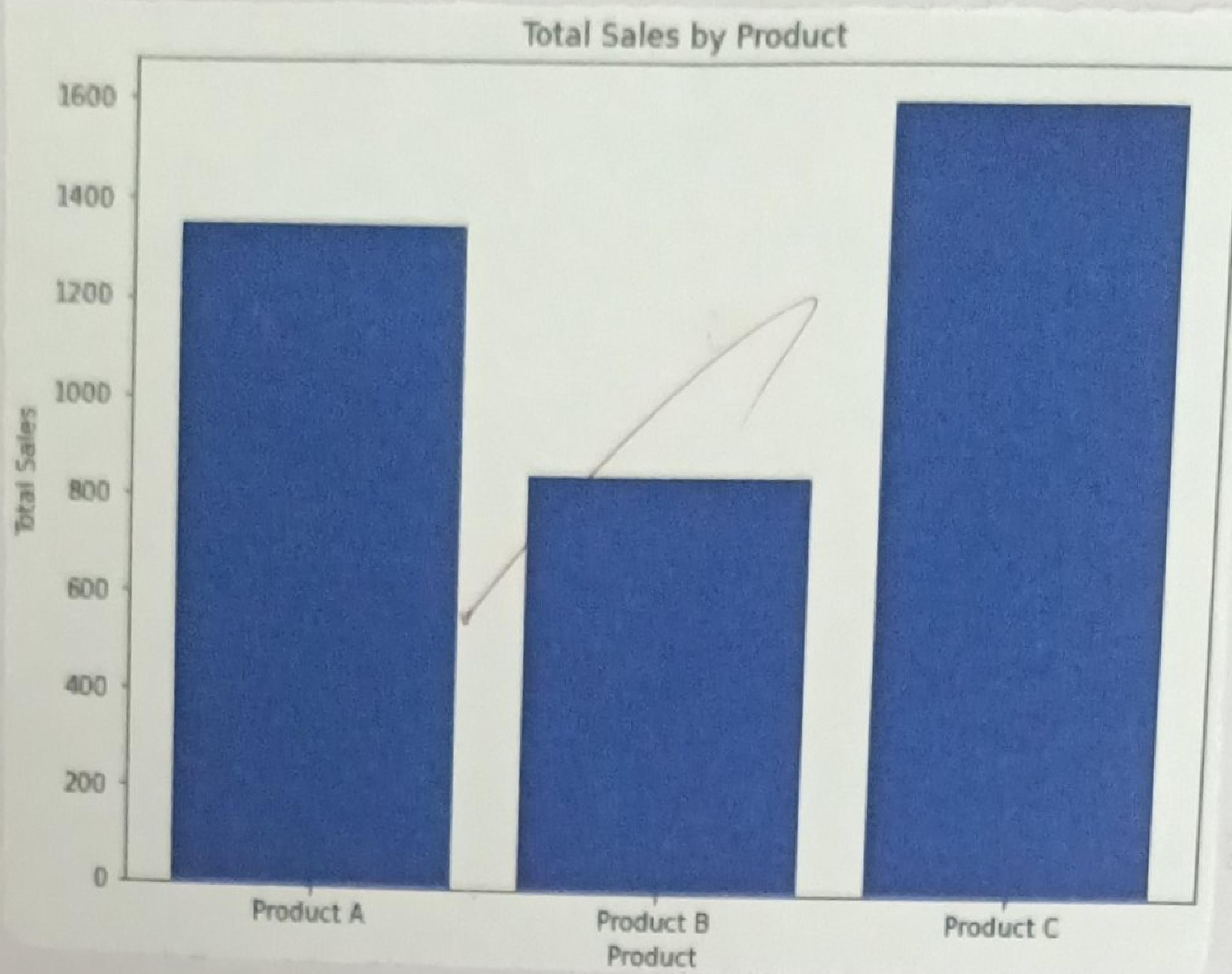
Description:

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

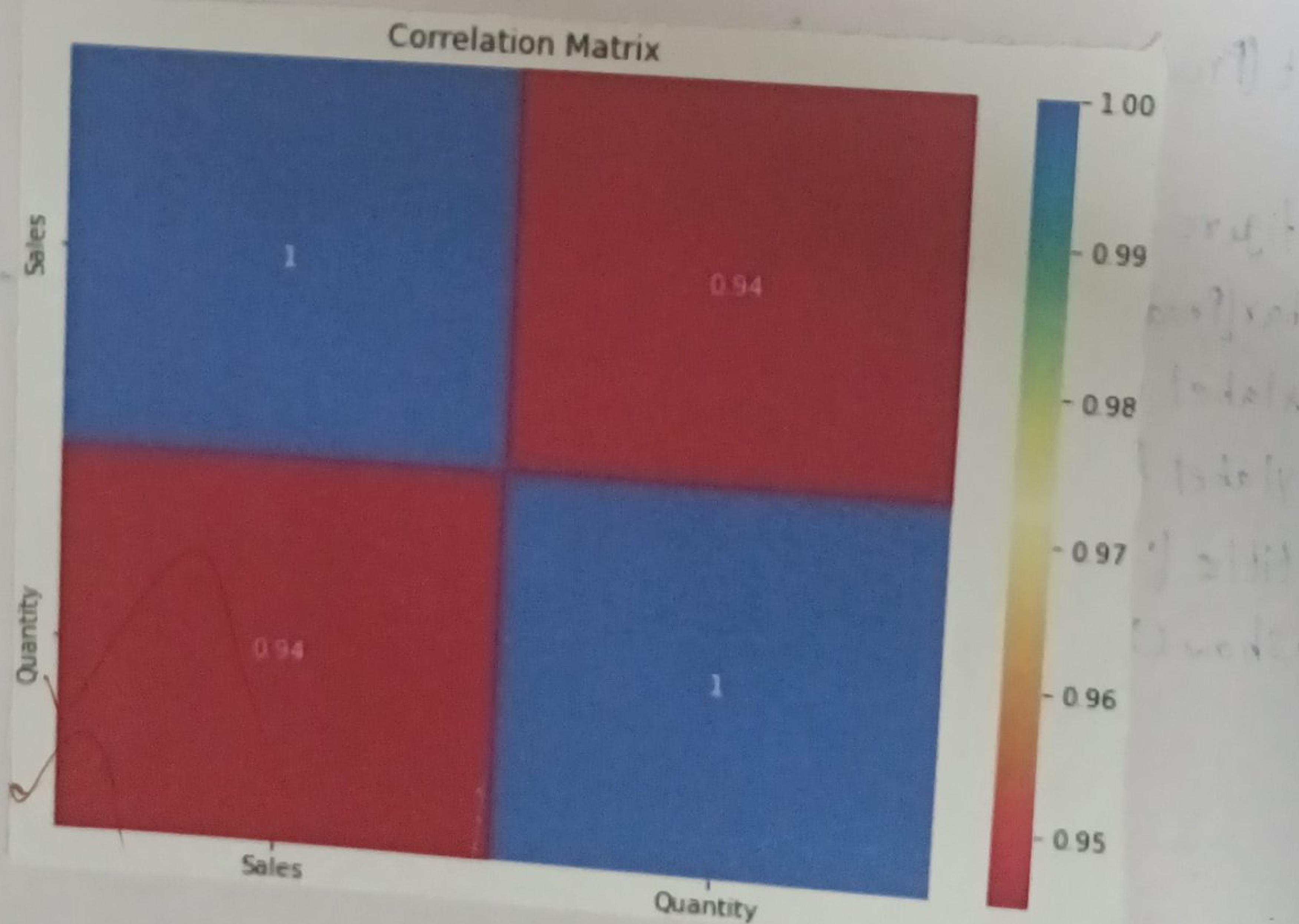
Code:

```
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['Date'], sales_over_time['Sales'])
plt.xlabel('Date')
plt.ylabel('Total Sales')
plt.title('Sales over time')
plt.show()
```

```
pivot_table = df.pivot_table(values='Sales', index='Region', columns='Product', aggfunc=np.sum, fill_value=0)
```

```
print(correlation_matrix)
```

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

(10
10)
A 100%
0.710615

Result:

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

Ex: 3

Data Preprocessing

Aim:

Experiment to understand the "data preprocessing" in Data Science.

code

```
# Rayyan 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
France	44.0	72000.0	No
Spain	27.0	48000.0	Yes
Germany	30.0	54000.0	No
Spain	38.0	61000.0	No
Germany	40.0	NaN	Yes
France	35.0	58000.0	Yes
Spain	Nan	52000.0	No
France	48.0	79000.0	Yes
Germany	50.0	83000.0	No
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().iloc[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

#Rayyan 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
1	2	30-35	5	LemonTree	Non-Veg	2000	3
2	3	25-30	6	RedFox	Veg	1322	2
3	4	20-25	-1	LemonTree	Veg	1234	2
4	5	35+	3	Ibis	Vegetarian	989	2
5	6	35+	3	Ibys	Non-Veg	1909	2
6	7	35+	4	RedFox	Vegetarian	1000	-1
7	8	20-25	7	LemonTree	Veg	2999	-10
8	9	25-30	2	Ibis	Non-Veg	3456	3
9	10	30-35	5	RedFox	non-Veg	-6755	4

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     object 
 2   Rating(1-5)     11 non-null     int64  
 3   Hotel            11 non-null     object 
 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	EstimatedSalary	Age_Group.1
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	20-25
10	10	30-35	5	RedFox	non-Veg	-6755	4	25-30

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	20-25	25-30
9	25-30	2	Ibis	Non-Veg	3456	3	-99999	25-30
10	30-35	5	RedFox	non-Veg	-6755	4	-99999	25-30

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
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
6	7	35+	3	Ibys	Non-Veg	1909	2	122220
7	8	20-25	4	RedFox	Vegetarian	1000	-1	21122
8	9	25-30	7	LemonTree	Veg	2999	-10	345673
9	10	30-35	2	Ibis	Non-Veg	3456	3	20-25
					-6755	4	-99999	25-30

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
1	2.0	30-35	5.0	LemonTree	Non-Veg	2000.0	3.0
2	3.0	25-30	NaN	RedFox	Veg	1322.0	2.0
3	4.0	20-25	NaN	LemonTree	Veg	1234.0	2.0
4	5.0	35+	3.0	Ibis	Vegetarian	989.0	2.0
5	6.0	35+	3.0	Ibys	Non-Veg	1909.0	2.0
6	7.0	35+	4.0	RedFox	Vegetarian	1000.0	NaN
7	8.0	20-25	NaN	LemonTree	Veg	2999.0	NaN
8	9.0	25-30	2.0	Ibis	Non-Veg	3456.0	3.0
9	10.0	30-35	5.0	RedFox	non-Veg	NaN	4.0

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
1	2.0	30-35	5.0	LemonTree	Non-Veg	2000.0	3
2	3.0	25-30	NaN	RedFox	Veg	1322.0	2
3	4.0	20-25	NaN	LemonTree	Veg	1234.0	2
4	5.0	35+	3.0	Ibis	Vegetarian	989.0	2
5	6.0	35+	3.0	Ibys	Non-Veg	1909.0	2
6	7.0	35+	4.0	RedFox	Vegetarian	1000.0	-1
7	8.0	20-25	NaN	LemonTree	Veg	2999.0	-10
8	9.0	25-30	2.0	Ibis	Non-Veg	3456.0	3
9	10.0	30-35	5.0	RedFox	non-Veg	NaN	4

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['FoodPreference'].replace(['Vegetarian', 'Veg'], 'Veg', inplace=True)
df['FoodPreference'].replace(['non-Veg'], 'Non-Veg', inplace=True)
df['EstimatedSalary'].fillna(round(df['NoOfPax'].mean()), 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
```

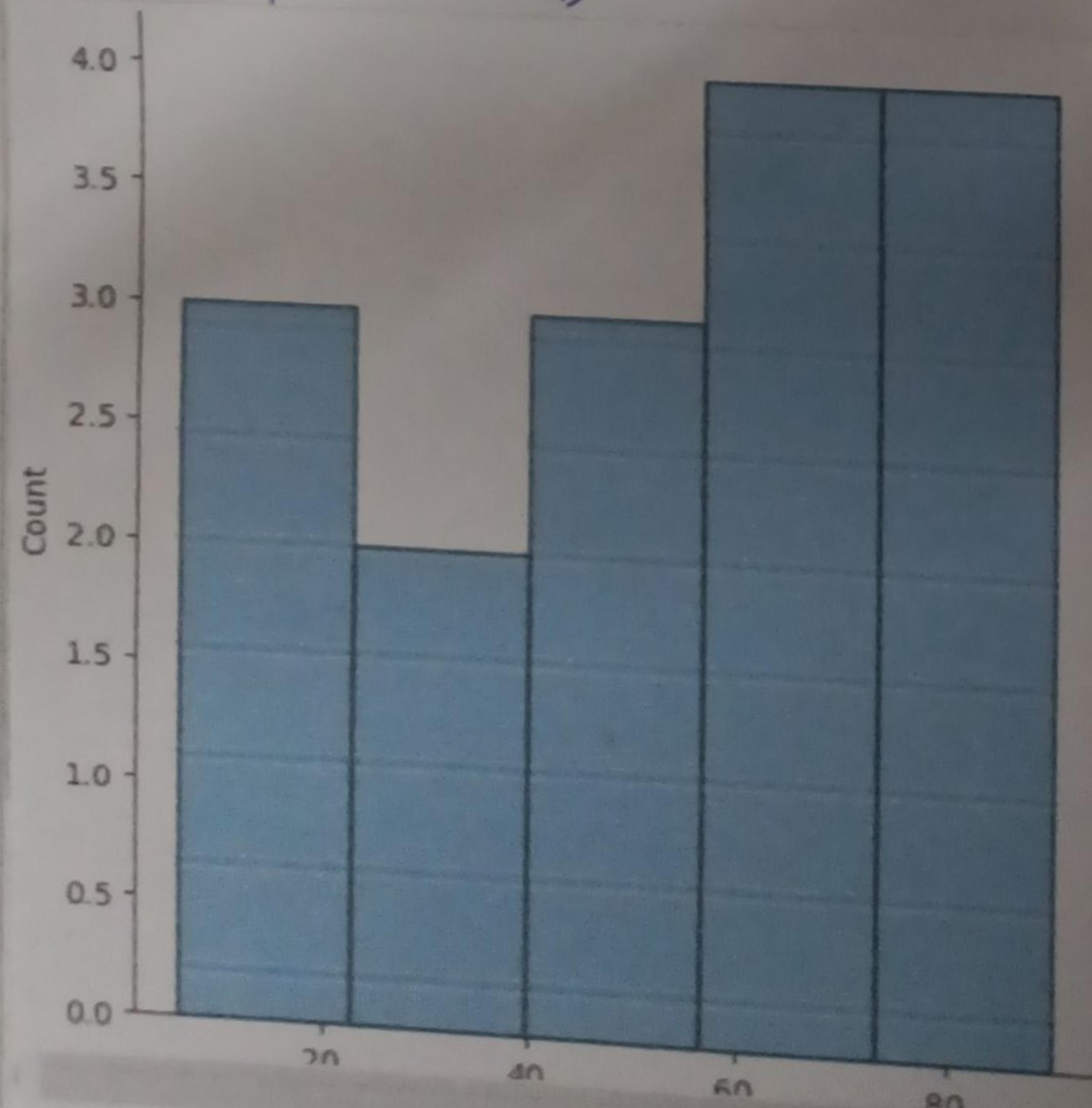
```
lr, ur = outDetection(array)
```

```
lr, ur
```

```
import seaborn as sns
```

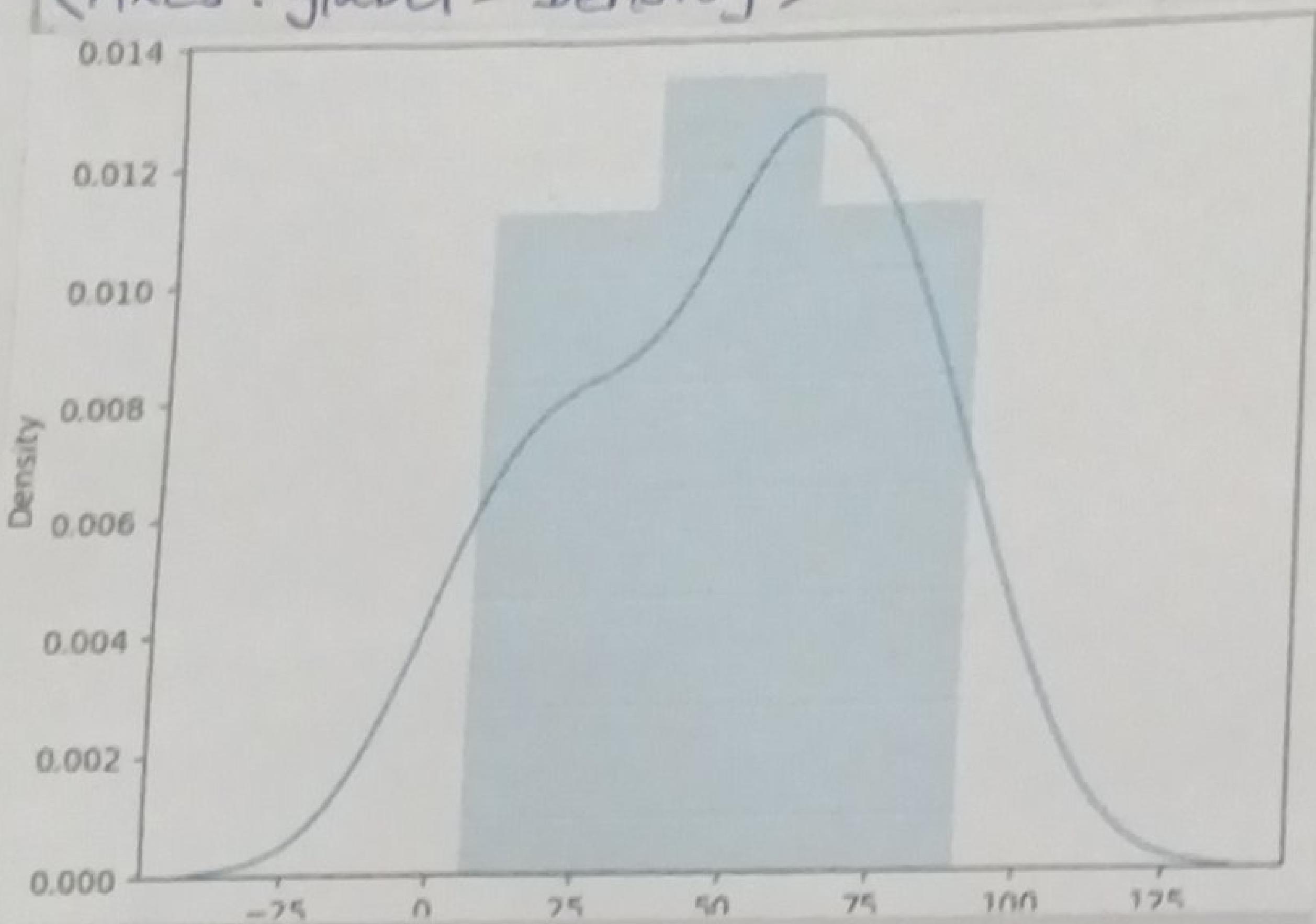
```
%matplotlib inline
```

```
sns.distplot(array)
```



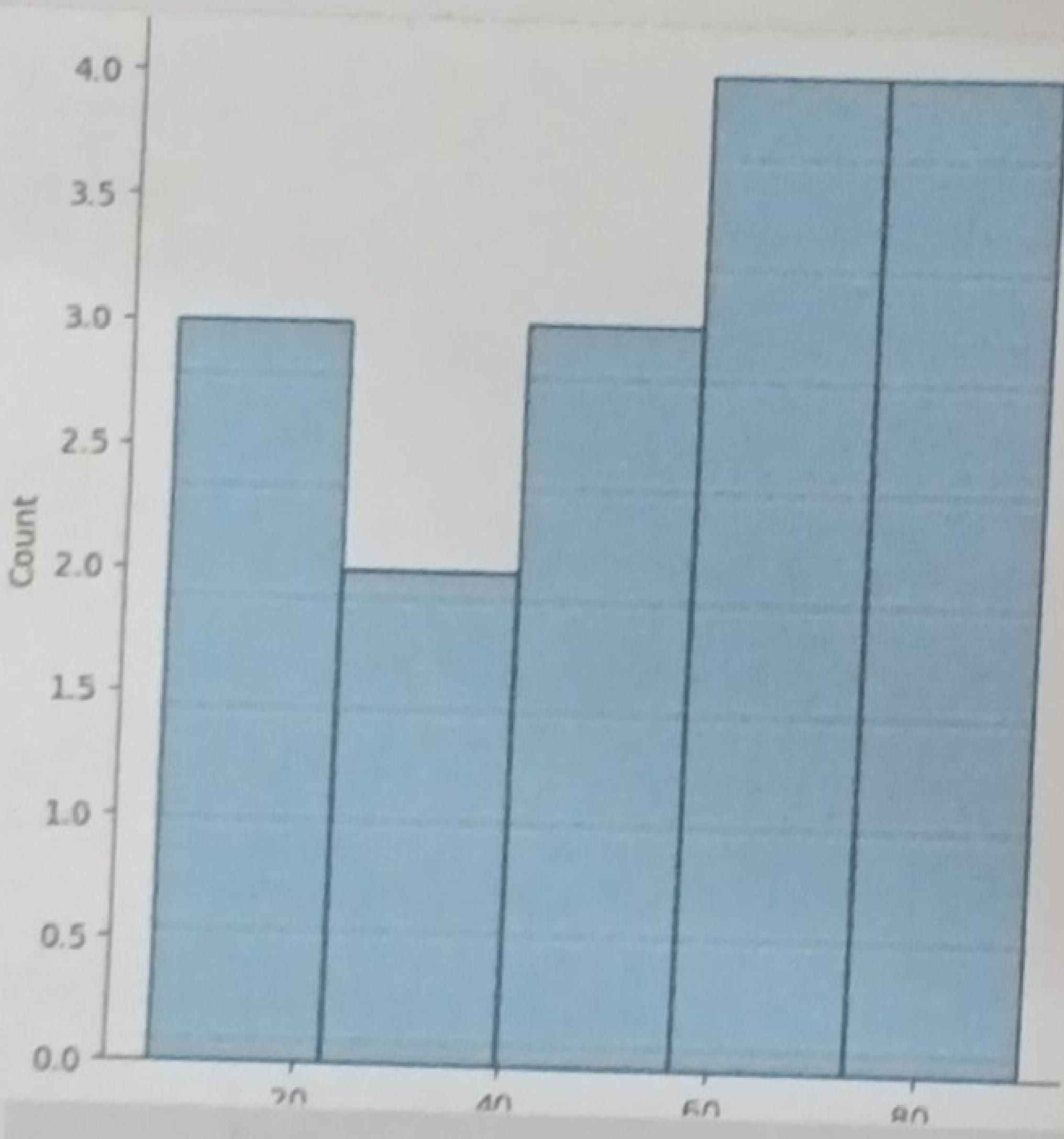
```
sns.distplot(array)
```

Sns.distplot(array)
(Axes: ylabel='Density')



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

Sns.distplot(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 feature Scaling

Code:

```
#
```

```
#
```

```
import numpy as np
```

```
import pandas as pd
```

```
df = pd.read_csv('pre-processdatasample.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[:, [0]])
```

```
* SimpleImputer  
SimpleImputer()  
|
```

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

```
* SimpleImputer  
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, 63777.7777777778],  
      ['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., 1., 0.],  
      [0., 0., 1.],  
      [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, 63777.7777777778],  
      [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([[ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
        7.58874362e-01,  7.49473254e-01],  
      [-1.0000000e+00, -5.0000000e-01,  1.52752523e+00,  
       -1.71150388e+00, -1.43817841e+00],  
      [-1.0000000e+00,  2.0000000e+00, -6.54653671e-01,  
       -1.27555478e+00, -8.91265492e-01],  
      [-1.0000000e+00, -5.0000000e-01,  1.52752523e+00,  
       -1.13023841e-01, -2.53200424e-01],  
      [-1.0000000e+00,  2.0000000e+00, -6.54653671e-01,  
       1.77608893e-01,  6.63219199e-16],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       -5.48972942e-01, -5.26656882e-01],  
      [-1.0000000e+00, -5.0000000e-01,  1.52752523e+00,  
       0.0000000e+00, -1.07356980e+00],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       1.34013983e+00,  1.38753832e+00],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       1.63077256e+00,  1.75214693e+00],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       2.59340209e-01,  2.75555556e-01],  
      [-1.0000000e+00, -5.0000000e-01,  1.52752523e+00,  
       0.0000000e+00, -1.07356980e+00],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       1.34013983e+00,  1.38753832e+00],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       1.63077256e+00,  1.75214693e+00],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       2.59340209e-01,  2.75555556e-01],  
      [-1.0000000e+00, -5.0000000e-01,  1.52752523e+00,  
       0.0000000e+00, -1.07356980e+00],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       1.34013983e+00,  1.38753832e+00],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       1.63077256e+00,  1.75214693e+00],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       2.59340209e-01,  2.75555556e-01],  
      [-1.0000000e+00, -5.0000000e-01,  1.52752523e+00,  
       0.0000000e+00, -1.07356980e+00],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       1.34013983e+00,  1.38753832e+00],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       1.63077256e+00,  1.75214693e+00],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       2.59340209e-01,  2.75555556e-01],  
      [-1.0000000e+00, -5.0000000e-01,  1.52752523e+00,  
       0.0000000e+00, -1.07356980e+00],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       1.34013983e+00,  1.38753832e+00],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       1.63077256e+00,  1.75214693e+00],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       2.59340209e-01,  2.75555556e-01],  
      [-1.0000000e+00, -5.0000000e-01,  1.52752523e+00,  
       0.0000000e+00, -1.07356980e+00],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       1.34013983e+00,  1.38753832e+00],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       1.63077256e+00,  1.75214693e+00],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       2.59340209e-01,  2.75555556e-01],  
      [-1.0000000e+00, -5.0000000e-01,  1.52752523e+00,  
       0.0000000e+00, -1.07356980e+00],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       1.34013983e+00,  1.38753832e+00],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       1.63077256e+00,  1.75214693e+00],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       2.59340209e-01,  2.75555556e-01],  
      [-1.0000000e+00, -5.0000000e-01,  1.52752523e+00,  
       0.0000000e+00, -1.07356980e+00],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       1.34013983e+00,  1.38753832e+00],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       1.63077256e+00,  1.75214693e+00],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       2.59340209e-01,  2.75555556e-01],  
      [-1.0000000e+00, -5.0000000e-01,  1.52752523e+00,  
       0.0000000e+00, -1.07356980e+00],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       1.34013983e+00,  1.38753832e+00],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       1.63077256e+00,  1.75214693e+00],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       2.59340209e-01,  2.75555556e-01],  
      [-1.0000000e+00, -5.0000000e-01,  1.52752523e+00,  
       0.0000000e+00, -1.07356980e+00],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       1.34013983e+00,  1.38753832e+00],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       1.63077256e+00,  1.75214693e+00],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       2.59340209e-01,  2.75555556e-01],  
      [-1.0000000e+00, -5.0000000e-01,  1.52752523e+00,  
       0.0000000e+00, -1.07356980e+00],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       1.34013983e+00,  1.38753832e+00],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       1.63077256e+00,  1.75214693e+00],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       2.59340209e-01,  2.75555556e-01],  
      [-1.0000000e+00, -5.0000000e-01,  1.52752523e+00,  
       0.0000000e+00, -1.07356980e+00],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       1.34013983e+00,  1.38753832e+00],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       1.63077256e+00,  1.75214693e+00],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       2.59340209e-01,  2.75555556e-01],  
      [-1.0000000e+00, -5.0000000e-01,  1.52752523e+00,  
       0.0000000e+00, -1.07356980e+00],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       1.34013983e+00,  1.38753832e+00],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       1.63077256e+00,  1.75214693e+00],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       2.59340209e-01,  2.75555556e-01],  
      [-1.0000000e+00, -5.0000000e-01,  1.52752523e+00,  
       0.0000000e+00, -1.07356980e+00],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       1.34013983e+00,  1.38753832e+00],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       1.63077256e+00,  1.75214693e+00],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       2.59340209e-01,  2.75555556e-01],  
      [-1.0000000e+00, -5.0000000e-01,  1.52752523e+00,  
       0.0000000e+00, -1.07356980e+00],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       1.34013983e+00,  1.38753832e+00],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       1.63077256e+00,  1.75214693e+00],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       2.59340209e-01,  2.75555556e-01],  
      [-1.0000000e+00, -5.0000000e-01,  1.52752523e+00,  
       0.0000000e+00, -1.07356980e+00],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       1.34013983e+00,  1.38753832e+00],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       1.63077256e+00,  1.75214693e+00],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       2.59340209e-01,  2.75555556e-01],  
      [-1.0000000e+00, -5.0000000e-01,  1.52752523e+00,  
       0.0000000e+00, -1.07356980e+00],  
      [ 1.0000000e+00, -5.0000000e-01, -6.54653671e-01,  
       1.34013983e+00,  1.38753832e
```

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

```
array([[1., 0., 0., 0.73913043, 0.68571429],  
       [0., 1., 0., 0., 0.],  
       [0., 0., 1., 0.13843478, 0.17142857],  
       [0., 0., 0., 0.47925097, 0.37142857],  
       [0., 1., 0., 0.56521799, 0.45879365],  
       [0., 0., 0., 0.34782589, 0.28571429],  
       [0., 0., 1., 0.51287729, 0.11142857],  
       [0., 0., 0., 0.91384348, 0.88571429],  
       [0., 0., 0., 0., 0.],  
       [0., 0., 0., 0.43478251, 0.54285714]])
```

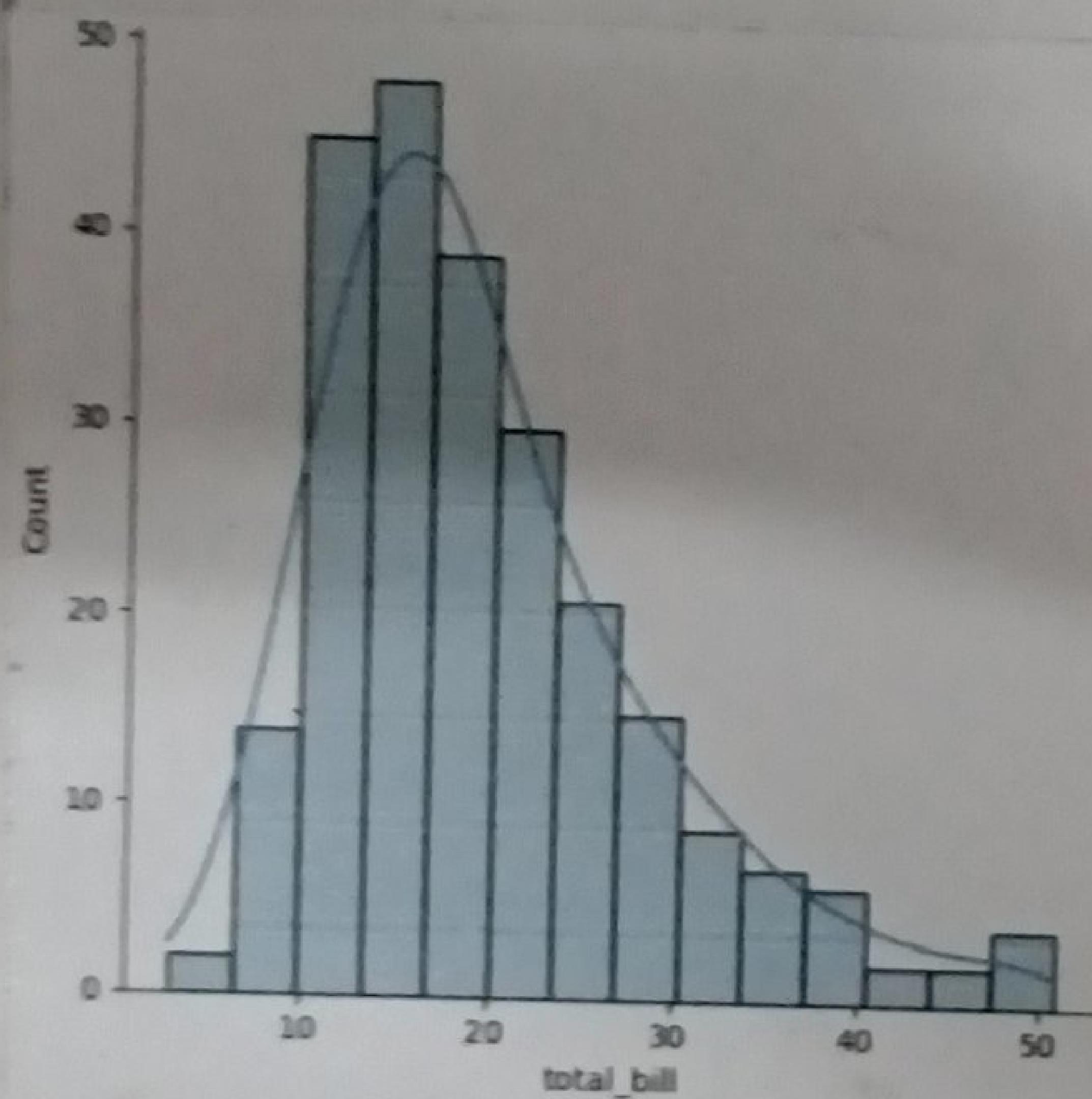
Exp: 7

Code:

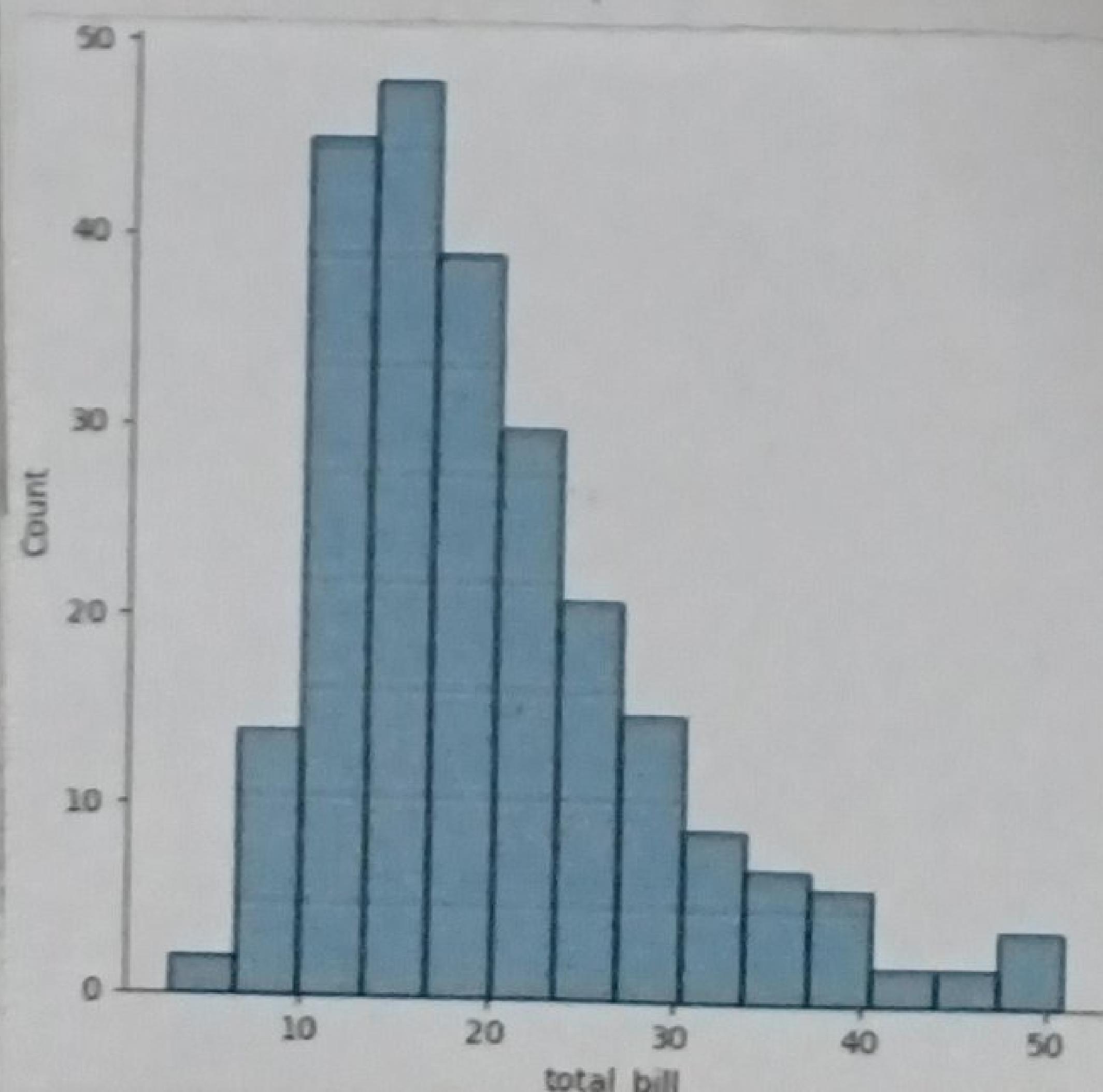
```
#  
#  
import seaborn as sns  
import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
%matplotlib inline  
tips = sns.load_dataset('tips')  
tips.head()
```

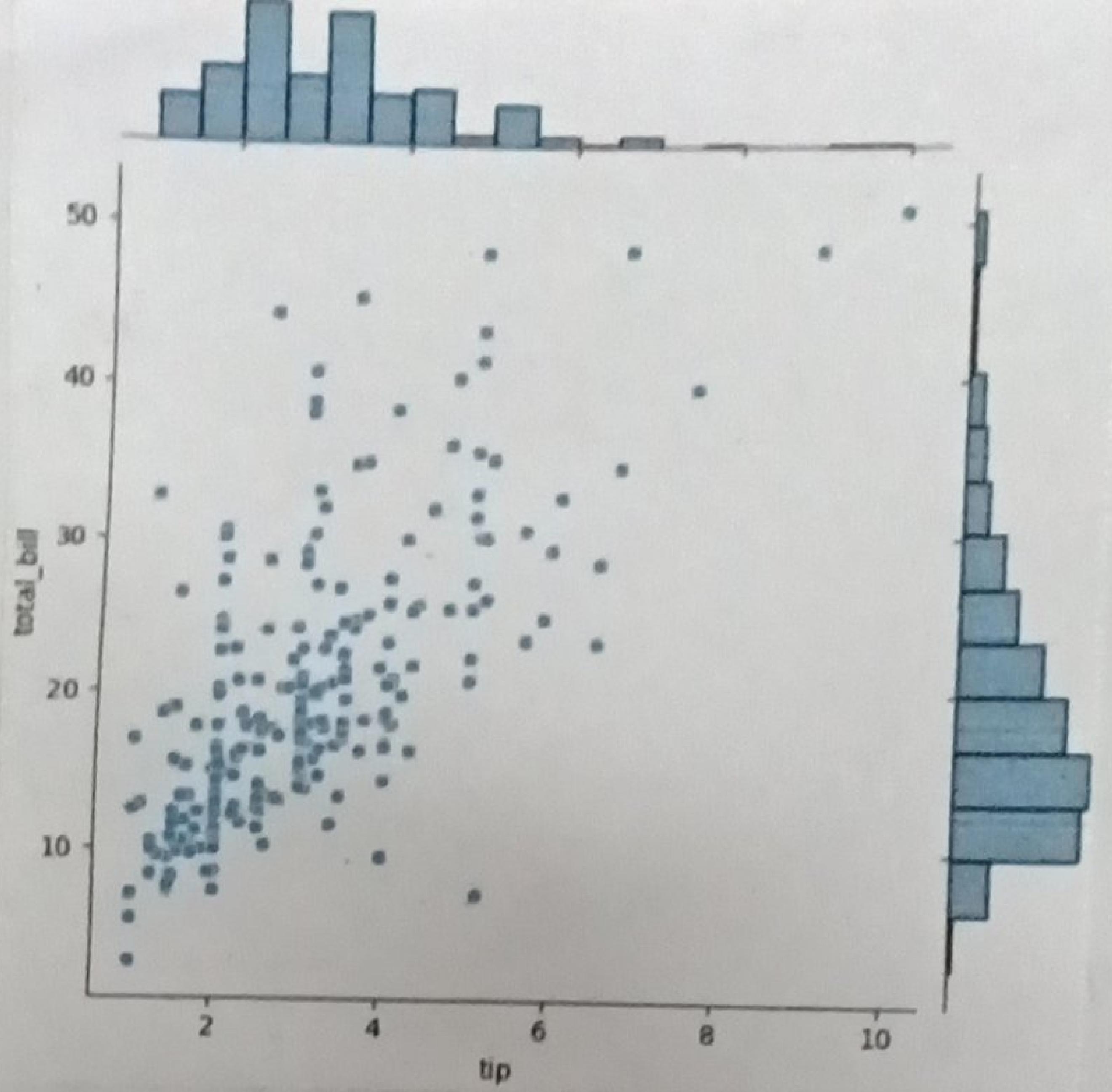
	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

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

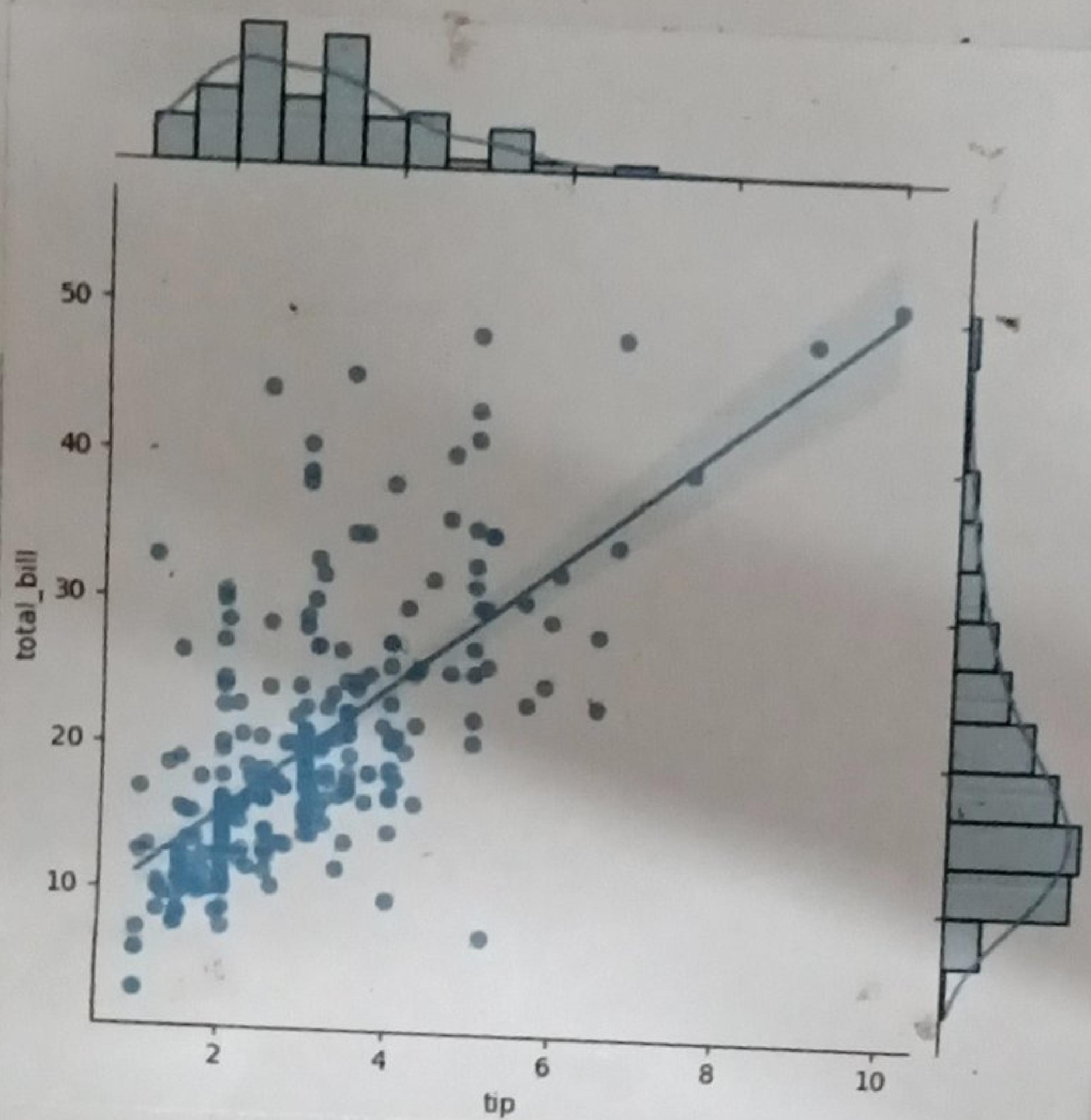


```
sns.displot(tips['total_bill'], kde=False)
```

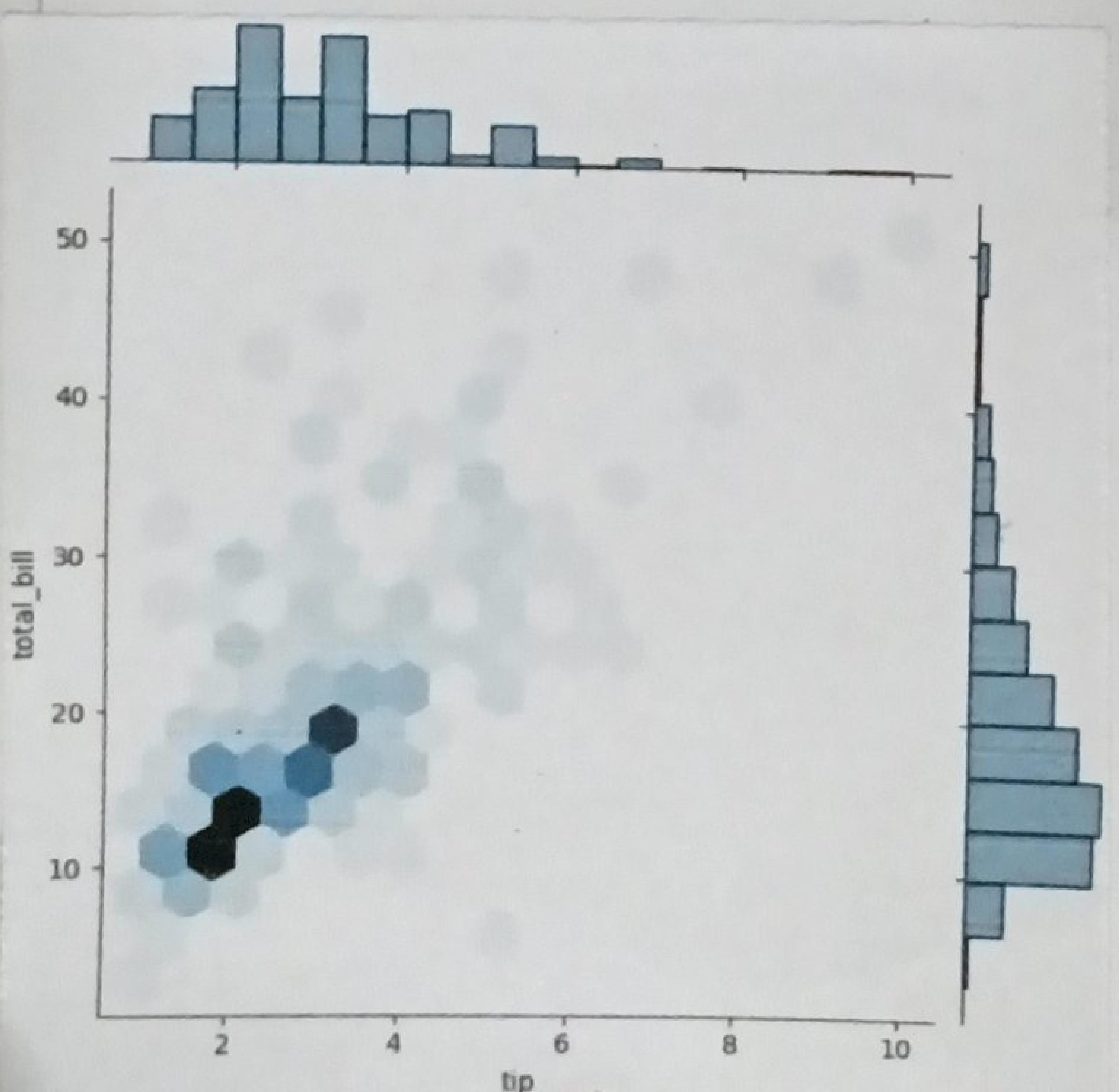


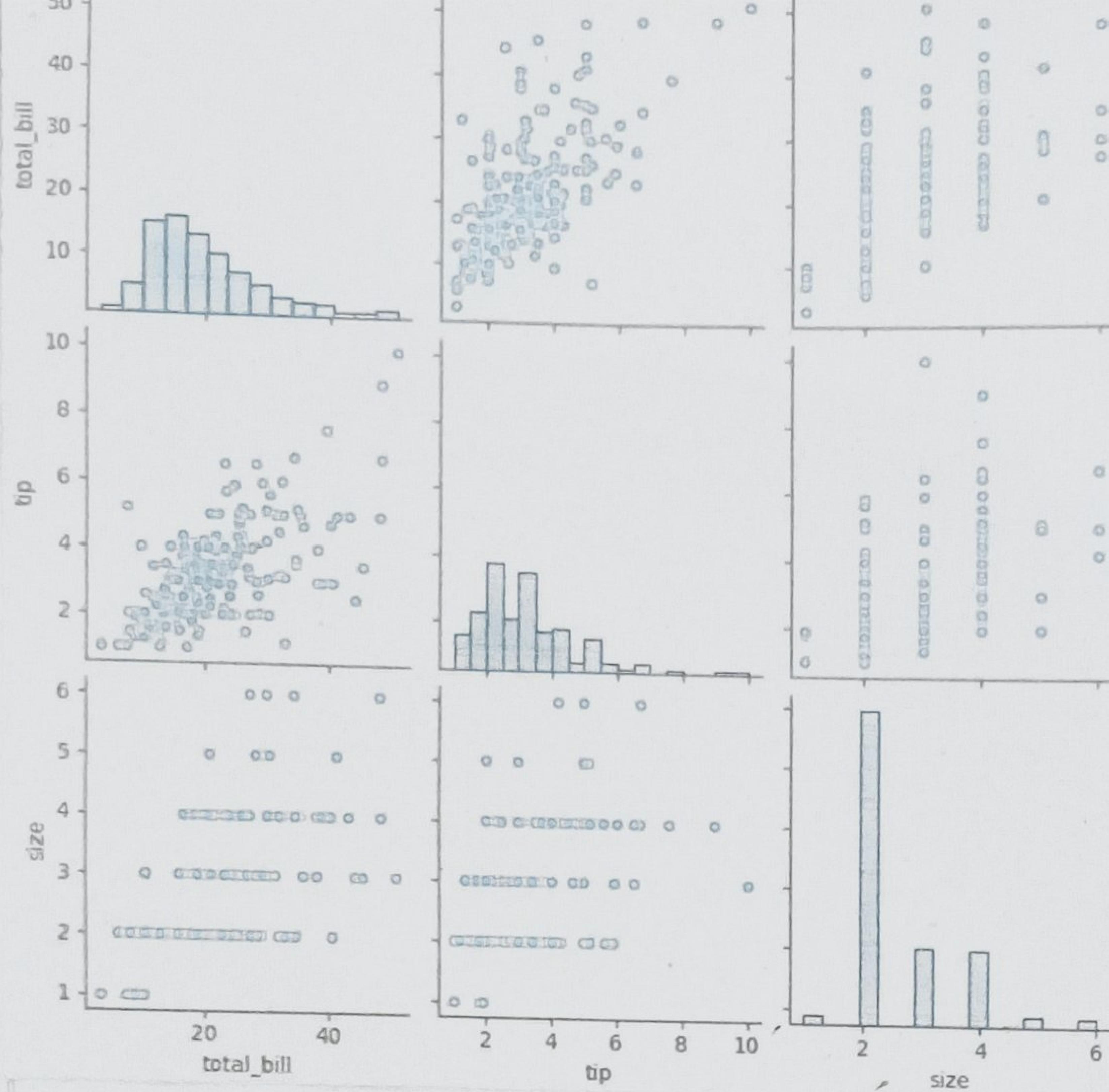


Sns.jointplot(x=tips.tip, y=tips.total_bill, kind="reg")



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

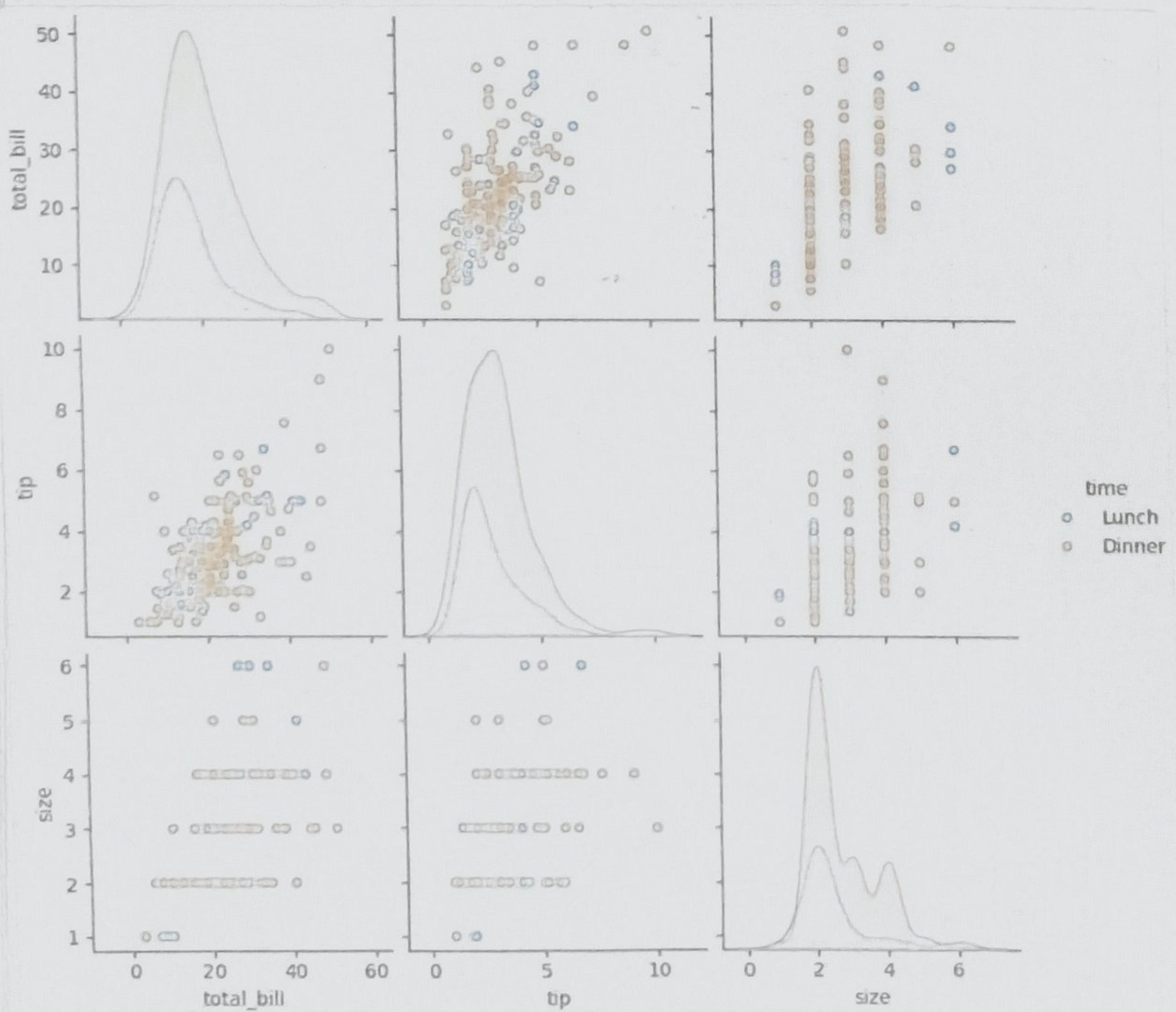




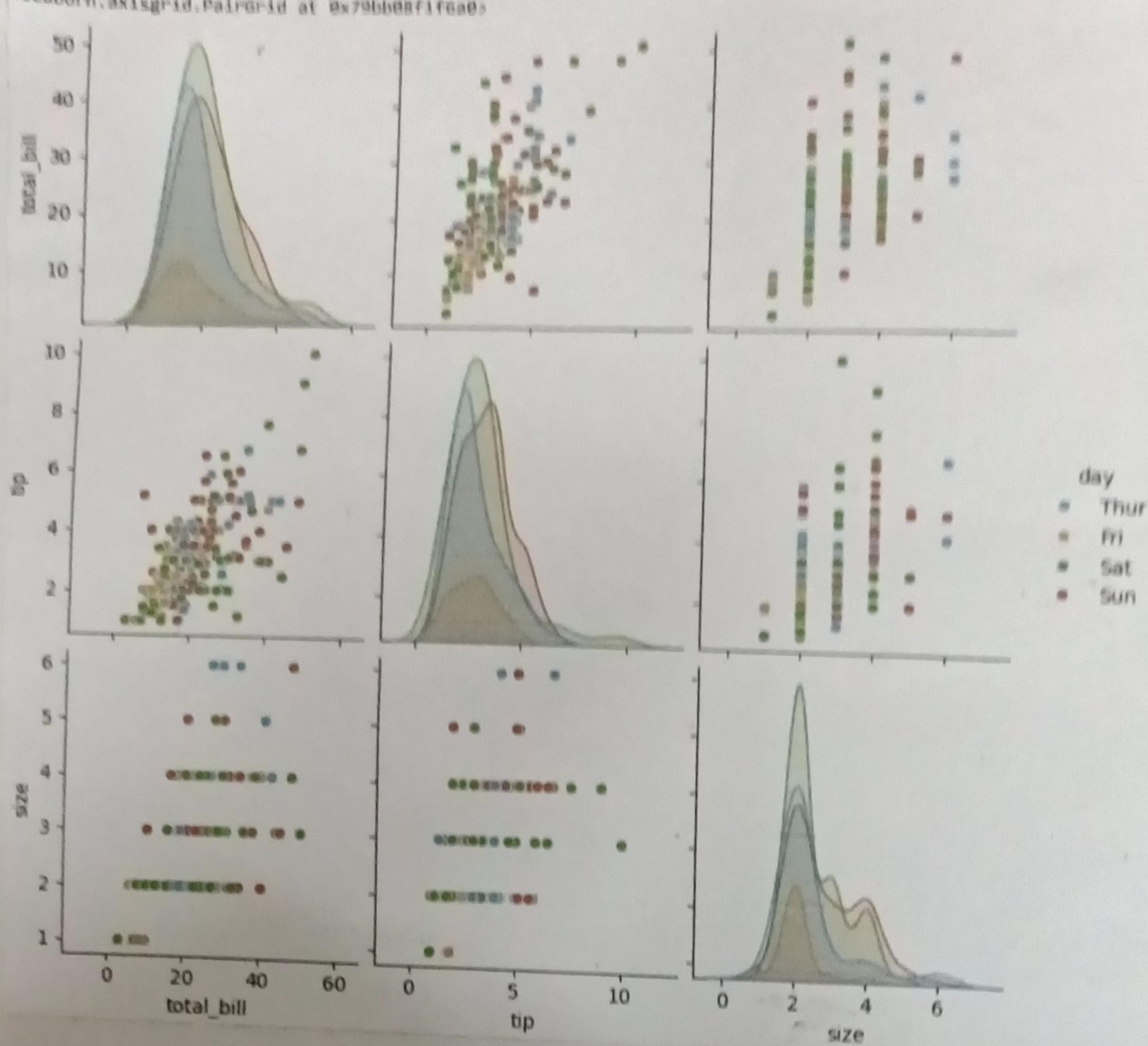
`tips.time.value_counts()`

time	count
Dinner	176
Lunch	68

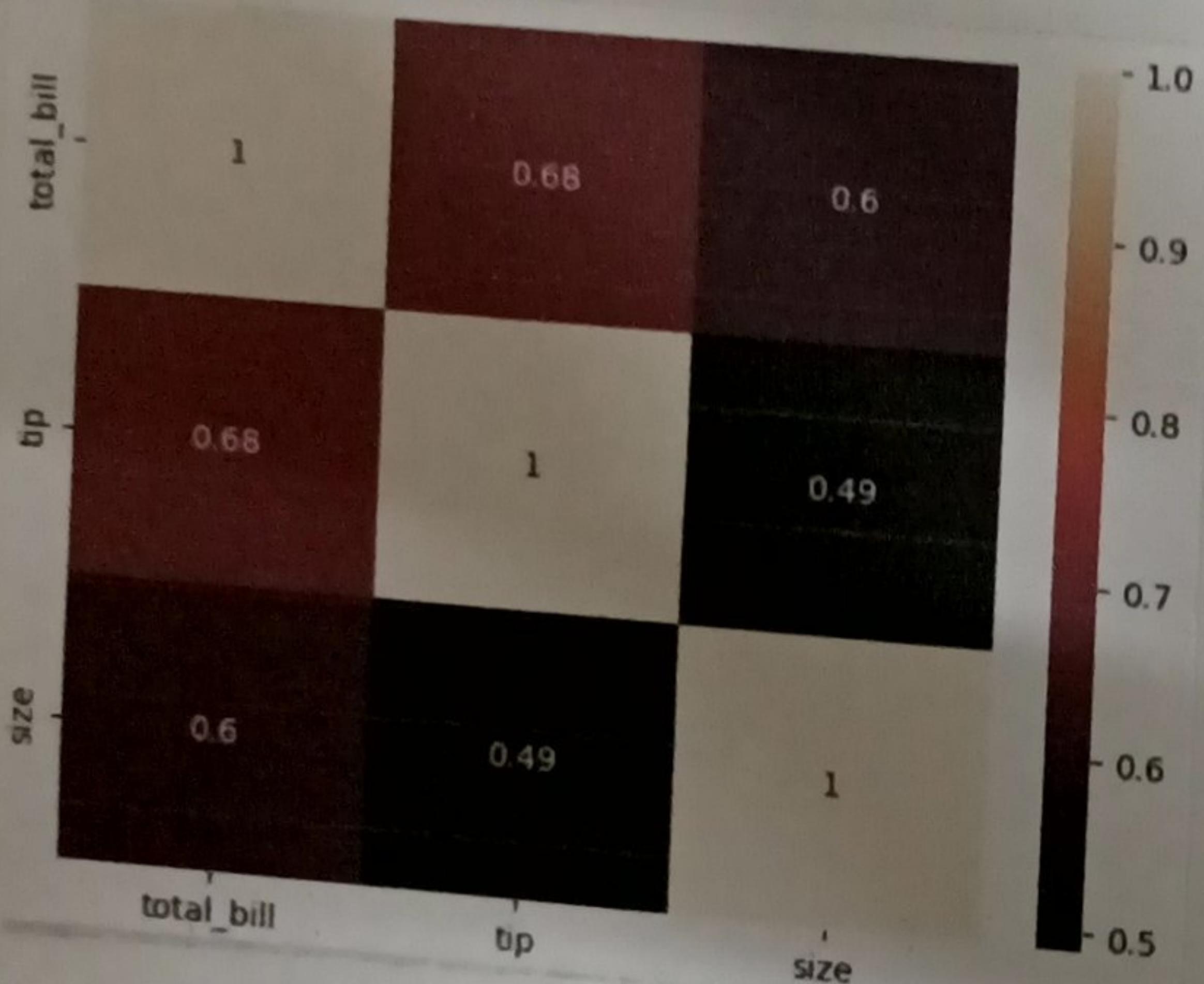
`sns.pairplot(tips, hue='time')`



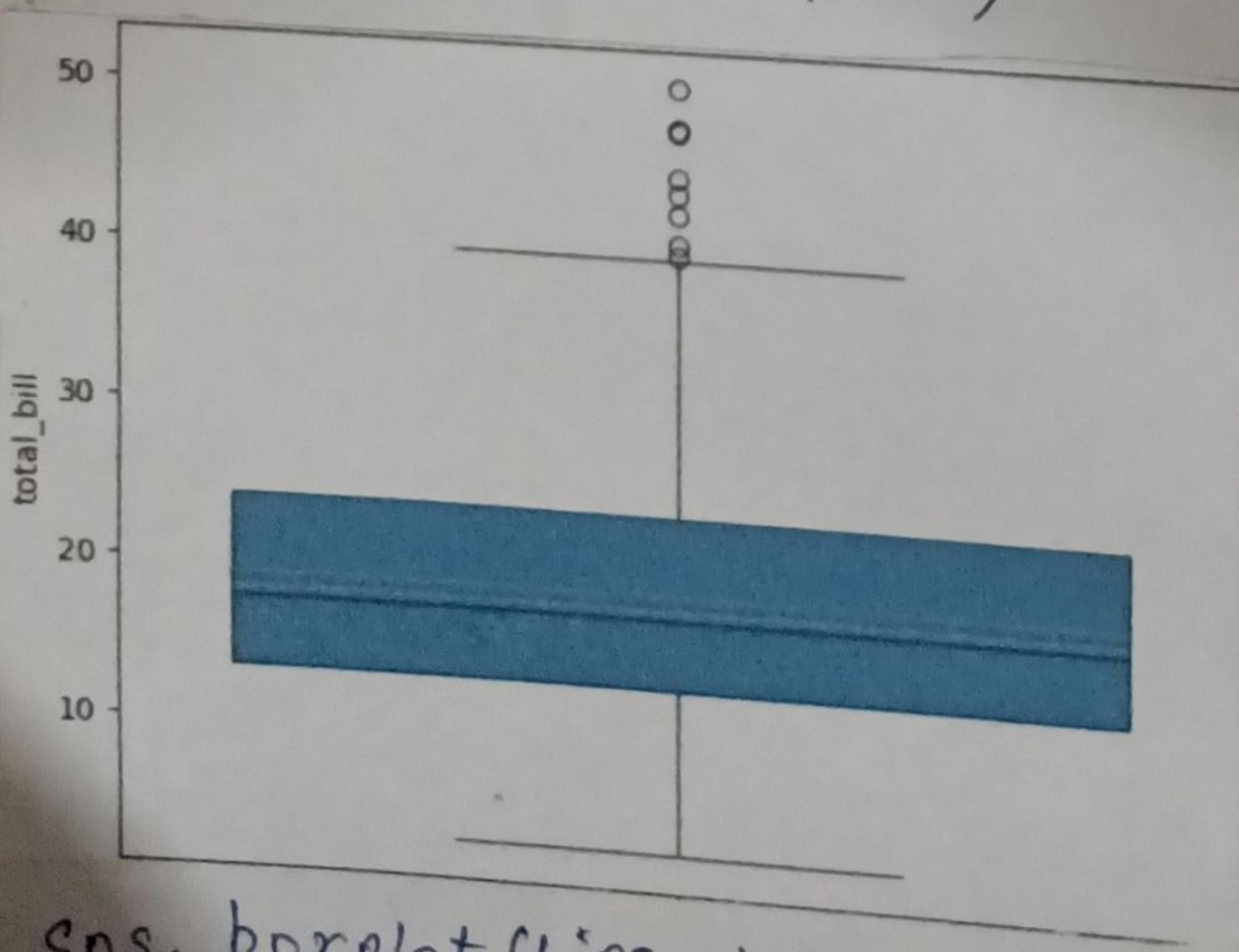
`sns.pairplot(tips, hue='day')`



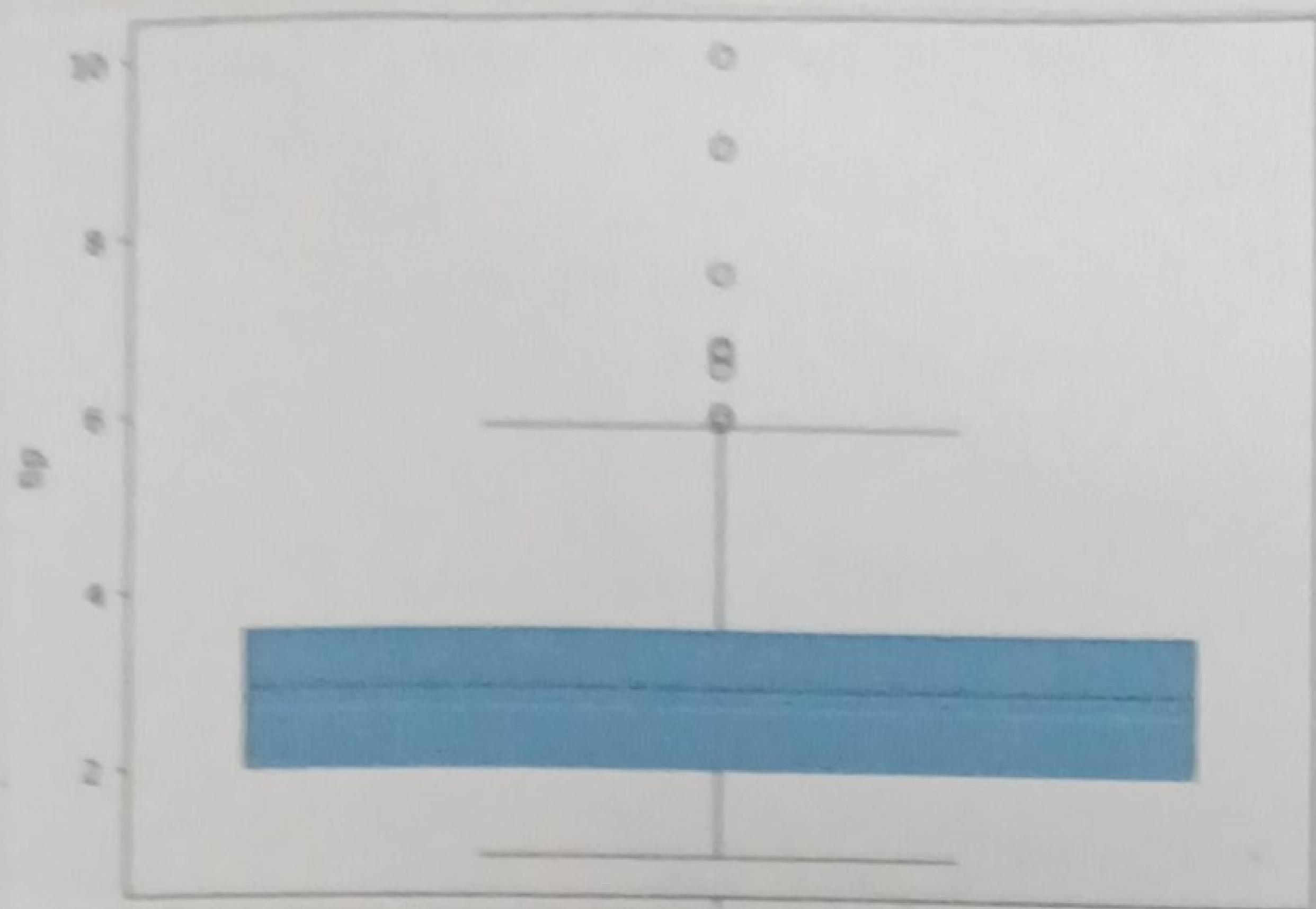
`Sns. heatmap(tips.corr(numeric_only=True), annot=True)`



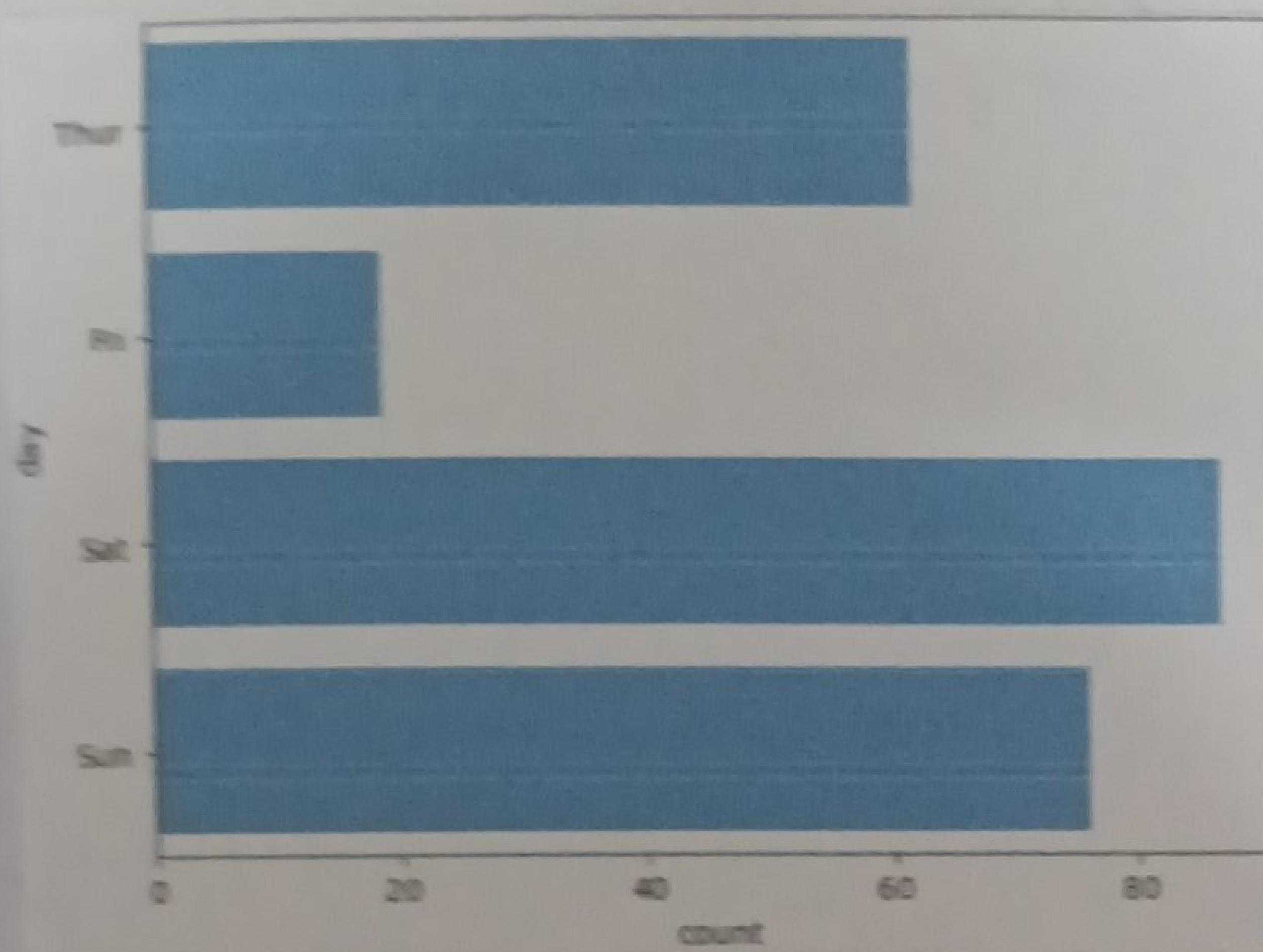
`Sns. boxplot(tips, total_bill)`



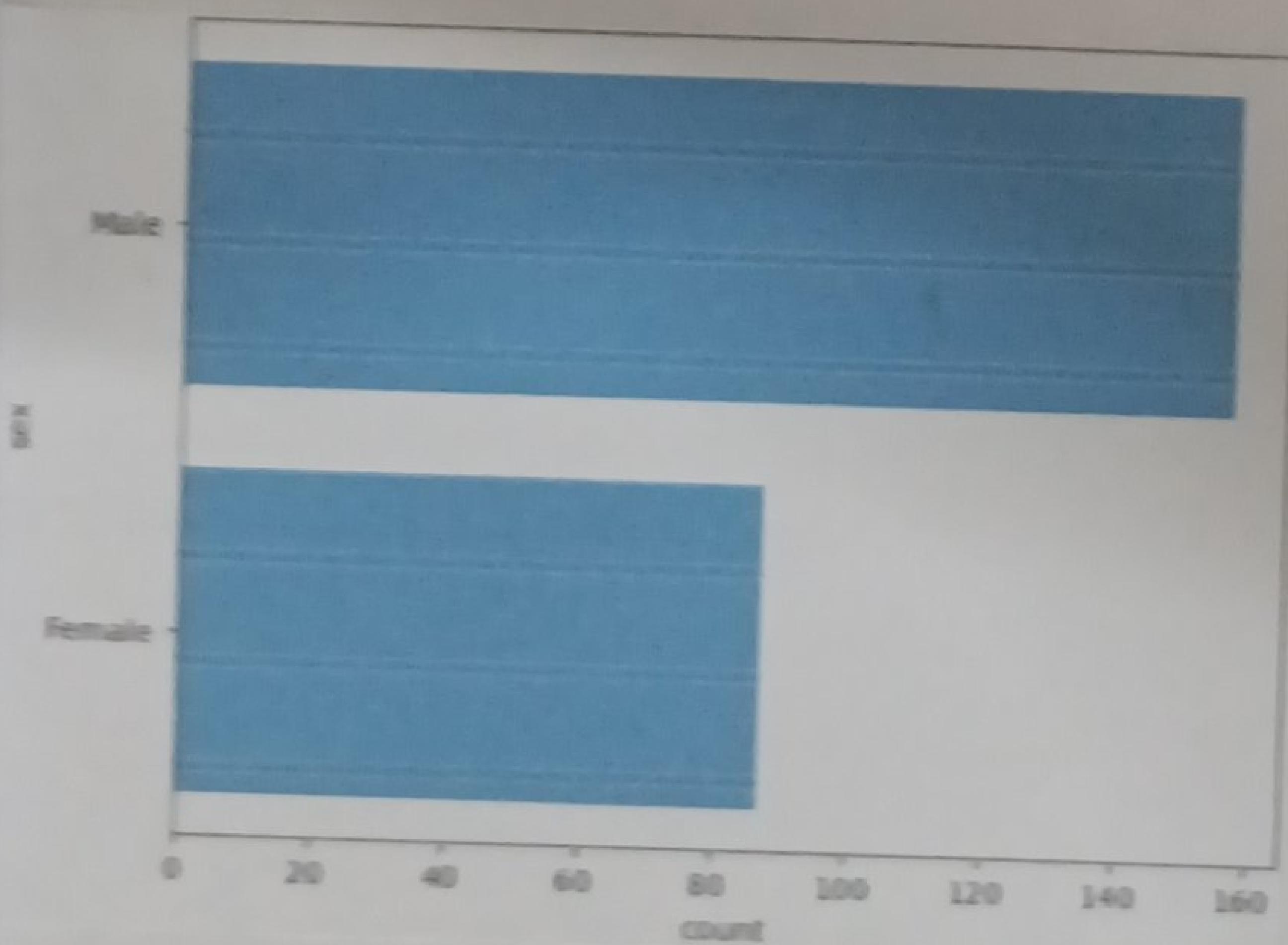
`Sns. boxplot(tips, tip)`



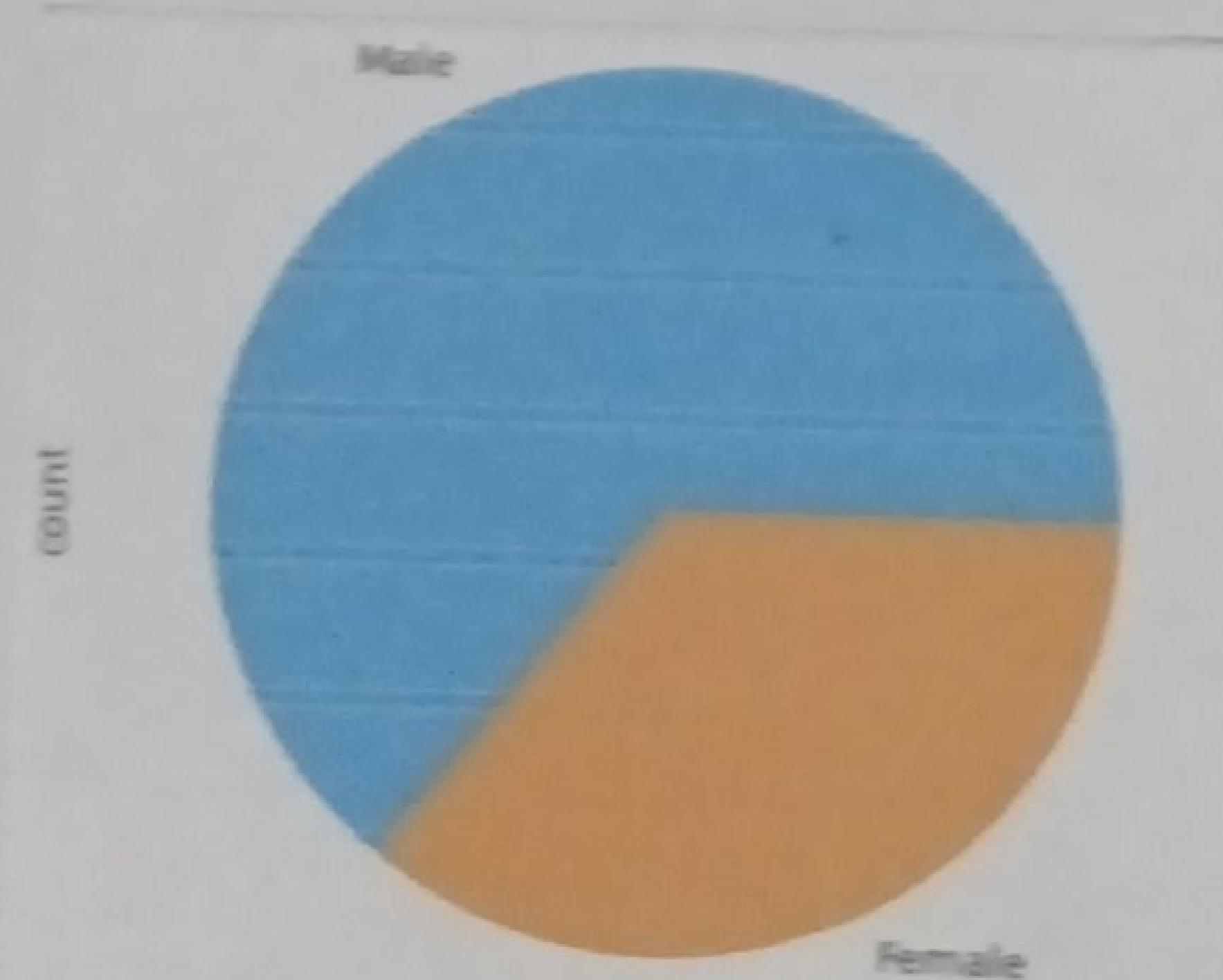
sns.countplot(tips.day)

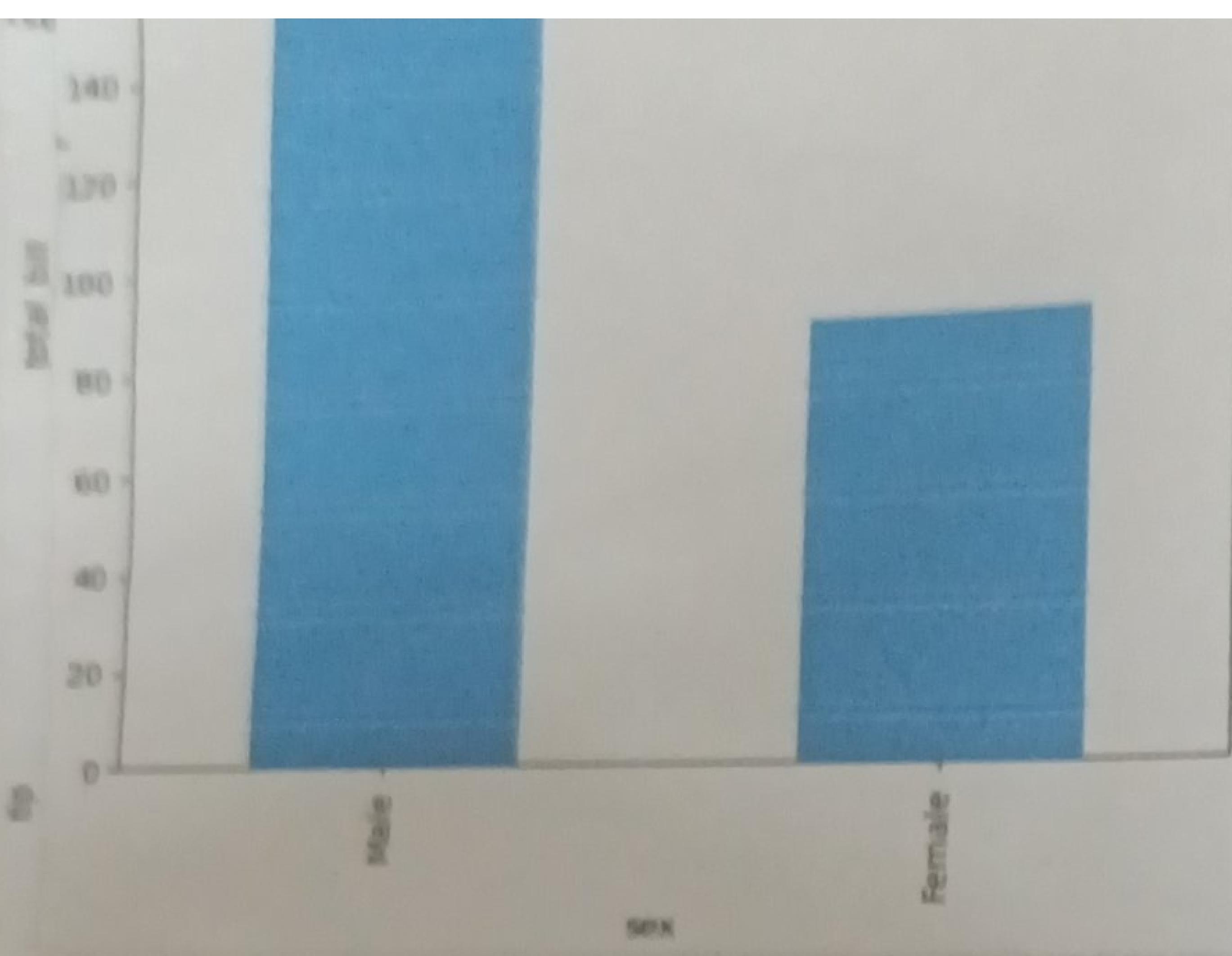


sns.countplot(tips.sex)

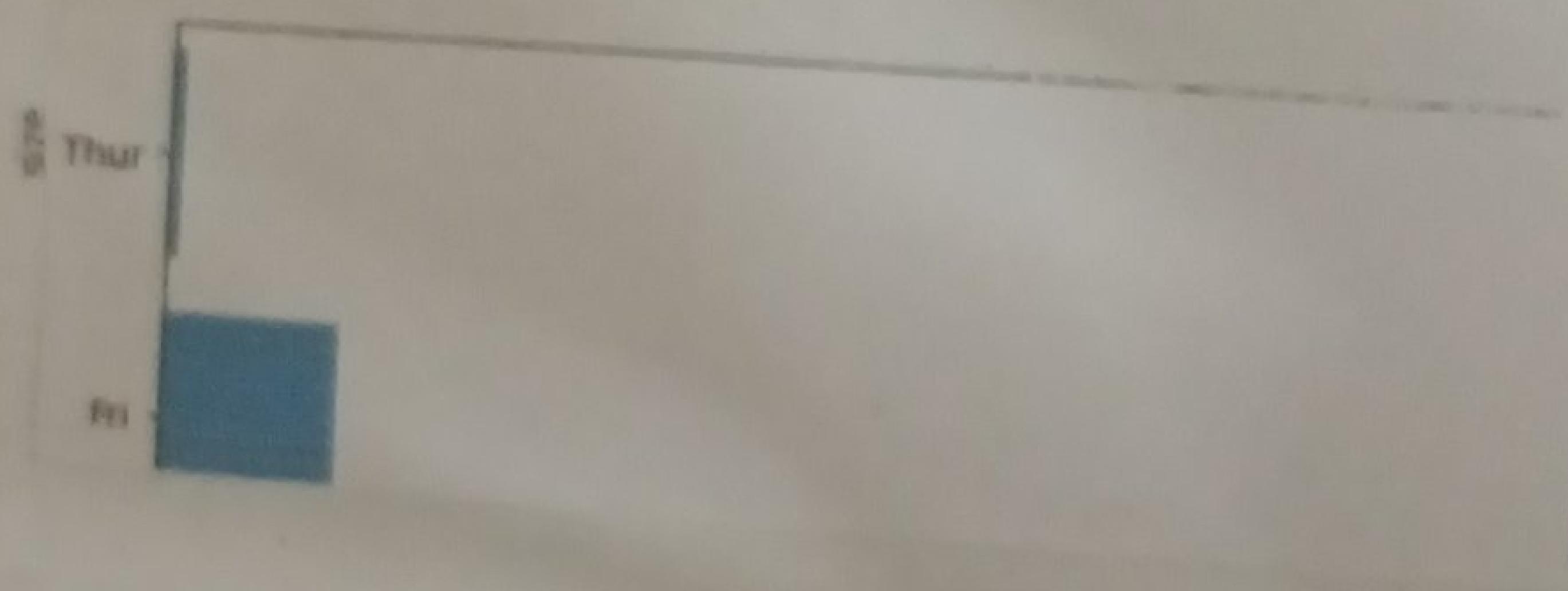


tips.sex.value_counts().plot(kind='pie')





`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.drop('C', axis=1).values
label = df['C'].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=42)

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

LinearRegression()
LinearRegression()

model.score(X_train, y_train)

0.999999999999

model.score(X_test, y_test)

0.999999999999

model.coef_

array([321.38888889])

model.intercept_

array(123333.33333333)

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],  
       ...]]
```

```
print (final Model.Score(x_train,y_train))  
print (final Model.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.neighbors import KNeighborsClassifier
```

```
xtrain,xtest,ytrain,ytest=train_test_split(features,label,test_size=.1)  
model_KNN=KNeighborsClassifier(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.metrics import confusion_matrix  
confusion_matrix(label,model_knn.predict(features))
```

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

```
from sklearn.metrics 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				
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 = stats.ttest_1samp(marks, mu_0)
print ("T-statistic: {} t-stat: .3f)".format(t_stat))
print ("P-value: {} p-value: .4f)".format(p_value))
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 $\sigma = 3$ g. At a 5% significance level, test the claim.

Code:

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

$$Z_{\text{Stat}} = (x_{\text{bar}} - \mu_0) / (\sigma / \sqrt{n})$$

$$p_{\text{value}} = 2 * (1 - \text{norm.cdf}(\text{abs}(Z_{\text{stat}})))$$

```
print ("Z-statistic: " + str(Z_stat))
print ("P-value: " + str(p_value))
```

$$\alpha = 0.05$$

```
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 scipy 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 \rightarrow Means are significantly different

Completed
J.W.
30/10/06