RAJALAKSHMI ENGINEERING COLLEGE

Approved by AICTE | Affiliated to Anna University | Accredited by NAAC

Department of Computer Science and Engineering

CS23334 Fundamentals of Data Science Lab

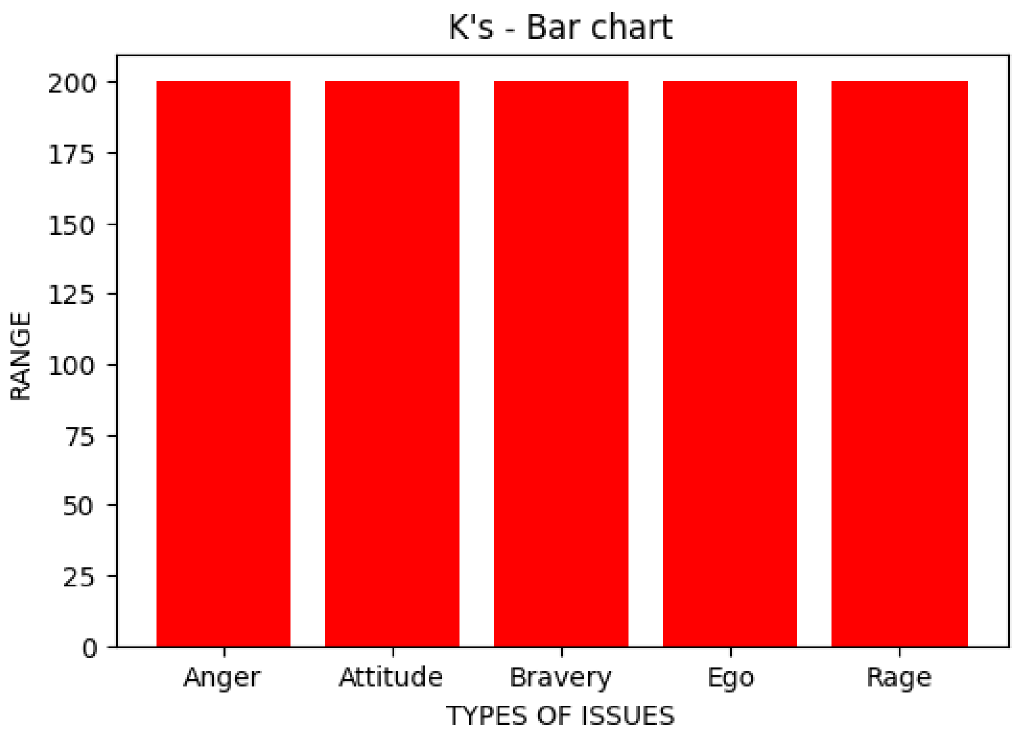
III semester II Year (2023R)

Name of the Student :

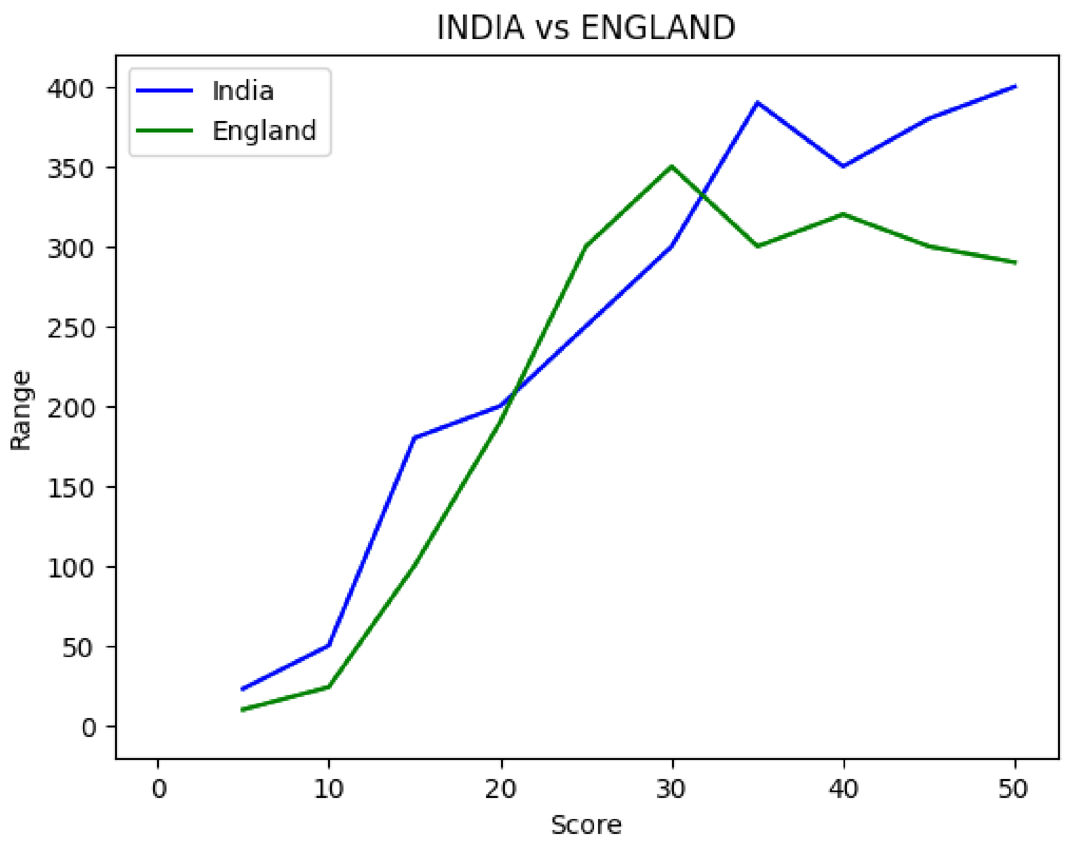
Karthik Subramanian S

Register Number :2116240701233

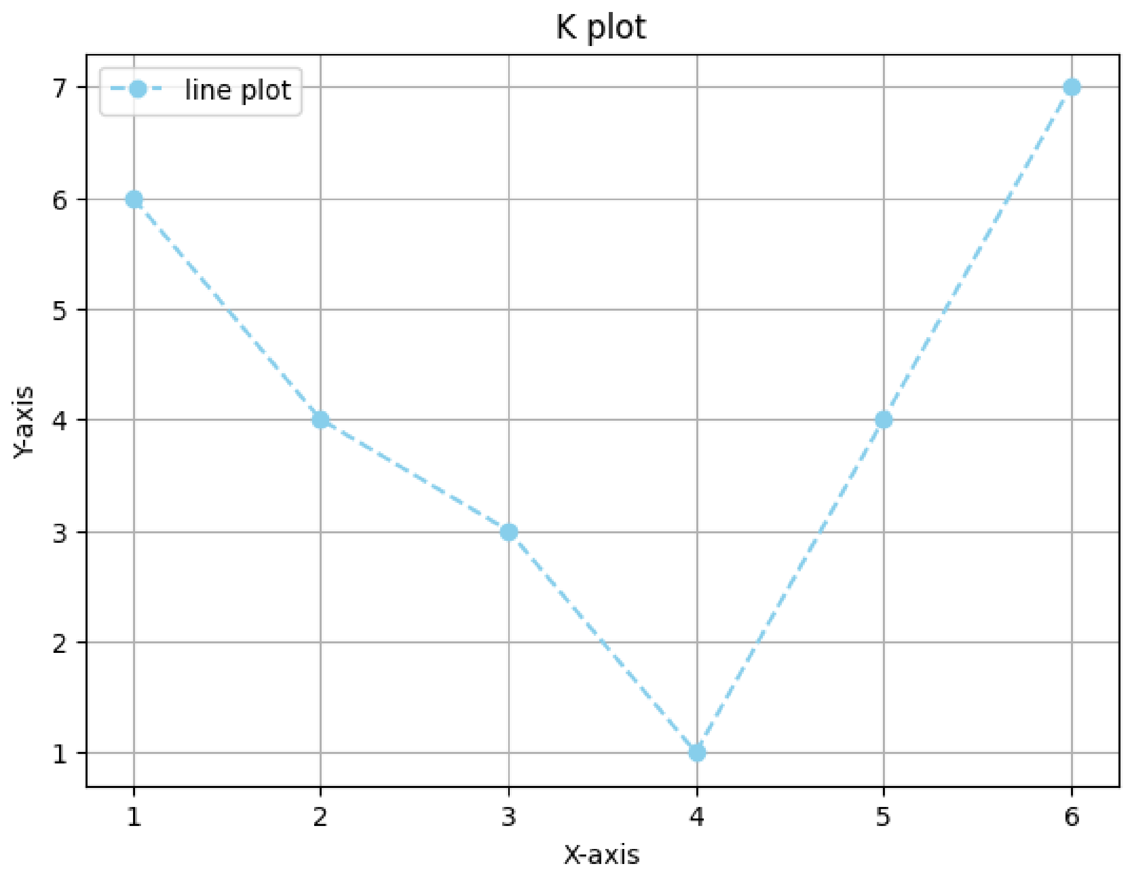
```
// Karthik Subramanian S

//240701233

// 24.07.2025

// Title: BAR CHART


import matplotlib.pyplot as plt

c=['Anger','Attitude','Bravery','Ego','Rage']

a=[200,200,200,200,200]

plt.figure(figsize=(6,4))

plt.bar(c,a,color='red') plt.title("K's - Bar

chart")

plt.xlabel("TYPES OF ISSUES")

plt.ylabel("RANGE") plt.show()
```


K's - Bar chart
RANGE
200
175
150
125
100
75
50
25
0
Anger
Attitude
Bravery
Ego
Rage
TYPES OF ISSUES

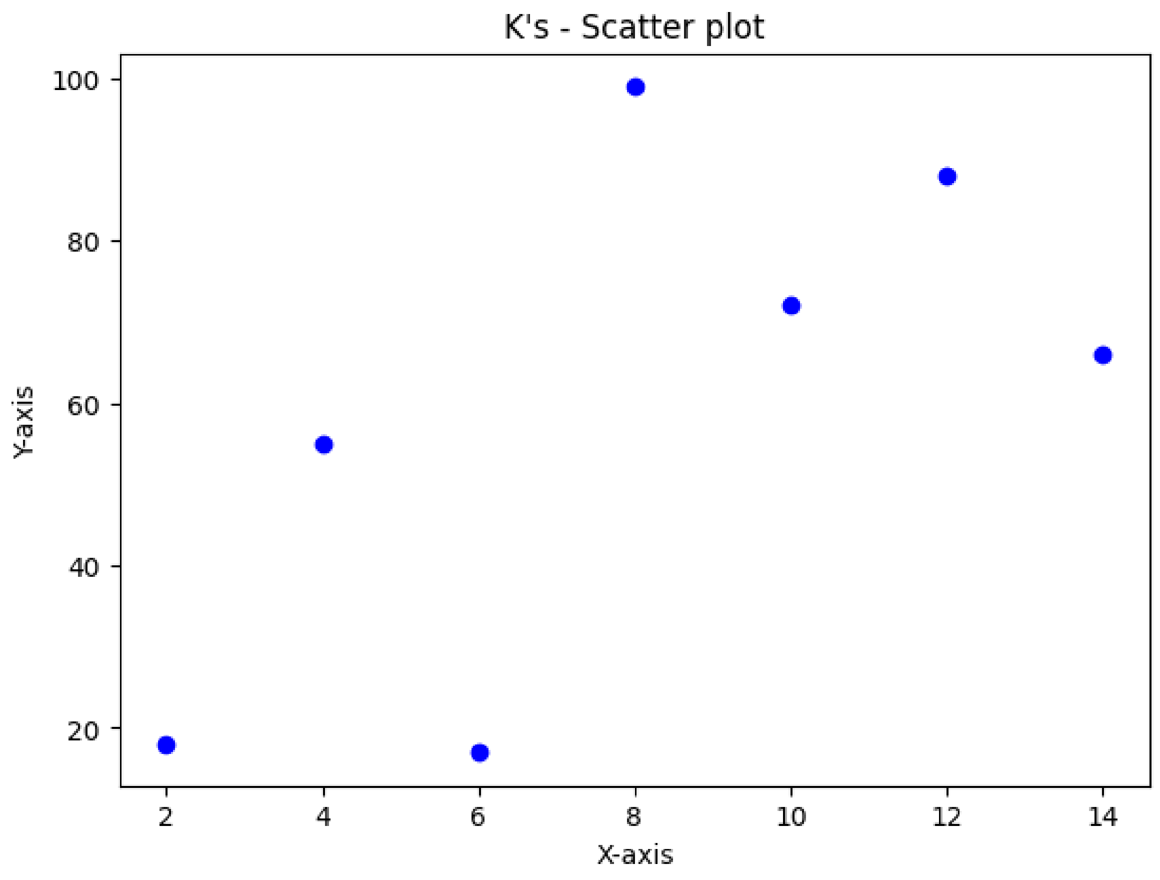
```
// Karthik Subramanian S

// 240701233

//23.07.2025

//    Title :  #Line plot


import matplotlib.pyplot as plt

a = list(range(5,51,5))

i=[23,50,180,200,250,300,390,350,380,400]

E=[10,24,100,190,300,350,300,320,300,290]

plt.plot(a,i,'color'=='blue')

plt.plot(a,E)

plt.title("INDIA vs ENGLAND")

plt.xlabel("Score")

plt.ylabel("Range")

plt.plot(a,i,color="blue",label="India")

plt.plot(a,E,color="green",label="England")

plt.legend()
```


INDIA vs ENGLAND
India
England
Range
Score
0
50
100
150
200
250
300
350
400
0
10
20
30
40
50

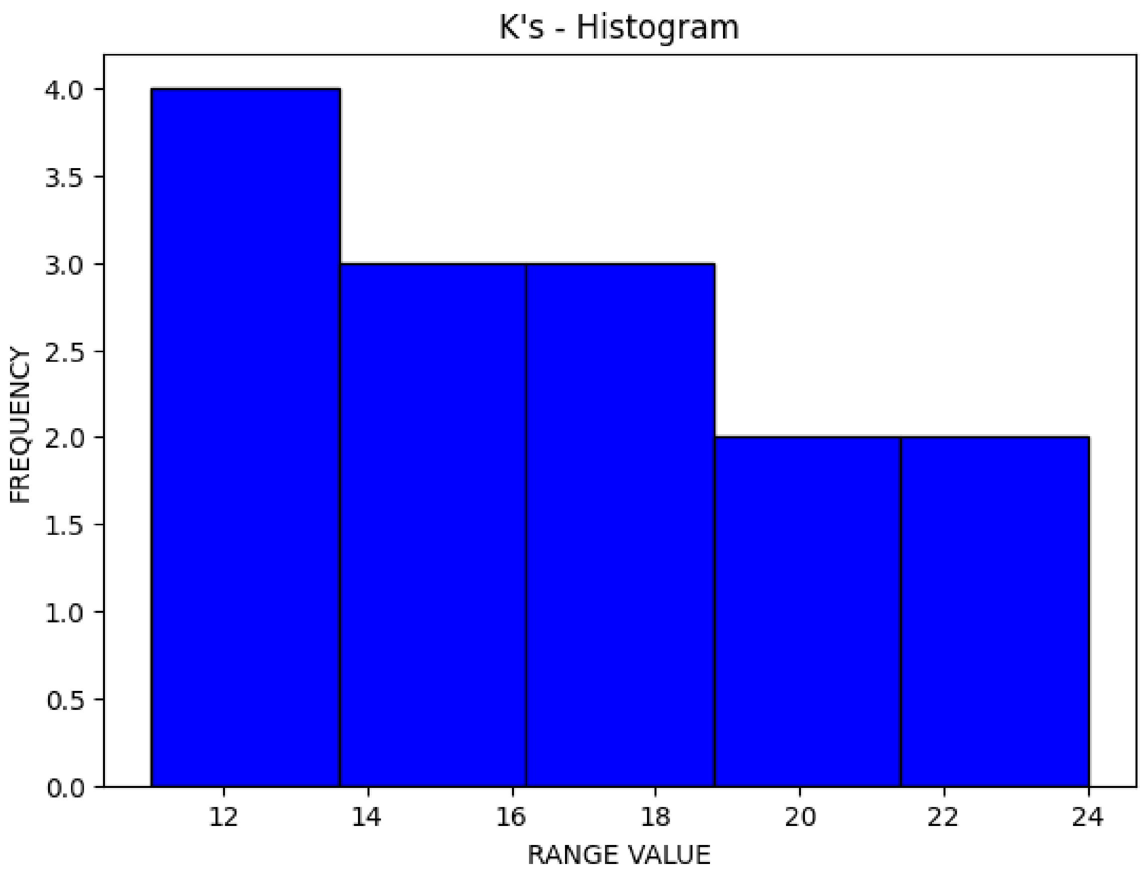
```
//Karthik Subramanian S

//240701233

//24.07.2025

// Title: LINE PLOT

import matplotlib.pyplot as plt

x=[1,2,3,4,5,6]

y=[6,4,3,1,4,7]

plt.figure(figsize=(7,5))

plt.plot(x,y,color="skyblue",linestyle='--',label='line plot',marker='o')

plt.title("K plot")

plt.xlabel("X-axis")

plt.ylabel("Y-axis")

plt.legend()

plt.grid(True)

plt.show()
```


K plot
line plot
Y-axis
X-axis
1
2
3
4
5
6
7

```
// Karthik Subramanian S

//240701233

//24.07.2025

// Title : Scatter plot


import matplotlib.pyplot as plt

x=[2,4,6,8,10,12,14]

y=[18,55,17,99,72,88,66]

plt.figure(figsize=(7,5))

plt.scatter(x,y,color='blue')

plt.title("K's - Scatter plot")

plt.xlabel("X-axis")

plt.ylabel("Y-axis")

plt.show()
```


K's - Scatter plot
Y-axis
X-axis
100
80
60
40
20
2
4
6
8
10
12
14

```
// Karthik Subramanian S

// 240701233

//24.07.2025

// Title : Histogram


import matplotlib.pyplot as plt

a=[12,15,13,17,21,11,20,17,11,14,18,16,22,24]

plt.figure(figsize=(7,5))

plt.hist(a,bins=5,color="blue",edgecolor='black')

plt.title("K's - Histogram")

plt.xlabel("RANGE VALUE")

plt.ylabel("FREQUENCY")

plt.show()
```


K's - Histogram
FREQUENCY
RANGE VALUE
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
12
14
16
18
20
22
24

```
//Karthik Subramanian S
//240701233
//19.08.2025
//Title: " Feature Scaling"

import numpy as np
import pandas as pd
df=pd.read_csv('pre_process_datasample.csv')
df
df.head()
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]])
Salary.fit(features[:,[2]])
SimpleImputer()
features[:,[1]]=age.transform(features[:,[1]])
features[:,[2]]=Salary.transform(features[:,[2]])
features
from sklearn.preprocessing import OneHotEncoder
oh = OneHotEncoder(sparse_output=False)
Country=oh.fit_transform(features[:,[0]])
Country
final_set=np.concatenate((Country,features[:,[1,2]]),axis=1)
final_set
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
sc.fit(final_set)
```

```
feat_standard_scaler=sc.transform(final_set)
feat_standard_scaler
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
```

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

	Country	Age	Salary	Purchased

	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

```
SimpleImputer ?i
SimpleImputer()
```

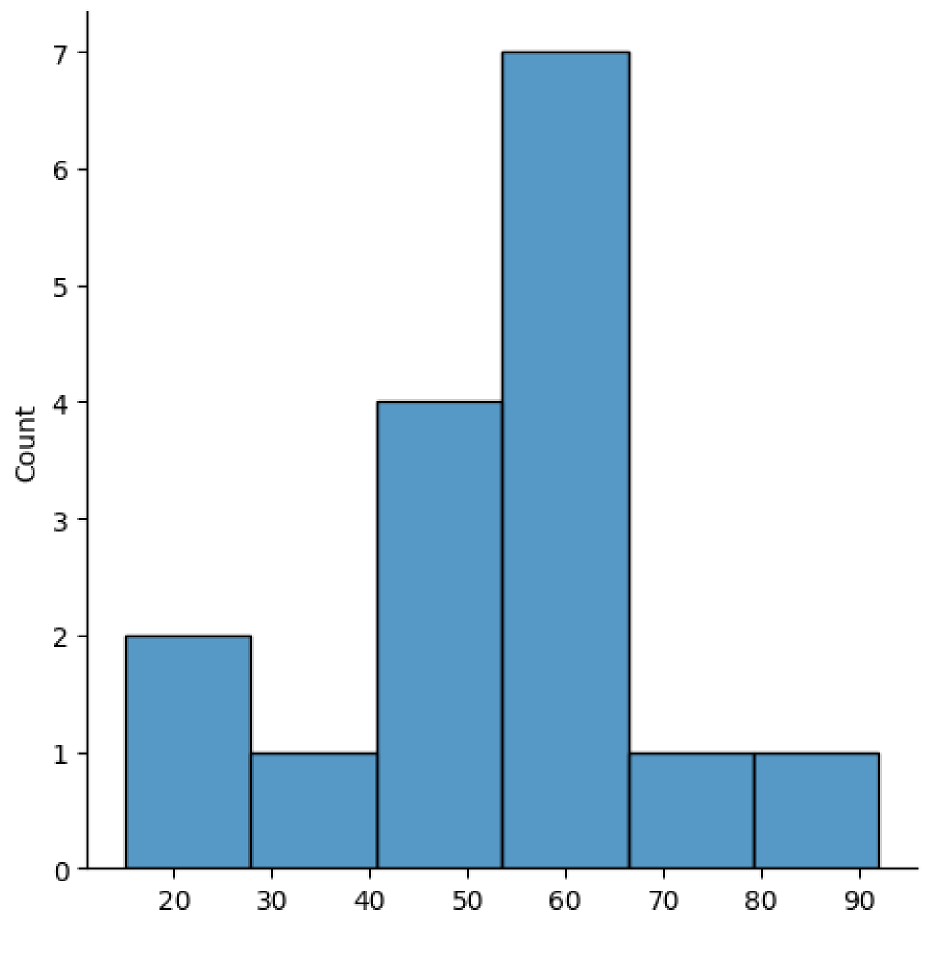
```
array([['France', 44.0, 72000.0],
       ['Spain', 27.0, 48000.0],
       ['Germany', 30.0, 54000.0],
       ['Spain', 38.0, 61000.0],
       ['Germany', 40.0, 63777.77777777778],
       ['France', 35.0, 58000.0],
       ['Spain', 38.77777777777778, 52000.0],
       ['France', 48.0, 79000.0],
  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

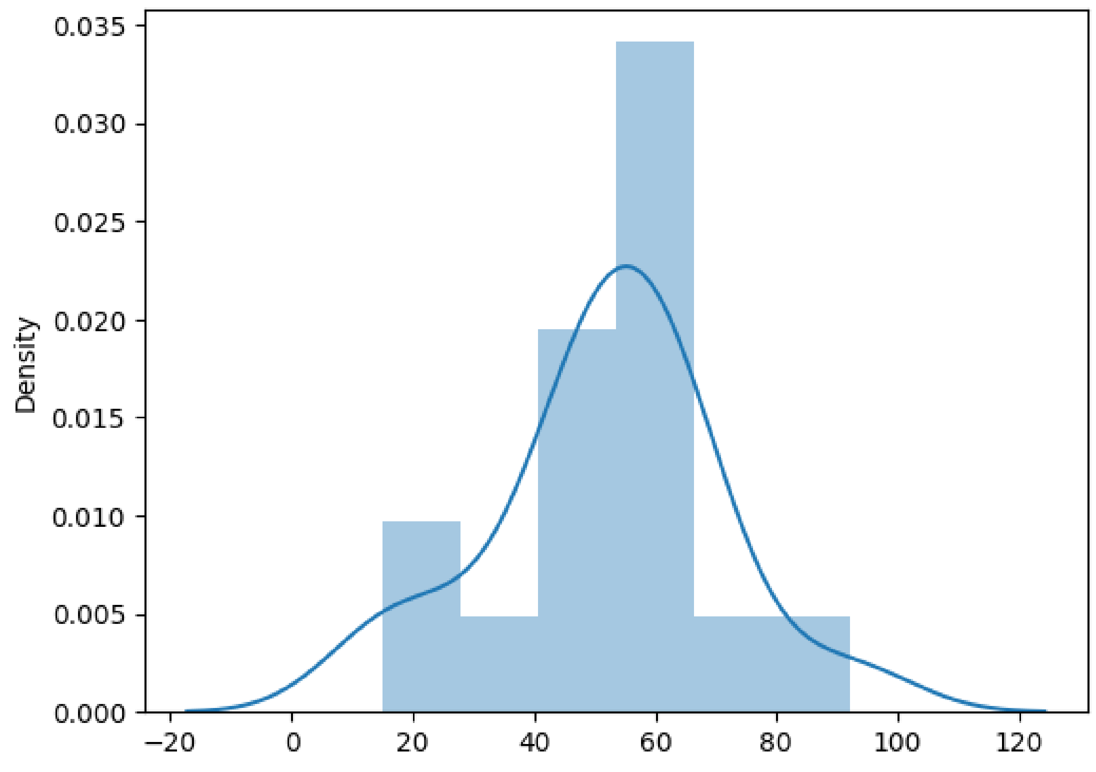
```
array([[1., 0., 0.],
      [0., 0., 1.],
       [0., 1., 0.],
       [0., 0., 1.],
       [0., 1., 0.],
       [1., 0., 0.],
       [0., 0., 1.],
       [1., 0., 0.],
       [0., 1., 0.],
       [1., 0., 0.]])

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],
       [0.0, 1.0, 0.0, 50.0, 83000.0],
       [1.0, 0.0, 0.0, 37.0, 67000.0]], dtype=object)

array([[ 1.22474487e+00, -6.54653671e-01, -6.54653671e-01,
         7.58874362e-01,  7.49473254e-01],
       [-8.16496581e-01, -6.54653671e-01,  1.52752523e+00,
        -1.71150388e+00, -1.43817841e+00],
       [-8.16496581e-01,  1.52752523e+00, -6.54653671e-01,
        -1.27555478e+00, -8.91265492e-01],
       [-8.16496581e-01, -6.54653671e-01,  1.52752523e+00,
        -1.13023841e-01, -2.53200424e-01],
       [-8.16496581e-01,  1.52752523e+00, -6.54653671e-01,
         1.77608893e-01,  6.63219199e-16],
       [ 1.22474487e+00, -6.54653671e-01, -6.54653671e-01,
        -5.48972942e-01, -5.26656882e-01],
       [-8.16496581e-01, -6.54653671e-01,  1.52752523e+00,
         0.00000000e+00, -1.07356980e+00],
       [ 1.22474487e+00, -6.54653671e-01, -6.54653671e-01,
         1.34013983e+00,  1.38753832e+00],
       [-8.16496581e-01,  1.52752523e+00, -6.54653671e-01,
         1.63077256e+00,  1.75214693e+00],
       [ 1.22474487e+00, -6.54653671e-01, -6.54653671e-01,
        -2.58340208e-01,  2.93712492e-01]])


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],
       [0.        , 1.        , 0.        , 1.        , 1.        ],
       [1.        , 0.        , 0.        , 0.43478261, 0.54285714]])
```

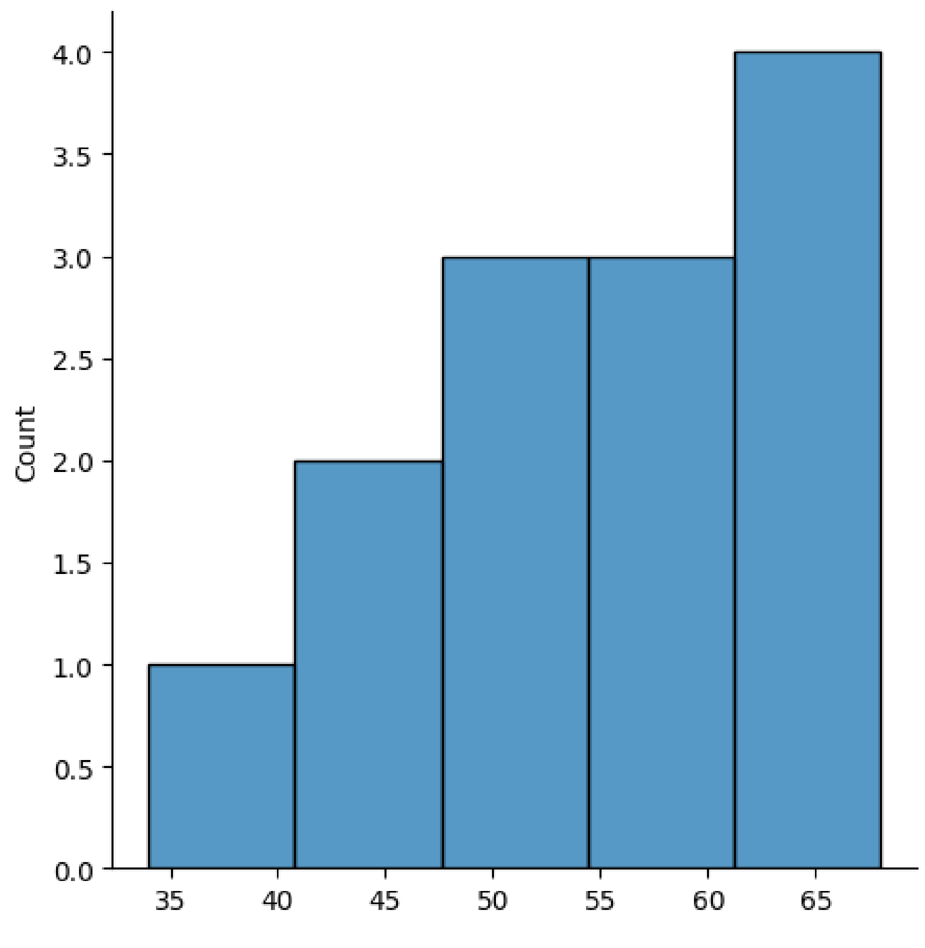
```
//Karthik Subramanian S
//240701233
//19.08.2025
//Title : “Outliers Detection”

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)
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)
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)
```

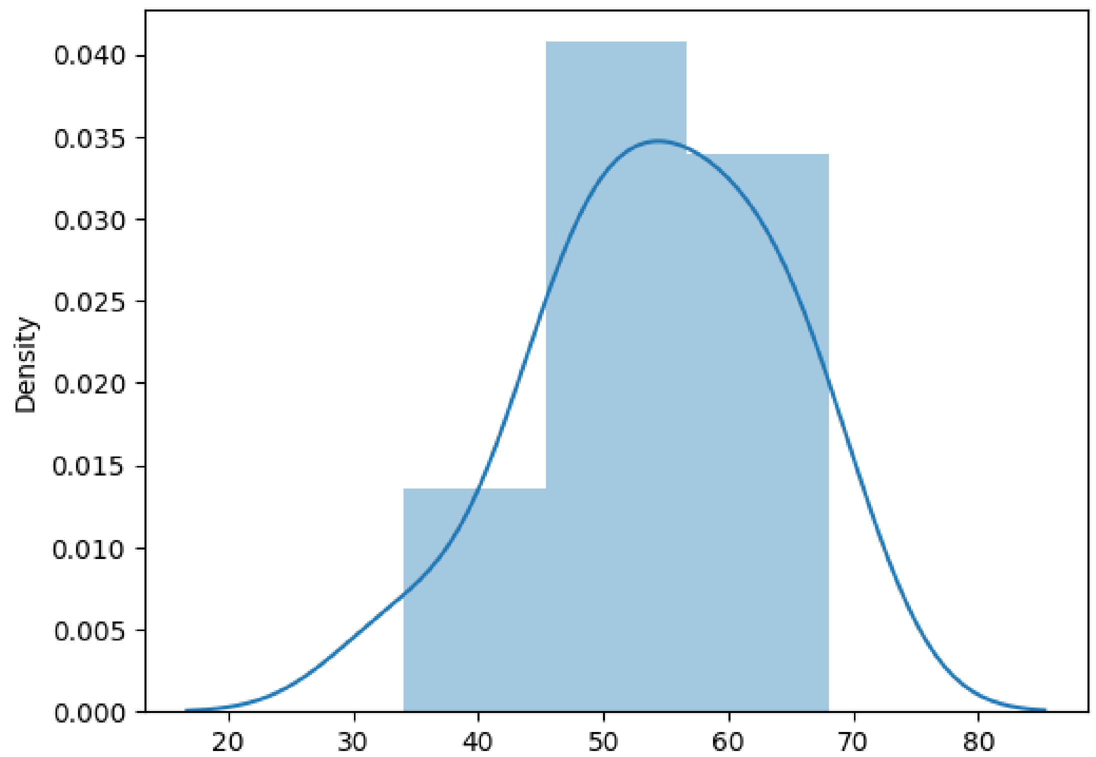
```
array([49, 55, 19, 48, 56, 68, 47, 63, 66, 92, 63, 15, 34, 45, 54, 58])

np.float64(52.0)

np.float64(46.5)

np.float64(54.5)

np.float64(63.0)

np.float64(92.0)

(np.float64(21.75), np.float64(87.75))

array([49, 55, 48, 56, 68, 47, 63, 66, 63, 34, 45, 54, 58])

(np.float64(25.5), np.float64(85.5))

array([49, 55, 48, 56, 68, 47, 63, 66, 63, 34, 45, 54, 58])
```


Count
0
1
2
3
4
5
6
7
20
30
40
50
60
70
80
90


Density
0.000
0.005
0.010
0.015
0.020
0.025
0.030
0.035
−20
0
20
40
60
80
100
120



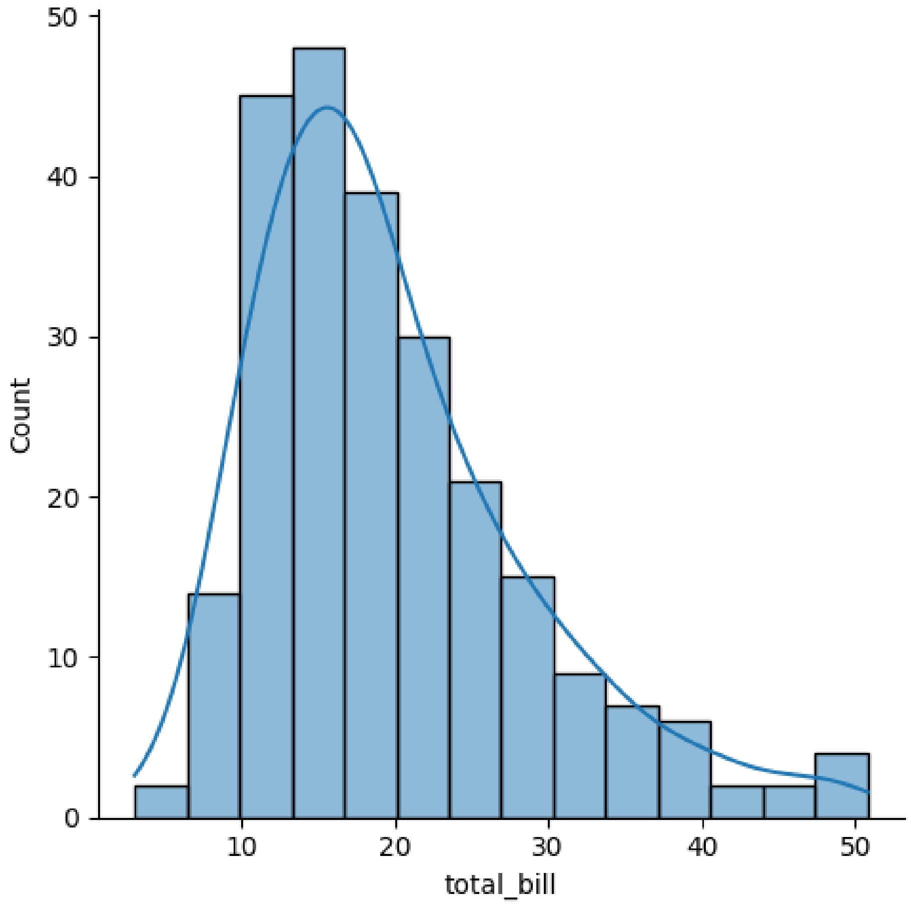
Count
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
35
40
45
50
55
60
65

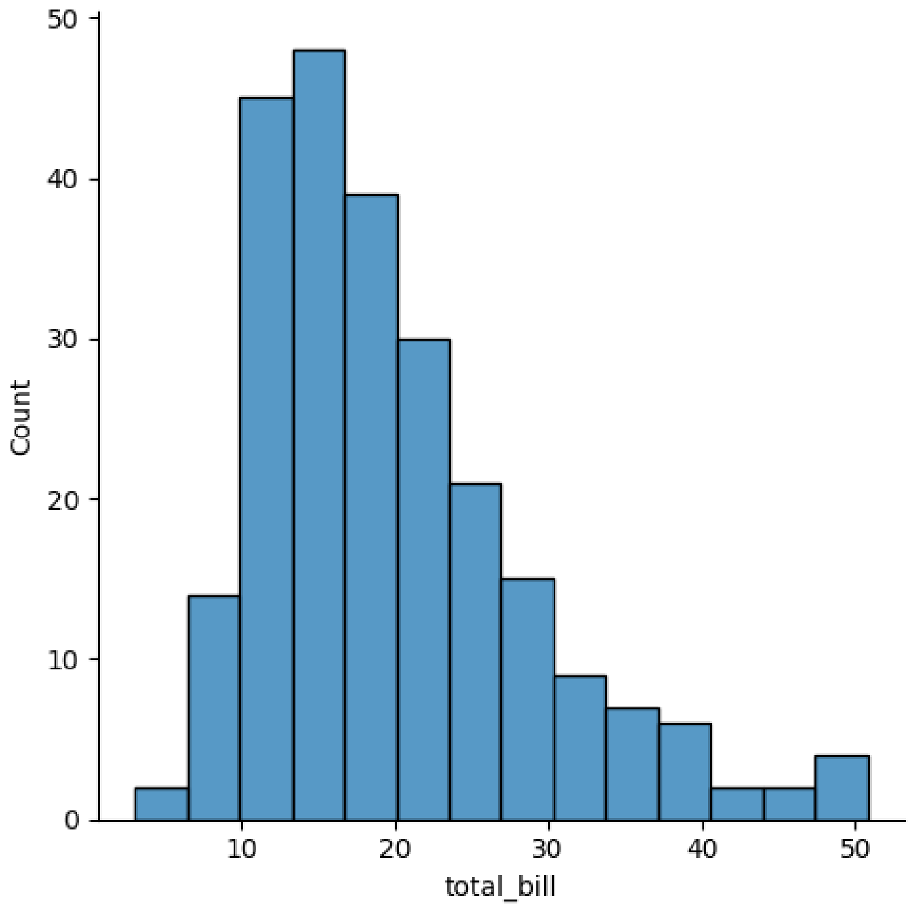


Density
0.000
0.005
0.010
0.015
0.020
0.025
0.030
0.035
0.040
20
30
40
50
60
70
80

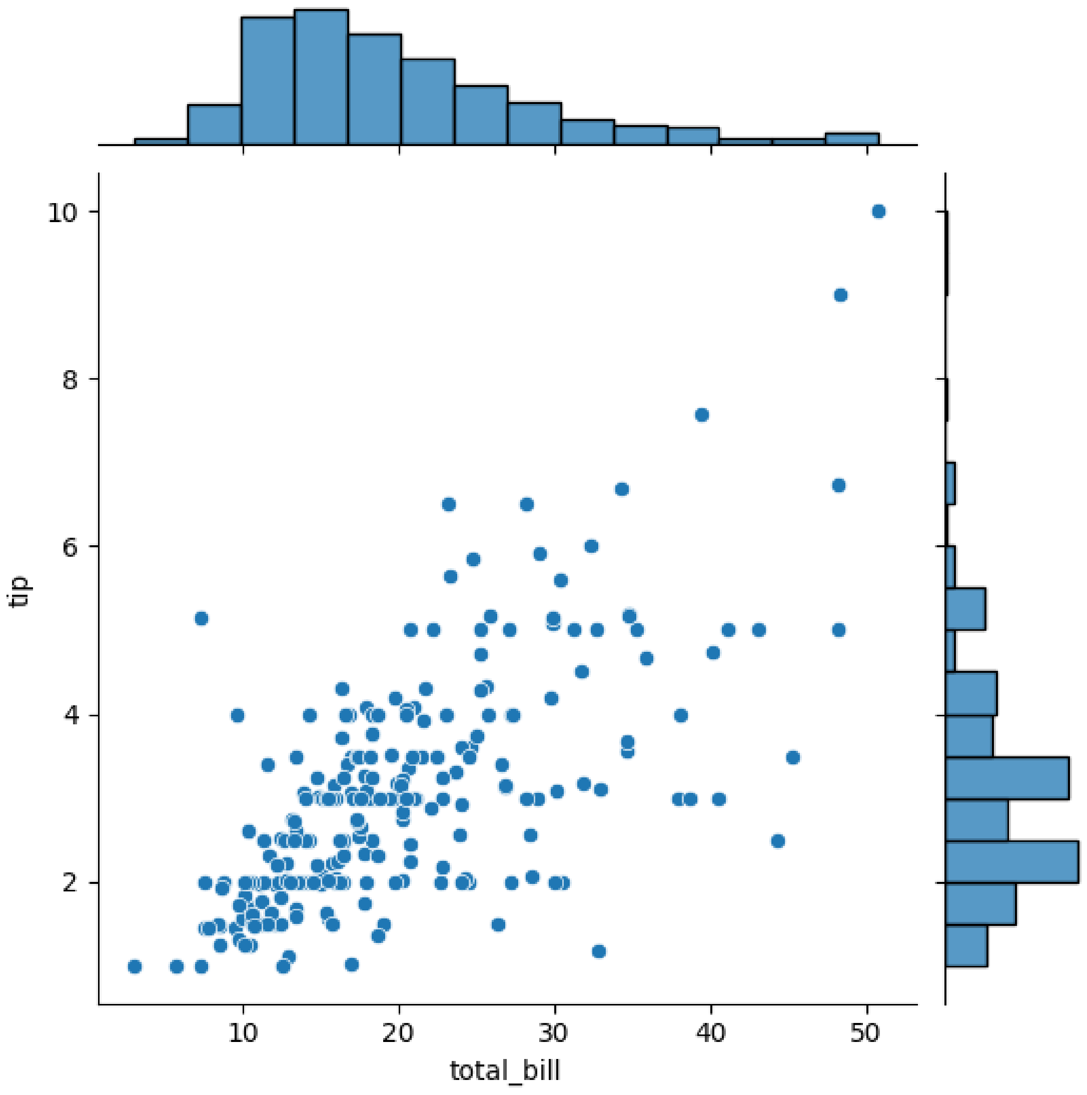
```
//Karthik Subramanian S

//240701233

//28.08.2025

//Title : ”Exploratory Data Analysis”


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()

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

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

sns.jointplot(x=tips.total_bill,y=tips.tip)

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

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

sns.pairplot(tips)

tips.time.value_counts()

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')

tips.sex.value_counts().plot(kind='bar')
```

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

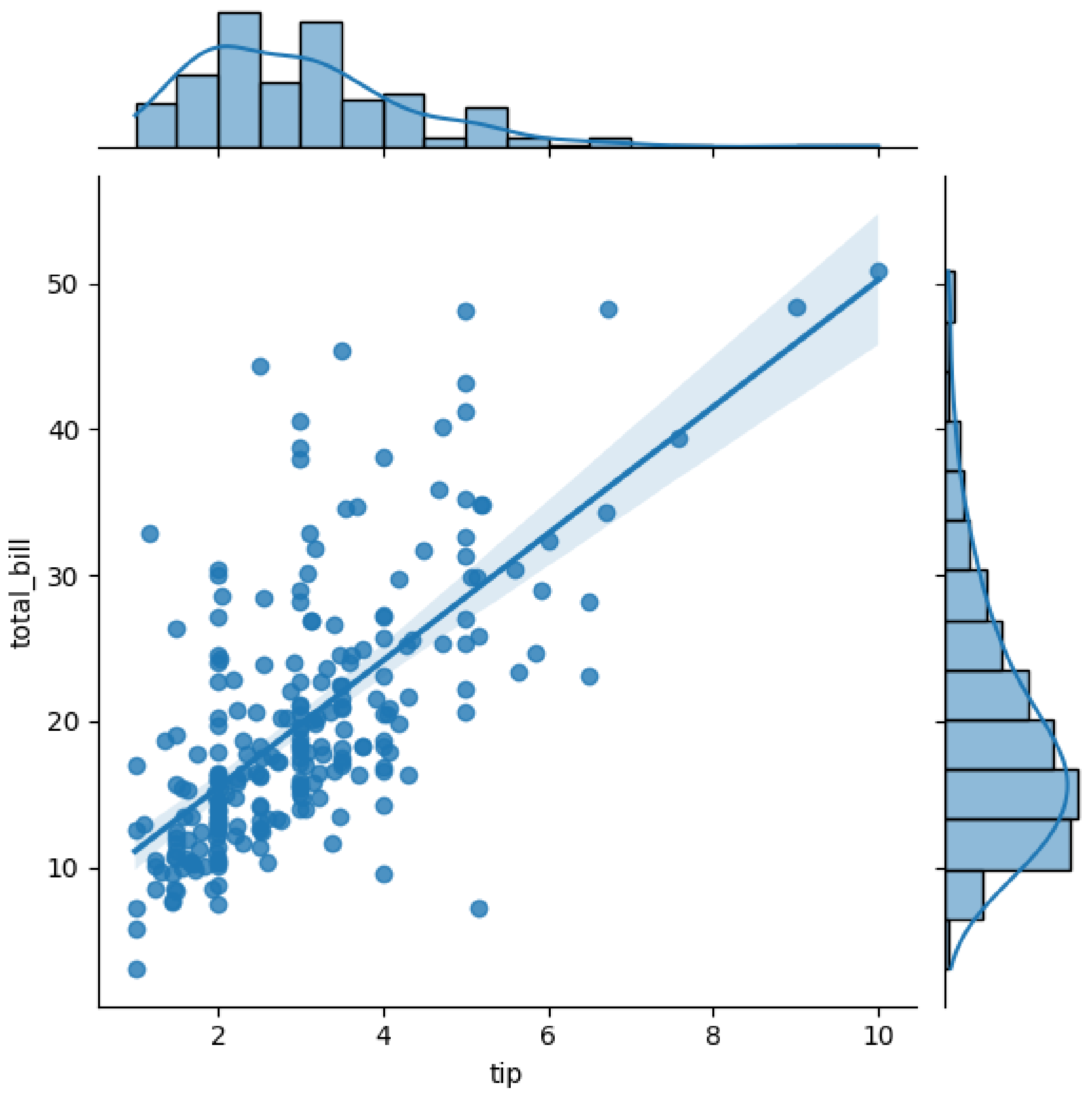




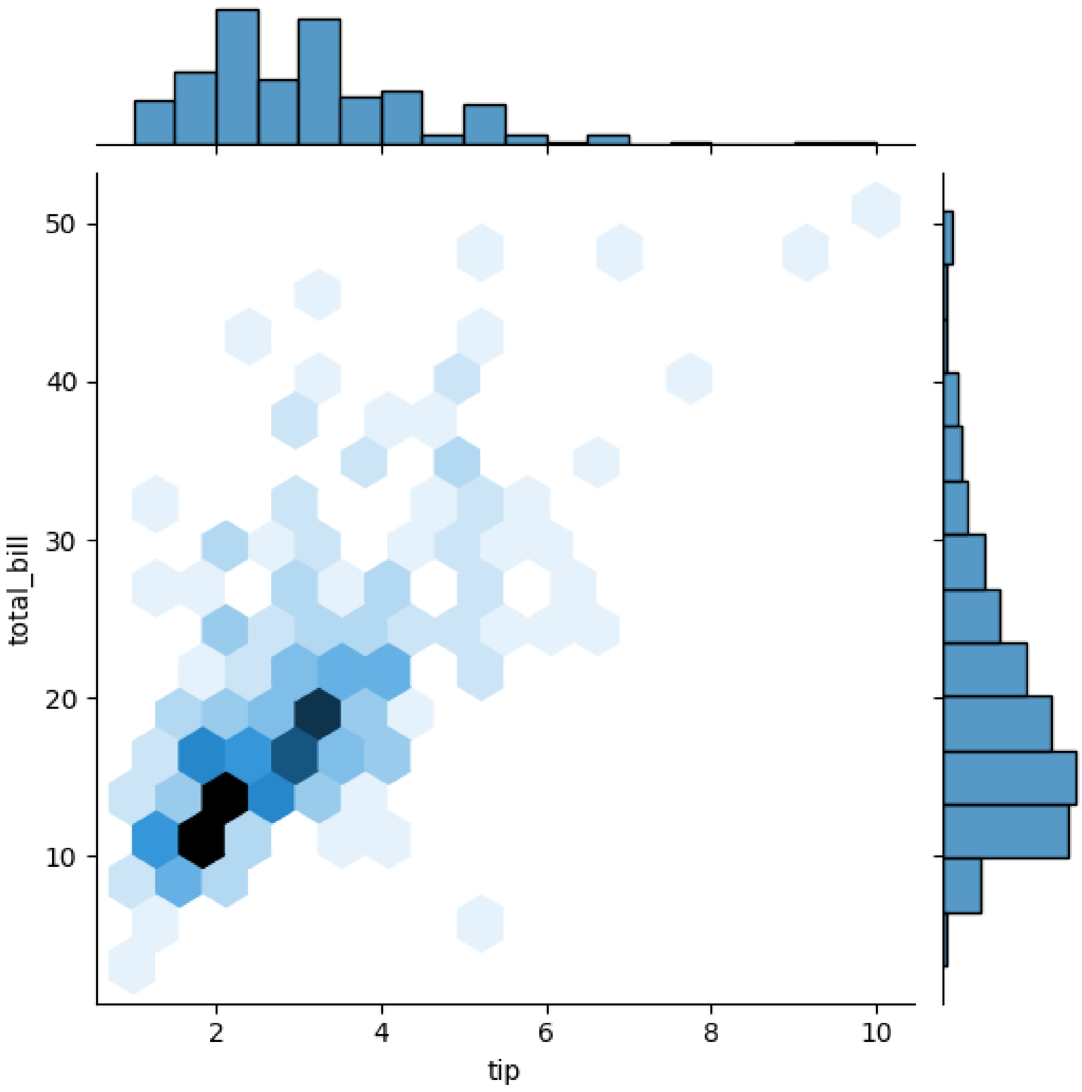
50
40
30
20
10
0
Count
10
20
30
40
50
total_bill



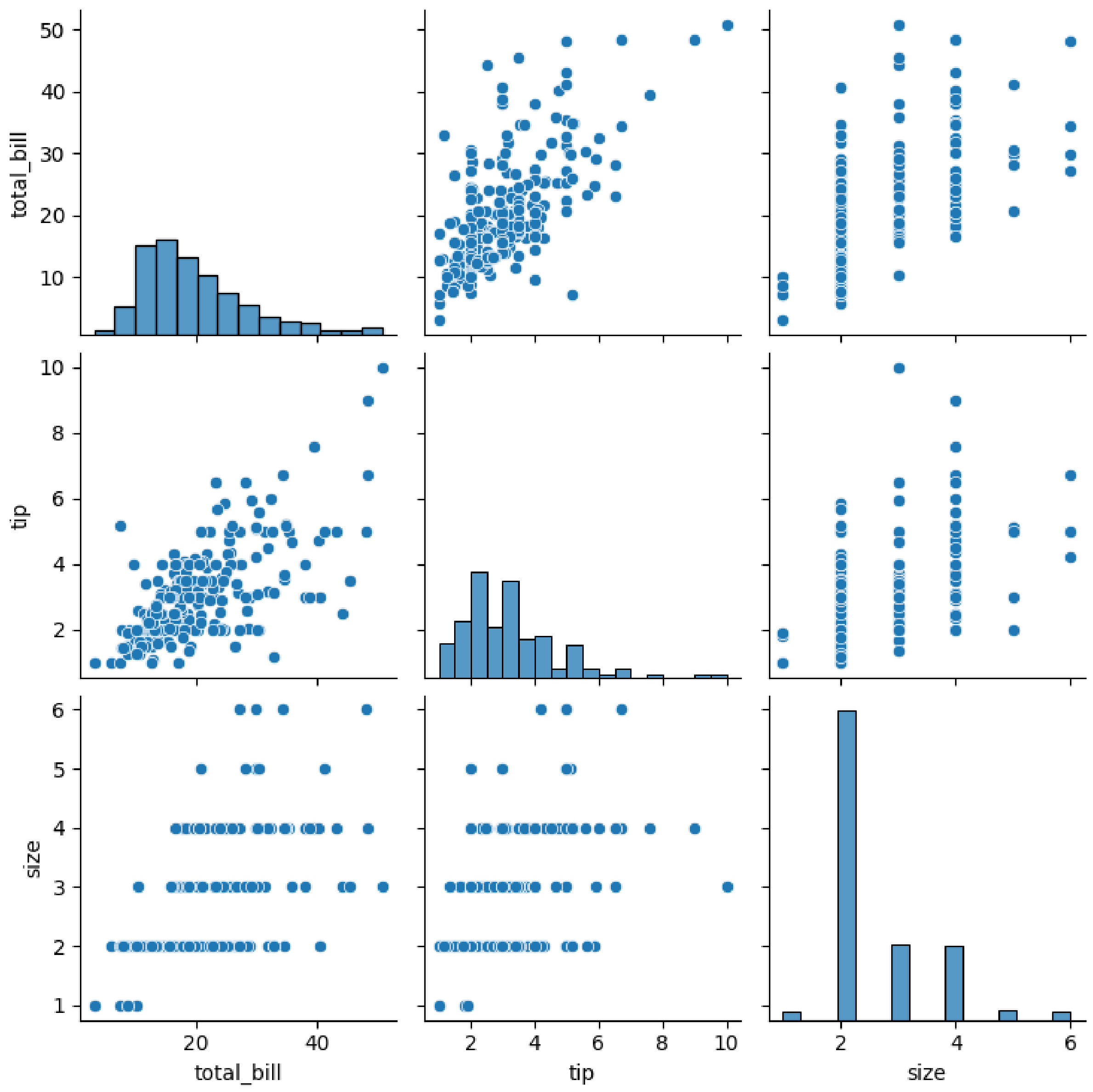
10
8
6
4
2
tip
10
20
30
40
50
total_bill

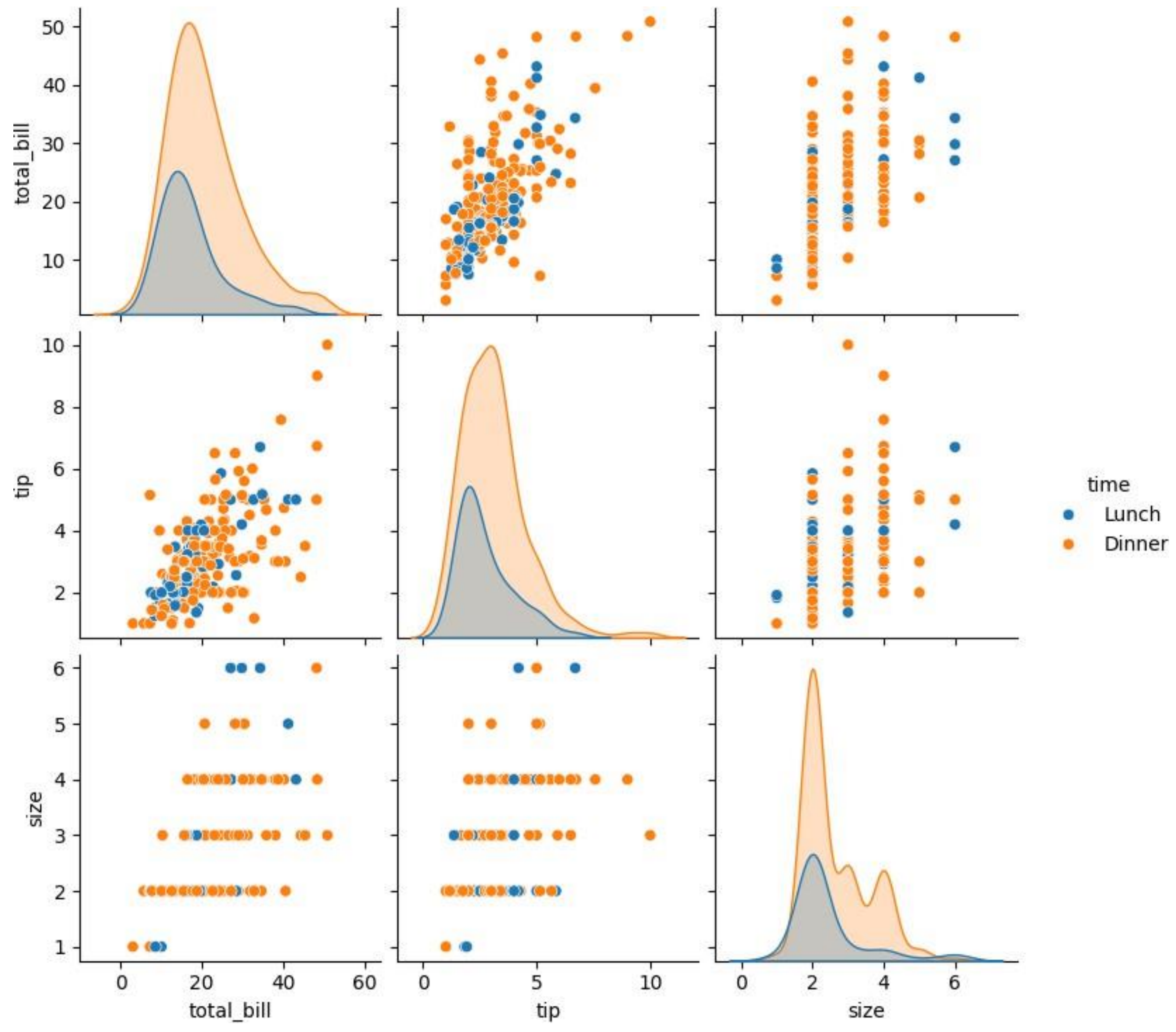


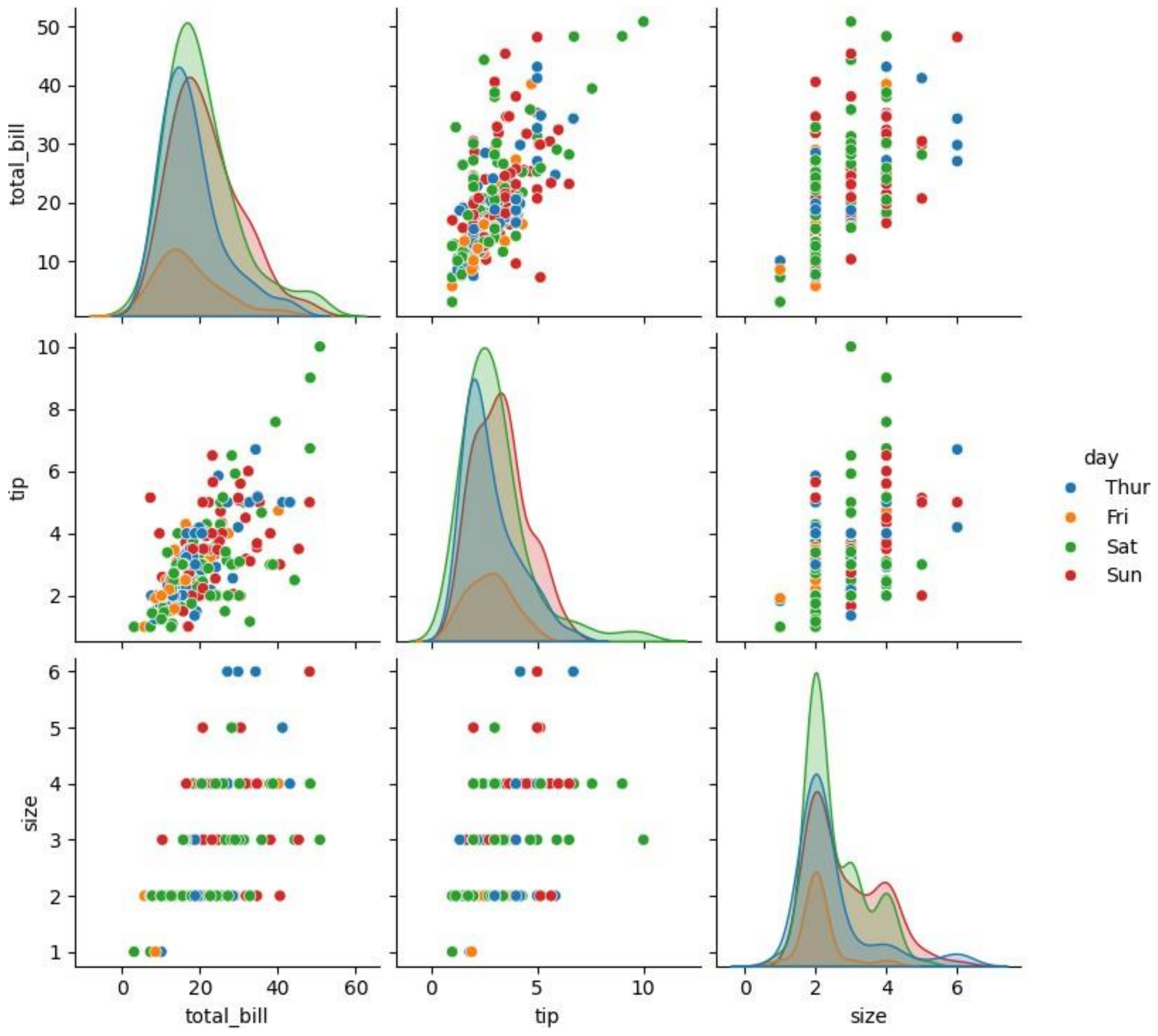
total_bill
tip
10
20
30
40
50
2
4
6
8
10

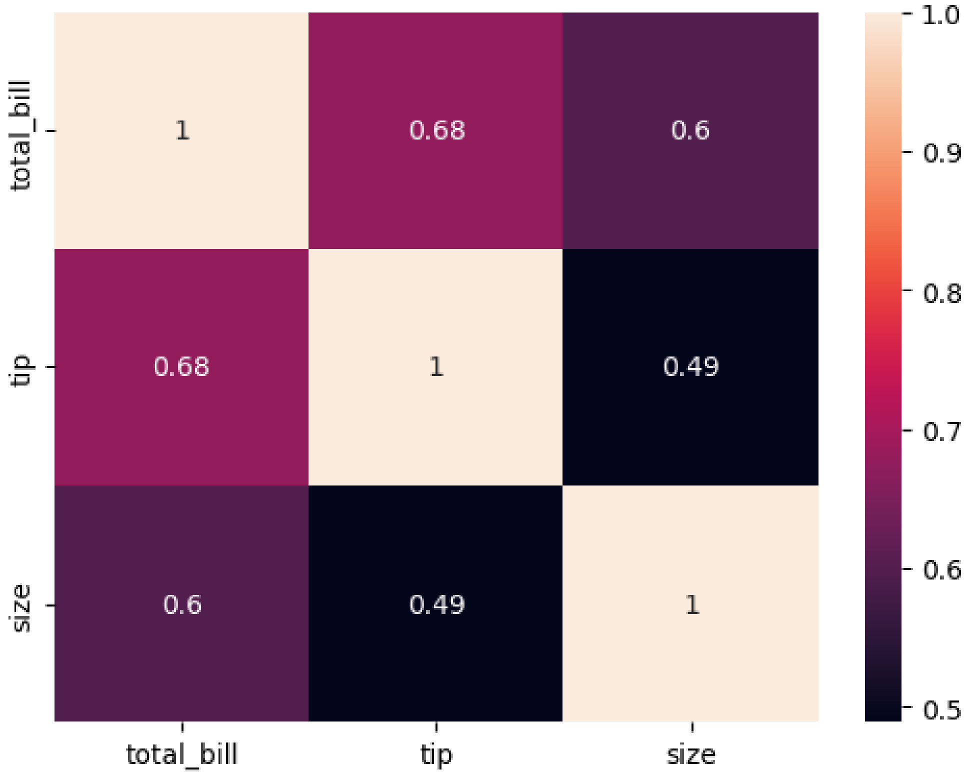


total_bill
tip
50
40
30
20
10
2
4
6
8
10

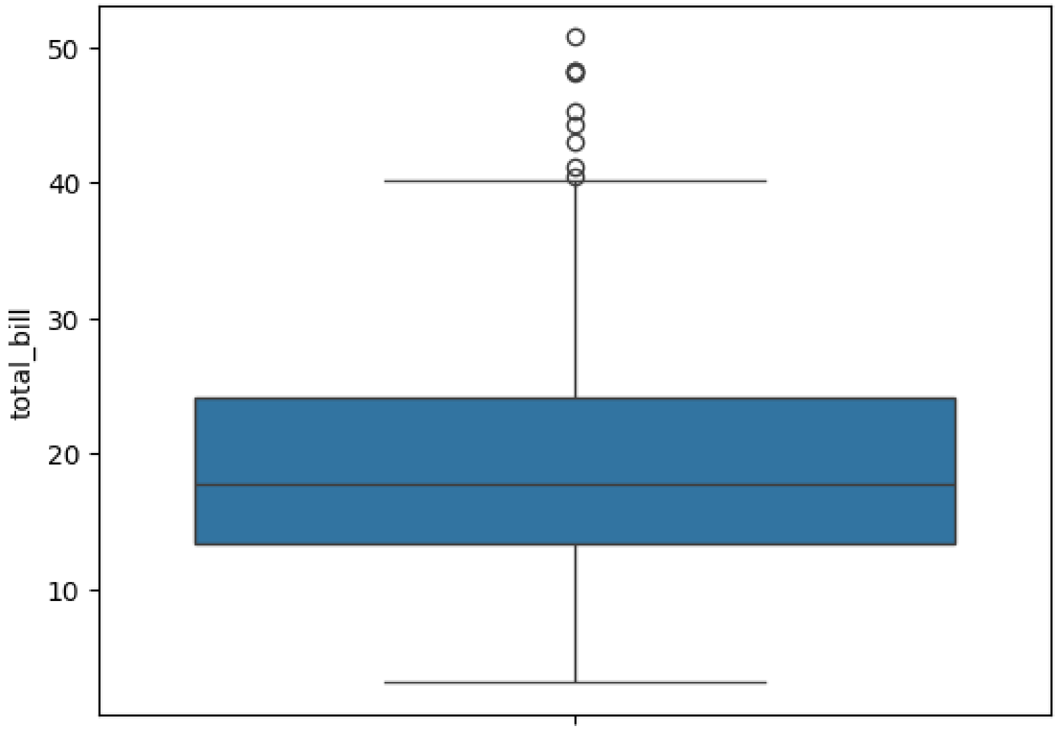

total_bill
tip
size
total_bill
tip
size


total_bill
tip
size
time
Lunch
Dinner

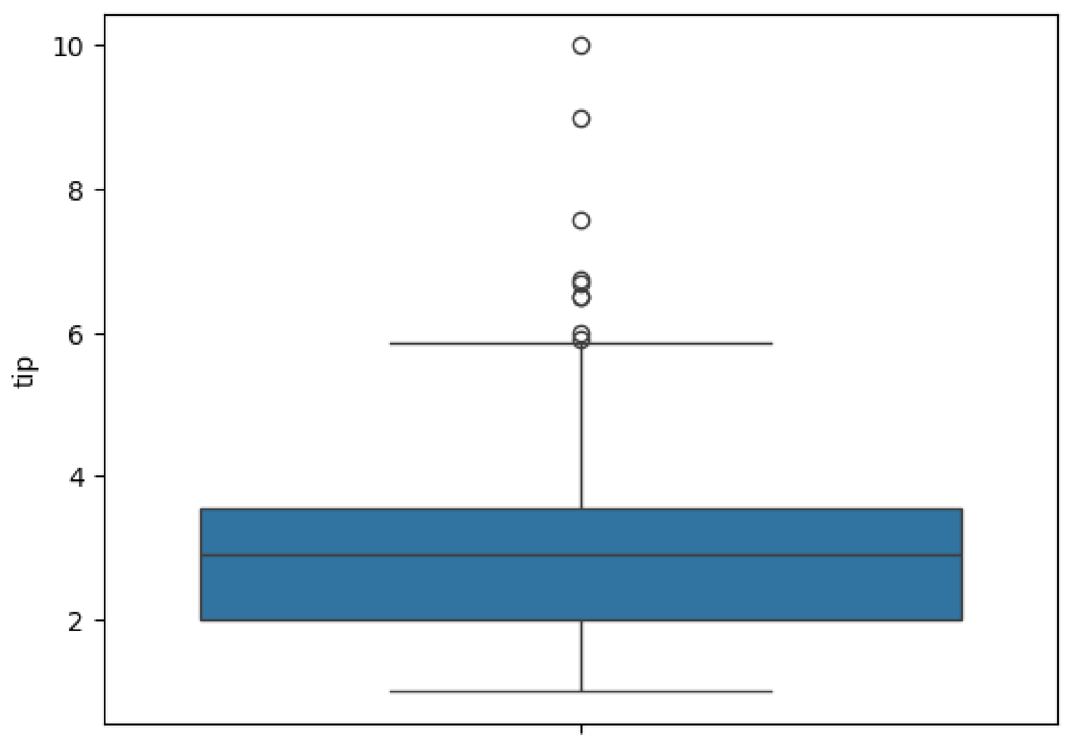

total_bill
tip
size
day
Thur
Fri
Sat
Sun
0
20
40
60
5
10
2
4
6
1
3
8
30
50

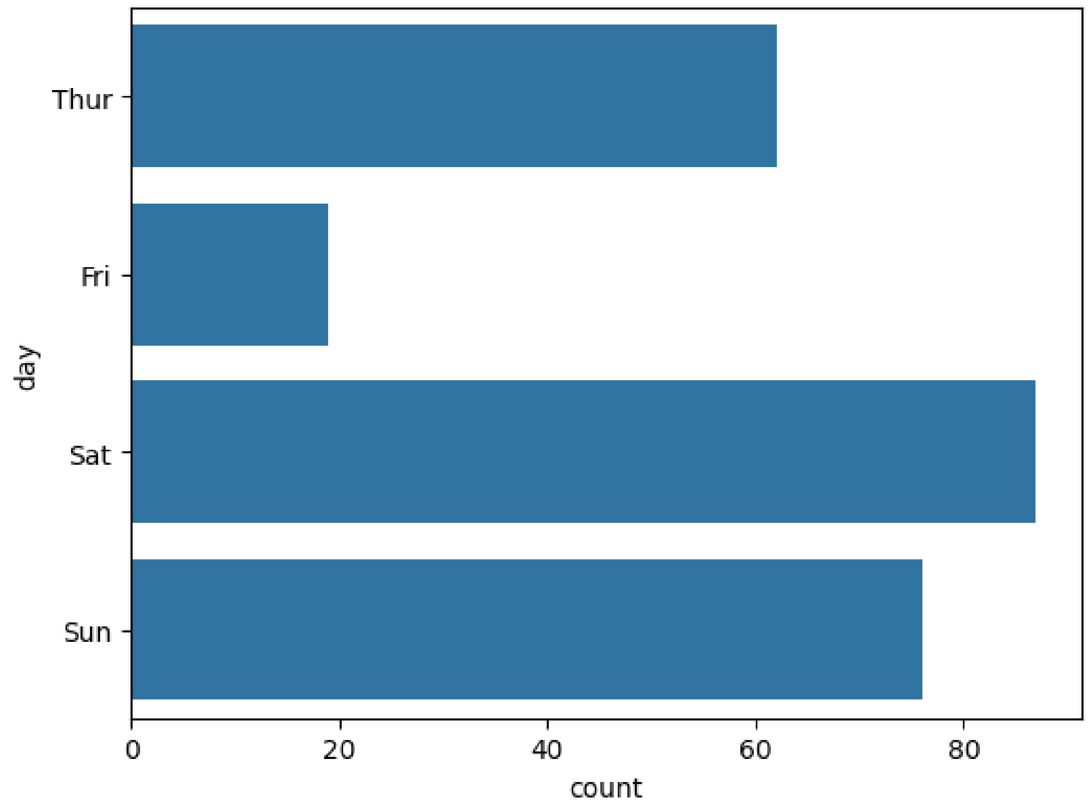


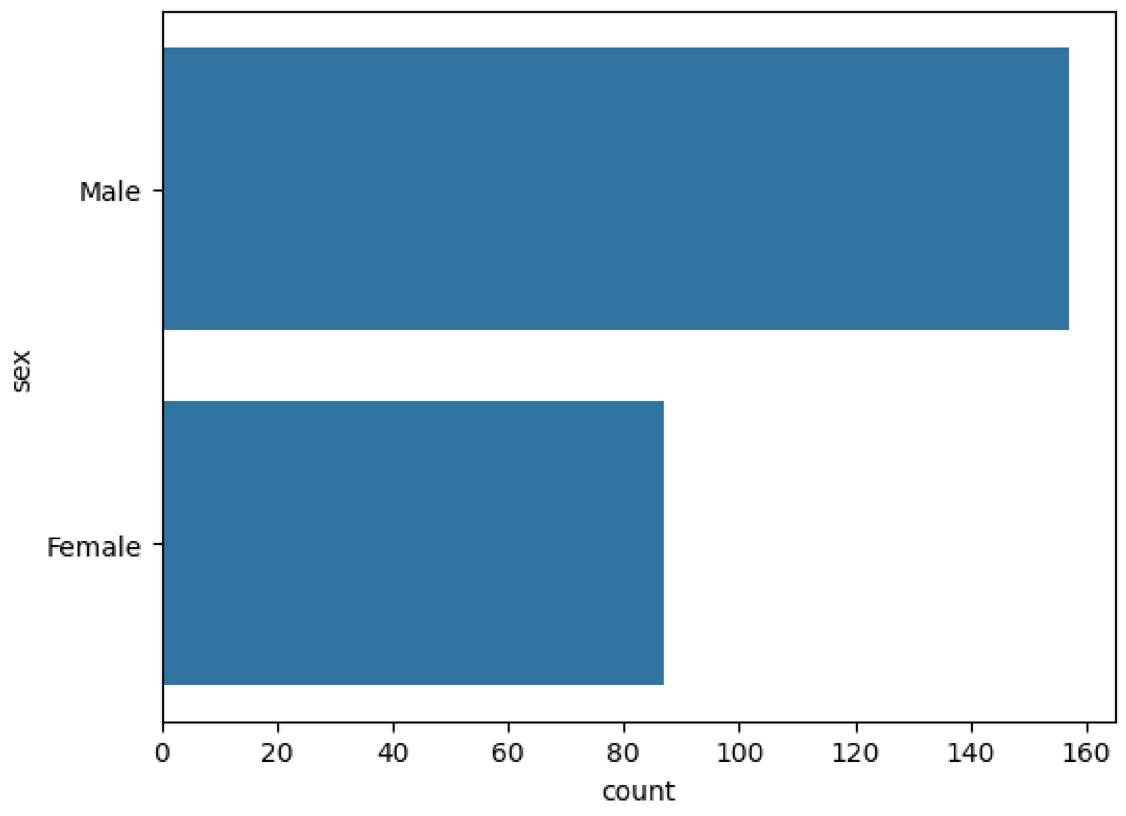
total_bill
tip
size
1
0.68
0.6
0.68
1
0.49
0.6
0.49
1
total_bill
tip
size
1.0
0.9
0.8
0.7
0.6
0.5



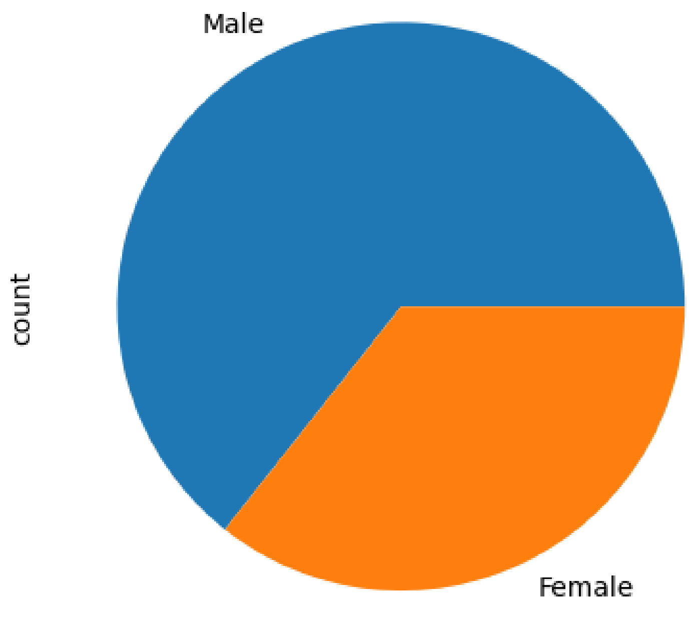
50
40
30
20
10
total_bill


10
8
6
4
2
tip

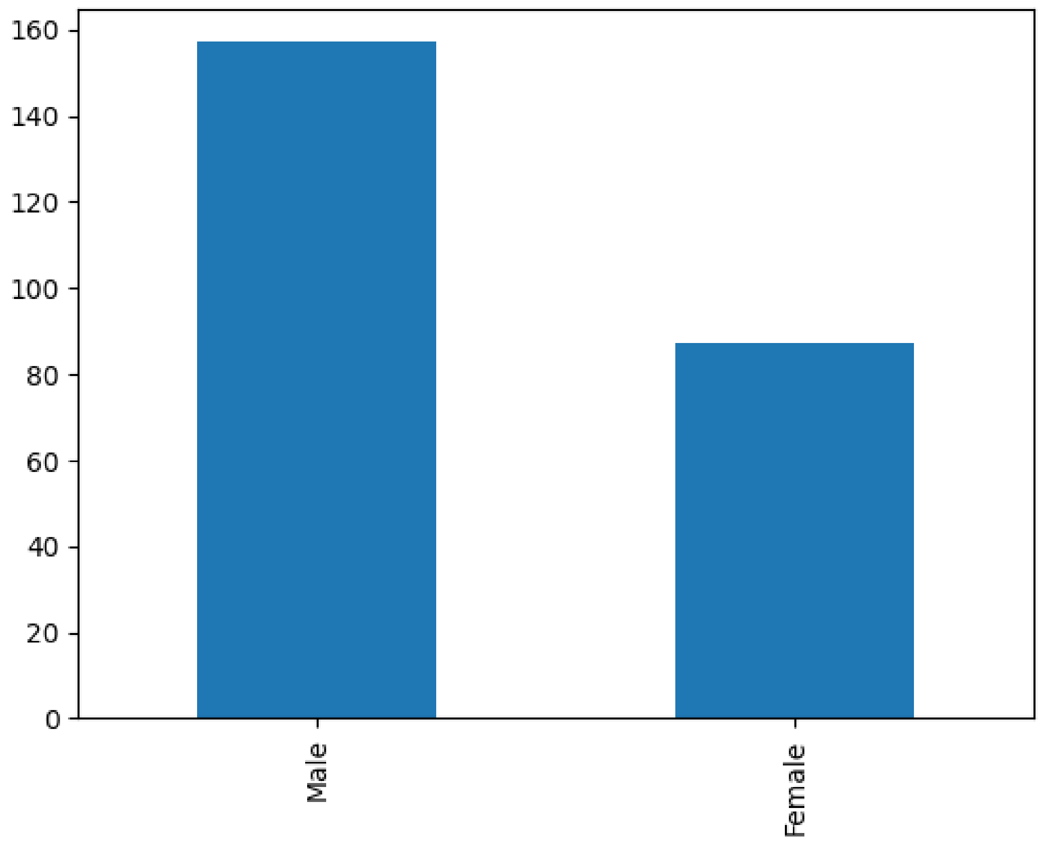

Thur
Fri
Sat
Sun
day
0
20
40
60
80
count



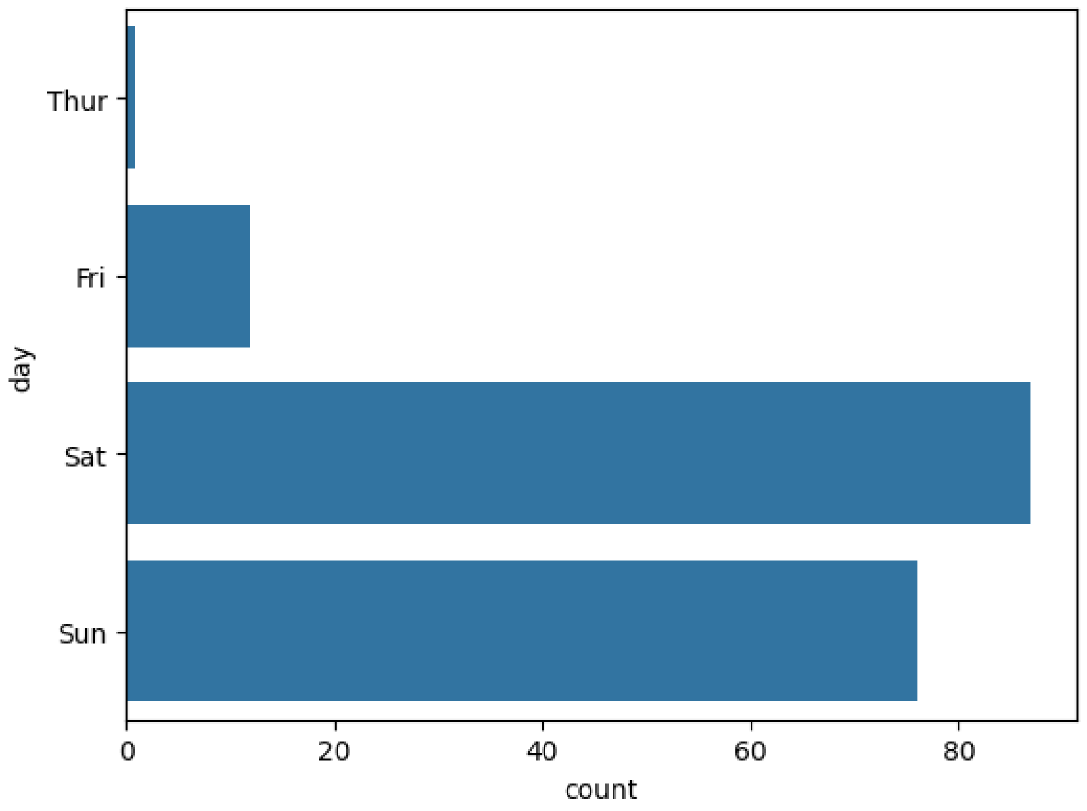
Male
Female
sex
0
20
40
60
80
100
120
140
160
count



Male
Female
count



160
140
120
100
80
60
40
20
0
Male
Female
sex



Thur
Fri
Sat
Sun
day
0
20
40
60
80
count

```
//Karthik Subramanian S

//240701233

// 7.08.2025

// Title : ”Handling Missing and Inappropriate Data Set”

import numpy as np

import pandas as pd

df=pd.read_csv("Hotel_Dataset.csv")

df

df.duplicated()

df.info()

df.drop_duplicates(inplace=True)

df

len(df)

index=np.array(list(range(0,len(df))))

df.set_index(index,inplace=True)

index

df

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

df

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

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

df

df.Age_Group.unique()

df.Hotel.unique()

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

df.FoodPreference.replace(['Vegetarian','veg'],'Veg',inplace=True)

df.FoodPreference.replace(['non-Veg'],'Non-Veg',inplace=True)

df.EstimatedSalary.fillna(round(df.EstimatedSalary.mean()),inplace=True)

df.NoOfPax.fillna(round(df.NoOfPax.median()),inplace=True)

df['Rating(1-5)'].fillna(round(df['Rating(1-5)'].median()), inplace=True)

df.Bill.fillna(round(df.Bill.mean()),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

CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill	NoOfPax	EstimatedSalary	Age_Group.1	
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	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

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

```
<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)
memory usage: 924.0+ bytes
```

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

CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill	NoOfPax	Estimated Salary	Age_Group.1	
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-

CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill	NoOfPax	EstimatedSalary	Age_Group.1	
									30
9	10	30-35	5	RedFox	non-Veg	-6755	4	87777	30-35

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

CustomerID	Age_Group	Rating (1-5)	Hotel	FoodPreference	Bill	NoOfPax	EstimatedSalary	
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

CustomerID	Age_Group	Rating (1-5)	Hotel	FoodPreference	Bill	NoOfPax	EstimatedSalary	
0	1.0	20-25	4	Ibis	veg	1300.0	2	40000.0
1	2.0	30-35	5	LemonTree	Non-Veg	2000.0	3	59000.0
2	3.0	25-30	6	RedFox	Veg	1322.0	2	30000.0
3	4.0	20-25	-1	LemonTree	Veg	1234.0	2	120000.0
4	5.0	35+	3	Ibis	Vegetarian	989.0	2	45000.0
5	6.0	35+	3	Ibys	Non-Veg	1909.0	2	122220.0

CustomerID	Age_Group	Rating (1-5)	Hotel	FoodPreference	Bill	NoOfPax	EstimatedSalary	
6	7.0	35+	4	RedFox	Vegetarian	1000.0	-1	21122.0
7	8.0	20-25	7	LemonTree	Veg	2999.0	-10	345673.0
8	9.0	25-30	2	Ibis	Non-Veg	3456.0	3	NaN
9	10.0	30-35	5	RedFox	non-Veg	NaN	4	87777.0

CustomerID	Age_Group	Rating (1-5)	Hotel	FoodPreference	Bill	NoOfPax	EstimatedSalary	
0	1.0	20-25	4	Ibis	veg	1300.0	2.0	40000.0
1	2.0	30-35	5	LemonTree	Non-Veg	2000.0	3.0	59000.0
2	3.0	25-30	6	RedFox	Veg	1322.0	2.0	30000.0
3	4.0	20-25	-1	LemonTree	Veg	1234.0	2.0	120000.0
4	5.0	35+	3	Ibis	Vegetarian	989.0	2.0	45000.0
5	6.0	35+	3	Ibys	Non-Veg	1909.0	2.0	122220.0

CustomerID	**Age_Group**	**Rating (1-5)**	**Hotel**	**FoodPreference**	**Bill**	**NoOfPax**	**EstimatedSalary**	
6	7.0	35+	4	RedFox	Vegetarian	1000.0	NaN	21122.0
7	8.0	20-25	7	LemonTree	Veg	2999.0	NaN	345673.0
8	9.0	25-30	2	Ibis	Non-Veg	3456.0	3.0	NaN
9	10.0	30-35	5	RedFox	non-Veg	NaN	4.0	87777.0

CustomerID	**Age_Group**	**Rating (1-5)**	**Hotel**	**FoodPreference**	**Bill**	**NoOfPax**	**EstimatedSalary**	
0	1.0	20-25	4	Ibis	Veg	1300.0	2.0	40000.0
1	2.0	30-35	5	LemonTree	Non-Veg	2000.0	3.0	59000.0
2	3.0	25-30	6	RedFox	Veg	1322.0	2.0	30000.0
3	4.0	20-25	-1	LemonTree	Veg	1234.0	2.0	120000.0
4	5.0	35+	3	Ibis	Veg	989.0	2.0	45000.0

CustomerID	**Age_Group**	**Rating (1-5)**	**Hotel**	**FoodPreference**	**Bill**	**NoOfPax**	**EstimatedSalary**	
5	6.0	35+	3	Ibis	Non-Veg	1909.0	2.0	122220.0
6	7.0	35+	4	RedFox	Veg	1000.0	2.0	21122.0
7	8.0	20-25	7	LemonTree	Veg	2999.0	2.0	345673.0
8	9.0	25-30	2	Ibis	Non-Veg	3456.0	3.0	96755.0
9	10.0	30-35	5	RedFox	Non-Veg	1801.0	4.0	87777.0

```
//Karthik Subramanian S

//240701233

// 7.08.2025

// Title: “Data Preprocessing in Data Science”


import numpy as np

import pandas as pd

df=pd.read_csv("pre_process_datasample.csv")

df

df.info()

df.Country.mode()

df.Country.mode()[0]

type(df.Country.mode())

df.Country.fillna(df.Country.mode()[0],inplace=True)

df

df.Age.fillna(df.Age.median(),inplace=True)

df

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

pd.get_dummies(df.Country)

updated_dataset=pd.concat([pd.get_dummies(df.Country),df.iloc[:,[1,2,3]]],axis=1)

updated_dataset

updated_dataset.Purchased.replace(['No','Yes'],[0,1],inplace=True)

updated_dataset
```

```
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
```

```
<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: 452.0+ bytes


0    France

Name: Country, dtype: object

'France'


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


          Germany    Spain
France

0         True       False    False
```

```
France   Germany   Spain

1        False     False   True

2        False     True    False

3        False     False   True

4        False     True    False

5        True      False   False

6        False     False   True

7        True      False   False

8        False     True    False

9        True      False   False



France   Germany   Spain   Age    Salary    Purchased

0        True      False   False  44.0   72000.0     No

1        False     False   True   27.0   48000.0     Yes

2        False     True    False  30.0   54000.0     No

3        False     False   True   38.0   61000.0     No

4        False     True    False  40.0   63778.0     Yes

5        True      False   False  35.0   58000.0     Yes
```

```
 France   Germany   Spain   Age    Salary    Purchased

 6        False     False   True   38.0      52000.0     No

 7        True      False   False  48.0      79000.0     Yes

 8        False     True    False  50.0      83000.0     No

 9        True      False   False  37.0      67000.0     Yes
```

```
<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: 452.0+ bytes
```

```
 France   Germany   Spain   Age    Salary    Purchased

 0        True      False   False  44.0      72000.0     0

 1        False     False   True   27.0      48000.0     1
```

France	**Germany**	**Spain**	**Age**	**Salary**	**Purchased**	
2	False	True	False	30.0	54000.0	0
3	False	False	True	38.0	61000.0	0
4	False	True	False	40.0	63778.0	1
5	True	False	False	35.0	58000.0	1
6	False	False	True	38.0	52000.0	0
7	True	False	False	48.0	79000.0	1
8	False	True	False	50.0	83000.0	0
9	True	False	False	37.0	67000.0	1

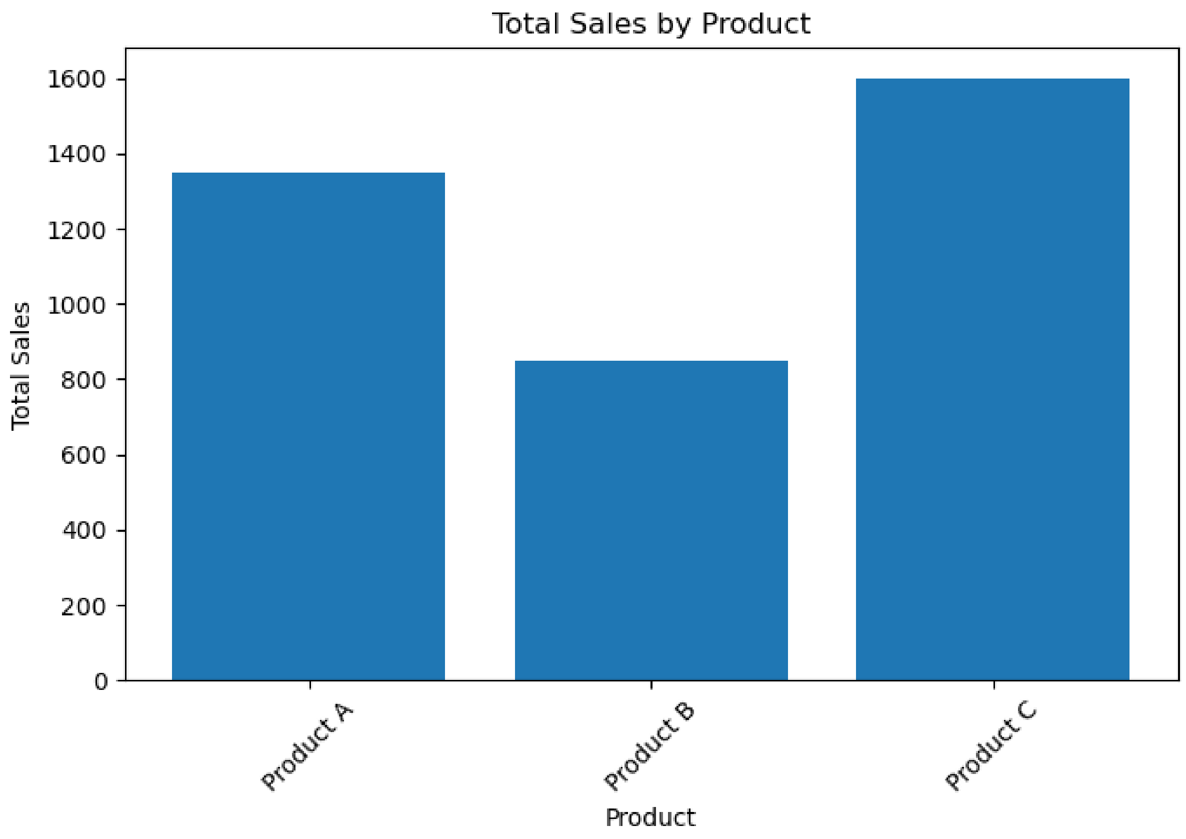
```
// Karthik Subramanian S
//240701233
//5.08.2025
// Title : Data preprocessing and visualization
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
file_path = 'sales_data.csv'
df = pd.read_csv(file_path)

print("First 5 rows of the dataset:")
print(df.head())
print("\nMissing values in each column:")
print(df.isnull().sum())
df['Sales'].fillna(df['Sales'].mean(),  inplace=True)
df.dropna(subset=['Product', 'Quantity', 'Region'], inplace=True)

print("\nSummary statistics:")
print(df.describe())

product_summary = df.groupby('Product').agg({
    'Sales': 'sum',
    'Quantity': 'sum'
}).reset_index()
print("\nProduct summary:")
print(product_summary)

plt.figure(figsize=(7,5))
plt.bar(product_summary['Product'], product_summary['Sales'])
plt.xlabel('Product')
plt.ylabel('Total Sales')
plt.title('Total Sales by Product')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

try:
   df['Date'] = pd.to_datetime(df['Date'], format='%d-%m-%Y', dayfirst=True)
except ValueError:
   df['Date'] = pd.to_datetime(df['Date'], dayfirst=True)

sales_over_time = df.groupby(pd.Grouper(key='Date', freq='D')).agg({'Sales': 'sum'}).reset_index()
plt.figure(figsize=(7,5))
plt.plot(sales_over_time['Date'], sales_over_time['Sales'])
plt.xlabel('Date')
plt.ylabel('Total Sales')
plt.title('Sales Over Time')
plt.gcf().autofmt_xdate()
plt.tight_layout()
plt.show()
pivot_table = df.pivot_table(values='Sales', index='Region', columns='Product',
                             aggfunc=np.sum, fill_value=0)
print("\nPivot table of sales by region and product:")
print(pivot_table)

numeric_df = df.select_dtypes(include=[np.number])  # Only include numeric columns
correlation_matrix = numeric_df.corr()
print("\nCorrelation matrix:")
print(correlation_matrix)

plt.figure(figsize=(7,5))
```

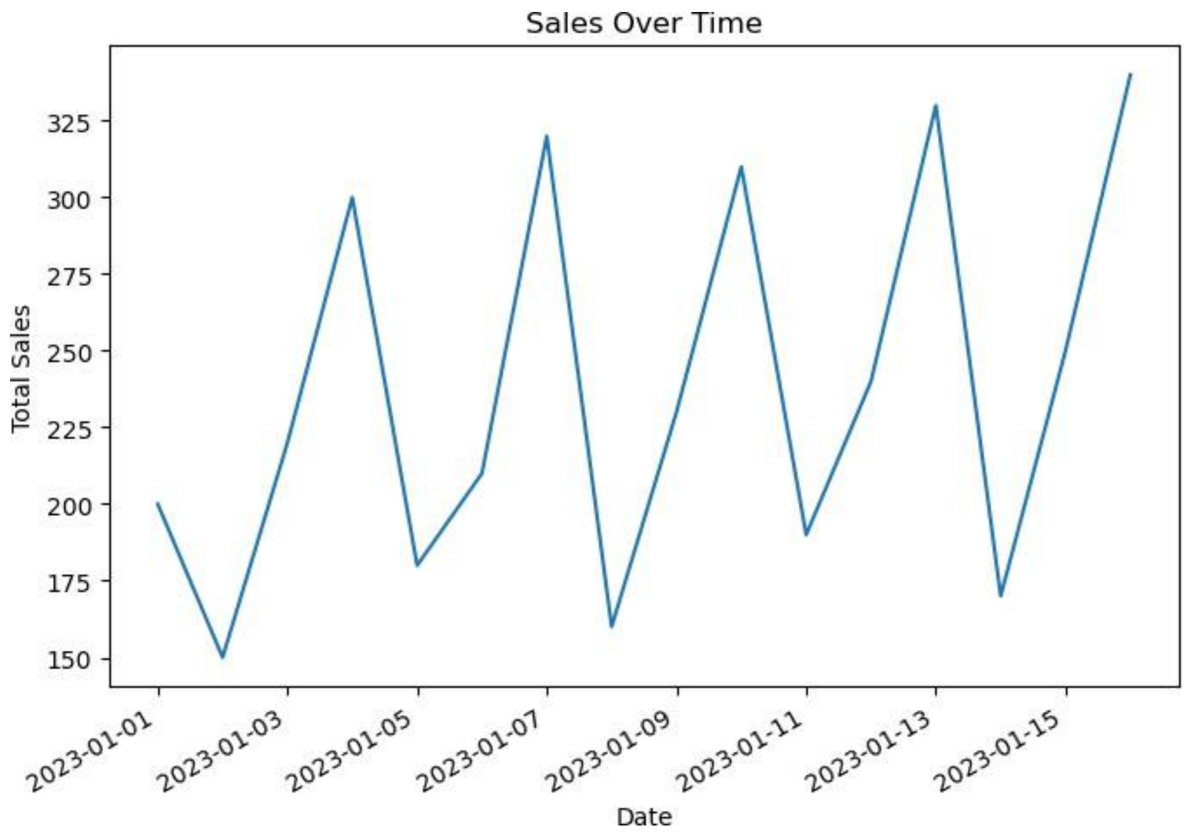
```
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Matrix')
plt.tight_layout()
plt.show()
```

```
First 5 rows of the dataset:
        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

Missing values in each column:
Date        0
Product     0
Sales       0
Quantity    0
Region      0
dtype: int64

Summary statistics:
            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
```

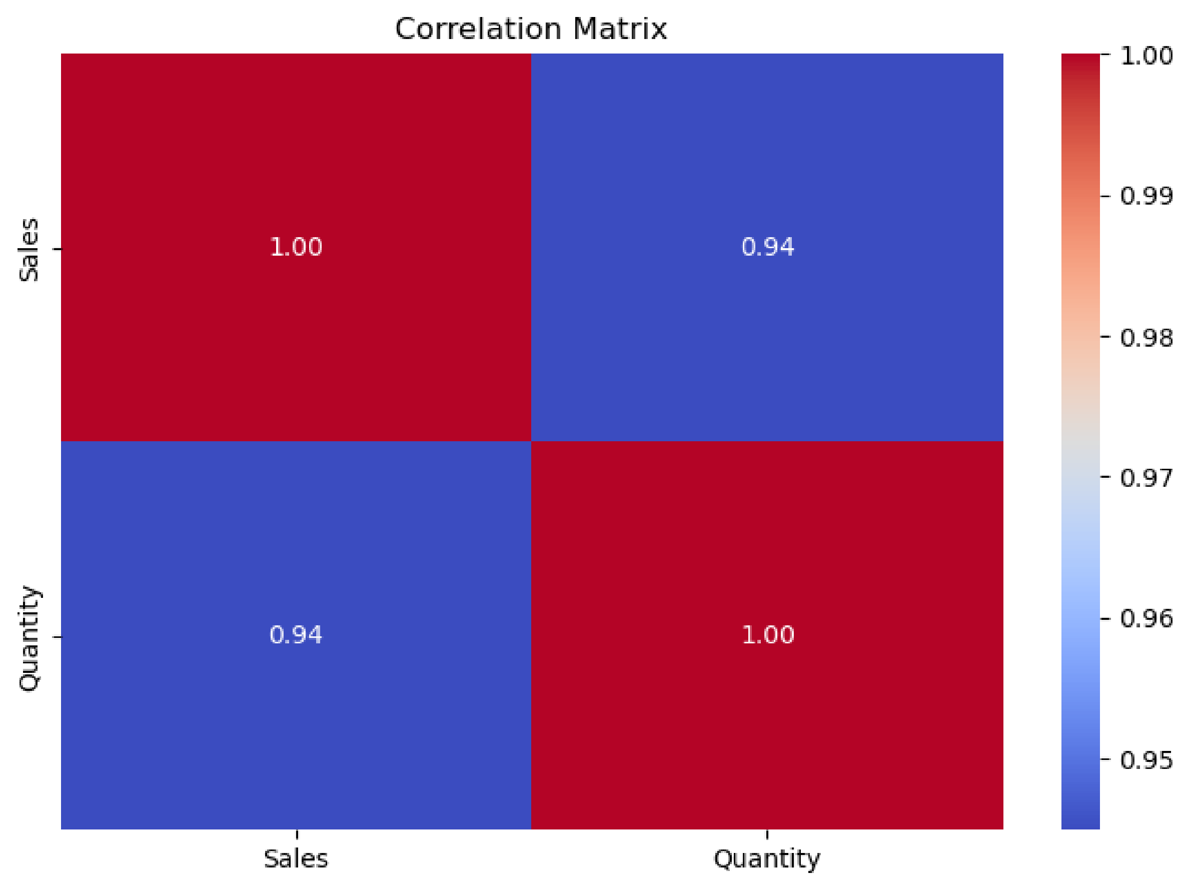
```
Product summary:
     Product  Sales  Quantity
0  Product A   1350        33
1  Product B    850        17
2  Product C   1600        36

First 5 rows of the dataset:
        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

Missing values in each column:
Date        0
Product     0
Sales       0
Quantity    0
Region      0
dtype: int64

Summary statistics:
           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 summary:
     Product  Sales  Quantity
0  Product A   1350        33
1  Product B    850        17
2  Product C   1600        36
```





```
Pivot table of sales by region and product:
Product  Product A  Product B  Product C
Region
East             0          0       1600
North         1350          0          0
South            0        480          0
West             0        370          0

Correlation matrix:
           Sales  Quantity
Sales     1.000000 0.944922
Quantity  0.944922  1.000000
```


Correlation Matrix
Sales
Quantity
1.00
0.94
0.94
1.00
Sales
Quantity
1.00
0.99
0.98
0.97
0.96
0.95

```
// Name: Karthik Subramanian S

// 240701233

// Title : “Linear Regression”

import numpy as np

import pandas as pd

df=pd.read_csv('Salary_data.csv')

df

df.info()

df.dropna(inplace=True)

df.info()

df.describe()

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=42)

from sklearn.linear_model import LinearRegression

model=LinearRegression()

model.fit(x_train,y_train)

model.score(x_train,y_train)

model.score(x_test,y_test)

model.coef_

model.intercept_

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))
```

```
<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
```

```
<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
```

	YearsExperience	**Salary**
count	30.000000	30.000000

```
       YearsExperience  Salary

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
```

```
LinearRegression?i
LinearRegression()



0.9645401573418146
0.9024461774180497

array([[9423.81532303]])

array([25321.58301178])

Enter Years of Experience: 44

Estimated Salary for 44.0 years of experience is [[439969.45722514]]:
```

```
//Karthik Subramanian S

//240701233

//Title : “Logistics Regression”


import numpy as np

import pandas as pd

df=pd.read_csv('Social_Network_Ads.csv')

df

df.head()

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

label=df.iloc[:,4].values

features

label

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_size=0.2,
random_state=i)

    model = LogisticRegression(max_iter=1000)

    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:

        print("Test: {} Train: {} Random State: {}".format(test_score, train_score, i))


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


x_train, x_test, y_train, y_test = train_test_split(features, label, test_size=0.2,
random_state=42)


finalModel = LogisticRegression(max_iter=1000)

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)))
```

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

```
 User ID   Gender     Age      EstimatedSalary   Purchased

 398       15755018   Male     36                33000       0

 399       15594041   Female   49                36000       1

400 rows × 5 columns
```

```
 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
```

```
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],
       [    48,  29000],
       [    45,  22000],
       [    47,  49000],
       [    48,  41000],
       [    45,  22000],
       [    46,  23000],
       [    47,  20000],
       [    49,  28000],
       [    47,  30000],
       [    29,  43000],
       [    31,  18000],
       [    31,  74000],
       [    27, 137000],
       [    21,  16000],
       [    28,  44000],
       [    27,  90000],
       [    35,  27000],
       [    33,  28000],
       [    30,  49000],
       [    26,  72000],
       [    27,  31000],
       [    27,  17000],
       [    33,  51000],
       [    35, 108000],
       [    30,  15000],
       [    28,  84000],
       [    23,  20000],
       [    25,  79000],
       [    27,  54000],
       [    30, 135000],
       [    31,  89000],
       [    24,  32000],
       [    18,  44000],
       [    29,  83000],
       [    35,  23000],
       [    27,  58000],
       [    24,  55000],
       [    23,  48000],
       [    28,  79000],
       [    22,  18000],
       [    32, 117000],
       [    27,  20000],
       [    25,  87000],
       [    23,  66000],
       [    32, 120000],
       [    59,  83000],
       [    24,  58000],
       [    24,  19000],
       [    23,  82000],
       [    22,  63000],
       [    31,  68000],
       [    25,  80000],
       [    24,  27000],
       [    20,  23000],
       [    33, 113000],
       [    32,  18000],
       [    34, 112000],
       [    18,  52000],
       [    22,  27000],
       [    28,  87000],
       [    26,  17000],
       [    30,  80000],
       [    39,  42000],
       [    20,  49000],
       [    35,  88000],
       [    30,  62000],
       [    31, 118000],
       [    24,  55000],
       [    28,  85000],
       [    26,  81000],
       [    35,  50000],
       [    22,  81000],
       [    30, 116000],
       [    26,  15000],
       [    29,  28000],
       [    29,  83000],
       [    35,  44000],
       [    35,  25000],
       [    28, 123000],
       [    35,  73000],
       [    28,  37000],
       [    27,  88000],
       [    28,  59000],
       [    32,  86000],
       [    33, 149000],
       [    19,  21000],
       [    21,  72000],
       [    26,  35000],
       [    27,  89000],
       [    26,  86000],
       [    38,  80000],
       [    39,  71000],
       [    37,  71000],
       [    38,  61000],
       [    37,  55000],
       [    42,  80000],
       [    40,  57000],
       [    35,  75000],
       [    36,  52000],
       [    40,  59000],
       [    41,  59000],
       [    36,  75000],
       [    37,  72000],
       [    40,  75000],
       [    35,  53000],
       [    41,  51000],
       [    39,  61000],
       [    42,  65000],
       [    26,  32000],
       [    30,  17000],
       [    26,  84000],
       [    31,  58000],
       [    33,  31000],
       [    30,  87000],
       [    21,  68000],
       [    28,  55000],
       [    23,  63000],
       [    20,  82000],
       [    30, 107000],
       [    28,  59000],
       [    19,  25000],
       [    19,  85000],
       [    18,  68000],
       [    35,  59000],
       [    30,  89000],
       [    34,  25000],
       [    24,  89000],
       [    27,  96000],
       [    41,  30000],
       [    29,  61000],
       [    20,  74000],
       [    26,  15000],
       [    41,  45000],
       [    31,  76000],
       [    36,  50000],
       [    40,  47000],
       [    31,  15000],
       [    46,  59000],
       [    29,  75000],
       [    26,  30000],
       [    32, 135000],
       [    32, 100000],
       [    25,  90000],
       [    37,  33000],
       [    35,  38000],
       [    33,  69000],
       [    18,  86000],
       [    22,  55000],
       [    35,  71000],
       [    29, 148000],
       [    29,  47000],
       [    21,  88000],
       [    34, 115000],
       [    26, 118000],
       [    34,  43000],
       [    34,  72000],
       [    23,  28000],
       [    35,  47000],
       [    25,  22000],
       [    24,  23000],
       [    31,  34000],
       [    26,  16000],
       [    31,  71000],
       [    32, 117000],
       [    33,  43000],
       [    33,  60000],
       [    31,  66000],
       [    20,  82000],
       [    33,  41000],
       [    35,  72000],
       [    28,  32000],
       [    24,  84000],
       [    19,  26000],
       [    29,  43000],
       [    19,  70000],
       [    28,  89000],
       [    34,  43000],
       [    30,  79000],
       [    20,  36000],
       [    26,  80000],
       [    35,  22000],
       [    35,  39000],
       [    49,  74000],
       [    39, 134000],
       [    41,  71000],
       [    58, 101000],
       [    47,  47000],
       [    55, 130000],
       [    52, 114000],
       [    40, 142000],
       [    46,  22000],
       [    48,  96000],
       [    52, 150000],
       [    59,  42000],
       [    35,  58000],
       [    47,  43000],
       [    60, 108000],
       [    49,  65000],
       [    40,  78000],
       [    46,  96000],
       [    59, 143000],
       [    41,  80000],
       [    35,  91000],
       [    37, 144000],
       [    60, 102000],
       [    35,  60000],
       [    37,  53000],
       [    36, 126000],
       [    56, 133000],
       [    40,  72000],
       [    42,  80000],
       [    35, 147000],
       [    39,  42000],
       [    40, 107000],
       [    49,  86000],
       [    38, 112000],
       [    46,  79000],
       [    40,  57000],
       [    37,  80000],
       [    46,  82000],
       [    53, 143000],
       [    42, 149000],
       [    38,  59000],
       [    50,  88000],
       [    56, 104000],
       [    41,  72000],
       [    51, 146000],
       [    35,  50000],
       [    57, 122000],
       [    41,  52000],
       [    35,  97000],
       [    44,  39000],
       [    37,  52000],
       [    48, 134000],
       [    37, 146000],
       [    50,  44000],
       [    52,  90000],
       [    41,  72000],
       [    40,  57000],
       [    58,  95000],
       [    45, 131000],
       [    35,  77000],
       [    36, 144000],
       [    55, 125000],
       [    35,  72000],
       [    48,  90000],
       [    42, 108000],
       [    40,  75000],
       [    37,  74000],
       [    47, 144000],
       [    40,  61000],
       [    43, 133000],
       [    59,  76000],
       [    60,  42000],
       [    39, 106000],
       [    57,  26000],
       [    57,  74000],
       [    38,  71000],
       [    49,  88000],
       [    52,  38000],
       [    50,  36000],
       [    59,  88000],
       [    35,  61000],
       [    37,  70000],
       [    52,  21000],
       [    48, 141000],
       [    37,  93000],
       [    37,  62000],
       [    48, 138000],
       [    41,  79000],
       [    37,  78000],
       [    39, 134000],
       [    49,  89000],
       [    55,  39000],
       [    37,  77000],
       [    35,  57000],
       [    36,  63000],
       [    42,  73000],
       [    43, 112000],
       [    45,  79000],
       [    46, 117000],
       [    58,  38000],
       [    48,  74000],
       [    37, 137000],
       [    37,  79000],
       [    40,  60000],
       [    42,  54000],
       [    51, 134000],
       [    47, 113000],
       [    36, 125000],
       [    38,  50000],
       [    42,  70000],
       [    39,  96000],
       [    38,  50000],
       [    49, 141000],
       [    39,  79000],
       [    39,  75000],
       [    54, 104000],
       [    35,  55000],
       [    45,  32000],
       [    36,  60000],
       [    52, 138000],
       [    53,  82000],
       [    41,  52000],
       [    48,  30000],
       [    48, 131000],
       [    41,  60000],
       [    41,  72000],
       [    42,  75000],
       [    36, 118000],
       [    47, 107000],
       [    38,  51000],
       [    48, 119000],
       [    42,  65000],
       [    40,  65000],
       [    57,  60000],
       [    36,  54000],
       [    58, 144000],
       [    35,  79000],
       [    38,  55000],
       [    39, 122000],
       [    53, 104000],
       [    35,  75000],
       [    38,  65000],
       [    47,  51000],
       [    47, 105000],
       [    41,  63000],
       [    53,  72000],
       [    54, 108000],
       [    39,  77000],
       [    38,  61000],
       [    38, 113000],
       [    37,  75000],
       [    42,  90000],
       [    37,  57000],
       [    36,  99000],
       [    60,  34000],
       [    54,  70000],
       [    41,  72000],
       [    40,  71000],
       [    42,  54000],
       [    43, 129000],
       [    53,  34000],
       [    47,  50000],
       [    42,  79000],
       [    42, 104000],
       [    59,  29000],
       [    58,  47000],
       [    46,  88000],
       [    38,  71000],
       [    54,  26000],
       [    60,  46000],
       [    60,  83000],
       [    39,  73000],
       [    59, 130000],
       [    37,  80000],
       [    46,  32000],
       [    46,  74000],
       [    42,  53000],
       [    41,  87000],
       [    58,  23000],
       [    42,  64000],
       [    48,  33000],
       [    44, 139000],
       [    49,  28000],
       [    57,  33000],
       [    56,  60000],
       [    49,  39000],
       [    39,  71000],
       [    47,  34000],
       [    48,  35000],
       [    48,  33000],
       [    47,  23000],
       [    45,  45000],
       [    60,  42000],
       [    39,  59000],
       [    46,  41000],
       [    51,  23000],
       [    50,  20000],
       [    36,  33000],
       [    49,  36000]])

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, 1, 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, 1, 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, 1, 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, 1, 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])

Test: 0.9 Train: 0.840625 Random State: 4
Test: 0.8625 Train: 0.85 Random State: 5
Test: 0.8625 Train: 0.859375 Random State: 6
Test: 0.8875 Train: 0.8375 Random State: 7
Test: 0.8625 Train: 0.8375 Random State: 9
Test: 0.9 Train: 0.840625 Random State: 10
Test: 0.8625 Train: 0.85625 Random State: 14
Test: 0.85 Train: 0.84375 Random State: 15
Test: 0.8625 Train: 0.85625 Random State: 16
Test: 0.875 Train: 0.834375 Random State: 18
Test: 0.85 Train: 0.84375 Random State: 19
```

```
Test: 0.875 Train: 0.84375 Random State: 20

Test: 0.8625 Train: 0.834375 Random State: 21

Test: 0.875 Train: 0.840625 Random State: 22

Test: 0.875 Train: 0.840625 Random State: 24

Test: 0.85 Train: 0.834375 Random State: 26

Test: 0.85 Train: 0.840625 Random State: 27

Test: 0.8625 Train: 0.834375 Random State: 30

Test: 0.8625 Train: 0.85625 Random State: 31

Test: 0.875 Train: 0.853125 Random State: 32

Test: 0.8625 Train: 0.84375 Random State: 33

Test: 0.875 Train: 0.83125 Random State: 35

Test: 0.8625 Train: 0.853125 Random State: 36

Test: 0.8875 Train: 0.840625 Random State: 38

Test: 0.875 Train: 0.8375 Random State: 39

Test: 0.8875 Train: 0.8375 Random State: 42

Test: 0.875 Train: 0.846875 Random State: 46

Test: 0.9125 Train: 0.83125 Random State: 47

Test: 0.875 Train: 0.83125 Random State: 51

Test: 0.9 Train: 0.84375 Random State: 54

Test: 0.85 Train: 0.84375 Random State: 57

Test: 0.875 Train: 0.84375 Random State: 58

Test: 0.925 Train: 0.8375 Random State: 61

Test: 0.8875 Train: 0.834375 Random State: 65

Test: 0.8875 Train: 0.840625 Random State: 68

Test: 0.9 Train: 0.83125 Random State: 72

Test: 0.8875 Train: 0.8375 Random State: 75

Test: 0.925 Train: 0.825 Random State: 76

Test: 0.8625 Train: 0.840625 Random State: 77
```

```
Test: 0.8625 Train: 0.859375 Random State: 81

Test: 0.875 Train: 0.8375 Random State: 82

Test: 0.8875 Train: 0.8375 Random State: 83

Test: 0.8625 Train: 0.853125 Random State: 84

Test: 0.8625 Train: 0.840625 Random State: 85

Test: 0.8625 Train: 0.840625 Random State: 87

Test: 0.875 Train: 0.846875 Random State: 88

Test: 0.9125 Train: 0.8375 Random State: 90

Test: 0.8625 Train: 0.85 Random State: 95

Test: 0.875 Train: 0.85 Random State: 99

Test: 0.85 Train: 0.840625 Random State: 101

Test: 0.85 Train: 0.840625 Random State: 102

Test: 0.9 Train: 0.825 Random State: 106

Test: 0.8625 Train: 0.840625 Random State: 107

Test: 0.85 Train: 0.834375 Random State: 109

Test: 0.85 Train: 0.840625 Random State: 111

Test: 0.9125 Train: 0.840625 Random State: 112

Test: 0.8625 Train: 0.85 Random State: 115

Test: 0.8625 Train: 0.840625 Random State: 116

Test: 0.875 Train: 0.834375 Random State: 119

Test: 0.9125 Train: 0.828125 Random State: 120

Test: 0.8625 Train: 0.859375 Random State: 125

Test: 0.85 Train: 0.846875 Random State: 128

Test: 0.875 Train: 0.85 Random State: 130

Test: 0.9 Train: 0.84375 Random State: 133

Test: 0.925 Train: 0.834375 Random State: 134

Test: 0.8625 Train: 0.85 Random State: 135

Test: 0.875 Train: 0.83125 Random State: 138
```

```
Test: 0.8625 Train: 0.85 Random State: 141

Test: 0.85 Train: 0.846875 Random State: 143

Test: 0.85 Train: 0.846875 Random State: 146

Test: 0.85 Train: 0.84375 Random State: 147

Test: 0.8625 Train: 0.85 Random State: 148

Test: 0.875 Train: 0.8375 Random State: 150

Test: 0.8875 Train: 0.83125 Random State: 151

Test: 0.925 Train: 0.84375 Random State: 152

Test: 0.85 Train: 0.840625 Random State: 153

Test: 0.9 Train: 0.84375 Random State: 154

Test: 0.9 Train: 0.840625 Random State: 155

Test: 0.8875 Train: 0.846875 Random State: 156

Test: 0.8875 Train: 0.834375 Random State: 158

Test: 0.875 Train: 0.828125 Random State: 159

Test: 0.9 Train: 0.83125 Random State: 161

Test: 0.85 Train: 0.8375 Random State: 163

Test: 0.875 Train: 0.83125 Random State: 164

Test: 0.8625 Train: 0.85 Random State: 169

Test: 0.875 Train: 0.840625 Random State: 171

Test: 0.85 Train: 0.840625 Random State: 172

Test: 0.9 Train: 0.825 Random State: 180

Test: 0.85 Train: 0.834375 Random State: 184

Test: 0.925 Train: 0.821875 Random State: 186

Test: 0.9 Train: 0.83125 Random State: 193

Test: 0.8625 Train: 0.85 Random State: 195

Test: 0.8625 Train: 0.840625 Random State: 196

Test: 0.8625 Train: 0.8375 Random State: 197

Test: 0.875 Train: 0.840625 Random State: 198
```

```
Test: 0.8875 Train: 0.8375 Random State: 199

Test: 0.8875 Train: 0.84375 Random State: 200

Test: 0.8625 Train: 0.8375 Random State: 202

Test: 0.8625 Train: 0.840625 Random State: 203

Test: 0.8875 Train: 0.83125 Random State: 206

Test: 0.8625 Train: 0.834375 Random State: 211

Test: 0.85 Train: 0.84375 Random State: 212

Test: 0.8625 Train: 0.834375 Random State: 214

Test: 0.875 Train: 0.83125 Random State: 217

Test: 0.9625 Train: 0.81875 Random State: 220

Test: 0.875 Train: 0.84375 Random State: 221

Test: 0.85 Train: 0.840625 Random State: 222

Test: 0.9 Train: 0.84375 Random State: 223

Test: 0.8625 Train: 0.853125 Random State: 227

Test: 0.8625 Train: 0.834375 Random State: 228

Test: 0.9 Train: 0.840625 Random State: 229

Test: 0.85 Train: 0.84375 Random State: 232

Test: 0.875 Train: 0.846875 Random State: 233

Test: 0.9125 Train: 0.840625 Random State: 234

Test: 0.8625 Train: 0.840625 Random State: 235

Test: 0.85 Train: 0.846875 Random State: 236

Test: 0.875 Train: 0.846875 Random State: 239

Test: 0.85 Train: 0.84375 Random State: 241

Test: 0.8875 Train: 0.85 Random State: 242

Test: 0.8875 Train: 0.825 Random State: 243

Test: 0.875 Train: 0.846875 Random State: 244

Test: 0.875 Train: 0.840625 Random State: 245

Test: 0.875 Train: 0.846875 Random State: 246
```

```
Test: 0.8625 Train: 0.859375 Random State: 247

Test: 0.8875 Train: 0.84375 Random State: 248

Test: 0.8625 Train: 0.85 Random State: 250

Test: 0.875 Train: 0.83125 Random State: 251

Test: 0.8875 Train: 0.84375 Random State: 252

Test: 0.8625 Train: 0.846875 Random State: 255

Test: 0.9 Train: 0.840625 Random State: 257

Test: 0.8625 Train: 0.85625 Random State: 260

Test: 0.8625 Train: 0.840625 Random State: 266

Test: 0.8625 Train: 0.8375 Random State: 268

Test: 0.875 Train: 0.840625 Random State: 275

Test: 0.8625 Train: 0.85 Random State: 276

Test: 0.925 Train: 0.8375 Random State: 277

Test: 0.875 Train: 0.846875 Random State: 282

Test: 0.85 Train: 0.846875 Random State: 283

Test: 0.85 Train: 0.84375 Random State: 285

Test: 0.9125 Train: 0.834375 Random State: 286

Test: 0.85 Train: 0.840625 Random State: 290

Test: 0.85 Train: 0.840625 Random State: 291

Test: 0.85 Train: 0.846875 Random State: 292

Test: 0.8625 Train: 0.8375 Random State: 294

Test: 0.8875 Train: 0.828125 Random State: 297

Test: 0.8625 Train: 0.834375 Random State: 300

Test: 0.8625 Train: 0.85 Random State: 301

Test: 0.8875 Train: 0.85 Random State: 302

Test: 0.875 Train: 0.846875 Random State: 303

Test: 0.8625 Train: 0.834375 Random State: 305

Test: 0.9125 Train: 0.8375 Random State: 306
```

```
Test: 0.875 Train: 0.846875 Random State: 308

Test: 0.9 Train: 0.84375 Random State: 311

Test: 0.8625 Train: 0.834375 Random State: 313

Test: 0.9125 Train: 0.834375 Random State: 314

Test: 0.875 Train: 0.8375 Random State: 315

Test: 0.9 Train: 0.846875 Random State: 317

Test: 0.9125 Train: 0.821875 Random State: 319

Test: 0.8625 Train: 0.85 Random State: 321

Test: 0.9125 Train: 0.828125 Random State: 322

Test: 0.85 Train: 0.846875 Random State: 328

Test: 0.85 Train: 0.8375 Random State: 332

Test: 0.8875 Train: 0.853125 Random State: 336

Test: 0.85 Train: 0.8375 Random State: 337

Test: 0.875 Train: 0.840625 Random State: 343

Test: 0.8625 Train: 0.84375 Random State: 346

Test: 0.8875 Train: 0.83125 Random State: 351

Test: 0.8625 Train: 0.85 Random State: 352

Test: 0.95 Train: 0.81875 Random State: 354

Test: 0.8625 Train: 0.85 Random State: 356

Test: 0.9125 Train: 0.840625 Random State: 357

Test: 0.8625 Train: 0.8375 Random State: 358

Test: 0.85 Train: 0.840625 Random State: 362

Test: 0.9 Train: 0.84375 Random State: 363

Test: 0.8625 Train: 0.853125 Random State: 364

Test: 0.9375 Train: 0.821875 Random State: 366

Test: 0.9125 Train: 0.840625 Random State: 369

Test: 0.8625 Train: 0.853125 Random State: 371

Test: 0.925 Train: 0.834375 Random State: 376
```

```
Test: 0.9125 Train: 0.828125 Random State: 377

Test: 0.8875 Train: 0.85 Random State: 378

Test: 0.8875 Train: 0.85 Random State: 379

Test: 0.8625 Train: 0.840625 Random State: 382

Test: 0.8625 Train: 0.859375 Random State: 386

Test: 0.85 Train: 0.8375 Random State: 387

Test: 0.875 Train: 0.828125 Random State: 388

Test: 0.85 Train: 0.84375 Random State: 394

Test: 0.8625 Train: 0.8375 Random State: 395

Test: 0.9 Train: 0.84375 Random State: 397

Test: 0.8625 Train: 0.84375 Random State: 400
```

```
LogisticRegression?i

LogisticRegression(max_iter=1000)
```

```
0.8375

0.8875
```

```
precision    recall  f1-score   support


       0      0.85      0.93      0.89       257

       1      0.85      0.70      0.77       143


   accuracy                         0.85       400

  macro avg      0.85      0.81      0.83       400

weighted avg      0.85      0.85      0.84       400
```

```
//Karthik Subramanian S

//240701233

// 7.10.2025

// Title : “KNN “

import numpy as np

import pandas as pd

df=pd.read_csv('Iris.csv')

df.info()

df.variety.value_counts()

df.head()

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=.2,random_state=32)

model_KNN=KNeighborsClassifier(n_neighbors=5)

model_KNN.fit(xtrain,ytrain)

print(model_KNN.score(xtrain,ytrain))

print(model_KNN.score(xtest,ytest))

from sklearn.metrics import confusion_matrix

confusion_matrix(label,model_KNN.predict(features))

from sklearn.metrics import classification_report

print(classification_report(label,model_KNN.predict(features)))
```

```
<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
```

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

	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

```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

0.9583333333333334

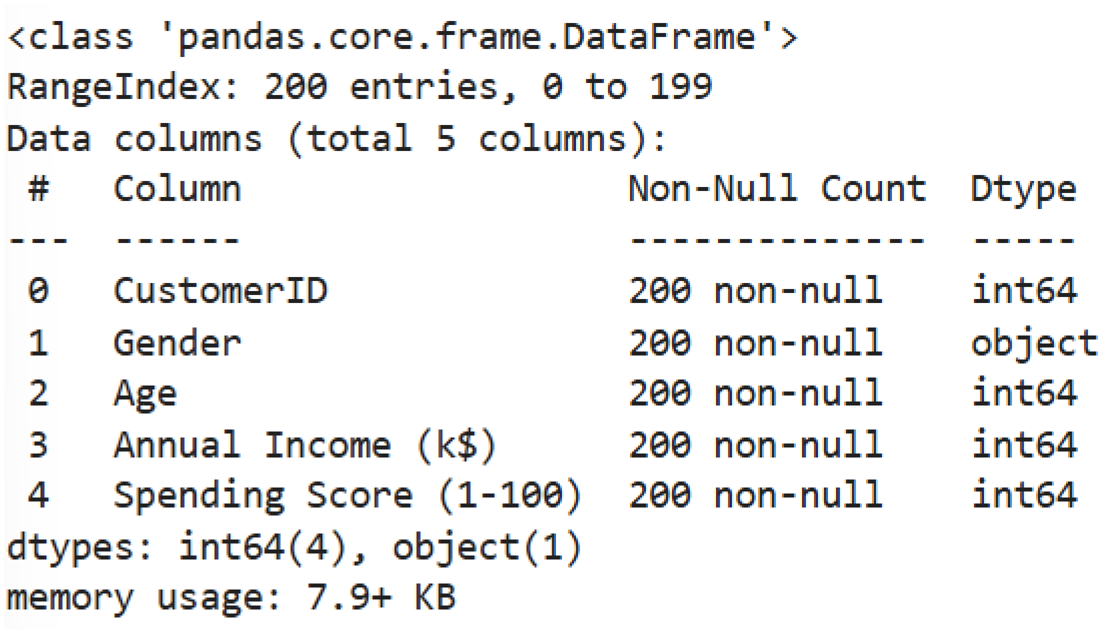
```
1.0
```

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

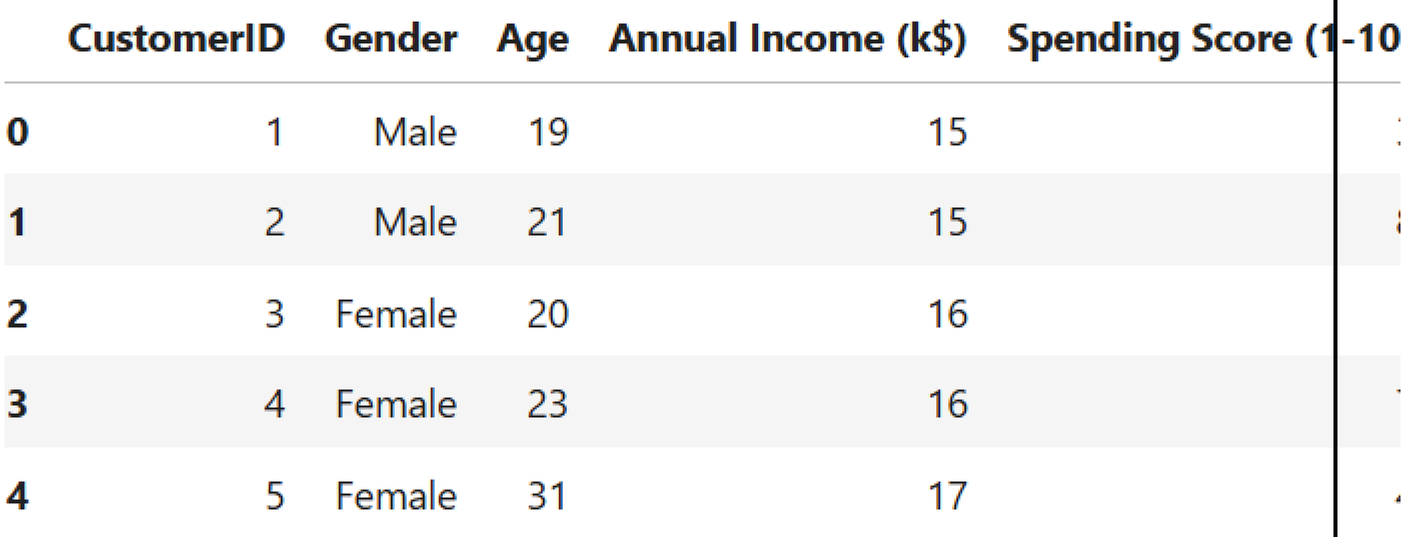
```
              precision    recall  f1-score   support

      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
```

```
// Karthik Subramanian S

// 240701233

//  7.10.2025

// Title : “K- means Clustering “

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

df=pd.read_csv('Mall_Customers.csv')

df.info()

df.head()

sns.pairplot(df)

plt.show()

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

from sklearn.cluster import KMeans

model=KMeans(n_clusters=5)

model.fit(features)

KMeans(n_clusters=5)

Final=df.iloc[:,[3,4]]

Final['label']=model.predict(features)

Final.head()

sns.set_style("whitegrid")

sns.FacetGrid(Final,hue="label",height=8)  \

.map(plt.scatter,"Annual Income (k$)", "Spending Score (1-100)") \

.add_legend();

plt.show()

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

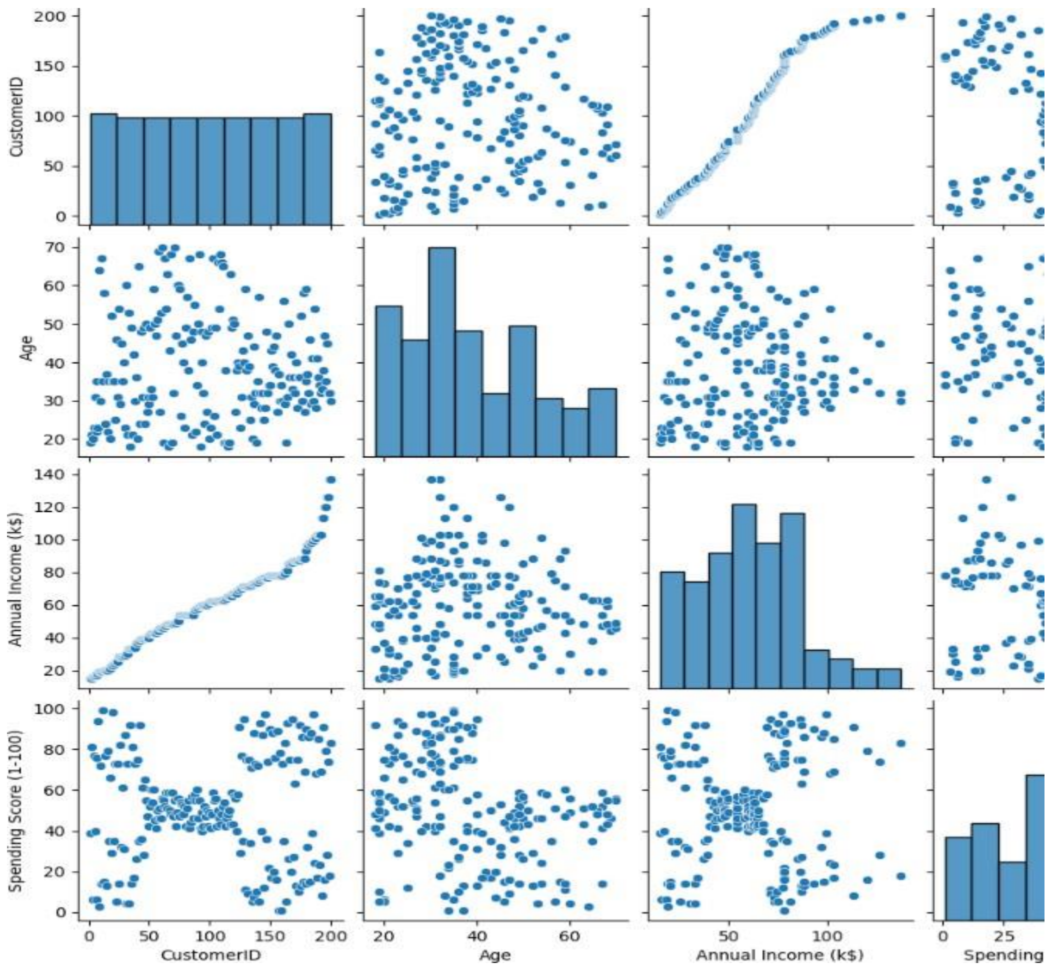
```
from sklearn.cluster import KMeans
wcss=[]
for i in range(1,10):
  model=KMeans(n_clusters=i)
  model.fit(features_el)
  wcss.append(model.inertia_)
plt.plot(range(1,10),wcss)
plt.show()
```



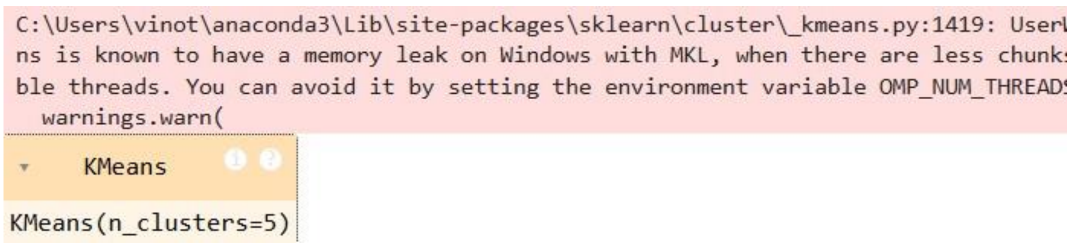
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   CustomerID              200 non-null    int64 
 1   Gender                  200 non-null    object
 2   Age                     200 non-null    int64 
 3   Annual Income (k$)      200 non-null    int64 
 4   Spending Score (1-100)  200 non-null    int64 
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
```



	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-10
0	1	Male	19	15	
1	2	Male	21	15	
2	3	Female	20	16	
3	4	Female	23	16	
4	5	Female	31	17	

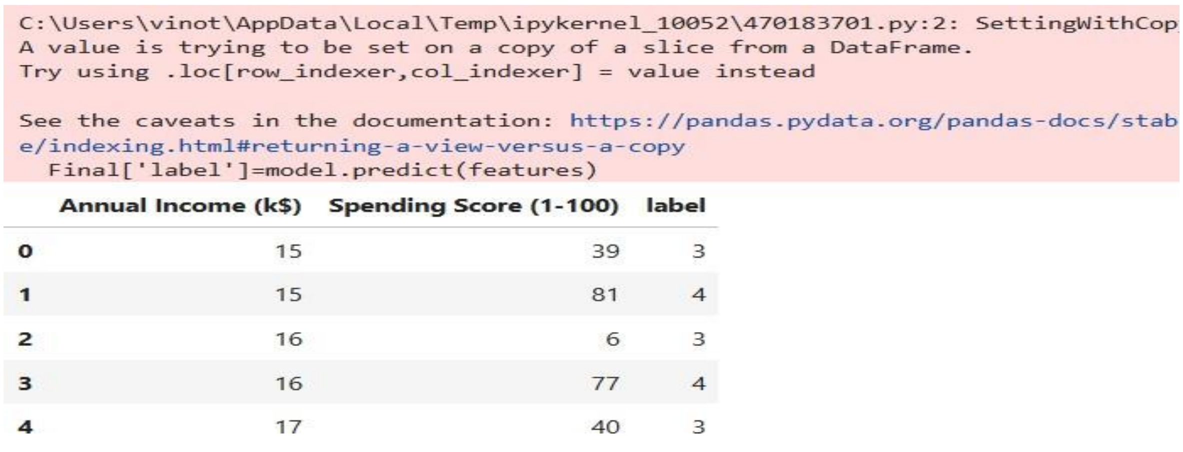


CustomerID
Age
Annual Income (k$)
Spending Score (1-100)
CustomerID
Age
Annual Income (k$)
Spending



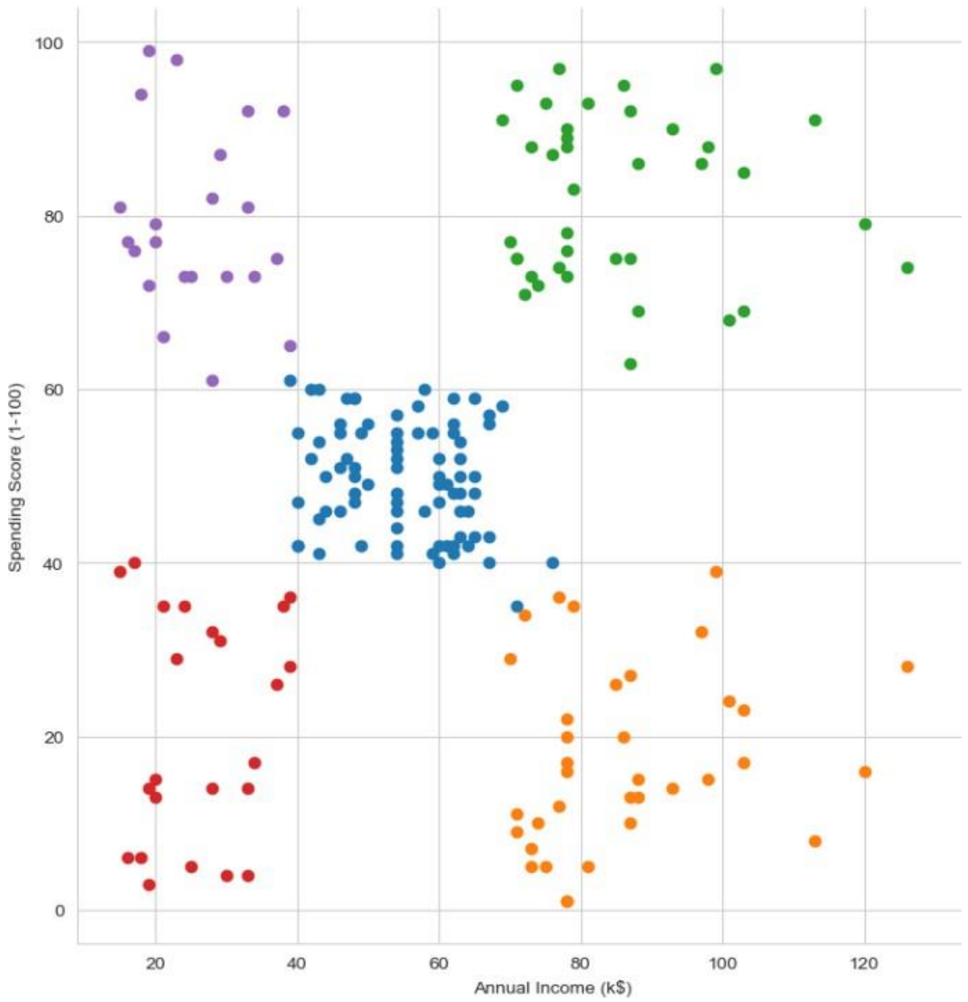
```
C:\Users\vinot\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1419: User
ns is known to have a memory leak on Windows with MKL, when there are less chunk
ble threads. You can avoid it by setting the environment variable OMP_NUM_THREAD
  warnings.warn(
```

```
KMeans(n_clusters=5)
```

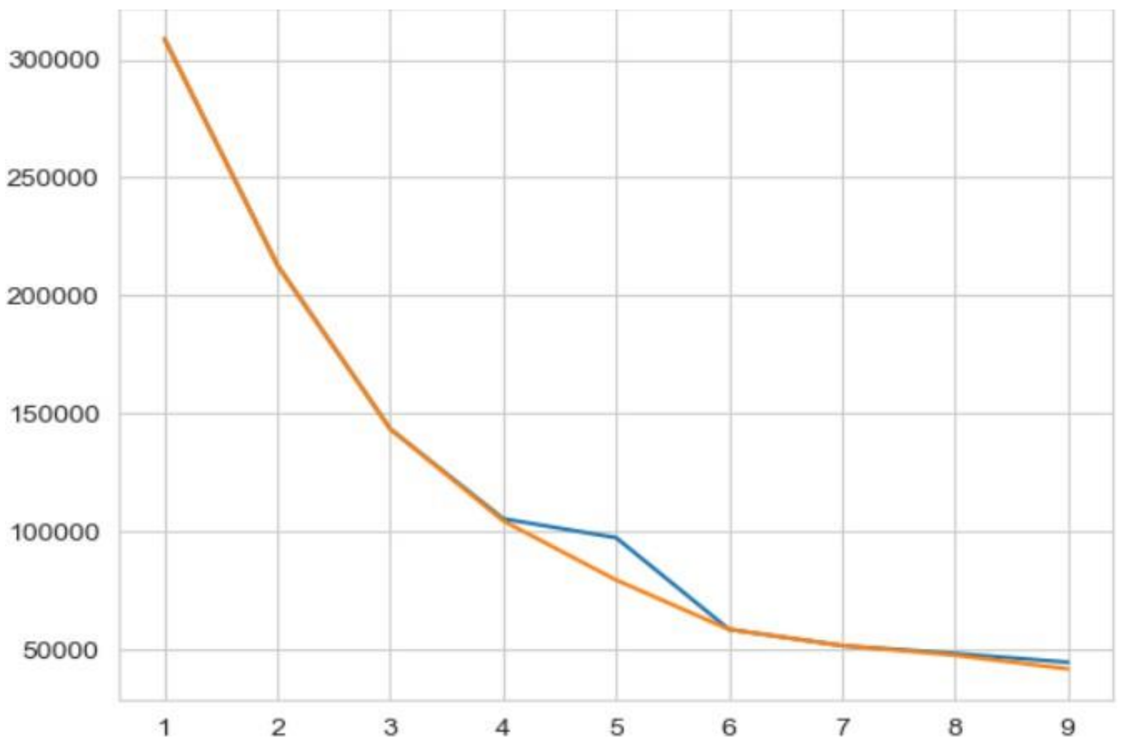


```
C:\Users\vinot\AppData\Local\Temp\ipykernel_10052\470183701.py:2: SettingWithCop
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stab
e/indexing.html#returning-a-view-versus-a-copy
  Final['label']=model.predict(features)
```

	Annual Income (k$)	Spending Score (1-100)	label
0	15	39	3
1	15	81	4
2	16	6	3
3	16	77	4
4	17	40	3



Spending Score (1-100)
Annual Income (k$)
0
20
40
60
80
100
120



300000
250000
200000
150000
100000
50000
1
2
3
4
5
6
7
8
9

// Karthik Subramanian S

//240701233

// 21.10.2025

// Title : “T-test”

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 (μ� = 70) at

5% significance level using python

Python Code:

```
import numpy as np

from scipy import stats


marks = np.array([72, 68, 75, 70, 74, 69, 71, 73, 70, 72])

mu_0 = 70

t_stat, p_value = stats.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 different 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.
```

//Karthik Subramanian S

//240701233

//21.10.2025

//Title : “Z – test”

Z-test

A manufacturer claims that the average weight of packets is 50 g.A random sample of 36 packets has

an average weight of 51.2 g with a known σ = 3 g.At a 5% significance level, test the claim.

Python Code:

```
import numpy as np

from math import sqrt

from scipy.stats import norm


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 different from 50 g.")

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 50 g.
```

// Karthik Subramanian S

//240701233

//21.10.2025

//Title: “Anova test”

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

Python Code:

```
import numpy as np

from scipy import stats


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.
```