```
#EX.NO:1.a        BasicPracticeExperiments(1to4)
#DATA : 30.07.2024

#NAME :Ganesan G

#ROLL NO : 230701089
#DEPARTMENT : B.E COMPUTER SCIENCE AND ENGINEERING - B
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

data=pd.read_csv('Iris.csv')
data
```

```
      Id SepalLengthCm SepalWidthCm PetalLengthCm PetalWidthCm\
0      1           5.1          3.5           1.4          0.2
1      2           4.9          3.0           1.4          0.2
2      3           4.7          3.2           1.3          0.2
3      4           4.6          3.1           1.5          0.2
4      5           5.0          3.6           1.4          0.2
..   ...           ...          ...           ...          ...
145  146           6.7          3.0           5.2          2.3
146  147           6.3          2.5           5.0          1.9
147  148           6.5          3.0           5.2          2.0
148  149           6.2          3.4           5.4          2.3
149  150           5.9          3.0           5.1          1.8

Species
  Iris-setosa
  Iris-setosa
  Iris-setosa
  Iris-setosa
  Iris-setosa
..    ...
  Iris-virginica
  Iris-virginica
  Iris-virginica
  Iris-virginica
  Iris-virginica

[150 rows x 6 columns]
```

```
data.info()
```

```
#    Column           Non-NullCountDtyp
                                      e
-----                                 --------
<class
'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
    -------           ---------------
```

```
0     Id    150 non-null     int64
· SepalLengthCm 150 non-null        float64
· SepalWidthCm   150 non-null       float64
· PetalLengthCm 150 non-null        float64
· PetalWidthCm   150 non-null       float64
5    Species          150   non-null
object dtypes: float64(4), int64(1),
object(1) memory usage: 7.2+ KB
```

data.describe()

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

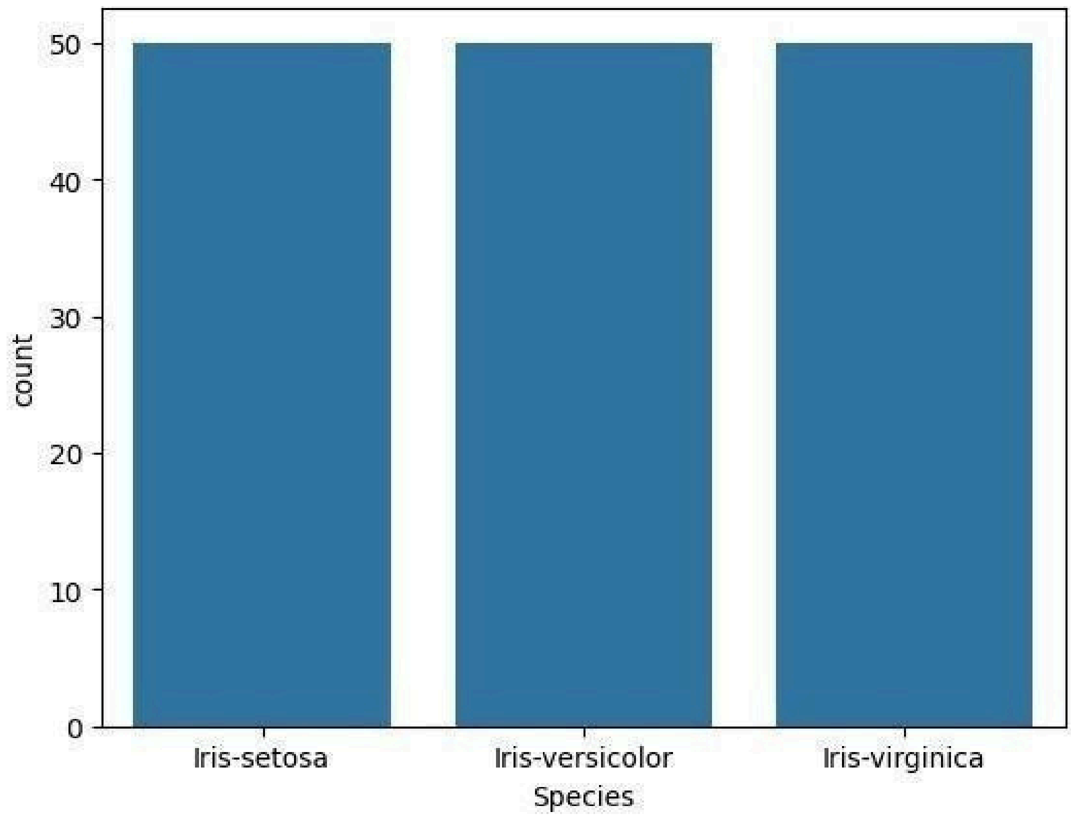
```
Iris-versicolor    50
Iris-virginica
```

data.value_counts('Species')

Species

Name: count, dtype: int64

```
sns.countplot(x='Species',data=data,)
plt.show()
```

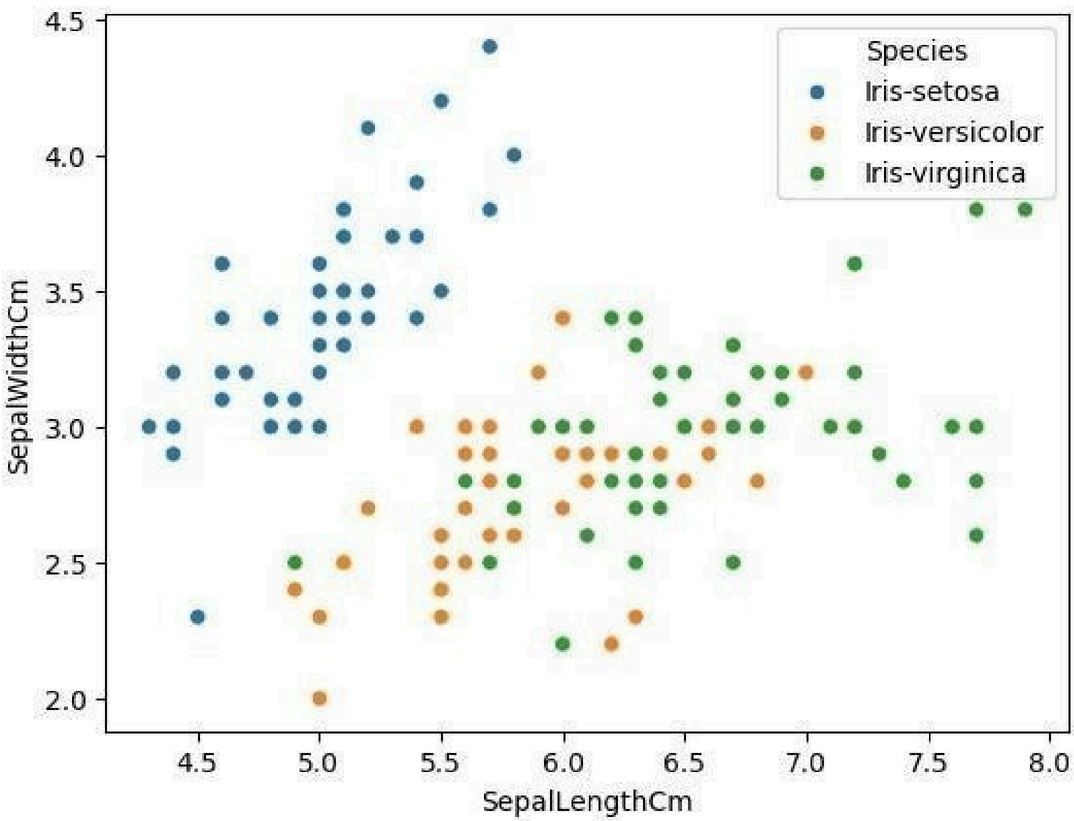



```
dummies=pd.get_dummies(data.Species)

FinalDataset=pd.concat([pd.get_dummies(data.Species),data.iloc[:,
[0,1,2,3]]],axis=1)

FinalDataset.head()
```

```
   Iris-setosa Iris-versicolor Iris-virginica Id     SepalLengthC  \
                                                                m
.  True      False False 1                                        5.1
.  True      False False 2                                        4.9
.  True      False False 3                                        4.7
.  True      False False 4                                        4.6
.                                                                 5.0
   SepalWidthCm PetalLengthCm
0     3.5   1.4
1     3.0   1.4
2     3.2   1.3
3     3.1   1.5
4     3.6   1.4
```

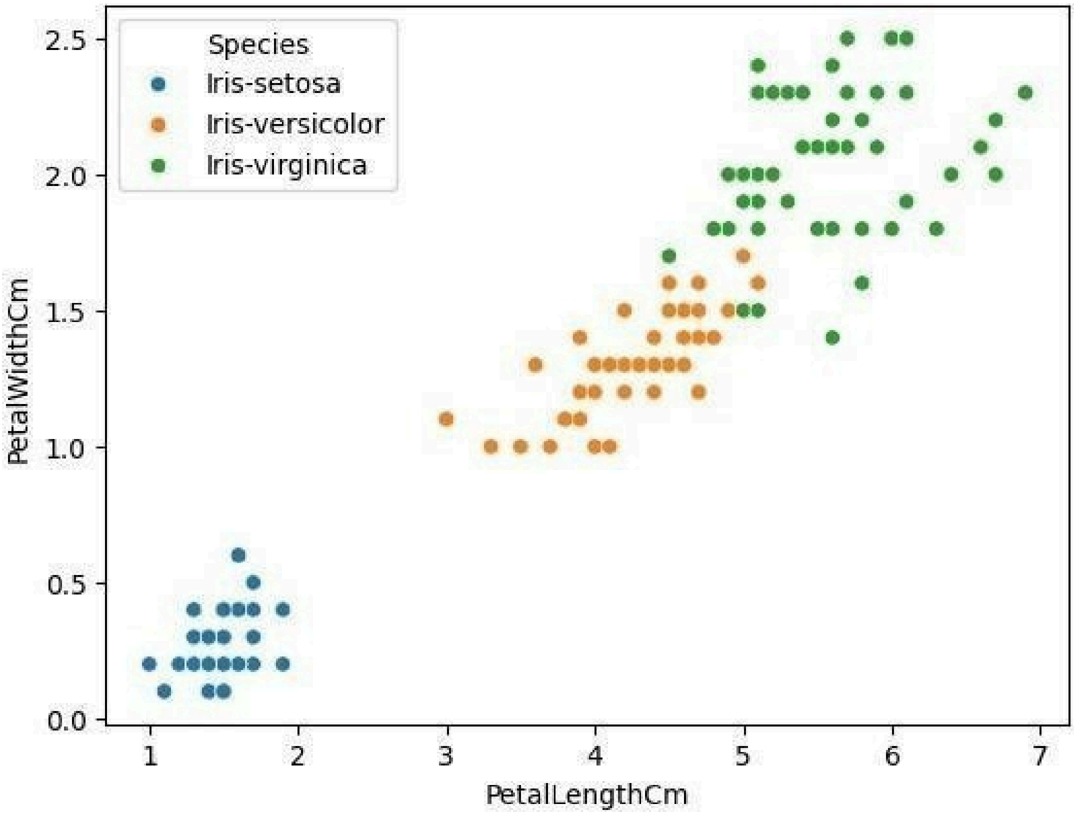
```
sns.scatterplot(x='SepalLengthCm',y='SepalWidthCm',hue='Species',data=
data,)
```

```
<Axes: xlabel='SepalLengthCm', ylabel='SepalWidthCm'>
```

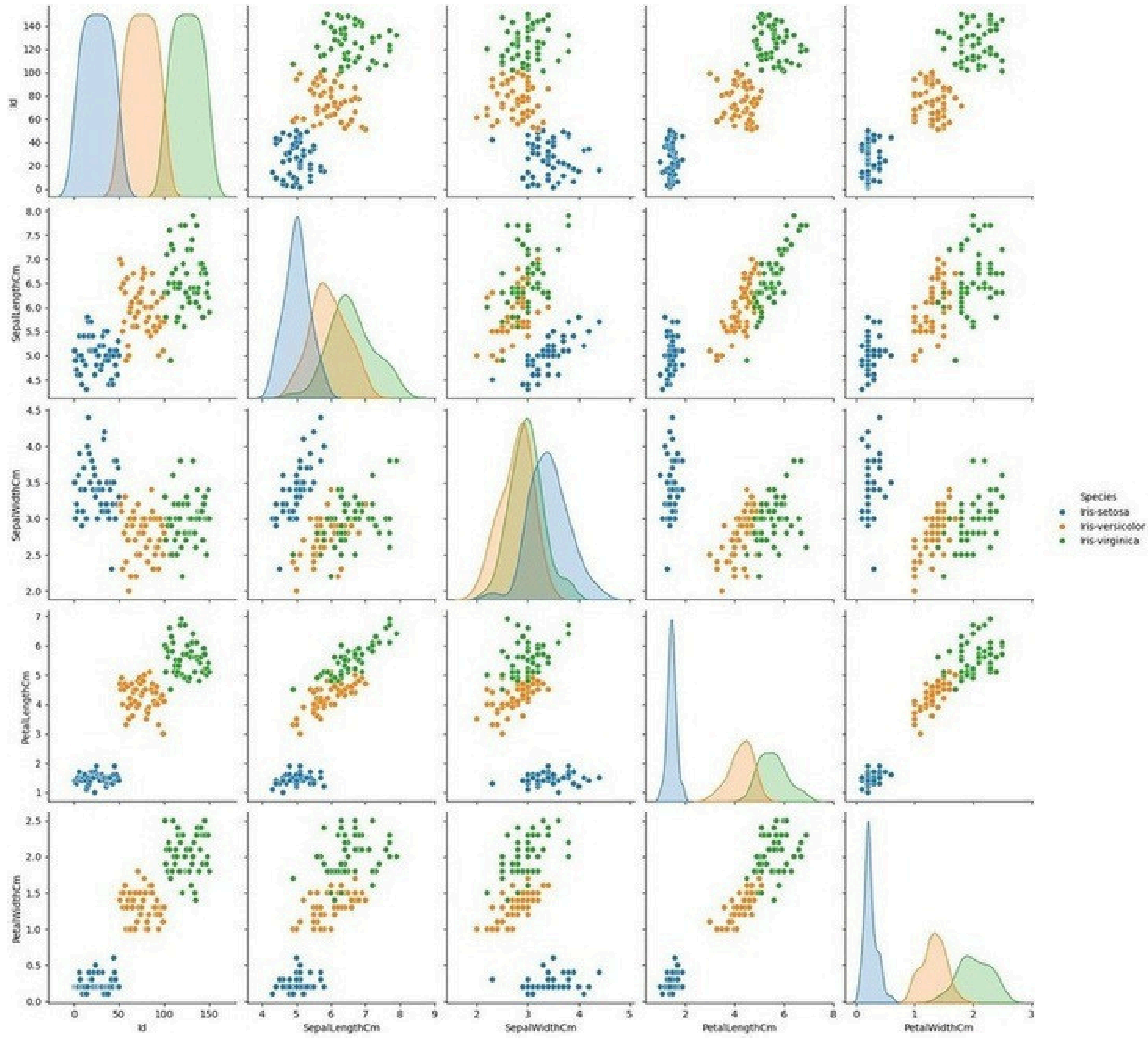



```
sns.scatterplot(x='PetalLengthCm',y='PetalWidthCm',hue='Species',data=
 data,)
```

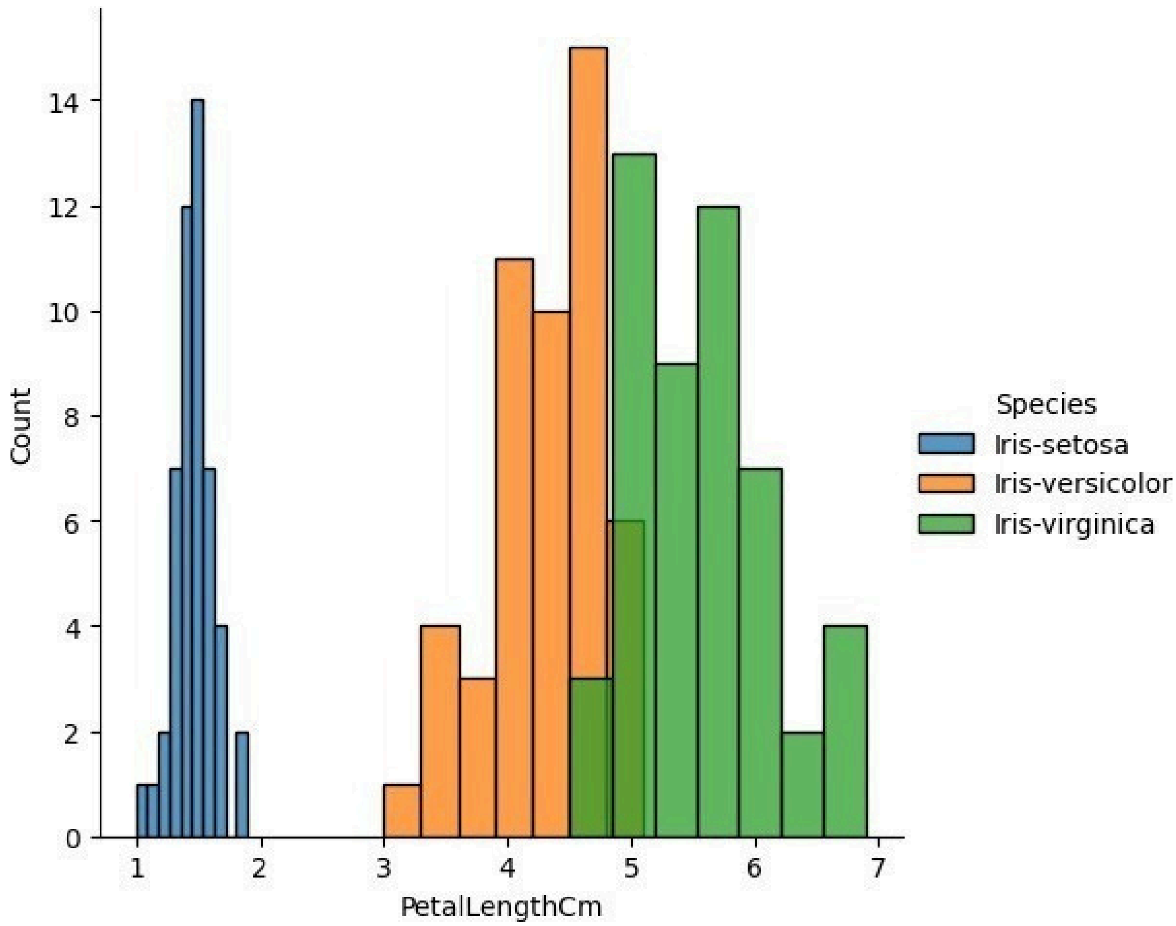
```
<Axes: xlabel='PetalLengthCm', ylabel='PetalWidthCm'>
```



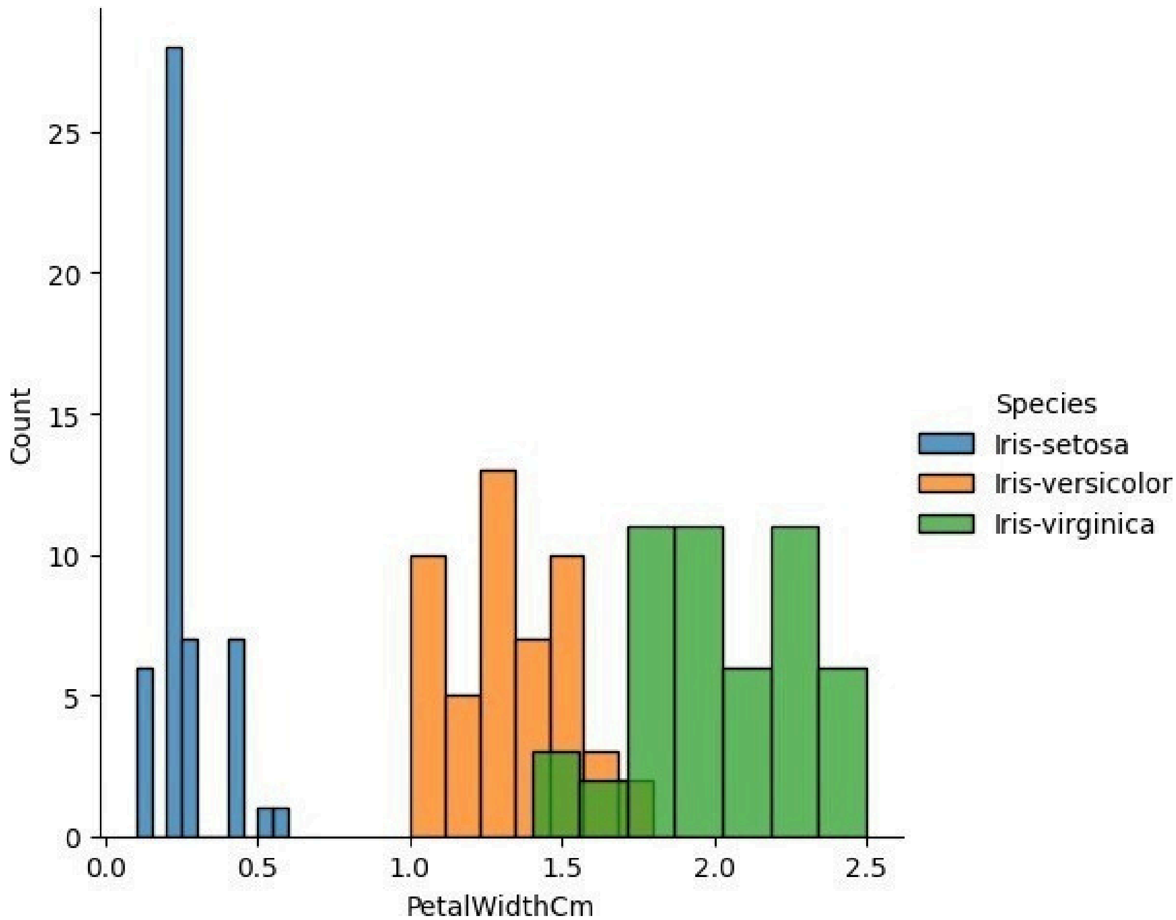

```
sns.pairplot(data,hue='Species',height=3);
```



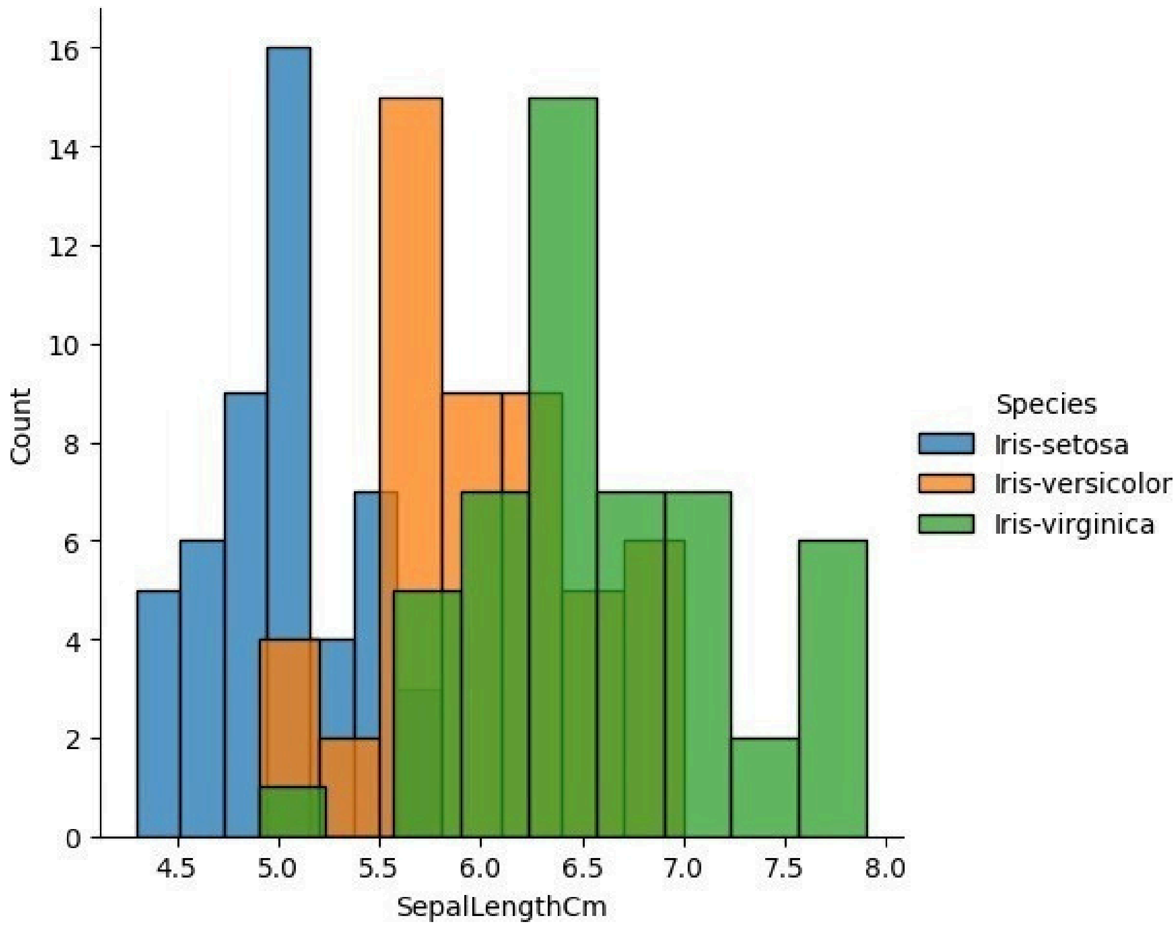

```
plt.show()

sns.FacetGrid(data,hue='Species',height=5).map(sns.histplot,'PetalLeng
thCm').add_legend();
plt.show();
```



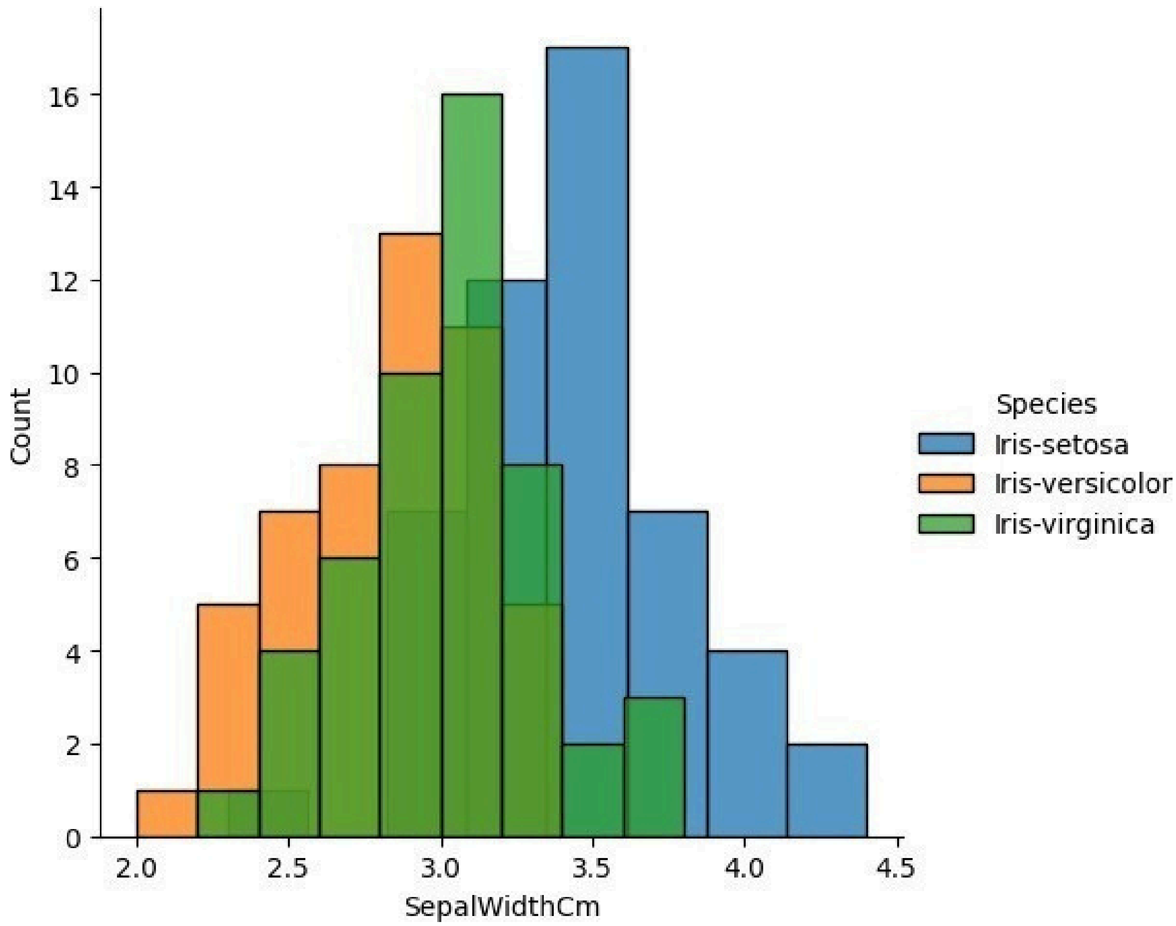

```
sns.FacetGrid(data,hue='Species',height=5).map(sns.histplot,'PetalWidt
hCm').add_legend();
plt.show();
```




```
sns.FacetGrid(data,hue='Species',height=5).map(sns.histplot,'SepalLeng
 thCm').add_legend();
plt.show();
```




```
sns.FacetGrid(data,hue='Species',height=5).map(sns.histplot,'SepalWidt
 hCm').add_legend();
plt.show();
```




```
#EX.NO :1.b Pandas Buit in function. Numpy Buit in fuction- Array
slicing, Ravel,Reshape,ndim
#DATA : 06.08.2024
#NAME :Ganesan G
#ROLL NO : 230701089
#DEPARTMENT : B.E COMPUTER SCIENCE AND ENGINEERING - B

import numpy as np
array=np.random.randint(1,100,9)
array

array([39, 97, 88, 58, 29, 87, 27, 88, 91])

np.sqrt(array)

array([6.244998 , 9.8488578 , 9.38083152, 7.61577311, 5.38516481,
 9.32737905, 5.19615242, 9.38083152, 9.53939201])

array.ndim
```

```
1
new_array=array.reshape(3,3)
new_array
```

```
     array([[39,  97,
                 88],
       [58,  29,  87],
[27,  88,  91]])
```

```
new_array.ndim
```

```
2
```

```
new_array.ravel()
```

```
array([39, 97, 88, 58, 29, 87, 27, 88, 91])
```

```
newm=new_array.reshape(3,3)
newm
```

```
     array([[39,  97,
                 88],
       [58,  29,  87],
[27,  88,  91]])
```

```
newm[2,1:3]
```

```
array([88, 91])
```

```
newm[1:2,1:3]
```

```
array([[29, 87]])
```

```
new_array[0:3,0:0]
```

```
array([], shape=(3, 0), dtype=int32)
```

```
new_array[1:3]
```

```
array([[58, 29, 87],
[27, 88, 91]])
```

```
#EX.NO :2 Outlier detection #DATA : 13.08.2024
#NAME : Ganesan
#ROLL NO : 230701089
#DEPARTMENT : B.E COMPUTER SCIENCE AND ENGINEERING - B
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

```
array=np.random.randint(1,100,16)
 array
```

```
array([37, 15, 49, 89, 30, 47, 2, 86, 53, 63, 41, 46, 42, 27, 5,
97])
```

```
array.mean()
```

```
 45.5625
```

```
np.percentile(array,25)
```

```
 29.25
```

```
np.percentile(array,50)
```

```
 44.0
```

```
np.percentile(array,75)
```

```
 55.5
```

```
np.percentile(array,100)
```

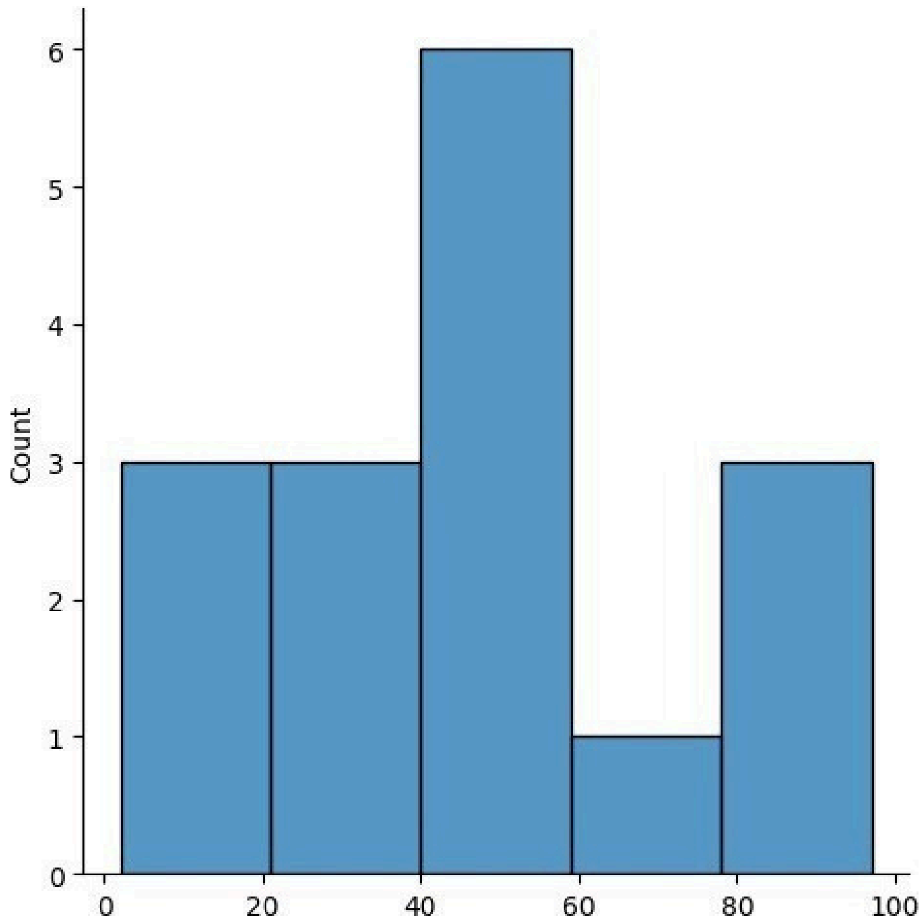
```
 97.0
```

```
 #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
```

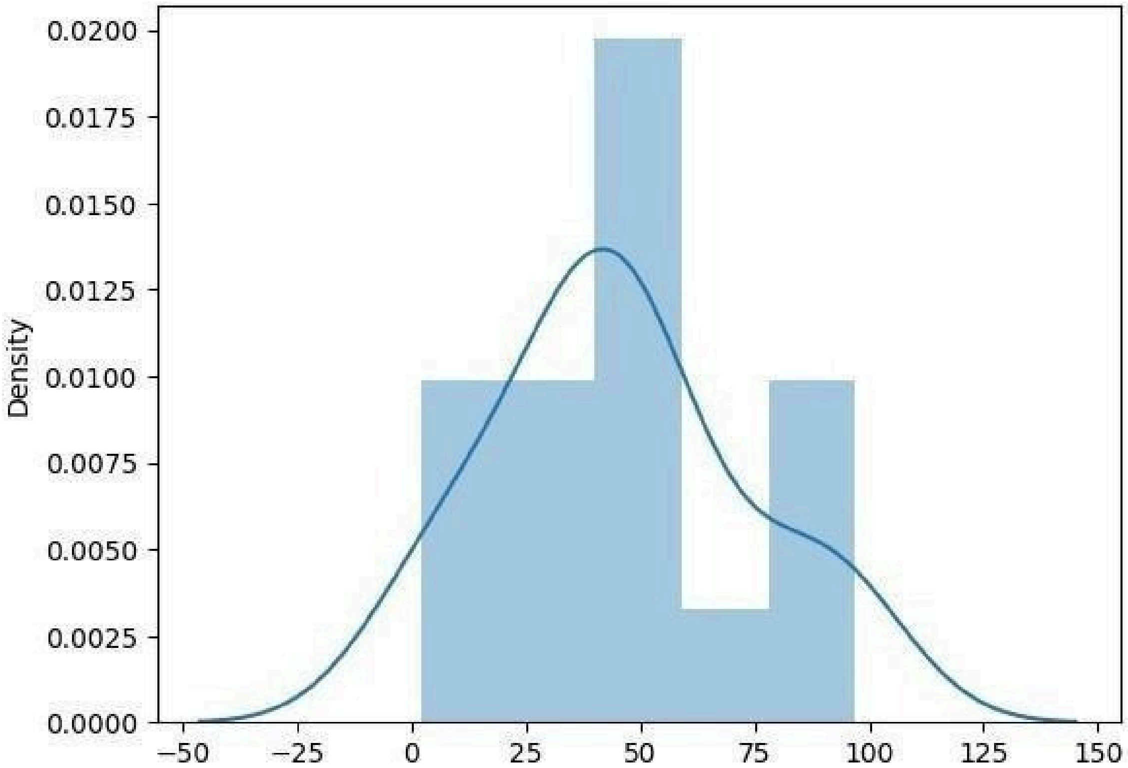
```
(-10.125, 94.875)
```

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

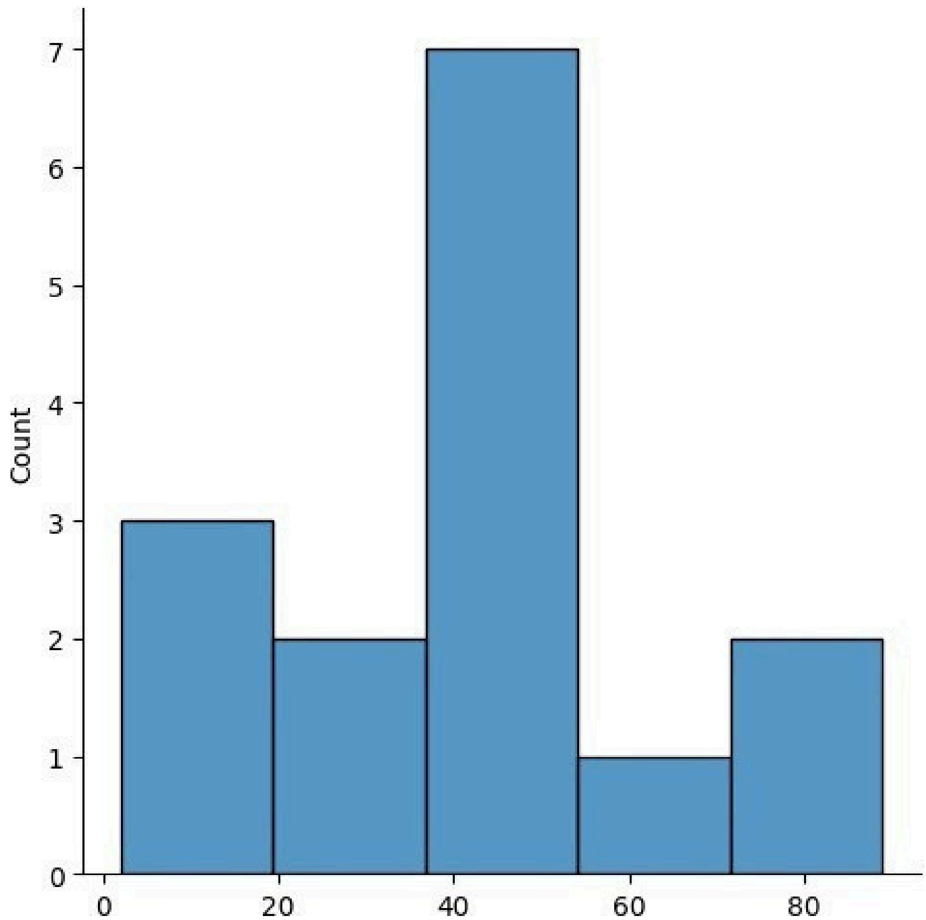
```
<seaborn.axisgrid.FacetGrid at 0x20d7cda3b50>
```




```
sns.distplot(array)

<Axes: ylabel='Density'>
```




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

array([37, 15, 49, 89, 30, 47, 2, 86, 53, 63, 41, 46, 42, 27, 5])

sns.displot(new_array)

<seaborn.axisgrid.FacetGrid at 0x20d7d02d950>
```




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

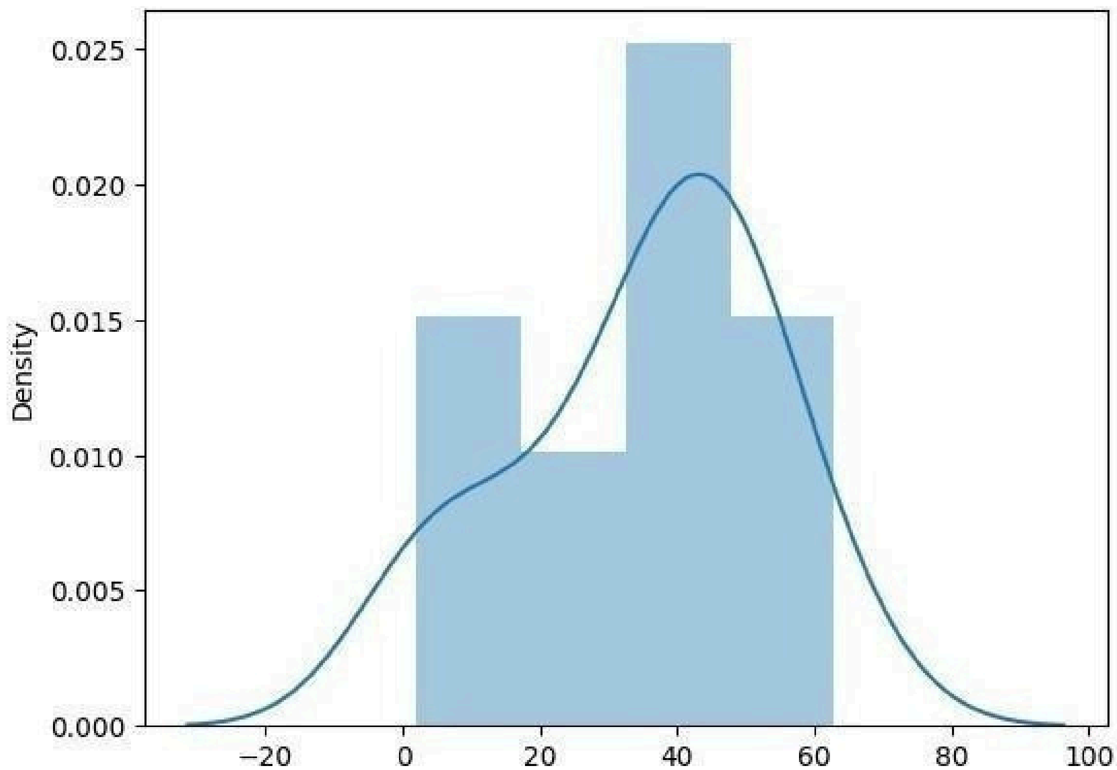
```
(-5.25, 84.75)
```

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

```
array([37, 15, 49, 30, 47, 2, 53, 63, 41, 46, 42, 27, 5])
```

```
sns.distplot(final_array)
```

```
<Axes: ylabel='Density'>
```




```
#EX.NO :3 Missing and inappropriate data
#DATA : 20.08.2024
```

#NAME : Ganesan G #ROLL NO : 230701089#DEPARTMENT : B.E COMPUTER SCIENCE AND ENGINEERING - B

```
import numpy as np
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
df=pd.read_csv("Hotel_Dataset.csv")
df
```

```
    CustomerID Age_GroupRating(1-5)        Hotel FoodPreference   Bill
\
0            1       20-25          4         Ibis            veg   1300
1            2       30-35          5    LemonTree        Non-Veg2000
2            3       25-30          6       RedFox            Veg   1322
3            4       20-25         -1    LemonTree              Veg1234
```

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

```
       NoOfPax EstimatedSalaryAge_Group.1
0  1       2         40000      20-25
2  3       3         59000      30-35
4  5       2         30000      25-30
6  7       2        120000      20-25
8  9       2         45000        35+
10         2        122220        35+
            -1       21122        35+
           -10      345673      20-25
             3      -99999      25-30
             3      -99999      25-30
           4         87777      30-35
```

df.duplicated()

```
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-NullCount    Dtype
---   -------       ---------------  -----
 0    CustomerID    11non-null       int64
 1    Age_Group     11 non-null      object
 2    Rating(1-5)   11 non-null      int64
```

```
· Hotel          11 non-null     object
· FoodPreference 11 non-null     int64
· Bill           11 non-null     object
· NoOfPax        11 non-null     int64
· EstimatedSalary 11 non-null    int64
·8   Age_Group.1    11  non-null
object dtypes: int64(5), object(4)
memory usage: 924.0+ bytes
```

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

	CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill \
0	1	20-25	4	Ibis	veg	1300
1	2	30-35	5	LemonTree	Non-Veg	2000
2	3	25-30	6	RedFox	Veg	1322
3	4	20-25	-	LemonTree	Veg	1234
4	5	35	1	Ibis	Vegetarian	989
5	6	+	3	Ibys	Non-Veg	1909
6	7	35+	3	RedFox	Vegetarian	1000
7	8	20+-25	4	LemonTree	Veg	2999
8	9	25-30	7	Ibis	Non-Veg	3456
10	10	30-35	2	RedFox	non-Veg	6755

5

	NoOfPax	EstimatedSalary	Age_Group.1
0	1 2	40000	20-25
2	3 3	59000	30-35
4	5 2	30000	25-30
6	7 2	120000	20-25
8	0 2	45000	35+
1	2	122220	35+
	- 1	21122	35+
10	-10	345673	20-25
	3	-99999	25-30
	4	87777	30-35

```
len(df)
```

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

array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

```
df
```

```
    CustomerIDAge_GroupRating(1-5)        Hotel  FoodPreference Bill
NoOfPax \
0            1      20-2            4       Ibis               veg 1300
                       5
2                    30-3
31                     5
2            2                      5  LemonTree          Non-Veg 2000
2

23           3       2503           6      RedFox              Veg   1322
4                   20-2
2
5            4                     -1  LemonTree              Veg   1234
2
6                      5
-1
7            5       35+            3       Ibis      Vegetarian      989
-10          6       35+            3
8            7       35+            4       Ibys        Non-Veg 1909
9            8                      7
                                           RedFox      Vegetarian 1000
4
  3
                    20-2               LemonTree      Veg 2999
                       5

             9      25-3            2       Ibis          Non-Veg 3456
                       0
            10      30-3            5      RedFox         non-Veg-6755
                       5
```

```
      EstimatedSalar Age_Group.1
                   y
0 1 2 3        40000        20-25
4 5 6 7        59000        30-35
8 9            30000        25-30
               120000       20-25
                45000         35+
               122220         35+
                21122         35+
              345673        20-25
df.drop(['Age_-G9r9o9u9p9. 1'],axis=215,in-3p0lace=True
               87777        30-35
```

```
)   df
```

```
0            1      20-25           4       Ibis              veg 130
                                                                    0
```

```
2

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

31 2 2	2	30-35	5	LemonTree	Non-Veg	2000
3 2 4 2	3	25-30	6	RedFox	Veg	1322
5 2 6 -1	4	20-25	-1	LemonTree	Veg	1234
7 -10	5	35	3	Ibis	Vegetarian	989
8 3 9 4	6	+	3	Ibys	Non-Veg	1909
	7	35	4	RedFox	Vegetarian	1000
	8	20+-2	7	LemonTree	Veg	2999
	9	5 253-	2	Ibis	Non-Veg	3456
	10	350	5	RedFox	non-Veg	-6755
		30+-3				
		5				

```
EstimatedSalary
0     40000
1     59000
2     30000
3     120000
4     45000
5     122220
6     21122
7     345673
8     -99999
9     87777
```

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

	CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill
\						
0	1.0	20-25	4	Ibis	veg	1300.0
1	2.0	30-35	5	LemonTree	Non-Veg	2000.
2	3.0	25-30	6	RedFox	Veg	0
3	4.0	20-25	-1	LemonTree	Veg	1232342..0

4	5.0	35	3	Ibis	Vegetarian	989.0
5	6. 0	+ 35	3	Ibys	Non-Veg	1909.0
6	7.0	+35+	4	RedFox	Vegetarian	1000.0
7	8.0	20-25	7	LemonTree	Veg	2999.0
8	9.0	25-30	2	Ibis	Non-Veg	3456.0
9	10.0	30-35	5	RedFox	non-Veg	NaN

	NoOfPax	EstimatedSalary
0	2	40000.0
1	3	59000.0
2	2	30000.0
3	2	120000.0
4	2	45000.0
5	2	122220.0
6	-1	21122.0
7	-10	345673.0
8	3	NaN
9	4	87777.0

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

	CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill \
0	1.0	20-25	4	Ibis	veg	1300.0
1	2.0	30-35	5	LemonTree	Non-Veg	2000.0
2	3.0	25-30	6	RedFox	Veg	1322.0
3	4.0					
4	5.0	20 35-52+5	-1	LemonTree	VeNgoeentarViea Veg g	0089930900
5	6.0				tlaonVVeeggNeoagn	3209095.06..00
6	7.0		3 43 7 2	RedFIbbyoisxs	-	
7	8.0	3355+				
8	9.0	2250--32+05				
				LemonTree		
				Ibis		

```
9            10.0        30-35                 5        RedFox           non-Veg       NaN
```

```
       NoOfPax EstimatedSalary
0          2.0                40000.0
1          3.0                59000.0
2          2.0                30000.0
```

```
3          2.0              120000.0
4          2.0                45000.0
5          2.0              122220.0
6          NaN                21122.0
7          NaN              345673.0
8          3.0                    NaN
9          4.0                87777.0
```

```
df.Age_Group.unique()
```

```
array(['20-25', '30-35', '25-30', '35+'], dtype=object)
```

```
df.Hotel.unique()
```

```
array(['Ibis', 'LemonTree', 'RedFox', 'Ibys'], dtype=object)
```

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

```
<boundmethodSeries.unique
                                        of0                 veg
9          g
           Non-Ve
           g
           non-Ve
           g

Name:                 FoodPreference,                  dtype:                 object>
```

```
df.FoodPreference.replace(['Vegetarian','veg'],'Veg',inplace=Tru
e) df.FoodPreference.replace(['non-Veg'],'Non-Veg',inplace=True)
df.EstimatedSalary.fillna(round(df.EstimatedSalary.mean()),inplace=T

ru  e)
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  \
0         1.0     20-25           4         Ibis            Veg   1300.0
1         2.0     30-35           5   LemonTree        Non-Veg2000.0
2         3.0     25-30           6       RedFox            Veg   1322.0
3         4.0     20-25           -   LemonTree            Veg1234.0
                                  1
4         5.0       35            3        Ibi             Veg      989.0
5         6.      + 35+           3          s          Non-Veg 1909.0
6         0       20+-2           4     RedIbFiox     Veg 1000.0
7         7.      5               7 LemosnTree        Veg 2999.0
8         09.0    25-30           2         Ibis      Non-Veg 3456.0
9         180..0  30-35           5       RedFox      Non-Veg 1801.0
          0

   NoOfPax EstimatedSalary
0     2.0         40000.0
1     3.0         59000.0
2     2.0         30000.0
3     2.0        120000.0
4     2.0         45000.0
5     2.0        122220.0
6     2.0         21122.0
7     2.0        345673.0
8     3.0         96755.0
9     4.0         87777.0
```

```
#EX.NO :4 Data Preprocessing
#DATA : 27.08.2024
#NAME : Ganesan G
#ROLL NO : 230701089
#DEPARTMENT : B.E COMPUTER SCIENCE AND
ENGINEERING - B

import numpy as np
import pandas as pd
import warnings
wd)anf=drpfings.filterwarnings('ignore')
    d.read_csv("pre_process_datasample.csv"
```

```
   Countr   Age    Salary
0       y  44.   72000.0 Purchased    N
1  France  0     48000.0              o
2   Spain  27.0  54000.0             Ye
3  German  30.0  61000.0              s
4       y  38.0      NaN             No
5   Spain  40.0  58000               No
```

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 4 columns):
 #   Column     Non-NullCount Dtyp
     -------    --------------- e
-----                           --------
 0   Country    10  non-null    object
 ·  Age          9              float6
                non-null        4
 ·  Salary
 3   Purchased10   non-null     float6
object dtypes: float64(2), object(2)
memory usage: 452.0+ bytes

df.Country.mode()

0      France
Name: Country, dtype:
object df.Country.mode()[0]

'France'

type(df.Country.mode())

pandas.core.series.Series

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
     Countr   Age    Salary
0         y   44.   72000.0 Purchased    N
1    France   0     48000.0              o
2     Spain   27.0  54000.0             Ye
3    German   30.0  61000.0              s
4         y   38.0  63778.0             No
5     Spain   40.0  58000.0             No
6    German   35.0  52000.0             Ye
7         y   38.0  79000.0              s
8    France   48.0  83000.0             Ye
9     Spain   50.0  67000.0              s
pd.get_dummies(df.Country)
     France   37.0                      No
                                        Ye
 France Germany Spain                    s
·  True      False False                No
·  False     False True                 Ye
·  False     True  False                 s
·  False     False True
·  False     True  False
```

```
5          True False     False
6         False False      True
7          True    False     False
8        False           True False
9        True       False     False
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 4 columns):
 #    Column      Non-NullCount Dtyp
      -------     ---------------   e
-----                               --------
 0    Country     10  non-null      object
 1    Age         10  non-null      float64
 2    Salary      10  non-null      float64
 3    Purchased10   non-null
object dtypes: float64(2), object(2)
memory usage: 452.0+ bytes
```

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

```
#EX.NO :5 EDA-Quantitative and Qualitative plots
#DATA : 27.08.2024
#NAME : Ganesan G
#ROLL NO : 230701089
#DEPARTMENT : B.E COMPUTER SCIENCE AND ENGINEERING - B

import numpy as np
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
df=pd.read_csv("pre_process_datasample.csv"


)  df
```

```
    Countr   Age   Salary
0        y  44.   72000.0Purchased      N
1   France  0     48000.0               o
2    Spain  27.0  54000.0              Ye
3   German  30.0  61000.0               s
4        y  38.0      NaN              No
5    Spain  40.0  58000.               No
6   German  35.0  0                    Ye
7        y   NaN  52000.0               s
8   France  48.   79000.0              Ye
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 4 columns):
 #    Column     Non-NullCount Dtyp
      -------    --------------- e
-----                            --------
 0    Country    10  non-null    object
 ·  Age          9                float6
                 non-null        4
 ·  Salary
 3    Purchased10   non-null     float6
object dtypes: float64(2), object(2)
memory usage: 452.0+ bytes
```

```
df.Country.mode()
```

```
0      France
Name: Country, dtype:
```

```
object df.Country.mode()[0]
```

```
'France'
```

```
type(df.Country.mode())
```

```
pandas.core.series.Series
```

```
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
```

```
    Countr   Age   Salary
0        y  44.   72000.0 Purchased   N
1   France  0     48000.0             o
2    Spain  27.0  54000.0            Ye
3   German  30.0  61000.0             s
4        y  38.0  63778.0            No
5    Spain  40.0  58000.0            No
6   German  35.0  52000.0            Ye
7        y  38.0  79000.0             s
8   France  48.0  83000.0            Ye
9    Spain  50.0  67000.0             s
    France  37.0                     No
                                     Ye
                                      s
                                     No
                                     Ye
                                      s
```

```
pd.get_dummies(df.Country)
```

```
 France Germany Spain
·  True      False False
·  False     False True
·  False     True  False
·  False     False True
·  False     True  False
```

```
5          True False   False
6         False False     True
7        True    False      False
8       False          True False
9       True      False     False
```

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

```
     France Germany Spain   Age    Salary
0         True False False  44.    72000.0 Purchased     N
1    False           False  0      48000.0               o
2    True  False            27.0   54000.0              Ye
3        True False False   30.0   61000.0               s
4               False True  38.0   63778.0              No
5    False                  40.0   58000.0              No
6    True False TrueFalse   35.0   52000.0              Ye
7    False False            38.0   79000.0               s
8    False      True True   48.0   83000.0              Ye
9    False False False      50.0   67000.0               s
               True False   37.0                        No
                                                        Ye
                                                         s
                                                        No
                                                        Ye
                                                         s
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 4 columns):
 #    Column       Non-NullCount Dtype
-----  --------    ---------------- --------
 0    Country      10  non-null       object
 1    Age          10  non-null       float64
 2    Salary       10  non-null       float64
 3    Purchased10   non-null
object dtypes: float64(2), object(2)
memory usage: 452.0+ bytes
```

```
updated_dataset
```

```
     France Germany Spain   Age    Salary
0         True False False  44.    72000.0 Purchased     N
1    False           False  0      48000.0               o
2    True  False            27.0   54000.0              Ye
3        True False False   30.0   61000.0               s
4               False True  38.0   63778.0              No
5    False                  40.0   58000.0              No
6    True False TrueFalse   35.0   52000.0              Ye
7    False False            38.0   79000.0               s
8    False      True True   48.0   83000.0              Ye
```

```
#EX.NO :5 EDA-Quantitative and Qualitative plots
#DATA : 03.09.2024
```

#NAME : Ganesan G #ROLL NO : 230701089 #DEPARTMENT : B.E COMPUTER SCIENCE AND ENGINEERING - B

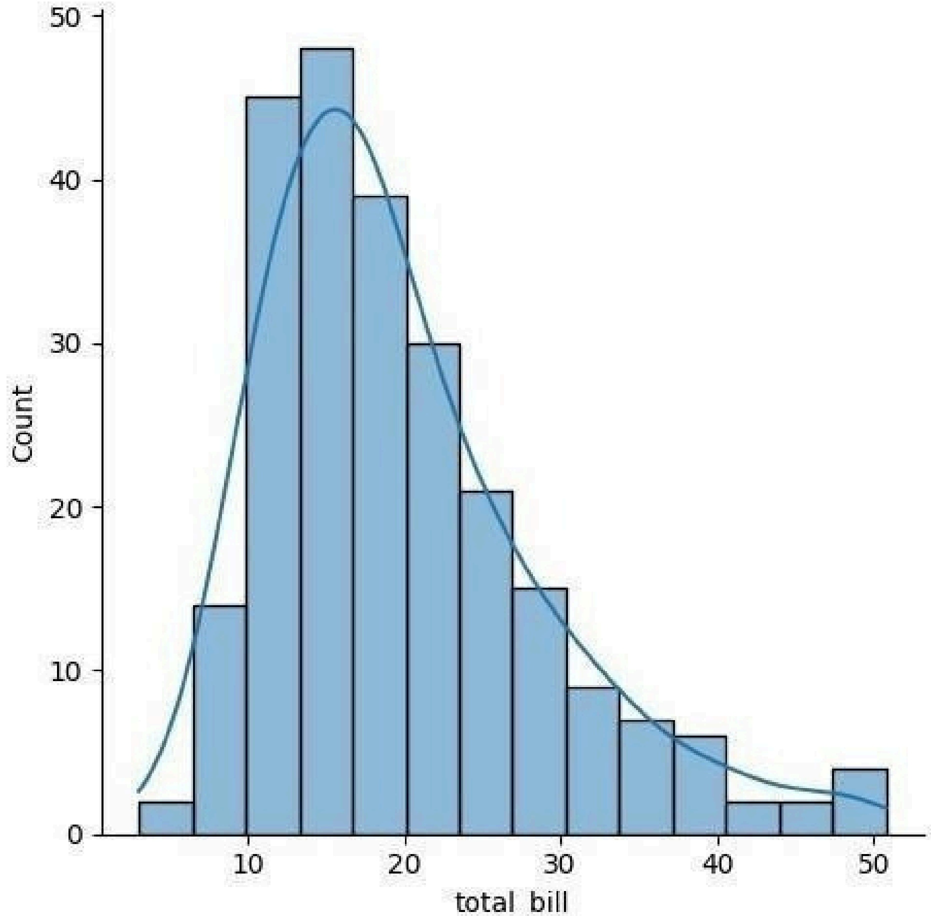
```
import seaborn as sns
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

```
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

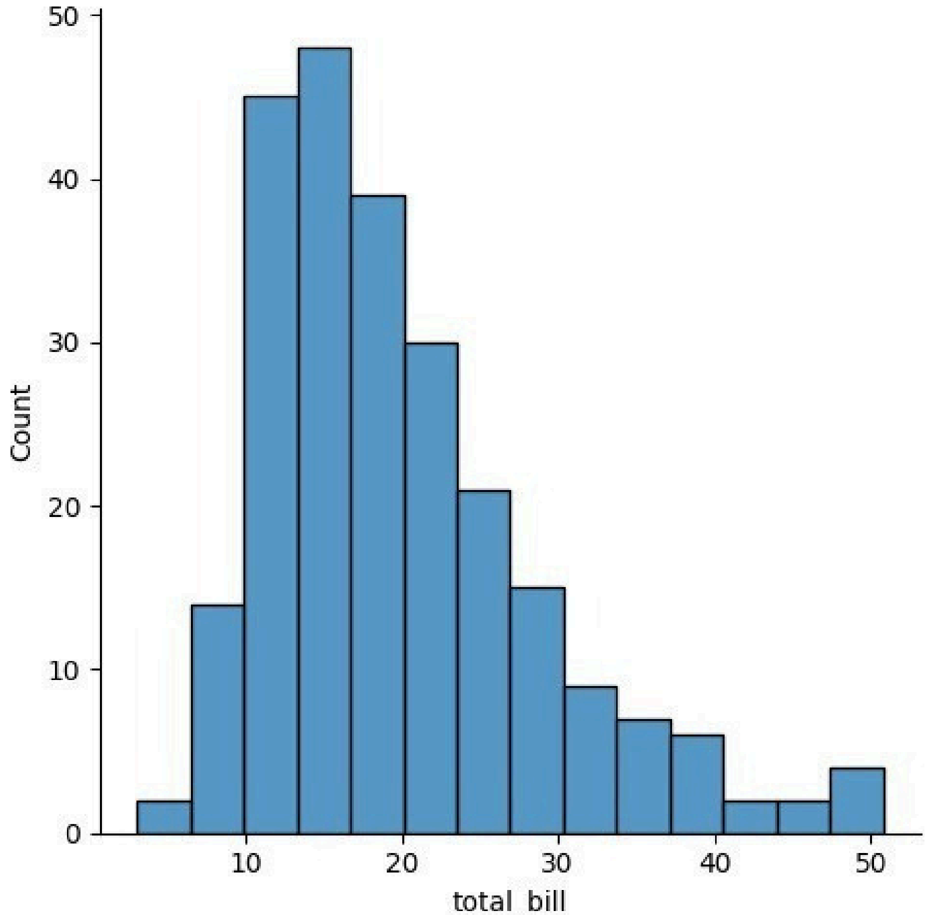
```
sns.displot(tips.total_bill,kde=True)
```

```
<seaborn.axisgrid.FacetGrid at 0x20d7dc69390>
```

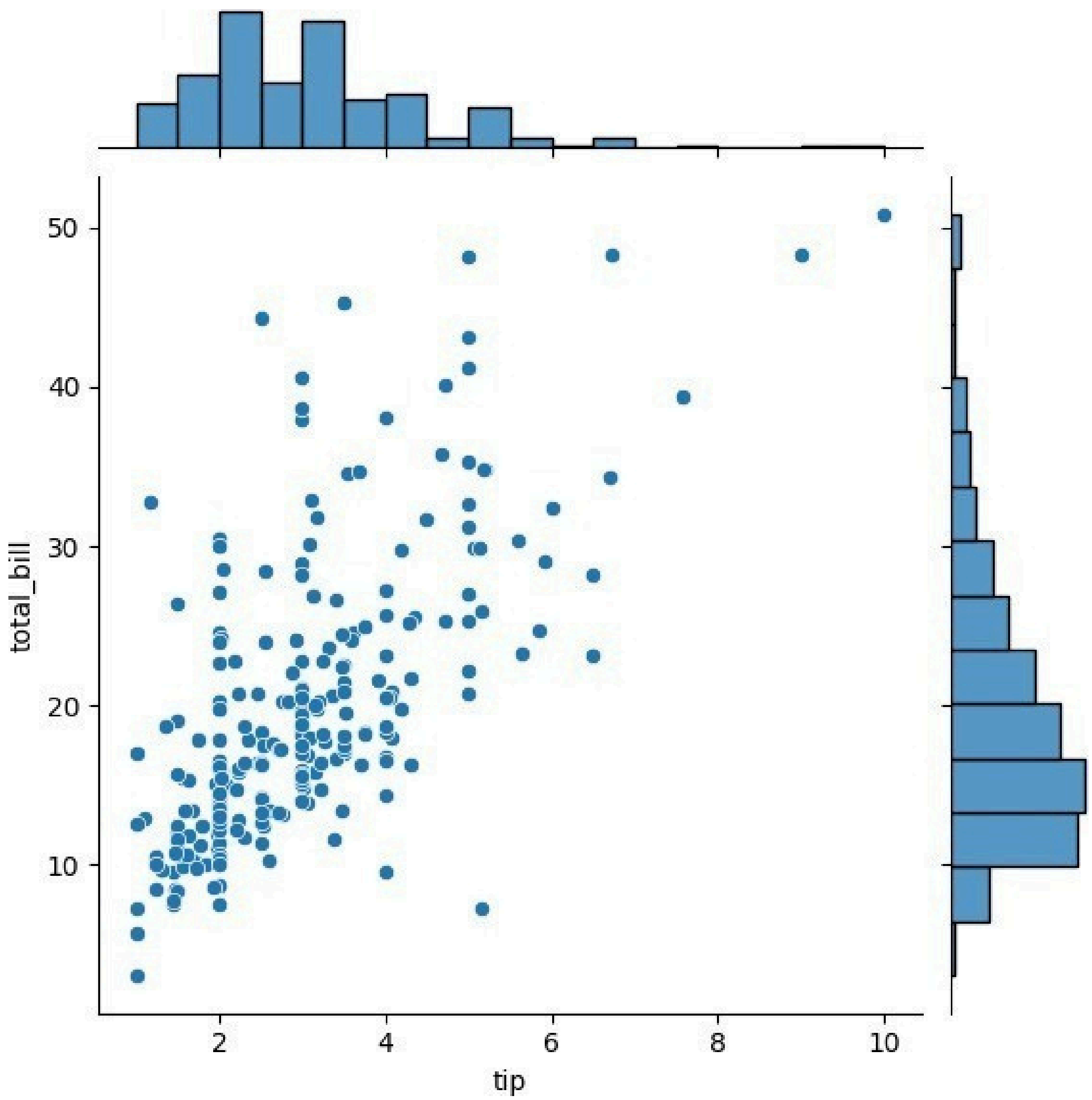



```
sns.displot(tips.total_bill,kde=False)
```

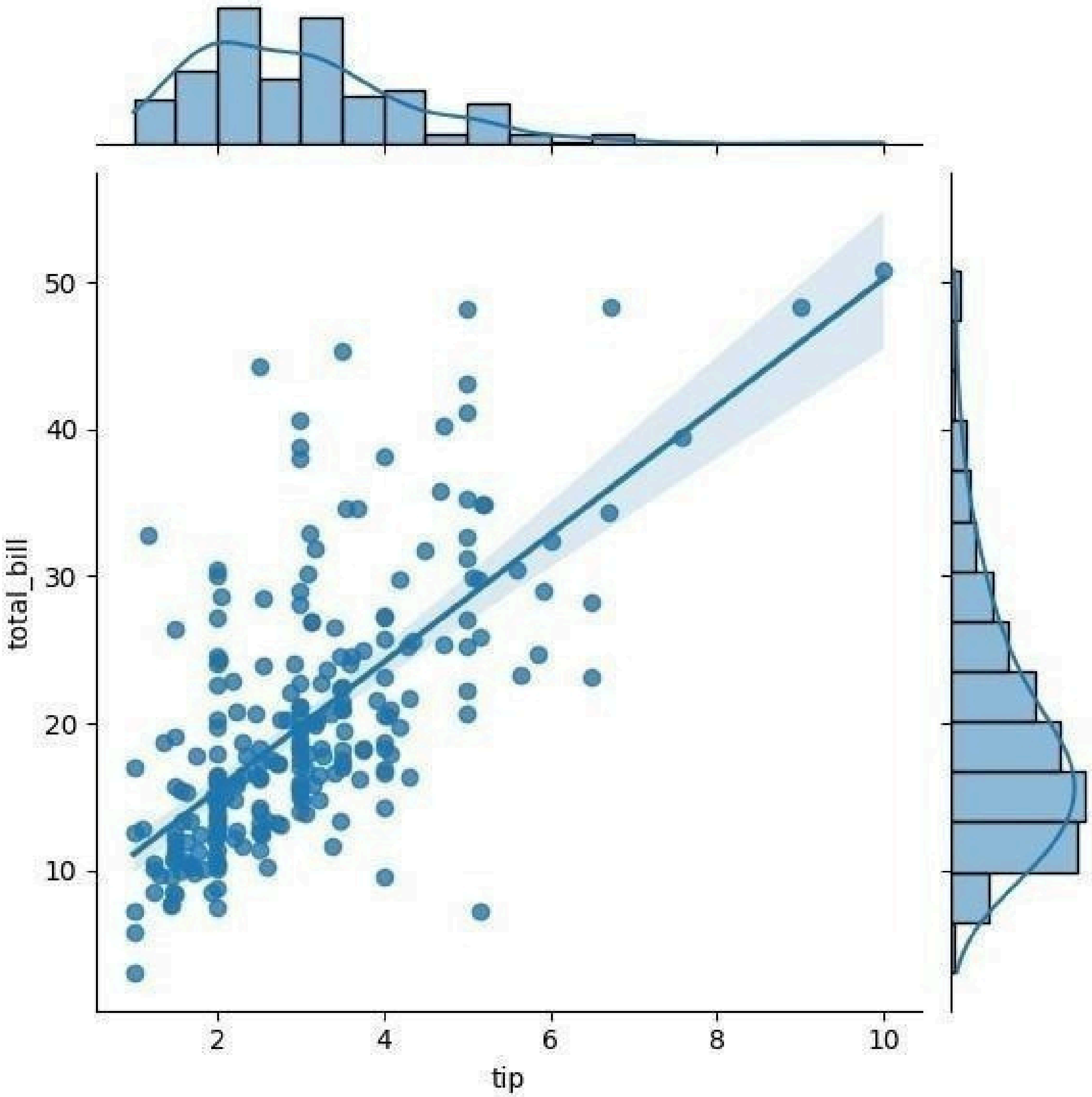
```
<seaborn.axisgrid.FacetGrid at 0x20d7dc22790>
```




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

<seaborn.axisgrid.JointGrid at 0x20d7dc2f2d0>
```

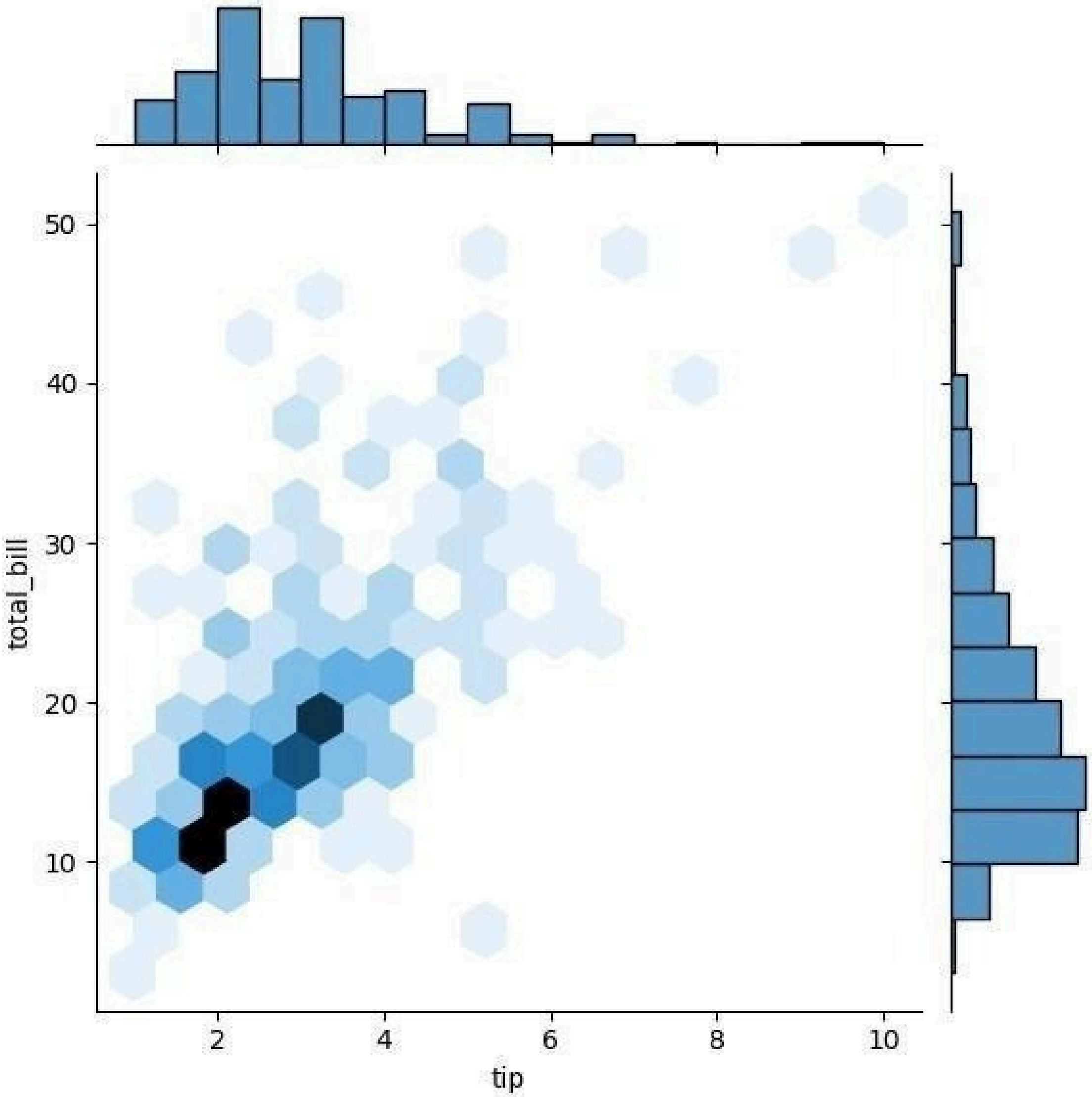



```
sns.jointplot(x=tips.tip,y=tips.total_bill,kind="reg")

<seaborn.axisgrid.JointGrid at 0x20d7ed32450>
```

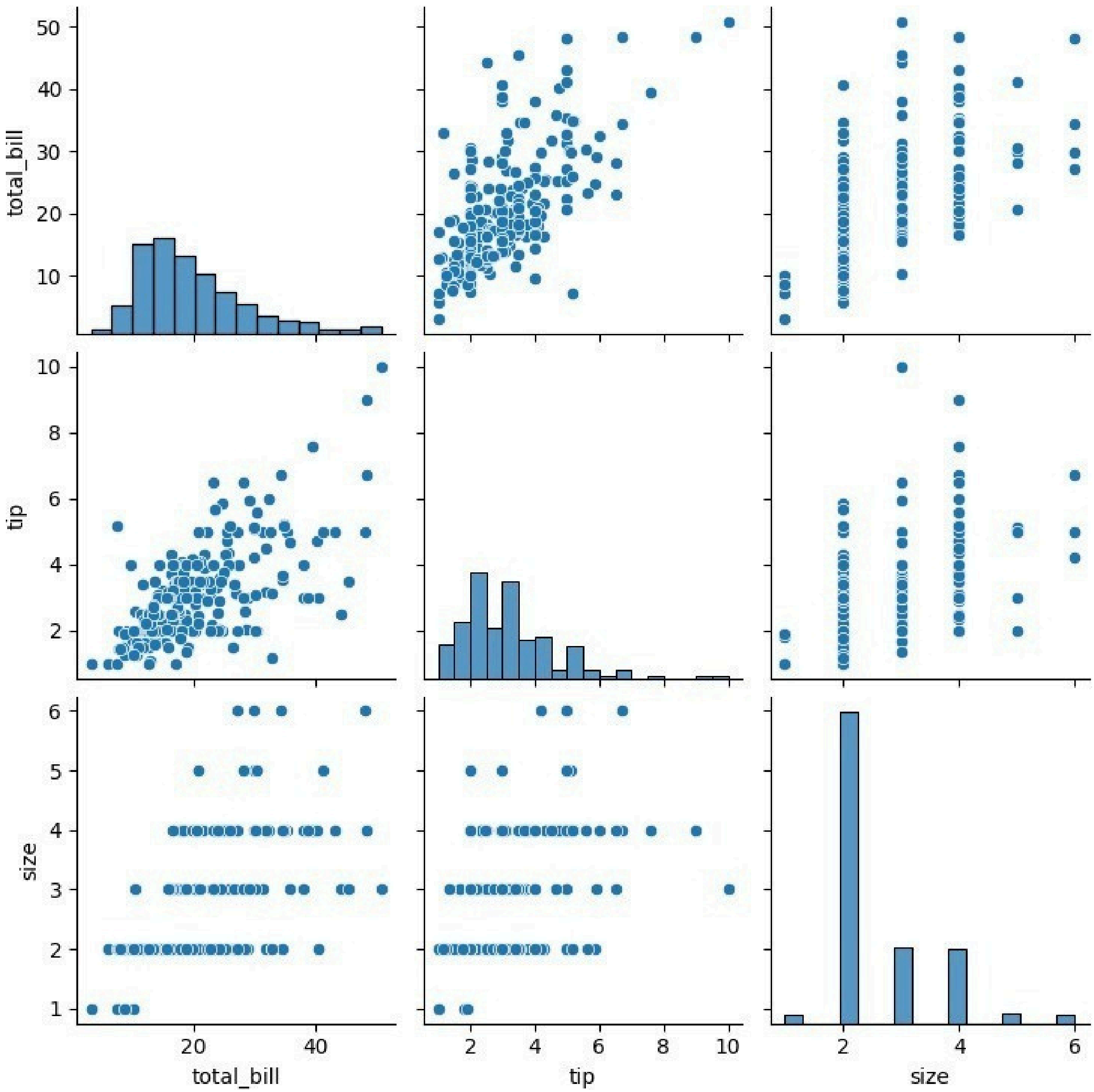



```
sns.jointplot(x=tips.tip,y=tips.total_bill,kind="hex")
```

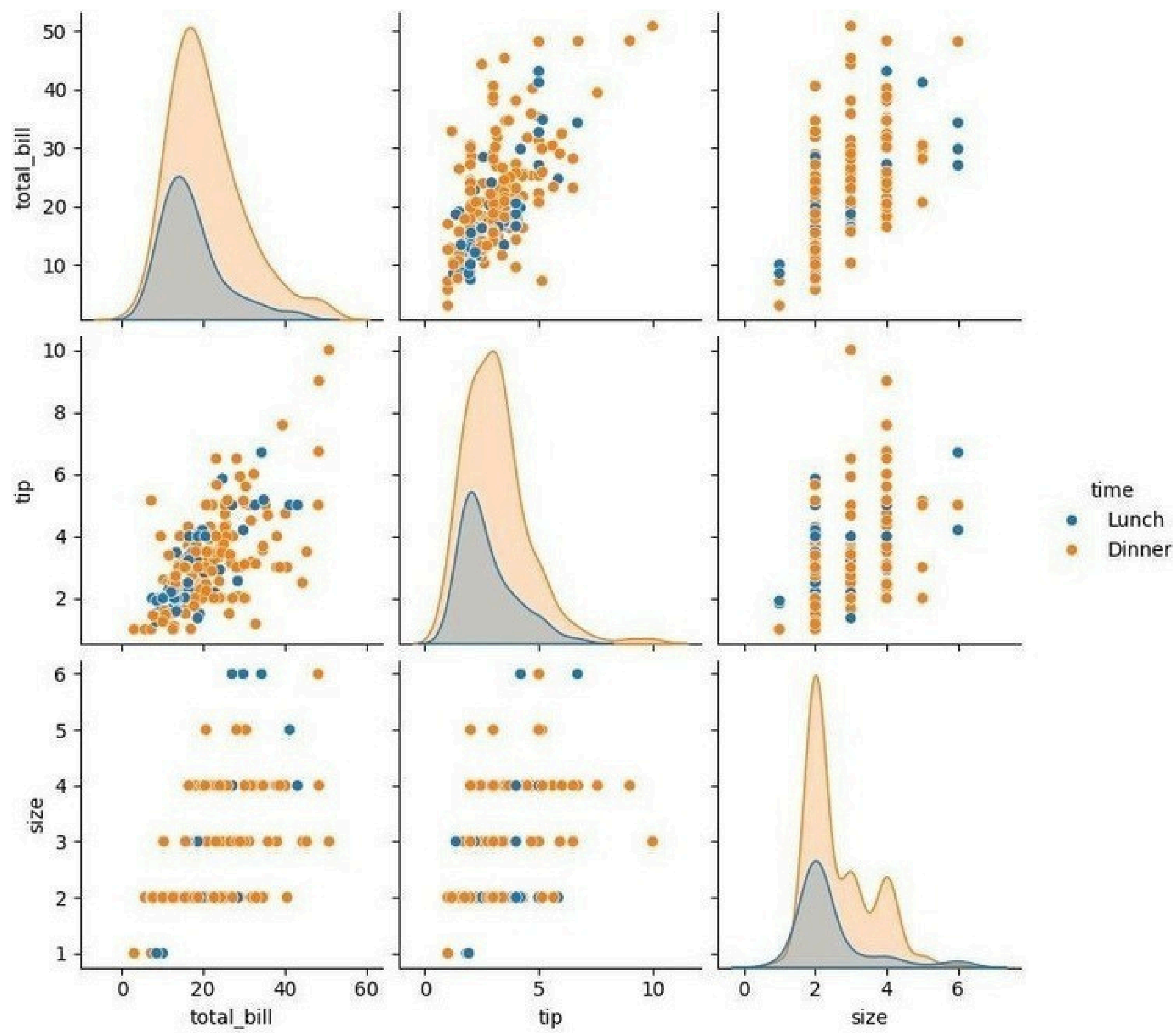
```
<seaborn.axisgrid.JointGrid at 0x20d7ed7d350>
```




```
sns.pairplot(tips)

<seaborn.axisgrid.PairGrid at 0x20d7f1c9cd0>
```

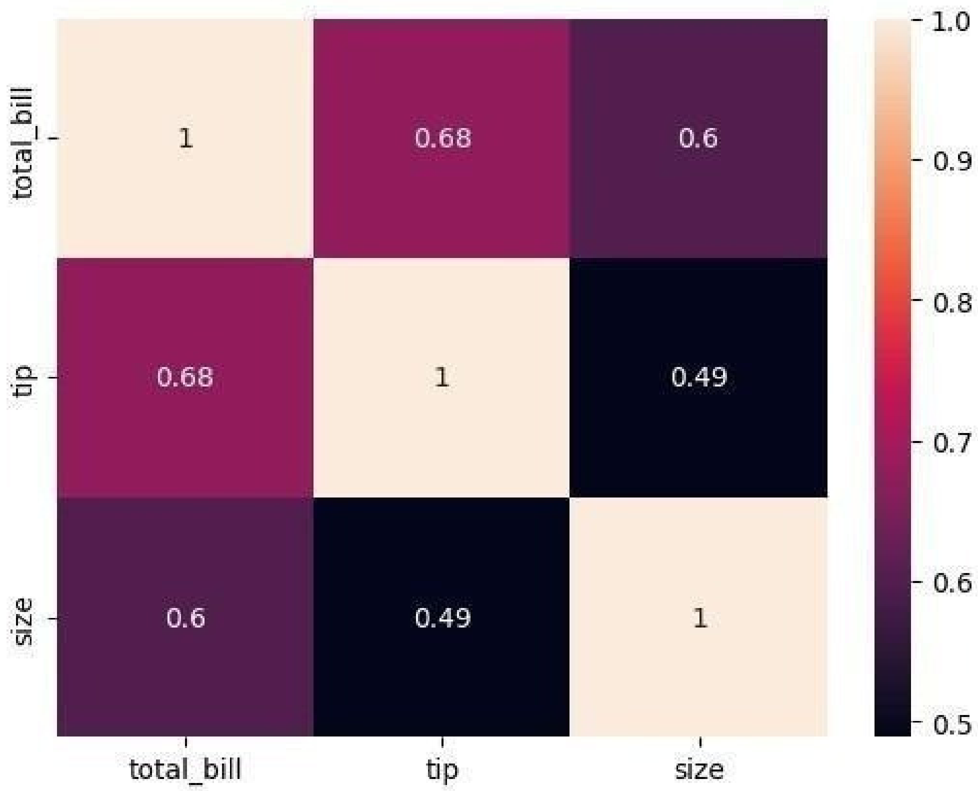



```
tips.time.value_counts()

time
Dinne      17
r           6
Name: count, dtype: int64

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

<seaborn.axisgrid.PairGrid at 0x20d7cc27990>
```



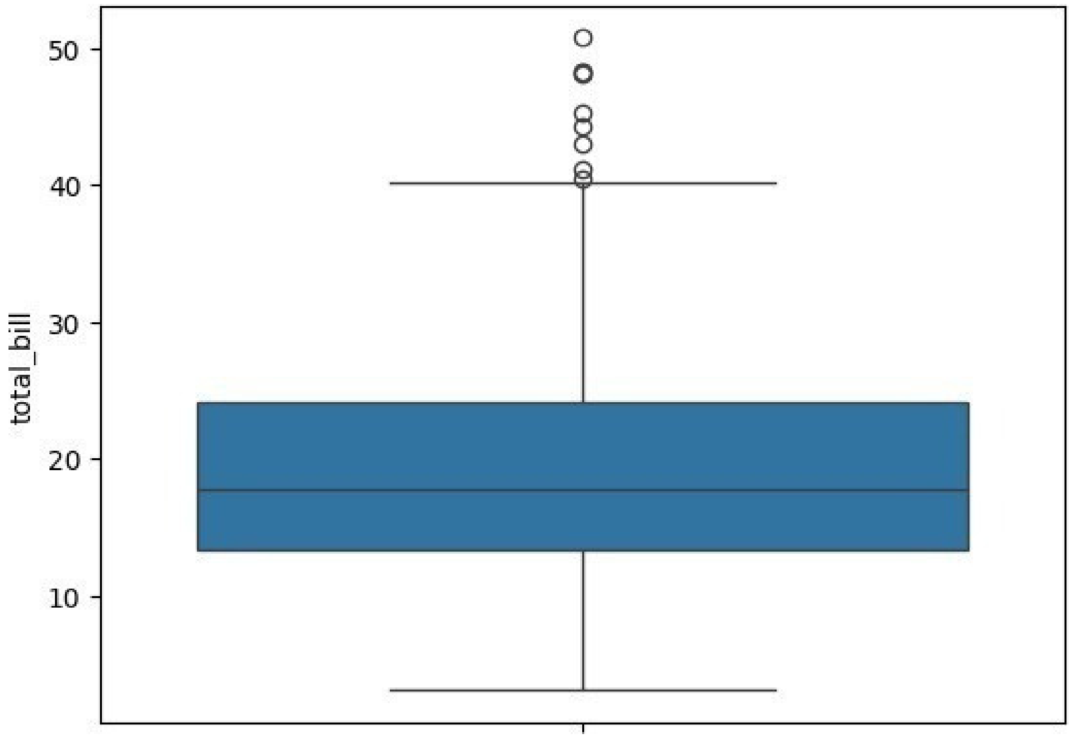

```
sns.heatmap(tips.corr(numeric_only=True),annot=True)
```

```
<Axes: >
```



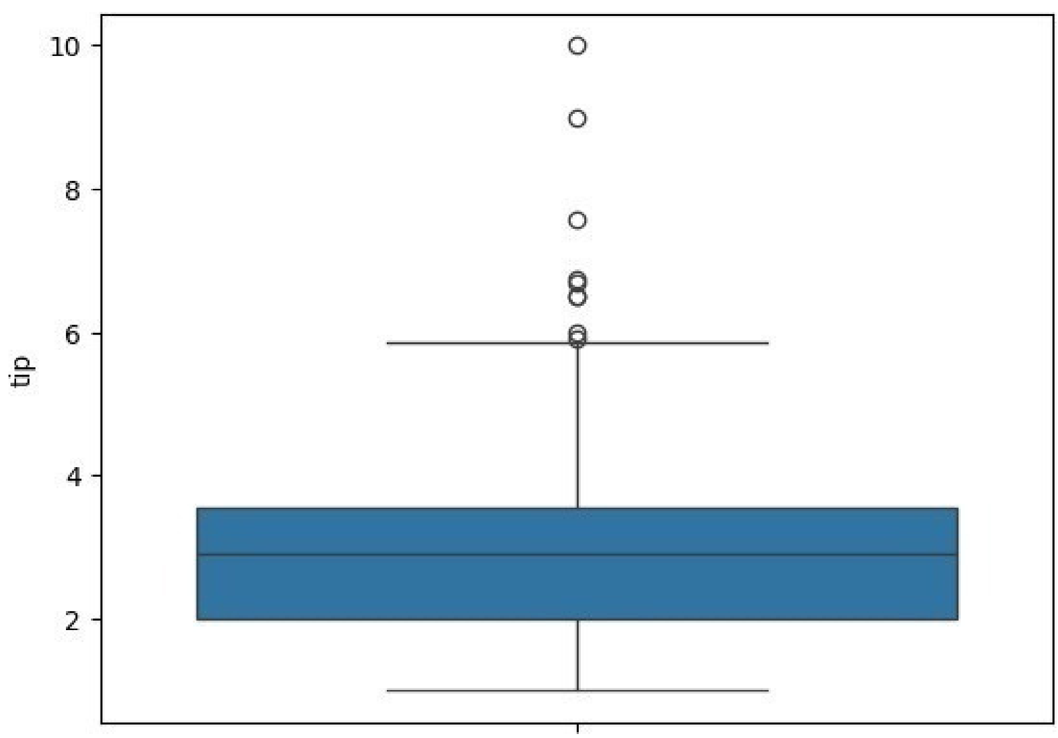

```
sns.boxplot(tips.total_bill)
```

```
<Axes: ylabel='total_bill'>
```



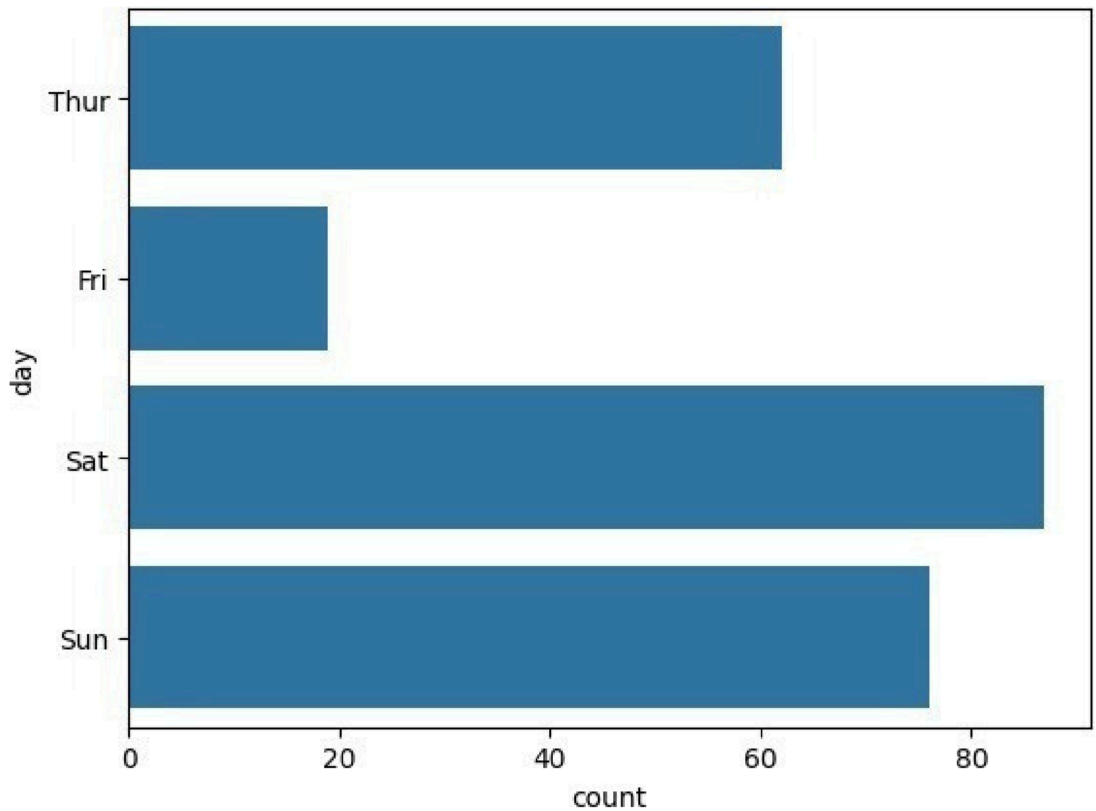

```
sns.boxplot(tips.tip)
```

```
<Axes: ylabel='tip'>
```



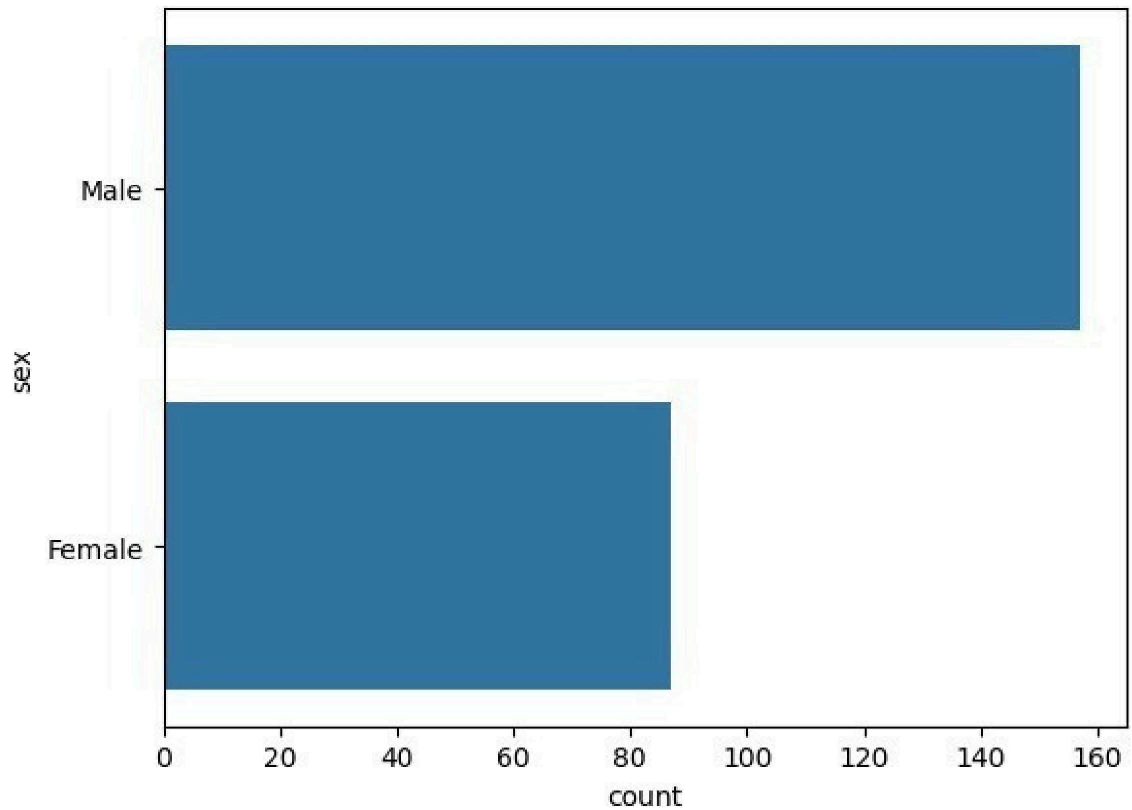
```
sns.countplot(tips.day)
```

<Axes: xlabel='count', ylabel='day'>



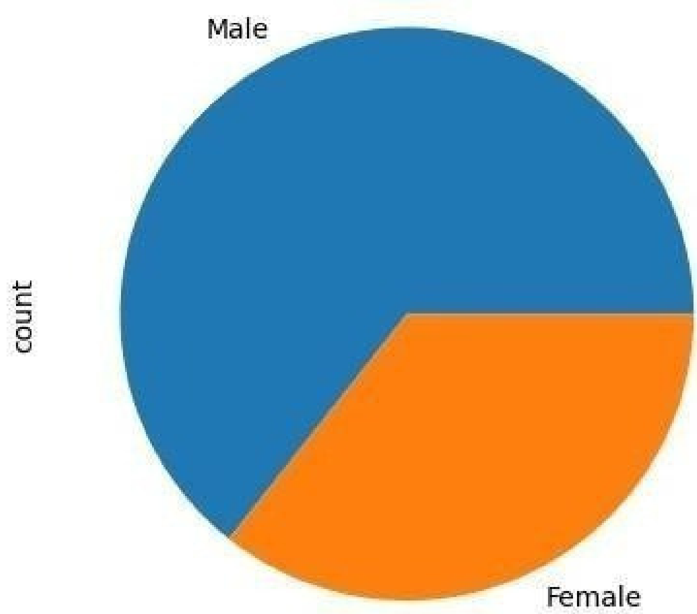

```
sns.countplot(tips.sex)
```

<Axes: xlabel='count', ylabel='sex'>



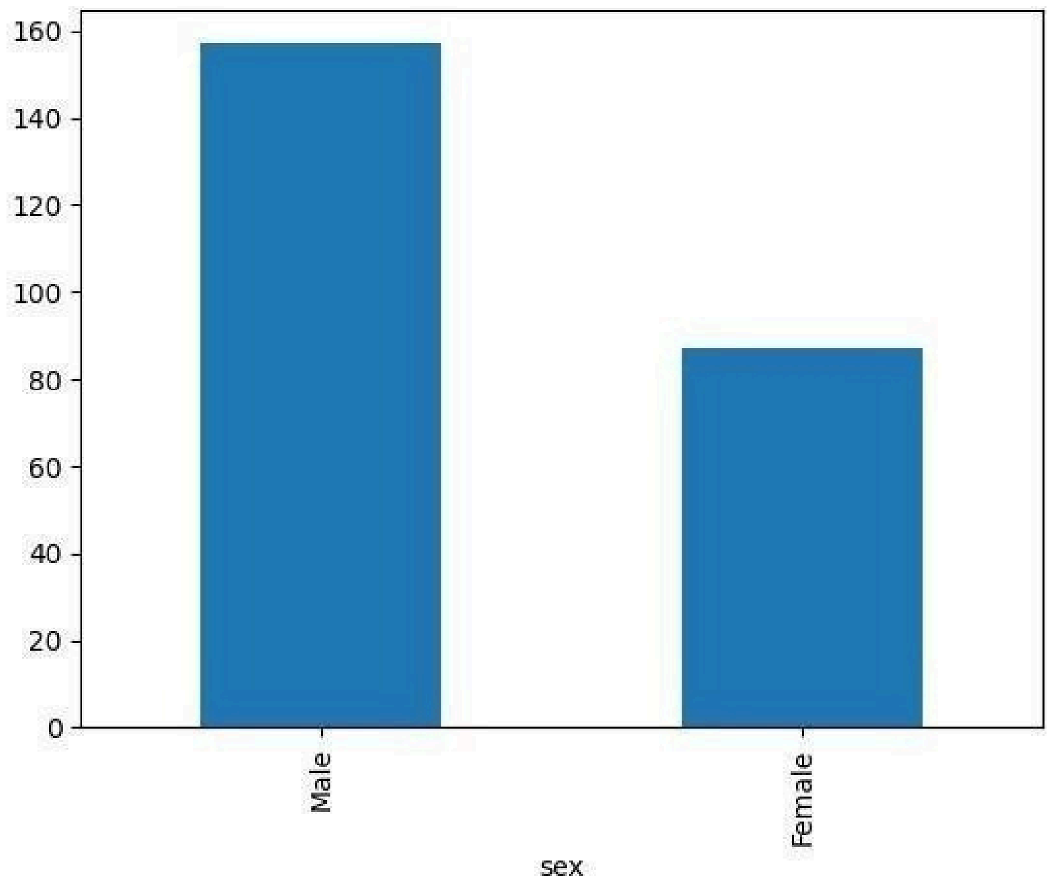

```
tips.sex.value_counts().plot(kind='pie')
```

```
<Axes: ylabel='count'>
```



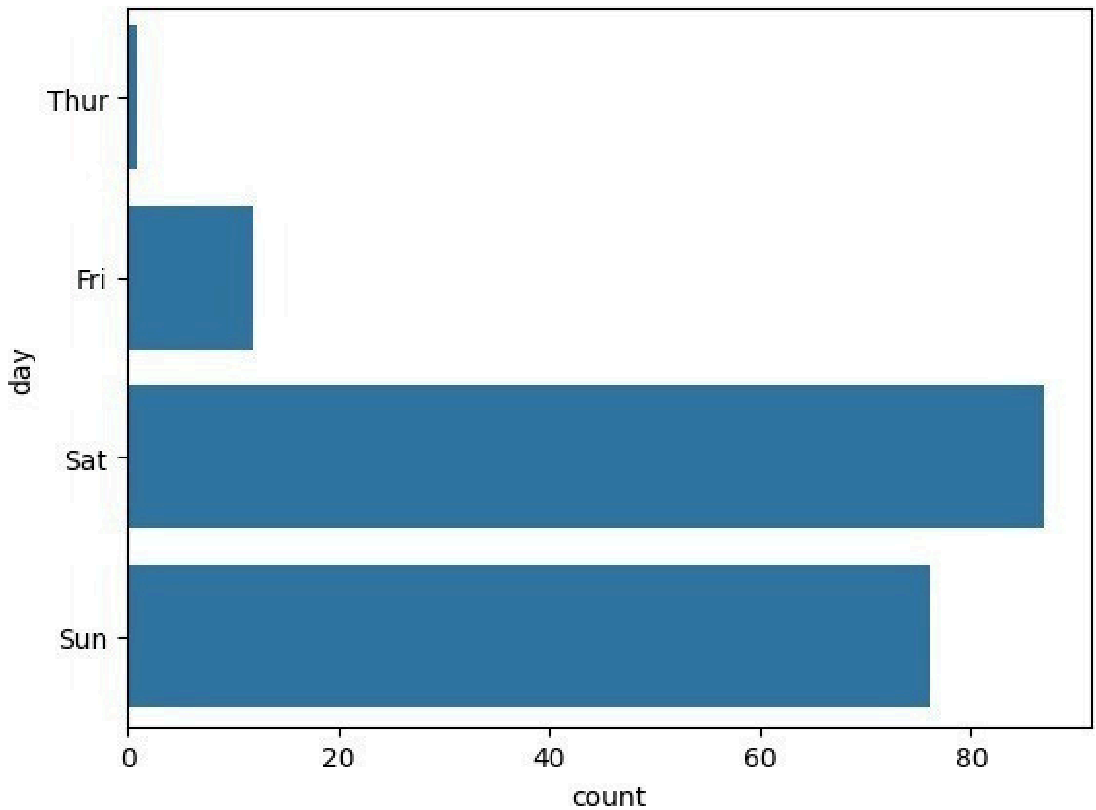

```
tips.sex.value_counts().plot(kind='bar')
```

```
<Axes: xlabel='sex'>
```




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

```
<Axes: xlabel='count', ylabel='day'>
```



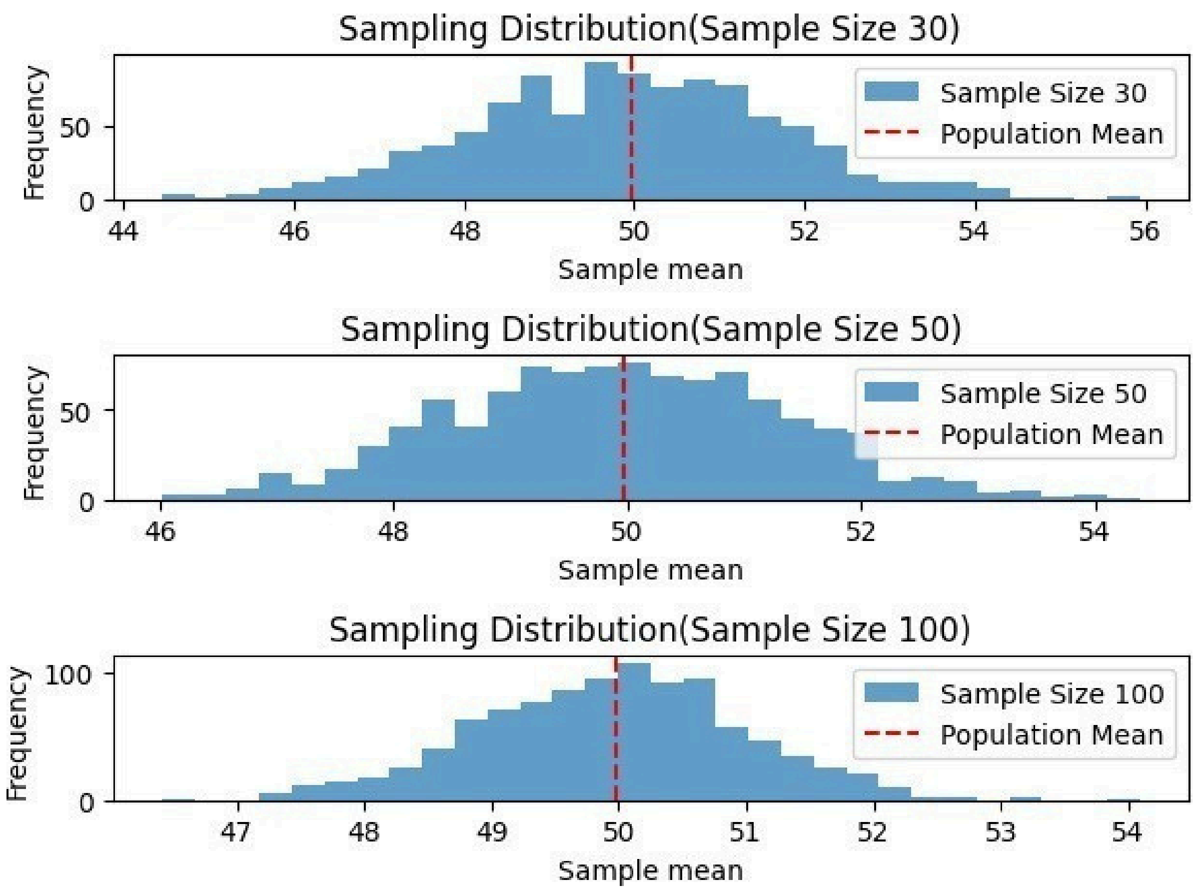

```
#EX.NO :6 Random Sampling and Sampling Distribution #DATA :
10.09.2024

#NAME : Ganesan G
#ROLL NO : 230701089
#DEPARTMENT : B.E COMPUTER SCIENCE AND ENGINEERING - B

import numpy as np
import matplotlib.pyplot as plt
population_mean = 50
population_std = 10
population_size = 100000
population = np.random.normal(population_mean, population_std,
population_size)

sample_sizes = [30, 50, 100]
num_samples = 1000
sample_means = {}
for size in sample_sizes:


    sample_means[size] = []
    for _ in range(num_samples):
        sample = np.random.choice(population, size=size, replace=False)
        sample_means[size].append(np.mean(sample))
```

```
plt.figure(figsize=(12, 8))
```

```
<Figure size 1200x800 with 0 Axes>
```

```
<Figure size 1200x800 with 0 Axes>
```

```
 for i, size in enumerate(sample_sizes):
    plt.subplot(len(sample_sizes), 1, i+1)
    plt.hist(sample_means[size], bins=30, alpha=0.7, label=f'Sample
Size {size}')
    plt.axvline(np.mean(population), color='red', linestyle= 'dashed',
 linewidth=1.5,
label= 'Population Mean')
plt.title(f'Sampling Distribution(Sample Size {size})')
    plt.xlabel('Sample mean')
plt.ylabel('Frequency') plt.legend()
plt.tight_layout()
 plt.show()
```




```
#EX.NO :7 Z-Test
#DATA : 10.09.2024
```

```
#NAME : Ganesan G
ROLL NO : 230701089
 #DEPARTMENT : B.E COMPUTER SCIENCE AND ENGINEERING - B

import numpy as np
import scipy.stats as stats
sample_data = np.array([152, 148, 151, 149, 147, 153, 150, 148, 152,
149,151, 150, 149, 152, 151, 148, 150, 152, 149, 150,148, 153, 151,
150, 149, 152, 148, 151, 150, 153])

population_mean = 150
sample_mean = np.mean(sample_data)
sample_std = np.std(sample_data, ddof=1)

n = len(sample_data)
z_statistic = (sample_mean - population_mean) / (sample_std /
np.sqrt(n))
p_value = 2 * (1 - stats.norm.cdf(np.abs(z_statistic)))
# Assuming sample_mean, z_statistic, and p_value have already been
calculated:
print(f"Sample
{sample_mean:.2f}\n")                    Mean:
print(f"Z-Statistic:
{z_statistic:.4f}\n")
{p_value:.4f}\n")              print(f"P-Value:

# Significance level
alpha = 0.05
# Decision based on p-value
if p_value < alpha:

    print("Reject the null hypothesis: The average weight is
significantly different from 150 grams.")
else:
    print("Fail to reject the null hypothesis: There is no significant
difference in average weight from 150 grams.")
```

Sample Mean: 150.20 Z-Statistic: 0.6406

P-Value: 0.5218 Fail to reject the null hypothesis: There is no

significant difference
in average weight from 150 grams.

```
#EX.NO :8 T-Test
#DATA : 08.10.2024
#NAME : Ganesan G
```

```
#ROLL NO : 230701089 #DEPARTMENT : B.E COMPUTER SCIENCE AND ENGINEERING - B

import numpy as np
import scipy.stats as
stats np.random.seed(42)
sample_size = 25
sample_data = np.random.normal(loc=102, scale=15, size=sample_size)

population_mean = 100
sample_mean = np.mean(sample_data)
sample_std = np.std(sample_data, ddof=1)

n = len(sample_data)
t_statistic, p_value = stats.ttest_1samp(sample_data,population_mean)
# Assuming sample_mean, t_statistic, and p_value have already been
calculated:
print(f"Sample                                    Mean:
{sample_mean:.2f}\n")
print(f"T-Statistic:
{t_statistic:.4f}\n") {p_value:.4pfr}i\nt"()f"P-Value:

# Significance level
alpha = 0.05
# Decision based on p-value
if p_value < alpha:


    print("Reject the null hypothesis: The average IQ score is
significantly different from 100.")
else:
    print("Fail to reject the null hypothesis: There is no significant
difference in average IQ score from 100.")
```

Sample Mean: 99.55

T-Statistic: -0.1577 P-Value: 0.8760 Fail to reject the null hypothesis:

There is no significant difference
in average IQ score from 100.

```
#EX.NO :9 Annova TEST
#DATA : 08.10.2024
#NAME : Ganesan G
#ROLL NO : 230701089
#DEPARTMENT : B.E COMPUTER SCIENCE AND ENGINEERING - B

import numpy as np
import scipy.stats as stats
```

```
from     statsmodels.stats.multicomp     import     pairwise_tukeyhsd

np.random.seed(42)
n_plants = 25
growth_A = np.random.normal(loc=10, scale=2, size=n_plants)
growth_B = np.random.normal(loc=12, scale=3, size=n_plants)
growth_C = np.random.normal(loc=15, scale=2.5, size=n_plants)

all_data     =     np.concatenate([growth_A,     growth_B,     growth_C])

treatment_labels = ['A'] * n_plants + ['B'] * n_plants + ['C'] *
n_plants
f_statistic, p_value = stats.f_oneway(growth_A, growth_B, growth_C)


m=ean_A
np.mean(growth_A) mean_B
=             np.mean(growth_B)
mean_C                              =
np.mean(growth_C)
print(f"Treatment A Mean Growth:
{mean_A:.4f}") print(f"Treatment B Mean
Growth: {mean_B:.4f}") print(f"Treatment C
Mean Growth: {mean_C:.4f}")
print(f"F-Statistic: {f_statistic:.4f}")
print(f"P-Value: {p_value:.4f}")

alpha = 0.05
if p_value < alpha:

a     print("Reject       the     null     hypothesis:     There     is
significant difference in mean growth rates among the three
treatments.") else:
      print("Fail       to   reject    the    null    hypothesis:    There    is    no
significant       difference    in    mean growth rates       among the     three
treatments.")



if p_value < alpha:

      tukey_results = pairwise_tukeyhsd(all_data, treatment_labels,
alpha=0.05)


      print("\nTukey's HSD Post-hoc Test:")
      print(tukey_results)
```

```
Treatment A Mean Growth: 9.6730
Treatment B Mean Growth:
11.1377 Treatment C Mean
Growth: 15.2652 F-Statistic:
36.1214
P-Value: 0.0000
Reject the null hypothesis: There is a significant difference in mean
growth rates among the three treatments.
Tukey's HSD Post-hoc Test:
Multiple Comparison of Means - Tukey HSD, FWER=0.05
```

```
=====================================================
=
group1 group2 meandiff p-adj      lower upper reject
----------------------------------------------------
1.4647A0 .0877B                  -0.16833.0977      False
   A        C     5.5923      0.   3.95937.2252     True
   B        C     4.1276      0    2.49465.7605     True
                              0.
----------------------------------------------------
                              0
```

#EX.NO :10 Feature Scaling #DATA : 22.10.2024

#NAME : Ganesan G

#ROLL NO : 230701089

#DEPARTMENT : B.E COMPUTER SCIENCE AND ENGINEERING - B

```
import numpy as np
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
df=pd.read_csv('pre_process_datasample.csv')

df.head()
```

```
   Countr   Age   Salary
0      y  44.   72000.0  Purchased    N
1  France  0    48000.0               o
2   Spain  27.0 54000.0              Ye
3  German  30.0 61000.0               s
4      y  38.0      NaN              No
   Spain  40.0                       No
  German                             Ye
       y                              s
```

```
df.Country.fillna(df.Country.mode()[0],inplace=True
) features=df.iloc[:,:-1].values 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, nan],
['France', 35.0, 58000.0],
['Spain', nan, 52000.0],
['France', 48.0, 79000.0],
['Germany', 50.0, 83000.0],
['France', 37.0, 67000.0]],

dtype=object) label=df.iloc[:,-1].values

from sklearn.impute import SimpleImputer
age=SimpleImputer(strategy="mean",missing_values=np.nan)
Salary=SimpleImputer(strategy="mean",missing_values=np.nan)
age.fit(features[:,[1]])
```

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

SimpleImputer()   SimpleImputer()    SimpleImputer()

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

array([['France', 44.0, 72000.0],

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

from sklearn.preprocessing import OneHotEncoder
oh = OneHotEncoder(sparse_output=False)
Country=oh.fit_transform(features[:,[0]])
Country

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

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

         array([[1.0, 0.0, 0.0, 44.0,
                        72000.0],
       [0.0, 0.0, 1.0, 27.0, 48000.0],
       [0.0, 1.0, 0.0, 30.0, 54000.0],
       [0.0, 0.0, 1.0, 38.0, 61000.0],
       [0.0, 1.0, 0.0, 40.0, 63777.77777777778],
       [1.0, 0.0, 0.0, 35.0, 58000.0],
       [0.0, 0.0, 1.0, 38.77777777777778, 52000.0],
```

```
[1.0, 0.0, 0.0, 48.0, 79000.0], [0.0, 1.0, 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([[ 1.22474487e+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]])
```

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

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

#EX.NO :11 Linear Regression
#DATA : 29.10.2024

```
#NAME : Ganesan G #ROLL NO : 230701089
#DEPARTMENT : B.E COMPUTER SCIENCE AND
ENGINEERING - B
import numpy as np
import pandas as pd
df = pd.read_csv('Salary_data.csv')
df
```

```
YearsExperience Salary
0    1.1   39343
1    1.3   46205
2    1.5   37731
3    2.0   43525
4    2.2   39891
5    2.9   56642
6    3.0   60150
7    3.2   54445
8    3.2   64445
9    3.7   57189
10   3.9   63218
11   4.0   55794
12   4.0   56957
13   4.1   57081
14   4.5   61111
15   4.9   67938
16   5.1   66029
17   5.3   83088
18   5.9   81363
19   6.0   93940
20   6.8   91738
21   7.1   98273
22   7.9 101302
23   8.2 113812
24   8.7 109431
25   9.0 105582
26   9.5 116969
27   9.6 112635
28   10.3 122391
29   10.5 121872
```

```
df.info()                                   <class
'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 2 columns):
 #     Column              Non-NullCountDtyp
       --------            ----------------  e
-----                                        --------
```

```
 0    YearsExperience 30 non-null         float64
 1   Salary                 30non-null
int64 dtypes: float64(1), int64(1)
memory usage: 612.0 bytes
```

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

```
YearsExperience Salary
0     1.1   39343
1     1.3   46205
2     1.5   37731
3     2.0   43525
4     2.2   39891
5     2.9   56642
6     3.0   60150
7     3.2   54445
8     3.2   64445
9     3.7   57189
10    3.9   63218
11    4.0   55794
12    4.0   56957
13    4.1   57081
14    4.5   61111
15    4.9   67938
16    5.1   66029
17    5.3   83088
18    5.9   81363
19    6.0   93940
20    6.8   91738
21    7.1   98273
22    7.9 101302
23    8.2 113812
24    8.7 109431
25    9.0 105582
26    9.5 116969
27    9.6 112635
28    10.3 122391
29    10.5 121872
```

```
df.info()                                   <class
'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 2 columns):
 #    Column              Non-NullCountDtyp
      -------             ---------------  e
-----                                      --------
 0    YearsExperience  30non-null          float64
 1    Salary              30non-null       int64
```

```
dtypes: float64(1), int64(1)
memory usage: 612.0 bytes
```

```python
df.describe() #descripte statical report
# find out lYER FOR BELOW META DATA
```

	YearsExperience	Salary
count		
mean	30.0050.30103033 37603003..000000000000	
std	2.83178080806325260755307004009708205 ..74.772.1	
min	0000000003200 00	
25%	0 7	
50%		
75%		
max	10.500000122391.000000	

```python
features = df.iloc[:,[0]].values # : - > all row , 0 -> first column

#iloc index based selection loc location based sentence

label = df.iloc[:,[1]].values

features
```

```
array([ 1.1],
[
[ 1.3],
[ 1.5],
[ 2. ],
[ 2.2],
[ 2.9],
[ 3. ],
[ 3.2],
[ 3.2],
[ 3.7],
[ 3.9],
[ 4. ],
[ 4. ],
[ 4.1],
[ 4.5],
[ 4.9],
[ 5.1],
[ 5.3],
[ 5.9],
[ 6. ],
[ 6.8],
[ 7.1],
[ 7.9],
[ 8.2],
[ 8.7],
[ 9. ],
```

```
[ 9.5],
[ 9.6],
[10.3],
        [10.5]])
```

label

```
array([ 39343]
[       ,
[ 46205]
       ,
[ 37731]
       ,
[ 43525]
       ,
[ 39891]
       ,
[ 56642]
       ,
[ 60150]
       ,
[ 54445]
       ,
[ 64445]
       ,
[ 57189]
       ,
[ 63218]
       ,
[ 55794]
       ,
[ 56957]
       ,
[ 57081]
       ,
[ 61111]
       ,
[ 67938]
       ,
[ 66029]
       ,
[ 83088]
       ,
[ 81363]
       ,
[ 93940]
       ,
[ 91738]
       ,
[ 98273]
       ,
[101302]
,
[113812]
,
[109431]
,
[105582]
```

```
,
[116969]
,
[112635]
,
[122391]
,
[121872]], dtype=int64)
```

```
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=2
3) # x independent input train 80 % test 20 %
'''
y is depenent ouput
0.2 allocate test for 20 % automatically train for 80 %
'''
```

```
'\ny is depenent ouput\n0.2 allocate test for 20 % automatically train
for 80 %\n'
```

```
=from sklearn.linear_model import LinearRegression model
LinearRegression() model.fit(x_train,y_train) ''' sk - size kit linear
means using linear regression fit means add data '''

'\nsk - size kit \nlinear means using linear regression \nfit
means add data \n'

model.score(x_train,y_train)
'''
accuracy calculating
96%
'''
'\naccuracy calculating\n96 %\n'

model.score(x_test,y_test)

'''
accuracy calculating
91%
'''
'\naccuracy calculating\n91 %\n' model.coef_ array([[9281.30847068]])

model.intercept_ array([27166.73682891]) import pickle
pickle.dump(model,open('SalaryPred.model','wb')

) '''
pickle momory obj to file
'''

'\npickle momory obj to file\n\n'

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

yr_of_exp = float(input("Enter years of expreience: "))
yr_of_exp_NP = np.array([[yr_of_exp]])
salary = model.predict(yr_of_exp_NP)
print("Estimated salary for {} years of expreience is {} .
".format(yr_of_exp,salary))
```

```
Enter years of expreience: 24
Estimated salary for 24.0 years of expreience is [[249918.14012525]] .
```

```
print(f" Estimated salary for {yr_of_exp} years of expreience is
{salary} . ")
```

```
 Estimated salary for 24.0 years of expreience is
[[249918.14012525]] .
```

```
#EX.NO:12       LogisticRegression
#DATA : 05.11.2024
#NAME : Ganesan G
#ROLL NO : 230701089
#DEPARTMENT : B.E COMPUTER SCIENCE AND
ENGINEERING - B

import numpy as np
import pandas as pd
import warnings
w)adrfnings.filterwarnings('ignore')
df=pd.read_csv('Social_Network_Ads.csv.csv'
```

```
        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    Mal.e..  1.9.        7600..0.       .0..
..                           46
             ...             5150
39515691863        Female                41000          1
31597606071          Male                23000          1
39715654296      Female 36               20000          1
39815755018          Male   49           33000          0
39915594041        Female                36000          1

[400 rows x 5 columns]
```

```
df.tail(20)
```

```
User ID Gender Age EstimatedSalary Purchased
380 15683758     Male  42    64000 0
381 15670615     Male  48    33000 1
382 15715622 Female   44    139000      1
383 15707634     Male  49    28000 1
384 15806901 Female   57    33000 1
385 15775335     Male  56    60000 1
386 15724150 Female   49    39000 1
387 15627220     Male  39    71000 0
```

388 1358697 2333900Male 4 3400 1
15807815367 68521 Female 7 0 1
Mal 4 3500 1
391 15592570 e 8 0 1
392 15748589 FeMmaalle 4 3300 1
393 15635893 157M57e6 a3l8e 0 1
394 Female 15691863 4 2300 0
395 Female 15706071 7 0 1
396 Male 4 4500 1
397 15654296 Female 5 0 1
3195785018 Male 6 4200 0
399 15594041 Female 0 0 1
3 5900
9 0
4 4100

```
df.head(25)
```

6 0
51 2300
5 0
0 2000
3 0
6 3300
4 0
9 3600
0

	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
5	15728773	Male	27	58000	0
6	15598044	Female	27	84000	0
7	15694829	Female	32	150000	1
8	15600575	Male	25	33000	0
9	15727311	Female	35	65000	0
10	15570769	Female	26	80000	0
11	15606274	Female	26	52000	0
12	15746139	Male	20	86000	0
13	15704987	Male	32	18000	0
14	15628972	Male	18	82000	0
15	15697686	Male	29	80000	0
16	15733883	Male	47	25000	1
17	15617482	Male	45	26000	1
18	15704583	Male	46	28000	1
19	15621083	Female	48	29000	1
20	15649487	Male	45	22000	1
21	15736760	Female	47	49000	1
22	15714658	Male	48	41000	1
23	15599081	Female	45	22000	1
24	15705113	Male	46	23000	1

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

```
array([ 19, 19000]
      [
      [ 35, ,
              20000]
              ,
```

```
[   26     43000]
              ,
[   ,      57000]
              ,
    27

    ,
[   19,    76000]
    27,       ,
[   27,    58000]
              ,
[          84000]
              ,
[   32, 150000]
             ,
[   25,    33000]
    35,       ,
[   26,     ,0500]
    26,    52000]0]
[   20,     800,
[   32,    86000]
    18,       ,
[   29,    18000]
              ,
[   47,    82000]
    45,       ,
[   46,    25000,0000]]
    48,    8  ,
[   45,    26000]
[   47,       ,
    48,     280
[   45,    29000]
    46,       ,
[   47,    22000]
    49,    49000]
[   47,       ,
           41000]
[          220,00]
[             ,
           23000]
[             ,
           20000]
[             ,
           28000]
[             ,
           30000]
[
[
```

```
[       29,      43000]
        31,
[       31,      18000]
[                74000]
[       27, 137000]
[       21,      16000]
        28,
[       27,      440,00]
        35,      27,0000]
[                0]
        33,      900,
[       30,      28000]
        26,
[                49000]
        27,
[                72000]
        27,
[       33,      310'00]
[                17000]
[
[
[       35, 1080501,00]0]
[       30,      15000]
        28,
[       23,      840,000]
        25,      2709,000]0]
[       27,      ,
                 54000]
[
[       30, 135000,]
[       31,      89000]
        24,
[       18,      32000]
        29,      ,0,
[                48430000]
[

[       35,      23000]
```

```
                ,
[      27,    58000]
       24,      ,
[      23,    55000]
       28,       ;
[      22,    79000]
                ,
[             18000]
                ,
[

[      32, 117000]
               ,
[      27,    20000]
       25,       ,
[
       23,    870,00]
[             66000]
[      32, 120000,]
               ,
[      59,    83000]
       24,    ,
[      24,    580,00]
[      23,      ,00]
               190
[      22,
              82000]
[      31,
       25,       ,
[      24,    63000]
              68000]
[      20,
                 ,
[             80000]
                ,
[             27000]
       33, 1130203,00,]00]
[
               ,
[      32,    18000]
                ,
[      34, 112000]
               ,
[      18,    52000]
       22,
[                ,
       28,    270,00]
[      26,      ,00]
       30,     870
[             17000]
                ,
[             80000]
```

```
[   39,   42000]
    20,     ,
[   35,   49000]
    30,     ,
[         88000]
            ,
[         62000]
            ,
[   31, 118000]
            ,
[   24,   55000]
    28,     ,
[
    26,   850,00]
[   35,   50,0000]0]
            810
    22,     ,
[         81000]
[
[   30, 116000,]
            ,
[   26,   15000]
    29,   ,
[
    29,   280,00]
[   35,   44,0000]0]
    35,   830,
[         25000]
[
[   28, 123000,]
            ,
[   35,   73000]
    28,   ,
[
    27,   370,00]
    28,     ,00]
[         85890000]
[           ,
          86000]
            ,
[   32,
[   33, 149000]
            ,
[   19,   21000]
    21,     ,
[   26,   72000]
            ,
[         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,          ,
[        20,     63000]
                      ,
[                82000]
                      ,
[        30, 107000]
                    ,
[        28,     59000]
         19,          ,
[
         19,      250,00]
[        18,     68,0000]0]
         35,       850,
[
         30,     59000]
         34,          ,
[                89000]
         24,          ,
[
         27,     25000]
[        41,          ,
                 986900,0000]
[        29,                ]
         20,          ,
[
         26,     30000]
[
                      ,
                  61000]
[
                      ,
                 74000]
[
                      ,
                  15000]
[                     ,00]
                   450
                 76000]
                 500,00]
                      ,
[        41,     47000]
         31,
[        36,      150,00]
         40,     59000]
[        31,
         46,          ,
[                75000]
         29,          ,
[        26,     30000]
[
[
[
[        32, 135000,]
```

```
[ 32, 100000]
[ 25, 90000]
  37,
[ 35, 33000]
  33, 380,00]
[ 18,
[ 22, 690 00]
  35, 86000]
[
       55000]
[
       71000]
[
[ 29, 148000,]
[ 29, 47000]
  21,
[      88000]
[ 34, 115000,]
[ 26, 118000]
[ 34, 43000]
  34,
[ 23, 72000]
[ 35, 47,0,000]0]
  25,
[      280
  24,
  31,  22000]
[ 26,
  31,  23000]
[
       34000]
[
       16000]
[      7010,0,]00]
[
[ 32, 1170
[ 33, 43000]
  33,
[
  31,  600,00]
[ 20,  6862000]
[
```

```
[33, 41000],
[35, 72000],
[28, 32000],
[24, 84000],
[19, 26000],
[29, 43000],
[19, 70000],
[28, 89000],
[34, 43000],
[30, 79000],
[20, 36000],
[26, 800,00]
[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,
            46,          78000]
[                        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,
            37,          57000]
            46,          ,0,0]
[                        802000]
[
[           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, 57000],
[40, 72000],
[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, ,0,0]
[   49,
[   52,   710
[   50,  88000]
[   59,  38000]
[   35,       ,
[   37,  36000],
[   52,  88000],
[            61000],
[            70000],
[
[   48, 1410201,00,0]0]
[   37,  93000],
[   37,  62000],
[   48, 138000,]
[   41,  79000],
[   37,  78000],
[   39, 134000,]
[   49,  89000],
[   55,  39000]
```

```
                 ,
[      37,      77000]
       35,           ,
[      36,     57000]
       42,           ,
[             63000]
                     ,
[             73000]
                     ,
[      43, 112000]
                ,

[      45,      79000]
[      46, 117000]
                ,
[      58,      38000]
       48,           ,
[               74000]
                     ,
[      37, 137000]
                ,
[      37,      79000]
       40,           ,
[
       42,      600,00]
[               54000]
[      51, 134000,]
                ,
[      47, 113000]
                ,
[      36, 125000]
                ,
[      38,      50000]
       42,           ,
[      39,      70000]
       38,        ,0,0]
[               95060000]
[
[      49, 141000,]
                ,
[      39,      79000]
       39,           ,
[               75000]
[      54, 10400,0]
                ,
[      35,      55000]
       45,           ,
[               32000]
```

```
[36, 60000],
[52, 138000],
[53, 82000],
[41, 52000],
[48, 30000],
[48, 131000],
[41, 60000],
[41,
[42, 72000],
[75000],
[36, 118000],
[47, 107000],
[38, 51000],
[48, 119000],
[42, 65000],
[40,
[57, 65000],
[36, ,0,0]
6504000]
[58, 144000],
[35, 79000],
[38, 55000],
[39, 122000],
[53, 104000],
[35, 75000],
[38,
[47, 65000],
[51000],
[47, 105000],
[41, 63000],
```

[53, 72000]
,

[54, 108000]
,

```
[ 39, 77000]
  38,  ,
[      61000]
        ,
[ 38, 113000]
        ,
[ 37, 75000]
  42,   ,
[ 37,  9000,00]
[ 36, 99,0000]0]
  60,  570,
[ 54, 34000]
[ 41,   ,
  40, 70000]
[       ,
  42,
[      72000]
        ,
[      71000]
        ,
[

[ 43, 12905040,]00]
        ,
[ 53, 34000]
  47,   ,
[ 42, 500,00]
[      79000]
[ 42, 10400,0]
        ,
[ 59, 29000]
  58,  ,
[ 46, 470,00]
[ 38,  ,00]
  54,  880
[ 60, 71000]
[ 60,   ,
  39, 26000]
[       ,
[      46000]
        ,
[      83000]
  59, 1300703,0,0]0]
[
        ,
[ 37, 80000]
  46,   ,
[
  46, 320,00]
[ 42,  ,00]
       7543000]
[       ,
```

```
[        41,       87000]
         58,            ,
[        42,       23000]
         48,      640,00]
[                       ,
                  33000]
[                       ,

[        44, 139000]
                      ,
[        49,      28000]
         57,            ,
[
         56,      330,00]
[        49,      39,0000]0]
         39,        600,
[        47,       71000]
[        48,            ,
                  34000]
         48,            ,
[        47,      35000]
[        45,            ,
         60,      23300,0000]]
[        39,      3     ,
         46,      45000]
[        51,
                  42000]
[                       ,
                  59000]
[                       ,
                  41000]
[                       ,
                  23000]
[                       ,

[
```

```
[ [      50    20000],
 [       ,     33000],
       4369,, 36000]], dtype=int64)
```

label

```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1
,
1,
                    1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
,
0,                    0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0
                    0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 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, 1, 0, 0, 0, 0, 0
,
0,                  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
,
0,                  0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0
,
0,                  0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0
,
```

```
0,
0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
,
0,
0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0
,
1,
0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1
,
0,
1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1
,
0,
1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0
,
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
1,
,
1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0
0,
,
0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0
,
1,

1,
           1, 1, 0, 1], dtype=int64)
```

```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
# Assuming `features` and `label` are already defined

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()
    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(f"Test Score: {test_score:.4f} | Train Score:
{train_score:.4f} | Random State: {i}")

'''


''' Test Score: 0.9000 | Train Score: 0.8406 | Random State: 4
Test Score: 0.8625 | Train Score: 0.8500 | Random State: 5
Test Score: 0.8625 | Train Score: 0.8594 | Random State: 6
Test Score: 0.8875 | Train Score: 0.8375 | Random State: 7
Test Score: 0.8625 | Train Score: 0.8375 | Random State: 9
Test Score: 0.9000 | Train Score: 0.8406 | Random State: 10
Test Score: 0.8625 | Train Score: 0.8562 | Random State: 14
Test Score: 0.8500 | Train Score: 0.8438 | Random State: 15
Test Score: 0.8625 | Train Score: 0.8562 | Random State: 16
Test Score: 0.8750 | Train Score: 0.8344 | Random State: 18
Test Score: 0.8500 | Train Score: 0.8438 | Random State: 19
Test Score: 0.8750 | Train Score: 0.8438 | Random State: 20
Test Score: 0.8625 | Train Score: 0.8344 | Random State: 21
Test Score: 0.8750 | Train Score: 0.8406 | Random State: 22
Test Score: 0.8750 | Train Score: 0.8406 | Random State: 24
Test Score: 0.8500 | Train Score: 0.8344 | Random State: 26
Test Score: 0.8500 | Train Score: 0.8406 | Random State: 27
Test Score: 0.8625 | Train Score: 0.8344 | Random State: 30
Test Score: 0.8625 | Train Score: 0.8562 | Random State: 31
Test Score: 0.8750 | Train Score: 0.8531 | Random State: 32
Test Score: 0.8625 | Train Score: 0.8438 | Random State: 33
Test Score: 0.8750 | Train Score: 0.8313 | Random State: 35
Test Score: 0.8625 | Train Score: 0.8531 | Random State: 36
Test Score: 0.8875 | Train Score: 0.8406 | Random State: 38
Test Score: 0.8750 | Train Score: 0.8375 | Random State: 39
Test Score: 0.8875 | Train Score: 0.8375 | Random State: 42
Test Score: 0.8750 | Train Score: 0.8469 | Random State: 46
Test Score: 0.9125 | Train Score: 0.8313 | Random State: 47
Test Score: 0.8750 | Train Score: 0.8313 | Random State: 51
Test Score: 0.9000 | Train Score: 0.8438 | Random State: 54
Test Score: 0.8500 | Train Score: 0.8438 | Random State: 57
Test Score: 0.8750 | Train Score: 0.8438 | Random State: 58
Test Score: 0.9250 | Train Score: 0.8375 | Random State: 61
```

```
Test Score: 0.8875 | Train Score: 0.8344 | Random State: 65 Test
Score: 0.8875 | Train Score: 0.8406 | Random State: 68 Test
Score: 0.9000 | Train Score: 0.8313 | Random State: 72 Test
Score: 0.8875 | Train Score: 0.8375 | Random State: 75 Test
Score: 0.9250 | Train Score: 0.8250 | Random State: 76 Test
Score: 0.8625 | Train Score: 0.8406 | Random State: 77 Test
Score: 0.8625 | Train Score: 0.8594 | Random State: 81 Test
Score: 0.8750 | Train Score: 0.8375 | Random State: 82 Test
Score: 0.8875 | Train Score: 0.8375 | Random State: 83 Test
Score: 0.8625 | Train Score: 0.8531 | Random State: 84 Test
Score: 0.8625 | Train Score: 0.8406 | Random State: 85 Test
Score: 0.8625 | Train Score: 0.8406 | Random State: 87 Test
Score: 0.8750 | Train Score: 0.8469 | Random State: 88 Test
Score: 0.9125 | Train Score: 0.8375 | Random State: 90 Test
Score: 0.8625 | Train Score: 0.8500 | Random State: 95 Test
Score: 0.8750 | Train Score: 0.8500 | Random State: 99 Test
Score: 0.8500 | Train Score: 0.8406 | Random State: 101 Test
Score: 0.8500 | Train Score: 0.8406 | Random State: 102 Test
Score: 0.9000 | Train Score: 0.8250 | Random State: 106 Test
Score: 0.8625 | Train Score: 0.8406 | Random State: 107 Test
Score: 0.8500 | Train Score: 0.8344 | Random State: 109 Test
Score: 0.8500 | Train Score: 0.8406 | Random State: 111 Test
Score: 0.9125 | Train Score: 0.8406 | Random State: 112 Test
Score: 0.8625 | Train Score: 0.8500 | Random State: 115 Test
Score: 0.8625 | Train Score: 0.8406 | Random State: 116 Test
Score: 0.8750 | Train Score: 0.8344 | Random State: 119 Test
Score: 0.9125 | Train Score: 0.8281 | Random State: 120 Test
Score: 0.8625 | Train Score: 0.8594 | Random State: 125 Test
Score: 0.8500 | Train Score: 0.8469 | Random State: 128 Test
Score: 0.8750 | Train Score: 0.8500 | Random State: 130 Test
Score: 0.9000 | Train Score: 0.8438 | Random State: 133 Test
Score: 0.9250 | Train Score: 0.8344 | Random State: 134 Test
Score: 0.8625 | Train Score: 0.8500 | Random State: 135 Test
Score: 0.8750 | Train Score: 0.8313 | Random State: 138 Test
Score: 0.8625 | Train Score: 0.8500 | Random State: 141 Test
Score: 0.8500 | Train Score: 0.8469 | Random State: 143 Test
Score: 0.8500 | Train Score: 0.8469 | Random State: 146 Test
Score: 0.8500 | Train Score: 0.8438 | Random State: 147 Test
Score: 0.8625 | Train Score: 0.8500 | Random State: 148 Test
Score: 0.8750 | Train Score: 0.8375 | Random State: 150 Test
Score: 0.8875 | Train Score: 0.8313 | Random State: 151 Test
Score: 0.9250 | Train Score: 0.8438 | Random State: 152 Test
Score: 0.8500 | Train Score: 0.8406 | Random State: 153 Test
Score: 0.9000 | Train Score: 0.8438 | Random State: 154 Test
Score: 0.9000 | Train Score: 0.8406 | Random State: 155 Test
Score: 0.8875 | Train Score: 0.8469 | Random State: 156 Test
Score: 0.8875 | Train Score: 0.8344 | Random State: 158 Test
Score: 0.8750 | Train Score: 0.8281 | Random State: 159 Test
Score: 0.9000 | Train Score: 0.8313 | Random State: 161
```

```
Test Score: 0.8500 | Train Score: 0.8375 | Random State: 163
Test Score: 0.8750 | Train Score: 0.8313 | Random State: 164
Test Score: 0.8625 | Train Score: 0.8500 | Random State: 169
Test Score: 0.8750 | Train Score: 0.8406 | Random State: 171
Test Score: 0.8500 | Train Score: 0.8406 | Random State: 172
Test Score: 0.9000 | Train Score: 0.8250 | Random State: 180
Test Score: 0.8500 | Train Score: 0.8344 | Random State: 184
Test Score: 0.9250 | Train Score: 0.8219 | Random State: 186
Test Score: 0.9000 | Train Score: 0.8313 | Random State: 193
Test Score: 0.8625 | Train Score: 0.8500 | Random State: 195
Test Score: 0.8625 | Train Score: 0.8406 | Random State: 196
Test Score: 0.8625 | Train Score: 0.8375 | Random State: 197
Test Score: 0.8750 | Train Score: 0.8406 | Random State: 198
Test Score: 0.8875 | Train Score: 0.8375 | Random State: 199
Test Score: 0.8875 | Train Score: 0.8438 | Random State: 200
Test Score: 0.8625 | Train Score: 0.8375 | Random State: 202
Test Score: 0.8625 | Train Score: 0.8406 | Random State: 203
Test Score: 0.8875 | Train Score: 0.8313 | Random State: 206
Test Score: 0.8625 | Train Score: 0.8344 | Random State: 211
Test Score: 0.8500 | Train Score: 0.8438 | Random State: 212
Test Score: 0.8625 | Train Score: 0.8344 | Random State: 214
Test Score: 0.8750 | Train Score: 0.8313 | Random State: 217
Test Score: 0.9625 | Train Score: 0.8187 | Random State: 220
Test Score: 0.8750 | Train Score: 0.8438 | Random State: 221
Test Score: 0.8500 | Train Score: 0.8406 | Random State: 222
Test Score: 0.9000 | Train Score: 0.8438 | Random State: 223
Test Score: 0.8625 | Train Score: 0.8531 | Random State: 227
Test Score: 0.8625 | Train Score: 0.8344 | Random State: 228
Test Score: 0.9000 | Train Score: 0.8406 | Random State: 229
Test Score: 0.8500 | Train Score: 0.8438 | Random State:
```

```
232
Test Score: 0.8750 | Train Score: 0.8469 | Random State:
233
Test Score: 0.9125 | Train Score: 0.8406 | Random State:
234
Test Score: 0.8625 | Train Score: 0.8406 | Random State:
235
Test Score: 0.8500 | Train Score: 0.8469 | Random State:
236
Test Score: 0.8750 | Train Score: 0.8469 | Random State:
239
Test Score: 0.8500 | Train Score: 0.8438 | Random State:
241
Test Score: 0.8875 | Train Score: 0.8500 | Random State:
242
Test Score: 0.8875 | Train Score: 0.8250 | Random State:
243
Test Score: 0.8750 | Train Score: 0.8469 | Random State:
244
Test Score: 0.8750 | Train Score: 0.8406 | Random State:
245
Test Score: 0.8750 | Train Score: 0.8469 | Random State:
246
Test Score: 0.8625 | Train Score: 0.8594 | Random State:
247
Test Score: 0.8875 | Train Score: 0.8438 | Random State:
248
Test Score: 0.8625 | Train Score: 0.8500 | Random State:
250
Test Score: 0.8750 | Train Score: 0.8313 | Random State:
251
Test Score: 0.8875 | Train Score: 0.8438 | Random State:
252
Test Score: 0.8625 | Train Score: 0.8469 | Random State:
255
Test Score: 0.9000 | Train Score: 0.8406 | Random State:
257
Test Score: 0.8625 | Train Score: 0.8562 | Random State:
260
```

Test Score: 0.8625 | Train Score: 0.8406 | Random State: 266
Test Score: 0.8625 | Train Score: 0.8375 | Random State: 268
Test Score: 0.8750 | Train Score: 0.8406 | Random State: 275
Test Score: 0.8625 | Train Score: 0.8500 | Random State: 276
Test Score: 0.9250 | Train Score: 0.8375 | Random State: 277
Test Score: 0.8750 | Train Score: 0.8469 | Random State: 282
Test Score: 0.8500 | Train Score: 0.8469 | Random State: 283
Test Score: 0.8500 | Train Score: 0.8438 | Random State: 285
Test Score: 0.9125 | Train Score: 0.8344 | Random State: 286
Test Score: 0.8500 | Train Score: 0.8406 | Random State: 290
Test Score: 0.8500 | Train Score: 0.8406 | Random State: 291
Test Score: 0.8500 | Train Score: 0.8469 | Random State: 292
Test Score: 0.8625 | Train Score: 0.8375 | Random State: 294
Test Score: 0.8875 | Train Score: 0.8281 | Random State: 297
Test Score: 0.8625 | Train Score: 0.8344 | Random State: 300
Test Score: 0.8625 | Train Score: 0.8500 | Random State: 301
Test Score: 0.8875 | Train Score: 0.8500 | Random State: 302
Test Score: 0.8750 | Train Score: 0.8469 | Random State: 303
Test Score: 0.8625 | Train Score: 0.8344 | Random State: 305
Test Score: 0.9125 | Train Score: 0.8375 | Random State: 306
Test Score: 0.8750 | Train Score: 0.8469 | Random State: 308
Test Score: 0.9000 | Train Score: 0.8438 | Random State: 311
Test Score: 0.8625 | Train Score: 0.8344 | Random State: 313
Test Score: 0.9125 | Train Score: 0.8344 | Random State: 314
Test Score: 0.8750 | Train Score: 0.8375 | Random State: 315
Test Score: 0.9000 | Train Score: 0.8469 | Random State: 317
Test Score: 0.9125 | Train Score: 0.8219 | Random State: 319
Test Score: 0.8625 | Train Score: 0.8500 | Random State: 321
Test Score: 0.9125 | Train Score: 0.8281 | Random State: 322
Test Score: 0.8500 | Train Score: 0.8469 | Random State:

328
Test Score: 0.8500 | Train Score: 0.8375 | Random State: 332
Test Score: 0.8875 | Train Score: 0.8531 | Random State: 336
Test Score: 0.8500 | Train Score: 0.8375 | Random State: 337
Test Score: 0.8750 | Train Score: 0.8406 | Random State: 343
Test Score: 0.8625 | Train Score: 0.8438 | Random State: 346
Test Score: 0.8875 | Train Score: 0.8313 | Random State: 351
Test Score: 0.8625 | Train Score: 0.8500 | Random State: 352
Test Score: 0.9500 | Train Score: 0.8187 | Random State: 354
Test Score: 0.8625 | Train Score: 0.8500 | Random State: 356
Test Score: 0.9125 | Train Score: 0.8406 | Random State: 357
Test Score: 0.8625 | Train Score: 0.8375 | Random State: 358
Test Score: 0.8500 | Train Score: 0.8406 | Random State: 362
Test Score: 0.9000 | Train Score: 0.8438 | Random State: 363
Test Score: 0.8625 | Train Score: 0.8531 | Random State: 364
Test Score: 0.9375 | Train Score: 0.8219 | Random State: 366
Test Score: 0.9125 | Train Score: 0.8406 | Random State: 369
Test Score: 0.8625 | Train Score: 0.8531 | Random State: 371
Test Score: 0.9250 | Train Score: 0.8344 | Random State: 376
Test Score: 0.9125 | Train Score: 0.8281 | Random State: 377

```
Test Score: 0.8875 | Train Score: 0.8500 | Random State: 378
Test Score: 0.8875 | Train Score: 0.8500 | Random State: 379
Test Score: 0.8625 | Train Score: 0.8406 | Random State: 382
Test Score: 0.8625 | Train Score: 0.8594 | Random State: 386
Test Score: 0.8500 | Train Score: 0.8375 | Random State: 387
Test Score: 0.8750 | Train Score: 0.8281 | Random State: 388
Test Score: 0.8500 | Train Score: 0.8438 | Random State: 394
Test Score: 0.8625 | Train Score: 0.8375 | Random State: 395
Test Score: 0.9000 | Train Score: 0.8438 | Random State: 397
Test Score: 0.8625 | Train Score: 0.8438 | Random State: 400
```

```
'\n\n\n'
```

```python
x_train,x_test,y_train,y_test=train_test_split(features,label,test_size=0.2,random_state=209)
finalModel=LogisticRegression()
finalModel.fit(x_train,y_train)
```

```
LogisticRegression()
```

```python
print(finalModel.score(x_train,y_train)
)
print(finalModel.score(x_train,y_train)
)
```

```
0.85
0.85
```

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

	precision	recall	f1-score	support
0	0.86	0.91	0.89	257
1	0.83	0.73	0.77	143
accuracy			0.85	400
macro avg	0.84	0.82	0.83	400
weighted avg	0.85	0.85	0.85	400