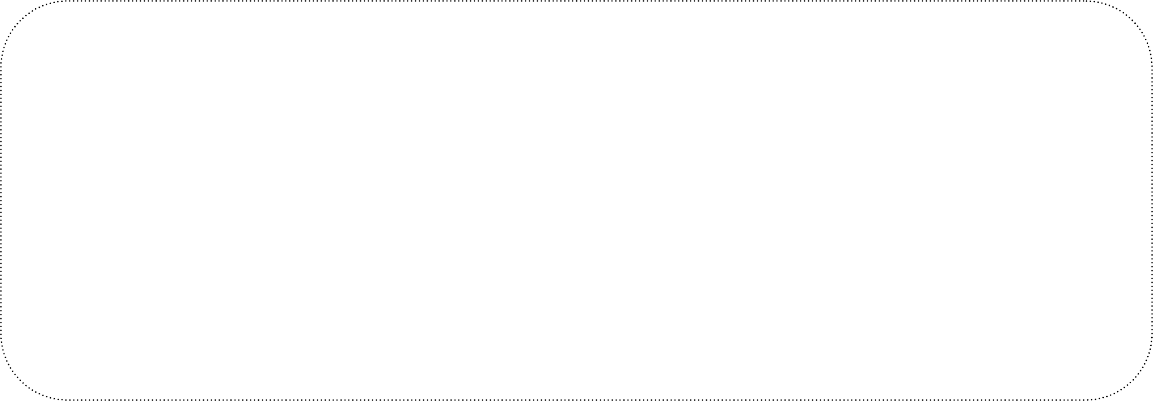


School of ComputingDEPARTMENTOFINFORMATIONTECHNOLOGY

DATASCIENCEANDDATAVISUALIZATION

(212INT3302)



Nameofthe Student :………………………….………………………………...……..

RegisterNo

:……………………………………………………….……………

Department

:…………………………….………………………………………

Year/Sem/Sec

:………………………………………………………….………...



**SchoolofComputing**

**DEPARTMENTOFINFORMATIONTECHNOLOGY**

**BONAFIDECERTIFICATE**

#### Bonafiderecordoftheworkdoneby……………………………………………...

………………………………………………….of………………………………………………………………

in**DataScienceandDataVisualization(212INT3302)**during**ODDsemester**

of**academicyear2024–2025**.

#### Staffin-charge HeadoftheDepartment

SubmittedtothePracticalExaminationheldatKAREon………………………

##### RegisterNumber

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |

**INTERNALEXAMINER EXTERNALEXAMINER**

**INDEX**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.No** | **Experimentname** | **PageNo.** | **Date ofExperiment** | **Marks** | **Sign** |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**ExpNo:01 Date:**

# PerformDataExplorationandpreprocessinginPython

**Aim:**

ToPerformDataExploration andpreprocessinginPython.

### Algorithm:

**Step 1:** Input the dataset and create a DataFrame with columns such as Name, Age, Gender,BP,and Cholesterol.

**Step2:**Fillmissingvalues.

For Age:Fill missing values with the mean.ForBP:Fillmissingvalueswiththemode.

ForCholesterol:Fillmissingvalueswiththemedian.

**Step 3:** Remove duplicate records. Check for and remove any duplicate rows in theDataFrame.

**Step 4:** Categorize data. Convert categorical values into numeric form for Gender and BP:Gender:Map 'F' to 0 and'M'to 1. BP: Map 'L' to0, 'M' to 1, and 'H'to 2.

**Step 5:** Normalize numeric data. Normalize Age and Cholesterol columns using min-maxnormalization.

**Step6:**Outputthecleanedandnormalizeddata. DisplaythefinalDataFrame.

### SourceCode:

import pandas as pdimportnumpyasnp

import matplotlib.pyplot as pltimportseabornas sns

data={

'Name':['Alice','Bob','Charlie','David','Eve','Alice'],'Age':[25, np.nan, 30,22, 25, 25],

'Sex':['F','M','M','M','F','F'],

'BP': ['High', 'Low', 'Medium', np.nan, 'High', 'High'],'Cholesterol':[np.nan,200, 180, 220,200, 200]

}

df=pd.DataFrame(data)

print("Original DataFrame:")print(df)

def fill\_missing\_values(df):df['Age'].fillna(df['Age'].mean(),inplace=True)

df['BP'].fillna(df['BP'].mode()[0], inplace=True)df['Cholesterol'].fillna(df['Cholesterol'].median(), inplace=True)returndf

df\_filled = fill\_missing\_values(df.copy())print("\nDataFrame after Filling Missing Values:")print(df\_filled)

def remove\_duplicates(df):df = df.drop\_duplicates()returndf

df\_no\_duplicates = remove\_duplicates(df\_filled.copy())print("\nDataFrame after Removing Duplicates:")print(df\_no\_duplicates)

defcategorize\_data(df):

df['Sex']=df['Sex'].map({'F':0,'M':1})

df['BP'] = df['BP'].map({'Low': 0, 'Medium': 1, 'High': 2})returndf

df\_categorized = categorize\_data(df\_no\_duplicates.copy())print("\nDataFrameafterCategorization:")print(df\_categorized)

defnormalize\_data(df):

numeric\_columns=['Age','Cholesterol']

df[numeric\_columns] = (df[numeric\_columns] - df[numeric\_columns].min()) /(df[numeric\_columns].max()-df[numeric\_columns].min())

returndf

df\_normalized = normalize\_data(df\_categorized.copy())print("\nDataFrameafterNormalization:")print(df\_normalized)

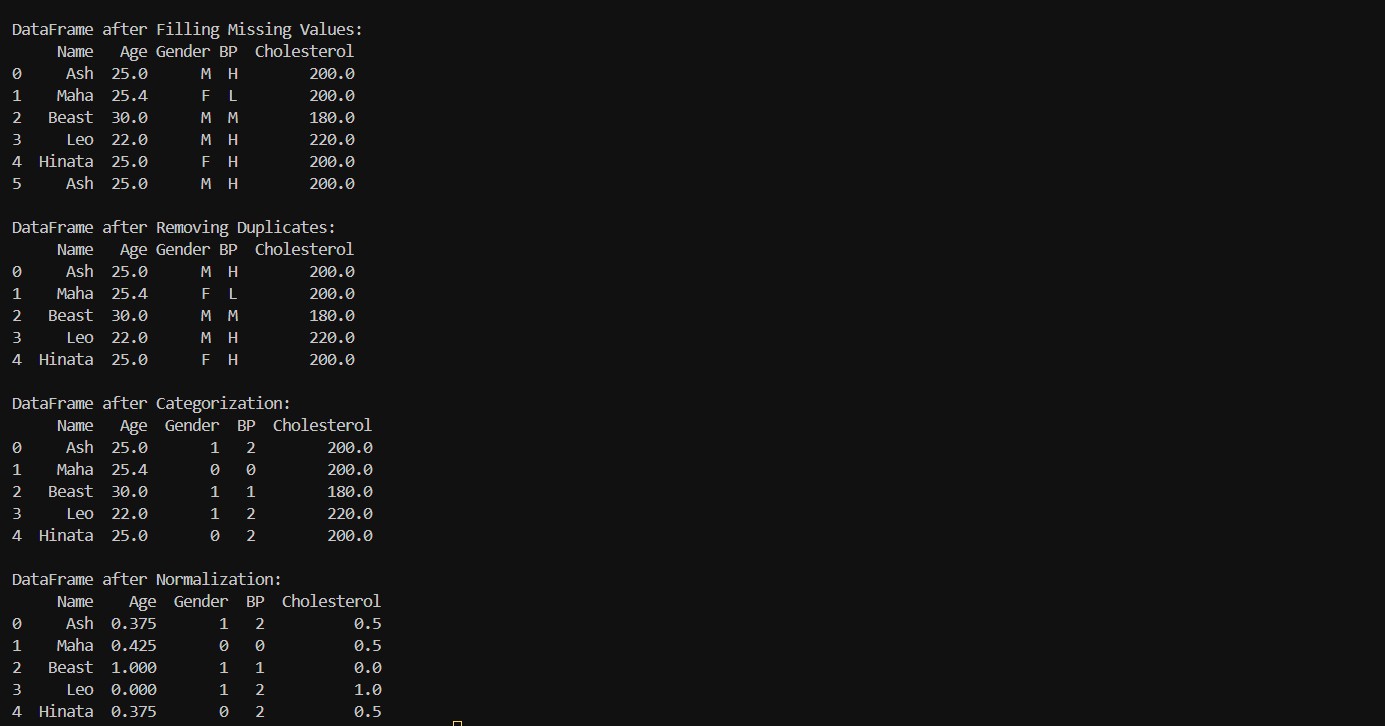
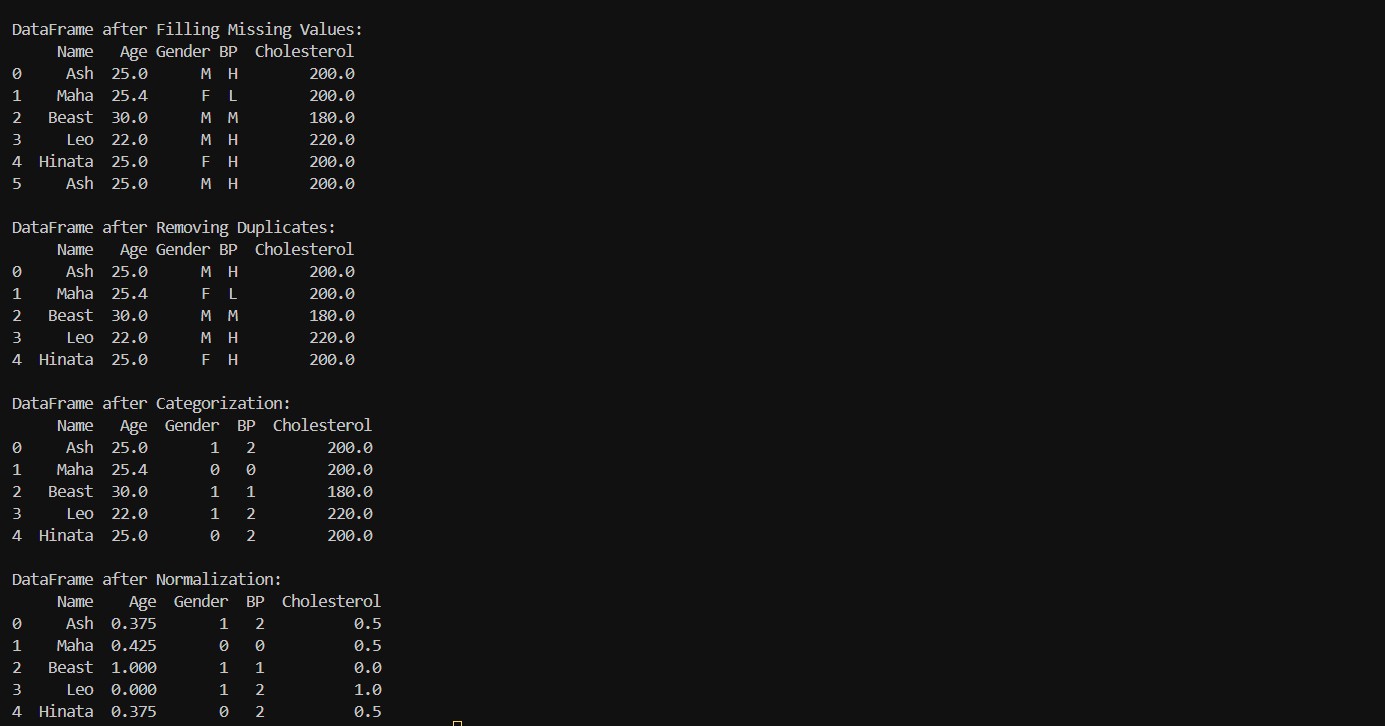
plt.figure(figsize=(8,6))

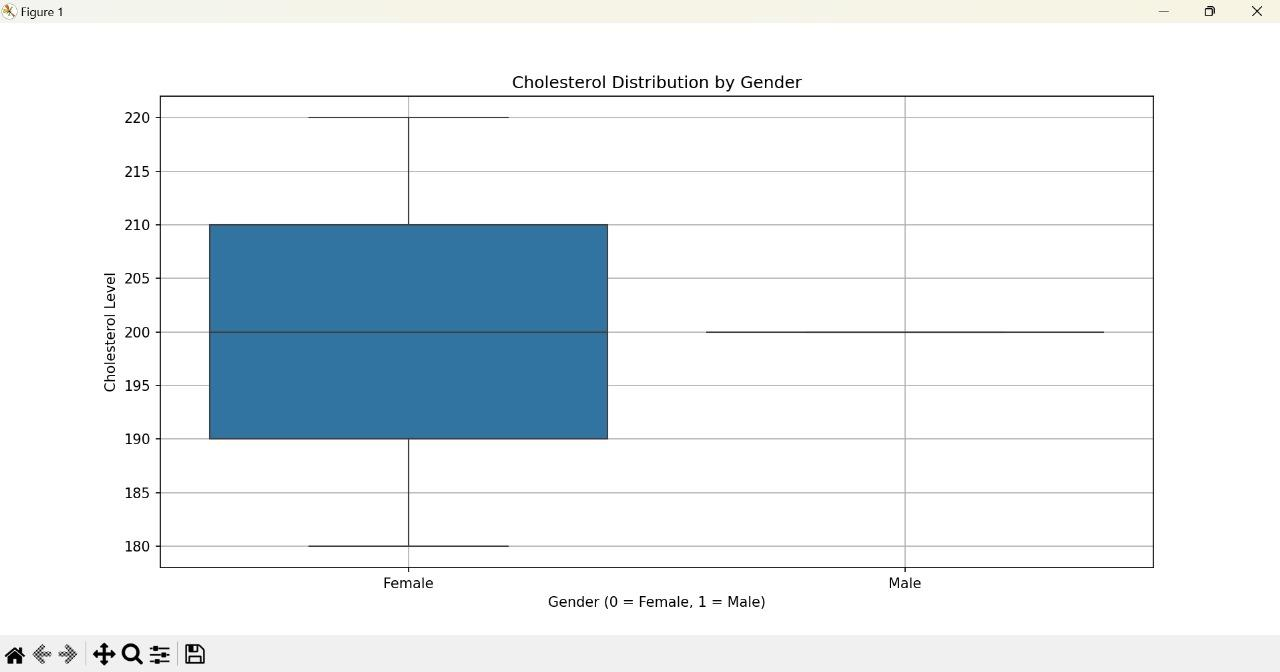
sns.boxplot(x='Gender', y='Cholesterol', data=data)plt.title('Cholesterol Distribution by Gender')plt.xlabel('Gender (0 = Female, 1 = Male)')plt.ylabel('CholesterolLevel')

plt.xticks(ticks=[0, 1], labels=['Female', 'Male'])plt.grid(True)

plt.show()

### Output:





**Result:**

ThuswehavePerformedDataExplorationandPreprocessinginpython.

**ExpNo:02 Date:**

# Performdataanalysisfrommultipledatasources

### Aim:

ToPerformdataanalysisfrommultiple datasources.

**Algorithm:**

1. **LoadData:**Loaddata fromCSV,Excel,andSQLitedatabaseintoDataFrames.
2. **MergeData:**MergetheDataFrames onacommonkey (e.g.,'id').
3. **Handle Missing Data:** Identify missing values in the merged DataFrame. Fill missingvalueswith adefault value(e.g., 0)or chooseanothermethod if appropriate.
4. **RemoveDuplicates:**DropanyduplicaterowsinthemergedDataFrame.
5. **Descriptive Statistics:** Compute and display descriptive statistics for the mergedDataFrame.
6. **Group Data:** Group the data by a specific column (e.g., 'category'). Calculate the mean ofnumericcolumns (e.g., 'sales', 'profit', 'feedback\_score') foreach group.
7. **FilterData:**Filterthe DataFrame basedonaspecificcondition (e.g.,'sales'>1000).

##### VisualizeData:

BarPlot:Plot abar chart showingsales bycategory.

CorrelationHeatmap:Selectonlynumericcolumns.Plotaheatmapofcorrelationsbetweennumericcolumns.

BoxPlot:Plot abox plotshowingthedistributionofsales by category.

PairPlot:Plotpairwiserelationshipsforselectednumericcolumns(e.g.,'sales','profit','feedback\_score').

### SourceCode:

import pandas as pdimport sqlite3

import matplotlib.pyplot as pltimportseabornas sns

csv\_data=pd.read\_csv('data1.csv')

excel\_data = pd.read\_excel('data2.xlsx')withsqlite3.connect('database.db')asconn:

sql\_data=pd.read\_sql\_query("SELECT\*FROMCustomerFeedback",conn)

merged\_data=pd.merge(csv\_data, excel\_data,on='id')

print("Merged Data:")print(merged\_data)

merged\_data.fillna(0, inplace=True)merged\_data.drop\_duplicates(inplace=True)

print("\nDescriptive Statistics:")print(merged\_data.describe())

grouped\_data = merged\_data.groupby('category').agg({'sales':'mean',

'profit': 'mean','feedback\_score':'mean'

})

print("\nGrouped Data:")print(grouped\_data)

filtered\_data = merged\_data[merged\_data['sales'] > 1000]print("\nFiltered Data where Sales > 1000:")print(filtered\_data)

plt.figure(figsize=(10,6))

sns.barplot(x='category', y='sales', data=merged\_data)plt.title('Salesby Category')

plt.show()

numeric\_data = merged\_data.select\_dtypes(include=['number'])plt.figure(figsize=(10,6))

sns.heatmap(numeric\_data.corr(), annot=True, cmap='coolwarm')plt.title('CorrelationHeatmap')

plt.show()plt.figure(figsize=(10,6))

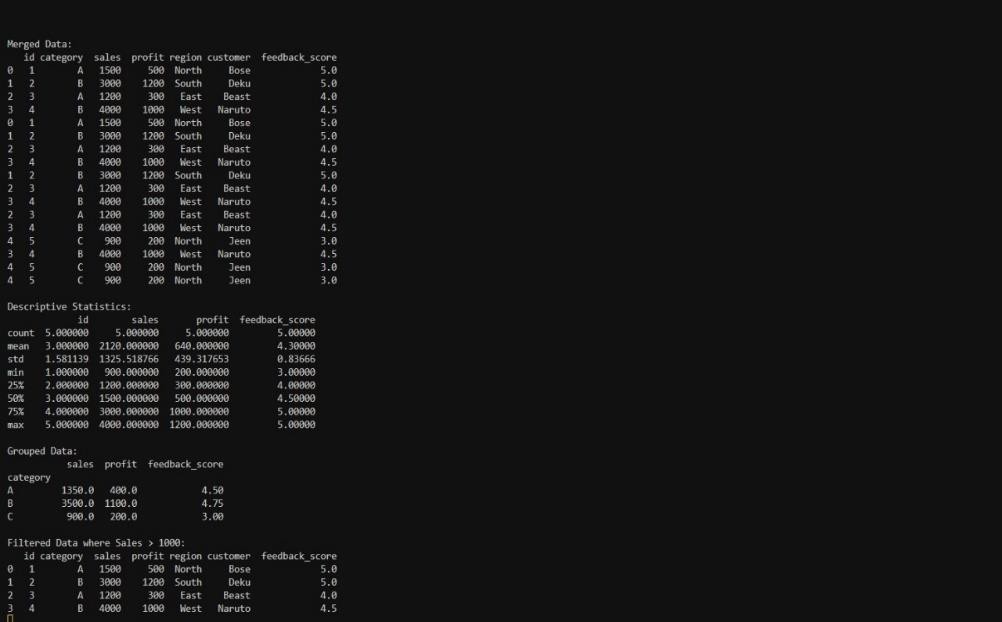
sns.boxplot(x='category', y='sales', data=merged\_data)plt.title('SalesDistribution byCategory')

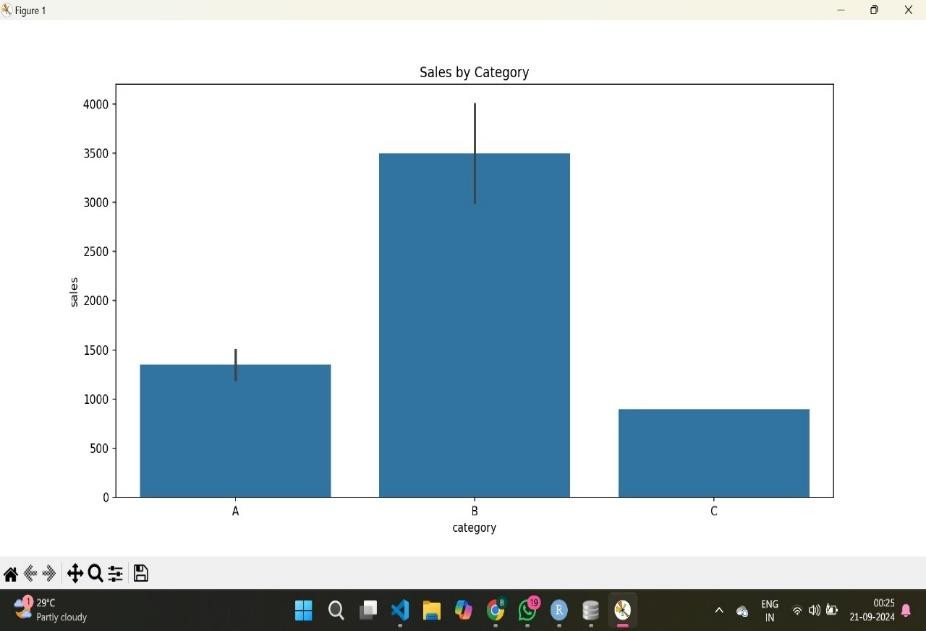
plt.show()plt.figure(figsize=(10,6))

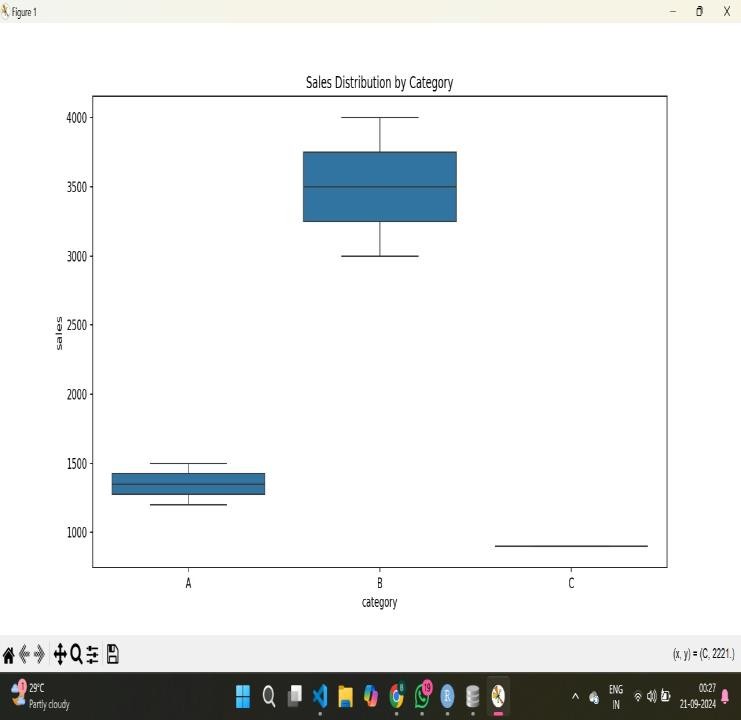
sns.pairplot(numeric\_data[['sales','profit','feedback\_score']])plt.title('Pairplot of Sales, Profit, and Feedback Score')plt.show()

print("Dataanalysisandvisualizations arecomplete.")

### Output :









**Result:**

ThuswePerformeddataanalysis frommultiple datasources.

**ExpNo:03 Date:**

# PerformUnivariantstatisticsUsingRprogramming

### Aim:

ToPerformUnivariantstatisticsUsingRprogramming.

1. **SummaryStatistics:**

**Algorithm:**

1. Inputthedata.
2. Calculatesummarystatistics:

**Mean:**Sumof allelementsdivided bythenumberofelements.

**Median:**Middlevalueinthesorteddataset.

**StandardDeviation(sd):**Measureoftheamountofvariationinthedata.

**Variance(var):**Thesquareof thestandard deviation.

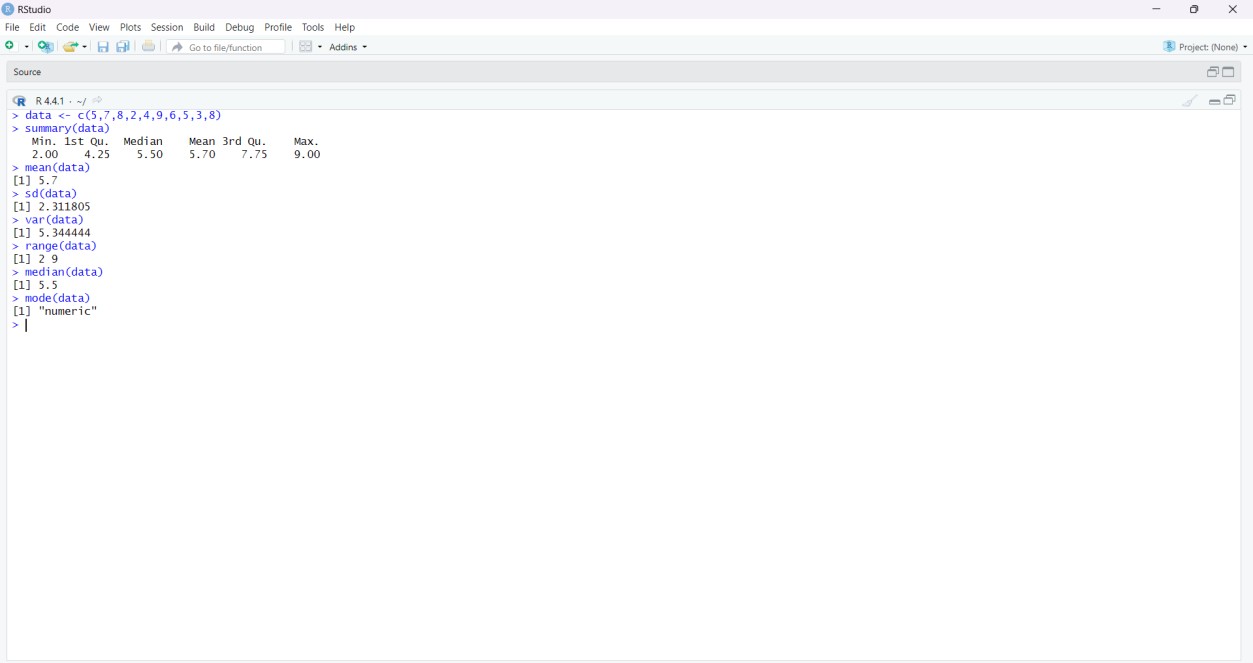
**Range:**Thedifferencebetween thelargest and smallest values.

### Sourcecode:

data<- c(5,7,8,2,4,9,6,5,3,8)

summary(data)mean(data)sd(data)var(data)range(data)median(data)mode(data)

### Output:



1. **Frequencytable :**

**Algorithm:**

1. Inputthedata.
2. Calculatefrequency:

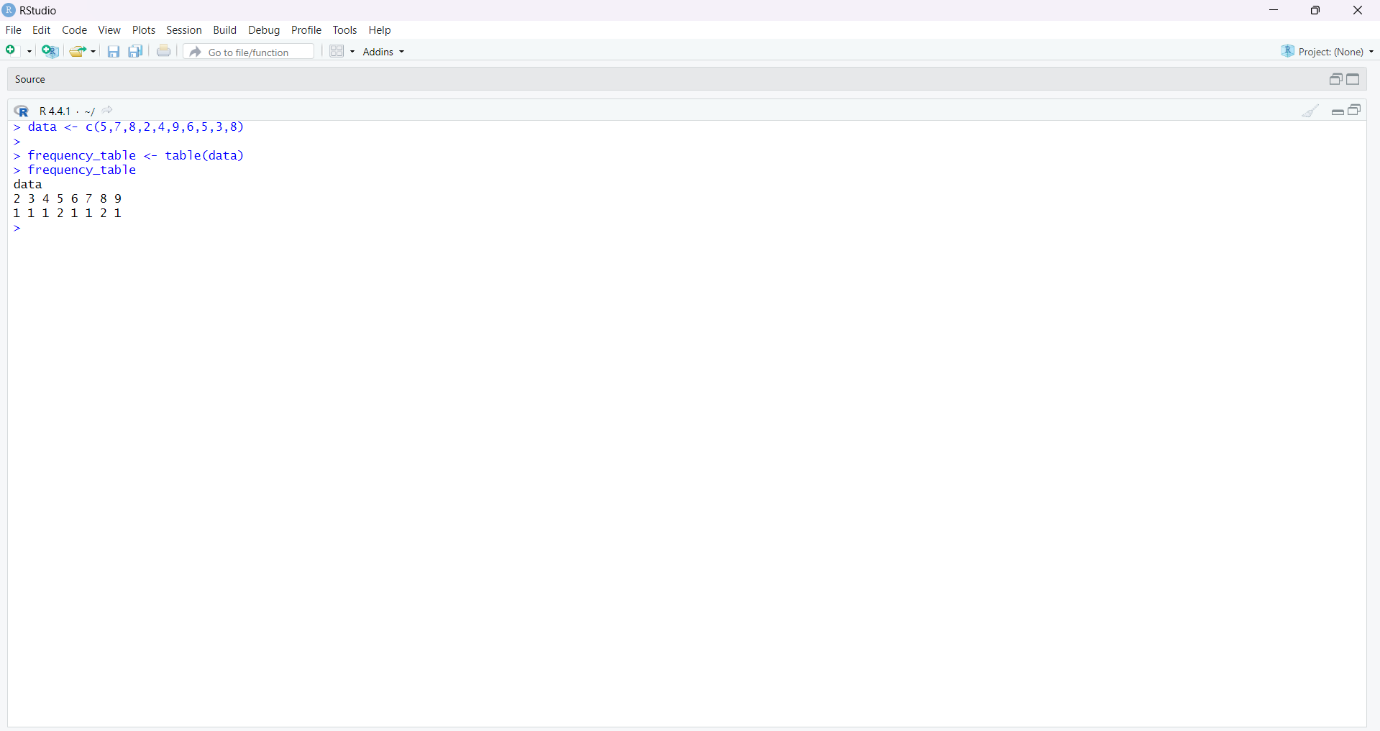
Use the table() function to count the frequency of each unique value in the dataset. Output thefrequencytable.

### Sourcecode:

data<- c(5,7,8,2,4,9,6,5,3,8)

frequency\_table <- table(data)frequency\_table

### Output:



1. **Charts:**

**Algorithm:**

1. Inputthedata.
2. Generateplots:

**Histogram:**Aplotto displaythe distribution ofthe data.

**BarPlot:**Abar chart representing the frequency of data points.

**Density Plot:** A smoothed version of the histogram showing the probability density of thedata.

### Sourcecode:

data<- c(5,7,8,2,4,9,6,5,3,8)

hist(data,main="HistogramofData",xlab="Value",ylab="Frequency",col="blue",border="black")

frequency\_table <-table(data)

barplot(data,main="BarPlotofData",xlab="Value",ylab="Frequency",col="lightgreen")boxplot(data,main="Box Plot of Data",ylab="value",col="lightcoral")plot(density(data),main="densityplotofdata",xlab="values",ylab="density")

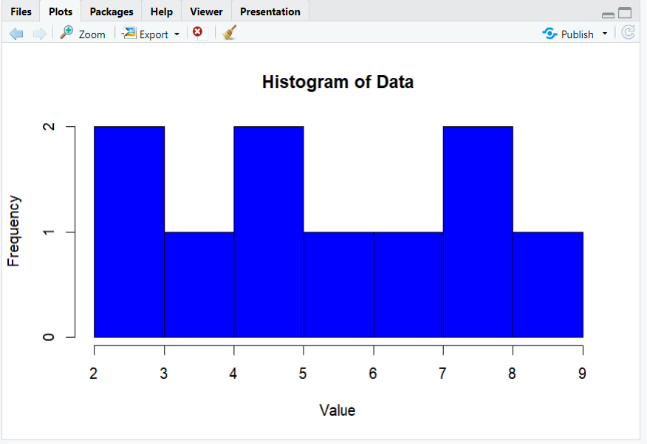
x <-c(210,450, 250, 100, 50, 90)

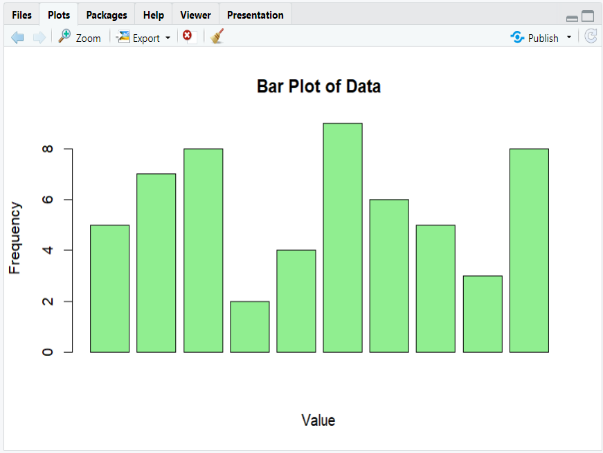
names(x) <- c("Algo", "DS", "Java", "C", "C++", "Python")png(file="piechart.png")

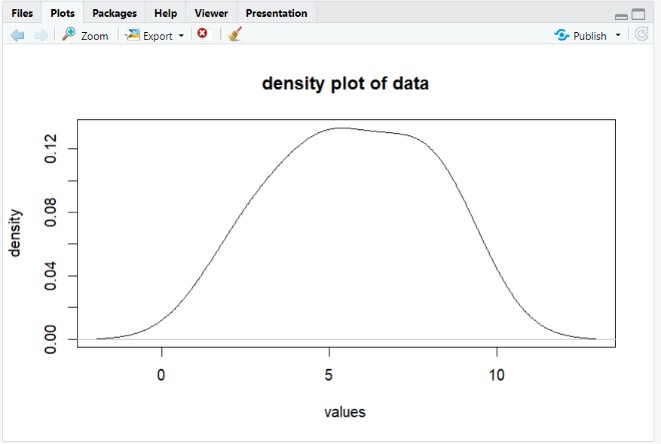
pie(x, labels = names(x), col = "white",main = "Articles", radius = -1,col.main= "darkgreen")

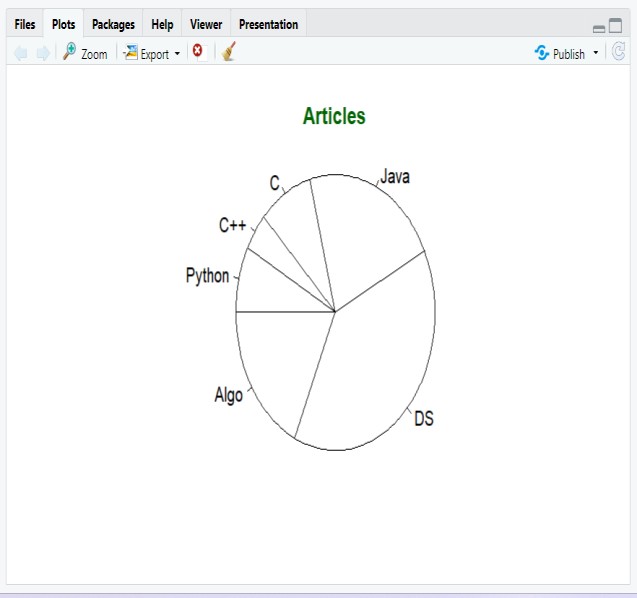
dev.off()

### Output:









**Result:**

ThusweperformedUnivariant statisticsUsingRprogramming.

**ExpNo:04 Date:**

# PerformMultivariateStatisticsUsingRProgramming

### Aim:

ToPerformMultivariate StatisticsUsingRProgramming

### Covariance:

**Algorithm:**

**Step1:**Inputthedata.

Definetwo vectorsXand Ycontaining numericvalues.

**Step 2:** Combine vectors into a data frame.Create a data frame data from vectors X and Y.**Step3:**Calculatethecovariancematrix.

Usethecov()function tocomputethecovariance matrix forthedata frame.

**Step 4:** Print the covariance matrix.Outputthecalculatedcovariancematrix.

### Sourcecode:

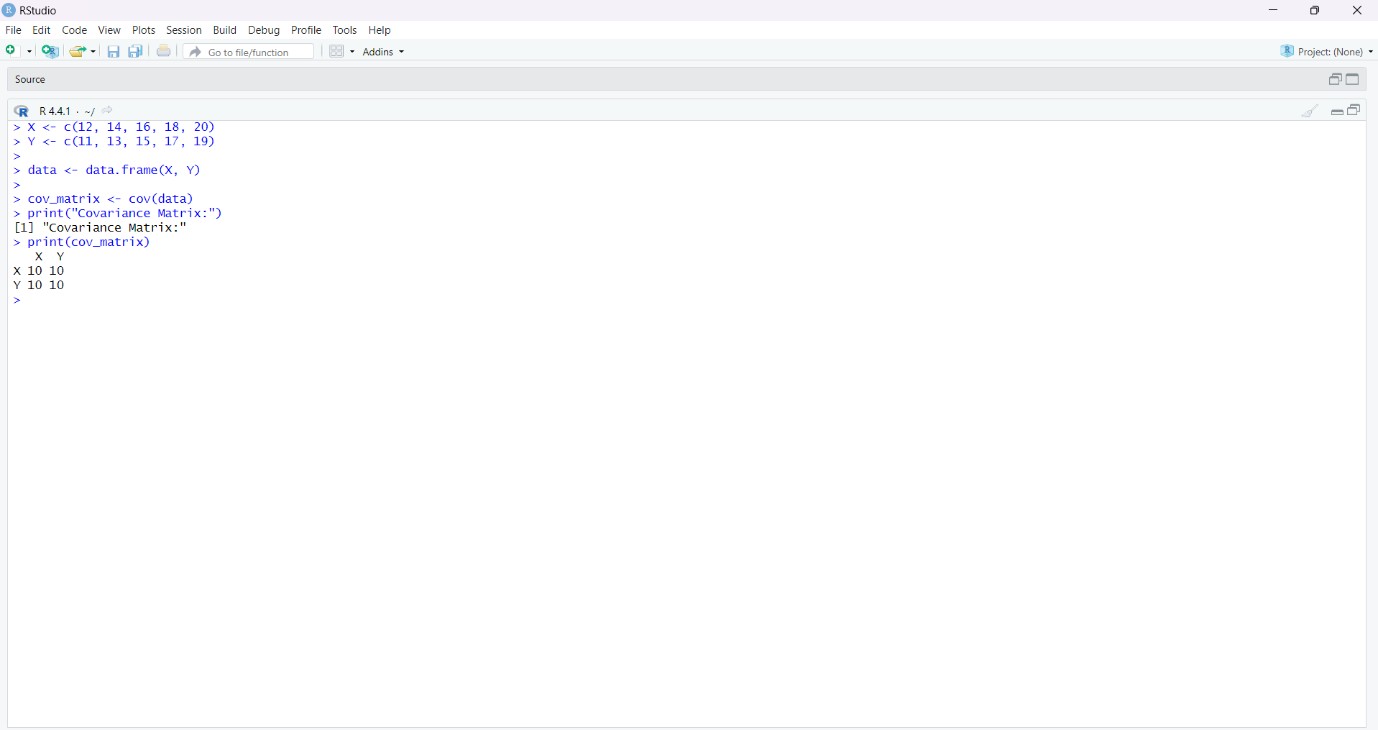
X<-c(12,14, 16, 18,20)

Y<-c(11,13, 15, 17,19)

data<-data.frame(X,Y)

cov\_matrix <- cov(data)print("Covariance Matrix:")print(cov\_matrix)

### Output:



1. **Correlation-Pearson, KendallandSpearmanCorrelationAlgorithm**

**Step1:**Inputthe data.

DefinetwonumericvectorsXandY.

**Step2:**Combinevectorsintoadataframe.CreateadataframedatacontainingXandY.

**Step3:**CalculatePearsoncorrelation.

Use the cor() function with the method "pearson" to compute the Pearson correlation betweenXandY.

**Step4:**CalculateKendallcorrelation.

Use the cor() function with the method "kendall" to compute the Kendall correlation betweenXandY.

**Step5:**CalculateSpearmancorrelation.

Use the cor() function with the method "spearman" to compute the Spearman correlationbetweenXandY.

**Step6:**Outputthecorrelations.

PrinttheresultsofPearson,Kendall,andSpearmancorrelations.

### Sourcecode:

pearson\_corr<-cor(data$X,data$Y,method="pearson")print("PearsonCorrelation:")

print(pearson\_corr)

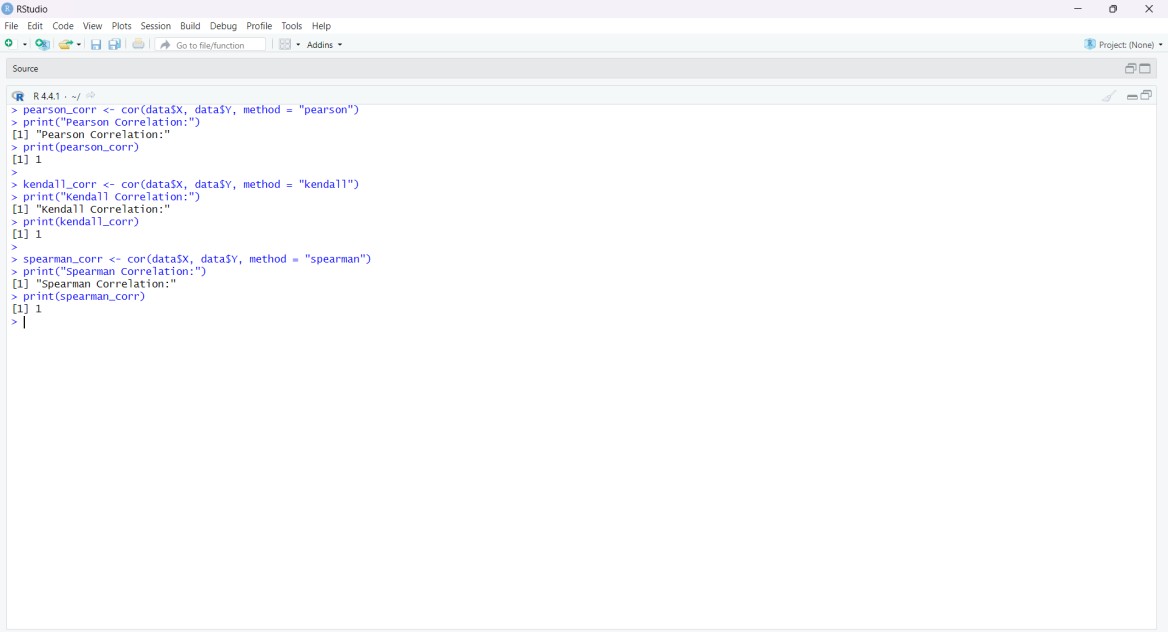
kendall\_corr<-cor(data$X,data$Y,method="kendall")print("KendallCorrelation:")

print(kendall\_corr)

spearman\_corr<-cor(data$X,data$Y,method="spearman")print("SpearmanCorrelation:")

print(spearman\_corr)

### Output:



**Result:**

ThuswePerformMultivariate StatisticsUsingRProgramming.

**ExpNo:05 Date:**

# DataAnalysiswithPython

### Aim:

ToPerformDataAnalysiswith Python.

### Covariance matrix:Algorithm:

**Step1:**Inputthedata.Define

twolistsofnumeric values,XandY.

**Step2:**Create aDataFrame.

UsethePandaslibrary tocombinethelists intoaDataFrame.

**Step3:**Calculatethecovariancematrix.

Usethecov()methodof theDataFrametocomputethecovariancematrix.

**Step4:**Outputthecovariancematrix.

Printthecovariancematrix todisplay theresult.

**SourceCode:**

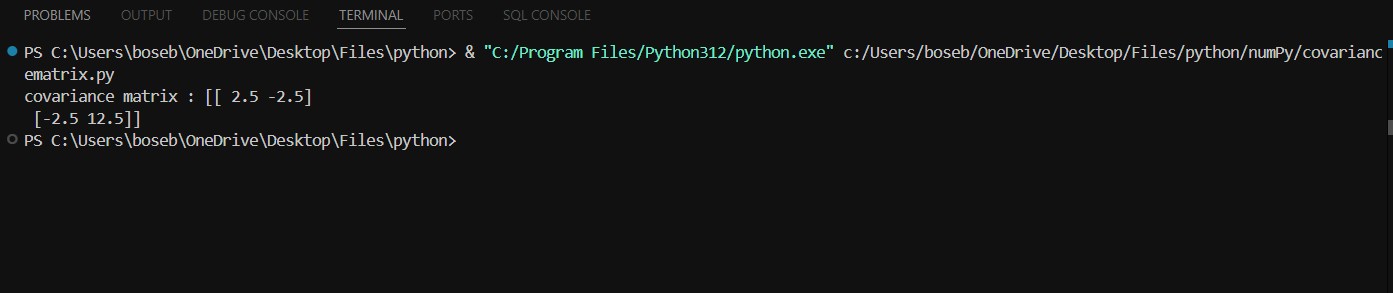
import numpy as npa=[1,2,3,4,5]

b=[6,7,8,9,0]

c=np.cov(a,b)

print(f'covariancematrix:{c}')

### Output:



1. **Partialandmultiplecorrelation:**

**Algorithm:**

Step1: Inputthedata.

DefinenumericdataforvariablesX,Y,andZ.

Step2: CreateaDataFrame.

CombinethedataintoaPandasDataFrame.

Step3:CalculatePartialCorrelation.

UsethepingouinlibrarytocomputethepartialcorrelationbetweenXandY,controllingforZ.

Step 4: Calculate Multiple Correlation.Preparethedataforregressionanalysis:

DefineX (independentvariables) andY(dependentvariable).AddaconstanttermtoXfortheintercept.

FitanOrdinaryLeastSquares(OLS)regression modelusingstatsmodels.

CalculatetheR-squaredvaluefromthemodeltoobtainthemultiplecorrelationsquared.

Step5:Outputtheresults.

Printthepartialcorrelationandthemultiplecorrelationsquared.

**Sourcecode:**

import pandas as pdimportpingouinaspg

importstatsmodels.apiassm

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| data={ |  | | | | |
| 'X': | [2, | 4, | 6, | 8, | 10], |
| 'Y': | [1, | 3, | 5, | 7, | 9], |
| 'Z': | [5, | 4, | 3, | 2, | 1] |

}

## df=pd.DataFrame(data)

partial\_corr = pg.partial\_corr(data=df, x='X', y='Y',covar='Z')

## print("Partial Correlation:")print(partial\_corr)

X = df[['X', 'Z']]Y =df['Y']

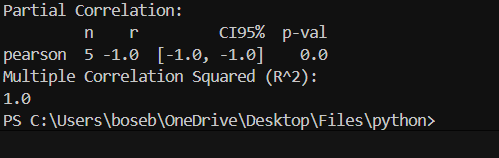
## X=sm.add\_constant(X)

model=sm.OLS(Y,X).fit()

## multiple\_corr\_squared=model.rsquared

print("Multiple Correlation Squared (R^2):")print(multiple\_corr\_squared)

### Output:



**Result:**

ThuswePerformedDataAnalysiswithPython.

**Exp No:06 Date :**

# DataVisualizationUsingR

### Aim:

ToPerformDataVisualizationUsingR

### ScatterPlot :

**Algorithm:**

**Step1:**Installand loadtherequiredpackage.

* + Checkifggplot2is installed.Ifnot,install it.
  + Loadtheggplot2library.

**Step2:**Preparethedata.

* + Setarandomseedforreproducibility.
  + Createadataframewithtwo continuousvariables(e.g.,xand y).

**Step3:**Createthescatter plot.

* + Useggplot()withgeom\_point()tocreate thescatterplot.
  + Addtitlesand labelsfortheaxes.

**Step4:**Displaytheplot.

* + Usetheprint()function to showthescatter plot.

### SourceCode:

if(!requireNamespace("ggplot2",quietly=TRUE)){install.packages("ggplot2")

}

library(ggplot2)

set.seed(42)

scatter\_data <- data.frame(x=rnorm(100),

y=rnorm(100)

)

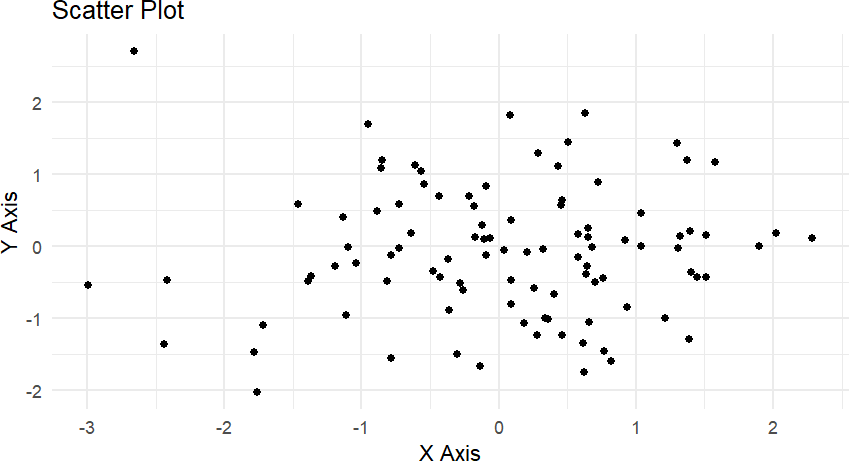
scatter\_plot<-ggplot(scatter\_data,aes(x=x,y= y))+

geom\_point()+

labs(title ="Scatter Plot", x ="XAxis", y ="YAxis") +theme\_minimal()

print(scatter\_plot)

### Output :



1. **BeeswarmPlot**

##### Algorithm:

**Step1:**Installandloadtherequiredpackages.

* + Checkif ggplot2and ggbeeswarmareinstalled. Ifnot, install them.
  + Loadbothggplot2andggbeeswarmlibraries.

**Step2:**Preparethedata.

* + Setarandomseedforreproducibility.
  + Createadataframewitha categoricalvariableandacontinuousvariable.

**Step3:**Createthebeeswarmplot.

* + Useggplot()with geom\_beeswarm()tocreatethebeeswarmplot.
  + Addtitlesand labelsfortheaxes.

**Step4:**Displaytheplot.

* + Usetheprint()function to show thebeeswarmplot.

### SourceCode:

if(!requireNamespace("ggplot2",quietly=TRUE)){install.packages("ggplot2")

}

if(!requireNamespace("ggbeeswarm",quietly=TRUE)){install.packages("ggbeeswarm")

}

library(ggplot2)library(ggbeeswarm)

set.seed(42)

categorical\_data<-data.frame(

category =rep(c("A","B","C"),each=30),

value=c(rnorm(30,mean =0),rnorm(30, mean=1), rnorm(30, mean=2))

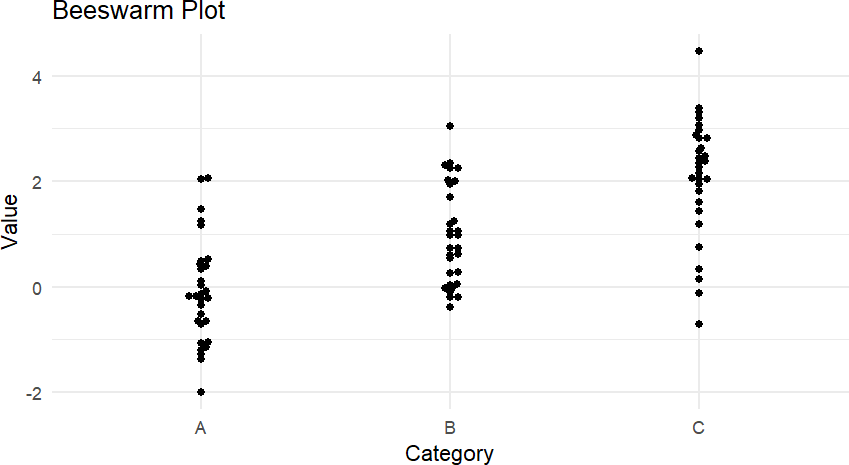
)

beeswarm\_plot<-ggplot(categorical\_data,aes(x=category,y=value))+geom\_beeswarm() +

labs(title="BeeswarmPlot",x="Category",y="Value")+theme\_minimal()

print(beeswarm\_plot)

### Output :



1. **ViolinPlot**

##### Algorithm:

**Step1:**Installand loadtherequiredpackage.

* + Checkifggplot2is installed.Ifnot,install it.
  + Loadtheggplot2library.

**Step2:**Preparethedata.

* + Usethesamedataframecreatedforthebeeswarmplot(categoricalvariableandcontinuousvariable).

**Step3:**Createtheviolinplot.

* + Useggplot()with geom\_violin()to create theviolin plot.
  + Addtitlesand labelsfortheaxes.

**Step4:**Displaytheplot.

* + Usetheprint()functiontoshowtheviolinplot.

### Sourcecode:

if(!requireNamespace("ggplot2",quietly=TRUE)){install.packages("ggplot2")

}

library(ggplot2)

set.seed(42)

categorical\_data<-data.frame(

category =rep(c("A","B","C"),each=30),

value=c(rnorm(30,mean =0),rnorm(30, mean=1), rnorm(30, mean=2))

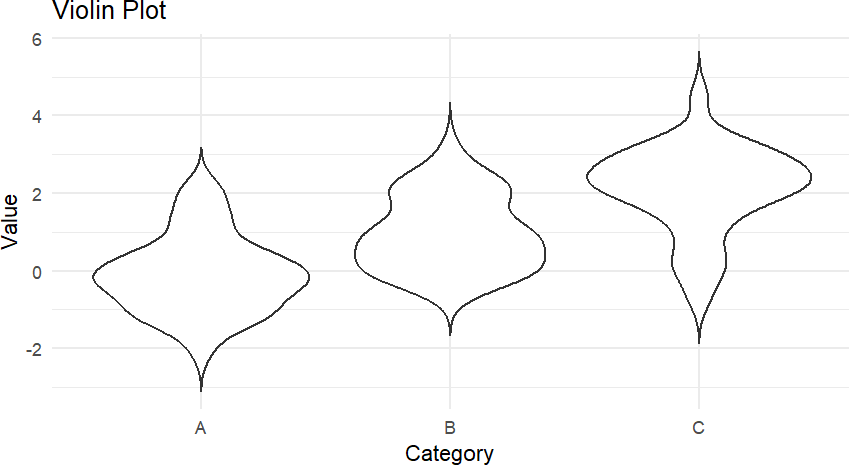
)

violin\_plot<-ggplot(categorical\_data,aes(x=category,y=value))+geom\_violin(trim=FALSE)+

labs(title="ViolinPlot",x="Category",y="Value")+theme\_minimal()

print(violin\_plot)

### Output :



#### Result :

ThusweSuccessfullyimplementedDataVisualizationUsingR

**Exp No:07 Date:**

# WorkingwithHeatMaps

### Aim:

ToworkwithHeatmaps

### CustomizedHeatmaps:

**Algorithm:**

**Step1:**Importthenecessarylibraries.

* + Importnumpyfornumericaloperations.
  + Importmatplotlib.pyplotforplotting.
  + Importmatplotlib.colorsforcustom colormaps.

**Step2:**Generate thedata.

* + Create an 8x8 array of random integers between 0 and 100 using

numpy.random.randint().

**Step3:**Definethecolormap.

* + Createalistof colors(e.g., ['#0099ff','#33cc33']).
  + CreateaListedColormapobject using the colorlist.

**Step4:**Createtheheatmap.

* + Useplt.imshow()todisplaythedataasanimagewiththespecifiedcolormapandvaluerange.
  + Settheextentto matchthedatadimensions.

**Step5:**Annotatetheheatmap.

* + Loopthrough thedata array.
  + Useplt.annotate()toplacetextannotationsatthecenterofeachcell,displayingthecorresponding data value.

**Step6:**Addacolorbar.

#### Createacolorbarwith plt.colorbar().

* + Setcustomticklabelsforthecolorbar(e.g.,['Low','Medium','High']).

**Step7:**Customizetheplot.

* + Add atitleto theplot.
  + Labelthex-axis andy-axis.

**Step8:**Displaytheplot.

* + Useplt.show()to render theheatmap.

### SourceCode:

import numpy as np

import matplotlib.pyplot as pltimportmatplotlib.colorsascolors

data = np.random.randint(0, 100, size=(8, 8))colors\_list=['#0099ff','#33cc33']

cmap=colors.ListedColormap(colors\_list)

plt.imshow(data, cmap=cmap, vmin=0, vmax=100, extent=[0, 8, 0, 8])fori in range(8):

forjinrange(8):

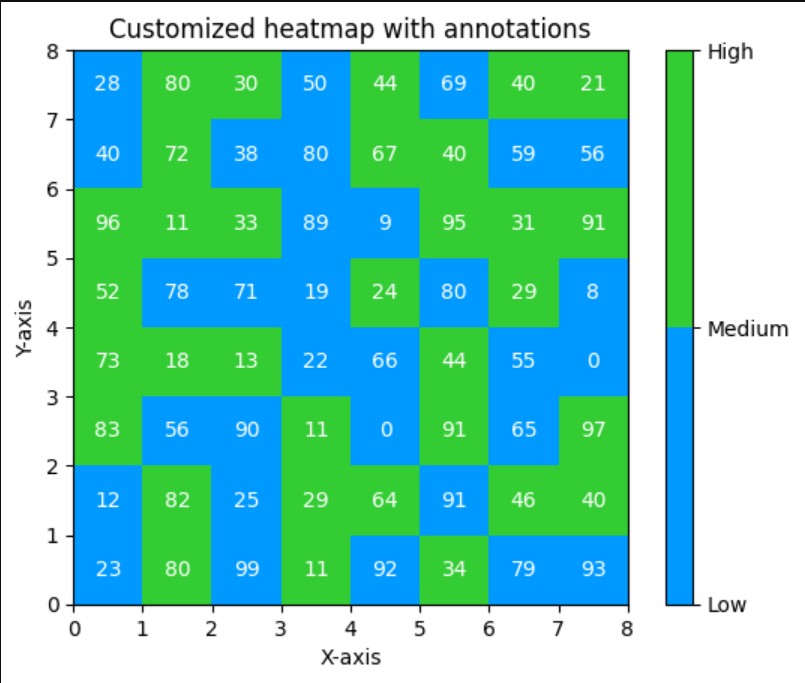
plt.annotate(str(data[i][j]), xy=(j + 0.5, i + 0.5),ha='center',va='center',color='white')

cbar = plt.colorbar(ticks=[0, 50, 100])cbar.ax.set\_yticklabels(['Low','Medium','High'])

plt.title("Customized Heatmap with Annotations")plt.xlabel("X-axis")

plt.ylabel("Y-axis")plt.show()

### Output :



**Result :**

ThuswesuccessfullyimplementedCustomizedHeatmap

### Exp No:8 Date:

**WorkingwithPythonPackages**

### Aim:

ToworkwithPythonPackages.

### ContourPlot

##### Algorithm:

**Step1:**Importthenecessarylibraries.

* + Importseaborn,matplotlib.pyplot, andnumpy.

**Step2:**Createdataforthecontour plot.

* + Generateagridof xandyvalues usingnumpy.meshgrid().
  + Calculate zvalues asafunction ofxand y.

**Step3:**Create the contourplot.

* + Use sns.contour()tovisualizethecontours.

**Step4:**Customizeand displaytheplot.

* + Addtitlesand labels,then displaytheplot withplt.show().

### SourceCode:

import numpy as np

importmatplotlib.pyplotasplt

x=np.linspace(-3, 3, 100)

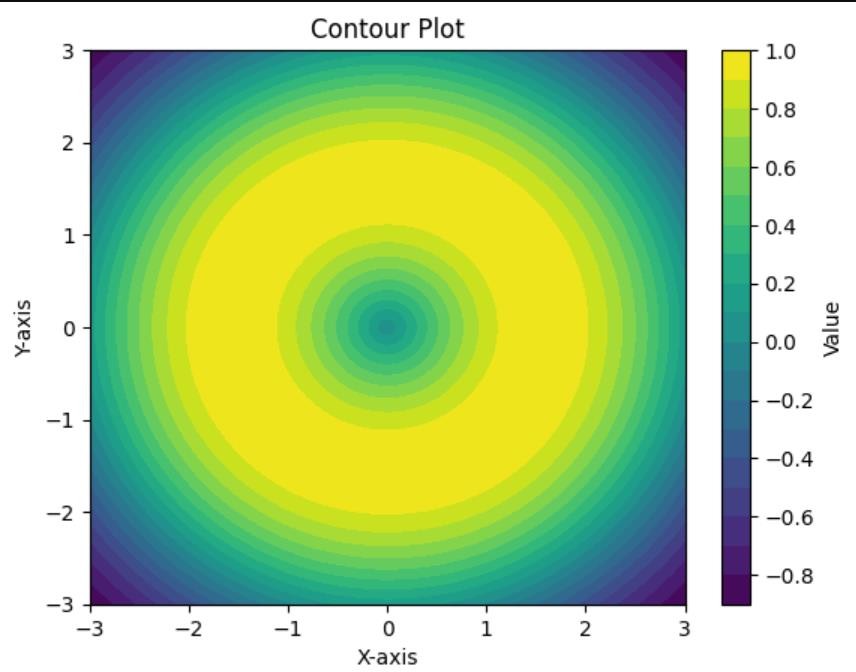
y = np.linspace(-3, 3, 100)X,Y=np.meshgrid(x,y)

Z=np.sin(np.sqrt(X\*\*2 +Y\*\*2))

plt.contourf(X,Y,Z,levels=20,cmap='viridis')

plt.title("Contour Plot")plt.xlabel("X-axis")plt.ylabel("Y-axis")plt.colorbar(label='Value')plt.show()

### Output :



1. **Box PlotAlgorithm:**

**Step1:**Importnecessarylibraries.

* + Useseabornandmatplotlib.pyplot.
  + Usenumpytogeneratesyntheticdata.

**Step2:**Generatesyntheticdatafortheboxplot.

* + CreateaDataFramewithrandom datagroupedbycategories.

**Step3:** CreatetheboxplotusingSeaborn.

### SourceCode:

importseabornassns

import matplotlib.pyplot as pltimportpandasaspd

np.random.seed(42)data={

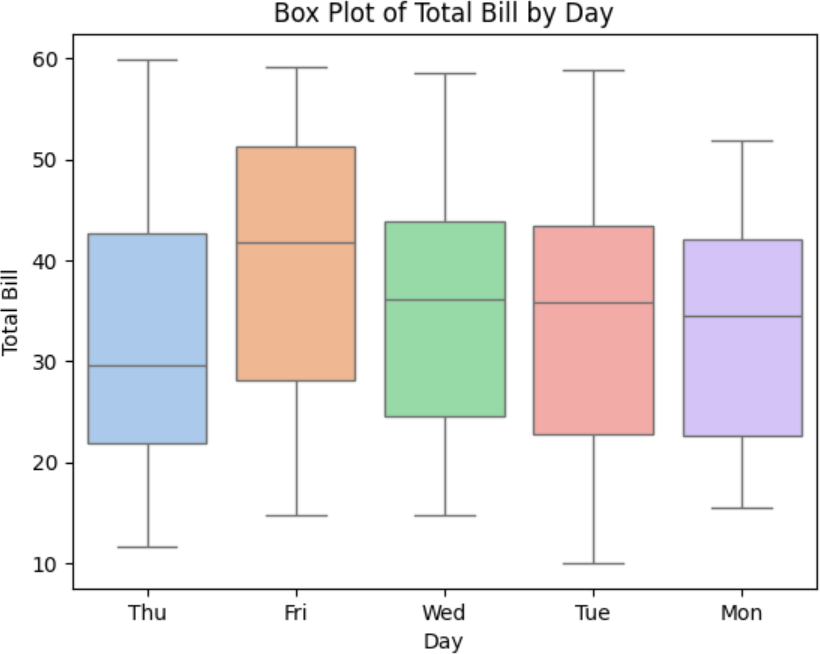
'day':np.random.choice(['Mon','Tue','Wed','Thu','Fri'],100),'total\_bill':np.random.uniform(10, 60, 100)

}df=pd.DataFrame(data)

sns.boxplot(x='day', y='total\_bill', data=df, palette='pastel')plt.title("BoxPlotofTotalBill byDay")

plt.xlabel("Day")plt.ylabel("Total Bill")plt.show()

### Output :



1. **Pair PlotAlgorithm:**

**Step1:**Importnecessarylibraries.

**Step2:**Generatesyntheticdataforthepairplot.

**Step3:**Createthepairplot usingSeaborn.

### SourceCode:

importseabornassns

import matplotlib.pyplot as pltimportpandasaspd

np.random.seed(42)num\_samples = 100data={

'feature1': np.random.normal(5, 1, num\_samples),'feature2': np.random.normal(10, 2, num\_samples),'feature3':np.random.normal(15,3,num\_samples),

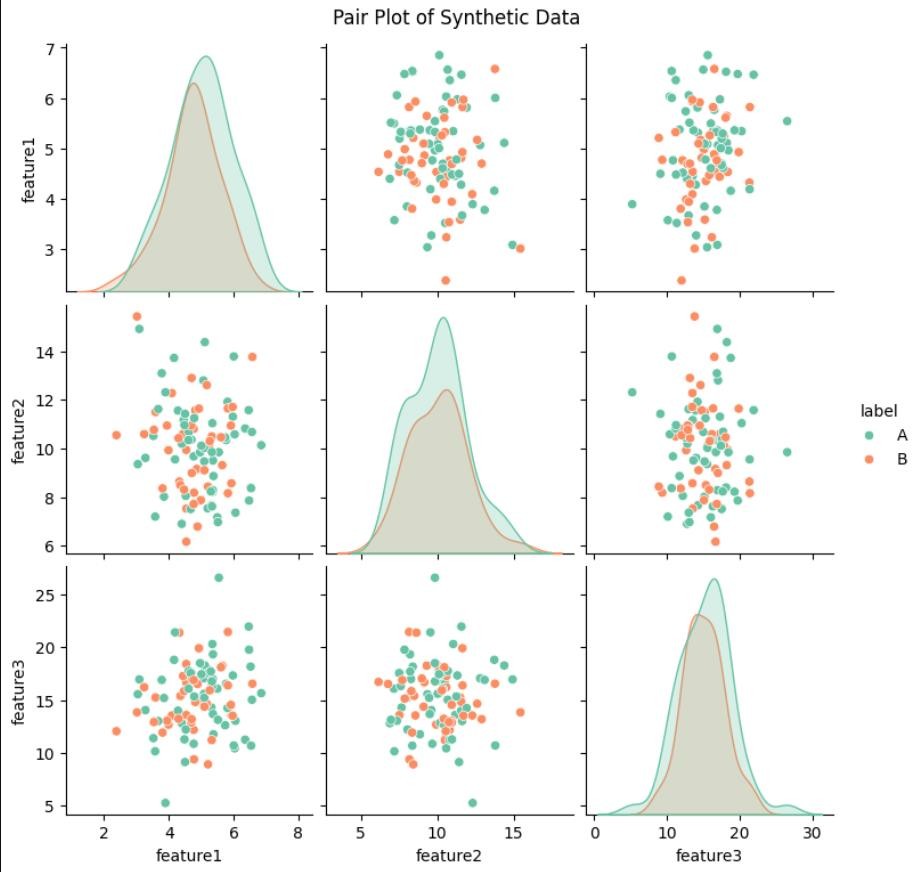
'label':np.random.choice(['A','B'],num\_samples)

}

df=pd.DataFrame(data)

sns.pairplot(df, hue='label', palette='Set2')plt.suptitle("Pair Plot of Synthetic Data", y=1.02)plt.show()

### Output :



1. **Violin PlotAlgorithm:**

**Step1:**Importnecessarylibraries.

**Step2:**Generatesyntheticdatafortheviolinplot.

* + CreateaDataFramewithrandomdatagroupedbycategories.

**Step3:**Createtheviolinplotusing Seaborn.

### Sourcecode:

importseabornassns

importmatplotlib.pyplotasplt

importpandasaspd

np.random.seed(42)data={

'day':np.random.choice(['Mon','Tue','Wed','Thu','Fri'],100),'total\_bill':np.random.uniform(10, 60, 100)

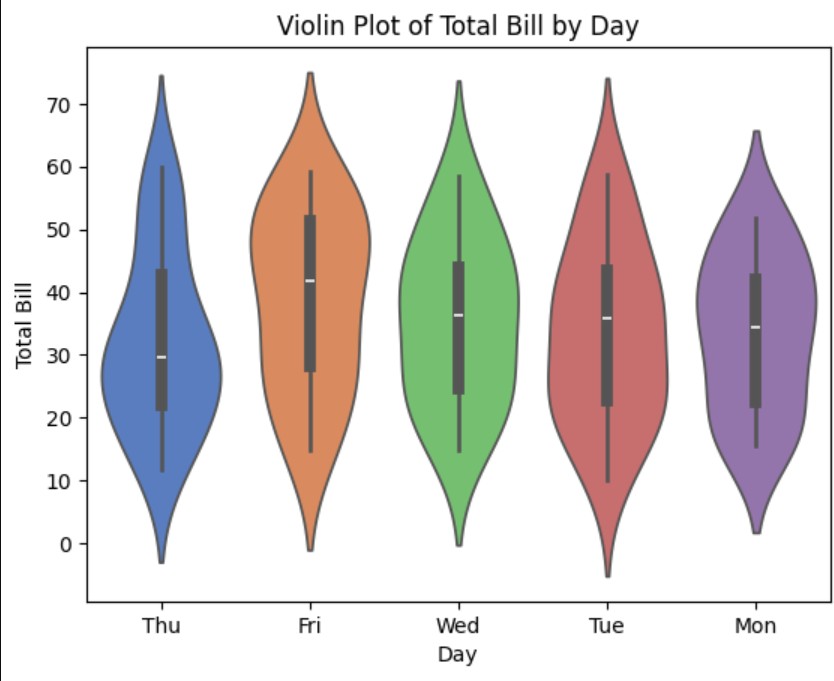
}

df=pd.DataFrame(data)

sns.violinplot(x='day',y='total\_bill',data=df,palette='muted')plt.title("ViolinPlotofTotalBillbyDay")

plt.xlabel("Day")plt.ylabel("Total Bill")plt.show()

### Ouput:



**Result :**

Thuswesuccessfullyworkedwithpythonpackagestoimplementvariousplots.

### Exp No:9 Date:

**WorkingonANOVAtool**

### Aim:

ToWorkwithANOVAtool

### Anova Test (F - Test)Algorithm

**Step1:**Importnecessarylibraries.

* + Importnumpyfor array handling.
  + Importstatsfrom scipyforstatistical functions.

**Step2:**Definethedataforeachgroup.

* + Createthreearraysrepresentingthedifferentfertilizers.

**Step3:**PerformANOVA.

* + Use stats.f\_oneway()tocomputethe F-statisticforthegroups.

**Step4:**Calculatedegreesoffreedom.

* + Determinethenumberofgroupskkk andthetotalnumberofobservations nnn.
  + Computedegreesoffreedomforthenumeratordfn=k−1\text{dfn}=k- 1dfn=k−1andforthedenominator dfd=n−k\text{dfd}=n-kdfd=n−k.

**Step5:**Determinethecriticalvalue.

* + Setthesignificancelevelα\alphaα.
  + Usestats.f.ppf()tofindthecriticalvaluebasedonthedegreesoffreedomandsignificancelevel.

**Step6:**ComparetheF-statistictothecriticalvalue.

* + PrinttheF-statisticandcriticalvalue.
  + Decidewhethertorejector failtorejectthenullhypothesisbasedon thecomparison.

### SourceCode:

import numpy as npfromscipyimportstats

fertilizer1 =np.array([6, 8, 4,5,3,4])

fertilizer2=np.array([8,12,9,11,6,8])

fertilizer3=np.array([13,9,11,8,7,12])

F\_statistic,\_=stats.f\_oneway(fertilizer1,fertilizer2,fertilizer3)

k =3

n = len(fertilizer1) + len(fertilizer2) + len(fertilizer3)dfn =k-1

dfd =n-k

alpha=0.05

critical\_value=stats.f.ppf(1-alpha,dfn,dfd)

print(f"F-statistic:{F\_statistic:.4f}")

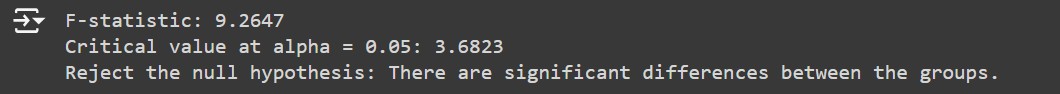
print(f"Criticalvalueatalpha={alpha}:{critical\_value:.4f}")

ifF\_statistic>critical\_value:

print("Rejectthenullhypothesis:Therearesignificantdifferencesbetweenthegroups.")else:

print("Failtorejectthenullhypothesis:Nosignificantdifferencesbetweenthegroups.")

### Output :



**Result :**

ThuswesuccessfullyimplementedANOVATest.

### ExNo:10 Date :

**Datavisualizationandsimulations**

### Aim:

ToPerformDatavisualizationandSimulationsUsingPython.

### 3D Line PlotAlgorithm:

**Step1**:Importnecessarylibraries.

* + Importnumpyfornumericaloperations.
  + Importmatplotlib.pyplotforplotting.
  + Importmplot3dfrommpl\_toolkitsfor3D plottingcapabilities.

**Step2**: Createafigurefortheplot.

* + Initializeafigureusing plt.figure().

**Step3**: Setup a3Dprojection.

* + Useplt.axes(projection='3d')tocreatea3Daxis.

**Step4**:Definethe3Dcoordinates.

* + Createanarrayzrangingfrom 0to 1 with 100 points.
  + Compute xandycoordinates usingparametricequationsinvolvingz.

**Step5:**Plotthe 3Dline.

* + Useax.plot3D(x,y,z,'color')to plotthe linein3Dspace.

**Step6:**Setthetitleoftheplot.

* + Useax.set\_title()to providea titleforthe plot.

**Step7:**Displaytheplot.

* + Useplt.show()to rendertheplot.

### SourceCode:

from mpl\_toolkits import mplot3dimport numpy as np

importmatplotlib.pyplotasplt

fig=plt.figure()

#Set upthe3Dprojection

ax=plt.axes(projection='3d')

#Definethe axes

z = np.linspace(0, 1, 100)x =z\* np.sin(25\* z)

y=z\*np.cos(25 \*z)

ax.plot3D(x,y,z,color='green')ax.set\_title('3D Line Plot')ax.set\_xlabel('X-axis')ax.set\_ylabel('Y-axis')ax.set\_zlabel('Z-axis')

plt.show()

### Output :

1. **Simulated3DSpiralCurves**

**Step1:**Importnecessarylibraries.

* + Importnumpyfornumericaloperations.
  + Importmatplotlib.pyplotforplotting.
  + Importmplot3dfrommpl\_toolkitsfor3D plottingcapabilities.

**Step2:**Defineafunction togeneratea3Dspiral.

* + Thefunctiongenerate\_spiraltakesparametersforamplitude,frequency,andthenumberofpoints.
  + Insidethefunction,createalinearlyspacedarrayforz,thencomputexandyusingparametricequations involvingz.

**Step3:**Createafigure for theplot.

* + Initializeafigureusing plt.figure().

**Step4:**Set upa 3Dprojection.

* + Useplt.axes(projection='3d')tocreatea3Daxis.

**Step5:**Generatemultiplespirals.

* + Loopthrough aspecifiednumberof curves (num\_curves).
  + In eachiteration,generaterandomvaluesforamplitudeandfrequency.
  + Callthegenerate\_spiralfunctiontogetx,y,andzcoordinates.
  + Ploteachspiralusing ax.plot3D().

**Step6:**Set thetitleandlabelsfortheplot.

* + Use ax.set\_title(), ax.set\_xlabel(), ax.set\_ylabel(), and

ax.set\_zlabel().

**Step7:**Add alegend.

* + Use ax.legend()todistinguishbetweendifferentcurves.

**Step8:**Displaytheplot.

* + Useplt.show()to rendertheplot.

### SourceCode:

from mpl\_toolkits import mplot3dimport numpy as np

importmatplotlib.pyplotasplt

#Functiontogeneratea3Dspiral

defgenerate\_spiral(amplitude,frequency,num\_points=100):z=np.linspace(0, 1,num\_points)

x = amplitude \* z \* np.sin(frequency \* z)y = amplitude \* z \* np.cos(frequency \* z)returnx, y,z

# Create a figurefig=plt.figure()

ax=plt.axes(projection='3d')

# Generate multiple spirals with different amplitudes and frequenciesnum\_curves=5

foriinrange(num\_curves):

amplitude = np.random.uniform(0.5, 1.5)# Random amplitudefrequency = np.random.uniform(20, 30) # Random frequencyx,y,z=generate\_spiral(amplitude,frequency)

ax.plot3D(x,y,z,label=f'Curve{i+1}')

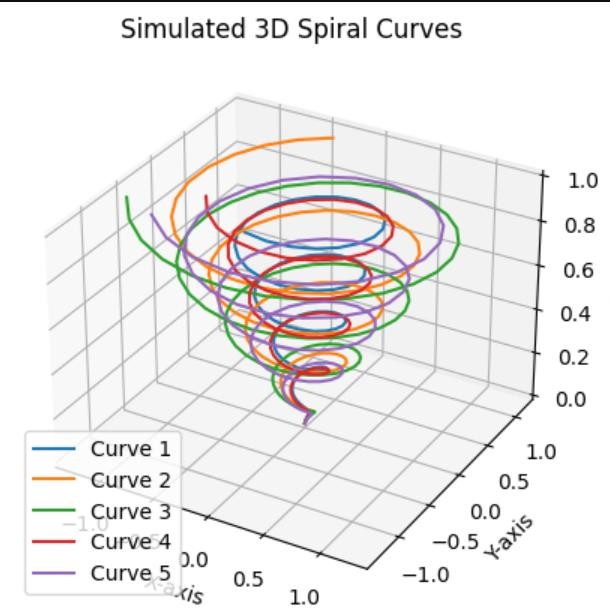
# Setting title and labelsax.set\_title('Simulated 3D Spiral Curves')ax.set\_xlabel('X-axis')

ax.set\_ylabel('Y-axis')ax.set\_zlabel('Z-axis')

# Add a legendax.legend()

# Show the plotplt.show()

### Output :



**Result :**

ThusweimplementedDatavisualizationandsimulationUsingPython.

**ExpNo:11 Date :**

# StudyofGISsystemsforDataAnalysis

### Aim:

TostudytheGISsystemsforDataAnalysis.

### Algorithm:

##### InitializeEnvironment

* 1. Importnecessarylibraries(pandas,geopandas,matplotlib,numpy,requests,zipfile, io).

##### LoadWeatherData

* 1. Definea functiontoloadCSVdataintoaDataFrame.
  2. Validatethepresenceofrequiredcolumns(e.g.,Date/Time,Temp\_C,Weather).

##### DataPreparation

* 1. Checkforandhandlemissing values(droporfill).
  2. Convertdate/timecolumntoappropriatedatetimeformat.
  3. Generateorextractlatitudeandlongitudecolumnsifnotpresent.

##### CreateGeoDataFrame

* 1. ConverttheDataFrametoaGeoDataFrameusinglatitudeand longitude.

1. **DataAnalysis**
   1. Calculateaveragetemperatureandotherstatisticsbyweathercondition.
   2. Identifyextremeweatherconditions(e.g.,highesttemperature,lowestvisibility).

##### DataVisualization

* 1. Loadaworldmapshapefile.
  2. Plotweatherconditionsontheworldmapusingdifferentmarkersforeachcondition.
  3. Createatimeseriesplotoftemperaturevariationsovertherecordedtime.
  4. Generatescatterplotstoanalyzecorrelations(e.g.,humidityvs.windspeed).

##### GenerateInsights

* 1. Summarizekeyfindingsandinsightsbasedontheanalysis.
  2. Identifytrendsovertimeorrelationshipsbetweenweathervariables.

##### SaveResults

* 1. Savevisualizationsandstatisticalsummariestofiles(e.g.,PNG,CSV).

##### Main Function

* 1. Coordinateallfunctions,ensuringsmoothexecution.

### SourceCode:

import pandas as pdimportgeopandasasgpd

import matplotlib.pyplot as pltimport numpy as np

import requestsimport zipfileimport io

def load\_weather\_data(file\_path):df = pd.read\_csv(file\_path)returndf

defprepare\_data(df):

print("NullValuesinEachColumn:")print(df.isnull().sum())df.dropna(inplace=True)

df['Latitude'] = np.random.uniform(-90, 90, df.shape[0])df['Longitude'] = np.random.uniform(-180, 180, df.shape[0])returndf

defcreate\_geodataframe(df):

gdf = gpd.GeoDataFrame(df, geometry=gpd.points\_from\_xy(df['Longitude'],df['Latitude']))

returngdf

defanalyze\_weather\_conditions(gdf):

avg\_temp=gdf.groupby('Weather')['Temp\_C'].mean().reset\_index()print("Average Temperature by Weather Condition:")print(avg\_temp)

defvisualize\_weather\_conditions(gdf):

url =

'https://naturalearth.s3.amazonaws.com/110m\_cultural/ne\_110m\_admin\_0\_countries.zip'response= requests.get(url)

with zipfile.ZipFile(io.BytesIO(response.content)) as z:z.extractall('naturalearth\_data')

world=gpd.read\_file('naturalearth\_data/ne\_110m\_admin\_0\_countries.shp')fig,ax =plt.subplots(figsize=(15, 10))

world.plot(ax=ax,color='lightgrey')

forconditioningdf['Weather'].unique():

gdf[gdf['Weather']==condition].plot(ax=ax,marker='o',markersize=5,label=condition)plt.title('WeatherConditions on Map')

plt.legend()plt.show()

defplot\_temperature\_over\_time(df):df['Date/Time']=pd.to\_datetime(df['Date/Time'])plt.figure(figsize=(12, 6))

plt.plot(df['Date/Time'],df['Temp\_C'],label='Temperature(°C)',color='blue')plt.title('TemperatureVariationOverTime')

plt.xlabel('Date/Time')plt.ylabel('Temperature (°C)')plt.xticks(rotation=45)plt.grid()

plt.legend()plt.show()

def plot\_humidity\_vs\_wind\_speed(df):plt.figure(figsize=(10, 6))

plt.scatter(df['RelHum\_%'],df['WindSpeed\_km/h'],alpha=0.5)plt.title('Humidityvs.WindSpeed')

plt.xlabel('Relative Humidity (%)')plt.ylabel('Wind Speed (km/h)')plt.grid()

plt.show()

defmain():

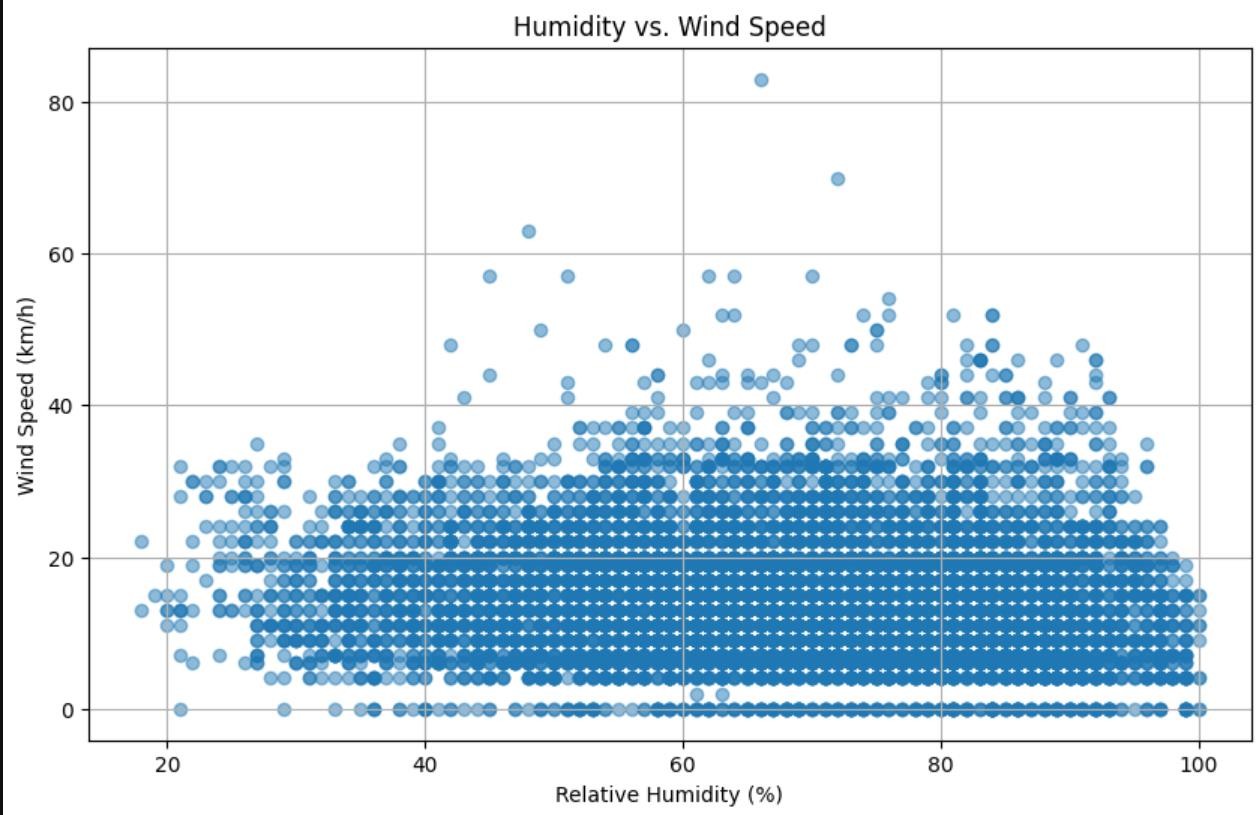
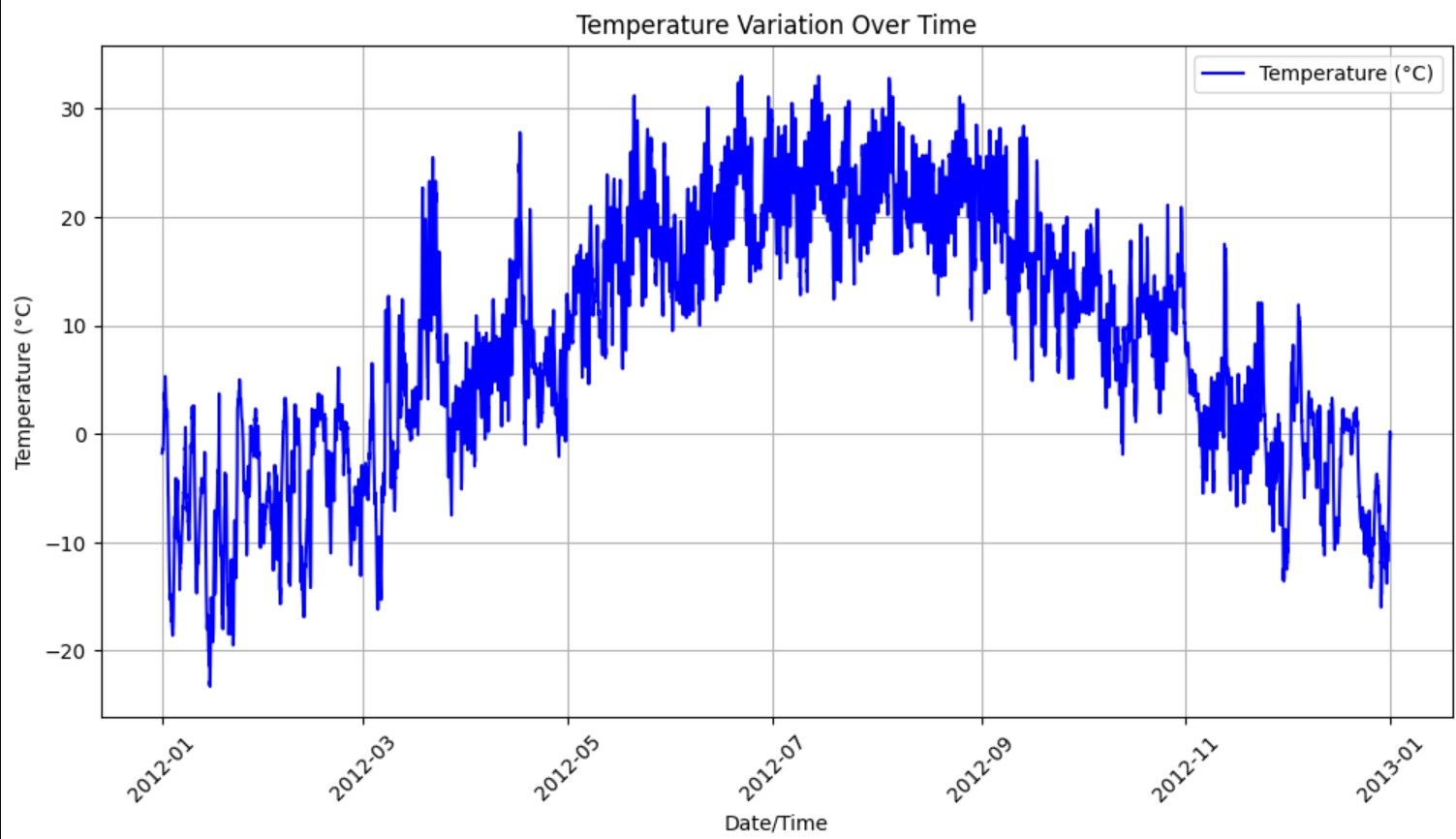
file\_path='/content/1.WeatherData.csv'df=load\_weather\_data(file\_path)

df=prepare\_data(df)

gdf = create\_geodataframe(df)analyze\_weather\_conditions(gdf)visualize\_weather\_conditions(gdf)plot\_temperature\_over\_time(df)plot\_humidity\_vs\_wind\_speed(df)

if name == "main":main()

### Output :





**Result :**

ThuswestudiedGISincontextofDataAnalysisUsingPython.

### Exp No:12 Date :

**StudyofCollaborativeVisualization**

### Aim:

ToStudyCollaborativeVisualization.

### PlotName:HistogramofTemperature

##### Algorithm:

* + Countthefrequencyoftemperaturevalues.
  + Createbinsforthetemperaturerange.
  + Plotthehistogram withtemperatureonthe x-axisand frequencyon they-axis.

### SourceCode:

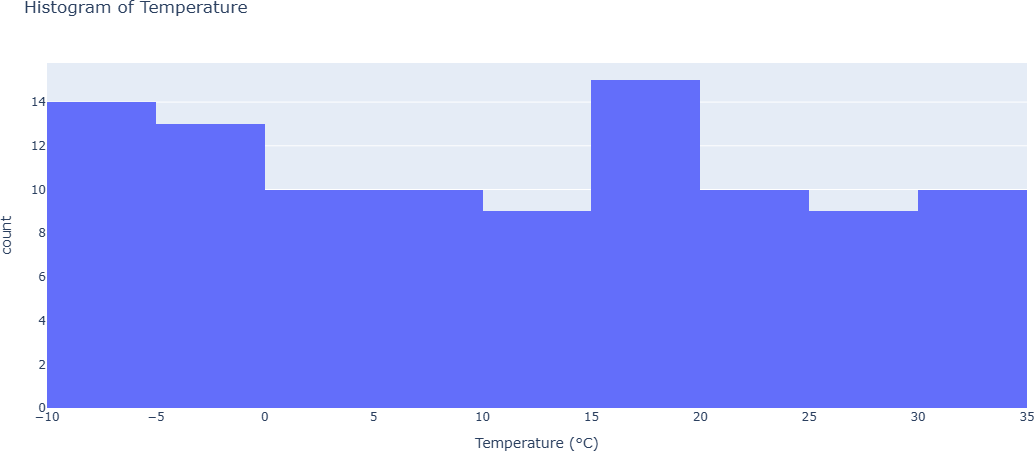
importplotly.expressaspx

defplot\_temperature\_histogram(df):

fig=px.histogram(df,x='Temp\_C',title='HistogramofTemperature',labels={'Temp\_C':'Temperature(°C)'}, nbins=20)

fig.show()

### Output :



1. **PlotName:AreaPlotforTemperatureOverTime**

### Algorithm:

* + Prepareatimeseriesoftemperaturedata.
  + Usethe areaplotto fillthespaceunderthetemperaturecurve.
  + Setthex-axistodateandthey-axisto temperature.

### SourceCode:

defplot\_temperature\_area(df):

fig=px.area(df,x='Date/Time',y='Temp\_C',title='AreaPlotofTemperatureOverTime',labels={'Temp\_C':'Temperature(°C)','Date/Time':'Date'})

fig.show()

### Output :

1. **Plot Name: Line Plot for Wind Speed Over TimeAlgorithm:**
   * Extractwindspeeddata overtime.
   * Createalineplotwithdateson thex-axisand wind speedon the y-axis.

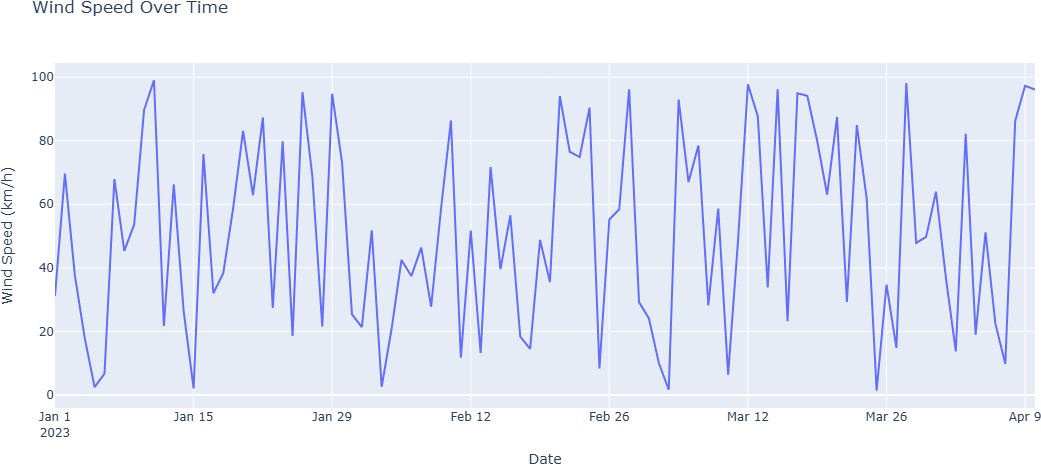
### SourceCode:

defplot\_wind\_speed\_over\_time(df):

fig=px.line(df,x='Date/Time',y='Wind\_Speed\_km\_h',title='WindSpeedOverTime',labels={'Wind\_Speed\_km\_h':'WindSpeed(km/h)','Date/Time':'Date'})

fig.show()

### Output :



1. **Plot Name: Sunburst Chart for Weather BreakdownAlgorithm:**
   * Countoccurrencesof eachweathercondition.
   * Createasunburstchart tovisuallyrepresentthebreakdownofweatherconditions.

### SourceCode:

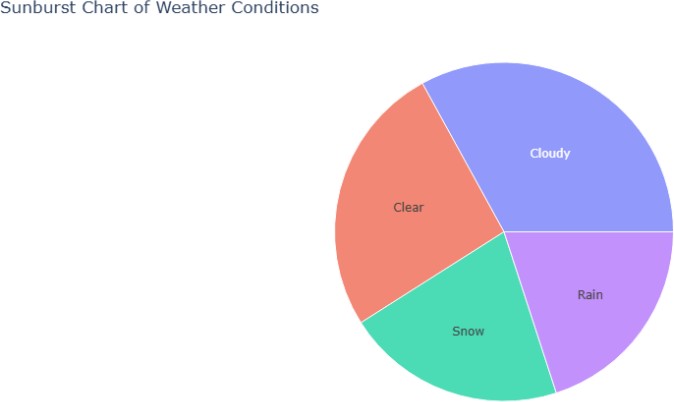
defplot\_weather\_sunburst(df):

weather\_counts=df['Weather'].value\_counts().reset\_index()weather\_counts.columns=['Weather','Count']

fig=px.sunburst(weather\_counts,path=['Weather'],values='Count',title='SunburstChartofWeatherConditions')

fig.show()

### Output :



1. **Plot Name: Scatter Plot MatrixAlgorithm:**
   * Selectnumerical columnsforanalysis.
   * Createscatterplotsforeach pairofselected columns.
   * Usedifferentcolorsfordifferent weatherconditions.

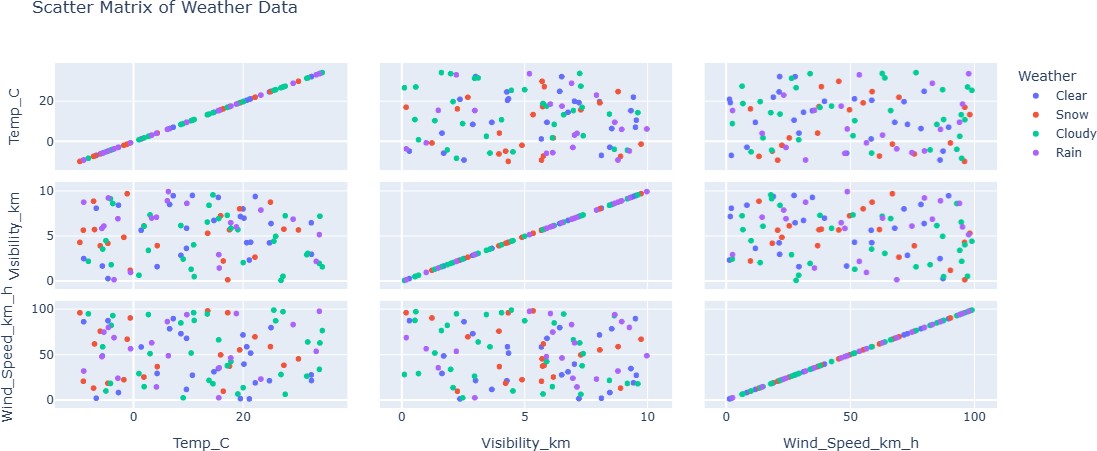
### SourceCode:

defplot\_scatter\_matrix(df):

fig=px.scatter\_matrix(df,dimensions=['Temp\_C','Visibility\_km','Wind\_Speed\_km\_h'],color='Weather',title='ScatterMatrixofWeather Data')

fig.show()

### Output :



**Result :**

#### ThusweStudiedaboutCollaborativeAnalysis