Installation of Python

Ex. No.:1

Date:

Aim:

To install the data analysis and visualization tool Python.

Algorithm:

Step1: Download the spyder open source software. It is an integrated development environment written in python programming

Step2: Install the software by following the instruction.

Step3: Open the new spyder window and type the given python code

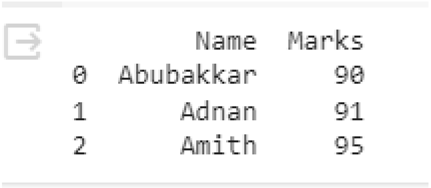
Step4: Save the program

Step5: Execute the program

Program:

```
import pandas as pd

student = {
  "Name": ['Abubakkar', 'Adnan', 'Amith'],
  "Marks": [90, 91, 95]
}
#load data into a DataFrame object:
df = pd.DataFrame(student)
print(df)
```

Output:



```
        Name  Marks
0  Abubakkar     90
1      Adnan     91
2      Amith     95
```

Result:

Thus spyder has been installed and the program has been executed successfully.

Exploratory Data Analysis on email dataset

Ex. No.:2

Date:

Aim:

To perform exploratory data analysis (EDA) on with datasets like email data set. Export all your emails as a dataset, import them inside a pandas data frame, visualize them and get different insights from the data using Python.

Algorithm:

Step1: Start the program

Step2: download the email dataset from kaggle

Step3: import the dataset downloaded

Step4: import pandas for visualize the data

Step5: Execute the program

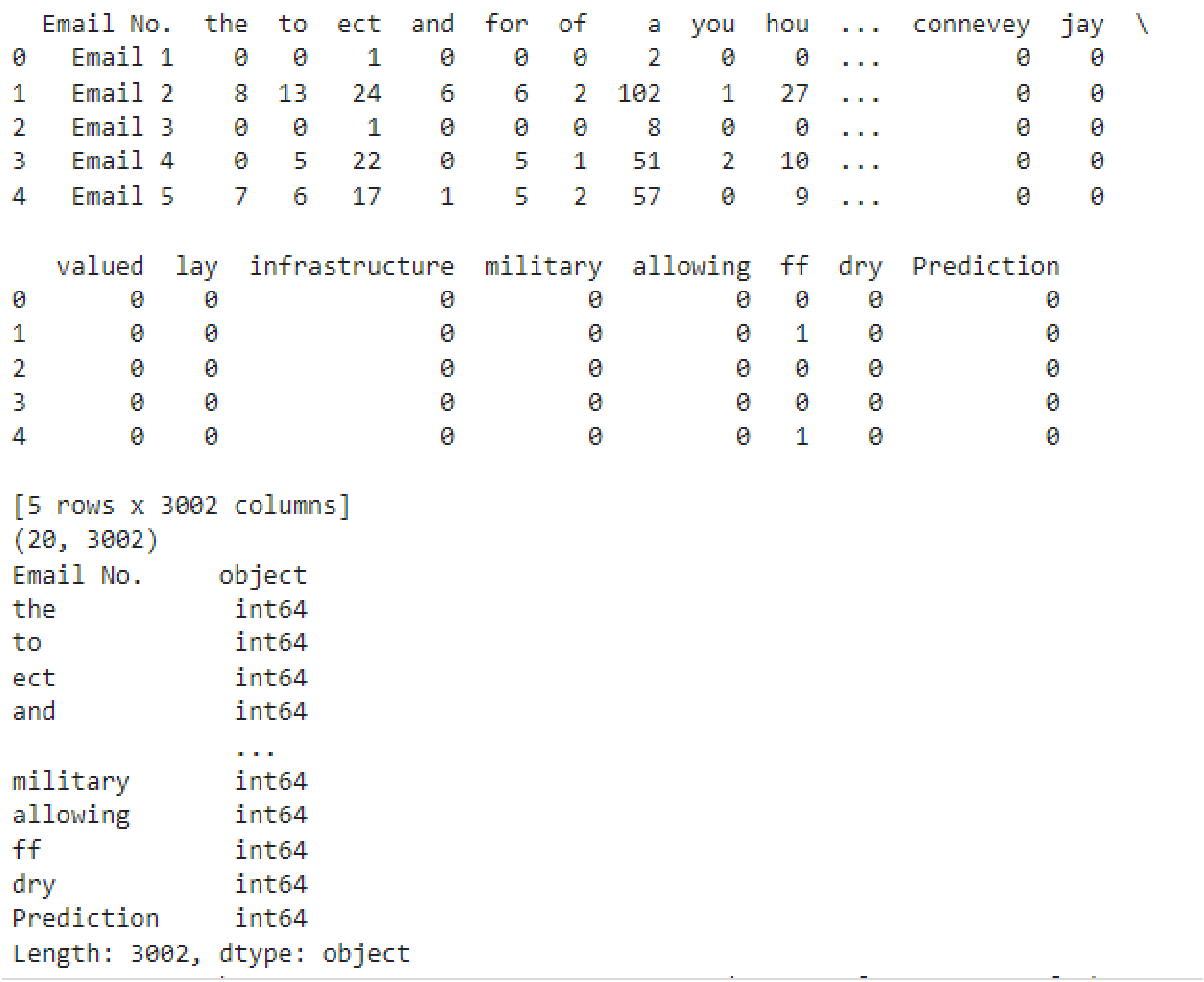
Step6: Display the output

Step7: Stop the program

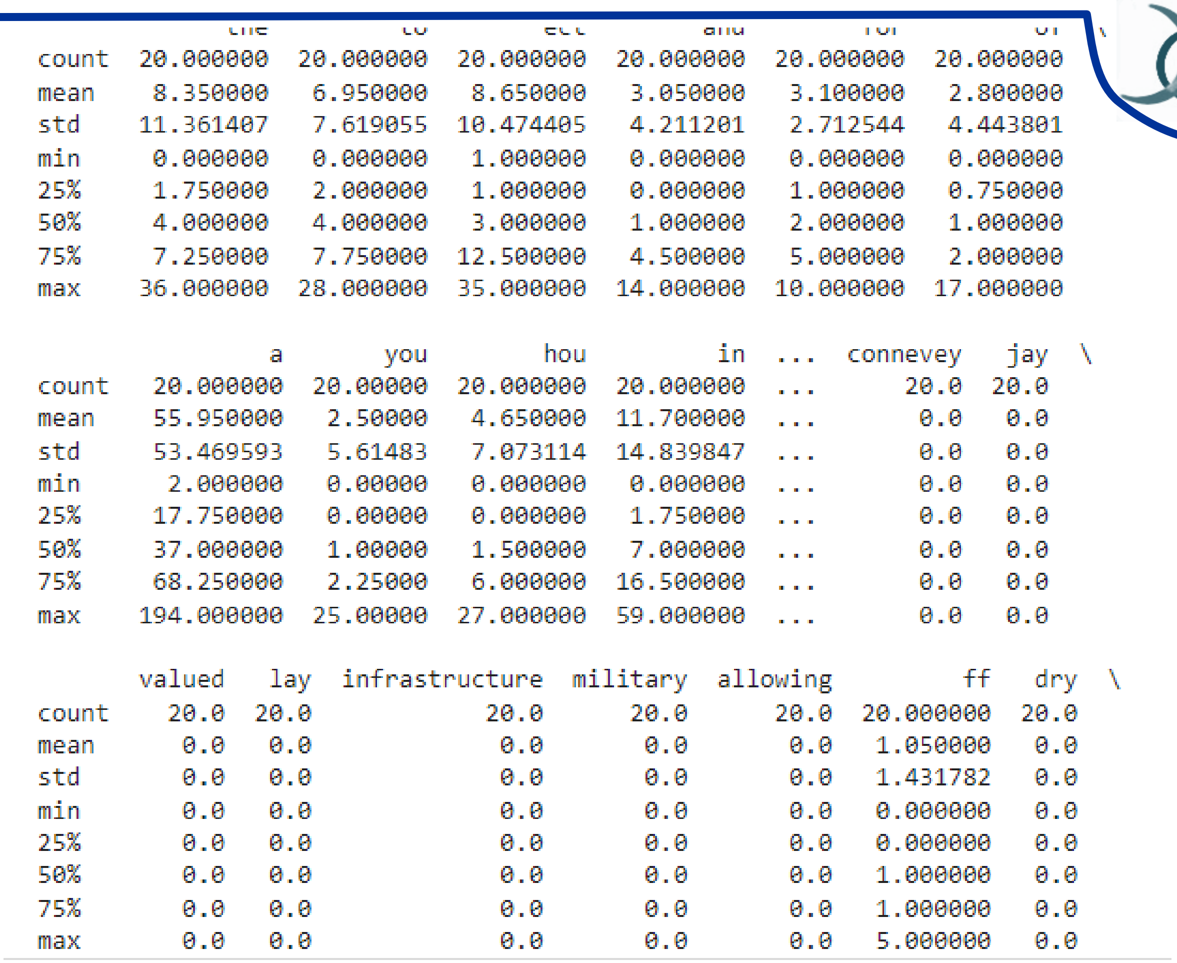
Program:

```
import pandas as pd
import matplotlib.pyplot as plt
emails = pd.read_csv("email.csv")
print(emails.head())
print(emails.shape)
print(emails.dtypes)
print(emails.describe())
print(emails.isnull().sum())
emails['length'].plot(kind='hist', bins=50)
plt.xlabel('Email Length')
plt.show()
emails['sender'].value_counts().plot(kind='bar')
plt.xlabel('Email Sender')
plt.show()
```

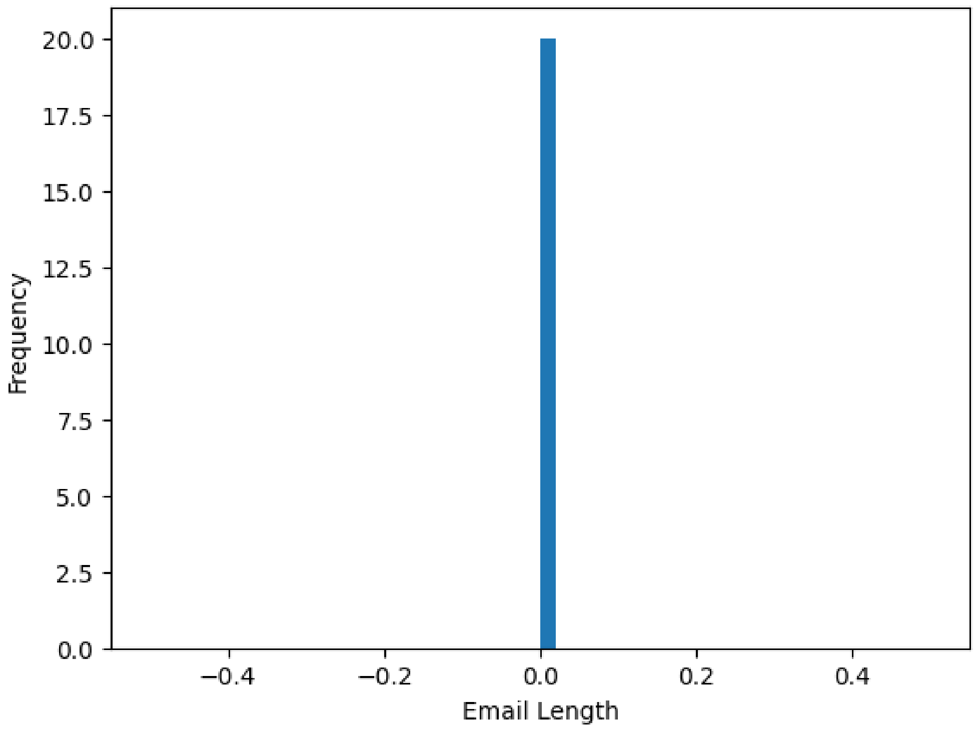
Output:

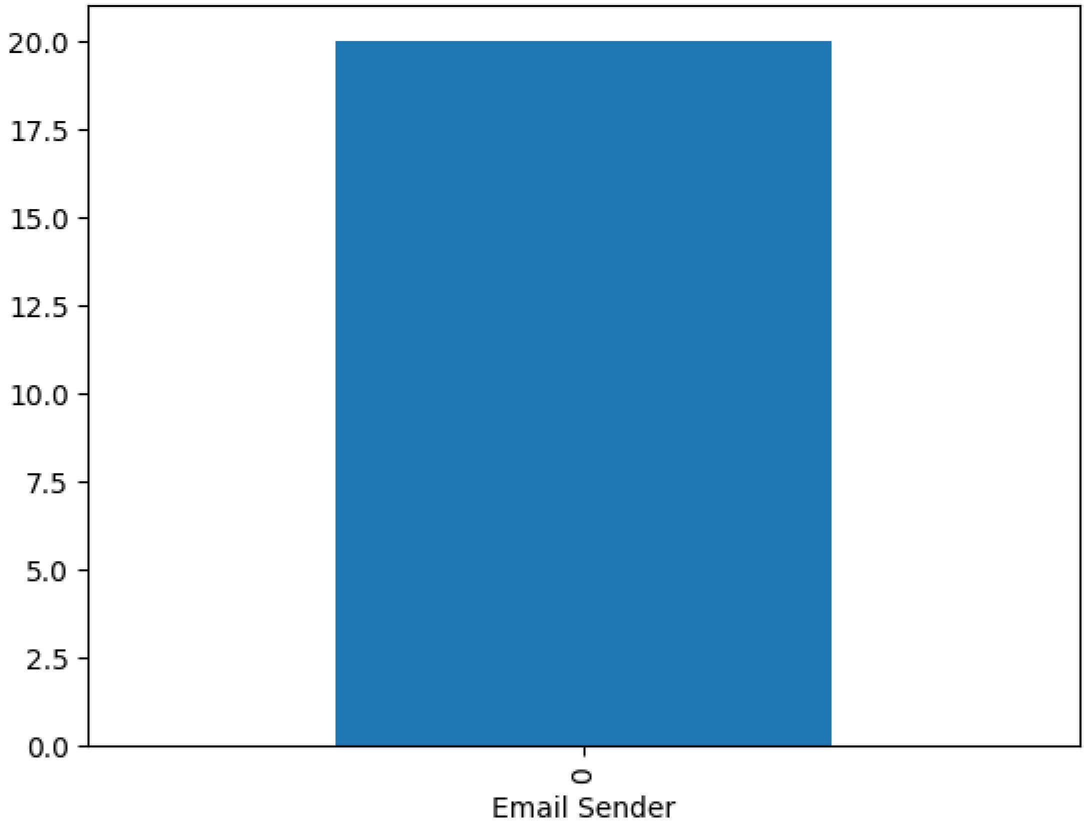


```
  Email No.  the  to  ect  and  for  of    a  you  hou  ...  connevey  jay  \
0   Email 1    0   0    1    0    0   0    2    0    0  ...         0    0
1   Email 2    8  13   24    6    6   2  102    1   27  ...         0    0
2   Email 3    0   0    1    0    0   0    8    0    0  ...         0    0
3   Email 4    0   5   22    0    5   1   51    2   10  ...         0    0
4   Email 5    7   6   17    1    5   2   57    0    9  ...         0    0

   valued  lay  infrastructure  military  allowing  ff  dry  Prediction
0       0    0               0         0         0   0    0           0
1       0    0               0         0         0   1    0           0
2       0    0               0         0         0   0    0           0
3       0    0               0         0         0   0    0           0
4       0    0               0         0         0   1    0           0

[5 rows x 3002 columns]
(20, 3002)
Email No.     object
the            int64
to             int64
ect            int64
and            int64
               ...
military       int64
allowing       int64
ff             int64
dry            int64
Prediction     int64
Length: 3002, dtype: object
```



```
             the         to        ect        and        for         of  \
count  20.000000  20.000000  20.000000  20.000000  20.000000  20.000000
mean    8.350000   6.950000   8.650000   3.050000   3.100000   2.800000
std    11.361407   7.619055  10.474405   4.211201   2.712544   4.443801
min     0.000000   0.000000   1.000000   0.000000   0.000000   0.000000
25%     1.750000   2.000000   1.000000   0.000000   1.000000   0.750000
50%     4.000000   4.000000   3.000000   1.000000   2.000000   1.000000
75%     7.250000   7.750000  12.500000   4.500000   5.000000   2.000000
max    36.000000  28.000000  35.000000  14.000000  10.000000  17.000000

                a       you        hou         in  ...  connevey   jay  \
count   20.000000  20.00000  20.000000  20.000000  ...      20.0  20.0
mean    55.950000   2.50000   4.650000  11.700000  ...       0.0   0.0
std     53.469593   5.61483   7.073114  14.839847  ...       0.0   0.0
min      2.000000   0.00000   0.000000   0.000000  ...       0.0   0.0
25%     17.750000   0.00000   0.000000   1.750000  ...       0.0   0.0
50%     37.000000   1.00000   1.500000   7.000000  ...       0.0   0.0
75%     68.250000   2.25000   6.000000  16.500000  ...       0.0   0.0
max    194.000000  25.00000  27.000000  59.000000  ...       0.0   0.0

       valued   lay  infrastructure  military  allowing         ff   dry  \
count    20.0  20.0            20.0      20.0      20.0  20.000000  20.0
mean      0.0   0.0             0.0       0.0       0.0   1.050000   0.0
std       0.0   0.0             0.0       0.0       0.0   1.431782   0.0
min       0.0   0.0             0.0       0.0       0.0   0.000000   0.0
25%       0.0   0.0             0.0       0.0       0.0   0.000000   0.0
50%       0.0   0.0             0.0       0.0       0.0   1.000000   0.0
75%       0.0   0.0             0.0       0.0       0.0   1.000000   0.0
max       0.0   0.0             0.0       0.0       0.0   5.000000   0.0
```





Result:

Thus the program has been executed successfully and the output has been verified.

Visualization of data using Matplotlib

Ex. No.:3

Date:

Aim:

To write a python program to work with numpy, pandas and visualize the data using the matplotlib.

Algorithm:

Step1: Start the program

Step2: import numpy package

Step3: import pandas package

Step4: import matplotlib

Step5: Execute the program

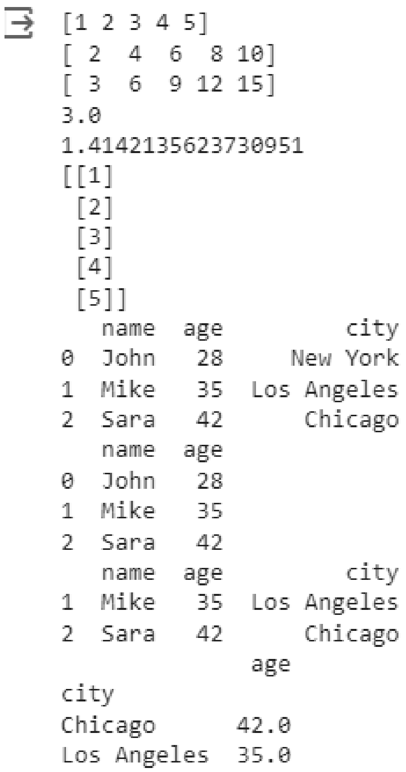
Step6: Display the output

Step7: Stop the program

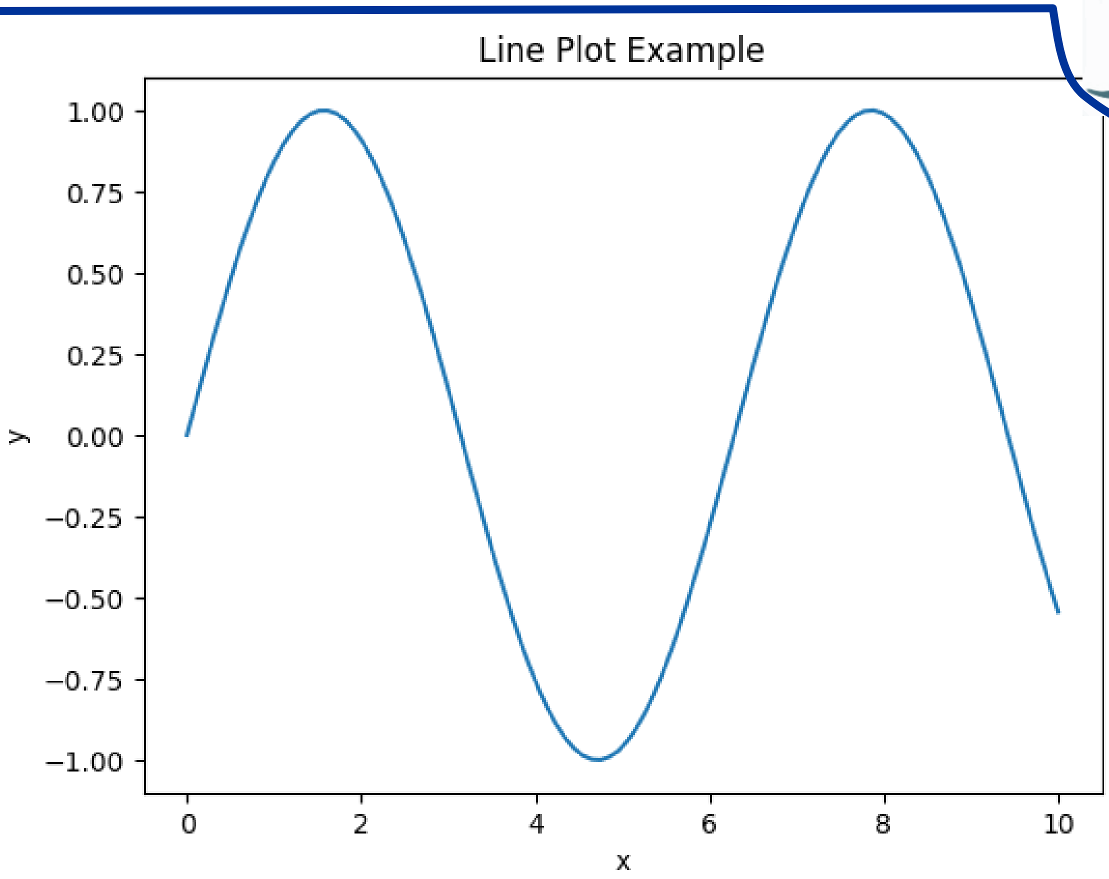
Program:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
a = np.array([1, 2, 3, 4, 5])
print(a)
b = a * 2
print(b)
c = a + b
print(c)
mean = np.mean(a)
print(mean)
std_dev = np.std(a)
print(std_dev)
a = a.reshape(5, 1)
print(a)
data = {'name': ['John', 'Mike', 'Sara'],
 'age': [28, 35, 42],
 'city': ['New York', 'Los Angeles', 'Chicago']}
df = pd.DataFrame(data)
print(df)
print(df[['name', 'age']])
print(df[df['age'] > 30])
print(df.groupby(['city']).mean())
df['income'] = [50000, 60000, 70000]
print(df)
x = np.linspace(0, 10, 100)
y = np.sin(x)
plt.plot(x, y)
plt.xlabel('x')
plt.ylabel('y')
plt.title('Line Plot Example')
plt.show()
```

Output:



```
[1 2 3 4 5]
[ 2  4  6  8 10]
[ 3  6  9 12 15]
3.0
1.4142135623730951
[[1]
 [2]
 [3]
 [4]
 [5]]
   name  age         city
0  John   28     New York
1  Mike   35  Los Angeles
2  Sara   42      Chicago
   name  age
0  John   28
1  Mike   35
2  Sara   42
   name  age         city
1  Mike   35  Los Angeles
2  Sara   42      Chicago
              age
city
Chicago      42.0
Los Angeles  35.0
```



Result:

Thus the program has been completed successfully and the output has been verified.

Explore the data using R

Ex. No.:4

Date:

Aim:

To explore various variable and row filters in R for cleaning data. Apply various plot features in R on sample data sets and visualize.

Algorithm:

Step1: Start the program

Step2: Download Rstudio tool

Step3: Install the tool

Step4: install the package ggplot2

Step5: Execute the program

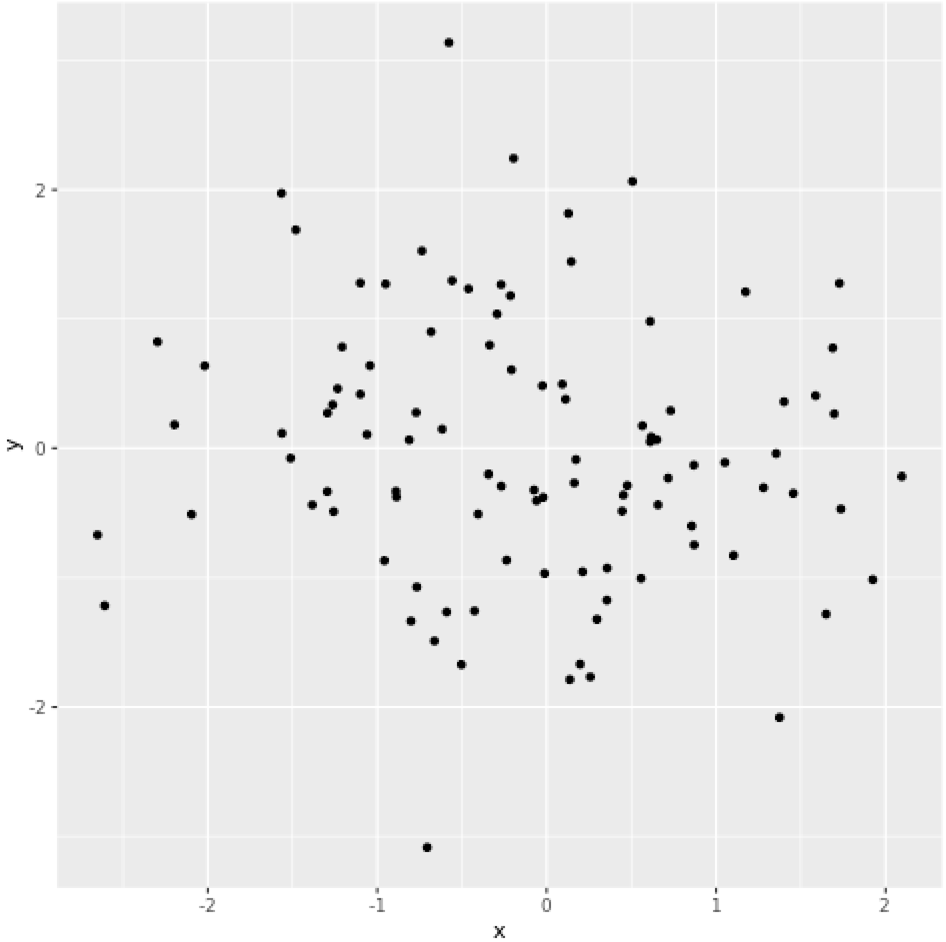
Step6: Display the output

Step7: Stop the program

Program:

```
library(ggplot2)
# Create a sample data frame
df<- data.frame(x = rnorm(100), y = rnorm(100))
# Create a scatter plot
ggplot(data = df, aes(x = x, y = y)) +
geom_point()
# Create a line plot
plot(df$x, type = "l")
```

Output:



Result:

Thus the program has been successfully completed and the output has been verified.

Visualization of Time Series Data Analysis

Ex. No.:5

Date:

Aim:

To perform Time Series Analysis and apply the various visualization techniques using R.

Algorithm:

Step1: Start the program

Step2: Download Rstudio tool

Step3: Install the tool

Step4: install the package ggplot2

Step5: Execute the program

Step6: Display the output

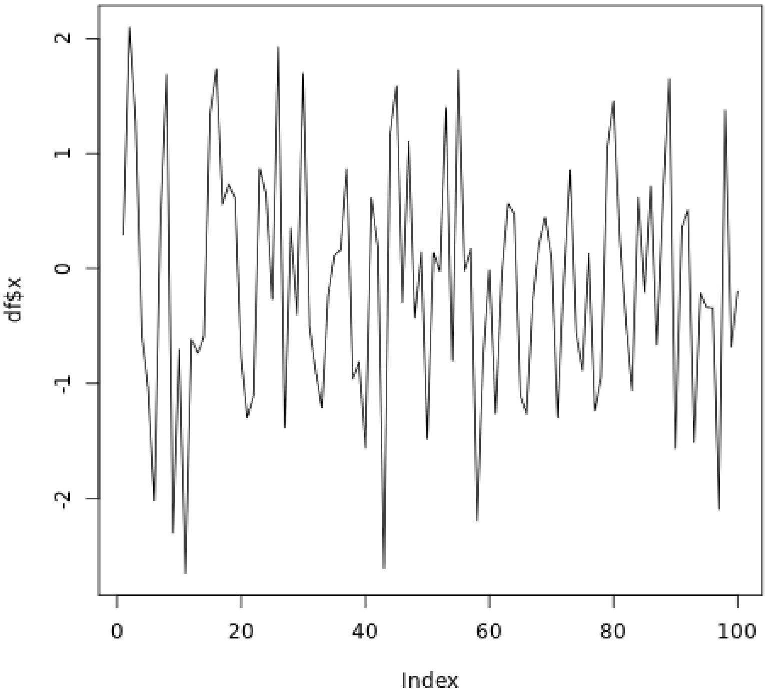
Step7: Stop the program

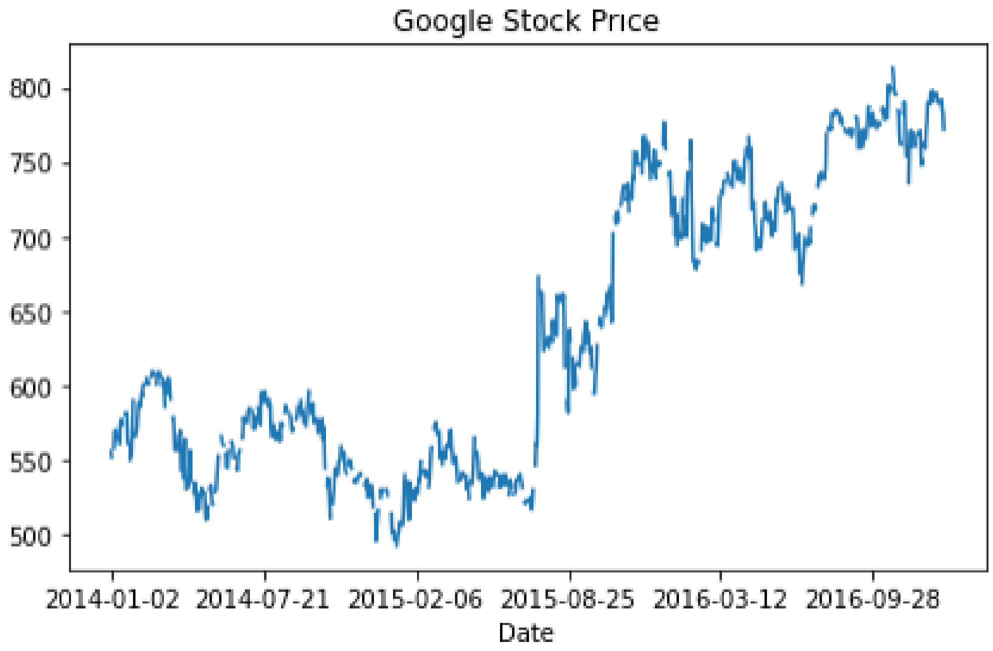
Program:

```
install.packages(forecast)
library(forecast)
# Create a sample time series
ts<- ts(rnorm(100), start = c(2010, 1), frequency = 12)
# Decompose the time series into its trend, seasonal, and residual components
decomposed_ts<- decompose(ts)
# Plot the decomposition
plot(decomposed_ts)
# Fit an exponential smoothing model to the time series
fit <- ets(ts)
# Forecast the next 10 periods
forecast(fit, h = 10)

#Here is an example of how to create a line plot of a time series using the ggplot2
package:
```

```
install.packages(ggplot2)

library(ggplot2)
# Create a line plot of the time series
ggplot(data = ts, aes(x = time(ts), y = ts)) + geom_line()
install.packages(ggplots)
install.packages(reshape2)
library(ggplot2)
library(reshape2)
# Melt the data
data_melt<- melt(ts, id = "time")
# Plot the heatmap
ggplot(data = data_melt, aes(x = time, y = variable, fill = value)) + geom_tile() +
                                                                  scale_fill_gradient()
```

Output:





Result:

Thus the program has been successfully executed and the output has been verified.

Data Analysis and representation on a Map

Ex. No.:6

Date:

Aim:

To perform Data Analysis and representation on a Map using various Map data sets with World Map with Pandas

Algorithm:

Step1: Start the program

Step2: import matplotlib package

Step3: import numpy package

Step4: import folium package

Step5: Execute the program

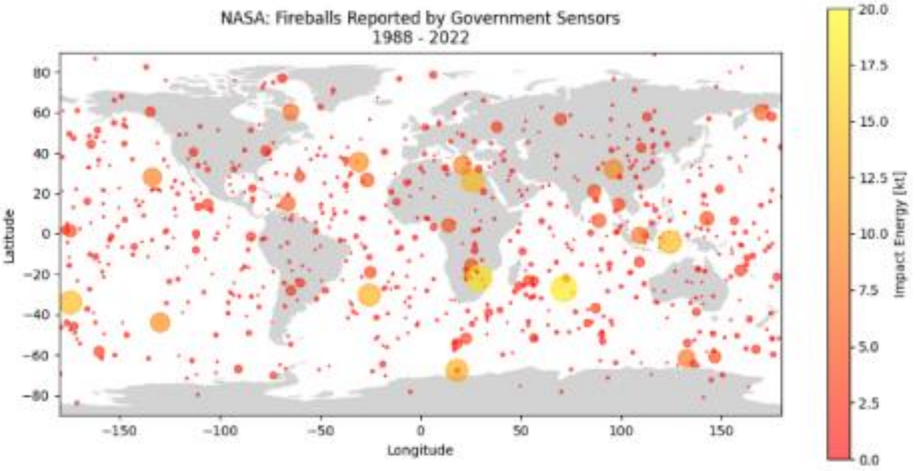
Step6: Display the output

Step7: Stop the program

Program:

```
import matplotlib.pyplot as plt

import numpy as np

import folium

# Create a map centered on a specific location

location = [40.693943, -73.985880]

map = folium.Map(location=location, zoom_start=13)

# Add data points to the map

for i in range(0, len(data)):

folium.Marker(data.loc[i, 'coordinates'], popup=data.loc[i, 'name']).add_to(map)
```

Output:



Result:

Thus the program has been successfully completed and the output has been verified.

Visualization of data for multiple datasets

Ex. No.:7

Date:

Aim:

To build cartographic visualization for multiple datasets involving various countries of the world; states and districts in India.

Algorithm:

Step1: Start the program

Step2: import matplotlib package

Step3: import numpy package

Step4: import folium package

Step5: Execute the program

Step6: Display the output

Step7: Stop the program

Program:

```
#Importing Libraries

import numpy as np

import pandas as pd

import shapefile as shp

import matplotlib.pyplot as plt

import seaborn as sns

#Initializing Visualization Set

sns.set(style=”whitegrid”, palette=”pastel”, color_codes=True) sns.mpl.rc(“figure”,
figsize=(10,6))

#Opening The Vector Map- A vector map is a group of several files with a .shp format.

shp_path = \\District_Boundary.shp
```

```
#reading the shape file by using reader function of the shape lib

sf = shp.Reader(shp_path)

#Number of different shapes which were imported by shp.reader

len(sf.shapes())

#The result will come out to be 33 which tells us that there are 33 shapes or we can say
cities in the region of Rajasthan.

#To explore those records:

sf.records()
```

Making accessing cities easier by converting shapefile data into a more relatable Pandas Dataframe format.

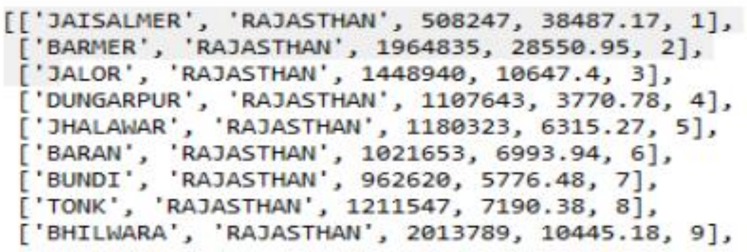
```
def read_shapefile(sf):

 #fetching the headings from the shape file

 fields = [x[0] for x in sf.fields][1:]

#fetching the records from the shape file

 records = [list(i) for i in sf.records()]

 shps = [s.points for s in sf.shapes()]

#converting shapefile data into pandas dataframe

 df = pd.DataFrame(columns=fields, data=records)

#assigning the coordinates

 df = df.assign(coords=shps)

 return df

#Visualization of data after being converted into Dataframes where it refers to rows and
columns

df = read_shapefile(sf)

df.shape()

df.sample(5)
```

```
def plot_shape(id, s=None):
 plt.figure()
 #plotting the graphical axes where map ploting will be done
 ax = plt.axes()
 ax.set_aspect('equal')
#storing the id number to be worked upon
 shape_ex = sf.shape(id)
#NP.ZERO initializes an array of rows and column with 0 in place of each elements
 #an array will be generated where number of rows will be(len(shape_ex,point))and
number of columns will be 1 and stored into the variable
 x_lon = np.zeros((len(shape_ex.points),1))
#an array will be generated where number of rows will be(len(shape_ex,point))and
number of columns will be 1
and stored into the variable
 y_lat = np.zeros((len(shape_ex.points),1))
 for ip in range(len(shape_ex.points)):
 x_lon[ip] = shape_ex.points[ip][0]
 y_lat[ip] = shape_ex.points[ip][1]
#plotting using the derived coordinated stored in array created by numpy
 plt.plot(x_lon,y_lat)
 x0 = np.mean(x_lon)
 y0 = np.mean(y_lat)
 plt.text(x0, y0, s, fontsize=10)
# use bbox (bounding box) to set plot limits
 plt.xlim(shape_ex.bbox[0],shape_ex.bbox[2])
 return x0, y0
```

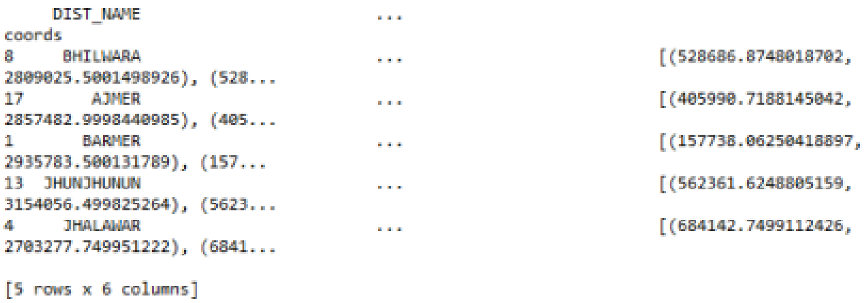
Setting The City Name To Plot Respective Map

```
DIST_NAME = ‘JAIPUR’

#to get the id of the city map to be plotted

com_id = df[df.DIST_NAME == ‘JAIPUR’].index.get_values()[0]

plot_shape(com_id, DIST_NAME)

sf.shape(com_id)

def plot_map(sf, x_lim = None, y_lim = None, figsize = (11,9)):

 plt.figure(figsize = figsize)

 id=0

 for shape in sf.shapeRecords():

 x = [i[0] for i in shape.shape.points[:]]

 y = [i[1] for i in shape.shape.points[:]]

 plt.plot(x, y, 'k')

 if (x_lim == None) & (y_lim == None):

 x0 = np.mean(x)

 y0 = np.mean(y)

 plt.text(x0, y0, id, fontsize=10)

 id = id+1

 if (x_lim != None) & (y_lim != None):

 plt.xlim(x_lim)

 plt.ylim(y_lim)

#calling the function and passing required parameters to plot the full map

plot_map(sf)
```

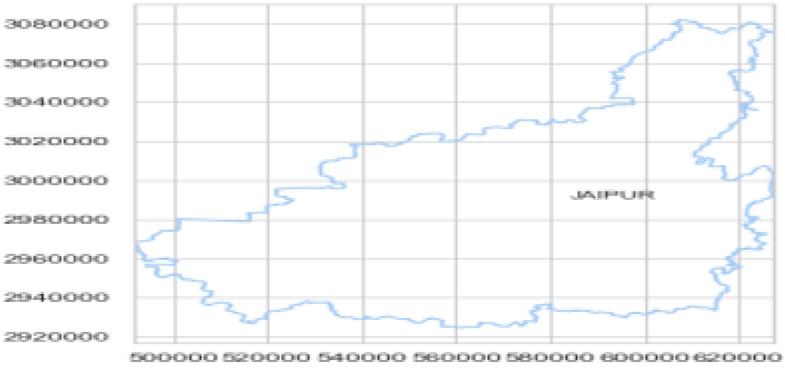
Output:

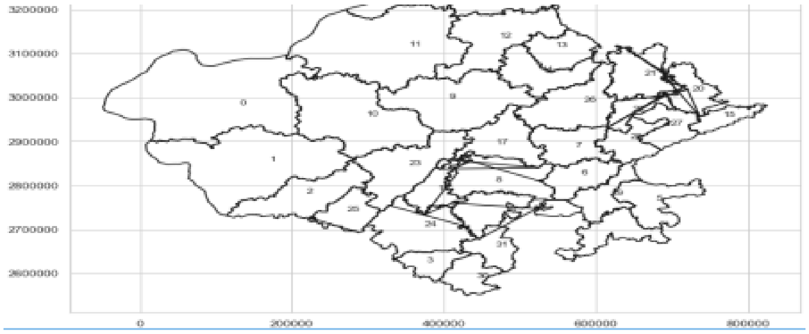


```
[['JAISALMER', 'RAJASTHAN', 508247, 38487.17, 1],
 ['BARMER', 'RAJASTHAN', 1964835, 28550.95, 2],
 ['JALOR', 'RAJASTHAN', 1448940, 10647.4, 3],
 ['DUNGARPUR', 'RAJASTHAN', 1107643, 3770.78, 4],
 ['JHALAWAR', 'RAJASTHAN', 1180323, 6315.27, 5],
 ['BARAN', 'RAJASTHAN', 1021653, 6993.94, 6],
 ['BUNDI', 'RAJASTHAN', 962620, 5776.48, 7],
 ['TONK', 'RAJASTHAN', 1211547, 7190.38, 8],
 ['BHILWARA', 'RAJASTHAN', 2013789, 10445.18, 9],
```



```
     DIST_NAME                         ...
coords
8      BHILWARA                        ...                        [(528686.8748018702,
2809025.5001498926), (528...
17        AJMER                        ...                        [(405990.7188145042,
2857482.9998440985), (405...
1        BARMER                        ...                        [(157738.06250418897,
2935783.500131789), (157...
13  JHUNJHUNUN                         ...                        [(562361.6248805159,
3154056.499825264), (5623...
4      JHALAWAR                        ...                        [(684142.7499112426,
2703277.749951222), (6841...

[5 rows x 6 columns]
```





Result:

Thus the program has been completed successfully and the result has been verified.

Perform EDA on Wine Quality datasets

Ex. No.:8

Date:

Aim:

To perform EDA on Wine Quality Data Set.

Algorithm:

Step1: Start the program

Step2: download wine quality dataset

Step3: import matplotlib package

Step4: import seaborn package

Step5: import pandas package

Step6: Execute the program

Step7: Display the output

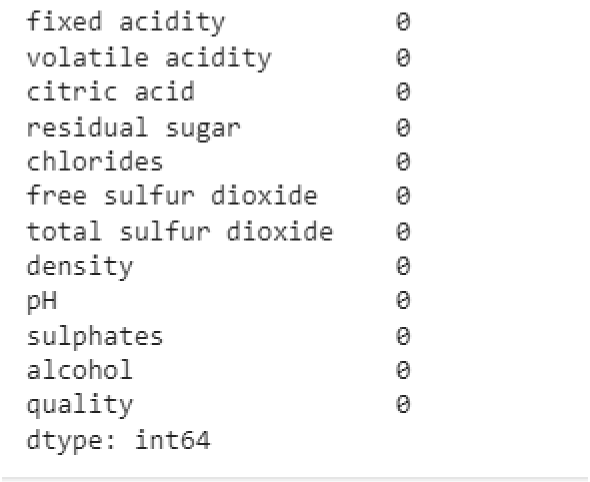
Step8: Stop the program

Program:

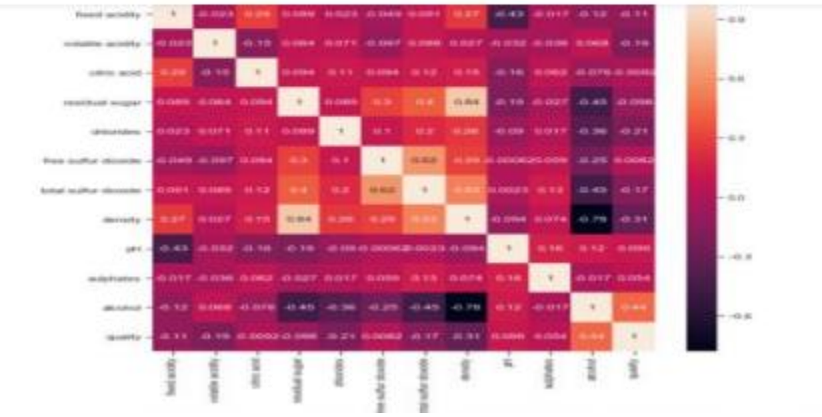
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
# Load the Wine Quality Data Set
data = pd.read_csv('wine.csv')
# Display the first few rows of the data
data.head()
# Display basic statistics about the data
data.describe()
# Check for missing values
data.isnull().sum()
plt.figure(figsize=(10,10))
sns.heatmap(df.corr(),color = ”k”, annot = True)
plt.figure(figsize=(10,15))
for i, col in enumerate(list(df.columns.values)):
        plt.subplot(4,3,i+1)
```

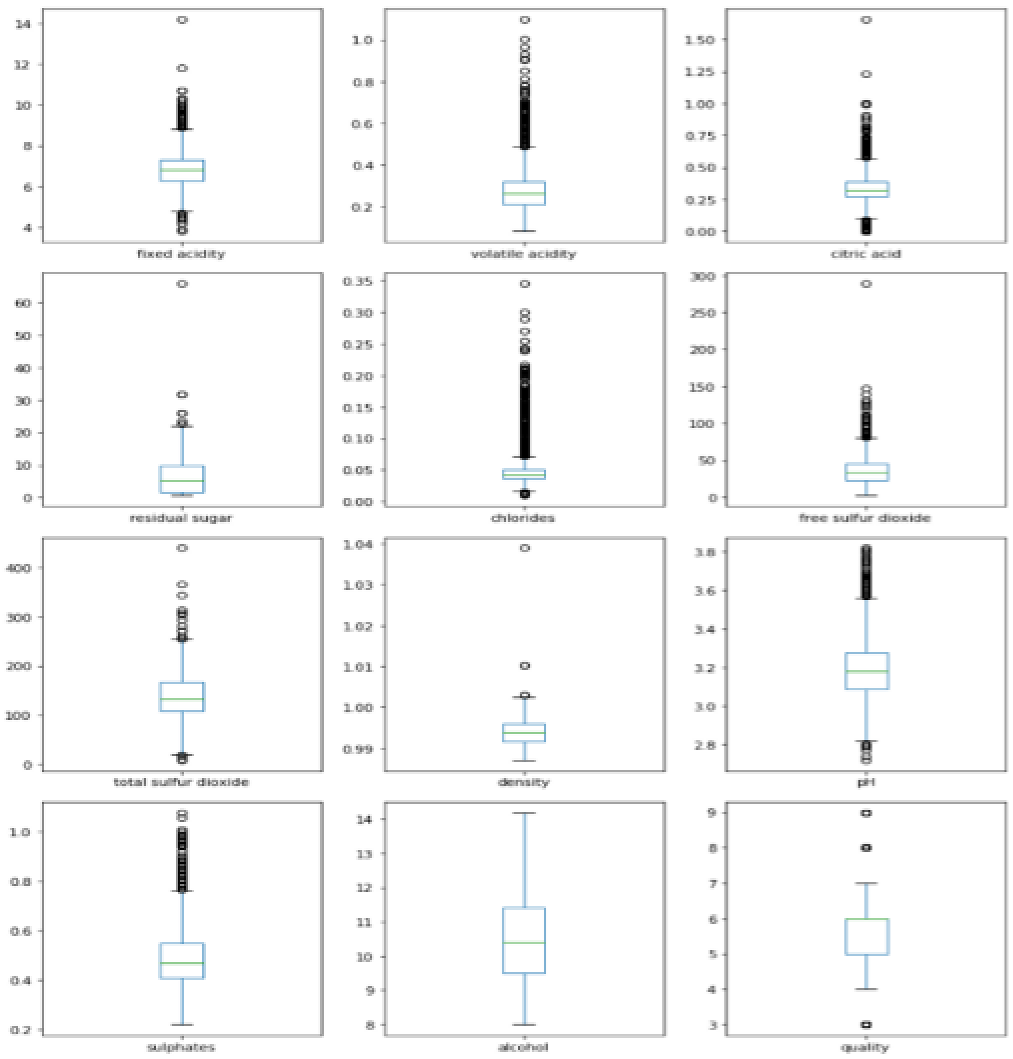
```
df.boxplot(col)
plt.grid()
plt.tight_layout()
```

Output:



```
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
dtype: int64
```





Result:

Thus the program has been successfully completed and the output has been verified.

Case study on retail dataset

Ex. No.:9

Date:

Aim:

To Use a case study on a data set and apply the various EDA and visualization techniques and present an analysis report.

Algorithm:

Step1: Start the program

Step2: download retail dataset

Step3: import matplotlib package

Step4: import seaborn package

Step5: import pandas package

Step6: Analyze the dataset

Step5: Execute the program

Step6: Display the output

Step7: Stop the program

Procedure:

Analytics in Retail:

With the retail market getting more and more competitive by the day, there has never been anything more important than the ability for optimizing service business processes when trying to satisfy the expectations of customers. Channelizing and managing data with the aim of working in favor of the customer as well as generating profits is very significant for survival. Ideally, a retailer’s customer data reflects the company’s success in reaching and nurturing its customers. Retailers built reports summarizing customer behavior using metrics such as conversion rate, average order value, recency of purchase and total amount spent in recent transactions. These measurements provided general insight into the behavioral tendencies of customers.

Customer intelligence is the practice of determining and delivering data-driven insi past and predicted future customer behavior. To be effective, customer intelligen combine raw transactional and behavioral data to generate derived measures. In a nutshell, for big retail players all over the world, data analytics is applied more these days at all stages of the retail process – taking track of popular products that are emerging, doing forecasts of sales and future demand via predictive simulation, optimizing placements of products and offers through heat-mapping of customers and many others.

About the Data

A Retail store is required to analyze the day-to-day transactions and keep a track of its customers spread across various locations along with their purchases/returns across various categories.

What can be done with the data?

Create a report and display the calculated metrics, reports and inferences.

Data Schema

This book has three sheets (Customer, Transaction, Product Hierarchy):

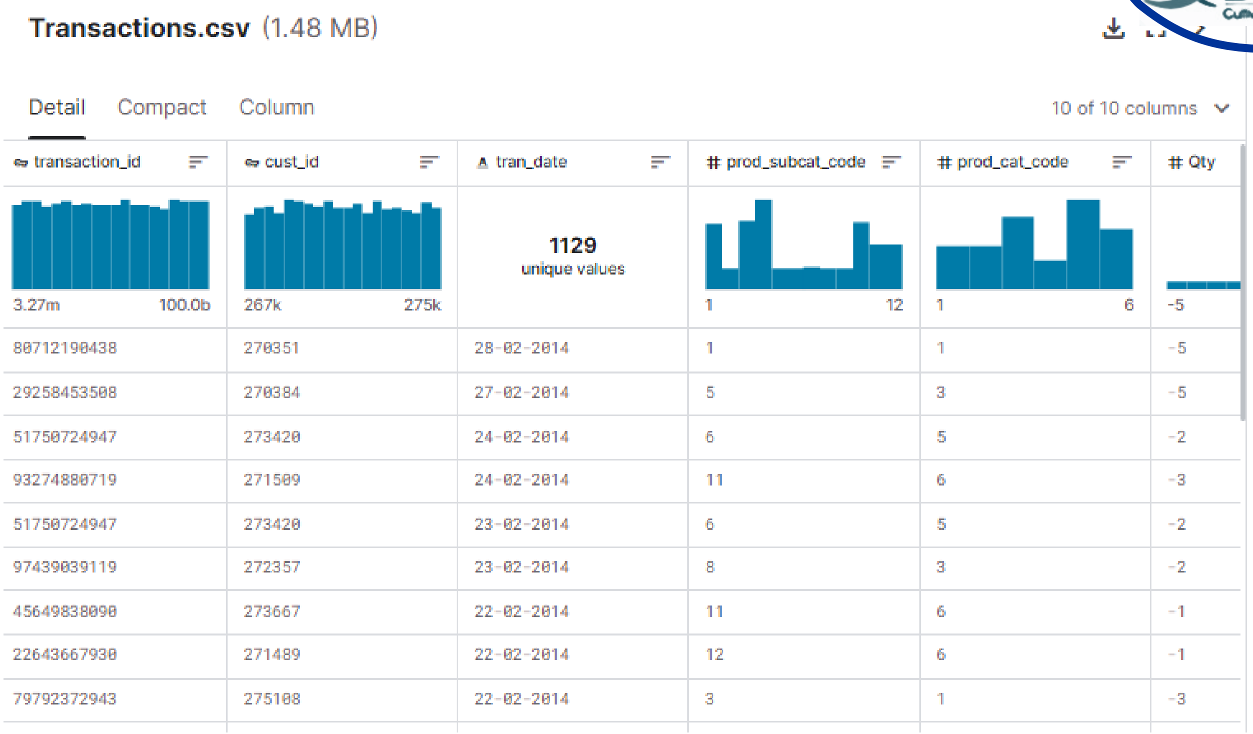
- Customer: Customer information including demographics
- Transaction: Transaction of customers
- Product Hierarchy: Product information

Program:

```
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
import matplotlib.style as style
from datetime import timedelta
import datetime as dt
import time
import os
```

```
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
transactions.insert(loc=3, column='year', value= transactions.tran_date.dt.year)
transactions.insert(loc=4, column='month', value= transactions.tran_date.dt.month)
transactions.insert(loc=5, column='day',
value=(transactions.tran_date.dt.weekday_name))
transactions.head()
orders = rdf.groupby(by=['Store_type'], as_index = False)['Qty'].count()
plt.figure(figsize=(6,4))
sns.set_style('whitegrid')
sns.barplot(x = "Store_type", y = 'Qty', data = orders,  palette= "magma")
plt.xlabel('Store Category')
plt.ylabel('Returned Orders')
plt.title('Total number of returned orders per store category')
plt.show()
category = rdf.groupby(by=['prod_cat'], as_index = False)['Qty'].count()
plt.figure(figsize=(8,4))
sns.set_style('whitegrid')
sns.barplot(x = "prod_cat", y = 'Qty', data = category,  palette= "inferno")
plt.xlabel('Product Category')
plt.ylabel('Returned Orders')
plt.title('Total number of returned orders per product category')
plt.show()
city = rdf.groupby(by= ['city_code'], as_index = False)['Qty'].count()
plt.figure(figsize=(8,4))
sns.set_style('whitegrid')
sns.barplot(x = "city_code", y = 'Qty', data = city,  palette= "viridis")
plt.xlabel('City Code')
plt.ylabel('Returned Orders')
plt.title('Total number of returned orders per city')
plt.show()
```

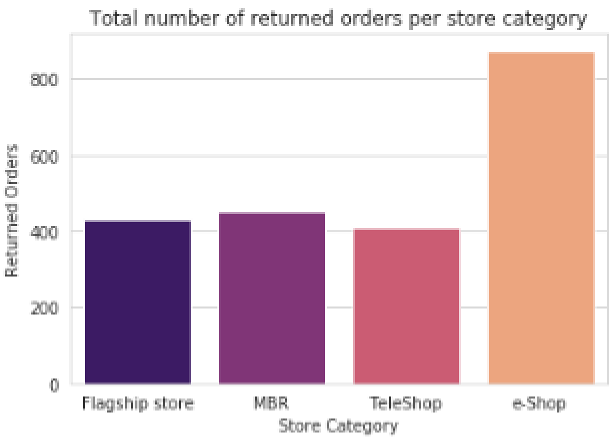
Output:

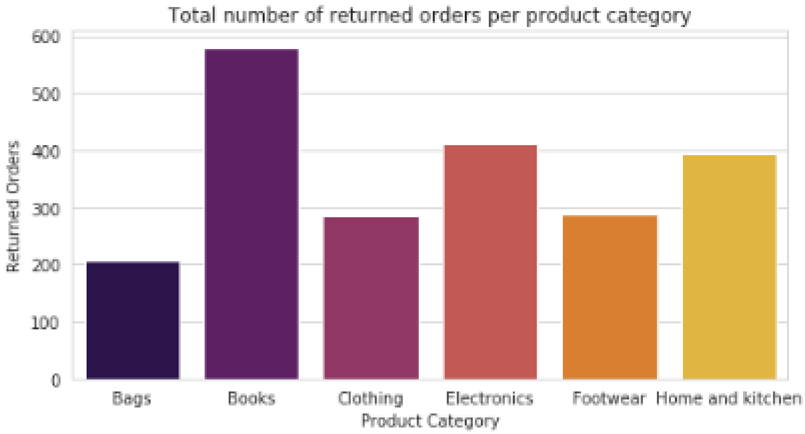


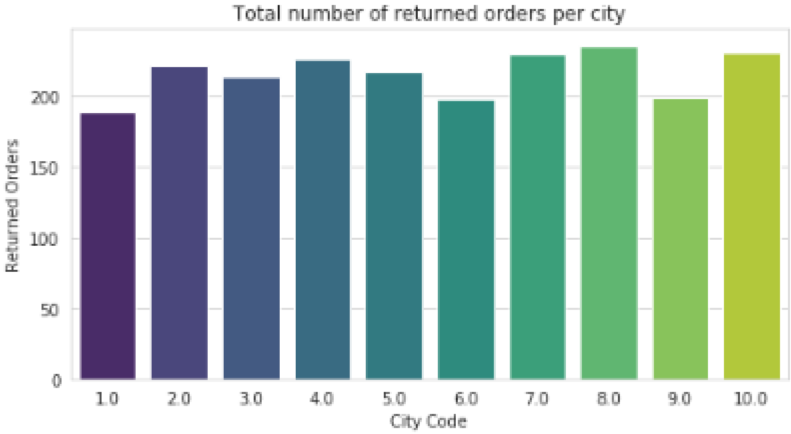
Transactions.csv (1.48 MB)

Detail Compact Column 10 of 10 columns

transaction_id	cust_id	tran_date	prod_subcat_code	prod_cat_code	Qty
3.27m 100.0b	267k 275k	1129 unique values	1 12	1 6	-5
80712190438	270351	28-02-2014	1	1	-5
29258453508	270384	27-02-2014	5	3	-5
51750724947	273420	24-02-2014	6	5	-2
93274880719	271509	24-02-2014	11	6	-3
51750724947	273420	23-02-2014	6	5	-2
97439039119	272357	23-02-2014	8	3	-2
45649838090	273667	22-02-2014	11	6	-1
22643667930	271489	22-02-2014	12	6	-1
79792372943	275108	22-02-2014	3	1	-3







Result:

Thus the case study has been done successfully.