EX.NO: 1	**INSTALL THE DATA ANALYSIS AND VISUALIZATION TOOLS**
DATE:	

AIM:

To Install the data Analysis and Visualization tools in Python.

PROCEDURE:

Step1: Install pandas is to use pip:

```
pip install pandas
```

Step2: Creating A DataFrame in Pandas

```
 assigning two series to s1 and s2
s1 = pd.Series([1,2])
s2 = pd.Series(["Ashish", "Sid"])
# framing series objects into data
df = pd.DataFrame([s1,s2])
# show the data frame
Df
# data framing in another wa
# taking index and column values
dframe = pd.DataFrame([[1,2],["Ashish", "Sid"]],
 index=["r1", "r2"],
columns=["c1", "c2"])
dframe
# framing in another way
# dict-like container
dframe = pd.DataFrame({
 "c1": [1, "Ashish"],
"c2": [2, "Sid"]})
dframe
```

Step 3: Importing Data with Pandas

```
# Import the pandas library, renamed as pd
import pandas as pd
# Read IND_data.csv into a DataFrame, assigned to df
df = pd.read_csv("IND_data.csv")

# Prints the first 5 rows of a DataFrame as default
df.head()
  # Prints no. of rows and columns of a DataFrame
df.shape
```

Step 4: Indexing DataFrames with Pandas

```
# prints first 5 rows and every column which replicates df.head()
df.iloc[0:5,:]
# prints entire rows and columns
df.iloc[:,:]
# prints from 5th rows and first 5 columns
df.iloc[5:,:5]
```

Step 5: Indexing Using Labels in Pandas

```
# prints first five rows including 5th index and every columns of df
df.loc[0:5,:]
# prints from 5th rows onwards and entire columns
df = df.loc[5:,:]
# Prints the first 5 rows of Time period
# value
df.loc[:5,"Time period"]
```

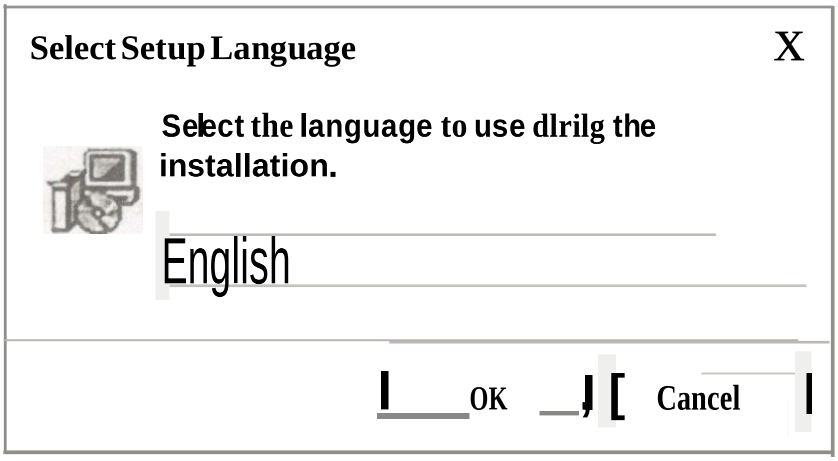
Step 5: DataFrame Math with Pandas

```
# computes various summary statistics, excluding NaN values
df.describe()
# for computing correlations
df.corr()
# computes numerical data ranks
df.rank()
```

Step 6: Pandas Plotting

```
# import the required module
import matplotlib.pyplot as plt
# plot a histogram
df['Observation Value'].hist(bins=10)

# shows presence of a lot of outliers/extreme values
df.boxplot(column='Observation Value', by = 'Time period')

# plotting points as a scatter plot
x = df["Observation Value"]
y = df["Time period"]
plt.scatter(x, y, label= "stars", color= "m",  marker= "*", s=30)
# x-axis label
plt.xlabel('Observation Value')
# frequency label
plt.ylabel('Time period')
# function to show the plot
plt.show()
```

OUTPUT:



Select Setup Language

Select the language to use dlrilg the installation.

English

RESULT:

Thus the procedure to install data analysis and visualization tool was completed successfully.

EX.NO: 2	**PERFORM EXPLORATORY DATA ANALYSIS**
DATE:	

AIM:

To perform exploratory data analysis on with email data set.

PROCEDURE:

STEP1: Importing libraries and loading the file

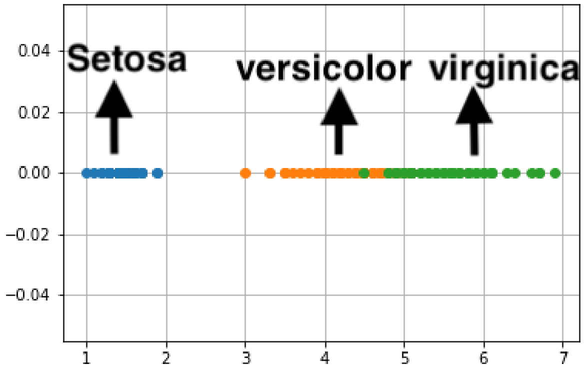
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns#Load Dataset
iris = pd.read_csv(“iris.csv”)
```

STEP 2: Understanding Data

```
print(iris.shape) #prints no. of row and columns
>(150,5)print(iris.columns) #prints name of columns
>Index(['sepal_length', 'sepal_width', 'petal_length',
'petal_width','species'],dtype='object')iris[“species”].value_counts()
>setosa       50
 virginica    50
 versicolor   50
 Name: species, dtype: int64
```

STEP 3: 1D Scatter plot

```
iris_setso = iris.loc[iris["species"] == "setosa"];
iris_virginica = iris.loc[iris["species"] == "virginica"];
iris_versicolor =iris.loc[iris["species"]"versicolor"];
plt.plot(iris_setso["petal_length"],np.zeros_like(iris_setso["petal_length"]),'o')
plt.plot(iris_versicolor["petal_length"],np.zeros_like(iris_versicolor["petal_length"]),'o')
plt.plot(iris_virginica["petal_length"],np.zeros_like(iris_virginica["petal_length"]), 'o')
plt.grid()
plt.show()
```

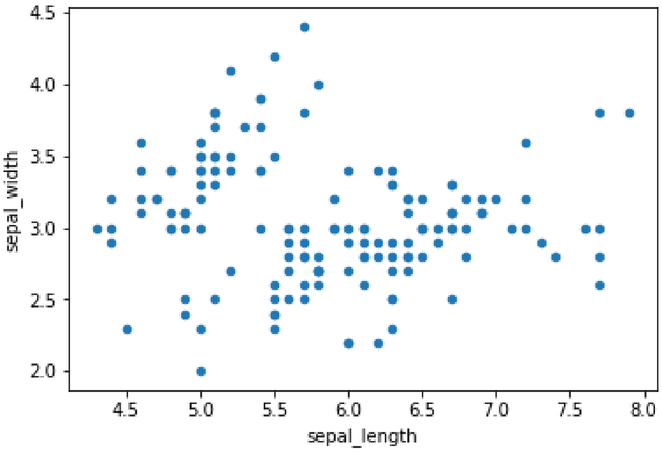


Conclusion

- Green points are Virginica, orange points are Versicolor and blue points are Setosa
- Virginica and Versicolor are overlapping
- 1D Scatter are very hard to read and understand

2D scatter plot

```
iris.plot(kind="scatter",x="sepal_length",y="sepal_width")
plt.show()
```



RESULT:

Thus the program for exploratory data analysis on datasets is verified and the output is verified.

EX.NO: 3	**WORKING WITH NUMPY ARRAYS,PANDAS DATA FRAMES USING MATPLOTLIB**
DATE:	

AIM:

To perform numpy arrays and Pandas data frames and basics plots using Matplotlib.

PROCEDURE:

Step 1:To create an ndarray
Step 2: Pass a list, tuple or any array-like object into the array() method,
Step3 :Converted into an ndarray

PROGRAM

```
# importing numpy module
import numpy as np

# creating list
list = [1, 2, 3, 4]

# creating numpy array
sample_array = np.array(list1)

print("List in python : ", list)

print("Numpy Array in python :",
      sample_array)
```

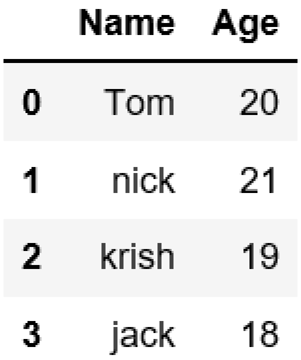
OUTPUT:

```
List in python :  [1, 2, 3, 4]
Numpy Array in python :  [1 2 3 4]
```

PROGRAM

```
# Python code demonstrate creating
# DataFrame from dict narray / lists
# By default addresses.

import pandas as pd

# intialise data of lists.
data = {'Name':['Tom', 'nick', 'krish', 'jack'],
        'Age':[20, 21, 19, 18]}

# Create DataFrame
df = pd.DataFrame(data)

# Print the output.
print(df)
```

OUTPUT:



	Name	Age
0	Tom	20
1	nick	21
2	krish	19
3	jack	18

RESULT:

Thus the program for Numpy arrays and Pandas data frames was executed and the output is verified successfully.

EX.NO: 4	**EXPLORE VARIOUS R CLEANING DATA**
DATE:	

AIM:

To perform various variable and row filters in R for cleaning data and apply various data sets and visualize.

PROCEDURE:

Step 1: Familiarize yourself with the data set
Step 2: Check for structural errors
Step 3: Check for data irregularities
Step 4: Decide how to deal with missing values
Step 5: Document data versions and changes made

PROGRAM:

Creating of Example data

```
data <- data.frame(x1 = c(1:4, 99999, 1, NA, 1, 1, NA),
 # Create example data frame
 x1 = c(1:5, 1, "NA", 1, 1, "NA"),
 x1 = c(letters[c(1:3)], "x  x",  "x", "   y    y y", "x", "a", "a", NA),
 x4 = "",x5 = NA)data
# Print example data frame
```

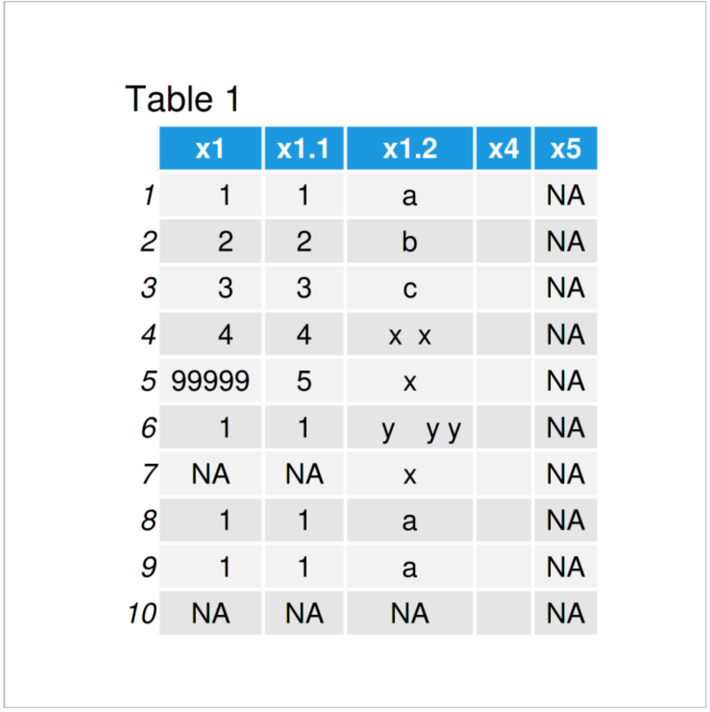


Table 1

	x1	x1.1	x1.2	x4	x5
1	1	1	a		NA
2	2	2	b		NA
3	3	3	c		NA
4	4	4	x x		NA
5	99999	5	x		NA
6	1	1	y y y		NA
7	NA	NA	x		NA
8	1	1	a		NA
9	1	1	a		NA
10	NA	NA	NA		NA

Remove Rows with Missing Values

```
data <- na.omit(data)                              # Delete rows with missing values
data                                   # Print updated data frame
```

OUTPUT:

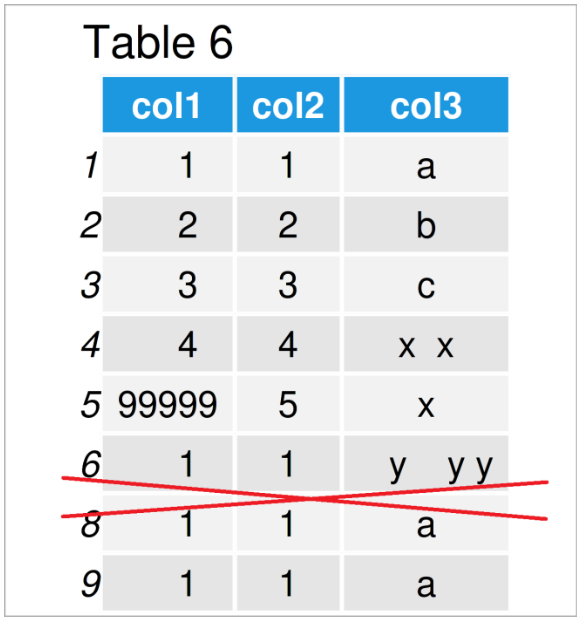


Table 6

	col1	col2	col3
1	1	1	a
2	2	2	b
3	3	3	c
4	4	4	x x
5	99999	5	x
~~6~~	~~1~~	~~1~~	~~y y y~~
~~8~~	~~1~~	~~1~~	~~a~~
9	1	1	a

RESULT:

Thus the program for various variable and row filters using R cleaning data was executed and the output is verified successfully.

EX.NO: 5	**TIME SERIES ANALYSIS TECHNIQUES**
DATE:	

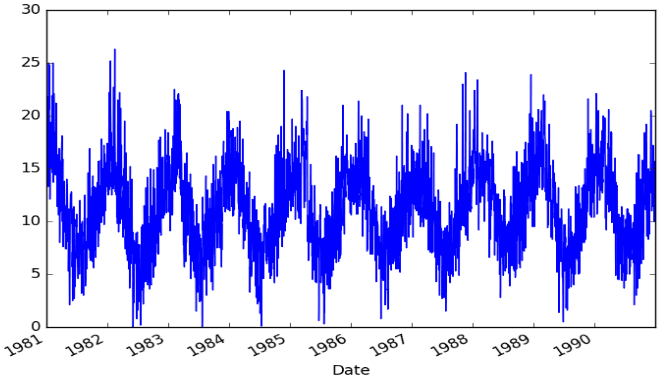
AIM:

To perform time series analysis and perform various visualization techniques.

PROCEDURE:

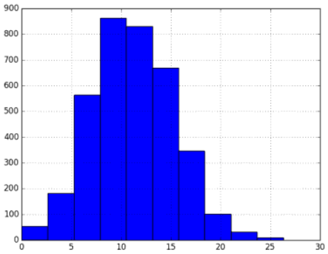
STEP 1: Time Series Line Plot

```
from pandas import read_csv
from matplotlib import pyplot
series = read_csv('daily-minimum-temperatures.csv', header=0, index_col=0,
parse_dates=True, squeeze=True)
series.plot()
pyplot.show()
```



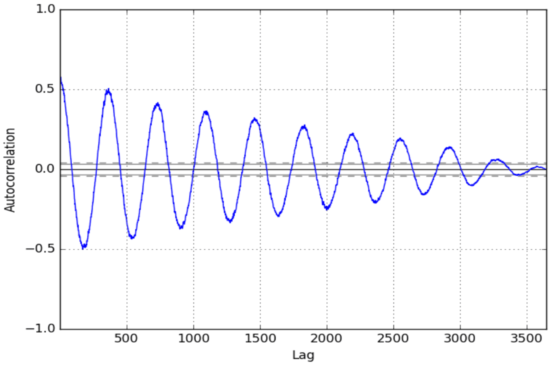
STEP 2: Time Series Histogram and Density Plots

```
from pandas import read_csv
from matplotlib import pyplot
series = read_csv('daily-minimum-temperatures.csv', header=0, index_col=0,
parse_dates=True, squeeze=True)
series.hist()
pyplot.show()
```



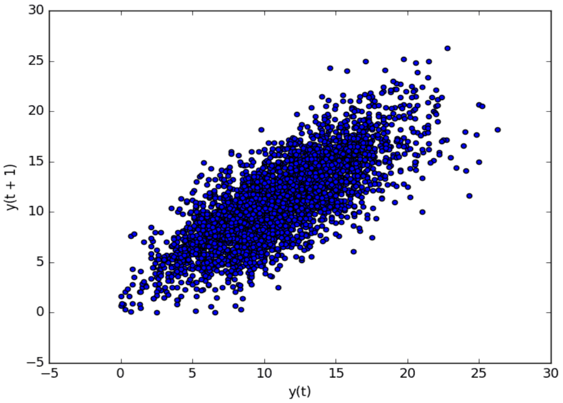
STEP 3 : Time Series Autocorrelation Plots

```
# create an autocorrelation plot
from pandas import read_csv
from matplotlib import pyplot
from pandas.plotting import autocorrelation_plot
series = read_csv('daily-minimum-temperatures.csv', header=0, index_col=0,
parse_dates=True, squeeze=True)
autocorrelation_plot(series)
pyplot.show()
```



STEP 4:Time Series Lag Scatter Plots

```
# create a scatter plot
from pandas import read_csv
from matplotlib import pyplot
from pandas.plotting import lag_plot
series = read_csv('daily-minimum-temperatures.csv', header=0, index_col=0,
parse_dates=True, squeeze=True)
lag_plot(series)
pyplot.show()
```



RESULT:

Thus the program for Time series analysis with various visualization techniques was executed and the output is verified sucessfully.

EX.NO: 6	**PERFORM DATA ANALYSIS ON A MAP**
DATE:	

AIM:
To perform data analysis and representation on a Map using various Mapdata sets.

PROCEDURE:

STEP 1: Installing Python Shapefile Library (PyShp)

```
pip install pyshp
```

STEP 2: Importing and initializing main Python libraries

```
import numpy as np
import pandas as pd
import shapefile as shp
import matplotlib.pyplot as plt
import seaborn as sns
```

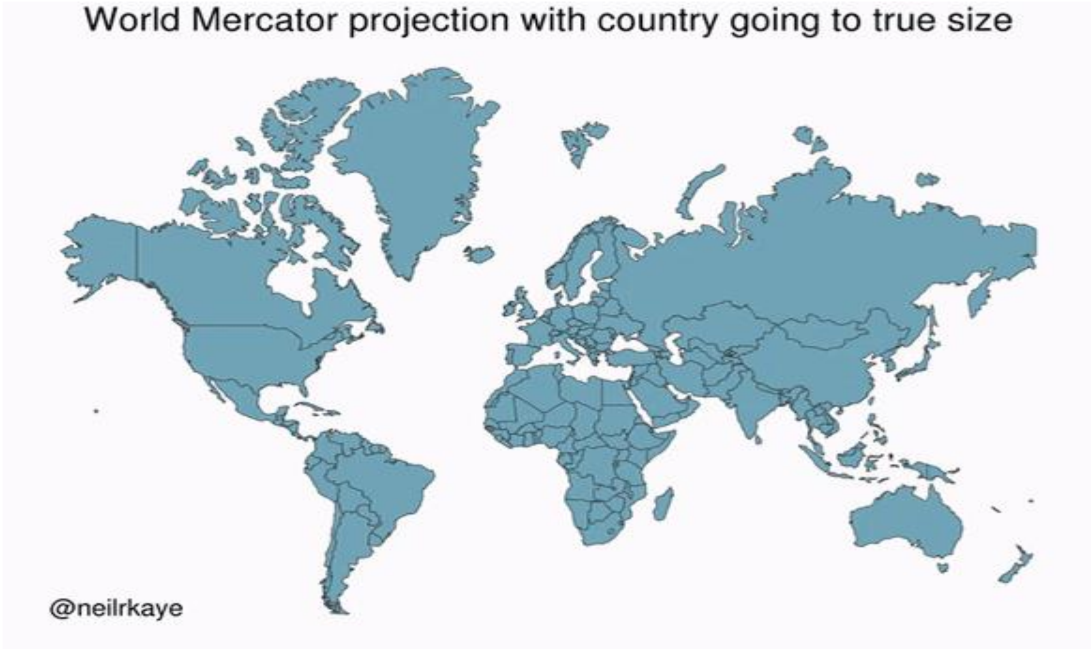
Initializing vizualization set

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

STEP 3: Opening a Vector Map

```
shp_path = “./Comunas_RM_Mapas_Vectoriales/Comuna.shp”
sf = shp.Reader(shp_path)
len(sf.shapes())
```

Centro
Norte
Oriente
Sureste
Poniente
Sur
Región Metropolitana



RESULT:

Thus the program for data analysis and representation on a map was executed and the output is verified successfully.

EX.NO: 7	**PERFORM CARTOGRAPHIC VISUALIZATION**
DATE:	

AIM:

To perform cartographic visualization for multiple datasets.

PROCEDURE:

Step 1 **:** Installing GeoPandas and Shapely

```
conda install -c conda-forge geopandas
pip install geopandas
```

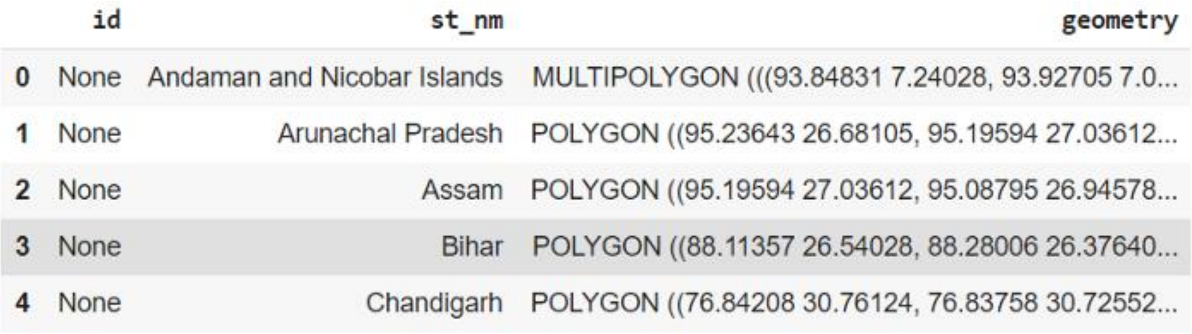
Step 2 : Importing the libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as pltimport seaborn as sns
import geopandas as gpd
import shapefile as shpfrom shapely.geometry import Pointsns.set_style('whitegrid')
```

Step 3 : Download the mapping data

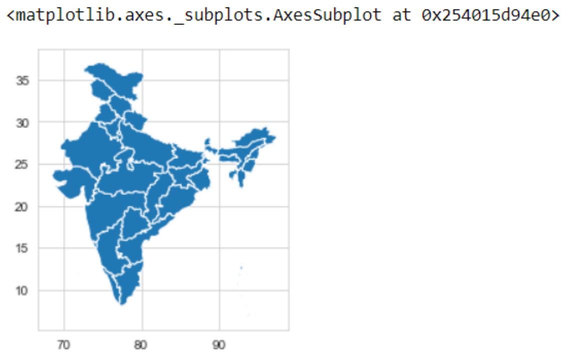
Step 4 : Load the data

```
fp = r'Maps_with_python\india-polygon.shp'
map_df = gpd.read_file(fp)
map_df_copy = gpd.read_file(fp)
map_df.head()
```



	id	st_nm	geometry
0	None	Andaman and Nicobar Islands	MULTIPOLYGON (((93.84831 7.24028, 93.92705 7.0...
1	None	Arunachal Pradesh	POLYGON ((95.23643 26.68105, 95.19594 27.03612...
2	None	Assam	POLYGON ((95.19594 27.03612, 95.08795 26.94578...
3	None	Bihar	POLYGON ((88.11357 26.54028, 88.28006 26.37640...
4	None	Chandigarh	POLYGON ((76.84208 30.76124, 76.83758 30.72552...

Step 5 : Plotting the Shapefiles



```
<matplotlib.axes._subplots.AxesSubplot at 0x254015d94e0>
```

Step 6 : Adding better data insights into the map

```
df = pd.read_csv('globallandslides.csv')
pd.set_option('display.max_columns', None)df = df[df.country_name=="India"]
df["Year"] = pd.to_datetime(df["event_date"]).dt.year
df = df[df.landslide_category=="landslide"]ls_df["admin_division_name"].replace("Nāgāland",
"Nagaland",inplace = True)
ls_df["admin_division_name"].replace("Meghālaya", "Meghalaya",inplace = True)
ls_df["admin_division_name"].replace("Tamil Nādu", "Tamil Nadu",inplace = True)
ls_df["admin_division_name"].replace("Karnātaka", "Karnataka",inplace = True)
ls_df["admin_division_name"].replace("Gujarāt", "Gujarat",inplace = True)
ls_df["admin_division_name"].replace("Arunāchal Pradesh", "Arunachal Pradesh",inplace =
True)state_df = ls_df["admin_division_name"].value_counts()
state_df = state_df.to_frame()
state_df.reset_index(level=0, inplace=True)
state_df.columns = ['State', 'Count']state_df.at[15,"Count"] = 69
state_df.at[0,"State"] = "Jammu and Kashmir"       state_df.at[20,"State"] = "Delhi"
state_df.drop(7)
```

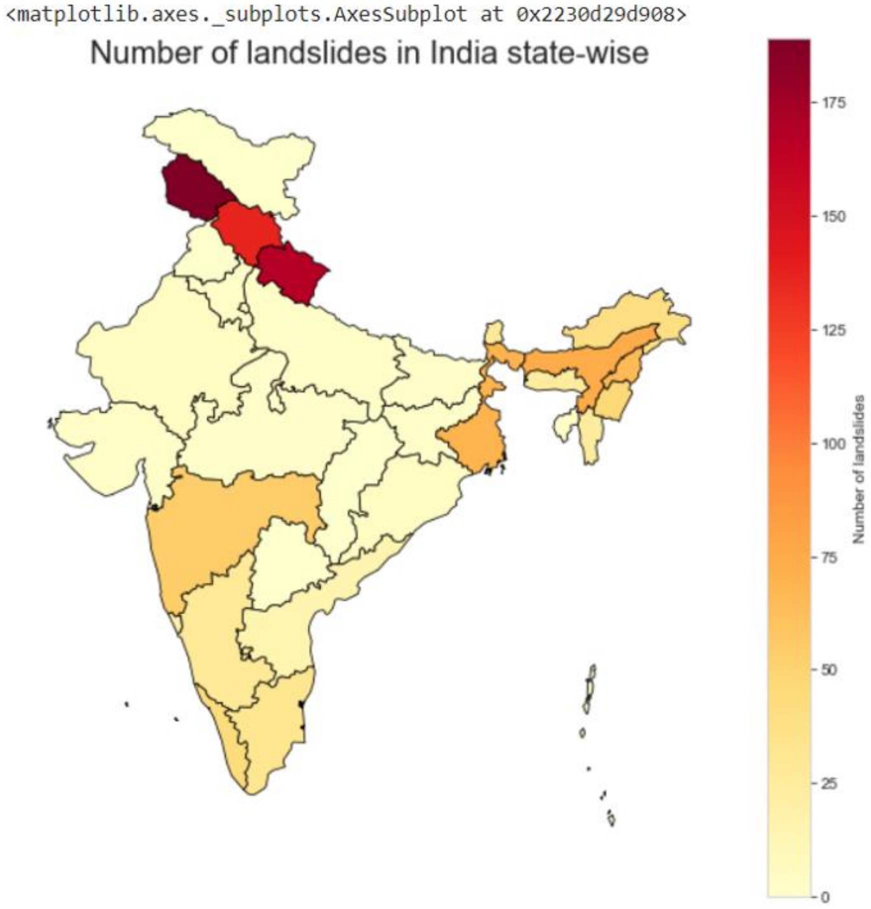
Step 7 : Merge the data

```
#Merging the data
merged = map_df.set_index('st_nm').join(state_df.set_index('State'))
merged['Count'] = merged['Count'].replace(np.nan, 0)
merged.head()
```

Step 8 : Plotting the data on the Shapefile

```
#Create figure and axes for Matplotlib and set the title
fig, ax = plt.subplots(1, figsize=(10, 10))
ax.axis('off')ax.set_title('Number of landslides in India state-wise', fontdict={'fontsize': '20',
'fontweight' : '10'})# Plot the figure
```

```
merged.plot(column='Count',cmap='YlOrRd', linewidth=0.8, ax=ax,
edgecolor='0',legend=True,markersize=[39.739192, -104.990337], legend_kwds={'label':
"Number of landslides"})
```



```
<matplotlib.axes._subplots.AxesSubplot at 0x2230d29d908>
```

RESULT:

Thus the program for cartographic visualization for multiple datasets was executed and the output is verified successfully.

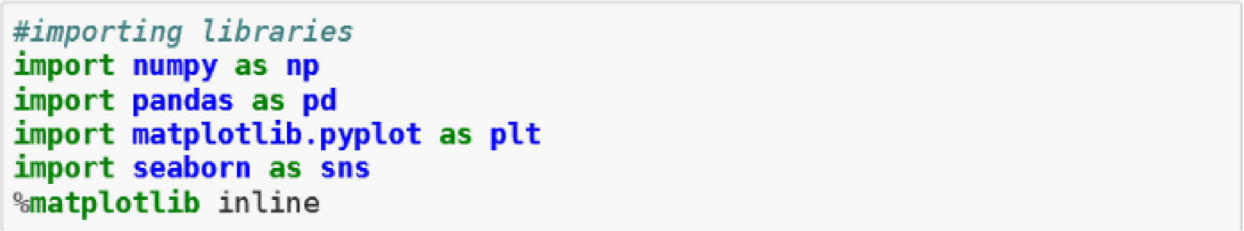
EX.NO: 8	**PERFORM EDA ON WINE QUALITY DATA SET**
DATE:	

AIM:

To perform EDA on Wine quality data set.

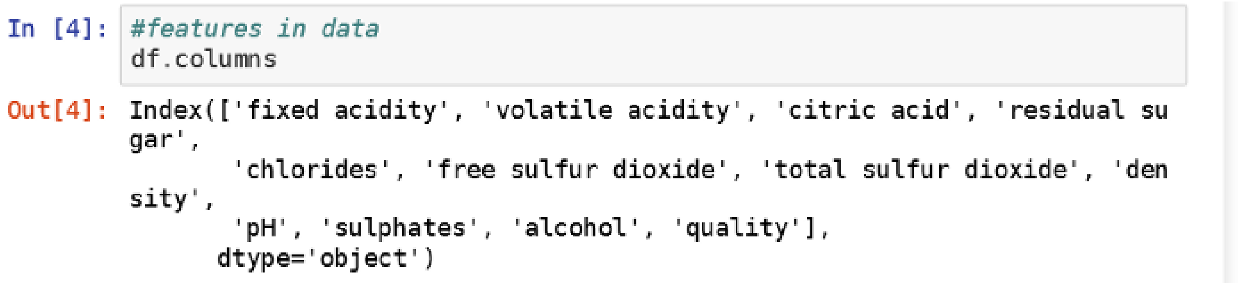
PROCEDURE:

STEP1: Import some essential libraries in Python.



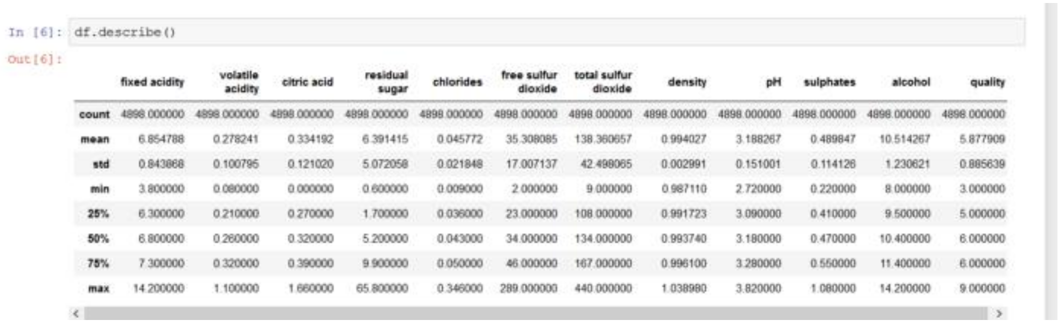
```
#importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

STEP 2: The columns of the data, we can do df.columns, it will give all the features name present in the data.



```
In [4]: #features in data
        df.columns

Out[4]: Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual su
        gar',
               'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'den
        sity',
               'pH', 'sulphates', 'alcohol', 'quality'],
              dtype='object')
```

- STEP 3: The describe () function in Python summarizes statistics. This function returns the count, mean, standard deviation, minimum and maximum values, and the quantiles of the data.



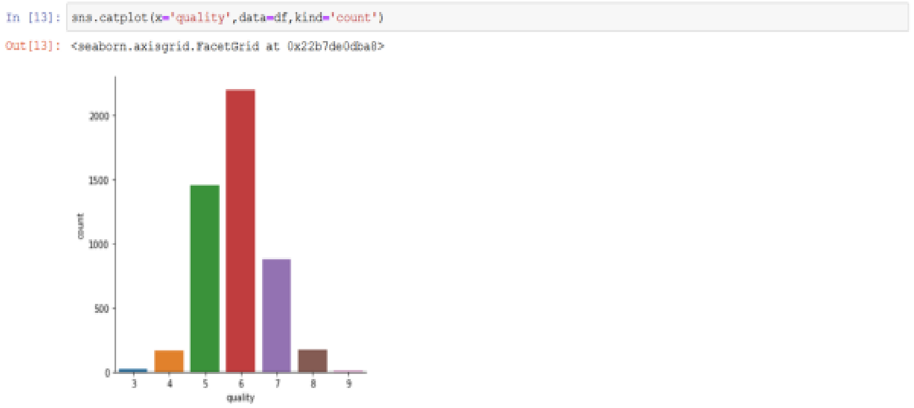
```
In [6]: df.describe()
Out[6]:
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
count	4898.000000	4898.000000	4898.000000	4898.000000	4898.000000	4898.000000	4898.000000	4898.000000	4898.000000	4898.000000	4898.000000	4898.000000
mean	6.854788	0.278241	0.334192	6.391415	0.045772	35.308085	138.360657	0.994027	3.188267	0.489847	10.514267	5.877909
std	0.843868	0.100795	0.121020	5.072058	0.021848	17.007137	42.498065	0.002991	0.151001	0.114126	1.230621	0.885639
min	3.800000	0.080000	0.000000	0.600000	0.009000	2.000000	9.000000	0.987110	2.720000	0.220000	8.000000	3.000000
25%	6.300000	0.210000	0.270000	1.700000	0.036000	23.000000	108.000000	0.991723	3.090000	0.410000	9.500000	5.000000
50%	6.800000	0.260000	0.320000	5.200000	0.043000	34.000000	134.000000	0.993740	3.180000	0.470000	10.400000	6.000000
75%	7.300000	0.320000	0.390000	9.900000	0.050000	46.000000	167.000000	0.996100	3.280000	0.550000	11.400000	6.000000
max	14.200000	1.100000	1.660000	65.800000	0.346000	289.000000	440.000000	1.038980	3.820000	1.080000	14.200000	9.000000

STEP 4: The feature that has a maximum unique value is *density*.

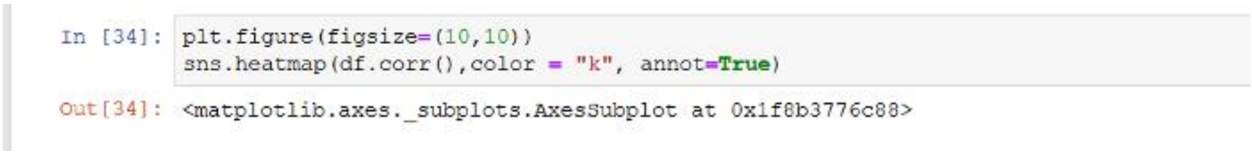
The feature that has a minimum unique value is quality.

seaborn.catplot — show the relationship between a numerical and one or more categorical variables using one of several visual representations.

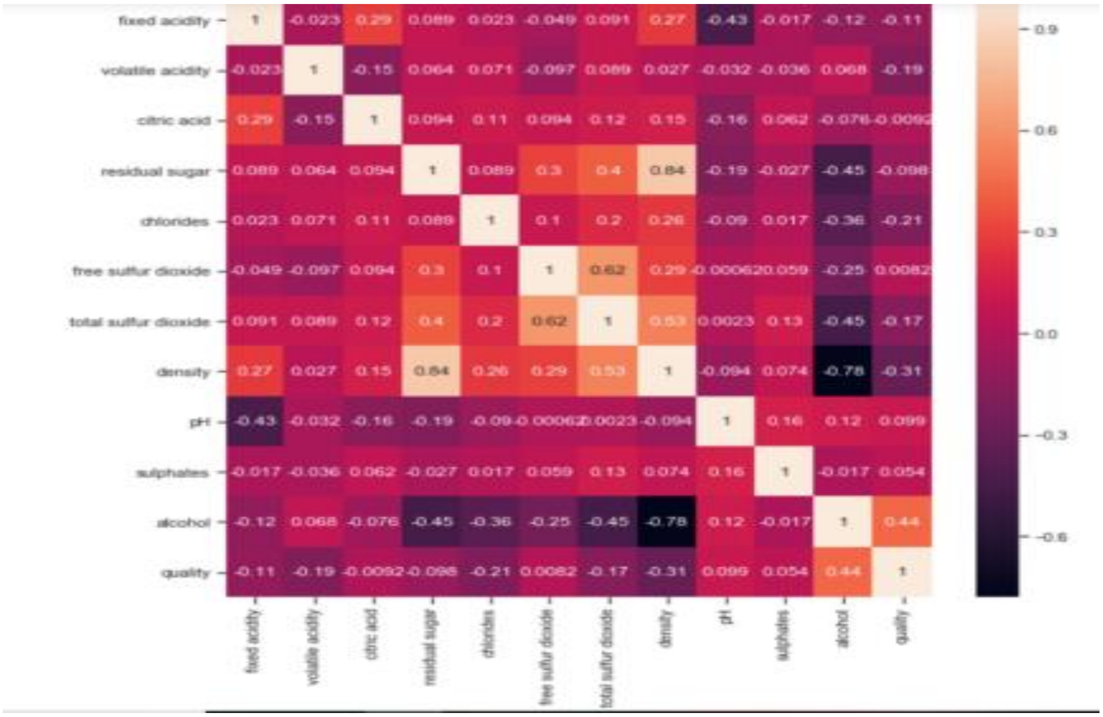


```
In [13]: sns.catplot(x='quality',data=df,kind='count')
Out[13]: <seaborn.axisgrid.FacetGrid at 0x22b7de0dba8>
```

STEP 5: Find correlations using pandas ".corr()" function and can visualize the correlation matrix using a heatmap in seaborn.



```
In [34]: plt.figure(figsize=(10,10))
         sns.heatmap(df.corr(),color = "k", annot=True)
Out[34]: <matplotlib.axes._subplots.AxesSubplot at 0x1f8b3776c88>
```



RESULT:

Thus the program for EDA on Wine Quality Data Set is executed and the output is verified successfully.

EX.NO: 9	**VISUALIZATION TECHNIQUES**
DATE:	

AIM:

To perform various EDA and Visualization techniques for analysis report.

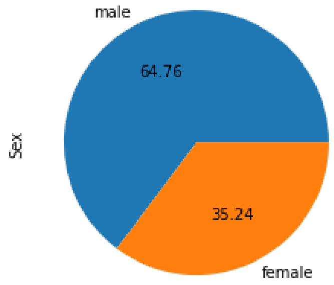
PROCEDURE:

STEP1: Importing libraries and loading Data

```
import numpy as np

import pandas pd

import matplotlib.pyplot as plt

import seaborn as sns

from seaborn import load_dataset

#titanic dataset

data = pd.read_csv("titanic_train.csv")

#tips dataset

tips = load_dataset("tips")
```

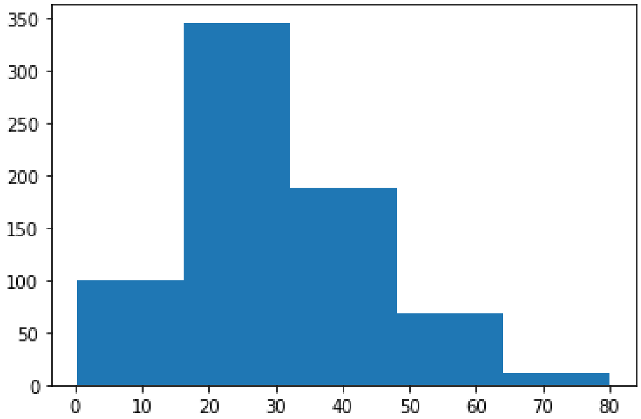
STEP 2: Pie Chart

```
data['Sex'].value_counts().plot(kind="pie", autopct="%.2f")

plt.show()
```



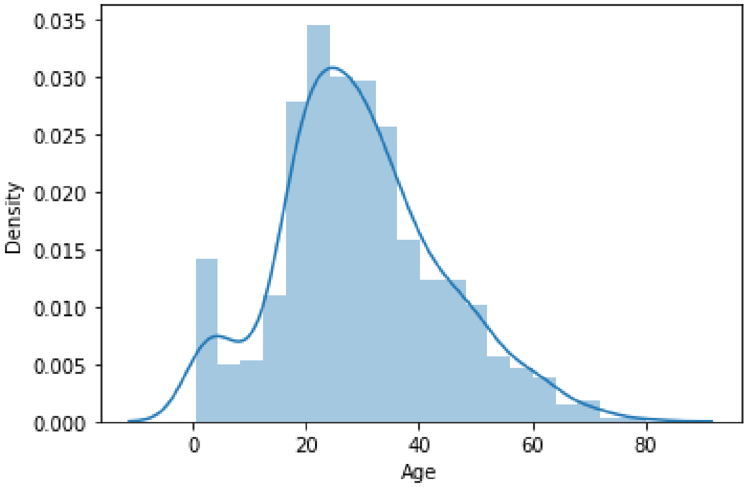
STEP 3: Histogram

```
plt.hist(data['Age'], bins=5)

plt.show()
```



STEP 4: Distplot

```
    sns.distplot(data['Age'])
     plt.show()
```

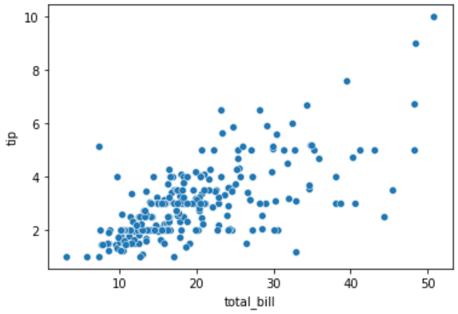


STEP 5: Boxplot

IQR = Q3 - Q1

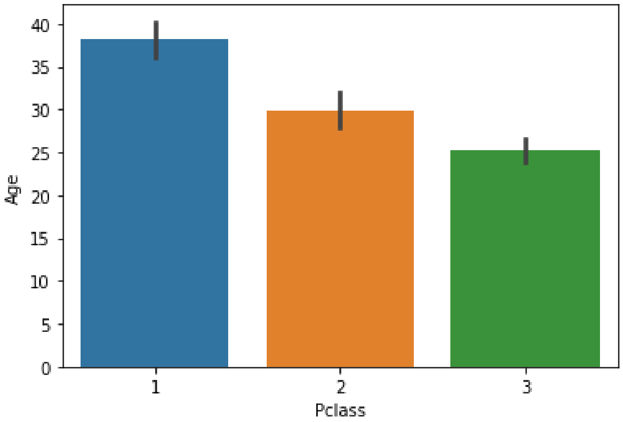
Lower_boundary = Q1 - 1.5 * IQR

Upper_bounday = Q3 + 1.5 * IQR



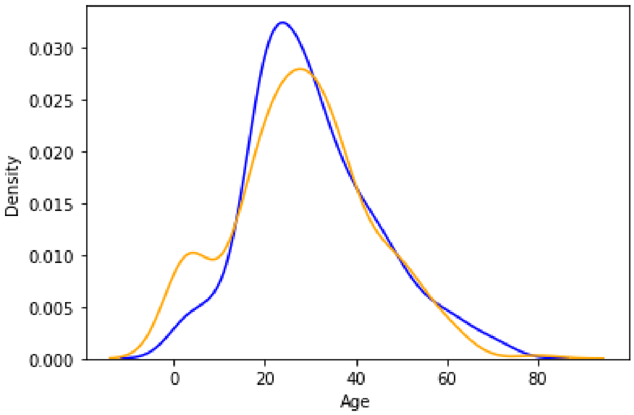
STEP 6: Bar Plot

```
sns.barplot(data['Pclass'], data['Age'])
plt.show()
```



Distplot

```
sns.distplot(data[data['Survived'] == 0]['Age'], hist=False, color="blue")
sns.distplot(data[data['Survived'] == 1]['Age'], hist=False, color="orange")
plt.show()
```



RESULT:
Thus the program for data set of various EDA is executed and the output is verified successfully.