Ex 1. DATE:	**Install the data Analysis and Visualization tool: R/ Python /Power BI.**

Aim:

To install the data analysis and visualization tool in python.

Algorithm:

1. Install Pandas in Python
2. Create a dataframe in pandas using pd.series method
3. Convert the data in csv file into pandas dataframe
4. Indexing dataframes with pandas using pandas.dataframe.iloc method
5. Indexing using Labels in pandas using pandas.dataframe.locmethod
6. Install matplotlib in python
7. Plot the dataframe using pandas

Program:

1. Installation:

pip install pandas

2. Creating A DataFrame in Pandas:

```
# assigning two series to s1 and s2s1 =
pd.Series([1,2])
s2 = pd.Series(["Ashish", "Sid"]) #
framing series objects into datadf =
pd.DataFrame([s1,s2])
# show the data framedf
# data framing in another way
# 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

3. Importing Data with Pandas

```
# Import the pandas library, renamed as pdimport
pandas as pd
# Read IND_data.csv into a DataFrame, assigned to dfdf =
pd.read_csv("IND_data.csv")
# Prints the first 5 rows of a DataFrame as defaultdf.head()
# Prints no. of rows and columns of a DataFramedf.shape
```

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]
```

5. Indexing Using Labels in Pandas

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

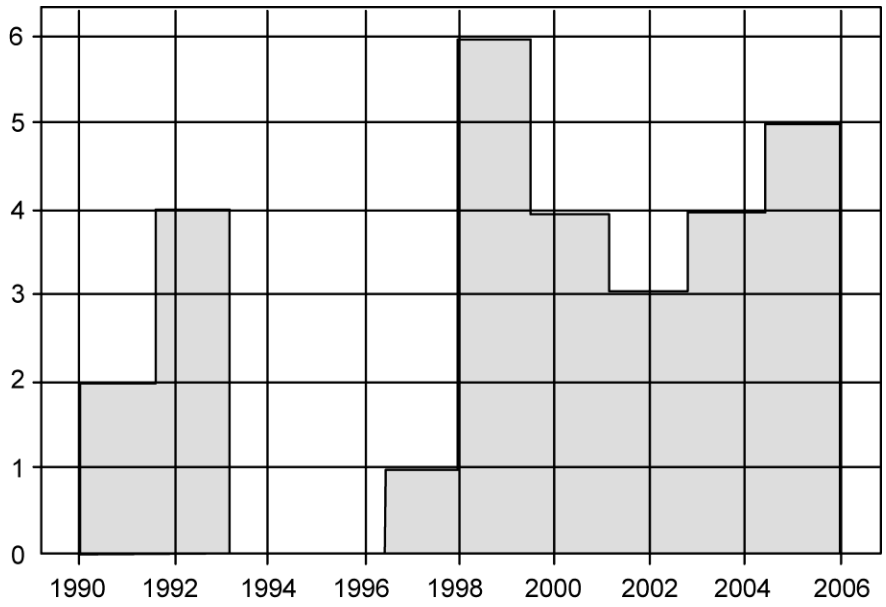
6. Installation

```
pip install matplotlib
```

7. 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:



Result:

Thus the program to Install the data Analysis and Visualization tool using python has been executed successfully.

Ex 2. DATE:	**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.**

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.

ALGORITHM:

1. Import the required libraries.
2. Create email dataset in python.
3. Export and all your email datasets.
4. Visualize the email dataset

PROGRAM:

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

!pip install mailbox

from google.colab import drive

drive.mount('/content/gdrive')

import mailbox

mboxfile = "gdrive/My Drive/Colab Notebooks/gmail.mbox"

mbox = mailbox.mbox(mboxfile)

mbox

for key in mbox[0].keys():
```

```
  print(key)
import csv
with open('mailbox.csv', 'w') as outputfile:
  writer = csv.writer(outputfile)
  writer.writerow(['subject','from','date','to','label','thread'])
  for message in mbox:
    writer.writerow([message['subject'], message['from'],  message['date'], message['to'],
message['X-Gmail-Labels'], message['X-GM-THRID']])
dfs = pd.read_csv('mailbox.csv', names=['subject', 'from', 'date', 'to', 'label', 'thread'])
dfs.dtypes
dfs['date'] = dfs['date'].apply(lambda x: pd.to_datetime(x, errors='coerce', utc=True))
dfs = dfs[dfs['date'].notna()]
dfs.to_csv('gmail.csv')
dfs.info()
dfs.head(10)
dfs.columns
def extract_email_ID(string):
  email = re.findall(r'<(.+?)>', string)
  if not email:
    email = list(filter(lambda y: '@' in y, string.split()))
  return email[0] if email else np.nan
dfs['from'] = dfs['from'].apply(lambda x: extract_email_ID(x))
myemail = 'itsmeskm99@gmail.com'
dfs['label'] = dfs['from'].apply(lambda x: 'sent' if x==myemail else 'inbox')
dfs.drop(columns='to', inplace=True)
```

```
dfs.head(10)
import datetime
import pytz
def refactor_timezone(x):
  est = pytz.timezone('US/Eastern')
  return x.astimezone(est)
dfs['date'] = dfs['date'].apply(lambda x: refactor_timezone(x))
dfs['dayofweek'] = dfs['date'].apply(lambda x: x.weekday_name)
dfs['dayofweek'] = pd.Categorical(dfs['dayofweek'], categories=[
   'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday',
   'Saturday', 'Sunday'], ordered=True)
dfs['timeofday'] = dfs['date'].apply(lambda x: x.hour + x.minute/60 + x.second/3600)
dfs['hour'] = dfs['date'].apply(lambda x: x.hour)
dfs['year_int'] = dfs['date'].apply(lambda x: x.year)
dfs['year'] = dfs['date'].apply(lambda x: x.year + x.dayofyear/365.25)
dfs.index = dfs['date']
del dfs['date']
print(dfs.index.min().strftime('%a, %d %b %Y %I:%M %p'))
print(dfs.index.max().strftime('%a, %d %b %Y %I:%M %p'))
print(dfs['label'].value_counts())
import matplotlib.pyplot as plt
from matplotlib.ticker import MaxNLocator
def plot_todo_vs_year(df, ax, color='C0', s=0.5, title=''):
  ind = np.zeros(len(df), dtype='bool')
  est = pytz.timezone('US/Eastern')
```

```
  df[~ind].plot.scatter('year', 'timeofday', s=s, alpha=0.6, ax=ax, color=color)
  ax.set_ylim(0, 24)
  ax.yaxis.set_major_locator(MaxNLocator(8))
  ax.set_yticklabels([datetime.datetime.strptime(str(int(np.mod(ts, 24))), "%H").strftime("%I
%p") for ts in ax.get_yticks()]);
  ax.set_xlabel('')
  ax.set_ylabel('')
  ax.set_title(title)
  ax.grid(ls=':', color='k')
  return ax
sent = dfs[dfs['label']=='sent']
received = dfs[dfs['label']=='inbox']
fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(15, 4))

plot_todo_vs_year(sent, ax[0], title='Sent')
plot_todo_vs_year(received, ax[1], title='Received')
def plot_number_perday_per_year(df, ax, label=None, dt=0.3, **plot_kwargs):
   year = df[df['year'].notna()]['year'].values
   T = year.max() - year.min()
   bins = int(T / dt)
   weights = 1 / (np.ones_like(year) * dt * 365.25)
   ax.hist(year, bins=bins, weights=weights, label=label, **plot_kwargs);
   ax.grid(ls=':', color='k')
from scipy import ndimage
```

```
def plot_number_perdhour_per_year(df, ax, label=None, dt=1, smooth=False,
                                  weight_fun=None, **plot_kwargs):
    tod = df[df['timeofday'].notna()]['timeofday'].values
    year = df[df['year'].notna()]['year'].values
    Ty = year.max() - year.min()
    T = tod.max() - tod.min()
    bins = int(T / dt)
    if weight_fun is None:
        weights = 1 / (np.ones_like(tod) * Ty * 365.25 / dt)
    else:
        weights = weight_fun(df)
    if smooth:
        hst, xedges = np.histogram(tod, bins=bins, weights=weights);
        x = np.delete(xedges, -1) + 0.5*(xedges[1] - xedges[0])
        hst = ndimage.gaussian_filter(hst, sigma=0.75)
        f = interp1d(x, hst, kind='cubic')
        x = np.linspace(x.min(), x.max(), 10000)
        hst = f(x)
        ax.plot(x, hst, label=label, **plot_kwargs)
    else:
        ax.hist(tod, bins=bins, weights=weights, label=label, **plot_kwargs);
ax.grid(ls=':', color='k')
    orientation = plot_kwargs.get('orientation')
    if orientation is None or orientation == 'vertical':
        ax.set_xlim(0, 24)
```

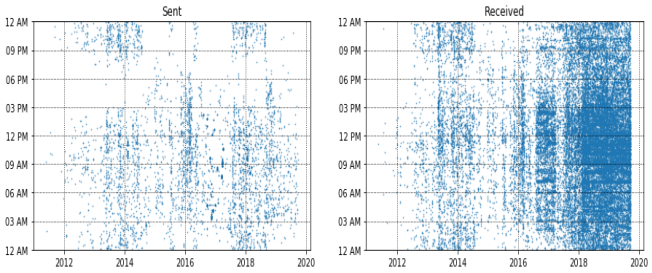
```
        ax.xaxis.set_major_locator(MaxNLocator(8))
        ax.set_xticklabels([datetime.datetime.strptime(str(int(np.mod(ts, 24))), "%H").strftime("%I %p")
                            for ts in ax.get_xticks()]);
    elif orientation == 'horizontal':
        ax.set_ylim(0, 24)
        ax.yaxis.set_major_locator(MaxNLocator(8))
        ax.set_yticklabels([datetime.datetime.strptime(str(int(np.mod(ts, 24))), "%H").strftime("%I %p")
                            for ts in ax.get_yticks()]);
class TriplePlot:
  def __init__(self):
    gs = gridspec.GridSpec(6, 6)
    self.ax1 = plt.subplot(gs[2:6, :4])
    self.ax2 = plt.subplot(gs[2:6, 4:6], sharey=self.ax1)
    plt.setp(self.ax2.get_yticklabels(), visible=False);
    self.ax3 = plt.subplot(gs[:2, :4])
    plt.setp(self.ax3.get_xticklabels(), visible=False);
  def plot(self, df, color='darkblue', alpha=0.8, markersize=0.5, yr_bin=0.1, hr_bin=0.5):
    plot_todo_vs_year(df, self.ax1, color=color, s=markersize)
    plot_number_perdhour_per_year(df, self.ax2, dt=hr_bin, color=color, alpha=alpha, orientation='horizontal')
    self.ax2.set_xlabel('Average emails per hour')
    plot_number_perday_per_year(df, self.ax3, dt=yr_bin, color=color, alpha=alpha)
    self.ax3.set_ylabel('Average emails per day')
for ct, addr in enumerate(addrs.index[idx]):
```

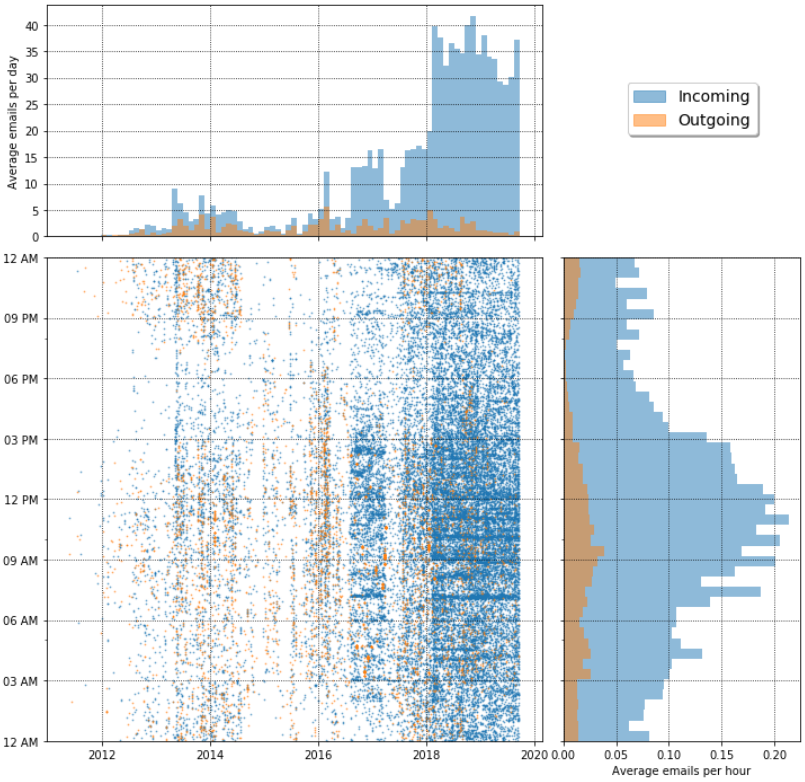
```
    tpl.plot(dfs[dfs['from'] == addr], color=colors[ct], alpha=0.3, yr_bin=0.5, markersize=1.0)
    labels.append(mpatches.Patch(color=colors[ct], label=addrs[0:4], alpha=0.5))
plt.legend(handles=labels, bbox_to_anchor=[1.4, 0.9], fontsize=12, shadow=True);
     sdw = sent.groupby('dayofweek').size() / len(sent)
rdw = received.groupby('dayofweek').size() / len(received)
df_tmp = pd.DataFrame(data={'Outgoing Email': sdw, 'Incoming Email':rdw})
df_tmp.plot(kind='bar', rot=45, figsize=(8,5), alpha=0.5)
plt.xlabel('');
plt.ylabel('Fraction of weekly emails');
plt.grid(ls=':', color='k', alpha=0.5)
import scipy.ndimage
from scipy.interpolate import interp1d
plt.figure(figsize=(8,5))
ax = plt.subplot(111)
for ct, dow in enumerate(dfs.dayofweek.cat.categories):
    df_r = received[received['dayofweek']==dow]
    weights = np.ones(len(df_r)) / len(received)
    plot_number_perdhour_per_year(df_r, ax, dt=1, smooth=True, color=f'C{ct}',
                    alpha=0.8, lw=3, label=dow, weight_fun=wfun)
    df_s = sent[sent['dayofweek']==dow]
    weights = np.ones(len(df_s)) / len(sent)
    wfun = lambda x: weights
    plot_number_perdhour_per_year(df_s, ax, dt=1, smooth=True, color=f'C{ct}',
                    alpha=0.8, lw=2, label=dow, ls='--', weight_fun=wfun)
ax.set_ylabel('Fraction of weekly emails per hour')
```

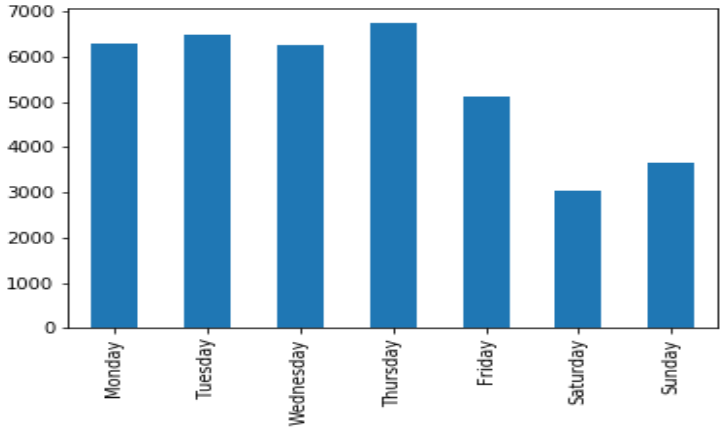
```
plt.legend(loc='upper left')
```

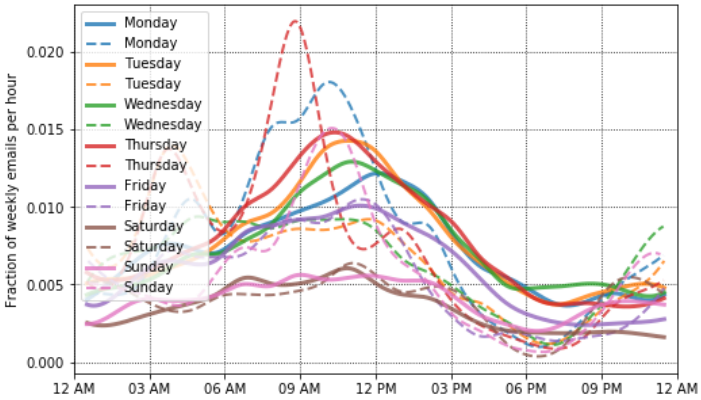
OUTPUT:

subject	from	date	label	thread	
1	New Books: The Python Journeyman + Understandi...	james@sitepoint.com	2019-09-20 14:07:05 +00:00	inbox	1645216686186738105
2	iPhone 11 Pro og iPhone 11 er her	News_Europe@InsideApple.Apple.com	2019-09-20 10:33:27 +00:00	inbox	1645190169696380553
3	=?utf-8?Q?Save=20on=20Burlap=20Bags=20Today=21...	support@totebagfactory.com	2019-09-20 15:32:31 +00:00	inbox	1645209548975264659
4	Hi there, looking for the best Dashain deals? ...	info@email.daraz.com.np	2019-09-17 06:19:10 +00:00	inbox	1644916038153843699
5	The file =?UTF-8?B?J0JyYW5kX0Jvb2sgdGVzdC5wZGY...	noreply@box.com	2019-09-20 19:04:16 +00:00	inbox	1645222431795507661


Sent
Received
12 AM
09 PM
06 PM
03 PM
12 PM
09 AM
06 AM
03 AM
12 AM
2012
2014
2016
2018
2020


Average emails per day
Incoming
Outgoing
12 AM
09 PM
06 PM
03 PM
12 PM
09 AM
06 AM
03 AM
12 AM
2012
2014
2016
2018
2020
0.00
0.05
0.10
0.15
0.20
Average emails per hour


7000
6000
5000
4000
3000
2000
1000
0
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday



RESULT:

Thus the program tp perform exploratory data analysis (EDA) on with datasets like email data set and exporting all your emails as a dataset, import them inside a pandas data frame, visualize them and get different insights from the data was executed and implemented successfully.

Ex 3. DATE:	**Working with Nupy arrays, Pandas data frames, Basic plots using Matplotlib.Numpy arrays using matplotlib**

Aim:

To Work with Numpy arrays, Pandas data frames, Basic plots using Matplotlib. Numpy arrays using matplotlib

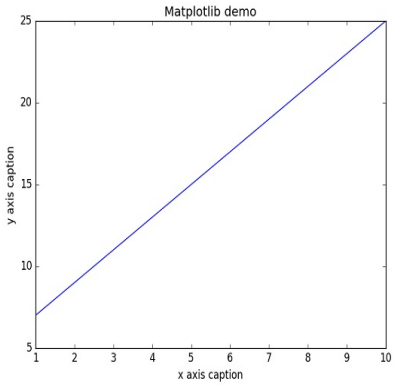
Algorithm:

1. Import the packages
2. Import numpy and matplotlib libraries in python.
3. np.arange() function as the values on the x axis. The corresponding values on the y axis are stored in another ndarray object y.
4. pyplot() is the most important function in matplotlib library, which is used to plot 2D data. The following script plots the equation y = 2x + 5.
5. Show the plot

Program:

```
import numpy as np
from matplotlib import pyplot as plt
x = np.arange(1,11)
y = 2 * x + 5
plt.title("Matplotlib demo")
plt.xlabel("x axis caption")
plt.ylabel("y axis caption")
plt.plot(x,y)
plt.show()
```

Output:

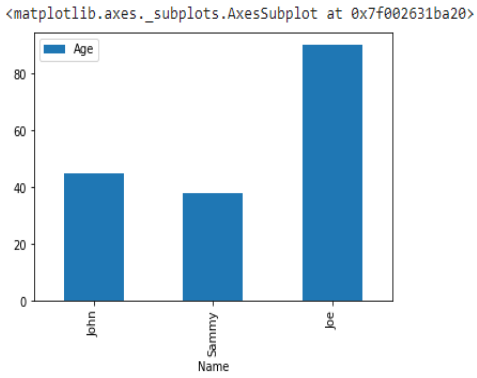


Result:

Thus the given program has been executed successfully.

3(b)Pandas dataframe using matplotlib

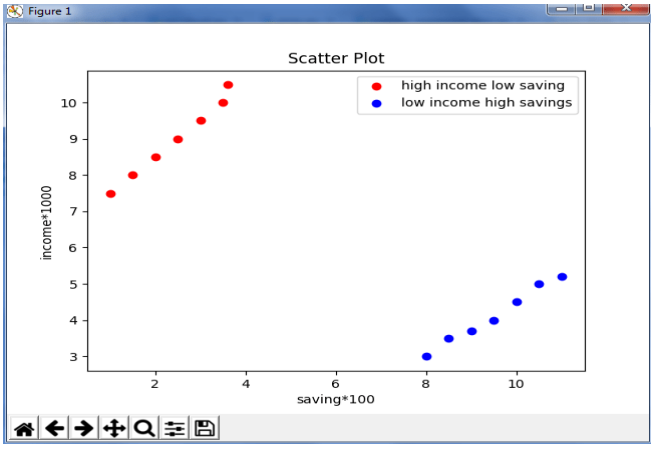
```
import pandas as pd

import matplotlib.pyplot as plt

df = pd.DataFrame({'Name': ['John', 'Sammy', 'Joe'],'Age': [45, 38, 90]})

df.plot(x="Name", y="Age", kind="bar")
```



Basic plots

```
import matplotlib.pyplot as plt
x = [1,1.5,2,2.5,3,3.5,3.6]
y = [7.5,8,8.5,9,9.5,10,10.5]
x1=[8,8.5,9,9.5,10,10.5,11]
y1=[3,3.5,3.7,4,4.5,5,5.2]
```

```
 plt.scatter(x,y, label='high income low saving',color='r')
plt.scatter(x1,y1,label='low income high savings',color='b')
plt.xlabel('saving*100')
plt.ylabel('income*1000')
plt.title('Scatter Plot')
plt.legend()
plt.show()
```



Result:

Thus the given program to work with Numpy arrays, Pandas data frames, Basic plots using Matplotlib,Numpy arrays using matplotlib has been executed successfully.

Ex 4. DATE:	**Explore various variable and row filters in python for cleaning data. Apply various plot features in python on sample data sets and visualize**

Aim:

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

Algorithm:

1. Load the data
2. View the first few rows
3. Use the subset function on rows and columns
4. Filter the rows and columns based on condition
5. Create a scatter plot and add a line of best fit
6. Similarly create bar plot,histogram and box plot

Program:

```
# import the pandas library

import pandas as pd

import numpy as np

df = pd.DataFrame(np.random.randn(5, 3), index=['a', 'c', 'e', 'f',

'h'],columns=['one', 'two', 'three'])

df = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])

print df
```

Output:

```
        one       two         three
a  0.077988  0.476149  0.965836
b       NaN       NaN       NaN
c -0.390208 -0.551605 -2.301950
d       NaN       NaN       NaN
e -2.000303 -0.788201  1.510072
```

Program :(Checking duplicate)

```
import pandas as pd
import numpy as np

df = pd.DataFrame(np.random.randn(5, 3), index=['a', 'c', 'e', 'f',
'h'],columns=['one', 'two', 'three'])

df = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])

print df['one'].isnull()
```

Output:

```
a  False
b  True
c  False
d  True
e  False
f  False
g  True
h  False
Name: one, dtype: bool
```

Program:(filling missing data)

```
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.randn(3, 3), index=['a', 'c', 'e'],columns=['one',
'two', 'three'])
df = df.reindex(['a', 'b', 'c'])
print df
print ("NaN replaced with '0':")
print df.fillna(0)
```

Output:

```
        one       two     three
a -0.576991 -0.741695  0.553172
b       NaN       NaN       NaN
c  0.744328 -1.735166  1.749580


NaN replaced with '0':
```

	one	two	three
a	-0.576991	-0.741695	0.553172
b	0.000000	0.000000	0.000000
c	0.744328	-1.735166	1.749580

Program:(Drop missing values)

```
import pandas as pd
import numpy as np

df = pd.DataFrame(np.random.randn(5, 3), index=['a', 'c', 'e', 'f',
'h'],columns=['one', 'two', 'three'])

df = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])
print df.dropna()
```

Output:

	one	two	three
a	0.077988	0.476149	0.965836
c	-0.390208	-0.551605	-2.301950
e	-2.000303	-0.788201	1.510072
f	-0.930230	-0.670473	1.14661

Program:(Replace missing or generic values)

```
import pandas as pd
import numpy as np
df = pd.DataFrame({'one':[10,20,30,40,50,2000],
'two':[1000,0,30,40,50,60]})
print df.replace({1000:10,2000:60})
```

Output:

	one	two
0	10	10
1	20	0
2	30	30
3	40	40
4	50	50
5	60	60

Result:

Thus the given program explore various variable and row filters in python for cleaning data and applying various plot features in python has been executed successfully.

Ex 5. **DATE:**	**Perform Time Series Analysis and apply the various visualization techniques**.

Aim:

To perform time series analysis and apply the various visualization techniques.

Algorithm:

1. Install the necessary packages.
2. Load the data.
3. Convert the column to date time.
4. Set date column as index.
5. Perform seasonal decomposition.
6. Plot the original data,trend,seasonal and residuals.
7. Show the plot.

Program:

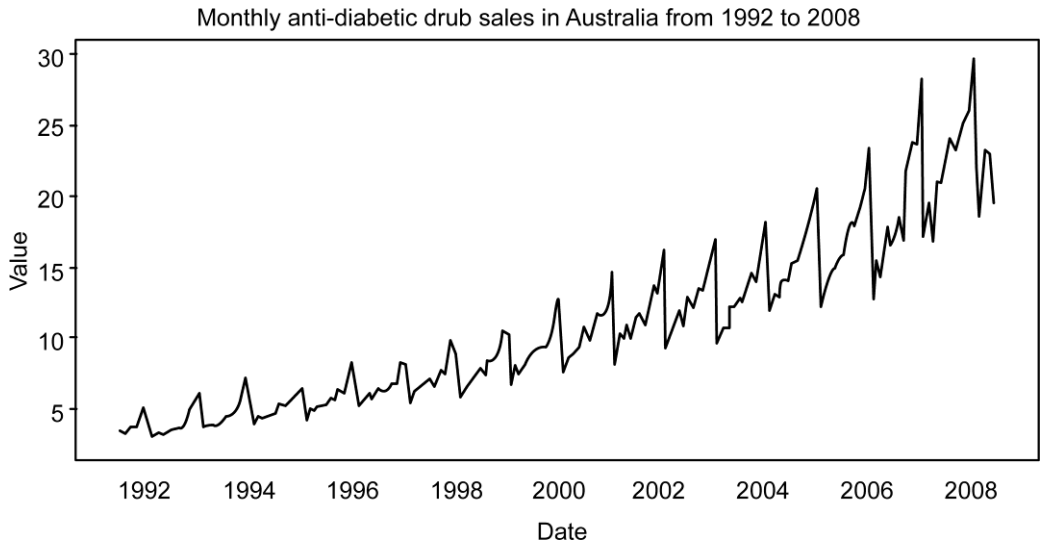
```
import matplotlib as mpl import
matplotlib.pyplot as plt import
seaborn as sns
import numpy as np
import pandas as pd
plt.rcParams.update({'figure.figsize': (10, 7), 'figure.dpi': 120})#
Import as Dataframe
df=pd.read_csv('https://raw.githubusercontent.com/selva86/datasets/master/a10.csv',
parse_dates=['date'])
df.head()
```

	Date	Value
0	1991-07-01	3.526591
1	1991-08-01	3.180891
2	1991-09-01	3.252221
3	1991-10-01	3.611003
4	1991-11-01	3.565869

Time series data source: fpp pacakge in R.

```
import matplotlib.pyplot as plt
df=pd.read_csv('https://raw.githubusercontent.com/selva86/datasets/master/a10.csv', parse_dates=['date'], index_col='date')
# Draw Plot
def plot_df(df, x, y, title="", xlabel='Date', ylabel='Value', dpi=100):
    plt.figure(figsize=(16,5), dpi=dpi)
    plt.plot(x, y, color='tab:red')
    plt.gca().set(title=title, xlabel=xlabel, ylabel=ylabel)
    plt.show()
    plot_df(df, x=df.index, y=df.value, title='Monthly anti-diabetic drug sales in Australia from 1992 to 2008.')
```

Output:



Result:

Thus the program to to perform time series analysis and apply the various visualization techniques is executed successfully.

Ex 6.	**Perform Data Analysis and representation on a Map using various Map data sets with**

DATE:	**Mouse Rollover effect, user interaction, etc..**

Aim:

To Perform Data Analysis and representation on a Map usingvarious Map data sets with Mouse Rollover effect, user interaction.

Algorithm:

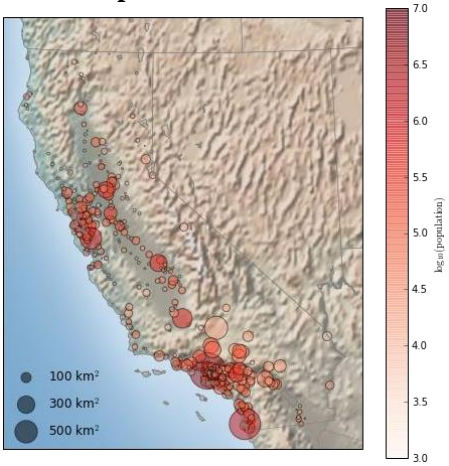
1. Import the necessary libraries (folium, pandas)
2. Load the data set from a CSV file using pandas library
3. Create a map centered on a specific location using folium
4. Perform data analysis on the loaded data set (e.g. group by and count the datapoints in each location)
5. Iterate through the location data and add markers to the map for each location
6. Iterate through the location data and add a mouse rollover effect to displayadditional information when the user hovers over the marker.
7. Add a button to the map to allow the user to toggle the visibility of the markers.
8. Display the map using folium.

Program:

```
Import folium

import pandas as

pd


# load your data set

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

# Create a map centered on a specific location
m = folium.Map(location=[45.523, -122.675], zoom_start=13)

# Perform some analysis on your data
# e.g. group by and count the data points in each location

location_data = data.groupby(['lat','lon']).count()

# Add markers to the map for each

locationfor i in
```

```
range(0,len(location_data)):
    folium.Marker(
        location= [location_data.iloc[i].name[0],
        location_data.iloc[i].name[1]],popup=f'Count:
        {location_data.iloc[i][0]}', icon=folium.Icon(color='red')
    ).add_to(m)
# Add a mouse rollover effect to display additional
information# when the user hovers over the marker
for i in
    range(0,len(location_data)):
    folium.Marker(
        location= [location_data.iloc[i].name[0],
        location_data.iloc[i].name[1]],popup=f'Count:
        {location_data.iloc[i][0]}', icon=folium.Icon(color='red')
    ).add_child(folium.Popup("Additional Info")).add_to(m)
# Add a button to the map to allow the user to toggle the visibility
# of the marker
folium.LayerControl().add_to(m)
# Display the
mapm
```

Output:



Result:

Thus, the program to perform data analysis and representation ona map using various map data sets with mouse rollover effect, user interaction, etc.. is written and executed successfully.

Ex 7. DATE:	**Build cartographic visualization for multiple datasets involving various countries of the worldstates and districts in India etc.**

Aim:

To build cartographic visualization for multiple datasets involving various countries of the worldstates and districts in India etc.

Algorithm:

1.Collecting and cleaning the data: This would likely involve using libraries such as Pandas to import and manipulate the data, and performing tasks such as removing missing values and ensuring that the data is in the correct format.

2.Mapping the data: Once the data is cleaned, you would need to use a library such as Folium or Plotly to create the actual map and overlay the data on top of it. You may also need to use shapefiles to define the boundaries of countries, states, and districts.

3.Styling the map: After the data is mapped, you would likely want to customize the appearance of the map to make it more visually appealing. This might involve using functions to change the colors of the map, add labels, and create interactive elements such as hover-over text.

4.Exporting the map: Finally, you would likely want to export the map in a format that can be easily shared or embedded on a website. This might involve using libraries such as Matplotlib or Seaborn to save the map as an image, or using libraries such as Plotly to create an interactive map that can be embedded in a webpage.

Program:

```
import folium
import pandas as pd

# Load the data
data = pd.read_csv("india_data.csv")

# Create the map
m = folium.Map(location=[20.5937, 78.9629], zoom_start=5)

# Add the data to the map
folium.Choropleth(
    geo_data="india_states_districts.json",
    name='choropleth',
    data=data,
    columns=['State', 'Value'],
    key_on='feature.properties.NAME_1',
    fill_color='YlGn',
    fill_opacity=0.7,
    line_opacity=0.2,
    legend_name='Value'
).add_to(m)

# Save the map
m.save("india_map.html")
```

Output:

Result:

Thus the given program to build cartographic visualization for multiple datasets involving various countries of the worldstates and districts in India has been executed successfully.

Ex 8. DATE:	**Perform EDA on Wine Quality Data Set**

Aim:

To Perform EDA on Wine Quality Data Set

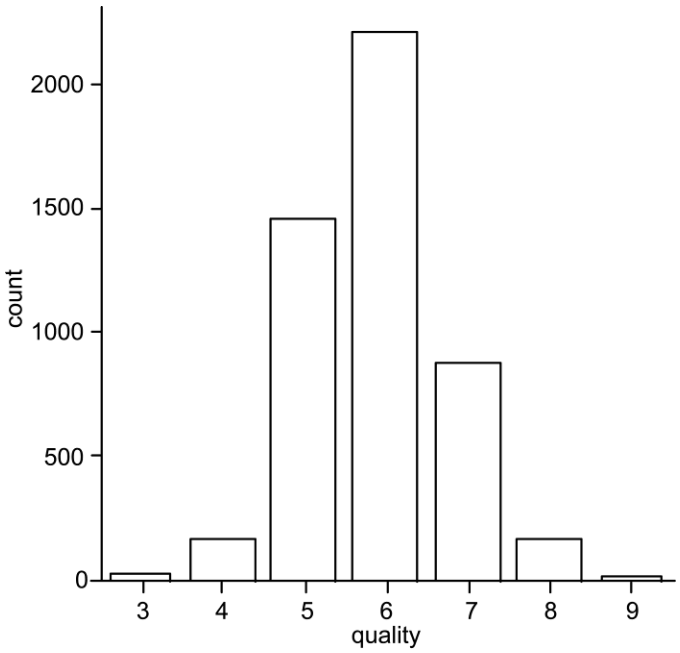
Algorithm:

1. Install the necessary packages.
2. Load the wine quality dataset.
3. Print the first 5 rows of the data.
4. Create histogram to visualize the distribution of each features.
5. Create a scatter matrix to visualize the relationship between the features.

Program:

```
import pandas as pd
import matplotlib.pyplot as plt

# load the wine quality dataset
data = pd.read_csv('winequality-red.csv', delimiter=';')

# print the first 5 rows of the data
print(data.head())

# print summary statistics of the data
print(data.describe())

# create histograms to visualize the distribution of each feature
data.hist(bins=50, figsize=(20,15))
plt.show()

# create a scatter matrix to visualize the relationship between features
pd.plotting.scatter_matrix(data, figsize=(20, 20))
plt.show()
```

Output:



Result:

Thus the given program to to Perform EDA on Wine Quality Data Set is executed successfully.

Ex 9. DATE:	**Use a case study on a data set and apply the various EDA and visualization techniques andpresent an analysis report**

Aim:

To use a case study on a data set and apply the various EDA and visualization techniques andpresent an analysis report

Algorithm:

1. Install the necessary packages.
2. Create a data set of date and price.
3. Fix the start date and end date.
4. Generate a price corresponding to the dates.
5. List the data by columns.
6. Group the columns by date.
7. Display the data .
8. Show the plot.
9. Present an analysis report.

Program:

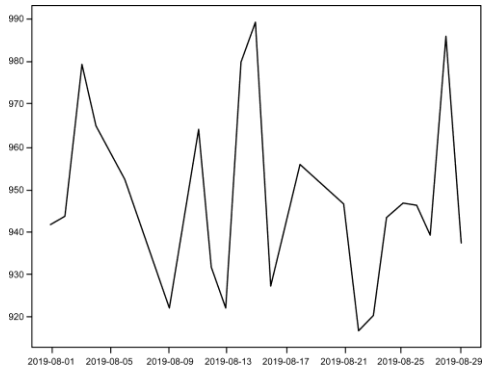
```
import datetime
import math
import pandas as pd import
random import radar
from faker import Faker fake =
Faker()
def generateData(n): listdata
    = []
    start = datetime.datetime(2019, 8, 1)
    end = datetime.datetime(2019, 8, 30) delta =
    end - start
    for _ in range(n):
        date = radar.random_datetime(start='2019-08-1', stop='2019-08-
30').strftime("%Y-%m-%d")
        price = round(random.uniform(900, 1000), 4)
```

Date	Price
2019-08-01	999.598900
2019-08-02	957.870150
2019-08-04	978.674200
2019-08-05	963.380375
2019-08-06	978.092900
2019-08-07	987.847700
2019-08-08	952.669900
2019-08-10	973.929400

```
listdata.append([date, price])
df = pd.DataFrame(listdata, columns = ['Date', 'Price']) df['Date']
= pd.to_datetime(df['Date'], format='%Y-%m-%d') df =
df.groupby(by='Date').mean()
import matplotlib.pyplot as plt
plt.rcParams['figure.figsize']          =(14,10)
plt.plot(df)
```

And the plotted graph looks something like this:

Output:



Result:

Thus the given program to apply the various EDA and visualization techniques and presenting an analysis report has been executed successfully.