1. Write a Pandas program to select distinct department id from employees file.

AIM:

To write a Pandas program to select distinct department id from employees file.

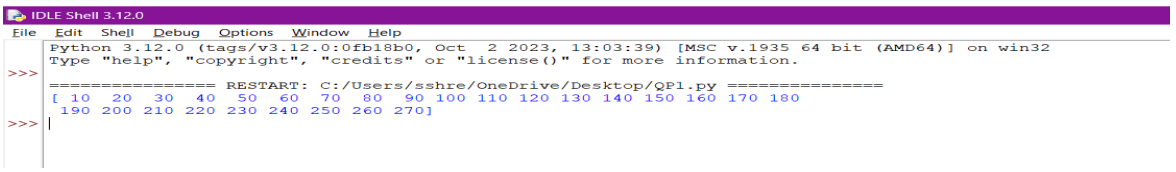
ALGORITHM:

Step 1: Import necessary libraries
Step 2: Create a dictionary to store data
Step 3: Create a Pandas DataFrame
Step 4: Get distinct department IDs
Step 5: Print the distinct department IDs

CODE:

```
import pandas as pd
data = {
   'DEPARTMENT_ID': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160, 170, 180, 190, 200, 210, 220, 230, 240, 250, 260, 270],
    'DEPARTMENT_NAME': ['Administration', 'Marketing', 'Purchasing', 'Human Resources', 'Shipping', 'IT', 'Public Relations', 'Sales', 'Executive', 'Finance', 'Accounting', 'Treasury', 'Corporate Tax', 'Control And Credit', 'Shareholder Services', 'Benefits', 'Manufacturing', 'Construction', 'Contracting', 'Operations', 'IT Support', 'NOC', 'IT Helpdesk', 'Government Sales', 'Retail Sales', 'Recruiting', 'Payroll'],
   'MANAGER_ID': [200, 201, 114, 203, 121, 103, 204, 145, 100, 108, 205, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'LOCATION_ID': [1700, 1800, 1700, 2400, 1500, 1400, 2700, 2500, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700]
}
df = pd.DataFrame(data)
distinct_department_ids = df['DEPARTMENT_ID'].unique()

print(distinct_department_ids)
```

OUTPUT:



2. Write a Pandas program to display the ID for those employees who did two or more jobs in the past.

AIM:

To write a Pandas program to display the ID for those employees who did two or more jobs in the past.

ALGORITHM:

Step 1: Import necessary libraries and create a dictionary to store data
Step 2: Create a Pandas DataFrame
Step 3: Convert date columns to datetime format
Step 4: Group by employee ID and count unique job IDs
Step 5: Filter employees with multiple jobs and print the result

CODE:

```
import pandas as pd
data = {
   'EMPLOYEE_ID': [102, 101, 101, 201, 114, 122, 200, 176, 176, 200],
      'START_DATE': ['2001-01-13', '1997-09-21', '2001-10-28', '2004-02-17', '2006-03-24', '2007-01-01', '1995-09-17', '2006-03-24', '2007-01-01', '2002-07-01'],
       'END_DATE': ['2006-07-24', '2001-10-27', '2005-03-15', '2007-12-19', '2007-12-31', '2007-12-31', '2001-06-17', '2006-12-31', '2007-12-31', '2006-12-31'],
         'JOB_ID': ['IT_PROG', 'AC_ACCOUNT', 'AC_MGR', 'MK_REP', 'ST_CLERK', 'ST_CLERK', 'AD_ASST', 'SA_REP', 'SA_MAN', 'AC_ACCOUNT'],
   'DEPARTMENT_ID': [60, 110, 110, 20, 50, 50, 90, 80, 80, 90]
}
df = pd.DataFrame(data)
df['START_DATE'] = pd.to_datetime(df['START_DATE'])
df['END_DATE'] = pd.to_datetime(df['END_DATE'])
employee_jobs = df.groupby('EMPLOYEE_ID')['JOB_ID'].nunique()
employees_with_multiple_jobs = employee_jobs[employee_jobs >= 2]
print("Employee IDs who did two or more jobs in the past:")
print(employees_with_multiple_jobs.index.tolist())
```

OUTPUT:

```
Employee IDs who did two or more jobs in the past:
[101, 176, 200]
>>>
```

3. Write a Pandas program to display the details of jobs in descending sequence on job title.

AIM:

To write a Pandas program to display the details of jobs in descending sequence on job title.

ALGORITHM:

Step 1: Import necessary libraries and create a dictionary to store data

Step 2: Create a Pandas DataFrame

Step 3: Sort the DataFrame by job title in descending order

Step 4: Print the header message

Step 5: Print the sorted DataFrame

CODE:

```
import pandas as pd

data = {

    'JOB_ID': ['AD_PRES', 'AD_VP', 'AD_ASST', 'FI_MGR', 'FI_ACCOUNT', 'AC_MGR', 'AC_ACCOUNT', 'SA_MAN', 'SA_REP', 'PU_MAN', 'PU_CLERK', 'ST_MAN', 'ST_CLERK', 'SH_CLERK', 'IT_PROG', 'MK_MAN', 'MK_REP', 'HR_REP', 'PR_REP'],

    'JOB_TITLE': ['President', 'Administration Vice President', 'Administration Assistant', 'Finance Manager', 'Accountant', 'Accounting Manager', 'Public Accountant', 'Sales Manager', 'Sales Representative', 'Purchasing Manager', 'Purchasing Clerk', 'Stock Manager', 'Stock Clerk', 'Shipping Clerk', 'Programmer', 'Marketing Manager', 'Marketing Representative', 'Human Resources Representative', 'Public Relations Representative'],

    'MIN_SALARY': [20080, 15000, 3000, 8200, 4200, 8200, 4200, 10000, 6000, 8000, 2500, 5500, 2008, 2500, 4000, 9000, 4000, 4000, 4500],

    'MAX_SALARY': [40000, 30000, 6000, 16000, 9000, 16000, 9000, 20080, 12008, 15000, 5500, 8500, 5000, 5500, 10000, 15000, 9000, 9000, 10500]

}

df = pd.DataFrame(data)

sorted_df = df.sort_values(by='JOB_TITLE', ascending=False)

print("Details of jobs in descending sequence on job title:")

print(sorted_df)
```

OUTPUT:

```
Details of jobs in descending sequence on job title:
        JOB_ID                      JOB_TITLE  MIN_SALARY  MAX_SALARY
11      ST_MAN                    Stock Manager        5500        8500
12    ST_CLERK                      Stock Clerk        2008        5000
13    SH_CLERK                   Shipping Clerk        2500        5500
8       SA_REP             Sales Representative        6000       12008
7       SA_MAN                    Sales Manager       10000       20080
9       PU_MAN               Purchasing Manager        8000       15000
10    PU_CLERK                 Purchasing Clerk        2500        5500
18      PR_REP  Public Relations Representative        4500       10500
6   AC_ACCOUNT                Public Accountant        4200        9000
14     IT_PROG                       Programmer        4000       10000
0      AD_PRES                        President       20080       40000
16      MK_REP         Marketing Representative        4000        9000
15      MK_MAN                Marketing Manager        9000       15000
17      HR_REP   Human Resources Representative        4000        9000
3       FI_MGR                  Finance Manager        8200       16000
1        AD_VP    Administration Vice President       15000       30000
2      AD_ASST         Administration Assistant        3000        6000
5       AC_MGR               Accounting Manager        8200       16000
4   FI_ACCOUNT                       Accountant        4200        9000
```

4. Write a Pandas program to create a line plot of the historical stock prices of Alphabet Inc. between two specific dates.

AIM: To write a Pandas program to create a line plot of the historical stock prices of Alphabet Inc. between two specific dates.

ALGORITHM:

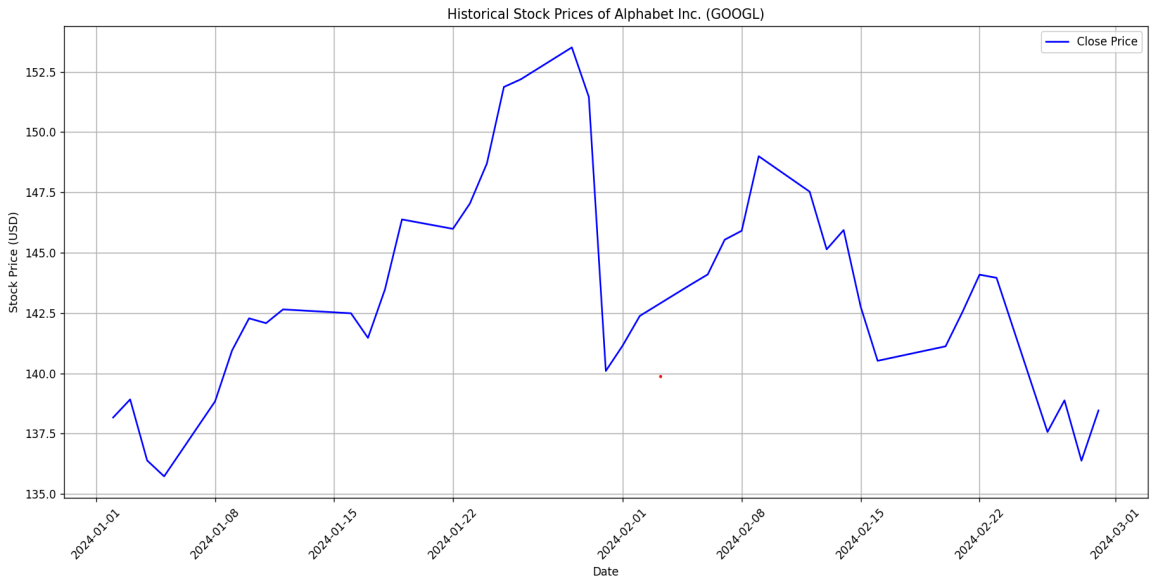
Step 1: Download Historical Stock Data
Step 2: Preprocess Data and Create Input Sequences
Step 3: Split Data into Training and Testing Sets
Step 4: Train Linear Regression Model
Step 5: Evaluate Model on Test Data

CODE:

```
import yfinance as yf
import pandas as pd
import matplotlib.pyplot as plt
ticker_symbol = 'GOOGL'
start_date = '2024-01-01'
end_date = '2024-03-01'
data = yf.download(ticker_symbol, start=start_date, end=end_date)

plt.figure(figsize=(10, 6))
plt.plot(data['Close'], label='Close Price', color='blue')
plt.title('Historical Stock Prices of Alphabet Inc. (GOOGL)')
```

```
plt.xlabel('Date')
plt.ylabel('Stock Price (USD)')
plt.xticks(rotation=45)
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

OUTPUT:



5. Write a Pandas program to create a bar plot of the trading volume of Alphabet Inc. stock between two specific dates.

AIM:

To write a Pandas program to create a bar plot of the trading volume of Alphabet Inc. stock between two specific dates.

ALGORITHM:

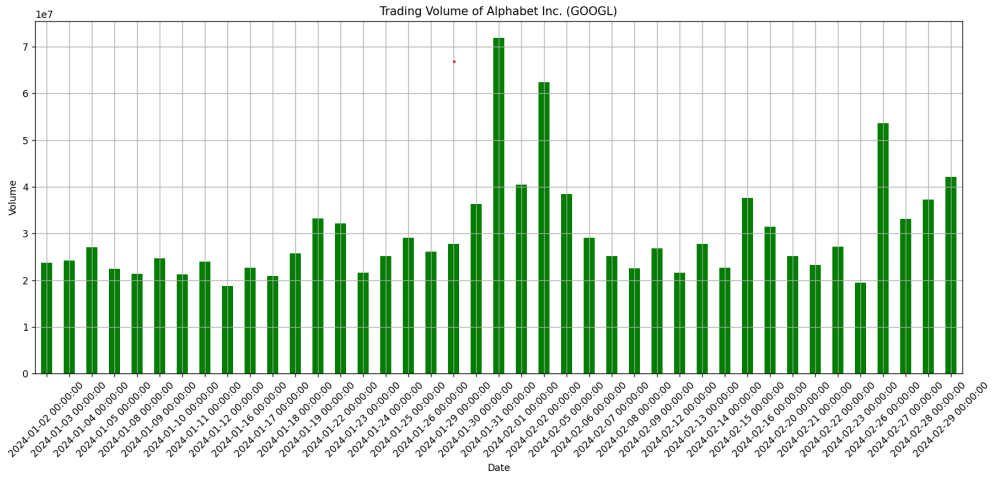
Step 1: Download Historical Stock Data
Step 2: Extract Trading Volume Data
Step 3: Create Bar Chart of Trading Volume
Step 4: Customize Chart Appearance
Step 5: Display Chart

CODE:

```
import yfinance as yf
import pandas as pd
```

```
import matplotlib.pyplot as plt
ticker_symbol = 'GOOGL'
start_date = '2024-01-01'
end_date = '2024-03-01'
data = yf.download(ticker_symbol, start=start_date, end=end_date)

plt.figure(figsize=(10, 6))
data['Volume'].plot(kind='bar', color='green')
plt.title('Trading Volume of Alphabet Inc. (GOOGL)')
plt.xlabel('Date')
plt.ylabel('Volume')
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```

OUTPUT:



6. Write a Pandas program to create a scatter plot of the trading volume/stock prices of Alphabet Inc. stock between two specific dates.
   alphabet_stock_data:

AIM:

To write a Pandas program to create a scatter plot of the trading volume/stock prices of Alphabet Inc. stock between two specific dates.

ALGORITHM:

Step 1: Download historical data

Step 2: Prepare Data for Visualization

Step 3: Create Scatter Plot

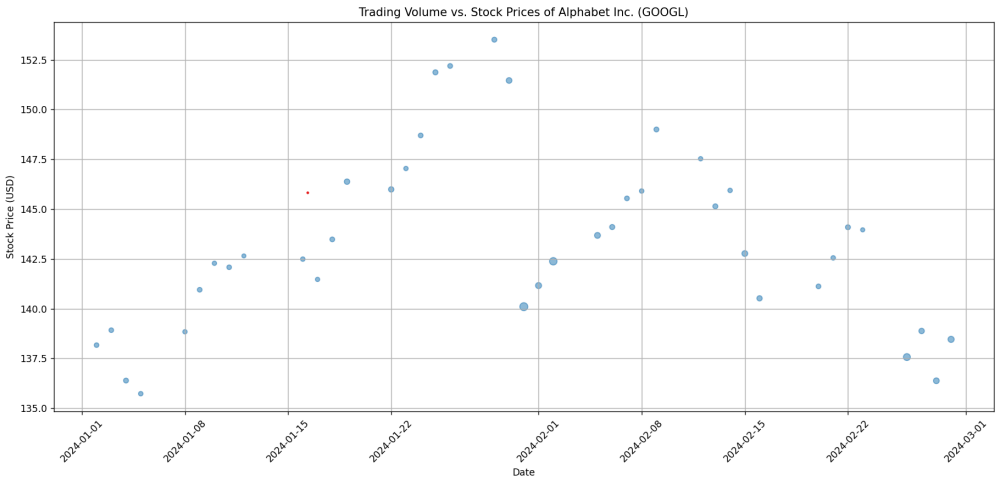
Step 4: Customize Chart Appearance

Step 5: Display Chart

CODE:

```
import yfinance as yf
import pandas as pd
import matplotlib.pyplot as plt
ticker_symbol = 'GOOGL'
start_date = '2024-01-01'
end_date = '2024-03-01'
data = yf.download(ticker_symbol, start=start_date, end=end_date)
plt.figure(figsize=(10, 6))
plt.scatter(data.index, data['Close'], s=data['Volume']/1e6, alpha=0.5)
plt.title('Trading Volume vs. Stock Prices of Alphabet Inc. (GOOGL)')
plt.xlabel('Date')
plt.ylabel('Stock Price (USD)')
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```

OUTPUT:



7. Write a Pandas program to create a Pivot table and find the maximum and minimum sale value of the items.(refer sales_data table)

AIM:

To write a Pandas program to create a Pivot table and find the maximum and minimum sale value of the items.

ALGORITHM:

Step 1: Create a Pandas DataFrame

Step 2: Create a Pivot Table

Step 3: Rename Columns

Step 4: Print the Result

Step 5: Display the Output

CODE:

```
import pandas as pd
data = {
    'OrderDate': ['1-6-18', '1-23-18', '2-9-18', '2-26-18', '3-15-18', '4-1-18', '4-18-18', '5-5-18', '5-22-18', '6-8-18', '6-25-18', '7-12-18', '7-29-18', '8-15-18', '9-1-18', '9-18-18', '10-5-18', '10-22-18'],
    'Region': ['East', 'Central', 'Central', 'Central', 'West', 'East', 'Central', 'Central', 'West', 'East', 'Central', 'East', 'East', 'East', 'Central', 'East', 'Central', 'East'],
    'Manager': ['Martha', 'Hermann', 'Hermann', 'Timothy', 'Timothy', 'Martha', 'Martha', 'Hermann', 'Douglas', 'Martha', 'Hermann', 'Martha', 'Douglas', 'Martha', 'Douglas', 'Martha', 'Hermann', 'Martha'],
```

```
    'SalesMan': ['Alexander', 'Shelli', 'Luis', 'David', 'Stephen', 'Alexander', 'Steven', 'Luis', 'Michael', 'Alexander', 'Sigal', 'Diana', 'Karen', 'Alexander', 'John', 'Alexander', 'Sigal', 'Alexander'],
    'Item': ['Television', 'Home Theater', 'Television', 'Cell Phone', 'Television', 'Home Theater', 'Television', 'Television', 'Television', 'Home Theater', 'Television', 'Home Theater', 'Home Theater', 'Television', 'Desk', 'Video Games', 'Home Theater', 'Cell Phone'],
    'Units': [95, 50, 36, 27, 56, 60, 75, 90, 32, 60, 90, 29, 81, 35, 2, 16, 28, 64],
    'Unit_price': [1198.00, 500.00, 1198.00, 225.00, 1198.00, 500.00, 1198.00, 1198.00, 1198.00, 500.00, 1198.00, 500.00, 500.00, 1198.00, 125.00, 58.50, 500.00, 225.00],
    'Sale_amt': [113810.00, 25000.00, 43128.00, 6075.00, 67088.00, 30000.00, 89850.00, 107820.00, 38336.00, 30000.00, 107820.00, 14500.00, 40500.00, 41930.00, 250.00, 936.00, 14000.00, 14400.00]
}

df = pd.DataFrame(data)
pivot_table = pd.pivot_table(df, values='Sale_amt', index='Item', aggfunc={'Sale_amt': ['max', 'min']})
pivot_table.columns = ['Max Sale', 'Min Sale']

print("Maximum and Minimum sale value of the items:")
print(pivot_table)
```

OUTPUT:

```
Maximum and Minimum sale value of the items:
              Max Sale  Min Sale
Item
Cell Phone     14400.0    6075.0
Desk             250.0     250.0
Home Theater   40500.0   14000.0
Television    113810.0   38336.0
Video Games      936.0     936.0
```

8. Write a Pandas program to create a Pivot table and find the item wise unit sold. .(refer sales_data table)

AIM:

To write a Pandas program to create a Pivot table and find the item wise unit sold.

ALGORITHM:

Step 1: Create a Pandas DataFrame
Step 2: Create a Pivot Table
Step 3: Specify the Aggregation Function
Step 4: Print the Result
Step 5: Display the Output

CODE:

```
import pandas as pd
data = {
    'OrderDate': ['1-6-18', '1-23-18', '2-9-18', '2-26-18', '3-15-18', '4-1-18', '4-18-18', '5-5-18', '5-22-18', '6-8-18', '6-25-18', '7-12-18', '7-29-18', '8-15-18', '9-1-18', '9-18-18', '10-5-18', '10-22-18'],
    'Region': ['East', 'Central', 'Central', 'Central', 'West', 'East', 'Central', 'Central', 'West', 'East', 'Central', 'East', 'East', 'East', 'Central', 'East', 'Central', 'East'],
    'Manager': ['Martha', 'Hermann', 'Hermann', 'Timothy', 'Timothy', 'Martha', 'Martha', 'Hermann', 'Douglas', 'Martha', 'Hermann', 'Martha', 'Douglas', 'Martha', 'Douglas', 'Martha', 'Hermann', 'Martha'],
    'SalesMan': ['Alexander', 'Shelli', 'Luis', 'David', 'Stephen', 'Alexander', 'Steven', 'Luis', 'Michael', 'Alexander', 'Sigal', 'Diana', 'Karen', 'Alexander', 'John', 'Alexander', 'Sigal', 'Alexander'],
    'Item': ['Television', 'Home Theater', 'Television', 'Cell Phone', 'Television', 'Home Theater', 'Television', 'Television', 'Television', 'Home Theater', 'Television', 'Home Theater', 'Home Theater', 'Television', 'Desk', 'Video Games', 'Home Theater', 'Cell Phone'],
    'Units': [95, 50, 36, 27, 56, 60, 75, 90, 32, 60, 90, 29, 81, 35, 2, 16, 28, 64],
    'Unit_price': [1198.00, 500.00, 1198.00, 225.00, 1198.00, 500.00, 1198.00, 1198.00, 1198.00, 500.00, 1198.00, 500.00, 500.00, 1198.00, 125.00, 58.50, 500.00, 225.00],
    'Sale_amt': [113810.00, 25000.00, 43128.00, 6075.00, 67088.00, 30000.00, 89850.00, 107820.00, 38336.00, 30000.00, 107820.00, 14500.00, 40500.00, 41930.00, 250.00, 936.00, 14000.00, 14400.00]
}
df = pd.DataFrame(data)

pivot_table = pd.pivot_table(df, values='Units', index='Item', aggfunc='sum')

print("Item-wise units sold:")
print(pivot_table)
```

OUTPUT:

```
Item-wise units sold:
              Units
Item
Cell Phone       91
Desk              2
Home Theater    308
Television      509
Video Games      16
```

9. Write a Pandas program to create a Pivot table and find the total sale amount region wise, manager wise, sales man wise. .(refer sales_data table)

AIM:

To write a Pandas program to create a Pivot table and find the total sale amount region wise, manager wise, sales man wise.

ALGORITHM:

Step 1: Create a Pandas DataFrame

Step 2: Create a Pivot Table

Step 3: Specify the Aggregation Function

Step 4: Print the Result

Step 5: Display the Output

CODE:

import pandas as pd

data = {

'OrderDate': ['1-6-18', '1-23-18', '2-9-18', '2-26-18', '3-15-18', '4-1-18', '4-18-18', '5-5-18', '5-22-18', '6-8-18', '6-25-18', '7-12-18', '7-29-18', '8-15-18', '9-1-18', '9-18-18', '10-5-18', '10-22-18'],

'Region': ['East', 'Central', 'Central', 'Central', 'West', 'East', 'Central', 'Central', 'West', 'East', 'Central', 'East', 'East', 'East', 'Central', 'East', 'Central', 'East'],

'Manager': ['Martha', 'Hermann', 'Hermann', 'Timothy', 'Timothy', 'Martha', 'Martha', 'Hermann', 'Douglas', 'Martha', 'Hermann', 'Martha', 'Douglas', 'Martha', 'Douglas', 'Martha', 'Hermann', 'Martha'],

'SalesMan': ['Alexander', 'Shelli', 'Luis', 'David', 'Stephen', 'Alexander', 'Steven', 'Luis', 'Michael', 'Alexander', 'Sigal', 'Diana', 'Karen', 'Alexander', 'John', 'Alexander', 'Sigal', 'Alexander'],

'Item': ['Television', 'Home Theater', 'Television', 'Cell Phone', 'Television', 'Home Theater', 'Television', 'Television', 'Television', 'Home Theater', 'Television', 'Home Theater', 'Home Theater', 'Television', 'Desk', 'Video Games', 'Home Theater', 'Cell Phone'],

'Units': [95, 50, 36, 27, 56, 60, 75, 90, 32, 60, 90, 29, 81, 35, 2, 16, 28, 64],

'Unit_price': [1198.00, 500.00, 1198.00, 225.00, 1198.00, 500.00, 1198.00, 1198.00, 1198.00, 500.00, 1198.00, 500.00, 500.00, 1198.00, 125.00, 58.50, 500.00, 225.00],

'Sale_amt': [113810.00, 25000.00, 43128.00, 6075.00, 67088.00, 30000.00, 89850.00, 107820.00, 38336.00, 30000.00, 107820.00, 14500.00, 40500.00, 41930.00, 250.00, 936.00, 14000.00, 14400.00]

}

df = pd.DataFrame(data)

pivot_table = pd.pivot_table(df, values='Sale_amt', index=['Region', 'Manager', 'SalesMan'], aggfunc='sum')

print("Total sale amount region-wise, manager-wise, and salesperson-wise:")

print(pivot_table)

OUTPUT:

```
Total sale amount region-wise, manager-wise, and salesperson-wise:
                          Sale_amt
Region  Manager SalesMan
Central Douglas John         250.0
        Hermann Luis       150948.0
                Shelli      25000.0
                Sigal      121820.0
        Martha  Steven      89850.0
        Timothy David        6075.0
East    Douglas Karen       40500.0
        Martha  Alexander  231076.0
                Diana       14500.0
West    Douglas Michael     38336.0
        Timothy Stephen     67088.0
```

10.Create a dataframe of ten rows, four columns with random values. Write a Pandas program to highlight the negative numbers red and positive numbers black.

AIM:

To create a dataframe of ten rows,four columns with random values.

ALGORITHM:

Step 1: Import libraries

Step 2: Generate random data

Step 3: Create a DataFrame

Step 4: Style cells

Step 5: Display styled DataFrame

CODE:

```
import pandas as pd

import numpy as np

np.random.seed(0)  # for reproducibility

data = np.random.randn(10, 4)  # generating random values

df = pd.DataFrame(data, columns=['A', 'B', 'C', 'D'])

styled_df = df.style.apply(lambda x: ['color: red' if val < 0 else 'color: black' for val in x])

styled_df
```

OUTPUT:

Code cell output actions

	A	B	C	D
0	1.764052	0.400157	0.978738	2.240893
1	1.867558	-0.977278	0.950088	-0.151357
2	-0.103219	0.410599	0.144044	1.454274
3	0.761038	0.121675	0.443863	0.333674
4	1.494079	-0.205158	0.313068	-0.854096
5	-2.552990	0.653619	0.864436	-0.742165
6	2.269755	-1.454366	0.045759	-0.187184
7	1.532779	1.469359	0.154947	0.378163
8	-0.887786	-1.980796	-0.347912	0.156349
9	1.230291	1.202380	-0.387327	-0.302303

11.Create a dataframe of ten rows, four columns with random values. Convert some values to nan values. Write a Pandas program which will highlight the nan values.

AIM:

To write a Pandas program which will highlight the nan values.

ALGORITHM:

Step 1:Create the DataFrame

Step 2: Introduce NaN Values (Missing Data)

Step 3:Apply Styling

Step 4:Display the Styled DataFrame

Step 5:Save the Styled DataFrame

CODE:

```
import pandas as pd

import numpy as np

df = pd.DataFrame(np.random.randn(10, 4), columns=list('ABCD'))

# Convert some values to nan

df.iloc[0, 2] = np.nan

df.iloc[3, 3] = np.nan

df.iloc[4, 1] = np.nan

df.iloc[9, 3] = np.nan

# Highlight the nan values

# The parameter name was changed to color

df.style.highlight_null(color='red')
```

OUTPUT:

	A	B	C	D
0	0.218123	-0.194423	nan	-0.389302
1	-0.057835	0.238146	1.837028	1.179470
2	-1.850733	1.383598	-2.273248	-0.667508
3	0.207374	0.992593	0.986581	nan
4	-2.263297	nan	0.814726	-1.379800
5	-1.825014	-0.082086	-0.296026	-0.459907
6	1.273190	-0.905904	1.115115	0.698353
7	0.056983	-1.061257	0.113792	0.762073
8	-0.493053	-0.578866	-0.338860	-1.506250
9	-0.596784	0.091329	-1.412540	nan

12.Create a dataframe of ten rows, four columns with random values. Write a Pandas program to set dataframe background Color black and font color yellow.

AIM:

To write a Pandas program to set dataframe background Color black and font color yellow.

ALGORITHM:

Step 1: Import necessary libraries

Step 2: Create a sample DataFrame

Step 3: Create a styled DataFrame object

Step 4: Apply styles using set_properties method

Step 5: Render the styled DataFrame

CODE:

```
import pandas as pd
```

```
import numpy as np
np.random.seed(0)
data = np.random.randn(10, 4)
df = pd.DataFrame(data, columns=['A', 'B', 'C', 'D'])
styled_df = df.style.set_properties(**{'background-color': 'black', 'color': 'yellow'})
styled_df
```

OUTPUT:

	A	B	C	D
0	1.764052	0.400157	0.978738	2.240893
1	1.867558	-0.977278	0.950088	-0.151357
2	-0.103219	0.410599	0.144044	1.454274
3	0.761038	0.121675	0.443863	0.333674
4	1.494079	-0.205158	0.313068	-0.854096
5	-2.552990	0.653619	0.864436	-0.742165
6	2.269755	-1.454366	0.045759	-0.187184
7	1.532779	1.469359	0.154947	0.378163
8	-0.887786	-1.980796	-0.347912	0.156349
9	1.230291	1.202380	-0.387327	-0.302303

13.Write a Pandas program to detect missing values of a given DataFrame. Display True or False.

AIM:

To write a Pandas program to detect missing values of a given DataFrame.

ALGORITHM:

Step 1: Import Pandas library

Step 2: Create a sample DataFrame with missing values

Step 3: Use ‘ isna() ‘ method to detect missing values

Step 4: Print the result

Step 5: Interpret the result

CODE:

```
import pandas as pd

data = {
```

```
    'A': [1, 2, None, 4, 5],

    'B': [None, 2, 3, None, 5],

    'C': [1, 2, 3, 4, 5]

}

df = pd.DataFrame(data)

missing_values = df.isna()

print(missing_values)
```

OUTPUT:

```
       A      B      C
0  False   True  False
1  False  False  False
2   True  False  False
3  False   True  False
4  False  False  False
```

14. Write a Pandas program to find and replace the missing values in a given DataFrame which do not have any valuable information.

AIM:

To find and replace the missing values in a given DataFrame which do not have any valuable information.

ALGORITHM:

Step 1:Import necessary libraries

Step 2:Create dataframe

Step 3:Print original dataframe

Step 4:Fill missing values

Step 5:Print dataframe after framing

CODE:

```
import pandas as pd
```

import numpy as np

df = pd.DataFrame({

'ord_no': [70001, np.nan, 70002, 70004, np.nan, 70005, '-', 70010, 70003, 70012, np.nan, 70013],

'purch_amt': [150.5, 270.65, 65.26, 110.5, 948.5, 2400.6, 5760, np.nan, 12.43, 2480.4, 250.45, 3045.6],

'ord_date': [np.nan, '2012-09-10', np.nan, '2012-08-17', '2012-09-10', '2012-07-27', '2012-09-10', '2012-10-10', '2012-10-10', '2012-06-27', '2012-08-17', '2012-04-25'],

'customer_id': [3002, 3001, 3001, 3003, 3002, 3001, 3001, 3004, np.nan, 3002, 3001, 3001],

'salesman_id': [5002, 5003, np.nan, 5001, np.nan, 5002, 5001, np.nan, 5003, 5002, 5003, np.nan]

})

print("Original DataFrame:")

print(df)

df.fillna(method='ffill', inplace=True)

print("\nDataFrame after replacing missing values:")

print(df)

OUTPUT:

```
Original DataFrame:
   ord_no  purch_amt    ord_date  customer_id  salesman_id
0   70001     150.50         NaN       3002.0       5002.0
1     NaN     270.65  2012-09-10       3001.0       5003.0
2   70002      65.26         NaN       3001.0          NaN
3   70004     110.50  2012-08-17       3003.0       5001.0
4     NaN     948.50  2012-09-10       3002.0          NaN
5   70005    2400.60  2012-07-27       3001.0       5002.0
6       -    5760.00  2012-09-10       3001.0       5001.0
7   70010        NaN  2012-10-10       3004.0          NaN
8   70003      12.43  2012-10-10          NaN       5003.0
9   70012    2480.40  2012-06-27       3002.0       5002.0
10    NaN     250.45  2012-08-17       3001.0       5003.0
11  70013    3045.60  2012-04-25       3001.0          NaN

DataFrame after replacing missing values:
   ord_no  purch_amt    ord_date  customer_id  salesman_id
0   70001     150.50         NaN       3002.0       5002.0
1   70001     270.65  2012-09-10       3001.0       5003.0
2   70002      65.26  2012-09-10       3001.0       5003.0
3   70004     110.50  2012-08-17       3003.0       5001.0
4   70004     948.50  2012-09-10       3002.0       5001.0
5   70005    2400.60  2012-07-27       3001.0       5002.0
6       -    5760.00  2012-09-10       3001.0       5001.0
7   70010    5760.00  2012-10-10       3004.0       5001.0
8   70003      12.43  2012-10-10       3004.0       5003.0
9   70012    2480.40  2012-06-27       3002.0       5002.0
10  70012     250.45  2012-08-17       3001.0       5003.0
11  70013    3045.60  2012-04-25       3001.0       5003.0
```

15.Write a Pandas program to keep the rows with at least 2 NaN values in a given DataFrame.

AIM:

To keep the rows with at least 2 NaN values in a given DataFrame.

ALGORITHM:

Step 1: Create a dataframe
Step 2: Count NaN values
Step 3:Filter rows
Step 4:Assign result
Step 5:Display result

CODE:

```
import pandas as pd
import numpy as np
data = {
   'A': [1, 2, None, 4, 5],
   'B': [None, None, None, None, None],
   'C': [1, 2, 3, None, 5]
}
df = pd.DataFrame(data)
df_with_nan = df.dropna(thresh=2)
print(df_with_nan)
```

OUTPUT:

```
     A     B    C
0  1.0  None  1.0
1  2.0  None  2.0
4  5.0  None  5.0
```

16.Write a Pandas program to split the following dataframe into groups based on school code. Also check the type of GroupBy object.

AIM:

To write a Pandas program to split the following dataframe into groups based on school code.

ALGORITHM:

Step 1: Create a DataFrame

Step 2: Group by school code

Step 3: Split data into groups

Step 4:Print each group

Step 5:Verify groupby object type

CODE:

```
import pandas as pd

student_data = pd.DataFrame({

    'school_code': ['s001', 's002', 's003', 's001', 's002', 's004'],

    'class': ['V', 'V', 'VI', 'VI', 'V', 'VI'],

    'name': ['Alberto Franco', 'Gino Mcneill', 'Ryan Parkes', 'Eesha Hinton', 'Gino Mcneill', 'David Parkes'],

    'date_Of_Birth': ['15/05/2002', '17/05/2002', '16/02/1999', '25/09/1998', '11/05/2002', '15/09/1997'],

    'age': [12, 12, 13, 13, 14, 12],

    'height': [173, 192, 186, 167, 151, 159],

    'weight': [35, 32, 33, 30, 31, 32],

    'address': ['street1', 'street2', 'street3', 'street1', 'street2', 'street4']

}, index=['S1', 'S2', 'S3', 'S4', 'S5', 'S6'])

print("Original DataFrame:")

print(student_data)

result = student_data.groupby(['school_code'])

print("\nSplit the data on school_code wise:")

for name, group in result:

    print("\nGroup:")

    print(name)

    print(group)

print("\nType of the object:")

print(type(result))
```

```
import pandas as pd

student_data = pd.DataFrame({

    'school_code': ['s001', 's002', 's003', 's001', 's002', 's004'],

    'class': ['V', 'V', 'VI', 'VI', 'V', 'VI'],

      'name': ['Alberto Franco', 'Gino Mcneill', 'Ryan Parkes', 'Eesha Hinton', 'Gino Mcneill',
'David Parkes'],

        'date_Of_Birth': ['15/05/2002', '17/05/2002', '16/02/1999', '25/09/1998', '11/05/2002',
'15/09/1997'],

    'age': [12, 12, 13, 13, 14, 12],

    'height': [173, 192, 186, 167, 151, 159],

    'weight': [35, 32, 33, 30, 31, 32],

    'address': ['street1', 'street2', 'street3', 'street1', 'street2', 'street4']

}, index=['S1', 'S2', 'S3', 'S4', 'S5', 'S6'])

print("Original DataFrame:")

print(student_data)

result = student_data.groupby(['school_code'])

print("\nSplit the data on school_code wise:")

for name, group in result:

    print("\nGroup:")

    print(name)

    print(group)

print("\nType of the object:")

print(type(result))
```

OUTPUT:

```
Split the data on school_code wise:

Group:
('s001',)
   school_code class            name date_Of_Birth  age  height  weight  \
S1        s001     V  Alberto Franco    15/05/2002   12     173      35
S4        s001    VI    Eesha Hinton    25/09/1998   13     167      30

    address
S1  street1
S4  street1

Group:
('s002',)
   school_code class          name date_Of_Birth  age  height  weight  address
S2        s002     V  Gino Mcneill    17/05/2002   12     192      32  street2
S5        s002     V  Gino Mcneill    11/05/2002   14     151      31  street2

Group:
('s003',)
   school_code class         name date_Of_Birth  age  height  weight  address
S3        s003    VI  Ryan Parkes    16/02/1999   13     186      33  street3

Group:
('s004',)
   school_code class          name date_Of_Birth  age  height  weight  address
S6        s004    VI  David Parkes    15/09/1997   12     159      32  street4

Type of the object:
<class 'pandas.core.groupby.generic.DataFrameGroupBy'>
```

17.Write a Pandas program to split the following dataframe by school code and get mean, min, and max value of age for each school.

AIM:

To write a Pandas program to split the following dataframe by school code and get mean, min, and max value of age for each school.

ALGORITHM:

Step 1:Create a DataFrame

Step 2: Group by school code

Step 3:Select age column

Step 4:Calculate mean, min, and max age

Step 5:Print result

CODE:

import pandas as pd

```
student_data = pd.DataFrame({
    'school_code': ['s001', 's002', 's003', 's001', 's002', 's004'],
    'class': ['V', 'V', 'VI', 'VI', 'V', 'VI'],
    'name': ['Alberto Franco', 'Gino Mcneill', 'Ryan Parkes', 'Eesha Hinton', 'Gino Mcneill', 'David Parkes'],
    'date_Of_Birth': ['15/05/2002', '17/05/2002', '16/02/1999', '25/09/1998', '11/05/2002', '15/09/1997'],
    'age': [12, 12, 13, 13, 14, 12],
    'height': [173, 192, 186, 167, 151, 159],
    'weight': [35, 32, 33, 30, 31, 32],
    'address': ['street1', 'street2', 'street3', 'street1', 'street2', 'street4']
}, index=['S1', 'S2', 'S3', 'S4', 'S5', 'S6'])

print("Original DataFrame:")
print(student_data)

# Group the data by school code and calculate mean, min, and max age
grouped_single = student_data.groupby('school_code').agg({'age': ['mean', 'min', 'max']})

print("\nMean, min, and max value of age for each school:")
print(grouped_single)
```

OUTPUT:

```
Mean, min, and max value of age for each school:
              age
             mean min max
school_code
s001         12.5  12  13
s002         13.0  12  14
s003         13.0  13  13
s004         12.0  12  12
```

18.Write a Pandas program to split the following given data frame into groups based on school code and class.

AIM:

To split the following given data frame into groups based on school code and class.

ALGORITHM:

Step 1:Create a DataFrame

Step 2: Group by school code and class

Step 3:Split data into groups

Step 4:Print each group

Step 5:Verify group details

CODE:

```
import pandas as pd

student_data = pd.DataFrame({

    'school_code': ['s001', 's002', 's003', 's001', 's002', 's004'],

    'class': ['V', 'V', 'VI', 'VI', 'V', 'VI'],

    'name': ['Alberto Franco', 'Gino Mcneill', 'Ryan Parkes', 'Eesha Hinton', 'Gino Mcneill', 'David Parkes'],

    'date_Of_Birth': ['15/05/2002', '17/05/2002', '16/02/1999', '25/09/1998', '11/05/2002', '15/09/1997'],

    'age': [12, 12, 13, 13, 14, 12],

    'height': [173, 192, 186, 167, 151, 159],
```

```
    'weight': [35, 32, 33, 30, 31, 32],
    'address': ['street1', 'street2', 'street3', 'street1', 'street2', 'street4']
}, index=['S1', 'S2', 'S3', 'S4', 'S5', 'S6'])
print("Original DataFrame:")
print(student_data)
result = student_data.groupby(['school_code', 'class'])
print("\nSplit the data on school_code and class wise:")
for name, group in result:
    print("\nGroup:")
    print(name)
    print(group)
```

OUTPUT:

```
Split the data on school_code and class wise:

Group:
('s001', 'V')
   school_code class            name date_Of_Birth  age  height  weight  \
S1        s001     V  Alberto Franco    15/05/2002   12     173      35

    address
S1  street1

Group:
('s001', 'VI')
   school_code class          name date_Of_Birth  age  height  weight  address
S4        s001    VI  Eesha Hinton    25/09/1998   13     167      30  street1

Group:
('s002', 'V')
   school_code class          name date_Of_Birth  age  height  weight  address
S2        s002     V  Gino Mcneill    17/05/2002   12     192      32  street2
S5        s002     V  Gino Mcneill    11/05/2002   14     151      31  street2

Group:
('s003', 'VI')
   school_code class         name date_Of_Birth  age  height  weight  address
S3        s003    VI  Ryan Parkes    16/02/1999   13     186      33  street3

Group:
('s004', 'VI')
   school_code class          name date_Of_Birth  age  height  weight  address
S6        s004    VI  David Parkes    15/09/1997   12     159      32  street4
```

19.Write a Pandas program to display the dimensions or shape of the World alcohol consumption dataset. Also extract the column names from the dataset.

AIM:

Write a Pandas program to display the dimensions or shape of the World alcohol consumption dataset and also extract the column names from the dataset.

ALGORITHM:

Step 1: Understand the Data
Step 2:Define Your Objective
Step 3:Import the Necessary Library
Step 4:Create a Pandas DataFrame
Step 5:Perform Data Analysis and Operations

CODE:

```
import pandas as pd
data = {'Year': [1986, 1986, 1985, 1986, 1987],
        'WHO region': ['Western Pacific', 'Americas', 'Africa', 'Americas', 'Americas'],
        'Country': ['Viet Nam', 'Uruguay', "Cte d'Ivoire", 'Colombia', 'Saint Kitts and Nevis'],
        'Beverage Types': ['Wine', 'Other', 'Wine', 'Beer', 'Beer'],
        'Display Value': [0.00, 0.50, 1.62, 4.27, 1.98]}
df = pd.DataFrame(data)
print("Dimensions of the dataset:", df.shape)
print("Column names:", df.columns)
```

OUTPUT:

```
Dimensions of the dataset: (5, 5)
Column names: Index(['Year', 'WHO region', 'Country', 'Beverage Types', 'Display Value'], dtype='object')
```

20.Write a Pandas program to find the index of a given substring of a DataFrame column.

AIM:

To write a Pandas program to find the index of a given substring of a DataFrame column.

ALGORITHM:

Step 1:Set Up Your DataFrame

Step 2:Define the Substring

Step 3:Use str.contains()

Step 4:Display the Results

CODE:

```
import pandas as pd

data = {

    'Column': ['apple', 'banana', 'orange', 'grape', 'watermelon']

}

df = pd.DataFrame(data)

substring = 'an'

indices = df[df['Column'].str.contains(substring)].index

print("Indices of rows containing the substring:", indices)
```

OUTPUT:

```
Indices of rows containing the substring: Index([1, 2], dtype='int64')
```

21.Write a Pandas program to swap the cases of a specified character column in a given DataFrame.

AIM:

To write a Pandas program to swap the cases of a specified character column in a given DataFrame.

ALGORITHM:

Step 1:Importing pandas

Step 2:Creating a sample DataFrame

Step 3:Specifying the column to swap cases

Step 4:Checking if the column exists

Step 5:Swapping cases if the column exists

CODE:

```
import pandas as pd

data = {'Name': ['John', 'Alice', 'Bob', 'David'],

        'Age': [25, 30, 35, 40],

        'City': ['New York', 'Los Angeles', 'Chicago', 'Houston']}

df = pd.DataFrame(data)
```

```
print("Original DataFrame:")
print(df)
column_to_swap = 'Name'
if column_to_swap in df.columns:
    # Swap the cases of the specified column
    df[column_to_swap] = df[column_to_swap].str.swapcase()
else:
    print("Column '{}' not found in the DataFrame.".format(column_to_swap))
print("\nDataFrame after swapping cases of column '{}':".format(column_to_swap))
print(df)
```

OUTPUT:

```
Original DataFrame:
    Name  Age         City
0   John   25     New York
1  Alice   30  Los Angeles
2    Bob   35      Chicago
3  David   40      Houston

DataFrame after swapping cases of column 'Name':
    Name  Age         City
0   jOHN   25     New York
1  aLICE   30  Los Angeles
2    bOB   35      Chicago
3  dAVID   40      Houston
```

22.Write a Python program to draw a line with suitable label in the x axis, y axis and a title.

AIM:

To write a Python program to draw a line with suitable label in the x axis, y axis and a title.

ALGORITHM:

Step 1:Import the Matplotlib Library

Step 2:Prepare Data

Step 3:Plot the Data

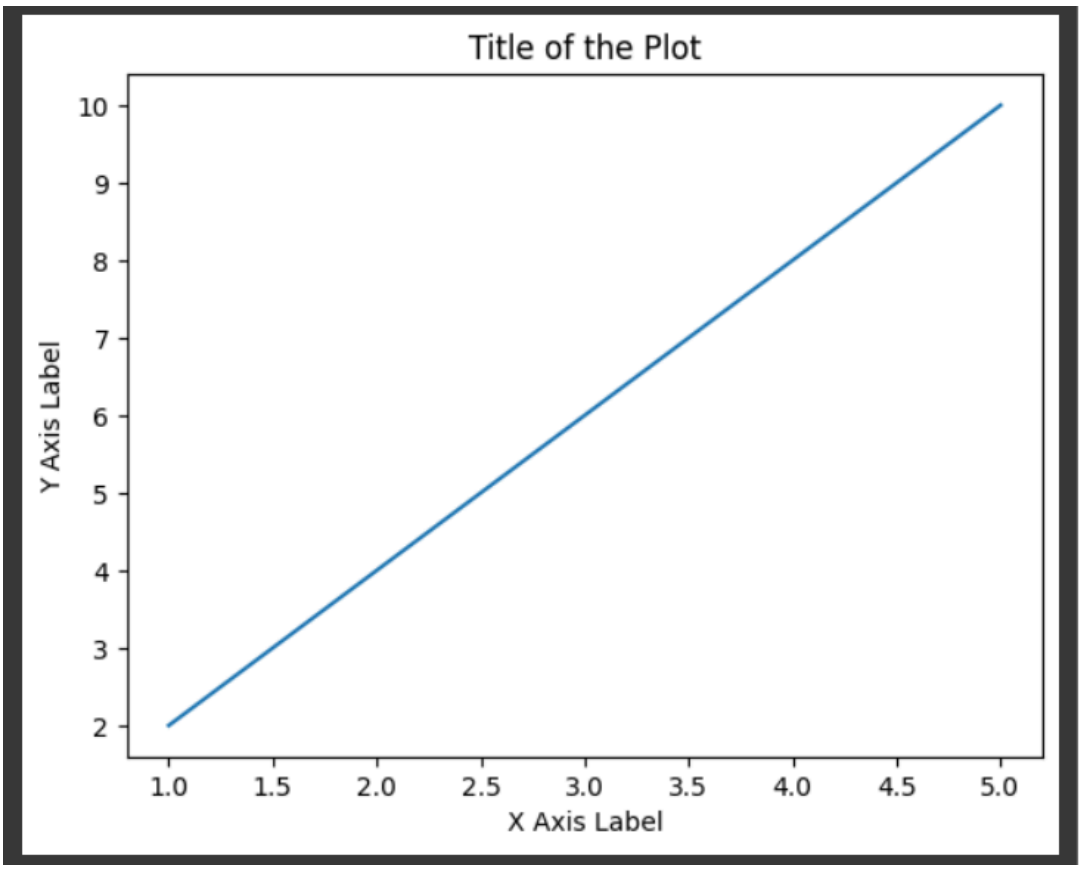
Step 4:Label the Axes and Add a Title

Step 5:Show the Plot

CODE:

```
import matplotlib.pyplot as plt

x_values = [1, 2, 3, 4, 5]

y_values = [2, 4, 6, 8, 10]

plt.plot(x_values, y_values)

plt.xlabel('X Axis Label')

plt.ylabel('Y Axis Label')

plt.title('Title of the Plot')

plt.show()
```

OUTPUT:



23.Write a Python program to draw a line using given axis values taken from a text file, with suitable label in the x axis, y axis and a title.

Test Data:

test.txt

1 2

2 4

3 1

AIM:

To write a Python program to draw a line using given axis values taken from a text file, with suitable label in the x axis, y axis and a title.

ALGORITHM:

Step 1: Import Necessary Libraries

Step 2: Read Data from Text File

Step 3: Extract X and Y Values

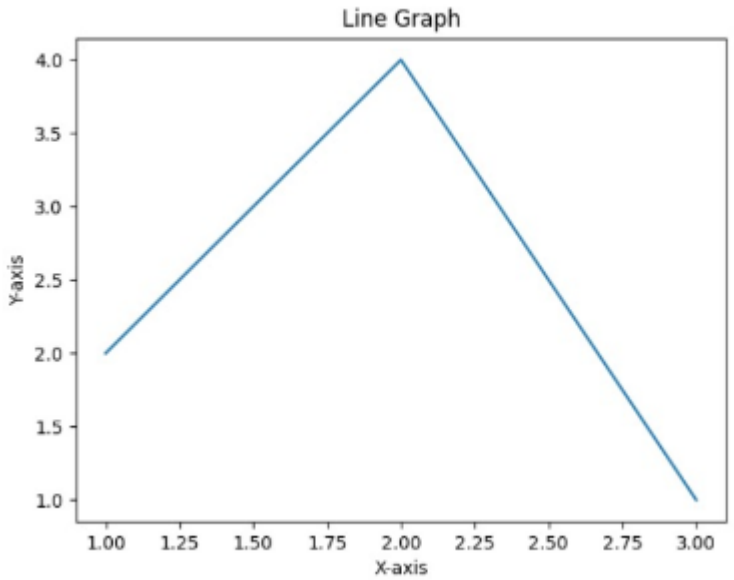
Step 4: Create the Plot

Step 5: Display the Plot

CODE:

```
import matplotlib.pyplot as plt

with open('test.txt', 'r') as file:

    data = file.readlines()

x_values = [float(line.split()[0]) for line in data]

y_values = [float(line.split()[1]) for line in data]

plt.plot(x_values, y_values)

plt.xlabel('X Axis Label')

plt.ylabel('Y Axis Label')

plt.title('Title of the Plot')

plt.show()
```

OUTPUT:



24.Write a Python program to draw line charts of the financial data of Alphabet Inc. between October 3, 2016 to October 7, 2016.
Sample Financial data (fdata.csv):
Date,Open,High,Low,Close
10-03-16,774.25,776.065002,769.5,772.559998
10-04-16,776.030029,778.710022,772.890015,776.429993
10-05-16,779.309998,782.070007,775.650024,776.469971
10-06-16,779,780.47998,775.539978,776.859985
10-07-16,779.659973,779.659973,770.75,775.080017

AIM:

To draw line charts of the financial data of Alphabet Inc. between October 3, 2016 to October 7, 2016.

ALGORITHM:

Step 1: Import Necessary Libraries and Load Data

Step 2: Convert Date Column to Datetime Format and Filter Data

Step 3: Create the Plot

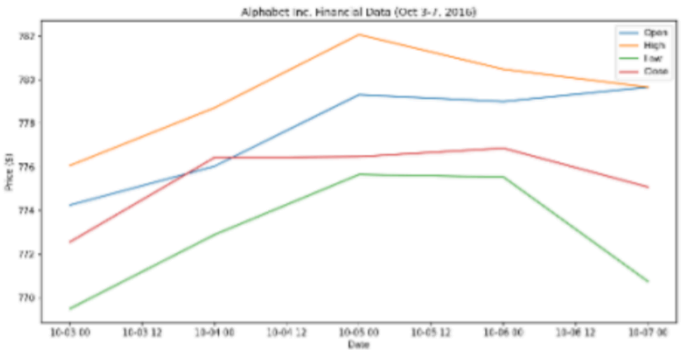
Step 4: Customize the Plot

Step 5: Display the Plot

CODE:

```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv('fdata.csv')
df['Date'] = pd.to_datetime(df['Date'])
start_date = '2016-10-03'
end_date = '2016-10-07'
filtered_data = df[(df['Date'] >= start_date) & (df['Date'] <= end_date)]
plt.figure(figsize=(10, 6))
plt.plot(filtered_data['Date'], filtered_data['Close'], marker='o', linestyle='-')
plt.xlabel('Date')
plt.ylabel('Closing Price')
plt.title('Financial Data of Alphabet Inc. (Oct 3, 2016 to Oct 7, 2016)')
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```

OUTPUT:



25.Write a Python program to plot two or more lines with legends, different widths and colors.

AIM:

To plot two or more lines with legends, different widths and colors.

ALGORITHM:

Step 1:Import libraries and define data

Step 2:Create the plots

Step 3:Add legend and labels

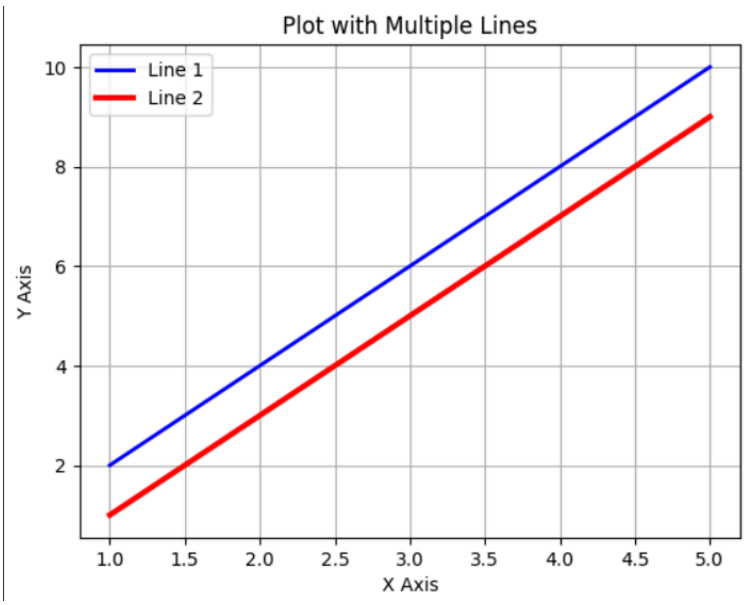
Step 4:Customize the plot

Step 5:Display the plot

CODE:

```
import matplotlib.pyplot as plt

x_values = [1, 2, 3, 4, 5]

y_values1 = [2, 4, 6, 8, 10]

y_values2 = [1, 3, 5, 7, 9]

plt.plot(x_values, y_values1, label='Line 1', color='blue', linewidth=2)

plt.plot(x_values, y_values2, label='Line 2', color='red', linewidth=3)

plt.legend()

plt.xlabel('X Axis')

plt.ylabel('Y Axis')

plt.title('Plot with Multiple Lines')

plt.grid(True)

plt.show()
```

OUTPUT:



26.Write a Python program to create multiple plots.

AIM:

To create multiple plots.

ALGORITHM:

Step 1:Import Libraries and Define Data

Step 2:Create the Figure and Subplots

Step 3:Plot the Data in Each Subplot

Step 4:Customize the Subplots

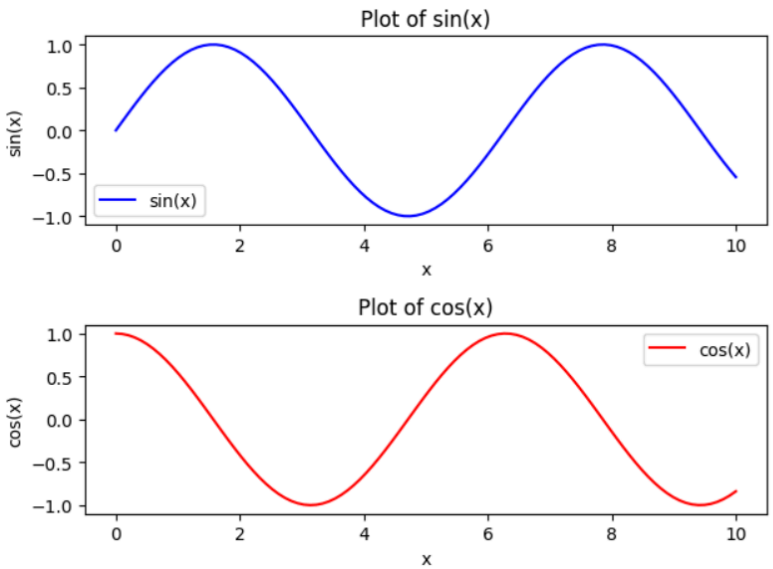
Step 5:Display the Figure

CODE:

```
import matplotlib.pyplot as plt

import numpy as np

x = np.linspace(0, 10, 100)

y1 = np.sin(x)
```

```
y2 = np.cos(x)

fig, axs = plt.subplots(2)  # 2 rows of plots

axs[0].plot(x, y1, color='blue', label='sin(x)')

axs[0].set_title('Plot of sin(x)')

axs[0].set_xlabel('x')

axs[0].set_ylabel('sin(x)')

axs[0].legend()

axs[1].plot(x, y2, color='red', label='cos(x)')

axs[1].set_title('Plot of cos(x)')

axs[1].set_xlabel('x')

axs[1].set_ylabel('cos(x)')

axs[1].legend()

plt.tight_layout()

plt.show()
```

OUTPUT:



27.Write a Python programming to display a bar chart of the popularity of programming Languages.

Sample data:

Programming languages: Java, Python, PHP, JavaScript, C#, C++
Popularity: 22.2, 17.6, 8.8, 8, 7.7, 6.7

AIM:

To display a bar chart of the popularity of programming Languages.

ALGORITHM:

Step 1:Import Libraries and Define Data

Step 2:Create the Figure and Bar Chart

Step 3:Add Labels and Title

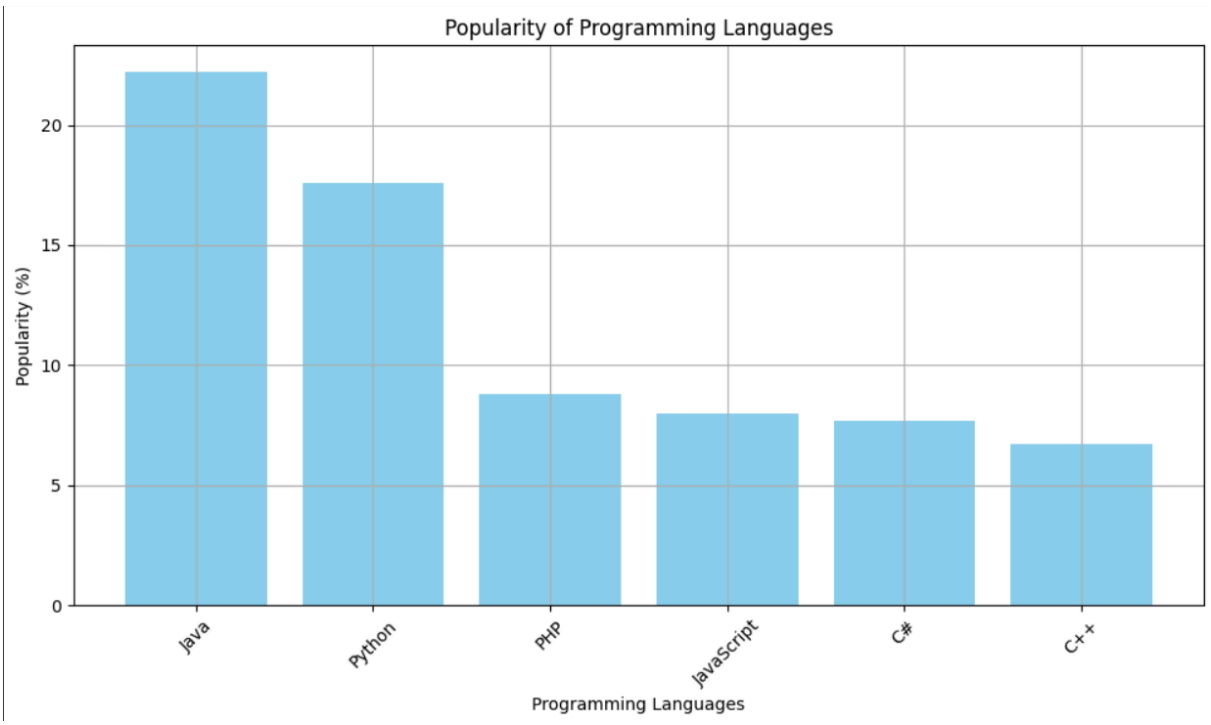
Step 4:Customize the Chart

Step 5:Display the Chart

CODE:

```
import matplotlib.pyplot as plt

languages = ['Java', 'Python', 'PHP', 'JavaScript', 'C#', 'C++']

popularity = [22.2, 17.6, 8.8, 8, 7.7, 6.7]

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

plt.bar(languages, popularity, color='skyblue')

plt.xlabel('Programming Languages')

plt.ylabel('Popularity (%)')

plt.title('Popularity of Programming Languages')

plt.xticks(rotation=45)

plt.grid(True)

plt.tight_layout()

plt.show()
```

OUTPUT:



28.Write a Python programming to display a horizontal bar chart of the popularity of programming Languages.
Sample data:
Programming languages: Java, Python, PHP, JavaScript, C#, C++
Popularity: 22.2, 17.6, 8.8, 8, 7.7, 6.7

AIM:

To display a horizontal bar chart of the popularity of programming Languages.

ALGORITHM:

Step 1:Import Libraries and Define Data

Step 2:Create the Figure and Horizontal Bar Chart

Step 3:Add Labels and Title

Step 4:Customize the Chart

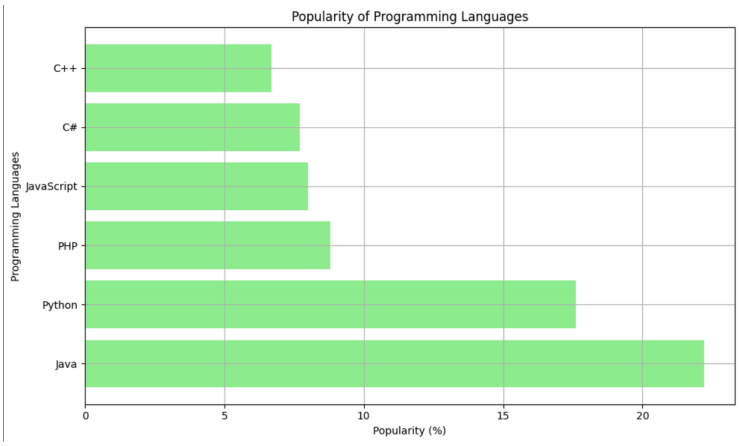
Step 5:Display the Chart

CODE:

```
import matplotlib.pyplot as plt

languages = ['Java', 'Python', 'PHP', 'JavaScript', 'C#', 'C++']

popularity = [22.2, 17.6, 8.8, 8, 7.7, 6.7]
```

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

plt.barh(languages, popularity, color='lightgreen')

plt.xlabel('Popularity (%)')

plt.ylabel('Programming Languages')

plt.title('Popularity of Programming Languages')

plt.grid(True)

plt.tight_layout()

plt.show()
```

OUTPUT:



29.Write a Python programming to display a bar chart of the popularity of programming Languages. Use different color for each bar.
Sample data:
Programming languages: Java, Python, PHP, JavaScript, C#, C++
Popularity: 22.2, 17.6, 8.8, 8, 7.7, 6.7

AIM:

To display a bar chart of popularity of programming languages.

ALGORITHM:

Step 1:Import Libraries and Define Data

Step 2:Create the Figure and colourful Bar Chart

Step 3:Add Labels and Title

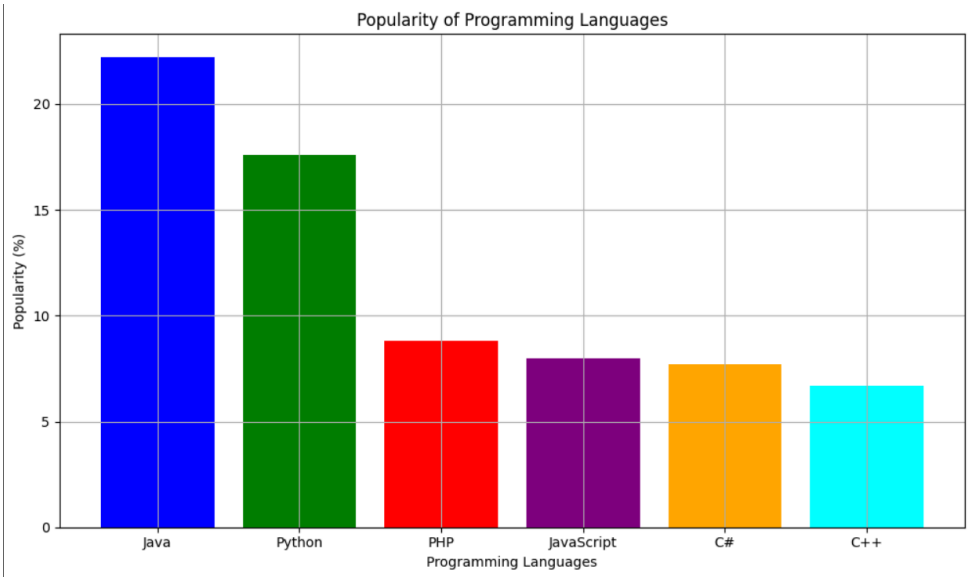
Step 4:Customize the Chart

Step 5:Display the Chart

CODE:

```
import matplotlib.pyplot as plt

languages = ['Java', 'Python', 'PHP', 'JavaScript', 'C#', 'C++']

popularity = [22.2, 17.6, 8.8, 8, 7.7, 6.7]

colors = ['blue', 'green', 'red', 'purple', 'orange', 'cyan']

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

plt.bar(languages, popularity, color=colors)

plt.xlabel('Programming Languages')

plt.ylabel('Popularity (%)')

plt.title('Popularity of Programming Languages')

plt.grid(True)

plt.tight_layout()

plt.show()
```

OUTPUT:



30.Write a Python program to create bar plot of scores by group and gender. Use multiple X values on the same chart for men and women.

AIM:

To create bar plot of scores by group and gender.

ALGORITHM:

Step 1: Define Chart Data and Colors

Step 2:Create Chart Figure and Bars

Step 3:Add Chart Labels and Title

Step 4:Customize Chart Appearance

Step 5:Display the Chart

CODE:

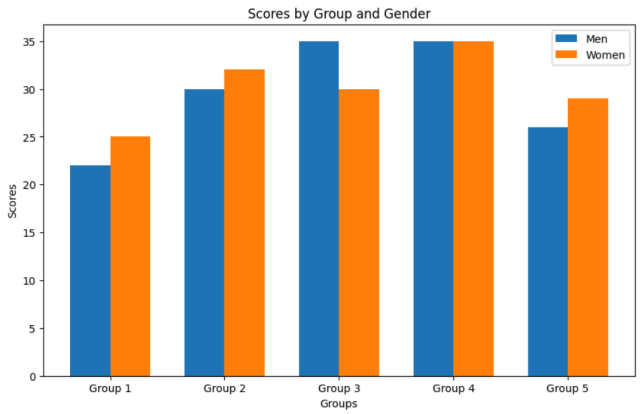
```
import matplotlib.pyplot as plt

import numpy as np

# Sample data

men_means = (22, 30, 35, 35, 26)

women_means = (25, 32, 30, 35, 29)

labels = ['Group 1', 'Group 2', 'Group 3', 'Group 4', 'Group 5']

num_groups = len(labels)

bar_width = 0.35

index = np.arange(num_groups)

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

bar1 = plt.bar(index, men_means, bar_width, label='Men')

bar2 = plt.bar(index + bar_width, women_means, bar_width, label='Women')

plt.xlabel('Groups')

plt.ylabel('Scores')
```

plt.title('Scores by Group and Gender')

plt.xticks(index + bar_width / 2, labels)

plt.legend()

OUTPUT:



31.Write a Python program to create a stacked bar plot with error bars.
Note: Use bottom to stack the women?s bars on top of the men?s bars.
Sample Data:
Means (men) = (22, 30, 35, 35, 26)
Means (women) = (25, 32, 30, 35, 29)
Men Standard deviation = (4, 3, 4, 1, 5)
Women Standard deviation = (3, 5, 2, 3, 3)

AIM:

To create a stacked bar plot with error bars.

ALGORITHM:

Step 1: Define Chart Data and Colors

Step 2:Create Chart Figure and Bars

Step 3:Add Chart Labels and Title

Step 4:Customize Chart Appearance

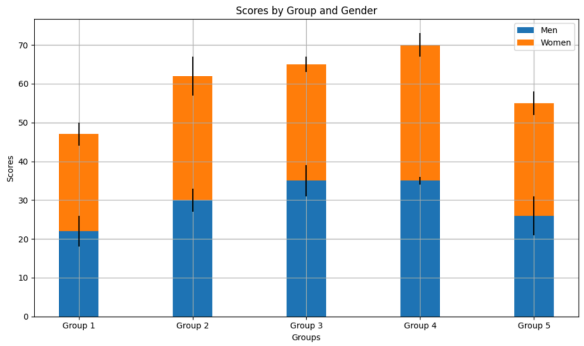
Step 5:Display the Chart

CODE:

import matplotlib.pyplot as plt

```
import numpy as np

men_means = (22, 30, 35, 35, 26)

women_means = (25, 32, 30, 35, 29)

men_std = (4, 3, 4, 1, 5)

women_std = (3, 5, 2, 3, 3)

labels = ['Group 1', 'Group 2', 'Group 3', 'Group 4', 'Group 5']

num_groups = len(labels)

bar_width = 0.35

index = np.arange(num_groups)

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

bar1 = plt.bar(index, men_means, bar_width, yerr=men_std, label='Men')

bar2 = plt.bar(index, women_means, bar_width, bottom=men_means, yerr=women_std, label='Women')

plt.xlabel('Groups')

plt.ylabel('Scores')

plt.title('Scores by Group and Gender')

plt.xticks(index, labels)

plt.legend()

plt.grid(True)

plt.tight_layout()

plt.show()
```

OUTPUT:



32.Write a Python program to draw a scatter graph taking a random distribution in X and Y and plotted against each other.

AIM:

To draw a scatter graph taking a random distribution in X and Y and plotted against each other.

ALGORITHM:

Step 1:Generate Random Data

Step 2:Create Chart Figure

Step 3:Create Scatter Plot

Step 4:Add Chart Labels and Title

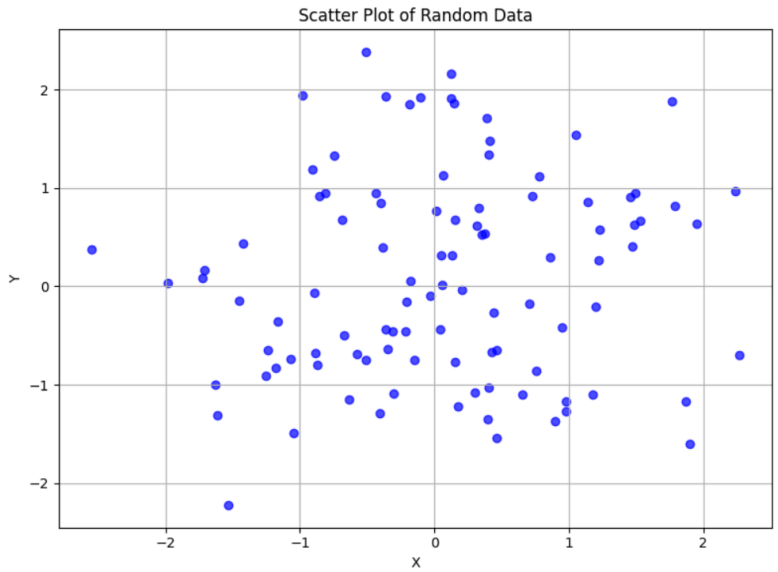
Step 5:Customize and Display the Chart

CODE:

```
import matplotlib.pyplot as plt

import numpy as np

np.random.seed(0)  # Setting seed for reproducibility

x = np.random.randn(100)  # Random X values

y = np.random.randn(100)  # Random Y values

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

plt.scatter(x, y, color='blue', alpha=0.7)  # Alpha controls transparency
```

```
plt.xlabel('X')

plt.ylabel('Y')

plt.title('Scatter Plot of Random Data')

plt.grid(True)

plt.tight_layout()

plt.show()
```

OUTPUT:



33.Write a Python program to draw a scatter plot with empty circles taking a random distribution in X and Y and plotted against each other.

AIM:

To draw a scatter plot with empty circles taking a random distribution in X and Y and plotted against each other.

ALGORITHM:

Step 1:Generate Random Data

Step 2:Create Chart Figure

Step 3:Create Scatter Plot with Custom Markers

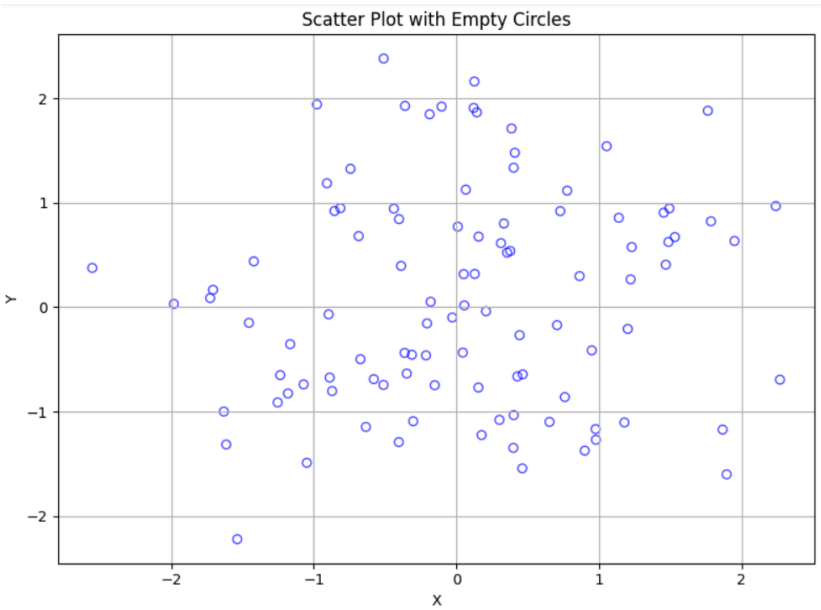
Step 4:Add Chart Labels and Title

Step 5:Customize and Display the Chart

CODE:

```
import matplotlib.pyplot as plt

import numpy as np

np.random.seed(0)  # Setting seed for reproducibility

x = np.random.randn(100)  # Random X values

y = np.random.randn(100)  # Random Y values

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

plt.scatter(x, y, color='blue', alpha=0.7, marker='o', facecolors='none', edgecolors='blue')

plt.xlabel('X')

plt.ylabel('Y')

plt.title('Scatter Plot with Empty Circles')

plt.grid(True)

plt.tight_layout()

plt.show()
```

OUTPUT:



34.Write a Python program to draw a scatter plot using random distributions to generate balls of different sizes.

AIM:

To draw a scatter plot using random distributions to generate balls of different sizes.

ALGORITHM:

Step 1:Generate Random Data

Step 2:Create Chart Figure

Step 3:Create Scatter Plot with Custom Markers

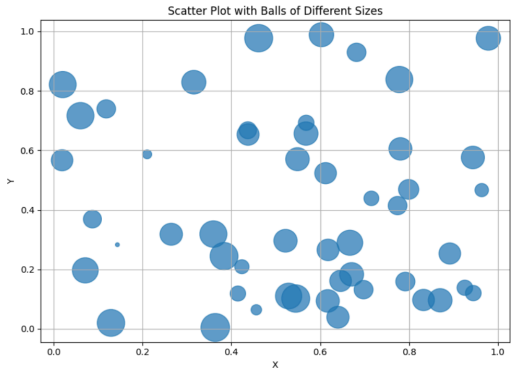
Step 4:Add Chart Labels and Title

Step 5:Customize and Display the Chart

CODE:

```
import matplotlib.pyplot as plt

import numpy as np

np.random.seed(0)  # Setting seed for reproducibility

x = np.random.rand(50)  # Random X values

y = np.random.rand(50)  # Random Y values

sizes = np.random.rand(50) * 1000  # Random sizes for the balls

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

plt.scatter(x, y, s=sizes, alpha=0.7)

plt.xlabel('X')

plt.ylabel('Y')

plt.title('Scatter Plot with Balls of Different Sizes')

plt.grid(True)

plt.tight_layout()

plt.show()
```

OUTPUT:



35.Write a Python program to draw a scatter plot comparing two subject marks of Mathematics and Science. Use marks of 10 students.
Sample data:

Test Data:
math_marks = [88, 92, 80, 89, 100, 80, 60, 100, 80, 34]
science_marks = [35, 79, 79, 48, 100, 88, 32, 45, 20, 30]
marks_range = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

AIM:

To draw a scatter plot comparing two subject marks of Mathematics and Science.

ALGORITHM:

Step 1:Define Data Points

Step 2:Create Chart Figure

Step 3:Create Scatter Plot

Step 4:Add Chart Labels and Title

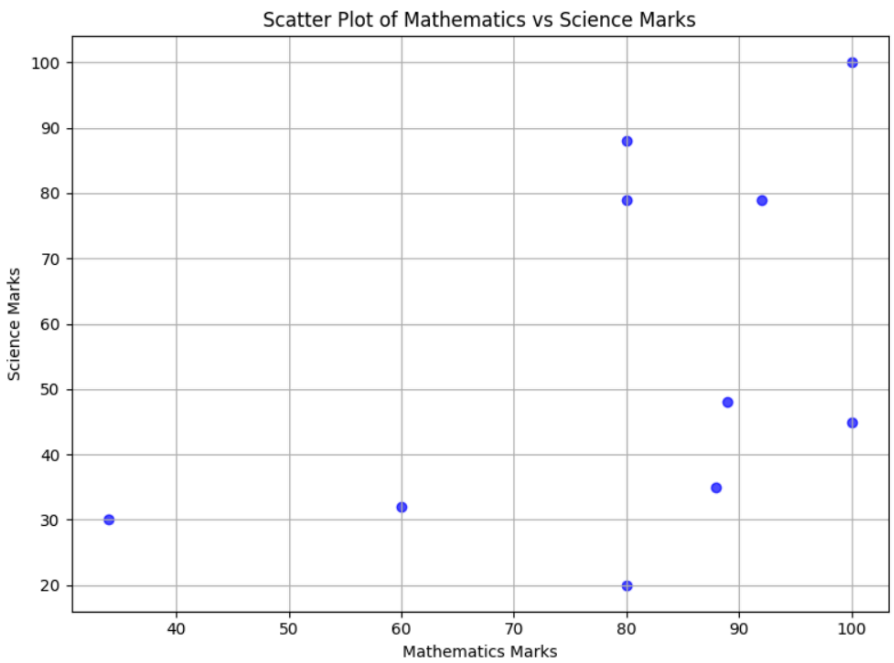
Step 5:Customize and Display the Chart

CODE:

```
import matplotlib.pyplot as plt

math_marks = [88, 92, 80, 89, 100, 80, 60, 100, 80, 34]

science_marks = [35, 79, 79, 48, 100, 88, 32, 45, 20, 30]

marks_range = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

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

```
plt.scatter(math_marks, science_marks, color='blue', alpha=0.7)


plt.xlabel('Mathematics Marks')

plt.ylabel('Science Marks')

plt.title('Scatter Plot of Mathematics vs Science Marks')


plt.grid(True)

plt.tight_layout()

plt.show()
```

OUTPUT:



36.Write a Python program to draw a scatter plot for three different groups comparing weights and heights.

AIM:

To draw a scatter plot for three different groups comparing weights and heights.

ALGORITHM:

Step 1:Define Data Points for Each Group

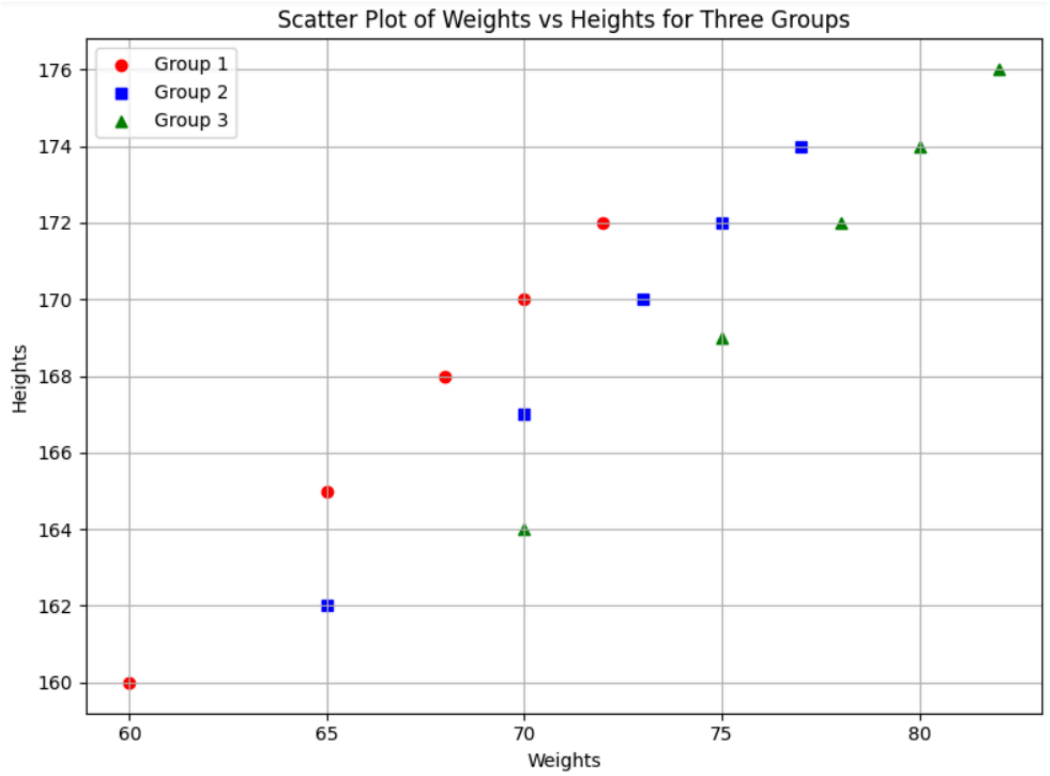
Step 2:Create Chart Figure

Step 3:Create Scatter Plots for Each Group

Step 4:Add Chart Labels, Title, and Legend

Step 5:Customize and Display the Chart

CODE:



37.Write a Pandas program to create a dataframe from a dictionary and display it.
Sample data: {'X':[78,85,96,80,86], 'Y':[84,94,89,83,86],'Z':[86,97,96,72,83]}

AIM:

To create a dataframe from a dictionary and display it.

ALGORITHM:

Step 1:Define the Data Dictionary

Step 2:Import the pandas Library

Step 3:Create a DataFrame Object

Step 4:Assign the DataFrame to a Variable

Step 5:Print the DataFrame

CODE:

```
import pandas as pd

data = {'X': [78, 85, 96, 80, 86],

        'Y': [84, 94, 89, 83, 86],

        'Z': [86, 97, 96, 72, 83]}

df = pd.DataFrame(data)

print("DataFrame:")

print(df)
```

OUTPUT:

```
DataFrame:
    X   Y   Z
0  78  84  86
1  85  94  97
2  96  89  96
3  80  83  72
4  86  86  83
```

38.Write a Pandas program to create and display a DataFrame from a specified dictionary data which has the index labels.
Sample Python dictionary data and list labels:
exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],
'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

AIM:

To create and display a DataFrame from a specified dictionary data which has the index labels.

ALGORITHM:

Step 1:Define the Data Dictionary

Step 2:Define the Custom Index

Step 3:Import necessary libraries

Step 4:Create a DataFrame Object with Custom Index

Step 5:Print the DataFrame

CODE:

```
import pandas as pd

import numpy as np

exam_data = {

    'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],

  'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],

  'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],

  'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']

}

labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

df = pd.DataFrame(exam_data, index=labels)

print("DataFrame:")

print(df)
```

OUTPUT:

```
DataFrame:
        name  score  attempts qualify
a  Anastasia   12.5         1     yes
b       Dima    9.0         3      no
c  Katherine   16.5         2     yes
d      James    NaN         3      no
e      Emily    9.0         2      no
f    Michael   20.0         3     yes
g    Matthew   14.5         1     yes
h      Laura    NaN         1      no
i      Kevin    8.0         2      no
j      Jonas   19.0         1     yes
```

39.Write a Pandas program to get the first 3 rows of a given DataFrame.
Sample Python dictionary data and list labels:
exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],
'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

AIM:

To get the first 3 rows of a given DataFrame.

ALGORITHM:

Step 1:Create a DataFrame Object

Step 2:Specify the Number of Rows to Select

Step 3:Use the head() Method

Step 4:Assign the Selected Rows to a New Variable

Step 5:Print the Selected Rows

CODE:

```
import pandas as pd

import numpy as np

exam_data = {
```

'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],

'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],

'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],

'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']

}

labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

df = pd.DataFrame(exam_data, index=labels)

first_3_rows = df.head(3)

print("First 3 rows of the DataFrame:")

print(first_3_rows)

OUTPUT:

```
First 3 rows of the DataFrame:
        name  score  attempts qualify
a  Anastasia   12.5         1     yes
b       Dima    9.0         3      no
c  Katherine   16.5         2     yes
```

40. Write a Pandas program to select the 'name' and 'score' columns from the following DataFrame.
    Sample Python dictionary data and list labels:
    exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],
    'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
    'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
    'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
    labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

AIM:

To select the 'name' and 'score' columns from the following DataFrame.

ALGORITHM:

Step 1:Create a DataFrame Object

Step 2:Specify the Columns to Select

Step 3:Use Square Bracket Notation

Step 4:Assign the Selected Columns to a New Variable

Step 5:Print the Selected Columns

CODE:

```
import pandas as pd

import numpy as np

exam_data = {

    'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],

  'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],

  'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],

  'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']

}

labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

df = pd.DataFrame(exam_data, index=labels)

selected_columns = df[['name', 'score']]


print("Selected columns:")

print(selected_columns)
```

OUTPUT:

```
Selected columns:
        name  score
a  Anastasia   12.5
b       Dima    9.0
c  Katherine   16.5
d      James    NaN
e      Emily    9.0
f    Michael   20.0
g    Matthew   14.5
h      Laura    NaN
i      Kevin    8.0
j      Jonas   19.0
```