**Name:**B.Sumanth

**Reg No:**192124114

**Course Code:**DSA0503

**Query Processing Partical Programs**

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

+---------------+----------------------+------------+-------------+

| DEPARTMENT\_ID | DEPARTMENT\_NAME | MANAGER\_ID | LOCATION\_ID |

+---------------+----------------------+------------+-------------+

| 10 | Administration | 200 | 1700 |

| 20 | Marketing | 201 | 1800 |

| 30 | Purchasing | 114 | 1700 |

| 40 | Human Resources | 203 | 2400 |

| 50 | Shipping | 121 | 1500 |

| 60 | IT | 103 | 1400 |

| 70 | Public Relations | 204 | 2700 |

| 80 | Sales | 145 | 2500 |

| 90 | Executive | 100 | 1700 |

| 100 | Finance | 108 | 1700 |

| 110 | Accounting | 205 | 1700 |

| 120 | Treasury | 0 | 1700 |

| 130 | Corporate Tax | 0 | 1700 |

| 140 | Control And Credit | 0 | 1700 |

| 150 | Shareholder Services | 0 | 1700 |

| 160 | Benefits | 0 | 1700 |

| 170 | Manufacturing | 0 | 1700 |

| 180 | Construction | 0 | 1700 |

| 190 | Contracting | 0 | 1700 |

| 200 | Operations | 0 | 1700 |

| 210 | IT Support | 0 | 1700 |

| 220 | NOC | 0 | 1700 |

| 230 | IT Helpdesk | 0 | 1700 |

| 240 | Government Sales | 0 | 1700 |

| 250 | Retail Sales | 0 | 1700 |

| 260 | Recruiting | 0 | 1700 |

| 270 | Payroll | 0 | 1700 |

+---------------+----------------------+------------+-------------

**Program:**

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]

}

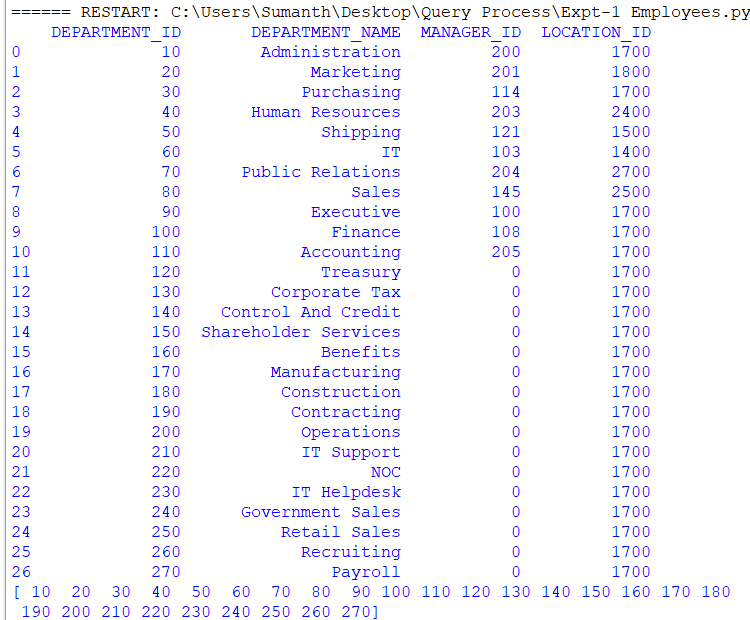
employees\_df = pd.DataFrame(data)

print(employees\_df)

distinct\_department\_ids = employees\_df['DEPARTMENT\_ID'].unique()

print(distinct\_department\_ids)

**Output:**

****

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

+-------------+------------+------------+------------+---------------+

| EMPLOYEE\_ID | START\_DATE | END\_DATE | JOB\_ID | DEPARTMENT\_ID |

+-------------+------------+------------+------------+---------------+

| 102 | 2001-01-13 | 2006-07-24 | IT\_PROG | 60 |

| 101 | 1997-09-21 | 2001-10-27 | AC\_ACCOUNT | 110 |

| 101 | 2001-10-28 | 2005-03-15 | AC\_MGR | 110 |

| 201 | 2004-02-17 | 2007-12-19 | MK\_REP | 20 |

| 114 | 2006-03-24 | 2007-12-31 | ST\_CLERK | 50 |

| 122 | 2007-01-01 | 2007-12-31 | ST\_CLERK | 50 |

| 200 | 1995-09-17 | 2001-06-17 | AD\_ASST | 90 |

| 176 | 2006-03-24 | 2006-12-31 | SA\_REP | 80 |

| 176 | 2007-01-01 | 2007-12-31 | SA\_MAN | 80 |

| 200 | 2002-07-01 | 2006-12-31 | AC\_ACCOUNT | 90 |

+-------------+------------+------------+------------+---------------+

**Program:**

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\_count = df.groupby('EMPLOYEE\_ID')['JOB\_ID'].nunique()

employees\_with\_multiple\_jobs = employee\_jobs\_count[employee\_jobs\_count >= 2]

print(employees\_with\_multiple\_jobs.index.tolist())

**Output:**



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

+------------+---------------------------------+------------+------------+

| JOB\_ID | JOB\_TITLE | MIN\_SALARY | MAX\_SALARY |

+------------+---------------------------------+------------+------------+

| AD\_PRES | President | 20080 | 40000 |

| AD\_VP | Administration Vice President | 15000 | 30000 |

| AD\_ASST | Administration Assistant | 3000 | 6000 |

| FI\_MGR | Finance Manager | 8200 | 16000 |

| FI\_ACCOUNT | Accountant | 4200 | 9000 |

| AC\_MGR | Accounting Manager | 8200 | 16000 |

| AC\_ACCOUNT | Public Accountant | 4200 | 9000 |

| SA\_MAN | Sales Manager | 10000 | 20080 |

| SA\_REP | Sales Representative | 6000 | 12008 |

| PU\_MAN | Purchasing Manager | 8000 | 15000 |

| PU\_CLERK | Purchasing Clerk | 2500 | 5500 |

| ST\_MAN | Stock Manager | 5500 | 8500 |

| ST\_CLERK | Stock Clerk | 2008 | 5000 |

| SH\_CLERK | Shipping Clerk | 2500 | 5500 |

| IT\_PROG | Programmer | 4000 | 10000 |

| MK\_MAN | Marketing Manager | 9000 | 15000 |

| MK\_REP | Marketing Representative | 4000 | 9000 |

| HR\_REP | Human Resources Representative | 4000 | 9000 |

| PR\_REP | Public Relations Representative | 4500 | 10500 |

+------------+---------------------------------+------------+------------+

**Program:**

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)

print("original\_data")

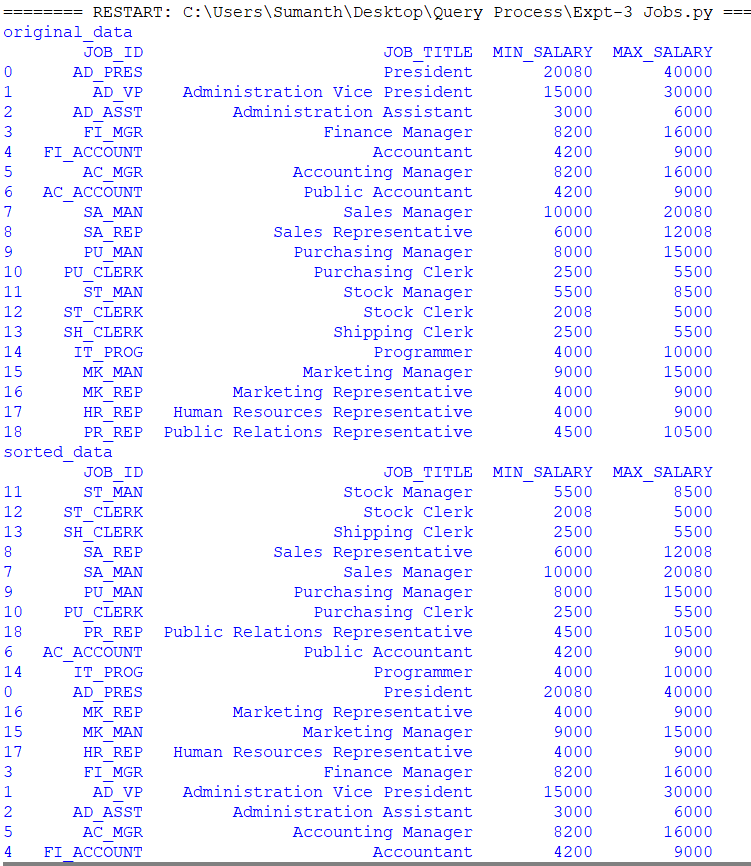
print(df)

df\_sorted = df.sort\_values(by='JOB\_TITLE', ascending=False)

print("sorted\_data")

print(df\_sorted)

**Output:**



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

**Program:**

import pandas as pd

import matplotlib.pyplot as plt

data = {

'Date': ['2023-01-02', '2023-01-03', '2023-01-04', '2023-01-05', '2023-01-06'],

'Close\_Price': [2800.00, 2820.00, 2830.00, 2840.00, 2860.00]

}

df = pd.DataFrame(data)

df['Date'] = pd.to\_datetime(df['Date'])

df.set\_index('Date', inplace=True)

start\_date = '2023-01-02'

end\_date = '2023-01-06'

filtered\_data = df[start\_date:end\_date]

plt.figure(figsize=(12, 6))

plt.plot(filtered\_data.index, filtered\_data['Close\_Price'], marker='o', linestyle='-', color='b')

plt.title('Alphabet Inc. Stock Prices')

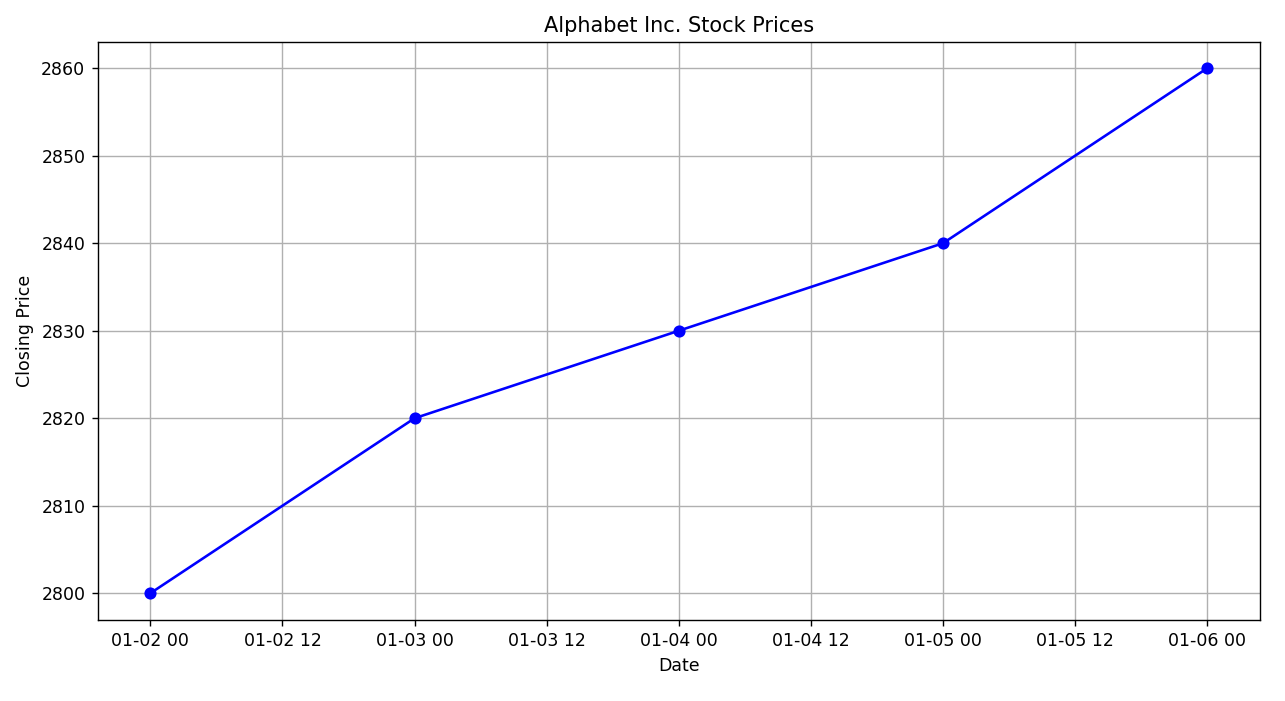
plt.xlabel('Date')

plt.ylabel('Closing Price')

plt.grid(True)

plt.show()

**Output:**

****

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

**Program:**

import pandas as pd

import matplotlib.pyplot as plt

data = {

'Date': ['2023-01-02', '2023-01-03', '2023-01-04', '2023-01-05', '2023-01-06'],

'Trading\_Volume': [1000000, 1200000, 800000, 1500000, 1100000]

}

df = pd.DataFrame(data)

df['Date'] = pd.to\_datetime(df['Date'])

df.set\_index('Date', inplace=True)

start\_date = '2023-01-02'

end\_date = '2023-01-06'

filtered\_data = df[start\_date:end\_date]

plt.figure(figsize=(12, 6))

plt.bar(filtered\_data.index, filtered\_data['Trading\_Volume'], color='b', alpha=0.7)

plt.title('Alphabet Inc. Trading Volume')

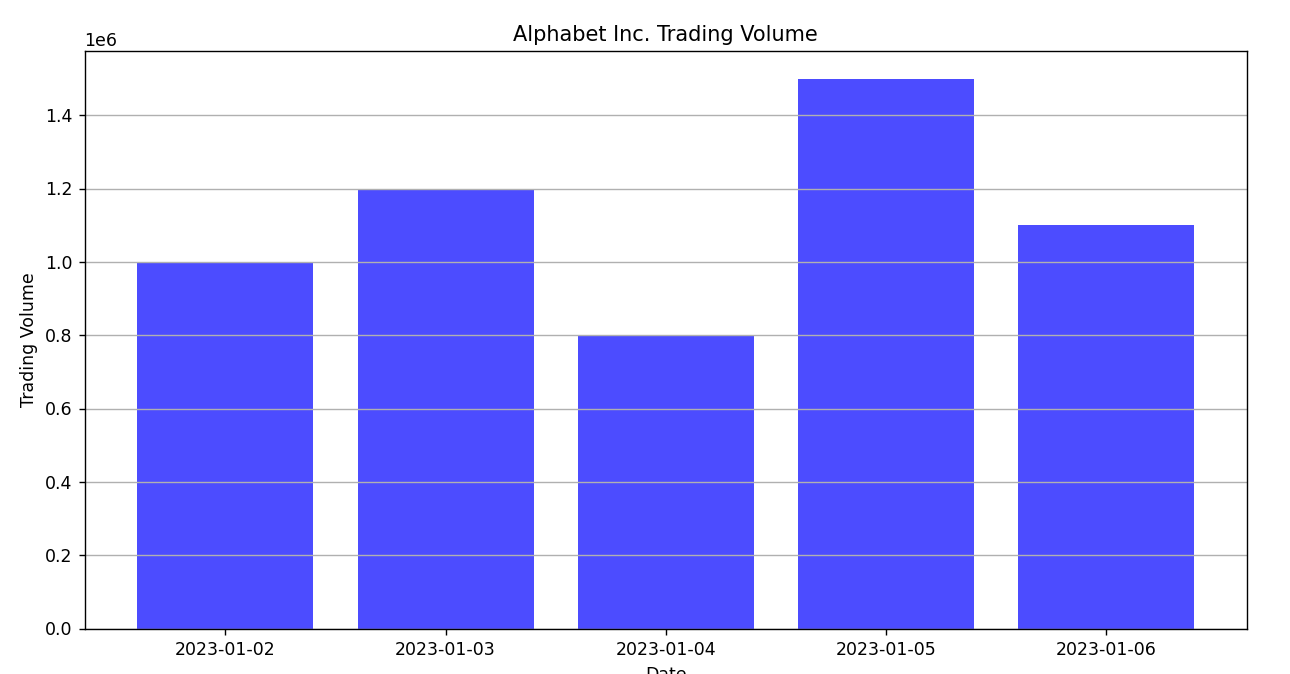
plt.xlabel('Date')

plt.ylabel('Trading Volume')

plt.grid(axis='y')

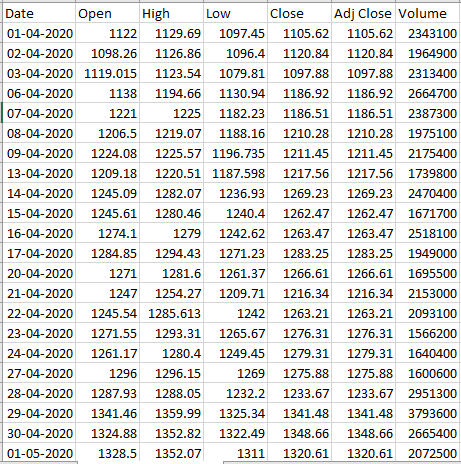
plt.show()

**Output:**

****

1. 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:**



**Program:**

import pandas as pd

import matplotlib.pyplot as plt

data = {

'Date': ['01-04-2020', '02-04-2020', '03-04-2020', '06-04-2020', '07-04-2020', '08-04-2020', '09-04-2020',

'13-04-2020', '14-04-2020', '15-04-2020', '16-04-2020', '17-04-2020', '20-04-2020', '21-04-2020',

'22-04-2020', '23-04-2020', '24-04-2020', '27-04-2020', '28-04-2020', '29-04-2020', '30-04-2020',

'01-05-2020'],

'Open': [1122, 1098.26, 1119.015, 1138, 1221, 1206.5, 1224.08, 1209.18, 1245.09, 1245.61, 1274.1, 1284.85, 1271, 1247, 1245.54, 1271.55, 1261.17, 1296, 1287.93, 1341.46, 1324.88, 1328.5],

'High': [1129.69, 1126.86, 1123.54, 1194.66, 1225, 1219.07, 1225.57, 1220.51, 1282.07, 1280.46, 1279, 1294.43, 1281.6, 1254.27, 1285.613, 1293.31, 1280.4, 1296.15, 1288.05, 1359.99, 1352.82, 1352.07],

'Low': [1097.45, 1096.4, 1079.81, 1130.94, 1182.23, 1188.16, 1196.735, 1187.598, 1236.93, 1240.4, 1242.62, 1271.23, 1261.37, 1209.71, 1242, 1265.67, 1249.45, 1269, 1232.2, 1325.34, 1322.49, 1311],

'Close': [1105.62, 1120.84, 1097.88, 1186.92, 1186.51, 1210.28, 1211.45, 1217.56, 1269.23, 1262.47, 1263.47, 1283.25, 1266.61, 1216.34, 1263.21, 1276.31, 1279.31, 1275.88, 1233.67, 1341.48, 1348.66, 1320.61],

'Adj Close': [1105.62, 1120.84, 1097.88, 1186.92, 1186.51, 1210.28, 1211.45, 1217.56, 1269.23, 1262.47, 1263.47, 1283.25, 1266.61, 1216.34, 1263.21, 1276.31, 1279.31, 1275.88, 1233.67, 1341.48, 1348.66, 1320.61],

'Volume': [2343100, 1964900, 2313400, 2664700, 2387300, 1975100, 2175400, 1739800, 2470400, 1671700, 2518100, 1949000, 1695500, 2153000, 2093100, 1566200, 1640400, 1600600, 2951300, 3793600, 2665400, 2072500]

}

data['Date'] = pd.to\_datetime(data['Date'], format='%d-%m-%Y')

df = pd.DataFrame(data)

start\_date = '2020-04-03'

end\_date = '2020-04-10'

filtered\_data = df[(df['Date'] >= start\_date) & (df['Date'] <= end\_date)]

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

plt.scatter(filtered\_data['Date'], filtered\_data['Volume'], c=filtered\_data['Close'], cmap='viridis', marker='o')

plt.title('Trading Volume vs. Stock Price')

plt.xlabel('Date')

plt.ylabel('Volume')

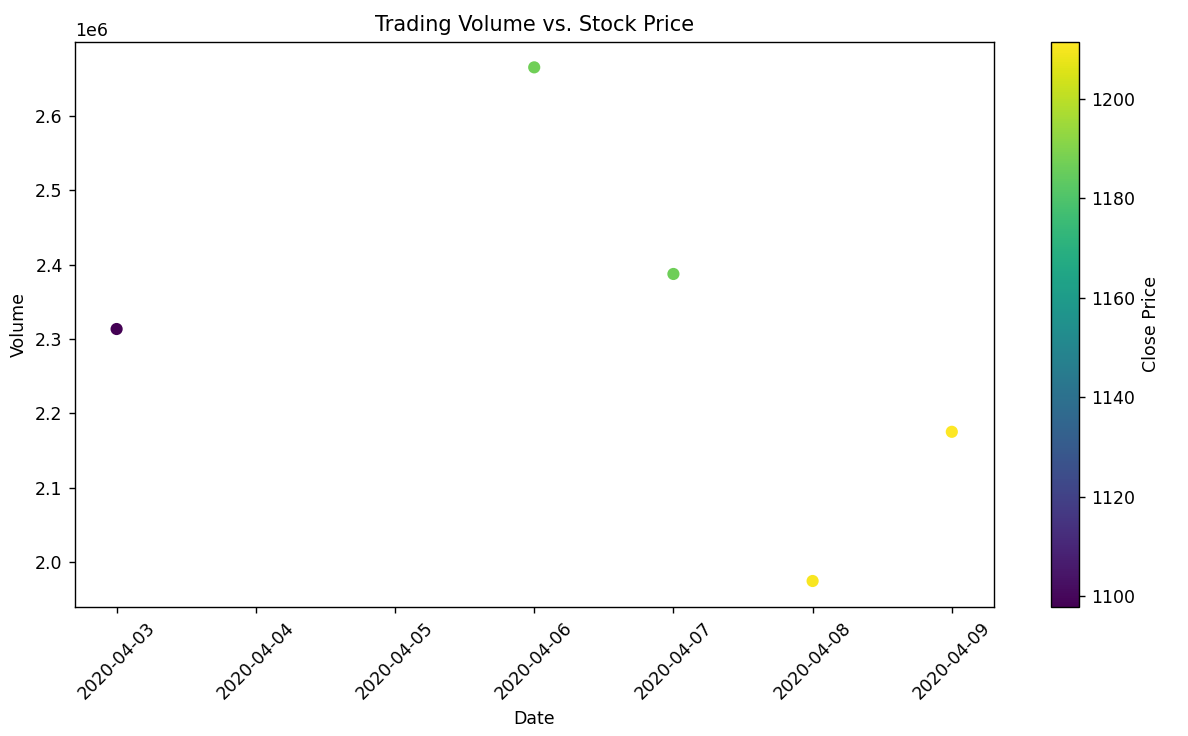
plt.colorbar(label='Close Price')

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

**Output:**

****

1. Write a Pandas program to create a Pivot table and find the maximum and minimum sale value of the items.(refer sales\_data table)

**Program:**

import pandas as pd

data = {

'Item': ['A', 'B', 'A', 'C', 'B', 'C', 'A', 'B', 'C'],

'Sale': [100, 150, 200, 120, 250, 180, 220, 130, 160]

}

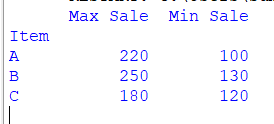
sales\_data = pd.DataFrame(data)

pivot\_table = sales\_data.pivot\_table(index='Item', values='Sale', aggfunc={'Sale': ['max', 'min']})

pivot\_table.columns = ['Max Sale', 'Min Sale']

print(pivot\_table)

**Output:**



1. Write a Pandas program to create a Pivot table and find the item wise unit sold. .(refer sales\_data table)

**Program:**

import pandas as pd

data = {

'Item': ['A', 'B', 'A', 'C', 'B', 'C', 'A', 'B', 'C'],

'Units Sold': [10, 15, 20, 12, 25, 18, 22, 13, 16]

}

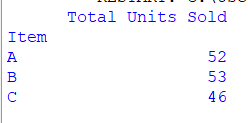
sales\_data = pd.DataFrame(data)

pivot\_table = sales\_data.pivot\_table(index='Item', values='Units Sold', aggfunc='sum')

pivot\_table.columns = ['Total Units Sold']

print(pivot\_table)

**Output:**



1. 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)

**Sales\_data:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **OrderDate** | **Region** | **Manager** | **SalesMan** | **Item** | **Units** | **Unit\_price** | **Sale\_amt** |
| 1-6-18 | East | Martha | Alexander | Television | 95 | 1,198.00 | 1,13,810.00 |
| 1-23-18 | Central | Hermann | Shelli | Home Theater | 50 | 500.00 | 25,000.00 |
| 2-9-18 | Central | Hermann | Luis | Television | 36 | 1,198.00 | 43,128.00 |
| 2-26-18 | Central | Timothy | David | Cell Phone | 27 | 225.00 | 6,075.00 |
| 3-15-18 | West | Timothy | Stephen | Television | 56 | 1,198.00 | 67,088.00 |
| 4-1-18 | East | Martha | Alexander | Home Theater | 60 | 500.00 | 30,000.00 |
| 4-18-18 | Central | Martha | Steven | Television | 75 | 1,198.00 | 89,850.00 |
| 5-5-18 | Central | Hermann | Luis | Television | 90 | 1,198.00 | 1,07,820.00 |
| 5-22-18 | West | Douglas | Michael | Television | 32 | 1,198.00 | 38,336.00 |
| 6-8-18 | East | Martha | Alexander | Home Theater | 60 | 500.00 | 30,000.00 |
| 6-25-18 | Central | Hermann | Sigal | Television | 90 | 1,198.00 | 1,07,820.00 |
| 7-12-18 | East | Martha | Diana | Home Theater | 29 | 500.00 | 14,500.00 |
| 7-29-18 | East | Douglas | Karen | Home Theater | 81 | 500.00 | 40,500.00 |
| 8-15-18 | East | Martha | Alexander | Television | 35 | 1,198.00 | 41,930.00 |
| 9-1-18 | Central | Douglas | John | Desk | 2 | 125.00 | 250.00 |
| 9-18-18 | East | Martha | Alexander | Video Games | 16 | 58.50 | 936.00 |
| 10-5-18 | Central | Hermann | Sigal | Home Theater | 28 | 500.00 | 14,000.00 |
| 10-22-18 | East | Martha | Alexander | Cell Phone | 64 | 225.00 | 14,400.00 |

**Program:**

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', 'Central', 'Central', 'East', 'Central', 'East'],

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

}

sales\_data = pd.DataFrame(data)

pivot\_table = pd.pivot\_table(sales\_data,

values='Sale\_amt',

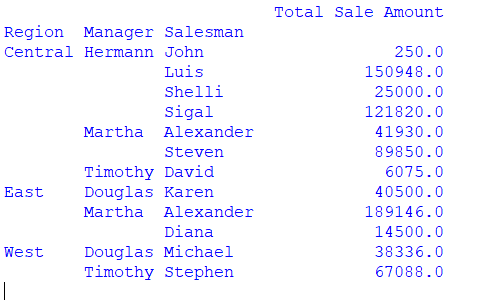
index=['Region', 'Manager', 'Salesman'],

aggfunc='sum')

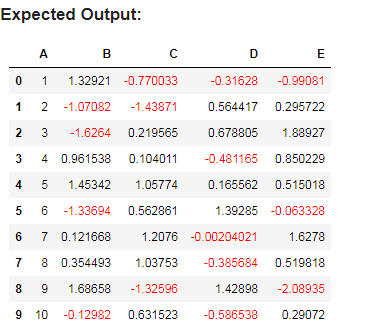
pivot\_table.columns = ['Total Sale Amount']

print(pivot\_table)

**Output:**



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.



**Program:**

import pandas as pd

import numpy as np

data = {

'Column1': np.random.uniform(-1, 1, 10),

'Column2': np.random.uniform(-1, 1, 10),

'Column3': np.random.uniform(-1, 1, 10),

'Column4': np.random.uniform(-1, 1, 10)

}

df = pd.DataFrame(data)

def color\_negative\_red(val):

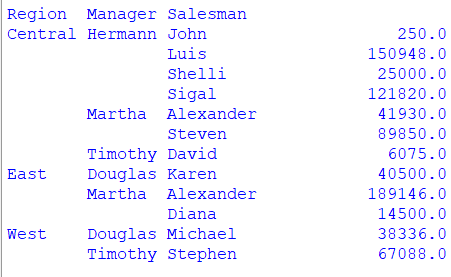
color = 'red' if val < 0 else 'black'

return f'color: {color}'

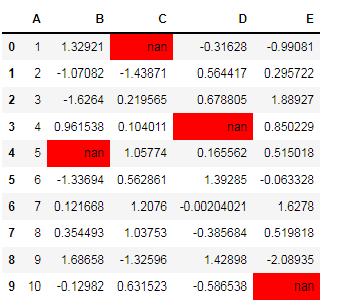
styled\_df = df.style.applymap(color\_negative\_red)

styled\_df

**Output:**



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.



**Program:**

import pandas as pd

import numpy as np

np.random.seed(24)

df = pd.DataFrame({'A': np.linspace(1, 10, 10)})

df = pd.concat([df, pd.DataFrame(np.random.randn(10, 4), columns=list('BCDE'))],

axis=1)

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

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

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

df.iloc[9, 4] = np.nan

print("Original array:")

print(df)

def color\_negative\_red(val):

color = 'red' if val < 0 else 'black'

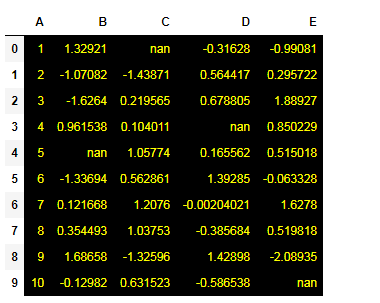
return 'color: %s' % color

print("\nNegative numbers red and positive numbers black:")

df.style.highlight\_null(null\_color='red')

**Output:**



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.

**Program:**

import pandas as pd

import numpy as np

from IPython.display import display, HTML

data = {

'Column1': np.random.rand(10),

'Column2': np.random.rand(10),

'Column3': np.random.rand(10),

'Column4': np.random.rand(10)

}

df = pd.DataFrame(data)

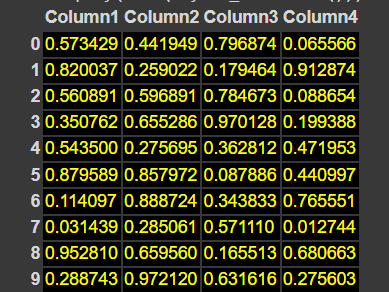
def highlight(val):

return 'background-color: black; color: yellow;'

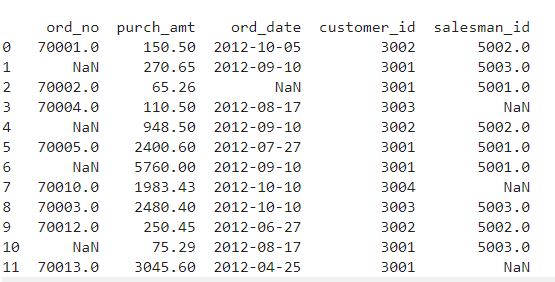
styled\_df = df.style.applymap(highlight)

display(HTML(styled\_df.render()))

**output:**



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



**Program:**

import pandas as pd

data = {'A': [1, 2, None, 4, 5],

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

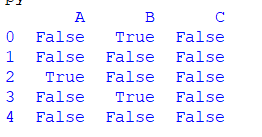
'C': [1, 2, 3, 4, 5]}

df = pd.DataFrame(data)

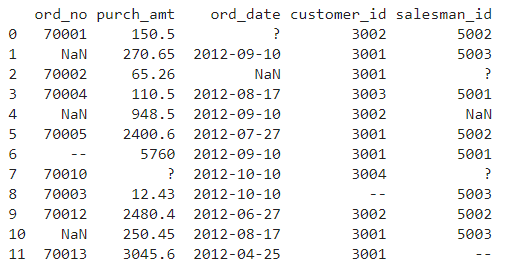
missing\_values = df.isna()

print(missing\_values)

**Output:**



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



**Program:**

import pandas as pd

import numpy as np

data = {'A': [1, 2, None, 4, 5],

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

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

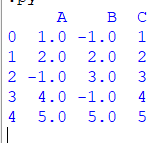
df = pd.DataFrame(data)

value\_to\_replace = -1

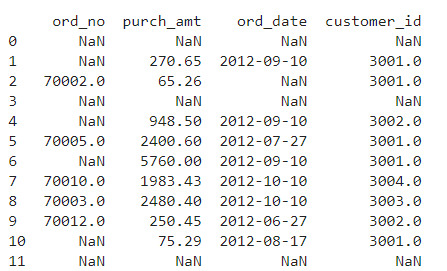
df\_filled = df.fillna(value\_to\_replace)

print(df\_filled)

**Output:**



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



**Program:**

import pandas as pd

import numpy as np

data = {'A': [1, 2, None, 4, None],

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

'C': [1, 2, None, None, 5]}

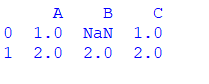
df = pd.DataFrame(data)

threshold = 2

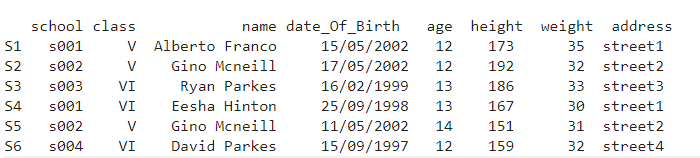
filtered\_df = df.dropna(thresh=threshold)

print(filtered\_df)

**Output:**



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



**Program:**

import pandas as pd

df = pd.DataFrame({'class': ['V', 'V', 'VI', 'VI', 'V', 'VI'],

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

'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],

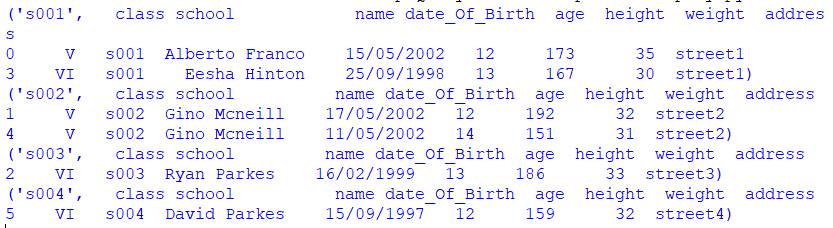
'address': ['street1', 'street2', 'street3', 'street1', 'street2', 'street4']})

groups = list(df.groupby('school'))

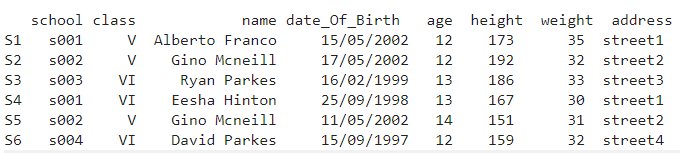
for group in groups:

print(group)

**Output:**



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.



**Program:**

import pandas as pd

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

'Age': [25, 22, 24, 23, 25],

'School Code': ['S001', 'S002', 'S001', 'S003', 'S002']}

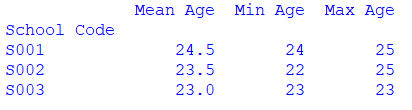
df = pd.DataFrame(data)

result = df.groupby('School Code')['Age'].agg(['mean', 'min', 'max'])

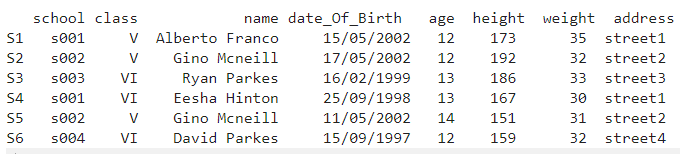
result = result.rename(columns={'mean': 'Mean Age', 'min': 'Min Age', 'max': 'Max Age'})

print(result)

**Output:**



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



**Program:**

import pandas as pd

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

'Age': [25, 22, 24, 23, 25],

'School Code': ['S001', 'S002', 'S001', 'S003', 'S002'],

'Class': ['A', 'B', 'A', 'C', 'B']}

df = pd.DataFrame(data)

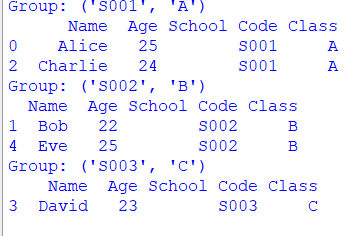
grouped = df.groupby(['School Code', 'Class'])

for group, group\_df in grouped:

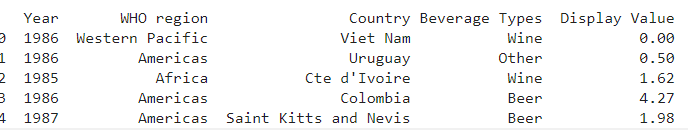
print("Group:", group)

print(group\_df)

**Output:**



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.



**Program:**

import pandas as pd

data = {

'Year': [1986, 1986, 1986, 1986, 1985],

'WHO region': ['Western Pacific', 'Americas', 'Africa', 'Americas', 'Americas'],

'Country': ["Viet Nam", "Uruguay", "Cote d'Ivoire", "Colombia", "Saint Kitts"],

'Beverage Types': ['Wine', 'Other', 'Wine', 'Beer', 'Beer'],

'Display Value': [0.00, 0.50, 3.62, 4.27, 1.98]

}

df = pd.DataFrame(data)

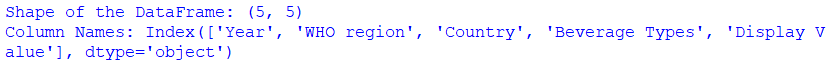
shape = df.shape

print("Shape of the DataFrame:", shape)

column\_names = df.columns

print("Column Names:", column\_names)

**Output:**



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

**Program:**

import pandas as pd

data = {'Column1': ['apple', 'banana', 'cherry', 'date', 'elderberry']}

df = pd.DataFrame(data)

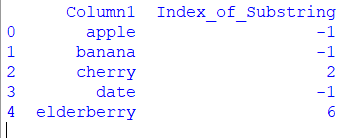
substring = 'erry'

index\_of\_substring = df['Column1'].str.find(substring)

df['Index\_of\_Substring'] = index\_of\_substring

print(df)

**Output:**



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

**Program:**

import pandas as pd

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

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

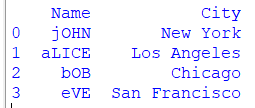
df = pd.DataFrame(data)

column\_to\_swap = 'Name'

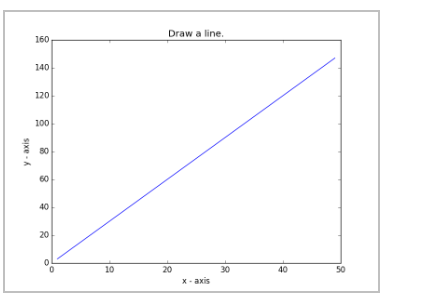
df[column\_to\_swap] = df[column\_to\_swap].str.swapcase()

print(df)

**Output:**



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

****

**Program:**

import matplotlib.pyplot as plt

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

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

plt.plot(x, y)

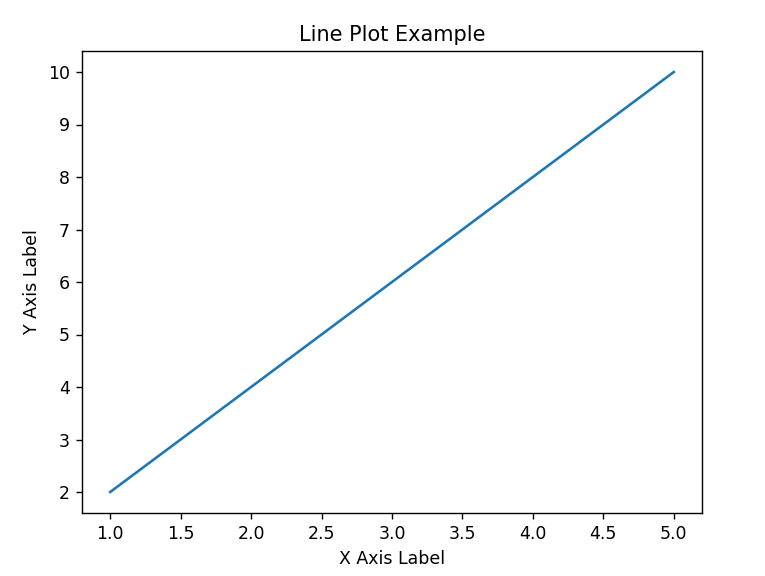
plt.xlabel('X Axis Label')

plt.ylabel('Y Axis Label')

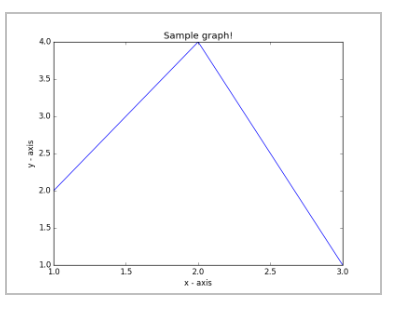
plt.title('Line Plot Example')

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



**Program:**

import matplotlib.pyplot as plt

x = [1,2,3]

y = [2,4,1]

plt.plot(x, y)

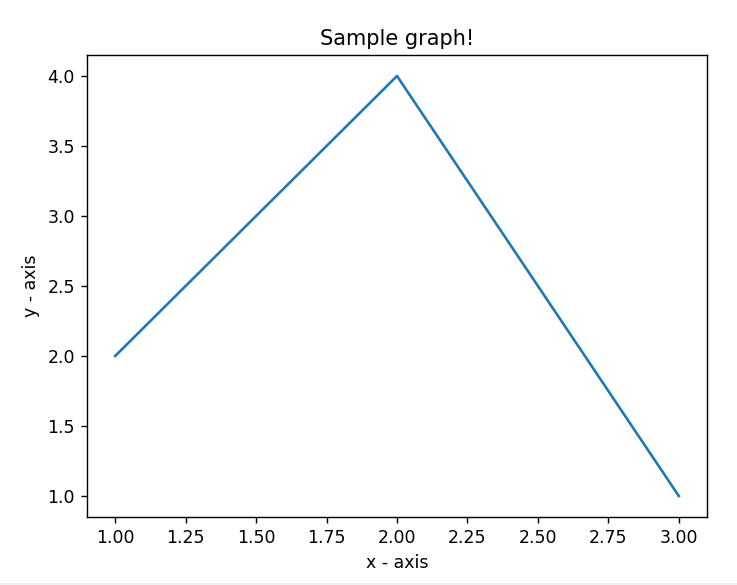
plt.xlabel('x - axis')

plt.ylabel('y - axis')

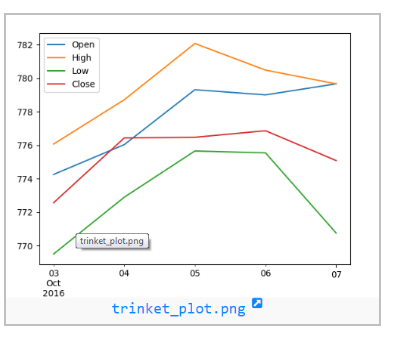
plt.title('Sample graph!')

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



**Program:**

import matplotlib.pyplot as plt

from datetime import datetime

data = [

{"Date": "10-03-16", "Open": 774.25, "High": 776.065002, "Low": 769.5, "Close": 772.559998},

{"Date": "10-04-16", "Open": 776.030029, "High": 778.710022, "Low": 772.890015, "Close": 776.429993},

{"Date": "10-05-16", "Open": 779.309998, "High": 782.070007, "Low": 775.650024, "Close": 776.469971},

{"Date": "10-06-16", "Open": 779, "High": 780.47998, "Low": 775.539978, "Close": 776.859985}, {"Date": "10-07-16", "Open":779.659973, "High": 779.659973, "Low": 770.75, "Close": 775.080017}]

dates = [datetime.strptime(entry["Date"], "%m-%d-%y") for entry in data]

close\_prices = [entry["Close"] for entry in data]

plt.figure(figsize=(10,5))

plt.plot(dates, close\_prices, marker='o', linestyle='-')

plt.xlabel('Date')

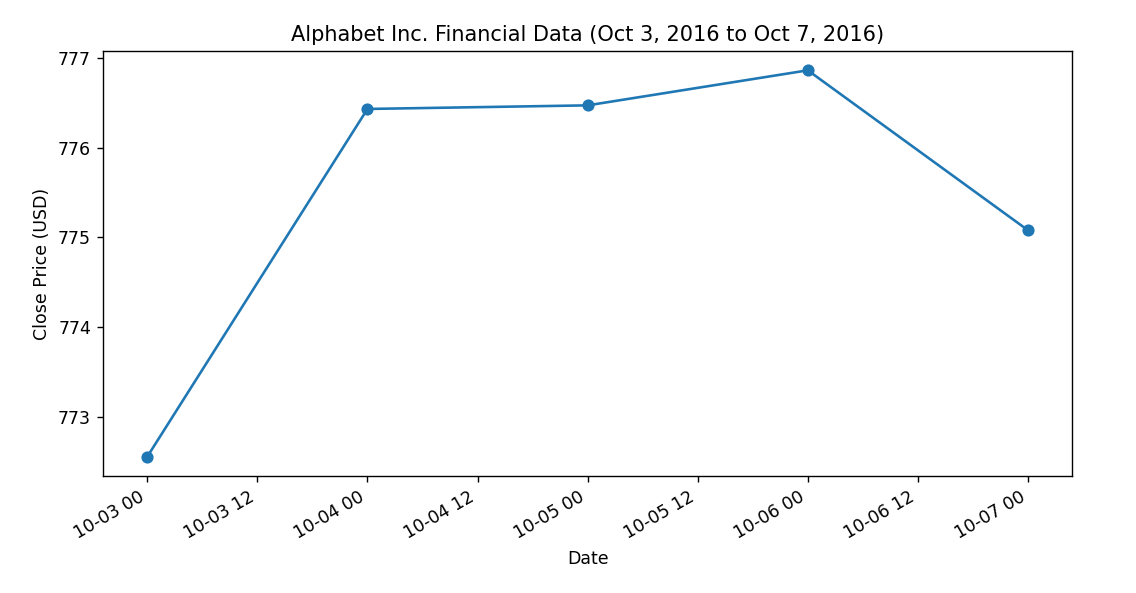
plt.ylabel('Close Price (USD)')

plt.title('Alphabet Inc. Financial Data (Oct 3, 2016 to Oct 7, 2016)')

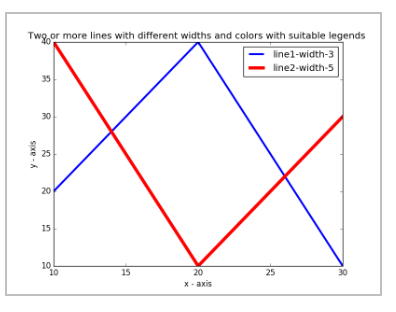
plt.gcf().autofmt\_xdate()

plt.show()

**Output:**



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



**Program:**

import matplotlib.pyplot as plt

x1 = [10,20,30]

y1 = [20,40,10]

x2 = [10,20,30]

y2 = [40,10,30]

plt.xlabel('x - axis')

plt.ylabel('y - axis')

plt.title('Two or more lines with different widths and colors with suitable legends ')

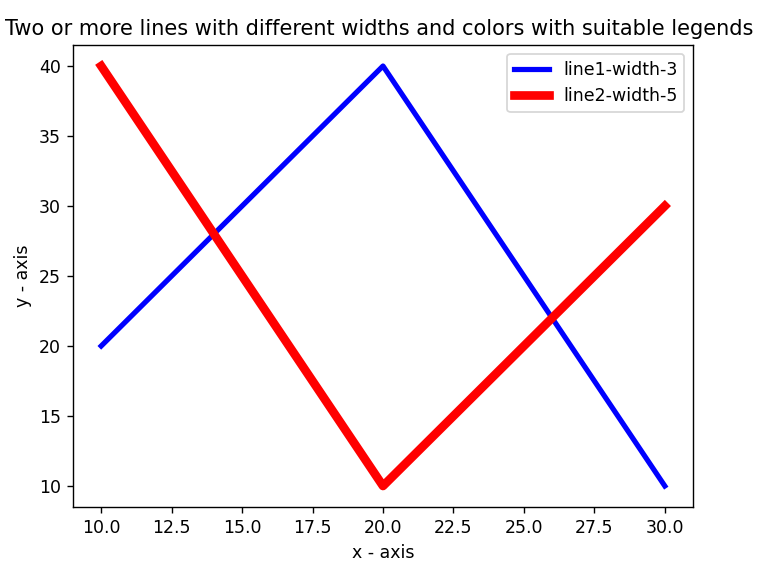
plt.plot(x1,y1, color='blue', linewidth = 3, label = 'line1-width-3')

plt.plot(x2,y2, color='red', linewidth = 5, label = 'line2-width-5')

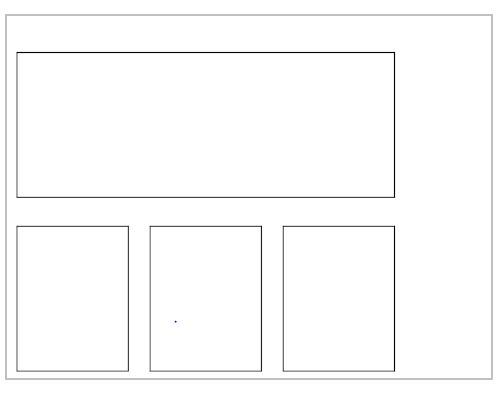
plt.legend()

plt.show()

**Output:**



26.Write a Python program to create multiple plots.



**Program:**

import matplotlib.pyplot as plt

fig = plt.figure()

fig.subplots\_adjust(bottom=0.020, left=0.020, top = 0.900, right=0.800)

plt.subplot(2, 1, 1)

plt.xticks(()), plt.yticks(())

plt.subplot(2, 3, 4)

plt.xticks(())

plt.yticks(())

plt.subplot(2, 3, 5)

plt.xticks(())

plt.yticks(())

plt.subplot(2, 3, 6)

plt.xticks(())

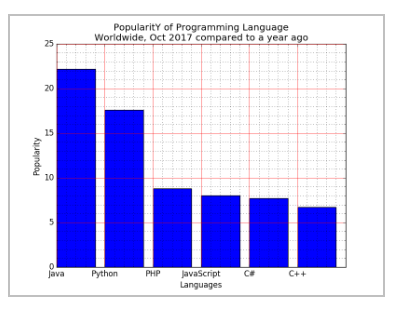
plt.yticks(())

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



**Program:**

import matplotlib.pyplot as plt

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

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

x\_pos = [i for i, \_ in enumerate(x)]

plt.bar(x\_pos, popularity, color='blue')

plt.xlabel("Languages")

plt.ylabel("Popularity")

plt.title("PopularitY of Programming Language\n" + "Worldwide, Oct 2017 compared to a year ago")

plt.xticks(x\_pos, x)

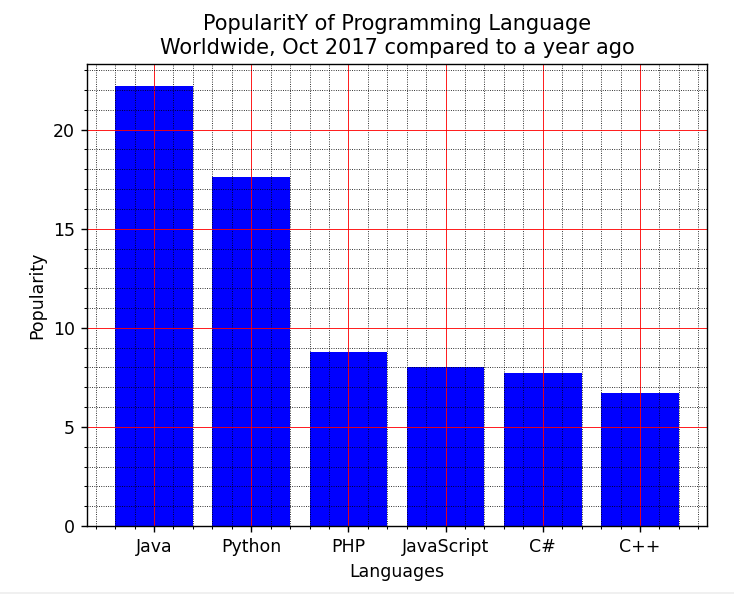
plt.minorticks\_on()

plt.grid(which='major', linestyle='-', linewidth='0.5', color='red')

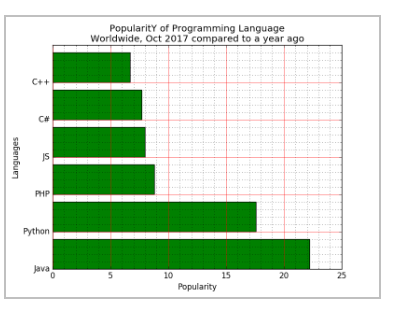
plt.grid(which='minor', linestyle=':', linewidth='0.5', color='black')

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



**Program:**

import matplotlib.pyplot as plt

x = ['Java', 'Python', 'PHP', 'JS', 'C#', 'C++']

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

x\_pos = [i for i, \_ in enumerate(x)]

plt.barh(x\_pos, popularity, color='green')

plt.xlabel("Popularity")

plt.ylabel("Languages")

plt.title("PopularitY of Programming Language\n" + "Worldwide, Oct 2017 compared to a year ago")

plt.yticks(x\_pos, x)

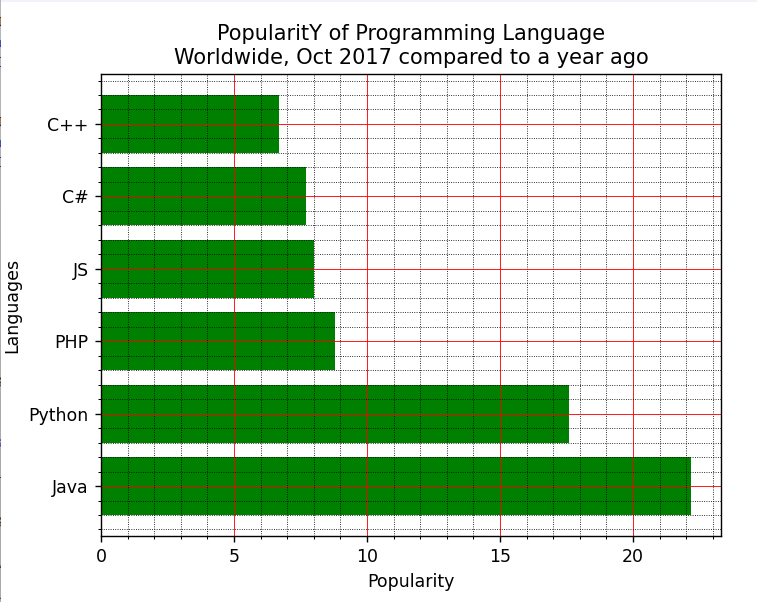
plt.minorticks\_on()

plt.grid(which='major', linestyle='-', linewidth='0.5', color='red')

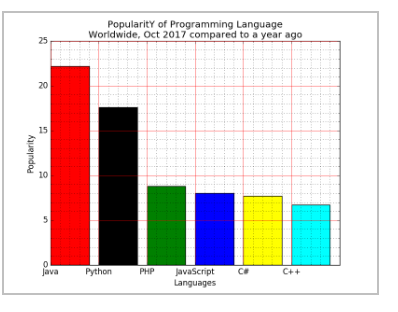
plt.grid(which='minor', linestyle=':', linewidth='0.5', color='black')

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



**Program:**

import matplotlib.pyplot as plt

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

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

x\_pos = [i for i, \_ in enumerate(x)]

plt.bar(x\_pos, popularity, color=['red', 'black', 'green', 'blue', 'yellow', 'cyan'])

plt.xlabel("Languages")

plt.ylabel("Popularity")

plt.title("PopularitY of Programming Language\n" + "Worldwide, Oct 2017 compared to a year ago")

plt.xticks(x\_pos, x)

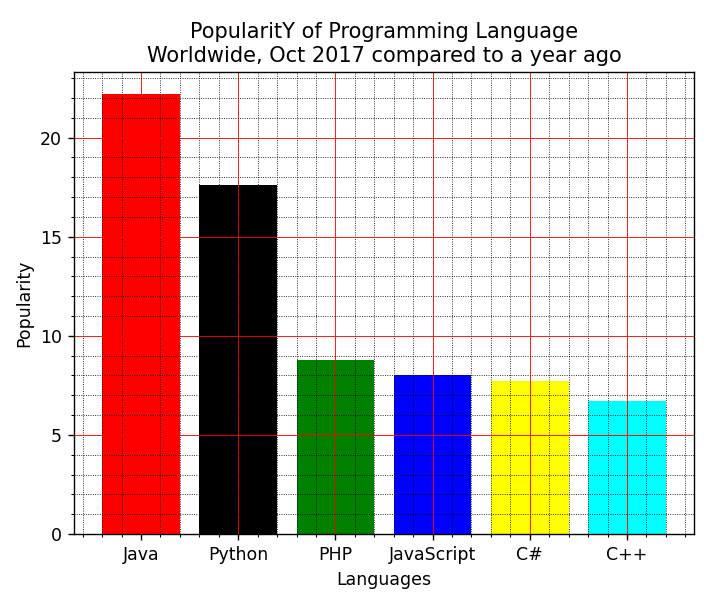
plt.minorticks\_on()

plt.grid(which='major', linestyle='-', linewidth='0.5', color='red')

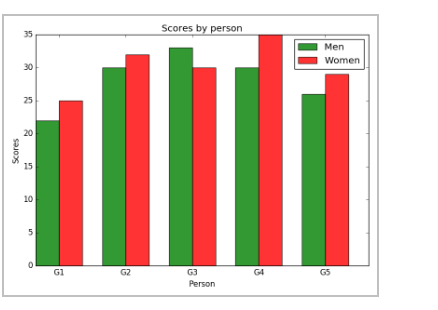
plt.grid(which='minor', linestyle=':', linewidth='0.5', color='black')

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.  
  
Sample Data:  
Means (men) = (22, 30, 35, 35, 26)  
Means (women) = (25, 32, 30, 35, 29)



**Program:**

import numpy as np

import matplotlib.pyplot as plt

n\_groups = 5

men\_means = (22, 30, 33, 30, 26)

women\_means = (25, 32, 30, 35, 29)

fig, ax = plt.subplots()

index = np.arange(n\_groups)

bar\_width = 0.35

opacity = 0.8

rects1 = plt.bar(index, men\_means, bar\_width,

alpha=opacity,

color='g',

label='Men')

rects2 = plt.bar(index + bar\_width, women\_means, bar\_width,

alpha=opacity,

color='r',

label='Women')

plt.xlabel('Person')

plt.ylabel('Scores')

plt.title('Scores by person')

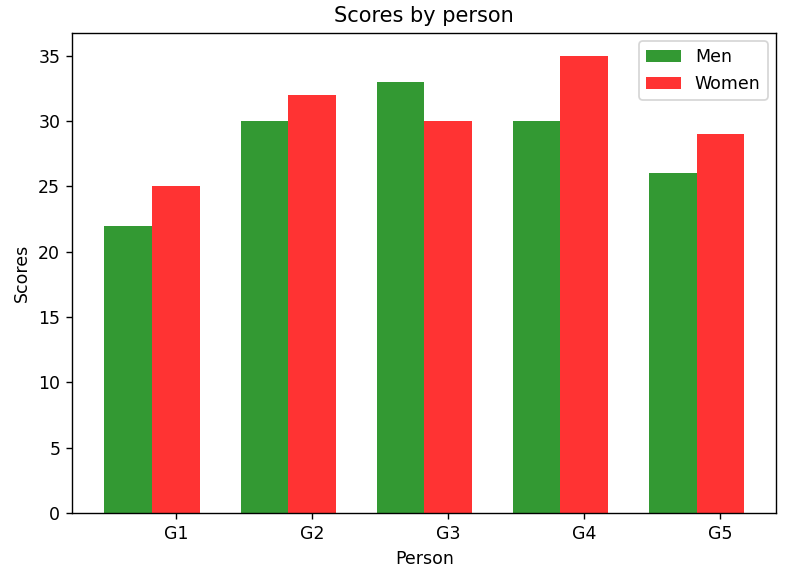
plt.xticks(index + bar\_width, ('G1', 'G2', 'G3', 'G4', 'G5'))

plt.legend()

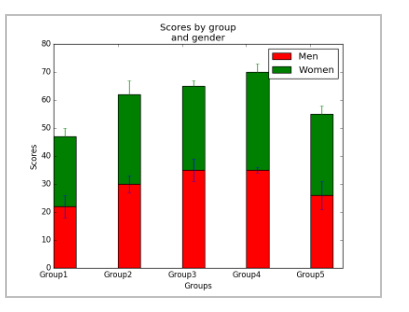
plt.tight\_layout()

plt.show()

**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)



**Program:**

import numpy as np

import matplotlib.pyplot as plt

N = 5

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

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

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

womenStd = (3, 5, 2, 3, 3)

ind = np.arange(N)

width = 0.35

p1 = plt.bar(ind, menMeans, width, yerr=menStd, color='red')

p2 = plt.bar(ind, womenMeans, width,

bottom=menMeans, yerr=womenStd, color='green')

plt.ylabel('Scores')

plt.xlabel('Groups')

plt.title('Scores by group\n' + 'and gender')

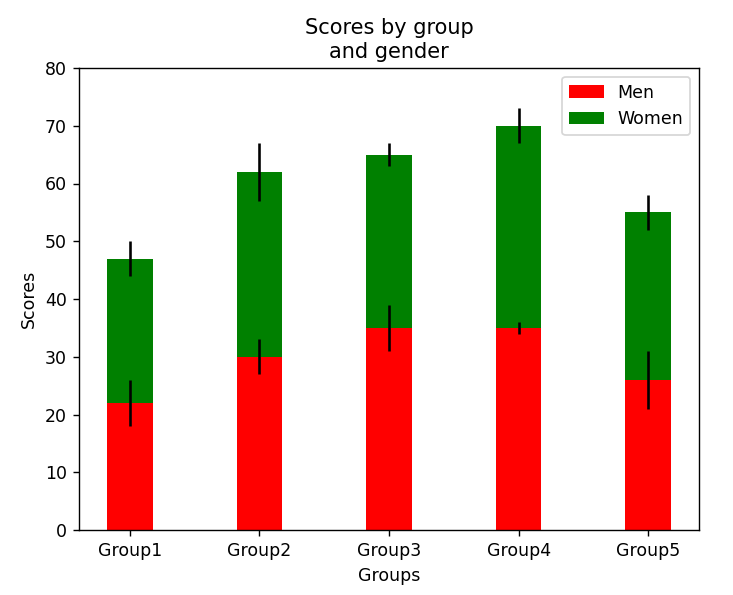
plt.xticks(ind, ('Group1', 'Group2', 'Group3', 'Group4', 'Group5'))

plt.yticks(np.arange(0, 81, 10))

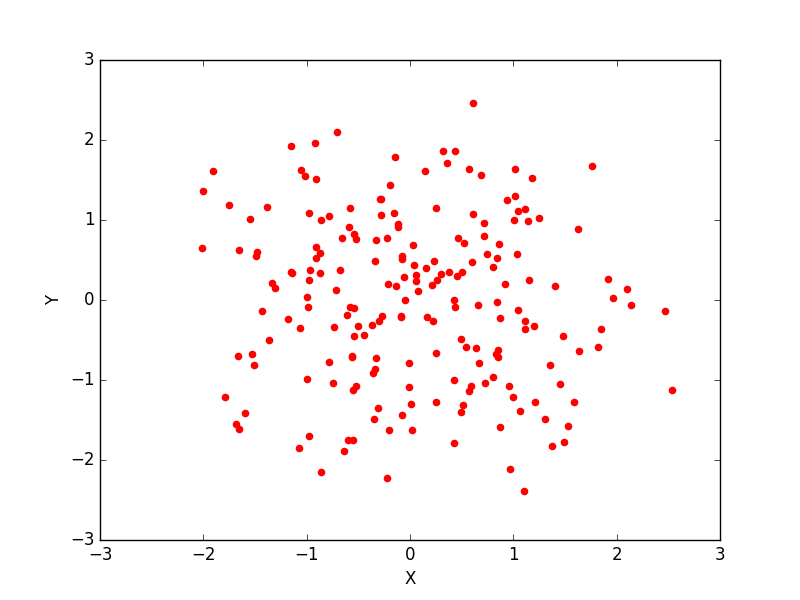
plt.legend((p1[0], p2[0]), ('Men', 'Women'))

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.



**Program:**

import matplotlib.pyplot as plt

import numpy as np

np.random.seed(0)

n\_points = 50

x = np.random.rand(n\_points)

y = np.random.rand(n\_points)

plt.scatter(x, y, label='Random Data', color='red', marker='o')

plt.xlabel('X')

plt.ylabel('Y')

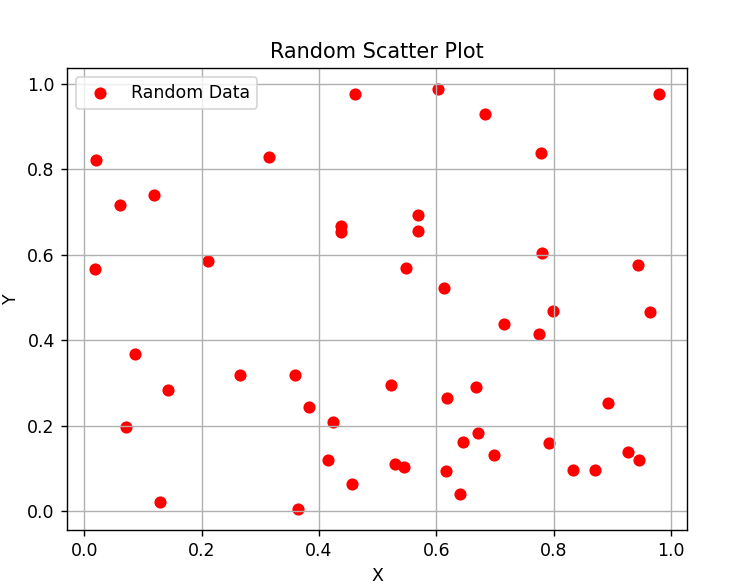
plt.title('Random Scatter Plot')

plt.legend()

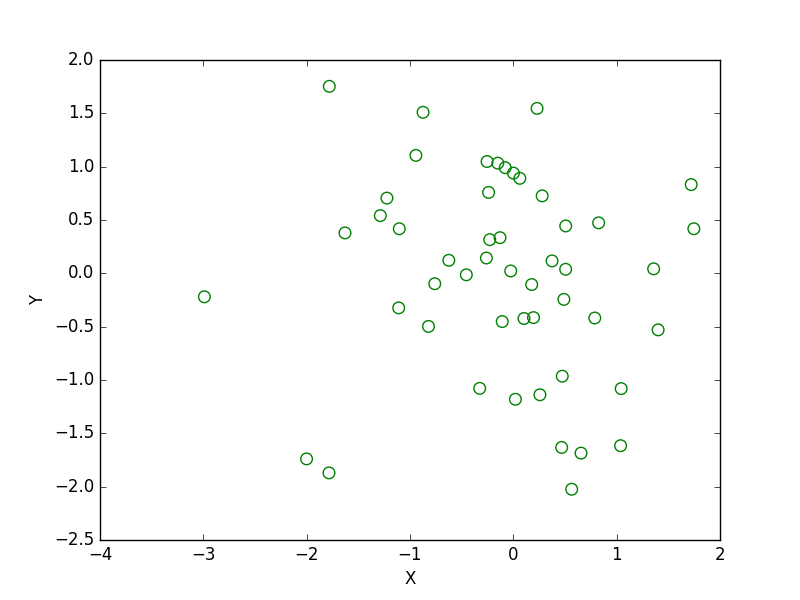
plt.grid(True)

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.



**Program:**

import matplotlib.pyplot as plt

import numpy as np

x = np.random.randn(50)

y = np.random.randn(50)

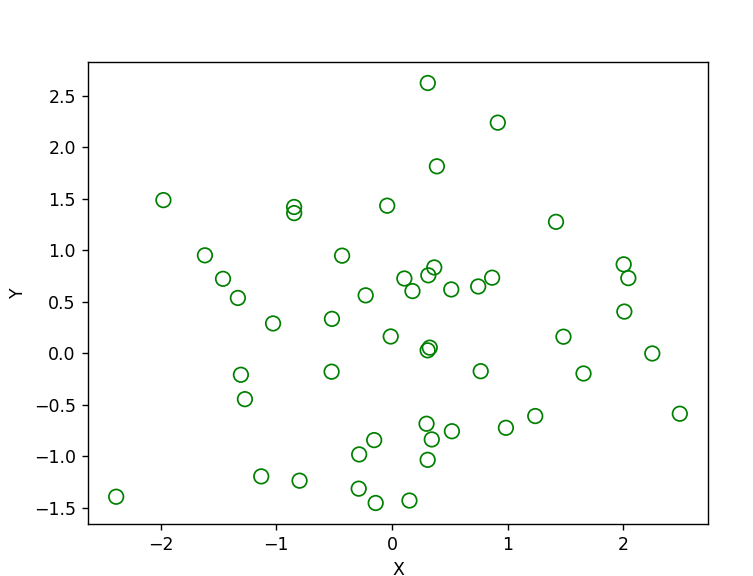
plt.scatter(x, y, s=70, facecolors='none', edgecolors='g')

plt.xlabel("X")

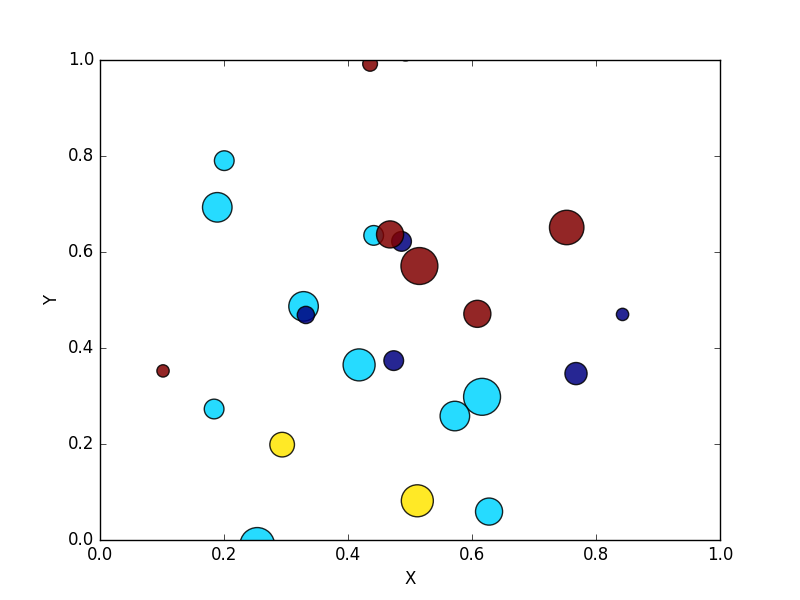
plt.ylabel("Y")

plt.show()

**Output:**



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



**Program:**

import math

import random

import matplotlib.pyplot as plt

no\_of\_balls = 25

x = [random.triangular() for i in range(no\_of\_balls)]

y = [random.gauss(0.5, 0.25) for i in range(no\_of\_balls)]

colors = [random.randint(1, 4) for i in range(no\_of\_balls)]

areas = [math.pi \* random.randint(5, 15)\*\*2 for i in range(no\_of\_balls)]

plt.figure()

plt.scatter(x, y, s=areas, c=colors, alpha=0.85)

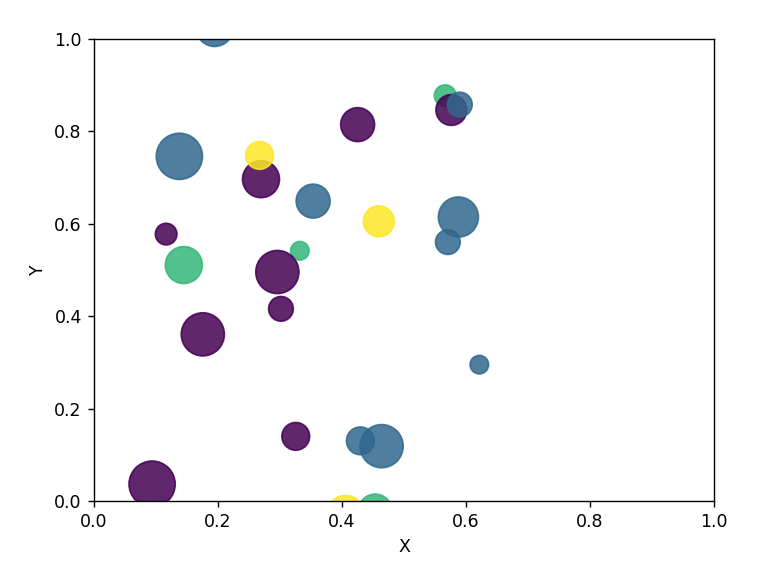
plt.axis([0.0, 1.0, 0.0, 1.0])

plt.xlabel("X")

plt.ylabel("Y")

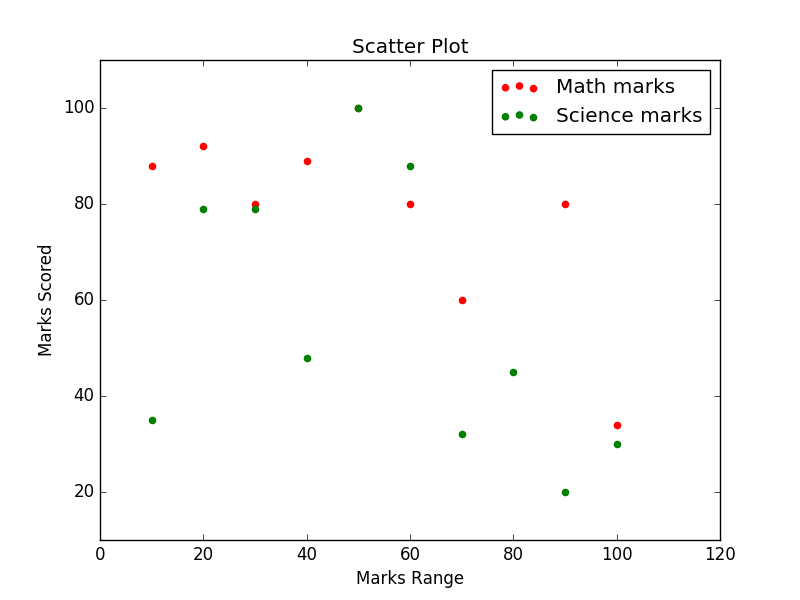
plt.show()

**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]



**Program:**

import matplotlib.pyplot as plt

import pandas as pd

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.scatter(marks\_range, math\_marks, label='Math marks', color='r')

plt.scatter(marks\_range, science\_marks, label='Science marks', color='g')

plt.title('Scatter Plot')

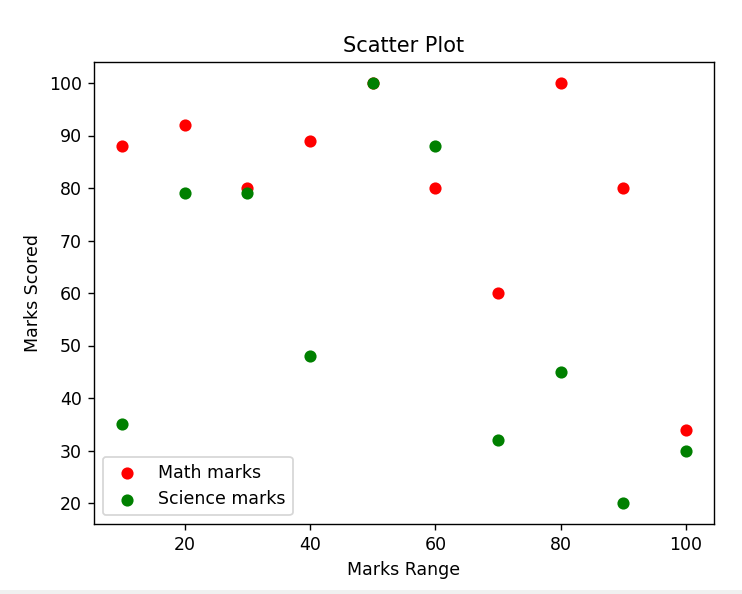
plt.xlabel('Marks Range')

plt.ylabel('Marks Scored')

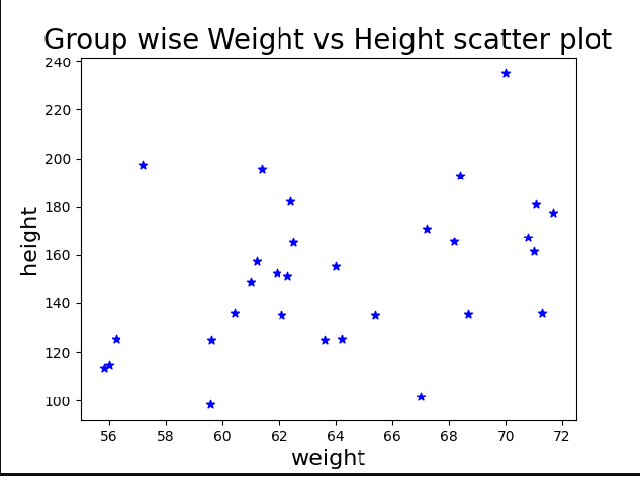
plt.legend()

plt.show()

**Output:**



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



**Program:**

import matplotlib.pyplot as plt

import numpy as np

weight1=[67,57.2,59.6,59.64,55.8,61.2,60.45,61,56.23,56]

height1=[101.7,197.6,98.3,125.1,113.7,157.7,136,148.9,125.3,114.9]

weight2=[61.9,64,62.1,64.2,62.3,65.4,62.4,61.4,62.5,63.6]

height2=[152.8,155.3,135.1,125.2,151.3,135,182.2,195.9,165.1,125.1]

weight3=[68.2,67.2,68.4,68.7,71,71.3,70.8,70,71.1,71.7]

height3=[165.8,170.9,192.8,135.4,161.4,136.1,167.1,235.1,181.1,177.3]

weight=np.concatenate((weight1,weight2,weight3))

height=np.concatenate((height1,height2,height3))

plt.scatter(weight, height, marker='\*', color=['blue'])

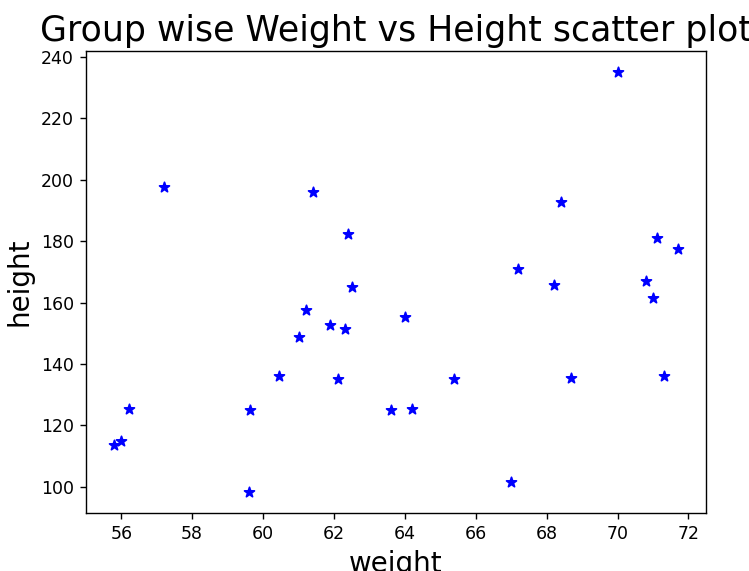
plt.xlabel('weight', fontsize=16)

plt.ylabel('height', fontsize=16)

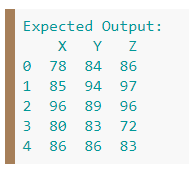
plt.title('Group wise Weight vs Height scatter plot',fontsize=20)

plt.show()

**Output:**



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]}



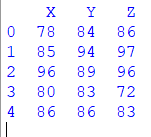
**Program:**

import pandas as pd

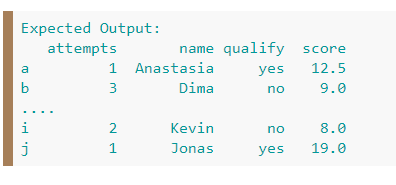
df = pd.DataFrame({'X':[78,85,96,80,86], 'Y':[84,94,89,83,86],'Z':[86,97,96,72,83]});

print(df)

**Output:**



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']



**Program:**

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],

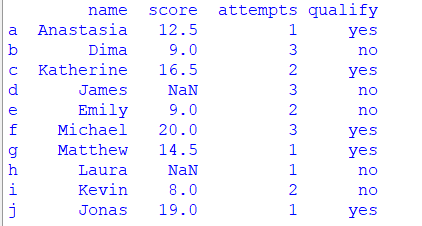
'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}

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

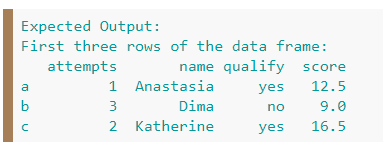
df = pd.DataFrame(exam\_data , index=labels)

print(df)

**Output:**



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']



**Program:**

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']}

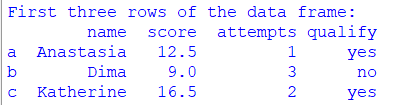
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

df = pd.DataFrame(exam\_data , index=labels)

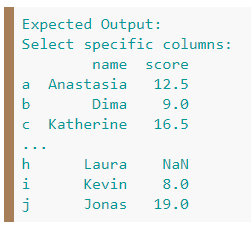
print("First three rows of the data frame:")

print(df.iloc[:3])

**Output:**



1. 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']



**Program:**

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("Select specific columns:")

print(df[['name', 'score']])

**Output:**

