LAB PROBLEMS

In [ ]:

# QUERY PROCESSING FOR DATA SCIENCE WITH DISTRIBUTED COMPUTING

# DSA0501

In [1]:

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

import pandas as pd

*# Create a DataFrame from the provided data*

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)

print(df)

*# Select distinct department IDs*

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

*# Print the distinct department IDs*

print("distinct\_department\_ids")

print(distinct\_department\_ids)

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

0 10 Administration 200 1700

1 20 Marketing 201 1800

2 30 Purchasing 114 1700

3 40 Human Resources 203 2400

4 50 Shipping 121 1500

5 60 IT 103 1400

6 70 Public Relations 204 2700

7 80 Sales 145 2500

8 90 Executive 100 1700

9 100 Finance 108 1700

10 110 Accounting 205 1700

11 120 Treasury 0 1700

12 130 Corporate Tax 0 1700

13 140 Control And Credit 0 1700

14 150 Shareholder Services 0 1700

15 160 Benefits 0 1700

16 170 Manufacturing 0 1700

17 180 Construction 0 1700

18 190 Contracting 0 1700

19 200 Operations 0 1700

20 210 IT Support 0 1700

21 220 NOC 0 1700

22 230 IT Helpdesk 0 1700

23 240 Government Sales 0 1700

24 250 Retail Sales 0 1700

25 260 Recruiting 0 1700

26 270 Payroll 0 1700

distinct\_department\_ids

[ 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]

In [2]:

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

import pandas as pd

*# Create a DataFrame from the provided data*

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)

print(df)

*# Group the data by EMPLOYEE\_ID and count the number of unique JOB\_IDs for each employee*

employee\_job\_counts = df.groupby('EMPLOYEE\_ID')['JOB\_ID'].nunique()

*# Filter for employees who did two or more jobs*

result = employee\_job\_counts[employee\_job\_counts >= 2]

*# Print the result*

print(result)

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

0 102 2001-01-13 2006-07-24 IT\_PROG 60

1 101 1997-09-21 2001-10-27 AC\_ACCOUNT 110

2 101 2001-10-28 2005-03-15 AC\_MGR 110

3 201 2004-02-17 2007-12-19 MK\_REP 20

4 114 2006-03-24 2007-12-31 ST\_CLERK 50

5 122 2007-01-01 2007-12-31 ST\_CLERK 50

6 200 1995-09-17 2001-06-17 AD\_ASST 90

7 176 2006-03-24 2006-12-31 SA\_REP 80

8 176 2007-01-01 2007-12-31 SA\_MAN 80

9 200 2002-07-01 2006-12-31 AC\_ACCOUNT 90

EMPLOYEE\_ID

101 2

176 2

200 2

Name: JOB\_ID, dtype: int64

In [3]:

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

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

import pandas as pd

*# Create a DataFrame from the provided data*

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("Orginal dataframe:\n",df)

*# Sort the DataFrame in descending order based on the 'JOB\_TITLE' column*

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

*# Print the sorted DataFrame*

print("Sorted dataframe:\n",df\_sorted)

Orginal dataframe:

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

0 AD\_PRES President 20080 40000

1 AD\_VP Administration Vice President 15000 30000

2 AD\_ASST Administration Assistant 3000 6000

3 FI\_MGR Finance Manager 8200 16000

4 FI\_ACCOUNT Accountant 4200 9000

5 AC\_MGR Accounting Manager 8200 16000

6 AC\_ACCOUNT Public Accountant 4200 9000

7 SA\_MAN Sales Manager 10000 20080

8 SA\_REP Sales Representative 6000 12008

9 PU\_MAN Purchasing Manager 8000 15000

10 PU\_CLERK Purchasing Clerk 2500 5500

11 ST\_MAN Stock Manager 5500 8500

12 ST\_CLERK Stock Clerk 2008 5000

13 SH\_CLERK Shipping Clerk 2500 5500

14 IT\_PROG Programmer 4000 10000

15 MK\_MAN Marketing Manager 9000 15000

16 MK\_REP Marketing Representative 4000 9000

17 HR\_REP Human Resources Representative 4000 9000

18 PR\_REP Public Relations Representative 4500 10500

Sorted dataframe:

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

In [6]:

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

import pandas as pd

import matplotlib.pyplot as plt

*# Load historical stock price data from the provided CSV file path*

file\_path = r"/content/GoogleStockPrices.csv" *# Replace with your file path*

df = pd.read\_csv(file\_path)

*# Assuming your CSV has 'Date' and 'Close' columns, you may need to adjust column names*

*# Make sure the 'Date' column is in datetime format*

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

*# Set the date range you want to plot*

start\_date = '2004-08-19'

end\_date = '2021-10-09'

*# Filter the data for the specified date range*

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

*# Create a line plot*

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

plt.plot(filtered\_df['Date'], filtered\_df['Close'], label='Alphabet Inc. Stock Price')

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

plt.xlabel('Date')

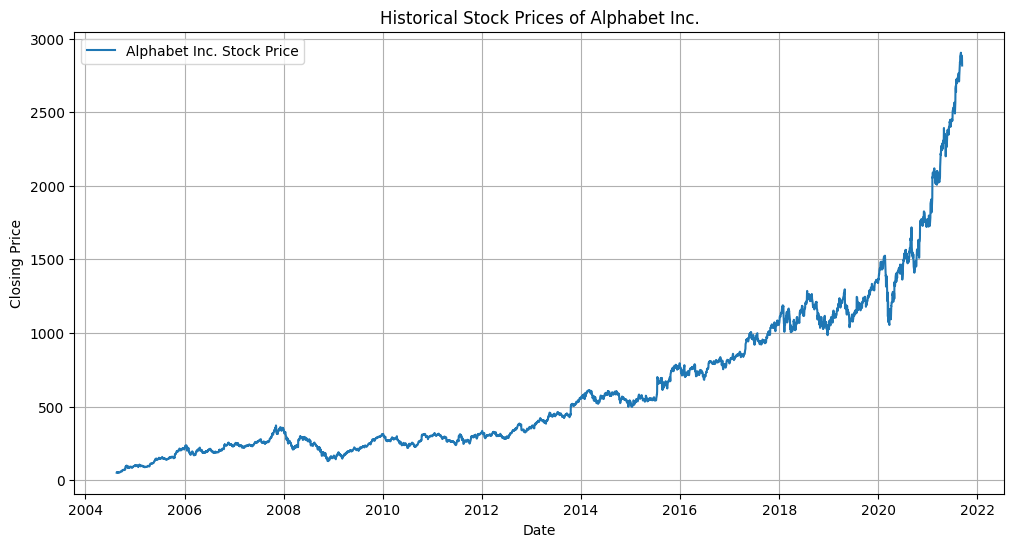
plt.ylabel('Closing Price')

plt.legend()

plt.grid(True)

*# Show the plot*

plt.show()

**

In [7]:

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

import pandas as pd

import matplotlib.pyplot as plt

*# Load historical stock price data from the provided CSV file path*

file\_path = r"/content/GoogleStockPrices.csv" *# Replace with your file path*

df = pd.read\_csv(file\_path)

*# Assuming your CSV has 'Date' and 'Volume' columns, you may need to adjust column names*

*# Make sure the 'Date' column is in datetime format*

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

*# Set the date range you want to plot*

start\_date = '2004-08-19'

end\_date = '2021-10-09'

*# Filter the data for the specified date range*

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

*# Create a bar plot for trading volume*

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

plt.bar(filtered\_df['Date'], filtered\_df['Volume'], color='royalblue', label='Trading Volume')

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

plt.xlabel('Date')

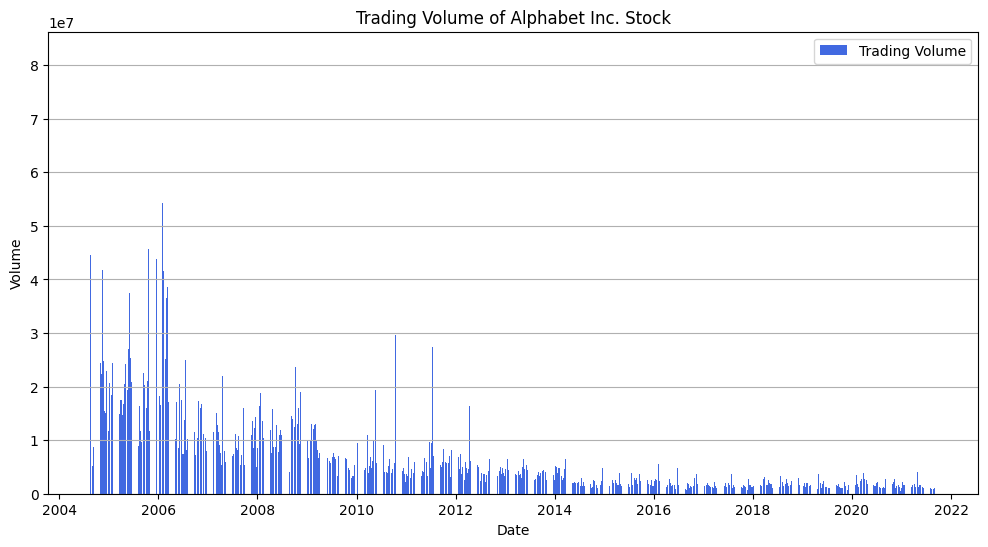
plt.ylabel('Volume')

plt.legend()

plt.grid(axis='y')

*# Show the plot*

plt.show()

**

In [8]:

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

import pandas as pd

import matplotlib.pyplot as plt

*# Load historical stock price data from the provided CSV file path*

file\_path = r"/content/GoogleStockPrices.csv" *# Replace with your file path*

df = pd.read\_csv(file\_path)

*# Assuming your CSV has 'Date', 'Volume', and 'Close' columns, you may need to adjust column names*

*# Make sure the 'Date' column is in datetime format*

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

*# Set the date range you want to plot*

start\_date = '2021-04-01'

end\_date = '2021-05-01'

*# Filter the data for the specified date range*

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

*# Create a scatter plot for trading volume vs. stock prices*

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

plt.scatter(filtered\_df['Volume'], filtered\_df['Close'], color='royalblue', label='Trading Volume vs. Stock Price')

plt.title('Scatter Plot of Trading Volume vs. Stock Prices of Alphabet Inc. Stock')

plt.xlabel('Volume')

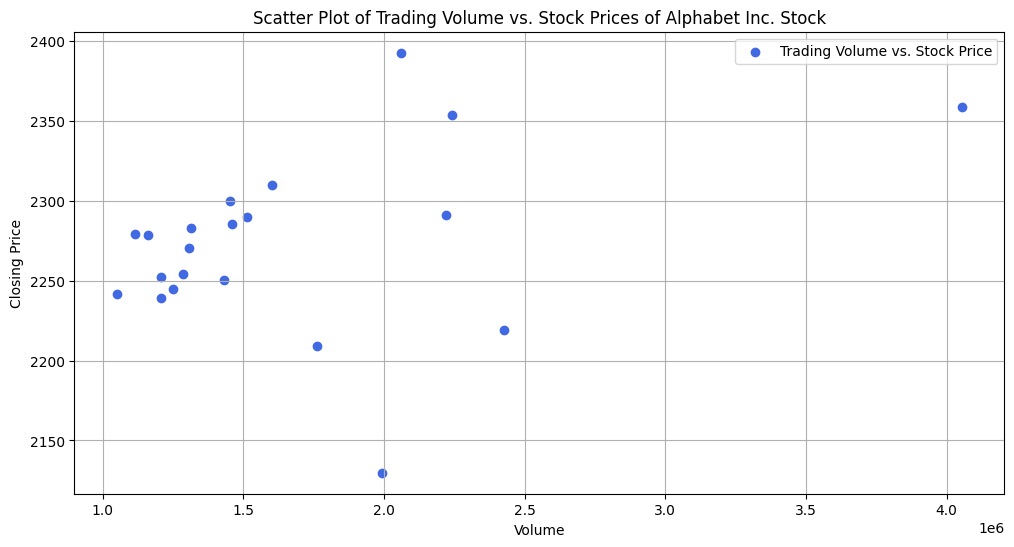
plt.ylabel('Closing Price')

plt.legend()

plt.grid(True)

*# Show the plot*

plt.show()

**

In [9]:

*#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)*

import pandas as pd

*# Create a DataFrame from the provided data*

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', 'Hermann', 'Martha', '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': ['1,13,810.00', '25,000.00', '43,128.00', '6,075.00', '67,088.00', '30,000.00', '89,850.00', '1,07,820.00', '38,336.00', '30,000.00', '1,07,820.00', '14,500.00', '40,500.00', '41,930.00', '250.00', '936.00', '14,000.00', '14,400.00']

}

df = pd.DataFrame(data)

*# Convert 'Sale\_amt' column to numeric by removing commas*

df['Sale\_amt'] = df['Sale\_amt'].str.replace(',', '', regex=True).astype(float)

*# Create a Pivot table to find the maximum and minimum sale values*

pivot\_table = pd.pivot\_table(df, values='Sale\_amt', index='Item', aggfunc=['max', 'min'])

*# Extract the maximum and minimum sale values*

max\_sale\_values = pivot\_table['max']['Sale\_amt']

min\_sale\_values = pivot\_table['min']['Sale\_amt']

*# Print the maximum and minimum sale values for each item*

print("Maximum Sale Values:")

print(max\_sale\_values)

print("\nMinimum Sale Values:")

print(min\_sale\_values)

Maximum Sale Values:

Item

Cell Phone 14400.0

Desk 250.0

Home Theater 40500.0

Television 113810.0

Video Games 936.0

Name: Sale\_amt, dtype: float64

Minimum Sale Values:

Item

Cell Phone 6075.0

Desk 250.0

Home Theater 14000.0

Television 38336.0

Video Games 936.0

Name: Sale\_amt, dtype: float64

In [10]:

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

import pandas as pd

*# Create a DataFrame from the provided data*

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', 'Hermann', 'Martha', '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': ['1,13,810.00', '25,000.00', '43,128.00', '6,075.00', '67,088.00', '30,000.00', '89,850.00', '1,07,820.00', '38,336.00', '30,000.00', '1,07,820.00', '14,500.00', '40,500.00', '41,930.00', '250.00', '936.00', '14,000.00', '14,400.00']

}

df = pd.DataFrame(data)

*# Convert 'Sale\_amt' column to numeric by removing commas*

df['Sale\_amt'] = df['Sale\_amt'].str.replace(',', '', regex=True).astype(float)

*# Create a Pivot table to find the item-wise unit sold*

pivot\_table = pd.pivot\_table(df, values='Units', index='Item', aggfunc='sum')

*# Print the item-wise unit sold*

print("Item-Wise Unit Sold:")

print(pivot\_table)

Item-Wise Unit Sold:

Units

Item

Cell Phone 91

Desk 2

Home Theater 308

Television 509

Video Games 16

In [11]:

*#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)*

import pandas as pd

*# Create a DataFrame from the provided data*

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', 'Hermann', 'Martha', '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': ['1,13,810.00', '25,000.00', '43,128.00', '6,075.00', '67,088.00', '30,000.00', '89,850.00', '1,07,820.00', '38,336.00', '30,000.00', '1,07,820.00', '14,500.00', '40,500.00', '41,930.00', '250.00', '936.00', '14,000.00', '14,400.00']

}

df = pd.DataFrame(data)

*# Convert 'Sale\_amt' column to numeric by removing commas*

df['Sale\_amt'] = df['Sale\_amt'].str.replace(',', '', regex=True).astype(float)

*# Create a Pivot table to find the total sale amount region-wise, manager-wise, and salesperson-wise*

pivot\_table = pd.pivot\_table(df, values='Sale\_amt', index=['Region', 'Manager', 'SalesMan'], aggfunc='sum')

*# Print the total sale amount for each region, manager, and salesperson*

print("Total Sale Amount (Region, Manager, SalesMan):")

print(pivot\_table)

Total Sale Amount (Region, Manager, SalesMan):

Sale\_amt

Region Manager SalesMan

Central Douglas John 250.0

Hermann Luis 150948.0

Shelli 25000.0

Sigal 107820.0

Martha Sigal 14000.0

Steven 89850.0

Timothy David 6075.0

East Douglas Karen 40500.0

Hermann Alexander 936.0

Martha Alexander 230140.0

Diana 14500.0

West Douglas Michael 38336.0

Timothy Stephen 67088.0

In [13]:

*#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.*

import pandas as pd

import random

*# Create the DataFrame with the provided data*

data = {

'A': range(10),

'B': [1.32921, -1.07082, -1.6264, 0.961538, 1.45342, -1.33694, 0.121668, 0.354493, 1.68658, -0.12982],

'C': [-0.770033, -1.43871, 0.219565, 0.104011, 1.05774, 0.562861, 1.2076, 1.03753, -1.32596, 0.631523],

'D': [-0.31628, 0.564417, 0.678805, -0.481165, 0.165562, 1.39285, -0.00204021, -0.385684, 1.42898, -0.586538],

'E': [-0.99081, 0.295722, 1.88927, 0.850229, 0.515018, -0.063328, 1.6278, 0.519818, -2.08935, 0.29072]

}

df = pd.DataFrame(data)

*# Function to format cell colors*

def color\_negative\_red(val):

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

return f'color: {color}'

*# Apply the color function to the entire DataFrame*

styled\_df = df.style.applymap(color\_negative\_red, subset=df.columns[1:])

*# Display the styled DataFrame*

styled\_df

Out[13]:

|  | **A** | **B** | **C** | **D** | **E** |
| --- | --- | --- | --- | --- | --- |
| **0** | 0 | 1.329210 | -0.770033 | -0.316280 | -0.990810 |
| **1** | 1 | -1.070820 | -1.438710 | 0.564417 | 0.295722 |
| **2** | 2 | -1.626400 | 0.219565 | 0.678805 | 1.889270 |
| **3** | 3 | 0.961538 | 0.104011 | -0.481165 | 0.850229 |
| **4** | 4 | 1.453420 | 1.057740 | 0.165562 | 0.515018 |
| **5** | 5 | -1.336940 | 0.562861 | 1.392850 | -0.063328 |
| **6** | 6 | 0.121668 | 1.207600 | -0.002040 | 1.627800 |
| **7** | 7 | 0.354493 | 1.037530 | -0.385684 | 0.519818 |
| **8** | 8 | 1.686580 | -1.325960 | 1.428980 | -2.089350 |
| **9** | 9 | -0.129820 | 0.631523 | -0.586538 | 0.290720 |

*#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.*

import pandas as pd

import numpy as np

*# Creating a DataFrame with random values*

data = {

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

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

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

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

}

df = pd.DataFrame(data)

*# Introducing NaN values at random positions*

nan\_indices = [(row, col) for row in range(10) for col in np.random.choice(range(4), 2)]

for row, col in nan\_indices:

df.iloc[row, col] = np.nan

*# Function to highlight NaN values*

def highlight\_nan(value):

if pd.isna(value):

return 'background-color:red' *# Applying yellow background to NaN values*

else:

return ''

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

styled\_df

Out[ ]:

|  | **A** | **B** | **C** | **D** |
| --- | --- | --- | --- | --- |
| **0** | nan | 0.109139 | 0.929502 | nan |
| **1** | 0.904271 | nan | 0.896683 | 0.986374 |
| **2** | nan | 0.917590 | 0.833670 | 0.031627 |
| **3** | nan | 0.127309 | 0.144332 | nan |
| **4** | nan | nan | 0.465524 | 0.823800 |
| **5** | 0.374063 | 0.491179 | nan | nan |
| **6** | nan | nan | 0.197088 | 0.308790 |
| **7** | 0.982039 | 0.040280 | 0.603822 | nan |
| **8** | 0.630489 | 0.947055 | 0.565202 | nan |
| **9** | nan | 0.469494 | nan | 0.661527 |

In [ ]:

*#11.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.*

import pandas as pd

*# Create the DataFrame*

data = {

'A': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],

'B': [1.32921, -1.07082, -1.6264, 0.961538, 1.45342, -1.33694, 0.121668, 0.354493, 1.68658, -0.12982],

'C': [-0.770033, -1.43871, 0.219565, 0.104011, 1.05774, 0.562861, 1.2076, 1.03753, -1.32596, 0.631523],

'D': [-0.31628, 0.564417, 0.678805, -0.481165, 0.165562, 1.39285, -0.00204021, -0.385684, 1.42898, -0.586538],

'E': [-0.99081, 0.295722, 1.88927, 0.850229, 0.515018, -0.063328, 1.6278, 0.519818, -2.08935, 0.29072]

}

df = pd.DataFrame(data)

*# Define a function to apply color based on value*

def color\_negative\_red(val):

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

return f'color: {color}'

*# Apply the style to the DataFrame*

styled\_df = df.style.applymap(color\_negative\_red, subset=pd.IndexSlice[:, 'B':'E'])

*# Display the styled DataFrame*

styled\_df

Out[ ]:

|  | **A** | **B** | **C** | **D** | **E** |
| --- | --- | --- | --- | --- | --- |
| **0** | 1 | 1.329210 | -0.770033 | -0.316280 | -0.990810 |
| **1** | 2 | -1.070820 | -1.438710 | 0.564417 | 0.295722 |
| **2** | 3 | -1.626400 | 0.219565 | 0.678805 | 1.889270 |
| **3** | 4 | 0.961538 | 0.104011 | -0.481165 | 0.850229 |
| **4** | 5 | 1.453420 | 1.057740 | 0.165562 | 0.515018 |
| **5** | 6 | -1.336940 | 0.562861 | 1.392850 | -0.063328 |
| **6** | 7 | 0.121668 | 1.207600 | -0.002040 | 1.627800 |
| **7** | 8 | 0.354493 | 1.037530 | -0.385684 | 0.519818 |
| **8** | 9 | 1.686580 | -1.325960 | 1.428980 | -2.089350 |
| **9** | 10 | -0.129820 | 0.631523 | -0.586538 | 0.290720 |

In [ ]:

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

import pandas as pd

import numpy as np

*# Creating a DataFrame with random values*

data = np.random.rand(10, 4)

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

*# Define a function to set background color to black and font color to yellow*

def set\_colors(val):

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

*# Apply the style to the DataFrame*

styled\_df = df.style.applymap(set\_colors)

*# Display the styled DataFrame*

styled\_df

Out[ ]:

|  | **A** | **B** | **C** | **D** |
| --- | --- | --- | --- | --- |
| **0** | 0.949229 | 0.708477 | 0.797638 | 0.248989 |
| **1** | 0.874047 | 0.828828 | 0.994215 | 0.230815 |
| **2** | 0.302407 | 0.444712 | 0.589820 | 0.494598 |
| **3** | 0.477678 | 0.874770 | 0.696375 | 0.330870 |
| **4** | 0.475231 | 0.140523 | 0.973171 | 0.718913 |
| **5** | 0.922042 | 0.629859 | 0.204268 | 0.605826 |
| **6** | 0.611342 | 0.205634 | 0.017364 | 0.747633 |
| **7** | 0.967213 | 0.388298 | 0.999878 | 0.804939 |
| **8** | 0.267222 | 0.396534 | 0.764169 | 0.712393 |
| **9** | 0.021338 | 0.205826 | 0.969449 | 0.792637 |

In [ ]:

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

import pandas as pd

import numpy as np

*# Creating the DataFrame with the provided values*

data = {

'ord\_no': [70001.0, np.nan, 70002.0, 70004.0, np.nan, 70005.0, np.nan, 70010.0, 70012.0, np.nan, 70013.0],

'purch\_amt': [150.50, 270.65, np.nan, 110.50, 948.50, 2400.60, 5760.00, np.nan, 2480.40, 75.29, 3045.60],

'ord\_date': ['2012-10-05', '2012-09-10', '2012-08-17', np.nan, '2012-09-10', '2012-07-27', '2012-09-10', np.nan, '2012-10-10', '2012-08-171', '2012-04-25'],

'customer\_id': [3002, 3001, 3003, 3002, 3001, 3001, 3001, 3003, 3002, 3001, 3001],

'salesman\_id': [5002.0, 5003.0, np.nan, np.nan, 5002.0, 5001.0, 5001.0, 5003.0, 5003.0, 5003.0, np.nan]

}

df = pd.DataFrame(data)

*# Detect missing values and display True or False*

missing\_values = df.isnull()

*# Display the DataFrame of True/False values for missing data*

print(missing\_values)

ord\_no purch\_amt ord\_date customer\_id salesman\_id

0 False False False False False

1 True False False False False

2 False True False False True

3 False False True False True

4 True False False False False

5 False False False False False

6 True False False False False

7 False True True False False

8 False False False False False

9 True False False False False

10 False False False False True

In [ ]:

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

import pandas as pd

import numpy as np

*# Creating the DataFrame with the provided data*

data = {

'ord\_no': [70001, np.nan, 70002, 70004, 70005, 70010, 70003, 70012, np.nan, 70013],

'purch\_amt': [150.5, 270.65, 65.26, 110.5, 2400.6, np.nan, 12.43, 2480.4, np.nan, 3045.6],

'ord\_date': ['2012-09-10', '5003', np.nan, '2012-08-17', '2012-07-27', '2012-10-10', '2012-10-10', '2012-06-27', '2012-08-17', '2012-04-25'],

'customer\_id': [3002, 3001, np.nan, 3003, 3001, 3003, 3002, 3001, 3001, np.nan],

'salesman\_id': [5002, 3001, 5001, np.nan, 5001, 5003, 5002, 5003, 3001, np.nan]

}

df = pd.DataFrame(data)

*# Define a function to replace the values that do not contain valuable information*

def replace\_missing(val):

if val == '?' or pd.isnull(val):

return np.nan *# Replace '?' and NaN with NaN*

return val

*# Apply the function to replace the values*

df = df.applymap(replace\_missing)

*# Display the DataFrame after replacing missing values*

print(df)

ord\_no purch\_amt ord\_date customer\_id salesman\_id

0 70001.0 150.50 2012-09-10 3002.0 5002.0

1 NaN 270.65 5003 3001.0 3001.0

2 70002.0 65.26 NaN NaN 5001.0

3 70004.0 110.50 2012-08-17 3003.0 NaN

4 70005.0 2400.60 2012-07-27 3001.0 5001.0

5 70010.0 NaN 2012-10-10 3003.0 5003.0

6 70003.0 12.43 2012-10-10 3002.0 5002.0

7 70012.0 2480.40 2012-06-27 3001.0 5003.0

8 NaN NaN 2012-08-17 3001.0 3001.0

9 70013.0 3045.60 2012-04-25 NaN NaN

In [ ]:

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

import pandas as pd

import numpy as np

*# Creating the DataFrame with the provided data*

data = {

'ord\_no': [np.nan, 70002.0, 4, np.nan, 70005.0, np.nan, 70010.0, 70003.0, 70012.0, np.nan, np.nan],

'purch\_amt': [np.nan, 65.26, 948.50, np.nan, 2400.60, 5760.00, 1983.43, 2480.40, 250.45, 75.29, np.nan],

'ord\_date': [np.nan, '2012-09-10', 3002.0, '2012-09-10', '2012-07-27', '2012-09-10', '2012-10-10', '2012-06-27', '3001.0', np.nan, np.nan],

'customer\_id': [np.nan, 3001.0, 3002.0, np.nan, 3001.0, 3001.0, 3004.0, 3002.0, 3001.0, np.nan, np.nan]

}

df = pd.DataFrame(data)

*# Filter rows with at least 2 NaN values*

filtered\_df = df[df.isnull().sum(axis=1) >= 2]

*# Display the filtered DataFrame*

print(filtered\_df)

ord\_no purch\_amt ord\_date customer\_id

0 NaN NaN NaN NaN

3 NaN NaN 2012-09-10 NaN

9 NaN 75.29 NaN NaN

10 NaN NaN NaN NaN

In [ ]:

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

import pandas as pd

*# Creating the DataFrame with the provided data*

data = {

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

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

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

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

'school': ['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']

}

df = pd.DataFrame(data)

*# Grouping the DataFrame based on school code*

grouped = df.groupby('school')

*# Checking the type of GroupBy object*

print(type(grouped))

*# Displaying the groups*

for school\_code, group in grouped:

print(f"School Code: {school\_code}")

print(group)

print("\n")

<class 'pandas.core.groupby.generic.DataFrameGroupBy'>

School Code: s001

age height weight address school class name date\_of\_birth

0 12 173 35 street1 s001 V Alberto Franco 15/05/2002

3 14 151 31 street2 s001 VI Eesha Hinton 25/09/1998

School Code: s002

age height weight address school class name date\_of\_birth

1 12 192 32 street2 s002 V Gino Mcneill 17/05/2002

4 12 159 32 street4 s002 V Gino Mcneill 11/05/2002

School Code: s003

age height weight address school class name date\_of\_birth

2 13 186 33 street3 s003 VI Ryan Parkes 16/02/1999

School Code: s004

age height weight address school class name date\_of\_birth

5 13 167 30 street1 s004 VI David Parkes 15/09/1997

In [ ]:

*#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.*

import pandas as pd

*# Creating the DataFrame with the provided data*

data = {

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

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

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

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

'school': ['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']

}

df = pd.DataFrame(data)

*# Convert 'age' column to numeric type*

df['age'] = pd.to\_numeric(df['age'], errors='coerce')

*# Grouping the DataFrame by school code and calculating mean, min, and max age for each school*

result = df.groupby('school')['age'].agg(['mean', 'min', 'max'])

*# Displaying the mean, min, and max age for each school*

print(result)

mean min max

school

s001 13.0 12 14

s002 12.0 12 12

s003 13.0 13 13

s004 13.0 13 13

In [ ]:

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

import pandas as pd

*# Creating the DataFrame with the provided data*

data = {

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

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

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

'school': ['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']

}

df = pd.DataFrame(data)

*# Grouping the DataFrame based on 'school' and 'class'*

grouped = df.groupby(['school', 'class'])

*# Displaying the groups*

for group\_name, group in grouped:

print(f"Group Name: {group\_name}")

print(group)

print("\n")

Group Name: ('s001', 'V')

height weight address school class name date\_of\_birth

0 173 35 street1 s001 V Alberto Franco 15/05/2002

Group Name: ('s001', 'VI')

height weight address school class name date\_of\_birth

3 151 31 street2 s001 VI Eesha Hinton 25/09/1998

Group Name: ('s002', 'V')

height weight address school class name date\_of\_birth

1 192 32 street2 s002 V Gino Mcneill 17/05/2002

4 159 32 street4 s002 V Gino Mcneill 11/05/2002

Group Name: ('s003', 'VI')

height weight address school class name date\_of\_birth

2 186 33 street3 s003 VI Ryan Parkes 16/02/1999

Group Name: ('s004', 'VI')

height weight address school class name date\_of\_birth

5 167 30 street1 s004 VI David Parkes 15/09/1997

In [ ]:

*#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.*

import pandas as pd

*# Assuming 'data' is the variable containing the World alcohol consumption dataset*

*# Replace this with your actual DataFrame or file loading method*

*# Example data for representation purposes*

data = {

'Year': [1986, 1986, 1986, 1985, 1986, 1987],

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

'Country': ['Viet Nam', 'Uruguay', "Cte d'Ivoire", 'Colombia', 'Saint Kitts and Nevis', 'Country6'],

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

'Display Value': [0.00, 0.50, 1.62, 4.27, 1.98, 2.0]

}

df = pd.DataFrame(data)

*# Displaying the dimensions (shape) of the dataset*

print("Dimensions or Shape of the DataFrame:")

print(df.shape)

*# Extracting the column names*

print("\nColumn Names:")

print(df.columns)

Dimensions or Shape of the DataFrame:

(6, 5)

Column Names:

Index(['Year', 'WHO region', 'Country', 'Beverage Types', 'Display Value'], dtype='object')

In [ ]:

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

import pandas as pd

*# Sample DataFrame for demonstration*

data = {

'Column1': ['apple', 'banana', 'orange', 'grape', 'melon'],

'Column2': [10, 20, 15, 25, 30]

}

df = pd.DataFrame(data)

*# Finding the index of a given substring in 'Column1'*

substring = 'an' *# Substring to find within the DataFrame column*

indices = df[df['Column1'].str.contains(substring)].index.tolist()

*# Displaying the indices of rows containing the substring*

print(f"Indices of rows containing '{substring}':")

print(indices)

Indices of rows containing 'an':

[1, 2]

In [ ]:

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

import pandas as pd

*# Sample DataFrame*

data = {

'Column1': ['Apple', 'Banana', 'Orange', 'Grape', 'Melon'],

'Column2': [10, 20, 15, 25, 30]

}

df = pd.DataFrame(data)

*# Specified column to swap cases (Column1 in this example)*

specified\_column = 'Column1'

*# Swapping cases in the specified column*

df[specified\_column] = df[specified\_column].str.swapcase()

*# Displaying the updated DataFrame*

print("DataFrame after swapping cases:")

print(df)

DataFrame after swapping cases:

Column1 Column2

0 aPPLE 10

1 bANANA 20

2 oRANGE 15

3 gRAPE 25

4 mELON 30

In [ ]:

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

import matplotlib.pyplot as plt

*# Sample data for the line*

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

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

*# Plotting the line*

plt.plot(x\_values, y\_values)

*# Adding labels and title*

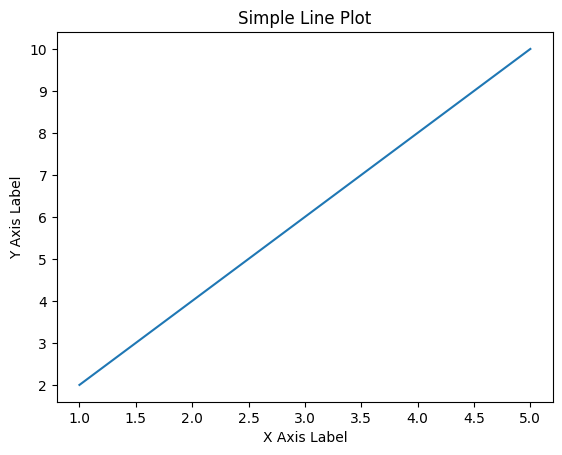
plt.xlabel('X Axis Label')

plt.ylabel('Y Axis Label')

plt.title('Simple Line Plot')

*# Show the plot*

plt.show()



In [ ]:

*#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.*

import matplotlib.pyplot as plt

*# Read data from the text file*

with open('/content/text.txt', 'r') as file:

data = file.read().splitlines()

x = []

y = []

*# Parse the data from the file*

for line in data:

values = line.split()

x.append(int(values[0]))

y.append(int(values[1]))

*# Plot the line*

plt.plot(x, y)

*# Add labels and title*

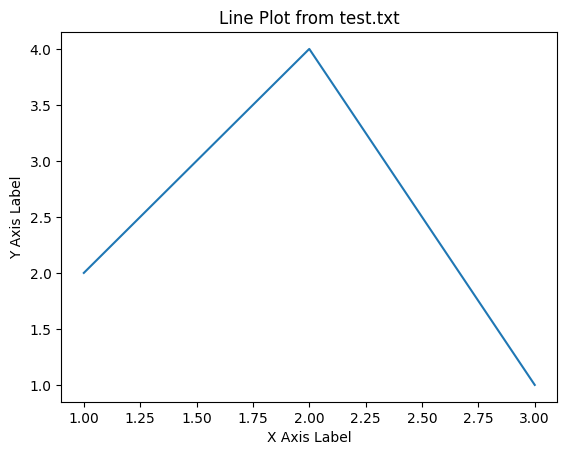
plt.xlabel('X Axis Label')

plt.ylabel('Y Axis Label')

plt.title('Line Plot from test.txt')

*# Show the plot*

plt.show()



In [ ]:

In [ ]:

*#24Write a Python program to draw line charts of the financial data of Alphabet Inc. between October 3, 2016 to October 7, 2016.*

import pandas as pd

import matplotlib.pyplot as plt

*# Load the financial data from the CSV file*

file\_path = "/content/GoogleStockPrices.csv"

df = pd.read\_csv(file\_path)

*# Filter data for the specified date range (October 3, 2016 to October 7, 2016)*

start\_date = '2016-10-03'

end\_date = '2016-10-07'

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

*# Extract Date and Closing Price columns*

dates = filtered\_data['Date']

closing\_prices = filtered\_data['Close']

*# Create a line chart*

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

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

plt.title('Alphabet Inc. Stock Prices (Oct 3, 2016 - Oct 7, 2016)')

plt.xlabel('Date')

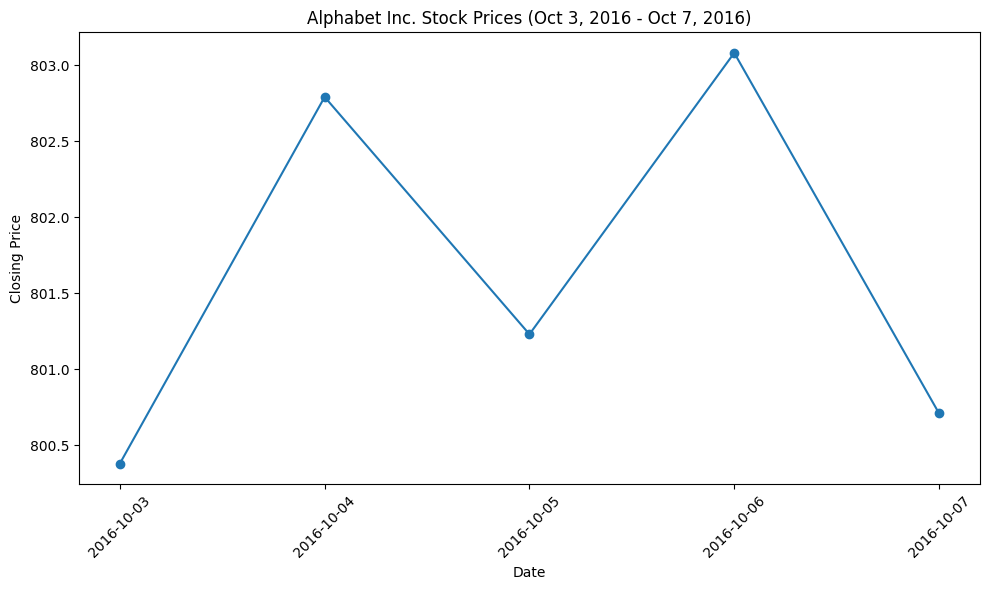
plt.ylabel('Closing Price')

plt.xticks(rotation=45) *# Rotate x-axis labels for readability*

plt.tight\_layout()

*# Show the line chart*

plt.show()



In [ ]:

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

import matplotlib.pyplot as plt

*# Sample data for the lines*

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

y1 = [3, 6, 9, 12, 15]

y2 = [2, 4, 6, 8, 10]

*# Create a line chart with legends, different widths, and colors*

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

*# Plot the first line with a blue color, label, and width*

plt.plot(x, y1, label='Line 1', color='blue', linewidth=2)

*# Plot the second line with a red color, label, and width*

plt.plot(x, y2, label='Line 2', color='red', linewidth=2)

*# Add labels and legend*

plt.xlabel('X-Axis')

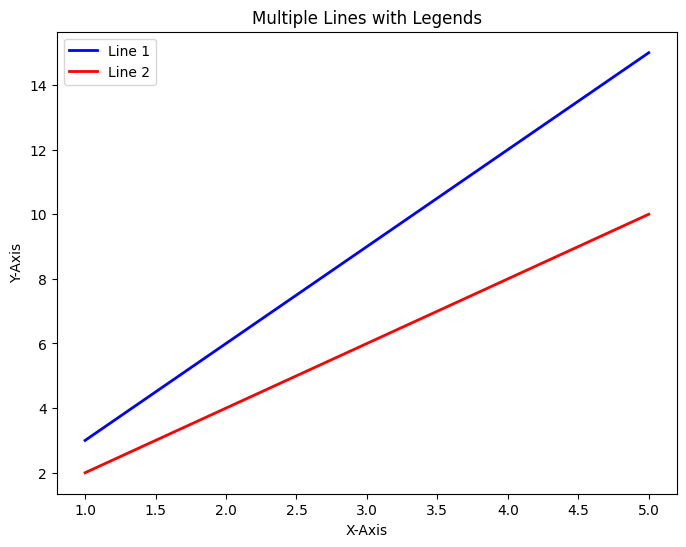
plt.ylabel('Y-Axis')

plt.title('Multiple Lines with Legends')

plt.legend()

*# Show the plot*

plt.show()



In [ ]:

*#26Write a Python program to create multiple plots.*

import matplotlib.pyplot as plt

import numpy as np

*# Create some sample data for plotting*

x = np.linspace(0, 2 \* np.pi, 100)

y1 = np.sin(x)

y2 = np.cos(x)

y3 = np.sin(2 \* x)

y4 = np.cos(2 \* x)

*# Create a figure with multiple subplots*

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

*# Create the first subplot (2 rows, 2 columns, first plot)*

plt.subplot(2, 2, 1)

plt.plot(x, y1, label='sin(x)', color='blue')

plt.title('Plot 1')

plt.legend()

*# Create the second subplot (2 rows, 2 columns, second plot)*

plt.subplot(2, 2, 2)

plt.plot(x, y2, label='cos(x)', color='red')

plt.title('Plot 2')

plt.legend()

*# Create the third subplot (2 rows, 2 columns, third plot)*

plt.subplot(2, 2, 3)

plt.plot(x, y3, label='sin(2x)', color='green')

plt.title('Plot 3')

plt.legend()

*# Create the fourth subplot (2 rows, 2 columns, fourth plot)*

plt.subplot(2, 2, 4)

plt.plot(x, y4, label='cos(2x)', color='purple')

plt.title('Plot 4')

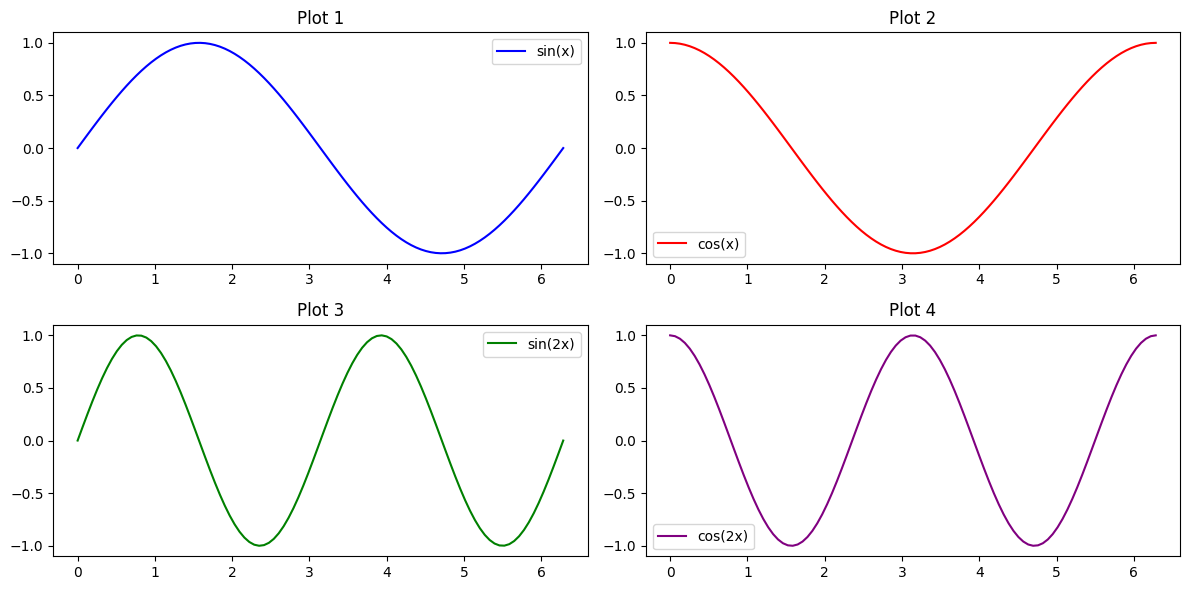
plt.legend()

*# Adjust layout to prevent overlap*

plt.tight\_layout()

*# Show the plots*

plt.show()



In [ ]:

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

import matplotlib.pyplot as plt

*# Sample data*

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

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

*# Create a bar chart*

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

plt.bar(languages, popularity, color=['blue', 'green', 'red', 'orange', 'purple', 'pink'])

*# Add labels and a title*

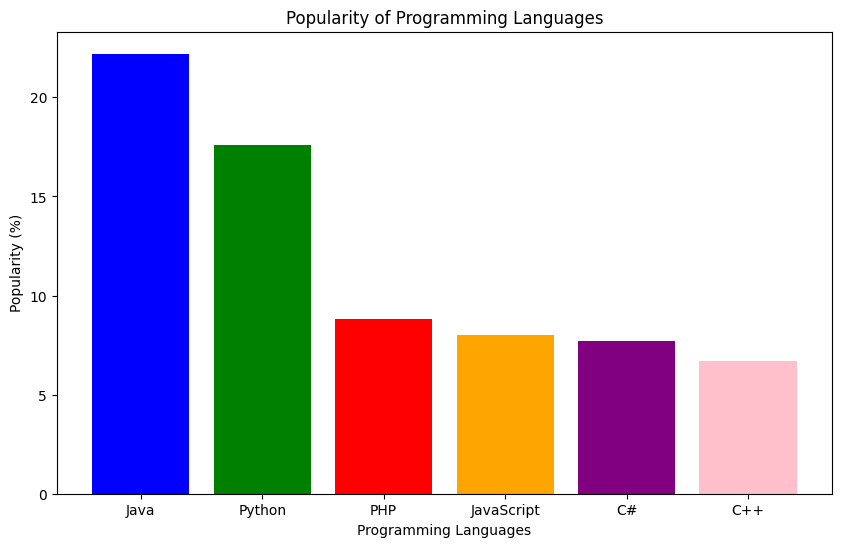
plt.xlabel('Programming Languages')

plt.ylabel('Popularity (%)')

plt.title('Popularity of Programming Languages')

*# Show the bar chart*

plt.show()



In [10]:

*#28Write a Python programming to display a horizontal bar chart of the popularity of programming Languages.*

import matplotlib.pyplot as plt

*# Sample data*

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

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

*# Create a horizontal bar chart*

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

plt.barh(languages, popularity, color=['blue', 'green', 'red', 'orange', 'purple', 'pink'])

*# Add labels and a title*

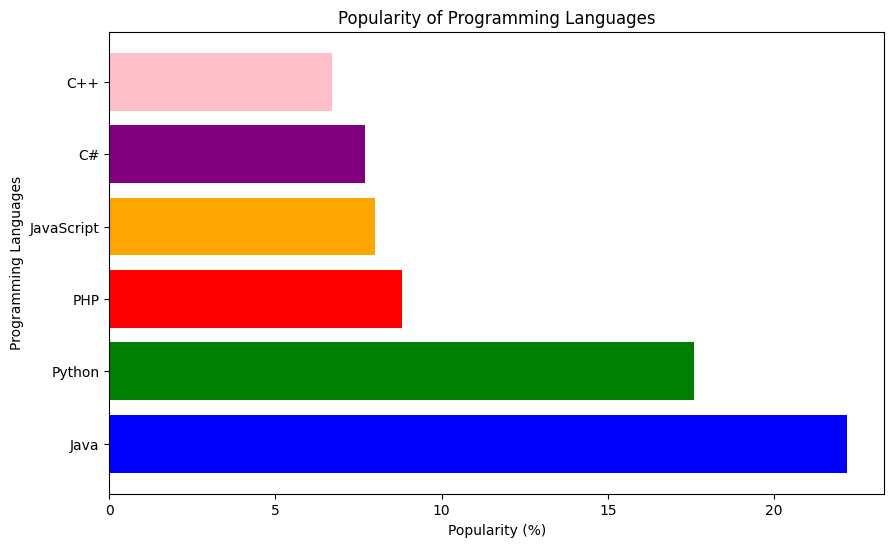
plt.xlabel('Popularity (%)')

plt.ylabel('Programming Languages')

plt.title('Popularity of Programming Languages')

*# Show the horizontal bar chart*

plt.show()



In [11]:

*#29Write a Python programming to display a bar chart of the popularity of programming Languages. Use different color for each bar.*

import matplotlib.pyplot as plt

import numpy as np

*# Sample data*

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

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

*# Generate different colors for each bar*

colors = plt.cm.viridis(np.linspace(0, 1, len(languages)))

*# Create a bar chart*

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

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

*# Add labels and a title*

plt.xlabel('Programming Languages')

plt.ylabel('Popularity (%)')

plt.title('Popularity of Programming Languages')

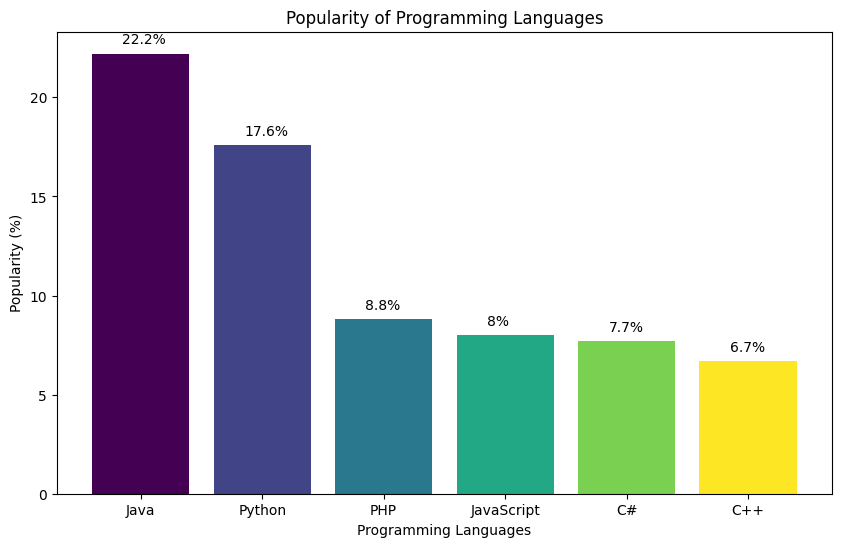
*# Adding data values above the bars*

for bar, pop in zip(bars, popularity):

plt.text(bar.get\_x() + bar.get\_width() / 2 - 0.15, bar.get\_height() + 0.5, f'{pop}%', fontsize=10)

*# Show the bar chart*

plt.show()



In [12]:

*#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.*

import matplotlib.pyplot as plt

import numpy as np

*# Sample data*

groups = ["Group 1", "Group 2", "Group 3", "Group 4", "Group 5"]

means\_men = [22, 30, 35, 35, 26]

means\_women = [25, 32, 30, 35, 29]

*# Generate X positions for the bars*

x = np.arange(len(groups))

bar\_width = 0.35

*# Create a bar plot for men*

plt.bar(x - bar\_width/2, means\_men, bar\_width, label='Men', color='b')

*# Create a bar plot for women*

plt.bar(x + bar\_width/2, means\_women, bar\_width, label='Women', color='r')

*# Add labels and a title*

plt.xlabel('Groups')

plt.ylabel('Scores')

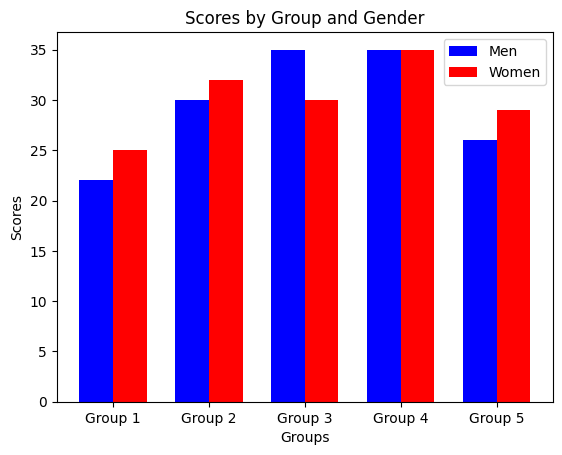
plt.title('Scores by Group and Gender')

plt.xticks(x, groups)

plt.legend()

*# Show the bar plot*

plt.show()



In [14]:

*#31Write a Python program to create a stacked bar plot with error bars.*

import matplotlib.pyplot as plt

import numpy as np

*# Sample data*

groups = ["Group 1", "Group 2", "Group 3", "Group 4", "Group 5"]

means\_men = [22, 30, 35, 35, 26]

means\_women = [25, 32, 30, 35, 29]

std\_dev\_men = [4, 3, 4, 1, 5]

std\_dev\_women = [3, 5, 2, 3, 3]

*# Generate X positions for the bars*

x = np.arange(len(groups))

*# Create a stacked bar plot*

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

plt.bar(x, means\_men, yerr=std\_dev\_men, label='Men', color='b', capsize=6)

plt.bar(x, means\_women, yerr=std\_dev\_women, bottom=means\_men, label='Women', color='r', capsize=6)

*# Add labels and a title*

plt.xlabel('Groups')

plt.ylabel('Scores')

plt.title('Stacked Bar Plot with Error Bars')

plt.xticks(x, groups)

plt.legend()

*# Show the stacked bar plot*

plt.show()



In [20]:

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

import matplotlib.pyplot as plt

import numpy as np

*# Generate random data for X and Y*

np.random.seed(0) *# For reproducibility*

num\_points = 100

X = np.random.rand(num\_points)

Y = np.random.rand(num\_points)

*# Create a scatter plot*

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

plt.scatter(X, Y, marker='o', color='r', label='Random Data')

*# Add labels and a title*

plt.xlabel('X-axis')

plt.ylabel('Y-axis')

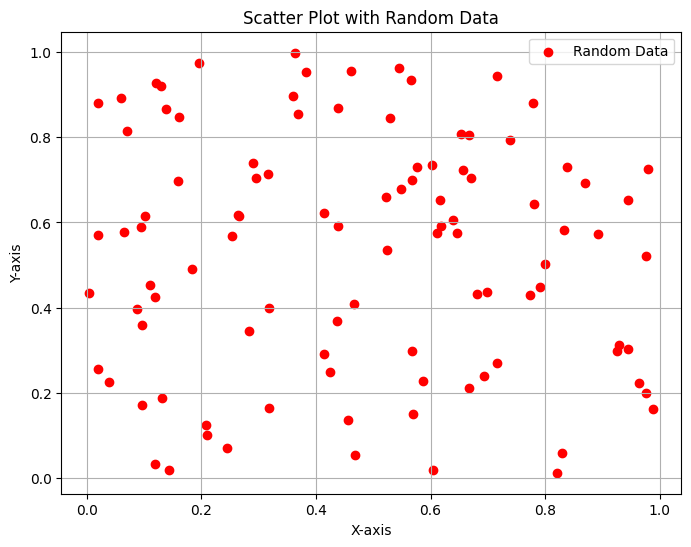
plt.title('Scatter Plot with Random Data')

*# Show the scatter plot*

plt.legend()

plt.grid(True)

plt.show()



In [22]:

*#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.*

import matplotlib.pyplot as plt

import numpy as np

*# Generate random data for X and Y*

np.random.seed(0) *# For reproducibility*

num\_points = 100

X = np.random.rand(num\_points)

Y = np.random.rand(num\_points)

*# Create a scatter plot with empty circles*

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

plt.scatter(X, Y, marker='o', facecolors='none', edgecolors='green', label='Random Data')

*# Add labels and a title*

plt.xlabel('X-axis')

plt.ylabel('Y-axis')

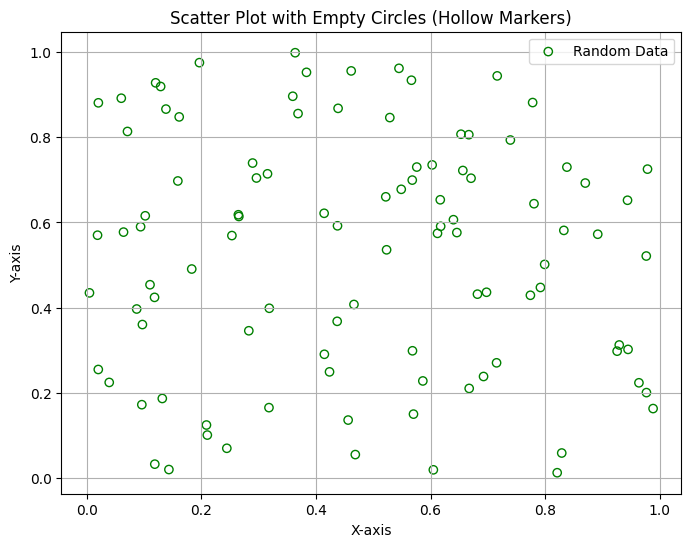
plt.title('Scatter Plot with Empty Circles (Hollow Markers)')

*# Show the scatter plot*

plt.legend()

plt.grid(True)

plt.show()



In [24]:

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

import matplotlib.pyplot as plt

import numpy as np

from matplotlib import cm

*# Generate random data for X, Y, and sizes*

np.random.seed(0) *# For reproducibility*

num\_points = 100

X = np.random.rand(num\_points)

Y = np.random.rand(num\_points)

sizes = np.random.rand(num\_points) \* 100 *# Random sizes between 0 and 100*

*# Create a colormap*

cmap = cm.get\_cmap('viridis', len(sizes))

*# Create a scatter plot with different-sized and colored balls*

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

sc = plt.scatter(X, Y, s=sizes, c=range(len(sizes)), cmap=cmap, marker='o', alpha=0.7, label='Random Data')

*# Add a colorbar*

cbar = plt.colorbar(sc)

cbar.set\_label('Size Index')

*# Add labels and a title*

plt.xlabel('X-axis')

plt.ylabel('Y-axis')

plt.title('Scatter Plot with Different-Sized and Colored Balls')

*# Show the scatter plot*

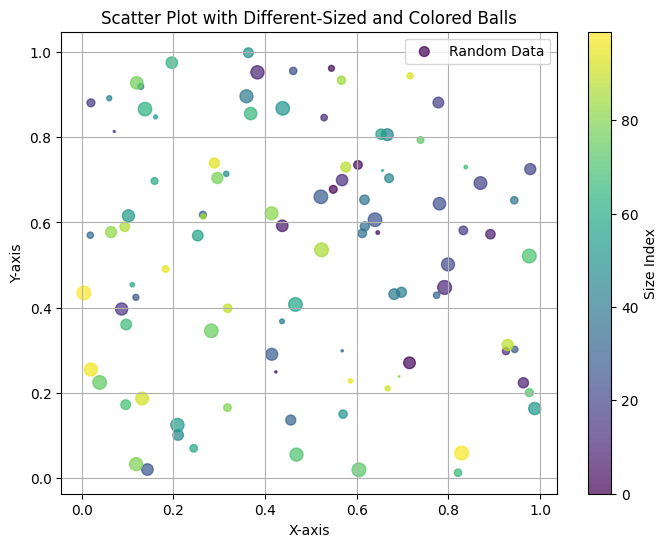
plt.legend()

plt.grid(True)

plt.show()

<ipython-input-24-cf3ff88bee71>:14: MatplotlibDeprecationWarning: The get\_cmap function was deprecated in Matplotlib 3.7 and will be removed two minor releases later. Use ``matplotlib.colormaps[name]`` or ``matplotlib.colormaps.get\_cmap(obj)`` instead.

cmap = cm.get\_cmap('viridis', len(sizes))



In [25]:

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

import matplotlib.pyplot as plt

*# Sample 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]

*# Create a scatter plot*

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

plt.scatter(math\_marks, science\_marks, s=100, c=marks\_range, cmap='viridis', edgecolors='k', alpha=0.7)

*# Add labels and a title*

plt.xlabel('Mathematics Marks')

plt.ylabel('Science Marks')

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

*# Add a colorbar to show the marks range*

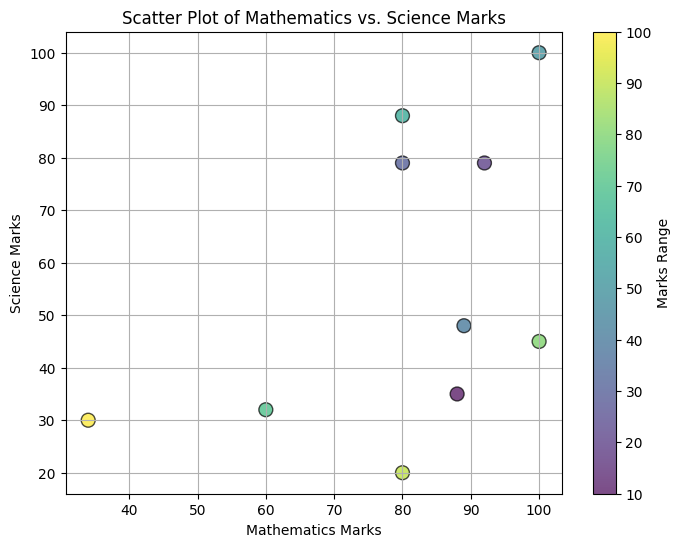
cbar = plt.colorbar()

cbar.set\_label('Marks Range')

*# Show the scatter plot*

plt.grid(True)

plt.show()



In [26]:

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

import matplotlib.pyplot as plt

import numpy as np

*# Sample data for three groups: Group 1, Group 2, and Group 3*

*# Each group has random weights and heights for 10 individuals*

np.random.seed(0) *# For reproducibility*

num\_points = 10

*# Group 1*

group1\_weights = np.random.uniform(50, 80, num\_points)

group1\_heights = np.random.uniform(150, 190, num\_points)

*# Group 2*

group2\_weights = np.random.uniform(60, 90, num\_points)

group2\_heights = np.random.uniform(160, 200, num\_points)

*# Group 3*

group3\_weights = np.random.uniform(70, 100, num\_points)

group3\_heights = np.random.uniform(170, 210, num\_points)

*# Create a scatter plot for each group*

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

plt.scatter(group1\_weights, group1\_heights, label='Group 1', color='blue', marker='o')

plt.scatter(group2\_weights, group2\_heights, label='Group 2', color='green', marker='s')

plt.scatter(group3\_weights, group3\_heights, label='Group 3', color='red', marker='^')

*# Add labels and a title*

plt.xlabel('Weight (kg)')

plt.ylabel('Height (cm)')

plt.title('Scatter Plot of Weight vs. Height for Three Groups')

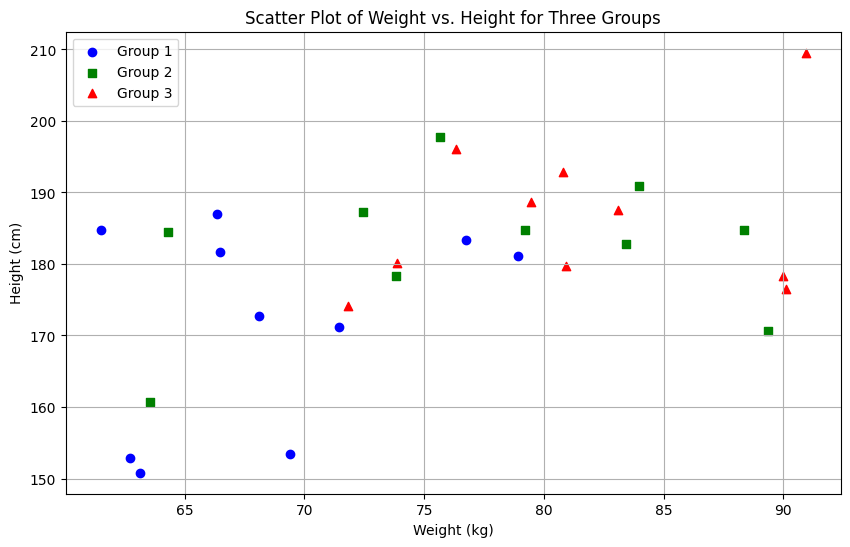
*# Show the legend*

plt.legend()

*# Show the scatter plot*

plt.grid(True)

plt.show()



In [27]:

*#37.Write a Pandas program to create a dataframe from a dictionary and display it.*

import pandas as pd

*# Sample data as a dictionary*

data = {'X': [78, 85, 96, 80, 86], 'Y': [84, 94, 89, 83, 86], 'Z': [86, 97, 96, 72, 83]}

*# Create a DataFrame*

df = pd.DataFrame(data)

*# Display the DataFrame*

print(df)

X Y Z

0 78 84 86

1 85 94 97

2 96 89 96

3 80 83 72

4 86 86 83

In [28]:

*#38.Write a Pandas program to create and display a DataFrame from a specified dictionary data which has the index labels.*

import pandas as pd

import numpy as np

*# 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']

*# Create a DataFrame with custom index labels*

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

*# Display the DataFrame*

print(df)

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

In [29]:

*#39.Write a Pandas program to get the first 3 rows of a given DataFrame.*

import pandas as pd

import numpy as np

*# 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']

*# Create a DataFrame with custom index labels*

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

*# Get the first 3 rows*

first\_3\_rows = df.head(3)

*# Display the first 3 rows*

print(first\_3\_rows)

name score attempts qualify

a Anastasia 12.5 1 yes

b Dima 9.0 3 no

c Katherine 16.5 2 yes

In [30]:

*#40.Write a Pandas program to select the 'name' and 'score' columns from the following DataFrame.*

import pandas as pd

import numpy as np

*# 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']

*# Create a DataFrame with custom index labels*

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

*# Select the 'name' and 'score' columns*

selected\_columns = df[['name', 'score']]

*# Display the selected columns*

print(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