**1.**

import pandas as pd

data = {

    'DEPARTMENT\_ID': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160, 170, 180, 190, 200, 210, 220, 230, 240, 250, 260, 270],

    'DEPARTMENT\_NAME': ['Administration', 'Marketing', 'Purchasing', 'Human Resources', 'Shipping', 'IT', 'Public Relations', 'Sales', 'Executive', 'Finance', 'Accounting', 'Treasury', 'Corporate Tax', 'Control And Credit', 'Shareholder Services', 'Benefits', 'Manufacturing', 'Construction', 'Contracting', 'Operations', 'IT Support', 'NOC', 'IT Helpdesk', 'Government Sales', 'Retail Sales', 'Recruiting', 'Payroll'],

    'MANAGER\_ID': [200, 201, 114, 203, 121, 103, 204, 145, 100, 108, 205, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],

    'LOCATION\_ID': [1700, 1800, 1700, 2400, 1500, 1400, 2700, 2500, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700]

}

df = pd.DataFrame(data)

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

print("Distinct department IDs:")

print(distinct\_department\_ids)

output: 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]

**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

# Sample 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]

}

# Creating DataFrame

df = pd.DataFrame(data)

# Converting START\_DATE and END\_DATE to datetime objects

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

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

# Grouping by EMPLOYEE\_ID and counting the number of unique JOB\_IDs

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

# Filtering employees who did two or more jobs in the past

employees\_with\_multiple\_jobs = job\_counts[job\_counts >= 2].index

# Displaying the IDs for those employees

print("Employees who did two or more jobs in the past:")

print(employees\_with\_multiple\_jobs)

output: Employees who did two or more jobs in the past:

Int64Index([101, 176, 200], dtype='int64', name='EMPLOYEE\_ID')

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

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)

# Sort dataframe by 'JOB\_TITLE' column in descending order

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

print(df\_sorted)

output: 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

1. FI\_ACCOUNT Accountant 4200 9000

**4.** Write a Pandas program to create a line plot of the historical stock

prices of Alphabet Inc. between two specific dates.

import pandas as pd

import matplotlib.pyplot as plt

# Sample data (replace this with your actual data)

data = {

    'Date': ['2024-01-01', '2024-01-02', '2024-01-03', '2024-01-04', '2024-01-05'],

    'Close': [100.50, 101.20, 99.80, 102.30, 104.50]

}

# Create a DataFrame from the sample data

df = pd.DataFrame(data)

# Convert the 'Date' column to datetime format

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

# Filter the data for the specific date range

start\_date = '2024-01-01'

end\_date = '2024-01-05'

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

# Plotting the data

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

plt.plot(filtered\_df['Date'], filtered\_df['Close'], marker='o', linestyle='-')

plt.title('Historical Stock Prices of Alphabet Inc. between {} and {}'.format(start\_date, end\_date))

plt.xlabel('Date')

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

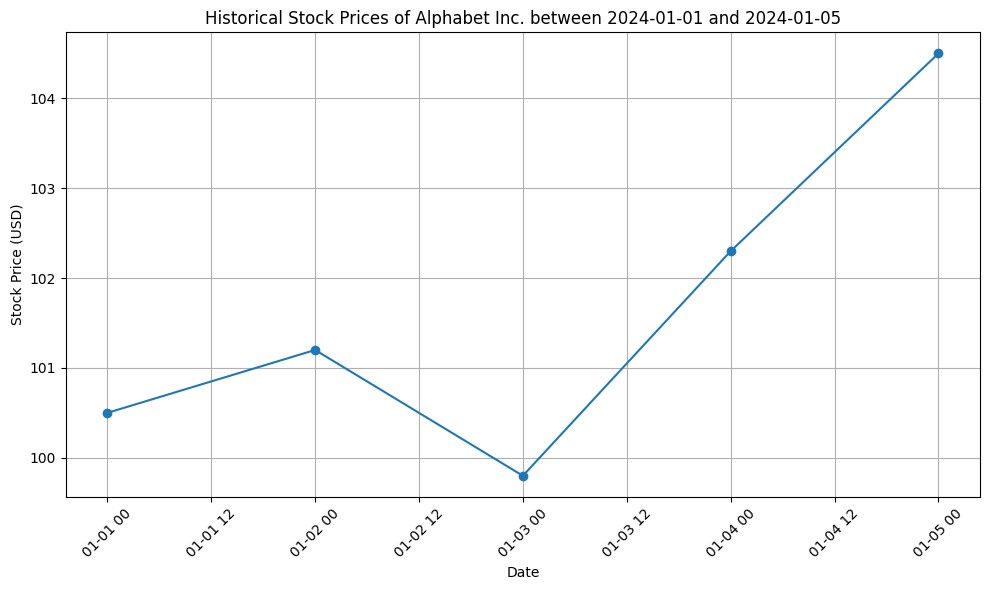
plt.xticks(rotation=45)

plt.grid(True)

plt.tight\_layout()

plt.show()

output:



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

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

# Generate sample data for demonstration

np.random.seed(0)

dates = pd.date\_range(start='2024-01-01', end='2024-03-01', freq='D')

volume = np.random.randint(10000, 50000, size=len(dates))

# Create a DataFrame

df = pd.DataFrame({'Date': dates, 'Volume': volume})

# Create a bar plot of trading volume

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

plt.bar(df['Date'], df['Volume'], color='skyblue')

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

plt.xlabel('Date')

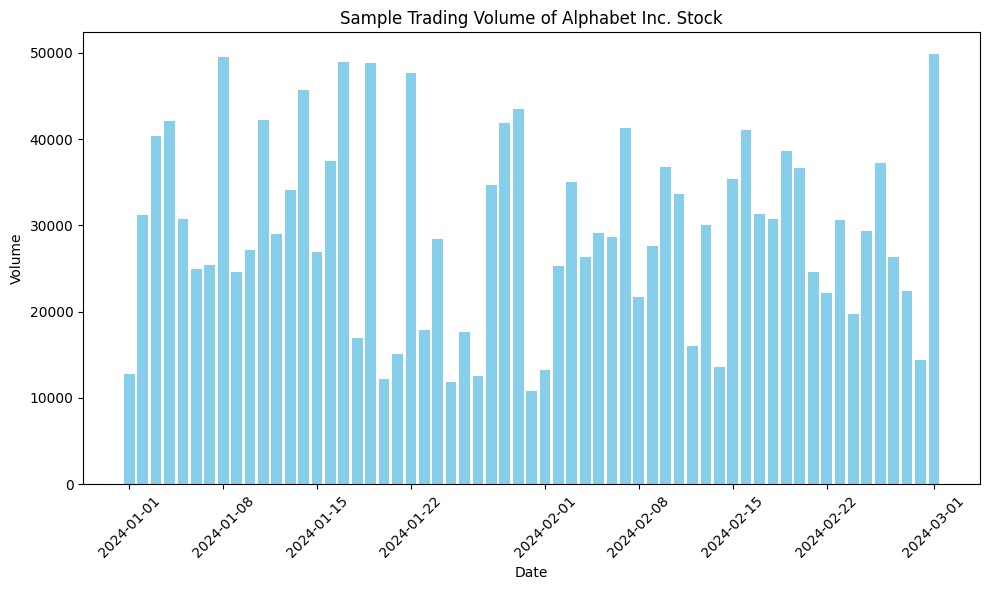
plt.ylabel('Volume')

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

output:



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

import numpy as np

import matplotlib.pyplot as plt

# Generate synthetic data for demonstration

num\_points = 100

dates = np.random.choice(pd.date\_range(start='2023-01-01', end='2024-01-01', freq='D'), size=num\_points, replace=False)

open\_prices = np.random.uniform(low=100, high=200, size=num\_points)

high\_prices = open\_prices + np.random.uniform(low=0, high=20, size=num\_points)

low\_prices = open\_prices - np.random.uniform(low=0, high=20, size=num\_points)

close\_prices = np.random.uniform(low=100, high=200, size=num\_points)

adj\_close\_prices = np.random.uniform(low=90, high=210, size=num\_points)

volumes = np.random.randint(low=10000, high=100000, size=num\_points)

# Plot scatter plot of trading volume and closing prices

plt.figure(figsize=(10, 4))

plt.scatter(volumes, close\_prices, color='pink', alpha=0.6)

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

plt.xlabel('Volume')

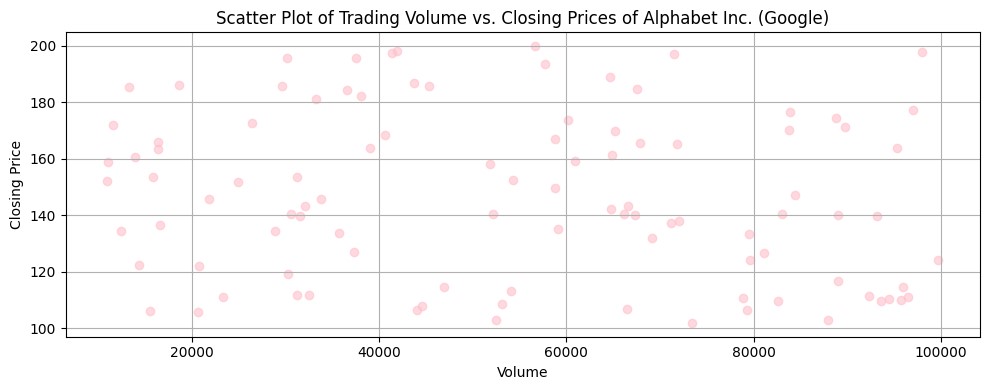
plt.ylabel('Closing Price')

plt.grid(True)

plt.tight\_layout()

plt.show()

output:



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

import pandas as pd

# Assuming sales\_data is a DataFrame containing sales data

sales\_data = {

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

    'Sale Amount': [100, 150, 200, 250, 300]

}

# Creating DataFrame from sales\_data

df = pd.DataFrame(sales\_data)

# Creating a Pivot table to find maximum and minimum sale value of items

pivot\_max\_min\_sales = pd.pivot\_table(df, values='Sale Amount', index='Item', aggfunc={'Sale Amount': [max, min]})

print("Pivot table for maximum and minimum sale value of items:")

print(pivot\_max\_min\_sales)

output:

Pivot table for maximum and minimum sale value of items:

max min

Item

A 200 100

B 250 150

C 300 300

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

# Sample sales data

sales\_data = {

    'Date': ['2024-01-01', '2024-01-01', '2024-01-02', '2024-01-02', '2024-01-03'],

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

    'Units Sold': [10, 15, 20, 25, 30],

    'Sale Amount': [100, 150, 200, 250, 300]

}

# Creating DataFrame from sales\_data

df = pd.DataFrame(sales\_data)

# Creating a Pivot table to find item wise unit sold

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

print("Pivot table for unit sold of each item:")

print(pivot\_unit\_sold)

output:

Pivot table for unit sold of each item:

Units Sold

Item

A 30

B 40

C 30

**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

# Assuming sales\_data is your DataFrame containing sales data

# Sample data

sales\_data = {

    'Region': ['North', 'North', 'South', 'South', 'East', 'East'],

    'Manager': ['John', 'John', 'Alice', 'Alice', 'Bob', 'Bob'],

    'Salesman': ['Sam', 'Tom', 'Rob', 'Ken', 'Jake', 'Mike'],

    'Sale\_Amount': [1000, 1500, 2000, 2500, 1800, 2200]

}

sales\_df = pd.DataFrame(sales\_data)

# Creating Pivot table

pivot\_table = pd.pivot\_table(sales\_df, index=['Region', 'Manager', 'Salesman'], aggfunc='sum')

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

print(pivot\_table)

output:

Pivot Table - Total sale amount region-wise, manager-wise, and salesperson-wise:

Sale\_Amount

Region Manager Salesman

East Bob Jake 1800

Mike 2200

North John Sam 1000

Tom 1500

South Alice Ken 2500

Rob 2000

**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 numpy as np

# Create a DataFrame with random values

np.random.seed(0)  # For reproducibility

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

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

# Define a function to apply color based on values

def color\_negative\_red(val):

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

    return 'color: %s' % color

# Apply styling to the DataFrame

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

# Display the styled DataFrame

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.

import pandas as pd

import numpy as np

# Create a DataFrame with random values

np.random.seed(0)  # for reproducibility

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

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

# Randomly convert some values to NaN

nan\_indices = np.random.choice(df.size, size=int(df.size \* 0.3), replace=False)

df.values.flat[nan\_indices] = np.nan

# Define a function to highlight NaN values

def highlight\_nan(val):

    if pd.isna(val):

        return 'background-color: yellow'

    else:

        return ''

# Apply the style

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

# Display the styled DataFrame

styled\_df

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.

import pandas as pd

import numpy as np

# Create a DataFrame with random values

np.random.seed(0)  # for reproducibility

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

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

# Display the styled DataFrame

styled\_df

output:

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

# Create a DataFrame with random values including NaNs

np.random.seed(0)  # for reproducibility

data = np.random.rand(5, 3)

data[np.random.choice([True, False], size=data.shape, p=[0.3, 0.7])] = np.nan

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

# Detect missing values and display True or False

missing\_values = df.isna()

print(missing\_values)

output:

A B C

0 True True False

1 False False False

2 False False False

3 True False True

4 False False False

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

import pandas as pd

import numpy as np

# Create a DataFrame with random values including NaNs

np.random.seed(0)  # for reproducibility

data = np.random.rand(5, 3)

data[np.random.choice([True, False], size=data.shape, p=[0.3, 0.7])] = np.nan

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

# Display the original DataFrame

print("Original DataFrame:")

print(df)

# Find and replace missing values

df.fillna(df.mean(), inplace=True)

# Display the DataFrame after replacing missing values

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

print(df)

**output:**

Original DataFrame:

A B C

0 NaN NaN 0.602763

1 0.544883 0.423655 0.645894

2 0.437587 0.891773 0.963663

3 NaN 0.791725 NaN

4 0.568045 0.925597 0.071036

DataFrame after replacing missing values:

A B C

0 0.516838 0.758187 0.602763

1 0.544883 0.423655 0.645894

2 0.437587 0.891773 0.963663

3 0.516838 0.791725 0.570839

4 0.568045 0.925597 0.071036

**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

# Create a DataFrame with random values including NaNs

np.random.seed(0)  # for reproducibility

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

data[np.random.choice([True, False], size=data.shape, p=[0.3, 0.7])] = np.nan

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

# Display the original DataFrame

print("Original DataFrame:")

print(df)

# Keep the rows with at least 2 NaN values

df\_with\_at\_least\_2\_nan = df.dropna(thresh=2)

# Display the DataFrame with at least 2 NaN values

print("\nDataFrame with at least 2 NaN values:")

print(df\_with\_at\_least\_2\_nan)

**output:**

Original DataFrame:

A B C D

0 0.548814 0.715189 0.602763 NaN

1 0.423655 0.645894 NaN NaN

2 0.963663 0.383442 0.791725 0.528895

3 0.568045 NaN NaN NaN

4 0.020218 NaN 0.778157 NaN

5 NaN NaN 0.461479 NaN

6 NaN 0.639921 0.143353 NaN

7 0.521848 NaN 0.264556 0.774234

8 0.456150 0.568434 0.018790 NaN

9 NaN NaN NaN NaN

DataFrame with at least 2 NaN values:

A B C D

0 0.548814 0.715189 0.602763 NaN

1 0.423655 0.645894 NaN NaN

2 0.963663 0.383442 0.791725 0.528895

4 0.020218 NaN 0.778157 NaN

6 NaN 0.639921 0.143353 NaN

7 0.521848 NaN 0.264556 0.774234

8 0.456150 0.568434 0.018790 NaN

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

# Create a sample DataFrame

data = {

    'school code': ['S001', 'S002', 'S001', 'S002', 'S003'],

    'student': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],

    'grade': [85, 90, 88, 92, 87]

}

df = pd.DataFrame(data)

# Split the DataFrame into groups based on school code

grouped = df.groupby('school code')

# Check the type of GroupBy object

print("Type of GroupBy object:", type(grouped))

# Display the groups

for name, group in grouped:

    print("\nSchool Code:", name)

    print(group)

**output:**

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

School Code: S001

school code student grade

0 S001 Alice 85

2 S001 Charlie 88

School Code: S002

school code student grade

1 S002 Bob 90

3 S002 David 92

School Code: S003

school code student grade

4 S003 Eve 87

**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

# Create a sample DataFrame

data = {

    'school code': ['S001', 'S002', 'S001', 'S002', 'S003'],

    'student': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],

    'age': [18, 20, 17, 19, 21]

}

df = pd.DataFrame(data)

# Split the DataFrame by school code and calculate mean, min, and max age for each school

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

# Display the result

print(result)

**output:**

mean min max

school code

S001 17.5 17 18

S002 19.5 19 20

S003 21.0 21 21

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

import pandas as pd

# Sample DataFrame

data = {

    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Emma', 'Frank', 'Grace', 'Henry', 'Ivy', 'Jack'],

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

    'Class': [10, 11, 10, 11, 10, 11, 10, 11, 10, 11],

    'Score': [85, 90, 88, 92, 89, 93, 87, 91, 86, 94]

}

df = pd.DataFrame(data)

# Grouping based on 'School Code' and 'Class'

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

# Printing the groups

for name, group in grouped:

    print("\nGroup:", name)

    print(group)

output:

Group: ('S001', 10)

Name School Code Class Score

0 Alice S001 10 85

2 Charlie S001 10 88

4 Emma S001 10 89

6 Grace S001 10 87

8 Ivy S001 10 86

Group: ('S002', 11)

Name School Code Class Score

1 Bob S002 11 90

3 David S002 11 92

5 Frank S002 11 93

7 Henry S002 11 91

9 Jack S002 11 94

**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

# Sample data

data = {

    'country': ['USA', 'Canada', 'UK', 'Germany', 'France'],

    'beer\_servings': [200, 220, 180, 250, 190],

    'spirit\_servings': [100, 120, 90, 150, 110],

    'wine\_servings': [50, 60, 70, 80, 90],

    'total\_litres\_of\_pure\_alcohol': [10.5, 11.0, 9.5, 12.0, 10.8]

}

# Create DataFrame

drinks\_df = pd.DataFrame(data)

# Display the dimensions or shape of the dataset

print("Dimensions of the World alcohol consumption dataset:")

print(drinks\_df.shape)

# Extract the column names from the dataset

column\_names = drinks\_df.columns

print("\nColumn names of the World alcohol consumption dataset:")

print(column\_names)

**output:**

Dimensions of the World alcohol consumption dataset:

(5, 5)

Column names of the World alcohol consumption dataset:

Index(['country', 'beer\_servings', 'spirit\_servings', 'wine\_servings',

'total\_litres\_of\_pure\_alcohol'],

dtype='object')

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

import pandas as pd

# Sample DataFrame

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

        'Age': [25, 30, 35, 40, 45,76]}

df = pd.DataFrame(data)

# Function to find the index of a substring in a column

def find\_substring\_index(dataframe, column\_name, substring):

    return dataframe[dataframe[column\_name].str.find(substring) != -1].index.tolist()

# Example usage

substring = 'al'

column\_name = 'Name'

indexes = find\_substring\_index(df, column\_name, substring)

if indexes:

    print(f"Indexes of rows containing '{substring}' in column '{column\_name}': {indexes}")

else:

    print(f"No matches found for '{substring}' in column '{column\_name}'")

**output:**

Indexes of rows containing 'al' in column 'Name': [5]

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

import pandas as pd

data = {'A': [1, 2, 3],

        'B': ['Hello', 'WORLD', 'HowAreYou']}

df = pd.DataFrame(data)

df['B'] = df['B'].str.swapcase()

print(df)

**output:**

A B

0 1 hELLO

1 2 world

2 3 hOWaREyOU

**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

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

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

plt.plot(x, y)

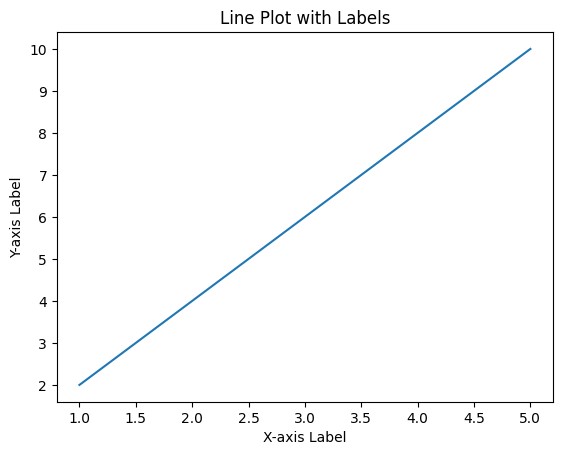
plt.xlabel('X-axis Label')

plt.ylabel('Y-axis Label')

plt.title('Line Plot with Labels')

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

import matplotlib.pyplot as plt

x = [1,2,3]

y = [2,4,1]

plt.plot(x, y)

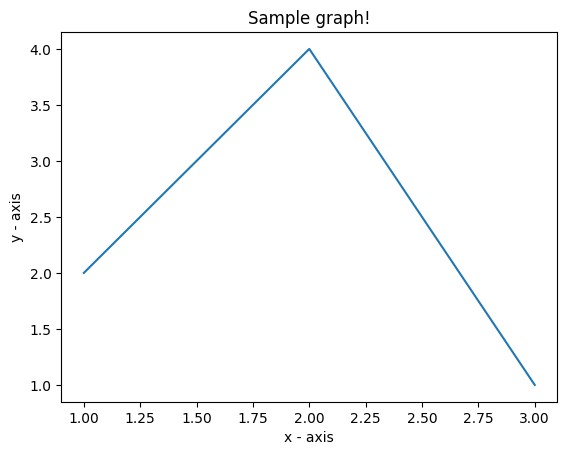
plt.xlabel('x - axis')

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

import matplotlib.pyplot as plt

import pandas as pd

data = {'date':['2010-03-16','2010-04-16','2010-05-16','2010-06-16','2010-07-16'],

        'open':[774.25, 776.35, 779.49, 779, 779.65],

        'high':[772.09, 775, 778, 774, 777.999],

        'low':[771, 772, 775, 773, 775],

        'close':[772.559998, 776.429993, 776.469971, 776.859985, 775.080017]

}

df= pd.DataFrame(data)

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

# Plot line chart

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

plt.plot(df.index, df['close'], marker='o', linestyle='-')

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

plt.xlabel('Date')

plt.ylabel('Closing Price')

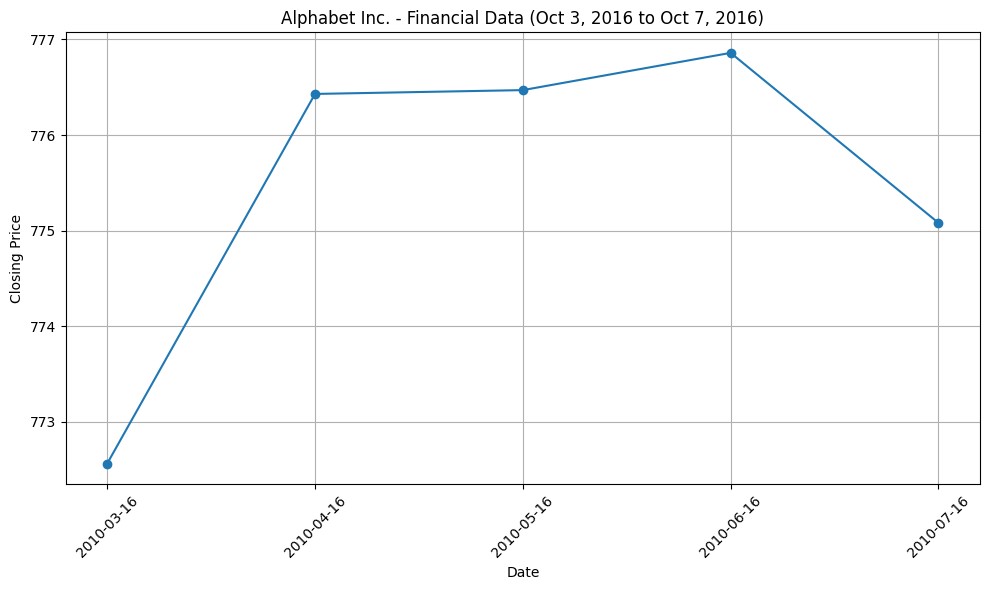
plt.grid(True)

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

**output:**



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

import matplotlib.pyplot as plt

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

y1 = [2, 4, 6, 8, 10]

y2 = [1, 3, 5, 7, 9]

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

plt.plot(x, y2, label='Line 2', linewidth=3, color='red')

plt.legend()

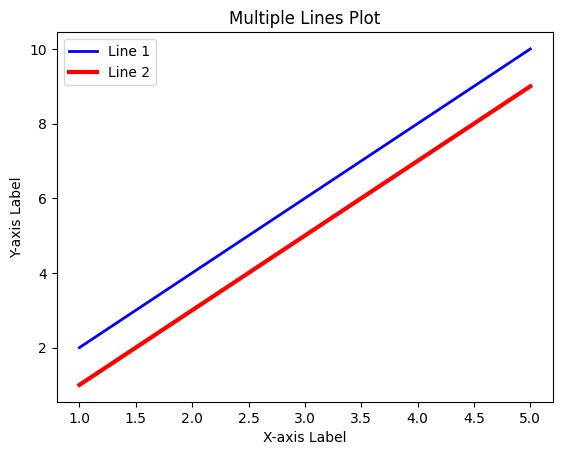
plt.xlabel('X-axis Label')

plt.ylabel('Y-axis Label')

plt.title('Multiple Lines Plot')

plt.show()

**output:**



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

import matplotlib.pyplot as plt

import numpy as np

# Generate some sample data

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

y1 = np.sin(x)

y2 = np.cos(x)

y3 = np.tan(x)

fig, axs = plt.subplots(3, 1, figsize=(8, 6))

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

axs[0].set\_title('Sine Function')

axs[1].plot(x, y2, color='green')

axs[1].set\_title('Cosine Function')

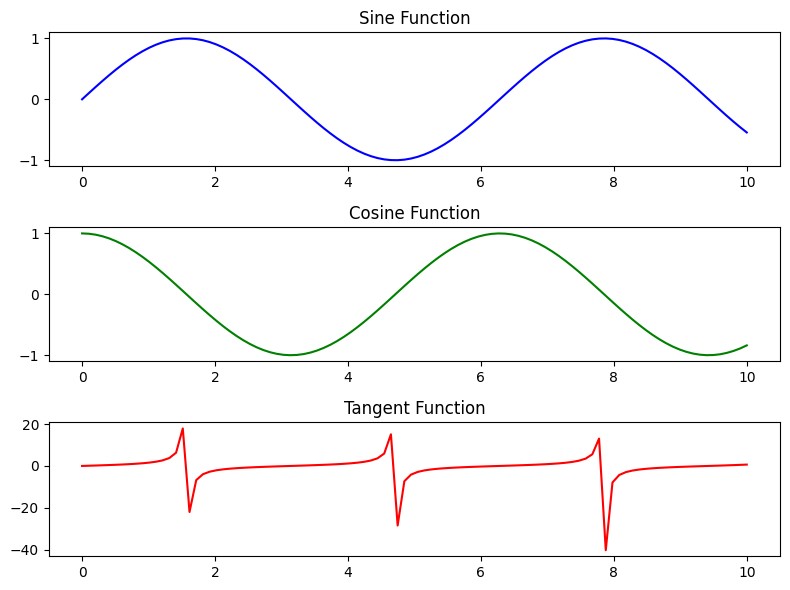
axs[2].plot(x, y3, color='red')

axs[2].set\_title('Tangent Function')

plt.tight\_layout()

plt.show()

**output:**



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

Sample data:

Programming languages: Java, Python, PHP, JavaScript, C#,

C++ Popularity: 22.2, 17.6, 8.8, 8, 7.7, 6.7

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 bar chart

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

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

# Add labels and title

plt.xlabel('Programming Languages')

plt.ylabel('Popularity (%)')

plt.title('Popularity of Programming Languages')

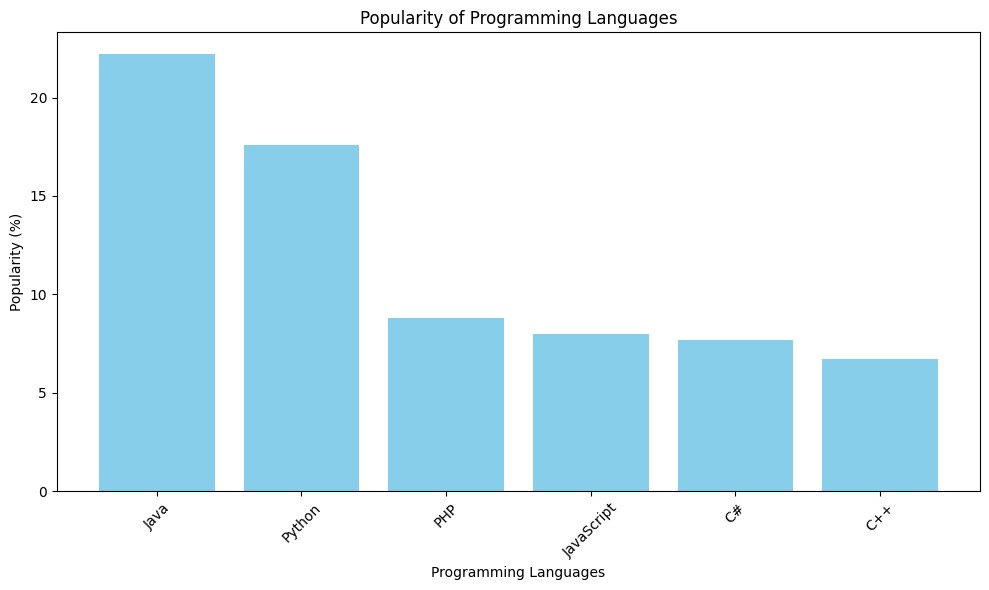
# Show the plot

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

**output:**



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

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

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 horizontal bar chart

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

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

# Add labels and title

plt.xlabel('Popularity (%)')

plt.ylabel('Programming Languages')

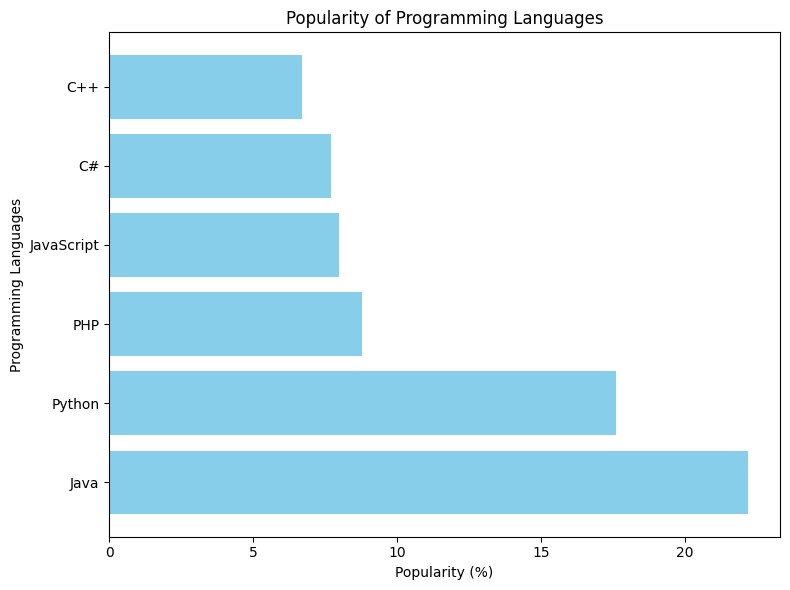
plt.title('Popularity of Programming Languages')

# Show the plot

plt.tight\_layout()

plt.show()

output:



29. Write a Python programming to display a bar chart of the popularity of programming Languages. Use different color for each bar. Sample data:

Programming languages: Java, Python, PHP, JavaScript, C#,

C++ Popularity: 22.2, 17.6, 8.8, 8, 7.7, 6.7

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]

# Define colors for each bar

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

# Create bar chart with different colors

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

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

# Add labels and title

plt.xlabel('Programming Languages')

plt.ylabel('Popularity (%)')

plt.title('Popularity of Programming Languages')

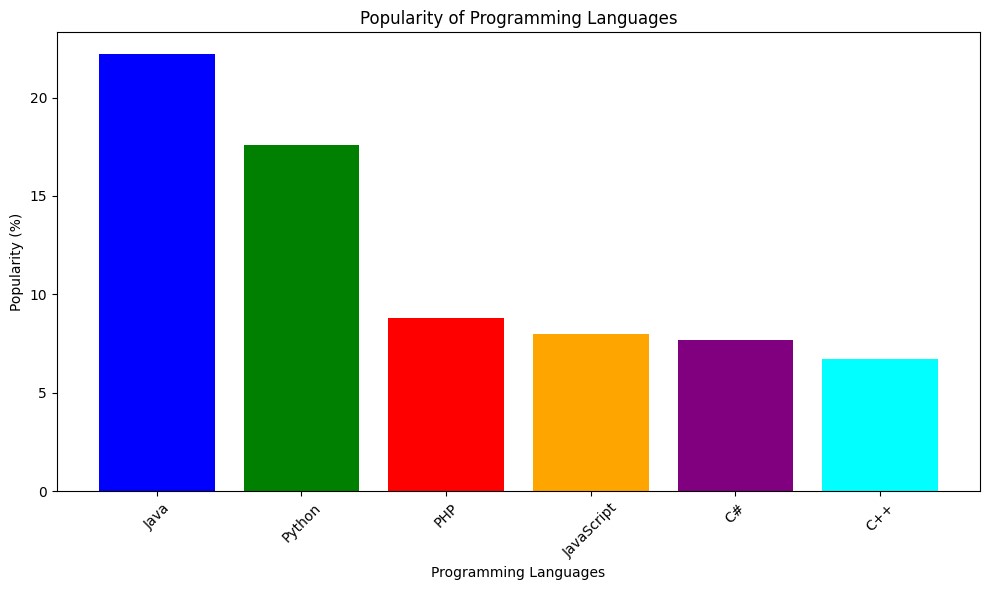
# Show the plot

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

output:



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

Sample Data:

Means (men) = (22, 30, 35, 35, 26)

Means (women) = (25, 32, 30, 35, 29)

import numpy as np

import matplotlib.pyplot as plt

# Sample data

men\_means = (22, 30, 35, 35, 26)

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

# Number of groups

N = len(men\_means)

# Width of each bar

width = 0.35

# Indices for the groups

ind = np.arange(N)

# Create bar plot for men

plt.bar(ind, men\_means, width, label='Men')

# Create bar plot for women, shift the x-values by the width of the bars

plt.bar(ind + width, women\_means, width, label='Women')

# Add labels and title

plt.xlabel('Group')

plt.ylabel('Scores')

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

plt.xticks(ind + width / 2, ('G1', 'G2', 'G3', 'G4', 'G5'))

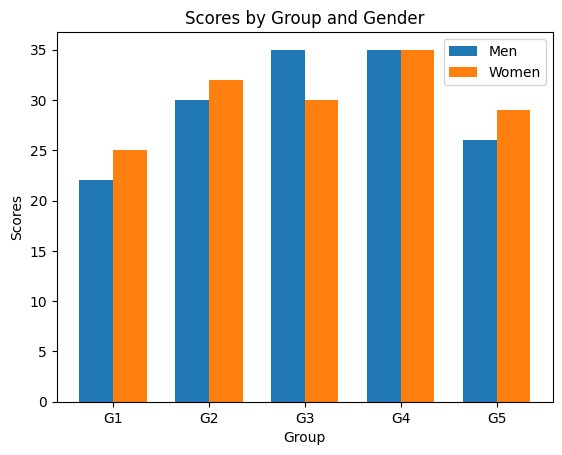
# Add legend

plt.legend()

# Show the plot

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)

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')

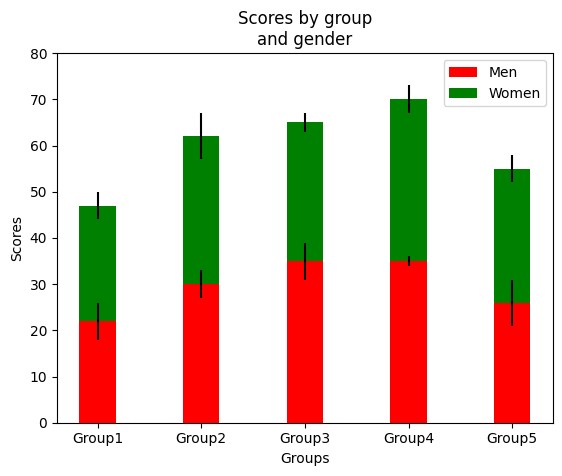
plt.xticks(ind, ('Group1', 'Group2', 'Group3', 'Group4', 'Group5'))

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

import matplotlib.pyplot as plt

import numpy as np

np.random.seed(0)

x = np.random.rand(50)

y = np.random.rand(50)

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

plt.scatter(x, y, color='blue', alpha=0.5)

plt.title('Random Scatter Plot')

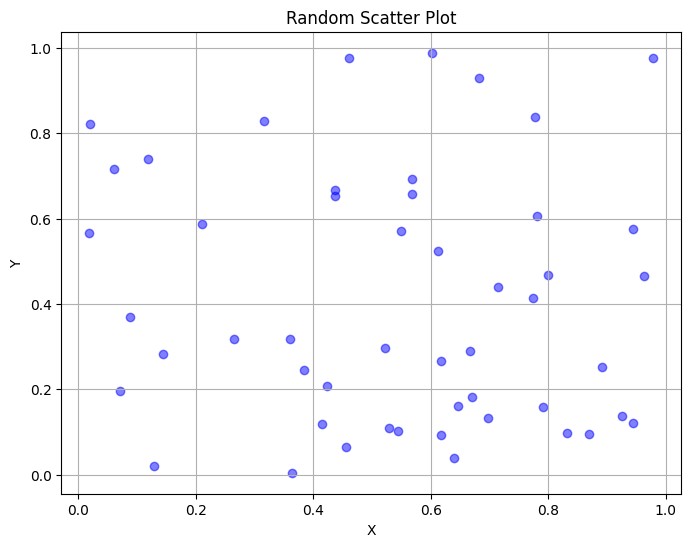
plt.xlabel('X')

plt.ylabel('Y')

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.

import matplotlib.pyplot as plt

import numpy as np

x = np.random.randn(50)

y = np.random.randn(50)

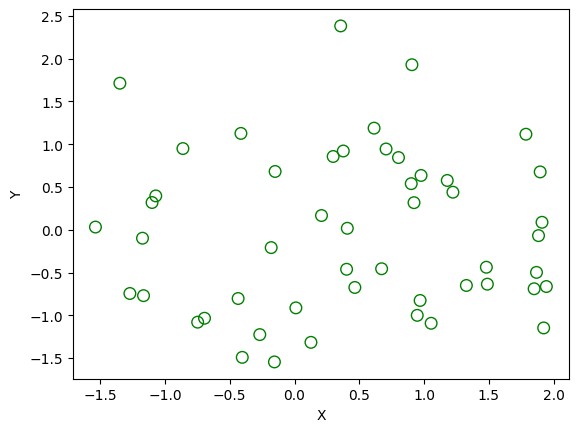
plt.scatter(x, y, s=70, facecolors='none', edgecolors='g')

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.

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)

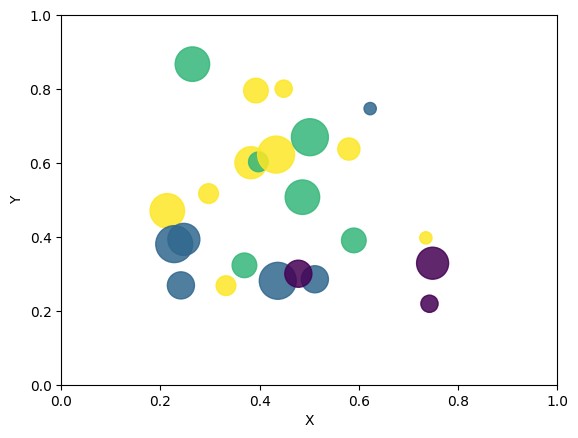
plt.axis([0.0, 1.0, 0.0, 1.0])

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]

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')

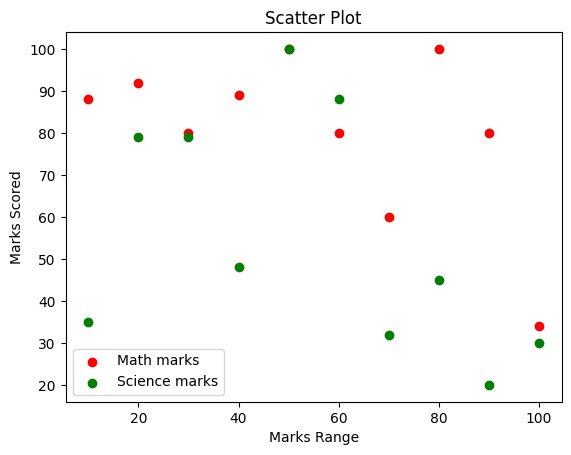
plt.xlabel('Marks Range')

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.

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'])

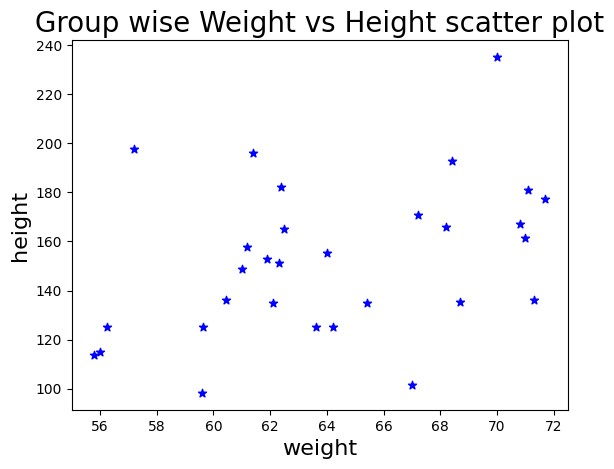
plt.xlabel('weight', fontsize=16)

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]}

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:

X Y Z

0 78 84 86

1 85 94 97

2 96 89 96

3 80 83 72

4 86 86 83

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

Sample Python dictionary data and list labels: exam\_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily',

'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],

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

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

'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']} labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

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:

name score attempts qualify

a Anastasia 12.5 1 yes

b Dima 9.0 3 no

c Katherine 16.5 2 yes

d James NaN 3 no

e Emily 9.0 2 no

f Michael 20.0 3 yes

g Matthew 14.5 1 yes

h Laura NaN 1 no

i Kevin 8.0 2 no

j Jonas 19.0 1 yes

39. Write a Pandas program to get the first 3 rows of a given DataFrame. Sample Python dictionary data and list labels: exam\_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily',

'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],

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

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

'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']} labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

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:

First three rows of the data frame:

name score attempts qualify

a Anastasia 12.5 1 yes

b Dima 9.0 3 no

c Katherine 16.5 2 yes

addCode

addText

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

Sample Python dictionary data and list labels: exam\_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily',

'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],

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

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

'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']} labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

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:

Select specific 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