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

PROGRAM:

import pandas as pd

# Dummy data

data = {'employee\_id': [101, 102, 103, 104, 105],

'employee\_name': ['Alice', 'Bob', 'Charlie', 'David', 'Emma'],

'department\_id': [101, 102, 101, 103, 102]}

# Create DataFrame

employees\_df = pd.DataFrame(data)

# Select distinct department IDs

distinct\_department\_ids = employees\_df['department\_id'].unique()

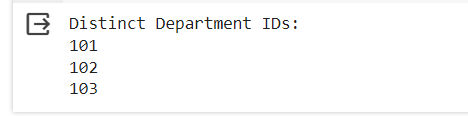
# Print the distinct department IDs

print("Distinct Department IDs:")

for department\_id in distinct\_department\_ids:

print(department\_id)

OUTPUT:



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

# Dummy data

data = {'employee\_id': [101, 102, 103, 104, 105, 101, 103, 104],

'job\_id': ['J1', 'J1', 'J2', 'J3', 'J2', 'J4', 'J5', 'J6']}

# Create DataFrame

jobs\_df = pd.DataFrame(data)

# Group by employee\_id and count the number of unique job\_ids

jobs\_count = jobs\_df.groupby('employee\_id')['job\_id'].nunique()

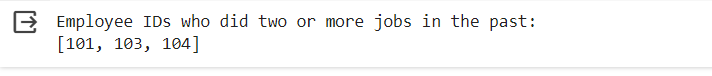
# Filter employees who have had two or more unique jobs

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

# Display the employee IDs

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

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



3. Write a Pandas program to display the details of jobs in descending

sequence on job title.

import pandas as pd

# Dummy data

data = {'job\_title': ['Manager', 'Engineer', 'Analyst', 'Developer'],

'salary': [80000, 60000, 55000, 70000],

'department': ['HR', 'Engineering', 'Finance', 'Engineering']}

# Create DataFrame

jobs\_df = pd.DataFrame(data)

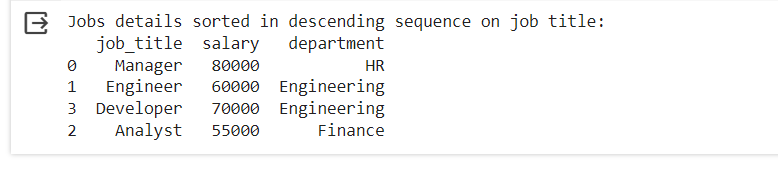
# Sort jobs in descending sequence based on job title

sorted\_jobs = jobs\_df.sort\_values(by='job\_title', ascending=False)

# Display the sorted job details

print("Jobs details sorted in descending sequence on job title:")

print(sorted\_jobs)



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 numpy as np

import matplotlib.pyplot as plt

# Generate sample data

dates = pd.date\_range(start='2023-01-01', end='2023-12-31', freq='D')

prices = np.random.normal(loc=100, scale=10, size=len(dates))

# Create DataFrame

stock\_data = pd.DataFrame({'Date': dates, 'Close': prices})

# Plot the line plot

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

plt.plot(stock\_data['Date'], stock\_data['Close'], color='blue', linestyle='-')

# Set plot title and labels

plt.title('Sample Historical Stock Prices of Alphabet Inc. (2023)')

plt.xlabel('Date')

plt.ylabel('Closing Price ($)')

# Rotate x-axis labels for better readability

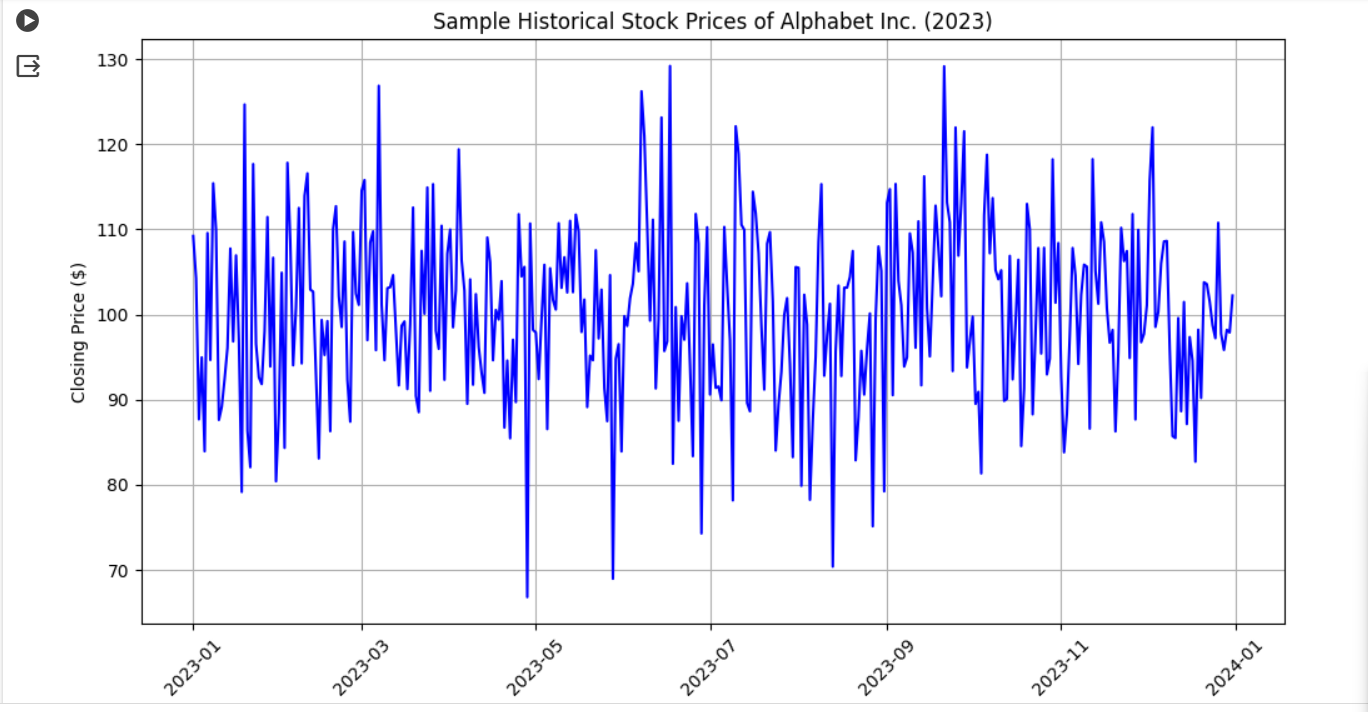
plt.xticks(rotation=45)

# Display the plot

plt.grid(True)

plt.tight\_layout()

plt.show()



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

dates = pd.date\_range(start='2023-01-01', end='2023-12-31', freq='D')

volume = np.random.randint(low=100000, high=1000000, size=len(dates))

# Create DataFrame

stock\_data = pd.DataFrame({'Date': dates, 'Volume': volume})

# Filter data between two specific dates (e.g., '2023-01-01' and '2023-12-31')

start\_date = '2023-01-01'

end\_date = '2023-12-31'

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

# Plot the bar plot

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

plt.bar(filtered\_data['Date'], filtered\_data['Volume'], color='green')

# Set plot title and labels

plt.title('Trading Volume of Alphabet Inc. Stock ({} to {})'.format(start\_date, end\_date))

plt.xlabel('Date')

plt.ylabel('Volume')

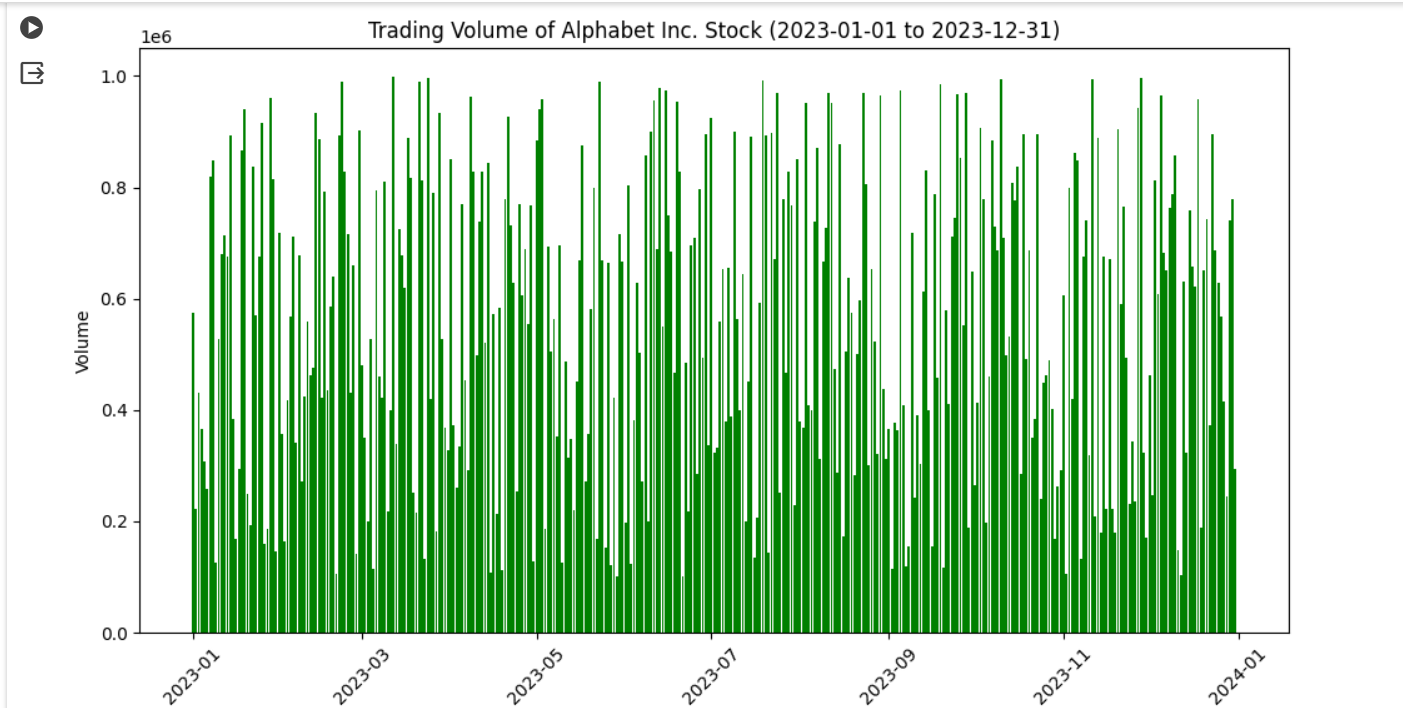
# Rotate x-axis labels for better readability

plt.xticks(rotation=45)

# Display the plot

plt.tight\_layout()

plt.show()



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 numpy as np

import matplotlib.pyplot as plt

# Generate sample data for demonstration

dates = pd.date\_range(start='2023-01-01', end='2023-12-31', freq='D')

prices = np.random.normal(loc=100, scale=10, size=len(dates))

volume = np.random.randint(low=100000, high=1000000, size=len(dates))

# Create DataFrame

stock\_data = pd.DataFrame({'Date': dates, 'Price': prices, 'Volume': volume})

# Filter data between two specific dates (e.g., '2023-01-01' and '2023-12-31')

start\_date = '2023-01-01'

end\_date = '2023-12-31'

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

# Plot the scatter plot

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

plt.scatter(filtered\_data['Volume'], filtered\_data['Price'], color='blue')

# Set plot title and labels

plt.title('Trading Volume vs. Stock Prices of Alphabet Inc. Stock ({} to {})'.format(start\_date, end\_date))

plt.xlabel('Volume')

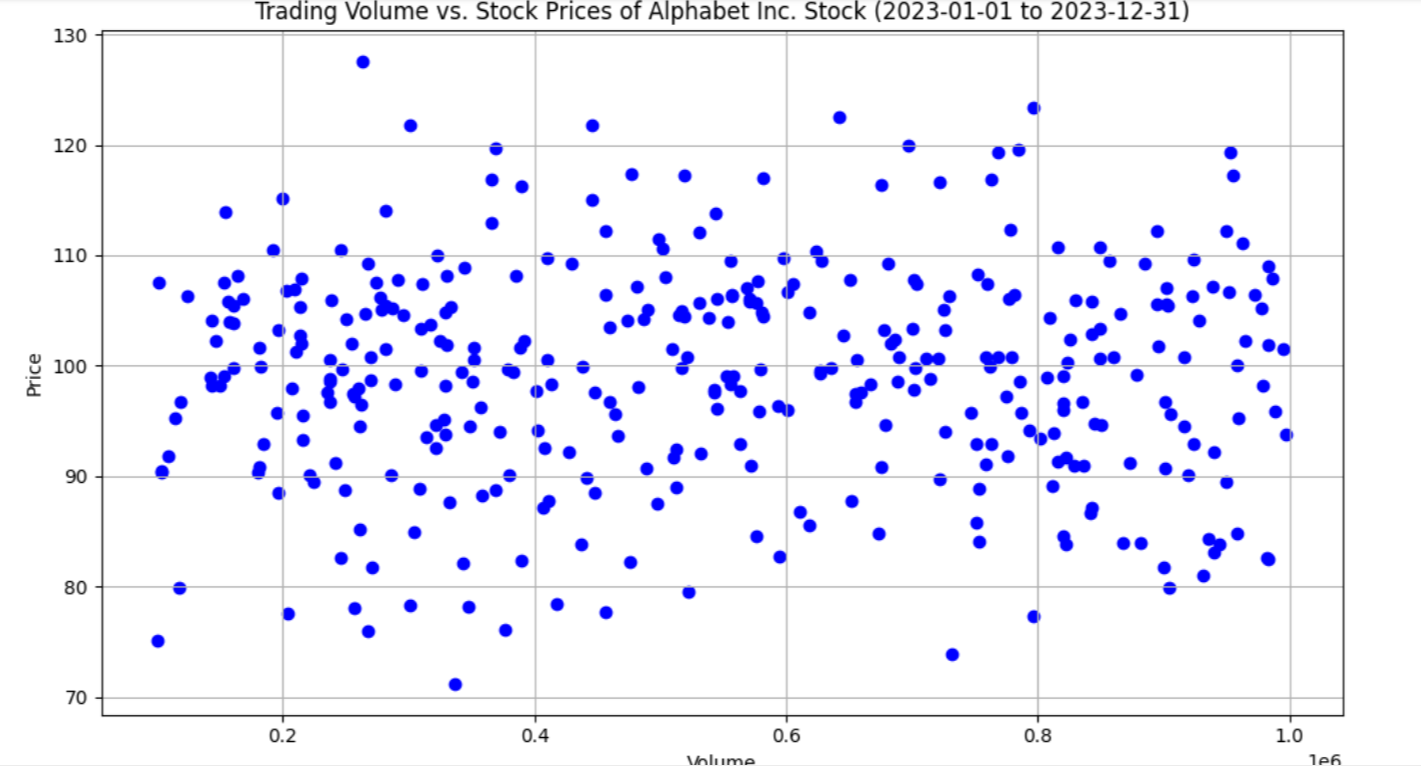
plt.ylabel('Price')

# Display the plot

plt.grid(True)

plt.tight\_layout()

plt.show()



7Write a Pandas program to create a Pivot table and find the maximum

and minimum sale value of the items.(refer sales\_data table) 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 = {

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

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

'Unit': [10, 15, 20, 12, 18, 25]

}

# Create DataFrame

df = pd.DataFrame(sales\_data)

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

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

# Display the pivot table

print("Maximum and minimum sale value of the items:")

print(pivot\_max\_min\_sale)

print()

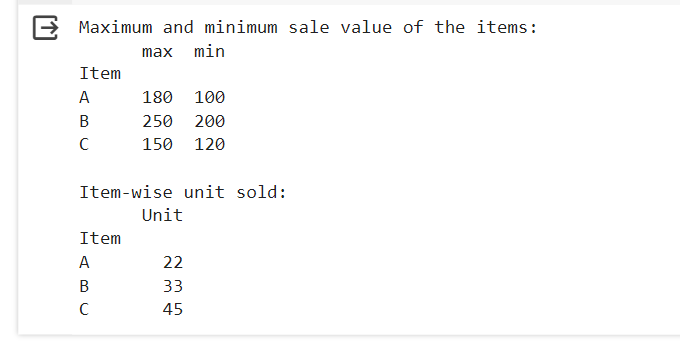
# Pivot table to find item-wise unit sold

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

# Display the pivot table

print("Item-wise unit sold:")

print(pivot\_unit\_sold)



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

# Sample sales data

sales\_data = {

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

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

'Salesperson': ['Amy', 'Bob', 'Cathy', 'David', 'Eva', 'Frank'],

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

}

# Create DataFrame

df = pd.DataFrame(sales\_data)

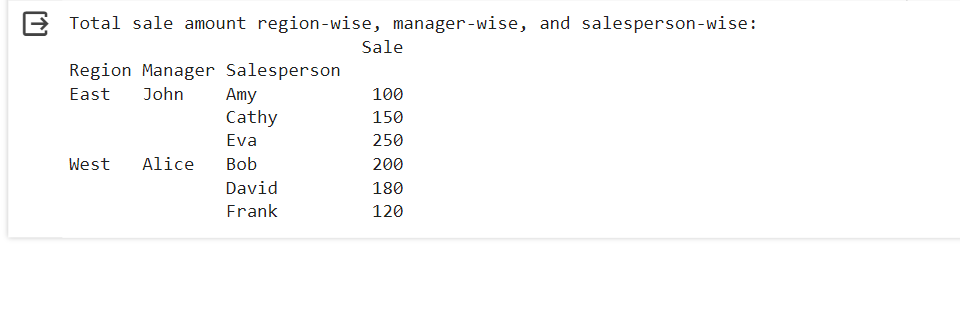
# Pivot table to find total sale amount region-wise, manager-wise, and salesperson-wise

pivot\_total\_sales = df.pivot\_table(index=['Region', 'Manager', 'Salesperson'], values='Sale', aggfunc='sum')

# Display the pivot table

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

print(pivot\_total\_sales)



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 ten rows and four columns with random values

np.random.seed(0) # For reproducibility

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

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

# Define a function to apply color based on the value

def color\_negative\_red(val):

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

return 'color: {}'.format(color)

# Apply the styling based on the function

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

# Display the styled DataFrame

styled\_df



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 ten rows and four columns with random values

np.random.seed(0) # For reproducibility

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

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

# Convert some random values to NaN

nan\_indices = [(i, j) for i in range(10) for j in range(4) if np.random.random() < 0.3] # 30% of values will be NaN

for i, j in nan\_indices:

df.iloc[i, j] = np.nan

# Define a function to apply color to NaN values

def highlight\_nan(val):

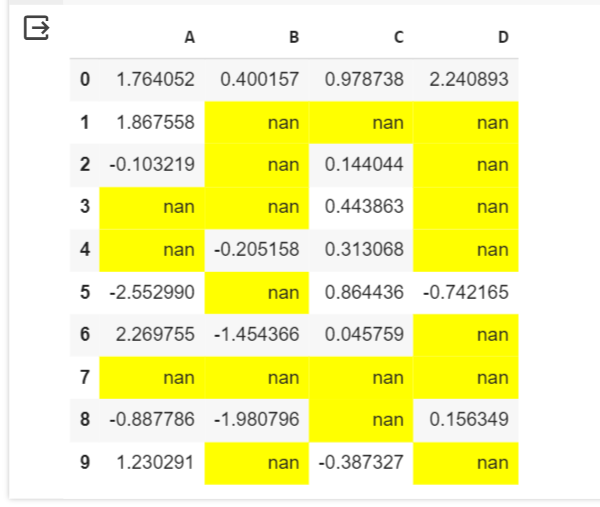
return ['background-color: yellow' if pd.isna(v) else '' for v in val]

# Apply the styling based on the function

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

# Display the styled DataFrame

styled\_df



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 ten rows and four columns with random values

np.random.seed(0) # For reproducibility

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

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

# Define a function to apply background and font color

def set\_colors(val):

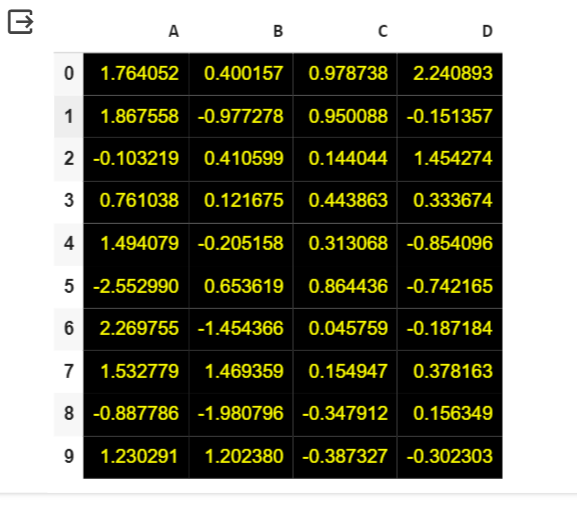
return 'background-color: black; color: yellow'

# Apply the styling based on the function

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

# Display the styled DataFrame

styled\_df



13.Write a Pandas program to detect missing values of a given DataFrame.

Display True or False.

import pandas as pd

# Sample DataFrame with missing values

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

'B': [5, None, 7, 8],

'C': [None, 10, 11, 12]}

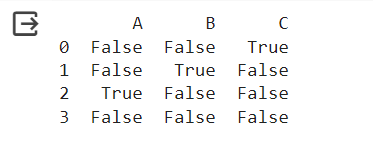
df = pd.DataFrame(data)

# Detect missing values

missing\_values = df.isna()

# Display True or False for each element indicating whether it's a missing value or not

print(missing\_values)



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

# Sample DataFrame with missing values

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

'B': [5, None, 7, 8],

'C': [None, 10, 11, 12]}

df = pd.DataFrame(data)

# Find and replace missing values

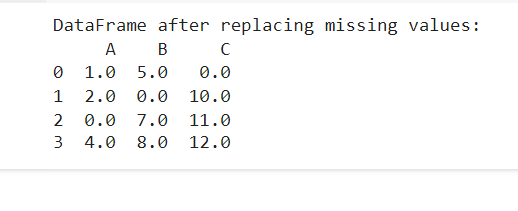
# For example, replace missing values with 0

df\_filled = df.fillna(0)

# Display the DataFrame after replacing missing values

print("DataFrame after replacing missing values:")

print(df\_filled)



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

# Sample DataFrame with missing values

data = {'A': [1, np.nan, None, 4],

'B': [np.nan, None, 7, 8],

'C': [None, 10, 11, np.nan]}

df = pd.DataFrame(data)

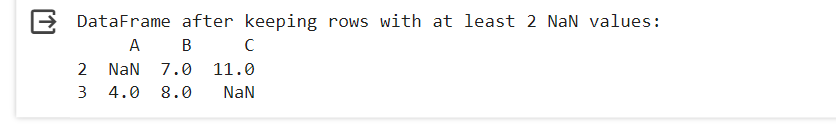
# Keep rows with at least 2 NaN values

df\_filtered = df.dropna(thresh=2)

# Display the DataFrame after keeping rows with at least 2 NaN values

print("DataFrame after keeping rows with at least 2 NaN values:")

print(df\_filtered)



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

# Sample DataFrame

data = {

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

'Student Name': ['Alice', 'Bob', 'Charlie', 'David', 'Emma'],

'Grade': [85, 92, 88, 78, 90]

}

df = pd.DataFrame(data)

# Group the DataFrame by school code

grouped = df.groupby('School Code')

# Check the type of GroupBy object

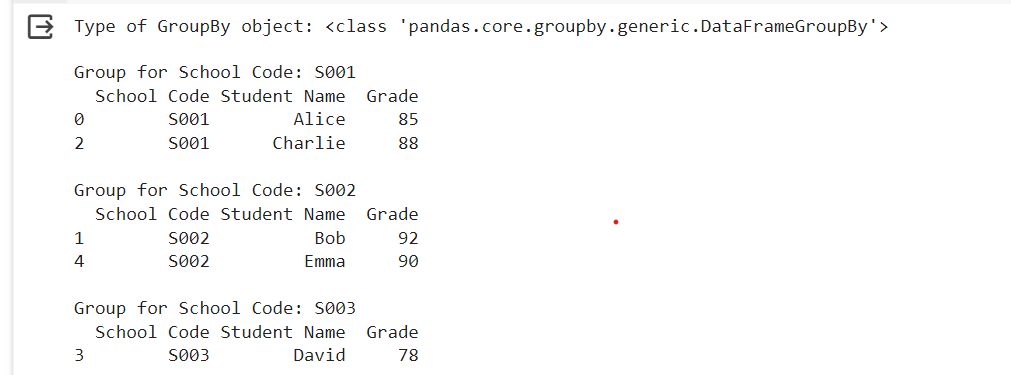
print("Type of GroupBy object:", type(grouped))

# Display the groups

for school\_code, group in grouped:

print("\nGroup for School Code:", school\_code)

print(group)



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

# Sample DataFrame

data = {

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

'Student Name': ['Alice', 'Bob', 'Charlie', 'David', 'Emma'],

'Age': [18, 17, 18, 16, 17]

}

df = pd.DataFrame(data)

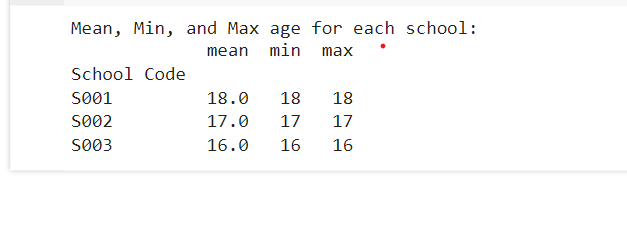
# Split the DataFrame by school code and calculate mean, min, and max age for each school

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

# Display the result

print("Mean, Min, and Max age for each school:")

print(result)



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 = {

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

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

'Student Name': ['Alice', 'Bob', 'Charlie', 'David', 'Emma'],

'Age': [18, 17, 18, 16, 17]

}

df = pd.DataFrame(data)

# Split the DataFrame by school code and class

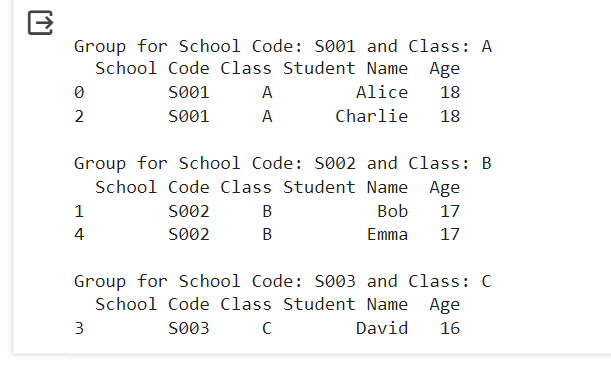
grouped = df.groupby(['School Code', 'Class'])

# Display the groups

for (school\_code, class\_name), group in grouped:

print("\nGroup for School Code:", school\_code, "and Class:", class\_name)

print(group)



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

# Example DataFrame (replace this with your actual DataFrame)

data = {

'Column1': [1, 2, 3],

'Column2': ['A', 'B', 'C'],

'Column3': [True, False, True]

}

df = pd.DataFrame(data)

# Display the dimensions (shape) of the DataFrame

print("Dimensions of the DataFrame:")

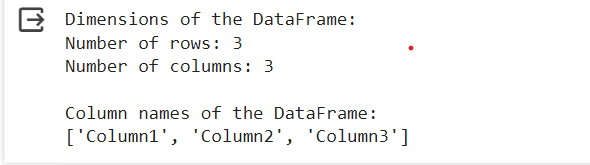
print("Number of rows:", df.shape[0])

print("Number of columns:", df.shape[1])

# Extract and display the column names

print("\nColumn names of the DataFrame:")

print(df.columns.tolist())



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

DataFrame column.

import pandas as pd

# Sample DataFrame

data = {

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

'Column2': [1, 2, 3, 4, 5]

}

df = pd.DataFrame(data)

# Find the index of rows containing a given substring in a DataFrame column

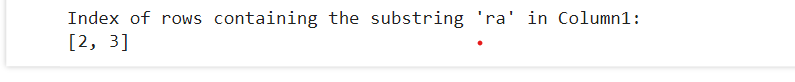
substring = 'ra'

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

# Display the index of rows containing the substring

print("Index of rows containing the substring '{}' in Column1:".format(substring))

print(index\_list)



21 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 = {'Char\_Column': ['Hello', 'World', 'Python', 'DataScience']}

df = pd.DataFrame(data)

# Function to swap case of characters in a column

def swap\_case(column):

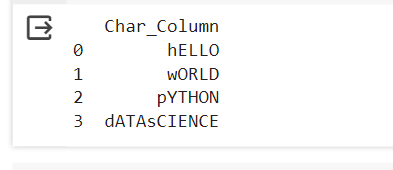
    return column.str.swapcase()

# Swap case of specified character column

df['Char\_Column'] = swap\_case(df['Char\_Column'])

# Display the DataFrame with swapped cases

print(df)



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

# Data for the line plot

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

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

# Plot the line

plt.plot(x, y)

# Add labels and title

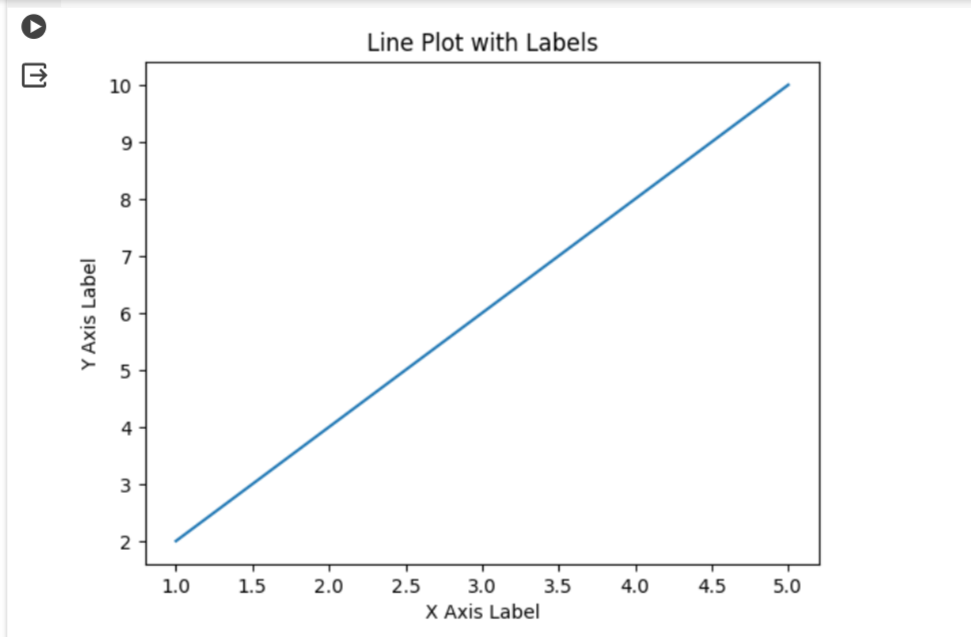
plt.xlabel('X Axis Label')

plt.ylabel('Y Axis Label')

plt.title('Line Plot with Labels')

# Display the plot

plt.show()



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

# Sample data from the text file

data = [

    (1, 2),

    (2, 4),

    (3, 1)

]

# Separate x and y values

x\_values = [point[0] for point in data]

y\_values = [point[1] for point in data]

# Plotting the line

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

# Adding labels and title

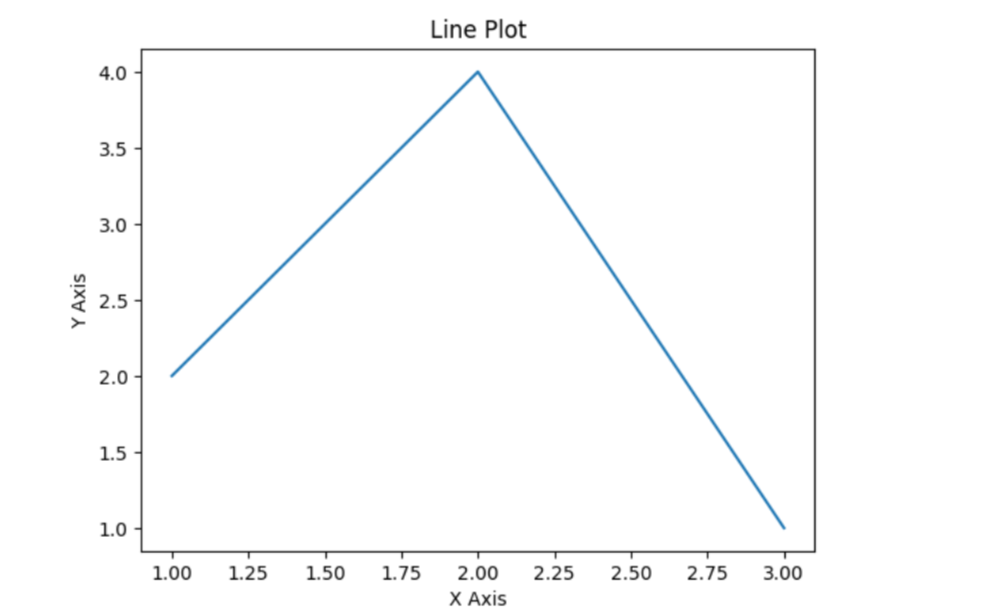
plt.xlabel('X Axis')

plt.ylabel('Y Axis')

plt.title('Line Plot')

# Displaying the plot

plt.show()



24.Write 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

# Sample Financial data

data = {

    'Date': ['10-03-16', '10-04-16', '10-05-16', '10-06-16', '10-07-16'],

    'Open': [774.25, 776.030029, 779.309998, 779, 779.659973],

    'High': [776.065002, 778.710022, 782.070007, 780.47998, 779.659973],

    'Low': [769.5, 772.890015, 775.650024, 775.539978, 770.75],

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

}

# Convert data to DataFrame

df = pd.DataFrame(data)

# Convert 'Date' column to datetime

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

# Filter data for the specified date range

start\_date = '2016-10-03'

end\_date = '2016-10-07'

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

# Plot line charts

plt.plot(filtered\_df['Date'], filtered\_df['Open'], label='Open')

plt.plot(filtered\_df['Date'], filtered\_df['High'], label='High')

plt.plot(filtered\_df['Date'], filtered\_df['Low'], label='Low')

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

# Add labels and title

plt.xlabel('Date')

plt.ylabel('Price')

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

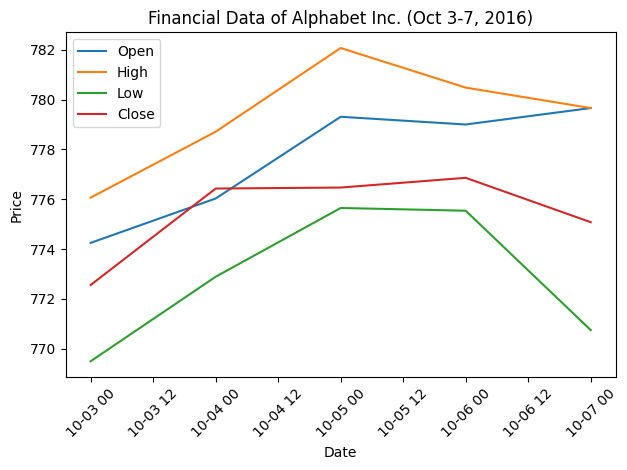
plt.xticks(rotation=45)

plt.legend()

# Display the plot

plt.tight\_layout()

plt.show()



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

widths and colors.

import matplotlib.pyplot as plt

# Data for the lines

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

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

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

y3 = [3, 6, 9, 12, 15]

# Plot the lines with different widths and colors, and add legends

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

plt.plot(x, y2, label='Line 2', color='red', linewidth=1.5)

plt.plot(x, y3, label='Line 3', color='green', linewidth=3)

# Add legend

plt.legend()

# Add labels and title

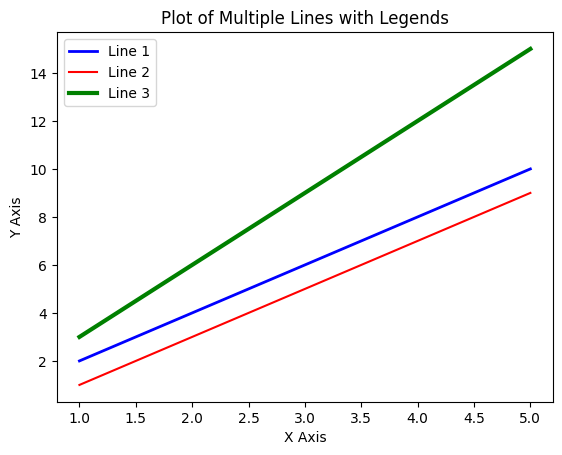
plt.xlabel('X Axis')

plt.ylabel('Y Axis')

plt.title('Plot of Multiple Lines with Legends')

# Display the plot

plt.show()



26.Write a Python program to create multiple plots.

import matplotlib.pyplot as plt

fig = plt.figure()

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

plt.subplot(2, 1, 1)

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

plt.subplot(2, 3, 4)

plt.xticks(())

plt.yticks(())

plt.subplot(2, 3, 5)

plt.xticks(())

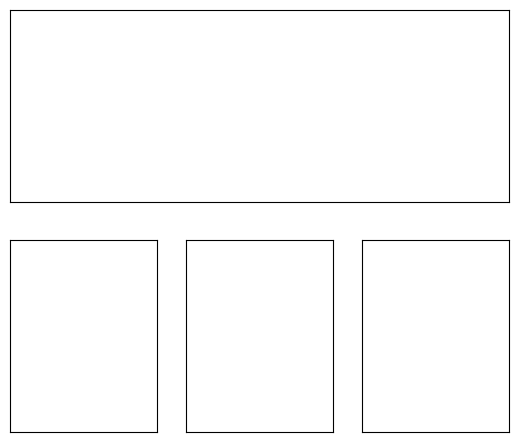
plt.yticks(())

plt.subplot(2, 3, 6)

plt.xticks(())

plt.yticks(())

plt.show()



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.bar(languages, popularity, color='red')

# Add labels and title

plt.xlabel('Programming Languages')

plt.ylabel('Popularity (%)')

plt.title('Popularity of Programming Languages')

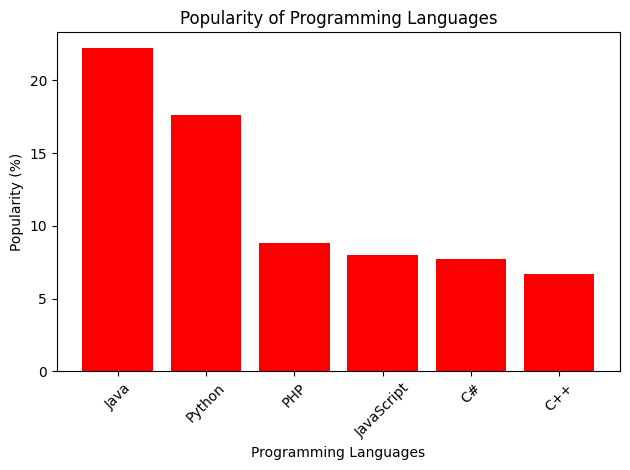
# Rotate x-axis labels for better readability

plt.xticks(rotation=45)

# Display the bar chart

plt.tight\_layout()

plt.show()



28.Write 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.barh(languages, popularity, color='lightgreen')

# Add labels and title

plt.xlabel('Popularity (%)')

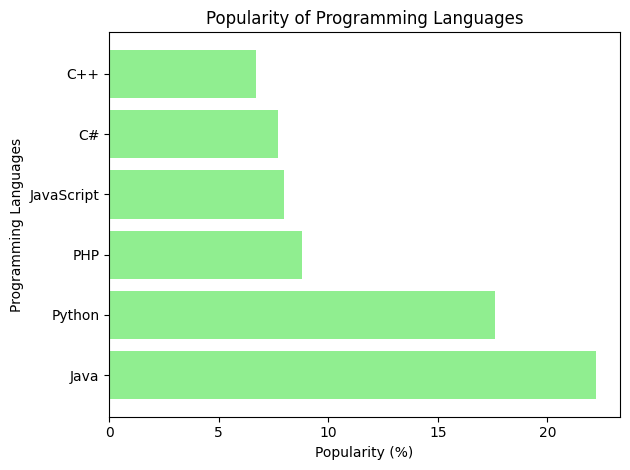
plt.ylabel('Programming Languages')

plt.title('Popularity of Programming Languages')

# Display the horizontal bar chart

plt.tight\_layout()

plt.show()



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#,

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]

colors = ['skyblue', 'orange', 'green', 'red', 'purple', 'pink']

# Create a bar chart with different colors for each bar

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

# Add labels and title

plt.xlabel('Programming Languages')

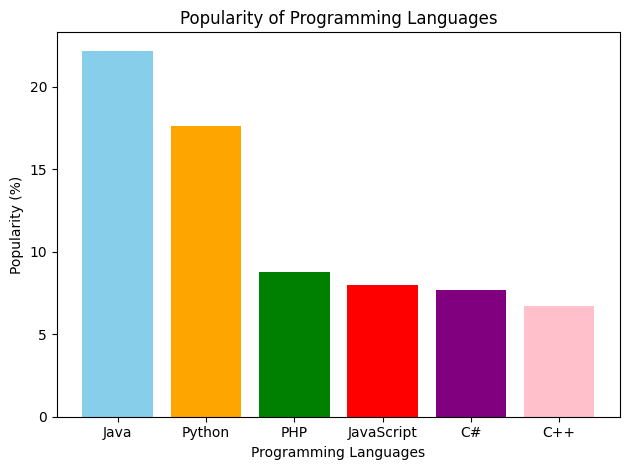
plt.ylabel('Popularity (%)')

plt.title('Popularity of Programming Languages')

# Display the bar chart

plt.tight\_layout()

plt.show()



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 numpy as np

import matplotlib.pyplot as plt

# Sample Data

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

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

groups = np.arange(len(men\_means))  # the x locations for the groups

# Width of the bars

bar\_width = 0.35

# Create bar plot

fig, ax = plt.subplots()

bars1 = ax.bar(groups - bar\_width/2, men\_means, bar\_width, label='Men')

bars2 = ax.bar(groups + bar\_width/2, women\_means, bar\_width, label='Women')

# Add labels, title, and legend

ax.set\_xlabel('Groups')

ax.set\_ylabel('Scores')

ax.set\_title('Scores by group and gender')

ax.set\_xticks(groups)

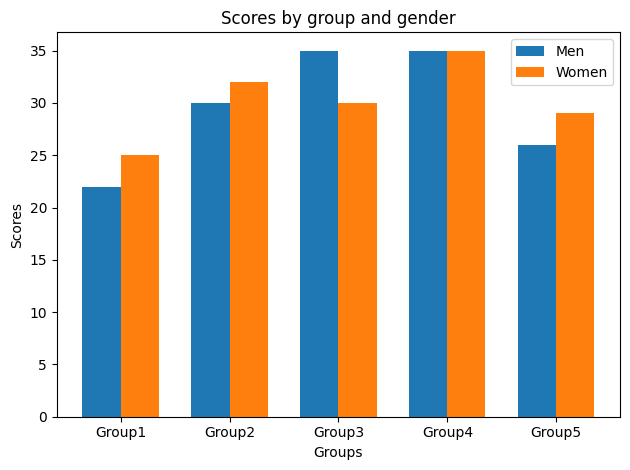
ax.set\_xticklabels(['Group1', 'Group2', 'Group3', 'Group4', 'Group5'])

ax.legend()

# Display the plot

plt.tight\_layout()

plt.show()



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.

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)

men\_std = (4, 3, 4, 1, 5)

women\_std = (3, 5, 2, 3, 3)

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

x = np.arange(len(labels))  # the label locations

width = 0.35  # the width of the bars

fig, ax = plt.subplots()

# Plot men's data

rects1 = ax.bar(x, men\_means, width, yerr=men\_std, label='Men')

# Plot women's data on top of men's data

rects2 = ax.bar(x, women\_means, width, yerr=women\_std, bottom=men\_means, label='Women')

# Add some text for labels, title and custom x-axis tick labels, etc.

ax.set\_ylabel('Scores')

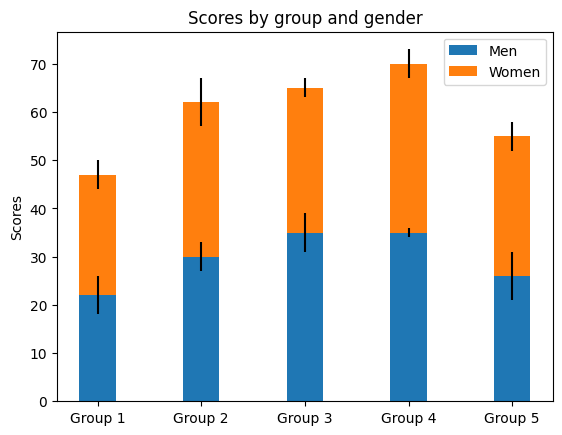
ax.set\_title('Scores by group and gender')

ax.set\_xticks(x)

ax.set\_xticklabels(labels)

ax.legend()

plt.show()



32.Write a Python program to draw a scatter graph taking a random

distribution in X and Y and plotted against each other.

import numpy as np

import matplotlib.pyplot as plt

# Generating random data

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

num\_points = 100

x = np.random.randn(num\_points)

y = np.random.randn(num\_points)

# Creating scatter plot

plt.scatter(x, y)

# Adding labels and title

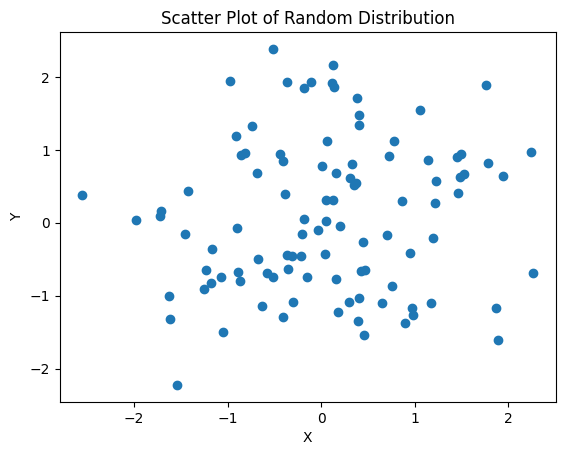
plt.xlabel('X')

plt.ylabel('Y')

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

# Displaying the plot

plt.show()



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 numpy as np

import matplotlib.pyplot as plt

# Generating random data

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

num\_points = 100

x = np.random.randn(num\_points)

y = np.random.randn(num\_points)

# Creating scatter plot with empty circles

plt.scatter(x, y, facecolors='none', edgecolors='blue')

# Adding labels and title

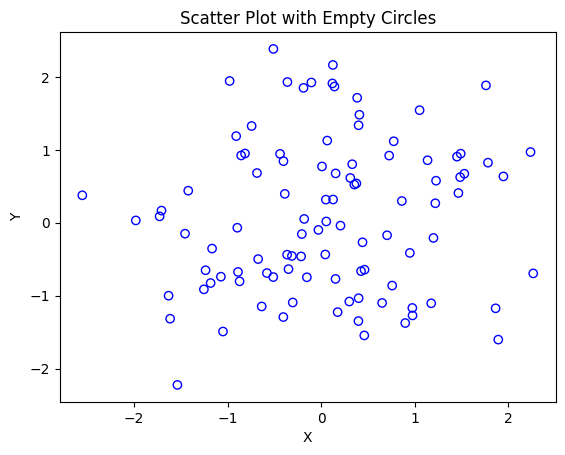
plt.xlabel('X')

plt.ylabel('Y')

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

# Displaying the plot

plt.show()



34.Write a Python program to draw a scatter plot using random distributions to

generate balls of different sizes.

import numpy as np

import matplotlib.pyplot as plt

# Generating random data

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

num\_points = 50

x = np.random.rand(num\_points) \* 100  # Generating random x-coordinates

y = np.random.rand(num\_points) \* 100  # Generating random y-coordinates

sizes = np.random.rand(num\_points) \* 1000  # Generating random sizes for the balls

# Creating scatter plot with balls of different sizes

plt.scatter(x, y, s=sizes, alpha=0.5)

# Adding labels and title

plt.xlabel('X')

plt.ylabel('Y')

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

# Displaying the plot

plt.show()

import numpy as np

import matplotlib.pyplot as plt

# Generating random data for three different groups

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

# Group 1 data

num\_points = 50

group1\_heights = np.random.normal(160, 10, num\_points)

group1\_weights = np.random.normal(60, 5, num\_points)

# Group 2 data

group2\_heights = np.random.normal(170, 12, num\_points)

group2\_weights = np.random.normal(70, 6, num\_points)

# Group 3 data

group3\_heights = np.random.normal(165, 8, num\_points)

group3\_weights = np.random.normal(65, 4, num\_points)

# Creating scatter plot for three different groups

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

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

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

# Adding labels and title

plt.xlabel('Height (cm)')

plt.ylabel('Weight (kg)')

plt.title('Scatter Plot for Three Different Groups')

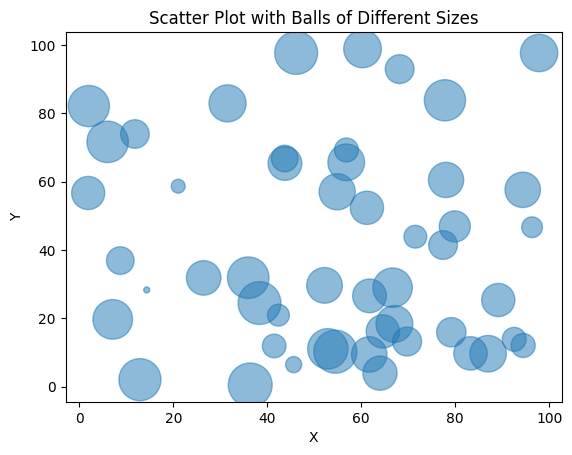
# Adding legend

plt.legend()

# Displaying the plot

plt.grid(True)

plt.show()



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:

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]

# Creating scatter plot comparing Mathematics and Science marks

plt.scatter(marks\_range, math\_marks, color='blue', label='Mathematics')

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

# Adding labels and title

plt.xlabel('Marks Range')

plt.ylabel('Marks Scored')

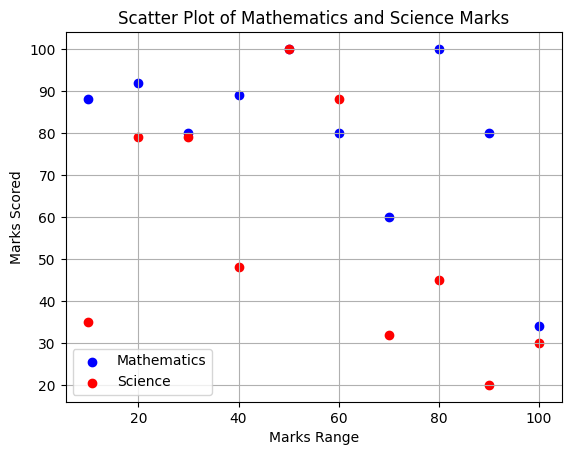
plt.title('Scatter Plot of Mathematics and Science Marks')

plt.legend()

# Displaying the plot

plt.grid(True)

plt.show()



36.Write a Python program to draw a scatter plot for three different groups

comparing weights and heights.

import numpy as np

import matplotlib.pyplot as plt

# Generating random data for three different groups

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

# Group 1 data

num\_points = 50

group1\_heights = np.random.normal(160, 10, num\_points)

group1\_weights = np.random.normal(60, 5, num\_points)

# Group 2 data

group2\_heights = np.random.normal(170, 12, num\_points)

group2\_weights = np.random.normal(70, 6, num\_points)

# Group 3 data

group3\_heights = np.random.normal(165, 8, num\_points)

group3\_weights = np.random.normal(65, 4, num\_points)

# Creating scatter plot for three different groups

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

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

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

# Adding labels and title

plt.xlabel('Height (cm)')

plt.ylabel('Weight (kg)')

plt.title('Scatter Plot for Three Different Groups')

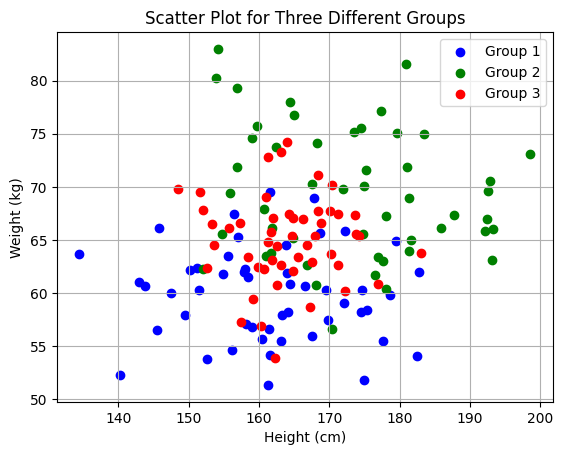
# Adding legend

plt.legend()

# Displaying the plot

plt.grid(True)

plt.show()



37.Write a Pandas program to create a dataframe from a dictionary and

display it.

import pandas as pd

# Sample data

sample\_data = {'X': [78, 85, 96, 80, 86],

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

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

# Creating DataFrame

df = pd.DataFrame(sample\_data)

# Displaying the DataFrame

print("DataFrame:")

print(df)

DataFrame:

X Y Z

0 78 84 86

1 85 94 97

2 96 89 96

3 80 83 72

4 86 86 83

38.Write a Pandas program to create and display a DataFrame from a

specified dictionary data which has the index labels.

import pandas as pd

import numpy as np

# Sample data and 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']

# Creating DataFrame with specified index labels

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

# Displaying the DataFrame

print("DataFrame with specified index labels:")

print(df)

DataFrame with specified index labels:

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:

import pandas as pd

import numpy as np

# Sample data and 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']

# Creating DataFrame with specified index labels

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

# Getting the first 3 rows

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

# Displaying the first 3 rows

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

print(first\_3\_rows)

First 3 rows of the DataFrame:

name score attempts qualify

a Anastasia 12.5 1 yes

b Dima 9.0 3 no

c Katherine 16.5 2 yes

40. Write a Pandas program to select the &#39;name&#39; and &#39;score&#39; columns from

the following DataFrame.

import pandas as pd

import numpy as np

# Sample data and 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']

# Creating DataFrame with specified index labels

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

# Selecting 'name' and 'score' columns

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

# Displaying the selected columns

print("Selected columns ('name' and 'score') from the DataFrame:")

print(selected\_columns)

Selected columns ('name' and 'score') from the DataFrame:

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

THE END