

**Session: 2022 – 2026**

**Submitted by:**

**ARBAZ KHAN 2022-CS-123**

**Supervised by:**

**Prof. Nauman Shafi**

**Department of Computer Science**

**University of Engineering and Technology**

**Lahore, Pakistan**

# Case Study 1: Weather Data Analysis

**Code:**

import matplotlib.pyplot as plt

import pandas as pd

import numpy as np

# Initialize the number of days

num\_of\_days = 365

# Generate date range

np.random.seed(10)

dates = pd.date\_range(start="2023-01-01", periods=num\_of\_days)

# Generate random data for weather conditions

temprature = np.random.uniform(10, 40, size=num\_of\_days)

humidity = np.random.uniform(30, 90, size=num\_of\_days)

wind\_speed = np.random.uniform(0, 20, size=num\_of\_days)

weather\_condition = np.random.choice(["sunny", "rainy", "cloudy"], size=num\_of\_days)

# Create a DataFrame

weather\_data = pd.DataFrame({

    'Date': dates,

    'Temprature (°C)': temprature,

    'Humidity (%)': humidity,

    'Wind Speed (km/h)': wind\_speed,

    'Weather Condition': weather\_condition

})

# Display first few rows of the dataset

print(weather\_data.head())

# Convert temperature column to numpy array

temp\_array = weather\_data['Temprature (°C)'].to\_numpy()

# Calculate mean, median, and standard deviation

temp\_mean = np.mean(temp\_array)

temp\_median = np.median(temp\_array)

temp\_standard\_deviation = np.std(temp\_array)

# Display statistics

print("Mean of the temperature: ", temp\_mean)

print("Median of the temperature: ", temp\_median)

print("Standard Deviation of the temperature: ", temp\_standard\_deviation)

# Filter the data for sunny days with temperature > 30°C

filter\_temp = weather\_data[(weather\_data['Temprature (°C)'] > 30) & (weather\_data['Weather Condition'] == 'sunny')]

total\_days = len(filter\_temp)

# Display number of filtered sunny days

print("Number of Sunny days: ", total\_days)

# Group the data by weather condition and calculate average humidity

average\_humidity = weather\_data.groupby('Weather Condition')['Humidity (%)'].mean()

print("Average humidity by weather condition: \n", average\_humidity)

# Plot the temperature variation over the year

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

plt.plot(weather\_data['Date'], weather\_data['Temprature (°C)'], color='blue', label='Temperature (°C)')

# Adding labels and title

plt.title('Temperature Variation Over the Year', fontsize=16)

plt.xlabel('Date', fontsize=12)

plt.ylabel('Temperature (°C)', fontsize=12)

plt.grid(True)

plt.legend()

plt.xticks(rotation=45)

# Show the plot

plt.tight\_layout()

plt.show()

# Create a bar plot for the number of days for each weather condition

weather\_condition\_counts = weather\_data['Weather Condition'].value\_counts()

# Plotting the bar chart

plt.figure(figsize=(8, 5))

weather\_condition\_counts.plot(kind='bar', color=['yellow', 'lightblue', 'gray'])

# Adding labels and title

plt.title('Number of Days for Each Weather Condition', fontsize=16)

plt.xlabel('Weather Condition', fontsize=12)

plt.ylabel('Number of Days', fontsize=12)

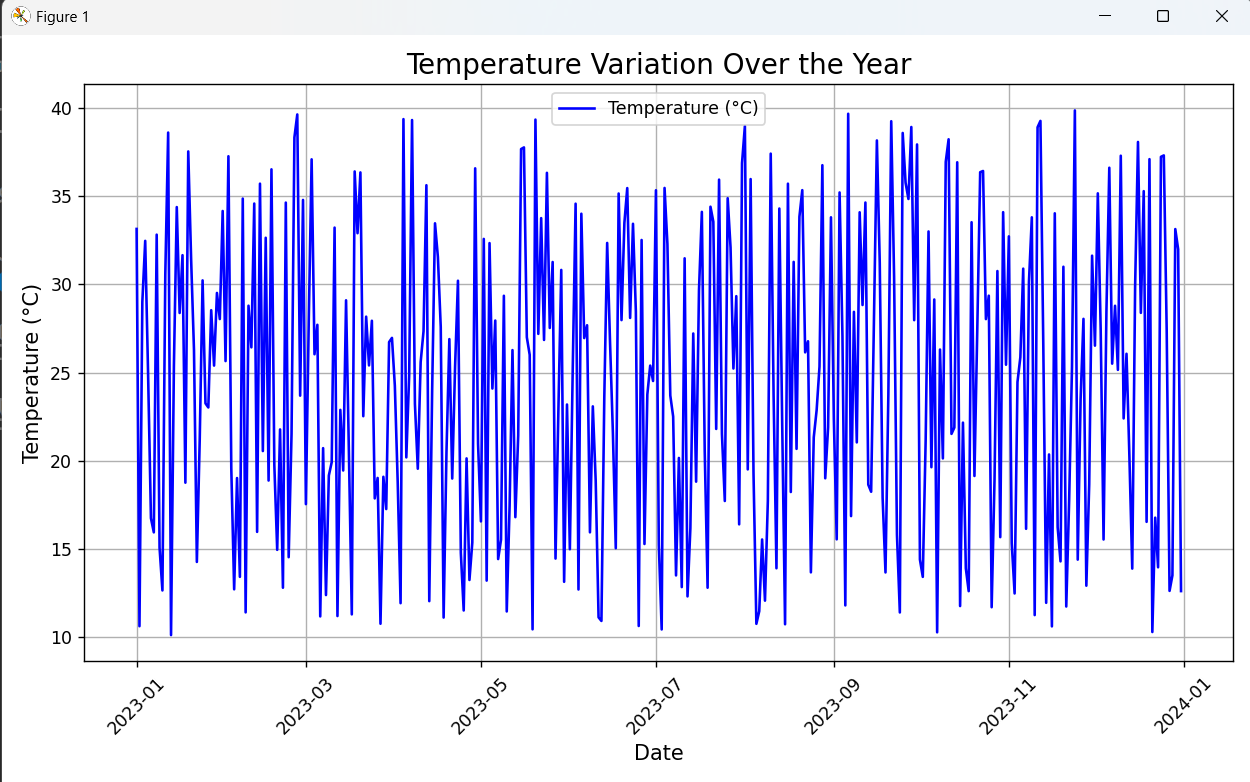
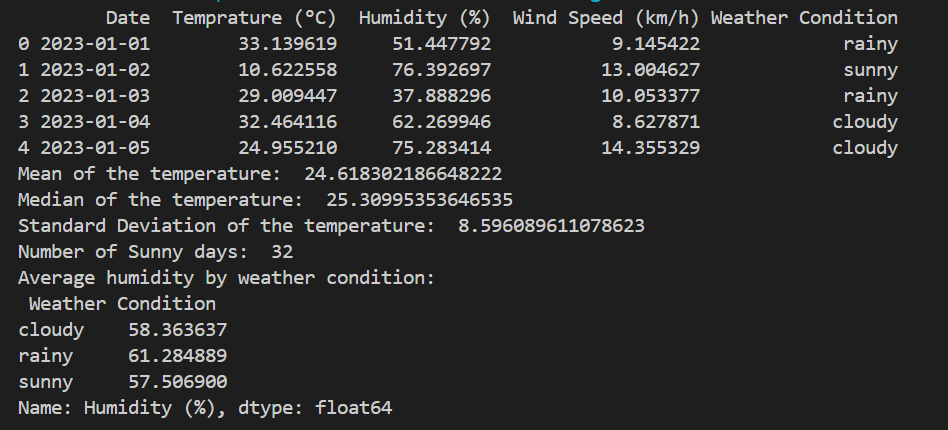
plt.grid(axis='y')

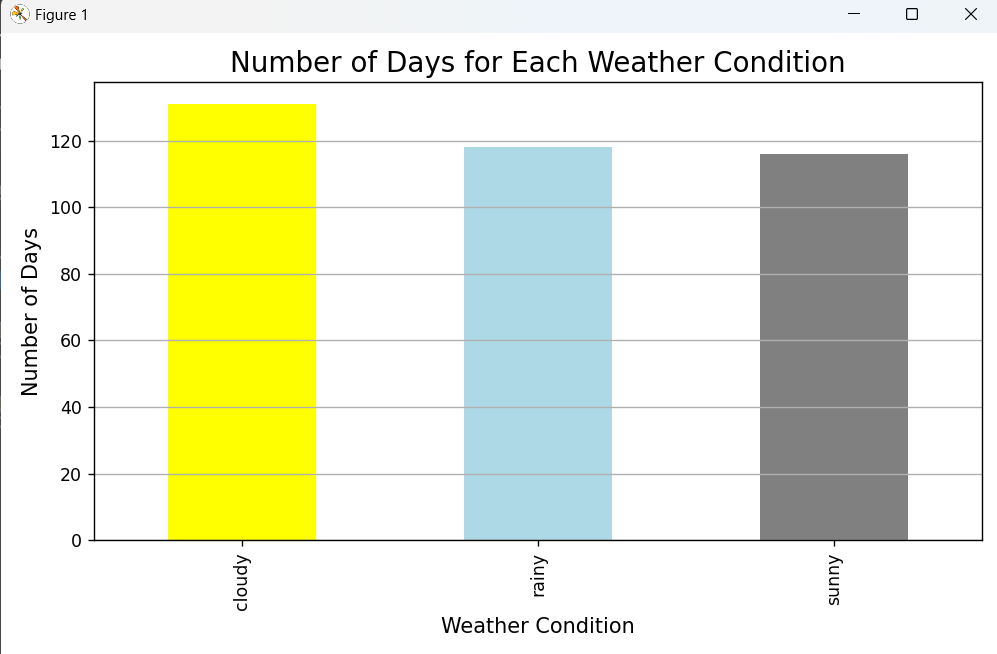
# Show the bar plot

plt.tight\_layout()

plt.show()

**Output:**

****



# Case Study 2: Sales Data Analysis

**Code:**

import pandas as pd

import numpy as np

from datetime import datetime, timedelta

import matplotlib.pyplot as plt

#generating number of rows

num\_of\_rows = 500

#generating same random numbers

np.random.seed(10)

#generating random data

product = np.random.choice(["Apple", "Mango", "Peach", "Banana", "Pear", "Gava", "Melon", "Plum", "Apricot", "Water Melon"], size=num\_of\_rows)

price = np.random.randint(10,1000, size=num\_of\_rows)

quantity = np.random.randint(1,20, size = num\_of\_rows)

start\_date = datetime.now() - timedelta(days=365)

date = [start\_date + timedelta(days=np.random.randint(0, 365)) for \_ in range(num\_of\_rows)]

#generating dataset

sales\_data = pd.DataFrame({

    'Order ID': [f'ORD{i}' for i in range(1, num\_of\_rows+1)],

    'Product': product,

    'Price ($)': price,

    'Quantity': quantity,

    'Date': date

})

#displays the dataset

print(sales\_data.head(10))

#converting price and quantity to arrays

price\_quantity\_array = sales\_data[['Price ($)', 'Quantity']].to\_numpy()

#finding the total sales

total\_sales = price\_quantity\_array[:,0] \* price\_quantity\_array[:,1]

#displays the total sales

print(total\_sales[:10])

#Calculating total sales and adding a new column

sales\_data['Total Sales'] = sales\_data['Price ($)'] \* sales\_data['Quantity']

#Filtering the data

filter\_data = sales\_data[sales\_data['Total Sales'] > 100]

#displays the filtered data

print("\nSales greater than 100$: \n", filter\_data[:5])

#grouping the dataset by product

total\_quantity = sales\_data.groupby('Product')['Quantity'].sum().reset\_index()

#displays the total quantity sold for each product

print(total\_quantity[:5])

# Create the scatter plot

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

plt.scatter(sales\_data['Price ($)'], sales\_data['Quantity'], alpha=0.5, color='b')

plt.title('Scatter Plot of Price vs Quantity Sold')

plt.xlabel('Price (in $)')

plt.ylabel('Quantity Sold')

plt.grid(True)

plt.show()

# Create the histogram

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

plt.hist(sales\_data['Total Sales'], bins=20, color='green', edgecolor='black', alpha=0.7)

plt.title('Distribution of Total Sales Values')

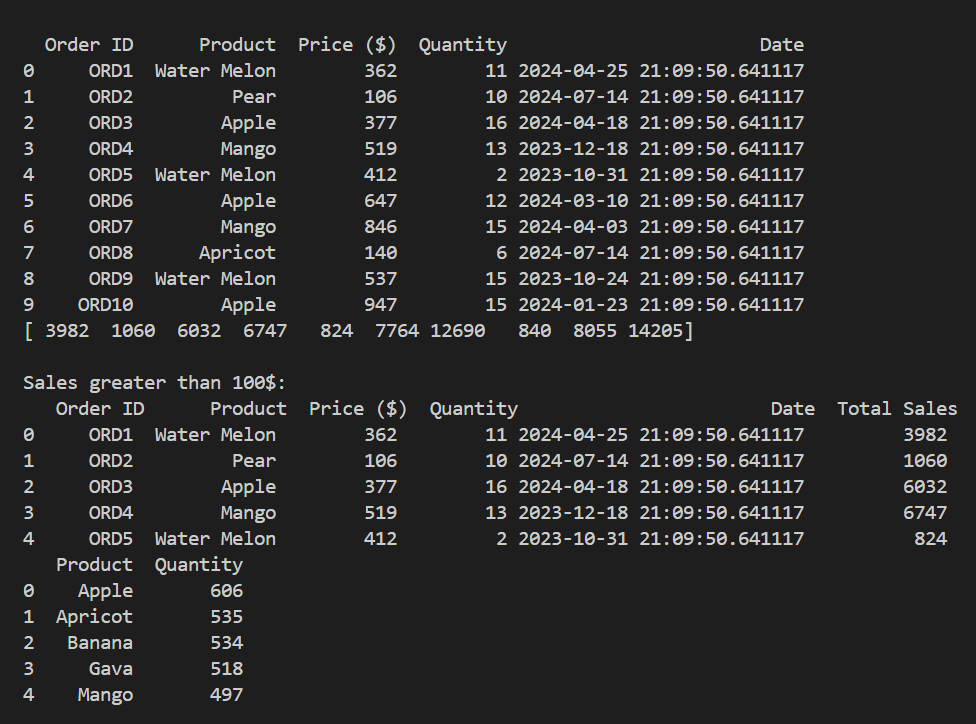
plt.xlabel('Total Sales (in $)')

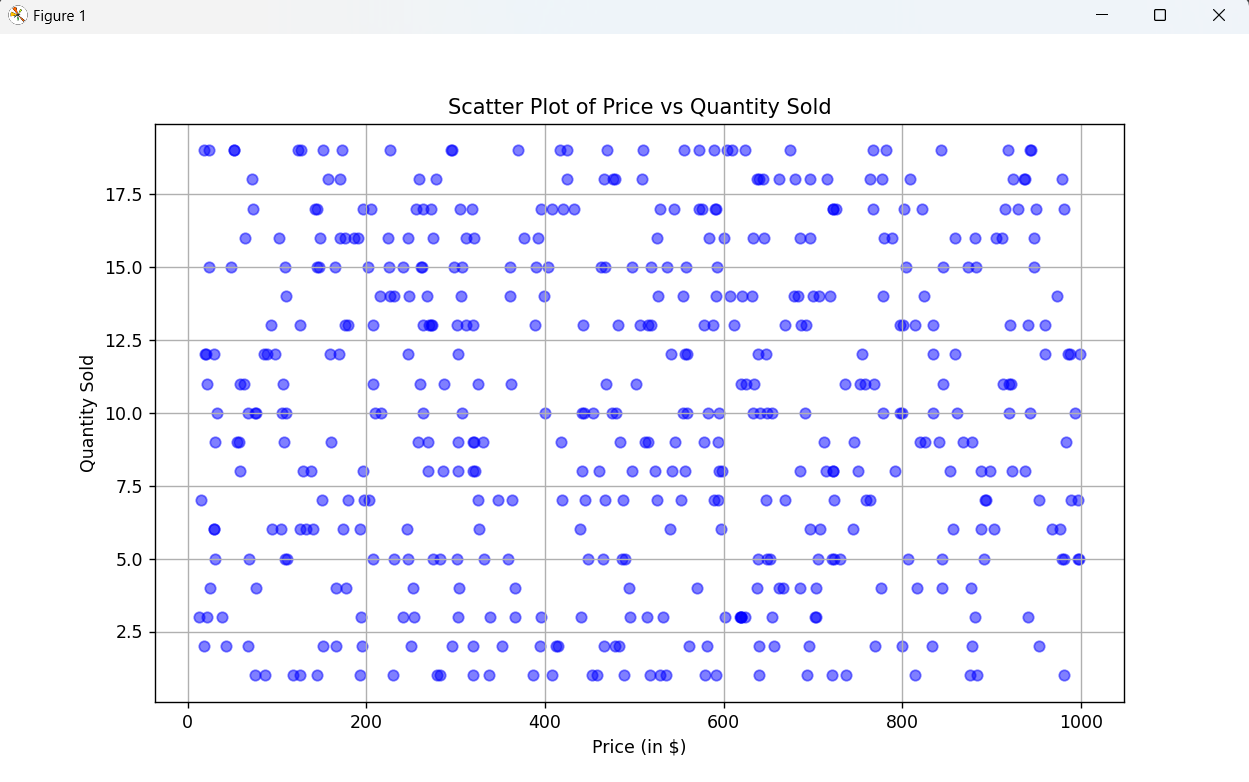
plt.ylabel('Frequency')

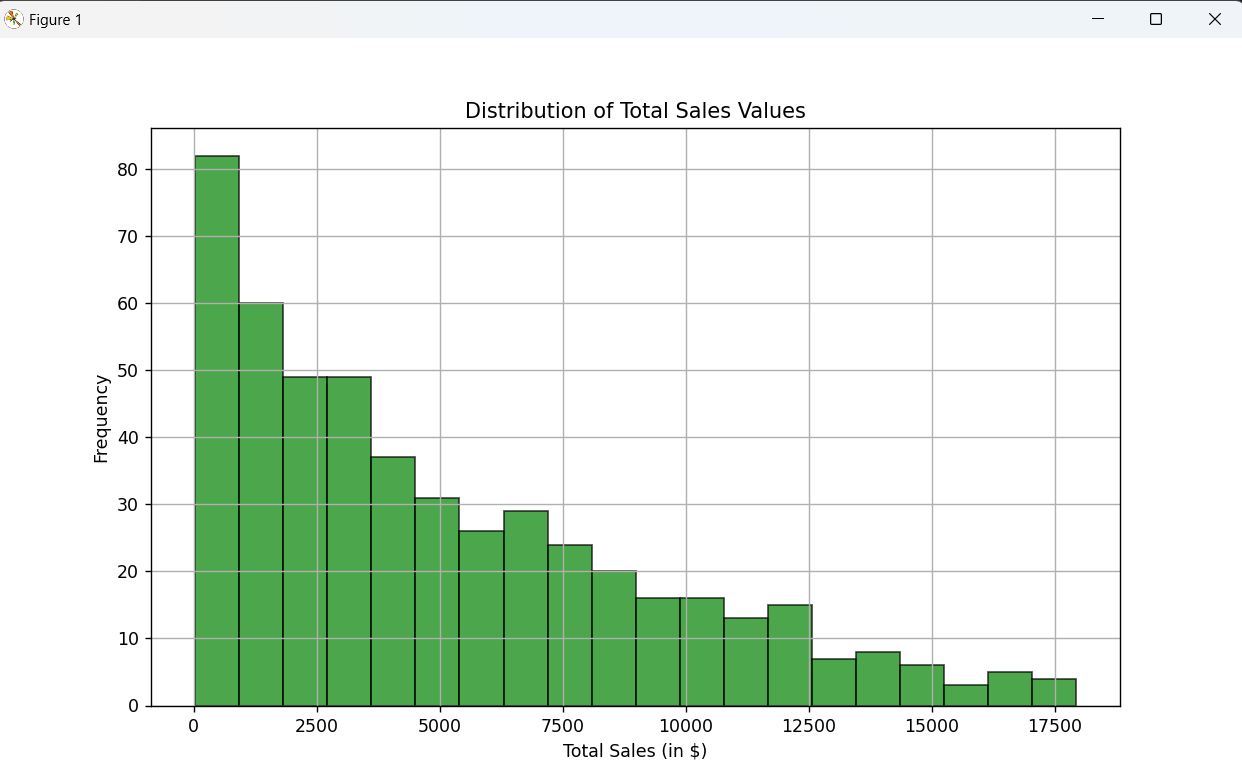
plt.grid(True)

plt.show()

**Output:**

****

****

****

# Case Study 3: Employee Salary Analysis

**Code:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

#generating number of rows

num\_of\_rows = 300

#generating repeated data

np.random.seed(10)

#generating data for dataset

names = np.random.choice(["Aleena","Amina","Aqsa","Iqra","Sumbal","Merub","Mirha","Tehreem","Zaima","Hoorain",

                          "Mursal","Mushaf","Wali","Abdullah","Ghani","Abu trab","Irfana","Khushi","Ayesha","Arbaz"], size=num\_of\_rows)

department = np.random.choice(["Mathematics","IT","CS","Sociology","SE"], size=num\_of\_rows)

salary = np.random.randint(30000,120000, size=num\_of\_rows)

years\_of\_exp = np.random.randint(1,25, size=num\_of\_rows)

#generating dataset

Employee\_data = pd.DataFrame({

    'Employee ID': [f'{i}' for i in range(1,num\_of\_rows+1)],

    'Names': names,

    'Department': department,

    'Salary': salary,

    'Years of Experience': years\_of\_exp

})

#displays the dataset

print(Employee\_data.head(10))

#converting salary column to array

salary\_array = Employee\_data['Salary'].to\_numpy()

avg\_salary = int(np.mean(salary\_array))

mini\_salary = np.min(salary\_array)

max\_salary = np.max(salary\_array)

#displays the average, minimum and maximum salary

print("Average Salary: ", avg\_salary)

print("Minimum Salary: ", mini\_salary)

print("Maximum Salary: ", max\_salary)

#filtering the data on the basis of salary and years of experience

filtered\_data = Employee\_data[(Employee\_data['Years of Experience'] > 5) & (Employee\_data['Salary'] > avg\_salary)]

#displays the filtered data

print(filtered\_data[:10])

#grouping the data by department

mean\_salary\_department = Employee\_data.groupby('Department')['Salary'].mean()

#displays the mean salary for each department

print("Mean Salary for each Department: \n", mean\_salary\_department)

# Bar plot showing average salary by department

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

mean\_salary\_department.plot(kind='bar', color='skyblue', edgecolor='black')

plt.title('Average Salary by Department', fontsize=16)

plt.xlabel('Department', fontsize=12)

plt.ylabel('Average Salary', fontsize=12)

plt.xticks(rotation=45)

plt.grid(axis='y')

plt.tight\_layout()

plt.show()

# Line plot for salary distribution by years of experience

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

Employee\_data\_sorted = Employee\_data.sort\_values(by='Years of Experience')

plt.plot(Employee\_data\_sorted['Years of Experience'], Employee\_data\_sorted['Salary'], marker='o', color='green', linestyle='-', markersize=5)

plt.title('Salary Distribution by Years of Experience', fontsize=16)

plt.xlabel('Years of Experience', fontsize=12)

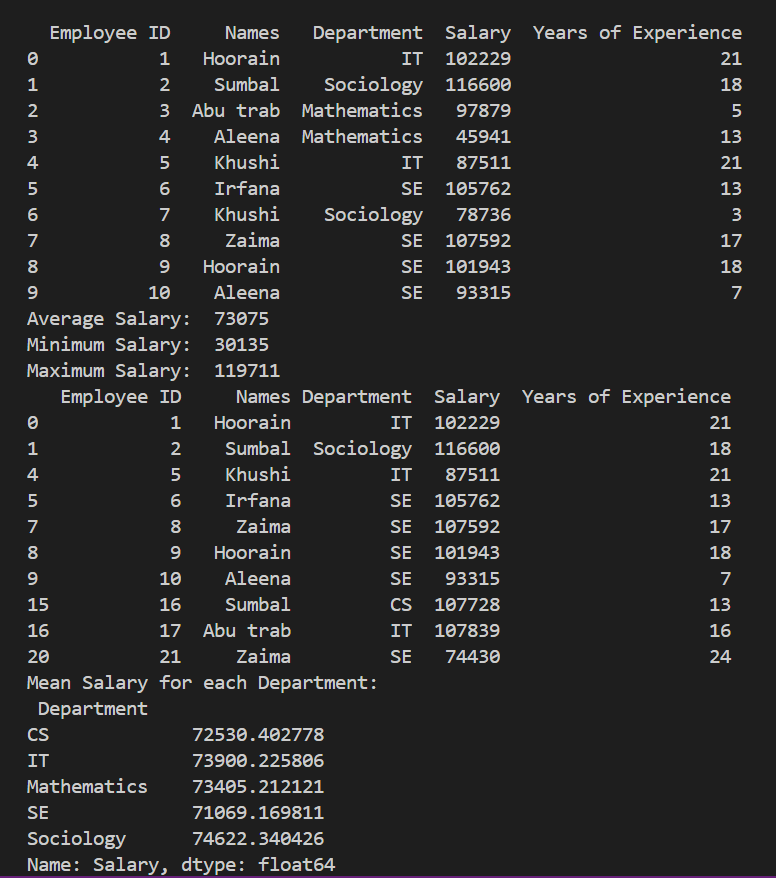
plt.ylabel('Salary', fontsize=12)

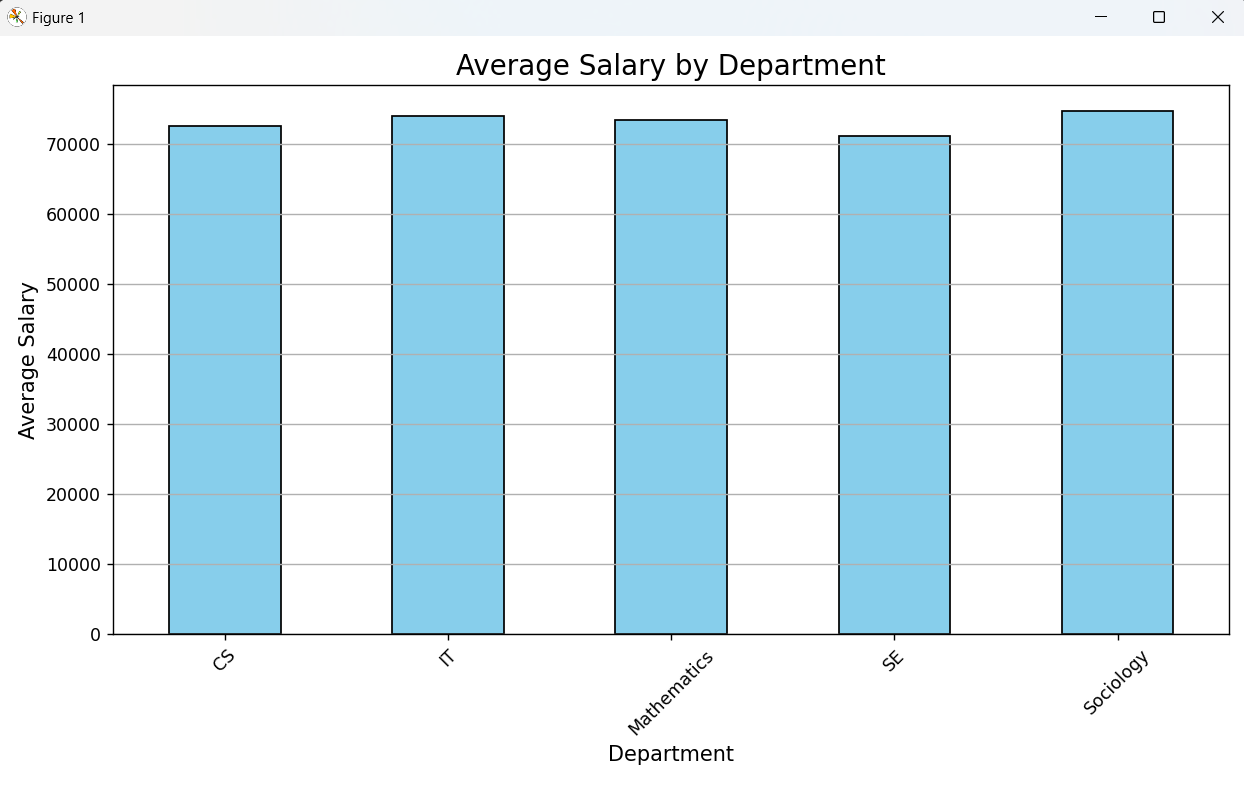
plt.grid(True)

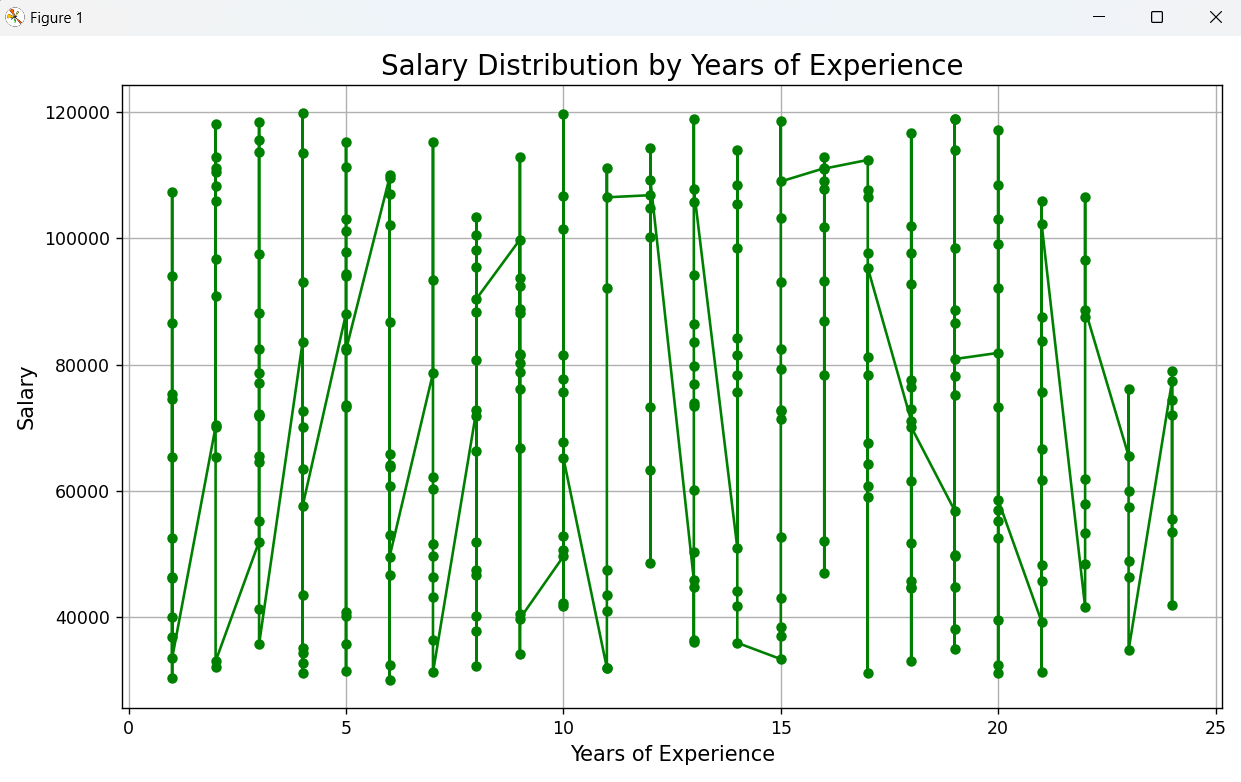
plt.tight\_layout()

plt.show()

**Output:**

****

****

****

# Case Study 4: Exam Score Analysis

**Code:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

# Defining the number of rows

num\_of\_rows = 200

# Generating same random data

np.random.seed(10)

# Generating random data

subject = np.random.choice(["Maths", "Computer", "English", "Urdu", "Science"], num\_of\_rows)

score = np.random.randint(0, 100, num\_of\_rows)

total\_marks = 100

# Generating dataset

exam\_data = pd.DataFrame({

    'Student ID': [f'ID{i}' for i in range(1, num\_of\_rows + 1)],

    'Name': [f'Name{i}' for i in range(1, num\_of\_rows + 1)],

    'Subject': subject,

    'Score': score,

    'Total Marks': total\_marks

})

# Displays the exam data

print(exam\_data.head(10))

# Converting score column to array

score\_array = exam\_data['Score'].to\_numpy()

score\_mean = np.mean(score\_array)

score\_median = np.median(score\_array)

score\_std\_dev = np.std(score\_array)

# Displays the mean, median and standard deviation of the scores

print("Mean of the score: ", score\_mean)

print("Median of the score: ", score\_median)

print("Standard Deviation of the score: ", score\_std\_dev)

# Filtering the data by exam score

filtered\_data = exam\_data[exam\_data['Score'] > 80]

total\_students = len(filtered\_data)

# Displays the number of students who achieved a score above 80%

print("Number of students who achieved score above 80%: ", total\_students)

# Grouping the data by subjects

avg\_score = exam\_data.groupby('Subject')['Score'].mean()

# Displays the average score of the subjects

print("Average score of the subjects: \n", avg\_score)

# Plotting a histogram to show the distribution of scores

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

plt.hist(exam\_data['Score'], bins=10, color='blue', edgecolor='black', alpha=0.7)

plt.title('Distribution of Scores Across All Students', fontsize=16)

plt.xlabel('Scores', fontsize=12)

plt.ylabel('Number of Students', fontsize=12)

plt.grid(True)

plt.tight\_layout()

plt.show()

# Plotting a bar chart to compare the average scores across different subjects

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

avg\_score.plot(kind='bar', color='orange', edgecolor='black')

plt.title('Average Scores by Subject', fontsize=16)

plt.xlabel('Subject', fontsize=12)

plt.ylabel('Average Score', fontsize=12)

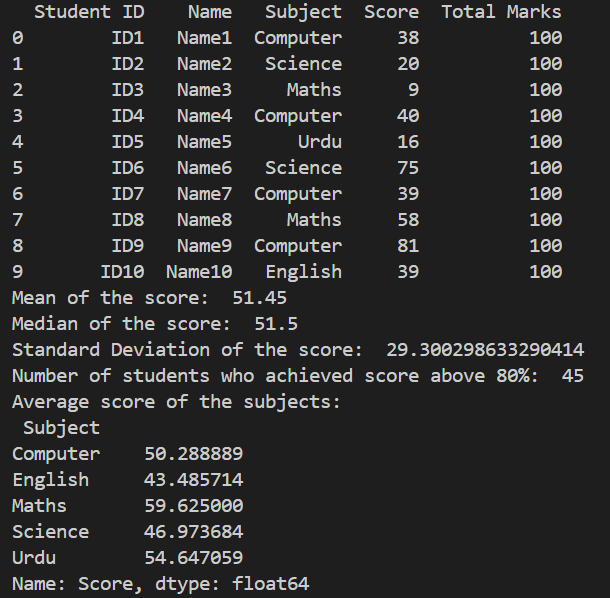
plt.xticks(rotation=45)

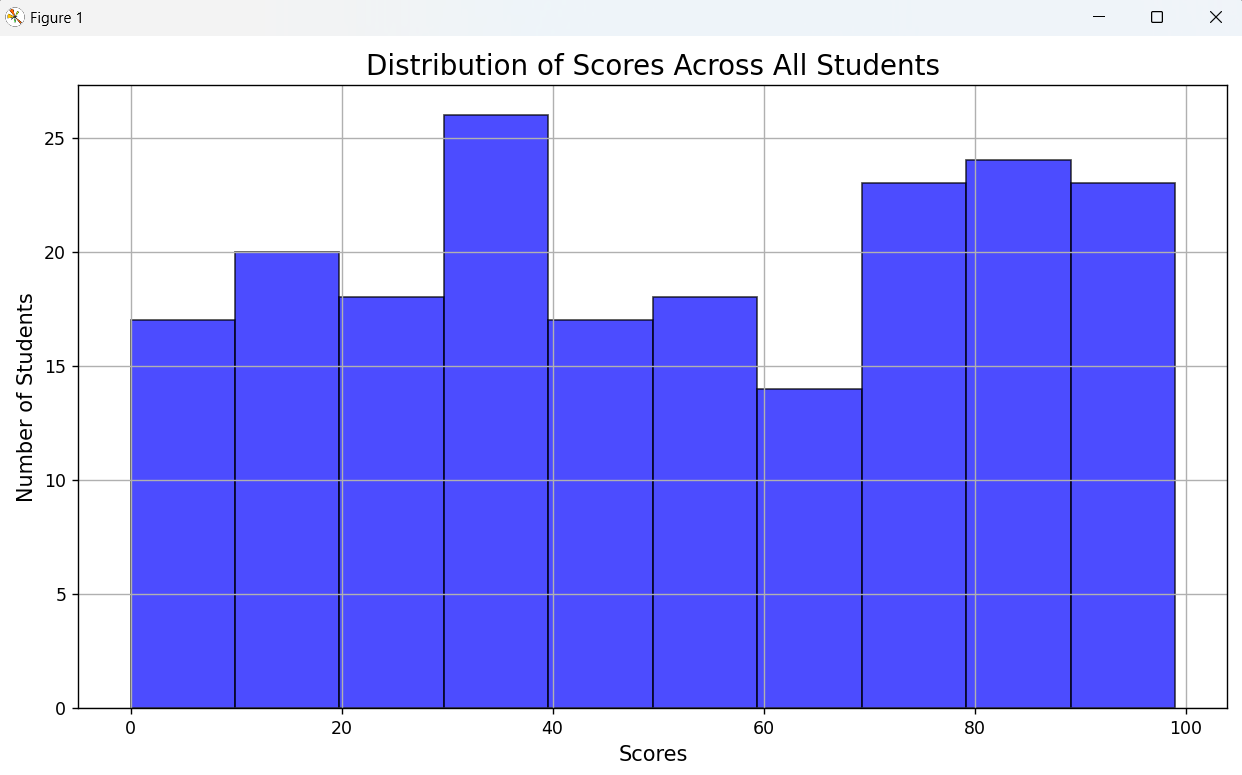
plt.grid(axis='y')

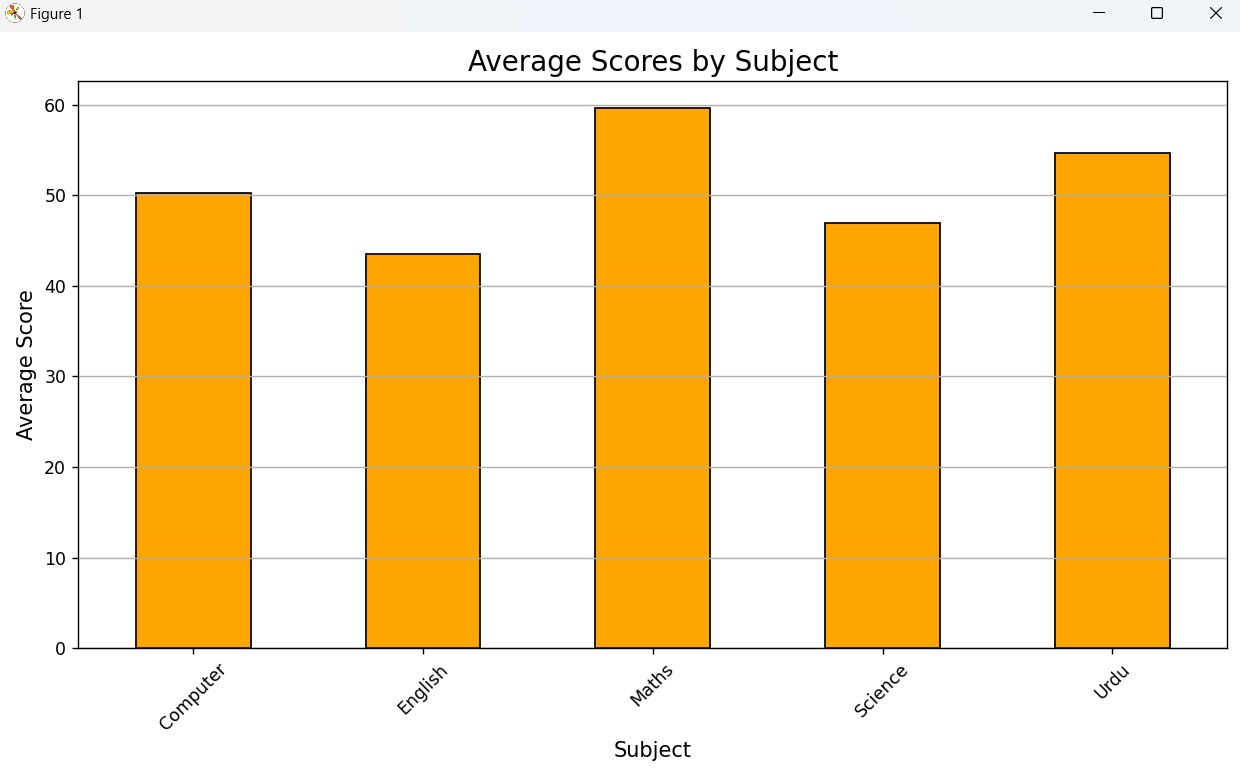
plt.tight\_layout()

plt.show()

**Output:**

****

****

****

# Case Study 5: Stock Market Analysis

**Code:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from datetime import datetime, timedelta

# Generating the number of rows

num\_of\_rows = 1000

# Generating same random data

np.random.seed(100)

# Generating data

start\_date = datetime.now() - timedelta(days=365\*2)

date = [start\_date + timedelta(days=np.random.randint(0, 365\*2)) for \_ in range(num\_of\_rows)]

company = np.random.choice(["Toyota", "Honda", "Suzuki", "Kia", "Nissan"], num\_of\_rows)

open\_price = np.random.randint(50, 500, num\_of\_rows)

close\_price = np.random.randint(50, 500, num\_of\_rows)

volume\_trade = np.random.randint(1000, 1000000, num\_of\_rows)

# Generating dataset

stock\_data = pd.DataFrame({

    'Date': date,

    'Company': company,

    'Open Price': open\_price,

    'Close Price': close\_price,

    'Volume Traded': volume\_trade

})

# Displays the dataset

print(stock\_data.head())

# Calculating percentage change in Close Price

close\_array = stock\_data['Close Price'].to\_numpy()

percentage\_changes = np.diff(close\_price) / close\_price[:-1] \* 100

# Adding percentage change to stock\_data

stock\_data['Percentage'] = np.append([0], percentage\_changes)

stock\_data.loc[stock\_data['Percentage'] < 0, 'Percentage'] = 0

print(stock\_data.head())

# Filtering the data based on percentage change

prev\_percentage = stock\_data['Percentage'].shift(1)

filter\_data = stock\_data[stock\_data['Percentage'] > (prev\_percentage + 2)]

print(filter\_data.head())

# Grouping the data to get total volume traded by company

total\_volume\_traded = stock\_data.groupby('Company')['Volume Traded'].sum().reset\_index()

print(total\_volume\_traded)

# Plotting the Close Price trend over time for a particular company (e.g., "Toyota")

company\_data = stock\_data[stock\_data['Company'] == 'Toyota']

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

plt.plot(company\_data['Date'], company\_data['Close Price'], color='blue', label='Close Price')

plt.title('Close Price Trend Over Time for Toyota', fontsize=16)

plt.xlabel('Date', fontsize=12)

plt.ylabel('Close Price ($)', fontsize=12)

plt.xticks(rotation=45)

plt.grid(True)

plt.legend()

plt.tight\_layout()

plt.show()

# Calculating average percentage change by company

avg\_percentage\_change = stock\_data.groupby('Company')['Percentage'].mean().reset\_index()

# Plotting a bar chart to compare average percentage change in Close Price for different companies

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

plt.bar(avg\_percentage\_change['Company'], avg\_percentage\_change['Percentage'], color=['orange', 'blue', 'green', 'red', 'purple'])

plt.title('Average Percentage Change in Close Price by Company', fontsize=16)

plt.xlabel('Company', fontsize=12)

plt.ylabel('Average Percentage Change (%)', fontsize=12)

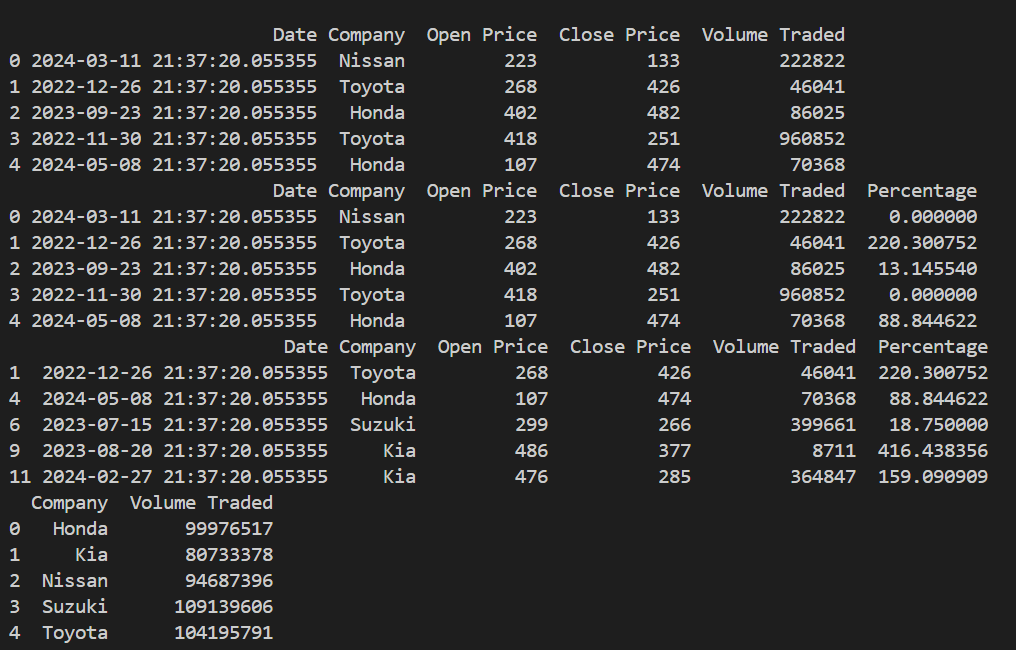
plt.xticks(rotation=45)

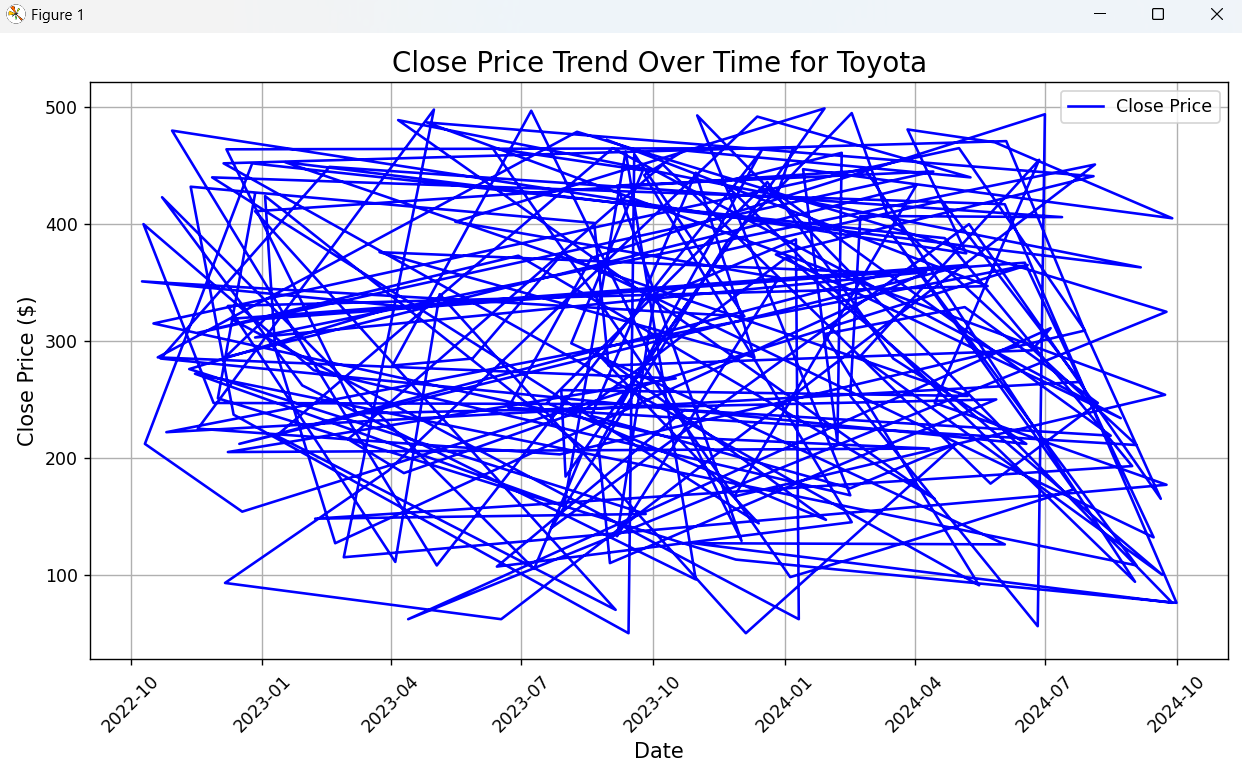
plt.grid(axis='y')

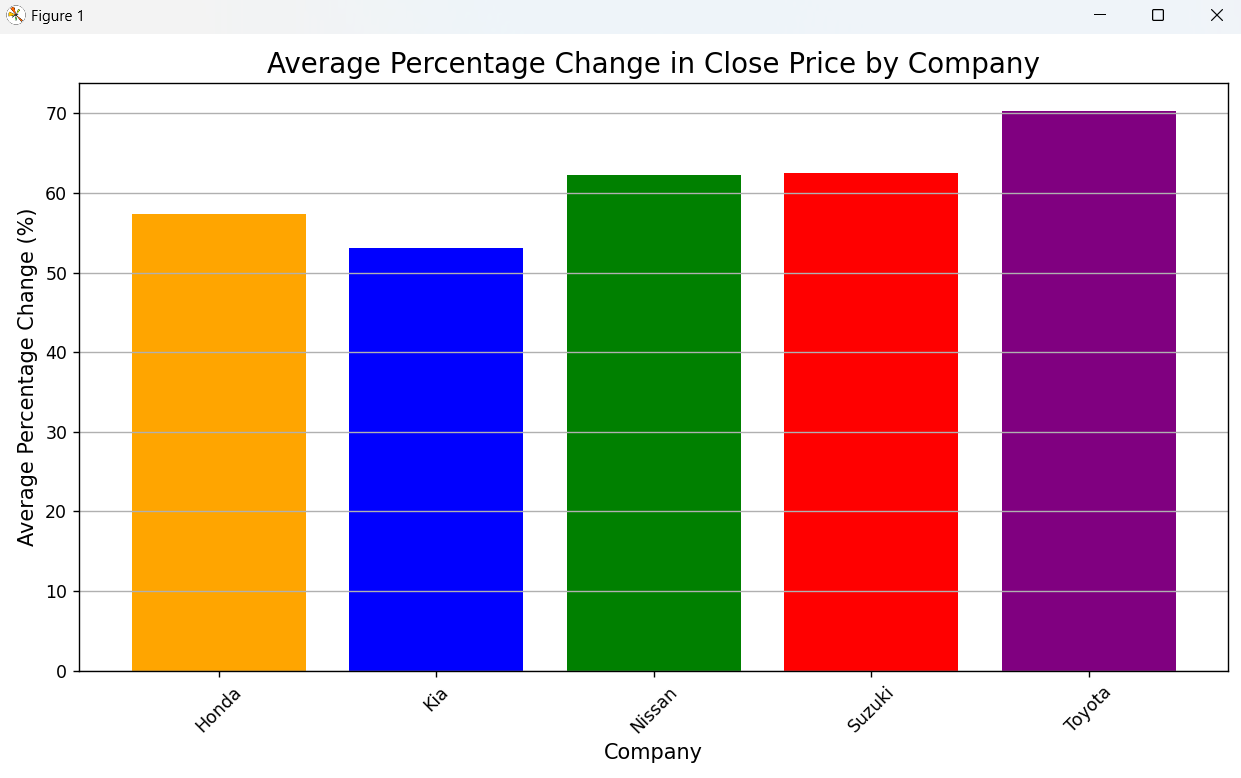
plt.tight\_layout()

plt.show()

**Output:**

****

****

****