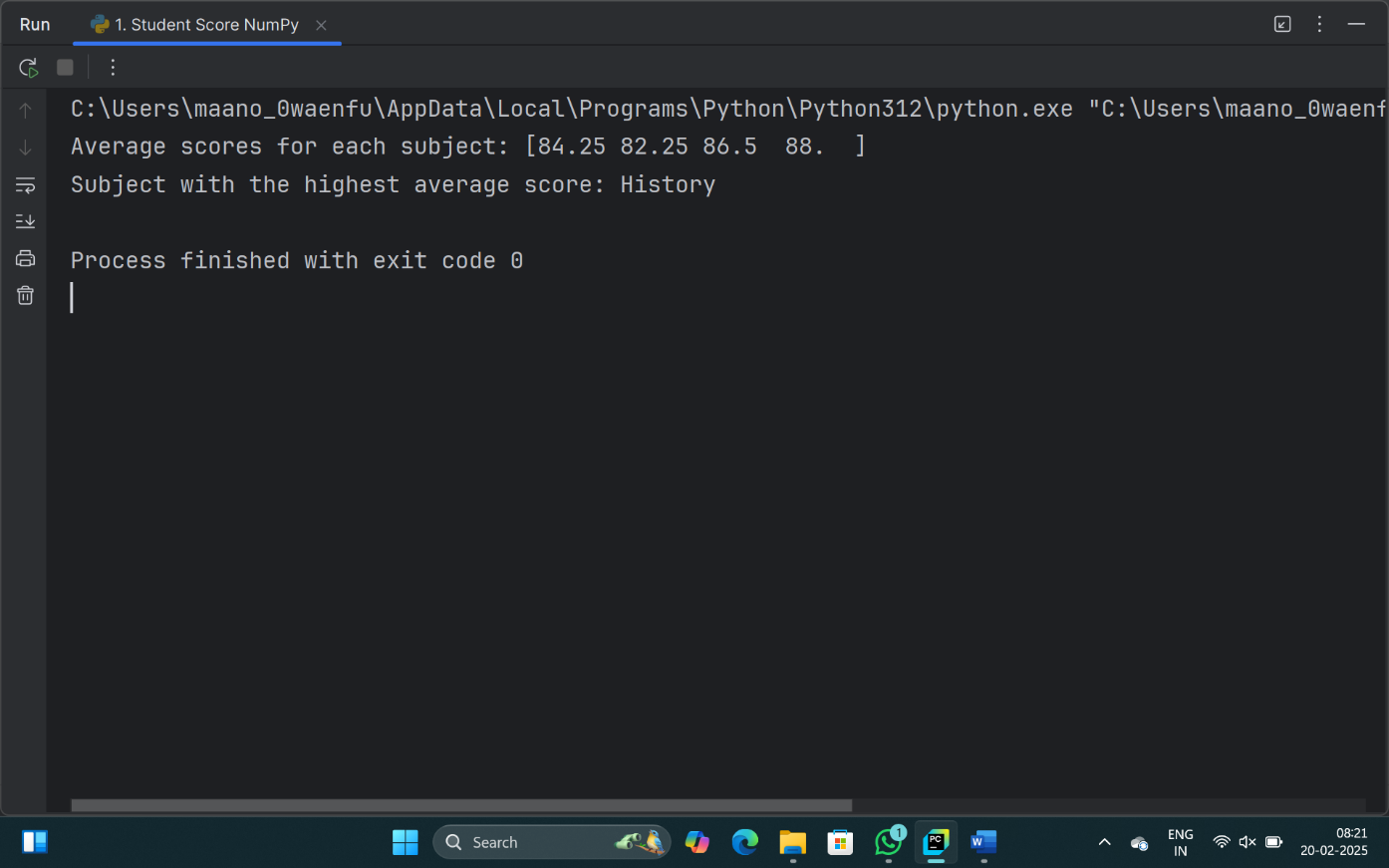
1. **Question:** How would you use NumPy arrays to calculate the average score for each subject and determine the subject with the highest average score? Assume 4x4 matrix that stores marks of each student in given order.

**Code:**

import numpy as np  
  
student\_scores = np.array([  
 [75, 85, 92, 88],  
 [89, 76, 81, 90],  
 [95, 88, 82, 85],  
 [78, 80, 91, 89]  
])  
  
average\_scores = np.mean(student\_scores, axis=0)  
  
subjects = ['Math', 'Science', 'English', 'History']  
highest\_avg\_score\_subject = subjects[np.argmax(average\_scores)]  
  
print("Average scores for each subject:", average\_scores)  
print("Subject with the highest average score:", highest\_avg\_score\_subject)

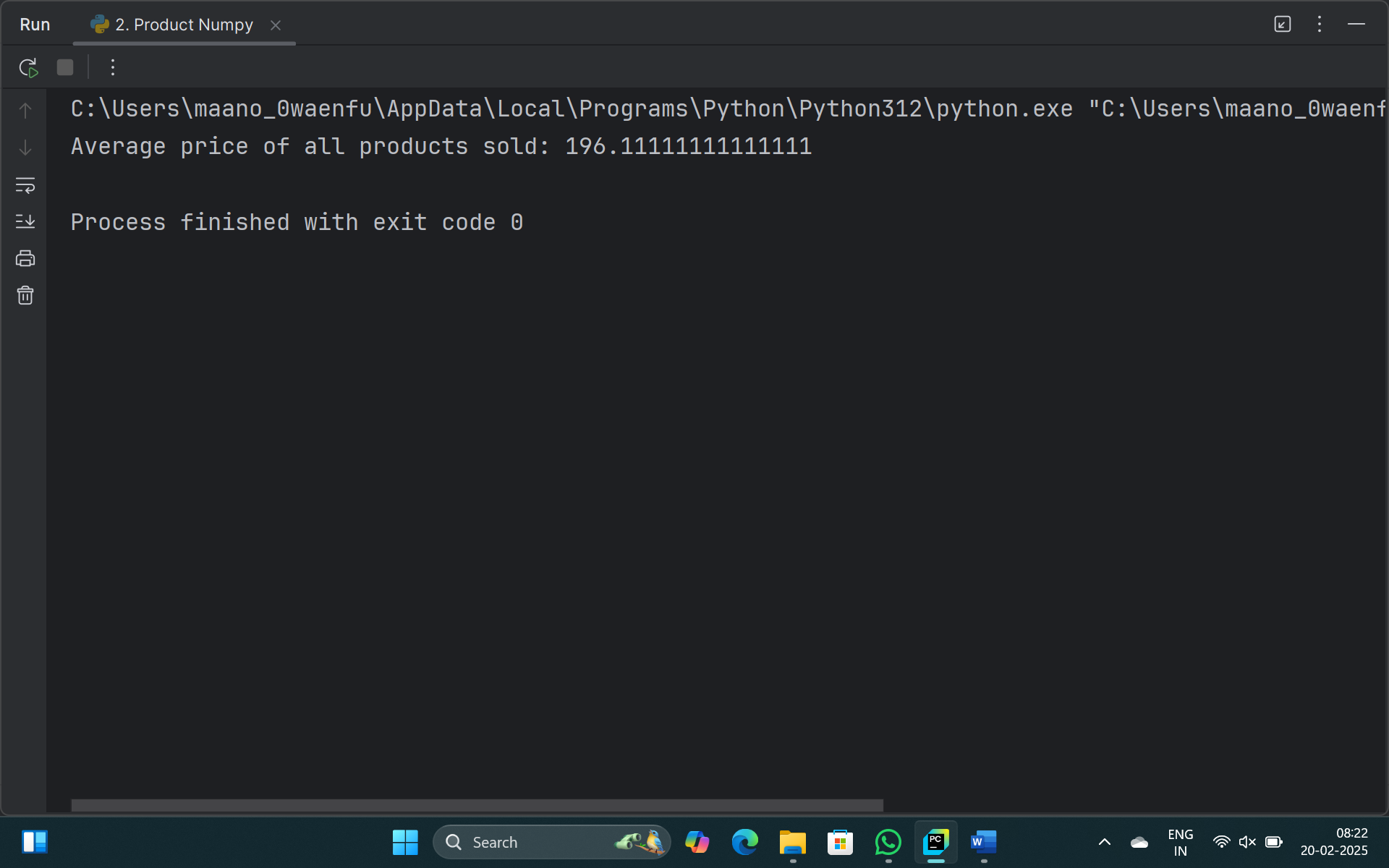
**Output:**

1. **Question:** How would you find the average price of all the products sold in the past month? Assume 3x3 matrix with each row representing the sales for a different product

**Code:**

import numpy as np  
  
sales\_data = np.array([  
 [150, 200, 250],  
 [100, 180, 230],  
 [175, 220, 260]  
])  
  
average\_price = np.mean(sales\_data)  
  
print("Average price of all products sold:", average\_price)

**Output:**

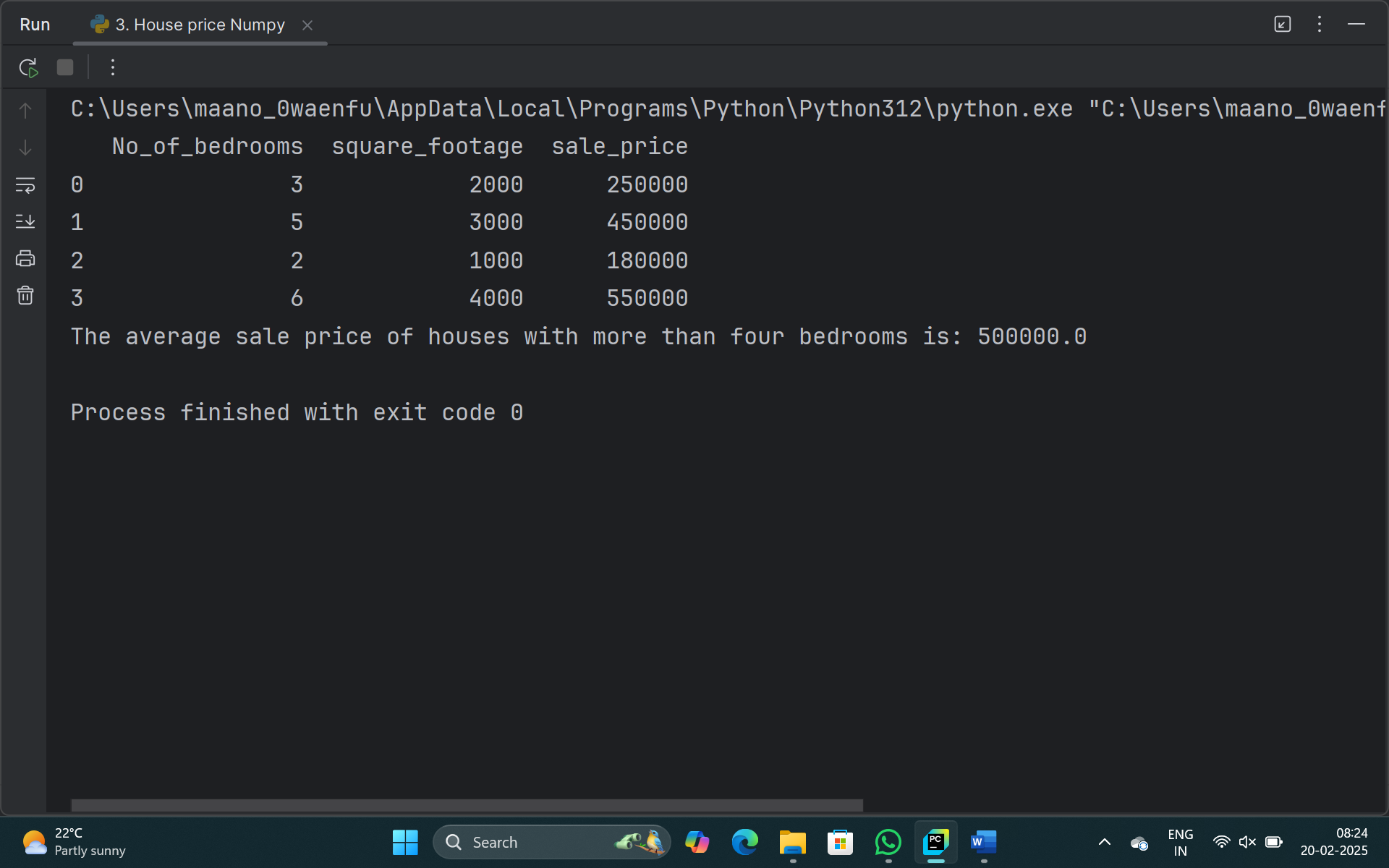


1. **Question:** Using NumPy arrays and operations, how would you find the average sale price of houses with more than four bedrooms in the neighborhood?

**Code:**

import numpy as np  
import pandas as pd  
  
df = pd.read\_csv("house\_data.csv")  
house\_data = df.to\_numpy()  
print(df)  
houses\_with\_more\_than\_four\_bedrooms = house\_data[house\_data[:, 0] > 4]  
  
sale\_prices = houses\_with\_more\_than\_four\_bedrooms[:, -1]  
  
average\_sale\_price = sale\_prices.mean()  
  
print(f"The average sale price of houses with more than four bedrooms is: {average\_sale\_price}")

**Output:**

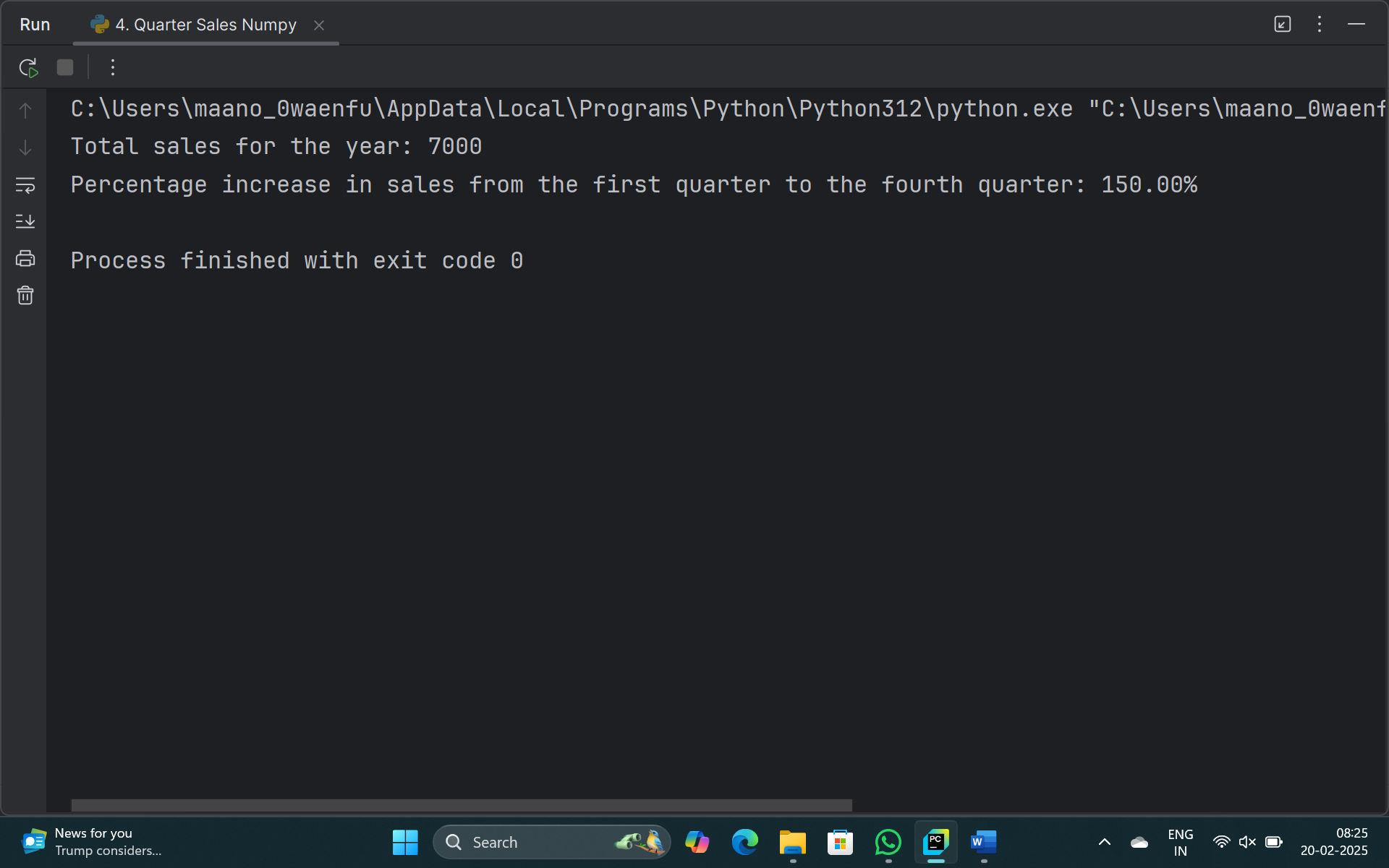


1. **Question:** Using NumPy arrays and arithmetic operations calculate the total sales for the year and determine the percentage increase in sales from the first quarter to the fourth quarter?

**Code:**

import numpy as np  
  
sales\_data = np.array([1000, 1500, 2000, 2500])  
  
total\_sales = sales\_data.sum()  
  
percentage\_increase = ((sales\_data[-1] - sales\_data[0]) / sales\_data[0]) \* 100  
  
print(f"Total sales for the year: {total\_sales}")  
print(f"Percentage increase in sales from the first quarter to the fourth quarter: {percentage\_increase:.2f}%")

**Output:**



1. **Question:** How would you use NumPy arrays and arithmetic operations to calculate the average fuel efficiency and determine the percentage improvement in fuel efficiency between two car models?

**Code:**

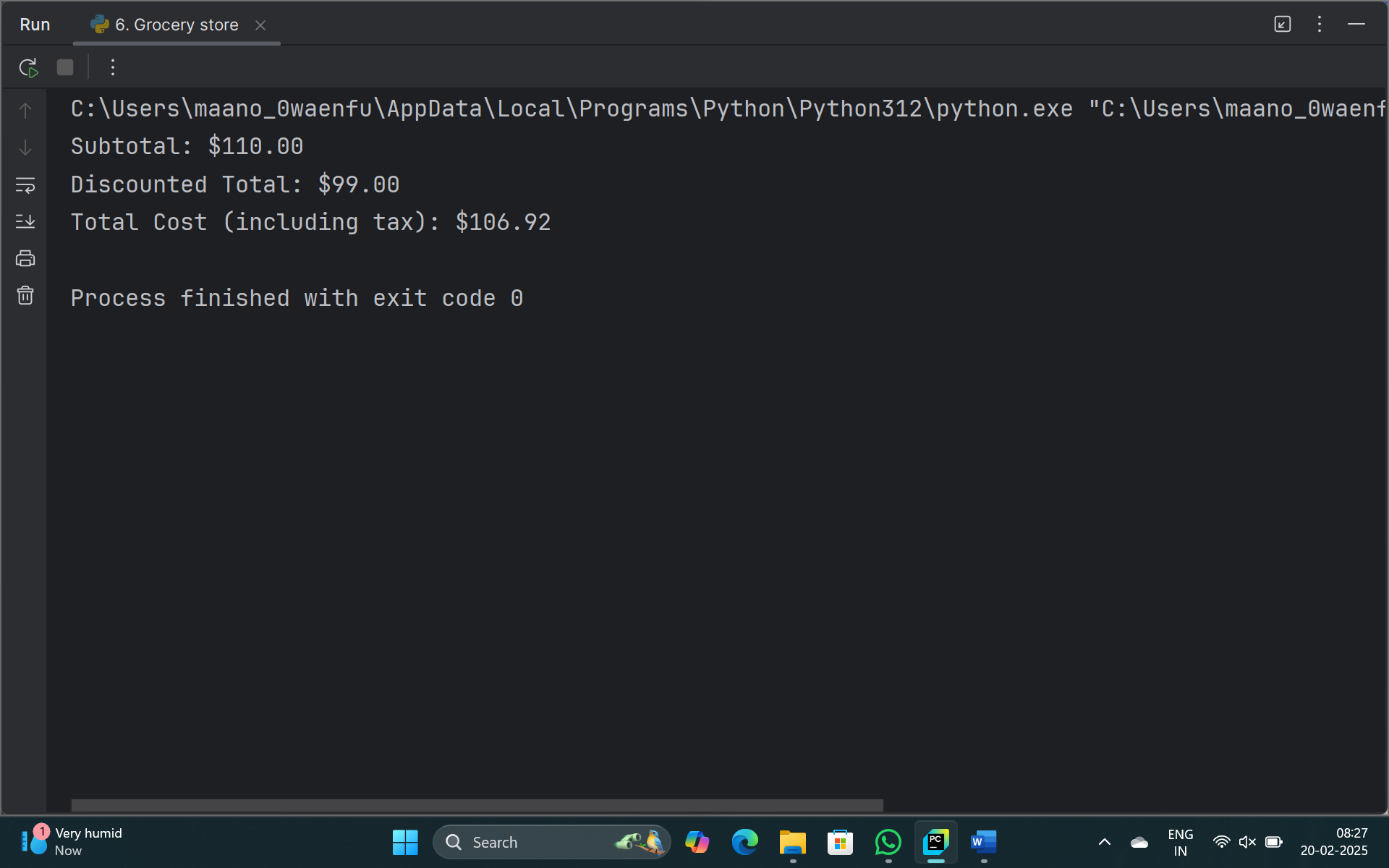
import numpy as np  
  
fuel\_efficiency = np.array([20, 25, 30, 35, 40])  
  
average\_fuel\_efficiency = np.mean(fuel\_efficiency)  
print(f"Average Fuel Efficiency: {average\_fuel\_efficiency} miles per gallon")  
  
old\_efficiency = fuel\_efficiency[0]  
new\_efficiency = fuel\_efficiency[1]  
  
percentage\_improvement = ((new\_efficiency - old\_efficiency) / old\_efficiency) \* 100  
print(f"Percentage Improvement: {percentage\_improvement}%")

**Output:** 

1. **Question:** Use arithmetic operations to calculate the total cost of a customer's purchase, including discounts and taxes, given the item prices, quantities, discount rate, and tax rate?

**Code:**

item\_prices = [10, 20, 30]  
item\_quantities = [2, 3, 1]  
  
discount\_rate = 10  
tax\_rate = 8  
  
subtotal = sum([price \* quantity for price, quantity in zip(item\_prices, item\_quantities)])  
  
discounted\_total = subtotal \* ((100 - discount\_rate) / 100)  
  
total\_cost = discounted\_total \* ((100 + tax\_rate) / 100)  
  
print(f"Subtotal: ${subtotal:.2f}")  
print(f"Discounted Total: ${discounted\_total:.2f}")  
print(f"Total Cost (including tax): ${total\_cost:.2f}")

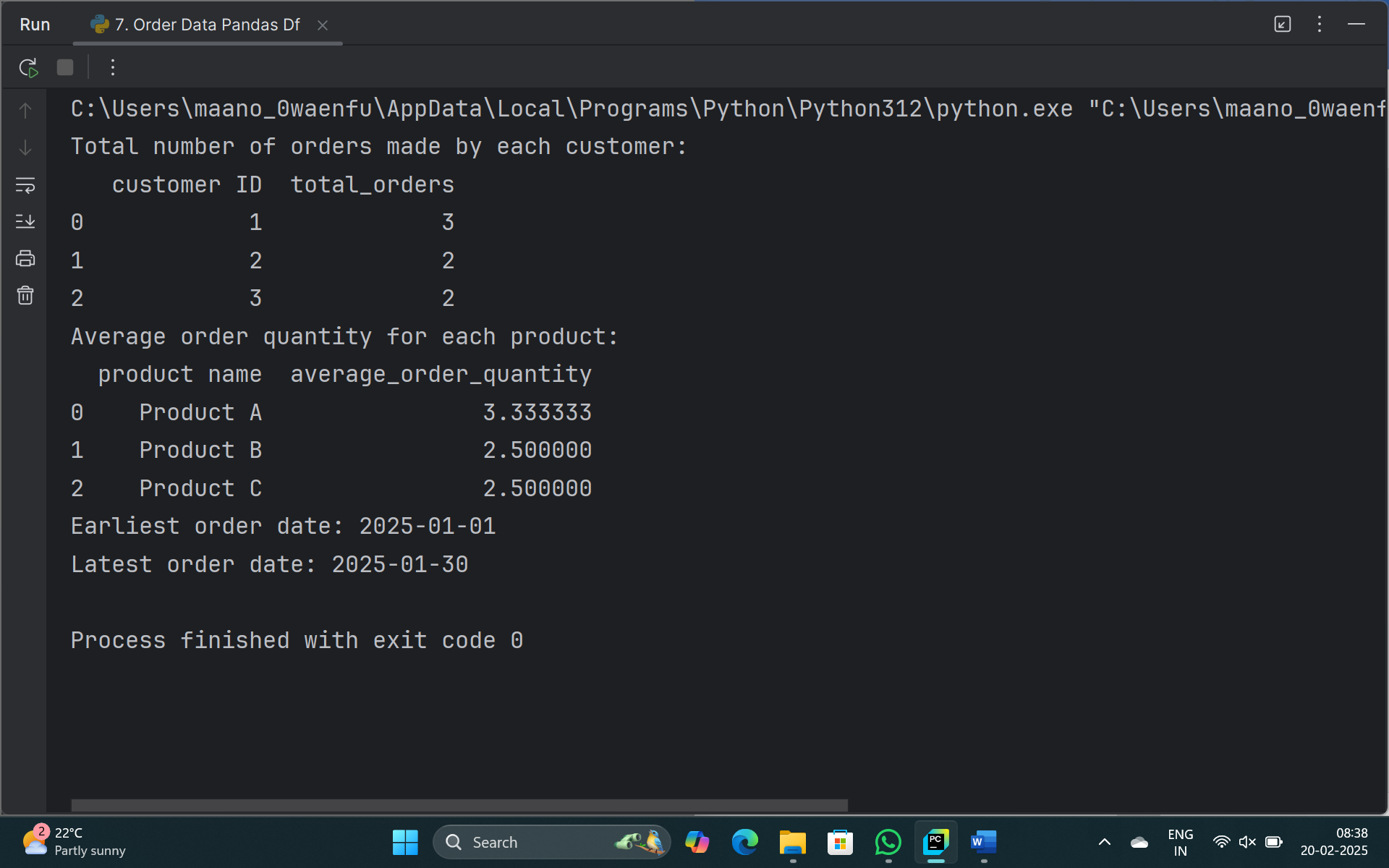
**Output:** 

1. **Question:** Using Pandas DataFrame operations, how would you find the following information from the order\_data DataFrame:
2. The total number of orders made by each customer.
3. The average order quantity for each product.
4. The earliest and latest order dates in the dataset.

**Code:**

import pandas as pd  
  
order\_data = pd.DataFrame({  
 'customer ID': [1, 2, 1, 3, 2, 3, 1],  
 'order date': ['2025-01-01', '2025-01-05', '2025-01-10', '2025-01-15', '2025-01-20', '2025-01-25', '2025-01-30'],  
 'product name': ['Product A', 'Product B', 'Product C', 'Product A', 'Product B', 'Product C', 'Product A'],  
 'order quantity': [2, 3, 1, 5, 2, 4, 3]  
})  
  
total\_orders\_per\_customer = order\_data.groupby('customer ID').size().reset\_index(name='total\_orders')  
print("Total number of orders made by each customer:")  
print(total\_orders\_per\_customer)  
  
average\_order\_quantity\_per\_product = order\_data.groupby('product name')['order quantity'].mean().reset\_index(name='average\_order\_quantity')  
print("Average order quantity for each product:")  
print(average\_order\_quantity\_per\_product)  
  
earliest\_order\_date = order\_data['order date'].min()  
latest\_order\_date = order\_data['order date'].max()  
print(f"Earliest order date: {earliest\_order\_date}")  
print(f"Latest order date: {latest\_order\_date}")

**Output:**

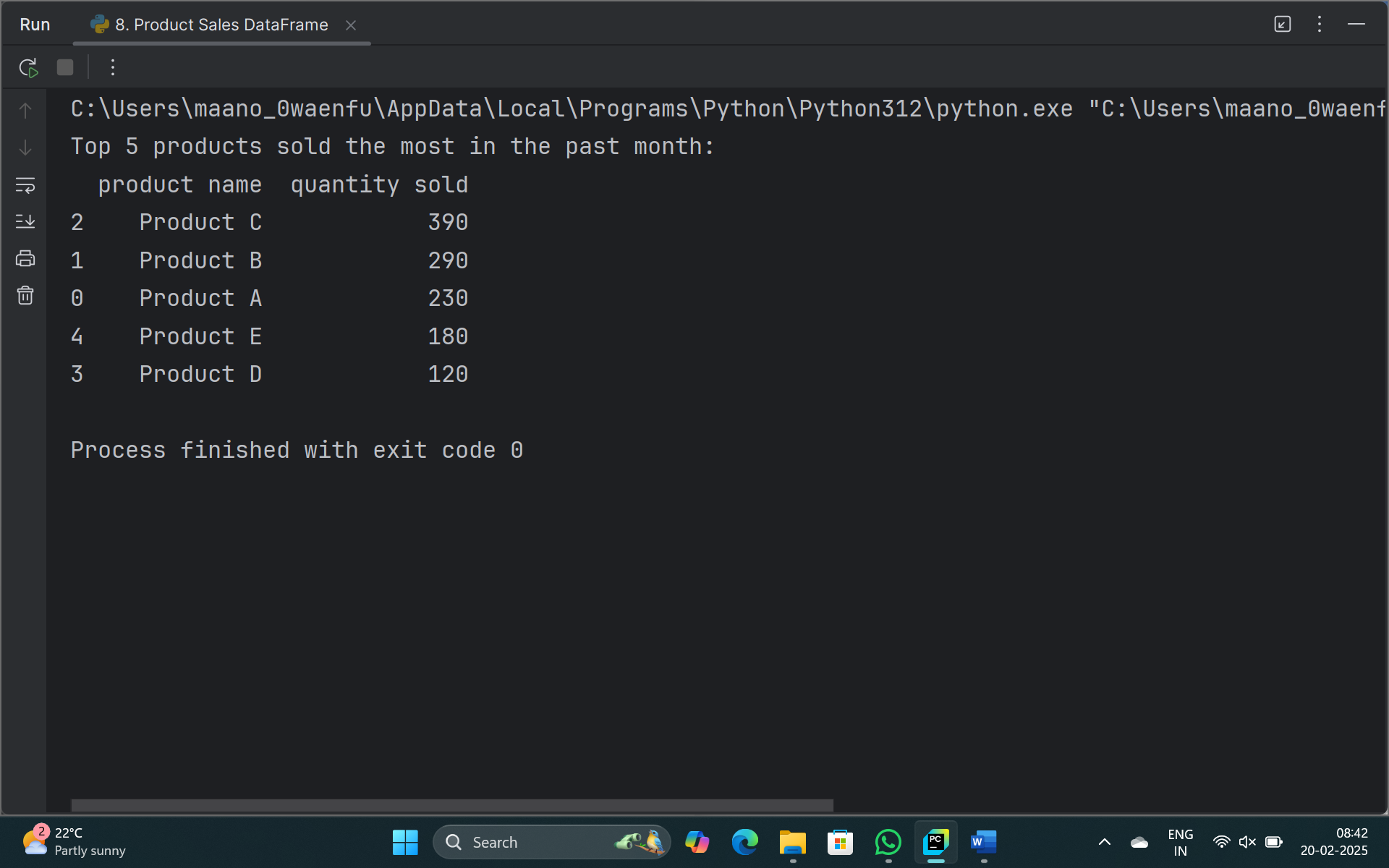


1. **Question:** How would you find the top 5 products that have been sold the most in the past month?

**Code:**

import pandas as pd  
  
data = {  
 'product name': ['Product A', 'Product B', 'Product C', 'Product D', 'Product E', 'Product A', 'Product B', 'Product F', 'Product G', 'Product C'],  
 'quantity sold': [100, 150, 200, 120, 180, 130, 140, 100, 110, 190]  
}  
  
sales\_data = pd.DataFrame(data)  
  
total\_quantity\_sold = sales\_data.groupby('product name')['quantity sold'].sum().reset\_index()  
  
top\_products = total\_quantity\_sold.sort\_values(by='quantity sold', ascending=False)  
  
top\_5\_products = top\_products.head(5)  
  
print("Top 5 products sold the most in the past month:")  
print(top\_5\_products)

**Output:**

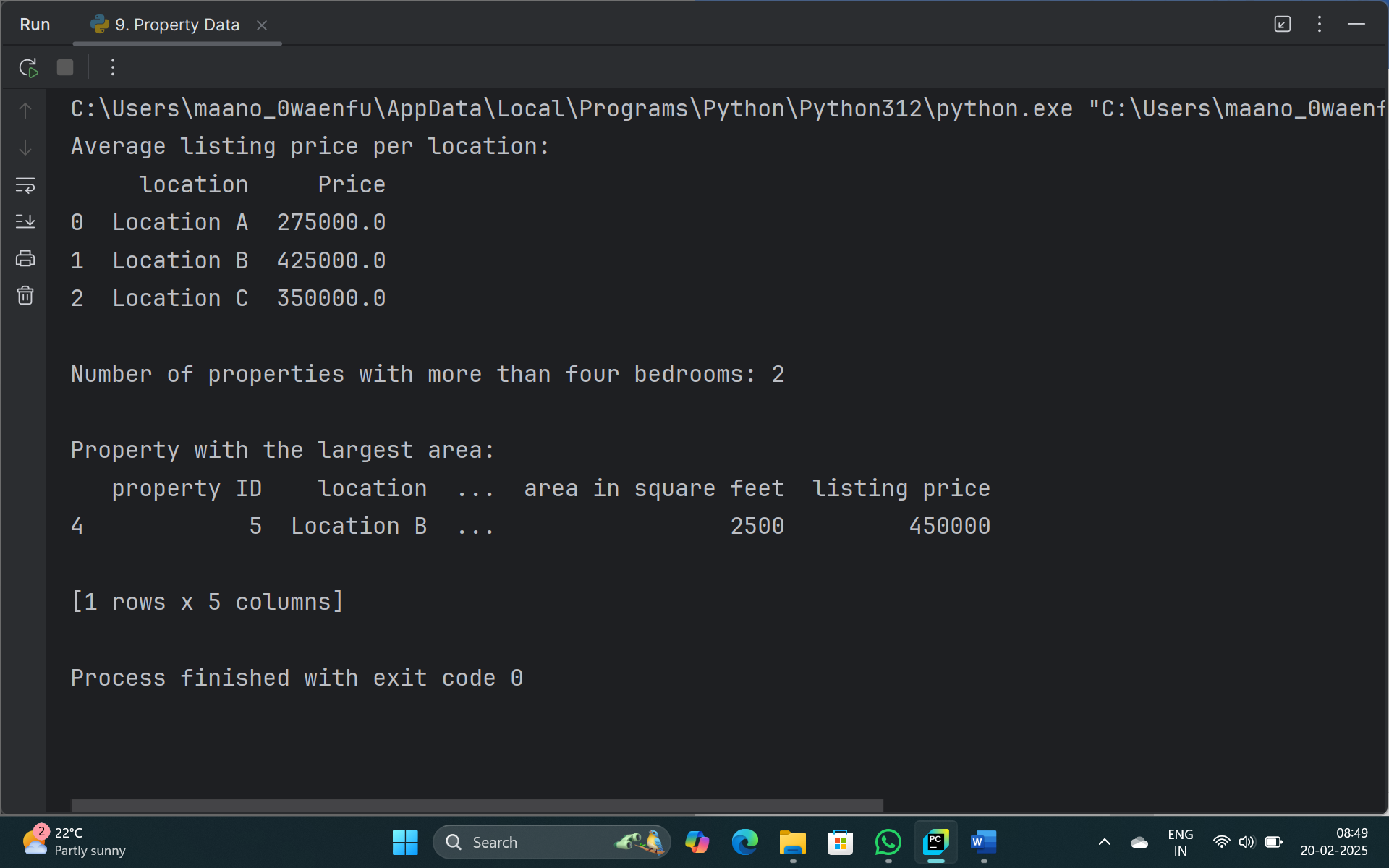


1. **Question:** Using Pandas DataFrame operations, how would you find the following information from the property\_data DataFrame:
2. The average listing price of properties in each location.
3. The number of properties with more than four bedrooms.
4. The property with the largest area.

**Code:**

import pandas as pd  
  
data = {  
 'property ID': [1, 2, 3, 4, 5],  
 'location': ['Location A', 'Location B', 'Location A', 'Location C', 'Location B'],  
 'number of bedrooms': [3, 5, 2, 4, 6],  
 'area in square feet': [1500, 2000, 1300, 1600, 2500],  
 'listing price': [300000, 400000, 250000, 350000, 450000]  
}  
  
property\_data = pd.DataFrame(data)  
  
average\_price\_per\_location = property\_data.groupby('location')['listing price'].mean().reset\_index(name="Price")  
print("Average listing price per location:")  
print(average\_price\_per\_location)  
  
properties\_more\_than\_four\_bedrooms = property\_data[property\_data['number of bedrooms'] > 4].shape[0]  
print("\nNumber of properties with more than four bedrooms:",properties\_more\_than\_four\_bedrooms)  
  
property\_largest\_area = property\_data[property\_data['area in square feet'] == property\_data['area in square feet'].max()]  
print("\nProperty with the largest area:")  
print(property\_largest\_area)

**Output:**

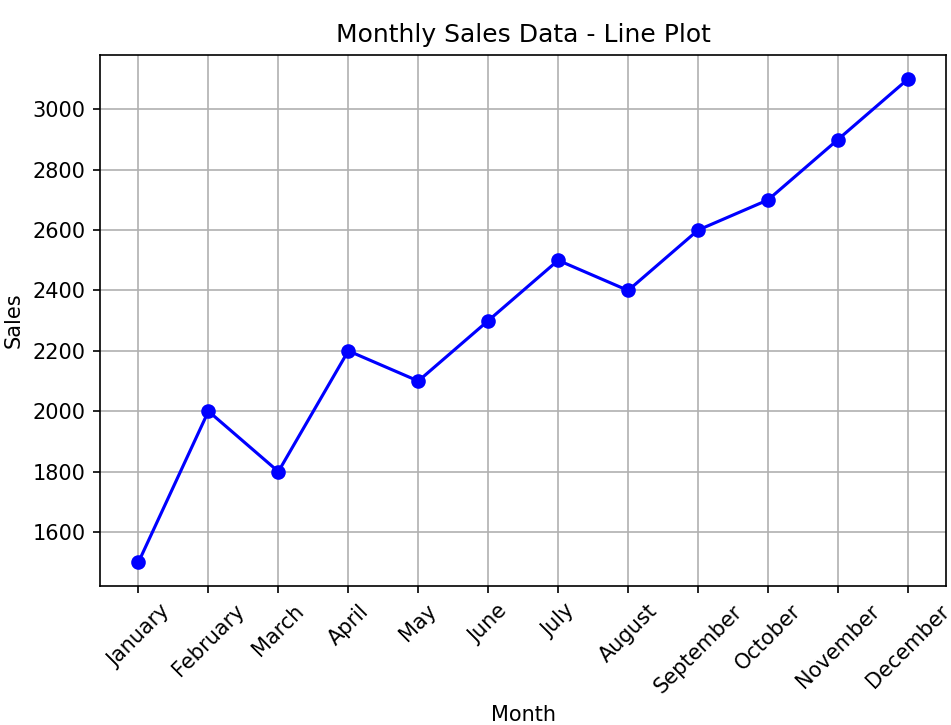
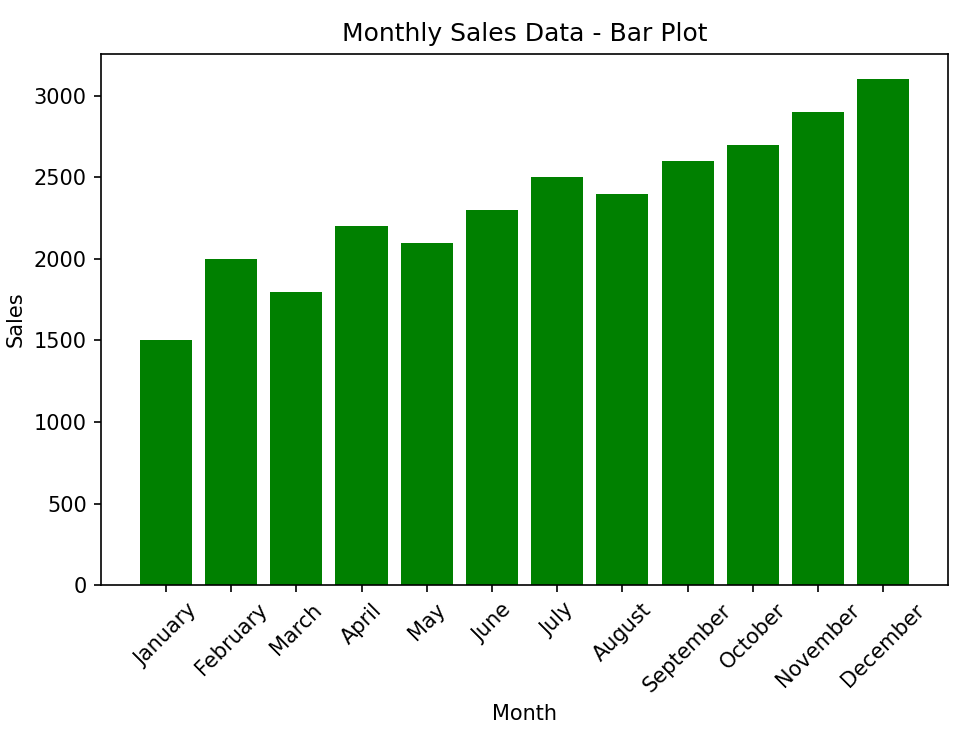


1. **Question:**
2. How would you develop a Python program to create a line plot of the monthly sales data?
3. How would you develop a Python program to create a bar plot of the monthly sales data?

**Code:**

import matplotlib.pyplot as plt  
  
months = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September', 'October', 'November', 'December']  
sales = [1500, 2000, 1800, 2200, 2100, 2300, 2500, 2400, 2600, 2700, 2900, 3100]  
  
plt.figure(figsize=(15, 5))  
  
plt.subplot(1, 2, 1)  
plt.plot(months, sales, marker='o', linestyle='-', color='b')  
plt.title('Monthly Sales Data - Line Plot')  
plt.xlabel('Month')  
plt.ylabel('Sales')  
plt.grid(True)  
plt.xticks(rotation=45)  
  
plt.subplot(1, 2, 2)  
plt.bar(months, sales, color='green')  
plt.title('Monthly Sales Data - Bar Plot')  
plt.xlabel('Month')  
plt.ylabel('Sales')  
plt.xticks(rotation=45)  
  
plt.tight\_layout()  
plt.show()

**Output:**

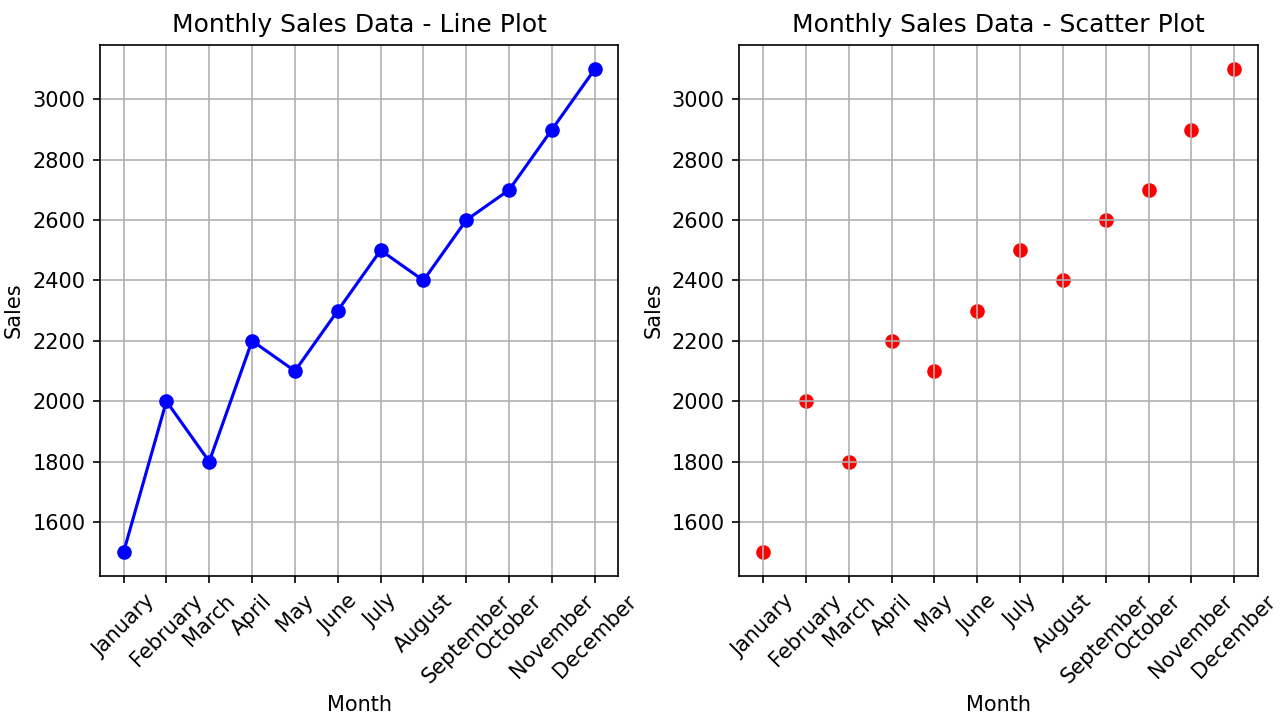
****

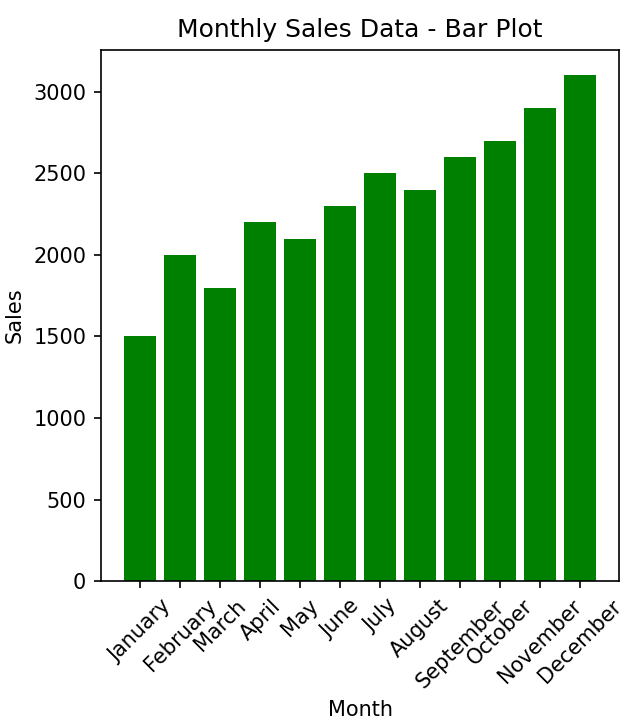
1. **Question:**
   1. Write code to create a simple line plot in Python using Matplotlib to predict sales happened in a month?
   2. Write code to create a scatter plot in Python using Matplotlib to predict sales happened in a month?
   3. Develop a Python program to create a bar plot of the monthly sales data.

**Code:**

import matplotlib.pyplot as plt  
  
months = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September', 'October', 'November', 'December']  
sales = [1500, 2000, 1800, 2200, 2100, 2300, 2500, 2400, 2600, 2700, 2900, 3100]  
  
plt.figure(figsize=(15, 5))  
  
plt.subplot(1, 3, 1)  
plt.plot(months, sales, marker='o', linestyle='-', color='b')  
plt.title('Monthly Sales Data - Line Plot')  
plt.xlabel('Month')  
plt.ylabel('Sales')  
plt.grid(True)  
plt.xticks(rotation=45)  
  
plt.subplot(1, 3, 2)  
plt.scatter(months, sales, color='red')  
plt.title('Monthly Sales Data - Scatter Plot')  
plt.xlabel('Month')  
plt.ylabel('Sales')  
plt.grid(True)  
plt.xticks(rotation=45)  
  
plt.subplot(1, 3, 3)  
plt.bar(months, sales, color='green')  
plt.title('Monthly Sales Data - Bar Plot')  
plt.xlabel('Month')  
plt.ylabel('Sales')  
plt.xticks(rotation=45)  
  
plt.tight\_layout()  
  
plt.show()

**Output:**



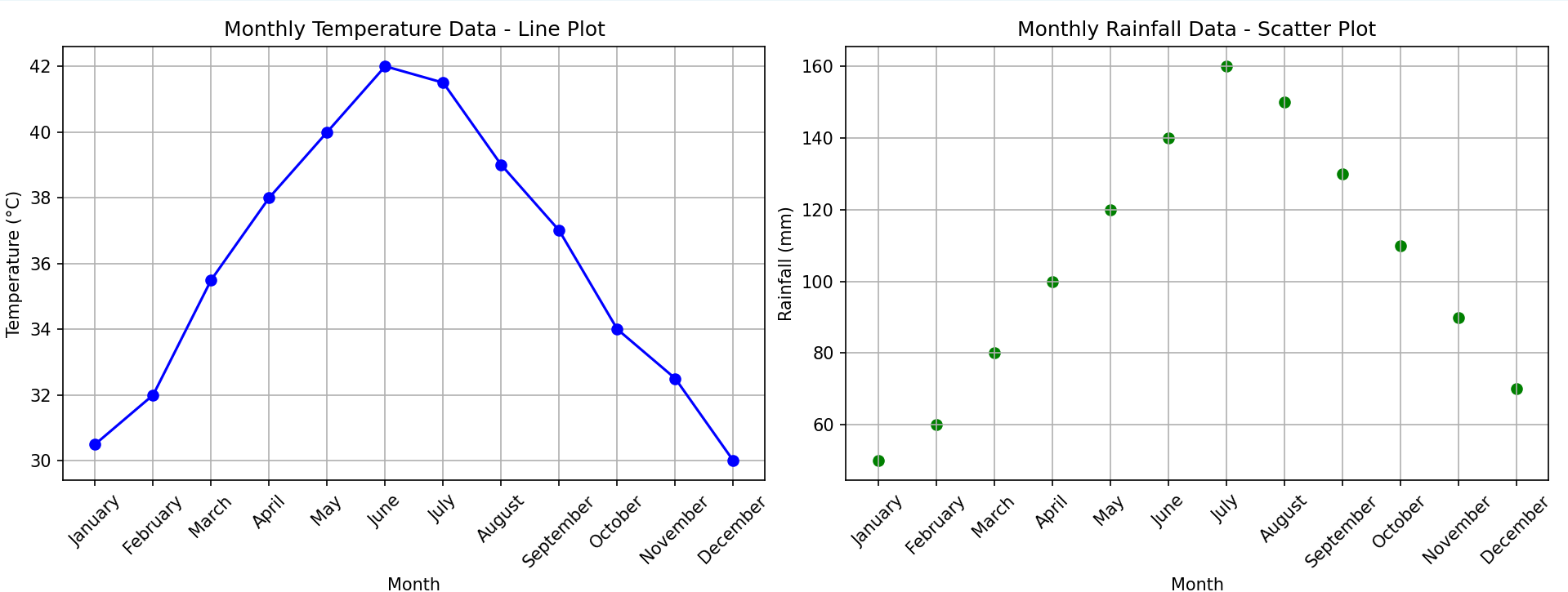


1. **Question:**
   1. Develop a Python program to create a line plot of the monthly temperature data.
   2. Develop a Python program to create a scatter plot of the monthly rainfall data.

**Code:**

import matplotlib.pyplot as plt  
  
months = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September', 'October', 'November', 'December']  
temperature = [30.5, 32.0, 35.5, 38.0, 40.0, 42.0, 41.5, 39.0, 37.0, 34.0, 32.5, 30.0]  
rainfall = [50, 60, 80, 100, 120, 140, 160, 150, 130, 110, 90, 70]  
  
plt.figure(figsize=(15, 5))  
  
plt.subplot(1, 2, 1)  
plt.plot(months, temperature, marker='o', linestyle='-', color='b')  
plt.title('Monthly Temperature Data - Line Plot')  
plt.xlabel('Month')  
plt.ylabel('Temperature (°C)')  
plt.grid(True)  
plt.xticks(rotation=45)  
  
plt.subplot(1, 2, 2)  
plt.scatter(months, rainfall, color='green')  
plt.title('Monthly Rainfall Data - Scatter Plot')  
plt.xlabel('Month')  
plt.ylabel('Rainfall (mm)')  
plt.grid(True)  
plt.xticks(rotation=45)  
  
plt.tight\_layout()  
plt.show()

**Output:**

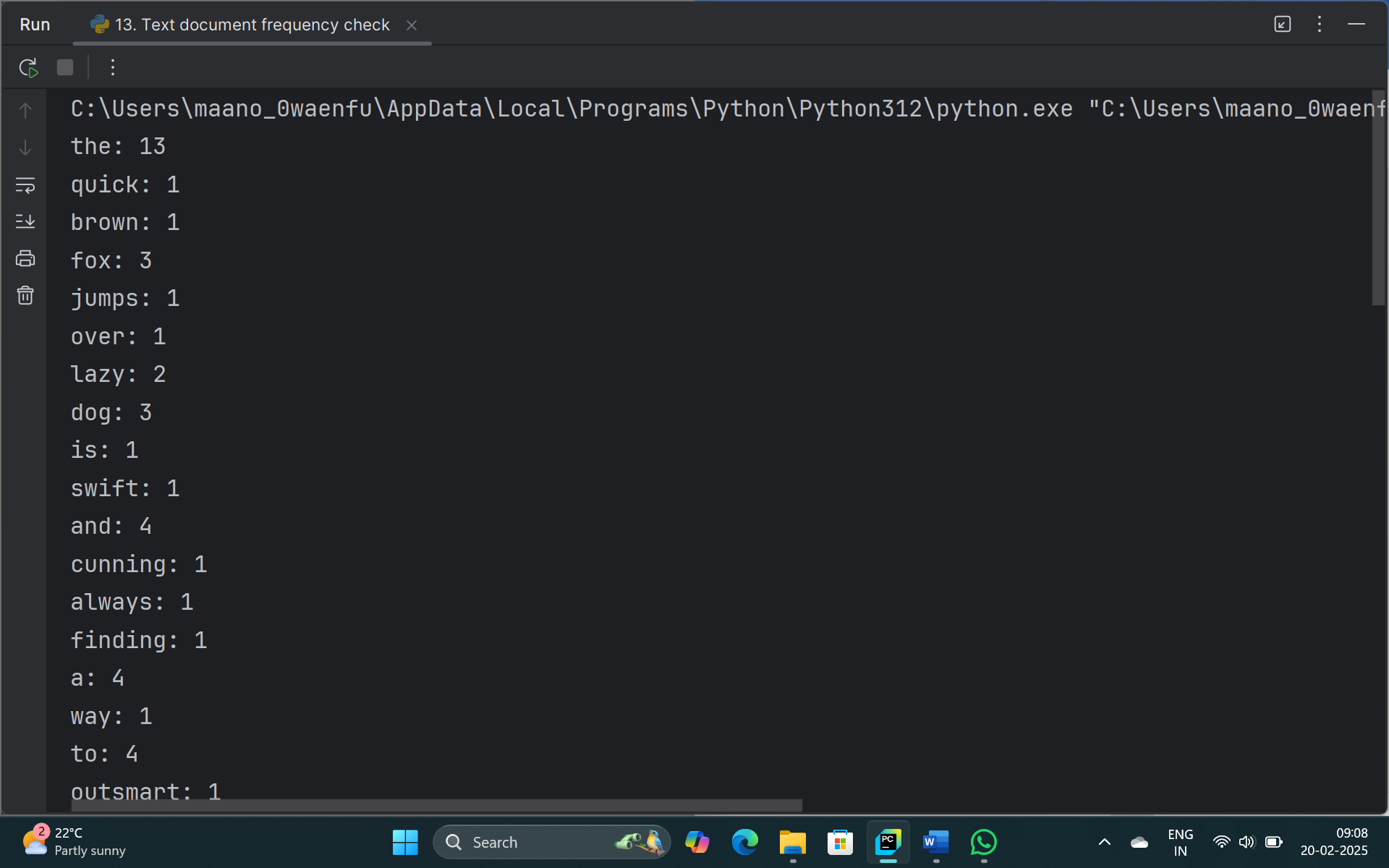
****

1. **Question:** How would you develop a Python program to calculate the frequency distribution of words in a text document?

**Code:**

import string  
from collections import Counter  
import nltk  
  
def process\_text(file\_path):  
 with open(file\_path, 'r') as file:  
 text = file.read()  
  
 text = text.lower()  
  
 text = text.translate(str.maketrans('', '', string.punctuation))  
  
 words = nltk.word\_tokenize(text)  
  
 return words  
  
def word\_frequency(words):  
 word\_counts = Counter(words)  
 return word\_counts  
  
file\_path = "sample\_text.txt"  
  
words = process\_text(file\_path)  
word\_counts = word\_frequency(words)  
  
for word, count in word\_counts.items():  
 print(f'{word}: {count}')

**Output:**

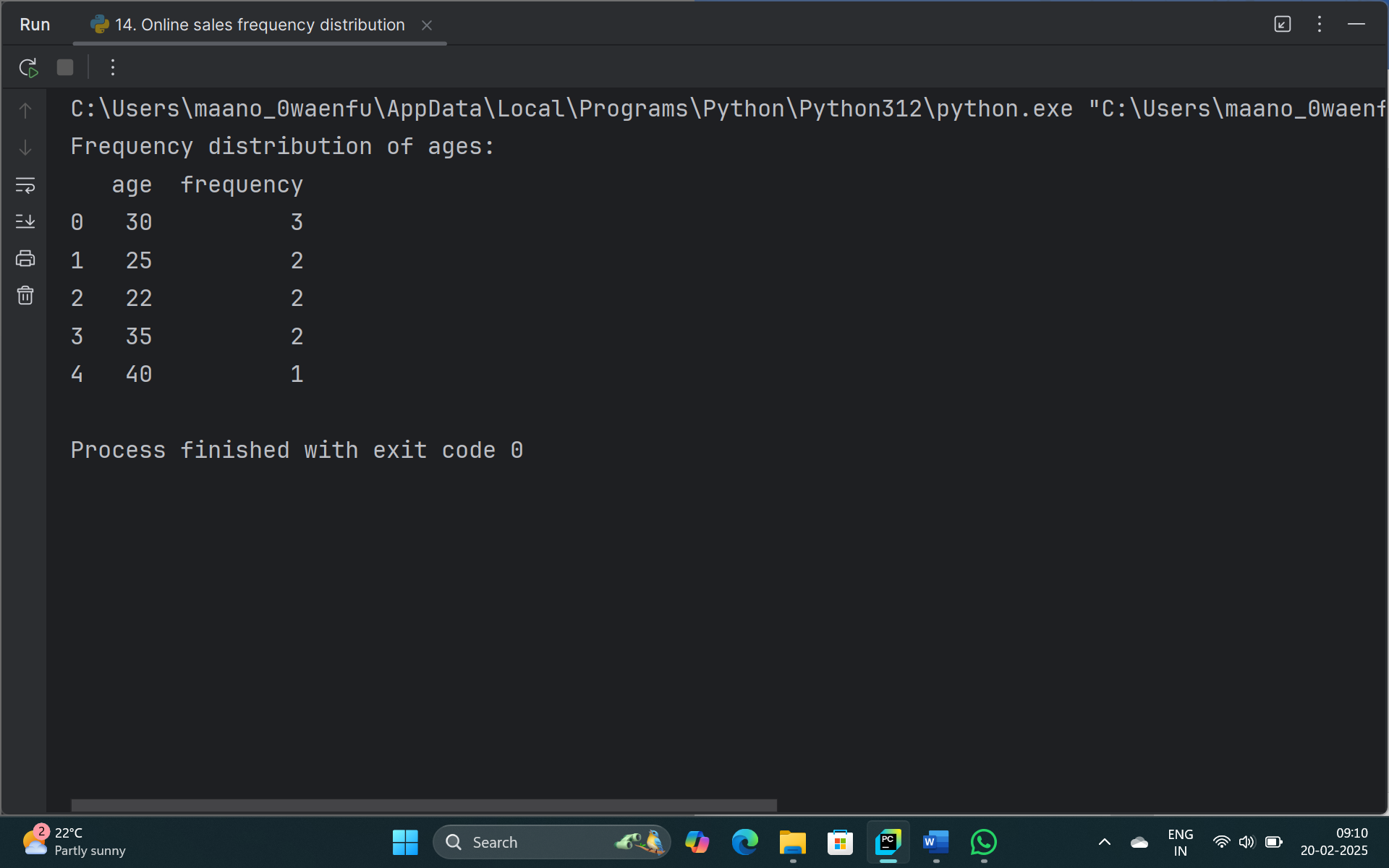


1. **Question:** Develop a code in python to find the frequency distribution of the ages of the customers who have made a purchase in the past month.

**Code:**

import pandas as pd  
from collections import Counter  
  
data = {  
 'customer\_id': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],  
 'age': [25, 30, 22, 35, 30, 25, 40, 22, 30, 35],  
 'purchase\_amount': [100, 150, 80, 200, 130, 120, 160, 70, 90, 140]  
}  
  
sales\_data = pd.DataFrame(data)  
  
age\_counts = sales\_data['age'].value\_counts().reset\_index(name="frequency")  
  
print("Frequency distribution of ages:")  
print(age\_counts)

**Output:**

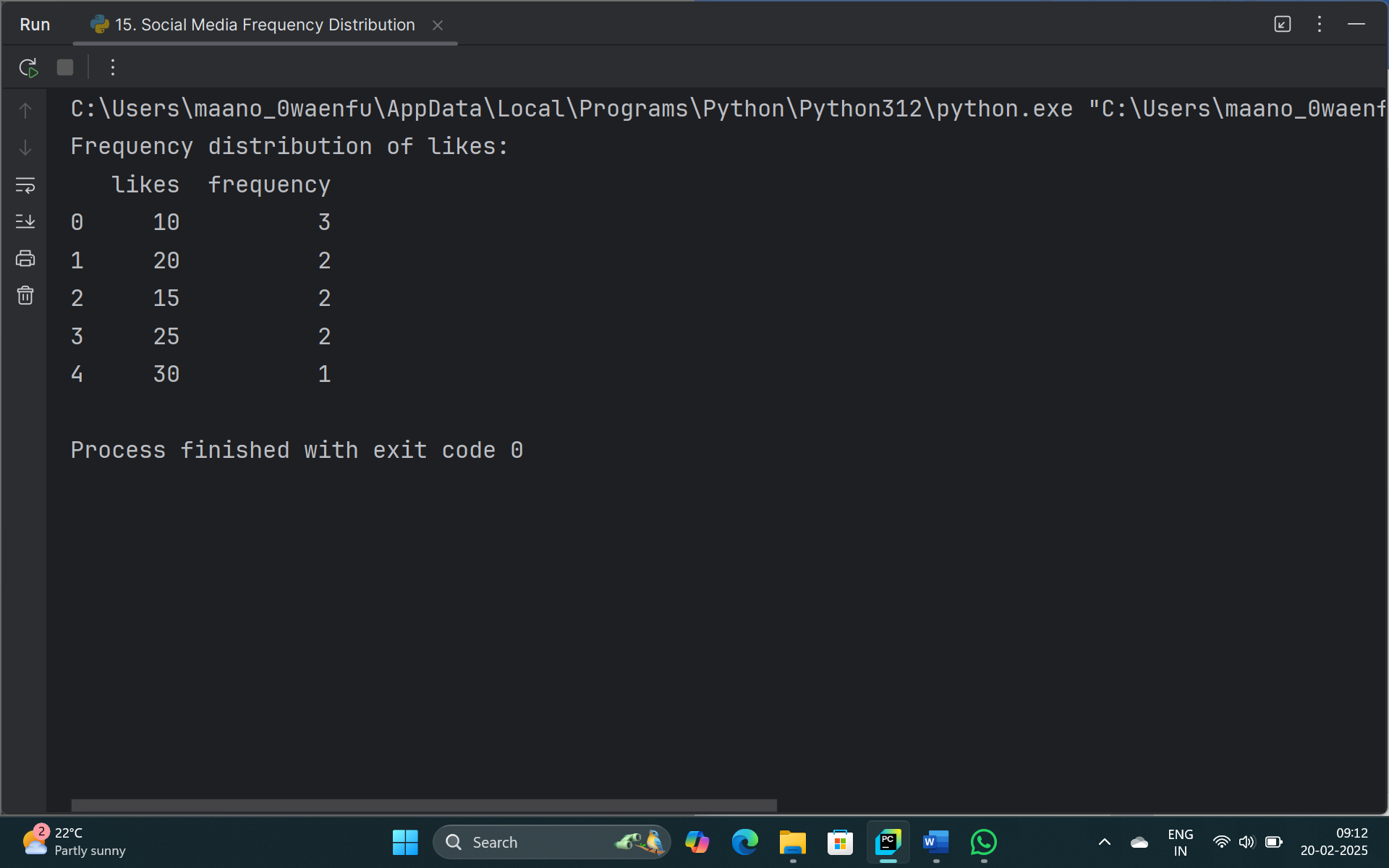


1. **Question:** Develop a Python program to calculate the frequency distribution of likes among the posts?

**Code:**

import pandas as pd  
from collections import Counter  
  
data = pd.read\_csv("social media.csv")  
interaction\_data = pd.DataFrame(data)  
  
like\_counts = interaction\_data['likes'].value\_counts().reset\_index(name="frequency")  
  
print("Frequency distribution of likes:")  
print(like\_counts)

**Output:**

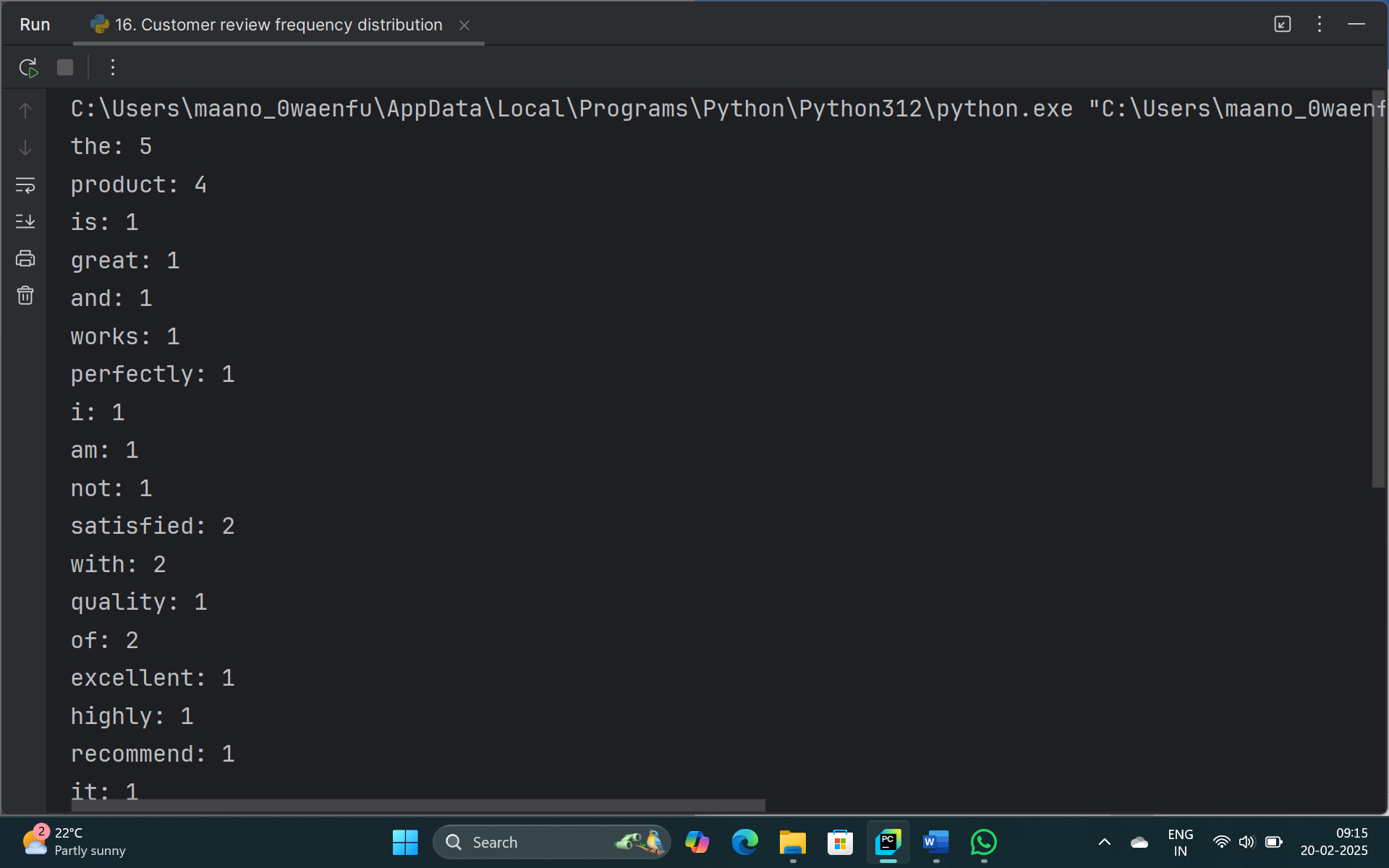


1. **Question:** Develop a Python program to calculate the frequency distribution of words in the customer reviews dataset?

**Code:**

import pandas as pd  
import string  
from collections import Counter  
import nltk  
  
data = pd.read\_csv("review.csv")  
  
reviews\_data = pd.DataFrame(data)  
  
def process\_text(text):  
  
 text = text.lower()  
  
 text = text.translate(str.maketrans('', '', string.punctuation))  
  
 words = nltk.word\_tokenize(text)  
 return words  
  
  
def word\_frequency(reviews):  
 all\_words = []  
 for review in reviews:  
 words = process\_text(review)  
 all\_words.extend(words)  
 word\_counts = Counter(all\_words)  
 return word\_counts  
  
  
review\_texts = reviews\_data['text']  
  
word\_counts = word\_frequency(review\_texts)  
  
for word, count in word\_counts.items():  
 print(f'{word}: {count}')

**Output:**

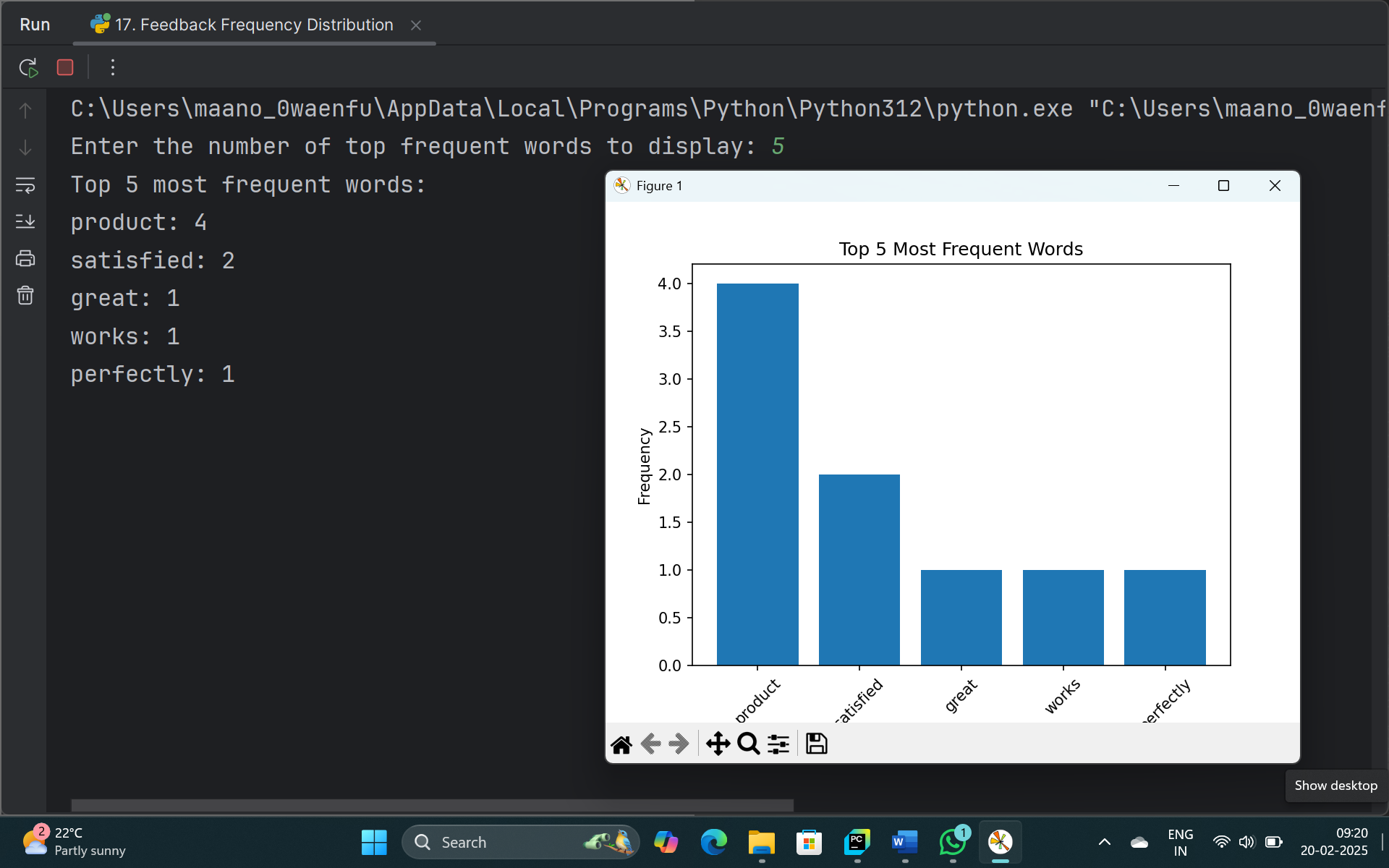


1. **Question**: Create a Python program that fulfills these requirements and gain insights from the customer feedback data.
   * Load the dataset from a CSV file (data.csv) containing a single column named "feedback" with each row representing a customer comment.
   * Preprocess the text data by removing punctuation, converting all text to lowercase, and eliminating any stop words (common words like "the," "and," "is," etc. that don't carry significant meaning).
   * Calculate the frequency distribution of words in the preprocessed dataset.
   * Display the top N most frequent words and their corresponding frequencies, where N is provided as user input.
   * Plot a bar graph to visualize the top N most frequent words and their frequencies.

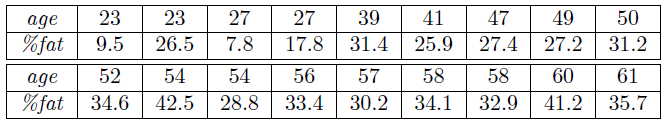
**Code:**

import pandas as pdimport string  
from collections import Counter  
from nltk.corpus import stopwords  
import matplotlib.pyplot as plt  
  
def preprocess\_text(text):  
 text = text.lower()  
 text = text.translate(str.maketrans('', '', string.punctuation))  
 stop\_words = set(stopwords.words('english'))  
 words = text.split() # Split text into words  
 filtered\_words = [word for word in words if word not in stop\_words]  
 return filtered\_words  
  
df = pd.read\_csv('data.csv')  
feedback\_texts = df['feedback'].tolist()  
  
all\_words = []  
for feedback in feedback\_texts:  
 all\_words.extend(preprocess\_text(feedback))  
  
word\_counts = Counter(all\_words)  
  
N = int(input("Enter the number of top frequent words to display: "))  
  
top\_n\_words = word\_counts.most\_common(N)  
print(f"Top {N} most frequent words:")  
for word, count in top\_n\_words:  
 print(f"{word}: {count}")  
  
words, counts = zip(\*top\_n\_words)  
plt.bar(words, counts)  
plt.xlabel('Words')  
plt.ylabel('Frequency')  
plt.title(f'Top {N} Most Frequent Words')  
plt.xticks(rotation=45)  
plt.show()

**Output:**



1. **Question:** Suppose a hospital tested the age and body fat data for 18 randomly selected adults with the following result.



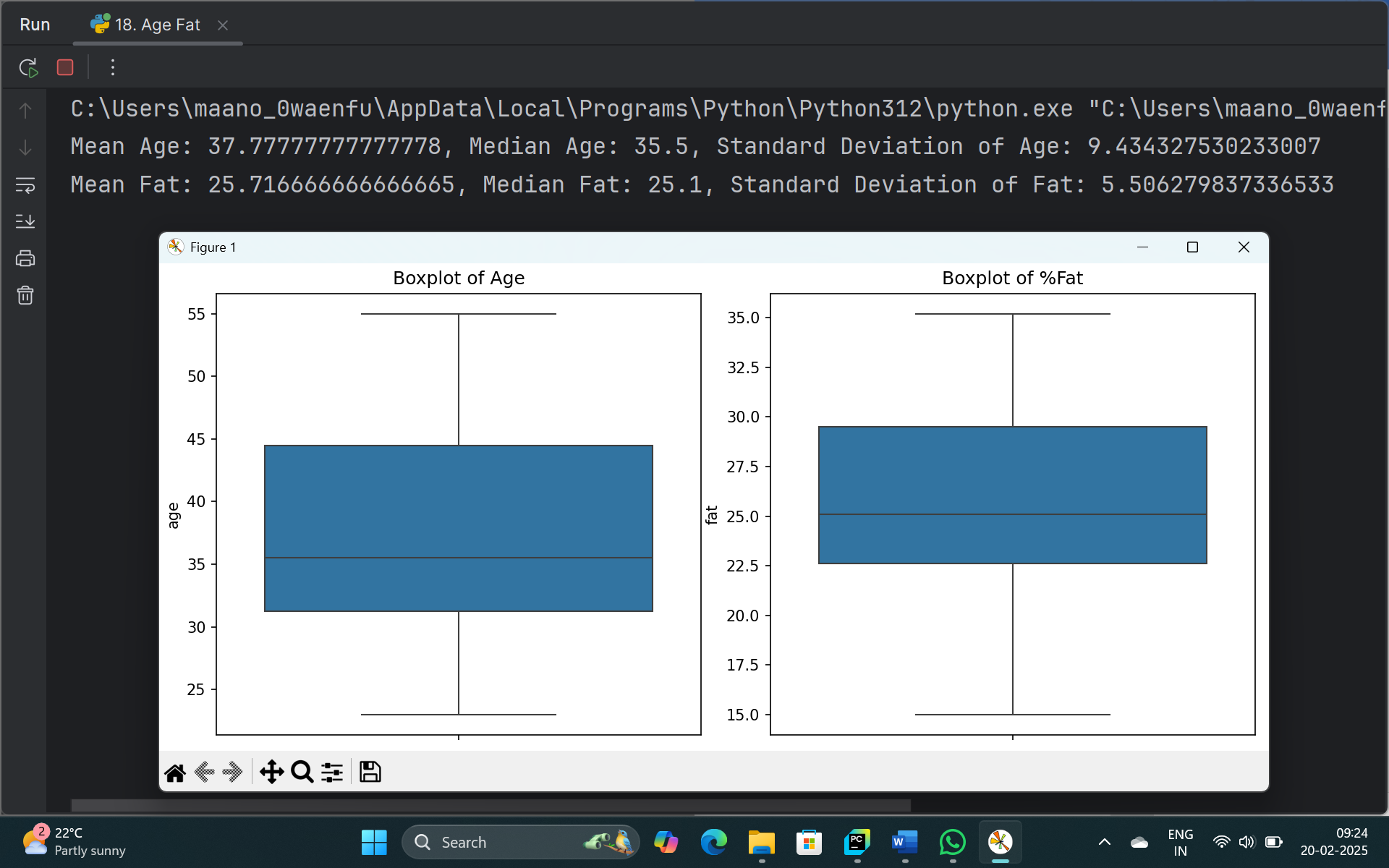
* Calculate the mean, median and standard deviation of age and %fat using Pandas.
* Draw the boxplots for age and %fat.
* Draw a scatter plot and a q-q plot based on these two variables.

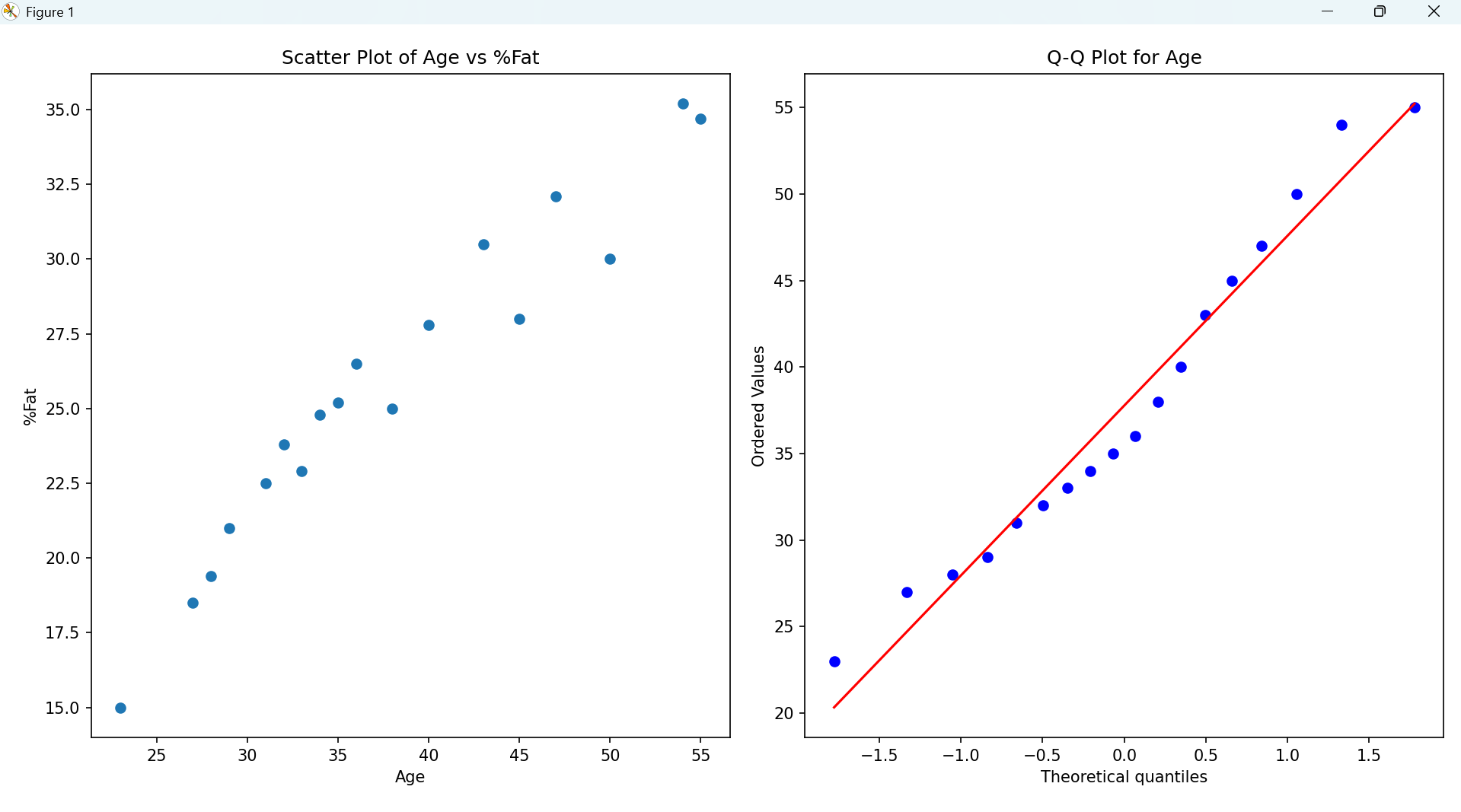
**Code:**

import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns  
import scipy.stats as stats  
  
data = {  
 'age': [23, 45, 31, 35, 43, 54, 29, 40, 36, 32, 55, 47, 50, 28, 38, 27, 34, 33],  
 'fat': [15.0, 28.0, 22.5, 25.2, 30.5, 35.2, 21.0, 27.8, 26.5, 23.8, 34.7, 32.1, 30.0, 19.4, 25.0, 18.5, 24.8, 22.9]  
}  
df = pd.DataFrame(data)  
  
# Calculating mean, median, and standard deviation  
mean\_age = df['age'].mean()  
median\_age = df['age'].median()  
std\_age = df['age'].std()  
  
mean\_fat = df['fat'].mean()  
median\_fat = df['fat'].median()  
std\_fat = df['fat'].std()  
  
print(f"Mean Age: {mean\_age}, Median Age: {median\_age}, Standard Deviation of Age: {std\_age}")  
print(f"Mean Fat: {mean\_fat}, Median Fat: {median\_fat}, Standard Deviation of Fat: {std\_fat}")

plt.figure(figsize=(12, 6))  
  
plt.subplot(1, 2, 1)  
sns.boxplot(df['age'])  
plt.title('Boxplot of Age')  
  
plt.subplot(1, 2, 2)  
sns.boxplot(df['fat'])  
plt.title('Boxplot of %Fat')  
  
plt.tight\_layout()  
plt.show()  
  
plt.figure(figsize=(12, 6))  
  
plt.subplot(1, 2, 1)  
plt.scatter(df['age'], df['fat'])  
plt.xlabel('Age')  
plt.ylabel('%Fat')  
plt.title('Scatter Plot of Age vs %Fat')  
  
plt.subplot(1, 2, 2)  
stats.probplot(df['age'], dist="norm", plot=plt)  
plt.title('Q-Q Plot for Age')  
  
plt.tight\_layout()  
plt.show()

**Output:**



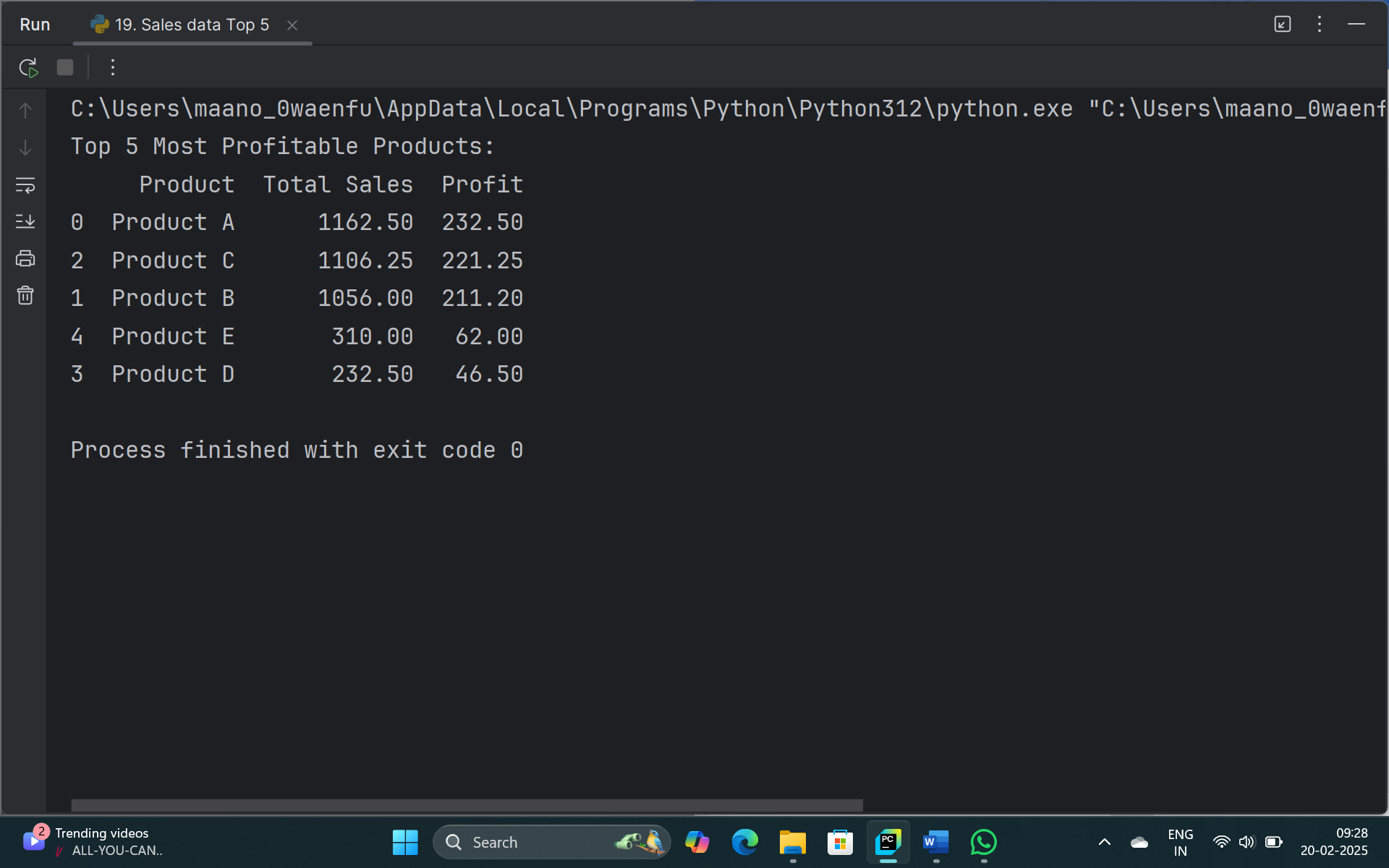


1. **Sales and Profit Analysis:**
   * 1. Load the "sales\_data.csv" file into a Pandas data frame, which contains columns "Date," "Product," "Quantity Sold," and "Unit Price."
     2. Create a new column named "Total Sales" that calculates the total sales for each transaction (Quantity Sold \* Unit Price).
     3. Calculate the total sales for each product and the overall profit, considering a 20% profit margin on each product. Display the top 5 most profitable products.

**Code:**

import pandas as pd  
  
df = pd.read\_csv('sales\_data.csv')  
  
df['Total Sales'] = df['Quantity Sold'] \* df['Unit Price']  
  
product\_sales = df.groupby('Product')['Total Sales'].sum().reset\_index()  
product\_sales['Profit'] = product\_sales['Total Sales'] \* 0.20  
top\_products = product\_sales.sort\_values(by='Profit', ascending=False).head(5)  
  
print("Top 5 Most Profitable Products:")  
print(top\_products)

**Output:**



1. **Customer Segmentation:**

a) Load the "customer\_data.csv" file into a Pandas data frame, which contains columns "Customer ID," "Age," "Gender," and "Total Spending."

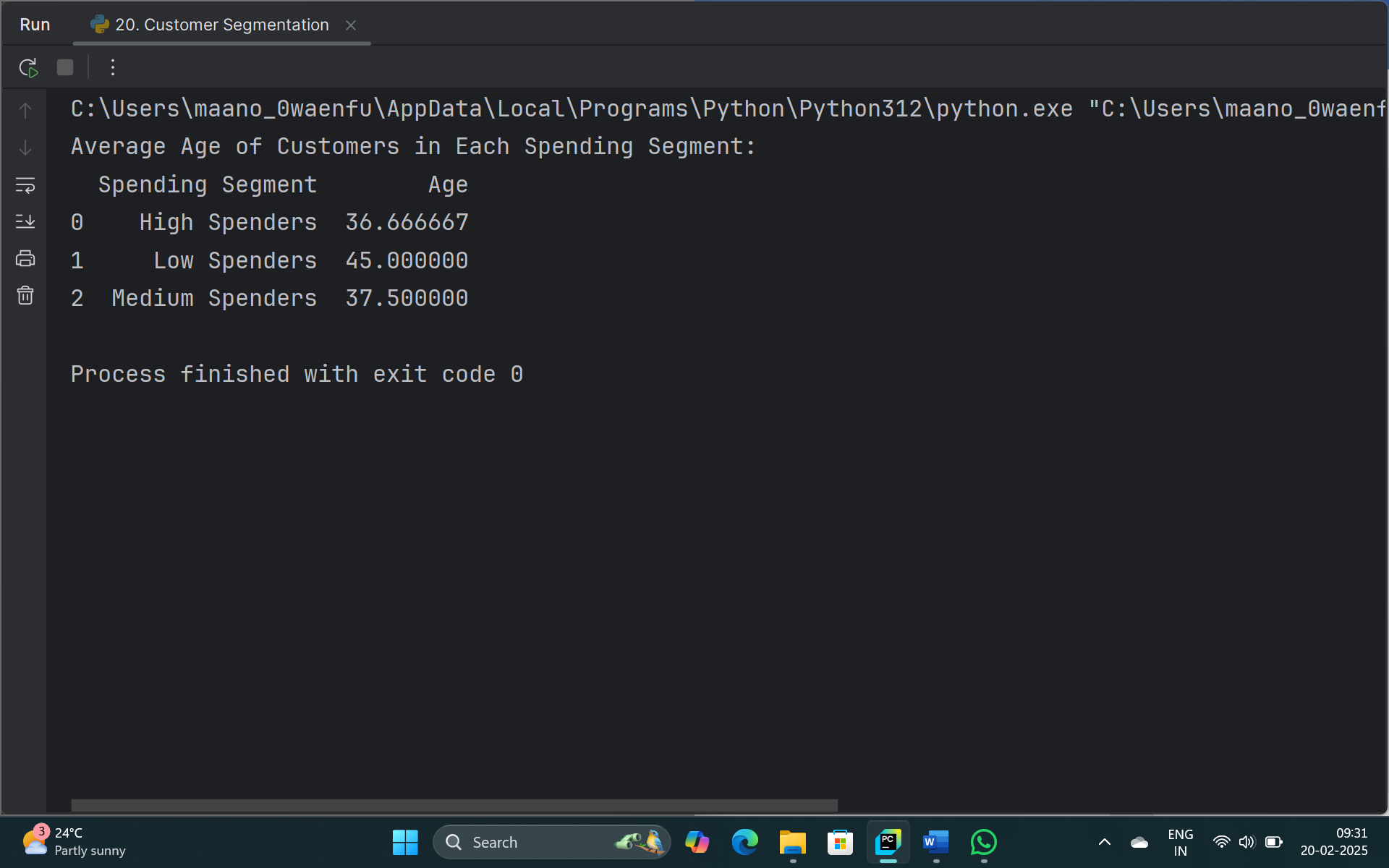
b) Segment customers into three groups based on their total spending: "High Spenders," "Medium Spenders," and "Low Spenders." Assign these segments to a new column in the data frame.

c) Calculate the average age of customers in each spending segment.

**Code:**

import pandas as pd  
  
df = pd.read\_csv('customer\_data.csv')  
  
def spending\_category(spending):  
 if spending > 1000:  
 return 'High Spenders'  
 elif spending > 500:  
 return 'Medium Spenders'  
 else:  
 return 'Low Spenders'  
  
df['Spending Segment'] = df['Total Spending'].apply(spending\_category)  
  
average\_age\_by\_segment = df.groupby('Spending Segment')['Age'].mean().reset\_index()  
  
print("Average Age of Customers in Each Spending Segment:")  
print(average\_age\_by\_segment)

**Output:**



1. **Data Cleaning and Transformation:**

a) Load the "employee\_data.csv" file into a Pandas data frame, which contains columns "Employee ID," "Full Name," "Department," and "Salary."

b) Convert the "Salary" column to numeric data type.

c) Remove any rows with missing values in the "Department" column.

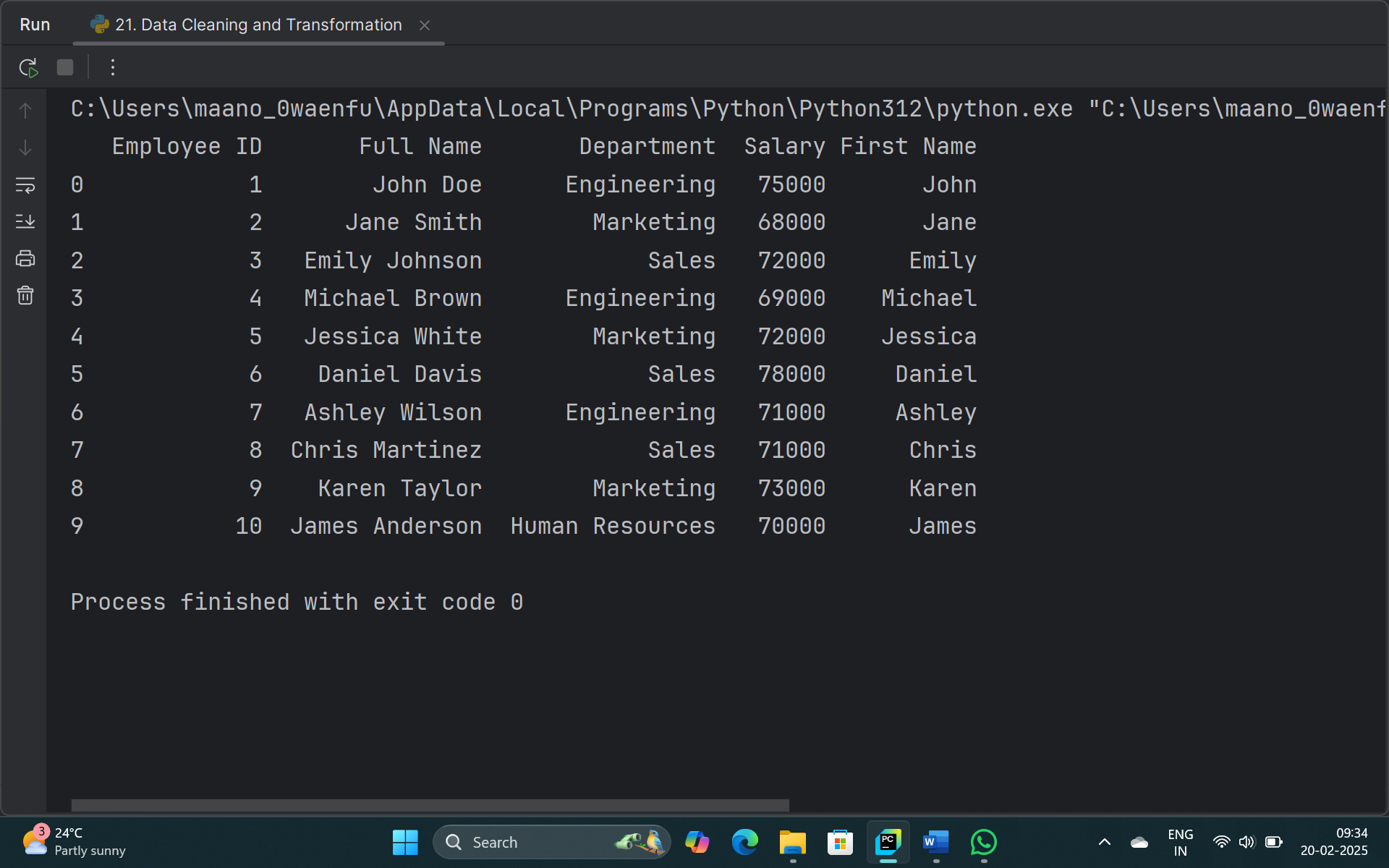
d) Create a new column named "First Name" that extracts the first name from the "Full Name" column.

**Code:**

import pandas as pd

df = pd.read\_csv('employee\_data.csv')  
  
df['Salary'] = pd.to\_numeric(df['Salary'], errors='coerce')  
  
df = df.dropna(subset=['Department'])  
  
df['First Name'] = df['Full Name'].apply(lambda x: x.split()[0])  
  
print(df)

**Output:**



1. **Time Series Analysis:**

a) Load the "temperature\_data.csv" file into a Pandas data frame, which contains columns "Date" and "Temperature (Celsius)."

b) Convert the "Date" column to a Pandas datetime data type.

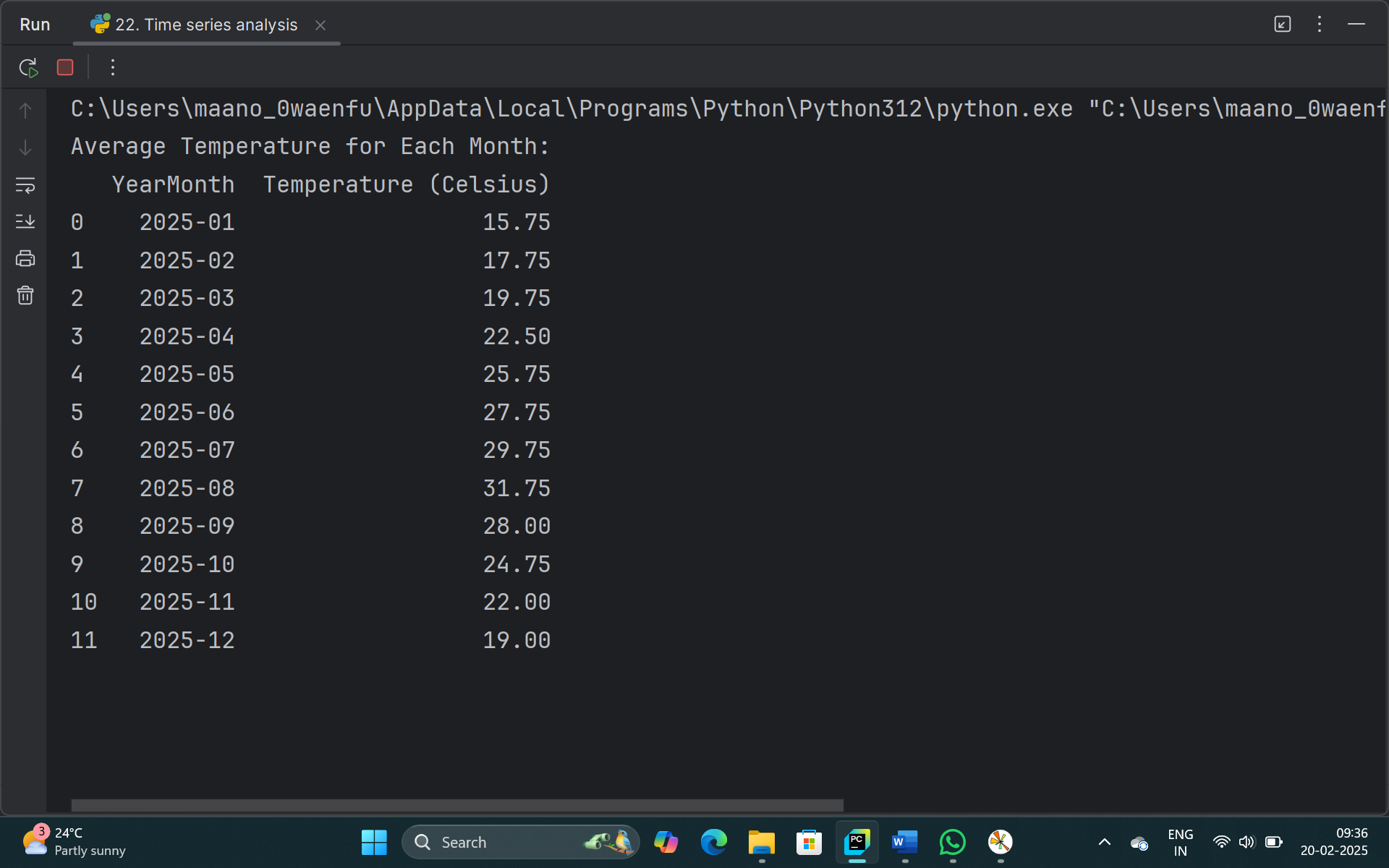
c) Calculate the average temperature for each month and display the results in chronological order.

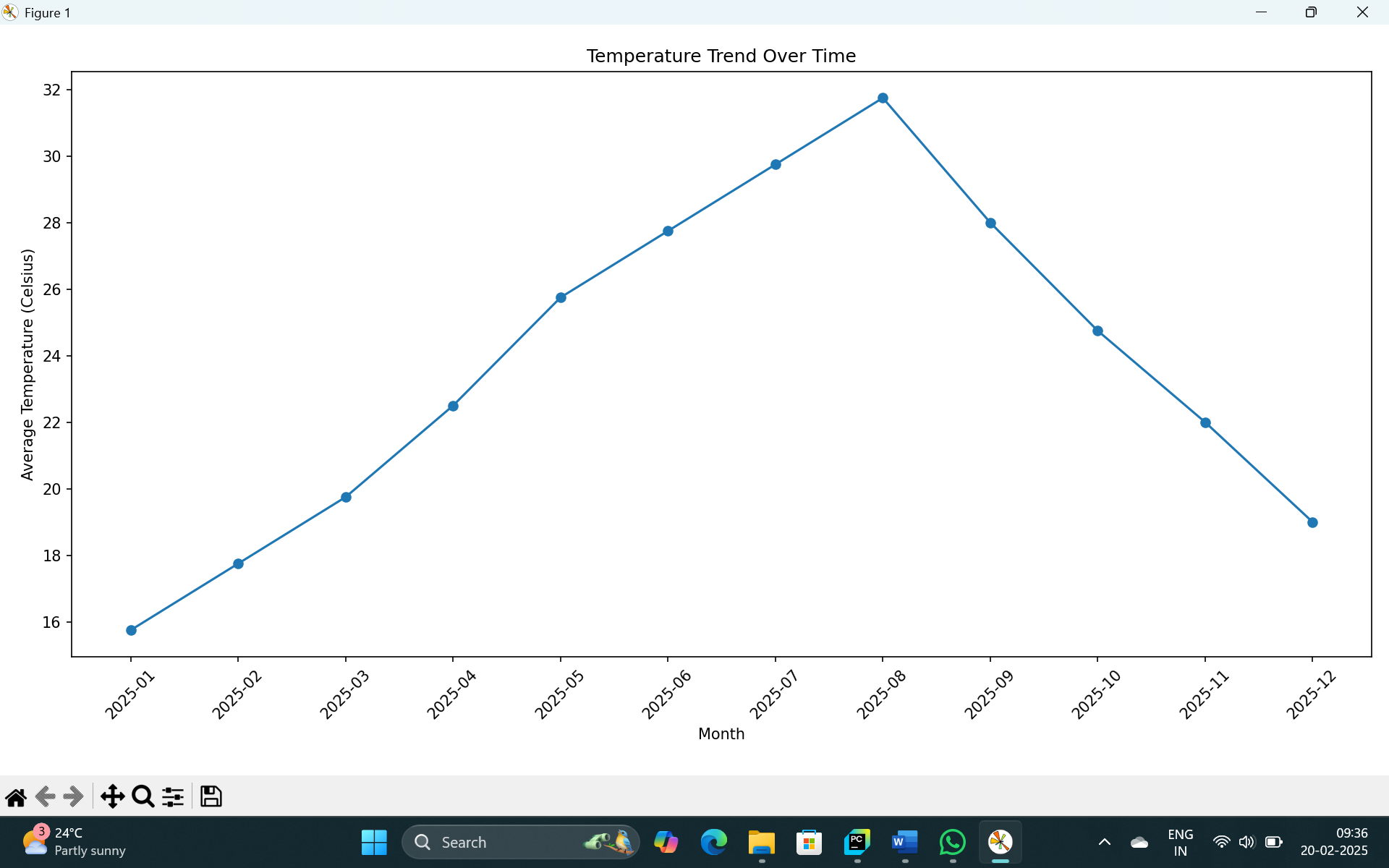
d) Plot a line chart to visualize the temperature trend over time.

**Code:**

import pandas as pd  
import matplotlib.pyplot as plt  
  
# Step a) Load the "temperature\_data.csv" file into a Pandas DataFrame  
df = pd.read\_csv('temperature\_data.csv')  
  
# Step b) Convert the "Date" column to a Pandas datetime data type  
df['Date'] = pd.to\_datetime(df['Date'])  
  
# Step c) Calculate the average temperature for each month  
df['YearMonth'] = df['Date'].dt.to\_period('M')  
monthly\_avg\_temp = df.groupby('YearMonth')['Temperature (Celsius)'].mean().reset\_index()  
  
# Display the average temperature for each month in chronological order  
print("Average Temperature for Each Month:")  
print(monthly\_avg\_temp)  
  
# Step d) Plot a line chart to visualize the temperature trend over time  
plt.figure(figsize=(12, 6))  
plt.plot(monthly\_avg\_temp['YearMonth'].astype(str), monthly\_avg\_temp['Temperature (Celsius)'], marker='o')  
plt.xlabel('Month')  
plt.ylabel('Average Temperature (Celsius)')  
plt.title('Temperature Trend Over Time')  
plt.xticks(rotation=45)  
plt.tight\_layout()  
plt.show()

**Output:**





1. **Joining Data frames**:

a) Load the "orders\_data.csv" file into a Pandas data frame, which contains columns "Order ID," "Customer ID," and "Order Date."

b) Load the "customer\_info.csv" file into another Pandas data frame, which contains columns "Customer ID," "Name," "Email," and "Phone Number."

c) Merge the two data frames based on the "Customer ID" column to create a new data frame that includes both order information and customer details.

d) Calculate the average time it takes for a customer to place another order after their first order (time between consecutive orders).

**Code:**

import pandas as pd

orders\_df = pd.read\_csv('orders\_data.csv')  
  
customers\_df = pd.read\_csv('customer\_info.csv')  
  
merged\_df = pd.merge(orders\_df, customers\_df, on='Customer ID')  
  
print("Merged DataFrame:")  
print(merged\_df)  
  
merged\_df['Order Date'] = pd.to\_datetime(merged\_df['Order Date'])  
  
merged\_df = merged\_df.sort\_values(by=['Customer ID', 'Order Date'])  
  
merged\_df['Time Difference'] = merged\_df.groupby('Customer ID')['Order Date'].diff()  
  
average\_time\_difference = merged\_df.groupby('Customer ID')['Time Difference'].mean().reset\_index()  
  
overall\_average\_time\_difference = average\_time\_difference['Time Difference'].mean()  
  
print(f"Overall Average Time Between Orders: {overall\_average\_time\_difference}")

**Output:**

