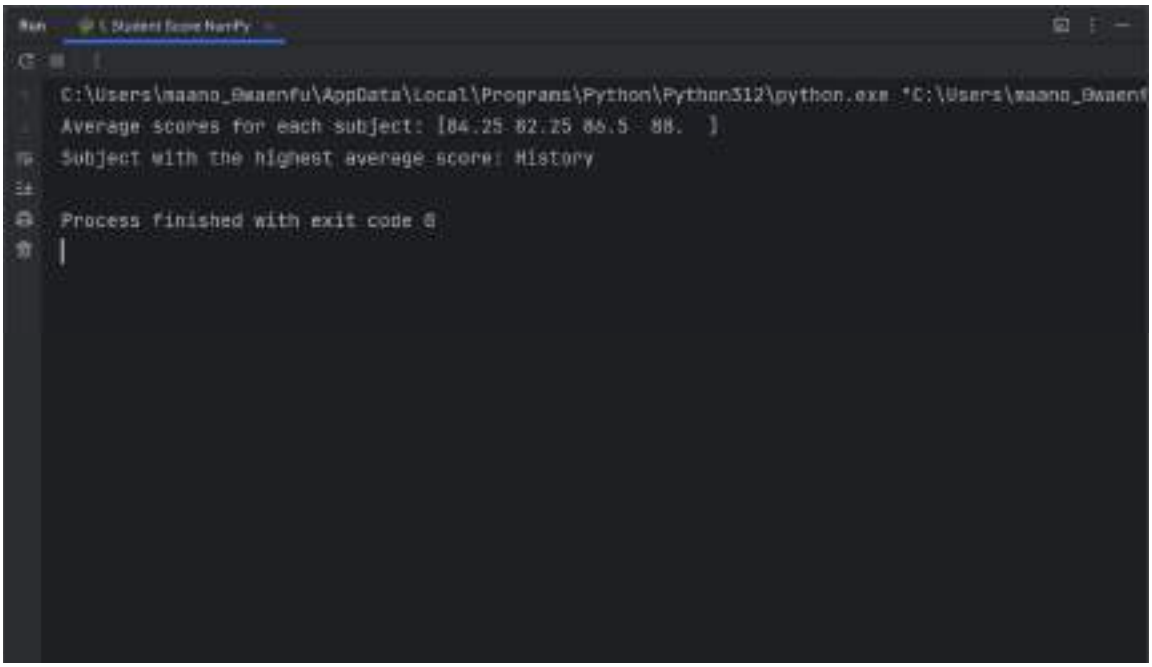
1. **Question:** How would you use NumPy arrays to calculate the average score for each subject and determine the subject with the highest average score? Assume 4x4 matrix that stores marks of each student in given order.

   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:

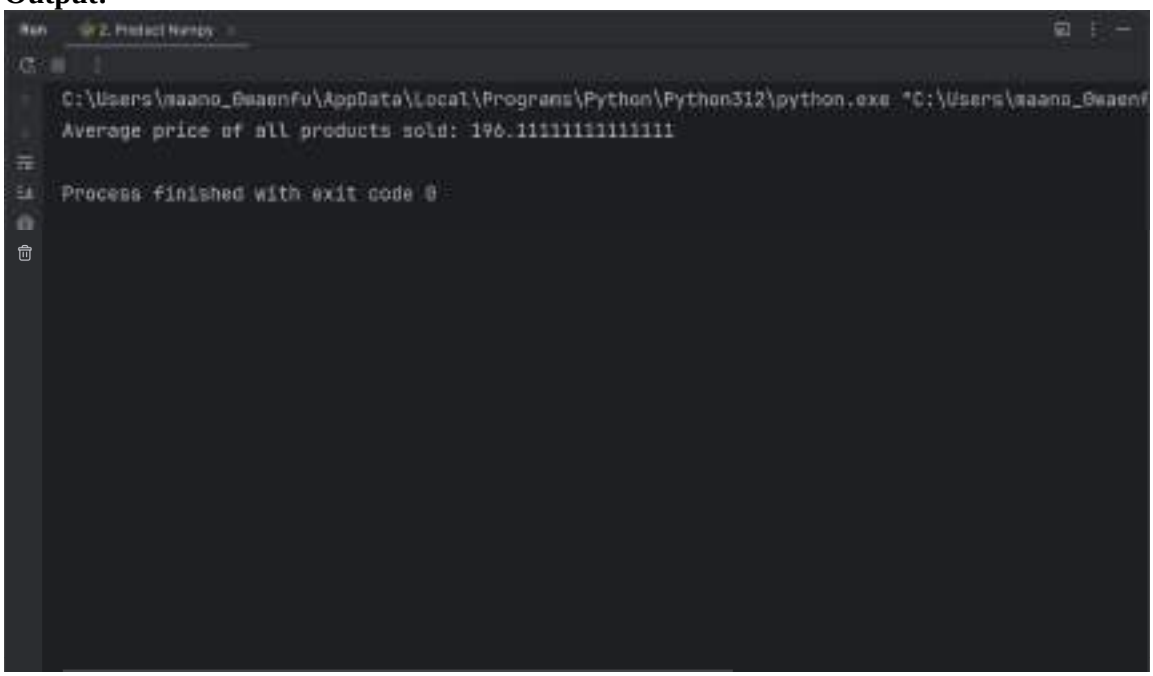


2. **Question:** How would you find the average price of all the products sold in the past month? Assume 3x3 matrix with each row representing the sales for a different product

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:

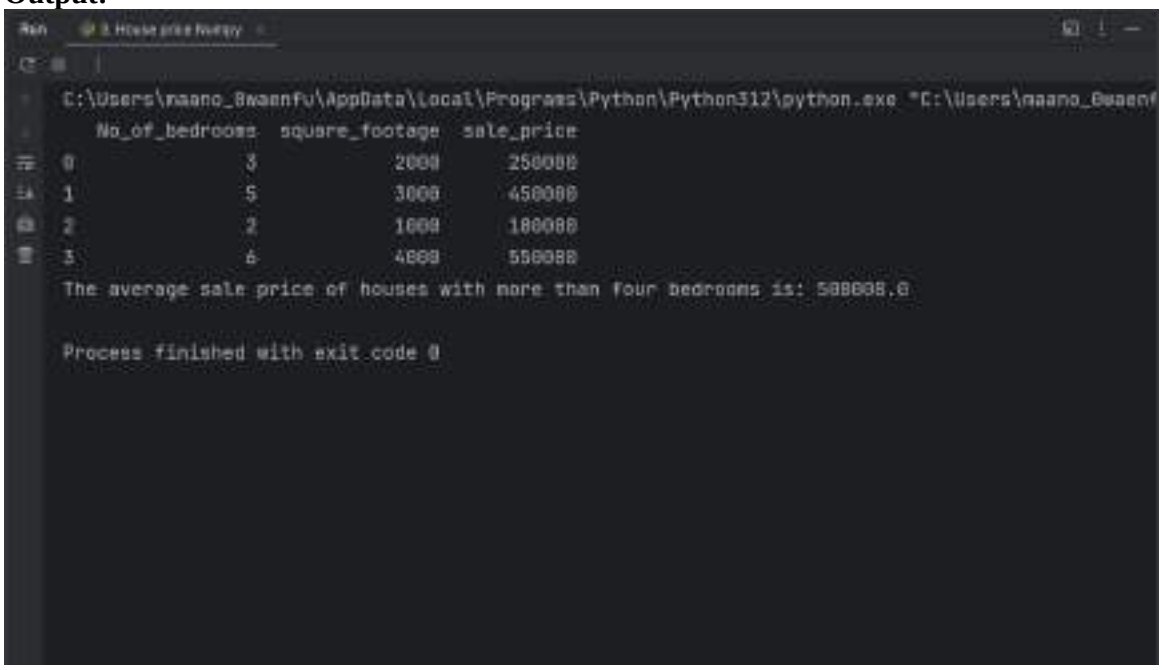


3. **Question:** Using NumPy arrays and operations, how would you find the average sale price of houses with more than four bedrooms in the neighborhood?

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:



```
C:\Users\maano_Owaenfu\AppData\Local\Programs\Python\Python312\python.exe "C:\Users\maano_Owaenf
   No_of_bedrooms  square_footage  sale_price
0               3            2000      250000
1               5            3000      450000
2               2            1000      180000
3               6            4000      550000
The average sale price of houses with more than four bedrooms is: 500000.0

Process finished with exit code 0
```

4. **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:

5. **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}%")
```

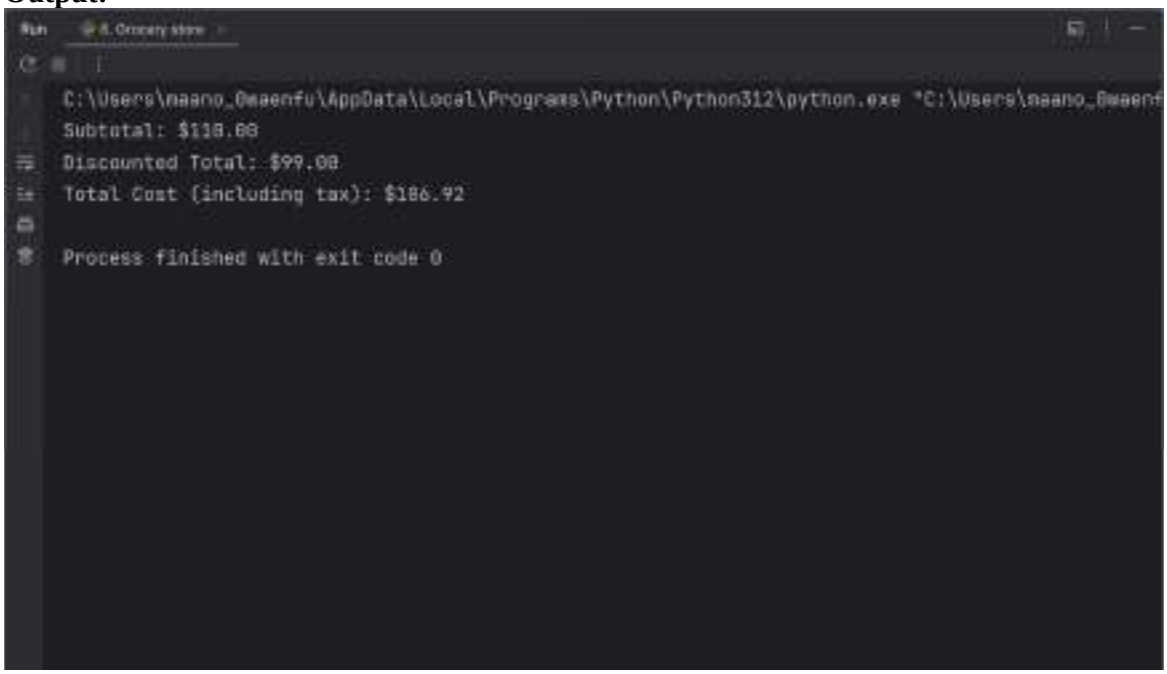
Output:

6. **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:

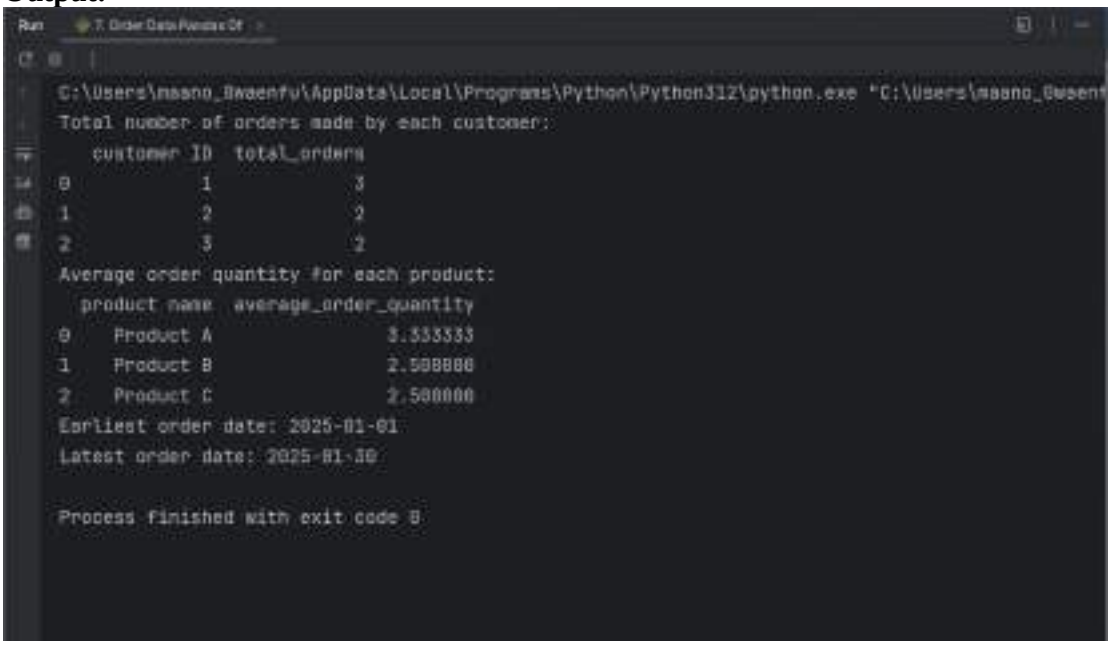


7. **Question:** Using Pandas DataFrame operations, how would you find the following information from the `order_data` DataFrame:
    1. The total number of orders made by each customer.
    2. The average order quantity for each product.
    3. The earliest and latest order dates in the dataset.

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:

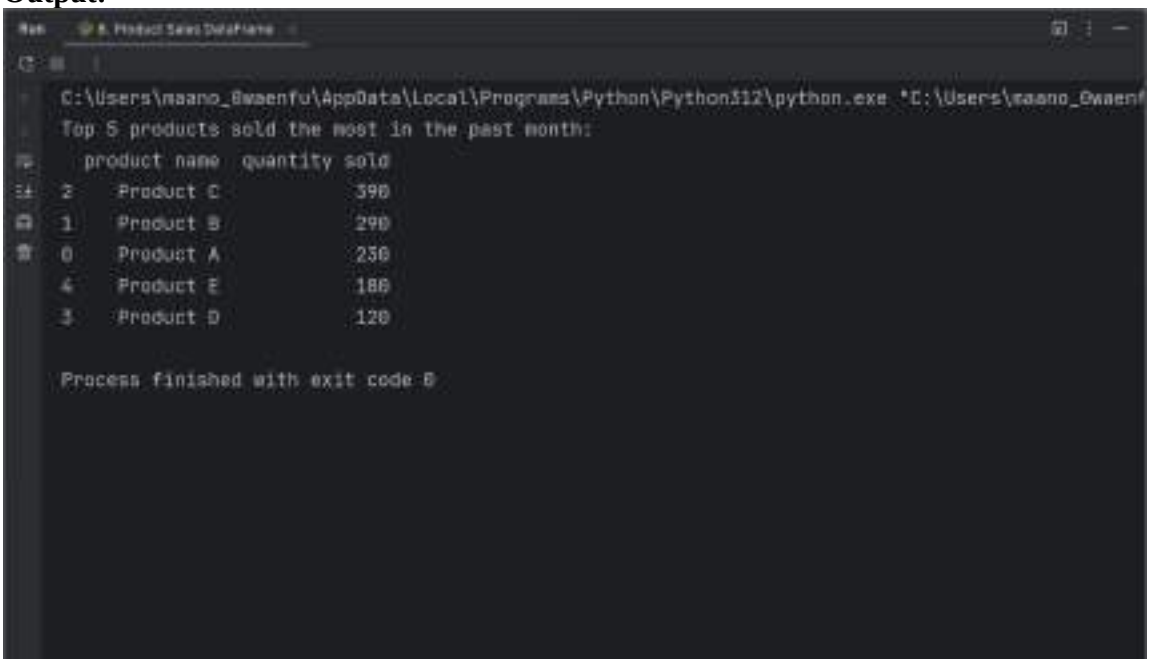


8. **Question:** How would you find the top 5 products that have been sold the most in the past month?

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:



```
C:\Users\maano_Owaenfu\AppData\Local\Programs\Python\Python312\python.exe "C:\Users\maano_Owaenf
Top 5 products sold the most in the past month:
  product name  quantity sold
2    Product C            390
1    Product B            290
0    Product A            230
4    Product E            180
3    Product D            120

Process finished with exit code 0
```

9. **Question:** Using Pandas DataFrame operations, how would you find the following information from the `property_data` DataFrame:
    1. The average listing price of properties in each location.
    2. The number of properties with more than four bedrooms.
    3. 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:

```
C:\Users\msano_Owaenfu\AppData\Local\Programs\Python\Python312\python.exe "C:\Users\msano_Owaenf
Average listing price per location:
       location     Price
0  Location A  275000.0
1  Location B  425000.0
2  Location C  350000.0

Number of properties with more than four bedrooms: 2

Property with the largest area:
   property ID    location  ...  area in square feet  listing price
4            5  Location B  ...                 2500         450000

[1 rows x 5 columns]

Process finished with exit code 0
```

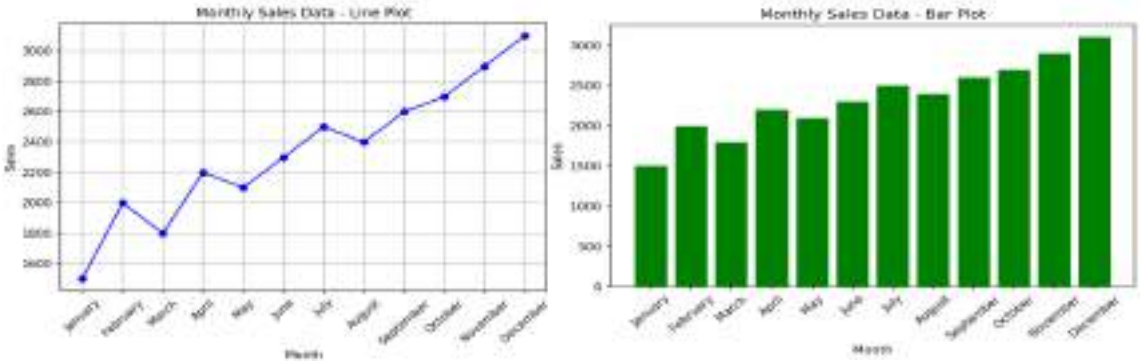
10. Question:

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

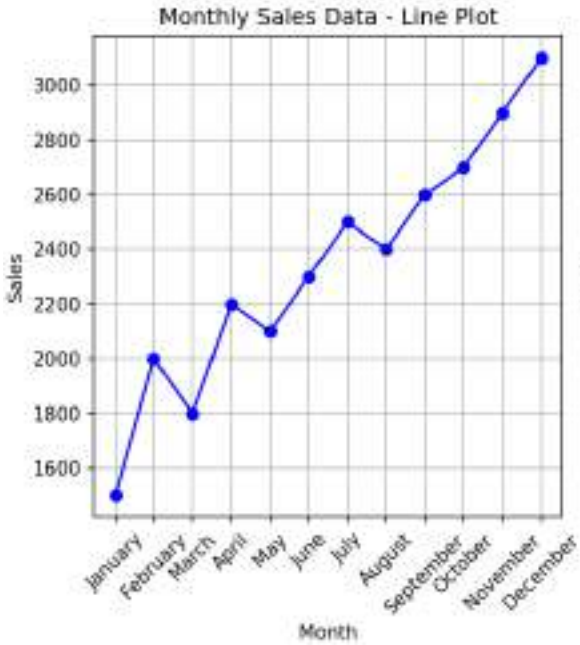


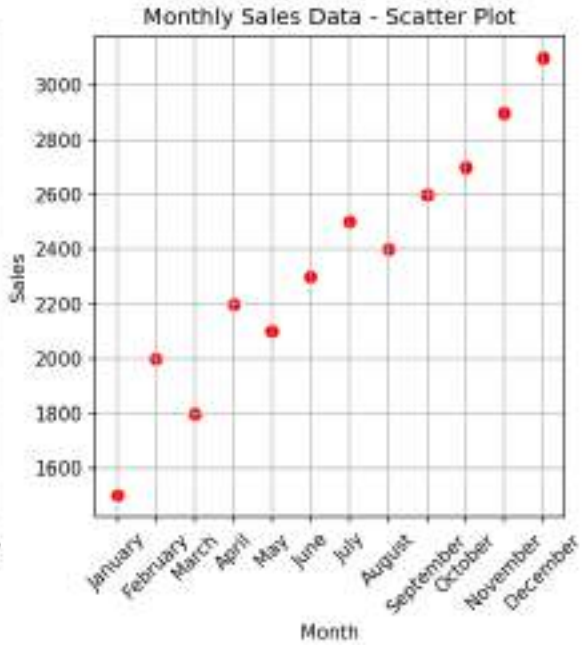
11. **Question:**
    1. Write code to create a simple line plot in Python using Matplotlib to predict sales happened in a month?
    2. Write code to create a scatter plot in Python using Matplotlib to predict sales happened in a month?
    3. Develop a Python program to create a bar plot of the monthly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

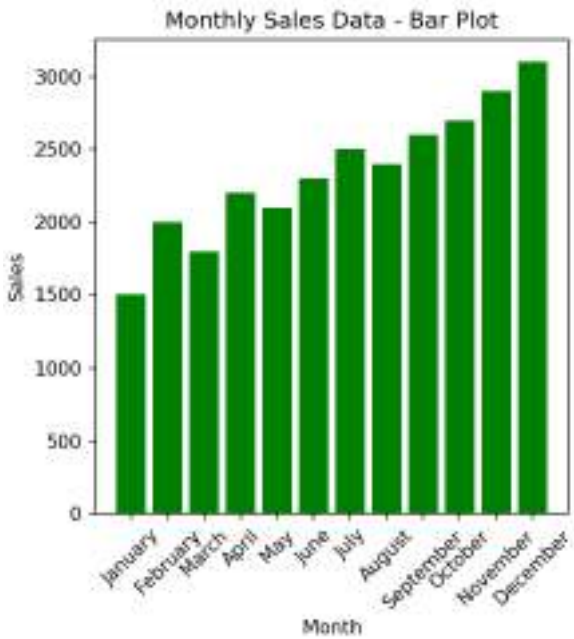
Code:

```
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:


Monthly Sales Data - Line Plot
Sales
Month
3000
2800
2600
2400
2200
2000
1800
1600
January
February
March
April
May
June
July
August
September
October
November
December


Monthly Sales Data - Scatter Plot
Sales
Month
3000
2800
2600
2400
2200
2000
1800
1600
January
February
March
April
May
June
July
August
September
October
November
December


Monthly Sales Data - Bar Plot
Sales
Month
3000
2500
2000
1500
1000
500
0
January
February
March
April
May
June
July
August
September
October
November
December

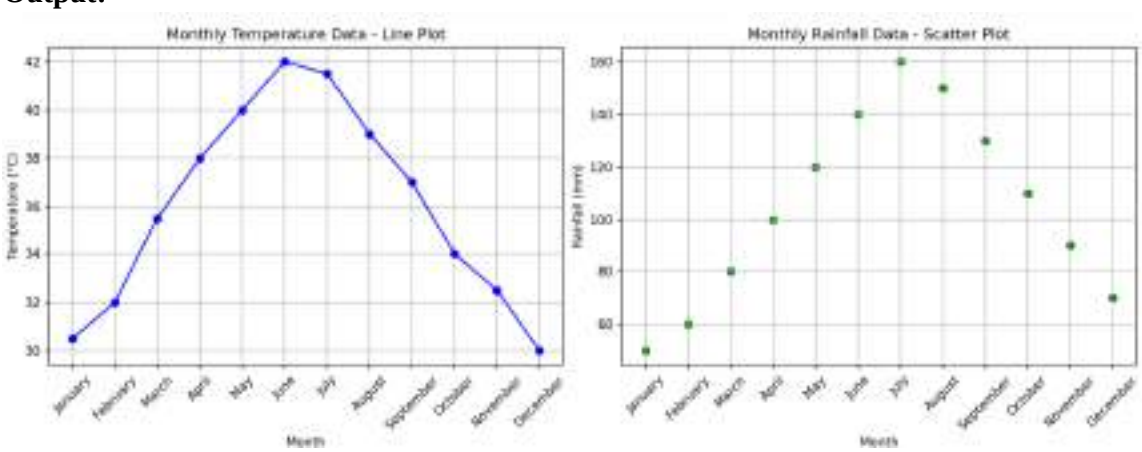
12. Question:

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:



13. **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:

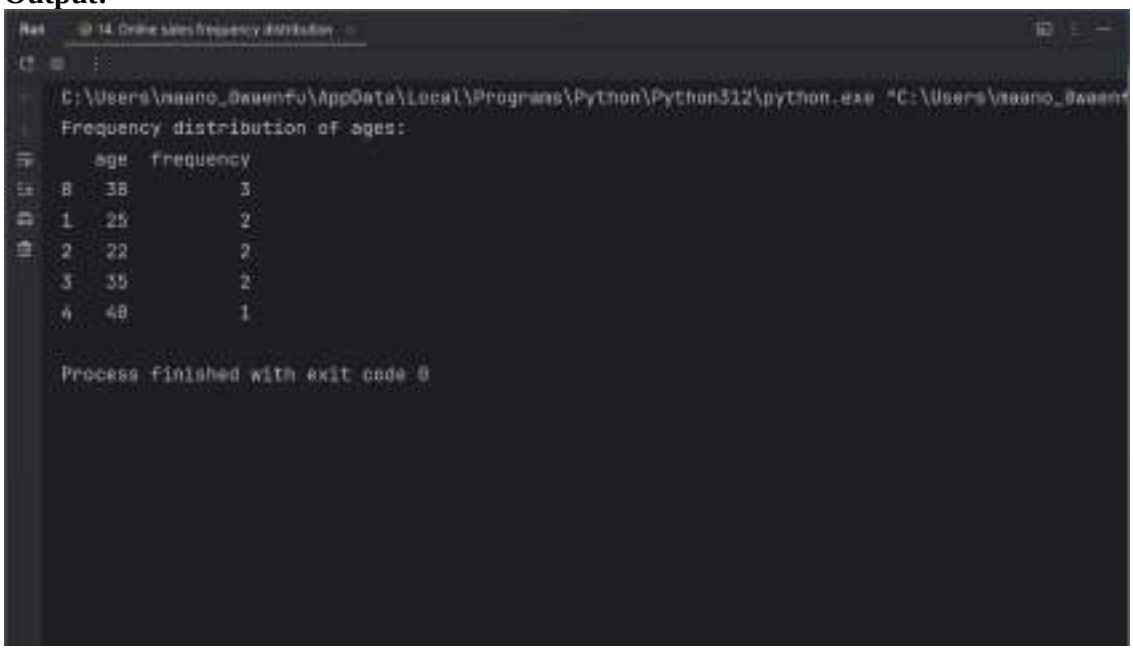
```
Run   13. Text document frequency check
C:\Users\naano_Owaenfu\AppData\Local\Programs\Python\Python312\python.exe "C:\Users\naano_Owaenf
the: 13
quick: 1
brown: 1
fox: 3
jumps: 1
over: 1
lazy: 2
dog: 3
is: 1
swift: 1
and: 4
cunning: 1
always: 1
finding: 1
a: 4
way: 1
to: 4
```

14. **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:

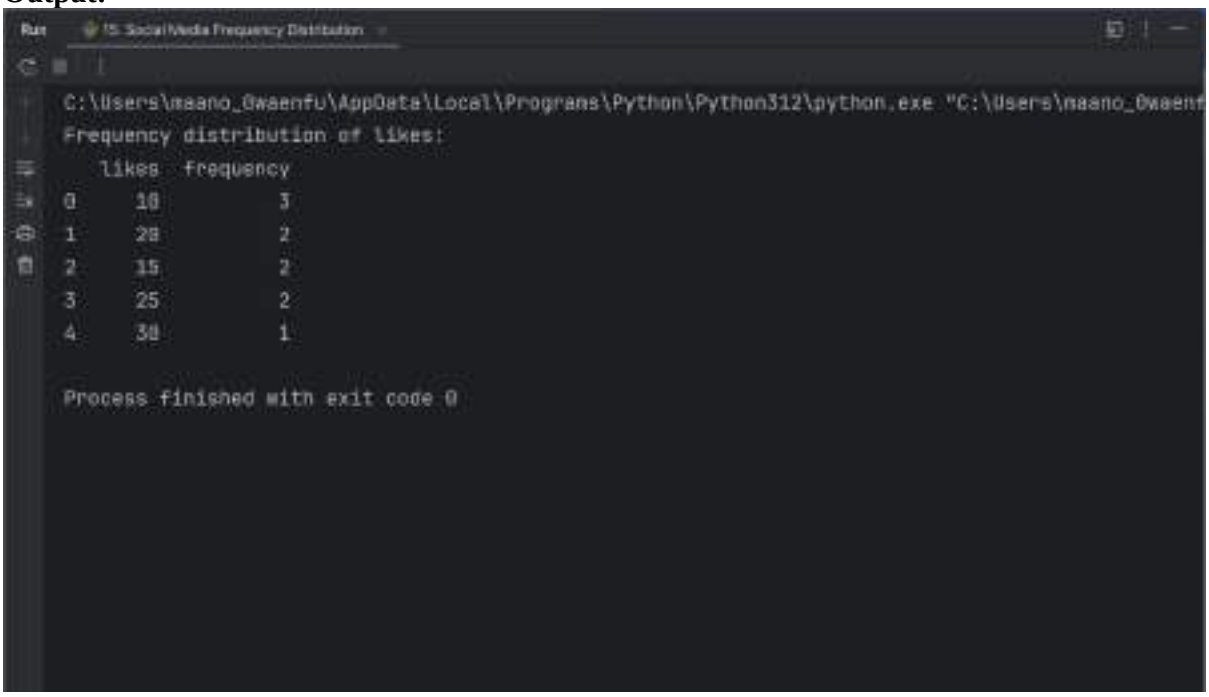


15. **Question:** Develop a Python program to calculate the frequency distribution of likes among the posts?

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:



16. **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}')
```

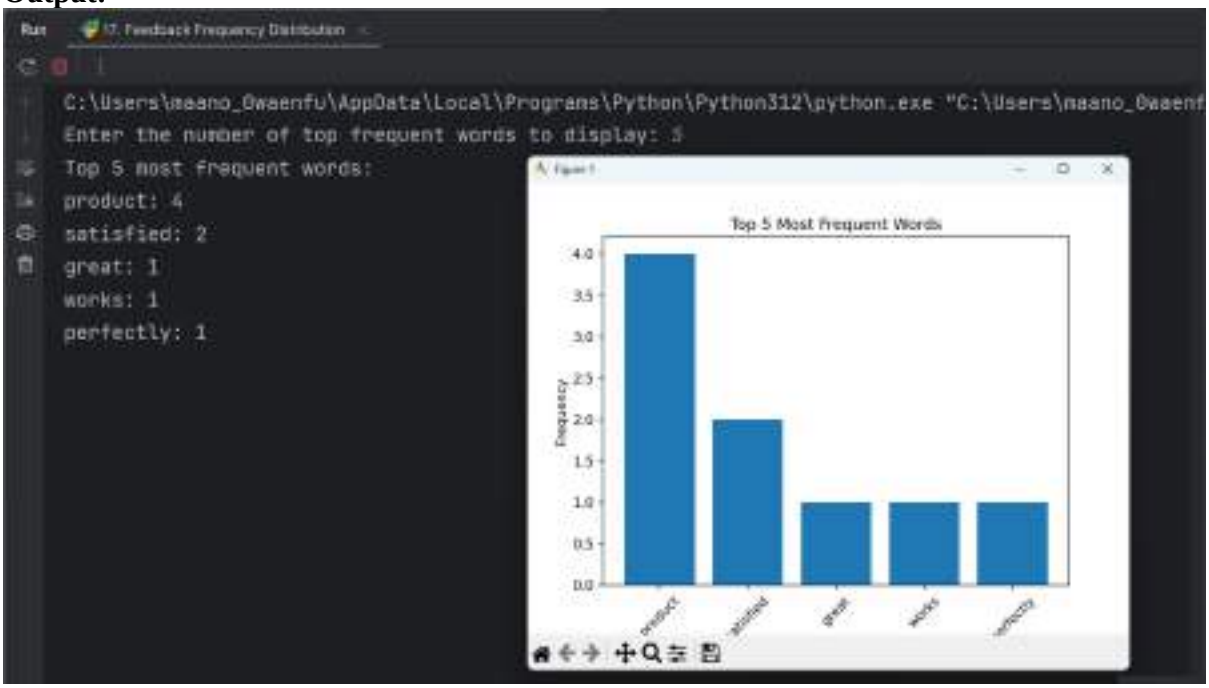
Output:

17. **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 pd
import 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:


Run
17. Feedback Frequency Distribution
C:\Users\maano_Owaenfu\AppData\Local\Programs\Python\Python312\python.exe "C:\Users\maano_Owaenf
Enter the number of top frequent words to display: 5
Top 5 most frequent words:
product: 4
satisfied: 2
great: 1
works: 1
perfectly: 1
Figure 1
Top 5 Most Frequent Words
Frequency
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
product
satisfied
great
works
perfectly

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

age	23	23	27	27	39	41	47	49	50
%fat	9.5	26.5	7.8	17.8	31.4	25.9	27.4	27.2	31.2
age	52	54	54	56	57	58	58	60	61
%fat	34.6	42.5	28.8	33.4	30.2	34.1	32.9	41.2	35.7

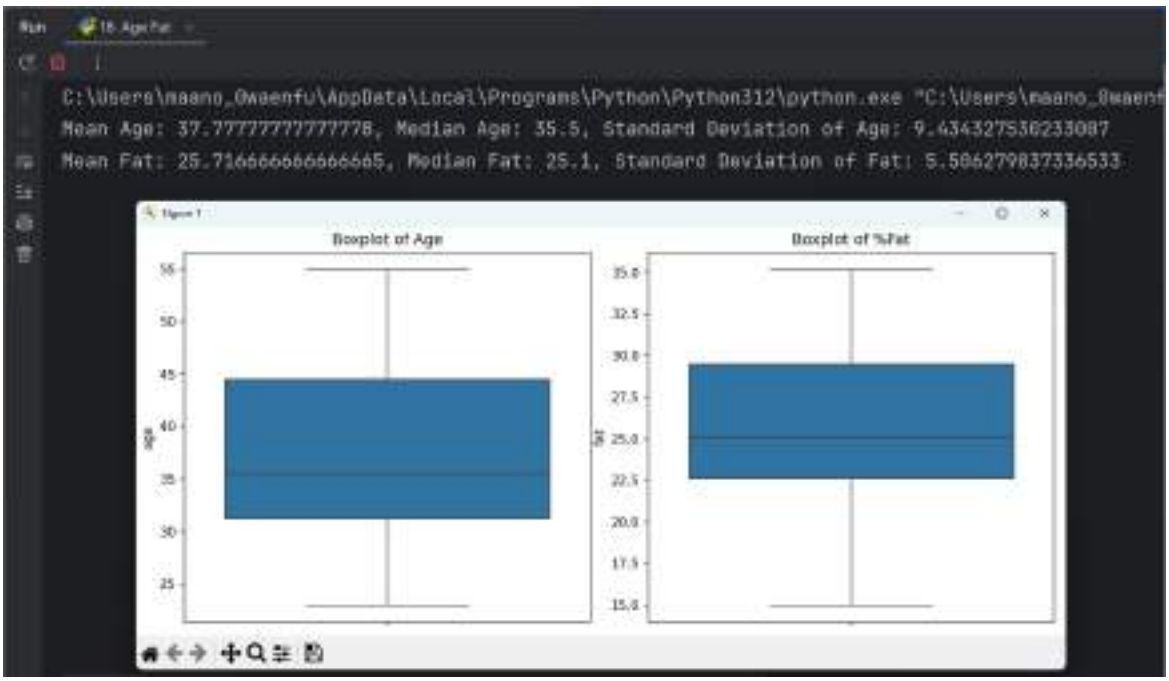
- Calculate the mean, median and standard deviation of age and %fat using Pandas.
- Draw the boxplots for age and %fat.
- Draw a scatter plot and a q-q plot based on these two variables.

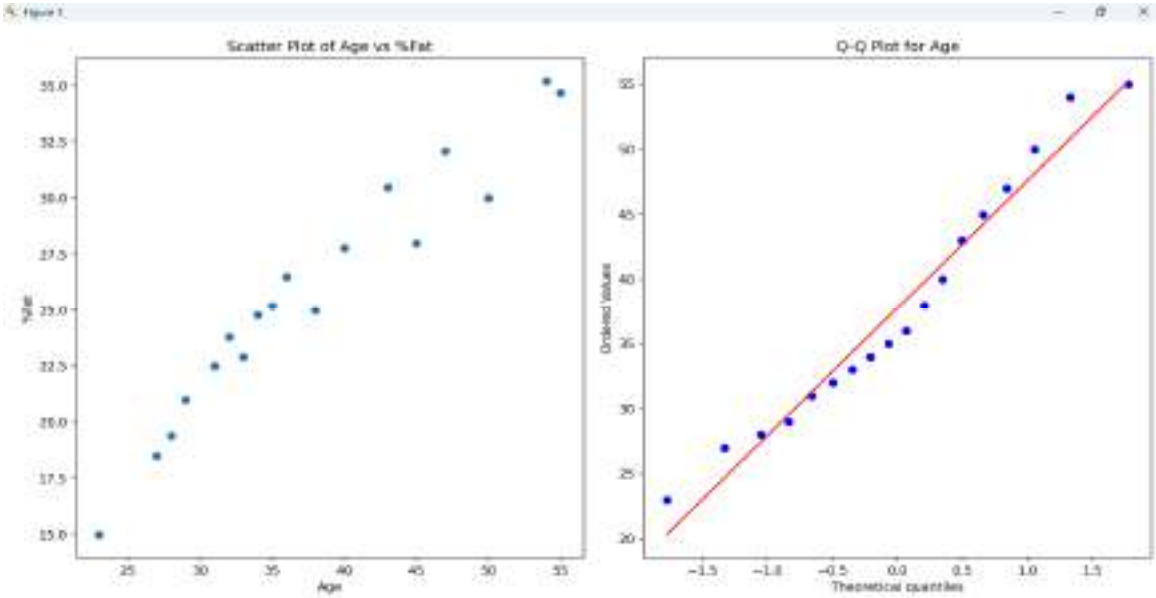
Code:

```
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:





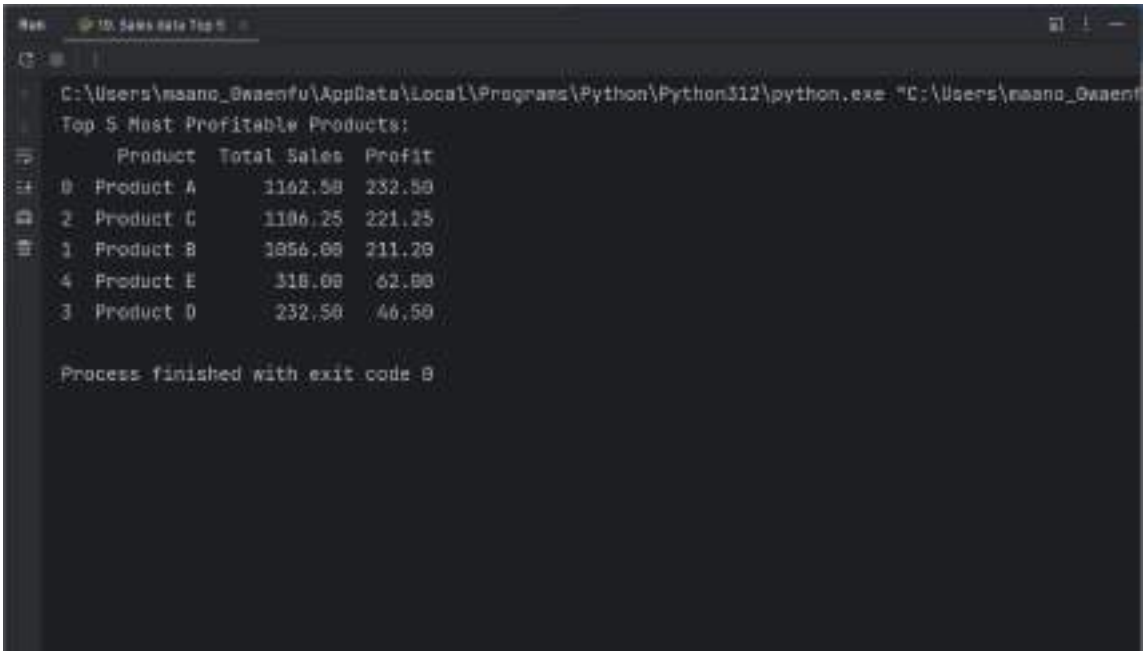
19. Sales and Profit Analysis:

a) Load the "sales_data.csv" file into a Pandas data frame, which contains columns "Date," "Product," "Quantity Sold," and "Unit Price."
b) Create a new column named "Total Sales" that calculates the total sales for each transaction (Quantity Sold * Unit Price).
c) 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:



```
C:\Users\maano_Owaenfu\AppData\Local\Programs\Python\Python312\python.exe "C:\Users\maano_Owaenf
Top 5 Most Profitable Products:
      Product  Total Sales  Profit
0  Product A      1162.50  232.50
2  Product C      1106.25  221.25
1  Product B      1056.00  211.20
4  Product E       310.00   62.00
3  Product D       232.50   46.50

Process finished with exit code 0
```

20. Customer Segmentation:

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

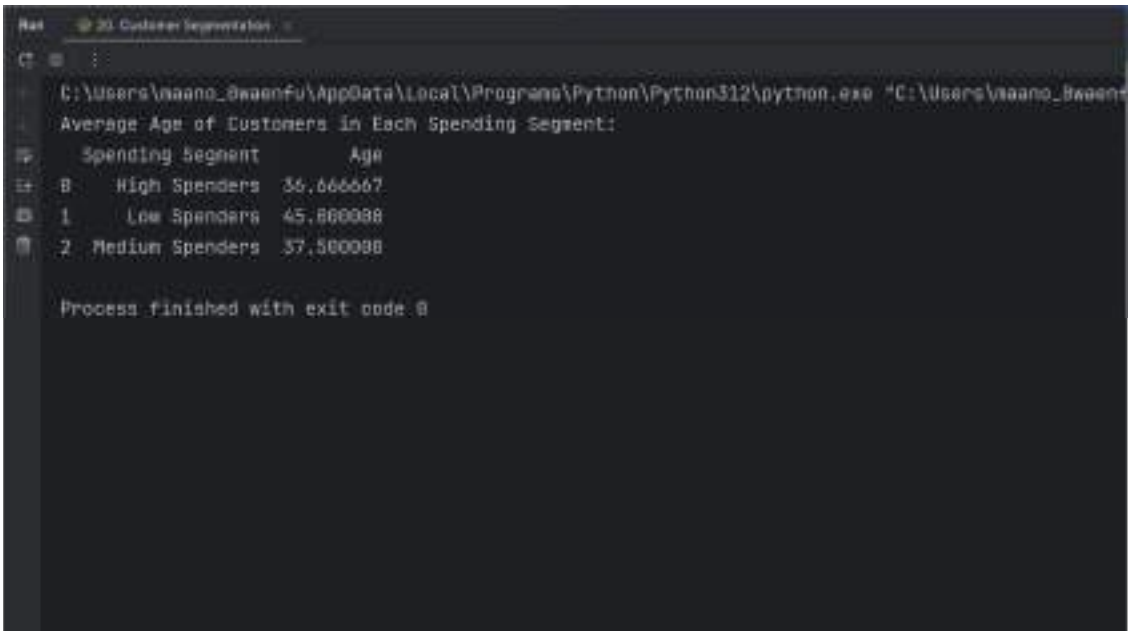
b) Segment customers into three groups based on their total spending: "High Spenders," "Medium Spenders," and "Low Spenders." Assign these segments to a new column in the data frame.

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:



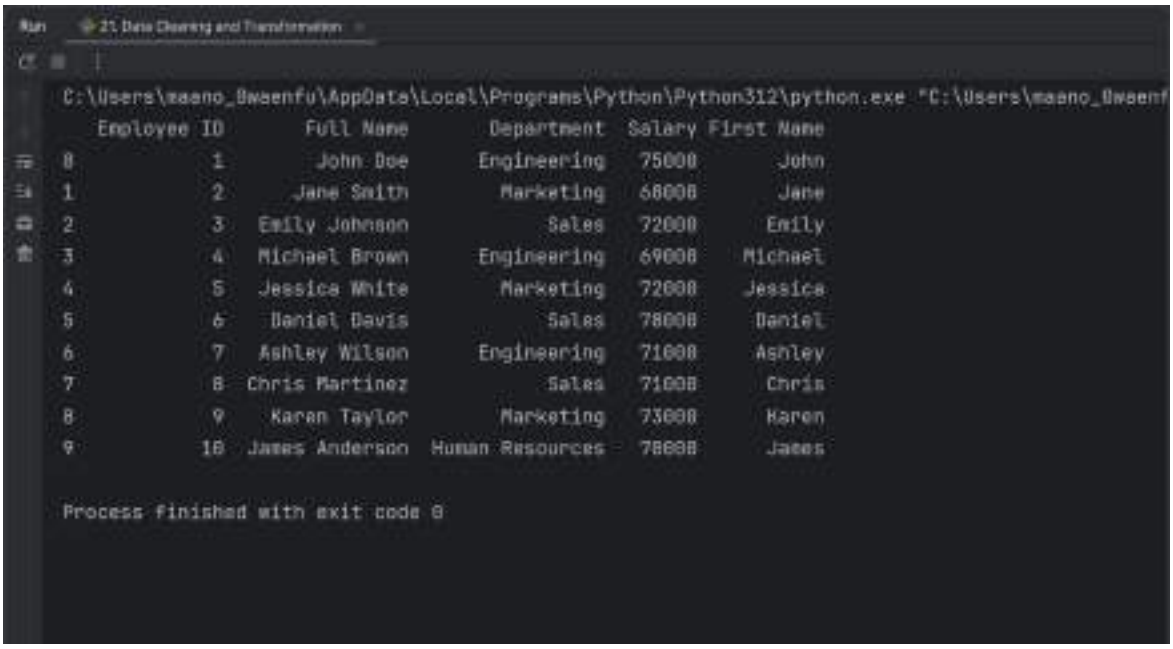
21. Data Cleaning and Transformation:

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:



```
C:\Users\maano_Owaenfu\AppData\Local\Programs\Python\Python312\python.exe "C:\Users\maano_Owaenf
   Employee ID       Full Name       Department  Salary First Name
0            1        John Doe      Engineering   75000       John
1            2      Jane Smith        Marketing   68000       Jane
2            3   Emily Johnson            Sales   72000      Emily
3            4   Michael Brown      Engineering   69000    Michael
4            5   Jessica White        Marketing   72000    Jessica
5            6    Daniel Davis            Sales   78000     Daniel
6            7   Ashley Wilson      Engineering   71000     Ashley
7            8  Chris Martinez            Sales   71000      Chris
8            9    Karen Taylor        Marketing   73000      Karen
9           10  James Anderson  Human Resources   78000      James

Process finished with exit code 0
```

22. 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.

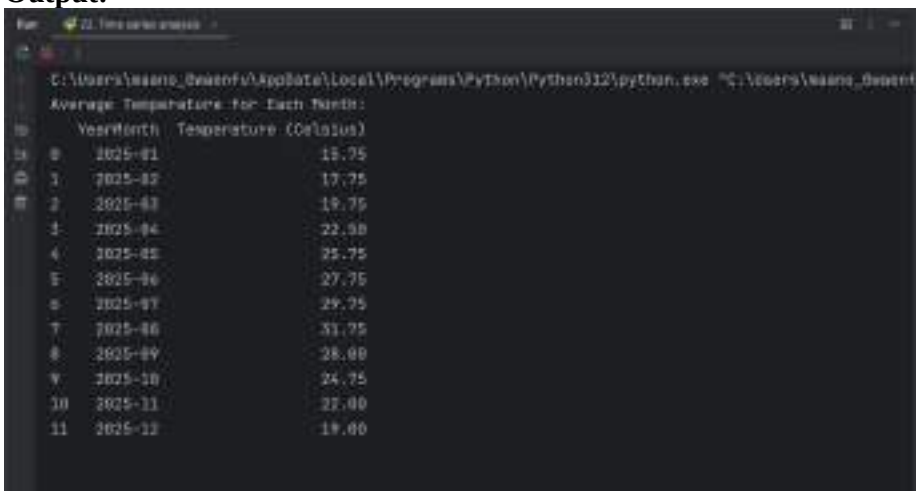
c) Calculate the average temperature for each month and display the results in chronological order.

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:



```
C:\Users\maans_0waanfv\AppData\Local\Programs\Python\Python312\python.exe "C:\Users\maans_0waanf
Average Temperature for Each Month:
   YearMonth  Temperature (Celsius)
0    2025-01                 15.75
1    2025-02                 17.75
2    2025-03                 19.75
3    2025-04                 22.50
4    2025-05                 25.75
5    2025-06                 27.75
6    2025-07                 29.75
7    2025-08                 31.75
8    2025-09                 28.00
9    2025-10                 24.75
10   2025-11                 22.00
11   2025-12                 19.00
```

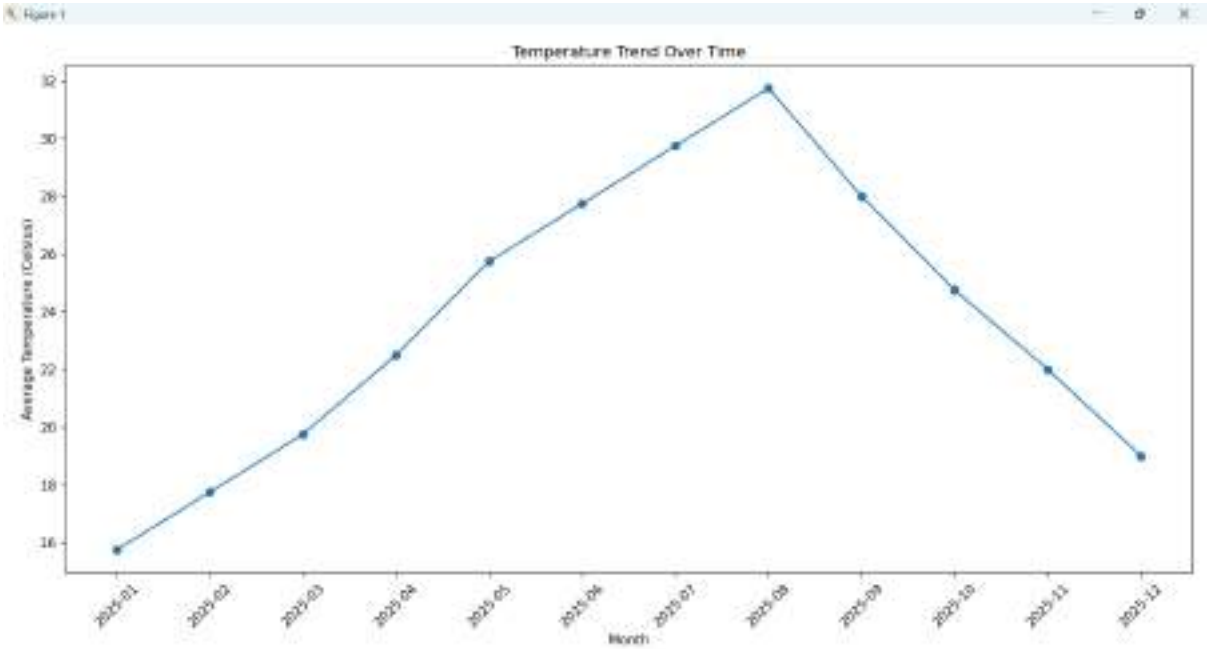


Figure 1
Temperature Trend Over Time
Average Temperature (Celsius)
32
30
28
26
24
22
20
18
16
2025-01
2025-02
2025-03
2025-04
2025-05
2025-06
2025-07
2025-08
2025-09
2025-10
2025-11
2025-12
Month

23. 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."

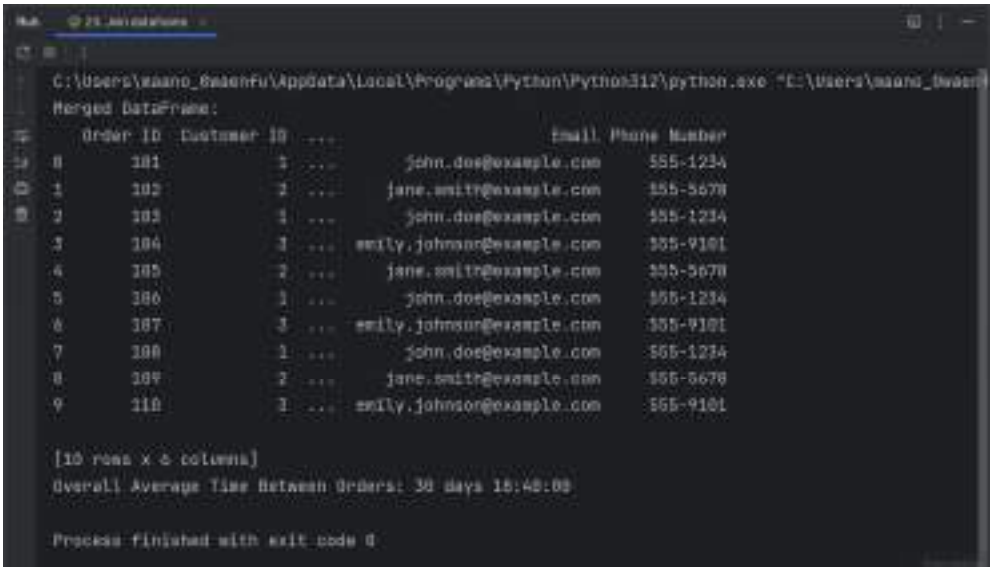
c) Merge the two data frames based on the "Customer ID" column to create a new data frame that includes both order information and customer details.

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:



```
C:\Users\maano_Owaenfu\AppData\Local\Programs\Python\Python312\python.exe "C:\Users\maano_Owaen
Merged DataFrame:
   Order ID  Customer ID ...                 Email Phone Number
0       101            1 ...       john.doe@example.com     555-1234
1       102            2 ...     jane.smith@example.com     555-5678
2       103            1 ...       john.doe@example.com     555-1234
3       104            3 ... emily.johnson@example.com     555-9101
4       105            2 ...     jane.smith@example.com     555-5678
5       106            1 ...       john.doe@example.com     555-1234
6       107            3 ... emily.johnson@example.com     555-9101
7       108            1 ...       john.doe@example.com     555-1234
8       109            2 ...     jane.smith@example.com     555-5678
9       110            3 ... emily.johnson@example.com     555-9101

[10 rows x 6 columns]
Overall Average Time Between Orders: 30 days 18:48:00

Process finished with exit code 0
```