Query Processing For Data Science With Fuzzy Matching \_- DSA0512

Experiment 1 :

Aim: To write a Pandas program to select distinct department id from employees file.

Code:

import pandas as pd

# Load the employees data from a CSV file

employees = pd.read\_csv('Employee.1csv.csv')

# Select distinct department IDs

distinct\_departments = employees['Department\_ID'].unique()

# Print the distinct department IDs

print(distinct\_departments)

Sample Input : employeejob2.csv

+---------------+----------------------+------------+-------------+

| DEPARTMENT\_ID | DEPARTMENT\_NAME | MANAGER\_ID | LOCATION\_ID |

+---------------+----------------------+------------+-------------+

| 10 | Administration | 200 | 1700 |

| 20 | Marketing | 201 | 1800 |

| 30 | Purchasing | 114 | 1700 |

| 40 | Human Resources | 203 | 2400 |

| 50 | Shipping | 121 | 1500 |

| 60 | IT | 103 | 1400 |

| 70 | Public Relations | 204 | 2700 |

| 80 | Sales | 145 | 2500 |

| 90 | Executive | 100 | 1700 |

| 100 | Finance | 108 | 1700 |

| 110 | Accounting | 205 | 1700 |

| 120 | Treasury | 0 | 1700 |

| 130 | Corporate Tax | 0 | 1700 |

| 140 | Control And Credit | 0 | 1700 |

| 150 | Shareholder Services | 0 | 1700 |

| 160 | Benefits | 0 | 1700 |

| 170 | Manufacturing | 0 | 1700 |

| 180 | Construction | 0 | 1700 |

| 190 | Contracting | 0 | 1700 |

| 200 | Operations | 0 | 1700 |

| 210 | IT Support | 0 | 1700 |

| 220 | NOC | 0 | 1700 |

| 230 | IT Helpdesk | 0 | 1700 |

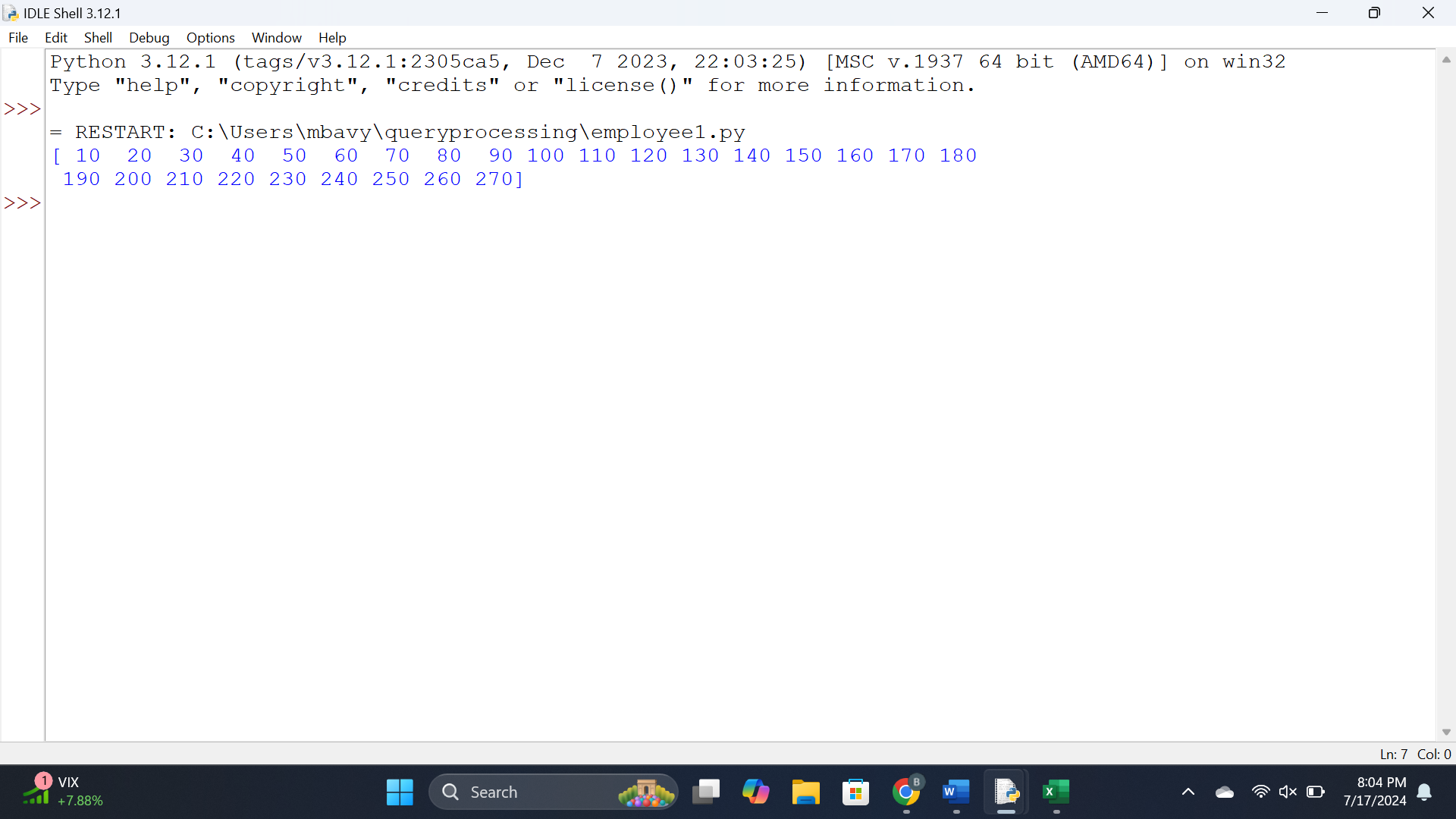
| 240 | Government Sales | 0 | 1700 |

| 250 | Retail Sales | 0 | 1700 |

| 260 | Recruiting | 0 | 1700 |

| 270 | Payroll | 0 | 1700 |

Sample Output:



Results :

Thus a pandas program to select distinct department id from employees file is done.

Experiment 2

Aim: To Write a Pandas program to display the ID for those employees who did two or more jobs in the past.

Code:

import pandas as pd

# Create DataFrame

df = pd.read\_csv('employeejob2.csv')

# Group by EMPLOYEE\_ID and count the number of jobs each employee has hadjob\_counts = df.groupby("Employee\_id").size()

# Filter employees who did two or more jobs

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

rint(employees\_multiple\_jobs.values)

Sample Input :

| EMPLOYEE\_ID | START\_DATE | END\_DATE | JOB\_ID | DEPARTMENT\_ID |

+-------------+------------+------------+------------+---------------+

| 102 | 2001-01-13 | 2006-07-24 | IT\_PROG | 60 |

| 101 | 1997-09-21 | 2001-10-27 | AC\_ACCOUNT | 110 |

| 101 | 2001-10-28 | 2005-03-15 | AC\_MGR | 110 |

| 201 | 2004-02-17 | 2007-12-19 | MK\_REP | 20 |

| 114 | 2006-03-24 | 2007-12-31 | ST\_CLERK | 50 |

| 122 | 2007-01-01 | 2007-12-31 | ST\_CLERK | 50 |

| 200 | 1995-09-17 | 2001-06-17 | AD\_ASST | 90 |

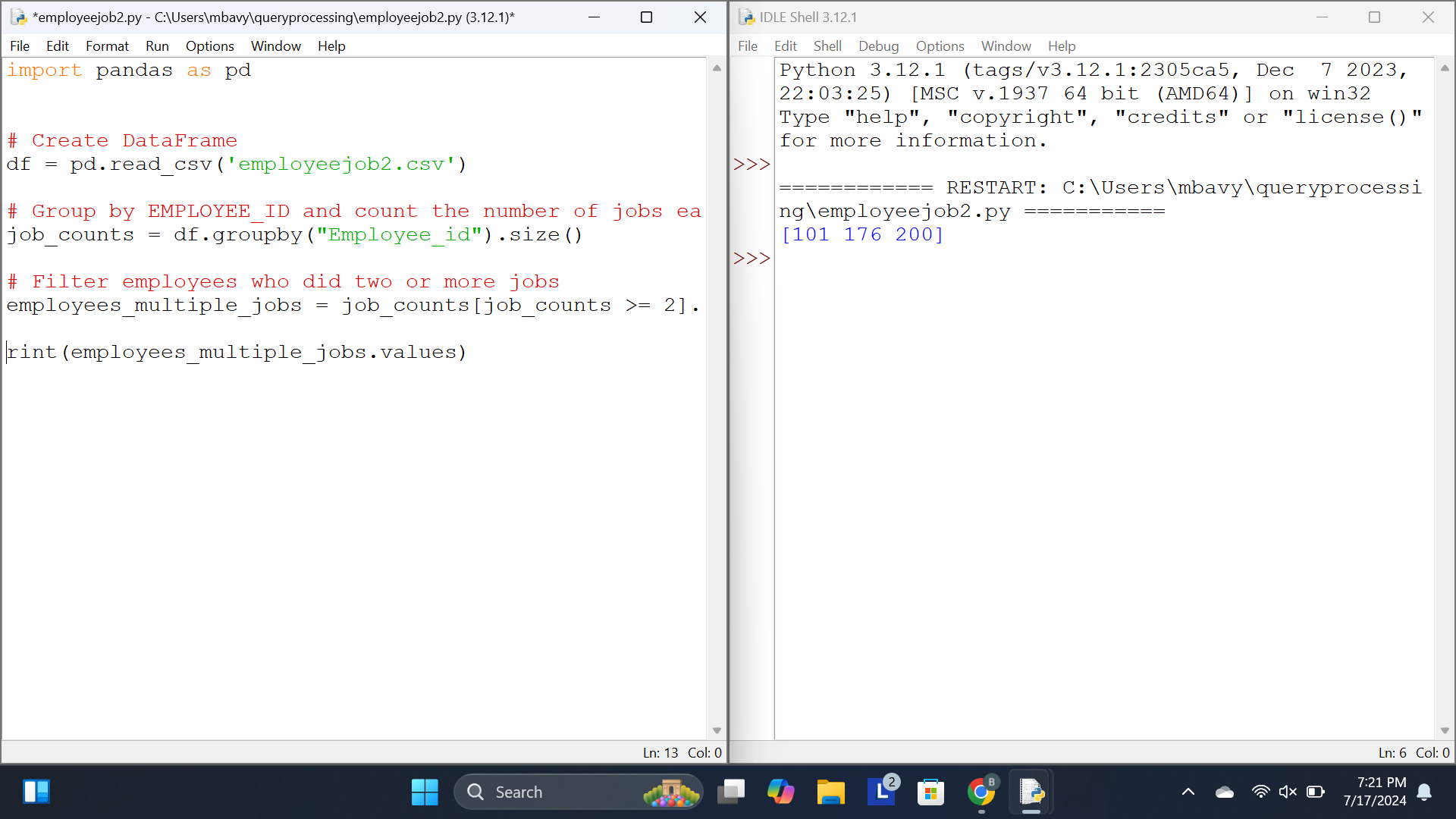
| 176 | 2006-03-24 | 2006-12-31 | SA\_REP | 80 |

| 176 | 2007-01-01 | 2007-12-31 | SA\_MAN | 80 |

| 200 | 2002-07-01 | 2006-12-31 | AC\_ACCOUNT | 90 |

+-------------+------------+------------+------------+---------------+

Sample Output:



Results: Thus a Pandas program to display the ID for those employees who did two or more jobs in the past is done.

Experiment 3:

Aim : To Write a Pandas program to display the details of jobs in descending sequence on job title.

Code :

import pandas as pd

# Load the job data from a CSV file

jobs = pd.read\_csv('Job\_title.csv')

# Sort the jobs in descending order based on job title

sorted\_jobs = jobs.sort\_values(by='JOB\_TITLE', ascending=False)

# Display the sorted job details

print(sorted\_jobs)

Sample Input :

+------------+---------------------------------+------------+------------+

| JOB\_ID | JOB\_TITLE | MIN\_SALARY | MAX\_SALARY |

+------------+---------------------------------+------------+------------+

| AD\_PRES | President | 20080 | 40000 |

| AD\_VP | Administration Vice President | 15000 | 30000 |

| AD\_ASST | Administration Assistant | 3000 | 6000 |

| FI\_MGR | Finance Manager | 8200 | 16000 |

| FI\_ACCOUNT | Accountant | 4200 | 9000 |

| AC\_MGR | Accounting Manager | 8200 | 16000 |

| AC\_ACCOUNT | Public Accountant | 4200 | 9000 |

| SA\_MAN | Sales Manager | 10000 | 20080 |

| SA\_REP | Sales Representative | 6000 | 12008 |

| PU\_MAN | Purchasing Manager | 8000 | 15000 |

| PU\_CLERK | Purchasing Clerk | 2500 | 5500 |

| ST\_MAN | Stock Manager | 5500 | 8500 |

| ST\_CLERK | Stock Clerk | 2008 | 5000 |

| SH\_CLERK | Shipping Clerk | 2500 | 5500 |

| IT\_PROG | Programmer | 4000 | 10000 |

| MK\_MAN | Marketing Manager | 9000 | 15000 |

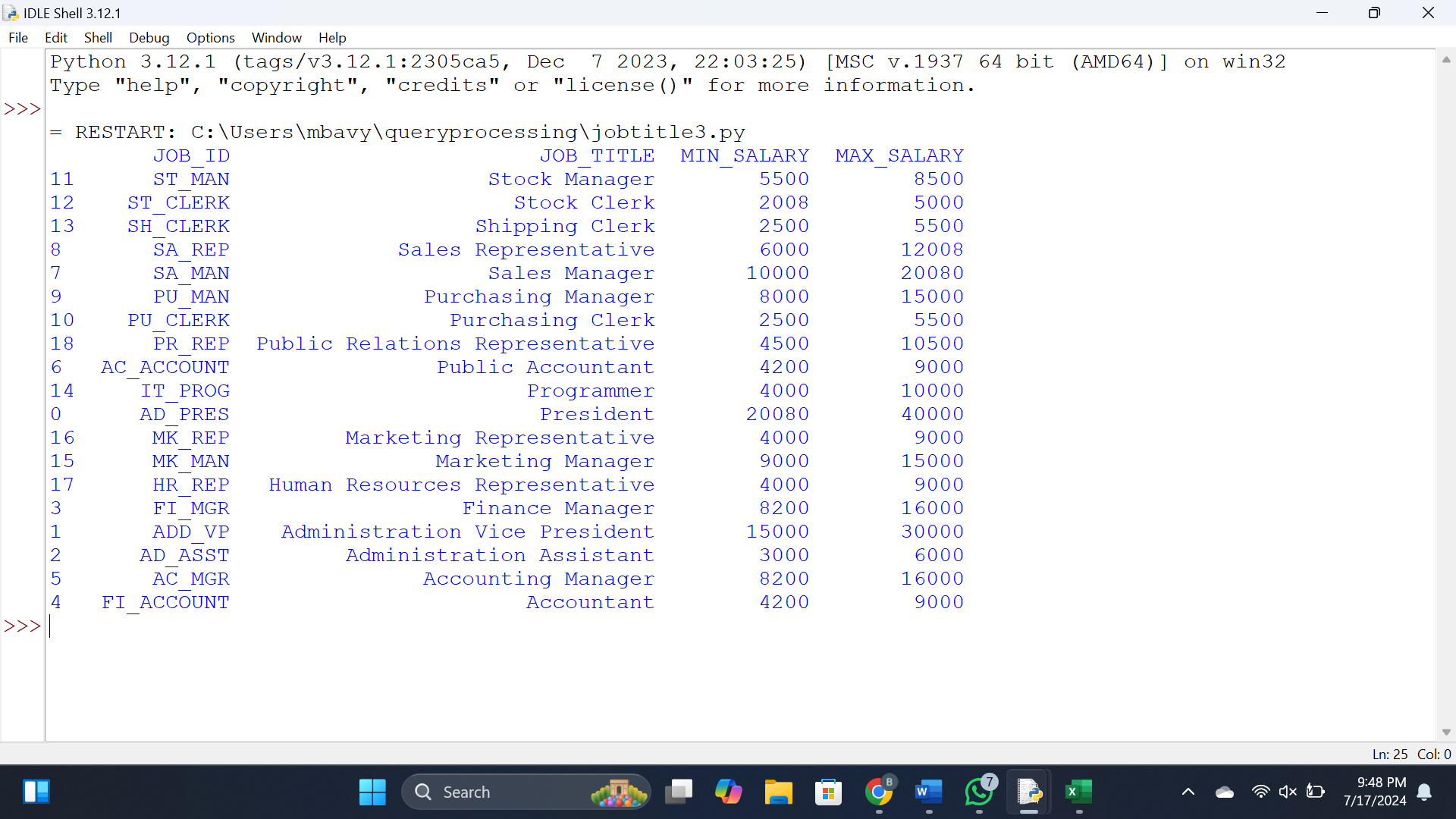
| MK\_REP | Marketing Representative | 4000 | 9000 |

| HR\_REP | Human Resources Representative | 4000 | 9000 |

| PR\_REP | Public Relations Representative | 4500 | 10500 |

+------------+---------------------------------+------------+----------

Sample Output:



Results:

Thus a Pandas program to display the details of jobs in descending sequence on job title is done.

Experiment 4:

Aim :To write a Pandas program to create a line plot of the historical stock prices of Alphabet Inc. between two specific dates.

Code:

import pandas as pd

import matplotlib.pyplot as plt

# Read the CSV file into a DataFrame

data = pd.read\_csv('Alphabet.csv')

# Define the start and end dates

start\_date ='01-04-2020'

end\_date = '15-04-2020'

# Filter the data between the specific dates

mask = (data['Date'] >= start\_date) & (data['Date'] <= end\_date)

filtered\_data = data.loc[mask]

# Set the 'Date' column as the index

filtered\_data.set\_index('Date', inplace=True)

# Plot the closing prices as a line plot

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

plt.plot(filtered\_data.index, filtered\_data['Close'], label='Close Price')

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

plt.xlabel('Date')

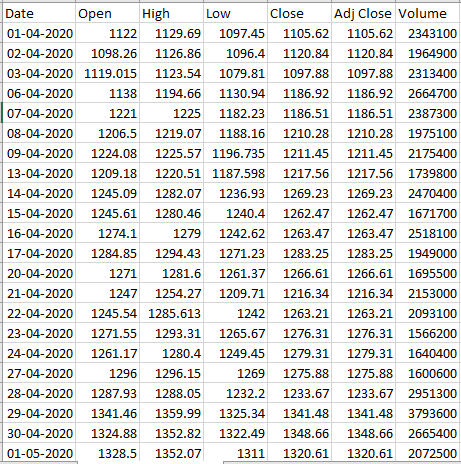
plt.ylabel('Stock Price ($)')

plt.legend()

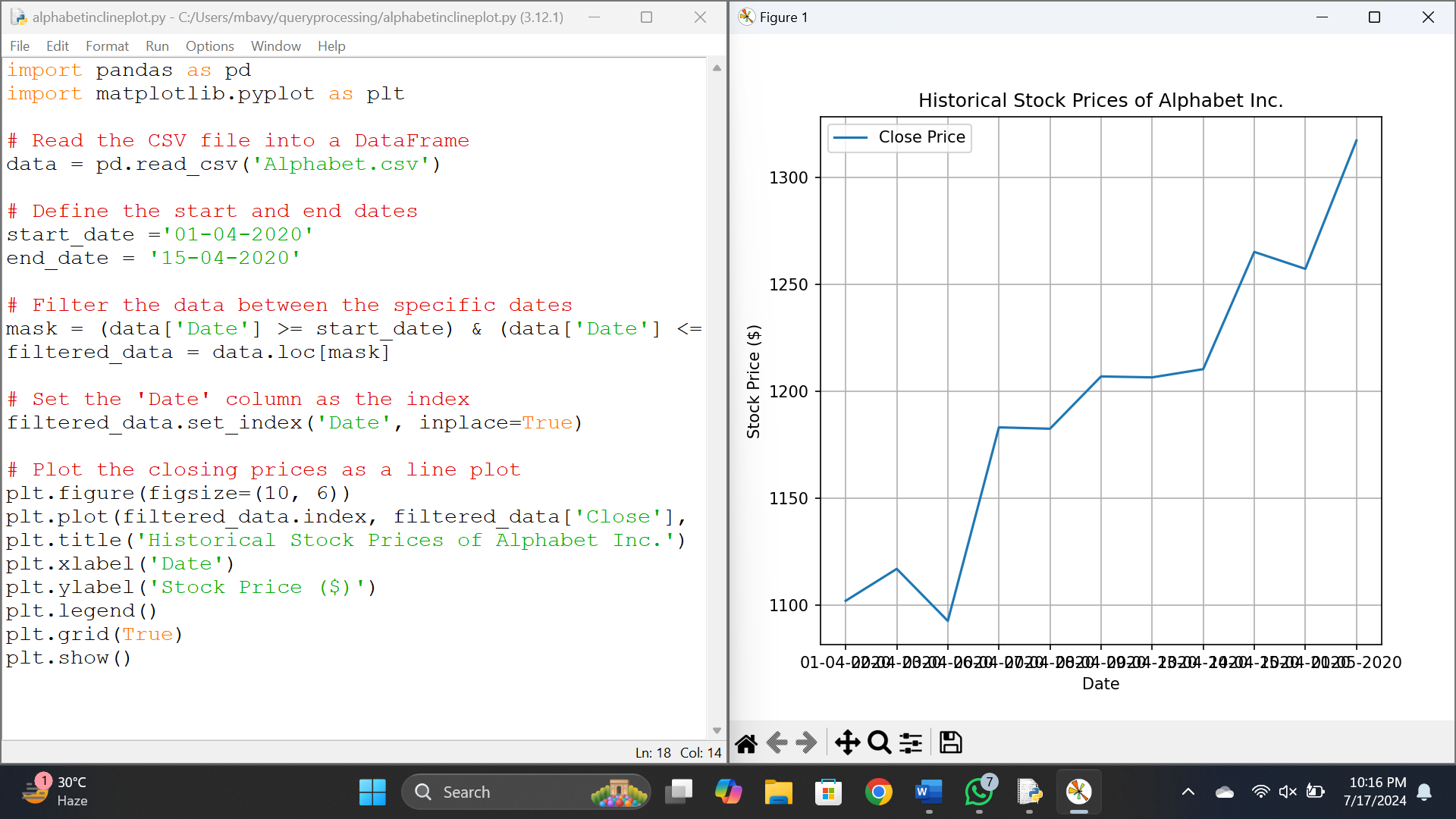
plt.grid(True)

plt.show()

Sample Input:



Sample Output:



Results:

Thus a Pandas program to create a line plot of the historical stock prices of Alphabet Inc. between two specific dates.

Experiment 5

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

Code:

import pandas as pd

import matplotlib.pyplot as plt

# Load the CSV file

data = pd.read\_csv('Alphabet.csv') # Update with your file path

# Define the date range

start\_date = '01-04-2020' # Update with your start date

end\_date = '15-04-2020' # Update with your end date

# Filter the data between the specified dates

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

# Plot the trading volume

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

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

plt.xlabel('Date')

plt.ylabel('Trading Volume')

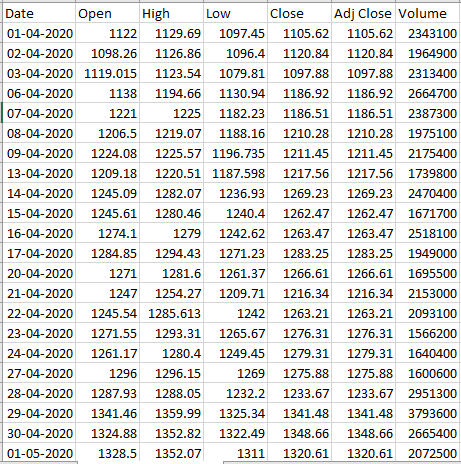
plt.title('Trading Volume of Alphabet Inc. Stock')

plt.xticks(rotation=45)

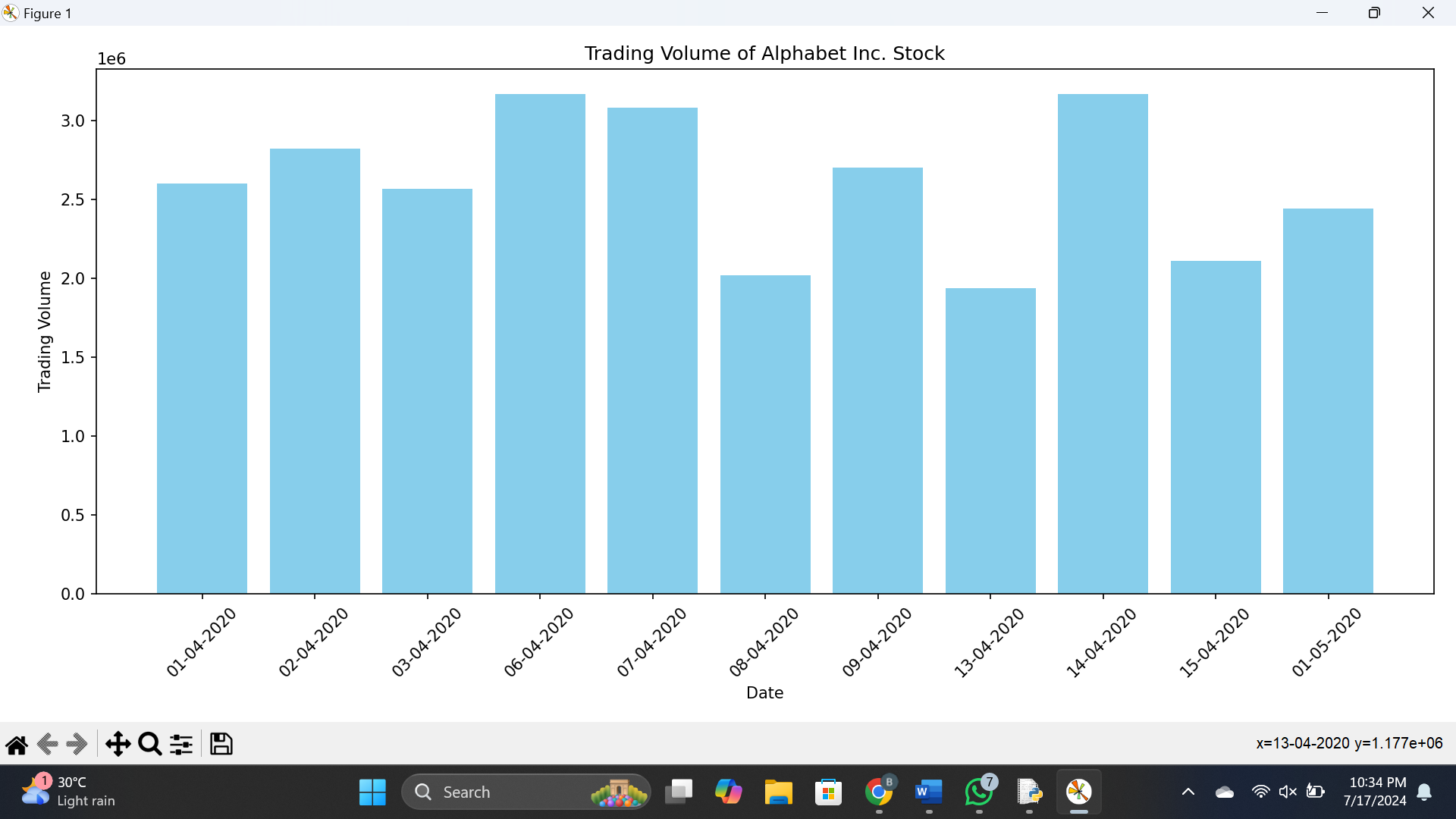
plt.tight\_layout()

plt.show()

Sample Input:



Sample Output :



Result :

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

Experiment 5

Aim: To Write a Pandas program to create a scatter plot of the trading volume/stock prices of Alphabet Inc. stock between two specific dates.

Code:

import pandas as pd

import matplotlib.pyplot as plt

# Load data from CSV file

data = pd.read\_csv('Alphabet.csv')

# Filter data between two specific dates

start\_date = '01-04-2020'

end\_date = '15-04-2020'

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

# Create scatter plot

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

plt.scatter(filtered\_data['Close'], filtered\_data['Volume'], marker='o', color='blue', alpha=0.7)

plt.title('Scatter Plot of Trading Volume vs Stock Prices')

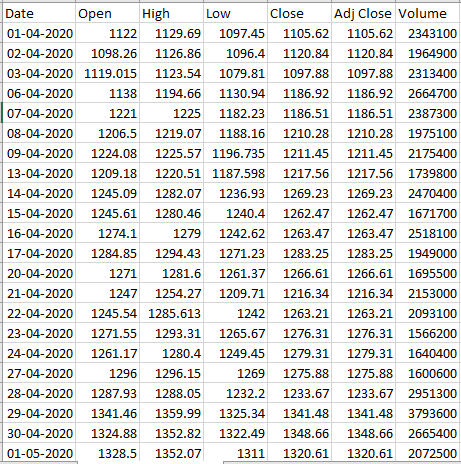
plt.xlabel('Stock Prices')

plt.ylabel('Trading Volume')

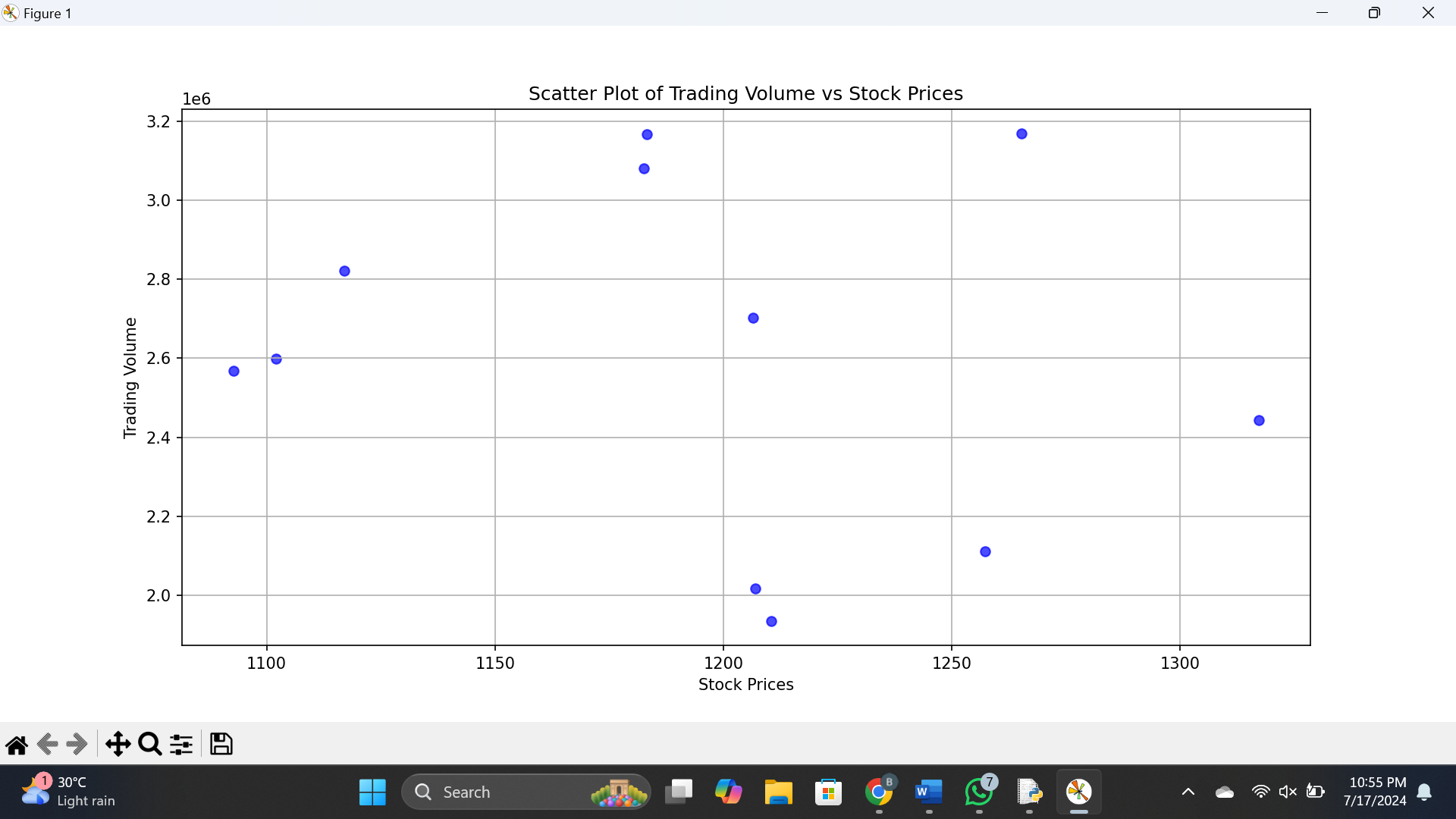
plt.grid(True)

plt.show()

Sample Input:



Sample Output:



Results:

Thus a Pandas program to create a scatter plot of the trading volume/stock prices of Alphabet Inc. stock between two specific dates.