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.

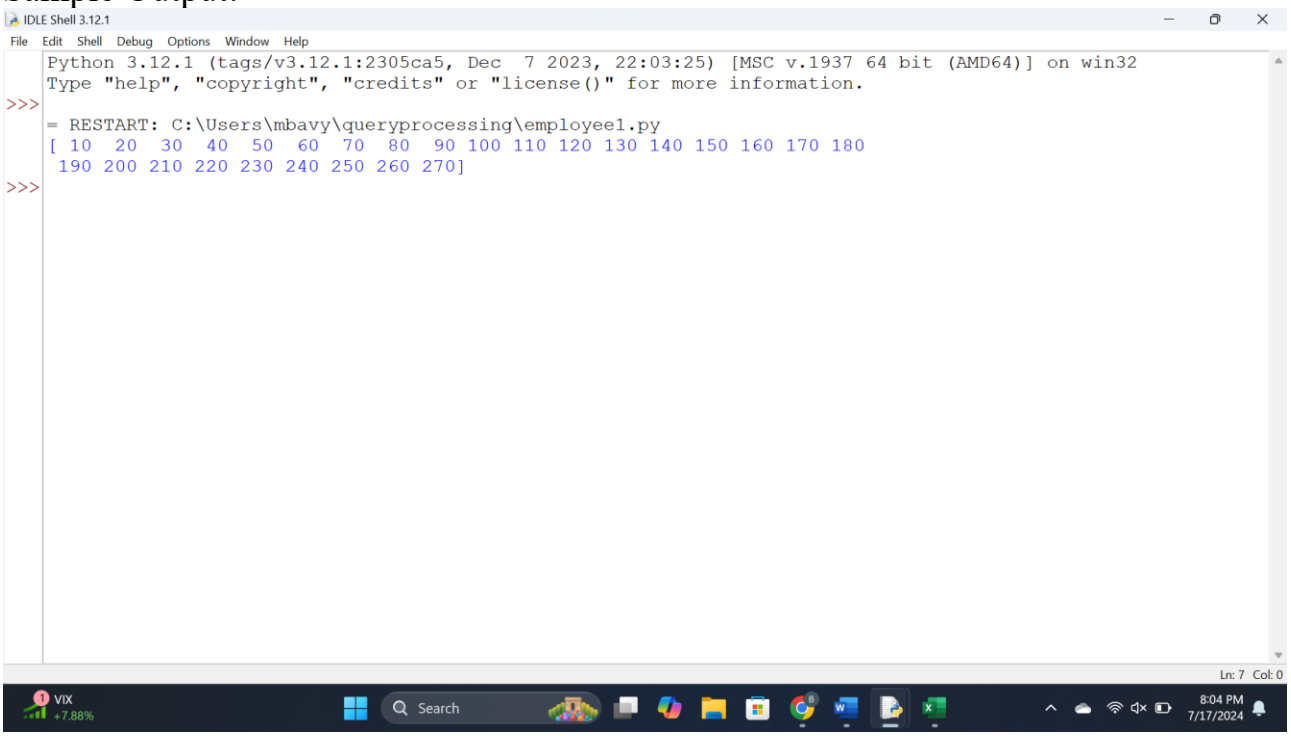
Code:

```
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.

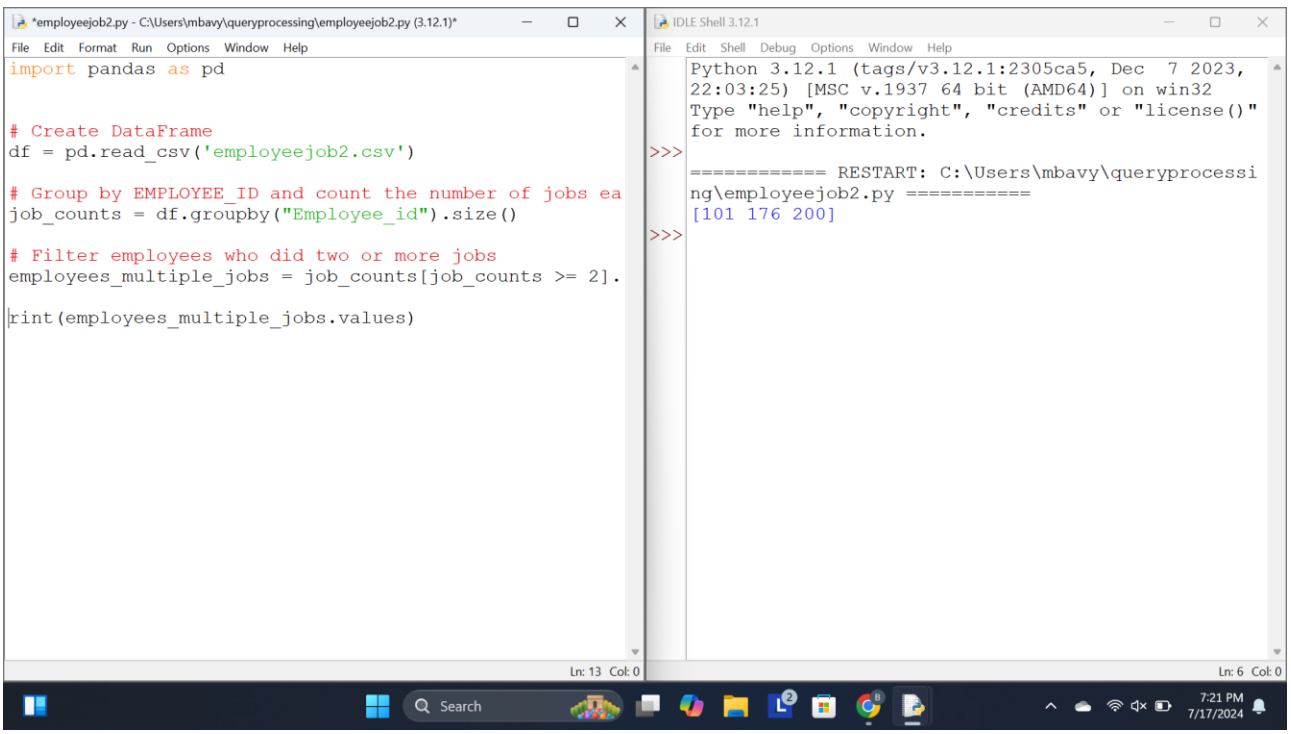
Code:

```
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 had

job_counts = df.groupby("Employee_id").size()

# Filter employees who did two or more jobs

employees_multiple_jobs = job_counts[job_counts >= 2].index

print(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 :

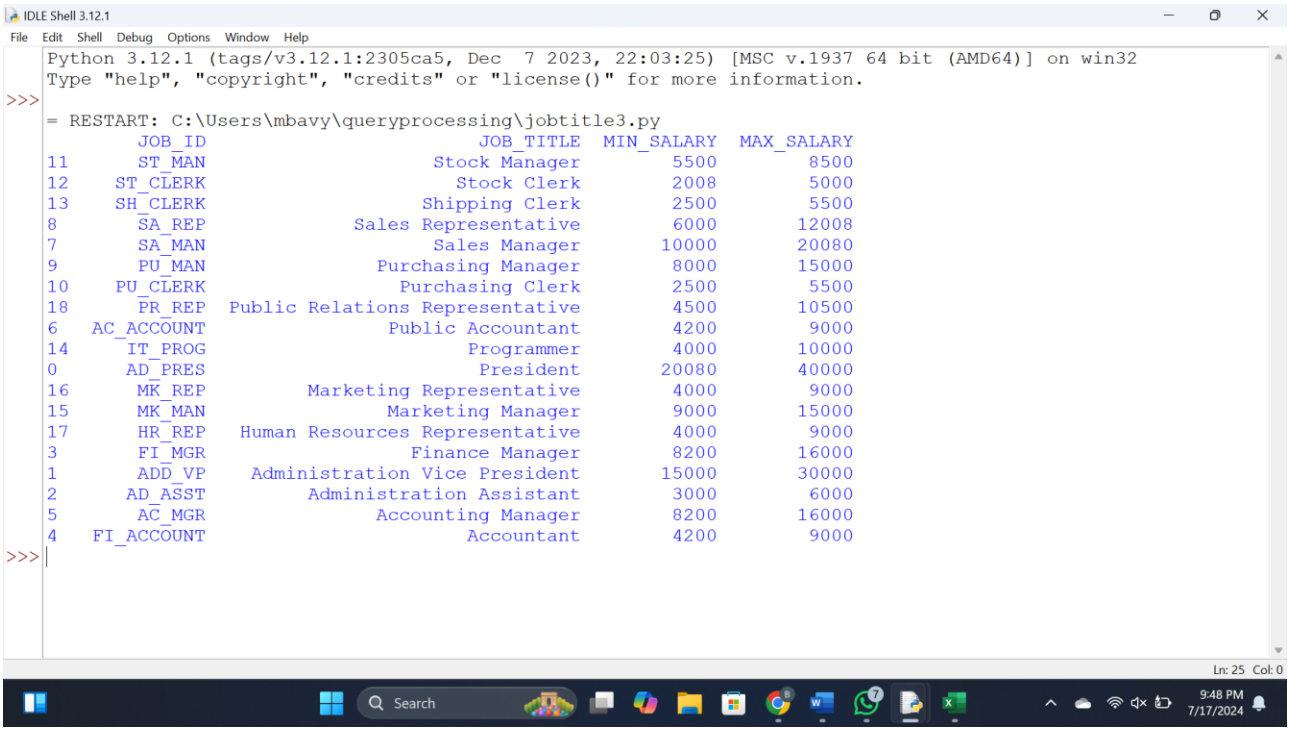
```
import pandas as pd

# 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:



```
IDLE Shell 3.12.1
File  Edit  Shell  Debug  Options  Window  Help
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023, 22:03:25) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\mbavy\queryprocessing\jobtitle3.py
        JOB_ID                      JOB_TITLE  MIN_SALARY  MAX_SALARY
11      ST_MAN                    Stock Manager        5500        8500
12    ST_CLERK                      Stock Clerk        2008        5000
13    SH_CLERK                   Shipping Clerk        2500        5500
8       SA_REP              Sales Representative        6000       12008
7       SA_MAN                    Sales Manager       10000       20080
9       PU_MAN                Purchasing Manager        8000       15000
10    PU_CLERK                  Purchasing Clerk        2500        5500
18      PR_REP  Public Relations Representative        4500       10500
6   AC_ACCOUNT                Public Accountant        4200        9000
14     IT_PROG                       Programmer        4000       10000
0      AD_PRES                        President       20080       40000
16      MK_REP         Marketing Representative        4000        9000
15      MK_MAN                Marketing Manager        9000       15000
17      HR_REP   Human Resources Representative        4000        9000
3       FI_MGR                  Finance Manager        8200       16000
1       ADD_VP    Administration Vice President       15000       30000
2      AD_ASST         Administration Assistant        3000        6000
5       AC_MGR               Accounting Manager        8200       16000
4   FI_ACCOUNT                       Accountant        4200        9000
>>>
Ln: 25  Col: 0
Search
9:48 PM
7/17/2024
```

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:

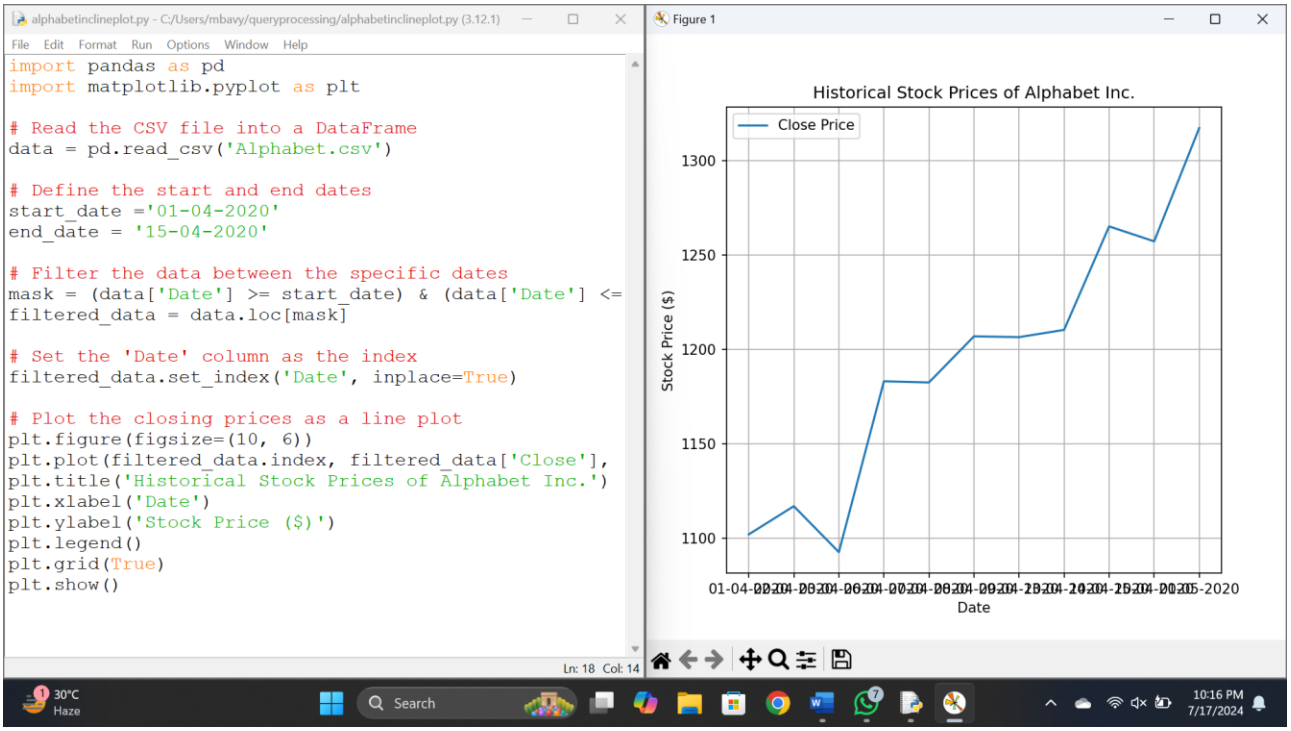
```
import pandas as pd

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:

Date	Open	High	Low	Close	Adj Close	Volume
01-04-2020	1122	1129.69	1097.45	1105.62	1105.62	2343100
02-04-2020	1098.26	1126.86	1096.4	1120.84	1120.84	1964900
03-04-2020	1119.015	1123.54	1079.81	1097.88	1097.88	2313400
06-04-2020	1138	1194.66	1130.94	1186.92	1186.92	2664700
07-04-2020	1221	1225	1182.23	1186.51	1186.51	2387300
08-04-2020	1206.5	1219.07	1188.16	1210.28	1210.28	1975100
09-04-2020	1224.08	1225.57	1196.735	1211.45	1211.45	2175400
13-04-2020	1209.18	1220.51	1187.598	1217.56	1217.56	1739800
14-04-2020	1245.09	1282.07	1236.93	1269.23	1269.23	2470400
15-04-2020	1245.61	1280.46	1240.4	1262.47	1262.47	1671700
16-04-2020	1274.1	1279	1242.62	1263.47	1263.47	2518100
17-04-2020	1284.85	1294.43	1271.23	1283.25	1283.25	1949000
20-04-2020	1271	1281.6	1261.37	1266.61	1266.61	1695500
21-04-2020	1247	1254.27	1209.71	1216.34	1216.34	2153000
22-04-2020	1245.54	1285.613	1242	1263.21	1263.21	2093100
23-04-2020	1271.55	1293.31	1265.67	1276.31	1276.31	1566200
24-04-2020	1261.17	1280.4	1249.45	1279.31	1279.31	1640400
27-04-2020	1296	1296.15	1269	1275.88	1275.88	1600600
28-04-2020	1287.93	1288.05	1232.2	1233.67	1233.67	2951300
29-04-2020	1341.46	1359.99	1325.34	1341.48	1341.48	3793600
30-04-2020	1324.88	1352.82	1322.49	1348.66	1348.66	2665400
01-05-2020	1328.5	1352.07	1311	1320.61	1320.61	2072500

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:

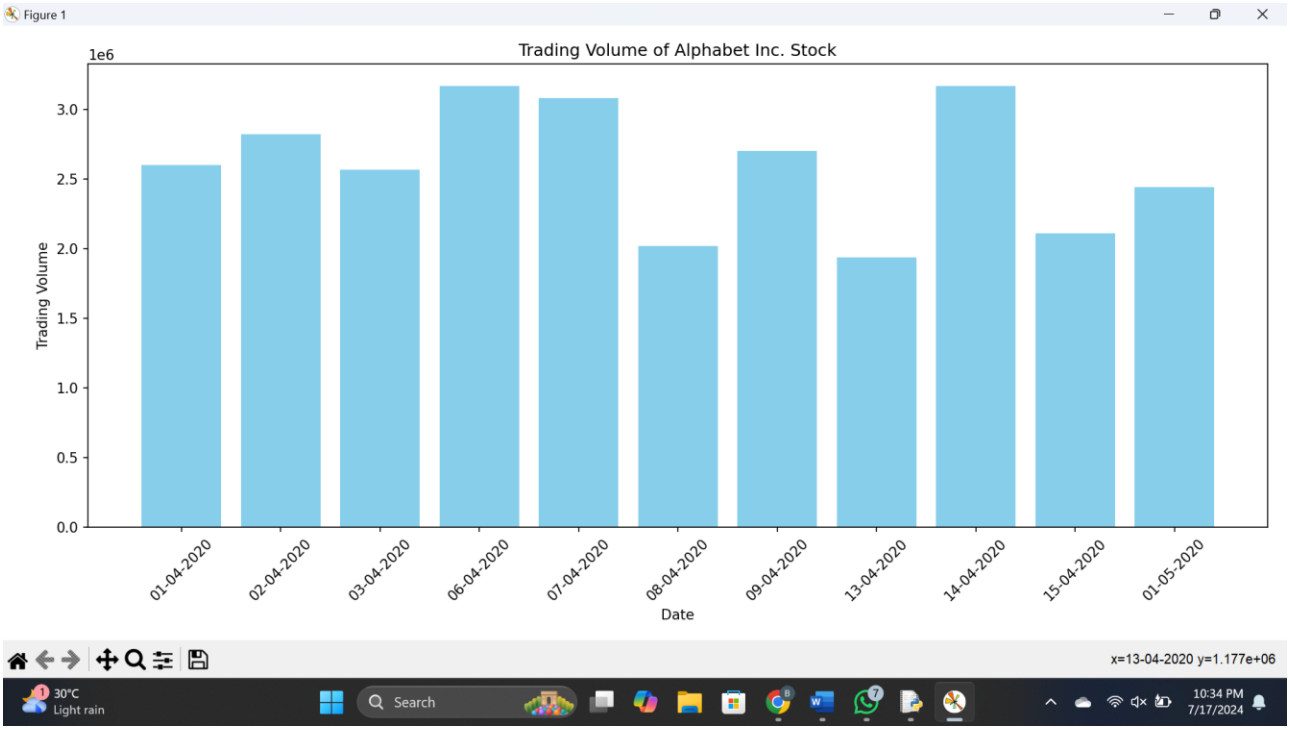
```
import pandas as pd

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:

Date	Open	High	Low	Close	Adj Close	Volume
01-04-2020	1122	1129.69	1097.45	1105.62	1105.62	2343100
02-04-2020	1098.26	1126.86	1096.4	1120.84	1120.84	1964900
03-04-2020	1119.015	1123.54	1079.81	1097.88	1097.88	2313400
06-04-2020	1138	1194.66	1130.94	1186.92	1186.92	2664700
07-04-2020	1221	1225	1182.23	1186.51	1186.51	2387300
08-04-2020	1206.5	1219.07	1188.16	1210.28	1210.28	1975100
09-04-2020	1224.08	1225.57	1196.735	1211.45	1211.45	2175400
13-04-2020	1209.18	1220.51	1187.598	1217.56	1217.56	1739800
14-04-2020	1245.09	1282.07	1236.93	1269.23	1269.23	2470400
15-04-2020	1245.61	1280.46	1240.4	1262.47	1262.47	1671700
16-04-2020	1274.1	1279	1242.62	1263.47	1263.47	2518100
17-04-2020	1284.85	1294.43	1271.23	1283.25	1283.25	1949000
20-04-2020	1271	1281.6	1261.37	1266.61	1266.61	1695500
21-04-2020	1247	1254.27	1209.71	1216.34	1216.34	2153000
22-04-2020	1245.54	1285.613	1242	1263.21	1263.21	2093100
23-04-2020	1271.55	1293.31	1265.67	1276.31	1276.31	1566200
24-04-2020	1261.17	1280.4	1249.45	1279.31	1279.31	1640400
27-04-2020	1296	1296.15	1269	1275.88	1275.88	1600600
28-04-2020	1287.93	1288.05	1232.2	1233.67	1233.67	2951300
29-04-2020	1341.46	1359.99	1325.34	1341.48	1341.48	3793600
30-04-2020	1324.88	1352.82	1322.49	1348.66	1348.66	2665400
01-05-2020	1328.5	1352.07	1311	1320.61	1320.61	2072500

Sample Output :



Result :

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

Experiment 6

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:

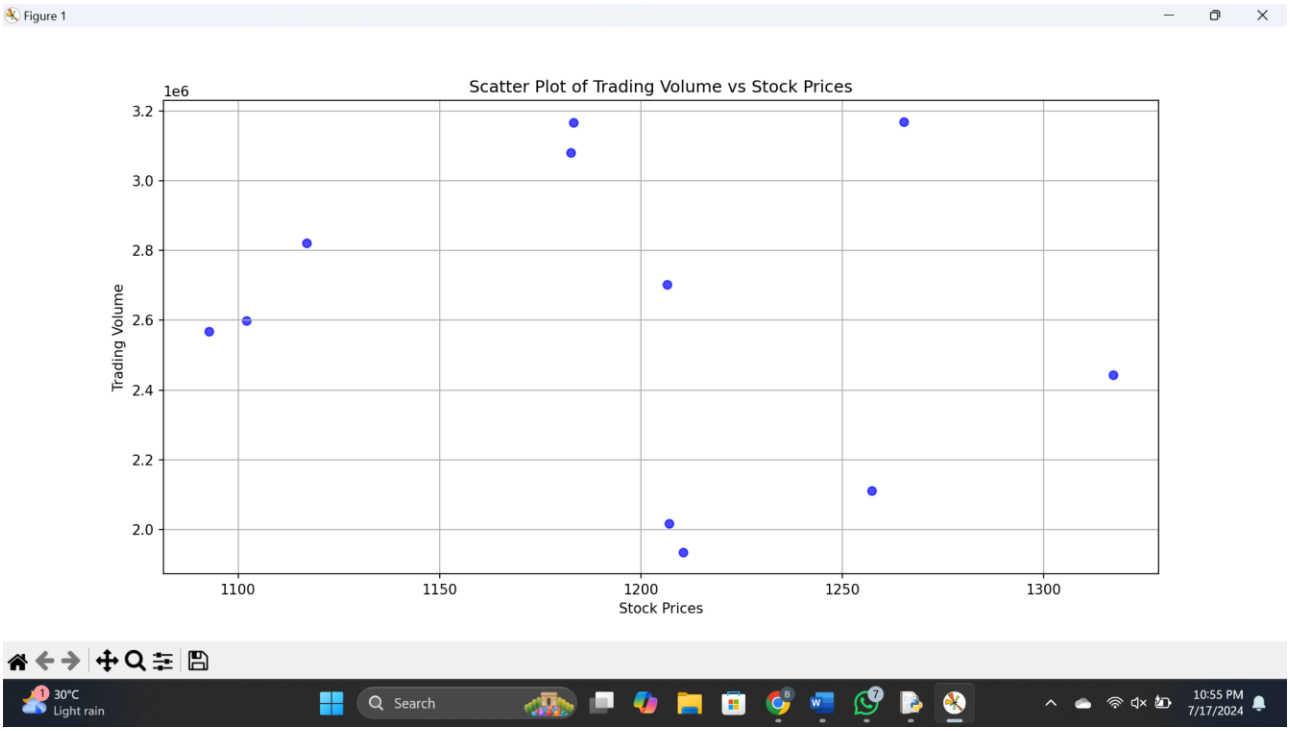
```
import pandas as pd

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:

Date	Open	High	Low	Close	Adj Close	Volume
01-04-2020	1122	1129.69	1097.45	1105.62	1105.62	2343100
02-04-2020	1098.26	1126.86	1096.4	1120.84	1120.84	1964900
03-04-2020	1119.015	1123.54	1079.81	1097.88	1097.88	2313400
06-04-2020	1138	1194.66	1130.94	1186.92	1186.92	2664700
07-04-2020	1221	1225	1182.23	1186.51	1186.51	2387300
08-04-2020	1206.5	1219.07	1188.16	1210.28	1210.28	1975100
09-04-2020	1224.08	1225.57	1196.735	1211.45	1211.45	2175400
13-04-2020	1209.18	1220.51	1187.598	1217.56	1217.56	1739800
14-04-2020	1245.09	1282.07	1236.93	1269.23	1269.23	2470400
15-04-2020	1245.61	1280.46	1240.4	1262.47	1262.47	1671700
16-04-2020	1274.1	1279	1242.62	1263.47	1263.47	2518100
17-04-2020	1284.85	1294.43	1271.23	1283.25	1283.25	1949000
20-04-2020	1271	1281.6	1261.37	1266.61	1266.61	1695500
21-04-2020	1247	1254.27	1209.71	1216.34	1216.34	2153000
22-04-2020	1245.54	1285.613	1242	1263.21	1263.21	2093100
23-04-2020	1271.55	1293.31	1265.67	1276.31	1276.31	1566200
24-04-2020	1261.17	1280.4	1249.45	1279.31	1279.31	1640400
27-04-2020	1296	1296.15	1269	1275.88	1275.88	1600600
28-04-2020	1287.93	1288.05	1232.2	1233.67	1233.67	2951300
29-04-2020	1341.46	1359.99	1325.34	1341.48	1341.48	3793600
30-04-2020	1324.88	1352.82	1322.49	1348.66	1348.66	2665400
01-05-2020	1328.5	1352.07	1311	1320.61	1320.61	2072500

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.