1. Write a Pandas program to select distinct department id from employees file.

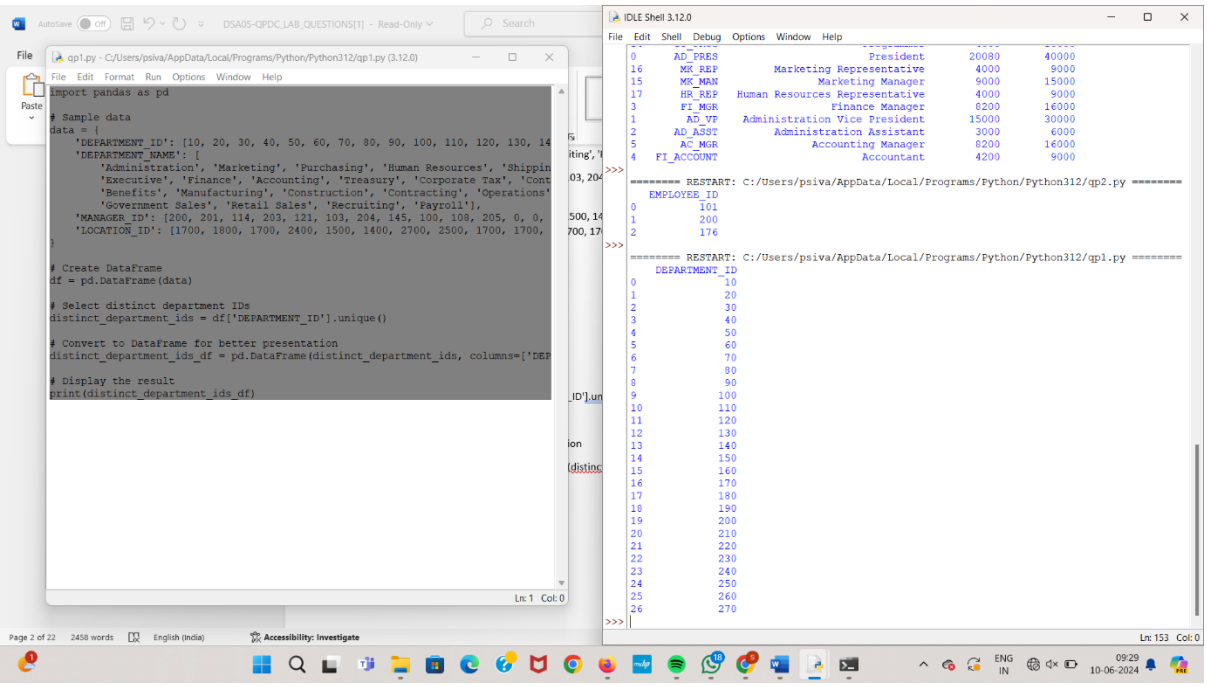
DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing	114	1700
40	Human Resources	203	2400
50	Shipping	121	1500
60	IT	103	1400
70	Public Relations	204	2700
80	Sales	145	2500
90	Executive	100	1700
100	Finance	108	1700
110	Accounting	205	1700
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

Code:

```
import pandas as pd


# Sample data

data = {

   'DEPARTMENT_ID': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160, 170, 180, 190, 200, 210, 220, 230, 240, 250, 260, 270],

   'DEPARTMENT_NAME': [

      'Administration', 'Marketing', 'Purchasing', 'Human Resources', 'Shipping', 'IT', 'Public Relations', 'Sales',

      'Executive', 'Finance', 'Accounting', 'Treasury', 'Corporate Tax', 'Control And Credit', 'Shareholder Services',
```

```
    'Benefits', 'Manufacturing', 'Construction', 'Contracting', 'Operations', 'IT Support', 'NOC', 'IT Helpdesk',
    'Government Sales', 'Retail Sales', 'Recruiting', 'Payroll'],
  'MANAGER_ID': [200, 201, 114, 203, 121, 103, 204, 145, 100, 108, 205, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
  'LOCATION_ID': [1700, 1800, 1700, 2400, 1500, 1400, 2700, 2500, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700]
}

# Create DataFrame
df = pd.DataFrame(data)

# Select distinct department IDs
distinct_department_ids = df['DEPARTMENT_ID'].unique()

# Convert to DataFrame for better presentation
distinct_department_ids_df = pd.DataFrame(distinct_department_ids, columns=['DEPARTMENT_ID'])

# Display the result
print(distinct_department_ids_df)
```

Output:



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

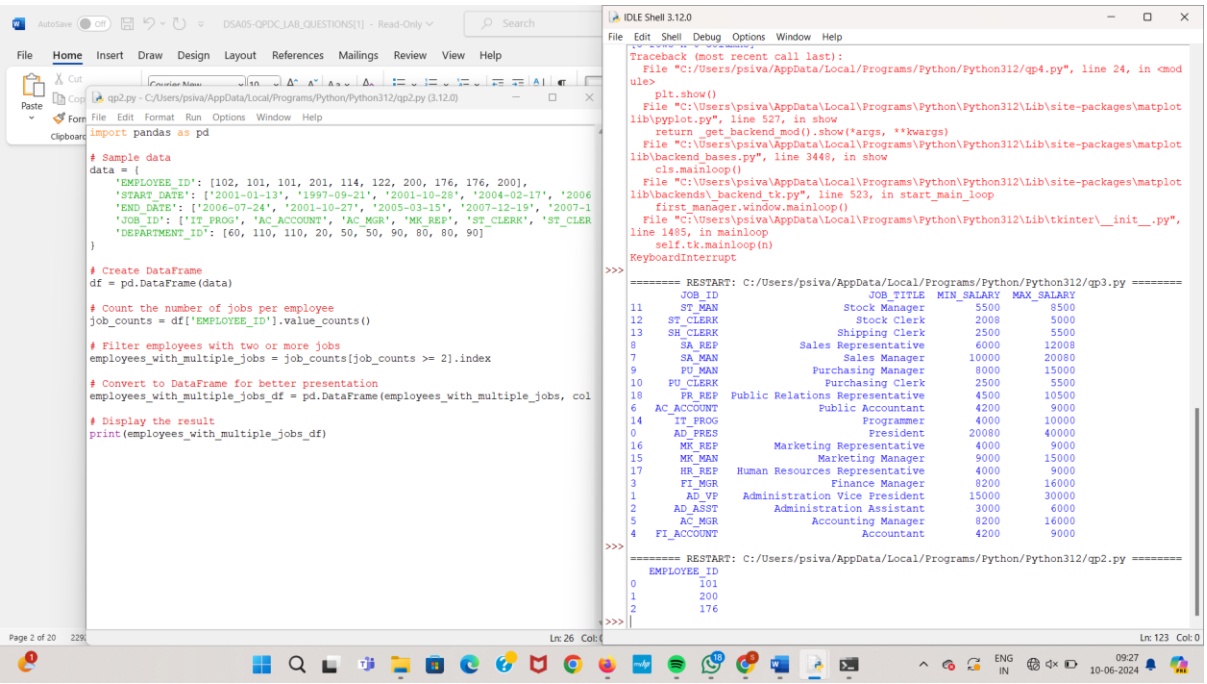
EMPLOYEE_ID	START_DATE	END_DATE	JOB_ID	DEPARTMENT_ID
102	2001-01-13	2006-07-24	IT_PROG	60
101	1997-09-21	2001-10-27	AC_ACCOUNT	110
101	2001-10-28	2005-03-15	AC_MGR	110
201	2004-02-17	2007-12-19	MK_REP	20
114	2006-03-24	2007-12-31	ST_CLERK	50
122	2007-01-01	2007-12-31	ST_CLERK	50
200	1995-09-17	2001-06-17	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

Code:

```
import pandas as pd

# Sample data
data = {
    'EMPLOYEE_ID': [102, 101, 101, 201, 114, 122, 200, 176, 176, 200],
    'START_DATE': ['2001-01-13', '1997-09-21', '2001-10-28', '2004-02-17',
'2006-03-24', '2007-01-01', '1995-09-17', '2006-03-24', '2007-01-01',
'2002-07-01'],
    'END_DATE': ['2006-07-24', '2001-10-27', '2005-03-15', '2007-12-19',
'2007-12-31', '2007-12-31', '2001-06-17', '2006-12-31', '2007-12-31',
'2006-12-31'],
```

```
    'JOB_ID': ['IT_PROG', 'AC_ACCOUNT', 'AC_MGR', 'MK_REP', 'ST_CLERK',
'ST_CLERK', 'AD_ASST', 'SA_REP', 'SA_MAN', 'AC_ACCOUNT'],
    'DEPARTMENT_ID': [60, 110, 110, 20, 50, 50, 90, 80, 80, 90]
}

# Create DataFrame
df = pd.DataFrame(data)

# Count the number of jobs per employee
job_counts = df['EMPLOYEE_ID'].value_counts()

# Filter employees with two or more jobs
employees_with_multiple_jobs = job_counts[job_counts >= 2].index

# Convert to DataFrame for better presentation
employees_with_multiple_jobs_df =
pd.DataFrame(employees_with_multiple_jobs, columns=['EMPLOYEE_ID'])

# Display the result
print(employees_with_multiple_jobs_df)
```

Output:



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

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 |
+------------+---------------------------------+------------+------------+
```

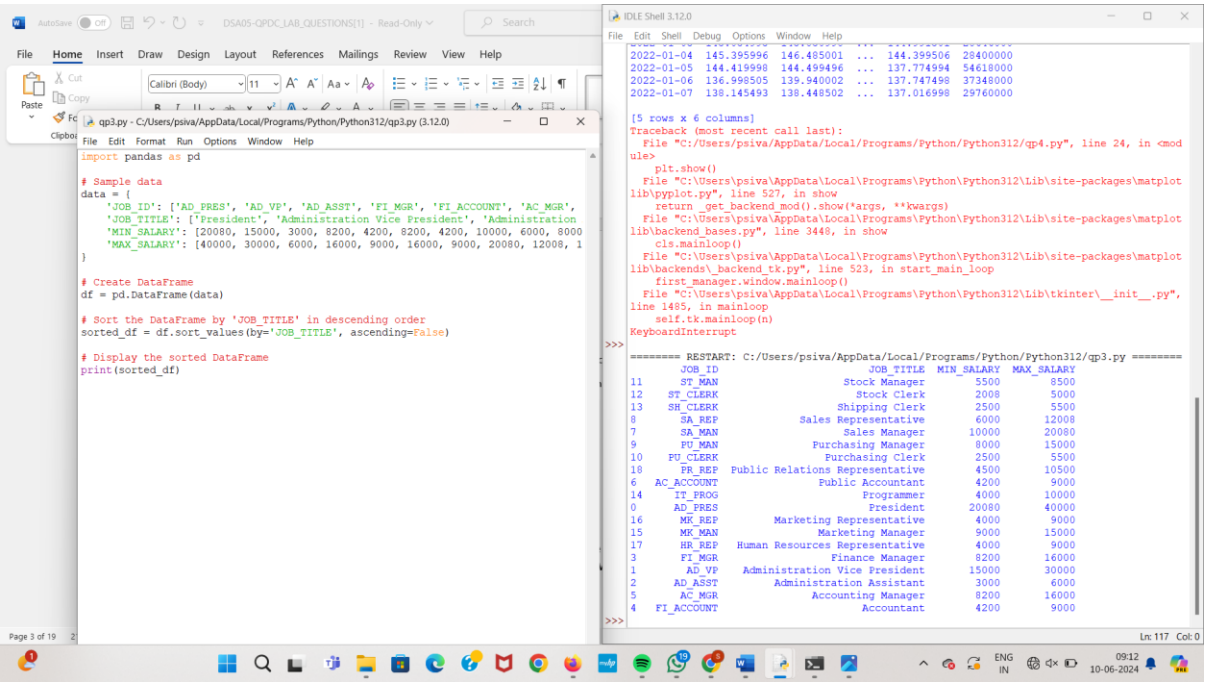
Code:

```
import pandas as pd


# Sample data

data = {

    'JOB_ID': ['AD_PRES', 'AD_VP', 'AD_ASST', 'FI_MGR', 'FI_ACCOUNT', 'AC_MGR', 'AC_ACCOUNT',
'SA_MAN', 'SA_REP', 'PU_MAN', 'PU_CLERK', 'ST_MAN', 'ST_CLERK', 'SH_CLERK', 'IT_PROG',
'MK_MAN', 'MK_REP', 'HR_REP', 'PR_REP'],

    'JOB_TITLE': ['President', 'Administration Vice President', 'Administration Assistant', 'Finance
Manager', 'Accountant', 'Accounting Manager', 'Public Accountant', 'Sales Manager', 'Sales
Representative', 'Purchasing Manager', 'Purchasing Clerk', 'Stock Manager', 'Stock Clerk', 'Shipping
Clerk', 'Programmer', 'Marketing Manager', 'Marketing Representative', 'Human Resources
Representative', 'Public Relations Representative'],

    'MIN_SALARY': [20080, 15000, 3000, 8200, 4200, 8200, 4200, 10000, 6000, 8000, 2500, 5500,
2008, 2500, 4000, 9000, 4000, 4000, 4500],

    'MAX_SALARY': [40000, 30000, 6000, 16000, 9000, 16000, 9000, 20080, 12008, 15000, 5500, 8500,
5000, 5500, 10000, 15000, 9000, 9000, 10500]

}


# Create DataFrame

df = pd.DataFrame(data)


# Sort the DataFrame by 'JOB_TITLE' in descending order

sorted_df = df.sort_values(by='JOB_TITLE', ascending=False)
```

Display the sorted DataFrame

print(sorted_df)

output:



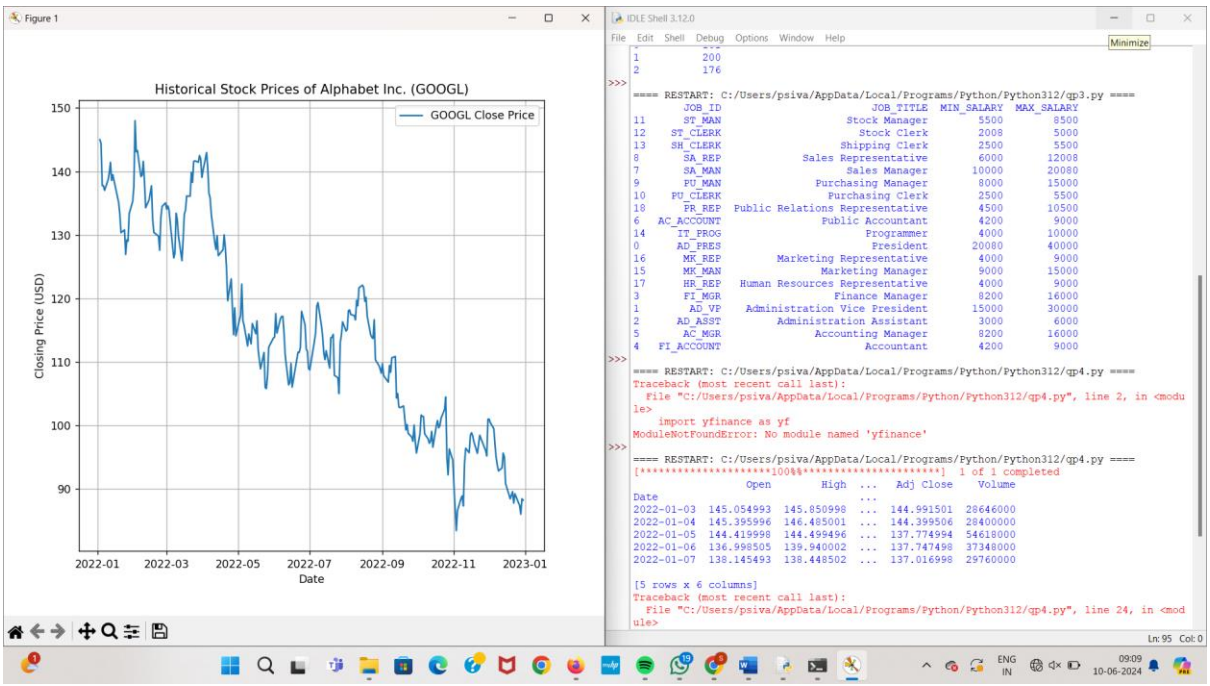
4. Write a Pandas program to create a line plot of the historical stock prices of Alphabet Inc. between two specific dates.

   Code:

```
import pandas as pd
import yfinance as yf
import matplotlib.pyplot as plt

# Define the ticker symbol and date range
ticker = 'GOOGL'
start_date = '2022-01-01'
end_date = '2023-01-01'

# Download historical stock prices for Alphabet Inc. (GOOGL)
data = yf.download(ticker, start=start_date, end=end_date)

# Display the first few rows of the data
print(data.head())
```

```
# Plot the closing prices
plt.figure(figsize=(10, 6))
plt.plot(data.index, data['Close'], label='GOOGL Close Price')
plt.xlabel('Date')
plt.ylabel('Closing Price (USD)')
plt.title(f'Historical Stock Prices of Alphabet Inc. ({ticker})')
plt.legend()
plt.grid(True)
plt.show()
```

Output:



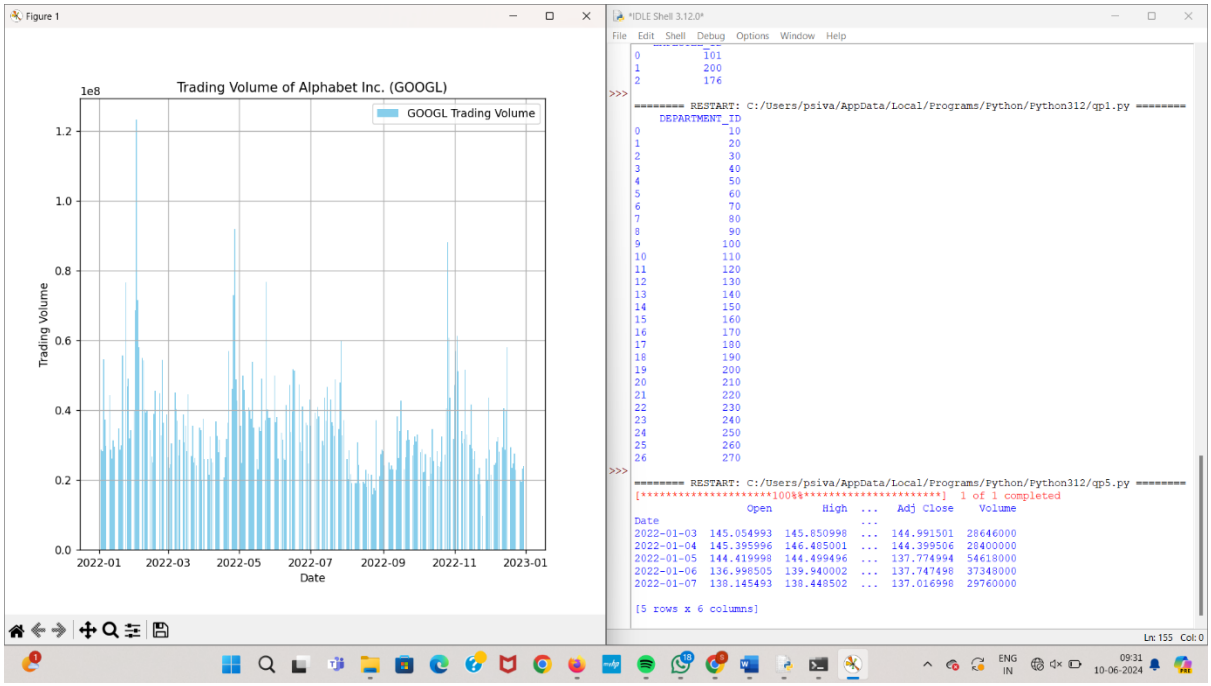
5. Write a Pandas program to create a bar plot of the trading volume of Alphabet Inc. stock between two specific dates.

Code:

```
import pandas as pd

import yfinance as yf

import matplotlib.pyplot as plt


# Define the ticker symbol and date range

ticker = 'GOOGL'
```

```
start_date = '2022-01-01'
end_date = '2023-01-01'

# Download historical stock prices for Alphabet Inc. (GOOGL)
data = yf.download(ticker, start=start_date, end=end_date)

# Display the first few rows of the data
print(data.head())

# Plot the trading volume
plt.figure(figsize=(12, 6))
plt.bar(data.index, data['Volume'], label='GOOGL Trading Volume',
color='skyblue')
plt.xlabel('Date')
plt.ylabel('Trading Volume')
plt.title(f'Trading Volume of Alphabet Inc. ({ticker})')
plt.legend()
plt.grid(True)
plt.show()
```

Output:



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

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

Code:

```
import pandas as pd

import matplotlib.pyplot as plt

# Creating a DataFrame from the provided data

data = {
```

'Date': ['01-04-2020', '02-04-2020', '03-04-2020', '06-04-2020', '07-04-2020', '08-04-2020', '09-04-2020',

'13-04-2020', '14-04-2020', '15-04-2020', '16-04-2020', '17-04-2020', '20-04-2020', '21-04-2020',

'22-04-2020', '23-04-2020', '24-04-2020', '27-04-2020', '28-04-2020', '29-04-2020', '30-04-2020',

'01-05-2020'],

'Open': [1122, 1098.26, 1119.015, 1138, 1221, 1206.5, 1224.08, 1209.18, 1245.09, 1245.61, 1274.1,

1284.85, 1271, 1247, 1245.54, 1271.55, 1261.17, 1296, 1287.93, 1341.46, 1324.88, 1328.5],

'High': [1129.69, 1126.86, 1123.54, 1194.66, 1225, 1219.07, 1225.57, 1220.51, 1282.07, 1280.46, 1279,

1294.43, 1281.6, 1254.27, 1285.613, 1293.31, 1280.4, 1296.15, 1288.05, 1359.99, 1352.82, 1352.07],

'Low': [1097.45, 1096.4, 1079.81, 1130.94, 1182.23, 1188.16, 1196.735, 1187.598, 1236.93, 1240.4,

1242.62, 1271.23, 1261.37, 1209.71, 1242, 1265.67, 1249.45, 1269, 1232.2, 1325.34, 1322.49, 1311],

'Close': [1105.62, 1120.84, 1097.88, 1186.92, 1186.51, 1210.28, 1211.45, 1217.56, 1269.23, 1262.47,

1263.47, 1283.25, 1266.61, 1216.34, 1263.21, 1276.31, 1279.31, 1275.88, 1233.67, 1341.48, 1348.66,

1320.61],

'Adj Close': [1105.62, 1120.84, 1097.88, 1186.92, 1186.51, 1210.28, 1211.45, 1217.56, 1269.23,

1262.47, 1263.47, 1283.25, 1266.61, 1216.34, 1263.21, 1276.31, 1279.31, 1275.88, 1233.67, 1341.48,

1348.66, 1320.61],

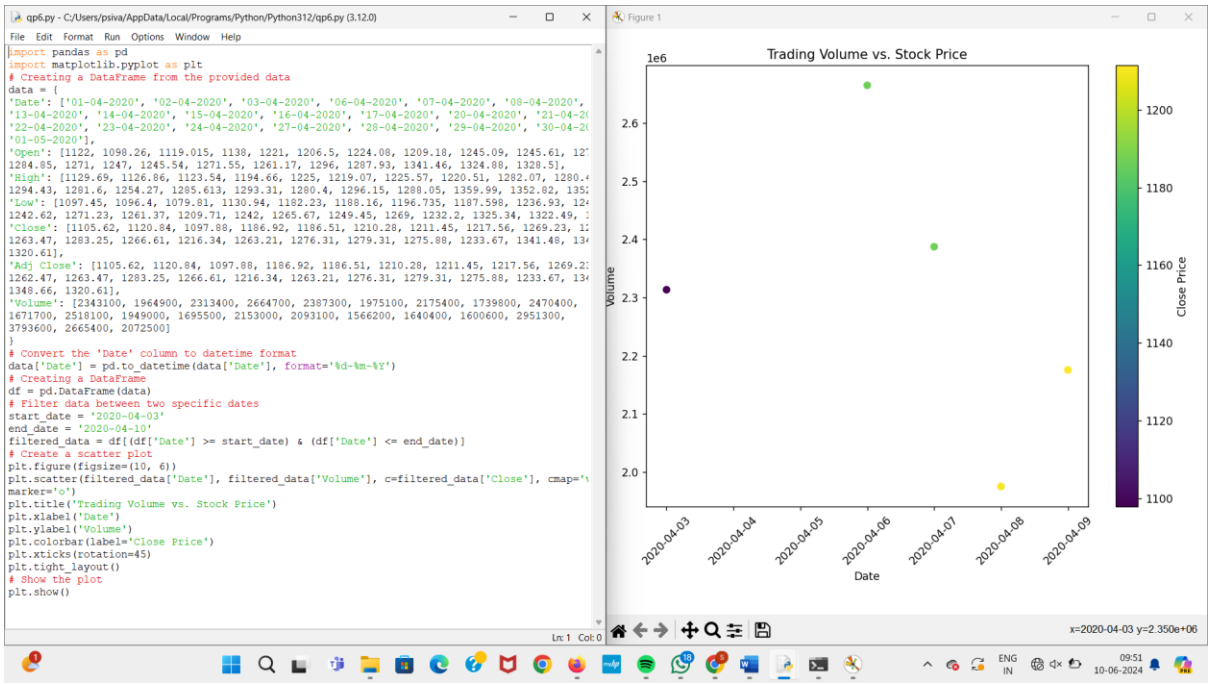
'Volume': [2343100, 1964900, 2313400, 2664700, 2387300, 1975100, 2175400, 1739800, 2470400,

1671700, 2518100, 1949000, 1695500, 2153000, 2093100, 1566200, 1640400, 1600600, 2951300,

3793600, 2665400, 2072500]

```
}
# Convert the 'Date' column to datetime format
data['Date'] = pd.to_datetime(data['Date'], format='%d-%m-%Y')
# Creating a DataFrame
df = pd.DataFrame(data)
# Filter data between two specific dates
start_date = '2020-04-03'
end_date = '2020-04-10'
filtered_data = df[(df['Date'] >= start_date) & (df['Date'] <= end_date)]
# Create a scatter plot
plt.figure(figsize=(10, 6))
plt.scatter(filtered_data['Date'], filtered_data['Volume'],
c=filtered_data['Close'], cmap='viridis',
marker='o')
plt.title('Trading Volume vs. Stock Price')
plt.xlabel('Date')
plt.ylabel('Volume')
plt.colorbar(label='Close Price')
plt.xticks(rotation=45)
plt.tight_layout()
# Show the plot
plt.show()
```

Output:



7. Write a Pandas program to create a Pivot table and find the maximum and minimum sale value of the items.(refer sales_data table)

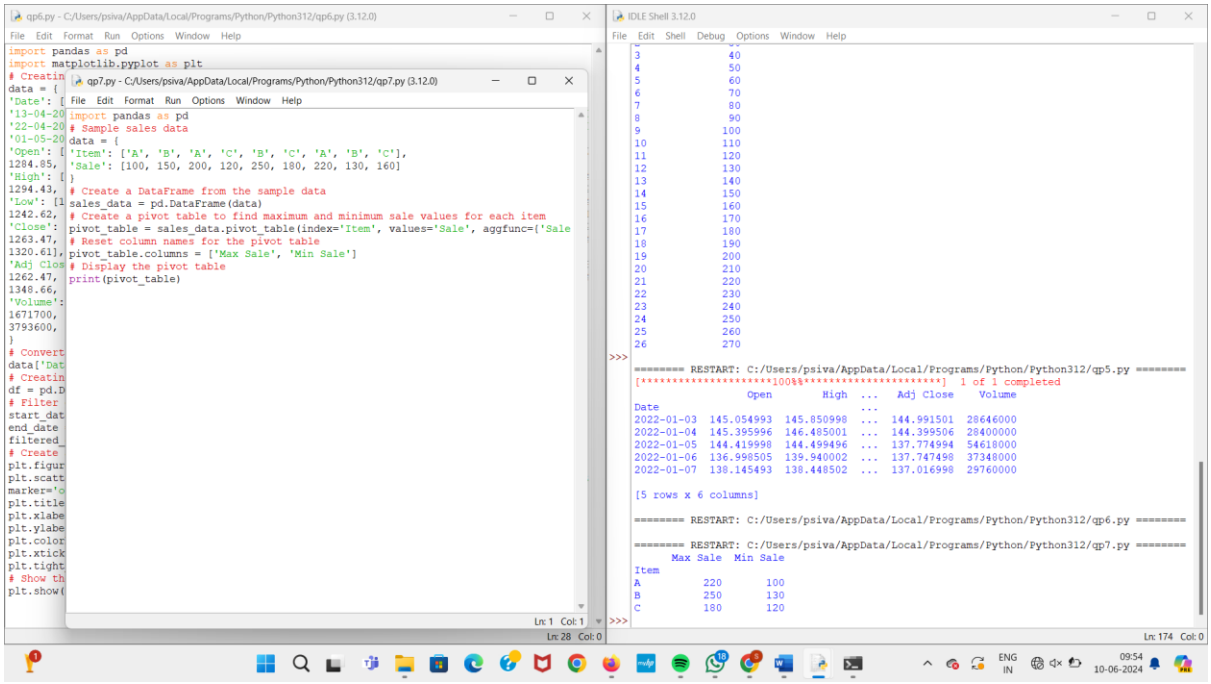
Code:

```
import pandas as pd

# Sample sales data

data = {

'Item': ['A', 'B', 'A', 'C', 'B', 'C', 'A', 'B', 'C'],

'Sale': [100, 150, 200, 120, 250, 180, 220, 130, 160]

}

# Create a DataFrame from the sample data

sales_data = pd.DataFrame(data)

# Create a pivot table to find maximum and minimum sale values for each item

pivot_table = sales_data.pivot_table(index='Item', values='Sale', aggfunc={'Sale': ['max', 'min']})

# Reset column names for the pivot table

pivot_table.columns = ['Max Sale', 'Min Sale']
```

Display the pivot table

print(pivot_table)

Output:



8. Write a Pandas program to create a Pivot table and find the item wise unit sold. .(refer sales_data table)

Code:

import pandas as pd

Sample sales data

data = {

'Item': ['A', 'B', 'A', 'C', 'B', 'C', 'A', 'B', 'C'],

'Units Sold': [10, 15, 20, 12, 25, 18, 22, 13, 16]

}

Create a DataFrame from the sample data

sales_data = pd.DataFrame(data)

Create a pivot table to find unit sold for each item

pivot_table = sales_data.pivot_table(index='Item', values='Units Sold', aggfunc='sum')

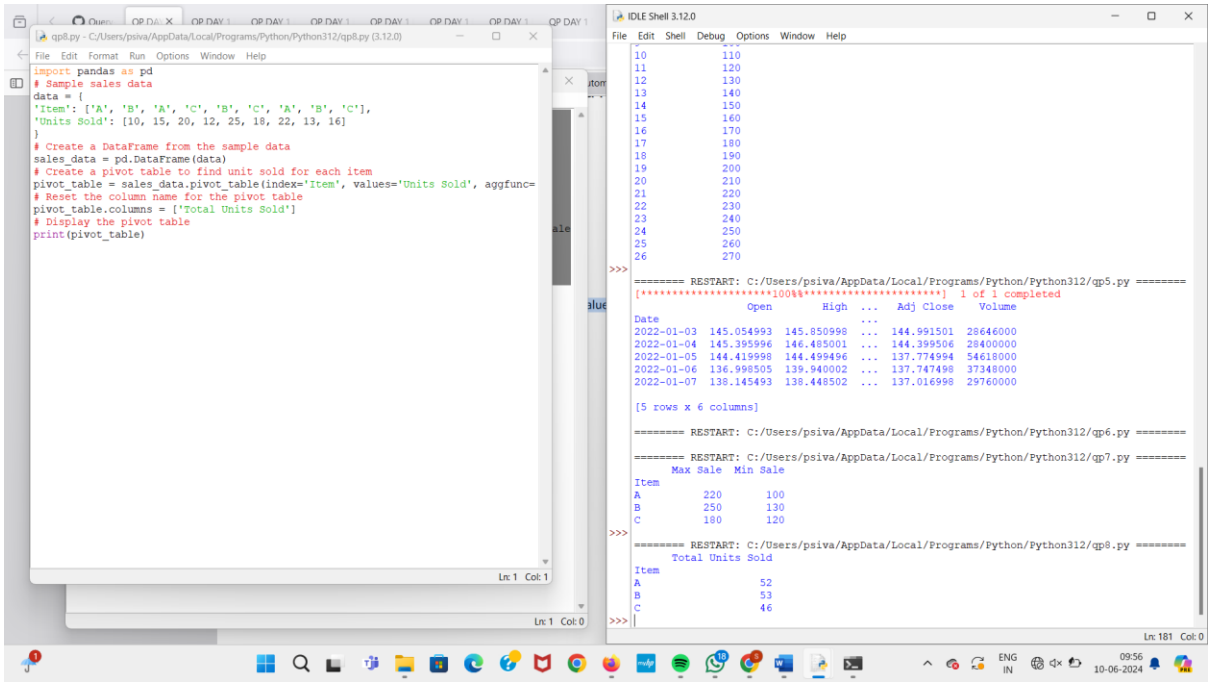
Reset the column name for the pivot table

pivot_table.columns = ['Total Units Sold']

Display the pivot table

print(pivot_table)

Output:



9. Write a Pandas program to create a Pivot table and find the total sale amount region wise, manager wise, sales man wise. .(refer sales_data table)

Sales_data:

OrderDate	Region	Manager	SalesMan	Item	Units	Unit_price	Sale_amt
1-6-18	East	Martha	Alexander	Television	95	1,198.00	1,13,810.00
1-23-18	Central	Hermann	Shelli	Home Theater	50	500.00	25,000.00
2-9-18	Central	Hermann	Luis	Television	36	1,198.00	43,128.00
2-26-18	Central	Timothy	David	Cell Phone	27	225.00	6,075.00
3-15-18	West	Timothy	Stephen	Television	56	1,198.00	67,088.00
4-1-18	East	Martha	Alexander	Home Theater	60	500.00	30,000.00
4-18-18	Central	Martha	Steven	Television	75	1,198.00	89,850.00
5-5-18	Central	Hermann	Luis	Television	90	1,198.00	1,07,820.00
5-22-18	West	Douglas	Michael	Television	32	1,198.00	38,336.00
6-8-18	East	Martha	Alexander	Home Theater	60	500.00	30,000.00
6-25-18	Central	Hermann	Sigal	Television	90	1,198.00	1,07,820.00

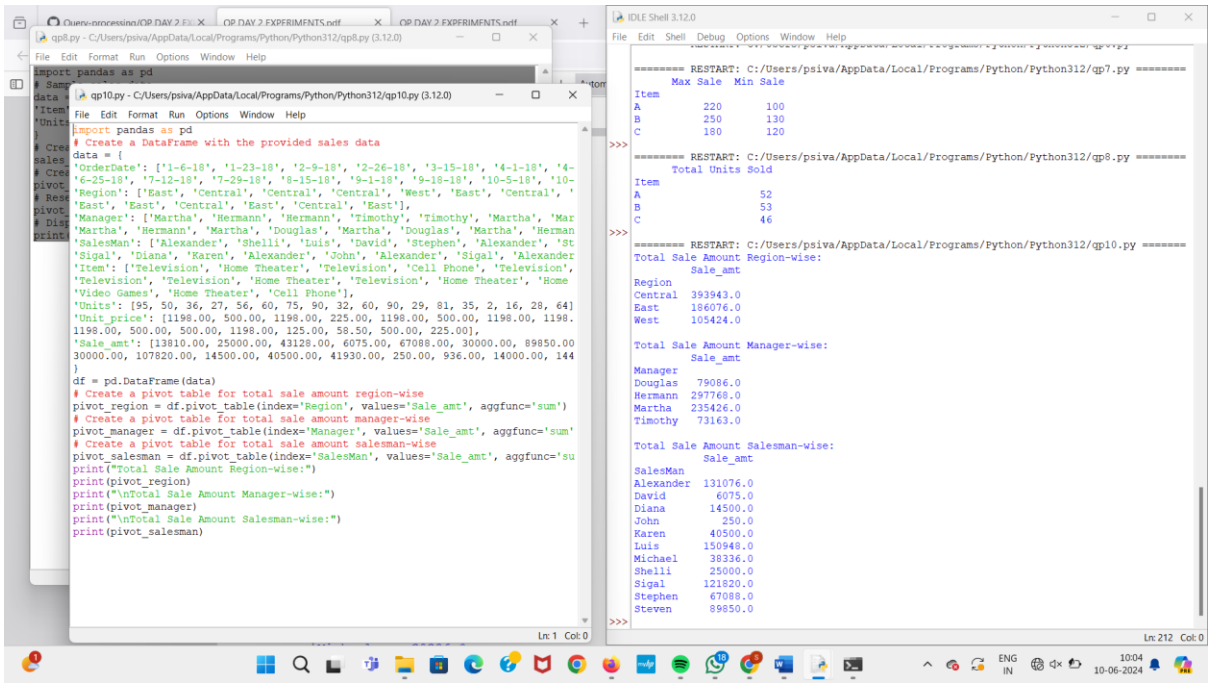
7-12-18	East	Martha	Diana	Home Theater	29	500.00	14,500.00
7-29-18	East	Douglas	Karen	Home Theater	81	500.00	40,500.00
8-15-18	East	Martha	Alexander	Television	35	1,198.00	41,930.00
9-1-18	Central	Douglas	John	Desk	2	125.00	250.00
9-18-18	East	Martha	Alexander	Video Games	16	58.50	936.00
10-5-18	Central	Hermann	Sigal	Home Theater	28	500.00	14,000.00
10-22-18	East	Martha	Alexander	Cell Phone	64	225.00	14,400.00

Code:

```
import pandas as pd

# Create a DataFrame with the provided sales data

data = {

'OrderDate': ['1-6-18', '1-23-18', '2-9-18', '2-26-18', '3-15-18', '4-1-18', '4-18-18', '5-5-18', '5-22-18', '6-8-18',

'6-25-18', '7-12-18', '7-29-18', '8-15-18', '9-1-18', '9-18-18', '10-5-18', '10-22-18'],

'Region': ['East', 'Central', 'Central', 'Central', 'West', 'East', 'Central', 'Central', 'West', 'East', 'Central', 'East',

'East', 'East', 'Central', 'East', 'Central', 'East'],

'Manager': ['Martha', 'Hermann', 'Hermann', 'Timothy', 'Timothy', 'Martha', 'Martha', 'Hermann', 'Douglas',

'Martha', 'Hermann', 'Martha', 'Douglas', 'Martha', 'Douglas', 'Martha', 'Hermann', 'Martha'],

'SalesMan': ['Alexander', 'Shelli', 'Luis', 'David', 'Stephen', 'Alexander', 'Steven', 'Luis', 'Michael', 'Alexander',

'Sigal', 'Diana', 'Karen', 'Alexander', 'John', 'Alexander', 'Sigal', 'Alexander'],

'Item': ['Television', 'Home Theater', 'Television', 'Cell Phone', 'Television', 'Home Theater', 'Television',

'Television', 'Television', 'Home Theater', 'Television', 'Home Theater', 'Home Theater', 'Television', 'Desk',
```

```
'Video Games', 'Home Theater', 'Cell Phone'],

'Units': [95, 50, 36, 27, 56, 60, 75, 90, 32, 60, 90, 29, 81, 35, 2, 16, 28, 64],

'Unit_price': [1198.00, 500.00, 1198.00, 225.00, 1198.00, 500.00, 1198.00,
1198.00, 1198.00, 500.00,

1198.00, 500.00, 500.00, 1198.00, 125.00, 58.50, 500.00, 225.00],

'Sale_amt': [13810.00, 25000.00, 43128.00, 6075.00, 67088.00, 30000.00,
89850.00, 107820.00, 38336.00,

30000.00, 107820.00, 14500.00, 40500.00, 41930.00, 250.00, 936.00,
14000.00, 14400.00]

}

df = pd.DataFrame(data)

# Create a pivot table for total sale amount region-wise

pivot_region = df.pivot_table(index='Region', values='Sale_amt',
aggfunc='sum')

# Create a pivot table for total sale amount manager-wise

pivot_manager = df.pivot_table(index='Manager', values='Sale_amt',
aggfunc='sum')

# Create a pivot table for total sale amount salesman-wise

pivot_salesman = df.pivot_table(index='SalesMan', values='Sale_amt',
aggfunc='sum')

print("Total Sale Amount Region-wise:")

print(pivot_region)

print("\nTotal Sale Amount Manager-wise:")

print(pivot_manager)

print("\nTotal Sale Amount Salesman-wise:")

print(pivot_salesman)
```

Output:



10.Create a dataframe of ten rows, four columns with random values. Write a Pandas program to highlight the negative numbers red and positive numbers black.

Expected Output:

	A	B	C	D	E
0	1	1.32921	-0.770033	-0.31628	-0.99081
1	2	-1.07082	-1.43871	0.564417	0.295722
2	3	-1.6264	0.219565	0.678805	1.88927
3	4	0.961538	0.104011	-0.481165	0.850229
4	5	1.45342	1.05774	0.165562	0.515018
5	6	-1.33694	0.562861	1.39285	-0.063328
6	7	0.121668	1.2076	-0.00204021	1.6278
7	8	0.354493	1.03753	-0.385684	0.519818
8	9	1.68658	-1.32596	1.42898	-2.08935
9	10	-0.12982	0.631523	-0.586538	0.29072

Code:

```
import pandas as pd

import numpy as np
```

```
np.random.seed(24)
df = pd.DataFrame({'A': np.linspace(1, 10, 10)})
df = pd.concat([df, pd.DataFrame(np.random.randn(10, 4),
columns=list('BCDE'))],
axis=1)
print("Original array:")
print(df)
def color_negative_red(val):
    color = 'red' if val < 0 else 'black'
    return 'color: %s' % color
print("\nNegative numbers red and positive numbers black:")
df.style.applymap(color_negative_red)
```

Output: