**DSA0512-QUERY PROCESSING FOR DATASCIENCE WITH FUZZY MATCHING**

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 |

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

**AIM:**

To develop a pandas program to select distinct department id from employees file.

**ALGORITHM:**

* Import the pandas library.
* Define the dataset as a dictionary with keys as column names and values as lists.
* Convert the dictionary to a Pandas DataFrame.
* Print a preview of the DataFrame.Extract unique department IDs from the
* "DEPARTMENT\_ID" column.
* Convert the unique department IDs to a new DataFrame and print it.

**PROGRAM:**

import pandas as pd

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],

"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]

}

df=pd.DataFrame(data)

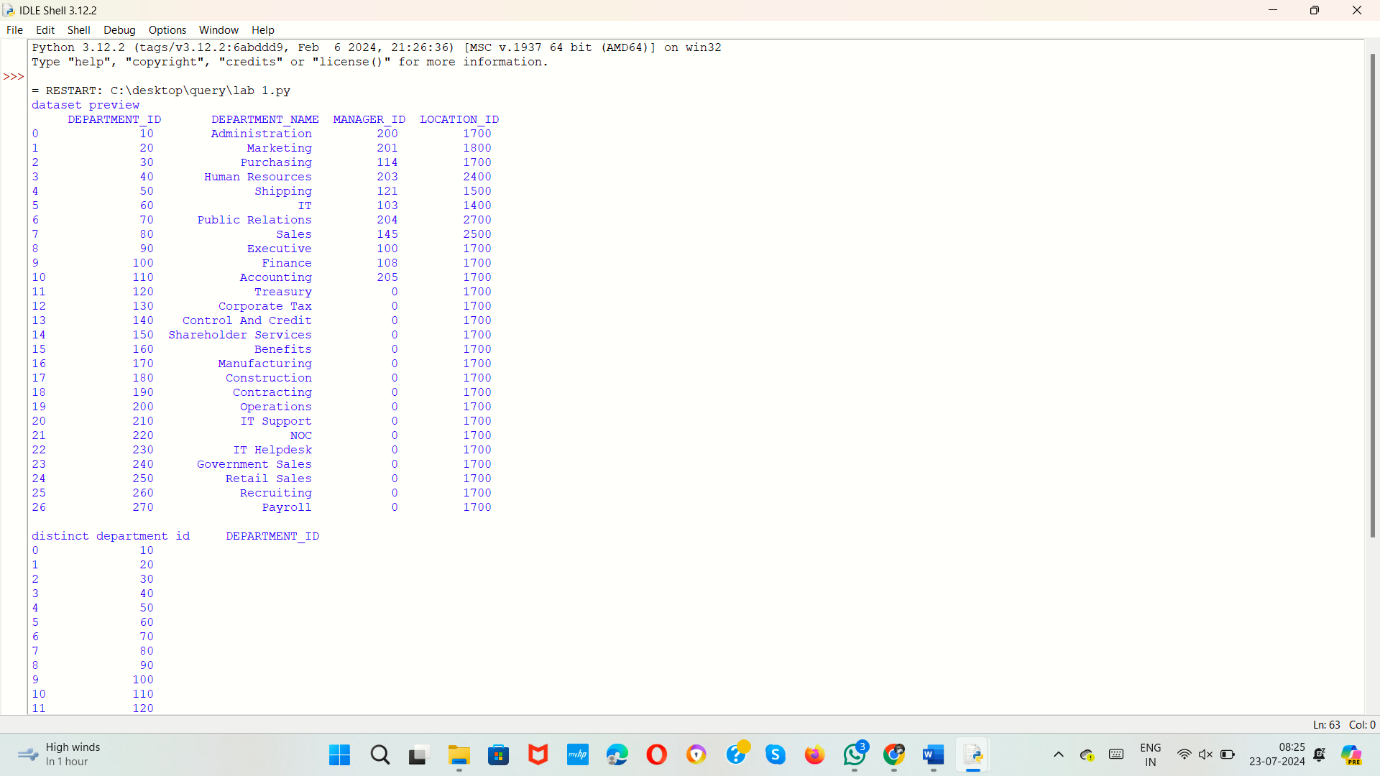
print("dataset preview\n",df)

distinct\_dept\_id=df["DEPARTMENT\_ID"].unique()

distinct\_dept\_id\_df=pd.DataFrame(distinct\_dept\_id,columns=["DEPARTMENT\_ID"])

print("\ndistinct department id",distinct\_dept\_id\_df)

**SAMPLE OUTPUT:**



1. 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 |

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

**AIM:**

To develop a pandas program to display the ID for those employees who did two or more jobs in the past.

**ALGORITHM:**

* Import the pandas library.
* Create a dictionary with employee data.
* Convert the dictionary into a pandas DataFrame.
* Print the DataFrame to preview the dataset.
* Group the DataFrame by "EMPLOYEE\_ID" and count the occurrences.
* Identify and print employees with two or more jobs in a new DataFrame**.**

**PROGRAM:**

import pandas as pd

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]

}

df=pd.DataFrame(data)

print("dataset preview:\n",df)

job\_counts=df.groupby("EMPLOYEE\_ID").size()

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

employee\_df=pd.DataFrame(employee\_with\_multiple\_jobs,columns=["EMPLOYEE\_ID"])

print("\nemployee id for those who are doing more than two jobs:\n",employee\_df)

**SAMPLE OUTPUT:**

****

1. 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 |

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

**AIM:**

To develop pandas program to display the details of jobs in descending sequence on job title.

**ALGORITHM:**

* Import the pandas library.
* Create a dataset using a dictionary with employee data.
* Convert the dictionary into a Pandas DataFrame.
* Print a preview of the DataFrame.
* Sort the DataFrame by the "JOB\_ID" column in descending order.
* Print the sorted DataFrame.

**PROGRAM:**

import pandas as pd

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]

}

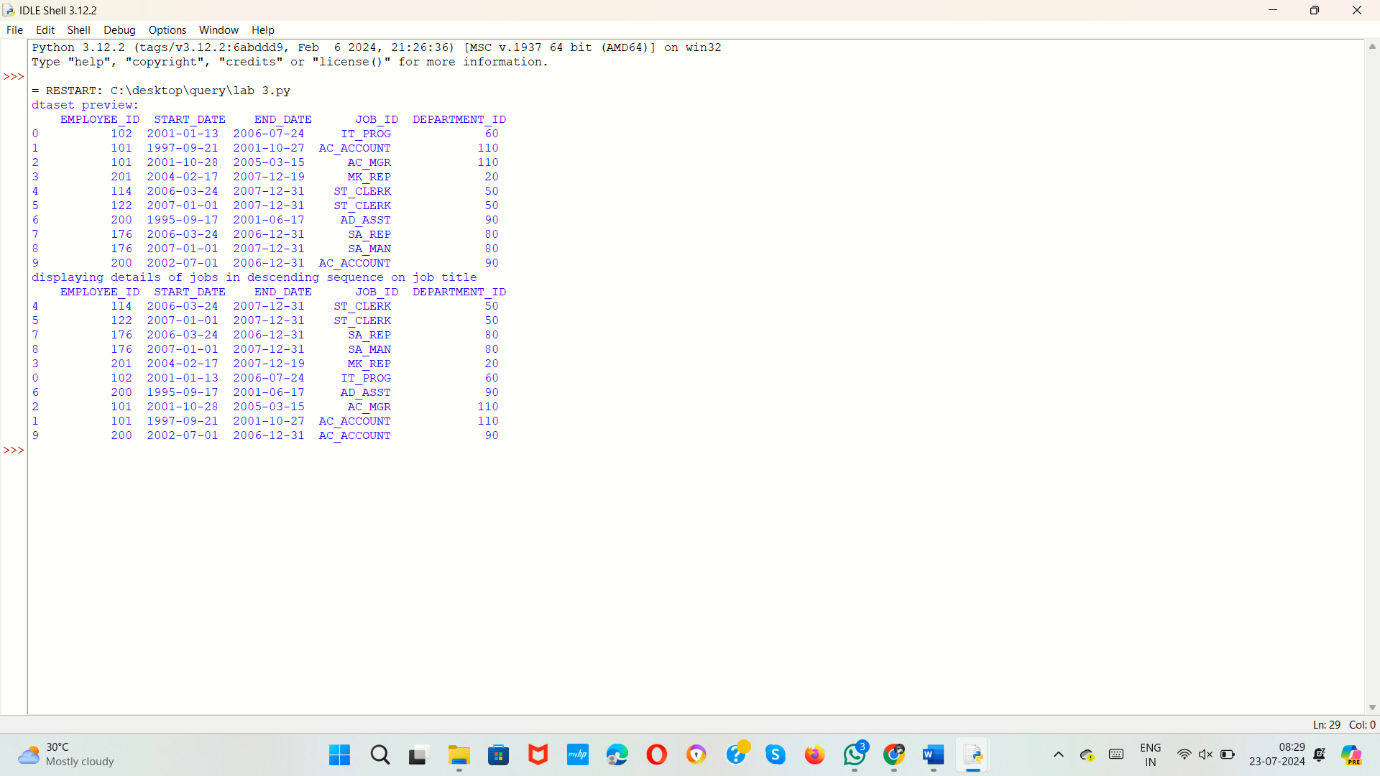
df=pd.DataFrame(data)

print("dtaset preview:\n",df)

sorted\_df=df.sort\_values(by="JOB\_ID",ascending=False)

print("displaying details of jobs in descending sequence on job title\n",sorted\_df)

**SAMPLE OUTPUT:**



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

**AIM:**

To develop a pandas program to create a line plot of the historical stock prices of Alphabet Inc. between two specific dates.

**ALGORITHM:**

1. Import `pandas`, `yfinance`, and `matplotlib.pyplot`.

2. Define the stock ticker symbol and date range.

3. Download the stock data using `yf.download()`.

4. Print the stock data to the console.

5. Create a plot with `plt.plot()` to visualize the closing price over time.

6. Customize and display the plot using labels, legend, grid, and `plt.show()`.

**PROGRAM:**

import pandas as pd

import yfinance as yf

import matplotlib.pyplot as plt

ticker="GOOGL"

start\_date="2023-01-01"

end\_date="2023-12-31"

data=yf.download(ticker,start=start\_date,end=end\_date)

print("stock data\n",data)

plt.figure(figsize=(12,6))

plt.plot(data.index,data['Close'],label='close price')

plt.xlabel("Date")

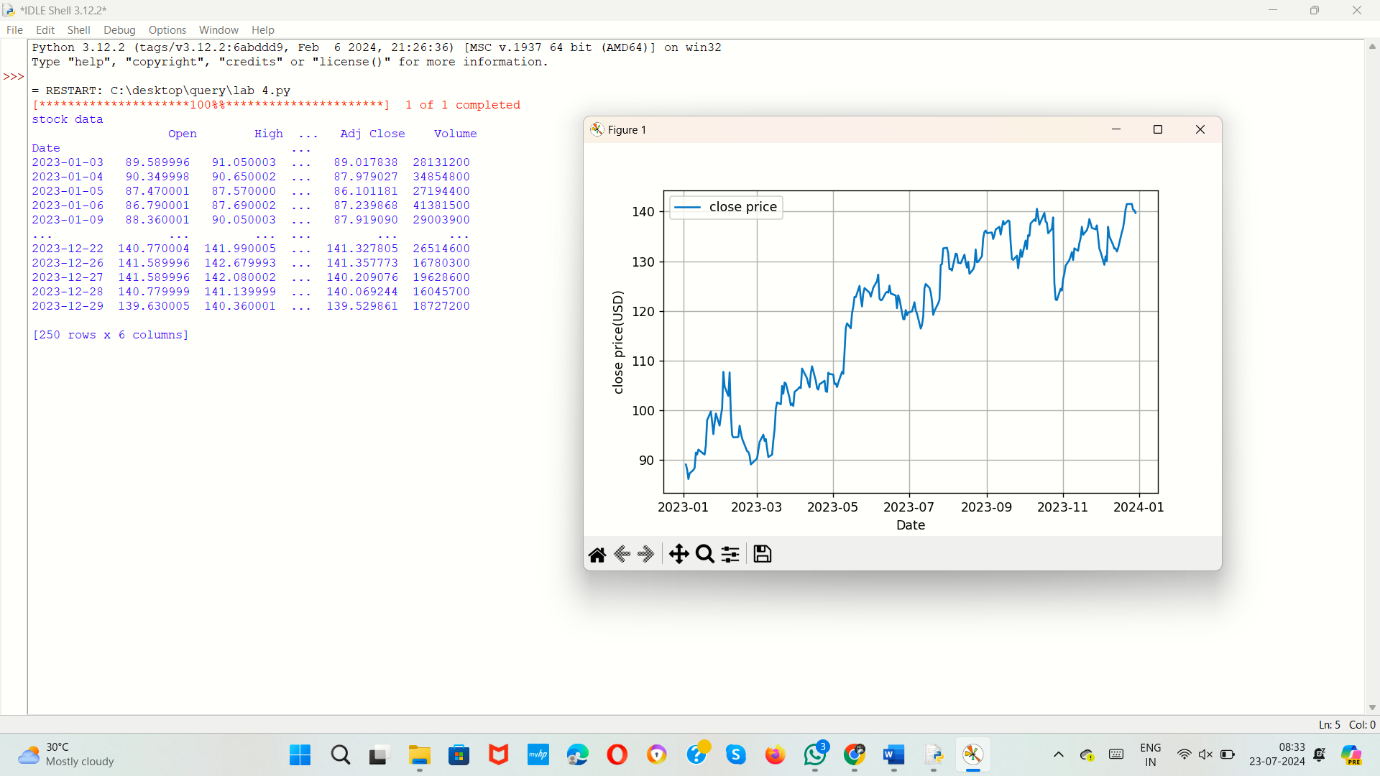
plt.ylabel("close price(USD)")

plt.legend()

plt.grid(True)

plt.show()

**SAMPLE OUTPUT:**



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

**AIM:**

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

**ALGORITHM:**

1. Import `pandas`, `yfinance`, and `matplotlib.pyplot`.

2. Set the stock ticker, start date, and end date.

3. Download stock data using `yf.download` with specified ticker and dates.

4. Print the retrieved stock data.

5. Create a bar chart of trading volume with dates on the x-axis and volume on the y-axis.

6. Add labels, title, legend, and grid, then display the plot.

**PROGRAM:**

import pandas as pd

import yfinance as yf

import matplotlib.pyplot as plt

ticker = "GOOGL"

start\_date = "2023-01-01"

end\_date = "2023-12-31"

data = yf.download(ticker, start=start\_date, end=end\_date)

print("stock data\n:",data)

plt.figure(figsize=(12, 6))

plt.bar(data.index, data['Volume'], label='Trading Volume', color='skyblue')

plt.xlabel('Date')

plt.ylabel('Trading Volume')

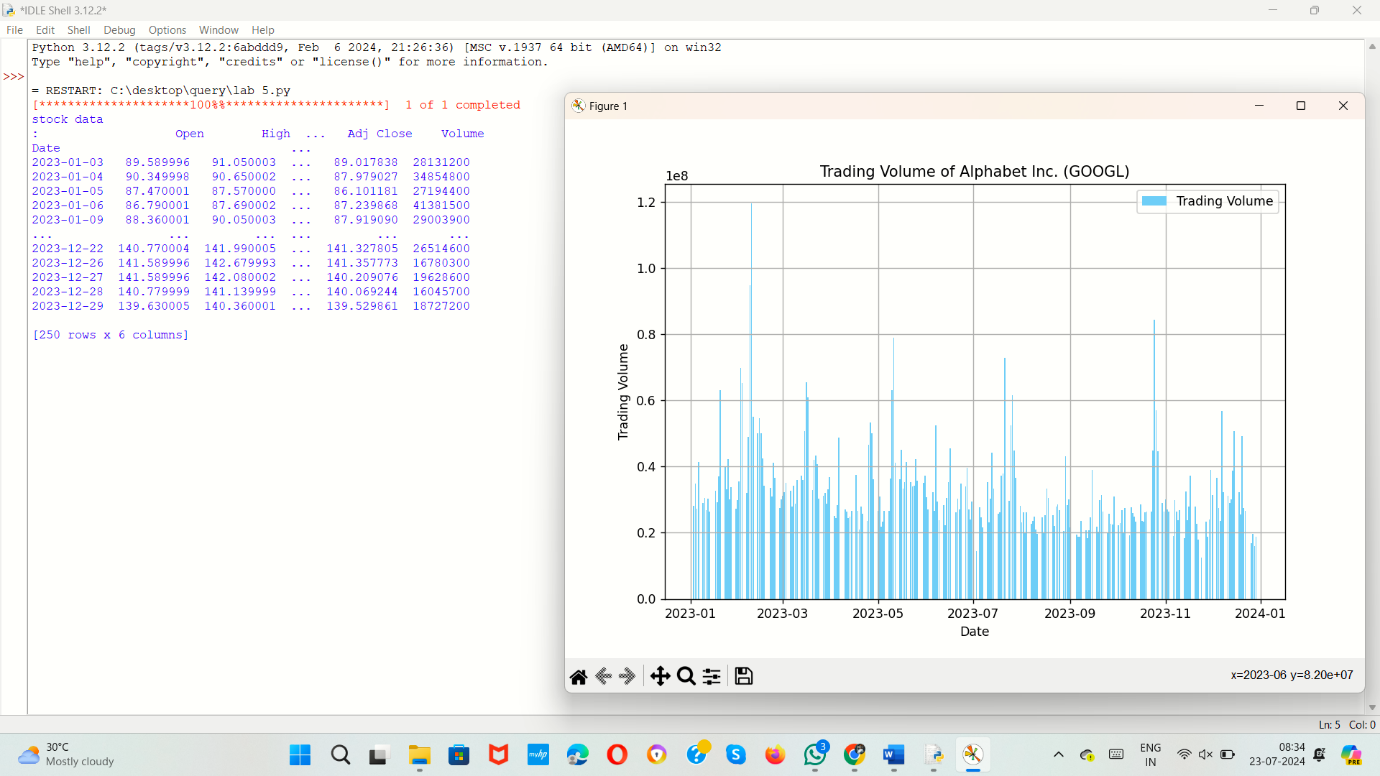
plt.title('Trading Volume of Alphabet Inc. (GOOGL)')

plt.legend()

plt.grid(True)

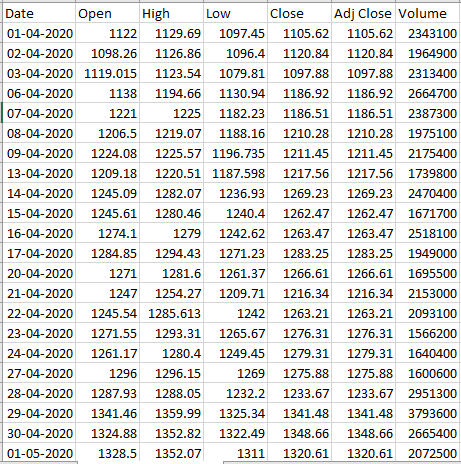
plt.show()

**SAMPLE 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:**



**AIM:**

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

**ALGORITHM:**

1. Import `pandas`, `yfinance`, and `matplotlib.pyplot`.

2. Set the stock ticker, start date, and end date.

3. Download stock data using `yf.download` with specified ticker and dates.

4. Print the retrieved stock data.

5. Create a scatter plot of trading volume with volume on the x-axis and closing price on the y-axis.

6. Add labels, title, and grid, then display the plot.

**PROGRAM:**

import pandas as pd

import matplotlib.pyplot as plt

import yfinance as yf

ticker="GOOGL"

start\_date="2023-01-01"

end\_date="2023-12-31"

data=yf.download(ticker,start=start\_date,end=end\_date)

print("stock data preview:\n",data)

plt.figure(figsize=(12,6))

plt.scatter(data['Volume'],data['Close'],alpha=0.5)

plt.xlabel("Trading volume")

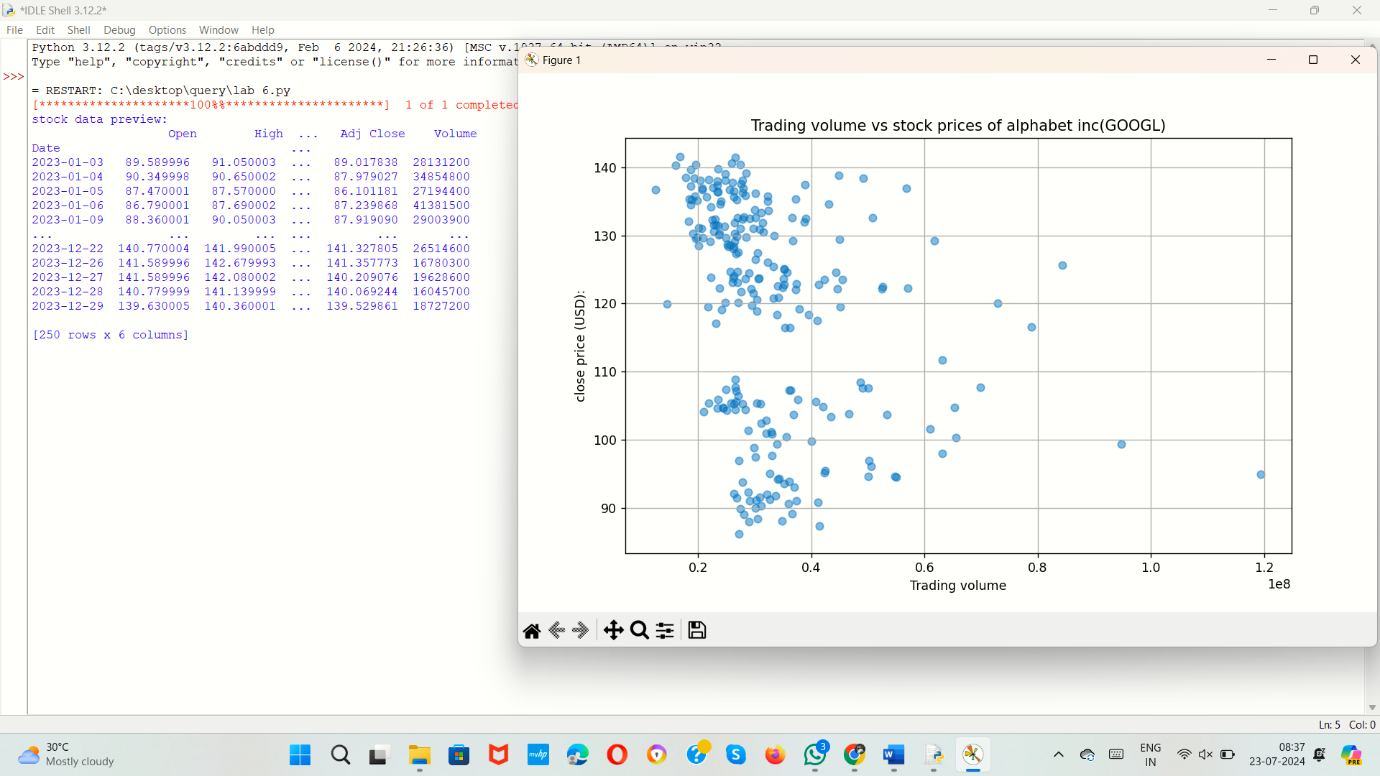
plt.ylabel("close price (USD):")

plt.title("Trading volume vs stock prices of alphabet inc(GOOGL)")

plt.grid(True)

plt.show()

**SAMPLE 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)

**AIM:**

To develop a pandas program to create a Pivot table and find the maximum and minimum sale value of the items.

**ALGORITHM:**

1. Import the `pandas` library.

2. Create a dictionary with data for order dates, items, units, and amounts.

3. Convert the dictionary into a DataFrame using `pd.DataFrame`.

4. Create a pivot table from the DataFrame with 'Amount' as values, 'Item' as index, and using 'max' and 'min' as aggregation functions.

5. Rename the columns of the pivot table to 'max\_sale' and 'min\_sale'.

6. Print the pivot table.

**PROGRAM:**

import pandas as pd

data={

"OrderDate": ["2023-01-01", "2023-01-01", "2023-01-02", "2023-01-02", "2023-01-03", "2023-01-03"],

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

"Units": [10, 5, 7, 10, 3, 15],

"Amount": [200, 150, 140, 300, 90, 300]

}

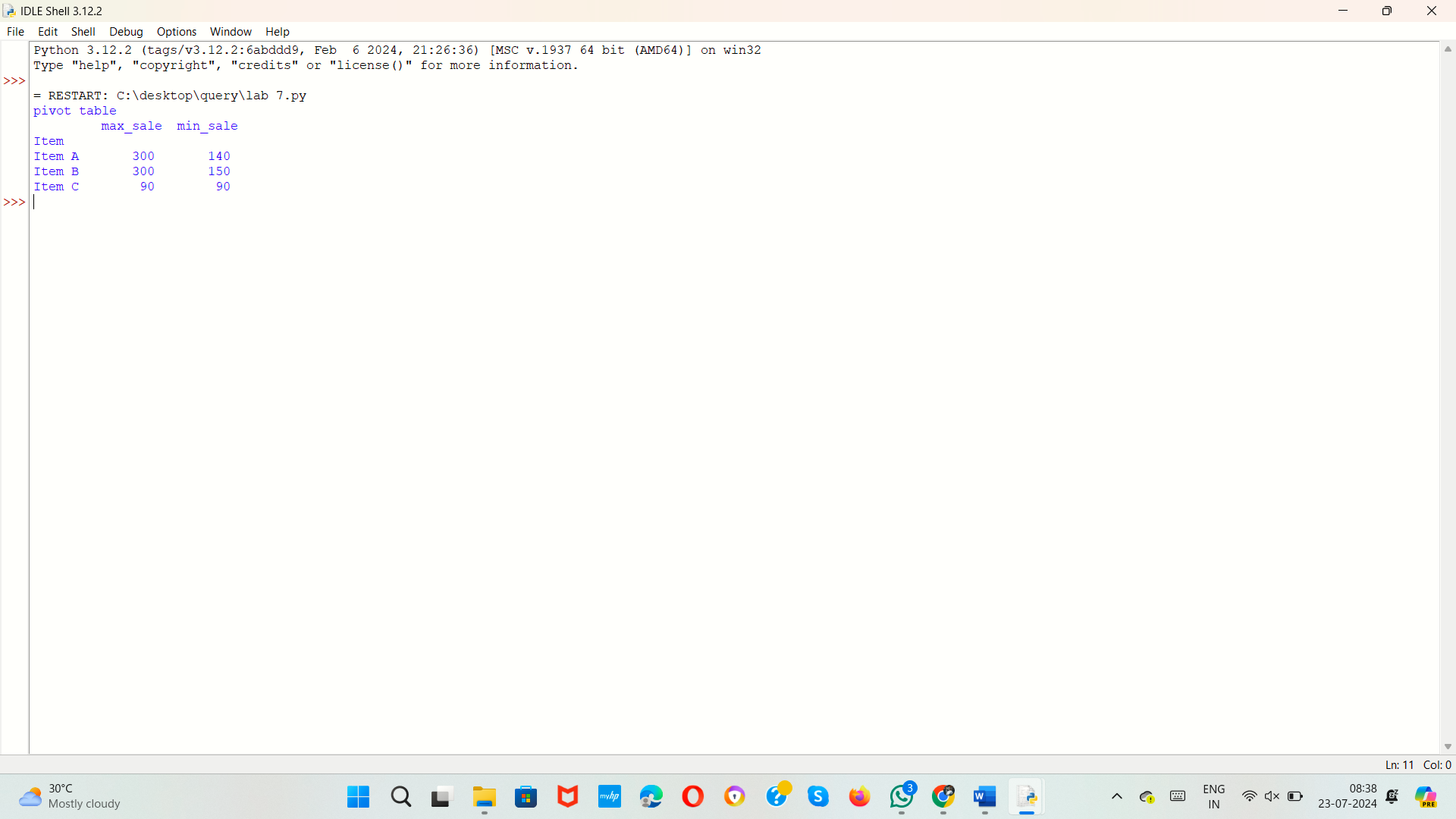
df=pd.DataFrame(data)

pivot\_table=df.pivot\_table(values='Amount',index='Item',aggfunc=['max','min'])

pivot\_table.columns=['max\_sale','min\_sale']

print("pivot table\n",pivot\_table)

**SAMPLE OUTPUT:**



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

**AIM:**

To develop a Pandas program to create a Pivot table and find the item wise unit sold.

**ALGORITHM:**

1. Import the `pandas` library.

2. Create a dictionary with data for order dates, items, units, and amounts.

3. Convert the dictionary into a DataFrame using `pd.DataFrame`.

4. Print the DataFrame.

5. Create a pivot table from the DataFrame with 'Units' as values, 'Item' as index, and using 'sum' as the aggregation function.

6. Rename the columns of the pivot table to 'Total\_unit\_sold'.

7. Print the pivot table.

**PROGRAM:**

import pandas as pd

data = {

"OrderDate": ["2023-01-01", "2023-01-01", "2023-01-02", "2023-01-02", "2023-01-03", "2023-01-03"],

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

"Units": [10, 5, 7, 10, 3, 15],

"Amount": [200, 150, 140, 300, 90, 300]

}

df=pd.DataFrame(data)

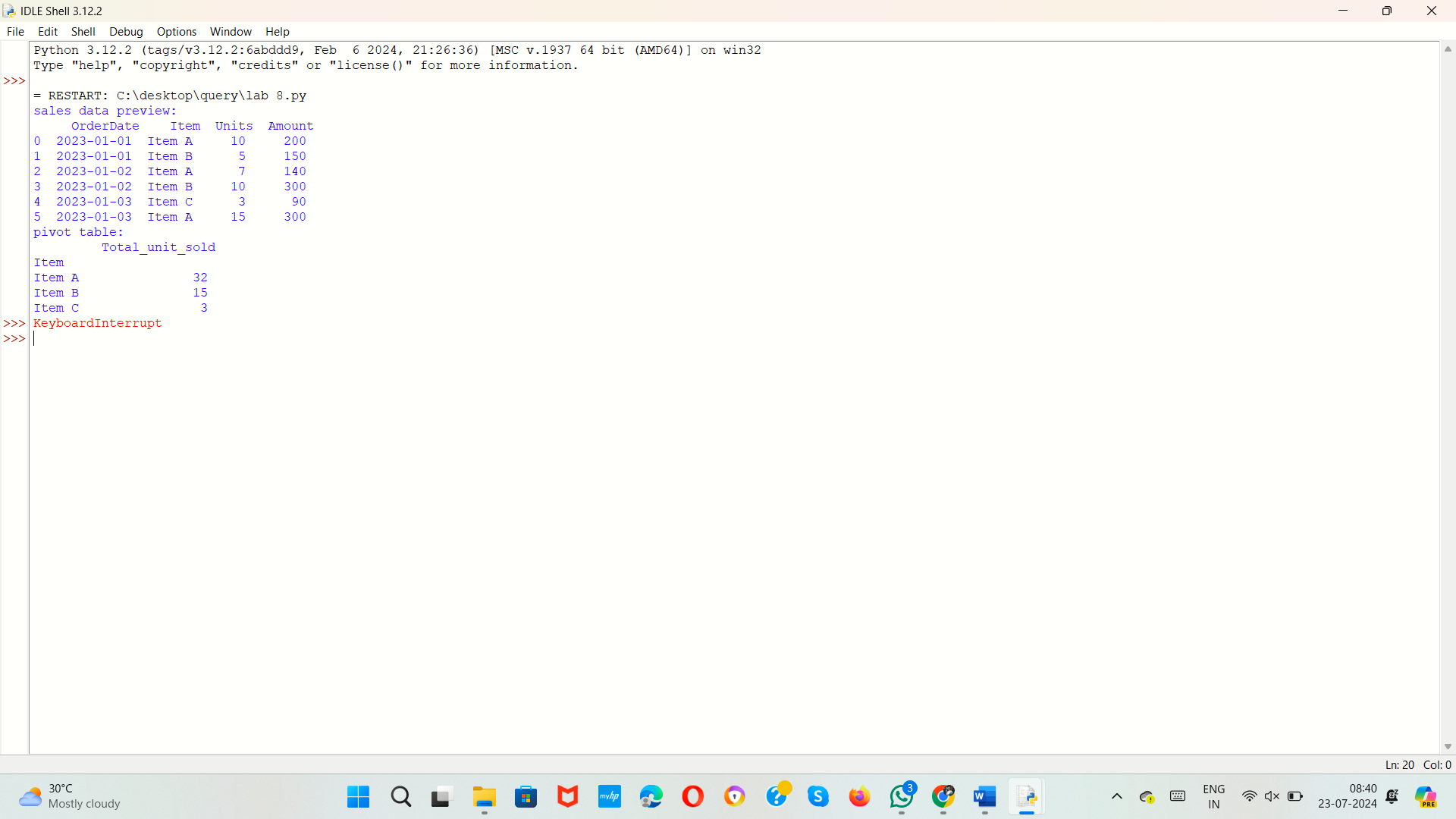
print("sales data preview:\n",df)

pivot\_table\_units=df.pivot\_table(values='Units',index='Item',aggfunc='sum')

pivot\_table\_units.columns=['Total\_unit\_sold']

print("pivot table:\n",pivot\_table\_units)

**SAMPLE 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 |

**AIM:**

To develop a Pandas program to create a Pivot table and find the total sale amount region wise, manager wise, sales man wise

**ALGORITHM:**

1. Import the `pandas` library.

2. Create a dictionary with data for order dates, regions, managers, salesmen, items, units, unit prices, and sale amounts.

3. Convert the dictionary into a DataFrame using `pd.DataFrame`.

4. Print the DataFrame.

5. Create a pivot table from the DataFrame with 'Sale\_amt' as values, ['Region', 'Manager', 'SalesMan'] as index, and using 'sum' as the aggregation function.

6. Print the pivot table for total sales by region, manager, and salesman.

**PROGRAM:**

import pandas as pd

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": [113810.00, 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]

}

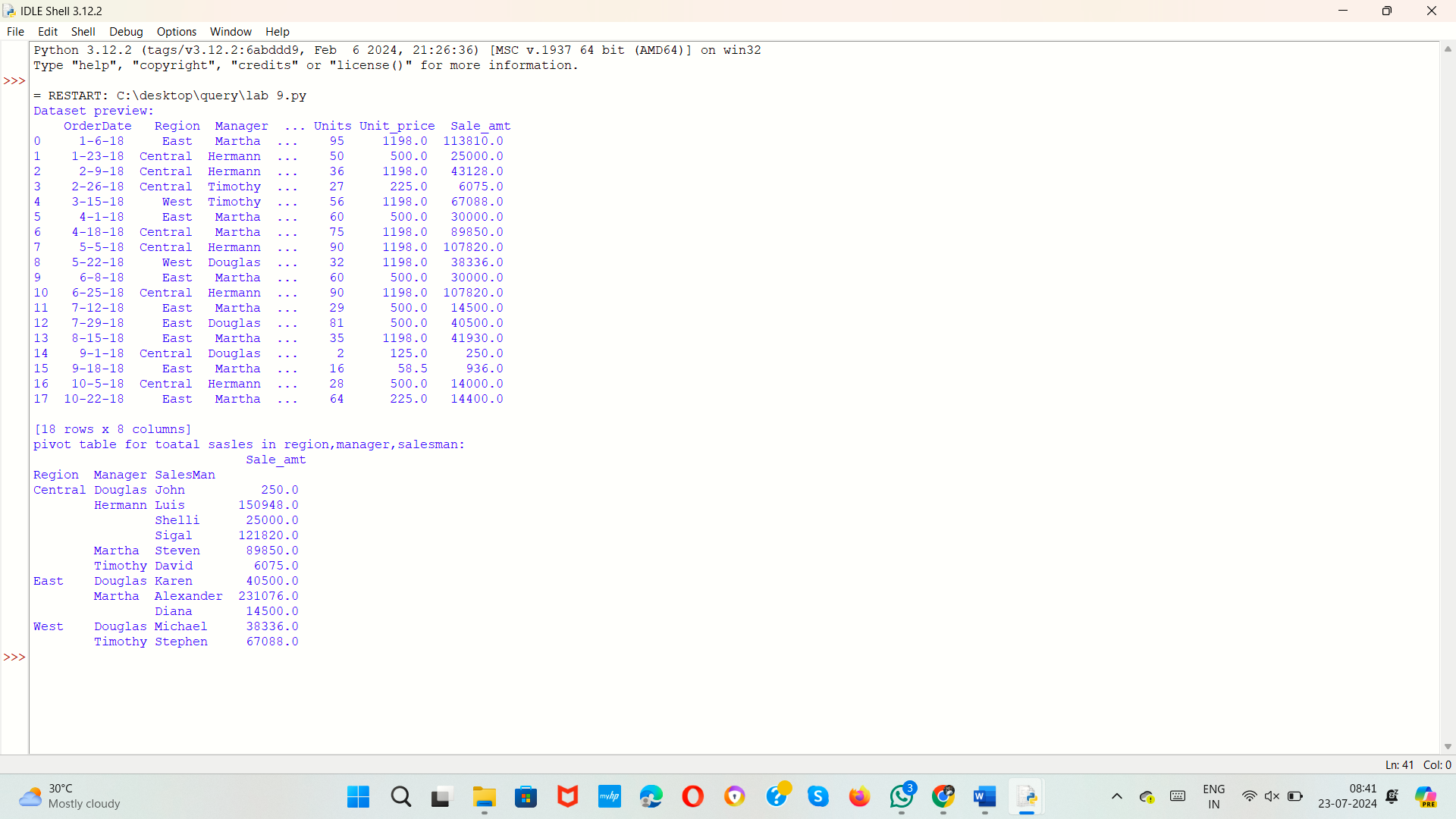
df=pd.DataFrame(data)

print("Dataset preview:\n",df)

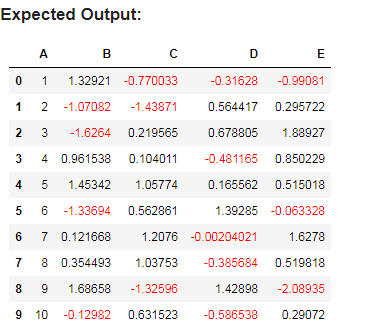
pivot\_table=df.pivot\_table(values='Sale\_amt',index=['Region','Manager','SalesMan'],aggfunc='sum')

print("pivot table for toatal sasles in region,manager,salesman:\n",pivot\_table)

**SAMPLE 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.



**AIM:**

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

**ALGORITHM:**

1. Import the `pandas` and `numpy` libraries.

2. Generate a 10x4 matrix of random numbers using `np.random.randn`.

3. Convert the matrix into a DataFrame using `pd.DataFrame` and assign column names 'A', 'B', 'C', and 'D'.

4. Print the DataFrame.

5. Define a function `highlight\_cells` that returns a background color of red if the cell value is less than 0, otherwise green.

6. Apply the `highlight\_cells` function to the DataFrame using `df.style.applymap`.

7. Display the styled DataFrame.

**PROGRAM:**

import pandas as pd

import numpy as np

data=np.random.randn(10,4)

df=pd.DataFrame(data,columns=list('ABCD'))

print("dataset preview:\n",df)

def highlight\_cells(val):

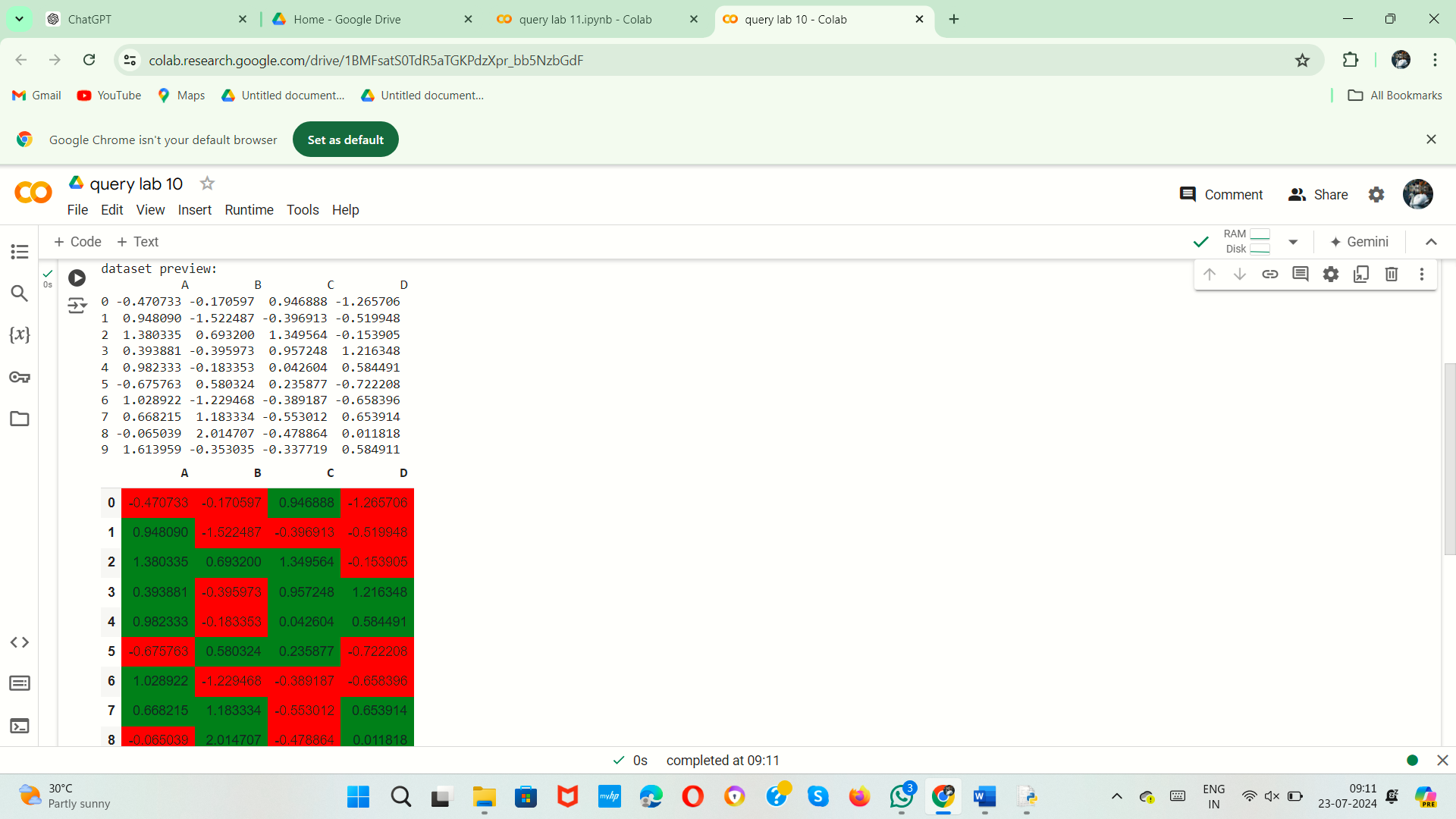
    color='background-color:red' if val<0 else 'background-color:green'

    return color

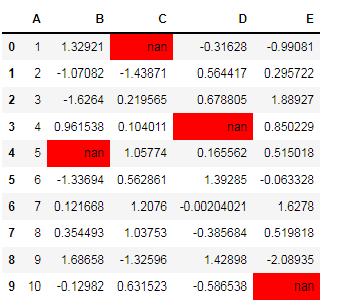
result\_df=df.style.applymap(highlight\_cells)

result\_df

**SAMPLE OUTPUT:**



11.Create a dataframe of ten rows, four columns with random values. Convert some values to nan values. Write a Pandas program which will highlight the nan values.



**AIM:**

To develop a dataframe of ten rows, four columns with random values. Convert some values to nan values. Write a Pandas program which will highlight the nan values

**ALGORITHM:**

1. Import the `pandas`, `numpy`, and `seaborn` libraries.

2. Set the random seed for reproducibility using `np.random.seed(0)`.

3. Generate a 10x4 matrix of random numbers using `np.random.randn`.

4. Convert the matrix into a DataFrame using `pd.DataFrame` and assign column names 'A', 'B', 'C', and 'D'.

5. Create a list of tuples indicating the positions of NaN values.

6. Iterate over the list of tuples and assign NaN values to the specified positions in the DataFrame.

7. Define a function `highlight\_nan` that returns a background color of yellow for NaN values and an empty string for non-NaN values.

8. Apply the `highlight\_nan` function to the DataFrame using `df.style.apply` with `axis=0`.

9. Display the styled DataFrame.

**PROGRAM:**

import pandas as pd

import numpy as np

import seaborn as sns

np.random.seed(0)

data=np.random.randn(10,4)

df=pd.DataFrame(data,columns=['A','B','C','D'])

nan\_values=[(0,0),(4,1),(7,2),(9,3)]

for i in nan\_values:

    df.iat[i[0],i[1]]=np.nan

def highlight\_nan(s):

    return['background-color:yellow' if pd.isna(v) else '' for v in s]

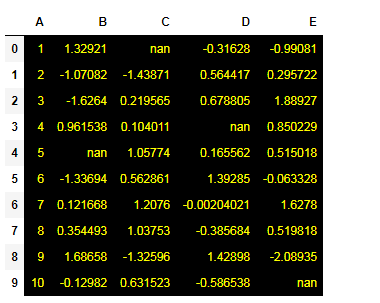
highlighted\_df=df.style.apply(highlight\_nan,axis=0)

highlighted\_df

**SAMPLE OUTPUT:**



12.Create a dataframe of ten rows, four columns with random values. Write a Pandas program to set dataframe background Color black and font color yellow.



**AIM:**

To develop a dataframe of ten rows, four columns with random values. Write a Pandas program to set dataframe background Color black and font color yellow.

**ALGORITHM:**

1. Import the `pandas` and `numpy` libraries.

2. Generate a 10x4 matrix of random numbers.

3. Convert the matrix into a DataFrame with column names.

4. Define a function to set the background color to black and the font color to yellow.

5. Apply the styling function to the entire DataFrame.

6. Display the styled DataFrame.

**PROGRAM:**

import pandas as pd

import numpy as np

# Generate a 10x4 matrix of random numbers

data = np.random.randn(10, 4)

# Convert the matrix into a DataFrame and assign column names

df = pd.DataFrame(data, columns=['A', 'B', 'C', 'D'])

# Define a function to set the background and font colors

def style\_dataframe(s):

return [

'background-color: black; color: yellow' for \_ in s

]

# Apply the style to the entire DataFrame

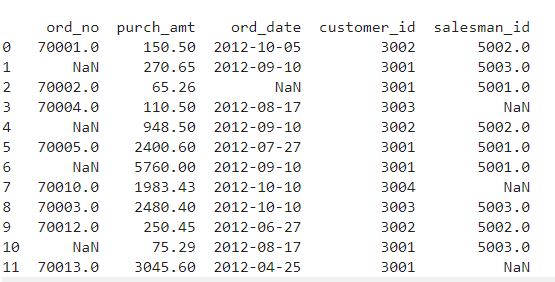
styled\_df = df.style.applymap(lambda x: 'background-color: black; color: yellow')

# Display the styled DataFrame

styled\_df

**SAMPLE OUTPUT:**

13.Write a Pandas program to detect missing values of a given DataFrame. Display True or False.



**AIM:**

To develop a Pandas program to detect missing values of a given DataFrame. Display True or False.

**ALGORITHM:**

1. Import the `pandas` and `numpy` libraries.

2. Create a dictionary with columns for order numbers, purchase amounts, order dates, customer IDs, and salesman IDs, including some `NaN` values.

3. Convert the dictionary into a DataFrame using `pd.DataFrame`.

4. Print the DataFrame to preview the dataset.

5. Identify missing values in the DataFrame using `df.isna()`.

6. Print the DataFrame showing the presence of missing values.

**PROGRAM:**

import pandas as pd

import numpy as np

data={

'ord\_no': [70001.0, np.nan, 70002.0, 70004.0, np.nan, 70005.0, np.nan, 70010.0, 70003.0, 70012.0, np.nan, 70013.0],

'purch\_amt': [150.50, 270.65, 65.26, 110.50, 948.50, 2400.60, 5760.00, 1983.43, 2480.40, 250.45, 75.29, 3045.60],

'ord\_date': ['2012-10-05', '2012-09-10', np.nan, '2012-08-17', '2012-09-10', '2012-07-27', '2012-09-10', '2012-10-10', '2012-10-10', '2012-06-27', '2012-08-17', '2012-04-25'],

'customer\_id': [3002, 3001, 3001, 3003, 3002, 3001, 3001, 3004, 3003, 3002, 3001, 3001],

'salesman\_id': [5002.0, 5003.0, 5001.0, np.nan, 5002.0, 5001.0, 5001.0, np.nan, 5003.0, 5002.0, 5003.0, np.nan]

}

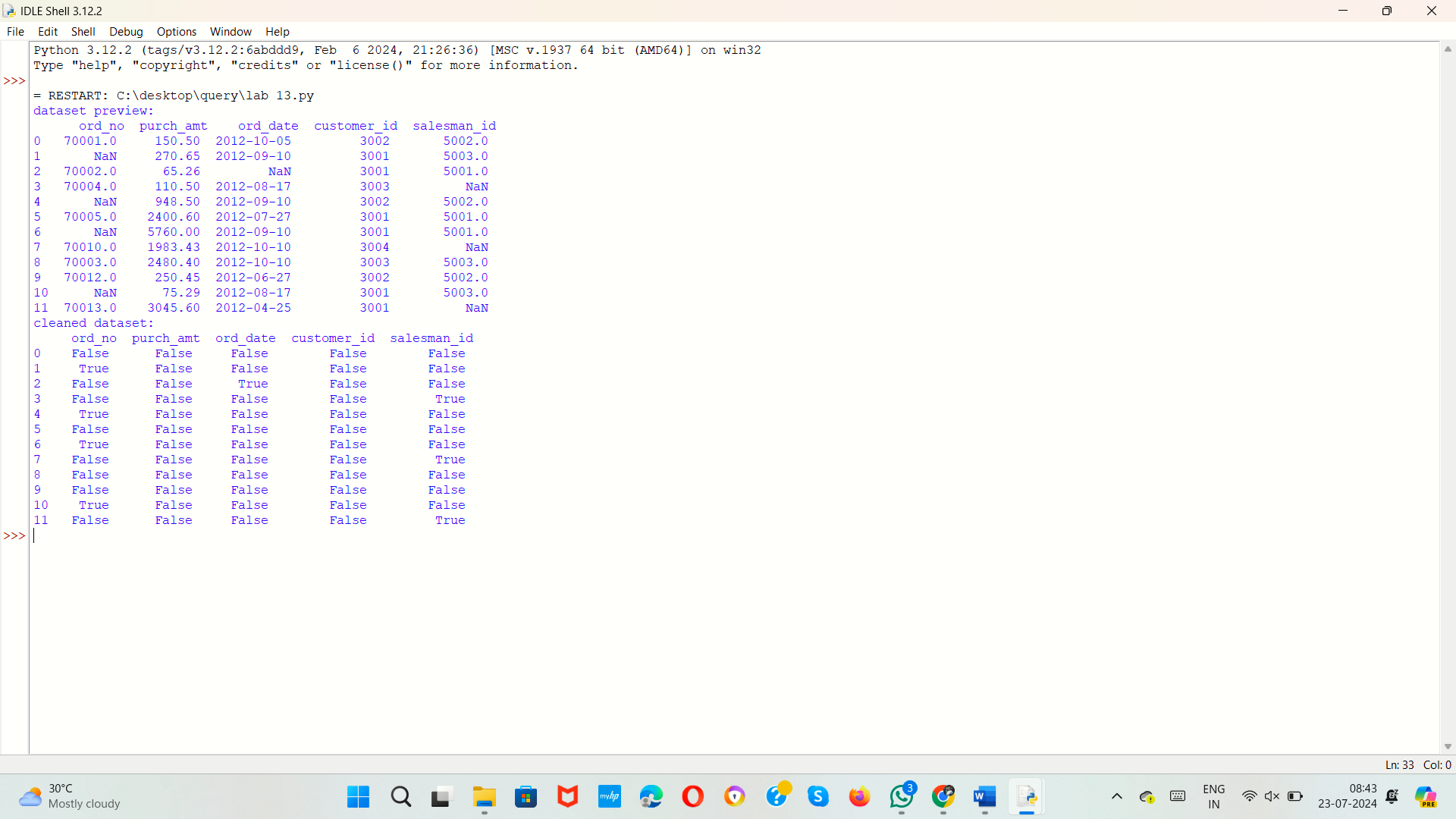
df=pd.DataFrame(data)

missing\_values=df.isna()

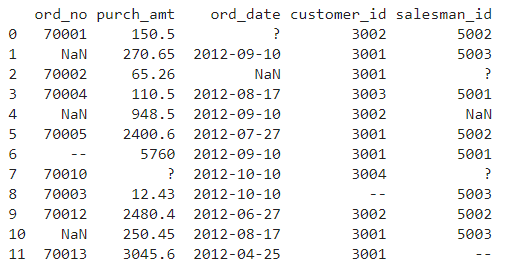
print("dataset preview:\n",df)

print("cleaned dataset:\n",missing\_values)

**SAMPLE OUTPUT:**



14. Write a Pandas program to find and replace the missing values in a given DataFrame which do not have any valuable information.



**AIM:**

To develop a Pandas program to find and replace the missing values in a given DataFrame which do not have any valuable information.

**ALGORITHM:**

1. Import the `pandas` and `numpy` libraries.

2. Create a dictionary with columns for order numbers, purchase amounts, order dates, customer IDs, and salesman IDs, including some `NaN` values.

3. Convert the dictionary into a DataFrame using `pd.DataFrame`.

4. Print the DataFrame to preview the dataset.

5. Fill missing values in the 'ord\_no' column with -1.

6. Fill missing values in the 'salesman\_id' column with -1.

7. Fill missing values in the 'ord\_date' column with '1900-01-01'.

8. Calculate the mean of the 'purch\_amt' column and use it to fill missing values in the 'purch\_amt' column.

9. Print the cleaned DataFrame.

**PROGRAM:**

import pandas as pd

import numpy as np

data={

'ord\_no': [70001.0, np.nan, 70002.0, 70004.0, np.nan, 70005.0, np.nan, 70010.0, 70003.0, 70012.0, np.nan, 70013.0],

'purch\_amt': [150.50, 270.65, 65.26, 110.50, 948.50, 2400.60, 5760.00, 1983.43, 2480.40, 250.45, 75.29, 3045.60],

'ord\_date': ['2012-10-05', '2012-09-10', np.nan, '2012-08-17', '2012-09-10', '2012-07-27', '2012-09-10', '2012-10-10', '2012-10-10', '2012-06-27', '2012-08-17', '2012-04-25'],

'customer\_id': [3002, 3001, 3001, 3003, 3002, 3001, 3001, 3004, 3003, 3002, 3001, 3001],

'salesman\_id': [5002.0, 5003.0, 5001.0, np.nan, 5002.0, 5001.0, 5001.0, np.nan, 5003.0, 5002.0, 5003.0, np.nan]

}

df=pd.DataFrame(data)

print("dataset preview:\n",df)

df['ord\_no']=df['ord\_no'].fillna(-1)

df['salesman\_id']=df['salesman\_id'].fillna(-1)

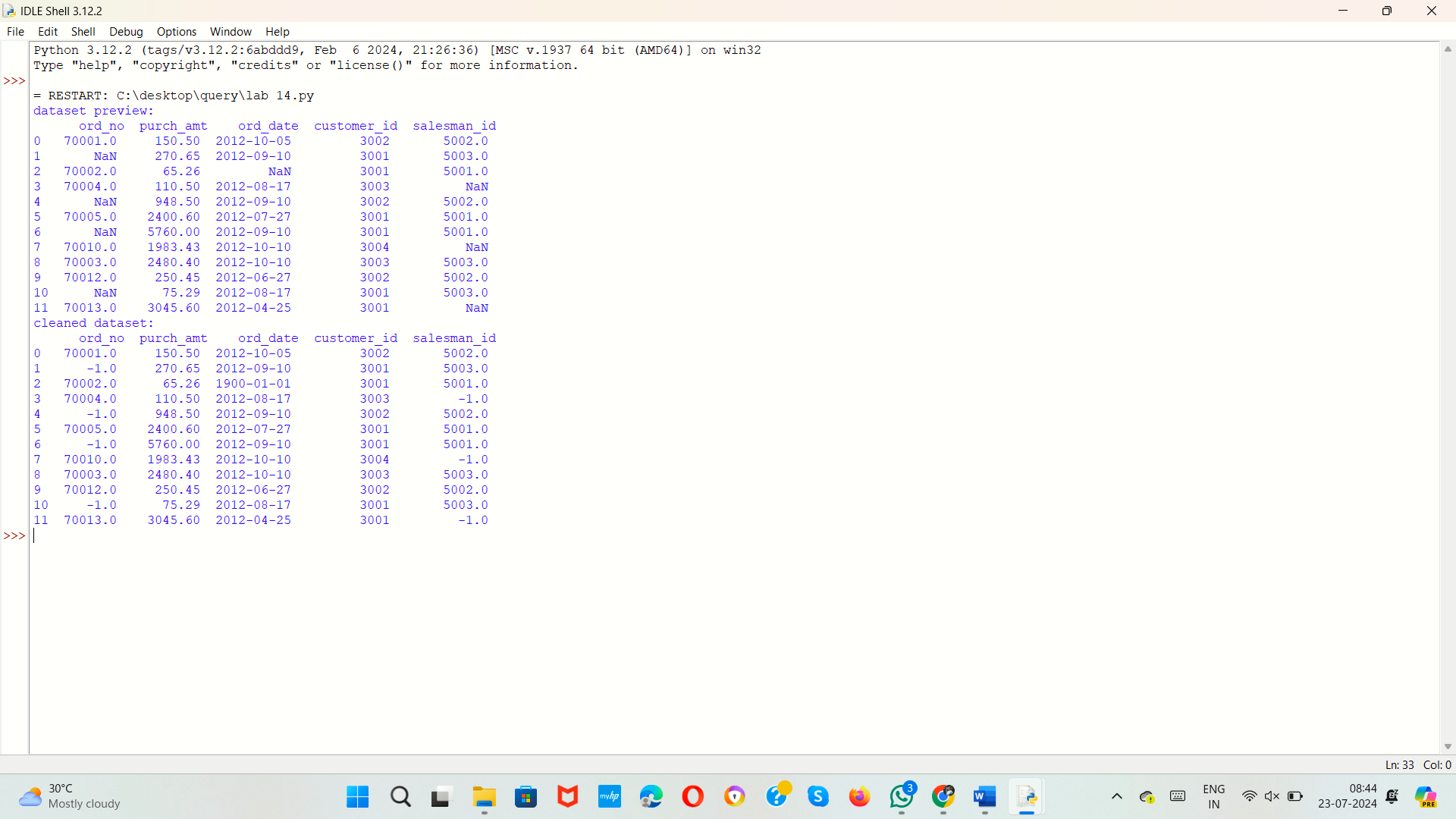
df['ord\_date']=df['ord\_date'].fillna('1900-01-01')

mean\_purch\_amt=df['purch\_amt'].mean()

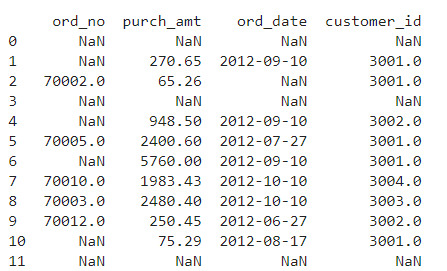
df['purch\_amt']=df['purch\_amt'].fillna(mean\_purch\_amt)

print("cleaned dataset:\n",df)

**SAMPLE OUTPUT:**



15.Write a Pandas program to keep the rows with at least 2 NaN values in a given DataFrame.



**AIM:**

To develop a Pandas program to keep the rows with at least 2 NaN values in a given DataFrame.

**ALGORITHM:**

1. Import the `numpy` and `pandas` libraries.

2. Create a dictionary with columns for order numbers, purchase amounts, order dates, customer IDs, and salesman IDs, including some `NaN` values.

3. Convert the dictionary into a DataFrame using `pd.DataFrame`.

4. Print the DataFrame to preview the dataset.

5. Add a new column 'NaN\_count' that counts the number of `NaN` values in each row.

6. Filter the DataFrame to keep only rows where 'NaN\_count' is greater than or equal to 2.

7. Drop the 'NaN\_count' column from the filtered DataFrame.

8. Print the resulting DataFrame with rows that have at least 2 `NaN` values.

**PROGRAM:**

import numpy as np

import pandas as pd

data = {

'ord\_no': [70001.0, np.nan, 70002.0, 70004.0, np.nan, 70005.0, np.nan, 70010.0, 70003.0, 70012.0, np.nan, 70013.0],

'purch\_amt': [150.50, 270.65, 65.26, 110.50, 948.50, 2400.60, 5760.00, 1983.43, 2480.40, 250.45, 75.29, 3045.60],

'ord\_date': ['2012-10-05', '2012-09-10', np.nan, '2012-08-17', '2012-09-10', '2012-07-27', '2012-09-10', '2012-10-10', '2012-10-10', '2012-06-27', '2012-08-17', '2012-04-25'],

'customer\_id': [3002, 3001, 3001, 3003, 3002, 3001, 3001, 3004, 3003, 3002, 3001, 3001],

'salesman\_id': [5002.0, 5003.0, 5001.0, np.nan, 5002.0, 5001.0, 5001.0, np.nan, 5003.0, 5002.0, 5003.0, np.nan]

}

df = pd.DataFrame(data)

print("Dataset preview:\n", df)

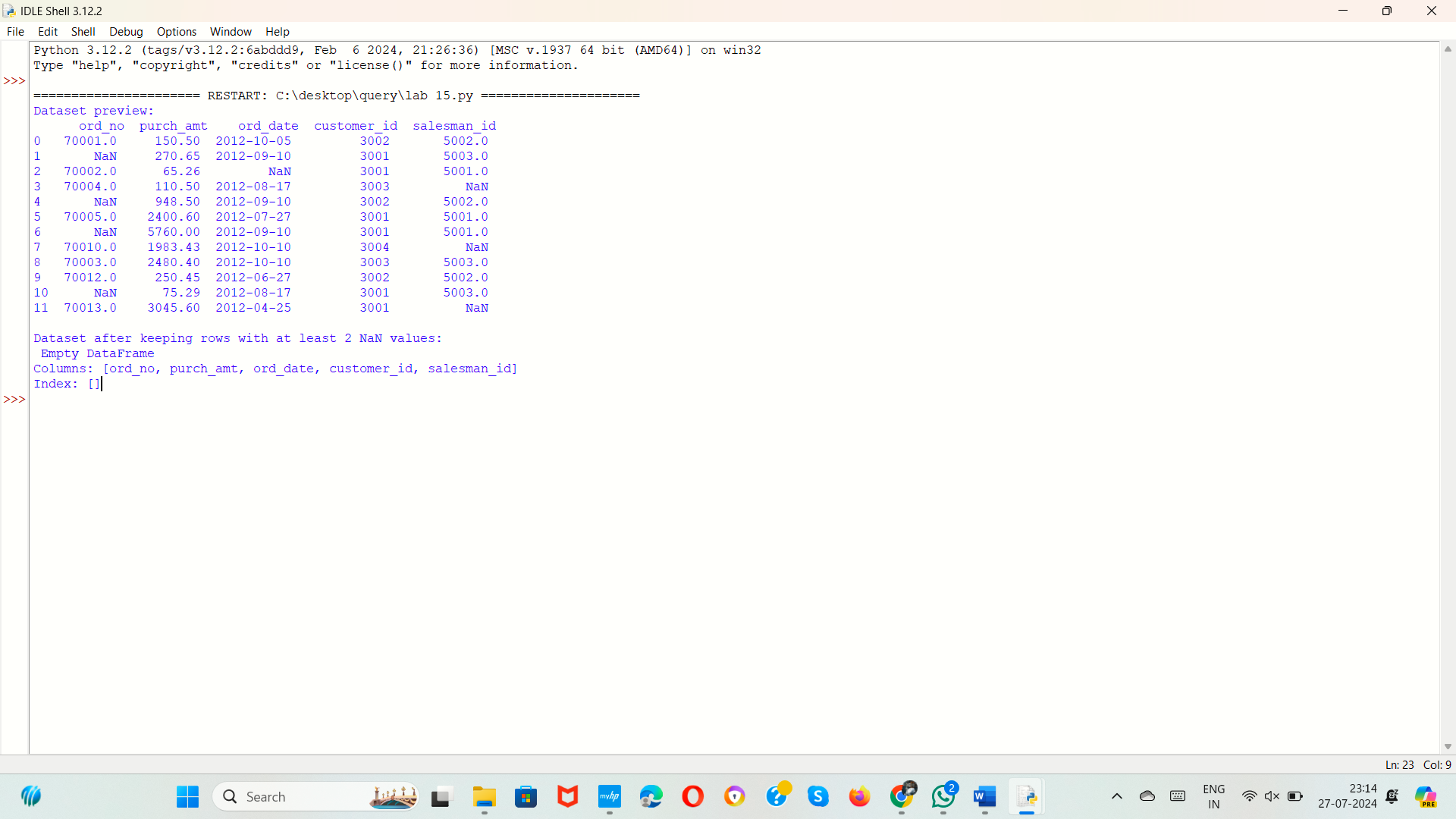
df['NaN\_count'] = df.isna().sum(axis=1)

rows\_with\_nan = df[df['NaN\_count'] >= 2]

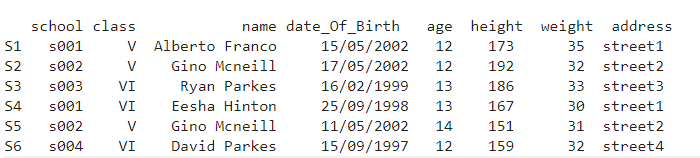
rows\_with\_nan = rows\_with\_nan.drop(columns=['NaN\_count'])

print("\nDataset after keeping rows with at least 2 NaN values:\n", rows\_with\_nan)

**SAMPLE OUTPUT:**



16.Write a Pandas program to split the following dataframe into groups based on school code. Also check the type of GroupBy object.



**AIM:**

To develop a pandas program to split the following dataframe into groups based on school code.

**ALGORITHM:**

**PROGRAM:**

import pandas as pd

import numpy as np

data = {

'school': ['S1', 'S2', 'S3', 'S4', 'S5', 'S6'],

'class': ['s001', 's002', 's003', 's001', '5002', 's004'],

'name': ['Alberto Franco', 'Gino Mcneill', 'Ryan Parkes', 'Eesha Hinton', 'Gino Mcneill', 'David Parkes'],

'date\_Of\_Birth': ['15/05/2002', '17/05/2002', '16/02/1999', '25/09/1998', '11/05/2002', '15/09/1997'],

'age': [12, 12, 13, 13, 14, 12],

'height': [173, 192, 186, 167, 151, 159],

'weight': [35, 32, 33, 30, 31, 32],

'address': ['street1', 'street2', 'street3', 'street1', 'street2', 'street4']

}

df=pd.DataFrame(data)

print("original dataset preview:\n",df)

grouped=df.groupby('school')

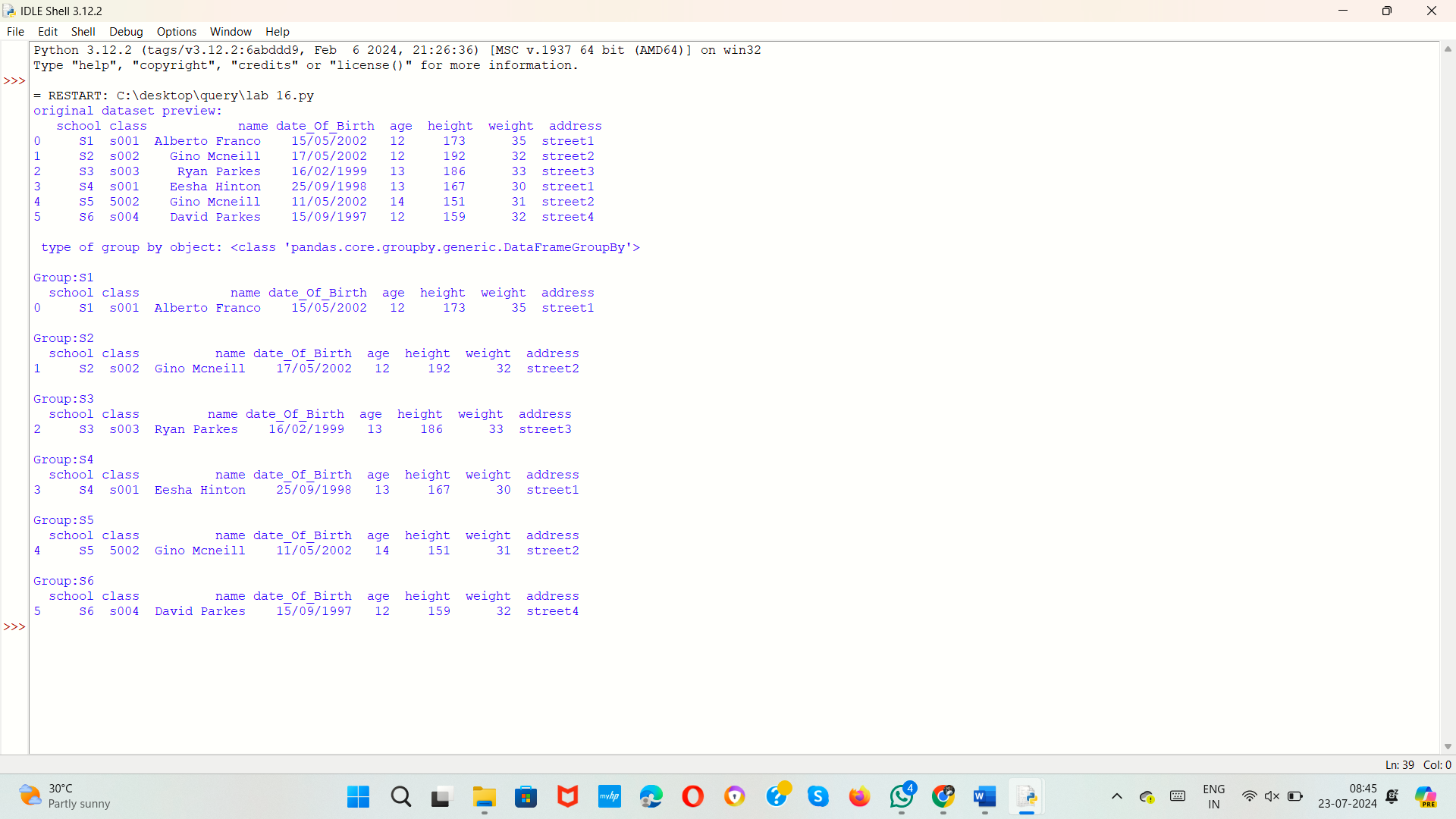
print("\n type of group by object:",type(grouped))

for name,group in grouped:

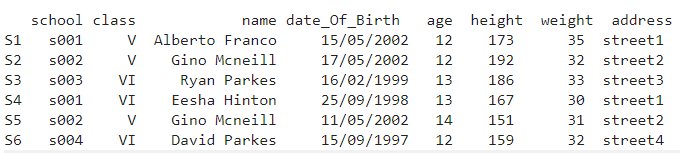
print(f"\nGroup:{name}")

print(group)

**SAMPLE OUTPUT:**



17.Write a Pandas program to split the following dataframe by school code and get mean, min, and max value of age for each school.



**AIM:**

To develop a Pandas program to split the following dataframe by school code and get mean, min, and max value of age for each school.

**ALGORITHM:**

1. Import the `pandas` and `numpy` libraries.

2. Create a dictionary with columns for school, class, name, date of birth, age, height, weight, and address.

3. Convert the dictionary into a DataFrame using `pd.DataFrame`.

4. Print the DataFrame to preview the dataset.

5. Group the DataFrame by the 'school' column and aggregate the 'age' column to calculate the mean, minimum, and maximum values.

6. Reset the index of the resulting DataFrame to convert the grouped data into a regular DataFrame.

7. Print the DataFrame showing the mean, minimum, and maximum age for each school.

**PROGRAM:**

import pandas as pd

import numpy as np

data = {

'school': ['S1', 'S2', 'S3', 'S4', 'S5', 'S6'],

'class': ['s001', 's002', 's003', 's001', '5002', 's004'],

'name': ['Alberto Franco', 'Gino Mcneill', 'Ryan Parkes', 'Eesha Hinton', 'Gino Mcneill', 'David Parkes'],

'date\_Of\_Birth': ['15/05/2002', '17/05/2002', '16/02/1999', '25/09/1998', '11/05/2002', '15/09/1997'],

'age': [12, 12, 13, 13, 14, 12],

'height': [173, 192, 186, 167, 151, 159],

'weight': [35, 32, 33, 30, 31, 32],

'address': ['street1', 'street2', 'street3', 'street1', 'street2', 'street4']

}

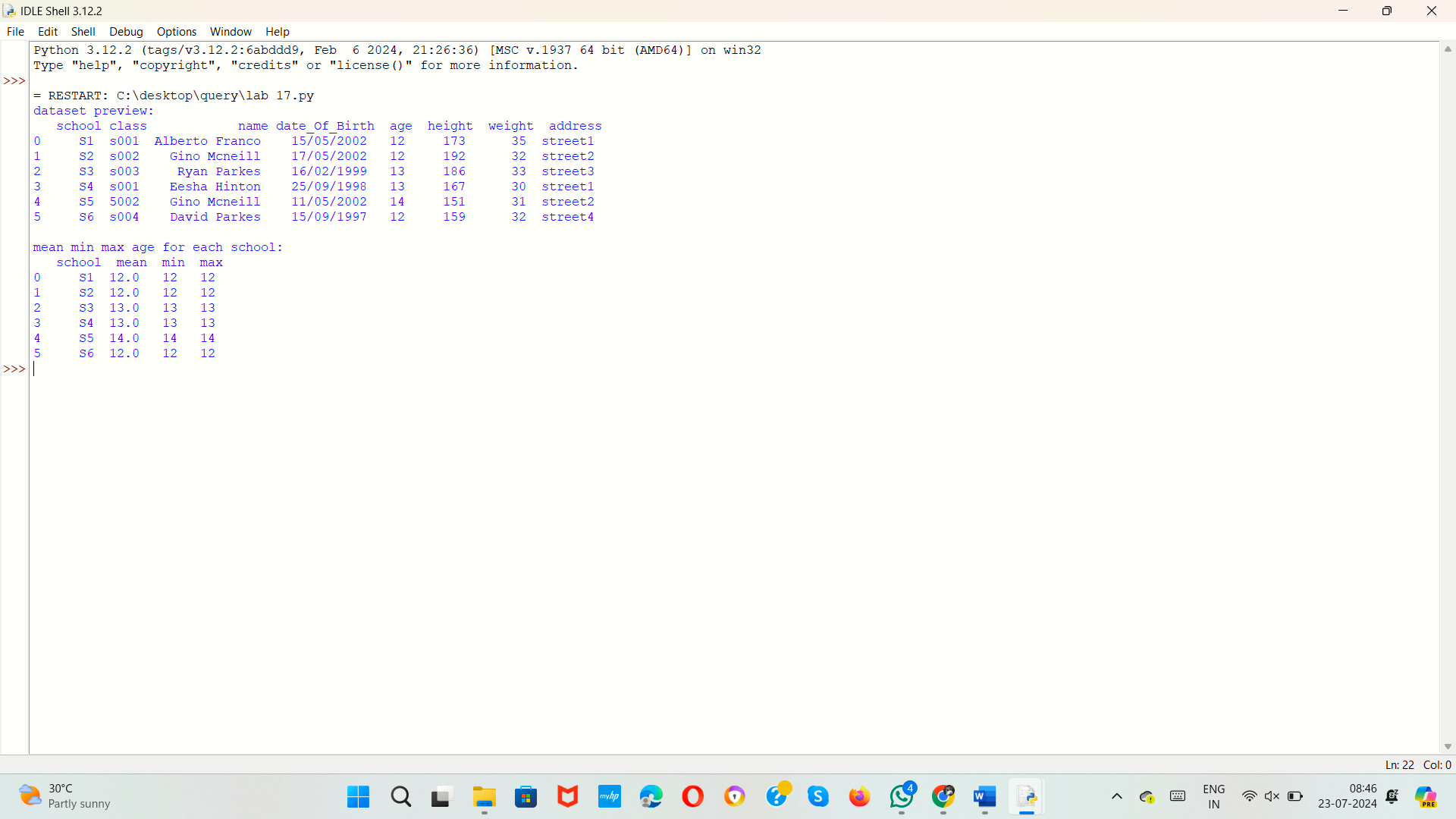
df=pd.DataFrame(data)

print("dataset preview:\n",df)

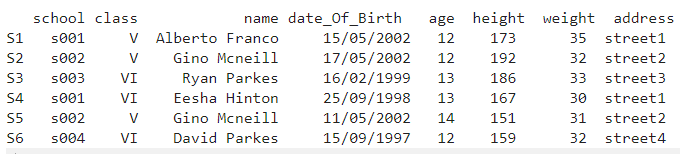
grouped=df.groupby('school')['age'].agg(['mean','min','max']).reset\_index()

print("\nmean min max age for each school:\n",grouped)

**SAMPLE OUTPUT:**



18.Write a Pandas program to split the following given dataframe into groups based on school code and class.



**AIM:**

To develop a Pandas program to split the following given dataframe into groups based on school code and class.

**ALGORITHM:**

1. Import the `pandas` library.

2. Create a dictionary with columns for school, class, name, date of birth, age, height, weight, and address.

3. Convert the dictionary into a DataFrame using `pd.DataFrame`.

4. Print the original DataFrame.

5. Group the DataFrame by both 'school' and 'class' columns using `groupby`.

6. Iterate through each group in the grouped DataFrame, printing the name of the group and the corresponding group DataFrame.

**PROGRAM:**

import pandas as pd

data = {

'school': ['S1', 'S2', 'S3', 'S4', 'S5', 'S6'],

'class': ['s001', 's002', 's003', 's001', '5002', 's004'],

'name': ['Alberto Franco', 'Gino Mcneill', 'Ryan Parkes', 'Eesha Hinton', 'Gino Mcneill', 'David Parkes'],

'date\_Of\_Birth': ['15/05/2002', '17/05/2002', '16/02/1999', '25/09/1998', '11/05/2002', '15/09/1997'],

'age': [12, 12, 13, 13, 14, 12],

'height': [173, 192, 186, 167, 151, 159],

'weight': [35, 32, 33, 30, 31, 32],

'address': ['street1', 'street2', 'street3', 'street1', 'street2', 'street4']

}

df=pd.DataFrame(data)

print("original dataframe:\n",df)

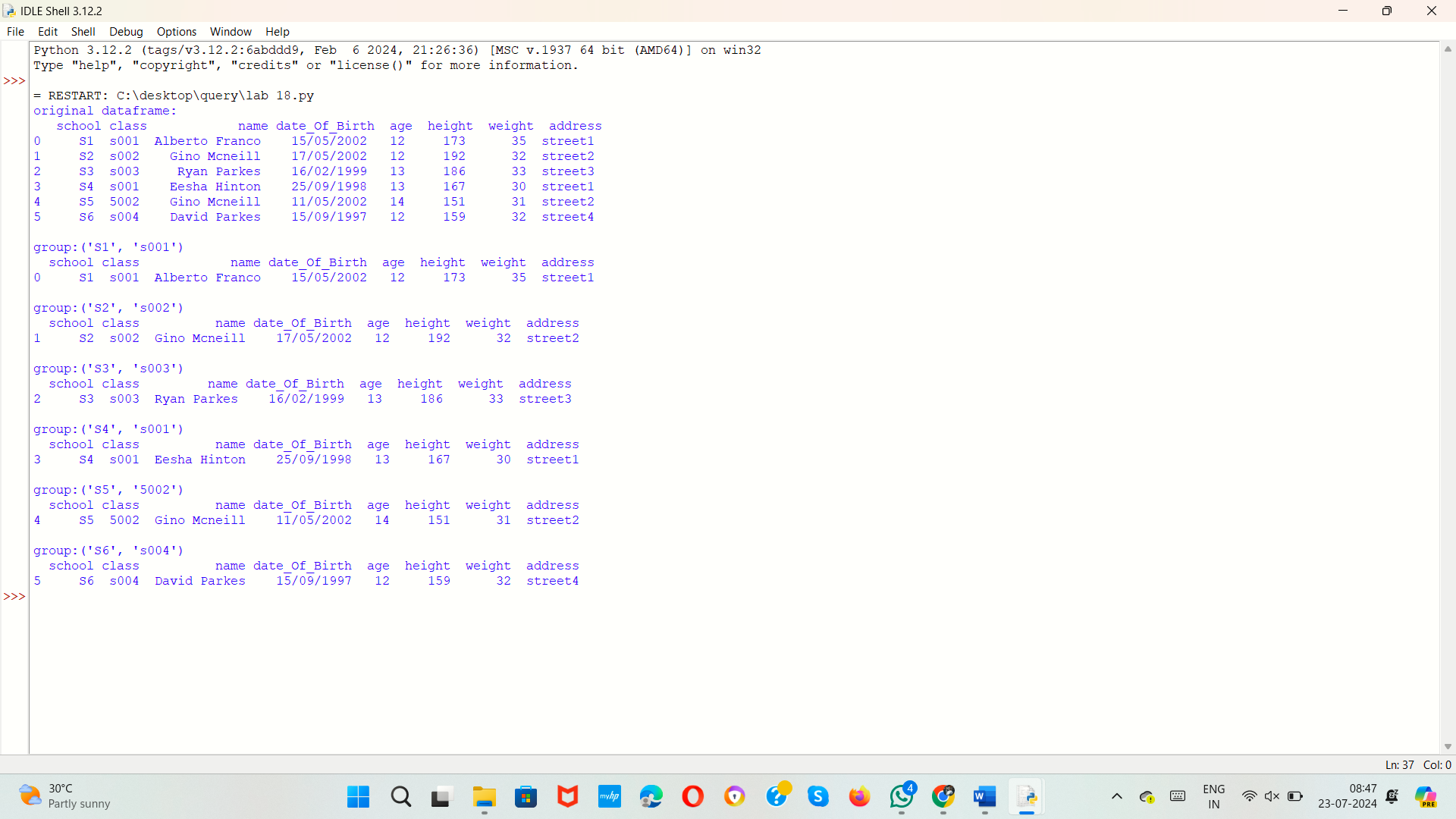
grouped=df.groupby(['school','class'])

for name,group in grouped:

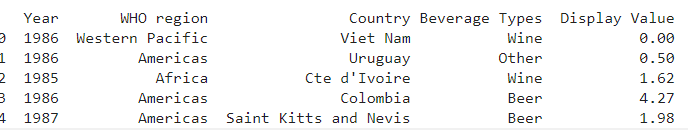
print(f"\ngroup:{name}")

print(group)

**SAMPLE OUTPUT:**



19.Write a Pandas program to display the dimensions or shape of the World alcohol consumption dataset. Also extract the column names from the dataset.



**AIM:**

To develop a pandas program to display the dimensions or shape of the World alcohol consumption dataset. Also extract the column names from the dataset

**ALGORITHM:**

**PROGRAM:**

import pandas as pd

data = {

'school': ['S1', 'S2', 'S3', 'S4', 'S5', 'S6'],

'class': ['s001', 's002', 's003', 's001', '5002', 's004'],

'name': ['Alberto Franco', 'Gino Mcneill', 'Ryan Parkes', 'Eesha Hinton', 'Gino Mcneill', 'David Parkes'],

'date\_Of\_Birth': ['15/05/2002', '17/05/2002', '16/02/1999', '25/09/1998', '11/05/2002', '15/09/1997'],

'age': [12, 12, 13, 13, 14, 12],

'height': [173, 192, 186, 167, 151, 159],

'weight': [35, 32, 33, 30, 31, 32],

'address': ['street1', 'street2', 'street3', 'street1', 'street2', 'street4']

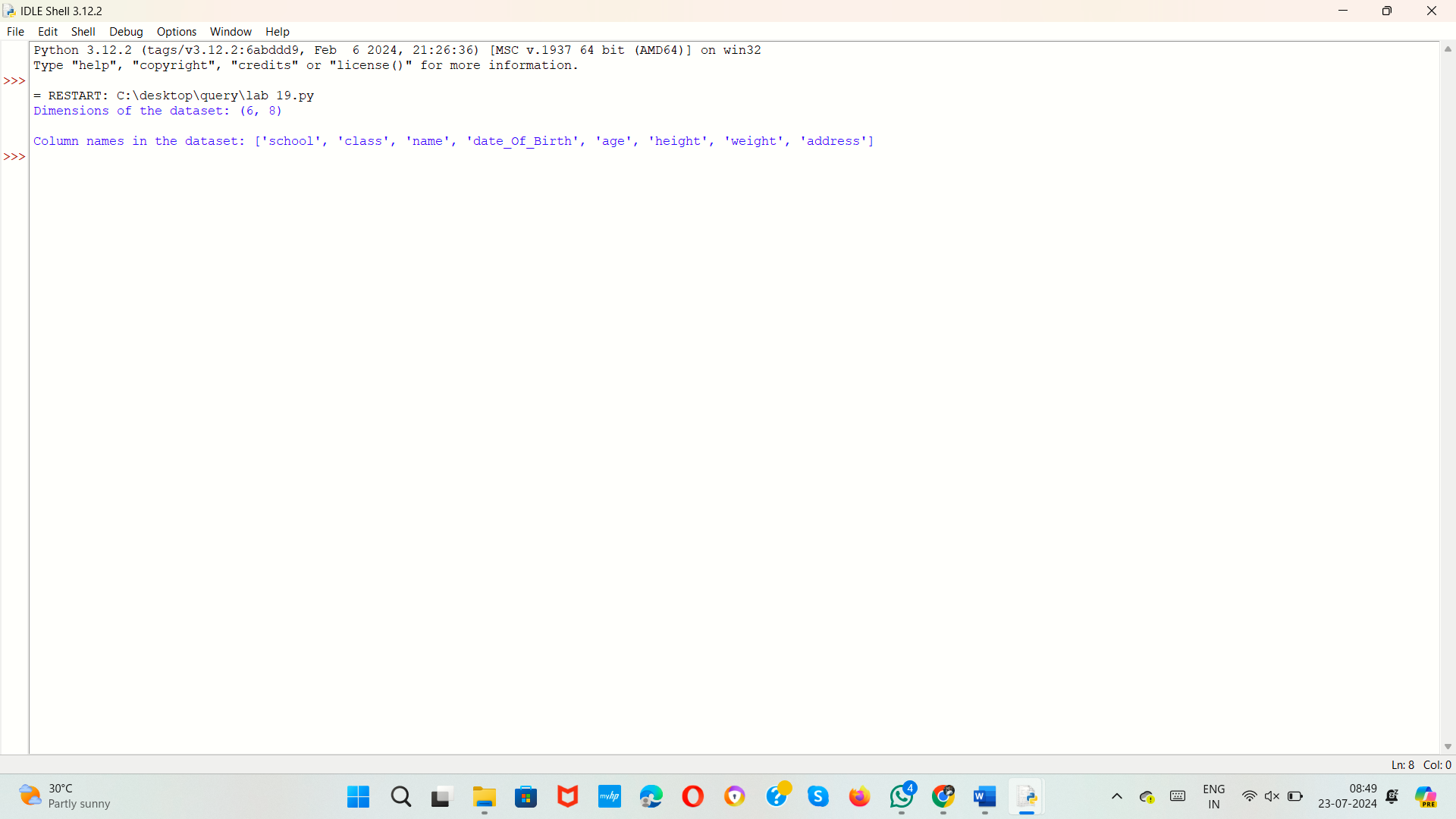
}

df = pd.DataFrame(data)

print("Dimensions of the dataset:", df.shape)

print("\nColumn names in the dataset:", df.columns.tolist())

**SAMPLE OUTPUT:**



20.Write a Pandas program to find the index of a given substring of a DataFrame column.

**AIM:**

To develop a Pandas program to find the index of a given substring of a DataFrame column.

**ALGORITHM:**

1. Import the `pandas` library.

2. Create a dictionary with columns for names and addresses.

3. Convert the dictionary into a DataFrame using `pd.DataFrame`.

4. Prompt the user to enter a substring to search for in the 'name' column.

5. Use the `str.contains` method to find rows in the 'name' column that contain the substring (case-insensitive) and get their indices.

6. Check if the indices are not empty.

- If not empty, print the indices of the rows containing the substring.

- If empty, print a message indicating that no rows were found containing the substring.

**PROGRAM:**

import pandas as pd

data = {

'name': ['Alberto Franco', 'Gino Mcneill', 'Ryan Parkes', 'Eesha Hinton', 'Gino Mcneill', 'David Parkes'],

'address': ['street1', 'street2', 'street3', 'street1', 'street2', 'street4']

}

df = pd.DataFrame(data)

substring = input("Enter the substring to search for in the 'name' column: ")

indices = df[df['name'].str.contains(substring, case=False, na=False)].index

if not indices.empty:

print(f"Indices of rows containing the substring '{substring}':", list(indices))

else:

print(f"No rows found containing the substring '{substring}'.")

**SAMPLE OUTPUT:**

