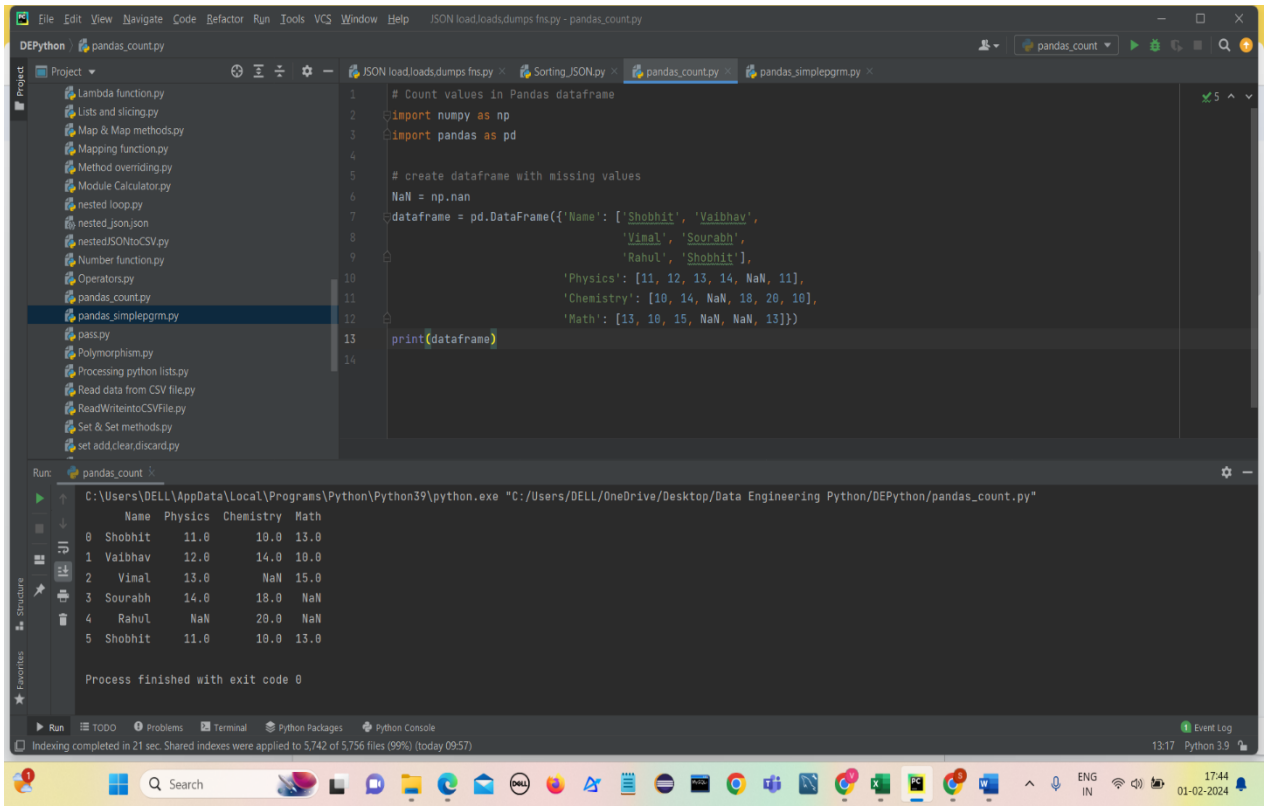
Day-10

Count values in Pandas Data Frame:

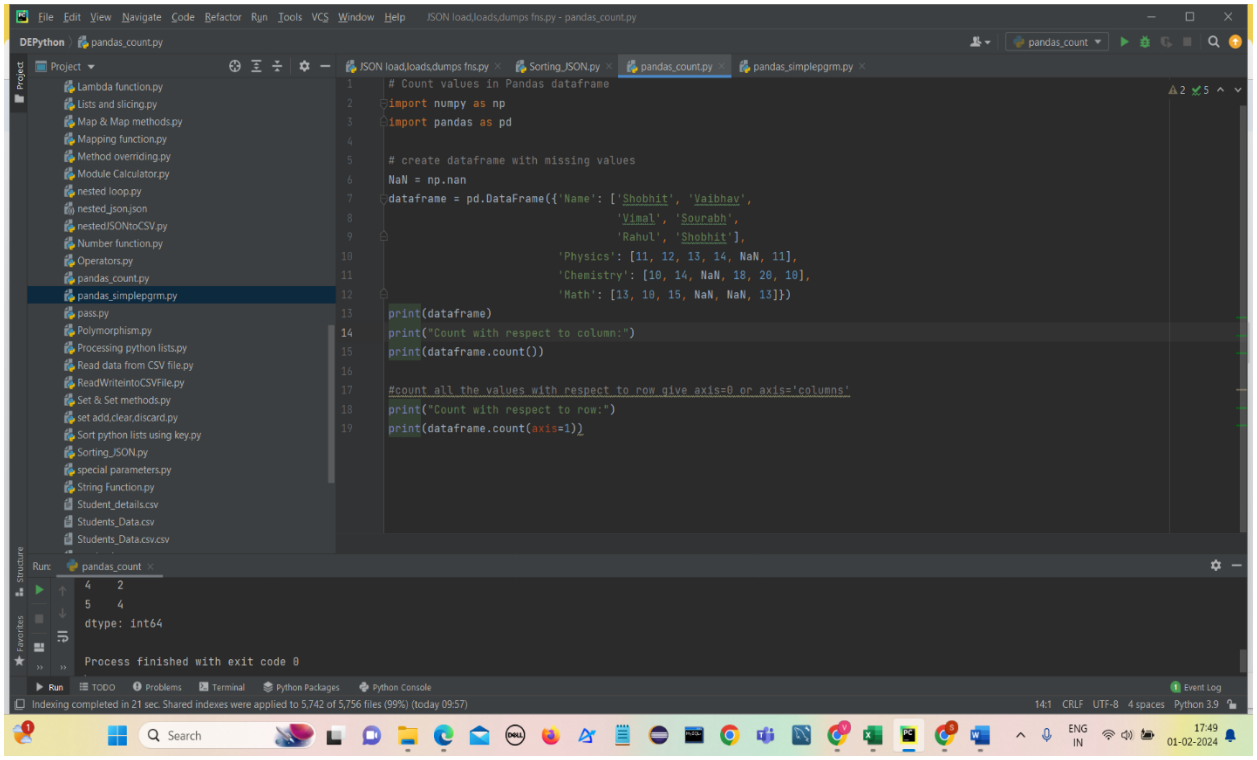
Syntax: DataFrame.count(axis=0, level=None, numeric_only=False)
Returns: It returns count of non-null values and if level is used it returns dataframe.

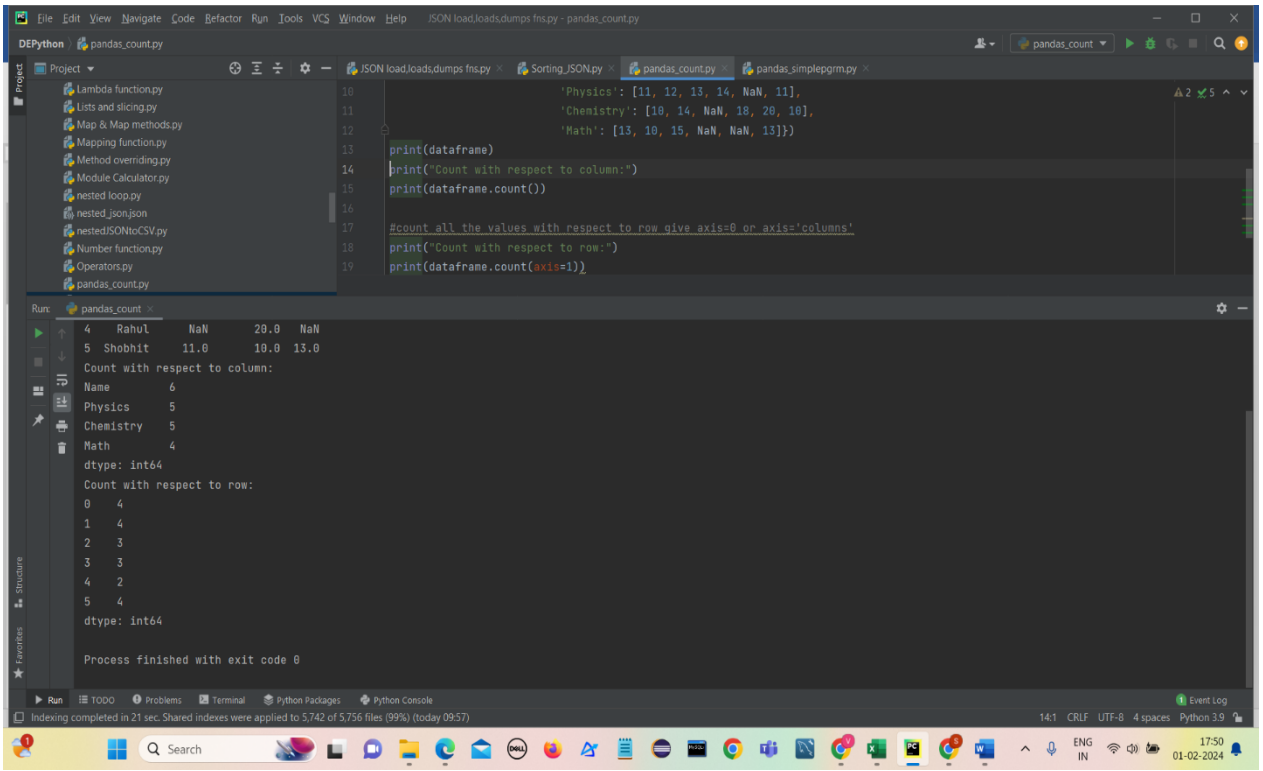
Steps:

1) Importing libraries
2) Create dataframe with missing values
3) use the **.count() function** to count all the values of different columns.
4) By default axis is 0 or index where counts are generated for each column.
5) If we want to count all the values with respect to row then we have to pass axis=1 or ‘columns’.
6) count null values in our dataframe.
7) Count based on some condition

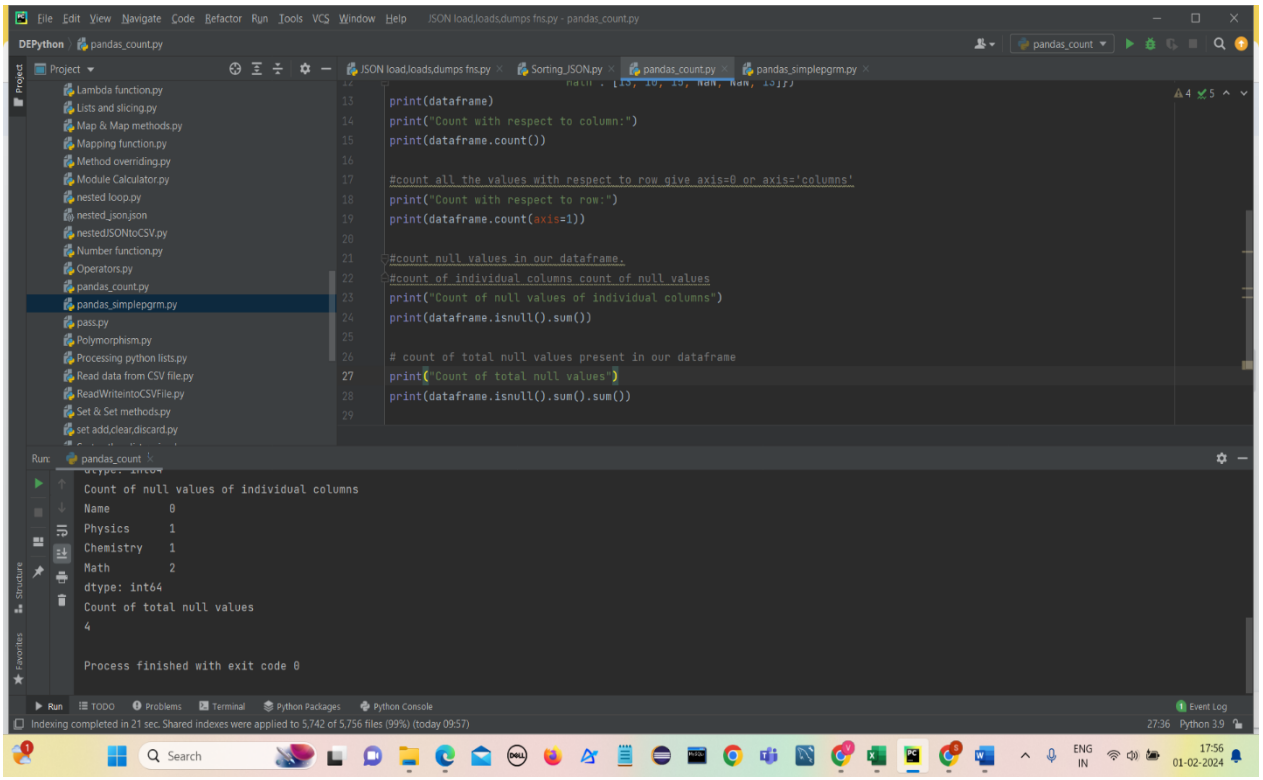


Count with respect to rows and columns

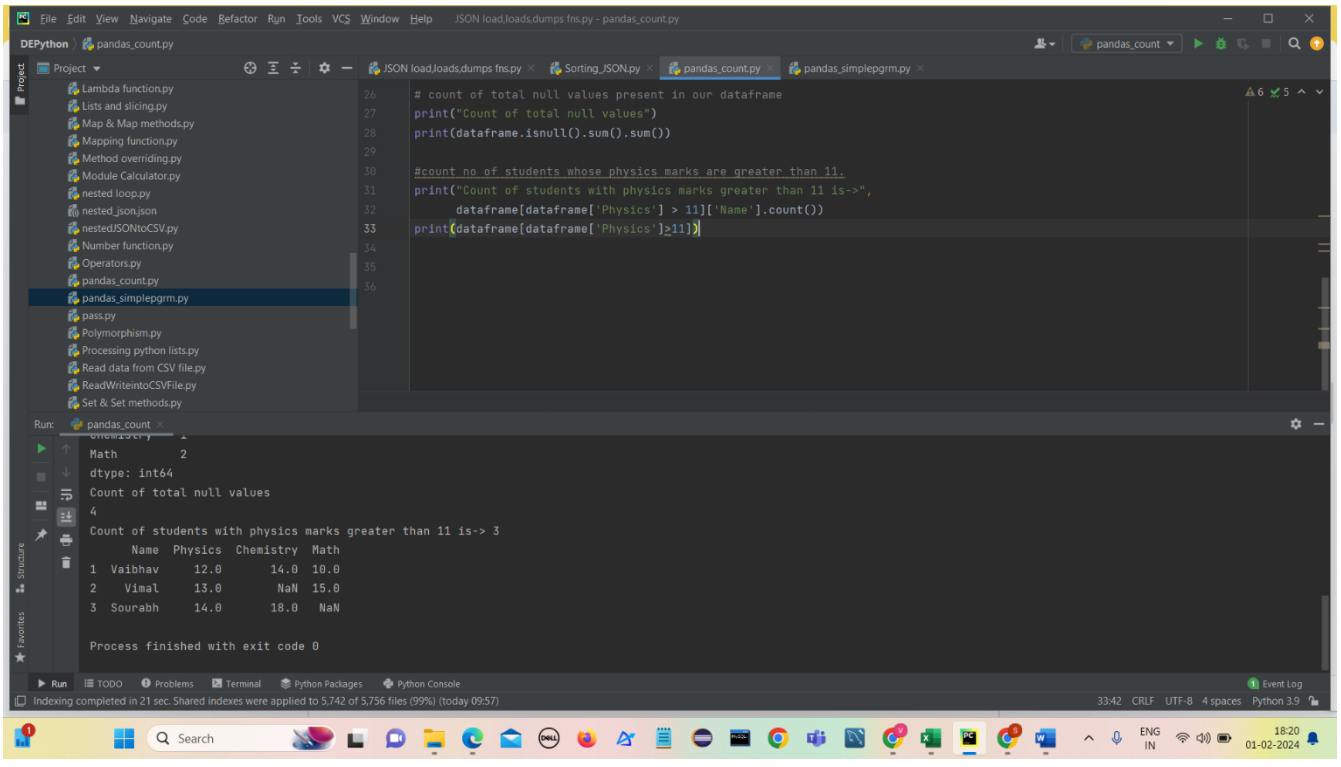




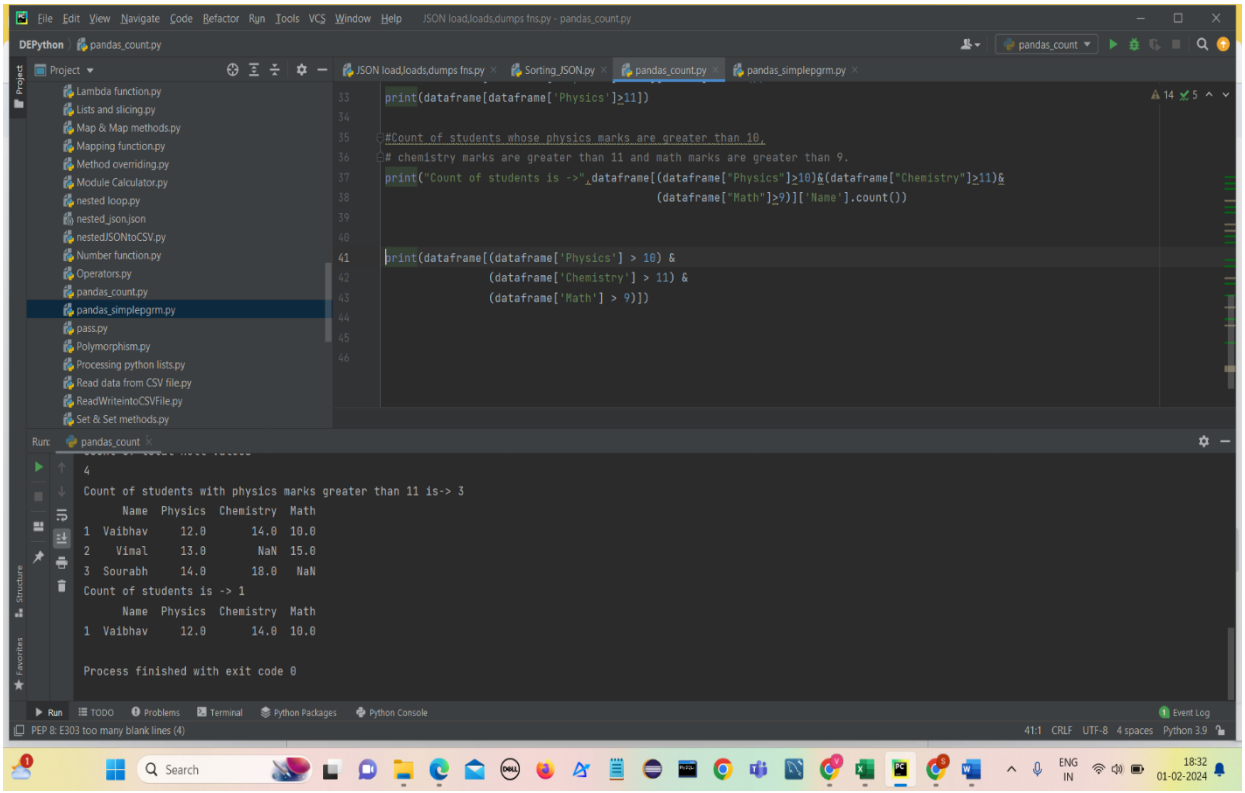
Count of null values in individual columns and total count of null values



count no of students whose physics marks are greater than 11.



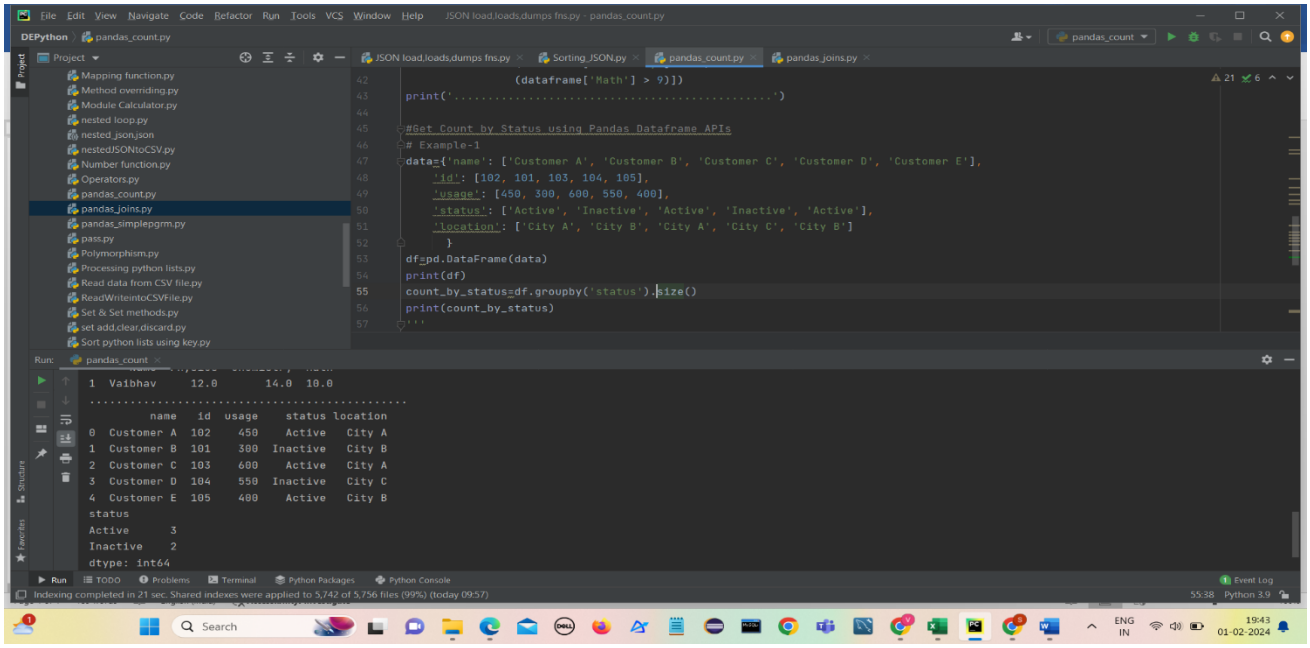
Count of students whose physics marks are greater than 10,chemistry marks are greater than 11 and math marks are greater than 9.



Get Count by Status using Pandas Dataframe APIs

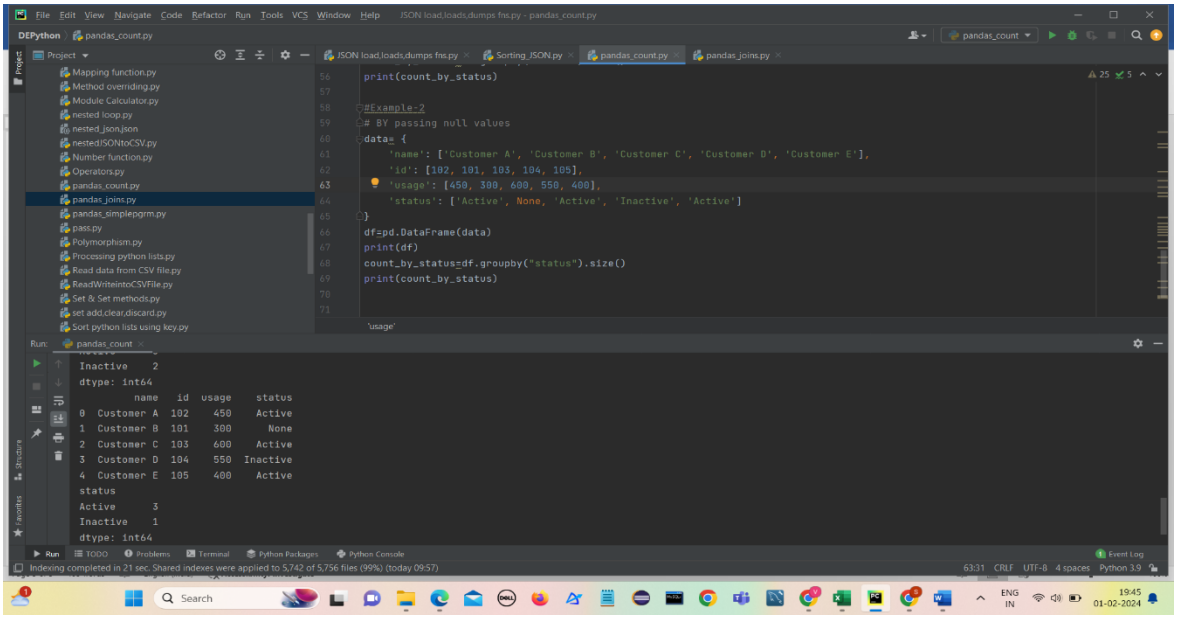
1. Create a DataFrame .
2. Use the **groupby()** function to group the data by the 'status' column.
3. Use the **size()** function to get the count of each group.

Example-1:



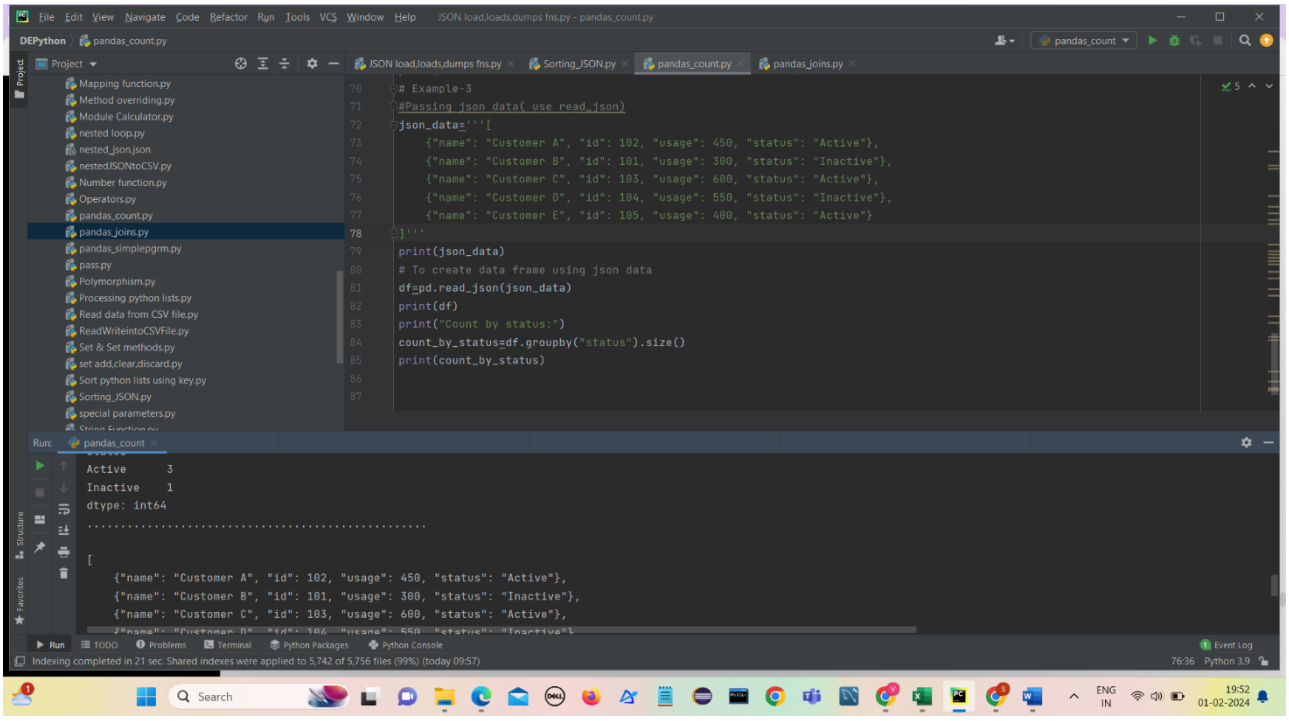
Example-2

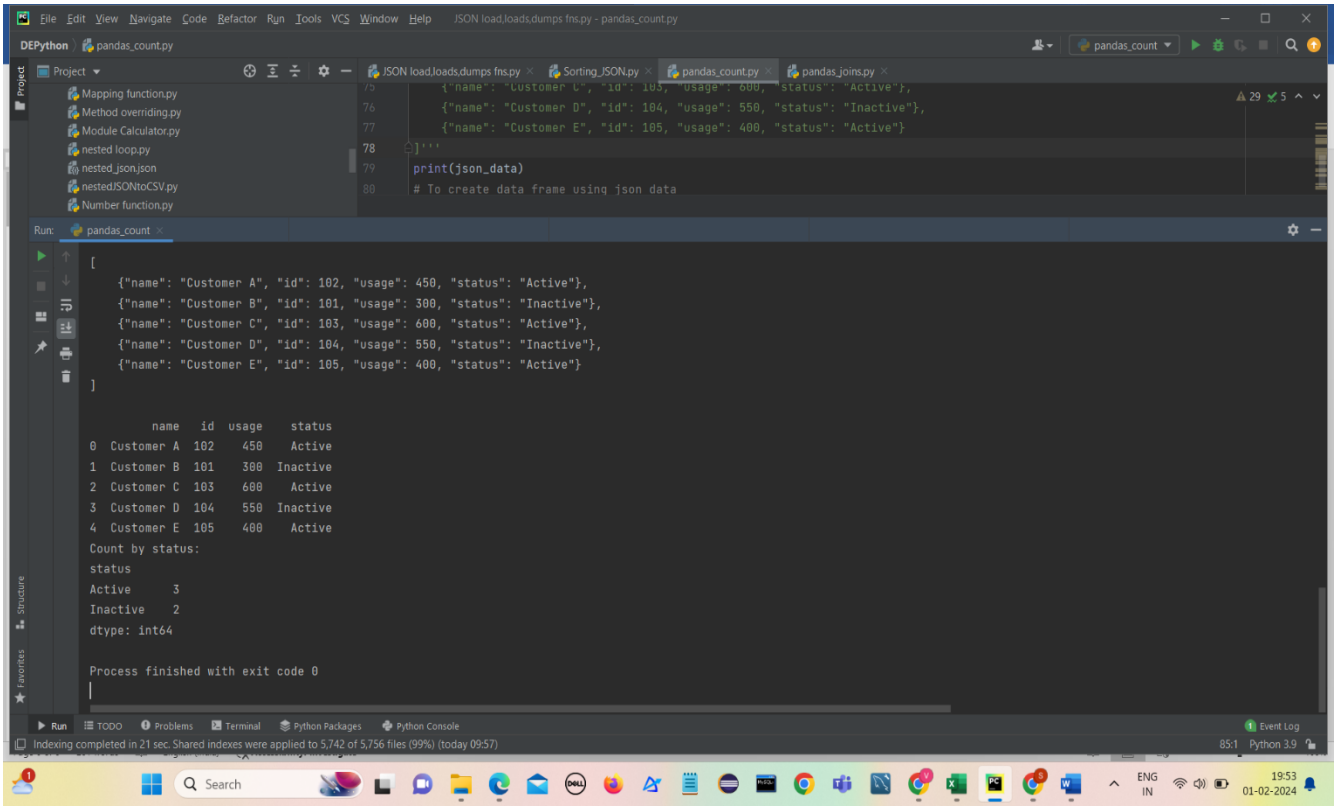
By passing null values



Example-3:

Passing json data(use read_json)

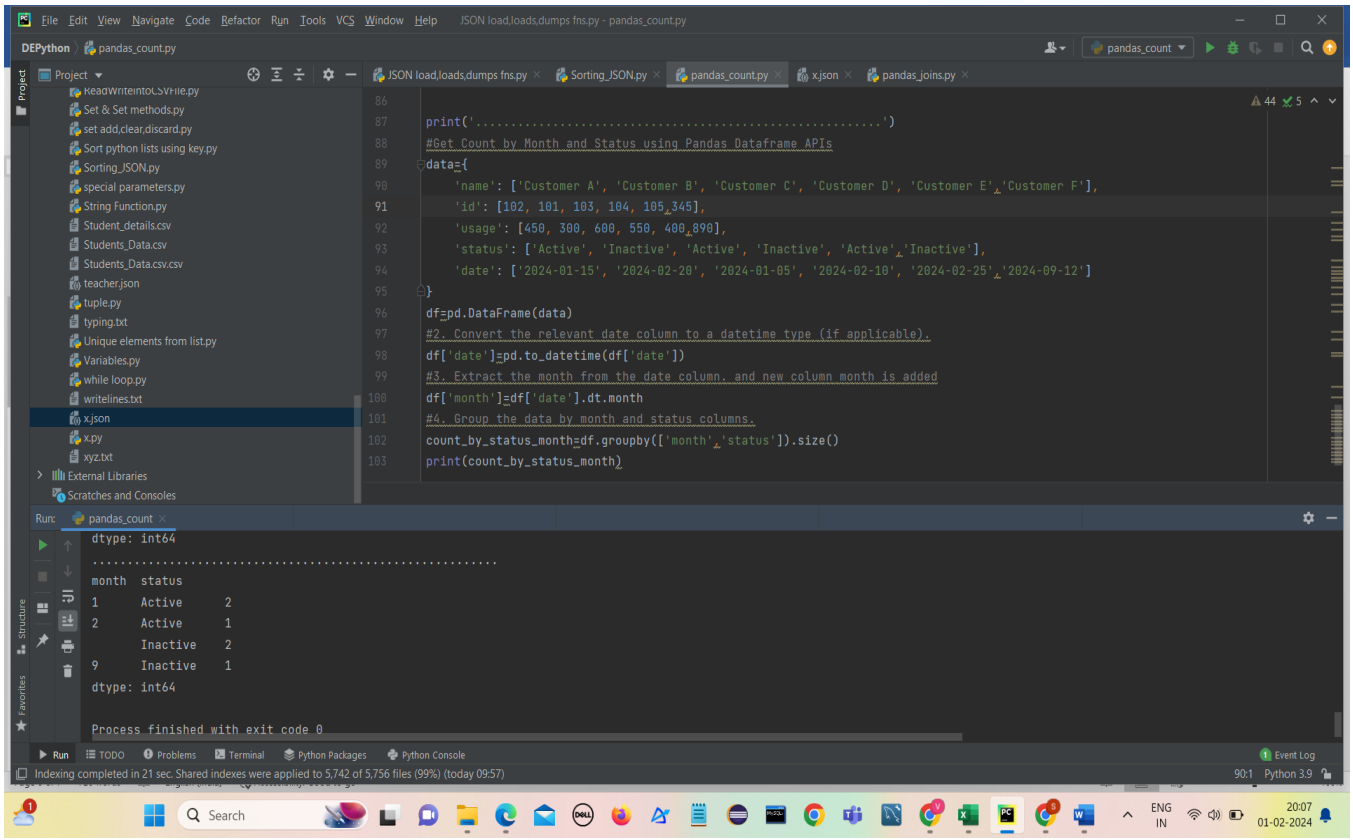




Get Count by Month and Status using Pandas Dataframe APIs

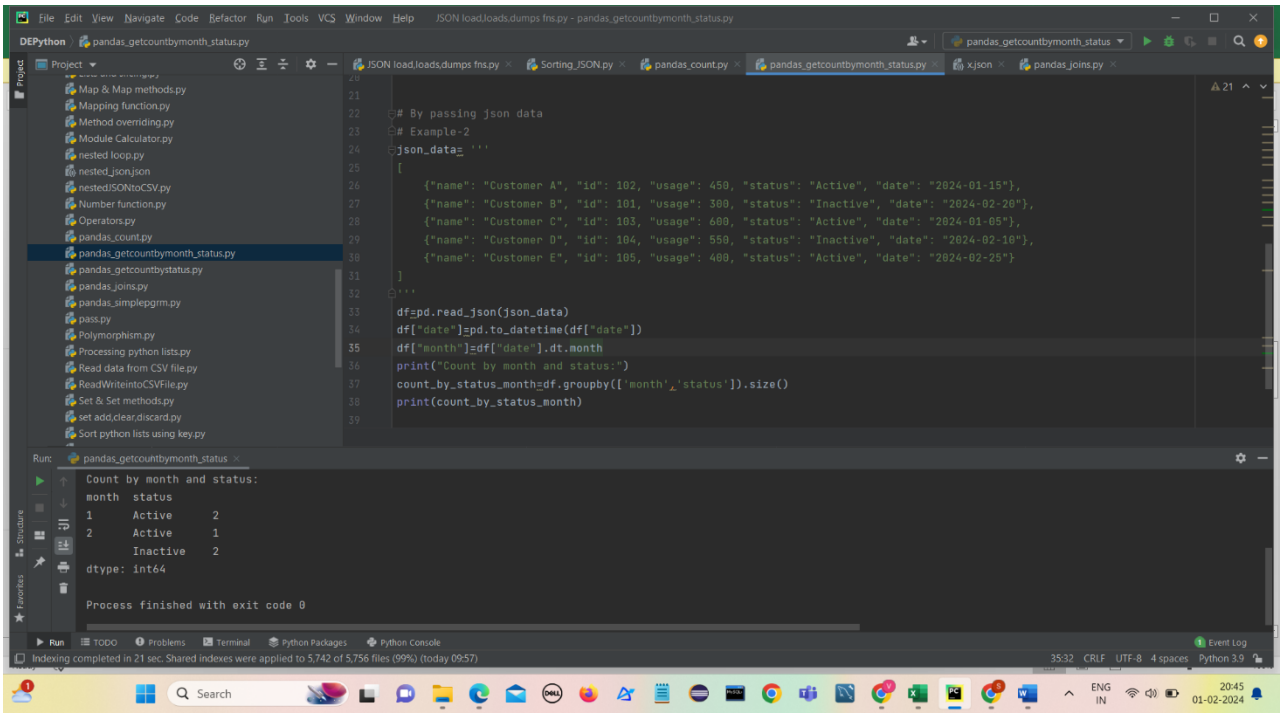
1. Create a DataFrame
2. Convert the relevant date column to a datetime type (if applicable).
3. Extract the month from the date column.
4. Group the data by month and status columns.
5. Use the **size()** function to get the count of each group

Example-1



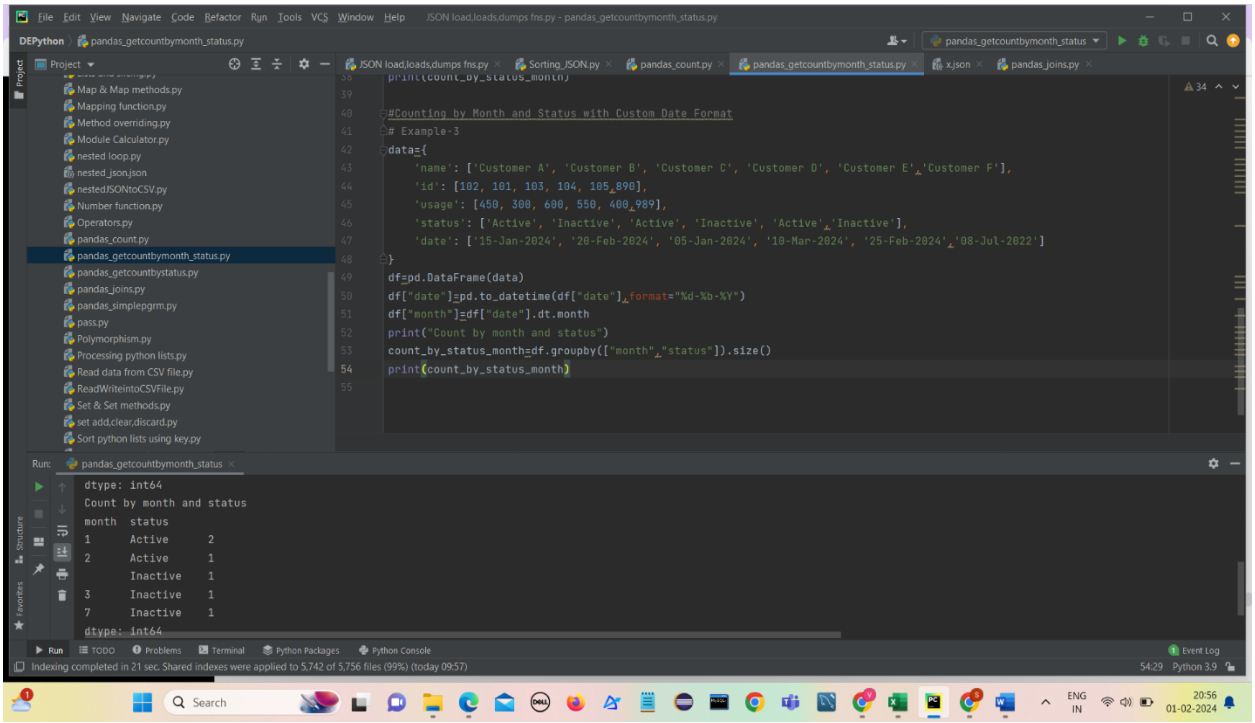
Example-2

By passing Json data



Example-3

Counting by Month and Status with Custom Date Format



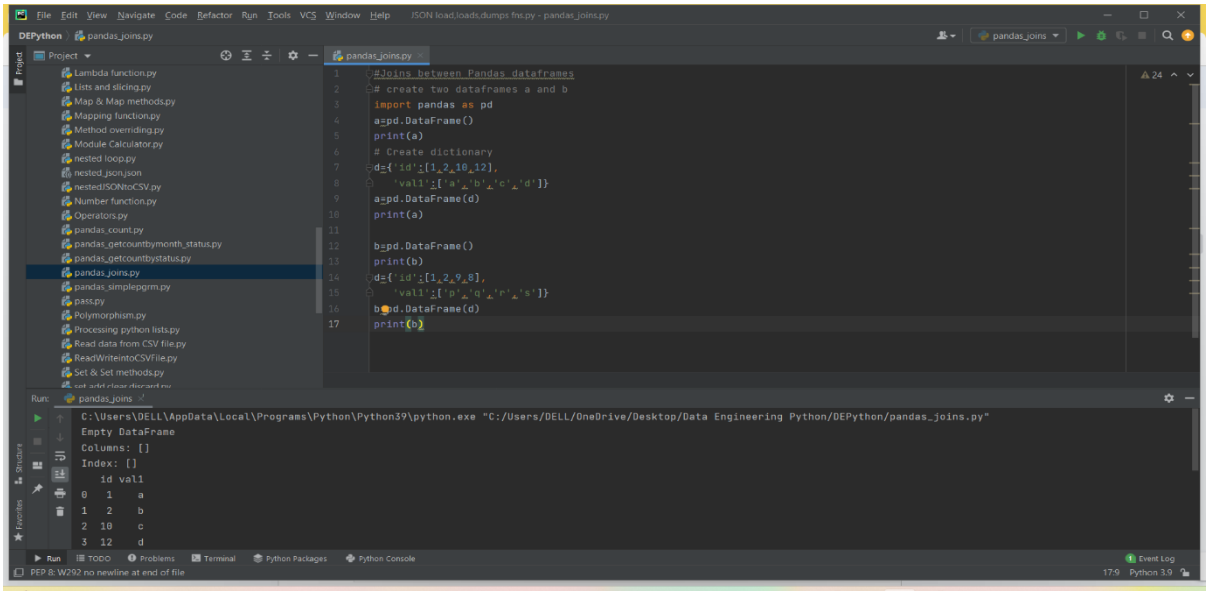
Joins between Pandas dataframes

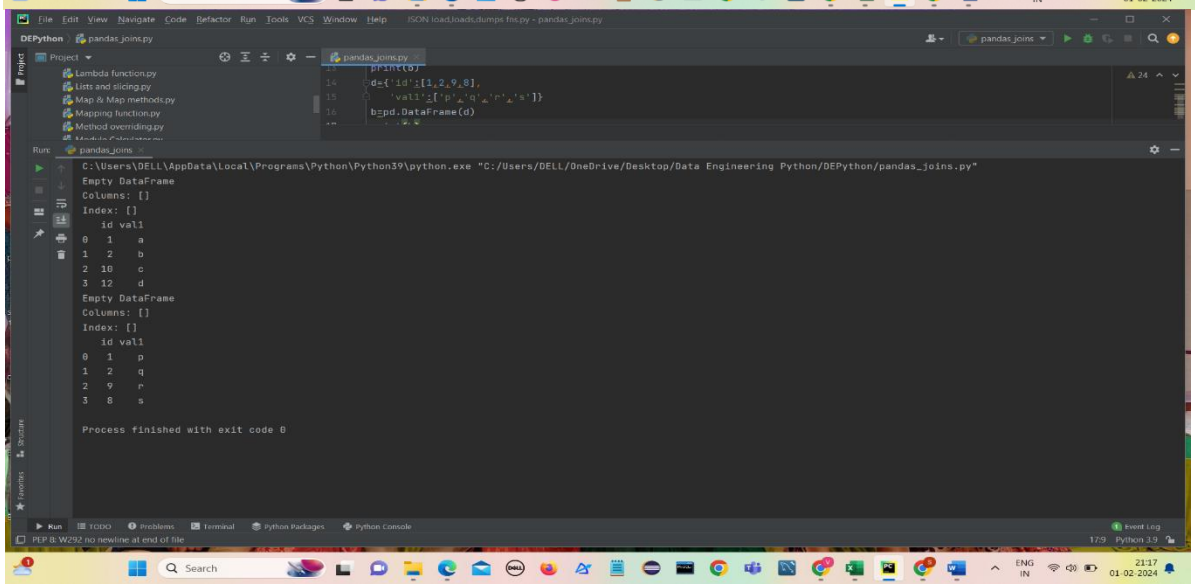
Different types of join operations can be performed on Pandas Dataframe.

There are five types of Joins in Pandas.

- Inner Join
- Left Outer Join
- Right Outer Join
- Full Outer Join or simply Outer Join
- Index Join

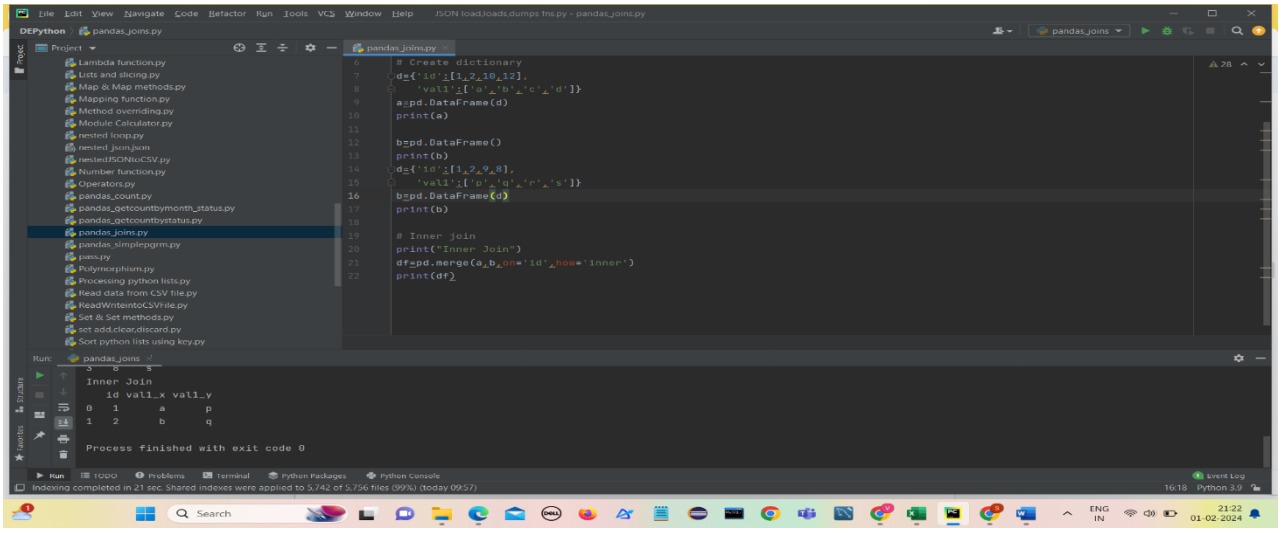
Create two dataframes a and b





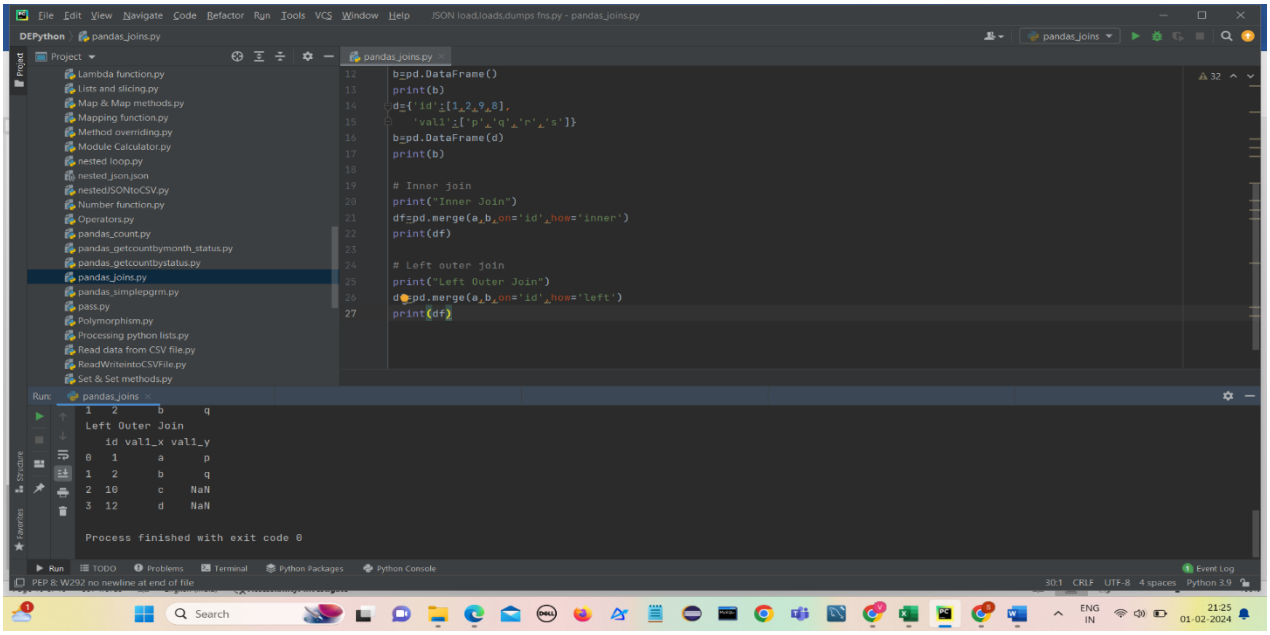
1) Inner Join

- It returns a Dataframe with only those rows that have common characteristics.
- This is similar to the intersection of two sets.



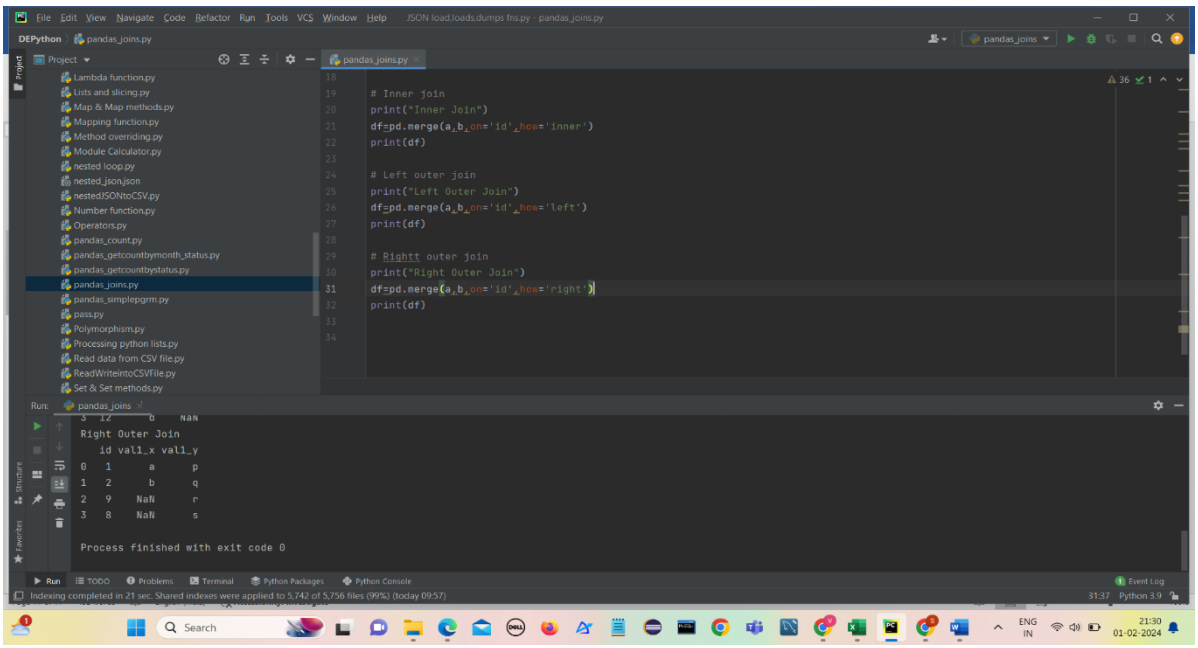
2) Left Outer Join

- All the records from the first Dataframe will be displayed, irrespective of whether the keys in the first Dataframe can be found in the second Dataframe.
- For the second Dataframe, only the records with the keys in the second Dataframe that can be found in the first Dataframe will be displayed.



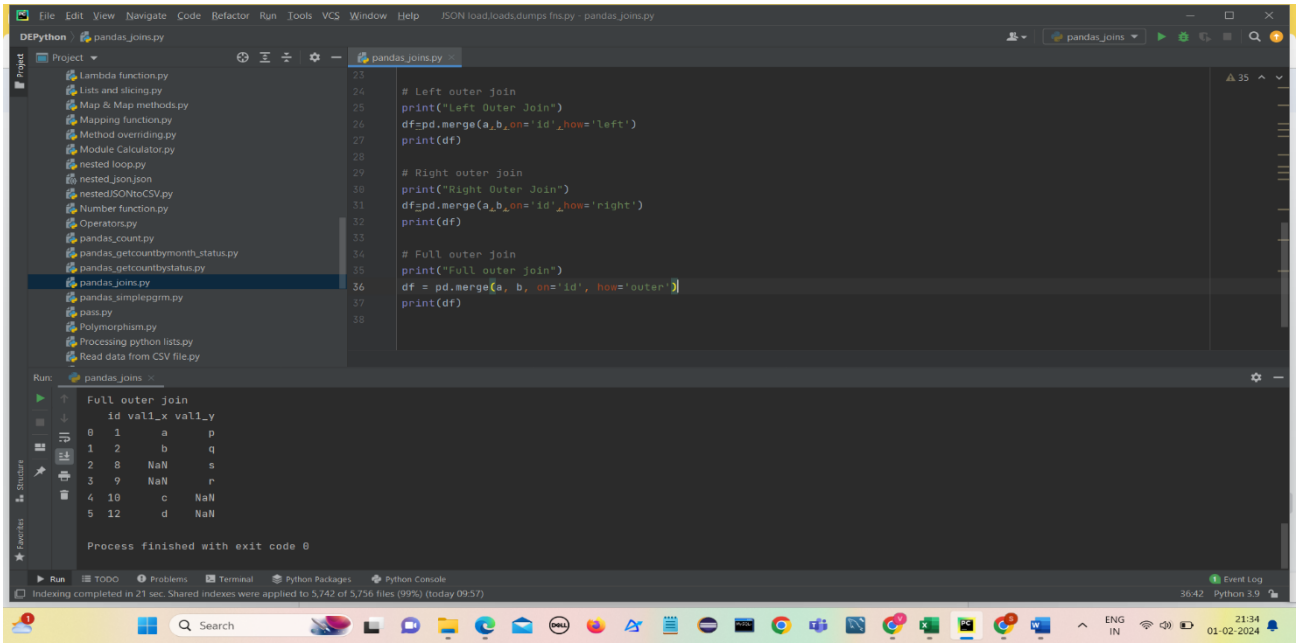
3) Right outer join

- All the records from the second Dataframe will be displayed.
- Only the records with the keys in the first Dataframe that can be found in the second Dataframe will be displayed.



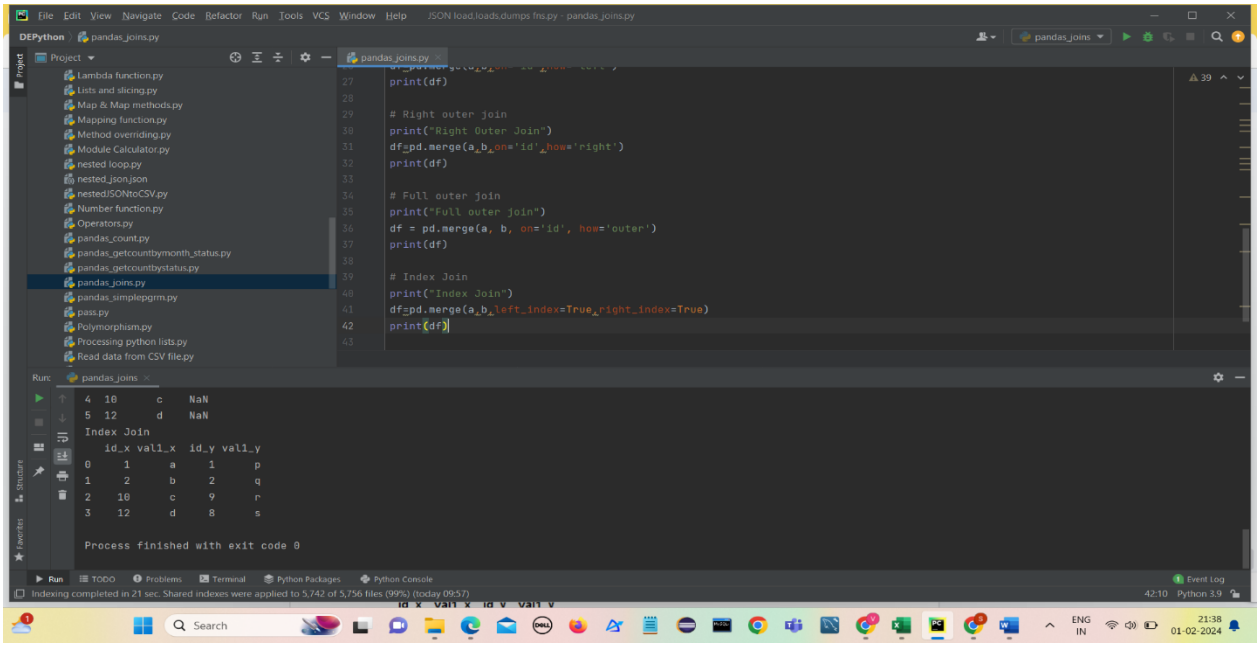
4) Full Outer Join

returns all the rows from the left Dataframe, and all the rows from the right Dataframe, and matches up rows where possible, with NaNs elsewhere.

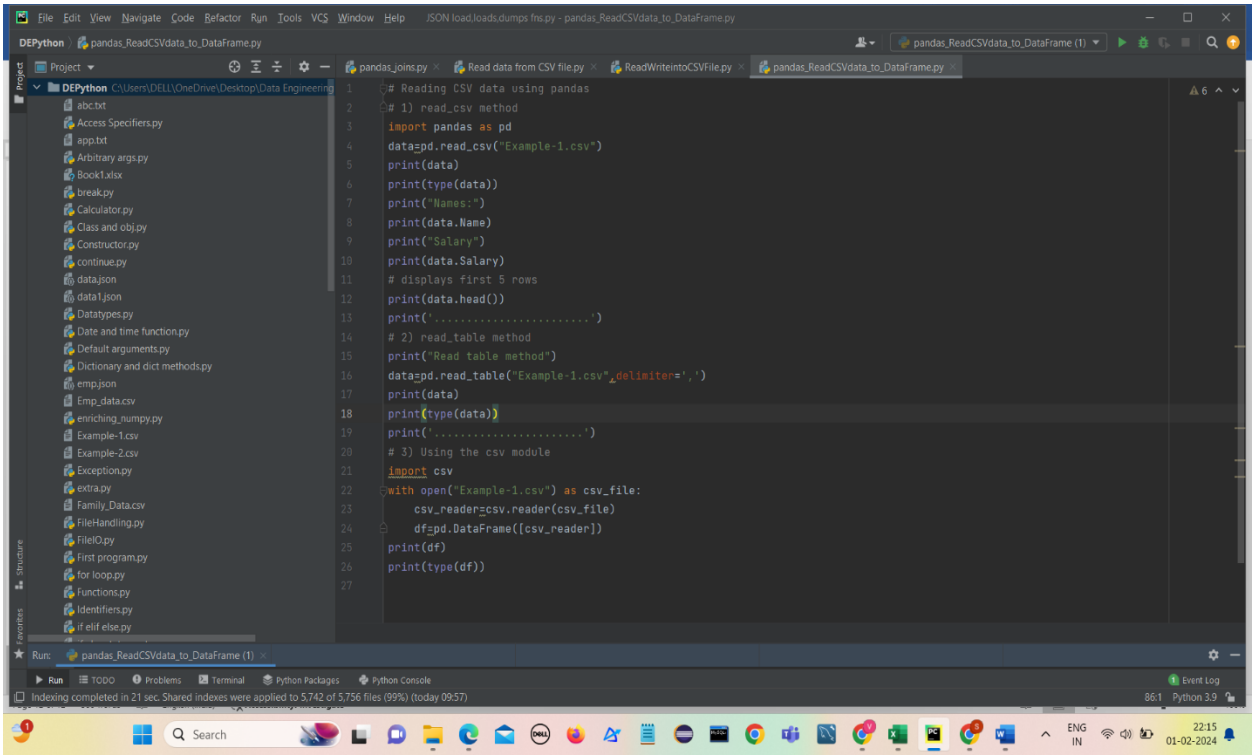


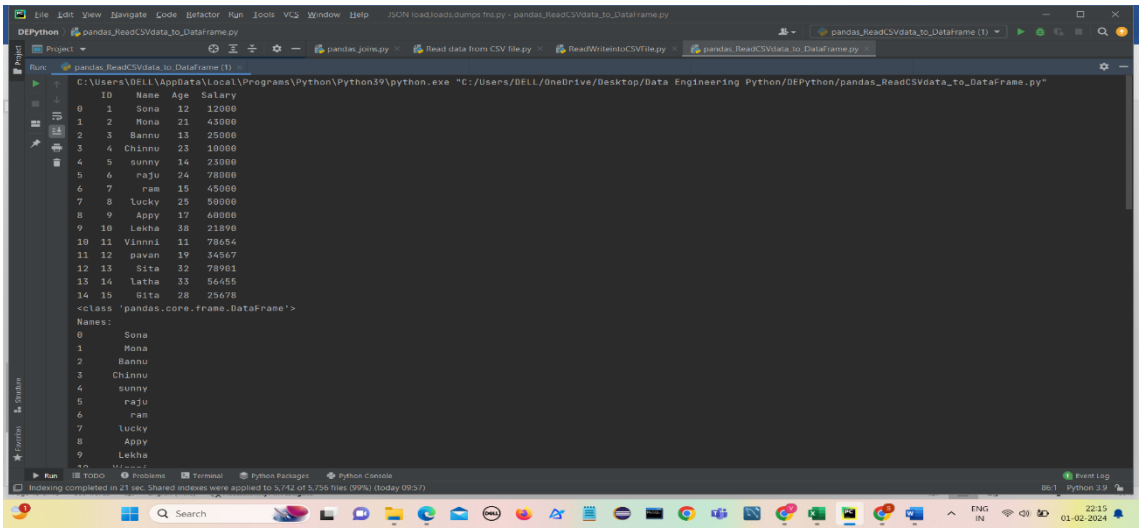
5) Index Join

- To merge the Dataframe on indices.
- pass the *left_index* and *right_index* arguments as True i.e. both the Dataframes are merged on an index using default Inner Join.

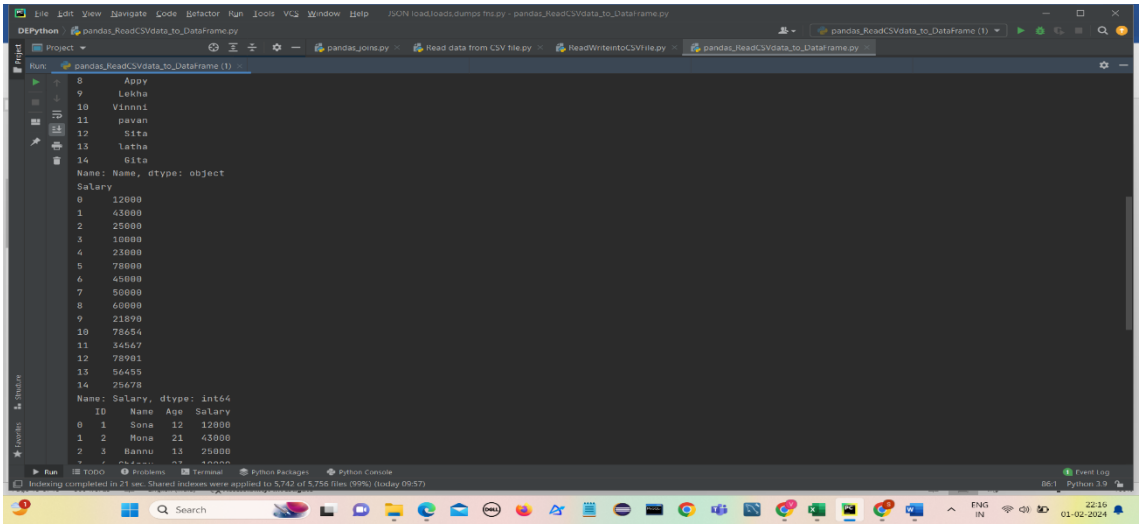


Read data from CSV files to Pandas Data Frames

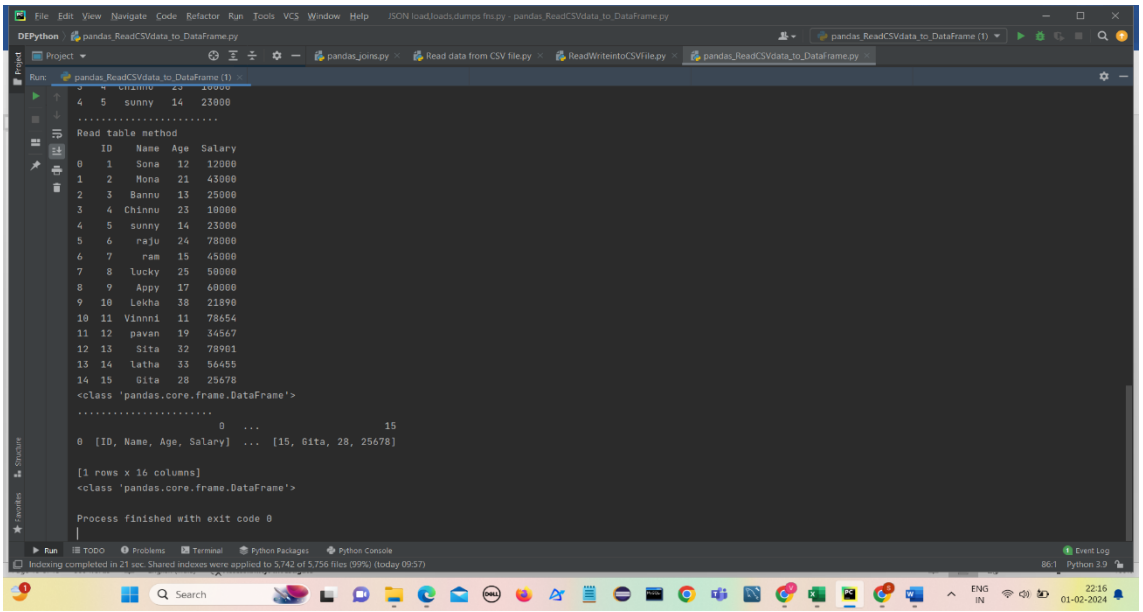




```
C:\Users\DELL\AppData\Local\Programs\Python\Python39\python.exe "C:/Users/DELL/OneDrive/Desktop/Data Engineering Python/DEPython/pandas_ReadCSVdata_to_DataFrame.py"
    ID    Name  Age  Salary
0    1    Sona   12   12000
1    2    Mona   21   43000
2    3   Bannu   13   25000
3    4  Chinnu   23   10000
4    5   sunny   14   23000
5    6    raju   24   78000
6    7     ram   15   45000
7    8   lucky   25   50000
8    9    Appy   17   60000
9   10   Lekha   38   21890
10  11  Vinnni   11   78654
11  12   pavan   19   34567
12  13    Sita   32   78901
13  14   latha   33   56455
14  15    Gita   28   25678
<class 'pandas.core.frame.DataFrame'>
Names:
0        Sona
1        Mona
2       Bannu
3      Chinnu
4       sunny
5        raju
6         ram
7       lucky
8        Appy
9       Lekha
```



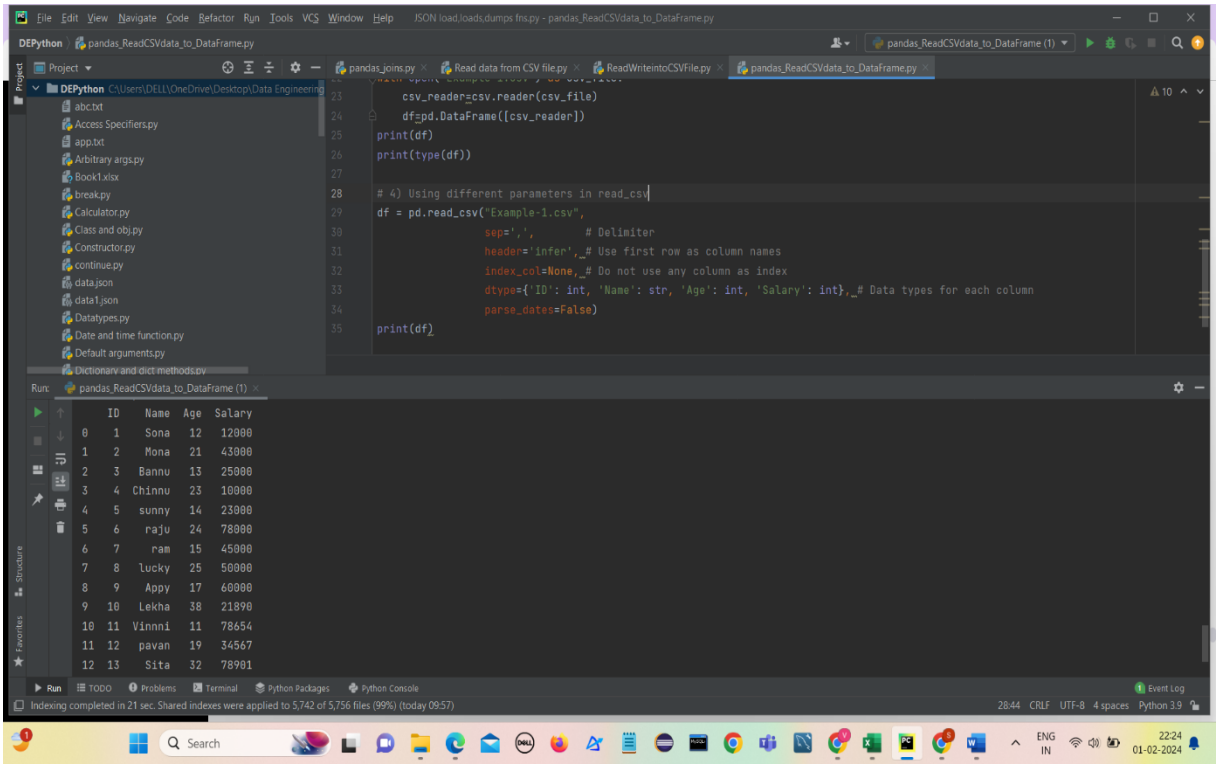
```
8        Appy
9       Lekha
10     Vinnni
11      pavan
12       Sita
13      latha
14       Gita
Name: Name, dtype: object
Salary
0     12000
1     43000
2     25000
3     10000
4     23000
5     78000
6     45000
7     50000
8     60000
9     21890
10    78654
11    34567
12    78901
13    56455
14    25678
Name: Salary, dtype: int64
   ID    Name  Age  Salary
0   1    Sona   12   12000
1   2    Mona   21   43000
2   3   Bannu   13   25000
```



```
4   5   sunny   14   23000
.........................
Read table method
    ID    Name  Age  Salary
0    1    Sona   12   12000
1    2    Mona   21   43000
2    3   Bannu   13   25000
3    4  Chinnu   23   10000
4    5   sunny   14   23000
5    6    raju   24   78000
6    7     ram   15   45000
7    8   lucky   25   50000
8    9    Appy   17   60000
9   10   Lekha   38   21890
10  11  Vinnni   11   78654
11  12   pavan   19   34567
12  13    Sita   32   78901
13  14   latha   33   56455
14  15    Gita   28   25678
<class 'pandas.core.frame.DataFrame'>
.........................
                                0   ...                                15
0  [ID, Name, Age, Salary]  ...  [15, Gita, 28, 25678]

[1 rows x 16 columns]
<class 'pandas.core.frame.DataFrame'>

Process finished with exit code 0
```



Filter data in pandas Dataframe using query

It is used to filter dataframe

Syntax: DataFrame.query(expr, inplace=False, **kwargs)

Parameters:

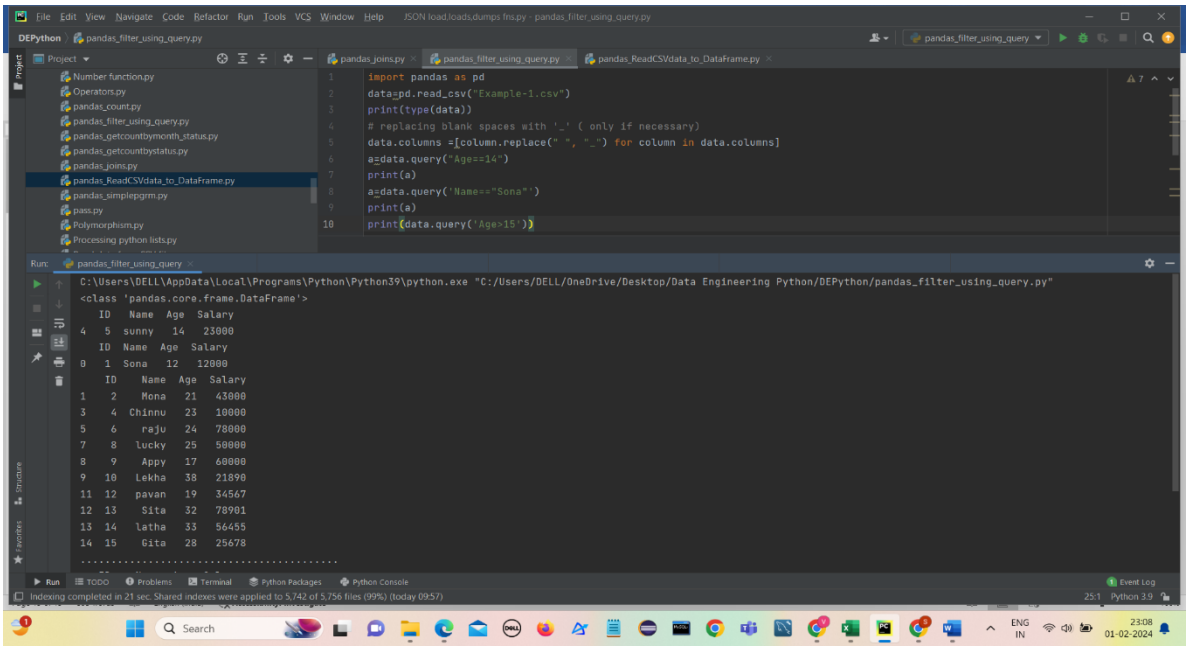
- **expr:** Expression in string form to filter data.
- **inplace:** Make changes in the original data frame if True
- **kwargs:** Other keyword arguments.

Return type: Filtered Data frame

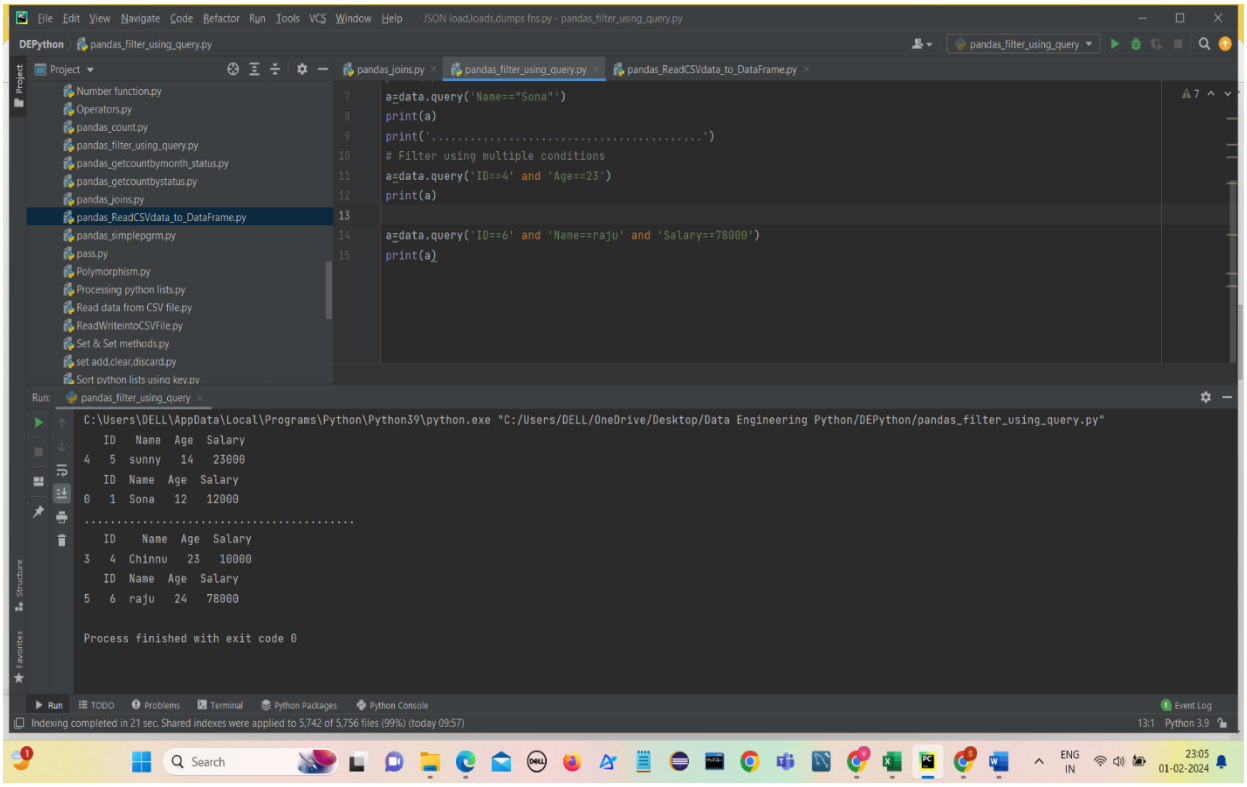
Dataframe.query() method only works if the column name doesn't have any empty spaces.

If any empty spaces are there, before using query method replace all empty spaces with "-"

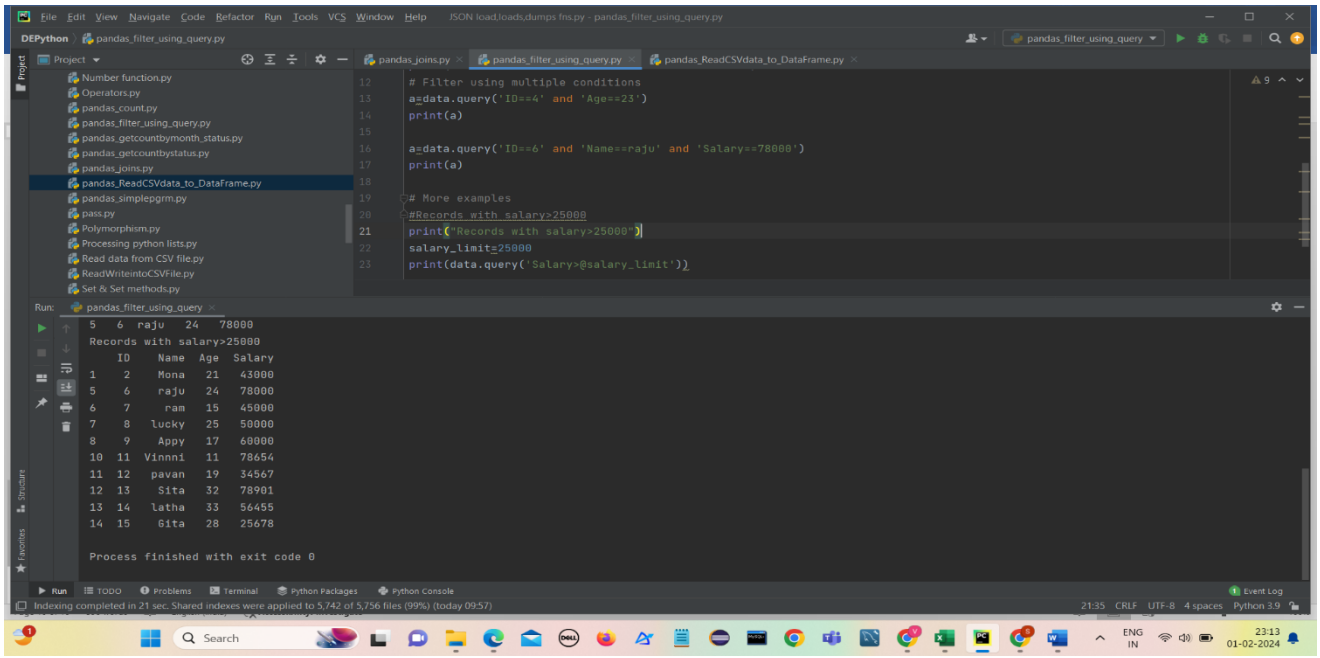
```
# replacing blank spaces with '_'
data.columns =
    [column.replace(" ", "_") for column in data.columns]
```



Filtering with multiple conditions



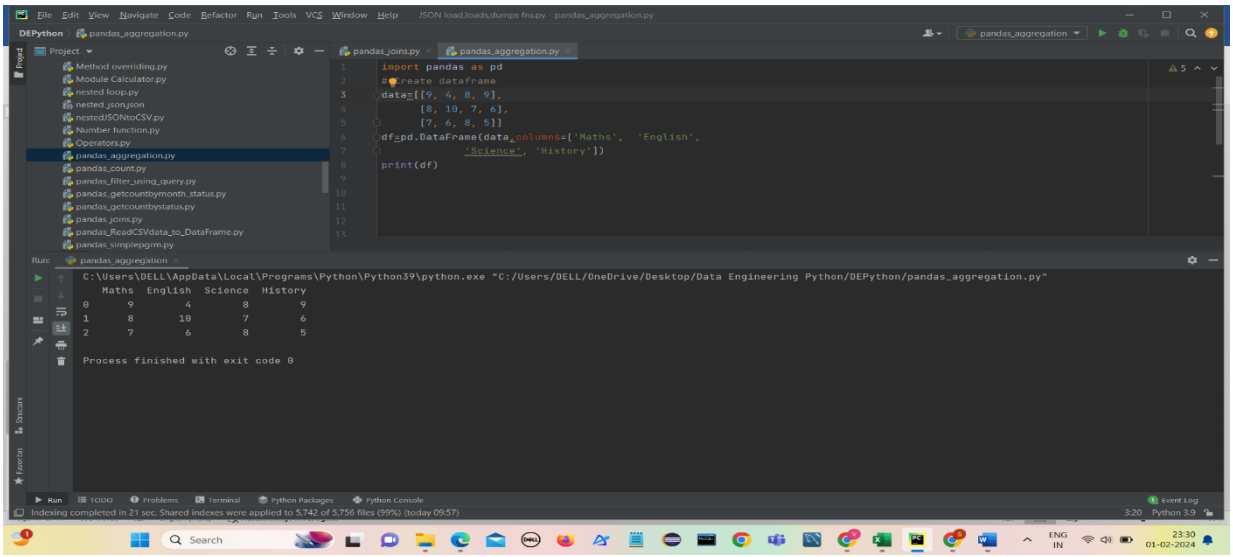
Records with salary>25000



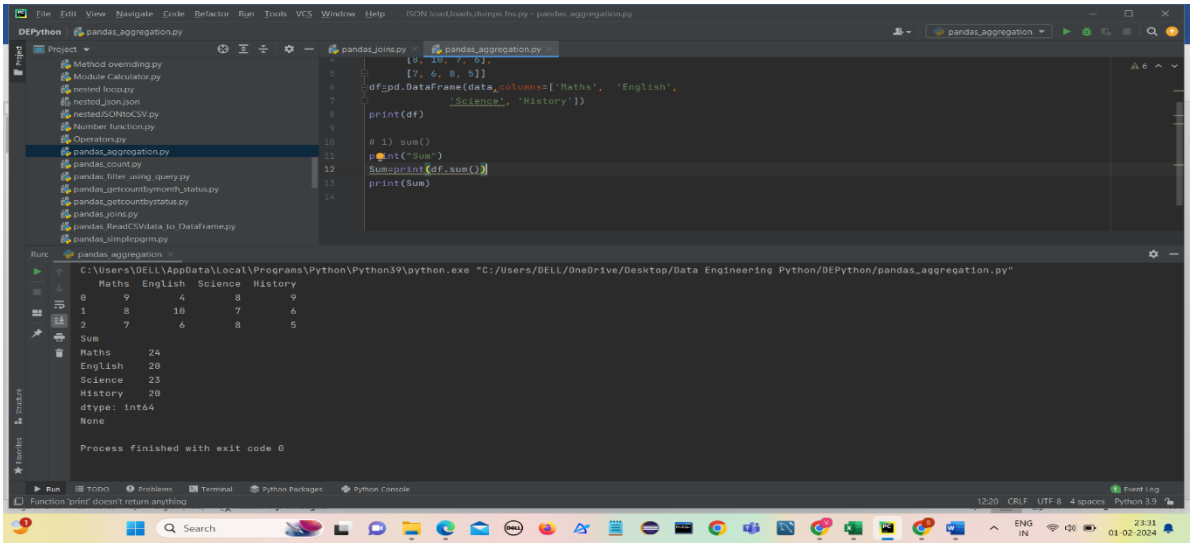
Aggregations in Pandas

Aggregation can be used to get a summary of columns in our dataset like getting sum, minimum, maximum, etc. from a particular column of our dataset.

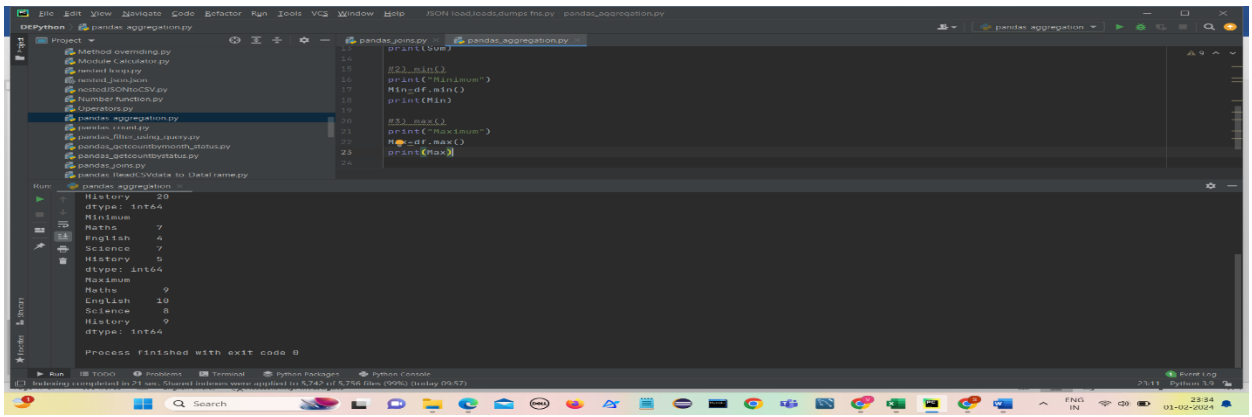
- **sum()** :Compute sum of column values
- **min()** :Compute min of column values
- **max()** :Compute max of column values
- **mean()** :Compute mean of column
- **size()** :Compute column sizes
- **describe()** :Generates descriptive statistics
- **first()** :Compute first of group values
- **last()** :Compute last of group values
- **count()** :Compute count of column values
- **std()** :Standard deviation of column
- **var()** :Compute variance of column
- **sem()** :Standard error of the mean of column



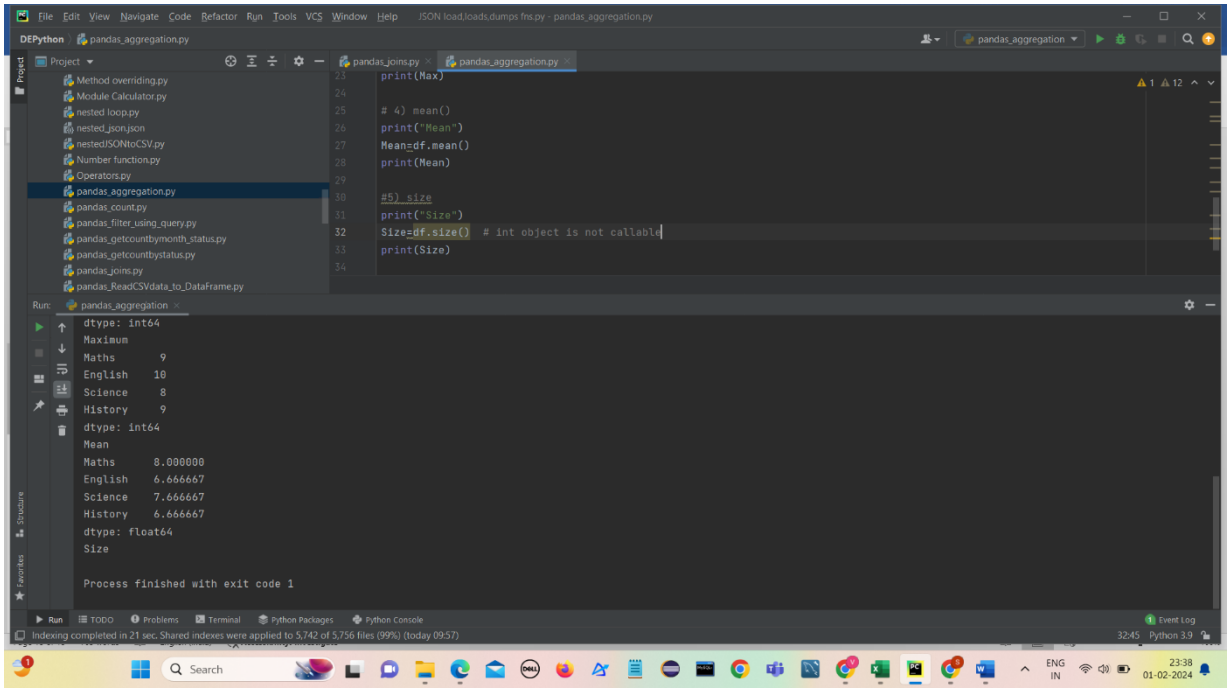
1) sum()



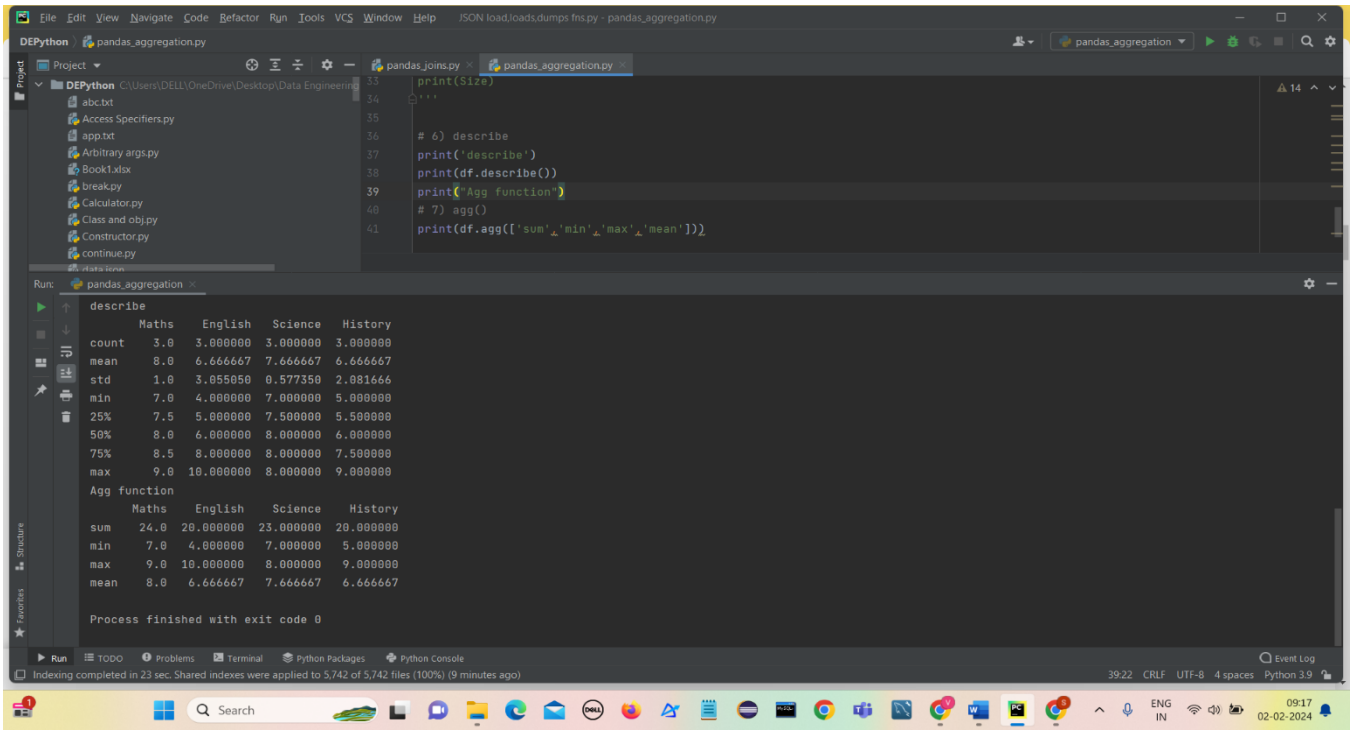
2) Minimum and
3) Maximum



4) Mean()
5) Size()



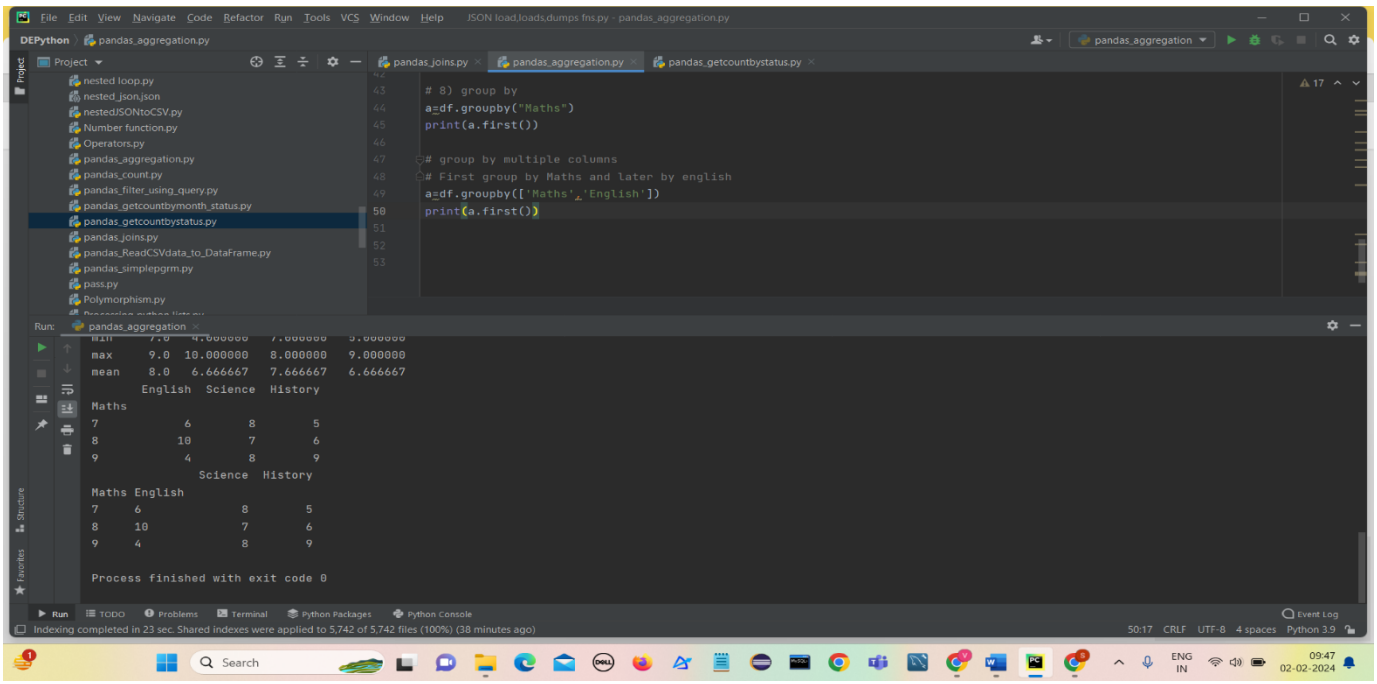
6) describe()
7) agg()



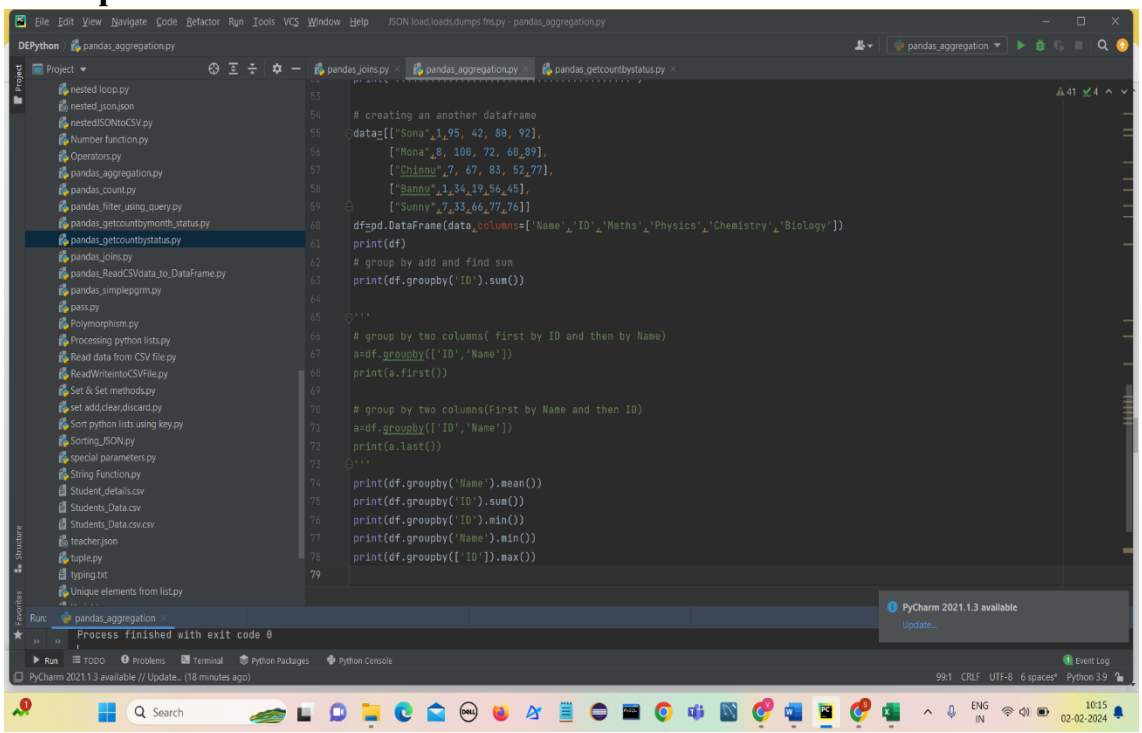
Grouping in Pandas

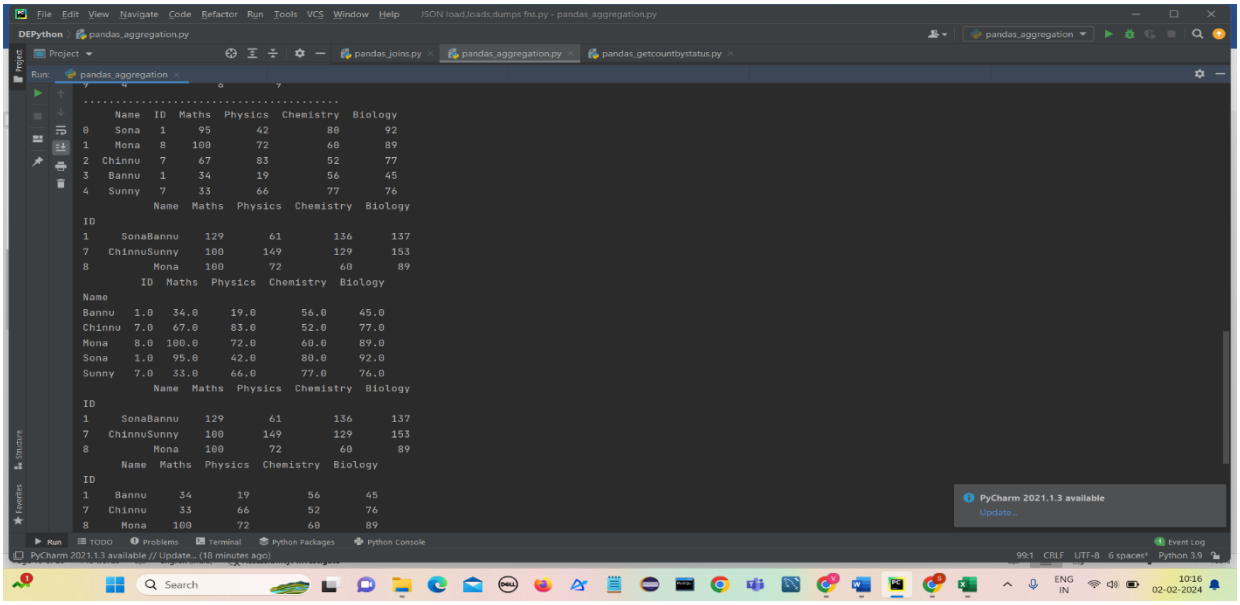
Grouping is used to group data using some criteria from our dataset. It is used as split-apply-combine strategy.

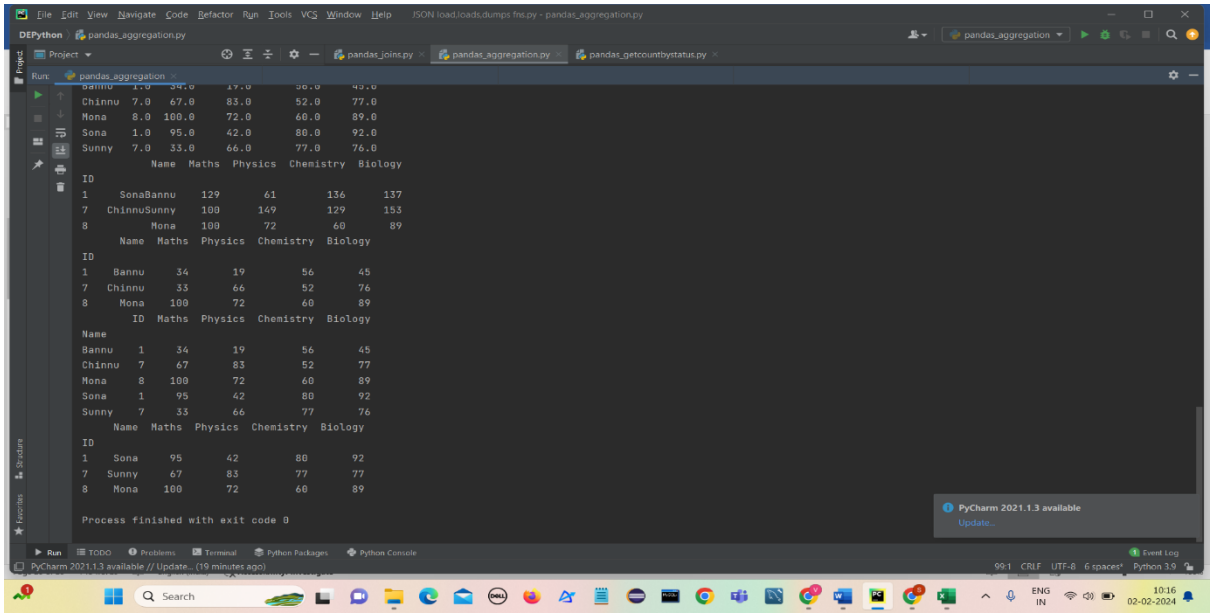
- Splitting the data into groups based on some criteria.
- Applying a function to each group independently.
- Combining the results

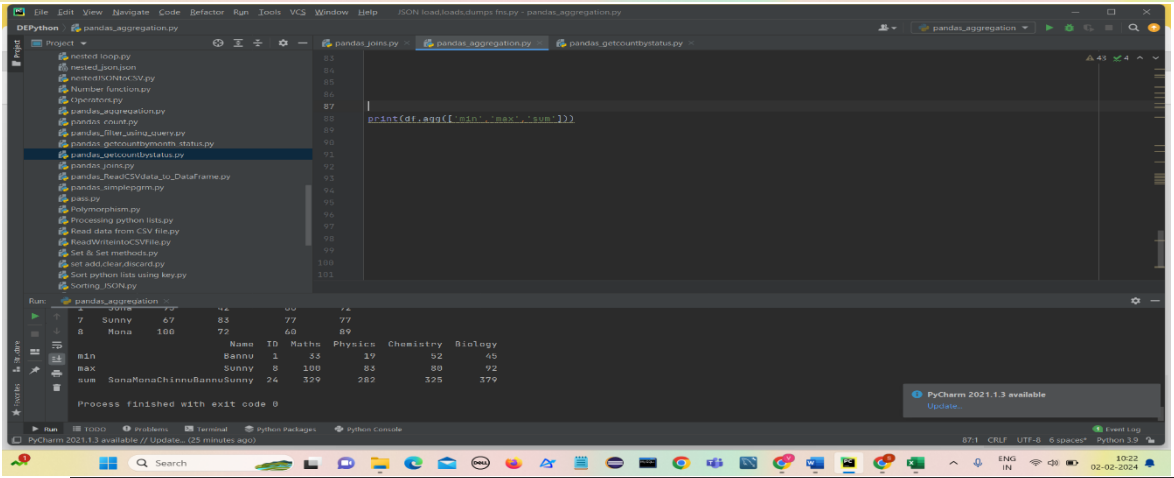


Example-2







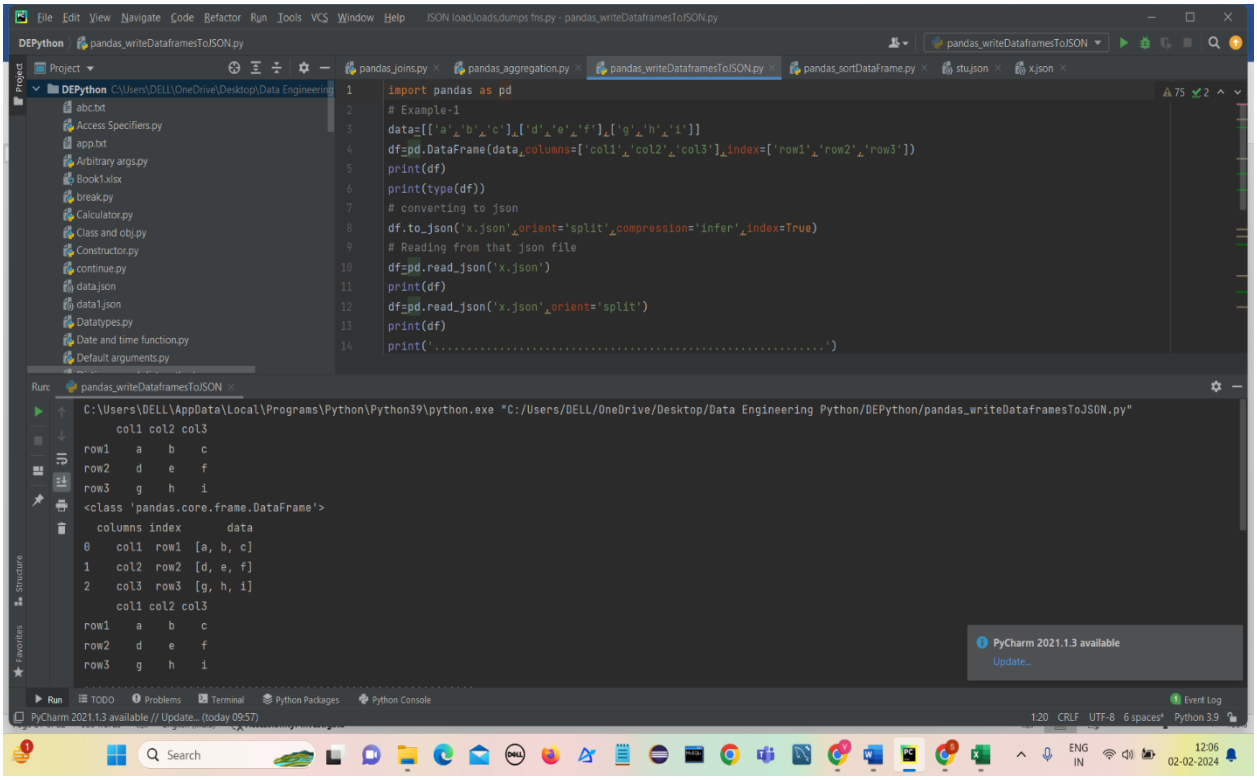



```
print(df.agg(['sum','min','max','mean','median','prod']))
```

Writing pandas data frames to JSON files

to_json() function is used to convert a DataFrame to a JSON string. This function provides several parameters to customize the output JSON format.

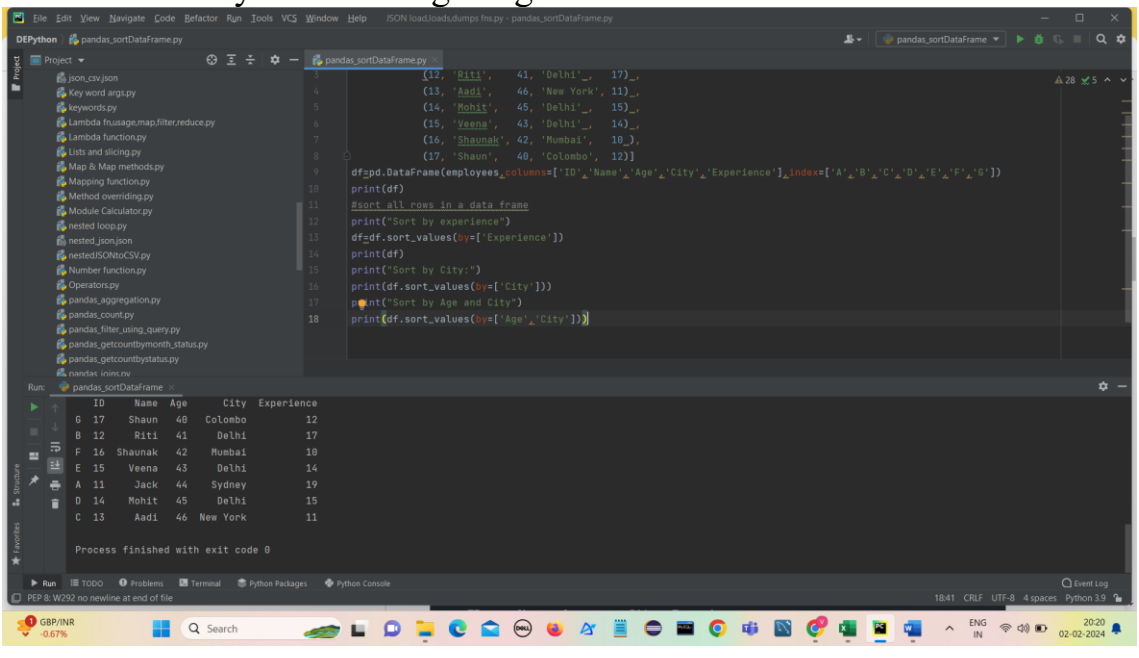
1. **path_or_buf**: Specifies the file path or a writable buffer where the JSON string will be written. If not provided, the JSON string will be returned as a string.
2. **orient**: Specifies the orientation of the JSON string. Possible values are:
    - **'split'**: Dict like {index -> [index], columns -> [columns], data -> [values]} The JSON output is split into three parts: index, columns, and data
    - **'records'**: List like [{column -> value}, … , {column -> value}]
    - **'index'**: Dict like {index -> {column -> value}}
    - **'columns'**: Dict like {column -> {index -> value}}
    - **'values'**: Just the values array
3. **date_format**: Specifies the date format to use for datetime-like objects. The default is 'iso'.
4. **double_precision**: Specifies the precision for floating point numbers. The default is 10.
5. **force_ascii**: If True, all non-ASCII characters in the JSON string will be escaped. The default is True.
6. **date_unit**: Specifies the time unit to encode dates. The default is 'ms' (milliseconds since epoch).
7. **lines**: If True, the output will be a JSON string per line. The default is False.
8. **compression**: Specifies the compression mode. Possible values are {'infer', 'gzip', 'bz2', 'xz', None}. The default is 'infer'(compresses based on extension of output file)

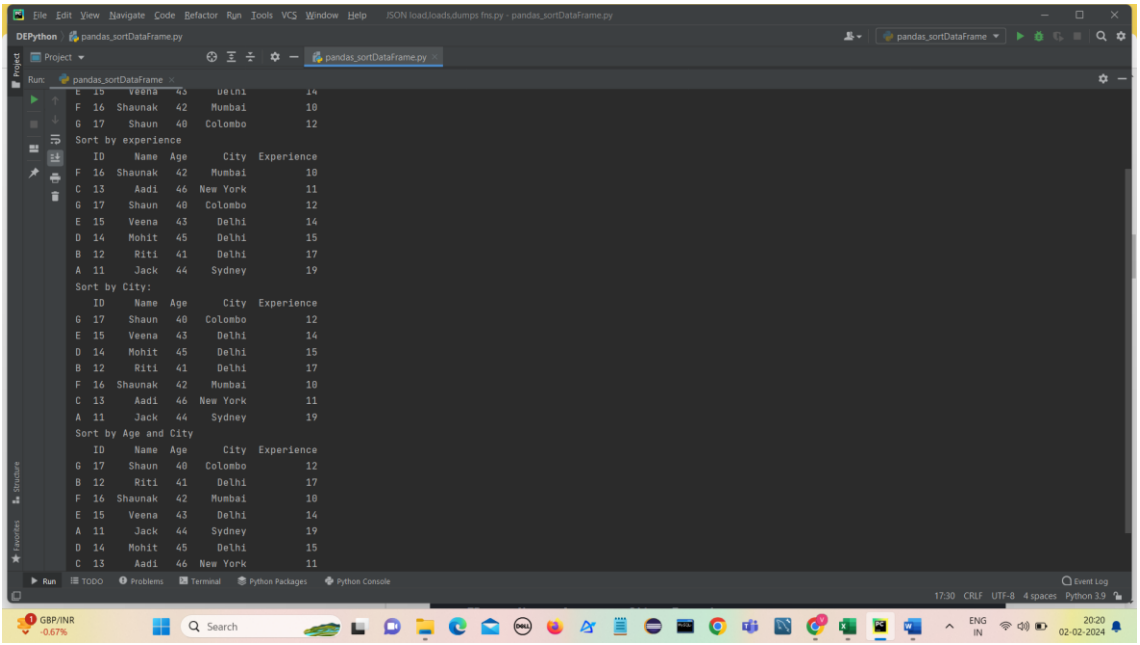


Sort data in Pandas Data Frames

Sort all rows of DataFrame by a column

In Pandas, the DataFrame provides a method sort_values(), and it sorts the DataFrame by values along the given axis.

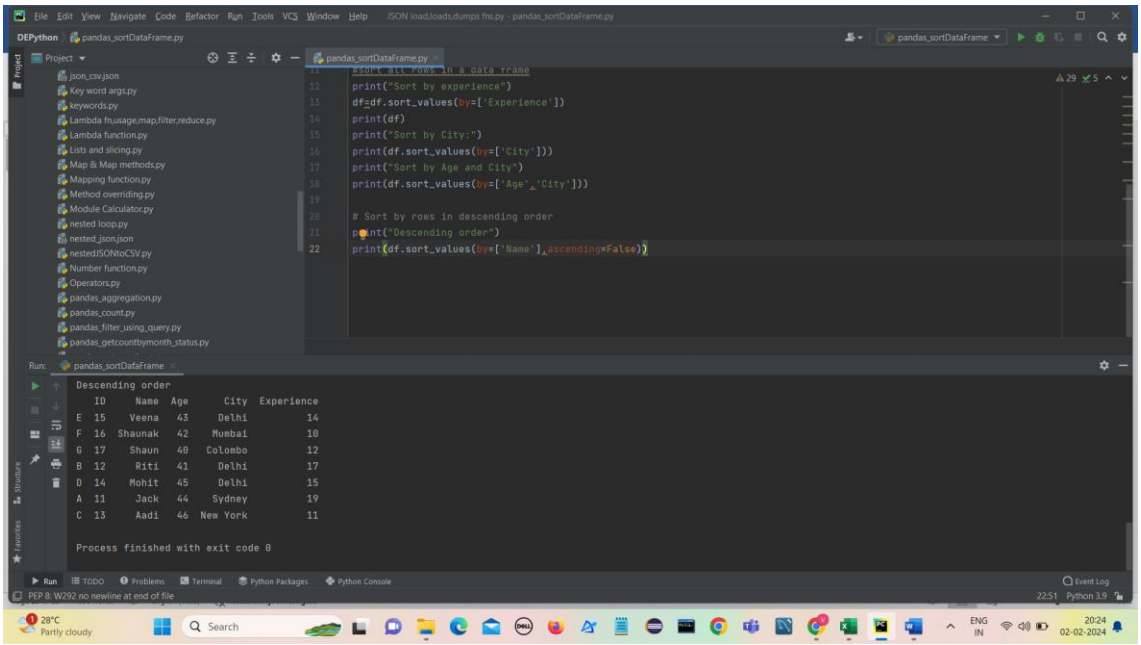




It sorted the DataFrame along the 'index' axis, i.e., sorted all the rows along the column 'Experience'.

Sort all rows of DataFrame by a column in Descending Order

To sort the DataFrame in descending order, pass the argument ascending=False in the sort_values() function.



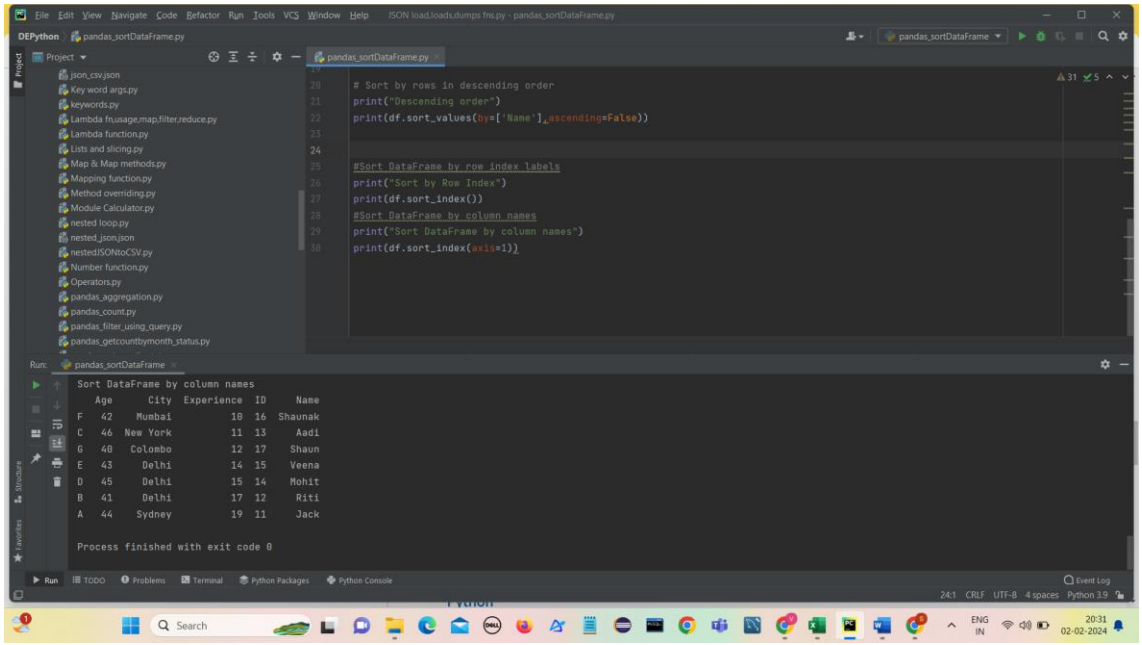
Sort DataFrame by row index labels

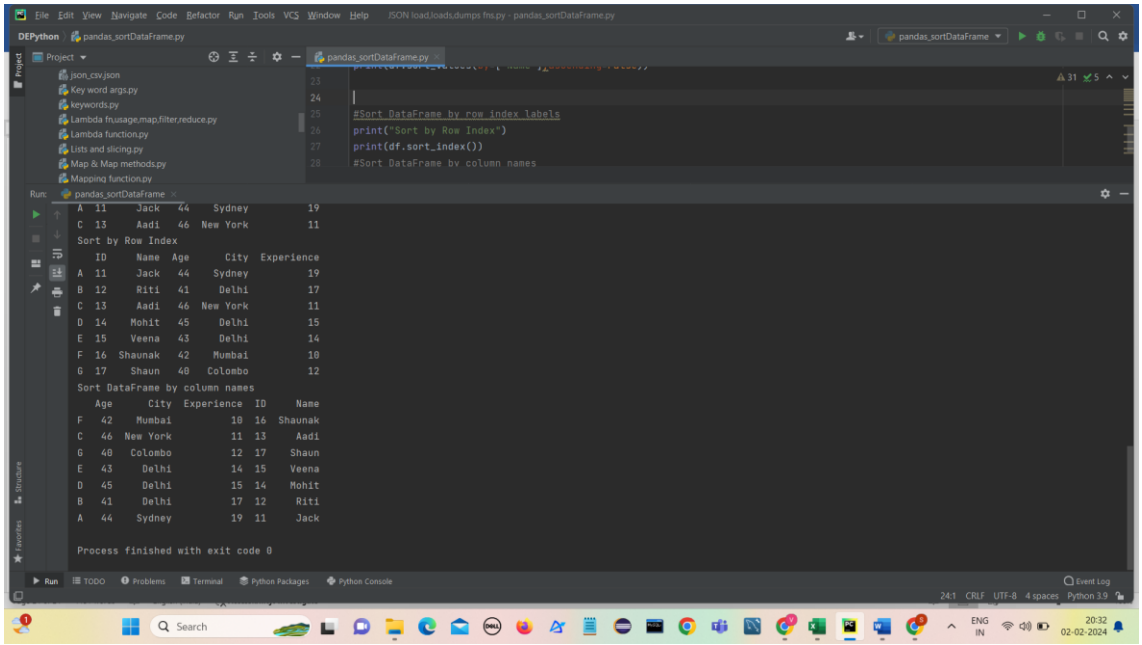
In Pandas, the DataFrame provides a method sort_index(), and it sorts the DataFrame by index labels along the given axis. By default, it sorts the rows of DataFrame based on row index labels.

Sort DataFrame by column names

Pass the axis=1 argument in the sort_index() method of DataFrame. It will sort the DataFrame by the column names.

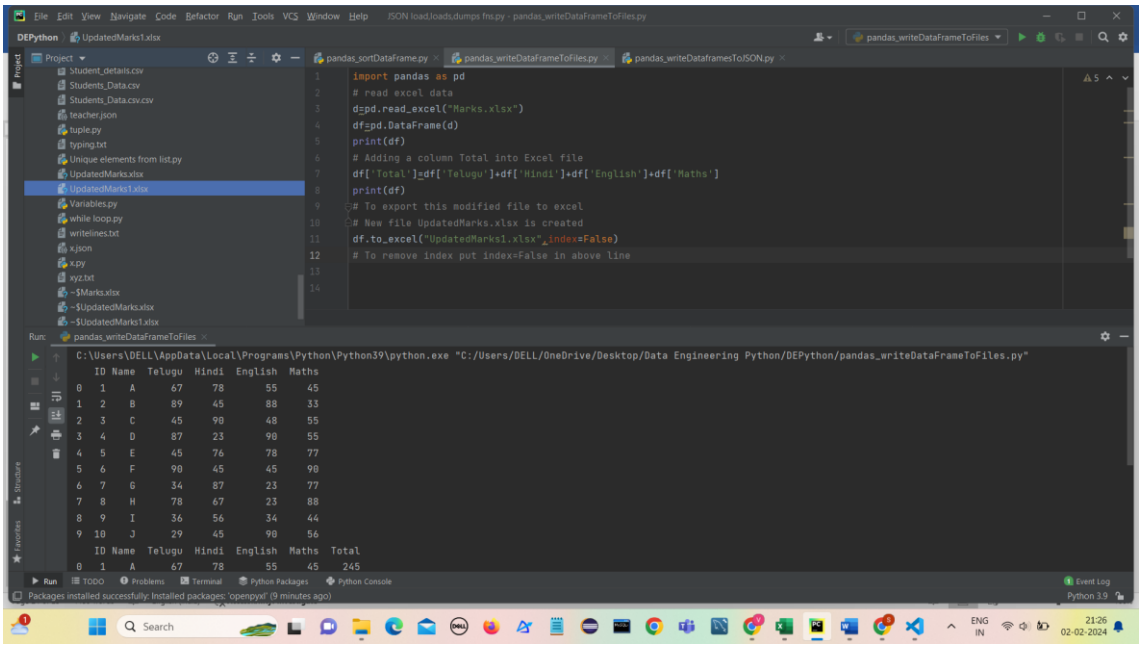
So Age comes first followed by City followed by Experience so on.

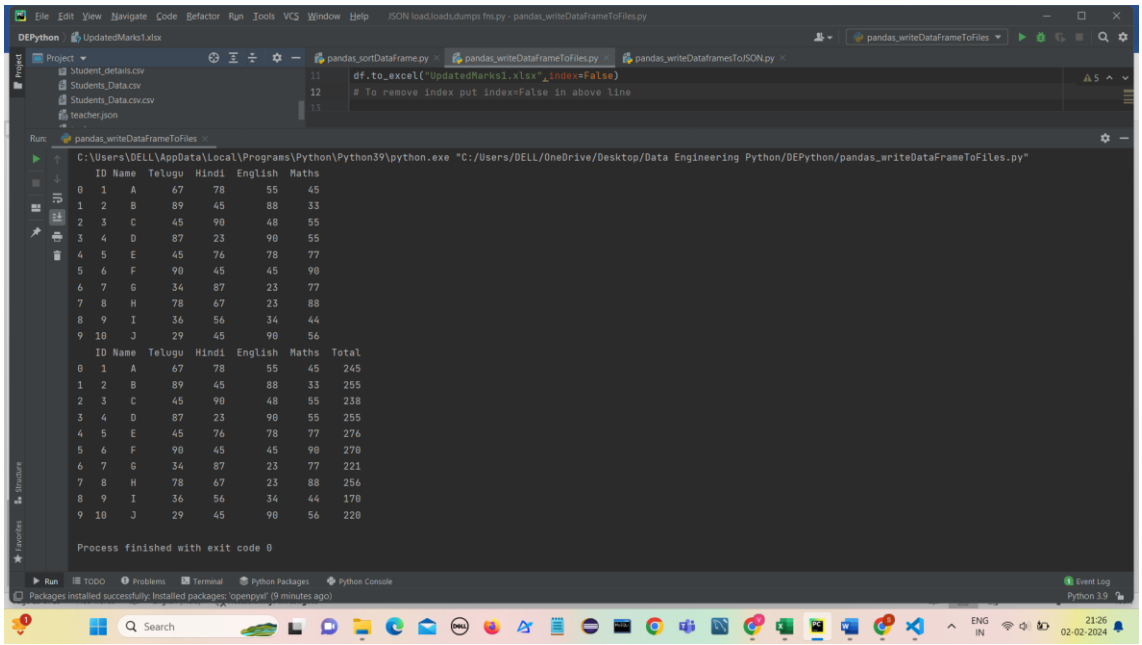




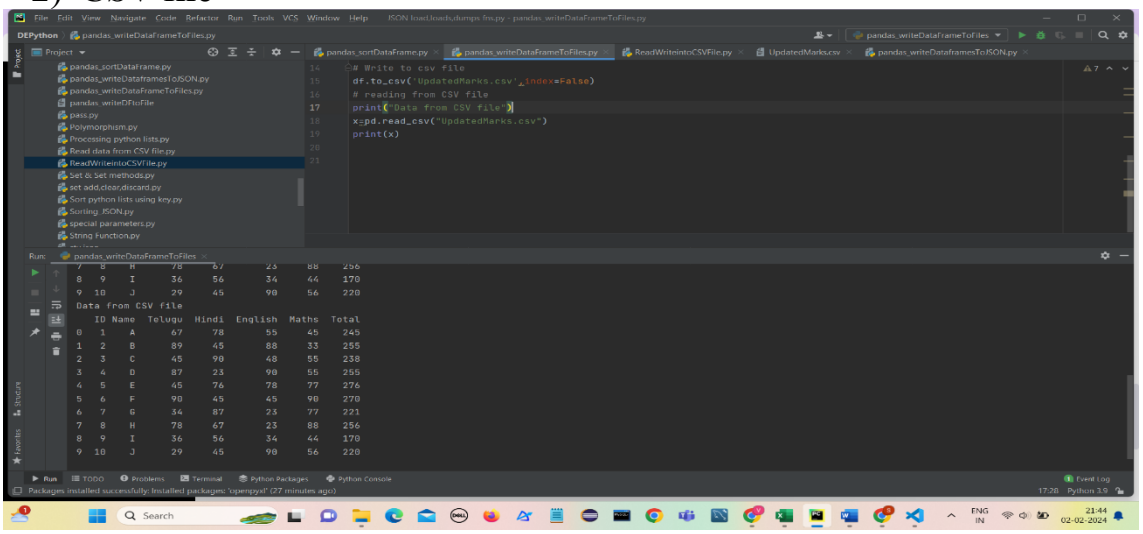
Writing Pandas DataFrames to files

1) Excel file

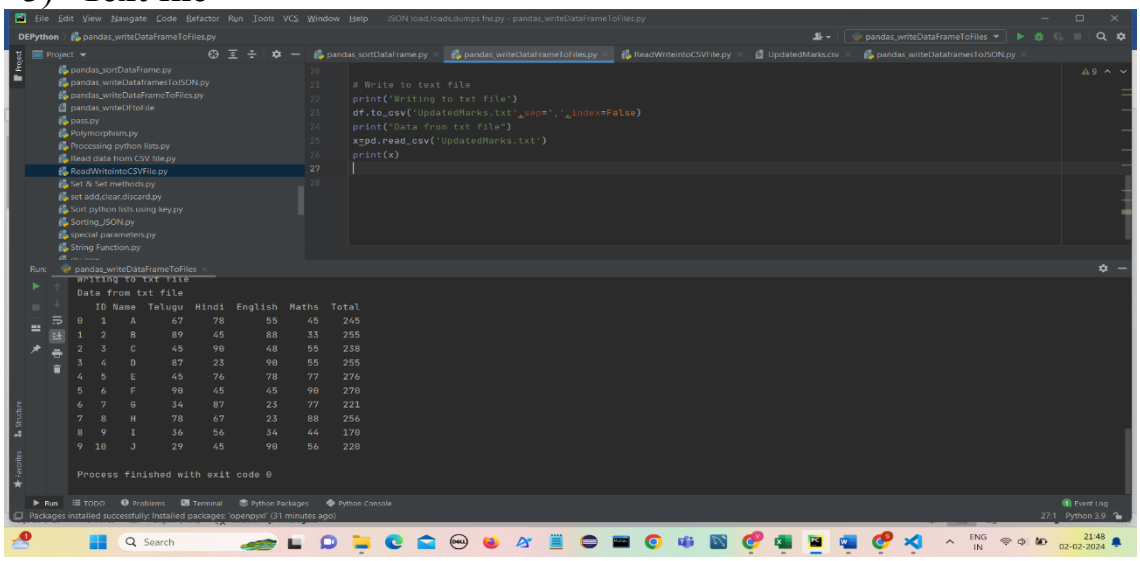




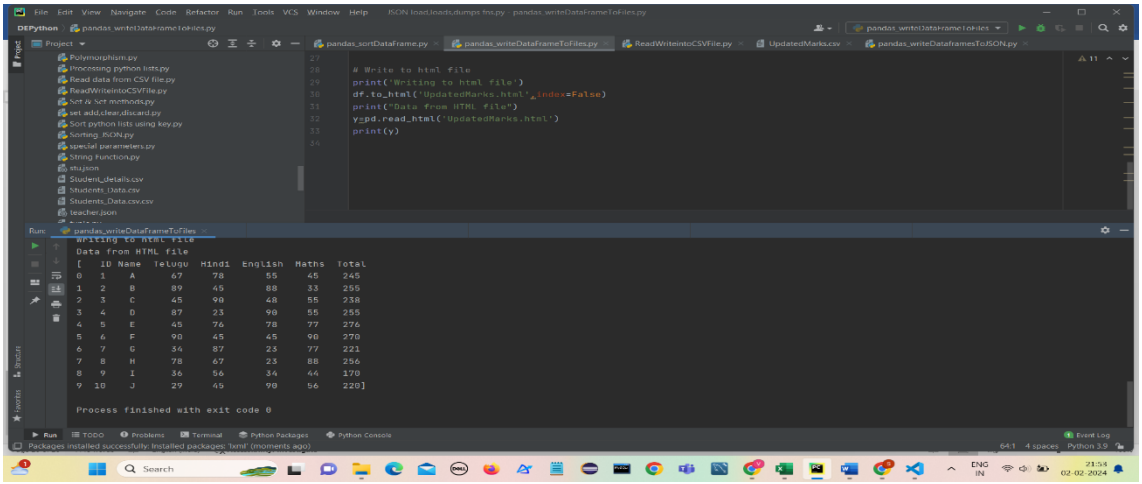
2) CSV file



3) Text file



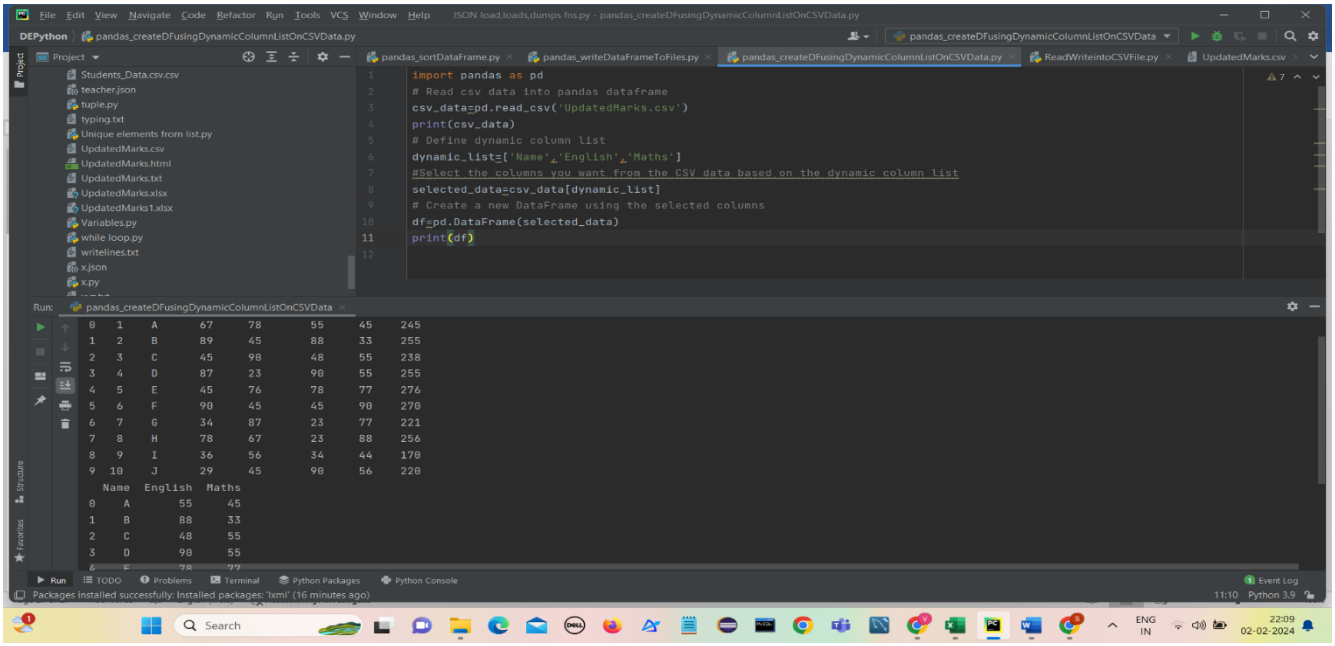
4) Html file

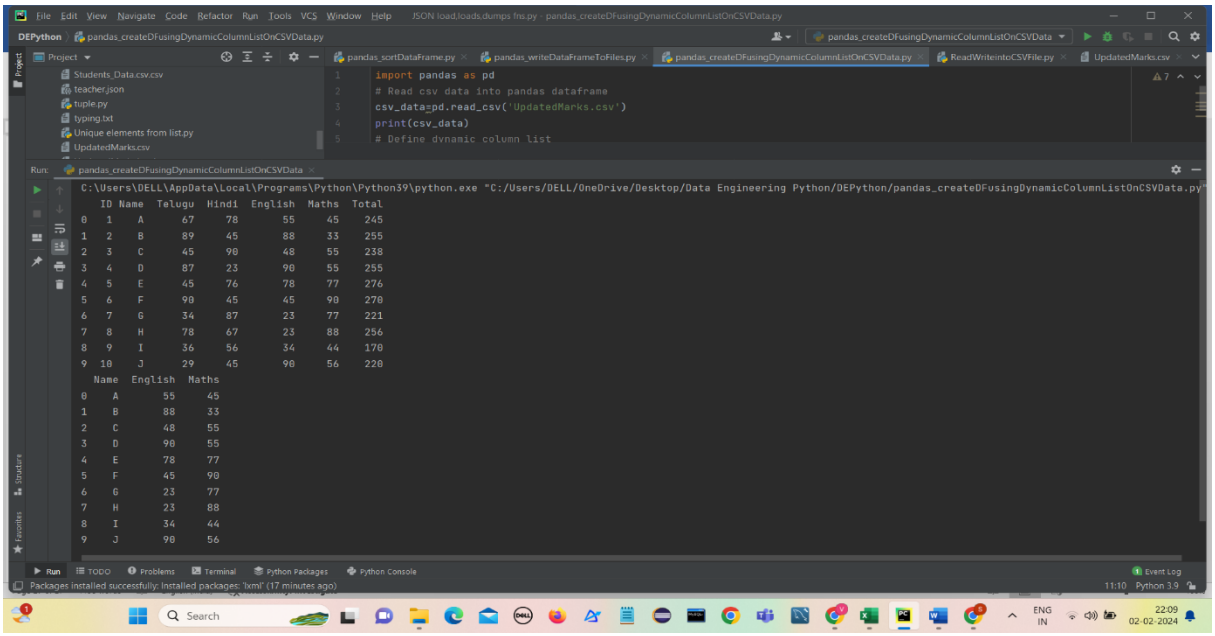


Create Data Frames using dynamic column list on csv data

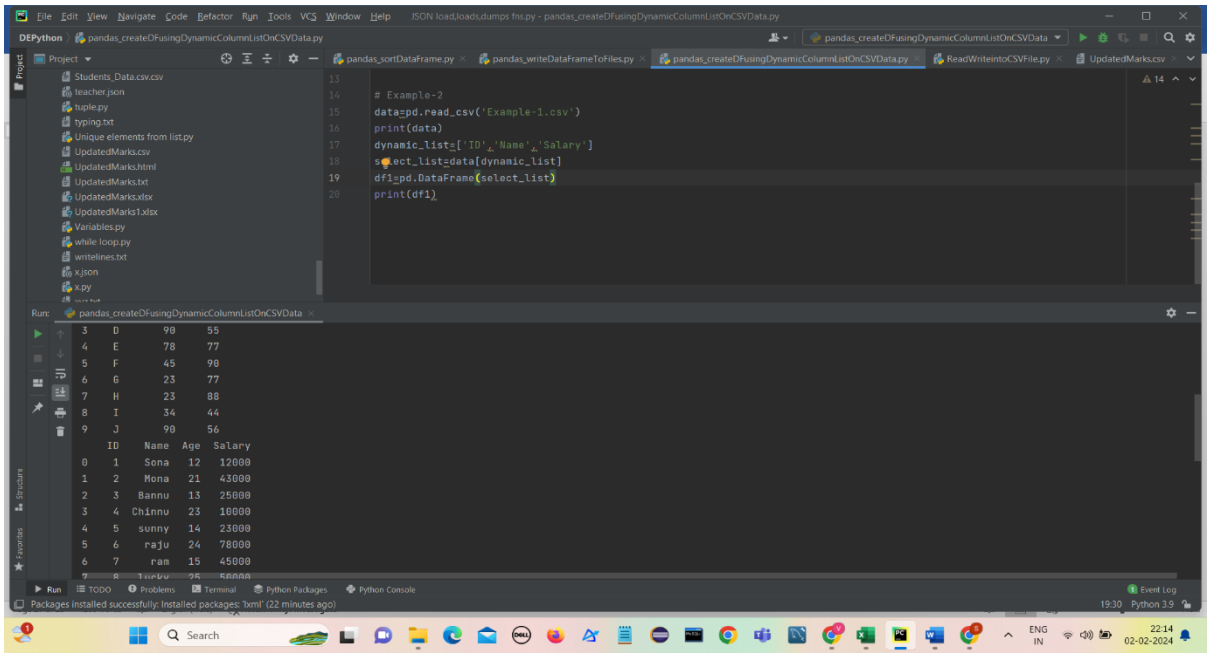
- Read the CSV data into a pandas DataFrame
- Define your dynamic column list
- Select the columns you want from the CSV data based on the dynamic column list
- Create a new DataFrame using the selected columns
- Display the new DataFrame

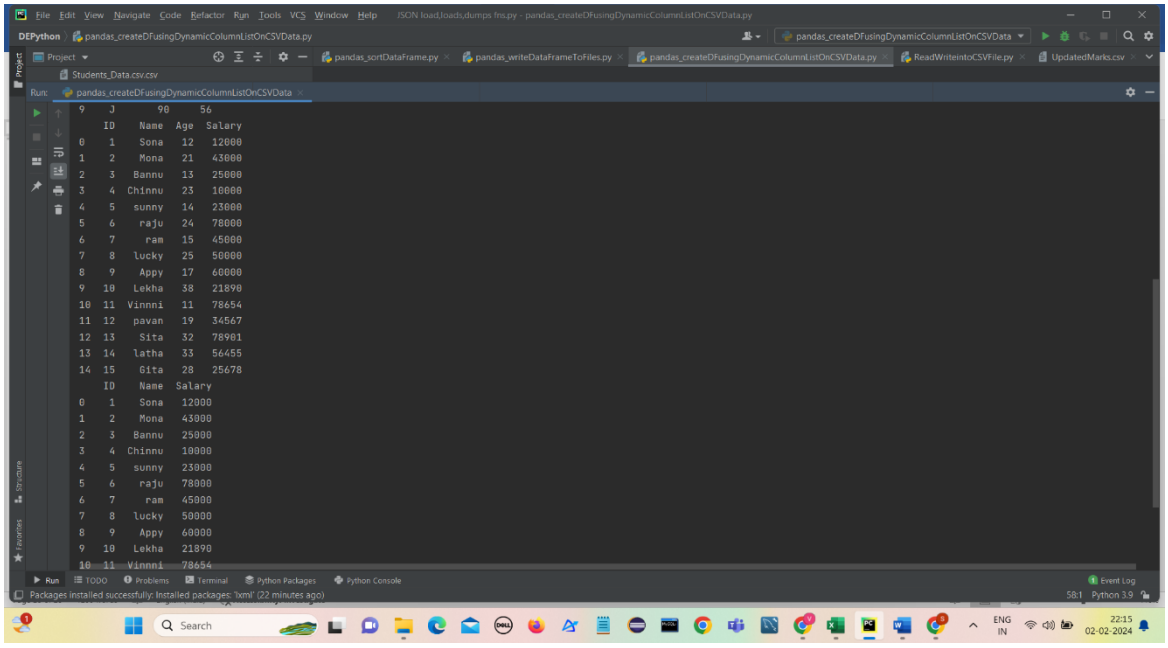
Example-1





Example-2


File Edit View Navigate Code Refactor Run Tools VCS Window Help
pandas_createDFusingDynamicColumnListOnCSVData.py
pandas_sortDataFrame.py
pandas_writeDataFrameToFiles.py
ReadWriteintoCSVFile.py
UpdatedMarks.csv
Students_Data.csv.csv
teacher.json
tuple.py
typing.txt
Unique elements from list.py
UpdatedMarks.csv
UpdatedMarks.html
UpdatedMarks.txt
UpdatedMarks.xlsx
UpdatedMarks1.xlsx
Variables.py
while loop.py
writelines.txt
x.json
x.py
# Example-2
data=pd.read_csv('Example-1.csv')
print(data)
dynamic_list=['ID','Name','Salary']
select_list=data[dynamic_list]
df1=pd.DataFrame(select_list)
print(df1)
3 D 90 55
4 E 78 77
5 F 45 90
6 G 23 77
7 H 23 88
8 I 34 44
9 J 90 56
ID Name Age Salary
0 1 Sona 12 12000
1 2 Mona 21 43000
2 3 Bannu 13 25000
3 4 Chinnu 23 10000
4 5 sunny 14 23000
5 6 raju 24 78000
6 7 ram 15 45000
Run
TODO
Problems
Terminal
Python Packages
Python Console
Event Log
Packages installed successfully: Installed packages: 'lxml' (22 minutes ago)
19:30 Python 3.9
Search
ENG IN
22:14 02-02-2024


File Edit View Navigate Code Refactor Run Tools VCS Window Help
pandas_createDFusingDynamicColumnListOnCSVData.py
pandas_sortDataFrame.py
pandas_writeDataFrameToFiles.py
ReadWriteintoCSVFile.py
UpdatedMarks.csv
Students_Data.csv.csv
9 J 90 56
ID Name Age Salary
0 1 Sona 12 12000
1 2 Mona 21 43000
2 3 Bannu 13 25000
3 4 Chinnu 23 10000
4 5 sunny 14 23000
5 6 raju 24 78000
6 7 ram 15 45000
7 8 lucky 25 50000
8 9 Appy 17 60000
9 10 Lekha 38 21890
10 11 Vinnni 11 78654
11 12 pavan 19 34567
12 13 Sita 32 78901
13 14 latha 33 56455
14 15 Gita 28 25678
ID Name Salary
0 1 Sona 12000
1 2 Mona 43000
2 3 Bannu 25000
3 4 Chinnu 10000
4 5 sunny 23000
5 6 raju 78000
6 7 ram 45000
7 8 lucky 50000
8 9 Appy 60000
9 10 Lekha 21890
10 11 Vinnni 78654
Run
TODO
Problems
Terminal
Python Packages
Python Console
Event Log
Packages installed successfully: Installed packages: 'lxml' (22 minutes ago)
58:1 Python 3.9
Search
ENG IN
22:15 02-02-2024