**Pyspark Coding Assessment:** **Abhiram Basa - 27-12-23**

**Questions:**

1. Implement Processing JSON and CSV data with PySpark
2. Explain ETL (Extract, Transform, Load) with PySpark
3. Using Spark SQL - Creating databases, tables
4. Using Spark SQL - Transformations such as Filter, Join, Simple Aggregations, GroupBy.

**Answers:**

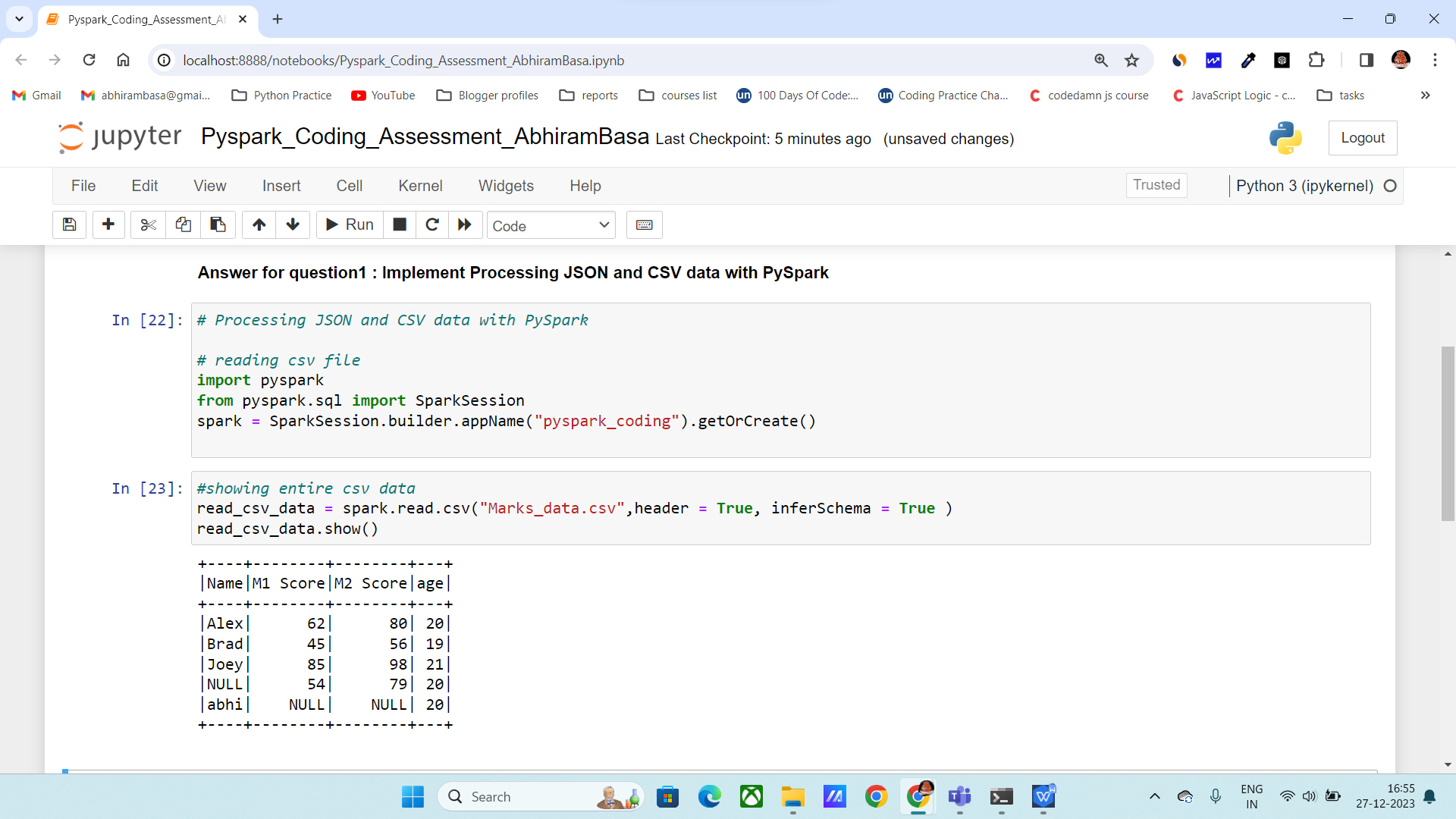
1. **Implement Processing JSON and CSV data with PySpark**

**Solution for 1** : We can read the csv files using the below syntax

**Syntax:** Data1 = data1.read.csv(“filename.csv”)

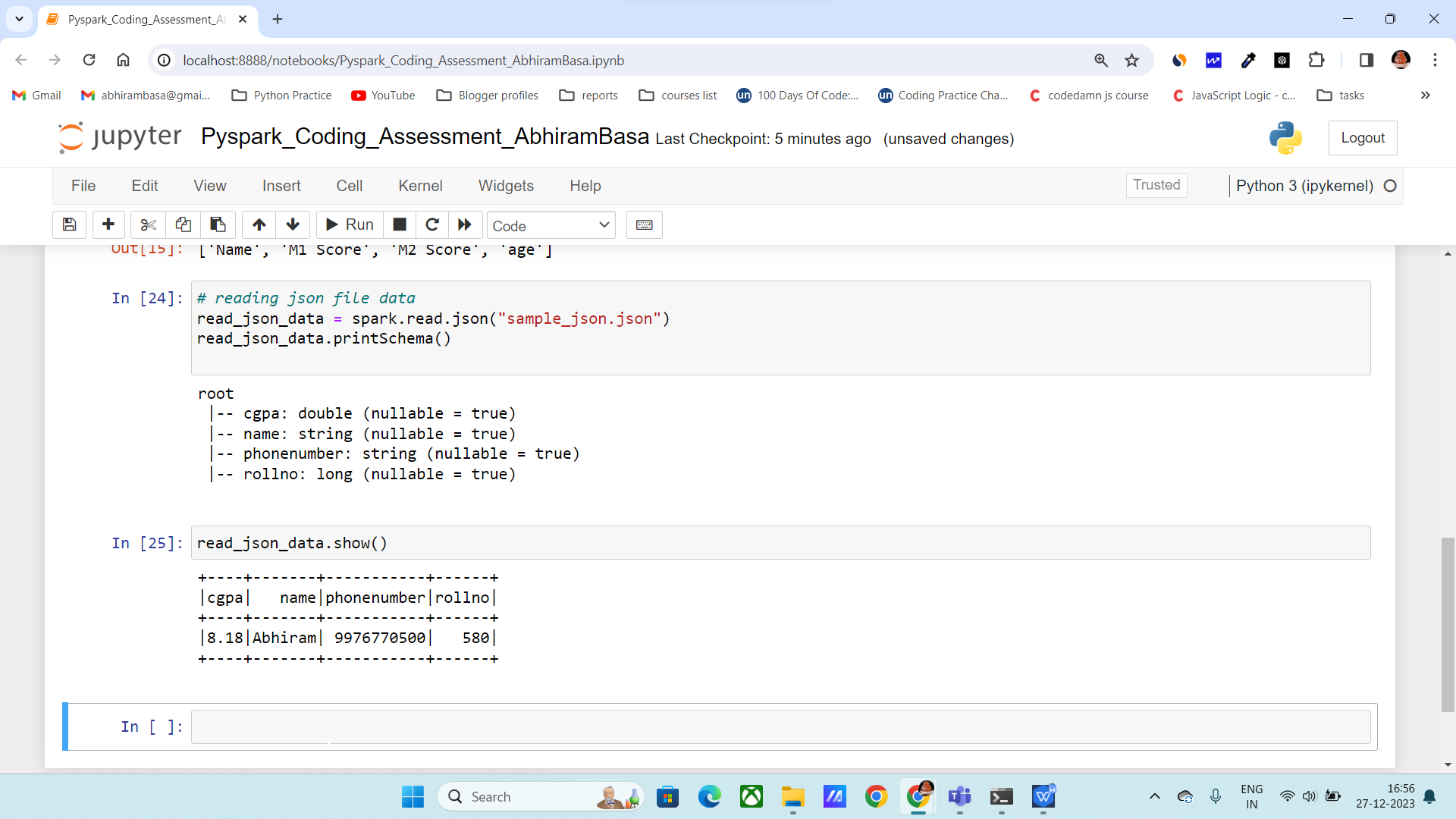
**Csv file processing:**

**Data1.show() :** It will print the output



**Json file processing:**

**Syntax:** Data2 = data1.read.json(“filename.json”)



1. **Explain ETL (Extract, Transform, Load) with PySpark**

**Solution 2 :**

ETL stands for

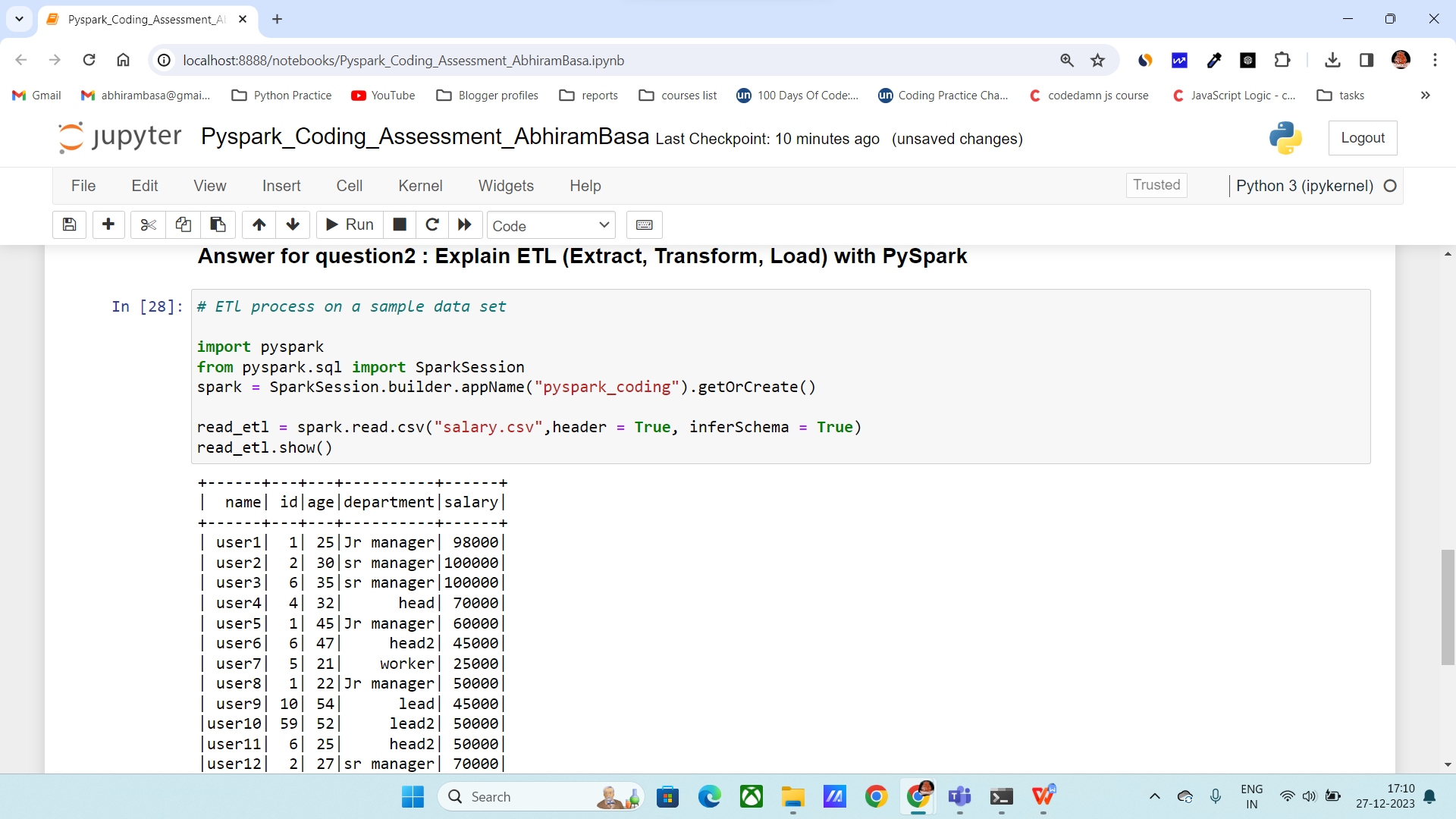
**Extract**: extract the data from the different sources

**Transform**: Transform the unstructured data into structured data. Transformations like cleaning, manipulation, etc.

**Load**: Load the transformed data into a location or date warehouse.

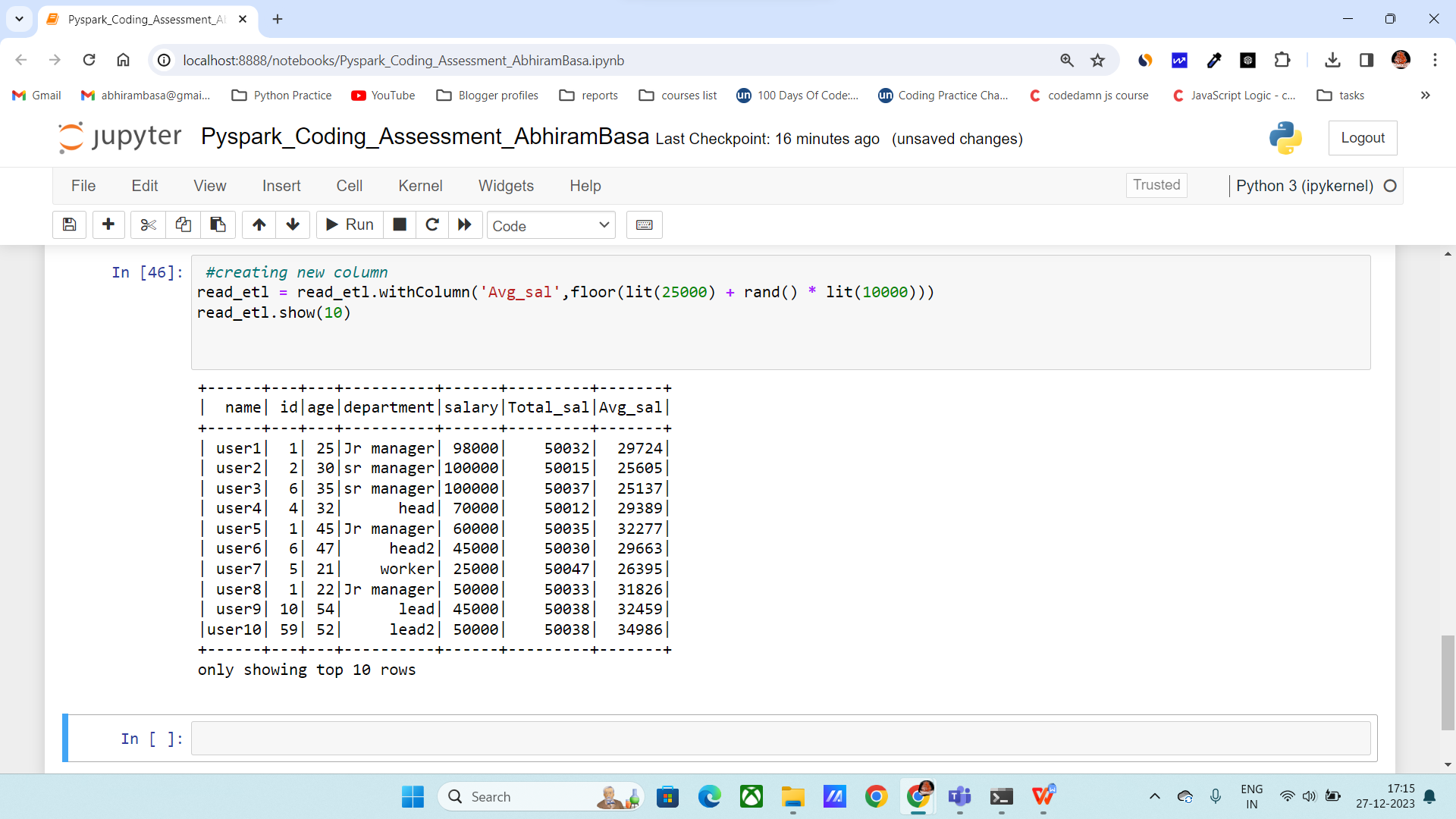
Below we performed ETL on sample **CSV** data.

we will load the spark session using **from pyspark.sql import SparkSession.**



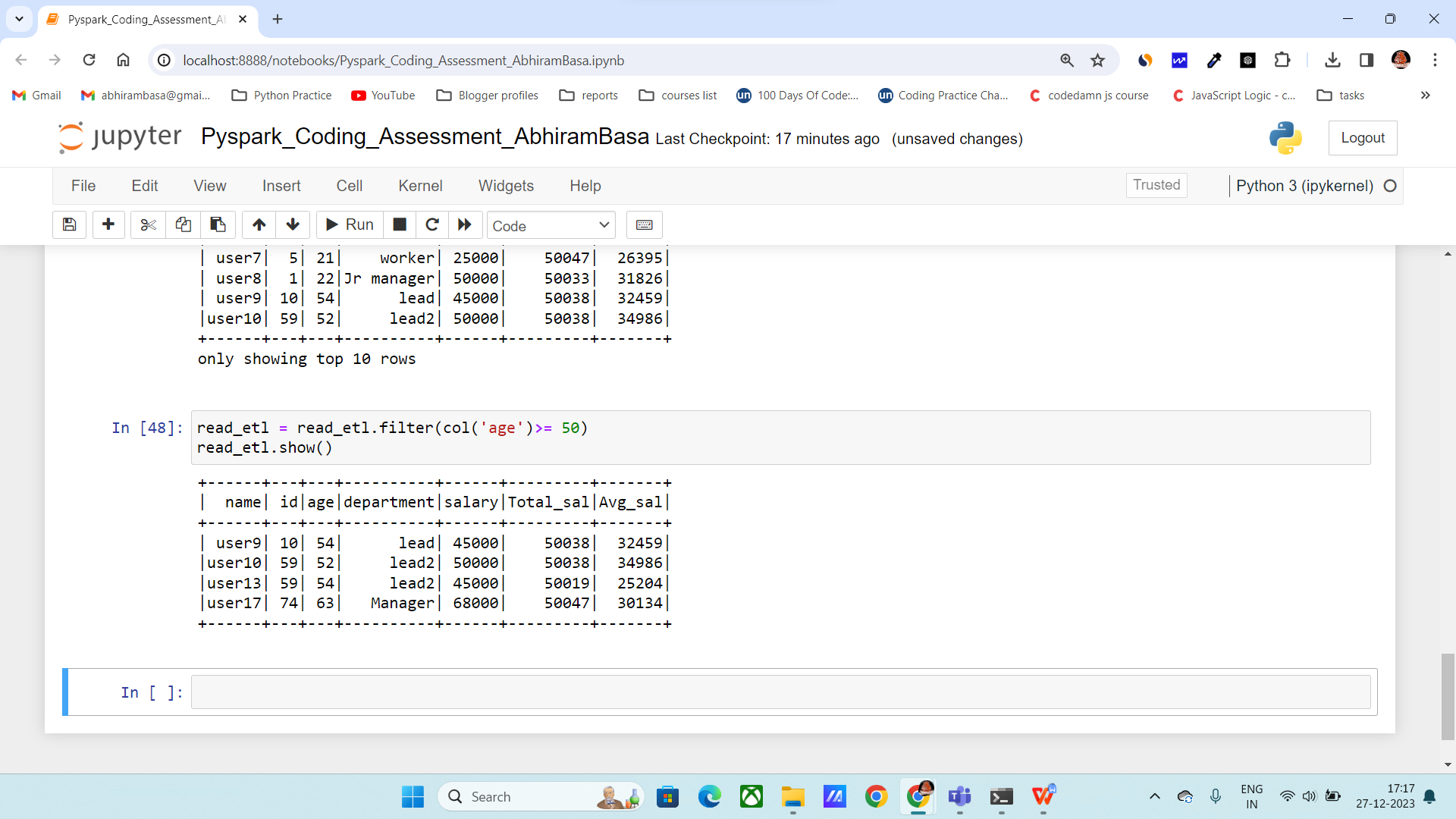
**Creating new columns:**

Syntax: data1.withcolumn(“new\_column\_name”, condition)



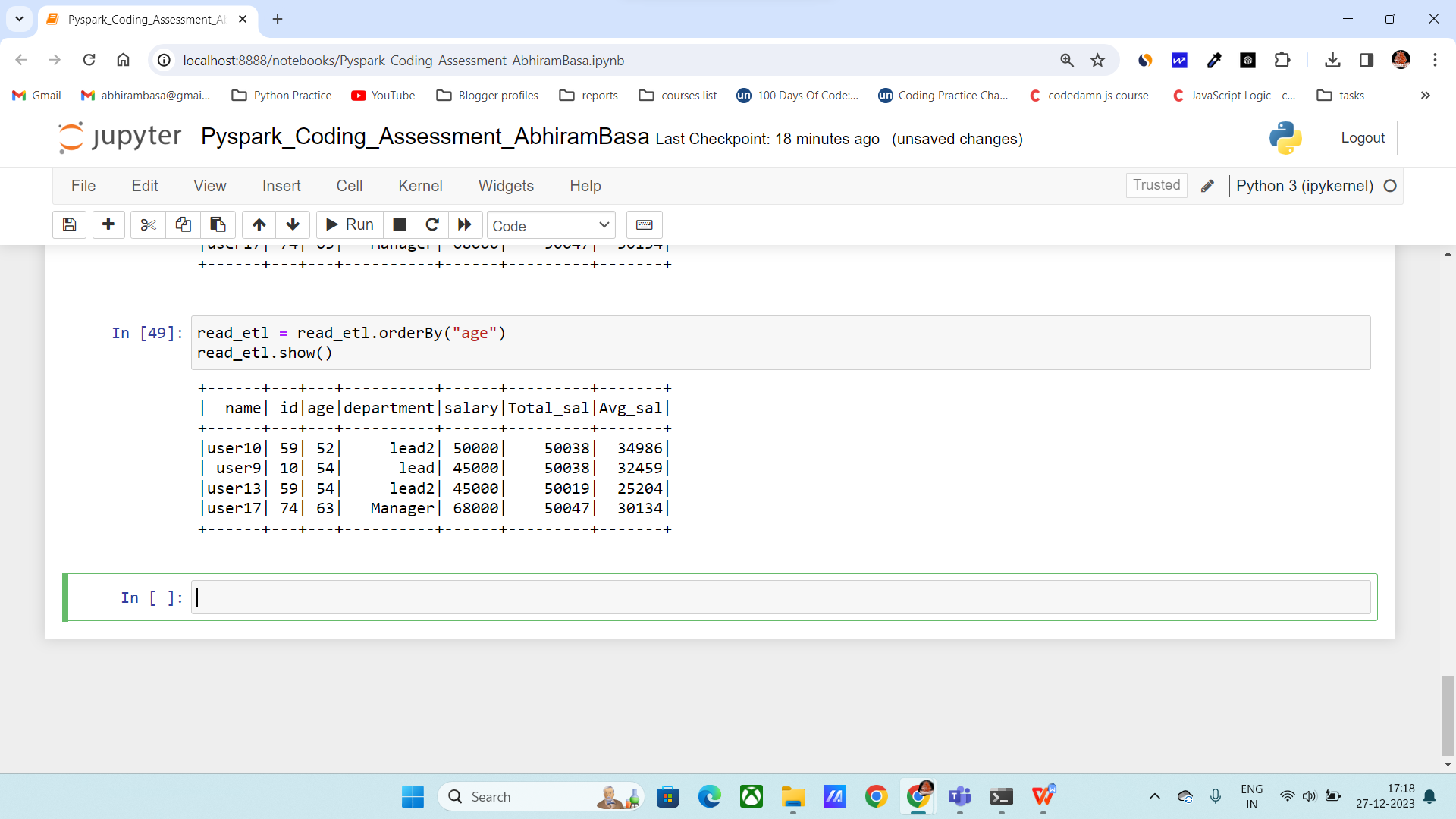
**Filter():**

Syntax: data1. filter(condition)



**Orderby:**

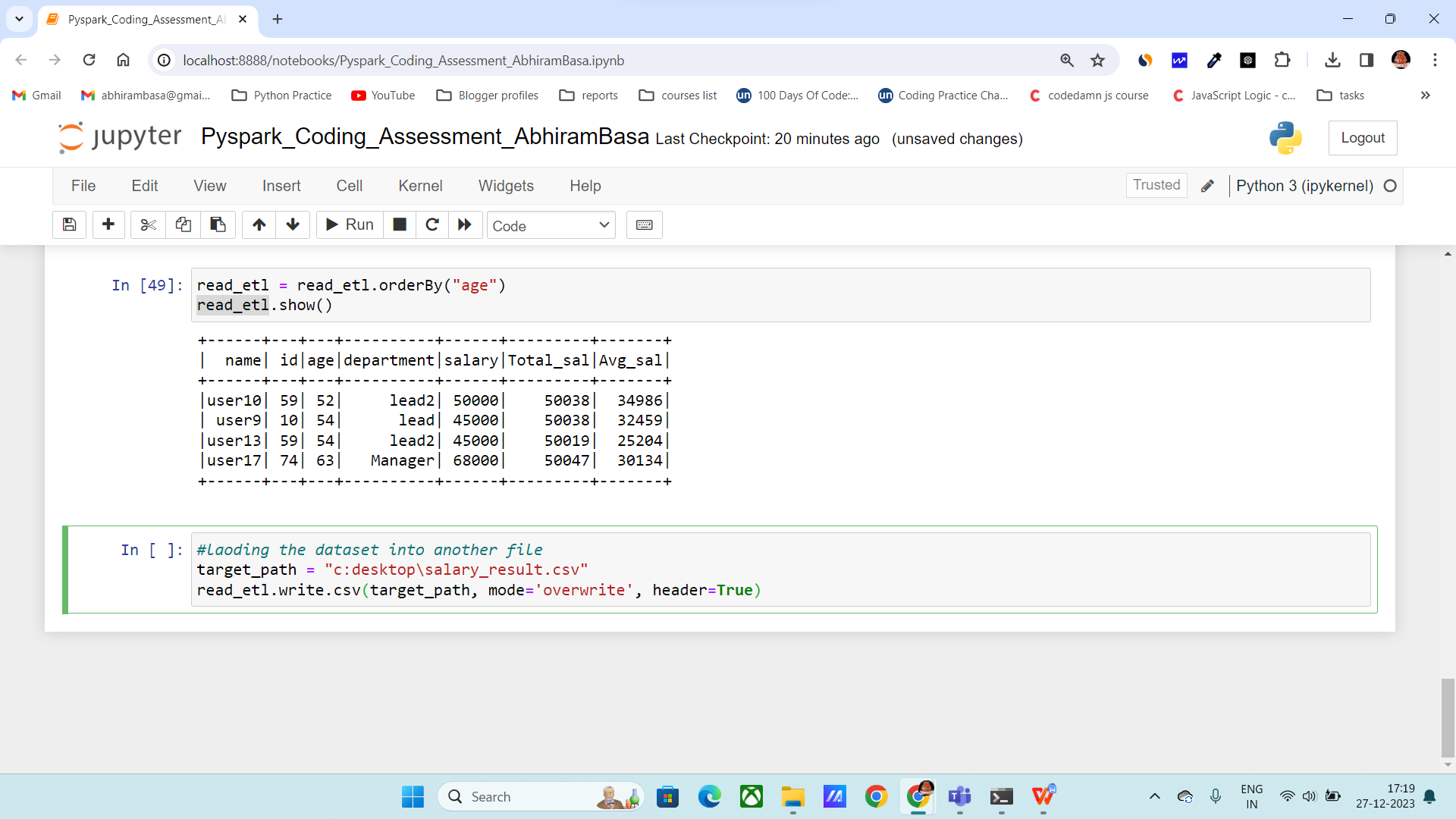
Syntax: data1.orderBy(“column\_name”)



**Loaded into another csv file:**

We can load the data into another csv file using

Syntax: data1.write.csv(target\_path, mode='overwrite', header=True)



1. **Using Spark SQL - Creating databases, tables**

**Solution 3 :**

**Creating a database :**

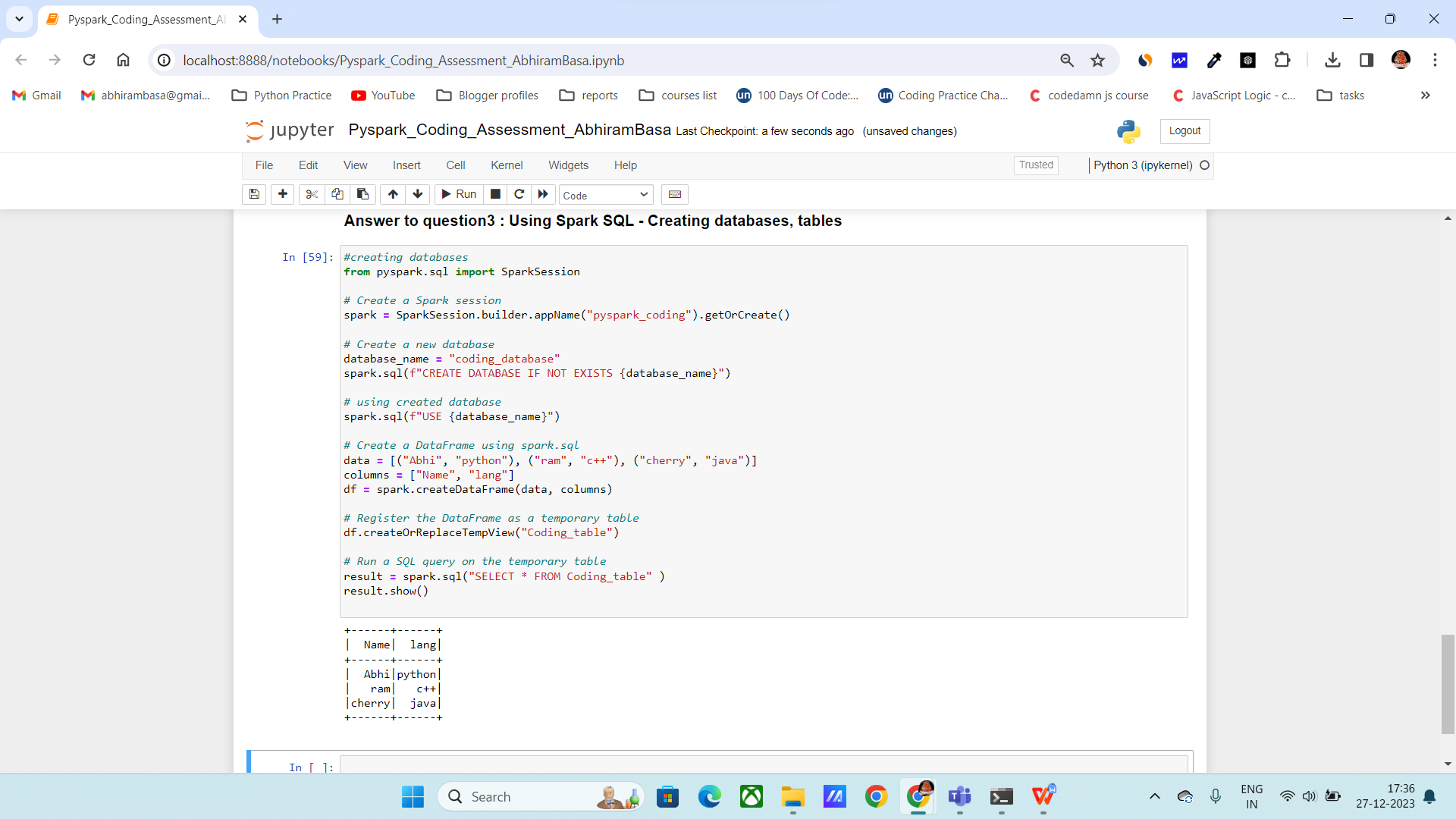
Syntax:

database\_name = "new\_database"

spark.sql(f” {database\_name}")

The above command will use the database specified

**Creating a table in pyspark:**



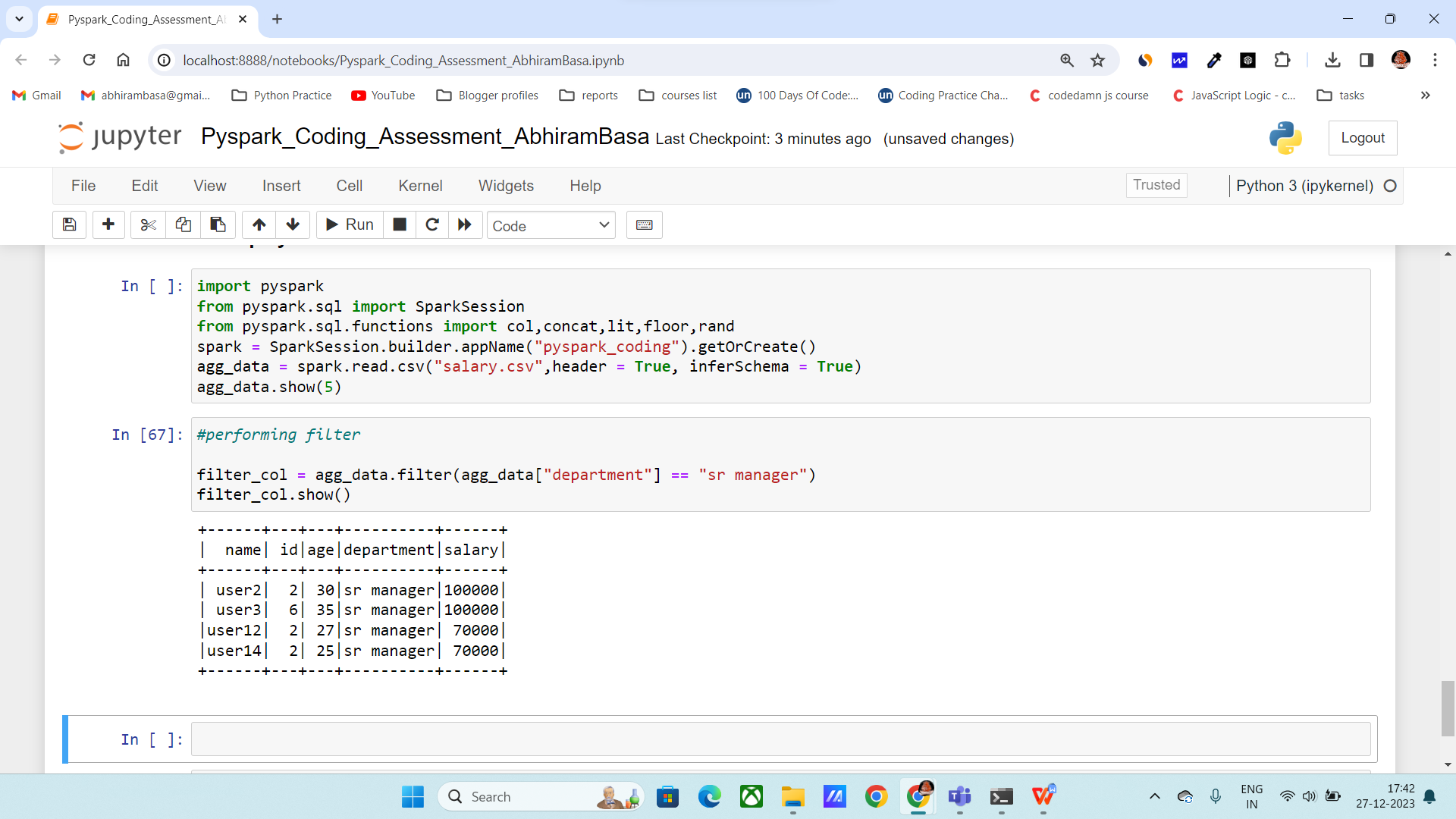
1. **Transformations such as Filter, Join, Simple Aggregations, GroupBy.**

**Solution 4:**

**Filter:** it will filter the data based on the specified condition.

Syntax: data1.filter(condition)

Performing filter ()

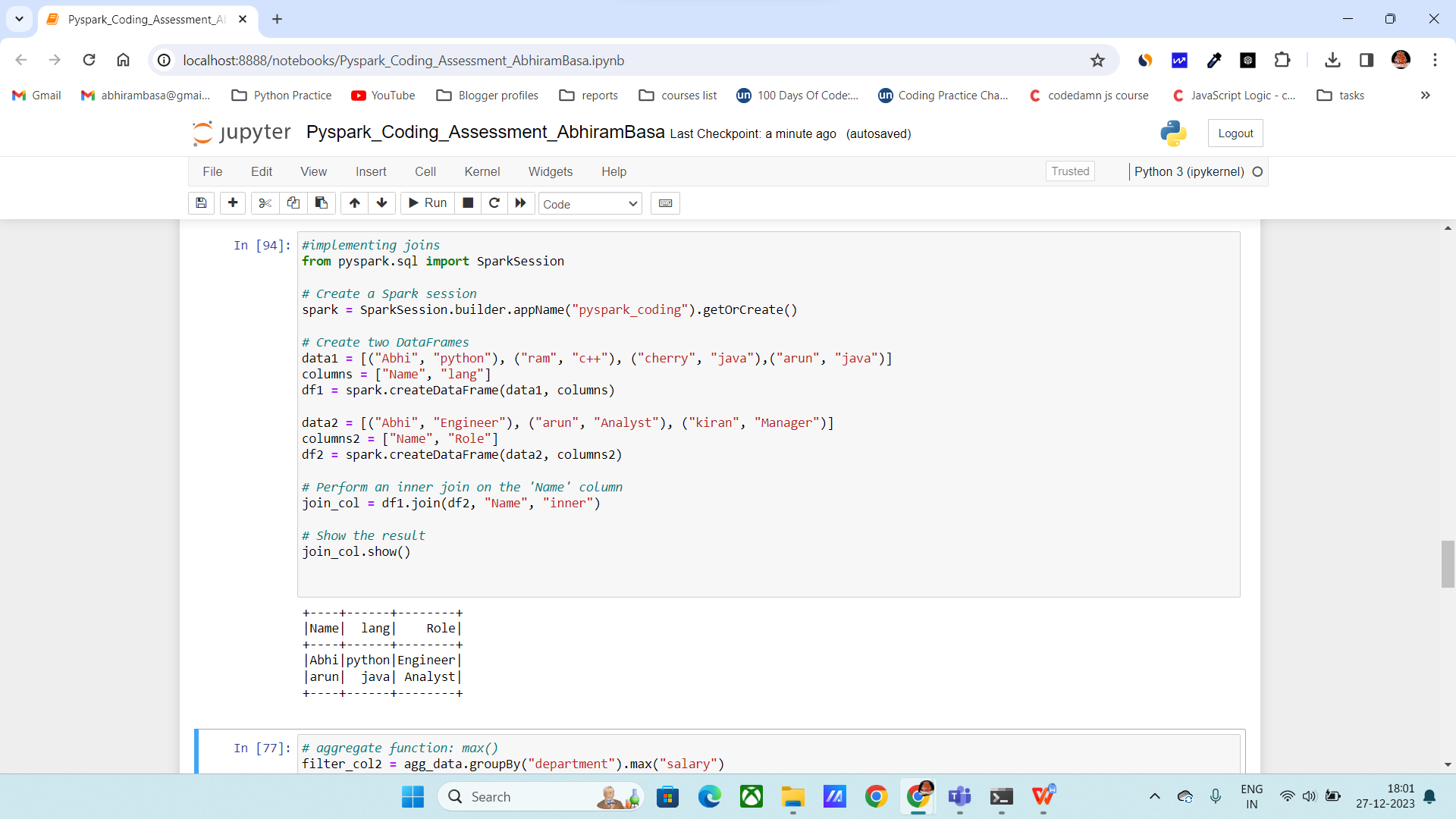


**Performing joins:**

Below I am performing inner join

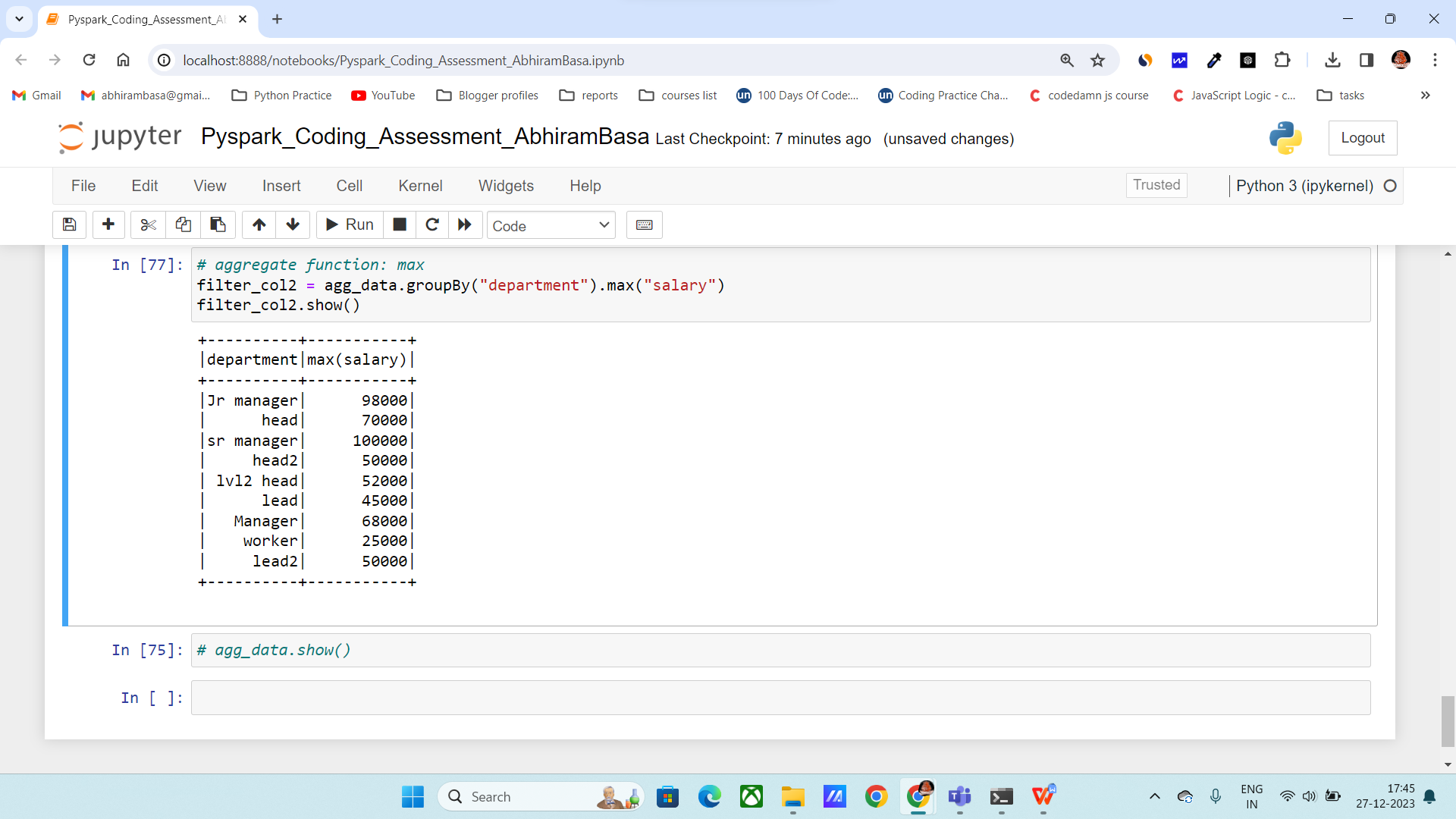
Inner join: It will gives the data which is matching in both the tables

**Syntax:** df1. join(df2, "Name", "inner")

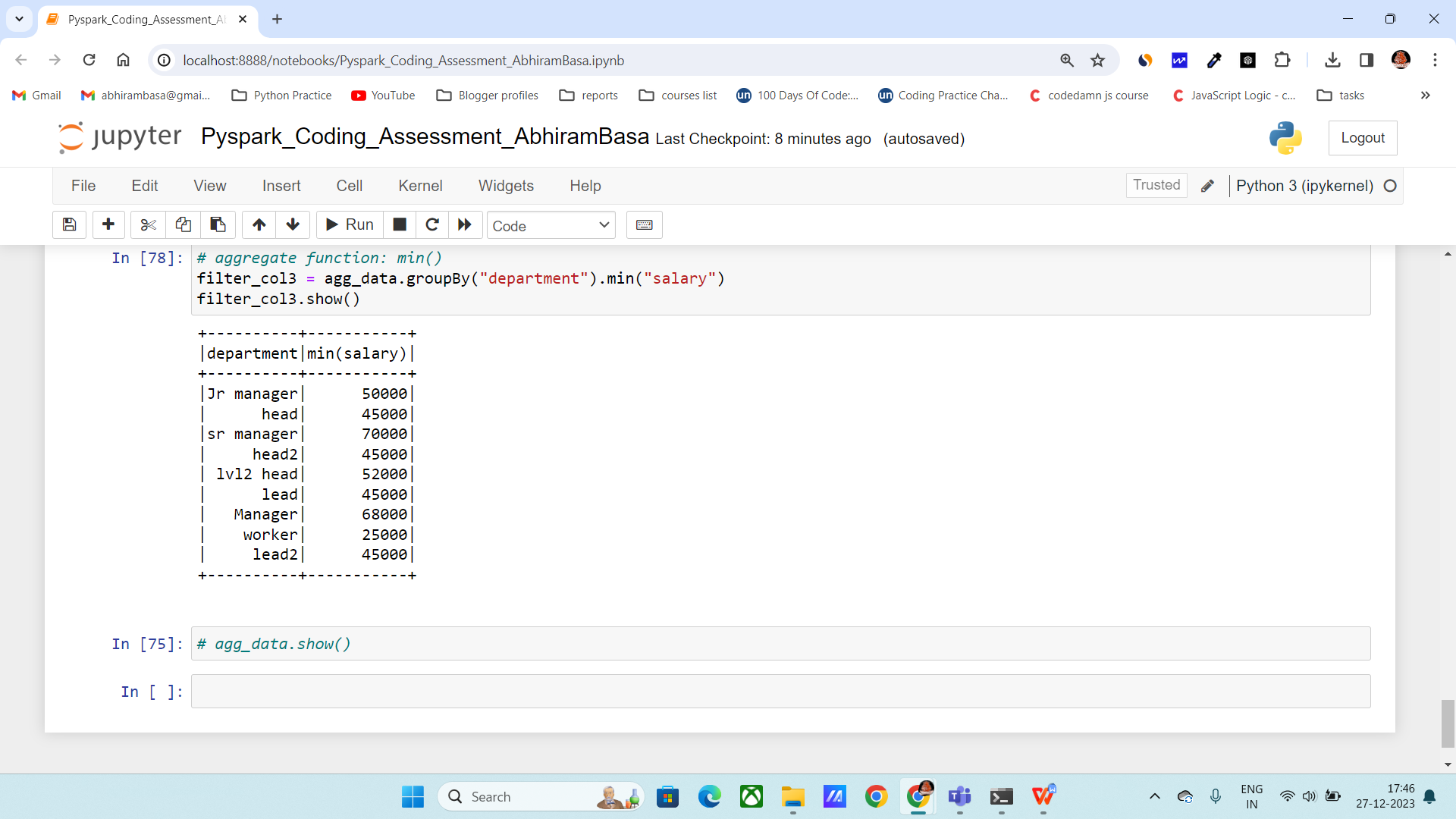


**Aggregate fun:**

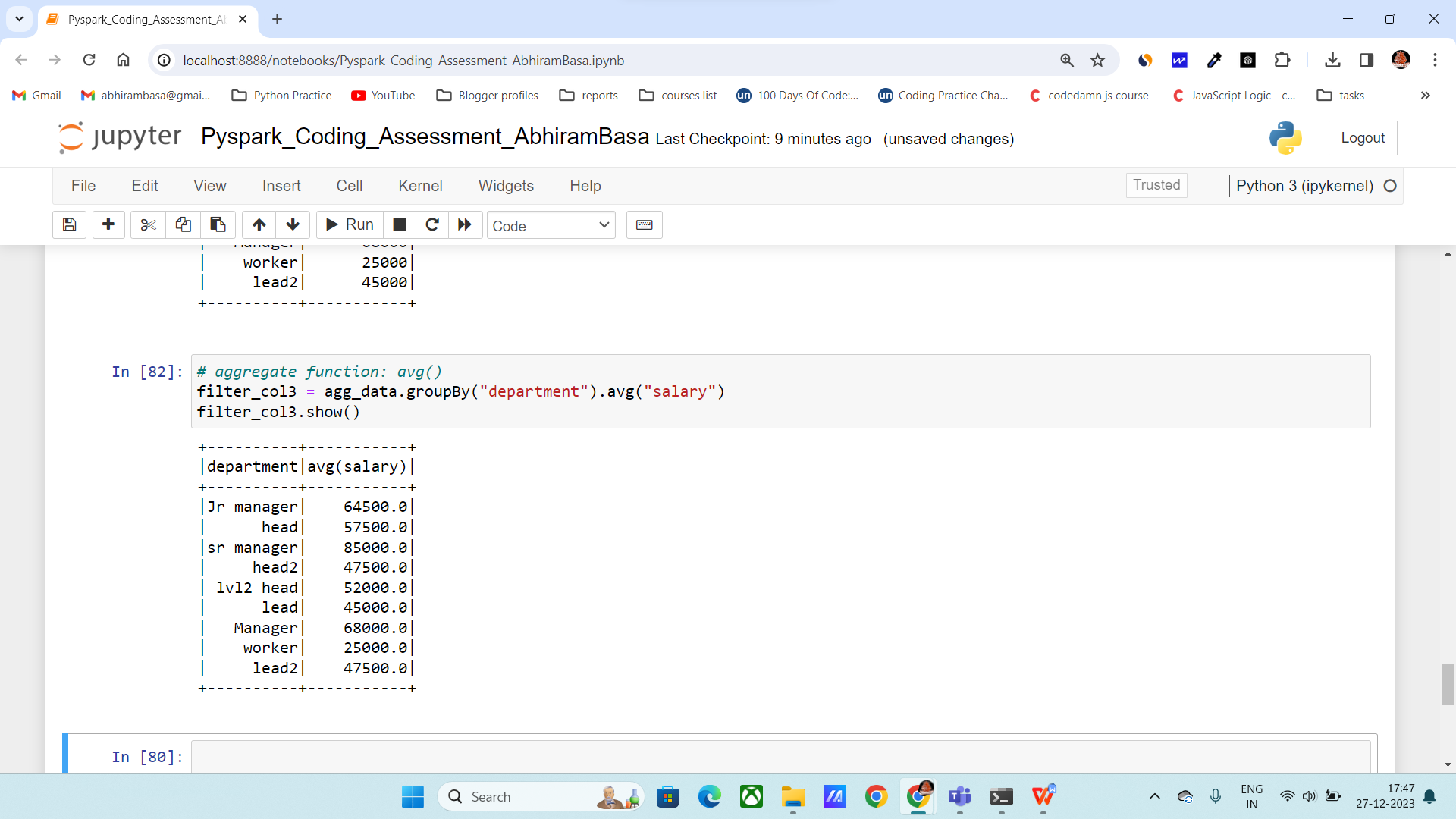
**max():** It will return the max of the salary column in below code.



**Min() :** It will return the min of the salary column in below code.



**Avg() :** It will return the avg of the salary column in below code.



**Sum() :** It will return the sum of the salary column in below code.

Here it will group by department

