**Day-3 PySpark**

* **RDD Operations**

PySpark RDD has a set of operations to accomplish any task. These operations are of two types:

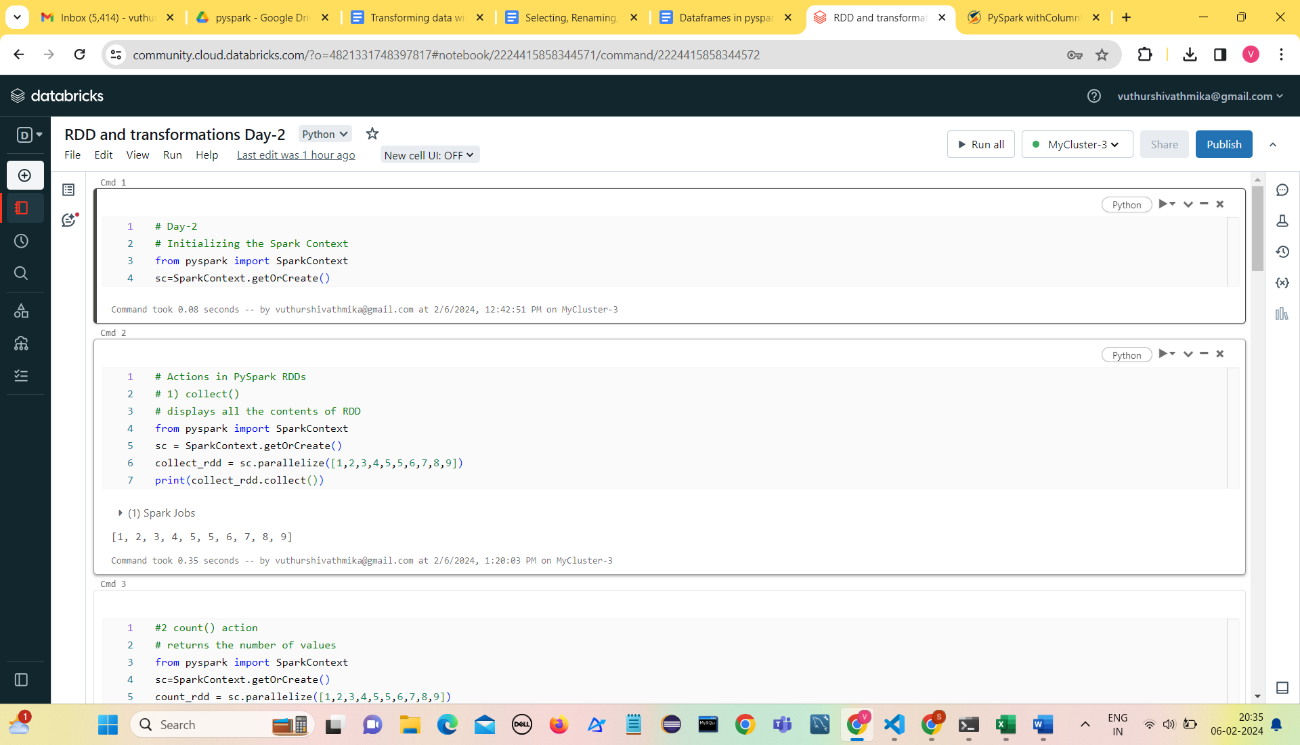
1. Transformations

2. Actions

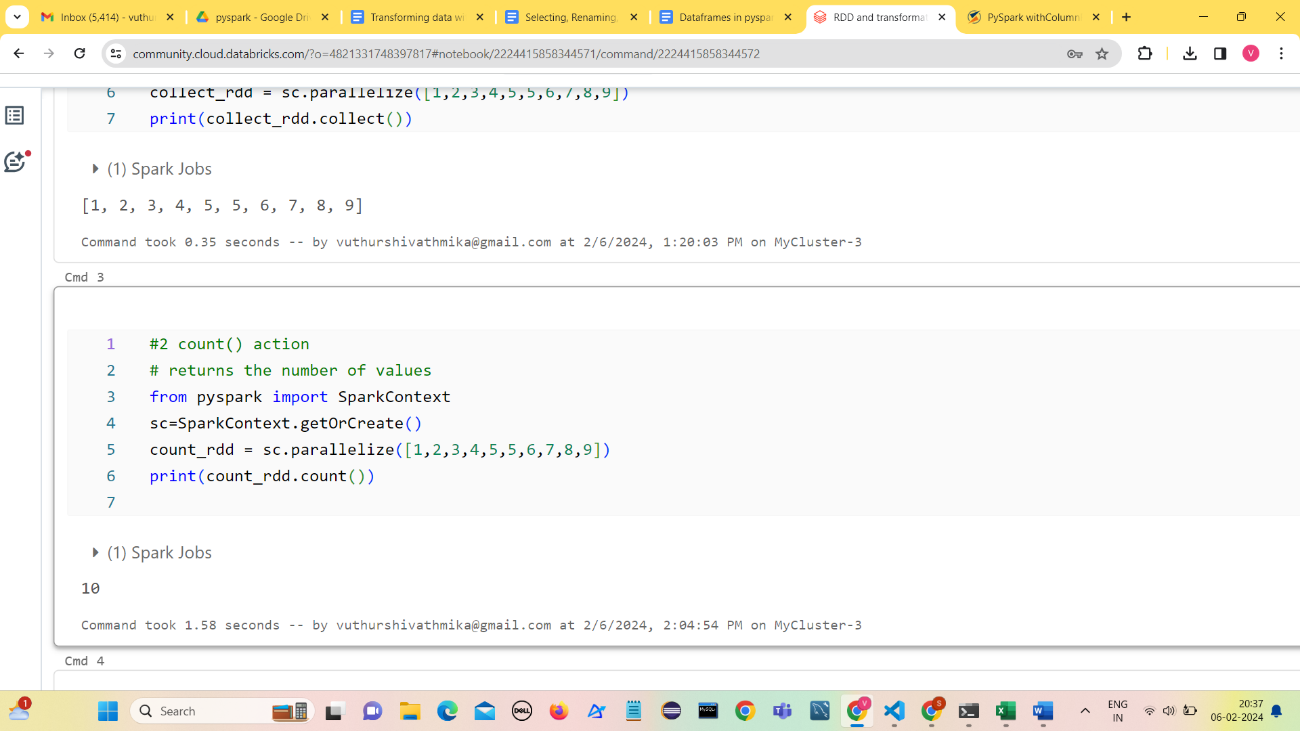
**1) Transformations**are a kind of operation that takes an RDD as input and produces another RDD as output

**2) Actions**are a kind of operation which are applied on an RDD to produce a single value.

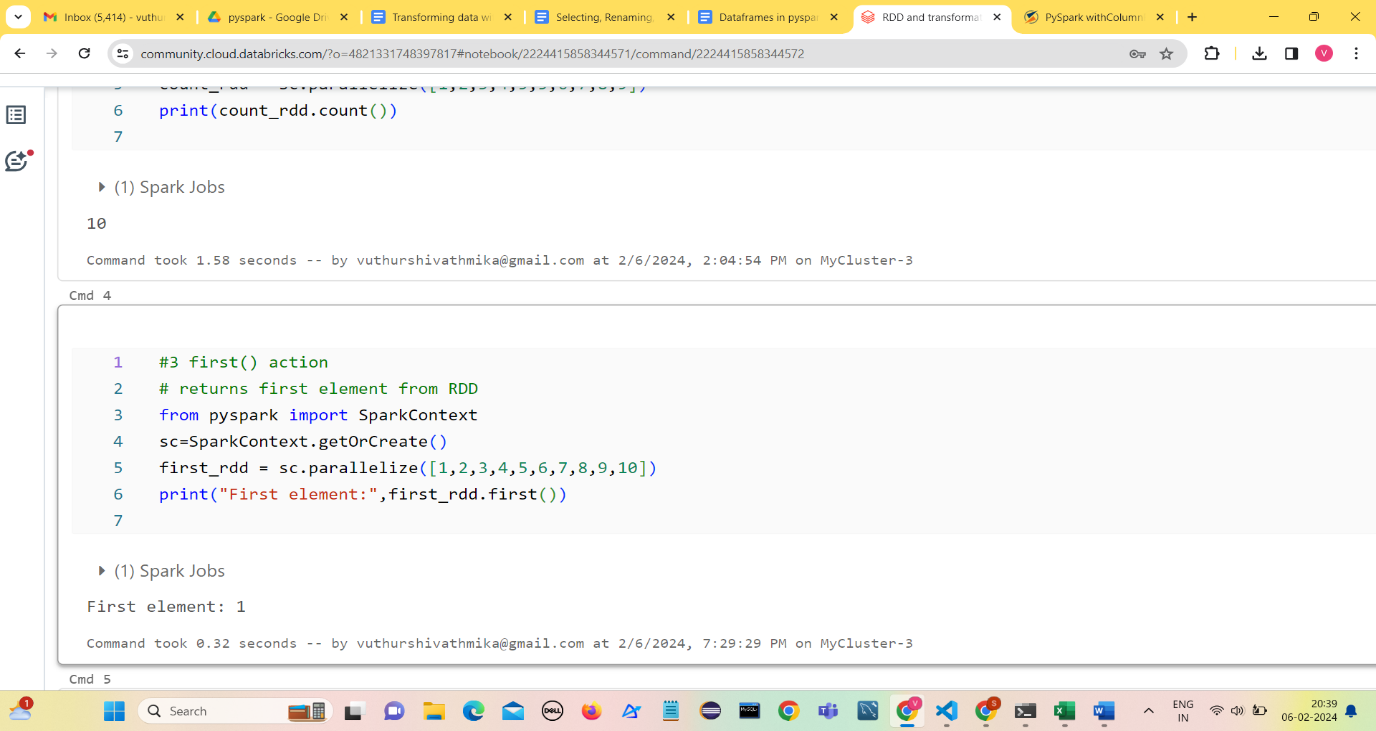
**a) Actions in PySpark RDDs**

1. **.collect() Action:** returns a list of all the elements of the RDD.
2. **.count() Action**

The **.count()** action on an RDD is an operation that returns the number of elements of our RDD.

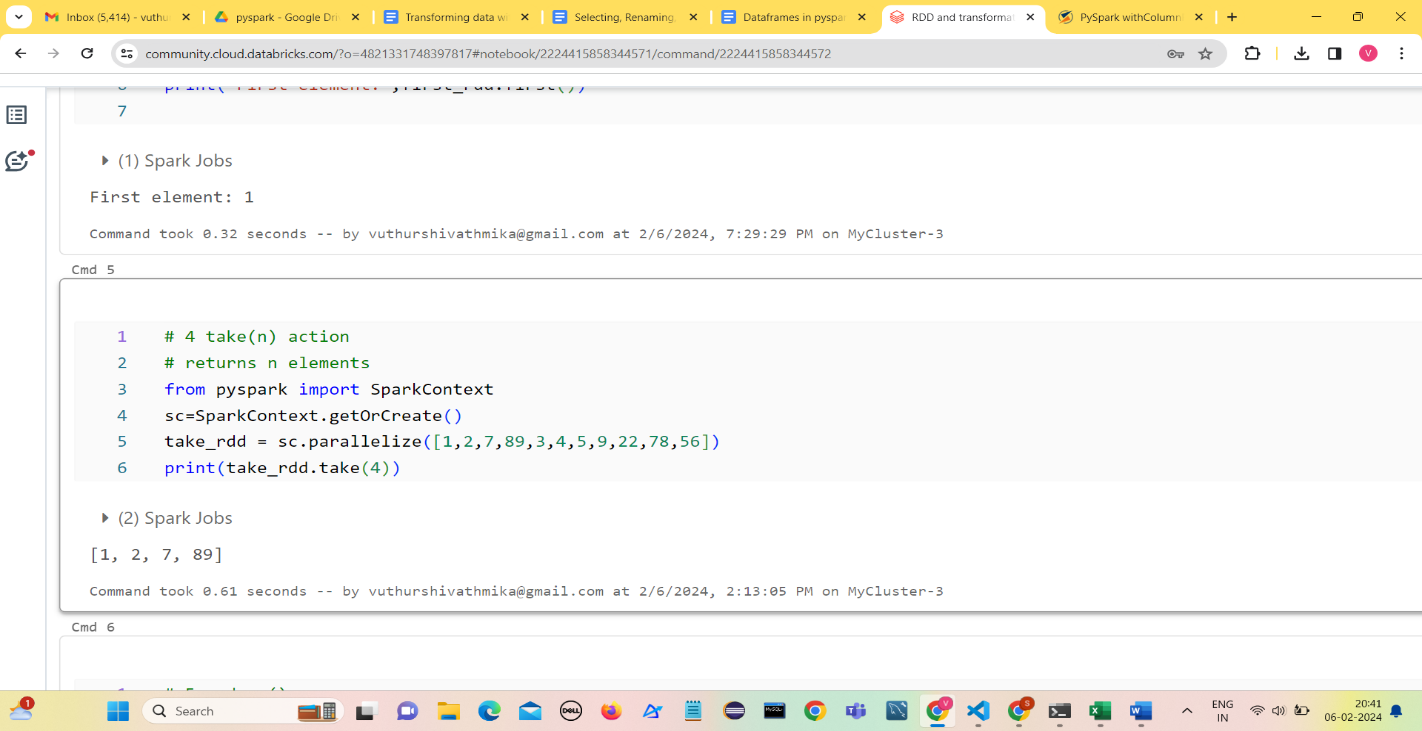


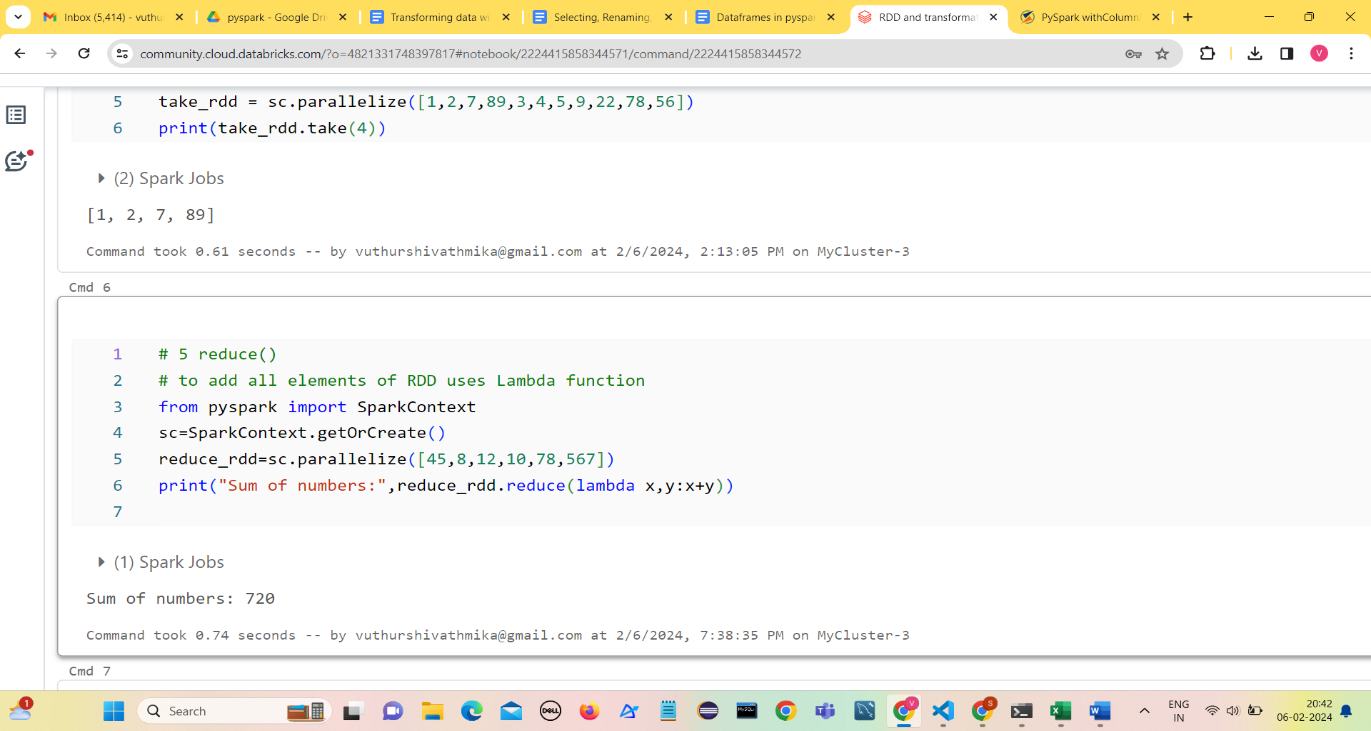
1. .**first() Action**

It returns the first element from our RDD.

1. **The .take() Action**

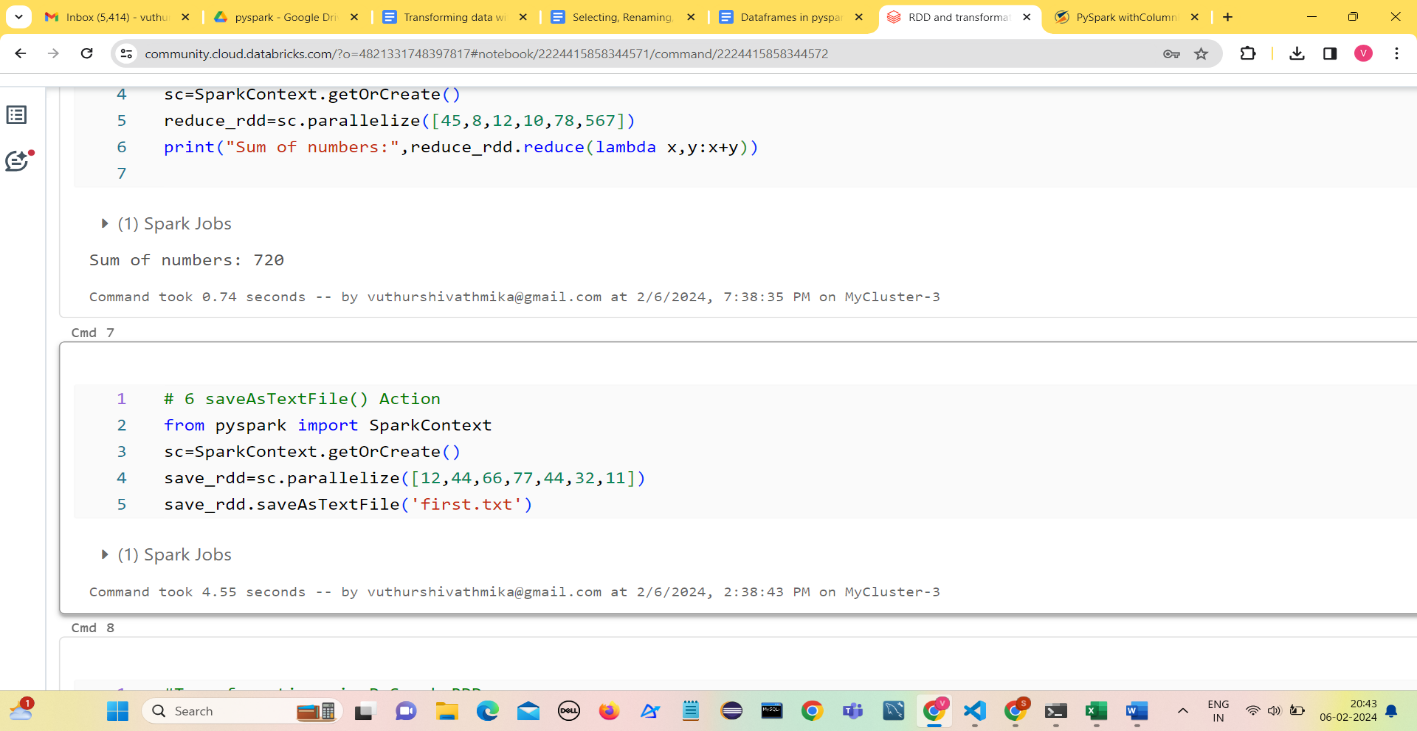
The **.take(n)** action on an RDD returns n number of elements from the RDD. The ‘n’ argument takes an integer which refers to the number of elements we want to extract from the RDD.

1. **The .reduce() Action**

The **.reduce()** Action takes two elements from the given RDD and operates. This operation is performed using an anonymous function or lambda.

1. **.saveAsTextFile() Action**

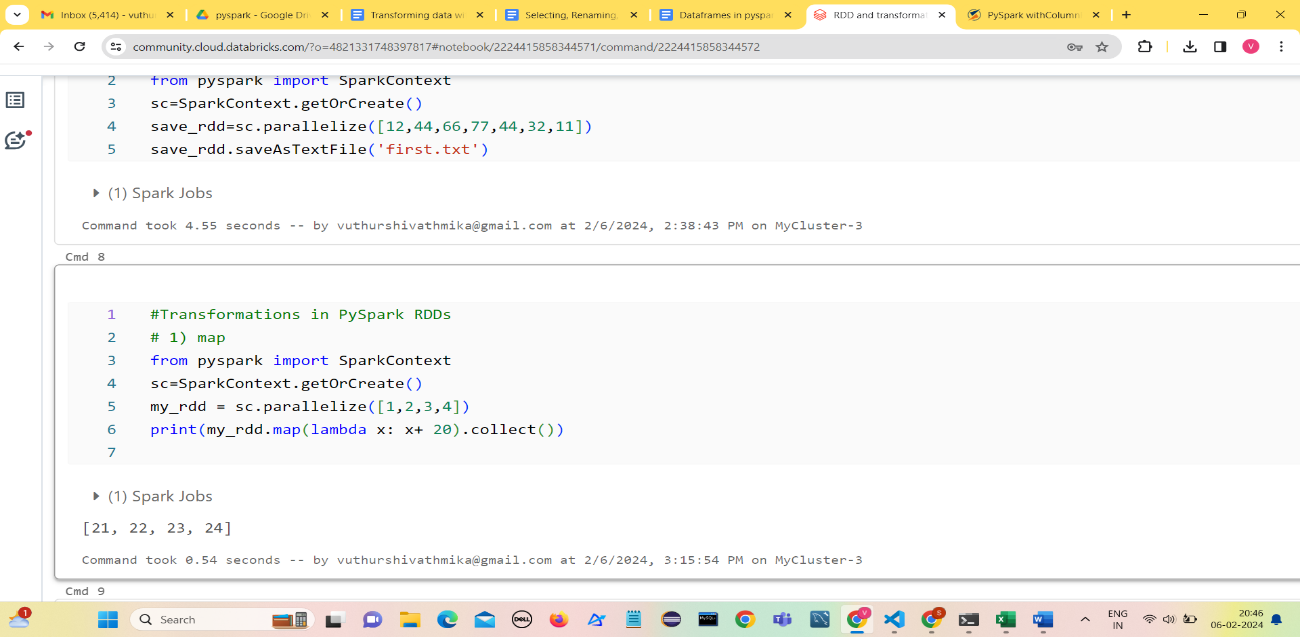
**.saveAsTextFile()** Action is used to serve the resultant RDD as a text file.



**b**) **Transformations in PySpark RDDs**

Transformations are the kind of operations that are performed on an RDD and return a new RDD.

1. **The .map() Transformation**

**.map()** transformation maps a value to the elements of an RDD. The**.map()** transformation takes in an anonymous function and applies this function to each of the elements in the RDD.

1. **The .filter() Transformation**

**.filter()** transformation is an operation in PySpark for filtering elements from a PySpark RDD.

The **.filter()** transformation takes in an anonymous function with a condition.

1. **The .union() Transformation**

 **.union()** transformation combines two RDDs and returns the union of the input two RDDs.

1. **.flatMap() Transformation**

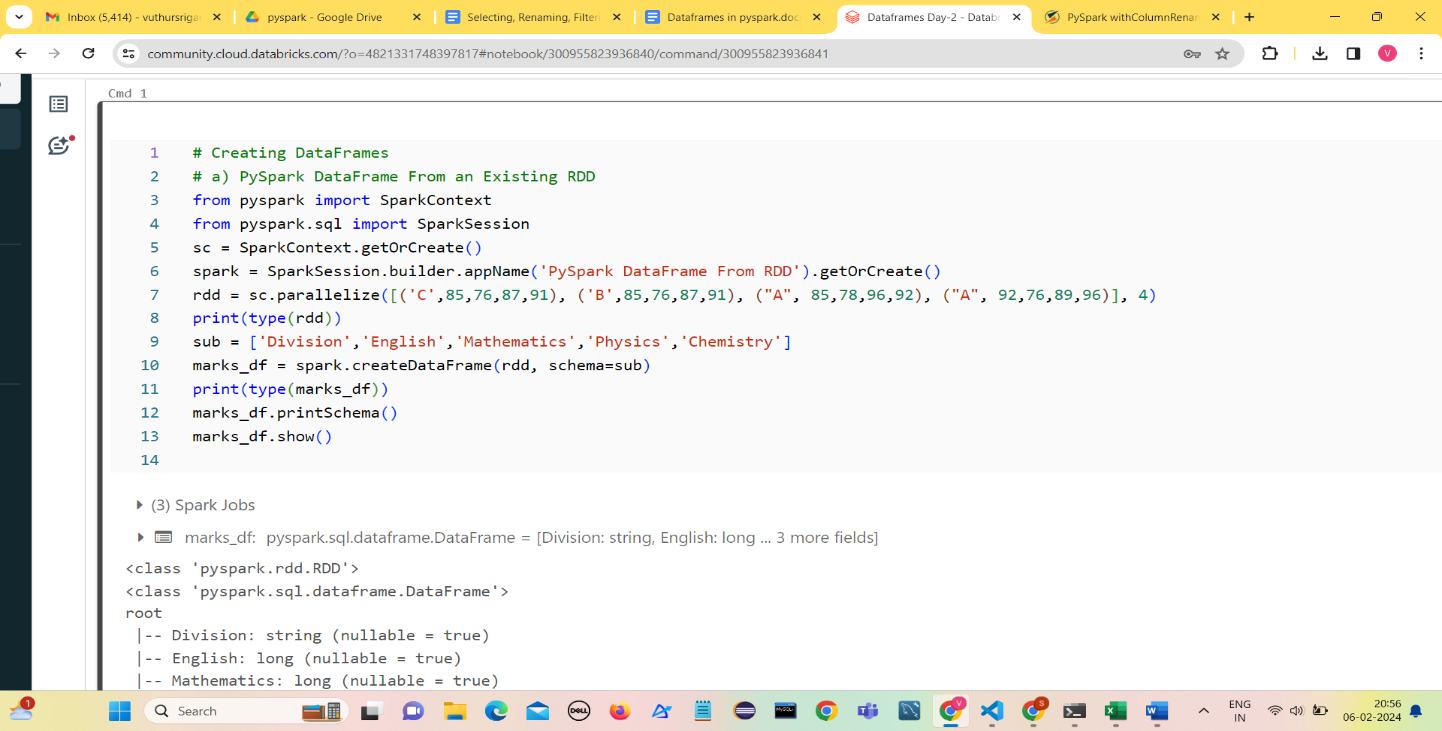
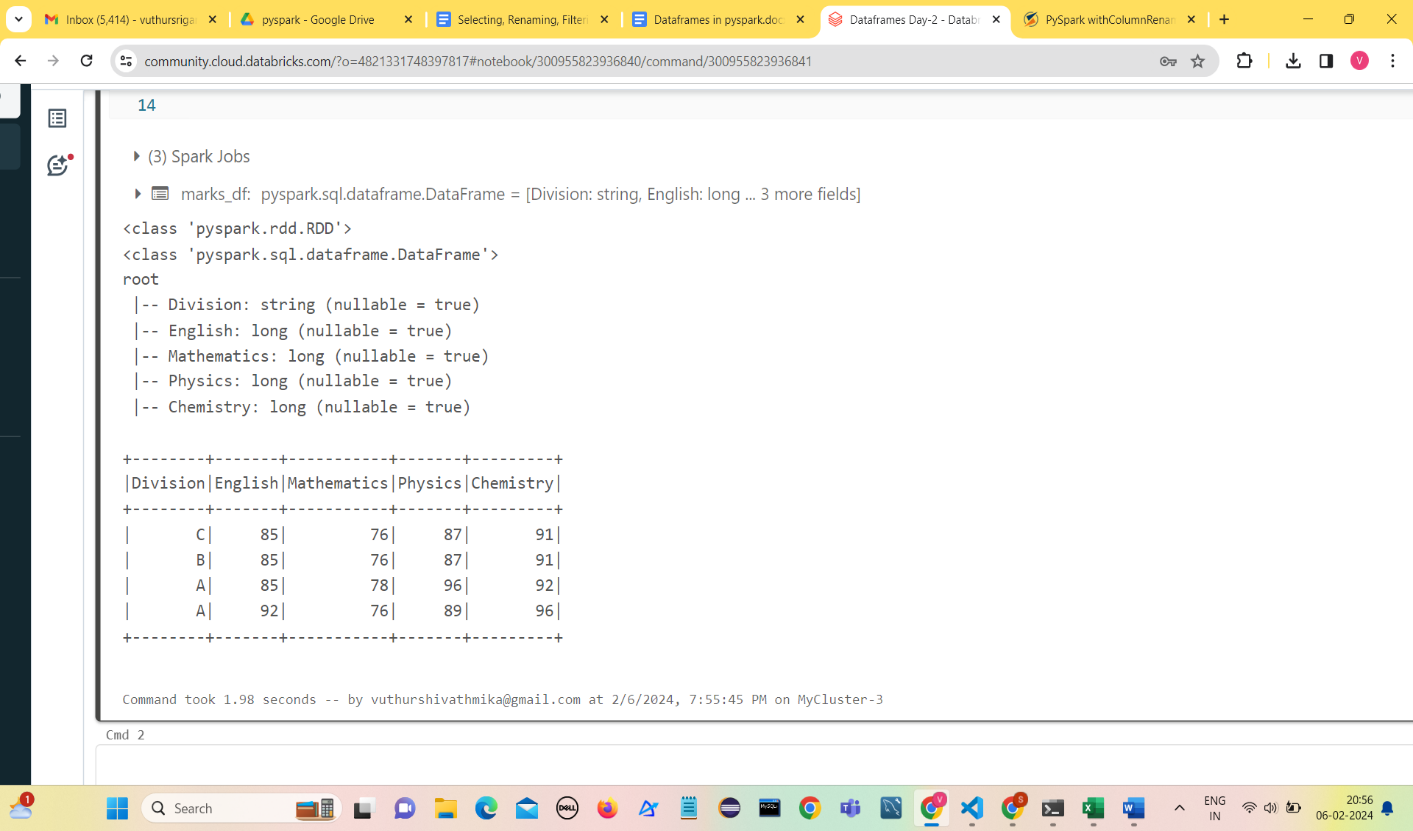
**.flatMap()** transformation peforms same as the **.map()** transformation except the fact that **.flatMap()** transformation return seperate values for each element from original RDD.

* **Creating dataframes**

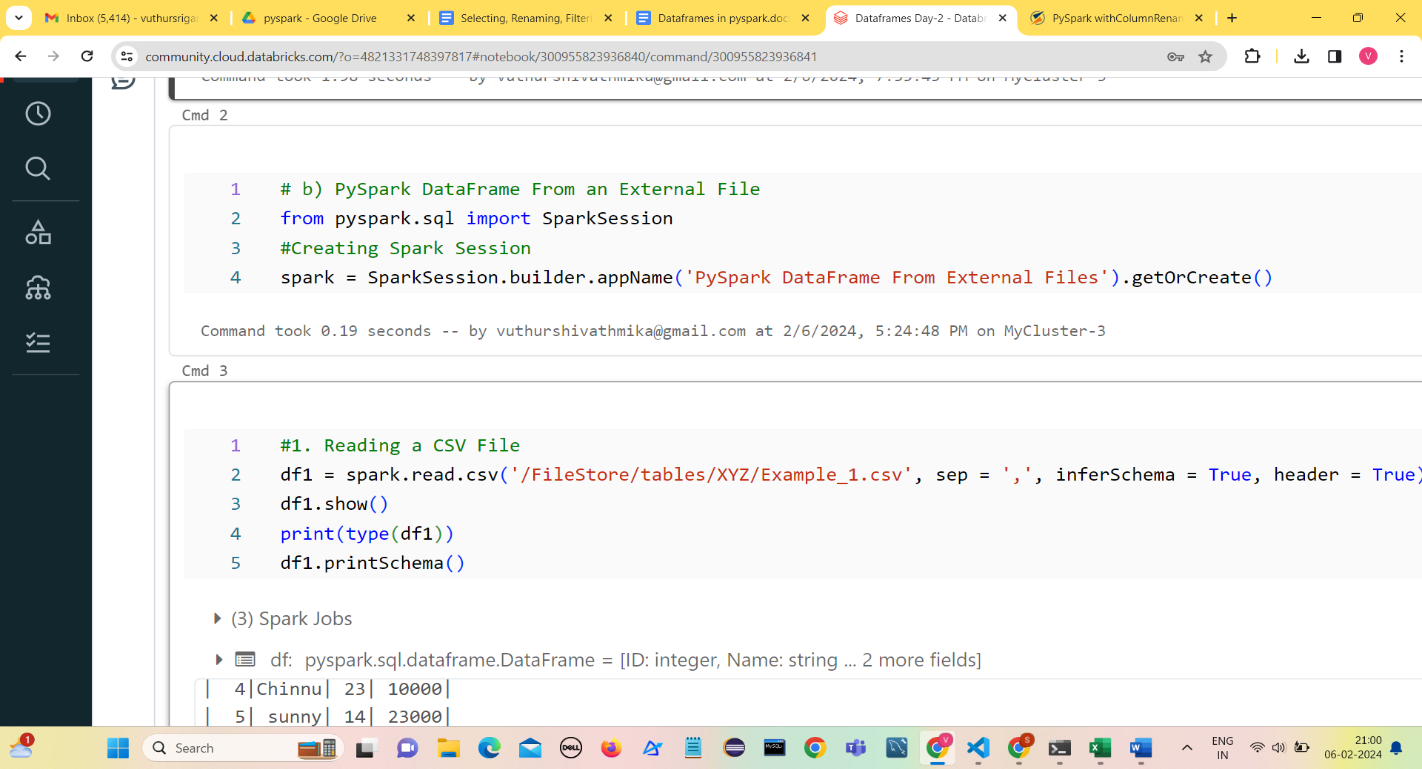
Dataframes in PySpark can be created primarily in two ways:

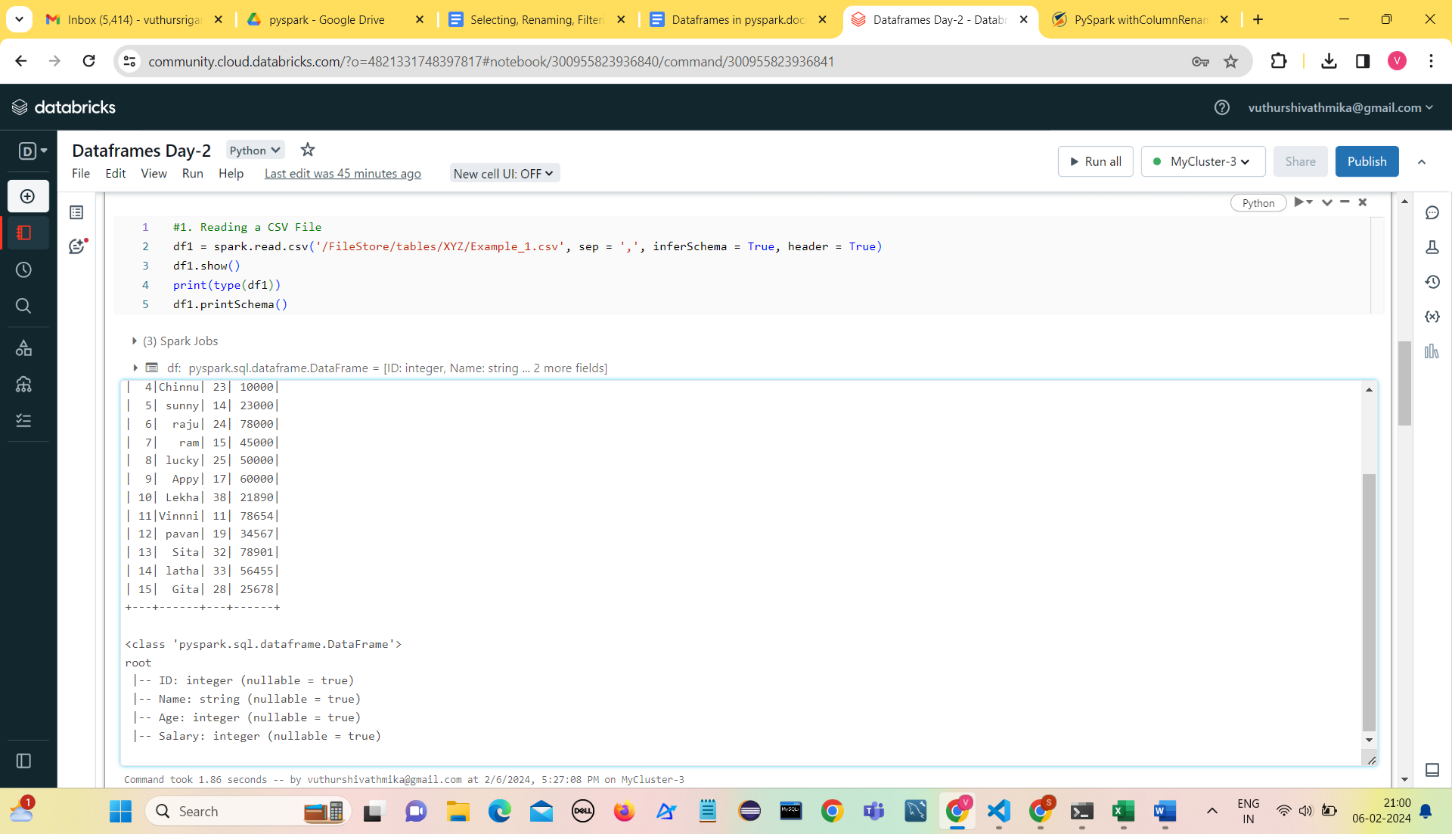
1. From an existing Resilient Distributed Dataset (RDD), which is a fundamental data structure in Spark
2. From external file sources, such as CSV, TXT, JSON
   1. **Create PySpark DataFrame From an Existing RDD**

To create a PySpark DataFrame from an existing RDD, we will first create an RDD using the **.parallelize()** method and then convert it into a PySpark DataFrame using the **.createDataFrame()**method of SparkSession.

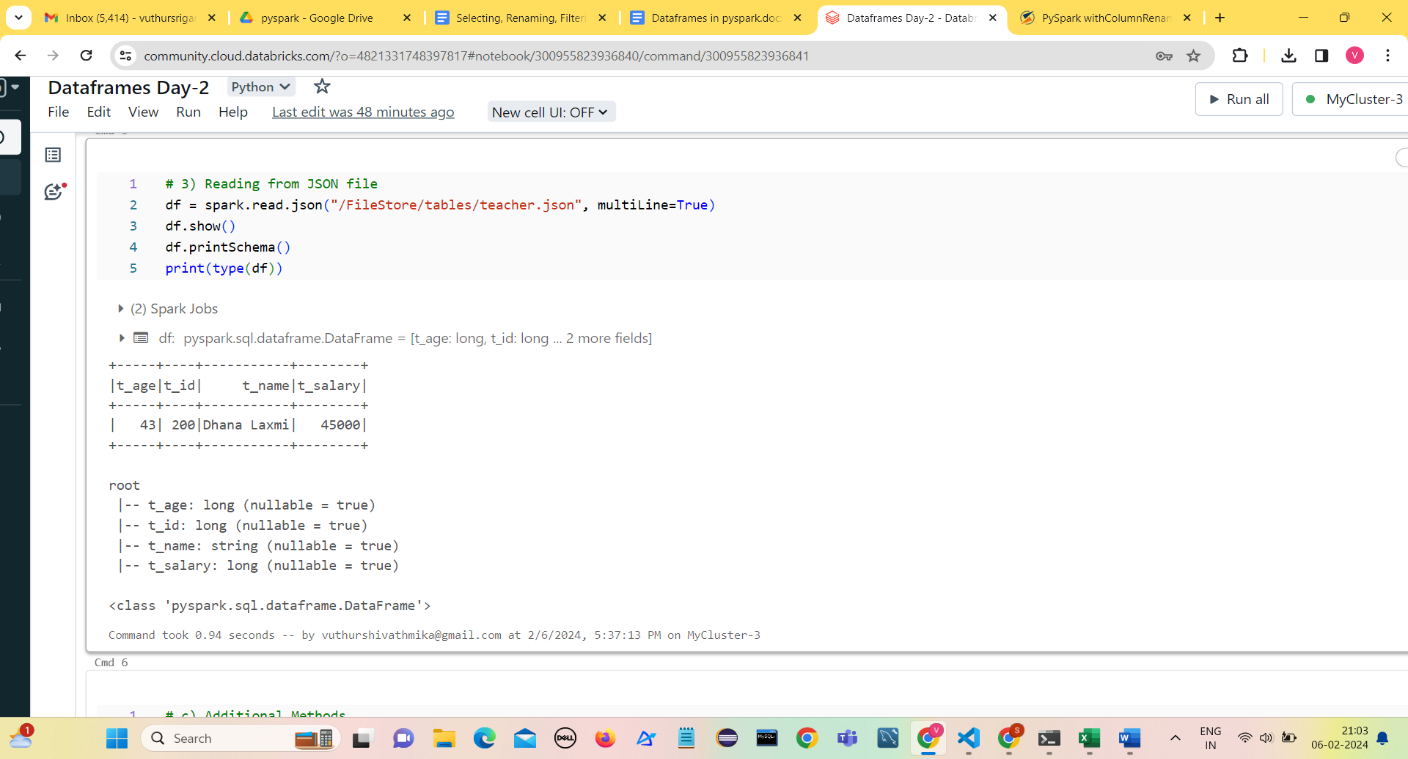
**2) Reading External Files into PySpark DataFrame**

**1) Reading a CSV File**





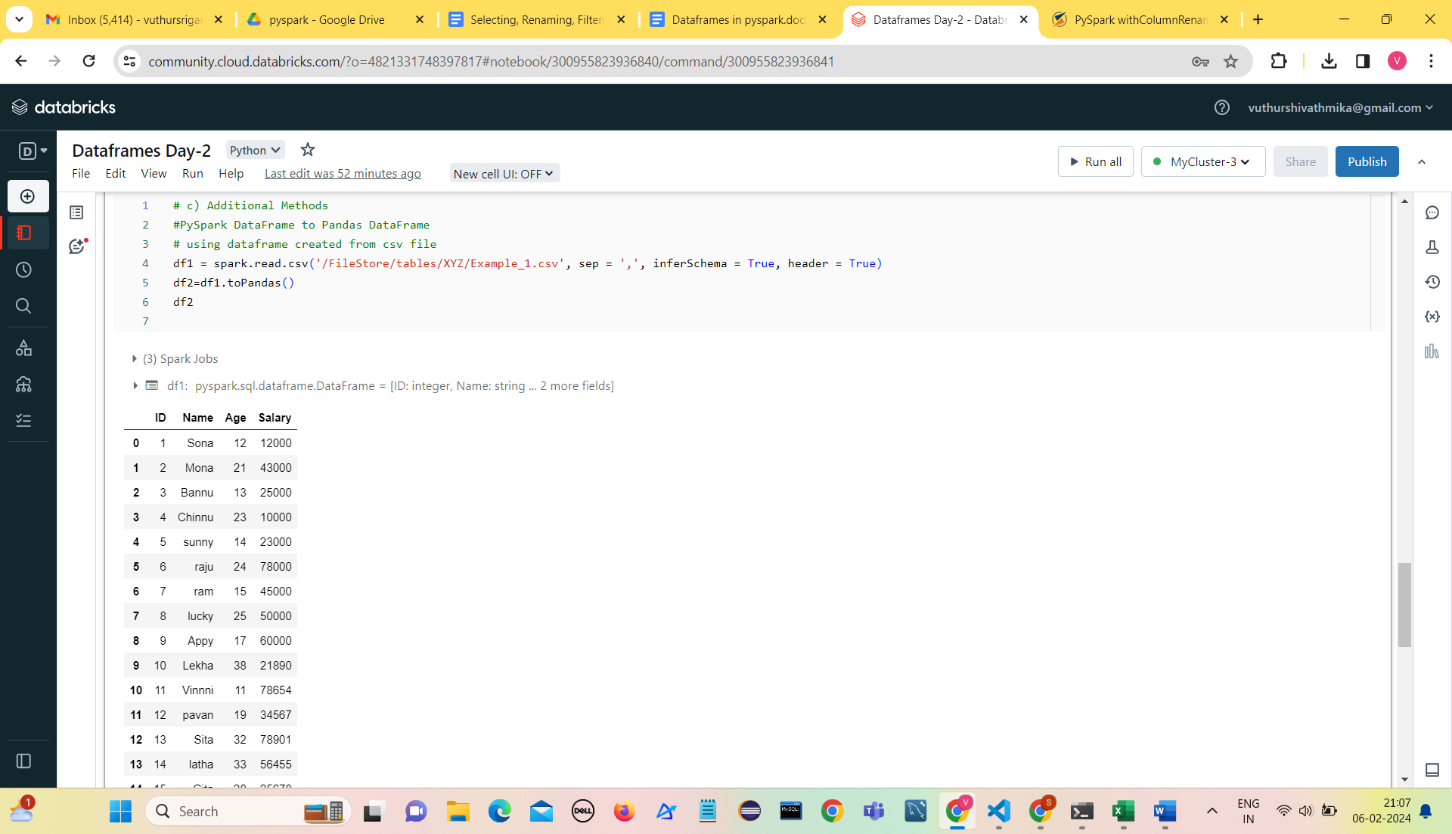
**2)  Reading a TXT File**

**3) Reading JSON file**

1. **Reading from more than one file at once**

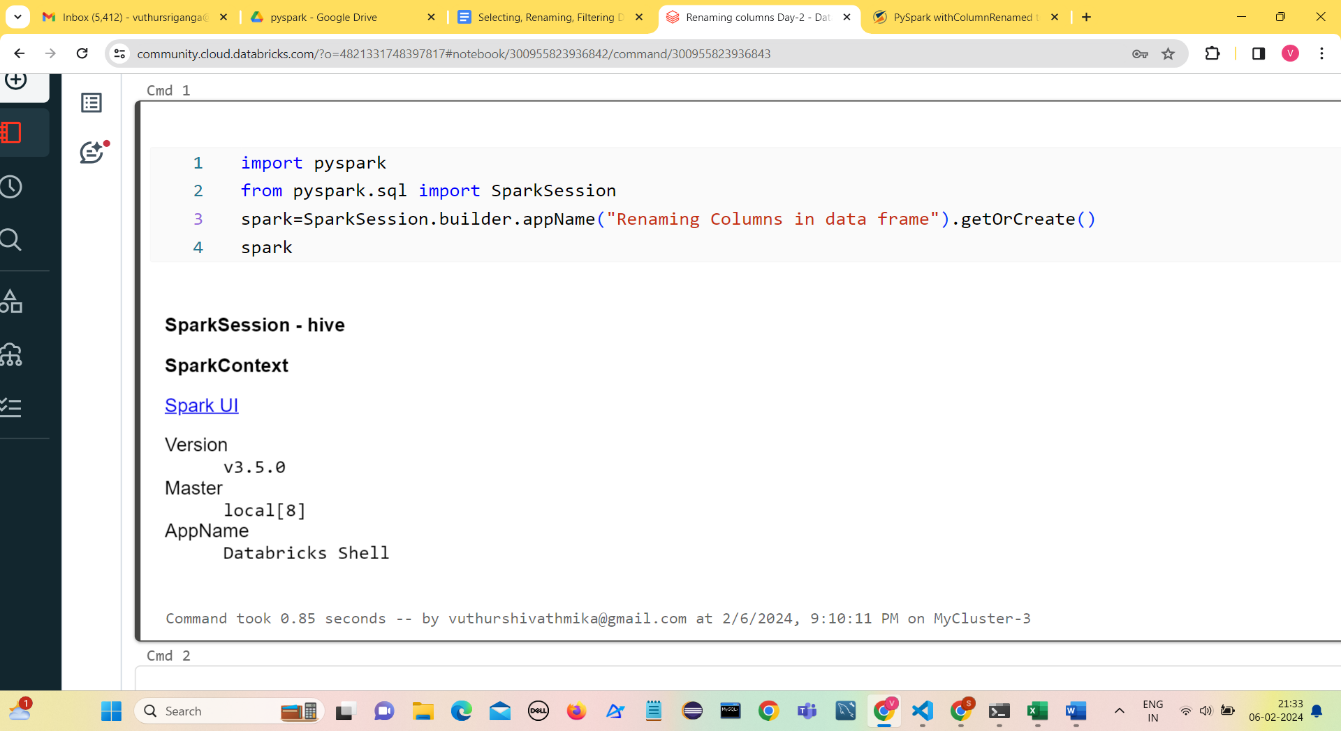
**Additional Useful Methods**

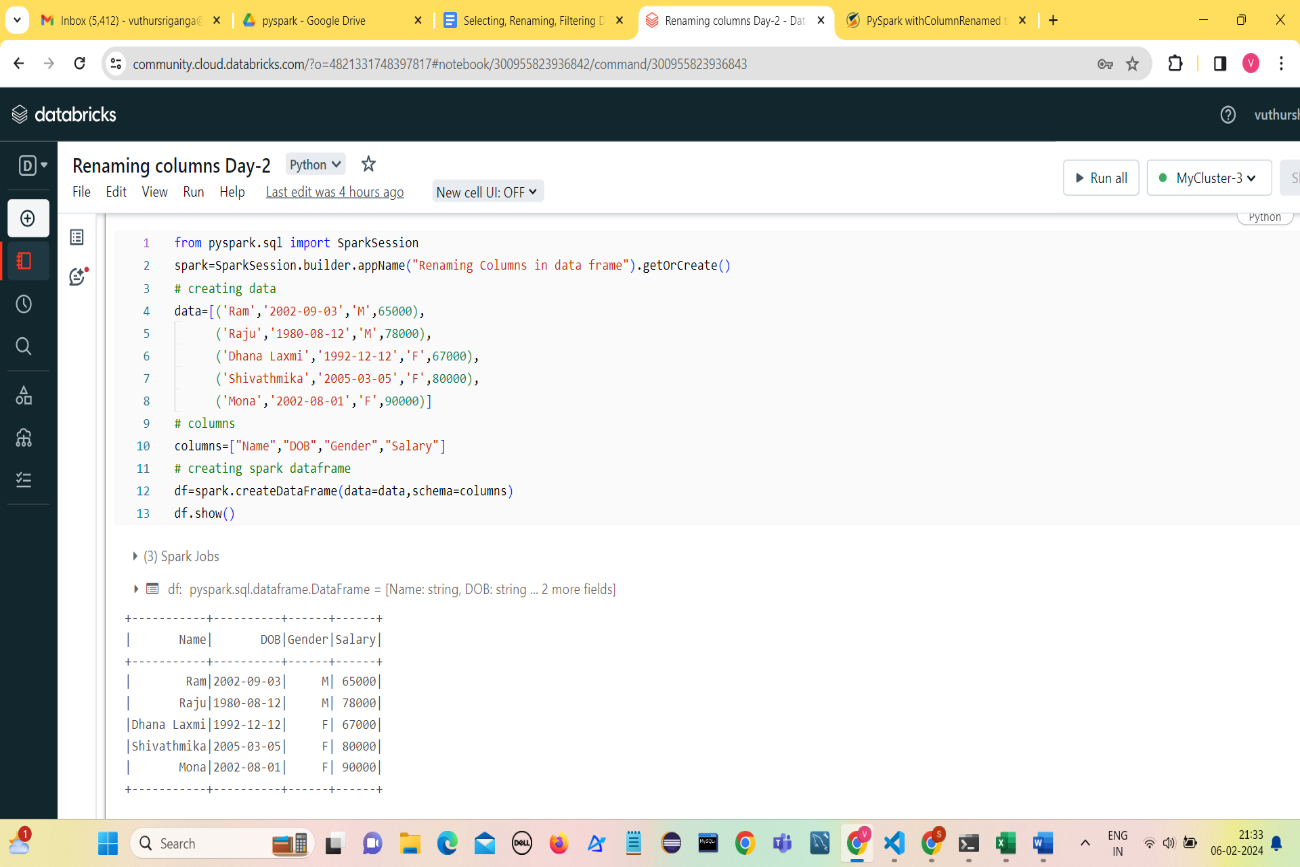
**PySpark DataFrame to Pandas DataFrame**

We can also convert the PySpark DataFrame into a **Pandas DataFrame**

* **Renaming Columns in data frames**

Create a PySpark Dataframe

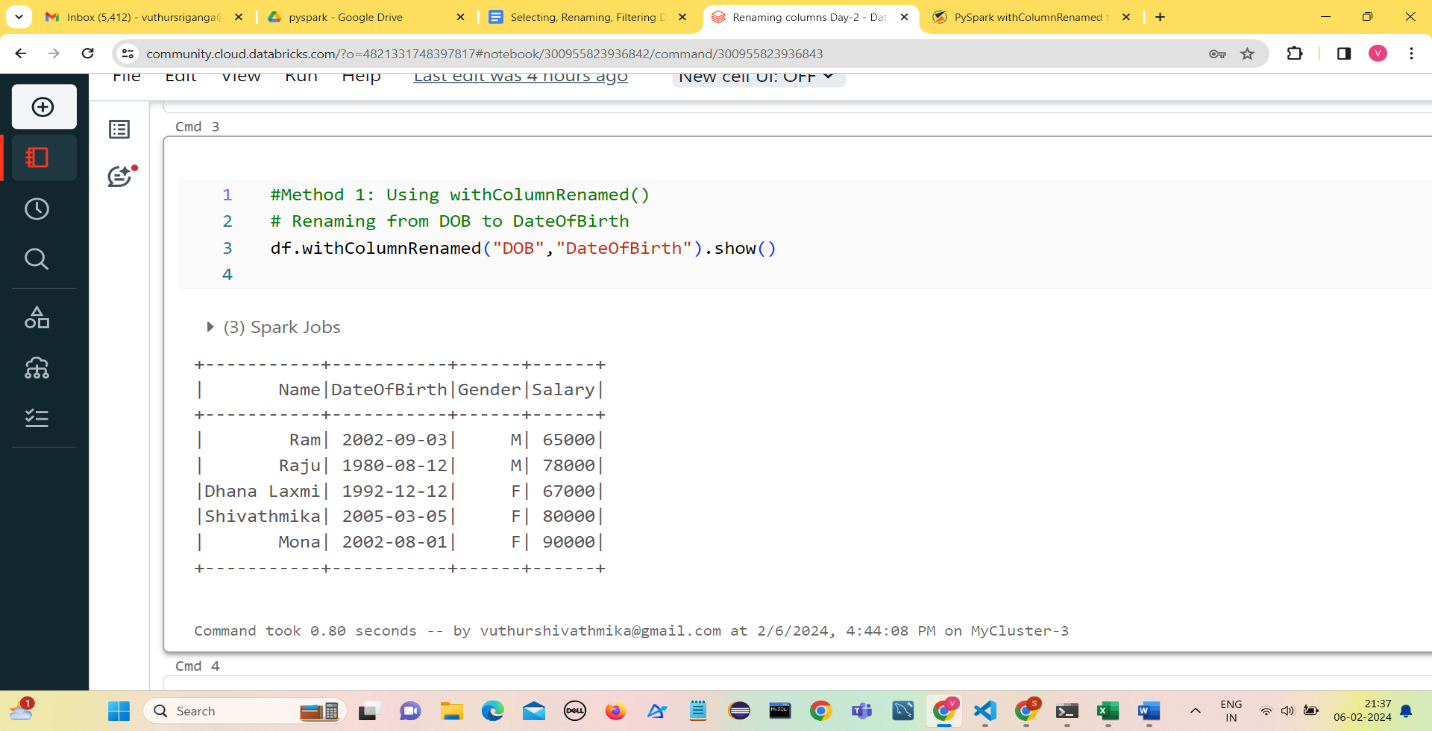


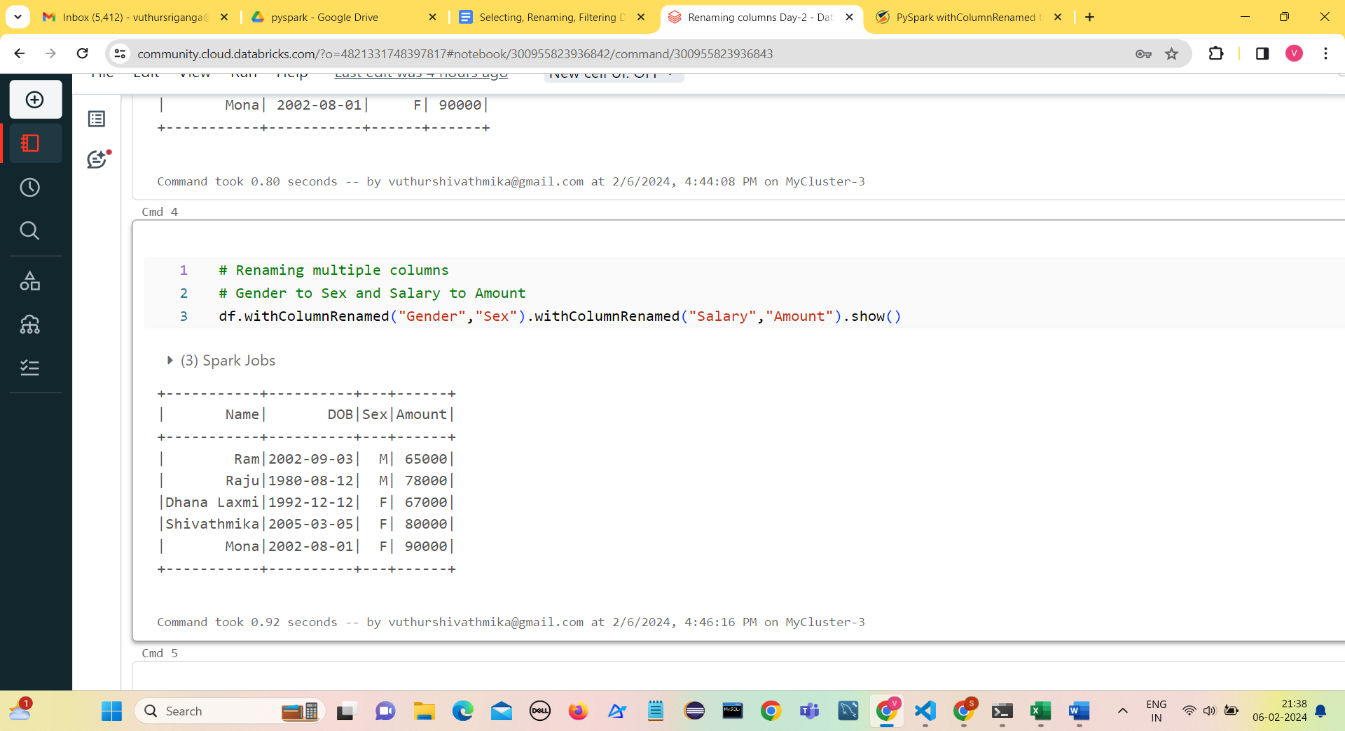


**Method 1: Using withColumnRenamed()**

**Syntax:**DataFrame.withColumnRenamed(existing, new)

**Parameters:**

* **existingstr:**Existing column name of data frame to rename.
* **newstr:**New column name.
* **Return type:**Returns a data frame by renaming an existing column.

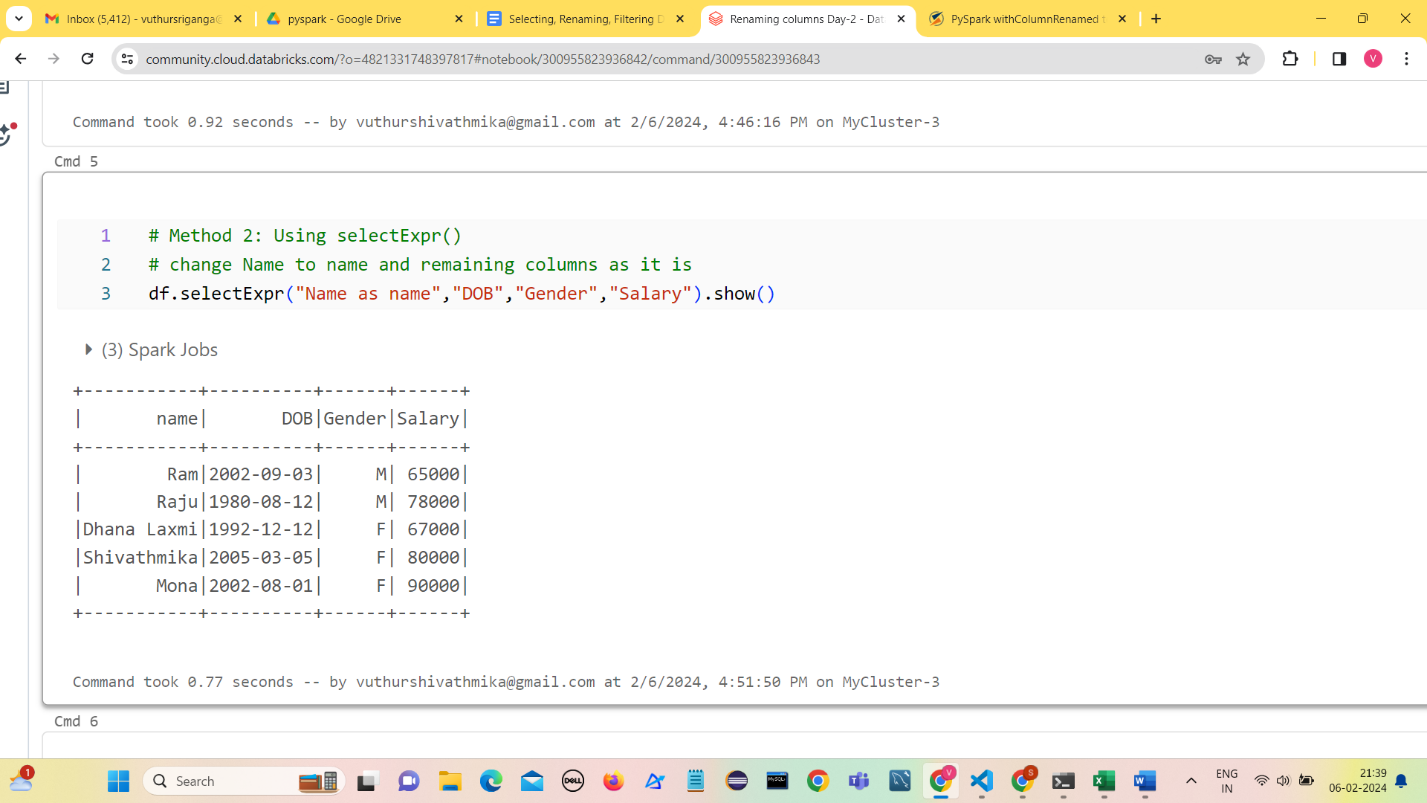
**Renaming multiple columns**

**Method 2: Using selectExpr()**

Renamingthe column names using**selectExpr()**method

**Syntax :**DataFrame.selectExpr(expr)

**Parameters :**

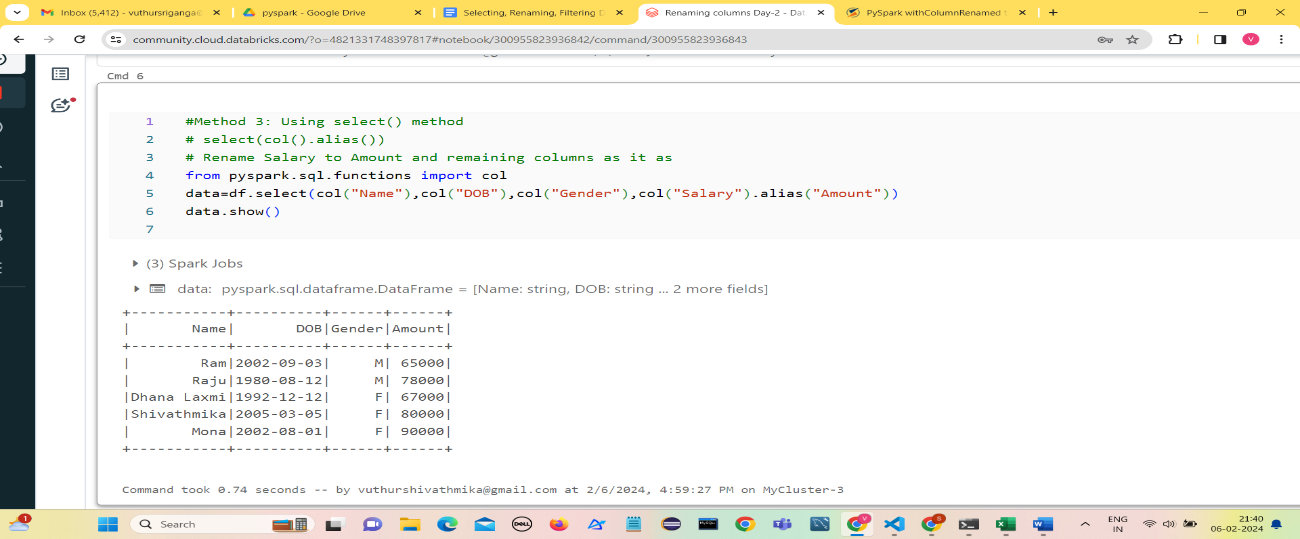
**expr :**It’s anSQL expression.

**Method 3: Using select() method**

**Syntax:**DataFrame.select(cols)

**Parameters :**

**cols:**List of column names as strings.

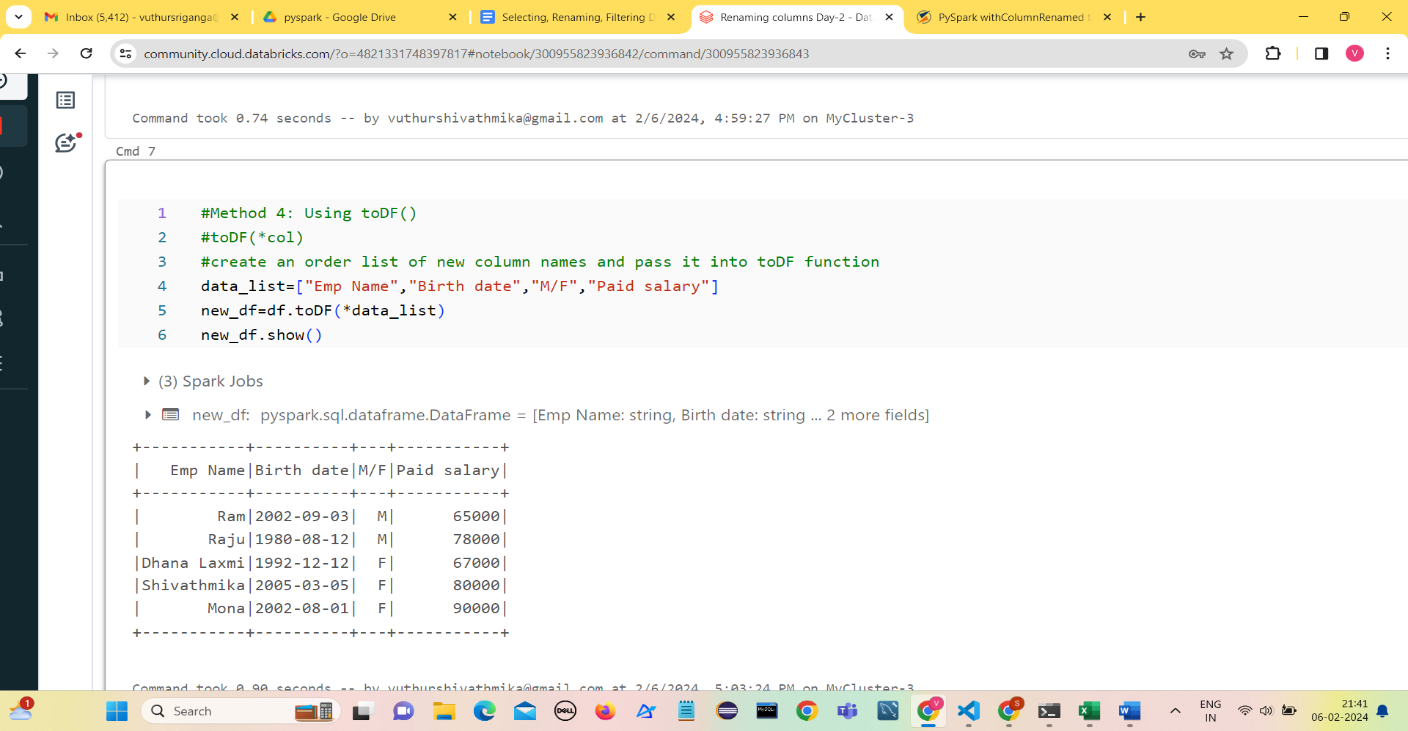
**Return type:**Selects the cols in the dataframe and returns a new DataFrame.

**Method 4: Using toDF()**

This function returns a new DataFrame that with new specified column names.

**Syntax:**toDF(\*col)

Where, col is a new column name

In this example, we will create an order list of new column names and pass it into toDF function

**To change some columns**