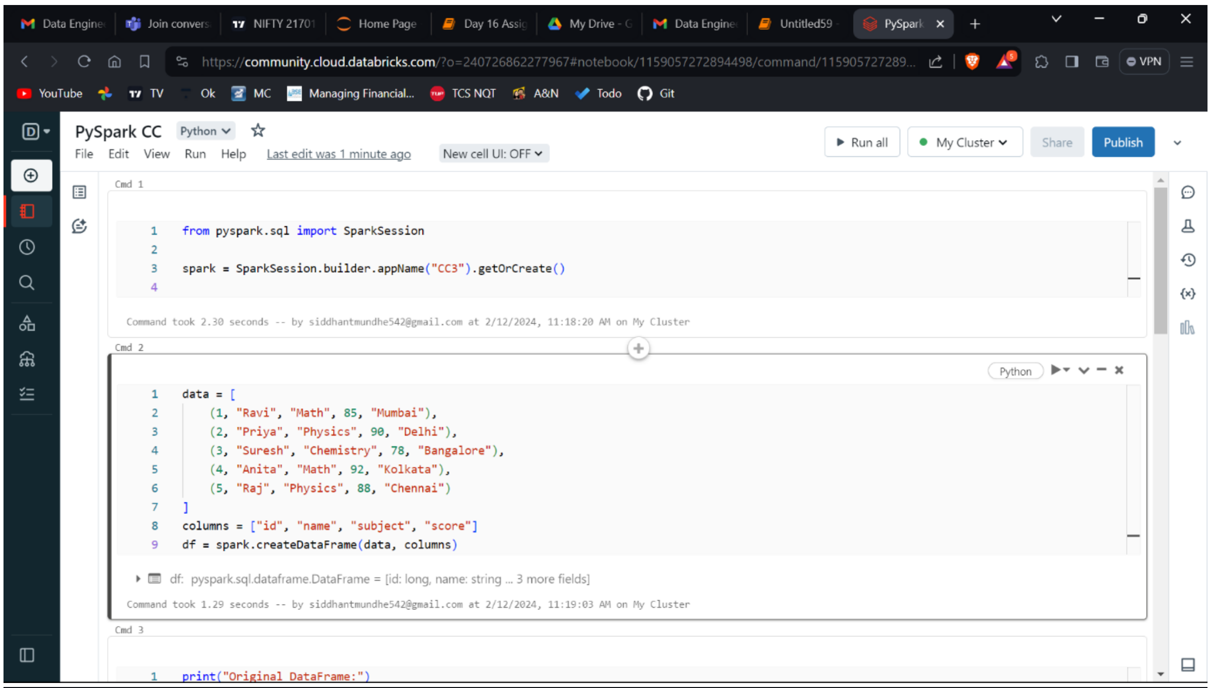
Siddhant Mundhe

Data engineering batch 1

Coding Challenge 3



Cmd 1

```python
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("CC3").getOrCreate()

```

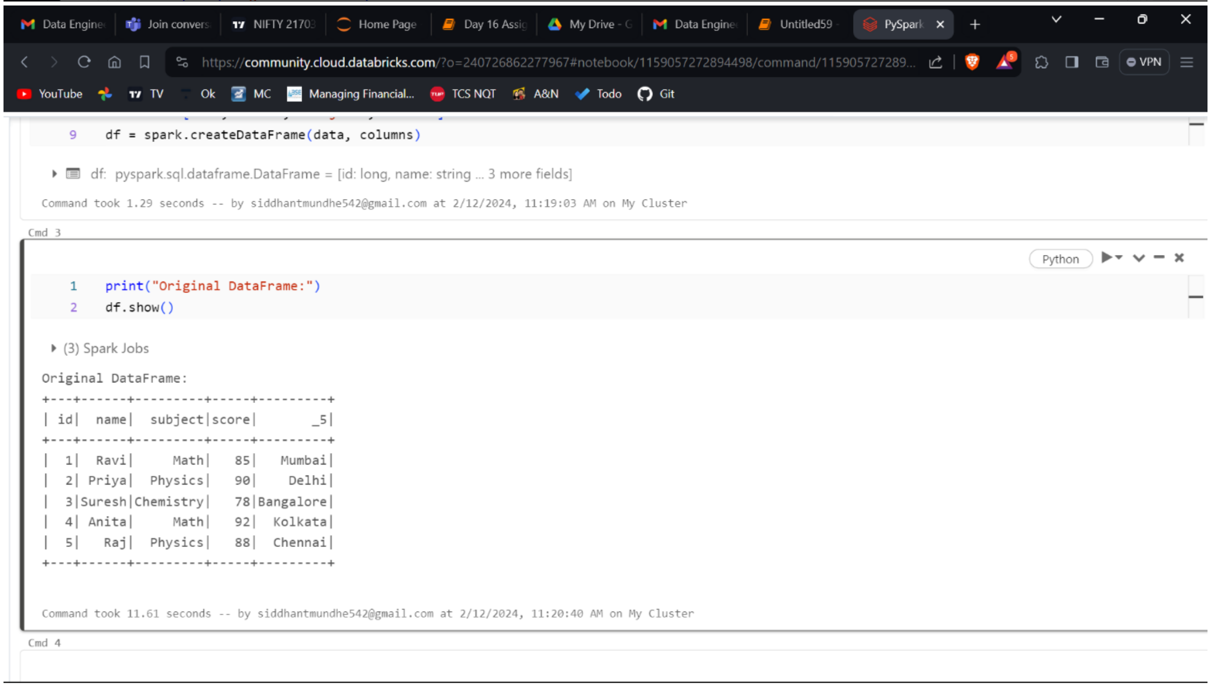
Command took 2.30 seconds -- by siddhantmundhe542@gmail.com at 2/12/2024, 11:18:20 AM on My Cluster

Cmd 2

```python
data = [
    (1, "Ravi", "Math", 85, "Mumbai"),
    (2, "Priya", "Physics", 90, "Delhi"),
    (3, "Suresh", "Chemistry", 78, "Bangalore"),
    (4, "Anita", "Math", 92, "Kolkata"),
    (5, "Raj", "Physics", 88, "Chennai")
]
columns = ["id", "name", "subject", "score"]
df = spark.createDataFrame(data, columns)
```

df: pyspark.sql.dataframe.DataFrame = [id: long, name: string ... 3 more fields]

Command took 1.29 seconds -- by siddhantmundhe542@gmail.com at 2/12/2024, 11:19:03 AM on My Cluster



Cmd 3

```python
print("Original DataFrame:")
df.show()
```

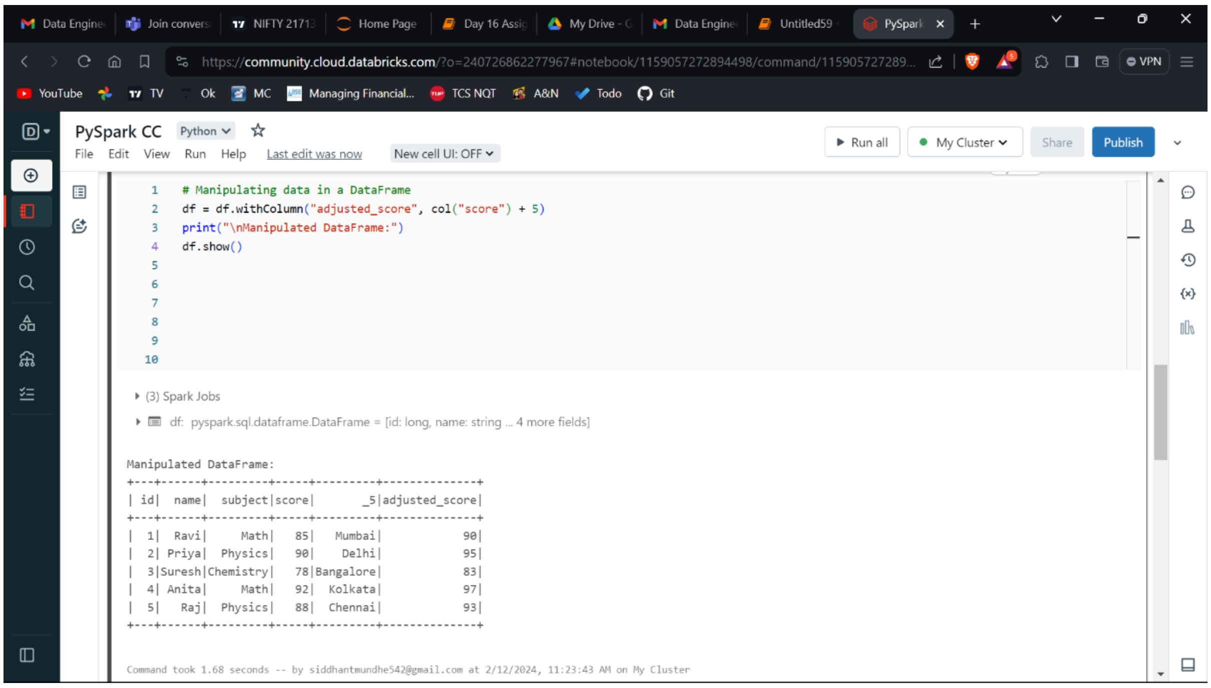
(3) Spark Jobs

```
Original DataFrame:
+---+------+---------+-----+---------+
| id|  name|  subject|score|       _5|
+---+------+---------+-----+---------+
|  1|  Ravi|     Math|   85|   Mumbai|
|  2| Priya|  Physics|   90|    Delhi|
|  3|Suresh|Chemistry|   78|Bangalore|
|  4| Anita|     Math|   92|  Kolkata|
|  5|   Raj|  Physics|   88|  Chennai|
+---+------+---------+-----+---------+
```

Command took 11.61 seconds -- by siddhantmundhe542@gmail.com at 2/12/2024, 11:20:40 AM on My Cluster

Cmd 4

Manipulating Data in a DataFrame

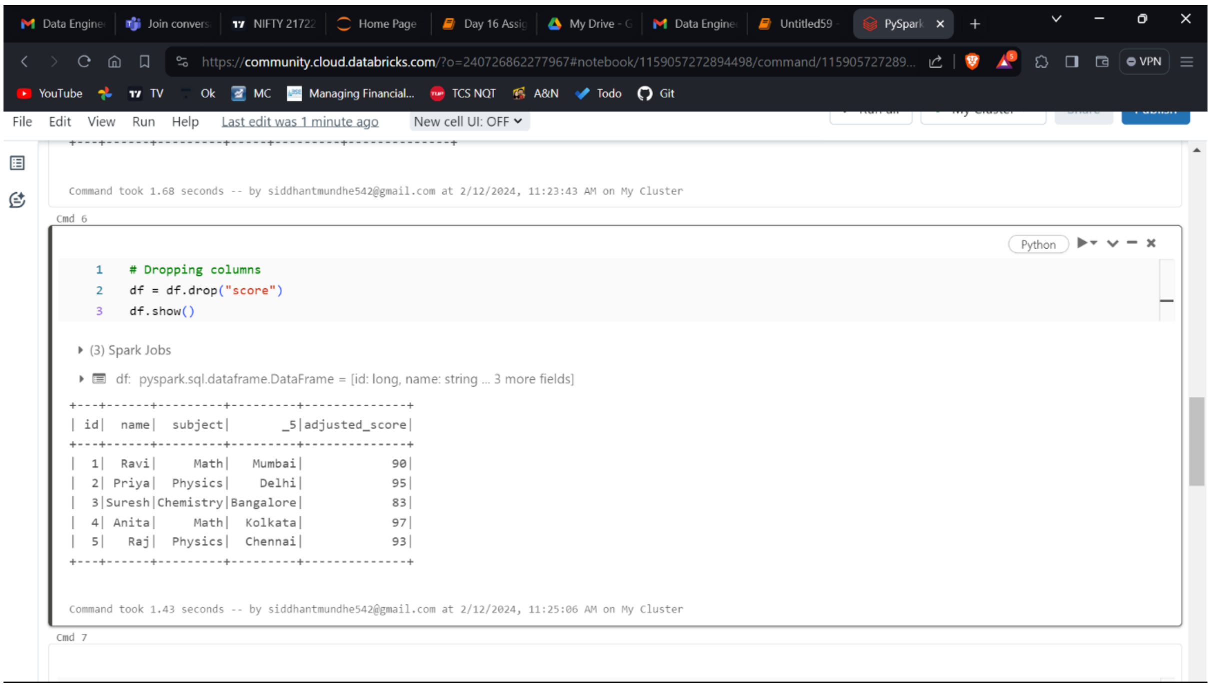


```python
# Manipulating data in a DataFrame
df = df.withColumn("adjusted_score", col("score") + 5)
print("\nManipulated DataFrame:")
df.show()
```

```
(3) Spark Jobs
df:  pyspark.sql.dataframe.DataFrame = [id: long, name: string ... 4 more fields]

Manipulated DataFrame:
+---+------+---------+-----+---------+--------------+
| id|  name|  subject|score|       _5|adjusted_score|
+---+------+---------+-----+---------+--------------+
|  1|  Ravi|     Math|   85|   Mumbai|            90|
|  2| Priya|  Physics|   90|    Delhi|            95|
|  3|Suresh|Chemistry|   78|Bangalore|            83|
|  4| Anita|     Math|   92|  Kolkata|            97|
|  5|   Raj|  Physics|   88|  Chennai|            93|
+---+------+---------+-----+---------+--------------+

Command took 1.68 seconds -- by siddhantmundhe542@gmail.com at 2/12/2024, 11:23:43 AM on My Cluster
```

Here, a new column called adjusted_score is created by adding 5 to the existing score column using the withColumn method.

Dropping Columns

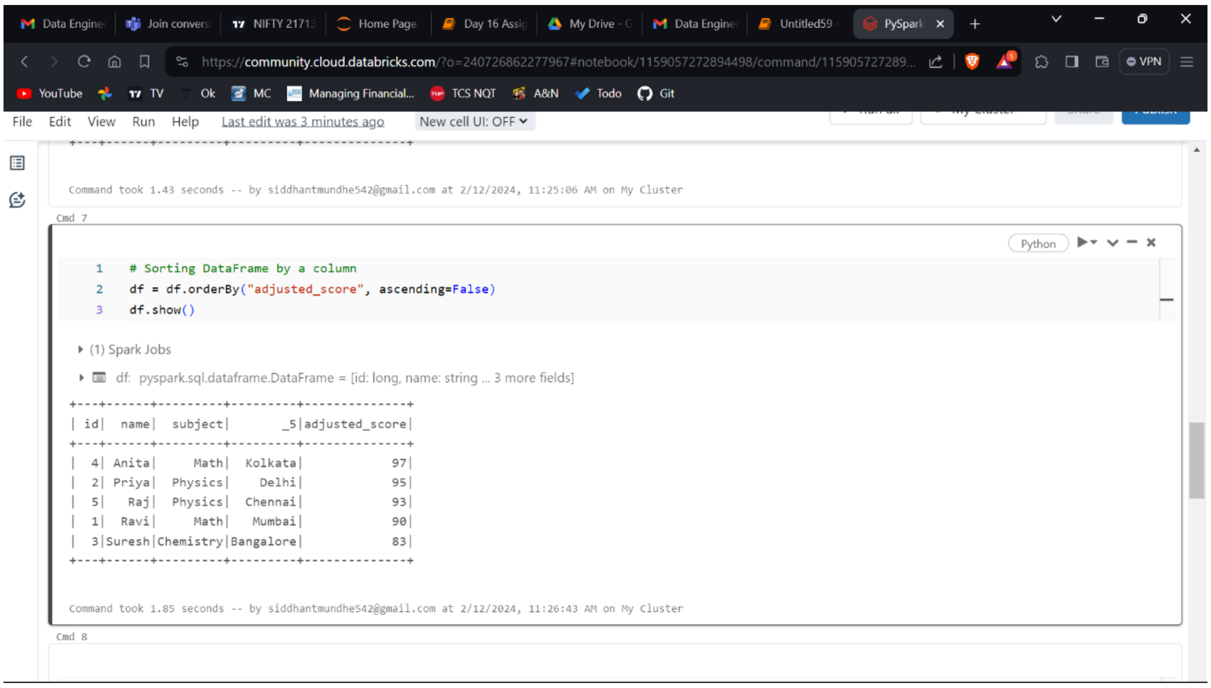


```python
# Dropping columns
df = df.drop("score")
df.show()
```

```
(3) Spark Jobs
df:  pyspark.sql.dataframe.DataFrame = [id: long, name: string ... 3 more fields]
+---+------+---------+---------+--------------+
| id|  name|  subject|       _5|adjusted_score|
+---+------+---------+---------+--------------+
|  1|  Ravi|     Math|   Mumbai|            90|
|  2| Priya|  Physics|    Delhi|            95|
|  3|Suresh|Chemistry|Bangalore|            83|
|  4| Anita|     Math|  Kolkata|            97|
|  5|   Raj|  Physics|  Chennai|            93|
+---+------+---------+---------+--------------+

Command took 1.43 seconds -- by siddhantmundhe542@gmail.com at 2/12/2024, 11:25:06 AM on My Cluster
```

The drop method is used to remove the "score" column from the DataFrame.

Sorting DataFrame by a Column



```
Command took 1.43 seconds -- by siddhantmundhe542@gmail.com at 2/12/2024, 11:25:06 AM on My Cluster
```

Cmd 7

```python
# Sorting DataFrame by a column
df = df.orderBy("adjusted_score", ascending=False)
df.show()
```

(1) Spark Jobs

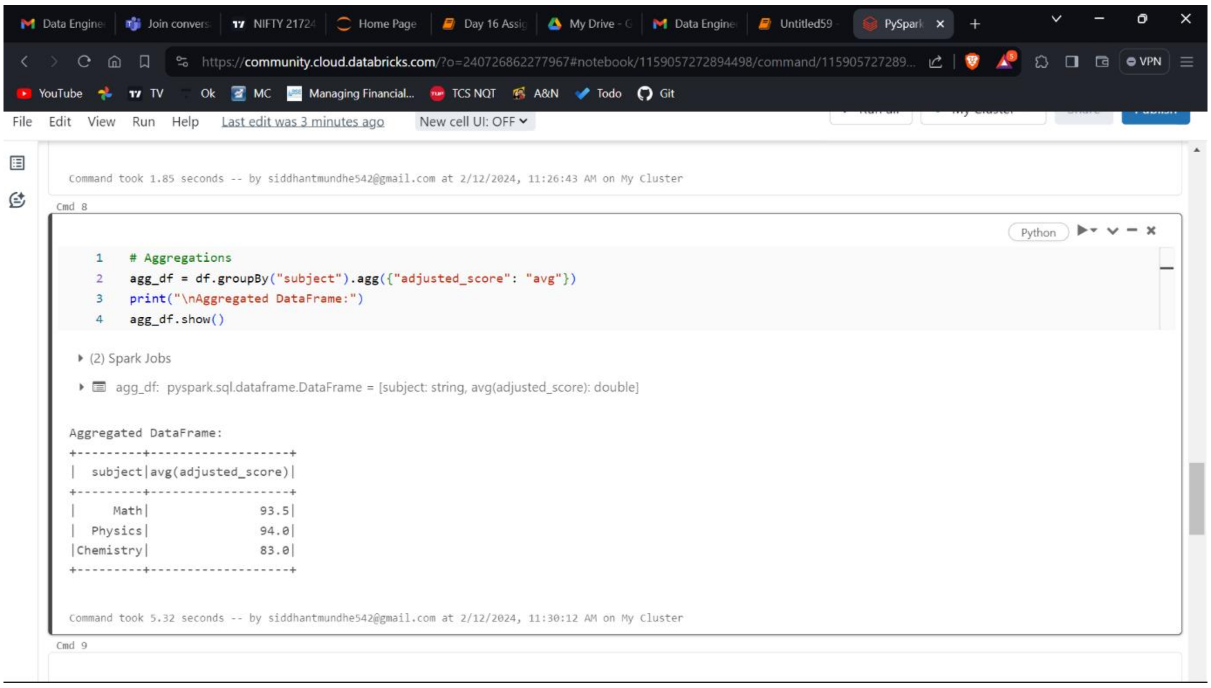
df: pyspark.sql.dataframe.DataFrame = [id: long, name: string ... 3 more fields]

```
+---+------+---------+---------+--------------+
| id|  name|  subject|       _5|adjusted_score|
+---+------+---------+---------+--------------+
|  4| Anita|     Math|  Kolkata|            97|
|  2| Priya|  Physics|    Delhi|            95|
|  5|   Raj|  Physics|  Chennai|            93|
|  1|  Ravi|     Math|   Mumbai|            90|
|  3|Suresh|Chemistry|Bangalore|            83|
+---+------+---------+---------+--------------+

Command took 1.85 seconds -- by siddhantmundhe542@gmail.com at 2/12/2024, 11:26:43 AM on My Cluster
```

Cmd 8

The DataFrame is sorted in descending order based on the adjusted_score column using the orderBy method.

Aggregations



```
Command took 1.85 seconds -- by siddhantmundhe542@gmail.com at 2/12/2024, 11:26:43 AM on My Cluster
```

Cmd 8

```python
# Aggregations
agg_df = df.groupBy("subject").agg({"adjusted_score": "avg"})
print("\nAggregated DataFrame:")
agg_df.show()
```

(2) Spark Jobs

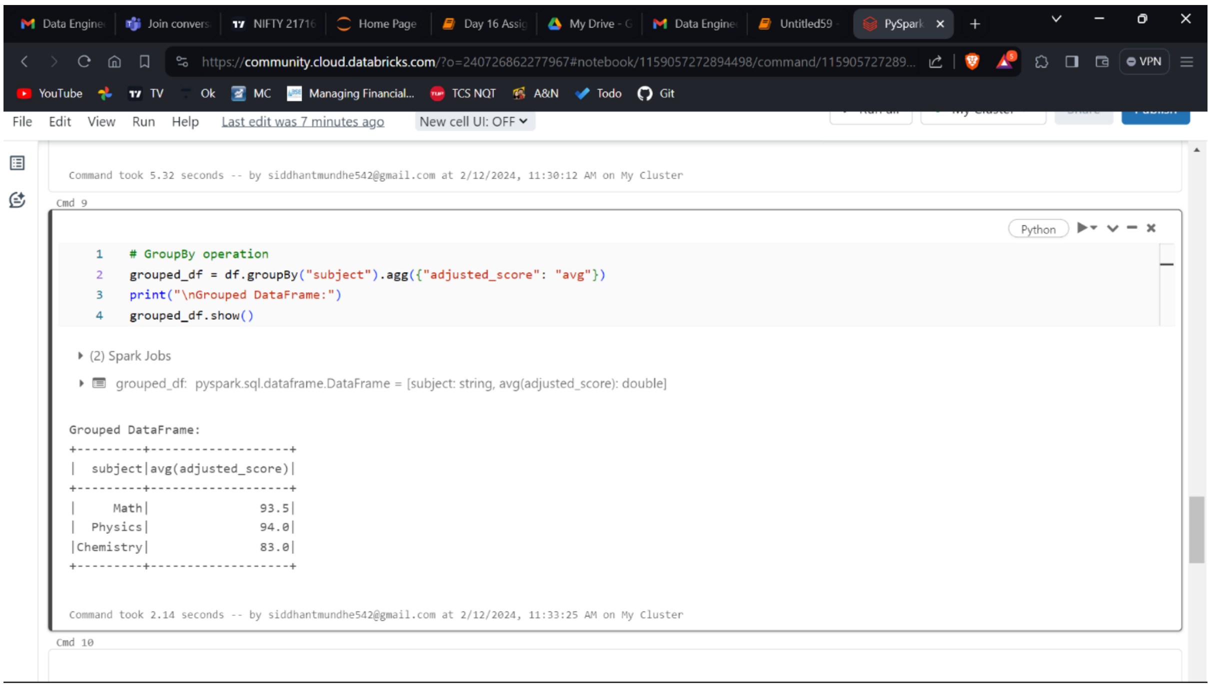
agg_df: pyspark.sql.dataframe.DataFrame = [subject: string, avg(adjusted_score): double]

```
Aggregated DataFrame:
+---------+-------------------+
|  subject|avg(adjusted_score)|
+---------+-------------------+
|     Math|               93.5|
|  Physics|               94.0|
|Chemistry|               83.0|
+---------+-------------------+

Command took 5.32 seconds -- by siddhantmundhe542@gmail.com at 2/12/2024, 11:30:12 AM on My Cluster
```

Cmd 9

This code groups the DataFrame by the "subject" column and calculates the average of the adjusted_score for each group using the agg method.

GroupBy Operation


```
Command took 5.32 seconds -- by siddhantmundhe542@gmail.com at 2/12/2024, 11:30:12 AM on My Cluster
```

```python
# GroupBy operation
grouped_df = df.groupBy("subject").agg({"adjusted_score": "avg"})
print("\nGrouped DataFrame:")
grouped_df.show()
```

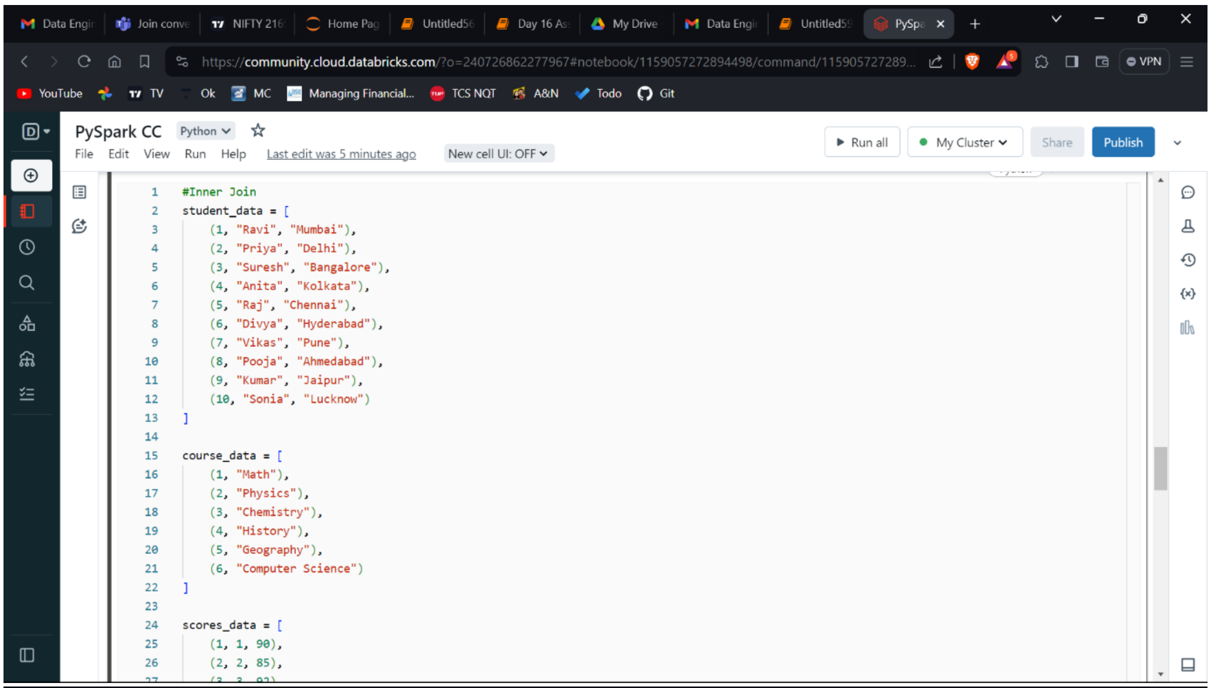
```
(2) Spark Jobs
grouped_df: pyspark.sql.dataframe.DataFrame = [subject: string, avg(adjusted_score): double]

Grouped DataFrame:
+---------+-------------------+
|  subject|avg(adjusted_score)|
+---------+-------------------+
|     Math|               93.5|
|  Physics|               94.0|
|Chemistry|               83.0|
+---------+-------------------+

Command took 2.14 seconds -- by siddhantmundhe542@gmail.com at 2/12/2024, 11:33:25 AM on My Cluster
```

This groups the DataFrame by the "subject" column and calculates the average of the adjusted_score for each group using the agg method.me

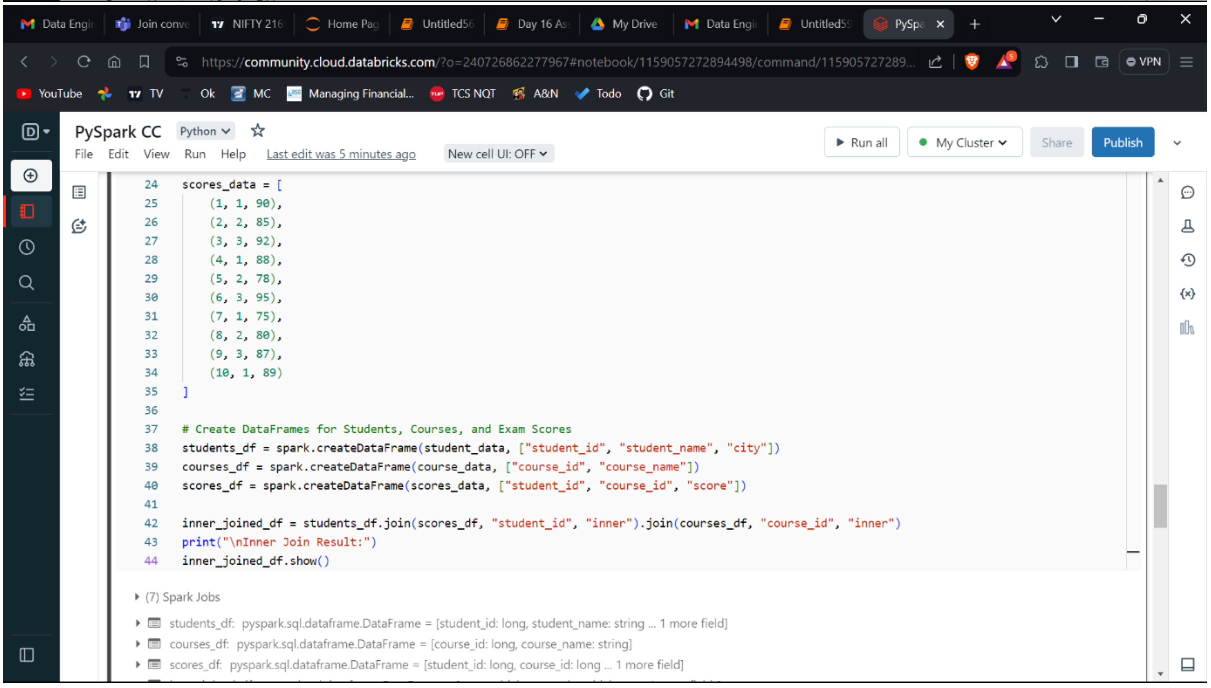
<u>**JOINS**</u>

<u>1. Inner Join:</u>

An inner join returns only the rows where there is a match in both DataFrames on the specified join condition.

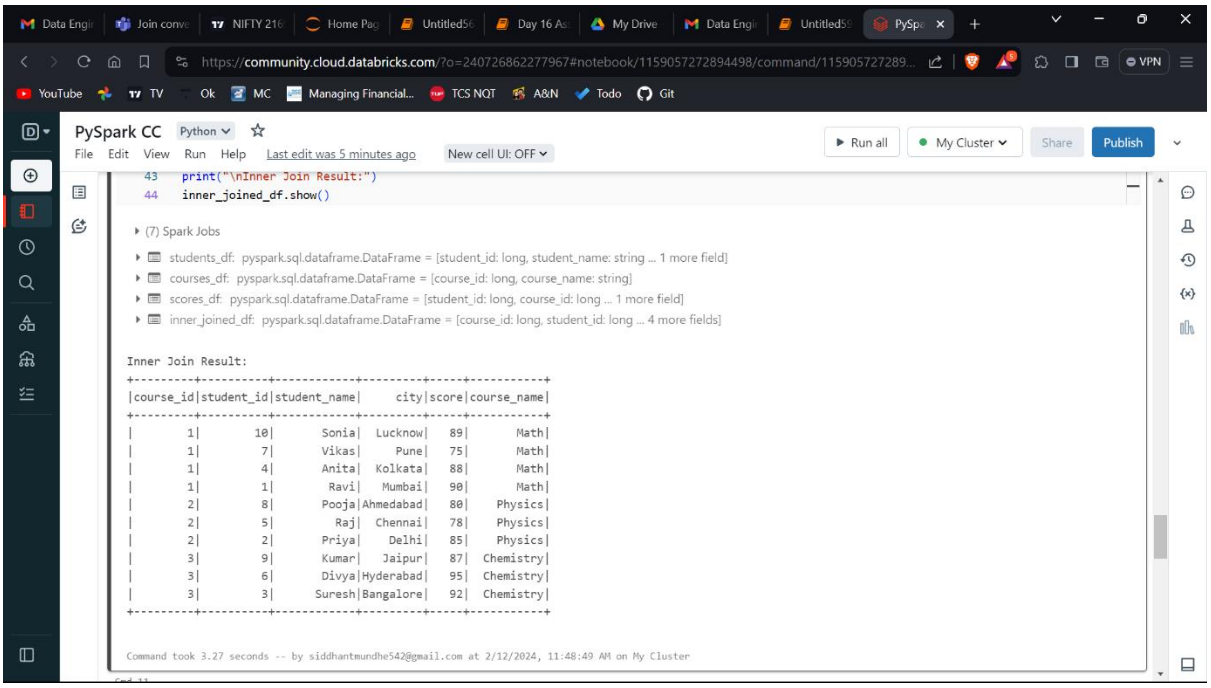


```
#Inner Join
student_data = [
    (1, "Ravi", "Mumbai"),
    (2, "Priya", "Delhi"),
    (3, "Suresh", "Bangalore"),
    (4, "Anita", "Kolkata"),
    (5, "Raj", "Chennai"),
    (6, "Divya", "Hyderabad"),
    (7, "Vikas", "Pune"),
    (8, "Pooja", "Ahmedabad"),
    (9, "Kumar", "Jaipur"),
    (10, "Sonia", "Lucknow")
]

course_data = [
    (1, "Math"),
    (2, "Physics"),
    (3, "Chemistry"),
    (4, "History"),
    (5, "Geography"),
    (6, "Computer Science")
]

scores_data = [
    (1, 1, 90),
    (2, 2, 85),
```



```
scores_data = [
    (1, 1, 90),
    (2, 2, 85),
    (3, 3, 92),
    (4, 1, 88),
    (5, 2, 78),
    (6, 3, 95),
    (7, 1, 75),
    (8, 2, 80),
    (9, 3, 87),
    (10, 1, 89)
]

# Create DataFrames for Students, Courses, and Exam Scores
students_df = spark.createDataFrame(student_data, ["student_id", "student_name", "city"])
courses_df = spark.createDataFrame(course_data, ["course_id", "course_name"])
scores_df = spark.createDataFrame(scores_data, ["student_id", "course_id", "score"])

inner_joined_df = students_df.join(scores_df, "student_id", "inner").join(courses_df, "course_id", "inner")
print("\nInner Join Result:")
inner_joined_df.show()
```

```
(7) Spark Jobs
students_df:  pyspark.sql.dataframe.DataFrame = [student_id: long, student_name: string ... 1 more field]
courses_df:  pyspark.sql.dataframe.DataFrame = [course_id: long, course_name: string]
scores_df:  pyspark.sql.dataframe.DataFrame = [student_id: long, course_id: long ... 1 more field]
```

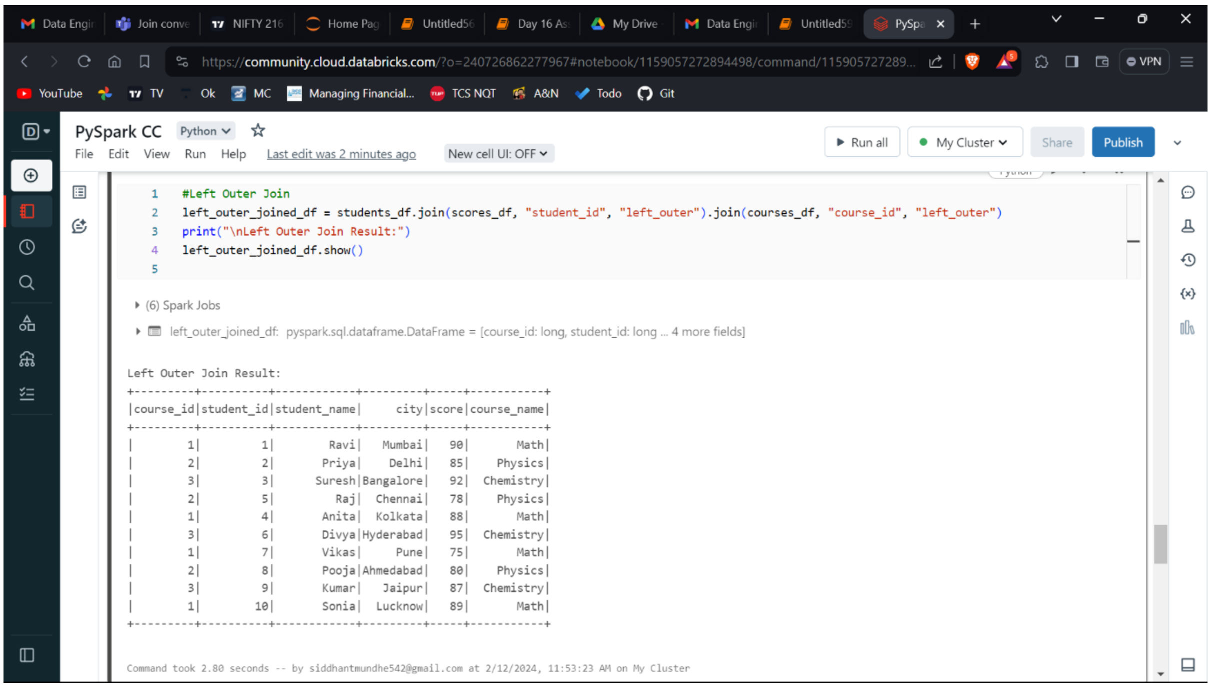


```python
print("\nInner Join Result:")
inner_joined_df.show()
```

```
(7) Spark Jobs
students_df:  pyspark.sql.dataframe.DataFrame = [student_id: long, student_name: string ... 1 more field]
courses_df:  pyspark.sql.dataframe.DataFrame = [course_id: long, course_name: string]
scores_df:  pyspark.sql.dataframe.DataFrame = [student_id: long, course_id: long ... 1 more field]
inner_joined_df:  pyspark.sql.dataframe.DataFrame = [course_id: long, student_id: long ... 4 more fields]

Inner Join Result:
+---------+----------+------------+---------+-----+-----------+
|course_id|student_id|student_name|     city|score|course_name|
+---------+----------+------------+---------+-----+-----------+
|        1|        10|       Sonia|  Lucknow|   89|       Math|
|        1|         7|       Vikas|     Pune|   75|       Math|
|        1|         4|       Anita|  Kolkata|   88|       Math|
|        1|         1|        Ravi|   Mumbai|   90|       Math|
|        2|         8|       Pooja|Ahmedabad|   80|    Physics|
|        2|         5|         Raj|  Chennai|   78|    Physics|
|        2|         2|       Priya|    Delhi|   85|    Physics|
|        3|         9|       Kumar|   Jaipur|   87|  Chemistry|
|        3|         6|       Divya|Hyderabad|   95|  Chemistry|
|        3|         3|      Suresh|Bangalore|   92|  Chemistry|
+---------+----------+------------+---------+-----+-----------+

Command took 3.27 seconds -- by siddhantmundhe542@gmail.com at 2/12/2024, 11:48:49 AM on My Cluster
```

2. Left Outer Join:

A left outer join returns all the rows from the left DataFrame and the matching rows from the right DataFrame. If there is no match, null values are filled in for the columns from the right DataFrame.

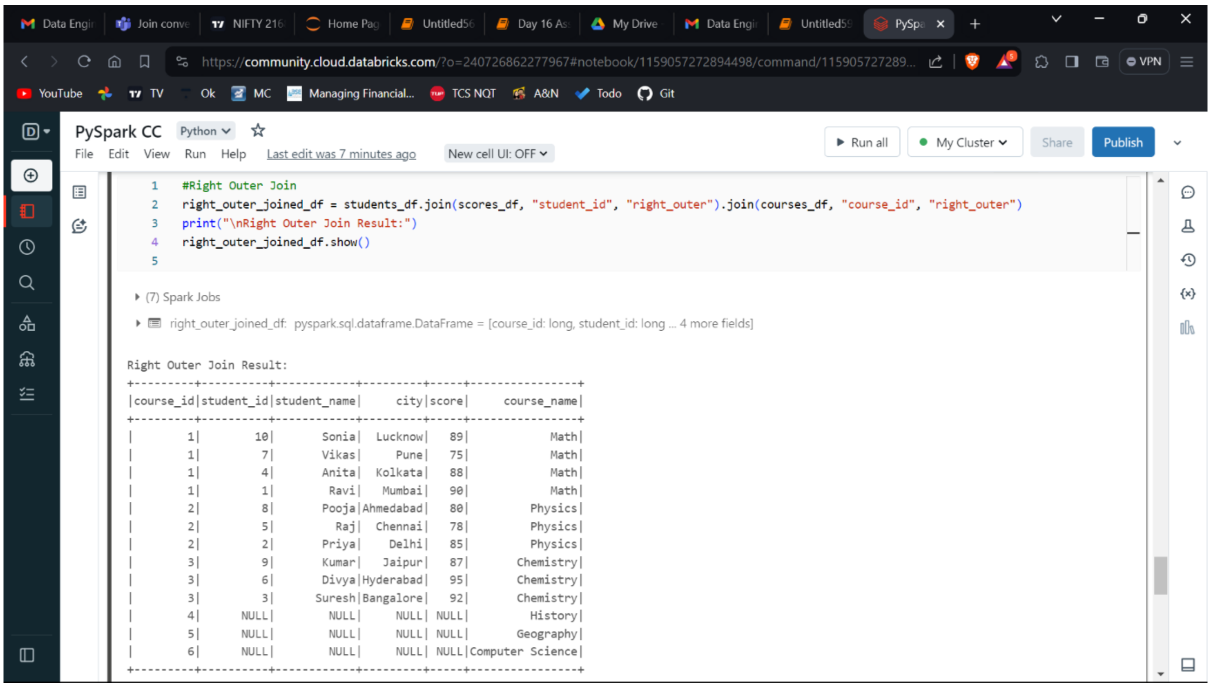


```python
#Left Outer Join
left_outer_joined_df = students_df.join(scores_df, "student_id", "left_outer").join(courses_df, "course_id", "left_outer")
print("\nLeft Outer Join Result:")
left_outer_joined_df.show()
```

```
(6) Spark Jobs
left_outer_joined_df:  pyspark.sql.dataframe.DataFrame = [course_id: long, student_id: long ... 4 more fields]

Left Outer Join Result:
+---------+----------+------------+---------+-----+-----------+
|course_id|student_id|student_name|     city|score|course_name|
+---------+----------+------------+---------+-----+-----------+
|        1|         1|        Ravi|   Mumbai|   90|       Math|
|        2|         2|       Priya|    Delhi|   85|    Physics|
|        3|         3|      Suresh|Bangalore|   92|  Chemistry|
|        2|         5|         Raj|  Chennai|   78|    Physics|
|        1|         4|       Anita|  Kolkata|   88|       Math|
|        3|         6|       Divya|Hyderabad|   95|  Chemistry|
|        1|         7|       Vikas|     Pune|   75|       Math|
|        2|         8|       Pooja|Ahmedabad|   80|    Physics|
|        3|         9|       Kumar|   Jaipur|   87|  Chemistry|
|        1|        10|       Sonia|  Lucknow|   89|       Math|
+---------+----------+------------+---------+-----+-----------+

Command took 2.80 seconds -- by siddhantmundhe542@gmail.com at 2/12/2024, 11:53:23 AM on My Cluster
```

3. Right Outer Join:

A right outer join returns all the rows from the right DataFrame and the matching rows from the left DataFrame. If there is no match, null values are filled in for the columns from the left DataFrame.



```python
#Right Outer Join
right_outer_joined_df = students_df.join(scores_df, "student_id", "right_outer").join(courses_df, "course_id", "right_outer")
print("\nRight Outer Join Result:")
right_outer_joined_df.show()
```

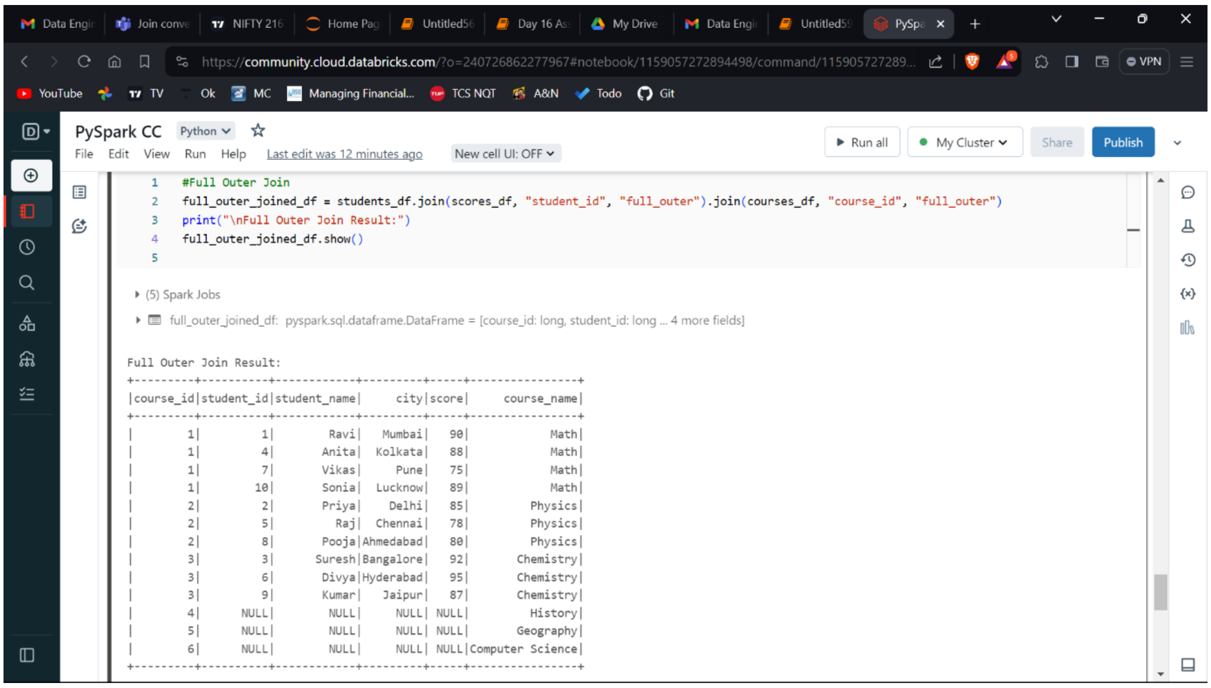
(7) Spark Jobs

right_outer_joined_df: pyspark.sql.dataframe.DataFrame = [course_id: long, student_id: long ... 4 more fields]

```
Right Outer Join Result:
+---------+----------+------------+---------+-----+----------------+
|course_id|student_id|student_name|     city|score|     course_name|
+---------+----------+------------+---------+-----+----------------+
|        1|        10|       Sonia|  Lucknow|   89|            Math|
|        1|         7|       Vikas|     Pune|   75|            Math|
|        1|         4|       Anita|  Kolkata|   88|            Math|
|        1|         1|        Ravi|   Mumbai|   90|            Math|
|        2|         8|       Pooja|Ahmedabad|   80|         Physics|
|        2|         5|         Raj|  Chennai|   78|         Physics|
|        2|         2|       Priya|    Delhi|   85|         Physics|
|        3|         9|       Kumar|   Jaipur|   87|       Chemistry|
|        3|         6|       Divya|Hyderabad|   95|       Chemistry|
|        3|         3|      Suresh|Bangalore|   92|       Chemistry|
|        4|      NULL|        NULL|     NULL| NULL|         History|
|        5|      NULL|        NULL|     NULL| NULL|       Geography|
|        6|      NULL|        NULL|     NULL| NULL|Computer Science|
+---------+----------+------------+---------+-----+----------------+
```

4. Full Outer Join:

A full outer join returns all rows when there is a match in either the left or the right DataFrame. If there is no match, null values are filled in for the columns from the DataFrame where there is no match.



```
#Full Outer Join
full_outer_joined_df = students_df.join(scores_df, "student_id", "full_outer").join(courses_df, "course_id", "full_outer")
print("\nFull Outer Join Result:")
full_outer_joined_df.show()
```

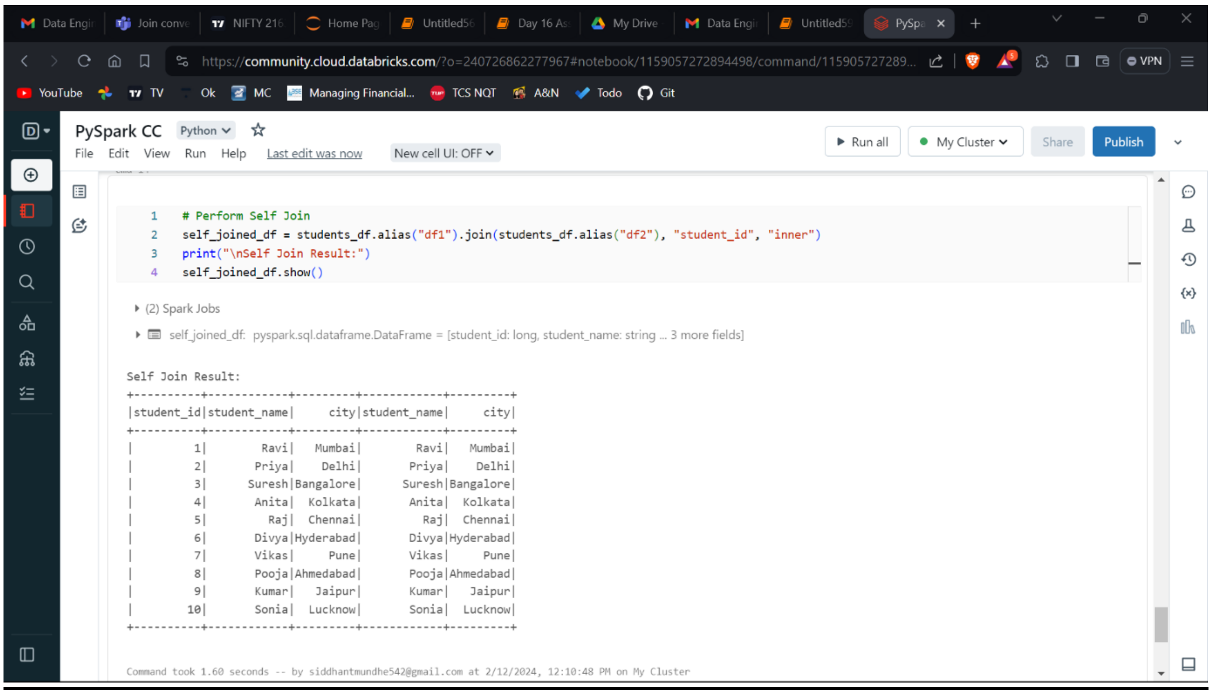
(5) Spark Jobs

full_outer_joined_df: pyspark.sql.dataframe.DataFrame = [course_id: long, student_id: long ... 4 more fields]

```
Full Outer Join Result:
+---------+----------+------------+---------+-----+----------------+
|course_id|student_id|student_name|     city|score|     course_name|
+---------+----------+------------+---------+-----+----------------+
|        1|         1|        Ravi|   Mumbai|   90|            Math|
|        1|         4|       Anita|  Kolkata|   88|            Math|
|        1|         7|       Vikas|     Pune|   75|            Math|
|        1|        10|       Sonia|  Lucknow|   89|            Math|
|        2|         2|       Priya|    Delhi|   85|         Physics|
|        2|         5|         Raj|  Chennai|   78|         Physics|
|        2|         8|       Pooja|Ahmedabad|   80|         Physics|
|        3|         3|      Suresh|Bangalore|   92|       Chemistry|
|        3|         6|       Divya|Hyderabad|   95|       Chemistry|
|        3|         9|       Kumar|   Jaipur|   87|       Chemistry|
|        4|      NULL|        NULL|     NULL| NULL|         History|
|        5|      NULL|        NULL|     NULL| NULL|       Geography|
|        6|      NULL|        NULL|     NULL| NULL|Computer Science|
+---------+----------+------------+---------+-----+----------------+
```

5. Self Join:

A self join is a join where a DataFrame is joined with itself based on a specified condition.



```
# Perform Self Join
self_joined_df = students_df.alias("df1").join(students_df.alias("df2"), "student_id", "inner")
print("\nSelf Join Result:")
self_joined_df.show()
```

(2) Spark Jobs

self_joined_df: pyspark.sql.dataframe.DataFrame = [student_id: long, student_name: string ... 3 more fields]

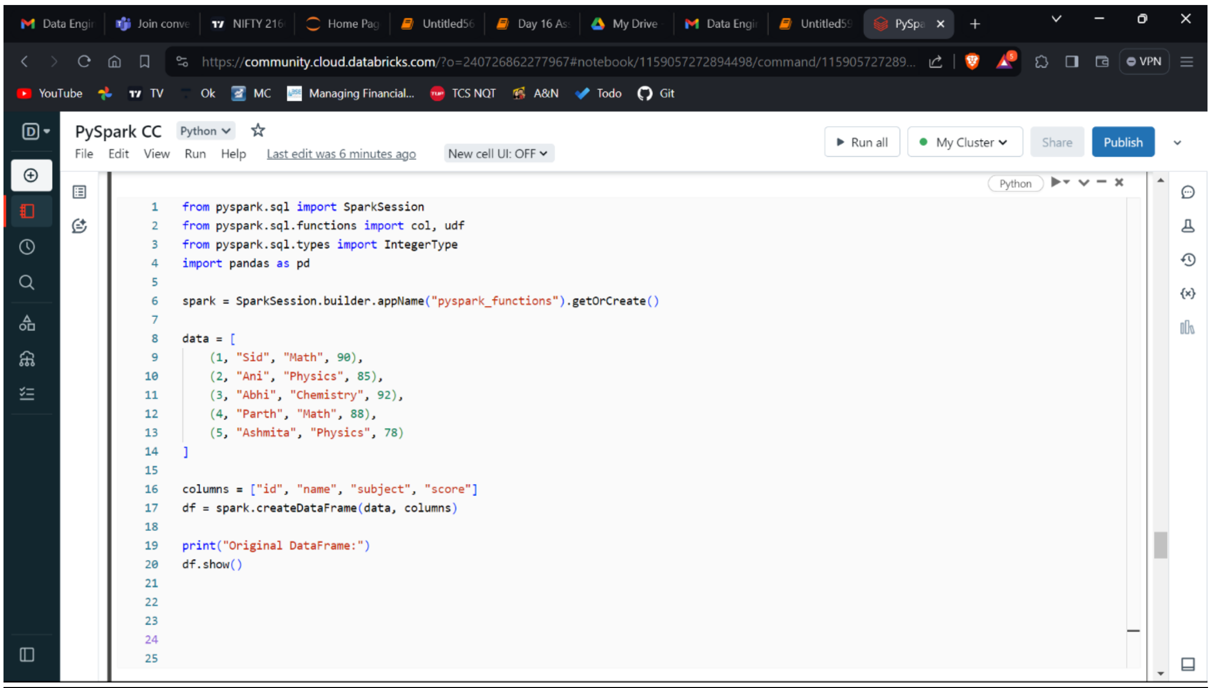
```
Self Join Result:
+----------+------------+---------+------------+---------+
|student_id|student_name|     city|student_name|     city|
+----------+------------+---------+------------+---------+
|         1|        Ravi|   Mumbai|        Ravi|   Mumbai|
|         2|       Priya|    Delhi|       Priya|    Delhi|
|         3|      Suresh|Bangalore|      Suresh|Bangalore|
|         4|       Anita|  Kolkata|       Anita|  Kolkata|
|         5|         Raj|  Chennai|         Raj|  Chennai|
|         6|       Divya|Hyderabad|       Divya|Hyderabad|
|         7|       Vikas|     Pune|       Vikas|     Pune|
|         8|       Pooja|Ahmedabad|       Pooja|Ahmedabad|
|         9|       Kumar|   Jaipur|       Kumar|   Jaipur|
|        10|       Sonia|  Lucknow|       Sonia|  Lucknow|
+----------+------------+---------+------------+---------+


Command took 1.60 seconds -- by siddhantmundhe542@gmail.com at 2/12/2024, 12:10:48 PM on My Cluster
```

Applying functions to a PySpark DataFrame

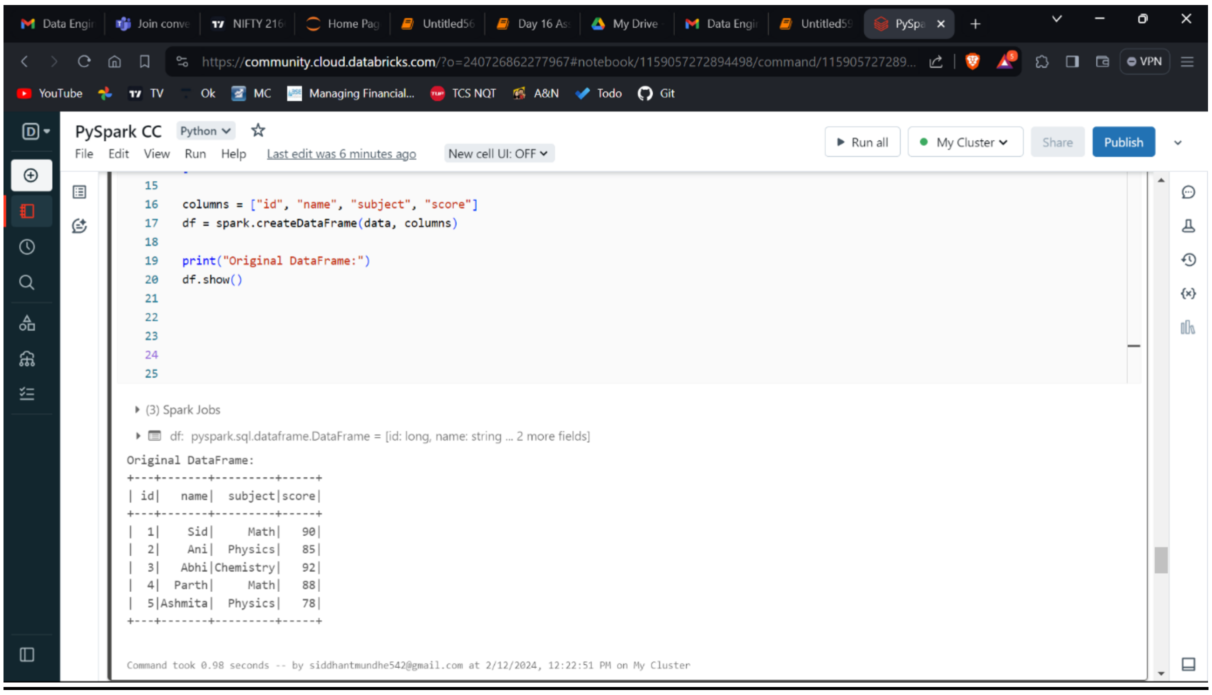
-Applying functions to a PySpark DataFrame is similar to working with functions in a Pandas DataFrame.

-PySpark provides a withColumn method to apply transformations and create new columns.

- Additionally, PySpark supports user-defined functions (UDFs) for more complex operations.



```python
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, udf
from pyspark.sql.types import IntegerType
import pandas as pd

spark = SparkSession.builder.appName("pyspark_functions").getOrCreate()

data = [
    (1, "Sid", "Math", 90),
    (2, "Ani", "Physics", 85),
    (3, "Abhi", "Chemistry", 92),
    (4, "Parth", "Math", 88),
    (5, "Ashmita", "Physics", 78)
]

columns = ["id", "name", "subject", "score"]
df = spark.createDataFrame(data, columns)

print("Original DataFrame:")
df.show()
```



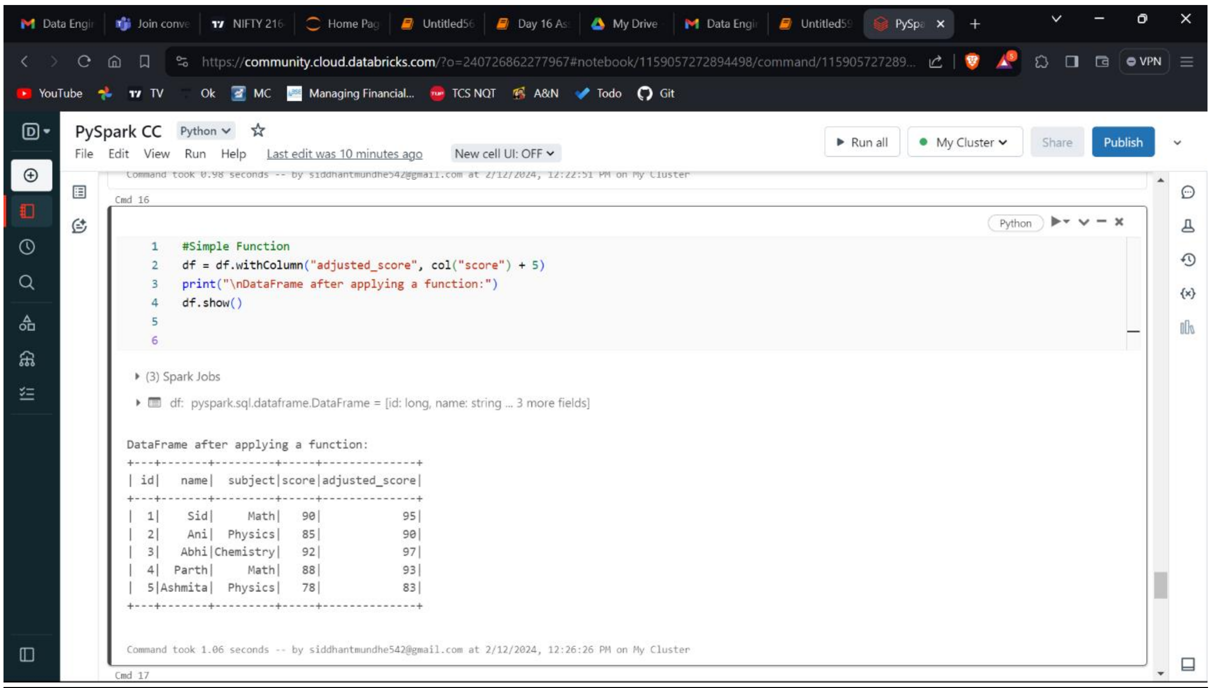
```python
columns = ["id", "name", "subject", "score"]
df = spark.createDataFrame(data, columns)

print("Original DataFrame:")
df.show()
```

(3) Spark Jobs

df: pyspark.sql.dataframe.DataFrame = [id: long, name: string ... 2 more fields]

```
Original DataFrame:
+---+-------+---------+-----+
| id|   name|  subject|score|
+---+-------+---------+-----+
|  1|    Sid|     Math|   90|
|  2|    Ani|  Physics|   85|
|  3|   Abhi|Chemistry|   92|
|  4|  Parth|     Math|   88|
|  5|Ashmita|  Physics|   78|
+---+-------+---------+-----+
```

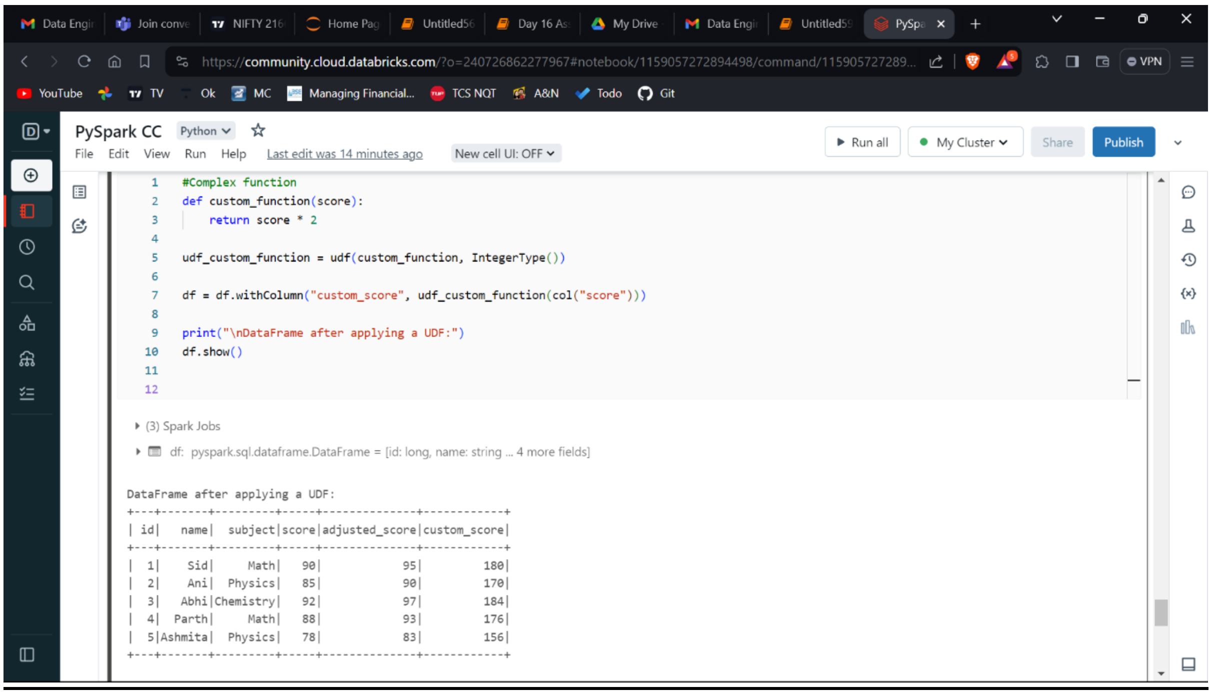
```
Command took 0.98 seconds -- by siddhantmundhe542@gmail.com at 2/12/2024, 12:22:51 PM on My Cluster
```



```
Command took 0.98 seconds -- by siddhantmundhe542@gmail.com at 2/12/2024, 12:22:51 PM on My Cluster
Cmd 16
```

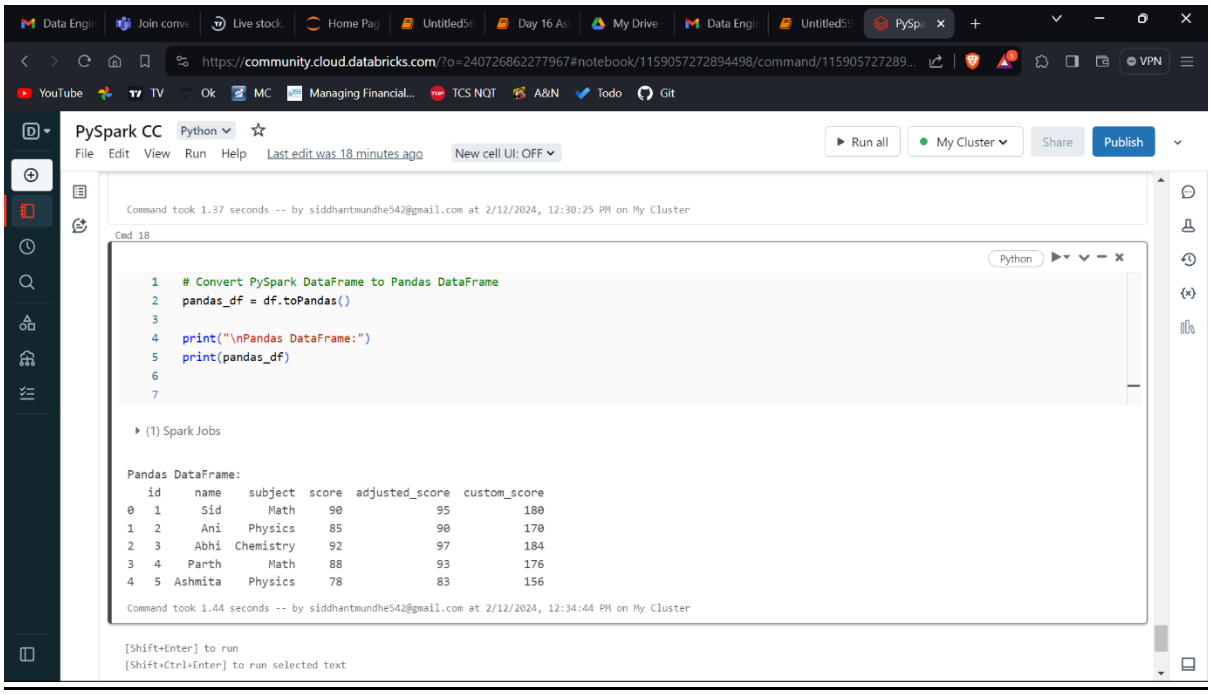
```python
#Simple Function
df = df.withColumn("adjusted_score", col("score") + 5)
print("\nDataFrame after applying a function:")
df.show()
```

```
(3) Spark Jobs
df: pyspark.sql.dataframe.DataFrame = [id: long, name: string ... 3 more fields]

DataFrame after applying a function:
+---+-------+---------+-----+--------------+
| id|   name|  subject|score|adjusted_score|
+---+-------+---------+-----+--------------+
|  1|    Sid|     Math|   90|            95|
|  2|    Ani|  Physics|   85|            90|
|  3|   Abhi|Chemistry|   92|            97|
|  4|  Parth|     Math|   88|            93|
|  5|Ashmita|  Physics|   78|            83|
+---+-------+---------+-----+--------------+

Command took 1.06 seconds -- by siddhantmundhe542@gmail.com at 2/12/2024, 12:26:26 PM on My Cluster
Cmd 17
```



```python
#Complex function
def custom_function(score):
    return score * 2

udf_custom_function = udf(custom_function, IntegerType())

df = df.withColumn("custom_score", udf_custom_function(col("score")))

print("\nDataFrame after applying a UDF:")
df.show()
```

```
(3) Spark Jobs
df: pyspark.sql.dataframe.DataFrame = [id: long, name: string ... 4 more fields]

DataFrame after applying a UDF:
+---+-------+---------+-----+--------------+------------+
| id|   name|  subject|score|adjusted_score|custom_score|
+---+-------+---------+-----+--------------+------------+
|  1|    Sid|     Math|   90|            95|         180|
|  2|    Ani|  Physics|   85|            90|         170|
|  3|   Abhi|Chemistry|   92|            97|         184|
|  4|  Parth|     Math|   88|            93|         176|
|  5|Ashmita|  Physics|   78|            83|         156|
+---+-------+---------+-----+--------------+------------+
```



```
Command took 1.37 seconds -- by siddhantmundhe542@gmail.com at 2/12/2024, 12:30:25 PM on My Cluster
```

Cmd 18

```python
# Convert PySpark DataFrame to Pandas DataFrame
pandas_df = df.toPandas()

print("\nPandas DataFrame:")
print(pandas_df)

```

(1) Spark Jobs

```
Pandas DataFrame:
   id     name    subject  score  adjusted_score  custom_score
0   1      Sid       Math     90              95           180
1   2      Ani    Physics     85              90           170
2   3     Abhi  Chemistry     92              97           184
3   4    Parth       Math     88              93           176
4   5  Ashmita    Physics     78              83           156
Command took 1.44 seconds -- by siddhantmundhe542@gmail.com at 2/12/2024, 12:34:44 PM on My Cluster

[Shift+Enter] to run
[Shift+Ctrl+Enter] to run selected text
```