

Name : Parth Nandedkar  
Date : 12 Feb 2024  
Topics : PySpark  
Batch : Data Engineering Batch-1

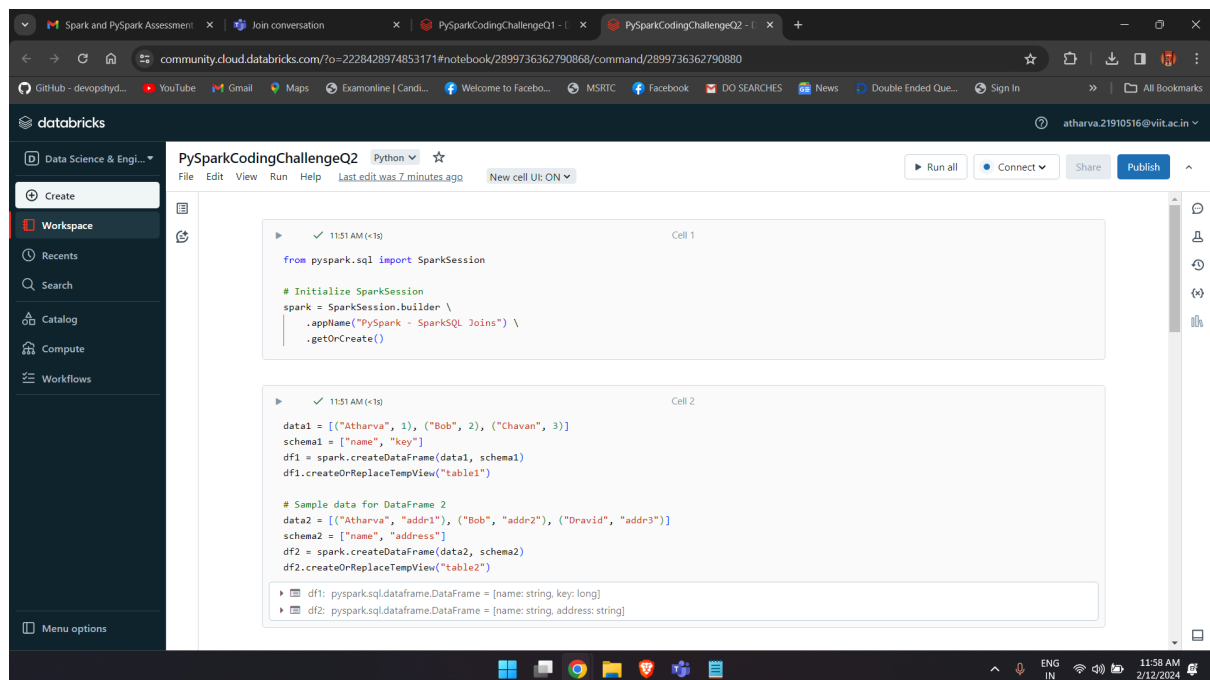
## Q2. Execute Pyspark -sparksql joins & Applying Functions in a Pandas DataFrame.

Spark SQL is a module in Apache Spark that provides a higher-level abstraction for working with structured and semi-structured data. It allows you to run SQL queries and access data using SQL statements, as well as perform various data manipulation and analysis tasks using DataFrame API and SQL functions.

Pandas are used to manipulate/work on big data. Here we are using spark sql and pandas so that data from databases can be handled by pandas.

Here I'm using databricks to run Join queries as well as Pandas functions to get desired output.

### Initialising the Spark Session :



```
from pyspark.sql import SparkSession

# Initialize SparkSession
spark = SparkSession.builder \
    .appName("PySpark - SparkSQL Joins") \
    .getOrCreate()

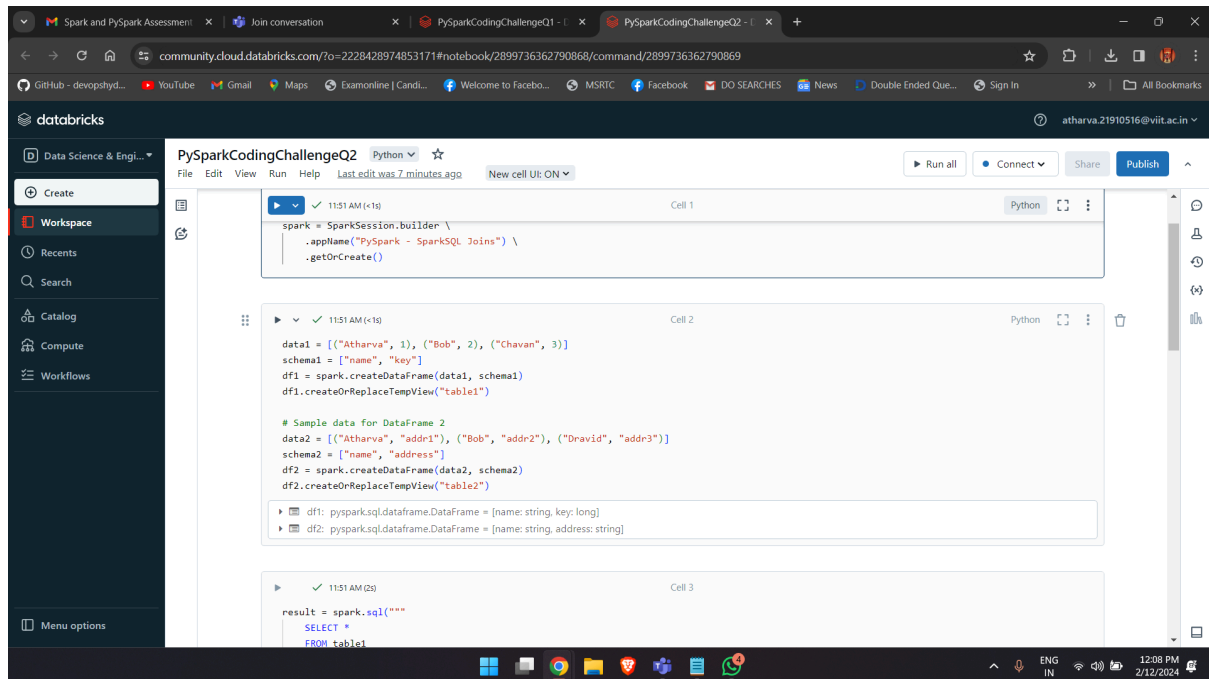
data1 = [("Atharva", 1), ("Bob", 2), ("Chavan", 3)]
schema1 = ["name", "key"]
df1 = spark.createDataFrame(data1, schema1)
df1.createOrReplaceTempView("table1")

# Sample data for DataFrame 2
data2 = [("Atharva", "addr1"), ("Bob", "addr2"), ("Dravid", "addr3")]
schema2 = ["name", "address"]
df2 = spark.createDataFrame(data2, schema2)
df2.createOrReplaceTempView("table2")

df1: pyspark.sql.dataframe.DataFrame = [name: string, key: long]
df2: pyspark.sql.dataframe.DataFrame = [name: string, address: string]
```

## Creating Sample Tables in PySpark :

As we can use data frames as table views and use spark sql queries on those to retrieve data .



The screenshot shows a Databricks notebook interface with three code cells. Cell 1 initializes the Spark session. Cell 2 creates two dataframes, df1 and df2, from Python lists and schemas, and then creates temporary views 'table1' and 'table2' from them. Cell 3 shows a Spark SQL query selecting from 'table1'. The notebook has a sidebar with navigation options like Workspace, Recents, Search, Catalog, Compute, and Workflows. The bottom status bar shows the time as 12:08 PM on 2/12/2024.

```
spark = SparkSession.builder \
    .appName("PySpark - SparkSQL Joins") \
    .getOrCreate()

data1 = [("Atharva", 1), ("Bob", 2), ("Chavan", 3)]
schema1 = ["name", "key"]
df1 = spark.createDataFrame(data1, schema1)
df1.createOrReplaceTempView("table1")

# Sample data for DataFrame 2
data2 = [("Atharva", "addr1"), ("Bob", "addr2"), ("Dnavid", "addr3")]
schema2 = ["name", "address"]
df2 = spark.createDataFrame(data2, schema2)
df2.createOrReplaceTempView("table2")

df1: pyspark.sql.dataframe.DataFrame = [name: string, key: long]
df2: pyspark.sql.dataframe.DataFrame = [name: string, address: string]

result = spark.sql("""
SELECT *
FROM table1
```

Here **data1** and **schema1** are used to create a dataframe and we created a data frame **df1** so as to get data in it .

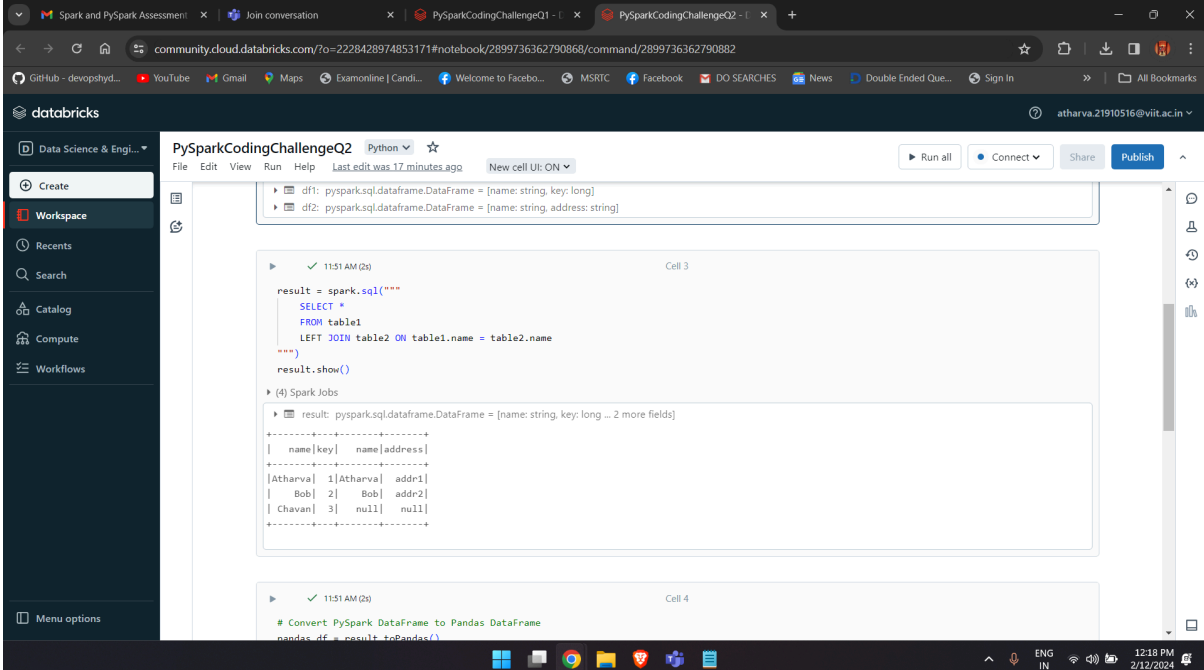
And by using the **createOrReplaceTempView()** method we created **table1** which means table1 will be representing the name and code column .

Same procedure is followed to create table two and use the name **table2** to df2 so that table2 can be used to recognize name and address.

## Using Spark SQL joins :

Spark SQL is an Integrated language to use databases using Spark service.

By using `spark.sql()` method we can give any query as input to retrieve results using spark.



The screenshot shows a Databricks workspace interface. The main area displays a PySpark SQL query in a code cell, which has been executed. The query performs a left join between two tables, `table1` and `table2`, on the `name` column. The results are shown in a table format below the code cell.

```
df1: pyspark.sql.dataframe.DataFrame = [name: string, key: long]
df2: pyspark.sql.dataframe.DataFrame = [name: string, address: string]

result = spark.sql("""
SELECT *
FROM table1
LEFT JOIN table2 ON table1.name = table2.name
""")
result.show()
```

(4) Spark Jobs

name	key	name	address
Atharva	1	Atharva	addr1
Bob	2	Bob	addr-2
Chavan	3	null	null

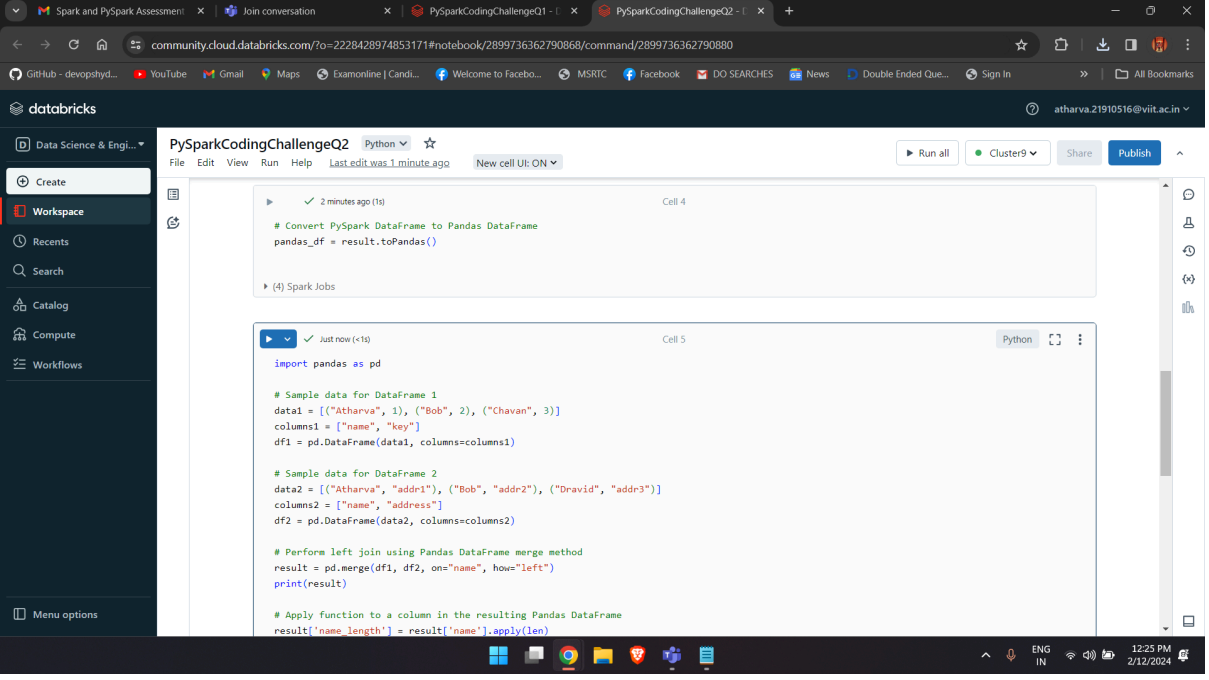
Cell 4

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

Here in the above example **table1** and **table2** are joined on names .  
As table1 is the left table and we are using left join all the values from table1 will be considered and table2 common values will be there.

## Functions in Pandas dataframe :

In pandas data set is converted and results are as follows :



The screenshot shows a Databricks notebook titled "PySparkCodingChallengeQ2". The interface includes a sidebar with navigation options like "Create", "Workspace", "Recents", "Search", "Catalog", "Compute", and "Workflows". The main area displays two code cells. Cell 4 contains a single line of Python code to convert a PySpark DataFrame to a Pandas DataFrame. Cell 5 contains a more complex Python script that creates two sample DataFrames, performs a left join using the merge method, and applies a function to calculate the length of the 'name' column.

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

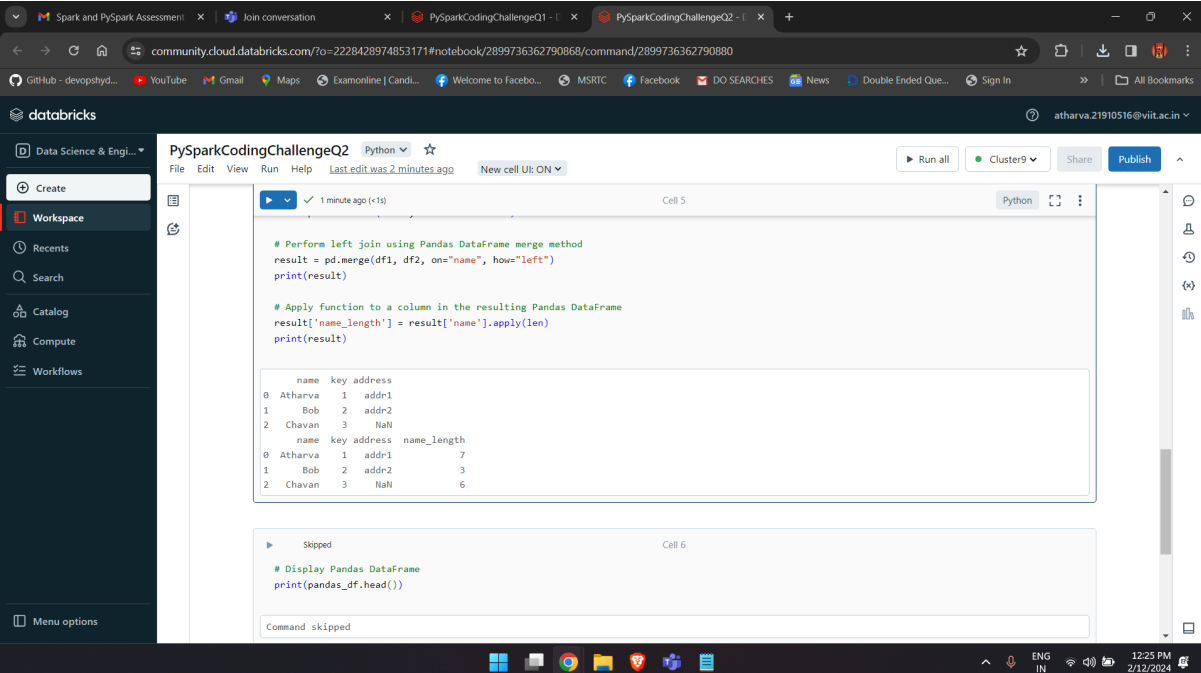
```
import pandas as pd

# Sample data for DataFrame 1
data1 = [("Atharva", 1), ("Bob", 2), ("Chavan", 3)]
columns1 = ["name", "key"]
df1 = pd.DataFrame(data1, columns=columns1)

# Sample data for DataFrame 2
data2 = [("Atharva", "addr1"), ("Bob", "addr2"), ("Draavid", "addr3")]
columns2 = ["name", "address"]
df2 = pd.DataFrame(data2, columns=columns2)

# Perform left join using Pandas DataFrame merge method
result = pd.merge(df1, df2, on="name", how="left")
print(result)

# Apply function to a column in the resulting Pandas DataFrame
result['name_length'] = result['name'].apply(len)
```



This screenshot shows the same Databricks notebook, but now Cell 5 has been executed, and its output is visible. The output consists of two tables. The first table is the result of the left join, showing columns 'name', 'key', and 'address'. The second table shows the result of applying the length function to the 'name' column, with an additional 'name\_length' column. Below the tables, Cell 6 is shown as "Skipped" because its code was not executed.

```
# Perform left join using Pandas DataFrame merge method
result = pd.merge(df1, df2, on="name", how="left")
print(result)

# Apply function to a column in the resulting Pandas DataFrame
result['name_length'] = result['name'].apply(len)
print(result)
```

	name	key	address
0	Atharva	1	addr1
1	Bob	2	addr2
2	Chavan	3	NaN

	name	key	address	name_length
0	Atharva	1	addr1	7
1	Bob	2	addr2	3
2	Chavan	3	NaN	6

```
# Display Pandas DataFrame
print(pandas_df.head())
```

The screenshot shows a Databricks notebook interface with the title "PySparkCodingChallengeQ2". The notebook is running on a cluster named "Cluster9". The code in the cell performs a left join between two DataFrames, `df1` and `df2`, using the `merge()` method. The resulting DataFrame is displayed, showing the columns `name`, `key`, `address`, and `name_length`. The output shows three rows: 0 (Atharva, 1, addr1, 7), 1 (Bob, 2, addr2, 3), and 2 (Chavan, 3, None, 6). The code also includes a comment: `# Display Pandas DataFrame` and `print(pandas_df.head())`.

```
name key address name_length
0 Atharva 1 addr1 7
1 Bob 2 addr2 3
2 Chavan 3 None 6
```

```
# Display Pandas DataFrame
print(pandas_df.head())
```

	name	key	name	address
0	Atharva	1	Atharva	addr1
1	Bob	2	Bob	addr2
2	Chavan	3	None	None

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

Above execution demonstrates how to perform a left join between two Pandas DataFrames (`df1` and `df2`) using the `merge()` method and then apply a function to a column in the resulting DataFrame (`result`). Adjust the data and column names as needed based on your specific use case.