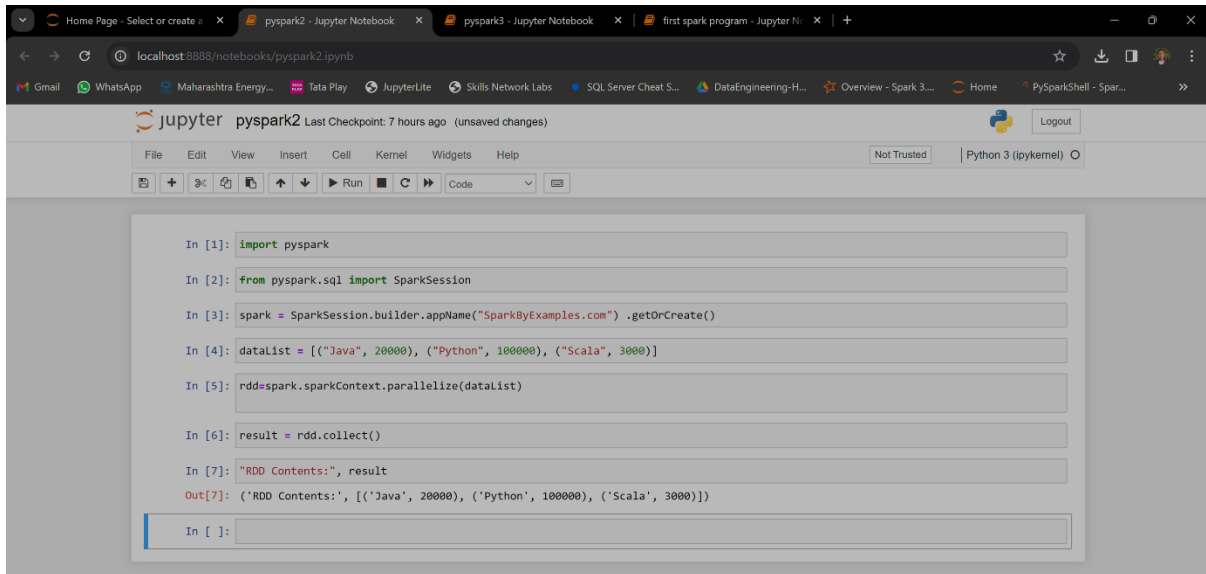


Name: Pradip Bochare

## PySpark Jupyter Notebook



```
In [1]: import pyspark

In [2]: from pyspark.sql import SparkSession

In [3]: spark = SparkSession.builder.appName("SparkByExamples.com").getOrCreate()

In [4]: dataList = [("Java", 20000), ("Python", 100000), ("Scala", 3000)]

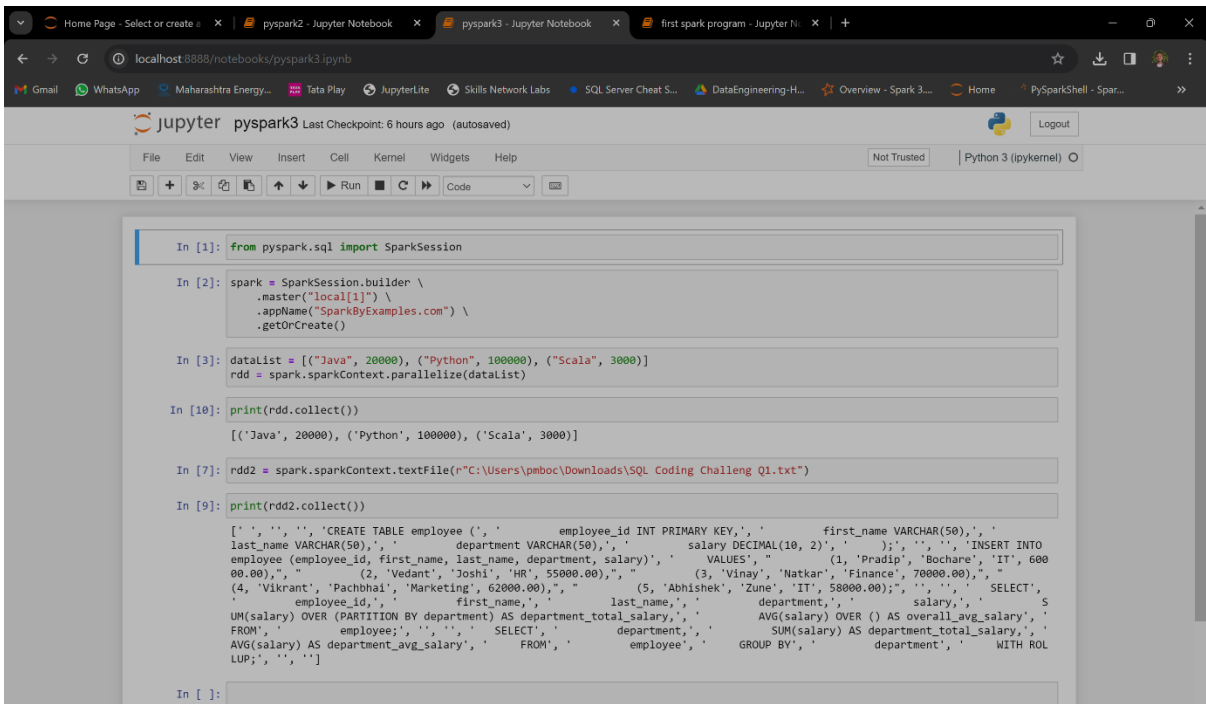
In [5]: rdd=spark.sparkContext.parallelize(dataList)

In [6]: result = rdd.collect()

In [7]: "RDD Contents:", result

Out[7]: ('RDD Contents:', [('Java', 20000), ('Python', 100000), ('Scala', 3000)])

In [ ]:
```



```
In [1]: from pyspark.sql import SparkSession

In [2]: spark = SparkSession.builder \
    .master("local[*]") \
    .appName("SparkByExamples.com") \
    .getOrCreate()

In [3]: dataList = [("Java", 20000), ("Python", 100000), ("Scala", 3000)]
rdd = spark.sparkContext.parallelize(dataList)

In [10]: print(rdd.collect())

[('Java', 20000), ('Python', 100000), ('Scala', 3000)]

In [7]: rdd2 = spark.sparkContext.textFile(r"C:\Users\pmboc\Downloads\SQL Coding Challenge Q1.txt")

In [9]: print(rdd2.collect())

[' ', ' ', ' ', 'CREATE TABLE employee (', ' ', 'employee_id INT PRIMARY KEY,', ' ', 'first_name VARCHAR(50),', ' ',
last_name VARCHAR(50),', ' ', 'department VARCHAR(50),', ' ', 'salary DECIMAL(10, 2),', ' ', ');', ' ', ' ', 'INSERT INTO
employee (employee_id, first_name, last_name, department, salary)', ' ', 'VALUES', ' ', '(1, 'Pradip', 'Bochare', 'IT', 600
00.00),', ' ', '(2, 'Vedant', 'Joshi', 'HR', 55000.00),', ' ', '(3, 'Vinay', 'Natkar', 'Finance', 70000.00),', ' ',
(4, 'Vikrant', 'Pachbhai', 'Marketing', 62000.00),', ' ', '(5, 'Abhishek', 'Zune', 'IT', 58000.00);', ' ', ' ', 'SELECT',
', ' ', 'employee_id,', ' ', 'first_name,', ' ', 'last_name,', ' ', 'department,', ' ', 'salary,', ' ', '
UM(salary) OVER (PARTITION BY department) AS department_total_salary,', ' ', 'AVG(salary) OVER () AS overall_avg_salary', '
FROM', ' ', 'employee;', ' ', ' ', 'SELECT', ' ', 'department,', ' ',
SUM(salary) AS department_total_salary,', '
AVG(salary) AS department_avg_salary', ' ', 'FROM', ' ', 'employee', ' ', 'GROUP BY', ' ', 'department', ' ', 'WITH ROL
LUP;', ' ', ' ']
```

```
In [1]: import pyspark

In [2]: from pyspark.sql import SparkSession

In [3]: spark = SparkSession.builder.appName('pyspark-by-examples').getOrCreate()

In [4]: ArrayData = [ ( "James",[[{"Java","Scala","C++"},{"Spark","Java"}]), ( "Michael",[[{"Spark","Java","C++"}, {"

In [*]: df = spark.createDataFrame(data=ArrayData, schema = ['name','subjects'])

In [*]: df.show()

In [*]: df.printSchema()

In [*]: df.show(truncate=False)

In [ ]:
```

## Result

```
In [6]: df.show()

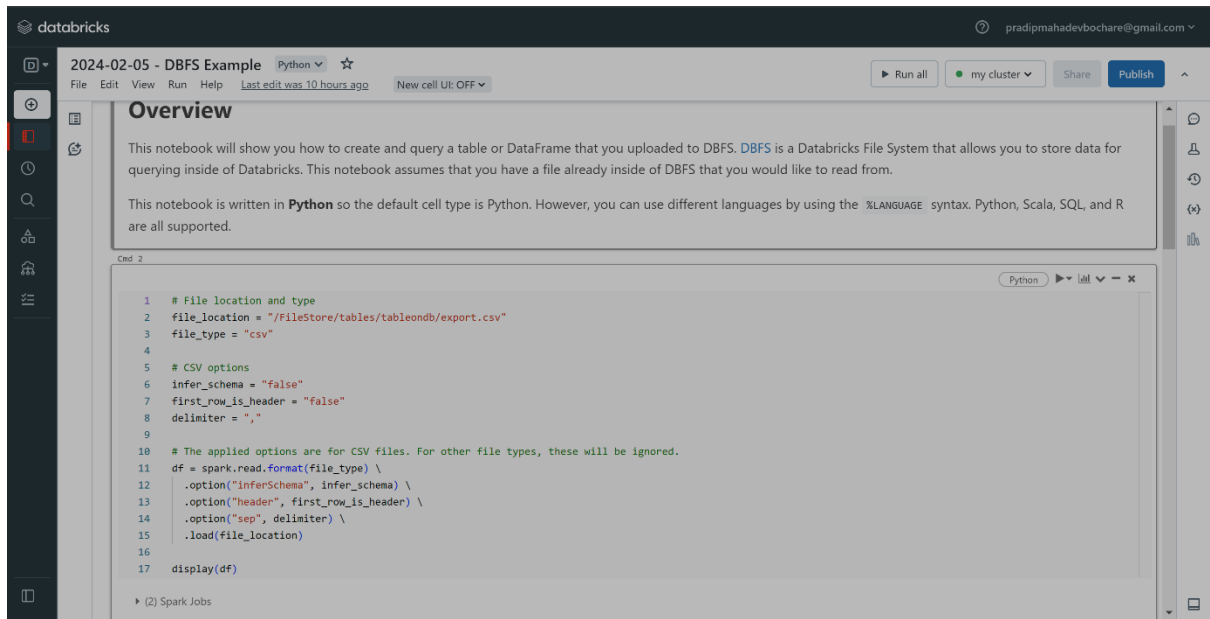
+-----+-----+
| name | subjects |
+-----+-----+
| James|[[Java, Scala, C+...|
| Michael|[[Spark, Java, C+...|
| Robert|[[CSharp, VB], [S...|
+-----+-----+

In [7]: df.printSchema()

root
 |-- name: string (nullable = true)
 |-- subjects: array (nullable = true)
 |    |-- element: array (containsNull = true)
 |    |    |-- element: string (containsNull = true)

In [8]: df.show(truncate=False)

+name |subjects |
+-----+-----+
|James |[[Java, Scala, C++, [Spark, Java]]|
|Michael|[[Spark, Java, C++, [Spark, Java]]|
|Robert |[[CSharp, VB], [Spark, Python]] |
+-----+-----+
```



databricks

2024-02-05 - DBFS Example

Python

☆

File Edit View Run Help

Last edit was 10 hours ago

New cell UI: OFF

Run all

my cluster

Share

Publish

(2) Spark Jobs

df: pyspark.sql.dataframe.DataFrame = [c0: string, \_c1: string ... 9 more fields]

Table

	_c0	_c1	_c2	_c3	_c4	_c5	_c6	_c7	_c8	_c9	_c10
1	_c0	carat	cut	color	clarity	depth	table	price	x	y	z
2	1	0.23	Ideal	E	SI2	61.5	55	326	3.95	3.98	2.43
3	2	0.21	Premium	E	SI1	59.8	61	326	3.89	3.84	2.31
4	3	0.23	Good	E	VS1	56.9	65	327	4.05	4.07	2.31
5	4	0.29	Premium	I	VS2	62.4	58	334	4.2	4.23	2.63
6	5	0.31	Good	J	SI2	63.3	58	335	4.34	4.35	2.75
7	6	0.24	Very Good	J	VVS2	62.8	57	336	3.94	3.96	2.48

Download

10,000 rows | Truncated data | 3.10 seconds runtime

Refreshed 4 minutes ago

Command took 3.10 seconds -- by pradipmahadevbochare@gmail.com at 05/02/2024, 22:12:58 on my cluster

Cmd 3

```

1 # Create a view or table
2
3 temp_table_name = "export_csv"
4
5 df.createOrReplaceTempView(temp_table_name)

```

Command took 0.24 seconds -- by pradipmahadevbochare@gmail.com at 05/02/2024, 22:12:58 on my cluster

Cmd 4

databricks

2024-02-05 - DBFS Example

Python

☆

File Edit View Run Help

Last edit was 10 hours ago

New cell UI: Off

Run all

my cluster

Share

Publish

1 %sql

2

3 /\* Query the created temp table in a SQL cell \*/

4

5 select \* from `export\_csv`

(1) Spark Jobs

\_sql\$df pyspark.sql.dataframe.DataFrame = [\_c0: string, \_c1: string ... 9 more fields]

Table

	_c0	_c1	_c2	_c3	_c4	_c5	_c6	_c7	_c8	_c9	_c10
1	_c0	carat	cut	color	clarity	depth	table	price	x	y	z
2	1	0.23	Ideal	E	SI2	61.5	55	326	3.95	3.98	2.43
3	2	0.21	Premium	E	SI1	59.8	61	326	3.89	3.84	2.31
4	3	0.23	Good	E	VSI	56.9	65	327	4.05	4.07	2.31
5	4	0.29	Premium	I	VSI	62.4	58	334	4.2	4.23	2.63
6	5	0.31	Good	J	SI2	63.3	58	335	4.34	4.35	2.75
7	6	0.24	Verv Good	J	VVS2	62.8	57	336	3.94	3.96	2.48

Download

10,000 rows | Truncated data | 1.78 seconds runtime

This result is stored as PySpark data frame \_sql\$df and in the IPython output cache as Out[3].

Learn more

Command took 1.78 seconds -- by pradipmahadevbochare@gmail.com at 05/02/2024, 22:12:58 on my cluster

\* **PySpark**:- Apache spark library written in Python application using Apache spark capabilities.

PySpark is a python API which is an analytical processing engine for large scale powerful distributed data processing and machine learning applications.

\* **Apache Spark**:-

open-source unified analytics engine used for large scale data processing

- spark can run on single-node machines or multi-node machines (cluster).

\* **PySpark Features**:-

- In memory computation

- Distributed processing using parallelize

- can be used with many cluster managers

- Fault tolerant

- Immutable

- Lazy evaluation

- cache & persistence

- Inbuilt optimization when using DataFrames

- supports ANSI SQL.

\* **Advantages of PySpark**:-

- general purpose, in-memory, distributed processing engine

- Applications running are 100x faster than traditional syst.

- data integration pipelines

- We can process data from Hadoop, AWS S3 & many file

- used to process real-time data using streaming & Kafka

- has machine learning & graph libraries.

Version supported to pyspark.

Language Supported version.

Python 3.8

Java Java 8, 11, 13, 17 & latest version  
Java 8 versions prior to 8u371 have  
been deprecated

Scala 2.12 and 2.13

R 3.5

Apache Kafka

open-source distributed event streaming platform  
used by thousands of companies for high-performance  
data pipelines, streaming analytics, data integration  
and mission-critical applications

Modules & packages

- PySpark RDD (pyspark.RDD)
- PySpark DataFrame and SQL (pyspark.sql)
- PySpark Streaming (pyspark.streaming)
- PySpark MLlib (pyspark.ml, pyspark.mllib)
- PySpark GraphFrames (GraphFrames)
- PySpark Resources (pyspark.resources)