

**VAISHALI BOKADIYA**  
**PYSPARK DAY 5 ASSESSMENT**

## Notes:

### Spark SQL

- Spark SQL is a Spark module for structured data processing.
- Spark SQL is a component on top of Spark core that introduces a new data abstraction called SchemaRDD.

Spark SQL	Spark streaming	MLlib	GraphX	

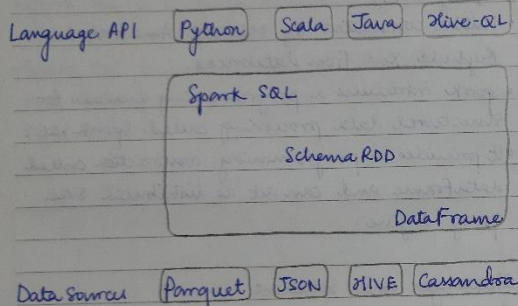
Apache Spark

- Spark SQL was first released in Spark 1.0 (May 2014)
- Initially committed by Michael Armbrust & Kyushik Xie from databricks.
- Spark introduces a programming module for structured data processing called Spark SQL.
- It provides a programming abstraction called dataframe and can act as distributed SQL query engine.

### Challenges and Solutions

Challenges	Solution
Perform ETL to and from various (semi or unstructured) data sources.	A dataframe API that can perform relational operations on both external data sources and Spark's built in RDDs.
Perform advanced analytics (eg ML, graph processing) that are hard to express in relational systems	A highly extensible optimizer, catalyst, that uses features of Scala to add composable rules, control rule, control code gen and define extensions.

### Spark SQL architecture



### Spark SQL architecture

- Language API:
  - Spark is compatible with different languages and Spark SQL.
  - It is also supported by these languages - API (python, java, scala, hiveQL)
- Schema RDD:
  - Spark Core is designed with special data structure called RDD.

- Generally, Spark SQL works on schemas, tables and records.
- Therefore, we can use the Schema RDD as temporary table.
- We can call this Schema RDD as DataFrame.

### Data Sources:

- Usually the data source for spark-core is a text file, avro file etc. However, the data sources for Spark SQL is different.
- These are parquet file, JSON document, HIVE tables, and Cassandra database.

### Features of Spark SQL:

- Integrated
  - Seamlessly mix SQL queries with spark programs.
  - Spark SQL lets you query structured data as a distributed dataset (RDD) in Spark, with integrated APIs in python, Scala and Java.
  - This tight integration makes it easy to run SQL queries alongside complex analytic algorithms.



### Unified data access

- Load and query data from a variety of sources.
- Schema-RDDs provide a single interface for efficiently working with structured data, including Apache Hive tables, parquet files and JSON files.

### Hive compatibility

- Run unmodified Hive queries on existing warehouses.
- Spark SQL reuses the Hive frontend and Metastore, giving you full compatibility with existing Hive data, queries, and UDFs.
- Simply install it alongside Hive.

```
SELECT COUNT(*) FROM hiveTable  
WHERE hive_udf(data)
```

### Standard connectivity

- Connect through JDBC or ODBC.
- Spark SQL includes a server mode with industry standard JDBC and ODBC connectivity.

Tableau      ZoomData      Qlik

### Scalability

- Use the same engine for both interactive and long query queries.
- Spark SQL takes advantage of the RDD model to support mid query fault tolerance, letting it scale to large jobs too.
- Do not worry about using a different engine for historical data.

### User defined functions

UDF: Plug in your processing code and invoke it from a HIVE query.

#### - UDF (plain UDF)

- input - single row, output - single row.

#### - UDAF (user defined aggregate function)

- input - multiple rows, output - single row.  
eg. COUNT and MAX.

#### - UDTF (user defined Table generating Functions)

- input - single row, output - multiple rows.

### Spark RDD (Resilient Distributed Datasets)

- RDD is a fundamental data structure of spark.
- It is an immutable distributed collection of objects that can be stored in memory or disk across a cluster.
- Each dataset in RDD is divided into logical partitions, which may be computed on different node of the cluster.
- Parallel functional transformations (map, filter...).
- Automatically rebuilt on failure.
- RDDs can contain any type of Python, Java, or Scala objects, including user defined classes.
- Formally, an RDD is a read only, partitioned collection of records.
- RDDs can be created through deterministic operations on either data on stable storage or other RDDs.
- RDD is a fault-tolerant collection of elements that can be operated on in parallel.

### There are two ways to create RDDs:

- parallelizing an existing collection in your driver program.
- referencing a dataset in an external storage system, such as a shared file system, HDFS, HBase, or any data source offering a Hadoop input format.

- Spark makes use of the concept of RDD to achieve faster and effective mapreduce operations.

### Dataset and dataframe

- A distributed collection of data, which is organised into named columns.
- Conceptually, it is equivalent to relational tables with good optimization techniques.
- A dataframe can be constructed from an array of different sources such as Hive tables, structured data files, external databases, or existing RDDs.
- This API was designed for modern big data and data science applications taking inspiration from dataframes in R programming and Pandas in python.



Dataframe - Data is organised into named columns, like a table in a relational database.

Dataset - a distributed collection of data.

- A new interface added in Spark 1.6.
- Static typing and runtime type safety.

Features of dataframe.

- Ability to process the data in the size of Kilobytes to Petabytes on a single node cluster to large cluster.
- Supports ~~data~~ different data formats (avro, csv, elastic search and cassandra) and storage systems (HDFS, HIVE tables, mysql, etc).
- State of art optimization and code generation through the Spark SQL Catalyst optimizer (tree transformation framework).
- Can be easily integrated with all Big data tools and frameworks via Spark-core.
- Provides API for python, Java, Scala, and R programming.

Spark SQL, The whole story

◦ Create and Run Spark Programs Faster:

1. Write less code
2. Read less data
3. Let the optimizer do the hard work.

◦ RDD vs dataframe

◦ unified interface to reading/writing data in a variety of formats

`df = sqlContext.read\`

- `format('json')\`
- `option('samplingRatio','0.1')\`
- `load('/home/data.json')`

◦ Builder methods are

used to specify:

`df.write\`

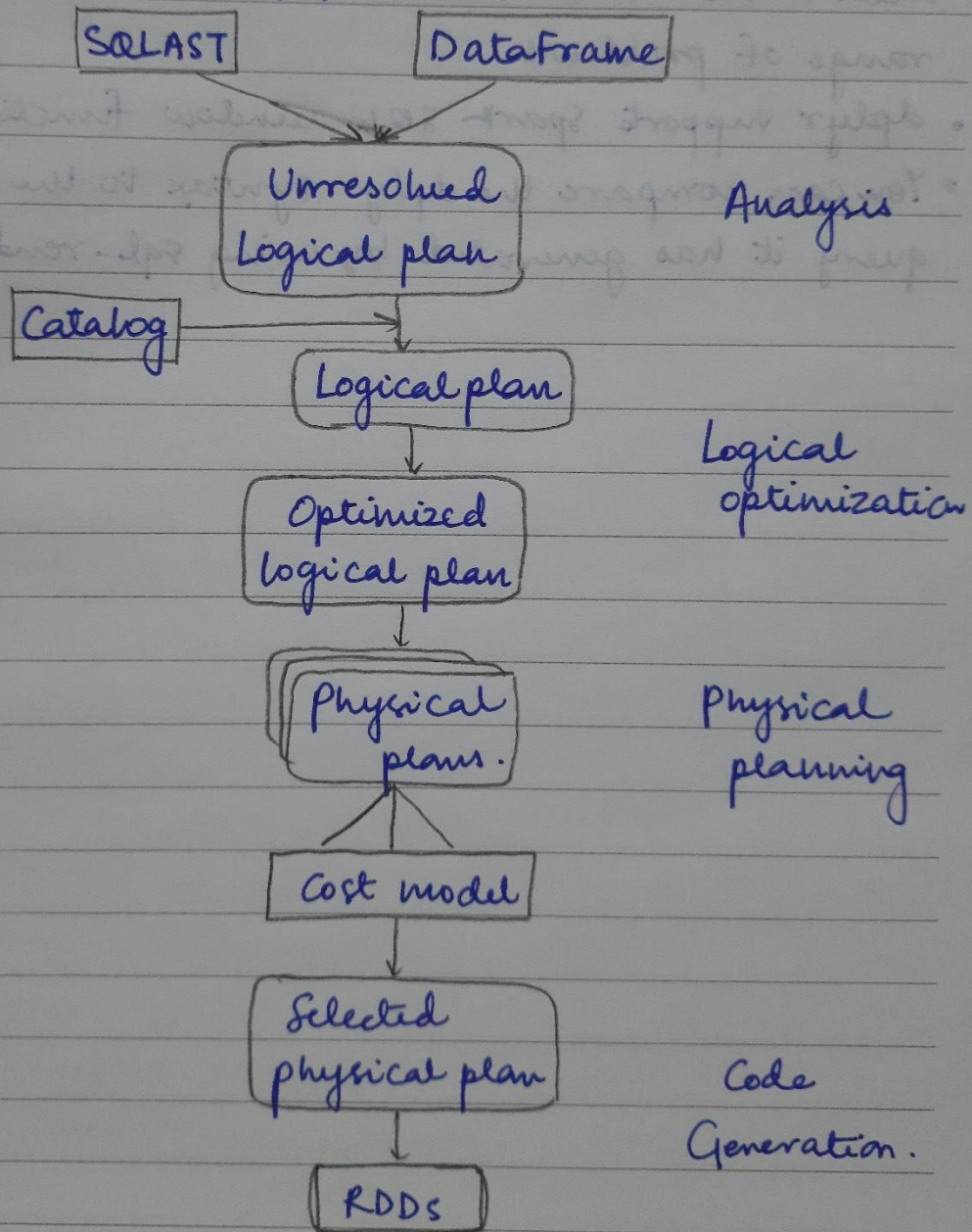
- `format('parquet')\` → format
- `mode('append')\` → partitioning
- `partitionBy('year')\` → handling of existing data and more
- `saveAsTable('FasterData')`
- `load(...)`, `save(...)` or `saveAsTable(...)` functions create new builders for doing I/O.

Apache Hive:

Apache Hive is a distributed, fault tolerant data warehouse system that enables analytics at a massive scale. Hive Metastore (HMS) provides a central repository of metadata that can easily be analyzed to make informed, data driven decisions, and therefore it is a critical component of many data lake architectures. Hive is built on top of Apache Hadoop and supports storage on S3, ads, gs etc through HDFS. Hive allows users to read, write and manage petabytes of data using SQL.

## Plan optimization and execution

DataFrames and SQL share the same optimization execution ~~SQL~~ pipeline.

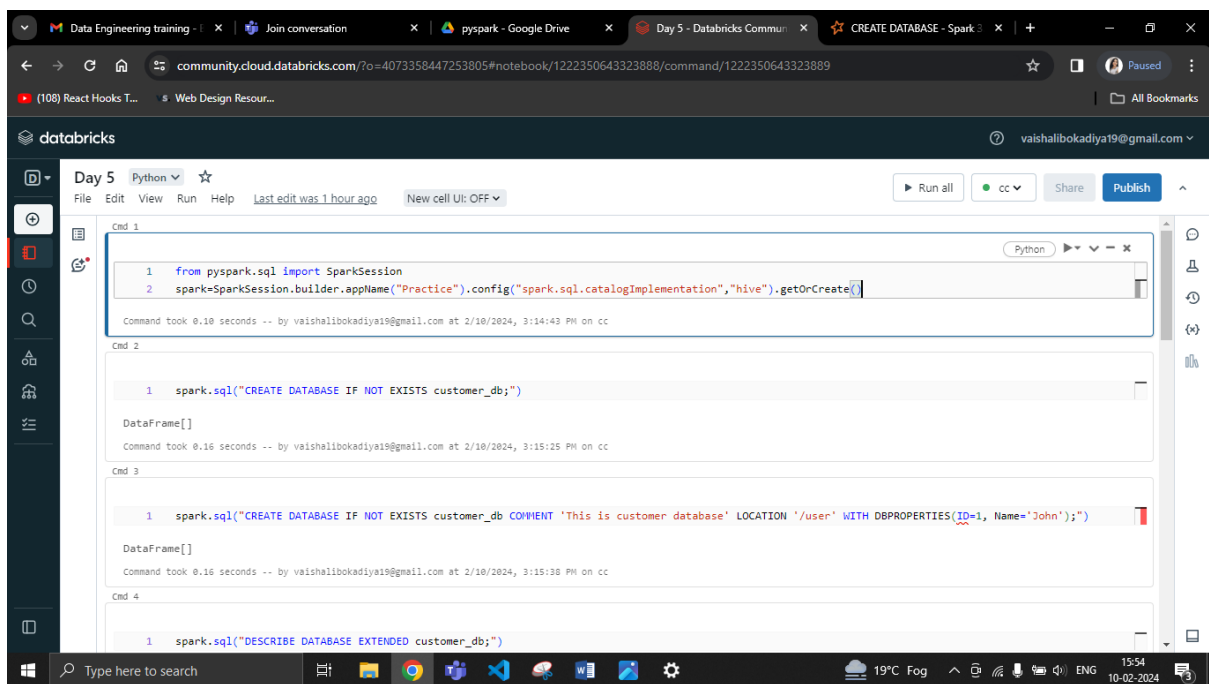




## Windows functions.

- windows functions are used in conjunction with mutate and filter to solve a wide range of problems.
- dplyr supports Spark SQL window functions.
- You can compare the dplyr syntax to the query it has generated by using `sql-render()`.

# Practice:



community.cloud.databricks.com/?o=4073358447253805#notebook/1222350643323888/command/1222350643323889

vaishalibokadiya19@gmail.com

Day 5 Python

File Edit View Run Help Last edit was 1 hour ago New cell UI: OFF

Run all CC Share Publish

Cmd 1

```
1 from pyspark.sql import SparkSession
2 spark=SparkSession.builder.appName("Practice").config("spark.sql.catalogImplementation","hive").getOrCreate()
```

Command took 0.10 seconds -- by vaishalibokadiya19@gmail.com at 2/10/2024, 3:14:43 PM on cc

Cmd 2

```
1 spark.sql("CREATE DATABASE IF NOT EXISTS customer_db;")
```

DataFrame[]

Command took 0.16 seconds -- by vaishalibokadiya19@gmail.com at 2/10/2024, 3:15:25 PM on cc

Cmd 3

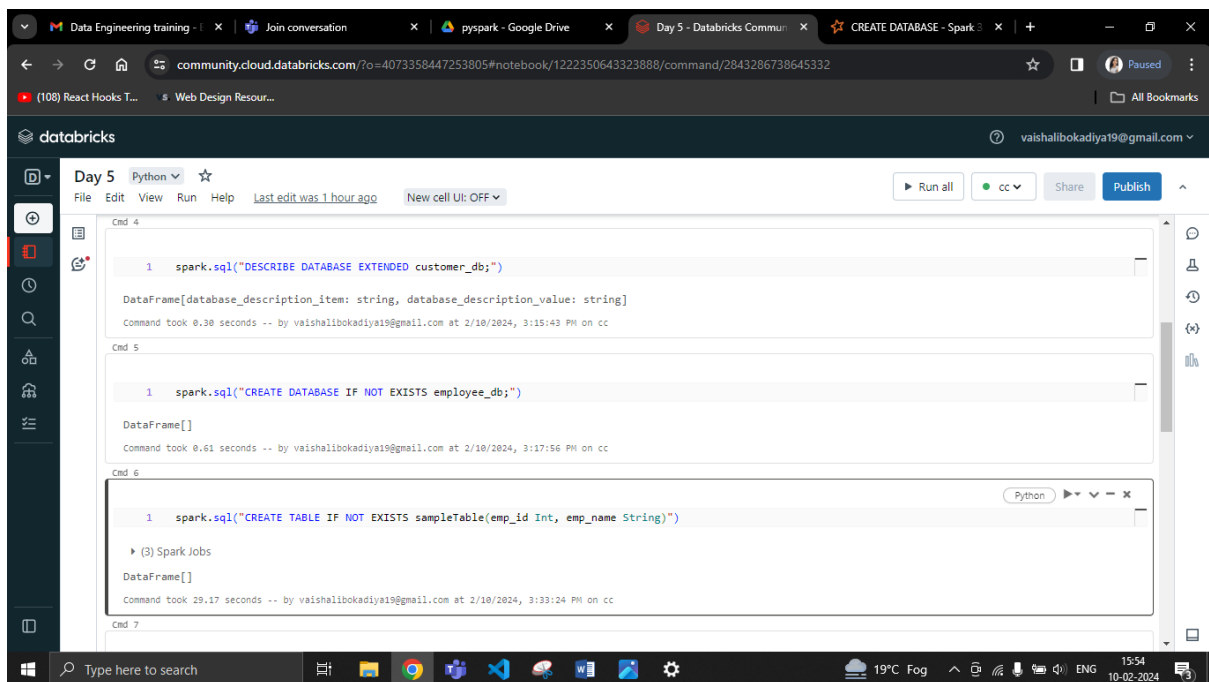
```
1 spark.sql("CREATE DATABASE IF NOT EXISTS customer_db COMMENT 'This is customer database' LOCATION '/user' WITH DBPROPERTIES(10=1, Name='John');")
```

DataFrame[]

Command took 0.16 seconds -- by vaishalibokadiya19@gmail.com at 2/10/2024, 3:15:30 PM on cc

Cmd 4

```
1 spark.sql("DESCRIBE DATABASE EXTENDED customer_db;")
```



community.cloud.databricks.com/?o=4073358447253805#notebook/1222350643323888/command/2843286738645332

vaishalibokadiya19@gmail.com

Day 5 Python

File Edit View Run Help Last edit was 1 hour ago New cell UI: OFF

Run all CC Share Publish

Cmd 4

```
1 spark.sql("DESCRIBE DATABASE EXTENDED customer_db;")
```

DataFrame[database\_description\_item: string, database\_description\_value: string]

Command took 0.30 seconds -- by vaishalibokadiya19@gmail.com at 2/10/2024, 3:15:43 PM on cc

Cmd 5

```
1 spark.sql("CREATE DATABASE IF NOT EXISTS employee_db;")
```

DataFrame[]

Command took 0.61 seconds -- by vaishalibokadiya19@gmail.com at 2/10/2024, 3:17:56 PM on cc

Cmd 6

```
1 spark.sql("CREATE TABLE IF NOT EXISTS sampleTable(emp_id Int, emp_name String)")
```

(3) Spark Jobs

DataFrame[]

Command took 29.17 seconds -- by vaishalibokadiya19@gmail.com at 2/10/2024, 3:33:24 PM on cc

Cmd 7

community.cloud.databricks.com/?o=4073358447253805#notebook/1222350643323888/command/2843286738645332

databricks vaishalibokadiya19@gmail.com

Day 5 Python

File Edit View Run Help Last edit was 1 hour ago New cell UI: OFF

Run all cc Share Publish

Cmd 7

```
1 spark.sql("INSERT INTO sampleTable VALUES (124,'Vaibhav Sharma')")
```

(6) Spark Jobs

DataFrame[num\_affected\_rows: bigint, num\_inserted\_rows: bigint]

Command took 13.68 seconds -- by vaishalibokadiya19@gmail.com at 2/18/2024, 3:34:29 PM on cc

Cmd 8

```
1 spark.sql("SELECT * FROM sampleTable").show()
```

(2) Spark Jobs

emp_id	emp_name
124	Vaibhav Sharma

Command took 5.21 seconds -- by vaishalibokadiya19@gmail.com at 2/18/2024, 3:35:38 PM on cc

Cmd 9

community.cloud.databricks.com/?o=4073358447253805#notebook/1222350643323888/command/2843286738645332

databricks vaishalibokadiya19@gmail.com

Day 5 Python

File Edit View Run Help Last edit was 1 hour ago New cell UI: OFF

Run all cc Share Publish

Cmd 8

```
1 spark.sql("SELECT * FROM sampleTable").show()
```

(2) Spark Jobs

emp_id	emp_name
124	Vaibhav Sharma

Command took 5.21 seconds -- by vaishalibokadiya19@gmail.com at 2/18/2024, 3:35:38 PM on cc

Cmd 9

```
1 spark.sql("ALTER TABLE sampleTable ADD COLUMNS (salary INT, DOB TIMESTAMP)")
```

(3) Spark Jobs

DataFrame[]

Command took 5.76 seconds -- by vaishalibokadiya19@gmail.com at 2/18/2024, 3:37:17 PM on cc

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



Day 5

Python

☆

File Edit View Run Help

Last edit was 1 minute ago

New cell UI: OFF

Run all

cc

Share

Publish

Cmd 10

```
1 spark.read.option("header",True).csv("dbfs:/FileStore/tables/department.csv").createOrReplaceTempView("Department")
```

Python ▶ ▾ ✕

▶ (1) Spark Jobs

Command took 1.11 seconds -- by vaishalibokadiya19@gmail.com at 2/10/2024, 4:09:57 PM on cc

Cmd 11

```
1 spark.sql("select * from department").show()
```

Python ▶ ▾ ✕

▶ (1) Spark Jobs

+-----+  
|department\_id|department\_name|  
+-----+  
101	Engineering
102	Sales
103	Marketing
104	HR
+-----+

Command took 0.49 seconds -- by vaishalibokadiya19@gmail.com at 2/10/2024, 4:11:00 PM on cc

Type here to search

19°C

16:11

10-02-2024