PySpark Assignment 5

Pratik Wani

- Notes:-

PAGE NO.:

Day 5

* Spark SQL

- module for structured data processing
- first relased in may 2014
- Spark introduced a programming model for structured data processing called spark sql
- Provide programing abstraction called dataframe.

* Challenge vs Soln

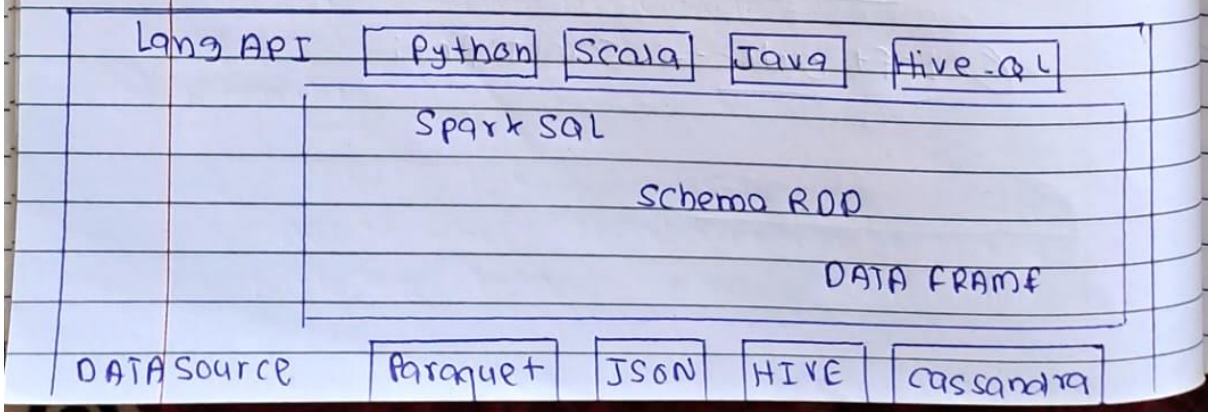
1) perform ETL to and from various data source

Soln DATAFRAME API (perform relational operations on both external data sources)

2) Advanced Analytics:-

Soln
- High extensible optimizer
- Catalyst
- uses features of scala to add composable rule

* Spark SQL Archietecture.



a) Lang API :-
- compitible with various Language and SQL
- support Lang (Py, Scala, Java)

b) Schema RDD :-
- designed with special ds called RDD
- works on table and records
- RDD can be used as temp table
- RDD also called as dataframe.

c) Data source :-
- Data source from spark-core are usually text file, AVRO file etc. but for spark-SQL data sources are different
- Those are parquet file, JSON docs, HIVE table, Cassandra database

* Features

① Integrated :-
- mix SQL queries with spark program
- can query structured data as distributed dataset (RDD) in spark, with integrated API's in python, Scala, Java.
- tight integration make it easy to run SQL query along with complex analytics algorithm.

PAGE NO.:

② Unified data access:-

- Load and query data from various sources
- Schema-RDD provide single interface for efficient working

③ Hive compatibility

- Run unmodified hive queries on existing warehous.
- Spark SQL reuses hive frontend and metastore gives you full compatibility with existing hive data.
- install along with Hive.

* UDF (Plug in our code and invoke it from Hive)
- user defined functions
- UDF (plain UDF)
- UDAF (user-defined Aggregate function)
- UDTF (user-defined Table generating function)

④ Standard connectivity
- JDBV or ODBC
- include a server mode with

⑤ Scalability
- same engine
- Not to worry about different engine for historical data

PAGE NO.:

* Spark RDD:-

- Resilient distributed datasets)
- fundamental data structure of spark
- immutable
- Stored across cluster.
- each dataset in RDD is divided into logical partitions, which may computed on different nodes of cluster.
- Parallel function transformation
- Automatic Rebuilt on failure
- RDD can contant any type of objects including user-defined classes.
- two ways:-
  a) parallelizing
  b) Ref a dataset in external storage.
- RDD is fault tolerant collection of elements that can be apreated in ||el

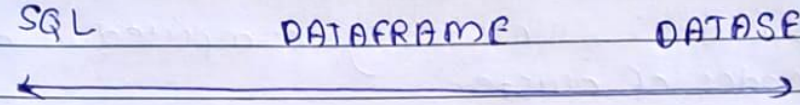
* Dataset and dataframe

- distributed collection of data, which is organized into named columns.
- equivalent to Relational table
- good optimization techn
- Dataframe can be constructed from an array of different sources
- This API was designed for modern big data and data science appn taking inspiration from Datfram in py.

PAGE NO.:

- Dataframe :-
  organised in table

- Dataset :-
  - Static-typing and runtime type-safety



Syn error | Runtime | compile time | compile time

Analysis error

* Feature of dataframe

- process large data
- support different data source
- state of art optimization and code generation through Spark SQL
- can be integrated with big data tools
- API for py, java, scala and R.

PAGE NO.:

* Spark SQL

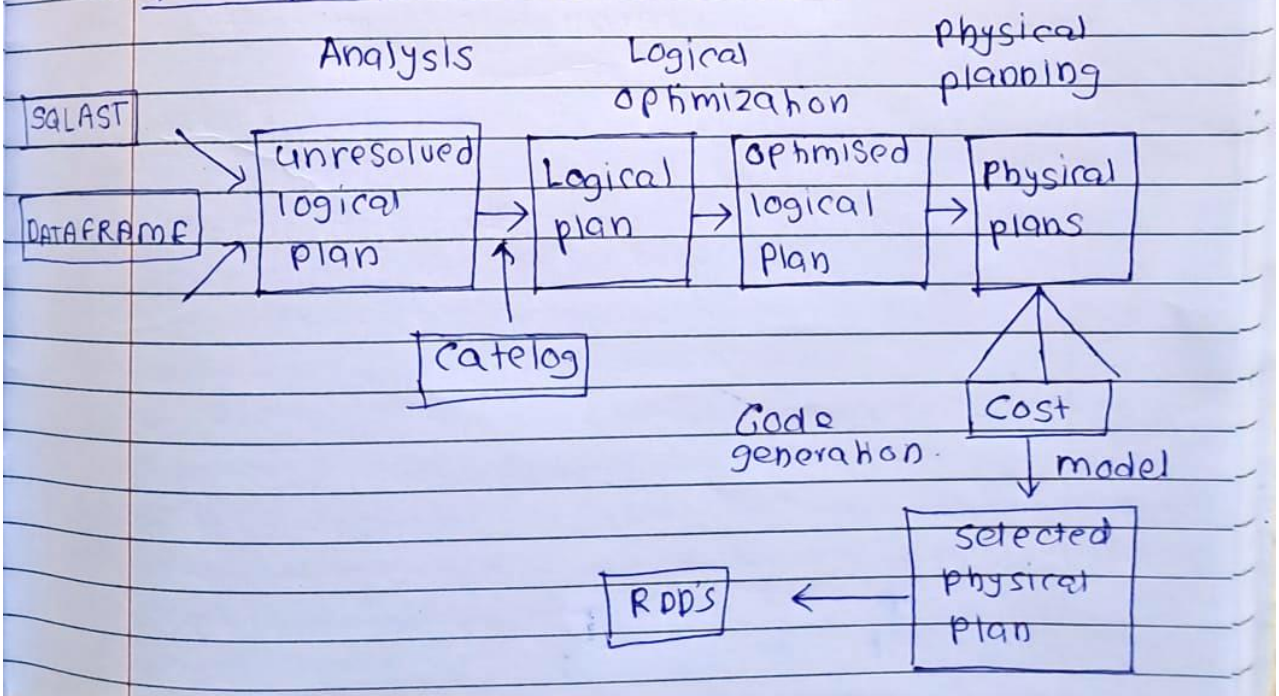
1) Less code 2) Less data

* RDD vs Df

* df = sql.context.read.format('json')
.option('samplingration', '0.1'
.load("path.json")

df.write.format("parquet")
.mode("append")
.partitionBy("year")
.saveASTable("fasterDATA")

* plan optimization and execution :-

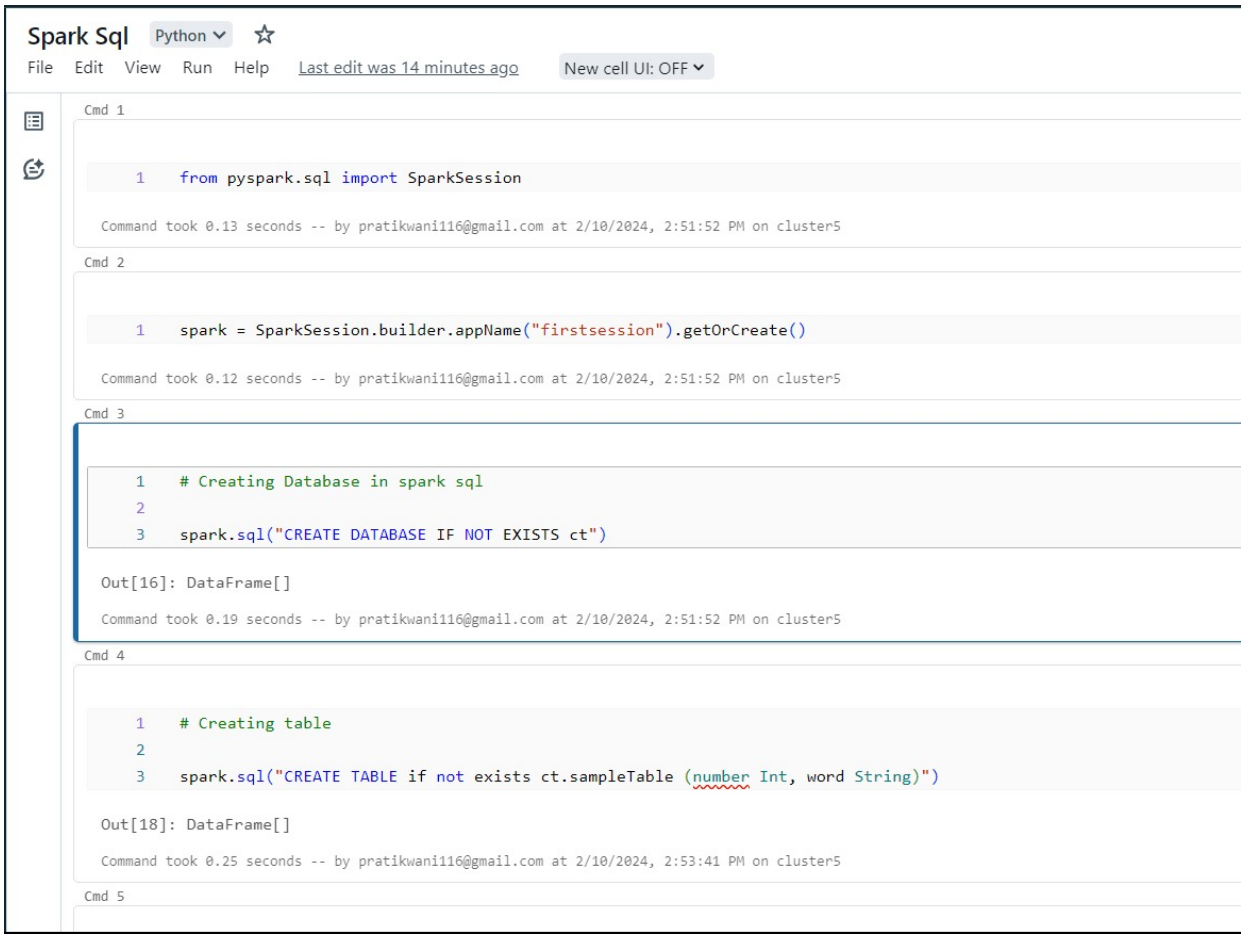


PAGE NO.:

* window function

- dplyr supports sparksql window function
- used in conjunction with mutate and filter to solve a wide rangeofproblem
- sql_render() use to compare dplyr syn.

- Spark SQL:


Spark Sql
Python
File Edit View Run Help
Last edit was 14 minutes ago
New cell UI: OFF
Cmd 1
1 from pyspark.sql import SparkSession
Command took 0.13 seconds -- by pratikwani116@gmail.com at 2/10/2024, 2:51:52 PM on cluster5
Cmd 2
1 spark = SparkSession.builder.appName("firstsession").getOrCreate()
Command took 0.12 seconds -- by pratikwani116@gmail.com at 2/10/2024, 2:51:52 PM on cluster5
Cmd 3
1 # Creating Database in spark sql
2
3 spark.sql("CREATE DATABASE IF NOT EXISTS ct")
Out[16]: DataFrame[]
Command took 0.19 seconds -- by pratikwani116@gmail.com at 2/10/2024, 2:51:52 PM on cluster5
Cmd 4
1 # Creating table
2
3 spark.sql("CREATE TABLE if not exists ct.sampleTable (number Int, word String)")
Out[18]: DataFrame[]
Command took 0.25 seconds -- by pratikwani116@gmail.com at 2/10/2024, 2:53:41 PM on cluster5
Cmd 5

Spark Sql Python

File Edit View Run Help Last edit was 27 minutes ago New cell UI: OFF

Cmd 5

```
1  spark.sql("INSERT INTO ct.sampleTable VALUES (124,'Yogita Yeole')")
```

▸ (6) Spark Jobs

Out[19]: DataFrame[num_affected_rows: bigint, num_inserted_rows: bigint]

Command took 6.55 seconds -- by pratikwani116@gmail.com at 2/10/2024, 2:54:06 PM on cluster5

Cmd 6

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

▸ (3) Spark Jobs

```
+------+------------+
|number|        word|
+------+------------+
|   124|Yogita Yeole|
|   123| Pratik Wani|
+------+------------+
```

Command took 2.27 seconds -- by pratikwani116@gmail.com at 2/10/2024, 2:54:16 PM on cluster5

Cmd 7

```
1  # Alter Table
2  spark.sql("ALTER TABLE ct.sampleTable ADD columns (salary int, DOB timestamp);")
```

▸ (3) Spark Jobs

Out[25]: DataFrame[]

Command took 4.38 seconds -- by pratikwani116@gmail.com at 2/10/2024, 3:01:29 PM on cluster5

Spark Sql Python

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

Cmd 8

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

▸ (3) Spark Jobs

```
+------+------------+------+-------------------+
|number|        word|salary|                DOB|
+------+------------+------+-------------------+
|   123| Pratik Wani|100000|2002-03-10 00:00:00|
|   124|Yogita Yeole|100000|               null|
+------+------------+------+-------------------+
```

Command took 2.43 seconds -- by pratikwani116@gmail.com at 2/10/2024, 3:25:47 PM on cluster5

Cmd 9

```
1  spark.sql("use ct")
```

Out[34]: DataFrame[]

Command took 0.24 seconds -- by pratikwani116@gmail.com at 2/10/2024, 3:08:34 PM on cluster5

Cmd 10

```
1  # Updating the records
2
3  spark.sql("UPDATE ct.sampletable set salary=100000")
```

▸ (7) Spark Jobs

Out[41]: DataFrame[num_affected_rows: bigint]

Command took 8.33 seconds -- by pratikwani116@gmail.com at 2/10/2024, 3:15:41 PM on cluster5

Spark Sql Python

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

Cmd 11

```
1  spark.sql("UPDATE ct.sampletable set DOB='2002-03-10' where number=123 ")
```

(8) Spark Jobs

```
Out[43]: DataFrame[num_affected_rows: bigint]
```

Command took 7.09 seconds -- by pratikwani116@gmail.com at 2/10/2024, 3:17:10 PM on cluster5

Cmd 12

```
1  spark.sql("UPDATE ct.sampletable set DOB='1995-06-24' where number=124 ")
```

(8) Spark Jobs

```
Out[45]: DataFrame[num_affected_rows: bigint]
```

Command took 7.25 seconds -- by pratikwani116@gmail.com at 2/10/2024, 3:26:42 PM on cluster5

Cmd 13

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

(3) Spark Jobs

```
+------+------------+------+-------------------+
|number|        word|salary|                DOB|
+------+------------+------+-------------------+
|   124|Yogita Yeole|100000|1995-06-24 00:00:00|
|   123| Pratik Wani|100000|2002-03-10 00:00:00|
+------+------------+------+-------------------+
```

Command took 2.36 seconds -- by pratikwani116@gmail.com at 2/10/2024, 3:27:06 PM on cluster5

Spark Sql Python

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

```
1  # inserted more data in table
2
3  spark.sql("INSERT INTO ct.sampletable (number, word, salary, DOB) VALUES\
4  (125, 'John Doe', 80000, '1990-05-15 00:00:00'),\
5  (126, 'Jane Smith', 90000, '1988-11-28 00:00:00'),\
6  (127, 'Alice Johnson', 95000, '1993-09-10 00:00:00'),\
7  (128, 'Bob Brown', 85000, '1995-03-20 00:00:00'),\
8  (129, 'Emily Davis', 82000, '1997-07-05 00:00:00'),\
9  (130, 'Michael Wilson', 87000, '1985-12-12 00:00:00'),\
10 (131, 'Sarah Miller', 91000, '1982-04-30 00:00:00'),\
11 (132, 'David Garcia', 92000, '1989-08-18 00:00:00'),\
12 (133, 'Jessica Martinez', 93000, '1991-02-25 00:00:00'),\
13 (134, 'Daniel Taylor', 94000, '1998-06-08 00:00:00');\
14 ")
```

(6) Spark Jobs

```
Out[51]: DataFrame[num_affected_rows: bigint, num_inserted_rows: bigint]
```

Command took 5.72 seconds -- by pratikwani116@gmail.com at 2/10/2024, 3:55:08 PM on cluster5

Cmd 15

```
1  # Total number of rows in table
2  spark.sql("select count(*) from ct.sampletable").show()
```

(4) Spark Jobs

```
+--------+
|count(1)|
+--------+
|      12|
+--------+
```

Spark Sql Python

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

```
1  #  Sorted the data salary wise
2
3  spark.sql("select * from ct.sampletable order by salary desc").show()
```

(1) Spark Jobs

```
+------+----------------+------+-------------------+
|number|            word|salary|                DOB|
+------+----------------+------+-------------------+
|   123|     Pratik Wani|100000|2002-03-10 00:00:00|
|   124|    Yogita Yeole|100000|1995-06-24 00:00:00|
|   127|   Alice Johnson| 95000|1993-09-10 00:00:00|
|   134|   Daniel Taylor| 94000|1998-06-08 00:00:00|
|   133|Jessica Martinez| 93000|1991-02-25 00:00:00|
|   132|    David Garcia| 92000|1989-08-18 00:00:00|
|   131|    Sarah Miller| 91000|1982-04-30 00:00:00|
|   126|      Jane Smith| 90000|1988-11-28 00:00:00|
|   130|  Michael Wilson| 87000|1985-12-12 00:00:00|
|   128|       Bob Brown| 85000|1995-03-20 00:00:00|
|   129|     Emily Davis| 82000|1997-07-05 00:00:00|
|   125|        John Doe| 80000|1990-05-15 00:00:00|
+------+----------------+------+-------------------+
```

Command took 1.39 seconds -- by pratikwani116@gmail.com at 2/10/2024, 3:56:16 PM on cluster5

Cmd 17

```
1  spark.sql("alter table ct.sampletable add column (department string)")
```

(3) Spark Jobs

Out[55]: DataFrame[]

Spark Sql Python

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

Cmd 18

```
1  # Updating the data in new added columns
2
3  spark.sql("update ct.sampletable set department='Data Engineeing' where number In(123,124,125,126)")
4  spark.sql("update ct.sampletable set department='Developers' where number In(127,128,129,130)")
5  spark.sql("update ct.sampletable set department='Testers' where number In(131,132,133,134)")
```

(33) Spark Jobs

Out[56]: DataFrame[num_affected_rows: bigint]

Command took 26.20 seconds -- by pratikwani116@gmail.com at 2/10/2024, 4:22:43 PM on cluster5

Cmd 19

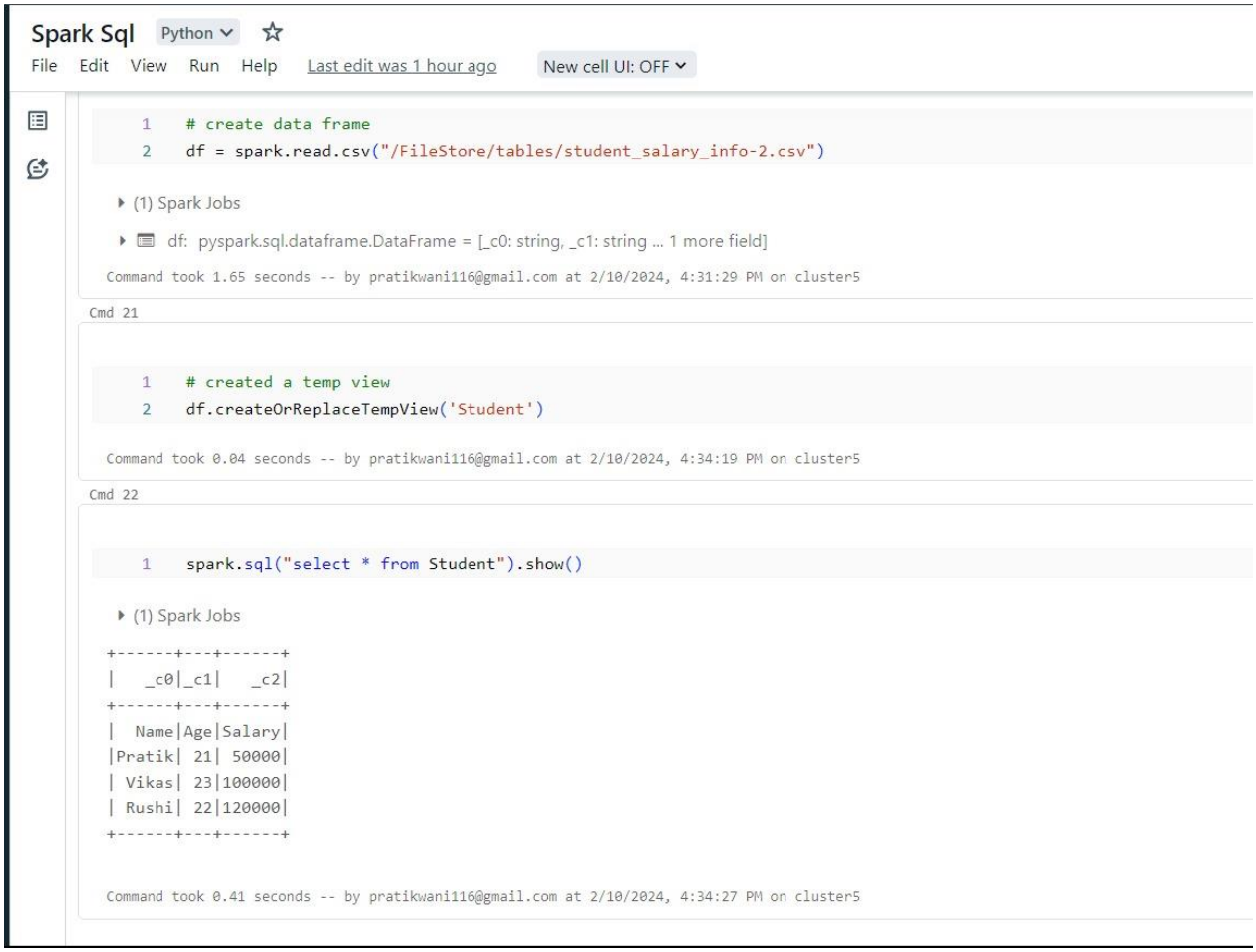
```
1  # Department wise maximun salary
2  spark.sql("select department, max(salary) from ct.sampletable group by department").show()
```

(4) Spark Jobs

```
+---------------+-----------+
|     department|max(salary)|
+---------------+-----------+
|     Developers|      95000|
|        Testers|      94000|
|Data Engineeing|     100000|
+---------------+-----------+
```

Command took 3.10 seconds -- by pratikwani116@gmail.com at 2/10/2024, 4:24:39 PM on cluster5

Cmd 20


Spark Sql
Python
File Edit View Run Help Last edit was 1 hour ago
New cell UI: OFF
# create data frame
df = spark.read.csv("/FileStore/tables/student_salary_info-2.csv")
(1) Spark Jobs
df: pyspark.sql.dataframe.DataFrame = [_c0: string, _c1: string ... 1 more field]
Command took 1.65 seconds -- by pratikwani116@gmail.com at 2/10/2024, 4:31:29 PM on cluster5
Cmd 21
# created a temp view
df.createOrReplaceTempView('Student')
Command took 0.04 seconds -- by pratikwani116@gmail.com at 2/10/2024, 4:34:19 PM on cluster5
Cmd 22
spark.sql("select * from Student").show()
(1) Spark Jobs
+------+---+------+
| _c0|_c1| _c2|
+------+---+------+
| Name|Age|Salary|
|Pratik| 21| 50000|
| Vikas| 23|100000|
| Rushi| 22|120000|
+------+---+------+
Command took 0.41 seconds -- by pratikwani116@gmail.com at 2/10/2024, 4:34:27 PM on cluster5