Faculty of Information Technology
University of Moratuwa
BSc Hons in Information Technology
BSc Hons in Information Technology Management
IN 4410 – Big Data Analytics

Level 1 - Semester 2 **Lab Sheet 08**

Spark SQL

(for Windows)

Spark introduces a programming module for structured data processing called Spark SQL. It provides a programming abstraction called Data Frame and can act as a distributed SQL query engine.

Unlike the basic Spark RDD API, the interfaces provided by Spark SQL provide Spark with more information about the structure of both the data and the computation being performed. Internally, Spark SQL uses this extra information to perform extra optimizations.

There are several ways to interact with Spark SQL including SQL and the Dataset API. When computing a result, the same execution engine is used, independent of which API/language you are using to express the computation. This unification means that developers can easily switch back and forth between different APIs based on which provides the most natural way to express a given transformation.

Let's install PySpark on your PC.

Pre-Requisites:

Method 1 - Jupyter Notebook

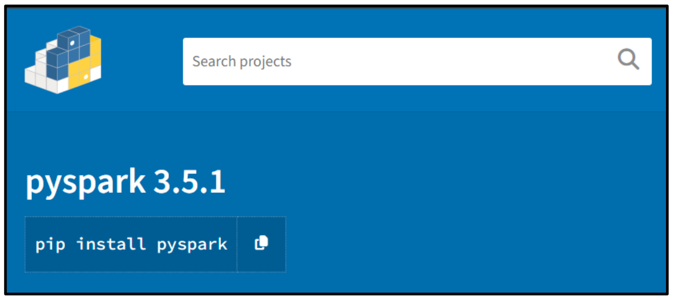
Or

Method 2 - Databricks Community Edition

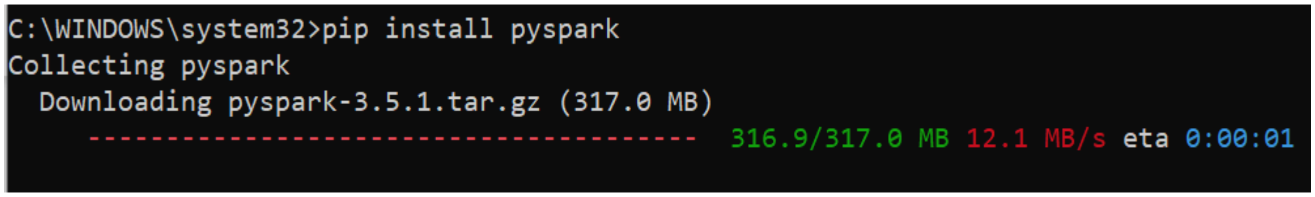
You can follow any of these two methods.

❖ Jupyter Notebook

1) Type Pyspark pypi in the browser and Go to the pyspark web page and copy the command **pip install pyspark.**

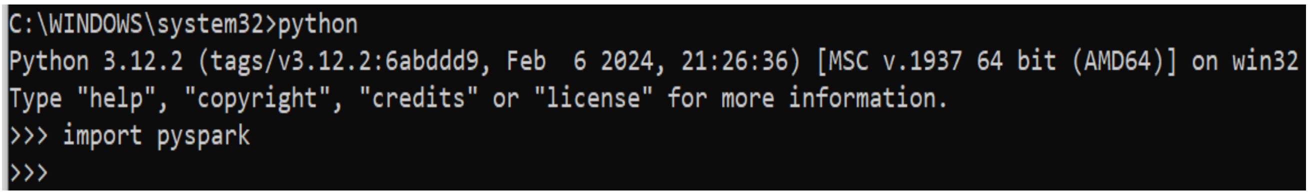


2) Open a command prompt as administrator and paste the command and install pyspark to the machine.



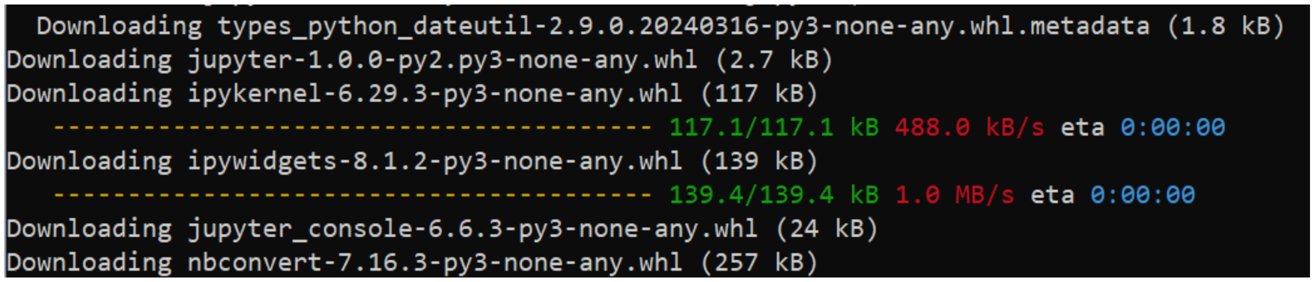
```
C:\WINDOWS\system32>pip install pyspark
Collecting pyspark
  Downloading pyspark-3.5.1.tar.gz (317.0 MB)
     ---------------------------------------  316.9/317.0 MB 12.1 MB/s eta 0:00:01
```

3) After installing pyspark type python and then import pyspark feature
If there are not any errors, it's successfully installed.



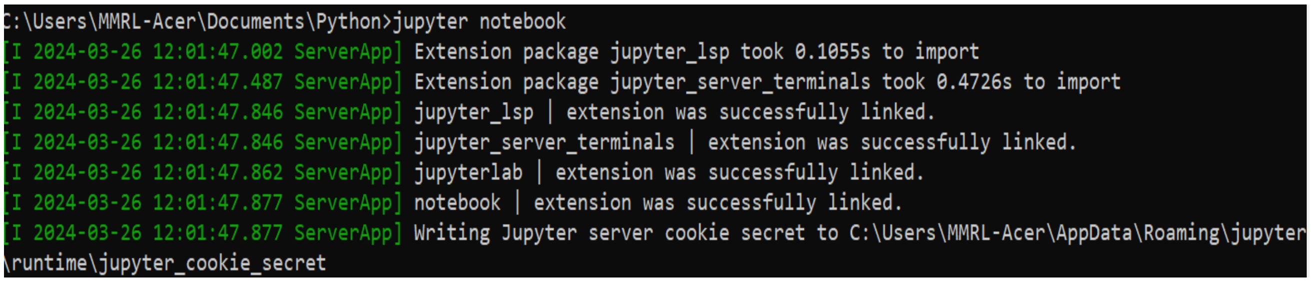
```
C:\WINDOWS\system32>python
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import pyspark
>>>
```

4) Then go to the Scripts folder of python and type cmd command in the path, it will open a new command prompt and type pip install jupyter. it will install the jupyter notebook.



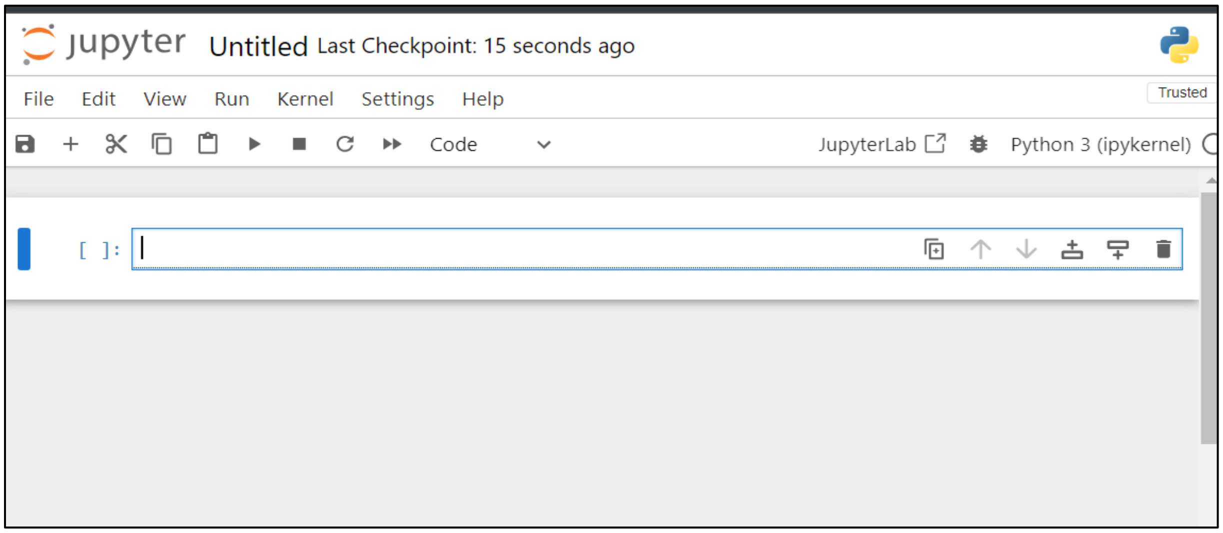
```
  Downloading types_python_dateutil-2.9.0.20240316-py3-none-any.whl.metadata (1.8 kB)
Downloading jupyter-1.0.0-py2.py3-none-any.whl (2.7 kB)
Downloading ipykernel-6.29.3-py3-none-any.whl (117 kB)
   ---------------------------------------- 117.1/117.1 kB 488.0 kB/s eta 0:00:00
Downloading ipywidgets-8.1.2-py3-none-any.whl (139 kB)
   ---------------------------------------- 139.4/139.4 kB 1.0 MB/s eta 0:00:00
Downloading jupyter_console-6.6.3-py3-none-any.whl (24 kB)
Downloading nbconvert-7.16.3-py3-none-any.whl (257 kB)
```

5) Create a new folder in any drive name called python and open that with cmd and type jupyter notebook. It will open a new jupyter notebook in browser.



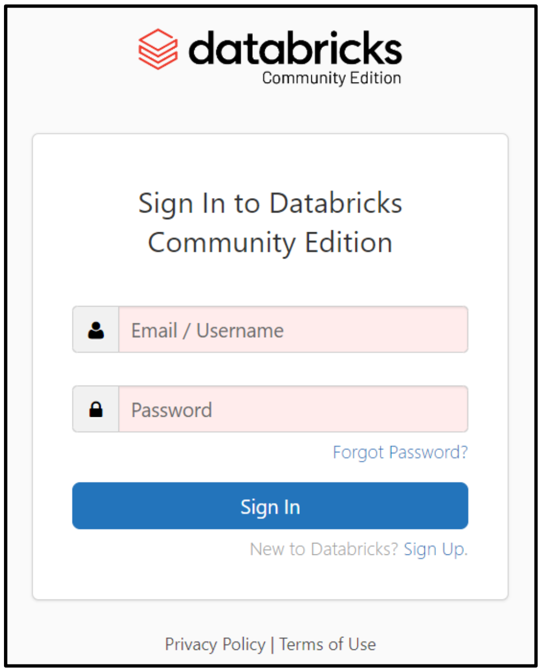
```
:\Users\MMRL-Acer\Documents\Python>jupyter notebook
I 2024-03-26 12:01:47.002 ServerApp] Extension package jupyter_lsp took 0.1055s to import
I 2024-03-26 12:01:47.487 ServerApp] Extension package jupyter_server_terminals took 0.4726s to import
I 2024-03-26 12:01:47.846 ServerApp] jupyter_lsp | extension was successfully linked.
I 2024-03-26 12:01:47.846 ServerApp] jupyter_server_terminals | extension was successfully linked.
I 2024-03-26 12:01:47.862 ServerApp] jupyterlab | extension was successfully linked.
I 2024-03-26 12:01:47.877 ServerApp] notebook | extension was successfully linked.
I 2024-03-26 12:01:47.877 ServerApp] Writing Jupyter server cookie secret to C:\Users\MMRL-Acer\AppData\Roaming\jupyter
\runtime\jupyter_cookie_secret
```

6) Then you can open the notebook and rename the shell as any name and select the python as the type and then you can use it.



❖ Databricks Community Edition

1) Create a new account in Databricks and sign in.



databricks
Community Edition

Sign In to Databricks Community Edition

Email / Username

Password

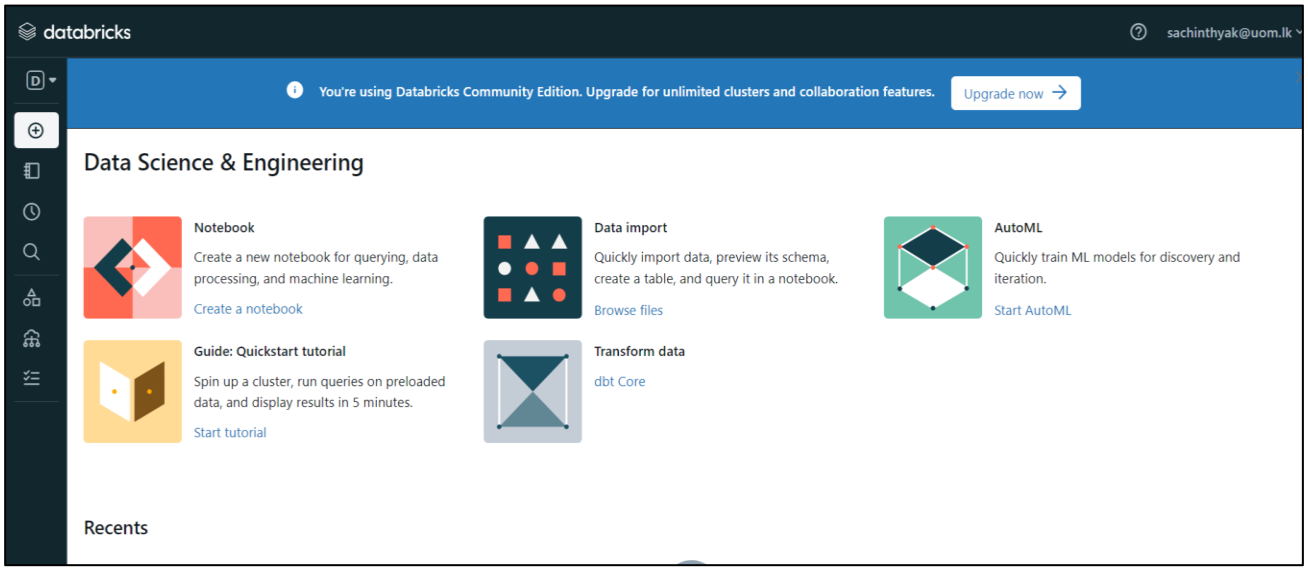
Forgot Password?

Sign In

New to Databricks? Sign Up.

Privacy Policy | Terms of Use

2) Open the Databricks cloud and rename the workspace as any name and select python as the type.



databricks   sachinthyak@uom.lk

You're using Databricks Community Edition. Upgrade for unlimited clusters and collaboration features. Upgrade now →

Data Science & Engineering

Notebook
Create a new notebook for querying, data processing, and machine learning.
Create a notebook

Data import
Quickly import data, preview its schema, create a table, and query it in a notebook.
Browse files

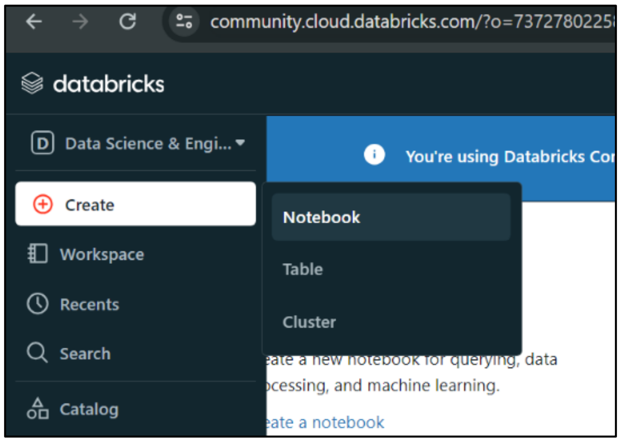
AutoML
Quickly train ML models for discovery and iteration.
Start AutoML

Guide: Quickstart tutorial
Spin up a cluster, run queries on preloaded data, and display results in 5 minutes.
Start tutorial

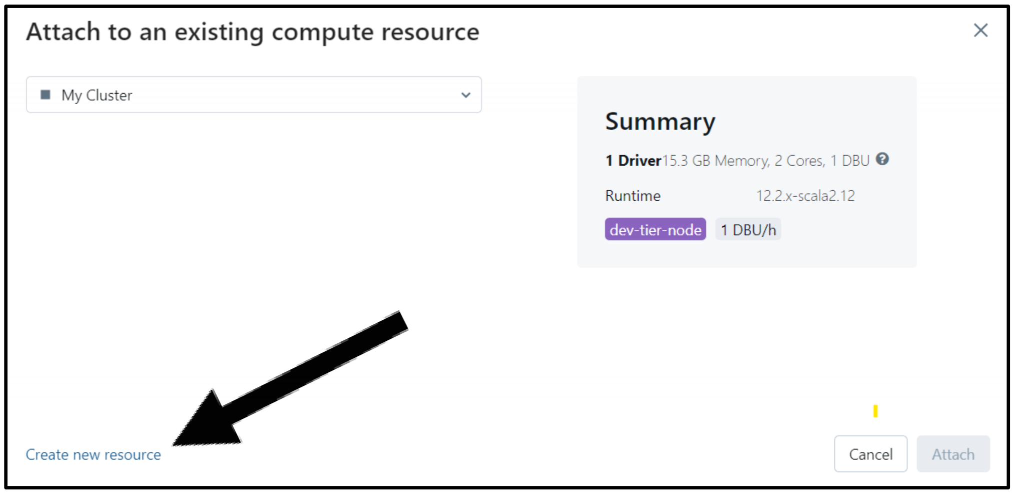
Transform data
dbt Core

Recents

3) Create a new **Notebook** to start coding.


community.cloud.databricks.com/?o=7372780225
databricks
Data Science & Engi...
You're using Databricks Co
Create
Workspace
Recents
Search
Catalog
Notebook
Table
Cluster
ate a new notebook for querying, data
ocessing, and machine learning.
ate a notebook

4) Create a new cluster for the analysis, press **Create new resource** tab.



Attach to an existing compute resource

My Cluster

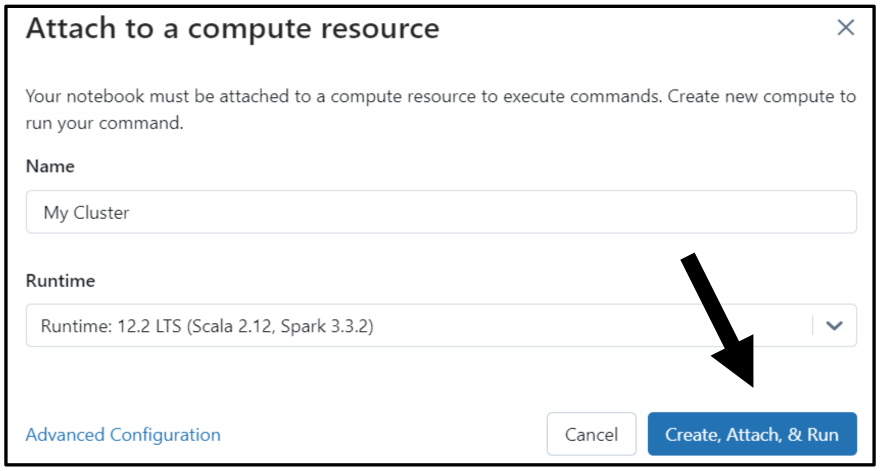
Summary

1 Driver 15.3 GB Memory, 2 Cores, 1 DBU

Runtime 12.2.x-scala2.12

dev-tier-node 1 DBU/h

5) Press **Create, Attach & Run tab** to create the cluster.



Attach to a compute resource

Your notebook must be attached to a compute resource to execute commands. Create new compute to run your command.

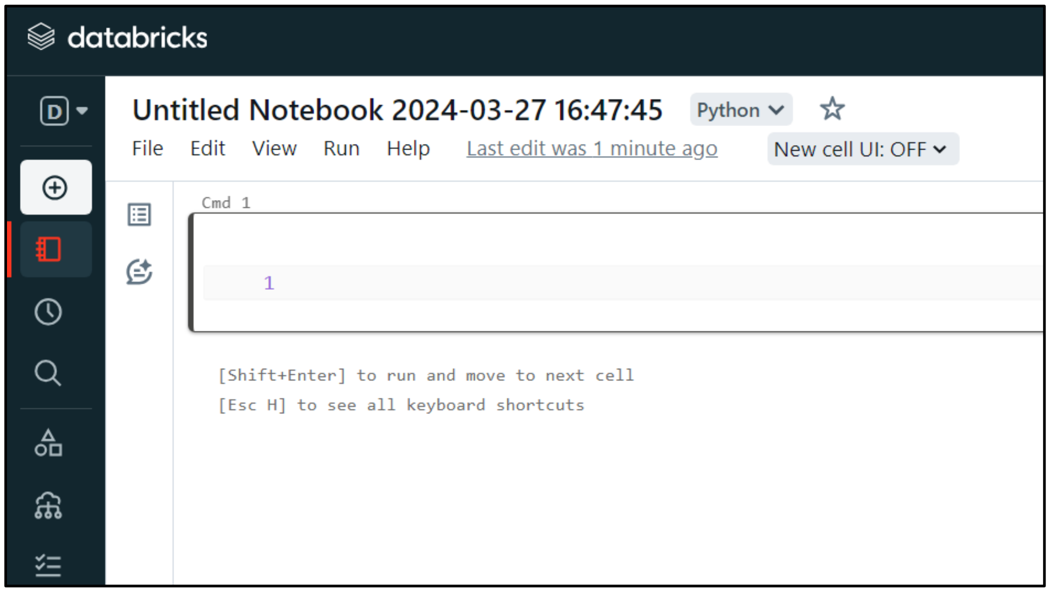
Name

My Cluster

Runtime

Runtime: 12.2 LTS (Scala 2.12, Spark 3.3.2)

6) Finally, it will appear a window as below as same as jupyter notebook then you can use it.


databricks
Untitled Notebook 2024-03-27 16:47:45
Python
File
Edit
View
Run
Help
Last edit was 1 minute ago
New cell UI: OFF
Cmd 1

```
[Shift+Enter] to run and move to next cell
[Esc H] to see all keyboard shortcuts
```

1. Create a PySpark RDD

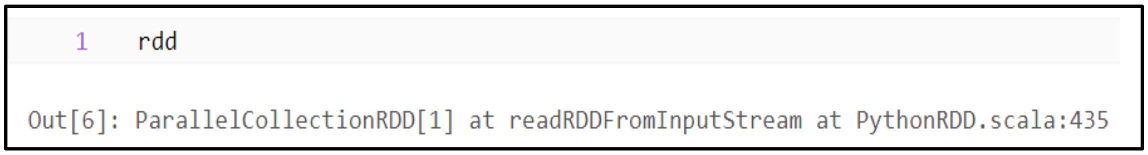
- Import and create a SparkSession:

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.getOrCreate()
```

- Create a rdd:

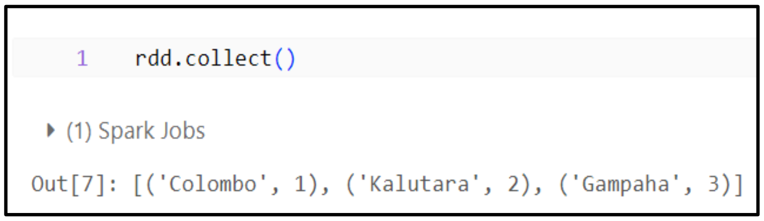
```
rdd = spark.sparkContext.parallelize([("Colombo", 1),
("Kalutara", 2), ("Gampaha", 3)])
```

```
rdd
```



```
rdd

Out[6]: ParallelCollectionRDD[1] at readRDDFromInputStream at PythonRDD.scala:435
```

```
rdd.collect()
```

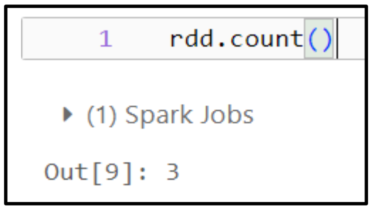


```
rdd.collect()
```

▸ (1) Spark Jobs

```
Out[7]: [('Colombo', 1), ('Kalutara', 2), ('Gampaha', 3)]
```

```
rdd.count()
```



```
rdd.count()
```

▸ (1) Spark Jobs

```
Out[9]: 3
```

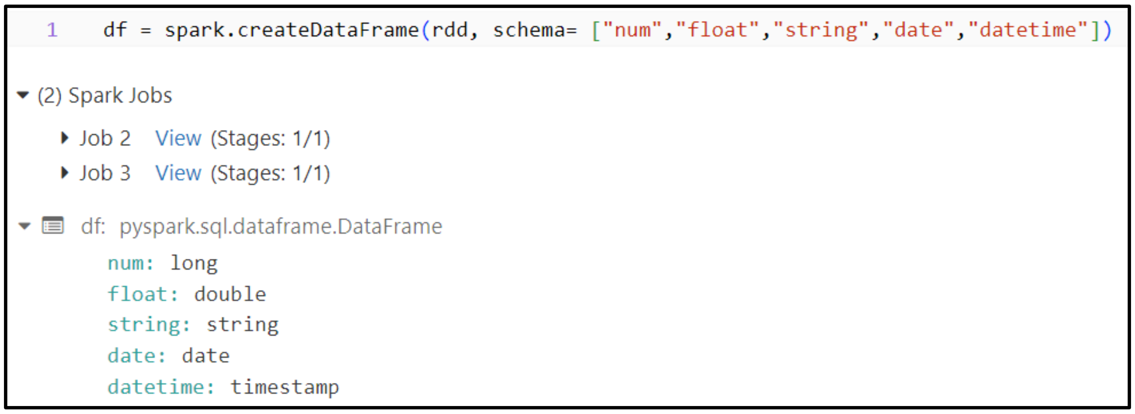
2. Create a PySpark Data Frame from RDD

- Import and create a Spark Session:

```
from pyspark.sql import SparkSession
from datetime import date, datetime
spark = SparkSession.builder.getOrCreate()
```

```
rdd = spark.SparkContext.parallelize([
    (1,1.0,"string1", date(2024,1,1), datetime(2024,1,1,12,0)),
    (2,2.0,"string2", date(2024,2,1), datetime(2024,2,1,12,0)),
    (3,3.0,"string3", date(2024,3,1), datetime(2024,3,1,12,0))
              ])



df=spark.createDataFrame(rdd,schema=["num","float","string","da
te","datetime"])
```

df



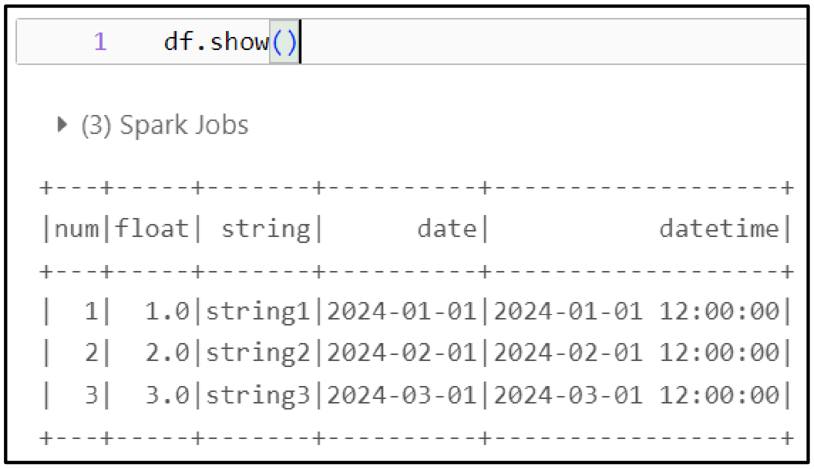
```
df = spark.createDataFrame(rdd, schema= ["num","float","string","date","datetime"])
```

(2) Spark Jobs
Job 2 View (Stages: 1/1)
Job 3 View (Stages: 1/1)

df: pyspark.sql.dataframe.DataFrame

```
num: long
float: double
string: string
date: date
datetime: timestamp
```

df.show()

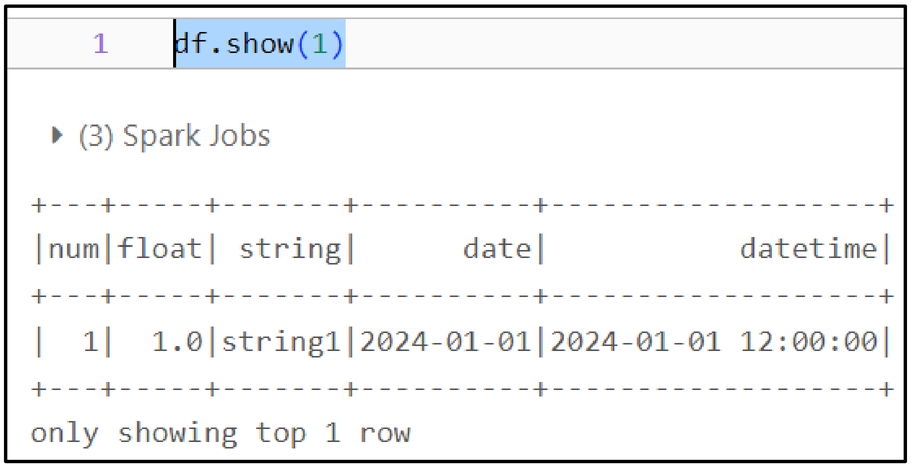


```
df.show()
```

(3) Spark Jobs

```
+---+-----+-------+----------+-------------------+
|num|float| string|      date|           datetime|
+---+-----+-------+----------+-------------------+
|  1|  1.0|string1|2024-01-01|2024-01-01 12:00:00|
|  2|  2.0|string2|2024-02-01|2024-02-01 12:00:00|
|  3|  3.0|string3|2024-03-01|2024-03-01 12:00:00|
+---+-----+-------+----------+-------------------+
```

df.show(1)

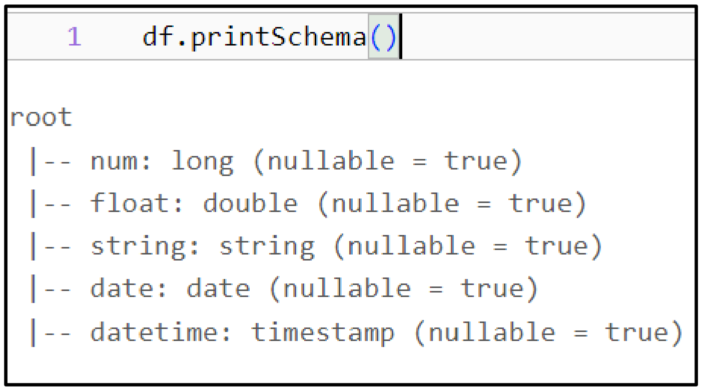


```
df.show(1)
```

(3) Spark Jobs

```
+---+-----+-------+----------+-------------------+
|num|float| string|      date|           datetime|
+---+-----+-------+----------+-------------------+
|  1|  1.0|string1|2024-01-01|2024-01-01 12:00:00|
+---+-----+-------+----------+-------------------+
only showing top 1 row
```

```
df.printSchema()
```



```
df.printSchema()
```

```
root
 |-- num: long (nullable = true)
 |-- float: double (nullable = true)
 |-- string: string (nullable = true)
 |-- date: date (nullable = true)
 |-- datetime: timestamp (nullable = true)
```

3. Create a Spark context in Python

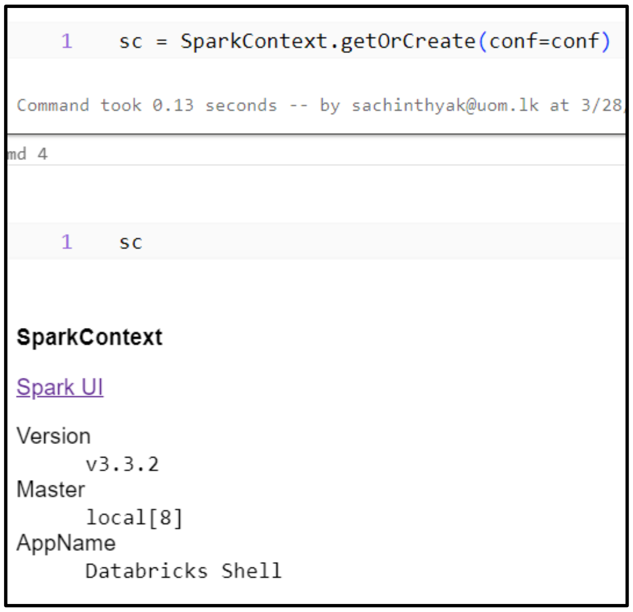
- Import and create a Spark Session, Spark Context and SQL Context:

```
import pyspark
from pyspark import SparkContext
from pyspark.sql import SparkSession
from pyspark.sql import SQLContext
```

- Memory allocation for spark Context

```
conf = pyspark.SparkConf().setMaster("local").setAppName("My
First Spark Practical").setAll([("spark.driver.memory", "40g"),
("spark.executor.memory", "50g")])
```

```
sc = SparkContext.getOrCreate(conf=conf)
```



```
sc = SparkContext.getOrCreate(conf=conf)
```

```
Command took 0.13 seconds -- by sachinthyak@uom.lk at 3/28
```

```
md 4
```

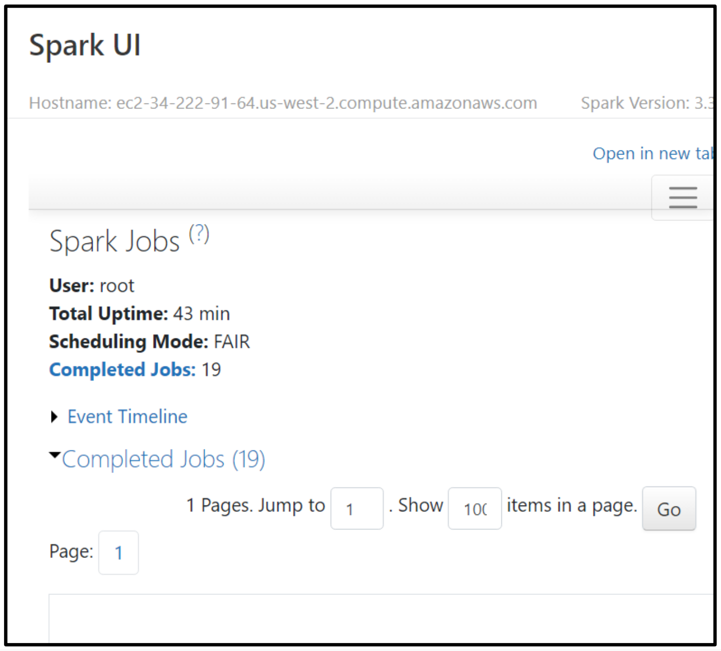
```
sc
```

SparkContext

Spark UI

Version
```
v3.3.2
```
Master
```
local[8]
```
AppName
```
Databricks Shell
```

After creating the spark context, simply click the Spark UI link and then you can move on to the Spark UI Shell


Spark UI

Hostname: ec2-34-222-91-64.us-west-2.compute.amazonaws.com Spark Version: 3.3

Open in new tal

Spark Jobs (?)

User: root
Total Uptime: 43 min
Scheduling Mode: FAIR
Completed Jobs: 19

To stop the spark context, you can type,

```
sc.stop()
```

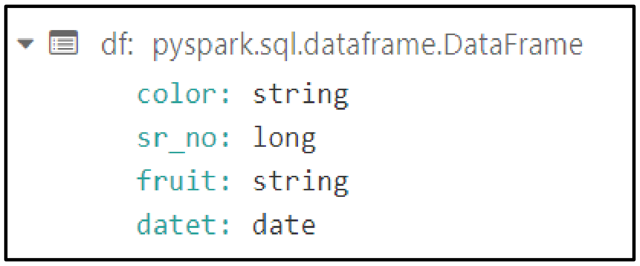
4. Create a PySpark Data Frame

- Import and create a SparkSession:

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.getOrCreate()
from datetime import date
```

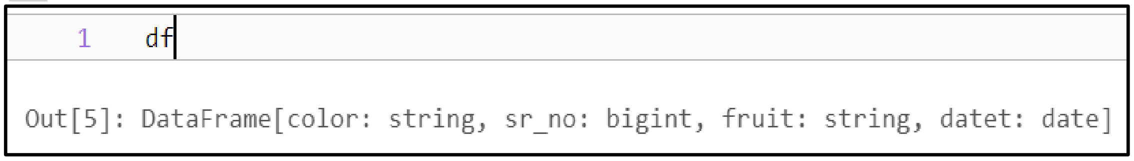
- Create data frame

```
df = spark.createDataFrame([
("Red", 1, "Apple", date(2024,1,1)),
("Black", 2, "Grapes", date(2024,2,1)),
("Yellow", 3, "Banana", date(2024,3,1))],
 schema = "color string, sr_no long, fruit string, datet date")
```



```
df:  pyspark.sql.dataframe.DataFrame
      color:  string
      sr_no:  long
      fruit:  string
      datet:  date
```

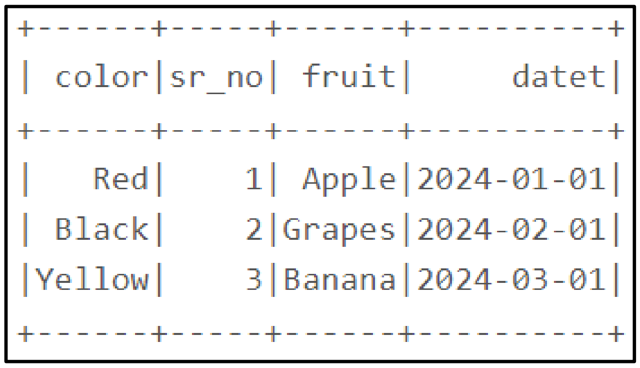
df



```
df
```

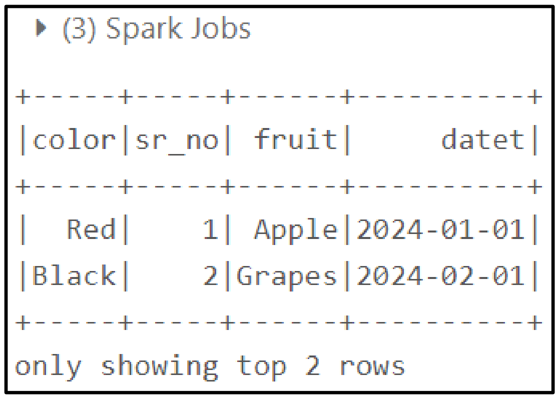
```
Out[5]: DataFrame[color: string, sr_no: bigint, fruit: string, datet: date]
```

df.show



```
+------+-----+------+----------+
| color|sr_no| fruit|     datet|
+------+-----+------+----------+
|   Red|    1| Apple|2024-01-01|
| Black|    2|Grapes|2024-02-01|
|Yellow|    3|Banana|2024-03-01|
+------+-----+------+----------+
```

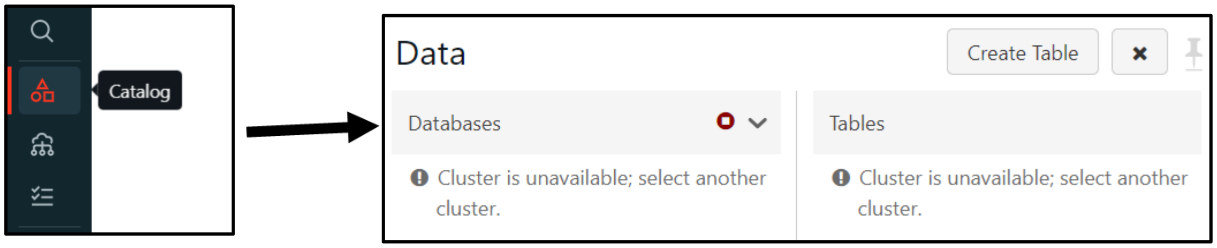
df.show(2)



```
+-----+-----+------+----------+
|color|sr_no| fruit|     datet|
+-----+-----+------+----------+
|  Red|    1| Apple|2024-01-01|
|Black|    2|Grapes|2024-02-01|
+-----+-----+------+----------+
only showing top 2 rows
```

5. Read a CSV file in Pyspark

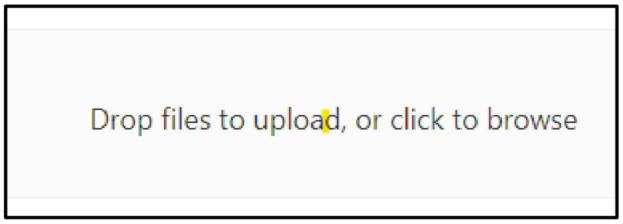
Go to the **Catalog** tab in left side corner and click the button



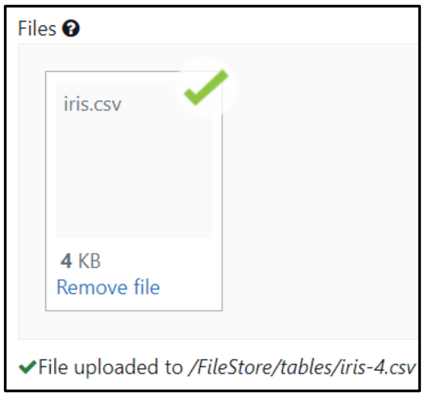
Cluster is unavailable; select another cluster.

Cluster is unavailable; select another cluster.

Press the **Create table** button, and then browse or drop the csv file.



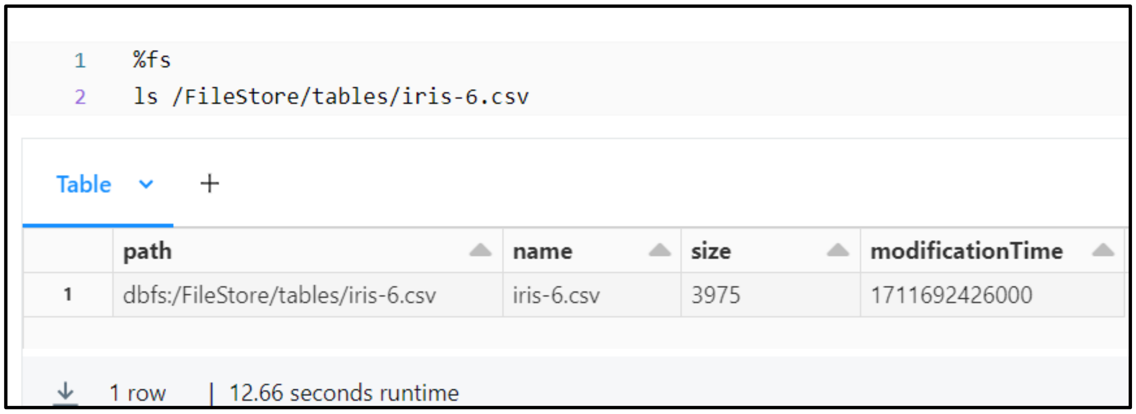
You can see it successfully uploaded.



✓File uploaded to */FileStore/tables/iris-4.csv*

Then press **Create a table in notebook**, it will automatically generate a DBFS system and command prompt area.

```
%fs
ls /FileStore/tables/iris-6.csv
```



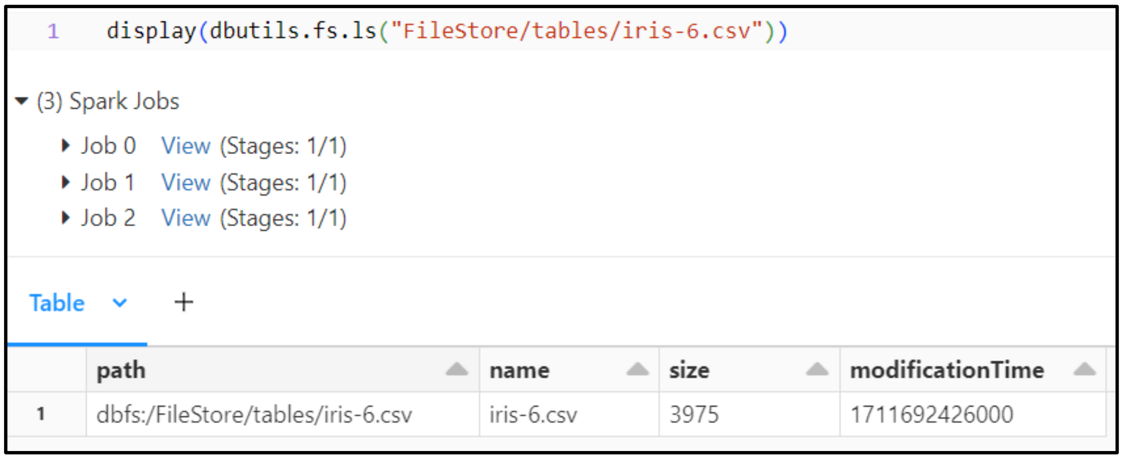
```
%fs
ls /FileStore/tables/iris-6.csv
```

Table

	path	name	size	modificationTime
1	dbfs:/FileStore/tables/iris-6.csv	iris-6.csv	3975	1711692426000

1 row | 12.66 seconds runtime

Now, you can load your csv file into your DBFS system



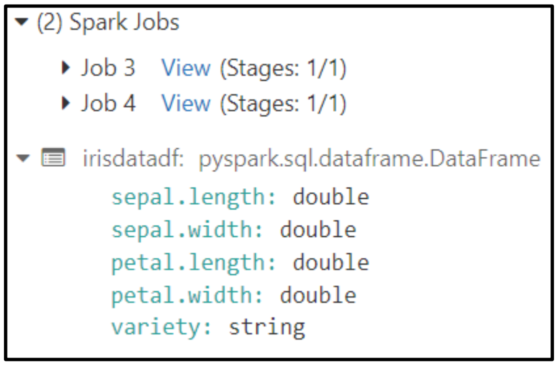
```
display(dbutils.fs.ls("FileStore/tables/iris-6.csv"))
```

(3) Spark Jobs
- Job 0 View (Stages: 1/1)
- Job 1 View (Stages: 1/1)
- Job 2 View (Stages: 1/1)

Table

	path	name	size	modificationTime
1	dbfs:/FileStore/tables/iris-6.csv	iris-6.csv	3975	1711692426000

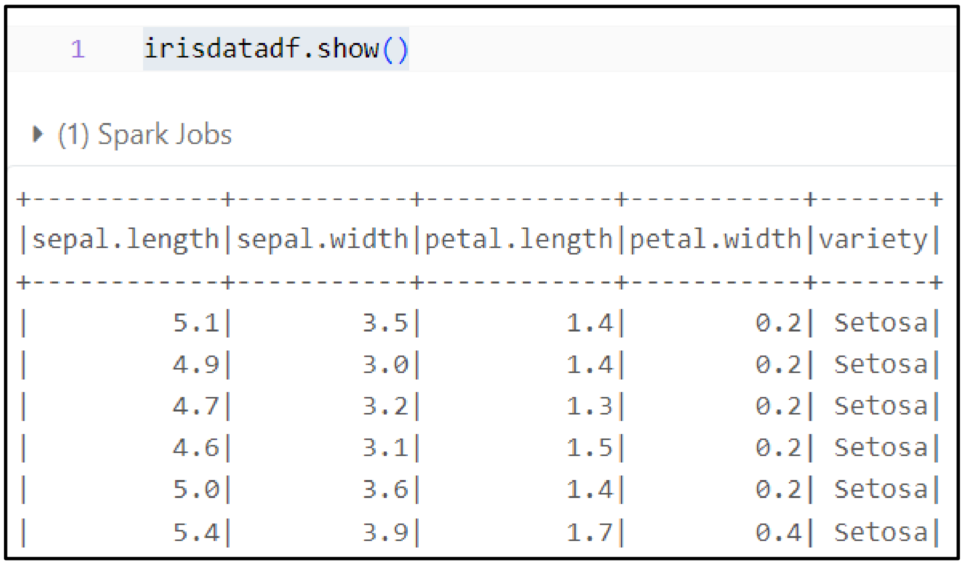
```
irisdatadf =
spark.read.format("csv").option("header","true").option("inferS
chema","true").load("/FileStore/tables/iris-6.csv")
```



(2) Spark Jobs
- Job 3 View (Stages: 1/1)
- Job 4 View (Stages: 1/1)

```
irisdatadf: pyspark.sql.dataframe.DataFrame
    sepal.length: double
    sepal.width: double
    petal.length: double
    petal.width: double
    variety: string
```

```
irisdatadf.show()
```



```
irisdatadf.show()
```

(1) Spark Jobs

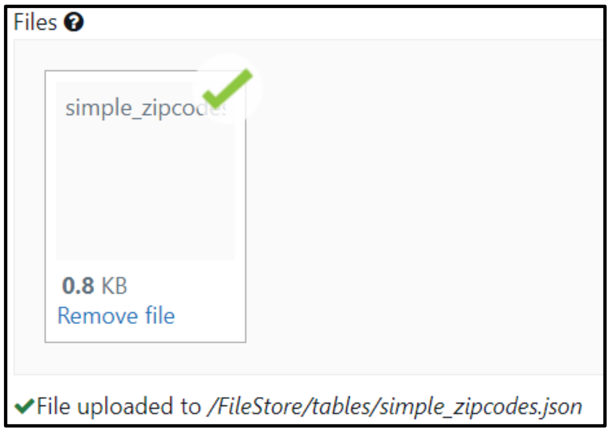
```
+------------+-----------+------------+-----------+-------+
|sepal.length|sepal.width|petal.length|petal.width|variety|
+------------+-----------+------------+-----------+-------+
|         5.1|        3.5|         1.4|        0.2| Setosa|
|         4.9|        3.0|         1.4|        0.2| Setosa|
|         4.7|        3.2|         1.3|        0.2| Setosa|
|         4.6|        3.1|         1.5|        0.2| Setosa|
|         5.0|        3.6|         1.4|        0.2| Setosa|
|         5.4|        3.9|         1.7|        0.4| Setosa|
```

6. Read a JSON file in Pyspark

- Type the code to see the Json methods

```
help(spark.read.json)
```

- Browse or drop the csv file



Files

simple_zipcode

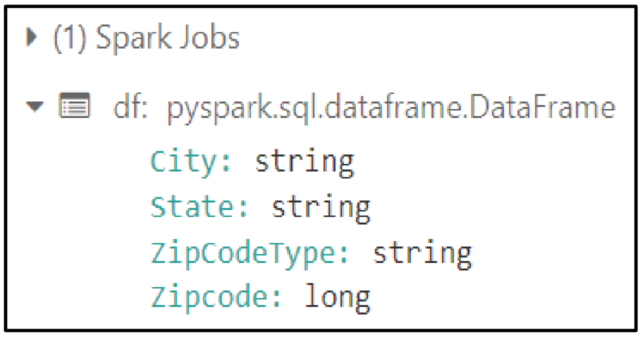
0.8 KB

Remove file

File uploaded to /FileStore/tables/simple_zipcodes.json

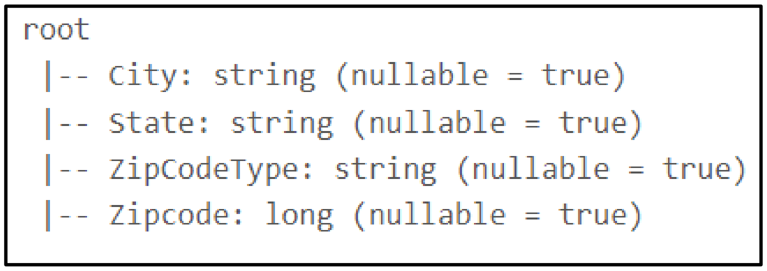
- Create data frame for json file

```
df =
spark.read.json(path="/FileStore/tables/simple_zipcodes.json")
```



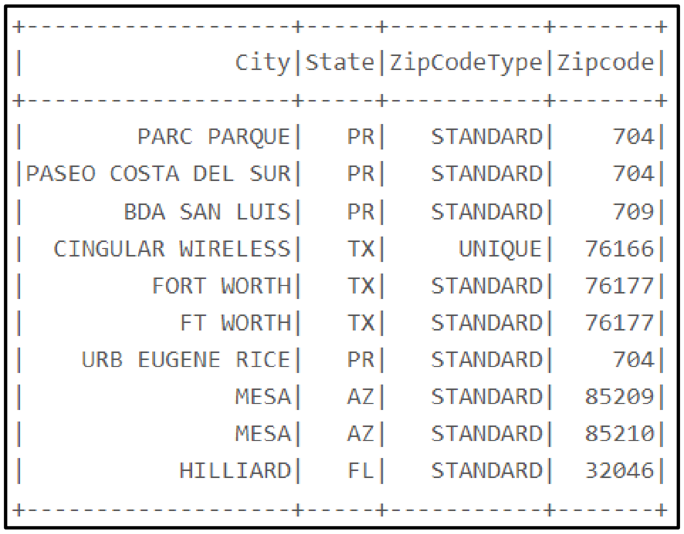
```
▸ (1) Spark Jobs
▾ df: pyspark.sql.dataframe.DataFrame
        City: string
        State: string
        ZipCodeType: string
        Zipcode: long
```

```
df.printSchema()
```



```
root
 |-- City: string (nullable = true)
 |-- State: string (nullable = true)
 |-- ZipCodeType: string (nullable = true)
 |-- Zipcode: long (nullable = true)
```

```
df.show()
```



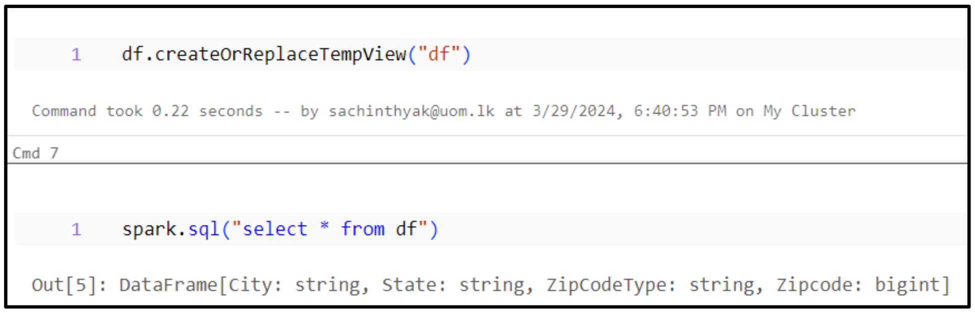
```
+-------------------+-----+-----------+-------+
|               City|State|ZipCodeType|Zipcode|
+-------------------+-----+-----------+-------+
|        PARC PARQUE|   PR|   STANDARD|    704|
|PASEO COSTA DEL SUR|   PR|   STANDARD|    704|
|       BDA SAN LUIS|   PR|   STANDARD|    709|
|  CINGULAR WIRELESS|   TX|     UNIQUE|  76166|
|         FORT WORTH|   TX|   STANDARD|  76177|
|           FT WORTH|   TX|   STANDARD|  76177|
|    URB EUGENE RICE|   PR|   STANDARD|    704|
|               MESA|   AZ|   STANDARD|  85209|
|               MESA|   AZ|   STANDARD|  85210|
|           HILLIARD|   FL|   STANDARD|  32046|
+-------------------+-----+-----------+-------+
```

7. Create Or Replace Temporary View in Data Frame using PySpark

```
spark.sql("select * from df")
```

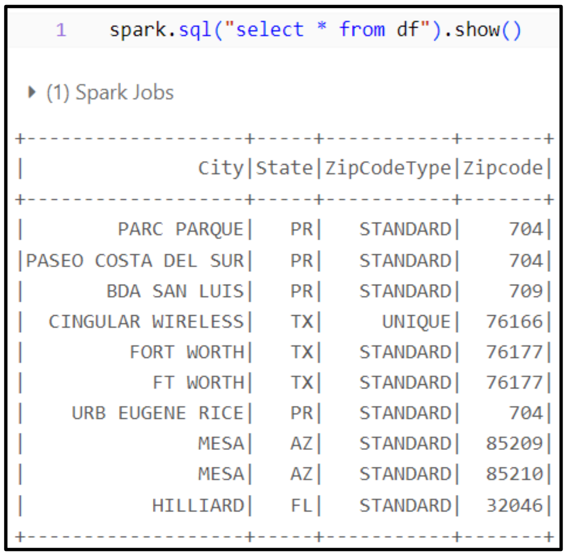
If you just type this code only, it will occur an error, so we have to change the code with this `df.createOrReplaceTempView` function so it will create a temporary view to this SQL session. Once you create this, you will be able to execute any SQL query on top of your data frame inside the spark session.

```
df.createOrReplaceTempView("df")
spark.sql("select * from df")
```



```
df.createOrReplaceTempView("df")

Command took 0.22 seconds -- by sachinthyak@uom.lk at 3/29/2024, 6:40:53 PM on My Cluster
Cmd 7

spark.sql("select * from df")

Out[5]: DataFrame[City: string, State: string, ZipCodeType: string, Zipcode: bigint]
```

```
spark.sql("select * from df").show()
```



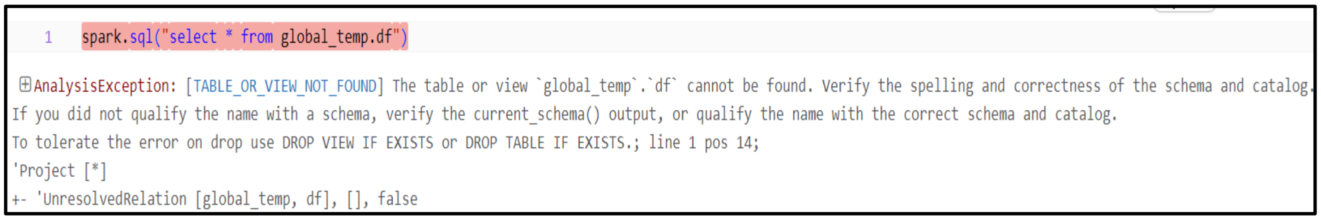
```
spark.sql("select * from df").show()
```

(1) Spark Jobs

```
+-------------------+-----+-----------+-------+
|               City|State|ZipCodeType|Zipcode|
+-------------------+-----+-----------+-------+
|        PARC PARQUE|   PR|   STANDARD|    704|
|PASEO COSTA DEL SUR|   PR|   STANDARD|    704|
|       BDA SAN LUIS|   PR|   STANDARD|    709|
|  CINGULAR WIRELESS|   TX|     UNIQUE|  76166|
|         FORT WORTH|   TX|   STANDARD|  76177|
|           FT WORTH|   TX|   STANDARD|  76177|
|    URB EUGENE RICE|   PR|   STANDARD|    704|
|               MESA|   AZ|   STANDARD|  85209|
|               MESA|   AZ|   STANDARD|  85210|
|           HILLIARD|   FL|   STANDARD|  32046|
+-------------------+-----+-----------+-------+
```

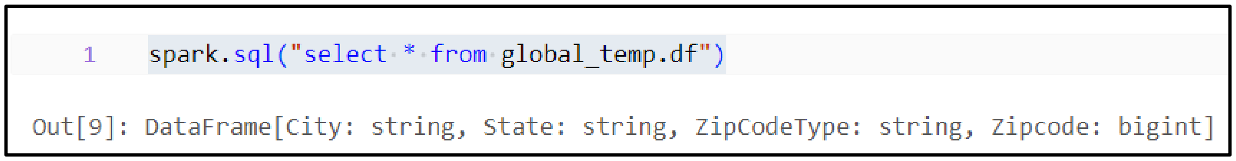
8. Create a Global Temporary View in Data Frame using PySpark

If you just type this select all query only, it will occur an error as same as the first temporary view, so we have to change the code with this df.createGlobalTempView function so it will create a temporary view to this SQL session. Once you create this, you will be able to execute any SQL query on top of your data frame inside the spark application.



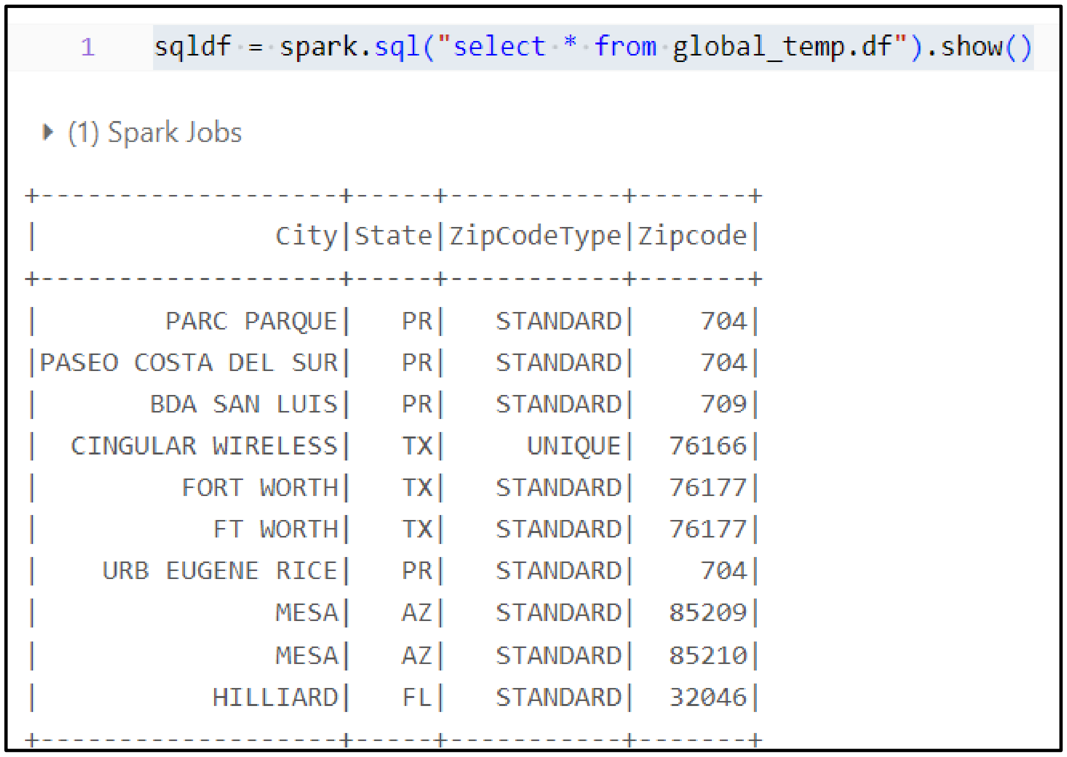
```
spark.sql("select * from global_temp.df")

AnalysisException: [TABLE_OR_VIEW_NOT_FOUND] The table or view `global_temp`.`df` cannot be found. Verify the spelling and correctness of the schema and catalog.
If you did not qualify the name with a schema, verify the current_schema() output, or qualify the name with the correct schema and catalog.
To tolerate the error on drop use DROP VIEW IF EXISTS or DROP TABLE IF EXISTS.; line 1 pos 14;
'Project [*]
+- 'UnresolvedRelation [global_temp, df], [], false
```

```
df.createGlobalTempView("df")
spark.sql("select * from global_temp.df")
```



```
spark.sql("select * from global_temp.df")

Out[9]: DataFrame[City: string, State: string, ZipCodeType: string, Zipcode: bigint]
```

```
sqldf = spark.sql("select * from global_temp.df").show()
```

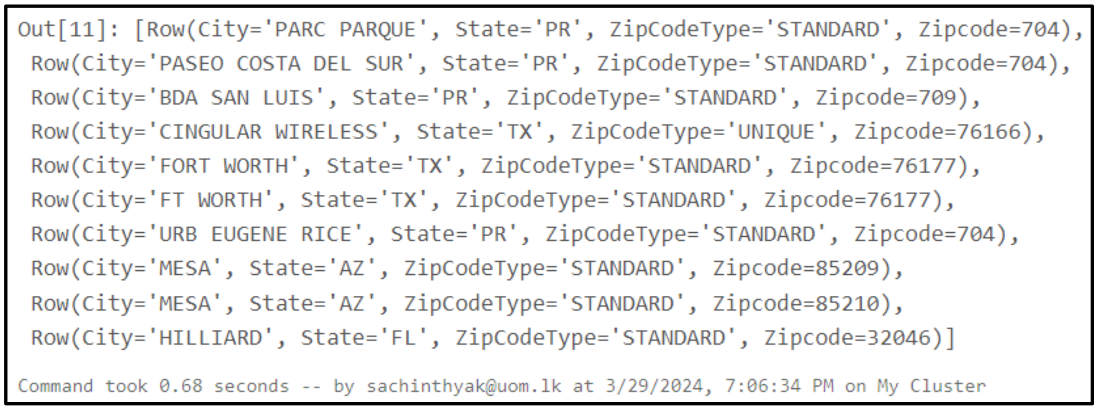


```
sqldf = spark.sql("select * from global_temp.df").show()

▸ (1) Spark Jobs

+-------------------+-----+-----------+-------+
|               City|State|ZipCodeType|Zipcode|
+-------------------+-----+-----------+-------+
|        PARC PARQUE|   PR|   STANDARD|    704|
|PASEO COSTA DEL SUR|   PR|   STANDARD|    704|
|       BDA SAN LUIS|   PR|   STANDARD|    709|
|  CINGULAR WIRELESS|   TX|     UNIQUE|  76166|
|         FORT WORTH|   TX|   STANDARD|  76177|
|           FT WORTH|   TX|   STANDARD|  76177|
|    URB EUGENE RICE|   PR|   STANDARD|    704|
|               MESA|   AZ|   STANDARD|  85209|
|               MESA|   AZ|   STANDARD|  85210|
|           HILLIARD|   FL|   STANDARD|  32046|
+-------------------+-----+-----------+-------+
```

9. PySpark Data Frame Functions

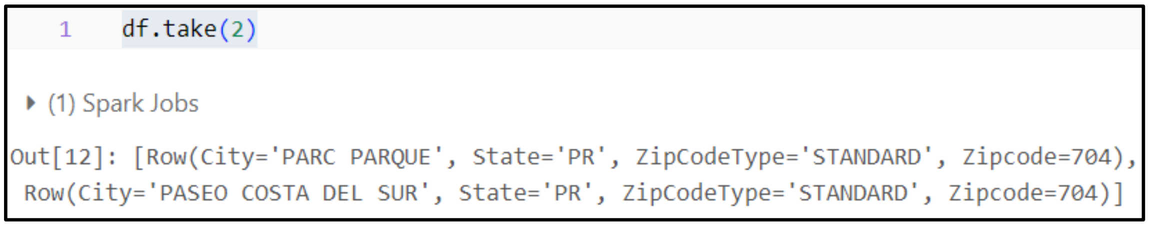
```
  i) collect()
 ii) take()
iii) count()
 iv) select()
  v) filter()
 vi) sort()
vii) describe()
```

```
df.collect()
```



```
Out[11]: [Row(City='PARC PARQUE', State='PR', ZipCodeType='STANDARD', Zipcode=704),
 Row(City='PASEO COSTA DEL SUR', State='PR', ZipCodeType='STANDARD', Zipcode=704),
 Row(City='BDA SAN LUIS', State='PR', ZipCodeType='STANDARD', Zipcode=709),
 Row(City='CINGULAR WIRELESS', State='TX', ZipCodeType='UNIQUE', Zipcode=76166),
 Row(City='FORT WORTH', State='TX', ZipCodeType='STANDARD', Zipcode=76177),
 Row(City='FT WORTH', State='TX', ZipCodeType='STANDARD', Zipcode=76177),
 Row(City='URB EUGENE RICE', State='PR', ZipCodeType='STANDARD', Zipcode=704),
 Row(City='MESA', State='AZ', ZipCodeType='STANDARD', Zipcode=85209),
 Row(City='MESA', State='AZ', ZipCodeType='STANDARD', Zipcode=85210),
 Row(City='HILLIARD', State='FL', ZipCodeType='STANDARD', Zipcode=32046)]

Command took 0.68 seconds -- by sachinthyak@uom.lk at 3/29/2024, 7:06:34 PM on My Cluster
```

```
df.take(2)
```

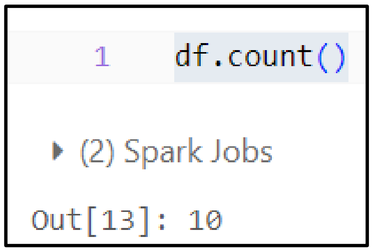


```
df.take(2)
```

▸ (1) Spark Jobs

```
Out[12]: [Row(City='PARC PARQUE', State='PR', ZipCodeType='STANDARD', Zipcode=704),
 Row(City='PASEO COSTA DEL SUR', State='PR', ZipCodeType='STANDARD', Zipcode=704)]
```

```
df.count()
```

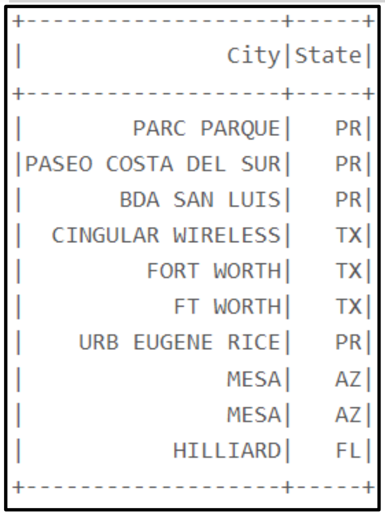


```
df.count()
```

▸ (2) Spark Jobs

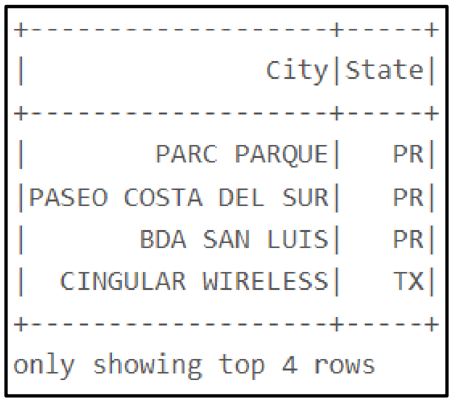
```
Out[13]: 10
```

```
df.select("City", "State").show()
```



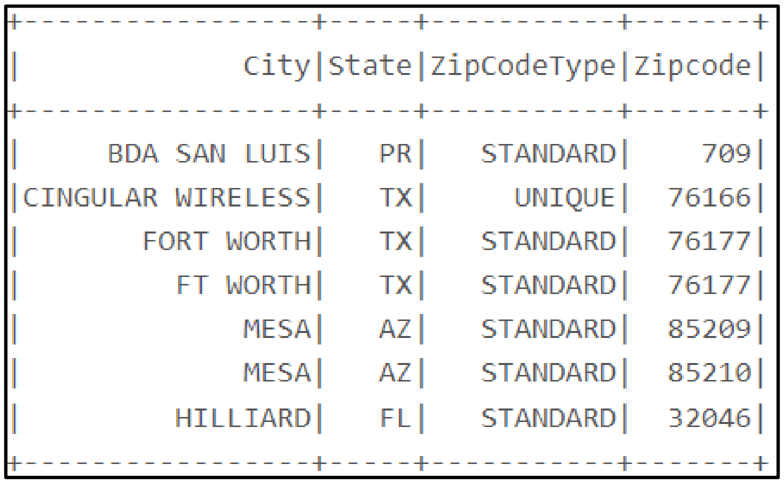
```
+------------------+-----+
|              City|State|
+------------------+-----+
|       PARC PARQUE|   PR|
|PASEO COSTA DEL SUR|   PR|
|      BDA SAN LUIS|   PR|
|  CINGULAR WIRELESS|   TX|
|        FORT WORTH|   TX|
|          FT WORTH|   TX|
|    URB EUGENE RICE|   PR|
|              MESA|   AZ|
|              MESA|   AZ|
|          HILLIARD|   FL|
+------------------+-----+
```

```
df.select("City", "State").show(4)
```



```
+------------------+-----+
|              City|State|
+------------------+-----+
|       PARC PARQUE|   PR|
|PASEO COSTA DEL SUR|   PR|
|      BDA SAN LUIS|   PR|
|  CINGULAR WIRELESS|   TX|
+------------------+-----+
only showing top 4 rows
```

```
df.show()
df.filter(df["Zipcode"]>704).show()
```



```
+------------------+-----+-----------+-------+
|              City|State|ZipCodeType|Zipcode|
+------------------+-----+-----------+-------+
|      BDA SAN LUIS|   PR|   STANDARD|    709|
|CINGULAR WIRELESS|   TX|     UNIQUE|  76166|
|        FORT WORTH|   TX|   STANDARD|  76177|
|          FT WORTH|   TX|   STANDARD|  76177|
|              MESA|   AZ|   STANDARD|  85209|
|              MESA|   AZ|   STANDARD|  85210|
|          HILLIARD|   FL|   STANDARD|  32046|
+------------------+-----+-----------+-------+
df.select("City").filter("City like 'M%' ").show()
```

```
+----+
|City|
+----+
|MESA|
|MESA|
+----+
```

```
df.select("City").filter("City like 'F%' ").show()
```

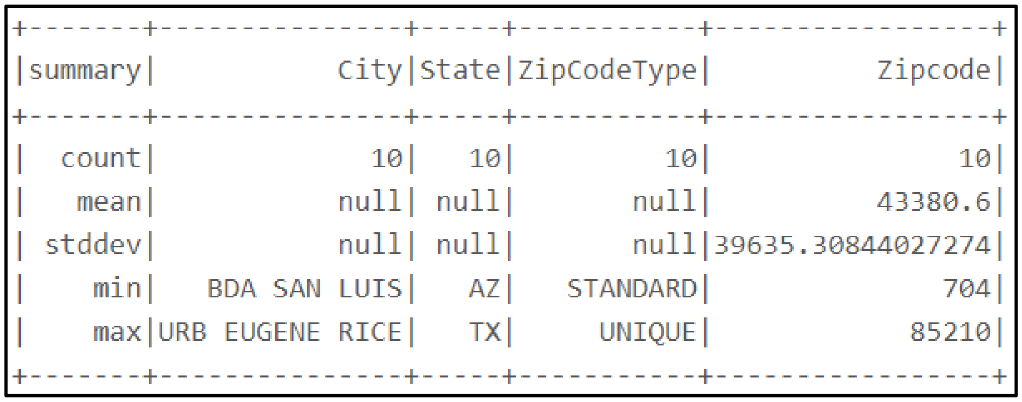
```
+----------+
|      City|
+----------+
|FORT WORTH|
|  FT WORTH|
+----------+
```

```
df.sort("State").show()
```

```
+-----------------+-----+-----------+-------+
|             City|State|ZipCodeType|Zipcode|
+-----------------+-----+-----------+-------+
|             MESA|   AZ|   STANDARD|  85209|
|             MESA|   AZ|   STANDARD|  85210|
|         HILLIARD|   FL|   STANDARD|  32046|
|      PARC PARQUE|   PR|   STANDARD|    704|
|PASEO COSTA DEL SUR|   PR|   STANDARD|    704|
|     BDA SAN LUIS|   PR|   STANDARD|    709|
|  URB EUGENE RICE|   PR|   STANDARD|    704|
| CINGULAR WIRELESS|   TX|     UNIQUE|  76166|
|       FORT WORTH|   TX|   STANDARD|  76177|
|         FT WORTH|   TX|   STANDARD|  76177|
+-----------------+-----+-----------+-------+
```

```
df.sort("City").show()
```

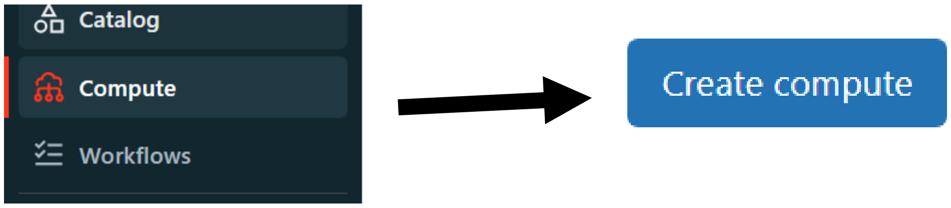
```
+-------------------+-----+-----------+-------+
|               City|State|ZipCodeType|Zipcode|
+-------------------+-----+-----------+-------+
|       BDA SAN LUIS|   PR|   STANDARD|    709|
|  CINGULAR WIRELESS|   TX|     UNIQUE|  76166|
|         FORT WORTH|   TX|   STANDARD|  76177|
|           FT WORTH|   TX|   STANDARD|  76177|
|           HILLIARD|   FL|   STANDARD|  32046|
|               MESA|   AZ|   STANDARD|  85209|
|               MESA|   AZ|   STANDARD|  85210|
|        PARC PARQUE|   PR|   STANDARD|    704|
|PASEO COSTA DEL SUR|   PR|   STANDARD|    704|
|    URB EUGENE RICE|   PR|   STANDARD|    704|
+-------------------+-----+-----------+-------+
df.describe().show()
```



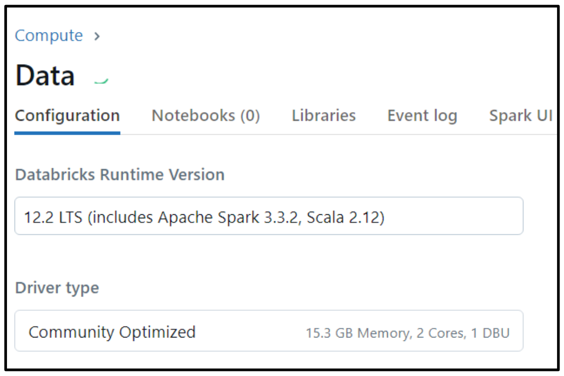
```
+-------+--------------+-----+-----------+-----------------+
|summary|          City|State|ZipCodeType|          Zipcode|
+-------+--------------+-----+-----------+-----------------+
|  count|            10|   10|         10|               10|
|   mean|          null| null|       null|          43380.6|
| stddev|          null| null|       null|39635.30844027274|
|    min|  BDA SAN LUIS|   AZ|   STANDARD|              704|
|    max|URB EUGENE RICE|   TX|     UNIQUE|            85210|
+-------+--------------+-----+-----------+-----------------+
```

10.PySpark SQL Database

1) To Create a Database, first go to the Compute tab and create a new cluster.



2) Give a name for the cluster
   Ex: Data.



Compute ›

Data

Configuration | Notebooks (0) | Libraries | Event log | Spark UI

Databricks Runtime Version

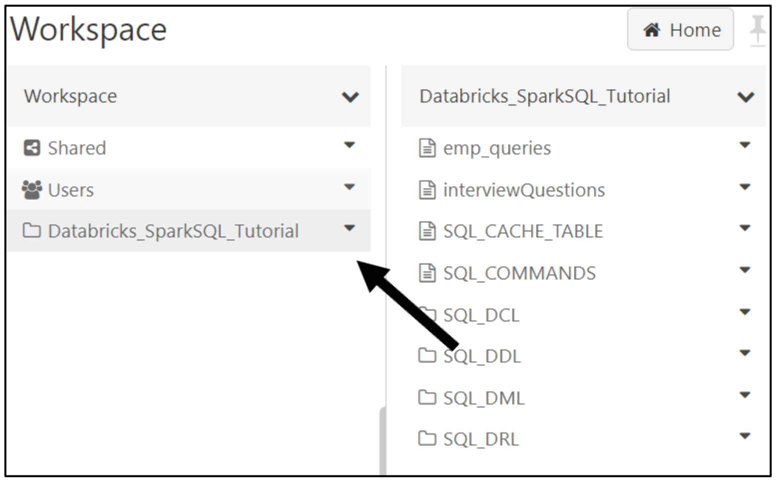
12.2 LTS (includes Apache Spark 3.3.2, Scala 2.12)

Driver type

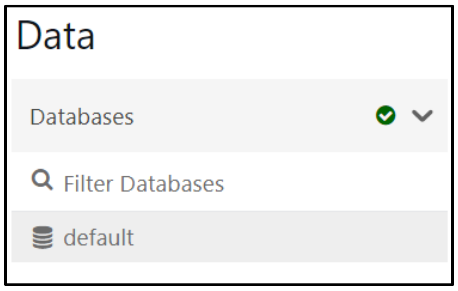
Community Optimized 15.3 GB Memory, 2 Cores, 1 DBU

3) Download this .dbc file and upload it to the workspace that you are working on.

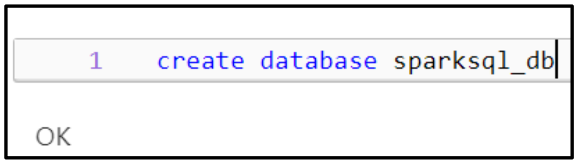
   https://github.com/raveendratal/PysparkRaveendra/blob/master/Databricks_SparkSQL_Tutorial.dbc



4) You can see the databases inside the cluster.



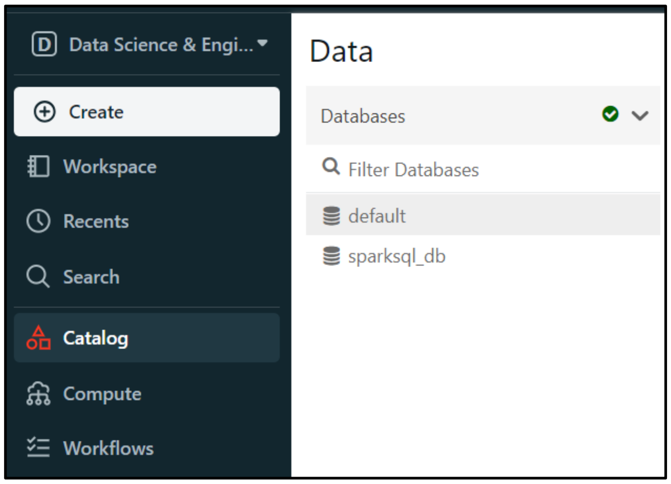
5) Create a database command.



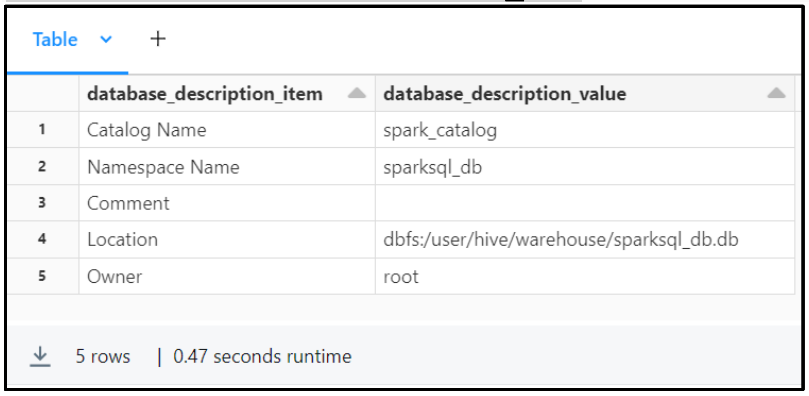
```
create database sparksql_db
```

```
OK
```

Then you can see the successfully created database inside the cluster.



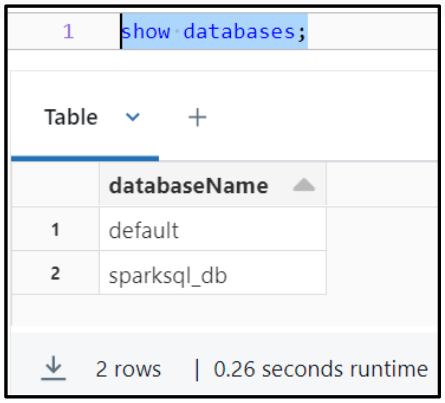
```
describe database sparksql_db;
```



	database_description_item	database_description_value
1	Catalog Name	spark_catalog
2	Namespace Name	sparksql_db
3	Comment	
4	Location	dbfs:/user/hive/warehouse/sparksql_db.db
5	Owner	root

5 rows | 0.47 seconds runtime

```
show databases;
```

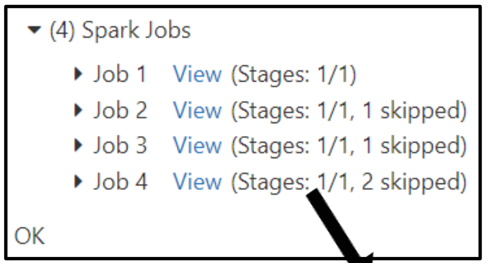


```
show databases;
```

	databaseName
1	default
2	sparksql_db

2 rows | 0.26 seconds runtime

```
create table sparksql_db.customer(id int, name string)
```



```
(4) Spark Jobs
    Job 1  View (Stages: 1/1)
    Job 2  View (Stages: 1/1, 1 skipped)
    Job 3  View (Stages: 1/1, 1 skipped)
    Job 4  View (Stages: 1/1, 2 skipped)
OK
```

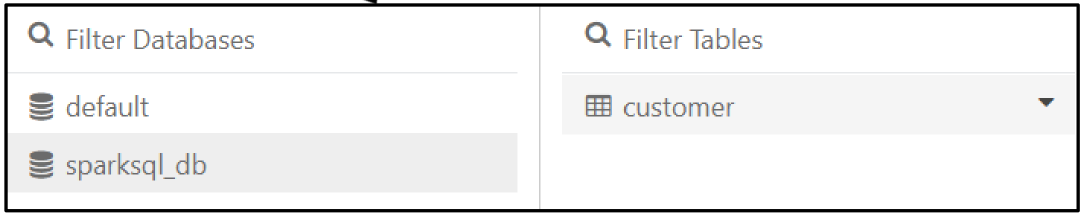
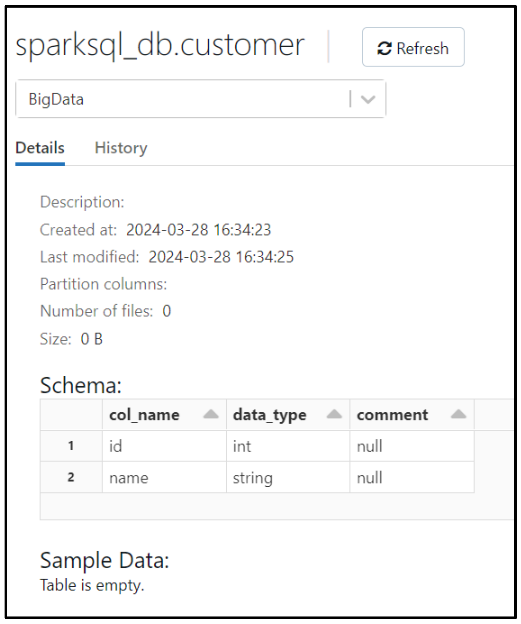


Table Details



sparksql_db.customer

Refresh

BigData

Details History

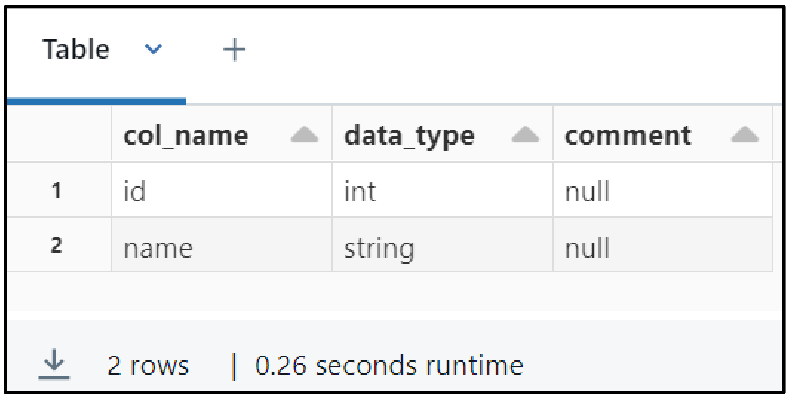
Description:
Created at: 2024-03-28 16:34:23
Last modified: 2024-03-28 16:34:25
Partition columns:
Number of files: 0
Size: 0 B

Schema:

	col_name	data_type	comment
1	id	int	null
2	name	string	null

Sample Data:
Table is empty.

```
describe sparksql_db.customer;
```

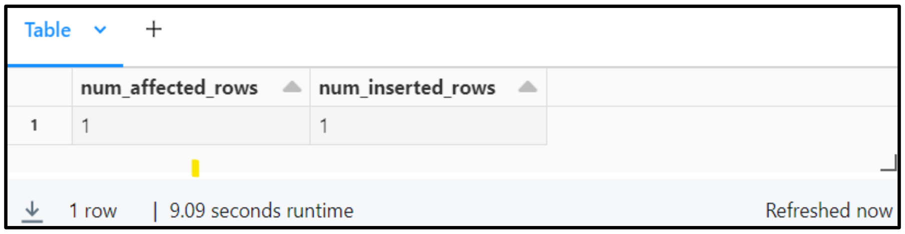


Table

	col_name	data_type	comment
1	id	int	null
2	name	string	null

2 rows | 0.26 seconds runtime

```
insert into sparksql_db.customer values(1, 'Ravi')
```

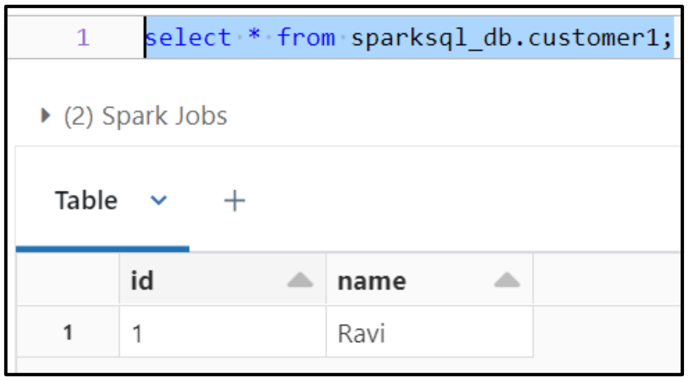


Table

	num_affected_rows	num_inserted_rows
1	1	1

1 row | 9.09 seconds runtime Refreshed now

```
select * from sparksql_db.customer;
```



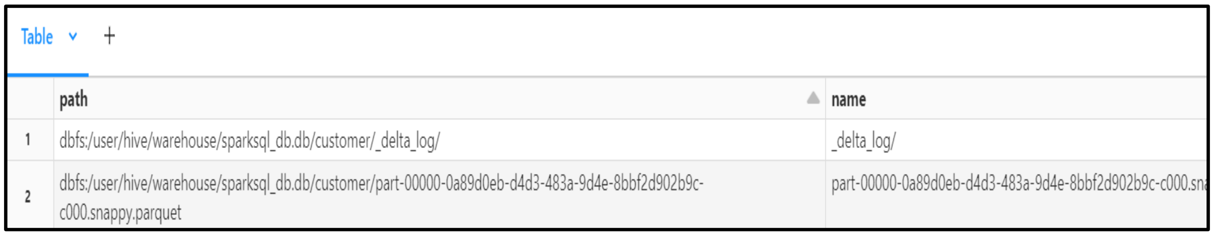
```
select * from sparksql_db.customer1;
```

(2) Spark Jobs

Table

	id	name
1	1	Ravi

```
%fs ls /user/hive/warehouse/sparksql_db.db/customer
```

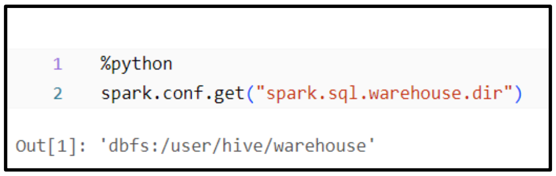


Table

	path	name
1	dbfs:/user/hive/warehouse/sparksql_db.db/customer/_delta_log/	_delta_log/
2	dbfs:/user/hive/warehouse/sparksql_db.db/customer/part-00000-0a89d0eb-d4d3-483a-9d4e-8bbf2d902b9c-c000.snappy.parquet	part-00000-0a89d0eb-d4d3-483a-9d4e-8bbf2d902b9c-c000.sna

```
%python
spark.conf.get("spark.sql.warehouse.dir")
```

Default warehouse location.



```
%python
spark.conf.get("spark.sql.warehouse.dir")

Out[1]: 'dbfs:/user/hive/warehouse'
```

Conclusion

In conclusion, Spark SQL is a module of Spark that analyses the structured data. It provides Scalability, it ensures high compatibility of the system. Thus, it provides the most natural way to express the Structured Data.