Akilesh K

k.akilesh123@gmail.com

Data engineering - Batch 1

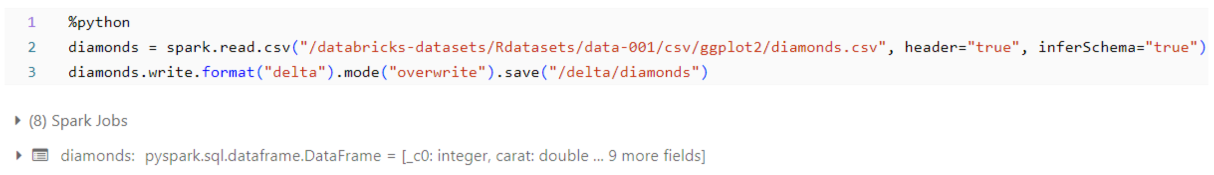
Date: 05-02-24

DAY 12 - PYSPARK, DATABRICKS INTRO

Features of Pyspark

- In-memory computation
- Distributed processing using parallelize
- Can be used with many cluster managers
- Fault-tolerant
- Immutable Lazy evaluation Cache & persistence

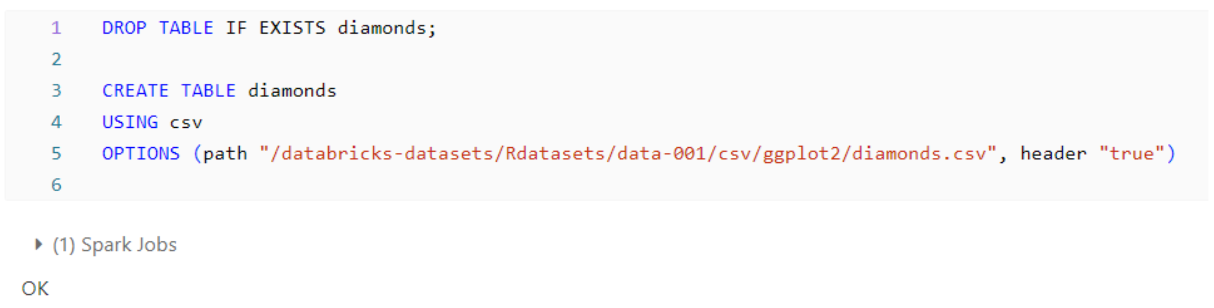
Dataset is loaded into databricks



```
%python
diamonds = spark.read.csv("/databricks-datasets/Rdatasets/data-001/csv/ggplot2/diamonds.csv", header="true", inferSchema="true")
diamonds.write.format("delta").mode("overwrite").save("/delta/diamonds")
```

```
(8) Spark Jobs
diamonds:  pyspark.sql.dataframe.DataFrame = [_c0: integer, carat: double ... 9 more fields]
```

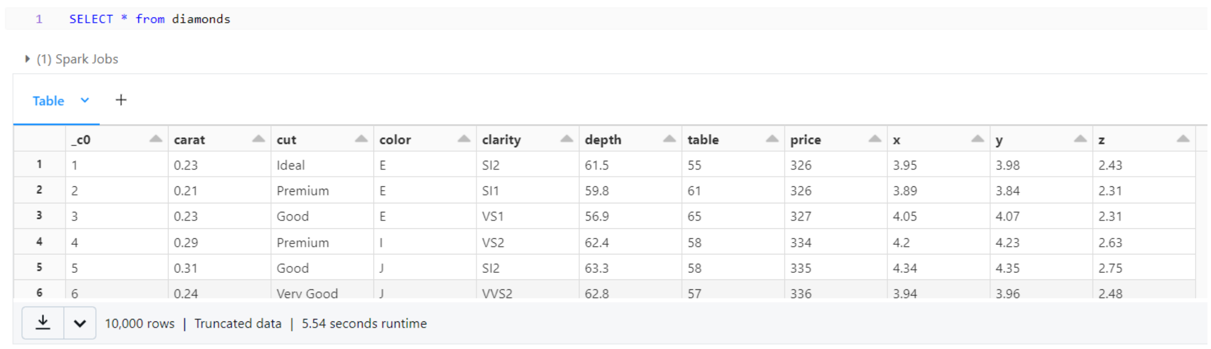
Table is created named diamonds



```
DROP TABLE IF EXISTS diamonds;

CREATE TABLE diamonds
USING csv
OPTIONS (path "/databricks-datasets/Rdatasets/data-001/csv/ggplot2/diamonds.csv", header "true")

```

```
(1) Spark Jobs
OK
```

Display table



```sql
SELECT * from diamonds
```

(1) Spark Jobs

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

10,000 rows | Truncated data | 5.54 seconds runtime

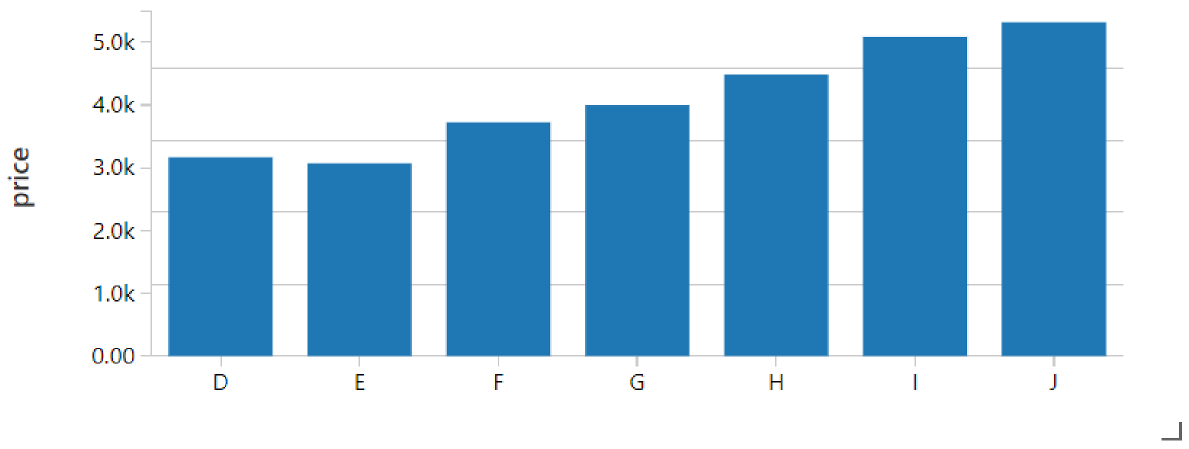
Display using visualization

```sql
SELECT color, avg(price) AS price FROM diamonds GROUP BY color ORDER BY color
```

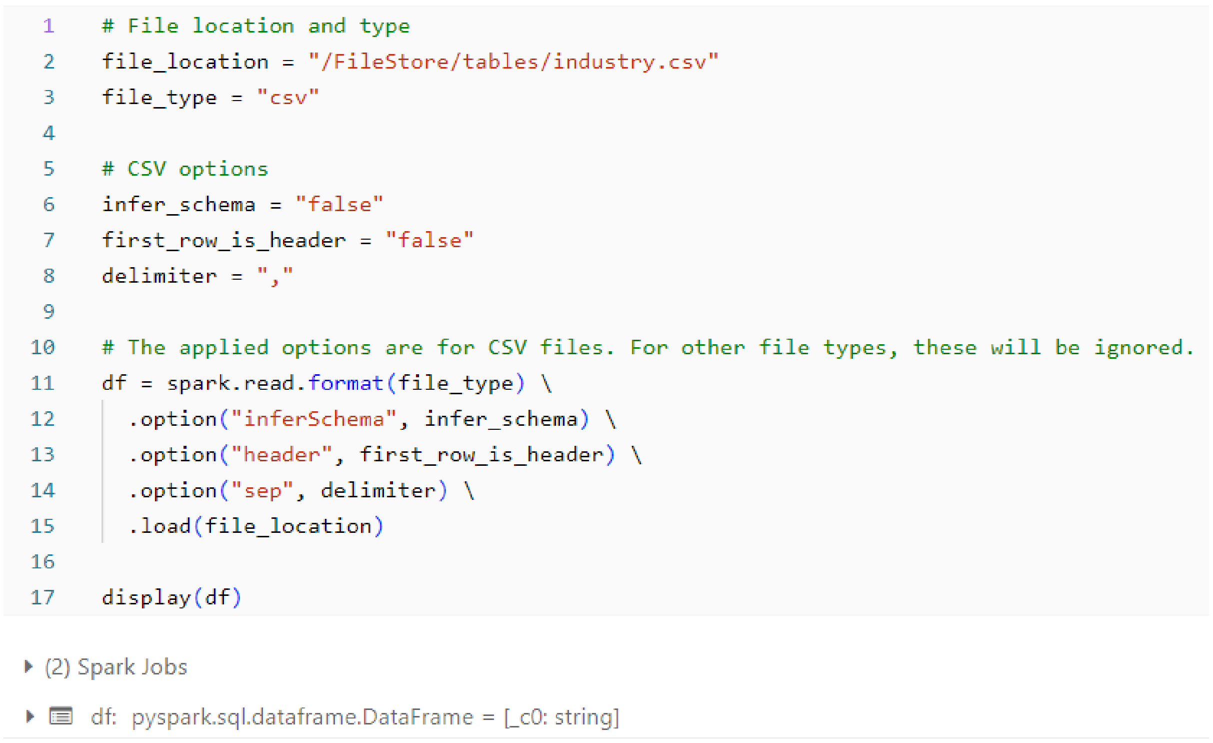
(2) Spark Jobs

Visualization +

(2) Spark Jobs



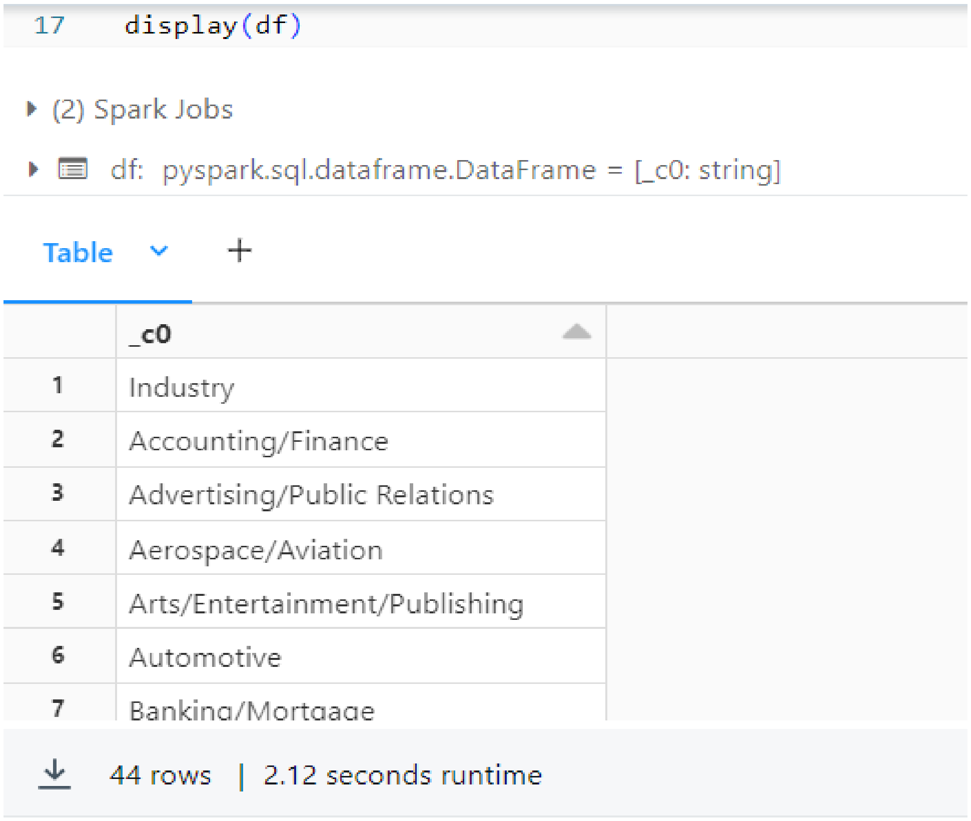
Creating a table using notebook



```python
# File location and type
file_location = "/FileStore/tables/industry.csv"
file_type = "csv"

# CSV options
infer_schema = "false"
first_row_is_header = "false"
delimiter = ","

# The applied options are for CSV files. For other file types, these will be ignored.
df = spark.read.format(file_type) \
  .option("inferSchema", infer_schema) \
  .option("header", first_row_is_header) \
  .option("sep", delimiter) \
  .load(file_location)

display(df)
```

```
(2) Spark Jobs
df:  pyspark.sql.dataframe.DataFrame = [_c0: string]
```

Displaying the table as a dataframe



```python
display(df)
```

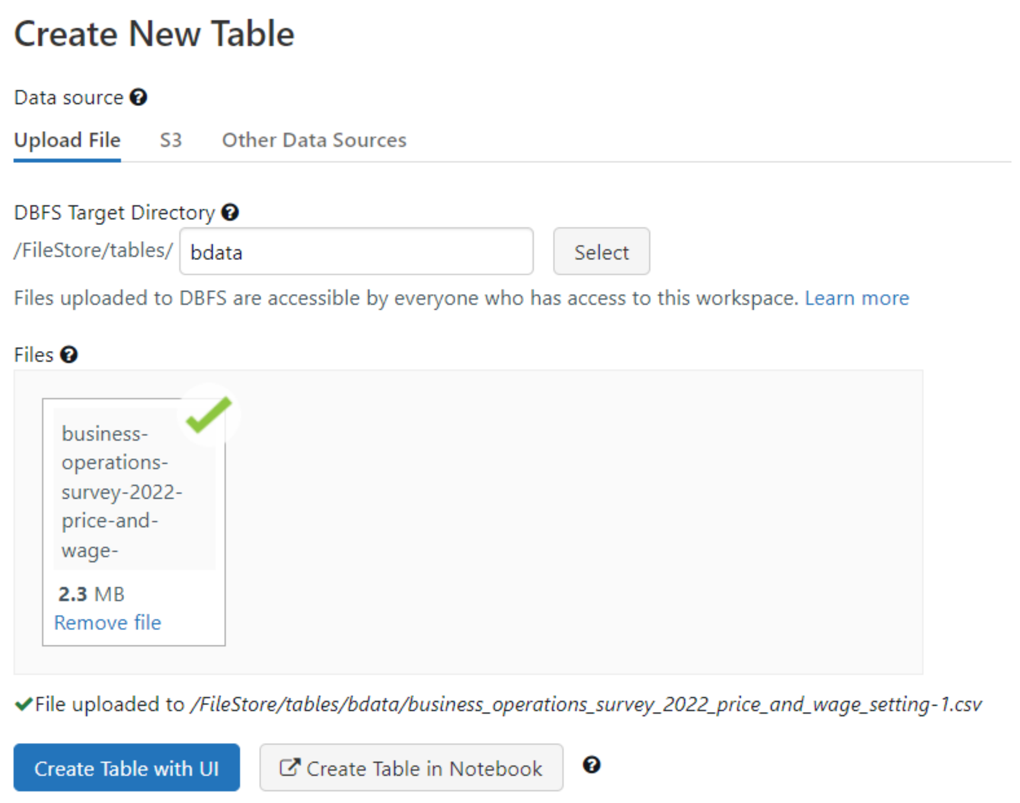
```
(2) Spark Jobs
df:  pyspark.sql.dataframe.DataFrame = [_c0: string]
```

Table

	_c0
1	Industry
2	Accounting/Finance
3	Advertising/Public Relations
4	Aerospace/Aviation
5	Arts/Entertainment/Publishing
6	Automotive
7	Banking/Mortgage

44 rows | 2.12 seconds runtime

Creating table using UI by uploading



Create New Table

Data source

Upload File | S3 | Other Data Sources

DBFS Target Directory

/FileStore/tables/ bdata

Files uploaded to DBFS are accessible by everyone who has access to this workspace. Learn more

Files

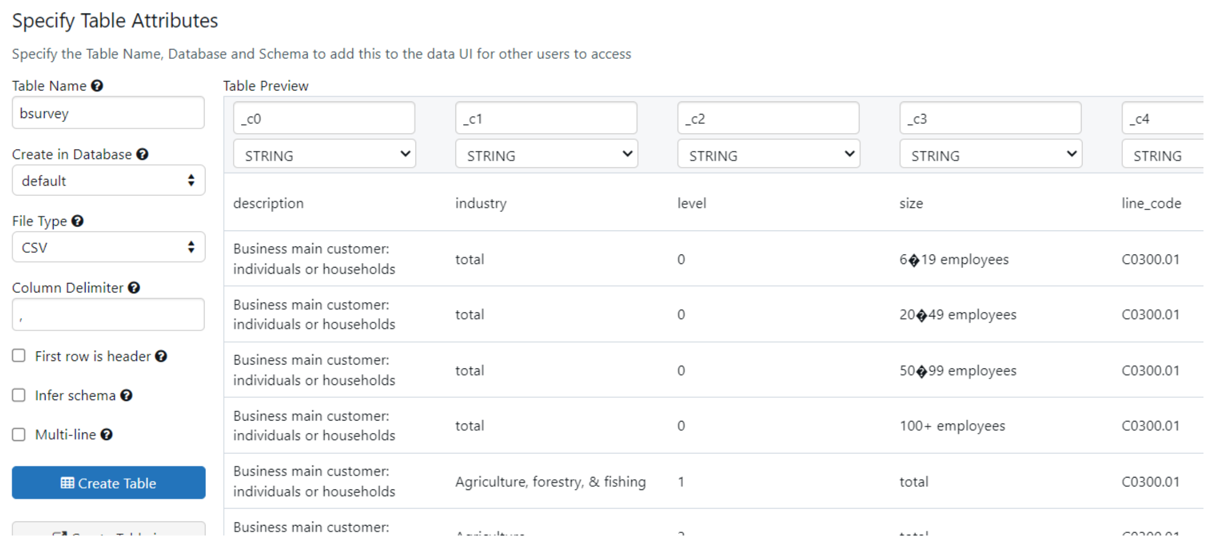
business-operations-survey-2022-price-and-wage-

2.3 MB

Remove file

✓File uploaded to */FileStore/tables/bdata/business_operations_survey_2022_price_and_wage_setting-1.csv*

Preview of the table created



Specify Table Attributes

Specify the Table Name, Database and Schema to add this to the data UI for other users to access

Table Name: bsurvey

Create in Database: default

File Type: CSV

Column Delimiter: ,

☐ First row is header

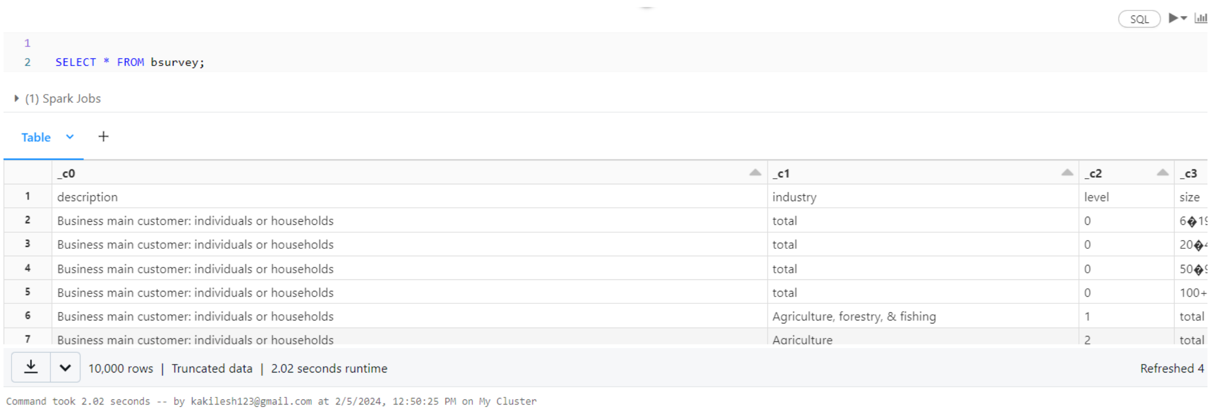
☐ Infer schema

☐ Multi-line

Table Preview

_c0	_c1	_c2	_c3	_c4
STRING	STRING	STRING	STRING	STRING
description	industry	level	size	line_code
Business main customer: individuals or households	total	0	6�19 employees	C0300.01
Business main customer: individuals or households	total	0	20�49 employees	C0300.01
Business main customer: individuals or households	total	0	50�99 employees	C0300.01
Business main customer: individuals or households	total	0	100+ employees	C0300.01
Business main customer: individuals or households	Agriculture, forestry, & fishing	1	total	C0300.01
Business main customer:				

Display table



```
SELECT * FROM bsurvey;
```

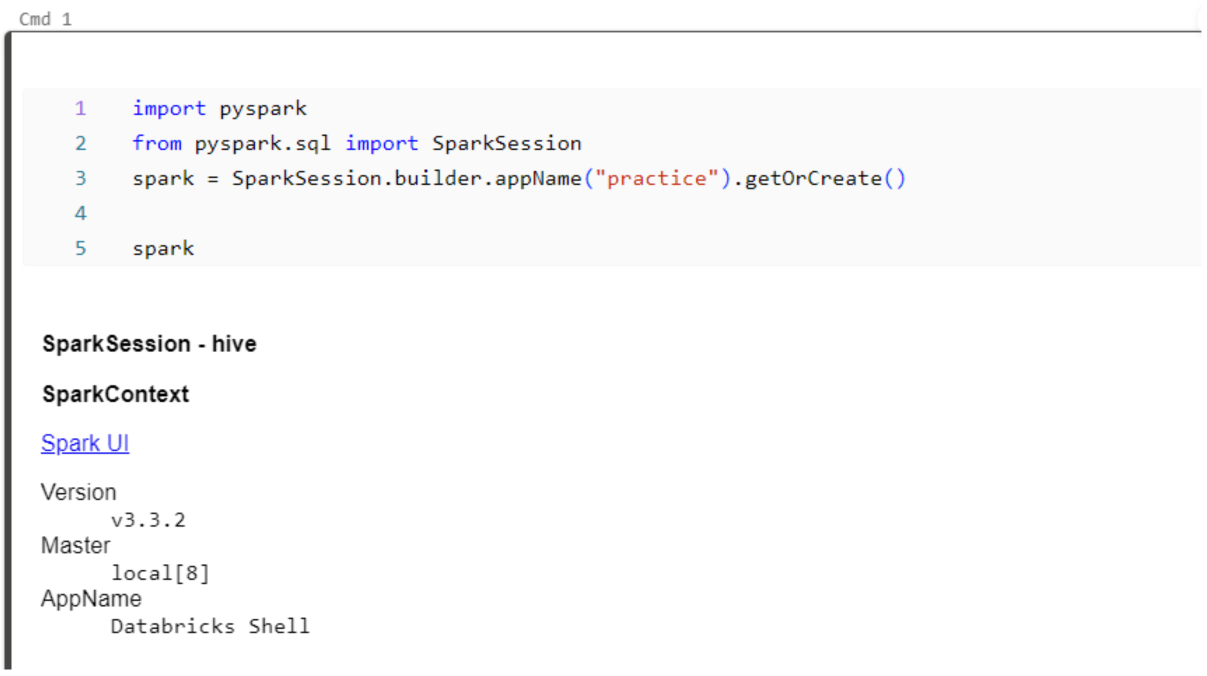
	_c0	_c1	_c2	_c3
1	description	industry	level	size
2	Business main customer: individuals or households	total	0	6�1
3	Business main customer: individuals or households	total	0	20�
4	Business main customer: individuals or households	total	0	50�
5	Business main customer: individuals or households	total	0	100+
6	Business main customer: individuals or households	Agriculture, forestry, & fishing	1	total
7	Business main customer: individuals or households	Agriculture	2	total

10,000 rows | Truncated data | 2.02 seconds runtime Refreshed 4

```
Command took 2.02 seconds -- by kakilesh123@gmail.com at 2/5/2024, 12:50:25 PM on My Cluster
```

Creating pyspark sessions

- **Import the SparkSession class from the pyspark.sql module. The SparkSession is a unified entry point for reading data, executing SQL queries, and working with DataFrames in Spark.**
- **Use the SparkSession.builder attribute to configure and create a SparkSession.**
- **Orcreate retrieves an existing SparkSession if one exists or creates a new one if none exists.**



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

spark
```

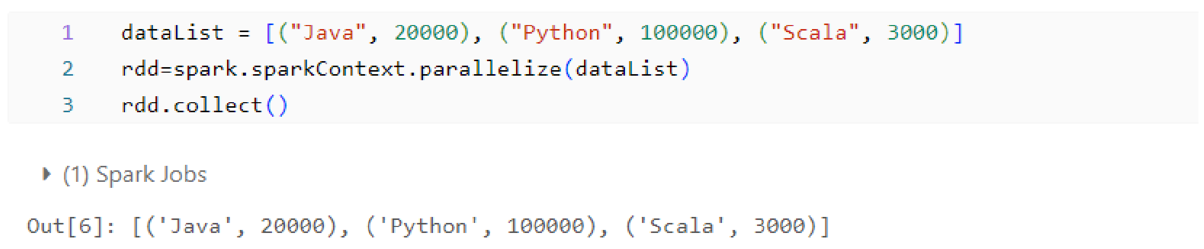
SparkSession - hive

SparkContext

Spark UI

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

RDD is created to parallelize the data

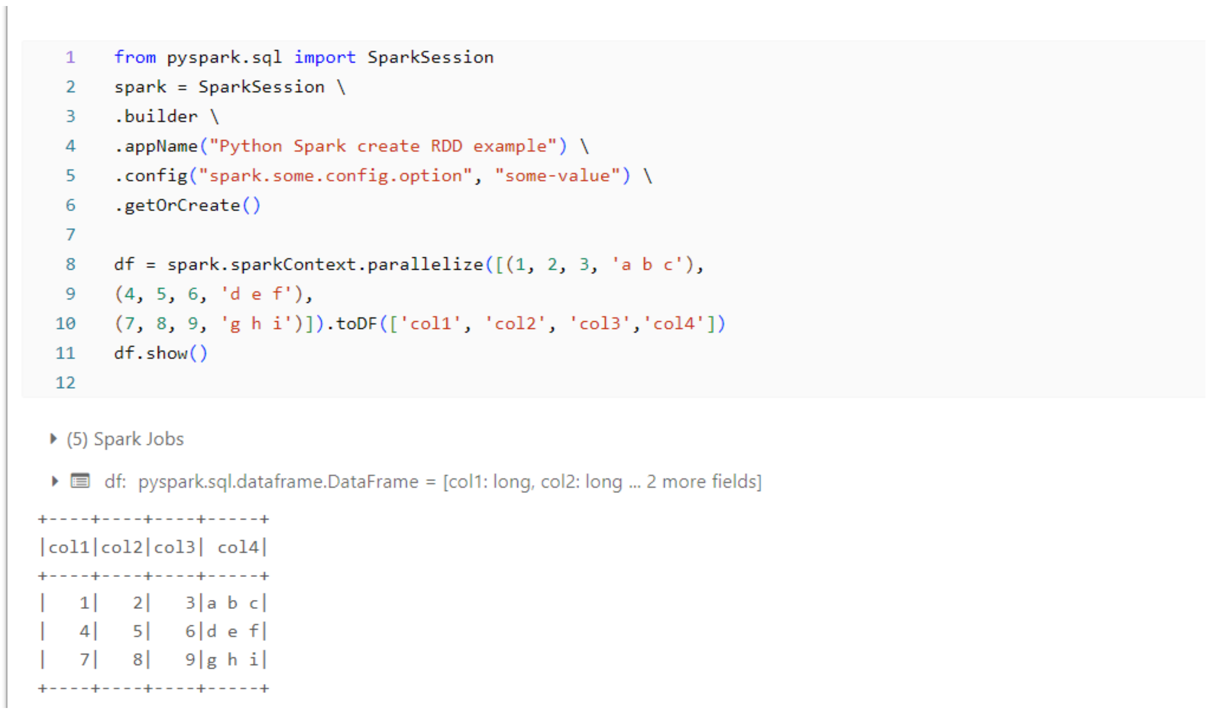


```
dataList = [("Java", 20000), ("Python", 100000), ("Scala", 3000)]
rdd=spark.sparkContext.parallelize(dataList)
rdd.collect()
```

▸ (1) Spark Jobs

```
Out[6]: [('Java', 20000), ('Python', 100000), ('Scala', 3000)]
```

Dataframe execution



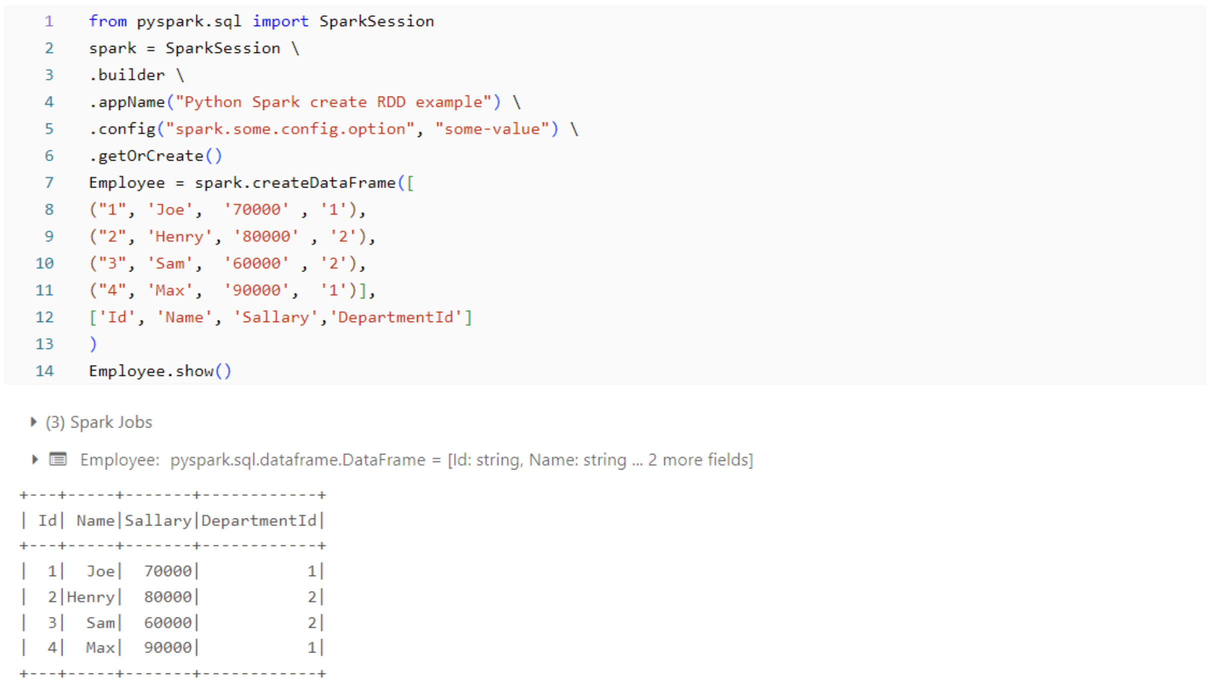
```
from pyspark.sql import SparkSession
spark = SparkSession \
.builder \
.appName("Python Spark create RDD example") \
.config("spark.some.config.option", "some-value") \
.getOrCreate()

df = spark.sparkContext.parallelize([(1, 2, 3, 'a b c'),
(4, 5, 6, 'd e f'),
(7, 8, 9, 'g h i')]).toDF(['col1', 'col2', 'col3','col4'])
df.show()

```

▸ (5) Spark Jobs

▸ df: pyspark.sql.dataframe.DataFrame = [col1: long, col2: long ... 2 more fields]

```
+----+----+----+-----+
|col1|col2|col3| col4|
+----+----+----+-----+
|   1|   2|   3|a b c|
|   4|   5|   6|d e f|
|   7|   8|   9|g h i|
+----+----+----+-----+
```

Creating a dataframe and displaying it



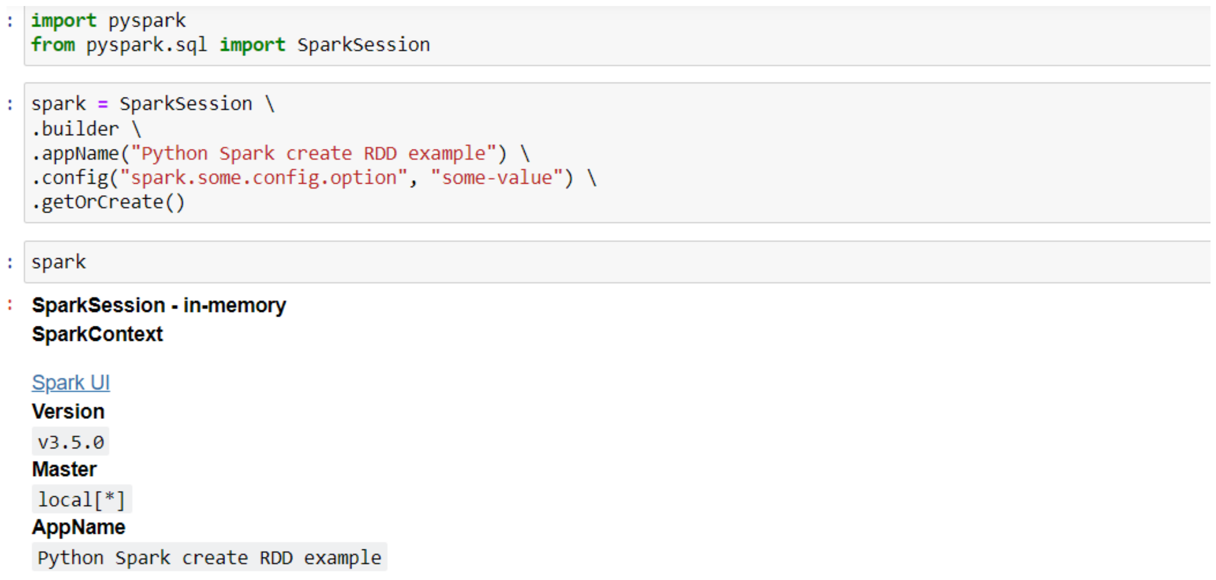
```python
from pyspark.sql import SparkSession
spark = SparkSession \
.builder \
.appName("Python Spark create RDD example") \
.config("spark.some.config.option", "some-value") \
.getOrCreate()
Employee = spark.createDataFrame([
("1", 'Joe',  '70000' , '1'),
("2", 'Henry', '80000' , '2'),
("3", 'Sam',  '60000' , '2'),
("4", 'Max',  '90000',  '1')],
['Id', 'Name', 'Sallary','DepartmentId']
)
Employee.show()
```

▸ (3) Spark Jobs

▸ Employee: pyspark.sql.dataframe.DataFrame = [Id: string, Name: string ... 2 more fields]

```
+---+-----+-------+------------+
| Id| Name|Sallary|DepartmentId|
+---+-----+-------+------------+
|  1|  Joe|  70000|           1|
|  2|Henry|  80000|           2|
|  3|  Sam|  60000|           2|
|  4|  Max|  90000|           1|
+---+-----+-------+------------+
```

Creating a session



```python
import pyspark
from pyspark.sql import SparkSession
```

```python
spark = SparkSession \
.builder \
.appName("Python Spark create RDD example") \
.config("spark.some.config.option", "some-value") \
.getOrCreate()
```

```python
spark
```

SparkSession - in-memory

SparkContext

Spark UI

Version

`v3.5.0`

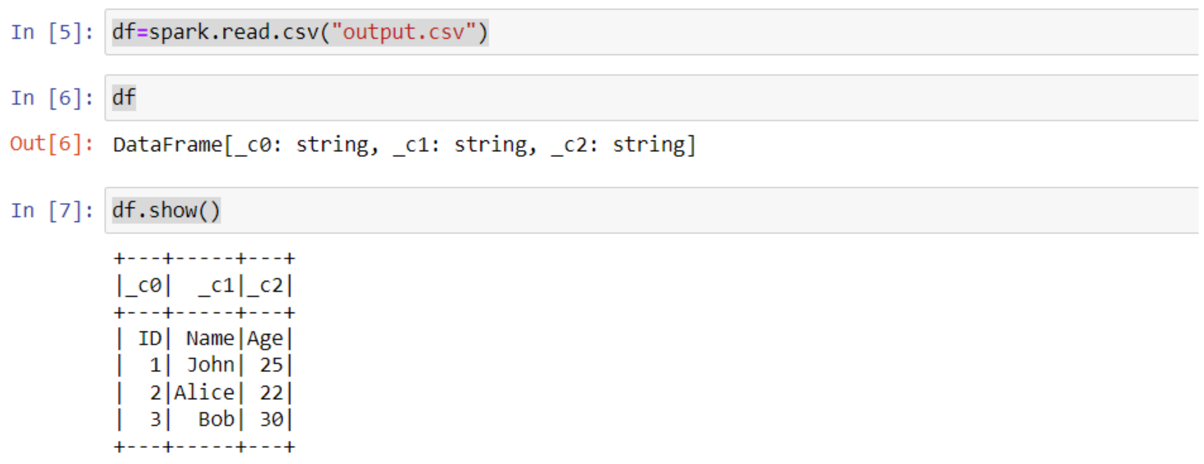
Master

`local[*]`

AppName

`Python Spark create RDD example`

Reading a CSV file and displaying it



```
In [5]: df=spark.read.csv("output.csv")

In [6]: df

Out[6]: DataFrame[_c0: string, _c1: string, _c2: string]

In [7]: df.show()

        +---+-----+---+
        |_c0|  _c1|_c2|
        +---+-----+---+
        | ID| Name|Age|
        |  1| John| 25|
        |  2|Alice| 22|
        |  3|  Bob| 30|
        +---+-----+---+
```