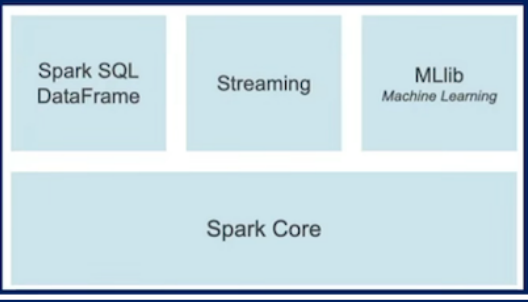
Create, Read, Write - CSV / Saving Modes / Read & Read Multi

JSON :-

PySpark

PySpark is an interface for Apache Spark in Python. It not only allows you to write Spark applications using Python APIs, but also provides the PySpark shell for interactively analyzing your data in a distributed environment. PySpark supports most of Spark's features such as Spark SQL, DataFrame, Streaming, MLlib (Machine Learning) and Spark Core.



createDataFrame()

DataFrame is **a distributed collection of data organized into named columns**. It is conceptually equivalent to a table in a relational database

I/P

①
```
data = [(1,'Maheer'),(2,'Wafa')]
schema = ['id','name']
df = spark.createDataFrame(data,schema)
df.show()
```

②
```
from pyspark.sql.types import *

data = [{'id':1,'name':'Maheer'},{'id':2,'name':'Wafa'}]
schema = StructType([StructField(name='id',dataType=IntegerType()),
                     StructField(name='name',dataType=StringType())])
df = spark.createDataFrame(data, schema)
df.show()
df.printSchema()
```

③
```
from pyspark.sql.types import *

data = [{'id':1,'name':'Maheer'},{'id':2,'name':'Wafa'}]

df = spark.createDataFrame(data)
df.show()
df.printSchema()
```

O/P

①
```
+---+------+
| id|  name|
+---+------+
|  1|Maheer|
|  2|  Wafa|
+---+------+

root
 |-- id: long (nullable = true)
 |-- name: string (nullable = true)
```

②
```
+---+------+
| id|  name|
+---+------+
|  1|Maheer|
|  2|  Wafa|
+---+------+

root
 |-- id: integer (nullable = true)
 |-- name: string (nullable = true)
```

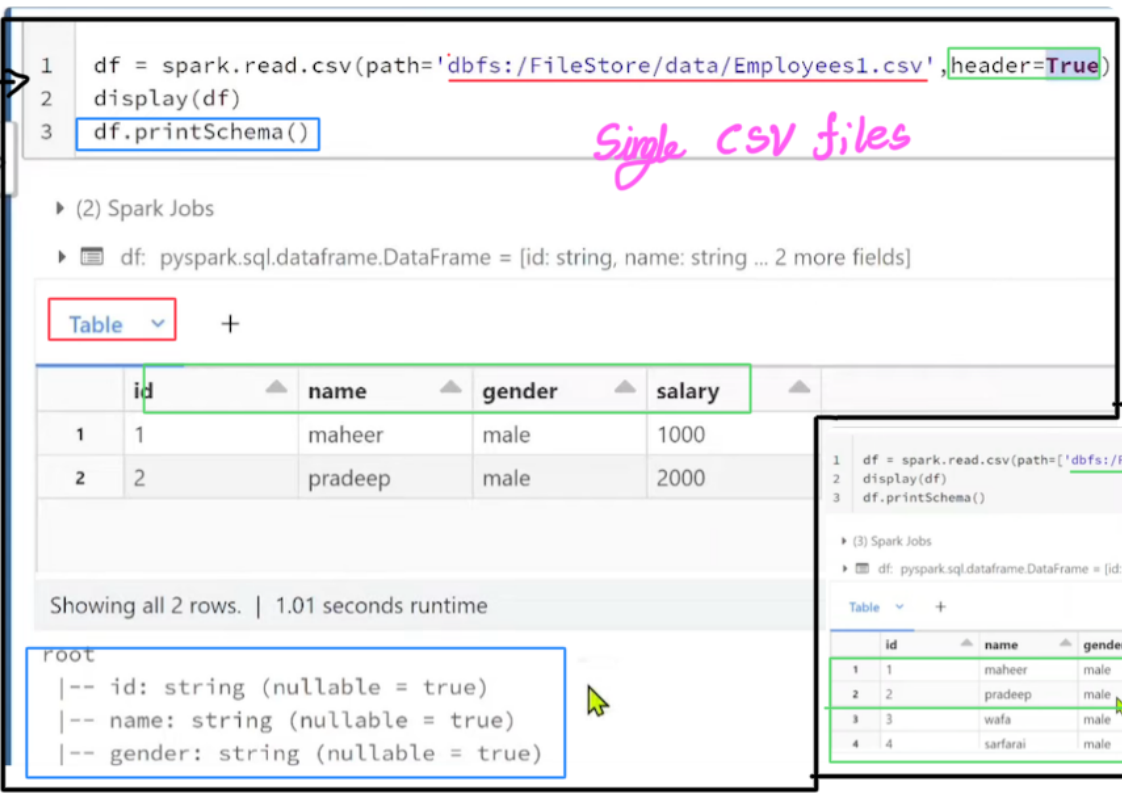
③
```
+---+------+
| id|  name|
+---+------+
|  1|Maheer|
|  2|  Wafa|
+---+------+

root
 |-- id: long (nullable = true)
 |-- name: string (nullable = true)
```

Read csv data into Dataframe

Using csv("path") or format("csv").load("path") of DataFrameReader, you can read a CSV file into a PySpark Dataframe

```
df = spark.read.format("csv")
                  .load("/tmp/resources/zipcodes.csv")
//       or
df = spark.read.format("org.apache.spark.sql.csv")
                  .load("/tmp/resources/zipcodes.csv")
df.printSchema()
```

```
df2 = spark.read.option("header",True) \
     .csv("/tmp/resources/zipcodes.csv")
```



```
df = spark.read.csv(path='dbfs:/FileStore/data/Employees1.csv',header=True)
display(df)
df.printSchema()
```

Single CSV files

▸ (2) Spark Jobs
▸ df: pyspark.sql.dataframe.DataFrame = [id: string, name: string ... 2 more fields]

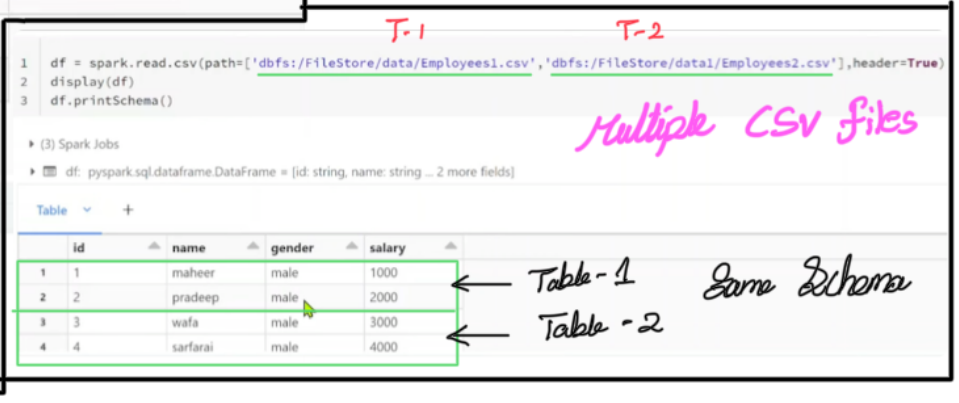
Table +

	id	name	gender	salary
1	1	maheer	male	1000
2	2	pradeep	male	2000

Showing all 2 rows. | 1.01 seconds runtime

```
root
 |-- id: string (nullable = true)
 |-- name: string (nullable = true)
 |-- gender: string (nullable = true)
```

['csv', 'csv2'] → List



T-1 T-2
```
df = spark.read.csv(path=['dbfs:/FileStore/data/Employees1.csv','dbfs:/FileStore/data1/Employees2.csv'],header=True)
display(df)
df.printSchema()
```

Multiple CSV files

▸ (3) Spark Jobs
▸ df: pyspark.sql.dataframe.DataFrame = [id: string, name: string ... 2 more fields]

Table +

	id	name	gender	salary
1	1	maheer	male	1000
2	2	pradeep	male	2000
3	3	wafa	male	3000
4	4	sarfarai	male	4000

← Table-1
← Table-2
Same Schema

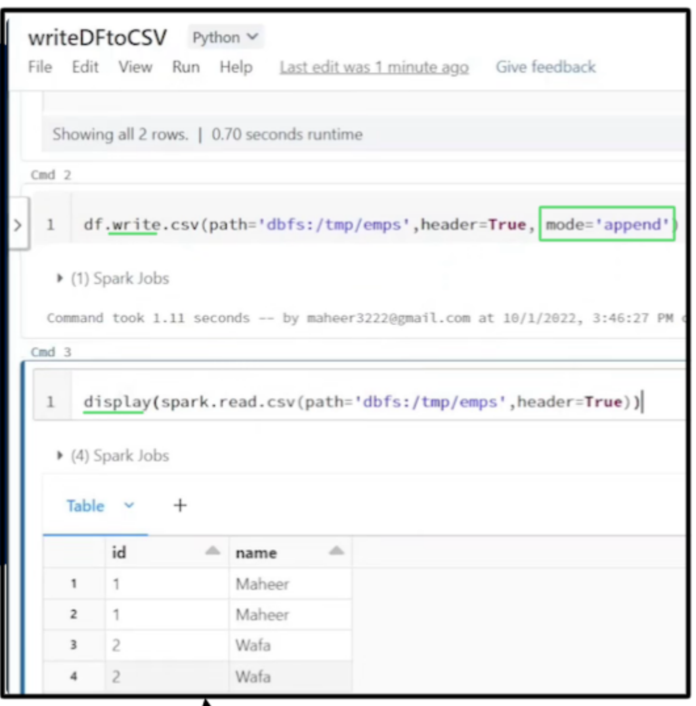
Write Dataframe into CSV

Use the write() method of the PySpark DataFrameWriter object to write PySpark DataFrame to a CSV file.

```
df.write.option("header",True).csv("/tmp/spark_output/zipcodes")
```

While writing a CSV file you can use several options. for example, header to output the DataFrame column names as header record and delimiter to specify the delimiter on the CSV output file.

```
df2.write.options(header='True', delimiter=',').csv("/tmp/spark_output/zipcodes")
```


writeDFtoCSV
Python
File Edit View Run Help Last edit was 1 minute ago Give feedback

Showing all 2 rows. | 0.70 seconds runtime

Cmd 2

```
df.write.csv(path='dbfs:/tmp/emps',header=True, mode='append')
```

(1) Spark Jobs

Command took 1.11 seconds -- by maheer3222@gmail.com at 10/1/2022, 3:46:27 PM

Cmd 3

```
display(spark.read.csv(path='dbfs:/tmp/emps',header=True))
```

(4) Spark Jobs

Table

	id	name
1	1	Maheer
2	1	Maheer
3	2	Wafa
4	2	Wafa

Saving Modes

PySpark DataFrameWriter also has a method mode() to specify saving mode.
overwrite – mode is used to overwrite the existing file.
append – To add the data to the existing file.
ignore – Ignores write operation when the file already exists.
error – This is a default option when the file already exists, it returns an error.

```
df2.write.mode('overwrite').csv("/data/emps")
#you can try below too
df2.write.format("csv").mode('overwrite').save("/data/emps")
```

Read JSON data into Dataframe

Using read.json("path") or read.format("json").load("path") you can read a JSON file into a PySpark DataFrame

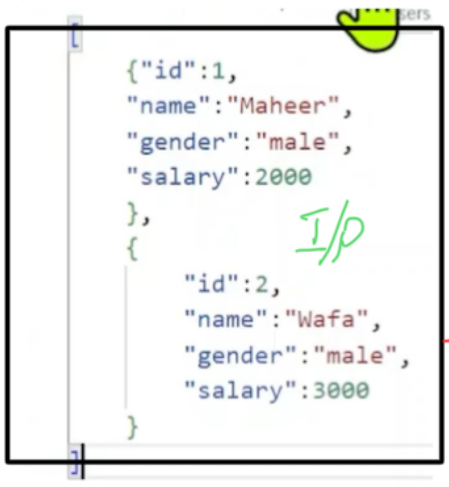
use multiline option to read JSON files scattered across multiple lines. By default multiline option, is set to false.

```
df = spark.read.format('org.apache.spark.sql.json') \
        .load('dbfs:/FileStore/data/emps.json')

df.printSchema()
df.show()
```

```
df = spark.read.json('dbfs:/FileStore/data/emps.json')
df.printSchema()
df.show()
```

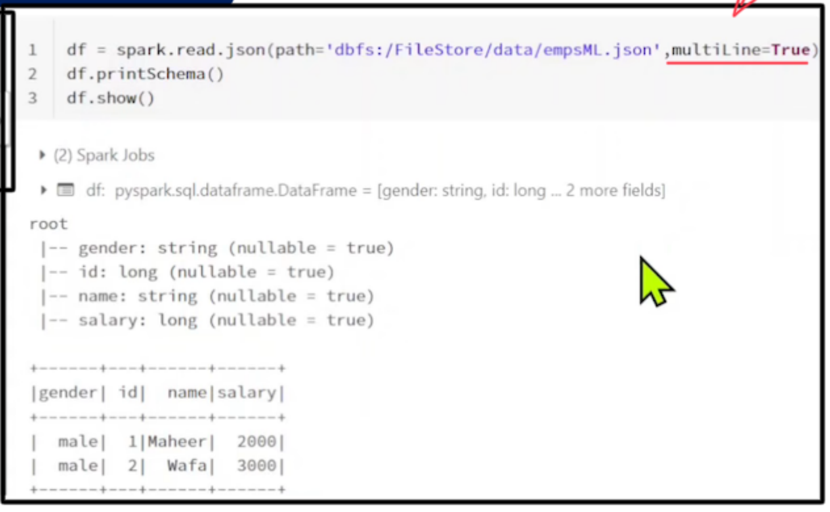
```
df = spark.read.json('dbfs:/FileStore/data/empsML.json',multiLine=True)
df.printSchema()
df.show()
```

Ex:- Multi line



I/P

```
[
  {"id":1,
  "name":"Maheer",
  "gender":"male",
  "salary":2000
  },
  {
     "id":2,
     "name":"Wafa",
     "gender":"male",
     "salary":3000
  }
]
```



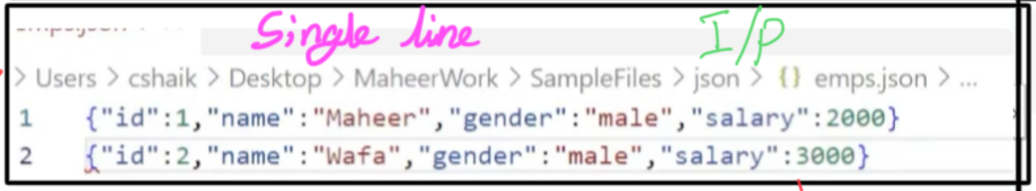
```
df = spark.read.json(path='dbfs:/FileStore/data/empsML.json',multiLine=True)
df.printSchema()
df.show()
```

(2) Spark Jobs

df: pyspark.sql.dataframe.DataFrame = [gender: string, id: long ... 2 more fields]

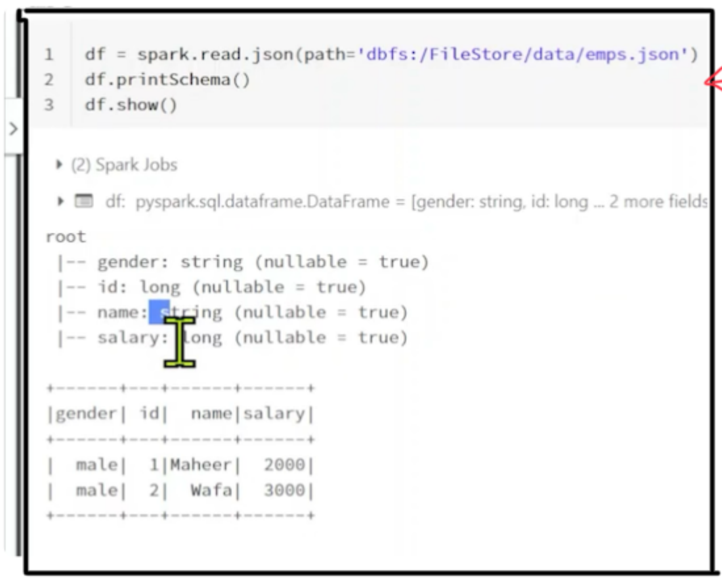
```
root
 |-- gender: string (nullable = true)
 |-- id: long (nullable = true)
 |-- name: string (nullable = true)
 |-- salary: long (nullable = true)

+------+---+------+------+
|gender| id|  name|salary|
+------+---+------+------+
|  male|  1|Maheer|  2000|
|  male|  2|  Wafa|  3000|
+------+---+------+------+
```

Single line I/P



> Users > cshaik > Desktop > MaheerWork > SampleFiles > json > {} emps.json > ...

```
{"id":1,"name":"Maheer","gender":"male","salary":2000}
{"id":2,"name":"Wafa","gender":"male","salary":3000}
```

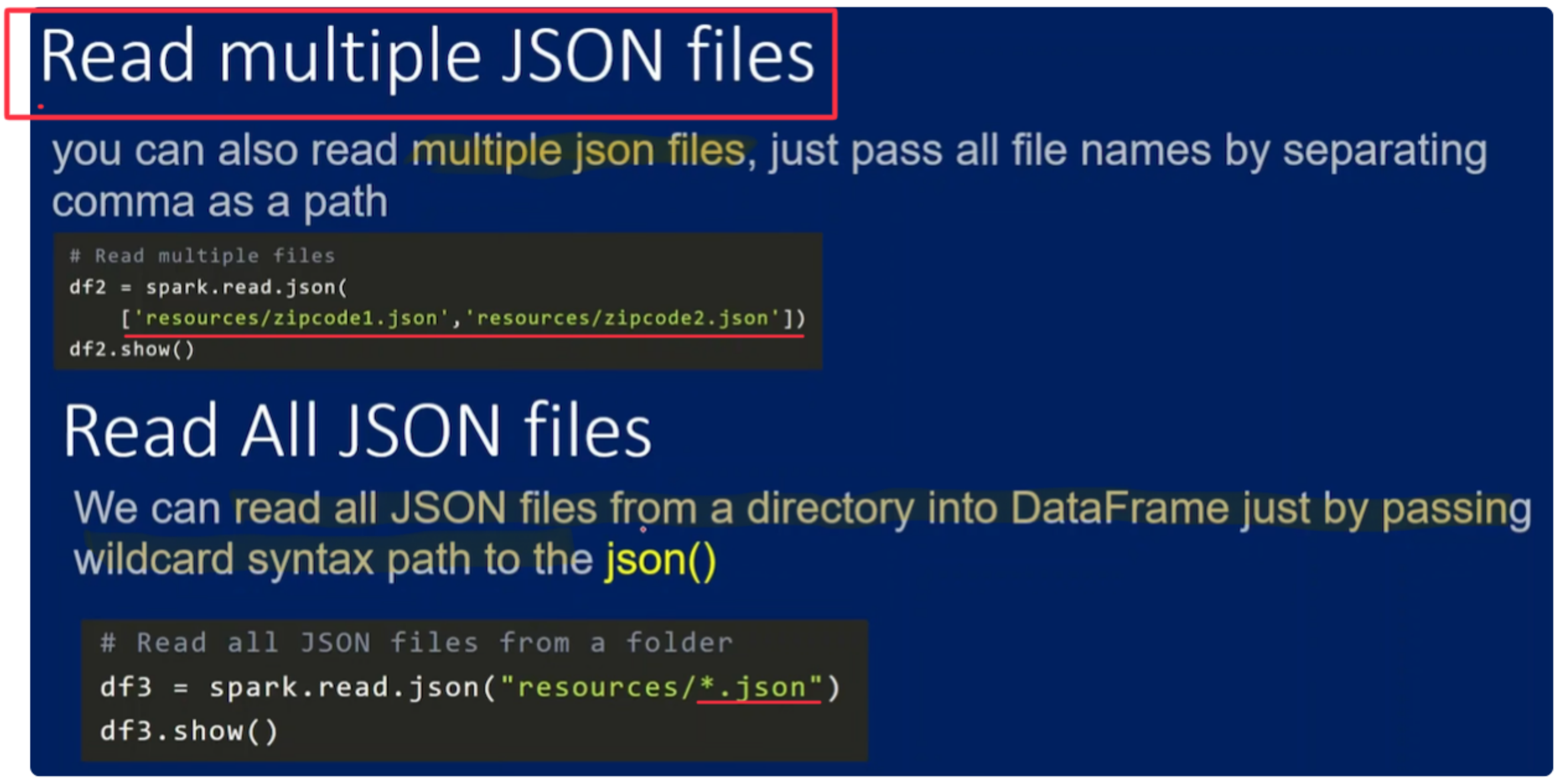


```
df = spark.read.json(path='dbfs:/FileStore/data/emps.json')
df.printSchema()
df.show()
```

(2) Spark Jobs

df: pyspark.sql.dataframe.DataFrame = [gender: string, id: long ... 2 more fields]

```
root
 |-- gender: string (nullable = true)
 |-- id: long (nullable = true)
 |-- name: string (nullable = true)
 |-- salary: long (nullable = true)

+------+---+------+------+
|gender| id|  name|salary|
+------+---+------+------+
|  male|  1|Maheer|  2000|
|  male|  2|  Wafa|  3000|
+------+---+------+------+
```



Read multiple JSON files

you can also read multiple json files, just pass all file names by separating comma as a path

```
# Read multiple files
df2 = spark.read.json(
    ['resources/zipcode1.json','resources/zipcode2.json'])
df2.show()
```

Read All JSON files

We can read all JSON files from a directory into DataFrame just by passing wildcard syntax path to the json()

```
# Read all JSON files from a folder
df3 = spark.read.json("resources/*.json")
df3.show()
```