

Java-Success.com

Prepare to fast-track, choose & go places with 800+ Java & Big Data Q&As with lots of code & diagrams.

[Home](#) [Why? ▾](#) [300+ Java FAQs ▾](#) [300+ Big Data FAQs ▾](#) [Courses ▾](#)[👤 Membership ▾](#) [Your Career ▾](#)[Home](#) > [bigdata-success.com](#) > [Tutorials - Big Data](#) > [TUT - Spark Scala on Zeppelin](#) >

03: Spark on Zeppelin – DataFrame Operations in Scala

03: Spark on Zeppelin – DataFrame Operations in Scala

 Posted on [July 25, 2018](#)

Pre-requisite: Docker is installed on your machine for Mac OS X (E.g. \$ brew cask install docker) or Windows 10. [Docker interview Q&As](#).

This tutorial extends [Apache Zeppelin on Docker Tutorial – Docker pull from Docker hub](#) and [Spark stand-alone to read a file from local file system](#)

```
1 val lines = sc.textFile("file:///zeppelin/seed/emp")
2
3 case class Employee (id: Integer, name: String, loc: String)
4
5 val dfEmployees = lines.map(s => s.split(",")).map{line => Employee(line(0).toInt, line(1), line(2))}
```

300+ Java Interview FAQs

300+ Java FAQs



16+ Java Key Areas Q&As



150+ Java Architect FAQs



80+ Java Code Quality Q&As



150+ Java Coding Q&As



300+ Big Data Interview FAQs

300+ Big Data FAQs



Tutorials - Big Data

TUT -  Starting Big Data

TUT - Starting Spark & Scala

6
7

1. Print the schema of the Dataframe

```
1 dfEmployees.printSchema()
2
```

root

|– id: integer (nullable = true)
 |– name: string (nullable = true)
 |– location: string (nullable = true)
 |– salary: double (nullable = true)

2. Show contents of a Dataframe

```
1 dfEmployees.show()
2
```

```
1
2 +---+-----+-----+-----+
3 | id|      name|  location|  salary|
4 +---+-----+-----+-----+
5 |  1|     John|      USA|100000.0|
6 |  2|    Peter| Australia|200000.0|
7 |  3|      Sam|      USA| 76000.0|
8 |  4|   Daniel|   France| 86000.0|
9 |  5|    Simon| Australia| 96000.0|
10 |  6| Roseanne|   France|156000.0|
11 +---+-----+-----+-----+
12
```

3. Count number of rows in a Dataframe

```
1 dfEmployees.count()
2
```

res12: Long = 6

4. Add a new column to a Dataframe

TUT - Starting with Python

TUT - Kafka

TUT - Pig

TUT - Apache Storm

TUT - Spark Scala on Zeppelin

TUT - Cloudera

TUT - Cloudera on Docker

TUT - File Formats

TUT - Spark on Docker

TUT - Flume

TUT - Hadoop (HDFS)

TUT - HBase (NoSQL)

TUT - Hive (SQL)

TUT - Hadoop & Spark

TUT - MapReduce

TUT - Spark and Scala

TUT - Spark & Java

TUT - PySpark on Databricks

TUT - Zookeeper

800+ Java Interview Q&As

300+ Core Java Q&As



300+ Enterprise Java Q&As



150+ Java Frameworks Q&As



120+ Companion Tech Q&As



Tutorials - Enterprise Java



```

1 dfEmployees.withColumn("bonus", dfEmployees.col("salary")
2   .show()
3

```

```

1
2 +---+-----+-----+-----+-----+
3 | id|      name| location| salary| bonus|
4 +---+-----+-----+-----+-----+
5 |  1|    John|      USA|100000.0|2000.0|
6 |  2|   Peter| Australia|200000.0|4000.0|
7 |  3|    Sam|      USA| 76000.0|1520.0|
8 |  4|  Daniel|   France| 86000.0|1720.0|
9 |  5|   Simon| Australia| 96000.0|1920.0|
10 |  6|Roseanne|   France|156000.0|3120.0|
11 +---+-----+-----+-----+-----+
12

```

You can drop a column with

`"dfEmployees.drop("location").show()"`

5. Select a few columns from a Dataframe

```

1
2 dfEmployees.withColumn("bonus", dfEmployees.col("salary")
3   .select("id", "bonus")
4   .show()
5

```

```

1
2 +---+-----+
3 | id| bonus|
4 +---+-----+
5 |  1|2000.0|
6 |  2|4000.0|
7 |  3|1520.0|
8 |  4|1720.0|
9 |  5|1920.0|
10 |  6|3120.0|
11 +---+-----+
12

```

6. Distinct values

```

1

```

```
2 dfEmployees.select("location")
3               .distinct()
4               .show()
5
```

```
1
2 +-----+
3 | location|
4 +-----+
5 | Australia|
6 |      USA|
7 |    France|
8 +-----+
9
```

7. Sorting

```
1
2 dfEmployees.orderBy("location")
3               .show()
4
```

```
1
2 +---+-----+-----+-----+
3 | id|      name| location| salary|
4 +---+-----+-----+-----+
5 |  2|    Peter| Australia|200000.0|
6 |  5|    Simon| Australia| 96000.0|
7 |  4|   Daniel|    France| 86000.0|
8 |  6| Roseanne|    France|156000.0|
9 |  1|    John|      USA|100000.0|
10 |  3|    Sam|      USA| 76000.0|
11 +---+-----+-----+-----+
12
```

8. Applying SQL queries

```
1
2 dfEmployees.registerTempTable("employees_tbl")
3
```

```
1
2 %sql
3
4 Select * from employees_tbl
5
```

9. Filtering by a predicate

```
1
2 dfEmployees.filter(dfEmployees.col("salary") > 100000)
3
```

```
1
2 +-----+-----+-----+-----+
3 | id|      name| location| salary|
4 +-----+-----+-----+-----+
5 |  2|    Peter| Australia|200000.0|
6 |  6| Roseanne|    France|156000.0|
7 +-----+-----+-----+-----+
8
```

10. Grouping & aggregation

```
1
2 dfEmployees.groupBy("location").agg(sum("salary"))
3
```

```
1
2 +-----+-----+
3 | location|sum(salary)|
4 +-----+-----+
5 | Australia|   296000.0|
6 |        USA|   176000.0|
7 |    France|   242000.0|
8 +-----+-----+
9
```

11. Map operations on Dataframe columns

We can apply a function on each row of DataFrame using map operation.

```
1
2 dfEmployees.select("id", "salary").map(row => (row
3
```

```
1
2 +-----+-----+
3 | _1|      _2|
4 +-----+-----+
5 |  1|100220.0|
```

```

6 | 2|200220.0|
7 | 3| 76220.0|
8 | 4| 86220.0|
9 | 5| 96220.0|
10| 6|156220.0|
11 +---+-----+
12

```

12. Get some stats on your data

```

1
2 dfEmployees.select("salary").describe().show()
3

```

```

1
2 +-----+-----+
3 |summary|          salary|
4 +-----+-----+
5 |  count|              6|
6 |   mean|         119000.0|
7 | stddev|48493.29850608226|
8 |   min|          76000.0|
9 |   max|         200000.0|
10 +-----+-----+
11

```

◀ 02: Spark on Zeppelin – read a file from local file system

04: Spark on Zeppelin – DataFrame joins in Scala ▶

Disclaimer

The contents in this Java-Success are copyrighted and from EmpoweringTech pty ltd. The EmpoweringTech pty ltd has the right to correct or enhance the current content without any prior notice. These are general advice only, and one needs to take his/her own circumstances into consideration. The EmpoweringTech pty ltd will not be held liable for any damages caused or alleged to be caused either directly or indirectly by these materials and resources. Any trademarked names or labels used in this blog remain the property of their respective trademark owners. Links to external sites do not imply endorsement of the linked-to sites. [Privacy Policy](#)