

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](#) >

09: Spark on Zeppelin – convert DataFrames to RDD and RDD to DataFrame

09: Spark on Zeppelin – convert DataFrames to RDD and RDD to DataFrame

 Posted on [September 11, 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 extends [setting up Apache Zeppelin Notebook](#).

Important: It is not a best practice to mutate values or to use RDD directly as opposed to using Dataframes. Use of groupBy operation on RDD is also discouraged as it causes wide shuffling. There are alternative solutions like using “**withColumn**” on Dataframe to

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

add a new column, **UDF functions**, **window functions**, **explode function**, etc to achieve the desired results more efficiently. There will be edge cases where use of the RDDs will give you more flexibility to achieve the desired outcome. The examples shown below is for the illustration purpose only.

Step 1: Pull this from the docker hub, and build the image with the following command.

```
1 $ docker pull apache/zeppelin:0.7.3
2
```

You can verify the image with the “docker images” command.

Step 2: Run the container with the above image.

```
1 $ docker run --rm -it -p 8080:8080 apache/zeppelin
2
```

Step 3: Open Zeppelin notebook via a web browser “http://localhost:8080”. Create a note book with “spark” as a default interpreter.

This extends **Spark convert DataFrames to RDD[Row]** and **RDD[Row] to DataFrame** with use of a case class named Employee.

```
1 %spark
2
3
4 import org.apache.spark.sql.types._
5 import org.apache.spark.sql.Row
6 import org.apache.spark.rdd.RDD
7
8 case class Employee (id: Integer, name: String, lo
9
```

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



```

10 val employees = Seq(
11     Employee(1, "John", "USA", 50000.0),
12     Employee(2, "Peter", "AU", 60000.0),
13     Employee(3, "Sam", "AU", 60000.0),
14     Employee(4, "Susan", "USA", 50000.0),
15     Employee(5, "David", "USA", 70000.0),
16     Employee(6, "Elliot", "AU", 50000.0)
17 )
18
19 val employee_df = spark.createDataFrame(
20     spark.sparkContext.parallelize(employees)
21 )
22
23 employee_df.show()
24
25 val rdd = employee_df.rdd
26     .groupBy(row => row(2)) //group by location
27     .flatMap(x => {          //flatMap
28         //x is a Tuple, x._1 is location, x._2 is list of employees
29         // filter salary 60k or more
30         for (row <- x._2 if row._3 >= 60000.0) yield row
31     })
32
33 val df2 = sqlContext.createDataFrame(rdd.map { case (location, employees) => employees })
34 df2.show()
35

```

Output:

```

1 import org.apache.spark.sql.types._
2 import org.apache.spark.sql.Row
3 import org.apache.spark.rdd.RDD
4 defined class Employee
5 employees: Seq[Employee] = List(Employee(1,John,USA,50000.0),Employee(2,Peter,AU,60000.0),Employee(3,Sam,AU,60000.0),Employee(4,Susan,USA,50000.0),Employee(5,David,USA,70000.0),Employee(6,Elliot,AU,50000.0))
6 employee_df: org.apache.spark.sql.DataFrame = [id: int, name: string, location: string, salary: double]
7
8 +---+-----+-----+-----+
9 | id | name | location | salary |
10 +---+-----+-----+-----+
11 | 1 | John | USA | 50000.0 |
12 | 2 | Peter | AU | 60000.0 |
13 | 3 | Sam | AU | 60000.0 |
14 | 4 | Susan | USA | 50000.0 |
15 | 5 | David | USA | 70000.0 |
16 | 6 | Elliot | AU | 50000.0 |
17 +---+-----+-----+-----+
18 rdd: org.apache.spark.rdd.RDD[org.apache.spark.sql.Row] = RDD[6 partitions]
19
20 df2: org.apache.spark.sql.DataFrame = [id: int, name: string, location: string, salary: double]
21
22 +---+-----+-----+-----+
23 | 2 | Peter | AU | 60000.0 |
24 | 3 | Sam | AU | 60000.0 |

```

24		5		David		USA		70000.0	
25	+-----+								
26									

◀ 08: Spark on Zeppelin – convert DataFrames to RDD[Row] and RDD[Row]
to DataFrame

10: Spark on Zeppelin – union, udf and explode ▶

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](#)