Extract column values of Dataframe as List in Apache Spark

Asked 5 years, 11 months ago Active 11 days ago Viewed 240k times

100

38

I want to convert a string column of a data frame to a list. What I can find from the `Dataframe` API is RDD, so I tried converting it back to RDD first, and then apply `toArray` function to the RDD. In this case, the length and SQL work just fine. However, the result I got from RDD has square brackets around every element like this `[A00001]`. I was wondering if there's an appropriate way to convert a column to a list or a way to remove the square brackets.

Any suggestions would be appreciated. Thank you!

scala apache-spark apache-spark-sql

Share Improve this question Follow

edited Oct 19 '19 at 6:05
mrsrinivas
28.6k 11 109 119

asked Aug 14 '15 at 0:39
SH Y.
1,569 3 16 21

ways to solve it with Spark 2.x – mrsrinivas May 20 '17 at 10:31

10 Answers

Active Oldest Votes

124

This should return the collection containing single list:

```
dataFrame.select("YOUR_COLUMN_NAME").rdd.map(r => r(0)).collect()
```

Without the mapping, you just get a Row object, which contains every column from the database.

Keep in mind that this will probably get you a list of Any type. Ïf you want to specify the result type, you can use .asInstanceOf[YOUR_TYPE] in `r => r(0).asInstanceOf[YOUR_TYPE]` mapping

P.S. due to automatic conversion you can skip the `.rdd` part.

Share Improve this answer Follow

edited Aug 25 '16 at 6:28

answered Aug 14 '15 at 7:49
TheMP
7,727 9 37 69

3 For some strange reason it works the other way round (Spark 2.1.0) `collect().map(r => r(0))` - does this order have any disadvantages ? – Boern Mar 14 '17 at 13:17

1 Can be slower - your solution first collects all the data on the driver, and after that it does the mapping on the driver (without executors aid), using only the processing power of single driver. – TheMP Mar 14 '17 at 15:44

81

With Spark 2.x and Scala 2.11

I'd think of 3 possible ways to convert values of a specific column to List.

Common code snippets for all the approaches

```
import org.apache.spark.sql.SparkSession

val spark = SparkSession.builder.getOrCreate
import spark.implicits._ // for .toDF() method

val df = Seq(
    ("first", 2.0),
    ("test", 1.5),
    ("choose", 8.0)
  ).toDF("id", "val")
```

Approach 1

```
df.select("id").collect().map(_(0)).toList
// res9: List[Any] = List(one, two, three)
```

What happens now? We are collecting data to Driver with `collect()` and picking element zero from each record.

This could not be an excellent way of doing it, Let's improve it with next approach.

Approach 2

```
df.select("id").rdd.map(r => r(0)).collect.toList
//res10: List[Any] = List(one, two, three)
```

How is it better? We have distributed map transformation load among the workers rather than single Driver.

I know `rdd.map(r => r(0))` does not seems elegant you. So, let's address it in next approach.

Approach 3

```
df.select("id").map(r => r.getString(0)).collect.toList
//res11: List[String] = List(one, two, three)
```

Here we are not converting DataFrame to RDD. Look at `map` it won't accept `r => r(0)` (or `_(0)`) as the previous approach due to encoder issues in DataFrame. So end up using `r => r.getString(0)` and it would be addressed in the next versions of Spark.

Conclusion

All the options give the same output but 2 and 3 are effective, finally 3rd one is effective and elegant(I'd think).

Databricks notebook

Share Improve this answer Follow

edited Jan 26 at 2:22

answered May 20 '17 at 6:44
mrsrinivas
28.6k 11 109 119

26

I know the answer given and asked for is assumed for Scala, so I am just providing a little snippet of Python code in case a PySpark user is curious. The syntax is similar to the given answer, but to properly pop the list out I actually have to reference the column name a second time in the mapping function and I do not need the select statement.

i.e. A DataFrame, containing a column named "Raw"

To get each row value in "Raw" combined as a list where each entry is a row value from "Raw" I simply use:

```
MyDataFrame.rdd.map(lambda x: x.Raw).collect()
```

Share Improve this answer Follow

answered Sep 30 '16 at 23:41
abby sobh
1,494 18 15

4 This gives a list of Row objects. What if you want a list of the values? – ThatDataGuy Nov 21 '16 at 16:45

This gives a list of values. – abby sobh Dec 1 '16 at 19:25

Thanks for sharing this! This works for me great just wondering if there is a way to speed this up, it runs pretty slow – Mojgan Mazouchi Jul 23 '20 at 23:39

5

In Scala and Spark 2+, try this (assuming your column name is "s"): `df.select('s).as[String].collect`

Share Improve this answer Follow

answered Jul 10 '17 at 17:20
kanielc
1,005 10 13

3

```
sqlContext.sql(" select filename from tempTable").rdd.map(r =>
r(0)).collect.toList.foreach(out_streamfn.println) //remove brackets
```

it works perfectly

Share Improve this answer Follow

edited Jun 19 '19 at 14:03
Oli

answered Dec 16 '17 at 5:58
Shaina Raza

6,589 2 10 32 689 5 7

3

```
from pyspark.sql.functions import col

df.select(col("column_name")).collect()
```

here collect is functions which in turn convert it to list. Be ware of using the list on the huge data set. It will decrease performance. It is good to check the data.

Share Improve this answer Follow

edited Jan 21 '20 at 13:29
Avi
3,271 8 18 29

answered Jan 21 '20 at 12:47
amarnath pimple
139 1 6

3

```
List<String> whatever_list = df.toJavaRDD().map(new Function<Row, String>() {
    public String call(Row row) {
        return row.getAs("column_name").toString();
    }
}).collect();

logger.info(String.format("list is %s",whatever_list)); //verification
```

Since no one has given any solution in java(Real Programming Language) Can thank me later

Share Improve this answer Follow

edited Apr 17 '20 at 2:06
Bob
1,301 10 27

answered Apr 16 '20 at 20:13
user12910640
59 3

2

Below is for Python-

```
df.select("col_name").rdd.flatMap(lambda x: x).collect()
```

Share Improve this answer Follow

answered Nov 26 '20 at 21:24
Nitin Mahajan
49 5

Others answers (such as stackoverflow.com/a/59841515/6807769) are similar – Vincent Doba Nov 26 '20 at 22:58

1 @VincentDoba - There will always be similar answers for a query. It's not the same and none of the answers use flatMap in python in this thread. It's very easy to downvote rather than helping people. Anyways. – Nitin Mahajan Jan 28 at 2:59

0

An updated solution that gets you a list:

```
dataFrame.select("YOUR_COLUMN_NAME").map(r => r.getString(0)).collect.toList
```

Share Improve this answer Follow

answered Apr 30 '20 at 16:00

Athanasios Tsiaras
101 6

-1

This is java answer.

```
df.select("id").collectAsList();
```

Share Improve this answer Follow

answered Feb 14 '20 at 8:43

vahbuna
1