

## Announcing Stack Overflow Documentation

We started with Q&A. Technical documentation is next, and we need your help.


Whether you're a beginner or an experienced developer, you *can* contribute.

[Sign up and start helping →](#)[Learn more about Documentation →](#)

## Updating a dataframe column in spark

Work on work you love. **From home.**



 stackoverflow

Looking at the new spark dataframe api, it is unclear whether it is possible to modify dataframe columns.

How would I go about changing a value in row  $x$  column  $y$  of a dataframe?

In pandas this would be `df.ix[x,y] = new_value`

[python](#) [apache-spark](#) [pyspark](#) [apache-spark-sql](#) [spark-dataframe](#)

edited Jan 15 at 16:34



[karlson](#)

1,512 1 7 27

asked Mar 17 '15 at 21:19



[Luke](#)

659 1 5 20

if you want to access the DataFrame by index, you need to build an index first. See, e.g. [stackoverflow.com/questions/26828815/...](http://stackoverflow.com/questions/26828815/...) Or add an index column with your own index. – [fanfabbb](#) Mar 31 '15 at 9:38

543 followers, 2.2k questions

[rss](#)

Apache Spark SQL is a tool for for "SQL and structured data processing" on Spark, a fast and general-purpose cluster computing system.

## 4 Answers

While you cannot modify a column as such, you may operate on a column and return a new DataFrame reflecting that change. For that you'd first create a `UserDefinedFunction` implementing the operation to apply and then selectively apply that function to the targeted column only. In Python:

```
from pyspark.sql.functions import UserDefinedFunction
from pyspark.sql.types import StringType

name = 'target_column'
udf = UserDefinedFunction(lambda x: 'new_value', StringType())
new_df = old_df.select(*[udf(column).alias(name) if column == name else column for column
in old_df.columns])
```

`new_df` now has the same schema as `old_df` (assuming that `old_df.target_column` was of type `StringType` as well) but all values in column `target_column` will be `new_value`.

edited Sep 10 '15 at 22:05



Kirk Broadhurst

15.7k 5 46 100

answered Mar 25 '15 at 13:35



karlson

1,512 1 7 27

1 this is an actual answer to the problem thanks! yet, the spark jobs don't finish for me, all executors get los. can you think of an alternative way? I use it with a bit more complex UDF where I do transformation to strings. There is no pandas-similar Syntax like `new_df = old_df.col1.apply(lambda x: func(x))`? – [fanfabbb](#) Mar 31 '15 at 7:58

5 there is also: `new_df = old_df.withColumn('target_column', udf(df.name))` – [fanfabbb](#) Mar 31 '15 at 13:13

@fanfabbb Would you mind posting your UDF? I don't see why a complex string operation would result in lost executors. Maybe decreasing partition sizes might help, e.g. by increasing the number of partitions. – [karlson](#) Mar 31 '15 at 14:44

1 Yes, that should work fine. Keep in mind that UDFs can only take columns as parameters. If you want to pass other data into the function you have to partially apply it first. – [karlson](#) May 18 '15 at 6:16

1 @KatyaHandler If you just want to duplicate a column, one way to do so would be to simply select it twice: `df.select([df[col], df[col].alias('same_column')])`, where `col` is the name of the column you want to duplicate. With the latest Spark release, a lot of the stuff I've used UDFs for can be done with the functions defined in `pyspark.sql.functions`. UDF performance in Pyspark is really poor, so that might really be worth looking into: [spark.apache.org/docs/latest/api/python/...](http://spark.apache.org/docs/latest/api/python/) – [karlson](#) Sep 23 '15 at 19:48



Just as [maasg](#) says you can create a new DataFrame from the result of a map applied to the old DataFrame. An example for a given DataFrame `df` with two rows:

```
val newDf = sqlContext.createDataFrame(df.map(row =>
  Row(row.getInt(0) + SOMETHING, applySomeDef(row.getAs[Double]("y")), df.schema)
```

Note that if the types of the columns change, you need to give it a correct schema instead of `df.schema`. Check out the api of `org.apache.spark.sql.Row` for available methods:

<https://spark.apache.org/docs/latest/api/java/org/apache/spark/sql/Row.html>

[Update] Or using UDFs in Scala:

```
import org.apache.spark.sql.functions._

val toLong = udf[Long, String] (_.toLong)

val modifiedDf = df.withColumn("modifiedColumnName",
  toLong(df("columnName"))).drop("columnName")
```

and if the column name needs to stay the same you can rename it back:

```
modifiedDf.withColumnRenamed("modifiedColumnName", "columnName")
```

edited Dec 23 '15 at 23:49

answered Nov 8 '15 at 21:19



[radek1st](#)

168 1 9

DataFrames are based on RDDs. RDDs are immutable structures and do not allow updating elements on-site. To change values, you will need to create a new DataFrame by transforming the original one either using the SQL-like DSL or RDD operations like `map`.

A highly recommended slide deck: [Introducing DataFrames in Spark for Large Scale Data Science](#).

edited Feb 24 at 21:56

answered Mar 17 '15 at 21:51

**Jacek Laskowski****15.9k** 4 51 120**maasg****17.7k** 3 41 66

3 What exactly is the dataframe abstraction adding then that couldn't already be done in the same amount of lines with a table? – [Luke](#) Mar 17 '15 at 22:25

" DataFrames introduce new simplified operators for filtering, aggregating, and projecting over large datasets. Internally, DataFrames leverage the Spark SQL logical optimizer to intelligently plan the physical execution of operations to work well on large datasets" - [databricks.com/blog/2015/03/13/announcing-spark-1-3.html](http://databricks.com/blog/2015/03/13/announcing-spark-1-3.html) – [maasg](#) Mar 17 '15 at 22:40

Commonly when updating a column, we want to map an old value to a new value. Here's a way to do that in pyspark without UDF's:

```
# update df[update_col], mapping old_value --> new_value
from pyspark.sql import functions as F
df = df.withColumn(update_col,
    F.when(df[update_col]==old_value,new_value).
    otherwise(df[update_col])).
```

answered Dec 21 '15 at 22:23

**Paul****537** 5 16