# How to change dataframe column names in pyspark?



I come from pandas background and am used to reading data from CSV files into a dataframe and then simply changing the column names to something useful using the simple command:

```
df.columns = new_column_name_list
```

However, the same doesn't work in pyspark dataframes created using sqlContext. The only solution I could figure out to do this easily is the following:

```
df = sqlContext.read.format("com.databricks.spark.csv").options(header='false',
inferschema='true', delimiter='\t').load("data.txt")
oldSchema = df.schema
for i,k in enumerate(oldSchema.fields):
  k.name = new_column_name_list[i]
df = sqlContext.read.format("com.databricks.spark.csv").options(header='false',
delimiter='\t').load("data.txt", schema=oldSchema)
```

This is basically defining the variable twice and inferring the schema first then renaming the column names and then loading the dataframe again with the updated schema.

Is there a better and more efficient way to do this like we do in pandas ?

My spark version is 1.5.0

python apache-spark pyspark pyspark-sql

asked Dec 3 '15 at 22:21

Shubhanshu Mishra
**1,131** 1 8 21

## 2 Answers

There are many ways to do that:

- Option 1. Using selectExpr.

```
data = sqlContext.createDataFrame([("Alberto", 2), ("Dakota", 2)],
                                  ["Name", "askdaosdka"])
data.show()
data.printSchema()

# Output
#+-------+----------+
#|   Name|askdaosdka|
#+-------+----------+
#|Alberto|         2|
#| Dakota|         2|
#+-------+----------+

#root
# |-- Name: string (nullable = true)
# |-- askdaosdka: long (nullable = true)

df = data.selectExpr("Name as name", "askdaosdka as age")
df.show()
df.printSchema()

# Output
#+-------+---+
```

```
#|   name|age|
#+-------+---+
#|Alberto|  2|
#| Dakota|  2|
#+-------+---+

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

- Option 2. Using withColumnRenamed, notice that this method allows you to "overwrite" the same column.

```
oldColumns = data.schema.names
newColumns = ["name", "age"]

df = reduce(lambda data, idx: data.withColumnRenamed(oldColumns[idx], newColumns[idx]),
xrange(len(oldColumns)), data)
df.printSchema()
df.show()
```

- Option 3. using alias, in Scala you can also use as.

```
from pyspark.sql.functions import *

data = data.select(col("Name").alias("name"), col("askdaosdka").alias("age"))
data.show()

# Output
#+-------+---+
#|   name|age|
#+-------+---+
#|Alberto|  2|
#| Dakota|  2|
#+-------+---+
```

- Option 4. Using sqlContext.sql, which lets you use SQL queries on `DataFrames` registered as tables.

```
sqlContext.registerDataFrameAsTable(data, "myTable")
df2 = sqlContext.sql("SELECT Name AS name, askdaosdka as age from myTable")

df2.show()

# Output
#+-------+---+
#|   name|age|
#+-------+---+
```

```
#|Alberto|  2|
#| Dakota|  2|
#+-------+---+
```

edited Jul 10 at 11:08

answered Dec 3 '15 at 22:54

Alberto Bonsanto

**3,555** 2 14 40



```
df = df.withColumnRenamed("colName", "newColName").withColumnRenamed("colName2",
"newColName2")
```

Advantage of using this way: With long list of columns you would like to change only few column names. This can be very convenient in these scenarios. Very useful when joining tables with duplicate column names.

edited Mar 30 at 8:13

David Arenburg

**61.1k** 7 46 98

answered Mar 30 at 7:25

Pankaj Kumar

**56** 1 2