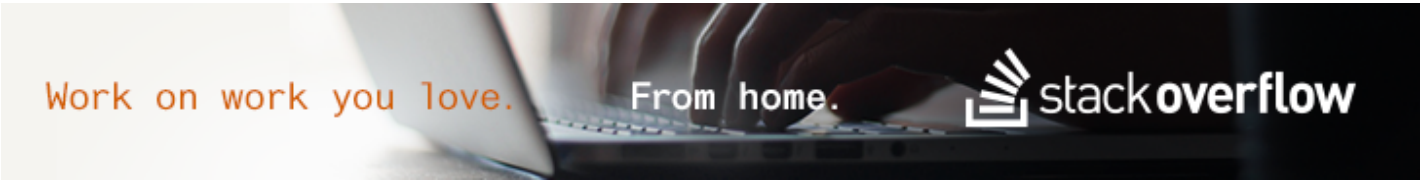
pyspark split a column to multiple columns without pandas



my question is how to split a column to multiple columns. I don't know why `df.toPandas()` does not work.

For example, I would like to change 'df_test' to 'df_test2'. I saw many examples using the pandas module. Is there another way? Thank you in advance.

```
df_test = sqlContext.createDataFrame([
(1, '14-Jul-15'),
(2, '14-Jun-15'),
(3, '11-Oct-15'),
], ('id', 'date'))
```

df_test2

```
id     day    month    year
1       14     Jul      15
2       14     Jun      15
```

```
1       11      Oct       15
```

python apache-spark pyspark apache-spark-sql

edited Dec 18 '15 at 22:04

zero323
66.1k 16 77 137

asked Dec 18 '15 at 19:10

nathanlim45
40 7

1 Answer

It is not possible to derive multiple top level columns in a single access. You can use structs or collection types with an UDF like this:

```
from pyspark.sql.types import StringType, StructType, StructField
from pyspark.sql import Row
from pyspark.sql.functions import udf, col

schema = StructType([
  StructField("day", StringType(), True),
  StructField("month", StringType(), True),
  StructField("year", StringType(), True)
])

def split_date_(s):
    try:
        d, m, y = s.split("-")
        return d, m, y
    except:
        return None

split_date = udf(split_date_, schema)

transformed = df_test.withColumn("date", split_date(col("date")))
transformed.printSchema()

## root
##  |-- id: long (nullable = true)
##  |-- date: struct (nullable = true)
##  |    |-- day: string (nullable = true)
##  |    |-- month: string (nullable = true)
##  |    |-- year: string (nullable = true)
```

but it is not only quite verbose in PySpark, but also expensive.

For date based transformations you can simply use built-in functions:

```
from pyspark.sql.functions import unix_timestamp, dayofmonth, year, date_format

transformed = (df_test
    .withColumn("ts",
        unix_timestamp(col("date"), "dd-MMM-yy").cast("timestamp"))
    .withColumn("day", dayofmonth(col("ts")).cast("string"))
    .withColumn("month", date_format(col("ts"), "MMM"))
    .withColumn("year", year(col("ts")).cast("string"))
    .drop("ts"))
```

Similarly you could use `regexp_extract` to split date string.

See also Derive multiple columns from a single column in a Spark DataFrame

Note:

If you use version not patched against SPARK-11724 this will require correction after `unix_timestamp(...)` and before `cast("timestamp")`.

edited May 3 at 15:15

answered Dec 18 '15 at 22:04

zero323

66.1k 16 77 137