Add column sum as new column in PySpark dataframe

I'm using PySpark and I have a Spark dataframe with a bunch of numeric columns. I want to add a column that is the sum of all the other columns.

Suppose my dataframe had columns "a", "b", and "c". I know I can do this:

```
df.withColumn('total_col', df.a + df.b + df.c)
```

The problem is that I don't want to type out each column individually and add them, especially if I have a lot of columns. I want to be able to do this automatically or by specifying a list of column names that I want to add. Is there another way to do this?

python apache-spark pyspark spark-dataframe

edited Aug 12 '15 at 6:23

Paul
11.7k 7 40 66

asked Aug 12 '15 at 2:59

plam
103 1 12

This is much easier with RDDs than dataframes e.g. if data is an array representing a row, then you can do `RDD.map(lambda data: (data, sum(data)))` . The main reason this is more difficult with a spark dataframe is figuring out what is allowed as a column expression in `withColumn` . It doesn't seem to be very well documented. – Paul Aug 12 '15 at 3:36

1 Answer

This was not obvious. I see no row-based sum of the columns defined in the spark Dataframes API.

Version 2

This can be done in a fairly simple way:

```
newdf = df.withColumn('total', sum(df[col] for col in df.columns))
```

`df.columns` is supplied by pyspark as a list of strings giving all of the column names in the Spark Dataframe. For a different sum, you can supply any other list of column names instead.

I did not try this as my first solution because I wasn't certain how it would behave. But it works.

Version 1

This is overly complicated, but works as well.

You can do this:

1. use `df.columns` to get a list of the names of the columns
2. use that names list to make a list of the columns
3. pass that list to something that will invoke the column's overloaded add function in a fold-type functional manner

With python's reduce, some knowledge of how operator overloading works, and the pyspark code for columns here that becomes:

```
def column_add(a,b):
     return  a.__add__(b)

newdf = df.withColumn('total_col',
         reduce(column_add, ( df[col] for col in df.columns ) ))
```

Note this is a python reduce, not a spark RDD reduce, and the parenthesis term in the second parameter to reduce requires the parenthesis because it is a list generator expression.

Tested, Works!

```
$ pyspark
>>> df = sc.parallelize([{'a': 1, 'b':2, 'c':3}, {'a':8, 'b':5, 'c':6}, {'a':3, 'b':1,
'c':0}]).toDF().cache()
>>> df
DataFrame[a: bigint, b: bigint, c: bigint]
>>> df.columns
['a', 'b', 'c']
>>> def column_add(a,b):
...     return a.__add__(b)
...
>>> df.withColumn('total', reduce(column_add, ( df[col] for col in df.columns )
)).collect()
[Row(a=1, b=2, c=3, total=6), Row(a=8, b=5, c=6, total=19), Row(a=3, b=1, c=0, total=4)]
```

edited Aug 12 '15 at 6:32

answered Aug 12 '15 at 3:55

Paul
11.7k 7 40 66

@Salmonerd Thanks. It helps sometimes to remember the spark dataframe class is immutable, and so to make any changes in the data you have to call something that returns a new dataframe. – Paul Aug 13 '15 at 3:58

1 Version 2 is not working with Spark 1.5.0 and CDH-5.5.2. and Python version 3.4. It is throwing an error : "AttributeError: 'generator' object has no attribute '_get_object_id" – Hemant May 31 at 9:08