UDF, UDAF and UDTF info on SparkSQL and HIVE

--------------------------------------------------------------------

1. <https://blog.cloudera.com/blog/2017/02/working-with-udfs-in-apache-spark/>
2. <https://github.com/bmc/spark-hive-udf/blob/master/src/test/scala/com/ardentex/spark/hiveudf/FormatTimestampSpec.scala>
3. <https://dzone.com/articles/writing-custom-hive-udf-andudaf>
4. <https://snowplowanalytics.com/blog/2013/02/08/writing-hive-udfs-and-serdes/>

1 > add jar /path/to/HiveSwarm.jar;

2 > create temporary function to\_upper as 'com.snowplowanalytics.hive.udf.ToUpper';

And then finally you can use our new UDF in your HiveQL queries, something like this:

1 > SELECT toupper(author\_name) FROM authors a;

2 STEPHEN KING

Note: if we need it temporarily like if we exit from Hive function is lost

Then we can use

**Python:**

df = sqlContext.read.json("temperatures.json")

df.registerTempTable("citytemps")

# Register the UDF with our SQLContext

sqlContext.**registerFunction**("**CTOF**", lambda degreesCelsius: ((degreesCelsius \* 9.0 / 5.0) + 32.0))

sqlContext.sql("SELECT city, **CTOF**(avgLow) AS avgLowF, CTOF(avgHigh) AS avgHighF FROM citytemps").show()

**Scala:**

df = sqlContext.read.json("temperatures.json")

 df.registerTempTable("citytemps")

// Register the UDF with our SQLContext

 sqlContext.**udf.register**("CTOF", (degreesCelcius: Double) => ((degreesCelcius \* 9.0 / 5.0) + 32.0))

sqlContext.sql("SELECT city, CTOF(avgLow) AS avgLowF, CTOF(avgHigh) AS avgHighF FROM citytemps").show()