Transfer data outside the Synapse workspace using the PySpark connector

Note:

You are not required to complete the processes, tasks, activities, or steps presented in this example. The various samples provided are for illustrative purposes only and it's likely that if you try this out you will encounter issues in your system.

You can transfer data to and from a dedicated SQL pool using a Pyspark Connector, which currently works with Scala.

Let's say that you have created or loaded a DataFrame called "pyspark_df", and then assume that you want to write that DataFrame into the data warehouse. How would you go about that task?

The first thing to do is to create a temporary table in a DataFrame in PySpark using the createOrReplaceTempView method

```
1
```

```
pyspark_df.createOrReplaceTempView("pysparkdftemptable")
```

The parameter that is passed through is the temporary table name, which in this case is called: "pysparkdftemptable". We are still using the pyspark_df DataFrame as you can see in the beginning of the statement. Next, you would have to run a Scala cell in the PySpark notebook using magics (since we're using different languages, and it will only work in Scala):

```
1
2
3
```

```
%%spark
val scala_df = spark.sqlContext.sql ("select * from pysparkdftemptable")
scala_df.write.sqlanalytics("sqlpool.dbo.PySparkTable", Constants.INTERNAL)
```

By using "val scala_df", we create a fixed value for the scala_dataframe, and then use the statement "select * from pysparkdftemptable", that returns all the data that we created in the temporary table in the previous step, and storing it in a table named sqlpool.dbo.PySparkTable

In the second line of the code, we specified following parameters:

- **DBName**: The database name that in the example above is named sqlpool
- **Schema**: The schema name that in the example above is named dbo
- **TableName**: The table name that in the example above is named PySparkTable
- **TableType**: Specifies the type of table, which has the value Constants.INTERNAL, which related to a managed table in the dedicated SQL pool.

Should you wish to read data using the PySpark connector, keep in mind that you read the data using scala first, then write it into a temporary table. Finally you use the Spark SQL in PySpark to query the temporary table into a DataFrame.