Exercise 0 - Spark SQL

Instructions for installing docker and running the container are repeated below, but they are the same as for Hive

Install docker and run the container

Ubuntu/Debian

You can install docker using apt: `apt-get install docker.io`.

Other Operating Systems

Go to `https://docs.docker.com/engine/installation/` and follow the instructions for your system.

Run the container

Run the container:

```
$ sudo docker run -d --name dwws --restart always \
-v /home/user/share/:/home/student/share/ acrrd/dw-workstation
```

where `/home/user/share` is the absolute path to the directory you want to use to share data with the container. The command download the image (∼1.3Gb) if not already present on the system. You have to run this command only once. You can use: `sudo docker ps -a` to see the active containers. You can use: `sudo docker start dwws` to start the container and `sudo docker stop dwws` to stop it.
The container embeds a working version of HIVE installed and it runs an ssh server.
We just need to know its ip address to connect to it, the command `sudo docker inspect dwws` output information about the container, search for `"IPAddress"` to find the ip.

```
$ sudo docker inspect dwws | grep \"IPAddress\"
   "IPAddress": "172.17.0.2",
           "IPAddress": "172.17.0.2",
```

We can login on the container using ssh with user `student` and password `foobar`:
`ssh student@172.17.0.2`.
If the connection does not work try to start the container.
On your host machine, move the two files `earthquakes.csv` and `GlobalSuperstoreOrders2016.csv` in the directory you want to share with the container.
On the container, runing `ls share` you should be able to see the two files above.
If you delete the container you loose all the data except the one in `/home/user/share`.
In case you need it, root password is `toor`.
To launch the HIVE shell we just need to run the command `hive`.

Import local data into Spark SQL DataFrames

From csv

We can create a Spark SQL Data Frame from the earthquakes.csv by relying on the `csv` method of the `DataFrameReader` class (the use is analogous to `json` for loading from JSON):

```
.csv("/home/student/share/earthquakes.csv")
```

We can make use of the `schemaInfer` option to infer the schema of the DataFrame. E.g.:

```
Dataset<Row> csvDF = spark.read()
               .format("csv")
               .option("header", "true")
               .option("schemaInfer","true")
               .option("nullValue", "")
               .csv("/home/student/share/earthquakes.csv")
               ;
```

Similarly, import the content of `GlobalSuperstoreOrders2016.csv` into a new table and check its content. We can make use of `show`, `printSchema` and `count` DataFrame operations.

From a PostgreSQL table

We can create a database with

```
$ createdb -U student dbname
```

and open the PostgreSQL client with

```
$ psql -U student dbname
```

We can access postgresql as user `'student'` password `'foobar'`.
Program `SparkSql.java` contains a small example that loads a table from PostgreSQL. We can compile it as

```
javac -cp \
 /opt/spark-2.0.1-bin-hadoop2.7/jars/spark-core_2.11-2.0.1.jar: \
 /opt/spark-2.0.1-bin-hadoop2.7/jars/spark-sql_2.11-2.0.1.jar: \
 /opt/spark-2.0.1-bin-hadoop2.7/jars/spark-hive_2.11-2.0.1.jar: \
 /opt/spark-2.0.1-bin-hadoop2.7/jars/scala-library-2.11.8.jar: \
 /opt/spark-2.0.1-bin-hadoop2.7/jars/spark-catalyst_2.11-2.0.1.jar
sparksql/SparkSql.java

jar cf sparksql.jar sparksql/
```

and execute it as

```
spark-submit \
--driver-class-path /home/student/postgresql-9.4.1212.jre6.jar \
--jars /home/student/postgresql-9.4.1212.jre6.jar \
--class sparksql.SparkSql.java sparksql.jar
```

Create and populate some tables and import their content. Check resulting DataFrame schema and content making use of `show`, `printSchema` and `count` operations.

Querying DataFrames

Query the above created DataFrames by means of `select`, `filter`, `distinct` and other RDD operations, as well as by registering the tables and making use of the `sql` operation.
Specifically, with reference to the order database, submit the following queries:

1. Select the orders with High Order Priority.
2. Select the orders that were placed from november 2014 to february 2015.
3. Select the five most profitable orders from the previous query.