Hadoop Training Apache Spark SQL

Playing With Key Features Of Spark SQL

In this series of exercise we practice SQL-related features of Spark such as SQL-like queries and DataFrame - both on files in HDFS and tables in Hive.

Getting ready

1.Login using ssh to edgenode cdh00.cl.ii.pw.edu.pl using your Linux account.

2.Go to home directory, create a data folder and copy sample Parquet file. (You can have a look at this file):

```
cp /data/local/datascience/data/users.parquet
/data/local/datascience/home/${USER}/data


cd /data/local/datascience/home/${USER}/data

head users.parquet

hdfs dfs -put users.parquet /user/${USER}

hdfs dfs -ls /user/${USER}
```

3. Start the spark-shell:

```
spark-shell --master yarn --deploy-mode client --executor-memory
2048m --num-executors 4 --conf spark.ui.port=95<YOUR-NUMBER>
```

DataFrame Operations On A Parquet File In HDFS

4. Now, load the Parquet file directly into SQL context:

```
scala> sc.setLogLevel("INFO")
scala> val user_df =
spark.sqlContext.read.parquet("/user/${USER}/users.parquet")
```

You can use the show() and printSchema() operators to preview the sample of the data and see the schema.

```
scala> user_df.show()

+---+---------+-------+------+----------+-----------+----------------+
| id|    fname|  lname|gender| birthdate|      state|    registration|
+---+---------+-------+------+----------+-----------+----------------+
|  1|    EDWIN|  BATES|     M|1965-03-14|   Colorado|      2013-12-25|
|  2| ISABELLE|  HEATH|     F|1999-09-12|     Oregon|      2013-05-15|
|  3|   ANNIKA|FREEMAN|     F|1967-09-23|   Michigan|      2013-09-22|
|  4|VALENTINA| LARSON|     F|1951-07-13|  Wisconsin|      2013-11-07|
...
| 19|   JIMENA|NAVARRO|     F|1985-04-12|  Minnesota|      2013-11-27|
| 20|   AUBREY|JOHNSON|     F|1965-04-18|Pennsylvania|      2013-10-25|
+---+---------+-------+------+----------+-----------+----------------+
only showing top 20 rows

scala> user_df.printSchema()

root
 |-- id: integer (nullable = true)
 |-- name: string (nullable = true)
 |-- surname: string (nullable = true)
 |-- gender: string (nullable = true)
 |-- bday: string (nullable = true)
 |-- state: string (nullable = true)
 |-- registration: string (nullable = true)
```

Show only one row of the dataset.

The DataFrame API provides a set of useful operators that you can use to manipulate your dataset. Go ahead and try to run few of them!

```
scala> user_df.select("name").show()

scala> user_df.select("name", "gender").show()

scala> user_df.select("gender").distinct().show()

scala> user_df.select("name").distinct().count()

scala> user_df.select("name").take(1)
```

As we reuse the same dataset over and over, let’s cache it.

```
scala> val user_df_cached = user_df.cache()
```

Spark caches datasets lazily. It is cached in the executors' memory the first time we access a given dataset. To warm up the cache, run simple command:

```
scala> user_df_cached.select("name").take(1)
```

From now on, you will be processing cached data. Run the same command again to see that this time we read our data from the memory:

```
scala> user_df_cached.select("name").take(1)
```

To practice a bit, run following lines of the code and don't be afraid of modifying them to experiment :

```
scala> user_df_cached.groupBy("state").count().show()

scala> user_df_cached.groupBy("state").count().filter("count > 50").show()

scala> user_df_cached.groupBy("state").count().orderBy("count").show()

scala>
user_df_cached.groupBy("state").count().orderBy(desc("count")).show()

scala>
user_df_cached.groupBy("state","gender").count().orderBy(asc("state")).show
()

scala> user_df_cached.groupBy("state").agg(min("bday")).show()
```

Feel free to investigate the API and try more operators.

Spark SQL Operations On Parquet File

To use Spark SQL statements on a Parquet file, we have to register it at first.

```
scala> user_df.createOrReplaceTempView("user")
```

Now you can query the user table using SQL statements:

```
scala> val count_by_gender = sql("select gender, count(*) as cnt from user
group by gender order by cnt desc")
```

```
16/02/22 15:09:38 INFO ParseDriver: Parsing command: select gender,
count(*) as cnt from user group by gender order by cnt desc
16/02/22 15:09:40 INFO ParseDriver: Parse Completed
```

Print the results using the regular Spark non-SQL code:

```
scala> count_by_gender.show()
+------+---+
|gender|cnt|
+------+---+
|     M|553|
|     F|448|
+------+---+
```

Hurray! It works :)

Write the following query:

a) **Return the state which had highest number of woman registered.**

Spark SQL in Hive Context

It is high time to query some tables directly from Hive!

```
scala> val df = spark.sqlContext.sql("SELECT * FROM
datascience.measured_data_orc")
scala> df.limit(5).show()

+---------+------------------+----------+--------------------+-------+----
-------+--------------+----------------+
|md_lsb_id|       md_timestamp|md_lsb_id2|
md_value|md_unit|md_timetype|md_quality_mark|         md_desc|
+---------+------------------+----------+--------------------+-------+----
-------+--------------+----------------+
|        1|2015-05-18 19:24:00|       236| 0.09920929584628235|     kW|
s|               D|Warsaw-Dereniowa|
|        2|2015-05-18 19:24:00|       237|  0.8467751295230832|     kW|
s|               D|Warsaw-Dereniowa|
|        3|2015-05-18 19:24:00|       238|  0.4608953190788161|     kW|
s|               D|Warsaw-Dereniowa|
|        4|2015-05-18 19:24:00|       239|0.029231154861803166|     kW|
s|               D|Warsaw-Dereniowa|
|        5|2015-05-18 19:24:00|       240|  0.7698375647136683|     kW|
s|               D|Warsaw-Dereniowa|
+---------+------------------+----------+--------------------+-------+----
-------+--------------+----------------+
```

Write the following query:

a) **Return the number (count) of observations and average value (MD_VALUE) for each observation timestamp (MD_TIMESTAMP)**

Analysis of Iris data:

a) Download Iris data from UCI database:
https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data
to CDH00.
b) Put the file into HDFS.
c) Create external table following the schema described in:
https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.names
d) Write SQL query which will answer the following question:

- For each class of irises, find the average sepal length.

e) Find the way to read csv data into spark, create spark dataframe and create an SQL query from point 4.