MIDS W205

Lab #	3	Lab Title	Defining Schema and Basic Queries with Hive and Spark
Related Module(s)	1-4	Goal	Understanding schema and query engines
Last Updated	1/25/16	Expected duration	40-60 minutes

Introduction:

In this lab we will be using the pseudo-distributed Big Data environment created in Lab 2 and learn to process data with it. We will use SQL, a query language, to create define multiple schema on a data set. Then, we will use 2 SQL-based processing engines, Apache Hive and Apache Spark-SQL, to explore the data and compare execution. In this lab, we will learn about the following:

- How do define schema on data stored in HDFS
- How to create tables with Apache Hive and Spark-SQL
- How to interactively use SQL with the Hive Command Line Interface (CLI)
- How to interactively use SQL with the Spark-SQL Command Line Interface (CLI)

You may have previous experience with SQL. However, many SQL engines have slightly different syntax. As such, it is useful to refer to the SQL documentation for Apache Hive, as it is shared between Hive and Spark-SQL. In the below table you can find links to useful and necessary resources discussed in this lab.

Resource	What
https://cwiki.apache.org/confluence/display/Hive/LanguageManual	Hive Language Manual
http://spark.apache.org/docs/1.5.2/sql-programming-guide.html	Spark SQL Guide

Step-1. Download Data and Place In HDFS

We need some data in order to create schema and, ultimately, process. The data we'll consider is a toy dataset regarding users and their weblogs. To download the data, do this:

1. Launch an instance of UCB W205 Spring 2016
    a. Attach your EBS volume from Lab 2
    b. Find the volume location, by typing `fdisk -l`

c. Mount the volume as follows: `mount -t ext4 /dev/<your device> /data`

d. Start HDFS, Hadoop Yarn and Hive: `/root/start-hadoop.sh`

e. Start Postgres: `/data/start_postgres.sh`

f. Change to the w205 user: `su - w205`

g. Make a new folder in HDFS for this lab: `hdfs dfs -mkdir /user/w205/lab_3`

h. Download the two datasets using wget. Type:

   i. `wget https://s3.amazonaws.com/ucbdatasciencew205/lab_datasets/userdata_lab.csv`

   ii. `wget https://s3.amazonaws.com/ucbdatasciencew205/lab_datasets/weblog_lab.csv`

i. Make an HDFS folder for each data set and place them in HDFS

   i. `hdfs dfs -mkdir /user/w205/lab_3/user_data`

   ii. `hdfs dfs -mkdir /user/w205/lab_3/weblog_data`

   iii. `hdfs dfs -put userdata_lab.csv /user/w205/lab_3/user_data`

   iv. `hdfs dfs -put weblog_lab.csv /user/w205/lab_3/weblog_data`

Step-3. Define Schema for The Data in Hive

Now that the data is in HDFS, we'd like to define schema on it. We'll start by creating and querying a simple table. Then we'll add schema for both weblogs and users.

First, enter the Hive CLI by typing: `hive`

We are now in the Hive interactive environment. We can use this environment to explore and integrate the data we've placed in HDFS. Let's start by defining a flat, undelimited schema over our weblogs. In the CLI, type:

```
CREATE EXTERNAL TABLE IF NOT EXISTS weblogs_flat
(weblog string)
ROW FORMAT DELIMITED
STORED AS TEXTFILE
LOCATION '/user/w205/lab_3/weblog_data';
```

Now we can access the weblogs interactively. Type:

```
SELECT * FROM weblogs_flat LIMIT 10;
```

You'll notice 10 weblogs return. We can filter out the header row with a query like this:

```
SELECT * FROM weblogs_flat WHERE weblog NOT LIKE 'date%' LIMIT 10;
```

However, we can't select individual fields or filter our results with very much nuance. To do that, we need to add a more detailed schema. Define a new table in the CLI as follows:

```
CREATE EXTERNAL TABLE IF NOT EXISTS weblogs_schema
(datetime string,
user_id string,
session_id string,
product_id string,
referrer string)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
STORED AS TEXTFILE
LOCATION '/user/w205/lab_3/weblog_data';
```

Now we can select out just fields we may be interested in. For example, we can count the 50 most frequently occurring user_ids as follows:

```
SELECT user_id, COUNT(user_id) AS log_count
FROM weblogs_schema GROUP BY user_id
ORDER BY log_count DESC
LIMIT 50;
```

Additionally, define a table on our user information. Create a table as follows:

```
CREATE EXTERNAL TABLE IF NOT EXISTS user_info
(
datetime string,
user_id string,
first_name string,
last_name string,
location string
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
STORED AS TEXTFILE
LOCATION '/user/w205/lab_3/user_data';
```

Exit the Hive CLI by typing: `exit;`

Step-3. Setup Spark, Use the SparkSQL CLI

As we explore different manifestations of processing, we'll pay special attention to Apache Spark. Spark can process SQL both programmatically, and through a CLI similar to Hive. First, we'll set up Spark via a script. This script both sets up Spark, but also creates simple scripts to

start and stops Hive's "Metastore," which provides a common repository of schema for multiple processing environments.

1. Download the setup script by running:

   ```
   wget https://s3.amazonaws.com/ucbdatasciencew205/setup_spark.sh
   ```

2. Run the script by typing: `bash ./setup_spark.sh`

3. Start the Hive metastore. Type: `/data/start_metastore.sh`

4. Start the SparkSQL CLI. Type: `/data/spark15/bin/spark-sql`

5. Check to see if the previously created tables are present. Type: `show tables;`

6. Run the aggregated query from the previous step. Compare the execution time:

   ```
   SELECT user_id, COUNT(user_id) AS log_count
   FROM weblogs_schema GROUP BY user_id
   ORDER BY log_count DESC
   LIMIT 50;
   ```

7. Convert the weblogs data to Parquet format:

   ```
   CREATE TABLE weblogs_parquet AS SELECT * FROM weblogs_schema;
   ```

8. Run the aggregation on the new table and compare the execution time.

   ```
   SELECT user_id, COUNT(user_id) AS log_count
   FROM weblogs_parquet GROUP BY user_id
   ORDER BY log_count DESC
   LIMIT 50;
   ```

9. Exit the CLI. Type: `exit;`

Submissions:

1- List the execution time of the weblog aggregation query for Hive, SparkSQL, and SparkSQL on Parquet.
2- How many jobs does Hive launch? Does SparkSQL launch jobs?

3- Write a query which joins weblogs_parquet to user_info and counts the top 5 locations. List the locations.