# MIDS W205

|  |  |  |  |
| --- | --- | --- | --- |
| *Lab* | 3 | *Lab Title* | Defining Schema and Basic Queries with Hive and Spark |
| *Related Modules(s)* | 1-4 | *Goal* | Understanding AWS / AMI concepts |
| *Last Updated* | 1/22/17 | *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. If you encounter any issues sometimes it may be pertinent to redo Lab 2 instead of trying to debug the issue as any number of small details could disrupt the system. Having an ephemeral infrastructure that you can rebuild quickly can lead to increased reliability of systems. To download the data, do this:

1. Launch an instance of UCB W205 Spring 2016
2. Attach your EBS volume from Lab 2
3. Find the volume location, by typing fdisk -l
4. Mount the volume as follows: mount -t ext4 /dev/<your device>/data
5. Start HDFS, Hadoop Yarn and Hive: /root/start-hadoop.sh
6. Start Postgres: /data/start\_postgres.sh
7. Change to the w205 user: su - w205
8. Make a new folder in HDFS for this lab: hdfs dfs -mkdir /user/w205/lab\_3
9. Download the two datasets using wget. Type:

* - wget https://s3.amazonaws.com/ucbdatasciencew205/lab\_datasets/userdata\_lab.csv  
    
  - wget https://s3.amazonaws.com/ucbdatasciencew205/lab\_datasets/weblog\_lab.csv
  1. Make an HDFS folder for each data set and place them in HDFS
     1. hdfs dfs -mkdir /user/w205/lab\_3/user\_data
     2. hdfs dfs -mkdir /user/w205/lab\_3/weblog\_data
     3. hdfs dfs -put userdata\_lab.csv /user/w205/lab\_3/user\_data
     4. hdfs dfs -put weblog\_lab.csv /user/w205/lab\_3/weblog\_data

## Step-2. 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';

Note that '' is a tab character. 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;

Let us additionally, define a table for 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

1. Run the script by typing: bash ./setup\_spark.sh
2. Start the Hive metastore. Type: /data/start\_metastore.sh
3. Start the SparkSQL CLI. Type: /data/spark15/bin/spark-sql
4. Check to see if the previously created tables are present. Type: show tables;
5. 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;

1. Convert the weblogs data to Parquet format:

* CREATE TABLE weblogs\_parquet AS SELECT \* FROM weblogs\_schema;

1. 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;

1. 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.