Lab # : 4; Lab Name : Apache Spark SQL - a few features; Subject Name : Information Storage and Retrieval; Week #: 3; Lab Duration : 20 to 30 mins

# 

# **Intro**

## SparkSQL

Spark SQL is a Spark module for which is used for structured data processing. It provides a programming abstraction called DataFrames. Spark SQL acts as a distributed SQL query engine.

In Spark, a DataFrame is a distributed collection of data organized into named columns. It is conceptually equivalent to a table in a relational database or a data frame in R/Python, but with richer optimizations. DataFrames can be constructed from a wide array of sources such as: structured data files, tables in Hive, external databases, or existing RDDs.

##### **Here is an example which shows how to construct DataFrames in Python programming language :**

//Constructs a DataFrame from the users table in Hive.

users = context.table(“users”)

//Constructs a DataFrame from from JSON files in S3

logs = context.load(“s3n://path/to/data.json”, “json”)

##### **Here are a few supported Data Formats and Sources :**

1. JSON files, Parquet files, Hive tables  
 2. Local file systems  
 3. Distributed file systems (HDFS)  
 4. Cloud storage (S3)   
 5. External relational database systems via JDBC  
 6. Extensions include Avro, CSV, ElasticSearch, and Cassandra

In this lab, we will go over Apache Spark features and common commands as below:

# 

# Accessing Apache Spark - CLI

# Accessing through Java database connectivity (JDBC)

# Accessing through Java database connectivity - Beenline

# Lets create a table and load data using CSV file

# Accessing Spark-SQL in Python Code

# Caching tabels and Un-caching tables

# 

# **Let’s go!**

# **Step-1.Accessing Apache Spark SQL (CLI)**

# Let’s access through command line interface (CLI)

From Spark Installation folder

cd /mnt/softwares/spark-1.3.0

./bin/spark-sql

//In cloudera CDH you can find here

(/opt/cloudera/parcels/CDH-5.4.0-1.cdh5.4.0.p0.27/lib/spark/bin/ )

Once you are in spark prompt, to list the tables, you can run as follows:

show tables;

##### 

# 

# **Step-2. Accessing through Java database connectivity (JDBC)**

# 

Spark SQL provides JDBC connectivity, which is useful for connecting business intelligence (BI) tools to a spark cluster and other users or applications.

**Here are the steps to launch a the Spark SQL JDBC server**

From Spark Installation folder

cd /mnt/softwares/spark-1.3.0

./sbin/start-thriftserver.sh –master sparkMaster

# 

# **Step-3. Let’s connect to the above server through Beenline:**

From Spark Installation folder

cd /mnt/softwares/spark-1.3.0

//10000 will be the default thrift server port

./bin/beenline -u jdbc:hive2://localhost:10000

0: jdbc:hive2://localhost:10000> show tables;

# **Step-4. Lets create a table and load data using CSV file:**

You can create a Spark SQL table using the following format.

create table Web\_Session\_Log

(DATETIME varchar(500),

USERID varchar(500),

SESSIONID varchar(500),

PRODUCTID varchar(500),

REFERERURL varchar(500))

row format delimited fields terminated by ‘\t’

stored as textfile;

Now let’s load data to this table from HDFS and Local:

HDFS :

LOAD DATA INFILE ‘/mnt/weblog.csv’ INTO TABLE Web\_Session\_Log;

Local file system :

LOAD DATA LOCAL INFILE ‘/mnt/weblog.csv’ INTO TABLE Web\_Session\_Log;

# 

# 

# 

# **Step-5. Accessing Spark-SQL in Python Code:**

Here is a sample code:

from pyspark.sql import SQLContext

sqlContext = SQLContext(sc)

Running SQL Queries Programmatically :

from pyspark.sql import SQLContext

sqlContext = SQLContext(sc)

df = sqlContext.sql(“SELECT \* FROM Web\_Session\_Log limit 10”)

Programmatically Specifying the Schema :

from pyspark.sql import \*

sqlContext = SQLContext(sc)

lines = sc.textFile(“/mnt/weblog.csv”)

parts = lines.map(lambda l: l.split(“\t”))

Web\_Session\_Log = parts.map(lambda p: (p[0], p[1],p[2], p[3],p[4]))

schemaString = “DATETIME USERID SESSIONID PRODUCTID REFERERURL”

fields = [StructField(field\_name, StringType(), True) for field\_name in schemaString.split()]

schema = StructType(fields)

schemaPeople = sqlContext.createDataFrame(Web\_Session\_Log, schema)

schemaPeople.registerTempTable(“Web\_Session\_Log”)

results = sqlContext.sql(“SELECT USERID FROM Web\_Session\_Log”)

names = results.map(lambda p: “Name: ” + p.USERID)

for name in names.collect():

print name

How to Run This Python Code:

You can just launch pyspark from Spark installation bin directory and paste above code.

# 

# **Step-6. Caching tables and Un-caching tables;**

CACHE TABLE and UNCACHE TABLE statements are available to do the above making it very easy.

To Cache a table :

CACHE TABLE logs\_last\_month;

To UnCache a table :

UNCACHE TABLE logs\_last\_month;