Lab # : 4; Lab Name : Hive 2 - a few features; Subject Name : Information Storage and Retrieval; Week #: 4; Lab Duration : 20 to 30 mins

# **Intro**

In this lab, we will go over a few more Hive features and commands, which are useful for managing data in Hive. We will go over the following features:

* Partitioning a Table
* Bucketing a Table
* Check Storage format of a Hive table
* Hive on MR vs Tez
* Hive Views
* User Defined Functions

Here are a few points to get to know Hive:

* Hive can organize data in tables in partitions based on chosen column/s in a table. Partitioning a table creates multiple HDFS folders for the respective partitioned data.
* Buckets in a hive table are individual files physically.
* Bucketing can be done along with Partitioning on Hive tables and even without partitioning.
* Hive lets checking metadata of a Hive table
* Hive on Tez is a new feature in Hive, which runs faster than Hive alone on MR.
* Hive views could be created to filter data. Even UDF could be applied to views.
* For creating analytical operators, you can create custom User Defined Functions in java, python and other languages.

NOTE: All the codes in labs are samples only. You can always use variations of the code as applies. The intention of these labs, are to get you familiar with these concepts at physical level. However, these are vast areas to learn within a few minutes.

# **Let’s go!**

# **Step-1.PARTITIONED Table**

# Partitions are horizontal slices of data, which allow large sets of data to be segmented into more manageable blocks. Partitioning creates folder at HDFS level.

# 

# CREATE TABLE Web\_Session\_Log\_Partitioned

# (

# DATETIME varchar(500), USERID varchar(500), SESSIONID varchar(500),

# PRODUCTID varchar(500), REFERERURL varchar(500))

# COMMENT 'This is the Twitter streaming data'

# PARTITIONED BY(DT STRING)

# ROW FORMAT DELIMITED

# FIELDS TERMINATED BY '\t'

# STORED AS TEXTFILE;

# Now, let’s load data into the same table.

# -- from the table you created in your previous lab; in case you don’t have, you need to recreate.

# FROM Web\_Session\_Log

# INSERT OVERWRITE TABLE Web\_Session\_Log\_Partitioned PARTITION (DT="2014-01-02 00:00:06 GMT") SELECT \*;

# Now, please check the folder and files for the table in HDFS. You might have to drill down to locate the folder.

# **Step-2. Bucketing a table**

# 

# Bucketing is a technique that allows you to cluster or segment large sets of data to optimize query performance.

# 

hive> CREATE TABLE Web\_Session\_Log\_Bucket

> (DATETIME varchar(500),

> USERID varchar(500),

> SESSIONID varchar(500),

> PRODUCTID varchar(500),

> REFERERURL varchar(500))

> COMMENT 'This is the Web Session Log data'

PARTITIONED BY( PRODUCTID STRING)

> CLUSTERED BY(USERID) INTO 2 BUCKETS ROW FORMAT DELIMITED

> FIELDS TERMINATED BY '\t'

# > STORED AS TEXTFILE;

# set hive.enforce.bucketing = true;

# FROM Web\_Session\_Log -- the table you created in your previous lab

# INSERT OVERWRITE TABLE Web\_Session\_Log\_Bucket PARTITION (PRDID="/product/MT65XF2YA")

# SELECT \*;

# Now, please check the folder and files for the table in HDFS. Do you see any difference? Please refer to Hive documentation, to further read on bucketing and partitioning as these are very much used in any hive implementation.

# 

# **Step-3. Let’s check an existing table about it’s table properties.**

describe Web\_Session\_Log;

Here is the sample output:

…….

datetime varchar(500)

userid varchar(500)

sessionid varchar(500)

productid varchar(500)

refererurl varchar(500)

Time taken: 0.111 seconds, Fetched: 5 row(s)

Here is another way to do the same for more details:

describe **formatted** Web\_Session\_Log;

col\_name data\_type comment

datetime varchar(500)

userid varchar(500)

sessionid varchar(500)

productid varchar(500)

refererurl varchar(500)

Detailed Table Information

Database: default

Owner: ubuntu

CreateTime: Thu May 28 06:11:32 UTC 2015

LastAccessTime: UNKNOWN

Protect Mode: None

Retention: 0

Location: hdfs://ip-10-85-31-243.eu-west-1.compute.internal:8020/user/hive/warehouse/web\_session\_log

Table Type: MANAGED\_TABLE

Table Parameters:

COLUMN\_STATS\_ACCURATE true

numFiles 1

numRows 0

rawDataSize 0

totalSize 4513792

transient\_lastDdlTime 1432793495

Storage Information

SerDe Library: org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe

InputFormat: org.apache.hadoop.mapred.TextInputFormat

OutputFormat: org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat

Compressed: No

Num Buckets: -1

Bucket Columns: []

Sort Columns: []

Storage Desc Params:

field.delim \t

serialization.format \t

Time taken: 0.1 seconds, Fetched: 36 row(s)

Please notice above Input and Output formats. These are all customizable as well.

# **Step-4. Let’s join two tables.**

# In Hive, you can do various kinds of joins like, inner join, left outer join, right outer join, etc.

# Here is an example query. You could create two tables of your choice and join them.

# SELECT Web\_Session\_Log.DATETIME,Web\_Session\_Log.USERID,User\_Data.FIRSTNAME,User\_Data.LASTNAME,User\_Data.LOCATION,Web\_Session\_Log.PRODUCTID,Web\_Session\_Log.REFERERURL from Web\_Session\_Log JOIN User\_Data ON (User\_Data.USERID=Web\_Session\_Log.USERID);

# 

# **Step-5. Hive on Tez.**

# Tez is a new application framework built on Hadoop Yarn that can execute complex directed acyclic graphs of general data processing tasks. In many ways it can be thought of as a more flexible and powerful successor of the map-reduce framework.

# 

# Here is a way to set Tez Environment Variable on hive

# set hive.execution.engine=tez;

# you can change back to MR

# set hive.execution.engine=mr;

# Please read further on this topic using your hive site. Hive on Tez is very much used in HDP version of Hadoop from Hortonworks.

# 

# **Step-6. UDF**

Let’s write a simple udf function in python as follows.

Streaming.py code:

import sys

from datetime import datetime

for line in sys.stdin.readlines():

boolVal = “false”

line = line.strip()

DATETIME = datetime.strptime(line, “%m/%d/%Y”)

print DATETIME

Now, let’s register the Python function in hive; You can run this command in hive prompt as follows:

add file streaming.py;

Now, you can use the UDF as follows:

create table dev\_schema.rpt\_asset\_extract as

select TRANSFORM(DATETIME) USING ‘streaming.py’ AS DATETIME from YOURTABLE;

You can try this in one of your tables you created earlier.

**Questions/For your further reading/research:**

Please use your reference materials like Apache site, Cloudera site for further research.

Q1: How would partitioning a table help?

Q2: Why buckets are created or used?

Q3: Using Hive on Tez feature, would it help?

Q4: What is a DAG?

Q5: Why do u need to register an UDF before using?