BDT – cs523

Lab 6 - Hive

- Submit your *own work* on time. No credit will be given if the lab is submitted after the due date.
- Note that the completed lab should be submitted in .doc, .docx, .rtf or .pdf format only.

This document is divided into two parts.

1. **Demo**

   You can try to run through all the steps and see if they work properly for you. No need to submit anything for this part.

   We'll look at the following things:

   - Create Database and tables
   - Observe Warehouse and Metastore
   - Run some more Hive Commands

2. **Homework**

   You need to submit a pdf document (not .zip) wherein I should be able to find all the instructions and commands for completing this part.

   Paste screenshots wherever applicable.

3. **Optional Homework – Hive with Avro**

   Create your own example to understand how Avro works with Hive. Take help from this link:

   https://cwiki.apache.org/confluence/display/Hive/AvroSerDe

Hive Demo Lab

We'll do some analysis on the weather dataset.

1. Get the given simplified version of NCDC sample weather dataset and put the file in the path '/home/cloudera/cs523/input/SampleWeatherForHive.txt'

2. Now we'll load this data into Hive's managed storage. Here we'll have Hive use the local filesystem for storage; later we'll see how to store tables in HDFS. So follow the steps below.

   i. Just like in RDBMS, Hive organizes its data into tables. We create a table to hold the weather data using the CREATE TABLE statement:

   ```
   CREATE TABLE records (year STRING, temperature INT, quality INT)
   ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';
   ```

   The first line declares a records table with three columns: year, temperature, and quality. The type of each column must be specified, too. Hive expects there to be three fields in each row, corresponding to the table columns, with fields separated by tabs and rows by newlines (as given in the command).
   After executing this create table statement, check the Metastore and the Warehouse directory and verify that "records" is created.

   ii. Next, populate this "records" Hive table with the sample weather data.

   ```
   LOAD DATA LOCAL INPATH
   '/home/cloudera/cs523/input/SampleWeatherForHive.txt'
   OVERWRITE INTO TABLE records;
   ```

   Running this command tells Hive to put the specified local file in its warehouse directory. There is no attempt, for example, to parse the file and store it in an internal database format, because Hive does not mandate any particular file format.

   Tables are stored as directories under Hive's warehouse directory, which is controlled by the *hive.metastore.warehouse.dir* property and defaults to `/user/hive/warehouse`.

   Thus, the files for the `records` table are found in the */user/hive/warehouse/records* directory on the local filesystem:
   In this case, there is only one file, `SampleWeatherForHive.txt`, but in general there can be more, and Hive will read all of them when querying the table.

iii. Think : How will you change the above commands if you want to load the data from HDFS directory or if you want to create an external table?

3. Now that the data is in the Hive table, we can run a query against it:

```
hive> SELECT year, MAX(temperature) FROM records WHERE temperature !=
9999 AND quality IN (0, 1, 4, 5)  GROUP BY year;
```

This SQL query is a SELECT statement with a GROUP BY clause for grouping rows into years, which uses the MAX aggregate function to find the maximum temperature for each year group.

The remarkable thing is that Hive transforms this query into a job, which it executes on our behalf, then prints the results to the console. It is the ability to execute SQL queries against our raw data that gives Hive its power.

4. Now delete the above created table and check warehouse directory, what do you observe?

```
hive> DROP TABLE records;
```


5. Now repeat the above steps but this time create records as "EXTERNAL" table in hive. Note down what happens when you delete this external table.

6. No need to submit anything for this "demo lab".

Hive Homework Lab

In this HW lab, you need to analyze the city of Chicago employees' dataset.

1) [5] **Put the given dataset (ChicagoEmployeesDataset.csv) in HDFS. Create EXTERNAL table in Hive for this dataset.**

2) [3] **Find out some interesting facts from this data.** For example, what's the maximum salary of employees in each department, etc.

Show at least 3 different analysis with "limited" rows.

3) [2] **Create dynamic partitions based on the "Salary or Hourly" column.**

What to submit :

1. All the HQL statements used to complete the above requirements.
2. Screenshots of the step-by-step process also showing the output.

Important considerations for this HW

- Correct table creation is crucial for Hive data processing.
- Use a SerDe to handle header rows and commas in names.
- Do not create 2 separate columns for "name".
  You are not allowed to modify the given .csv file!
- If you plan to use a special SerDe then there might be an issue with datatypes. Check datatypes using **DESCRIBE** and adjust if necessary.
- Your created table should work with any valid SQL query.