Hive Installation and Configuration

This document explains the procedure to setup hive. You are expected to know basic UNIX commands and VI editor commands. If you are not familiar with UNIX and vi commands, you should brush up your UNIX basics before proceeding

You need perform the steps (execute the commands) marked only in this color.

If you have followed my earlier tutorials of setting up Hadoop environment, you should have an hduser created already and we will continue to install and configure hive on it.

Requirements

- Java 1.6
- Hadoop 1.0.4.

1. Installing Hive from a Stable Release

Start by downloading the most recent stable release of Hive from one of the Apache download mirrors

You can download manually and place the tarball on your home directory or you can use wget as follows.

Open a terminal and go to your home directory and give the following command

```
   hduser@master:~$ wget http://archive.apache.org/dist/hive/hive-
0.9.0/hive-0.9.0.tar.gz
```

You will get the hive-0.9.0.tar.gz file will be saved on to your home directory.

Next you need to unpack the tarball. This will result in the creation of a subdirectory named hive-x.y.z:

```
hduser@master:~$ tar -xzf hive-0.9.0.tar.gz
```

```
hduser@master:~$ ln -s hive-0.9.0 hive
```

Set the environment variable HIVE_HOME to point to the installation directory (optional, but recommended).

```
hduser@master:~$ export HIVE_HOME=/home/hduser/hive
```

```
hduser@master:~$ export PATH=$PATH:/home/hduser/hive/bin/
```

2. Running Hive

Hive uses Hadoop that means:

- you must have Hadoop in your path OR
- export HADOOP_HOME=<hadoop-install-dir>

In addition, you must create /tmp and
/user/hive/warehouse (aka
hive.metastore.warehouse.dir) and set them chmodg+w
in HDFS before a table can be created in Hive.

Commands to perform this setup

```
hduser@master:~$ hadoop fs -mkdir /tmp
hduser@master:~$ hadoop fs -mkdir /user/hive/warehouse
hduser@master:~$ hadoop fs -chmod 765 /tmp
hduser@master:~$ hadoop fs -chmod 765 /user/hive/warehouse
```

3. Configuration management overview

- Hive default configuration is stored in <install-dir>/conf/hive-default.xml Configuration variables can be changed by (re-)defining them in <install- dir>/conf/hive-site.xml
- The location of the Hive configuration directory can be changed by setting the HIVE_CONF_DIR environment variable.
- Log4j configuration is stored in <install-dir>/conf/hive-log4j.properties

- Hive configuration is an overlay on top of hadoop - meaning the hadoop configuration variables are inherited by default.

- Hive configuration can be manipulated by:

  Editing hive-site.xml and defining any desired variables (including hadoop variables) in it

- From the cli using the set command (see

  below) By invoking hive using the

  syntax:

  $ bin/hive -hiveconf x1=y1 -hiveconf x2=y2
  this sets the variables x1 and x2 to y1 and y2 respectively

4. Creating Hive tables and browsing through them

Open a terminal and give 'hive' and you should get hive shell.

```
hduser@master:~$ hive
```

```
hive>CREATE TABLE pokes (foo INT, bar STRING);
```

Creates a table called pokes with two columns, the first being an integer and the other a string

```
hive>CREATE TABLE invites (invites_foo INT, invites_bar STRING)
PARTITIONED BY (ds STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY
```

```
',';
```

Creates a table with name 'invites' having two columns and a partition column called ds. The partition column is a virtual column. It is not part of the data itself but is derived from the partition that a particular dataset is loaded into.

By default, tables are assumed to be of text input format and the delimiters are assumed to be ^A(ctrl-a).

```
hive>SHOW TABLES;
```

It lists all the tables

```
hive>SHOW TABLES '.*s';
```

Lists all the table that end with 's'. The pattern matching follows Java regular expressions. Check out this link for documentation http://java.sun.com/javase/6/docs/api/java/util/regex/Pattern.html

```
hive>DESCRIBE invites;
```

Shows the list of columns

As for altering tables, table names can be changed and additional columns can be dropped:

```
hive>ALTER TABLE pokes ADD COLUMNS (new_col INT);
```

Dropping tables:

```
hive>DROP TABLE  pokes;
```

5. Metadata Store

Metadata is in an embedded Derby database whose disk storage location is determined by the hive configuration variable named javax.jdo.option.ConnectionURL. By default(see conf/hive-default.xml), this location is ./metastore_db

Right now, in the default configuration, this metadata can only be seen by one user at a time.

Metastore can be stored in any database that is supported by JPOX. The location and the type of the RDBMS can be controlled by the two variables javax.jdo.option.ConnectionURL and javax.jdo.option.ConnectionDriverName. Refer to JDO (or JPOX) documentation for more details on supported databases. The database schema is defined in JDO metadata annotations file package.jdo at src/contrib/hive/metastore/src/model.

In the future, the metastore itself can be a standalone server.

If you want to run the metastore as a network server so it can be accessed from multiple nodes try HiveDerbyServerMode.

6. DML Operations

Loading data from flat files into Hive:

```
hive>CREATE TABLE students (id INT, name STRING) ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';
```

Come out of the hive shell by giving the below command

```
hive> quit;
```

Create a file with name students.txt on UNIX file system.

```
hduser@master:~$vi students.txt
```

Enter the below content and save the file and then we will load the same file into hive table that we have created earlier.

```
1000,Anil
1002,Ram
1003,Ravi
1004,Radha
1005,Syam
1006,Balu
1007,Pavan
1008,Kamal
1009,Aravid
1010,Raju
```

Connect back to hive shell by giving the below command.

```
hduser@master:~$hive
```

```
hive>LOAD DATA LOCAL INPATH 'students.txt' OVERWRITE INTO TABLE students ;
```

Loads a file that contains two columns separated by ctrl-a into students table. 'local' signifies that the input file is on the local file system. If 'local' is omitted then it looks for the file in HDFS.

The keyword 'overwrite' signifies that existing data in the table is deleted.
If the 'overwrite' keyword is omitted, data files are appended to existing data sets.

NOTES:

- NO verification of data against the schema is performed by the load command.

- If the file is in hdfs, it is moved into the Hive-controlled file system namespace.
- The root of the Hive directory is specified by the option hive.metastore.warehouse.dirin hive-default.xml.

```
hive>LOAD DATA LOCAL INPATH 'students.txt' OVERWRITE INTO TABLE invites
PARTITION (ds='2008-08-15');
```

```
hive>LOAD DATA LOCAL INPATH 'students.txt' OVERWRITE INTO TABLE invites
PARTITION (ds='2008-08-08');
```

The two LOAD statements above load data into two different partitions of the table invites. Table invites must be created as partitioned by the key ds for this to succeed.

```
hduser@master:~$cp  students.txt kv2.txt

hduser@master:~$hadoop fs -copyFromLocal kv2.txt  kv2.txt
```

```
hive>LOAD DATA INPATH '/user/cloudera/kv2.txt' OVERWRITE INTO TABLE invites
PARTITION (ds='2008-08-08');
```

The above command will load data from an HDFS file/directory to the table. Note that loading data from HDFS will result in moving the file/directory. As a result, the operation is almost instantaneous.

7. SQL Operations

```
hive>SELECT a.invites_foo FROM invites a WHERE a.ds='2008-08-15';
```

Selects column 'foo' from all rows of partition ds=2008-08-15 of the invites table. The results are not stored anywhere, but are displayed on the console.

Observe that the above query is actually runs as a map reduce program.

```
hive>INSERT OVERWRITE DIRECTORY '/tmp/hdfs_out' SELECT a.* FROM invites a
WHERE a.ds='2008-08-15';
```

Selects all rows from partition ds=2008-08-15 of the invites table into an HDFS directory.

Partitioned tables must always have a partition selected in the WHERE clause of the statement.

```
hive>INSERT OVERWRITE LOCAL DIRECTORY '/tmp/local_out' SELECT a.* FROM pokes
a;
```

Selects all rows from pokes table into a local directory

```
hive>CREATE TABLE students2 (id INT, name STRING) ROW FORMAT DELIMITED FIELDS
TERMINATED BY ',';
```

```
hive>CREATE TABLE students3 (id INT, name STRING) ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';

hive>INSERT OVERWRITE TABLE students2 SELECT a.* FROM students a;

hive>INSERT OVERWRITE TABLE students3 SELECT a.* FROM students a WHERE
a.id< 1005;

hive>INSERT OVERWRITE LOCAL DIRECTORY '/tmp/reg3' SELECT a.* FROM students3
a;

hive>INSERT OVERWRITE DIRECTORY '/tmp/reg_5' SELECT a.* FROM students3 a;
```

8. More SQL Operations

You can override this default location for the new directory as shown in this example:

```
hive>CREATE DATABASE financials LOCATION '/my/preferred/directory';
```

You can add a descriptive comment to the database, which will be shown by the DESCRIBE DATABASE <database> command.

```
hive>CREATE DATABASE financials2 COMMENT 'Holds all financial tables';
```

You can associate key-value properties with the database, although their only function currently is to provide a way of adding information.

```
hive>CREATE DATABASE financials3 WITH DBPROPERTIES ('creator' = 'Mark
Moneybags', 'date' = '2012-01-02');

hive>DESCRIBE DATABASE financials;
financials hdfs://master-server/user/hive/warehouse/financials.db

hive>DESCRIBE DATABASE EXTENDED financials;
financials hdfs://master-server/user/hive/warehouse/financials.db
{date=2012-01-02, creator=Mark Moneybags);
```

Set current db print mode to be true.

```
hive>set hive.cli.print.current.db=true;

hive (financials)>USE default;

hive>DROP DATABASE IF EXISTS financials;
```

By default, Hive won't permit you to drop a database if it contains tables. You can either drop the tables first or append the CASCADE keyword to the command, which will cause the Hive to drop the tables in the database first:

```
hive>DROP DATABASE IF EXISTS financials CASCADE;
```

A query across all partitions could trigger an enormous MapReduce job if the table data and number of partitions are large. A highly suggested safety measure is putting Hive into 'strict' mode, which prohibits queries of partitioned tables without a WHERE clause that filters on partitions. You can set the mode to 'nonstrict' or 'strict'

```
hive>set hive.mapred.mode=strict;
```

You can see all the partitions of a table with below query

```
hive>SHOW PARTITIONS invites;
```

Create a file containing some hive queries

```
hduser@master:~$
           vi /home/hduser/withqueris.hql
           drop database if exists hr2 cascade;
           create database hr2;
           use hr2;
           create table payment ( eid int, bonus float);
```

Run set of commands stored in a file by running the below statement from UNIX prompt.

```
hduser@master:~$hive -f /home/hduser/withqueries.hql
```

Run a statement and comeout from hive by running below command

```
hduser@master:~$hive -e 'select * from students'
```

Run a command in silent more(The OK and time taken will not be shown)

```
hduser@master:~$hive –S  -e 'select * from students'
```

If you are already in hive shell and you want to run hive queries written a file, run below command from hive shell

```
hive>source /home/hduser/withqueries.hql;
```

The 70+ line of Java code for wordcount can be replaced with the below simple query.

```
hive>CREATE TABLE docs (line STRING);
hive>LOAD DATA INPATH 'docs' OVERWRITE INTO TABLE docs;
hive>CREATE TABLE word_counts AS
     SELECT word, count(1) AS count FROM
     (SELECT explode(split(line, ' ')) AS word FROM docs) w
     GROUP BY word ORDER BY word;
```

9. Work with bigger and production grade data

In this section, I will demonstrate the following activities

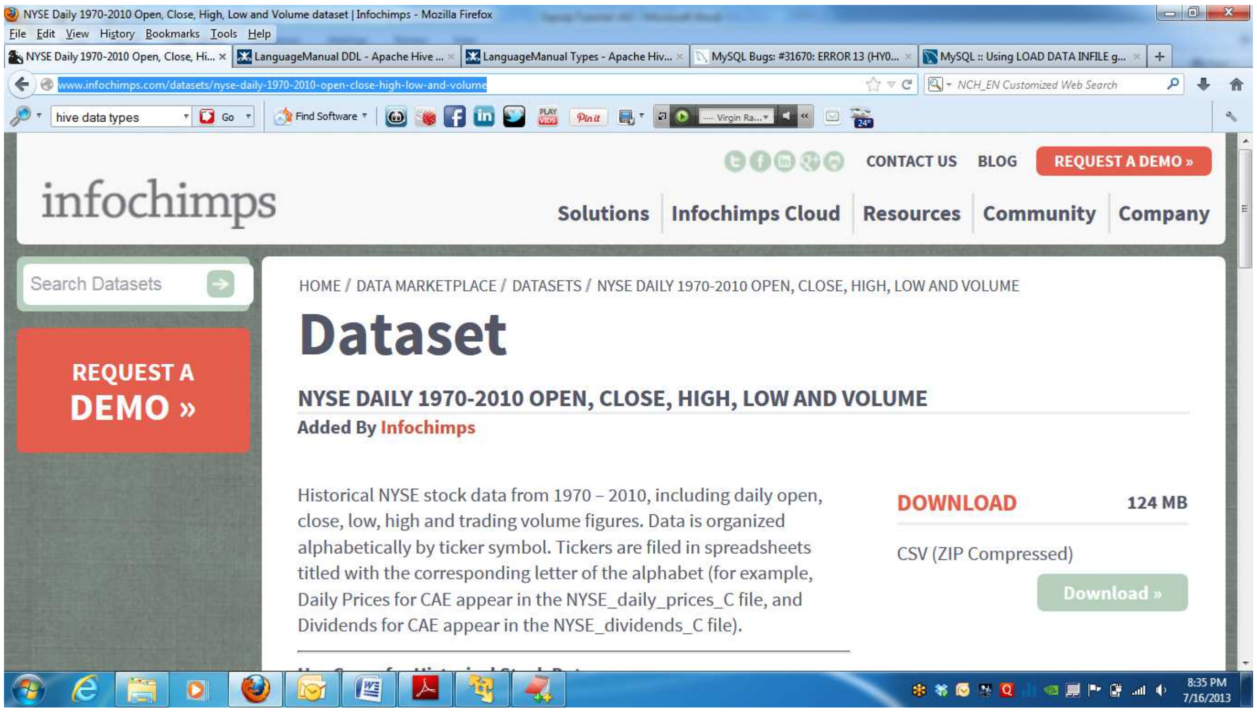
9.1. Download trade data from NYSE (NewYork Stock Exchange).

9.2. Load the data into Hive database that we created in 8

9.1. Download trade data from NYSE (NewYork Stock Exchange).

http://infochimps.com maintains thousands of datasets that you can download and use it for practice.

Goto to http://www.infochimps.com/datasets/nyse-daily-1970-2010-open-close-high-low-and-volume and click on 'Download' button there on right side of the page


NYSE Daily 1970-2010 Open, Close, High, Low and Volume dataset | Infochimps - Mozilla Firefox
File Edit View History Bookmarks Tools Help
NYSE Daily 1970-2010 Open, Close, Hi...
LanguageManual DDL - Apache Hive ...
LanguageManual Types - Apache Hiv...
MySQL Bugs: #31670: ERROR 13 (HY0...
MySQL :: Using LOAD DATA INFILE g...
www.infochimps.com/datasets/nyse-daily-1970-2010-open-close-high-low-and-volume
NCH_EN Customized Web Search
hive data types
Go
Find Software
infochimps
CONTACT US
BLOG
REQUEST A DEMO »
Solutions
Infochimps Cloud
Resources
Community
Company
Search Datasets
REQUEST A DEMO »
HOME / DATA MARKETPLACE / DATASETS / NYSE DAILY 1970-2010 OPEN, CLOSE, HIGH, LOW AND VOLUME
DOWNLOAD
124 MB
CSV (ZIP Compressed)
Download »
8:35 PM
7/16/2013

Dataset

NYSE DAILY 1970-2010 OPEN, CLOSE, HIGH, LOW AND VOLUME

Added By Infochimps

Historical NYSE stock data from 1970 – 2010, including daily open, close, low, high and trading volume figures. Data is organized alphabetically by ticker symbol. Tickers are filed in spreadsheets titled with the corresponding letter of the alphabet (for example, Daily Prices for CAE appear in the NYSE_daily_prices_C file, and Dividends for CAE appear in the NYSE_dividends_C file).

Unzip the downloaded file in your windows computer

Copy the NYSE folder that you got after unzipping into your shared path.

Copy the NYSE folder from the shared path to your home directory with below command

```
hduser@master:~$ cp -r /mnt/hgfs/NYSE ~
```

Open the file ~/NYSE/NYSE_daily_prices_A.csv and remove the first line as it contains headings.

```
hduser@master:~$ vi ~/NYSE/NYSE_daily_prices_A.csv
```

Connect to hive database by running below command

```
hduser@master:~$ hive
```

Use the financials database by running below command

```
hduser@master:~$ use financials
```

Create a table NYSE_DAILY_PRICES_HIVE by running below command

```
hive (financials)>create table NYSE_DAILY_PRICES_HIVE ( s_exchange
string, stock_symbol string, tdate date, stock_price_open DOUBLE,
stock_price_high DOUBLE, stock_price_low DOUBLE, stock_price_close DOUBLE,
stock_volume bigint, stock_price_adj_close DOUBLE ) row format delimited
fields terminated by ',';
```

Load the data into table created in above step

```
      hive (financials)>load data local inpath 'NYSE/NYSE_daily_prices_A.csv'
overwrite into table NYSE_DAILY_PRICES_HIVE;
```

Schema on Read
When you write data to a traditional database, either through loading external data, writing the output of a query, doing UPDATE statements, etc., the database has total control over the storage. The database is the "gatekeeper." An important implication of this control is that the database can enforce the schema as data is written. This is called schema on write.

Hive has no such control over the underlying storage. There are many ways to create, modify, and even damage the data that Hive will query. Therefore, Hive can only enforce queries on read. This is called schema on read.

Hadoop dfs Commands from Inside Hive
You can run the hadoop dfs ... commands from within the hive CLI; just drop the hadoop word from the command and add the semicolon at the end:

```
hive> dfs -ls / ;
Found 3 items
drwxr-xr-x - root supergroup 0 2011-08-17 16:27 /etl
drwxr-xr-x - edward supergroup 0 2012-01-18 15:51 /flag
drwxrwxr-x - hadoop supergroup 0 2010-02-03 17:50 /users
```

Query Column Headers

```
hive>set hive.cli.print.header=true;
```

```
hive>CREATE TABLE IF NOT EXISTS spractice.employees2 LIKE spractice.employees;
hive>CREATE TABLE IF NOT EXISTS spractice.employees3 AS select * from
spractice.employees;
```

If we aren't in the same database, we can still list the tables in that database:

```
hive (default)> show tables in spractice;
```

```
hive>describe extended spractice.employees;
```

Formatted gives the output in much readable format

```
hive>describe formatted spractice.employees;
```

Dropping Tables

The familiar DROP TABLE command from SQL is supported:

```
hive> DROP TABLE stocks123456;
hive> DROP TABLE IF EXISTS stocks123456;
```

The IF EXISTS keywords are optional. If not used and the table doesn't exist, Hive returns an error.

10. Managed tables Vs External tables

Managed Tables

The tables we have created so far are called managed tables or sometimes called internal tables, because Hive controls the lifecycle of their data (more or less). As we've seen, Hive stores the data for these tables in a subdirectory under the directory defined by hive.metastore.warehouse.dir (e.g., /user/hive/warehouse), by default. When we drop a managed table Hive deletes the data in the table.

However, managed tables are less convenient for sharing with other tools. For example, suppose we have data that is created and used primarily by Pig or other tools, but we want to run some queries against it, but not give Hive ownership of the data. We can define an external table that points to that data, but doesn't take ownership of it.

External Tables

Suppose we are analyzing data from the stock markets. Periodically, we ingest the data for NASDAQ and the NYSE from a source like Infochimps (http://infochimps.com/datasets) and we want to study this data with many tools.

The following table declaration creates an external table that can read all the data files for this comma-delimited data in /data/stocks:

```
hive> CREATE EXTERNAL TABLE IF NOT EXISTS stocks (
exchange STRING,
symbol STRING,
ymd STRING,
price_open FLOAT,
price_high FLOAT,
price_low FLOAT,
price_close FLOAT,
volume INT,
price_adj_close FLOAT)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
LOCATION '/user/cloudera/mylocation';
```

```
hive> CREATE EXTERNAL TABLE IF NOT EXISTS dividends (
ymd STRING,
dividend FLOAT,
exchange STRING,
symbol STRING)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
LOCATION '/user/cloudera/mylocation';
```

Because it's external, Hive does not assume it owns the data. Therefore, dropping the table does not delete the data, although the metadata for the table will be deleted.

However, it's important to note that the differences between managed and external tables are smaller than they appear at first. Even for managed tables, you know where they are located, so you can use other tools, hadoop dfs commands, etc., to modify and even delete the files in the directories for managed tables. Hive may technically own these directories and files, but it doesn't have full control over them!

Hive really has no control over the integrity of the files used for storage and whether or not their contents are consistent with the table schema. Even managed tables don't give us this control. Still, a general principle of good software design is to express intent. If the data is shared between tools, then creating an external table makes this ownership explicit.

Use dist cp command to copy data from one location to other and alter table location with below commands

```
hive> desc formatted stocks;

hduser@master:~$hadoop distcp
hdfs://nn.myexperiments.in:8020/user/hduser/mylocation
hdfs://nn.myexperiments.in:8020/user/hduser/mylocation2

hive> ALTER TABLE stocks SET LOCATION
'hdfs://nn.myexperiments.in:8020/user/hduser/mylocation2';

hive> desc formatted stocks;
```

.

11. Table with custom delimiters

Create a table with below structure

```
CREATE TABLE t_employees (
name STRING,
salary FLOAT,
subordinates ARRAY<STRING>,
deductions MAP<STRING, FLOAT>,
address STRUCT<street:STRING, city:STRING, state:STRING, zip:INT>
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
COLLECTION ITEMS TERMINATED BY '|'
MAP KEYS TERMINATED BY '#'
LINES TERMINATED BY '\n'
STORED AS TEXTFILE;
```

Load the table with below data

```
John Doe,100000.0,Mary Smith|Todd Jones,Federal Taxes#.2|State
Taxes#.05|Insurance#.1,1 Michigan Ave.|Chicago|IL|60600
Mary Smith,80000.0,Bill King,Federal Taxes#.2|State Taxes#.05|Insurance#.1,100
Ontario St.|Chicago|IL|60601
Todd Jones,70000.0,,Federal Taxes#.15|State Taxes#.03|Insurance#.1,200 Chicago
Ave.|Oak Park|IL|60700
Bill King,60000.0,,Federal Taxes#.15|State Taxes#.03|Insurance#.1,300 Obscure
Dr.|Obscuria|IL|60100
```

```
hive> SELECT name, subordinates FROM t_employees;
hive> SELECT name, address FROM t_employees;
hive> SELECT name, deductions FROM t_employees;
hive> SELECT name, subordinates[0] FROM t_employees;
```

LIKE and RLIKE predicate operators.
You have probably seen LIKE before, a standard SQL operator.

```
hive> SELECT name, address.street FROM employees WHERE address.street LIKE
'%Ave.';
John Doe 1 Michigan Ave.
Todd Jones 200 Chicago Ave.

hive> SELECT name, address.city FROM employees WHERE address.city LIKE 'O%';
Todd Jones Oak Park
Bill King Obscuria
```

```
hive> SELECT name, address.street FROM employees WHERE address.street LIKE
'%Chi%';
Todd Jones 200 Chicago Ave.
```

A Hive extension is the RLIKE clause, which lets us use Java *regular expressions*, a more powerful minilanguage for specifying matches.

```
hive> SELECT name, address.street
> FROM employees WHERE address.street RLIKE '.*(Chicago|Ontario).*';
Mary Smith 100 Ontario St.
Todd Jones 200 Chicago Ave.
```

11. Table with JSON SerDe

Download JSON jar from net. (It's available in our GDrive @ GDrive\Shared\DatasetsOtherDependencies\ForHive) and copy the jar file to a directory in the VM, for example into /home/cloudera/Downloads.

Similarly, copy sample data files also from GDrive into some folder in VM, say Downloads

```
hive> add jar /home/cloudera/Downloads/json-serde-1.3.6-SNAPSHOT-jar-with-
dependencies.jar;
```

Create a table with below structure

```
CREATE TABLE json_test1 (
    one boolean,
    three array<string>,
    two double,
    four string )
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
STORED AS TEXTFILE;
```

Load the table with data.txt

```
hive> LOAD DATA LOCAL INPATH '/home/cloudera/Downloads/data.txt' OVERWRITE INTO
TABLE  json_test1 ;
hive> select * from json_test1;
hive> select three[1] from json_test1;

CREATE TABLE json_nested_test (
    country string,
    languages array<string>,
    religions map<string,array<int>>)
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
STORED AS TEXTFILE;


hive> LOAD DATA LOCAL INPATH '/home/cloudera/Downloads/ nesteddata.txt'
OVERWRITE INTO TABLE  json_nested_test;
hive> select * from json_nested_test;

hive> select languages[0] from json_nested_test;

hive> select religions['catholic'][0] from json_nested_test;
```

13. Mysql as Hive's metastore

1. Ensure that Mysql and Hadoop daemons are running

2. Copy& Paste the mysql connector to hive/lib folder

3. Change the properties values in hive-site.xml (hive/conf folder)
(Note: if you don't find it, just copy the hive-default.xml and rename to hive-site.xml)

   a. Connection URL

```
        <property>
                <name>javax.jdo.option.ConnectionURL</name>

<value>jdbc:mysql://<hostname>:3306/hive_Metastore?createDatabaseIfNotExist
=true</value>
            </property>
```

   b. Driver

```
                <property>
                      <name>javax.jdo.option.ConnectionDriverName</name>
                      <value>com.mysql.jdbc.Driver</value>
                 </property>
```

   c. Username

```
                  <property>
                       <name>javax.jdo.option.ConnectionUserName</name>
                       <value>user name</value>
                  </property>
```

   d. Password

```
                    <property>
                        <name>javax.jdo.option.ConnectionPassword</name>
                         <value>password</value>
                    </property>
```

4. Start Hive
5. Check the database and hive metadata tables created in mysql db
6. Create test table and check mysql tables.

GROUP BY Clauses
The GROUP BY statement is often used in conjunction with aggregate functions to group the result set by one or more columns and then perform an aggregation over each group.

```
hive> SELECT year(ymd), avg(price_close) FROM stocks
> WHERE exchange = 'NASDAQ' AND symbol = 'AAPL'
> GROUP BY year(ymd);
```