# Apache Hive Lab

In this lab, we are going to use Apache Hive to access our data on our Hadoop cluster and run SQL statements to process and analyze our data.

### Step 1.

Install Apache Derby. The zip file will be made available to you along with the instructions in the README file for installing this on your system.

Edit your .bashrc file to add a DERBY_HOME variable (probably */usr/*lib/derby)

as well as editing your PATH variable to add $DERBY_HOME/bin

Also edit the CLASSPATH variable to add the following:

$DERBY_HOME/lib/derby.jar:$DERBY_HOME/lib/derbytools.jar

Create a directory in Derby to store the metadata

```
$ mkdir $DERBY_HOME/data
```

Edit the hive-site.xml file and add the following lines.

```
<property>
   <name>javax.jdo.option.ConnectionURL</name>
   <value>jdbc:derby://localhost:1527/metastore_db;create=true </value>
   <description>JDBC connect string for a JDBC metastore </description>
</property>
```

In the conf directory, create a file called jpox.properties and add the following text into the file:

```
javax.jdo.PersistenceManagerFactoryClass =


org.jpox.PersistenceManagerFactoryImpl
```

```
org.jpox.autoCreateSchema = false

org.jpox.validateTables = false

org.jpox.validateColumns = false

org.jpox.validateConstraints = false

org.jpox.storeManagerType = rdbms

org.jpox.autoCreateSchema = true

org.jpox.autoStartMechanismMode = checked

org.jpox.transactionIsolation = read_committed

javax.jdo.option.DetachAllOnCommit = true

javax.jdo.option.NontransactionalRead = true

javax.jdo.option.ConnectionDriverName = org.apache.derby.jdbc.ClientDriver

javax.jdo.option.ConnectionURL = jdbc:derby://hadoop1:1527/metastore_db;create =
true

javax.jdo.option.ConnectionUserName = APP

javax.jdo.option.ConnectionPassword = mine
```

### Step 2. Create directories in HDFS for Hive

Create a *tmp directory and a usrhive*warehouse folder inside HDFS using the hdfs dfs mkdir

command. Set permissions on these folders so that group has write access (g+w)

### Step 3. Run Hive.

Run hive by issuing the following commands.

cd $HIVE_HOME

bin/hive

You should now see a prompt that looks like this:

hive>

Run a show tables command.

You should see a responsible similar to this:

```
hive> show tables;
```

```
OK

Time taken: 2.798 seconds

hive>
```

## Step 4. Create a database schema.

In hive, run the following command:s

```
hive > CREATE DATABASE [IF NOT EXISTS] userdb;
hive> SHOW DATABASES;
The output should look like this:
```

```
default

userdb
```

## Step 5. Load data

Generally, after creating a table in SQL, we can insert data using the Insert statement. But in Hive, we can insert data using the LOAD DATA statement.

While inserting data into Hive, it is better to use LOAD DATA to store bulk records. There are two ways to load data: one is from local file system and second is from Hadoop file system.

```
LOAD DATA [LOCAL] INPATH 'filepath' [OVERWRITE] INTO TABLE tablename
[PARTITION (partcol1=val1, partcol2=val2 ...)]
```

LOCAL is identifier to specify the local path. It is optional.

- OVERWRITE is optional to overwrite the data in the table.
- PARTITION is optional.

You will be provided a file called ALBBSalaries.2003.formatted.csv

This file will contain salaries for every player in the American baseball league for the year 2003 by team.

Using HIVE SQL, load this data, and run an aggregate query that will give the following results:

team: total salary