

Group Members

Name : Abdul Qadeer Khan **ERPID** : 18155

Name : Ali Asghar Marv **ERPID** : 07896

Name : Asim Farooq **ERPID** : 21074

Name : Saadia Rasheed **ERPID** : 09812

Hive Lab

Prerequisites

- Docker
 - To follow docker installation, go to the link <https://runnable.com/docker/install-docker-on-linux> for Linux, and <https://docs.docker.com/docker-for-windows/install/> for Windows.
- Computer with 8GB ram with at least 60GB in the root partition.

PULL image

Pulling an image will take a few hours so it is advised to do before.

```
docker pull cloudera/quickstart
```

For more information about this image go to the link below:

<https://hub.docker.com/r/cloudera/quickstart/>

RUN

Update the shared volume path accordingly

```
-v E:\workspace\docker:/user/cloudera/shared
```

Here the path before the “:” is on the host machine and the path after the “:” is in docker, update the host path to wherever you want to be on the host machine.

Following command will start the container and connect you to the **shell**. We will be running commands on this next.

If you want to get out of the container without stopping it use **Ctrl+p Ctrl+q** .

Linux (/home/ace/Desktop/dockerab/hive)

```
docker run --hostname=quickstart.cloudera --privileged=true -t -v  
/home/ace/Desktop/dockerab/hive:/user/cloudera/shared -i -p 8889:8888 -p  
7180:7180 cloudera/quickstart /usr/bin/docker-quickstart
```

Windows (E:\workspace\docker)

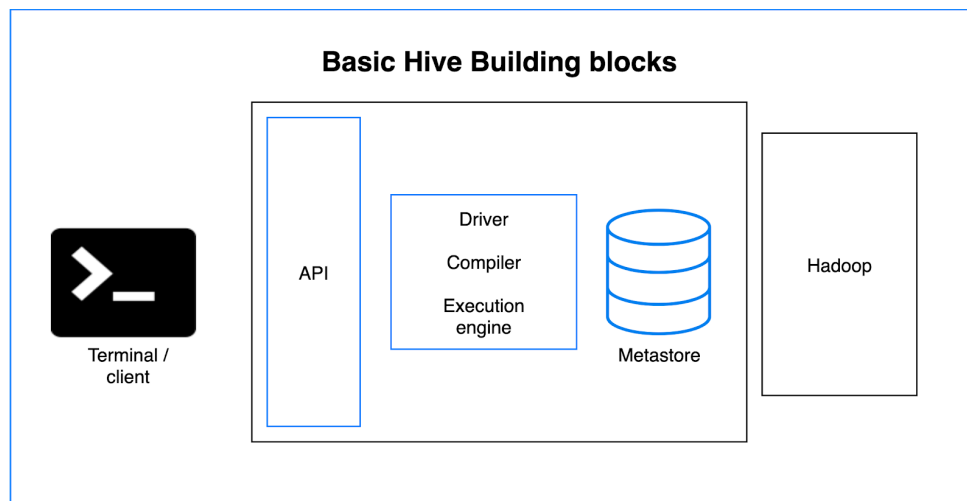
Disclaimer: docker should be running linux containers on windows which are only available on the pro version of windows i.e windows 10 pro.

```
docker run --hostname=quickstart.cloudera --privileged=true -t -v  
E:\workspace\docker:/user/cloudera/shared -i -p 8889:8888 -p 7180:7180  
cloudera/quickstart /usr/bin/docker-quickstart
```

The above command

Port **8889** is for hue and port **7180** is for cloudera manager.

Overview of hive:



Hive comes with a driver compiler execution engine metastore and api. As shown in the above image.

Hive uses hadoop as underlying storage.

This builtin metastore uses the Derby database. More info can be found at this link:

<https://dzone.com/articles/hive-metastore-a-basic-introduction>

Right now we will be working with only one session on hive to allow multiple session a remote or local meta store needs to be configured.

Hive is schema on read and the schema to file mapping is stored in the meta store.

Hive is also not acid compliant by default , we can make tables acid compliant but that often defeats the purpose of big data processing, next we are going to look at the hive command line.

Hive command line

Once inside the quickstart/cloudera docker container, type in **hive** which should connect you to the hive command line hive client.

The default hive client is deprecated and hive also comes with beeline client which we won't be using in this tutorial.

```
show databases;
```

This query will show the default database. First query can take up to 2 mins.
Remember all queries end with **;**. Try out the following queries.

```
create database BDA;

Use BDA;

create table student( id bigint,name string,remarks string );

show tables;

describe student;

INSERT INTO TABLE student VALUES
(1111,"qadeer","okay"),(1112,"ashly","okay");

Select * from student;

Select * from student where name = "qadeer";

Select * from student where id > 1111;

exit;
```

Next we will look at the GUI

Access

Localhost:8889

Username : cloudera

Password : cloudera

Go to the **Query editor** and then click **Hive** the refresh button should be able to see the student database.

The screenshot shows the Hue Hive Editor interface. The top navigation bar includes 'Hive Editor', 'Query Editor', 'My Queries', 'Saved Queries', and 'History'. The left sidebar shows a tree view with 'bda' and 'student' tables. The main query editor area contains a single query: 'show databases;'. A red box highlights the refresh button (a circular arrow icon) next to the query. A red arrow points from the refresh button to the text 'click on the refresh button when editing from the gui and cli at the same time'. Below the query editor are buttons for 'Execute', 'Save as...', 'Explain', 'Format', 'or create a', and 'New query'. The bottom section shows the 'Results' tab with a table of database names.

	database_name
1	bda
2	default

Find where hive is

whereis hive

HDFS and HIVE

As hive stores x in HDFS next we will look at exploring how hive manages data in hdfs. Hive does not apply any protection over the file it writes.

Files can be written as normal text csv files that can be read by other softwares e.g pig and spark.

<https://mindmajix.com/hadoop-hdfs-commands-with-examples>

```
hadoop fs -ls /user/hive/warehouse
```

Because of previously created bda database, we should be able to see this:

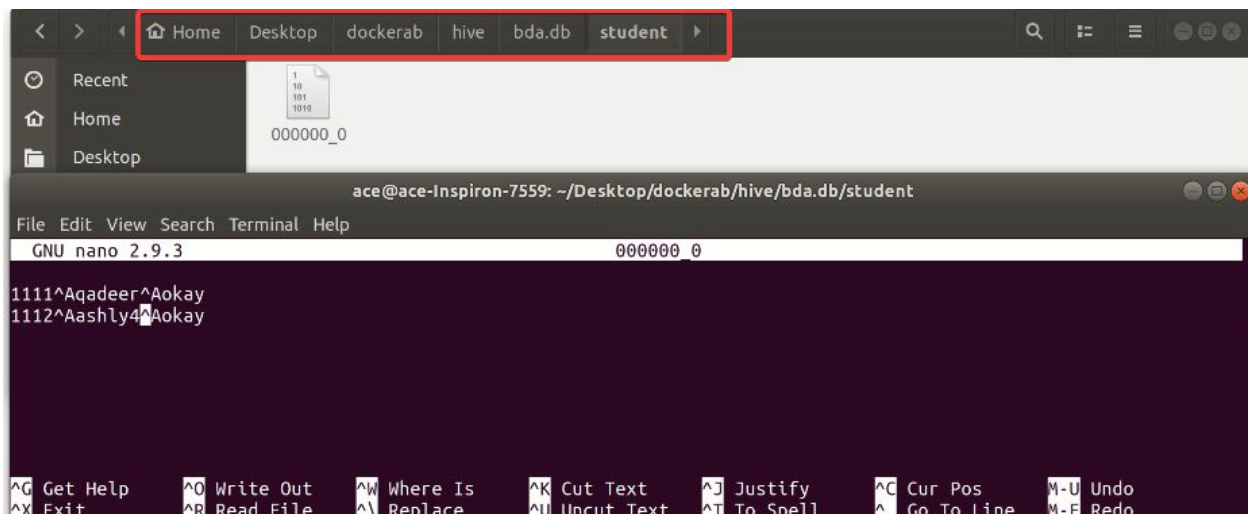
```
[root@quickstart usr]# hadoop fs -ls /user/hive/warehouse
Found 4 items
drwxrwxrwx - root supergroup 0 2020-05-21 22:52 /user/hive/warehouse/bda.db
drwxrwxrwx - cloudera supergroup 0 2020-05-21 22:26 /user/hive/warehouse/sample_07
drwxrwxrwx - cloudera supergroup 0 2020-05-21 22:26 /user/hive/warehouse/sample_08
drwxrwxrwx - cloudera supergroup 0 2020-05-21 22:26 /user/hive/warehouse/web_logs
```

Copy this file in you local machine to inspect

```
hadoop fs -get /user/hive/warehouse/bda.db /user/cloudera/shared/
```

Now go in your shared path that you passed in the docker run command

/home/ace/Desktop/dockerab/hive



You can see here that the bda.db is a folder and there is a student folder in it which has a file. Opening that file we can see that its a plane text file with the data we entered.

Now let edit one entry and put the file back to see if the values are updated i have changed "ashly" to "ashly4".

Once we have edited we will have to use the put command to put the file from local into hdfs.

As path already exists we have to pass the `-f` argument

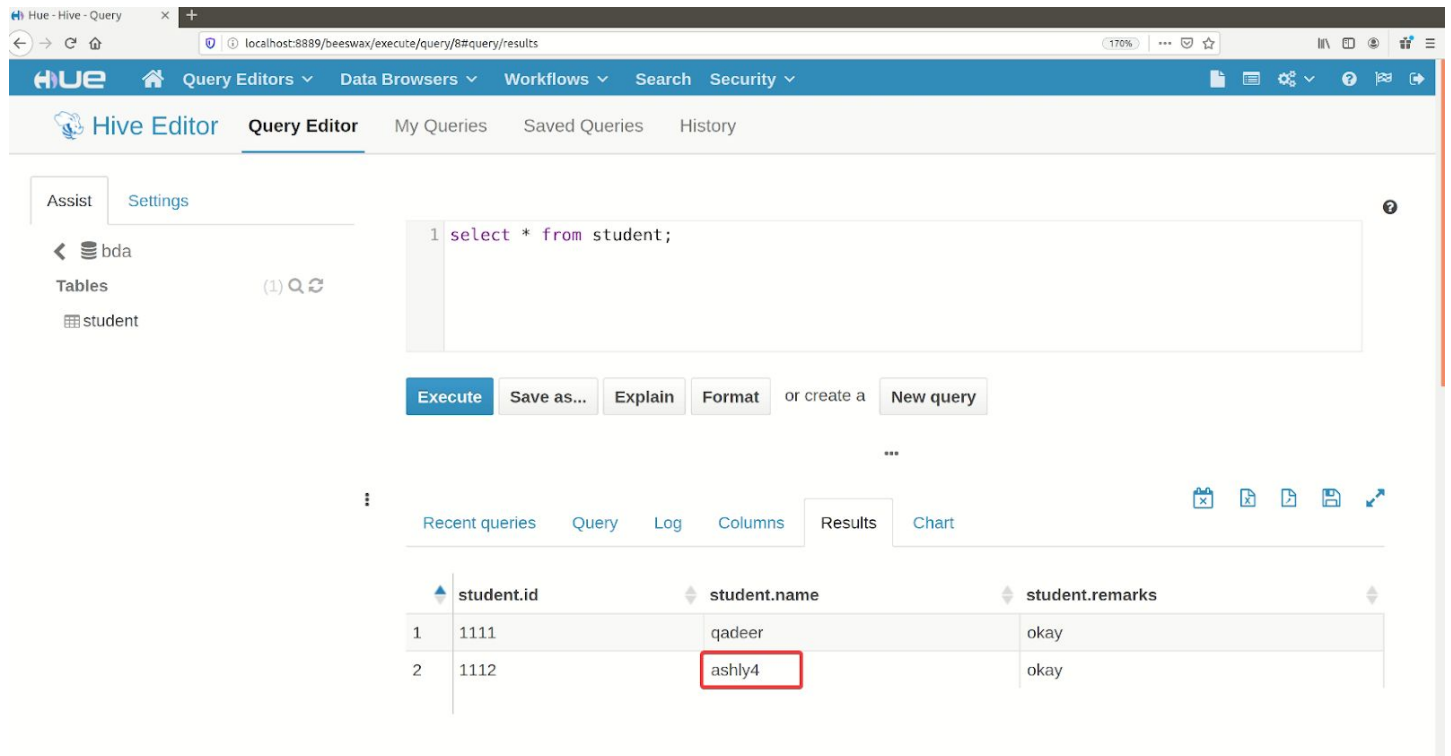
```
hadoop fs -put -f /user/cloudera/shared/bda.db/ /user/hive/warehouse/
```

```
hadoop fs -cat /user/hive/warehouse/bda.db/student/000000_0
```

Now when we do a

```
select * from student;
```

We should see “ashly4”



The screenshot shows the Hue Hive Query Editor interface. The query editor contains the query `select * from student;`. Below the query editor, there are buttons for `Execute`, `Save as...`, `Explain`, `Format`, and `New query`. The `Execute` button has been clicked, and the results are displayed in a table. The table has three columns: `student.id`, `student.name`, and `student.remarks`. The results are as follows:

	student.id	student.name	student.remarks
1	1111	qadeer	okay
2	1112	ashly4	okay

How is hive configures with hdfs

Previously we saw the “whereis hive” command

```
cd /usr/lib/hive
ls
cd conf
```

```
cat hive-site.xml
```

With this we can see/update hive jdbc connector and other settings.

```
cat hive-env.sh.template
```

With this we can see that location for hdfs is being picked up from **HADOOP_HOME** environment variable

Hive concepts

Importing data from a file

Lets download the csv files from the link into our host machine and move it into hdfs (trucks.csv , geolocation.csv)

<https://app.box.com/v/hadoopcrashcoursedata>

Download and extract on your machine.

Copy in your docker we are going to use **docker cp** command

In your container cli run the following commands

```
cd /home/cloudera/Desktop/  
mkdir geolocation
```

On your **host** extract the two files in **~/Desktop/data/** and then run the below commands :

Replace **8891310e5b13** with your container ID: (`docker ps`)

```
docker cp ~/Desktop/data/geolocation.csv
```

```
8891310e5b13:/home/cloudera/Desktop/geolocation
```

```
docker cp ~/Desktop/data/trucks.csv 8891310e5b13:/home/cloudera/Desktop/geolocation/
```

Now copy from the docker into HDFS

Let's make a new directory in **hdfs** run the following in your container and copy the data there:

```
hadoop fs -mkdir /user/data/  
  
hadoop fs -put -f /home/cloudera/Desktop/geolocation /user/data/
```

Let's check :

```
hadoop fs -ls /user/data/geolocation
```

```
[root@quickstart geolocation]# hadoop fs -ls /user/data/geolocation
Found 2 items
-rw-r--r--  1 root supergroup    526677 2020-05-25 03:23 /user/data/geolocation/geolocation.csv
-rw-r--r--  1 root supergroup    61378 2020-05-25 03:23 /user/data/geolocation/trucks.csv
[root@quickstart geolocation]#
```

Note : we could have also put these files in the shared volume and copy into hdfs from there.

Reference

<https://docs.docker.com/engine/reference/commandline/cp/>

```
docker cp [OPTIONS] SRC_PATH|- CONTAINER:DEST_PATH
```

Now we are going to import it into hive from hue

<https://gethue.com/hadoop-tutorial-create-hive-tables-with-headers-and/>

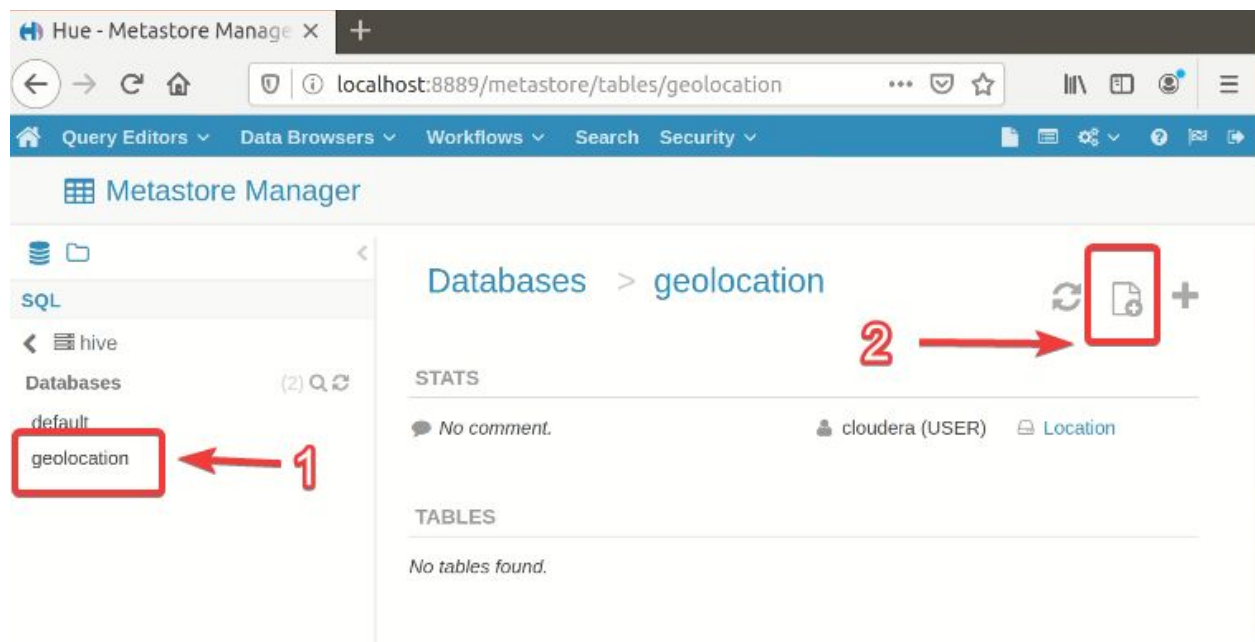
Let's make a new database named **geolocation**

GOTO (host machine browser) : <http://localhost:8889/beeswax/create/database#step/1>

After creating the DB

Go in **Data Browsers** and click on **Metastore Tables**

Select the database then click create new table from file button



Create the tables for geolocation

Name : geolocation

Path : /user/data/geolocation/geolocation.csv

HUE Query Editors Data Browsers Workflows Search Security

Metastore Manager

SQL

geolocation

Tables

The database has no tables

HDFS

/user/hive

warehouse

Databases > geolocation > Create a new table from a file

Step 1: Choose File Step 2: Choose Delimiter Step 3: Define Columns

Name Your Table and Choose A File

Table Name

Name of the new table. Table names must be globally unique. Table names tend to correspond to the directory where the data will be stored.

Description

Use a table comment to describe the table. For example, note the data's provenance and any caveats users need to know.

Input File

..

The HDFS path to the file on which to base this new table definition. It can be compressed (gzip) or not.

Import data from file ☒

Check this box to import the data in this file after creating the table definition. Leave it unchecked to define an empty table.

Warning: The selected file is going to be moved during the import.

Next

HIVE Query Editors Data Browsers Workflows Search Security

Metastore Manager

Databases > geolocation > Create a new table from a file

Step 1: Choose File **Step 2: Choose Delimiter** Step 3: Define Columns

Choose a Delimiter

Beeswax has determined that this file is delimited by **commas**.

Delimiter: **Preview**

Enter the column delimiter which must be a single character. Use syntax like "\001" or "\t" for special characters.

Table preview

col_1	col_2	col_3	col_4	col_5	col_6	col_7	col_8	col_9
truckid	driverid	event	latitude	longitude	city	state	velocity	event
A54	A54	normal	38.440467	-122.714431	Santa Rosa	California	17	0
A20	A20	normal	36.977173	-121.899402	Aptos	California	27	0
A40	A40	overspeed	37.957702	-121.29078	Stockton	California	77	1
A31	A31	normal	39.409608	-123.355566	Willits	California	22	0
A71	A71	normal	33.683947	-117.794694	Irvine	California	43	0
A50	A50	normal	38.40765	-122.947713	Occidental	California	0	0
A51	A51	normal	37.639097	-120.996878	Modesto	California	0	0
A19	A19	normal	37.962146	-122.345526	San Pablo	California	0	0
A77	A77	normal	37.962146	-122.345526	San Pablo	California	25	0

Previous **Next**

To query the data go to **Query Editor > Hive** or the below url

<http://localhost:8889/beeswax/#query>

Examples:

Run the below queries

```
select * from geolocation;
select * from geolocation where event = "overspeed";
select * from geolocation where velocity < 40;
select distinct event from geolocation;
select avg(velocity) from geolocation where event ="lane departure" group by event;
select * from geolocation order by velocity desc limit 5;
```

Exercises

1. Get all the distinct cities
2. Get all the cities for driver id A54
3. Get driver who have visited the least amount of cities
4. Get driver who have visited the most amount of cities
5. Get driver who on average drives the slowest
6. Using the like statement get all cities with name starting from A

Answers

```
select distinct city from geolocation;
select distinct city from geolocation where driverid="A54";
select count(distinct city),driverid from geolocation group by driverid;
select count(distinct city) as sum_ ,driverid from geolocation group by
driverid order by sum_ desc;
```

Create table for trucks (we are going to create an external table)

External tables are not managed by hive dropping the table will not delete the actual data, while the managed table the tile is moved into the hive warehouse directory.

Name : trucks

Path : /user/data/geolocation/trucks.csv

Run the following command in hue

Run to find hdfs url :

```
hdfs getconf -confKey fs.defaultFS
```

The query continues on the next page please select the whole query

```
CREATE EXTERNAL TABLE IF NOT EXISTS geolocation.trucks
(driverid STRING,truckid STRING,model STRING,jun13_miles INT,jun13_gas
INT,may13_miles INT,may13_gas INT,apr13_miles INT,apr13_gas INT,mar13_miles
INT,mar13_gas INT,feb13_miles INT,feb13_gas INT,jan13_miles INT,jan13_gas
INT,dec12_miles INT,dec12_gas INT,nov12_miles INT,nov12_gas INT,oct12_miles
INT,oct12_gas INT,sep12_miles INT,sep12_gas INT,aug12_miles INT,aug12_gas
INT,jul12_miles INT,jul12_gas INT,jun12_miles INT,jun12_gas INT,may12_miles
INT,may12_gas INT,apr12_miles INT,apr12_gas INT,mar12_miles INT,mar12_gas
INT,feb12_miles INT,feb12_gas INT,jan12_miles INT,jan12_gas INT,dec11_miles
INT,dec11_gas INT,nov11_miles INT,nov11_gas INT,oct11_miles INT,oct11_gas
INT,sep11_miles INT,sep11_gas INT,aug11_miles INT,aug11_gas INT,jul11_miles
INT,jul11_gas INT,jun11_miles INT,jun11_gas INT,may11_miles INT,may11_gas
```

```

INT,apr11_miles INT,apr11_gas INT,mar11_miles INT,mar11_gas INT,feb11_miles
INT,feb11_gas INT,jan11_miles INT,jan11_gas INT,dec10_miles INT,dec10_gas
INT,nov10_miles INT,nov10_gas INT,oct10_miles INT,oct10_gas INT,sep10_miles
INT,sep10_gas INT,aug10_miles INT,aug10_gas INT,jul10_miles INT,jul10_gas
INT,jun10_miles INT,jun10_gas INT,may10_miles INT,may10_gas INT,apr10_miles
INT,apr10_gas INT,mar10_miles INT,mar10_gas INT,feb10_miles INT,feb10_gas
INT,jan10_miles INT,jan10_gas INT,dec09_miles INT,dec09_gas INT,nov09_miles
INT,nov09_gas INT,oct09_miles INT,oct09_gas INT,sep09_miles INT,sep09_gas
INT,aug09_miles INT,aug09_gas INT,jul09_miles INT,jul09_gas INT,jun09_miles
INT,jun09_gas INT,may09_miles INT,may09_gas INT,apr09_miles INT,apr09_gas
INT,mar09_miles INT,mar09_gas INT,feb09_miles INT,feb09_gas INT,jan09_miles
INT,jan09_gas INT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
LOCATION 'hdfs://quickstart.cloudera:8020/user/data/geolocation/';

```

Examples

```

select * from trucks;
select count(model) as modelCount ,model from trucks group by model order by
modelCount desc;

```

Order By:

Order by query is used in HiveQL in order to sort data with reference to a specific column. For whatever column will be called by the order query, the data will be sorted according to its ascending or descending order.

```

select * from geolocation ORDER BY truckid;

```

Output:

@quickstart:/home/cloudera/Desktop									
File Edit View Search Terminal Tabs Help									
ace@ace-tpirion-7558 ~									
@quickstart:/home/cloudera/Desktop									
A98	A98	normal	39.017395	-122.05775	Arbuckle	California	74	0	0
A98	A98	normal	37.941326	-122.4847	San Quentin	California	20	0	0
A98	A98	normal	32.81116	-115.380264	Holtville	California	0	0	1
A98	A98	normal	37.106674	-117.486725	San Dimas	California	64	0	0
A98	A98	normal	37.957783	-121.29078	Stockton	California	46	0	0
A98	A98	normal	38.80546	-123.01722	Cloverdale	California	39	0	0
A98	A98	normal	38.752125	-121.28801	Roseville	California	16	0	0
A98	A98	normal	38.716686	-121.264426	Antelope	California	54	0	0
A98	A98	normal	37.441883	-122.14302	Palo Alto	California	27	0	0
A98	A98	normal	38.716686	-121.364426	Antelope	California	74	0	0
A98	A98	normal	37.957783	-121.29078	Stockton	California	55	0	0
A98	A98	normal	33.68395	-117.79469	Irvine	California	90	1	0
A98	A98	lane departure	33.19587	-117.379486	Oceanside	California	55	1	0
A98	A98	normal	32.61768	-116.18742	Jacumba	California	32	0	0
A98	A98	normal	32.61768	-116.18742	Jacumba	California	0	1	0
A98	A98	normal	37.941326	-122.4847	San Quentin	California	63	0	0
A98	A98	normal	33.68395	-117.79469	Irvine	California	58	0	0
A98	A98	unsafe following distance	33.68395	-117.79469	Irvine	California	33	1	0
A98	A98	normal	36.471867	-6.196595	San Fernando	California	37	0	0
A98	A98	normal	32.81116	-115.380264	Holtville	California	34	0	0
A98	A98	normal	38.80546	-123.01722	Cloverdale	California	33	0	0
A98	A98	normal	37.957783	-121.29078	Stockton	California	27	0	0
A98	A98	normal	38.716686	-121.264426	Antelope	California	25	0	0
A98	A98	normal	36.471867	-6.196595	San Fernando	California	0	1	0
A98	A98	normal	37.941326	-122.4847	San Quentin	California	75	0	0
A98	A98	overspeed	38.80546	-123.01722	Cloverdale	California	78	1	0
A98	A98	normal	33.19587	-117.379486	Oceanside	California	27	0	0
A98	A98	normal	38.80546	-123.01722	Cloverdale	California	53	0	0
A98	A98	normal	38.752125	-121.28801	Roseville	California	60	0	0
A98	A98	normal	37.941326	-122.4847	San Quentin	California	46	0	0
A98	A98	normal	32.61768	-116.18742	Jacumba	California	32	0	0
A99	A99	normal	38.448468	-122.71443	Santa Rosa	California	25	0	0
A99	A99	normal	38.16186	-121.61162	Isleton	California	68	0	0
A99	A99	normal	38.448468	-122.71443	Santa Rosa	California	50	0	0
A99	A99	normal	34.44805	-119.24289	Ojai	California	53	0	0
A99	A99	normal	34.44805	-119.24289	Ojai	California	27	0	0
A99	A99	normal	33.846405	-118.04673	La Palma	California	0	1	0
A99	A99	normal	33.846405	-118.04673	La Palma	California	55	0	0
A99	A99	normal	40.563763	-122.23889	Palo Cedro	California	45	0	0
A99	A99	normal	34.44805	-119.24289	Ojai	California	18	0	0
A99	A99	normal	39.15017	-123.20779	Ukiah	California	68	1	0
A99	A99	unsafe following distance	33.743876	-117.1092	Homeland	California	0	1	1
A99	A99	normal	38.16186	-121.61162	Isleton	California	73	0	0
A99	A99	normal	38.019367	-122.13413	Martinez	California	46	0	0
A99	A99	normal	38.019367	-122.13413	Martinez	California	63	0	0
A99	A99	normal	34.44805	-119.24289	Ojai	California	50	0	0
A99	A99	unsafe following distance	39.15017	-123.20779	Ukiah	California	62	1	0
A99	A99	normal	39.15017	-123.20779	Ukiah	California	22	0	0
A99	A99	normal	33.743076	-117.1092	Homeland	California	74	0	0
A99	A99	normal	39.15017	-123.20779	Ukiah	California	16	0	0
A99	A99	normal	33.846405	-118.04673	La Palma	California	69	0	0
A99	A99	normal	34.50893	-117.185975	Apple Valley	California	56	0	0
A99	A99	normal	37.957783	-121.29078	Stockton	California	40	0	0
A99	A99	normal	38.448468	-122.71443	Santa Rosa	California	54	0	0

Group By:

In Group By query, a particular column is called and its values are grouped together. The rest of the data in the table is sorted accordingly.

```
select city, count(*) from geolocation GROUP BY city;
```

Output:

```
MapReduce Total cumulative CPU time: 6 seconds 370 msec
Ended Job = job_1590605280755_0002
MapReduce Jobs Launched:
Stage:Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 6.37 sec HDFS Read: 534133 HDFS Write: 708 SUCCESS
Total MapReduce CPU Time Spent: 6 seconds 370 msec
OK
Antelope 315
Apple Valley 315
Aptos 236
Arbuckle 236
Bakersfield 158
Cloverdale 78
Fresno 79
Gilroy 79
Glen Ellen 79
Glendora 79
Goleta 79
Hollister 79
Holtville 79
Homeland 79
Irvine 158
Isleton 158
Jacumba 158
Klamath 79
Kneeland 79
Knights Landing 79
Knightsen 79
La Palma 158
La Puente 79
La Quinta 79
Lodi 156
Mariposa 78
Markleeville 78
Martinez 78
Marysville 78
Modesto 158
Napa 79
```

Sort By:

Sort By performs sorting on the table with respect to the called column in either ascending or descending order. ASC or DESC is called to sort the data in ascending or descending order respectively.

```
select * from geolocation SORT BY driverid ASC;
```

Output:

A89	A89	normal	38.440468	-122.71443	Santa Rosa	California	69	0	0	
A89	A89	unsafe tail distance	38.716686	-121.364426	Antelope	California	53		1	
A89	A89	normal	34.953033	-120.43572	Santa Maria	California	68	0	0	
A89	A89	normal	37.96881	-121.66801	Knightsen	California	64	0	0	
A89	A89	normal	38.440468	-122.71443	Santa Rosa	California	42	0	0	
A89	A89	normal	36.977173	-121.8994	Aptos	California	41	0	0	
A89	A89	normal	34.953033	-120.43572	Santa Maria	California	42	0	0	
A89	A89	normal	38.440468	-122.71443	Santa Rosa	California	37	0	0	
A89	A89	normal	38.716686	-121.364426	Antelope	California	19	0	0	
A89	A89	normal	38.440468	-122.71443	Santa Rosa	California	43	0	0	
A89	A89	normal	39.409607	-123.35557	Willits	California	16	0	0	
A89	A89	normal	37.77493	-122.41942	San Francisco	California	0	0	1	
A89	A89	normal	37.77493	-122.41942	San Francisco	California	43	0	0	
A89	A89	normal	39.017395	-122.05775	Arbuckle	California	75	0	0	
A89	A89	normal	38.440468	-122.71443	Santa Rosa	California	46	0	0	
A89	A89	normal	34.953033	-120.43572	Santa Maria	California	17	0	0	
A89	A89	normal	37.639095	-120.99688	Modesto	California	0	0	1	
A89	A89	normal	37.962147	-122.34553	San Pablo	California	19	0	0	
A89	A89	normal	34.953033	-120.43572	Santa Maria	California	71	0	0	
A89	A89	normal	36.977173	-121.8994	Aptos	California	50	0	0	
A89	A89	normal	37.962147	-122.34553	San Pablo	California	75	0	0	
A89	A89	normal	39.409607	-123.35557	Willits	California	0	0	1	
A9	A9	overspeed	33.846405	-118.04673	La Palma	California	88	1		0
A9	A9	normal	34.019455	-118.49119	Santa Monica	California	0	0	1	
A9	A9	normal	38.716686	-121.364426	Antelope	California	68	0	0	
A9	A9	normal	41.52651	-124.03841	Klamath	California	40	0	0	
A9	A9	normal	33.846405	-118.04673	La Palma	California	75	0	0	
A9	A9	normal	41.52651	-124.03841	Klamath	California	46	0	0	
A9	A9	normal	38.130196	-121.272446	Lodi	California	62	0	0	
A9	A9	normal	38.130196	-121.272446	Lodi	California	0	0	1	
A9	A9	normal	38.716686	-121.364426	Antelope	California	67	0	0	
A9	A9	normal	38.29754	-122.286865	Napa	California	35	0	0	
A9	A9	lane departure	33.846405	-118.04673	La Palma	California	74	1		0
A9	A9	normal	37.957703	-121.29078	Stockton	California	0	0	1	

Distribute By:

Distribute BY clause used on tables present in Hive. Hive uses the columns in Distribute by to distribute the rows among reducers. All Distribute BY columns will go to the same reducer.

- It ensures each of N reducers gets non-overlapping ranges of column
- It doesn't sort the output of each reducer

```
select driverid, city from geolocation DISTRIBUTE BY driverid;
```

Output:

A49	Santa Rosa
A46	Apple Valley
A38	Apple Valley
A100	Occidental
A79	Palo Alto
A59	Jacumba
A24	Hollister
A6	Santa Paula
A33	San Dimas
A7	Palo Cedro
A81	Santa Maria
A39	Antelope
A71	Holtville
A12	Santa Rosa
A63	Irvine
A52	San Quentin
A62	Willits
A5	Napa
A85	Modesto
A34	Palo Cedro
A21	Roseville
A50	Bakersfield
A70	Markleeville
A20	Modesto
A9	San Diego
A9	San Diego
A31	Willits

Cluster By:

Cluster by query is an alternative to SORT BY and DISTRIBUTE BY queries in HiveQL. Hive makes use of the columns in the cluster to distribute the rows among reducers.

```
select driverid, city from geolocation CLUSTER BY driverid;
```

Output:

A92	Fresno
A92	San Diego
A92	Santa Rosa
A92	Apple Valley
A92	Willits
A92	Santa Rosa
A92	Santa Rosa
A92	San Diego
A92	Bakersfield
A92	San Diego
A92	Occidental
A92	Aptos
A92	Apple Valley
A92	Willits
A92	Santa Rosa
A92	San Diego
A92	Apple Valley
A92	Santa Rosa
A92	Apple Valley
A92	Willits
A92	Bakersfield
A92	Santa Rosa
A92	Santa Rosa
A92	Apple Valley
A92	Knights Landing
A92	Palo Cedro
A92	Aptos
A92	Willits
A92	San Diego
A92	Santa Rosa
A93	Hollister
A93	Modesto
A93	Modesto
A93	Modesto
A93	San Dimas
A93	San Dimas
A93	Markleeville
A93	Oceano
A93	Knightsen
A93	Knightsen
A93	Knightsen
A93	Hollister
A93	San Pablo
A93	Santa Rosa
A93	Aptos

Update Data:

In order to update data, this simple query allows every row in the table to be updated depending on the WHERE clause.

```
UPDATE geolocation SET driverid = null WHERE event = "normal";
```

Merge Data:

Merge query allows conditional insertion, deletion and update of data.

```
MERGE INTO geolocation USING (SELECT * FROM trucks) ON
geolocation.driverid=trucks.driverid
WHEN MATCHED THEN UPDATE geolocation SET truckid = trucks.truckid, driverid =
trucks.driverid
WHEN NOT MATCHED THEN INSERT VALUES (trucks.truckid, trucks.driverid)
```

Joins:

Some points to be observed in joins:

- Only equality joins are allowed in Hive.
- More than two tables can be joined from within the same query.
- Hive allows LEFT, RIGHT, FULL OUTER joins in order to preserve more control over ON clause which has no alternative.
- Commutative joins are not possible.
- Irrespective of being LEFT or RIGHT joins, all joins are left-associative.

There are 4 types of joins:

- Inner join
- Left outer join
- Right outer join
- Full outer join

Inner join will retrieve the records common to both tables.

```
select geolocation.driverid,trucks.model,geolocation.city
from geolocation join trucks
where geolocation.driverid=trucks.driverid;
```

In this query, JOIN is implemented between tables geolocation and trucks where the driverid columns between both is the common factor. The output will show the common records present in both tables.

Left outer join query will return all the records from the left table even if it's not in common with the right table.

```
select geolocation.driverid,trucks.model,geolocation.city
from geolocation left outer join trucks
where geolocation.driverid=trucks.driverid;
```

In this query, LEFT OUTER JOIN is implemented between tables geolocation and trucks where the driverid columns between both is the common factor. The output will show the common records present in both tables and NULL value will be inserted in the records where nothing is common with the right table.

Right outer join query will return all the records from the right table even if it's not in common with the left table.

```
select geolocation.driverid,trucks.model,geolocation.city
from geolocation right outer join trucks
where geolocation.driverid=trucks.driverid;
```

In this query, RIGHT OUTER JOIN is implemented between tables geolocation and trucks where the driverid columns between both is the common factor. The output will show the common records present in both tables and NULL value will be inserted in the records where nothing is common with the left table.

Full outer join query returns complete records from both tables and inserts null values in areas where one table has no common record with the other table.

```
select geolocation.driverid,trucks.model,geolocation.city
from geolocation full outer join trucks
where geolocation.driverid=trucks.driverid;
```

In this query, FULL OUTER JOIN is implemented between tables geolocation and trucks where the driverid columns between both is the common factor.

Exercise

1. What is the schema on read?
2. Display truckid, driverid and model for every abnormal event.

```
select geolocation.driverid,trucks.truckid,trucks.model
from trucks JOIN geolocation
where geolocation.driverid=trucks.driverid AND event != "normal";
```

3. Display truckid, driverid, model and city for velocity > 25.

```
select trucks.driverid,trucks.truckid,trucks.model
from trucks LEFT JOIN geolocation
where geolocation.driverid=trucks.driverid AND velocity>25;
```

4. Display complete record of the trucks for events where they were on unsafe following distance.

```
select * from trucks FULL OUTER JOIN geolocation
where geolocation.driverid=trucks.driverid AND event = "unsafe following distance";
```

5. Explain the 4 types of joins in HiveQL and when would you use each of them?
6. What are the differences between INNER JOIN and LEFT JOIN?

Delete

Deleting data conditionally from within the table.

```
DELETE FROM geolocation WHERE city = "Santa Rosa";
```

Dropping tables

```
DROP TABLE trucks;  
DROP TABLE geolocation;
```

Partitioning in hive

Partitioning is similar to indexing in relational databases, we will now see how to create partitions and how they are represented in hdfs.

We create partitions so we can query a subset of data and don't have to go through all the data.

Like in the dataset before, if we only want to see the data from only one city we have to go through all the data, with partitions we can put data for different cities in different partitions so our query runs faster if we only want to analyse data from one city.

Lets create an order table

```
create database erp;  
  
Use erp;  
  
create table orders(id string, customer_id string, product_id string,  
quantity int, amount double, zipcode char(5))  
partitioned by (state char(2))  
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
```

In the above query state is a virtual column and the table is partitioned on it. Hadoop just used sub folders to represent the value of state.

Use the below query to see the created table (show partitions not return anything until we have

entered data into the table)

```
describe orders;
show partitions orders;
```

Now let's add some data in :

```
insert into orders
partition(state="CA")
values
("o1","c1","p1",1,379.99,"90210"),
("o2","c2","p2",1,8.99,"90002"),
("o3","c3","p100",1,100,"94305"),
("o4","c4","p992",1,669,"94304");
```

Now let's see how this data is stored in hdfs;

```
hadoop fs -ls /user/hive/warehouse/erp.db/orders
hadoop fs -ls /user/hive/warehouse/erp.db/orders/state=CA
```

```
[root@quickstart /]# hadoop fs -ls /user/hive/warehouse/erp.db/orders
Found 1 items
drwxrwxrwx - cludera supergroup          0 2020-05-27 19:38 /user/hive/warehouse/erp.db/orders/state=CA
[root@quickstart /]#
```

Hive has created a folder named **state=CA** and has pushed the data for ca partition into that, and entering data in other partitions will result in new folders being created. Let's add some more data into partitions.

```
insert into orders
partition(state="WA")
values
("o11","c100","p101",1,379.99,"98001"),
("o22","c200","p201",1,8.99,"98015"),
("o33","c3300","p100",1,100,"98052"),
("o44","c4440","p992",1,669,"98052");
insert into orders
partition(state="NJ")
values
("o200","c333","p101",2,50,"07301"),
("o400","c44","p992",1,99,"07308");
```

Checking into hdfs we can see that for each partition hive is creating a new directory.

```
[root@quickstart /]# hadoop fs -ls /user/hive/warehouse/erp.db/orders
Found 3 items
drwxrwxrwx - cloudera supergroup 0 2020-05-27 20:01 /user/hive/warehouse/erp.db/orders/state=CA
drwxrwxrwx - cloudera supergroup 0 2020-05-27 20:02 /user/hive/warehouse/erp.db/orders/state=NJ
drwxrwxrwx - cloudera supergroup 0 2020-05-27 20:02 /user/hive/warehouse/erp.db/orders/state=WA
[root@quickstart /]#
```

Running the show partition command now will show 3 partitions.

```
hive> show partitions orders;
OK
state=CA
state=NJ
state=WA
Time taken: 0.252 seconds, Fetched: 3 row(s)
```

We can also see data by doing cat on the file inside these partitions (update the below file name accordingly)

```
hadoop fs -cat /user/hive/warehouse/erp.db/orders/state=NJ/000000_0
```

Review:

Making partition will give us speed only if we use the query accordingly, in our case if we want to alkalize all the orders in NJ state our query will only look in that directory, but doing a **select *** on **orders** will give us no performance advantage.

We have to user partitioning intelligently with column that will occur the most in where clauses

Exercise

1. Will below query provide any performance advantage on the above scenario why or why not.

```
select * from orders;
```

2. Will below query provide any performance advantage on the above scenario why or why not.

```
select * from orders where state = "WA" and amount > 10;
```

3. Will below query provide any performance advantage on the above scenario why or why not.

```
select * from orders where (state="WA" or state="CA") and amount > 100.0;
```

4. Will below query provide any performance advantage on the above scenario why or why not.

```
select * from orders where product_id like("p%");
```

Let's look at more ways to load data into our partitions.

Loading data from file

Let's open a text editor and put in the below csv and save it as orders_CT.csv (CT is for connecticut)

```
cd /home/cloudera/Desktop/  
mkdir orders_data  
cd orders_data  
vi orders_CT.csv
```

Press i and paste in the below data (notice the state data will not be in the file)

```
o12,c101,p101,1,400,06010  
o13,c201,p201,1,56.7,06911
```

Press **esc** then **:** then write **wq** and then press **enter**

```
cat /home/cloudera/Desktop/orders_data/orders_CT.csv
```

```
load data local inpath '/home/cloudera/Desktop/orders_data/orders_CT.csv'  
into table orders partition (state="CT");
```

Check if the data is loaded and a new partition is created.

```
show partitions orders;  
Select * from orders;
```

Dropping a partition

```
ALTER TABLE orders DROP IF EXISTS PARTITION(state = "CA");
```

Trade offs

To partitions external tables the data must be divided into folders already we will later see how we can import data from an sql database. We are not covering partitioning external tables at the moment.

Dynamic partitioning

In dynamic partitioning a new partition is created when hive encounters a new value for a column , by default dynamic partition is turned off, we will look at this in an exercise later.

Bucketing

Bucketing uses a hash function to divide rows into separate files. To speed up lookup.

Example, There are 3 buckets/files which an entry will fall into. We are using the hash function “%3” which will look at the remainder after dividing by 3, we are going to apply it on an id which is a sequence (1,2,3,4,5,6,7,8....)

Below we are seeing which file and it will go into

When **id** is **1** we will have **1%3=1** so the entry will go in the bucket with hash 1

When **id** is **2** we will have **2%3=2** so the entry will go in the bucket with hash 2

When **id** is **3** we will have **3%3=0** so the entry will go in the bucket with hash 0

4%3=1

5%3=2

6%3=0

7%3=1

8%3=2

9%3=0

10%3=1

This way we can break the values into three separate files making the look up faster just by looking at the id we will know where the corresponding row will be.

Now lets look at the example implementing the above scenario.

Lets create a csv file with data for products (inside the docker container)

```
cd /home/cloudera/Desktop/  
mkdir products_data  
cd products_data  
vi products_data.csv
```

Put the data in the above created file. (Press **esc** then **:** then write **wq** and then press **enter** to save and exit)

```
1,iphone,379.99,mobile  
2,doll,9.88,toys  
3,samsungphone,23.34,mobile  
4,jeans,73,fashion  
5,nokiaphone,45.45,mobile  
6,truck,100,toys  
7,makeup,74,fashion  
8,earrings,23,fashion  
9,chair,235,furniture  
10,table,473,furniture  
12,shirt,37,fashion  
11,waterpistol,9,toys  
13,glasses,53,fashion  
14,lipgloss,13,fashion  
15,cap,97,fashion
```

First we are going to turn on bucketing and create a database (if you are using GUI then switch the database)

```
Create database inventory;  
Use inventory;  
SET hive.enforce.bucketing=true;
```

Here we are going to create 2 tables and then load data from one table into another.

```
Create table products_no_buckets  
(id int,  
name string,  
cost double,  
category string)  
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
```

```
Create table products
(id int,
name string,
cost double,
category string)
clustered by (id) into 4 buckets
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
```

We are now going to load data into our products_no_bucket table

```
load data local inpath
'/home/cloudera/Desktop/products_data/products_data.csv' into table
products_no_buckets;
```

```
from products_no_buckets
insert into table products
select id, name , cost ,category;
```

Now let's look at the created files

```
hadoop fs -ls /user/hive/warehouse/inventory.db/products
```

We can see 4 files are created in the products directory

```
[root@quickstart conf]# hadoop fs -ls /user/hive/warehouse/inventory.db/products
Found 5 items
drwxrwxrwx - cloudera supergroup      0 2020-05-29 01:30 /user/hive/warehouse/inventory.db/products/.hive-stag
-rwxrwxrwx 1 cloudera supergroup    45 2020-05-29 01:30 /user/hive/warehouse/inventory.db/products/000000_0
-rwxrwxrwx 1 cloudera supergroup    73 2020-05-29 01:30 /user/hive/warehouse/inventory.db/products/000001_0
-rwxrwxrwx 1 cloudera supergroup    61 2020-05-29 01:30 /user/hive/warehouse/inventory.db/products/000002_0
-rwxrwxrwx 1 cloudera supergroup    74 2020-05-29 01:30 /user/hive/warehouse/inventory.db/products/000003_0
[root@quickstart conf]#
```

Looking at the no 1 file we can see it has id 8 and 4 which are divisible by 4 (look at other files)

```
[root@quickstart conf]# hadoop fs -cat /user/hive/warehouse/inventory.db/products/000000_0
8,earrings,23.0,fashion
4,jeans,73.0,fashion
[root@quickstart conf]#
```

Exercise

1. Search and write what the below flags do(set these flags after selecting the inventory DB they will be used in the next question)

```
set hive.exec.dynamic.partition=true;
set hive.exec.dynamic.partition.mode=nonstrict;
```

```
set hive.exec.max.dynamic.partitions.pernode=1000;
set hive.enforce.bucketing = true;
```

2. Run the below scenario and explaining the directory and file structure created

```
Create table products_
(id int,
name string,
cost double)
PARTITIONED BY (category VARCHAR(64))
CLUSTERED BY (id) INTO 4 BUCKETS
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
```

```
from products_no_buckets
insert into table products_
PARTITION (category)
select id, name , cost ,category;
```

Search and write about the below flags

```
SET hive.optimize.bucketmapjoin=true;
set hive.auto.convert.sortmerge.join=true;
set hive.optimize.bucketmapjoin = true;
set hive.optimize.bucketmapjoin.sortedmerge = true;
```

Cloudera-manager

Cloudera manager allows you to manage services and view their health , when you start **cm()** you will need to manually start services using the cm. (you might need to also add --force flag in some cases)

```
cd /home/cloudera/
./cloudera-manager --express
```

Cloudera manager commands

```
sudo service cloudera-scm-server start
sudo service cloudera-scm-server stop
sudo service cloudera-scm-server restart
```

https://docs.cloudera.com/documentation/enterprise/5-13-x/topics/cm_ag_server.html

Importing Data Into Hive From MYSQL

Scenario (By Ali Asghar Marvi):

- 1) Setup MySQL database.
- 2) Use Sqoop to pull data to Hive
- 3) Check on Hue Interface

Setting Up MYSQL

- Image uses CentOS 6.6, verify it using this command at root@quickstart

```
[root@quickstart /]# cat /etc/*release
CentOS release 6.6 (Final)
LSB_VERSION=base-4.0-amd64:base-4.0-noarch:core-4.0-amd64:core-4.0-noarch
CentOS release 6.6 (Final)
CentOS release 6.6 (Final)
[root@quickstart /]#
```

- Tutorial requires to update OS but I am not going to do that and risk already running and configured services. Guess I will begin with the installation right away.

```
sudo yum install mysql-server
```

- Now start the service:

```
sudo service mysqld start
```

- Run

```
sudo mysql_secure_installation
```

This is not necessary but in order to set a password or users it's your call to execute it or not.

- Now run mysql in command prompt:

```
mysql -u root -p
```

- Oh and also keep sql dump ready for hr database. Link at :

<https://github.com/nomemory/hr-schema-mysql/blob/master/hr-schema-mysql.sql>

- Now in order to execute a .sql file in already running image, I kinda went lengths to download the .sql. I will mention my approach, if you have a shorter one please use that. Download and install these dependencies first:

```
Sudo yum install wget
Sudo yum install unzip
```

- Now wget master.zip from github

```
Wget https://github.com/nomemory/hr-schema-mysql/archive/master.zip
```

Unzip master.zip

```
Unzip master.zip
```

Go to folder where file is extracted:

```
Cd -hr-schema-mysql-master
```

Execute mysql command:

```
Mysql -u root -p < hr-schema-mysql.zip
```

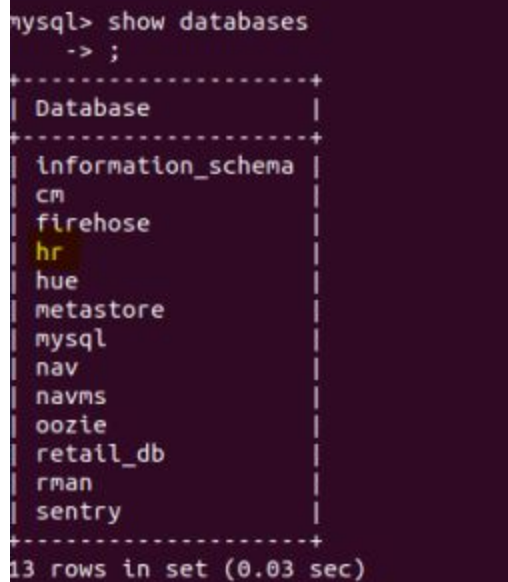
Now connect to mysql bash using:

```
Mysql -u root -p
```

Execute sql query:

```
Show databases;
```

- You will find hr in the list:



```
mysql> show databases
-> ;
+-----+
| Database |
+-----+
| information_schema |
| cm |
| firehose |
| hr |
| hue |
| metastore |
| mysql |
| nav |
| navms |
| oozie |
| retail_db |
| rman |
| sentry |
+-----+
13 rows in set (0.03 sec)
```

Now run some queries to ensure entire database is there:

```
USE hr;
select * from employees;
select * from departments;
```

```
exit;
```

MYSQL HAS BEEN SETUP WITH DATABASE TO PLAY WITH. NOW LETS MOVE ON TO NEXT STEP. "SQOOPING IT TO HDFS"

Scenario 2: (Importing MySQL hr data to hive)

Hue interface gives an option to do so using Sqoop Transfer, but the feature wasnt working at my end. Hence I decided to use terminal.

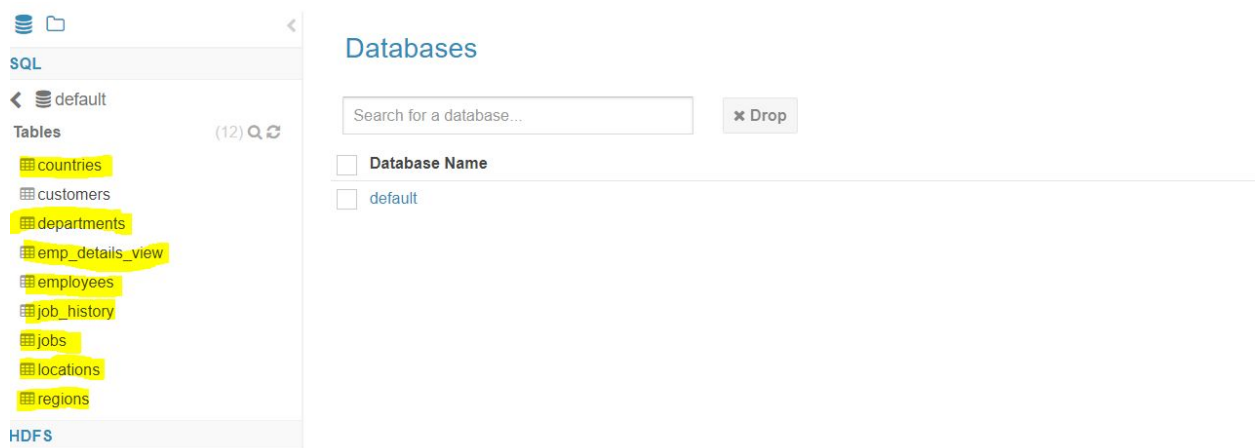
Execute the following command:

```
Sqoop import-all-tables --connect jdbc:mysql://localhost:3360/hr  
--username root --password root --hive-import --autoreset-to-one-mapper  
--hive-overwrite --create-hive-table --warehouse-dir  
/user/hive/warehouse/hr_hive.db
```

Give it a while where each of the tables is being transferred to hive.

Now launch Hue UI (or log out and login again!). Select 'MetaStore Tables' from Data Browser.

Click default and you will see your newly added tables:



You can also preview them:

Data sample for countries

	countries.country_id	countries.country_name	countries.region_id
1	AR	Argentina	2
2	AU	Australia	3
3	BE	Belgium	1
4	BR	Brazil	2
5	CA	Canada	2
6	CH	Switzerland	1
7	CN	China	3
8	DE	Germany	1
9	DK	Denmark	1
10	EG	Egypt	4
11	FR	France	1
12	HK	HongKong	3
13	IL	Israel	4

Activate Windows

Data sample for customers

	customers.id	customers.name	customers.email_preferences	customers.addresses
1	75012	Dorothy Wilk	["email_format":"html","frequency":"weekly","categories":{"promos":true,"surveys":false}]	["shipping":{"street_1":"1102 Sussex Court","street_2":"","city":"Waco","state":"TX","zip_code":"76710"},"b
2	17254	Martin Johnson	["email_format":"text","frequency":"daily","categories":{"promos":true,"surveys":true}]	["shipping":{"street_1":"158 Jadewood Drive","street_2":"Apt 2","city":"Gary","state":"IN","zip_code":"4640
3	12532	Melvin Garcia	["email_format":"html","frequency":"daily","categories":{"promos":true,"surveys":false}]	["shipping":{"street_1":"1740 Hawks Nest Lane","street_2":"","city":"Waco","state":"TX","zip_code":"76710"
4	42632	Raymond S. Vestal	["email_format":"html","frequency":"monthly","categories":{"promos":true,"surveys":false}]	["billing":{"street_1":"2913 Chestnut Street","street_2":"","city":"Menasha","state":"WI","zip_code":"54952"
5	77913	Betty J. Giambrone	["email_format":"text","frequency":"monthly","categories":{"promos":false,"surveys":false}]	["billing":{"street_1":"3303 John Calvin Drive","street_2":"","city":"Jackson","state":"TN","zip_code":"38301"
6	38807	Rebecca T. Johnson	["email_format":"html","frequency":"monthly","categories":{"promos":true,"surveys":false}]	["billing":{"street_1":"1116 Hunken Avenue","street_2":"","city":"Osceola","state":"IN","zip_code":"46561"},"s
7	71843	David B. Allison	["email_format":"text","frequency":"daily","categories":{"promos":true,"surveys":false}]	["billing":{"street_1":"3947 Heavener Avenue","street_2":"","city":"Jackson","state":"TN","zip_code":"38301"
8	67099	Jay N. Weaver	["email_format":"text","frequency":"daily","categories":{"promos":false,"surveys":false}]	["billing":{"street_1":"3286 Windmill Road","street_2":"","city":"Waco","state":"TX","zip_code":"76710"},"sh
9	83510	Carol B. Houser	["email_format":"html","frequency":"never","categories":{"promos":false,"surveys":false}]	["billing":{"street_1":"1396 Alexander Drive","street_2":"","city":"Chicago","state":"IL","zip_code":"60601"},"
10	48072	Octaviana Gutierrez	["email_format":"text","frequency":"never","categories":{"promos":false,"surveys":true}]	["billing":{"street_1":"4338 Denton Street","street_2":"","city":"Providence","state":"RI","zip_code":"02906"
11	89630	Michelle Coffman	["email_format":"text","frequency":"daily","categories":{"promos":true,"surveys":false}]	["billing":{"street_1":"1325 Douglas Dairy Road","street_2":"","city":"Jackson","state":"TN","zip_code":"383
12	98751	Heather L. Southerland	["email_format":"text","frequency":"weekly","categories":{"promos":true,"surveys":false}]	["billing":{"street_1":"4630 Peck Street","street_2":"","city":"Providence","state":"RI","zip_code":"02906"},"
13	88702	Diane Lee	["email_format":"html","frequency":"monthly","categories":{"promos":false,"surveys":true}]	["biling":{"street_1":"2796 Southaven Avenue","street_2":"","city":"Providence","state":"RI","zip_code":"029

Ok

Data sample for employees

	employees.employee_id	employees.first_name	employees.last_name	employees.email	employees.phone_number	employees.hire_date	employees.job_id	employees.salary	employees.commission_pct	employee
1	100	Steven	King	SKING	515 123 4567	1987-06-17	AD_PRES	24000	NULL	NULL
2	101	Neena	Kochhar	NKOCHHAR	515 123 4568	1989-09-21	AD_VP	17000	NULL	100
3	102	Lex	De Haan	LDEHAAN	515 123 4569	1993-01-13	AD_VP	17000	NULL	100
4	103	Alexander	Hunold	AHUNOLD	590 423 4567	1990-01-03	IT_PROG	9000	NULL	102
5	104	Bruce	Ernst	BERNST	590 423 4568	1991-05-21	IT_PROG	6000	NULL	103
6	105	David	Austin	DAUSTIN	590 423 4569	1997-06-25	IT_PROG	4800	NULL	103
7	106	Valli	Pataballa	VPATABAL	590 423 4560	1998-02-05	IT_PROG	4800	NULL	103
8	107	Diana	Lorentz	DLORENTZ	590 423 5567	1999-02-07	IT_PROG	4200	NULL	103
9	108	Nancy	Greenberg	NGREENBE	515 124 4569	1994-08-17	FL_MGR	12000	NULL	101
10	109	Daniel	Faviet	DFAVIET	515 124 4169	1994-08-16	FL_ACCOUNT	9000	NULL	108
11	110	John	Chen	JCHEN	515 124 4269	1997-09-28	FL_ACCOUNT	8200	NULL	108
12	111	Ismael	Sciarra	ISCIARRA	515 124 4369	1997-09-30	FL_ACCOUNT	7700	NULL	108
13	112	Jose Manuel	Urman	JMURMAN	515 124 4469	1998-03-07	FL_ACCOUNT	7800	NULL	108

Ok

Playing with Hive Query Editor:

The screenshot shows the Hive Query Editor interface. The top navigation bar includes 'HIVE', 'Query Editors', 'Data Browsers', 'Workflows', 'Search', 'Security', 'File Browser', 'Job Browser', 'cloudera', and user icons. The 'Query Editor' tab is active, showing 'My Queries', 'Saved Queries', and 'History'. On the left, a sidebar lists 'Tables' (12) including countries, customers, departments, emp_details_view, employees, job_history, jobs, locations, regions, sample_07, sample_08, and web_logs. The main area contains a query editor with the text: `1 SELECT * FROM employees WHERE salary<30000;`. Below the editor are buttons for 'Execute', 'Save as...', 'Explain', 'Format', 'or create a', and 'New query'. Below these buttons are tabs for 'Recent queries', 'Query', 'Log', 'Columns', 'Results', and 'Chart'. The 'Results' tab is selected, displaying a table with 9 rows of employee data. A watermark 'Activate Windows. Go to Settings to activate Windows.' is visible in the bottom right corner of the results area.

	employees.employee_id	employees.first_name	employees.last_name	employees.email	employees.phone_number	employees.hire_date	employees.job_id	em
1	100	Steven	King	SKING	515.123.4567	1987-06-17	AD_PRES	
2	101	Neena	Kochhar	NKOCHHAR	515.123.4568	1989-09-21	AD_VP	
3	102	Lex	De Haan	LDEHAAN	515.123.4569	1993-01-13	AD_VP	
4	103	Alexander	Hunold	AHUNOLD	590.423.4567	1990-01-03	IT_PROG	
5	104	Bruce	Ernst	BERNST	590.423.4568	1991-05-21	IT_PROG	
6	105	David	Austin	DAUSTIN	590.423.4569	1997-06-25	IT_PROG	
7	106	Valli	Pataballa	VPATABAL	590.423.4560	1998-02-05	IT_PROG	
8	107	Diana	Lorentz	DLORENTZ	590.423.5567	1999-02-07	IT_PROG	
9	108	Nancy	Greenberg	NGREENBE	515.124.4569	1994-08-17	IT_PROG	

HR Data Query Exercises on HIVE:

Display details of jobs where the minimum salary is greater than 10000.

```
SELECT * FROM JOBS WHERE MIN_SALARY > 10000
```

Display the first name and join date of the employees who joined between 2002 and 2005.

```
SELECT FIRST_NAME, HIRE_DATE FROM EMPLOYEES  
WHERE TO_CHAR(HIRE_DATE, 'YYYY') BETWEEN 2002 AND 2005 ORDER BY HIRE_DATE
```

Display first name and join date of the employees who is either IT Programmer or Sales Man.

```
SELECT FIRST_NAME, HIRE_DATE  
FROM EMPLOYEES WHERE JOB_ID IN ('IT_PROG', 'SA_MAN')
```

Display employees who joined after 1st January 2008.

```
SELECT * FROM EMPLOYEES where hire_date > '01-jan-2008'
```

Display details of employee with ID 150 or 160.

```
SELECT * FROM EMPLOYEES WHERE EMPLOYEE_ID in (150,160)
```

Display first name, salary, commission pct, and hire date for employees with salary less than 10000.

```
SELECT FIRST_NAME, SALARY, COMMISSION_PCT, HIRE_DATE FROM EMPLOYEES WHERE SALARY < 10000
```

Display job Title, the difference between minimum and maximum salaries for jobs with max salary in the range 10000 to 20000.

```
SELECT JOB_TITLE, MAX_SALARY-MIN_SALARY DIFFERENCE FROM JOBS WHERE MAX_SALARY BETWEEN 10000  
AND 20000
```

Display first name, salary, and round the salary to thousands.

```
SELECT FIRST_NAME, SALARY, ROUND(SALARY, 3) FROM EMPLOYEES
```

Display details of jobs in the descending order of the title.

```
SELECT * FROM JOBS ORDER BY JOB_TITLE
```