

Module 5

Understanding Hive

Thanachart Numnonda, Executive Director, IMC Institute

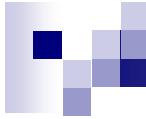
Thanisa Numnonda, Faculty of Information Technology,
King Mongkut's Institute of Technology Ladkrabang

Introduction

A Petabyte Scale Data Warehouse Using Hadoop

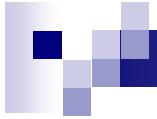


Hive is developed by Facebook, designed to enable easy data summarization, ad-hoc querying and analysis of large volumes of data. It provides a simple query language called Hive QL, which is based on SQL



What Hive is NOT

Hive is not designed for online transaction processing and does not offer real-time queries and row level updates. It is best used for batch jobs over large sets of immutable data (like web logs, etc.).



Sample HiveQL

The Query compiler uses the information stored in the metastore to convert SQL queries into a sequence of map/reduce jobs, e.g. the following query

```
SELECT * FROM t where t.c = 'xyz'
```

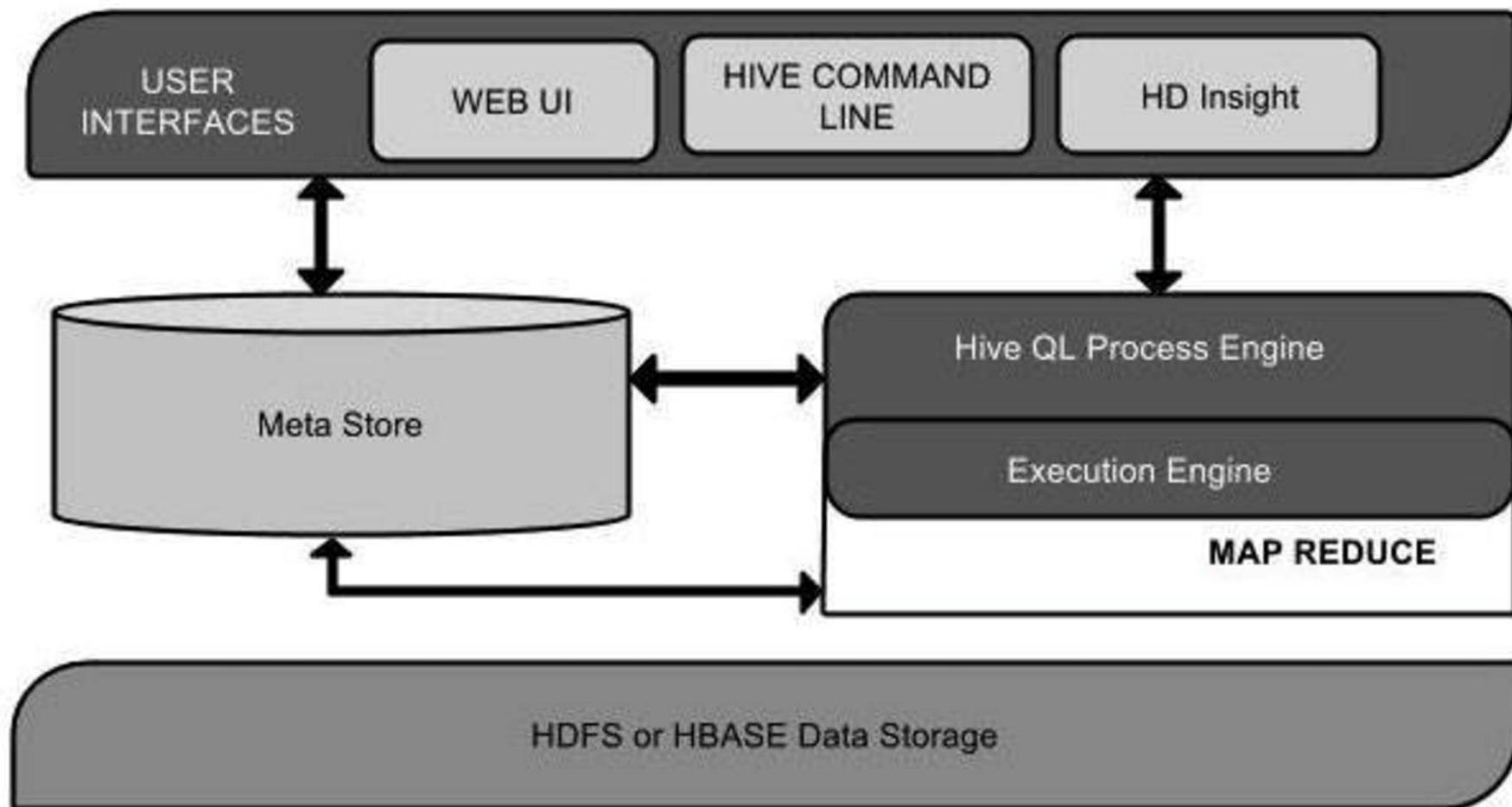
```
SELECT t1.c2 FROM t1 JOIN t2 ON (t1.c1 = t2.c1)
```

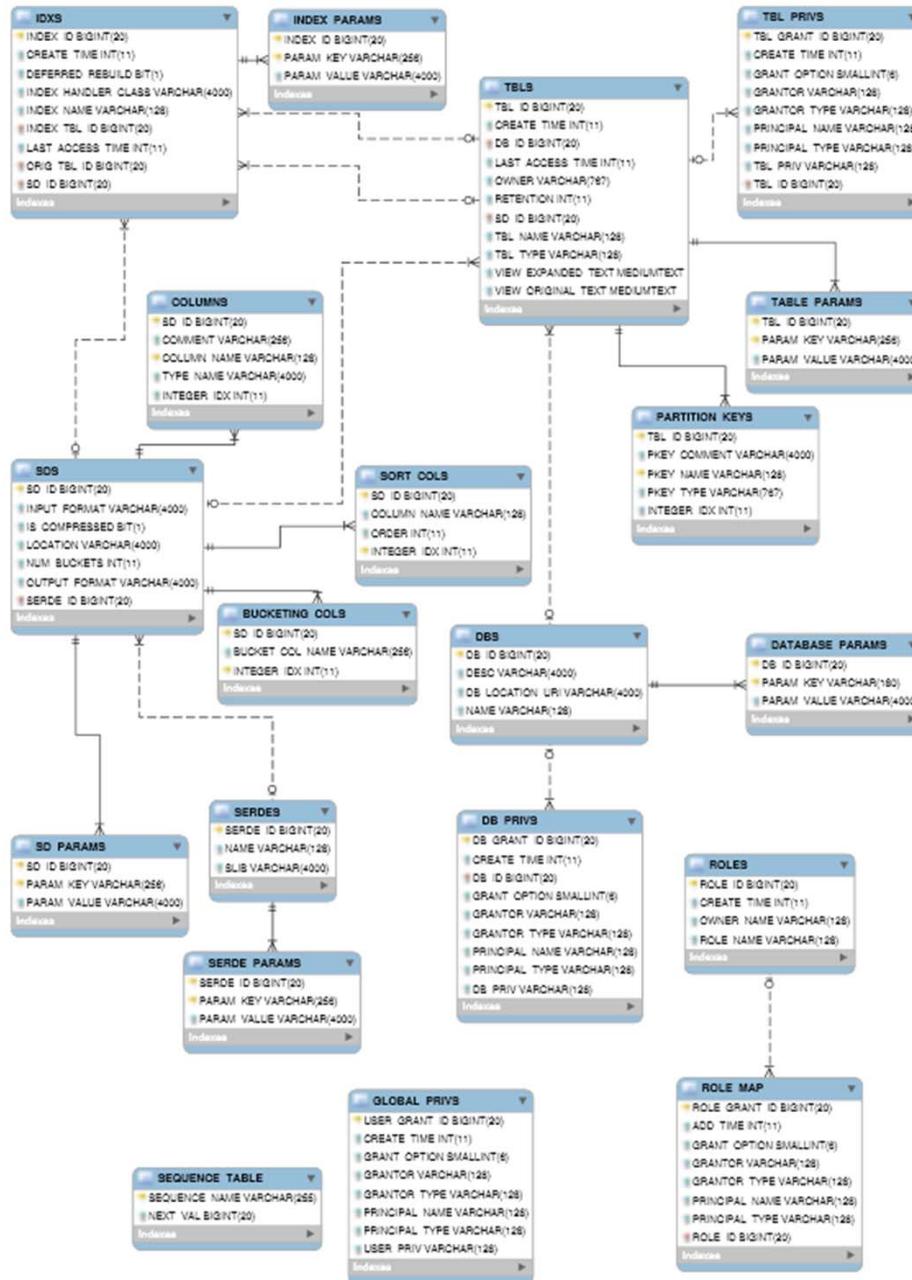
```
SELECT t1.c1, count(1) from t1 group by t1.c1
```

System Architecture and Components

User Interface	Hive is a data warehouse infrastructure software that can create interaction between user and HDFS. The user interfaces that Hive supports are Hive Web UI, Hive command line, and Hive HD Insight (In Windows server).
Meta Store	Hive chooses respective database servers to store the schema or Metadata of tables, databases, columns in a table, their data types, and HDFS mapping.
HiveQL Process Engine	HiveQL is similar to SQL for querying on schema info on the Metastore. It is one of the replacements of traditional approach for MapReduce program. Instead of writing MapReduce program in Java, we can write a query for MapReduce job and process it.
Execution Engine	The conjunction part of HiveQL process Engine and MapReduce is Hive Execution Engine. Execution engine processes the query and generates results as same as MapReduce results. It uses the flavor of MapReduce.
HDFS or HBase	Hadoop distributed file system or HBase are the data storage techniques to store data into file system.

Architecture Overview

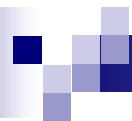




Hive Metastore

Hive Metastore is a repository to keep all Hive metadata; Tables and Partitions definition.

By default, Hive will store its metadata in Derby DB



Hive Built in Functions

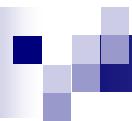
hive.apache.org

Return Type	Function Name (Signature)	Description
BIGINT	round(double a)	returns the rounded BIGINT value of the double
BIGINT	floor(double a)	returns the maximum BIGINT value that is equal or less than the double
BIGINT	ceil(double a)	returns the minimum BIGINT value that is equal or greater than the double
double	rand(), rand(int seed)	returns a random number (that changes from row to row). Specifying the seed will make sure the generated random number sequence is deterministic.
string	concat(string A, string B,...)	returns the string resulting from concatenating B after A. For example, concat('foo', 'bar') results in 'foobar'. This function accepts arbitrary number of arguments and return the concatenation of all of them.
string	substr(string A, int start)	returns the substring of A starting from start position till the end of string A. For example, substr('foobar', 4) results in 'bar'
string	substr(string A, int start, int length)	returns the substring of A starting from start position with the given length e.g. substr('foobar', 4, 2) results in 'ba'

Hive Built in Functions (Cont.)

hive.apache.org

Return Type	Function Name (Signature)	Description
string	upper(string A)	returns the string resulting from converting all characters of A to upper case e.g. upper('fOoBaR') results in 'FOOBAR'
string	ucase(string A)	Same as upper
string	lower(string A)	returns the string resulting from converting all characters of B to lower case e.g. lower('fOoBaR') results in 'foobar'
string	lcase(string A)	Same as lower
string	trim(string A)	returns the string resulting from trimming spaces from both ends of A e.g. trim(' foobar ') results in 'foobar'
string	ltrim(string A)	returns the string resulting from trimming spaces from the beginning(left hand side) of A. For example, ltrim(' foobar ') results in 'foobar '
string	rtrim(string A)	returns the string resulting from trimming spaces from the end(right hand side) of A. For example, rtrim(' foobar ') results in ' foobar'
string	regexp_replace(string A, string B, string C)	returns the string resulting from replacing all substrings in B that match the Java regular expression syntax(See Java regular expressions syntax) with C. For example, regexp_replace('foobar', 'oolar',) returns 'fb'



Hive Built in Functions (Cont.)

hive.apache.org

Return Type	Function Name (Signature)	Description
string	from_unixtime(int unixtime)	convert the number of seconds from unix epoch (1970-01-01 00:00:00 UTC) to a string representing the timestamp of that moment in the current system time zone in the format of "1970-01-01 00:00:00"
string	to_date(string timestamp)	Return the date part of a timestamp string: to_date("1970-01-01 00:00:00") = "1970-01-01"
int	year(string date)	Return the year part of a date or a timestamp string: year("1970-01-01 00:00:00") = 1970, year("1970-01-01") = 1970
int	month(string date)	Return the month part of a date or a timestamp string: month("1970-11-01 00:00:00") = 11, month("1970-11-01") = 11
int	day(string date)	Return the day part of a date or a timestamp string: day("1970-11-01 00:00:00") = 1, day("1970-11-01") = 1
string	get_json_object(string json_string, string path)	Extract json object from a json string based on json path specified, and return json string of the extracted json object. It will return null if the input json string is invalid

Hive Aggregate Functions (Cont.)

Return Type	Aggregation Function Name (Signature)	Description
BIGINT	count(*), count(expr), count(DISTINCT expr[, expr_...])	count(*) - Returns the total number of retrieved rows, including rows containing NULL values; count(expr) - Returns the number of rows for which the supplied expression is non-NULL; count(DISTINCT expr[, expr]) - Returns the number of rows for which the supplied expression(s) are unique and non-NULL.
DOUBLE	sum(col), sum(DISTINCT col)	returns the sum of the elements in the group or the sum of the distinct values of the column in the group
DOUBLE	avg(col), avg(DISTINCT col)	returns the average of the elements in the group or the average of the distinct values of the column in the group
DOUBLE	min(col)	returns the minimum value of the column in the group
DOUBLE	max(col)	returns the maximum value of the column in the group

Examples

```
hive> SELECT floor(2.6);      => 2
```

```
hive> SELECT ceil(2.1);      => 3
```

```
hive> SELECT round(2.1);     => 2.0
```

```
hive> SELECT round(2.5);     => 3.0
```

Running Hive

Hive Shell

Interactive
hive

Script
hive -f myscript

Inline
*hive -e 'SELECT * FROM mytable'*

Hive Tables

- Managed- **CREATE TABLE**
 - LOAD- File moved into Hive's data warehouse directory
 - DROP- Both data and metadata are deleted.
- External- **CREATE EXTERNAL TABLE**
 - LOAD- No file moved
 - DROP- Only metadata deleted
 - Use when sharing data between Hive and Hadoop applications or you want to use multiple schema on the same data

Hive External Table

- `CREATE EXTERNAL TABLE external_Table (dummy STRING)`
- `LOCATION '/user/notroot/external_table';`

Dropping External Table using Hive:-

Hive will delete metadata from metastore

Hive will NOT delete the HDFS file

You need to manually delete the HDFS file

HiveQL and MySQL Comparison

Metadata

Function	MySQL	HiveQL
Selecting a database	USE database;	USE database;
Listing databases	SHOW DATABASES;	SHOW DATABASES;
Listing tables in a database	SHOW TABLES;	SHOW TABLES;
Describing the format of a table	DESCRIBE table;	DESCRIBE (FORMATTED EXTENDED) table;
Creating a database	CREATE DATABASE db_name;	CREATE DATABASE db_name;
Dropping a database	DROP DATABASE db_name;	DROP DATABASE db_name (CASCADE);

HiveQL and MySQL Query Comparison

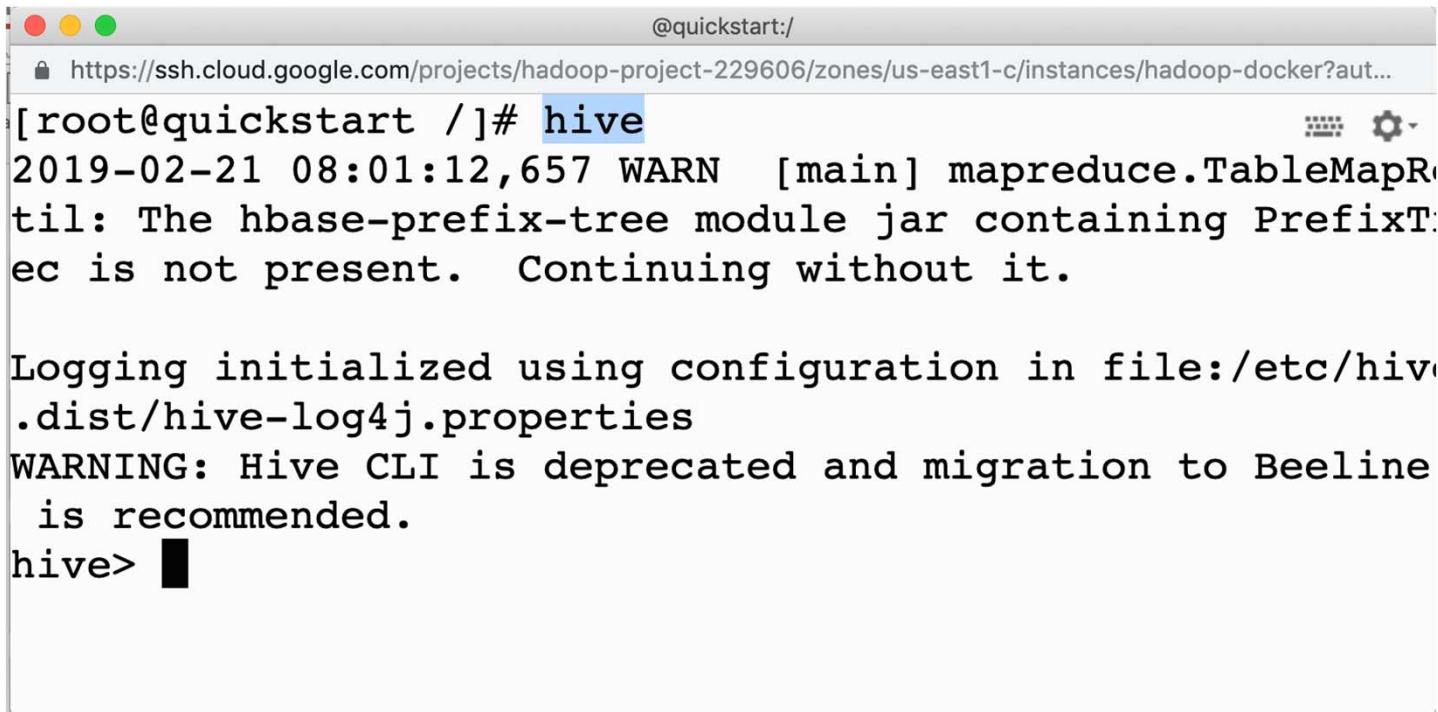
Query

Function	MySQL	HiveQL
Retrieving information	<code>SELECT from_columns FROM table WHERE conditions;</code>	<code>SELECT from_columns FROM table WHERE conditions;</code>
All values	<code>SELECT * FROM table;</code>	<code>SELECT * FROM table;</code>
Some values	<code>SELECT * FROM table WHERE rec_name = "value";</code>	<code>SELECT * FROM table WHERE rec_name = "value";</code>
Multiple criteria	<code>SELECT * FROM table WHERE rec1="value1" AND rec2="value2";</code>	<code>SELECT * FROM TABLE WHERE rec1 = "value1" AND rec2 = "value2";</code>
Selecting specific columns	<code>SELECT column_name FROM table;</code>	<code>SELECT column_name FROM table;</code>
Retrieving unique output records	<code>SELECT DISTINCT column_name FROM table;</code>	<code>SELECT DISTINCT column_name FROM table;</code>
Sorting	<code>SELECT col1, col2 FROM table ORDER BY col2;</code>	<code>SELECT col1, col2 FROM table ORDER BY col2;</code>
Sorting backward	<code>SELECT col1, col2 FROM table ORDER BY col2 DESC;</code>	<code>SELECT col1, col2 FROM table ORDER BY col2 DESC;</code>
Counting rows	<code>SELECT COUNT(*) FROM table;</code>	<code>SELECT COUNT(*) FROM table;</code>
Grouping with counting	<code>SELECT owner, COUNT(*) FROM table GROUP BY owner;</code>	<code>SELECT owner, COUNT(*) FROM table GROUP BY owner;</code>
Maximum value	<code>SELECT MAX(col_name) AS label FROM table;</code>	<code>SELECT MAX(col_name) AS label FROM table;</code>
Selecting from multiple tables (Join same table using alias w/"AS")	<code>SELECT pet.name, comment FROM pet, event WHERE pet.name = event.name;</code>	<code>SELECT pet.name, comment FROM pet JOIN event ON (pet.name = event.name);</code>



Hands-On: Loading Data using Hive

Start Hive



A screenshot of a terminal window titled '@quickstart:/'. The URL in the address bar is <https://ssh.cloud.google.com/projects/hadoop-project-229606/zones/us-east1-c/instances/hadoop-docker?aut...>. The terminal output shows:

```
[root@quickstart /]# hive
2019-02-21 08:01:12,657 WARN  [main] mapreduce.TableMapReduceUtil: The hbase-prefix-tree module jar containing PrefixTree is not present. Continuing without it.

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.properties
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
hive> █
```

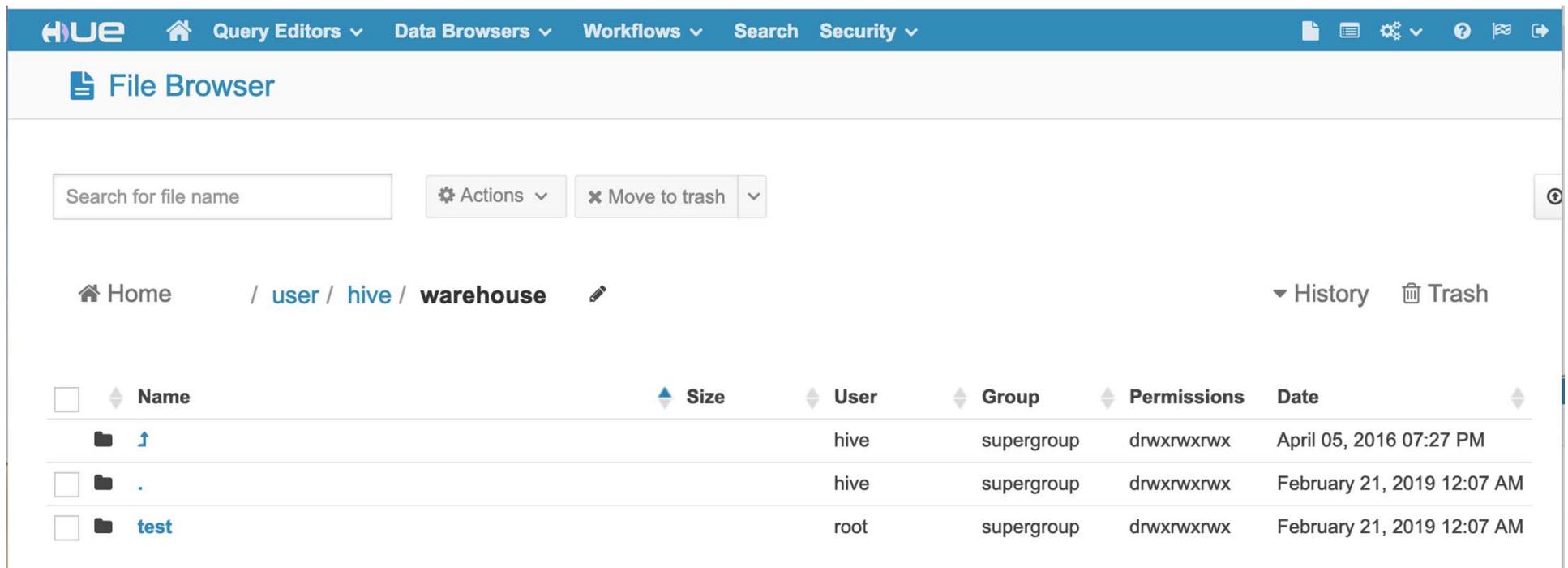
Quit from Hive

```
hive> quit;
```

Create Hive Table

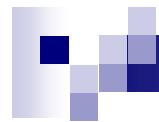
```
hive> CREATE TABLE test(id INT, country STRING) ROW FORMAT DELIMITED  
      FIELDS TERMINATED BY ',' STORED AS TEXTFILE;  
OK  
Time taken: 0.894 seconds  
hive> show tables;  
OK  
test  
Time taken: 0.139 seconds, Fetched: 1 row(s)  
hive> describe test;  
OK  
id          int  
country      string  
Time taken: 0.146 seconds, Fetched: 2 row(s)
```

Reviewing Hive Table in HDFS



The screenshot shows the Hue File Browser interface. The top navigation bar includes links for Home, Query Editors, Data Browsers, Workflows, Search, Security, and various system icons. The main title is "File Browser". Below the title are search and action buttons: "Search for file name", "Actions", and "Move to trash". The breadcrumb navigation shows the path: Home / user / hive / warehouse. To the right of the breadcrumb are "History" and "Trash" buttons. The main content area is a table listing files and directories in the warehouse folder:

	Name	Size	User	Group	Permissions	Date
<input type="checkbox"/>	↑		hive	supergroup	drwxrwxrwx	April 05, 2016 07:27 PM
<input type="checkbox"/>	.		hive	supergroup	drwxrwxrwx	February 21, 2019 12:07 AM
<input type="checkbox"/>	test		root	supergroup	drwxrwxrwx	February 21, 2019 12:07 AM



Preparing Large Dataset

<http://grouplens.org/datasets/movielens/>

grouplens

about datasets publications blog

MovieLens

GroupLens Research has collected and made available rating data sets from the MovieLens web site (<http://movielens.org>). The data sets were collected over various periods of time, depending on the size of the set. Before using these data sets, please review their README files for the usage licenses and other details.

Help our research lab: Please [take a short survey](#) about the MovieLens datasets

MovieLens 100k

100,000 ratings from 1000 users on 1700 movies.

- [README.txt](#)
- [ml-100k.zip](#)
- [Index of unzipped files](#)

MovieLens 1M

1 million ratings from 6000 users on 4000 movies.

- [README.txt](#)

Datasets

[MovieLens](#)

[HetRec 2011](#)

[WikiLens](#)

[Book-Crossing](#)

[Jester](#)

[EachMovie](#)

MovieLen Dataset

1) Type command > `wget`

`http://files.grouplens.org/datasets/movielens/ml-100k.zip`

2) Type command > `yum install unzip`

3) Type command > `unzip ml-100k.zip`

4) Type command > `ls` เหมือนเอาชื่อไฟล์เก่า มาตั้งชื่อไฟล์ใหม่

5) Type command > `more ml-100k/u.user`

```
[root@quickstart guest1]# more ml-100k/u.user
1|24|M|technician|85711
2|53|F|other|94043
3|23|M|writer|32067
4|24|M|technician|43537
5|33|F|other|15213
6|42|M|executive|98101
7|57|M|administrator|91344
8|36|M|administrator|05201
9|29|M|student|01002
10|53|M|lawyer|90703
11|39|F|other|30329
```

Moving dataset to HDFS

- 1) Type command > `cd ml-100k`
- 2) Type command > `hdfs dfs -mkdir /user/cloudera/movielens`
- 3) Type command > `hdfs dfs -put u.user /user/cloudera/movielens`
- 4) Type command > `hdfs dfs -ls /user/cloudera/movielens`

```
[root@quickstart ml-100k]# hdfs dfs -ls /user/cloudera/movielens
Found 1 items
-rw-r--r--    1 root cloudera      22628 2019-02-21 08:25 /user/cloudera/
movielens/u.user
[root@quickstart ml-100k]# █
```

CREATE & SELECT Table

```
hive> CREATE EXTERNAL TABLE users (userid INT, age INT,  
    > gender STRING, occupation STRING, zipcode STRING) ROW FORMAT  
    > DELIMITED FIELDS TERMINATED BY '|' STORED AS TEXTFILE  
    > LOCATION '/user/cloudera/movielens';  
OK  
Time taken: 0.646 seconds  
hive> SELECT * FROM users;  
OK  


|   |    |   |               |       |
|---|----|---|---------------|-------|
| 1 | 24 | M | technician    | 85711 |
| 2 | 53 | F | other         | 94043 |
| 3 | 23 | M | writer        | 32067 |
| 4 | 24 | M | technician    | 43537 |
| 5 | 33 | F | other         | 15213 |
| 6 | 42 | M | executive     | 98101 |
| 7 | 57 | M | administrator | 91344 |
| 8 | 36 | M | administrator | 05201 |


```



Bay Area Bike Share (BABS)

<http://www.bayareabikeshare.com/open-data>

A screenshot of the Bay Area Bike Share website's open data page. The header features the BABS logo and navigation links: OPEN DATA, GIFT STORE, ABOUT, RESOURCES, APP, CONTACT, and LOGIN. Below the header are links for SIGN UP, HOW IT WORKS, SUGGEST A STATION, STATION MAP, and PRICING, along with social media icons for Facebook, Twitter, and Instagram.

OPEN DATA

Here you'll find Bay Area Bike Share's trip data for public use. So whether you're a designer, developer, or just plain curious, feel free to download it and bring it to life!

YEAR 1 DATA

(August 2013 - August 2014)

YEAR 2 DATA

(September 2014 - August 2015)

THE DATA

Each trip is anonymized and includes:

- Bike number
- Trip start day and time
- Trip end day and time

Preparing a bike data

1. Get dataset from

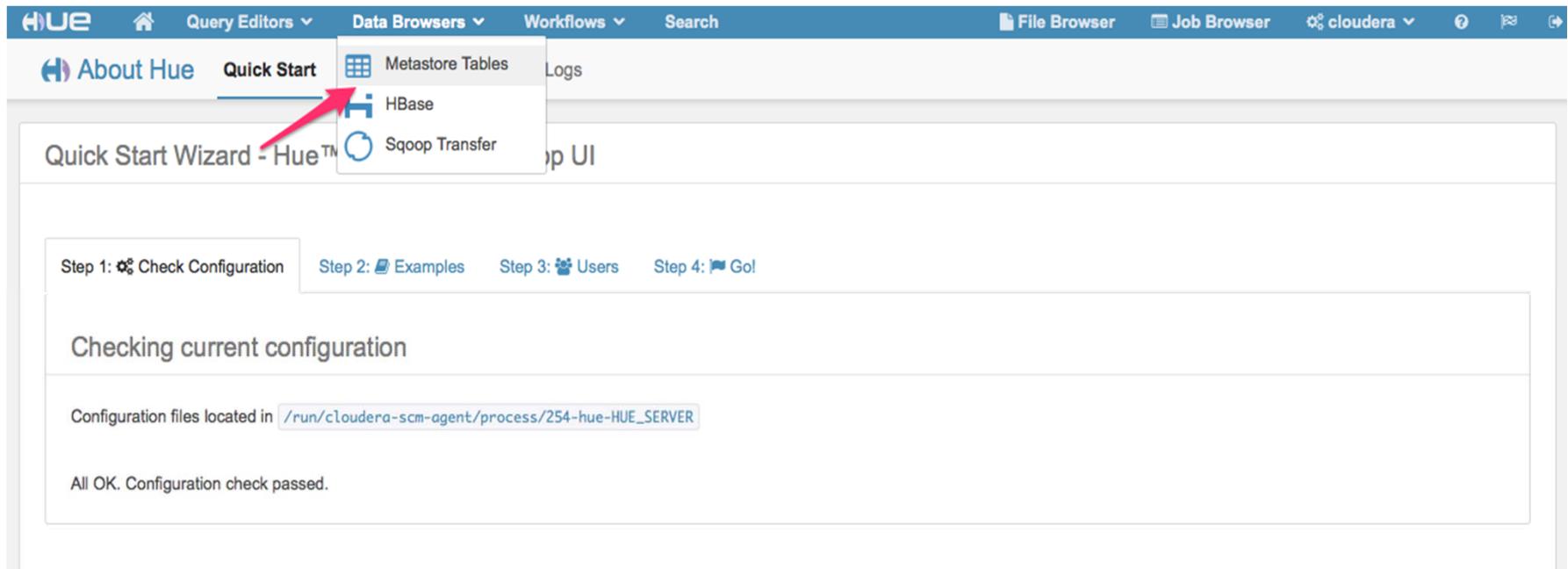
https://s3.amazonaws.com/babs-open-data/babs_open_data_year_1.zip

2. Put file : 201402_trip_data.csv
from folder : 201402_babs_open_data
to HDFS : /user/cloudera

```
[root@quickstart 201402_babs_open_data]# hadoop fs -ls /user/cloudera
Found 4 items
-rw-r--r--  1 root      cloudera  17219022 2016-06-14 08:13 /user/cloudera/201402_t
rip_data.csv
drwxr-xr-x  - cloudera cloudera          0 2016-06-14 04:29 /user/cloudera/input
drwxr-xr-x  - root      cloudera          0 2016-06-14 08:04 /user/cloudera/movielen
s
drwxr-xr-x  - cloudera cloudera          0 2016-06-14 04:32 /user/cloudera/output
[root@quickstart 201402_babs_open_data]#
```

Importing CSV Data with the Metastore App

The BABS data set contains 4 CSVs that contain data for stations, trips, rebalancing (availability), and weather. We will import **trips** dataset using Metastore Tables



The screenshot shows the Hue web interface with the following details:

- Top Navigation Bar:** Includes links for Query Editors, Data Browsers, Workflows, Search, File Browser, Job Browser, cloudera, and various system icons.
- Left Sidebar:** Features a "Quick Start" tab (highlighted by a red arrow) and other options like Metastore Tables, HBase, and Sqoop Transfer.
- Content Area:** Titled "Quick Start Wizard - Hue™". It displays a progress bar with four steps:
 - Step 1: Check Configuration (Completed)
 - Step 2: Examples
 - Step 3: Users
 - Step 4: Go!
- Step 1 Details:** Sub-titled "Checking current configuration". It shows the configuration files are located in `/run/cloudera-scm-agent/process/254-hue-HUE_SERVER`. Below this, a message states "All OK. Configuration check passed."

Select: Create a new table from a file

The screenshot shows the Hue Metastore Manager interface. The top navigation bar includes links for Query Editors, Data Browsers, Workflows, Search, Security, and various system icons. The main title is "Metastore Manager". On the left, there's a sidebar with icons for Databases and Folders, and a "Tables" section listing "sample_07". The main content area shows the "Databases > default" view. It displays "STATS" for the database, including the message "Default Hive database", the "public (ROLE)", and the "Location". Below this is a "TABLES" section with a search bar and buttons for "View", "Browse Data", and "Drop". A table lists the single table "sample_07" with columns for "Table Name", "Comment", and "Type". The "Table Name" column shows a small icon followed by "sample_07". The "Comment" and "Type" columns are empty. In the top right corner of the main content area, there are three icons: a refresh symbol, a document with a plus sign (the "Create" icon), and a plus sign. A red arrow points to the "Create" icon.

Name a table and select a file

Databases > default > 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

trip

Name of the new table. Table names must be globally unique. Table names tend to correspond to the column names.

Description

Bay Area Bike Share

Use a table comment to describe the table. For example, note the data's provenance and a

Input File

/user/cloudera/201408_trip_data.csv

The HDFS path to the file on which to base this new table definition. It can be compressed (.zip) 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.

Choose a file

Home / user / cloudera



..



201402_trip_data.csv



input



movielens



output



Upload a file

Choose Delimiter

Databases > default > 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	col_10	col_11
	Trip ID	Duration	Start Date	Start Station	Start Terminal	End Date	End Station	End Terminal	Bike #	Subscriber Type	Zip Code
	432946	406	8/31/2014	Mountain View Caltrain	28	8/31/2014	Castro Street and El Cami...	32	17	Subscriber	94040
			22:31	St...		22:38					
	432945	468	8/31/2014	Beale at Market	56	8/31/2014	Market at 4th	76	509	Customer	11231
			22:07			22:15					

Define Column Types

Databases > default > Create a new table from a file

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

Define your columns

Use first row as column names

Bulk edit column names

Column name	Column Type	Sample Row #1	Sample Row #2
TripID	int	432946	432945
Duration	int	406	468
StartDate	string	8/31/2014 22:31	8/31/2014 22:07
StartStation	string	Mountain View Caltrain Station	Beale at Market
StartTerminal	tinyint	28	56
EndDate	string	8/31/2014 22:38	8/31/2014 22:15

Create Table : Done

Databases > default > trip

Comment: Bay Area Bike Share

Columns Sample Properties

	Name	Type	Comment
0	tripid	int	
1	duration	int	
2	startdate	string	
3	startstation	string	
4	startterminal	tinyint	
5	enddate	string	
6	endstation	string	
7	endterminal	tinyint	
8	bike	smallint	
9	subscribertype	string	
10	zipcode	smallint	

The screenshot shows the HUE Metastore Manager interface. The top navigation bar includes links for Query Editors, Data Browsers, Workflows, Search, Security, and various system icons. The main title is "Metastore Manager". On the left, a sidebar shows a tree view under "default" database, with "Tables" expanded to show "test_tbl", "trip", and "users". The main content area displays the "Databases > default" page. It includes a "STATS" section with details: "Default Hive database", "public (ROLE)", and "Location". Below this is a "TABLES" section with a search bar and buttons for "View", "Browse Data", and "Drop". A table lists the three tables with columns for checkboxes, Table Name, Comment, and Type.

	Table Name	Comment	Type
<input type="checkbox"/>	test_tbl		
<input type="checkbox"/>	trip		
<input type="checkbox"/>	users		

Starting Hive Editor

The screenshot shows the Hue interface with the 'Hive Editor' tab selected. On the left, there's a sidebar for 'Assist' and 'Settings'. Under 'DATABASE', 'default' is selected. Below it, a table list includes 'airline_data', 'trip' (selected), 'tripid (int)', 'duration (int)', 'startdate (string)', 'startstation (string)', 'startterminal (tinyint)', 'enddate (string)', 'endstation (string)', 'endterminal (tinyint)', 'bike (smallint)', 'subscribertype (string)', and 'zipcode (smallint)'. The main area has a query input field with placeholder text 'Example: SELECT * FROM tablename, or press CTRL + space'. Below the input are buttons for 'Execute', 'Save as...', 'Explain', and 'New query'. A 'Recent queries' section lists three previous queries:

Time	Query	Result
11/04/2015 2:49:28 PM	DROP TABLE `default`.`babs`	See results...
11/04/2015 2:46:08 PM	SELECT startterminal, startstation, COUNT(1) AS count FROM babs GROUP BY startterminal, startstation ORDER BY count DESC LIMIT 10	See results...
11/04/2015 2:45:42 PM	SELECT startterminal, startstation, COUNT(1) AS count FROM bikeshare.trips GROUP BY startterminal, startstation ORDER BY count	

Find the top 10 most popular start stations based on the trip data

```
SELECT startterminal, startstation, COUNT(1) AS count FROM trip  
GROUP BY startterminal, startstation ORDER BY count DESC LIMIT 10
```

The screenshot shows the Hue interface with the 'Query Editor' tab selected. On the left, there's a sidebar with 'Tables' containing 'test_tbl', 'trip', and 'users'. The main area displays the query code and its results.

Query Editor:

```
1 SELECT startterminal, startstation, COUNT(1) AS count FROM trip GROUP BY startterminal, startstation ORDER BY count DESC LIMIT 10
```

Buttons: Execute, Save as..., Explain, Format, or create a New query.

Results:

	startterminal	startstation	count
1	70	San Francisco Caltrain (Townsend at 4th)	9838
2	50	Harry Bridges Plaza (Ferry Building)	7343
3	60	Embarcadero at Sansome	6545
4	77	Market at Sansome	5922
5	55	Temporary Transbay Terminal (Howard at Beale)	5113
6	76	Market at 4th	5030
7	61	2nd at Townsend	4987
8	69	San Francisco Caltrain 2 (330 Townsend)	4976

```
1 SELECT startterminal, startstation, COUNT(1) AS count FROM trip GROUP BY startte
```

Execute

Save as...

Explain

Format

or create a

New query

...



Recent queries

Query

Log

Columns

Results

Chart

