

# APACHE HIVE



# 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
```

# Hive Built in Functions

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'
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', 'oo ar', ) returns 'fb'
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 Commands

## Command Line

Function	Hive
Run query	<code>hive -e 'select a.col from tab1 a'</code>
Run query silent mode	<code>hive -S -e 'select a.col from tab1 a'</code>
Set hive config variables	<code>hive -e 'select a.col from tab1 a' -hiveconf hive.root.logger=DEBUG,console</code>
Use initialization script	<code>hive -i initialize.sql</code>
Run non-interactive script	<code>hive -f script.sql</code>

## Hive Shell

Function	Hive
Run script inside shell	<code>source file_name</code>
Run ls (dfs) commands	<code>dfs -ls /user</code>
Run ls (bash command) from shell	<code>!ls</code>
Set configuration variables	<code>set mapred.reduce.tasks=32</code>
TAB auto completion	<code>set hive.&lt;TAB&gt;</code>
Show all variables starting with hive	<code>set</code>
Revert all variables	<code>reset</code>
Add jar to distributed cache	<code>add jar jar_path</code>
Show all jars in distributed cache	<code>list jars</code>
Delete jar from distributed cache	<code>delete jar jar_name</code>

# 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	<code>USE database;</code>	<code>USE database;</code>
Listing databases	<code>SHOW DATABASES;</code>	<code>SHOW DATABASES;</code>
Listing tables in a database	<code>SHOW TABLES;</code>	<code>SHOW TABLES;</code>
Describing the format of a table	<code>DESCRIBE table;</code>	<code>DESCRIBE (FORMATTED EXTENDED) table;</code>
Creating a database	<code>CREATE DATABASE db_name;</code>	<code>CREATE DATABASE db_name;</code>
Dropping a database	<code>DROP DATABASE db_name;</code>	<code>DROP DATABASE db_name (CASCADE);</code>



# 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>

# Loading Data using Hive

## Start Hive

**\$ hive**

## Quit from Hive

**hive> quit;**

```
[cloudera@quickstart guest1]$ hive
2016-10-13 02:08:13,500 WARN [main] mapreduce.TableMapReduceUtil: The hbase-prefix-tree module jar containing PrefixTreeCodec 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> █
```

# Create Hive Table








**hive> CREATE TABLE TEST\_TBL(ID INT,COUNTRY STRING) ROW FORMAT  
DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE ;**


**hive> SHOW TABLES;**





**hive> describe test\_tbl;**




```
hive> CREATE TABLE TEST_TBL(ID INT,COUNTRY STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE ;
OK
Time taken: 1.2 seconds
hive> SHOW TABLES;
OK
test_tbl
Time taken: 0.234 seconds, Fetched: 1 row(s)
hive> describe test_tbl
> ;
OK
id                int
country           string
Time taken: 0.209 seconds, Fetched: 2 row(s)
```




# Reviewing Hive Table in HDFS

**HUE**  Query Editors ▾ Data Browsers ▾ Workflows ▾ Search Security ▾    ▾   

 File Browser

Search for file name  Actions ▾  Move to trash ▾  Upload ▾  New ▾

 Home / [user](#) / [hive](#) / **warehouse**  History  Trash

<input type="checkbox"/>	Name	Size	User	Group	Permissions	Date
<input type="checkbox"/>	 <a href="#">↑</a>		hive	supergroup	drwxrwxrwx	August 10, 2016 01:09 PM
<input type="checkbox"/>	 <a href="#">.</a>		hive	supergroup	drwxrwxrwx	October 13, 2016 02:43 AM
<input type="checkbox"/>	 <a href="#">test_tbl</a>		cloudera	supergroup	drwxrwxrwx	October 13, 2016 02:43 AM

# Alter and Drop Hive Table

```
Hive > alter table test_tbl add columns (remarks STRING);
```

```
hive > describe test_tbl;
```

```
OK
```

```
id int
```

```
country string
```

```
remarks string
```

```
Time taken: 0.077 seconds
```

```
hive > drop table test_tbl;
```

```
OK
```

```
Time taken: 0.9 seconds
```

# 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](#)

# MovieLens Dataset

1)Open Terminal

2)Type command > mkdir movielens\_dataset

3)Type command >cd movielens\_dataset

4)Type command > wget http://files.grouplens.org/datasets/movielens/ml-100k.zip

5)Type command > unzip ml-100k.zip

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

```
[cloudera@quickstart ~]$ mkdir movielens_dataset
[cloudera@quickstart ~]$ cd movielens_dataset/
[cloudera@quickstart movielens_dataset]$ wget http://files.grouplens.org/datasets/movielens/ml-100k.zip
--2016-10-13 03:07:20-- http://files.grouplens.org/datasets/movielens/ml-100k.zip
Resolving files.grouplens.org... 128.101.34.146
Connecting to files.grouplens.org|128.101.34.146|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4924029 (4.7M) [application/zip]
Saving to: "ml-100k.zip"

100%[=====>] 4,924,029    226K/s   in 36s

2016-10-13 03:08:02 (135 KB/s) - "ml-100k.zip" saved [4924029/4924029]

[cloudera@quickstart movielens_dataset]$ unzip ml-100k.zip
Archive:  ml-100k.zip
  creating: ml-100k/
  inflating: ml-100k/allbut.pl
  inflating: ml-100k/mku.sh
  inflating: ml-100k/README
  inflating: ml-100k/u.data
```

```
[cloudera@quickstart movielens_dataset]$ 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
12|28|F|other|06405
13|47|M|educator|29206
14|45|M|scientist|55106
15|49|F|educator|97301
16|21|M|entertainment|10309
17|30|M|programmer|06355
```

# Moving dataset to HDFS

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

```
[cloudera@quickstart movielens_dataset]$ cd ml-100k
[cloudera@quickstart ml-100k]$ hadoop fs -mkdir /user/cloudera/movielens
[cloudera@quickstart ml-100k]$ hadoop fs -put u.user /user/cloudera/movielens
[cloudera@quickstart ml-100k]$ hadoop fs -ls /user/cloudera/movielens
Found 1 items
-rw-r--r--    1 cloudera cloudera      22628 2016-10-13 03:16 /user/cloudera/movielens/u.user
[cloudera@quickstart ml-100k]$
```



# CREATE & SELECT TABLE

Type command > hive

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' ;  
hive > select \* from users;

```
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

# CREATE EXTERNAL TABLE Not File Moved

**HUE** Home Query Editors ▾ Data Browsers ▾ Workflows ▾ Search Security ▾

File Browser

Search for file name Actions ▾ Move to trash ▾ Upload ▾ New ▾

Home / user / hive / warehouse History Trash

<input type="checkbox"/>	Name	Size	User	Group	Permissions	Date
<input type="checkbox"/>	↑		hive	supergroup	drwxrwxrwx	August 10, 2016 01:09 PM
<input type="checkbox"/>	.		hive	supergroup	drwxrwxrwx	October 13, 2016 02:43 AM
<input type="checkbox"/>	test_tbl		cloudera	supergroup	drwxrwxrwx	October 13, 2016 02:43 AM

**HUE** Home Query Editors ▾ Data Browsers ▾ Workflows ▾ Search Security ▾

Metastore Manager

default

Databases > default

STATS

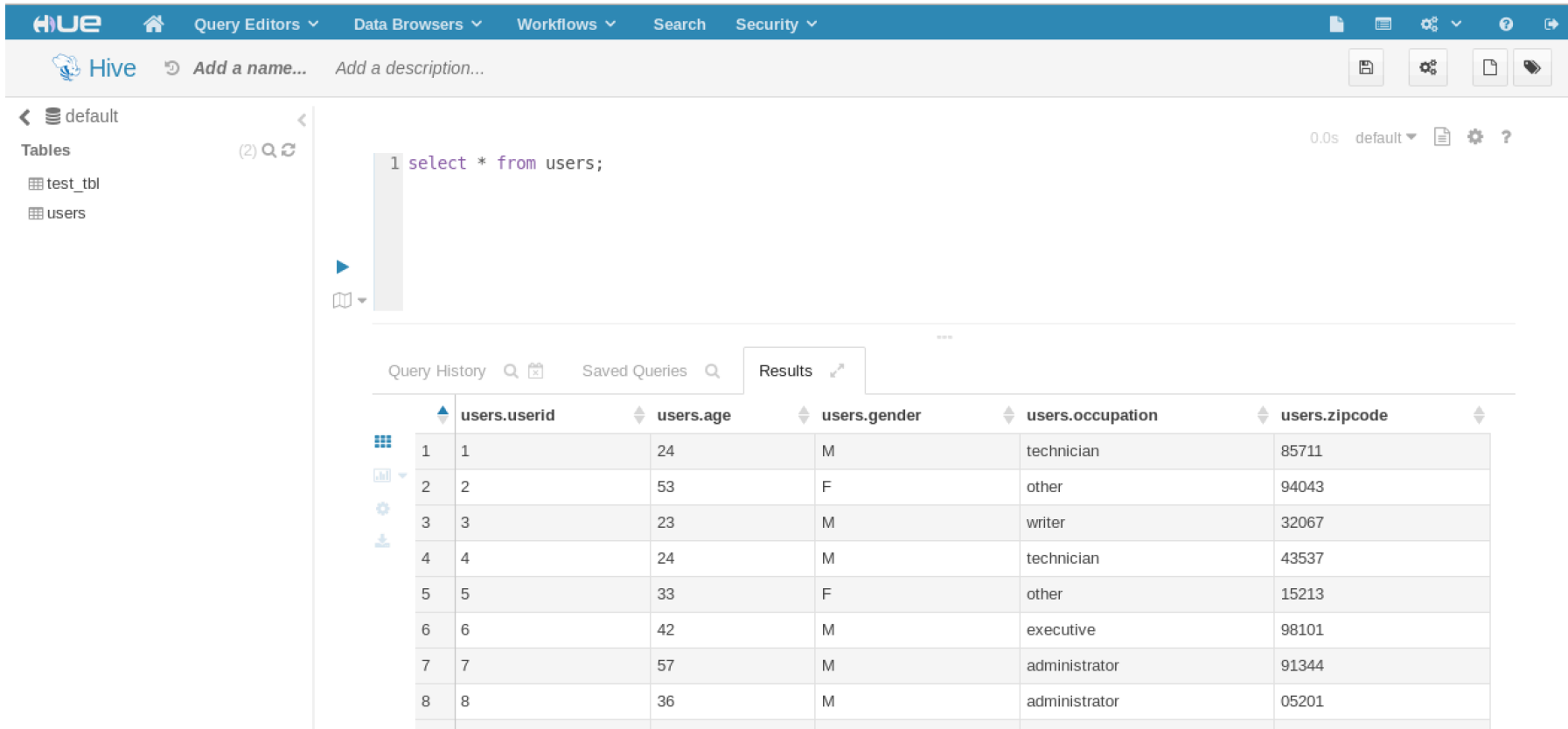
Default Hive database public (ROLE) Location

TABLES

Search for a table... View Browse Data Drop

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

# Starting Hive Editor on HUE



The screenshot displays the HUE Hive Editor interface. The top navigation bar includes links for Home, Query Editors, Data Browsers, Workflows, Search, and Security. Below this, the Hive logo and options to 'Add a name...' and 'Add a description...' are visible. On the left, a sidebar shows the 'default' database and a list of tables: 'test\_tbl' and 'users'. The main query editor contains the SQL statement: `1 select * from users;`. Below the editor, the 'Results' tab is active, showing a table with 8 rows of data. The table columns are 'users.userid', 'users.age', 'users.gender', 'users.occupation', and 'users.zipcode'.

	users.userid	users.age	users.gender	users.occupation	users.zipcode
1	1	24	M	technician	85711
2	2	53	F	other	94043
3	3	23	M	writer	32067
4	4	24	M	technician	43537
5	5	33	F	other	15213
6	6	42	M	executive	98101
7	7	57	M	administrator	91344
8	8	36	M	administrator	05201