Databases in Hive

By default is default.

>create database [if not exists] databaseName;

>show databases;

>show database like 'a.\*'

Whenever database is created then hive creates the directory in the dataware house location with an extentsion of .db.

We can override the location by using the following syntax:

>create database databaseName location '/our own path' comment 'some comments' (used for describing something about database)

>describe database databaseName;

>create database databaseName location '/our own path' comment 'some comments' with dbproperties ('' = '', ''='')

>describe database extended databaseName;

to use database:

>use databaseName - TO change the database.

>show databases - TO list thee databases

>drop database [if exixts ] databaseName

>drop database [if exixts ] databaseName By default, Hive won’t permit you to drop a database if it contains tables.

You can either drop the tables first or append the CASCADE keyword to the command

>drop database [if exixts ] databaseName cascade

To alter database

>alter database databaseName set dbproperties ('' = '');

table creation:

We have two types of Tables:

1. Managed Tables (Internal Tables)

2. External Tables

Managed table>create table if not exists tableName

External table>create external table if not exists tableName

command for complete sysntax:

CREATE TABLE [external][if not exists] employees ( name STRING, salary FLOAT, subordinates ARRAY<STRING>, deductions MAP<STRING, FLOAT>, address

STRUCT<street:STRING, city:STRING, state:STRING, zip:INT>)ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' COLLECTION ITEMS TERMINATED BY ',' MAP KEYS TERMINATED BY ':'

LINES TERMINATED BY '\n' STORED AS TEXTFILE COMMENT 'Description of the table' TBLPROPERTIES ('creator'='name', 'created\_at'='date', ...) LOCATION

'/user/hive/warehouse/dbname.db/employees';

Copying the Schema:

CREATE TABLE [IF NOT EXISTS] employee LIKE employees;

Display the list of tables in the database:

>show tables;

>show tables 'ma.\*";

Partitions:

To create partition while creating the table:

CREATE TABLE employees (name STRING, salary FLOAT, subordinates ARRAY<STRING>, deductions MAP<STRING, FLOAT>, address STRUCT<street:STRING, city:STRING, state:STRING,

zip:INT>) PARTITIONED BY (country STRING, state STRING);

To show the partitions:

> SHOW PARTITIONS employees

> SHOW PARTITIONS employees PARTITION(country='countryName');

Partitions with external tables:

CREATE EXTERNAL TABLE xyz ( hms INT, severity STRING, server STRING, process\_id INT, message STRING) PARTITIONED BY (year INT, month INT, day INT) ROW FORMAT

DELIMITED FIELDS TERMINATED BY '\t';

Alter table to add partitions:

ALTER TABLE tableName ADD PARTITION(year = 2012, month = 1, day = 2)

Alter table to drop partitions:

ALTER TABLE log\_messages DROP IF EXISTS PARTITION(year = 2011, month = 12, day = 2);

Buckets:

CREATE EXTERNAL TABLE IF NOT EXISTS stocks ( exchange STRING, symbol STRING, ymd STRING, price\_open FLOAT, price\_high FLOAT, price\_low FLOAT, price\_close FLOAT,

volume INT, price\_adj\_close FLOAT) CLUSTERED BY (exchange, symbol) SORTED BY (ymd ASC) INTO 96 BUCKETS ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'

Dropping Tables:

>DROP TABLE IF EXISTS employees;

Altering Tables:

Renaming Tables:> ALTER TABLE employees RENAME TO employee;

Altering the columns:>ALTER TABLE log\_messages CHANGE COLUMN hms hours\_minutes\_seconds INT

Adding Columns:>ALTER TABLE log\_messages ADD COLUMNS ( app\_name STRING COMMENT 'Application name', session\_id LONG COMMENT 'The current session id');

Deleting/Replacing columns:> ALTER TABLE log\_messages REPLACE COLUMNS ( hours\_mins\_secs INT COMMENT 'hour, minute, seconds from timestamp', severity STRING COMMENT

'The message

severity' message STRING COMMENT 'The rest of the message');

Alter table properties:> ALTER TABLE log\_messages SET TBLPROPERTIES ( 'notes' = 'The process id is no longer captured; this column is always NULL');

=======================================================================================================================================================================

================

Hive DML:

\*\*\*\*\*\*\*\*\*\*\*\*

Loading data into Hive Tables;

------------------------------

We will load this data from local file system / Distributed file system into Hive Table:

>LOAD DATA [LOCAL] INPATH 'employees' OVERWRITE INTO TABLE employees PARTITION (country = 'US', state = 'CA');

Inserting data into tables from queries:

>INSERT OVERWRITE TABLE employees PARTITION (country = 'US', state = 'OR') SELECT \* FROM staged\_employees se WHERE se.cnty = 'US' AND se.st = 'OR';

Creating Tables and Loading Them in One Query:

> CREATE TABLE ca\_employees AS SELECT name, salary, address FROM employees WHERE se.state = 'CA';

Exporting Data:

>INSERT OVERWRITE LOCAL DIRECTORY '/tmp/ca\_employees' SELECT name, salary, address FROM employees WHERE se.state = 'CA';

=======================================================================================================================================================================

=================

Hive DQL/DRL

------------

Simple select statement:

> SELECT name, salary FROM employees;

Specify columns with regular expressions:

>SELECT symbol, `price.\*` FROM stocks;

It supports lots of functions:

1. mathematical functions - round

2. Aggregate functions - avg

3. Table generating functions - explode

4. Other Built-in functions - length

LIMIT CLAUSE:

This is used to limit the results

> SELECT name, salary FROM employees limit 10;

LIKE and RLIKE:

> SELECT name, address.street FROM employees WHERE address.street LIKE '%muru.';

>SELECT name, address.street FROM employees WHERE address.state RLIKE '.\*(karnataka|andhra).\*';

Group by clauses:

> SELECT year(ymd), avg(price\_close) FROM stocks GROUP BY year(ymd);

Having clause:

> SELECT year(ymd), avg(price\_close) FROM stocks GROUP BY year(ymd) HAVING avg(price\_close) > 50.0

Joins

------

1. inner join

2. outer join

Inner Join:

>SELECT a.ymd, a.price\_close, b.price\_close FROM stocks a JOIN stocks b ON a.ymd = b.ymd WHERE a.symbol = 'AAPL' AND b.symbol = 'IBM';

Outer Join:

Left outer join:

>SELECT s.ymd, s.symbol, s.price\_close, d.dividend FROM stocks s LEFT OUTER JOIN dividends d ON s.ymd = d.ymd AND s.symbol = d.symbol WHERE s.symbol = 'AAPL';

Cartesian Product JOINs:

>SELECTS \* FROM stocks JOIN dividends;

ORDER BY and SORT BY:

order by --> Total out is going to sorted (single reducer)

sort by --> The output of every reducer is going to sorted.

DISTRIBUTE BY: it controls how map output is divided among reducers.

> SELECT s.ymd, s.symbol, s.price\_close FROM stocks s DISTRIBUTE BY s.symbol SORT BY s.symbol ASC, s.ymd ASC;