**Hive Data Definitions**

Data definitions commands are the commands which are related to structure of tables, databases, views etc. Below are some data definitions commands in hive:

1. **CREATE**
2. **ALTER**
3. **DROP**
4. **TRUNCATE**

**CREATE**: Create command is used to create a new database object.

**create table employee**

**(empid int,**

**empname string,**

**salary float,**

**deptno int)**

**row format delimited**

**fields terminated by ',';**

**ALTER :** Alter table statements enable you to change the structure of an existing table. You can add columns/partitions, change SerDe, add table and SerDe properties, or rename the table itself. Similarly, alter table partition statements allow you change the properties of a specific partition in the named table.

alter table employee rename to emp\_details;

alter table emp\_details add column(address string);

**DROP:** DROP TABLE removes metadata and data for this table. The data is actually moved to the .Trash/Current directory if Trash is configured (and PURGE is not specified). The metadata is completely lost.

When dropping an EXTERNAL table, data in the table will *NOT* be deleted from the file system.

When dropping a table referenced by views, no warning is given (the views are left dangling as invalid and must be dropped or recreated by the user).

Otherwise, the table information is removed from the metastore and the raw data is removed as if by 'hadoop dfs -rm'. In many cases, this results in the table data being moved into the user's .Trash folder in their home directory; users who mistakenly DROP TABLEs may thus be able to recover their lost data by recreating a table with the same schema, recreating any necessary partitions, and then moving the data back into place manually using Hadoop. This solution is subject to change over time or across installations as it relies on the underlying implementation; users are strongly encouraged not to drop tables capriciously.

**DROP TABLE [IF EXISTS] emp\_details**;

**TRUNCATE:** Removes all rows from a table or partition(s). The rows will be trashed if the filesystem Trash is enabled, otherwise they are deleted (as of Hive 2.2.0). Currently the target table should be native/managed table or an exception will be thrown. User can specify partial partition\_spec for truncating multiple partitions at once and omitting partition\_spec will truncate all partitions in the table.

**TRUNCATE TABLE emp\_details [PARTITION partition\_spec];**

**Hive Data Manupulations**

Manipulating data is the process of exchanging, moving, sorting, and transforming the data. This technique is used in many situations, such as cleaning data, searching patterns, creating trends, and so on. Hive offers various query statements, keywords, operators, and functions to carry out data manipulation. Below are some Data Manipulation commands in Hive:

1. **INSERT**
2. **UPDATE (available from hive 0.14)**
3. **DELETE (available from hive 0.14)**

**INSERT:**

Query Results can be inserted into tables by using the insert clause.

**insert overwrite table emp\_target**

**select \* from emp\_details**

data can be inserted into partitions as well using below command :

**INSERT INTO TABLE tablename1 [PARTITION (partcol1=val1, partcol2=val2 ...)] select statement1 FROM tablename;**

**UPDATE (available from hive 0.14)**

Update is available starting in Hive 0.14

Updates can only be performed on tables that support ACID.

Syntax:

**UPDATE tablename SET column = value [, column = value ...] [WHERE expression]**

**Note:**

* The referenced column must be a column of the table being updated.
* The value assigned must be an expression that Hive supports in the select clause.  Thus arithmetic operators, UDFs, casts, literals, etc. are supported.  Subqueries are not supported.
* Only rows that match the WHERE clause will be updated.
* Partitioning columns cannot be updated.
* Bucketing columns cannot be updated.
* In Hive 0.14, upon successful completion of this operation the changes will be auto-committed.

**DELETE (available from hive 0.14)**

Delete is available starting in Hive 0.14

Deletes can only be performed on tables that support ACID.

Syntax:

**DELETE FROM tablename [WHERE expression]**

**Note:**

* Only rows that match the WHERE clause will be deleted.
* In Hive 0.14, upon successful completion of this operation the changes will be auto-committed.

**HiveQL Manupulations**

**LOAD DATA LOCAL INPATH   
'path'  
OVERWRITE INTO TABLE employees  
PARTITION (country = '  
US  
', state = '  
CA  
');**  
This command will first create the directory for the partition, if it doesn’t already exist, then copy the data to it.  
If the target table is not partitioned, you omit the PARTITION clause.  
Here is an example for the state of Oregon,  
where we presume the data is already in another table called staged\_employees.

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

With OVERWRITE, any previous contents of the  
partition (or whole table if not partitioned)  
are replaced.  
  
If you drop the keyword OVERWRITE or replace  
it with INTO, Hive appends the data rather  
than replaces it. This feature is only available in   
Hive v0.8.0 or later.  
i.Dynamic Partition Inserts.  
There’s still one problem with this syntax: if you have a lot of partitions to create, you have to write a lot of SQL! fortunately, Hive also supports a   
dynamic partition   
feature,where it can infer the partitions to create based on query parameters. By comparison,up until now we have considered only static partitions.  
Consider this change to the previous example:  
  
**INSERT OVERWRITE TABLE employees  
PARTITION (country, state)  
SELECT ..., se.cnty, se.st  
FROM staged\_employees se;**

Creating Tables and Loading   
Them in One Query  
You can also create a table and insert query results into it in one statement:  
  
**CREATE TABLE ca\_employees  
AS SELECT name, salary, address  
FROM employees  
WHERE se.state = 'CA';**

This table contains just the name, salary, and  
address columns from the employee table  
records for employees in California. The schema  
for the new table is taken from the  
SELECT clause.