DDL stands for Data Definition Language. Now Hive being a Data warehouse tool provides a broad set of DDL. Sine it is very similar to MySQL, the DDL facilities are also very similar as well.

Basic DDL statements that are used for MySQL are-

1. CREATE
2. ALTER
3. DROP etc.

The DDLs can be used for DATABASES/SCHEMA, TABLES, INDEX, VIEW, FUNCTION, PARTITIONS etc.

Below are the DDL statements available for **DATABASES** in Hive-

1. **CREATE-**This is used to create DATABASE or SCHEMA.

Below is the syntax-

CREATE (DATABASE|SCHEMA) [IF NOT EXISTS] database\_name

  [COMMENT database\_comment]

  [LOCATION hdfs\_path]

  [WITH DBPROPERTIES (property\_name=property\_value, ...)];

The above statement is used to define database/schema in HIVE. The **IF NOT EXISTS** statement makes sure to create the DATABASE if it does not exist. The DATABASE name can also be replaced by SCHEMA name.

We can also add any comment to the database if required and give Location also where we want to create database.

1. **DROP-** This is used to drop a database/schema.

Below is the syntax-

DROP (DATABASE|SCHEMA) [IF EXISTS] database\_name [RESTRICT|CASCADE];

In above statement IF EXIST is optional and will suppress warnings if the database doesn’t exist. By default, Hive doesn’t permit to drop a database if it contains tables. We can either drop the tables first or append the CASCADE keyword to the command, which will cause the Hive to drop the tables in the database first. Using the RESTRICT keyword instead of CASCADE is equivalent to the default behavior, where existing tables must be dropped before dropping the database. When a database is dropped, its directory is also deleted.

1. **ALTER-** Below are the ALTER facilities available to ALTER the database-

ALTER (DATABASE|SCHEMA) database\_name SET DBPROPERTIES (property\_name=property\_value, ...)

ALTER (DATABASE|SCHEMA) database\_name SET OWNER [USER|ROLE] user\_or\_role;

ALTER (DATABASE|SCHEMA) database\_name SET LOCATION hdfs\_path;

We can set key-value pairs in the DBPROPERTIES associated with a database using the ALTER DATABASE command. No other metadata about the database can be changed, including its name and directory location.

1. **SHOW -**At any time, we can see the databases that already exist as follows-

SHOW (DATABASES|SCHEMAS) [LIKE 'identifier\_with\_wildcards'];

We can also give Wildcard characters to enhance our search.

1. **DESCRIBE-** DESCRIBE DATABASE shows the name of the database, its comment (if one has been set), and its root location on the file system. The uses of SCHEMA and DATABASE are interchangeable – they mean the same thing.

DESCRIBE DATABASE [EXTENDED] db\_name;

Below are the DDL statements available for HIVE tables-

1. **CREATE-** As the name suggests CREATE statement is used to CREATE a table. Below is the syntax-

CREATE TABLE IF NOT EXISTS mydb.employees (

Name STRING COMMENT 'Employee name',

salary FLOAT COMMENT 'Employee salary',

subordinates ARRAY<STRING> COMMENT 'Names of subordinates',

deductions MAP<STRING, FLOAT>

COMMENT 'Keys are deductions names, values are percentages',

Address STRUCT<street:STRING, city:STRING, state:STRING, zip:INT>

COMMENT 'Home address')

COMMENT 'Description of the table'

TBLPROPERTIES ('creator'='me', 'created\_at'='2012-01-02 10:00:00', ...)

LOCATION '/user/hive/warehouse/mydb.db/employees';

The CREATE TABLE statement follows SQL conventions, but Hive’s version offers significant extensions to support a wide range of flexibility where the data files for tables are stored, the formats used, etc. If we add the option IF NOT EXISTS, Hive will silently ignore the statement if the table already exists. We can add a comment to any column, after the type. Like databases, you can attach a comment to the table itself and you can define one or more table *properties*. In most cases, the primary benefit of TBLPROPERTIES is to add additional documentation in a key-value format. Hive automatically adds two table properties: last\_modified\_by holds the username of the last user to modify the table, and last\_modified\_time holds the epoch time in seconds of that modification. By default, Hive always creates the table’s directory under the directory for the enclosing database. The exception is the *default* database.

1. **TRUNCATE-** It Removes all rows from a table or partition(s). The rows will be trashed if the file system Trash is enabled, otherwise they are deleted. Currently the target table should be native/managed table or an exception will be thrown.  Below is the syntax-

TRUNCATE TABLE table\_name [PARTITION partition\_spec];

1. **ALTER-** Alter table modifies table’s metadata only. The data for the table is untouched. Its upto us to ensure that any modifications are consistent with the actual data.

Below are some of the ALTER table options available-

1. Rename- ALTER TABLE table\_name RENAME TO new\_table\_name;
2. Adding metadata to tables🡪 ALTER TABLE table\_name SET TBLPROPERTIES table\_properties;
3. Giving Comment🡪 ALTER TABLE table\_name SET TBLPROPERTIES ('comment' = new\_comment);
4. Changing table’s physical storage properties🡪

**ALTER TABLE table\_name CLUSTERED BY (col\_name, col\_name, ...) [SORTED BY (col\_name, ...)]**

**INTO num\_buckets BUCKETS;**

1. **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.

Syntax- DROP TABLE [IF EXISTS] table\_name [PURGE];

**GRANT**-If a user is granted a privilege WITH GRANT OPTION on a table or view, then the user can also grant/revoke privileges of other users and roles on those objects.

Syntax- GRANT

    priv\_type [, priv\_type ] ...

    ON table\_or\_view\_name

    TO principal\_specification [, principal\_specification] ...

    [WITH GRANT OPTION];

**REVOKE-**  The grant option for a privilege can be removed while still keeping the privilege by using REVOKE GRANT OPTION FOR <privilege>

**REVOKE [GRANT OPTION FOR]**

**priv\_type [, priv\_type ] ...**

**ON table\_or\_view\_name**

**FROM principal\_specification [, principal\_specification] ... ;**

**principal\_specification**

**: USER user**

**| ROLE role**

**priv\_type**

**: INSERT | SELECT | UPDATE | DELETE | ALL**