ASSIGNMENT 7.3

**Hive Data Definition**s

**Hive Data Definition** Language. **Hive Data Definition** Language (DDL) is a subset of **Hive** SQL statements that describe the **data** structure in **Hive** by creating, deleting, or altering schema objects such as databases, tables, views, partitions, and buckets.

Following are few operations

1. To **CREATE** table

CREATE TABLE Batch(

ID INT,

Name STRING,

Score INT,

Skill STRING

)

1. To **ALTER** table

ALTER TABLE ID RENAME TO Sno;

1. To DROP Table:

DROP TABLE IF EXISTS Batch;

**Hive Data Manipulation**

Data manipulation language parts that are used to put data into tables and to extract data from tables to the filesystem.

In Data Manipulation, we can perform following functions:

1. **Loading files into Table**: HIVE does not do any transformation while loading data into tables. Load operations are currently pure copy/move operations that move datafiles into locations corresponding to HIVE tables.

**LOAD DATA INPATH ‘filepath’ OVERWRITE INTO TABLE <table name> [PARTITION (partcol1-val1, partcol2-val2,…)]**

1. **Inserting data into HIVE table**: Query Results can be inserted into tables by using the insert clause.

**INSERT OVERWRITE TABLE <tablename1> [PARTITION(partcol1-val1, partcol2-val2,…) [IF NOT EXISTS]] select\_statement1 FROM <from\_statement>;**

**INSERT INTO TABLE <tablename1> [PARTITION(partcol1-val1, partcol2-val2,…)] <select\_statement1> FROM <select\_statement>;**

INSERT OVERWRITE will overwrite any existing data in the table or partition.

INSERT INTO will append to the table or partition, keeping the existing data intact.

1. **Writing data into the filesystem from queries:** Queries results can be inserted into file system directories.

**INSERT OVERWRITE <LOCAL> DIRECTORY directory1 <select\_statement1**

1. **Inserting values into tables from SQL:**

**INSERT INTO TABLE <table name> [PARTITION (partcol1[=val1], partcol2[=val2]…)] VALUES values\_row[ values\_row..]**

1. **Update**

**UPDATE <tablename> SET column = value[, colum=value…] <WHERE clause>**

1. **Delete**

**DELETE FROM <table name> <WHERE clause>**

1. **Merge**

**MERGE INTO <target table> AS T USING <source expression/table> AS S ON <Boolean expression1>**

1. **Modify** : There are multiple ways to modify data in HIVE:
   * + - * LOAD
         * INSERT

Into HIVE tables

Into directories

* + - * + UPDATE
        + DELETE
        + MERGE

**HiveQL: Data Manipulation**

1.Loading Data into Managed Tables.  
2.Inserting Data into Tables from Queries.  
i.Dynamic Partition Inserts.  
3.Creating Tables and Loading Them in One Query.  
4.Exporting Data.

**Loading Data into Managed Tables :-**

Since Hive has no row-level insert, update, and  
delete operations, the only way to put  
data into an table is to use one of the “bulk” load operations. Or you can just write files  
in the correct directories by other means

Loading Data into Managed Tables.  
The INSERT statement lets you load data into a table from a query.  
Inserting Data into Tables from Queries  
How do we get data out of tables? If the data files are already formatted the way you want, then it’s simple enough to copy the directories or files:  
  
hadoop fs -cp source\_path target\_path  
  
Otherwise, you can use INSERT … DIRECTORY …,  
as in this example:  
INSERT OVERWRITE LOCAL DIRECTORY '/tmp/ca\_employees'  
SELECT name, salary, address  
FROM employees  
WHERE se.state = 'CA';

**Inserting Data into Tables from Queries :-**

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.

**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.  
  
A common use for this feature is to extract a convenient subset of data from a larger,  
more unwieldy table.