**Hive 7.3 Assignment:**

**Problem Statement:**

**Explain the below concepts with an example in brief.**

**Hive Data Definitions**

*Data definition language* parts of HiveQL, which are used for creating, altering, and dropping databases, tables, views, functions, and indexes.

hive> CREATE DATABASE financials;

Hive will throw an error if financials already exists. You can suppress these warnings with this variation:

hive> CREATE DATABASE IF NOT EXISTS financials;

hive> SHOW DATABASES;

DESCRIBE DATABASE financials;

DROP DATABASE IF EXISTS financials;

USE Database;

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)

ROW FORMAT DELIMITED FIELDS TERMINATED BY ','

LOCATION '/data/stocks';

CREATE EXTERNAL TABLE IF NOT EXISTS mydb.employees3

LIKE mydb.employees LOCATION '/path/to/data';

PARTITIONED :

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);

SHOW PARTITIONS employees;

DROP TABLE IF EXISTS employees;

ALTER TABLE log\_messages RENAME TO logmsgs;

**Hive Data Manipulations**

Loading Files into tables

Ex:

INSERT OVERWRITE TABLE page\_view PARTITION(dt='2008-06-08', country)

SELECT pvs.viewTime, pvs.userid, pvs.page\_url, pvs.referrer\_url, null, null, pvs.ip, pvs.cnt from view

Inserting data into Hive tables from Queries/SQL

Ex:

INSERT INTO merge\_data.merge\_source VALUES

Update

INSERT OVERWRITE TABLE page\_view PARTITION(dt='2008-06-08', country)

SELECT pvs.viewTime, pvs.userid, pvs.page\_url, pvs.referrer\_url, null, null, pvs.ip, pvs.cnt from view

Merge

Step 1:

CREATE DATABASE merge\_data;

CREATE TABLE merge\_data.transactions(

ID int,

TranValue string,

last\_update\_user string)

PARTITIONED BY (tran\_date string)

CLUSTERED BY (ID) into 5 buckets

STORED AS ORC TBLPROPERTIES ('transactional'='true');

CREATE TABLE merge\_data.merge\_source(

ID int,

TranValue string,

tran\_date string)

STORED AS ORC;

Step 2:

INSERT INTO merge\_data.transactions PARTITION (tran\_date) VALUES

(1, 'value\_01', 'creation', '20170410'),

(2, 'value\_02', 'creation', '20170410'),

(3, 'value\_03', 'creation', '20170410'),

(4, 'value\_04', 'creation', '20170410'),

(5, 'value\_05', 'creation', '20170413'),

(6, 'value\_06', 'creation', '20170413'),

(7, 'value\_07', 'creation', '20170413'),

(8, 'value\_08', 'creation', '20170413'),

(9, 'value\_09', 'creation', '20170413'),

(10, 'value\_10','creation', '20170413');

INSERT INTO merge\_data.merge\_source VALUES

(1, 'value\_01', '20170410'),

(4, NULL, '20170410'),

(7, 'value\_77777', '20170413'),

(8, NULL, '20170413'),

(8, 'value\_08', '20170415'),

(11, 'value\_11', '20170415');

Step 3:

MERGE INTO merge\_data.transactions AS T

USING merge\_data.merge\_source AS S

ON T.ID = S.ID and T.tran\_date = S.tran\_date

WHEN MATCHED AND (T.TranValue != S.TranValue AND S.TranValue IS NOT NULL) THEN UPDATE SET TranValue = S.TranValue, last\_update\_user = 'merge\_update'

WHEN MATCHED AND S.TranValue IS NULL THEN DELETE

WHEN NOT MATCHED THEN INSERT VALUES (S.ID, S.TranValue, 'merge\_insert', S.tran\_date);

**HiveQL Manipulations:**

HiveQL is the Hive query language. Like all SQL dialects in widespread use, it doesn’t fully conform to any particular revision of the ANSI SQL standard. It is perhaps closest to MySQL’s dialect, but with significant differences. Hive offers no support for row-level inserts, updates, and deletes. Hive doesn’t support transactions

the Hive query language, focusing on the data manipulation language parts that are used to put data into tables and to extract data from tables to the filesystem.

Ex:

LOAD DATA LOCAL INPATH '${env:HOME}/california-employees' OVERWRITE INTO TABLE employees

PARTITION (country = 'US', state = 'CA');