**Assignment 1**

1. create a database named assign1

create database assign1;

1. What is the database path on HDFS?

DESCRIBE Database assign1;

hdfs://namenode:8020/user/hive/warehouse/assign1.db

-3 create a database name assign1\_loc and set its location to /hp\_db/[db\_name]

create database assign1\_loc

location '/hp\_dp/assign1\_loc' ;

- 4create a hive managed table assign1\_intern\_tab inside the assign1 database with the right data types to host the data file employees

use assign1;

hive> CREATE TABLE IF NOT EXISTS assign1.assign1\_intern\_tab(

> ID TINYINT ,

> NAME VARCHAR(25),

> AGE TINYINT,

> COUNTRY VARCHAR(25))

> ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' ;

-5 What is the table path in HDFS?

hive> describe formatted assign1\_intern\_tab;

Location: hdfs://namenode:8020/user/hive/warehouse/assign1.db/assign1\_intern\_tab

6- load the data from the local file system into the table using two different commands

hive> LOAD DATA LOCAL INPATH '/employee/employee\_details.txt' INTO TABLE assign1\_intern\_tab;

hive> !hdfs dfs -mkdir /course\_demo;

hive> !hdfs dfs -copyFromLocal -f /employee/employee\_details.txt /course\_demo ;

hive> LOAD DATA INPATH '/course\_demo/employee\_details.txt' INTO TABLE assign1\_intern\_tab;

7- select 10 records from the table as a sample to ensure the data was correctly loaded

hive> select \* from assign1\_intern\_tab

> limit 10 ;

8- Create external table assign1\_intern\_tab inside the assign1\_loc database

hive> create external table assign1\_loc.assign1\_intern\_tab(

> ID TINYINT,

> NAME VARCHAR(25),

> AGE TINYINT,

> COUNTRY VARCHAR(25))

> ROW FORMAT DELIMITED

> FIELDS TERMINATED BY ','

> STORED AS TEXTFILE

> LOCATION '/course\_demo' ;

9- What is the table path in HDFS?

hive> describe formatted assign1\_loc.assign1\_intern\_tab;

Location: hdfs://namenode:8020/course\_demo

10- move the data from local filesystem to the directory created in step 7

Already done in creation

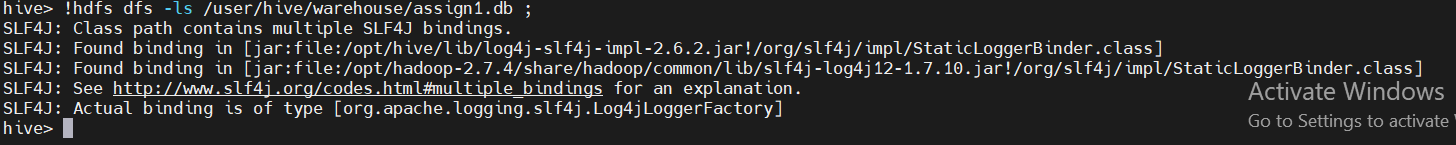
11- drop both tables, is the data present after deletion or not?

hive> drop table assign1\_loc.assign1\_intern\_tab; -- still data existing in directory

hive> drop table assign1.assign1\_intern\_tab; -- no data existing in directory

A computer screen capture

Description automatically generated with medium confidence



12- recreate both tables

hive> CREATE TABLE IF NOT EXISTS assign1.assign1\_intern\_tab(

> ID TINYINT ,

> NAME VARCHAR(25),

> AGE TINYINT,

> COUNTRY VARCHAR(25))

> ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' ;

OK

Time taken: 0.546 seconds

hive> create external table assign1\_loc.assign1\_intern\_tab(

> ID TINYINT,

> NAME VARCHAR(25),

> AGE TINYINT,

> COUNTRY VARCHAR(25))

> ROW FORMAT DELIMITED

> FIELDS TERMINATED BY ','

> STORED AS TEXTFILE

> LOCATION '/course\_demo' ;

13- list both table directories

Location: hdfs://namenode:8020/user/hive/warehouse/assign1.db/assign1\_intern\_tab

Location: hdfs://namenode:8020/course\_demo

14- create internal table 'staging' inside the assign1 database

hive> CREATE TABLE IF NOT EXISTS assign1.staging(

> ID TINYINT ,

> NAME VARCHAR(25),

> AGE TINYINT,

> COUNTRY VARCHAR(25)

> )

> ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' ;

15- load the staging table with the data from file employees

hive> LOAD DATA INPATH '/course\_demo/employee\_details.txt' INTO TABLE assign1.staging ;

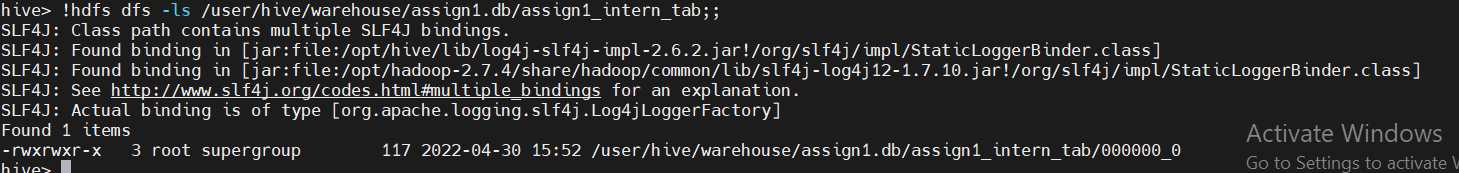
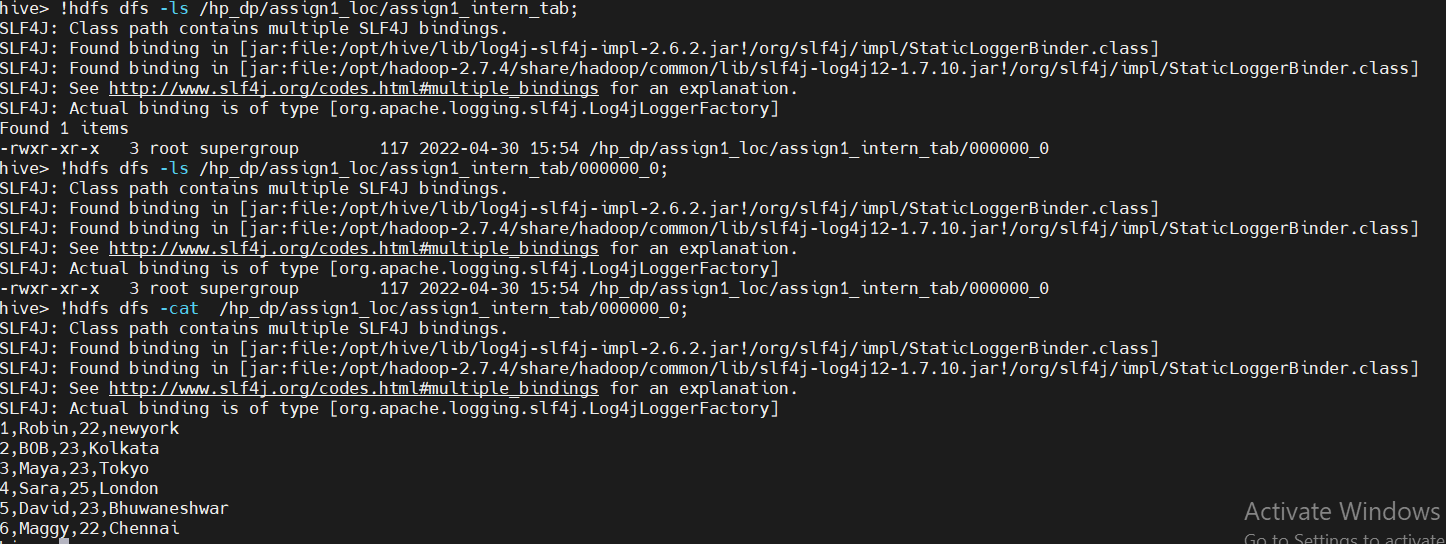
16- load tables assign1\_intern\_tab and assign1\_extern\_tab from the staging table using INSERT SELECT statement

hive> INSERT INTO assign1.assign1\_intern\_tab select \* from assign1.staging ;

hive> INSERT INTO assign1\_loc.assign1\_intern\_tab select \* from assign1.staging ;

17- List both directory tables and check if there is data or not

There are data in two directories



18- count the lines inside the file songs

hive> ! wc -l /employee/songs.csv ;

80

19- create a table with the right types to host the data in file.

hive> create table assign1.songs(

> artist\_id STRING,

> artist\_latitude DOUBLE ,

> artist\_location STRING,

> artist\_longtitude DOUBLE,

> artist\_name STRING,

> duration DOUBLE,

> num\_songs INT ,

> song\_id STRING,

> title STRING,

> year STRING)

> ROW FORMAT DELIMITED

> FIELDS TERMINATED BY ',' ;

20- select 10 records from the table to ensure it's loaded correctly

hive> LOAD DATA LOCAL INPATH '/employee/songs.csv' INTO TABLE assign1.songs ;

hive> select \* from assign1.songs

> limit 10 ;

21- count the number of records

hive> select count(\*) from assign1.songs;

80

22- is the hive count similar to the file count? is the data quality ok? If there is an issue, show how to resolve it

Count as the same

There are some issues in data quality

hive> create table assign1.songs(

> artist\_id STRING,

> artist\_latitude DOUBLE ,

> artist\_location STRING,

> artist\_longtitude DOUBLE,

> artist\_name STRING,

> duration DOUBLE,

> num\_songs INT ,

> song\_id STRING,

> title STRING,

> year STRING)

> ROW FORMAT DELIMITED

> FIELDS TERMINATED BY ',' lines terminated BY '\n'

> tblproperties("skip.header.line.count"="1");

hive> create table assign1.songs(

> artist\_id STRING,

> artist\_latitude DOUBLE ,

> artist\_location STRING,

> artist\_longtitude DOUBLE,

> artist\_name STRING,

> duration DOUBLE,

> num\_songs INT ,

> song\_id STRING,

> title STRING,

> year STRING)

> ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde';

23- create external table ...... to host

create external table assign1.songs\_ext(

artist\_id STRING,

artist\_latitude DOUBLE ,

artist\_location STRING,

artist\_longtitude DOUBLE,

artist\_name STRING,

duration DOUBLE,

num\_songs INT ,

song\_id STRING,

title STRING,

year STRING)

ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde';

24- load the table using put command

hive> !hdfs dfs -put /employee/songs.csv /user/hive/warehouse/assign1.db/songs\_ext ;

25- is the data readable through the table? Why?

Yes, because when we create a table we already defined serde which can read csv format correctly then make external table refer to the location of file which is loaded from local system into hdfs.

26- select [logic] from table [] through shell without accessing hive or beeline

myVar=$(eval "hive -S -e 'Select count(\*) from assign1.songs;' ")

echo $myVar

27- create a hive script that drop table if exists, creates it and load data with data.

Create\_table.hql:

DROP TABLE IF EXISTS assign1.assign1\_intern\_tab ;

use assign1;

CREATE TABLE IF NOT EXISTS assign1.assign1\_intern\_tab(

ID TINYINT ,

NAME VARCHAR(25),

AGE TINYINT,

COUNTRY VARCHAR(25))

ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' ;

LOAD DATA LOCAL INPATH '/employee/employee\_details.txt' INTO TABLE assign1\_intern\_tab;

28- execute it from shell without accessing hive CLI /beeline

hive -f create\_table.hql

29- What is a hive Temp table? how can you create it? why would someone use a temp table?

* Hive temporary tables are local to user session.
* You can use it repeatedly within user session
* Automatically deleted at the end of the session
* You can use it to manage intermediate data generated from a complex query automatically
* create temporary table t3(col1 int, col2 string);

30- move the table assign1\_intern\_tab from one database to another

hive> use assign1;

hive> alter table assign1\_intern\_tab rename to testdb.assign1\_intern\_tab;

31- check the table directory and list its components

hive> describe formatted testdb.assign1\_intern\_tab;

Location: hdfs://namenode:8020/user/hive/warehouse/testdb.db/assign1\_intern\_tab

**Assignment 2**

- Create a database named assign2

hive> create database assign2;

- Create table for songs table partitioned by artist and year. ensure the right data types are selected and the right SERDEPROPERTIES are used

create external table assign2.partitioned\_songs(

artist\_id STRING,

artist\_latitude DOUBLE ,

artist\_location STRING,

artist\_longtitude DOUBLE,

duration DOUBLE,

num\_songs INT ,

song\_id STRING,

title STRING)

partitioned by (artist\_name STRING, year STRING)

ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde';

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

hive> create table assign2.songs(

> artist\_id STRING,

> artist\_latitude DOUBLE ,

> artist\_location STRING,

> artist\_longtitude DOUBLE,

> duration DOUBLE,

> num\_songs INT ,

> song\_id STRING,

> title STRING)

> partitioned by (year STRING , artist\_name STRING)

> ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde';

- Load data into table HDFS directory using put command

hive> !hdfs dfs -put /employee/songs.csv /user/hive/warehouse/assign2.db/partitioned\_songs;

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

hive> !hdfs dfs -put /employee/songs.csv /user/hive/warehouse/assign2.db/songs;

- Run a SELECT check on the table, is there any data found? why?

There is no any data because the dynamic partition table must be loaded from another table(staging table).

- Add static partition using Alter and set partitions location in a separate directory from that of the table

hive> ALTER TABLE partitioned\_songs ADD

> PARTITION (artist\_name='AR8IEZO1187B99055E',year='2008')

> LOCATION '/partitions\_page\_views' ;

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

hive> ALTER TABLE songs ADD PARTITION (artist\_name='AR8IEZO1187B99055E',year='2008') LOCATION '/partitions\_page\_views' ;

- Load data to the created partitions

hive> !hdfs dfs -put /employee/songs.csv /partitions\_page\_views ;

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

hive> LOAD DATA LOCAL INPATH '/employee/songs.csv' INTO TABLE songs partition(artist\_name='AR8IEZO1187B99055E',year='2008');

- List the partition directories to check for presence of files

hive> describe formatted partitioned\_songs PARTITION (artist\_name='AR8IEZO1187B99055E',year='2008') ;

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

hive> describe formatted songs partition(artist\_name='AR8IEZO1187B99055E',year='2008');

- Create a staging table to host songs data

create external table assign2.staging\_songs(

artist\_id STRING,

artist\_latitude DOUBLE ,

artist\_location STRING,

artist\_longtitude DOUBLE,

artist\_name STRING,

duration DOUBLE,

num\_songs INT ,

song\_id STRING,

title STRING,

year STRING)

ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'

LOCATION '/data/songs';

- Load the data from the staging table into songs table partitions dynamically

hive> from staging\_songs

> insert overwrite table partitioned\_songs partition (artist\_name,year)

> select \* ;

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

hive> from staging\_songs

> insert overwrite table songs partition(artist\_name,year)

> select artist\_id , artist\_latitude , artist\_location , artist\_longtitude , artist\_name , duration , num\_songs , song\_id , title , year ;

- Truncate songs table and ensures no data in the table

- I’ve maded external table can not truncated

- Use multi inserts to reload the data into the table fully dynamically

hive> create table assign2.partitioned\_songs\_managed(

> artist\_id STRING,

> artist\_latitude DOUBLE ,

> artist\_location STRING,

> artist\_longtitude DOUBLE,

> duration DOUBLE,

> num\_songs INT ,

> song\_id STRING,

> title STRING)

> partitioned by (artist\_name STRING, year STRING)

> ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde';

hive> from staging\_songs

> insert overwrite table partitioned\_songs\_managed partition (artist\_name , year)

> select artist\_id , artist\_latitude , artist\_location , artist\_longtitude , duration , num\_songs , song\_id ,title , artist\_name , year ;

- Truncate

hive> truncate table partitioned\_songs\_managed ;

- Use multi inserts to reload the data statically over year and dynamically by artist

Note : - I created the table with partition (artitst\_name,year) , so it can not be done because it must the static partition comes first so I will create another table.

hive> from staging\_songs

> insert overwrite table songs partition (year='2008' ,artist\_name )

> select artist\_id , artist\_latitude , artist\_location , artist\_longtitude , duration , num\_songs , song\_id ,title , artist\_name

> where year='2008'

> insert overwrite table songs partition (year='2003' , artist\_name )

> select artist\_id , artist\_latitude , artist\_location , artist\_longtitude , duration , num\_songs , song\_id ,title , artist\_name

> where year='2003'

> insert overwrite table songs partition (year='2004' , artist\_name)

> select artist\_id , artist\_latitude , artist\_location , artist\_longtitude , duration,num\_songs , song\_id ,title , artist\_name

> where year='2004'

> insert overwrite table songs partition (year='1994' , artist\_name)

> select artist\_id , artist\_latitude , artist\_location , artist\_longtitude , duration

> , num\_songs , song\_id ,title , artist\_name

> where year='1994'

> insert overwrite table songs partition (year='0' , artist\_name)

> select artist\_id , artist\_latitude , artist\_location , artist\_longtitude , duration

> , num\_songs , song\_id ,title , artist\_name

> where year='0'

> insert overwrite table songs partition (year='2000' , artist\_name)

> select artist\_id , artist\_latitude , artist\_location , artist\_longtitude , duration

> , num\_songs , song\_id ,title , artist\_name

> where year='2000'

> insert overwrite table songs partition (year='2005' , artist\_name)

> select artist\_id , artist\_latitude , artist\_location , artist\_longtitude , duration

> , num\_songs , song\_id ,title , artist\_name

> where year='2005'

> insert overwrite table songs partition (year='1999' , artist\_name)

> select artist\_id , artist\_latitude , artist\_location , artist\_longtitude , duration

> , num\_songs , song\_id ,title , artist\_name

> where year='1999'

> insert overwrite table songs partition (year='1993' , artist\_name)

> select artist\_id , artist\_latitude , artist\_location , artist\_longtitude , duration

> , num\_songs , song\_id ,title , artist\_name

> where year='1993'

> insert overwrite table songs partition (year='1985' , artist\_name)

> select artist\_id , artist\_latitude , artist\_location , artist\_longtitude , duration

> , num\_songs , song\_id ,title , artist\_name

> where year='1985'

> insert overwrite table songs partition (year='1964' , artist\_name)

> select artist\_id , artist\_latitude , artist\_location , artist\_longtitude , duration

> , num\_songs , song\_id ,title , artist\_name

> where year='1964'

> insert overwrite table songs partition ( year='1992' , artist\_name)

> select artist\_id , artist\_latitude , artist\_location , artist\_longtitude , duration

> , num\_songs , song\_id ,title , artist\_name

> where year='1992'

> insert overwrite table songs partition (year='1661' , artist\_name)

> select artist\_id , artist\_latitude , artist\_location , artist\_longtitude , duration

> , num\_songs , song\_id ,title , artist\_name

> where year='1661'

> insert overwrite table songs partition (year='1972' , artist\_name)

> select artist\_id , artist\_latitude , artist\_location , artist\_longtitude , duration

> , num\_songs , song\_id ,title , artist\_name

> where year='1972'

> insert overwrite table songs partition (year='1997' , artist\_name)

> select artist\_id , artist\_latitude , artist\_location , artist\_longtitude , duration

> , num\_songs , song\_id ,title , artist\_name

> where year='1997'

> insert overwrite table songs partition (year='1986', artist\_name)

> select artist\_id , artist\_latitude , artist\_location , artist\_longtitude , duration

> , num\_songs , song\_id ,title , artist\_name

> where year='1986'

> insert overwrite table songs partition (year='1987' , artist\_name)

> select artist\_id , artist\_latitude , artist\_location , artist\_longtitude , duration

> , num\_songs , song\_id ,title , artist\_name

> where year='1987'

> insert overwrite table songs partition (year='1984',artist\_name)

> select artist\_id , artist\_latitude , artist\_location , artist\_longtitude , duration

> , num\_songs , song\_id ,title , artist\_name

> where year='1984'

> insert overwrite table songs partition (year='1982',artist\_name)

> select artist\_id , artist\_latitude , artist\_location , artist\_longtitude , duration

> , num\_songs , song\_id ,title , artist\_name

> where year='1982'

> insert overwrite table songs partition (year='1969',artist\_name)

> select artist\_id , artist\_latitude , artist\_location , artist\_longtitude , duration

> , num\_songs , song\_id ,title , artist\_name where year='1969';

- Use CREATE TABLE LIKE statement to create a table with a schema similar to the staging table. The new table should be able to read Avro files

CREATE TABLE staging\_avro LIKE songs;  
ALTER TABLE staging\_avro  
SET SERDE 'org.apache.hadoop.hive.serde2.avro.AvroSerDe';

- Use CREATE TABLE LIKE statement to create a table with a schema similar to the staging table. The new table should be able to read Parquet files

CREATE TABLE staging\_parquet LIKE songs;  
ALTER TABLE staging\_parquet  
SET SERDE 'parquet.hive.serde.ParquetHiveSerDe';

- use the avro-tools getschema [avro\_file\_name] command to get the avro schema of the file.

-

**Assignment 3**

File to be used events.csv

1. Create a table with the right data types and SERDEPROPERTIES to host the data from the events.csv files

hive> create table assign2.events\_mg(

> artist STRING,

> auth STRING,

> firstName STRING ,

> gender STRING,

> itemInSession STRING,

> lastName STRING,

> length DOUBLE ,

> level STRING,

> location STRING,

> method STRING,

> page STRING,

> registeration STRING,

> sessionId INT,

> song STRING,

> status INT,

> ts STRING,

> userAgent STRING,

> userId INT)

> ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde';

1. Load the file from local filesystem to the hive table using LOAD statement

hive> LOAD DATA LOCAL INPATH '/employee/events.csv' INTO TABLE assign2.events\_mg ;

1. Select the user, session, first song and last song played per session

select userId , sessionId , FIRST\_VALUE(song)OVER(PARTITION BY sessionId ORDER BY ts

ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) as first\_song ,

> LAST\_VALUE(song)OVER(PARTITION BY sessionId ORDER BY ts ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) as Last\_song

> from events\_mg ;

1. Rank users according to the number of distinct songs they played. If two users shared the same counts, they should have the same rank

SELECT t1.userId , RANK()OVER(ORDER BY t1.count\_distinct) as ranking

> FROM

> (SELECT userId , COUNT(DISTINCT song) as count\_distinct

> FROM events\_mg

> GROUP BY userId )t1 ;

1. Rank users according to the number of distinct songs they played. If two users shared the same counts, each user should have his/her own number. Note that records indicating s a played song are those with column ‘page’ equals to NextPage

SELECT t1.userId , ROW\_NUMBER() OVER(ORDER BY t1.count\_distinct) as ranking

> FROM

> (SELECT userId , COUNT(DISTINCT song) as count\_distinct

> FROM events\_mg

> WHERE page='NextSong'

> GROUP BY userId)t1;

1. In the same table, show the count of songs played per location and artists, per location only and the total count

hive> SELECT location , artist , COUNT(song) as count\_songs

> FROM events\_mg

> GROUP BY location , artist

> GROUPING SETS ((location,artist),location,()) ;

1. In the same table, show the count of songs played per location and artists, per location only , per artist only and the total count

hive> SELECT location , artist , COUNT(song) as count\_songs

> FROM events\_mg

> GROUP BY location , artist

> GROUPING SETS ((location,artist),location,artist,()) ;

1. For each song played by a user, get the previous song and next song played. Get the count of each path, and fetch the top 10 paths found

hive> SELECT userId , song ,

> LEAD(song,1) OVER (PARTITION BY userId ORDER BY ts) as NEXT\_SONG,

> LAG(song,0) OVER (PARTITION BY userId ORDER BY ts) as PREV\_SONG

> FROM events\_mg

> WHERE page='NextSong' ;

1. Select userid, song ordered by userid, song, ts. The query should be written to run on a single reducer

hive> SELECT userId , song , ts

> FROM events\_mg

> ORDER BY userId , song ,ts ;

1. Select userid, song ordered by userid, song, ts. The query should be written to run on a multiple reducers

hive> SELECT userId , song , ts

> FROM events\_mg

> CLUSTER BY userId , song ,ts ;