**Assignment 1**

1- create a database named assign1

create database assign1;

2- What is the database path on HDFS?

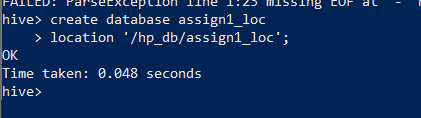
Describe database extended assign1;



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

Create database assign1\_loc

Location ‘/hp\_db/assign1\_loc’;



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

Use assign1;

create table assign1\_intern\_tab (

eid int,

ename string,

age int,

jobtype string,

storeid int,

storelocation string,

salary bigint,

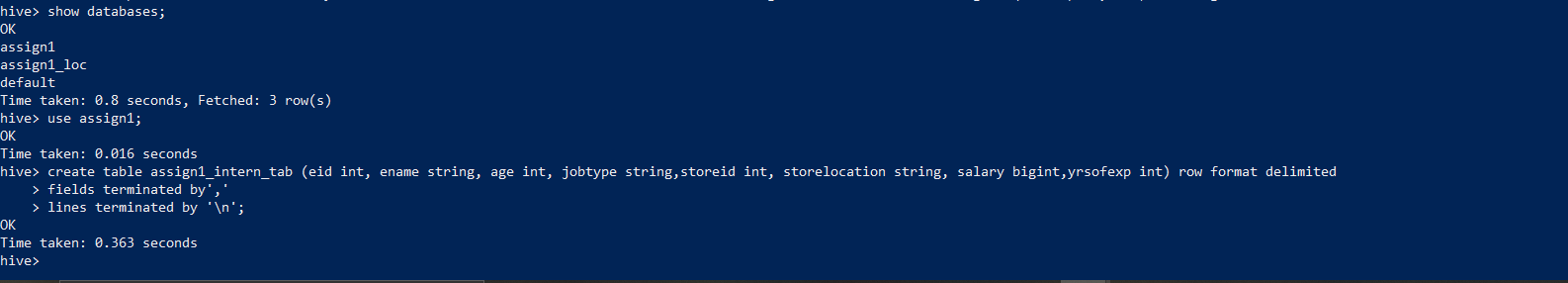
yrsofexp int

)

row format delimited

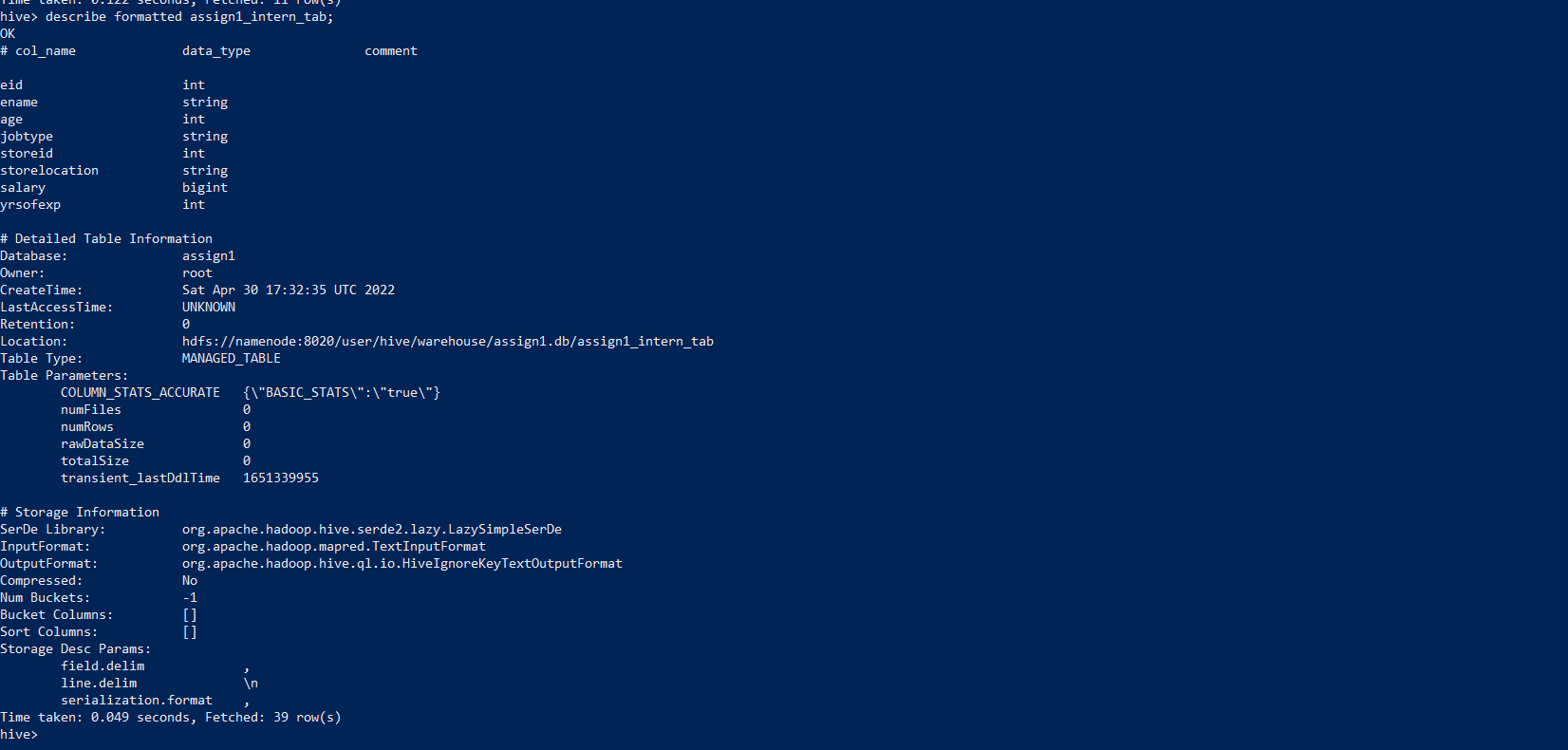
fields terminated by ','

lines terminated by '\n'



5- What is the table path in HDFS?

Describe formatted assign1\_intern\_tab;



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

Method1:

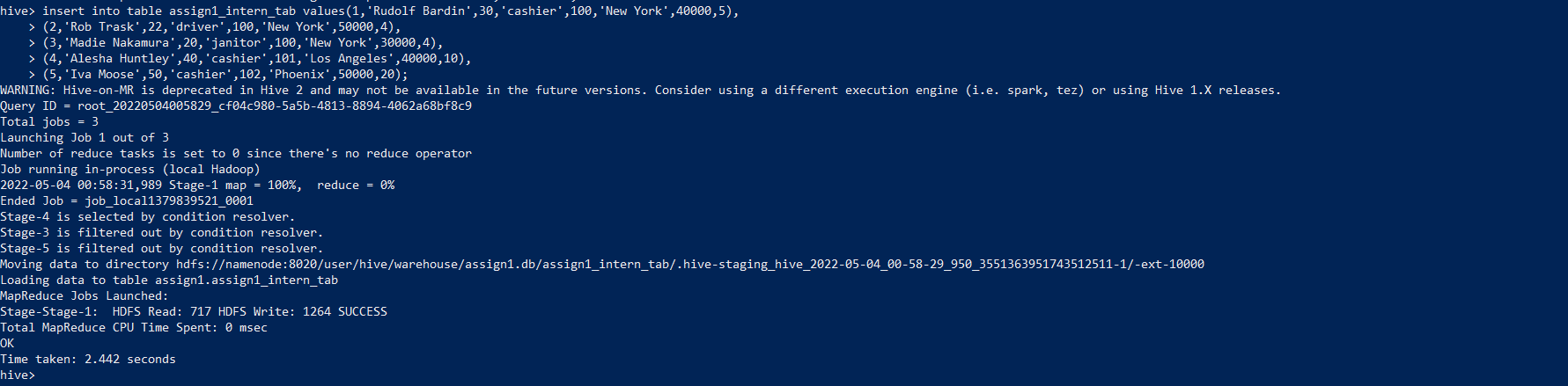
insert into table assign1\_intern\_tab values(1,'Rudolf Bardin',30,'cashier',100,'New York',40000,5),

(2,'Rob Trask',22,'driver',100,'New York',50000,4),

(3,'Madie Nakamura',20,'janitor',100,'New York',30000,4),

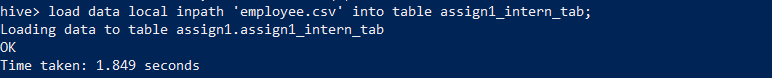
(4,'Alesha Huntley',40,'cashier',101,'Los Angeles',40000,10),

(5,'Iva Moose',50,'cashier',102,'Phoenix',50000,20);



Method2:

load data local inpath 'employee.csv' into table assign1\_intern\_tab;



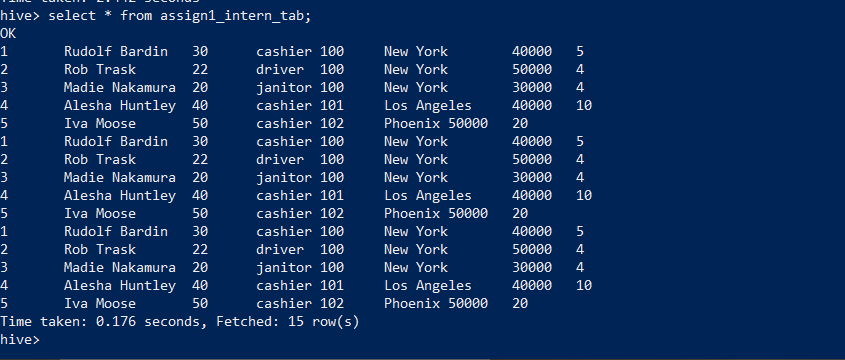
Method3:

!hadoop fs -mkdir /shaden;

!hadoop fs -put employee.csv /shaden;

load data inpath '/shaden/employee.csv' into table assign1\_intern\_tab ;

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



\*\*\*the result shows 15 rows as the data was loaded in three different ways

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

Use the assign1\_loc;

create external table assign1\_extern\_tab (

eid int,

ename string,

age int,

jobtype string,

storeid int,

storelocation string,

salary bigint,

yrsofexp int

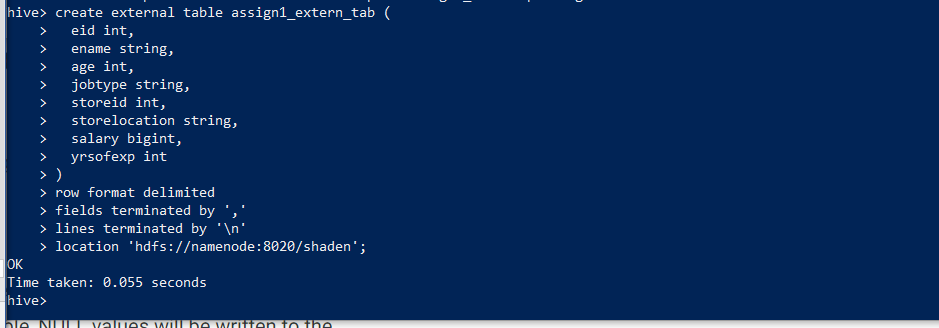
)

row format delimited

fields terminated by ','

lines terminated by '\n'

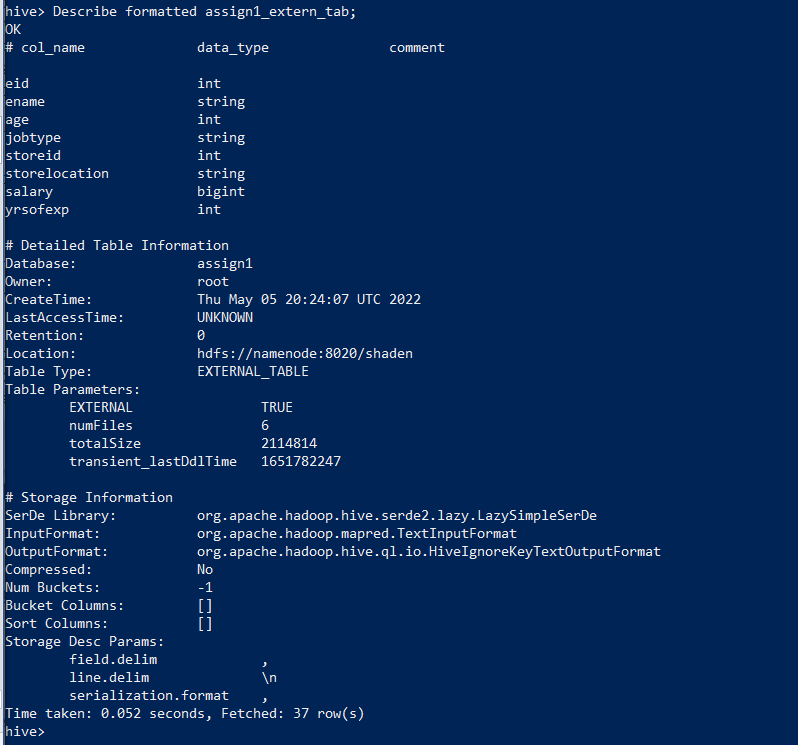
location 'hdfs://namenode:8020/shaden';



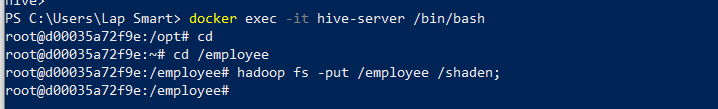
9- What is the table path in HDFS?

Describe formatted assign1\_extern\_tab;

Output: Hdfs://namenode:8020/shaden

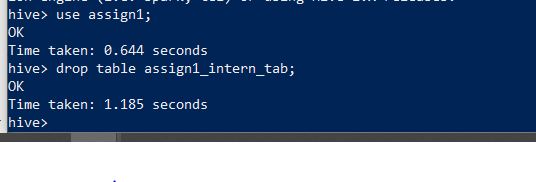


10- move the data from local filesystem to the directory created in step 7



Hadoop fs –put /employee /shaden;

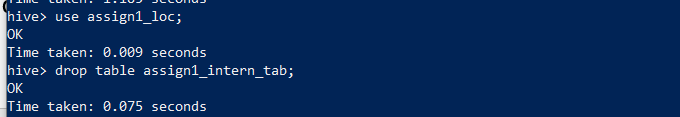
11- drop both tables, is the data present after deletion or not?



Use assign1;

Drop table assign1\_intern\_tab;

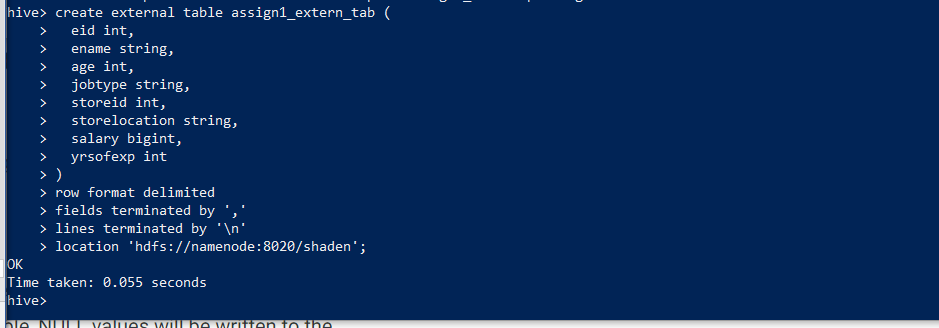
The other table:



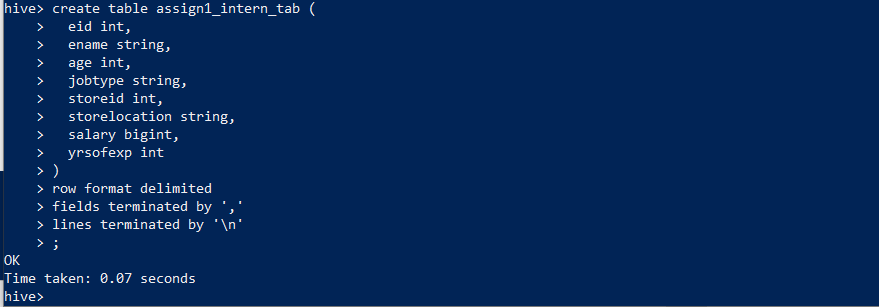
Use assign1\_loc;

Drop table assign1\_extern\_tab;

12- recreate both tables

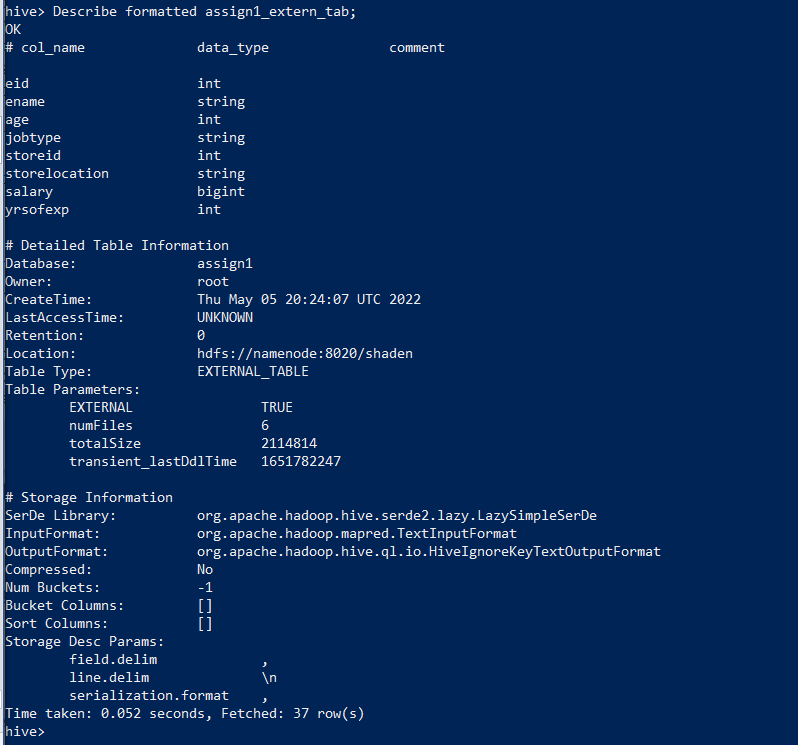


The other table:

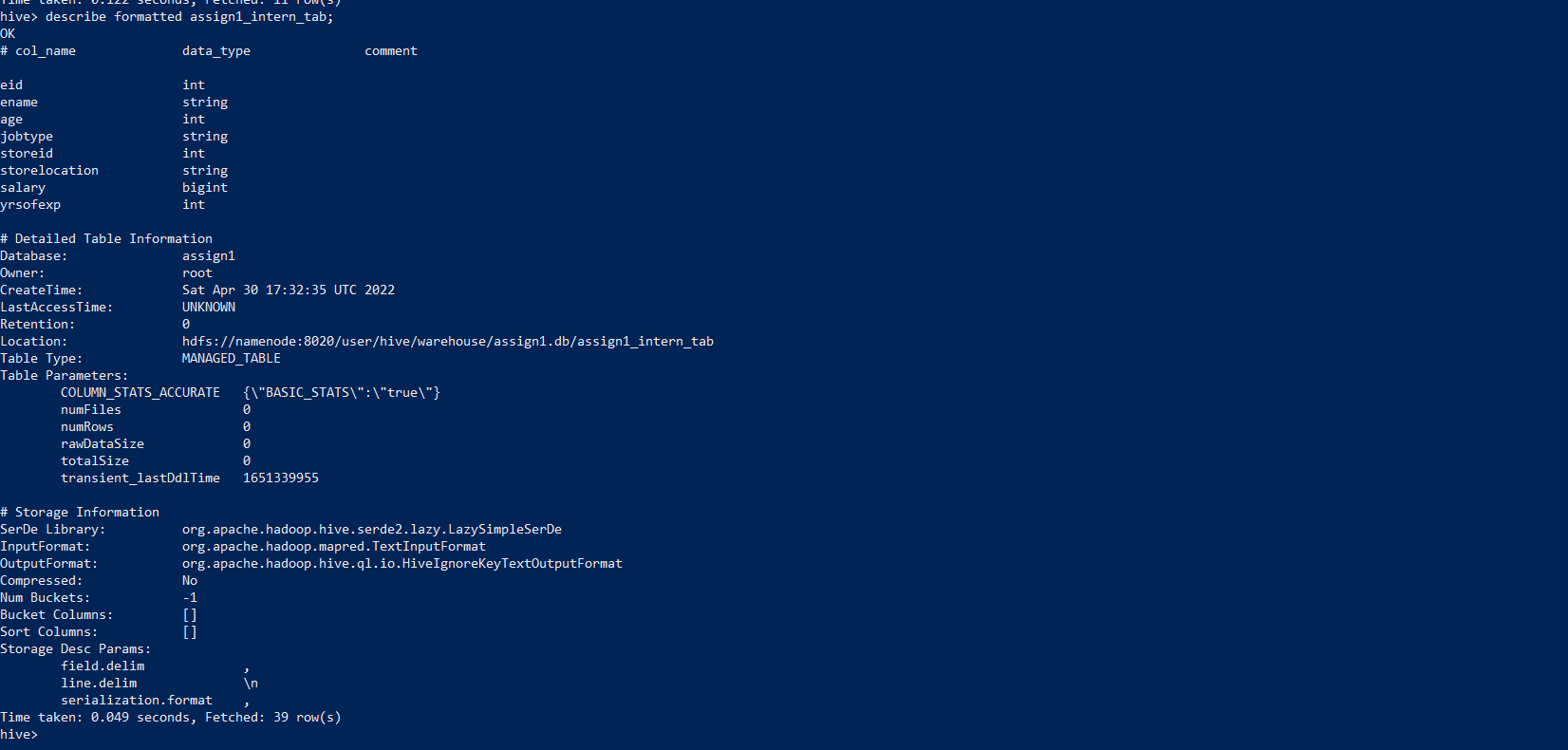


13- list both table directories

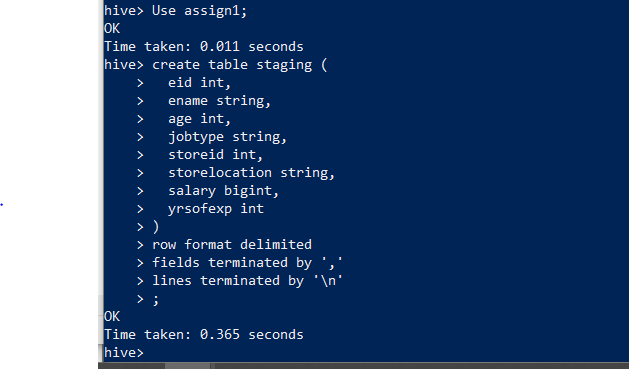
Describe formatted assign1\_extern\_tab;



Describe formatted assign1\_intern\_tab;



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



Use assign1;

create table staging (

eid int,

ename string,

age int,

jobtype string,

storeid int,

storelocation string,

salary bigint,

yrsofexp int

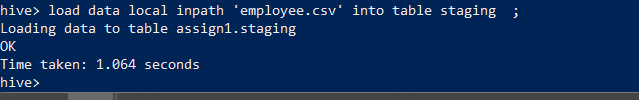
)

row format delimited

fields terminated by ','

lines terminated by '\n';

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

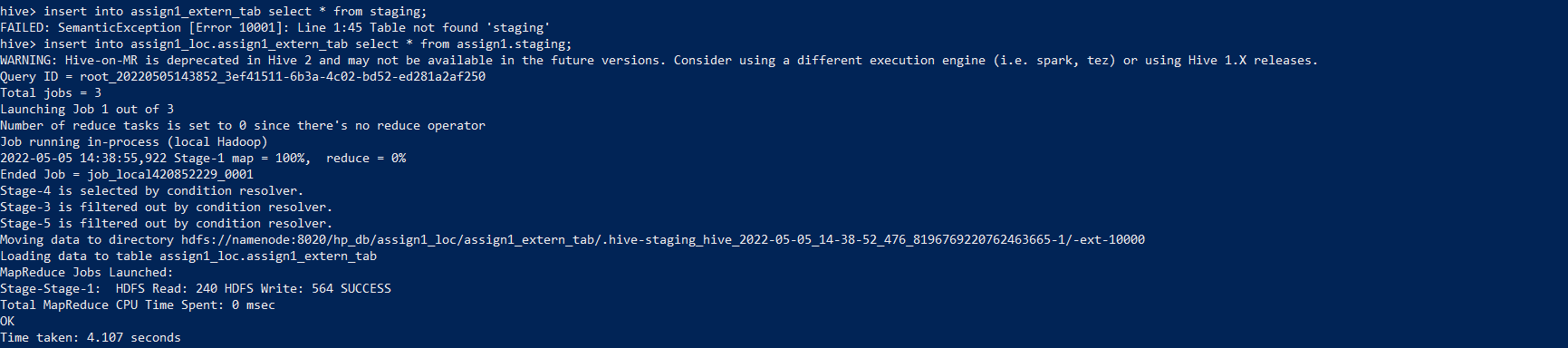


load data local inpath 'employee.csv' into table staging;

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

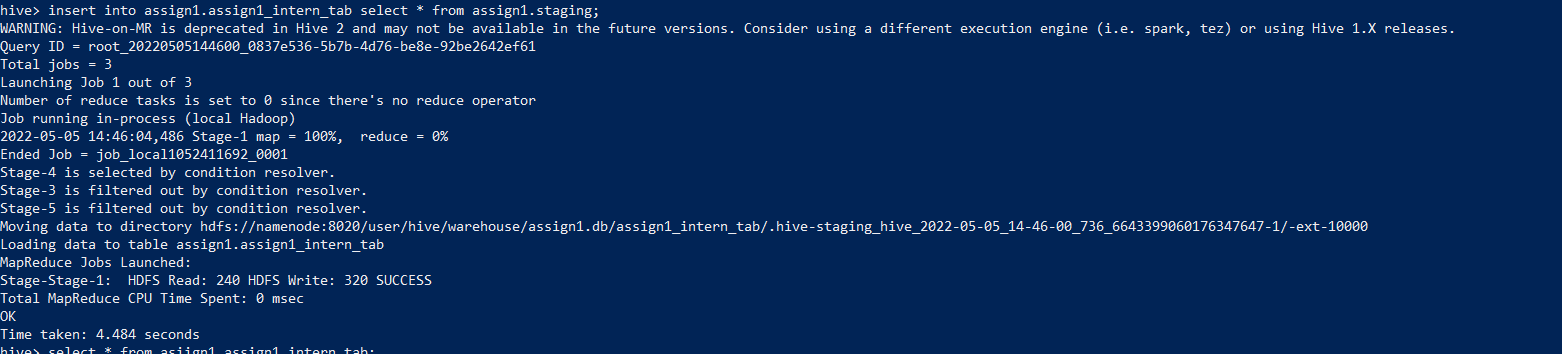
assign1\_extern\_tab loading:

insert into assign1\_loc.assign1\_extern\_tab select \* from assign1.staging;



assign1\_intern\_tab loading:

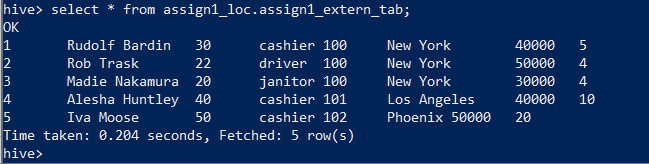
insert into assign1.assign1\_intern\_tab select \* from assign1.staging;

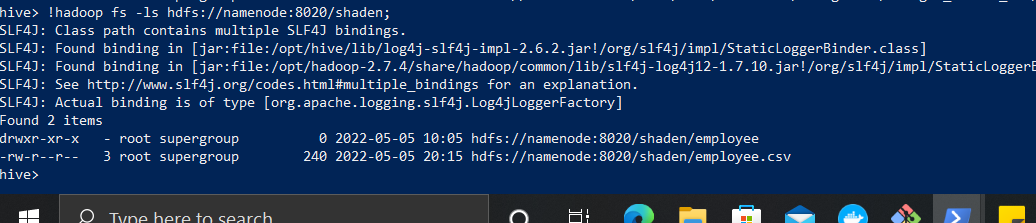


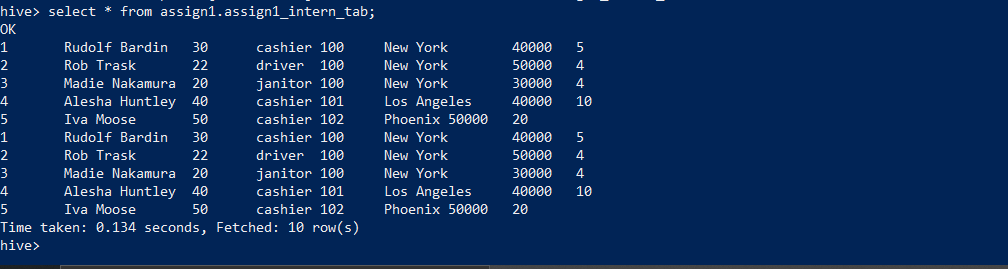
17- List both directory tables and check if there is data or not

!hadoop fs -ls / hdfs://namenode:8020/user/hive/warehouse/assign1.db/assign1\_intern\_tab;  
!hadoop fs -ls hdfs://namenode:8020/shaden;



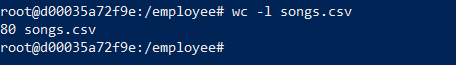






18- count the lines inside the file songs

wc –l songs.csv



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

create table songs\_tab6(

artist\_id string,

artist\_latitude string,

artist\_location string,

artist\_longitude string,

artist\_name string,

duration string,

num\_songs string,

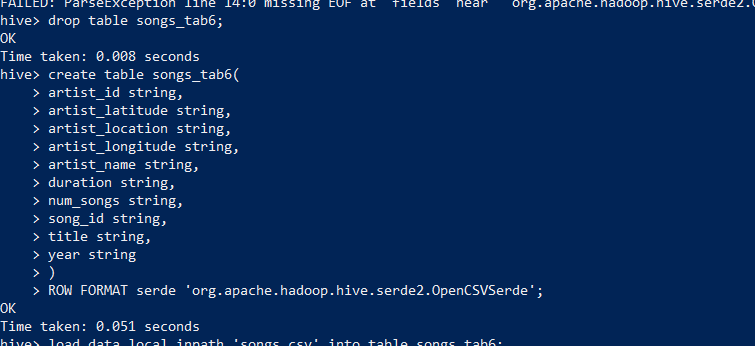
song\_id string,

title string,

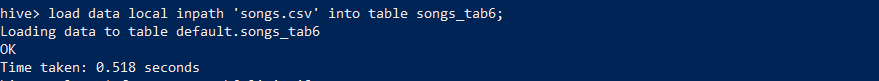
year string

)

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

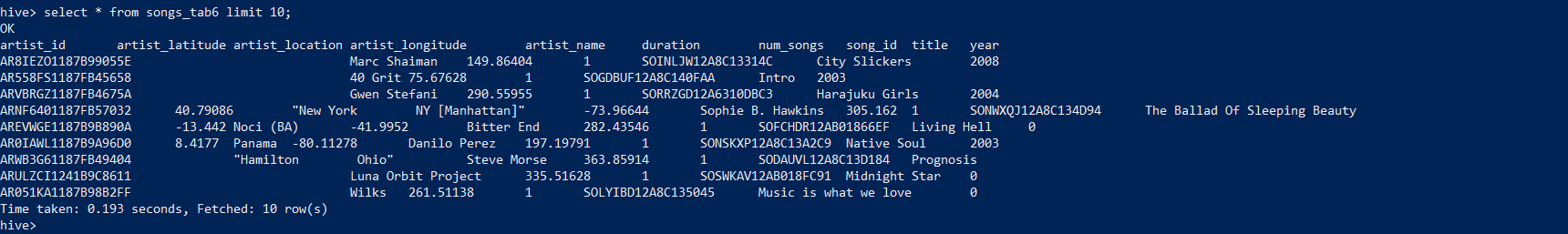


load data local inpath 'songs.csv' into table songs\_tab6;



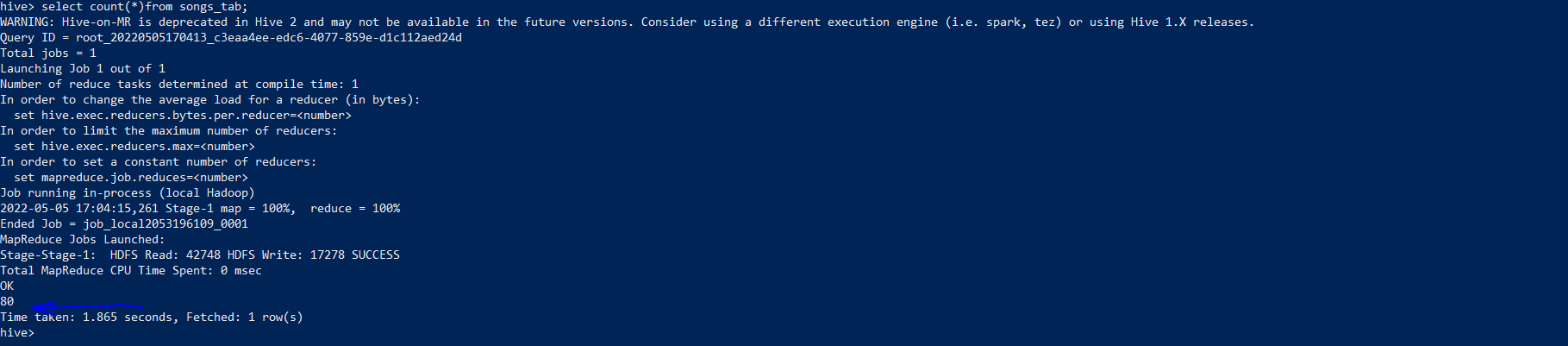
20- select 10 records from the table to ensure it's loaded correctly

select \* from songs\_tab6 limit 10;

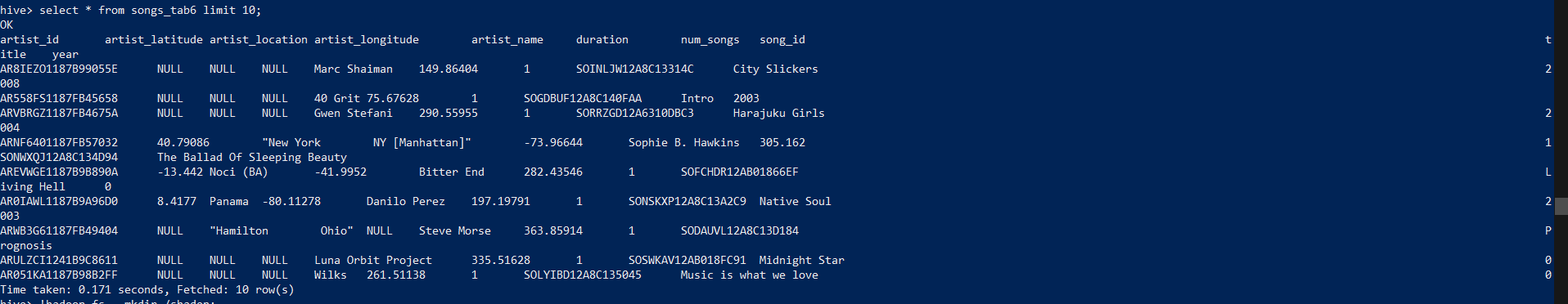


21- count the number of records

Select count (\*) from songs\_tab;



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



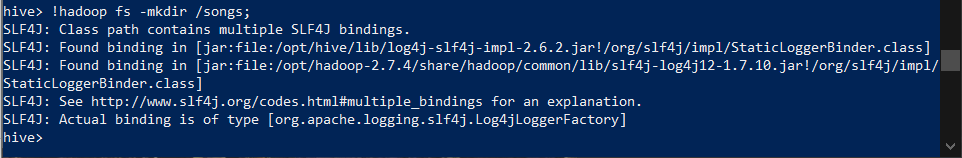
Yes same count. Data quality is not ok. Replace the string gaps with ‘null’

ALTER TABLE songs\_tab6

SET TBLPROPERTIES ('serialization.null.format'='');

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

!hadoop fs –mkdir /songs;



create external table songs\_extern\_tab(

artist\_id string,

artist\_latitude string,

artist\_location string,

artist\_longitude string,

artist\_name string,

duration string,

num\_songs string,

song\_id string,

title string,

year string

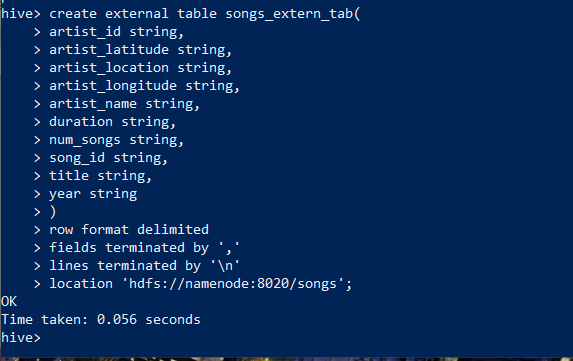
)

row format delimited

fields terminated by ','

lines terminated by '\n'

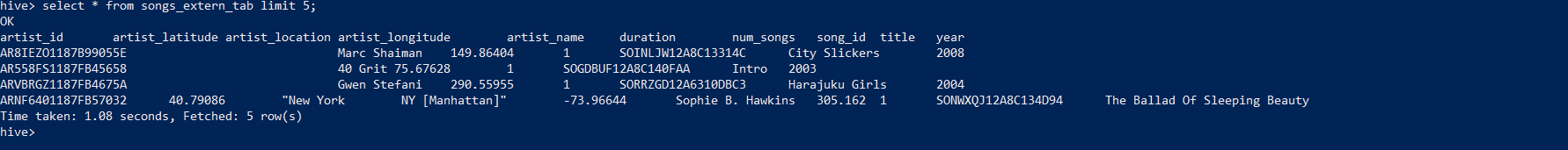
location 'hdfs://namenode:8020/songs';



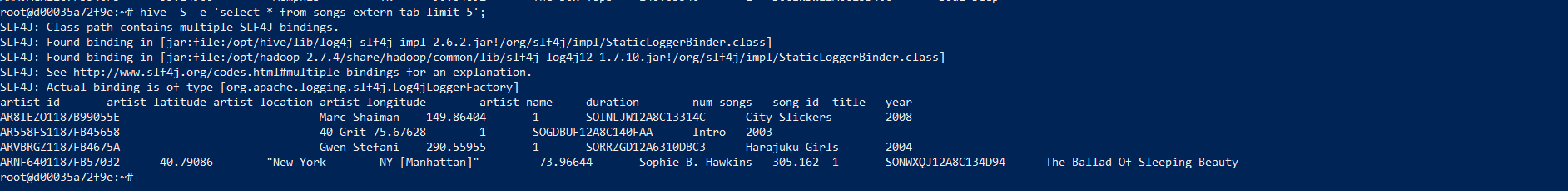
24- load the table using put command

Hadoop fs –put songs.csv /songs;



25- is the data readable through the table? Why? No , the artist location is not readable . proper serde properties was not applied .

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



Hive –S –e ‘select \*from songs\_extern\_tab limit 5’;

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

Sample.hql

DROP TABLE IF EXISTS assign1\_intern\_tab;

create table if not exists assign1\_intern\_tab (emp\_id int , emp\_name string, age int, job\_title string, dept\_id int, city string, salary int, kilos\_from\_home int)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ',';

load data local inpath 'employee.csv' into table assign1\_intern\_tab;

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

Hive –f sample.hql

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

What:

temporary table in Hive is used to store the data temporarily within an active session and the temporary tables get automatically removed when the active session end.

Create temporary table:

CREATE TEMPORARY TABLE assign6.employee\_tmp (

id int,

name string,

age int,

gender string)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ',';

Why:

temporary tables are created at the run time to store the intermediate data that are used to perform further data processing. once the processing is done either you can explicitly drop the temporary table or session termination will drop these tables.

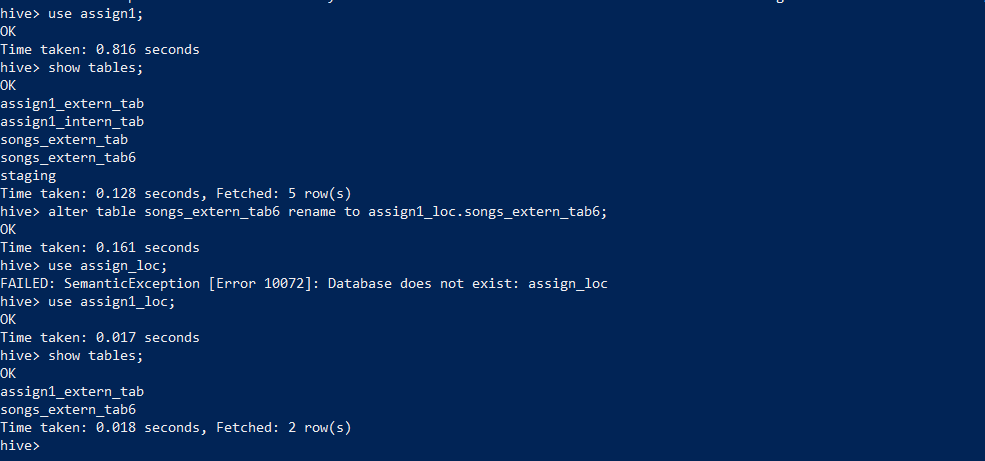
30- move the table assign1\_intern\_tab from one database to another

use old\_database;

alter table table\_a rename to new\_database.table\_a;

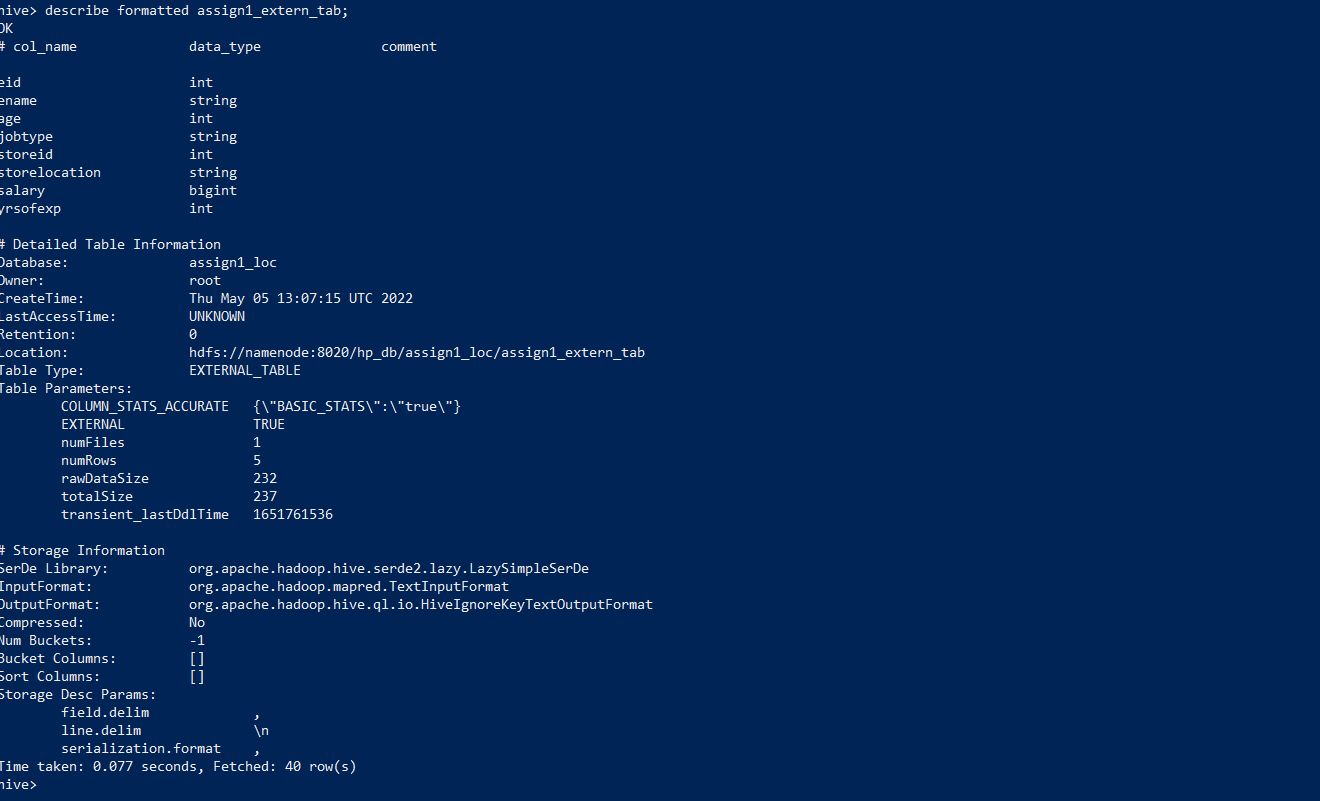
use assign1;

alter table songs\_extern\_tab6 rename to assign1\_loc.songs\_extern\_tab6;



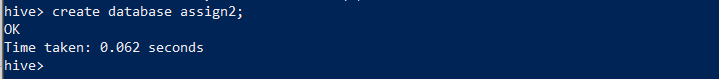
31- check the table directory and list its components

Describe formatted assign1\_extern\_tab;



**Assignment 2**

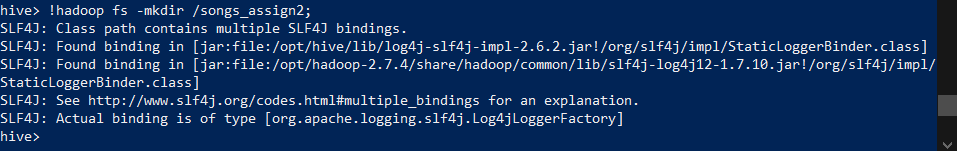
1. Create a database named assign2



Create table assign2;

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

!hadoop fs -mkdir -p /songs\_assign2/assign2;



create external table songs\_extern2(

artist\_id string,

artist\_latitude string,

artist\_location string,

artist\_longitude string,

duration string,

num\_songs string,

song\_id string,

title string

)

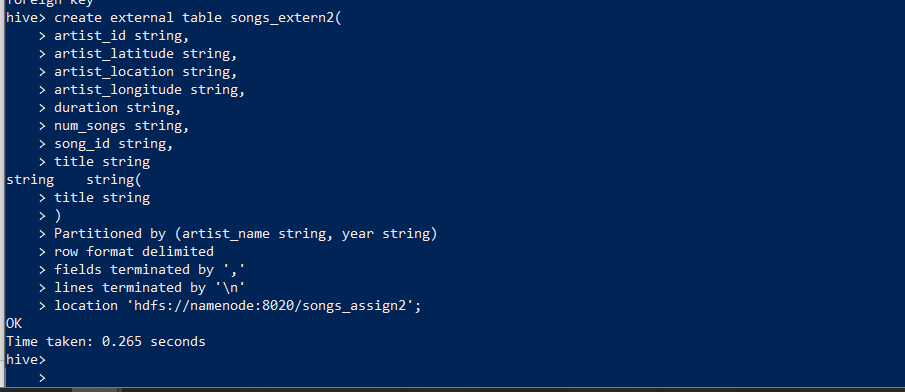
Partitioned by (artist\_name string, year string)

row format delimited

fields terminated by ','

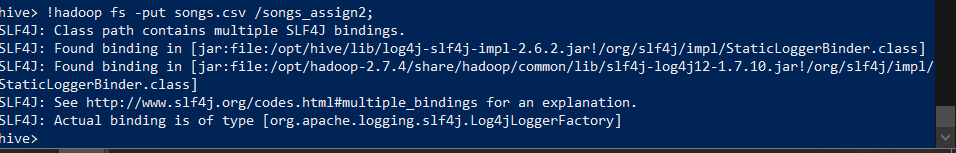
lines terminated by '\n'

location 'hdfs://namenode:8020/songs\_assign2';



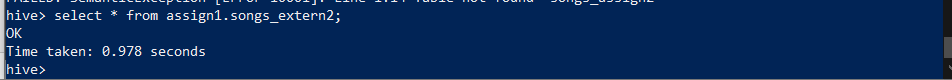
1. Load data into table HDFS directory using put command

!hadoop fs -put songs.csv /songs\_assign2;



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

Select \* from songs\_extern2;

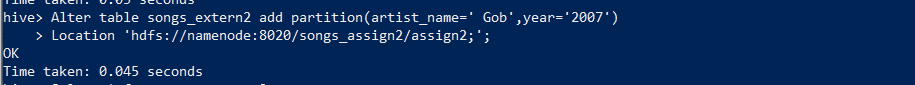


No data, as the table partitioned so each partition should point to certain data location.

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

Alter table songs\_extern2 add partition(artist\_name=' Gob',year='2007')

Location 'hdfs://namenode:8020/songs\_assign2/assign2';



1. Load data to the created partitions

!hadoop fs -put songs.csv /songs\_assign2/assign2;

1. List the partition directories to check for presence of files

!hadoop fs –ls /songs\_assign2/assign2;

Select \* from songs\_extern2;

1. Create a staging table to host songs data

create table staging2 (

artist\_id string,

artist\_latitude string,

artist\_location string,

artist\_longitude string,

artist\_name string,

duration string,

num\_songs string,

song\_id string,

title string,

year string

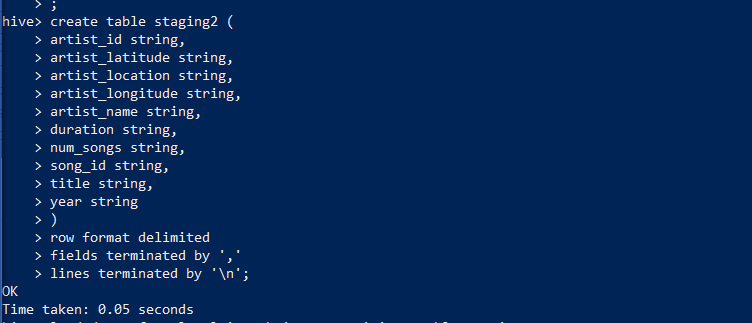
)

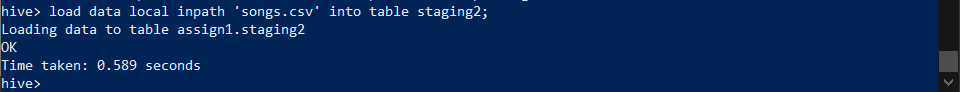
row format delimited

fields terminated by ','

lines terminated by '\n';

load data local inpath 'songs.csv' into table staging2;





1. Load the data from the staging table into songs table partitions dynamically

Insert overwrite table songs\_extern2 partition (artist\_name , year)

select artist\_id,

artist\_latitude,

artist\_location,

artist\_longitude,

artist\_name,

duration,

num\_songs,

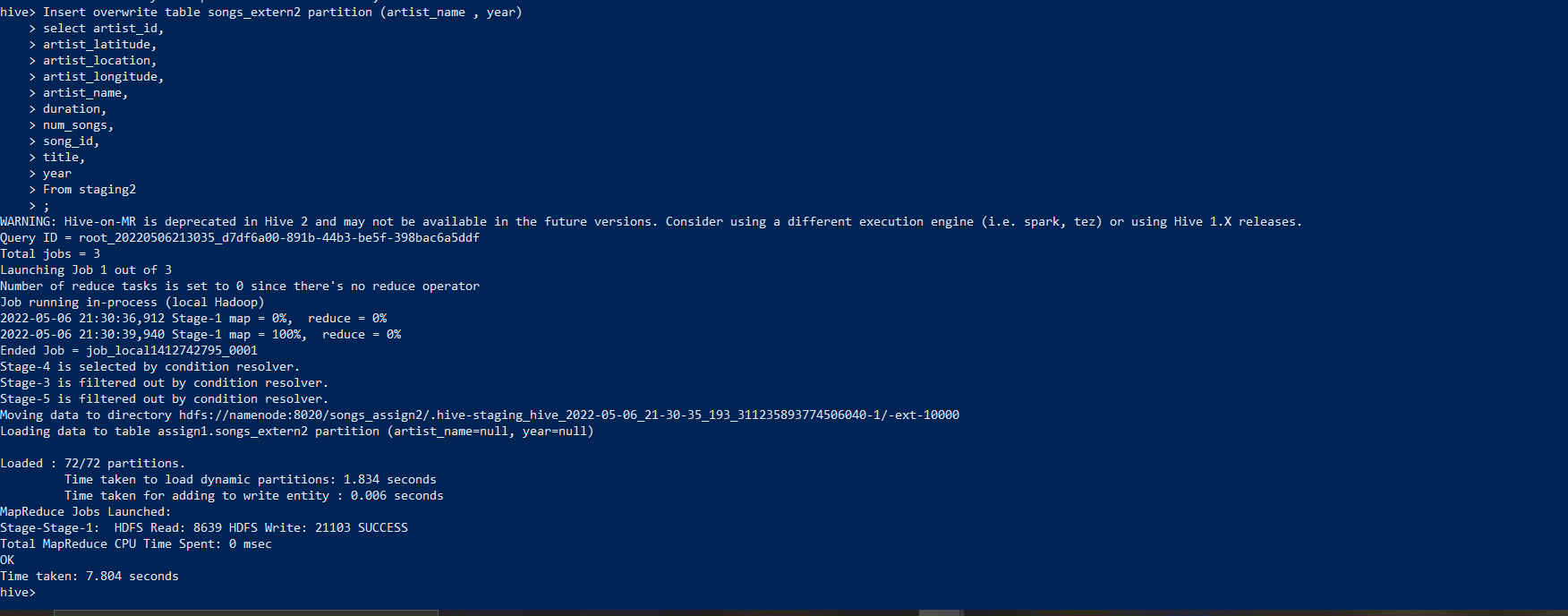
song\_id,

title,

year

From staging2

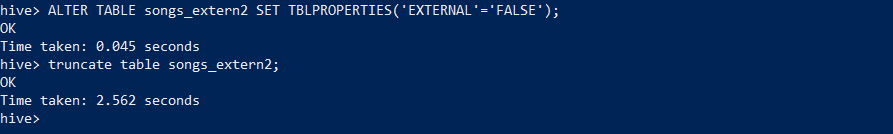
;

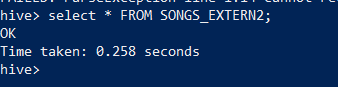


1. Truncate songs table and ensures no data in the table

truncate table songs\_extern2;

Select \* from songs\_ extern2;





1. Use multi inserts to reload the data into the table fully dynamically

Insert overwrite table songs\_extern2 partition (artist\_name , year)

select artist\_id,

artist\_latitude,

artist\_location,

artist\_longitude,

artist\_name,

duration,

num\_songs,

song\_id,

title,

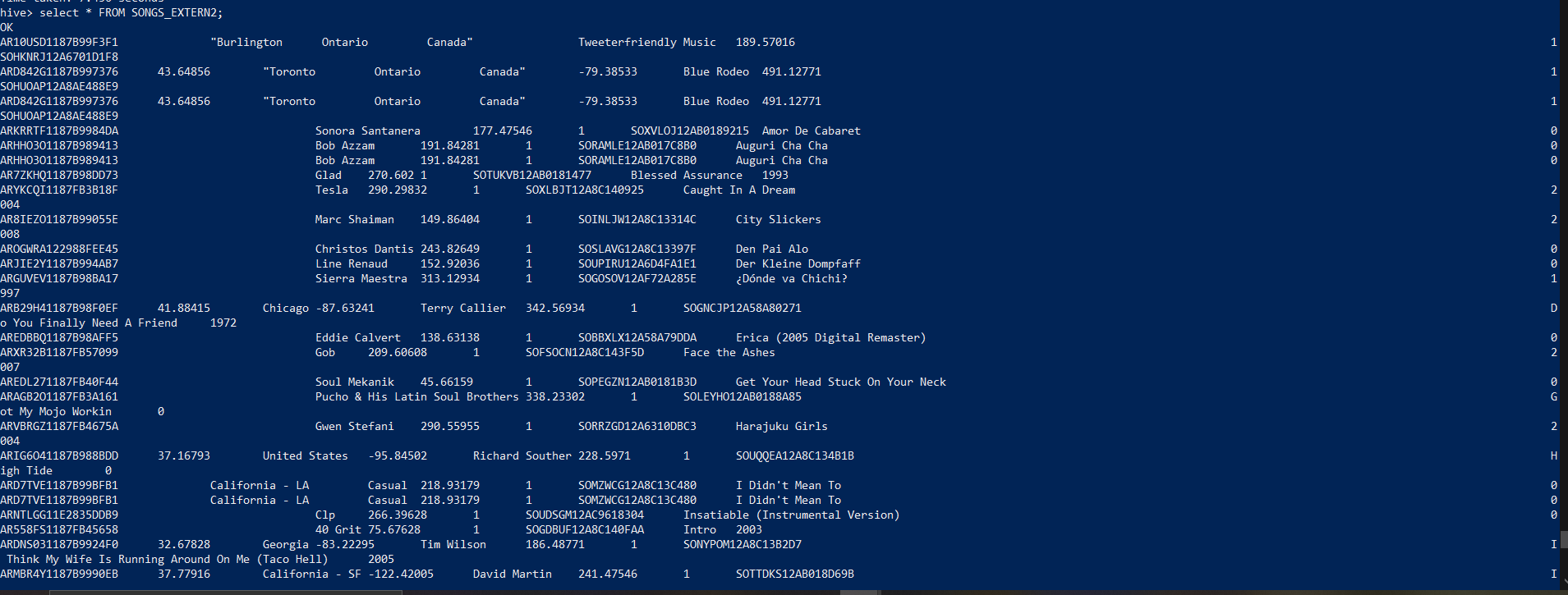
year

From staging2

;

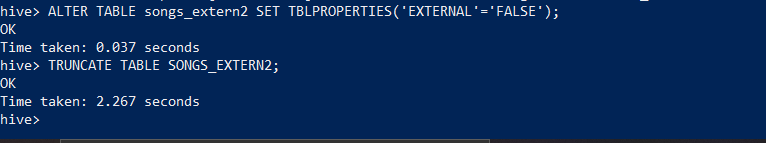


SELECT \* FROM SONGS\_EXTERN2;



1. Truncate

Truncate table songs\_EXTERN2;



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

Drop table songs\_extern2;

create table songs\_extern2(

artist\_id string,

artist\_latitude string,

artist\_location string,

artist\_longitude string,

duration string,

num\_songs string,

song\_id string,

title string

)

Partitioned by (year string,artist\_name string)

row format delimited

fields terminated by ','

lines terminated by '\n';

Insert overwrite table songs\_extern2 partition (year='2007', artist\_name)

select artist\_id,

artist\_latitude,

artist\_location,

artist\_longitude,

artist\_name,

duration,

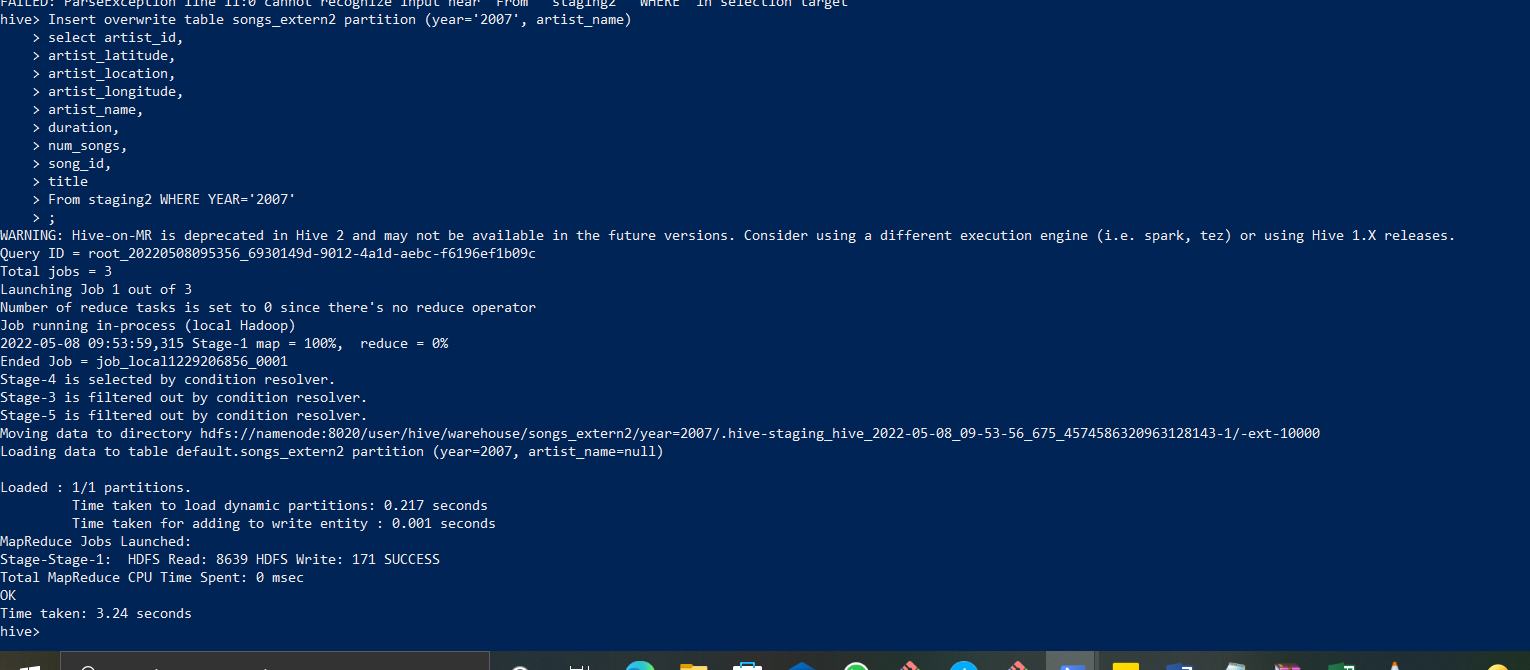
num\_songs,

song\_id,

title

From staging2 WHERE YEAR='2007'

;



1. 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

{

"type" : "record",

"name" : "Likestaging",

"namespace" : "namespace": "com.loudacre.data",

"fields" : [

{ "name" : "eid" , "type" : "int"},

{ "name" : "ename" , "type" : "string"},

{ "name" : "age" , "type" : "int"},

{ "name" : "jobtype" , "type" : "string"},

{ "name" : "storeid" , "type" : "int"},

{ "name" : "storelocation" , "type" : "string"},

{ "name" : "salary" , "type" : "bigint"},

{ "name" : "yrsofexp" , "type" : "int"}

]

}

1. 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\_like\_parquet (Eid int,

ename string,

age int,

jobtype string,

storied int,

storelocation string,

name bigint,

Yrsofexp int) STORED AS PARQUET;

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

java -jar "avro-tools-1.8.2.jar" getschema menna.avro

"avro-tools-1.8.2.jar" getschema shaden.avro

- File to be used events.csv

**Assignment 3**

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

use assign1;

create table event\_tab(

artist string,

auth string,

firstName string,

gender string,

itemInSession string,

lastName string,

length string,

level string,

location string,

method string,

page string,

registration string,

sessionId string,

song string,

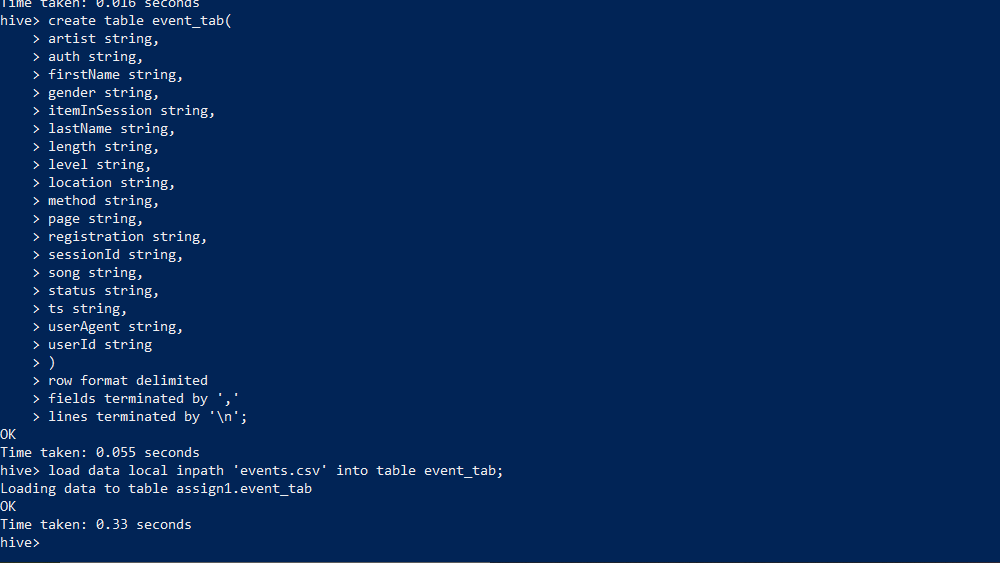
status string,

ts string,

userAgent string,

userId string

)

row format serde 'org.apache.hadoop.hive.serde2.OpenCSVSerde';

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

load data local inpath 'events.csv' into table event\_tab;

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

Select userId, song, sessionId, first\_value(song)over(partition by sessionId order by itemInSession ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED following) , last\_value(song)over(partition by sessionId order by itemInSession ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED following)

from event\_tab limit 50;

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 userId,count(distinct song), RANK() OVER (Order BY COUNT(distinct song) DESC) FROM event\_tab group by userId;

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 userId,count(distinct song), Row\_number() OVER (Order BY COUNT(distinct song) DESC) FROM event\_tab group by userId;

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

SELECT COUNT(song) FROM event\_tab 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

SELECT COUNT(song) FROM event\_tab 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
2. Select userid, song ordered by userid, song, ts. The query should be written to run on a single reducer

Select userId,song,ts from event\_tab 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

Select userId,song,ts from event\_tab cluster by userId,song,ts;