**Assignment 1**

1- create a database named assign1

CREATE DATABASE assign1;

2- What is the database path on HDFS?

!hdfs dfs -ls /user/hive/warehouse

describe database assign1;

/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\_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

create table assign1.assign1\_intern\_tab(

name STRING,

age INT,

job\_title STRING,

mgr\_id INT,

loc STRING,

salary FLOAT,

no INT

)

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

stored as textfile

TBLPROPERTIES("skip.header.line.count"="1");

5- What is the table path in HDFS?

describe formatted assign1.assign1\_intern\_tab;

/user/hive/warehouse/assign1.db/assign1\_intern\_tab

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

!hadoop dfs -put employee.csv /user/hive/warehouse/assign1.db/assign1\_intern\_tab;

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

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

select \* from assign1.assign1\_intern\_tab

limit 10;

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

CREATE external table assign1\_loc.assign1\_intern\_tab (name STRING,age INT,job\_title STRING,mgr\_id INT,loc STRING,salary FLOAT,no INT)

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

stored as textfile

location '/hp\_db/assign1\_loc'

TBLPROPERTIES("skip.header.line.count"="1");

9- What is the table path in HDFS?

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

/hp\_db/assign1\_loc

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

Load data local inpath 'employee.csv' into table assign1\_extern\_tab;

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

DROP TABLE assign1\_intern\_tab;--> internal tab data deleted also its meta data

DROP TABLE assign1\_extern\_tab ;--> external data in hdfs and meta data deleted.

12- recreate both tables

Done!

13- list both table directories

!hdfs dfs -ls /hp\_db/assign1\_loc;

!hdfs dfs -ls /user/hive/warehouse/assign1.db/assign1\_intern\_tab;

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

CREATE table assign1.staging (name STRING, age INT,job\_title STRING,mgr\_id INT,loc STRING,salary FLOAT,no INT) row format serde 'org.apache.hadoop.hive.serde2.OpenCSVSerde'

stored as textfile;

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 to the staging table using INSERT SELECT statement

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

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

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

Data still in files in tables dir .

18- count the lines inside the file songs

! wc -l /employee/songs.csv;

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

CREATE table songs (id STRING,number FLOAT,city STRING,abb STRING,number\_to FLOAT,Category STRING,other\_num FLOAT,availablity INT,code STRING,type STRING,year DATE);

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

select \* from songs

limit 10;

21- count the number of records

Select count (\*) from songs;

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 both have count 80 line/record.

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

CREATE external table assign1.song (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'

stored as textfile

location'/user/song'

TBLPROPERTIES("skip.header.line.count"="1");

24- load the table using put command

!hadoop dfs -put /employee/songs.csv /user/song;

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

Include a lot of null values but its readable more than in file .

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

!cat /employee/songs.csv;

Cat table file

Or hive -s -e

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

show tables;

DROP TABLE IF EXISTS employee ;

show tables;

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

hive -f script.hql

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

Manage intermediate data generated during a large or complex query execution. local within user session.  Hive automatically deletes all temporary tables at the end of the Hive session .

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

alter table assign1\_intern\_tab rename to assign1\_loc.assign1\_intern\_tab;

31- check the table directory and list its components

Managed table still include data and dir of table changed but the external table have a problem.

**Assignment 2**

1- Create a database named assign2

create database assign2;

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

CREATE table songs (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) partitioned by (year\_created STRING,artist STRING )

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

stored as textfile

TBLPROPERTIES("skip.header.line.count"="1");

3- Load data into table HDFS directory using put command

load data local inpath 'songs.csv' into table songs partition(year\_created = '2022', artist= 'nada');

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

select \* from assign2.songs;

data found as I load it.

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

alter table songs add partition (year\_created = '2021' , artist = 'nada')

location '/user/static/2021/the\_one';

6- Load data to the created partitions

load data local inpath 'songs.csv' into table songs partition(year\_created = '2021', artist ='nada' );

7- List the partition directories to check for presence of files

Show partitions songs;

8- Create a staging table to host songs data

CREATE table assign2.staging (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'

stored as textfile

TBLPROPERTIES("skip.header.line.count"="1");

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

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

insert overwrite table songs partition (year\_created = '2021', artist ='nada' ) select \* from staging;

10- Truncate songs table and ensures no data in the table

truncate table songs;

select \* from songs;

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

set hive.exec.dynamic.partition.mode=nonstrict;

drop table songs;

CREATE table songs (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 serde 'org.apache.hadoop.hive.serde2.OpenCSVSerde'

stored as textfile

TBLPROPERTIES("skip.header.line.count"="1");

set hive.exec.dynamic.partition.mode=nonstrict;

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

select artist\_id ,artist\_latitude ,artist\_location ,artist\_longitude ,duration ,num\_songs ,song\_id ,title,year,artist\_name from staging

12-Truncate

Truncate table songs;

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

from assign2.staging

insert overwrite table assign2.songs partition(year = '2005',artist\_name)

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

where year='2005' ;

14- 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 same\_stage LIKE staging

stored as AVRO;

-15 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 same\_stage\_PRQ LIKE staging

stored as PARQUET;

16- 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

CREATE table events(artist STRING,auth STRING,firstName STRING,gender

STRING,itemInSession STRING,lastName STRING, length FLOAT,level STRING,location STRING,method STRING,page STRING,registration FLOAT,sessionId INT,song STRING,status INT,ts FLOAT,userAgent STRING,userId INT)

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

stored as textfile

TBLPROPERTIES("skip.header.line.count"="1");

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

Load data local inpath 'events.csv' into table events;

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) AS f\_song, last\_value(song) OVER(PARTITION BY sessionId ORDER BY ts) AS l\_song

FROM events;

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 x.userId,Rank() Over(order by x.cn\_song desc) as rank

from

(select userId,count (distinct song) as cn\_song

from events

group by userId

) x

order by rank desc;

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 x.userId,ROW\_NUMBER() Over(order by x.cn\_song desc) as rank

from

(select userId,count (distinct song) as cn\_song

from events

where page ='NextSong'

group by userId

) x

order by rank desc;

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

select location,artist,count(page) as cn\_song from events

where page ='NextSong'

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 location,artist,count(page) as cn\_song from events

where page ='NextSong'

group by location,artist

grouping sets ((location,artist),location);

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 from events 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 events CLUSTER BY userId,song,ts;