**Assignment 1**

- create a database named assign1

**create database assign1\_menna;**

- What is the database path on HDFS?

**describe database assign1\_menna;**

**hdfs://namenode:8020/user/hive/warehouse/assign1\_menna.db**

**/user/hive/warehouse/assign1\_menna.db**

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

**create database assign1\_loc location '/hp\_db/assign1\_loc' ;**

- 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\_menna;**

**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 ',';**

- What is the table path in HDFS?

**describe formatted assign1\_intern\_tab;**

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

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

First:

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

second:

**hadoop fs -mkdir /emp\_dir**

**hadoop fs -put employee.csv /emp\_dir**

**load data inpath '/emp\_dir/employee.csv' into table assign1\_intern\_tab;**

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

**select \* from assign1\_menna.assign1\_intern\_tab limit 10;**

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

**create external table assign1\_loc.assign1\_ext\_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 ','**

**location 'hdfs://namenode:8020/emp\_dir';**

- What is the table path in HDFS?

**hdfs://namenode:8020/emp\_dir**

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

**hadoop fs -put employee.csv /emp\_dir;**

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

**In internal: doesn’t present**

**In external: present**

- recreate both tables

**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 ',';**

**create external table assign1\_loc.assign1\_ext\_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 ','**

**location 'hdfs://namenode:8020/emp\_dir';**

- list both table directories

**hdfs://namenode:8020/emp\_dir**

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

- create internal table 'staging' inside the assign1 database

**create table if not exists assign1\_menna.staging**

**(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 the staging table with the data from file employees

**load data local inpath 'employee.csv' into table assign1\_menna.staging;**

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

**INSERT INTO assign1\_menna.assign1\_intern\_tab SELECT \* FROM assign1\_menna.staging;**

**INSERT INTO assign1\_loc.assign1\_ext\_tab SELECT \* FROM assign1\_menna.staging;**

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

**describe formatted assign1\_menna.assign1\_intern\_tab;**

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

**describe formatted assign1\_loc.assign1\_ext\_tab;**

**hdfs://namenode:8020/emp\_dir**

**Yes, there’s data**

- count the lines inside the file songs

**wc –l songs.csv**

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

**use assign1\_menna;**

**create table if not exists songs**

**(artist\_id string, artist\_latitude float , artist\_location string, artist\_longitude float, artist\_name string, duration float , num\_songs int, song\_id string, title string, year date)**

**ROW FORMAT DELIMITED**

**FIELDS TERMINATED BY ',';**

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

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

**select \* from songs limit 10;**

- count the number of records

**select count(\*) from songs;**

- 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, similar, 80 row**

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

**!hadoop -mkdir /songs**

**create external table if not exists songs1**

**(artist\_id string, artist\_latitude float , artist\_location string, artist\_longitude float, artist\_name string, duration float , num\_songs int, song\_id string, title string, year date)**

**ROW FORMAT DELIMITED**

**FIELDS TERMINATED BY ','**

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

- load the table using put command

**!hdfs dfs -put songs.csv /songs;**

- is the data readable through the table? Why?

**No**

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

**!hive -e ' select \* from songs1 limit 5';**

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

**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;**

- execute it from shell without accessing hive CLI /beeline

**hive -f test.hql**

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

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

**use assign1\_menna;**

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

- check the table directory and list its components

**Describe formatted assign1\_intern\_tab;**

**# col\_name data\_type comment**

**#Detailed Table Information**

**Database, Owner, CreateTime, LastAccessTime, Retention, Location,Table Type, Table Parameters**

**# Storage Information**

**SerDe Library, InputFormat, OutputFormat,Compressed ,Num Buckets ,Bucket Columns ,Sort Columns, Storage Desc Params**

**Assignment 2**

- Create a database named assign2

**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

**!hadoop -mkdir /songs**

**create external table if not exists songs2**

**(artist\_id string, artist\_latitude float , artist\_location string, artist\_longitude float, artist\_name string, duration float , num\_songs int, song\_id string, title string, year date)**

**Partitioned by (artist string, year\_ string)**

**ROW FORMAT DELIMITED**

**FIELDS TERMINATED BY ','**

**STORED AS TEXTFILE**

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

- Load data into table HDFS directory using put command

**!hdfs dfs -put songs.csv /songs;**

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

**No, because I didn’t give the partition the location**

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

**!hdfs dfs -mkdir -p /songs/artist/year;**

**alter table songs2**

**add partition(artist='Marc Shaiman' , year\_ ='2003')**

**location '/songs/artist/year';**

- Load data to the created partitions

**!hdfs dfs -put songs.csv /songs/artist/year;**

- List the partition directories to check for presence of files

**!hadoop fs -ls /songs/artist/year;**

- 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;

- 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

;

- Truncate songs table and ensures no data in the table

truncate table songs\_extern2;

Select \* from songs\_extern2;

- 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

;

- Truncate

Truncate table songs\_EXTERN2;

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

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'

;

- 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 new like staging;

- 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

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

avro-tools-1.9.1.jar getschema menna.avro

**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 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, last\_value(song)over(partition by sessionId order by itemInSession ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED following), first\_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. userId,song, ts; 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;