**Assignment 1**

- create a database named assign1

Create database assign1\_youssef;

- What is the database path on HDFS? /user/hive/warehouse

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

create database assign1\_loc\_youssef  
location ‘/hp\_db/assign1\_loc\_youssef’;

- 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\_youssef.assign1\_intern\_tab

(

id smallint,

name string,

age tinyint,

city string

)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n'

STORED AS TEXTFILE;

- What is the table path in HDFS? /user/hive/warehouse/assign1\_youssef.db/assign1\_intern\_tab

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

1- load data local inpath '/employee/employee.csv' into table assign1\_youssef.assign1\_intern\_tab;

2- hdfs dfs -copyFromLocal /employee/employee.csv /user/hive/warehouse/assign1\_youssef.db/assign1\_intern\_tab

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

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

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

CREATE EXTERNAL TABLE assign1\_loc\_youssef.assign1\_extern\_tab

(

id smallint,

name string,

age tinyint,

city string

)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n'

STORED AS TEXTFILE

LOCATION '/Youssef/employee';

- What is the table path in HDFS? hdfs://namenode:8020/Youssef/employee

- move the data from local filesystem to the directory created in step 7 hdfs dfs -put /employee/employee.csv /Youssef/employee

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

In case of managed table the whole table directory is deleted

In case of external table the directory and data file are both present

- recreate both tables

- list both table directories DESCRIBE FORMATTED <table name>

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

hdfs://namenode:8020/Youssef/employee

- create internal table 'staging' inside the assign1 database

CREATE TABLE assign1\_youssef.staging

(

id smallint,

name string,

age tinyint,

city string

)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n'

STORED AS TEXTFILE;

- load the staging table with the data from file employees

load data local inpath '/employee/employee.csv' into table assign1\_youssef.staging

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

FROM assign1\_youssef.staging

INSERT INTO assign1\_youssef.assign1\_intern\_tab SELECT id, name, age, city

INSERT INTO assign1\_loc\_youssef.assign1\_extern\_tab SELECT id, name, age, city;

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

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

!hdfs dfs -ls hdfs://namenode:8020/Youssef/employee;

- count the lines inside the file songs 80

Wc -l songs.csv

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

CREATE TABLE songs (

artist\_id varchar(18),

artist\_lat decimal(7,5),

artist\_loc varchar(80),

artist\_lon decimal(7,5),

artist\_name string,

duration decimal(9,5),

num\_songs tinyint,

song\_id varchar(18),

title string,

year smallint)

ROW FORMAT SERDE

'org.apache.hadoop.hive.serde2.OpenCSVSerde'

WITH SERDEPROPERTIES (

"separatorChar" = ","

)

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

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

SELECT \* FROM songs LIMIT 10;

- count the number of records 79

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

File counted 80 lines (header included), hive counted 79 records (header excluded). Data quality is fine after using the correct serde format

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

CREATE EXTERNAL TABLE assign1\_loc\_youssef.ext\_table

(

id smallint,

name string,

age tinyint,

city string

)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n'

STORED AS TEXTFILE

LOCATION '/Youssef/temp';

- load the table using put command

!hdfs dfs -put /employee/employee.csv /Youssef/temp;

- is the data readable through the table? Why?

Yes, because the data file is available under the location where the external table is pointing to.

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

hive -e 'SELECT \* FROM assign1\_youssef.staging;'

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

Graphical user interface, text

Description automatically generated

- execute it from shell without accessing hive CLI /beeline

Hive -f staging\_table.hql

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

What is a hive temp table?

It is used to store the data temporarily within an active session and the temporary tables get automatically removed when the active session ends.

Temporary tables don’t store data in the Hive warehouse directory instead the data get stored in the user’s scratch directory /tmp/hive/<user>/\* on HDFS.

How can you create it?

By using CREATE TEMPORARY TABLE you can create a temporary table in Hive, The TEMPORARY clause is optional, not using it will not return an error instead it creates a regular managed table.

Why would someone use a temp table?

They are created at the run time to store the intermediate data that are used to perform further data processing.

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

CREATE DATABASE new\_db;

ALTER TABLE assign1\_youssef.assign1\_intern\_tab rename to new\_db.assign1\_intern\_tab;

- check the table directory and list its components

hdfs dfs -ls /user/hive/warehouse/new\_db.db

**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

CREATE TABLE songs\_part (

artist\_id VARCHAR(18),

artist\_latitude DECIMAL(7,5),

artist\_loc VARCHAR(80),

artist\_longitude DECIMAL(7,5),

duration DECIMAL(9,5),

num\_songs TINYINT,

song\_id VARCHAR(18),

title STRING)

PARTITIONED BY(year SMALLINT, artist\_name STRING)

ROW FORMAT SERDE

'org.apache.hadoop.hive.serde2.OpenCSVSerde'

WITH SERDEPROPERTIES (

"separatorChar" = ","

)

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

- Load data into table HDFS directory using put command

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

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

No, because the data inside the directory isn’t yet partitioned by artist and year.

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

ALTER TABLE songs\_part ADD PARTITION(artist='amr diab', year='1999')  
LOCATION '/Youssef/songs/amr diab/1999';

- Load data to the created partitions

LOAD DATA LOCAL INPATH '/employee/songs.csv' INTO TABLE songs\_part  
PARTITION (artist='amr diab', year='1999');

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

!hdfs dfs -ls /Youssef/songs/amr\*/1999;

- Create a staging table to host songs data

Table assign1\_youssef.songs already exists

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

FROM assign1\_youssef.songs src  
INSERT OVERWRITE TABLE songs\_part PARTITION (year, artist\_name)  
SELECT src.artist\_id, src.artist\_lat, src.artist\_loc, src.artist\_lon, src.duration, src.num\_songs, src.song\_id, src.title, src.year, src.artist\_name;

- Truncate songs table and ensures no data in the table

truncate table songs\_part;

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

FROM assign1\_youssef.songs src  
INSERT OVERWRITE TABLE songs\_part PARTITION (year, artist\_name)  
SELECT src.artist\_id, src.artist\_lat, src.artist\_loc, src.artist\_lon, src.duration, src.num\_songs, src.song\_id, src.title, src.year, src.artist\_name;

- Truncate

truncate table songs\_part;

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

FROM assign1\_youssef.songs  
INSERT INTO songs\_part partition(year='1987', artist\_name)  
SELECT artist\_id, artist\_lat, artist\_loc, artist\_lon, duration, num\_songs, song\_id, title, artist\_name  
WHERE year='1987';

- 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

Use assign1\_youssef;  
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.

apt-get update

apt-get install wget

wget <http://archive.apache.org/dist/avro/avro-1.8.2/java/avro-tools-1.8.2.jar>

java -jar /avro-tools-1.8.2.jar fromtext songs.csv songs-avro

java -jar /avro-tools-1.8.2.jar getschema songs-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 events(

artist STRING, auth STRING, firstName String, gender STRING, itemInSession SMALLINT, lastName String, length DECIMAL(9,5), level STRING, location STRING, method STRING, page STRING, registration BIGINT, sessionId SMALLINT, song STRING, status SMALLINT, ts STRING, userAgent STRING, userId SMALLINT

)

ROW FORMAT SERDE  
'org.apache.hadoop.hive.serde2.OpenCSVSerde'  
WITH SERDEPROPERTIES (  
"separatorChar" = ","  
)  
TBLPROPERTIES("skip.header.line.count"="1");

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

!hdfs dfs -put /employee/events.csv /user/hive/warehouse/events;

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

SELECT userId, sessionId, first\_song, last\_song FROM

(SELECT

userId, sessionId, first\_value(song, TRUE) OVER(PARTITION BY sessionId ORDER BY ts) first\_song,  
last\_value(song, TRUE) OVER(PARTITION BY sessionId ORDER BY ts ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) last\_song, ROW\_NUMBER() OVER(PARTITION BY sessionId ORDER BY ts DESC) rn

FROM events  
ORDER BY sessionId) sq

WHERE rn = 1;

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, distinct\_songs, RANK() OVER(ORDER BY distinct\_songs) rnk

FROM

(SELECT userId, COUNT(DISTINCT(song)) distinct\_songs  
FROM events  
GROUP BY userId) sq

ORDER BY rnk;

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, distinct\_songs, ROW\_NUMBER() OVER(ORDER BY distinct\_songs) rnk

FROM

(SELECT userId, COUNT(DISTINCT(song)) distinct\_songs  
FROM events  
WHERE page = 'NextSong'  
GROUP BY userId) sq

ORDER BY rnk;

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, cnt\_loc\_artist, cnt\_loc, cnt\_total

FROM

(SELECT location, artist,   
count(song) OVER(PARTITION BY location, artist) cnt\_loc\_artist,   
count(song) OVER(PARTITION BY location) cnt\_loc,   
count(song) OVER() cnt\_total, ROW\_NUMBER() OVER(PARTITION BY location, artist) rn

FROM events) sq

WHERE rn = 1;

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, cnt\_loc\_artist, cnt\_loc, cnt\_total

FROM

(SELECT location, artist,   
count(song) OVER(PARTITION BY location, artist) cnt\_loc\_artist,   
count(song) OVER(PARTITION BY location) cnt\_loc,   
count(song) OVER(PARTITION BY artist) cnt\_artist,   
count(song) OVER() cnt\_total, ROW\_NUMBER() OVER(PARTITION BY location, artist) rn

FROM events) sq

WHERE rn = 1;

????

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

SELECT userId, LAG(song, 1) OVER(PARTITION BY userId ORDER BY ts) prev\_song, LEAD(song, 1) OVER(PARTITION BY userId ORDER BY ts) next\_song

FROM events;

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

SELECT userId, song, ts

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

SORT BY userId, song, ts;