**Assignment 1**

**First :**

# hadoop fs -mkdir /salma01

to show my dir

# hdfs dfs -ls hdfs:/

Text

Description automatically generated

- create a database named assign1

# create database assign1\_salma

- What is the database path on HDFS?

# describe database assign1\_salma

A picture containing chart

Description automatically generated

\*\*

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

# hive> create database assign1\_loc location '/user/hive/warehouse/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

# create table assign1\_salma.assign1\_intern\_tab ( id int, name String, age int , jobtype String, storeid int, storelocation String, salary bigint , yrsofexp int )

# row format delimited

# fields terminated by ',';

- What is the table path in HDFS?

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

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

# hadoop fs -put employee.csv /salma01;

# 1)load data local inpath 'employee.csv' into table assign1\_salma.assign1\_intern\_tab;

# 2) load data local inpath 'hdfs://namenode:8020/salma01’ into table assign1\_salma.assign1\_intern\_tab;

# 3) hadoop fs -put employee.csv hdfs://namenode:8020/user/hive/warehouse/ assign1\_salma.db/ assign1\_intern\_tab

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

# select \* from assign1\_intern\_tab limit 10;

A picture containing text

Description automatically generated

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

# ! hadoop fs -mkdir /mydata

# > ;

# create external table assign1\_loc. assign1\_extern\_tab (id int, name String, age int , job string , store int , loc string , salary bigint, yrs int )

# > row format delimited

# > fields terminated by ','

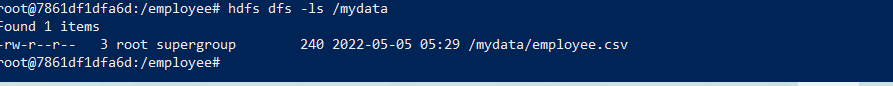
# > location 'hdfs://namenode:8020/mydata';

- What is the table path in HDFS?

# Location: hdfs://namenode:8020/mydata

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

# !hadoop fs -put employee.csv /mydata;



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

# drop table assign1\_salma.assign1\_intern\_tab;

# drop table assign1\_loc.assign1\_intern\_tab;

in managed table data removed from directory

in external table , DATA NOT REMOVED FROM DIRECTORY

- recreate both tables

# create table assign1\_salma.assign1\_intern\_tab ( id int, name String, age int , jobtype String, storeid int, storelocation String, salary bigint , yrsofexp int )

# row format delimited

# fields terminated by ',';

# create external table assign1\_loc. assign1\_extern\_tab (id int, name String, age int , job string , store int , loc string , salary bigint, yrs int )

# row format delimited

# fields terminated by ','

# location 'hdfs://namenode:8020/mydata';

list both table directories

internal :

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

external: hdfs://namenode:8020/mydata

- create internal table 'staging' inside the assign1 database

# create table assign1\_salma.staging(id int, name String, age int , job string , store int , loc string , salary bigint, yrs 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\_salma.staging;

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

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

INSERT INTO assign1\_loc.assign1\_extern\_tab SELECT \* FROM assign1\_salma.staging;

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

# describe formatted assign1\_salma.assign1\_intern\_tab;

# describe formatted assign1\_loc.assign1\_extern\_tab;

# 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.

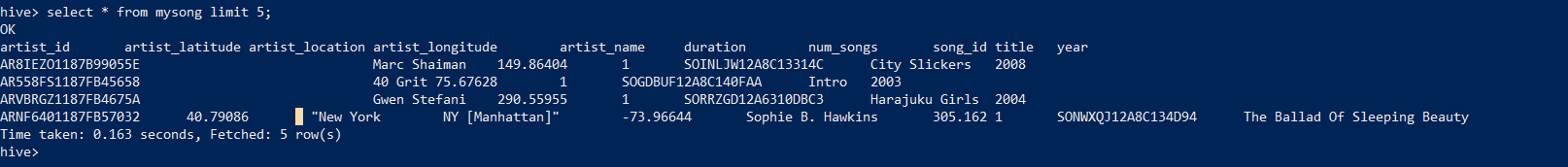
# create table mysong (id string,latitude string, loc string, long string , name string, duration string, num string , song\_id string, title string , year string) row format delimited

# > fields terminated by ',';

# load data local inpath 'songs.csv' into table mysong;

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

# Select \* from mysong limit 10;



- count the number of records

select count(\*) from songs;

# 80

- 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 to file count

80

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

# !hadoop fs -mkdir /externaldir;

# 

# create external table mysong1 (id string,latitude string, loc string, long string , name string, duration string, num string , song\_id string, title string , year string) row format delimited

# > fields terminated by ','

# > location 'hdfs://namenode:8020/externaldir';

- load the table using put command

# !hadoop fs -put songs.csv /externaldir;

- is the data readable through the table? Why?

No

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

# hive -S -e 'select \* from mysong1 limit 5';

Graphical user interface, application

Description automatically generated

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

# DROP TABLE IF EXISTS mytable;

# create table if not exists mytable

# (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 mytable;

- execute it from shell without accessing hive CLI /beeline

**hive -f script.hql**

**A picture containing graphical user interface

Description automatically generated**

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

# Hive temporary tables are similar to temporary tables that exist in SQL Server or any RDBMS databases, As the name suggests these tables are created temporarily within an active session.

# Create it :

# CREATE TEMPORARY TABLE tablename(

# id int,

# name string

# )

# ROW FORMAT DELIMITED

# FIELDS TERMINATED BY ',';

# Why??

# used to store the data temporarily within an active session and the temporary tables get automatically removed when the active session end

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

# use old\_database;

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

- 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\_salma;

- 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 assign2\_salma.songs( artist\_id String, artist\_latitude decimal,artist\_location String, artist\_longitude decimal ,artist\_name String,duration date , num\_songs int, song\_id String ,titile String , year bigint)

# partitioned by (artist string, year1 bigint)

# row format delimited

# fields terminated by ',';

- Load data into table HDFS directory using put command

# hive> !hadoop fs -put songs.csv hdfs:///salma01;

Text

Description automatically generated

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

Text

Description automatically generated

No data found because no load operation I did

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

# alter table assign2\_salma.songs add partition (artist ='salma', year1=2009) location 'hdfs:///salma01';

- Load data to the created partitions

# In previous step , give alter table location where data

Text

Description automatically generated

To show select from table as show data under partition

A picture containing text

Description automatically generated

- List the partition directories to check for presence of files

# show partitions assign2\_salma.songs;

- Create a staging table to host songs data

# create external table staging2 (id string,latitude decimal, loc string, long decimal , name string, duration date, num int , song\_id string, title string , year bigint)

# > row format delimited

# > fields terminated by ','

# > location 'hdfs://namenode:8020/salma01';

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

# insert overwrite table assign2\_salma.songs partition (artist,year1)

# select id,latitude,loc,long,name,duration,num,song\_id,title,year,name,year from staging2;

- Truncate songs table and ensures no data in the table

# truncate table assign2\_salma.songs;

Text

Description automatically generated

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

# from staging2

# insert overwrite table assign2\_salma.songs partition(artist='salma',year1=2008)

# select id,latitude,loc,long,name,duration,num,song\_id,title,year

# where name='salma' and year =2008

# insert overwrite table assign2\_salma.songs partition(artist='marc',year1=2006)

# select id,latitude,loc,long,name,duration,num,song\_id,title,year

# where name='marc' and year =2006

# ;

- Truncate

# truncate table assign2\_salma.songs;

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

1) drop table

2) then create again table but partition year first then artist because its parent directory

# create table assign2\_salma.songs( artist\_id String, artist\_latitude decimal,artist\_location String, artist\_longitude decimal ,duration date , num\_songs int, song\_id String ,titile String )

# partitioned by (year bigint, artist\_name string)

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

# stored as textfile

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

then

# from staging2

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

# select id,latitude,loc,long,name,duration,num,song\_id,title

# where year='2006' ;

- 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 avro\_new like staging2 ROW FORMAT 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

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

- download avro tools then

Text

Description automatically generated

avro-tools-1.9.1.jar getschema myschema.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 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 events;

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 events 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 events 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 events where page ='NextPage' 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 events 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 events 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

# select sessionId, userId,lead(song) over

# (partition by userId order by sessionId desc) from events

# order by sessionId desc;

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 event\_tab

# cluster by userId, song, ts;