Apache Hadoop: Hive

Core Concepts, Architecture, and Optimizations

Lab Exercise Workbook

HIVE Lab exercises

Exercise 1: Create Movies table (Managed)

Task 1: Create a table named “movies” with the following syntax

```
CREATE TABLE movies(movieid int , title string, genre int, year int) ROW FORMAT DELIMITED FIELDS TERMINATED BY ‘:’;
```

Task 2: Load the data in to the table

```
LOAD DATA LOCAL INPATH 'movies.dat' INTO TABLE movies;

//Check the data in HDFS
hdfs dfs  -ls /user/hive/warehouse/movies
hdfs dfs  -cat /user/hive/warehouse/movies/movies.dat
```

Task 3: Fire an SQL query to find out the count of movies released in 1950,genre contains Drama

```
$hive
Select count(*) FROM movies WHERE year = 1950;
Select * from movies Where genre like ‘%Drama%’;
Select * from movies Where genre like ‘Drama’;
Or
$beeline
$beeline>!connect jdbc:hive2://server:10000/default <username> <password>
Select count(*) FROM movies WHERE year = 1950;
Select * from movies Where genre like ‘%Drama%’;
```

HIVE Lab exercises

Exercise 2: Create Users table (Managed)

Task 1: Creating a hive table manually

```
$hive
hive> CREATE TABLE users(userid int , gender string,age int,occupation int, zipcode int)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ':';
```

Task 2: Load data in to a hive table

```
For hive shell
LOAD DATA local INPATH ‘users.dat’ OVERWRITE INTO TABLE users;
```

Exercise 3: Create UserRatings table (External)

Task 3: Creating userratings EXTERNAL TABLE

```
CREATE EXTERNAL TABLE userratings (userid int,movieid int,rating int,createtimestamp int) ROW FORMAT DELIMITED FIELDS TERMINATED BY ':' LOCATION '/user/bigdata/userratings'
```

Task 4: Load data in to a hive table

```
For hive shell
LOAD DATA local INPATH 'ratings.dat' OVERWRITE INTO TABLE
userratings;
```

HIVE Lab exercises

Exercise 4 - Joins

Task 1: Inner Join

Display the userid, movieid, movie name and the rating from userratings and movies tables.

```
SELECT userid,m.movieid,title,rating
FROM userratings u JOIN movies m ON (u.movieid = m.movieid) limit 10000;
```

Exercise 5 – Insert Overwrite

Task 1: Create a duplicate schema for users table

```
CREATE TABLE userslt3000 (userid int , gender string,age int,occupation int, zipcode int)
```

Task 2: Use insert overwrite statement to copy users data into usersle300

```
Insert overwrite table userslt3000 select * from users where userid<3000;
```

Task 3: Using CTAS

```
Create table usersgt3000 as select * from users where userid > 3000
```

HIVE Lab exercises

Exercise 6: Parquet format

Task 1: Create a hive table in text format(Staging table)

```
create table temps_txt (statecode string, countrycode string, sitenum string, paramcode string, poc string, latitude string, longitude string, datum string, param string, datelocal string, timelocal string, dategmt string, timegmt string, degrees double, uom string, mdl string, uncert string, qual string, method string, methodname string, state string, county string, dateoflastchange string) row format delimited fields terminated by ',';
```

Task 2: Load data in to hive table

```
hive>load data local inpath '/home/bigdata/training_materials/developer/data/weatherdata/hourly_TEMP_1990.csv '  into table temps_txt;
```

Task 3: Query the table

```
select avg(degrees) from temps_txt;
```

Task 4: Create a Parquet table

```
create table temps_par (statecode string, countrycode string, sitenum string, paramcode string, poc string, latitude string, longitude string, datum string, param string, datelocal string, timelocal string, dategmt string, timegmt string, degrees double, uom string, mdl string, uncert string, qual string, method string, methodname string, state string, county string, dateoflastchange string)
  STORED AS PARQUET;
```

HIVE Lab exercises

Task 5: Load data into the parquet table

```
insert overwrite table temps_par select * from temps_txt;
```

Task 6: Query the parquet table and observe the response time compared to the above query

```
select avg(degrees) from temps_par;
```