Apache Hadoop:Hive
Core Concepts, Architecture, and Optimizations

Lab Exercise-2 Workbook

HIVE Lab exercises

Lab Exercise - 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 (userratings.movieid =
movies.movieid) limit 100;
```

Lab Exercise– CTAS/Insert Overwrite

Task 1: Using CTAS

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

Task 2: Create a duplicate schema for users table

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

Task 3: Use insert overwrite statement to copy users data into userslt3000

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

HIVE Lab exercises

Lab Exercise: Parquet format

Task 1: Create a hive table Employee

```
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/ubuntu/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;
```

Lab Exercise - Partitioning

Task 1: Create a partitioned table

```
CREATE TABLE userratings_partition (movieid int,rating int,
createtimestamp int) partitioned by(userid int);
```

Task 2: Set the partition properties

```
Set hive.exec.dynamic.partition=true;
Set hive.exec.dynamic.partition.mode=nonstrict;
Set hive.exec.max.dynamic.partitions.pernode=6040;
```

Task 3: Load the data into the partitions

```
Insert overwrite table userratings_partition partition(userid) select
movieid,rating, createtimestamp,userid from userratings limit 10000;
```

Task 4: Querying partitions

```
SHOW PARTITIONS userratings_partition;
SELECT count(*)  FROM userratings_partition WHERE userid=77;
```

HIVE Lab exercises

Lab Exercise - Bucketing

Task 1: Create a bucketed table

```
SET hive.enforce.bucketing=true;

CREATE TABLE userratings_buckets (userid int,movieid int,rating
int,createtimestamp int) clustered by(userid) into 10 buckets;
```

Task 2: Load the data in to bucketed table

```
Insert overwrite table userratings_buckets select * from userratings
limit 50000;
```

Task 3: Query the table

```
Select * from userratings_buckets where userid=88;
SELECT * FROM userratings_buckets
TABLESAMPLE(BUCKET 1 OUT OF 10 ON userid);
```

Lab Exercise : JSON SerDe

Task 1: Create a table with JSON SerDe

```
CREATE EXTERNAL TABLE employees (
  name        STRING,
  salary      int,
  subordinates ARRAY<STRING>,
  deductions   MAP<STRING, int>,
  address      STRUCT<street:STRING, city:STRING, state:STRING, ip:INT>)
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
LOCATION '/user/bigdata/myserde';
```

HIVE Lab exercises

Task 2: Copy the data to hdfs

```
hive>load data local inpath
'/home/bigdata/training_materials/developer/exercises/Hive/emp.json
'  into table employees;
beeline>
hdfs dfs -put
/home/bigdata/training_materials/developer/exercises/Hive/emp.json
/user/hive
load data  inpath 'emp.json' into table employees;
```

Task 3: Query the data

```
select * from employees;
select subordinates[0] from employees;
```

Lab Exercise - UDFs

Task 1: Write a java implementation of a UDF as follows

```
import org.apache.hadoop.hive.ql.exec.Description;
 import org.apache.hadoop.hive.ql.exec.UDF;
 import org.apache.hadoop.io.Text;
 @Description( name="SayHello",    value="returns 'hello x', where x is
whatever you give it (STRING)",    extended="SELECT
simpleudfexample('world') from foo limit 1;" )
 public class SayHello extends UDF {
   public Text evaluate(Text input) {
     if(input == null) return null;
     return new Text("Hello " + input.toString());
   }
 }
```

Task 2: Compile the java program

HIVE Lab exercises

```
javac -classpath /opt/cloudera/parcels/CDH-6.2.1-1.cdh6.2.1.p0.1425774/jars/hive-exec-2.1.1-cdh6.2.1-core.jar:/opt/cloudera/parcels/CDH-6.2.1-1.cdh6.2.1.p0.1425774/jars/hadoop-common-3.0.0-cdh6.2.1.jar:/opt/cloudera/parcels/CDH-6.2.1-1.cdh6.2.1.p0.1425774/jars/hadoop-mapreduce-client-core-3.0.0-cdh6.2.1.jar SimpleUDFExample.java
```

Task 3: Create a jar file of the udf

```
jar cvf sayhello.jar SimpleUDFExample.class
```

Task 4: Add the jar to the hive classpath

```
add jar sayhello.jar;
```

Task 5: Create a new function

```
create  function sayhello as 'SimpleUDFExample';
```

Task 6: Run a query using the udf

```
select sayhello(title) from movies limit 10;
```

Lab Exercise – Hive transactions

```
set hive.txn.manager=org.apache.hadoop.hive.ql.lockmgr.DbTxnManager;
set hive.support.concurrency=true;
set hive.enforce.bucketing=true;
set hive.compactor.initiator.on=true;
set hive.compactor.worker.threads=2;
```

HIVE Lab exercises

Create table statement

```
create table hive_transactions(empid int, name string) clustered by
(empid) into 4 buckets stored
as ORC TBLPROPERTIES ('transactional'='true');
```

```
insert into hive_transactions values(1,"raju");
insert into hive_transactions values(2,"John");

update hive_transactions set j=Rakesh where empid=1
```