Lab 5: Hive Views and Joins

These labs assume you are using the class repository from the Nuvelab VM. You can also clone the repo to your HDP VM but that variation is not covered in the lab instructions. Commands used in this lab are listed in the accompanying “commands.txt” file.

Data Setup

1. Log into the Ambari file explorer
2. Make sure still have the /user/maria_dev/data you created in the first lab. If not, recreate it using the instructions in lab1.
3. Make sure the git repo is up to date but executing the command “git pull” in the Nuvelab VM.
4. Upload the SampleData.csv file into the data directory and make it fully writeable like you did for the CSV file in lab1.

Creating the Tables

The schema for the SampleData.csv data is the same as for the previous lab:

```
orderdate   string
region      string
rep         string
item        string
units       integer
unitcost    double
total       double
```

1. For this lab, we will just use the default database.
2. Create an internal table s1 and a second table exactly the same way called s2. Confirm you have both tables created by running “show tables’”

```
 create table s1(
       orderdate   string,
       region      string,
       rep         string,
       item        string,
       units       int,
       unitcost    double,
       total       double)
row format delimited fields terminated by ','
lines terminated by '\n' stored as textfile;
```

3. Skip the first row of the table s1

```
alter table s1 set tblproperties("skip.header.line.count"="1");
```

4. Load the data into the table s1

```
load data inpath'/user/maria_dev/data/SampleData.csv' into table s1;
```

5. Confirm that the data is in s1 but not in s2;

```
select * from s1;

select * from s2;
```

Creating Views

Recall that views are mappings into existing base tables.

1. Create a view called v from table s2 and confirm that v is empty.

```
create view v as select * from s2;

select * from v;
```

2. Insert data into s2 and notice that since the base table now has data, we can see it through the view

```
insert into s2 select * from s1;

select * from v;
```

3. Now drop the base table s2 and try to use the view. Explain what happened.

```
drop table s2;

select * from v;
```

4. Use the show command to list the views, then drop the view you just created

```
show views;

drop view v;
```

Creating a Custom View

1. A view can also be thought as a snapshot of a query. In the following command, we create a view which only has two of the columns of the original table, and renames those columns, then selects only the rows where the region is 'East'.

   ```
   create view east as select region as area, item as product from s1
               where region = 'East' ;
   ```

   ```
   select * from v;
   ```

2. Create a view from a view

   ```
   create view v as select * from s1;
   ```

   ```
   create view w as select region from v;
   ```

   ```
   select * from w;
   ```

3. What happens to w if you drop the view v? How would you explain the results?

Joins in Hive

Hive has the same types of joins as in SQL. This lab is the first of two on joins in Hive.

1. You will be using transaction data from a file with the following schema

   ```
   id           int,
   account_id   int,
   vendor_id    int,
   transtime    string,
   city         string,
   state        string,
   amount       double
   ```

2. And joining it to the vendor table:

   ```
   id           int,
   name         string,
   city         string,
   state        string,
   category     string,
   swipe_rate   double
   ```

3. Create the tables and load the data

   ```
   create external table trans1000 (
         id          int,
         account_id  int,
         vendor_id   int,
         transtime   string,
         city        string,
         state       string,
         amount      double)
   row format delimited fields terminated by ','  stored as textfile
   location '/user/maria_dev/data/trans1000';

   load data inpath'/user/maria_dev/data/trans_1000.csv' into table trans1000;

   create external table vendors (
         id          int,
         name        string,
         city        string,
         state       string ,
         category    string ,
         swipe_rate  double)
   row format delimited fields terminated by ','stored as textfile
   location '/user/maria_dev/data/vend';

   load data inpath'/user/maria_dev/data/vendors.csv' into table vendors;
   ```

4. There are four basic joins that are analogous to standard RDBMS joins.
   - Inner Join
   - Left Outer Join
   - Right outer Join
   - Outer Join

5. These are illustrated with the following queries. First we set the auto join feature to false to keep Hive from trying to optimize our joins

   ```
   set hive.auto.convert.join=false;
   ```

6. The problem that the logical join column, the vendor id, has the same ten values in both tables which means all joins are going to map to the same output. To create some more illustrative data to work with, we will create two views:

7. The first vend1, will have vendor ids only less that 5. And the second, trans1, will have only transactions where the vendor_id is greater than 2. Now the only common vendor ids between the two tables are 3 and 4.

```
create view vend1 as select * from vendors where id < 5;

create view trans1 as select * from trans1000 where vendor_id > 2;
```

8. You should confirm this is true by running select statements to get a list of vendor ids in each view. For the trans1 view, using ‘distinct’ will make the output easier to read.

9. Run the following joins and examine the output, especially the numbers of rows of output. Compare the number of rows returned to the number of rows in each view

```
select trans1.*,vend1.* from trans1 join vend1 on
                              (trans1.vendor_id = vend1.id);

select trans1.*,vend1.* from trans1 left outer join vend1 on
                              (trans1.vendor_id = vend1.id);

select trans1.*,vend1.* from trans1 right outer join vend1 on
                              (trans1.vendor_id = vend1.id);

select trans1.*,vend1.* from trans1 full outer join vend1 on
                              (trans1.vendor_id = vend1.id);
```