Lab 3: Hive Sorting

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.

Data Setup

1. Log into the Ambari file explorer
2. Create a new lab3 directory in /user/maria_dev that is fully writeable, just like you did in lab1.
3. Make sure still have the /user/maria_dev/data you created in the first lab. If not, recreate it using the instructions in lab1.
4. Make sure the git repo is up to date but executing the command “git pull” in the Nuvelab VM.
5. Upload the sales.csv file into the lab3 directory and make it fully writeable like you did for the CSV file in lab1.

Creating the Table

The schema for the table is the same as for the last 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 sales3

```
create table sales3
  (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. Load the table and use select * to confirm the data is there

```
load data inpath'/user/maria_dev/lab3/sales.csv'
into table sales3;
```

Order by

The “order by” clause uses one reducer in order to order the whole table. Notice that only one reducer is involved when you run the operation below. Why must only one reducer be used to get the output result?

1. Order the sales3 table by the units column. The first time you run this, it can take up to five minutes to execute. Subsequent runs take significantly less time.

```
select * from sales3 order by units;
```

2. Run the command ordering by the rep column and then order by the total column and confirm that the whole table is being ordered.

Sort by

The “sort by” maps the table onto multiple reducers. Each segment is ordered like the “order by” cause

1. Sort the table by the units column. Notice that the number of reducers being used is two. Also try this with the rep column

```
select * from sales3 sort by units;

select * from sales3 sort by rep;
```

2. Using the select * command, can you tell which rows were sorted by with reducer? Try sorting by different columns and verify the result

Distribute by

1. “Distribute by” maps rows using a column value so that all rows with the same value for that column are on the same reducer. This can be difficult to see.
2. Run the following commands and compare the output.

```
select item, units from sales3;

select item, units from sales3 distribute by item;

select item, units from sales3 distribute by units;
```

3. Notice that the ordering of the rows in each case is different depending on how the original rows were distributed.

Cluster by

1. This clause has the effect of performing a “distribute by” followed by a “sort by” on the table. The output will look very much like “sort by” but notice that in the “sort by,” the row values are assigned to the reducers arbitrary. That means the same column value can appear in both reducers’ sorted list. The “cluster by” ensures that each column value appears in only one reducer’s sorted list.
2. Compare the output of the commands in the “sort by” section above with these versions:

```
select * from sales3 cluster by units;

select * from sales3 cluster by rep;
```

Cluster by