Lab 10: Materialized Views

Materialized views store the results of a computation for later use. When using a normal view, changes to the underlying tables are reflected in changes to any computational queries that are displayed in the view. This can be computationally expensive.

Materialized views store the result of the computation when the view is created. When the view is accessed, the computation is not rerun, but rather the stored result is used. To refresh or recompute the view when the data changes, the alter command has to be used

In this lab, you will explore create a normal view “norm” and a materialized view “mat” that illustrate this.

1. Create the data

For this lab, you can reuse the two tables trans1000 and vendors from the previous labs. If you no longer have those tables, then refer to previous labs to recreate them.

Now create the table that will contain the changing data. The vendors table has ten entries with id values from 1 to 10. In the v1 table, only the values of 1 to 5 will be inserted.

```
create table v1 (

      id          int,
      name        string,
      city        string,
      state       string ,
      category    string ,
      swipe_rate  double)
 row format delimited fields terminated by ','
 lines terminated by '\n' stored as textfile;

 insert into v1 select * from vendors where id < 6;
```

2. Create the Views

Just as we did before:

```
create view norm as select trans1000.*,v1.*
      from trans1000,v1 where trans1000.vendor_id = v1.id;

create materialized view mat as select trans1000.*,v1.*
      from trans1000,v1  where trans1000.vendor_id = v1.id;
```

Use a select to see the range of vendor ids in both views

```
select count(*), vendor_id from mat group by(vendor_id);
select count(*), vendor_id from norm group by(vendor_id);
```

Both views should show exactly the same results.

3. Change the Data

Now change the data in table v1 which is one of the base tables for the views that you just created.

```
insert overwrite v1 select * from vendors where id > 5;
```

Use a select query to ensure that the data in v1 has changed.

Run a select of each of the two views. The view norm will show the updated data since it re-computed the query, however the mat view still shows the result of the original query.

```
select count(*), vendor_id from mat group by(vendor_id);
select count(*), vendor_id from norm group by(vendor_id)
```

4. Update the Materialized View

In order to get the materialized view to update, the alter command is run. Running the select command show that the view is now using the new data

```
alter materialized view mat rebuild;
select count(*), vendor_id from mat group by(vendor_id);
```