Visa

Big Data 101

Student Lab 3

Web Age Solutions Inc.
USA: 1-877-517-6540
Canada: 1-866-206-4644
Web: http://www.webagesolutions.com

For customizations of this book or other sales inquiries, please contact us at:
USA: 1-877-517-6540, email: getinfousa@webagesolutions.com
Canada: 1-866-206-4644 toll free, email: getinfo@webagesolutions.com

Contents

Lab 3 – Working with Views and Partitions - 1 1/2 Hours

Overview

HiveQL Views and Partitions

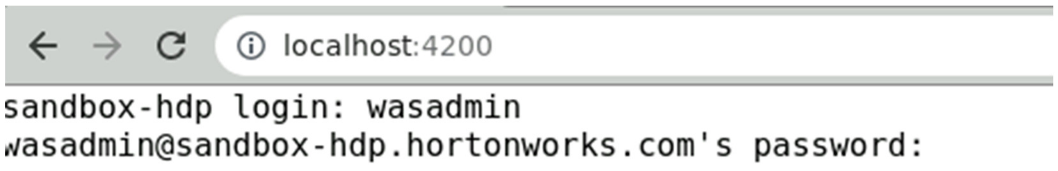
In this lab we will explore HiveQL and Hive CLI settings for more efficient results especially when working with tables.
We will continue to utilize the accounts, orders, vendors, and transactions tables. These 4 tables represent Credit Card data that has already been given a schema in the Hive Metastore.

Step 1. Login to the Lab environment

Please connect to the assigned Nuvepro Virtual Machine where the labs will be run from.
On the Linux desktop you will see the HDP VM has already been started and is showing a couple of ways to login to the system.

Open a web browser on the Nuvepro Virtual Machine. Enter this address into the URL to open the SSH connection to HortonWorks HDP Sandbox.

http://localhost:4200



```
sandbox-hdp login: wasadmin
wasadmin@sandbox-hdp.hortonworks.com's password:
```

Login to the console and provide the password when prompted.

Login username: **wasadmin**
Password: **wasadmin**

Change to the LabFiles folder.

cd LabFiles

Start the Hive CLI so we can run the HiveQL commands to investigate the table data.

hive

Now set the database to use with your HiveQL queries.

use wasadmin_db;

Step 2. Materialized View

Using a materialized view, the optimizer can compare old and new tables, rewrite queries to accelerate processing, and manage maintenance of the materialized view when data updates occur. The optimizer can use a materialized view to rewrite projections, filters, joins, and aggregations fully or partially.

Hive stores materialized views in the Hive warehouse. Hive performs view maintenance incrementally if possible, refreshing the view to reflect any data inserted into ACID tables. Hive does a full rebuild if an incremental one is impossible. More information on Altering a View: https://docs.cloudera.com/HDPDocuments/HDP3/HDP-3.1.0/materialized-view/content/hive_alter_materialized_view_rebuild.html

There are various commands that can be used when working with Materialized Views like Create, Alter, Describe, Show and Drop.

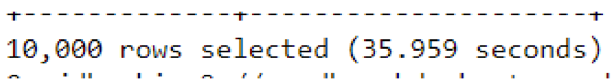
Create and use a materialized view

Use a materialized view of a query to calculate and store results of an expensive operation, such as join. Then query on the new view.

Let’s run a select summing the transactions amounts by account_id field.

select account_id, SUM(amount) as total from transactions group by account_id;

There are 10000 accounts, and this query must run through each row to calculate the total for each. This takes considerable time as you can see.



```
+-------------+--------------------+
10,000 rows selected (35.959 seconds)
```

To work with Materialized Views the table must support an ACID compliant format such as OCR. The Optimized Row Columnar (ORC) file format provides a highly efficient way to store Hive data. It was designed to overcome limitations of the other Hive file formats. Using ORC files improves performance when Hive is reading, writing, and processing data. Therefore, we need to create a table type that will support ACID operations by creating a transactions table with ORC.

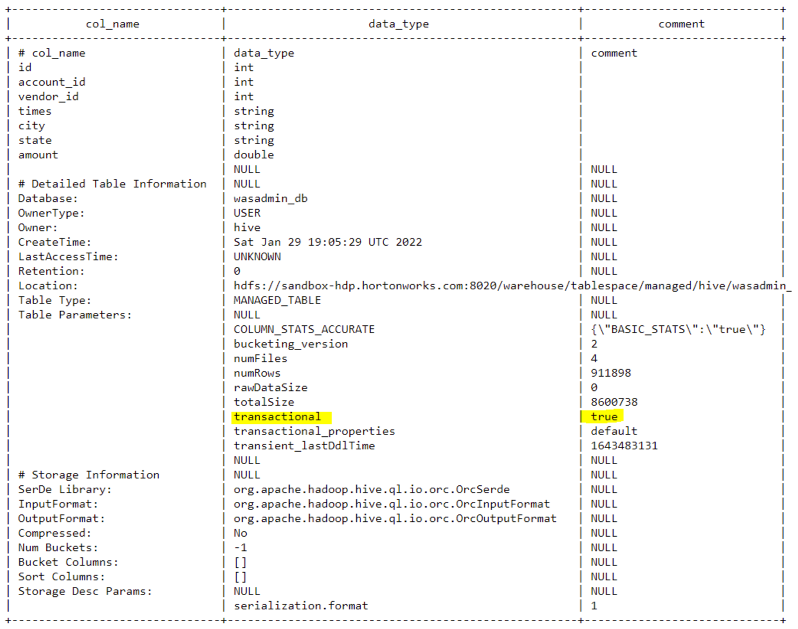
create table transactions_orc stored as orc as select * from transactions;

Check that you were able to create the new table.

show tables;

Does the new table support ACID transactions? Look for *true* value.

describe formatted transactions_orc;



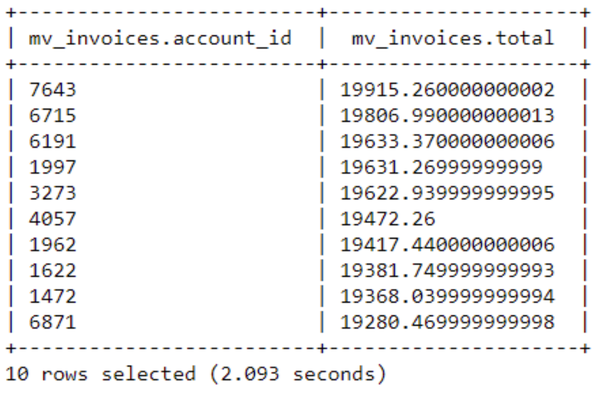
```
+--------------------------------+--------------------------------------------------+---------------------------------+
|            col_name            |                    data_type                     |             comment             |
+--------------------------------+--------------------------------------------------+---------------------------------+
| # col_name                     | data_type                                        | comment                         |
| id                             | int                                              |                                 |
| account_id                     | int                                              |                                 |
| vendor_id                      | int                                              |                                 |
| times                          | string                                           |                                 |
| city                           | string                                           |                                 |
| state                          | string                                           |                                 |
| amount                         | double                                           |                                 |
|                                | NULL                                             | NULL                            |
| # Detailed Table Information   | NULL                                             | NULL                            |
| Database:                      | wasadmin_db                                      | NULL                            |
| OwnerType:                     | USER                                             | NULL                            |
| Owner:                         | hive                                             | NULL                            |
| CreateTime:                    | Sat Jan 29 19:05:29 UTC 2022                     | NULL                            |
| LastAccessTime:                | UNKNOWN                                          | NULL                            |
| Retention:                     | 0                                                | NULL                            |
| Location:                      | hdfs://sandbox-hdp.hortonworks.com:8020/warehouse/tablespace/managed/hive/wasadmin_
| Table Type:                    | MANAGED_TABLE                                    | NULL                            |
| Table Parameters:              | NULL                                             | NULL                            |
|                                | COLUMN_STATS_ACCURATE                            | {\"BASIC_STATS\":\"true\"}      |
|                                | bucketing_version                                | 2                               |
|                                | numFiles                                         | 4                               |
|                                | numRows                                          | 911898                          |
|                                | rawDataSize                                      | 0                               |
|                                | totalSize                                        | 8600738                         |
|                                | transactional                                    | true                            |
|                                | transactional_properties                         | default                         |
|                                | transient_lastDdlTime                            | 1643483131                      |
|                                | NULL                                             | NULL                            |
| # Storage Information          | NULL                                             | NULL                            |
| SerDe Library:                 | org.apache.hadoop.hive.ql.io.orc.OrcSerde        | NULL                            |
| InputFormat:                   | org.apache.hadoop.hive.ql.io.orc.OrcInputFormat  | NULL                            |
| OutputFormat:                  | org.apache.hadoop.hive.ql.io.orc.OrcOutputFormat | NULL                            |
| Compressed:                    | No                                               | NULL                            |
| Num Buckets:                   | -1                                               | NULL                            |
| Bucket Columns:                | []                                               | NULL                            |
| Sort Columns:                  | []                                               | NULL                            |
| Storage Desc Params:           | NULL                                             | NULL                            |
|                                | serialization.format                             | 1                               |
+--------------------------------+--------------------------------------------------+---------------------------------+
```

Time to create our new mv_invoices Materialized View. Notice the select statement that allows us to build the mv_invoices view.

create materialized view mv_invoices as select account_id, sum(amount) as total from transactions_orc group by account_id order by total desc;

Let's query our new materialized view table

select * from mv_invoices limit 10;

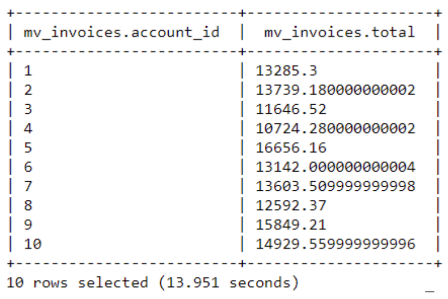


```
+-------------------------+---------------------+
| mv_invoices.account_id  |  mv_invoices.total  |
+-------------------------+---------------------+
| 7643                    | 19915.260000000002  |
| 6715                    | 19806.990000000013  |
| 6191                    | 19633.370000000006  |
| 1997                    | 19631.26999999999   |
| 3273                    | 19622.939999999995  |
| 4057                    | 19472.26            |
| 1962                    | 19417.440000000006  |
| 1622                    | 19381.749999999993  |
| 1472                    | 19368.039999999994  |
| 6871                    | 19280.469999999998  |
+-------------------------+---------------------+
10 rows selected (2.093 seconds)
```

Q: What does this output represent?
A: Total sales by account

Challenge 1: (Solutions are not shown, confer with others if stuck on a question)

Q: Can you create a result that looks like the following resultset?



```
+--------------------------+-----------------------+
| mv_invoices.account_id  |  mv_invoices.total   |
+--------------------------+-----------------------+
| 1                        | 13285.3               |
| 2                        | 13739.180000000002    |
| 3                        | 11646.52              |
| 4                        | 10724.280000000002    |
| 5                        | 16656.16              |
| 6                        | 13142.000000000004    |
| 7                        | 13603.509999999998    |
| 8                        | 12592.37              |
| 9                        | 15849.21              |
| 10                       | 14929.559999999996    |
+--------------------------+-----------------------+
10 rows selected (13.951 seconds)
```

Q: What about the ten largest account_id's and there totals?

Q: What other queries would make sense for Materialized Views? Identify some good candidates you may be asked!

Quit the hive shell when you are finished.

!q

Step 3. Hive Partitions

We can load data into Hive, let's create a staging folder. Remember data can be moved using hdfs or hadoop commands. For more information:

https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/FileSystemShell.html#cp

hdfs dfs -mkdir -p wasadmin/staging/

We will need to copy some data into our new HDFS staging folder. Run the hadoop file system command to copy the file in our newly created folder staging.

```
hadoop fs -cp /data/transactions/in/transaction-2015-01-01.csv
/user/wasadmin/wasadmin/staging/
```

```
hadoop fs -cp /data/transactions/in/transaction-2015-01-02.csv
/user/wasadmin/wasadmin/staging/
```

Check to see the file was loaded properly into the folder staging.

```
hadoop fs -ls -R wasadmin/staging/
```

You should see the file listed in the folder. If not check the command again that everything was spelled the same as shown above.

Let’s set permissions for our folders to allow writing. You may see a warning; you should be able to ignore that.

```
hadoop fs -chmod -R 777 /user/wasadmin/
```

Step 4: Create a partition table

Start the Hive CLI Shell.

```
hive
```

Now set the database to wasadmin_db properly.

```
use wasadmin_db;
```

Use the show tables command to see your current listing.

```
show tables;
```

Time to create a table.

CREATE EXTERNAL TABLE transactions_p (id STRING, `timestamp` TIMESTAMP, mti STRING, card_number STRING, amount_customer DECIMAL (10,2), merchant_type STRING, merchant_id STRING, merchant_address STRING, ref_id STRING, amount_merchant DECIMAL (10,2), response_code STRING) PARTITIONED BY (dt STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' stored as textfile LOCATION '/user/wasadmin/wasadmin/transactions/in-part/' ;

Review that the table was successfully created by describing the table structure.

DESC transactions_p;

See if any data is residing in the new table structure you created.

select * from transactions_p limit 10;

Should return no rows

Partitions can be viewed like tables by using the show partitions command and passing the table name.

show partitions transactions_p;

Step 5: Load data into partitions

load data inpath '/user/wasadmin/wasadmin/staging/transaction-2015-01-01.csv' INTO TABLE transactions_p partition (dt='2015-01-01');

load data inpath '/user/wasadmin/wasadmin/staging/transaction-2015-01-02.csv' INTO TABLE transactions_p partition (dt='2015-01-02');

Note : We have to specify the partition.

Notice the files in staging directory have been moved.

Step 6: Using partitions

Let's run some queries using partitions

Run the show partitions command to see the partition information.

show partitions transactions_p;

Now count all data in table, there should be 111888 total records.

select count(*) from transactions_p;

You could have also look at each partition specifically.

select count(*) from transactions_p where dt='2015-01-01';

select count(*) from transactions_p where dt='2015-01-02';

Note the count results. 2015-0-01 should have 11888 and 2015-01-02 should contain 100000 total records.

Step 7: Run a histogram on amount

histogram_numeric(): Estimating frequency distributions
Histograms represent frequency distributions from empirical data. The kind that is referred to here are histograms with variable-sized bins. Specifically, this UDAF will return a list of (x,y) pairs that represent histogram bin centers and heights.

https://cwiki.apache.org/confluence/display/hive/statisticsanddatamining#StatisticsAndDataMining-histogram_numeric():Estimatingfrequencydistributions
Suppose we have a dataset such as

```
id   |   num
---------------
1        2.0
1        4.0
2        5.0
1        7.0
1        8.0
2        8.0
1        3.0
1        5.0
1        6.0
3        7.0
```

If you perform the query 'select histogram_numeric(num, 3) from table' this will produce a histogram grouped into 3 bins represented as an array of structs.

```
[{'x':2.5, 'y:2.0'}, {'x':5.0, 'y':4.0}, {'x':7.5, 'y':4.0}]
```

Hive has a function to transform this into table form for ease of use, say plotting in Excel or Tableau. Use the inline function like this:

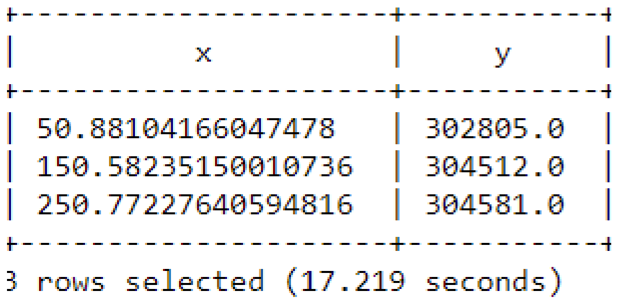
```
select inline(histogram_numeric(num, 3)) from table;
```

This would return

```
x   |   y
-------------
2.5     2.0
5.0     4.0
7.5     4.0
```

Let’s try a Hive inline histogram on the transaction amounts.

select inline(histogram_numeric(amount, 3)) from transactions;



```
+---------------------+-----------+
|          x          |     y     |
+---------------------+-----------+
| 50.88104166047478   | 302805.0  |
| 150.58235150010736  | 304512.0  |
| 250.77227640594816  | 304581.0  |
+---------------------+-----------+
3 rows selected (17.219 seconds)
```

Q: What does the above results mean?

Q: How can this be used?
A: In visualizations

Step 8: Perform a histogram on account totals

Find a histogram on account totals. Note that to do this, you must first generate a table in a subquery to calculate account totals, then perform the histogram as in the previous step 7.

select accounts.id, sum(transactions.amount) from transactions join accounts on transactions.account_id = accounts.id group by accounts.id;

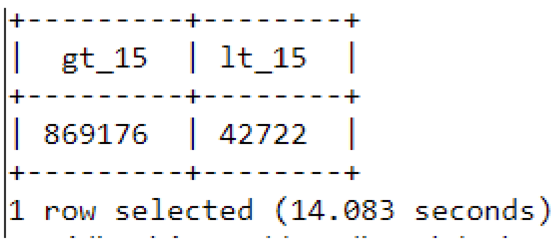
Q: What does the above results for 1000 rows mean?
A: Totals for each account_id.

Step 9: Do a binary binning exercise

Sometimes we want to do a binary binning exercise, for example, count transactions above or below a certain amount by customer. This is a case of fixed-width bins.

Unfortunately, using the histogram_numeric() function becomes difficult in this case. Like SQL Hive supports the CASE function, so we can easily do the following.

select count(CASE WHEN amount > 15.0 THEN amount END) AS gt_15, count(CASE WHEN amount <= 15.0 then amount END) as lt_15 from transactions;



```
+---------+--------+
|  gt_15  | lt_15  |
+---------+--------+
| 869176  | 42722  |
+---------+--------+
1 row selected (14.083 seconds)
```

Q: How could we figure out the same information PER vendor category?
HINT: Use a group by -- remember that vendor category is in the vendors table and not the transactions table and so requires a join.

Remember to exit hive when you have finished.

!q

Review

We have seen that HiveQL Views and Partitions syntax are very similar to SQL but have the capacity to work with very large table data.
In the next Lab you will incorporate all that you have learned for the labs to solve some issues in a Mini-Project.