Hive Homework

In the homework that follows, get a sheet of paper or open a file editor and, as you work, write down your aswers to the **questions in red**. Turn in a text file with your answers.

For latest cloudera distribution, the data ingest has changed, though this will still work with changes.

Data Ingest

Importing Database Tables into HDFS with Sqoop

Homework data is stored in a MySQL database. In the next few steps, you will use Sqoop to import its tables into HDFS and then use Hive to create tables from those files using information from AVRO.

Step 1: Run the Sqoop command.

This imports tables from mysql as files in HDFS. This sqoop command uses AVRO to describe and import the tables. This command may take a while to complete as it launches MapReduce jobs to export the data from the MySQL database and put those file in Avro format in HDFS. It is also creating the Avro schemas, so that we can create our Hive tables later.

```
$ sqoop import-all-tables \
  -m 1 \
  --connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
  --username=retail_dba \
  --password=cloudera \
  --compression-codec=snappy \
  --as-avrodatafile \
  --warehouse-dir=/user/hive/warehouse
```

Verify that it worked:

When the command completes, confirm that your Avro data files exist in HDFS by typing:

$ hadoop fs -ls /user/hive./warehouse

This should show a folder for each table. Check that the directories contain data files by typing:

$ hadoop fs -ls /user/hive/warehouse/categories/

Finally, list the avro schema files:

```
$ ls -l *.avsc
```

This should show 6 .avsc files for the six tables that in retail_db.

Step 2. Move schema files into HDF

Hive will need the schema files, so let's copy them into HDFS.

```
$ sudo -u hdfs hadoop fs -mkdir /user/examples
$ sudo -u hdfs hadoop fs -chmod +rw /user/examples
$ hadoop fs -copyFromLocal ~/*.avsc /user/examples
```

Step 3: Create tables in Hive

Start Hive by typing:

```
$ hive
```

Copy and paste the 5 "create table" commands below. After each, hit enter.

```
CREATE EXTERNAL TABLE categories
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.avro.AvroSerDe'
STORED AS INPUTFORMAT 'org.apache.hadoop.hive.ql.io.avro.AvroContainerInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.avro.AvroContainerOutputFormat'
LOCATION 'hdfs:///user/hive/warehouse/categories'
TBLPROPERTIES ('avro.schema.url'='hdfs:///user/examples/categories.avsc');

CREATE EXTERNAL TABLE customers
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.avro.AvroSerDe'
STORED AS INPUTFORMAT 'org.apache.hadoop.hive.ql.io.avro.AvroContainerInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.avro.AvroContainerOutputFormat'
LOCATION 'hdfs:///user/hive/warehouse/customers'
TBLPROPERTIES ('avro.schema.url'='hdfs:///user/examples/customers.avsc');

CREATE EXTERNAL TABLE departments
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.avro.AvroSerDe'
STORED AS INPUTFORMAT 'org.apache.hadoop.hive.ql.io.avro.AvroContainerInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.avro.AvroContainerOutputFormat'
LOCATION 'hdfs:///user/hive/warehouse/departments'
TBLPROPERTIES ('avro.schema.url'='hdfs:///user/examples/departments.avsc');
```

```
CREATE EXTERNAL TABLE orders
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.avro.AvroSerDe'
STORED AS INPUTFORMAT 'org.apache.hadoop.hive.ql.io.avro.AvroContainerInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.avro.AvroContainerOutputFormat'
LOCATION 'hdfs:///user/hive/warehouse/orders'
TBLPROPERTIES ('avro.schema.url'='hdfs:///user/examples/orders.avsc');

CREATE EXTERNAL TABLE order_items
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.avro.AvroSerDe'
STORED AS INPUTFORMAT 'org.apache.hadoop.hive.ql.io.avro.AvroContainerInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.avro.AvroContainerOutputFormat'
LOCATION 'hdfs:///user/hive/warehouse/order_items'
TBLPROPERTIES ('avro.schema.url'='hdfs:///user/examples/order_items.avsc');

CREATE EXTERNAL TABLE products
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.avro.AvroSerDe'
STORED AS INPUTFORMAT 'org.apache.hadoop.hive.ql.io.avro.AvroContainerInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.avro.AvroContainerOutputFormat'
LOCATION 'hdfs:///user/hive/warehouse/products'
TBLPROPERTIES ('avro.schema.url'='hdfs:///user/examples/products.avsc');
```

Verification

Now you should have the following tables:

```
hive> show tables
OK
categories
customers
departments
order_items
orders
products

hive> exit;
```

This is the end of the data ingest.

Part I: Develop and Run Simple Queries

Step #1: Running a Query from the Hive Shell

Image that your company (Dualcore) ran a contest in which customers posted videos of interesting ways to use their new tablets. A $5,000 prize will be awarded to the customer whose video received the highest rating.

However, the registration data was lost due to an RDBMS crash, and the only information they have is from the videos. The winning customer introduced herself only as “Brian from Chicago” in his video.

You will need to run a Hive query that identifies the possible winner’s record in the customer database so that Dualcore can send him the $5,000 prize.

1. Start Hive:

```
$ hive
```

> **Hive Prompt**
>
> To make it easier to copy queries and paste them into your terminal window, we do not show the hive> prompt in subsequent steps. Steps prefixed with $ should be executed on the UNIX command line; the rest should be run in Hive unless otherwise noted.

2. Make the query results easier to read by setting the property that will make Hive show column headers:

```
hive> set hive.cli.print.header=true;
```

3. Solve this problem:

All you know about the winner is that the first name is Brian and he lives in Chicago. Use Hive's LIKE operator to do a wildcard search for names such as "Bryan” and “Brian”. Remember to filter on the customer's city.

Question: Which customers did your query identify as the winner of the $5,000 prize?

Step #2: Running a Query Directly from the Command Line

It seems that some products are coming up with very low prices when customers come to the register. There are reports of at least 2 different products with a price of $0. You will now run a query to identify the products that currently have the lowest prices.

1. Exit the Hive shell and return to the command line by typing `quit;`

2. Although HiveQL statements are terminated by semicolons in the Hive shell, it is not necessary to do this when running a single query from the command line using the -e option. Run the following command to execute the quoted HiveQL statement:

```
$ hive -e 'SELECT product_id, product_price, product_name
FROM PRODUCTS ORDER BY product_price LIMIT 10'
```

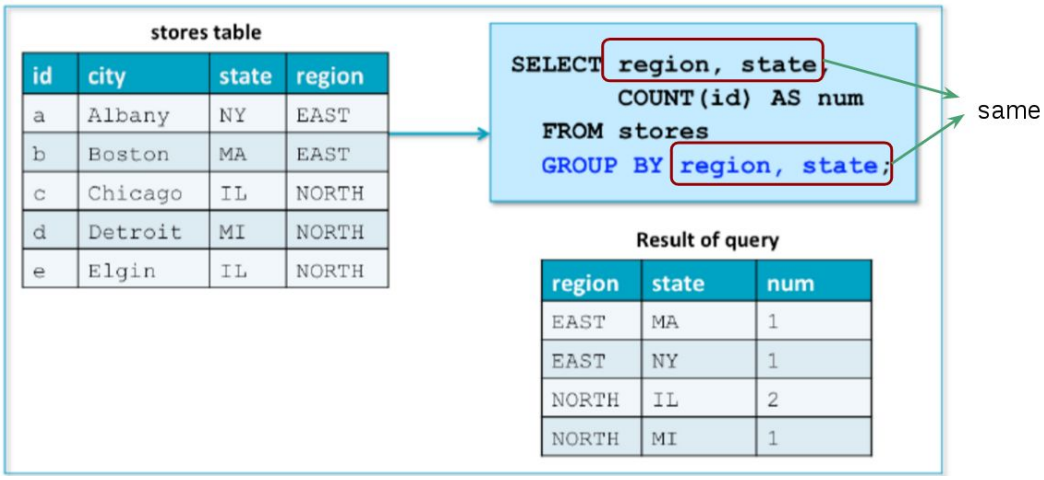
Question: Which two product names have a price of zero?

3. Restart Hive and answer the following questions:

Question: How many customers ids are in the customers table?

Question: How may households are in the customers table? (Hint, try concatenating the customers' address and zipcode and count the number of distinct households)

Question: Using customer_id, which state has the most customers? Hint: notice, you can group by one field and then order by another. To refresh your memory on "group by":



Calculating Revenue and Profit

Several more questions are described below and you will need to write a query to answer them. Use the hive shell.

- **Question:** Which top three product_ids had the most orders? Show your query.
- **Extra credit:** What were the product names? Show your query.

Problem: The programmers loading date information made a mistake, and all of the dates in the orders table have extra zeros. The correct date should be an integer, length of 10.

Create a new table, called orders_corrected, with the dates corrected and written as date strings, not longs. Here is the query:

```
hive> create table orders_corrected as select *,
from_unixtime(cast(substring(order_date,0,10) as INT)) as order_dateStr
from orders;
```

• **Question:** Using the orders_corrected table, count the number of orders (using order_id) that had a status of COMPLETE, on May 17, 2014. Show your query. (Notice, you can specify MONTH, YEAR and DAY as built-in functions to retreive the month, year or day from a date string).

• **Question:** What was Dualcore's total revenue from completed orders on May 17, 2014? (Hint: use a left semi join). Show your query.

• The result of the above query is in scientific notation. Rewrite the last query to format the value in dollars and cents (e.g., $2000000.00). To do this, format the result using the PRINTF function and the format string "$%.2f". Show your query.

This is the end of Part I:
Develop and Run Simple Queries

Part II: Data Management with Hive

In this lab you will practice using several common techniques for creating and populating Hive tables. You will also create and query a table containing each of the complex field types we studied: array, map, and struct.

Create and Load a Hive-Managed Table

In the following exercise, we will be using data available from the course website. Go to the downloads page:

```
/sites.google.com/site/hadoopti/downloads
```

and download the following file into your **datasets** directory on your VM

- `ratings_2012.txt`
- `ratings_2013.txt`
- `loyalty_data.txt`

Next, you will create and then load a Hive-managed table with product ratings data.

1. Create a table named ratings for storing tab-delimited records using this structure:

Field Name	Field Type
posted	TIMESTAMP
cust_id	INT
prod_id	INT
rating	TINYINT
message	STRING

2. Show the table description and verify that its fields have the correct order, names, and types:

```
DESCRIBE ratings;
```

4. Next, open a separate terminal window so you can run the following shell command. This will populate the table directly by using the hadoop fs command to copy product ratings data from 2012 to that directory in HDFS:

```
$ hadoop fs -put ~/datasets/ratings_2012.txt \
/user/hive/warehouse/ratings
```

Leave the window open afterwards so that you can easily switch between Hive and the command prompt.

4. Next, verify that Hive can read the data we just added. Run the following query in Hive to count the number of records in this table (the result should be 464):

```
SELECT COUNT(*) FROM ratings;
```

5. Another way to load data into a Hive table is through the LOAD DATA command. The next few commands will lead you through the process of copying a local file to HDFS and loading it into Hive. First, copy the 2013 ratings data to HDFS, into your user directory:

```
$ hadoop fs -put ~/datasets/ratings_2013.txt ratings_2013.txt
```

6. Verify that the file is there:

```
$ hadoop fs -ls ratings_2013.txt
```

7. Use the `LOAD DATA` statement in Hive to load that file into the ratings table:

```
LOAD DATA INPATH '/user/cloudera/ratings_2013.txt'
INTO TABLE ratings;
```

8. The LOAD DATA INPATH command moves the file to the table's directory. Verify that the file is no longer present in the original directory:

```
$ hadoop fs -ls ratings_2013.txt
```

9. Verify that the file is shown alongside the 2012 ratings data in the table's directory:

```
$ hadoop fs -ls /user/hive/warehouse/ratings
```

10. Finally, count the records in the ratings table to ensure that all 21,997 are available:

```
SELECT COUNT(*) FROM ratings;
```

Question: how many ratings are there?

Create, Load, and Query a Table with Complex Fields

The loyalty_data.tx file contains a sample of the data that contains information about customers who have signed up for a retailer's loyalty program. The file contains: phone numbers (as a map), a list of past order IDs (as an array), and a struct that summarizes the minimum, maximum, average, and total value of past orders. You will create the table, populate it with the provided data, and then run a few queries to practice referencing these types of fields.

Run the following statement in Hive to create the table:

```
CREATE TABLE loyalty_program
      (cust_id INT,
      fname STRING,
      lname STRING,
      email STRING,
      level STRING,
      phone MAP<STRING, STRING>,
      order_ids ARRAY<INT>,
      order_value STRUCT<min:INT,
                         max:INT,
                         avg:INT,
                         total:INT>)
ROW FORMAT DELIMITED
      FIELDS TERMINATED BY '|'
      COLLECTION ITEMS TERMINATED BY ','
      MAP KEYS TERMINATED BY ':';
```

Change into the directory containing the loyalty_data.txt file and examine the data in loyalty_data.txt to see how it corresponds to the fields in the table. Then, load it into Hive:

```
LOAD DATA LOCAL INPATH 'loyalty_data.txt' INTO TABLE
loyalty_program;
```

Create the following queries:

1. Select the HOME phone number (**Hint**: Map keys are case-sensitive) for customer ID 1200866. You should see 408-555-4914 as the result.

2. Select the third element from the order_ids array for customer ID 1200866 (**Hint**: Elements are indexed from zero). The query should return 5278505.

3. Select the total attribute from the order_value struct for customer ID 1200866. The query should return 401874.

Show the 3 queries that you ran.

Alter and Drop a Table

1. Use ALTER TABLE to rename the **`level`** column to **`status`**.

2. Use the DESCRIBE command on the loyalty_program table to verify the change.

3. Use ALTER TABLE to rename the entire table to reward_program.

4. Although the ALTER TABLE command often requires that we make a corresponding change to the data in HDFS, renaming a table or column does not.

You can verify this by running a query on the table using the new names (the result should be “SILVER”):

```
SELECT status FROM reward_program WHERE cust_id = 1200866;
```

5. As sometimes happens in the corporate world, priorities have shifted and the program is now canceled. Drop the reward_program table.

Show the queries that you ran for steps 1 - 5.

This is the end of Part II: Data Management with Hive.

Bonus question (+5 pts)

On your VM, in the `datasets` directory, there is a file called `access_log.gz` and it contains compressed log entries.

Create a table, using RegexSerde, to load this file into Hive. Your new table should contain entries for IP address, date_and_time, request, response and bytes_read.

Bonus question: Show the query you ran to create the table.