Big data analytical project

Version 0.01

Dataco

Data, Database, Dataset, Datamining, Datamart, Data Warehouse, Dataops, Data Lakes, Data Literacy, Data Analysis, Data Visualization,
Big data case study : Walmart, IKEA,

Darraj Hossain

November 17, 2020

A. Dataco project Details

1. Business Details
2. Tables Details
3. Schema Details
4. Total Dataset as Tablewise
2. Total Data as Tablewise
3. Outcome through data analysis

B. Platform

1. Open source platform
2. AWS Core Solution and local
3. Cloudera Vendor Solution

C. Software

1. Oracle VirtualBox
2. Cloudera Vendor solution
3. Centos
4. Ubuntu
5. Hadoop
6. Hive
7. MySQL
8. Sqoop
9. Power BI
10. GitHub
11. Python
12. Pandas
13. SQLite

D. Admin Job

1. Virtual Box setup
2. Linux Command
3. Hadoop admin command
4. Cloudera Admin command
5. Hive Admin command
6. Power BI ODBC

E. Data analytical process

1. Hive CLI
2. HUE
3. Impala
4. HiveServer2
5. PowerBI ODBC
6. MySQL
7. Sqoop
8. Power BI

F. SQL Command for run the business

1. Select
2. Bucket
3. Partitioning

G. Data visualization

1. SQL command
2. Group & Graph

H. Data visualization

1. GitHub
2. Colab
3. Python
4. Panda
5, SQLite command
2. Analysis

I. One million Dataset Lab under Big Data Cloud Lab

Project Papers

Dataco project Details

1. Business Details
2. Tables Details
3. Schema Details
4. Total Dataset as Tablewise
2. Total Data as Tablewise
3. Outcome through data analysis

Define a Business Question

For the remainder of this tutorial, we will present examples in the context of a made up corporation called **DataCo**, and our mission is to help the organization get better insight by asking bigger questions.

Scenario:

Your Management: is talking euphorically about Big Data...

You: are carefully skeptical, as it will most likely all land on your desk anyway. Alternatively it has already landed on you, with the nice project description of: Go figure this Hadoop thing out...

Good to Know

Any successful PoC needs to address something your organization cares about. Hence, the first thing you need to do is to: define a business question.

It won't just impress your manager that you think big and have perspective on the business needs of your organisation (which in English means you just helped your manager to look good in front of his management). It will also help you to go through a well scoped PoC and get the investments you need to be successful.

Without a well defined question, you won't know how to properly model your data, i.e. what structure to apply at query time, or what data sets and tools to use to best serve the use case.

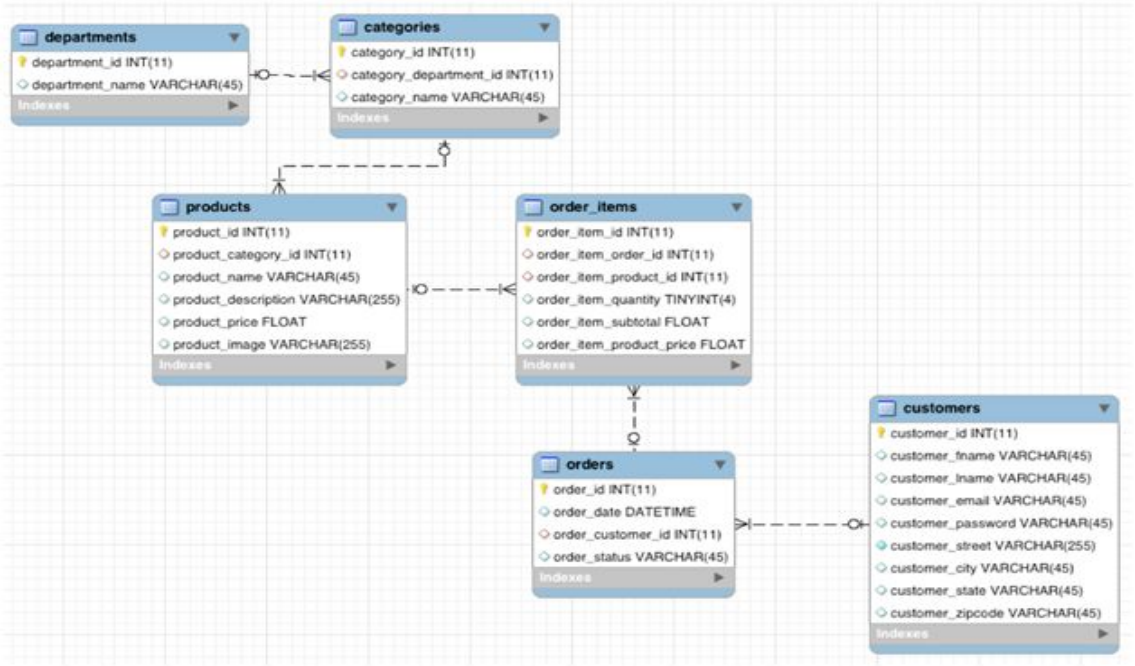
Tutorial Exercise 1

Ingest and Query Relational Data

In this scenario, DataCo's business question is: What products do our customers like to buy? To answer this question, the first thought might be to look at the transaction data, which should indicate what customers actually do buy and like to buy, right?

This is probably something you can do in your regular RDBMS environment, but a benefit with Cloudera's platform is that you can do it at greater scale at lower cost, on the same system that you may also use for many other types of analysis.

What this exercise demonstrates is how to do exactly the same thing you may already know how to do with traditional databases, but in CDH. Seamless integration is important when evaluating any new infrastructure. Hence, it's important to be able to do what you normally do, and not break any regular BI reports or workloads over the dataset you plan to migrate.



About Sqoop:

Apache Sqoop is a tool that uses MapReduce to transfer data between Hadoop clusters and relational databases very efficiently. It works by spawning tasks on multiple data nodes to download various portions of the data in parallel. When you're finished, each piece of data is replicated to ensure reliability, and spread out across the cluster to ensure you can process it in parallel on your cluster.

There are 2 versions of Sqoop included in Cloudera's platform. Sqoop 1 is a "thick client" and is what you use in this tutorial. The command you run will directly submit the MapReduce jobs to transfer the data. Sqoop 2 consists of a central server that submits the MapReduce jobs on behalf of clients, and a much lighter weight client that you use to connect to the server. The "Sqoop" you see in Cloudera Manager is the Sqoop 2 server, although Cloudera Manager will make sure that both the "sqoop" and "sqoop2" command are correctly configured on all your machines.

To analyze the transaction data in the new platform, we need to ingest it into the Hadoop Distributed File System (HDFS). We need to find a tool that easily transfers structured data from a RDBMS to HDFS, while preserving structure. That enables us to query the data, but not interfere with or break any regular workload on it.

Apache Sqoop, which is part of CDH, is that tool. The nice thing about Sqoop is that we can automatically load our relational data from MySQL into HDFS, while preserving the structure.

MySQL Database :

1. [cloudera@quickstart ~]$ mysql -uretail_dba -pcloudera
2. mysql> show databases;
3. mysql> use retail_db:
4. mysql > show tables:
   Here, you can see like below:
   ±--------------------+
   | Tables_in_retail_db |
   ±--------------------+
   | categories |
   | customers |
   | departments |
   | order_items |
   | orders |
   | products |
   ±--------------------
5. Write your sqoop import/export/eval etc etc commands in your sqoop scripts which you can migrate data from MySQL to HDFS/HIVE and vice versa.

With a few additional configuration parameters, we can take this one step further and load this relational data directly into a form ready to be queried by Impala (the open source analytic query engine included with CDH). Given that we may want to leverage the power of the Apache Avro file format for other workloads on the cluster (as Avro is a Hadoop optimized file format), we will take a few extra steps to load this data into Impala using the Avro file format, so it is readily available for Impala as well as other workloads.

You should first open a terminal, which you can do by clicking the black "Terminal" icon at the top of your screen. Once it is open, you can launch

the Sqoop job:

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

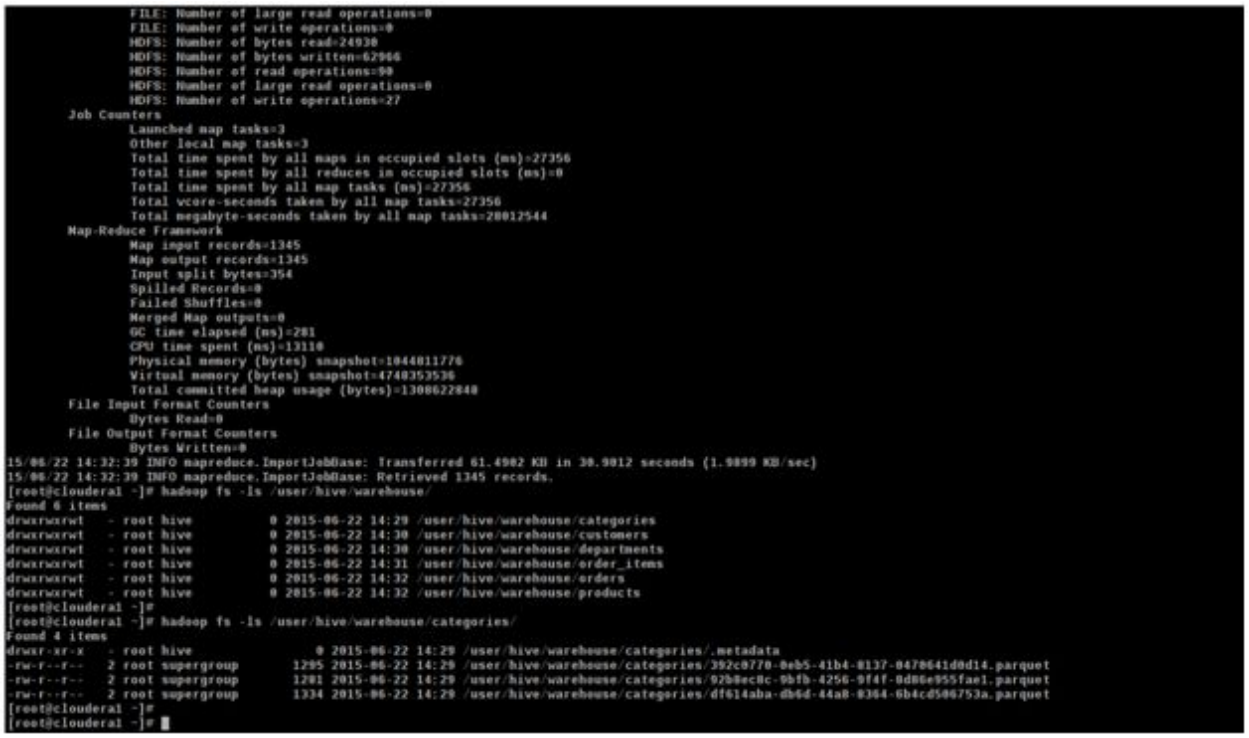
This command may take a while to complete, but it is doing a lot. It is launching MapReduce jobs to pull the data from our MySQL database and write the data to HDFS, distributed across the cluster in Apache Parquet format. It is also creating tables to represent the HDFS files in Impala / Apache Hive with a matching schema.

Parquet is a format designed for analytical applications on Hadoop. Instead of grouping your data into rows like typical data formats, it groups your data into columns. This is ideal for many analytical queries where instead of retrieving data from specific records, you're analyzing relationships between specific variables across many records. Parquet is designed to optimize data storage and retrieval in these scenarios.

Once the command is complete we can confirm that our data was imported into HDFS:

```
[cloudera@quickstart ~]$ hadoop fs -ls /user/hive/warehouse/

[cloudera@quickstart ~]$ hadoop fs -ls /user/hive/warehouse/categories/
```

These commands will show the directores and the files inside them that make up our tables:

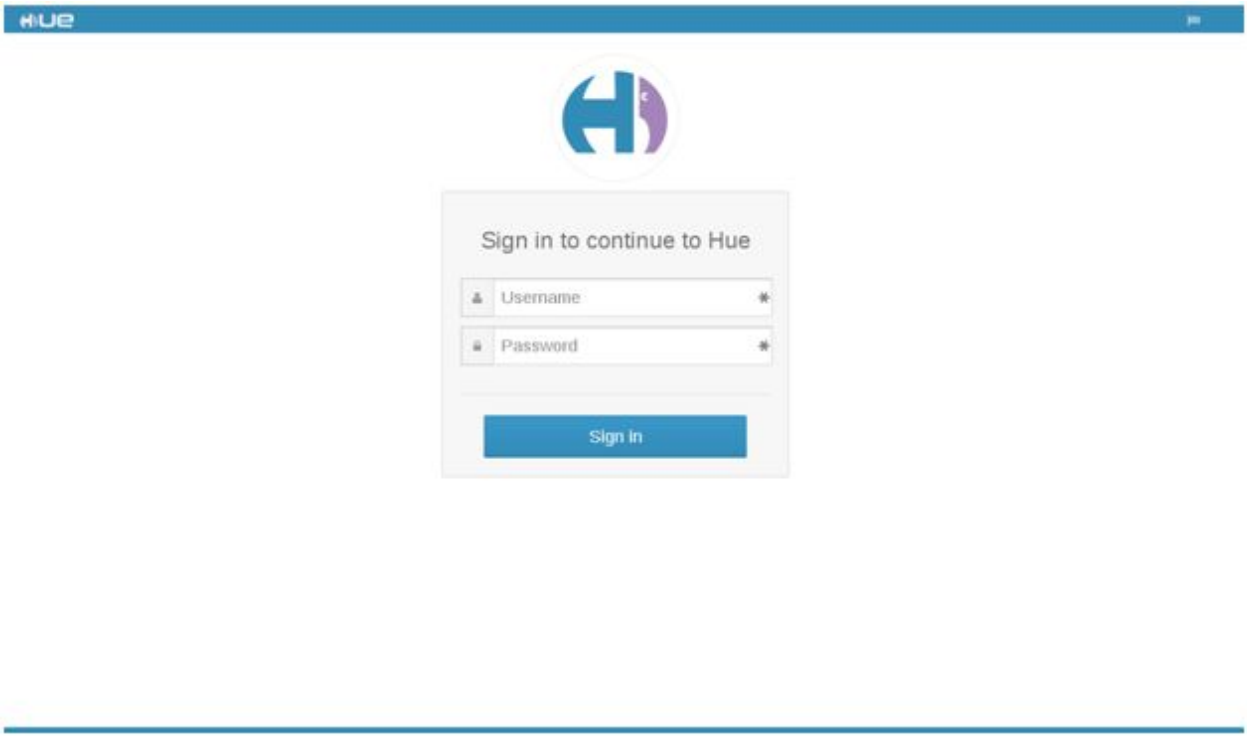


Note: The number of .parquet files shown will be equal to the number of mappers used by Sqoop. On a single-node you will just see one, but larger clusters will have a greater number of files.

Hive and Impala also allow you to create tables by defining a schema over existing files with **'CREATE EXTERNAL TABLE'** statements, similar to traditional relational databases. But Sqoop already created these tables for us, so we can go ahead and query them.

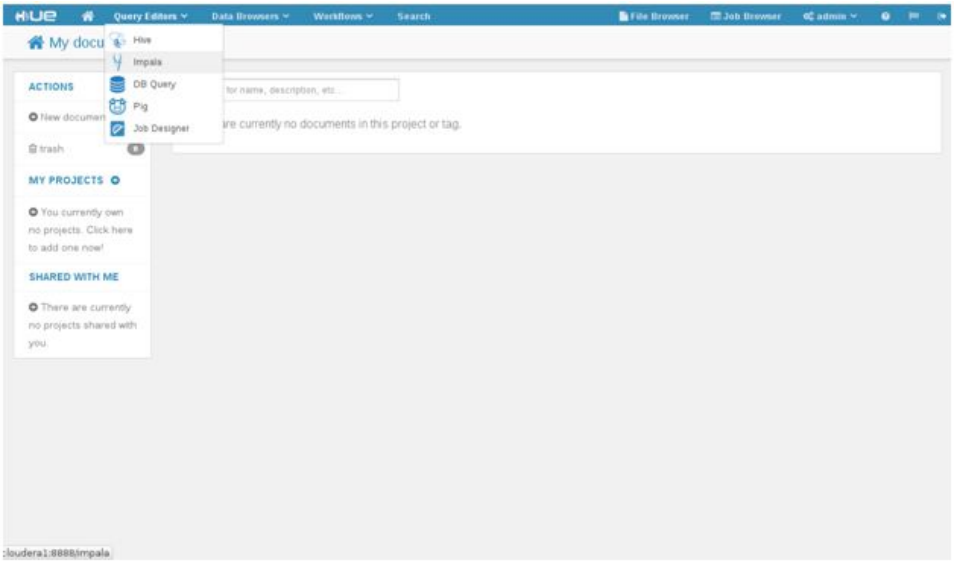
We're going to use Hue's Impala app to query our tables. Hue provides a web-based interface for many of the tools in CDH and can be found on port 8888 of your Manager Node (here). In the QuickStart VM, the administrator username for Hue is 'cloudera' and the password is 'cloudera'.

Once you are inside of Hue, click on Query Editors, and open the Impala Query Editor.



HIVE Database :

Once you are inside of Hue, click on Query Editors, and open the Impala Query Editor.

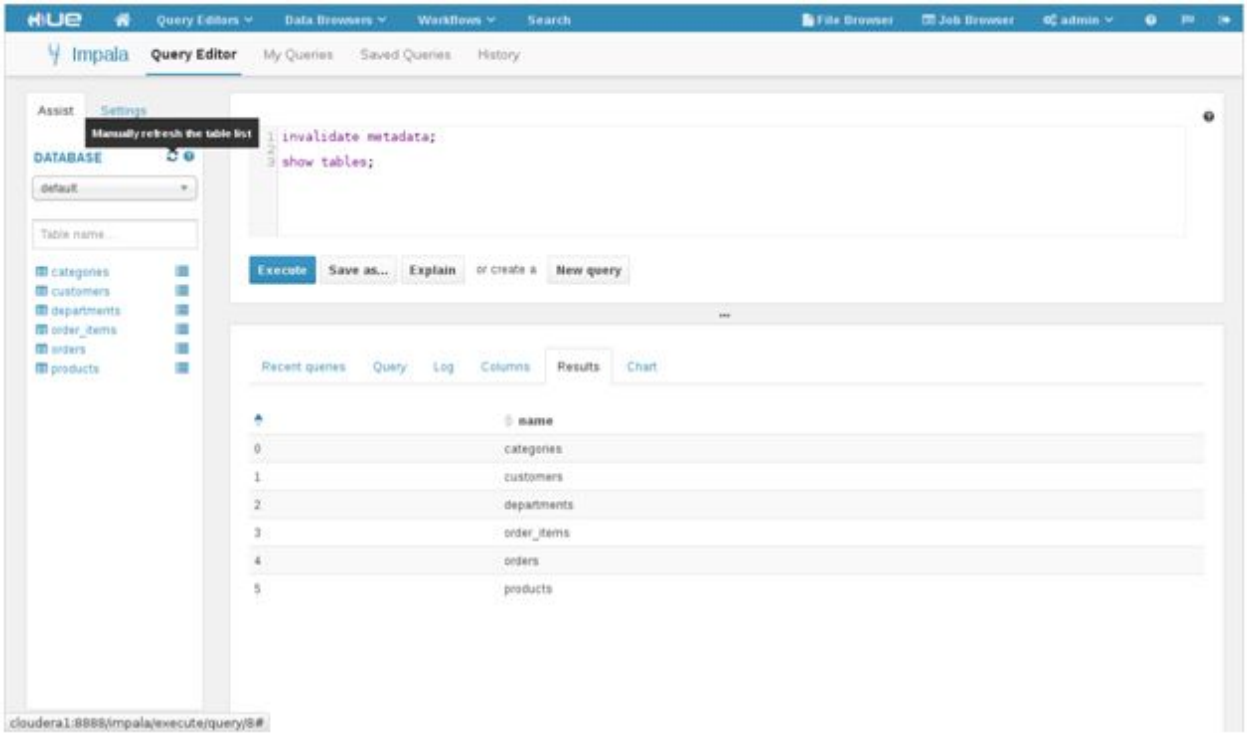


To save time during queries, Impala does not poll constantly for metadata changes. So the first thing we must do is tell Impala that its metadata is out of date. Then we should see our tables show up, ready to be queried:

invalidate metadata;

show tables;

You can also click on the "Refresh Table List" icon on the left to see your new tables in the side menu.

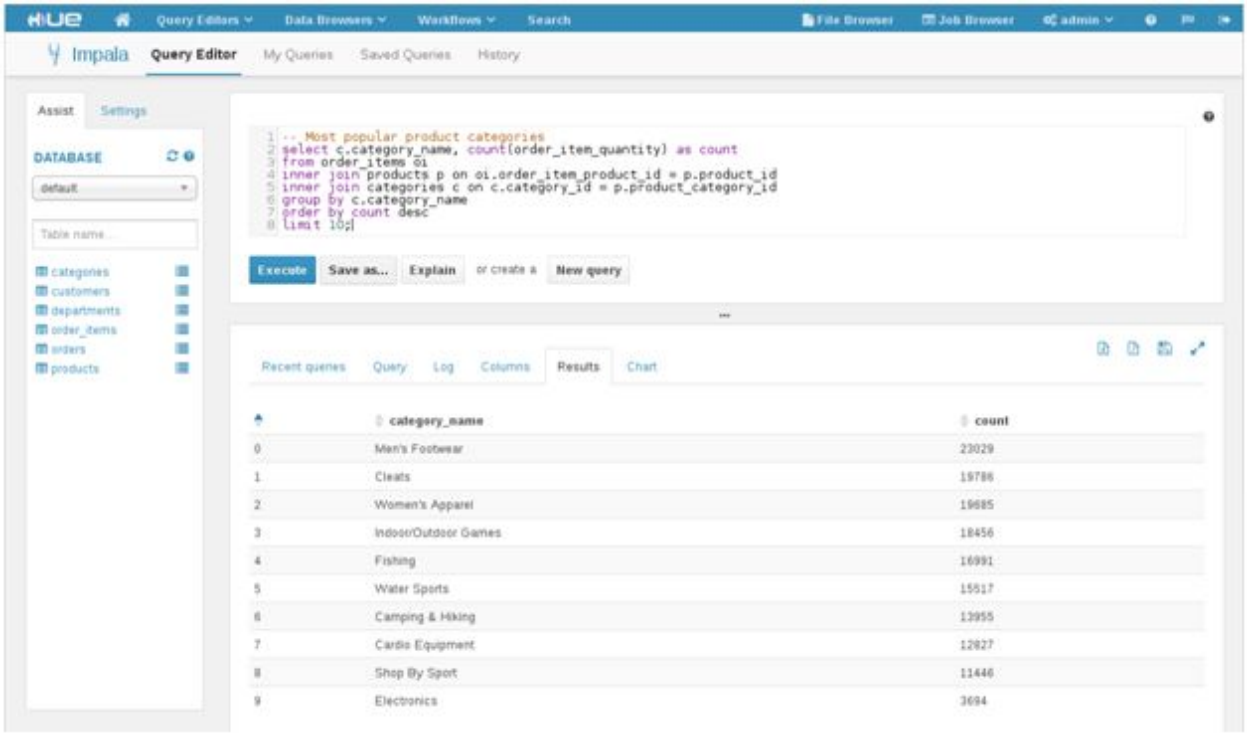


Business model query :

Now that your transaction data is readily available for structured queries in CDH, it's time to address DataCo's business question. Copy and paste or type in the following standard SQL example queries for calculating total revenue per product and showing the top 10 revenue generating products:

```
-- Most popular product categories
select c.category_name, count(order_item_quantity) as count
from order_items oi
inner join products p on oi.order_item_product_id = p.product_id
inner join categories c on c.category_id = p.product_category_id
group by c.category_name
order by count desc
limit 10;
```

You should see results of the following form:

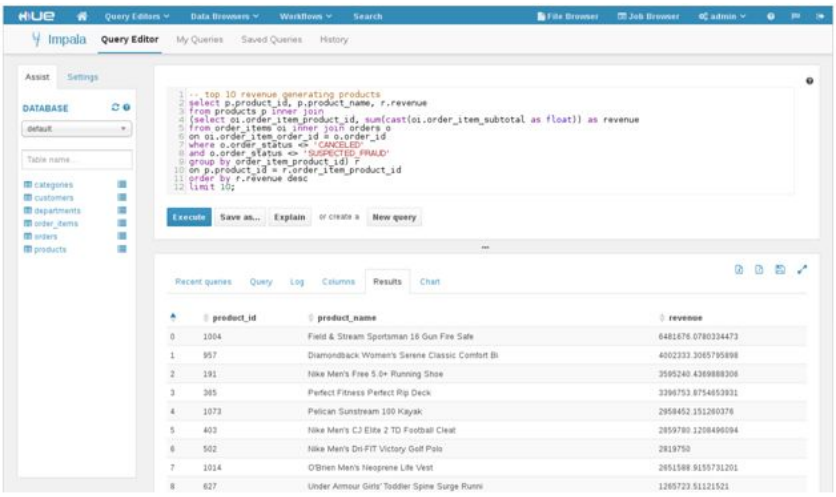


Business Model query 2 :

Clear out the previous query, and replace it with the following:

```
-- top 10 revenue generating products

select p.product_id, p.product_name, r.revenue

from products p inner join

(select oi.order_item_product_id, sum(cast(oi.order_item_subtotal as float)) as revenue

from order_items oi inner join orders o

on oi.order_item_order_id = o.order_id

where o.order_status <> 'CANCELED'

and o.order_status <> 'SUSPECTED_FRAUD'

group by order_item_product_id) r

on p.product_id = r.order_item_product_id

order by r.revenue desc

limit 10;
```

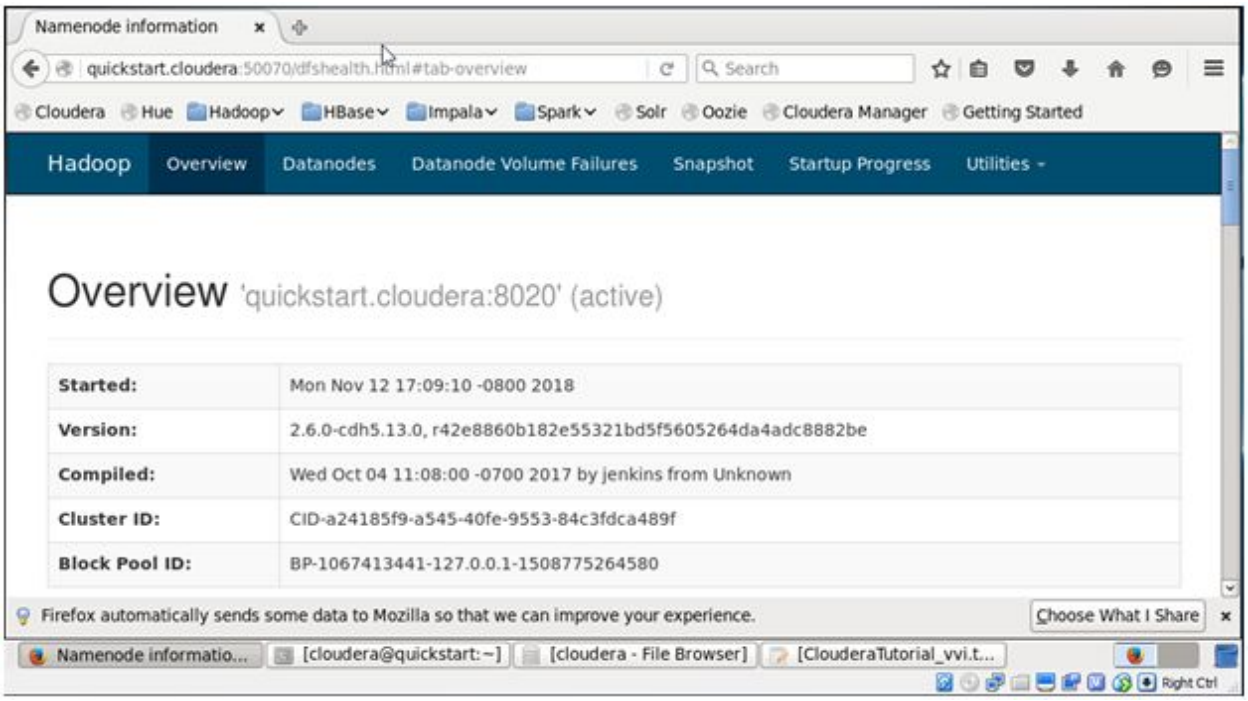
You should see results similar to this:



You may notice that we told Sqoop to import the data into Hive but used Impala to query the data. This is because Hive and Impala can share both data files and the table metadata. Hive works by compiling SQL queries into MapReduce jobs, which makes it very flexible, whereas Impala executes queries itself and is built from the ground up to be as fast as possible, which makes it better for interactive analysis. We'll use Hive later for an ETL (extract-transform-load) workload.

If one of these steps fails, please reach out to our Cloudera Community and get help. Otherwise continue.

Hadoop Big Data Platform and Hive Data warehouse:



CONCLUSION

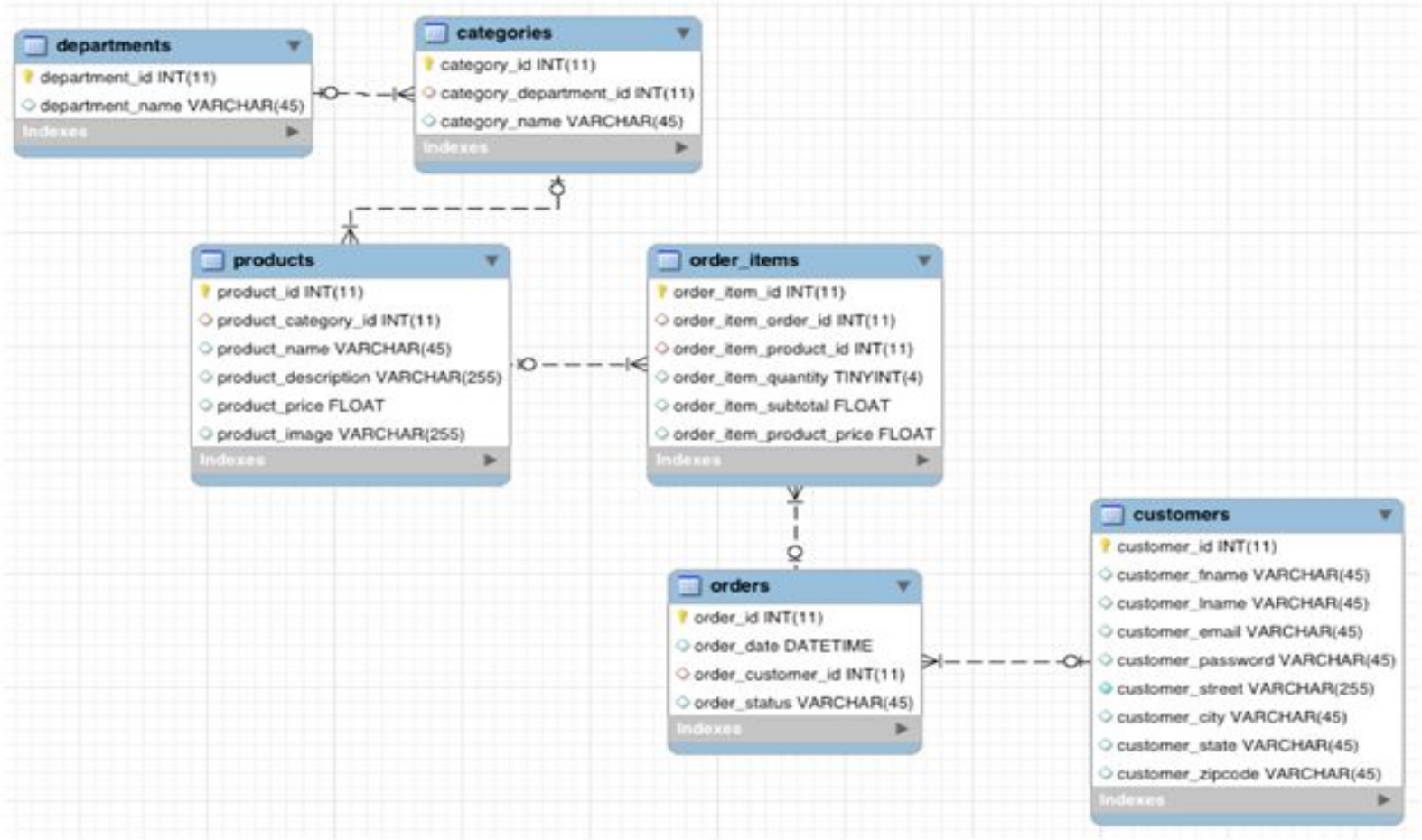
Now you have gone through the first basic steps to Sqoop structured data into HDFS, transform it into Avro file format (you can read about the benefits of Avro as a common format in Hadoop here), and import the schema files for use when we query this data.

Now you have learned how to create and query tables using Impala and that you can use regular interfaces and tools (such as SQL) within a Hadoop environment as well. The idea here being that you can do the same reports you usually do, but where the architecture of Hadoop vs traditional systems provides much larger scale and flexibility.

Dataco Data Analysis Module

Step by step

1.Database structure:



2.Datasets: 256,270 (250K)

Name of Database : retail_db

Tables Name (6) :

1. customers - 12435
2. categories - 58
3. order_items - 172198
4. Orders - 68883
5. departments - 6
6. Products - 1345

3.Data Type:

int
String
float
bigint

4.Dataset Example :

Customer
12435 Laura Horton XXXXXXXXX XXXXXXXXX 5736 Honey Downs Summerville SC 29483

categories
58 8 NFL Players

order_items
172198 68883 502 3 150.0 50.0

orders
68883 1406098800000 5533 COMPLETE

departments - 6
6 Fan Shop

products
944 43 GoPro HERO3+ Black Edition Camera 399.99
http://images.acmesports.sports/GoPro+HERO3%2B+Black+Edition+Camera

5.Database schema:

create table **customers**
(customer_id INT,
 customer_fname STRING,
 customer_lname STRING,
customer_email string,
customer_password string,
 customer_street STRING,
 customer_city STRING,
 customer_state STRING,
 customer_zipcode STRING) row format delimited Fields terminated by ',';

 create table **categories**
(category_id int,
category_department_id int,
category_name string) row format delimited Fields terminated by ',';

create table **order_items**
(order_item_id int,
order_item_order_id int,
order_item_product_id int,
order_item_quantity int,
order_item_subtotal float,
order_item_product_price float) row format delimited Fields terminated by ',';

create table **orders**
(order_id int,
order_date bigint,
order_customer_id int,
order_status string) row format delimited Fields terminated by ',';

create table **departments**
(department_id int,
department_name string) row format delimited Fields terminated by ',';

create table **products**
(product_id int,
product_category_id int,
product_name string,
product_description string,
product_price float,
product_image string) row format delimited Fields terminated by ',';

6.Data Loading into TABLE TO Hive Database:

LOAD DATA LOCAL INPATH '/home/hduser/Desktop/CustomerNY' INTO TABLE customers;

7.Hive SQL Warmup command:

```
 INSERT OVERWRITE LOCAL DIRECTORY
'/media/hduser/1TB/00.MDC@HDW_Jan12.2019/06.DataSet/ClouderaQuickStart/ClouderaQuickStart/'
        ROW FORMAT DELIMITED
        FIELDS TERMINATED BY ','
        LINES TERMINATED BY '\n'
        STORED AS TEXTFILE
        select * from customers where customer_city='Brooklyn';

hive>SHOW DATABASES;

hive> use retailsdb;

hive> show tables;

hive> select * from customers;

hive> select count(*) from customers;
```

Complex SQL command:

A. **Most popular product categories**

```
select c.category_name, count(order_item_quantity) as count
from order_items oi
inner join products p on oi.order_item_product_id = p.product_id
inner join categories c on c.category_id = p.product_category_id
group by c.category_name
order by count desc
limit 10;
```

B. **top 10 revenue generating products**

```
select p.product_id, p.product_name, r.revenue
from products p inner join
(select oi.order_item_product_id, sum(cast(oi.order_item_subtotal as float)) as revenue
from order_items oi inner join orders o
on oi.order_item_order_id = o.order_id
where o.order_status <> 'CANCELED'
and o.order_status <> 'SUSPECTED_FRAUD'
group by order_item_product_id) r
on p.product_id = r.order_item_product_id
order by r.revenue desc
limit 10;
```

SQL Command lists:

1. Show databases
2. Create database
3. Use database
4. Create table
5. Drop table
6. Drop database
7. Load data
8. Select Data
9. Describe table
10. Alter add/ column in Existing table
11. Alter replace=remove/ column in Existing table without data lost
12. Where clause
13. Group clause
14. Join two tables
15. Partitioning Table
16. Buckets
17. External Table
18. Sequence Table
19. Map Join Table
20. Storing Data into File
21. Indexing
22. Hive UDF

Partitioned table:

```
partitioned by (state STRING)
clustered by (customer_state)
Sorted by (customer_state)
 INTO 10 buckets
row format delimited Fields terminated by ',';

 Configure hive file
 set hive.exec.dynamic.partition.mode=nonstrict;
set hive.exec.dynamic.partition=true;
set hive.enforce.bucketing=true;

 (customer_id INT,
 customer_fname STRING,
 customer_lname STRING,
customer_email string,
customer_password string,
 customer_street STRING,
 customer_city STRING,
 customer_state STRING,
 customer_zipcode STRING)
```

Load data into partition table by category