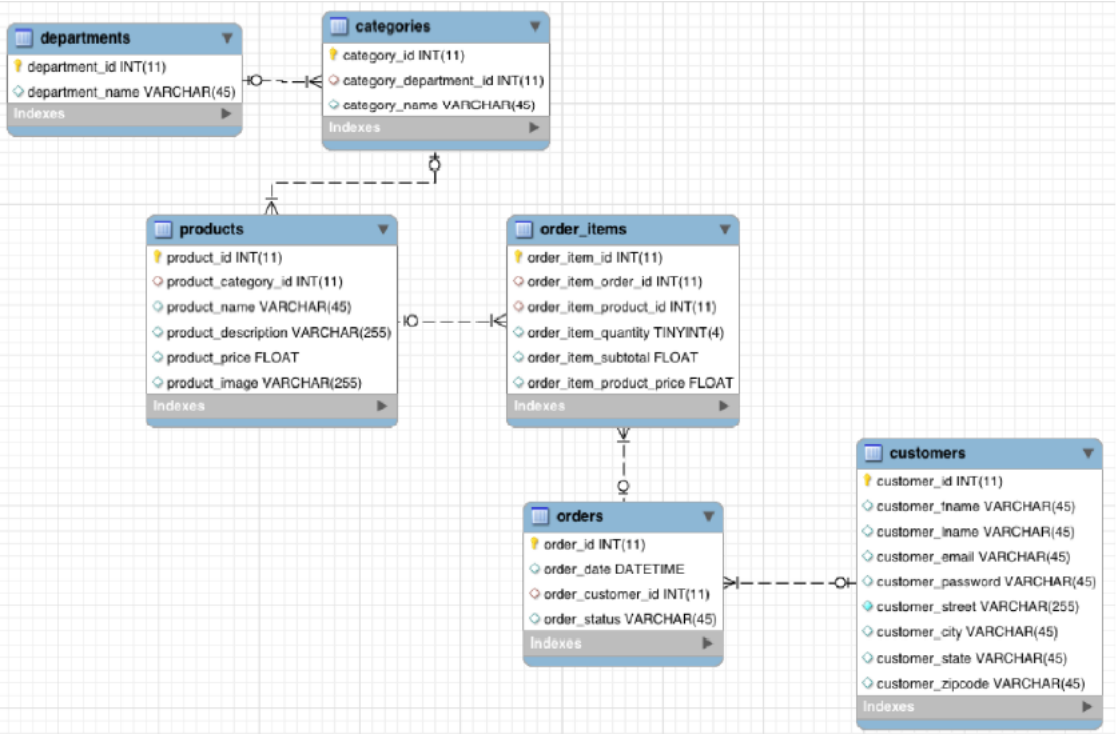
เอกสารภาคปฏิบัติ

(LAB)



ศส.กรอ.

Tutorial Exercise 1

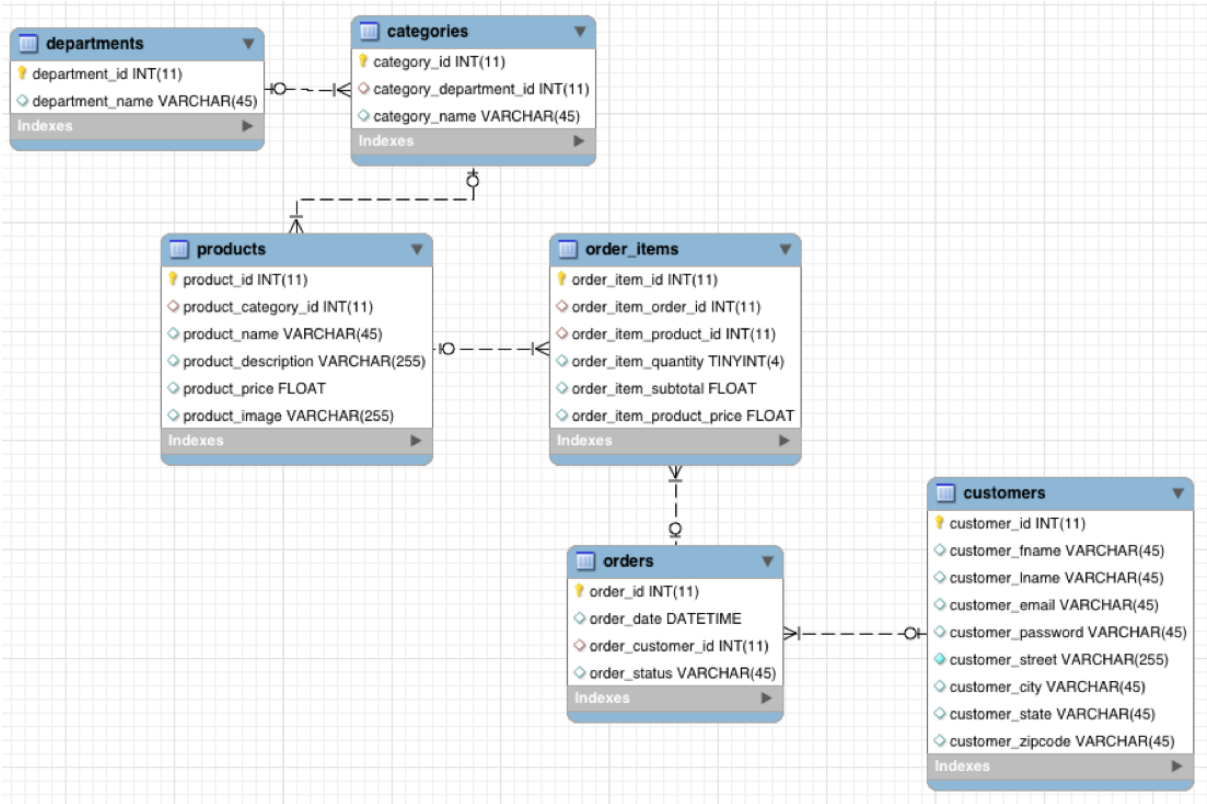
Ingest and Query Relational Data

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.



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. 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 matching schema.

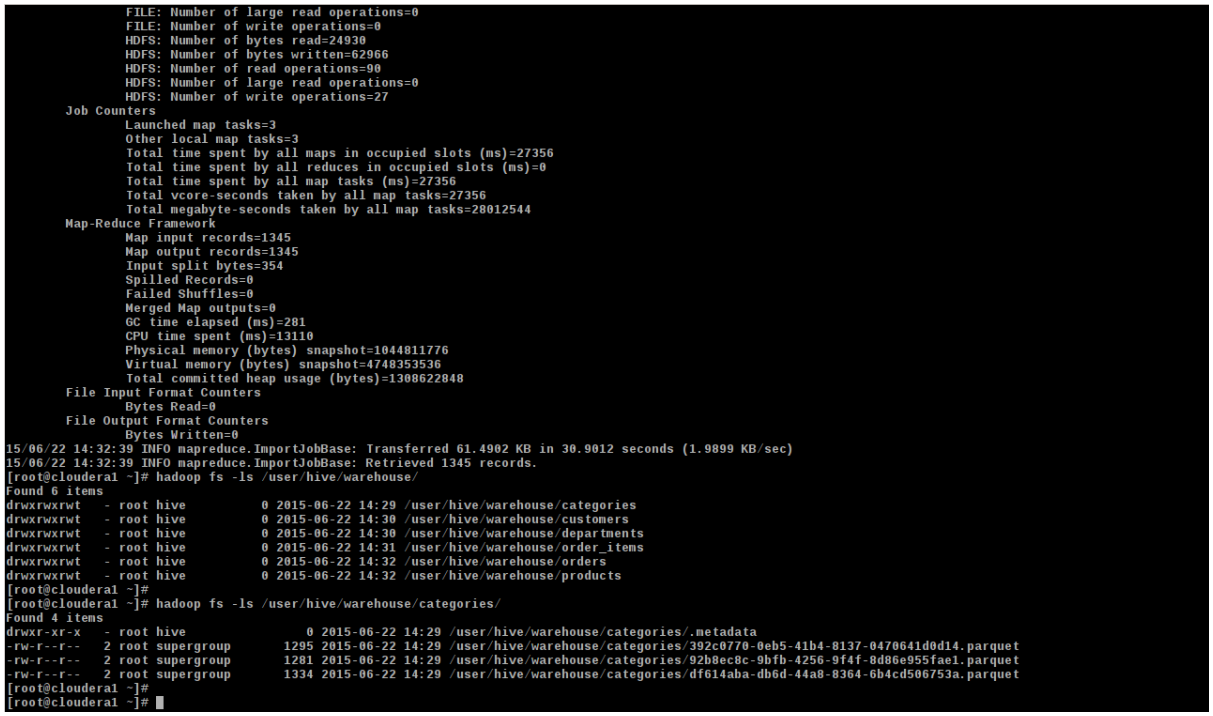
Parquet is a format designed for analytical applications on Hadoop. Instead of grouping your data into rows like typical data formats, it groups your data into columns. This is ideal for many analytical queries where instead of retrieving data from specific records, you're

analyzing relationships between specific variables across many records. Parquet is designed to optimize data storage and retriveval 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:

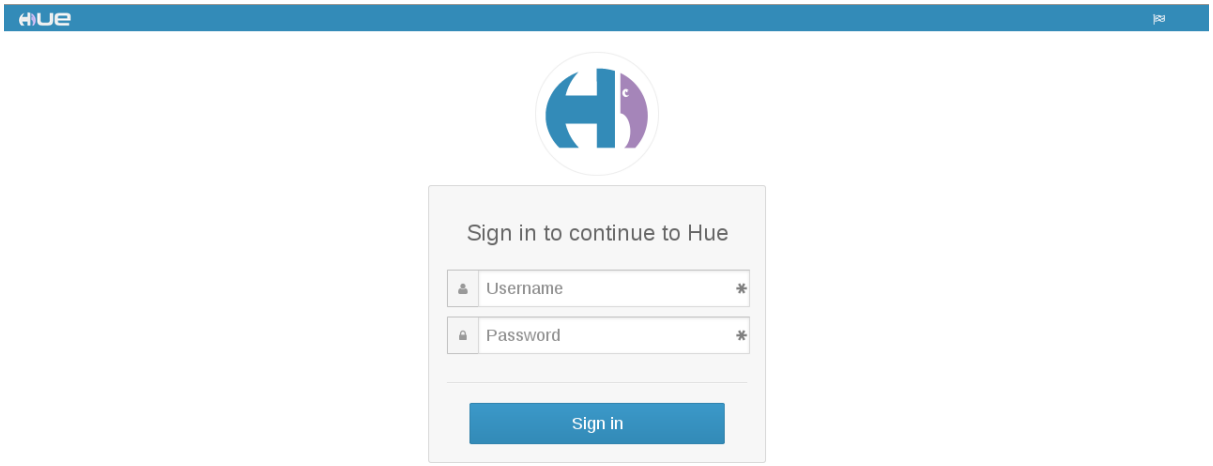


Note: The number of .parquet files shown will be equal to the number of mappers used by Sqoop. On a single-node you will just see one, but larger clusters will have a greater number of files.

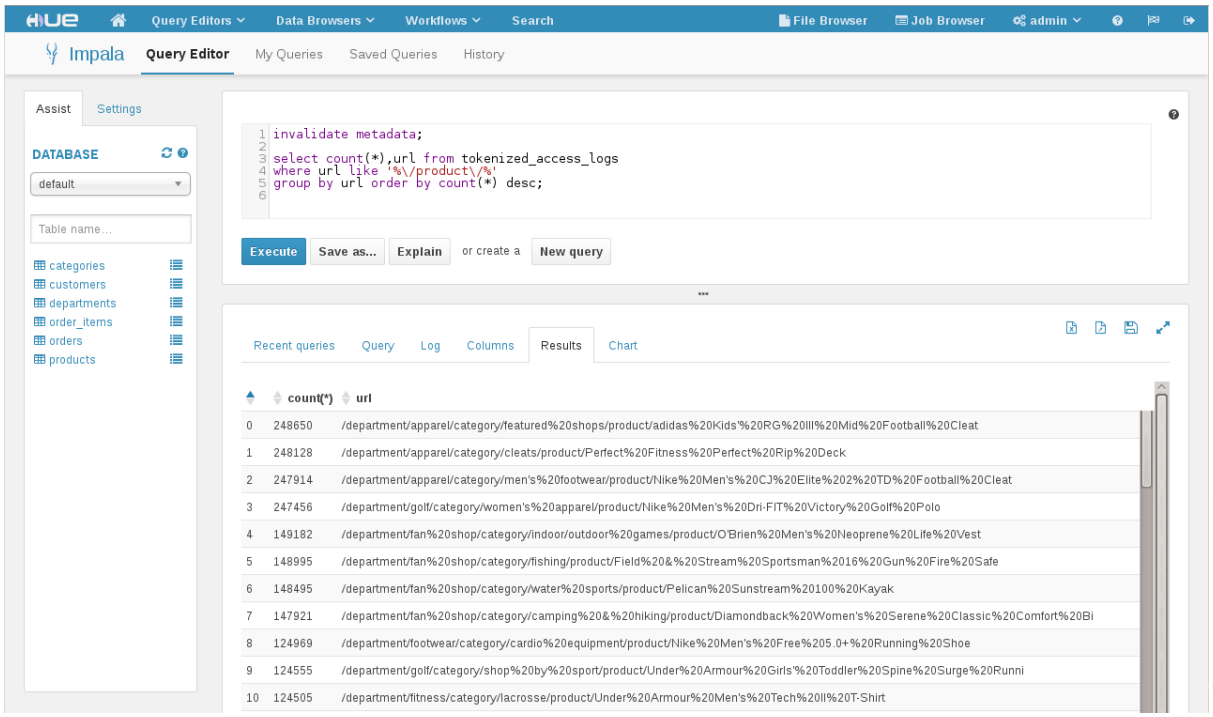
Hive and Impala also allow you to create tables by defining a schema over existing files with 'CREATE EXTERNAL TABLE' statements, similar to traditional relational databases. But Sqoop already created these tables for us, so we can go ahead and query them.

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.

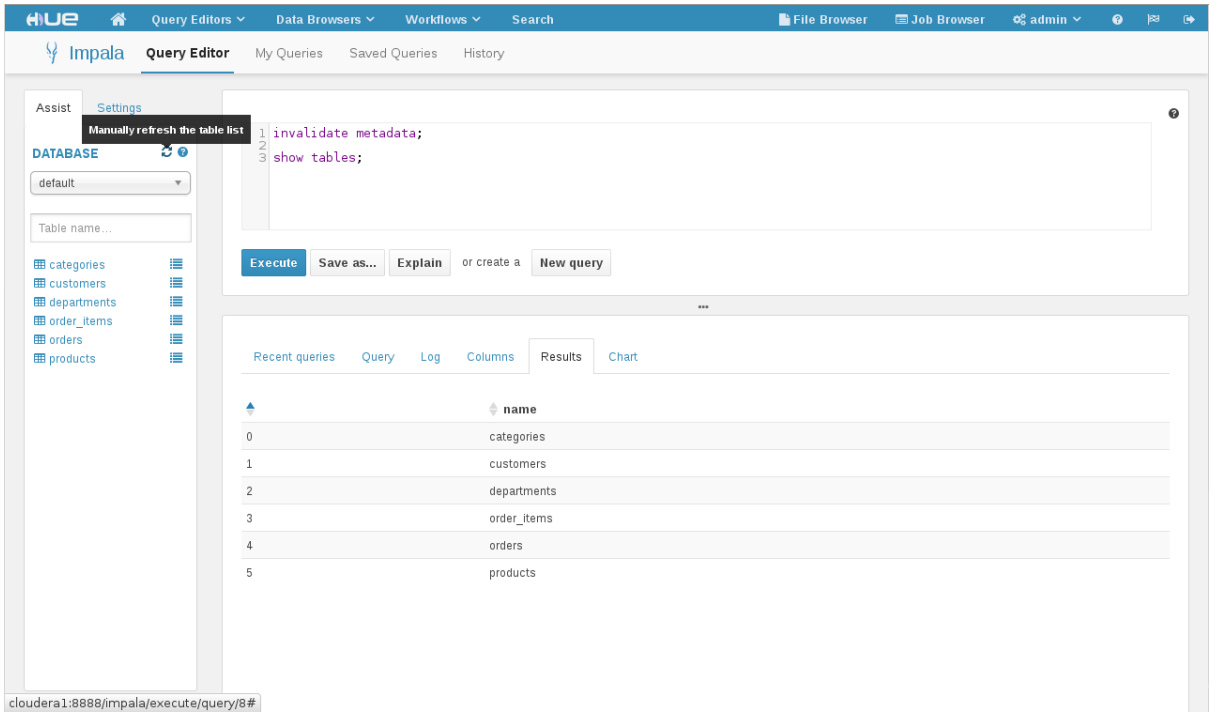


To save time during queries, Impala does not poll constantly for metadata changes.

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;
```

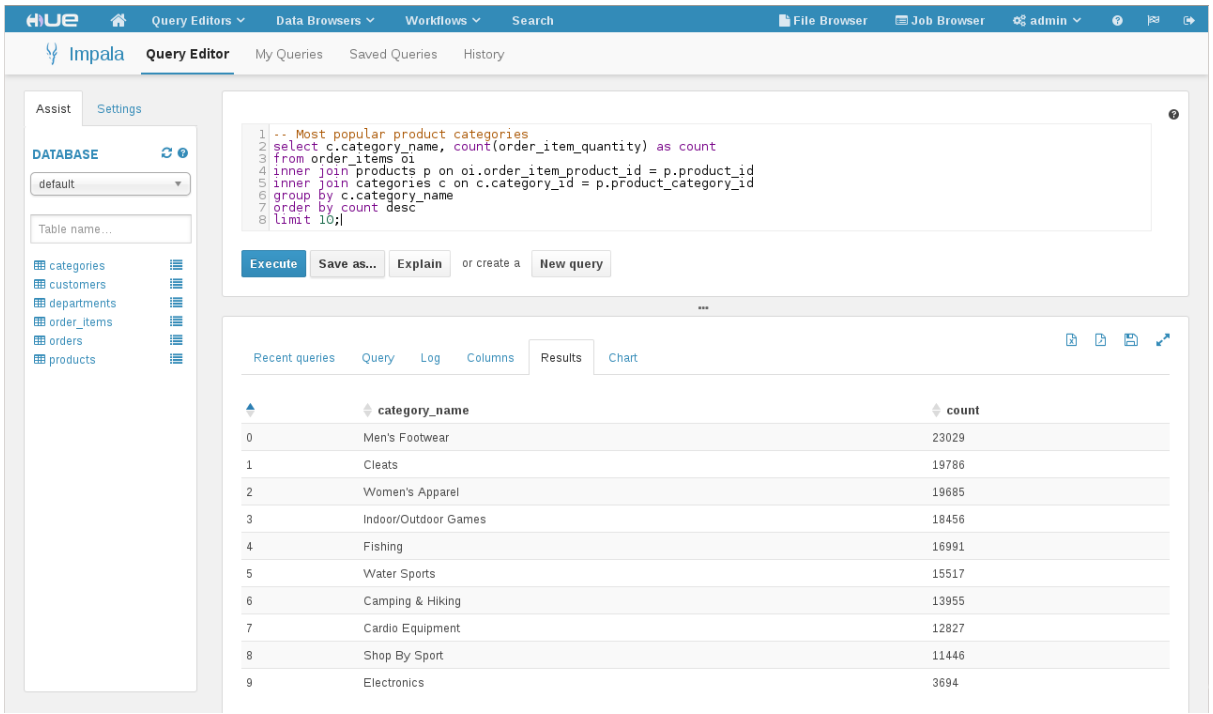
You can also click on the "Refresh Table List" icon on the left to see your new tables in the side menu.



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
```

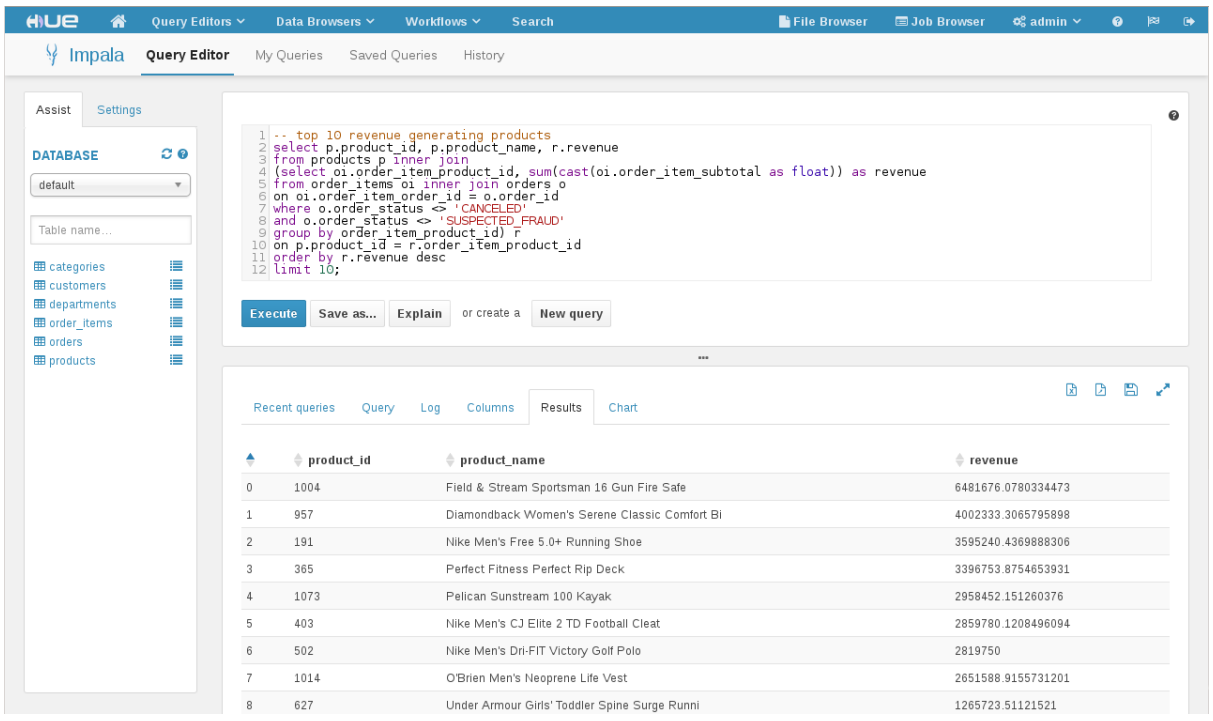
You should see results of the following form:



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:



	product_id	product_name	revenue
0	1004	Field & Stream Sportsman 16 Gun Fire Safe	6481676.0780334473
1	957	Diamondback Women's Serene Classic Comfort Bi	4002333.3065795898
2	191	Nike Men's Free 5.0+ Running Shoe	3595240.4369888306
3	365	Perfect Fitness Perfect Rip Deck	3396753.8754653931
4	1073	Pelican Sunstream 100 Kayak	2958452.151260376
5	403	Nike Men's CJ Elite 2 TD Football Cleat	2859780.1208496094
6	502	Nike Men's Dri-FIT Victory Golf Polo	2819750
7	1014	O'Brien Men's Neoprene Life Vest	2651588.9155731201
8	627	Under Armour Girls' Toddler Spine Surge Runni	1265723.51121521

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.

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.

on.

Tutorial Exercise 2

Correlate Structured Data with Unstructured Data

Since you are a pretty smart data person, you realize another interesting business question would be: *are the most viewed products also the most sold?* (or for other scenarios, the most searched for, the most chatted about...). Since Hadoop can store unstructured and semi-structured data alongside structured data without remodelling an entire database, you can just as well ingest, store and process web log events. Let's find out what site visitors have actually viewed the most.

For this, you need the web clickstream data. The most common way to ingest web clickstream is to use Flume. Flume is a scalable real-time ingest framework that allows you to route, filter, aggregate, and do “mini-operations” on data on its way in to the scalable processing platform.

In Exercise 4, later in this tutorial, you can explore a Flume configuration example, to use for real-time ingest and transformation of our sample web clickstream data. However, for the sake of tutorial-time, in this step, we will not have the patience to wait for three days of data to be ingested. Instead, we prepared a web clickstream data set (just pretend you fast forwarded three days) that you can bulk upload into HDFS directly.

Bulk Upload Data

For convenience, we have loaded a sample (about 180K lines) of one month's worth of access log data into /opt/examples/log_data/access.log.2.

Let's move this data from the local filesystem, into HDFS.

Go back to your terminal and execute the following commands from your **Manager Node**.

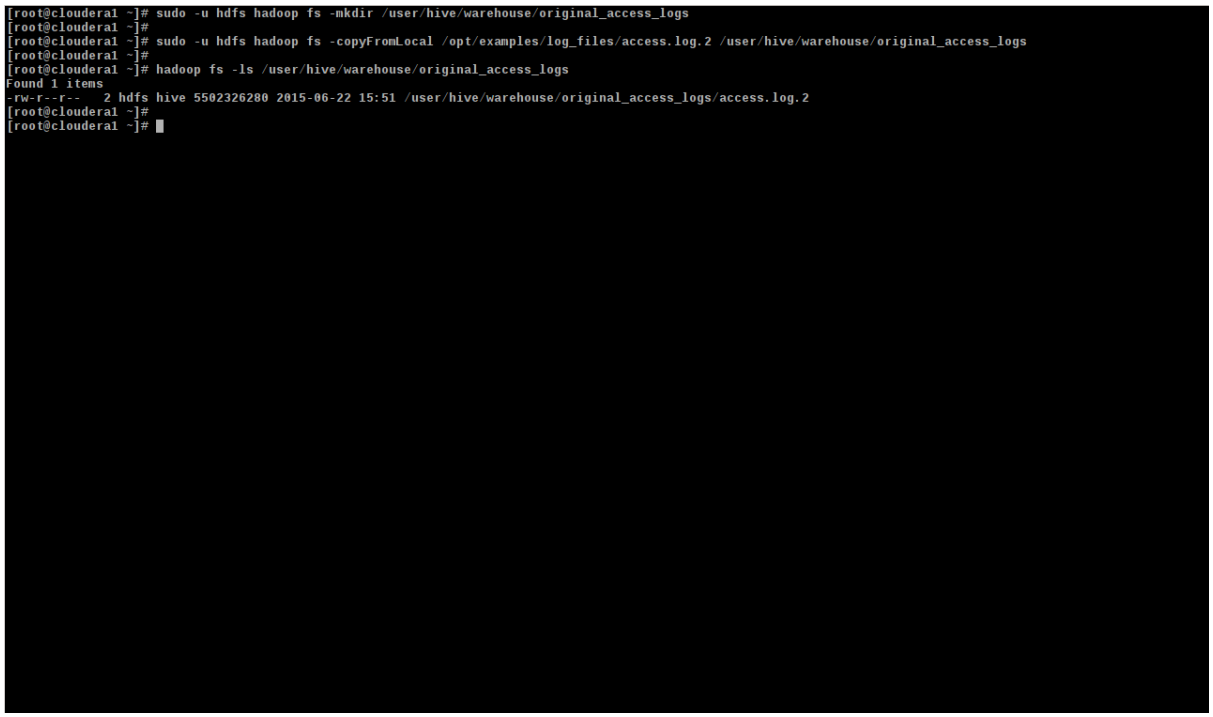
```
[cloudera@quickstart ~]$ sudo -u hdfs hadoop fs -mkdir
/user/hive/warehouse/original_access_logs
[cloudera@quickstart ~]$ sudo -u hdfs hadoop fs -copyFromLocal
/opt/examples/log_files/access.log.2 /user/hive/warehouse/original_access_logs
```

The copy command may take several minutes to complete.

Verify that your data is in HDFS by executing the following command:

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

You should see a result similar to the following:



Now you can build a table in Hive and query the data via Impala and Hue. You'll build this table in 2 steps. First, you'll take advantage of Hive's flexible SerDes (serializers / deserializers) to parse the logs into individual fields using a regular expression. Second, you'll transfer the data from this intermediate table to one that does not require any special SerDe. Once the data is in this table, you can query it much faster and more interactively using Impala.

We'll use the Hive Query Editor app in Hue to execute the following queries:

```
CREATE EXTERNAL TABLE intermediate_access_logs (
    ip STRING,
    date STRING,
    method STRING,
    url STRING,
    http_version STRING,
    code1 STRING,
    code2 STRING,
    dash STRING,
    user_agent STRING)
ROW FORMAT SERDE 'org.apache.hadoop.hive.contrib.serde2.RegexSerDe'
WITH SERDEPROPERTIES (
    'input.regex' = '([^ ]*) - - \\[([^\\]]*)\\] "([^\ ]*) ([^\ ]*) ([^\ ]*)" (\\d*) (\\d*)
"([^"]*)" "([^"]*)"',
    'output.format.string' = "%1$$s %2$$s %3$$s %4$$s %5$$s %6$$s %7$$s
%8$$s %9$$s")
LOCATION '/user/hive/warehouse/original_access_logs';
```

```
CREATE EXTERNAL TABLE tokenized_access_logs (
    ip STRING,
    date STRING,
    method STRING,
    url STRING,
    http_version STRING,
    code1 STRING,
    code2 STRING,
    dash STRING,
    user_agent STRING)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
LOCATION '/user/hive/warehouse/tokenized_access_logs';
ADD JAR /usr/lib/hive/lib/hive-contrib.jar;

INSERT OVERWRITE TABLE tokenized_access_logs SELECT * FROM
intermediate_access_logs;
```

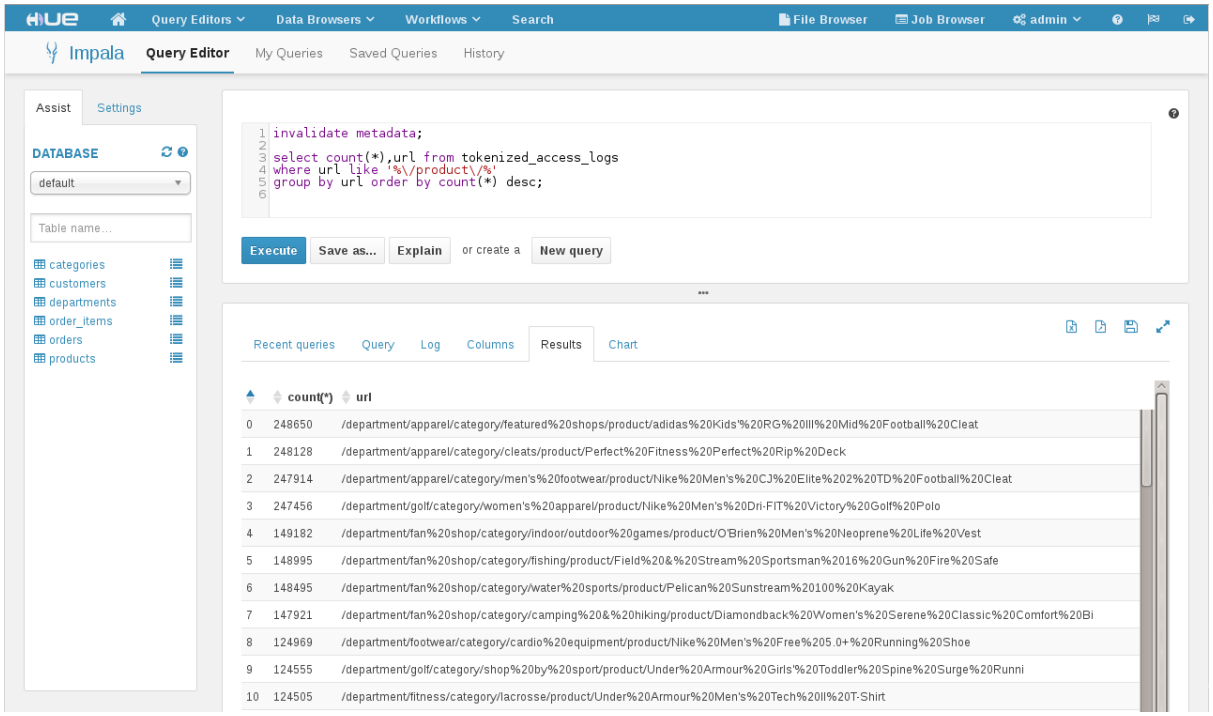
The final query will take a minute to run. It is using a MapReduce job, just like our Sqoop import did, to transfer the data from one table to the other in parallel. Again, we need to tell Impala that some tables have been created through a different tool. Switch back to the Impala Query Editor app, and enter the following command:

```
invalidate metadata;
```

Now, if you enter the 'show tables;' query or refresh the table list in the left-hand column, you should see the two new external tables in the default database. Paste the following query into the Query Editor:

```
select count(*),url from tokenized_access_logs
where url like '%\/product\/%'
group by url order by count(*) desc;
```

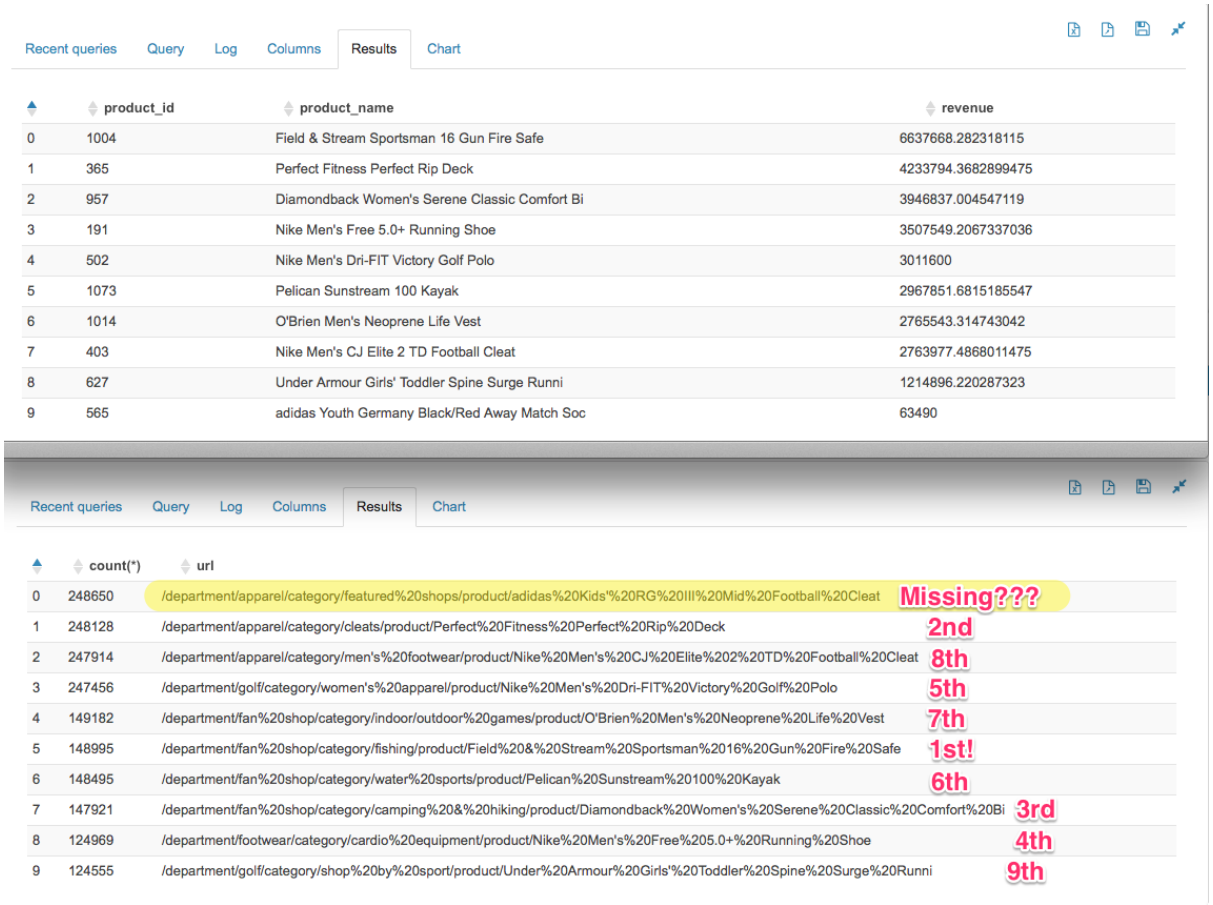
You should see a result similar to the following:



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

By introspecting the results you quickly realize that this list contains many of the products on the most sold list from previous tutorial steps, but there is one product that did not show up in the previous result. There is one product that seems to be viewed a lot, but

never purchased. Why?



Recent queries | Query | Log | Columns | Results | Chart

	product_id	product_name	revenue
0	1004	Field & Stream Sportsman 16 Gun Fire Safe	6637668.282318115
1	365	Perfect Fitness Perfect Rip Deck	4233794.3682899475
2	957	Diamondback Women's Serene Classic Comfort Bi	3946837.004547119
3	191	Nike Men's Free 5.0+ Running Shoe	3507549.2067337036
4	502	Nike Men's Dri-FIT Victory Golf Polo	3011600
5	1073	Pelican Sunstream 100 Kayak	2967851.6815185547
6	1014	O'Brien Men's Neoprene Life Vest	2765543.314743042
7	403	Nike Men's CJ Elite 2 TD Football Cleat	2763977.4868011475
8	627	Under Armour Girls' Toddler Spine Surge Runni	1214896.220287323
9	565	adidas Youth Germany Black/Red Away Match Soc	63490

Well, in our example with DataCo, once these odd findings are presented to your manager, it is immediately escalated. Eventually, someone figures out that on that view page, where most visitors stopped, the sales path of the product had a typo in the price for the item. Once the typo was fixed, and a correct price was displayed, the sales for that SKU started to rapidly increase.

CONCLUSION

If you hadn't had an efficient and interactive tool enabling analytics on high-volume semi-structured data, this loss of revenue would have been missed for a long time. There is risk of loss if an organization looks for answers within partial data. Correlating two data sets for the same business question showed value, and being able to do so within the same platform made life easier for you and for the organization.