Welcome!

Cloudera Quickstart Beginner Tutorial

Welcome to the Cloudera QuickStart VM tutorial! Following this tutorial will not only give you examples on how to get started with some of the tools provided in CDH (Cloudera's open source distribution including Apache Hadoop), but also give you a taste of what it means to ask bigger questions. By the end of this tutorial you will:

- Understand how to use some of the powerful tools in CDH
- Know how to setup and execute some basic business intelligence and analytics use cases
- Be able to explain to your manager why they need to give you a raise!

Getting Started

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.

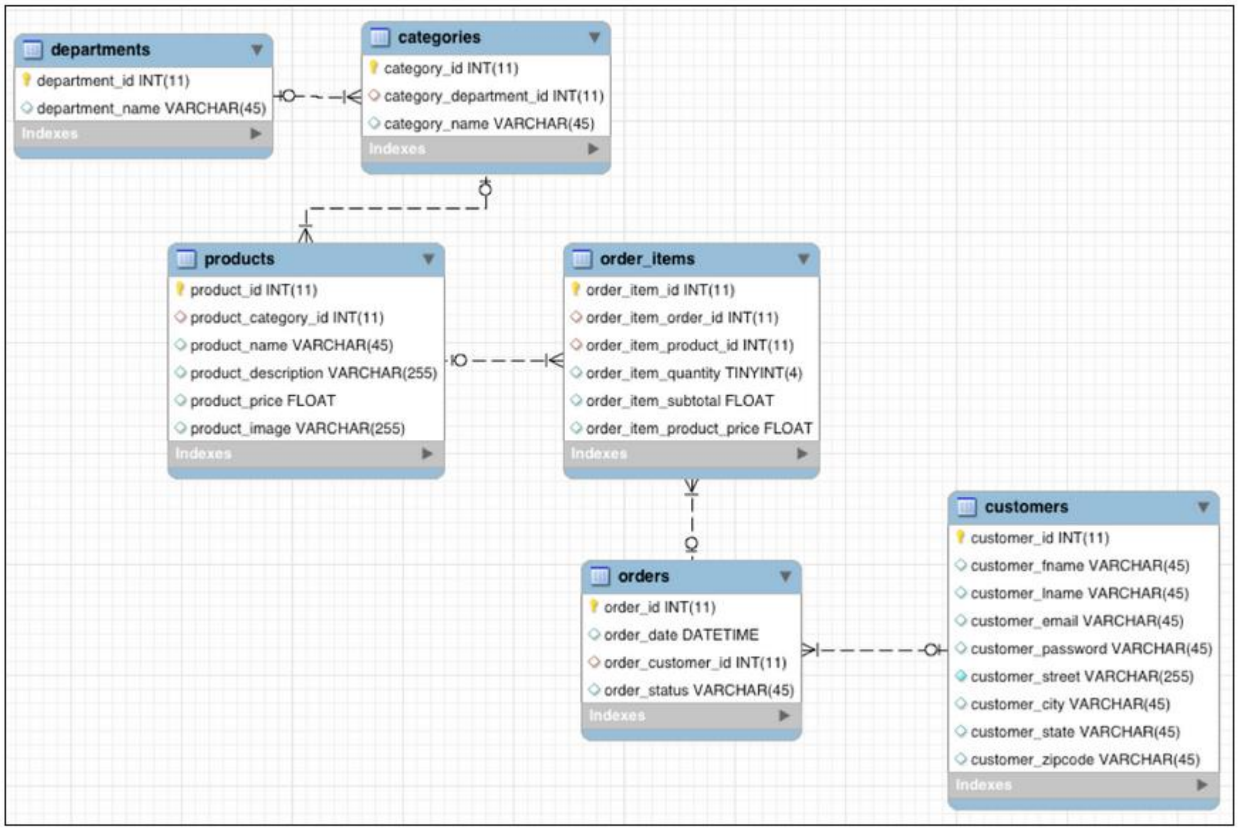
Tutorial Exercise 1

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.


departments
department_id INT(11)
department_name VARCHAR(45)
Indexes
categories
category_id INT(11)
category_department_id INT(11)
category_name VARCHAR(45)
Indexes
products
product_id INT(11)
product_category_id INT(11)
product_name VARCHAR(45)
product_description VARCHAR(255)
product_price FLOAT
product_image VARCHAR(255)
Indexes
order_items
order_item_id INT(11)
order_item_order_id INT(11)
order_item_product_id INT(11)
order_item_quantity TINYINT(4)
order_item_subtotal FLOAT
order_item_product_price FLOAT
Indexes
orders
order_id INT(11)
order_date DATETIME
order_customer_id INT(11)
order_status VARCHAR(45)
Indexes
customers
customer_id INT(11)
customer_fname VARCHAR(45)
customer_lname VARCHAR(45)
customer_email VARCHAR(45)
customer_password VARCHAR(45)
customer_street VARCHAR(255)
customer_city VARCHAR(45)
customer_state VARCHAR(45)
customer_zipcode VARCHAR(45)
Indexes

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.

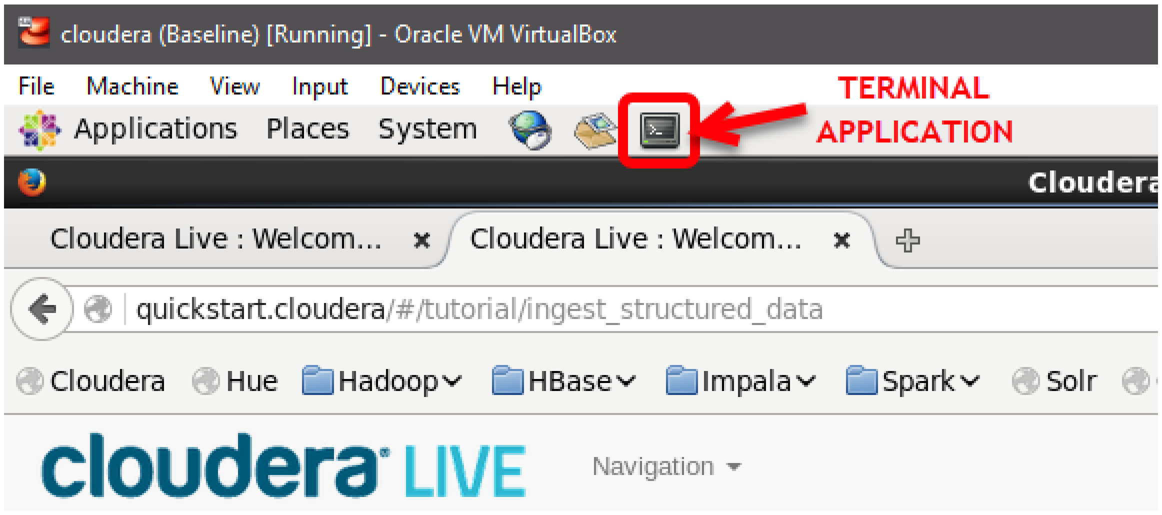
To analyze the transaction data in the new platform, we need to ingest it into the Hadoop Distributed File System (HDFS). We need to find a tool that easily transfers structured data from a RDBMS to HDFS, while preserving structure. That enables us to query the data, but not interfere with or break any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

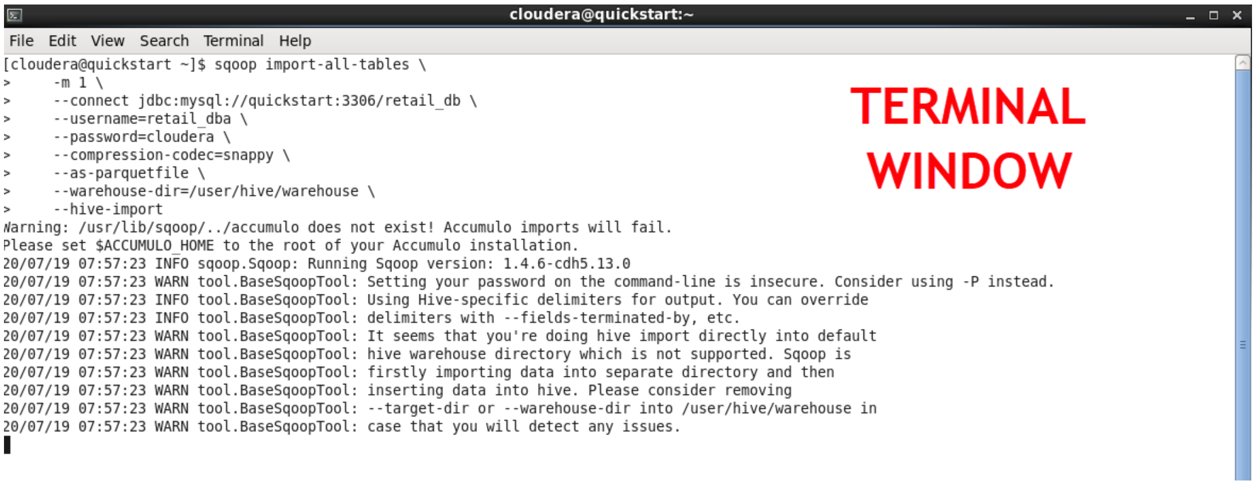
Apache Sqoop, which is part of CDH, is that tool. The nice thing about Sqoop is that we can automatically load our relational data from MySQL into HDFS, while preserving the structure.

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:

```
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
```


cloudera (Baseline) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System
TERMINAL APPLICATION
Cloudera Live : Welcom...
Cloudera Live : Welcom...
quickstart.cloudera/#/tutorial/ingest_structured_data
Cloudera Hue Hadoop HBase Impala Spark Solr
cloudera LIVE
Navigation



```
[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
Warning: /usr/lib/sqoop/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
20/07/19 07:57:23 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.13.0
20/07/19 07:57:23 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.
20/07/19 07:57:23 INFO tool.BaseSqoopTool: Using Hive-specific delimiters for output. You can override
20/07/19 07:57:23 INFO tool.BaseSqoopTool: delimiters with --fields-terminated-by, etc.
20/07/19 07:57:23 WARN tool.BaseSqoopTool: It seems that you're doing hive import directly into default
20/07/19 07:57:23 WARN tool.BaseSqoopTool: hive warehouse directory which is not supported. Sqoop is
20/07/19 07:57:23 WARN tool.BaseSqoopTool: firstly importing data into separate directory and then
20/07/19 07:57:23 WARN tool.BaseSqoopTool: inserting data into hive. Please consider removing
20/07/19 07:57:23 WARN tool.BaseSqoopTool: --target-dir or --warehouse-dir into /user/hive/warehouse in
20/07/19 07:57:23 WARN tool.BaseSqoopTool: case that you will detect any issues.
```

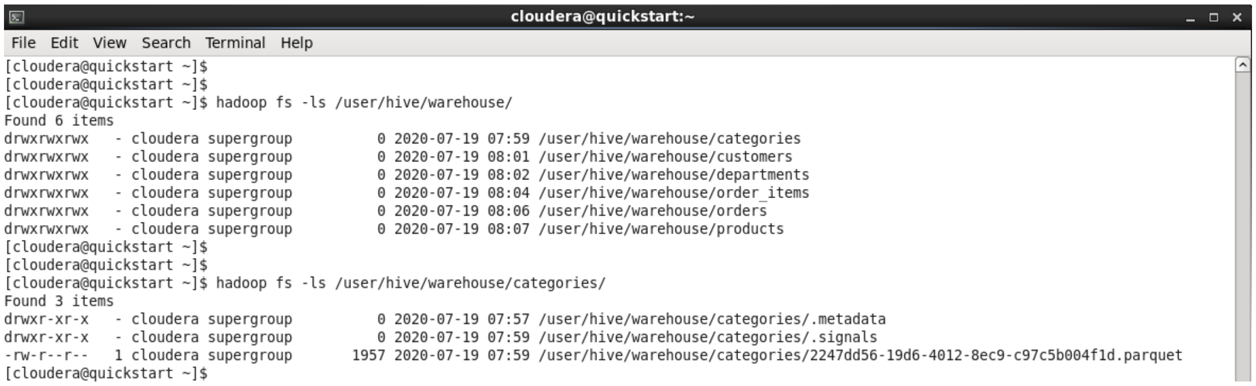
This command may take a while to complete, but it is doing a lot. It is launching MapReduce jobs to pull the data from our MySQL database and write the data to HDFS, distributed across the cluster in Apache Parquet format. It is also creating tables to represent the HDFS files in Impala / Apache Hive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

Once the Sqoop job is complete, we can confirm that our data was imported into HDFS via the following commands:

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

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

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



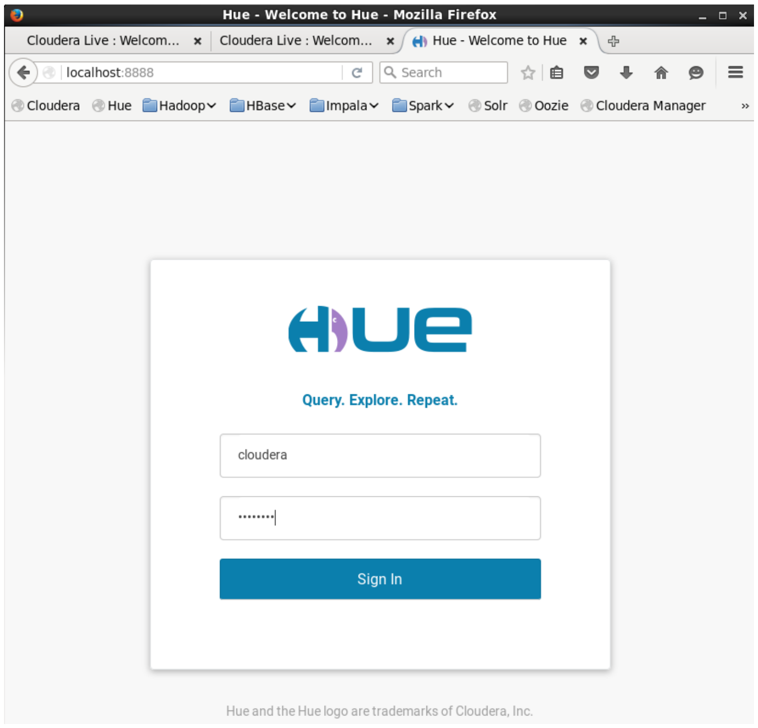
```
[cloudera@quickstart ~]$
[cloudera@quickstart ~]$
[cloudera@quickstart ~]$ hadoop fs -ls /user/hive/warehouse/
Found 6 items
drwxrwxrwx   - cloudera supergroup          0 2020-07-19 07:59 /user/hive/warehouse/categories
drwxrwxrwx   - cloudera supergroup          0 2020-07-19 08:01 /user/hive/warehouse/customers
drwxrwxrwx   - cloudera supergroup          0 2020-07-19 08:02 /user/hive/warehouse/departments
drwxrwxrwx   - cloudera supergroup          0 2020-07-19 08:04 /user/hive/warehouse/order_items
drwxrwxrwx   - cloudera supergroup          0 2020-07-19 08:06 /user/hive/warehouse/orders
drwxrwxrwx   - cloudera supergroup          0 2020-07-19 08:07 /user/hive/warehouse/products
[cloudera@quickstart ~]$
[cloudera@quickstart ~]$
[cloudera@quickstart ~]$ hadoop fs -ls /user/hive/warehouse/categories/
Found 3 items
drwxr-xr-x   - cloudera supergroup          0 2020-07-19 07:57 /user/hive/warehouse/categories/.metadata
drwxr-xr-x   - cloudera supergroup          0 2020-07-19 07:59 /user/hive/warehouse/categories/.signals
-rw-r--r--   1 cloudera supergroup       1957 2020-07-19 07:59 /user/hive/warehouse/categories/2247dd56-19d6-4012-8ec9-c97c5b004f1d.parquet
[cloudera@quickstart ~]$
```

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.

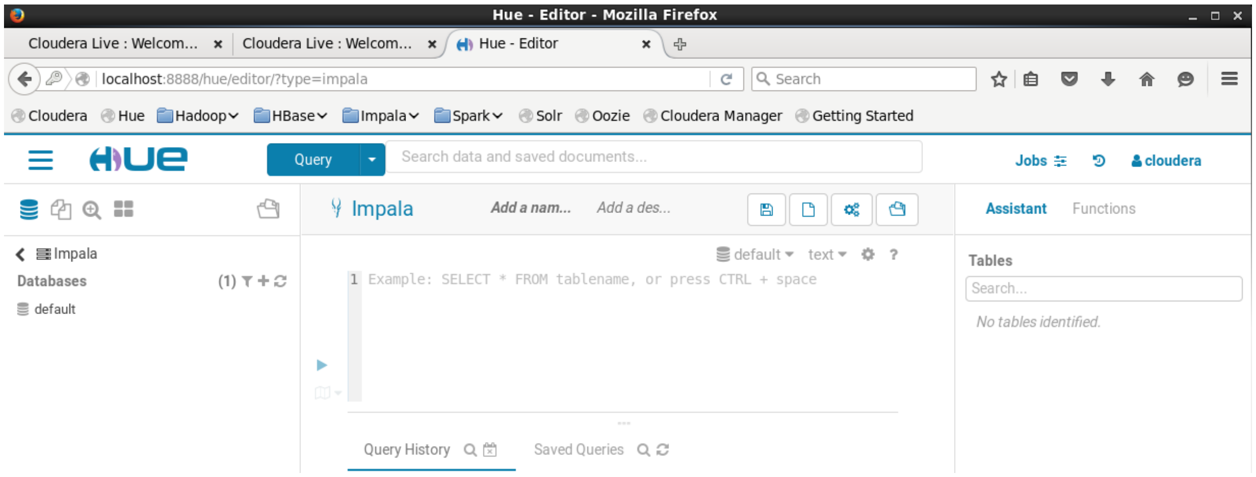
Hive and Impala also allow you to create tables by defining a schema over existing files with 'CREATE EXTERNAL TABLE' statements, similar to traditional relational databases. But Sqoop already created these tables for us, so we can go ahead and query them.

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 (localhost:8888).

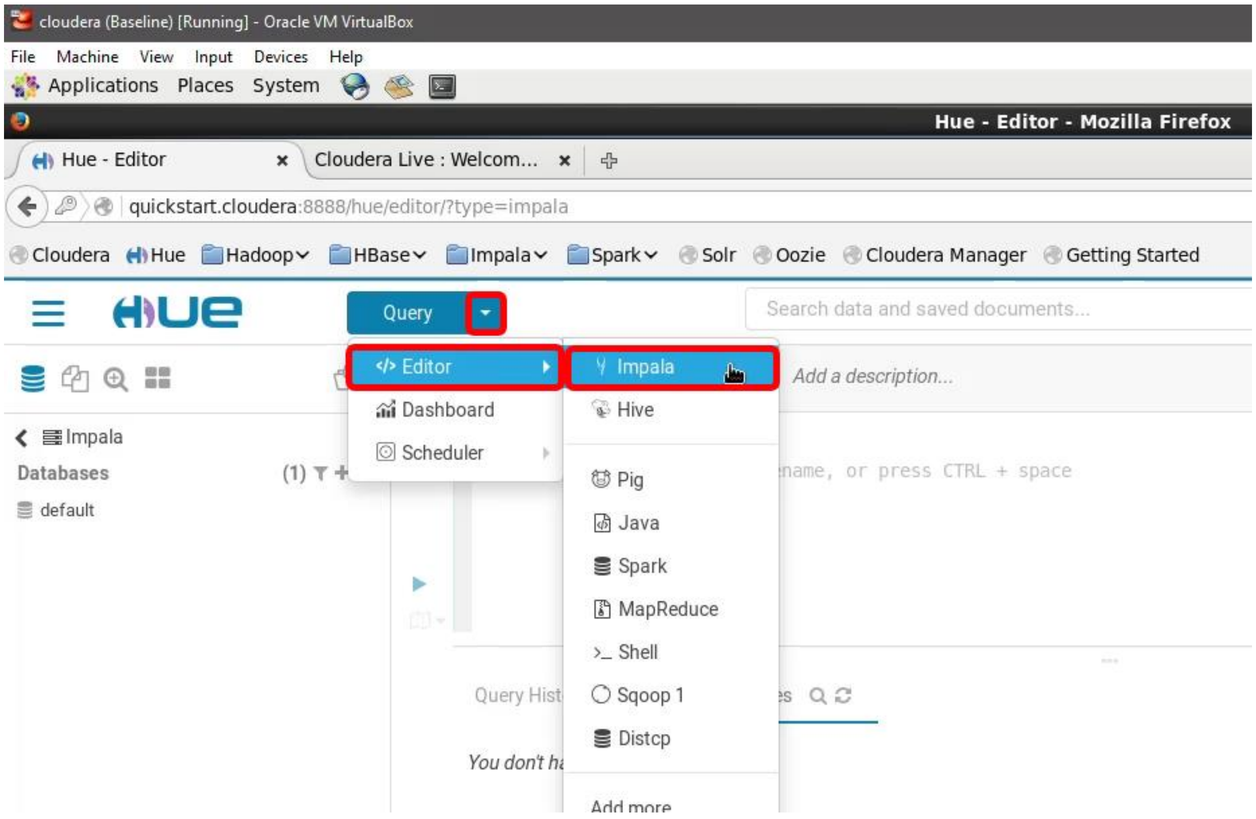
In the QuickStart VM, the administrator username for Hue is '**cloudera**' and the password is '**cloudera**'.


Hue - Welcome to Hue - Mozilla Firefox
localhost:8888
Query. Explore. Repeat.
cloudera
Sign In
Hue and the Hue logo are trademarks of Cloudera, Inc.

You'll see the Hue application screen after you log in.


Hue - Editor - Mozilla Firefox
localhost:8888/hue/editor/?type=impala
Query
Search data and saved documents...
Jobs
cloudera
Impala
Databases
default
Add a nam...
Add a des...
Assistant
Functions
Tables
No tables identified.
Example: SELECT * FROM tablename, or press CTRL + space
Query History
Saved Queries

Click on the down arrow (▾) next to the QUERY label at the top of the Hue screen. Then select **EDITOR→IMPALA** from the drop-down menus, as illustrated below.

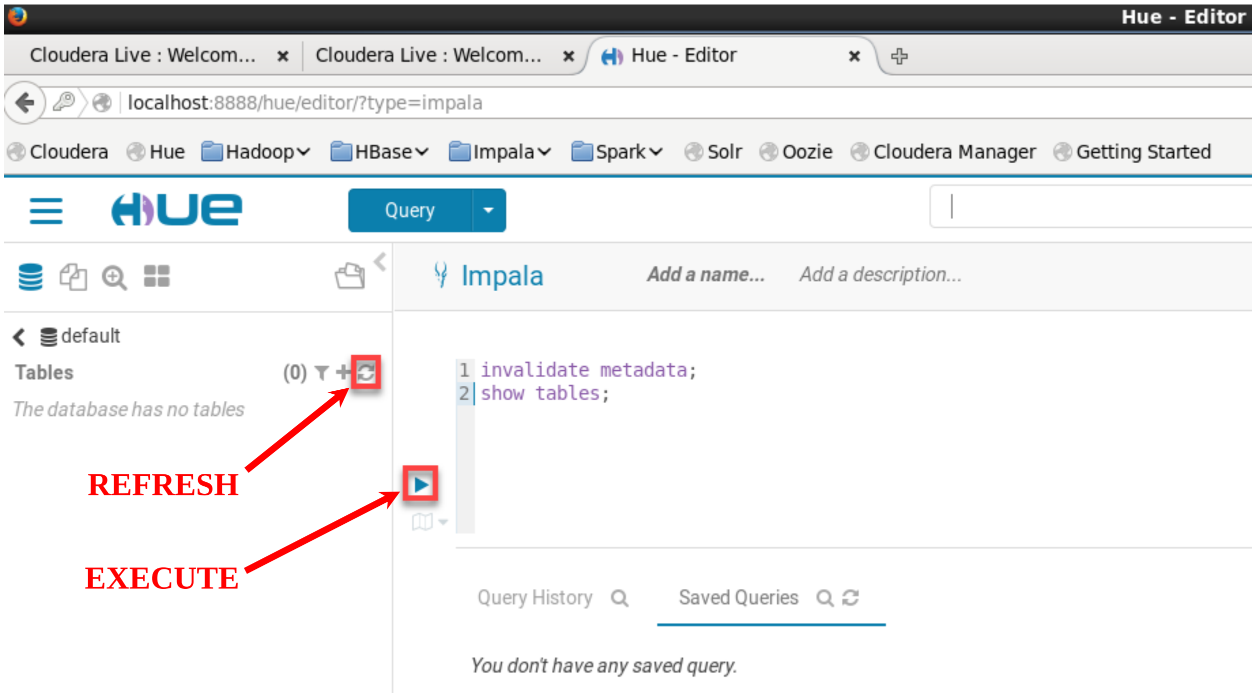

cloudera (Baseline) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System
Hue - Editor - Mozilla Firefox
Hue - Editor
Cloudera Live : Welcom...
quickstart.cloudera:8888/hue/editor/?type=impala
Cloudera Hue Hadoop HBase Impala Spark Solr Oozie Cloudera Manager Getting Started
HUE
Query
Search data and saved documents...
Editor
Dashboard
Scheduler
Impala
Hive
Pig
Java
Spark
MapReduce
Shell
Sqoop 1
Distcp
Add more
Add a description...
Impala
Databases
(1)
default
Query Hist
You don't ha

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;
```

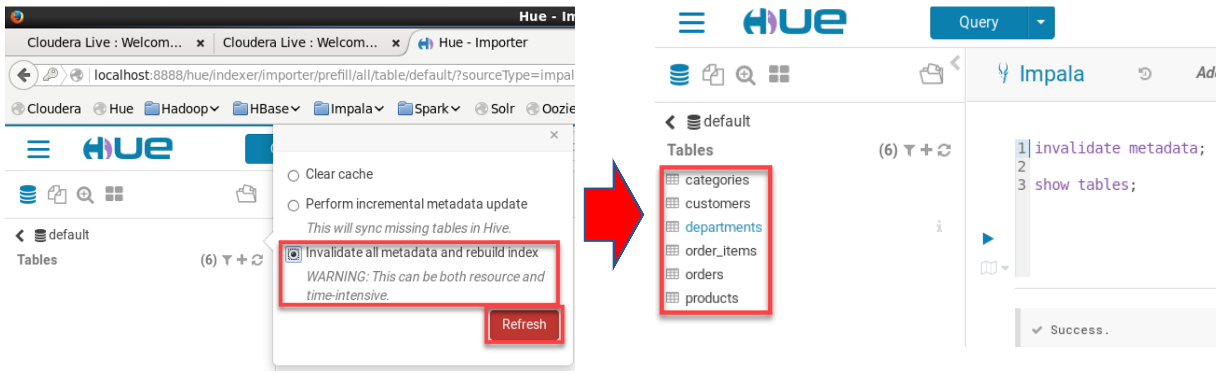
Click on the execute button (▶) to execute the above statements.

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



```
invalidate metadata;
show tables;
```

A prompt will appear after you click the refresh button, select the "Invalidate all metadata and rebuild index" option and the click the red REFRESH button.


Hue - Im
Cloudera Live : Welcom...
Cloudera Live : Welcom...
Hue - Importer
localhost:8888/hue/indexer/importer/prefill/all/table/default/?sourceType=impal
Cloudera
Hue
Hadoop
HBase
Impala
Spark
Solr
Oozie
default
Tables
(6)
Clear cache
Perform incremental metadata update
This will sync missing tables in Hive.
Invalidate all metadata and rebuild index
WARNING: This can be both resource and time-intensive.
Refresh
Query
Impala
default
Tables
(6)
categories
customers
departments
order_items
orders
products

```
invalidate metadata;

show tables;
```

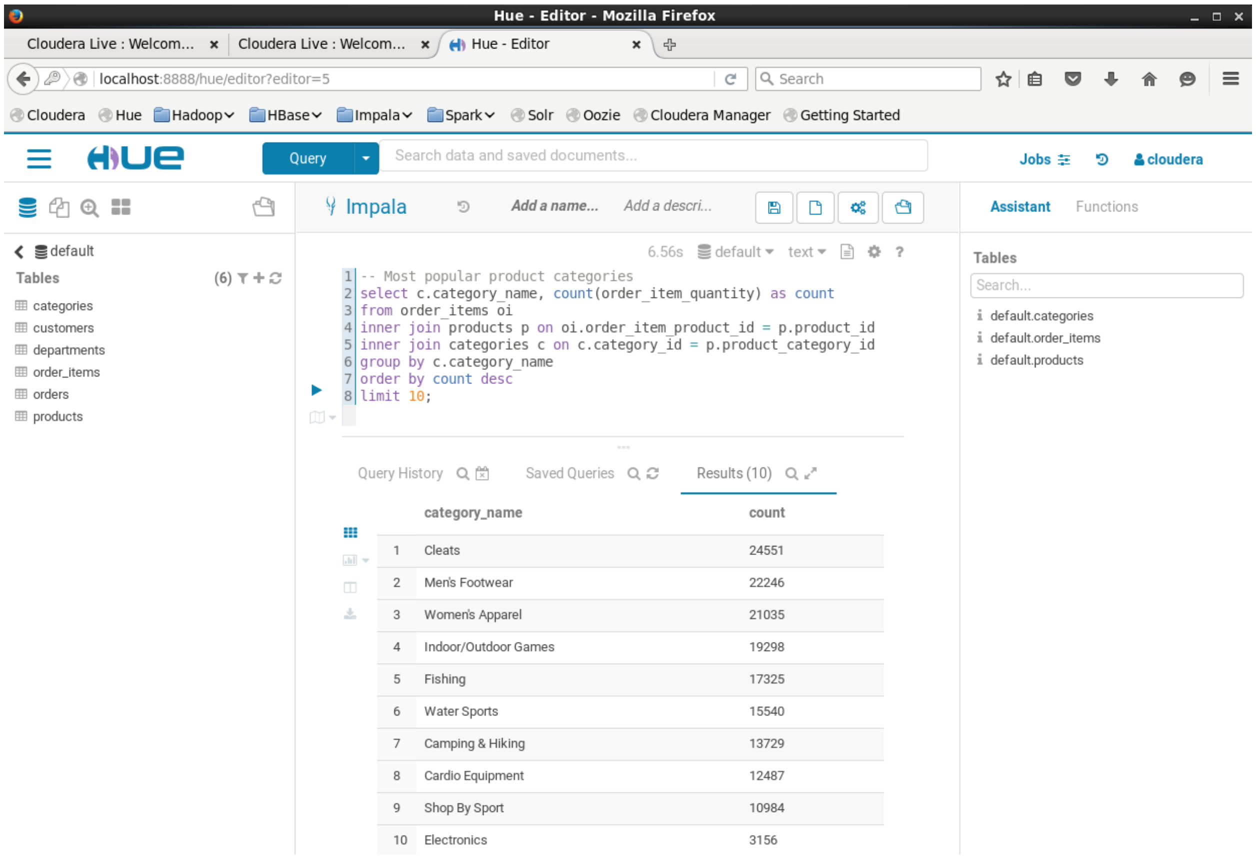
```
✔ Success.
```

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;
```

Click on the execute button (▶) to execute the above statement.

You should see results of the following form:



```
-- 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;
```

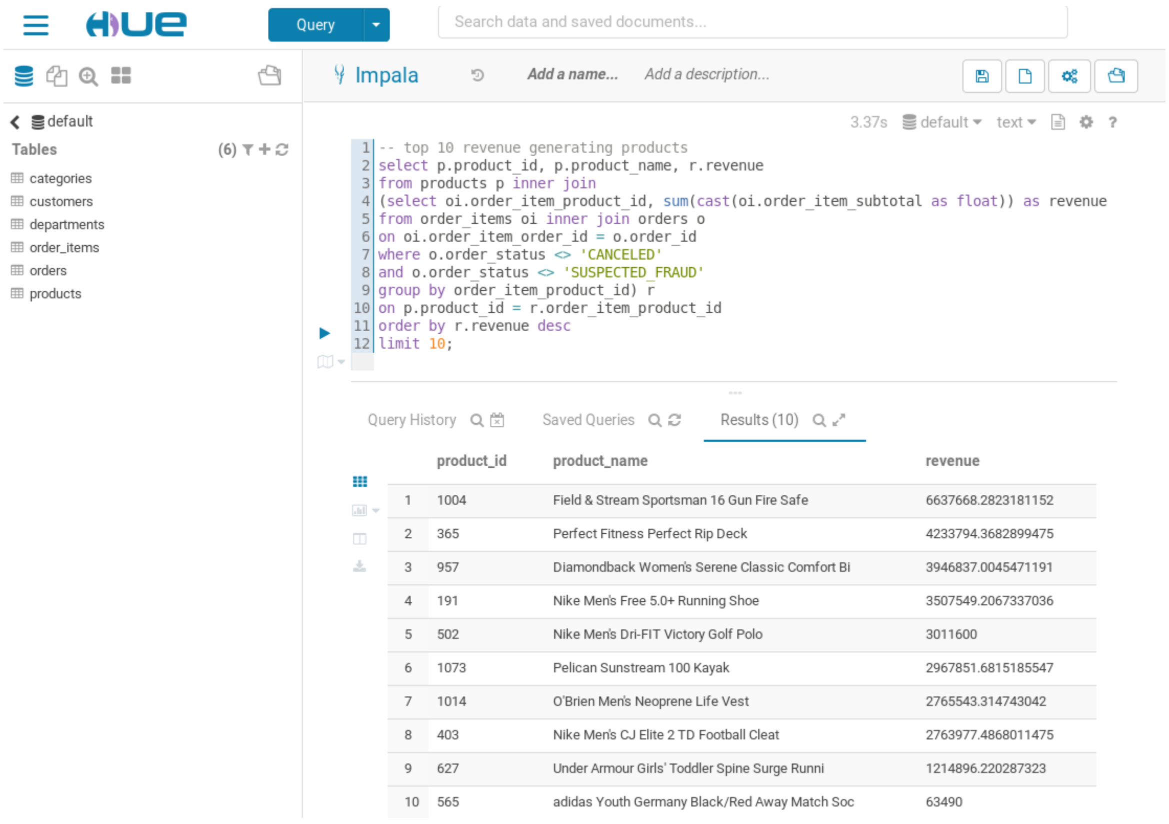
	category_name	count
1	Cleats	24551
2	Men's Footwear	22246
3	Women's Apparel	21035
4	Indoor/Outdoor Games	19298
5	Fishing	17325
6	Water Sports	15540
7	Camping & Hiking	13729
8	Cardio Equipment	12487
9	Shop By Sport	10984
10	Electronics	3156

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;
```

Click on the execute button (▶) to execute the above statement.

You should see results similar to this:



```
-- 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;
```

	product_id	product_name	revenue
1	1004	Field & Stream Sportsman 16 Gun Fire Safe	6637668.2823181152
2	365	Perfect Fitness Perfect Rip Deck	4233794.3682899475
3	957	Diamondback Women's Serene Classic Comfort Bi	3946837.0045471191
4	191	Nike Men's Free 5.0+ Running Shoe	3507549.2067337036
5	502	Nike Men's Dri-FIT Victory Golf Polo	3011600
6	1073	Pelican Sunstream 100 Kayak	2967851.6815185547
7	1014	O'Brien Men's Neoprene Life Vest	2765543.314743042
8	403	Nike Men's CJ Elite 2 TD Football Cleat	2763977.4868011475
9	627	Under Armour Girls' Toddler Spine Surge Runni	1214896.220287323
10	565	adidas Youth Germany Black/Red Away Match Soc	63490

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.

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.

Showing Big Data Value

Going a Step Beyond

Scenario:

Your Management: is indifferent, you produced what you always produce - a report on structured data, but you really didn't prove any additional value.

You: are either also indifferent and just go back to what you have always done... or you have an ace up your sleeve...

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 remodeling 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 into 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

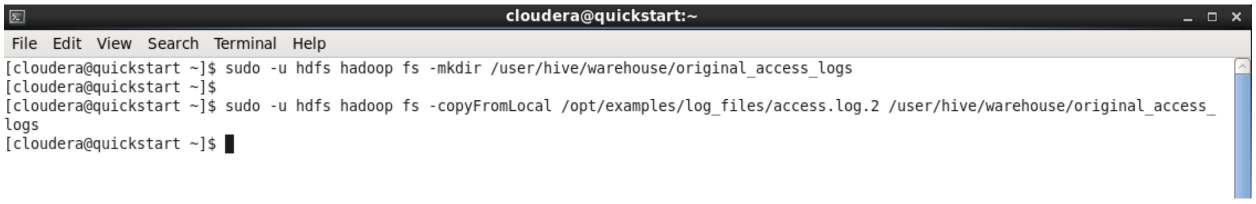
For convenience, we have loaded a sample (about 180K lines) of one month's worth of access log data into **`/opt/examples/log_data/access.log.2`**.

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

Go back to your Terminal window and execute the following commands from your Manager Node (i.e., the Cloudera Quickstart VM).

```
sudo -u hdfs hadoop fs -mkdir /user/hive/warehouse/original_access_logs

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.

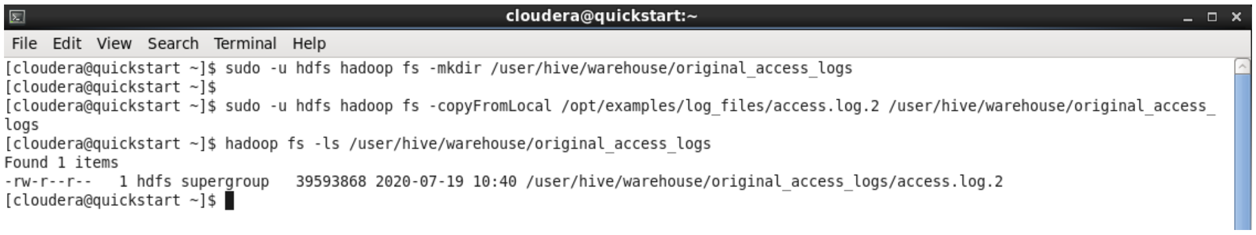


```
[cloudera@quickstart ~]$ sudo -u hdfs hadoop fs -mkdir /user/hive/warehouse/original_access_logs
[cloudera@quickstart ~]$
[cloudera@quickstart ~]$ sudo -u hdfs hadoop fs -copyFromLocal /opt/examples/log_files/access.log.2 /user/hive/warehouse/original_access_
logs
[cloudera@quickstart ~]$
```

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

```
hadoop fs -ls /user/hive/warehouse/original_access_logs
```

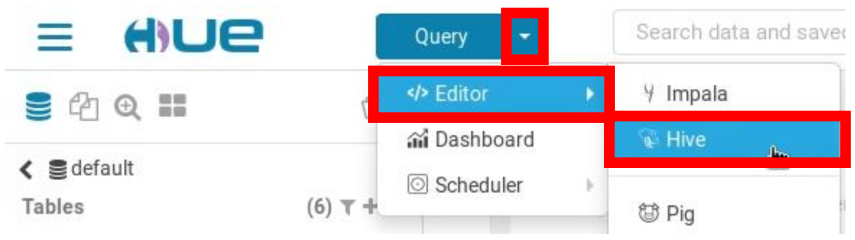
You should see a result similar to the following:



```
[cloudera@quickstart ~]$ sudo -u hdfs hadoop fs -mkdir /user/hive/warehouse/original_access_logs
[cloudera@quickstart ~]$
[cloudera@quickstart ~]$ sudo -u hdfs hadoop fs -copyFromLocal /opt/examples/log_files/access.log.2 /user/hive/warehouse/original_access_
logs
[cloudera@quickstart ~]$ hadoop fs -ls /user/hive/warehouse/original_access_logs
Found 1 items
-rw-r--r--   1 hdfs supergroup   39593868 2020-07-19 10:40 /user/hive/warehouse/original_access_logs/access.log.2
[cloudera@quickstart ~]$
```

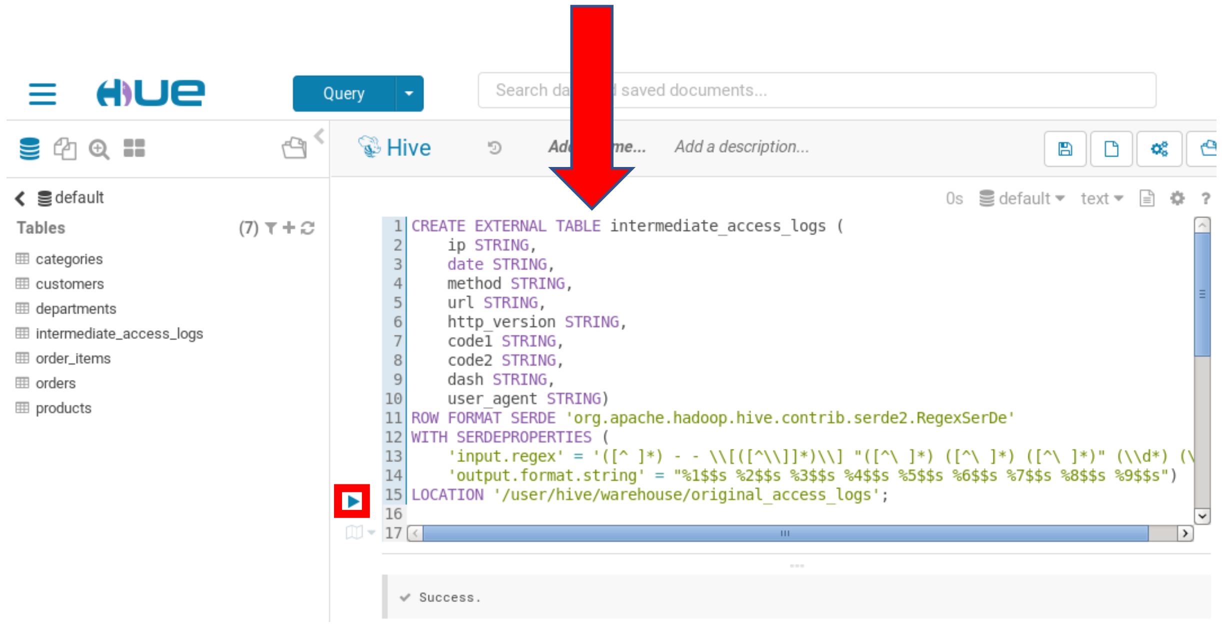
Now you can build a table in Hive and query the data via Impala and Hue. You'll build this table in 2 steps. First, you'll take advantage of Hive's flexible SerDes (serializers / deserializers) to parse the logs into individual fields using a regular expression. Second, you'll transfer the data from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

Click on the down arrow (▾) next to the QUERY label at the top of the Hue screen. Then select **EDITOR→HIVE** from the drop-down menus, as illustrated below.



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

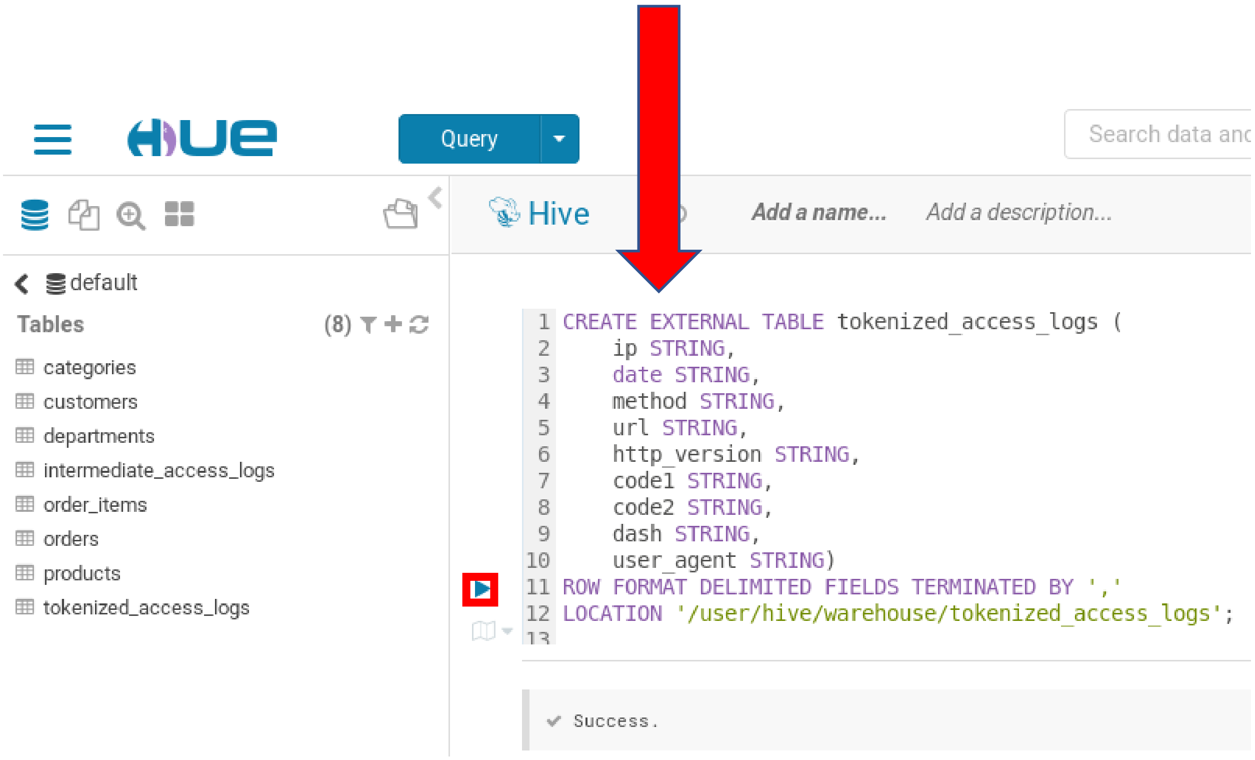
```
CREATE EXTERNAL TABLE intermediate_access_logs (
    ip STRING,
    date STRING,
    method STRING,
    url STRING,
    http_version STRING,
    code1 STRING,
    code2 STRING,
    dash STRING,
    user_agent STRING)
ROW FORMAT SERDE 'org.apache.hadoop.hive.contrib.serde2.RegexSerDe'
WITH SERDEPROPERTIES (
    '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 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*) (\
    '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';
```

```
✔ Success.
```

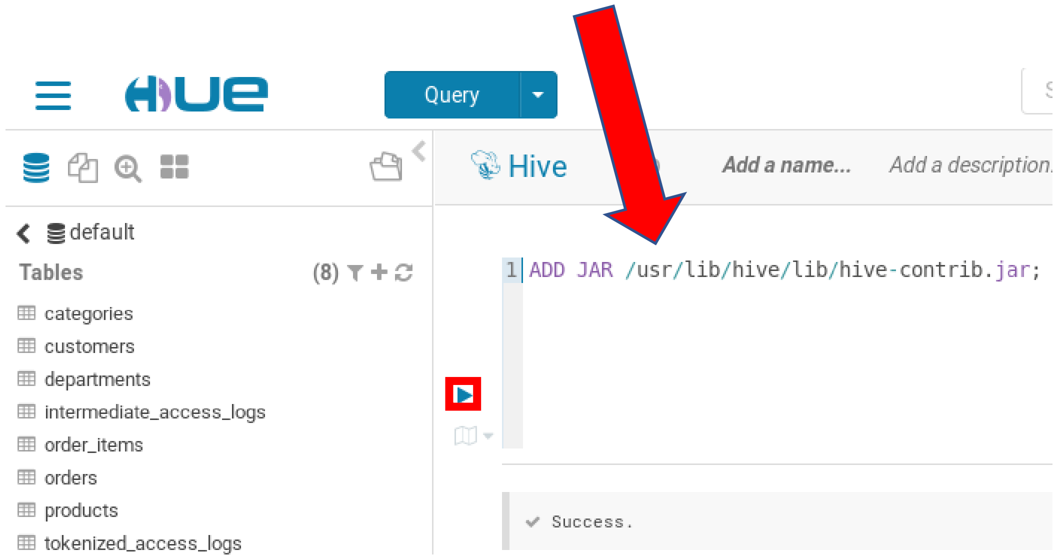
```
CREATE EXTERNAL TABLE tokenized_access_logs (
    ip STRING,
    date STRING,
    method STRING,
    url STRING,
    http_version STRING,
    code1 STRING,
    code2 STRING,
    dash STRING,
    user_agent STRING)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
LOCATION '/user/hive/warehouse/tokenized_access_logs';
```



```
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';
```

```
✔ Success.
```

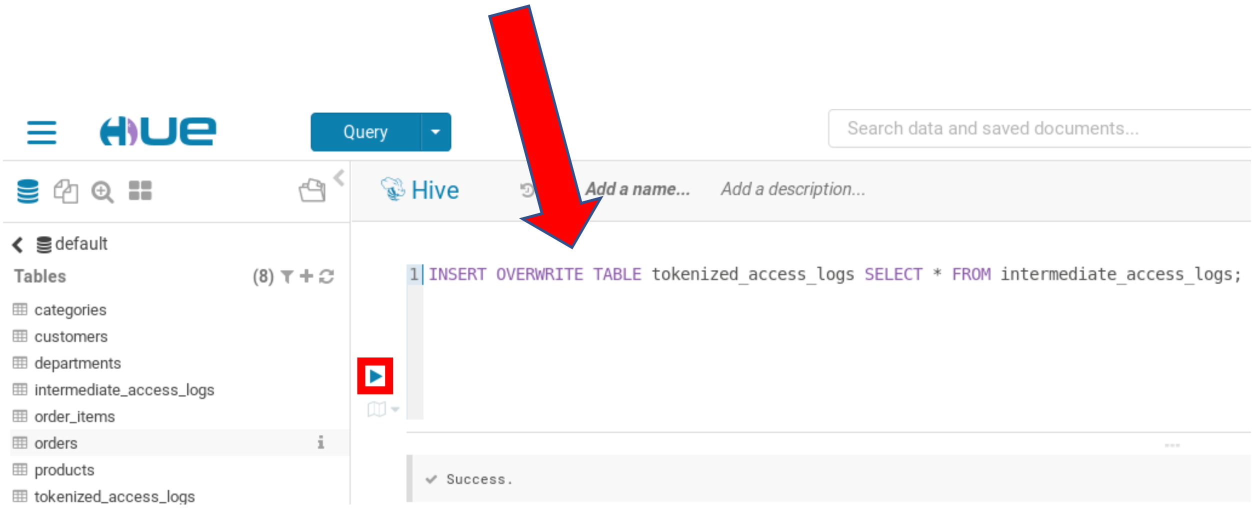
```
ADD JAR /usr/lib/hive/lib/hive-contrib.jar;
```



```
ADD JAR /usr/lib/hive/lib/hive-contrib.jar;
```

```
Success.
```

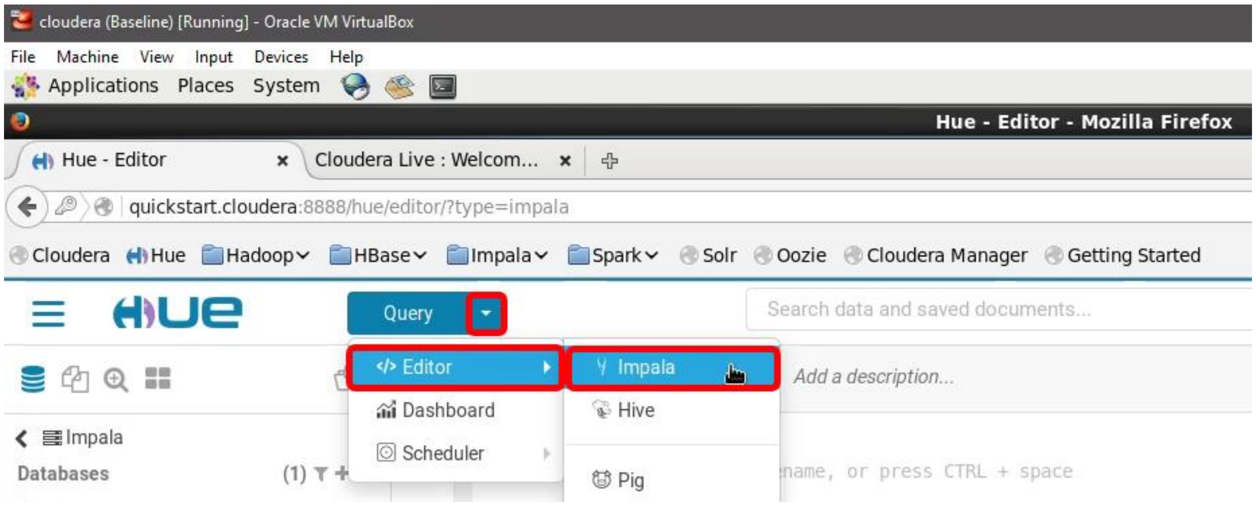
```
INSERT OVERWRITE TABLE tokenized_access_logs
SELECT * FROM intermediate_access_logs;
```



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

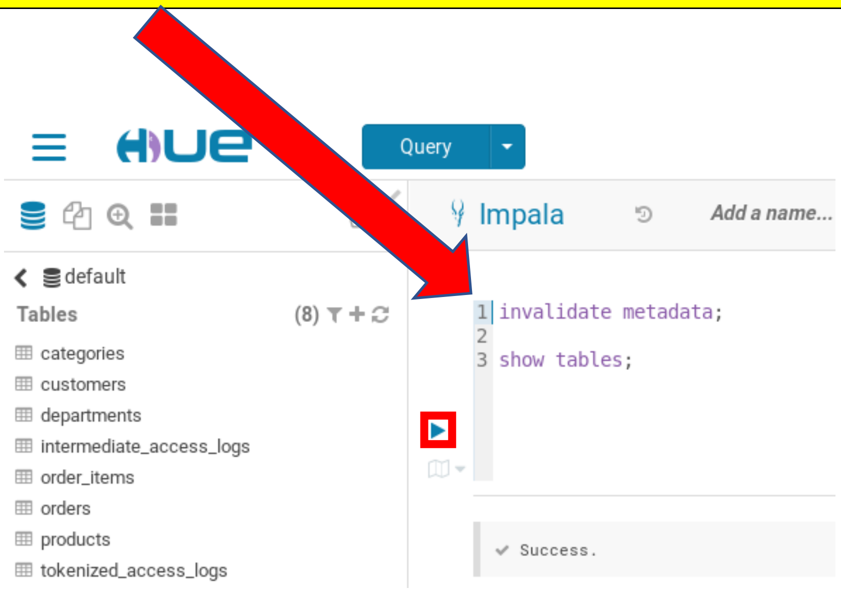
```
Success.
```

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 by clicking on the down arrow (▾) next to the QUERY label at the top of the Hue screen. Then select **EDITOR→IMPALA** from the drop-down menus, as illustrated below.


cloudera (Baseline) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System
Hue - Editor - Mozilla Firefox
Hue - Editor
Cloudera Live : Welcom...
quickstart.cloudera:8888/hue/editor/?type=impala
Cloudera Hue Hadoop HBase Impala Spark Solr Oozie Cloudera Manager Getting Started
Query
Search data and saved documents...
Editor
Impala
Add a description...
Dashboard
Hive
Scheduler
Pig
Impala
Databases (1)
name, or press CTRL + space

Enter the following commands:

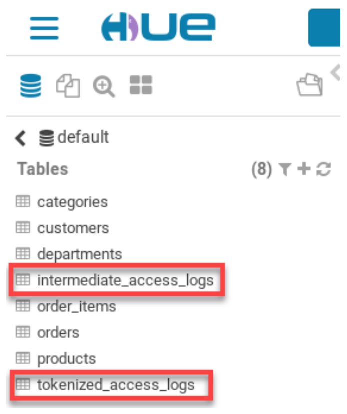
```
invalidate metadata;

show tables;
```



```
invalidate metadata;

show tables;
```

```
Success.
```

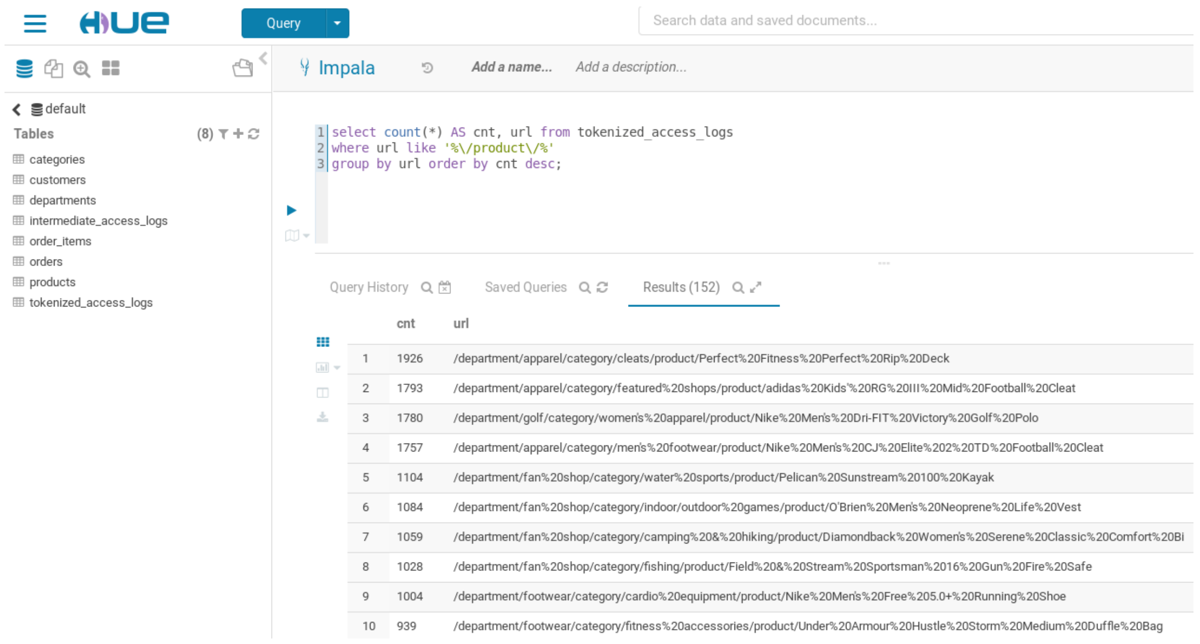
You should see the two new external tables in the default database.



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;
```

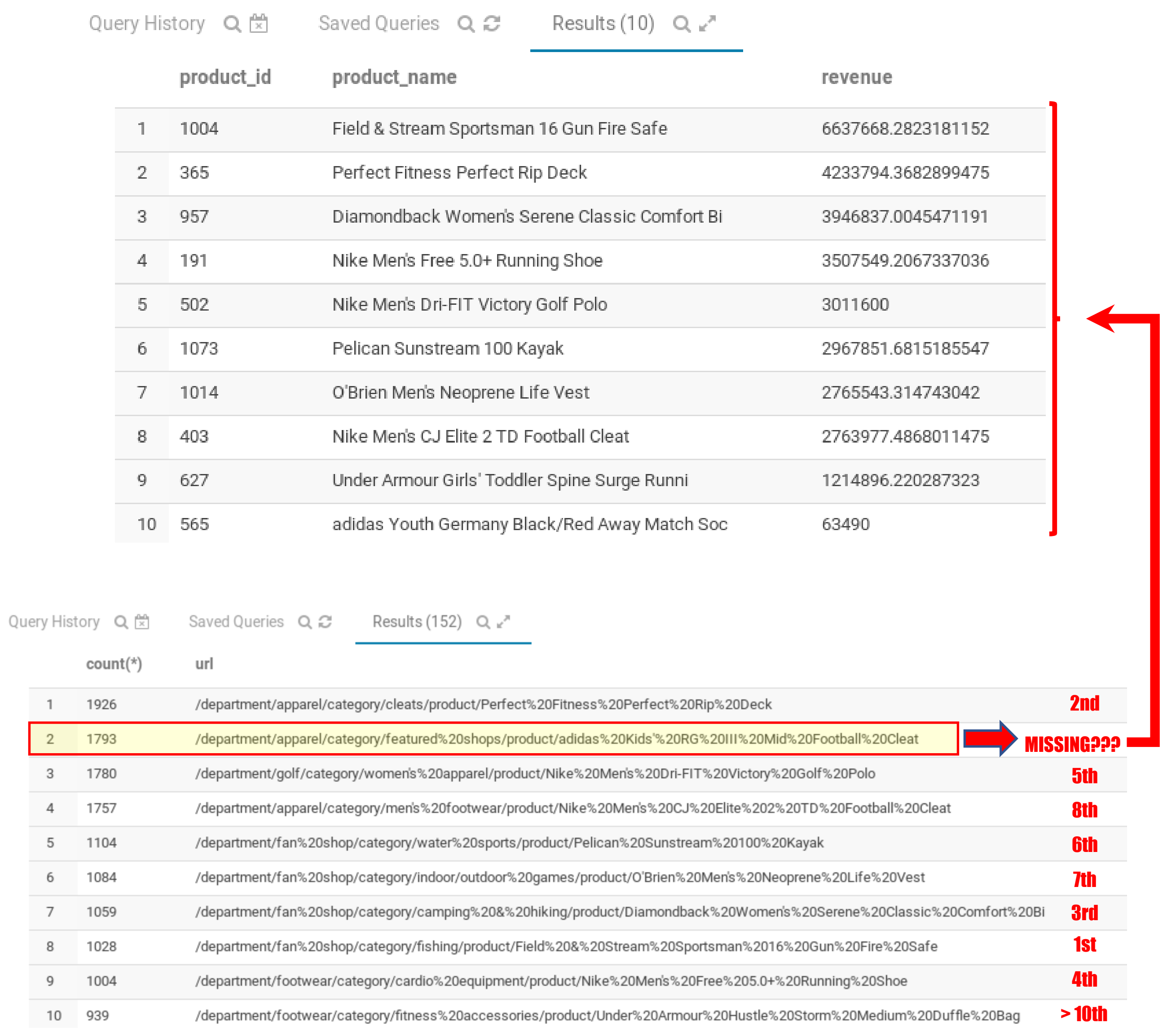
You should see a result similar to the following:



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

	cnt	url
1	1926	/department/apparel/category/cleats/product/Perfect%20Fitness%20Perfect%20Rip%20Deck
2	1793	/department/apparel/category/featured%20shops/product/adidas%20Kids'%20RG%20III%20Mid%20Football%20Cleat
3	1780	/department/golf/category/women's%20apparel/product/Nike%20Men's%20Dri-FIT%20Victory%20Golf%20Polo
4	1757	/department/apparel/category/men's%20footwear/product/Nike%20Men's%20CJ%20Elite%202%20TD%20Football%20Cleat
5	1104	/department/fan%20shop/category/water%20sports/product/Pelican%20Sunstream%20100%20Kayak
6	1084	/department/fan%20shop/category/indoor/outdoor%20games/product/O'Brien%20Men's%20Neoprene%20Life%20Vest
7	1059	/department/fan%20shop/category/camping%20&%20hiking/product/Diamondback%20Women's%20Serene%20Classic%20Comfort%20Bi
8	1028	/department/fan%20shop/category/fishing/product/Field%20&%20Stream%20Sportsman%2016%20Gun%20Fire%20Safe
9	1004	/department/footwear/category/cardio%20equipment/product/Nike%20Men's%20Free%205.0+%20Running%20Shoe
10	939	/department/footwear/category/fitness%20accessories/product/Under%20Armour%20Hustle%20Storm%20Medium%20Duffle%20Bag

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?



	product_id	product_name	revenue
1	1004	Field & Stream Sportsman 16 Gun Fire Safe	6637668.2823181152
2	365	Perfect Fitness Perfect Rip Deck	4233794.3682899475
3	957	Diamondback Women's Serene Classic Comfort Bi	3946837.0045471191
4	191	Nike Men's Free 5.0+ Running Shoe	3507549.2067337036
5	502	Nike Men's Dri-FIT Victory Golf Polo	3011600
6	1073	Pelican Sunstream 100 Kayak	2967851.6815185547
7	1014	O'Brien Men's Neoprene Life Vest	2765543.314743042
8	403	Nike Men's CJ Elite 2 TD Football Cleat	2763977.4868011475
9	627	Under Armour Girls' Toddler Spine Surge Runni	1214896.220287323
10	565	adidas Youth Germany Black/Red Away Match Soc	63490

	count(*)	url	
1	1926	/department/apparel/category/cleats/product/Perfect%20Fitness%20Perfect%20Rip%20Deck	2nd
2	1793	/department/apparel/category/featured%20shops/product/adidas%20Kids'%20RG%20III%20Mid%20Football%20Cleat	MISSING???
3	1780	/department/golf/category/women's%20apparel/product/Nike%20Men's%20Dri-FIT%20Victory%20Golf%20Polo	5th
4	1757	/department/apparel/category/men's%20footwear/product/Nike%20Men's%20CJ%20Elite%202%20TD%20Football%20Cleat	8th
5	1104	/department/fan%20shop/category/water%20sports/product/Pelican%20Sunstream%20100%20Kayak	6th
6	1084	/department/fan%20shop/category/indoor/outdoor%20games/product/O'Brien%20Men's%20Neoprene%20Life%20Vest	7th
7	1059	/department/fan%20shop/category/camping%20&%20hiking/product/Diamondback%20Women's%20Serene%20Classic%20Comfort%20Bi	3rd
8	1028	/department/fan%20shop/category/fishing/product/Field%20&%20Stream%20Sportsman%2016%20Gun%20Fire%20Safe	1st
9	1004	/department/footwear/category/cardio%20equipment/product/Nike%20Men's%20Free%205.0+%20Running%20Shoe	4th
10	939	/department/footwear/category/fitness%20accessories/product/Under%20Armour%20Hustle%20Storm%20Medium%20Duffle%20Bag	> 10th

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.

Advanced Analytics in the Same Platform

Scenario:

Your Management: is of course thrilled with the recent discoveries you helped them with - you basically saved them a lot of money! They start giving you bigger questions, and more funding (we really hope the latter!)

You: are excited to dive into more advanced use cases, but you know that you'll need even more funding by the organization. You decide to really show off!

Note: Advanced analytics is what makes the difference in a very competitive market. It is what the industry is drooling about. Spark is an excellent tool to perform more advanced data processing: K-means, graph processing, and multi-step real-time heavy-duty ETL. If you are not familiar with these terms, there is plenty of documentation "on the internet"...

Tutorial Exercise 3

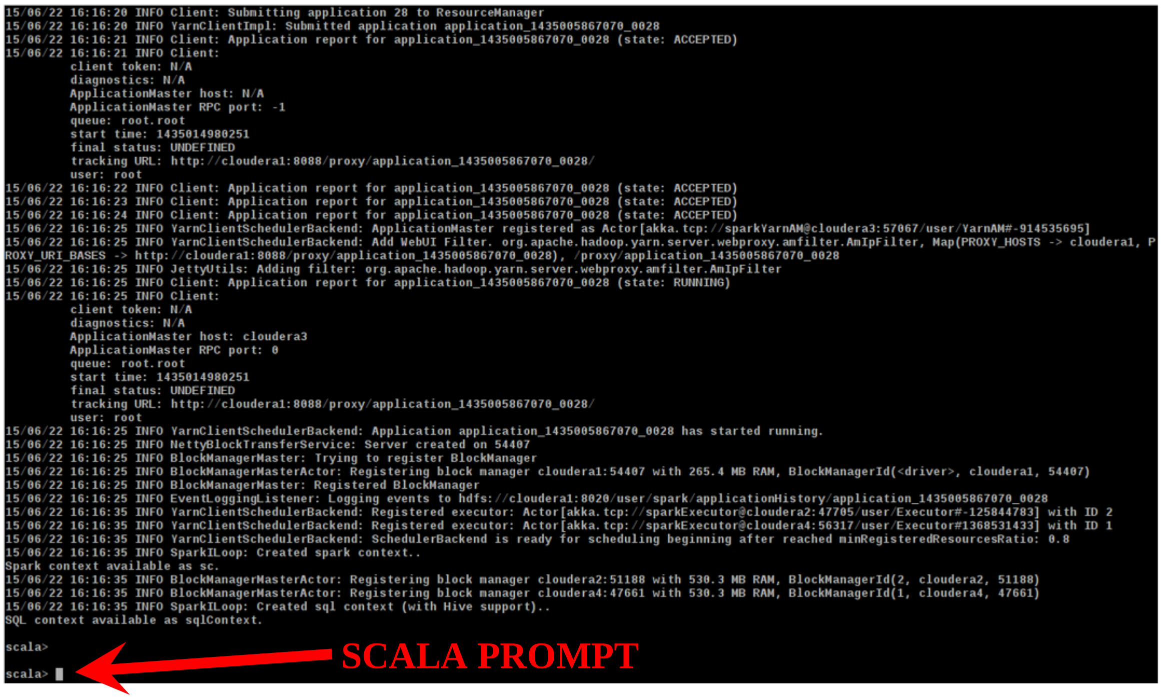
Relationship strength analytics using Spark

You come up with a great idea that it would be interesting for the marketing team which products are most commonly purchased together. Perhaps there are optimizations to be made in marketing campaigns to position components together that will generate a strong lead pipeline? Perhaps they can use product correlation data to help up sales for the lesser viewed products? Or recover revenue for the product that was on the top 10 viewed, but not top 10 sold from last exercise?

The tool in CDH best suited for quick analytics on object relationships is Apache Spark. You can compose a Spark job to do this work and give you insight on product relationships. Enter the following statement at the Linux Terminal prompt:

```
spark-shell --master yarn-client
```

Note: It may take a bit of time before the Scala prompt appears. If left alone for some time, the **scala** > prompt may become covered up by log messages from the cluster. Simply hit enter to refresh the prompt.

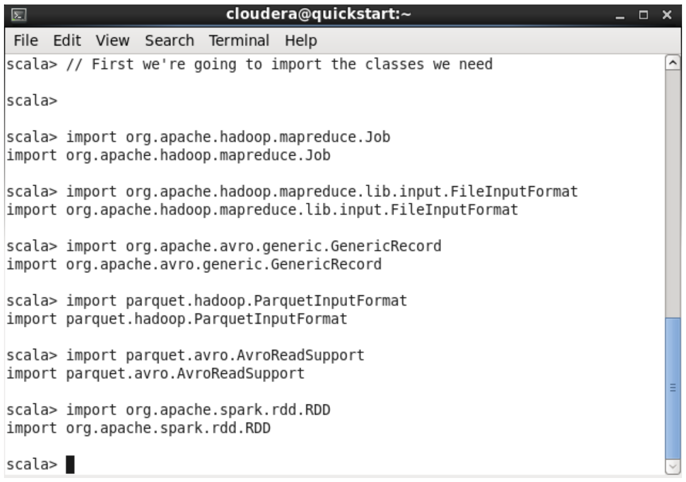


```
15/06/22 16:16:20 INFO Client: Submitting application 28 to ResourceManager
15/06/22 16:16:20 INFO YarnClientImpl: Submitted application application_1435005867070_0028
15/06/22 16:16:21 INFO Client: Application report for application_1435005867070_0028 (state: ACCEPTED)
15/06/22 16:16:21 INFO Client:
         client token: N/A
         diagnostics: N/A
         ApplicationMaster host: N/A
         ApplicationMaster RPC port: -1
         queue: root.root
         start time: 1435014980251
         final status: UNDEFINED
         tracking URL: http://cloudera1:8088/proxy/application_1435005867070_0028/
         user: root
15/06/22 16:16:22 INFO Client: Application report for application_1435005867070_0028 (state: ACCEPTED)
15/06/22 16:16:23 INFO Client: Application report for application_1435005867070_0028 (state: ACCEPTED)
15/06/22 16:16:24 INFO Client: Application report for application_1435005867070_0028 (state: ACCEPTED)
15/06/22 16:16:25 INFO YarnClientSchedulerBackend: ApplicationMaster registered as Actor[akka.tcp://sparkYarnAM@cloudera3:57067/user/YarnAM#-914535695]
15/06/22 16:16:25 INFO YarnClientSchedulerBackend: Add WebUI Filter. org.apache.hadoop.yarn.server.webproxy.amfilter.AmIpFilter, Map(PROXY_HOSTS -> cloudera1, P
ROXY_URI_BASES -> http://cloudera1:8088/proxy/application_1435005867070_0028), /proxy/application_1435005867070_0028
15/06/22 16:16:25 INFO JettyUtils: Adding filter: org.apache.hadoop.yarn.server.webproxy.amfilter.AmIpFilter
15/06/22 16:16:25 INFO Client: Application report for application_1435005867070_0028 (state: RUNNING)
15/06/22 16:16:25 INFO Client:
         client token: N/A
         diagnostics: N/A
         ApplicationMaster host: cloudera3
         ApplicationMaster RPC port: 0
         queue: root.root
         start time: 1435014980251
         final status: UNDEFINED
         tracking URL: http://cloudera1:8088/proxy/application_1435005867070_0028/
         user: root
15/06/22 16:16:25 INFO YarnClientSchedulerBackend: Application application_1435005867070_0028 has started running.
15/06/22 16:16:25 INFO NettyBlockTransferService: Server created on 54407
15/06/22 16:16:25 INFO BlockManagerMaster: Trying to register BlockManager
15/06/22 16:16:25 INFO BlockManagerMasterActor: Registering block manager cloudera1:54407 with 265.4 MB RAM, BlockManagerId(<driver>, cloudera1, 54407)
15/06/22 16:16:25 INFO BlockManagerMaster: Registered BlockManager
15/06/22 16:16:25 INFO EventLoggingListener: Logging events to hdfs://cloudera1:8020/user/spark/applicationHistory/application_1435005867070_0028
15/06/22 16:16:35 INFO YarnClientSchedulerBackend: Registered executor: Actor[akka.tcp://sparkExecutor@cloudera2:47705/user/Executor#-125844783] with ID 2
15/06/22 16:16:35 INFO YarnClientSchedulerBackend: Registered executor: Actor[akka.tcp://sparkExecutor@cloudera4:56317/user/Executor#1368531433] with ID 1
15/06/22 16:16:35 INFO YarnClientSchedulerBackend: SchedulerBackend is ready for scheduling beginning after reached minRegisteredResourcesRatio: 0.8
15/06/22 16:16:35 INFO SparkILoop: Created spark context..
Spark context available as sc.
15/06/22 16:16:35 INFO BlockManagerMasterActor: Registering block manager cloudera2:51188 with 530.3 MB RAM, BlockManagerId(2, cloudera2, 51188)
15/06/22 16:16:35 INFO BlockManagerMasterActor: Registering block manager cloudera4:47661 with 530.3 MB RAM, BlockManagerId(1, cloudera4, 47661)
15/06/22 16:16:35 INFO SparkILoop: Created sql context (with Hive support)..
SQL context available as sqlContext.

scala>

scala> █
```

Once the **scala>** prompt has appeared, paste the following batches of code:

```
// First we're going to import the classes we need

import org.apache.hadoop.mapreduce.Job
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat
import org.apache.avro.generic.GenericRecord
import parquet.hadoop.ParquetInputFormat
import parquet.avro.AvroReadSupport
import org.apache.spark.rdd.RDD
```

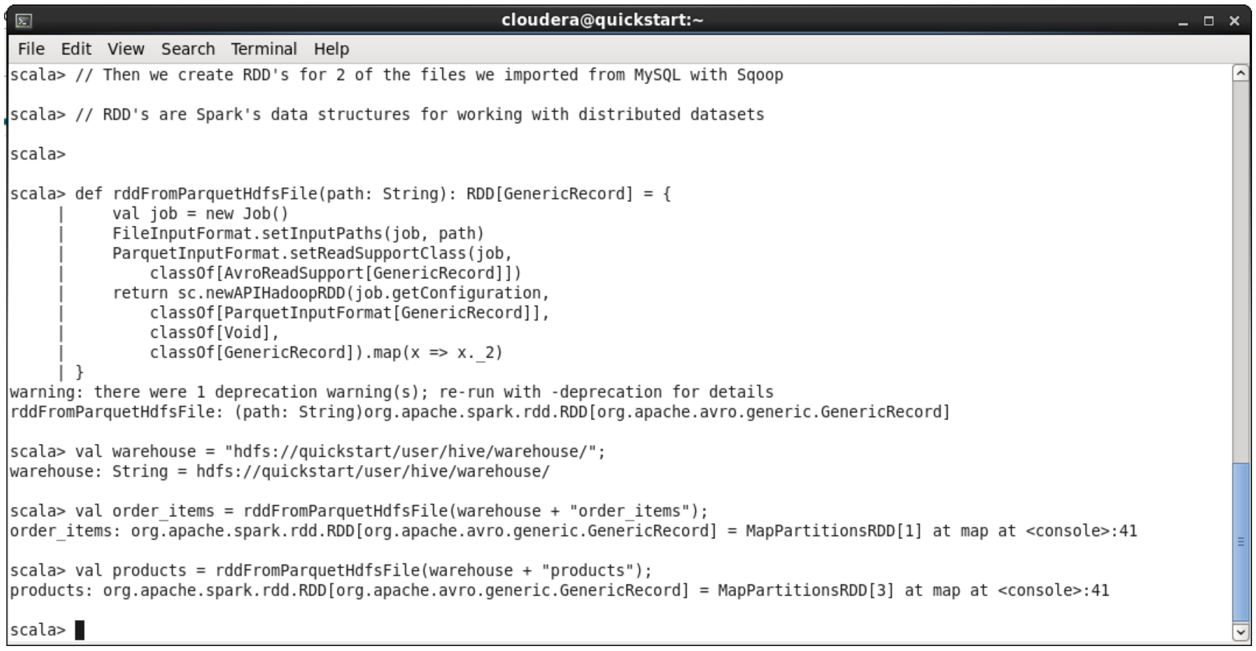
EXAMPLE EXECUTION & OUTPUT:



```
scala> // First we're going to import the classes we need

scala>

scala> import org.apache.hadoop.mapreduce.Job
import org.apache.hadoop.mapreduce.Job

scala> import org.apache.hadoop.mapreduce.lib.input.FileInputFormat
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat

scala> import org.apache.avro.generic.GenericRecord
import org.apache.avro.generic.GenericRecord

scala> import parquet.hadoop.ParquetInputFormat
import parquet.hadoop.ParquetInputFormat

scala> import parquet.avro.AvroReadSupport
import parquet.avro.AvroReadSupport

scala> import org.apache.spark.rdd.RDD
import org.apache.spark.rdd.RDD

scala> █
```

```
// Then we create RDD's for 2 of the files we imported from MySQL with Sqoop
// RDD's are Spark's data structures for working with distributed datasets

def rddFromParquetHdfsFile(path: String): RDD[GenericRecord] = {
    val job = new Job()
    FileInputFormat.setInputPaths(job, path)
    ParquetInputFormat.setReadSupportClass(job,
        classOf[AvroReadSupport[GenericRecord]])
    return sc.newAPIHadoopRDD(job.getConfiguration,
        classOf[ParquetInputFormat[GenericRecord]],
        classOf[Void],
        classOf[GenericRecord]).map(x => x._2)
}
val warehouse = "hdfs://quickstart/user/hive/warehouse/";
val order_items = rddFromParquetHdfsFile(warehouse + "order_items");
val products = rddFromParquetHdfsFile(warehouse + "products");
```

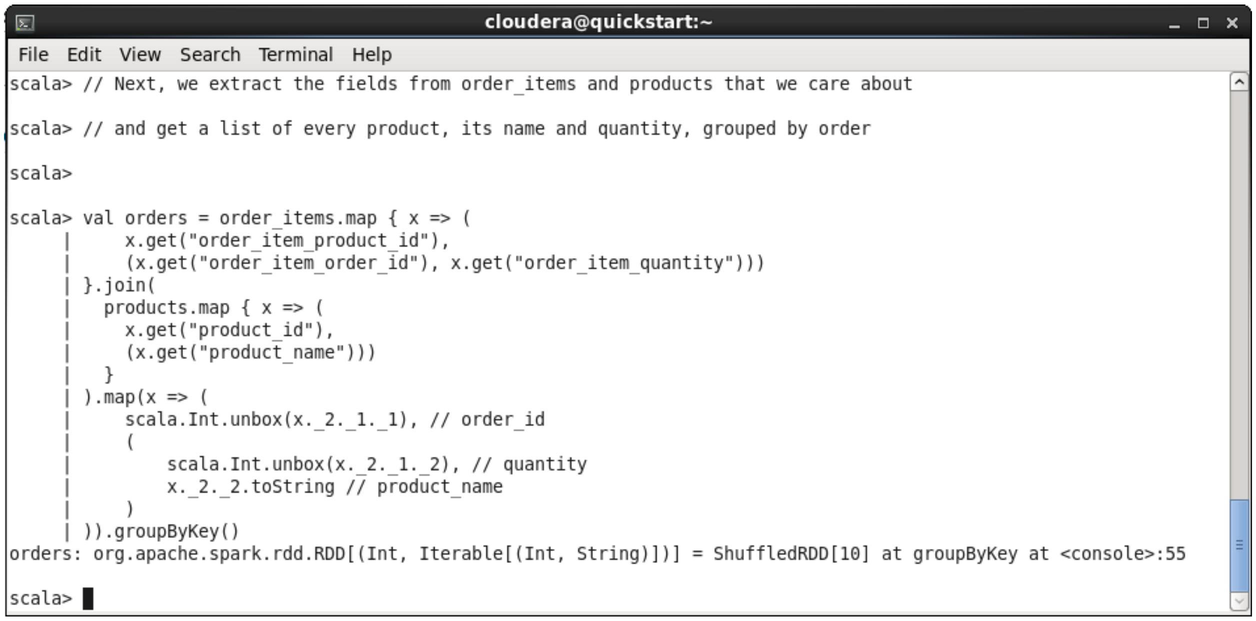
EXAMPLE EXECUTION & OUTPUT:



```
scala> // Then we create RDD's for 2 of the files we imported from MySQL with Sqoop

scala> // RDD's are Spark's data structures for working with distributed datasets

scala>

scala> def rddFromParquetHdfsFile(path: String): RDD[GenericRecord] = {
     |     val job = new Job()
     |     FileInputFormat.setInputPaths(job, path)
     |     ParquetInputFormat.setReadSupportClass(job,
     |         classOf[AvroReadSupport[GenericRecord]])
     |     return sc.newAPIHadoopRDD(job.getConfiguration,
     |         classOf[ParquetInputFormat[GenericRecord]],
     |         classOf[Void],
     |         classOf[GenericRecord]).map(x => x._2)
     | }
warning: there were 1 deprecation warning(s); re-run with -deprecation for details
rddFromParquetHdfsFile: (path: String)org.apache.spark.rdd.RDD[org.apache.avro.generic.GenericRecord]

scala> val warehouse = "hdfs://quickstart/user/hive/warehouse/";
warehouse: String = hdfs://quickstart/user/hive/warehouse/

scala> val order_items = rddFromParquetHdfsFile(warehouse + "order_items");
order_items: org.apache.spark.rdd.RDD[org.apache.avro.generic.GenericRecord] = MapPartitionsRDD[1] at map at <console>:41

scala> val products = rddFromParquetHdfsFile(warehouse + "products");
products: org.apache.spark.rdd.RDD[org.apache.avro.generic.GenericRecord] = MapPartitionsRDD[3] at map at <console>:41

scala> 
```

```
// Next, we extract the fields from order_items and products that we care about
// and get a list of every product, its name and quantity, grouped by order

val orders = order_items.map { x => (
    x.get("order_item_product_id"),
    (x.get("order_item_order_id"), x.get("order_item_quantity")))
}.join(
  products.map { x => (
    x.get("product_id"),
    (x.get("product_name")))
  }
).map(x => (
    scala.Int.unbox(x._2._1._1), // order_id
    (
        scala.Int.unbox(x._2._1._2), // quantity
        x._2._2.toString // product_name
    )
)).groupByKey()
```

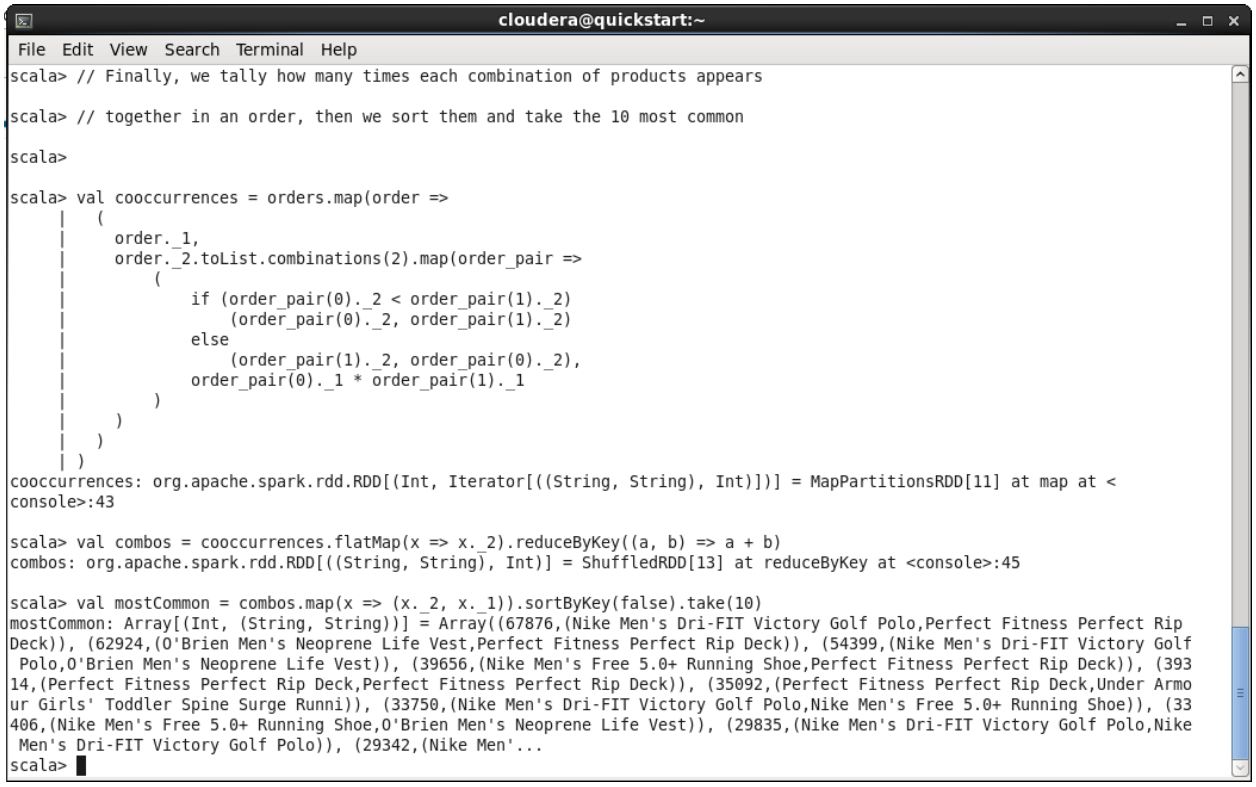
EXAMPLE EXECUTION & OUTPUT:



```
scala> // Next, we extract the fields from order_items and products that we care about

scala> // and get a list of every product, its name and quantity, grouped by order

scala>

scala> val orders = order_items.map { x => (
     |     x.get("order_item_product_id"),
     |     (x.get("order_item_order_id"), x.get("order_item_quantity")))
     | }.join(
     |   products.map { x => (
     |     x.get("product_id"),
     |     (x.get("product_name")))
     |   }
     | ).map(x => (
     |     scala.Int.unbox(x._2._1._1), // order_id
     |     (
     |         scala.Int.unbox(x._2._1._2), // quantity
     |         x._2._2.toString // product_name
     |     )
     | )).groupByKey()
orders: org.apache.spark.rdd.RDD[(Int, Iterable[(Int, String)])] = ShuffledRDD[10] at groupByKey at <console>:55

scala> 
```

```
// Finally, we tally how many times each combination of products appears
// together in an order, then we sort them and take the 10 most common

val cooccurrences = orders.map(order =>
  (
    order._1,
    order._2.toList.combinations(2).map(order_pair =>
          (
              if (order_pair(0)._2 < order_pair(1)._2)
                  (order_pair(0)._2, order_pair(1)._2)
              else
                  (order_pair(1)._2, order_pair(0)._2),
              order_pair(0)._1 * order_pair(1)._1
          )
      )
  )
)
val combos = cooccurrences.flatMap(x => x._2).reduceByKey((a, b) => a + b)
val mostCommon = combos.map(x => (x._2, x._1)).sortByKey(false).take(10)
```

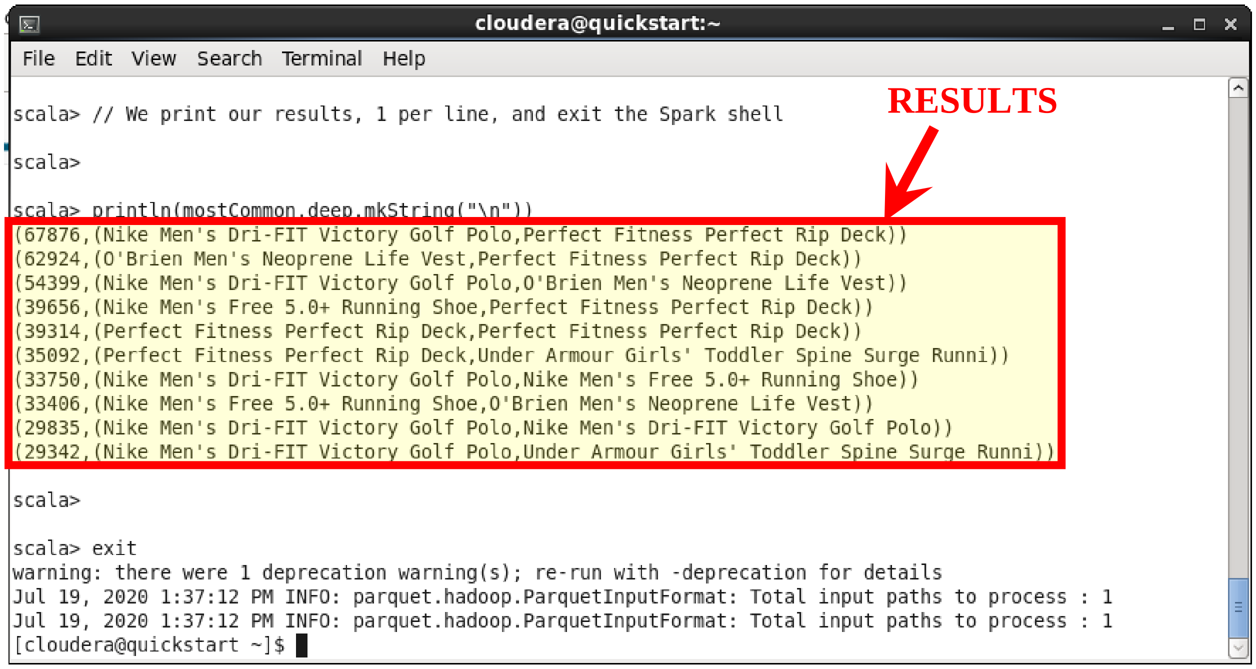
EXAMPLE EXECUTION & OUTPUT:



```
scala> // Finally, we tally how many times each combination of products appears

scala> // together in an order, then we sort them and take the 10 most common

scala>

scala> val cooccurrences = orders.map(order =>
     |    (
     |      order._1,
     |      order._2.toList.combinations(2).map(order_pair =>
     |          (
     |              if (order_pair(0)._2 < order_pair(1)._2)
     |                  (order_pair(0)._2, order_pair(1)._2)
     |              else
     |                  (order_pair(1)._2, order_pair(0)._2),
     |              order_pair(0)._1 * order_pair(1)._1
     |          )
     |      )
     |    )
     | )
cooccurrences: org.apache.spark.rdd.RDD[(Int, Iterator[((String, String), Int)])] = MapPartitionsRDD[11] at map at <
console>:43

scala> val combos = cooccurrences.flatMap(x => x._2).reduceByKey((a, b) => a + b)
combos: org.apache.spark.rdd.RDD[((String, String), Int)] = ShuffledRDD[13] at reduceByKey at <console>:45

scala> val mostCommon = combos.map(x => (x._2, x._1)).sortByKey(false).take(10)
mostCommon: Array[(Int, (String, String))] = Array((67876,(Nike Men's Dri-FIT Victory Golf Polo,Perfect Fitness Perfect Rip
Deck)), (62924,(O'Brien Men's Neoprene Life Vest,Perfect Fitness Perfect Rip Deck)), (54399,(Nike Men's Dri-FIT Victory Golf
 Polo,O'Brien Men's Neoprene Life Vest)), (39656,(Nike Men's Free 5.0+ Running Shoe,Perfect Fitness Perfect Rip Deck)), (393
14,(Perfect Fitness Perfect Rip Deck,Perfect Fitness Perfect Rip Deck)), (35092,(Perfect Fitness Perfect Rip Deck,Under Armo
ur Girls' Toddler Spine Surge Runni)), (33750,(Nike Men's Dri-FIT Victory Golf Polo,Nike Men's Free 5.0+ Running Shoe)), (33
406,(Nike Men's Free 5.0+ Running Shoe,O'Brien Men's Neoprene Life Vest)), (29835,(Nike Men's Dri-FIT Victory Golf Polo,Nike
 Men's Dri-FIT Victory Golf Polo)), (29342,(Nike Men'...
scala> 
```

```
// We print our results, 1 per line, and exit the Spark shell

println(mostCommon.deep.mkString("\n"))

exit
```

EXAMPLE EXECUTION & OUTPUT:



```
scala> // We print our results, 1 per line, and exit the Spark shell

scala>

scala> println(mostCommon.deep.mkString("\n"))
(67876,(Nike Men's Dri-FIT Victory Golf Polo,Perfect Fitness Perfect Rip Deck))
(62924,(O'Brien Men's Neoprene Life Vest,Perfect Fitness Perfect Rip Deck))
(54399,(Nike Men's Dri-FIT Victory Golf Polo,O'Brien Men's Neoprene Life Vest))
(39656,(Nike Men's Free 5.0+ Running Shoe,Perfect Fitness Perfect Rip Deck))
(39314,(Perfect Fitness Perfect Rip Deck,Perfect Fitness Perfect Rip Deck))
(35092,(Perfect Fitness Perfect Rip Deck,Under Armour Girls' Toddler Spine Surge Runni))
(33750,(Nike Men's Dri-FIT Victory Golf Polo,Nike Men's Free 5.0+ Running Shoe))
(33406,(Nike Men's Free 5.0+ Running Shoe,O'Brien Men's Neoprene Life Vest))
(29835,(Nike Men's Dri-FIT Victory Golf Polo,Nike Men's Dri-FIT Victory Golf Polo))
(29342,(Nike Men's Dri-FIT Victory Golf Polo,Under Armour Girls' Toddler Spine Surge Runni))

scala>

scala> exit
warning: there were 1 deprecation warning(s); re-run with -deprecation for details
Jul 19, 2020 1:37:12 PM INFO: parquet.hadoop.ParquetInputFormat: Total input paths to process : 1
Jul 19, 2020 1:37:12 PM INFO: parquet.hadoop.ParquetInputFormat: Total input paths to process : 1
[cloudera@quickstart ~]$
```

To better understand this script, you could read through the comments which aim to explain what each block does and the basic process we're going through.

When we do a 'map', we specify a function that will take each record and output a modified record. This is useful when we only need a couple of fields from each record or when we need the record to use a different field as the key: we simply invoke map with a function that takes in the entire record, and returns a new record with the fields and the key we want.

The 'reduce' operations - like 'join' and 'groupBy' - will organize these records by their keys so we can group similar records together and then process them as a group. For instance, we group every purchased item by which specific order it was in - allowing us to determine all the combinations of products that were part of the same order.

CONCLUSION

If it weren't for Spark, doing cooccurrence analysis like this would be an extremely arduous and time-consuming task. However, using Spark, and a few lines of Scala, you were able to produce a list of the items most frequently purchased together in very little time.

Showing "Data Hub" Value

Unified Storage for All of Your Data

Scenario:

Your Management: can't believe the magic you do with data and is about to promote you and invest in a new team under your lead... when all hell breaks loose. You get an emergency call - as you are now the go-to person - and your manager is screaming about the loss of sales over the last three days...

You: from slightly excited to under the gun in seconds...well, lucky for you there might be a quick way to find out what is happening...

Tutorial Exercise 4

Explore Log Events Interactively

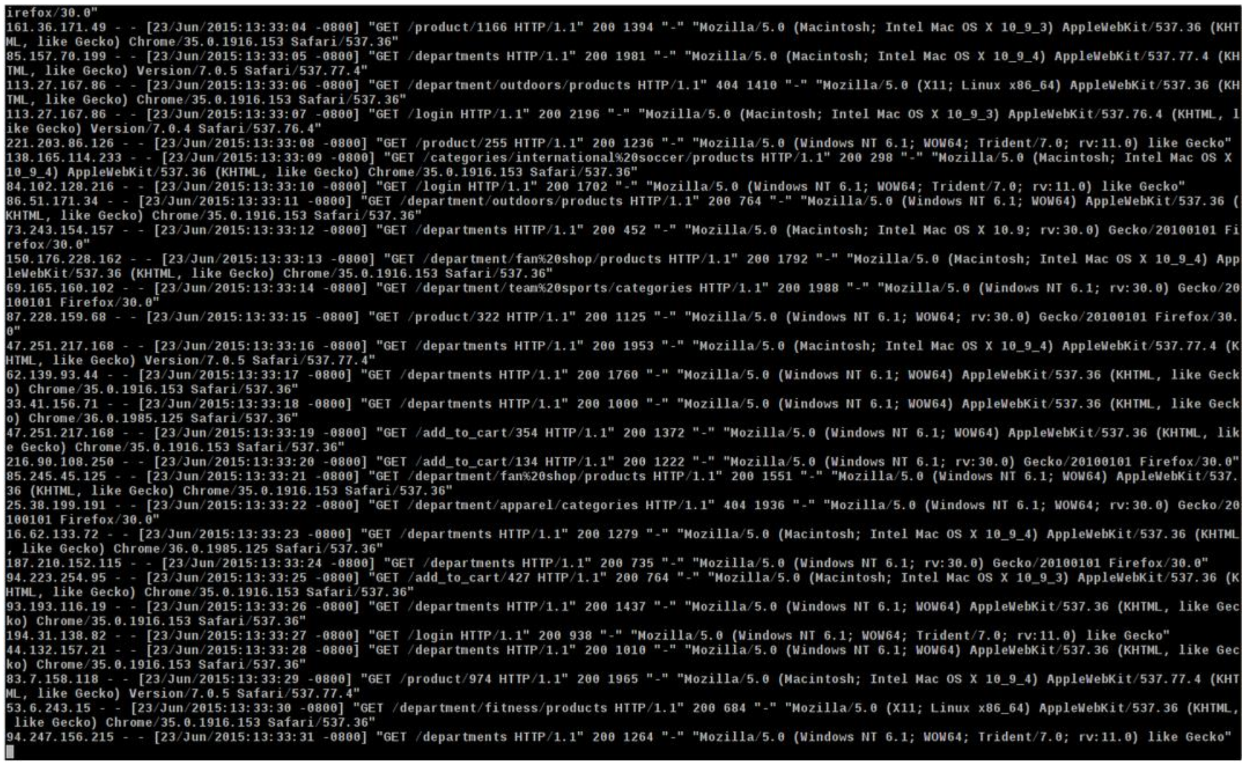
What you can do to enable guided drill down and exploration of data is to make it searchable. By indexing your data using any of the indexing options provided by Cloudera Search, your data can be searchable to a variety of audiences. You can choose to batch index data using the MapReduce Indexing tool, or as in our example below, extend the Apache Flume configuration that is already ingesting the web log data to also post events to Apache Solr for indexing in real-time.

Good to Know

Flume is a scalable, real-time ingest framework that allows you to route, filter, aggregate, and perform "mini-operations" on data on its way into a scalable processing platform like CDH. However, you do want to minimize the logic done on its way into the cluster. This will assure ready availability for other workloads and prevent ingest bottlenecks. It still allows you to utilize the huge scalability of the CDH cluster for more heavy-duty processing. If you need to do some heavy-duty aggregations or multi-step ETL of incoming data, you should use Spark - an in-memory processing framework that scales with the rest of the processing framework and has advanced analytic capabilities built in.

Note also that in real production systems it might be a better option to pipe any log events through syslog. This provides a more robust production deployment, as it does not depend on file appends.

The web log data in a standard web server log may look something like this:



```
irefox/30.0"
161.36.171.49 - - [23/Jun/2015:13:33:04 -0800] "GET /product/1166 HTTP/1.1" 200 1394 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_3) AppleWebKit/537.36 (KHT
ML, like Gecko) Chrome/35.0.1916.153 Safari/537.36"
85.157.70.199 - - [23/Jun/2015:13:33:05 -0800] "GET /departments HTTP/1.1" 200 1981 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4) AppleWebKit/537.77.4 (KH
TML, like Gecko) Version/7.0.5 Safari/537.77.4"
113.27.167.86 - - [23/Jun/2015:13:33:06 -0800] "GET /department/outdoors/products HTTP/1.1" 404 1410 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KH
TML, like Gecko) Chrome/35.0.1916.153 Safari/537.36"
113.27.167.86 - - [23/Jun/2015:13:33:07 -0800] "GET /login HTTP/1.1" 200 2196 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_3) AppleWebKit/537.76.4 (KHTML, l
ike Gecko) Version/7.0.4 Safari/537.76.4"
221.203.86.126 - - [23/Jun/2015:13:33:08 -0800] "GET /product/255 HTTP/1.1" 200 1236 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko"
138.165.114.233 - - [23/Jun/2015:13:33:09 -0800] "GET /categories/international%20soccer/products HTTP/1.1" 200 298 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X
10_9_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/35.0.1916.153 Safari/537.36"
84.102.128.216 - - [23/Jun/2015:13:33:10 -0800] "GET /login HTTP/1.1" 200 1702 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko"
86.51.171.34 - - [23/Jun/2015:13:33:11 -0800] "GET /department/outdoors/products HTTP/1.1" 200 764 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (
KHTML, like Gecko) Chrome/35.0.1916.153 Safari/537.36"
73.243.154.157 - - [23/Jun/2015:13:33:12 -0800] "GET /departments HTTP/1.1" 200 452 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:30.0) Gecko/20100101 Fi
refox/30.0"
150.176.228.162 - - [23/Jun/2015:13:33:13 -0800] "GET /department/fan%20shop/products HTTP/1.1" 200 1792 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4) App
leWebKit/537.36 (KHTML, like Gecko) Chrome/35.0.1916.153 Safari/537.36"
69.165.160.102 - - [23/Jun/2015:13:33:14 -0800] "GET /department/team%20sports/categories HTTP/1.1" 200 1988 "-" "Mozilla/5.0 (Windows NT 6.1; rv:30.0) Gecko/20
100101 Firefox/30.0"
87.228.159.68 - - [23/Jun/2015:13:33:15 -0800] "GET /product/322 HTTP/1.1" 200 1125 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:30.0) Gecko/20100101 Firefox/30.
0"
47.251.217.168 - - [23/Jun/2015:13:33:16 -0800] "GET /departments HTTP/1.1" 200 1953 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4) AppleWebKit/537.77.4 (K
HTML, like Gecko) Version/7.0.5 Safari/537.77.4"
62.139.93.44 - - [23/Jun/2015:13:33:17 -0800] "GET /departments HTTP/1.1" 200 1760 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Geck
o) Chrome/35.0.1916.153 Safari/537.36"
33.41.156.71 - - [23/Jun/2015:13:33:18 -0800] "GET /departments HTTP/1.1" 200 1000 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Geck
o) Chrome/36.0.1985.125 Safari/537.36"
47.251.217.168 - - [23/Jun/2015:13:33:19 -0800] "GET /add_to_cart/354 HTTP/1.1" 200 1372 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, lik
e Gecko) Chrome/35.0.1916.153 Safari/537.36"
216.90.108.250 - - [23/Jun/2015:13:33:20 -0800] "GET /add_to_cart/134 HTTP/1.1" 200 1222 "-" "Mozilla/5.0 (Windows NT 6.1; rv:30.0) Gecko/20100101 Firefox/30.0"
85.245.45.125 - - [23/Jun/2015:13:33:21 -0800] "GET /department/fan%20shop/products HTTP/1.1" 200 1551 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.
36 (KHTML, like Gecko) Chrome/35.0.1916.153 Safari/537.36"
25.38.199.191 - - [23/Jun/2015:13:33:22 -0800] "GET /department/apparel/categories HTTP/1.1" 404 1936 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:30.0) Gecko/20
100101 Firefox/30.0"
16.62.133.72 - - [23/Jun/2015:13:33:23 -0800] "GET /departments HTTP/1.1" 200 1279 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4) AppleWebKit/537.36 (KHTML
, like Gecko) Chrome/36.0.1985.125 Safari/537.36"
187.210.152.115 - - [23/Jun/2015:13:33:24 -0800] "GET /departments HTTP/1.1" 200 735 "-" "Mozilla/5.0 (Windows NT 6.1; rv:30.0) Gecko/20100101 Firefox/30.0"
94.223.254.95 - - [23/Jun/2015:13:33:25 -0800] "GET /add_to_cart/427 HTTP/1.1" 200 764 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_3) AppleWebKit/537.36 (K
HTML, like Gecko) Chrome/35.0.1916.153 Safari/537.36"
93.193.116.19 - - [23/Jun/2015:13:33:26 -0800] "GET /departments HTTP/1.1" 200 1437 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gec
ko) Chrome/35.0.1916.153 Safari/537.36"
194.31.138.82 - - [23/Jun/2015:13:33:27 -0800] "GET /login HTTP/1.1" 200 938 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko"
44.132.157.21 - - [23/Jun/2015:13:33:28 -0800] "GET /departments HTTP/1.1" 200 1010 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gec
ko) Chrome/35.0.1916.153 Safari/537.36"
83.7.158.118 - - [23/Jun/2015:13:33:29 -0800] "GET /product/974 HTTP/1.1" 200 1965 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4) AppleWebKit/537.77.4 (KHT
ML, like Gecko) Version/7.0.5 Safari/537.77.4"
53.6.243.15 - - [23/Jun/2015:13:33:30 -0800] "GET /department/fitness/products HTTP/1.1" 200 684 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML,
 like Gecko) Chrome/35.0.1916.153 Safari/537.36"
94.247.156.215 - - [23/Jun/2015:13:33:31 -0800] "GET /departments HTTP/1.1" 200 1264 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko"
```

Solr organizes data similarly to the way a SQL database does. Each record is called a 'document' and consists of fields defined by the schema: just like a row in a database table. Instead of a table, Solr calls it a 'collection' of documents. The difference is that data in Solr tends to be more loosely structured. Fields may be optional, and instead of always matching exact values, you can also enter text queries that partially match a field, just like you're searching for web pages. You'll also see Hue refer to 'shards' - and that's just the way Solr breaks collections up to spread them around the cluster so you can search all your data in parallel.

Here is how you can start real-time-indexing via Cloudera Search and Flume over the sample web server log data and use the Search UI in Hue to explore it:

Create your search index

Ordinarily when you are deploying a new search schema, there are four steps:

1. Creating an empty configuration

   For the sake of this tutorial, you won't need to actually execute steps 1 or 2, as we have included the configuration and the schema file in your cluster already. They can be reviewed by exploring **`/opt/examples/flume/solr_configs`**.

   If you were doing this on your own, you would generate the configs by executing the following command:

   ```
   [cloudera@quickstart ~]$ solrctl --zk quickstart:2181/solr instancedir --generate solr_configs
   ```

   `You don't need to do this for this tutorial. We have already generated the configuration for you. This instruction is here in case you want to create your own index.`

2. Edit your schema

   You can view the modified sample schema at the following location in the Cloudera Quickstart virtual environment:

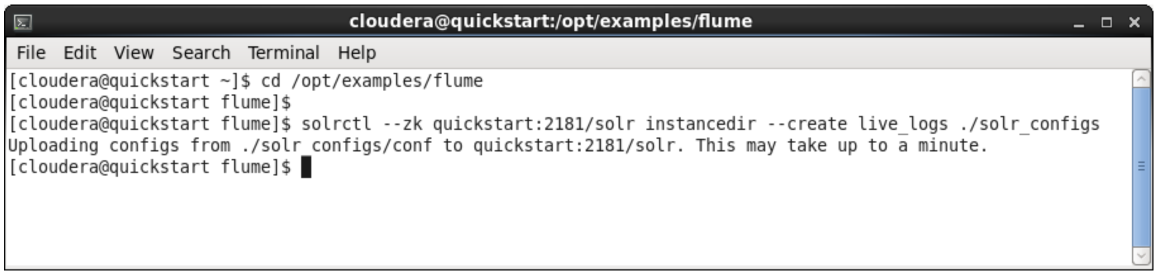
   `/var/lib/cloudera-quickstart/tutorial/media/schema.xml.txt`

   The most common area that you would be interested in is the <fields></fields> section. From this area you can define the fields that are present and searchable in your index.

3. Uploading your configuration

   Execute the following statements at the Linux Terminal prompt:

```
cd /opt/examples/flume

solrctl --zk quickstart:2181/solr instancedir --create
live_logs ./solr_configs
```

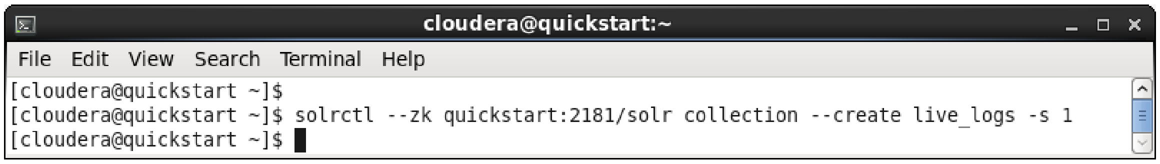


```
cloudera@quickstart:/opt/examples/flume
[cloudera@quickstart ~]$ cd /opt/examples/flume
[cloudera@quickstart flume]$
[cloudera@quickstart flume]$ solrctl --zk quickstart:2181/solr instancedir --create live_logs ./solr_configs
Uploading configs from ./solr_configs/conf to quickstart:2181/solr. This may take up to a minute.
[cloudera@quickstart flume]$
```

4. Creating your collection

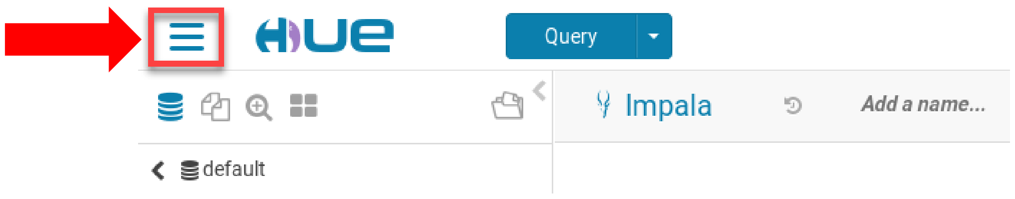
   Execute the following statement at the Linux Terminal prompt:

```
solrctl --zk quickstart:2181/solr collection --create
live_logs -s 1
```

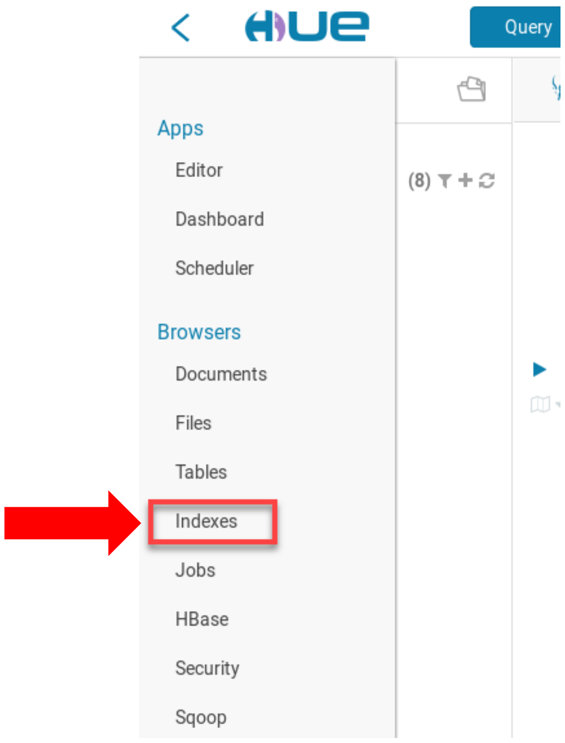


```
cloudera@quickstart:~
[cloudera@quickstart ~]$
[cloudera@quickstart ~]$ solrctl --zk quickstart:2181/solr collection --create live_logs -s 1
[cloudera@quickstart ~]$
```

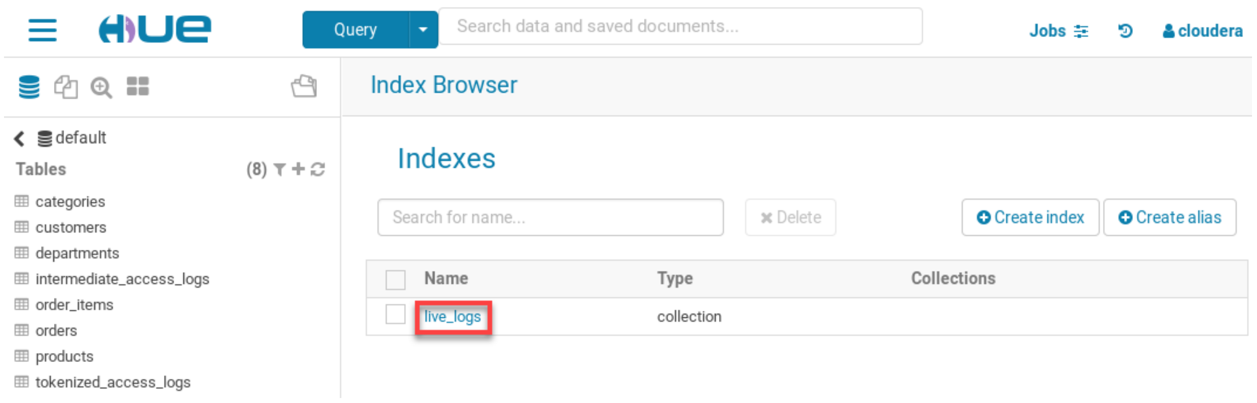
You can verify that you successfully created your collection in Solr by going to Hue and clicking on the menu icon at the upper left portion of the Hue window.



Select the INDEXES item from the Hue menu.

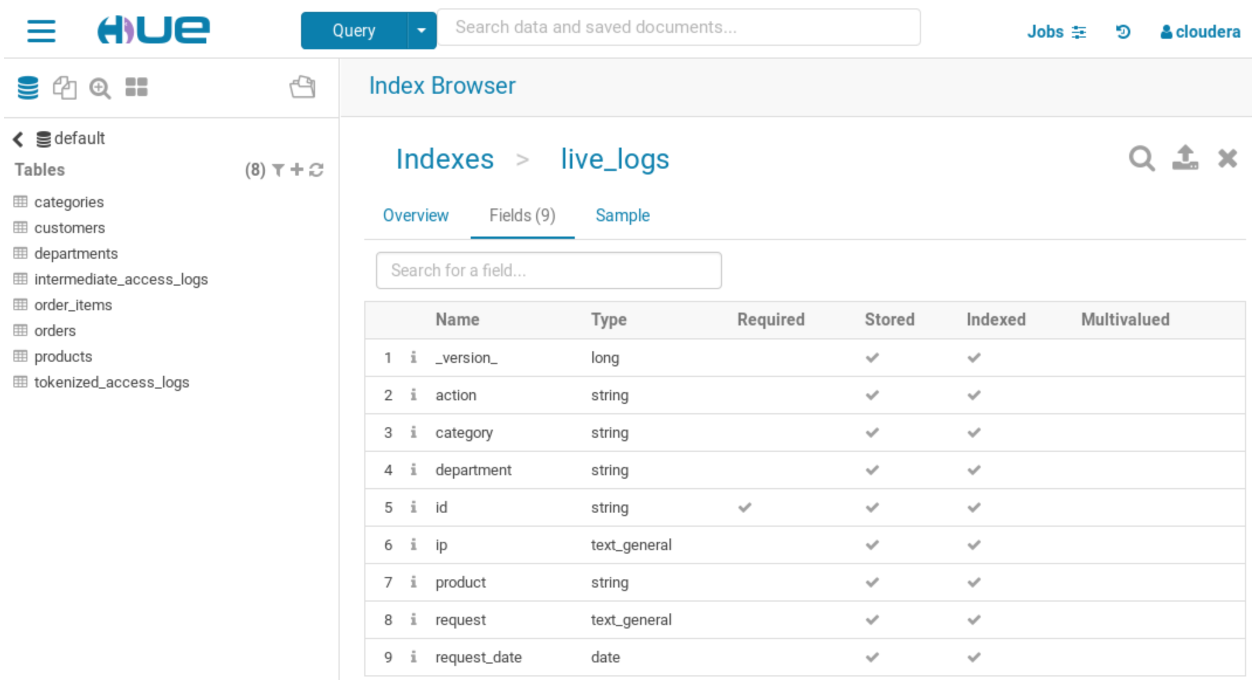


Now you can see the collection that we just created, **live_logs**, click on it.



Name	Type	Collections
live_logs	collection	

You are now viewing the fields that we defined in our schema.xml file



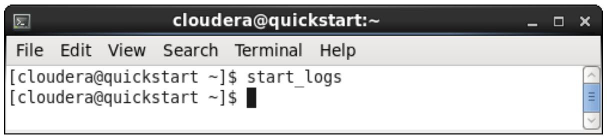
		Name	Type	Required	Stored	Indexed	Multivalued
1	i	_version_	long		✓	✓	
2	i	action	string		✓	✓	
3	i	category	string		✓	✓	
4	i	department	string		✓	✓	
5	i	id	string	✓	✓	✓	
6	i	ip	text_general		✓	✓	
7	i	product	string		✓	✓	
8	i	request	text_general		✓	✓	
9	i	request_date	date		✓	✓	

Now that you have verified that your search collection/index was created successfully, we can start putting data into it using Flume and Morphlines. Flume is a tool for ingesting streams of data into your cluster from sources such as log files, network streams, and more. Morphlines is a Java library for doing ETL on-the-fly, and it's an excellent companion to Flume. It allows you to define a chain of tasks like reading records, parsing and formatting individual fields, and deciding where to send them, etc. We've defined a morphline that reads records from Flume, breaks them into the fields we want to search on, and loads them into Solr. This example Morphline is defined at **`/opt/examples/flume/conf/morphline.conf`**, and we're going to use it to index our records in real-time as they're created and ingested by Flume.

Starting the Log Generator

Your Cloudera Live cluster has a log generator for use with sample data. Start the log generator by running the following command at the Linux Terminal prompt:

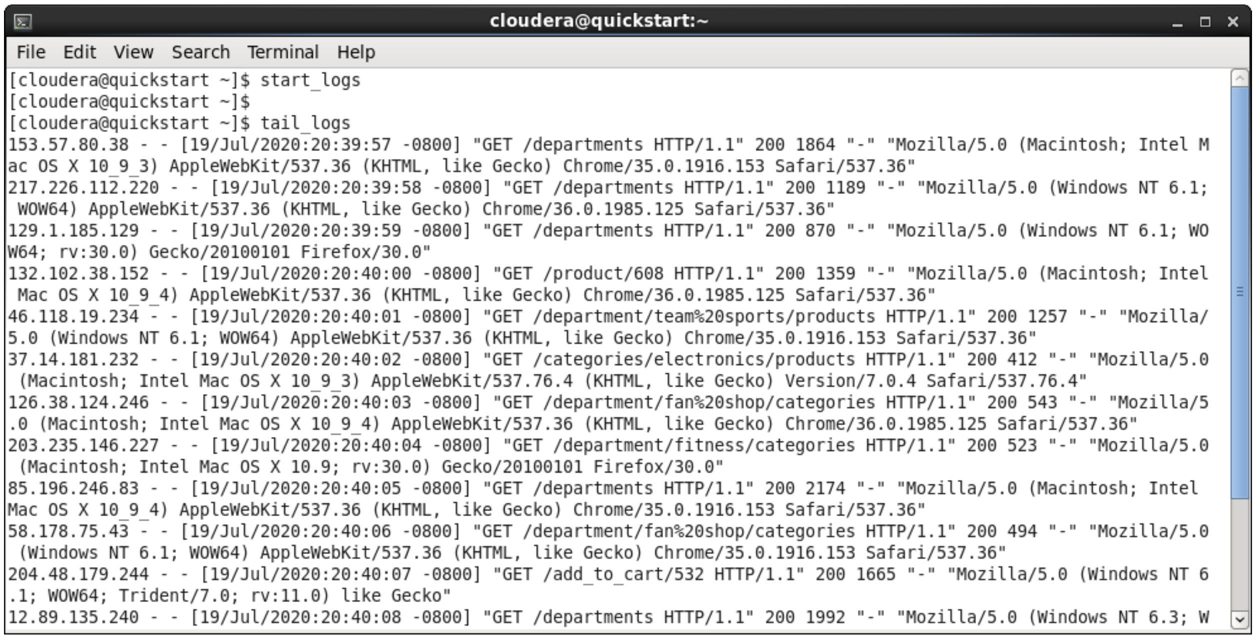
```
start_logs
```



```
[cloudera@quickstart ~]$ start_logs
[cloudera@quickstart ~]$ 
```

You can verify that the log generator has started by running the following command at the Linux Terminal prompt:

```
tail_logs
```



cloudera@quickstart:~

File Edit View Search Terminal Help

```
[cloudera@quickstart ~]$ start_logs
[cloudera@quickstart ~]$
[cloudera@quickstart ~]$ tail_logs
153.57.80.38 - - [19/Jul/2020:20:39:57 -0800] "GET /departments HTTP/1.1" 200 1864 "-" "Mozilla/5.0 (Macintosh; Intel M
ac OS X 10_9_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/35.0.1916.153 Safari/537.36"
217.226.112.220 - - [19/Jul/2020:20:39:58 -0800] "GET /departments HTTP/1.1" 200 1189 "-" "Mozilla/5.0 (Windows NT 6.1;
 WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/36.0.1985.125 Safari/537.36"
129.1.185.129 - - [19/Jul/2020:20:39:59 -0800] "GET /departments HTTP/1.1" 200 870 "-" "Mozilla/5.0 (Windows NT 6.1; WO
W64; rv:30.0) Gecko/20100101 Firefox/30.0"
132.102.38.152 - - [19/Jul/2020:20:40:00 -0800] "GET /product/608 HTTP/1.1" 200 1359 "-" "Mozilla/5.0 (Macintosh; Intel
 Mac OS X 10_9_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/36.0.1985.125 Safari/537.36"
46.118.19.234 - - [19/Jul/2020:20:40:01 -0800] "GET /department/team%20sports/products HTTP/1.1" 200 1257 "-" "Mozilla/
5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/35.0.1916.153 Safari/537.36"
37.14.181.232 - - [19/Jul/2020:20:40:02 -0800] "GET /categories/electronics/products HTTP/1.1" 200 412 "-" "Mozilla/5.0
 (Macintosh; Intel Mac OS X 10_9_3) AppleWebKit/537.76.4 (KHTML, like Gecko) Version/7.0.4 Safari/537.76.4"
126.38.124.246 - - [19/Jul/2020:20:40:03 -0800] "GET /department/fan%20shop/categories HTTP/1.1" 200 543 "-" "Mozilla/5
.0 (Macintosh; Intel Mac OS X 10_9_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/36.0.1985.125 Safari/537.36"
203.235.146.227 - - [19/Jul/2020:20:40:04 -0800] "GET /department/fitness/categories HTTP/1.1" 200 523 "-" "Mozilla/5.0
 (Macintosh; Intel Mac OS X 10.9; rv:30.0) Gecko/20100101 Firefox/30.0"
85.196.246.83 - - [19/Jul/2020:20:40:05 -0800] "GET /departments HTTP/1.1" 200 2174 "-" "Mozilla/5.0 (Macintosh; Intel
Mac OS X 10_9_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/35.0.1916.153 Safari/537.36"
58.178.75.43 - - [19/Jul/2020:20:40:06 -0800] "GET /department/fan%20shop/categories HTTP/1.1" 200 494 "-" "Mozilla/5.0
 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/35.0.1916.153 Safari/537.36"
204.48.179.244 - - [19/Jul/2020:20:40:07 -0800] "GET /add_to_cart/532 HTTP/1.1" 200 1665 "-" "Mozilla/5.0 (Windows NT 6
.1; WOW64; Trident/7.0; rv:11.0) like Gecko"
12.89.135.240 - - [19/Jul/2020:20:40:08 -0800] "GET /departments HTTP/1.1" 200 1992 "-" "Mozilla/5.0 (Windows NT 6.3; W
```

When you're done watching the logs, you can hit <Ctrl + C> to return to your terminal.

Flume and the morphline

Now that we have an empty Solr index, and live log events coming in to our fake access.log, we can use Flume and morphlines to load the index with the real-time log data.

The key player in this tutorial is Flume. Flume is a system for collecting, aggregating, and moving large amounts of log data from many different sources to a centralized data source.

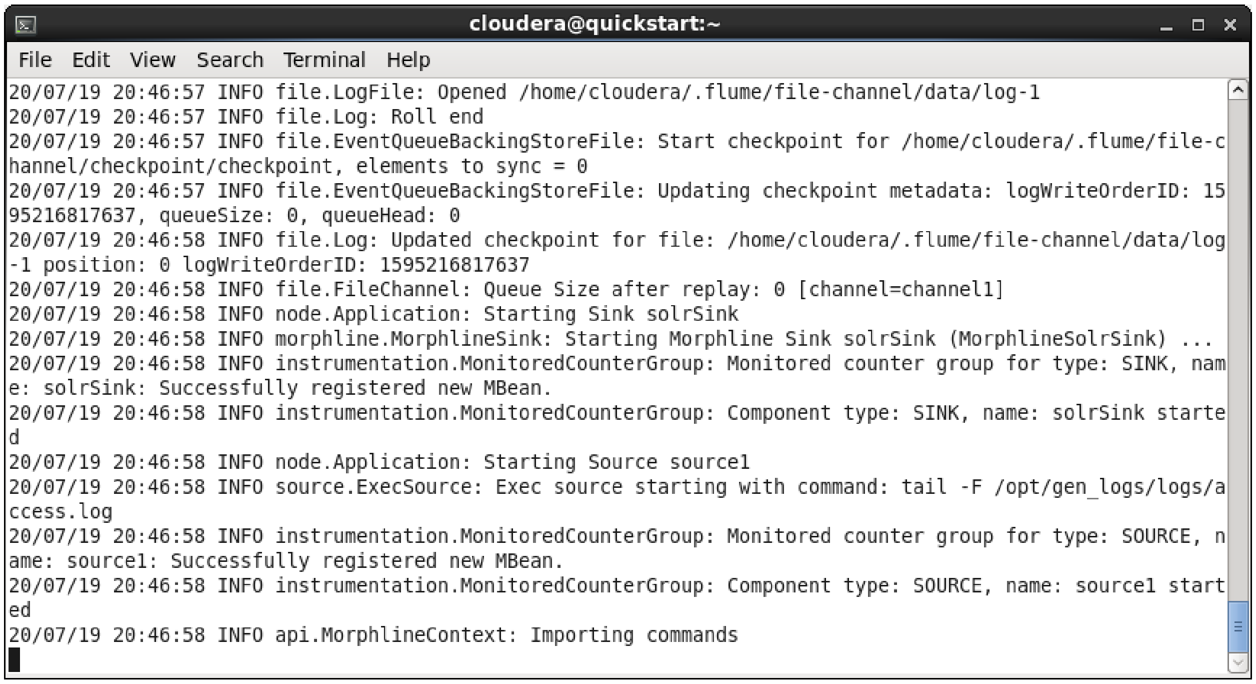
With a few simple configuration files, we can use Flume and a morphline (a simple way to accomplish on-the-fly ETL,) to load our data into our Solr index.

You can use Flume to load many other types of data stores; Solr is just the example we are using for this tutorial.

Start the Flume agent by executing the following command at the Linux Terminal prompt:

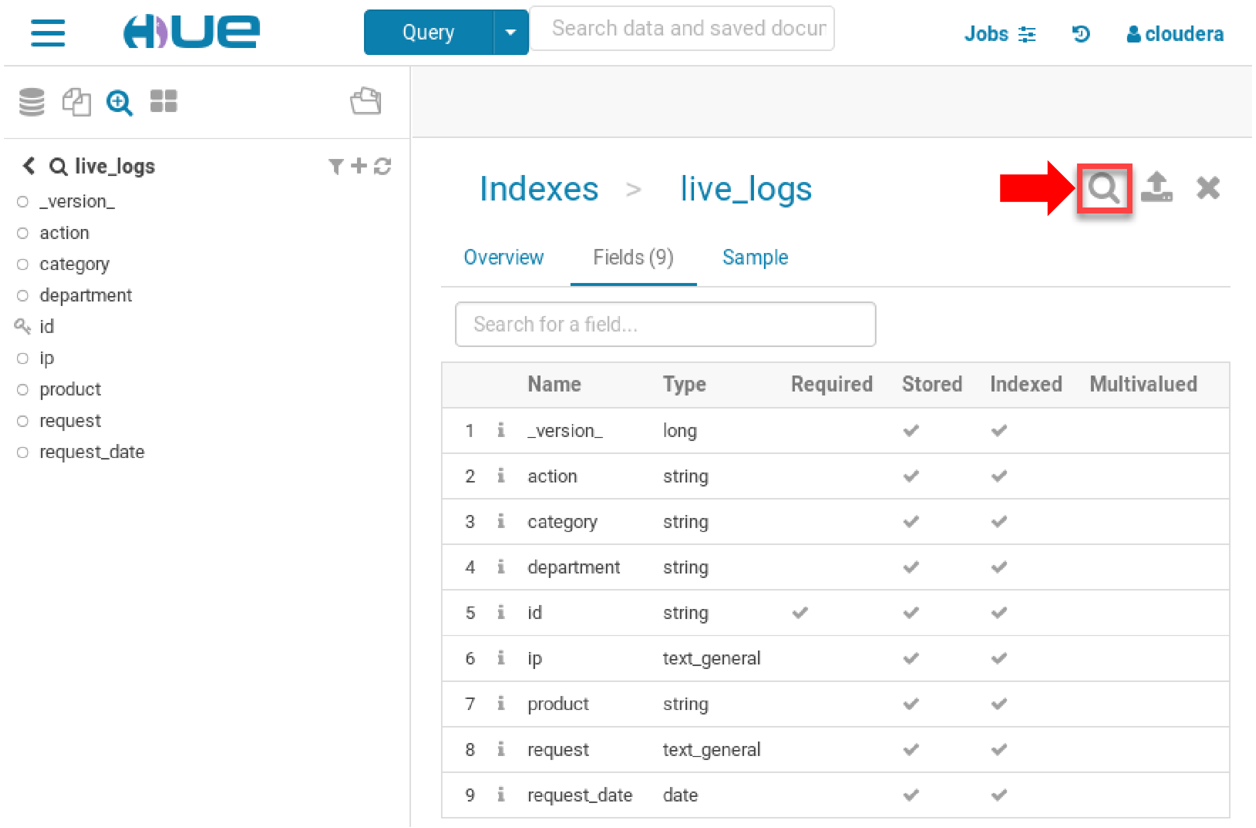
```
flume-ng agent \
    --conf /opt/examples/flume/conf \
    --conf-file /opt/examples/flume/conf/flume.conf \
    --name agent1 \
    -Dflume.root.logger=DEBUG,INFO,console
```

This will start running the Flume agent in the foreground. Once it has started, and is processing records, you should see something like:



```
20/07/19 20:46:57 INFO file.LogFile: Opened /home/cloudera/.flume/file-channel/data/log-1
20/07/19 20:46:57 INFO file.Log: Roll end
20/07/19 20:46:57 INFO file.EventQueueBackingStoreFile: Start checkpoint for /home/cloudera/.flume/file-c
hannel/checkpoint/checkpoint, elements to sync = 0
20/07/19 20:46:57 INFO file.EventQueueBackingStoreFile: Updating checkpoint metadata: logWriteOrderID: 15
95216817637, queueSize: 0, queueHead: 0
20/07/19 20:46:58 INFO file.Log: Updated checkpoint for file: /home/cloudera/.flume/file-channel/data/log
-1 position: 0 logWriteOrderID: 1595216817637
20/07/19 20:46:58 INFO file.FileChannel: Queue Size after replay: 0 [channel=channel1]
20/07/19 20:46:58 INFO node.Application: Starting Sink solrSink
20/07/19 20:46:58 INFO morphline.MorphlineSink: Starting Morphline Sink solrSink (MorphlineSolrSink) ...
20/07/19 20:46:58 INFO instrumentation.MonitoredCounterGroup: Monitored counter group for type: SINK, nam
e: solrSink: Successfully registered new MBean.
20/07/19 20:46:58 INFO instrumentation.MonitoredCounterGroup: Component type: SINK, name: solrSink starte
d
20/07/19 20:46:58 INFO node.Application: Starting Source source1
20/07/19 20:46:58 INFO source.ExecSource: Exec source starting with command: tail -F /opt/gen_logs/logs/a
ccess.log
20/07/19 20:46:58 INFO instrumentation.MonitoredCounterGroup: Monitored counter group for type: SOURCE, n
ame: source1: Successfully registered new MBean.
20/07/19 20:46:58 INFO instrumentation.MonitoredCounterGroup: Component type: SOURCE, name: source1 start
ed
20/07/19 20:46:58 INFO api.MorphlineContext: Importing commands
```

Now you can go back to the Hue user interface and click search icon (🔍) from the right window panel.

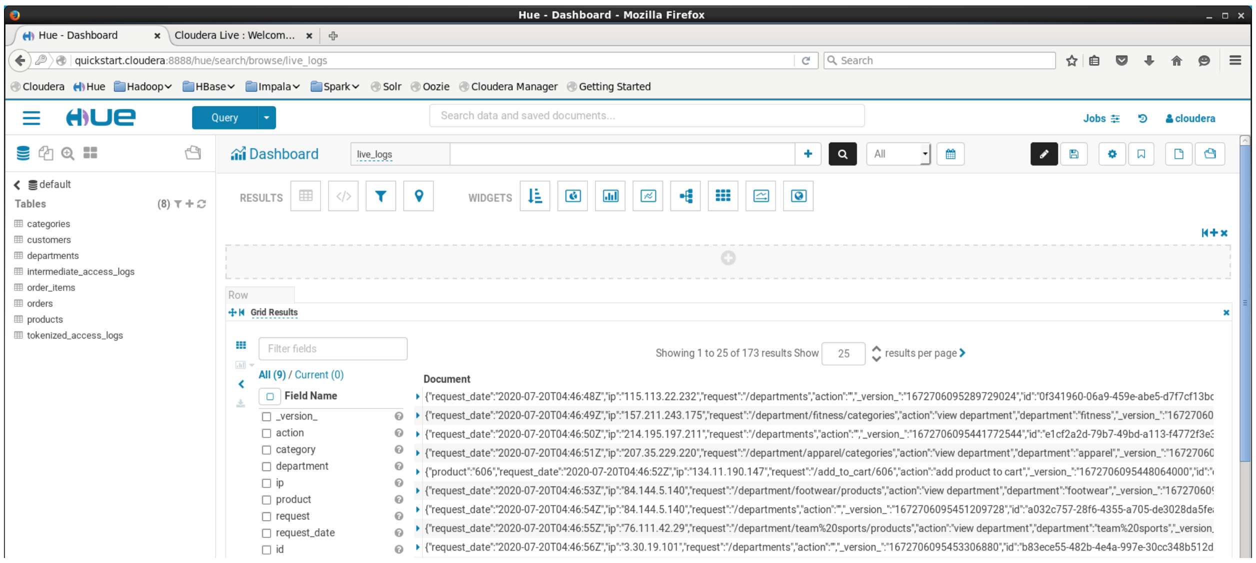


Indexes > live_logs

Overview | Fields (9) | Sample

		Name	Type	Required	Stored	Indexed	Multivalued
1	i	_version_	long		✓	✓	
2	i	action	string		✓	✓	
3	i	category	string		✓	✓	
4	i	department	string		✓	✓	
5	i	id	string	✓	✓	✓	
6	i	ip	text_general		✓	✓	
7	i	product	string		✓	✓	
8	i	request	text_general		✓	✓	
9	i	request_date	date		✓	✓	

You will be able to search, drill down into, and browse the events that have been indexed.



Showing 1 to 25 of 173 results Show 25 results per page

```
Document
{"request_date":"2020-07-20T04:46:48Z","ip":"115.113.22.232","request":"/departments","action":"","_version_":"1672706095289729024","id":"0f341960-06a9-459e-abe5-d7f7cf13bc
{"request_date":"2020-07-20T04:46:49Z","ip":"157.211.243.175","request":"/department/fitness/categories","action":"view department","department":"fitness","_version_":"16727060
{"request_date":"2020-07-20T04:46:50Z","ip":"214.195.197.211","request":"/departments","action":"","_version_":"1672706095441772544","id":"e1cf2a2d-79b7-49bd-a113-f4772f3e3
{"request_date":"2020-07-20T04:46:51Z","ip":"207.35.229.220","request":"/department/apparel/categories","action":"view department","department":"apparel","_version_":"16727060
{"product":"606","request_date":"2020-07-20T04:46:52Z","ip":"134.11.190.147","request":"/add_to_cart/606","action":"add product to cart","_version_":"1672706095448064000","id":"
{"request_date":"2020-07-20T04:46:53Z","ip":"84.144.5.140","request":"/department/footwear/products","action":"view department","department":"footwear","_version_":"16727060
{"request_date":"2020-07-20T04:46:54Z","ip":"84.144.5.140","request":"/departments","action":"","_version_":"1672706095451209728","id":"a032c757-28f6-4355-a705-de3028da5fe
{"request_date":"2020-07-20T04:46:55Z","ip":"76.111.42.29","request":"/department/team%20sports/products","action":"view department","department":"team%20sports","_version_
{"request_date":"2020-07-20T04:46:56Z","ip":"3.30.19.101","request":"/departments","action":"","_version_":"1672706095453306880","id":"b83ece55-482b-4e4a-997e-30cc348b512d
```

For our story’s sake, we pretend that you started indexing data the same time as you started ingesting it (via Flume) to the platform, so that when your manager escalated the issue, you could immediately drill down into data from the last three days and explore what happened. For example, perhaps you noted a lot of distributed denial-of-service (DDoS) events and could take the right measures to preempt the attack. Problem solved! Management is fantastically happy with your recent contributions, which of course leads to a great bonus or something similar.

CONCLUSION

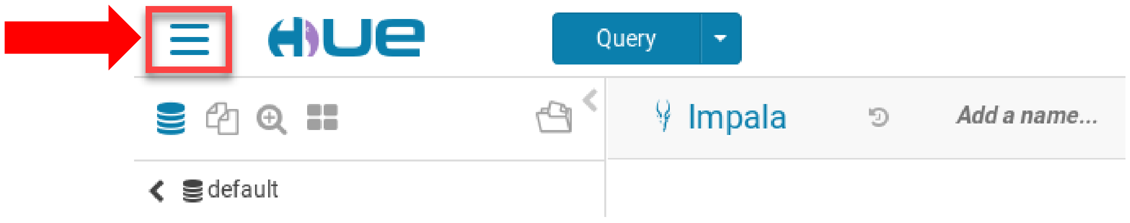
Now you have learned how to use Cloudera Search to allow exploration of data in real time, using Flume and Solr and Morphlines. Further, you now understand how you can serve multiple use cases over the same data - as well as from previous steps: serve multiple data sets to provide bigger insights. The flexibility and multi-workload capability of a Hadoop-based Enterprise Data Hub are some of the core elements that have made Hadoop valuable to organizations worldwide.

Tutorial Exercise 5

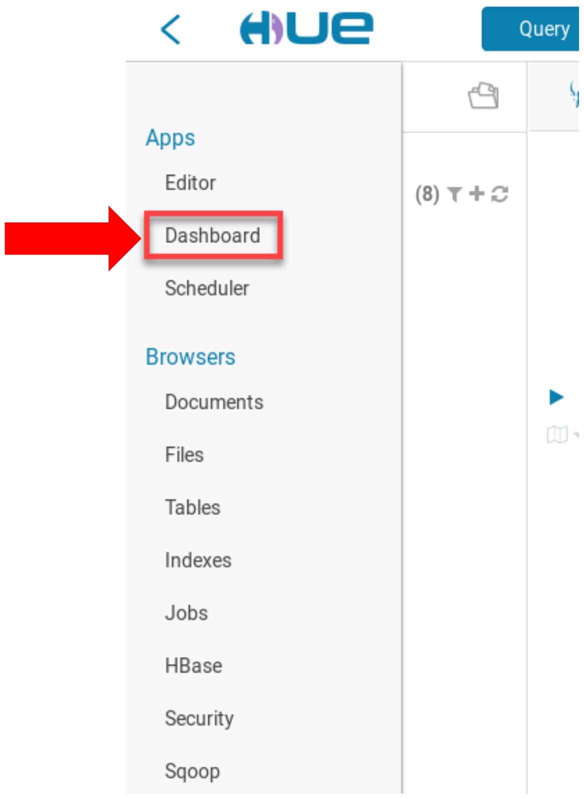
Building a Dashboard

In this exercise you will establish a dashboard using Hue. Please note that some of the actions performed in the exercise may take some time due to system performance limitations.

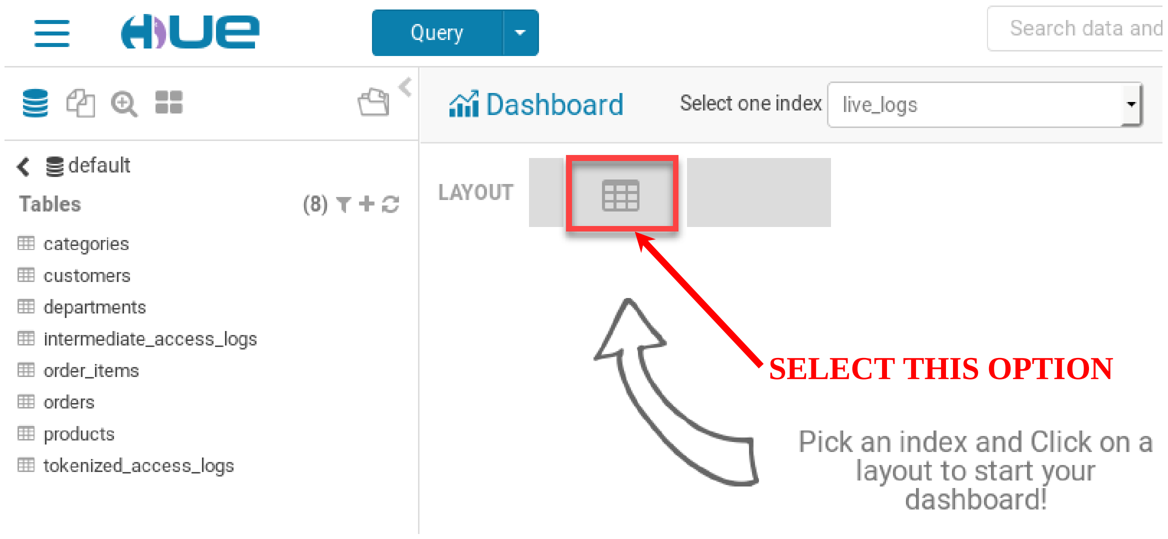
Click on the menu icon at the upper left potion of the Hue window.



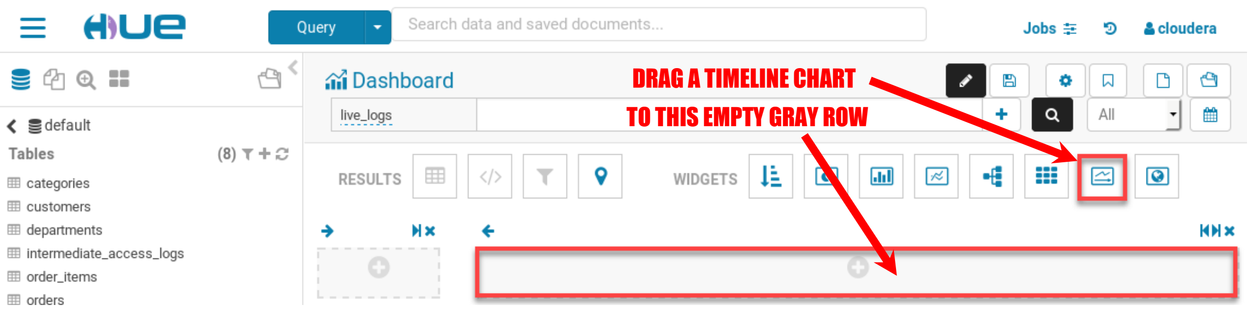
Select the DASHBOARD item from the Hue menu.



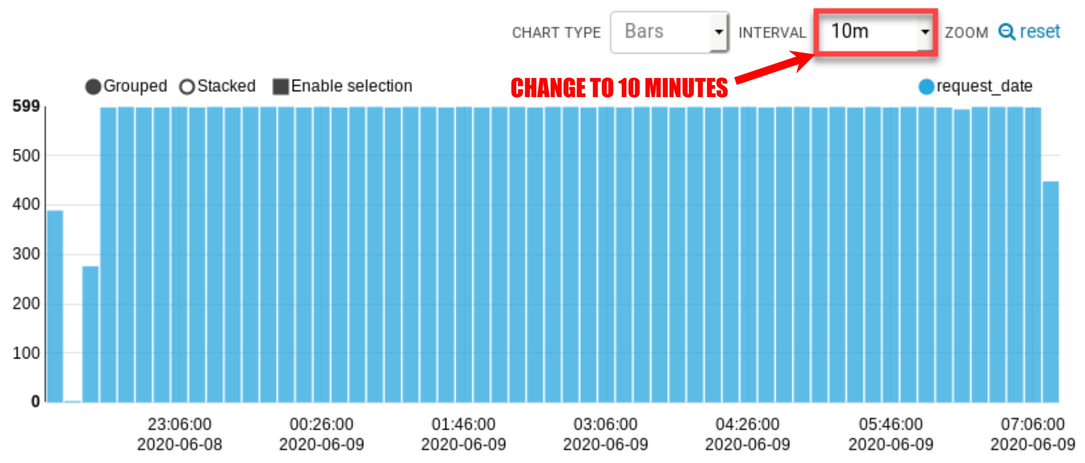
Click on the middle option in the list of Layout options.



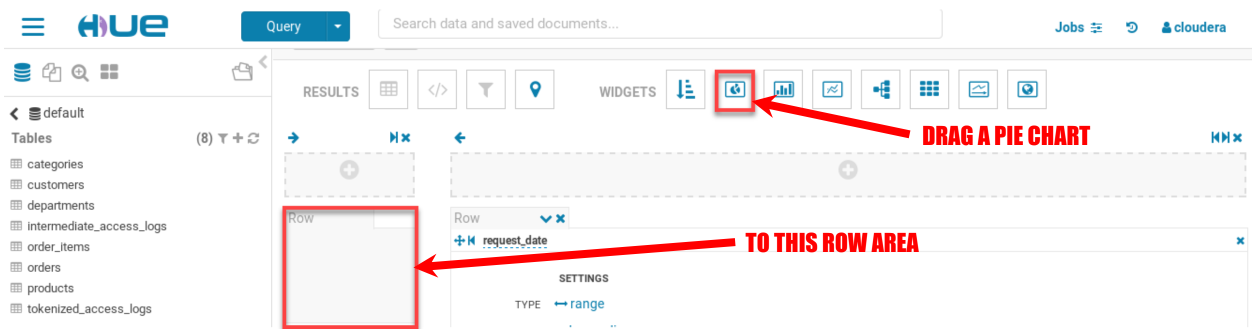
This will take you into the edit-mode where you choose different widgets and layouts that you would like to see. You can choose a number of options and configuration here, but for now, just drag a timeline chart (📈) into the top gray row.



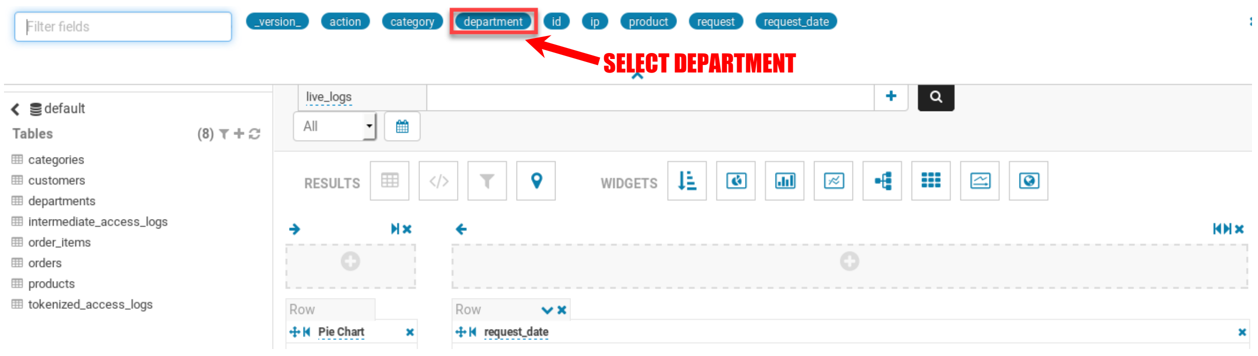
The system will automatically detect that the **request_date** attribute serves as the timeline parameter in the chart. Change the time interval in the chart to 10 minutes. Please note your system performance may be very slow when you attempt to do this due to processing being performed by the Cloudera Quickstart system to generate the chart. Also, the chart that is generated will likely look different from the illustration below.



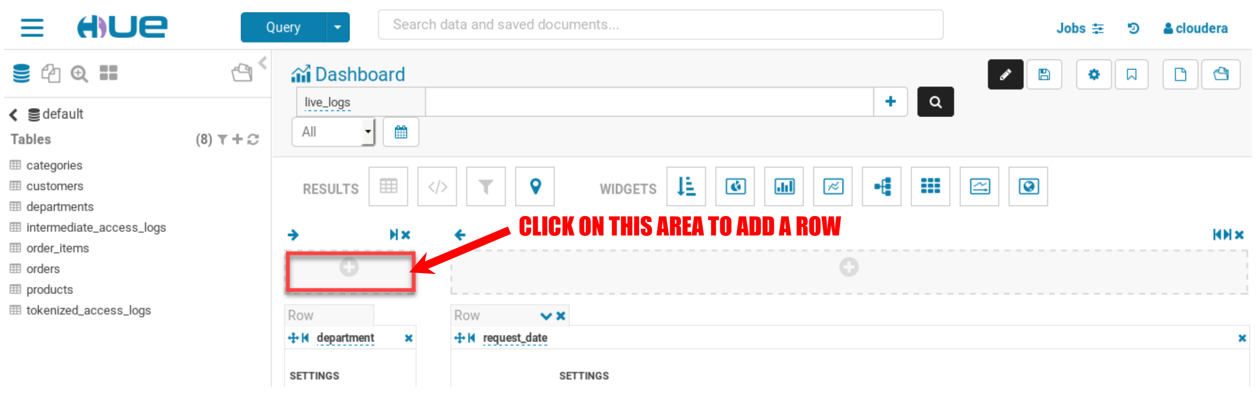
Now we are going to add a pie chart to the dashboard. Drag the pie chart icon (◙) over to the row area at the left side of the dashboard panel.



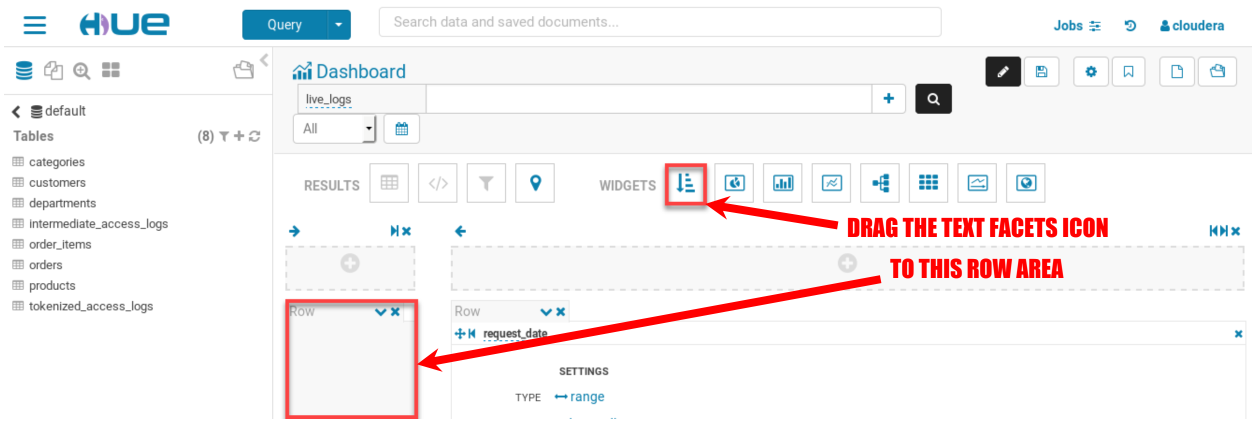
You'll then need to select the pie chart factor. Select DEPARTMENT from the list of parameter options located at the top of the dashboard panel.



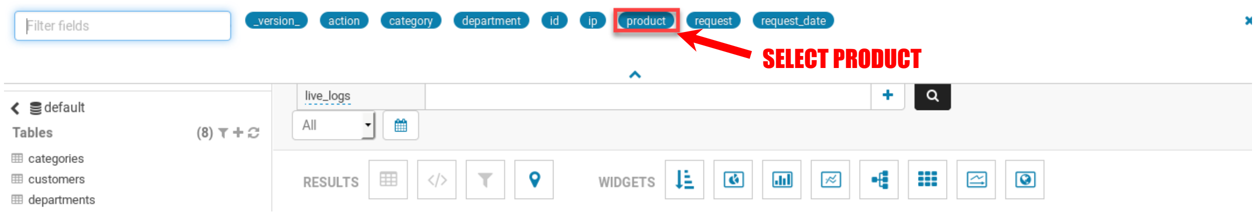
We will now add a text facet to the dashboard. First, we need to create a new row in the left dashboard panel area. Click on the plus item (⊕) located in the upper left portion of the dashboard panel.



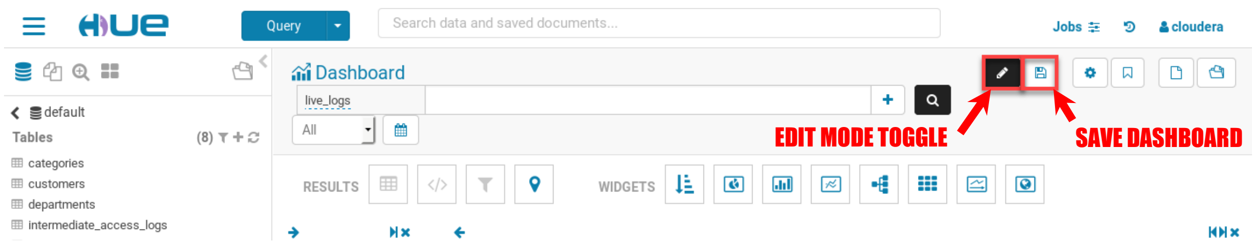
The new row has now been established. Now drag the text facets icon (⇂☰) over to the new row area at the left side of the dashboard panel.



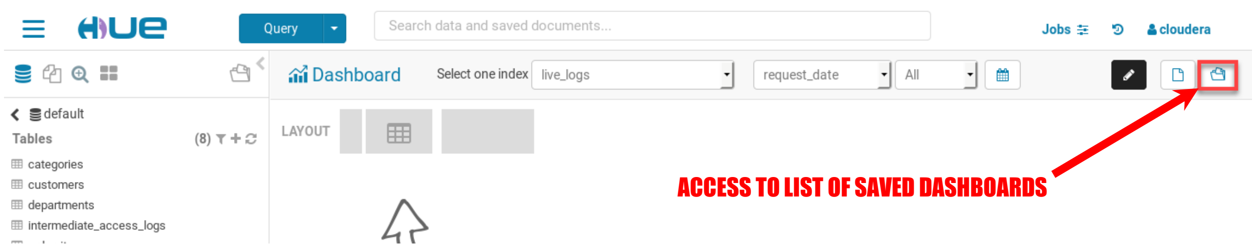
You'll then need to select the text facets factor. Select PRODUCT from the list of parameter options located at the top of the dashboard panel.



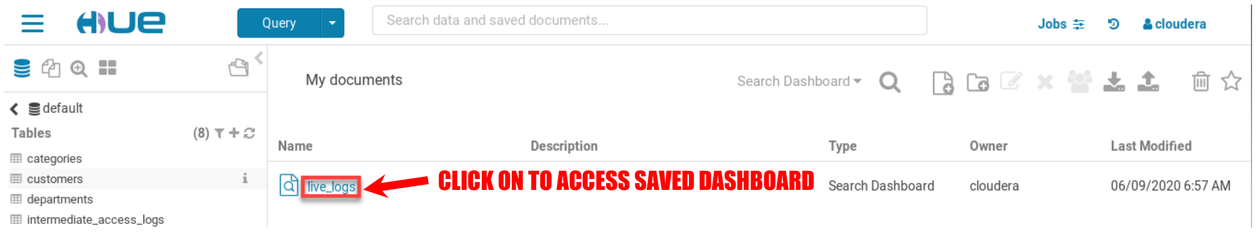
You've completed the creation of the dashboard. Click on the pencil icon (✎) to exit out of editor mode. Then click on the disk icon (💾) to save the dashboard.



To return to the saved dashboard at a future date, enter into the dashboards area as previously specified and click on the dashboards icon ( ) located in the upper right portion of the panel.

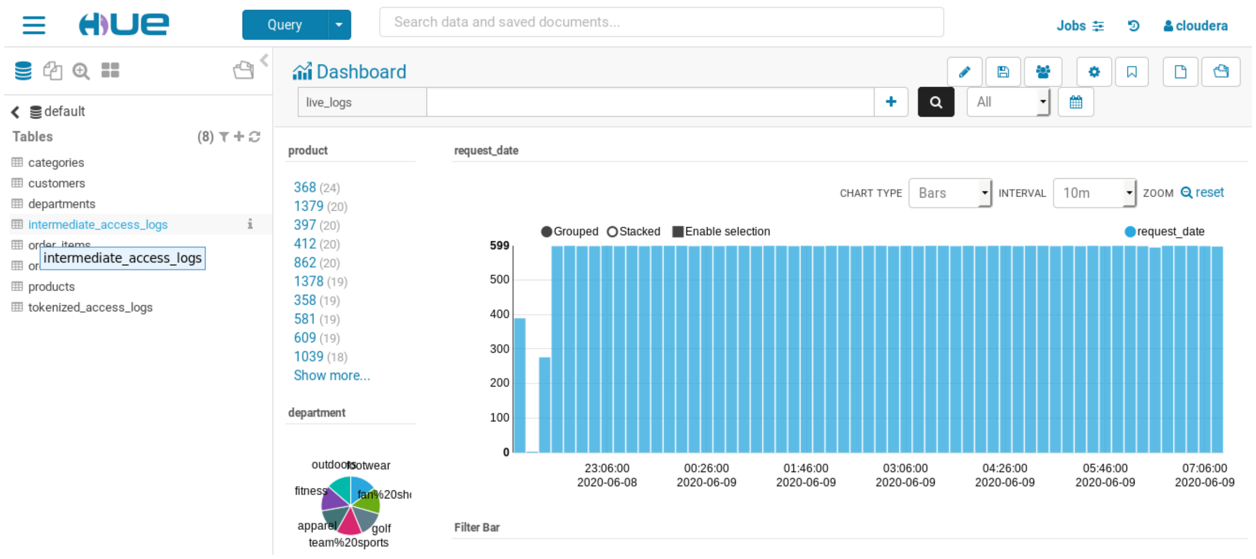


The saved dashboard will appear in the list. Click on the **live_logs** item to access the saved dashboard.



Name	Description	Type	Owner	Last Modified
live_logs		Search Dashboard	cloudera	06/09/2020 6:57 AM

The saved dashboard should now appear in the dashboards area.



368 (24)
1379 (20)
397 (20)
412 (20)
862 (20)
1378 (19)
358 (19)
581 (19)
609 (19)
1039 (18)
Show more...

The End Game

What Did We Learn?

We hope you have enjoyed this basic tutorial, and that you:

- Have a better understanding of some of the popular tools in CDH
- Know how to set up some basic and familiar BI use cases, as well as web log analytics and real-time search
- Are able to explain to your manager why you deserve a raise!