

Tutorial 01 – Conhecendo o Hadoop

Este tutorial foi escrito para funcionar com o *Cloudera Quickstart 5.13 CDH*, que é uma das distribuições *freemium* do *Hadoop* mais famosa.

Ele tem por objetivo demonstrar algumas ferramentas fornecidas pelo *CDH*, que é uma versão de código aberto do *Cloudera* com as principais ferramentas do *Apache Hadoop*.

Algumas respostas a dúvidas comuns sobre este tutorial:

- a) O *Cloudera Quickstart* foi escolhido porque não exige máquina dedicada e possui uma máquina virtual que pode rodar em apenas 4GB RAM, sendo ideal 6GB RAM.
- b) Algumas partes deste tutorial foram desabilitadas porque só funcionam no *Cloudera Express*, também gratuito, que exige pelo menos 8GB RAM e 2 CPUS. Estas partes também podem ser executadas no *Cloudera Express* que exige ao menos 10GB RAM e 2 CPUS.

No final deste tutorial você irá:

- > Compreender como usar algumas das ferramentas do *CDH*
- > Configurar e executar alguns casos básicos de inteligência de negócios e análise
- > Ser capaz de explicar ao seu gerente porque a empresa precisa lhe dar um aumento!

Início de Tudo: defina uma questão de Negócio, defina um Problema

Este tutorial irá apresentar exemplos no contexto de uma corporação chamada *GoData*. A nossa missão é ajudar a organização a obter uma visão melhor do seu negócio, fazendo e respondendo perguntas.

Cenário

Seu Gerente: esta empolgado e falando euforicamente sobre *Big Data* depois de assistir algumas palestras sobre *Data Science* ...

Você: esta cuidadosamente cético, pois provavelmente isto irá cair no seu “colo” de alguma forma. De certo modo isto já apareceu como sua nova missão com uma boa descrição do projeto: Vá entender esse tal de *Hadoop* ...

Dados Relacionais e a GoData

Nesse cenário, a pergunta de negócio da *GoData* é: **quais produtos os nossos clientes gostam de comprar?** Para responder a esta pergunta, o primeiro pensamento pode ser analisar os dados da transação, o que deve indicar o que os clientes realmente compram e gostam de comprar, certo?

Isto é provavelmente algo que você pode fazer em seu ambiente *RDBMS* regular, mas um benefício com a plataforma da *Cloudera (Hadoop)* é que você pode fazê-lo em maior escala a um custo menor, no mesmo sistema que você também pode usar para muitos outros tipos de análise.

O que este exercício demonstra é como fazer exatamente a mesma coisa que você já sabe como fazer com bancos de dados tradicionais, mas em *CDH*. A integração perfeita é importante ao avaliar qualquer nova infraestrutura. Portanto, é importante fazer o que você faz normalmente e não quebrar relatórios de *BI (Business Intelligence)* ou cargas de trabalho regulares sobre o conjunto de dados que você planeja migrar.

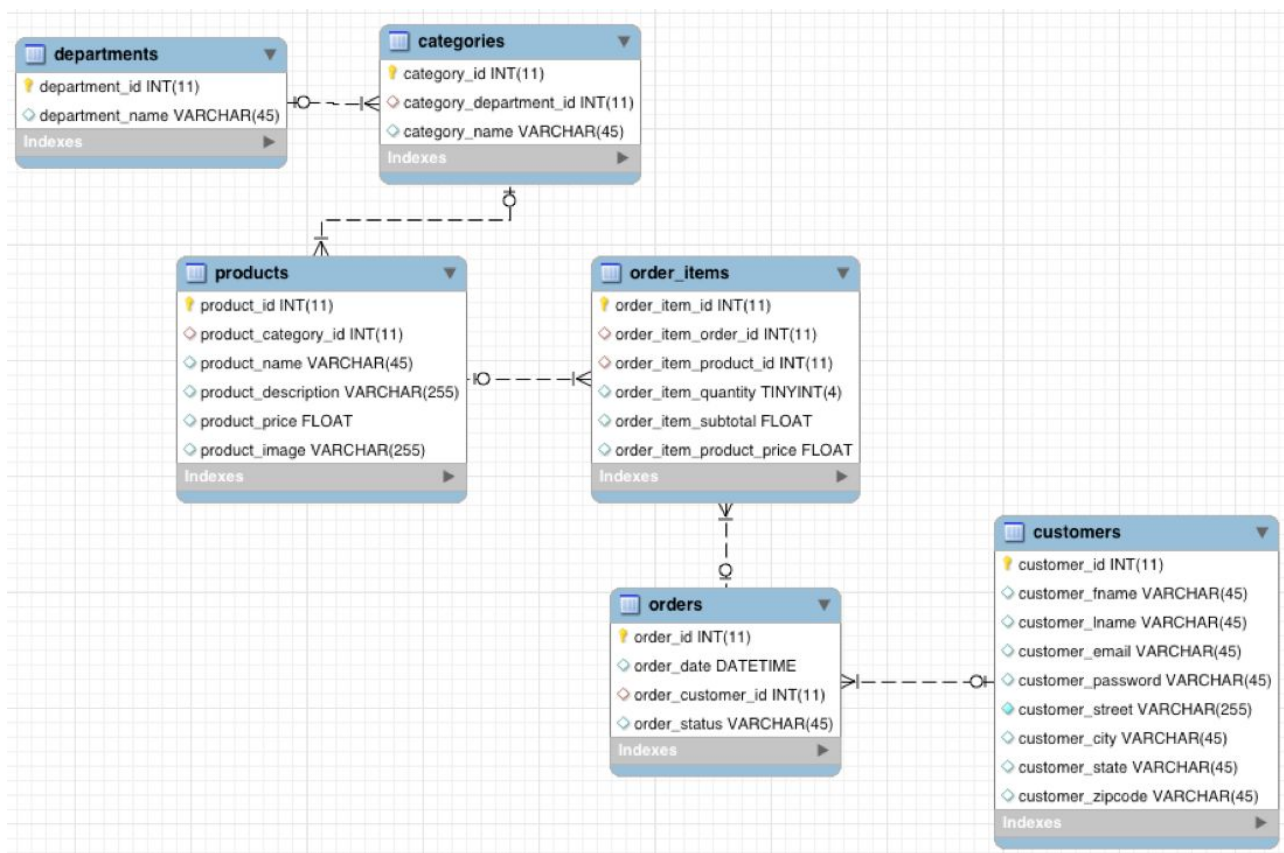


Figura 1: Modelo ER das tabelas do MySQL disponíveis no CDH

Para analisar os dados das transação na nova plataforma, precisamos inseri-las no sistema de arquivos distribuídos **Hadoop (HDFS)**. Precisamos encontrar uma ferramenta que transfira facilmente dados estruturados de um **RDBMS** para **HDFS**, preservando a estrutura. Isso nos permite consultar os dados, mas não interferir ou quebrar qualquer carga de trabalho normal nele.

Apache Sqoop, que é parte da **CDH**, é essa ferramenta. A coisa agradável sobre o **Sqoop** é que podemos carregar automaticamente nossos dados relacionais de qualquer banco de dados relacional, tal como o **MySQL** em **HDFS**, preservando a sua estrutura.

Com alguns parâmetros de configuração adicionais, podemos dar um passo adiante e carregar esses dados relacionais diretamente em um formulário pronto para ser consultado pelo **Apache Impala** (o mecanismo de consulta analítica de código aberto incluído com **CDH**). Dado que podemos querer aproveitar o poder do formato de arquivo do **Apache Avro** para outras cargas de trabalho no **cluster** (como o **Avro** é um formato de arquivo otimizado **Hadoop**), daremos algumas etapas extras para carregar esses dados no **Impala** usando o formato de arquivo **Avro**, por isso está prontamente disponível para **Impala**, bem como outras cargas de trabalho.

Primeiro, você deve abrir um terminal, o que você pode fazer, clicando no ícone "Terminal" preto na parte superior da tela. Uma vez que está aberto, você pode iniciar o trabalho do **Sqoop**:

```
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
```

Este comando irá demorar um pouco para ser concluído porque está executando muitas tarefas. Está lançando tarefas do **MapReduce** para extrair os dados do nosso banco de dados **MySQL** e escrevendo os dados no **HDFS**, que serão distribuídos em todo o **cluster** no formato **Apache Parquet**. Também está criando tabelas que representam os arquivos **HDFS** em **Impala** / **Apache Hive** com seu esquema correspondente.

Parquet é um formato concebido para **aplicações analíticas** no **Hadoop**. Em vez de **agrupar** seus dados em linhas como formatos de dados típicos, agrupa seus **dados em colunas**. Isso é ideal para muitas consultas analíticas em que, em vez de recuperar dados de registros específicos, você está analisando as relações entre variáveis específicas em vários registros. **Parquet** é projetado para otimizar o armazenamento de dados e recuperar informações.

Uma vez que o comando está completo, podemos confirmar que nossos dados foram importados para o HDFS digitando as seguintes linhas no terminal:

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

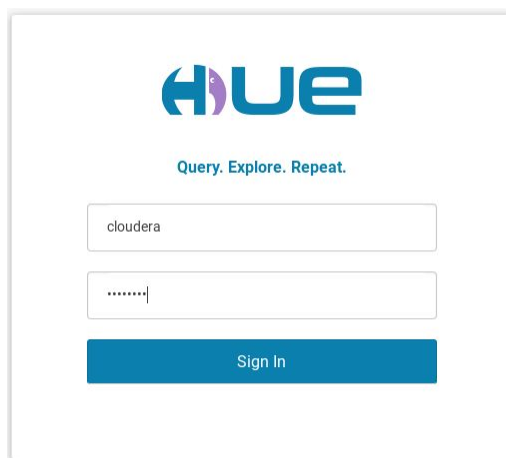
Esses comandos mostrarão os diretórios e os arquivos dentro deles que compõem nossas tabelas:

```
cloudera@quickstart:~  
File Edit View Search Terminal Help  
Merged Map outputs=0  
GC time elapsed (ms)=66  
CPU time spent (ms)=3200  
Physical memory (bytes) snapshot=389873664  
Virtual memory (bytes) snapshot=1606160384  
Total committed heap usage (bytes)=299892736  
File Input Format Counters  
  Bytes Read=0  
File Output Format Counters  
  Bytes Written=0  
17/11/22 04:50:52 INFO mapreduce.ImportJobBase: Transferred 46.1328 KB in 58.605  
7 seconds (806.0647 bytes/sec)  
17/11/22 04:50:52 INFO mapreduce.ImportJobBase: Retrieved 1345 records.  
[cloudera@quickstart ~]$ hadoop fs -ls /user/hive/warehouse/  
Found 6 items  
drwxrwxrwx - cloudera supergroup          0 2017-11-22 04:45 /user/hive/wareho  
use/categories  
drwxrwxrwx - cloudera supergroup          0 2017-11-22 04:46 /user/hive/wareho  
use/customers  
drwxrwxrwx - cloudera supergroup          0 2017-11-22 04:47 /user/hive/wareho  
use/departments  
drwxrwxrwx - cloudera supergroup          0 2017-11-22 04:48 /user/hive/wareho  
use/order_items  
drwxrwxrwx - cloudera supergroup          0 2017-11-22 04:49 /user/hive/wareho  
use/orders  
drwxrwxrwx - cloudera supergroup          0 2017-11-22 04:50 /user/hive/wareho  
use/products  
[cloudera@quickstart ~]$ hadoop fs -ls /user/hive/warehouse/categories/  
Found 5 items  
drwxr-xr-x - cloudera supergroup          0 2017-11-07 17:13 /user/hive/wareho  
use/categories/.metadata  
drwxr-xr-x - cloudera supergroup          0 2017-11-22 04:45 /user/hive/wareho  
use/categories/.signals  
-rw-r--r--  1 cloudera supergroup        1957 2017-11-22 04:45 /user/hive/wareho  
use/categories/28f1d898-a714-4185-ac95-d7984ecc11bf.parquet  
-rw-r--r--  1 root supergroup            1957 2017-11-07 17:32 /user/hive/wareho  
use/categories/8fb55696-4788-4d64-971e-b9429b3d2d1f.parquet  
-rw-r--r--  1 cloudera supergroup        1957 2017-11-07 17:14 /user/hive/wareho  
use/categories/d679753f-5048-41a1-b77c-a92a9e5d6d94.parquet  
[cloudera@quickstart ~]$
```

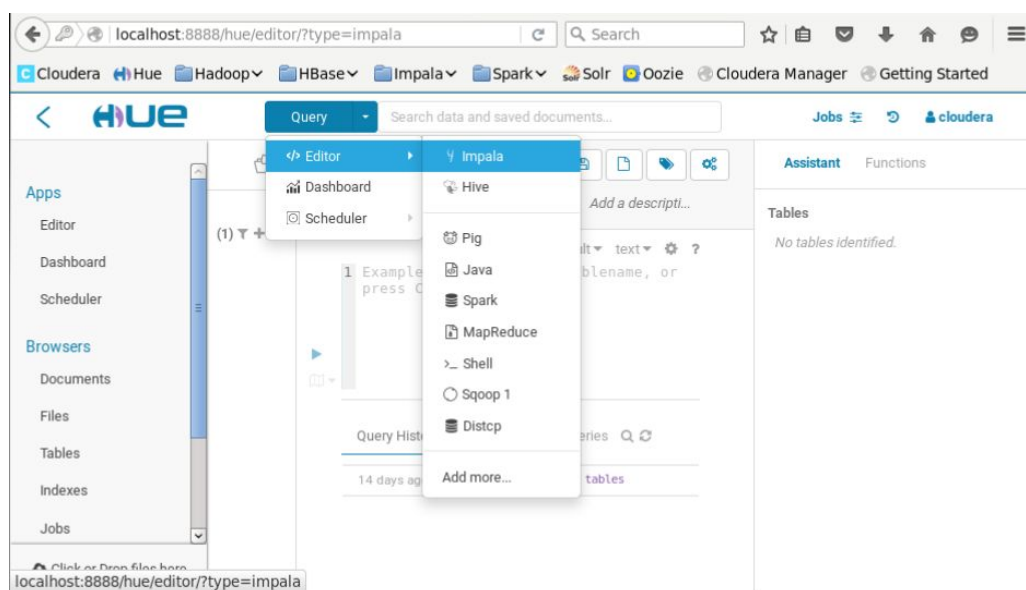
Nota: O número de arquivos *.parquet* mostrados será igual ao número de mapeadores usados pelo *Sqoop*. Em um único nó, você verá apenas um, mas clusters maiores terão um número maior de arquivos.

[Apache Hive](#) e [Apache Impala](#) também permitem que você crie tabelas definindo um esquema sobre arquivos existentes com instruções '**CREATE EXTERNAL TABLE**', semelhantes aos bancos de dados relacionais tradicionais. Mas o **Sqoop** já criou essas tabelas para nós, para que possamos prosseguir e consultá-las.

Vamos usar o aplicativo **Impala** de **Hue** para consultar nossas tabelas. O [Hue \(Hadoop User Experience\)](#) fornece uma interface baseada na web para muitas das ferramentas em *CDH* e pode ser encontrada na porta 8888 do Nó do Gerenciador (<http://localhost:8888>). Na VM do *QuickStart*, o nome de usuário do **administrador** para **Hue** é '**cloudera**' e a **senha** é '**cloudera**'.

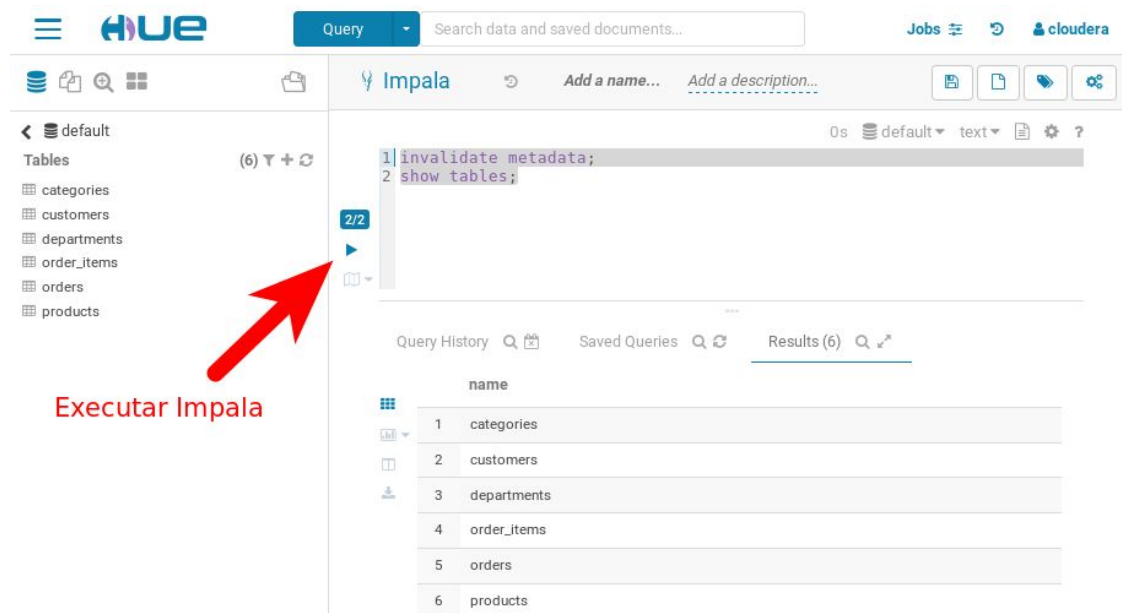


Uma vez que você estiver dentro da *Hue*, clique em Query -> Editor e abra o Impala Query Editor.



Para economizar tempo durante as consultas, o Impala não pesquisa constantemente as alterações de metadados. Então, a primeira coisa que devemos fazer é dizer ao Impala que seus metadados estão desatualizados. Então, devemos ver nossas mesas aparecerem, prontas para serem consultadas. Copie e cole os comandos abaixo no Impala e execute pressionando no botão ▶

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



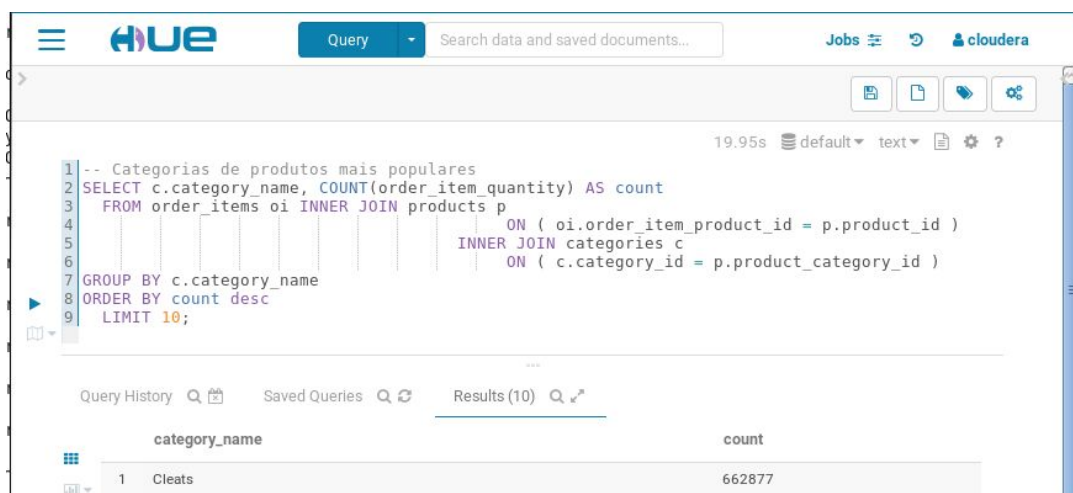
Executar Impala

	name
1	categories
2	customers
3	departments
4	order_items
5	orders
6	products

Agora que seus dados de transação estão prontamente disponíveis para consultas estruturadas em CDH, é hora de abordar a questão comercial da GoData. Copie e cole ou escreva as seguintes consultas de exemplo SQL padrão para calcular a receita total por produto e mostrando os 10 principais produtos geradores de receita:

```
-- Categorias de produtos mais populares  
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;
```

Você deve ver resultados semelhantes a este:

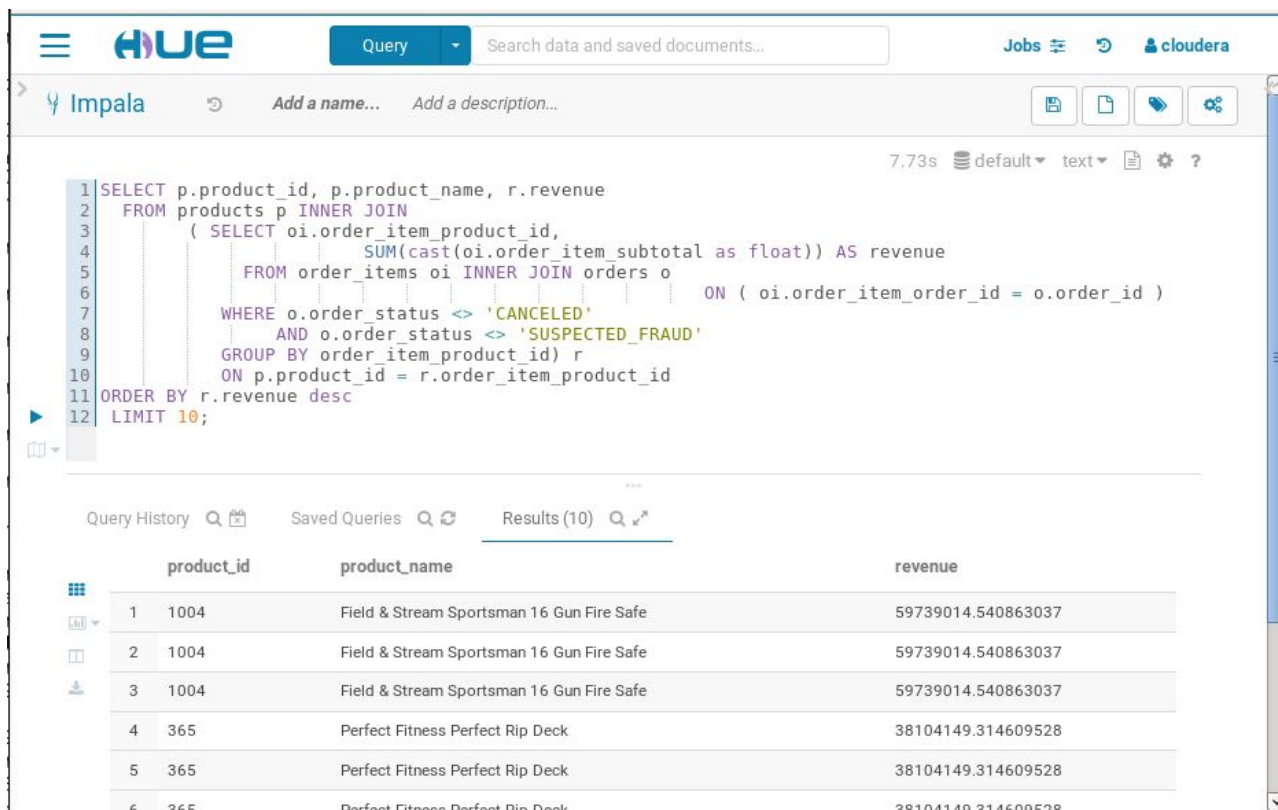


category_name	count
1 Cleats	662877

Limpe a consulta anterior e substitua-a pelo seguinte:

```
-- 10 produtos que mais geram receita
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;
```

Você deve ver resultados semelhantes a este:



The screenshot shows the Hue web interface. At the top, there's a navigation bar with the Hue logo, a 'Query' dropdown, a search bar, and links for 'Jobs', 'cloudera', and a user profile. Below this is a toolbar with icons for saving, refreshing, and other actions. The main area displays a SQL query in a text editor, with line numbers 1 through 12. The query is the same as the one provided in the previous block. Below the query editor, there's a section for 'Query History', 'Saved Queries', and 'Results (10)'. The 'Results (10)' section is active, showing a table with 4 columns: 'product_id', 'product_name', and 'revenue'. The table contains 6 rows of data, with the first row having a 'product_id' of 1004 and a 'revenue' of 59739014.540863037. The second row has a 'product_id' of 1004 and a 'revenue' of 59739014.540863037. The third row has a 'product_id' of 1004 and a 'revenue' of 59739014.540863037. The fourth row has a 'product_id' of 365 and a 'revenue' of 38104149.314609528. The fifth row has a 'product_id' of 365 and a 'revenue' of 38104149.314609528. The sixth row has a 'product_id' of 365 and a 'revenue' of 38104149.314609528.

	product_id	product_name	revenue
1	1004	Field & Stream Sportsman 16 Gun Fire Safe	59739014.540863037
2	1004	Field & Stream Sportsman 16 Gun Fire Safe	59739014.540863037
3	1004	Field & Stream Sportsman 16 Gun Fire Safe	59739014.540863037
4	365	Perfect Fitness Perfect Rip Deck	38104149.314609528
5	365	Perfect Fitness Perfect Rip Deck	38104149.314609528
6	365	Perfect Fitness Perfect Rip Deck	38104149.314609528

Você deve ter notado que dissemos ao *Sqoop* que importasse os dados para o *Hive*, mas usou o *Impala* para consultar os dados. Isso ocorre porque o *Hive* e *Impala* podem compartilhar os arquivos de dados e os metadados da tabela. O *Hive* funciona compilando consultas SQL nas tarefas do MapReduce, o que o torna muito flexível, ao passo que o *Impala* executa as próprias consultas e é construído desde o início para ser o mais rápido possível, o que o torna melhor para a análise interativa. Nós usaremos o *Hive* mais tarde para uma carga de trabalho *ETL* (extração-transformação-carga).

CONCLUSÃO

Agora você passou pelas primeiras etapas básicas para os dados estruturados do *Sqoop* em *HDFS* transformando no formato de arquivo Avro (você pode ler sobre os benefícios do Avro como um formato comum no Hadoop [aqui](#)) e importar os arquivos de esquema para uso quando consultamos esses dados.

Também aprendemos a criar e consultar tabelas usando *Impala* e que podem usar interfaces e ferramentas regulares (como SQL) dentro de um ambiente *Hadoop* também. A idéia é que você possa desenvolver os mesmos relatórios que esta acostumado, mas onde a arquitetura dos sistemas tradicionais *Hadoop* oferecem uma escala e flexibilidade muito maiores.