cloudera
Ask Bigger Questions

Introduction to Hive

Chapter 9

Course Chapters

- Introduction
- Hadoop Fundamentals
- Introduction to Pig
- Basic Data Analysis with Pig
- Processing Complex Data with Pig
- Multi-Dataset Operations with Pig
- Extending Pig
- Pig Troubleshooting and Optimization
- **Introduction to Hive**
- Relational Data Analysis with Hive
- Hive Data Management
- Text Processing With Hive
- Hive Optimization
- Extending Hive
- Introduction to Impala
- Analyzing Data with Impala
- Choosing the Best Tool for the Job
- Conclusion

Introduction to Hive

In this chapter, you will learn

- **What Hive is**
- **How Hive differs from a relational database**
- **Ways in which organizations use Hive**
- **How to invoke and interact with Hive**

Chapter Topics

Introduction to Hive

- **What is Hive?**
- Hive Schema and Data Storage
- Comparing Hive to Traditional Databases
- Hive Use Cases
- Interacting with Hive
- Conclusion

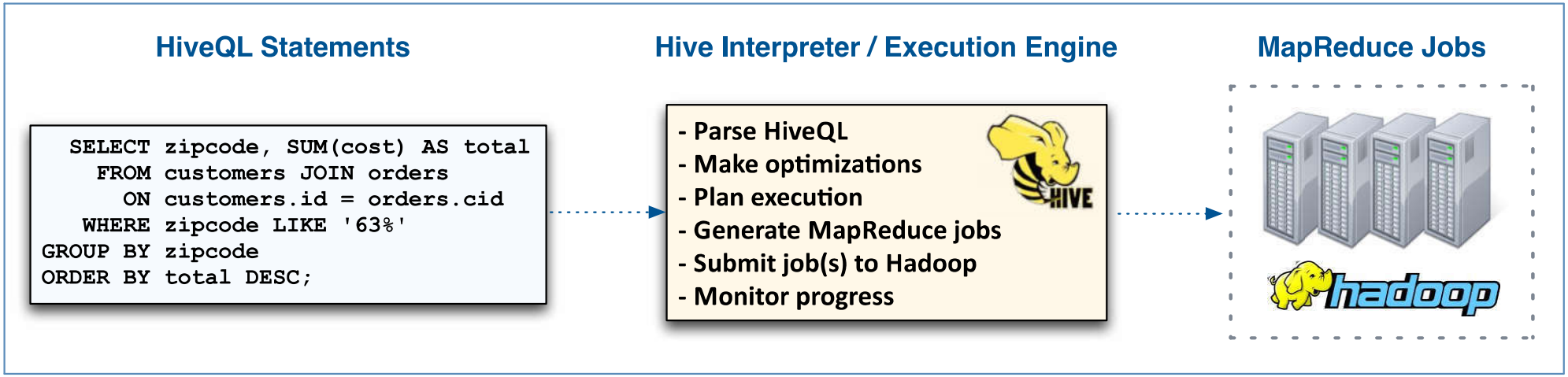
Overview of Apache Hive

- **Apache Hive is a high-level abstraction on top of MapReduce**
  - Uses a SQL-like language called HiveQL
  - Generates MapReduce jobs that run on the Hadoop cluster
  - Originally developed by Facebook for data warehousing
    - Now an open-source Apache project


```
  SELECT zipcode, SUM(cost) AS total
    FROM customers
    JOIN orders
      ON customers.cust_id = orders.cust_id
   WHERE zipcode LIKE '63%'
GROUP BY zipcode
ORDER BY total DESC;
```

High-Level Overview for Hive Users

- **Hive runs on the client machine**
  - Turns HiveQL queries into MapReduce jobs
  - Submits those jobs to the cluster



Why Use Apache Hive?

- **More productive than writing MapReduce directly**
  - Five lines of HiveQL might be equivalent to 100 lines or more of Java
- **Brings large-scale data analysis to a broader audience**
  - No software development experience required
  - Leverage existing knowledge of SQL
- **Offers interoperability with other systems**
  - Extensible through Java and external scripts
  - Many business intelligence (BI) tools support Hive

Where to Get Hive

- **CDH (Cloudera's Distribution including Apache Hadoop) is the easiest way to install Hadoop and Hive**
  - A Hadoop distribution which includes core Hadoop, Pig, Hive, Sqoop, HBase, Oozie, and other ecosystem components
  - Available as RPMs, Ubuntu/Debian/SuSE packages, or a tarball
  - Simple installation
  - 100% free and open source
- **Installation is outside the scope of this course**
  - Cloudera offers a training course for System Administrators, *Cloudera Administrator Training for Apache Hadoop*

Chapter Topics

Introduction to Hive

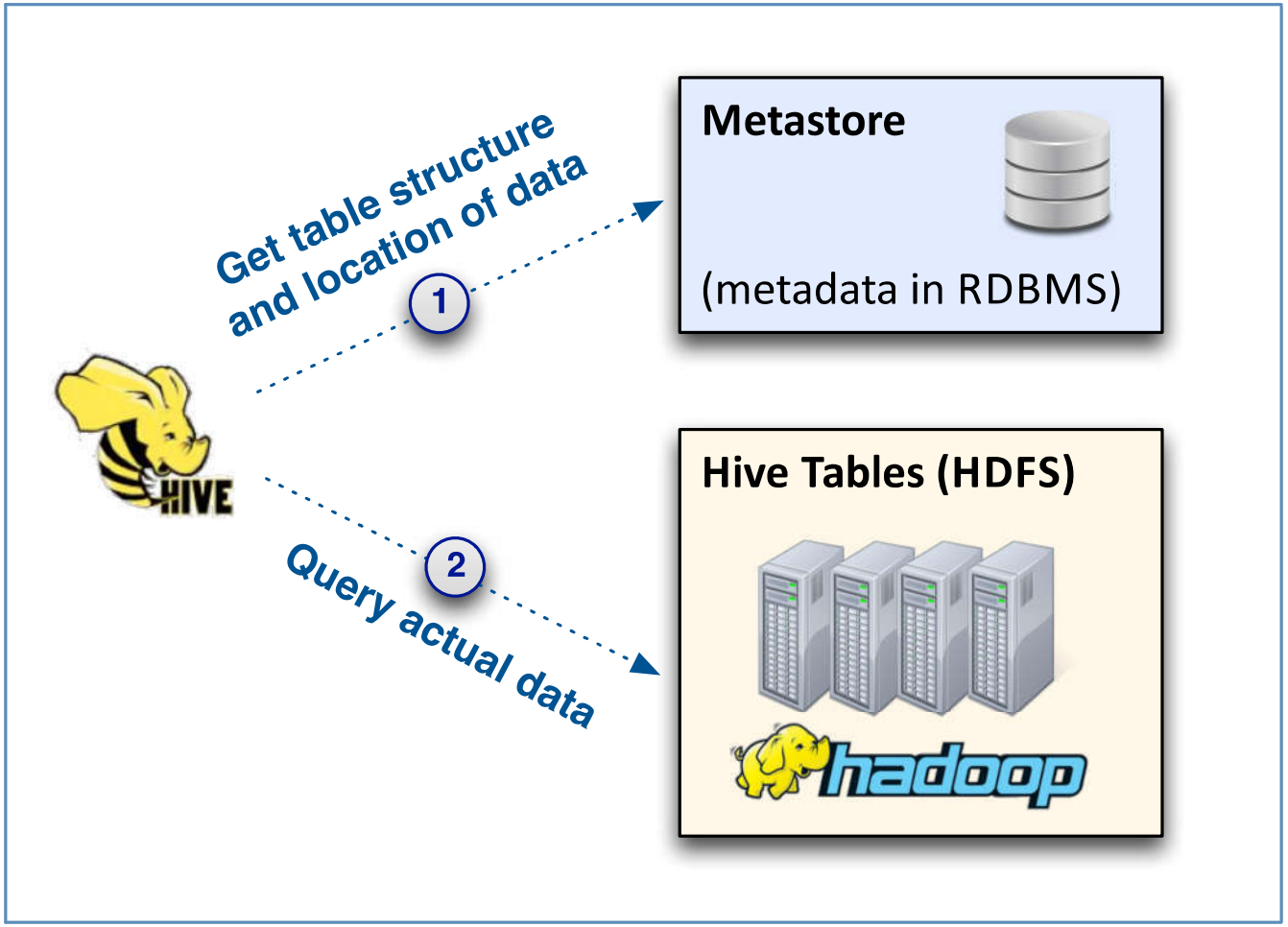
- What is Hive?
- **Hive Schema and Data Storage**
- Comparing Hive to Traditional Databases
- Hive Use Cases
- Interacting with Hive
- Conclusion

How Hive Loads and Stores Data (1)

- **Hive's queries operate on tables, just like in an RDBMS**
  - A table is simply an HDFS directory containing one or more files
  - Default path: `/user/hive/warehouse/<table_name>`
  - Hive supports many formats for data storage and retrieval
- **How does Hive know the structure and location of tables?**
  - These are specified when tables are created
  - This metadata is stored in Hive's *metastore*
    - Contained in an RDBMS such as MySQL

How Hive Loads and Stores Data (2)

- **Hive consults the metastore to determine data format and location**
  - The query itself operates on data stored on a filesystem (typically HDFS)



Chapter Topics

Introduction to Hive

- What is Hive?
- Hive Schema and Data Storage
- **Comparing Hive to Traditional Databases**
- Hive Use Cases
- Interacting with Hive
- Conclusion

Your Cluster is Not a Database Server

- **Client-server database management systems have many strengths**
  - Very fast response time
  - Support for transactions
  - Allows modification of existing records
  - Can serve thousands of simultaneous clients
- **Hive does not turn your Hadoop cluster into an RDBMS**
  - It simply produces MapReduce jobs from HiveQL queries
  - Limitations of HDFS and MapReduce still apply

Comparing Hive To A Relational Database

	Relational Database	Hive
Query language	SQL	HiveQL
Update individual records	Yes	No
Delete individual records	Yes	No
Transactions	Yes	No
Index support	Extensive	Limited
Latency	Very low	High
Data size	Terabytes	Petabytes

Chapter Topics

Introduction to Hive

- What is Hive?
- Hive Schema and Data Storage
- Comparing Hive to Traditional Databases
- **Hive Use Cases**
- Interacting with Hive
- Conclusion

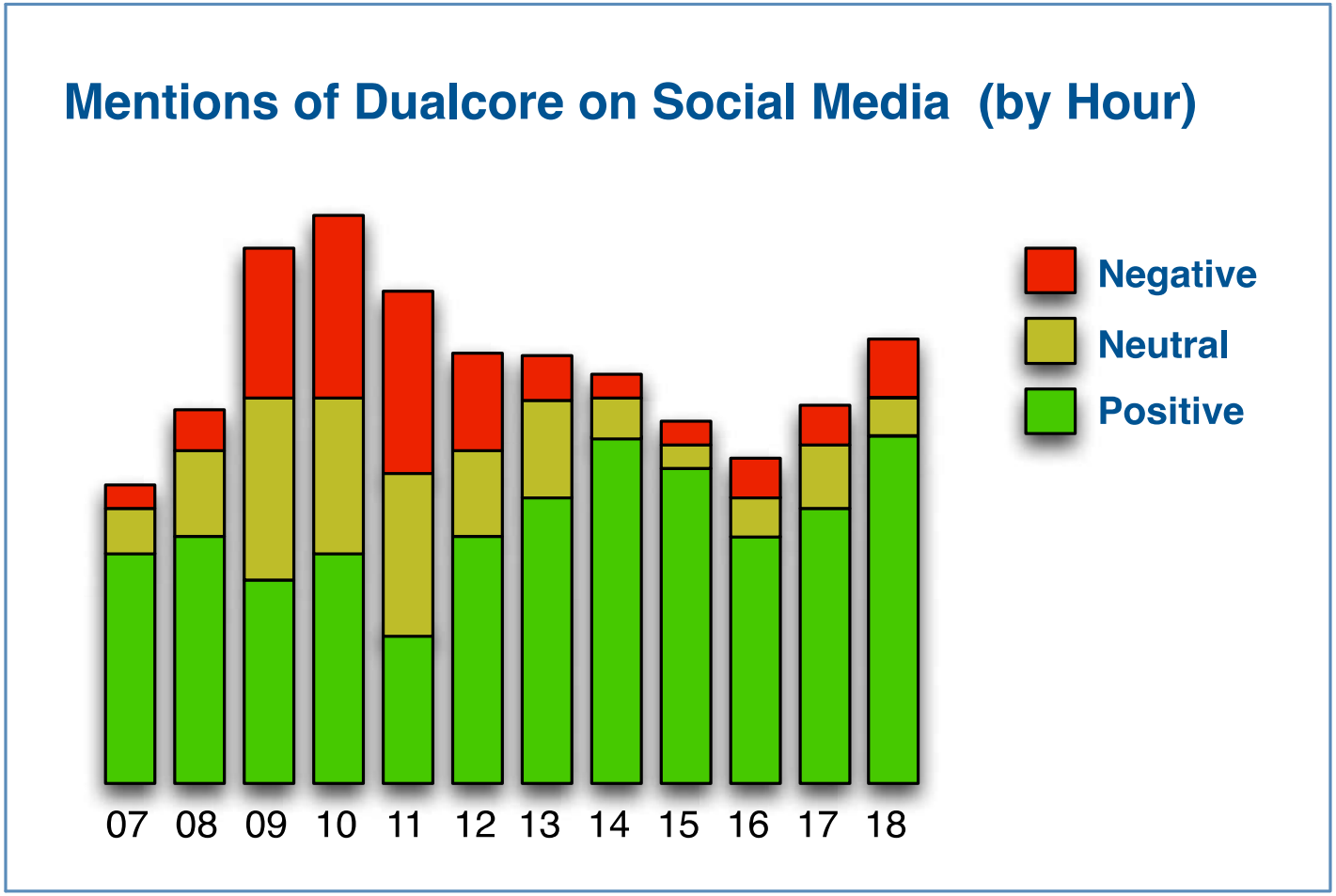
Use Case: Log File Analytics

- **Server log files are an important source of data**
- **Hive allows you to treat a directory of log files like a table**
  - Allows SQL-like queries against raw data

Dualcore Inc. Public Web Site (June 1 - 8)					
Product	**Unique Visitors**	**Page Views**	**Average Time on Page**	**Bounce Rate**	**Conversion Rate**
Tablet	5,278	5,894	17 seconds	23%	65%
Notebook	4,139	4,375	23 seconds	47%	31%
Stereo	2,873	2,981	42 seconds	61%	12%
Monitor	1,749	1,862	26 seconds	74%	19%
Router	987	1,139	37 seconds	56%	17%
Server	314	504	53 seconds	48%	28%
Printer	86	97	34 seconds	27%	64%

Use Case: Sentiment Analysis

- **Many organizations use Hive to analyze social media coverage**



Chapter Topics

Introduction to Hive

- What is Hive?
- Hive Schema and Data Storage
- Comparing Hive to Traditional Databases
- Hive Use Cases
- **Interacting with Hive**
- Conclusion

Using the Hive Shell

- **You can execute HiveQL statements in the Hive Shell**
  - This interactive tool is similar to the MySQL shell
- **Run the `hive` command to start the Hive shell**
  - The Hive shell will display its `hive>` prompt
  - Each statement must be terminated with a semicolon
  - Use the `quit` command to exit the Hive shell

```
$ hive
hive> SELECT cust_id, fname, lname
      FROM customers WHERE zipcode=20525;
1000000    Quentin    Shepard
1000001    Brandon    Louis
1000002    Marilyn    Ham
hive> quit;
$
```

Accessing Hive from the Command Line

- **You can also execute a file containing HiveQL code using the `-f` option**

```
$ hive -f myquery.hql
```

- **Or use HiveQL directly from the command line using the `-e` option**

```
$ hive -e 'SELECT * FROM users'
```

- **Use the `-S` (silent) option to suppress informational messages**
  - Can also be used with `-e` or `-f` options

```
$ hive -S
```

Hive Properties

- **Many aspects of Hive's behavior are configured through properties**
  - Use `set -v` in Hive to see current values

```
hive> set -v;
```

- **You can also use set to specify property values**
  - The following enables columns headers in query results

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

- **Hive runs the `.hiverc` file in your home directory at startup**
  - Useful for specifying per-user defaults

Interacting with the Operating System and HDFS

- **Use `!` to execute system commands from within Hive**
  - Neither pipes nor globs (wildcards) are supported

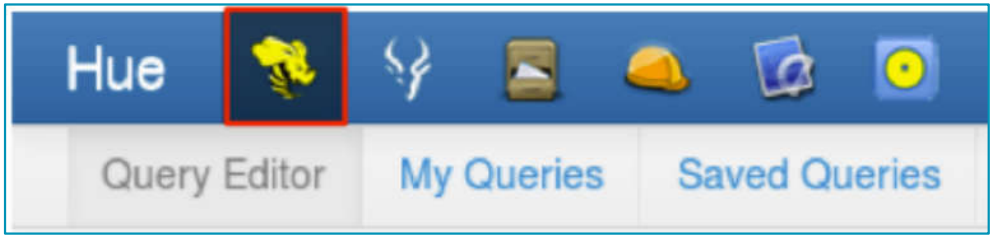
```
hive> ! date;
Mon May 20 16:44:35 PDT 2013
```

- **Prefix HDFS commands with `dfs` to use them from within Hive**

```
hive> dfs -mkdir /reports/sales/2013;
```

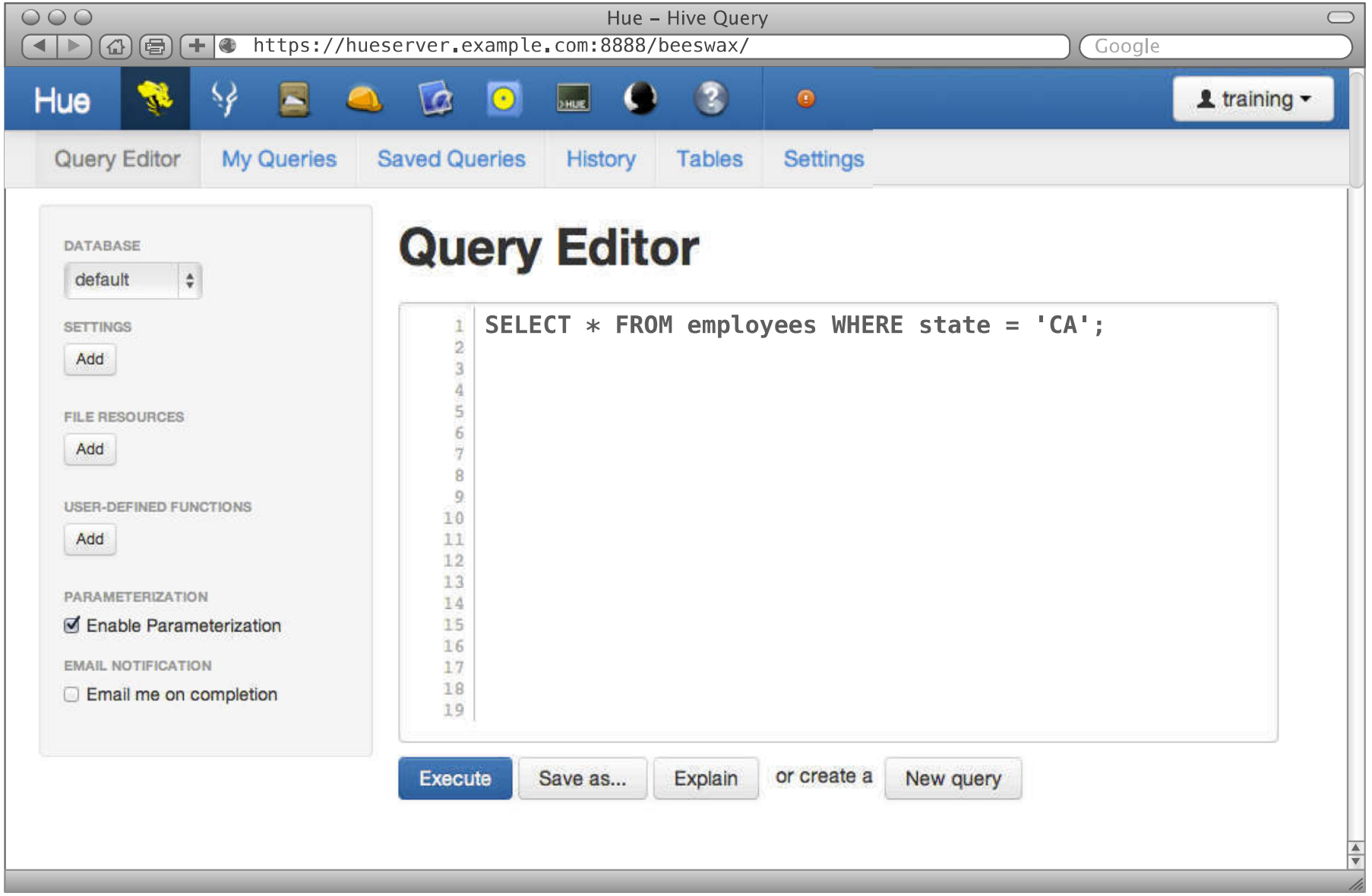
Accessing Hive with Hue

- **Alternatively, you can access Hive through Hue**
  - Web-based UI for many Hadoop-related services, including Hive
- **To use Hue, browse to `http://hue_server:8888/`**
  - May need to start Hue service first (`sudo service hue start`)
- **Hue's Hive interface is called Beeswax**
  - Launch by clicking its icon
- **Beeswax features include:**
  - Creating tables
  - Running queries
  - Browsing tables
  - Saving queries for later execution

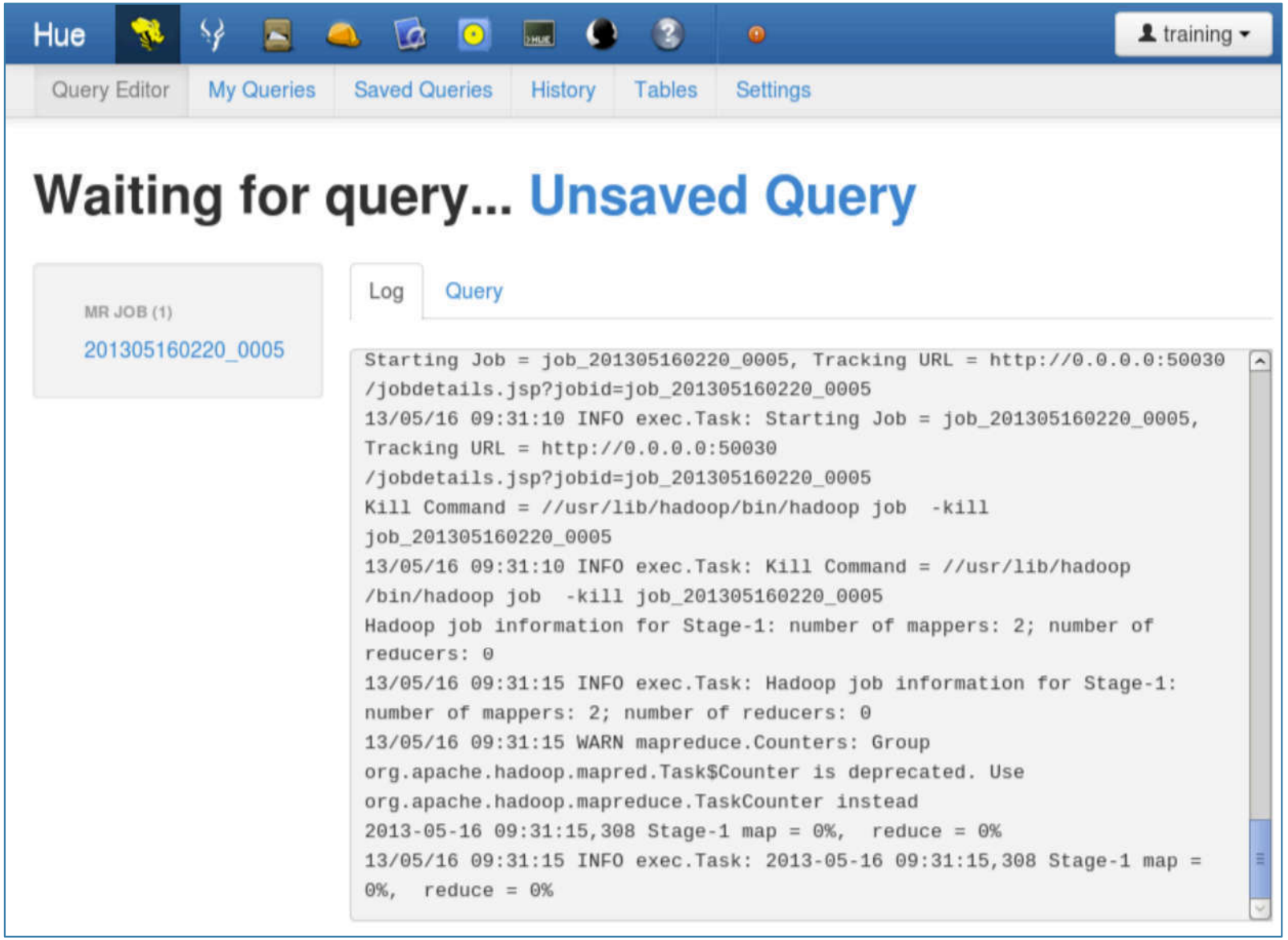


Beeswax icon in Hue

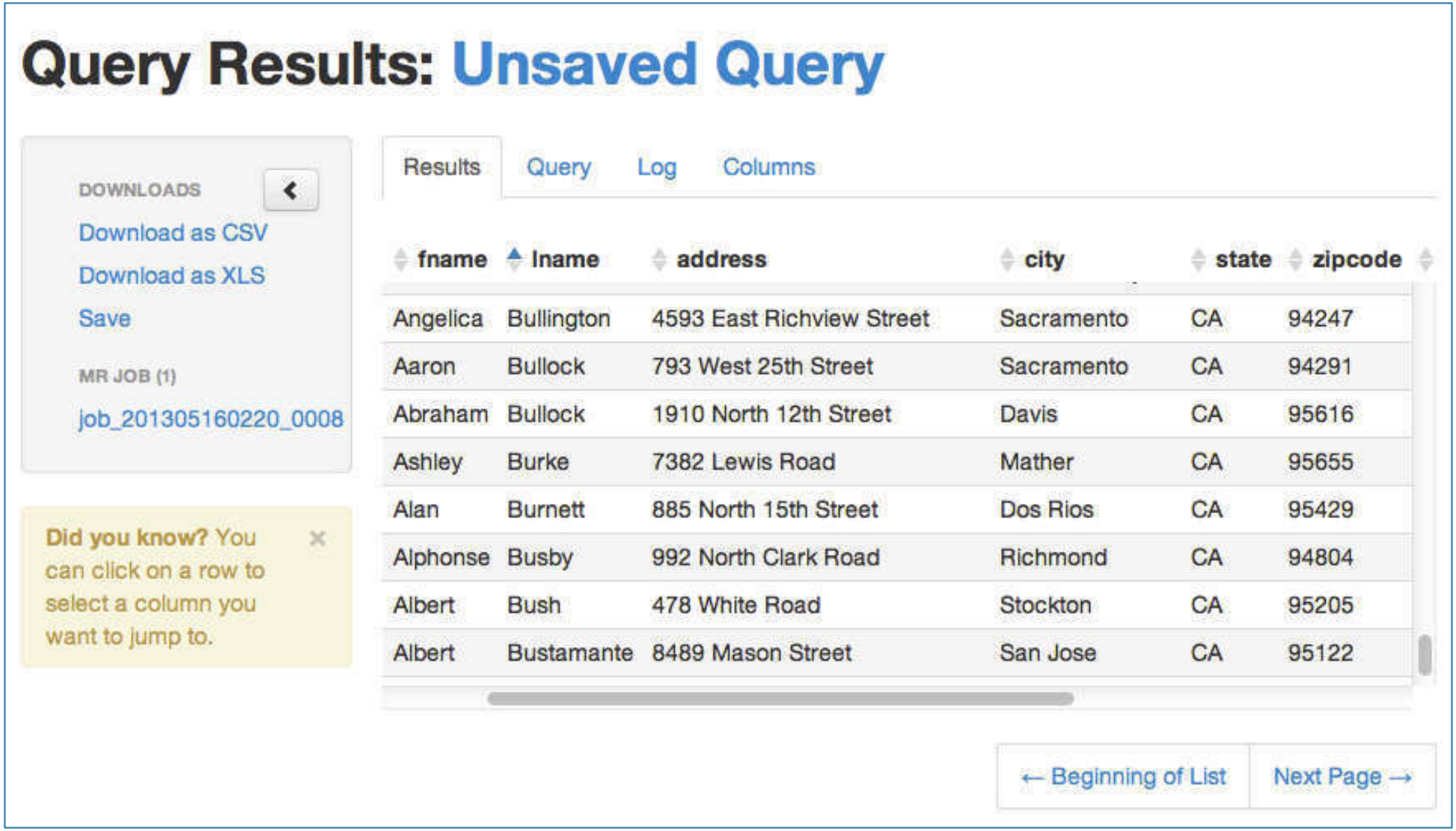
Hue's Beeswax Query Editor



Query Execution in Beeswax

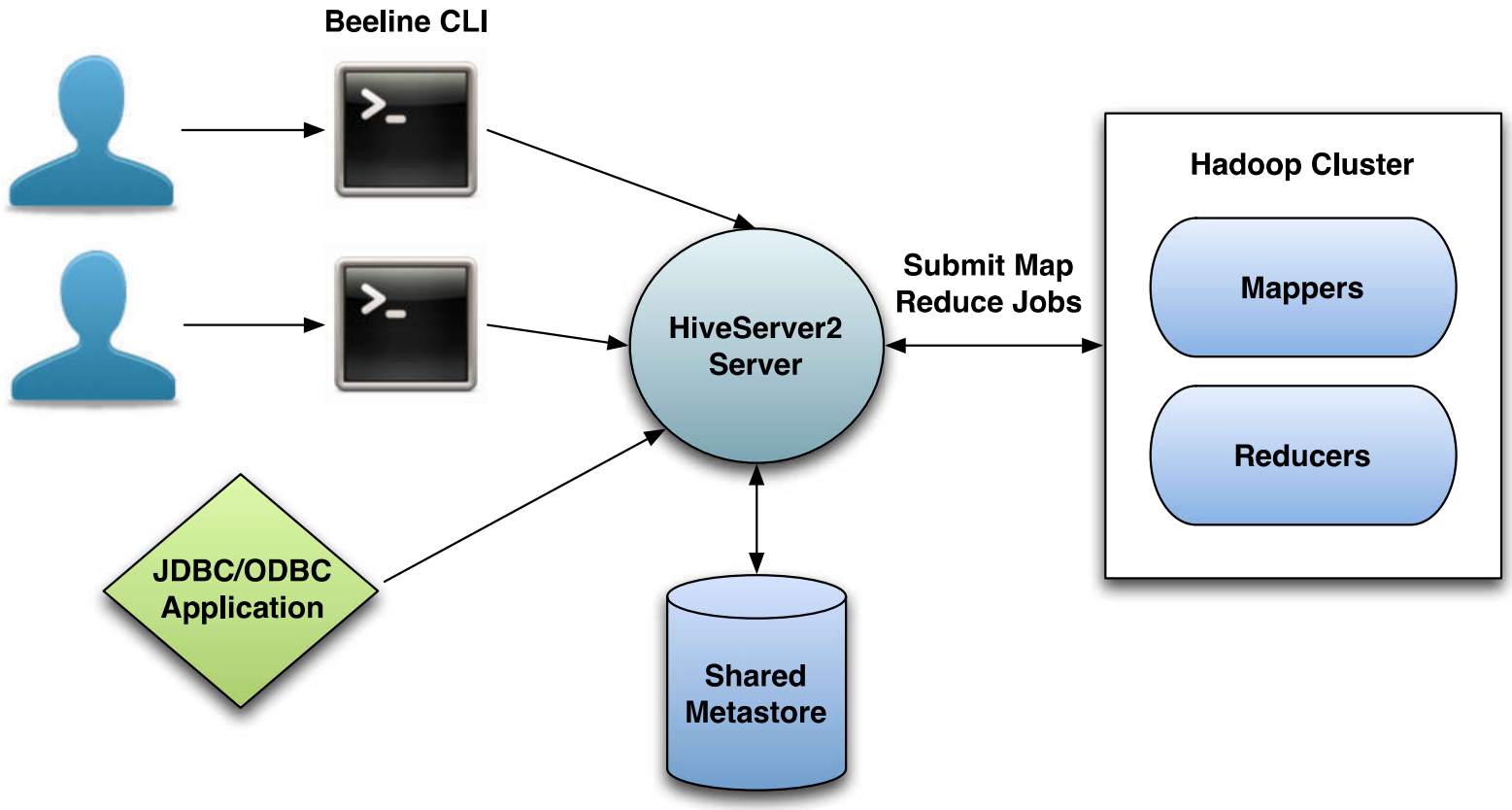


Query Results in Beeswax



Interacting With HiveServer2 – Hive as a Service (1)

- **HiveServer2 can be deployed to provide a centralized Hive service**
  - Uses a JDBC or ODBC connection
  - Supports Kerberos authentication



Interacting With HiveServer2 – Hive as a Service (2)

- **To connect to HiveServer2, use Hue or the Beeline CLI**
  - You cannot use the Hive shell
  - For secure deployments, supply your user ID and password
- **Example: starting Beeline and connecting to HiveServer2**

```
[training@localhost analyst]$ beeline
Beeline version 0.10.0-cdh4.2.1 by Apache Hive

beeline> !connect jdbc:hive2://localhost:10000
training mypwd org.apache.hive.jdbc.HiveDriver

Connecting to jdbc:hive2://localhost:10000
Connected to: Hive (version 0.10.0)
Driver: Hive (version 0.10.0-cdh4.2.1)
Transaction isolation: TRANSACTION_REPEATABLE_READ

beeline>
```

Chapter Topics

Introduction to Hive

- What is Hive?
- Hive Schema and Data Storage
- Comparing Hive to Traditional Databases
- Hive Use Cases
- Interacting with Hive
- **Conclusion**

Essential Points

- **Hive is a high-level abstraction on top of MapReduce**
  - Runs MapReduce jobs on Hadoop based on HiveQL statements
  - Originally developed for data warehousing at Facebook
- **HiveQL is very similar to SQL**
  - Easy to learn for those with relational database experience
  - However, Hive does *not* replace your RDBMS
- **Hive tables are really directories of files in HDFS**
  - Information about those tables is kept in Hive's metastore
- **Hive and Pig are similar in many ways**
  - But each also has its own distinct advantages

Bibliography

The following offer more information on topics discussed in this chapter

- **Hive Web Site**
  - http://hive.apache.org/
- **Beeline CLI reference**
  - http://sqlline.sourceforge.net/
- **Programming Hive (book)**
  - http://tiny.cloudera.com/dac09a
- **Data Analysis with Hadoop and Hive at Orbitz**
  - http://tiny.cloudera.com/dac09b
- **Sentiment Analysis Using Apache Hive**
  - http://tiny.cloudera.com/dac09c

cloudera
Ask Bigger Questions

Relational Data Analysis with Hive

Chapter 10

Course Chapters

- Introduction
- Hadoop Fundamentals
- Introduction to Pig
- Basic Data Analysis with Pig
- Processing Complex Data with Pig
- Multi-Dataset Operations with Pig
- Extending Pig
- Pig Troubleshooting and Optimization
- Introduction to Hive
- **Relational Data Analysis with Hive**
- Hive Data Management
- Text Processing With Hive
- Hive Optimization
- Extending Hive
- Introduction to Impala
- Analyzing Data with Impala
- Choosing the Best Tool for the Job
- Conclusion

Relational Data Analysis with Hive

In this chapter, you will learn

- **How to explore databases and tables in Hive**
- **How HiveQL syntax compares to SQL**
- **Which data types Hive supports**
- **Which types of join operations Hive supports and how to use them**
- **How to use many of Hive's built-in functions**

Chapter Topics

Relational Data Analysis with Hive

- **Hive Databases and Tables**
- Basic HiveQL Syntax
- Data Types
- Joining Datasets
- Common Built-in Functions
- Hands-On Exercise: Running Hive Queries from the Shell, Scripts, and Hue
- Conclusion

Hive Tables

- **Data for Hive's tables is stored on the filesystem (typically HDFS)**
  - Each table maps to a single directory
- **A table's directory may contain multiple files**
  - Typically delimited text files, but Hive supports many formats
  - Subdirectories are not allowed
- **Hive uses the metastore to give context to this data**
  - Helps map raw data in HDFS to named columns of specific types

Hive Databases

- **Each Hive table belongs to a specific database**
- **Early versions of Hive supported only a single database**
  - It placed all tables in the same database (named `default`)
  - This is still the default behavior
- **Hive supports multiple databases as of release 0.7.0**
  - Helpful for organization and authorization

Exploring Hive Databases and Tables (1)

- **Which databases are available?**

```
hive> SHOW DATABASES;
accounting
default
sales
```

- **Switch between databases with the `USE` command**

```
$ hive
hive> SELECT * FROM customers;
hive> USE sales;
hive> SELECT * FROM customers;
```

Queries table in the `default` database

Queries table in the `sales` database

Exploring Hive Databases and Tables (2)

- **Which tables does the current database contain?**

```
hive> USE accounting;
hive> SHOW TABLES;
invoices
taxes
```

- **Which tables are contained in a different database?**

```
hive> SHOW TABLES IN sales;
customers
prospects
```

Exploring Hive Databases and Tables (3)

- The `DESCRIBE` command displays basic structure for a table

```
hive> DESCRIBE orders;
order_id    int
cust_id     int
order_date  timestamp
```

- `DESCRIBE FORMATTED` shows more detailed information

```
hive> DESCRIBE FORMATTED orders;
# col_name          data_type          comment
order_id            int                None
cust_id             int                None
order_date          timestamp          None
# Detailed Table Information    ... More follows...
```

Chapter Topics

Relational Data Analysis with Hive

- Hive Databases and Tables
- **Basic HiveQL Syntax**
- Data Types
- Joining Datasets
- Common Built-in Functions
- Hands-On Exercise: Running Hive Queries from the Shell, Scripts, and Hue
- Conclusion

An Introduction To HiveQL

- **HiveQL is Hive's query language**
  - Based on a subset of SQL-92, plus Hive-specific extensions
- **Some limitations compared to 'standard' SQL**
  - Some features are not supported
  - Others are only partially implemented
- **HiveQL also has some features not offered in SQL**

HiveQL Basics

- **Hive keywords are not case-sensitive**
  - Though they are often capitalized by convention
- **Statements are terminated by a semicolon**
  - A statement may span multiple lines
- **Comments begin with --** ***(double hyphen)***
  - Only supported in Hive scripts
  - There are no multi-line comments in Hive

```
$ cat myscript.hql

SELECT cust_id, fname, lname
   FROM customers
   WHERE zipcode='60601';     -- downtown Chicago
```

Selecting Data from Hive Tables

- **The `SELECT` statement retrieves data from Hive tables**
  - Can specify an ordered list of individual columns

```
hive> SELECT cust_id, fname, lname FROM customers;
```

  - An asterisk matches all columns in the table

```
hive> SELECT * FROM customers;
```

Limiting and Sorting Query Results

- **The `LIMIT` clause sets the maximum number of rows returned**

```
hive> SELECT fname, lname FROM customers LIMIT 10;
```

- **Caution: no guarantee regarding *which* 10 results are returned**
  - Use `ORDER BY` for top-N queries
  - The field(s) you `ORDER BY` must be selected

```
hive> SELECT cust_id, fname, lname FROM customers
      ORDER BY cust_id DESC LIMIT 10;
```

Using a WHERE Clause to Restrict Results

- **WHERE clauses restrict rows to those matching specified criteria**
  - String comparisons are case-sensitive

```
hive> SELECT * FROM orders WHERE order_id=1287;
```

```
hive> SELECT * FROM customers WHERE state
      IN ('CA', 'OR', 'WA', 'NV', 'AZ');
```

- **You can combine expressions using AND or OR**

```
hive> SELECT * FROM customers
      WHERE fname LIKE 'Ann%'
      AND (city='Seattle' OR city='Portland');
```

Table Aliases

- **Table aliases can help simplify complex queries**

```
hive> SELECT o.order_date, c.fname, c.lname
      FROM customers c JOIN orders o
      ON c.cust_id = o.cust_id
      WHERE c.zipcode='94306';
```

  - Note: Using `AS` to specify table aliases is *not* supported

Combining Query Results with `UNION ALL`

- **Unifies output from `SELECT`s into a single result set**
  - The name, order, and types of columns in each query must match
  - Hive only supports `UNION ALL`

```
SELECT emp_id, fname, lname, salary
      FROM employees
      WHERE state='CA' AND salary > 75000
UNION ALL
SELECT emp_id, fname, lname, salary
      FROM employees
      WHERE state != 'CA' AND salary > 50000;
```

- **`UNION ALL` can also be used with subqueries**

Subqueries in Hive

- **Hive only supports subqueries in the `FROM` clause of the `SELECT` statement**
  - It does not support correlated subqueries

```
SELECT prod_id, brand, name
FROM (SELECT *
        FROM products
        WHERE (price - cost) / price > 0.65
        ORDER BY price DESC
        LIMIT 10) high_profits
WHERE price > 1000
ORDER BY brand, name;
```

- **Hive allows arbitrary levels of subqueries**
  - Each subquery must be named (like `high_profits` above)

Chapter Topics

Relational Data Analysis with Hive

- Hive Databases and Tables
- Basic HiveQL Syntax
- **Data Types**
- Joining Datasets
- Common Built-in Functions
- Hands-On Exercise: Running Hive Queries from the Shell, Scripts, and Hue
- Conclusion

Hive's Data Types

- **Each column in Hive is associated with a data type**
- **Hive supports more than a dozen types**
  - Most are similar to ones found in relational databases
  - Hive also supports three complex types
- **Use the `DESCRIBE` command to see a table's column types**

```
hive> DESCRIBE products;
prod_id       int
brand         string
name          string
price         int
cost          int
shipping_wt   int
```

Hive's Integer Types

- **Integer types are appropriate for whole numbers**
  - Both positive and negative values allowed

Name	Description	Example Value
TINYINT	Range: -128 to 127	17
SMALLINT	Range: -32,768 to 32,767	5842
INT	Range: -2,147,483,648 to 2,147,483,647	84127213
BIGINT	Range: ~ -9.2 quintillion to ~ 9.2 quintillion	632197432180964

Hive's Decimal Types

- **Decimal types are appropriate for floating point numbers**
  - Both positive and negative values allowed
  - **Caution**: avoid using when exact values are required!

Name	Description	Example Value
`FLOAT`	Decimals	`3.14159`
`DOUBLE`	Very precise decimals	`3.141592653589793238 46`

Other Simple Types in Hive

- **Hive can also store several other types of information**
  - Only one character type (variable length)

Name	Description	Example Value
`STRING`	Character sequence	`Betty F. Smith`
`BOOLEAN`	True or False	`TRUE`
`TIMESTAMP`*	Instant in time	`2013-06-14 16:51:05`
`BINARY`*	Raw bytes	*N/A*

* Not available in older versions of Hive

Complex Column Types in Hive

- **Hive also has a few complex data types**
  - These are capable of holding multiple values

Column Type	Usage in Query
`ARRAY` Ordered list of values, all of the same type	`departments[0]`
`MAP` Key-value pairs, each of the same type	`employees['BF5314']`
`STRUCT` Named fields, of possibly mixed types	`address.street`

Chapter Topics

Relational Data Analysis with Hive

- Hive Databases and Tables
- Basic HiveQL Syntax
- Data Types
- **Joining Datasets**
- Common Built-in Functions
- Hands-On Exercise: Running Hive Queries from the Shell, Scripts, and Hue
- Conclusion

Joins in Hive

- **Joining disparate data sets is a common operation in Hive**
- **Hive supports several types of joins**
  - Inner joins
  - Outer joins (left, right, and full)
  - Cross joins (supported in Hive 0.10 and later)
  - Left semi joins
- **Only equality conditions are allowed in joins**
  - Valid: `customers.cust_id = orders.cust_id`
  - Invalid: `customers.cust_id <> orders.cust_id`
  - Outputs records where the specified key is found in each table
- **For best performance, list the largest table last in your query**

Join Syntax

- **Hive requires the following syntax for joins**

```
SELECT c.cust_id, name, total
FROM customers c
JOIN orders o ON (c.cust_id = o.cust_id);
```

- **The above example is an inner join**
  - Can replace `JOIN` with another type (e.g. `RIGHT OUTER JOIN`)
- **The following join syntax is not valid in Hive**

```
SELECT c.cust_id, name, total
FROM customers c, orders o
WHERE (c.cust_id = o.cust_id);
```

Inner Join Example

customers table

cust_id	name	country
a	Alice	us
b	Bob	ca
c	Carlos	mx
d	Dieter	de

orders table

order_id	cust_id	total
1	a	1539
2	c	1871
3	a	6352
4	b	1456
5	z	2137

```
SELECT c.cust_id, name, total
FROM customers c
JOIN orders o
ON (c.cust_id = o.cust_id);
```

Result of query

cust_id	name	total
a	Alice	1539
a	Alice	6352
b	Bob	1456
c	Carlos	1871

Left Outer Join Example

customers table

cust_id	name	country
a	Alice	us
b	Bob	ca
c	Carlos	mx
d	Dieter	de

orders table

order_id	cust_id	total
1	a	1539
2	c	1871
3	a	6352
4	b	1456
5	z	2137

```
SELECT c.cust_id, name, total
FROM customers c
LEFT OUTER JOIN orders o
ON (c.cust_id = o.cust_id);
```

Result of query

cust_id	name	total
a	Alice	1539
a	Alice	6352
b	Bob	1456
c	Carlos	1871
d	Dieter	NULL

Right Outer Join Example

customers table

cust_id	name	country
a	Alice	us
b	Bob	ca
c	Carlos	mx
d	Dieter	de

orders table

order_id	cust_id	total
1	a	1539
2	c	1871
3	a	6352
4	b	1456
5	z	2137

```
SELECT c.cust_id, name, total
FROM customers c
RIGHT OUTER JOIN orders o
ON (c.cust_id = o.cust_id);
```

Result of query

cust_id	name	total
a	Alice	1539
a	Alice	6352
b	Bob	1456
c	Carlos	1871
NULL	NULL	2137

Full Outer Join Example

customers table

cust_id	name	country
a	Alice	us
b	Bob	ca
c	Carlos	mx
d	Dieter	de

orders table

order_id	cust_id	total
1	a	1539
2	c	1871
3	a	6352
4	b	1456
5	z	2137

```
SELECT c.cust_id, name, total
FROM customers c
FULL OUTER JOIN orders o
ON (c.cust_id = o.cust_id);
```

Result of query

cust_id	name	total
a	Alice	1539
a	Alice	6352
b	Bob	1456
c	Carlos	1871
d	Dieter	NULL
NULL	NULL	2137

Using An Outer Join to Find Unmatched Entries

customers table

cust_id	name	country
a	Alice	us
b	Bob	ca
c	Carlos	mx
d	Dieter	de

orders table

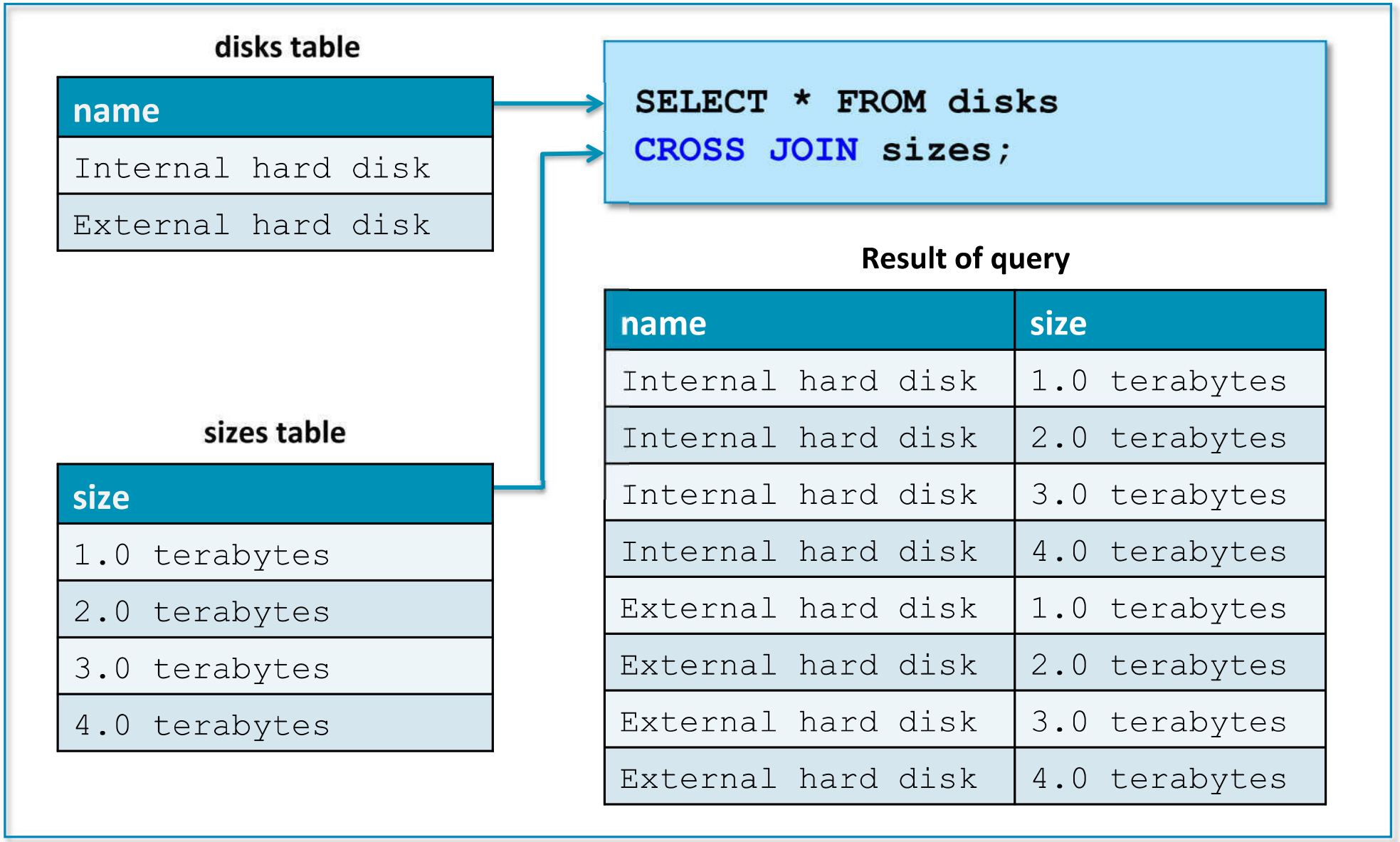
order_id	cust_id	total
1	a	1539
2	c	1871
3	a	6352
4	b	1456
5	z	2137

```
SELECT c.cust_id, name, total
FROM customers c
FULL OUTER JOIN orders o
ON (c.cust_id = o.cust_id)
WHERE c.cust_id IS NULL
OR o.total IS NULL;
```

Result of query

cust_id	name	total
d	Dieter	NULL
NULL	NULL	2137

Cross Join Example



Left Semi Joins (1)

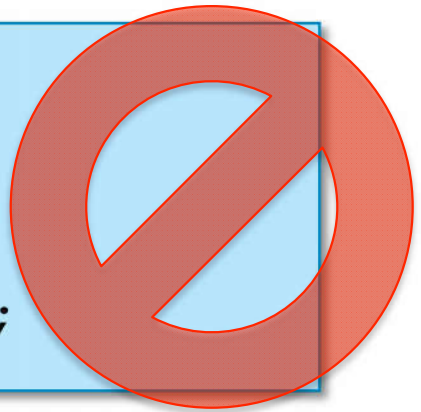
- **A less common type of join is the `LEFT SEMI JOIN`**
  - They are a special (and efficient) type of inner join
  - They behave more like a filter than a join
- **Left semi joins include additional criteria in the `ON` clause**
  - Only unique records that match these criteria are returned
  - Fields listed in `SELECT` are limited to the left-side table

```
SELECT c.cust_id
FROM customers c
LEFT SEMI JOIN orders o
ON (c.cust_id = o.cust_id
  AND YEAR(o.order_date) = '2012');
```

Left Semi Joins (2)

- **Hive does not support `IN/EXISTS` subqueries**

```
SELECT c.cust_id FROM customers c
WHERE o.cust_id IN
    (SELECT o.cust_id FROM orders o
     WHERE YEAR(o.order_date = '2012'));
```



- **Using a `LEFT SEMI JOIN` is a common workaround**

```
SELECT c.cust_id
FROM customers c
LEFT SEMI JOIN orders o
ON (c.cust_id = o.cust_id
  AND YEAR(o.order_date) = '2012');
```

Chapter Topics

Relational Data Analysis with Hive

- Hive Databases and Tables
- Basic HiveQL Syntax
- Data Types
- Joining Datasets
- **Common Built-in Functions**
- Hands-On Exercise: Running Hive Queries from the Shell, Scripts, and Hue
- Conclusion

Hive Functions

- **Hive offers dozens of built-in functions**
  - Many are identical to those found in SQL
  - Others are Hive-specific
- **Example function invocation**
  - Function names are not case-sensitive

```
hive> SELECT CONCAT(fname, ' ', lname) AS fullname
      FROM customers;
```

- **To see information about a function**

```
hive> DESCRIBE FUNCTION UPPER;
      UPPER(str) - Returns str with all characters
      changed to uppercase
```

Example Built-in Functions (1)

- **These functions operate on numeric values**

Function Description	Example Invocation	Input	Output
Rounds to specified # of decimals	`ROUND(total_price, 2)`	`23.492`	`23.49`
Returns nearest integer above	`CEIL(total_price)`	`23.492`	`24`
Returns nearest integer below	`FLOOR(total_price)`	`23.492`	`23`
Return absolute value	`ABS(temperature)`	`-49`	`49`
Returns square root	`SQRT(area)`	`64`	`8`
Returns a random number	`RAND()`		`0.584977`

Example Built-in Functions (2)

- **These functions operate on timestamp values**

Function Description	Example Invocation	Input	Output
Convert to UNIX format	`UNIX_TIMESTAMP(order_dt)`	`2013-06-14 16:51:05`	`1371243065`
Convert to string format	`FROM_UNIXTIME(mod_time)`	`1371243065`	`2013-06-14 16:51:05`
Extract date portion	`TO_DATE(order_dt)`	`2013-06-14 16:51:05`	`2013-06-14`
Extract year portion	`YEAR(order_dt)`	`2013-06-14 16:51:05`	`2013`
Returns # of days between dates	`DATEDIFF(order_dt, ship_dt)`	`2013-06-14, 2013-06-17`	`3`

Example Built-in Functions (3)

- **Here are some other interesting functions**

Function Description	Example Invocation	Input	Output
Converts to uppercase	`UPPER(fname)`	`Bob`	`BOB`
Extract portion of string	`SUBSTRING(name, 0, 2)`	`Alice`	`Al`
Selectively return value	`IF(price > 1000, 'A', 'B')`	`1500`	`A`
Convert to another type	`CAST(weight as INT)`	`3.581`	`3`
Returns size of array or map	`SIZE(array_field)`	**`N/A`**	`6`

Record Grouping and Aggregate Functions

- **`GROUP BY` groups selected data by one or more columns**
  - **Caution**: Columns not part of aggregation must be listed in `GROUP BY`

stores table

id	city	state	region
a	Albany	NY	EAST
b	Boston	MA	EAST
c	Chicago	IL	NORTH
d	Detroit	MI	NORTH
e	Elgin	IL	NORTH

```
SELECT region, state,
       COUNT(id) AS num
  FROM stores
  GROUP BY region, state;
```

Result of query

region	state	num
EAST	MA	1
EAST	NY	1
NORTH	IL	2
NORTH	MI	1

Built-in Aggregate Functions

- **Hive offers many aggregate functions, including**

Function Description	Example Invocation
Count all rows	`COUNT(*)`
Count all rows where field is not null	`COUNT(fname)`
Count all rows where field is unique and not null	`COUNT(DISTINCT fname)`
Returns the largest value	`MAX(salary)`
Returns the smallest value	`MIN(salary)`
Adds all supplied values and returns result	`SUM(price)`
Returns the average of all supplied values	`AVG(salary)`

Chapter Topics

Relational Data Analysis with Hive

- Hive Databases and Tables
- Basic HiveQL Syntax
- Data Types
- Joining Datasets
- Common Built-in Functions
- **Hands-On Exercise: Running Hive Queries from the Shell, Scripts, and Hue**
- Conclusion

Hands-on Exercise: Running Hive Queries

- **In this Hands-On Exercise, you will run Hive queries from the Hive shell, the command line, Hive scripts, and Hue.**
- **Please refer to the Hands-On Exercise Manual for instructions**

Chapter Topics

Relational Data Analysis with Hive

- Hive Databases and Tables
- Basic HiveQL Syntax
- Data Types
- Joining Datasets
- Common Built-in Functions
- Hands-On Exercise: Running Hive Queries from the Shell, Scripts, and Hue
- **Conclusion**

Essential Points

- **Every Hive table belongs to exactly one database**
  - The `SHOW DATABASES` command lists databases
  - The `USE` command switches the active database
  - The `SHOW TABLES` command lists all tables in a database
- **Every column in a Hive table has an associated data type**
  - Most simple column types are similar to SQL
  - Hive also supports a few complex types
- **HiveQL syntax is familiar to those who know SQL**
  - A subset of SQL-92, plus Hive-specific extensions
  - Supports inner, outer, and left semi joins
  - Many SQL functions are built into Hive

Bibliography

The following offer more information on topics discussed in this chapter

- **HiveQL Language Manual**
  - `http://tiny.cloudera.com/dac10a`
- **Hive Built-In Functions**
  - `http://tiny.cloudera.com/dac10b`