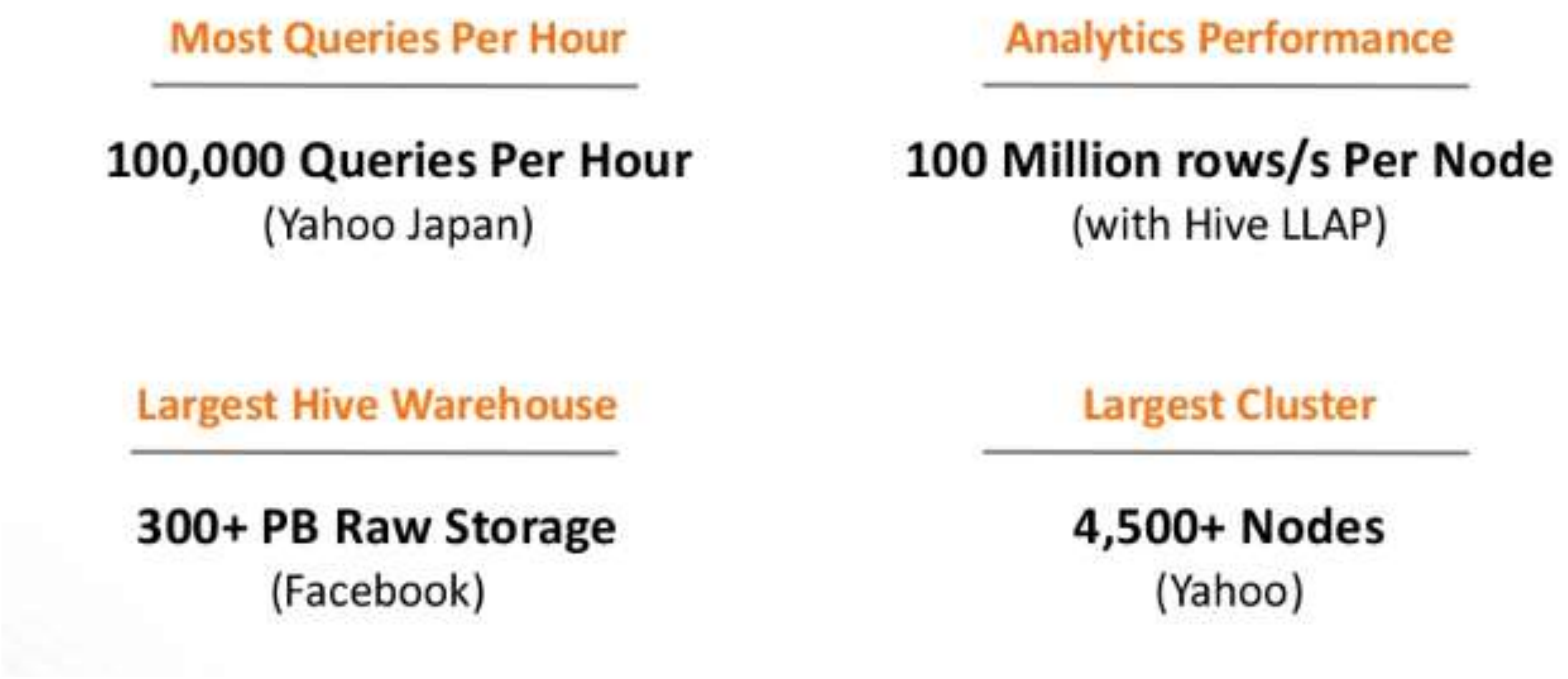
What Is Hive

- Apache **Hive** is a data warehouse (initially developed by Facebook) software project built on top of Apache Hadoop for providing data summarization, query and analysis.

- **Hive** gives an SQL-like interface to query data stored in various databases and file systems that integrate with Hadoop. It provides a SQL interface to query data stored in Hadoop distributed file system (HDFS) or Amazon S3 (an AWS implementation) through an HDFS-like abstraction layer called EMRFS (Elastic MapReduce File System).

Apache Hive: Fast Facts



Most Queries Per Hour

100,000 Queries Per Hour
(Yahoo Japan)

Analytics Performance

100 Million rows/s Per Node
(with Hive LLAP)

Largest Hive Warehouse

300+ PB Raw Storage
(Facebook)

Largest Cluster

4,500+ Nodes
(Yahoo)

Apache Hive: Journey to SQL:2011 Analytics

Data Types
Numeric
FLOAT, DOUBLE
DECIMAL
INT, TINYINT, SMALLINT, BIGINT
BOOLEAN
String
CHAR, VARCHAR
BLOB (BINARY), CLOB (String)
Date, Time
DATE, TIMESTAMP, Interval Types
Complex Types
ARRAY / MAP / STRUCT / UNION
Nested Data Analytics
Nested Data Traversal
Lateral Views
Procedural Extensions
HPL/SQL

SQL Features
Core SQL Features
Date, Time and Arithmetical Functions
INNER, OUTER, CROSS and SEMI Joins
Derived Table Subqueries
Correlated + Uncorrelated Subqueries
UNION ALL
UDFs, UDAFs, UDTFs
Common Table Expressions
UNION DISTINCT
Advanced Analytics
OLAP and Windowing Functions
OLAP: Partition, Order by UDAF
CUBE and Grouping Sets
ACID Transactions
INSERT / UPDATE / DELETE
Constraints
Primary / Foreign Key (Non Validated)

File Formats
Columnar
ORCFile
Parquet
Text
CSV
Logfile
Nested / Complex
Avro
JSON
XML
Custom Formats
Other Features
XPath Analytics

Hive 2
ACID MERGE
Multi Subquery
Scalar Subqueries
Non-Equijoins
INTERSECT / EXCEPT

Recursive CTEs
NOT NULL Constraints
Default Values
Multi-statement Transactions

Legend

🟧	New
🟨	Hive 2
🟦	Future work

Track Hive **SQL:2011 Complete:** HIVE-13554

HORTONWORKS

Hive Is Not...

- **A Relational Database**
  - Hive uses a database to store metadata, but the data that Hive processes is stored in HDFS
- **Designed for on-Line Transaction Processing**
  - Hive runs on Hadoop (a batch-processing system where jobs can have high latency with substantial overhead)
  - Therefore, latency for Hive queries is generally high (even for small jobs)
- **Suited for real-time queries and row-level updates**
  - Hive is best used for batch jobs over large sets of immutable data (such as web logs)

Limitations

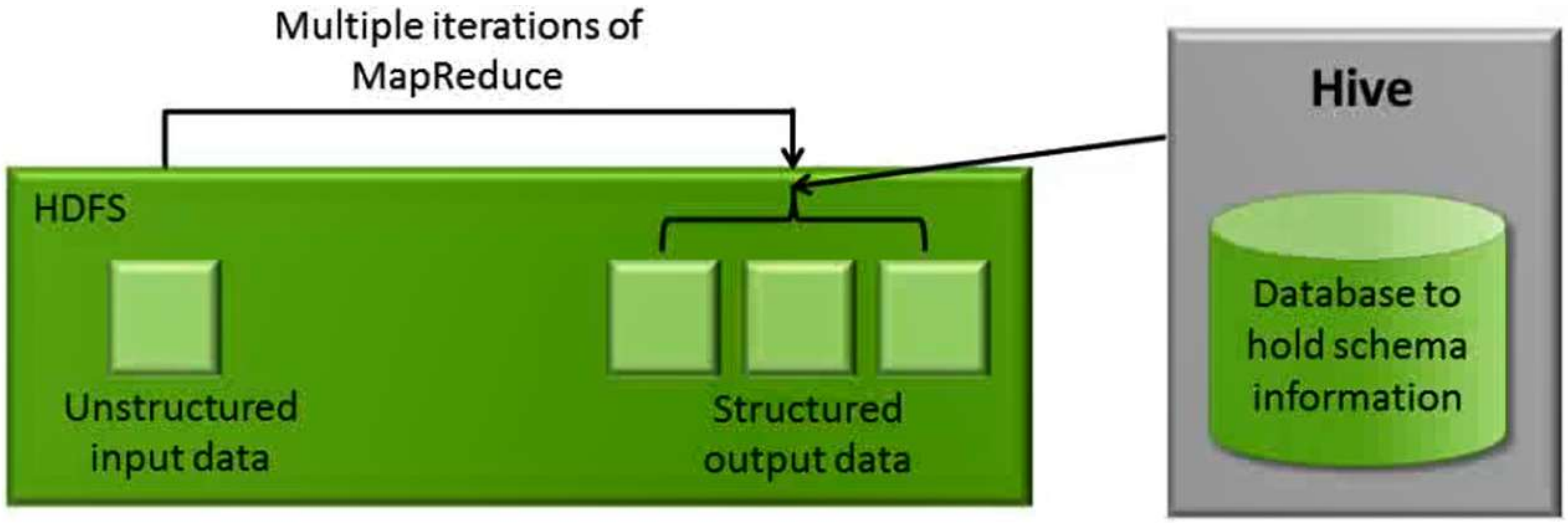
- Certain standard SQL functions, such as NOT IN, NOT LIKE, and NOT EQUAL, do not exist or require certain workarounds.

- Hive is not made for low-latency, real-time, or near-real-time querying.

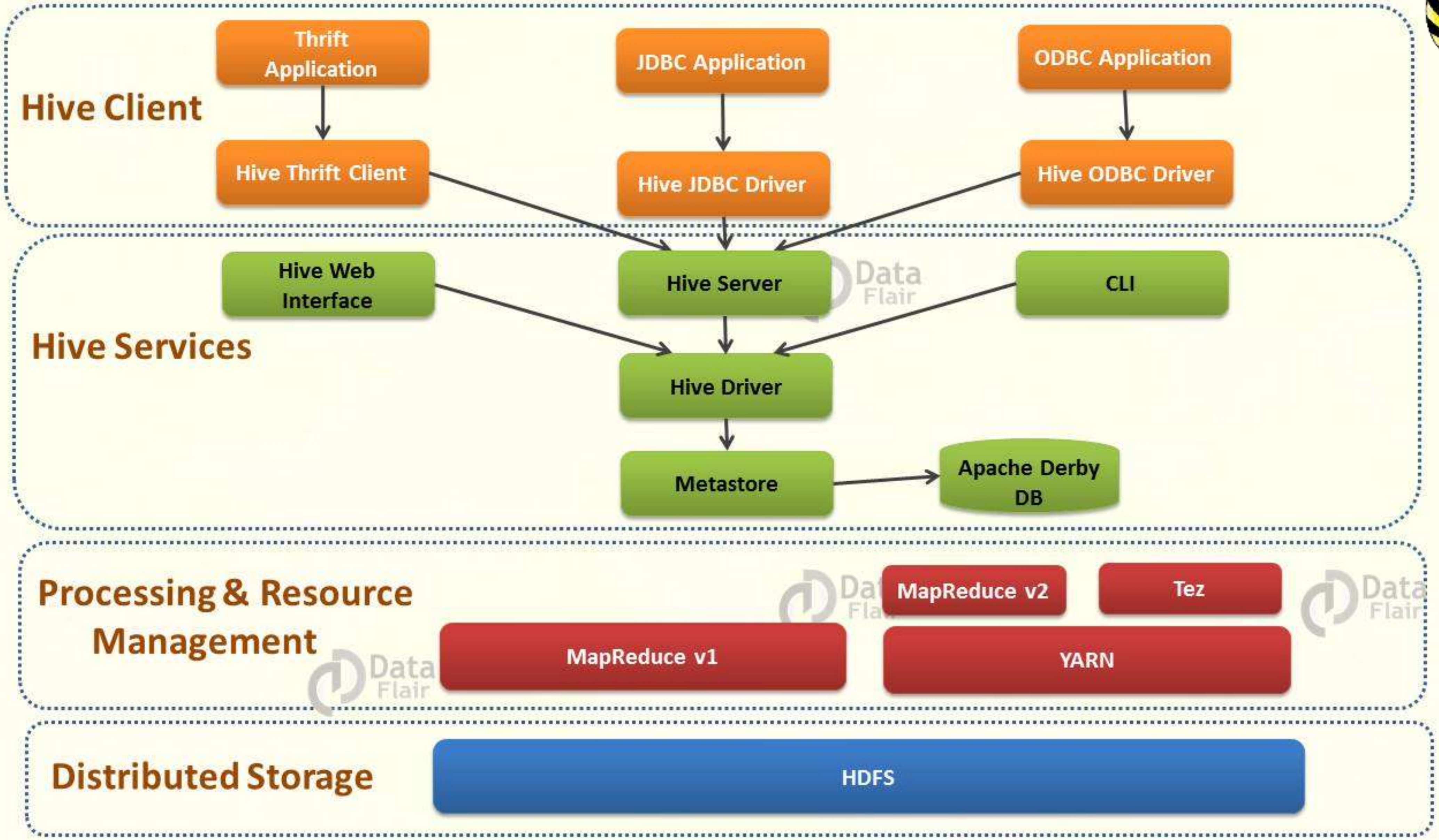
  SQL queries translate to MapReduce, which means slower performance for certain queries compared to traditional RDBMS.

Typical Use Case

- Supports uses shuch as: Ad-hoc queries, Summarization, Data Analysis


Multiple iterations of MapReduce
HDFS
Unstructured input data
Structured output data
Hive
Database to hold schema information

Hive Architecture & its Components

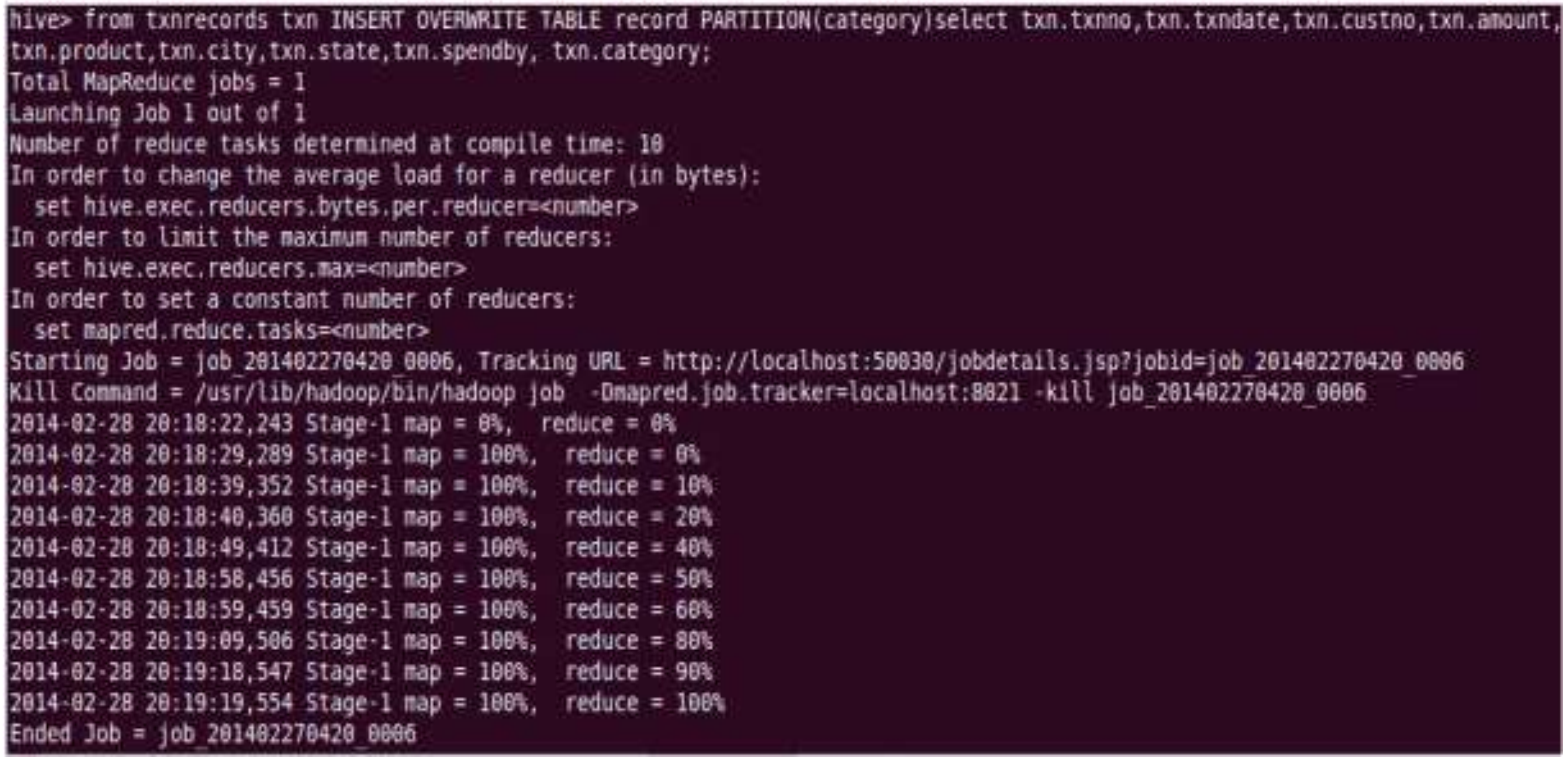


Hadoop Hive Components

- **Hive Clients –** Apache Hive supports all application written in languages like C++, Java, Python etc. using JDBC, Thrift and ODBC drivers. Thus, one can easily write Hive client application written in a language of their choice.
- **Hive Services –** Hive provides various services like web Interface, CLI etc. to perform queries.
- **Processing framework and Resource Management –** Hive internally uses Hadoop MapReduce framework to execute the queries.
- **Distributed Storage –** As seen above that Hive is built on the top of Hadoop, so it uses the underlying HDFS for the distributed storage.

Hive Services

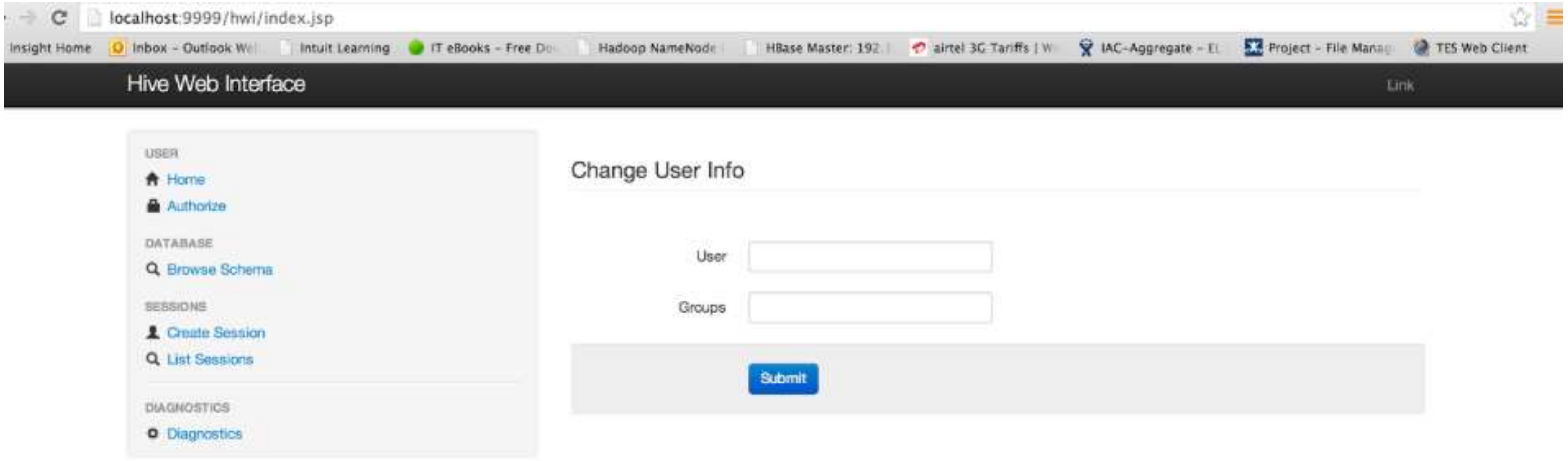
- **a) CLI(Command Line Interface) –** This is the default shell that Hive provides, in which you can execute your Hive queries and command directly.



```
hive> from txnrecords txn INSERT OVERWRITE TABLE record PARTITION(category)select txn.txnno,txn.txndate,txn.custno,txn.amount,
txn.product,txn.city,txn.state,txn.spendby, txn.category;
Total MapReduce jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 10
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapred.reduce.tasks=<number>
Starting Job = job_201402270420_0006, Tracking URL = http://localhost:50030/jobdetails.jsp?jobid=job_201402270420_0006
Kill Command = /usr/lib/hadoop/bin/hadoop job  -Dmapred.job.tracker=localhost:8021 -kill job_201402270420_0006
2014-02-28 20:18:22,243 Stage-1 map = 0%,  reduce = 0%
2014-02-28 20:18:29,289 Stage-1 map = 100%,  reduce = 0%
2014-02-28 20:18:39,352 Stage-1 map = 100%,  reduce = 10%
2014-02-28 20:18:40,360 Stage-1 map = 100%,  reduce = 20%
2014-02-28 20:18:49,412 Stage-1 map = 100%,  reduce = 40%
2014-02-28 20:18:58,456 Stage-1 map = 100%,  reduce = 50%
2014-02-28 20:18:59,459 Stage-1 map = 100%,  reduce = 60%
2014-02-28 20:19:09,506 Stage-1 map = 100%,  reduce = 80%
2014-02-28 20:19:18,547 Stage-1 map = 100%,  reduce = 90%
2014-02-28 20:19:19,554 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_201402270420_0006
```

Hive Services

- **b) Web Interface –** Hive also provides web based GUI for executing Hive queries and commands.

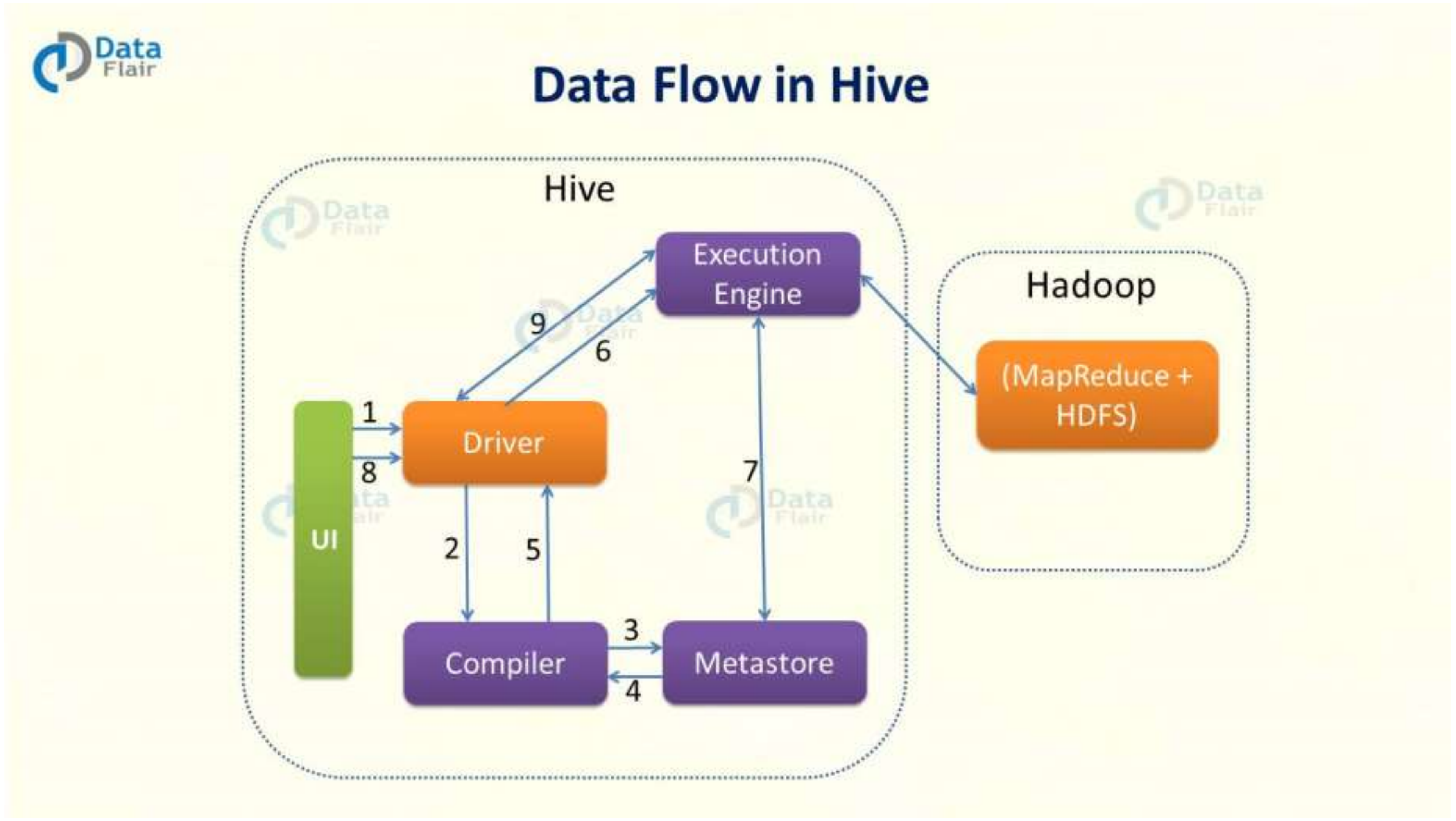


Pig vs Hive

Pig and **Hive** work well together and many businesses use both

Pig	Hive
Used by Programmers and Researchers	Used by Analysts
Used for Programming	Used for Reporting
Procedural data-flow language	Declarative SQLish language
Works on the Client side of a Cluster	Works on the Server side of a Cluster
For Semi-Structured Data	For Structured Data

How to process data with Apache Hive?



How to process data with Apache Hive?

- User Interface (UI) calls the execute interface to the Driver.
- The driver creates a session handle for the query. Then it sends the query to the compiler to generate an execution plan.
- The compiler needs the metadata. So it sends a request for *getMetaData*. Thus receives the *sendMetaData* request from Metastore.
- Now compiler uses this metadata to type check the expressions in the query. The compiler generates the plan which is **DAG** of stages with each stage being either a **map/reduce job**, a metadata operation or an operation on **HDFS**. The plan contains map operator trees and a reduce operator tree for map/reduce stages.

How to process data with Apache Hive?

- Now execution engine submits these stages to appropriate components. After in each task the deserializer associated with the table or intermediate outputs is used to read the rows from HDFS files. Then pass them through the associated operator tree. Once it generates the output, write it to a temporary HDFS file through the serializer. Now temporary file provides the subsequent map/reduce stages of the plan. Then move the final temporary file to the table’s location for DML operations.
- Now for queries, execution engine directly read the contents of the temporary file from HDFS as part of the fetch call from the Driver.

```
INFO  : Compiling command(queryId=hive_20180612053333_da2d1552-808e-4959-b365-197e69a9d841): SELECT *
FROM customers
INFO  : Semantic Analysis Completed
INFO  : Returning Hive schema: Schema(fieldSchemas:[FieldSchema(name:customers.id, type:int, comment:null), FieldSchema
(name:customers.name, type:string, comment:null), FieldSchema(name:customers.email_preferences,
type:struct<email_format:string,frequency:string,categories:struct<promos:boolean,surveys:boolean>>, comment:null), FieldSchema
(name:customers.addresses, type:map<string,struct<street_1:string,street_2:string,city:string,state:string,zip_code:string>>,
comment:null), FieldSchema(name:customers.orders,
type:array<struct<order_id:string,order_date:string,items:array<struct<product_id:int,sku:string,name:string,price:double,qty:int>>>>
, comment:null)], properties:null)
INFO  : Completed compiling command(queryId=hive_20180612053333_da2d1552-808e-4959-b365-197e69a9d841); Time taken: 0.031 seconds
INFO  : Executing command(queryId=hive_20180612053333_da2d1552-808e-4959-b365-197e69a9d841): SELECT *
FROM customers
INFO  : Completed executing command(queryId=hive_20180612053333_da2d1552-808e-4959-b365-197e69a9d841); Time taken: 0.001 seconds
INFO  : OK
```

HiveQL

HiveQL Features

- HiveQL is similar to other SQLs

-Use familiar relational database concepts (tables, rows, Schema…)

- Support multi-table inserts via your code

-accesses Big Data via table

- Converts SQL queries into MapReduce jobs

-user doesn’t need to know MapReduce

Getting Started With Hive

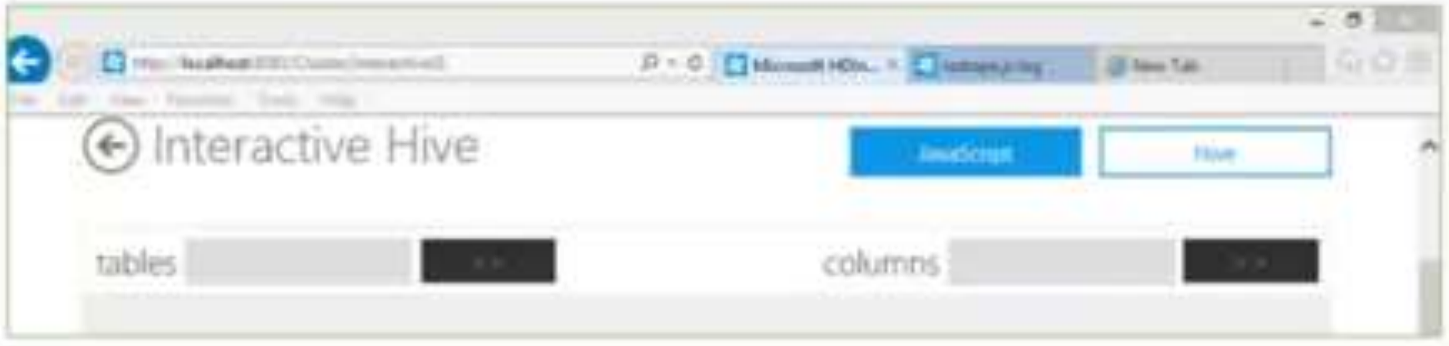
- The Hive shell is started using the hive executable:

```
$ hive
hive>
```

- Use the -f flag to specify a file that contains a Hive script:

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

- Microsoft HDInsight provides a Hive Console



Hive Table

Hive Table

- A Hive Table:

-Data: file or group of files in HDFS

-Schema: in the form of metadata stored in a relational database

Schema and Data are separate

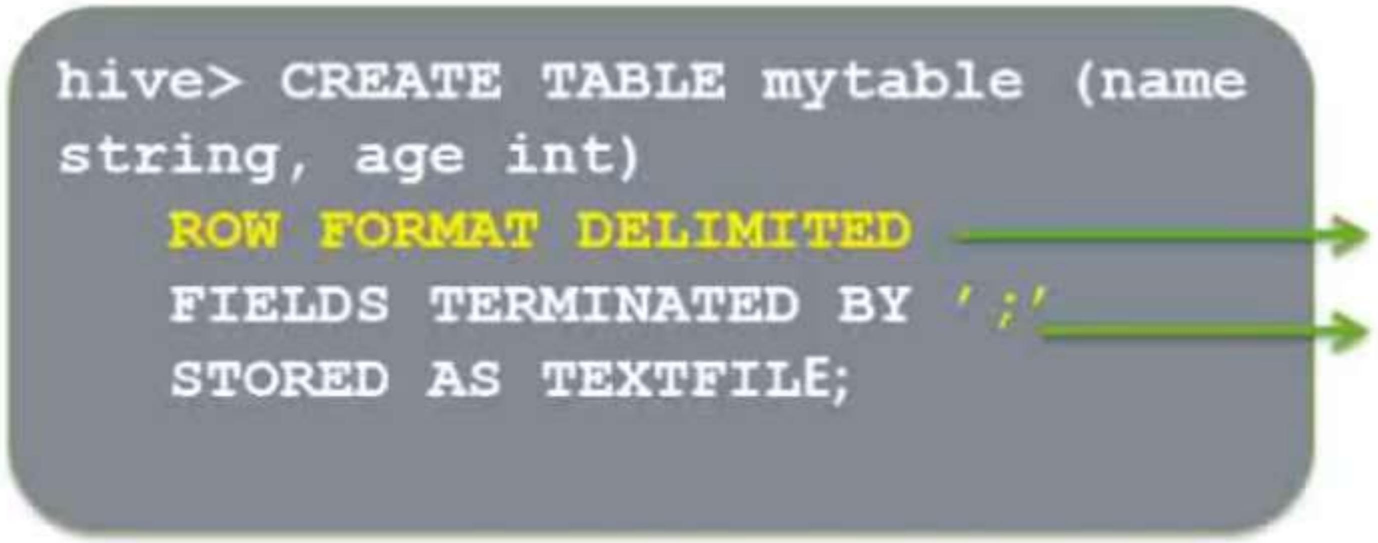
-Schema can be defined for existing data

-data can added or removed independently

-Hive can be pointed at existing data

You have to define a schema if you have existing data in HDFS that you want to use in Hive

Defining a Table

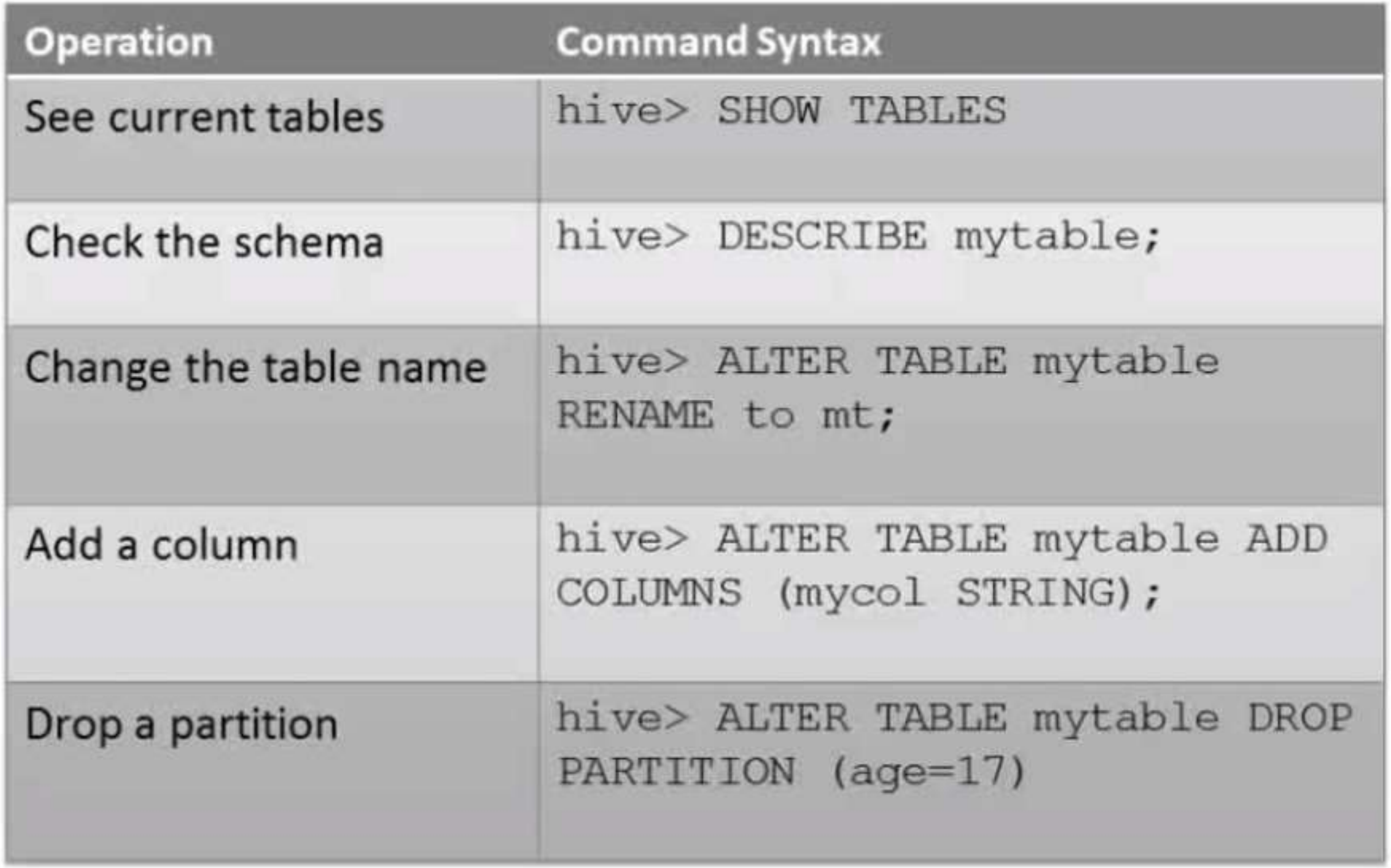


```
hive> CREATE TABLE mytable (name
string, age int)
    ROW FORMAT DELIMITED
    FIELDS TERMINATED BY ';'
    STORED AS TEXTFILE;
```

Each row is comma delimited text

HiveQL statements are terminated with a semicolon

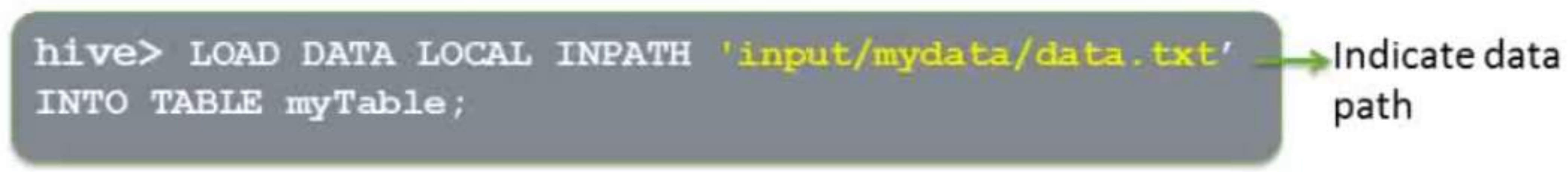
Managing Table



Operation	Command Syntax
See current tables	`hive> SHOW TABLES`
Check the schema	`hive> DESCRIBE mytable;`
Change the table name	`hive> ALTER TABLE mytable RENAME to mt;`
Add a column	`hive> ALTER TABLE mytable ADD COLUMNS (mycol STRING);`
Drop a partition	`hive> ALTER TABLE mytable DROP PARTITION (age=17)`

Loading Data

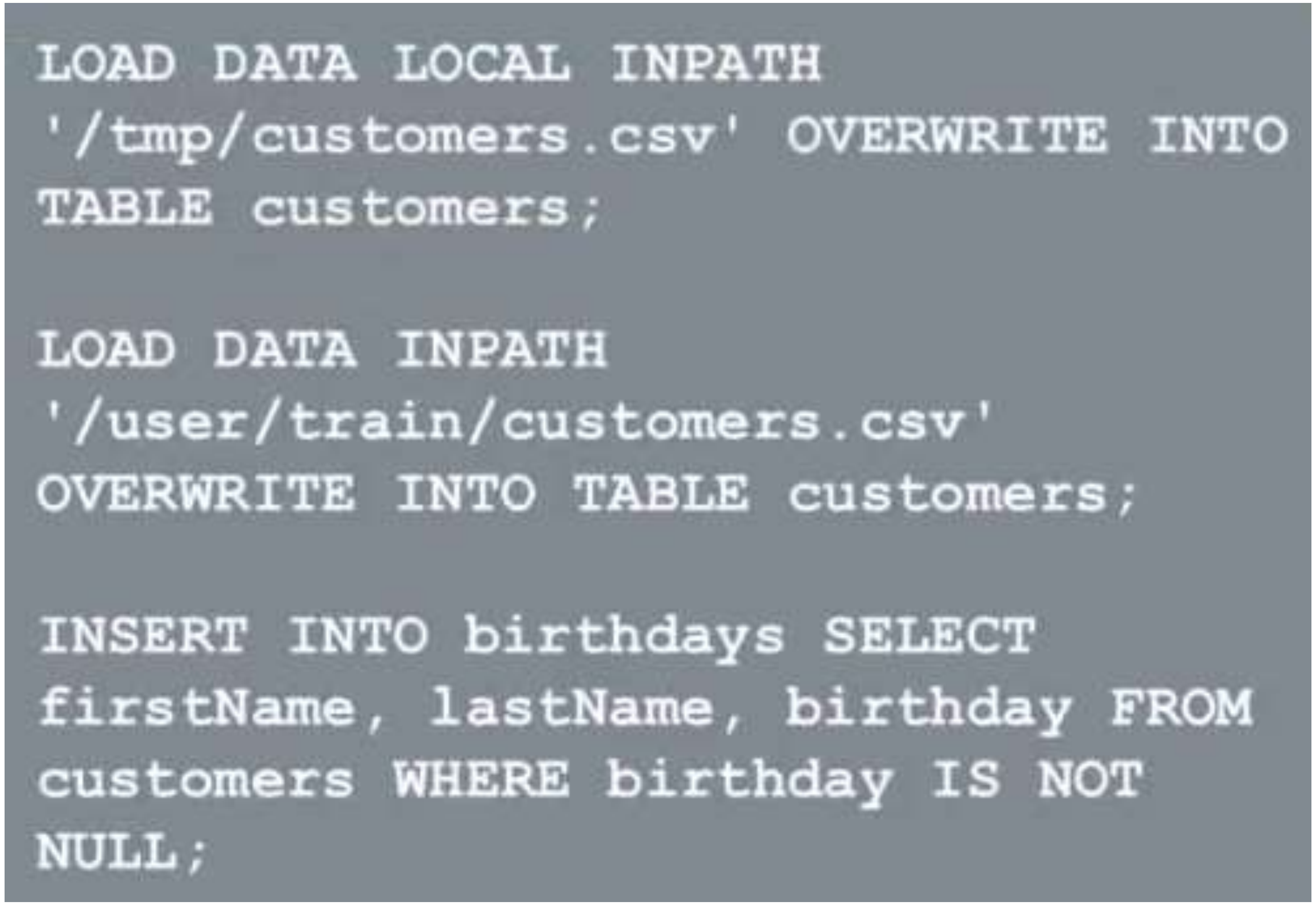
- Use LOAD DATA to import data into Hive Table



```
hive> LOAD DATA LOCAL INPATH 'input/mydata/data.txt'
INTO TABLE myTable;
```

- Not modified by Hive – Loaded as is
- Use the word OVERWRITE to write over a file of the same name
- The schema is checked when the data is queried
  - If a node does not match, it will be read as a null

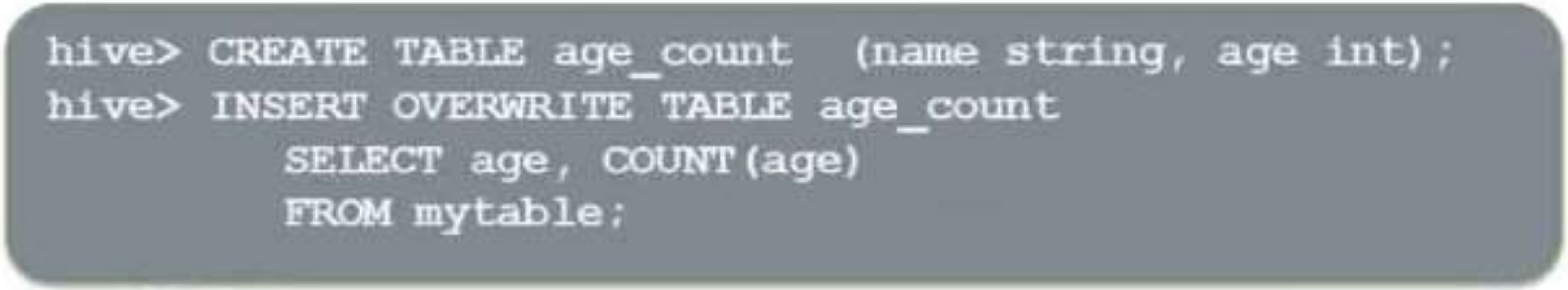
Loading Data



```
LOAD DATA LOCAL INPATH
'/tmp/customers.csv' OVERWRITE INTO
TABLE customers;

LOAD DATA INPATH
'/user/train/customers.csv'
OVERWRITE INTO TABLE customers;

INSERT INTO birthdays SELECT
firstName, lastName, birthday FROM
customers WHERE birthday IS NOT
NULL;
```

Insert

- Use INSERT statement to populate data into a table from another Hive table.
- Since query results are usually large it is best to use an INSERT clause to tell Hive where to store your query



```
hive> CREATE TABLE age_count  (name string, age int);
hive> INSERT OVERWRITE TABLE age_count
        SELECT age, COUNT(age)
        FROM mytable;
```

Insert Overwrite

- Overwrite is used to replace the data in the table, Otherwise the data is appended to the table
  - Append Happens by adding files to the directory holding the table data

```
INSERT OVERWRITE newtable SELECT * FROM mytable;
```

Can Write to directory in HDFS

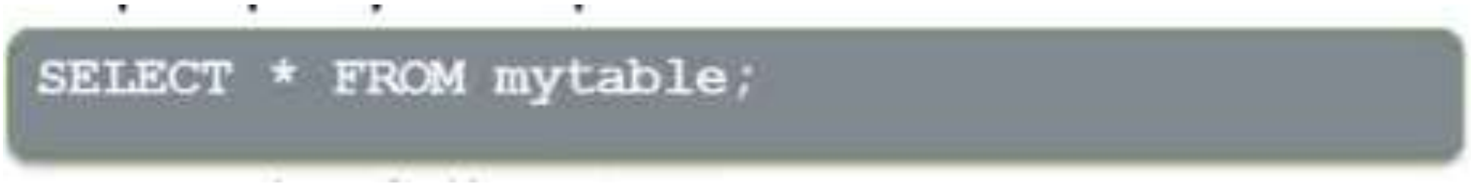
```
INSERT OVERWRITE DIRECTORY ‘/hdfs/myresult’ SELECT
* FROM mytable;
```

Can Write to Local directory

```
INSERT OVERWRITE LOCAL DIRECTORY...
```

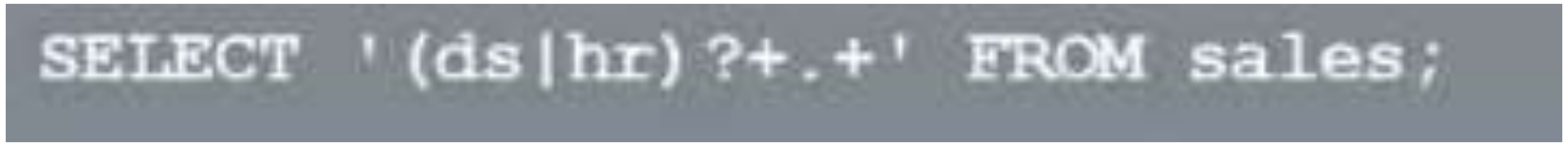
Performing Queries (HiveQL)

- SELECT



```
SELECT * FROM mytable;
```

- Supports the following:
  - WHERE clause
  - UNION ALL and DISTINCT
  - GROUP BY and HAVING
  - LIMIT clause
  - Can use REGEX column Specification



```
SELECT '(ds|hr)?+.+' FROM sales;
```

- `SELECT * FROM customers;`
- `SELECT COUNT(1) FROM customers;`

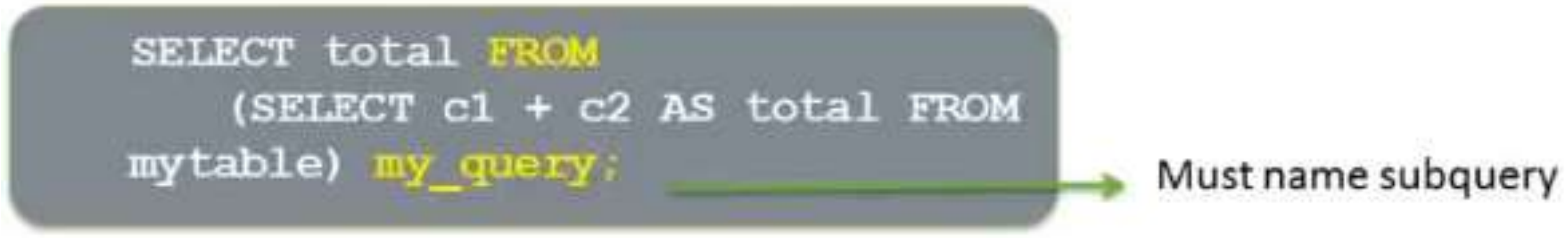
-
  ```
  SELECT firstName, lastName, address,
  zip FROM customers WHERE orderID > 0
  GROUP BY zip;
  ```

-
  ```
  SELECT customers.*, orders.* FROM
  customers JOIN orders ON
  (customers.customerID =
  orders.customerID);
  ```

-
  ```
  SELECT customers.*, orders.* FROM
  customers LEFT OUTER JOIN orders ON
  (customers.customerID =
  orders.customerID);
  ```

Subqueries

- Hive support subqueries only in the FROM clause

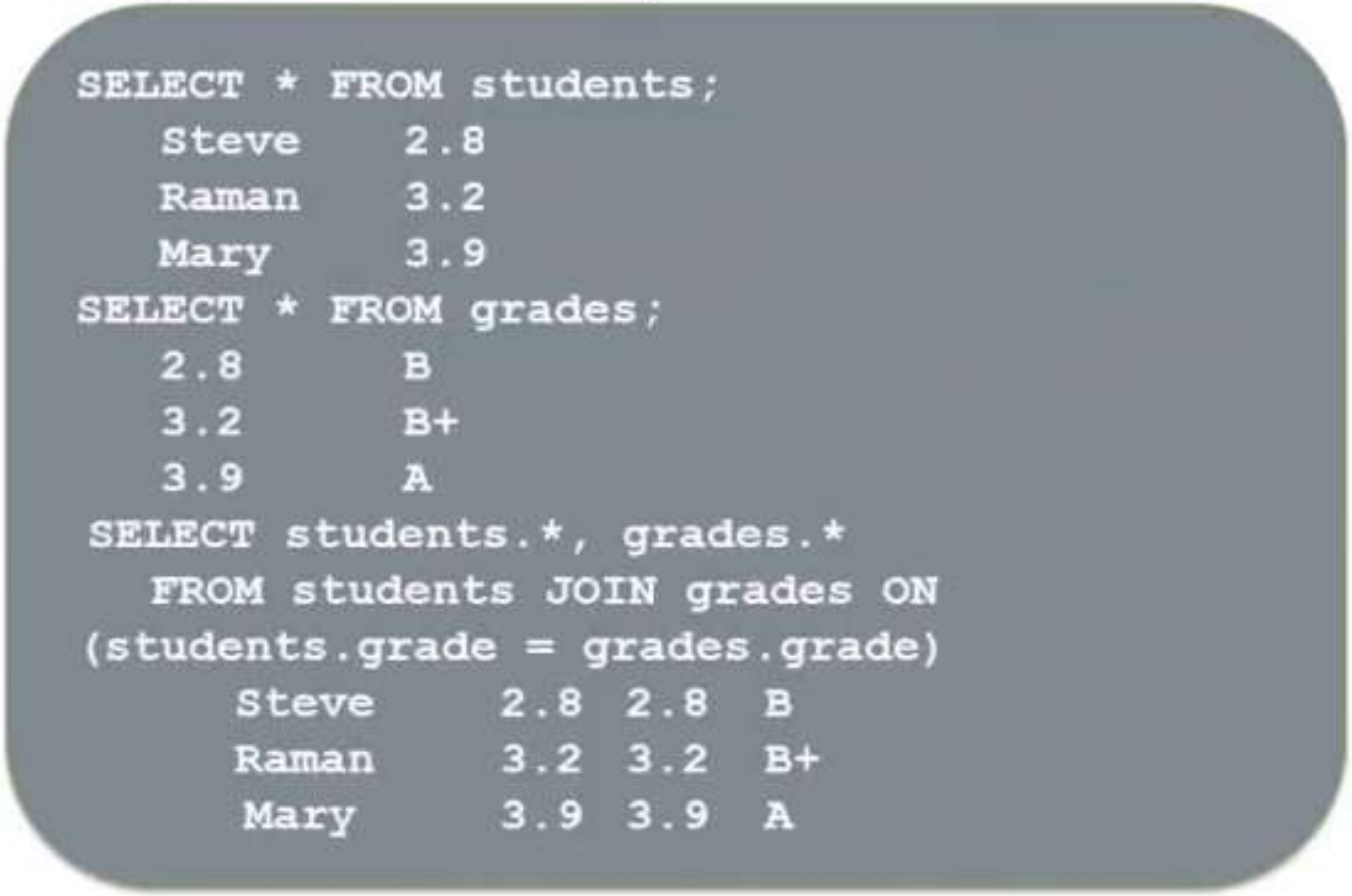


```
SELECT total FROM
    (SELECT c1 + c2 AS total FROM
mytable) my_query;
```

- The columns in the subquery SELECT list are available in outer query

JOIN – Inner Joins

Inner joins – Intersection of the two tables (returns rows they have in common)



```
SELECT * FROM students;
   Steve    2.8
   Raman    3.2
   Mary     3.9
SELECT * FROM grades;
   2.8      B
   3.2      B+
   3.9      A
SELECT students.*, grades.*
  FROM students JOIN grades ON
(students.grade = grades.grade)
     Steve     2.8  2.8  B
     Raman     3.2  3.2  B+
     Mary      3.9  3.9  A
```

JOIN – Outer Joins

- Allows for the finding of rows with non-matches in the tables being joined
- Three Types:
  - LEFT OUTER JOIN
    - Returns a row for every row in the first table entered
  - RIGHT OUTER JOIN
    - Returns a row for every row in the second table entered
  - FULL OUTER JOIN
    - Returns a row for every from both tables

Sorting

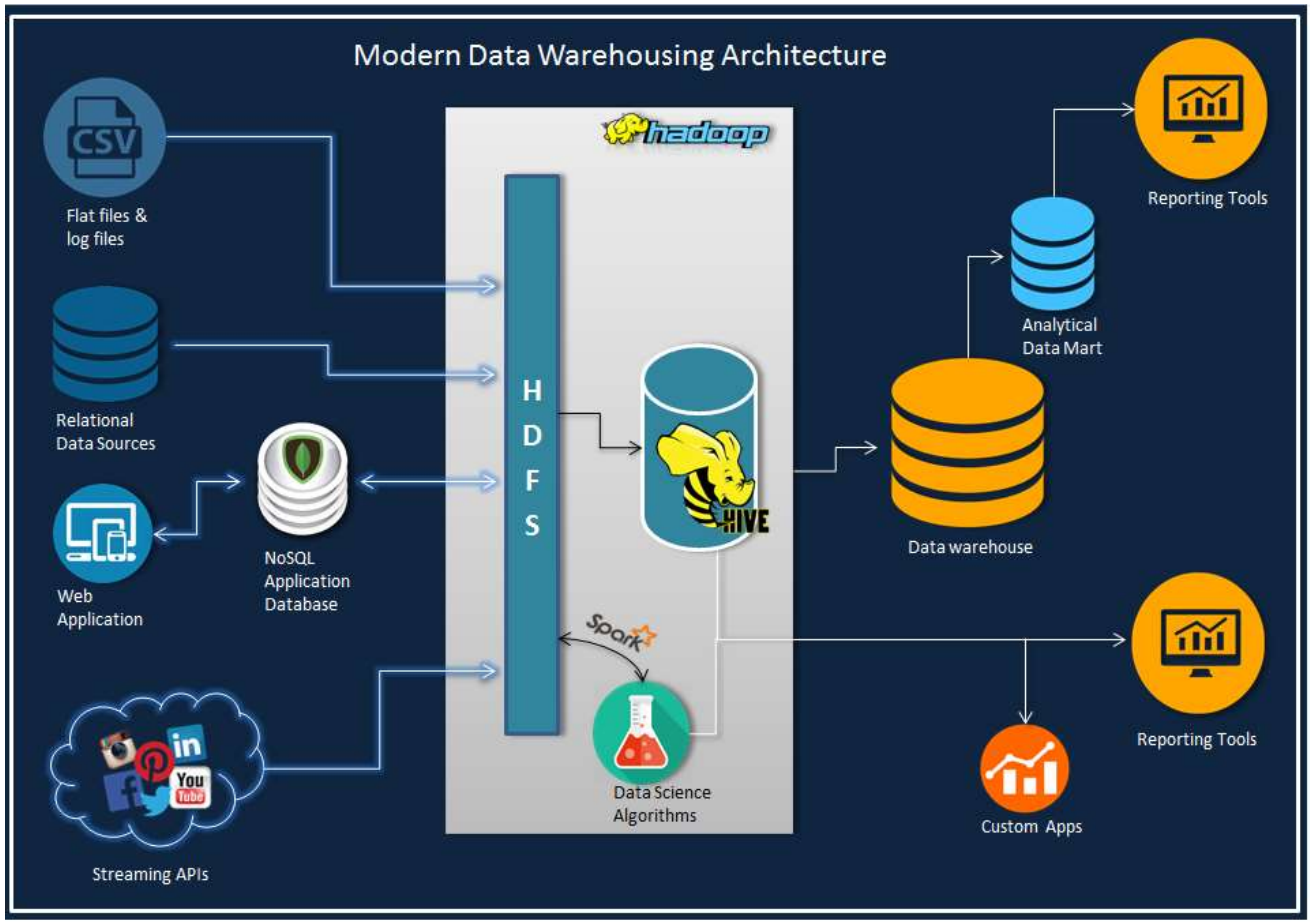
- ORDER BY
  - Sort but sets the number of reducers to 1
- SORT BY
  - Multiple reducers with a sorted file from each

Summary

- Hive works with Hadoop to allow you to query and manage large-scale data using a familiar SQL-like interface
- Hive provides CLI access to the shell and Microsoft HDInsight provides console access
- Hive tables consist of data and schema; data and schema are separated for maximum flexibility
- Hive query language supports familiar SQL operations including joins, subqueries, order by, sort by, and so on

Build a data warehouse with Hive

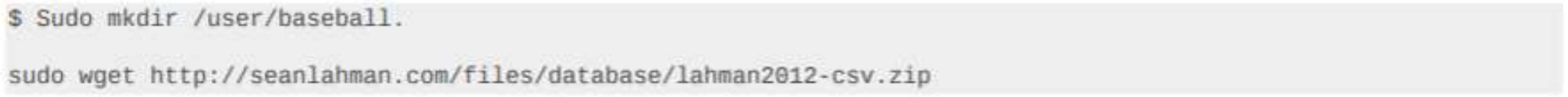
- If you want to build a data library using Hadoop, but have no Java or MapReduce knowledge, Hive can be a great alternative (if you know SQL).
    - It also allows you to build star schemas on top of HDFS.



Example: Build a data warehouse for baseball information

- Begin by downloading the CSV file that contains statistics about baseball and baseball players. From within Linux, create a directory, then run:

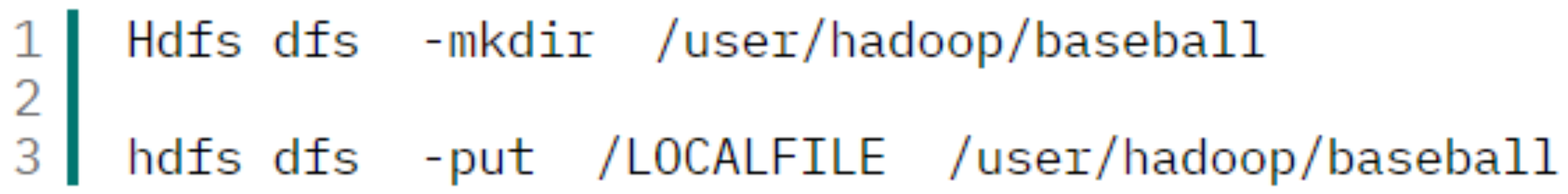
- The example contains four main tables, each with a unique column



```
$ Sudo mkdir /user/baseball.

sudo wget http://seanlahman.com/files/database/lahman2012-csv.zip
```

Load data into HDFS or Hive

- Different theories and practices are used to load data into Hadoop. Sometimes, you ingest raw files directly into HDFS. You might create certain directories and subdirectories to organize the files, but it's a simple process of copying or moving files from one location to another.



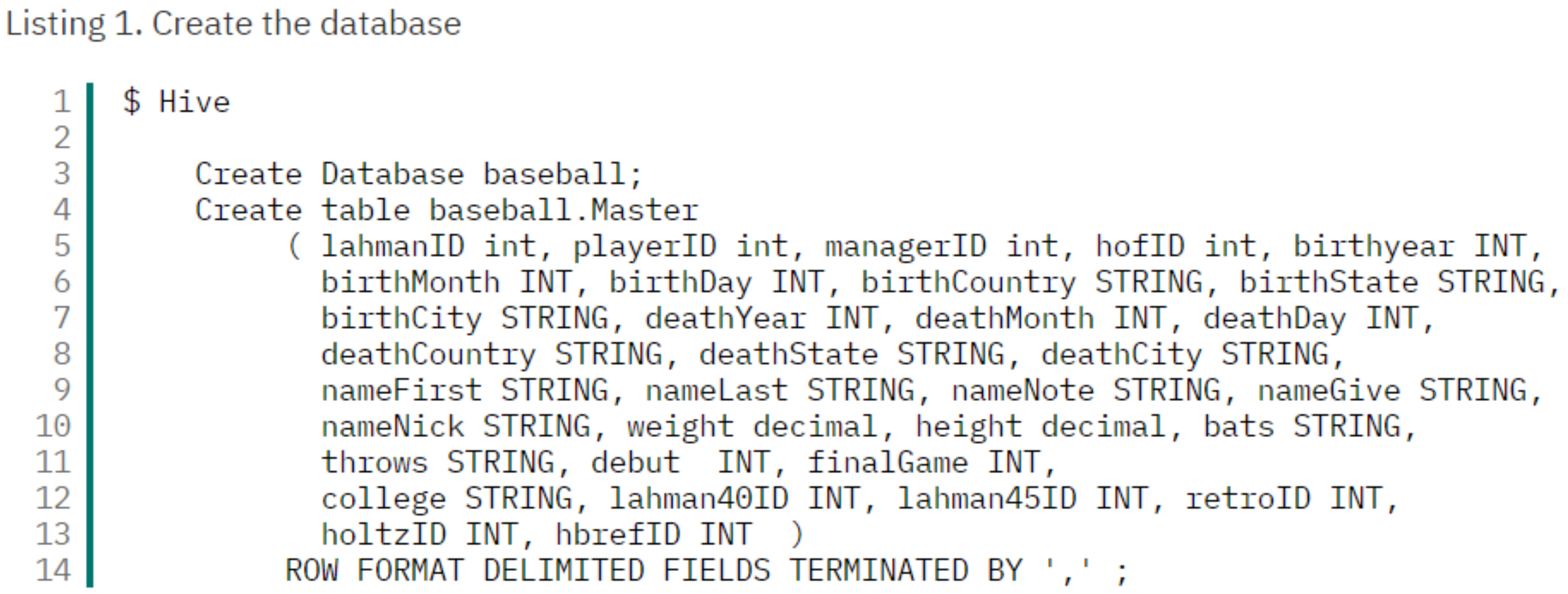
```
Hdfs dfs  -mkdir  /user/hadoop/baseball

hdfs dfs  -put  /LOCALFILE  /user/hadoop/baseball
```

Build the data library with Hive

- To keep it simple, let's use the Hive shell. The high-level steps are:

1. Create the baseball database
2. Create the tables
3. Load the tables
4. Verify that the tables are correct

Build the data library with Hive



Listing 1. Create the database

```
$ Hive

    Create Database baseball;
    Create table baseball.Master
         ( lahmanID int, playerID int, managerID int, hofID int, birthyear INT,
           birthMonth INT, birthDay INT, birthCountry STRING, birthState STRING,
           birthCity STRING, deathYear INT, deathMonth INT, deathDay INT,
           deathCountry STRING, deathState STRING, deathCity STRING,
           nameFirst STRING, nameLast STRING, nameNote STRING, nameGive STRING,
           nameNick STRING, weight decimal, height decimal, bats STRING,
           throws STRING, debut  INT, finalGame INT,
           college STRING, lahman40ID INT, lahman45ID INT, retroID INT,
           holtzID INT, hbrefID INT  )
         ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' ;
```

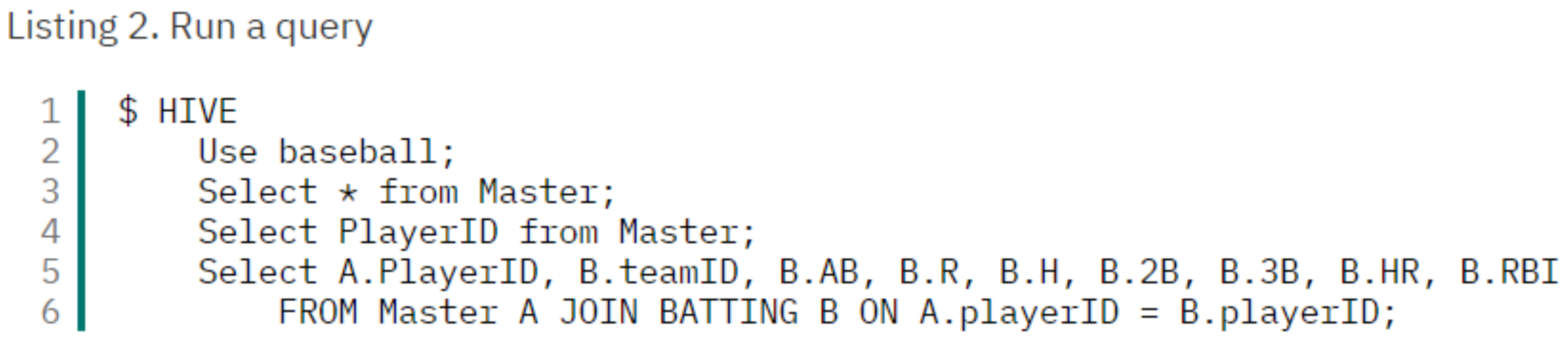
Build the data library with Hive

- To load data into the Hive table, open the Hive shell again, then run the following code:

```
$hive
    LOAD DATA LOCAL INPATH Master.csv OVERWRITE INTO TABLE baseball.Master;
```

Build a normalized database with Hive

- The baseball database is more or less normalized: You have the four main tables and several secondary tables. Again, Hive is a Schema on Read, so you have to do most of the work in the data analysis and ETL stages because there is no indexing or referential integrity such as in traditional RDBMSes.



Listing 2. Run a query

```
$ HIVE
    Use baseball;
    Select * from Master;
    Select PlayerID from Master;
    Select A.PlayerID, B.teamID, B.AB, B.R, B.H, B.2B, B.3B, B.HR, B.RBI
        FROM Master A JOIN BATTING B ON A.playerID = B.playerID;
```