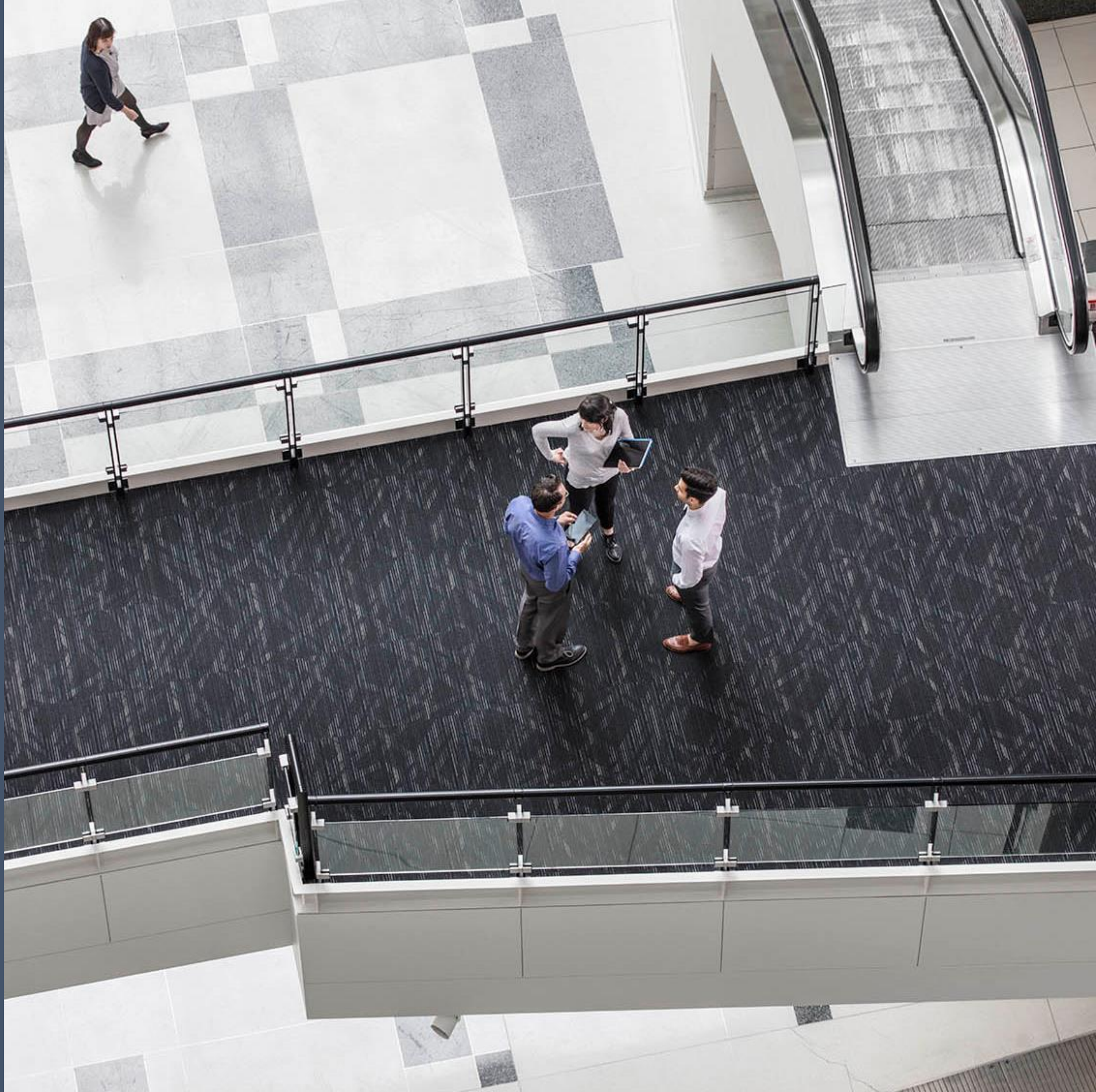
Microsoft

HIveQL

November 2020

Brolinskyi Sergii



Plan of presentation

- Definition Hive
- HiveQL
- Hive vs classic MapReduce
- Pig Latin vs Hive
- Summary

Transactional and Analytical Processing

Transactional Processing	Analytical Processing
Analyzes individual entries	Analyzes large batches of data
Access to recent data, from the last few hours or days	Access to older data going back months, or even years
Updates data	Only reads data
Fast real-time access	Long running jobs
Usually a single data source	Multiple data sources

Transactional and Analytical Processing

Small Data

Both these objectives could be achieved using the same database system

Transactional and Analytical Processing

Transactional Processing

Analytical Processing

Traditional RDBMS

Data Warehouse

Data Warehouse

System used for reporting and data analysis and is considered a core component of business intelligence. DWs are central repositories of integrated data from one or more disparate sources. They store current and historical data in one single place that are used for creating analytical reports for workers throughout the enterprise.

Data Warehouse

Long running batch jobs

Optimized for read operations

Holds data from multiple sources

Holds data over a long period of time

Data may be lagged, not real-time

Data Warehouse

Apache Hive is an open-source data warehouse

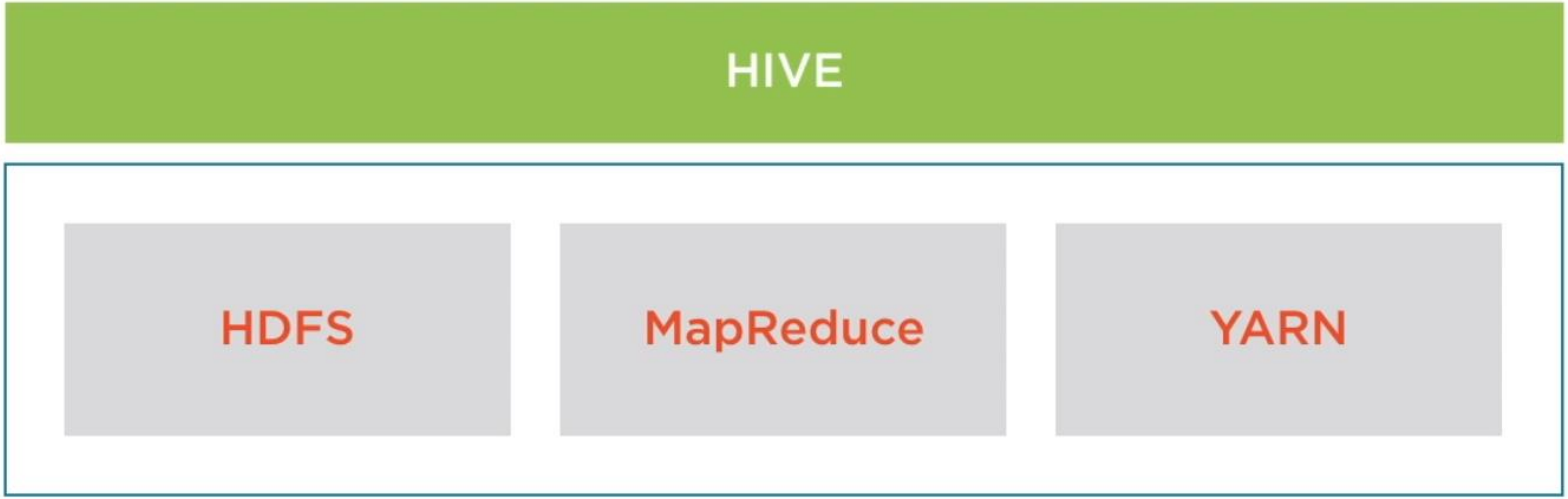
Hive

- Apache Hive is a data warehouse software project built on top of Apache Hadoop for providing data query and analysis.

Why Hive over MapReduce?

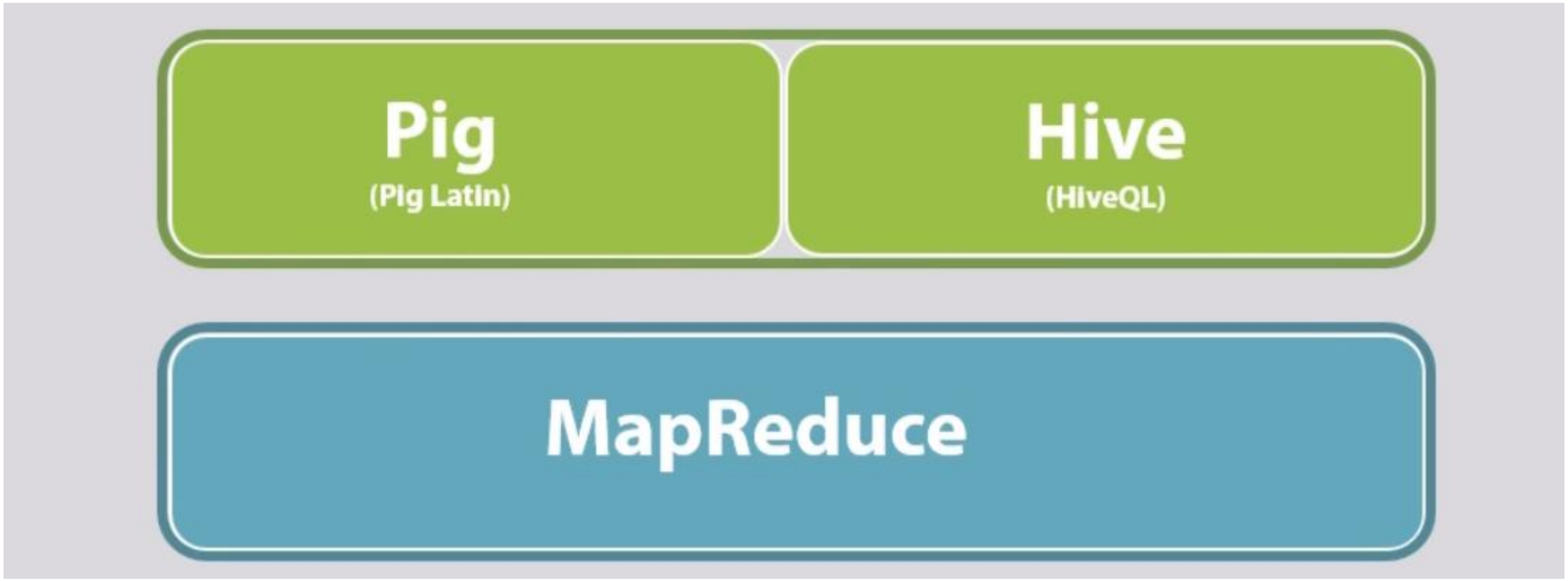
- Fewer lines of code
- Quickly test queries
- No Java experience

Hive on Hadoop



Do we have to write MapReduce code to work with Hive?

No


Pig
(Pig Latin)
Hive
(HiveQL)
MapReduce

Hive vs Pig

HiveQL

- Declarative language based on SQL and schema bound

Pig Latin

- Procedural or data flow programming language with ability to declare schema at runtime

https://cwiki.apache.org/confluence/display/Hive/LanguageManual

HiveQL

- HiveQL is the Hive query language. An SQL abstraction to integrate SQL-like queries into the underlying Java without the need to implement queries in the low-level Java API

HiveQL

Hive Query Language

A SQL-like interface to the underlying data

HiveQL

Modeled on the Structured Query Language (SQL)

Familiar to analysts and engineers

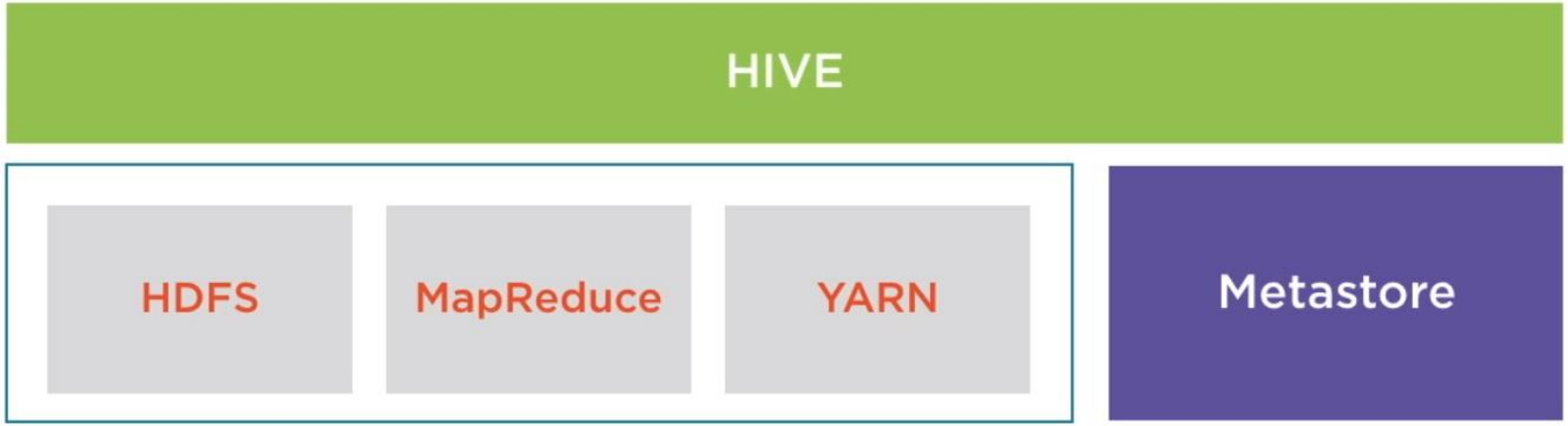
Simple query constructs

- select
- group by
- join

HiveQL

Hive exposes files in HDFS in the form of tables to the user

The Hive Metastore

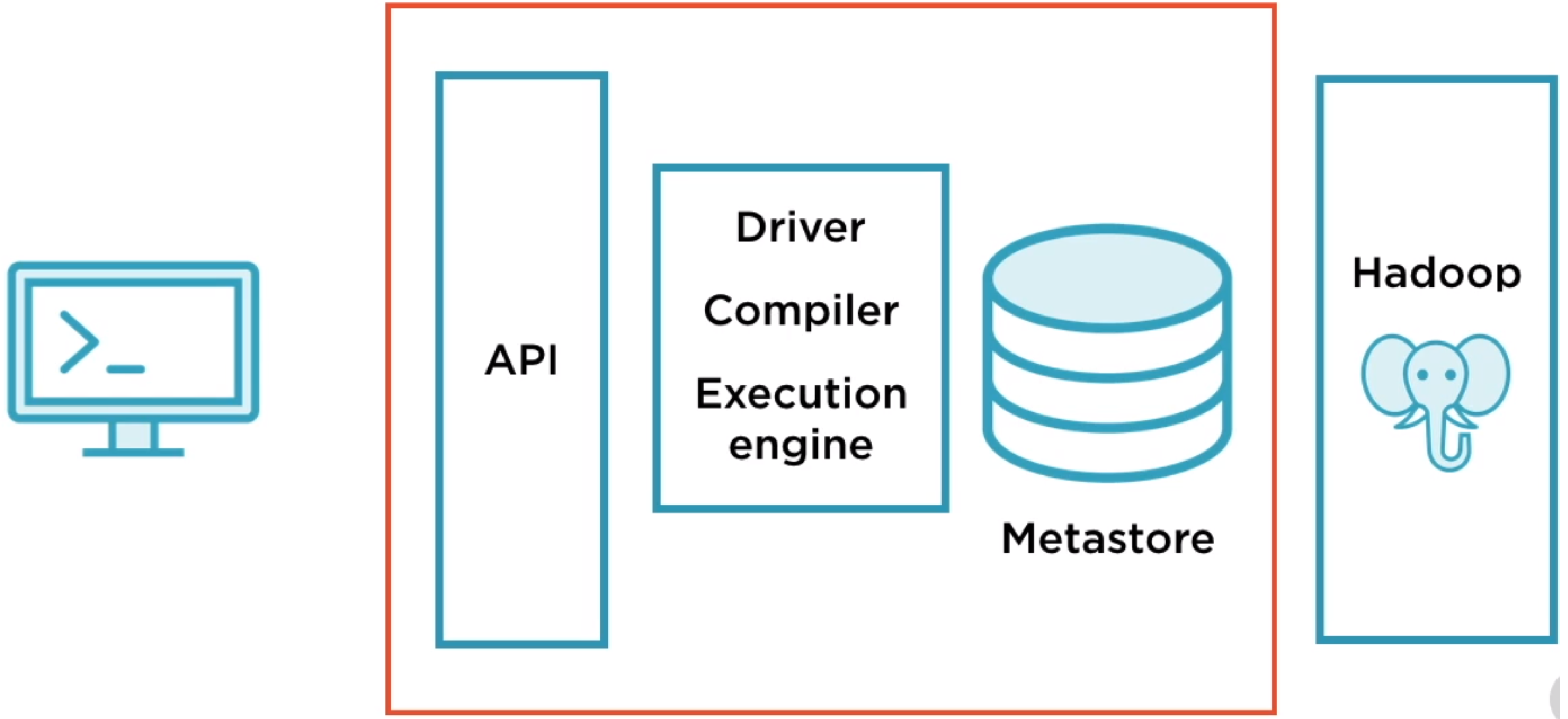


Exposes the file-based storage of HDFS in the form of tables

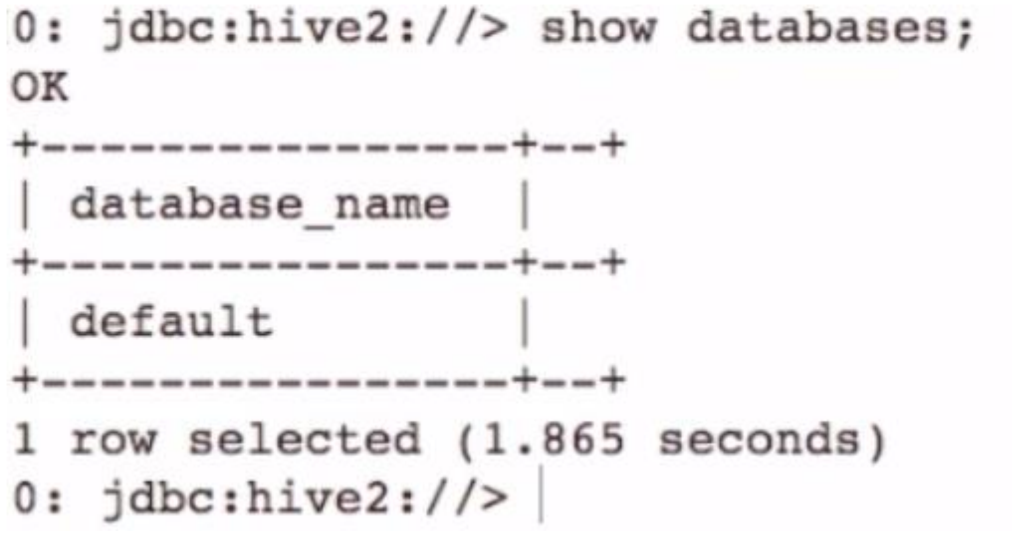
Hive vs. RDBMS

Hive	RDBMS
Large datasets	Small datasets
Parallel computations	Serial computations
High latency	Low latency
Read operations	Read/write operations
Not ACID compliant by default	ACID compliant
HiveQL	SQL

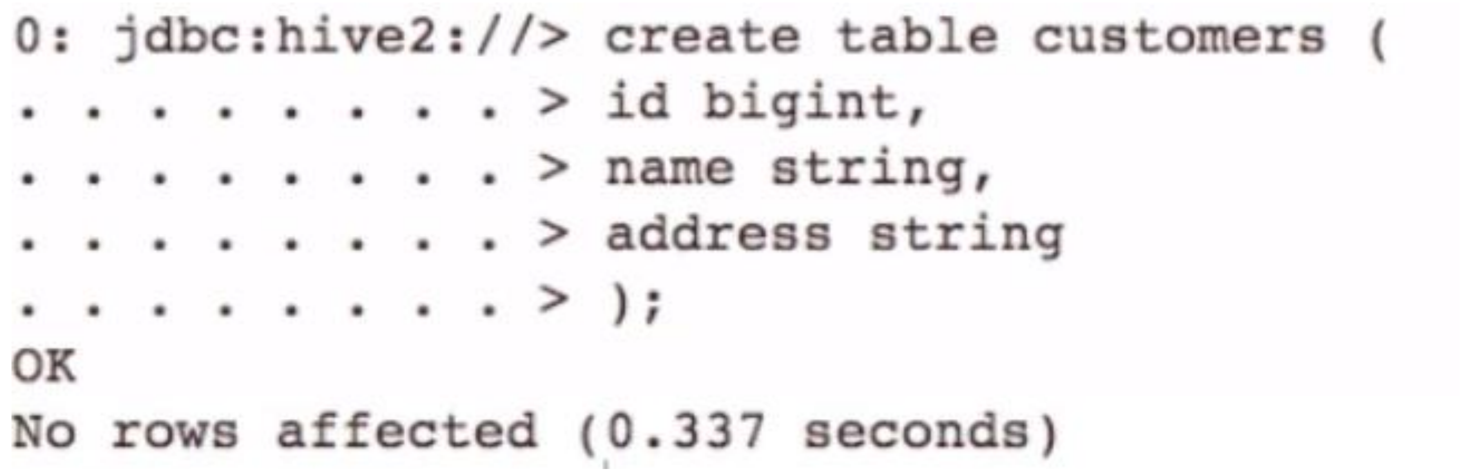
Basic Hive Building Blocks



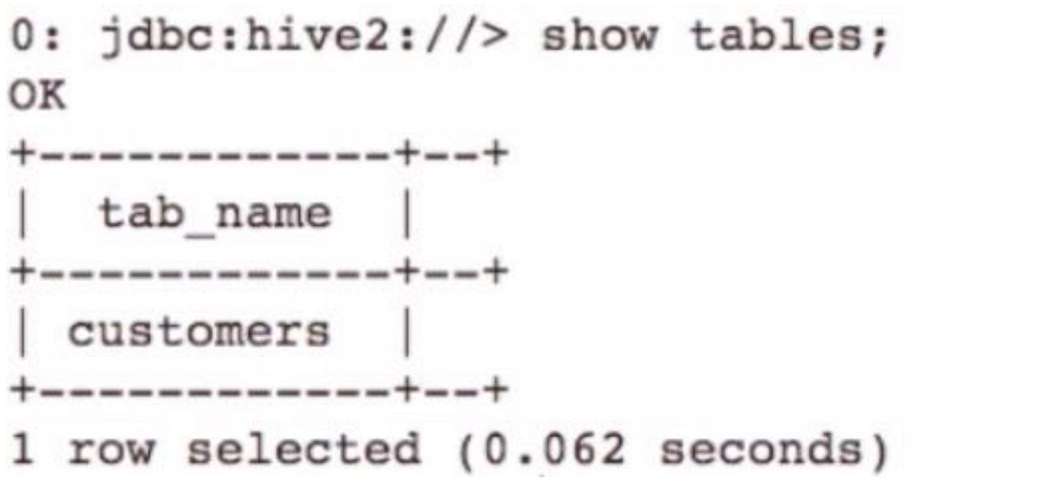
Basic Hive operations



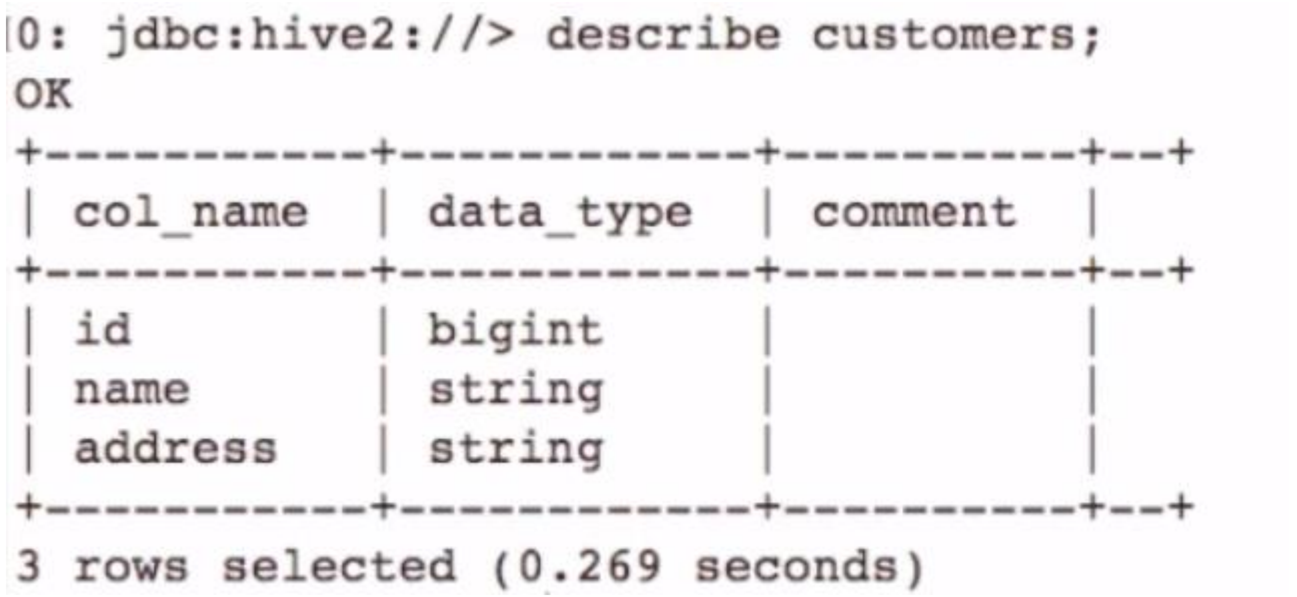
```
0: jdbc:hive2://> show databases;
OK
+----------------+--+
| database_name  |
+----------------+--+
| default        |
+----------------+--+
1 row selected (1.865 seconds)
0: jdbc:hive2://>
```



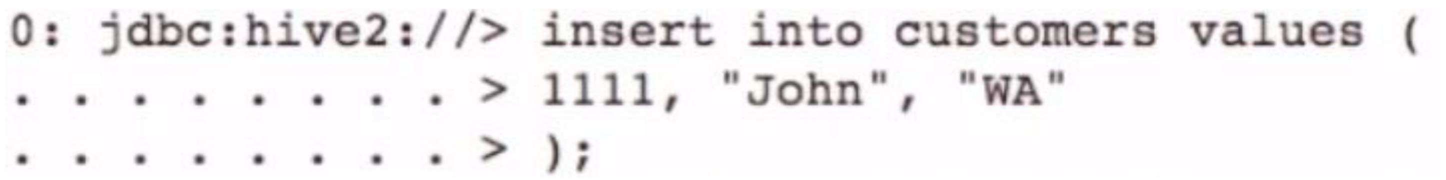
```
0: jdbc:hive2://> create table customers (
. . . . . . . . > id bigint,
. . . . . . . . > name string,
. . . . . . . . > address string
. . . . . . . . > );
OK
No rows affected (0.337 seconds)
```



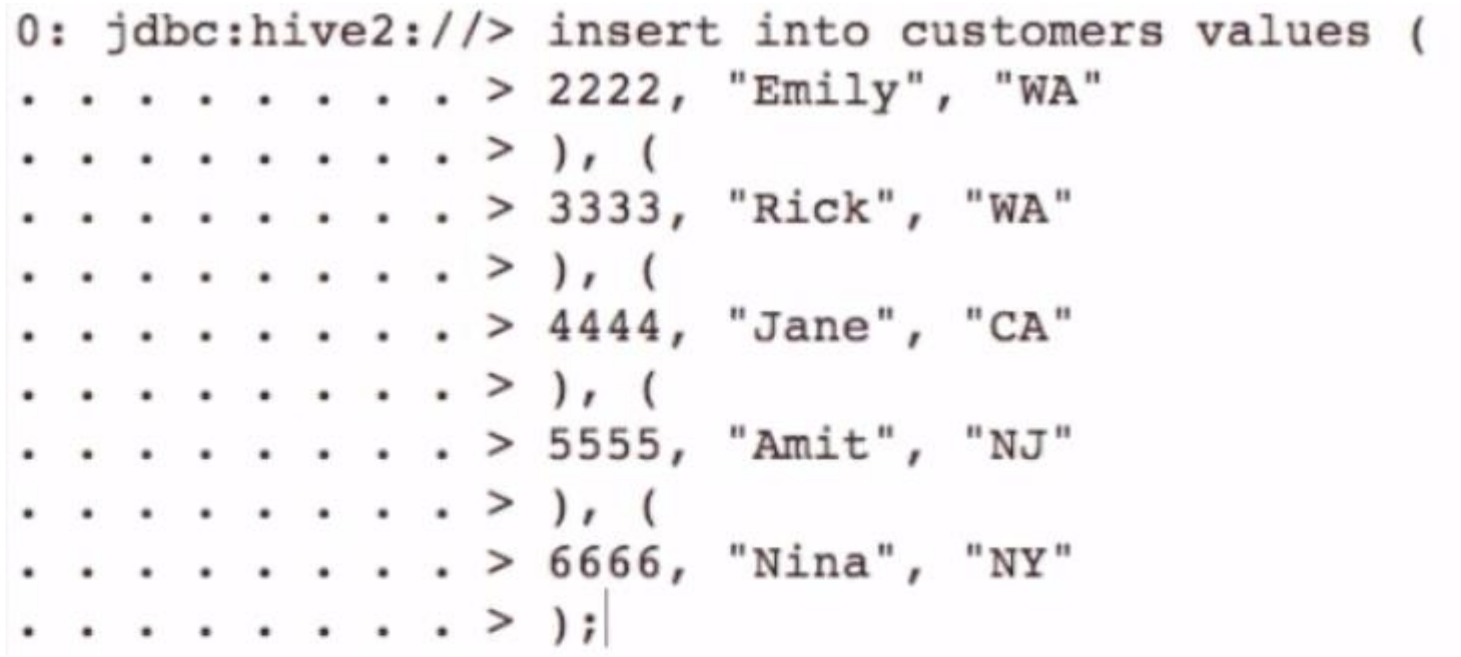
```
0: jdbc:hive2://> show tables;
OK
+------------+--+
|  tab_name  |
+------------+--+
| customers  |
+------------+--+
1 row selected (0.062 seconds)
```



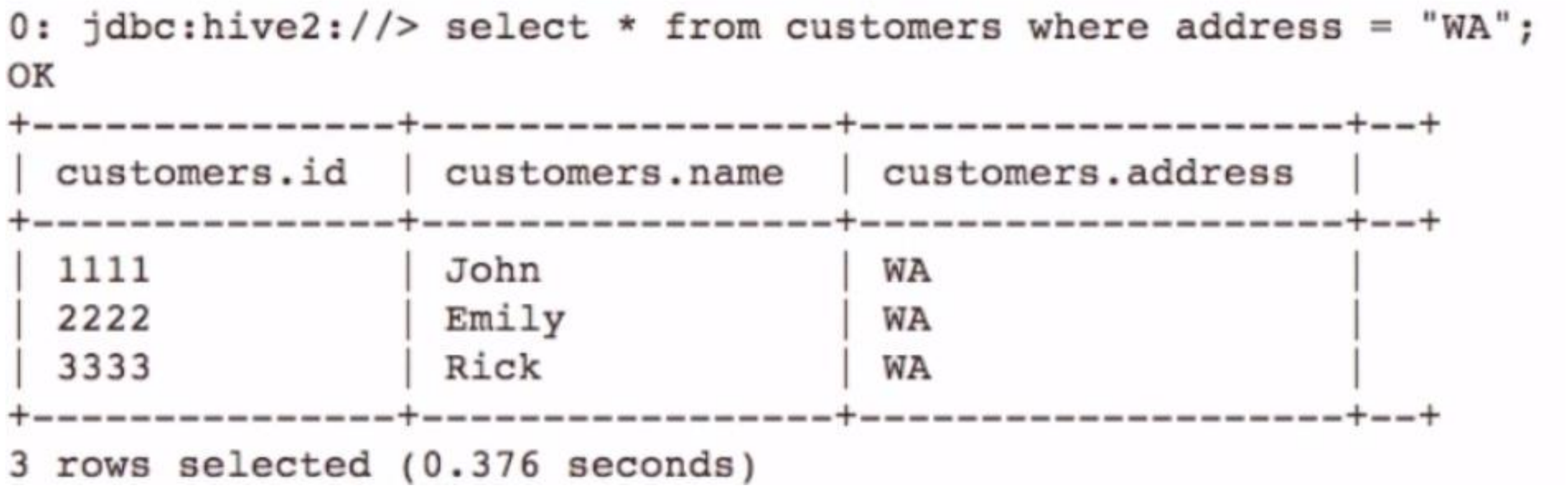
```
0: jdbc:hive2://> describe customers;
OK
+-----------+------------+-----------+--+
| col_name  | data_type  | comment   |
+-----------+------------+-----------+--+
| id        | bigint     |           |
| name      | string     |           |
| address   | string     |           |
+-----------+------------+-----------+--+
3 rows selected (0.269 seconds)
```



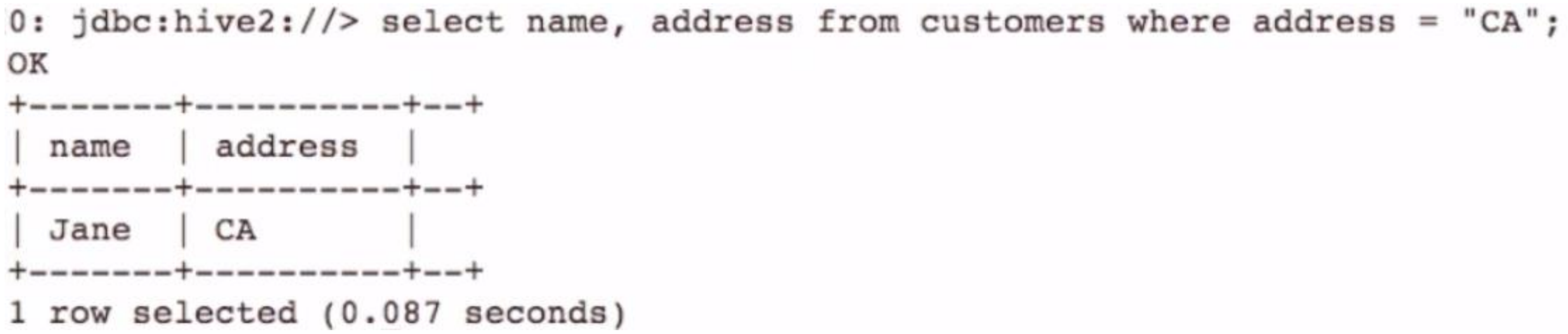
```
0: jdbc:hive2://> insert into customers values (
. . . . . . . . > 1111, "John", "WA"
. . . . . . . . > );
```



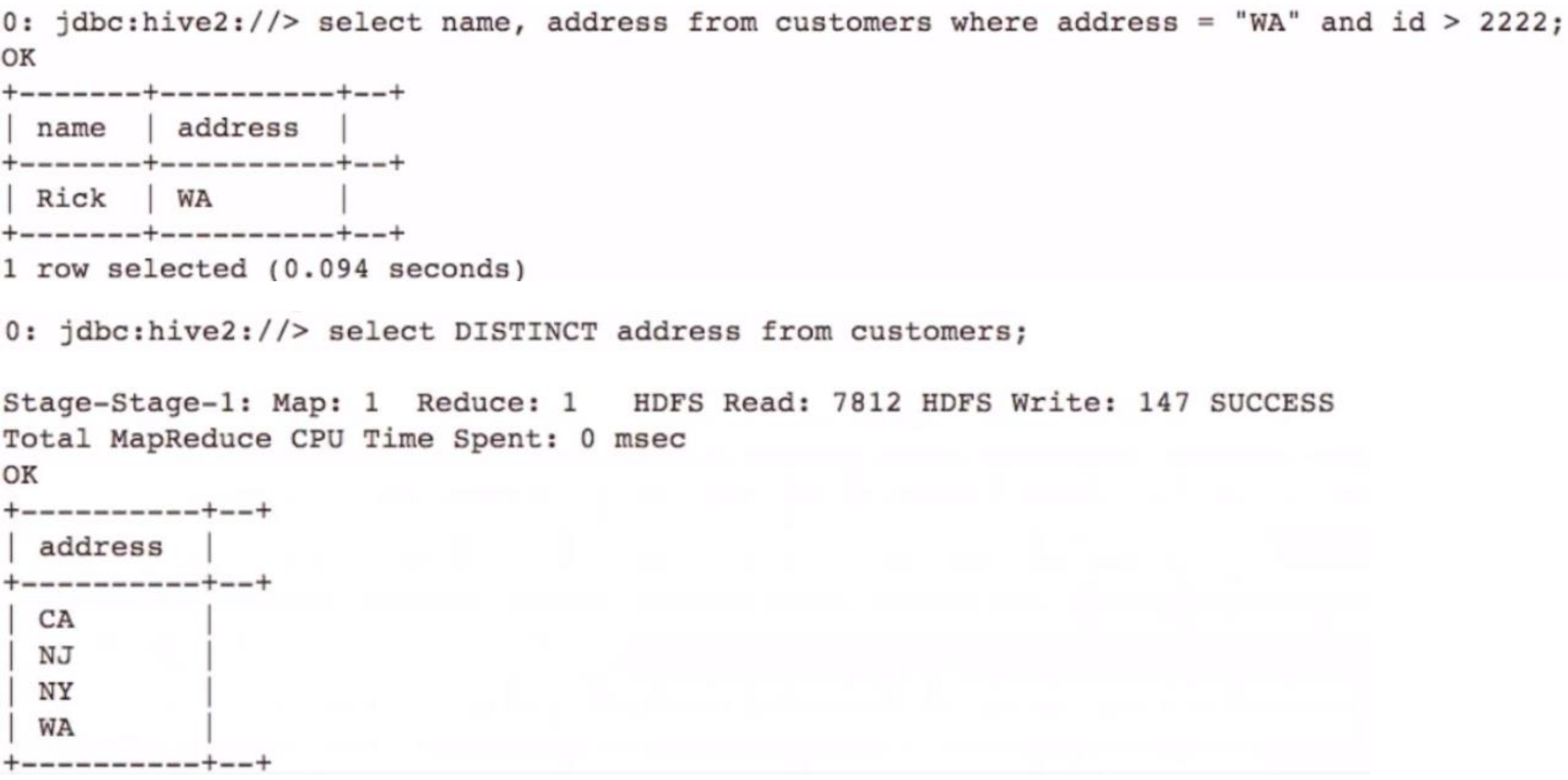
```
0: jdbc:hive2://> insert into customers values (
. . . . . . . . > 2222, "Emily", "WA"
. . . . . . . . > ), (
. . . . . . . . > 3333, "Rick", "WA"
. . . . . . . . > ), (
. . . . . . . . > 4444, "Jane", "CA"
. . . . . . . . > ), (
. . . . . . . . > 5555, "Amit", "NJ"
. . . . . . . . > ), (
. . . . . . . . > 6666, "Nina", "NY"
. . . . . . . . > );
```



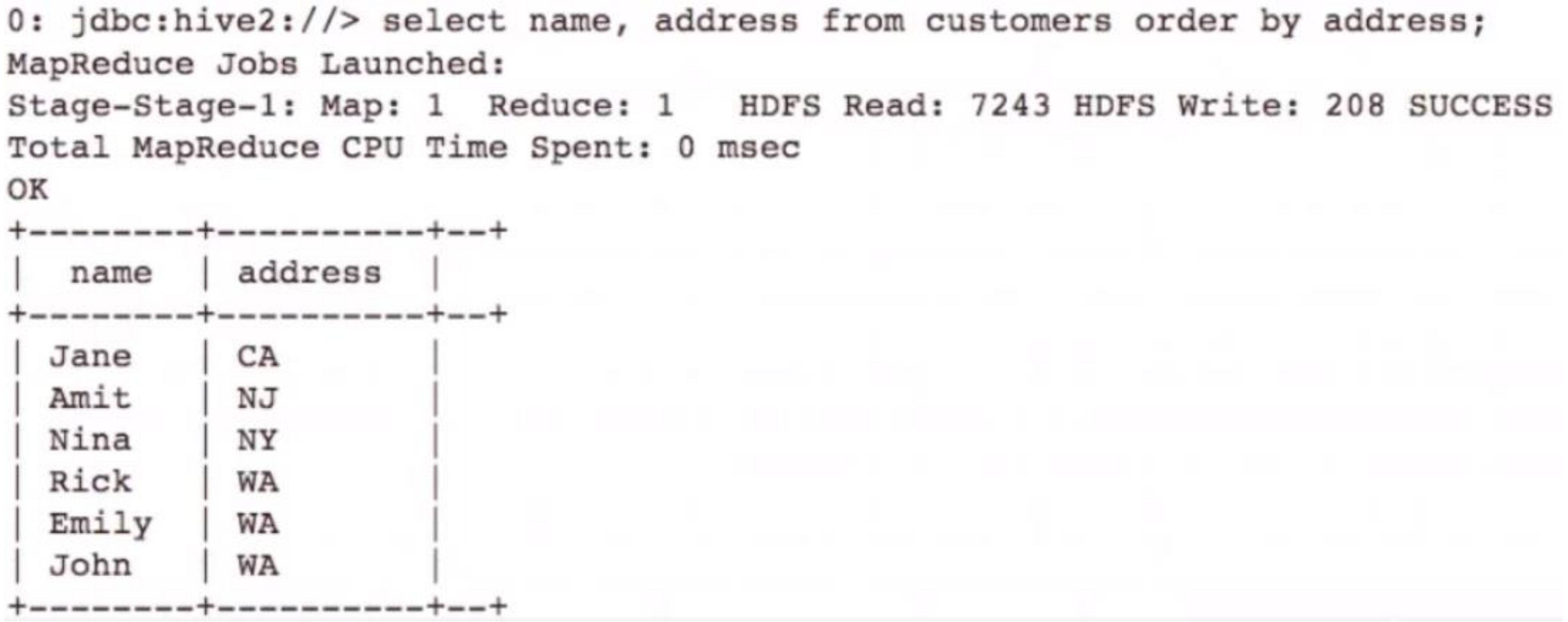
```
0: jdbc:hive2://> select * from customers where address = "WA";
OK
+---------------+-----------------+---------------------+--+
| customers.id  | customers.name  | customers.address   |
+---------------+-----------------+---------------------+--+
| 1111          | John            | WA                  |
| 2222          | Emily           | WA                  |
| 3333          | Rick            | WA                  |
+---------------+-----------------+---------------------+--+
3 rows selected (0.376 seconds)
```



```
0: jdbc:hive2://> select name, address from customers where address = "CA";
OK
+-------+----------+--+
| name  | address  |
+-------+----------+--+
| Jane  | CA       |
+-------+----------+--+
1 row selected (0.087 seconds)
```



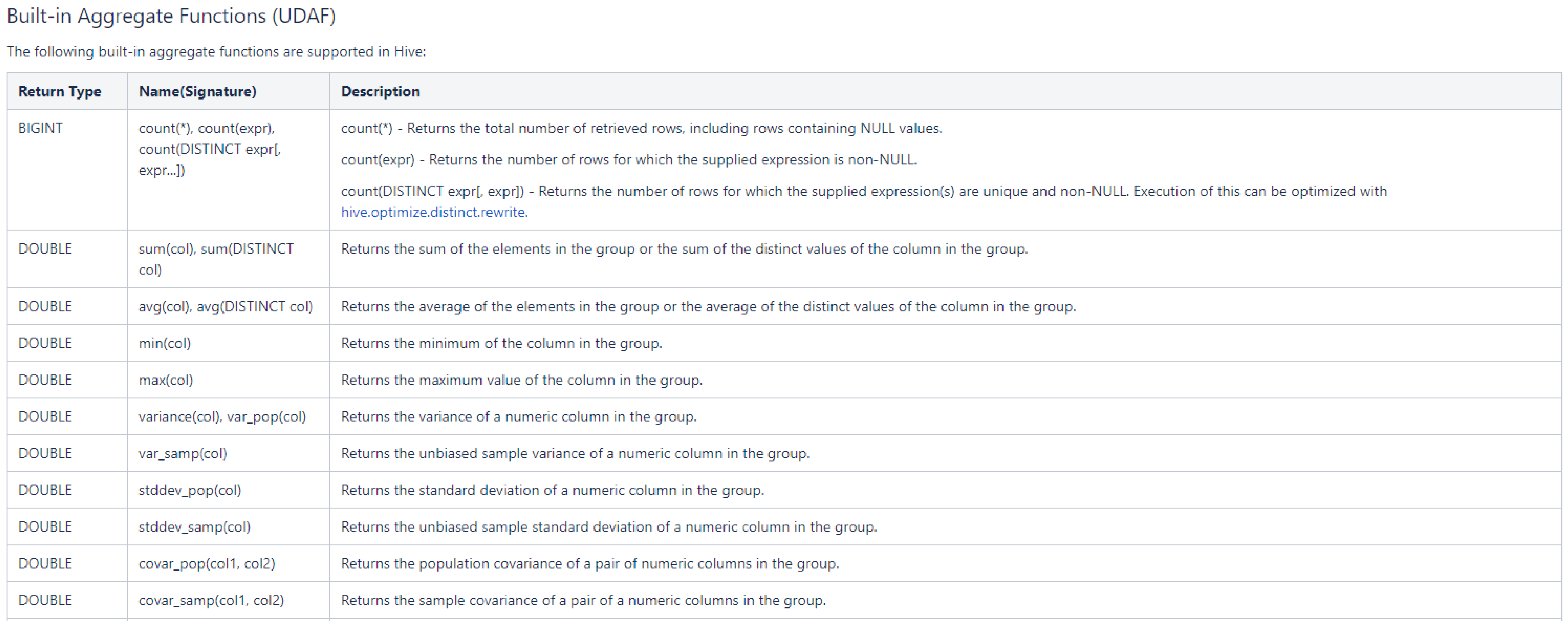
```
0: jdbc:hive2://> select name, address from customers where address = "WA" and id > 2222;
OK
+-------+----------+--+
| name  | address  |
+-------+----------+--+
| Rick  | WA       |
+-------+----------+--+
1 row selected (0.094 seconds)

0: jdbc:hive2://> select DISTINCT address from customers;

Stage-Stage-1: Map: 1  Reduce: 1   HDFS Read: 7812 HDFS Write: 147 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
+----------+--+
| address  |
+----------+--+
| CA       |
| NJ       |
| NY       |
| WA       |
+----------+--+
```



```
0: jdbc:hive2://> select name, address from customers order by address;
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   HDFS Read: 7243 HDFS Write: 208 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
+--------+----------+--+
|  name  | address  |
+--------+----------+--+
| Jane   | CA       |
| Amit   | NJ       |
| Nina   | NY       |
| Rick   | WA       |
| Emily  | WA       |
| John   | WA       |
+--------+----------+--+
```

```
0: jdbc:hive2://> select count(*) from customers;
```

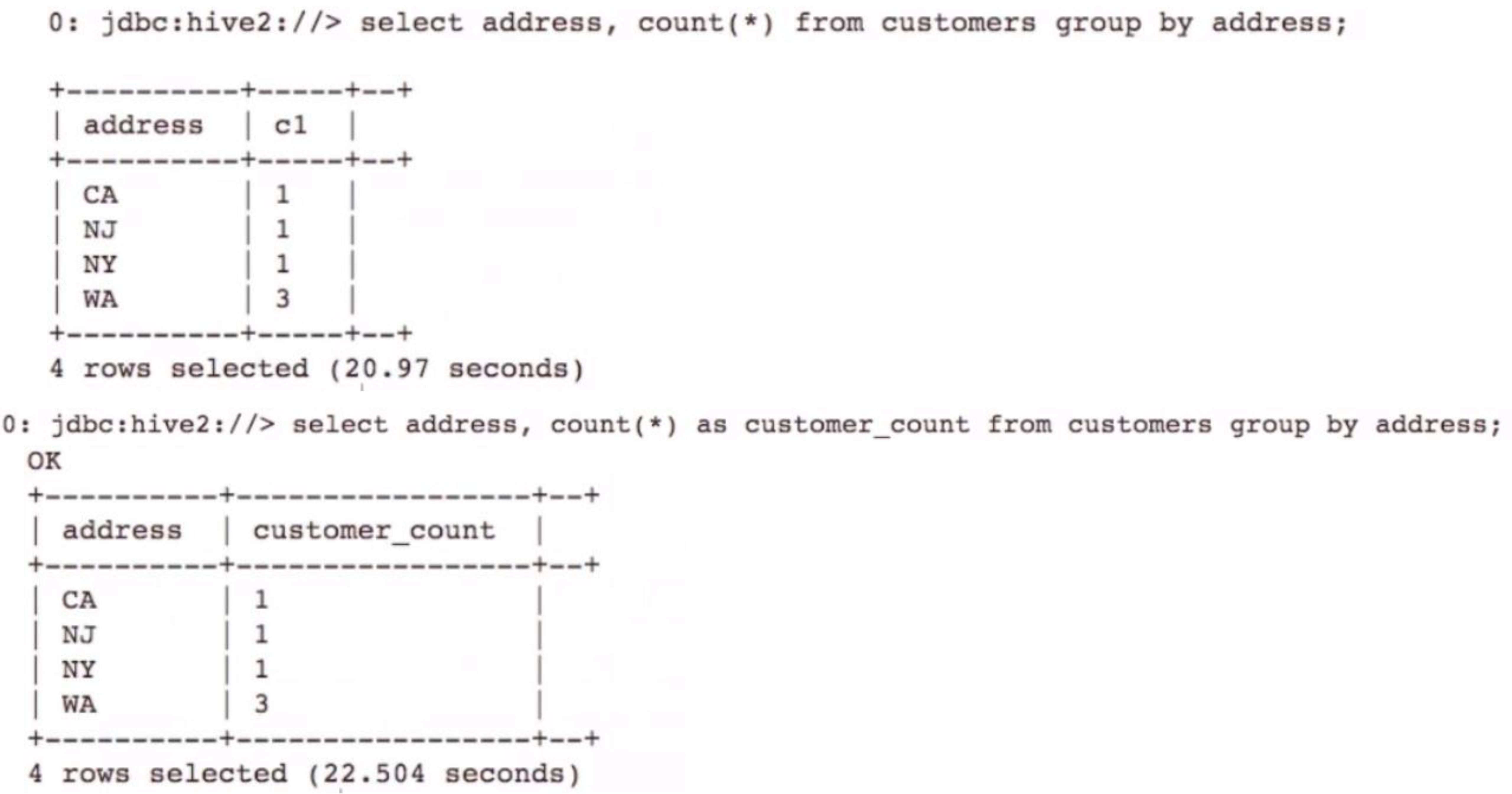
https://cwiki.apache.org/confluence/display/Hive/LanguageManual+UDF



Built-in Aggregate Functions (UDAF)

The following built-in aggregate functions are supported in Hive:

Return Type	**Name(Signature)**	**Description**
BIGINT	count(*), count(expr), count(DISTINCT expr[, expr...])	count(*) - Returns the total number of retrieved rows, including rows containing NULL values. count(expr) - Returns the number of rows for which the supplied expression is non-NULL. count(DISTINCT expr[, expr]) - Returns the number of rows for which the supplied expression(s) are unique and non-NULL. Execution of this can be optimized with hive.optimize.distinct.rewrite.
DOUBLE	sum(col), sum(DISTINCT col)	Returns the sum of the elements in the group or the sum of the distinct values of the column in the group.
DOUBLE	avg(col), avg(DISTINCT col)	Returns the average of the elements in the group or the average of the distinct values of the column in the group.
DOUBLE	min(col)	Returns the minimum of the column in the group.
DOUBLE	max(col)	Returns the maximum value of the column in the group.
DOUBLE	variance(col), var_pop(col)	Returns the variance of a numeric column in the group.
DOUBLE	var_samp(col)	Returns the unbiased sample variance of a numeric column in the group.
DOUBLE	stddev_pop(col)	Returns the standard deviation of a numeric column in the group.
DOUBLE	stddev_samp(col)	Returns the unbiased sample standard deviation of a numeric column in the group.
DOUBLE	covar_pop(col1, col2)	Returns the population covariance of a pair of numeric columns in the group.
DOUBLE	covar_samp(col1, col2)	Returns the sample covariance of a pair of a numeric columns in the group.



```
  0: jdbc:hive2://> select address, count(*) from customers group by address;

  +----------+-----+--+
  | address  | c1  |
  +----------+-----+--+
  | CA       | 1   |
  | NJ       | 1   |
  | NY       | 1   |
  | WA       | 3   |
  +----------+-----+--+
  4 rows selected (20.97 seconds)
0: jdbc:hive2://> select address, count(*) as customer_count from customers group by address;
 OK
 +----------+-----------------+--+
 | address  | customer_count  |
 +----------+-----------------+--+
 | CA       | 1               |
 | NJ       | 1               |
 | NY       | 1               |
 | WA       | 3               |
 +----------+-----------------+--+
 4 rows selected (22.504 seconds)
```

Hive Data Types

Boolean

Numeric

String

Timestamp

Primitive data types

Boolean

true or false

yes/no questions

Integers

Tinyint: 1 byte, range -128 to 128

Smallint: 2 bytes, range -2^{15} to $2^{15} - 1$

Int: 4 bytes, range -2^{31} to $2^{31} - 1$

Bigint: 8 bytes, range -2^{63} to $2^{63} - 1$

Decimals

Float: 4 bytes

Double: 8 bytes

Decimal: Arbitrary precision

- dec(10, 2)

String

String: Unbounded, variable length character string

Char: Fixed length character string

Varchar: Bounded, variable length character string

Timestamp

Integers: Unix timestamp in nanoseconds

Floats: Unix timestamp in seconds with decimal precision

String: JDBC compliant "YYYY-MM-DD HH:MM:SS.ffffffff"

Storing Data in Hive

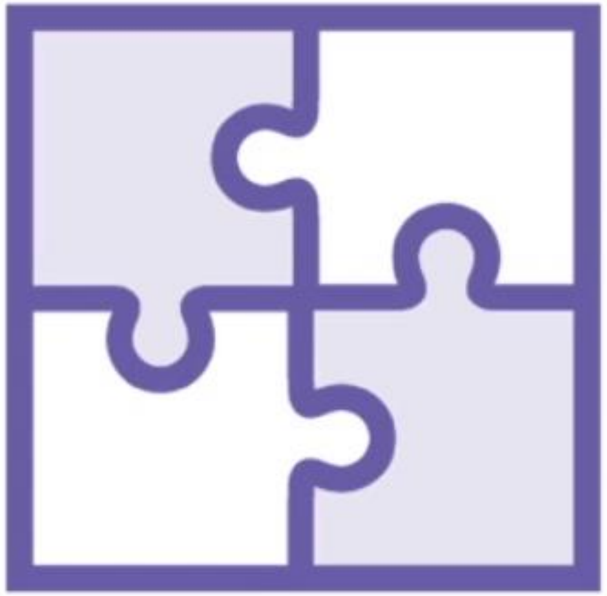
Data

The records in the table which holds the actual data

Metadata

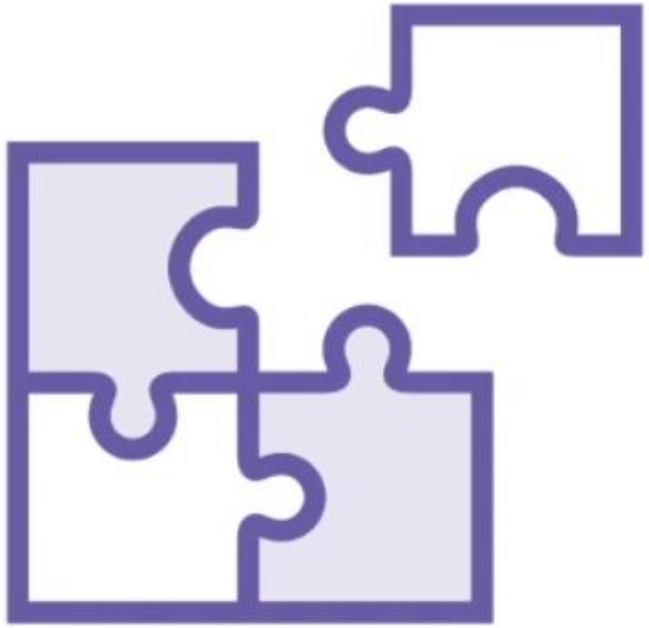
Information about the underlying data in the table

Hive Tables



Managed

Data managed by Hive and stored in the warehouse directory



External

Data not fully managed by Hive and exists outside the warehouse directory

Managed Tables

All tables so far have been managed tables

Hive owns the files and directories

These can be modified by other technologies

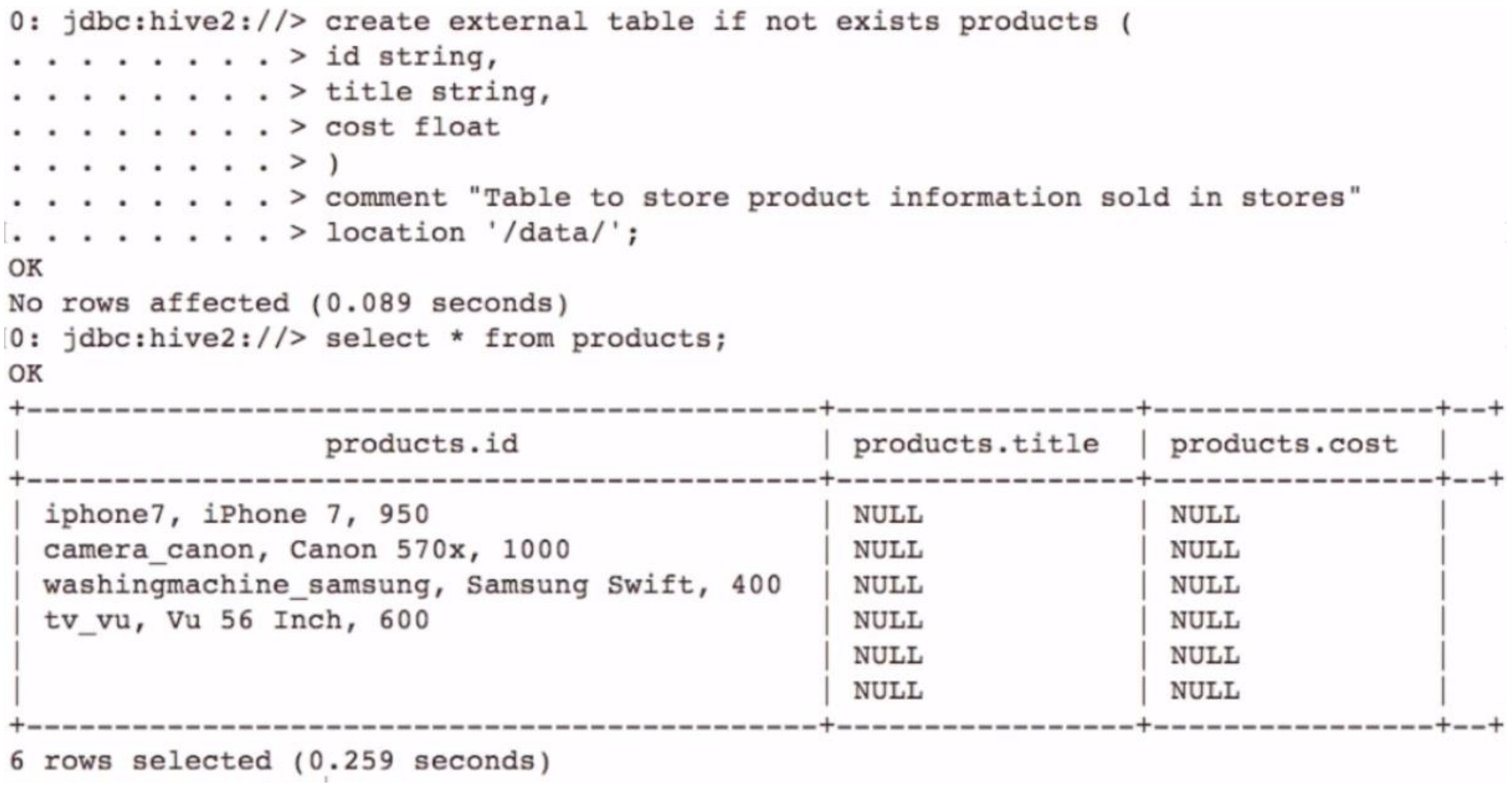
Deleting a managed table deletes both data and metadata

External Tables

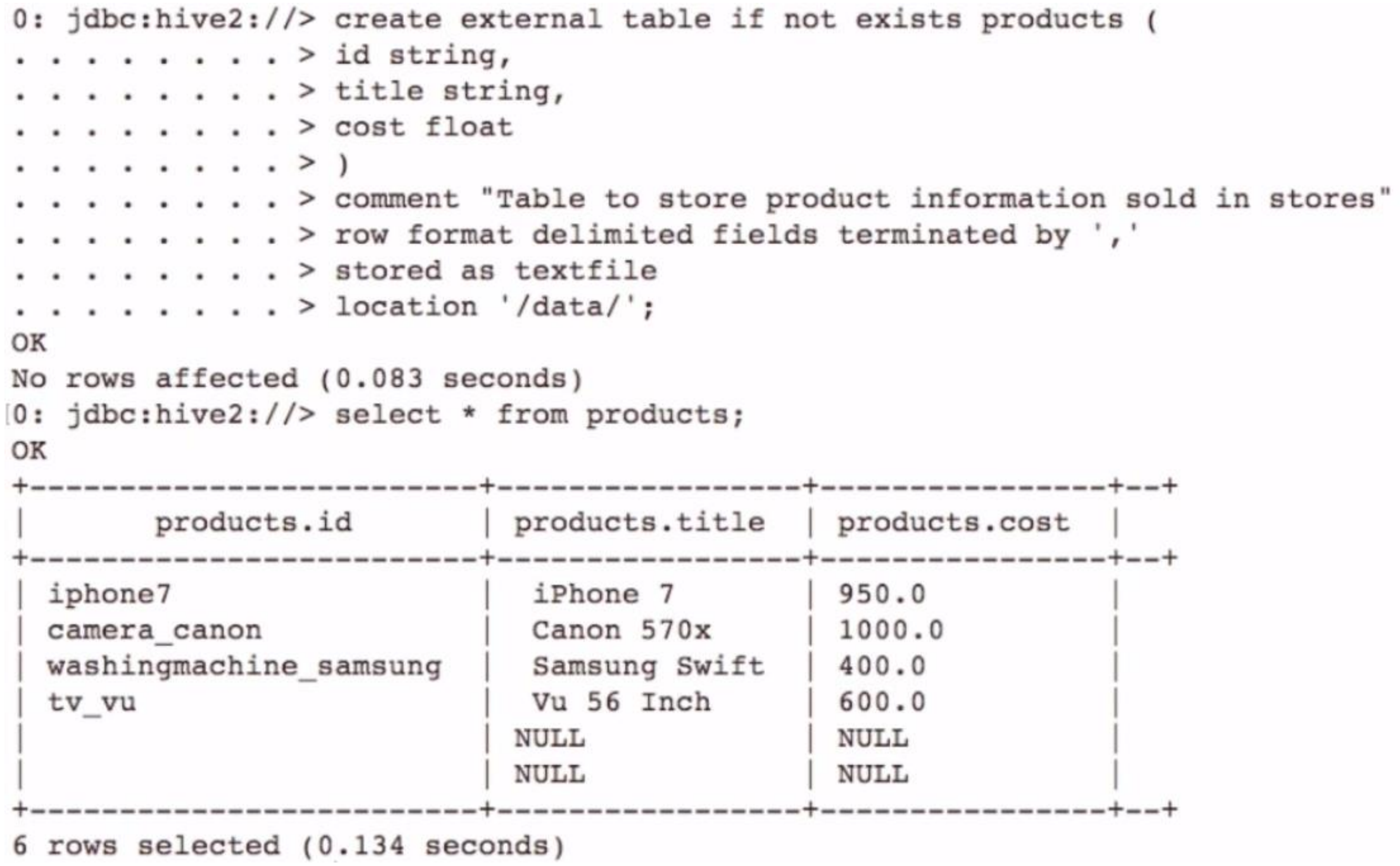
Share the underlying data across other technologies

Hadoop, Pig, HBase all of these may access and edit those files

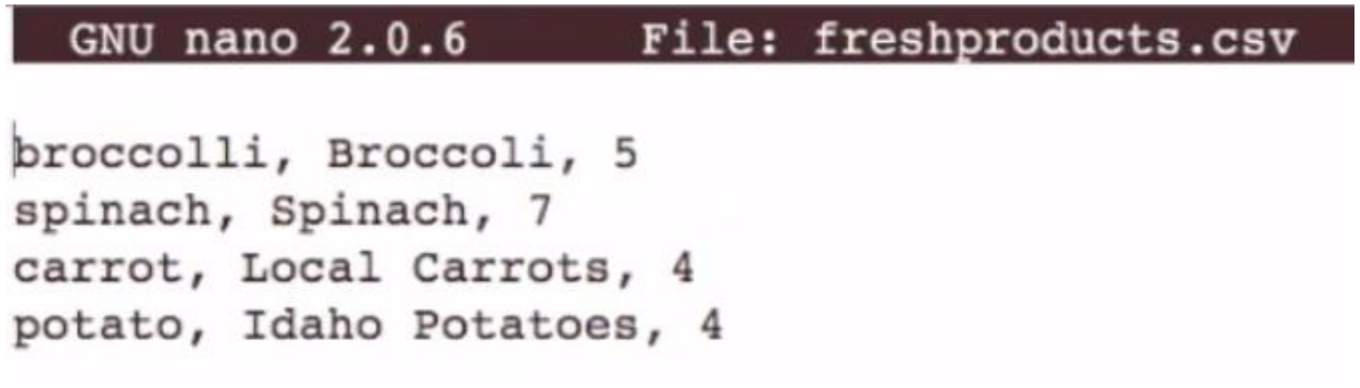
Deleting an external table deletes only the metadata



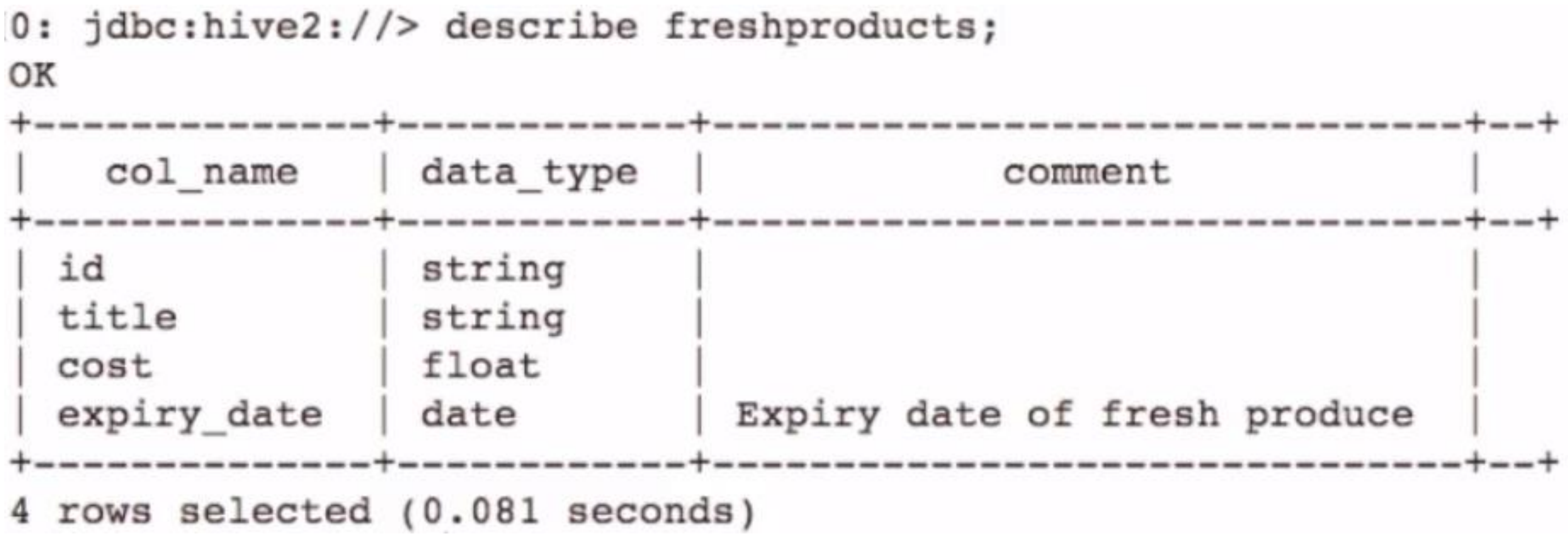
```
0: jdbc:hive2://> create external table if not exists products (
. . . . . . . . > id string,
. . . . . . . . > title string,
. . . . . . . . > cost float
. . . . . . . . > )
. . . . . . . . > comment "Table to store product information sold in stores"
. . . . . . . . > location '/data/';
OK
No rows affected (0.089 seconds)
0: jdbc:hive2://> select * from products;
OK
+----------------------------------------------+-----------------+-----------------+--+
|                 products.id                  | products.title  | products.cost   |
+----------------------------------------------+-----------------+-----------------+--+
| iphone7, iPhone 7, 950                       | NULL            | NULL            |
| camera_canon, Canon 570x, 1000               | NULL            | NULL            |
| washingmachine_samsung, Samsung Swift, 400   | NULL            | NULL            |
| tv_vu, Vu 56 Inch, 600                       | NULL            | NULL            |
|                                              | NULL            | NULL            |
|                                              | NULL            | NULL            |
+----------------------------------------------+-----------------+-----------------+--+
6 rows selected (0.259 seconds)
```



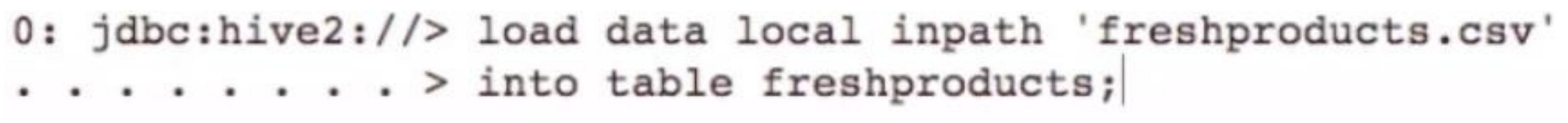
```
0: jdbc:hive2://> create external table if not exists products (
. . . . . . . . > id string,
. . . . . . . . > title string,
. . . . . . . . > cost float
. . . . . . . . > )
. . . . . . . . > comment "Table to store product information sold in stores"
. . . . . . . . > row format delimited fields terminated by ','
. . . . . . . . > stored as textfile
. . . . . . . . > location '/data/';
OK
No rows affected (0.083 seconds)
0: jdbc:hive2://> select * from products;
OK
+-------------------------+-----------------+-----------------+--+
|       products.id       | products.title  | products.cost   |
+-------------------------+-----------------+-----------------+--+
| iphone7                 |  iPhone 7       | 950.0           |
| camera_canon            |  Canon 570x     | 1000.0          |
| washingmachine_samsung  |  Samsung Swift  | 400.0           |
| tv_vu                   |  Vu 56 Inch     | 600.0           |
|                         | NULL            | NULL            |
|                         | NULL            | NULL            |
+-------------------------+-----------------+-----------------+--+
6 rows selected (0.134 seconds)
```



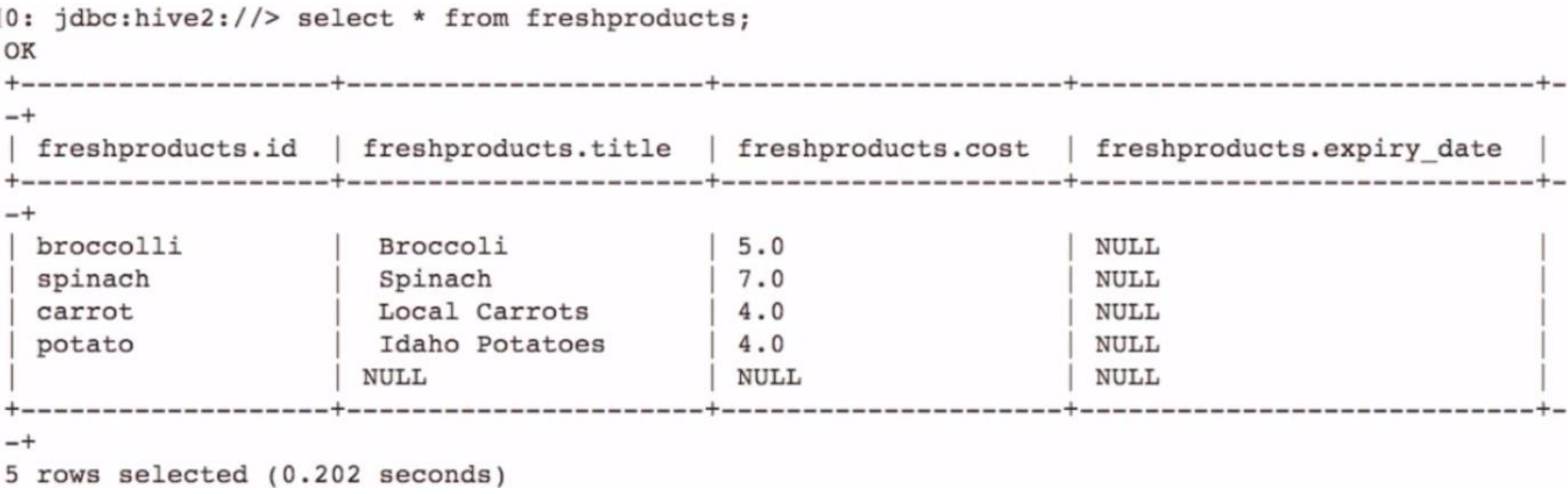
```
  GNU nano 2.0.6        File: freshproducts.csv

broccolli, Broccoli, 5
spinach, Spinach, 7
carrot, Local Carrots, 4
potato, Idaho Potatoes, 4
```



```
0: jdbc:hive2://> describe freshproducts;
OK
+--------------+------------+--------------------------------+--+
|   col_name   | data_type  |            comment             |
+--------------+------------+--------------------------------+--+
| id           | string     |                                |
| title        | string     |                                |
| cost         | float      |                                |
| expiry_date  | date       | Expiry date of fresh produce   |
+--------------+------------+--------------------------------+--+
4 rows selected (0.081 seconds)
```



```
0: jdbc:hive2://> load data local inpath 'freshproducts.csv'
. . . . . . . . > into table freshproducts;
```



```
0: jdbc:hive2://> select * from freshproducts;
OK
+--------------------+----------------------+---------------------+---------------------------+-
-+
| freshproducts.id   | freshproducts.title  | freshproducts.cost  | freshproducts.expiry_date  |
+--------------------+----------------------+---------------------+---------------------------+-
-+
| broccolli          |  Broccoli            | 5.0                 | NULL                       |
| spinach            |  Spinach             | 7.0                 | NULL                       |
| carrot             |  Local Carrots       | 4.0                 | NULL                       |
| potato             |  Idaho Potatoes      | 4.0                 | NULL                       |
|                    | NULL                 | NULL                | NULL                       |
+--------------------+----------------------+---------------------+---------------------------+-
-+
5 rows selected (0.202 seconds)
```

Summary

Hive is democratizing Map Reduce system for Data Analytics and offers SQL developers a technology to analyze lots of data from different sources.

HiveQL although being a SQL like language has its differences especially when talking about ACID compliance.

Microsoft