

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**



**Báo Cáo Giữa Kỳ**

**Môn học:** Phân tích dữ liệu lớn

**Đề tài**: Apache Hive

GVHD: Th.s Lê Thị Minh Châu

Sinh viên thực hiện:

Trần Công Tuấn Mạnh - 19133035

Phan Hoàng Việt - 17133072

Tp. HCM, ngày 04 tháng 05 năm 2022

MỤC LỤC

[**I - Giới thiệu Hive**](#_gblsdpgu8wo7) **6**

[Hive là gì ?](#_307vdxucsvql) 6

[Đặc điểm về Hive](#_i061ufdwejue) 6

[Kiến trúc của Hive](#_gudqkf7luhnn) 7

[Quy trình làm việc của Hive](#_n84r98rv0i97) 8

[**II - Cài đặt Hive**](#_t4in9hr1l7d5) **11**

[Bước 1: Tải và cài đặt Hive](#_gjdgxs) 11

[Bước 2: Tải và cài đặt Derby](#_30j0zll) 14

[Bước 3: Cấu hình Metastore cho Hive](#_1fob9te) 15

[Bước 4: Kiểm tra Hive](#_3znysh7) 17

[**III - Cú pháp Hive**](#_so305v6yhfqk) **19**

[Các loại dữ liệu trong Hive](#_6kr06gup2awh) 19

[Tạo Database](#_g5yhflmrfmbo) 23

[Xóa Database](#_1vomk389tvl8) 23

[Tạo Table](#_lmx81qgwqhi6) 24

[Alter Table](#_42n2aug6b65t) 24

[Drop Table](#_neeml68atx2i) 26

[Phân vùng (Partitioning):](#_1p7gpuprpknb) 26

[Các toán tử tích hợp (Built-in Operators)](#_8tebgzm1p00p) 27

[Các hàm tích hợp trong Hive (Built-in Functions) 28](#_ooh9wi3wvk9j) View và Indexes 29

Hive QL 30

**IV - Demo**

Ý tưởng Demo 34

Build Demo 36

Run Demo 38

# I - Giới thiệu Hive

## Hive là gì ?

Hive là một công cụ cơ sở hạ tầng kho dữ liệu để xử lý dữ liệu có cấu trúc trong Hadoop. Nó nằm trên Hadoop để tóm tắt Dữ liệu lớn và giúp dễ dàng truy vấn, phân tích.

## Đặc điểm về Hive

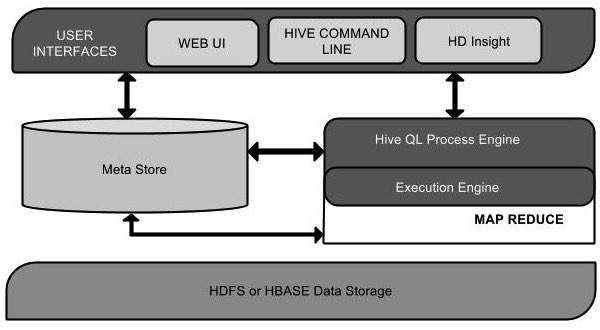
### 2.1 Hive không phải là

* Cơ sở dữ liệu quan hệ
* Một thiết kế cho Xử lý giao dịch trực tuyến (OLTP)
* Một ngôn ngữ cho các truy vấn thời gian thực và cập nhật cấp hàng

### 2.2 Hive là

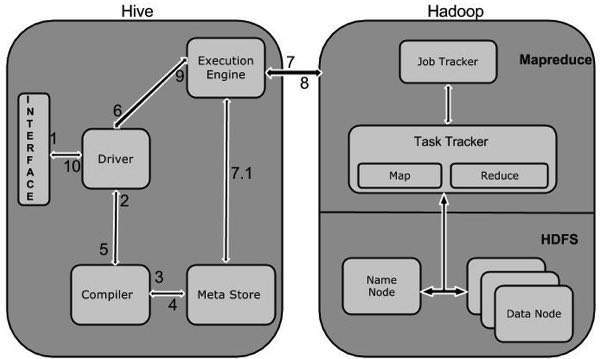
* Nó lưu trữ lược đồ trong cơ sở dữ liệu và dữ liệu được xử lý thành HDFS.
* Nó được thiết kế cho OLAP.
* Nó cung cấp ngôn ngữ kiểu SQL để truy vấn được gọi là HiveQL hoặc HQL.
* Nó quen thuộc, nhanh chóng, có thể mở rộng và mở rộng.

## Kiến trúc của Hive



|  |  |
| --- | --- |
| **Unit Name** | **Operation** |
| User Interface | Hive là một phần mềm cơ sở hạ tầng kho dữ liệu có thể tạo ra sự tương tác giữa người dùng và HDFS. Các giao diện người dùng mà Hive hỗ trợ là Hive Web UI, dòng lệnh Hive và Hive HD Insight. |
| Meta Store | Hive chọn các máy chủ cơ sở dữ liệu tương ứng để lưu trữ lược đồ hoặc Siêu dữ liệu của bảng, cơ sở dữ liệu, cột trong bảng, kiểu dữ liệu của chúng và ánh xạ HDFS. |
| HiveQL Process Engine | HiveQL tương tự như SQL để truy vấn thông tin lược đồ trên Metastore. Đây là một trong những phương pháp thay thế cách tiếp cận truyền thống cho chương trình MapReduce. Thay vì viết chương trình MapReduce bằng Java, chúng ta có thể viết một truy vấn cho công việc MapReduce và xử lý nó. |
| Execution Engine | Phần kết hợp của HiveQL process Engine và MapReduce là Công cụ thực thi Hive. Công cụ thực thi xử lý truy vấn và tạo kết quả giống như kết quả MapReduce. |
| HDFS or HBASE | Hệ thống tệp phân tán Hadoop hoặc HBASE là các kỹ thuật lưu trữ dữ liệu để lưu trữ dữ liệu vào hệ thống tệp. |

## Quy trình làm việc của Hive



|  |  |
| --- | --- |
| **Bước** | **Hoạt động** |
| 1 | Thực thi truy vấn  Giao diện Hive như Command Line hoặc Web UI gửi truy vấn đến Driver ( Trình điều khiển) (bất kỳ trình điều khiển cơ sở dữ liệu nào như JDBC, ODBC, v.v.) để thực thi. |
| 2 | Nhận kế hoạch  Trình điều khiển có sự trợ giúp của trình biên dịch truy vấn phân tích cú pháp truy vấn để kiểm tra cú pháp và kế hoạch truy vấn hoặc yêu cầu của truy vấn. |
| 3 | Nhận Metadata  Trình biên dịch gửi yêu cầu metadata đến Metastore (bất kỳ cơ sở dữ liệu nào). |
| 4 | Gửi metadata  Metastore gửi metadata dưới dạng phản hồi tới trình biên dịch. |
| 5 | Gửi kế hoạch  Trình biên dịch kiểm tra yêu cầu và gửi lại kế hoạch cho trình điều khiển. Đến đây, quá trình phân tích cú pháp và biên dịch truy vấn đã hoàn tất. |
| 6 | Thực hiện kế hoạch  Trình điều khiển gửi kế hoạch thực thi đến execution engine. |
| 7 | Thực thi công việc  Trong nội bộ, quá trình thực thi công việc là một công việc MapReduce. Execution engine sẽ gửi công việc đến JobTracker, nằm trong Name node và nó chỉ định công việc này cho TaskTracker, nằm trong Data node. Tại đây, truy vấn thực hiện công việc MapReduce. |
| 7.1 | Hoạt động siêu dữ liệu  Trong khi thực thi, Execution engine có thể thực thi các hoạt động siêu dữ liệu với Metastore. |
| 8 | Tìm nạp kết quả  Execution engine nhận kết quả từ các Data node. |
| 9 | Gửi kết quả  Execution engine gửi các giá trị kết quả đó đến trình điều khiển( Driver). |
| 10 | Gửi kết quả  Trình điều khiển gửi kết quả đến Hive Interfaces. |

# II - Cài đặt Hive

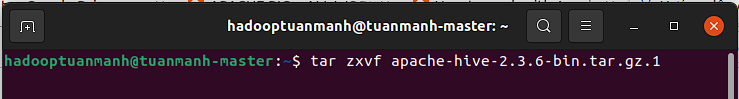
## Bước 1: Tải và cài đặt Hive

* Dùng lệnh *wget* để tải Hive phiên bản 2.3.6

Text

Description automatically generated

* Giải nén bằng lệnh *tar zxvf*



* Đổi tên thành *Hive* cho dễ quản lý

Text

Description automatically generated

Text

Description automatically generated

* Đặt biến môi trường cho *Hive* bằng cách gỗ lệnh ‘vim ~/.bashrc’

Text

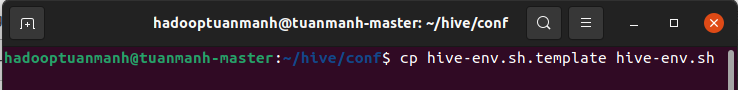
Description automatically generated

* Vào thư mục hive

Graphical user interface, text

Description automatically generated

* Copy file *hive.env.sh.template* thành file mới có tên *hive.env.sh*



* Vào file *hive-env.sh*

Text

Description automatically generated

* Thêm biến môi trường cho Hadoop

Text

Description automatically generated

## Bước 2: Tải và cài đặt Derby

* Cài đặt Derby phiên bản 10.14.1

Text

Description automatically generated

Text

Description automatically generated

* Giải nén và đổi tên Derby

Text

Description automatically generated

* Đặt biến môi trường cho *Derby* bằng cách gỗ lệnh ‘vim ~/.bashrc’ và thêm biến môi trường vào.

Text

Description automatically generated

* Source ~/.bashrc

Text

Description automatically generated

* Tạo thư mục *data* để lưu dữ liệu

Text

Description automatically generated





## Bước 3: Cấu hình Metastore cho Hive

* Vào thư mục *conf*

Text

Description automatically generated

* Copy file *hive-default.xml.template* thành file mới có tên *hive.site.xml*

Text

Description automatically generated

* Truy cập hive-site.xml

Text

Description automatically generated

* Thêm URL connection như bên dưới:

Text

Description automatically generated

* Tạo file jpox.properties

Text

Description automatically generated

* Thêm nội dung vào file

Text

Description automatically generated

## Bước 4: Kiểm tra Hive

* Trước khi start Hive, tạo một thư mục **/tmp** và thư mục Hive riêng trong HDFS. Ở đây chúng ta sẽ tạo thư mục  **/tuanmanh/hive/warehouse**:

Text

Description automatically generated

* Tạo thư mục **/tuanmanh** trên hdfs

Text

Description automatically generated

* Tiếp theo tạo thư mục /**hive**

Text

Description automatically generated

* Tạo thư mục warehouse

Text

Description automatically generated

* Cấp quyền cho thư mục

Text

Description automatically generated

Text

Description automatically generated

* Kiểm tra Hive đã cài đặt được chưa

Text

Description automatically generated

# III - Cú pháp Hive

## Các loại dữ liệu trong Hive

* Dữ liệu trong Hive được phân loại thành bốn kiểu, cụ thể như sau:
* Các loại cột
* Chữ viết
* Giá trị rỗng
* Các loại phức tạp
* Các loại cột

Kiểu cột được sử dụng làm kiểu dữ liệu cột của Hive. Chúng như sau:

* Các loại tích phân

Dữ liệu kiểu số nguyên có thể được chỉ định bằng cách sử dụng kiểu dữ liệu tích phân, INT. Khi phạm vi dữ liệu vượt quá phạm vi INT, bạn cần sử dụng BIGINT và nếu phạm vi dữ liệu nhỏ hơn INT, bạn sử dụng SMALLINT. TINYINT nhỏ hơn SMALLINT.

Bảng sau mô tả các kiểu dữ liệu INT khác nhau:

|  |  |  |
| --- | --- | --- |
| **Gõ phím** | **Postfix** | **Ví dụ** |
| TINYINT | Y | 10Y |
| SMALLINT | S | 10S |
| INT | - | 10 |
| BIGINT | L | 10L |

* Các loại chuỗi

Kiểu dữ liệu kiểu chuỗi có thể được chỉ định bằng cách sử dụng dấu nháy đơn ('') hoặc dấu nháy kép (""). Nó chứa hai kiểu dữ liệu: VARCHAR và CHAR. Hive theo sau các nhân vật thoát hiểm loại C.

Bảng sau mô tả các kiểu dữ liệu CHAR khác nhau:

|  |  |
| --- | --- |
| **Loại dữ liệu** | **Chiều dài** |
| VARCHAR | 1 đến 65355 |
| CHAR | 255 |

* Timestamp

Nó hỗ trợ dấu thời gian UNIX truyền thống với độ chính xác nano giây tùy chọn. Nó hỗ trợ định dạng java.sql.Timestamp “YYYY-MM-DD HH: MM: SS.fffffffff” và định dạng “yyyy-mm-dd hh: mm: ss.ffffffffff”.

* Dates

Giá trị DATE được mô tả ở định dạng năm/tháng/ngày trong biểu mẫu {{YYYY-MM-DD}}.

* Số thập phân

Kiểu DECIMAL trong Hive cũng giống như định dạng Big Decimal của Java. Nó được sử dụng để biểu diễn độ chính xác tùy ý bất biến. Cú pháp và ví dụ như sau:

DECIMAL(precision, scale)

decimal(10,0)

* Các loại Union

Union là một tập hợp các kiểu dữ liệu không đồng nhất. Bạn có thể tạo một thể hiện bằng cách sử dụng create union . Cú pháp và ví dụ như sau:

UNIONTYPE<int, double, array<string>, struct<a:int,b:string>>

{0:1}

{1:2.0}

{2:["three","four"]}

{3:{"a":5,"b":"five"}}

{2:["six","seven"]}

{3:{"a":8,"b":"eight"}}

{0:9}

{1:10.0}

* Chữ viết

Các ký tự sau được sử dụng trong Hive:

* Các loại dấu chấm động

Các loại dấu phẩy động không có gì khác ngoài các số có dấu thập phân. Nói chung, loại dữ liệu này bao gồm kiểu dữ liệu DOUBLE.

* Loại thập phân

Dữ liệu kiểu thập phân không là gì ngoài giá trị dấu phẩy động với phạm vi cao hơn kiểu dữ liệu DOUBLE. Phạm vi của loại thập phân là khoảng -10 -308 đến 10 308 .

* Giá trị rỗng

Các giá trị bị thiếu được biểu thị bằng giá trị đặc biệt NULL.

* Các loại phức tạp

Các kiểu dữ liệu phức tạp của Hive như sau:

* Mảng

Mảng trong Hive được sử dụng giống như cách chúng được sử dụng trong Java.

Syntax: ARRAY<data\_type>

* Bản đồ

Bản đồ trong Hive tương tự như Bản đồ Java.

Syntax: MAP<primitive\_type, data\_type>

* Cấu trúc

Các cấu trúc trong Hive tương tự như việc sử dụng dữ liệu phức tạp

Syntax: STRUCT<col\_name : data\_type [COMMENT col\_comment], ...>

## Tạo Database

Create Database là một câu lệnh được sử dụng để tạo cơ sở dữ liệu trong Hive. Database trong Hive là một không gian tên hoặc một tập hợp các bảng. Cú pháp của câu lệnh này như sau:

CREATE DATABASE|SCHEMA [IF NOT EXISTS] <database name>

Truy vấn sau đây được thực thi để tạo cơ sở dữ liệu có tên **userdb**:

hive> CREATE DATABASE [IF NOT EXISTS] userdb;

hoặc

hive> CREATE SCHEMA userdb;

## Xóa Database

Drop Database là một câu lệnh loại bỏ tất cả các bảng và xóa cơ sở dữ liệu. Cú pháp của nó là như sau:

DROP DATABASE StatementDROP (DATABASE|SCHEMA) [IF EXISTS] database\_name

[RESTRICT|CASCADE];

Các truy vấn sau đây được sử dụng để xóa cơ sở dữ liệu. Chúng ta hãy giả sử rằng tên cơ sở dữ liệu là userdb.

hive> DROP DATABASE IF EXISTS userdb;

Các truy vấn sau đây loại bỏ cơ sở dữ liệu bằng cách sử dụng CASCADE. Nó có nghĩa là bỏ các bảng tương ứng trước khi bỏ cơ sở dữ liệu.

hive> DROP DATABASE IF EXISTS userdb CASCADE;

Các truy vấn sau đây loại bỏ cơ sở dữ liệu bằng cách sử dụng SCHEMA.

hive> DROP SCHEMA userdb;

## Tạo Table

1. Create Table là một câu lệnh dùng để tạo bảng trong Hive. Cú pháp và ví dụ như sau:

CREATE [TEMPORARY] [EXTERNAL] TABLE [IF NOT EXISTS] [db\_name.] table\_name

[(col\_name data\_type [COMMENT col\_comment], ...)]

[COMMENT table\_comment]

[ROW FORMAT row\_format]

[STORED AS file\_format]

Nếu bạn thêm option IF NOT EXISTS Hive sẽ bỏ qua câu lệnh tạo trong trường hợp table đã tồn tại.

1. Load dữ liệu: Trong Hive, chúng ta có thể chèn dữ liệu bằng cách sử dụng câu lệnh LOAD DATA.Trong khi chèn dữ liệu vào Hive, tốt hơn là sử dụng LOAD DATA để lưu trữ các bản ghi hàng loạt. Có hai cách để tải dữ liệu: một là từ hệ thống tệp cục bộ và thứ hai là từ hệ thống tệp Hadoop.

Cú pháp như sau:

LOAD DATA [LOCAL] INPATH 'filepath' [OVERWRITE] INTO TABLE tablename

[PARTITION (partcol1=val1, partcol2=val2 ...)]

trong đó:

* LOCAL là định danh để chỉ định đường dẫn cục bộ. Nó là optional.
* OVERWRITE là optional để overwrite dữ liệu trong bảng.
* PARTITION là optional .

## Alter Table

1. Nó được sử dụng để thay đổi một bảng trong Hive.  
   Cú pháp:

ALTER TABLE name RENAME TO new\_name

ALTER TABLE name ADD COLUMNS (col\_spec[, col\_spec ...])

ALTER TABLE name DROP [COLUMN] column\_name

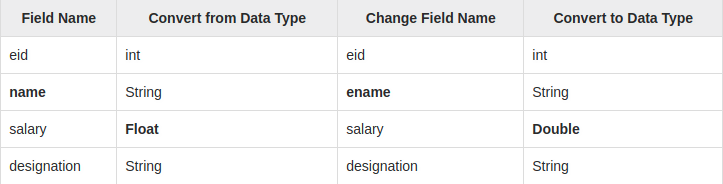
ALTER TABLE name CHANGE column\_name new\_name new\_type

ALTER TABLE name REPLACE COLUMNS (col\_spec[, col\_spec ...])

1. Câu lệnh rename:

Truy vấn sau đây đổi tên bảng từ employee thành emp

hive> ALTER TABLE employee RENAME TO emp;

1. Câu lệnh thay đổi:  
   Bảng sau bao gồm các trường của table employee, những trường được thay đổi sẽ được in đậm  
   

Hai câu truy vấn sau sẽ thực hiện sự thay đổi trên

hive> ALTER TABLE employee CHANGE name ename String;

hive> ALTER TABLE employee CHANGE salary salary Double;

1. Câu lệnh thêm cột  
   Câu truy vấn sau thêm cột dept vào bảng employee  
   hive> ALTER TABLE employee ADD COLUMNS (

dept STRING COMMENT 'Department name');

1. Câu lệnh thay thế

Câu truy vấn sau sẽ xóa hết các cột trong table employee và thay thế chúng với cột emp và name:  
hive> ALTER TABLE employee REPLACE COLUMNS (

eid INT empid Int,

ename STRING name String);

1. Xóa Table  
   Cú pháp:  
   DROP TABLE [IF EXISTS] table\_name;

## Drop Table Cú pháp như sau: DROP TABLE [IF EXISTS] table\_name;

Truy vấn sau đây đưa ra một bảng có tên là **employee**:  
hive> DROP TABLE IF EXISTS employee;

## Phân vùng (Partitioning):

a. Thêm Phân vùng  
Chúng ta có thể thêm phân vùng vào bảng bằng cách thay đổi bảng. Giả sử chúng ta có một bảng được gọi là employee với các trường như Id, Name, Salary, Designation, Dept, and yoj.  
Cú pháp:  
 ALTER TABLE table\_name ADD [IF NOT EXISTS] PARTITION partition\_spec

[LOCATION 'location1'] partition\_spec [LOCATION 'location2'] ...;

partition\_spec:

: (p\_column = p\_col\_value, p\_column = p\_col\_value, ...)

Truy vấn sau được sử dụng để thêm phân vùng vào bảng nhân viên.  
hive> ALTER TABLE employee

> ADD PARTITION (year=’2012’)

> location '/2012/part2012';

b. Đổi tên phân vùng  
ALTER TABLE table\_name PARTITION partition\_spec RENAME TO PARTITION partition\_spec;

Truy vấn sau được sử dụng để đổi tên phân vùng:

hive> ALTER TABLE employee PARTITION (year=’1203’)

> RENAME TO PARTITION (Yoj=’1203’);

c. Bỏ phân vùng  
Cú pháp sau được sử dụng để thả một phân vùng:  
ALTER TABLE table\_name DROP [IF EXISTS] PARTITION partition\_spec, PARTITION partition\_spec,...;

Truy vấn sau được sử dụng để thả phân vùng:

hive> ALTER TABLE employee DROP [IF EXISTS]

> PARTITION (year=’1203’);

## Các toán tử tích hợp (Built-in Operators)

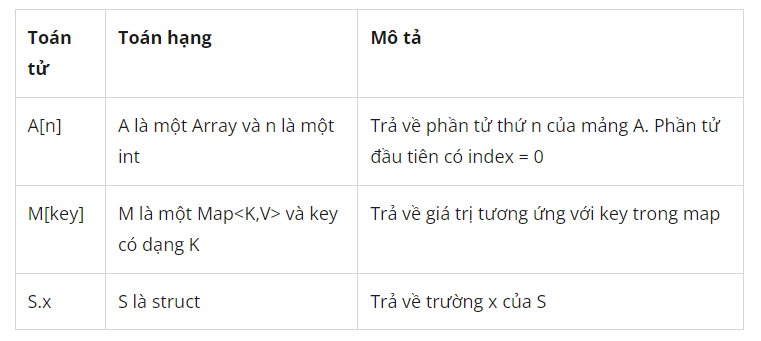
Có bốn loại toán tử trong Hive:

* Toán tử quan hệ
* Toán tử số học
* Toán tử Logic
* Toán tử phức tạp

a. Relational Operators - Toán tử quan hệ  
Các toán tử này được sử dụng để so sánh hai toán hạng. Bảng sau đây mô tả các toán tử quan hệ có sẵn trong Hive:  


b.Arithmetic Operators - Toán tử số học  
Các toán tử này hỗ trợ các phép toán số học phổ biến khác nhau trên các toán hạng. Tất cả đều trả về các loại số. Bảng sau đây mô tả các toán tử số học có sẵn trong Hive:  


c.Logical Operators - Toán tử logic  
Các toán tử là biểu thức logic. Tất cả đều trả về TRUE hoặc FALSE.  
  
  
d.Complex Operators - Toán tử phức hợp  
Các toán tử này cung cấp một biểu thức để truy cập các phần tử của Complex Types.



## Các hàm tích hợp trong Hive (Built-in Functions) a.Built-in Functions Hive hỗ trợ các hàm có tích hợp sẵn sau đây:

|  |  |  |
| --- | --- | --- |
| **Kiểu trả về** | **Cú pháp** | **Mô tả** |
| BIGINT | round(double a) | Trả về giá trị làm tròn BIGINT của double |
| BIGINT | floor(double a) | Trả về giá trị lớn nhất BIGINT bằng hoặc bé hơn double |
| BIGINT | ceil(double a) | Trả về giá trị bé nhất BIGINT bằng hoặc lớn hơn double |
| double | rand(), rand(int seed) | Trả về giá trị ngẫu nhiên thay đổi từ hàng này sang hàng khác |
| string | concat(string A, string B,...) | Trả về kết quả nối chuỗi các string, B sau A |
| string | substr(string A, int start) | Trả về chuỗi con của A bắt đầu từ vị trí bắt đầu cho đến hết chuỗi A. |
| string | substr(string A, int start, int length) | Trả về chuỗi con của A bắt đầu từ vị trí bắt đầu có độ dài length |
| string | upper(string A) | Trả về chuỗi kết quả từ việc chuyển đổi tất cả các ký tự của A thành chữ hoa |
| string | ucase(string A) | Tương tự như trên |
| string | lower(string A) | Trả về chuỗi kết quả từ việc chuyển đổi tất cả các ký tự của A thành chữ thường |
| string | lcase(string A) | Tương tự như trên |
| string | trim(string A) | Trả về chuỗi kết quả từ việc cắt kí tự space từ cả hai đầu của A. |
| string | ltrim(string A) | Trả về chuỗi kết quả từ việc cắt kí tự space từ đầu bên tay trái của A |
| string | rtrim(string A) | Trả về chuỗi kết quả từ việc cắt kí tự space từ đầu bên tay phải của A |
| string | regexp\_replace(string A, string B, string C) | Nó trả về chuỗi kết quả từ việc thay thế tất cả các chuỗi con trong B khớp với cú pháp biểu thức chính quy Java bằng C. |
| int | size(Map<K.V>) | Trả về số lượng elements trong kiểu map |
| int | size(Array<T>) | Trả về số lượng elements trong kiểu mảng |
| giá trị của | cast(<expr> as <type>) | Nó chuyển đổi kết quả của biểu thức expr thành <type> ví dụ: cast ('1' dưới dạng BIGINT) chuyển đổi chuỗi '1' thành biểu diễn tích phân. Một NULL được trả về nếu chuyển đổi không thành công. |
| string | from\_unixtime(int unixtime) | Chuyển đổi số giây trong Unix time (mốc 1970-01-01 00:00:00 UTC) sang một chuỗi đại diện cho thời gian hiện tại của timezone hiện tại với dạng "1970-01-01 00:00:00" |
| string | to\_date(string timestamp) | Trả về phần ngày tháng năm của chuỗi: to\_date("1970-01-01 00:00:00") = "1970-01-01" |
| int | year(string date) | Trả về phần năm của date: year("1970-01-01 00:00:00") = 1970, year("1970-01-01") = 1970 |
| int | month(string date) | Trả về phần tháng của date: month("1970-11-01 00:00:00") = 11, month("1970-11-01") = 11 |
| int | day(string date) | Trả về phần ngày của date: day("1970-11-01 00:00:00") = 1, day("1970-11-01") = 1 |
| string | get\_json\_object (string json\_string, string path) | Trích xuất đối tượng json từ chuỗi json dựa trên đường dẫn json được chỉ định và trả về chuỗi json của đối tượng json được trích xuất. Nó trả về NULL nếu chuỗi json đầu vào không hợp lệ. |

# 

# 

b.Aggregate Functions  
Hive hỗ trợ các hàm aggregate có sẵn sau. Việc sử dụng các hàm này cũng tương tự như các làm aggregate trong SQL  


# 

## View và Indexes

## Bài viết này mô tả cách tạo và quản lý view. View được tạo dựa trên yêu cầu của người dùng. Bạn có thể lưu bất kỳ dữ liệu tập kết quả nào dưới dạng view. Cách sử dụng view trong Hive giống như trong SQL. Đây là một khái niệm RDBMS tiêu chuẩn. Chúng ta có thể thực hiện tất cả các hoạt động DML trên một khung nhìn.

a.Tạo view  
Bạn có thể tạo một view tại thời điểm thực thi một câu lệnh SELECT. Cú pháp như sau:  
CREATE VIEW [IF NOT EXISTS] view\_name [(column\_name [COMMENT column\_comment], ...) ]

[COMMENT table\_comment]

AS SELECT …  
  
b.Xóa view  
Sử dụng cú pháp sau để xóa một view:  
DROP VIEW view\_name

Câu truy vấn sau đây sẽ xóa một view tên là emp\_30000:  
hive> DROP VIEW emp\_30000;  
  
c.Tạo Index  
Một index (chỉ mục) không là gì ngoài một con trỏ trên một cột cụ thể của bảng. Tạo một chỉ mục có nghĩa là tạo một con trỏ trên một cột cụ thể của bảng. Cú pháp của nó như sau:  
 CREATE INDEX index\_name

ON TABLE base\_table\_name (col\_name, ...)

AS 'index.handler.class.name'

[WITH DEFERRED REBUILD]

[IDXPROPERTIES (property\_name=property\_value, ...)]

[IN TABLE index\_table\_name]

[PARTITIONED BY (col\_name, ...)]

[

[ ROW FORMAT ...] STORED AS ...

| STORED BY ...

]

[LOCATION hdfs\_path]

[TBLPROPERTIES (...)]

d.Xóa Index  
Cú pháp sau đây được sử dụng để xóa index có tên index\_salary:  
 DROP INDEX <index\_name> ON <table\_name>  
Câu truy vấn sau xóa một index có tên index\_salary:  
 hive> DROP INDEX index\_salary ON employee;

## HIVE QL

### 11.1 Select-Where Ngôn ngữ truy vấn Hive - Hive Query Language (HiveQL) - là một ngôn ngữ truy vấn dành cho Hive để tiến hành và phân tích dữ liệu có cấu trúc trong một Metastore. Phần này sẽ giải thích cách để sử dụng câu lệnh SELECT với mệnh đề WHERE. Câu lệnh SELECT thường được dùng để lấy dữ liệu từ một bảng. Mệnh đề WHERE làm việc tương tự một điều kiện. Nó lọc dữ liệu sử dụng điều kiện và trả lại cho bạn một kết quả có giới hạn. Toán tử và hàm có sẵn tạo ra một biểu thức đáp ứng điều kiện.

### Syntax

Cho bên dưới đây là cú pháp của câu truy vấn SELECT:

SELECT [ALL | DISTINCT] select\_expr, select\_expr, ...

FROM table\_reference

[WHERE where\_condition]

[GROUP BY col\_list]

[HAVING having\_condition]

[CLUSTER BY col\_list | [DISTRIBUTE BY col\_list] [SORT BY col\_list]]   
[LIMIT number];

### 11.2 Select Order By ORDER BY được sử dụng để lấy chi tiết dựa trên một cột và sắp xếp kết quả với thứ tự tăng dần hoặc giảm dần.

Syntax  
 Dưới đây là cú pháp của ORDER BY:

SELECT [ALL | DISTINCT] select\_expr, select\_expr, ...

FROM table\_reference

[WHERE where\_condition]

[GROUP BY col\_list]

[HAVING having\_condition]

[ORDER BY col\_list]]

[LIMIT number];

### 11.3 Select Group By Mệnh đề GROUP BY được sử dụng để nhóm tất cả các bản ghi trong một tập hợp kết quả bằng cách sử dụng một cột tập hợp cụ thể. Nó được sử dụng để truy vấn một nhóm bản ghi. Cú pháp của mệnh đề GROUP BY như sau: SELECT [ALL | DISTINCT] select\_expr, select\_expr, ...

FROM table\_reference

[WHERE where\_condition]

[GROUP BY col\_list]

[HAVING having\_condition]

[ORDER BY col\_list]]

[LIMIT number];

### 11.4 Select-Joins JOIN là một mệnh đề được sử dụng để kết hợp các trường cụ thể từ hai bảng bằng cách sử dụng các giá trị chung cho mỗi bảng. Nó được sử dụng để kết hợp các bản ghi từ hai hoặc nhiều bảng trong cơ sở dữ liệu. Cú pháp: join\_table:

table\_reference JOIN table\_factor [join\_condition]

| table\_reference {LEFT|RIGHT|FULL} [OUTER] JOIN table\_reference

join\_condition

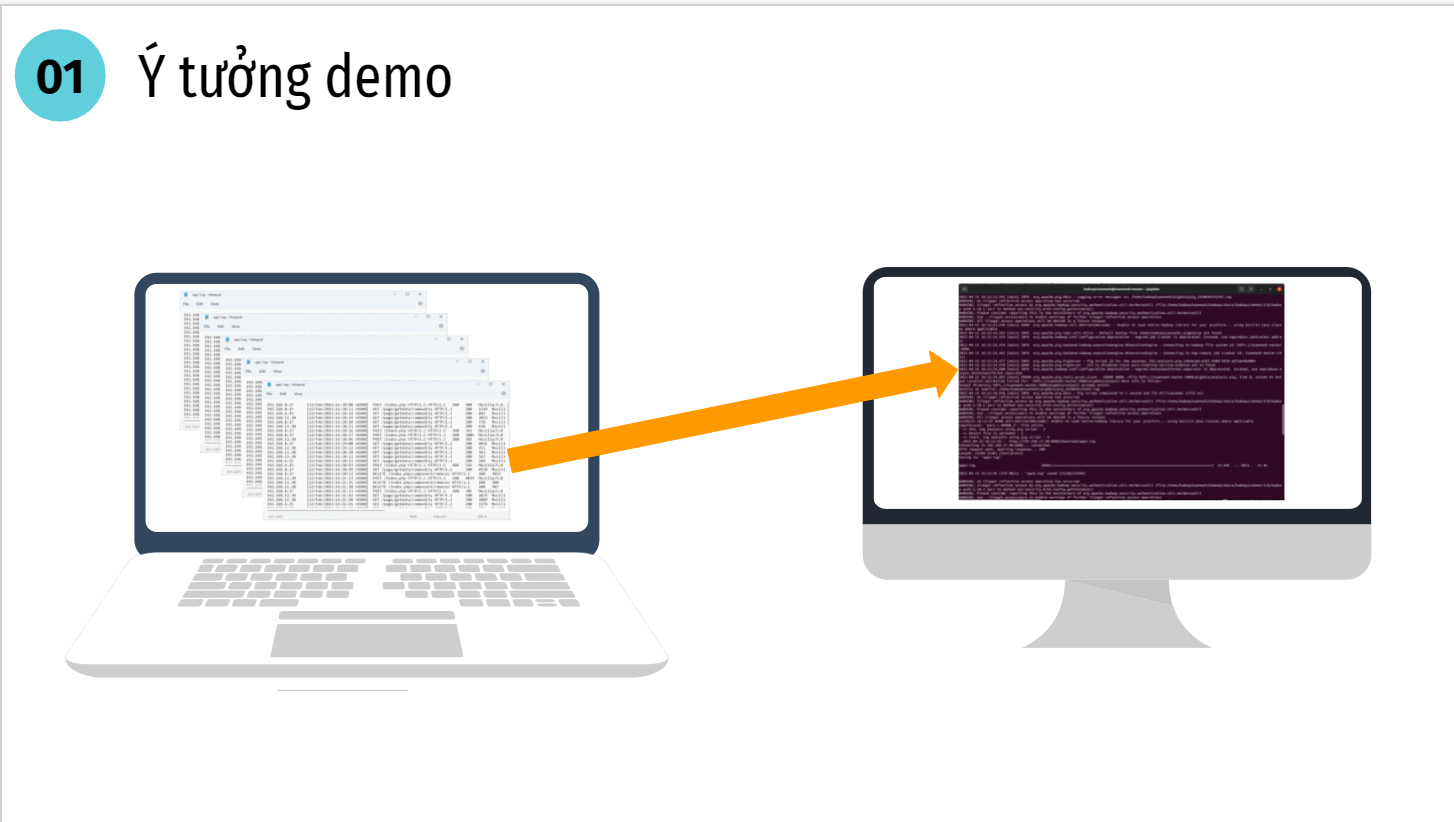
| table\_reference LEFT SEMI JOIN table\_reference join\_condition

| table\_reference CROSS JOIN table\_reference [join\_condition]

# IV - Demo

Mục tiêu của demo này là tạo một dự án Apache Hive cho tập dữ liệu Doanh số bán hàng, trong đó tập dữ liệu Doanh số bán hàng được lấy từ hệ thống khách hàng và sau đó xử lý nó trong hệ thống Hive. Sau khi xử lý dữ liệu từ hệ thống Hive, đẩy dữ liệu đến hệ thống khách hàng và hiển thị dữ liệu ở định dạng cột.

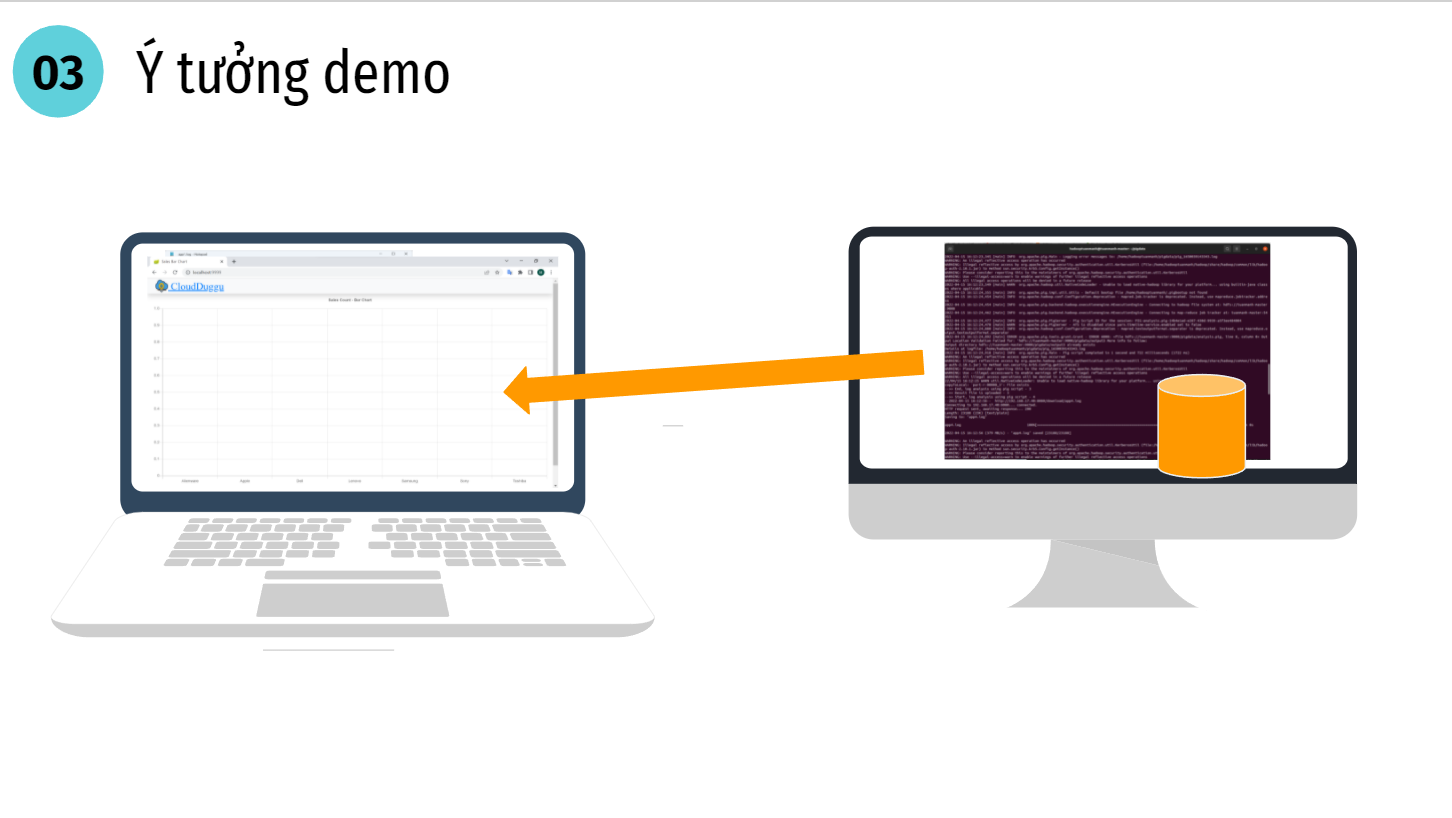
## Ý tưởng demo



- Tìm nạp dữ liệu thô từ hệ thống client.



* Xử lý dữ liệu thô thông qua Hệ thống Hive.



* Đẩy dữ liệu kết quả từ Hệ thống Hive sang Client.

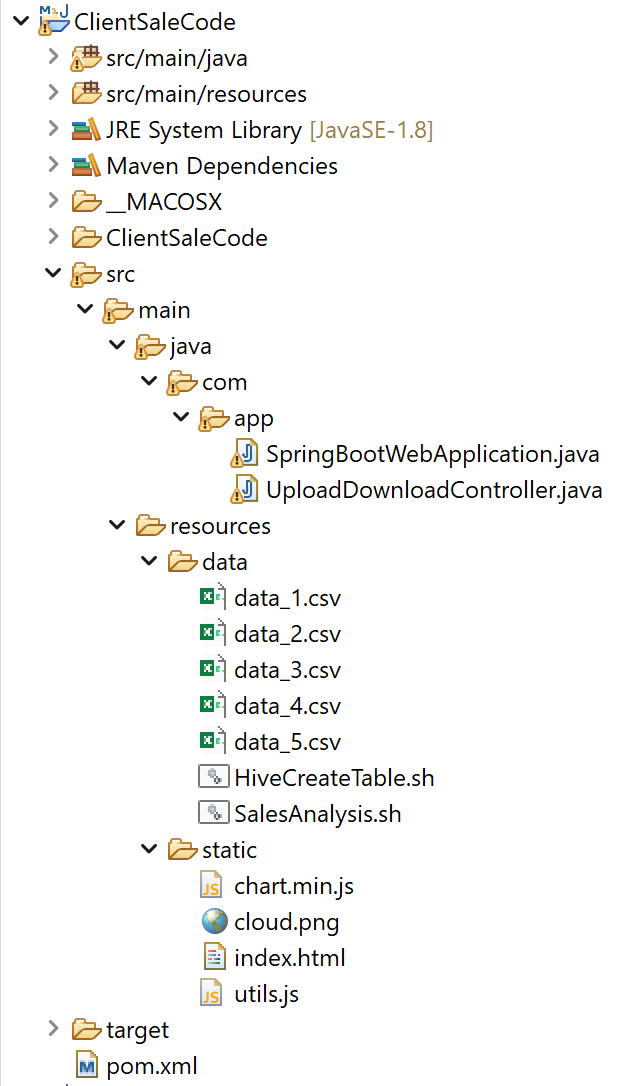


* Hiển thị dữ liệu kết quả trong Client thông qua biểu đồ cột.

## Build Demo

Để chạy demo này, đầu tiên cần cài đặt VM (Máy ảo) trên hệ thống cục bộ của mình và định cấu hình Hive, Derby, Hadoop trên đó. Sau cấu hình này, hệ thống cục bộ của bạn sẽ hoạt động như một hệ thống khách và Hive VM sẽ hoạt động như một hệ thống Hive. Ngoài ra, bạn có thể sử dụng hai hệ thống đang giao tiếp với nhau và trên một trong những hệ thống được cấu hình Hive.

1. Client systems



* Sử dụng Springboot Java Framework, khi build app này sẽ tạo ra file **client.jar.**
* Trong đó có code, file dữ liệu, trang web html.
* Java code có hai file *SpringBootWebApplication.java* and *UploadDownloadController.java*
* SpringBootWebApplication.java là tệp chính, chịu trách nhiệm xây dựng mã và chạy nó trên một máy chủ nhúng.
* UploadDownloadController.java được sử dụng để cung cấp các dịch vụ HTTP URL download và upload. Để tải xuống các tệp dữ liệu, nó sử dụng URL máy khách tải xuống và để tải lên các tệp kết quả.
* Thư mục data chứa các file CSV (data\_1.csv, data\_2.csv, data\_3.csv ...)
* Thư mục static có mã trang HTML biểu đồ (index.html) và các tệp js phụ thuộc (chart.min.js, utils.js). Đây là trang chính hiển thị dữ liệu kết quả quá trình Hive.
* pom.xml là một tập tin công cụ xây dựng dự án. Tệp này có phụ thuộc java và chi tiết cấu hình xây dựng.
* Để tạo file “client.jar” chạy lệnh “ mvn clean package”

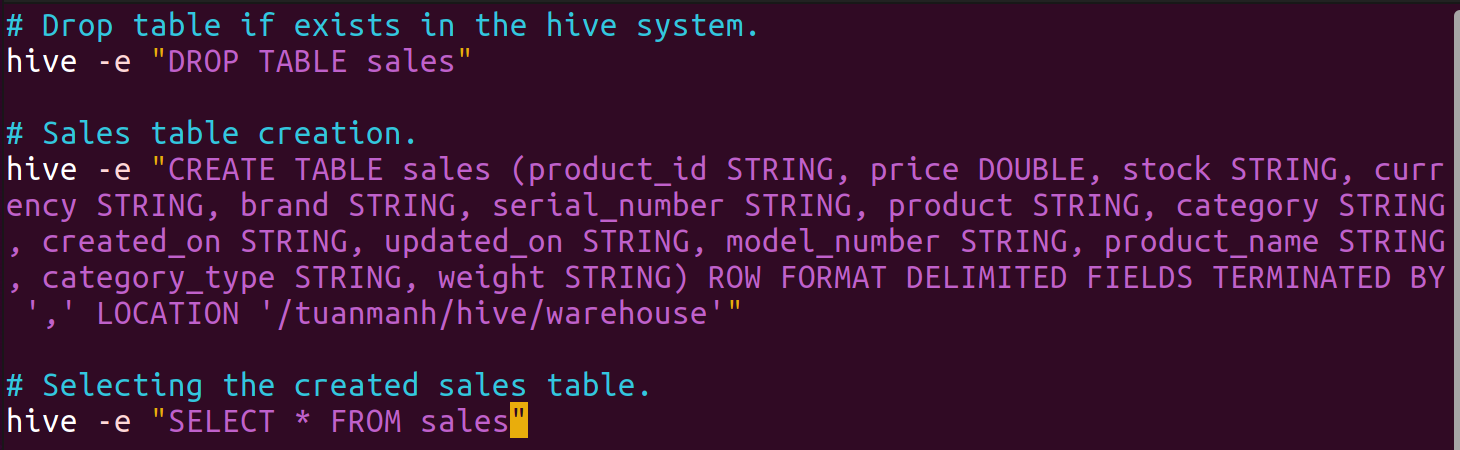
1. Hive systems

Tải xuống tệp tập lệnh shell HiveCreateTable.sh & SalesAnalysis.sh.

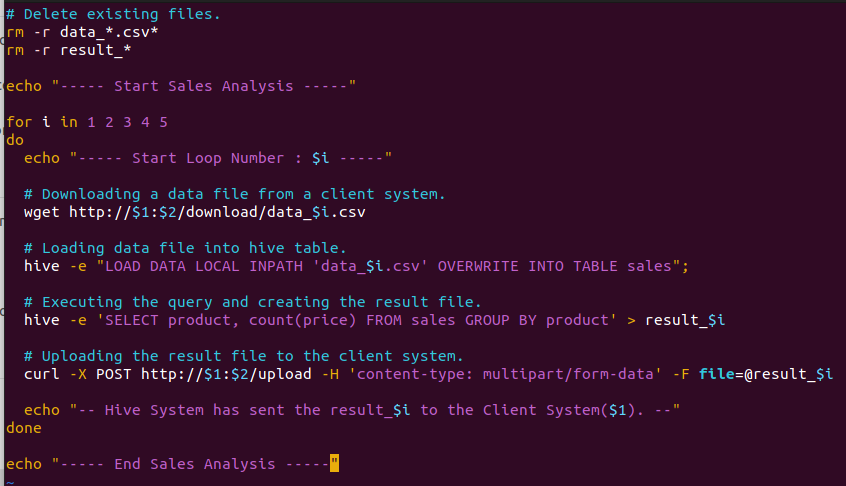
HiveCreateTable.sh, tệp kịch bản shell này có các truy vấn hive để tạo bảng sales.

SalesAnalysis.sh, tệp kịch bản shell này có một tập hợp các mã hive để tải và chọn dữ liệu kết quả từ bảng.

* file HiveCreateTable.sh :

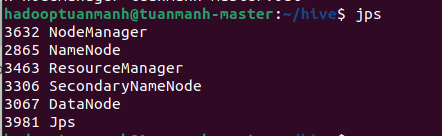


* file SalesAnalysis.sh:



## Run Demo

* Yêu cầu trước khi run demo:
* Đảm bảo start-all trước

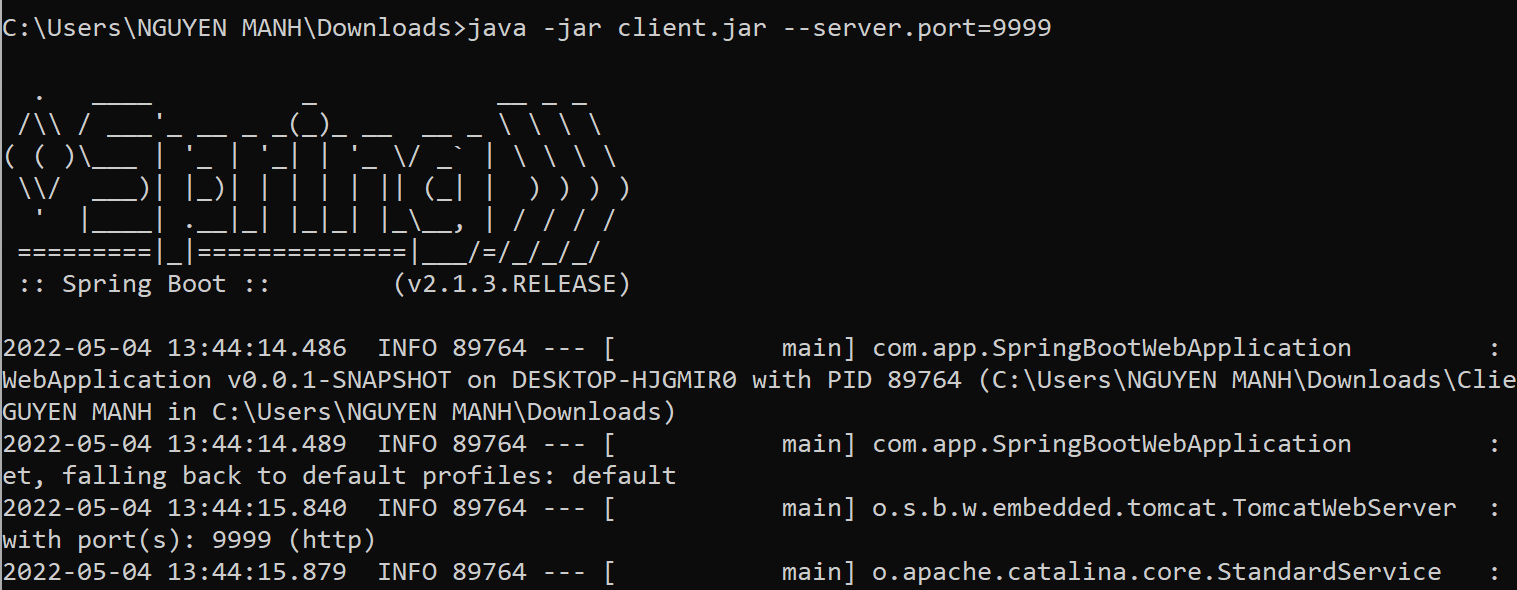


* Đảm bảo Apache Hive chạy bình thường.

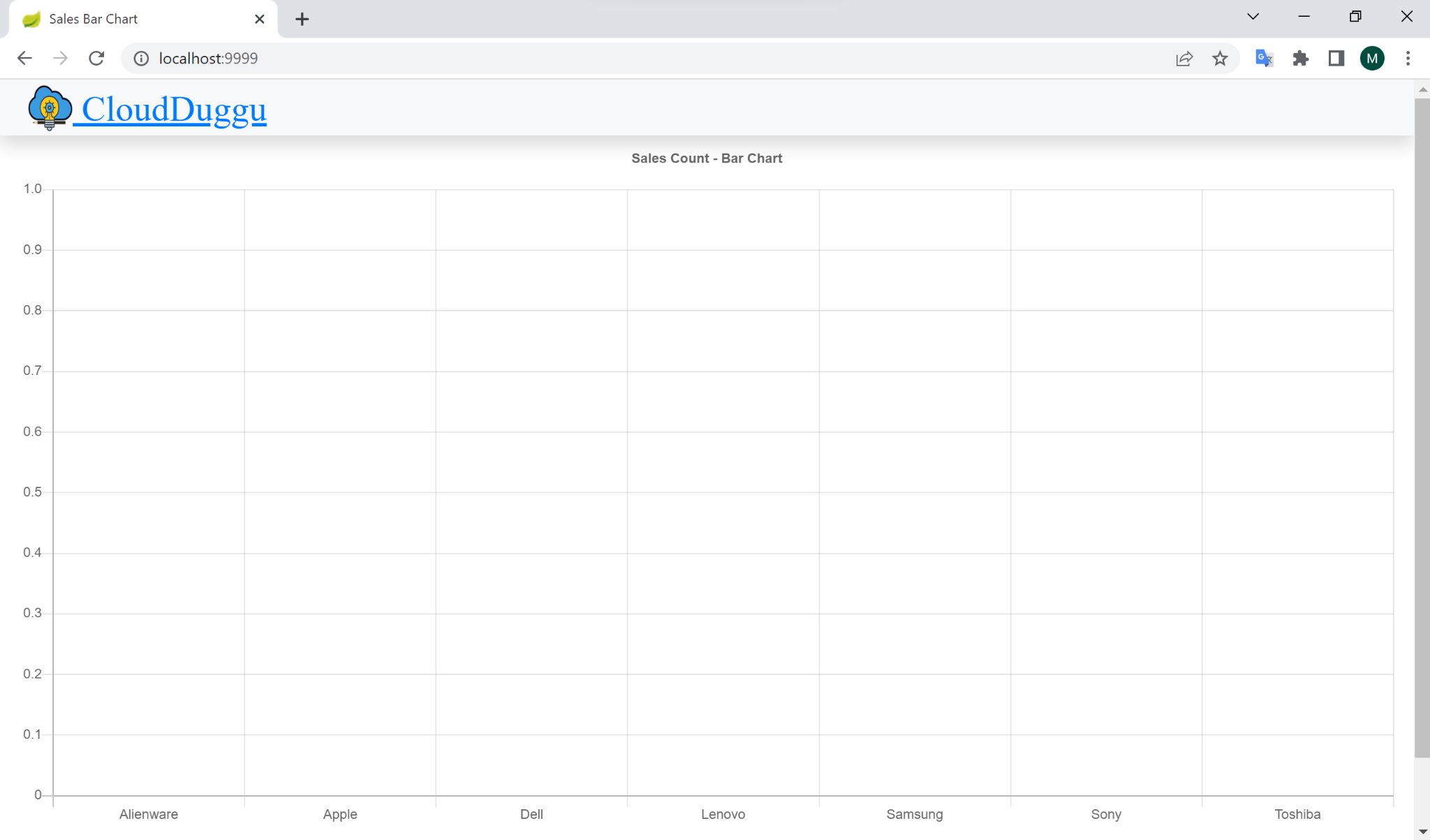
### Bước 1: Run file client.jar ở máy client.

* Dùng lệnh java -jar client.jar --server.port=9999

với port 9999.

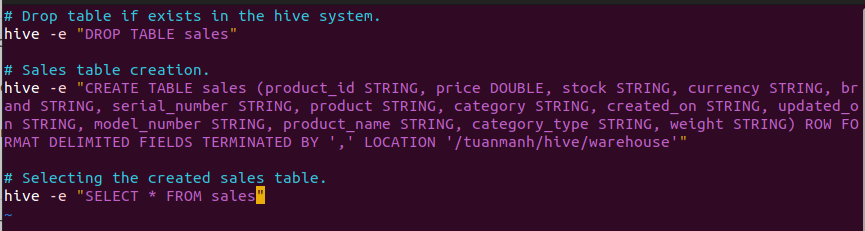


* Kiểm tra địa chỉ [http://localhost:9999](http://localhost:9999/) trên trình duyệt.



### Bước 2: Thực thi tập lệnh shell HiveCreateTable.sh trong Hive System

* Nội dung trong file HiveCreateTable.sh:

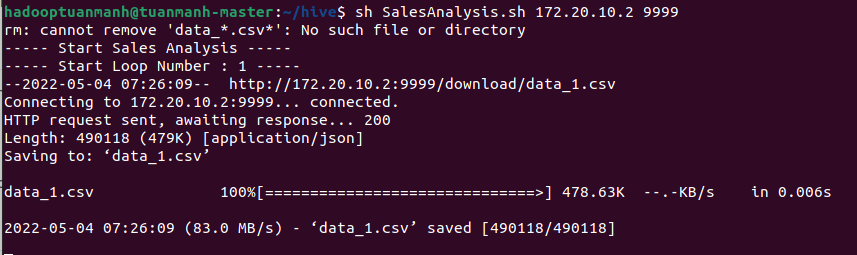


* Đầu tiên nó sẽ xóa bảng cũ đi, nếu như không có sẽ bỏ qua.
* Tiếp theo là truy vấn tạo bảng Sales bằng HiveQL.
* Cuối cùng là dùng lệnh SELECT để truy vấn từ bảng *sales* vừa tạo.

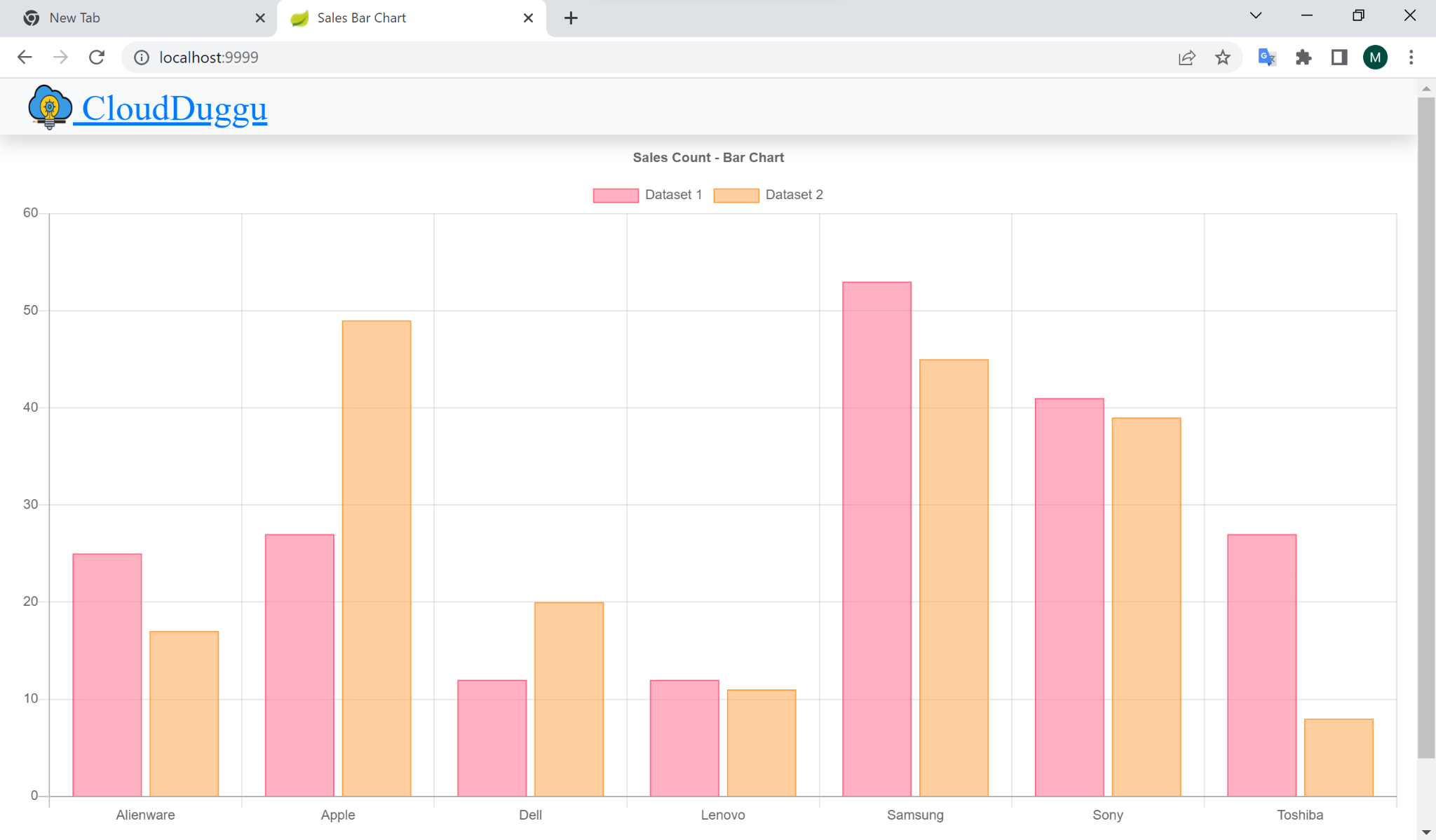
### Bước 3: Thực thi tập lệnh shell SalesAnalysis.sh trong Hive System

* Sau khi thực thi thành công tập lệnh tạo bảng, thực hiện tệp tập lệnh tiếp theo SalesAnalysis.sh. Tại thời điểm chạy tập lệnh, truyền vào địa chỉ ip máy client và số port máy client bằng lệnh:

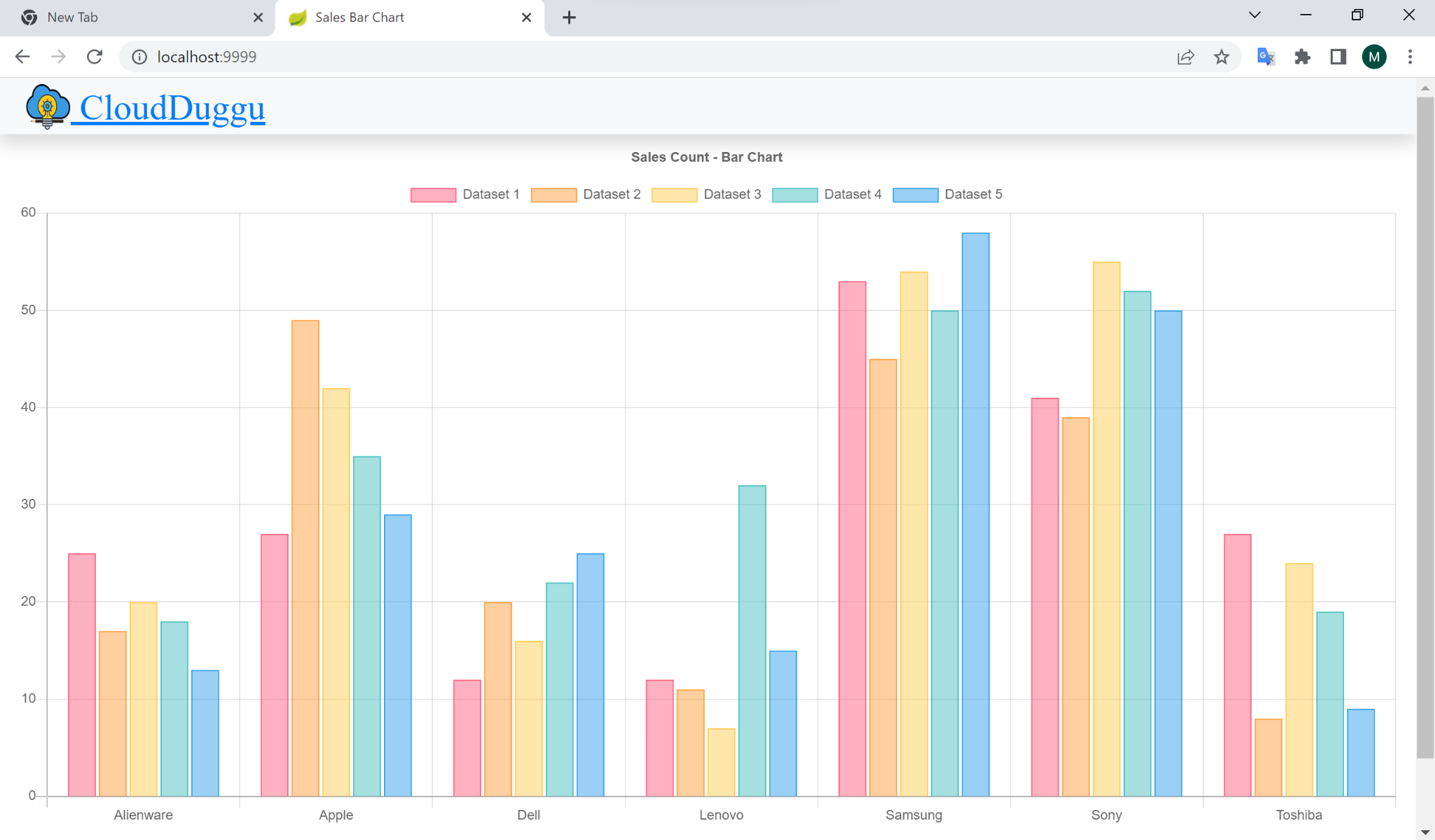
sh SalesAnalysis.sh 192.168.17.40 9999



* Sau khi run, các file dữ liệu từ client sẽ được tải về Hive system theo thứ tự. Sau khi tải xong mỗi file, Hive sẽ xử lý và upload trở lại máy *client*.



* Mỗi cột đại diện cho một dataset đã được Hive xử lý.
* Sau khi xử lý xong, biểu đồ hiển thị trên máy client sẽ như hình dưới:



Tham khảo

<https://www.cloudduggu.com/hive/project-01/>

<https://www.tutorialspoint.com/hive/hive_create_database.htm>