**TRƯỜNG ĐH SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH**

**🕯✡🕮🕮✡🕯**

**MÔN HỌC: Big data Analysis**

**APACHE HIVE**

**GVHD: Cô Lê Thị Minh Châu**

**Nhóm SV thực hiện: Nhóm 12**

Vũ Đặng Quỳnh Giang 17133015

Nguyễn Thị Ngọc 17133041

Lê Kha 17133030

**TP. Hồ Chí Minh, tháng 7 năm 2020**

MỤC LỤC

[**I - Tìm hiểu chung về Apache Hive: 3**](#_Toc47559381)

[**1. Sơ lược về Hadoop: 3**](#_Toc47559382)

[**2. Apache Hive 4**](#_Toc47559383)

[**3. Cài đặt trên hđh Ubuntu Server hoặc các nền tảng khác. 4**](#_Toc47559384)

[**4. Cách làm việc của Hive 11**](#_Toc47559385)

[**II – Mở rộng phân tích dữ liệu lớn với Apache Hive 24**](#_Toc47559386)

[**1. Các nhược điểm của Hive 24**](#_Toc47559387)

[**2. Các phương pháp phân tích 27**](#_Toc47559388)

[**3. Các giai đoạn phân tích big data 28**](#_Toc47559389)

[**4. Các xu hướng dữ liệu lớn hiện nay tích hợp Apache Hive 29**](#_Toc47559390)

[**III – DEMO 31**](#_Toc47559391)

[**1. Giới thiệu tập dữ liệu 31**](#_Toc47559392)

[**2. Cơ sở dữ liệu 31**](#_Toc47559393)

[**3. Chuẩn bị dữ liệu 31**](#_Toc47559394)

[**1) Làm sạch dữ liệu 31**](#_Toc47559395)

[**2) Tạo bảng 33**](#_Toc47559396)

[**3) Configure Hive 35**](#_Toc47559397)

[**4. Các câu truy vấn Hive 38**](#_Toc47559398)

[**IV - Tài liệu tham khảo 50**](#_Toc47559407)

# I - Tìm hiểu chung về Apache Hive:

## Sơ lược về Hadoop:

Thuật ngữ Big Data được sử dụng cho các bộ tập dữ liệu khổng lồ bao gồm khối lượng lớn, tốc độ cao và nhiều loại dữ liệu đang tăng lên từng ngày. Sử dụng các hệ thống quản lý dữ liệu truyền thống, rất khó để xử lý Big data. Do đó, Quỹ phần mềm Apache (Apache Software Foundation) đã giới thiệu một framework tên là Hadoop để giải quyết các thách thức quản lý và xử lý Big data.

Hadoop là một framework open-source để lưu trữ và xử lý Big data trong môi trường phân tán. Nó chứa hai mô-đun, một là MapReduce và một mô-đun khác là Hệ thống tệp phân tán Hadoop (Hadoop Distributed File System - HDFS).

* **MapReduce**: Đây là mô hình lập trình song song để xử lý một lượng lớn dữ liệu có cấu trúc, bán cấu trúc và không cấu trúc trên các cụm lớn của phần cứng thương mại (commodity hardware).
* **HDFS**: Hệ thống tệp phân tán Hadoop là một phần của framework Hadoop, được sử dụng để lưu trữ và xử lý các bộ dữ liệu. Nó cung cấp một hệ thống tập tin chịu lỗi để chạy trên phần cứng thương mại.

Hệ sinh thái Hadoop chứa các sub-project (tool) khác nhau như Sqoop, Pig và Hive được sử dụng để trợ giúp các mô-đun Hadoop.

* **Sqoop**: Nó được sử dụng để nhập và xuất dữ liệu đến và đi giữa HDFS và RDBMS.
* **Pig**: Đây là một nền tảng ngôn ngữ thủ tục được sử dụng để phát triển tập lệnh cho các hoạt động của MapReduce.
* **Hive**: Đây là một nền tảng được sử dụng để phát triển các tập lệnh loại SQL để thực hiện các hoạt động MapReduce.

**Chú ý**: Có nhiều cách khác nhau để thực hiện các hoạt động MapReduce:

* Cách tiếp cận truyền thống sử dụng chương trình Java MapReduce cho dữ liệu có cấu trúc, bán cấu trúc và không cấu trúc.
* Cách tiếp cận dùng câu lệnh cho MapReduce để xử lý dữ liệu có cấu trúc và bán cấu trúc bằng Pig.
* Ngôn ngữ truy vấn Hive (HiveQL hoặc HQL) cho MapReduce để xử lý dữ liệu có cấu trúc bằng Hive.

## Apache 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 đỉnh Hadoop để tóm tắt Dữ liệu lớn và giúp truy vấn và phân tích dễ dàng.

Ban đầu Hive được phát triển bởi Facebook, sau đó Quỹ Phần mềm Apache đã lấy và phát triển nó thành một nguồn mở dưới tên Apache Hive. Nó được sử dụng bởi các công ty khác nhau. Ví dụ: Amazon sử dụng nó trong Amazon Elastic MapReduce.

Hive không phải là:

* + Một CSDL quan hệ
  + Một thiết kế để xử lý giao dịch Online (OnLine Transaction Processing - OLTP)
  + Một ngôn ngữ cho các truy vấn thời gian thực và cập nhật cấp hàng
* **Đặc trưng của Hive**
  + Nó lưu trữ lược đồ trong cơ sở dữ liệu và xử lý dữ liệu vào HDFS.
  + Nó được thiết kế cho OLAP.
  + Nó cung cấp ngôn ngữ kiểu SQL để truy vấn được gọi là HiveQL hoặc HQL.
  + Nó là quen thuộc, nhanh chóng, có khả năng mở rộng.

## 3. Cài đặt trên hđh Ubuntu Server hoặc các nền tảng khác.

* **Chạy Apache Hive trên** Windows

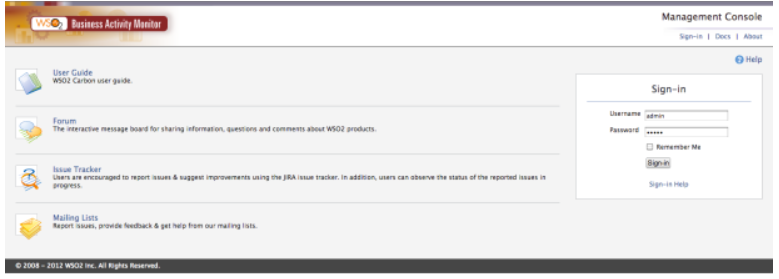
**Lưu ý** : Bạn cần cài đặt [cygwin](http://www.cygwin.com/) để chạy hướng dẫn này, vì Hadoop (cần bởi Hive) cần cygwin để chạy trên windows. Tối thiểu, các gói Cơ bản, Mạng (Các gói OpenSSH, tcp\_wrapper) và Bảo mật liên quan đến Bảo mật cần phải có trong hệ thống.

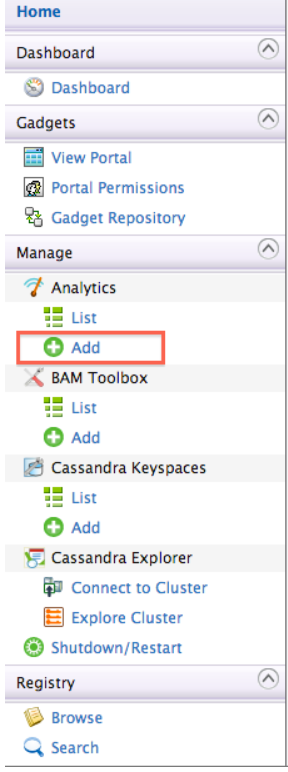
**Bước 1:** Tải xuống [WSO2 BAM 2.0.0](http://wso2.com/products/business-activity-monitor/) . Nó là nguồn mở và miễn phí.

**Bước 2:** Trích xuất nó đến một vị trí ưa thích. Hãy gọi nó là $ BAM\_HOME.

**Bước 3:** Khởi động máy chủ bằng cách thực thi tệp wso2server.bat có trong $ BAM\_HOME / bin. Máy chủ sẽ khởi động trên cổng mặc định 9443 trên IP của máy.

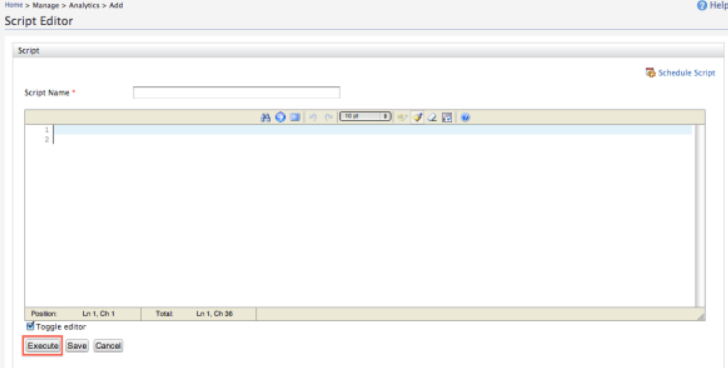
**Bước 4:** Đăng nhập vào bảng điều khiển web tại https: // localhost: 9443 bằng thông tin đăng nhập mặc định, tức là tên người dùng: admin, mật khẩu: admin và nhấp vào Đăng nhập

**Bước 5.** Điều hướng đến tùy chọn Thêm Thêm Analytics Analytics bằng cách nhấp vào mục menu trên menu bên trái.



**Bước 6:** Bây giờ thực thi tập lệnh Hive của bạn, bằng cách nhập tập lệnh và nhấp vào thực thi!

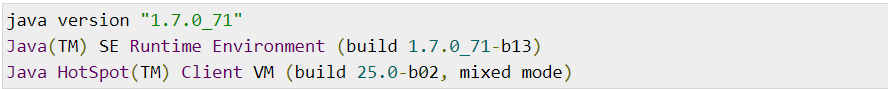
Lưu ý: Thực hiện theo [tài liệu mẫu KPI](http://docs.wso2.org/wiki/display/BAM200/KPI+Monitoring+Sample) này để xem mẫu làm việc cho bạn ngay lập tức, với kết quả xuất hiện trên bảng điều khiển. Ngoài ra, lưu ý rằng bạn cũng có thể lên lịch cho kịch bản Hive.



Thực thi tập lệnh Apache Hive

* **Chạy Apache Hive trên Linux**

Bước 1: Cài Java



Bước 2: Cài Hadoop



Bước 3: Cài Hive

Tải Hive, khi tải thành công có phản hồi sau:

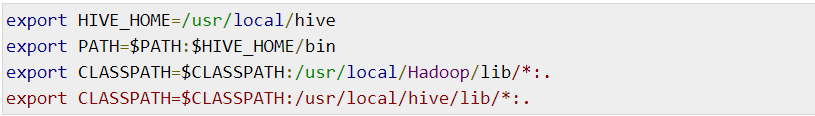




Sao chép tập tin vào thư mục / usr / local / hive



Thiết lập môi trường cho Hive



Cấu hình Hive

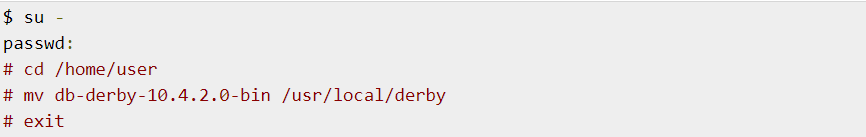
Chỉnh sửa tệp hive-env.sh bằng cách nối thêm dòng sau:

Tải xuống và cài đặt Apache Derby

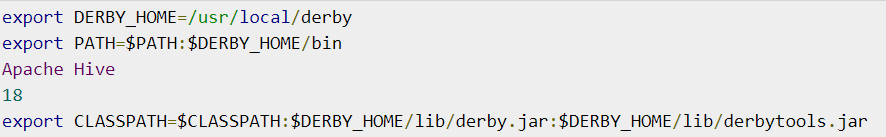
Trích xuất và xác minh kho lưu trữ Derby



Sao chép tập tin vào thư mục / usr / local / derby



Thiết lập môi trường cho Derby



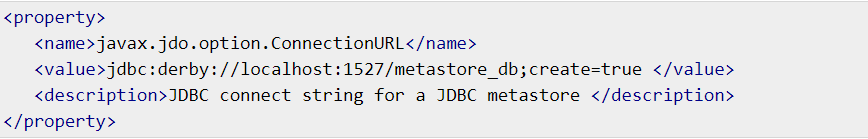
Tạo một thư mục để lưu trữ Metastore



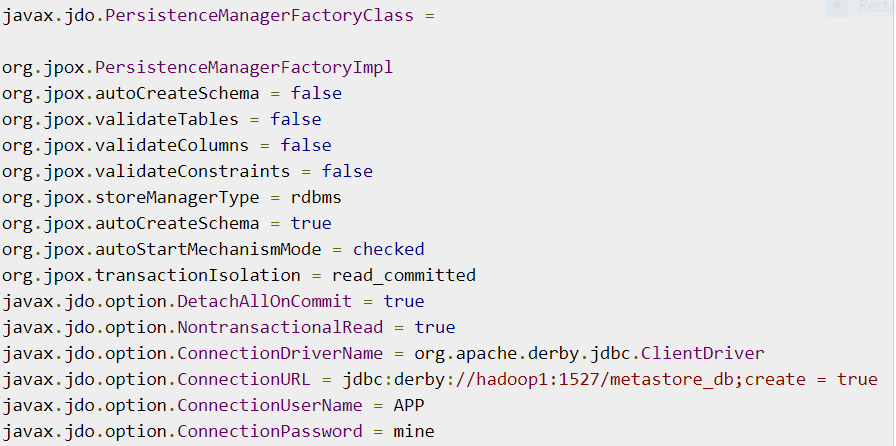
Cấu hình Metastore của Hive



Chỉnh sửa **hive-site.xml** và nối các dòng sau giữa các thẻ <configure> và </ configure>:



Tạo một tệp có tên jpox.properies và thêm các dòng sau vào đó:

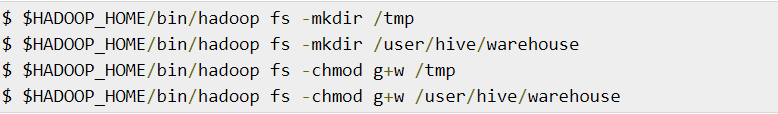


Xác minh cài đặt Hive

Trước khi chạy Hive, bạn cần tạo thư mục **/ tmp** và thư mục Hive riêng trong HDFS. Ở đây, chúng tôi sử dụng thư mục **/ user / hive / kho** . Bạn cần đặt quyền ghi cho các thư mục mới được tạo như dưới đây:



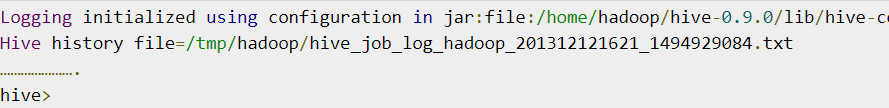
Bây giờ đặt chúng trong HDFS trước khi xác minh Hive. Sử dụng các lệnh sau:



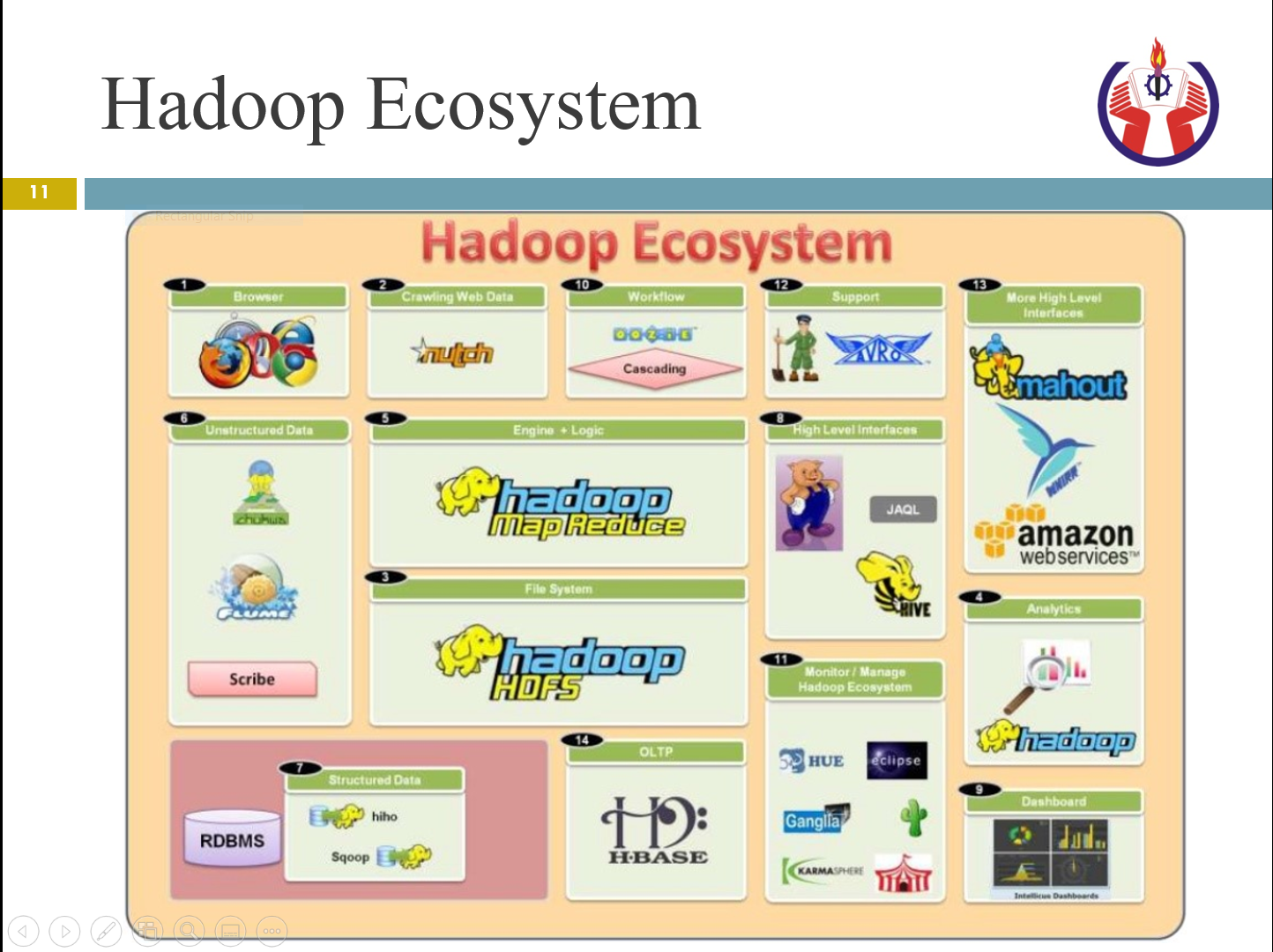
Các lệnh sau được sử dụng để xác minh cài đặt Hive:



Khi cài đặt thành công Hive, bạn có thể thấy phản hồi sau:

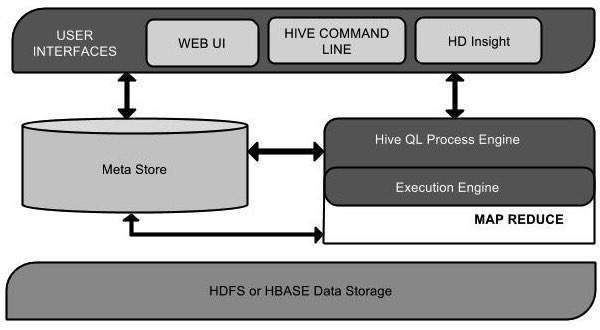


1. **Tìm hiểu lý thuyết về Hive:**
2. **Vị trí, vai trò hệ sinh thái của Hive trong hệ sinh thái Hadoop**



Vai trò: cung cấp sự tổng hợp dữ liệu, truy vấn và phân tích. Nó hỗ trợ phân tích các tập dữ liệu lớn được lưu trong HDFS của Hadoop cũng như trên Amazon S3. Điểm hay của HIVE là hỗ trợ truy xuất giống SQL đến dữ liệu có cấu trúc, được biết với tên HiveSQL (hoặc HQL), được xây dựng cho các ứng dụng khai thác dữ liệu (data mining). Các ứng dụng khai thác dữ liệu có thể mất nhiều phút đến nhiều giờ để phân tích dữ liệu và HIVE được dùng chủ yếu.

1. **Kiến trúc Hive:**

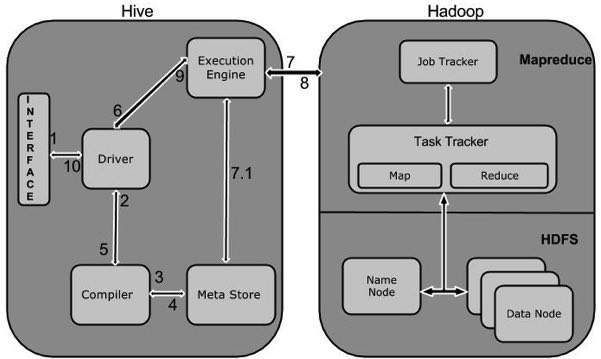
Sơ đồ sau đây mô tả kiến trúc của Hive:

Sơ đồ thành phần này chứa các đơn vị khác nhau.

* 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, Hive command line và Hive HD Insight (Trong máy chủ Windows).
* Meta Store: Hive chọn các máy chủ cơ sở dữ liệu tương ứng để lưu trữ lược đồ hoặc metadata của các bảng, cơ sở dữ liệu, các cột trong một bảng, các loại 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 thay thế của phương pháp 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 công cụ xử lý HiveQL và MapReduce là Công cụ thực thi Hive (Hive Execution Engine). Công cụ thực thi xử lý truy vấn và tạo kết quả giống như kết quả MapReduce.
* HDFS hoặc 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.

## 4. Cách làm việc của Hive

Sơ đồ sau mô tả quy trình làm việc giữa Hive và Hadoop.



Cách Hive tương tác với framework Hadoop:

1. Thực thi query: Giao diện Hive như Command line hoặc Giao diện người dùng web gửi truy vấn đến 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 để 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 như một phản hồi cho 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, việc phân tích cú pháp và biên dịch một truy vấn đã hoàn tất.
6. Kế hoạch thực hiện: Trình điều khiển gửi kế hoạch thực hiện đến công cụ thực thi.
7. Thực thi công việc: Trong nội bộ, quá trình thực thi công việc là một công việc MapReduce. Công cụ thực thi gửi công việc đến JobTracker, trong node Name và nó gán công việc này cho TaskTracker, trong node Data. Ở đây, truy vấn thực thi công việc MapReduce. **Hoạt động metadata**: Trong khi thực hiện, công cụ thực thi có thể thực thi các hoạt động metadata với Metastore.
8. Lấy kết quả: Công cụ thực thi nhận kết quả từ các node Data.
9. Gửi kết quả: Công cụ thực thi gửi các giá trị kết quả đó đến trình điều khiển.
10. Gửi kết quả: Trình điều khiển gửi kết quả đến Giao diện Hive.
11. **Kiểu dữ liệu, cách tạo các đối tượng bảng, view…**

**Primitive Types**

1. **Kiểu Numeric**

**TINYINT**: số nguyên 1 byte, từ -128 đến 127

**SMALLINT**: số nguyên 2 byte, từ -32,768 đến 32,767

**INT**: Số nguyên 4 byte, từ -2,147,483,648 đến 2,147,483,647

**BIGINT**: Số nguyên 8 byte, từ -9,223,372,036,854,775,809 đến 9,223,372,036,854,775,807

**FLOAT**: Độ chính xác đơn 4 byte

**DOUBLE**: Độ chính xác kép kép 8 byte

**DECIMAL**: Hive 0.13.0 giới thiệu độ chính xác và tỷ lệ người dùng có thể xác định

1. **Chuỗi**

**STRING:** chuỗi các ký tự trong một bộ ký tự được chỉ định

**VARCHAR:** chuỗi các ký tự trong một bộ ký tự được chỉ định với độ dài tối đa

**CHAR:** chuỗi ký tự trong một bộ ký tự được chỉ định với độ dài xác định

1. **Date/Time**

**TIMESTAMP:** Ngày và thời gian không có múi giờ

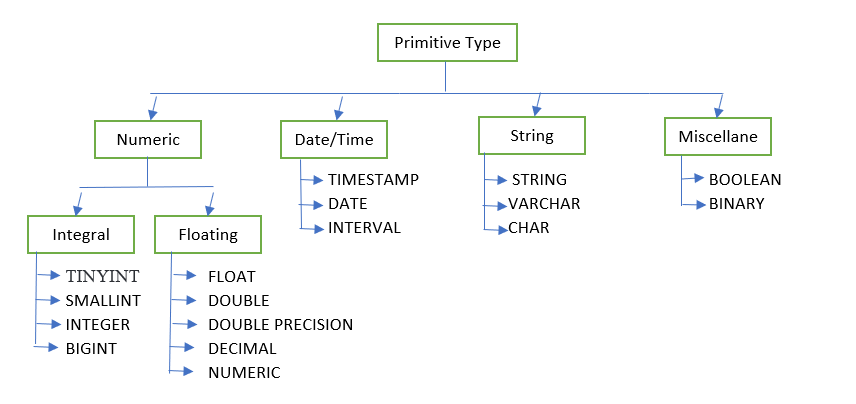
**DATE:** Một ngày

1. **Hỗn hợp (Miscellaneous)**

**BOOLEAN:** True/False

**BINARY:** một chuỗi các byte

Thứ bậc được sắp xếp như sau



**Complex Types**

1. **Array (mảng)**

ARRAY <data\_type>

Một mảng có thể chứa một hoặc nhiều giá trị của cùng loại dữ liệu. Các phần tử có thể được truy cập bằng cách sử dụng ký hiệu [n] trong đó n được chỉ mục (zero-based) vào mảng. Ví dụ: Mảng A có các phần tử ['a', 'b', 'c'], A [1] truy xuất 'b'.

1. **Map (bản đồ)**

MAP<primitive\_type, data\_type>

Bản đồ là tập hợp các cặp khóa-giá trị nơi các trường được truy cập bằng cách sử dụng ký hiệu mảng của các khóa. Ví dụ: trong bản đồ M bao gồm ánh xạ từ 'team' -> giá trị gid có thể được truy cập bằng M ['team']

1. **Structs (cấu trúc)**

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

Struct là một kiểu dữ liệu lưu trữ record, bao gồm một bộ các cột thuộc nhiều kiểu dữ liệu nguyên thủy. Một phần tử trong kiểu STRUCT có thể được truy cập bằng cách sử dụng ký hiệu DOT (.). Ví dụ: đối với một cột c có kiểu STRUCT {a INT; b INT}, một lĩnh vực được truy cập bởi biểu thức c.a

Sử dụng Primitive Types và các kiểu cấu trúc để tạo Complex Types, các kiểu có mức lồng nhau tùy ý có thể được tạo. Ví dụ: một User có thể bao gồm các thông tin sau:

* Gender – string
* Active – boolean

**TẠO CÁC ĐỐI TƯỢNG**

1. **Create Database**

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

1. **Drop Database**

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

[RESTRICT|CASCADE];

1. **Create Table**

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]

1. **Alter Table**

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. **Drop Table**

DROP TABLE [IF EXISTS] table\_name;

1. **Create View**

CREATE VIEW [IF NOT EXISTS] [db\_name.]view\_name [(column\_name [COMMENT column\_comment], ...) ]

1. **Alter View**

ALTER VIEW [db\_name.]view\_name SET TBLPROPERTIES table\_properties;

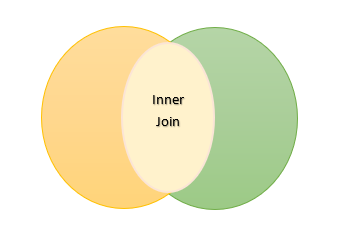
table\_properties:

  : (property\_name = property\_value, property\_name = property\_value, ...)

1. **JOIN Statement**

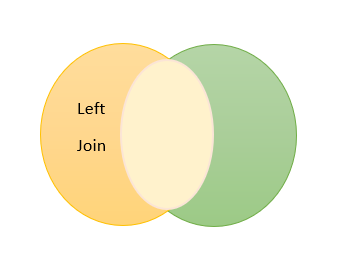
JOIN được sử dụng khi ta có nhu cầu kết hợp 2 bảng riêng biệt lại với nhau. Phổ biến nhất đó là nối 2 bảng lại chứa các cột từ cả 2 bảng.

Types of Joins

SELECT x.coll, … x.coln, y.coll, … y.coln

FROM x

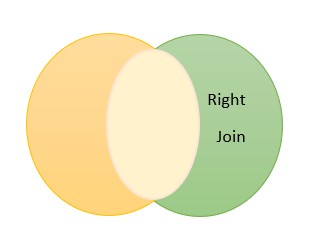
JOIN y ON (x.id = y.id);



SELECT x.coll, … x.coln, y.coll, … y.coln

FROM x

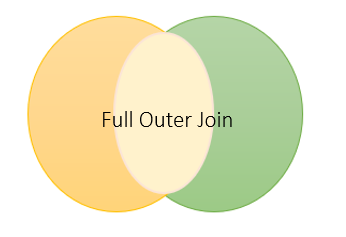
LEFT OUTER JOIN y ON (x.id = y.id);



SELECT x.coll, … x.coln, y.coll, … y.coln

FROM x

RIGHT OUTER JOIN y ON (x.id = y.id);

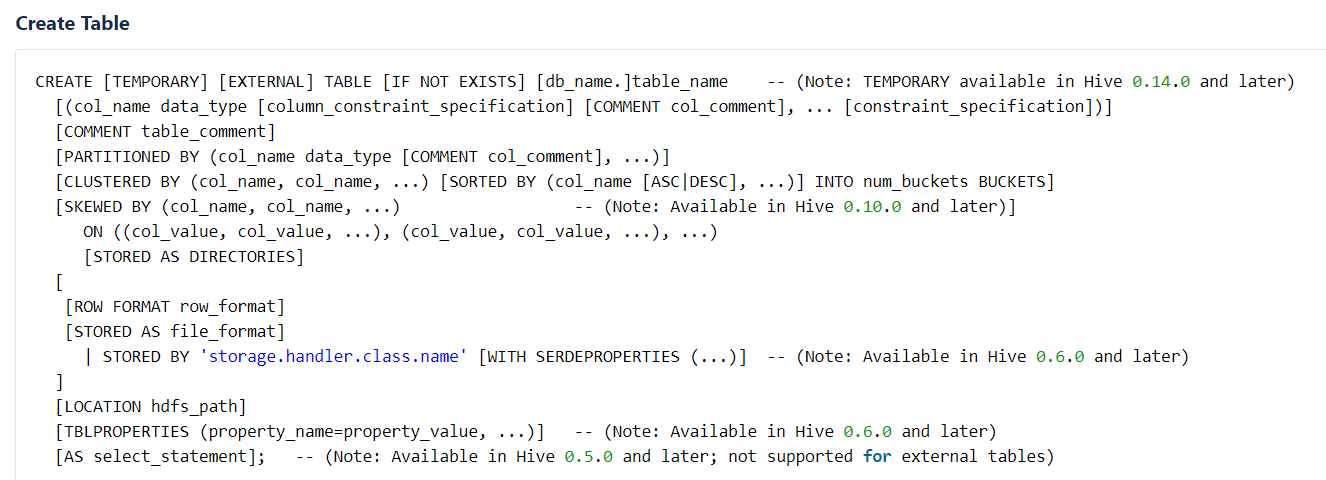


SELECT x.coll, … x.coln, y.coll, … y.coln

FROM x

RIGHT OUTER JOIN y ON (x.id = y.id);

1. **Giải thích thuật ngữ Hive**



* [TEMPORARY TABLE] (bảng tạm thời): là một loại bảng đặc biệt cho phép bạn lưu trữ một tập kết quả tạm thời, có thể sử dụng lại nhiều lần.

Được tạo bằng cách sử dụng câu lệnh CREATE TEMPORARY TABLE

Một bảng tạm thời có thể có cùng tên với bảng gốc. Nếu ta tạo 1 bảng tạm thời khachhang trong CSDL, thì bảng khachhang gốc sẽ không truy cập được. Mọi câu truy vấn đưa ra đối với khachhang sẽ truy cập vào bảng tạm thời. Khi ta xóa mất khachhang bảng tạm thời, thì khachhang bảng gốc sẽ thay bảng tạm thời và truy cập.

Mặc dù vậy bảng tạm thời có thể có cùng tên với một bảng gốc. Tuy nhiên nó không được khuyến khích vì có thể sẽ dẫn đến nhầm lẫn và có khả năng gây mất dữ liệu một cách bất ngời.

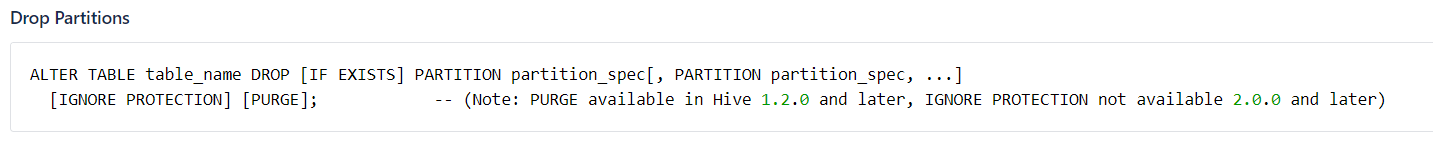
* [EXTERNAL TABLE] (bảng bên ngoài): Có thể được truy cập và quản lý bởi các quy trình bên ngoài Hive. Bảng bên ngoài có thể truy cập dữ liệu được lưu trữ bên trong các Khối lưu trữ Azure (ASV) hoặc các vị trí HDFS từ xa. Sử dụng External Table khi các tệp có sẵn hoặc ở vị trí xa và các tệp vẫn sẽ tồn tại ngay cả khi bảng bị xóa.
* [PARTITIONED BY] (phân vùng): Do Hive chuyển đổi các câu truy vấn SQL thành MapReduce sau đó gửi đến các cụm Hadoop. Khi ta gửi một câu truy vấn SQL, Hive sẽ đọc toàn bộ tập dữ liệu, do đó nó sẽ không hiệu quả khi chạy MapReduce trên bộ dữ liệu lớn. Do đó tạo phân vùng trong bảng sẽ giúp Hive triển khai các phân vùng này rất dễ dàng bằng cách tạo sơ đồ phân vùng tại thời điểm tạo bảng.

**Ưu điểm:**

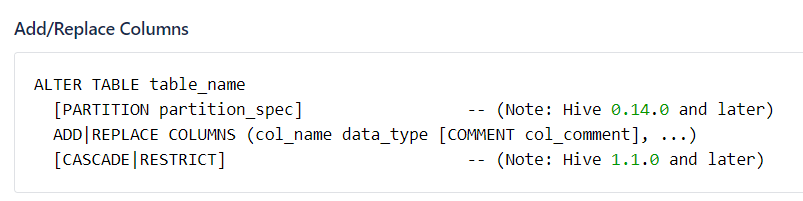
* Phân vùng trong Hive chia các công việc theo chiều ngang, nghĩa là chia đều công việc cho các node trong cluster HDFS
* Thực hiện nhanh hơn các truy vấn có khối lượng dữ liệu thấp. Ví dụ dân số tìm kiếm ở TPHCM sẽ trả về rất nhanh thay vì tìm kiếm toàn bộ dân số thế giới

**Nhược điểm:**

* Nếu phân vùng không đúng cách có thể xảy ra trường hợp tạo ra các phân vùng quá nhỏ, đồng nghĩa với quá nhiều thư mục con.
* Phân vùng hiệu quả với dữ liệu khối lượng thấp, nhưng một số truy vấn như nhóm theo khối các dữ liệu lớn phải mất rất nhiều thời gian. Ví dụ nhóm dân số ở Trung Quốc sẽ mất nhiều thời gian hơn so với nhóm dân số ở TPHCM
* Không cần tìm kiếm toàn bộ cột bảng cho một bản ghi
* [CLUSTERED BY]: phân phối các khóa vào từng buckets khác nhau
* [SKEWED BY]: là một loại bảng đặc biệt trong đó các giá trị của bảng xuất hiện thường xuyên được chia thành các tệp riêng biệt và phần còn lại của các giá trị sẽ chuyển sang một số tệp khác. Ví dụ: Ví dụ: trong số 100 bệnh nhân nhiễm COVID, 90 bệnh nhân chạy thận nhân tạo và 10 bệnh nhân khác bị sốt, cảm, v.v ... Vì vậy, một phân vùng sẽ được tạo cho 90 bệnh nhân và một phân vùng sẽ được tạo cho 10 bệnh nhân còn lại.
* [STORED AS DIRECTORIES] là một tham số tùy chọn. nó là một bảng skewed mà còn có tính năng khóa danh sách
* [SERDEPROPERTIES]: liên kết các thuộc tính SerDes với bảng bằng cách chỉ định các cặp key-value
* [TBLPROPERTIES] được sử dụng để liên kết các mục của siêu dữ liệu với một bảng. Lấy danh sách các cặp key-values được phân tách bằng dấu phẩy. Lưu trữ các cặp key-values trong metastore database.



* [IGNORE PROTECTION] (bỏ qua bảo vệ): Đây là tính năng cho phép ta bỏ qua bảo vệ cho các phân vùng khi xóa. Trong trường hợp ta sử dụng di chuyển phân vùng giữa các clusters. Các bảng không được bảo vệ và được bảo vệ là non-atomic và do đó tại sao metadata của ta ở trạng thái bẩn
* [PURGE]: PURGE khi ta hoàn toàn chắc chắn xóa dữ liệu của bảng.



* [CASCADE|RESTRICT]: CASCADE Thay đổi các cột của siêu dữ liệu của bảng, RESTRICT là default, giới hạn thay đổi cột chỉ với siêu dữ liệu bảng.

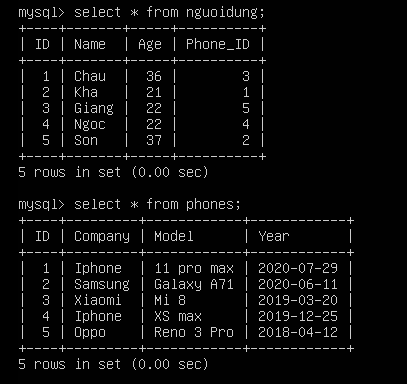
1. **Ngôn ngữ truy vấn HiveSQL**

Ngôn ngữ truy vấn được sử dụng trong Apache Hive được gọi là HiveQL. HiveQL cung cấp nhiều phần mở rộng hơn so với SQL thông thường. Còn có thêm các thuộc tính ACID (Atomicity, Consistency, Isolation, and Durability) đầy đủ trong HiveQL cùng với các chức năng Update, Insert và Delete. Ta có thể thực hiện chèn nhiều bảng và sử dụng cú pháp như là ‘create table as select’ và ‘create table like’ trong HiveQL.

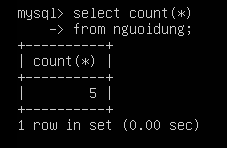
Cách thức mà Hive lưu trữ và truy vấn dữ liệu gần giống với cơ sở dữ liệu thông thường. Nhưng vì Hive được xây dựng dựa trên Hadoop, nên nó phải tuân thủ các quy tắc được đặt ra bởi Hadoop.

1. **Ví dụ minh họa về truy vấn dùng HiveSQL:**

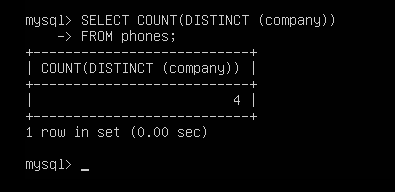
Ta tạo 2 bảng nguoidung và phones dưới đây



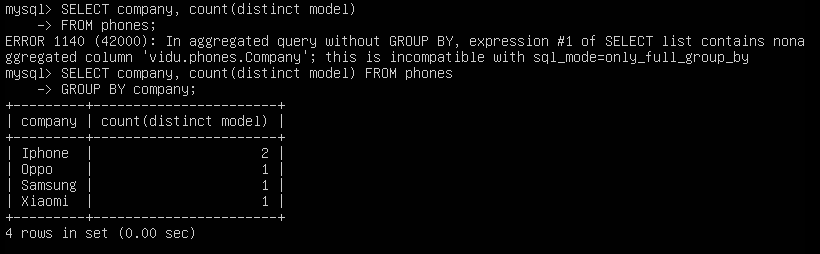
Query 1: Đếm tổng số hàng trong bảng



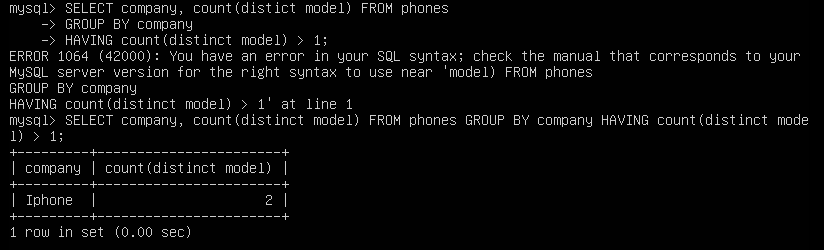
Query 2: Đếm có bao nhiêu hãng điện thoại khác nhau



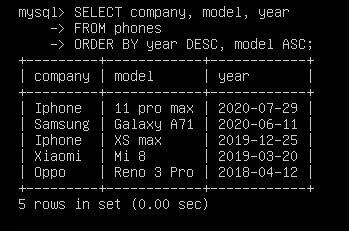
Query 3: Đếm số lượng hãng điện thoại



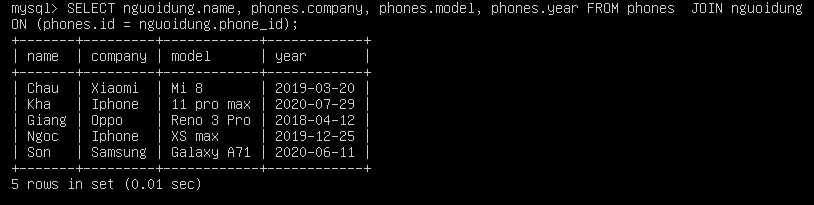
Query 4: Đếm số lượng hã ng điện thoại có nhiều hơn 1 (dung HAVING)



Query 5: Sắp xếp company, model theo thứ tự từ mới đến cũ (sử dụng ORDER BY)



Query 6: Kết hợp 2 bảng lại với nhau



# II – Mở rộng phân tích dữ liệu lớn với Apache Hive

## Các nhược điểm của Hive

Hadoop đang ngày càng trở thành một công cụ thông dụng dùng cho lưu trữ và xử lý dữ liệu ở nhiều doanh nghiệp đa dạng. Nhờ đó, Hive đã trở thành một công cụ để thông qua nó ta có thể thao tác với dữ liệu, thu thập dữ liệu được lưu trữ và quản lý trên Hadoop. Và do Hive ngày càng trở nên thông dụng trong các lĩnh vực, do đó khối lượng dữ liệu mà Hive quản lý cũng theo đó tăng lên gấp nhiều lần. Lượng câu truy vấn ở trên Hive đã trở nên đa dạng hơn nhiều.

Xét về kho lưu trữ (data storage), người dùng mong muốn rằng Hive có thể lưu trữ một khối lượng dữ liệu lớn một cách hiệu quả trong một không gian lưu trữ có giới hạn (nhỏ). Việc này giúp cho người dùng có thể giảm thiểu chi phí doanh nghiệp khi sử dụng HDFS.

Xét về thực hiện câu truy vấn (query execution), Hive đã có khả năng xử lý một lượng dữ liệu lớn, thông qua Hadoop MapReduce thực hiện at run-time. Tuy nhiên với số lượng người dùng ngày càng tăng, Hive phải đáp ứng được thời gian trả lời câu truy vấn nhanh hơn. Một nguyên do khác mà Hive cần phải cải thiện thời gian thực hiện truy vấn là do Hive đang dần được tích hợp với các công cụ business intelligence. Người dùng sử dụng công cụng này mong muốn rằng các câu truy vấn phải được trả lời nhanh (interactive).

Để đáp ứng hai yêu cầu này thì hiệu suất của kho lưu trữ và hiệu suất thực hiện câu truy vấn của Hive phải được cải tiến hơn. Trong phần sau đây chúng tôi sẽ nói về những thành phần (component) quan trọng của Hive, sẽ quyết định hiệu suất lưu trữ và hiệu suất thực hiện câu truy vấn, và chúng tôi sẽ xác định những thiếu sót trong các thành phần này của Hive.

Ở trong Hive, hiệu suất của kho lưu trữ (data storage) được quy định bởi phương thức SerDes (Serializer – Deserializer) và các định dạng file (file format). Ban đầu, Hive chỉ sử dụng hai định dạng file đơn giản được cung cấp bởi Hadoop chính là TextFile và SequenceFile. Một tập tin được lưu dưới dạng TextFile chứa dữ liệu plain text. Ngược lại, một file được lưu dưới dạng SequenceFile là một dạng file phẳng (flat file: file không có cấu trúc ở bên trong) lưu trữ các cặp key-value. Bởi vì hai file format này là data-type-agnostic (nghĩa là nó không quan đến định dạng của dữ liệu, nó không quan tâm dữ liệu là string hay float hay integer). Vì lý do đó, Hive sử dụng SerDes để serialize mỗi dòng của dữ liệu thành plain-text hoặc các chuỗi binary (binary sequences) để ghi dữ liệu vào HDFS, và Hive sẽ deserialize dữ liệu theo hai định dạng này (TextFile và SequenceFile) lại thành các hàng (row) theo schema ban đầu của nó lúc mà đọc dữ liệu từ HDFS. Và bởi vì 2 định dạng file này lưu trữ dòng theo kiểu row-by-row (dòng theo dòng), rất khó để dữ liệu có thể được nén một cách hiệu quả. Trong Hive 0.4, Record Columnar File (RCFile) được giới thiệu. RCFile là một định dạng file theo kiểu cột do đó nó đạt được hiệu suất lưu trữ tốt hơn, tuy nhiên RCFile vẫn là một định dạng file data-type-agnostic, phương thức SerDes của nó sẽ serialize từng dòng, one-row-at-a-time. **Một trong những nhược điểm đầu tiên của Hive về hiệu suất lưu trữ kho là các định dạng file data-type-agnostic, và phương pháp serialization one row at a time (serialize từng dòng) khiến cho các giá trị dữ liệu khó được nén một cách hiệu quả.**

Hiệu suất của việc thực thi câu truy vấn thì được quy định bởi thành phần định dạng tệp (file format component), thành phần lập kế hoạch thực thi truy vấn (query planner component), và thành phần thực hiện các câu truy vấn (query execution component) chứa đựng cách thực thi các toán tử (operator). Mặc dù định dạng file RCFile là định dạng cột, phương thức SerDes của nó không cho phép phân rã các loại dữ liệu phức tạp ra (ví dụ: Map). Do đó, khi một query cần truy cập vào một field của một loại dữ liệu phức tạp, thì tất cả các field thuộc loại dữ liệu này cần phải được đọc qua chứ không thể đọc riêng lẻ được. Do đó quá trình đọc dữ liệu có đôi chút không hiệu quả. Ngoài ra RCFile chủ yếu được thiết kế để đọc dữ liệu theo kiểu tuần tự (sequential). RCFile không có sử dụng chỉ mục (index) và nó không tận dụng thông tin về ngữ nghĩa (semantic information) được cung cấp bởi câu truy vấn, giúp bỏ qua các phần dữ liệu không cần thiết**. Do đó một trong những thiếu sót thứ hai về thành phần định dạng tệp (file format component) là hiệu suất đọc dữ liệu bị hạn chế do thiếu chỉ mục (index) và không có khả năng phân rã các cột có kiểu dữ liệu phức tạp.**

Trình lập kế hoạch (query planner) trong Hive sẽ dịch mỗi một công việc trong câu truy vấn thành một toán tử (operator). Khi một công việc yêu cầu dữ liệu đầu vào được phân vùng theo cách nhất định thì Hive sẽ chèn RSOps làm ranh giới trên giữa giai đoạn Map và giai đoạn Reduce. Sau đó, trình lập kế hoạch sẽ phân tích cây toán tử thành các job MapReduce giữ trên ranh giới của cây này. Trong quá trình lập kế hoạch thực hiện câu truy vấn và tối ưu hóa truy vấn, trình lập kế hoạch chỉ tập trung vào một hoạt động duy nhất, hoặc một MapReduce job duy nhất tại một thời điểm. phương pháp lập kế hoạch truy vấn này có thể làm giảm đáng kể hiệu quả thực thi câu truy vấn đó nó đưa ra rất nhiều các operation không cần thiết và tốn thời gian. Ví dụ như là trình lập kế hoạch không nhận thức được mối tương quan giữa các operation trong một câu truy vấn**. Do đó thiếu sót thứ ba của query planner là nó bỏ qua mối quan hệ giữa các operation ở trong query làm giảm hiệu suất của việc thực thi câu truy vấn.**

Thành phần thực thi truy vấn của Hive (query execution component) bị ảnh hưởng nặng nề bởi mô hình hoạt động của Hadoop MapReduce. Trong một task Map hoặc task Reduce, thì MapReduce engine sẽ lấy một cặp key/value và chuyển tiếp nó tới hàm Map hoặc hàm Reduce tại từng thời điểm. Ví dụ: trong wordcount, mỗi cái task Map sẽ xử lý một dòng trong tệp văn bản. Hive đã kế thừa mô hình này của Hadoop, và Hive xử lý các hàng theo phương thức one row at a time, Tuy nhiên mô hình này không phù hợp với kiến trúc của máy CPU thời hiện đại. Mô hình hoạt động này có chi phí thông dịch cao, đồng thời không sử dụng triệt để tính năng song song (parallelism) sẵn có là bản chất của Apache Hadoop, ngoài ra Hive còn có hiệu suất lưu trữ cache thấp, và có chi phí gọi hàm khá cao. **Do đó thiếu sót thứ tư của thành phần thực thi truy vấn (query execution component) là hiệu suất của trình biên dịch trong thời gian thực realtime hay bị giới hạn bởi mô hình thực thi theo kiểu một hàng one row at a time.**

## Các phương pháp phân tích

Big data chỉ hữu dụng nếu ta biết khai thác nó, nhưng làm sao để phân tích dữ liệu một cách hiệu quả? Ta có thể sử dụng nhiều phương pháp phân tích để đạt được hiệu suất doanh nghiệp cao nhất

|  |  |
| --- | --- |
| Phương pháp phân tích | Mô tả |
| Phân tích cơ bản để đưa ra insight | Slice và dice dữ liệu, báo cáo (reporting), các phương pháp trực quan hóa dữ liệu đơn giản, giám sát dữ liệu |
| Phân tích chuyên sâu | Các phương pháp phân tích nâng cao hơn như predictive modeling và các phương pháp khác sử dụng pattern matching |
| Operationalized analytics | Phân tích dữ liệu thành một quá trình họat động của doanh nghiệp |
| Monetized analytics | Hành động phân tích dữ liệu tạo ra nguồn lợi nhuận trực tiếp |

## Các giai đoạn phân tích big data

Bốn giai đoạn là một phần của quy trình lập kế hoạch áp dụng dữ liệu lớn. Khi nhiều doanh nghiệp bắt đầu sử dụng đám mây như một cách để triển khai các dịch vụ cho khách hàng, vai trò của phân tích dữ liệu cũng bùng nổ. Do đó, hãy xem xét quy trình lập kế hoạch phân tích dữ liệu:

**Giai đoạn 1: Lập kế hoạch**

Cách duy nhất để đảm bảo rằng các nhà lãnh đạo doanh nghiệp nắm vững các yếu tố của doanh nghiệp là hiểu rõ các nguồn dữ liệu tới từ đâu và chúng liên quan với nhau ra sao. Doanh nghiệp cần một lộ trình xác định dữ liệu nào là cần thiết để lập kế hoạch cho các chiến lược mới và hướng đi mới.

**Giai đoạn 2: Thực hiện phân tích**

Phân tích dữ liệu lớn đòi hỏi ta phải học một bộ công cụ mới và các kỹ năng mới. Nhiều tổ chức phải thuê nhà khoa học dữ liệu lớn, những người biế cách lấy được lượng dữ liệu lớn và đa dạng và hiểu dữ liệu liên quan đến bối cảnh của vấn đề.

**Giai đoạn 3: Kiểm tra kết quả**

Hãy đảm bảo rằng bạn không dựa vào các nguồn dữ liệu sẽ đưa bạn đi sai hướng. Nhiều công ty sẽ sử dụng các nguồn dữ liệu của bên thứ ba và không bỏ thời gian để kiểm tra chất lượng dữ liệu, nhưng ta phải đảm bảo rằng nền tảng dữ liệu của ta là vững chắc.

**Giai đoạn 4: Thực hiện kế hoạch**

Mỗi khi một doanh nghiệp bắt đầu một chiến lược mới, điều quan trọng là phải liên tục tạo ra một chu kỳ đánh giá dữ liệu lớn. Cách tiếp cận hành động này dựa trên kết quả phân tích dữ liệu lớn và sau đó thử nghiệm kết quả thực hiện chiến lược kinh doanh là chìa khóa thành công.

**Giai đoạn 5: Giám sát theo thời gian thực**

Phân tích dữ liệu lớn cho phép bạn giám sát dữ liệu thời gian thực một cách chủ động. Điều này có thể có tác động lớn đến doanh nghiệp.

Ví dụ: Một công ty dược phẩm tiến hành thử nghiệm lâm sàng, họ có thể điều chỉnh hoặc hủy thử nghiệm để tránh bị kiện.

**Giai đoạn 6: Điều chỉnh tác động**

Khi công ty có các công cụ để giám sát liên tục, họ có khả năng điều chỉnh các quy trình và chiến lược dựa trên phân tích dữ liệu. Có thể giám sát nhanh chóng có nghĩa là một quy trình có thể được thay đổi sớm hơn và dẫn đến chất lượng tổng thể tốt hơn.

**Giai đoạn 7: Thử nghiệm**

Kết hợp thử nghiệm với giám sát thời gian thực và điều chỉnh có thể giúp thay đổi chiến lược kinh doanh. Bạn có ít rủi ro hơn với thử nghiệm vì bạn có thể thay đổi hướng và kết quả dễ dàng hơn nếu bạn được trang bị dữ liệu phù hợp.

## Các xu hướng dữ liệu lớn hiện nay tích hợp Apache Hive

**Hadoop đang trở thành nền tảng của quản lý dữ liệu lớn phân tán.** Hadoop là một hệ thống tệp phân tán có thể được sử dụng cùng với MapReduce để xử lý và phân tích lượng dữ liệu khổng lồ, thúc đẩy xu hướng dữ liệu lớn. Hadoop sẽ được tích hợp chặt chẽ vào các công nghệ data warehousing để dữ liệu có cấu trúc và không cấu trúc có thể được tích hợp hiệu quả hơn.

**Dữ liệu lớn giúp có thể tận dụng dữ liệu từ các cảm biến để thay đổi kết quả kinh doanh.** Ngày càng có nhiều doanh nghiệp sử dụng các cảm biến rất tinh vi trên các thiết bị trong doanh nghiệp. Những cải tiến mới trong công nghệ dữ liệu lớn giúp các doanh nghiệp nâng cao hiểu biết về các vấn đề có thể khắc phục để bảo vệ lợi nhuận.

**Dữ liệu lớn có thể giúp một sáng kiến kinh doanh trở thành một hành động thời gian thực để tăng doanh thu.** Các công ty trong các thị trường như bán lẻ đang sử dụng phân tích dữ liệu phát trực tuyến theo thời gian thực để theo dõi hành động của khách hàng và cung cấp các ưu đãi để tăng doanh thu trên mỗi khách hàng.

**Dữ liệu lớn có thể thay đổi cách quản lý dịch bệnh, bệnh tật bằng cách thêm các phân tích dự đoán.** Ngày càng nhiều các chuyên gia chăm sóc sức khỏe tìm kiếm các giải pháp dữ liệu lớn để hiểu rõ hơn về bệnh tật bằng cách so sánh các triệu chứng và kết quả xét nghiệm với cơ sở dữ liệu về kết quả từ hàng trăm ngàn trường hợp khác. Điều này cho phép các học viên dự đoán nhanh hơn kết quả.

# III – DEMO

## Giới thiệu tập dữ liệu

Google Play Store dataset là tập dữ liệu được lấy từ cửa hàng ứng dụng Android trên Google Play Store sử dụng phương pháp web scraping. Dữ liệu về Play Store có tiềm năng rất lớn để giúp các công ty chế tạo ứng dụng phát triển hơn trên thị trường Android, thông qua các hành động phân tích được từ các insight từ dữ liệu.

Dữ liệu được lấy từ thời gian 2010 đến 2018.

Mỗi một dòng trong bảng mô tả thông tin về ứng dụng, rating, kích thước, vv

## Cơ sở dữ liệu

|  |  |
| --- | --- |
| Đối tượng | Mô tả |
| Database: googleplaystore | Gồm 2 bảng |
| Table: apps | Số dòng: 9144  Số cột: 13  Tên cột: app, category, reviews, size, installs, type, price, content\_rating, genres, last\_updated, current\_ver, android\_ver |
| Table: reviews | Số dòng: 37432  Số cột: 5  Tên cột: app, review, sentiment, sentiment\_polarity, sentiment\_subjectivity |

## Chuẩn bị dữ liệu

### Làm sạch dữ liệu

#### Biến đổi giá trị chuỗi thành giá trị số thực: size, installs, price Xử lý giá trị null: rating, reviews, size,

|  |
| --- |
| import pandas as pddf = pd.read\_csv("C:/Users/8570w/Downloads/google play store apps/googleplaystore.csv")#APPdf["App"]=df["App"].str.replace(",",'')df["App"]=df["App"].str.replace(";",'')#RATINGdf['Rating'] = df['Rating'].fillna(0)#SIZEdf["Size"]= df["Size"].str.replace("M",'')df.loc[df["Size"].str.endswith('k'),"Size"] = '0.' + df['Size']df["Size"]= df["Size"].str.replace("k",'')df.drop(df[df['Size'] == '1,000+'].index, inplace=True)df.loc[df['Size'] == '1,000+']df.drop(df[df['Size'] == 'Varies with device'].index, inplace=True)df.loc[df['Size'] == 'Varies with device']df.drop(df[df['Size'] == '0.8.5'].index, inplace=True)#INSTALLSdf["Installs"]=df["Installs"].str.replace("+",'')df["Installs"]=df["Installs"].str.replace(",",'')#PRICEdf["Price"]=df["Price"].str.replace("$",'')dfdf["App"].replace({',':''}, regex=True)#GENRESdf["Genres"]=df["Genres"].str.replace(";",'')#LAST\_UPDATEDdf["Last Updated"]=df["Last Updated"].str.replace(",",'')df["Last Updated"]=df["Last Updated"].str.replace("January",'1')df["Last Updated"]=df["Last Updated"].str.replace("February",'2')df["Last Updated"]=df["Last Updated"].str.replace("March",'3')df["Last Updated"]=df["Last Updated"].str.replace("April",'4')df["Last Updated"]=df["Last Updated"].str.replace("May",'5')df["Last Updated"]=df["Last Updated"].str.replace("June",'6')df["Last Updated"]=df["Last Updated"].str.replace("July",'7')df["Last Updated"]=df["Last Updated"].str.replace("August",'8')df["Last Updated"]=df["Last Updated"].str.replace("September",'9')df["Last Updated"]=df["Last Updated"].str.replace("October",'10')df["Last Updated"]=df["Last Updated"].str.replace("November",'11')df["Last Updated"]=df["Last Updated"].str.replace("December",'12')df["Last Updated"]=df["Last Updated"].str.replace(" ",'-')print(df["Last Updated"])#CURRENT VERdf["Current Ver"]=df["Current Ver"].str.replace(",",'.')df["Current Ver"]=df["Current Ver"].str.replace(";",'.')df.to\_csv('C:/Users/8570w/Downloads/google play store apps/apps.csv',index=False, header=False) |

#### Reviews

|  |
| --- |
| import pandas as pddf = pd.read\_csv("C:/Users/8570w/Downloads/google play store apps/googleplaystore\_user\_reviews.csv")#APPdf["App"]=df["App"].str.replace(",",'')df["App"]=df["App"].str.replace(";",'')#reviewsdf["Translated\_Review"]=df["Translated\_Review"].str.replace(",",'.')df["Translated\_Review"]=df["Translated\_Review"].str.replace(";",'.')#delete nullsdf.drop(df[df['Sentiment'].isnull() & df['Sentiment\_Polarity'].isnull() & df['Sentiment\_Subjectivity'].isnull()].index, inplace=True)#Reorder columnsdf = df[['App', 'Sentiment', 'Sentiment\_Polarity', 'Sentiment\_Subjectivity', 'Translated\_Review']]df.to\_csv('C:/Users/8570w/Downloads/google play store apps/apps\_reviews.csv',index=False,header=False) |

### Tạo bảng

<https://medium.com/@acpanjan/download-google-drive-files-using-wget-3c2c025a8b99>

|  |
| --- |
| wget --no-check-certificate 'https://docs.google.com/uc?export=download&id=FILEID' -O FILENAME |

wget --no-check-certificate 'https://docs.google.com/uc?export=download&id= 1euNAVLLnbV2yrrpjl4l4WX1RsvSoVIq3' -O apps.csv

#### a. Tạo table apps

|  |
| --- |
| **DROP DATABASE IF EXISTS googleplaystore CASCADE;**  **CREATE SCHEMA IF NOT EXISTS googleplaystore;**  **DROP TABLE IF EXISTS googleplaystore.apps\_temp;**  **CREATE EXTERNAL TABLE IF NOT EXISTS googleplaystore.apps\_temp**  **(app STRING**  **,category STRING**  **,rating FLOAT**  **,reviews INT**  **,size FLOAT**  **,installs INT**  **,type STRING**  **,price FLOAT**  **,content\_rating STRING**  **,genres STRING**  **,last\_updated STRING**  **,current\_ver STRING**  **,android\_ver STRING**  **)**  **ROW FORMAT DELIMITED FIELDS TERMINATED BY ','**  **STORED AS TEXTFILE;**  **SET skip.header.line.count = 1;**  **CREATE EXTERNAL TABLE IF NOT EXISTS googleplaystore.apps**  **(app STRING**  **,category STRING**  **,rating FLOAT**  **,reviews INT**  **,size FLOAT**  **,installs INT**  **,type STRING**  **,price FLOAT**  **,content\_rating STRING**  **,genres STRING**  **,last\_updated STRING**  **,current\_ver STRING**  **,android\_ver STRING**  **)**  **ROW FORMAT DELIMITED FIELDS TERMINATED BY ','**  **STORED AS TEXTFILE;**  **-- LOAD DATA TO APPS\_TEMP**  **LOAD DATA LOCAL INPATH '/home/hadoopuser/apps.csv' OVERWRITE INTO TABLE googleplaystore.apps\_temp;**  **-- LOAD DATA TO APPS**  **USE googleplaystore;**  **INSERT INTO TABLE googleplaystore.apps**  **SELECT app,category,rating,reviews,size,installs,type,price,content\_rating,**  **genres,**  **from\_unixtime(unix\_timestamp(last\_updated,'mm-dd-yyyy'),'yyyy-mm-dd')**  **, current\_ver, android\_ver**  **from googleplaystore.apps\_temp;**  **-- DISPLAY INSERTED DATA**  **SELECT last\_updated FROM googleplaystore.apps LIMIT 2;**  **-- REMOVE TEMP TABLE** DROP TABLE IF EXISTS googleplaystore.apps\_temp; |

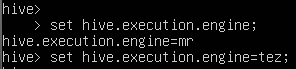
#### b. Tạo table reviews

|  |
| --- |
| **CREATE EXTERNAL TABLE IF NOT EXISTS googleplaystore.apps\_reviews**  **(app STRING**  **,sentiment STRING**  **,sentiment\_polarity FLOAT**  **,sentiment\_subjectivity FLOAT**  **,review STRING**  **)**  **ROW FORMAT DELIMITED FIELDS TERMINATED BY ','**  **STORED AS TEXTFILE;**  **LOAD DATA LOCAL INPATH '/home/hadoopuser/apps\_reviews.csv' OVERWRITE INTO TABLE googleplaystore.apps\_reviews;** |

### Configure Hive

Sử dụng execution engine là tez thay vì mapreduce

hive> set hive.execution.engine = tez;



Giải thích [kế hoạch thực hiện](https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Explain) của một query

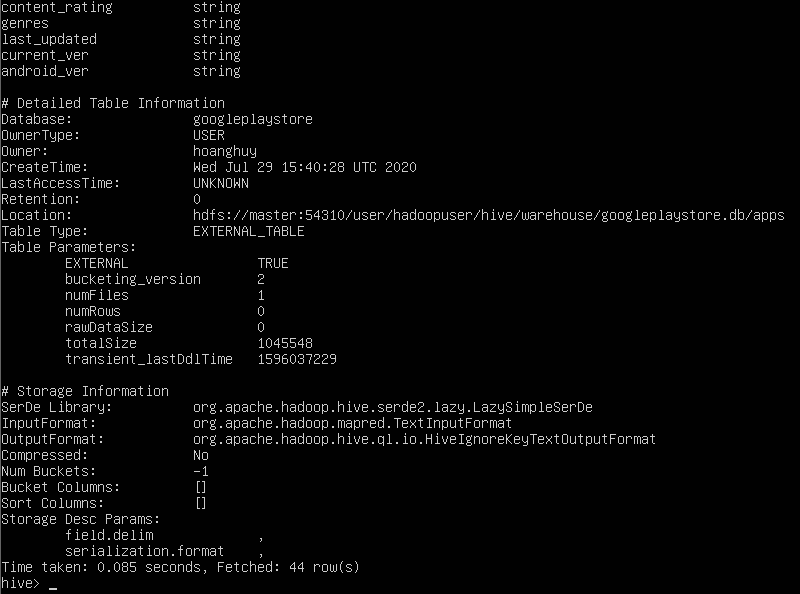
hive> explain select \* from apps;

In ra màn hình tên cột mỗi lần thực hiện query

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

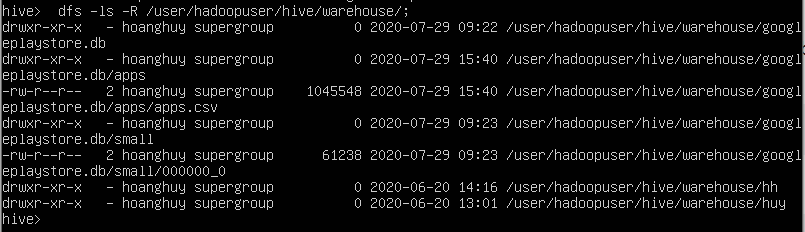
In ra màn hình tóm tắt, các chi tiết và thông tin được format của database và partition của nó (nếu có)

hive> describe formatted googleplaystore.apps;



Hiển thị tất cả database và table và file trên HDFS

hive> dfs –ls –R user/hadoopuser/hive/warehouse/ ;



Xóa table khỏi Hive (đồng nghĩa với xóa khỏi HDFS)

hive> dfs –rm –r user/hadoopuser/hive/warehouse/googleplaystore.db/apps ;

## Các câu truy vấn Hive

### Tạo partition table theo năm của app

drop table if exists partition\_year;

create table partition\_year

(

app STRING

,category STRING

,rating FLOAT

,last\_updated STRING

)

PARTITIONED BY (year int)

row format delimited

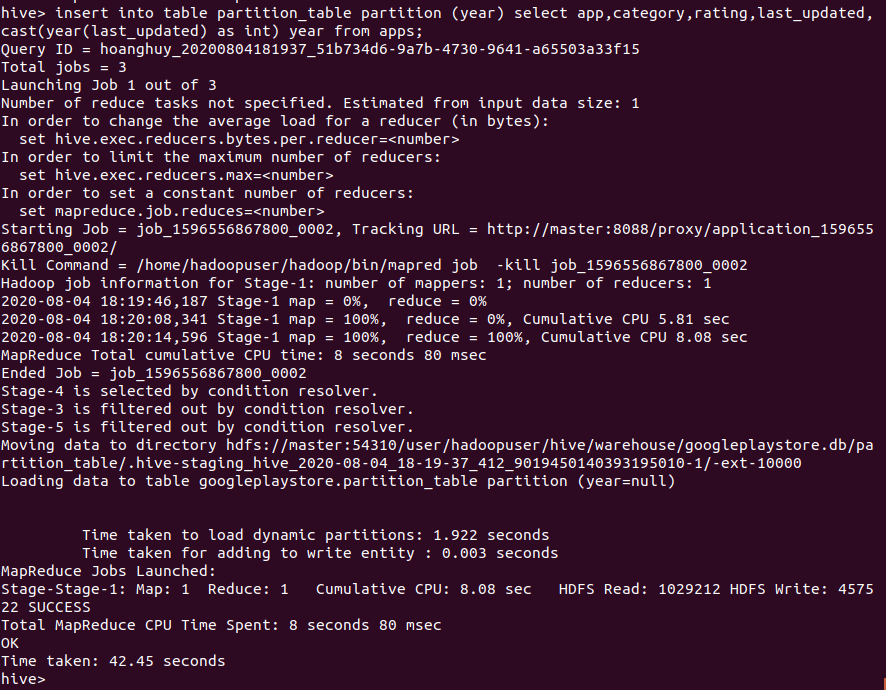
fields terminated by ','

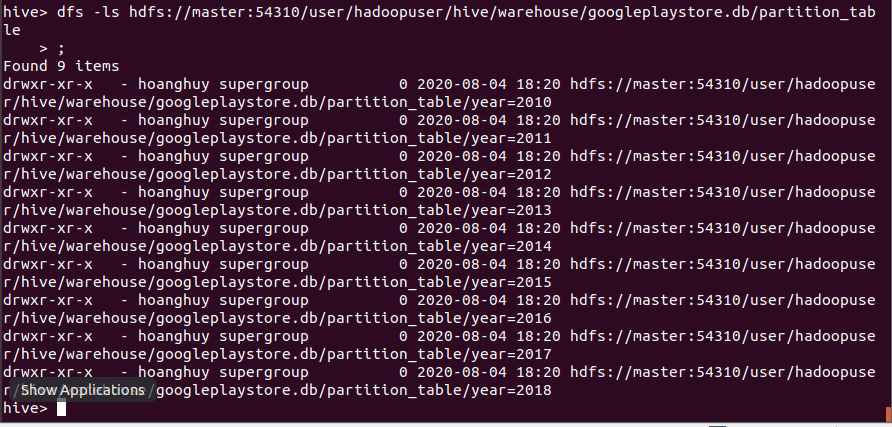
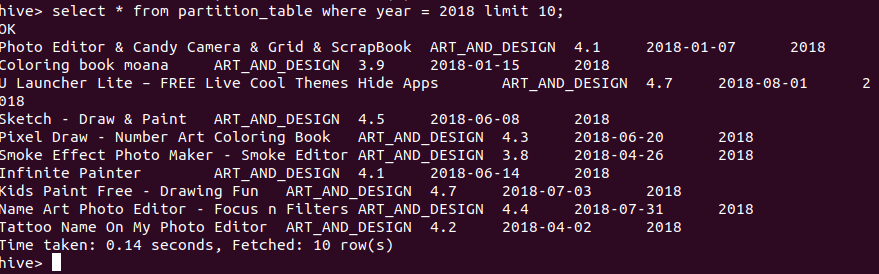
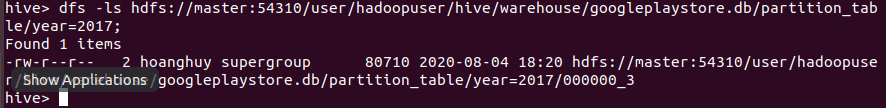
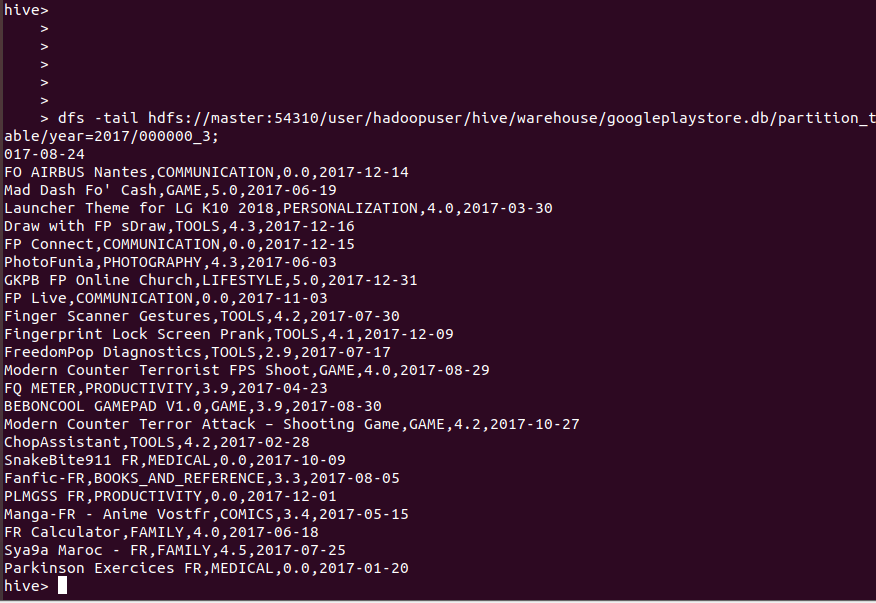
stored as textfile;

set hive.exec.dynamic.partition.mode=nonstrict;

insert into table partition\_year partition (year)

select app,category,rating,last\_updated,cast(year(last\_updated) as int) year from apps;



### Tìm những app có điểm đánh giá user (polarity) tốt

drop table if exists partition\_sentiment;

create table partition\_sentiment

(

app STRING

,rating FLOAT

,polarity float

,subjectivity float

)

PARTITIONED BY (polarity\_range STRING, subjectivity\_range STRING)

row format delimited

fields terminated by ','

stored as textfile;

set hive.exec.dynamic.partition.mode=nonstrict;

insert into table partition\_sentiment partition (polarity\_range,subjectivity\_range)

select r.app app,a.rating rating, r.polarity polarity, r.subjectivity subjectivity,

case when polarity between -1 and -0.5 then 'very bad'

when polarity between -0.4 and 0 then 'bad'

when polarity between 0.1 and 0.5 then 'good'

when polarity between 0.6 and 1 then 'very good'

else '0'

end as polarity\_range,

case when polarity > 0.4 then 'subjective'

else 'objective'

end as subjectivity\_range

from

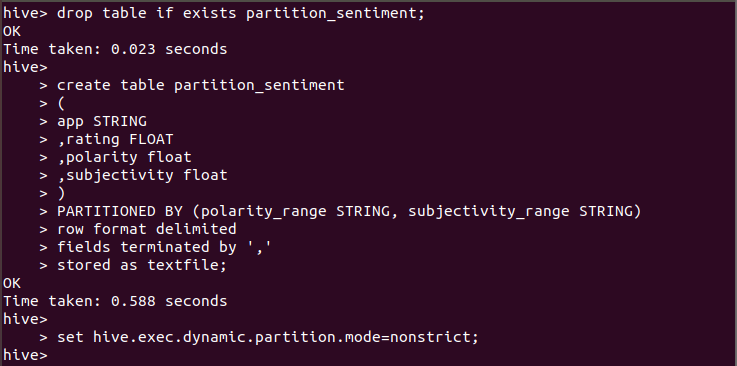
(

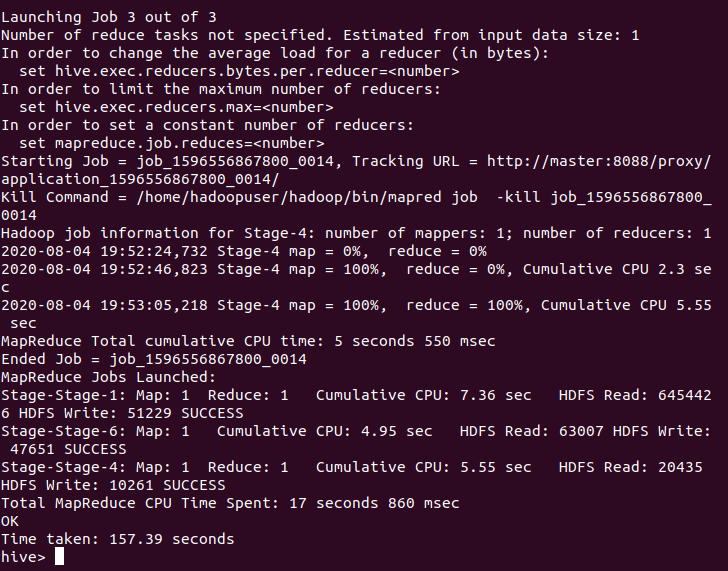
select app, avg(sentiment\_polarity) polarity, avg(sentiment\_subjectivity) subjectivity

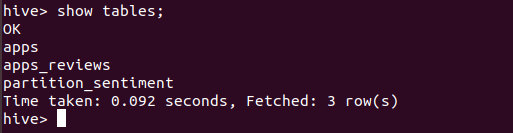
from apps\_reviews

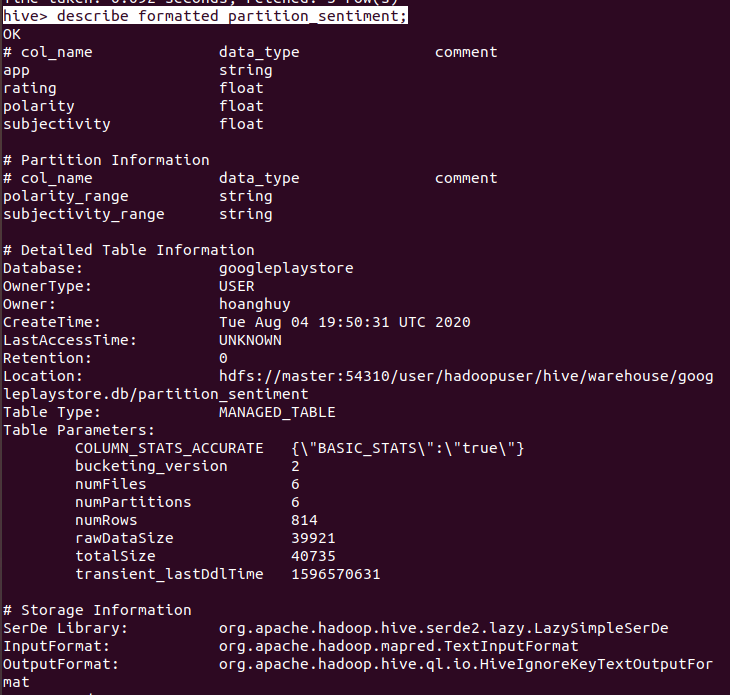
group by app

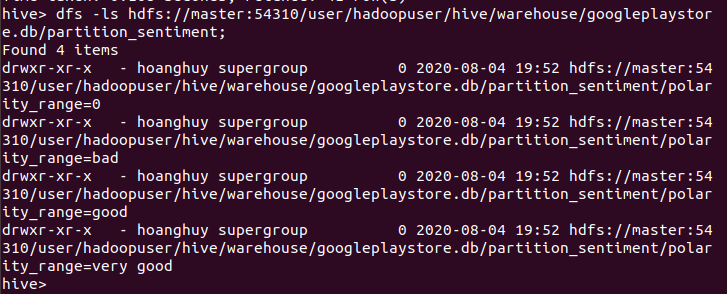
) r inner join apps a on r.app = a.app;

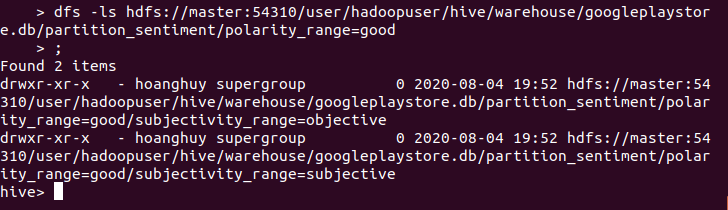


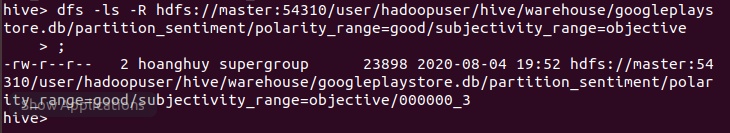


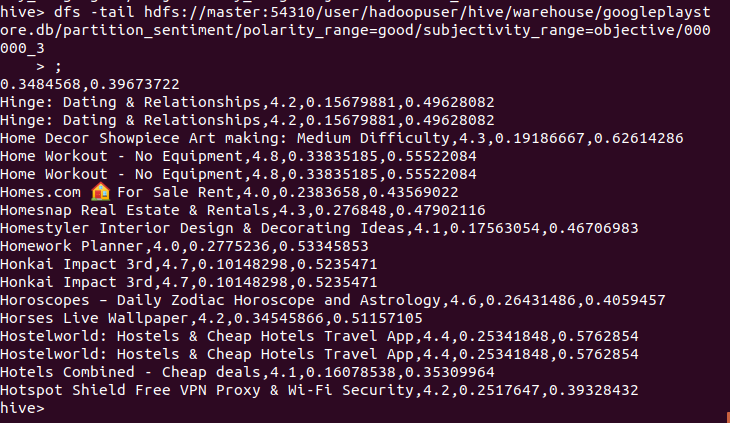












### Tìm các ứng dụng có trả phí có lượt tải từ cao nhất đến thấp nhất. Cho biết tên ứng dụng, lượt tải và giá tiền

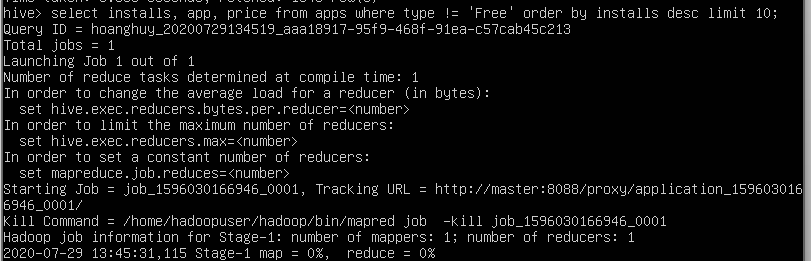
select installs, app, price

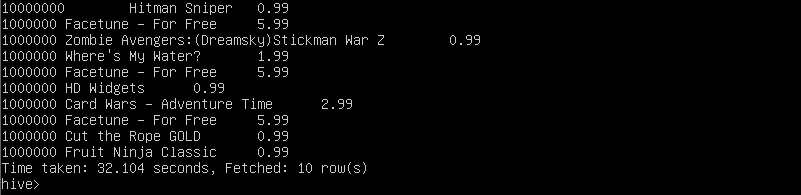
from apps

where type != 'Free'

order by installs desc;

select installs, app, price from apps where type != 'Free' order by installs;





### Xem các tier lượt tải qua từng năm đếm số lượng app, số lượng review trung bình cho mỗi app, và rating trung bình cho nhóm tải

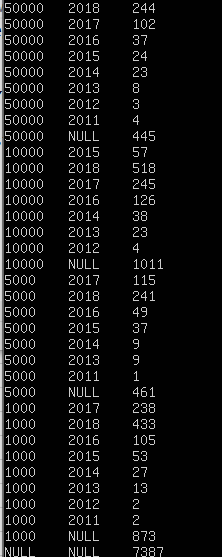
select Installs, Year(Last\_Updated),count(App)

from apps

where Installs > 500

group by Rollup(Installs,Year(Last\_Updated))

order by Installs desc;



### Sắp xếp các app theo điểm review xếp từ cao xuống thấp. Cho biết tên app, số điểm, sentiment, số lượng người tải, thể loại, kích thước

create view if not exists app\_sentiment as

select apps.app, avg(sentiment\_polarity) as avgsentiment

from apps\_reviews r inner join apps on r.app = apps.app

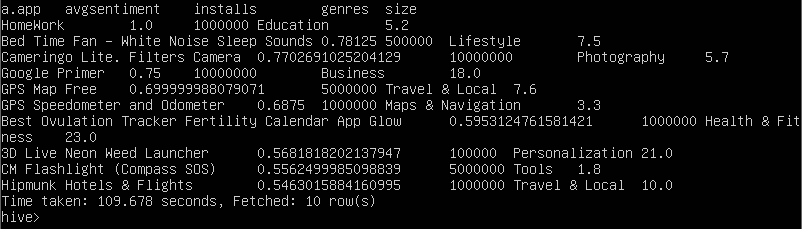
group by apps.app

select distinct a.app, avgsentiment, installs, genres, size

from

app\_sentiment f inner join apps a on f.app = a.app

order by avgsentiment desc limit 10;

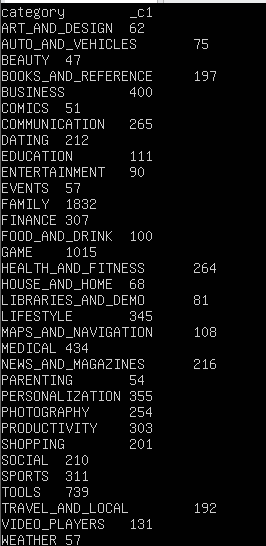


### Tìm số lượng app cho mỗi thể loại

select Category, count(App)

from apps

group by Category;



### Sắp xếp thể loại theo số lượng app và đếm phần tram của thể loại

create view if not exists category\_count\_by\_apps as

select Category, count(App) as apps

from apps

group by Category

select category, apps, apps \*100.0/ (select count(App) from apps) as popularity

from category\_count\_by\_apps

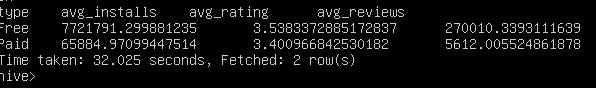
order by popularity desc;

### Số lượng lượt tải trung bình và lượng review trung bình và điểm review trung bình của loại app miễn phí và app có trả phí

select type,avg(installs) as avg\_installs, avg(rating) as avg\_rating, avg(reviews) as avg\_reviews

from apps

group by type;



# IV - Tài liệu tham khảo

<https://www.cs.rit.edu/usr/local/pub/GraduateProjects/2161/ksc8618/Report.pdf>

<https://www.youtube.com/watch?v=y7Z79b2sGQ8>

<https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Explain>

<https://www.rishabhsoft.com/blog/what-is-big-data-analytics-and-types>

<http://web.eecs.umich.edu/~mosharaf/Readings/Hive.pdf>

<https://semode.tistory.com/26>