**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC THĂNG LONG**



**CHUYÊN ĐỀ TỐT NGHIỆP**

**Thiết kế và xây dựng hệ thống trực quan hóa dữ liệu cho phần mềm quản lí bán hàng**

Sinh viên thực hiện: Nguyễn Thành Hưng

Mã sinh viên: A40509

Chuyên ngành: Công nghệ thông tin

Giảng viên hướng dẫn: CN. Đoàn Trung Phong

**Hà Nội - 2024**

**LỜI CẢM ƠN**

Lời đầu tiên, tôi xin gửi lời cảm ơn chân thành nhất tới CN. Đoàn Trung Phong đã trực tiếp hướng dẫn tôi hoàn thành chuyên đề tốt nghiệp. Suốt quá trình nghiên cứu, thầy đã dành cho tôi rất nhiều thời gian, tâm huyết và sự tận tâm. Thầy đã luôn kiên nhẫn giải đáp những thắc mắc, chỉ ra những thiếu sót trong đề tài và hướng dẫn tôi đi đúng hướng nghiên cứu.

Hơn hết, tôi xin gửi lời cảm ơn đến quý thầy cô giáo trong Khoa Công nghệ thông tin Trường Đại học Thăng Long đã tận tình giảng dạy, truyền đạt kiến thức cho tôi trong suốt quá trình học tập tại trường. Tôi chân thành cảm ơn sự giúp đỡ đến các thành viên khoa Toán-Tin không chỉ là người hỗ trợ mà còn là người bạn đồng hành quan trọng, đóng góp không ngừng vào sự phát triển của tôi. Cảm ơn vì những buổi thảo luận học thuật, sự hỗ trợ vô điều kiện và tất cả những lời khuyên hữu ích hỗ trợ tôi hoàn thành chuyên đề này.

Cuối cùng, tôi muốn gửi lời tri ân sâu sắc đến gia đình và bạn bè đã luôn là nguồn động viên, tiếp thêm sức mạnh cho tôi trong suốt quá trình học tập và nghiên cứu.

Trong quá trình thực hiện chuyên đề của mình, do nhiều yếu tố chủ quan và khách quan nên bài viết còn nhiều hạn chế, thiếu sót, kính mong sự đóng góp ý kiến của thầy cô và các bạn quan tâm đến đề tài này.

Tôi xin chân thành cảm ơn!

*Sinh viên thực hiện*

Nguyễn Thành Hưng

**LỜI CAM ĐOAN**

Tôi xin cam đoan đề tài này được thực hiện dựa trên lý thuyết, và các kỹ thuật liên quan về xử lí dữ liệu được trình bày trong chuyên đề là do tôi thực hiện dưới sự hướng dẫn của CN. Đoàn Trung Phong.

Mọi tài liệu, các bài báo và công cụ của các tác giả khác được sử dụng đều có trích dẫn tường minh về nguồn gốc và tác giả trong danh sách tài liệu tham khảo.

Tôi xin chịu hoàn toàn trách nhiệm về lời cam đoan này!

Hà Nội, ngày 01 tháng 07 năm 2024

*Sinh viên thực hiện*

Nguyễn Thành Hưng

# Bảng chú giải các thuật ngữ

**SQL (Structured Query Language)** - Ngôn ngữ truy vấn có cấu trúc

**SparkSQL** - Ngôn ngữ truy vấn của Spark

**Dataframe** - Khung dữ liệu

**File** - Tệp

**HDFS (Hadoop Distributed File System)** - Hệ thống lưu trữ phân tán Hadoop

**Batch Processing** - Xử lí dữ liệu theo lô

**Mini-batch Processing** - Xử lí dữ liệu theo từng lô nhỏ

**Layer** - Vùng, lớp

**Staging** - Tạm thời

**Aggregate** - Tổng hợp

**Request** - Yêu cầu tới máy chủ

**Metadata** - Siêu thông tin

**API (Application Programing Interface)** - Giao diện lập trình ứng dụng

# Mục lục

[Bảng chú giải các thuật ngữ 4](#_Toc282639263)

[Mục lục 5](#_Toc81400669)

[Chương 1. Tổng quan về bài toán 7](#_Toc1239203565)

[1.1 Đặt vấn đề 7](#_Toc1969929608)

[1.2 Hướng giải quyết 7](#_Toc2017556760)

[1.3 Chi tiết các bước giải quyết vấn đề 8](#_Toc544966656)

[1.3.1 Thu thập dữ liệu giao dịch từ nguồn 8](#_Toc473053122)

[1.3.2 Xử lí và tính toán dữ liệu trong môi trường phân tán 8](#_Toc1595909961)

[1.3.3 Xây dựng các báo cáo để đưa ra thông tin có ích 8](#_Toc639024995)

[Chương 2. Cơ sở lí thuyết 9](#_Toc714101973)

[2.1 Mô hình Snow Flake và Star Schema 10](#_Toc2144542404)

[2.2 Lambda Architecture 12](#_Toc1150881904)

[2.3 Microsoft SQL Server 2022 13](#_Toc527876999)

[2.4 Apache Hadoop 14](#_Toc1472588190)

[Các thành phần chính 15](#_Toc382572558)

[Ưu điểm của Apache Hadoop 16](#_Toc1374482234)

[2.5 Apache Spark 16](#_Toc1274261052)

[Ưu điểm của Apache Spark 16](#_Toc1925340367)

[2.6 Apache Airflow 17](#_Toc1516191574)

[Ưu điểm của Apache Airflow 17](#_Toc1841863521)

[2.7 ClickHouse 18](#_Toc457785504)

[Ưu điểm nổi bật của ClickHouse 18](#_Toc329344179)

[2.8 Apache Superset 19](#_Toc1546669612)

[Ưu điểm của Apache Superset 19](#_Toc493517615)

[2.9 Flask 20](#_Toc27625682)

[2.10 Pandas 20](#_Toc960551090)

[Chương 3. Thiết kế cơ sở dữ liệu 20](#_Toc153921521)

[3.1 Danh sách các bảng 21](#_Toc435959943)

[3.1.1 Bảng Customers 21](#_Toc1028193856)

[3.1.2 Bảng Staffs 22](#_Toc1920155569)

[3.1.3 Bảng Stores 22](#_Toc928073305)

[3.1.4 Bảng Orders 22](#_Toc765779803)

[3.1.5 Bảng OrderItems 23](#_Toc470164126)

[3.1.5 Bảng Categories 23](#_Toc631322478)

[3.1.6 Bảng Brands 23](#_Toc919237753)

[3.1.7 Bảng Stocks 23](#_Toc2038843108)

[3.1.8 Bảng Products 24](#_Toc819788645)

[3.2 Giải thích các quan hệ trong thiết kế cơ sở dữ liệu 24](#_Toc898909498)

[3.2.1 Quan hệ Khách hàng - Đơn hàng 25](#_Toc463584576)

[3.2.2 Quan hệ Cửa hàng - Đơn hàng 25](#_Toc1648060011)

[3.2.3 Quan hệ Nhân viên - Đơn hàng 26](#_Toc1949707546)

[3.2.4 Quan hệ Đơn hàng - Thành phần đơn hàng 27](#_Toc647296002)

[3.2.5 Quan hệ Sản phẩm - Thành phần đơn hàng 28](#_Toc419351633)

[3.2.6 Quan hệ Nhân viên - Quản lí 29](#_Toc1193710033)

[3.2.7 Quan hệ Nhân viên - Cửa hàng 30](#_Toc597548680)

[3.2.8 Quan hệ Cửa hàng - Sản phẩm - Tồn kho 31](#_Toc359462110)

[3.2.9 Quan hệ Sản phẩm - Chủng loại 32](#_Toc1909630428)

[3.2.10 Quan hệ Sản phẩm - Nhãn hàng 33](#_Toc1561019583)

[Chương 4. Xây dựng luồng xử lí dữ liệu 34](#_Toc260023716)

[4.1 Nhập dữ liệu từ phần mềm quản lý 35](#_Toc1593812560)

[4.2 Đồng bộ dữ liệu từ cơ sở dữ liệu sản phẩm về hệ thống phân tích 36](#_Toc1860989682)

[4.2.1 Luồng đồng bộ dữ liệu theo lô 36](#_Toc615918)

[4.2.2 Luồng đồng bộ dữ liệu theo từng lô nhỏ 38](#_Toc2009496328)

[4.3 Tính toán từ layer staging ra các bảng thông tin 38](#_Toc332751091)

[4.4 Đưa dữ liệu đã tính toán ra bảng fact sang ClickHouse để phục vụ truy vấn báo cáo 40](#_Toc1777868716)

[Chương 5. Trực quan hóa dữ liệu 41](#_Toc711562664)

[5.1 Sale dashboard - Phân tích dữ liệu kinh doanh theo thời gian 42](#_Toc1824688612)

[5.1.1 Biểu đồ Revenue By Time 42](#_Toc180306706)

[5.1.2 Biểu đồ Top customers 43](#_Toc1249673722)

[5.1.3 Biểu đồ Orders status 44](#_Toc473216127)

[5.2 Human Resource - Phân tích dữ liệu nhân viên tại các cửa hàng 45](#_Toc1637996674)

[5.2.1 Biểu đồ Store map 45](#_Toc1096409416)

[5.2.2 Biểu đồ Revenue by stores 46](#_Toc254107705)

[5.2.3 Biểu đồ Human Resouce 47](#_Toc213011215)

[5.2.4 Biểu đồ Percentage of revenue by staff 48](#_Toc577879135)

[Chương 6. Tổng kết và định hướng 49](#_Toc52632234)

[6.1 Tổng kết 50](#_Toc980184247)

[6.2 Định hướng 50](#_Toc2110628157)

[Phụ lục 50](#_Toc1840150439)

[Giải thích các khái niệm 51](#_Toc503020612)

[Relational Database Management System (RDBMS) 51](#_Toc322310921)

[Data Warehouse 51](#_Toc370920181)

[Transactional Data 52](#_Toc647835407)

[Dim Table và Fact Table trong thiết kế Data Warehouse 52](#_Toc692491415)

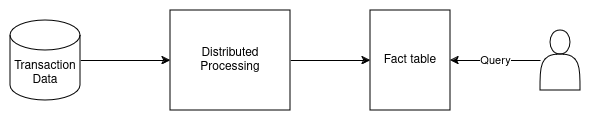
# Chương 1. Tổng quan về bài toán

## 1.1 Đặt vấn đề

Sự phát triển nhanh, mạnh mẽ của mạng Internet và công nghệ thông tin hiện nay khiến lượng thông tin người dùng có thể tra cứu được lớn hơn rất nhanh qua từng ngày. Các thông tin có thể được mọi người truy cập, đăng tải lên mạng Internet dưới rất nhiều dạng hình thức, nội dung và đa dạng về mục đích, thể loại. Khi lượng dữ liệu sinh ra và cần xử lí trở nên quá lớn như vậy, các công nghệ truyền thống không còn có thể đáp ứng được khả năng xử lí trong thời gian yêu cầu nữa. Vậy nên các công nghệ xử lí Big Data ra đời nhằm phục vụ việc xử lí dữ liệu nhanh, đúng đắn và trong thời gian cho phép. Các công nghệ Big Data đươc phát triển từ các mô hình tính toán cổ điển nhưng thực hiện theo phương pháp phân tán. Tức là tập dữ liệu rất lớn ban đầu sẽ được chia nhỏ để xử lí đồng thời trên nhiều máy chủ. Điều này sẽ khiến việc xử lí dữ liệu có thể hoàn thành trong thời gian yêu cầu.

Chuyên đề của tôi sẽ thực hiện xây thực một cơ sở dữ liệu cho một phần mềm quản lí bán hàng và thiết kế một mô hinh xử lí dữ liệu theo tiêu chuẩn hiện nay để có thể từ các dữ liệu giao dịch của phần mềm quản lí bán hàng đó, qua quá trình xử lí, làm giàu thông tin sẽ đưa ra đươc các thông tin có ích cho người dùng.

## 1.2 Hướng giải quyết

**Hình 1.2.1:** Định hướng giải quyết

## 1.3 Chi tiết các bước giải quyết vấn đề

### 1.3.1 Thu thập dữ liệu giao dịch từ nguồn

Các dữ liệu tạo ra từ giao dịch cần được thu thập để đưa vào hệ thống xử lí. Việc thu thập cần đảm bảo các yêu cầu sau:

* **Thu thập dữ liệu nhanh:** dữ liệu có thể được lấy theo các mô hình batch, mini-batch, streaming sao cho đáp ứng nhu cầu nghiệp vụ của báo cáo
* **Thu thập dữ liệu chính xác:** dữ liệu khi được đưa vào quá trình xử lí không được xảy ra mất mát, sai lệch. Cần đảm bảo đúng về số lượng cũng như tính đúng đắn và toàn vẹn.
* **Không ảnh hưởng tiêu cực đến cơ sở dữ liệu nguồn:** cơ sở dữ liệu nguồn phục vụ cho quá trình xử lí giao dịch của khách hàng, việc lấy dữ liệu để xử lí không được gây ra khối lượng công việc quá lớn, không được chiếm dụng nhiều tài nguyên của cơ sở dữ liệu nguồn.

### 1.3.2 Xử lí và tính toán dữ liệu trong môi trường phân tán

Dữ liệu sau khi được thu thập sẽ được đưa vào xử lí trong môi trường phân tán, quá trình tính toán này cần được tối ưu sao cho sử dụng tối ưu lượng tài nguyên ít nhất và tốn ít thời gian nhất. Dữ liệu sau khi được tính toán cần có giá trị tái sử dụng và có thể dụng được cho nhiều nghiệp vụ khác nhau để tránh lãng phí không gian lưu trữ.

### 1.3.3 Xây dựng các báo cáo để đưa ra thông tin có ích

Sau khi đã tính toán xong và đưa ra môi trường phục vụ cho việc truy vấn, lúc này sẽ cần xây dựng các báo cáo, biểu đồ để giúp đọc dữ liệu dễ dàng hơn cho người dùng. Các báo cáo, truy vấn này sẽ được thực hiện ở một môi trường riêng tách biệt với môi trường tính toán dữ liệu do đặc thù công nghệ. Môi trường phục vụ báo cáo cần đáp ứng đươc nhiều truy cập đồng thời cùng lúc cũng như thời gian phản hồi nhanh.

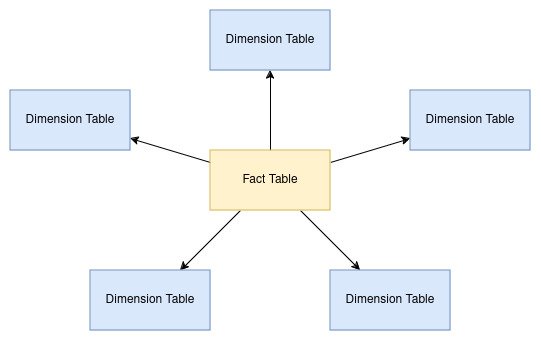
# Chương 2. Cơ sở lí thuyết

## 2.1 Mô hình Snow Flake và Star Schema

**Mô hình Star schema** và **mô hình Snowflake** là hai dạng thiết kế mô hình dữ liệu phổ biến hiện nay được sử dụng trong thiết kế kho chứa dữ liệu (data warehouse). Cả hai đều có những ưu và nhược điểm riêng, phù hợp với những trường hợp sử dụng khác nhau.

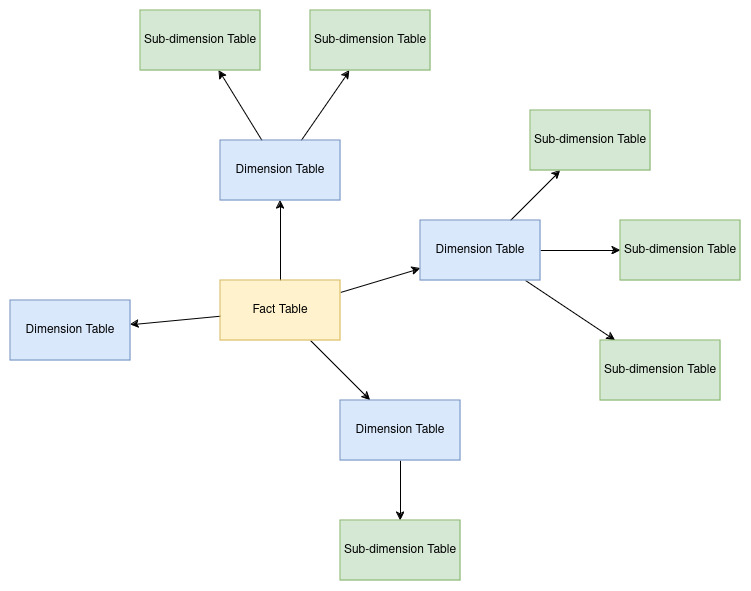
**Mô hình Star schema** được ví như một ngôi sao với bảng **thông tin** (fact table) ở trung tâm và các bảng **thứ nguyên** (dimension table) liên kết với nó như các cánh của ngôi sao, do đó có tên gọi là Star schema.

* **Ưu điểm:**
  + Thiết kế đơn giản, dễ hiểu và dễ truy vấn.
  + Mang lại hiệu năng tốt cho việc truy vấn truy xuất dữ liệu đơn giản và báo cáo tổng hợp.
* **Nhược điểm:**
  + Dữ liệu có thể bị trùng lặp, dẫn đến lãng phí dung lượng lưu trữ.
  + Khó khăn trong việc mở rộng mô hình khi thêm dữ liệu mới hoặc thay đổi cấu trúc dữ liệu.
  + Không hiệu quả cho các truy vấn phức tạp đòi hỏi nhiều phép nối bảng do thiết kế mô hình quá đơn giản.

**Hình 2.1.1:** Mô hình Star Schema

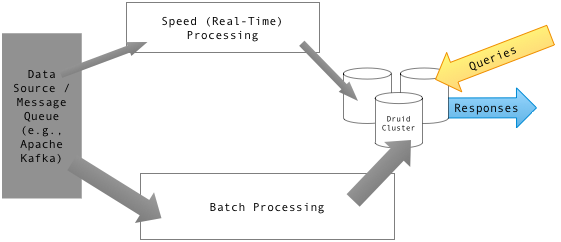
**Mô hình Snowflake** là mô hình thiết kế phức tạp và chuẩn hóa hơn so với mô hình **Star schema**. Trong mô hình Snowflake, các bảng thứ nguyên được chia thành các bảng con theo cấp bậc, giúp giảm bớt sự trùng lặp dữ liệu. Có thể nói Snow Flake là phiên bản phức tạp hơn của Star Schema, do đó sẽ có những đánh đổi nhất định.

* **Ưu điểm:**
  + Tối ưu dung lượng lưu trữ.
  + Dễ dàng trong việc mở rộng, cập nhật, thay đổi mô hình do các bảng thứ nguyên nhỏ gọn hơn.
  + Hỗ trợ rất tốt cho các truy vấn phân tích phức tạp.
* **Nhược điểm:**
  + Phức tạp, khó hiểu hơn, sẽ tạo ra nhiều bảng nhỏ.
  + Yêu cầu thực hiện nhiều phép JOIN bảng hơn để lấy ra được dữ liệu yêu cầu, do đó các lệnh truy vấn sẽ phức tạp hơn và có thể được xử lí chậm hơn.



**Hình 2.1.2:** Mô hình Snow Flake

## 2.2 Lambda Architecture

**Hình 2.2.1:** Kiến trúc Lambda cho việc đồng bộ dữ liệu thời gian thực

**Lambda Architecture** là một kiến trúc phần mềm được đề xuất bởi Nathan Marz vào năm 2011, với mục đích tối ưu việc xử lý dữ liệu lớn và có thể đáp ứng được nhu cầu cả xử lý dữ liệu theo thời gian thực. Kiến trúc này tổng hợp hai phương pháp xử lý dữ liệu: batch processing (xử lý theo lô) và stream processing (xử lý theo luồng).

Các thành phần chính của Lambda Architecture bao gồm:

* **Batch Layer (Tầng xử lí dữ liệu theo lô)**:
  + Xử lý dữ liệu theo các lô (batch) lớn, với tần suất không cao, thường là 1 ngày 1 lần.
  + Việc tính toán sẽ được thực hiện trên toàn bộ tập dữ liệu để đảm bảo dữ liệu sẽ được cập nhật đầy đủ, không bị bỏ lỡ bất cứ thời điểm nào.
* **Speed Layer (Tầng xử lý dữ liệu thời gian thực)**:
  + Đồng bộ, cập nhật và xử lí dữ liệu theo thời gian thực (real-time).
  + Cập nhật dữ liệu theo các luồng nhỏ để có hiệu năng tốt hơn.
* **Serving Layer (Lớp phục vụ)**:
  + Lưu trữ dữ liệu ở dạng cuối và phục vụ dữ liệu cho ứng dụng người dùng.
  + Có thể sử dụng các công nghệ tối ưu cho việc truy vấn như ClickHouse để đem lại hiệu năng truy vấn tốt nhất.

Nguồn: [Wikipedia](https://en.wikipedia.org/wiki/Lambda_architecture#/media/File:Diagram_of_Lambda_Architecture_(Druid_data_store).png)

## 2.3 Microsoft SQL Server 2022

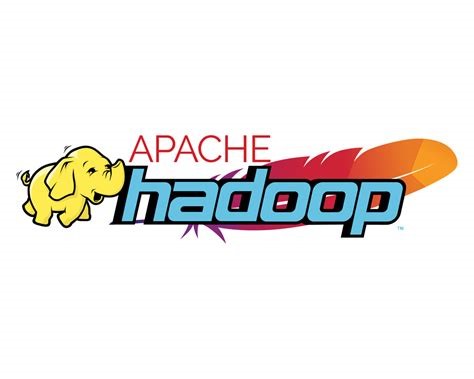
**Hình 2.3.1:** Microsoft SQL Server 2022

SQL Server 2022 hiện tại là phiên bản mới nhất của hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) của Microsoft. Phiên bản này mang đến cho người dùng nhiều cải tiến và rất nhiều các tính năng mới nhằm nâng cao hiệu suất, khả năng mở rộng, bảo mật, và khả năng phân tích dữ liệu.

Các thành phần chính của SQL Server 2022 gồm có:

* **Database Engine:** Trình nền Cơ sở dữ liệu là dịch vụ cốt lõi để lưu trữ, xử lý và bảo mật dữ liệu. Database Engine cung cấp quyền truy cập được kiểm soát và xử lý giao dịch để đáp ứng các yêu cầu của các ứng dụng sử dụng dữ liệu khắt khe nhất trong doanh nghiệp của bạn.
* **Machine Learning Services (MLS):** Machine Learning Services hỗ trợ tích hợp học máy vào quy trình làm việc của doanh nghiệp với việc sử dụng các ngôn ngữ phổ biến nhất hiện nay như R và Python.
* **Integration Services (SSIS):** SQL Server Integration Services là một nền tảng để xây dựng các giải pháp tích hợp dữ liệu, bao gồm các thư viện hỗ trợ việc trích xuất, biến đổi và tải dữ liệu cho việc xây dựng kho chứa dữ liệu.
* **Analysis Services (SSAS):** SQL Server Analysis Services là một nền tảng phân tích dữ liệu cung cấp các công cụ cho cá nhân, đội nhóm để thực hiện các phân tích báo cáo thông minh. SSAS có thể tương thích với các giải pháp OLAP truyền thống hoặc các ứng dụng như Power Pivot, Excel, SharePoint. SSAS còn bao gồm các công cụ khai phá dữ liệu để giúp bạn tìm ra các mối liên hệ được có thể có trong một lượng dữ liệu rất lớn.
* **Reporting Services (SSRS):** SQL Server Reporting Services cung cấp các tính năng xây dựng báo cáo cho doanh nghiệp. Bạn có thể tạo báo cáo từ nhiều nguồn dữ liệu, đăng tải báo cáo dưới nhiều dạng thức và bảo mật tập trung cho những người được phép theo dõi.
* **Replication:** SQL Server Replication là một nhóm các công cụ để sao chép và phân tán dữ liệu và các thực thể trong cơ sở dữ liệu từ nơi này đến nơi khác, hoặc thực hiện việc đồng bộ giữa các cơ sở dữ liệu để đảm bảo tính nhất quán.
* **Data Quality Services (DQS):** Data Quality Services cung cấp cho bạn một giải pháp làm sạch dữ liệu. DQS cho phép bạn xây dựng một cơ sở kiến thức, mà có thể dựa vào đó để thực hiện các phương pháp chuẩn hóa dữ liệu.

## 2.4 Apache Hadoop

**Hình 2.4.1:** Apache Hadoop

Apache Hadoop là công nghệ mã nguồn mở, hiện tại đang được sử dụng tại các công ty có trung tâm dữ liệu lớn để lưu trữ và xử lý các tập dữ liệu lớn một cách hiệu quả. Hadoop được thiết kế và tối ưu để mở rộng phạm vi tính toán từ một máy chủ độc lập sang rất nhiều máy chủ khác để đạt được hiệu năng tính toán song song và tận dụng tối đa tài nguyên có thể.

### Các thành phần chính

* **Hadoop Distributed Filesystem** - Hệ thống tệp phân tán Hadoop (HDFS): Cung cấp các giao diện lập trình và khả năng lưu trữ, đồng bộ trên rất nhiều máy chủ cùng một lúc.
* **Mô hình tính toán MapReduce**: Là giao diện lập trình cho phép thực hiện tính toán phân tán trên các tập tin có sẵn trong HDFS. MapReduce có thể thực hiện việc tính toán trên một tập dữ liệu rất lớn và không bị giới hạn bởi tài nguyên khả dụng của một máy chủ cụ thể nào cả.

### Ưu điểm của Apache Hadoop

* **Khả năng mở rộng:** Hadoop có thể được dễ dàng mở rộng bằng cách mở rộng theo chiều ngang, chỉ cần thêm máy chủ và khai báo với máy quản lí là có thể thêm được tài nguyên máy vào cụm Hadoop.
* **Hiệu năng xử lí tính toán rất cao:** Hadoop có thể xử lí tính toán các tập dữ liệu lớn một cách hiệu quả nhờ vào cơ chế phân tán dữ liệu ra các máy và tổng hợp kết quả lại sau cùng.
* **Đa dụng, hỗ trợ nhiều định dạng:** Hadoop hỗ trợ nhiều định dạng file và có thể được tích hợp với rất nhiều hệ thống hiện nay.
* **Chi phí sử dụng thấp:** Hadoop có thể chạy tốt trên cả phần cứng giá rẻ, cũ lẫn cả những phần cứng tốt nhất hiện nay. Một cụm Hadoop có thể được sử dụng trên một vài hoặc rất nhiều máy.

## 2.5 Apache Spark

**Hình 2.5.1:** Apache Spark

Apache Spark là một công nghệ mã nguồn mở sử dụng cho việc tính toán phân tán. Spark được thiết kế và tối ưu để xử lý dữ liệu lớn một cách rất nhanh chóng và an toàn. Spark sử dụng Hadoop làm hệ thống tệp và có thể tận dụng việc đọc, ghi trên môi trường phân tán đã được tối ưu để cho ra tốc độ xử lí rất nhanh.

### Ưu điểm của Apache Spark

* **Hiệu năng tính toán phân tán cao:** Spark có thể xử lý dữ liệu nhanh hơn MapReduce tới 100 lần.
* **Dễ dàng sử dụng và tích hợp:** Spark cung cấp giao diện lập trình đơn giản và dễ sử dụng bằng Scala, Python, Java, SQL và R.
* **Đa dụng trong việc xây dựng các ứng dụng:** Apache Spark có thể được sử dụng cho nhiều loại ứng dụng xử lý dữ liệu lớn: phân tích dữ liệu theo lô, xây dựng mô hình học máy và xử lí dữ liệu theo thời gian thực.
* **Khả năng mở rộng dễ dàng:** cũng như Hadoop, Spark có thể mở rộng cụm bằng cách mở rộng theo chiều ngang, dễ dàng và an toàn.

## 2.6 Apache Airflow

**Hình 2.6.1:** Apache Airflow

Apache Airflow là công nghệ nguồn mở dùng để lên lịch trình, quản lý và theo dõi các tác vụ hoặc luồng xử lý dữ liệu. Airflow hiện đang được sử dụng rộng rãi trong việc lên lịch, theo dõi các hệ thống xử lý dữ liệu lớn để tự động hóa các quy trình có tính lặp lại, thay vì phải thao tác thủ công dễ gây ra lỗi.

### Ưu điểm của Apache Airflow

* **Dễ dàng sử dụng và tích hợp:** Airflow cung cấp giao diện người dùng web có thể dũy dàng truy cập, đơn giản trong quá trình sử dụng và trực quan, độ trễ thấp để người dùng có thể theo dõi, vận hành các quy trình xử lý dữ liệu.
* **Dễ dàng mở rộng:** Apache Airflow cũng hỗ trợ khả năng mở rộng theo chiều ngang, dễ dàng thêm máy chủ khác vào cụm.
* **Linh hoạt và dễ dàng tích hợp:** Airflow hỗ trợ việc tích hợp với nhiều hệ thống và công cụ khác nhau một cách rất dễ dàng.

## 2.7 ClickHouse

  
**Hình 2.7.1:** ClickHouse

ClickHouse là hệ thống quản trị cơ sở dữ liệu (DBMS) mã nguồn mở, dạng cột, được thiết kế và tối ưu cho các truy vấn phân tích (OLAP) và khả năng tính toán trên môi trường phân tán. ClickHouse có nguồn gốc được phát triển bởi Yandex và hiện nay đang được sử dụng rộng rãi để phục vụ việc truy vấn nhanh.

### Ưu điểm nổi bật của ClickHouse

* **Hiệu năng rất cao:** ClickHouse là một trong những cơ sở dữ liệu tối ưu cho việc truy vấn nhanh nhất hiện nay, có thể xử lý truy vấn hàng tỷ bản ghi mỗi giây trên ngay cả những phần cứng thông thường.
* **Dễ dàng mở rộng:** ClickHouse cũng hỗ trợ việc mở rộng theo chiều ngang để gia tăng hiệu năng xử lí theo tuyến tính.
* **Dễ dàng cho người sử dụng:** ClickHouse cũng hỗ trợ truy vấn bằng ngôn ngữ SQL giống như các cơ sở dữ liệu truyền thống
* **Hỗ trợ nhiều định dạng:** ClickHouse hỗ trợ nhiều định dạng dữ liệu tệp nhau, ví dụ như: CSV, JSON, Avro và Parquet.
* **Đa dạng cho các nhu cầu sử dụng:** ClickHouse hiện tại đang có phiên bản mã nguồn mở cho mọi người sử dụng và cả phiên bản thương mại chạy trên môi trường điện toán đám mây.

## 2.8 Apache Superset

**Hình 2.8.1:** Apache Superset

Apache Superset là công nghệ mã nguồn mở. Superset được sử dụng để giúp phân tích dữ liệu, trực quan hóa và khám phá dữ liệu trong thời gian rất ngắn. Superset cung cấp giao diện người dùng web trực quan cho phép người dùng truy vấn, phân tích và trực quan hóa dữ liệu từ nhiều nguồn khác nhau, bao gồm cơ sở dữ liệu quan hệ như SQL Server, các hệ thống kho dữ liệu phân tán như Apache Hive, Apache Spark hoặc các cơ sỡ dữ liệu OLAP như ClickHouse

### Ưu điểm của Apache Superset

* **Dễ sử dụng và nhanh chóng:** Superset có hỗ trợ truy cập qua giao diện web, cho phép người dùng dễ dàng truy cập, truy vấn, phân tích và trực quan hóa dữ liệu và có thể chia sẻ cho những người khác qua giao diện web mà không cần biết kiến thức lập trình.
* **Linh hoạt và nhiều tính năng:** Superset hỗ trợ việc truy cập tới nhiều nguồn dữ liệu khác nhau, Superset cũng hỗ trợ rất nhiều loại biểu đồ để đáp ứng đủ hầu hết các nhu cầu sử dụng.

## 2.9 Flask

  
**Hình 2.9.1:** Flask

Python Flask là một khung làm việc (framework) được sử dụng để viết các ứng dụng web với Python. Flask rất dễ dàng cài đặt và nhẹ trong việc xử lí. Flask có thể được sử dụng trong các ứng dụng đơn giản với độ phức tạp không quá cao.

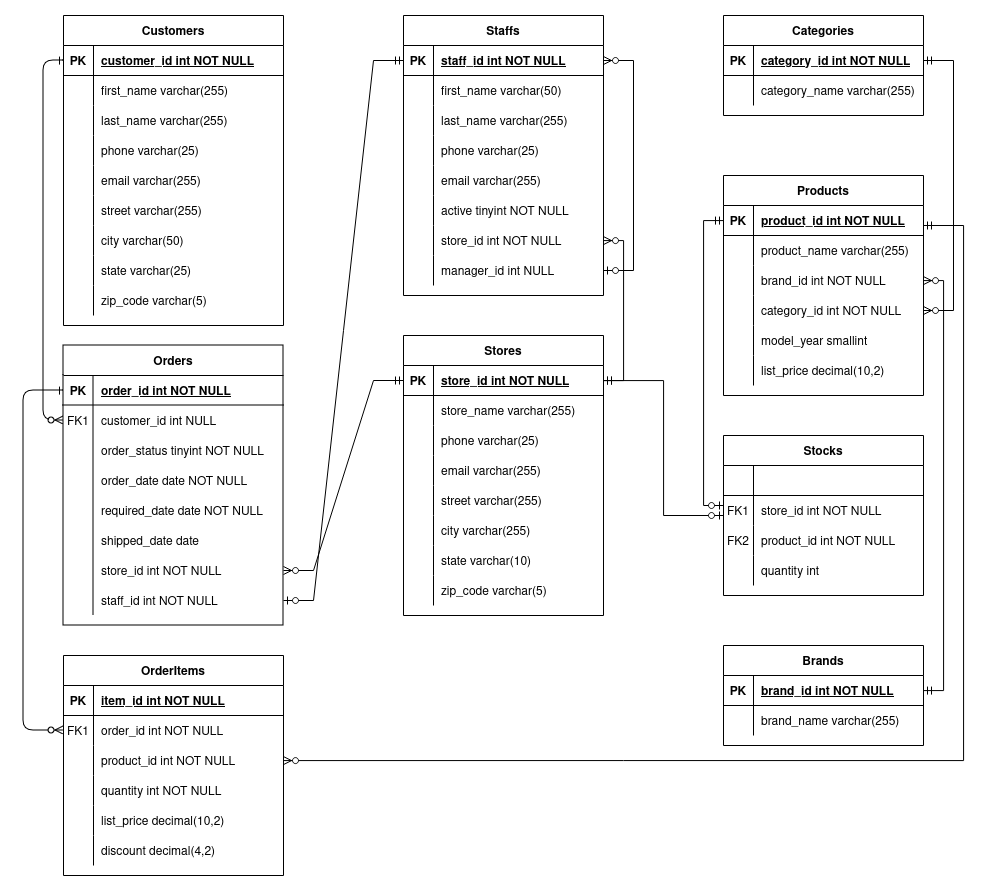
## 2.10 Pandas

Pandas là một thư viện Python cung cấp khả năng xử lí dữ liệu nhanh, linh hoạt giúp việc thao tác với dữ liệu quan hệ dạng bảng dễ dàng và nhanh chóng hơn. Pandas hướng tới việc trở thành công cụ mạnh mẽ và đa dụng nhất trong việc thao tác với dữ liệu.

# Chương 3. Thiết kế cơ sở dữ liệu

Dưới đây là mô hình cơ sở dữ liệu phục vụ cho một phần mềm quản lí bán hàng. Phần mềm có các chức năng chính:

* Quản lí sản phẩm, tồn kho
* Quản lí đơn hàng, đặt hàng
* Quản lí thông tin khách hàng
* Quản lí thông tin các cửa hàng
* Quản lí nhân viên của cửa hàng

**Hình 3:** Mô hình cơ sở dữ liệu BikeStores

## 3.1 Danh sách các bảng

### 3.1.1 Bảng Customers

Bảng Customers lưu trữ thông tin khách hàng mua hàng

|  |  |  |  |
| --- | --- | --- | --- |
| **Tên trường** | **Kiểu dữ liệu** | **Giải thích** | **Cho phép null** |
| customer\_id | INT | Mã khách hàng | Không |
| first\_name | varchar(255) | Tên khách hàng | Có |
| last\_name | varchar(255) | Họ khách hàng | Có |
| phone | varchar(25) | Số điện thoại | Có |
| email | varchar(255) | Email | Có |
| street | varchar(255) | Tên đường | Có |
| city | varchar(50) | Tên thành phố | Có |
| state | varchar(25) | Tên bang | Có |
| zip\_code | varchar(5) | Mã bưu chính | Có |

### 3.1.2 Bảng Staffs

Bảng Staffs lưu trữ thông tin nhân viên của các cửa hàng

|  |  |  |  |
| --- | --- | --- | --- |
| **Tên trường** | **Kiểu dữ liệu** | **Giải thích** | **Cho phép null** |
| staff\_id | INT | Mã khách hàng | Không |
| first\_name | varchar(255) | Tên khách hàng | Không |
| last\_name | varchar(255) | Họ khách hàng | Không |
| phone | varchar(25) | Số điện thoại | Có |
| email | varchar(255) | Email | Không |
| active | tinyint | Đang hoạt động | Có |
| store\_id | INT | Mã cửa hàng | Có |
| manager\_id | INT | Mã quản lí | Có |

### 3.1.3 Bảng Stores

Bảng Stores lưu trữ thông tin về các cửa hàng

|  |  |  |  |
| --- | --- | --- | --- |
| **Tên trường** | **Kiểu dữ liệu** | **Giải thích** | **Cho phép null** |
| store\_id | INT | Mã cửa hàng | Không |
| store\_name | varchar(255) | Tên cửa hàng | Có |
| phone | varchar(255) | Số điện thoại | Có |
| email | varchar(255) | Email | Có |
| street | varchar(255) | Tên đường | Có |
| city | varchar(255) | Tên thành phố | Có |
| state | varchar(10) | Tên bang | Có |
| zip\_code | varchar(5) | Mã bưu chính | Có |

### 3.1.4 Bảng Orders

Bảng Orders lưu trữ thông tin các đơn hàng

|  |  |  |  |
| --- | --- | --- | --- |
| **Tên trường** | **Kiểu dữ liệu** | **Giải thích** | **Cho phép null** |
| order\_id | INT | Mã đơn hàng | Không |
| customer\_id | INT | Mã khách hàng | Không |
| order\_status | tinyint | Trạng thái đơn | Không |
| order\_date | date | Ngày đặt hàng | Không |
| required\_date | date | Ngày yêu cầu giao | Không |
| shipped\_date | date | Ngày giao hàng | Có |
| store\_id | INT | Mã cửa hàng | Có |
| staff\_id | INT | Mã nhân viên | Có |

### 3.1.5 Bảng OrderItems

Bảng OrderItems lưu trữ thông tin các sản phẩm trong đơn hàng

|  |  |  |  |
| --- | --- | --- | --- |
| **Tên trường** | **Kiểu dữ liệu** | **Giải thích** | **Cho phép null** |
| order\_id | INT | Mã đơn hàng | Không |
| item\_id | INT | Mã vật phẩm | Không |
| product\_id | INT | Mã sản phẩm | Không |
| quantity | INT | Số lượng | Không |
| list\_price | decimal(10,2) | Đơn giá | Không |
| discount | decimal(4,2) | Giảm giá | Có |

### 3.1.5 Bảng Categories

Bảng Categories lưu trữ thông tin danh mục của hàng hóa

|  |  |  |  |
| --- | --- | --- | --- |
| **Tên trường** | **Kiểu dữ liệu** | **Giải thích** | **Cho phép null** |
| category\_id | INT | Mã danh mục | Không |
| category\_name | varchar(255) | Tên danh mục | Có |

### 3.1.6 Bảng Brands

Bảng Brands lưu trữ thông tin thương hiệu của các sản phẩm

|  |  |  |  |
| --- | --- | --- | --- |
| **Tên trường** | **Kiểu dữ liệu** | **Giải thích** | **Cho phép null** |
| brand\_id | INT | Mã thương hiệu | Không |
| brand\_name | varchar(255) | Tên thương hiệu | Có |

### 3.1.7 Bảng Stocks

Bảng Stocks lưu trữ thông tin tồn kho của các sản phẩm tại cửa hàng

|  |  |  |  |
| --- | --- | --- | --- |
| **Tên trường** | **Kiểu dữ liệu** | **Giải thích** | **Cho phép null** |
| store\_id | INT | Mã cửa hàng | Không |
| product\_id | INT | Mã sản phẩm | Không |
| quantity | INT | Số lượng tồn kho | Có |

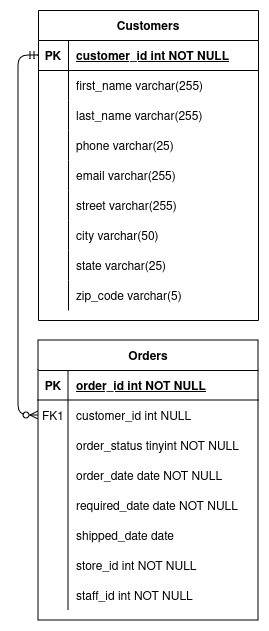
### 3.1.8 Bảng Products

Bảng Products lưu trữ thông tin sản phẩm

|  |  |  |  |
| --- | --- | --- | --- |
| **Tên trường** | **Kiểu dữ liệu** | **Giải thích** | **Cho phép null** |
| product\_id | INT | Mã sản phẩm | Không |
| product\_name | varchar(255) | Số lượng tồn kho | Có |
| brand\_id | INT | Mã thương hiệu | Không |
| category\_id | INT | Mã danh mục | Không |
| model\_year | smallint | Mã năm | Có |
| list\_price | decimal(10,2) | Đơn giá | Có |

## 3.2 Giải thích các quan hệ trong thiết kế cơ sở dữ liệu

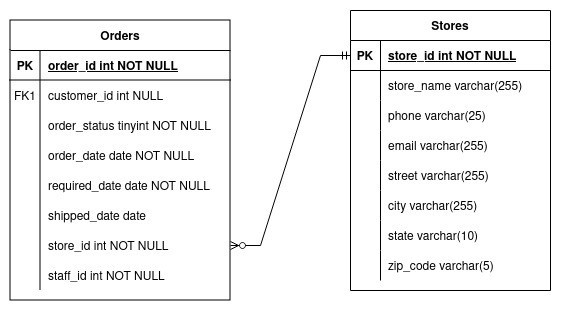
### 3.2.1 Quan hệ Khách hàng - Đơn hàng

  
**Hình 3.2.1:** Quan hệ Khách hàng - Đơn hàng

Kiểu quan hệ: một (bắt buộc) - nhiều (tùy chọn)

Giải nghĩa: khi khách hàng thực hiện mua hàng sẽ sinh ra một bản ghi trong bảng Orders. Một khách hàng có thể mua nhiều đơn hàng hoặc cũng có thể không mua đơn hàng nào. Nhưng một đơn hàng chỉ có thể được mua bởi một khách hàng duy nhất.

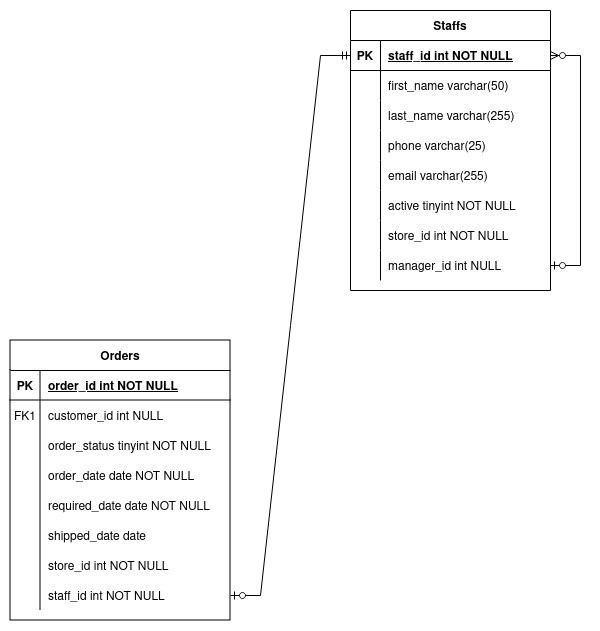
### 3.2.2 Quan hệ Cửa hàng - Đơn hàng

  
**Hình 3.2.2:** Quan hệ Cửa hàng - Đơn hàng

Kiểu quan hệ: một (bắt buộc) - nhiều (tùy chọn)

Giải nghĩa: một đơn hàng khi được mua phải có thông tin là được mua tại cửa hàng nào. Mỗi đơn hàng chỉ có thể được mua tại một cửa hàng duy nhất còn một cửa hàng có thể bán được nhiều đơn hàng, hoặc cũng có thể không bán được đơn hàng nào.

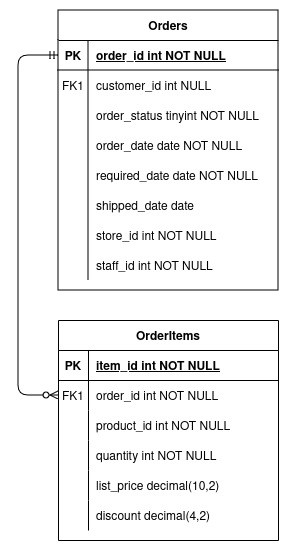
### 3.2.3 Quan hệ Nhân viên - Đơn hàng

  
**Hình 3.2.3:** Quan hệ Nhân viên - Đơn hàng

Kiểu quan hệ: một (bắt buộc) - nhiều (tùy chọn)

Giải nghĩa: đơn hàng cần được bán bởi nhân viên, mỗi đơn hàng chỉ được bán bởi một nhân viên duy nhất. Mỗi nhân viên có thể bán được nhiều đơn hàng hoặc cũng có thể không bán được đơn hàng nào cả.

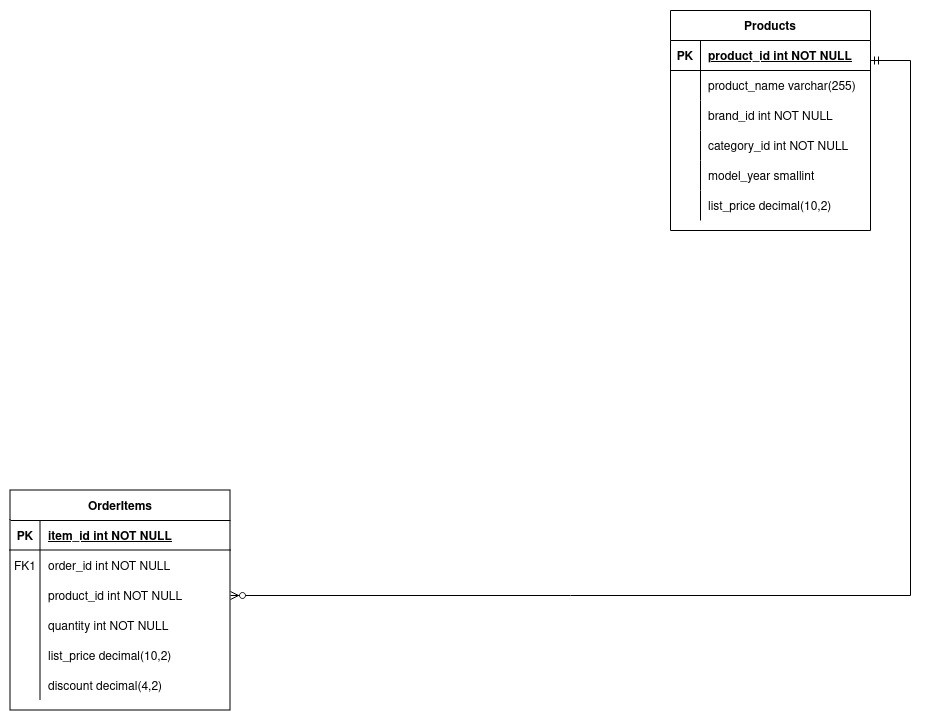
### 3.2.4 Quan hệ Đơn hàng - Thành phần đơn hàng

  
**Hình 3.2.4:** Quan hệ Đơn hàng - Thành phần đơn hàng

Kiểu quan hệ: một (bắt buộc) - nhiều (tùy chọn)

Giải nghĩa: mỗi đơn hàng có thể chứa nhiều sản phẩm được bán trong đơn hàng đó, và mỗi món đồ trong đơn hàng chỉ thuộc về một đơn hàng duy nhất.

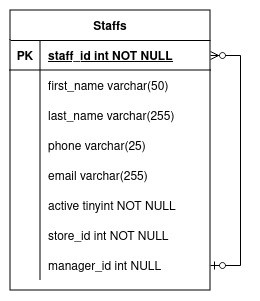
### 3.2.5 Quan hệ Sản phẩm - Thành phần đơn hàng

**Hình 3.2.5:** Quan hệ Sản phẩm - Thành phần đơn hàng

Kiểu quan hệ: một (bắt buộc) - nhiều (tùy chọn)

Giải nghĩa: mỗi sản phẩm chỉ xuất hiện một lần trong một đơn hàng dưới dạng thành phần của đơn hàng đó. Mỗi sản phẩm thì lại có thể xuất hiện trong nhiều đơn hàng. Một sản phẩm cũng có thể không xuất hiện trong đơn hàng nào cả.

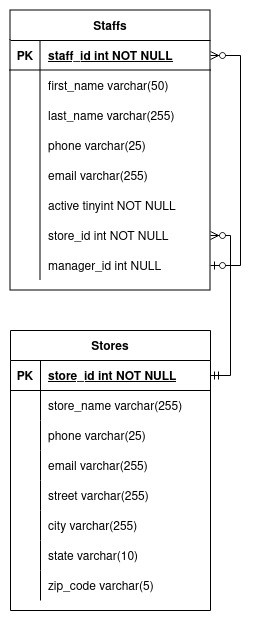
### 3.2.6 Quan hệ Nhân viên - Quản lí

  
**Hình 3.2.6:** Quan hệ Nhân viên - Quản lí

Kiểu quan hệ: một (tùy chọn) - nhiều (tùy chọn)

Giải nghĩa: một nhân viên có thể là quản lí hoặc không, và một nhân viên có thể không dưới sự quản lí của nhân viên nào khác. Quản lí cũng được coi là một nhân viên nên sẽ được lưu chung vào bảng Nhân viên

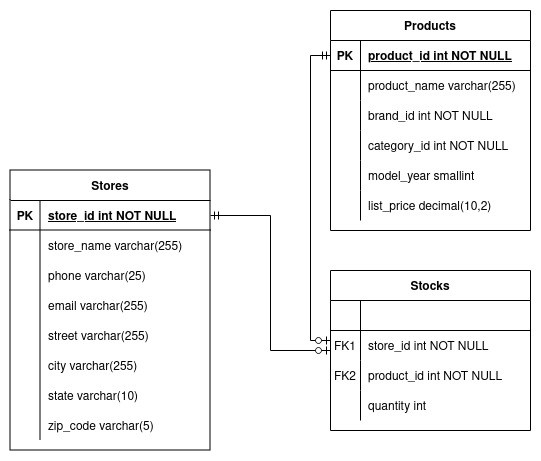
### 3.2.7 Quan hệ Nhân viên - Cửa hàng

  
**Hình 3.2.7:** Quan hệ Nhân viên - Cửa hàng

Kiểu quan hệ: một (bắt buộc) - nhiều (tùy chọn)

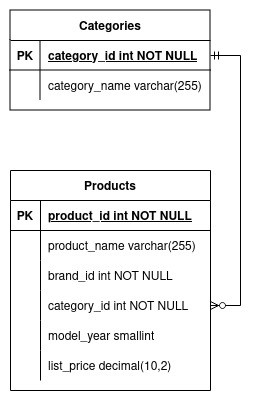
Giải nghĩa: một nhân viên phải thuộc về một cửa hàng nào đó, còn cửa hàng có thể có nhiều nhân viên hoặc không có cửa hàng nào cả, cửa hàng sẽ không có nhân viên nào trong trường hợp cửa hàng đó mới thành lập và chưa khai trương mở bán.

### 3.2.8 Quan hệ Cửa hàng - Sản phẩm - Tồn kho

  
**Hình 3.2.8:** Quan hệ Cửa hàng - Sản phẩm - Tồn kho

Giải nghĩa: Mối quan hệ giữa 3 bảng này thể hiện rằng tại cửa hàng này, sản phẩm đó đang là bao nhiêu. Về cơ bản đây là mối quan hệ nhiều - nhiều giữa Sản phẩm và Cửa hàng. Bảng Tồn kho sẽ ánh xạ tới bảng Cửa hàng và bảng Sản phẩm.

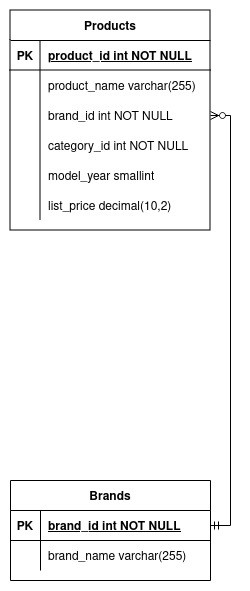
### 3.2.9 Quan hệ Sản phẩm - Chủng loại

  
**Hình 3.2.9:** Quan hệ Sản phẩm - Chủng loại

Kiểu quan hệ: một (bắt buộc) - nhiều (tùy chọn)

Giải nghĩa: mỗi sản phẩm sẽ thuộc về một chủng loại nhất định, còn một chủng loại thì có thể có nhiều sản phẩm.

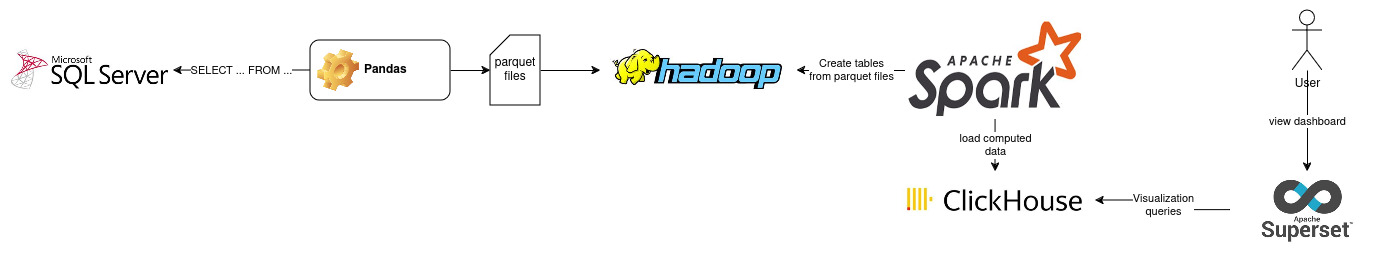
### 3.2.10 Quan hệ Sản phẩm - Nhãn hàng

  
**Hình 3.2.10:** Quan hệ Sản phẩm - Nhãn hàng

Kiểu quan hệ: một (bắt buộc) - nhiều (tùy chọn)

Giải nghĩa: mỗi sản phẩm sẽ thuộc về một nhãn hàng nhất định, còn một nhãn hàng thì có thể có nhiều sản phẩm.

# Chương 4. Xây dựng luồng xử lí dữ liệu

**Hình 4:** Mô hình luồng xử lí dữ liệu

Một luồng đồng bộ và xử lí dữ liệu gồm các bước sau:

* **Bước 1:** Đồng bộ dữ liệu từ nguồn về hệ thống tính toán báo cáo gồm có 2 thành phần sau:
  + Luồng đồng bộ theo lô hàng ngày chạy 1 lần (batch process):
    - Sử dụng Pandas để đọc dữ liệu từ SQL Server và trích xuất ra file định dạng parquet
    - Đẩy các file parquet trích xuất được vào Hadoop
    - Sử dụng Spark để tạo bảng staging từ các file này
    - Tạo bảng trong Data Warehouse từ các bảng staging này
  + Luồng đồng bộ thời gian thực (stream process):
    - Sử dụng debezium để đọc vào file transaction logs của SQL Server, cập nhật các sự kiện mới nhất và đưa vào Kafka
    - Sử dung Kafka để ghi các sự kiện thành file định dạng parquet và đẩy vào Hadoop
    - Sử dụng Spark để đọc các file ghi được từ Kafka và dùng lệnh MERGE để cập nhật các thay đổi vào bảng trong Data Warehouse
* **Bước 2:** Tính toán dữ liệu từ Data Warehouse ra các bảng fact để phục vụ cho việc đưa ra báo cáo
* **Bước 3:** Đẩy dữ liệu đã tinh toán ra ClickHouse để phục vụ các truy vấn báo cáo (speed layer)
* **Bước 4:** Tạo các báo cáo trên Superset và lấy dữ liệu từ ClickHouse để trực quan hóa cho người dùng.

## 4.1 Nhập dữ liệu từ phần mềm quản lý

Kịch bản: có một khách hàng đặt mua sản phẩm tại cửa hàng, nhân viên nhập các thông tin của khách hàng và tạo đơn hàng thành công cho khách

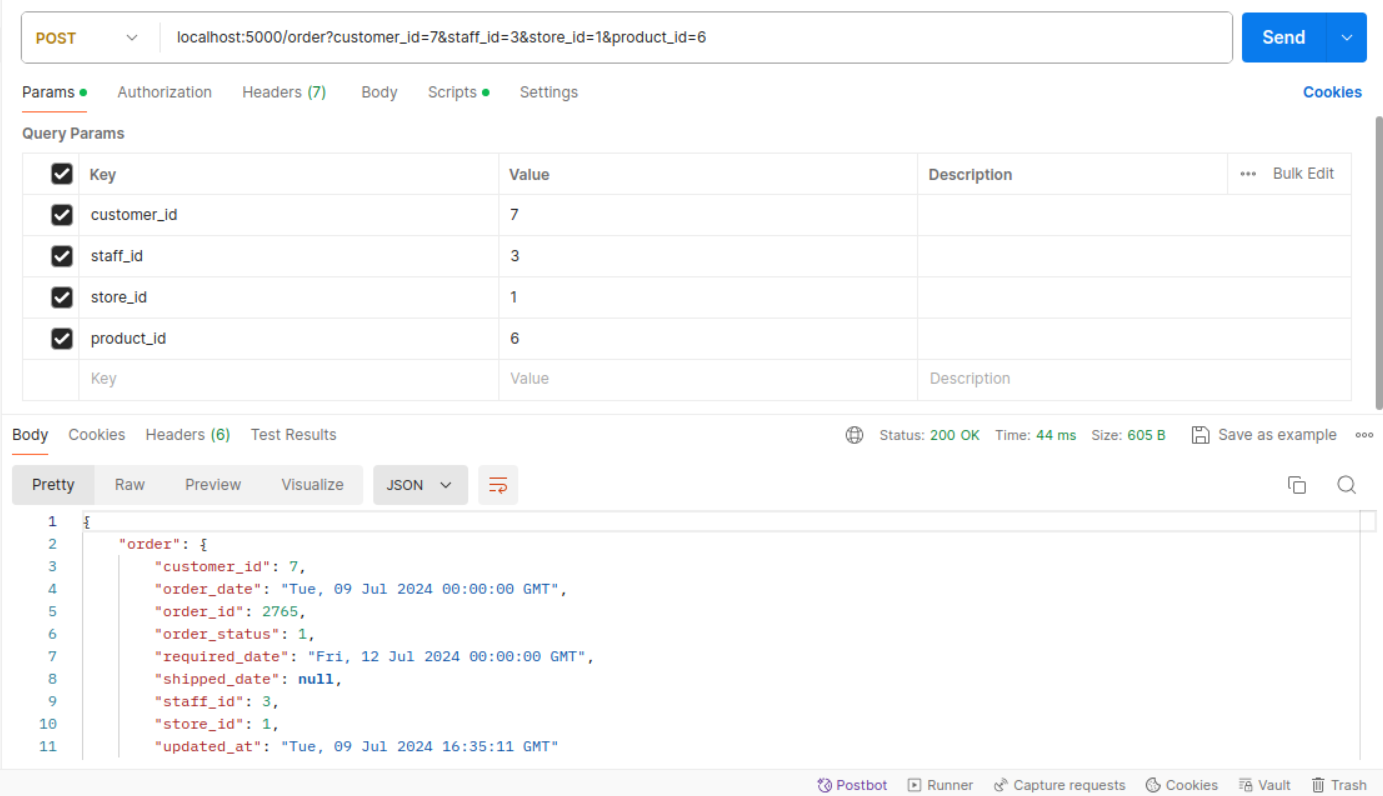
Các thông tin về đơn hàng sẽ được gửi tới server dưới dạng một yêu cầu HTTP. Nếu request được thực hiện thành công, thông tin của đơn hàng mới tạo sẽ được lưu vào cơ sở dữ liệu SQL Server.

Để tạo một đơn hàng, phần mềm khách cần gửi tới máy chủ một yêu cầu có các thông tin sau:

* Đường dẫn: localhost:5000/order
* Phương thức HTTP: POST
* Danh sách các tham số:
  + Mã khách hàng: customer\_id
  + Mã nhân viên: staff\_id
  + Mã cửa hàng: store\_id
  + Mã sản phẩm: product\_id

Nếu máy chủ phản hồi với mã 200 có nghĩa là đơn hàng đã được tạo và lưu thành công vào cơ sở dữ liệu.

Ví dụ sử dụng phần mềm Postman để gửi một request HTTP tạo đơn hàng như sau:

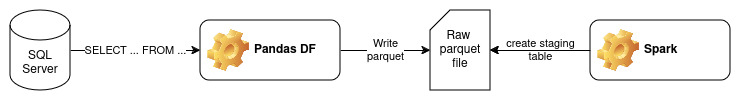
**Hình 4.1:** Tạo một đơn hàng thành công qua API

## 4.2 Đồng bộ dữ liệu từ cơ sở dữ liệu sản phẩm về hệ thống phân tích

Mục đích: Do cơ sở dữ liệu sản phẩm (Microsoft SQL Server 2022) không được tối ưu cho các truy vấn phân tích nên cần đồng bộ ra một hệ thống riêng phục vụ việc tính toán ra các bảng tính sẵn để việc truy vấn đáp ứng được hiệu năng yêu cầu.

Sau đây tôi sẽ mô tả các luồng xử lí để đồng bộ dữ liệu thô từ SQL Server về hạ tầng tính toán Spark.

### 4.2.1 Luồng đồng bộ dữ liệu theo lô



**Hình 4.2.1:** Mô hình luồng đồng bộ dữ liệu nguồn theo lô

Luồng đồng bộ theo lô sẽ trích xuất được toàn bộ dữ liệu giao dịch và đưa vào trong layer Staging để cập nhật vào trong Data Warehouse. Dữ liệu tại các bảng nguồn sẽ được đồng bộ bằng truy vấn theo từng ngày. Các dữ liệu được truy vấn sẽ được ghi ra một file có định dạng parquet, file parquet này được lưu vào HDFS. Sau đó hệ thống tính toán Spark sẽ đọc các file parquet này để tạo ra các bảng trong layer staging.

**Bước 1**: Khởi tạo các tham số và kết nối tới Microsoft SQL Server và HDFS

hdfs\_raw\_dir = f"{BASE\_PATH}{RAW}/{table}/"

dest\_file\_name = f"{table}\_01.snappy.parquet"

mssql\_conn = pymssql.connect('mssql:1433', 'sa', 'root@@@123', "BikeStores")

hdfs\_client = pyhdfs.HdfsClient(hosts='namenode:9870')

**Bước 2**: Sử dụng truy vấn để tải dữ liệu từ SQL Server vào Pandas Dataframe:

# extract raw data to parquet files

extract\_query = f"SELECT \* FROM {table};"

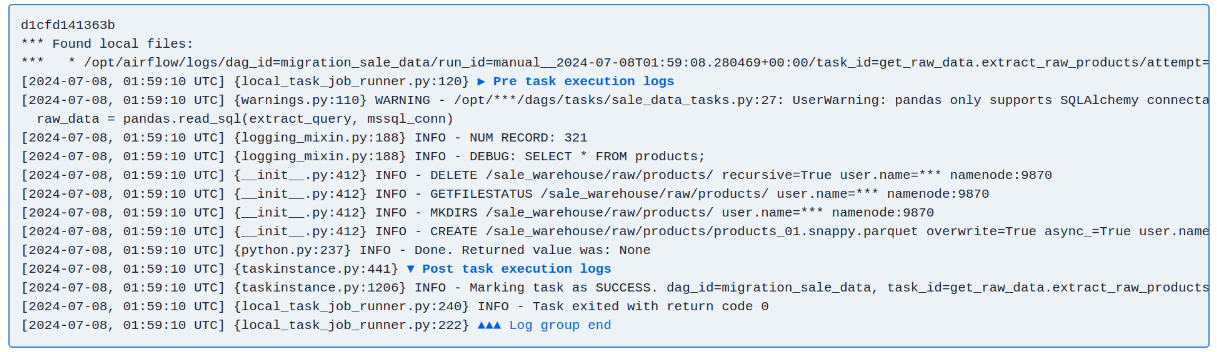
raw\_data = pandas.read\_sql(extract\_query, mssql\_conn)

print(extract\_query)

temp\_file = NamedTemporaryFile()

raw\_data.to\_parquet(temp\_file.name)

Sau khi thực thi thành công tác vụ này, trên giao diện Airflow sẽ hiển thị truy vấn được sử dụng để trích xuất dữ liệu từ nguồn và số bản ghi lấy được như sau:

**Hình 4.2.2:** Giao diện Airflow thể hiện khi chạy xong tác vụ  
trích xuất dữ liệu ra file parquet

Hình trên là ví dụ cho một tác vụ trích xuất dữ liệu từ nguồn thành công với 321 bản ghi từ bảng Products với truy vấn:

SELECT \* FROM products;

**Bước 3:** Tải file parquet vừa lấy được vào HDFS

# upload to HDFS

hdfs\_client.delete(hdfs\_raw\_dir, recursive=True)

if not hdfs\_client.exists(hdfs\_raw\_dir):

hdfs\_client.mkdirs(hdfs\_raw\_dir)

hdfs\_client.copy\_from\_local(

localsrc=temp\_file.name,

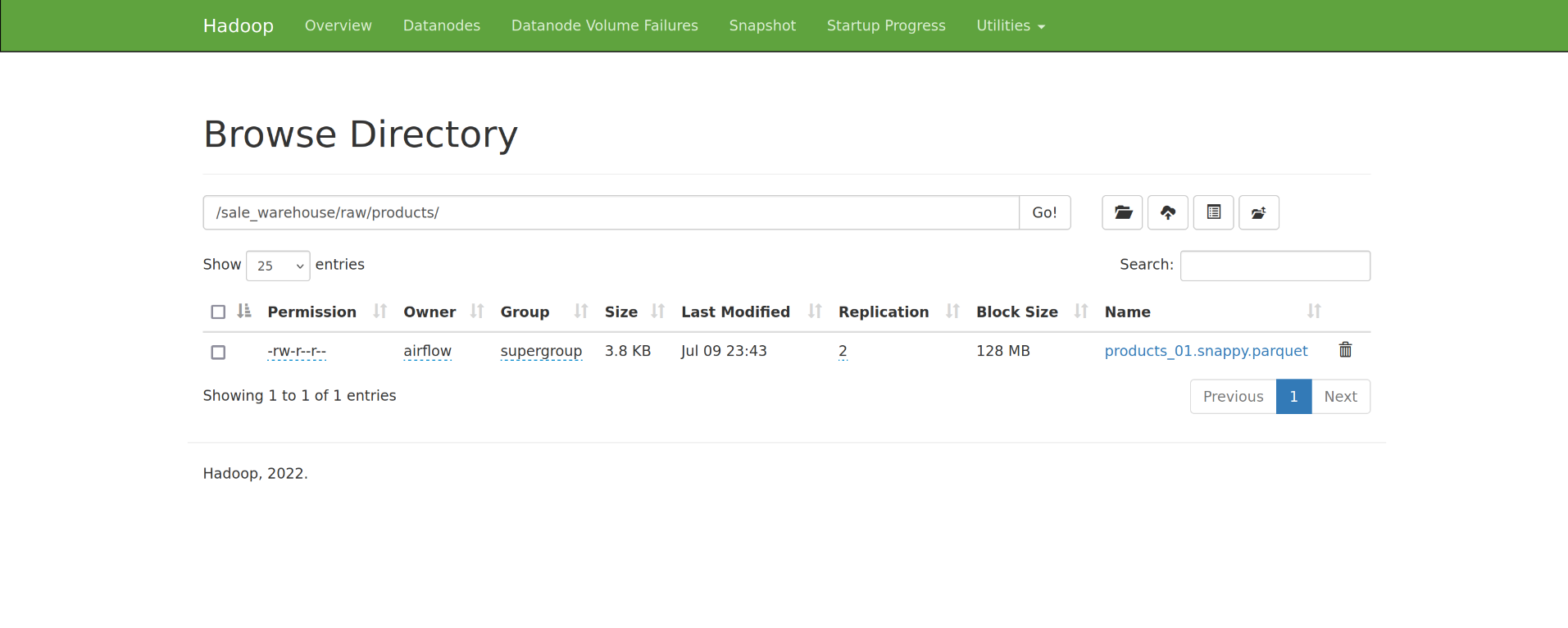
dest=hdfs\_raw\_dir + dest\_file\_name,

overwrite=True,

async\_=True

)

Sau khi thực thi xong đoạn mã trên, file parquet vừa được tạo ra sẽ được tải lên layer staging để chuẩn bị cho các bước tính toán tiếp theo, cũng chính là HDFS.

**Hình 4.2.3:** Giao diện HDFS khi file parquet chứa dữ liệu thô  
được tải lên thành công

**Bước 4**: Khởi tạo tham số và kết nối tới Spark

connection = hive.connect(host='spark-thriftserver', port=10000)

cursor = connection.cursor()

raw\_table\_name = f"default.{table}"

**Bước 5**: Tạo các bảng trong layer staging bằng SparkSQL để cập nhật các dữ liệu mới nhất

Sau khi dữ liệu thô đươc tải lên HDFS, tôi sẽ tiến hành tạo bảng trong Spark để có thể tương tác được với dữ liệu thô đó.

cursor.execute(f"""

DROP TABLE IF EXISTS {raw\_table\_name}

""")

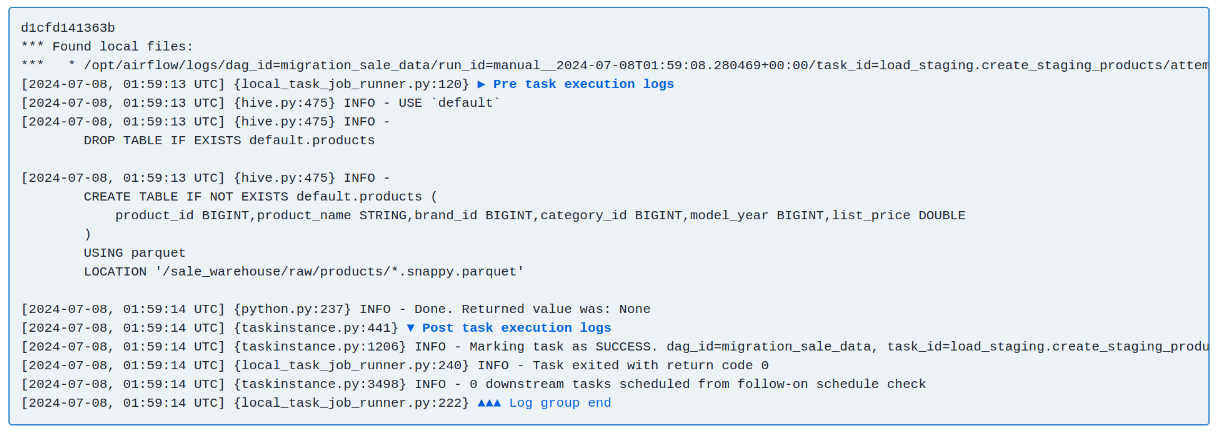
cursor.execute(f"""

CREATE TABLE IF NOT EXISTS {raw\_table\_name}

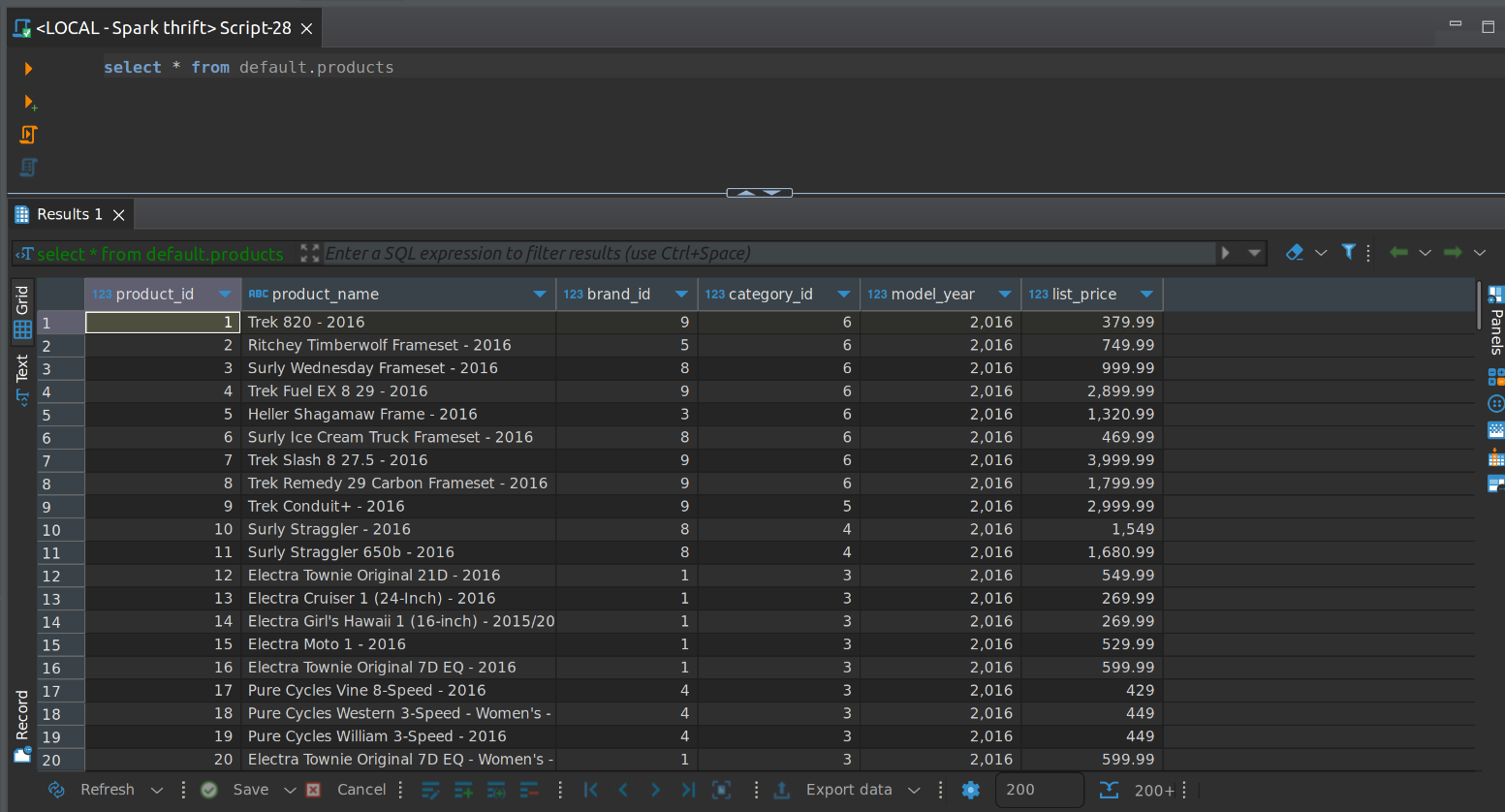
USING parquet

LOCATION '{BASE\_PATH}{RAW}/{table}/\*.snappy.parquet'

""")

**Hình 4.2.4:** Giao diện Airflow thể hiện các lệnh   
tạo bảng với SparkSQL đã thực thi thành công

Sau khi lệnh tạo bảng ở layer staging được thực hiện thành công, lúc này trong SparkSQL đã có các bảng chứa dữ liệu vừa trích xuất được, dữ liệu lúc này đã sẵn sàng để được đưa vào trong layer data warehouse.

**Hình 4.2.5:** Sử dụng truy vấn để kiểm tra   
dữ liệu bảng Products ở layer staging

**Bước 6** (tùy chọn): Tạo lại các bảng trong layer warehouse với lần đồng bộ đầu tiên

Vì lần đầu tiên đồng bộ dữ liệu có thể chưa có các bảng này nên cần chạy lại các câu SQL tạo bảng:

def recreate\_warehouse\_table(table):

warehouse\_table\_name = f"iceberg.warehouse.{table}"

cursor = hive.connect(host='spark-thriftserver'

, port=10000).cursor()

cursor.execute(f"""

DROP TABLE IF EXISTS {warehouse\_table\_name}

""")

cursor.execute(f"""

CREATE TABLE IF NOT EXISTS {warehouse\_table\_name} (

{",".join(list(map(lambda col: f'{col["name"]} {col["type"]}', ALL\_TABLES[table]["schema"])))}

)

USING iceberg

LOCATION '{BASE\_PATH}{WAREHOUSE}/{table}/\*.snappy.parquet'

""")

**Bước 7**: Tải dữ liệu từ layer staging vào layer warehouse để chuẩn bị tính toán

Sau khi dữ liệu mới đã được cập nhật vào staging layer, sẽ thực thi một lệnh MERGE vào các bảng trong Data Warehouse

def insert\_warehouse\_table(table):

raw\_table\_name = f"default.{table}"

warehouse\_table\_name = f"iceberg.warehouse.{table}"

cursor=hive.connect(host='spark-thriftserver', port=10000).cursor()

count\_query = f"SELECT COUNT(\*) FROM {raw\_table\_name}"

cursor.execute(count\_query)

has\_staging\_data = cursor.fetchone()[0]

print(count\_query, "---", has\_staging\_data)

if has\_staging\_data == 0:

print("STAGING HAS NO DATA, SKIPPING...")

else:

cursor.execute(f"""

MERGE INTO {warehouse\_table\_name} t

USING {raw\_table\_name} s

ON {' AND '.join(list(map(lambda col\_name: f"t.{col\_name}=s.{col\_name}", ALL\_TABLES[table]["primary\_key"])))}

WHEN MATCHED

THEN UPDATE SET

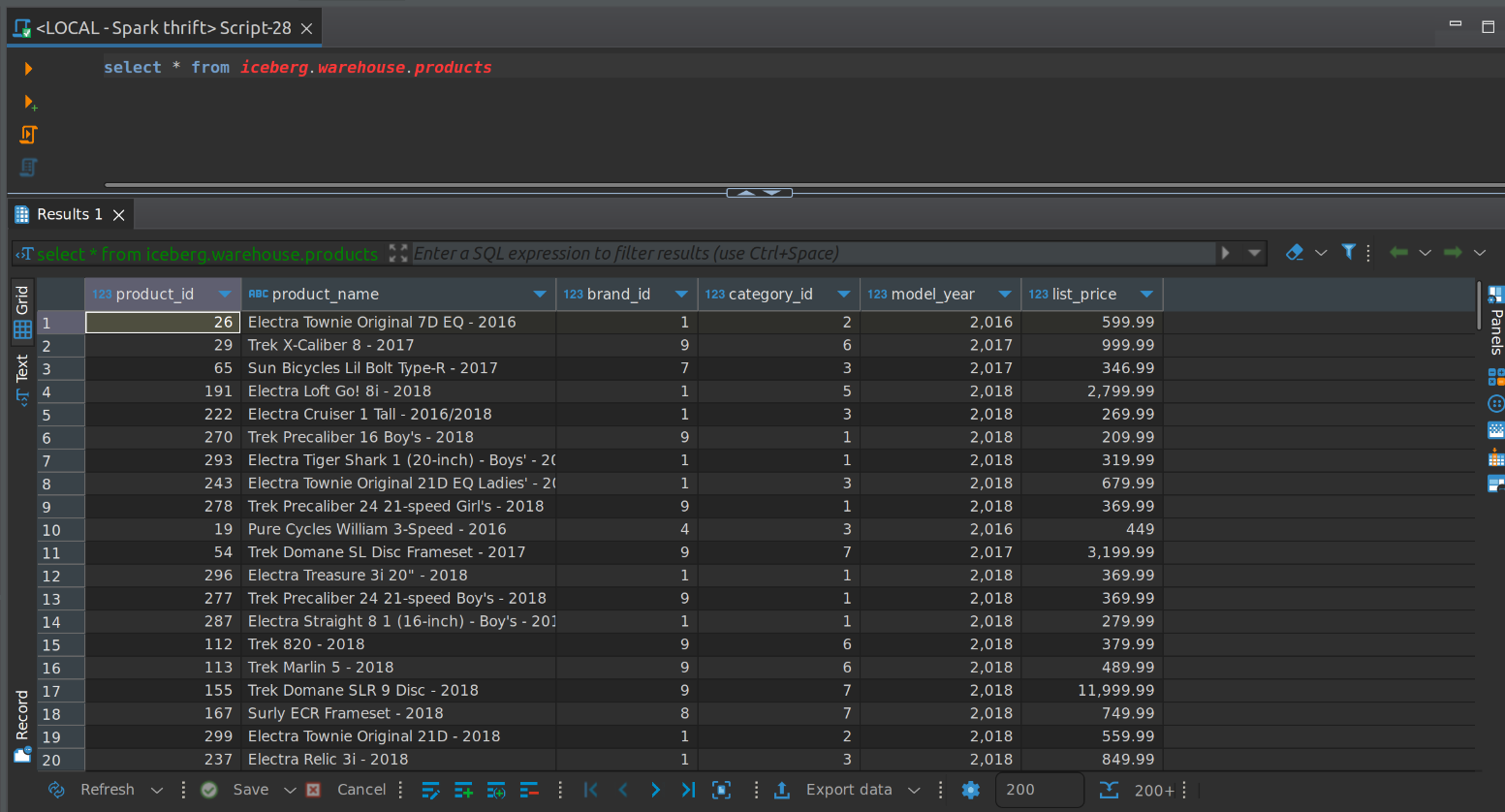
{','.join(list(map(lambda col: f"t.{col["name"]}=s.{col["name"]}", ALL\_TABLES[table]["schema"])))}

WHEN NOT MATCHED

THEN INSERT \*

""")

Sau khi thực thi thành công tác vụ trên, các bảng trong layer data warehouse đã có dữ liệu mới nhất. Có thể sử dụng câu lệnh SparkSQL để kiểm tra như sau:

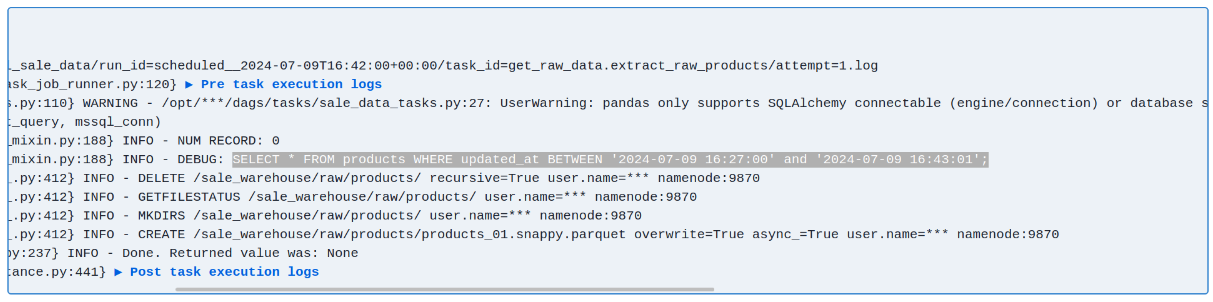
**Hình 4.2.5:** Sử dụng truy vấn để kiểm tra   
dữ liệu bảng Products ở layer data warehouse

### 4.2.2 Luồng đồng bộ dữ liệu theo từng lô nhỏ

Để đảm bảo cho các thông tin trên báo cáo được cập nhật theo các giao dịch mới nhất được tạo, bên cạnh luồng đồng bộ dữ liệu theo lô chạy mỗi ngày 1 lần thì ta cần có thêm một luồng đồng bộ dạng mini-batch, luồng này mỗi lần chạy sẽ thực hiện các thay đổi mới nhất vào hệ thống phân tích để giúp các báo cáo được cập nhật thông tin kịp thời. Luồng đồng bộ này có thể chạy từ mỗi phút một lần để đảm bảo độ trễ thông tin trên báo cáo tối đa chỉ là 1 phút. Có thể thay đổi thông số này tùy vào nhu cầu làm mới của báo cáo và khả năng chịu tải của hệ thống.

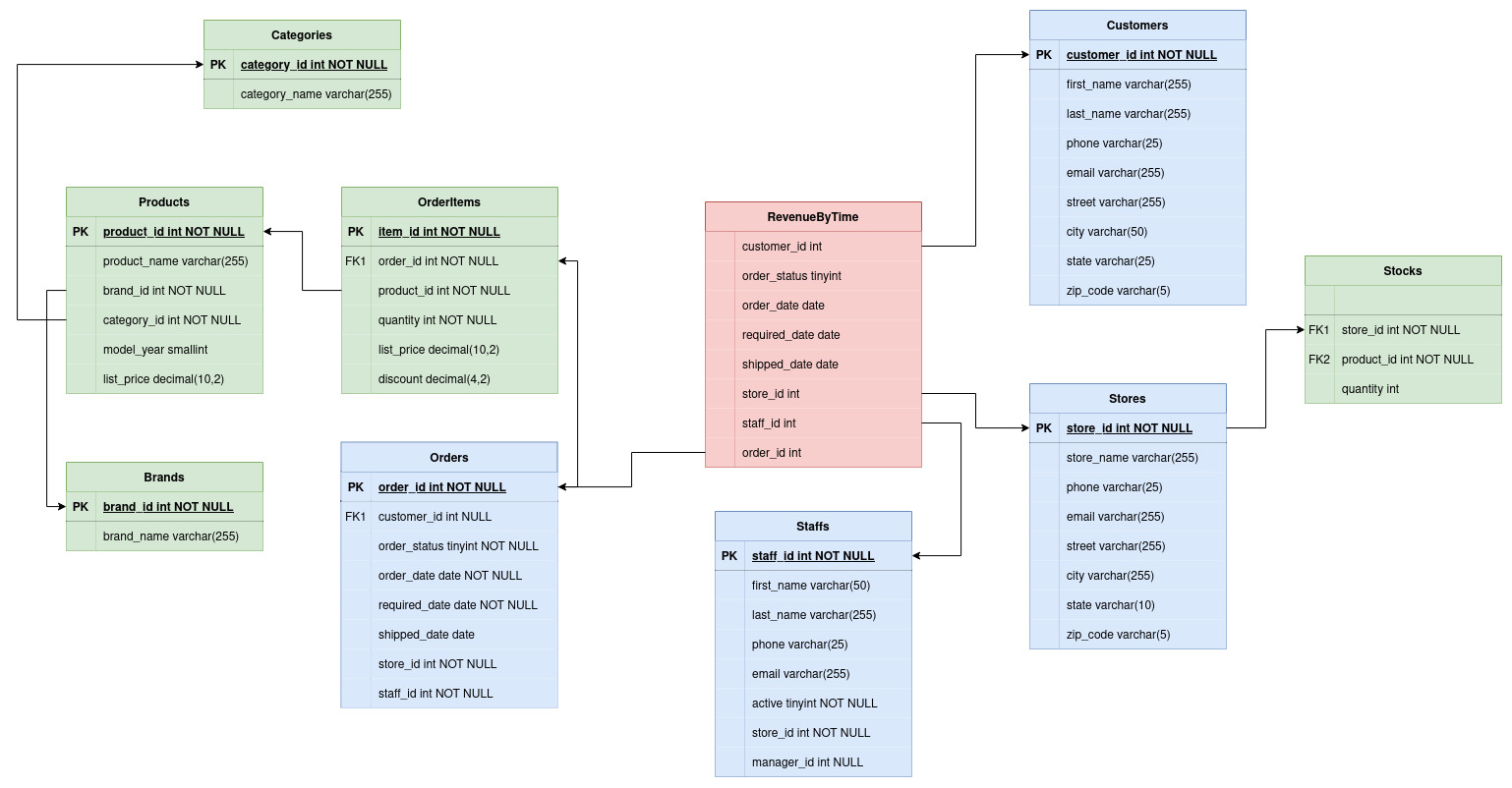
Luồng này cũng bao gồm 7 bước giống như luồng đồng bộ theo lô ở mục trên, chỉ khác với luồng đồng bộ theo lô ở **Bước 2** lấy dữ liệu từ nguồn, thay vì một thời điểm lấy toàn bộ dữ liệu của bảng thì luồng đồng bộ này chỉ lấy một phần nhỏ dữ liệu theo các khoảng thời gian. Truy vấn được thực thi để trích xuất dữ liệu sẽ có dạng như sau:

SELECT \* FROM {table} WHERE updated\_at BETWEEN '{ext\_from}' and '{ext\_to}';

**Hình 4.2.6:** Truy vấn được thực hiện để trích xuất dữ liệu từ nguồn theo lô nhỏ

## 4.3 Tính toán từ layer data warehouse ra các bảng tổng hợp

### 4.3.1 Bảng tổng hợp RevenueByTime

**Hình 4.3.1:** Mô hình dữ liệu của bảng tổng hợp   
**RevenueByTime** theo thiết kế Snowflake

Bảng tổng hợp: **RevenueByTime**

Bảng này được sử dụng để tính toán doanh thu của toàn bộ các gian hàng theo thời gian. Từ bảng dữ liệu này có thể thu được các thông tin như sau:

* Doanh thu của toàn bộ hệ thống theo giời gian (đơn vị nhỏ nhất là ngày)
* Doanh thu theo thời gian của từng gian hàng (ánh xạ theo order\_date và store\_id)
* Doanh thu theo thời gian của từng khách hàng (ánh xạ theo order\_date và customer\_id)
* Doanh thu theo thời gian của từng nhân viên bán hàng (ánh xạ theo order\_date và staff\_id)
* Tình trạng các đơn hàng theo từng ngày (ánh xạ theo order\_date và order\_status)

Câu lệnh SQL để tạo bảng:

CREATE OR REPLACE TABLE iceberg.aggr\_warehouse.revenue\_by\_time USING parquet AS

SELECT o.order\_id,

o.customer\_id,

o.store\_id,

o.staff\_id,

CASE

WHEN o.order\_status = 1 THEN 'Pending'

WHEN o.order\_status = 2 THEN 'Processing'

WHEN o.order\_status = 3 THEN 'Rejected'

WHEN o.order\_status = 4 THEN 'Completed'

END as order\_status,

o.order\_date,

(oi.list\_price \* oi.quantity) as revenue,

(oi.list\_price \* oi.quantity \* oi.discount) as discount\_revenue,

(revenue - discount\_revenue) as net\_revenue,

CONCAT(c.first\_name, ' ', c.last\_name) as customer\_fullname

FROM iceberg.warehouse.orders o

JOIN iceberg.warehouse.order\_items oi ON o.order\_id = oi.order\_id

LEFT JOIN iceberg.warehouse.customers c ON o.customer\_id = c.customer\_id

### 4.3.2 Bảng tổng hợp HumanResource

## 4.4 Đưa dữ liệu đã tính toán ra bảng fact sang ClickHouse để phục vụ truy vấn báo cáo

Dữ liệu sau khi tính toán ra các bảng thông tin cần đượ đưa tới speed layer (ClickHouse) để phuc vụ việc truy vấn nhanh hơn.

**Bước 1**: Khởi tạo kết nối tới ClickHouse

clickhouse\_client = Client('clickhouse1', database="default")

**Bước 2**: Đẩy dữ liệu đã tính toán tới ClickHouse

clickhouse\_client.execute(f"""

CREATE OR REPLACE TABLE {table}\_hdfs

ENGINE=HDFS('hdfs://namenode:9000/user/hive/warehouse/aggr\_warehouse.db/{table}\_parquet/\*.snappy.parquet', 'Parquet')

""")

clickhouse\_client.execute(f"""

CREATE OR REPLACE TABLE {table} ON CLUSTER clickhouse\_cluster

AS {table}\_hdfs

ENGINE=MergeTree()

ORDER BY (date(order\_date))

SETTINGS allow\_nullable\_key=true

""")

clickhouse\_client.execute(f"""

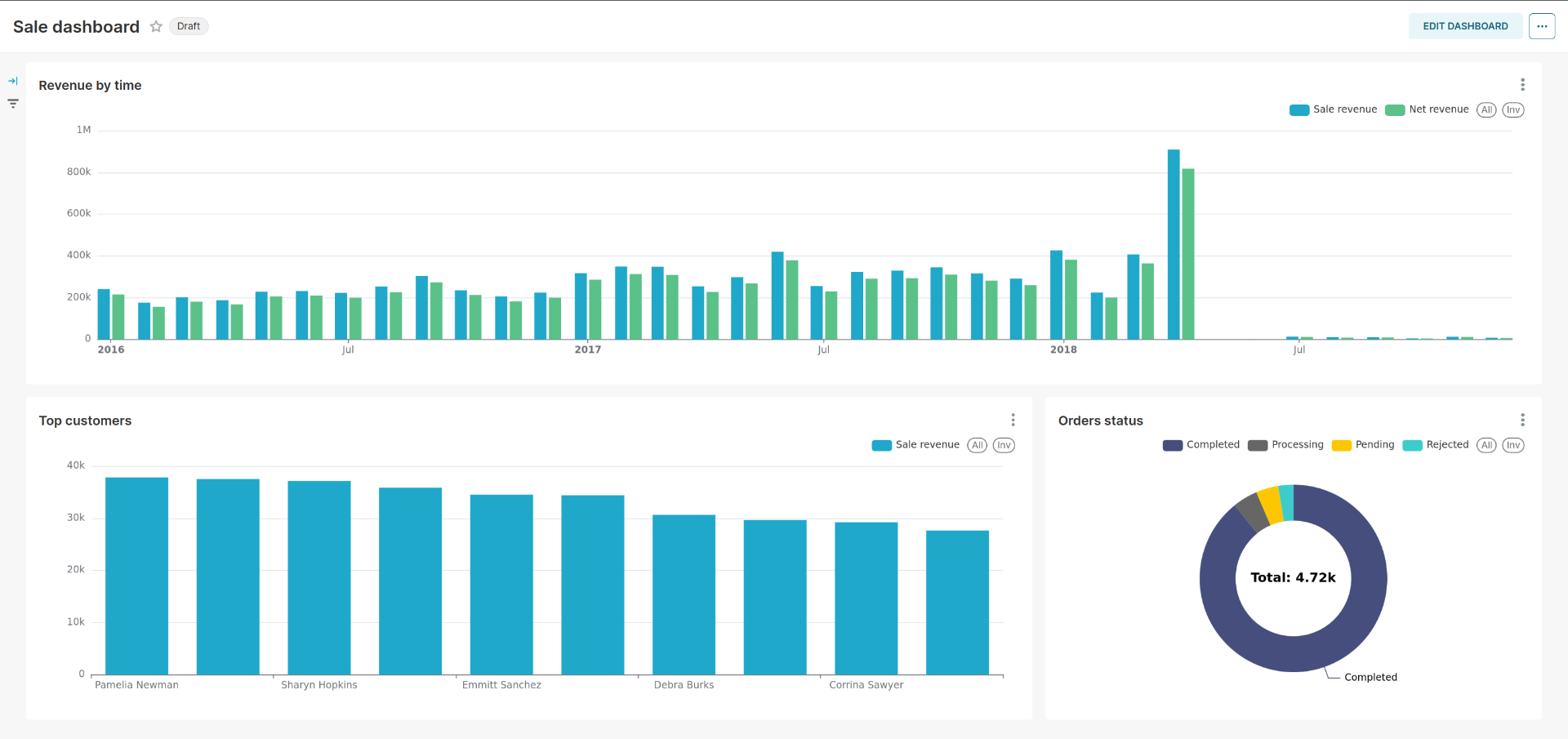
INSERT INTO {table} SELECT \* FROM {table}\_hdfs

""")

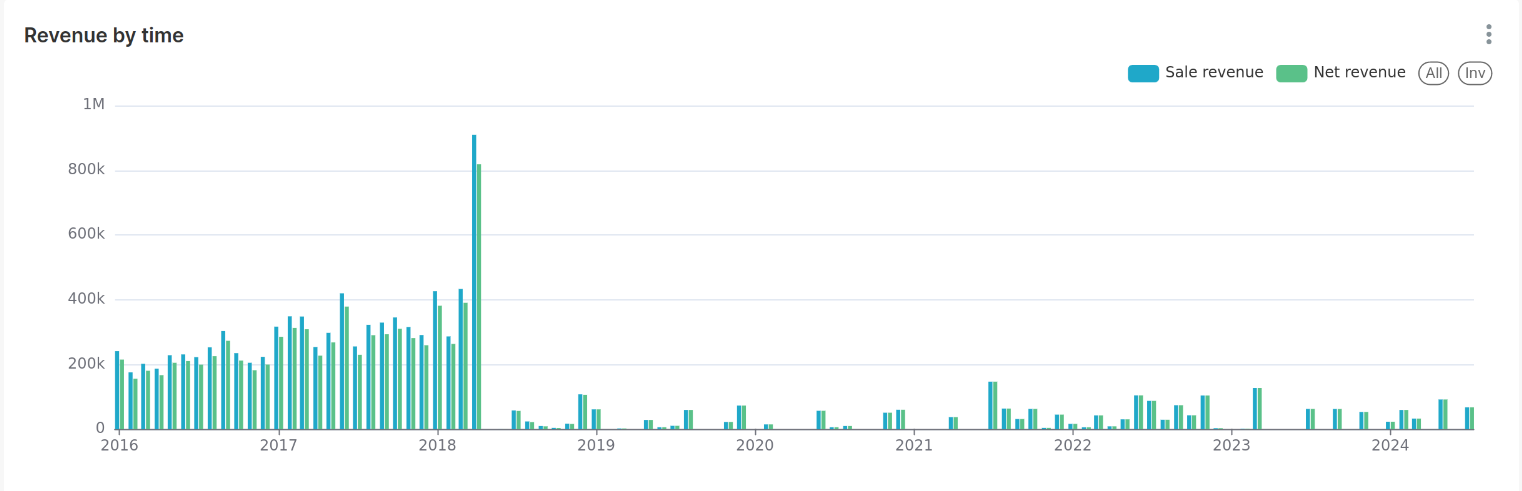
# Chương 5. Trực quan hóa dữ liệu

Sau khi đưa dữ liệu thành công tới ClickHouse, ta có thể tạo đươc báo cáo từ Superset để trực quan hóa các bảng dữ liệu.

## 5.1 Sale dashboard - Phân tích dữ liệu kinh doanh theo thời gian

**Hình 5.1:** Báo cáo dữ liệu kinh doanh theo thời gian

### 5.1.1 Biểu đồ Revenue By Time

**Hình 5.1.1:** Biểu đồ Revenue by time

**Revenue by time** là biểu đồ dạng cột, thể hiện tổng doanh thu của các cửa hàng theo trục tham chiếu là thời gian. Biểu đồ này có thể cho người dùng thấy sự thay đổi doanh thu của cửa hàng tăng giảm theo thời gian. Mỗi cột thể hiện doanh thu của một tháng, thời gian tăng dần từ trái sang phải.

Trục hoành: thời gian, đơn vị là tháng

Trục tung: doanh thu, đơn vị là USD

*Câu lệnh SQL của biểu đồ* ***Revenue by time*** *được tạo bởi Superset:*

SELECT toStartOfMonth(toDateTime(`order\_date`)) AS `order\_date\_f07f60`,

sum(`revenue`) AS `Sale revenue\_b3a53a`,

sum(`net\_revenue`) AS `Net revenue\_fcff0c`

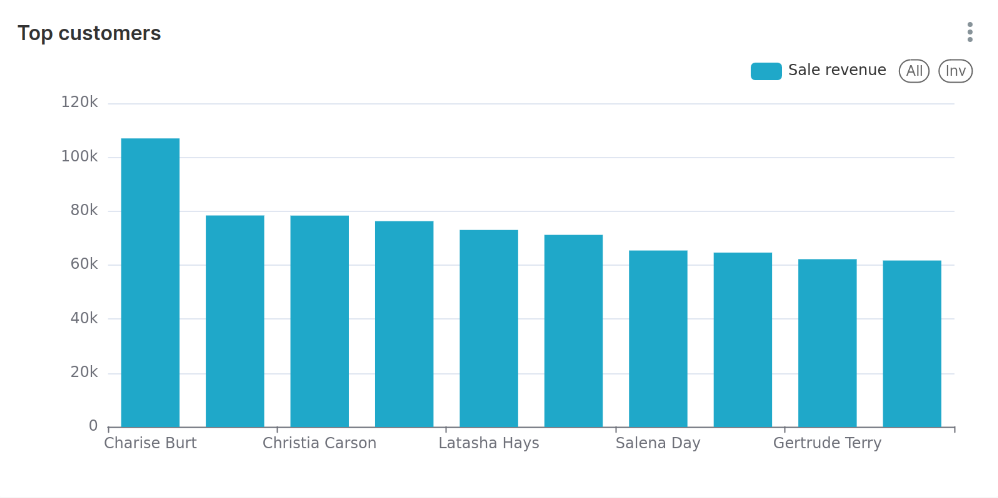
FROM `default`.`revenue\_by\_time`

GROUP BY toStartOfMonth(toDateTime(`order\_date`))

ORDER BY `Sale revenue\_b3a53a` DESC

LIMIT 10000;

### 5.1.2 Biểu đồ Top customers

**Hình 5.1.2**: Biểu đồ Top customers

**Top customers** là biểu đồ dạng cột, thể hiện danh sách 10 khách hàng có tổng doanh thu các đơn hàng cao nhất trên toàn bộ các cửa hàng của hệ thống.

Trục tung: tổng doanh thu của khách, đơn vị tính là USD

Trục hoành: tên khách hàng, sắp xếp theo thứ tự khách có doanh thu giảm dần từ trái sang phải.

*Câu lệnh SQL của biểu đồ* ***Top customers*** *được tạo bởi Superset:*

SELECT `customer\_fullname` AS `customer\_fullname\_29b5ce`,

sum(`revenue`) AS `Sale revenue\_b3a53a`

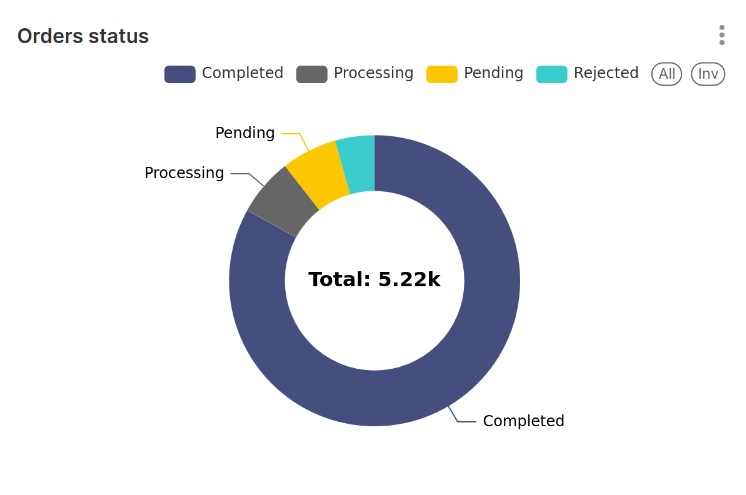
FROM `default`.`revenue\_by\_time`

GROUP BY `customer\_fullname`

ORDER BY `Sale revenue\_b3a53a` DESC

LIMIT 10;

### 5.1.3 Biểu đồ Orders status

**Hình 5.1.3**: Biểu đồ Orders status

**Orders status** là biểu đồ tròn, thể hiện tỉ lệ các trạng thái đơn hàng đóng góp bao nhiêu vào tổng số đơn. Như hình trên đang thể hiện có tổng số 5220 đơn hàng trên toàn hệ thống. Phần lớn các đơn hàng đang ở trạng thái đã hoàn thành (Completed).

*Câu lệnh SQL của biểu đồ* ***Orders status*** *được tạo bởi Superset:*

SELECT `order\_status` AS `order\_status\_2a809d`,

COUNT(\*) AS `count\_e2942a`

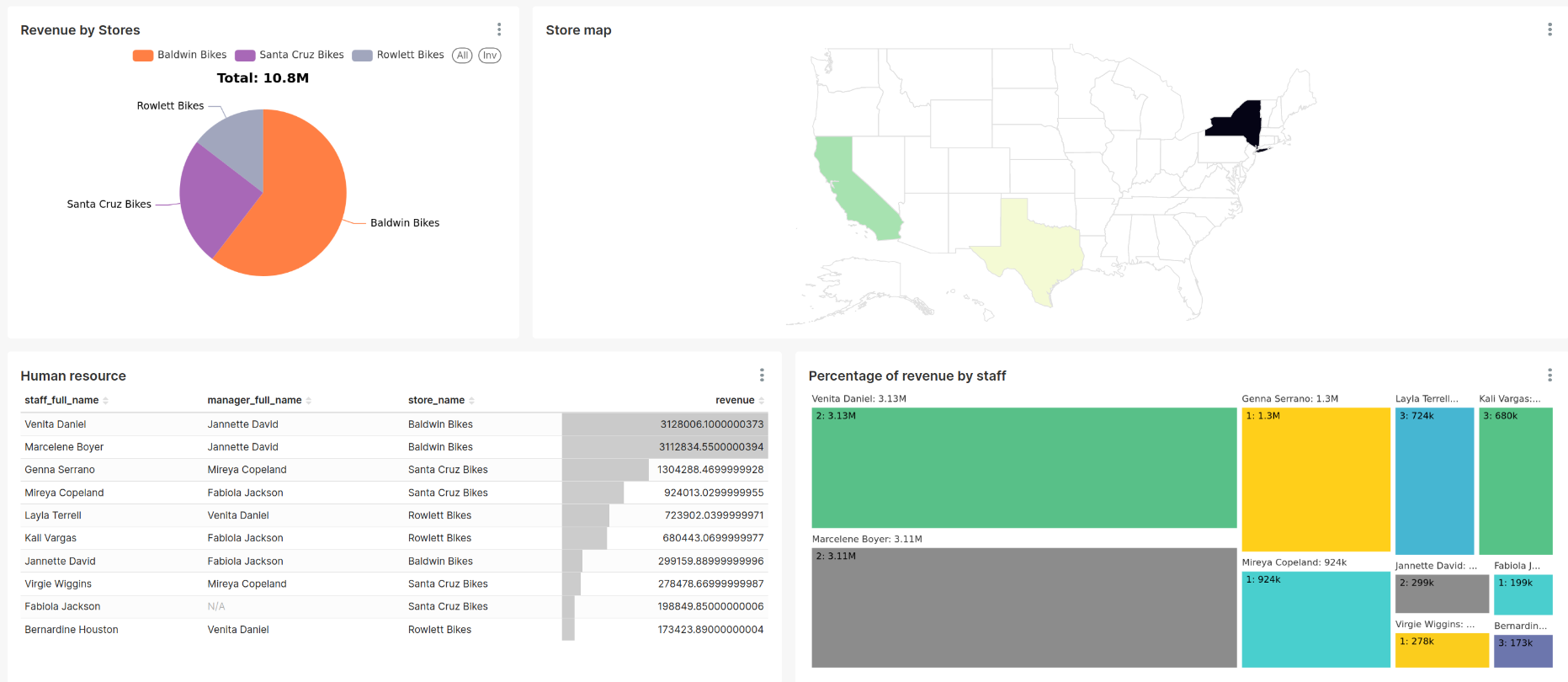
FROM `default`.`revenue\_by\_time`

GROUP BY `order\_status`

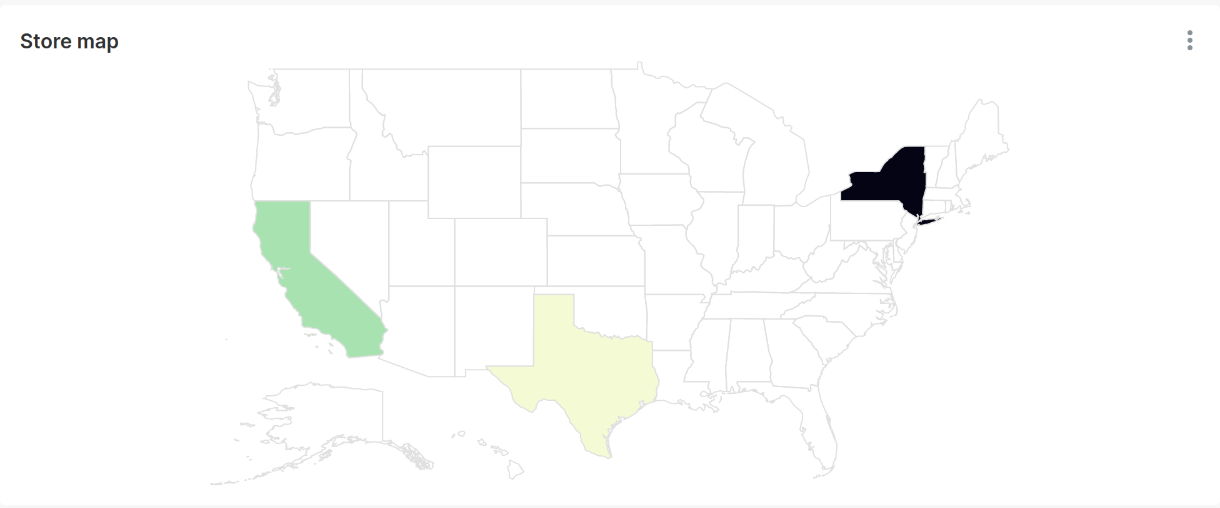
ORDER BY `count\_e2942a` DESC

LIMIT 100;

## 5.2 Human Resource - Phân tích dữ liệu nhân viên tại các cửa hàng

**Hình 5.2:** Báo cáo Human Resouce - tình hình nhân sự   
và doanh thu tại các cửa hàng

### 5.2.1 Biểu đồ Store map

**Hình 5.2.1:** Biểu đồ Store map

**Store map** là biểu đồ dạng bản đồ, thể hiện vị trí địa lí của các cửa hàng tại các bang trên bản đồ nước Mĩ. Có thể di chuột vào một bang để xem tổng doanh thu của các cửa hàng tại bang đấy theo đơn vị USD

*Câu lệnh SQL của biểu đồ* ***Store map*** *được tạo bởi Superset:*

SELECT `store\_state` AS `store\_state\_745aaf`,

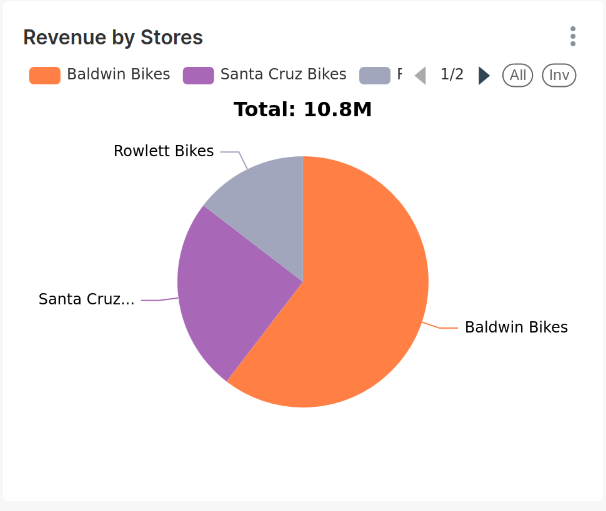
sum(`revenue`) AS `SUM(revenue)\_ba9a51`

FROM `default`.`human\_resource`

GROUP BY `store\_state`

LIMIT 50000;

### 5.2.2 Biểu đồ Revenue by stores



**Hình 5.2.3:** Biểu đồ Revenue by stores

**Revenue by stores** là biểu đồ tròn, thể hiện tỉ lệ đóng góp của từng cửa hàng vào tổng doanh thu. Nhìn hình trên có thể thấy tổng doanh thu của toàn bộ 3 cửa hàng là 10.8 triệu USD, và trong đó cửa hàng Baldwin Bikes đóng góp nhiều nhất với hơn 60% tổng doanh thu.

*Câu lệnh SQL của biểu đồ* ***Revenue by stores*** *được tạo bởi Superset:*

SELECT `store\_name` AS `store\_name\_b7cd12`,

sum(`revenue`) AS `SUM(revenue)\_ba9a51`

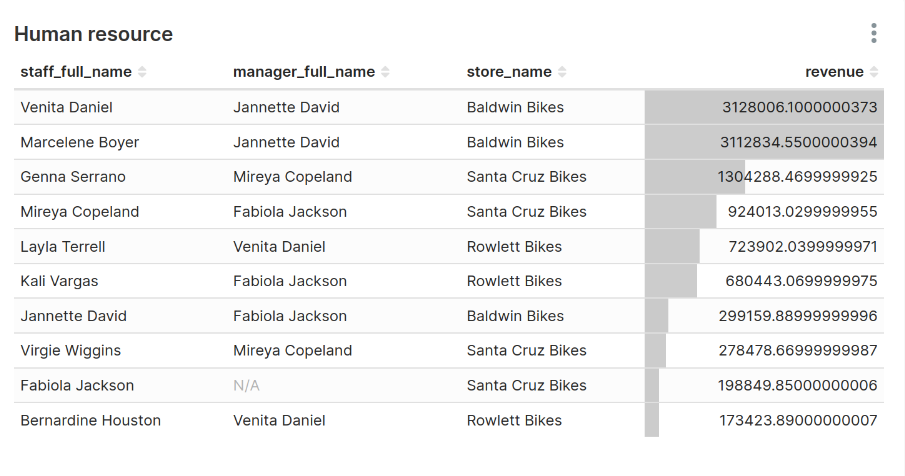
FROM `default`.`human\_resource`

GROUP BY `store\_name`

ORDER BY `SUM(revenue)\_ba9a51` DESC

LIMIT 100;

### 5.2.3 Biểu đồ Human Resouce

**Hình 5.2.3:** Biểu đồ Human resouce

**Human resouce** - là biểu đồ dạng bảng liệt kê danh sách nhân viên, quản lí, cửa hàng và doanh thu của từng nhân viên đo. Bảng này được sắp xếp theo thứ tự doanh thu của nhân viên từ cao đến thấp. Những nhân viên là quản lí sẽ để trống trường thông tin “manager\_full\_name”.

*Câu lệnh SQL của biểu đồ* ***Human resouce*** *được tạo bởi Superset:*

SELECT `staff\_full\_name` AS `staff\_full\_name\_7836c2`,

`manager\_full\_name` AS `manager\_full\_name\_4ceb6a`,

`store\_name` AS `store\_name\_b7cd12`,

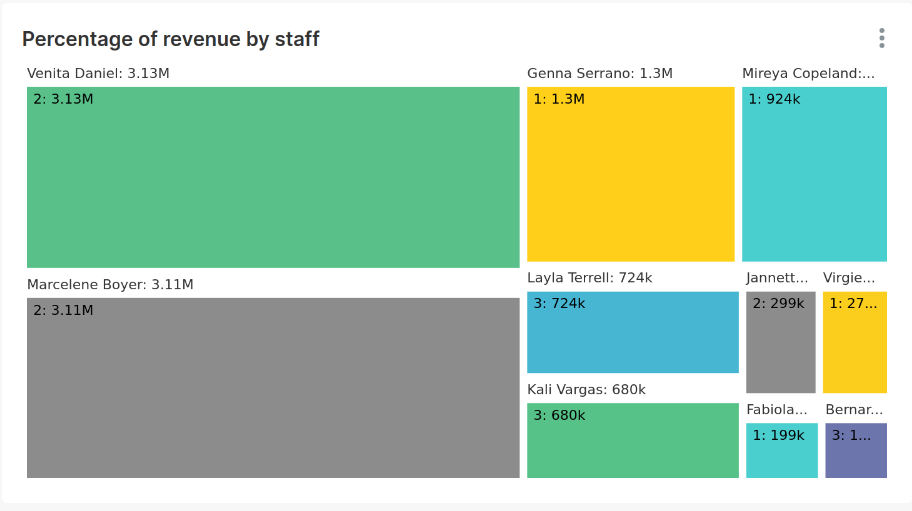
`revenue` AS `revenue\_67362d`

FROM `default`.`human\_resource`

ORDER BY `revenue` DESC

LIMIT 1000;

### 5.2.4 Biểu đồ Percentage of revenue by staff

**Hình 5.2.4:** Biểu đồ Percentage of revenue by staff

**Percentage of revenue by staff** là biểu đồ dạng cây thể hiện tỉ lệ đống góp của các nhân viên vào tổng doanh thu của toàn bộ hệ thống. Nhân viên chiếm nhiều diện tích nhất có doanh thu cao nhất. Biểu đồ được sắp xếp theo thứ tự doanh thu giảm dần từ trái sang phải và từ trên xuống dưới. Nhìn vào biểu đồ này ta có thể thấy được nhân viên nào đang đóng góp nhiều doanh thu nhất, và nhân viên nào có doanh thu ít nhất.

*Câu lệnh SQL của biểu đồ* ***Revenue by stores*** *được tạo bởi Superset:*

SELECT `store\_name` AS `store\_name\_b7cd12`,

sum(`revenue`) AS `SUM(revenue)\_ba9a51`

FROM `default`.`human\_resource`

GROUP BY `store\_name`

ORDER BY `SUM(revenue)\_ba9a51` DESC

LIMIT 100;

# Chương 6. Tổng kết và định hướng

## 6.1 Tổng kết

Trong chuyên đề này, tôi đã thực hiện các việc sau: thiết kế, xây dựng, vận hành một hệ cơ sở dữ liệu cho hệ thống quản lí bán hàng. Xây dựng, vận hành hệ thống tính toán dữ liệu phân tích và xây dựng các báo cáo để trực quan hóa ra thông tin cho người dùng.

Với phần đầu tiên là thiết kế cơ sở dữ liệu cho phần mềm quản lí bán hàng. Mô hình dữ liệu của tôi đã đảm bảo đáp ứng được đầy đủ các nghiệp vụ của một hệ thống bán hàng cần có. Các bảng được thiết kế theo chuẩn 3NF để đảm bảo tối ưu lưu trữ và không gây dư thừa dữ liệu.

Với phần xây dựng hệ thống tính toán phân tích, tôi đã xây dựng hoàn chỉnh một luồng xử lí với các công nghệ tính toán phân tán như Apache Hadoop, Apache Spark, Apache Airflow để thực hiện việc đồng bộ đữ liệu từ nguồn và tính toán ra các bảng tổng hợp với độ trễ thấp và tính đúng đắn cao.

Phần cuối cùng là thực hiện trực quan hóa, tôi đã sử dụng ClickHouse để tối ưu cho việc các truy vấn dạng tổng hợp trên cột để cho hiệu năng tốt hơn. Các báo cáo có thời gian tải rất nhanh và thông tin đưa ra từ báo cáo sẽ có ích cho người đọc.

## 6.2 Định hướng

Do hạn chế về hạ tầng máy chủ, hiện tại trong phạm vi chuyên đề này tôi chỉ có thể sử dụng công nghệ ảo hóa Docker để mô phỏng một môi trường phân tán. Trong tương lai tôi sẽ tiến hành triển khai hạ tầng cơ sở dữ liệu, hạ tầng tính toán và báo cáo lên môi trường máy chủ thật và chạy phân tán trong thực tế.

Tiếp theo, tôi có thể sẽ triển khai các mô đun báo cáo thông minh khác như: dự kiến ngày hết hàng của sản phẩm, dự đoán các sản phẩm đang được bán chậm để tìm cách bán hết các sản phẩm đó nhanh hơn, đề xuất các sản phẩm theo thói quen mua hàng của khách.

# Phụ lục

## Giải thích các khái niệm

### Relational Database Management System (RDBMS)

Relational Database Management System - RDBMS là hệ quản trị cơ sở dữ liệu lưu trữ dữ liệu theo dạng các bảng có quan hệ với nhau. Dữ liệu trong RDBMS được tổ chức theo mô hình quan hệ, trong đó có các bảng đại diện cho các thực thể và quan hệ giữa chúng được thể hiện qua các khóa.

Đặc điểm chính của RDBMS:

* **Sử dụng mô hình dữ liệu quan hệ:** Dữ liệu được lưu trữ tcác bảngbảng và các bảng đó được liên kết với nhau bằng các khóa ngoại.
* **Sử dung ngôn ngữ truy vấn cấu trúc (Sequelize - SQL):** RDBMS sử dụng SQL để truy vấn, thao tác và quản lý dữ liệu.
* **Tính nhất quán dữ liệu:** đảm bảo tính nhất quán dữ liệu bằng cách yêu cầu các ràng buộc dữ liệu trong thiết kế và thao tác chỉnh sửa, cập nhật.
* **Tính bảo mật:** cung cấp các công cụ để phân quyền và bảo mật dữ liệu.

### Data Warehouse

Data Warehouse (kho chứa dữ liệu) là một hệ quản trị cơ sở dữ liệu được thiết kế để lưu trữ, quản lý, truy vấn và phân tích dữ liệu từ nhiều nguồn khác nhau. Dữ liệu được lưu trữ trong kho dữ liệu được tối ưu hóa cho việc truy vấn và phân tích để đưa ra thông tin nhanh chóng.

Mục đích chính của Data Warehouse là hỗ trợ quá trình theo dõi hoạt động kinh doanh, dự đoán và hỗ trợ ra quyết định của doanh nghiệp.

Một Data Warehouse có các đặc điểm sau

* **Tính tích hợp:** Dữ liệu từ nhiều nguồn khác nhau được tích hợp và lưu trữ trong một kho dữ liệu duy nhất.
* **Tính chủ đề:** Dữ liệu được tổ chức theo chủ đề, chẳng hạn như bán hàng, khách hàng, sản phẩm, v.v.
* **Tính lịch sử:** Dữ liệu được lưu trữ theo thời gian là trục tham chiếu chính, cho phép người dùng có thể theo dõi lịch xử, xu hướng và tương quan các thay đổi theo thời gian.
* **Tính nhất quán:** Dữ liệu trong data warehouse thường không được thay đổi sau khi được tổng hợp.

### Transactional Data

**Transactional Data - Dữ liệu giao dịch** là một loại dữ liệu ghi lại các sự kiện hoặc giao dịch cụ thể diễn ra trong một hệ thống hoặc quy trình kinh doanh. Dữ liệu này thường được tạo ra tại thời điểm giao dịch diễn ra và được sử dụng để theo dõi và quản lý các hoạt động kinh doanh.

Ví dụ về dữ liệu giao dịch:

* **Giao dịch bán hàng:** Dữ liệu về việc bán sản phẩm hoặc dịch vụ, bao gồm thông tin về khách hàng, sản phẩm, số lượng, giá cả, v.v.
* **Giao dịch tài chính:** Dữ liệu về các khoản thanh toán, chuyển khoản, cổ tức, v.v.
* **Dữ liệu hoạt động:** Dữ liệu về các hoạt động diễn ra trong một hệ thống, chẳng hạn như đăng nhập, đăng xuất, thay đổi cấu hình, v.v.

### Dim Table và Fact Table trong thiết kế Data Warehouse

Trong thiết kế kho dữ liệu, hai thành phần cốt lõi là bảng Dim và bảng Fact. Mỗi loại bảng đóng vai trò quan trọng trong việc lưu trữ và tổ chức dữ liệu để phục vụ cho mục đích phân tích và báo cáo.

Bảng Dim (Dimension Table)

* Chứa thông tin định danh và mô tả cho các đối tượng trong kho dữ liệu.
* Ví dụ: khách hàng, sản phẩm, thời gian, khu vực địa lý, v.v.
* Cấu trúc thường bao gồm:
  + Khóa chính (Primary Key): Cột duy nhất để xác định từng bản ghi riêng lẻ.
  + Thuộc tính mô tả: Các thuộc tính cung cấp thông tin chi tiết về đối tượng.
  + Mối quan hệ: Khóa ngoại để liên kết với các bảng khác trong kho dữ liệu.

Bảng Fact (Fact Table)

* Lưu trữ các **số** liệu đo lường liên quan đến các sự kiện kinh doanh.
* Ví dụ: doanh số bán hàng, lượt truy cập trang web, thời gian phản hồi khách hàng, v.v.
* Cấu trúc:
  + Khóa chính: Cột duy nhất hoặc kết hợp các cột để xác định từng sự kiện riêng lẻ.
  + Khóa ngoại: Liên kết đến các bảng Dim tương ứng để cung cấp ngữ cảnh cho các số liệu đo lường.
  + Số liệu đo lường: Các giá trị số thể hiện các sự kiện kinh doanh.

## Hướng dẫn cài đặt

Đường dẫn truy cập mã nguồn của chuyên đề: <https://github.com/huwngnosleep/complete_lakehouse_techstack>

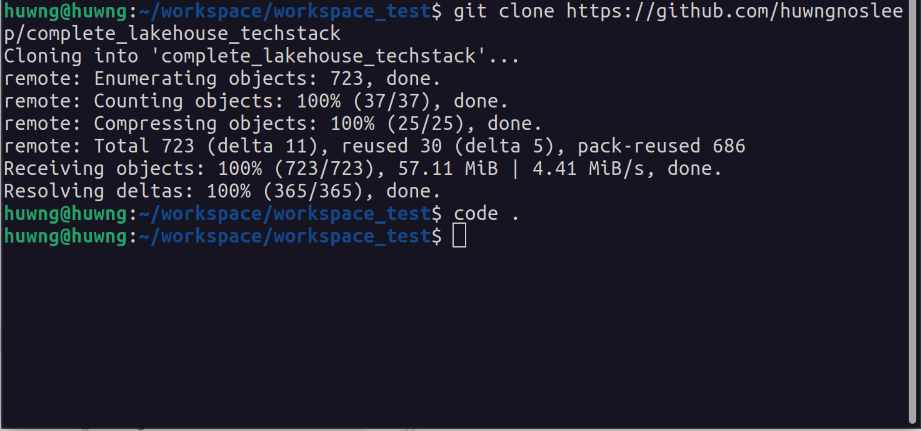
### Yêu cầu tiên quyết

* Máy tính cá nhân có cấu hình tối thiểu 32GB RAM, 16 CPUs
* Hệ điều hành Ubuntu 22.04
* Máy tính đã cài đặt phần mềm Docker phiên bản mới nhất
* Máy tính đã cài đặt Java 8
* Máy tính đã cài đặt phần mềm quản lí phiên bản Git

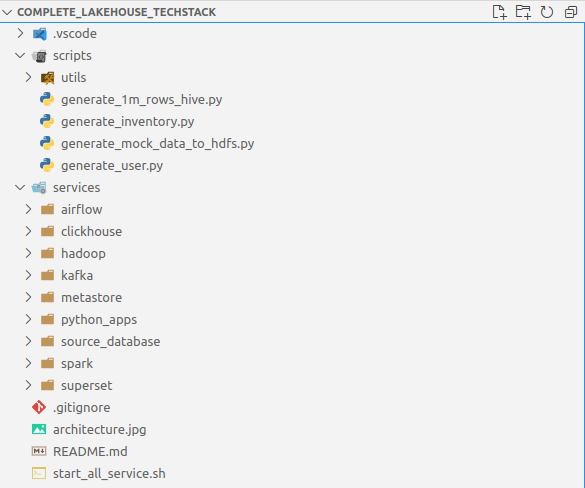
### Bước 1: Tải mã nguồn về máy tính cá nhân

Mở cửa sổ terminal, chạy lệnh sau để tải mã nguồn của dự án từ Github:

git clone <https://github.com/huwngnosleep/complete_lakehouse_techstack>

Tải xuống mã nguồn của dự án từ Github

Sau khi tải thành công, thư mục chứa mã nguồn của dự án sẽ có cấu trúc như sau:

  
Cấu trúc thư mục mã nguồn

Giải thích các thành phần trong thư mục:

* Thư mục ~/scripts: chứa các tệp Python dùng để khởi tạo dữ liệu hoặc kiểm thử thủ công
* Thư mục ~/services: chứa các cấu hình cho các công nghê được sử dụng trong chuyên đề
* Tệp lệnh bash start\_all\_service.sh: dùng để khởi chạy nhanh toàn bộ hệ thống

### Bước 2: Khởi tạo dịch vụ Hive Metastore

Hive Metastore là một dịch vụ lưu trữ metadata để phục vụ quá trình thực thi của Apache Spark. Lần đầu tiên khi khởi chạy dự án cần phải thực hiện quá trình khởi tạo dữ liệu cho Hive Metastore một cách thủ công như sau: