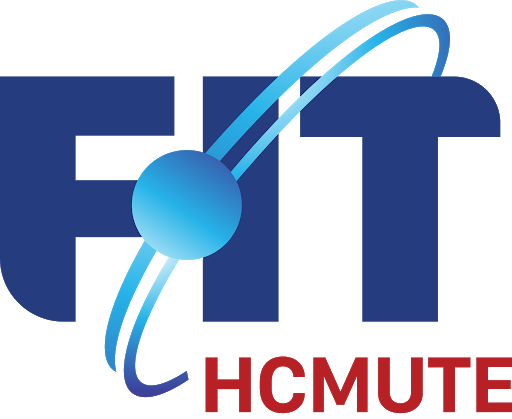
**.**

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**

**BỘ MÔN ĐIỆN TOÁN ĐÁM MÂY**

-----🙞🙜🕮🙞🙜-----

****

**TÌM HIỂU DELTA LAKE VÀ MÔ HÌNH KẾT NỐI AWS**

**GIẢNG VIÊN HƯỚNG DẪN**

**TS. HUỲNH XUÂN PHỤNG**

**Nhóm sinh viên thực hiện:**

**NINH PHẠM TRUNG THÀNH - 19110456**

**DƯƠNG ĐỨC THẮNG – 19110461**

**NGUYỄN HIẾU ĐAN - 19110345**

**KHÓA 2019 – 2023**

|  |  |
| --- | --- |
| **ĐH SƯ PHẠM KỸ THUẬT TP. HCM**  **KHOA CNTT**  \*\*\*\*\*\*\* | **CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**  **Độc lập – Tự do – Hạnh Phúc**  \*\*\*\*\*\*\* |

PHIẾU NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

Họ và tên Sinh viên 1: **Nguyễn Hiếu Đan** MSSV 1: **19110345**

Họ và tên Sinh viên 2: **Dương Đức Thắng** MSSV 2: **19110461**

Họ và tên Sinh viên 3: **Ninh Phạm Trung Thành** MSSV 2: **19110456**

Ngành: **Công nghệ thông tin**

Tên đề tài: **Tìm hiểu về Delta-lake và mô hình kết nối AWS**

Họ và tên Giáo viên phản biện: **TS. Huỳnh Xuân Phụng**

**NHẬN XÉT**

1. Về nội dung đề tài & khối lượng thực hiện:

2. Ưu điểm:

3. Khuyết điểm:

4. Đề nghị cho bảo vệ hay không?

5. Đánh giá loại:

6. Điểm:

|  |  |  |  |
| --- | --- | --- | --- |
| *Tp. Hồ Chí Minh, ngày tháng năm 2021* | | | |
|  | | Giáo viên hướng dẫn  *(Ký & ghi rõ họ tên)* |
| **ĐH SƯ PHẠM KỸ THUẬT TP. HCM**  **KHOA CNTT**  \*\*\*\*\*\*\* | | **CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**  **Độc lập – Tự do – Hạnh Phúc**  \*\*\*\*\*\*\* | | | |

PHIẾU NHẬN XÉT CỦA GIÁO VIÊN PHẢN BIỆN

Họ và tên Sinh viên 1: **Nguyễn Hiếu Đan** MSSV 1: **19110345**

Họ và tên Sinh viên 2: **Dương Đức Thắng** MSSV 2: **19110461**

Họ và tên Sinh viên 3: **Ninh Phạm Trung Thành** MSSV 2: **19110456**

Ngành: **Công nghệ thông tin**

Tên đề tài: **Tìm hiểu về Delta-lake và mô hình kết nối AWS**

Họ và tên Giáo viên phản biện: **TS. Huỳnh Xuân Phụng**

**NHẬN XÉT**

1. Về nội dung đề tài & khối lượng thực hiện:

2. Ưu điểm:

3. Khuyết điểm:

4. Đề nghị cho bảo vệ hay không?

5. Đánh giá loại:

6. Điểm:

*Tp. Hồ Chí Minh, ngày tháng năm 2021*

Giáo viên phản biện

(Ký & ghi rõ họ tên)

**LỜI CẢM ƠN**

*Lời đầu tiên nhóm xin phép được gửi lời cảm ơn chân thành và sâu sắc nhất đến với Khoa Công Nghệ Thông Tin – Trường Đại Học Sư Phạm Kỹ Thuật Thành Phố Hồ Chí Minh đã tạo điều kiện cho nhóm chúng em được học tập, phát triển nền tảng kiến thức sâu sắc và thực hiện đề tài này.*

*Bên cạnh đó nhóm chúng em xin gửi đến thầy Huỳnh Xuân Phụng lời cảm ơn sâu sắc nhất. Trải qua một quá trình dài học tập và thực hiện đề tài trong thời gian qua. Thầy đã tận tâm chỉ bảo nhiệt tình nhóm chúng em trong suốt quá trình từ lúc bắt đầu cũng như kết thúc đề tài này.*

*Với sự hướng dẫn nhiệt tình, giảng dạy tận tình đầy đủ kiến thức của thầy Huỳnh Xuân Phụng, chúng em đã học tập và hiểu được những kiến thức cơ bản về Delta –lake một framework mã nguồn mở dung cho việc xử lý dữ liệu trong datalake. Qua đó tụi em biết cách cài đặt và sử dụng Delta lake trên Aws.*

*Tuy nhiên lượng kiến thức là vô tận và với khả năng hạn hẹp chúng em đã rất cố gắng để hoàn thành một cách tốt nhất. Chính vì vậy việc xảy ra những thiếu sót là điều khó có thể tránh khỏi. Chúng em hi vọng nhận được sự góp ý tận tình của quý thầy (cô) qua đó chúng em có thể rút ra được bài học kinh nghiệm và hoàn thiện và cải thiện nâng cấp lại sản phẩm của mình một cách tốt nhất có thể.*

*Chúng em xin chân thành cảm ơn!*

***Nhóm thực hiện***

*Nguyễn Hiếu Đan - 19110345*

*Dương Đức Thắng – 19110461*

*Ninh Phạm Trung Thành – 19110456*

**MỤC LỤC**

[PHẦN MỞ ĐẦU 1](#_Toc121073026)

[**1. Tính cấp thiết của đề tài** 1](#_Toc121073027)

[**2. Đối tượng nghiên cứu** 1](#_Toc121073028)

[**3. Phạm vi nghiên cứu** 1](#_Toc121073029)

[**4. Kết quả dự kiến đạt được** 1](#_Toc121073030)

[PHẦN NỘI DUNG 2](#_Toc121073031)

[CHƯƠNG 1: CƠ SỞ LÝ THUYẾT VỀ DELTA LAKE 2](#_Toc121073032)

[**1.1** **DataLake là gì** 2](#_Toc121073033)

[**1.2** **Hadoop là gì** 2](#_Toc121073034)

[**1.3** **Spark là gì:** 2](#_Toc121073035)

[**1.4 Delta-lake là gì** 3](#_Toc121073036)

[**1.5 Tại sao phải sử dụng Delta-lake** 3](#_Toc121073037)

[**1.6 Kiến trúc Delta-lake** 3](#_Toc121073038)

[CHƯƠNG 2: MÔ HÌNH KẾT NỐI VỚI AWS 5](#_Toc121073039)

[**2.1 Mô hình kết nối** 5](#_Toc121073040)

[CHƯƠNG 3: CÀI ĐẶT, SỬ DỤNG DELTALAKE TRÊN AWS 6](#_Toc121073041)

[**3.1 Cài đặt bằng EC2** 6](#_Toc121073042)

[**3.1.1 Tạo EC2 Instance** 6](#_Toc121073043)

[**3.1.2 Tạo S3 bucket** 8](#_Toc121073044)

[**3.1.3 Cài đặt môi trường delta lake trên EC2** 9](#_Toc121073045)

[**3.2 Tạo bằng EMR** 14](#_Toc121073046)

[**3.3 Tạo NoteBook để sử dụng khai thác dữ liệu** 18](#_Toc121073047)

[**3.4 Test NoteBook hoạt động.** 19](#_Toc121073048)

[**3.5 Demo khai thác dữ liệu.** 23](#_Toc121073049)

[**3.5.1 Chuẩn bị dữ liệu** 23](#_Toc121073050)

[**3.5.2 Đọc dữ liệu** 25](#_Toc121073051)

[**3.5.3 Cập nhật dữ liệu** 26](#_Toc121073052)

[**3.5.4 Xóa dữ liệu** 27](#_Toc121073053)

[**3.5.5 Du hành thời gian (Time travel)** 28](#_Toc121073054)

[**3.5.6 Upsert dữ liệu** 28](#_Toc121073055)

[PHẦN KẾT LUẬN 30](#_Toc121073056)

[**1.** **Kết quả đạt được** 30](#_Toc121073057)

[**1.1. Kiến thức tìm hiểu được** 30](#_Toc121073058)

[**1.2. Chương trình đã làm được** 30](#_Toc121073059)

[**2.** **Ưu điểm** 30](#_Toc121073060)

[**3.** **Hướng phát triển** 30](#_Toc121073061)

[TÀI LIỆU THAM KHẢO 31](#_Toc121073062)

# **PHẦN MỞ ĐẦU**

## **1. Tính cấp thiết của đề tài**

Hiện nay, 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. Vì vậy nên sự ra đời của WareHouse, DataLake để giải quyết, cả 2 phương pháp trên đều có vấn đề của nó. Với WareHouse thì nó không thể giải quyết được dữ liệu không và bán cấu trúc, còn DataLake thì không có các tính năng như transaction và tính ACID. Vì vậy kiến trúc lake house ra đời để hỗ trợ kết hợp cả hai phương pháp trên nhằm giải quyết tất cả các vấn đề của chúng. Những công cụ được xây dựng dựa trên kiến trúc lake house này là : Azure Synapse, Google Bigquery, AWS Redshift… ngoài những tool phải trả phí này ra chúng ta còn có framework mã nguồn mỡ delta lake cũng là chủ đề chính của đề tài này.

## **2. Đối tượng nghiên cứu**

Đối với đề tài này, đối tượng nghiên cứu là Big Data. Đồng thời kèm theo đó là các công nghệ áp dụng để xây dựng lake house đơn giản, cụ thể như:

* JDK và JRE: Bộ công cụ phát triển Java.
* Python và thư viện hỗ trợ Spark: PySpark.
* Spark: Apache Spark là một framework mã nguồn mở tính toán cụm, được phát triển sơ khởi vào năm 2009 bởi AMPLab.
* Delta lake: Là một framework mã nguồn mở hỗ trợ triển khai các kiến trúc lake house hiện đại.
* Amazon và EMR (Elastic MapReduce)
* Amazon Simple Storage (S3) và Amazon Elastic Compute(EC2)

## **3. Phạm vi nghiên cứu**

Đề tài này chủ yếu tập trung vào nghiên cứu delta lake và cách kết hợp với 1 số dịch vụ AWS nhằm triển khai mô hình datalake kết hợp delta lake đơn giản.

## **4. Kết quả dự kiến đạt được**

* Cài đặt được Spark và delta lake trên EC2.
* Xây dựng được một data lake đơn giản bằng các câu truy vấn.
* Phân tích, thống kê các dữ liệu từ các câu truy vấn tùy vào mục đích của người dùng.
* Cài đặt Delta lake trên AWS sử dụng EMR (Elastic MapReduce) và thao tác dữ liệu qua notebook.

# **PHẦN NỘI DUNG**

# **CHƯƠNG 1: CƠ SỞ LÝ THUYẾT VỀ DELTA LAKE**

## **DataLake là gì**

Data Lake (hay Hồ dữ liệu) là một kho lưu trữ tập trung được thiết kế để lưu trữ, xử lý và bảo mật một lượng lớn dữ liệu có cấu trúc, bán cấu trúc và phi cấu trúc. Nó có thể lưu trữ dữ liệu ở định dạng gốc và xử lý mọi loại dữ liệu khác nhau, bỏ qua các giới hạn về kích thước. Nó cung cấp số lượng dữ liệu cao để tăng hiệu suất phân tích và tích hợp gốc.

## **Hadoop là gì**

Hadoop là một Apache framework mã nguồn mở cho phép phát triển các ứng dụng phân tán (distributed processing) để lưu trữ và quản lý các tập dữ liệu lớn. Hadoop hiện thực mô hình MapReduce, mô hình mà ứng dụng sẽ được chia nhỏ ra thành nhiều phân đoạn khác nhau được chạy song song trên nhiều node khác nhau. Hadoop được viết bằng Java tuy nhiên vẫn hỗ trợ C++, Python, Perl bằng cơ chế streaming. Hadoop giải quyết:

* Xử lý và làm việc khối lượng dữ liệu khổng lồ tính bằng Petabyte.
* Xử lý trong môi trường phân tán, dữ liệu lưu trữ ở nhiều phần cứng khác nhau, yêu cầu xử lý đồng bộ.
* Băng thông giữa các phần cứng vật lý chứa dữ liệu phân tán có giới hạ

## **Spark là gì:**

Apache Spark là một framework mã nguồn mở tính toán cụm, được phát triển sơ khởi vào năm 2009 bởi AMPLab. Sau này, Spark đã được trao cho Apache Software Foundation vào năm 2013 và được phát triển cho đến nay.

Tốc độ xử lý của Spark có được do việc tính toán được thực hiện cùng lúc trên nhiều máy khác nhau. Đồng thời việc tính toán được thực hiện ở bộ nhớ trong (in-memories) hay thực hiện hoàn toàn trên RAM.

Spark cho phép xử lý dữ liệu theo thời gian thực, vừa nhận dữ liệu từ các nguồn khác nhau đồng thời thực hiện ngay việc xử lý trên dữ liệu vừa nhận được ( Spark Streaming).

Spark không có hệ thống file của riêng mình, nó sử dụng hệ thống file khác như: HDFS, Cassandra, S3…. Spark hỗ trợ nhiều kiểu định dạng file khác nhau (text, csv, json…) đồng thời nó hoàn toàn không phụ thuộc vào bất cứ một hệ thống file nào.

## **1.4 Delta-lake là gì**

Delta lake là một framework lưu trữ mã nguồn mở được ra mắt vào năm 2019 cho phép xây dựng kiến trúc data-lake với các công cụ tính toán như Spark, PrestoDB, Flink, Trino, Hive và các API sử dụng Scala, Java, Rust, Ruby, and Python.

* + Delta Lake là một lớp lưu trữ nguồn mở đảm bảo độ tin cậy cho các hồ dữ liệu.
  + Nó được thiết kế đặc biệt để hoạt động với Databricks File System (DBFS) và Apache Spark.
  + Nó cung cấp khả năng xử lý dữ liệu hàng loạt và phát trực tuyến hợp nhất, giao dịch ACID và xử lý siêu dữ liệu có thể mở rộng.
  + Nó lưu trữ dữ liệu của bạn dưới dạng tệp Apache Parquet trong DBFS và duy trì nhật ký giao dịch theo dõi chính xác các thay đổi đối với bảng.
  + Nó làm cho dữ liệu sẵn sàng để phân tích.

## **1.5 Tại sao phải sử dụng Delta-lake**

* Tận dụng sức mạnh xử lý phân tán của Spark để xử lý tất cả siêu dữ liệu cho các bảng quy mô hàng petabyte với hàng tỷ file một cách dễ dàng.
* Time travel - Dễ dàng truy cập và hoàn lại những phiên bản trước của dữ liệu.
* Định dạng mở lưu trữ dưới Parquet file.
* Hợp nhất Batch & Streaming, Source & Sink.
* Dễ dàng thay đổi lược đồ hiện tại của bảng để phù hợp với dữ liệu.
* Hiệu suất nhanh với Apache Spark.
* Hỗ trợ MERGE, UPDATE, DELETE và phát trực tuyến UPSERTS.
* Hỗ trợ ACID transaction.

## **1.6 Kiến trúc Delta-lake**

Delta Lake thuộc Transaction Layer nằm trên đỉnh storage layer của data lake để nhận dữ liệu đáng tin cậy trong các hồ dữ liệu đám mây như Amazon S3 và ADLS Gen2. Delta Lake đảm bảo dữ liệu nhất quán, đáng tin cậy với các giao dịch ACID, lập phiên bản dữ liệu tích hợp và kiểm soát để đọc và ghi đồng thời. Nó còn cho phép tái tạo báo cáo dễ dàng và đáng tin cậy.

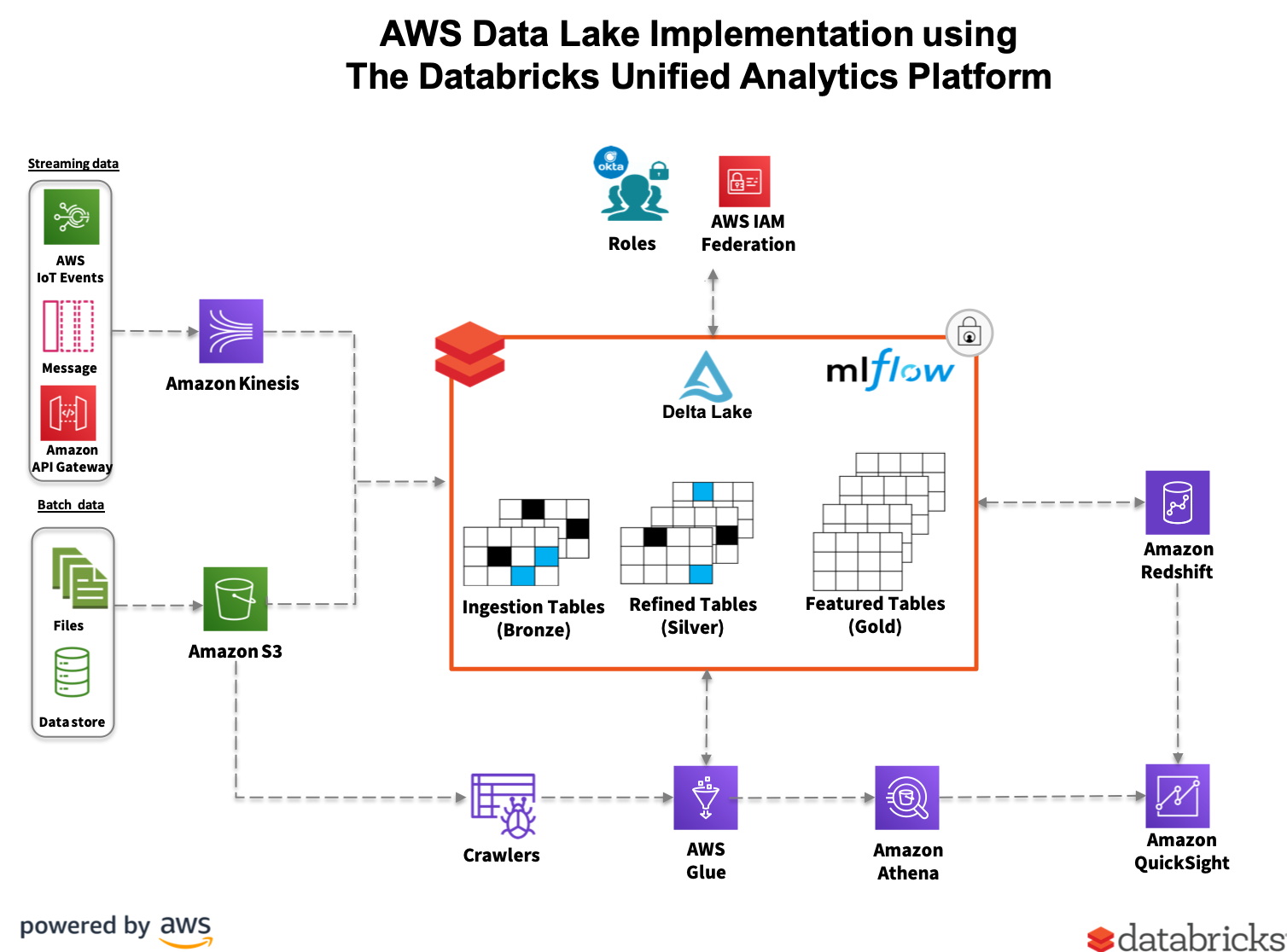
Kiến trúc Delta Lake là một cải tiến lớn so với kiến trúc Lambda truyền thống.

Ở mỗi giai đoạn, Delta-lake dữ liệu của mình thông qua một quy trình được kết nối cho phép chúng kết hợp luồng công việc theo lô(Batch workflow) và luồng(Stream) thông qua kho lưu trữ tệp được chia sẻ với các giao dịch tuân thủ ACID.

Delta lake sắp xếp dữ liệu của mình thành các lớp hoặc thư mục được xác định là Bronze, Silver và Gold như sau:

* Bảng Bronze có dữ liệu thô được nhập từ nhiều nguồn khác nhau (dữ liệu RDBMS, tệp JSON, dữ liệu IoT, v.v.).
* Các bảng Silver sẽ cung cấp một cái nhìn tinh tế hơn về dữ liệu của chúng tôi. Delta lake kết hợp các trường từ nhiều bảng Bronze khác nhau để cải thiện hồ sơ phát trực tuyến (streaming records) hoặc cập nhật trạng thái tài khoản dựa trên hoạt động gần đây.
* Bảng Gold cung cấp tổng hợp các bản báo cáo ở business-level thường được sử dụng để lập bảng điều khiển và báo cáo. Điều này sẽ bao gồm các tổng hợp như doanh số hàng tuần trên mỗi cửa hàng, người dùng trang web hoạt động hàng ngày hoặc tổng doanh thu mỗi quý của bộ phận. Kết quả cuối cùng là thông tin chi tiết có thể hành động, bảng điều khiển và báo cáo về các số liệu kinh doanh.**CHƯƠNG 2: MÔ HÌNH KẾT NỐI VỚI AWS**

## **2.1 Mô hình kết nối**

****

Nhờ vào delta lake kết hợp với các dịch vụ AWS mà chúng ta sẽ xây dựng được 1 hệ thống lakehouse có đầy đủ tính năng hoàn chỉnh.

Trong đó:

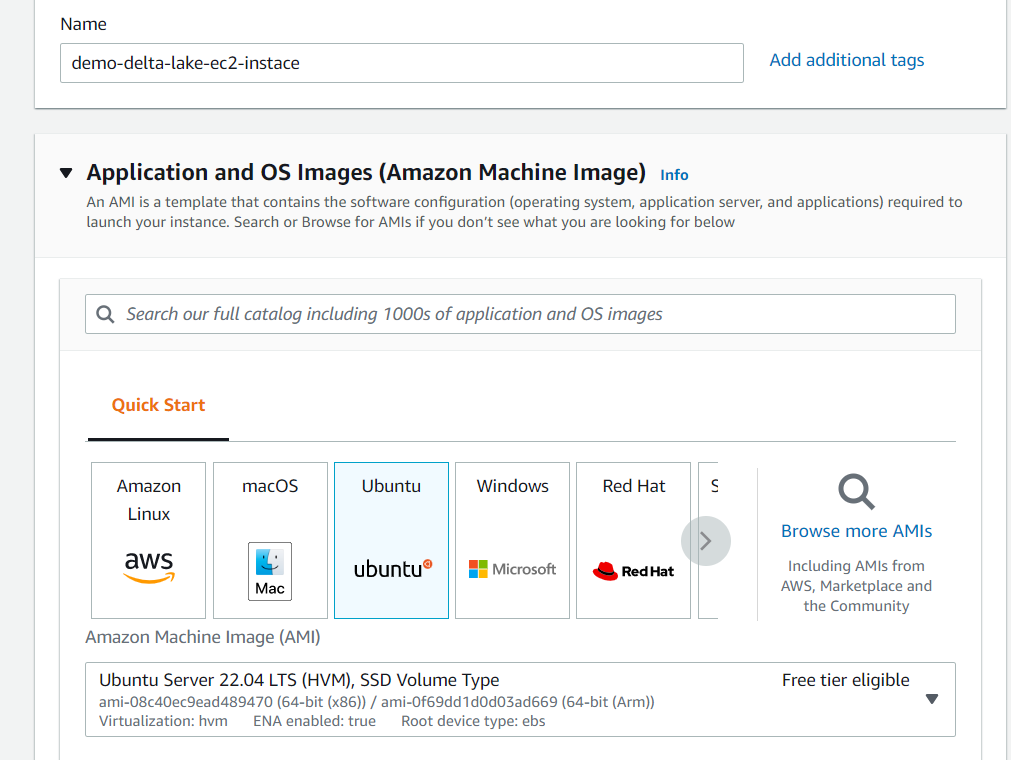
* S3 là nơi lưu trữ chính của các Batch data
* Kinesis là nơi xử lý tiếp nhận dữ liệu thời gian thực theo luồng từ các dịch vụ khác của aws hoặc ngoài aws.
* Delta lake được deploy trên các nền tảng của AWS có thể kiểm soát đăng nhập bảo mật bằng IAM và Role
* Tiếp là Delta lake sẽ kết hợp cũng AWS glue để xử lý dữ liệu
* Ridshift và Athena cho phép truy vấn sắp xếp lại dữ liệu để đưa ra QuickSight thành những bản báo cáo và gửi đến người dùng cuối

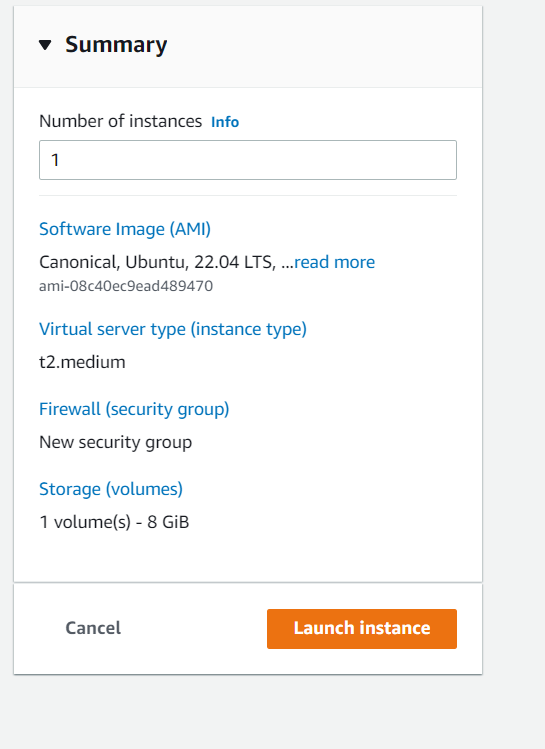
# **CHƯƠNG 3: CÀI ĐẶT, SỬ DỤNG DELTALAKE TRÊN AWS**

## **3.1 Cài đặt bằng EC2**

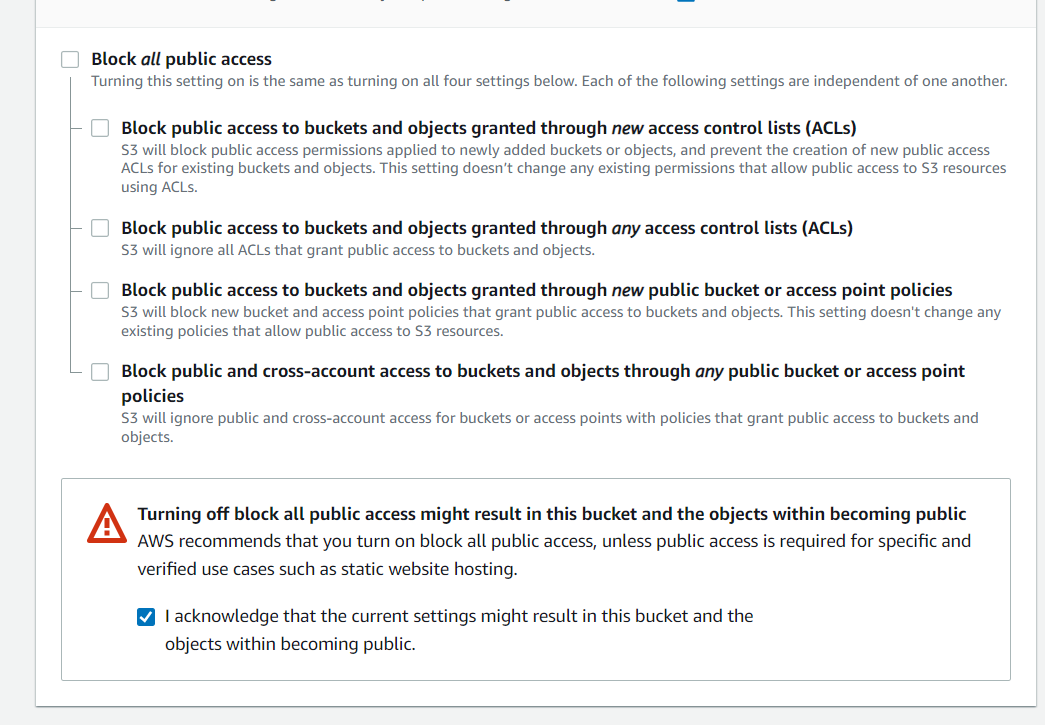
### **3.1.1 Tạo EC2 Instance**

Tạo EC2 instance và cấu hình như bên dưới, lưu ý rằng à cấu hình không được quá thấp ít nhất là phải có 2 vCPUs. Ở đây em chọn loại instance là: t2.medium, đó là máy chủ 2 vCPUs và 8GB bộ nhớ do AWS cung cấp.





### **3.1.2 Tạo S3 bucket**



Tạo bucket S3 như hình trên.

Lưu ý có thể để private nhưng do tài khoản lab leaner không cung cấp IAM để em tạo user IAM để access vào S3 nên phải để public để chương trình delta lake có thể truy cập vào được

{

"Version": "2012-10-17",

"Statement": [

{

"Sid": "Stmt1594969687722",

"Effect": "Allow",

"Principal": "\*",

"Action": "s3:\*",

"Resource": [

"arn:aws:s3:::delta-lake-ute",

"arn:aws:s3:::delta-lake-ute/\*"

]

}

]

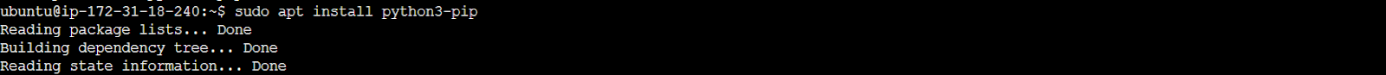
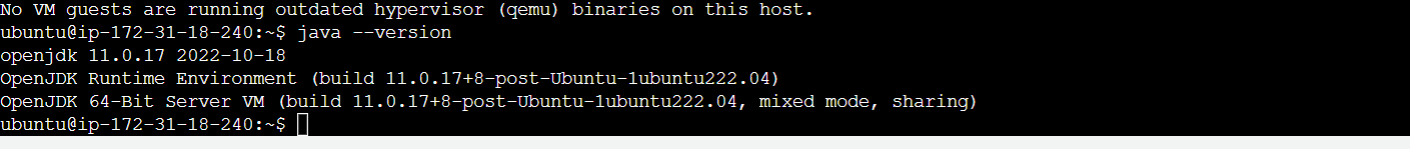
}

Config với đoạn code trên



### **3.1.3 Cài đặt môi trường delta lake trên EC2**

Cài đặt python và java.



Cài đặt pySpark

python3 –verion

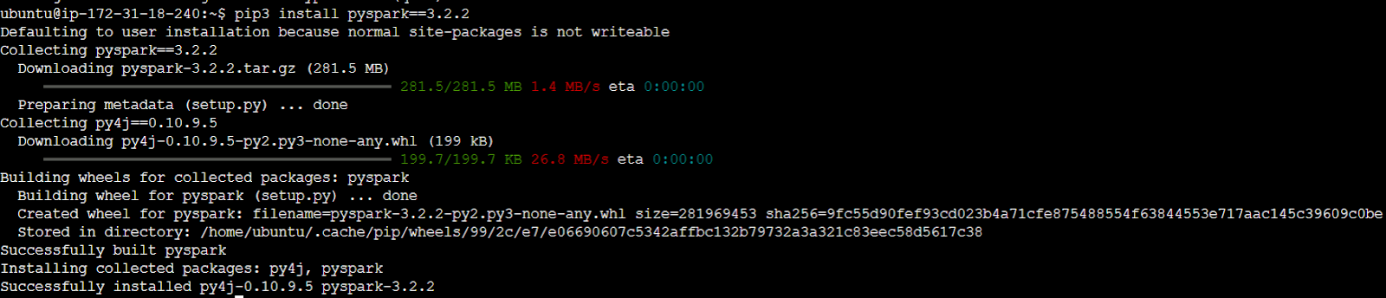
java –version

apt-get update

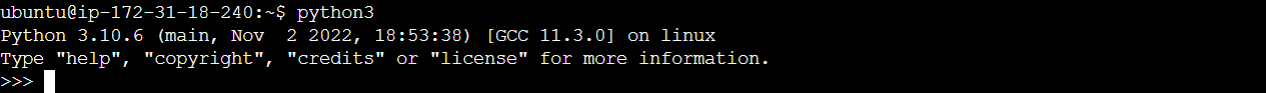
apt install openjdk-11-jre-headless

sudo apt install python3-pip

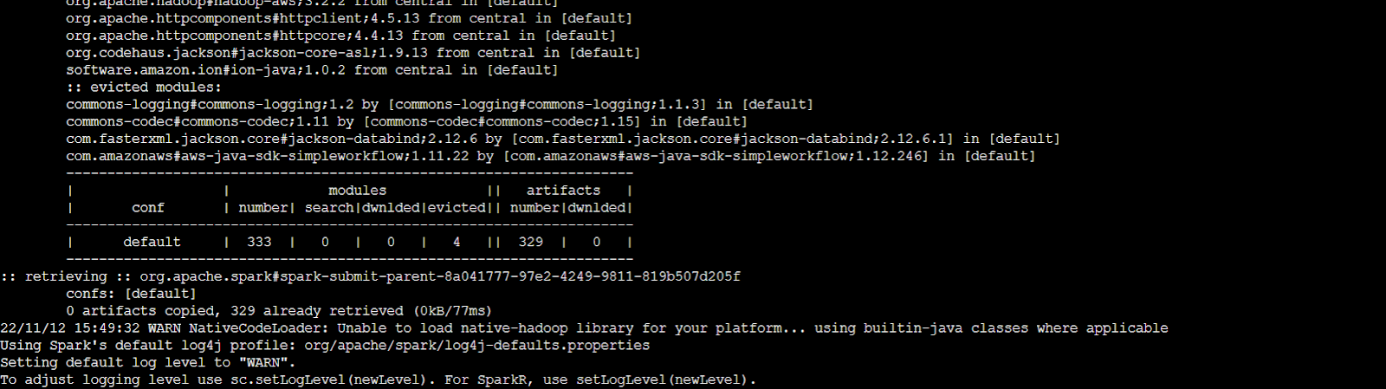
pip install pyspark==3.2.2



Cài đặt môi trường chạy PySpark và delta-lake

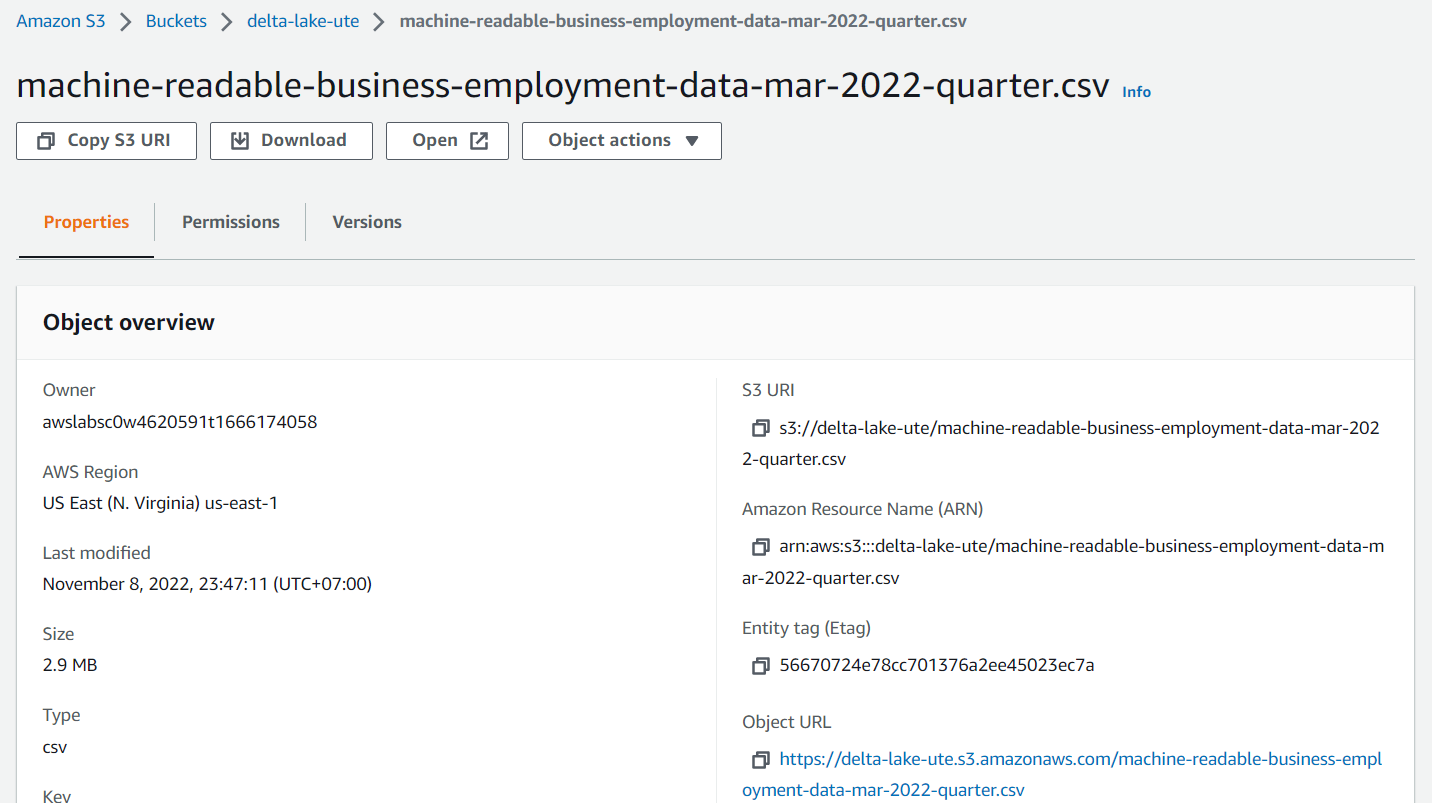






Import 1 file csv vào S3 để thực hiện truy vấn.

Import dữ liệu từ: https://stats.govt.nz/large-datasets/csv-files-for-download/



Thực hiện truy vấn để test hoạt động 

Sử dụng các lệnh sau như ảnh:

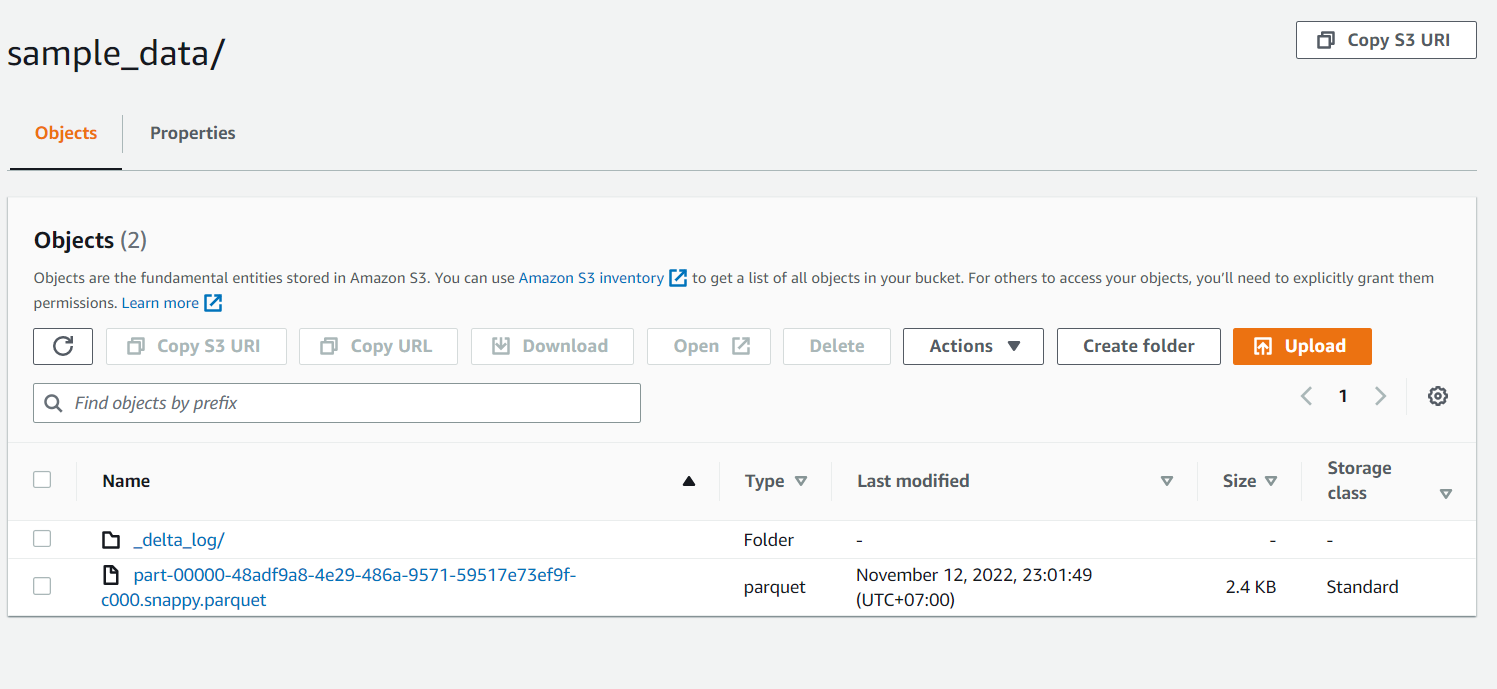
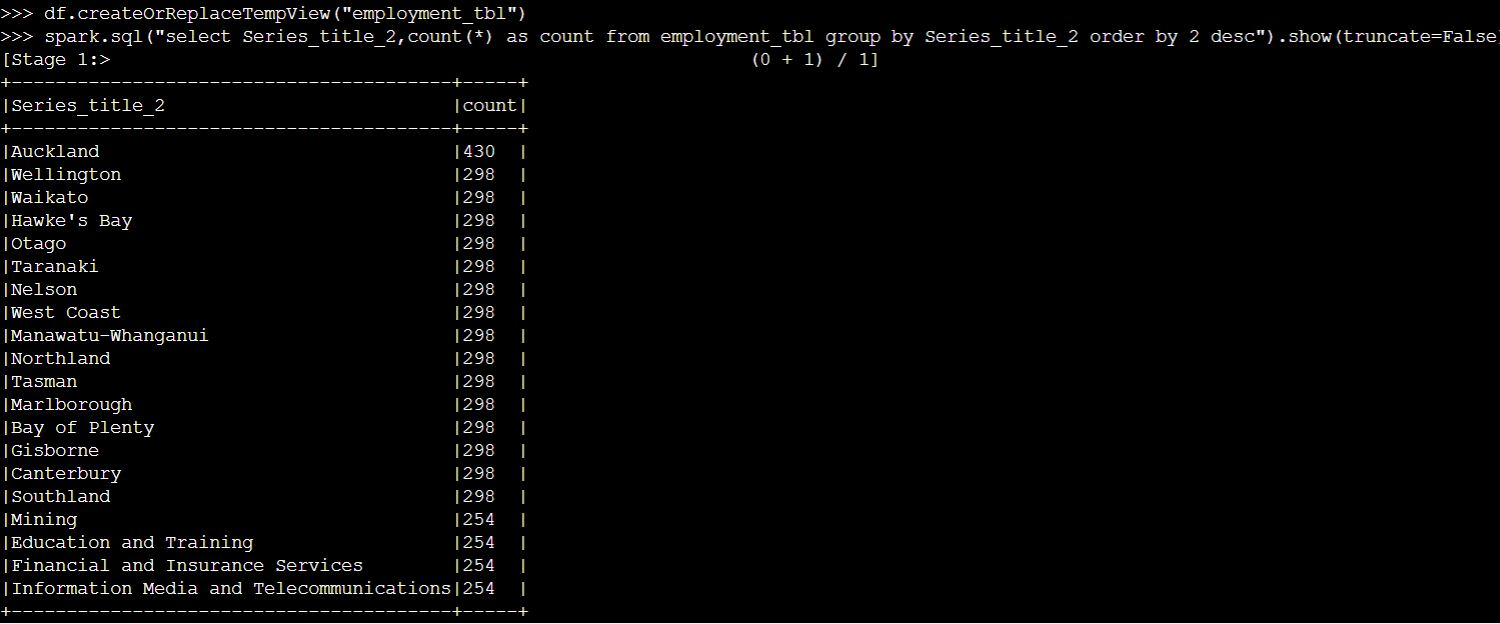
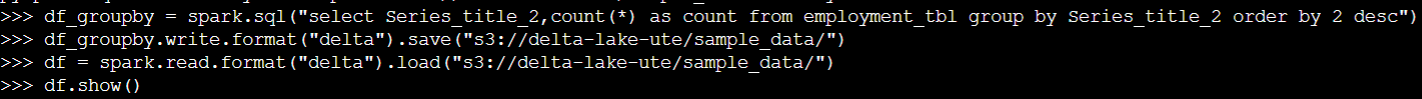
df = spark.read.option("recursiveFileLookup", "true").option("header","true").csv("s3://delta-lake-ute/machine-readable-business-employment-data-mar-2022-quarter.csv")

Sau đó thực hiện tạo một view tạm và thực hiện câu truy vấn:

df.createOrReplaceTempView("employment\_tbl")

spark.sql("select Series\_title\_2,count(\*) as count from employment\_tbl group by Series\_title\_2 order by 2 desc").show(truncate=False)

Mục đích của câu lệnh này là lấy ra các tile và đếm số lượng của nhân viên trong title đó từ file csv



Sử dụng các câu lệnh sau bằng spark và delta lake:

Tạo data frame:

df\_groupby = spark.sql("select Series\_title\_2,count(\*) as count from employment\_tbl group by Series\_title\_2 order by 2 desc")

Lưu data frame dưới dạng delta format:

df\_groupby.write.format("delta").save("s3://delta-lake-ute/sample\_data/")

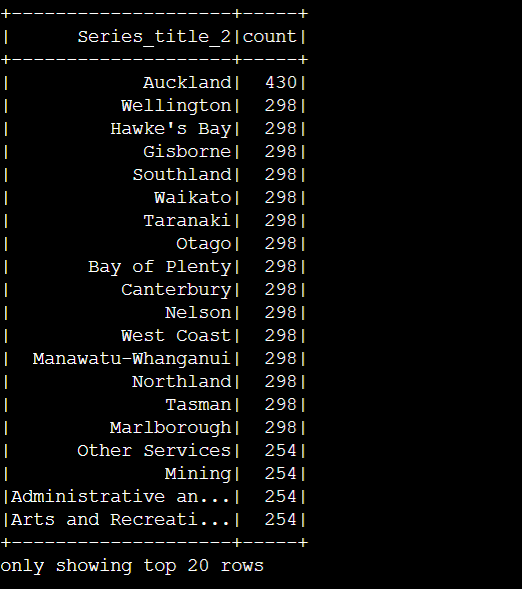
Đọc data delta format:

df = spark.read.format("delta").load("s3://delta-lake-ute/sample\_data/")

Show ra data đã đọc:

df.show()

Kiểm tra S3 ta thấy file có phần mở rộng là panquet đó là dạng file của delta làm việc Dạng file này sẽ cho tốc độ truy vấn nhanh gấp nhiều lần csv



## **3.2 Tạo bằng EMR**

Tạo 2 bucket và file boottrap với nội dung như trên:

Ute-delta-lake: dùng để lưu dữ liệu khi làm việc

Ute-erm-boostrap: dùng để lưu file cho việc bootstrap của emr service

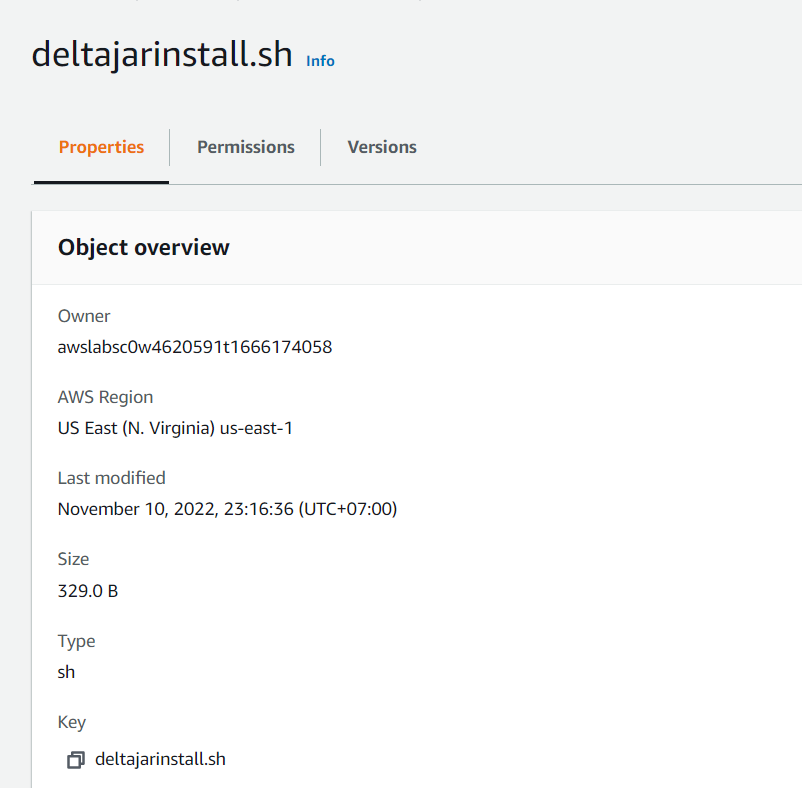
Tạo 1 file với tên là deltajarinstall.sh với nội dung:

#!/bin/bash

sudo curl -O --output-dir /usr/lib/spark/jars/ https://repo1.maven.org/maven2/io/delta/delta-core\_2.12/2.0.0/delta-core\_2.12-2.0.0.jar

sudo curl -O --output-dir /usr/lib/spark/jars/ https://repo1.maven.org/maven2/io/delta/delta-storage/2.0.0/delta-storage-2.0.0.jar

sudo python3 -m pip install delta-spark==2.0.0



Có thể tạo EMR instance với đoạn code như sau vào AWS cloudshell:

aws emr create-cluster \

--name "emr-delta-lake-blog" \

--release-label emr-6.7.0 \

--applications Name=Hadoop Name=Hive Name=Livy Name=Spark Name=JupyterEnterpriseGateway \

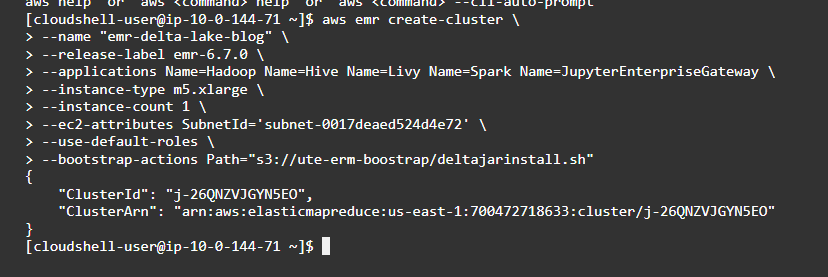
--instance-type m5.xlarge \

--instance-count 1 \

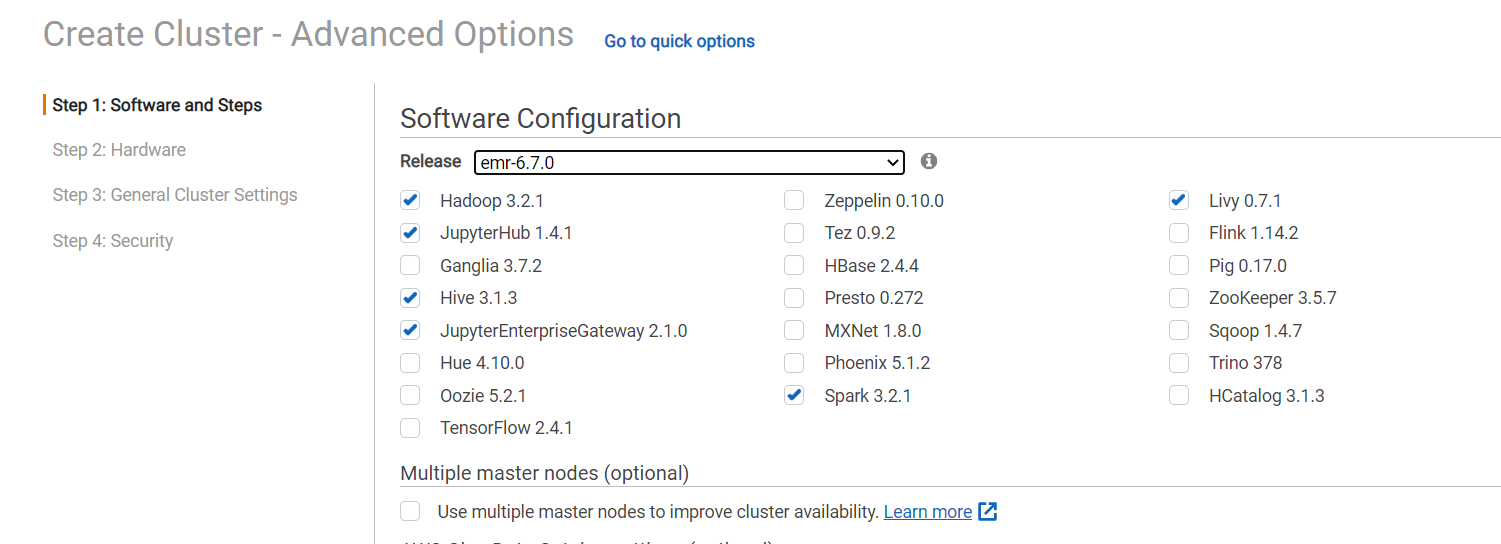
--ec2-attributes SubnetId='subnet-0017deaed524d4e72' \

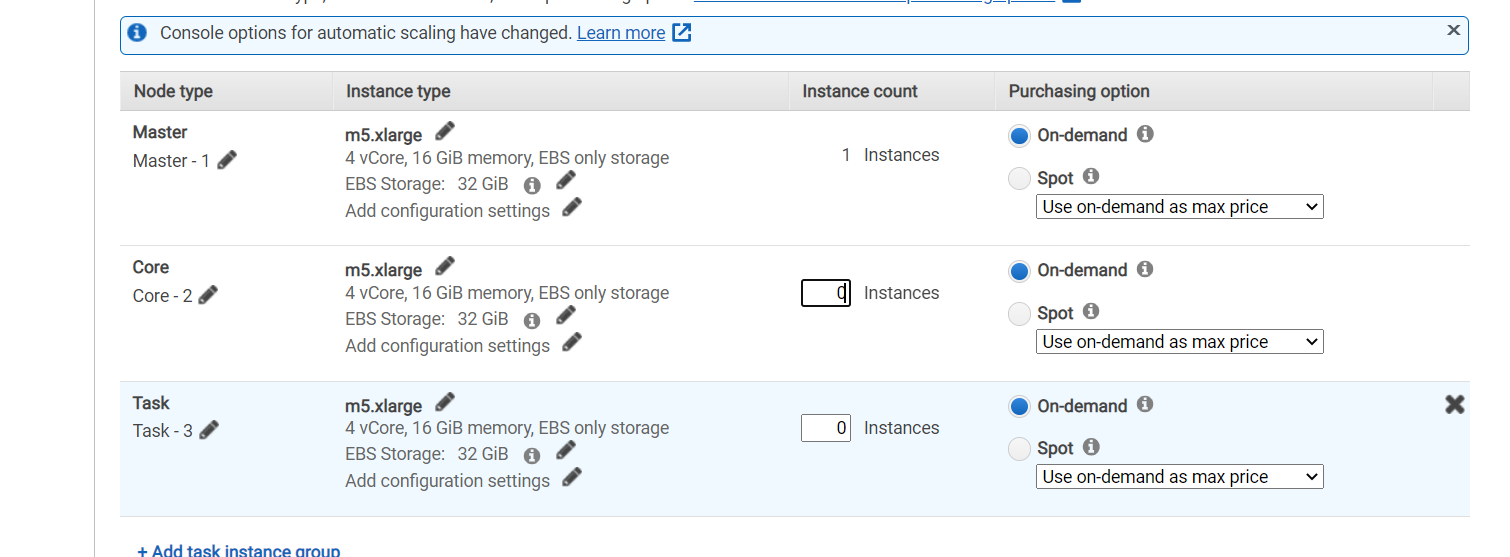
--use-default-roles \

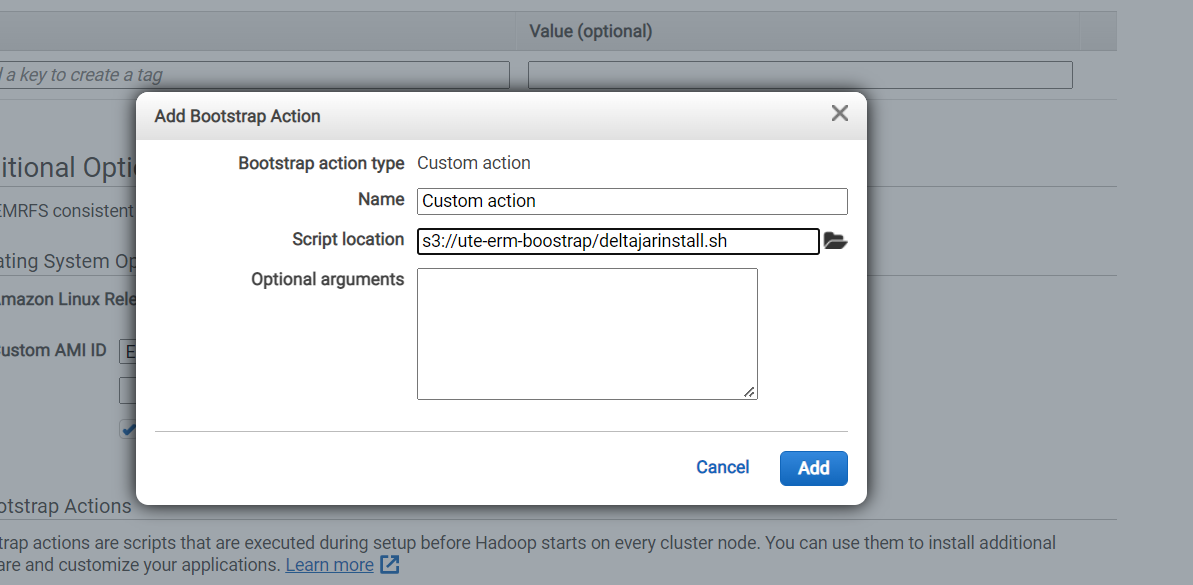
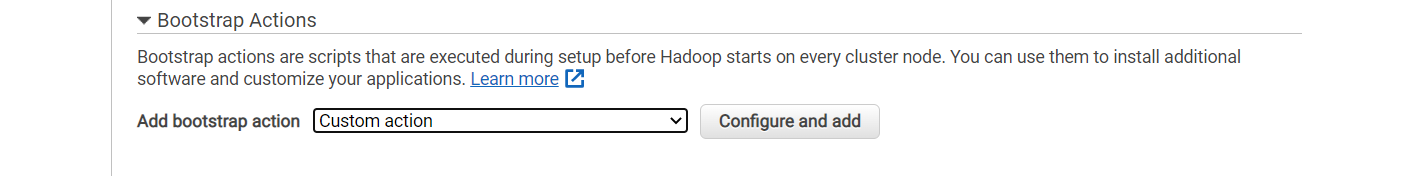
--bootstrap-actions Path="s3://ute-erm-boostrap/deltajarinstall.sh"

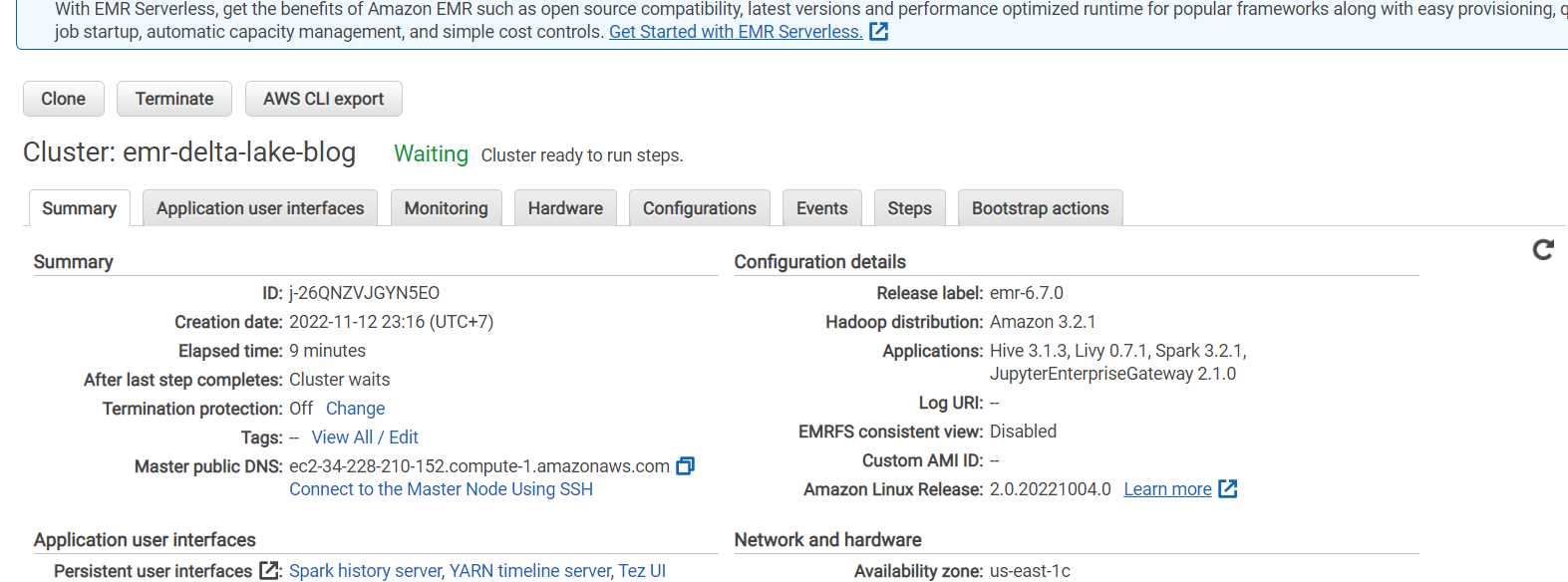
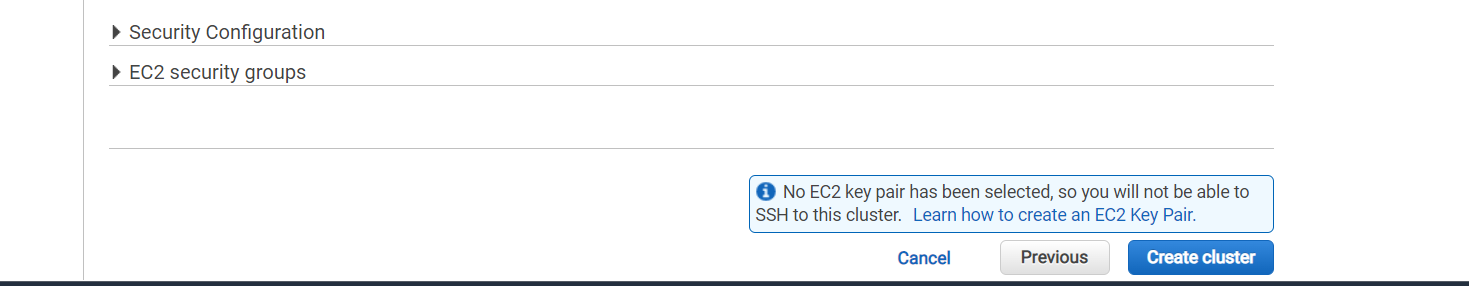


Hoặc tạo thủ công như các hình bên dưới

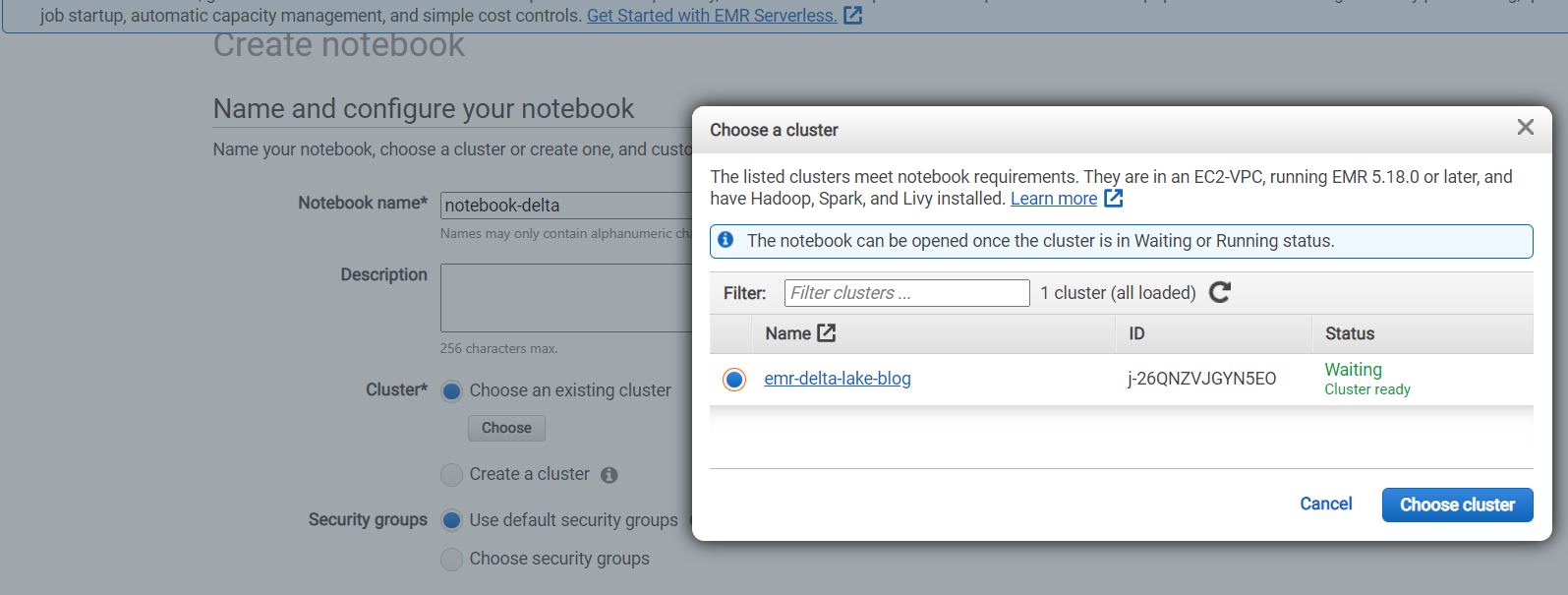
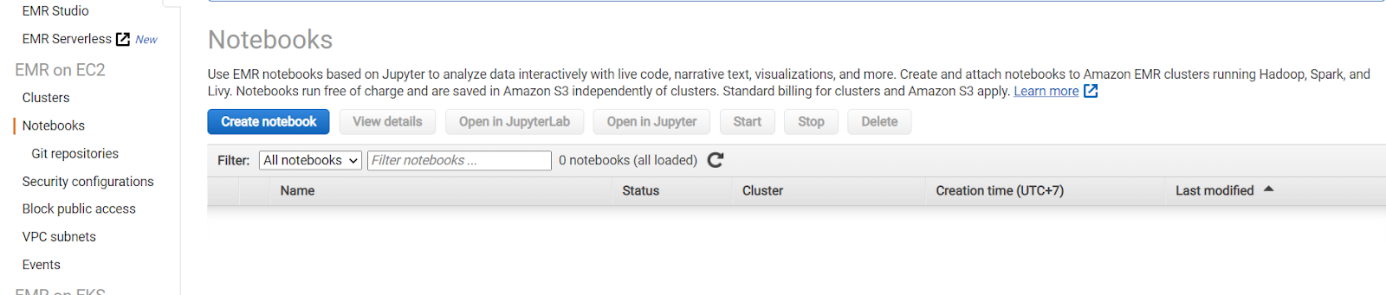


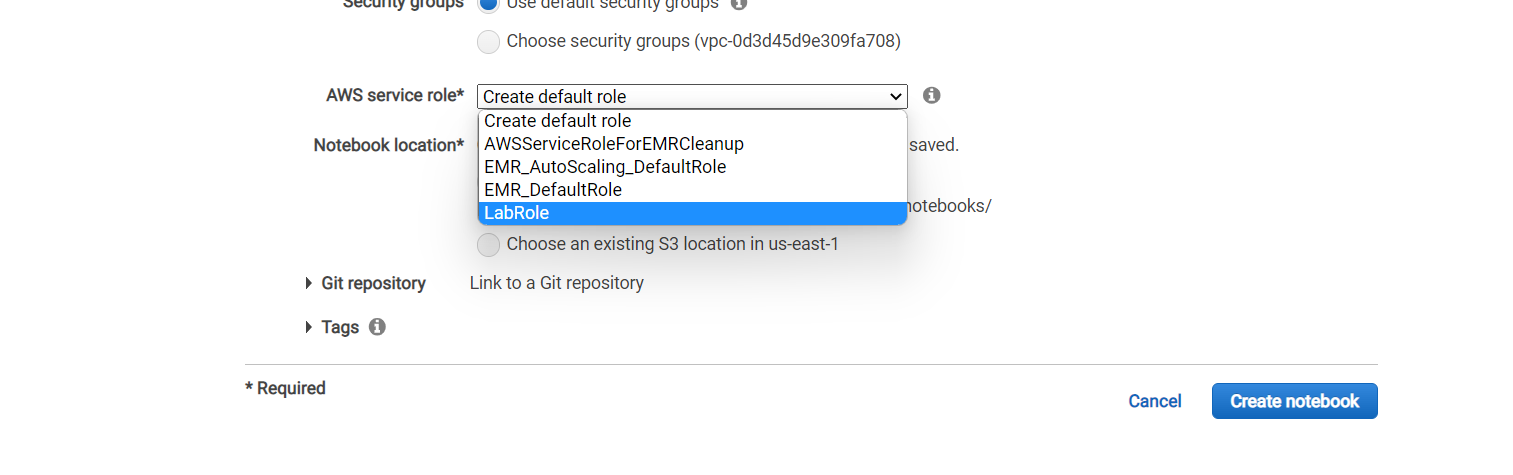




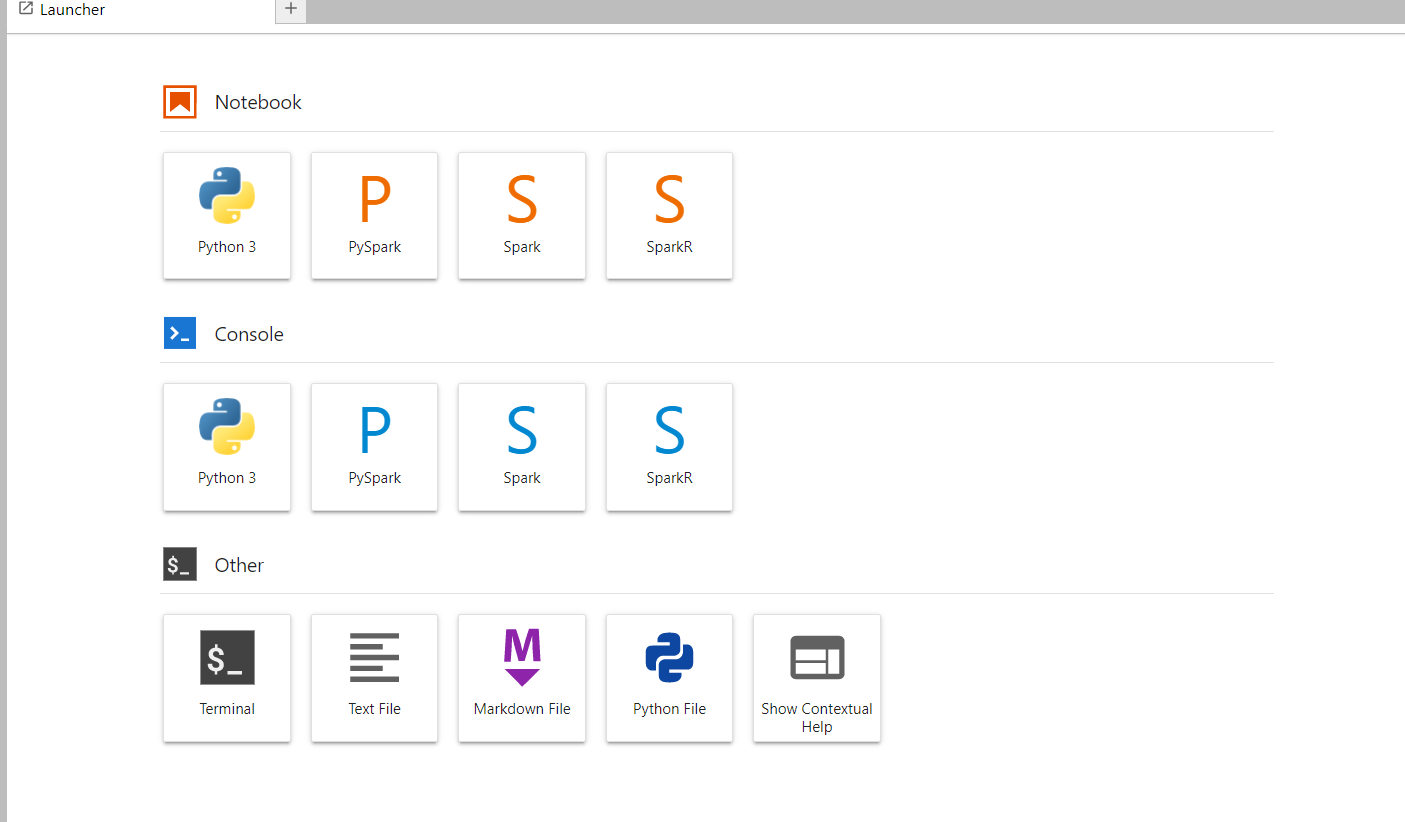


## **3.3 Tạo NoteBook để sử dụng khai thác dữ liệu**





Chọn PySpark kernel để sử dụng.



## **3.4 Test NoteBook hoạt động.**

Cài đặt chạy apache spark với delta core Sau đó import thư viện dùng để chạy các lệnh truy vấn delta:

%%configure -f

{

"conf": {

"spark.sql.extensions": "io.delta.sql.DeltaSparkSessionExtension",

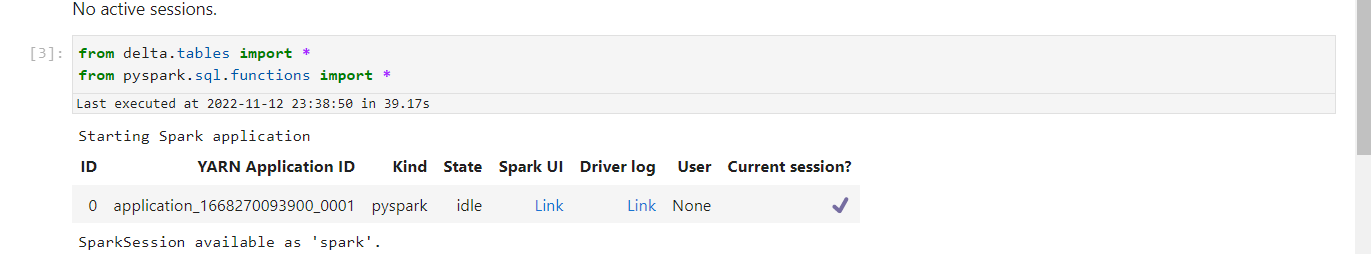
"spark.sql.catalog.spark\_catalog": "org.apache.spark.sql.delta.catalog.DeltaCatalog"

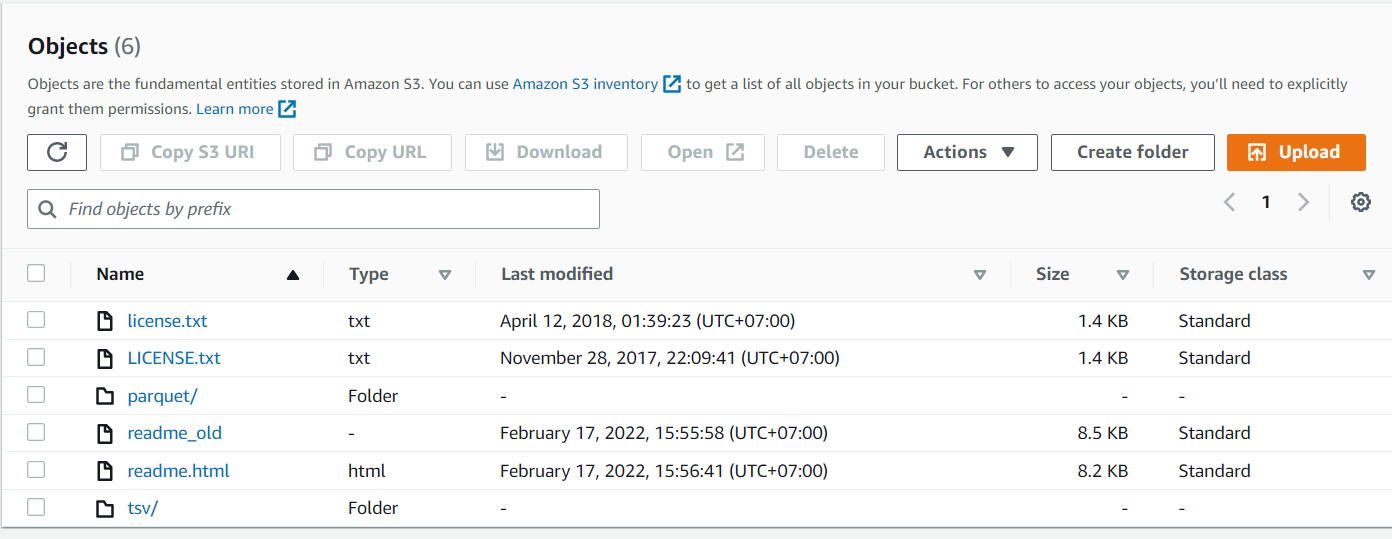
}

}

from delta.tables import \*

from pyspark.sql.functions import \*





Sử dụng bucket được public của aws(Amazon Product Reviews Dataset) để đọc và viết data delta: <https://us-east-1.console.aws.amazon.com/s3/home?region=us-east-1&bucket=amazon-reviews-pds>

Set deltaPath là vị trí s3 bucket được tạo ởbước 1 Kiểm tra spark đọc file banquet từ public bucket:

deltaPath = "s3://ute-delta-lake/delta-amazon-reviews-pds/"

df\_parquet = spark.read.parquet("s3://amazon-reviews-pds/parquet/product\_category=Gift\_Card/\*.parquet"

df\_parquet.printSchema()

Tiến hành lưu các file theo định dạng delta xuống bucket S3 đã tạo ở bước 1 Kiểm tra thấy lưu thành công nghĩa là việc cài đặt đã hoàn tất thành công:

df\_parquet.write.mode("overwrite").format("delta").partitionBy("year").save(deltaPath)

Dùng spark để đọc dữ liệu lên Dùng lệnh show để tiến hành show ra dữ liệu đã đọc:

df\_delta = spark.read.format("delta").load(deltaPath)

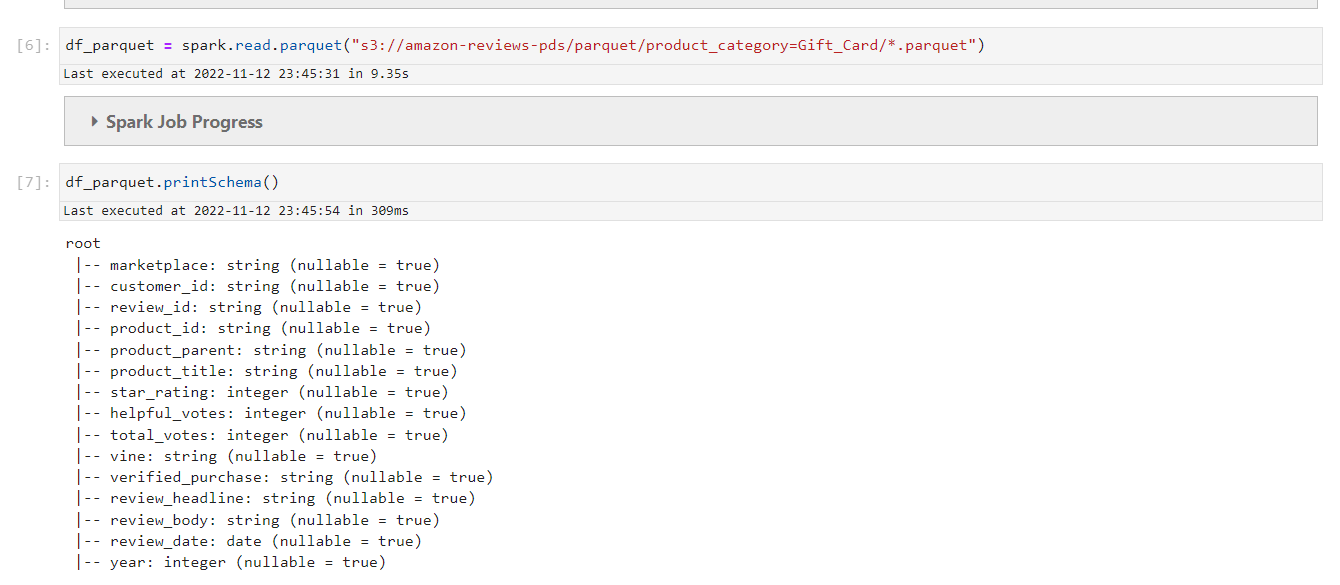
df\_delta.show()

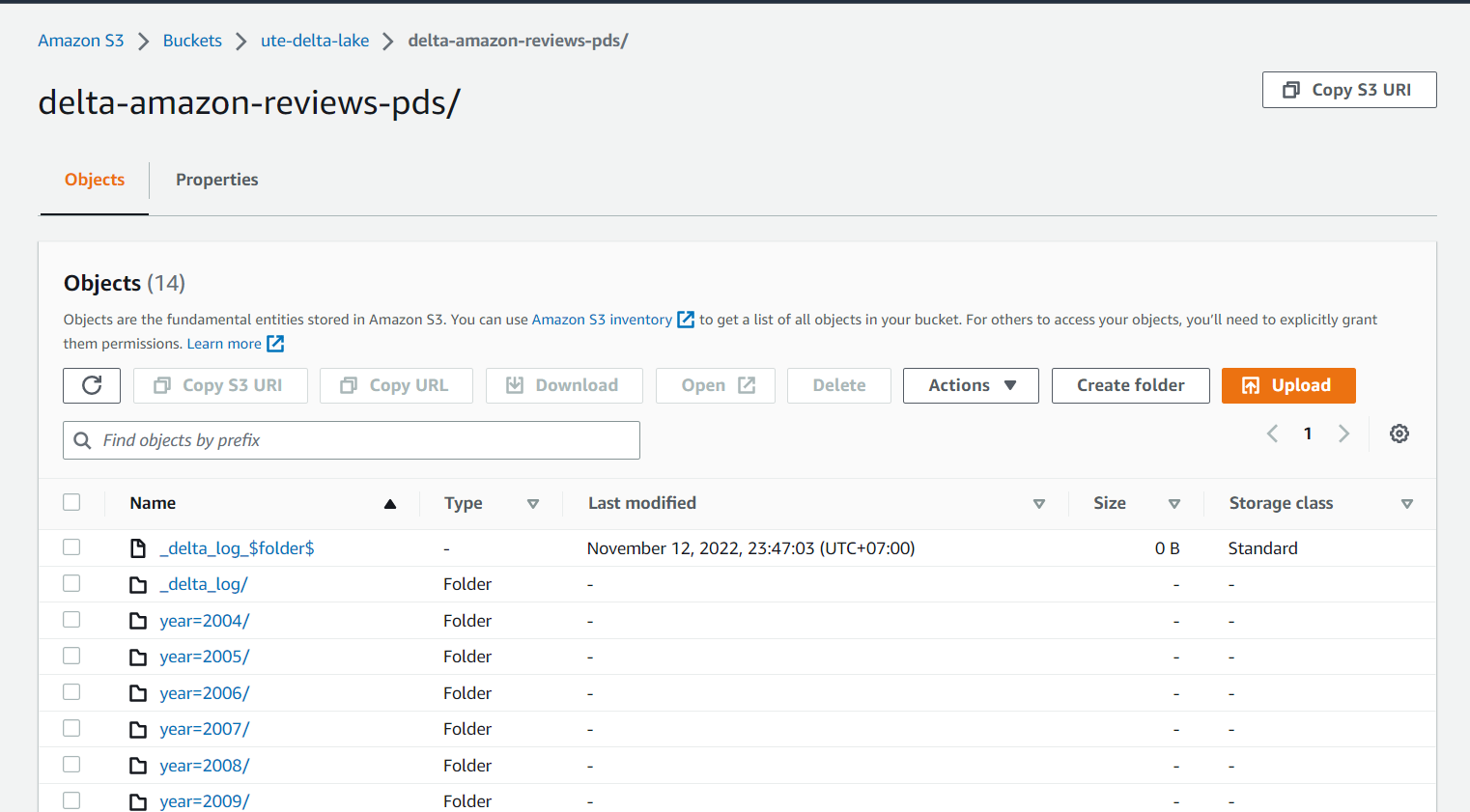
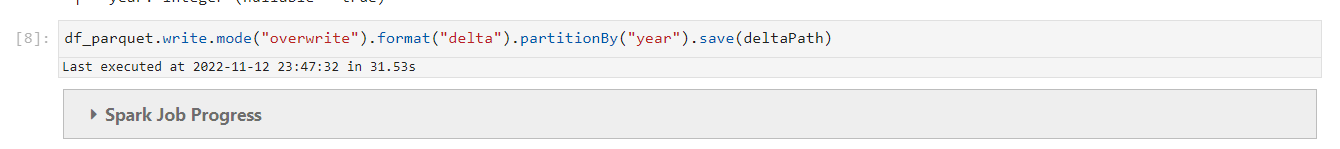
%%sql

SELECT \* FROM delta.`s3://ute-delta-lake/delta-amazon-reviews-pds/` LIMIT 10

df\_delta.createOrReplaceTempView("aws\_product\_review")

spark.sql("select marketplace,customer\_id,review\_date from aws\_product\_review LIMIT 30").show(30)

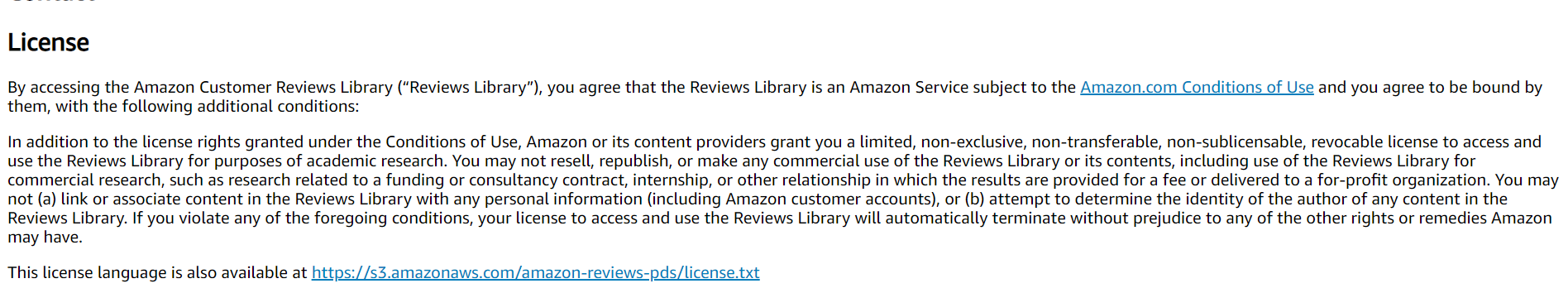


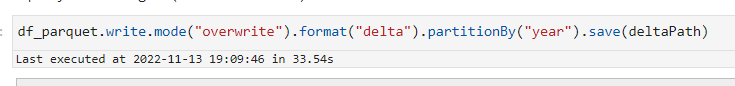




## **3.5 Demo khai thác dữ liệu.**

### **3.5.1 Chuẩn bị dữ liệu**





Sử dụng lại kho dữ liệu của amazone customer review dataset được quyền sủ dụng cho mục đích học thuật:

<https://us-east-1.console.aws.amazon.com/s3/home?region=us-east-1&bucket=amazon-reviews-pds>

Sau đó chúng ta đọc toàn bộ file .parquet từ danh mục Gift\_Card của dữ liệu aws public và lưu vào df\_parquet Sau đó chúng ta sử dụng SQL để lưu về bucket đã chuẩn bị từ trước của chúng ta theo từng năm.

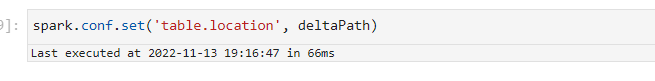
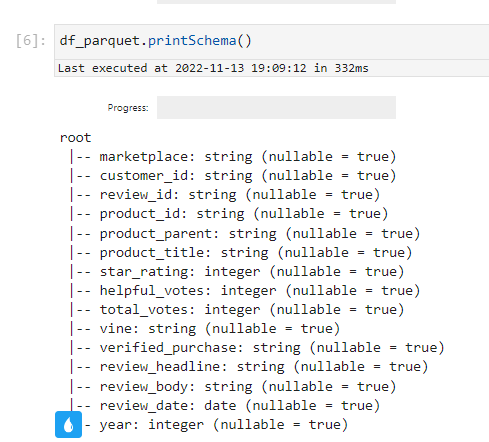
deltaPath = "s3://ute-delta-lake/delta-amazon-reviews-pds/"

df\_parquet = spark.read.parquet("s3://amazon-reviews-pds/parquet/product\_category=Gift\_Card/\*.parquet")

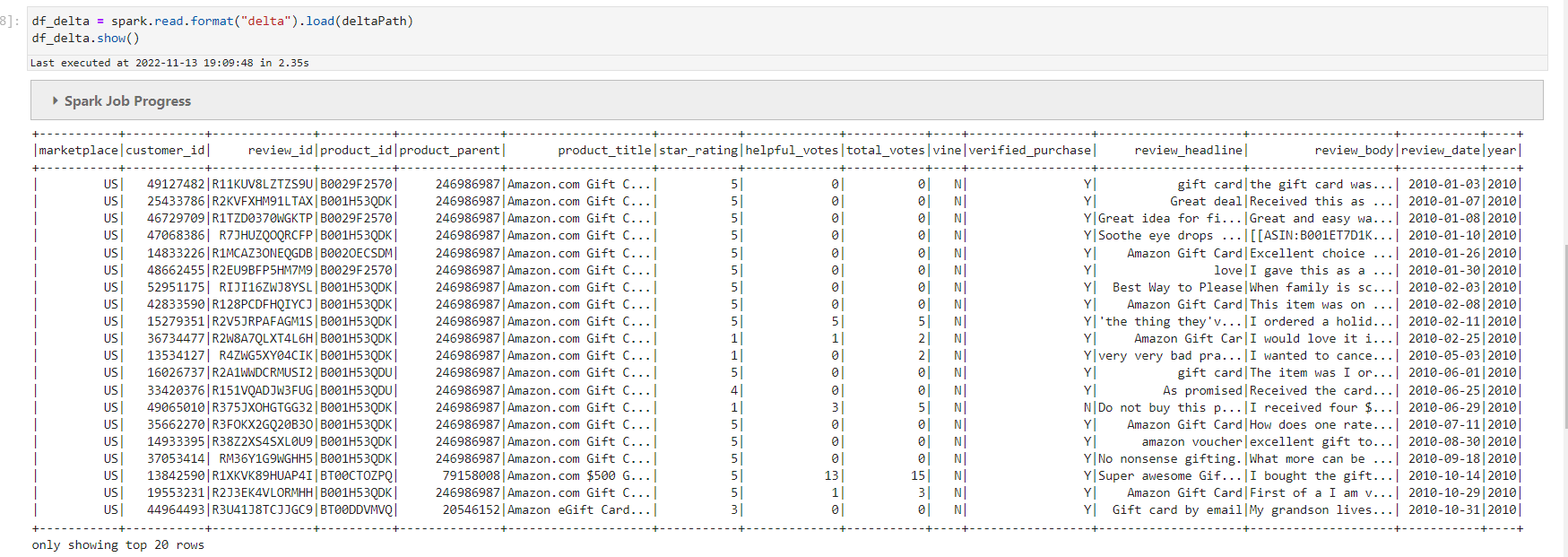
df\_parquet.printSchema()

df\_parquet.write.mode("overwrite").format("delta").partitionBy("year").save(deltaPath)

spark.conf.set('table.location', deltaPath)



### **3.5.2 Đọc dữ liệu**



Dùng spark để đọc dữ liệu lên Dùng lệnh show để tiến hành show ra dữ liệu đã đọc

df\_delta = spark.read.format("delta").load(deltaPath)

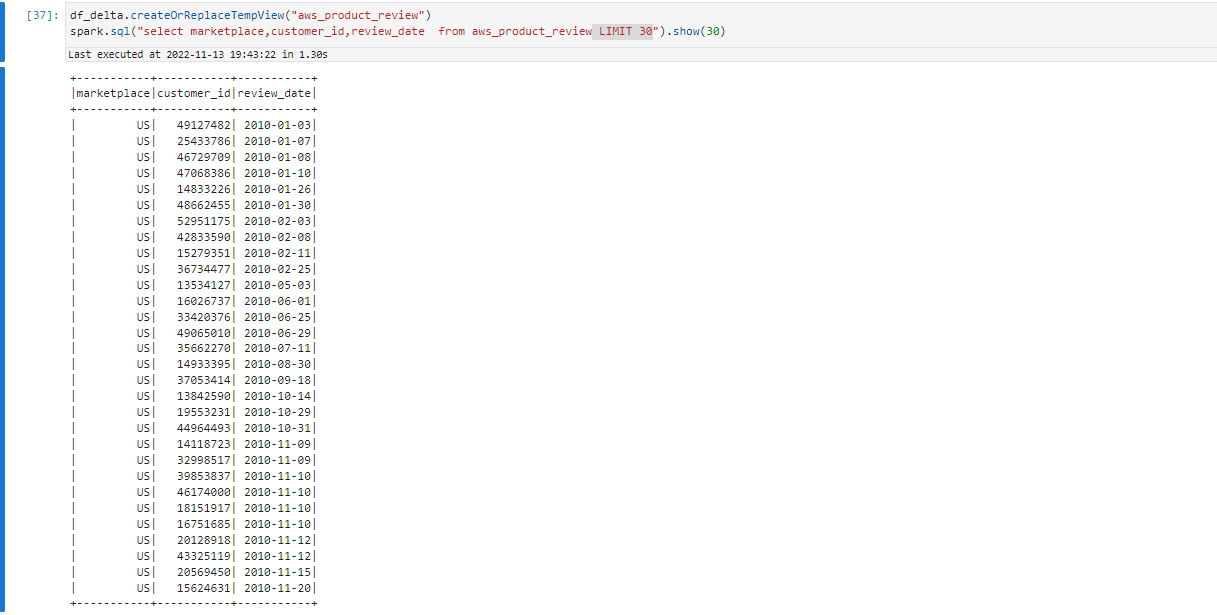
df\_delta.show()

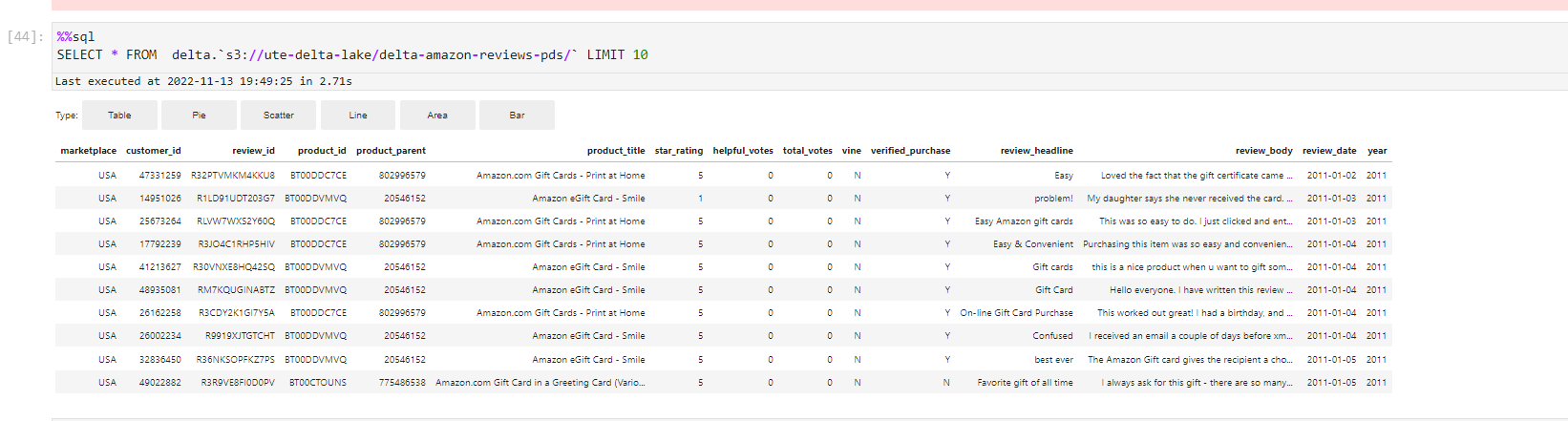
%%sql

SELECT \* FROM delta.`s3://ute-delta-lake/delta-amazon-reviews-pds/` LIMIT 10

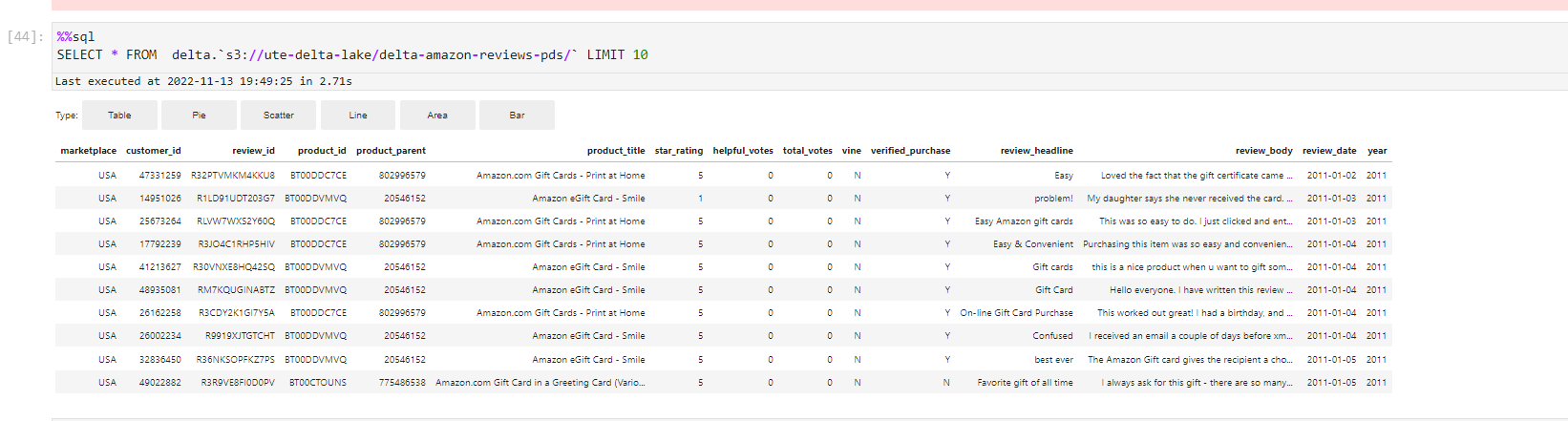
df\_delta.createOrReplaceTempView("aws\_product\_review")

spark.sql("select marketplace,customer\_id,review\_date from aws\_product\_review LIMIT 30").show(30)





### **3.5.3 Cập nhật dữ liệu**



Chuyển US sang USA:

deltaTable = DeltaTable.forPath(spark, deltaPath) #khai báo 1 datatable

deltaTable.update("marketplace = 'US'",{ "marketplace":"'USA'"}) # Thay đổi toàn bộ marketplace US thành USA

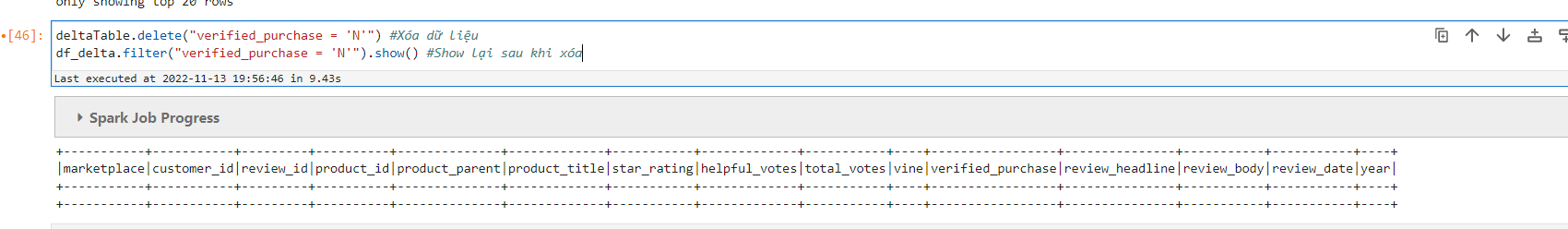
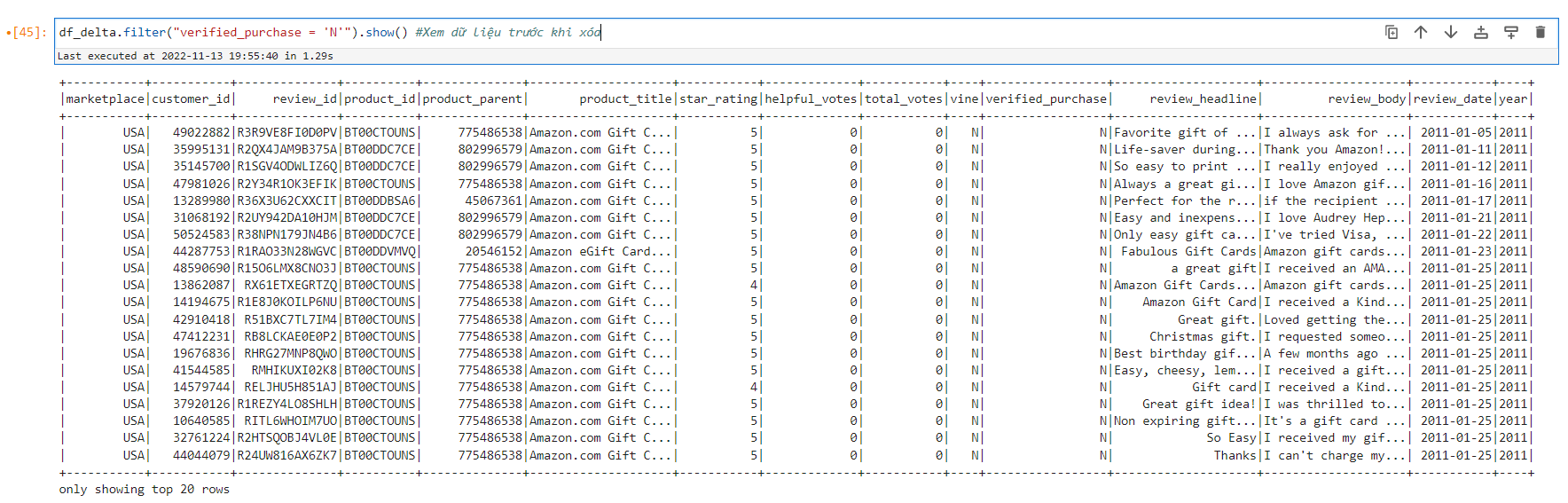
#hoặc

%%sql

update delta.`s3://ute-delta-lake/delta-amazon-reviews-pds/`

set marketplace = 'USA' where marketplace = 'US'

### **3.5.4 Xóa dữ liệu**



df\_delta.filter("verified\_purchase = 'N'").show() #Xem dữ liệu trước khi xóa

deltaTable.delete("verified\_purchase = 'N'") #Xóa dữ liệu

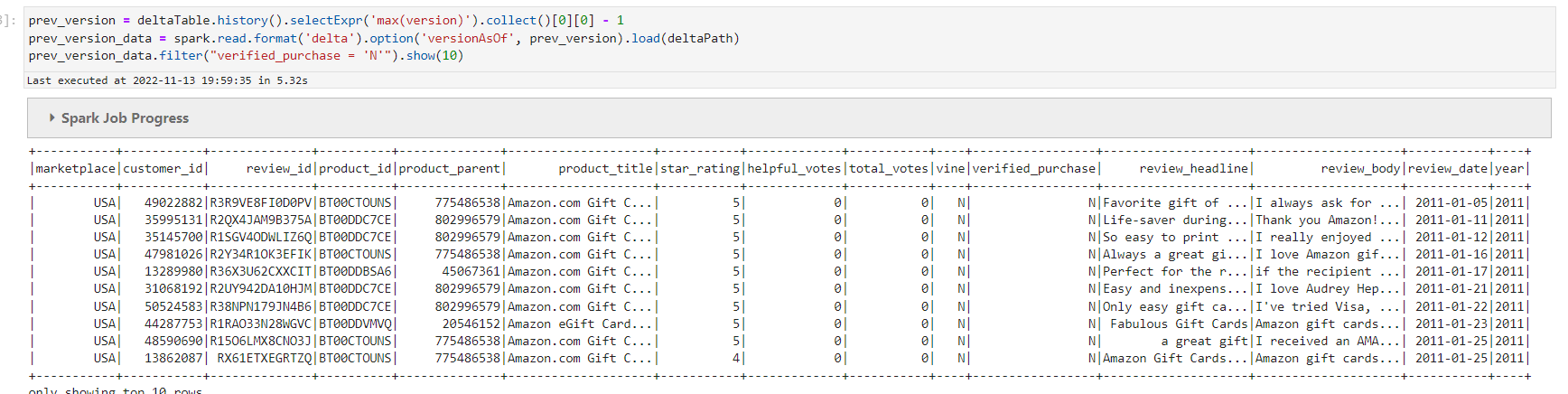
df\_delta.filter("verified\_purchase = 'N'").show() #Show lại sau khi xóa

Lưu ý rằng phương thức xóa chỉ loại bỏ dữ liệu khỏi phiên bản mới nhất của bảng. Những hồ sơ này vẫn có mặt trong các snap shot cũ hơn của dữ liệu.

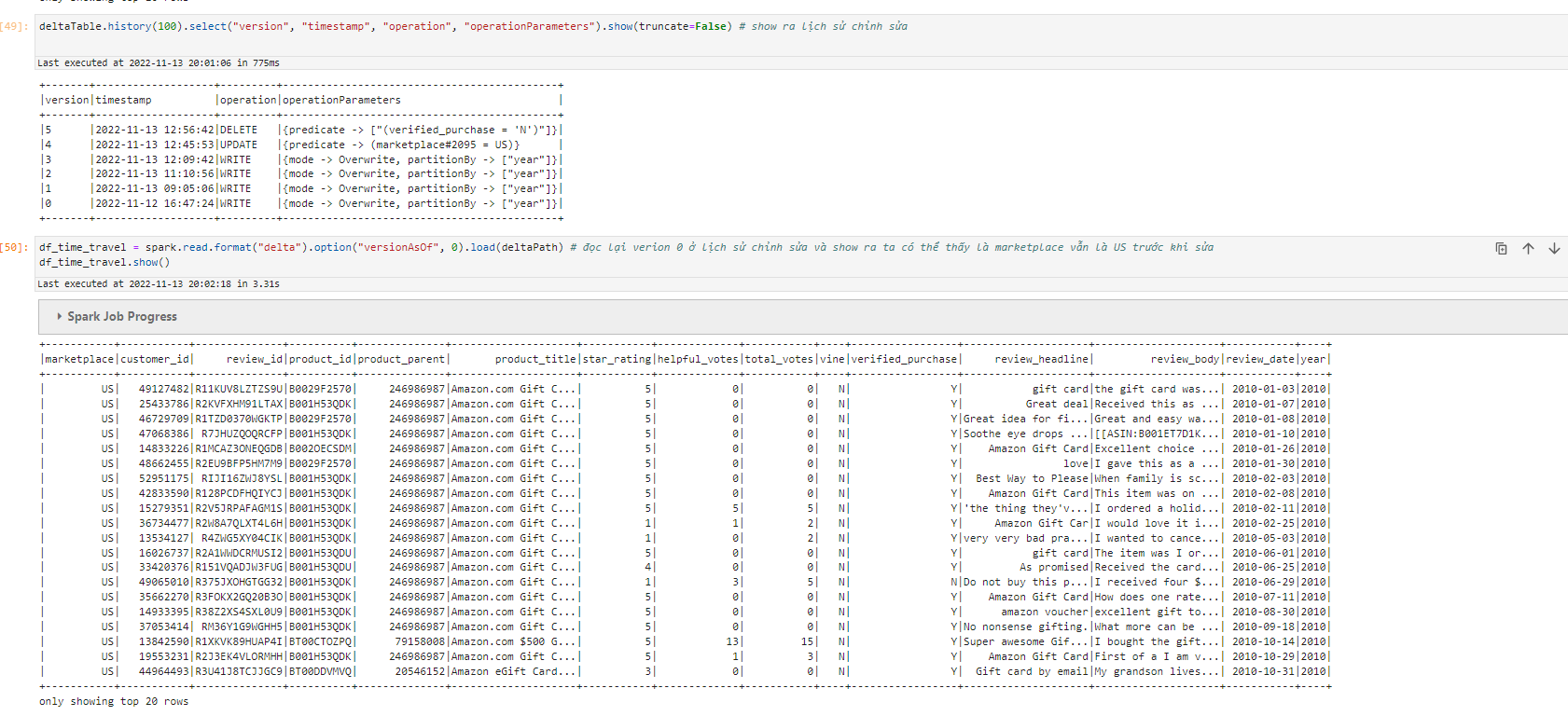
prev\_version = deltaTable.history().selectExpr('max(version)').collect()[0][0] - 1

prev\_version\_data = spark.read.format('delta').option('versionAsOf', prev\_version).load(deltaPath)

prev\_version\_data.filter("verified\_purchase = 'N'").show(10)



### **3.5.5 Du hành thời gian (Time travel)**

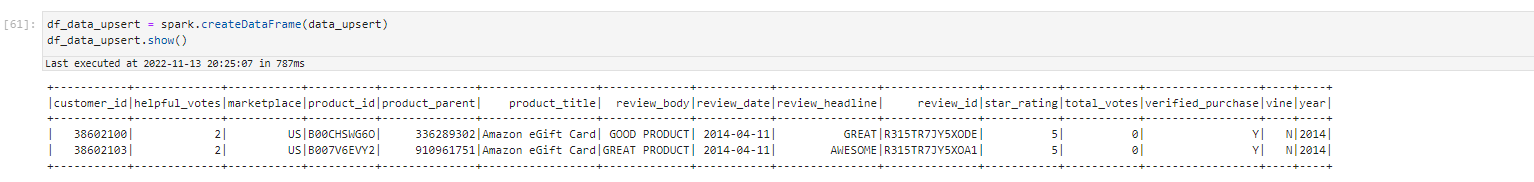
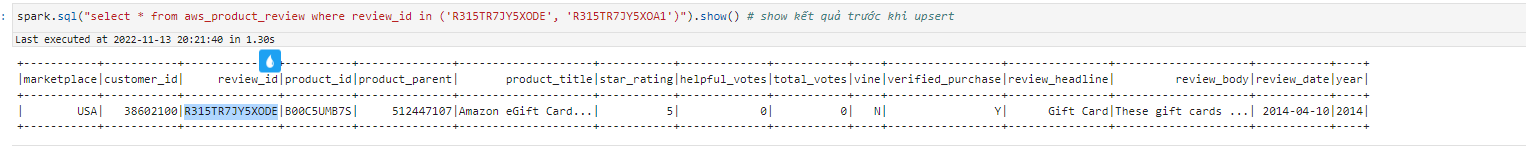


deltaTable.history(100).select("version", "timestamp", "operation", "operationParameters").show(truncate=False) # show ra lịch sử chỉnh sửa

df\_time\_travel = spark.read.format("delta").option("versionAsOf", 0).load(deltaPath) # đọc lại verion 0 ở lịch sử chỉnh sửa và show ra ta có thể thấy là marketplace vẫn là US trước khi sửa

df\_time\_travel.show()

### **3.5.6 Upsert dữ liệu**



Tạo ra 1 list gồm 2 item để thực hiện upsert update XODE và insert XOA1 vì nó chưa tồn tại Sau đó tạo 1 frame chứa data để upsert:

spark.sql("select \* from aws\_product\_review where review\_id in ('R315TR7JY5XODE', 'R315TR7JY5XOA1')").show() # show kết quả trước khi upsert

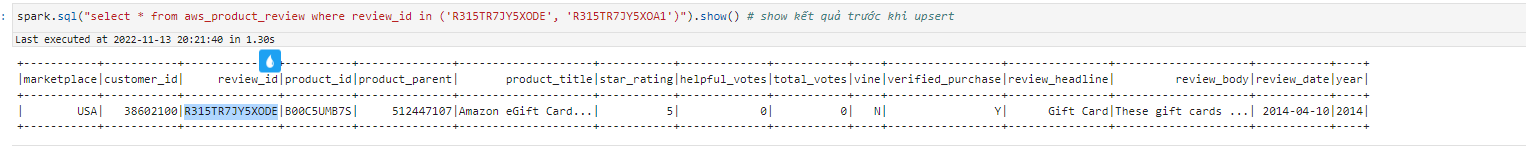
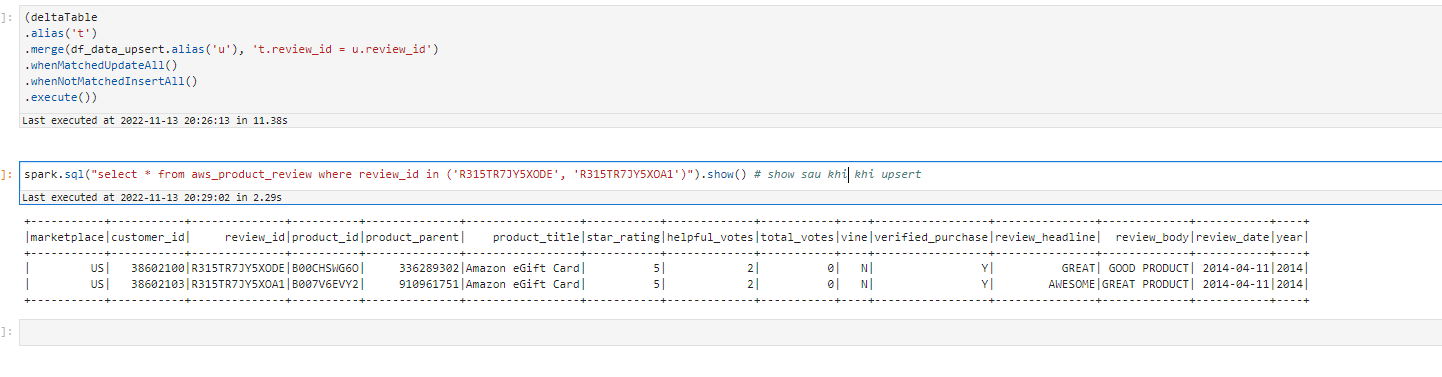
data\_upsert = [ {"marketplace":'US',"customer\_id":'38602100', "review\_id":'R315TR7JY5XODE',"product\_id":'B00CHSWG6O',"product\_parent":'336289302',"product\_title" :'Amazon eGift Card', "star\_rating":'5', "helpful\_votes":'2',"total\_votes":'0',"vine":'N',"verified\_purchase":'Y',"review\_headline":'GREAT',"review\_body":'GOOD PRODUCT',"review\_date":'2014-04-11',"year":'2014'},

{"marketplace":'US',"customer\_id":'38602103', "review\_id":'R315TR7JY5XOA1',"product\_id":"B007V6EVY2","product\_parent":'910961751',"product\_title" :'Amazon eGift Card', "star\_rating":'5', "helpful\_votes":'2',"total\_votes":'0',"vine":'N',"verified\_purchase":'Y',"review\_headline":'AWESOME',"review\_body":'GREAT PRODUCT',"review\_date":'2014-04-11',"year":'2014'}

]

df\_data\_upsert = spark.createDataFrame(data\_upsert)

df\_data\_upsert.show()



(deltaTable

.alias('t')

.merge(df\_data\_upsert.alias('u'), 't.review\_id = u.review\_id')

.whenMatchedUpdateAll()

.whenNotMatchedInsertAll()

.execute())

spark.sql("select \* from aws\_product\_review where review\_id in ('R315TR7JY5XODE', 'R315TR7JY5XOA1')").show() # show sau khi khi upsert

# **PHẦN KẾT LUẬN**

## **Kết quả đạt được**

Sau một thời gian nghiên cứu và thực hiện đề tài “***Tìm hiểu Delta-lake và mô hình kêt nối với aws***”, nhóm chúng em đã đạt được những kết quả như sau:

### **1.1. Kiến thức tìm hiểu được**

Nắm bắt được các kiến thức cũng như những vấn đề liên quan trọng về Delta-lake, t và áp dụng kiến thức để thiết kế và xây dựng một Lakehouse. Biết được cách Delta-lake hoạt động.

Nắm bắt được các dịch vụ của aws để xây dụng lakehouse.

### **1.2. Chương trình đã làm được**

Xây dựng hoàn chỉnh một data lakehouse bằng deltalake với các chức năng cơ bản như:

* Thực hiện các câu lệnh truy vấn và tiến hành phân tích từ các câu lệnh truy vấn đó.
* Khai thác dữ liệu thông qua delta-lake trên notebook sử dụng dịch vụ EMR của AWS

## **Ưu điểm**

* Hoạt động của delta-lake diễn ra một cách trơn tru và chính xác, không xảy ra tình trạng lỗi trong hệ thống.
* Lưu trữ được lượng dữ liệu lớn.
* Xử lý thông tin, truy vấn dữ liệu chính xác và nhanh chóng.
* ACID transaction

## **Hướng phát triển**

* Tiếp tục hoàn thiện các chức năng còn thiếu.
* Xây dựng và quản lý data lakehouse lớn hơn nhiều dịch vụ.
* Thực hiện thêm nhiều các câu truy vấn.
* Đánh giá chi phí sử dụng trên AWS

# **TÀI LIỆU THAM KHẢO**

[1]. *Getting started with Delta Lake & Spark in AWS*

Link:https://towardsdatascience.com/getting-started-with-delta-lake-spark-in-aws-the-easy-way-9215f2970c58

[2]. *Build a high-performance, transactional data lake using open-source Delta Lake on Amazon EMR*

Link:<https://aws.amazon.com/vi/blogs/big-data/build-a-high-performance-transactional-data-lake-using-open-source-delta-lake-on-amazon-emr/>

[3]. *Building a Data Lake on AWS*

Link: https://garystafford.medium.com/building-a-simple-data-lake-on-aws-df21ca092e32