

BỘ GIÁO DỤC VÀ ĐÀO TẠO  
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM  
KHOA CÔNG NGHỆ THÔNG TIN



**ĐỒ ÁN CUỐI KỲ**

**TRIỂN KHAI HỆ THỐNG LAKEHOUSE TÍCH HỢP LUU  
TRỮ TRÊN HDFS, XỬ LÝ BẰNG SPARK VÀ AIRFLOW,  
TRỰC QUAN HÓA VỚI POWER BI CHO DỮ LIỆU  
STREAMING VÀ BATCH**

**MÃ MÔN HỌC: BDAN333977\_01**

**HỌC KỲ 2 – NĂM HỌC 2024-2025**

**Thực hiện:** Nhóm 3

**Giảng viên hướng dẫn:** Th.S Lê Thị Minh Châu

*Thành phố Hồ Chí Minh, Tháng 5 năm 2025*

**BỘ GIÁO DỤC VÀ ĐÀO TẠO  
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT  
THÀNH PHỐ HỒ CHÍ MINH  
KHOA CÔNG NGHỆ THÔNG TIN  
BỘ MÔN PHÂN TÍCH DỮ LIỆU LỚN**

**CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM  
Độc lập- Tự do - Hạnh phúc**

*TP. HCM, tháng 5 năm 2025*

**DANH SÁCH THÀNH VIÊN NHÓM LÀM ĐỒ ÁN CUỐI KỲ  
MÔN PHÂN TÍCH DỮ LIỆU LỚN  
HỌC KỲ II NĂM HỌC 2024-2025**

- Mã lớp môn học:** BDAN333977\_01
- Giảng viên hướng dẫn:** Th.S Lê Thị Minh Châu
- Tên đề tài:**
- Danh sách thành viên nhóm:**

HỌ VÀ TÊN SINH VIÊN	Mã số sinh viên	Tỉ lệ tham gia %	Kí tên
Trần Diễm Quỳnh	22133046	100	
Phạm Quỳnh Thư	23133060	100	
Nguyễn Thị Hồng Thơ	21151305	100	
Nguyễn Ngọc Hiếu Hảo	23133015	100	
Vy Gia Nghi	22133037	100	

- Tỷ lệ % = 100%

- Trưởng nhóm: Nguyễn Thị Hồng Thơ

### *Nhận xét của giảng viên*

*Tháng 5 năm 2025*

## Giảng viên chấm điểm

## BẢNG PHÂN CÔNG

MSSV	Họ và tên	Nhiệm vụ
22151305	Nguyễn Thị Hồng Thơ	<ul style="list-style-type: none"> <li>- Cài đặt Apache Kafka</li> <li>- Xử lý luồng dữ liệu streaming từ API (Producer)</li> <li>- Xử lý Data Streaming qua 3 lớp: Bronze - Silver - Gold</li> <li>- Phát triển mô hình dự báo SARIMA từ dữ liệu lớp Gold</li> </ul>
22133046	Trần Diễm Quỳnh	<ul style="list-style-type: none"> <li>- Xử lý dữ liệu trong các lớp Silver</li> <li>- Trực quan bằng Power BI</li> <li>- Cài đặt Airflow</li> <li>- Thiết lập pipeline ETL tự động cho dữ liệu Batch</li> </ul>
22133060	Phạm Quỳnh Thư	<ul style="list-style-type: none"> <li>- Xây dựng mô hình phân loại mức độ ô nhiễm không khí bằng thuật toán Random Forest từ dữ liệu lớp Silver</li> <li>- Xây dựng Lakehouse 3 lớp sử dụng Azure và Databrick</li> <li>- Xây dựng mô hình dự đoán chỉ số ô nhiễm PM25 dùng CatBoost</li> <li>- Tạo biểu đồ so sánh thực tế và dự đoán chỉ số ô nhiễm PM25</li> </ul>
22133015	Nguyễn Ngọc Hiếu Hảo	<ul style="list-style-type: none"> <li>- Cài đặt Apache Airflow</li> <li>- Cài đặt Apache Kafka</li> <li>- Thiết lập pipeline ETL tự động cho cả dữ liệu batch và streaming</li> </ul>
22133037	Vy Gia Nghi	<ul style="list-style-type: none"> <li>- Đưa dữ liệu tinh vào hệ thống HDFS</li> <li>- Xử lý dữ liệu qua lớp Bronze</li> <li>- Xử lý dữ liệu qua lớp Silver</li> <li>- Trực quan hóa dữ liệu bằng Power BI</li> <li>- Tổng kết nội dung, viết báo cáo tổng hợp</li> </ul>

MỞ ĐẦU .....	6
CHƯƠNG I – LÝ THUYẾT .....	7
I – Công nghệ và công cụ .....	7
1. Apache Spark .....	7
2. Apache Airflow .....	7
3. Apache Kafka .....	8
4. HDFS (Hadoop Distributed File System) .....	9
5. Delta Lake .....	11
6. Nền tảng triển khai trên Cloud .....	13
II – Kiến trúc dữ liệu và lưu trữ trong mô hình Lakehouse .....	15
1. Lưu trữ 3 lớp Lake house .....	15
2. Giải thích tập dữ liệu .....	18
3. Tổng quan kiến trúc Lakehouse trên Ubuntu .....	23
CHƯƠNG II – THỰC HÀNH .....	26
I – CHUẨN BỊ MÔI TRƯỜNG .....	26
1. AirFlow .....	26
2. Kafka .....	41
3. Chuẩn bị các thư mục .....	45
4. Delta Data .....	47
II – XÂY DỰNG VÀ KIỂM THỦ LAKEHOUSE THỦ CÔNG .....	48
1. Đưa dữ liệu tĩnh vào hệ thống HDFS .....	48
2. Xử lý và làm sạch dữ liệu qua lớp Bronze .....	50
3. Xử lý dữ liệu qua lớp Silver .....	60
4. Xây dựng mô hình từ dữ liệu lớp Silver .....	70
5. Phát triển mô hình dự báo SARIMA từ dữ liệu lớp Gold .....	87
6. Trực quan hóa dữ liệu bằng PowerBI .....	105
III – TRIỂN KHAI VÀ TỰ ĐỘNG HÓA LAKEHOUSE .....	111
1. Thiết lập pipeline ETL tự động cho cả dữ liệu batch và streaming .....	111

2. Xử lý luồng dữ liệu streaming từ API (Producer) .....	156
3. Xử lý Data Streaming qua 3 lớp: Bronze - Silver – Gold .....	171
V – MỞ RỘNG TRIỀN KHAI LAKEHOUSE TRÊN CLOUD (AZURE + DATABRICKS) .....	184
1. Mục tiêu tổng thể.....	185
2. Quy trình triển khai theo kiến trúc 3 lớp (Bronze – Silver – Gold) trên Cloud ..	185
KẾT LUẬN .....	221
TÀI LIỆU THAM KHẢO .....	223

## MỞ ĐẦU

Trong kỷ nguyên số hóa và dữ liệu lớn (Big Data) hiện nay, việc thu thập, lưu trữ, xử lý và phân tích dữ liệu một cách hiệu quả đã trở thành yếu tố then chốt quyết định lợi thế cạnh tranh của các tổ chức và doanh nghiệp. Các kiến trúc dữ liệu truyền thống như Data Warehouse và Data Lake riêng lẻ đôi khi gặp phải những hạn chế về tính linh hoạt, khả năng quản lý giao dịch, hoặc chất lượng dữ liệu. Kiến trúc Lakehouse ra đời như một giải pháp tối ưu, kết hợp những ưu điểm của cả Data Warehouse (quản lý dữ liệu có cấu trúc, giao dịch ACID, hiệu suất truy vấn cao) và Data Lake (khả năng lưu trữ dữ liệu đa dạng, chi phí thấp, tính linh hoạt).

Dự án này tập trung vào việc "Tìm hiểu và triển khai cách tạo pipeline tự động hóa Lakehouse, tạo thành một Lakehouse thống nhất". Mục tiêu cốt lõi là xây dựng một hệ thống xử lý dữ liệu đầu cuối, từ thu thập dữ liệu batch (CSV) và streaming (API) đến các bước tiền xử lý, biến đổi, làm giàu dữ liệu qua các lớp Bronze, Silver, Gold và Platinum, cuối cùng là phục vụ cho mục đích phân tích và trực quan hóa thông qua Power BI.

Báo cáo này sẽ trình bày chi tiết các bước thực hiện, từ cấu hình môi trường, cài đặt công cụ, thiết kế pipeline, triển khai các DAGs trong Airflow, xử lý dữ liệu qua các lớp Bronze, Silver, Gold, Platinum, đến việc tích hợp dữ liệu streaming và trực quan hóa kết quả. Mỗi bước sẽ đi kèm với các minh chứng cụ thể về cấu hình và kết quả đạt được, nhằm mang lại cái nhìn toàn diện và thực tiễn về quá trình xây dựng một Lakehouse hiện đại.

# CHƯƠNG I – LÝ THUYẾT

## I – Công nghệ và công cụ

### 1. Apache Spark

**Mục đích:** Apache Spark là một hệ thống xử lý dữ liệu phân tán và xử lý dữ liệu lớn theo thời gian thực hoặc theo lô. Spark được thiết kế để làm việc trên các tập dữ liệu lớn và có thể được triển khai trên các cụm máy tính phân tán. Nó nhanh hơn nhiều so với các hệ thống xử lý dữ liệu theo lô như Hadoop MapReduce vì Spark xử lý dữ liệu chủ yếu trong bộ nhớ (in-memory computing).

**Chức năng chính:**

- **Xử lý dữ liệu theo lô (Batch Processing):** Spark có thể xử lý lượng lớn dữ liệu được lưu trữ trong các file hệ thống hoặc các cơ sở dữ liệu theo từng đợt (batch), giúp thực hiện các phép toán phức tạp như phân tích dữ liệu lịch sử hoặc tổng hợp báo cáo.
- **Xử lý dữ liệu theo dòng (Stream Processing):** Spark Streaming giúp xử lý dữ liệu theo dòng khi nó được tạo ra, rất hữu ích cho các ứng dụng như giám sát hệ thống, phân tích dữ liệu cảm biến, phân tích log thời gian thực, v.v.
- **Machine Learning (MLlib):** Spark cung cấp thư viện MLlib cho phép chạy các thuật toán học máy như phân loại, hồi quy, phân nhóm trên lượng dữ liệu lớn.
- **GraphX:** Đây là thư viện của Spark dùng để xử lý và phân tích đồ thị, rất hữu ích trong các ứng dụng như phân tích mạng xã hội, dự báo liên kết.
- **DataFrame và SQL:** Spark có một API mạnh mẽ cho phép làm việc với dữ liệu có cấu trúc thông qua DataFrame, tương tự như các bảng trong SQL, hỗ trợ truy vấn SQL để thực hiện các phép toán như lọc, nhóm, tính toán và phân tích.

**Ưu điểm:** Spark có khả năng xử lý dữ liệu nhanh chóng nhờ vào việc sử dụng bộ nhớ, giúp giảm thiểu thời gian chờ đợi dữ liệu trên đĩa cứng. Nó cũng hỗ trợ các giao diện lập trình thân thiện như Python, Java, Scala và R, khiến cho việc triển khai trở nên dễ dàng hơn.

### 2. Apache Airflow

**Mục đích:** Apache Airflow là một công cụ lên lịch và tự động hóa các workflow (dòng công việc) phức tạp trong các hệ thống ETL (Extract, Transform, Load). Airflow giúp theo

dõi, quản lý và điều phối các công việc trong một pipeline dữ liệu, đặc biệt là khi các công việc này cần phải thực thi theo một thứ tự hoặc có sự phụ thuộc vào nhau.

### **Chức năng chính:**

- **DAGs (Directed Acyclic Graphs):** Airflow sử dụng cấu trúc DAG để định nghĩa các tác vụ (tasks) và mối quan hệ phụ thuộc giữa các tác vụ đó. Điều này giúp đảm bảo các tác vụ được thực thi theo đúng thứ tự, với các điều kiện như "Task A phải xong thì Task B mới chạy".
- **Quản lý và theo dõi công việc:** Airflow cung cấp giao diện người dùng (UI) cho phép theo dõi tiến độ và trạng thái của các tác vụ trong các DAGs, từ đó dễ dàng phát hiện lỗi và quản lý các pipeline.
- **Tích hợp nhiều công cụ khác:** Airflow hỗ trợ tích hợp với các công cụ khác như Spark, Hadoop, Hive, Kafka, và các dịch vụ đám mây như AWS, Google Cloud để giúp điều phối và xử lý dữ liệu một cách linh hoạt.
- **Lên lịch và tự động hóa:** Với Airflow, người dùng có thể lên lịch các công việc tự động, ví dụ như chạy pipeline hàng ngày vào lúc 3h sáng. Airflow cũng hỗ trợ chạy các công việc định kỳ, ví dụ như thu thập dữ liệu từ các API, chạy các thuật toán phân tích, lưu trữ kết quả.

**Ưu điểm:** Airflow dễ dàng mở rộng và có khả năng tự động hóa các pipeline phức tạp, giúp người dùng dễ dàng điều phối và theo dõi các công việc trong hệ thống xử lý dữ liệu.

### **3. Apache Kafka**

**Mục đích:** Apache Kafka là một hệ thống truyền tải và xử lý luồng dữ liệu phân tán, mạnh mẽ. Kafka giúp truyền tải và lưu trữ các luồng dữ liệu trong thời gian thực, thích hợp cho việc xây dựng các ứng dụng cần xử lý thông tin ngay lập tức (real-time processing).

### **Chức năng chính:**

- **Producer:** Kafka Producer là thành phần tạo và gửi các thông điệp vào Kafka. Dữ liệu được gửi từ các producer sẽ được ghi vào các topic trong Kafka.

- **Consumer:** Kafka Consumer nhận các thông điệp từ Kafka và xử lý chúng. Một consumer có thể đọc và xử lý dữ liệu theo thời gian thực từ nhiều topic khác nhau.
- **Streams API:** Kafka Streams là một thư viện cho phép xử lý luồng dữ liệu ngay lập tức khi dữ liệu được gửi vào. Nó hỗ trợ các tính toán phức tạp, như phân nhóm dữ liệu, windowing (cửa sổ), và join các luồng dữ liệu.
- **Storage:** Kafka cung cấp khả năng lưu trữ dữ liệu dạng topic trong thời gian dài, làm nơi giữ lại các bản ghi dữ liệu trước khi chúng được tiêu thụ bởi các consumer.

**Ứng dụng:** Kafka rất hữu ích trong các hệ thống cần truyền tải dữ liệu phân tán và xử lý dữ liệu theo thời gian thực, chẳng hạn như các hệ thống giám sát, phân tích log, thông báo thời gian thực, hay là một trung tâm kết nối dữ liệu giữa các dịch vụ.

### Tích hợp các công cụ trong dự án Big Data

Trong một dự án Big Data, **Apache Kafka**, **Apache Spark**, và **Apache Airflow** sẽ làm việc cùng nhau để tạo thành một hệ sinh thái dữ liệu mạnh mẽ:

- **Kafka** sẽ thu thập và truyền tải dữ liệu thời gian thực từ các nguồn khác nhau vào hệ thống, ví dụ như từ cảm biến, các hệ thống giám sát, hay từ các API.
- **Spark** sẽ tiếp nhận dữ liệu từ Kafka và thực hiện các phép toán phức tạp trên dữ liệu, bao gồm phân tích dữ liệu, học máy, và các phép toán thống kê.
- **Airflow** sẽ điều phối các công việc trong hệ thống, lên lịch các tác vụ ETL, giám sát quá trình xử lý và đảm bảo các tác vụ được thực hiện theo đúng thứ tự.

## 4. HDFS (*Hadoop Distributed File System*)

**Khái niệm:** HDFS là một hệ thống tệp phân tán được thiết kế để chạy trên phần cứng thương mại (commodity hardware). Đây là hệ thống lưu trữ chính được sử dụng trong Apache Hadoop, một framework mã nguồn mở dành cho việc xử lý và lưu trữ dữ liệu lớn (Big Data).

### Đặc điểm nổi bật:

- **Khả năng chịu lỗi cao (Fault-tolerant):** HDFS được thiết kế để phát hiện lỗi và tự động phục hồi nhanh chóng, đảm bảo tính liên tục và độ tin cậy. Khi một node bị lỗi, dữ liệu vẫn được bảo toàn do cơ chế sao chép dữ liệu.

- **Hiệu suất cao (High Throughput):** HDFS tối ưu cho việc đọc dữ liệu lớn theo luồng (streaming data access), phù hợp với các ứng dụng xử lý hàng loạt (batch processing). Nó đạt được tốc độ truy cập dữ liệu cao bằng cách phân phối dữ liệu trên nhiều node.
- **Quản lý tập dữ liệu lớn:** HDFS được xây dựng để xử lý các tập dữ liệu có kích thước lên tới petabyte và exabyte.
- **Phần cứng thương mại (Commodity Hardware):** HDFS có thể triển khai trên các máy chủ giá rẻ, giảm chi phí đầu tư.
- **Tính sẵn sàng cao (High Availability):** Dữ liệu được lưu trữ phân tán và sao chép (replication) trên nhiều DataNode, giảm thiểu rủi ro mất dữ liệu.
- **Nguyên tắc "ghi một lần, đọc nhiều lần" (Write-once-read-many):** HDFS tối ưu cho các trường hợp dữ liệu được ghi vào một lần và sau đó được đọc nhiều lần.
- **Data Locality:** HDFS cố gắng di chuyển quá trình tính toán đến nơi dữ liệu cư trú (move computation to data), thay vì di chuyển dữ liệu đến nơi tính toán (move data to computation). Điều này giúp giảm thiểu lưu lượng mạng và tăng hiệu suất.

**Kiến trúc chính:** HDFS hoạt động theo kiến trúc Master-Slave với hai thành phần chính:

- **NameNode (Master):**
  - o Là "bộ não" của HDFS, quản lý không gian tên (namespace) của hệ thống tệp (ví dụ: tên tệp, quyền truy cập, cấu trúc thư mục).
  - o Lưu trữ siêu dữ liệu (metadata) của tất cả các tệp và thư mục trong hệ thống.
  - o Quyết định cách các khối dữ liệu (data blocks) được lưu trữ trên các DataNode.
  - o Theo dõi trạng thái của các DataNode và xử lý lỗi của chúng.
  - o NameNode rất quan trọng, nếu nó bị lỗi, toàn bộ hệ thống HDFS có thể không hoạt động. Do đó, thường có các cơ chế dự phòng (Secondary NameNode hoặc High Availability NameNode) để đảm bảo tính sẵn sàng.
- **DataNode (Slave/Worker):**
  - o Là các node thực hiện việc lưu trữ dữ liệu thực tế.
  - o Mỗi DataNode lưu trữ các khối dữ liệu (blocks) của các tệp. Kích thước khối mặc định thường là 128MB hoặc 256MB (có thể cấu hình).

- Chịu trách nhiệm đọc và ghi dữ liệu theo yêu cầu từ NameNode.
- Định kỳ gửi "heartbeat" (tín hiệu sóng) và báo cáo khối (block reports) cho NameNode để NameNode biết trạng thái của chúng và vị trí của các khối dữ liệu.
- Thực hiện việc sao chép khối dữ liệu theo chỉ dẫn của NameNode để đảm bảo tính chịu lỗi.

### **Cách hoạt động cơ bản:**

Khi một ứng dụng muốn *ghi* một tệp vào HDFS:

- Step 1: Client liên hệ với NameNode để yêu cầu ghi tệp.
- Step 2: NameNode trả về một danh sách các DataNode nơi các khối dữ liệu sẽ được lưu trữ.
- Step 3: Client ghi dữ liệu trực tiếp đến các DataNode được chỉ định.
- Step 4: DataNode sao chép các khối dữ liệu sang các DataNode khác theo cấu hình sao chép (mặc định là 3 bản sao).
- Step 5: Sau khi ghi thành công, DataNode xác nhận với NameNode.
- Khi một ứng dụng muốn *đọc* một tệp từ HDFS:
- Step 1: Client liên hệ với NameNode để lấy vị trí của các khối dữ liệu của tệp.
- Step 2: NameNode trả về danh sách các DataNode chứa các khối dữ liệu đó.
- Step 3: Client đọc dữ liệu trực tiếp từ DataNode gần nhất hoặc có tải thấp nhất.

### **5. Delta Lake**

**Khái niệm:** Delta Lake là một lớp lưu trữ mã nguồn mở hoạt động trên đỉnh của Data Lake (ví dụ: trên HDFS, S3, Azure Data Lake Storage). Nó mang lại độ tin cậy và hiệu suất của Data Warehouse vào môi trường Data Lake bằng cách bổ sung các tính năng quan trọng mà Data Lake truyền thống còn thiếu. Delta Lake thường được xây dựng trên định dạng Parquet.

### **Các tính năng nổi bật:**

- **ACID Transactions (Atomicity, Consistency, Isolation, Durability):** Đây là tính năng quan trọng nhất của Delta Lake. Nó đảm bảo các giao dịch trên Data Lake đáng tin cậy, ngay cả khi có nhiều người dùng hoặc ứng dụng truy cập dữ liệu cùng lúc.

- **Atomicity:** Một giao dịch hoặc hoàn thành toàn bộ hoặc không có gì cả.
  - **Consistency:** Dữ liệu luôn ở trạng thái hợp lệ trước và sau giao dịch.
  - **Isolation:** Các giao dịch đồng thời không ảnh hưởng lẫn nhau.
  - **Durability:** Sau khi giao dịch cam kết, các thay đổi sẽ được lưu trữ vĩnh viễn, ngay cả khi hệ thống gặp lỗi.
- **Scalable Metadata Handling:** Delta Lake có thể xử lý hiệu quả siêu dữ liệu cho các bảng lớn với hàng tỷ tệp mà không gặp vấn đề về hiệu suất. Nó sử dụng một transaction log để theo dõi tất cả các thay đổi của bảng.
  - **Unified Batch and Streaming Data Processing:** Delta Lake cho phép sử dụng cùng một bảng để xử lý cả dữ liệu theo lô (batch data) và dữ liệu theo luồng (streaming data), đơn giản hóa kiến trúc dữ liệu.
  - **Schema Enforcement (Enforcement):** Delta Lake tự động kiểm tra lược đồ (schema) để đảm bảo dữ liệu mới được ghi vào bảng phù hợp với lược đồ hiện có, ngăn chặn các lỗi do dữ liệu không đúng định dạng.
  - **Schema Evolution (Tiến hóa):** Delta Lake hỗ trợ việc thay đổi lược đồ bảng theo thời gian (ví dụ: thêm cột mới, thay đổi kiểu dữ liệu) một cách an toàn và có kiểm soát.
  - **Time Travel (Data Versioning):** Delta Lake duy trì lịch sử của tất cả các thay đổi của bảng trong transaction log. Điều này cho phép người dùng truy vấn các phiên bản cũ của dữ liệu (ví dụ: để phân tích dữ liệu tại một thời điểm cụ thể trong quá khứ, hoàn tác các thay đổi sai).
  - **Upserts and Deletes:** Hỗ trợ các thao tác cập nhật (UPDATE), xóa (DELETE) và hợp nhất (MERGE INTO) dữ liệu hiệu quả, điều mà Data Lake truyền thống gặp khó khăn.
  - **Data Skipping (Bỏ qua dữ liệu):** Bằng cách lưu trữ thông kê dữ liệu (min/max values) trong transaction log, Delta Lake có thể bỏ qua các tệp không chứa dữ liệu cần thiết cho một truy vấn, cải thiện hiệu suất.
  - **Z-ordering and Liquid Clustering:** Các kỹ thuật tối ưu hóa vật lý dữ liệu giúp cải thiện hiệu suất truy vấn bằng cách sắp xếp các bản ghi tương tự lại gần nhau trên đĩa.

**Cách hoạt động:** Delta Lake hoạt động bằng cách thêm một transaction log (hay còn gọi là Delta Log) vào thư mục gốc của bảng Delta. Transaction log này ghi lại mọi thay đổi

được thực hiện trên bảng (thêm, xóa, sửa tệp, thay đổi lược đồ, v.v.). Mỗi giao dịch là một bản ghi trong transaction log, và các bản ghi này được lưu trữ dưới dạng các tệp JSON.

Khi thực hiện một thao tác trên bảng Delta (ví dụ: ghi dữ liệu, cập nhật), Delta Lake sẽ:

Step 1: Tạo ra một tập hợp các tệp dữ liệu mới (thường là Parquet) cho các thay đổi.

Step 2: Ghi một bản ghi vào transaction log mô tả những thay đổi này (ví dụ: thêm tệp A, xóa tệp B).

Step 3: Cam kết giao dịch (commit) vào transaction log.

Bằng cách này, Delta Lake đảm bảo rằng các đọc giả luôn thấy một trạng thái nhất quán của bảng, ngay cả khi các ghi đồng thời đang diễn ra.

## **6. Nền tảng triển khai trên Cloud**

### **a. Mục tiêu tổng thể**

Mục tiêu của việc triển khai kiến trúc Lakehouse trên nền tảng Cloud (Azure) kết hợp với Databricks là nhằm xây dựng một hệ thống quản lý, xử lý và phân tích dữ liệu hiện đại, đáp ứng nhu cầu về:

- **Thu thập và lưu trữ dữ liệu lớn từ nhiều nguồn** lên dịch vụ lưu trữ đám mây (Azure Data Lake Storage Gen2).
- **Tổ chức và quản lý dữ liệu theo kiến trúc 3 lớp (Bronze – Silver – Gold)** theo mô hình Lakehouse hiện đại.
- **Tích hợp phân tích và học máy (ML)** thông qua nền tảng Databricks dựa trên Apache Spark.
- **Trực quan hóa kết quả phân tích dữ liệu** phục vụ ra quyết định bằng Power BI.
- Thông qua hệ thống này, doanh nghiệp có thể chuẩn hóa quy trình khai thác dữ liệu, nâng cao chất lượng phân tích và tối ưu hóa các hoạt động dự báo dựa trên mô hình học máy.

### **b. Quy trình triển khai theo kiến trúc 3 lớp (Bronze – Silver – Gold)**

#### **b.1. Kiến trúc tổng quan**

Hệ thống được xây dựng theo kiến trúc phân tầng ba lớp như sau:

- **Bronze Layer:** Lưu trữ dữ liệu thô (raw data) từ nguồn đầu vào. Dữ liệu chưa qua xử lý, thường ở định dạng CSV.

- **Silver Layer:** Chứa dữ liệu đã được xử lý sơ bộ như chuẩn hóa định dạng, ép kiểu dữ liệu, xử lý giá trị thiếu. Dữ liệu được lưu dưới định dạng Parquet.
- **Gold Layer:** Bao gồm dữ liệu đã qua phân tích nâng cao (Feature Engineering), trích xuất đặc trưng thời gian, sẵn sàng phục vụ các bài toán học máy và trực quan hóa.

## b.2. Các bước triển khai chi tiết

### Bước 1: Upload dữ liệu lên Azure Storage

- Tạo Storage Account trên Azure Portal, bật tính năng Hierarchical namespace để hỗ trợ Data Lake Gen2.
- Tạo container tên lakehouse để lưu dữ liệu.
- Sử dụng Azure Storage Explorer để upload dữ liệu CSV (ví dụ: AirQualityHoChiMinhCity.csv) vào thư mục bronze/air\_quality/.

### Bước 2: Kết nối Azure Storage với Databricks

- Tạo một App Registration trong Azure Active Directory để xác thực OAuth.
- Cấp quyền truy cập dữ liệu blob cho ứng dụng.
- Sử dụng thông tin ứng dụng để mount Azure Data Lake vào Databricks bằng đoạn mã câu lệnh trong Spark Notebook.

### Bước 3: Xây dựng pipeline 3 lớp trong Databricks

#### Step 1. Bronze → Silver:

- Đọc dữ liệu CSV từ thư mục Bronze.
- Ép kiểu dữ liệu (timestamp, float), loại bỏ giá trị null.
- Lưu dữ liệu đã xử lý vào thư mục silver/air\_quality/ dưới định dạng Parquet.

#### Step 2. Silver → Gold:

- Trích xuất các đặc trưng thời gian từ cột datetime như: giờ, ngày, thứ, tháng.
- Tạo dataset sẵn sàng phục vụ huấn luyện mô hình.
- Lưu dữ liệu vào thư mục gold/air\_quality/.

### Bước 4: Huấn luyện mô hình dự đoán chất lượng không khí

- Đọc dữ liệu từ Gold Layer.
- Tạo lag features (ví dụ: pm25\_lag1, pm25\_lag2, pm25\_lag3) để tận dụng yếu tố thời gian.

- Chuyển đổi sang Pandas DataFrame và tách dữ liệu train/test.
- Huấn luyện mô hình CatBoostRegressor và đánh giá hiệu quả thông qua các chỉ số RMSE, R<sup>2</sup>.
- Lưu mô hình và kết quả dự đoán vào thư mục platinum/air\_quality\_classified/.

#### **Bước 5: Trực quan hóa với Power BI**

- Chuyển đổi dữ liệu từ định dạng Delta sang Parquet để Power BI có thể đọc được.
- Tải dữ liệu về máy và nạp vào Power BI.
- Tạo biểu đồ đường (Line Chart) để so sánh dữ liệu PM2.5 thực tế và dự đoán theo thời gian.

## **II – Kiến trúc dữ liệu và lưu trữ trong mô hình Lakehouse**

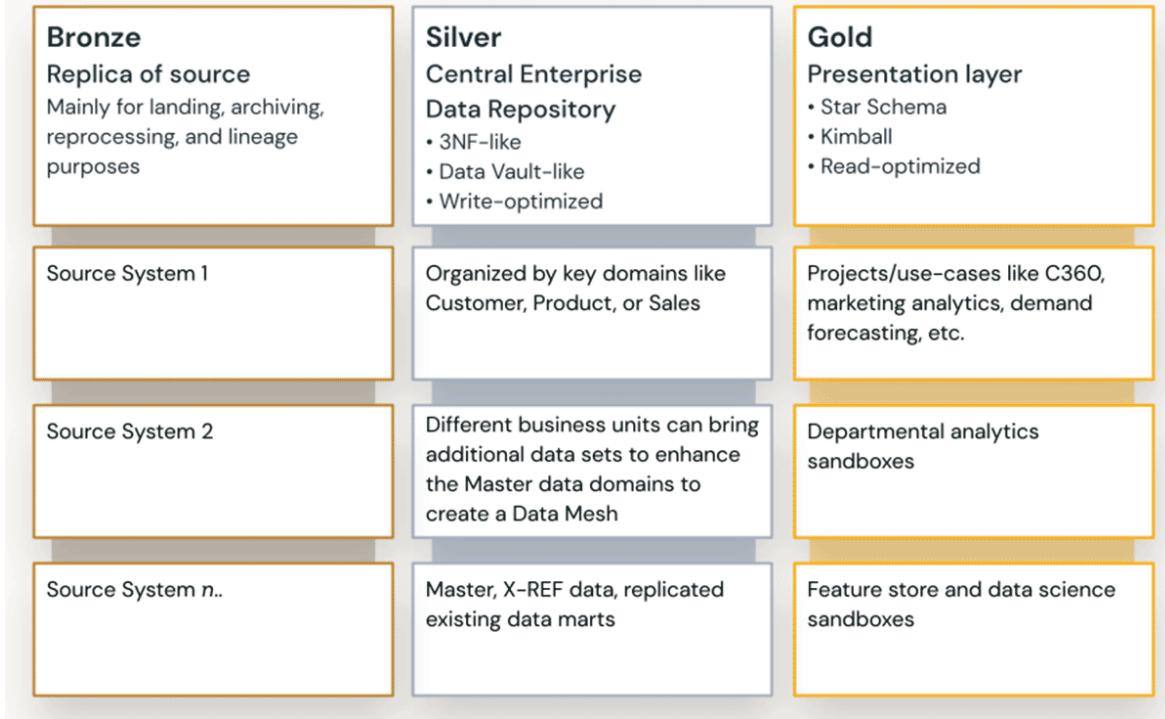
### **1. Lưu trữ 3 lớp Lake house**

Kiến trúc Lakehouse kết hợp những ưu điểm của Data Lake (khả năng lưu trữ dữ liệu thô, đa dạng, chi phí thấp) và Data Warehouse (cấu trúc, hiệu suất truy vấn, quản lý giao dịch). Một trong những cách phổ biến để tổ chức dữ liệu trong Lakehouse là sử dụng kiến trúc Medallion (hay còn gọi là kiến trúc ba lớp: Bronze, Silver, Gold). Kiến trúc này giúp incrementally cải thiện cấu trúc và chất lượng dữ liệu khi nó di chuyển qua từng lớp.

**Mục tiêu:** Đảm bảo tính nguyên tử (atomicity), nhất quán (consistency), cô lập (isolation) và bền vững (durability) khi dữ liệu đi qua nhiều lớp xác thực và chuyển đổi trước khi được lưu trữ trong một bộ cục được tối ưu hóa cho phân tích hiệu quả.

**Các lớp lưu trữ:**

# Data Lakehouse Architecture



## 1. Bronze Layer (Lớp Đồng - Raw Data):

- Mục đích:** Lưu trữ dữ liệu thô, nguyên bản từ các nguồn dữ liệu mà không có bất kỳ sửa đổi hoặc làm sạch nào. Đây là bản sao trung thực của dữ liệu nguồn.
- Định dạng:** Thường giữ nguyên định dạng gốc của dữ liệu nguồn (JSON, CSV, XML, Avro, Parquet, v.v.).
- Đặc điểm:**
  - Dữ liệu được ghi theo kiểu "append-only" (chỉ thêm vào) và có thể giữ lại lịch sử thay đổi đầy đủ.
  - Không có quá trình làm sạch, chuẩn hóa, hoặc xác thực dữ liệu ở lớp này.
  - Chủ yếu được sử dụng bởi các kỹ sư dữ liệu để làm giàu dữ liệu cho các lớp tiếp theo.
  - Cung cấp một điểm khôi phục (recovery point) trong trường hợp có lỗi trong quá trình xử lý hạ nguồn.

- **Ví dụ:** Dữ liệu IoT chưa qua xử lý, logs hệ thống, dữ liệu bán hàng thô từ hệ thống POS.

## 2. Silver Layer (Lớp Bạc - Refined/Conformed Data):

- **Mục đích:** Chứa dữ liệu đã được làm sạch, xác thực, chuyển đổi và chuẩn hóa. Đây là lớp dữ liệu đáng tin cậy hơn, đã được xử lý để loại bỏ lỗi, trùng lặp và các giá trị không hợp lệ.
- **Định dạng:** Thường được lưu trữ ở định dạng cột tối ưu hóa như Parquet hoặc Delta Lake để tăng hiệu suất truy vấn.
- **Đặc điểm:**
  - Dữ liệu từ lớp Bronze được hợp nhất, làm giàu (enrichment) và được tạo cấu trúc để phù hợp với các mô hình dữ liệu.
  - Thực hiện các bước như deduplication (loại bỏ trùng lặp), xử lý giá trị null, chuẩn hóa kiểu dữ liệu, và kết hợp dữ liệu từ nhiều nguồn khác nhau.
  - Phù hợp cho các nhà khoa học dữ liệu (Data Scientists) và kỹ sư dữ liệu để xây dựng các mô hình phân tích và khám phá dữ liệu.
- **Ví dụ:** Dữ liệu khách hàng đã được làm sạch và hợp nhất từ nhiều nguồn, dữ liệu giao dịch đã được xác thực và chuẩn hóa.

## 3. Gold Layer (Lớp Vàng - Curated/Aggregated Data):

- **Mục đích:** Chứa dữ liệu đã được tổng hợp, tổng hợp lại và tối ưu hóa cho các ứng dụng kinh doanh cụ thể, báo cáo và phân tích chuyên sâu. Đây là lớp dữ liệu đã sẵn sàng cho người dùng cuối (business users) và các công cụ BI.
- **Định dạng:** Thường là các bảng tổng hợp, bảng dữ liệu chiều (dimension tables) và bảng dữ kiện (fact tables) được tối ưu hóa cao cho các truy vấn phân tích nhanh chóng, sử dụng các định dạng như Parquet hoặc Delta Lake.
- **Đặc điểm:**
  - Dữ liệu từ lớp Silver được tổng hợp, biến đổi thành các cấu trúc bảng dễ hiểu, phục vụ cho các trường hợp sử dụng cụ thể (ví dụ: báo cáo doanh số hàng ngày, phân tích hiệu suất sản phẩm).

- Thường được tối ưu hóa cho hiệu suất truy vấn bằng cách áp dụng các kỹ thuật như phân vùng (partitioning), nhóm (clustering) và tối ưu hóa file nhỏ.
- Chủ yếu được tiêu thụ bởi các công cụ Business Intelligence (BI), ứng dụng báo cáo, và các ứng dụng phân tích định hướng kinh doanh.
- **Ví dụ:** Bảng tổng hợp doanh số theo tháng, bảng hiệu suất chiến dịch marketing, bảng phân tích khách hàng.

### Lợi ích của kiến trúc lưu trữ 3 lớp trong Lakehouse:

- **Kiểm soát chất lượng dữ liệu:** Đảm bảo dữ liệu được làm sạch và xác thực ở từng giai đoạn, tăng độ tin cậy của dữ liệu.
- **Tính linh hoạt:** Cho phép lưu trữ dữ liệu thô đa dạng ở lớp Bronze, sau đó tinh chỉnh theo nhu cầu cụ thể.
- **Quản lý hiệu suất:** Tối ưu hóa dữ liệu ở lớp Gold cho các truy vấn phân tích, cải thiện hiệu suất.
- **Tái sử dụng:** Các lớp trung gian (Silver) có thể được tái sử dụng cho nhiều trường hợp sử dụng khác nhau.
- **Dễ dàng bảo trì:** Giúp dễ dàng theo dõi và gỡ lỗi các quy trình dữ liệu.

## 2. Giải thích tập dữ liệu

Trong khuôn khổ dự án này, chúng em sử dụng tập dữ liệu về chất lượng không khí và các yếu tố khí tượng được thu thập tại **Thành phố Hồ Chí Minh**. Dữ liệu này đóng vai trò là nguồn đầu vào chính cho pipeline Lakehouse, phục vụ cho các quá trình xử lý, phân tích và trực quan hóa.

### Nguồn gốc và Đặc điểm chung:

- **Địa điểm thu thập:** Dữ liệu được thu thập từ [các trạm quan trắc/cảm biến đặt tại các vị trí khác nhau trên địa bàn Thành phố Hồ Chí Minh.]
- **Khoảng thời gian:** Tập dữ liệu bao gồm các bản ghi từ giữa tháng 2 năm 2021 đến giữa tháng 6 năm 2022.
- **Tần suất ghi nhận:** Dữ liệu được ghi nhận theo từng giờ, cung cấp cái nhìn chi tiết về sự biến động của chất lượng không khí và điều kiện thời tiết trong ngày.

## Cấu trúc và Giải thích các trường dữ liệu:

Tập dữ liệu bao gồm các cột (trường thông tin) sau, với ý nghĩa và đơn vị đo lường cụ thể:

Tên cột	Ý nghĩa	Đơn vị	Kiểu dữ liệu
date	Ngày và giờ cụ thể khi dữ liệu được ghi nhận. Định dạng thường là DD-MM-YYYY HH:MM hoặc tương tự.	-	Datetime
Station_No	Mã số của trạm quan trắc không khí tại Thành phố Hồ Chí Minh nơi dữ liệu được thu thập. Trường này giúp phân biệt dữ liệu từ các địa điểm khác nhau.	-	Categorical/Text/Số
TSP	<b>Total Suspended Particulates (Tổng các hạt rắn lơ lửng):</b> Chỉ nồng độ của tất cả các hạt rắn và lỏng lơ lửng trong không khí, bao gồm bụi, đất, bồ hóng, khói. Đây là một chỉ số tổng quát về ô nhiễm bụi.	$\mu\text{g}/\text{m}^3$	Số thực
PM2.5	<b>Particulate Matter 2.5 (Bụi mịn PM2.5):</b> Chỉ nồng độ của các hạt bụi mịn có đường kính khí động học nhỏ hơn hoặc bằng 2.5 micromet. Đây là loại hạt có khả năng xâm nhập sâu vào hệ hô hấp và gây ảnh hưởng tiêu cực đến sức khỏe.	$\mu\text{g}/\text{m}^3$	Số thực

O3	<b>Ozone (Ozone mặt đất):</b> Chỉ nồng độ khí Ozone ở tầng đối lưu. Ozone mặt đất là một chất ô nhiễm thứ cấp, hình thành từ phản ứng của các oxit nitơ (NOx) và hợp chất hữu cơ dễ bay hơi (VOCs) dưới tác động của ánh sáng mặt trời.	μg/m <sup>3</sup>	Số thực
CO	<b>Carbon Monoxide:</b> Chỉ nồng độ khí Carbon Monoxide, một loại khí độc không màu, không mùi, sinh ra chủ yếu từ quá trình đốt cháy không hoàn toàn nhiên liệu hóa thạch (ví dụ: khí thải xe cộ, hoạt động công nghiệp).	μg/m <sup>3</sup> (hoặc ppm)	Số thực
NO2	<b>Nitrogen Dioxide:</b> Chỉ nồng độ khí Nitrogen Dioxide, một khí có màu nâu đỏ, mùi hăng, chủ yếu từ các quá trình đốt cháy ở nhiệt độ cao (động cơ xe, nhà máy điện) và góp phần hình thành mưa axit, PM2.5.	μg/m <sup>3</sup>	Số thực
SO2	<b>Sulfur Dioxide:</b> Chỉ nồng độ khí Sulfur Dioxide, một khí không màu, mùi hăng, chủ yếu từ việc đốt cháy nhiên liệu hóa thạch chứa lưu huỳnh (than đá, dầu). SO2 là nguyên nhân chính gây mưa axit.	μg/m <sup>3</sup>	Số thực

Temperature	<b>Nhiệt độ:</b> Nhiệt độ không khí tại thời điểm và địa điểm đo.	°C (độ Celsius)	Số thực
Humidity	<b>Độ ẩm:</b> Độ ẩm tương đối của không khí, là tỷ lệ phần trăm giữa lượng hơi nước thực tế trong không khí so với lượng hơi nước tối đa mà không khí có thể chứa ở nhiệt độ đó.	%	Số thực

## Chỉ số AQI:

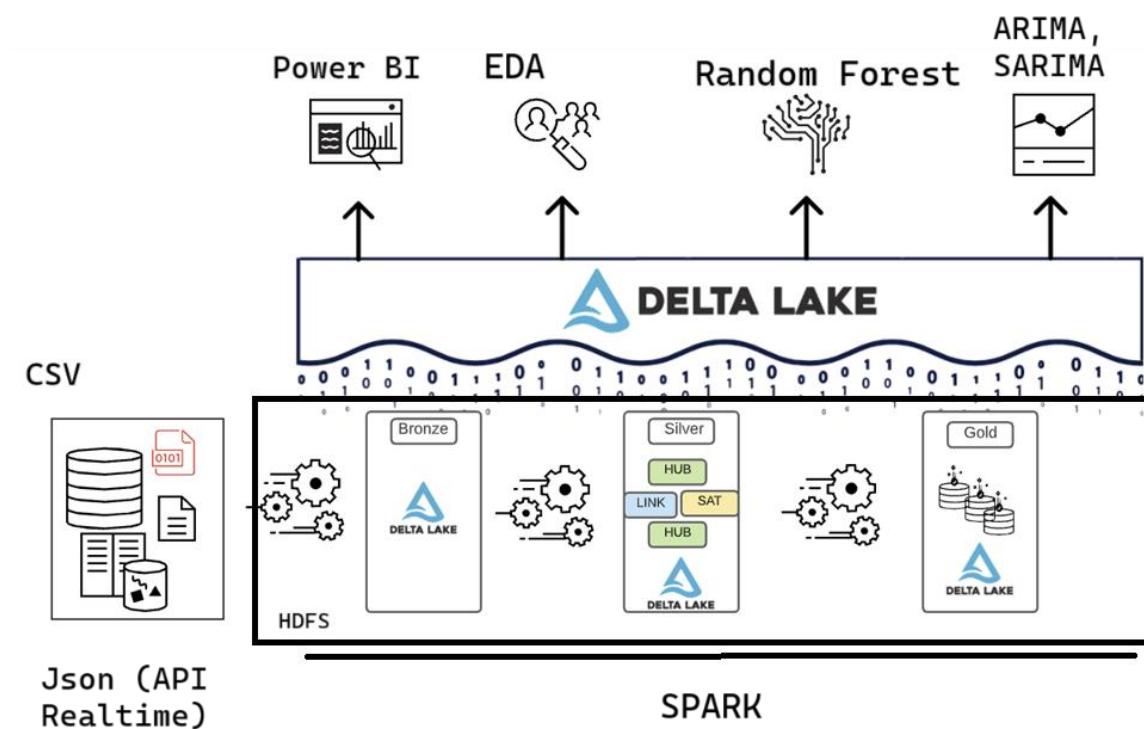
Nhóm đánh giá chỉ số chất lượng không khí trong dự án này theo chuẩn EPA trước đây (do tập data từ năm 2022 trở về trước):

**2024 AQI for Fine Particle Pollution**  
**(Breakpoints are in micrograms per cubic meter)**

AQI Category and Index Value	Previous AQI Category Breakpoints	Updated AQI Category Breakpoints	What changed?
<b>Good (0 – 50)</b>	0.0 to 12.0	0.0 to 9.0	EPA updated the breakpoint between Good and Moderate to reflect the updated annual standard of 9 micrograms per cubic meter
<b>Moderate (51 – 100)</b>	12.1 to 35.4	9.1 to 35.4	
<b>Unhealthy for Sensitive Groups (101 – 150)</b>	35.5 to 55.4	35.5 to 55.4	No change, because EPA retained the 24-hour fine PM standard of 35 micrograms per cubic meter.
<b>Unhealthy (151 – 200)</b>	55.5 to 150.4	55.5 to 125.4	EPA updated the breakpoints at the upper end of the unhealthy, very unhealthy, and hazardous categories based on scientific evidence about particle pollution and health.
<b>Very Unhealthy (201 – 300)</b>	150.5 to 250.4	125.5 to 225.4	The Agency also combined two sets of breakpoints for the Hazardous category into one.
<b>Hazardous (301+)</b>	250.5 to 350.4 and 350.5 to 500	225.5+	

Nguồn: U.S. Environmental Protection Agency (EPA), “Final Reconsideration of the National Ambient Air Quality Standards for Particulate Matter (PM),” [Online]. Available: <https://www.epa.gov/system/files/documents/2024-02/pm-naaqs-air-quality-index-fact-sheet.pdf>

### 3. Tổng quan kiến trúc Lakehouse trên Ubuntu



Mô hình Lakehouse kết hợp các ưu điểm của Data Lake (lưu trữ linh hoạt, chi phí thấp) và Data Warehouse (phân tích dữ liệu chính xác, có cấu trúc).

Trong hệ thống này, dữ liệu được xử lý theo kiến trúc phân tầng 3 lớp: Bronze, Silver, và Gold, thêm một lớp Platinum được xây dựng trên nền tảng lưu trữ HDFS:

```

hadoophongtho@hongtho-master:~$ hdfs dfs -ls /lakehouse
Found 5 items
drwxr-xr-x  - hadoophongtho supergroup      0 2025-05-23 10:55 /lakehouse/bronze
drwxr-xr-x  - hadoophongtho supergroup      0 2025-05-23 11:18 /lakehouse/checkpoints
drwxr-xr-x  - hadoophongtho supergroup      0 2025-05-23 08:23 /lakehouse/gold
drwxr-xr-x  - hadoophongtho supergroup      0 2025-05-23 08:23 /lakehouse/platinum
drwxr-xr-x  - hadoophongtho supergroup      0 2025-05-25 08:12 /lakehouse/silver
hadoophongtho@hongtho-master:~$ |

```

Mô hình Lakehouse tích hợp công cụ quản lý và phân tích dữ liệu:

**1. Bronze Layer – Raw Data:** Lưu trữ dữ liệu thô chưa qua xử lý, từ các nguồn như file CSV hoặc API.

Lưu trữ: Dạng Delta Table trên HDFS.

Công cụ:

- Kafka Producer: thu thập dữ liệu từ API (ví dụ AQICN).
- HDFS: lưu trữ dữ liệu thô.
- Spark Structured Streaming: ghi trực tiếp vào Delta Table ở lớp Bronze.

**2. Silver Layer – Cleaned, Structured Data:** Làm sạch dữ liệu (xử lý missing values, duplicate), thêm cột dữ liệu cần thiết, tiền xử lý.

Xử lý: PySpark thực hiện các bước ETL.

Lưu trữ: Dạng Delta Table trên HDFS

Công cụ:

- Spark: xử lý và chuyển dữ liệu từ Bronze → Silver.

**3. Gold Layer – Business Aggregated Data:** Tính toán, tổng hợp phục vụ phân tích.

Xử lý: Tính toán mức độ AQI theo ngày, phân nhóm theo mức độ ô nhiễm,...

Lưu trữ: Delta Table trên HDFS.

Công cụ:

- Spark: xử lý logic phân tích.
- Power BI: kết nối và trực quan hóa dữ liệu.

**4. Platinum Layer – Serving Layer:** lớp cuối cùng trong kiến trúc Lakehouse, dùng để chuẩn hóa và lưu trữ dữ liệu đã được phân tích, phân loại hoặc dự đoán, phục vụ trực quan hóa hoặc xuất kết quả mô hình.

Lưu trữ:

- Delta Table: chứa dữ liệu đã phân loại, dự đoán (kết quả từ mô hình SARIMA, Random Forest).
- CSV: chứa dữ liệu đã phân loại, dự đoán.
- Model: lưu mô hình ML đã huấn luyện (dạng .pkl, .joblib) .

Công cụ:

- Spark/Pandas: xử lý dữ liệu đầu ra.
- Power BI: sử dụng dữ liệu này để trực quan hóa kết quả mô hình dự đoán.
- Joblib / Pickle: lưu mô hình ML đã huấn luyện vào thư mục model để sử dụng sau này

Mô hình sử dụng:

- Random Forest dự đoán nhãn AIQ (mức độ ô nhiễm là Good, Average, Moderate hay Unhealthy)
- ARIMA/SARIMA cho chuỗi thời gian dự báo giá trị trung bình của chỉ số PM 2.5 trong 3 tháng tiếp theo.

\* **Mở rộng:**

- *Tự động hóa pipeline ETL từ Ingest → Xử lý → Lưu trữ 3 lớp: Apache Airflow.*
- *Xây dựng Lakehouse trên nền tảng Cloud sử dụng: Microsoft Azure, Spark chạy trên Databricks.*

# CHƯƠNG II – THỰC HÀNH

## I – CHUẨN BỊ MÔI TRƯỜNG

### 1. AirFlow

Cài đặt môi trường ảo Airflow để tránh xung đột các thư viện python

```
$ sudo apt update  
$ sudo apt install -y software-properties-common
```

```
root@hieuhaomaster:~# sudo apt install -y software-properties-common

Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  python3-software-properties
The following packages will be upgraded:
  python3-software-properties software-properties-common
2 upgraded, 0 newly installed, 0 to remove and 225 not upgraded.
Need to get 44.3 kB of archives.
After this operation, 1,024 B of additional disk space will be used.
Get:1 http://vn.archive.ubuntu.com/ubuntu noble-updates/main amd64 software-properties-common all 0.99.49.2 [14.4 kB]
Get:2 http://vn.archive.ubuntu.com/ubuntu noble-updates/main amd64 python3-software-properties all 0.99.49.2 [29.8 kB]
Fetched 44.3 kB in 0s (358 kB/s)
(Reading database ... 141447 files and directories currently installed.)
Preparing to unpack .../software-properties-common_0.99.49.2_all.deb ...
Unpacking software-properties-common (0.99.49.2) over (0.99.48) ...
Preparing to unpack .../python3-software-properties_0.99.49.2_all.deb ...
Unpacking python3-software-properties (0.99.49.2) over (0.99.48) ...
Setting up python3-software-properties (0.99.49.2) ...
Setting up software-properties-common (0.99.49.2) ...
Processing triggers for man-db (2.12.0-4build2) ...
Processing triggers for dbus (1.14.10-4ubuntu4) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
root@hieuhaomaster:~#
```

```
$ sudo add-apt-repository ppa:deadsnakes/ppa
```

```
- Note: for Jammy and noble, older python versions require libssl1.3 so they are not currently built
- If you need these, reach out to asottile to set up a private ppa

The packages may also work on other versions of Ubuntu or Debian, but that is not tested or supported.

Packages
=====
The packages provided here are loosely based on the debian upstream packages with some modifications to make them more usable as non-default pythons and on ubuntu. As such, the packages follow debian's patterns and often do not include a full python distribution with just `apt install python#.#`. Here is a list of packages that may be useful along with the default install:
- `python#.#-dev` : includes development headers for building C extensions
- `python#.#-venv` : provides the standard library `venv` module
- `python#.#-distutils` : provides the standard library `distutils` module
- `python#.#-lib2to3` : provides the `2to3-#.#` utility as well as the standard library `lib2to3` module
- `python#.#-gdbm` : provides the standard library `dbm.gnu` module
- `python#.#-tk` : provides the standard library `tkinter` module

Third-Party Python Modules
=====
Python modules in the official Ubuntu repositories are packaged to work with the Python interpreters from the official repositories. Accordingly, they generally won't work with the Python interpreters from this PPA. As an exception, pure-Python modules for Python 3 will work, but any compiled extension modules won't.

To install 3rd-party Python modules, you should use the common Python packaging tools. For an introduction into the Python packaging ecosystem and its tools, refer to the Python Packaging User Guide: https://packaging.python.org/installing/

Sources
=====
The package sources are available at: https://github.com/deadsnakes/

Nightly Builds
=====
For nightly builds, see ppa:deadsnakes/nightly https://launchpad.net/~deadsnakes/+archive/ubuntu/nightly
More info: https://launchpad.net/~deadsnakes/+archive/ubuntu/ppa
Adding repository...
Hit:1 http://vn.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://vn.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://vn.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:5 https://ppa.launchpadcontent.net/deadsnakes/ppa/ubuntu noble InRelease [17.8 kB]
Get:6 https://ppa.launchpadcontent.net/deadsnakes/ppa/ubuntu noble/main amd64 Packages [28.7 kB]
Get:7 https://ppa.launchpadcontent.net/deadsnakes/ppa/ubuntu noble/main Translation-en [5,408 B]
Fetched 51.8 kB in 2s (31.8 kB/s)
Reading package lists... Done
root@hieuhaoo-master:~#
```

```
$ sudo apt update
```

```
Reading package lists... done
root@hieuhaoo-master:~# sudo apt update

Hit:1 http://vn.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://vn.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://vn.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:5 https://ppa.launchpadcontent.net/deadsnakes/ppa/ubuntu noble InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
225 packages can be upgraded. Run 'apt list --upgradable' to see them.
root@hieuhaoo-master:~#
```

```
$ sudo apt install -y python3.10 python3.10-venv python3.10-dev
```

```
Preparing to unpack .../1-python3.10-minimal_3.10.17-1+noble1_amd64.deb ...
Unpacking python3.10-minimal (3.10.17-1+noble1) ...
Selecting previously unselected package libpython3.10-stdlib:amd64.
Preparing to unpack .../2-libpython3.10-stdlib_3.10.17-1+noble1_amd64.deb ...
Unpacking libpython3.10-stdlib:amd64 (3.10.17-1+noble1) ...
Selecting previously unselected package libpython3.10:amd64.
Preparing to unpack .../3-libpython3.10_3.10.17-1+noble1_amd64.deb ...
Unpacking libpython3.10:amd64 (3.10.17-1+noble1) ...
Selecting previously unselected package libpython3.10-dev:amd64.
Preparing to unpack .../4-libpython3.10-dev_3.10.17-1+noble1_amd64.deb ...
Unpacking libpython3.10-dev:amd64 (3.10.17-1+noble1) ...
Selecting previously unselected package python3.10.
Preparing to unpack .../5-python3.10_3.10.17-1+noble1_amd64.deb ...
Unpacking python3.10 (3.10.17-1+noble1) ...
Selecting previously unselected package python3.10-dev.
Preparing to unpack .../6-python3.10-dev_3.10.17-1+noble1_amd64.deb ...
Unpacking python3.10-dev (3.10.17-1+noble1) ...
Selecting previously unselected package python3.10-lib2to3.
Preparing to unpack .../7-python3.10-lib2to3_3.10.17-1+noble1_all.deb ...
Unpacking python3.10-lib2to3 (3.10.17-1+noble1) ...
Selecting previously unselected package python3.10-distutils.
Preparing to unpack .../8-python3.10-distutils_3.10.17-1+noble1_all.deb ...
Unpacking python3.10-distutils (3.10.17-1+noble1) ...
Selecting previously unselected package python3.10-venv.
Preparing to unpack .../9-python3.10-venv_3.10.17-1+noble1_amd64.deb ...
Unpacking python3.10-venv (3.10.17-1+noble1) ...
Setting up python3.10-lib2to3 (3.10.17-1+noble1) ...
Setting up libpython3.10-minimal:amd64 (3.10.17-1+noble1) ...
Setting up python3.10-distutils (3.10.17-1+noble1) ...
Setting up python3.10-minimal (3.10.17-1+noble1) ...
Setting up libpython3.10-stdlib:amd64 (3.10.17-1+noble1) ...
Setting up libpython3.10:amd64 (3.10.17-1+noble1) ...
Setting up python3.10 (3.10.17-1+noble1) ...
Setting up libpython3.10-dev:amd64 (3.10.17-1+noble1) ...
Setting up python3.10-dev (3.10.17-1+noble1) ...
Setting up python3.10-venv (3.10.17-1+noble1) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
root@hieuhao-master:~# _
```

Chuyển qua user Hadoop

Tạo môi trường ảo

```
$ python3.10 -m venv ~/airflow_venv
```

```
hadoophieuhao@hieuhao-master:~$ python3 -m venv ~/airflow_venv
hadoophieuhao@hieuhao-master:~$ 
hadoophieuhao@hieuhao-master:~$
```

## Đặt biến môi trường AIRFLOW\_HOME

```
$ vi ~/.bashrc
export AIRFLOW_HOME=~/airflow
$ source ~/.bashrc
```

```
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
export HADOOP_YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"

export PIG_HOME=/home/hadoopphieuhoa/pig
export PATH=$PATH:$PIG_HOME/bin
export PIG_CLASSPATH=$HADOOP_CONF_DIR

export HIVE_HOME=/home/hadoopphieuhoa/hive
export PATH=$PATH:$HIVE_HOME/bin
export CLASSPATH=$CLASSPATH:/home/hadoopphieuhoa/hadoop/lib/*
export CLASSPATH=$CLASSPATH:/home/hadoopphieuhoa/hive/lib/*

export SPARK_HOME=/home/hadoopphieuhoa/spark
export PATH=$PATH:$SPARK_HOME/bin:$PATH

export PATH="$HOME/scala-2.12.15/bin:$PATH"
export AWS_ACCESS_KEY_ID=minioadmin
export AWS_SECRET_ACCESS_KEY=minioadmin123

export AIRFLOW_HOME=~/airflow
".bashrc" 168L, 4836B written
hadoopphieuhoa@hieuhoa-master:~$ source ~/.bashrc
hadoopphieuhoa@hieuhoa-master:~$ _
```

## Kích hoạt môi trường ảo

```
$ source ~/airflow_venv/bin/activate
hadoopphieuhoa@hieuhoa-master:~$ 
hadoopphieuhoa@hieuhoa-master:~$ source ~/airflow_venv/bin/activate
(airflow_venv) hadoopphieuhoa@hieuhoa-master:~$ 
(airflow_venv) hadoopphieuhoa@hieuhoa-master:~$
```

## Cài đặt pip mới nhất

```
$ pip install --upgrade pip setuptools wheel
```

```
(airflow_venv) hadoopphieuhao@hieuhao-master: ~$ export AIRFLOW_HOME=/airflow
(airflow_venv) hadoopphieuhao@hieuhao-master: ~$ pip install --upgrade pip setuptools wheel
Requirement already satisfied: pip in ./airflow_venv/lib/python3.10/site-packages (23.0.1)
Collecting pip
  Downloading pip-25.1.1-py3-none-any.whl (1.8 MB)
    1.8/1.8 MB 9.8 MB/s eta 0:00:00
Requirement already satisfied: setuptools in ./airflow_venv/lib/python3.10/site-packages (65.5
Collecting setuptools
  Downloading setuptools-80.8.0-py3-none-any.whl (1.2 MB)
    1.2/1.2 MB 36.1 MB/s eta 0:00:00
Collecting wheel
  Downloading wheel-0.45.1-py3-none-any.whl (72 kB)
    72.5/72.5 KB 8.9 MB/s eta 0:00:00
Installing collected packages: wheel, setuptools, pip
  Attempting uninstall: setuptools
    Found existing installation: setuptools 65.5.0
    Uninstalling setuptools-65.5.0:
      Successfully uninstalled setuptools-65.5.0
  Attempting uninstall: pip
    Found existing installation: pip 23.0.1
    Uninstalling pip-23.0.1:
      Successfully uninstalled pip-23.0.1
Successfully installed pip-25.1.1 setuptools-80.8.0 wheel-0.45.1
(airflow_venv) hadoopphieuhao@hieuhao-master: ~$ 
(airflow_venv) hadoopphieuhao@hieuhao-master: ~$
```

### Cài đặt Airflow (ví dụ: 2.8.1)

```
$ pip install apache-airflow==2.8.1 --constraint  
"https://raw.githubusercontent.com/apache/airflow/constraints-2.8.1/constraints-  
3.10.txt"
```

## 2. Khởi tạo cơ sở dữ liệu Airflow

```
$ airflow db init
```

```
(airflow_venv) hadoopiehuhao@hieuhao-master:~$ airflow db init
/home/hadoopiehuhao/airflow_venv/lib/python3.10/site-packages/airflow/cli/commands/db_command.py:47: DeprecationWarning: `db init` is deprecated. Use `db migrate` instead to migrate the db and/or airflow connections create-default-connections to create the default connections
DB: sqlite:///home/hadoopiehuhao/airflow/airflow.db
[2025-05-23T08:31:32,584+0000] {migration_py:216} INFO - Context impl SQLiteImpl.
[2025-05-23T08:31:32,590+0000] {migration_py:219} INFO - Will assume non-transactional DDL.
INFO [alembic.runtime.migration] Context impl SQLiteImpl.
INFO [alembic.runtime.migration] Will assume non-transactional DDL.
INFO [alembic.runtime.migration] Running stamp_revision -> 88344cid9134
WARNING [airflow.models.crypto] empty cryptography key - values will not be stored encrypted.
Initialization done
(airflow_venv) hadoopiehuhao@hieuhao-master:~$ 
(airflow_venv) hadoopiehuhao@hieuhao-master:~$
```

## Tạo người dùng admin để đăng nhập Web UI

```
$ airflow users create --username admin --firstname Admin --lastname User --role Admin --email admin@example.com --password admin
```

```
[2025-05-23T08:33:30.962+0000] {override.py:1820} INFO - Added Permission can read on Passwords to role Admin
[2025-05-23T08:33:30.972+0000] {override.py:1769} INFO - Created Permission View: can edit on My Password
[2025-05-23T08:33:30.978+0000] {override.py:1820} INFO - Added Permission can edit on My Password to role Admin
[2025-05-23T08:33:30.983+0000] {override.py:1769} INFO - Created Permission View: can read on My Password
[2025-05-23T08:33:30.990+0000] {override.py:1820} INFO - Added Permission can read on My Password to role Admin
[2025-05-23T08:33:31.000+0000] {override.py:1769} INFO - Created Permission View: can edit on My Profile
[2025-05-23T08:33:31.004+0000] {override.py:1820} INFO - Added Permission can edit on My Profile to role Admin
[2025-05-23T08:33:31.011+0000] {override.py:1769} INFO - Created Permission View: can read on My Profile
[2025-05-23T08:33:31.018+0000] {override.py:1820} INFO - Added Permission can read on My Profile to role Admin
[2025-05-23T08:33:31.042+0000] {override.py:1769} INFO - Created Permission View: can create on Users
[2025-05-23T08:33:31.049+0000] {override.py:1820} INFO - Added Permission can create on Users to role Admin
[2025-05-23T08:33:31.056+0000] {override.py:1769} INFO - Created Permission View: can read on Users
[2025-05-23T08:33:31.061+0000] {override.py:1820} INFO - Added Permission can read on Users to role Admin
[2025-05-23T08:33:31.069+0000] {override.py:1769} INFO - Created Permission View: can edit on Users
[2025-05-23T08:33:31.074+0000] {override.py:1820} INFO - Added Permission can edit on Users to role Admin
[2025-05-23T08:33:31.084+0000] {override.py:1769} INFO - Created Permission View: can delete on Users
[2025-05-23T08:33:31.092+0000] {override.py:1820} INFO - Added Permission can delete on Users to role Admin
[2025-05-23T08:33:31.103+0000] {override.py:1769} INFO - Created Permission View: menu access on List Users
[2025-05-23T08:33:31.119+0000] {override.py:1820} INFO - Added Permission menu access on List Users to role Admin
[2025-05-23T08:33:31.122+0000] {override.py:1769} INFO - Created Permission View: menu access on Security
[2025-05-23T08:33:31.126+0000] {override.py:1820} INFO - Added Permission menu access on Security to role Admin
[2025-05-23T08:33:31.145+0000] {override.py:1769} INFO - Created Permission View: can create on Roles
[2025-05-23T08:33:31.151+0000] {override.py:1820} INFO - Added Permission can create on Roles to role Admin
[2025-05-23T08:33:31.159+0000] {override.py:1769} INFO - Created Permission View: can read on Roles
[2025-05-23T08:33:31.166+0000] {override.py:1820} INFO - Added Permission can read on Roles to role Admin
[2025-05-23T08:33:31.174+0000] {override.py:1769} INFO - Created Permission View: can edit on Roles
[2025-05-23T08:33:31.182+0000] {override.py:1820} INFO - Added Permission can edit on Roles to role Admin
[2025-05-23T08:33:31.188+0000] {override.py:1769} INFO - Created Permission View: can delete on Roles
[2025-05-23T08:33:31.197+0000] {override.py:1820} INFO - Added Permission can delete on Roles to role Admin
[2025-05-23T08:33:31.207+0000] {override.py:1769} INFO - Created Permission View: menu access on List Roles
[2025-05-23T08:33:31.212+0000] {override.py:1820} INFO - Added Permission menu access on List Roles to role Admin
[2025-05-23T08:33:31.229+0000] {override.py:1769} INFO - Created Permission View: can read on User Stats Chart
[2025-05-23T08:33:31.235+0000] {override.py:1820} INFO - Added Permission can read on User Stats Chart to role Admin
[2025-05-23T08:33:31.246+0000] {override.py:1769} INFO - Created Permission View: menu access on User's Statistics
[2025-05-23T08:33:31.251+0000] {override.py:1820} INFO - Added Permission menu access on User's Statistics to role Admin
[2025-05-23T08:33:31.272+0000] {override.py:1769} INFO - Created Permission View: can read on Permissions
[2025-05-23T08:33:31.279+0000] {override.py:1820} INFO - Added Permission can read on Permissions to role Admin
[2025-05-23T08:33:31.289+0000] {override.py:1769} INFO - Created Permission View: menu access on Actions
[2025-05-23T08:33:31.294+0000] {override.py:1820} INFO - Added Permission menu access on Actions to role Admin
[2025-05-23T08:33:31.315+0000] {override.py:1769} INFO - Created Permission View: can read on View Menus
[2025-05-23T08:33:31.321+0000] {override.py:1820} INFO - Added Permission can read on View Menus to role Admin
[2025-05-23T08:33:31.332+0000] {override.py:1769} INFO - Created Permission View: menu access on Resources
[2025-05-23T08:33:31.337+0000] {override.py:1820} INFO - Added Permission menu access on Resources to role Admin
[2025-05-23T08:33:31.358+0000] {override.py:1769} INFO - Created Permission View: can read on Permission Views
[2025-05-23T08:33:31.365+0000] {override.py:1820} INFO - Added Permission can read on Permission Views to role Admin
[2025-05-23T08:33:31.375+0000] {override.py:1769} INFO - Created Permission View: menu access on Permission Pairs
[2025-05-23T08:33:31.381+0000] {override.py:1820} INFO - Added Permission menu access on Permission Pairs to role Admin
[2025-05-23T08:33:31.381+0000] {override.py:1458} INFO - Added user admin
User "admin" created with role "Admin"
(airflow_venv) hadoopieuhao@hieuhao-master:~$ _
```

## Chạy Webserver

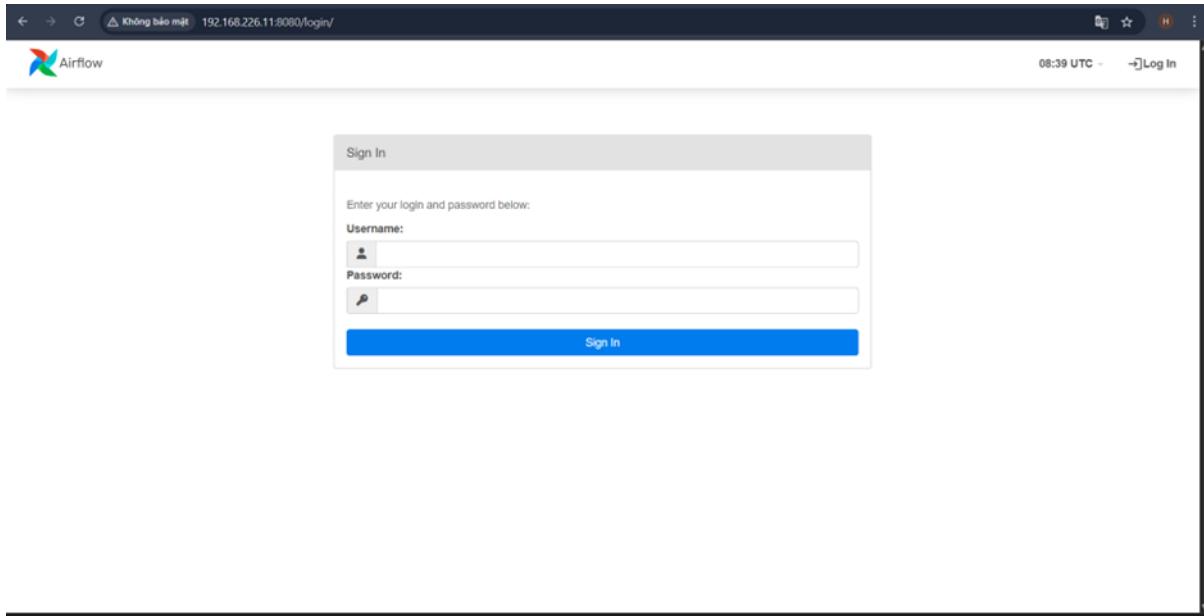
```
$ airflow webserver --port 8080
```

```
Running the Gunicorn Server with:
Workers: 4 sync
Host: 0.0.0.0:8080
Timeout: 120
Logfiles: -
Access Logformat:
=====
/home/hadoopieuhao/airflow_venv/lib/python3.10/site-packages/flask_limiter/extension.py:336 UserWarning: Using the in-memory storage for tracking rate limits is no storage was explicitly specified. This is not recommended for production use. See: https://flask-limiter.readthedocs.io#configuring-a-storage-backend for d
ocumentation about configuring the storage backend.
[2025-05-23T08:38:17.983+0000] {options.py:83} WARNING - The swagger_ui directory could not be found.
Please install connexion with extra install: pip install connexion[swagger-ui]
or provide the path to your local installation by passing swagger_path=<your path>
[2025-05-23T08:38:17.987+0000] {options.py:83} WARNING - The swagger_ui directory could not be found.
Please install connexion with extra install: pip install connexion[swagger-ui]
or provide the path to your local installation by passing swagger_path=<your path>
[2025-05-23 08:38:18 +0000] [3750] [INFO] Starting gunicorn 21.2.0
[2025-05-23 08:38:18 +0000] [3750] [INFO] Listening at: http://0.0.0.0:8080 (3750)
[2025-05-23 08:38:18 +0000] [3750] [INFO] Using worker: sync
[2025-05-23 08:38:18 +0000] [3752] [INFO] Booting worker with pid: 3752
[2025-05-23 08:38:18 +0000] [3753] [INFO] Booting worker with pid: 3753
[2025-05-23 08:38:18 +0000] [3754] [INFO] Booting worker with pid: 3754
[2025-05-23 08:38:18 +0000] [3755] [INFO] Booting worker with pid: 3755
```

## Xem web UI trên máy tính cá nhân

http://IPmayao:8080

ví dụ http://192.168.226.11:8080



đăng nhập username: admin

password: admin

A screenshot of the Airflow DAGs page. The top navigation bar includes links for 'DAGs', 'Cluster Activity', 'Datasets', 'Security', 'Browse', 'Admin', and 'Docs'. The timestamp is '08:40 UTC'. A message at the top states: 'The scheduler does not appear to be running. The DAGs list may not update, and new tasks will not be scheduled.' Below this, two more messages are displayed: 'Do not use \$SQLite as metadata DB in production – it should only be used for dev/testing. We recommend using Postgres or MySQL. Click here for more information.' and 'Do not use the SequentialExecutor in production. Click here for more information.' The main content area shows a table for managing DAGs, with columns for 'All', 'Active', 'Paused', 'Running', 'Failed', 'Owner', 'Schedule', 'Last Run', 'Recent Tasks', 'Actions', and 'Links'. A note says 'No results'. At the bottom, it shows 'Showing 0-0 of 0 DAGs'. The footer contains version information: 'Version: v2.8.1' and 'Git Version: release:c0ffea9c5d96625c88ded9562632674ed366b5eb3'.

Mở terminal mới Ctrl + Alt + F2

Chạy lại môi trường ảo

```
$ source ~/airflow_venv/bin/activate
```

```
Ubuntu 24.04 LTS hieuhao-master tty2

hieuhao-master login: hadoophieuhao
Password:
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-57-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

System information as of Fri May 23 08:36:54 AM UTC 2025

 System load:  0.08      Processes:           229
 Usage of /:   67.3% of 17.83GB  Users logged in:       1
 Memory usage: 28%          IPv4 address for ens33: 192.168.226.11
 Swap usage:   0%

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
 just raised the bar for easy, resilient and secure K8s cluster deployment.

 https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is not enabled.

242 updates can be applied immediately.
46 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

hadoopieuhao@hieuhao-master:~$ source ~/airflow_venv/bin/activate
(airflow_venv) hadoopieuhao@hieuhao-master:~$ _
```

## Lệnh thoát môi trường ảo

```
$ deactivate
```

## Tạo thư mục chứa các DAGS

```
$ mkdir -p /home/hadoopieuhao/airflow/dags
```

```
(airflow_venv) hadoopieuhao@hieuhao-master:~$ mkdir -p /home/hadoopieuhao/airflow/dags
(airflow_venv) hadoopieuhao@hieuhao-master:~$ 
(airflow_venv) hadoopieuhao@hieuhao-master:~$ _
```

## Tạo DAG

```
$ nano /home/hadoopieuhao/airflow/dags/hello_world.py
```

```
from airflow import DAG
from airflow.operators.python import PythonOperator
from datetime import datetime, timedelta

def print_hello():
    print("Hello World")

default_args = {
```

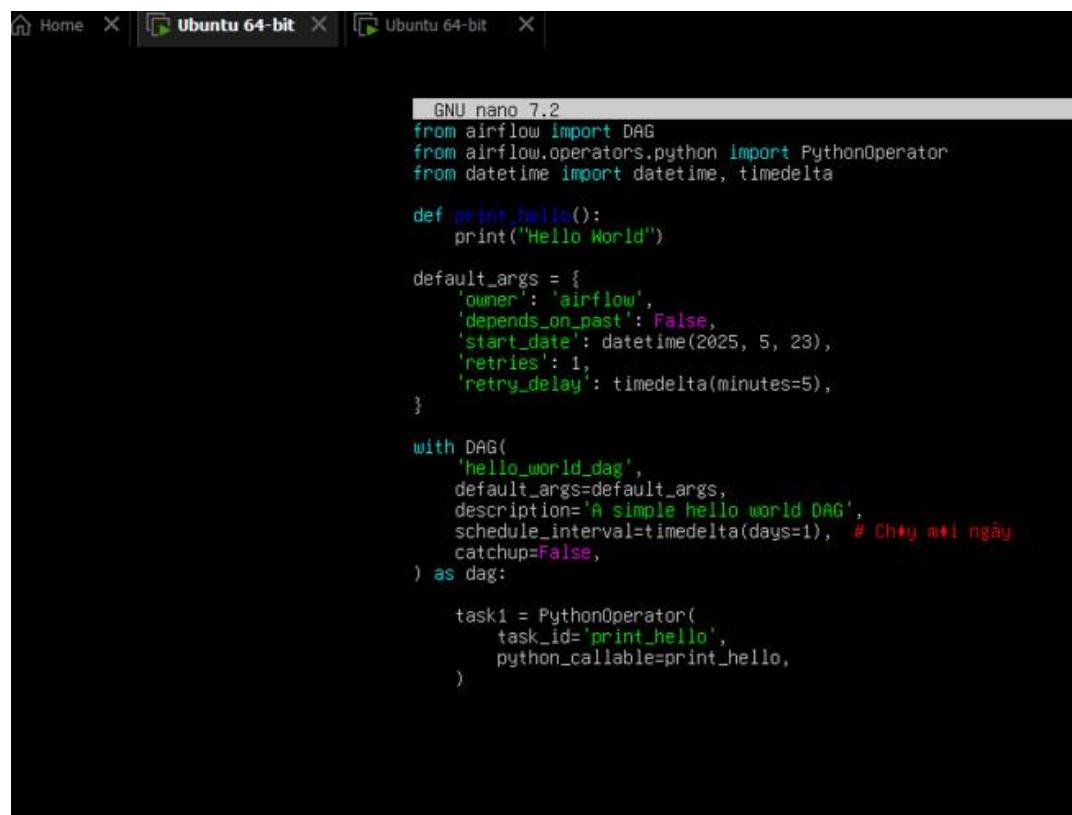
```

'owner': 'airflow',
'depends_on_past': False,
'start_date': datetime(2025, 5, 23),
'retries': 1,
'retry_delay': timedelta(minutes=5),
}

with DAG(
    'hello_world_dag',
    default_args=default_args,
    description='A simple hello world DAG',
    schedule_interval=timedelta(days=1), # Chạy mỗi ngày
    catchup=False,
) as dag:

    task1 = PythonOperator(
        task_id='print_hello',
        python_callable=print_hello,
    )

```



```

GNU nano 7.2
from airflow import DAG
from airflow.operators.python import PythonOperator
from datetime import datetime, timedelta

def print_hello():
    print("Hello World")

default_args = {
    'owner': 'airflow',
    'depends_on_past': False,
    'start_date': datetime(2025, 5, 23),
    'retries': 1,
    'retry_delay': timedelta(minutes=5),
}

with DAG(
    'hello_world_dag',
    default_args=default_args,
    description='A simple hello world DAG',
    schedule_interval=timedelta(days=1), # Chạy mỗi ngày
    catchup=False,
) as dag:

    task1 = PythonOperator(
        task_id='print_hello',
        python_callable=print_hello,
    )

```

Chạy lại môi trường ảo

```
$ source ~/airflow_venv/bin/activate
```

(airflow\_venv) hadoopieuhao@hieuhao-master:~\$

Cài thêm package virtualenv vào môi trường ảo airflow venv:

```
$ pip install virtualenv
```

```
(airflow_venv) hadoophieuhanh@hieuhao-master:~$ pip install virtualenv
Collecting virtualenv
  Downloading virtualenv-20.31.2-py3-none-any.whl.metadata (4.5 kB)
Collecting distlib<1,>=0.3.7 (from virtualenv)
  Downloading distlib-0.3.9-py2.py3-none-any.whl.metadata (5.2 kB)
Collecting filelock<4,>=3.12.2 (from virtualenv)
  Downloading filelock-3.18.0-py3-none-any.whl.metadata (2.9 kB)
Collecting platformdirs<5,>=3.9.1 (from virtualenv)
  Downloading platformdirs-4.3.8-py3-none-any.whl.metadata (12 kB)
  Downloading virtualenv-20.31.2-py3-none-any.whl (6.1 MB)
                                                 6.1/6.1 MB 22.8 MB/s eta 0:00:00
Downloaded distlib-0.3.9-py2.py3-none-any.whl (468 kB)
Downloaded filelock-3.18.0-py3-none-any.whl (16 kB)
Downloaded platformdirs-4.3.8-py3-none-any.whl (18 kB)
Installing collected packages: distlib, platformdirs, filelock, virtualenv
Successfully installed distlib-0.3.9 filelock-3.18.0 platformdirs-4.3.8 virtualenv-20.31.2
(airflow_venv) hadoophieuhanh@hieuhao-master:~$  
(airflow_venv) hadoophieuhanh@hieuhao-master:~$
```

## Tắt các dags mẫu

```
$ export AIRFLOW_CORE_LOAD_EXAMPLES=False
```

```
(airflow_venv) hadoophieuhao@hieuhao-master:~$ export AIRFLOW__CORE__LOAD_EXAMPLES=False  
(airflow_venv) hadoophieuhao@hieuhao-master:~$  
(airflow_venv) hadoophieuhao@hieuhao-master:~$ _
```

## Chạy Scheduler

```
$ airflow scheduler
```

```
(airflow_venv) hadoopheuhao@hieuhaos-MacBook-Pro:~/airflow$ airflow scheduler

[2025-05-23T09:00:20, 318+0000] {task_context_logger.py:63} INFO - Task context logging is enabled
[2025-05-23T09:00:20, 321+0000] {executor_loader.py:115} INFO - Loaded executor: SequentialExecutor
[2025-05-23 09:00:21 +0000] [3940] [INFO] Starting gunicorn 21.2.0
[2025-05-23T09:00:21, 302+0000] {scheduler_job_runner.py:808} INFO - Starting the scheduler
[2025-05-23T09:00:21, 322+0000] {scheduler_job_runner.py:815} INFO - Processing each file at most -1 times
[2025-05-23 09:00:21 +0000] [3940] [INFO] Listening at: http://[::]:8793 (3940)
[2025-05-23 09:00:21 +0000] [3940] [INFO] Using worker: sync
[2025-05-23T09:00:21, 473+0000] {manager.py:169} INFO - Launched DagFileProcessorManager with pid: 3942
[2025-05-23 09:00:21 +0000] [3941] [INFO] Booting worker with pid: 3941
[2025-05-23T09:00:21, 517+0000] {scheduler_job_runner.py:1619} INFO - Adopting or resetting orphaned tasks for active dag runs
[2025-05-23 09:00:21 +0000] [3943] [INFO] Booting worker with pid: 3943
[2025-05-23T09:00:21, 756+0000] {settings.py:68} INFO - Configured default timezone UTC
[2025-05-23T09:00:22, 058+0000] {manager.py:392} WARNING - Because we cannot use more than 1 thread (parsing_processes = 2) when using sqlite. So we set parallelism to 1.
```

## Kiểm tra trên Web UI

The screenshot shows the Airflow web interface at 09:06 UTC. The top navigation bar includes links for DAGs, Cluster Activity, Datasets, Security, Browse, Admin, and Docs. A message at the top left advises against using SQLite as a metadata database. Below the header is a search bar and a 'DAGs' section. The 'hello\_world\_dag' entry is listed, showing it was last run on 2025-05-23 at 00:00:00 and has a green status circle. The bottom of the page displays version information: Version: v2.8.1 and Git Version: release:c0ffa9c5d96625c68ded9562632674ed366b5eb3.

## Enable dags

This screenshot shows the same Airflow interface at 09:11 UTC. The 'hello\_world\_dag' entry is now highlighted with a blue border and a yellow background, indicating it is selected. The 'Paused' button is highlighted in blue, and the 'Active' button is greyed out. The DAG's status circle is yellow, and its last run time is shown as 2025-05-23, 09:11:25. The bottom of the page shows the same version information as the previous screenshot.

Click vào Trigger DAG để chạy thử

The screenshot shows the Airflow web interface at version v2.8.1. The top navigation bar includes links for Airflow, DAGs, Cluster Activity, Datasets, Security, Browse, Admin, and Docs. The time is 09:11 UTC. A message at the top states: "Do not use SQLite as metadata DB in production – it should only be used for dev/testing. We recommend using Postgres or MySQL. Click here for more information." Another message below it says: "Do not use the SequentialExecutor in production. Click here for more information." The main section is titled "DAGs" and displays a single entry: "hello\_world\_dag". The entry details are as follows:

DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks
hello_world_dag	airflow	1	1 day, 0:00:00	2025-05-23, 09:11:25	2025-05-23, 00:00:00	1

Actions and Links buttons are available, along with a "Trigger DAG" button. Below the table, a pagination bar shows page 1 of 1 DAGs.

Click ô số màu viền xanh lá chõ runs để xem

The screenshot shows the Airflow web interface at version v2.8.1. The top navigation bar and time (09:12 UTC) are identical to the previous screenshot. The main section is titled "DAGs" and displays the same "hello\_world\_dag" entry. The "Runs" column now shows a green circle with the word "success" next to it, indicating the run was successful. The rest of the table and interface elements are identical to the first screenshot.

Chạy thành công

The screenshot shows the Airflow interface for listing DAG runs. At the top, there's a search bar and a 'Record Count: 2' indicator. Below is a table with columns: State, Dag id, Logical Date, Run Id, Run Type, Queued At, Start Date, End Date, Note, External Trigger, Conf, and Duration. Two rows are listed:

State	Dag id	Logical Date	Run Id	Run Type	Queued At	Start Date	End Date	Note	External Trigger	Conf	Duration
success	hello_world_dag	2025-05-23, 09:11:25	manual_2025-05-23T09:11:25.821060+00:00	manual	2025-05-23, 09:11:25	2025-05-23, 09:11:26	2025-05-23, 09:11:29	True	0	3s	
success	hello_world_dag	2025-05-23, 09:08:29	manual_2025-05-23T09:08:29.653653+00:00	manual	2025-05-23, 09:08:29	2025-05-23, 09:08:30	2025-05-23, 09:08:32	True	0	1s	

Để xem chi tiết

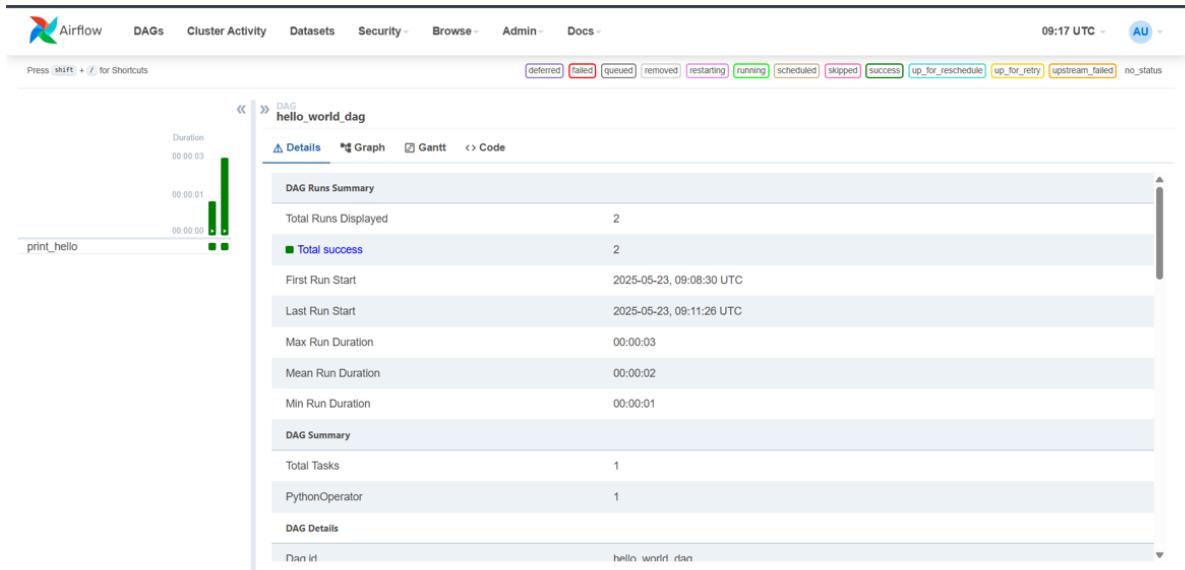
Click vào tên DAG

The screenshot shows the Airflow 'DAGs' page. At the top, there are filters for All, Active, Paused, Running, Failed, and a search bar. The 'hello\_world\_dag' DAG is selected, highlighted with a blue circle. The DAG details below show it's owned by 'airflow', has 1 run, and was last run at 2025-05-23, 09:11:25. The page also includes a note about not using SQLite in production and a footer with version information.

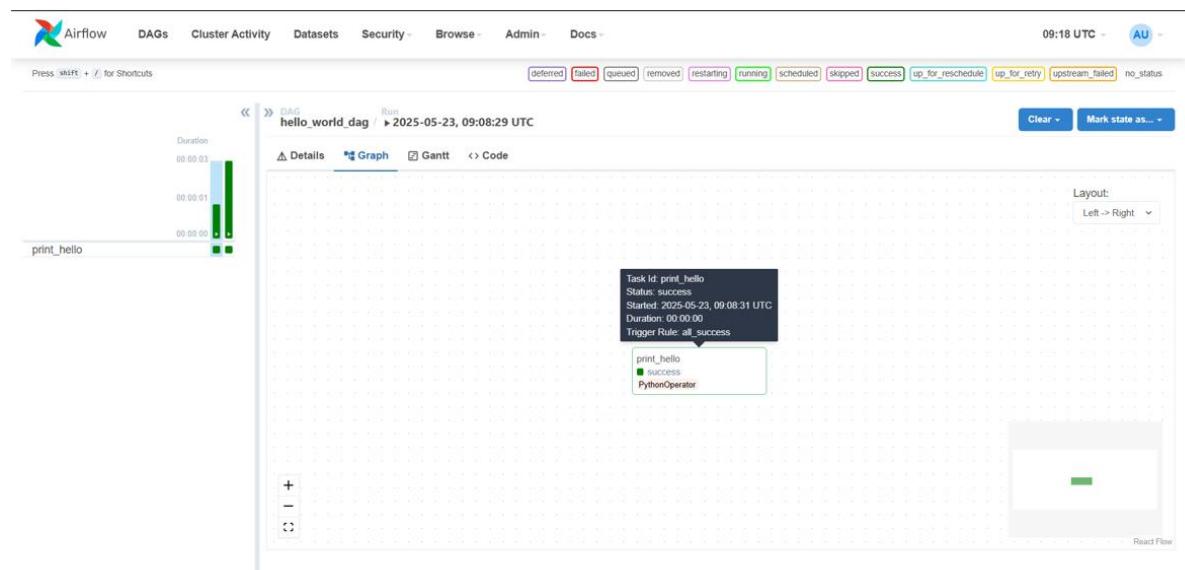
Showing 1-1 of 1 DAGs

Version: v2.8.1  
197.188.226.11:8080/dags/hello\_world\_dag/grid  
c588ded9562632874ed366b5eb3

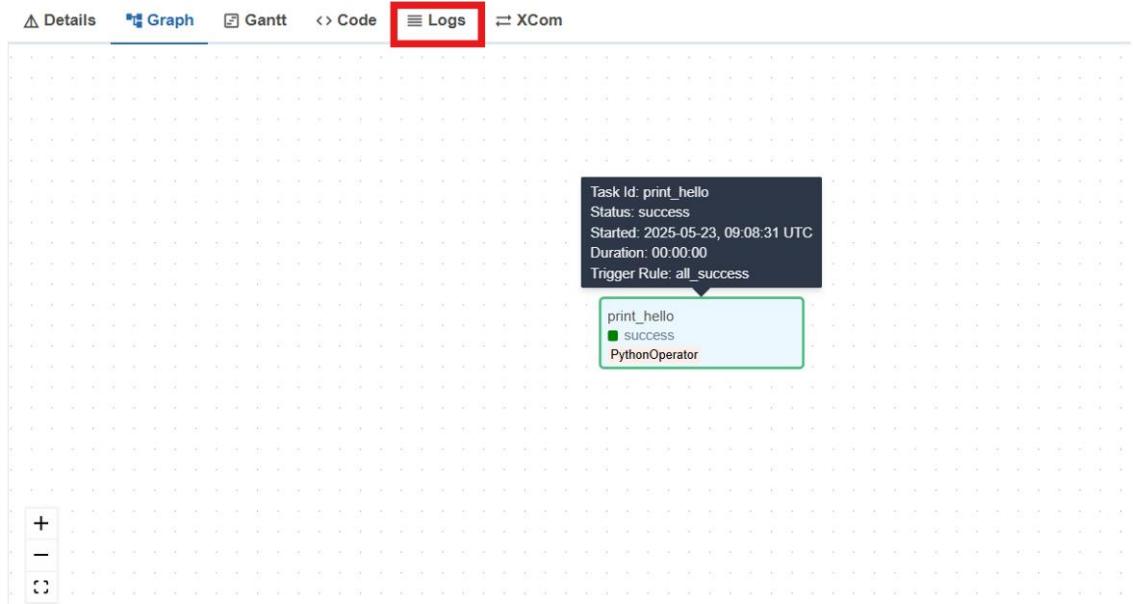
Chọn mục Graph



## Chọn task thực hiện



## Chọn mục Logs cùng hàng với mục Graph



Dòng Hello Word đã được in ra

```

hieuaho-master
*** Found local files:
***   * /home/hadoophieuaho/airflow/logs/dag_id=hello_world_dag/run_id=manual_2025-05-23T09:08:29.653653+00:00/task_id=print_hello/attempt=1.log
[2025-05-23, 09:08:31 UTC] {taskinstance.py:1956} INFO - Dependencies all met for dep_context=non-requireable deps ti=<TaskInstance: hello_world_dag.print_hello manual_2025-05-23T09:08:29.653653+00:00>
[2025-05-23, 09:08:31 UTC] {taskinstance.py:1956} INFO - Dependencies all met for dep_context=requeueable deps ti=<TaskInstance: hello_world_dag.print_hello manual_2025-05-23T09:08:29.653653+00:00>
[2025-05-23, 09:08:31 UTC] {taskinstance.py:2170} INFO - Starting attempt 1 of 2
[2025-05-23, 09:08:31 UTC] {taskinstance.py:2191} INFO - Executing <Task(PythonOperator): print_hello> on 2025-05-23 09:08:29.653653+00:00
[2025-05-23, 09:08:31 UTC] {standard_task_runner.py:87} INFO - Running: ['airflow', 'tasks', 'run', 'hello_world_dag', 'print_hello', 'manual_2025-05-23T09:08:29.653653+00:00', '--job-id', '4', '--raw']
[2025-05-23, 09:08:31 UTC] {standard_task_runner.py:60} INFO - Started process 4268 to run task
[2025-05-23, 09:08:31 UTC] {standard_task_runner.py:88} INFO - Job 4: Subtask print_hello
[2025-05-23, 09:08:31 UTC] {task_command.py:423} INFO - Running <TaskInstance: hello_world_dag.print_hello manual_2025-05-23T09:08:29.653653+00:00> [running] on host hieuaho-master
[2025-05-23, 09:08:31 UTC] {taskinstance.py:2480} INFO - Exporting env vars: AIRFLOW_CTX_DAG_OWNER='airflow' AIRFLOW_CTX_DAG_ID='hello_world_dag' AIRFLOW_CTX_TASK_ID='print_hello' AIRFLOW_CTX_EXECUTOR=''
[2025-05-23, 09:08:31 UTC] {logging_mixin.py:188} INFO - Hello World
[2025-05-23, 09:08:31 UTC] {python.py:201} INFO - Done. Returned value was: None
[2025-05-23, 09:08:31 UTC] {taskinstance.py:1138} INFO - Marking task as SUCCESS. dag_id=hello_world_dag, task_id=print_hello, execution_date=20250523T090829, start_date=20250523T090831, end_date=20250523T090831
[2025-05-23, 09:08:31 UTC] {local_task_job_runner.py:234} INFO - Task exited with return code 0
[2025-05-23, 09:08:31 UTC] {taskinstance.py:3288} INFO - 0 downstream tasks scheduled from follow-on schedule check

```

## 2. Kafka

Cài Kafka:

Tải file trực tiếp trên trang: <https://kafka.apache.org/downloads>

Tải phiên bản 2.8.2. là phiên bản cuối cùng Kafka hỗ trợ Java 8.

 A screenshot of a file manager window. It shows a file named 'kafka\_2.12-2.8.2.tgz' with a size of 70,068 KB, modified on 23/05/2025 at 4:19 CH, and created by WinRAR.

Đưa vào /home/hadoophongtho

```
root@hongtho-master:/home/hadoophongtho# cd
root@hongtho-master:~# cd /mnt/hgfs/MapReduce/
root@hongtho-master:/mnt/hgfs/MapReduce# ls
AirQualityHoChiMinhCity.csv          'hadoop-daemon.sh start datanode.txt'
apache-hive-4.0.1-bin.tar.gz          hadoop-streaming-3.4.0.jar
apache-mahout-distribution-0.13.0.tar.gz hbase-2.5.10-bin.tar.gz
apache-maven-3.9.8                   hive-site.xml
apache-zookeeper-3.9.3-bin.tar.gz    hongtho.txt
books.txt                            htrace-core4-4.0.1-incubating.jar
DimCustomers.txt                     ImportRecommendations.java
employee.csv                         jansi-1.18.jar
greeneggsandham.txt                 kafka_2.12-2.8.2.tgz
hadoop-core-1.2.1.jar               kafka_2.12-3.9.1.tgz
```

```
$ /mnt/hgfs/MapReduce# cp kafka_2.12-2.8.2.tgz /home/hadoophongtho/
$ /mnt/hgfs/MapReduce# chmod 777 /home/hadoophongtho/kafka_2.12-2.8.2.tgz
```

```
root@hongtho-master:/mnt/hgfs/MapReduce# cp kafka_2.12-2.8.2.tgz /home/hadoophongtho/
root@hongtho-master:/mnt/hgfs/MapReduce# chmod /home/hadoophongtho/kafka_2.12-2.8.2.tgz
chmod: missing operand after '/home/hadoophongtho/kafka_2.12-2.8.2.tgz'
Try 'chmod --help' for more information.
root@hongtho-master:/mnt/hgfs/MapReduce# chmod 777 /home/hadoophongtho/kafka_2.12-2.8.2.tgz
```

Giải nén:

```
$ tar -xzf kafka_2.12-2.8.2.tgz
$ mv kafka_2.12-2.8.2 kafka
```

```
hadoopphongtho@hongtho-master:~$ tar -xzf kafka_2.12-2.8.2.tgz
hadoopphongtho@hongtho-master:~$ mv kafka_2.12-2.8.2 kafka
```

```
hadoopphongtho@hongtho-master:~/kafka$ ls
bin config libs LICENSE licenses NOTICE site-docs
```

Bật Zookeeper trước khi test: Kafka cần bật Zookeeper vì Zookeeper đóng vai trò hệ thống quản lý và điều phối cho Kafka. Trong Kafka, mỗi partition có một broker làm leader. Zookeeper giúp xác định và bầu ra broker làm leader khi cần, ví dụ khi một broker bị crash.

```
$ zkServer.sh start  
$ zkServer.sh status  
  
hadoopphongtho@hongtho-master:~/kafka$ zkServer.sh status  
ZooKeeper JMX enabled by default  
Using config: /home/hadoopphongtho/zookeeper/bin/../conf/zoo.cfg  
Client port found: 2181. Client address: localhost. Client SSL: false.  
Mode: standalone  
hadoopphongtho@hongtho-master:~/kafka$
```

Do xung đột về CLASSPATH của Hive và Kafka nên cấu hình Kafka chạy bằng file script cho an toàn:

```
$ vim run_kafka.sh  
$ chmod +x run_kafka.sh
```

```
hadoopphongtho@hongtho-master:~/kafka$ vim run_kafka.sh
```

```
# unset các biến môi trường liên quan Hadoop/Hive để tránh xung đột thư viện  
unset CLASSPATH  
unset HADOOP_HOME  
unset HIVE_HOME  
unset HADOOP_CONF_DIR  
unset HIVE_CONF_DIR  
  
#  
## Chạy lệnh Kafka được truyền vào script  
/home/hadoopphongtho/kafka/bin/kafka-server-start.sh /home/hadoopphongtho/kafka/config/server.properties  
~  
~
```

```
hadoopphongtho@hongtho-master:~/kafka$ chmod +x run_kafka.sh
```

```
$ ./run_kafka.sh -version
```

```
hadoopphongtho@hongtho-master:~/kafka$ ./run_kafka.sh --version  
2.8.2 (Commit:3146c6ff4a24cc24)
```

Khởi động Kafka Server:

```
$ ./run_kafka.sh
```

```

[hadoopphongtho@hongtho-master:~/kafka]
[2025-05-23 10:05:05,974] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from __consumer_offsets-0 in 26 milliseconds for epoch 0, of which 26 milliseconds was spent in the sync loop
[2025-05-23 10:05:05,974] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from __consumer_offsets-3 in 26 milliseconds for epoch 0, of which 26 milliseconds was spent in the sync loop
[2025-05-23 10:05:05,974] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from __consumer_offsets-6 in 26 milliseconds for epoch 0, of which 26 milliseconds was spent in the sync loop
[2025-05-23 10:05:05,974] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from __consumer_offsets-9 in 26 milliseconds for epoch 0, of which 26 milliseconds was spent in the sync loop
[2025-05-23 10:05:05,974] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from __consumer_offsets-12 in 26 milliseconds for epoch 0, of which 26 milliseconds was spent in the sync loop
[2025-05-23 10:05:05,974] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from __consumer_offsets-15 in 26 milliseconds for epoch 0, of which 26 milliseconds was spent in the sync loop
[2025-05-23 10:05:05,974] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from __consumer_offsets-18 in 25 milliseconds for epoch 0, of which 25 milliseconds was spent in the sync loop
[2025-05-23 10:05:05,975] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from __consumer_offsets-21 in 26 milliseconds for epoch 0, of which 25 milliseconds was spent in the sync loop
[2025-05-23 10:05:05,975] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from __consumer_offsets-24 in 26 milliseconds for epoch 0, of which 26 milliseconds was spent in the sync loop
[2025-05-23 10:05:05,976] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from __consumer_offsets-27 in 27 milliseconds for epoch 0, of which 26 milliseconds was spent in the sync loop
[2025-05-23 10:05:05,976] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from __consumer_offsets-30 in 27 milliseconds for epoch 0, of which 27 milliseconds was spent in the sync loop
[2025-05-23 10:05:05,976] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from __consumer_offsets-33 in 26 milliseconds for epoch 0, of which 26 milliseconds was spent in the sync loop
[2025-05-23 10:05:05,977] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from __consumer_offsets-36 in 27 milliseconds for epoch 0, of which 27 milliseconds was spent in the sync loop
[2025-05-23 10:05:05,978] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from __consumer_offsets-39 in 28 milliseconds for epoch 0, of which 28 milliseconds was spent in the sync loop
[2025-05-23 10:05:05,978] INFO [GroupCoordinator 0]: Preparing to rebalance group console-consumer-97554 in state PreparingRebalance with old generation 0 (__consumer_offsets-31) (reason: Adding new member consumer-console-consumer-97554-1-9b1e74fa-0201-4090-8080-05ffcc466cfe with group instance Id None)
[2025-05-23 10:05:06,071] INFO [GroupCoordinator 0]: Stabilized group console-consumer-97554 generation 1 (__consumer_offsets-31) with 1 members (kafka.coordinator.group.GroupCoordinator)
[2025-05-23 10:05:06,102] INFO [GroupCoordinator 0]: Assignment received from leader for group console-consumer-97554 for generation 1. The group has 1 members, 0 of which are static. (kafka.coordinator.group.GroupCoordinator)

```

Chạy thành công, sang Terminal khác kiểm tra các tiến trình đang chạy:

```
$ jps
```

```

hadoopphongtho@hongtho-master:~$ jps
17312 Jps
16769 Kafka
10248 ResourceManager
10072 SecondaryNameNode
10604 DataNode
16284 QuorumPeerMain
9838 NameNode

```

Kiểm tra xem Kafka nhận dữ liệu streaming: Tạo một topic mới

Một topic là một danh mục (hoặc tên luồng dữ liệu) mà các message sẽ được gửi tới. Đối với mỗi topic, Kafka sẽ duy trì một log (nhật ký) phân vùng. Tạo một topic tên là my-topic với một partition và một bản sao (replica):

```
$ cd kafka/
$ vim create_topic.sh
```

```

hadoopphongtho@hongtho-master:~$ cd kafka/
hadoopphongtho@hongtho-master:~/kafka$ vim create_topic.sh
#!/bin/bash
# Create a new topic named 'my-topic' with 1 partition and 1 replica
# Usage: ./create_topic.sh <topic_name>
topic=$1
if [ -z "$topic" ]; then
    echo "Usage: $0 <topic_name>"
    exit 1
fi
echo "Creating topic '$topic' with 1 partition and 1 replica..."
kafka-topics --create --topic $topic --partitions 1 --replicas 1 --bootstrap-server localhost:9092

```

```
hadophongtho@hongtho-m ~ % hadophongtho@hongtho-m ~ % Windows PowerShell ~ % + - X
```

```
#!/bin/bash

unset CLASSPATH
unset HADOOP_HOME
unset HIVE_HOME
unset HADOOP_CONF_DIR
unset HIVE_CONF_DIR

# Script để tạo một Kafka topic

TOPIC_NAME="my-topic"
BOOTSTRAP_SERVER="localhost:9092"
PARTITIONS=1
REPLICATION_FACTOR=1

# Tạo topic
bin/kafka-topics.sh \
--create \
--topic "$TOPIC_NAME" \
--bootstrap-server "$BOOTSTRAP_SERVER" \
--partitions "$PARTITIONS" \
--replication-factor "$REPLICATION_FACTOR"
```

Kiểm tra Kafka bằng cách gửi và nhận message từ topic my-topic:

Tạo file sent\_message:

```
$ vim sent_message.sh
```

```
hadophongtho@hongtho-m ~ % hadophongtho@hongtho-m ~ % + - X
```

```
#!/bin/bash
# run_kafka.sh

# Unset môi trường Hadoop/Hive để tránh xung đột thư viện
unset CLASSPATH
unset HADOOP_HOME
unset HIVE_HOME
unset HADOOP_CONF_DIR
unset HIVE_CONF_DIR

bin/kafka-console-producer.sh --topic my-topic --bootstrap-server localhost:9092
```

```
$ chmod +x sent_message.sh
$ ./sent_message.sh
```

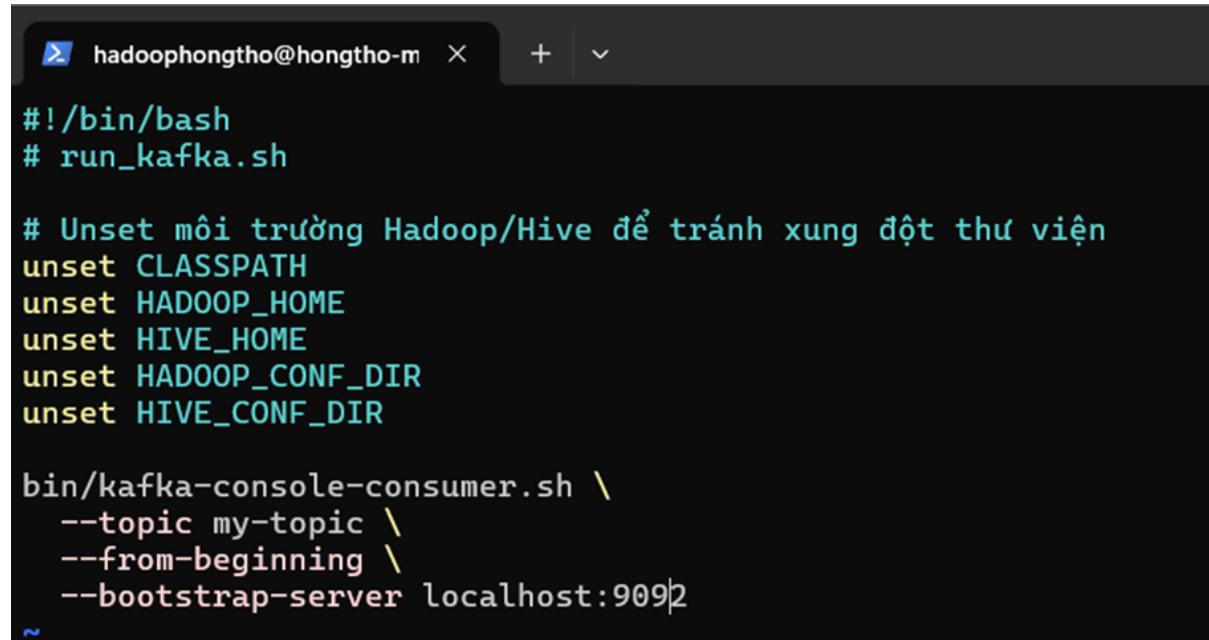
```
hadophongtho@hongtho-master:~/kafka$ chmod +x sent_message.sh
hadophongtho@hongtho-master:~/kafka$ ./sent_message.sh
>hello
>my fkafka
>hello streaming hadophongtho
```

Nhận data:

Mở một terminal khác và tạo file:

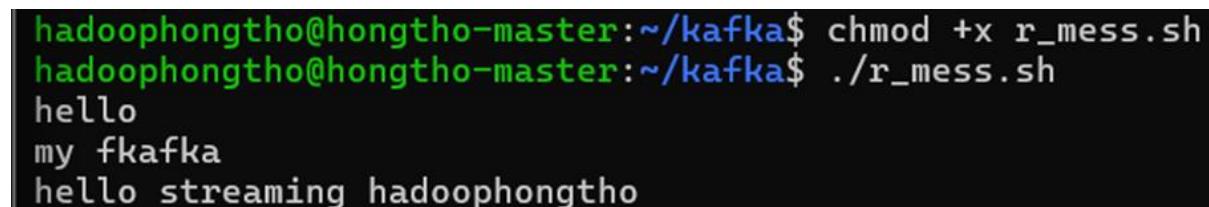
Đọc (consume) các message đã gửi vào topic my-topic:

```
$ cd kafka  
$ vim r_mess.sh
```



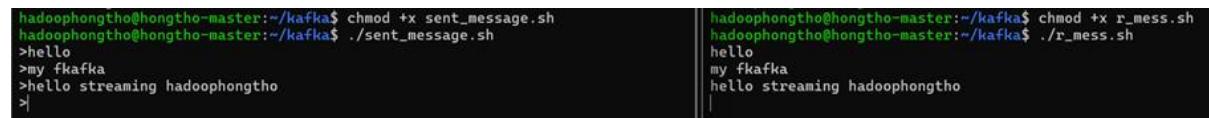
```
#!/bin/bash  
# run_kafka.sh  
  
# Unset môi trường Hadoop/Hive để tránh xung đột thư viện  
unset CLASSPATH  
unset HADOOP_HOME  
unset HIVE_HOME  
unset HADOOP_CONF_DIR  
unset HIVE_CONF_DIR  
  
bin/kafka-console-consumer.sh \  
  --topic my-topic \  
  --from-beginning \  
  --bootstrap-server localhost:9092
```

```
$ chmod +x r_mess.sh  
$ ./r_mess.sh
```



```
hadoopphongtho@hongtho-master:~/kafka$ chmod +x r_mess.sh  
hadoopphongtho@hongtho-master:~/kafka$ ./r_mess.sh  
hello  
my fkafka  
hello streaming hadoopphongtho
```

Kết quả:



```
hadoopphongtho@hongtho-master:~/kafka$ chmod +x sent_message.sh  
hadoopphongtho@hongtho-master:~/kafka$ ./sent_message.sh  
>hello  
>my fkafka  
>hello streaming hadoopphongtho  
|  
hadoopphongtho@hongtho-master:~/kafka$ chmod +x r_mess.sh  
hadoopphongtho@hongtho-master:~/kafka$ ./r_mess.sh  
hello  
my fkafka  
hello streaming hadoopphongtho
```

### 3. Chuẩn bị các thư mục

```
/lakehouse/
  bronze/
    raw/
      air_quality.csv
  delta/
    air_quality/
  streaming/
    air_quality/
  silver/
    air_quality/
  gold/
    air_quality/
  platinum/
    air_quality_classified/
    air_quality_forecast/
  export/
    air_quality_platinum/
```

```
$ hdfs dfs -mkdir -p /lakehouse/bronze/raw
$ hdfs dfs -mkdir -p /lakehouse/bronze/delta/air_quality
$ hdfs dfs -mkdir -p /lakehouse/bronze/streaming/air_quality
$ hdfs dfs -mkdir -p /lakehouse/silver/air_quality
$ hdfs dfs -mkdir -p /lakehouse/gold/air_quality
$ hdfs dfs -mkdir -p /lakehouse/platinum/air_quality_classified
$ hdfs dfs -mkdir -p /lakehouse/platinum/air_quality_forecast
$ hdfs dfs -mkdir -p /lakehouse/export/air_quality_platinum

hadoop@hadoop-master:~$ hdfs dfs -mkdir -p /lakehouse/
hadoop@hadoop-master:~$ hdfs dfs -mkdir -p /lakehouse/bronze/
hadoop@hadoop-master:~$ hdfs dfs -mkdir -p /lakehouse/bronze/raw
hadoop@hadoop-master:~$ hdfs dfs -mkdir -p /lakehouse/bronze/raw/
hadoop@hadoop-master:~$ hdfs dfs -ls /lakehouse/bronze/raw/
hadoop@hadoop-master:~$ hdfs dfs -ls /lakehouse/
Found 1 items
drwxr-xr-x  - hadoop supergroup          0 2025-05-23 08:21 /lakehouse/bronze
hadoop@hadoop-master:~$
```

```
hadoop@hadoop-master:~$ hdfs dfs -mkdir -p /lakehouse/sliver/
hadoop@hadoop-master:~$ hdfs dfs -mkdir -p /lakehouse/gold/
hadoop@hadoop-master:~$ hdfs dfs -mkdir -p /lakehouse/platinum/
```

```
hadoopphongtho@hongtho-master:~$ hdfs dfs -ls /lakehouse/
Found 4 items
drwxr-xr-x  - hadoopphongtho supergroup      0 2025-05-23 08:21 /lakehouse/bronze
drwxr-xr-x  - hadoopphongtho supergroup      0 2025-05-23 08:23 /lakehouse/gold
drwxr-xr-x  - hadoopphongtho supergroup      0 2025-05-23 08:23 /lakehouse/platinum
drwxr-xr-x  - hadoopphongtho supergroup      0 2025-05-23 08:23 /lakehouse/sliver
hadoopphongtho@hongtho-master:~$
```

#### *4. Delta Data*

kiểm tra version scala

```
$ spark-shell
```

```
scala> scala.util.Properties.versionString
```

```
scala> :quit
```

Ta thấy scala có version 2.12.18 và Apache Spark có version 3.5.3

Dùng version Delta Lake tương thích

Phải chạy câu lệnh này mỗi lần muốn dùng Delta Lake

```
$ pyspark --packages io.delta:delta-spark_2.12:3.3.1
```

```
[hadoopgianghi@gianghi-mas ~] [hadoopgianghi@gianghi-mas ~] + 
hadoopgianghi@gianghi-master:~$ pyspark --packages io.delta:delta-spark_2.12:3.3.1
Python 3.12.3 (main, Feb  4 2025, 14:48:35) [GCC 13.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
:: loading settings :: url = jar:file:/home/hadoopgianghi/spark/jars/ivy-2.5.1.jar!/org/apache/ivy/core/settings/ivysettings.xml
Ivy Default Cache set to: /home/hadoopgianghi/.ivy2/cache
The jars for the packages stored in: /home/hadoopgianghi/.ivy2/jars
io.delta#delta-spark_2.12 added as a dependency
:: resolving dependencies :: org.apache.spark#spark-submit-parent-f28ee496-5c17-4611-8602-2d2b49bf7784;1.0
  confs: [default]
    found io.delta#delta-spark_2.12;3.3.1 in central
    found io.delta#delta-storage;3.3.1 in central
    found org.antlr#antlr4-runtime;4.9.3 in local-m2-cache
downloading https://repo.maven.org/maven2/io/delta/delta-spark_2.12/3.3.1/delta-spark_2.12-3.3.1.jar ...
  [SUCCESSFUL ] io.delta#delta-spark_2.12;3.3.1!delta-spark_2.12.jar (8262ms)
downloading https://repo.maven.org/maven2/io/delta/delta-storage/3.3.1/delta-storage-3.3.1.jar ...
  [SUCCESSFUL ] io.delta#delta-storage;3.3.1!delta-storage.jar (378ms)
downloading file:/home/hadoopgianghi/.m2/repository/org/antlr/antlr4-runtime/4.9.3/antlr4-runtime-4.9.3.jar ...
  [SUCCESSFUL ] org.antlr#antlr4-runtime;4.9.3!antlr4-runtime.jar (42ms)
:: resolution report :: resolve 3181ms :: artifacts dl 8693ms
  :: modules in use:
    io.delta#delta-spark_2.12;3.3.1 from central in [default]
    io.delta#delta-storage;3.3.1 from central in [default]
    org.antlr#antlr4-runtime;4.9.3 from local-m2-cache in [default]
-----
|      conf      |           modules           ||   artifacts   | | | | |
|      conf      | number| search|dwnlded|evicted|| number|dwnlded|
|      default    |  3   |  3   |  3   |  0   ||  3   |  3   |
|      default    |      |      |      |      ||      |      |
:: retrieving :: org.apache.spark#spark-submit-parent-f28ee496-5c17-4611-8602-2d2b49bf7784
  confs: [default]
  3 artifacts copied, 0 already retrieved (7313kB/77ms)
25/05/23 19:14:52 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where a
```

## II – XÂY DỰNG VÀ KIỂM THỦ LAKEHOUSE THỦ CÔNG

## **1. Đưa dữ liệu tĩnh vào hệ thống HDFS**

```

$ /usr/bin/vmhgfs-fuse .host:/ /mnt/hgfs -o subtype=vmhgfs-fuse,allow_other
$ cd /mnt/hgfs/MapReduce
$ ls

root@hongtho-master:~# /usr/bin/vmhgfs-fuse .host:/ /mnt/hgfs -o subtype=vmhgfs-fuse,allow_other
root@hongtho-master:~# cd /mnt/hgfs/
root@hongtho-master:/mnt/hgfs# cd MapReduce/

root@hongtho-master:/mnt/hgfs/MapReduce# ls
AirQualityHoChiMinhCity.csv          pg20417.txt
apache-hive-4.0.1-bin.tar.gz         pg4300.txt
apache-mahout-distribution-0.13.0.tar.gz  pg5000.txt
apache-maven-3.9.8                  pom.xml
apache-zookeeper-3.9.3-bin.tar.gz    ProcessUnits.java
books.txt                           ratings.csv
DimCustomers.txt                   retails.csv
employee.csv                      retails-master
greeneggsandham.txt                sample.txt
hadoop-core-1.2.1.jar              'Scaffold-DbContext Data Source=DESK.txt
'hadoop-daemon.sh start datanode.txt'
hadoop-streaming-3.4.0.jar          scala-2.12.18.tgz
hbase-2.5.10-bin.tar.gz            spark-3.5.3-bin-hadoop3.tgz
hive-site.xml                     spark-core_2.12-3.5.3.jar
hongtho.txt                       spark-core_2.13-3.5.3.jar
htrace-core4-4.0.1-incubating.jar   sparkml-master
ImportRecommendations.java          spark-streaming-operation-master
jansi-1.18.jar                    SparkWordCount.scala
logs-analyzer-master               sqoop-1.4.7-bin_hadoop-2.6.0.tar.gz
Main2.java                         src
MaSP.txt                          student_data.txt
page_view.txt                     tstat-sample.txt
part-m-00000                      X-news-master
root@hongtho-master:/mnt/hgfs/MapReduce#

```

 AirQualityHoChiMinhCity.csv	15/05/2025 2:54 SA	Microsoft Excel Co...	4,653 KB
---	--------------------	-----------------------	----------

```

$ cp AirQualityHoChiMinhCity.csv /home/hadoophongtho/
$ chmod 777 /home/hadoophongtho/AirQualityHoChiMinhCity.csv

root@hongtho-master:/mnt/hgfs/MapReduce# cp AirQualityHoChiMinhCity.csv /home/hadoophongtho/
root@hongtho-master:/mnt/hgfs/MapReduce# chmod 777 /home/hadoophongtho/AirQualityHoChiMinhCity.csv
root@hongtho-master:/mnt/hgfs/MapReduce#

```

Đẩy vào HDFS:

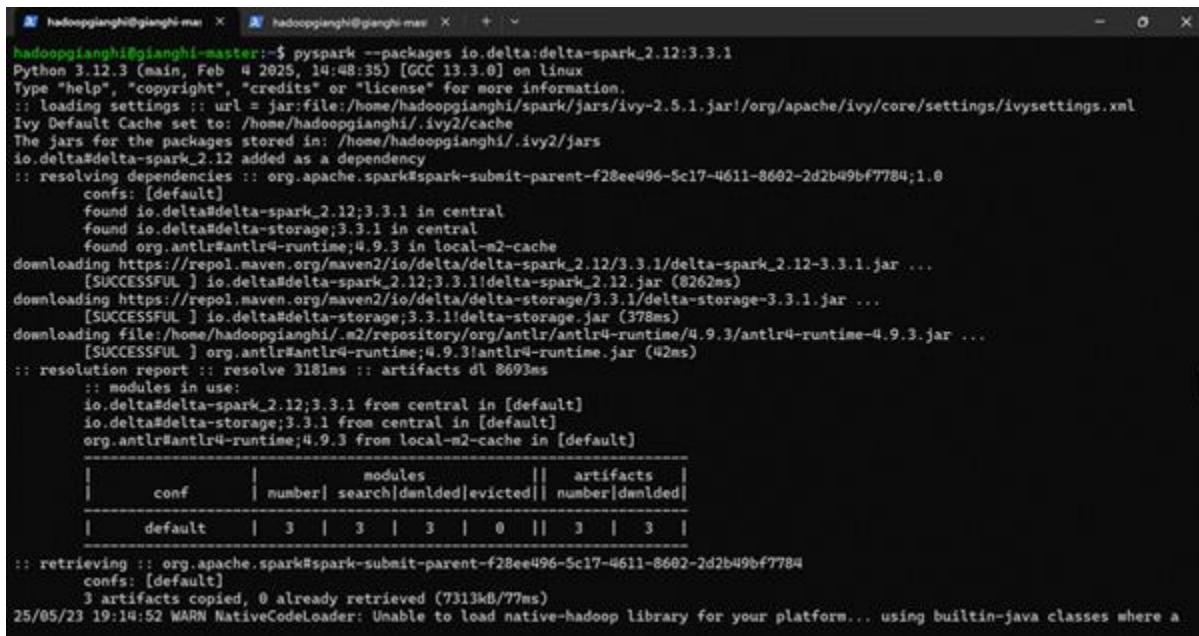
```

$ hdfs dfs -put /home/hadoophongtho/AirQualityHoChiMinhCity.csv
/lakehouse/bronze/raw/
$ hdfs dfs -ls /lakehouse/bronze/raw/
hadoophongtho@hongtho-master:~$ hdfs dfs -put AirQualityHoChiMinhCity.csv /lakehouse/bronze/raw/
hadoophongtho@hongtho-master:~$ hdfs dfs -ls /lakehouse/bronze/raw/
Found 1 items
-rw-r--r-- 2 hadoophongtho supergroup 4763898 2025-05-23 08:32 /lakehouse/bronze/raw/AirQualityHoChiMinhCity.csv
hadoophongtho@hongtho-master:~$ 

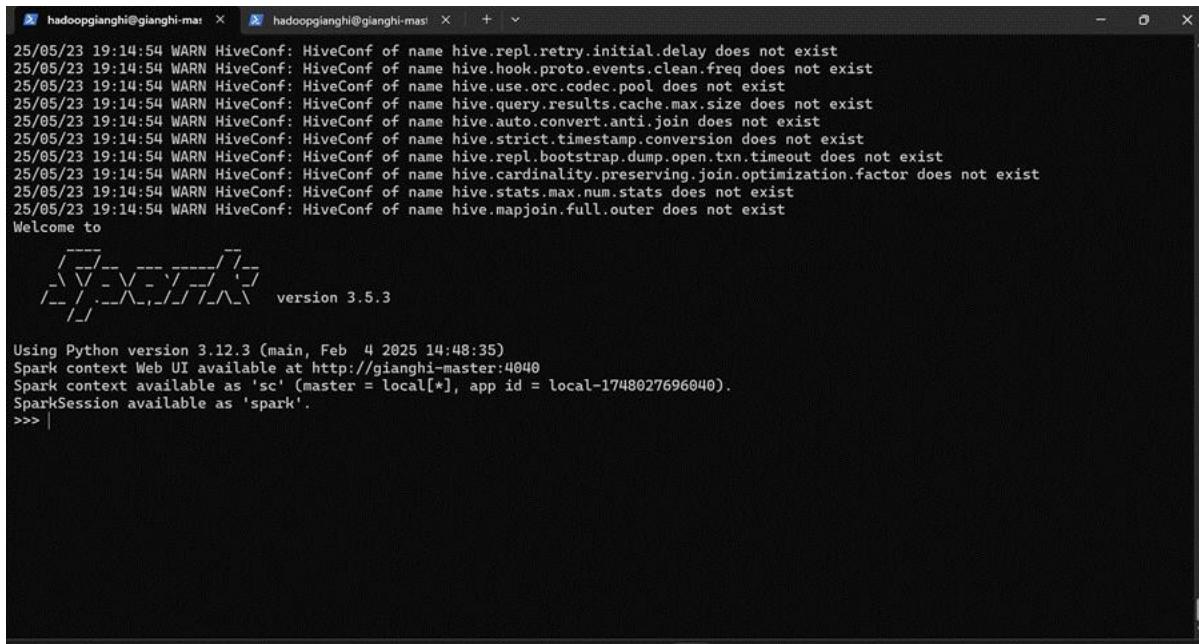
```

## 2. Xử lý và làm sạch dữ liệu qua lớp Bronze

```
$ pyspark --packages io.delta:delta-spark_2.12:3.3.1
```



```
hadoopgianghi@gianghi-ma:~$ pyspark --packages io.delta:delta-spark_2.12:3.3.1
Python 3.12.3 (main, Feb  4 2025, 14:48:35) [GCC 13.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
:: loading settings :: url = jar:file:/home/hadoopgianghi/spark/jars/ivy-2.5.1.jar!/org/apache/ivy/core/settings/ivysettings.xml
Ivy Default Cache set to: /home/hadoopgianghi/.ivy2/cache
The jars for the packages stored in: /home/hadoopgianghi/.ivy2/jars
io.delta#delta-spark_2.12 added as a dependency
:: resolving dependencies :: org.apache.spark#spark-submit-parent-f28ee496-5c17-4611-8602-2d2b49bf7784;1.0
  confs: [default]
    found io.delta#delta-spark_2.12;3.3.1 in central
    found io.delta#delta-storage;3.3.1 in central
    found org.antlr#antlr4-runtime;4.9.3 in local-m2-cache
downloading https://repol.maven.org/maven2/io/delta/delta-spark_2.12/3.3.1/delta-spark_2.12-3.3.1.jar ...
  [SUCCESSFUL ] io.delta#delta-spark_2.12;3.3.1!delta-spark_2.12.jar (826ms)
downloading https://repol.maven.org/maven2/io/delta/delta-storage/3.3.1/delta-storage-3.3.1.jar ...
  [SUCCESSFUL ] io.delta#delta-storage;3.3.1!delta-storage.jar (378ms)
downloading file:/home/hadoopgianghi/.m2/repository/org/antlr/antlr4-runtime/4.9.3/antlr4-runtime-4.9.3.jar ...
  [SUCCESSFUL ] org.antlr#antlr4-runtime;4.9.3!antlr4-runtime.jar (42ms)
:: resolution report :: resolve 3181ms :: artifacts dl 8693ms
  :: modules in use:
    io.delta#delta-spark_2.12;3.3.1 from central in [default]
    io.delta#delta-storage;3.3.1 from central in [default]
    org.antlr#antlr4-runtime;4.9.3 from local-m2-cache in [default]
    +-----+ | modules |-----+ | artifacts +-----+
    | conf | | number| search|downloaded|evicted || number|downloaded|
    | default | | 3 | 3 | 3 | 0 || 3 | 3 |
:: retrieving :: org.apache.spark#spark-submit-parent-f28ee496-5c17-4611-8602-2d2b49bf7784
  confs: [default]
    3 artifacts copied, 0 already retrieved (7313kB/77ms)
25/05/23 19:14:52 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where a
```



```
25/05/23 19:14:54 WARN HiveConf: HiveConf of name hive.repl.retry.initial.delay does not exist
25/05/23 19:14:54 WARN HiveConf: HiveConf of name hive.hook.proto.events.clean.freq does not exist
25/05/23 19:14:54 WARN HiveConf: HiveConf of name hive.use.orc.codec.pool does not exist
25/05/23 19:14:54 WARN HiveConf: HiveConf of name hive.query.results.cache.max.size does not exist
25/05/23 19:14:54 WARN HiveConf: HiveConf of name hive.auto.convert.anti.join does not exist
25/05/23 19:14:54 WARN HiveConf: HiveConf of name hive.strict.timestamp.conversion does not exist
25/05/23 19:14:54 WARN HiveConf: HiveConf of name hive.repl.bootstrap.dump.open.txn.timeout does not exist
25/05/23 19:14:54 WARN HiveConf: HiveConf of name hive.cardinality.preserving.join.optimization.factor does not exist
25/05/23 19:14:54 WARN HiveConf: HiveConf of name hive.stats.max.num.stats does not exist
25/05/23 19:14:54 WARN HiveConf: HiveConf of name hive.mapjoin.full.outer does not exist
Welcome to
   _,-'`--,_---'_-'`-
  / \ - \ - ' /`--' `-
  /-- /--\_,/_/ /_\`-` version 3.5.3
  /`-
Using Python version 3.12.3 (main, Feb  4 2025 14:48:35)
Spark context Web UI available at http://gianghi-master:4040
Spark context available as 'sc' (master = local[*], app id = local-1748027696040).
SparkSession available as 'spark'.
>>>
```

## Tạo Delta Table lớp Bronze bằng PySpark

```
from pyspark.sql import SparkSession
spark = SparkSession.builder \
    .appName("DeltaLakeBronze") \
```

```

.config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
.config("spark.sql.catalog.spark_catalog",
"org.apache.spark.sql.delta.catalog.DeltaCatalog") \
.getOrCreate()

```

```

>>> df = spark.read.option("header",
"true").csv("/lakehouse/bronze/raw/air_quality.csv")
>>> df.printSchema()

```

```

hadoopgiangi@gianghi-ma: ~ hadoopgiangi@gianghi-ma: ~ + v
25/05/23 19:14:54 WARN HiveConf: HiveConf of name hive.mapjoin.full.outer does not exist
Welcome to
   ___
  / \  - \  - \  / \  - \  /
 /--\ / .-. \ / \ / \ \  version 3.5.3
 /_/
Using Python version 3.12.3 (main, Feb 4 2025 14:48:35)
Spark context Web UI available at http://gianghi-master:4040
Spark context available as 'sc' (master = local[*], app id = local-1748027696040).
SparkSession available as 'spark'.
>>> from pyspark.sql import SparkSession
>>> spark = SparkSession.builder \
...     .appName("DeltaLakeBronze") \
...     .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
...     .config("spark.sql.catalog.spark_catalog", "org.apache.spark.sql.delta.catalog.DeltaCatalog") \
...     .getOrCreate()
25/05/23 20:53:57 WARN SparkSession: Using an existing Spark session; only runtime SQL configurations will take effect.
>>> df = spark.read.option("header", "true").csv("/lakehouse/bronze/raw/air_quality.csv")
>>> df.printSchema()
root
 |-- date: string (nullable = true)
 |-- Station_No: string (nullable = true)
 |-- TSP: string (nullable = true)
 |-- PM2.5: string (nullable = true)
 |-- O3: string (nullable = true)
 |-- CO: string (nullable = true)
 |-- NO2: string (nullable = true)
 |-- SO2: string (nullable = true)
 |-- Temperature: string (nullable = true)
 |-- Humidity: string (nullable = true)

```

```

>>> df.show(5)

```

```

hadoopgiangi@gianghi-ma: ~ hadoopgiangi@gianghi-ma: ~ + v
>>> spark = SparkSession.builder \
...     .appName("DeltaLakeBronze") \
...     .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
...     .config("spark.sql.catalog.spark_catalog", "org.apache.spark.sql.delta.catalog.DeltaCatalog") \
...     .getOrCreate()
25/05/23 20:53:57 WARN SparkSession: Using an existing Spark session; only runtime SQL configurations will take effect.
>>> df = spark.read.option("header", "true").csv("/lakehouse/bronze/raw/air_quality.csv")
>>> df.printSchema()
root
 |-- date: string (nullable = true)
 |-- Station_No: string (nullable = true)
 |-- TSP: string (nullable = true)
 |-- PM2.5: string (nullable = true)
 |-- O3: string (nullable = true)
 |-- CO: string (nullable = true)
 |-- NO2: string (nullable = true)
 |-- SO2: string (nullable = true)
 |-- Temperature: string (nullable = true)
 |-- Humidity: string (nullable = true)

>>> df.show(5)
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| date|Station_No| TSP | PM2.5| O3  | CO  | NO2 | SO2 |Temperature| Humidity|
+-----+-----+-----+-----+-----+-----+-----+-----+
| 23-02-2021 21:00| 1|32.93571429| 15.6047619|55.43138095|1330.451429|112.7407619| 393|28.36190476|63.18809524|
| 23-02-2021 22:00| 1|30.93235294|14.5941765|58.19717647|1200.603529|112.3664706|377.5882353|28.32058824|63.77352941|
| 23-02-2021 23:00| 1| 27.645|13.43666667|55.02943333| 1177.897|112.7004333|372.4766667|28.33666667| 64.205|
| 24-02-2021 00:00| 1| 24.38| 12.365| 54.7677| 1267.476|112.4808667| 389.07| 28.395| 64.735|
| 24-02-2021 01:00| 1|22.52166667|11.63666667| 53.7862| 1322.293| 114.3315| 393| 28.3|65.18833333|
+-----+-----+-----+-----+-----+-----+-----+-----+
only showing top 5 rows
>>> |

```

Ghi kết quả vào /lakehouse/bronze/delta/air\_quality

```
>>>
df.write.format("delta").mode("overwrite").save("/lakehouse/bronze/delta/air_quality")
)
>>> df2 = spark.read.format("delta").load("/lakehouse/bronze/delta/air_quality")
>>> df2.show(5)
```

```
-- SO2: string (nullable = true)
-- Temperature: string (nullable = true)
-- Humidity: string (nullable = true)

>>> df.show(5)
+-----+-----+-----+-----+-----+-----+-----+-----+
| date|Station_No|TSP|PM2.5|O3|CO|NO2|SO2|Temperature|Humidity|
+-----+-----+-----+-----+-----+-----+-----+-----+
| 23-02-2021 21:00| 1|32.93571429|15.6047619|55.43138095|1330.451429|112.7407619|393|28.36190476|63.18809524|
| 23-02-2021 22:00| 1|30.93235294|14.59411765|58.19717647|1200.603529|112.3664706|377.5882353|28.32058824|63.77352941|
| 23-02-2021 23:00| 1| 27.645|13.43666667|55.02943333| 1177.897|112.7004333|372.4766667|28.33666667| 64.205|
| 24-02-2021 00:00| 1| 24.38| 12.365| 54.7677| 1267.476|112.4808667| 389.07| 28.305| 64.735|
| 24-02-2021 01:00| 1|22.52166667|11.63666667| 53.7862| 1322.293| 114.3315| 393| 28.3|65.18833333|
+-----+-----+-----+-----+-----+-----+-----+-----+
only showing top 5 rows

>>> df.write.format("delta").mode("overwrite").save("/lakehouse/bronze/delta/air_quality")
>>> df2 = spark.read.format("delta").load("/lakehouse/bronze/delta/air_quality")
>>> df2.show(5)
25/05/23 21:17:27 WARN SparkStringUtils: Truncated the string representation of a plan since it was too large. This behavior can be adjusted by setting 'spark.sql.debug.maxToStringFields'.
+-----+-----+-----+-----+-----+-----+-----+-----+
| date|Station_No|TSP|PM2.5|O3|CO|NO2|SO2|Temperature|Humidity|
+-----+-----+-----+-----+-----+-----+-----+-----+
| 23-02-2021 21:00| 1|32.93571429|15.6047619|55.43138095|1330.451429|112.7407619|393|28.36190476|63.18809524|
| 23-02-2021 22:00| 1|30.93235294|14.59411765|58.19717647|1200.603529|112.3664706|377.5882353|28.32058824|63.77352941|
| 23-02-2021 23:00| 1| 27.645|13.43666667|55.02943333| 1177.897|112.7004333|372.4766667|28.33666667| 64.205|
| 24-02-2021 00:00| 1| 24.38| 12.365| 54.7677| 1267.476|112.4808667| 389.07| 28.305| 64.735|
| 24-02-2021 01:00| 1|22.52166667|11.63666667| 53.7862| 1322.293| 114.3315| 393| 28.3|65.18833333|
+-----+-----+-----+-----+-----+-----+-----+-----+
only showing top 5 rows

>>> |
```

```
>>> df2.printSchema()
>>> df2.count()
```

```
-- SO2: string (nullable = true)
-- Temperature: string (nullable = true)
-- Humidity: string (nullable = true)

>>> df2.show(5)
+-----+-----+-----+-----+-----+-----+-----+-----+
| date|Station_No|TSP|PM2.5|O3|CO|NO2|SO2|Temperature|Humidity|
+-----+-----+-----+-----+-----+-----+-----+-----+
| 23-02-2021 21:00| 1|32.93571429|15.6047619|55.43138095|1330.451429|112.7407619|393|28.36190476|63.18809524|
| 23-02-2021 22:00| 1|30.93235294|14.59411765|58.19717647|1200.603529|112.3664706|377.5882353|28.32058824|63.77352941|
| 23-02-2021 23:00| 1| 27.645|13.43666667|55.02943333| 1177.897|112.7004333|372.4766667|28.33666667| 64.205|
| 24-02-2021 00:00| 1| 24.38| 12.365| 54.7677| 1267.476|112.4808667| 389.07| 28.305| 64.735|
| 24-02-2021 01:00| 1|22.52166667|11.63666667| 53.7862| 1322.293| 114.3315| 393| 28.3|65.18833333|
+-----+-----+-----+-----+-----+-----+-----+-----+
only showing top 5 rows

>>> df2.printSchema()
root
 |-- date: string (nullable = true)
 |-- Station_No: string (nullable = true)
 |-- TSP: string (nullable = true)
 |-- PM2.5: string (nullable = true)
 |-- O3: string (nullable = true)
 |-- CO: string (nullable = true)
 |-- NO2: string (nullable = true)
 |-- SO2: string (nullable = true)
 |-- Temperature: string (nullable = true)
 |-- Humidity: string (nullable = true)

>>> df2.count()
52548
>>> |
```

Mở 1 terminal khác để kiểm tra thư mục trên HDFS có tồn tại hay không

```
$ hdfs dfs -ls /lakehouse/bronze/delta/air_quality
$ hdfs dfs -ls /lakehouse/bronze/delta/air_quality/_delta_log
```

```

hadoopgianghi@gianghi-master:~$ hdfs dfs -ls /lakehouse/bronze/delta/air_quality
Found 3 items
drwxr-xr-x  - hadoopgianghi supergroup          0 2025-05-23 21:09 /lakehouse/bronze/delta/air_quality/_delta_log
-rw-r--r--  2 hadoopgianghi supergroup  1603400 2025-05-23 21:08 /lakehouse/bronze/delta/air_quality/part-00000-8702d0dd-1122-4929
-a3f5-e74bbe86550f-c000.snappy.parquet
-rw-r--r--  2 hadoopgianghi supergroup  159618 2025-05-23 21:08 /lakehouse/bronze/delta/air_quality/part-00001-25cf94fe-7f50-46f6
-8e1f-9f732e59c233-c000.snappy.parquet
hadoopgianghi@gianghi-master:~$ hdfs dfs -ls /lakehouse/bronze/delta/air_quality/_delta_log
Found 3 items
-rw-r--r--  2 hadoopgianghi supergroup      2881 2025-05-23 21:09 /lakehouse/bronze/delta/air_quality/_delta_log/00000000000000000000
00.crc
-rw-r--r--  2 hadoopgianghi supergroup    3090 2025-05-23 21:09 /lakehouse/bronze/delta/air_quality/_delta_log/00000000000000000000
00.json
drwxr-xr-x  - hadoopgianghi supergroup      0 2025-05-23 21:07 /lakehouse/bronze/delta/air_quality/_delta_log/_commits
hadoopgianghi@gianghi-master:~$ 

```

## Tiền xử lý dữ liệu

Thay đổi kiểu dữ liệu

```

>>> from pyspark.sql.functions import col, to_timestamp
>>> from pyspark.sql.types import DoubleType, DateType
silver_df = df2 \
    .withColumn("TSP", col("TSP").cast(DoubleType())) \
    .withColumn(`PM2.5`, col(`PM2.5`).cast(DoubleType())) \
    .withColumn("O3", col("O3").cast(DoubleType())) \
    .withColumn("CO", col("CO").cast(DoubleType())) \
    .withColumn("NO2", col("NO2").cast(DoubleType())) \
    .withColumn("SO2", col("SO2").cast(DoubleType())) \
    .withColumn("Temperature", col("Temperature").cast(DoubleType())) \
    .withColumn("Humidity", col("Humidity").cast(DoubleType())) \
    .withColumn("date", to_timestamp("date", "dd-MM-yyyy HH:mm"))
Dấu ở PM2.5 là backtick, không phải dấu nháy đơn.

```

```

>>> silver_df = df2 \
...     .withColumn("TSP", col("TSP").cast(DoubleType())) \
...     .withColumn(`PM2.5`, col(`PM2.5`).cast(DoubleType())) \
...     .withColumn("O3", col("O3").cast(DoubleType())) \
...     .withColumn("CO", col("CO").cast(DoubleType())) \
...     .withColumn("NO2", col("NO2").cast(DoubleType())) \
...     .withColumn("SO2", col("SO2").cast(DoubleType())) \
...     .withColumn("Temperature", col("Temperature").cast(DoubleType())) \
...     .withColumn("Humidity", col("Humidity").cast(DoubleType())) \
...     .withColumn("date", col("date").cast(DateType()))
>>>

```

Kiểm tra

```

>>> df.printSchema()
>>> df.show(5, truncate=False)

```

```

>>> for col_name in cols_to_cast:
...     df = df.withColumn(col_name.replace("'", ""), col(col_name).cast("double"))
...
>>> df.printSchema()
root
 |-- date: timestamp (nullable = true)
 |-- Station_No: integer (nullable = true)
 |-- TSP: double (nullable = true)
 |-- PM2.5: double (nullable = true)
 |-- O3: double (nullable = true)
 |-- CO: double (nullable = true)
 |-- NO2: double (nullable = true)
 |-- SO2: double (nullable = true)
 |-- Temperature: double (nullable = true)
 |-- Humidity: double (nullable = true)

>>> df.select("`PM2.5`").show(5)
+-----+
| PM2.5|
+-----+
| 15.6047619|
| 14.59411765|
| 13.43666667|
| 12.365|
| 11.63666667|
+-----+
only showing top 5 rows

>>> df.show(5, truncate=False)
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|date |Station_No|TSP |PM2.5 |O3   |CO   |NO2 |SO2  |Temperature|Humidity |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|2021-02-23 21:00:00|1 |32.93571429|15.6047619|55.43138095|1330.451429|112.7407619|393.0 |28.36190476|63.18809524|
|2021-02-23 22:00:00|1 |30.93235294|14.59411765|58.19717647|1200.603529|112.3664706|377.5882353|28.32058824|63.77352941|
|2021-02-23 23:00:00|1 |27.645 |13.43666667|55.02943333|1177.897 |112.7004333|372.4766667|28.33666667|64.205 |
|2021-02-24 00:00:00|1 |24.38  |12.365 |54.7677 |1267.476 |112.4808667|389.07 |28.305  |64.735 |
|2021-02-24 01:00:00|1 |22.52166667|11.63666667|53.7862 |1322.293 |114.3315 |393.0 |28.3  |65.18833333|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
only showing top 5 rows

>>> |

```

Đổi tên cột

```
>>> df = df.withColumnRenamed("PM2.5", "PM25")
```

### Thêm cột AQI\_Level

```

from pyspark.sql.functions import when, col
df = df.withColumn(
    "AQI_Level",
    when(col("PM25") <= 12, "Good")
    .when((col("PM25") >= 13) & (col("PM25") <= 35), "Average")
    .when((col("PM25") >= 36) & (col("PM25") <= 55), "Moderate")
    .otherwise("Unhealthy")
)
df.select("PM25", "AQI_Level").show(10, False)
df.printSchema()
df.columns

```

```

>>> df.printSchema()
root
 |-- date: timestamp (nullable = true)
 |-- Station_No: integer (nullable = true)
 |-- TSP: double (nullable = true)
 |-- PM25: double (nullable = true)
 |-- O3: double (nullable = true)
 |-- CO: double (nullable = true)
 |-- NO2: double (nullable = true)
 |-- SO2: double (nullable = true)
 |-- Temperature: double (nullable = true)
 |-- Humidity: double (nullable = true)
 |-- AQI_Level: string (nullable = false)

>>> from pyspark.sql.functions import col
>>> df.columns
['date', 'Station_No', 'TSP', 'PM25', 'O3', 'CO', 'NO2', 'SO2', 'Temperature', 'Humidity', 'AQI_Level']

df.show(5, truncate=False)

>>> df.show(5, truncate=False)
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|date |Station_No|TSP |PM25 |O3  |CO   |NO2 |SO2  |Temperature|Humidity |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|2021-02-23 21:00:00|1  |32.93571429|15.6947619|55.43138095|1330.451429|112.7407619|393.0 |28.36190476|63.18809524|Average |
|2021-02-23 22:00:00|1  |38.93235294|14.59411765|58.19717647|1200.603529|112.3664706|377.5882353|28.32058824|63.77352941|Average |
|2021-02-23 23:00:00|1  |27.645  |13.43666667|55.02943333|1177.897 |112.7004333|372.4766667|28.33666667|64.205  |Average |
|2021-02-24 00:00:00|1  |24.38   |12.365  |54.7677 |1267.476 |112.4808667|389.07 |28.305  |64.735  |Unhealthy|
|2021-02-24 01:00:00|1  |22.52166667|11.63666667|53.7862 |1322.293 |114.3315 |393.0 |28.3  |65.18833333|Good  |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
only showing top 5 rows
>>>

```

## Kiểm tra số lượng giá trị thiếu (NULL) của từng cột

```

from pyspark.sql.functions import col, sum, when
df.select([
    sum(when(col(c).isNull(), 1).otherwise(0)).alias(c)
    for c in df.columns
]).show()

```

```

>>> from pyspark.sql.functions import col, sum, when
>>> df.select([
...     sum(when(col(c).isNull(), 1).otherwise(0)).alias(c)
...     for c in df.columns
... ]).show()
+-----+-----+-----+-----+-----+-----+-----+-----+
|date |Station_No|TSP |PM25 |O3  |CO   |NO2 |SO2  |Temperature|Humidity |
+-----+-----+-----+-----+-----+-----+-----+-----+
|    0|          0| 60 | 0|10610|9065|5666|11006|        4437|      4432|      0|
+-----+-----+-----+-----+-----+-----+-----+-----+
>>>

```

## Điền giá trị trung bình cho từng cột

```

from pyspark.sql.functions import avg
impute_values = df.select(
    *[avg(c).alias(c) for c in ["TSP", "O3", "CO", "NO2", "SO2", "Temperature",
"Humidity"]])
.first().asDict()
df_clean = df.fillna(impute_values)

```

```

>>> from pyspark.sql.functions import avg
>>> impute_values = df.select(
...     *[avg(c).alias(c) for c in ["TSP", "O3", "CO", "NO2", "SO2", "Temperature", "Humidity"]])
... ).first().asDict()
>>>
>>> df_clean = df.fillna(impute_values)
>>>

```

```

df_clean.select([
    sum(when(col(c).isNull(), 1).otherwise(0)).alias(c)
    for c in df_clean.columns
]).show()

```

```

>>> df_clean.select([
...     sum(when(col(c).isNull(), 1).otherwise(0)).alias(c)
...     for c in df_clean.columns
... ]).show()
+-----+-----+-----+-----+-----+-----+-----+-----+
|date|Station_No|TSP|PM25| O3 | CO |NO2|SO2|Temperature|Humidity|AQI_Level|
+-----+-----+-----+-----+-----+-----+-----+-----+
| 0 |          0 | 0 | 0 | 0 | 0 | 0 | 0 |          0 |          0 | 0 |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

## Thống kê tổng quan (mean, min, max, count)

```
df_clean.describe().show()
```

```

>>> df_clean.describe().show()
+-----+-----+-----+-----+-----+-----+-----+-----+
|summary| Station_No|       TSP|      PM25|       O3|       CO|      NO2|      SO2| Temperature|
+-----+-----+-----+-----+-----+-----+-----+-----+
| count| 52548| 52548| 52548| 52548| 52548| 52548| 52548| 52548| |
| mean| 3.536662266879889| 43.55036919930689| 21.12603131358202| 94.22952165329676| 993.9237262101148| 96.44458603046212| 224.61191021902476| 27.81209825881941|
| stddev| 1.6951224724121795| 35.49360034317673| 14.229703061493222| 33.53222370493598| 560.0734301783439| 67.60073376292567| 101.72726881383875| 4.106419978347255|
| 24.60991641225817| NULL| NULL| NULL| NULL| NULL| NULL| NULL| NULL|
| min| 1| 0.0| 0.0| 0.0| 0.0| 0.0| 0.0| 2.62|
| max| 11.32| Average| 938.1983333| 403.6883333| 377.2886| 21092.57077| 461.09| 699.9766667| 42.8883333|
| 99.28333333|Unhealthy|
+-----+-----+-----+-----+-----+-----+-----+-----+
>>> |

```

## Tính trung bình cho từng cột

```
df_clean.selectExpr("avg(PM25)").show()
```

```

>>> df_clean.selectExpr("avg(PM25)").show()
+-----+
|      avg(PM25) |
+-----+
| 21.12603131358202 |
+-----+

```

```
df_clean.selectExpr("avg(TSP)").show()
```

```
>>> df_clean.selectExpr("avg(TSP)").show()
+-----+
|      avg(TSP) |
+-----+
| 43.55036919930689 |
+-----+
```

```
df_clean.selectExpr("avg(O3)").show()
```

```
>>> df_clean.selectExpr("avg(O3)").show()
+-----+
|      avg(O3) |
+-----+
| 94.22952165329676 |
+-----+
```

```
>>> |
```

```
df_clean.selectExpr("avg(CO)").show()
```

```
>>> df_clean.selectExpr("avg(CO)").show()
+-----+
|      avg(CO) |
+-----+
| 993.9237262101148 |
+-----+
```

```
>>> |
```

```
df_clean.selectExpr("avg(N02)").show()
```

```
>>> df_clean.selectExpr("avg(N02)").show()
+-----+
|      avg(N02) |
+-----+
| 96.44458603046212 |
+-----+
```

```
df_clean.selectExpr("avg(S02)").show()
```

```
>>> df_clean.selectExpr("avg(SO2)").show()
+-----+
|      avg(SO2) |
+-----+
| 224.61191021902476 |
+-----+
```

## Lọc ra các dòng trùng lặp

```
df_clean.groupBy(df_clean.columns) \
    .count() \
    .filter("count > 1") \
    .show(truncate=False)
```

```
>>> df_clean.groupBy(df_clean.columns) \
...     .count() \
...     .filter("count > 1") \
...     .show(truncate=False)
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|date|Station_No|TSP|PM25|O3 |CO  |NO2|SO2|Temperature|Humidity|AQI_Level|count|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
```

```
from pyspark.sql.functions import hour, dayofmonth, month
df_clean = df_clean.withColumn("hour", hour(col("date"))) \
    .withColumn("day", dayofmonth(col("date"))) \
    .withColumn("month", month(col("date")))
df_clean.printSchema()
df_clean.select("date", "hour", "day", "month").show(10, truncate=False)
```

```

>>> df_clean.printSchema()
root
 |-- date: timestamp (nullable = true)
 |-- Station_No: integer (nullable = true)
 |-- TSP: double (nullable = false)
 |-- PM25: double (nullable = true)
 |-- O3: double (nullable = false)
 |-- CO: double (nullable = false)
 |-- NO2: double (nullable = false)
 |-- SO2: double (nullable = false)
 |-- Temperature: double (nullable = false)
 |-- Humidity: double (nullable = false)
 |-- AQI_Level: string (nullable = false)
 |-- hour: integer (nullable = true)
 |-- day: integer (nullable = true)
 |-- month: integer (nullable = true)

>>> df_clean.select("date", "hour", "day", "month").show(10, truncate=False)
+-----+---+---+---+
|date          |hour|day|month|
+-----+---+---+---+
|2021-02-23 21:00:00|21  |23 |2
|2021-02-23 22:00:00|22  |23 |2
|2021-02-23 23:00:00|23  |23 |2
|2021-02-24 00:00:00|0   |24 |2
|2021-02-24 01:00:00|1   |24 |2
|2021-02-24 02:00:00|2   |24 |2
|2021-02-24 03:00:00|3   |24 |2
|2021-02-24 04:00:00|4   |24 |2
|2021-02-24 05:00:00|5   |24 |2
|2021-02-24 06:00:00|6   |24 |2
+-----+---+---+---+
only showing top 10 rows

>>> |

```

## Lưu vài silver

```

df_clean.write.format("delta").mode("overwrite").save("hdfs:///lakehouse/silver/air_q
uality_cleaned/")
df_check =
spark.read.format("delta").load("hdfs:///lakehouse/silver/air_quality_cleaned/")

>>> df_clean.write.format("delta").mode("overwrite").save("hdfs:///lakehouse/silver/air_quality_cleaned/")
>>> df_check = spark.read.format("delta").load("hdfs:///lakehouse/silver/air_quality_cleaned/")
>>> |

```

## Kiểm tra:

```

df_check = spark.read.parquet("hdfs:///lakehouse/silver/air_quality_cleaned/")
df_check.show(5, truncate=False)
df_check.printSchema()
df_check.count()

```

```

>>> df_check = spark.read.parquet("hdfs:///lakehouse/silver/air_quality_cleaned/")
>>> df_clean.write.format("delta").mode("overwrite").save("hdfs:///lakehouse/silver/air_quality_cleaned/")
>>> df_check = spark.read.format("delta").load("hdfs:///lakehouse/silver/air_quality_cleaned/")
>>> df_check = spark.read.parquet("hdfs:///lakehouse/silver/air_quality_cleaned/")
df_check.printSchema()
df_check.count()
>>> df_check.show(5, truncate=False)
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|date |Station_No|TSP |PM25 |O3  |CO   |NO2 |SO2  |Temperature|Humidity |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|2021-02-23 21:00:00|1  |32.93571429|15.6047619|55.43138095|1330.451429|112.7407619|393.0 |28.36190476|63.18809524|Average |
|2021-02-23 22:00:00|1  |38.93235294|14.59411765|58.19717647|1200.683529|112.3664706|377.5882353|28.32058824|63.77352941|Average |
|2021-02-23 23:00:00|1  |27.645  |13.43666667|55.02943333|1177.897 |112.7004333|372.4766667|28.33666667|64.285  |Average |
|2021-02-24 00:00:00|1  |24.38   |12.365  |54.7677 |1267.476 |112.4888667|389.07 |28.305  |64.735  |Unhealthy |
|2021-02-24 01:00:00|1  |22.52166667|11.63666667|53.7862 |1322.293 |114.3315 |393.0 |28.3  |65.18833333|Good  |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
only showing top 5 rows

>>> df_check.printSchema()
root
 |-- date: timestamp (nullable = true)
 |-- Station_No: integer (nullable = true)
 |-- TSP: double (nullable = true)
 |-- PM25: double (nullable = true)
 |-- O3: double (nullable = true)
 |-- CO: double (nullable = true)
 |-- NO2: double (nullable = true)
 |-- SO2: double (nullable = true)
 |-- Temperature: double (nullable = true)
 |-- Humidity: double (nullable = true)
 |-- AQI_Level: string (nullable = true)
 |-- hour: integer (nullable = true)
 |-- day: integer (nullable = true)
 |-- month: integer (nullable = true)

>>> df_check.count()
185096
>>>

```

```

$ hdfs dfs -ls /lakehouse/silver/air_quality_cleaned
$ hdfs dfs -ls /lakehouse/silver/air_quality_cleaned/_delta_log

```

```

hadoopgianghi@gianghi-mast:~$ hdfs dfs -ls /lakehouse/bronze/delta/air_quality
Found 3 items
drwxr-xr-x  - hadoopgianghi supergroup      0 2025-05-23 21:09 /lakehouse/bronze/delta/air_quality/_delta_log
-rw-r--r--  2 hadoopgianghi supergroup 1603400 2025-05-23 21:08 /lakehouse/bronze/delta/air_quality/part-00000-8702d0dd-1122-4929
-a3f5-e74bbe86550f-c000.snappy.parquet
-rw-r--r--  2 hadoopgianghi supergroup 159618 2025-05-23 21:08 /lakehouse/bronze/delta/air_quality/part-00001-25cf94fe-7f50-46f6
-8elf-9f732e59c233-c000.snappy.parquet
hadoopgianghi@gianghi-mast:~$ hdfs dfs -ls /lakehouse/bronze/delta/air_quality/_delta_log
Found 3 items
-rw-r--r--  2 hadoopgianghi supergroup 2881 2025-05-23 21:09 /lakehouse/bronze/delta/air_quality/_delta_log/00000000000000000000
00.crc
-rw-r--r--  2 hadoopgianghi supergroup 3090 2025-05-23 21:09 /lakehouse/bronze/delta/air_quality/_delta_log/00000000000000000000
00.json
drwxr-xr-x  - hadoopgianghi supergroup 0 2025-05-23 21:07 /lakehouse/bronze/delta/air_quality/_delta_log/_commits
hadoopgianghi@gianghi-mast:~$ hdfs dfs -ls /lakehouse/silver/air_quality
Found 3 items
drwxr-xr-x  - hadoopgianghi supergroup 0 2025-05-23 22:57 /lakehouse/silver/air_quality/_delta_log
-rw-r--r--  2 hadoopgianghi supergroup 1608241 2025-05-23 22:57 /lakehouse/silver/air_quality/part-00000-c956fc9c-d3a2-4c42-92c5-
278b440f62aa-c000.snappy.parquet
-rw-r--r--  2 hadoopgianghi supergroup 150195 2025-05-23 22:57 /lakehouse/silver/air_quality/part-00001-b5f6003d-70cb-4f9a-8e6e-
335b27b98407-c000.snappy.parquet
hadoopgianghi@gianghi-mast:~$ hdfs dfs -ls /lakehouse/silver/air_quality/_delta_log
Found 3 items
-rw-r--r--  2 hadoopgianghi supergroup 3046 2025-05-23 22:57 /lakehouse/silver/air_quality/_delta_log/00000000000000000000.crc
-rw-r--r--  2 hadoopgianghi supergroup 3255 2025-05-23 22:57 /lakehouse/silver/air_quality/_delta_log/00000000000000000000.json
drwxr-xr-x  - hadoopgianghi supergroup 0 2025-05-23 22:57 /lakehouse/silver/air_quality/_delta_log/_commits
hadoopgianghi@gianghi-mast:~|

```

### 3. Xử lý dữ liệu qua lớp Silver

#### a. Giá trị trung bình theo thời gian

```

# daily_avg.py
from pyspark.sql import SparkSession
from pyspark.sql.functions import avg, to_date, when, col

spark = SparkSession.builder \
    .appName("Daily AQI Average") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \

```

```

    .config("spark.sql.catalog.spark_catalog",
"org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .getOrCreate()

df_silver =
spark.read.format("delta").load("hdfs:///lakehouse/silver/air_quality_cleaned/")

df_gold = df_silver.withColumn("date_only", to_date("date")) \
    .groupBy("date_only") \
    .agg(
        avg("PM25").alias("avg_PM25"),
        avg("Temperature").alias("avg_temp"),
        avg("Humidity").alias("avg_humidity")
    ) \
    .withColumn(
        "AQI_Level",
        when(col("avg_PM25") <= 12, "Good")
            .when((col("avg_PM25") > 12) & (col("avg_PM25") <= 35), "Average")
            .when((col("avg_PM25") > 35) & (col("avg_PM25") <= 55), "Moderate")
            .otherwise("Unhealthy")
    )
)

df_gold.write.format("delta") \
    .mode("overwrite") \
    .save("hdfs:///lakehouse/gold/air_quality_daily_avg")

```

```

>>> df_silver = spark.read.format("delta").load("hdfs:///lakehouse/silver/air_quality_cleaned/")
>>> df_gold = df_silver.withColumn("date_only", to_date("date")) \
...     .groupBy("date_only") \
...     .agg(
...         avg("PM25").alias("avg_PM25"),
...         avg("Temperature").alias("avg_temp"),
...         avg("Humidity").alias("avg_humidity")
...     ) \
...     .withColumn(
...         "AQI_Level",
...         when(col("avg_PM25") <= 12, "Good")
...             .when((col("avg_PM25") > 12) & (col("avg_PM25") <= 35), "Average")
...             .when((col("avg_PM25") > 35) & (col("avg_PM25") <= 55), "Moderate")
...             .otherwise("Unh...           avg("Humidity").alias("avg_humidity")
...     ) \
...     .withColumn(
...         "AQI_Level",
...         when(col("avg_PM25") <= 12, "Good")
...             .when((col("avg_PM25") > 12) & (col("avg_PM25") <= 35), "Average")
...             .when((col("avg_PM25") > 35) & (col("avg_PM25") <= 55), "Moderate")
...             .otherwise("Unhealthy")
...     )
>>>
>>>
>>>
>>>
>>> df_gold.write.format("delta") \
...     .mode("overwrite") \
...     .save("hdfs:///lakehouse/gold/air_quality_daily_avg")
>>> |

```

```

# hourly_avg.py
from pyspark.sql import SparkSession
from pyspark.sql.functions import avg, hour, month
df_hourly_avg = df_silver.withColumn("hour", hour("date")) \
    .groupBy("hour") \

```

```
.agg(
    avg("PM25").alias("avg_PM25"),
    avg("CO").alias("avg_CO"),
    avg("Temperature").alias("avg_temp")
) \
.orderBy("hour")

df_hourly_avg.write.format("delta") \
.mode("overwrite") \
.save("hdfs:///lakehouse/gold/air_quality_hourly_avg")
```

```
>>> df_silver = spark.read.format("delta").load("hdfs:///lakehouse/silver/air_quality_cleaned/")
>>> df_hourly_avg = df_silver.withColumn("hour", hour("date")) \
...     .groupBy("hour") \
...     .agg(
...         avg("PM25").alias("avg_PM25"),
...         avg("CO").alias("avg_CO"),
...         avg("Temperature").alias("avg_temp")
...     ) \
...     .orderBy("hour")
\
    .save("hdfs:///lakehouse/gold/air_quality_hourly_avg")
>>>
>>> df_hourly_avg.write.format("delta") \
...     .mode("overwrite") \
...     .save("hdfs:///lakehouse/gold/air_quality_hourly_avg")
>>>
```

```
df_monthly_avg = df_silver.withColumn("month", month("date")) \
    .groupBy("month") \
    .agg(
        avg("PM25").alias("avg_PM25"),
        avg("CO").alias("avg_CO"),
        avg("O3").alias("avg_O3"),
        avg("SO2").alias("avg_SO2"),
        avg("Temperature").alias("avg_temp"),
        avg("Humidity").alias("avg_humidity")
    )

df_monthly_avg.write.format("delta") \
    .mode("overwrite") \
    .save("hdfs:///lakehouse/gold/air_quality_monthly_avg")
```

```

>>> df_monthly_avg = df_silver.withColumn("month", month("date")) \
...     .groupBy("month") \
...     .agg(
...         avg("PM25").alias("avg_PM25"),
...         avg("CO").alias("avg_CO"),
...         avg("O3").alias("avg_O3"),
...         avg("SO2").alias("avg_SO2"),
...         avg("Temperature").alias("avg_temp"),
...         avg("Humidity").alias("avg_humidity")
...     )
te") \
    .save("hdfs:///lakehouse/gold/air_quality_monthly_avg")>>>
>>> df_monthly_avg.write.format("delta") \
...     .mode("overwrite") \
...     .save("hdfs:///lakehouse/gold/air_quality_monthly_avg")
>>> |

```

```

# aqi_distribution.py
from pyspark.sql import SparkSession
df_silver.groupBy("AQI_Level").count().orderBy("count", ascending=False).show()

```

```

>>> df_silver.groupBy("AQI_Level").count().orderBy("count", ascending=False).show()
+-----+---+
|AQI_Level|count|
+-----+---+
| Average|31949|
| Good|11669|
| Unhealthy| 4696|
| Moderate| 4234|
+-----+---+
>>> |

```

```

from pyspark.sql import SparkSession
from pyspark.sql.functions import to_date, avg, when, col
from pyspark.sql.functions import hour, month, desc
df_gold = df_silver.withColumn("date_only", to_date("date")) \
    .groupBy("date_only") \
    .agg(
        avg("PM25").alias("avg_PM25"),
        avg("Temperature").alias("avg_temp"),
        avg("Humidity").alias("avg_humidity")
    ) \
    .withColumn(
        "AQI_Level",
        when(col("avg_PM25") <= 12, "Good")
            .when((col("avg_PM25") > 12) & (col("avg_PM25") <= 35), "Average")
            .when((col("avg_PM25") > 35) & (col("avg_PM25") <= 55), "Moderate")
            .otherwise("Unhealthy")
    )
df_gold.orderBy(col("avg_PM25").desc()).select("date_only", "avg_PM25",
"AQI_Level").show(10)

```

```

>>> df_gold.orderBy(col("avg_PM25").desc()).select("date_only", "avg_PM25", "AQI_Level").show(10)
+-----+-----+
| date_only | avg_PM25|AQI_Level|
+-----+-----+
| 2021-03-22 | 43.869647834210525 | Moderate |
| 2021-04-10 | 43.08636693583332 | Moderate |
| 2021-12-12 | 41.8978048478322 | Moderate |
| 2022-01-13 | 40.68837239618054 | Moderate |
| 2022-01-04 | 40.40928126625 | Moderate |
| 2022-01-06 | 40.082738114732805 | Moderate |
| 2022-01-07 | 40.07033921055556 | Moderate |
| 2021-03-21 | 39.97042310133334 | Moderate |
| 2021-04-08 | 39.65315181336736 | Moderate |
| 2022-01-15 | 39.4388702267361 | Moderate |
+-----+-----+
only showing top 10 rows

>>> |

```

## Kiểm tra

```
$ hdfs dfs -ls /lakehouse/gold/
```

```

hadoopdiemquynh@diemquynh-master:~$ hdfs dfs -ls /lakehouse/gold/
Found 4 items
drwxr-xr-x - hadoopdiemquynh supergroup          0 2025-05-24 15:02 /lakehouse/gold/air_quality_daily_avg
drwxr-xr-x - hadoopdiemquynh supergroup          0 2025-05-24 11:58 /lakehouse/gold/air_quality_gold
drwxr-xr-x - hadoopdiemquynh supergroup          0 2025-05-24 15:43 /lakehouse/gold/air_quality_hourly_avg
drwxr-xr-x - hadoopdiemquynh supergroup          0 2025-05-24 15:44 /lakehouse/gold/air_quality_monthly_avg
hadoopdiemquynh@diemquynh-master:~$ hdfs dfs -ls /lakehouse/gold/air_quality_daily_avg/
Found 2 items
drwxr-xr-x - hadoopdiemquynh supergroup          0 2025-05-24 15:02 /lakehouse/gold/air_quality_daily_avg/_delta_log
-rw-r--r--  2 hadoopdiemquynh supergroup        14570 2025-05-24 15:02 /lakehouse/gold/air_quality_daily_avg/part-00000-3b15fe2b-8278-403b-b4ec-c3793cbcf482-c000.snappy.parquet

```

```
$ hdfs dfs -ls /lakehouse/gold/air_quality_daily_avg/
```

```
$ hdfs dfs -ls /lakehouse/gold/air_quality_hourly_avg/
$ hdfs dfs -ls /lakehouse/gold/air_quality_monthly_avg/
```

```

hadoopdiemquynh@diemquynh-master:~$ hdfs dfs -ls /lakehouse/gold/air_quality_daily_avg/
Found 2 items
drwxr-xr-x - hadoopdiemquynh supergroup          0 2025-05-24 15:02 /lakehouse/gold/air_quality_daily_avg/_delta_log
-rw-r--r--  2 hadoopdiemquynh supergroup        14570 2025-05-24 15:02 /lakehouse/gold/air_quality_daily_avg/part-00000-3b15fe2b-8278-403b-b4ec-c3793cbcf482-c000.snappy.parquet
hadoopdiemquynh@diemquynh-master:~$ hdfs dfs -ls /lakehouse/gold/air_quality_hourly_avg/
Found 3 items
drwxr-xr-x - hadoopdiemquynh supergroup          0 2025-05-24 15:43 /lakehouse/gold/air_quality_hourly_avg/_delta_log
-rw-r--r--  2 hadoopdiemquynh supergroup        1914 2025-05-24 15:43 /lakehouse/gold/air_quality_hourly_avg/part-00000-5f8acd44-be96-4d91-ab4b-c6a0f02d91f7-c000.snappy.parquet
-rw-r--r--  2 hadoopdiemquynh supergroup        1914 2025-05-24 15:41 /lakehouse/gold/air_quality_hourly_avg/part-00000-fca26ee9-65dd-4034-ac14-bbbccb061fc9-c000.snappy.parquet
hadoopdiemquynh@diemquynh-master:~$ hdfs dfs -ls /lakehouse/gold/air_quality_monthly_avg/
Found 3 items
drwxr-xr-x - hadoopdiemquynh supergroup          0 2025-05-24 15:44 /lakehouse/gold/air_quality_monthly_avg/_delta_log
-rw-r--r--  2 hadoopdiemquynh supergroup        2651 2025-05-24 15:44 /lakehouse/gold/air_quality_monthly_avg/part-00000-257920ea-d574-4158-a226-d9752ac5440-7-c000.snappy.parquet
-rw-r--r--  2 hadoopdiemquynh supergroup        2651 2025-05-24 15:44 /lakehouse/gold/air_quality_monthly_avg/part-00000-fbaee4a6-755d-41ff-8088-8b67d357384f-c000.snappy.parquet
hadoopdiemquynh@diemquynh-master:~$ |

```

```
$ mkdir -p ~/air_quality_export
```

```
$ hdfs dfs -get /lakehouse/gold/air_quality_daily_avg/part-00000-3b15fe2b-8278-403b-b4ec-c3793cbcf482-c000.snappy.parquet ~/air_quality_export/daily.parquet
$ hdfs dfs -get /lakehouse/gold/air_quality_hourly_avg/part-00000-5f8acd44-be96-4d91-ab4b-c6a0f02d91f7-c000.snappy.parquet ~/air_quality_export/hourly.parquet
$ hdfs dfs -get /lakehouse/gold/air_quality_monthly_avg/part-00000-fbaee4a6-755d-41ff-8088-8b67d357384f-c000.snappy.parquet ~/air_quality_export/monthly.parquet
```

```

hadoopdiemquynh@diemquynh-master:~$ mkdir -p ~/air_quality_export
hadoopdiemquynh@diemquynh-master:~$ hdfs dfs -get /lakehouse/gold/air_quality_daily_avg/part-00000-3b15fe2b-8278-403b-b4ec-c3793cbcf482-c000.snappy.parquet ~/air_quality_export/daily.parquet
hdfs dfs -get /lakehouse/gold/air_quality_hourly_avg/part-00000-5f8acd44-be96-4d91-ab4b-c6a0f02d91f7-c000.snappy.parquet ~/air_quality_export/hourly.parquet
hadoopdiemquynh@diemquynh-master:~$ hdfs dfs -get /lakehouse/gold/air_quality_monthly_avg/part-00000-fbaee4a6-755d-41ff-8088-8b67d357384f-c000.snappy.parquet ~/air_quality_export/monthly.parquet
hadoopdiemquynh@diemquynh-master:~$ |

```

Chạy trên cmd

```
> scp hadoopdiemquynh@192.168.245.11:~/air_quality_export/*.parquet .
```

```
C:\Users\ACER>scp hadoopdiemquynh@192.168.245.11:~/air_quality_export/*.parquet .
hadoopdiemquynh@192.168.245.11's password:
daily.parquet
hourly.parquet
monthly.parquet

100%   14KB   4.6MB/s  00:00
100% 1914   934.6KB/s  00:00
100% 2651    1.3MB/s  00:00

C:\Users\ACER>
```

## Power BI

The screenshot shows the 'Get Data' interface in Power BI. On the left, there is a sidebar with a search bar and a list of categories: All, File, Database, Microsoft Fabric, Power Platform, Azure, Online Services, and Other. The 'All' category is currently selected. On the right, a list of data sources is displayed under the heading 'All'. The 'Parquet' option is highlighted with a blue selection bar. Below it, there is a tooltip with the text 'Import data from a Parquet document.' and a link to 'SQL Server database'. At the bottom of the interface, there are buttons for 'Certified Connectors' and 'Template Apps', and two main action buttons: 'Connect' (in a green box) and 'Cancel'.

Get Data

Search

All

File

Database

Microsoft Fabric

Power Platform

Azure

Online Services

Other

All

Excel Workbook

Text/CSV

XML

JSON

Folder

PDF

Parquet

SharePoint folder

Import data from a Parquet document.

SQL Server database

Access database

SQL Server Analysis Services database

Oracle database

IBM Db2 database

IBM Informix database (Beta)

IBM Netezza

MySQL database

Certified Connectors | Template Apps

Connect

Cancel

## Parquet

Basic  Advanced

URL

D:\Nam3\_Ky2\PTDLL\gold\monthly.parquet

OK

Cancel

month	avg_PM25	avg_CO	avg_O3	avg_SO2	avg_temp	avg_humidity
12	27.5109131	1074.641247	107.5086429	199.3528874	26.63715961	59.58638686
1	29.87740607	1057.311644	109.6122544	194.4656389	26.80651502	60.3106597
6	18.67007995	976.3918706	91.44029311	235.0944913	28.24102325	62.57578172
3	21.6669444	941.4497139	85.44500729	257.8715525	28.2528322	60.73301899
5	17.75331182	1068.96523	93.93988571	257.8054781	26.63888641	67.80957834
9	13.38114485	775.8403857	83.22906516	131.6904022	27.99774493	72.61686405
4	22.11744915	1103.794731	101.834215	265.2129738	28.77025936	67.0462808
8	14.75456738	762.3066884	80.20773629	140.8926695	26.41857441	58.9847414
7	14.74267823	752.5201684	77.42192478	16.122258	27.30321403	64.2740785
10	23.73105528	993.2858177	89.99674315	212.1337374	26.88261954	65.68953042
11	25.05065957	1086.535842	96.9257022	228.6624437	27.33973224	63.01425429
2	18.66733285	951.9794811	93.44337213	214.4421595	27.67570582	58.97039838

Tương tự các file parquet khác

b. Khám phá các giá trị khác

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import avg, hour, month, when, col, to_date
from pyspark.sql.functions import count, isnan, stddev, min, max

spark = SparkSession.builder \
    .appName("EDA_af") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog",
"org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .getOrCreate()

df_silver = spark.read.format("delta").load("/lakehouse/silver/air_quality/")

df_silver.printSchema()
df_silver.show(5)
```

```

>>> from pyspark.sql import SparkSession
>>> from pyspark.sql.functions import avg, hour, month, when, col, to_date
>>> from pyspark.sql.functions import count, isnan, stddev, min, max
>>> spark = SparkSession.builder \
...     .appName("EDA_af") \
...     .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
...     .config("spark.sql.catalog.spark_catalog", "org.apache.spark.sql.delta.catalog.DeltaCatalog") \
...     .getOrCreate()
...     .getOrCreate()
25/05/25 20:52:00 WARN SparkSession: Using an existing Spark session; only runtime SQL configurations will take effect.
>>> df_silver = spark.read.format("delta").load("/lakehouse/silver/air_quality/")
>>> df_silver.printSchema()
root
|-- date: timestamp (nullable = true)
|-- Station_No: string (nullable = true)
|-- TSP: double (nullable = true)
|-- PM25: double (nullable = true)
|-- O3: double (nullable = true)
|-- CO: double (nullable = true)
|-- NO2: double (nullable = true)
|-- SO2: double (nullable = true)
|-- Temperature: double (nullable = true)
|-- Humidity: double (nullable = true)
|-- AQI_Level: string (nullable = true)
|-- hour: integer (nullable = true)
|-- day: integer (nullable = true)
|-- month: integer (nullable = true)

>>> df_silver.show(5)
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|date|Station_No|TSP|PM25|O3|CO|NO2|SO2|Temperature|Humidity|AQI_Level|hour|day|month|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|2021-02-23 21:00:00|1|32.93571429|15.6047619|55.43138095|1330.451429|112.7407619|393.0|28.36190476|63.18809524|Average|21|23|2|
|2021-02-23 22:00:00|1|39.93235294|14.59411765|58.19717647|1200.603529|112.3664766|377.5882353|28.32058824|63.77352941|Average|22|23|2|
|2021-02-23 23:00:00|1|27.645|13.43666667|55.02943333|1177.897|112.7004333|372.4766667|28.33666667|64.205|Average|23|23|2|
|2021-02-24 00:00:00|1|24.38|12.365|54.7677|1267.476|112.4886667|389.07|28.305|64.735|Unhealthy|0|24|2|
|2021-02-24 01:00:00|1|22.52166667|11.63666667|53.7862|1322.293|114.3315|393.0|28.3|65.18833333|Good|1|24|2|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
only showing top 5 rows

```

## # 1. Thống kê mô tả cơ bản các cột số

```
df_silver.describe().show()
```

```

>>> df_silver.describe().show()
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|summary|Station_No|TSP|PM25|O3|CO|NO2|SO2|Temperature|Humidity|AQI_Level|hour|day|month|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|count|52548|52548|52548|52548|52548|52548|52548|52548|52548|52548|52548|52548| |
|mean|3.536062266879805|43.55036919930689|21.12603131358202|94.22952165329676|993.9237262101148|96.44458603046212|
|stddev|1.6951224724121705|35.49360034317673|14.229703061493222|33.53222370493598|560.0734301783439|67.60073376292567|
|min|1|0.0|0.0|0.0|0.0|0.0|0.0|0.0|0.0|0.0|0.0|0.0|
|max|6|938.1983333|403.6883333|377.2886|21092.57077|461.09|699.9766667|42.8083333|99.28333333|Unhealthy|23|31|12|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+

```

```

>>> df_silver.describe().show()
+-----+-----+-----+-----+-----+-----+-----+
|summary|Station_No|TSP|PM25|O3|CO|NO2|
+-----+-----+-----+-----+-----+-----+-----+
|count|52548|52548|52548|52548|52548|52548|
|mean|3.536062266879805|43.55036919930689|21.12603131358202|94.22952165329676|993.9237262101148|96.44458603046212|
|stddev|1.6951224724121705|35.49360034317673|14.229703061493222|33.53222370493598|560.0734301783439|67.60073376292567|
|min|1|0.0|0.0|0.0|0.0|0.0|
|max|6|938.1983333|403.6883333|377.2886|21092.57077|461.09|
+-----+-----+-----+-----+-----+-----+-----+

```

```

+-----+-----+-----+-----+-----+-----+
|SO2|Temperature|Humidity|AQI_Level|hour|day|month|
+-----+-----+-----+-----+-----+-----+
|52548|52548|52548|52548|52548|52548|52548|
|224.6119021902476|27.81209825881941|63.559275393562174|NULL|11.503748953337901|15.89900662516556|5.83200121793408|
|101.72726881383875|4.106419978347255|24.60991641225817|NULL|6.890869748710853|8.874126942525592|3.3460216641373255|
|2.62|18.49166667|11.32|Average|0|1|1|
|699.9766667|42.80833333|99.28333333|Unhealthy|23|31|12|
+-----+-----+-----+-----+-----+-----+-----+

```

## # 2. Kiểm tra số giá trị null và NaN trong từng cột

```
from pyspark.sql.types import DoubleType, FloatType
```

```

cols = []
for c, t in df_silver.dtypes:
    if t in ('double', 'float'):
        # Kiểm tra null hoặc isnan cho cột số
        cols.append(count(when(col(c).isNull() | isnan(col(c)), c)).alias(c))
    else:

```

```

# Chỉ kiểm tra null cho cột khác
cols.append(count(when(col(c).isNull(), c)).alias(c))

df_silver.select(cols).show()

```

```

>>> from pyspark.sql.types import DoubleType, FloatType
>>> cols = []
>>> for c, t in df_silver.dtypes:
...     if t in ('double', 'float'):
...         # Kiểm tra null hoặc isnan cho cột số
...         cols.append(count(when(col(c).isNull() | isnan(col(c)), c)).alias(c))
...     else:
...         # Chỉ kiểm tra null cho cột khác
...         cols.append(count(when(col(c).isNull(), c)).alias(c))
...
>>> df_silver.select(cols).show()
+-----+-----+-----+-----+-----+-----+-----+-----+
|date|Station_No|TSP|PM25| O3| CO|NO2|SO2|Temperature|Humidity|AQI_Level|hour|day|month|
+-----+-----+-----+-----+-----+-----+-----+-----+
|    0|          0|   0|   0|   0|   0|   0|   0|           0|           0|      0|   0|   0|   0|
+-----+-----+-----+-----+-----+-----+-----+-----+
>>>

```

```

# 3. Đếm số lượng bản ghi theo từng mức AQI_Level
df_silver.groupBy("AQI_Level").count().show()

```

```

>>> df_silver.groupBy("AQI_Level").count().show()
[Stage 110:=====
110:=====
----+
|AQI_Level|count|
----+----+
|Unhealthy| 4696|
| Average|31949|
| Good|11669|
| Moderate| 4234|
----+----+

```

- Nhận xét:
  - + **Dữ liệu nghiêng về mức "Average" (~58%)**
    - Cho thấy phần lớn thời gian chất lượng không khí ở mức trung bình — chưa nguy hiểm nhưng cũng chưa đạt chuẩn tốt.
    - + "**Good**" chiếm ~21%
    - Một phần nhỏ dữ liệu thể hiện chất lượng không khí tốt — có thể rơi vào các khung giờ ít thuận tiện, ban đêm, hoặc điều kiện thời tiết tốt.
  - + "**Unhealthy**" và "**Moderate**" chiếm tổng cộng ~16%
    - Dữ liệu ở mức cảnh báo cao (Unhealthy) chiếm ~8.5%, cảnh báo nhẹ (Moderate) ~7.7% — cần phân tích thêm vào thời gian nào trong ngày, tháng nào, hay trạm nào.

- + **Phân bố lệch (imbalanced classes)**

→ Nếu dùng cho mô hình phân loại (classification), nên xem xét xử lý mất cân bằng (ví dụ: oversampling, under-sampling, hoặc weighted loss).

```
# 4. Thống kê trung bình, min, max của PM25 theo ngày (tạo cột date_only)
```

```
df_silver = df_silver.withColumn("date_only", to_date(col("date")))

df_silver.groupBy("date_only") \
    .agg(
        avg("PM25").alias("avg_PM25"),
        min("PM25").alias("min_PM25"),
        max("PM25").alias("max_PM25")
    ).orderBy("date_only").show(10)
```

```
Σ hadoopgianghi@gianghi-mas: X Σ hadoopgianghi@gianghi-mas: X Σ hadoopgianghi@gianghi-mas: X

>>> df_silver = df_silver.withColumn("date_only", to_date(col("date")))
>>> df_silver.groupBy("date_only") \
...     .agg(
...         avg("PM25").alias("avg_PM25"),
...         min("PM25").alias("min_PM25"),
...         max("PM25").alias("max_PM25")
...     ).orderBy("date_only").show(10)
[Stage 115:=====] (21[Stage 115:=====] (46
-----+-----+-----+
| date_only |      avg_PM25 |      min_PM25 |      max_PM25 |
-----+-----+-----+
| 2021-02-23 | 13.235969722000004 | 11.381666667 | 15.6047619 |
| 2021-02-24 | 16.092616122811474 | 9.503333333 | 41.6 |
| 2021-02-25 | 13.298235805416668 | 10.95 | 15.345 |
| 2021-02-26 | 22.128216409833335 | 15.91176471 | 28.405 |
| 2021-02-27 | 32.4338610735 | 26.066666667 | 37.66 |
| 2021-02-28 | 13.457750000116667 | 9.063333333 | 27.36 |
| 2021-03-01 | 21.23695462300832 | 9.613333333 | 55.30714286 |
| 2021-03-02 | 20.212568590924377 | 10.66833333 | 52.24 |
| 2021-03-03 | 25.81056630266667 | 11.48333333 | 50.73166667 |
| 2021-03-04 | 22.466905537358475 | 10.66166667 | 43.04166667 |
-----+-----+-----+
only showing top 10 rows
```

```
# 5. Thống kê số bản ghi theo giờ trong ngày
```

```
df_silver.groupBy("hour").count().orderBy("hour").show(24)
```

hour	count
0	2148
1	2170
2	2191
3	2175
4	2175
5	2175
6	2175
7	2176
8	2177
9	2203
10	2232
11	2230
12	2226
13	2218
14	2218
15	2218
16	2238
17	2213
18	2190
19	2172
20	2162
21	2161
22	2149
23	2156

- Nhận xét: Dữ liệu phân bố đều theo giờ.

```
# Ngày bắt đầu và ngày kết thúc
df_silver.select(
    min("date").alias("start_date"),
    max("date").alias("end_date")
).show()
```

start_date	end_date
2021-02-23 21:00:00	2022-06-21 17:00:00

#### 4. Xây dựng mô hình từ dữ liệu lớp Silver

Dùng dữ liệu từ lớp Silver để xây dựng mô hình phân loại AQI\_Level, gồm các bước:

Giai đoạn	Mục tiêu
1	Đọc dữ liệu từ Delta lake (silver)
2	Chuyển đổi kiểu dữ liệu phù hợp

3	Encode biến mục tiêu AQI_Level thành số (label)
4	Chọn các đặc trưng phù hợp (CO, NO2, PM25, Temp, Humidity, v.v.)
5	Vector hóa và chuẩn hóa các đặc trưng
6	Chia dữ liệu train/test
7	Huấn luyện mô hình phân loại
8	Dự đoán và đánh giá
9	Lưu mô hình + kết quả dự đoán ra Delta

Tạo file platinum\_classify\_aqi\_from\_silver.py

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, when
from pyspark.ml.feature import VectorAssembler, StandardScaler, StringIndexer
from pyspark.ml.classification import RandomForestClassifier
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

# 1. Tạo SparkSession
spark = (
    SparkSession.builder
    .appName("MyApp")
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension")
    .config("spark.sql.catalog.spark_catalog",
"org.apache.spark.sql.delta.catalog.DeltaCatalog")
    .getOrCreate()
)

# 2. Đọc dữ liệu từ lớp Silver
df = spark.read.format("delta").load("/lakehouse/silver/air_quality_cleaned/")
df.show()
# 3. Ép kiểu dữ liệu phù hợp
df = df.select(
    col("CO").cast("double"),
    col("PM25").cast("double"),
    col("NO2").cast("double"),
    col("Temp").cast("double"),
    col("Humidity").cast("double"),
    col("Wind").cast("double"),
    col("AQI_Level").cast("double")
)
```

```

    col("NO2").cast("double"),
    col("PM25").cast("double"),
    col("Temperature").cast("double"),
    col("Humidity").cast("double"),
    col("AQI_Level"),
    col("date"),
    col("hour"),
    col("day"),
    col("month"),
    col("Station_No")
).dropna()

# 4. Encode nhãn AQI_Level: "Good" → 0, "Moderate" → 1, "Unhealthy" → 2, ...
indexer = StringIndexer(inputCol="AQI_Level", outputCol="label")
df = indexer.fit(df).transform(df)

# 5. Vector hóa các đặc trưng đầu vào
feature_cols = ["CO", "NO2", "PM25", "Temperature", "Humidity"]
assembler = VectorAssembler(inputCols=feature_cols, outputCol="raw_features")
df_vector = assembler.transform(df)

# 6. Chuẩn hóa đặc trưng
scaler = StandardScaler(inputCol="raw_features", outputCol="features", withMean=True,
withStd=True)
scaler_model = scaler.fit(df_vector)
df_scaled = scaler_model.transform(df_vector)

# 7. Chia dữ liệu train/test
train_data, test_data = df_scaled.randomSplit([0.8, 0.2], seed=123)

# 8. Huấn luyện mô hình Random Forest
rf = RandomForestClassifier(labelCol="label", featuresCol="features", numTrees=50)
model = rf.fit(train_data)

# 9. Dự đoán và đánh giá
predictions = model.transform(test_data)
evaluator = MulticlassClassificationEvaluator(labelCol="label",
predictionCol="prediction", metricName="accuracy")
accuracy = evaluator.evaluate(predictions)
print(f"\n\n\nĐộ chính xác mô hình: {accuracy:.2%}")

# 10. Lưu mô hình và kết quả dự đoán ra Delta
model.save("/lakehouse/platinum/air_quality_classified /aqi_rf_model_from_silver")
predictions.select("CO", "NO2", "PM25", "Temperature", "Humidity", "AQI_Level",
"label", "prediction") \
.write.format("delta") \
.mode("overwrite") \
.save("/lakehouse/platinum/air_quality_classified/air_quality_predictions_from_silver")

# Dừng SparkSession
spark.stop()

```

```

from pyspark.sql import SparkSession
from pyspark.sql.functions import col, when
from pyspark.ml.feature import VectorAssembler, StandardScaler, StringIndexer
from pyspark.ml.classification import RandomForestClassifier
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

# 1. Tạo SparkSession
spark = (
    SparkSession.builder
        .appName("MyApp")
        .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension")
        .config("spark.sql.catalog.spark_catalog", "org.apache.spark.sql.delta.catalog.DeltaCatalog")
        .getOrCreate()
)

# 2. Đọc dữ liệu từ lớp Silver
df = spark.read.format("delta").load("/lakehouse/silver/air_quality_cleaned/")
df.show()

# 3. Ép kiểu dữ liệu phù hợp
df = df.select(
    col("CO").cast("double"),
    col("NO2").cast("double"),
    col("PM25").cast("double"),
    col("Temperature").cast("double"),
    col("Humidity").cast("double"),
    col("AQI_Level"),
    col("date"),
    col("hour"),
    col("day"),
    col("month"),
    col("Station_No")
).dropna()

# 4. Encode nhãn AQI_Level: "Good" → 0, "Moderate" → 1, "Unhealthy" → 2, ...
indexer = StringIndexer(inputCol="AQI_Level", outputCol="label")
df = indexer.fit(df).transform(df)

# 5. Vector hóa các đặc trưng đầu vào
feature_cols = ["CO", "NO2", "PM25", "Temperature", "Humidity"]
assembler = VectorAssembler(inputCols=feature_cols, outputCol="raw_features")
df_vector = assembler.transform(df)

# 6. Chuẩn hóa đặc trưng
scaler = StandardScaler(inputCol="raw_features", outputCol="features", withMean=True, withStd=True)
scaler_model = scaler.fit(df_vector)
df_scaled = scaler_model.transform(df_vector)

# 7. Chia dữ liệu train/test
train_data, test_data = df_scaled.randomSplit([0.8, 0.2], seed=123)

# 8. Huấn luyện mô hình Random Forest
rf = RandomForestClassifier(labelCol="label", featuresCol="features", numTrees=50)
model = rf.fit(train_data)

# 9. Dự đoán và đánh giá
predictions = model.transform(test_data)
evaluator = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction", metricName="accuracy")
accuracy = evaluator.evaluate(predictions)
print(f"Độ chính xác mô hình: {accuracy:.2%}")

# 10. Lưu mô hình và kết quả dự đoán ra Delta
model.save("/lakehouse/platinum/air_quality_classified/aqi_rf_model_from_silver")
predictions.select("CO", "NO2", "PM25", "Temperature", "Humidity", "AQI_Level", "label", "prediction") \
    .write.format("delta") \
    .mode("overwrite") \
    "platinum_classify_aqi_from_silver.py" 72L, 2654B

```

## Đầu ra

Dạng	Đường dẫn
Mô hình đã huấn luyện	/lakehouse/platinum/air_quality_classified/aqi_rf_model_from_silver

Dự đoán  
và nhän  
thực tế

/lakehouse/platinum/air\_quality\_classified/air\_quality\_predictions\_from\_silver

## Chạy file

```
$ spark-submit --packages io.delta:delta-spark_2.12:3.0.0  
platinum_classify_aqi_from_silver.py
```



```
hadoopquynhthu@quynhthu:~$ spark-submit --packages io.delta:delta-spark_2.12:3.0.0 platinum_classify_aqi_from_silver.py  
:: loading settings :: url = jar:file:/home/hadoopquynhthu/spark/jars/ivy-2.5.1.jar!/org/apache/ivy/core/settings/ivysettings.xml  
Ivy Default Cache set to: /home/hadoopquynhthu/.ivy2/cache  
The jars for the packages stored in: /home/hadoopquynhthu/.ivy2/jars  
io.delta#delta-spark_2.12 added as a dependency  
:: resolving dependencies :: org.apache.spark#spark-submit-parent-ac6156e7-3d9e-402d-bc74-237e0489ae83;1.0  
confs: [default]  
found io.delta#delta-spark_2.12;3.0.0 in central  
found io.delta#delta-storage;3.0.0 in central  
found org.antlr#antlr4-runtime;4.9.3 in central  
:: resolution report :: resolve 322ms :: artifacts dl 28ms  
:: modules in use:  
io.delta#delta-spark_2.12;3.0.0 from central in [default]  
io.delta#delta-storage;3.0.0 from central in [default]  
org.antlr#antlr4-runtime;4.9.3 from central in [default]  
-----  
| | modules || artifacts  
| conf | number| search|dwnlded|evicted|| number|dwnlded  
|-----  
| default | 3 | 0 | 0 | 0 || 3 | 0  
|-----  
: retrieving :: org.apache.spark#spark-submit-parent-ac6156e7-3d9e-402d-bc74-237e0489ae83  
conf: [default]  
0 artifacts copied, 3 already retrieved (0kB/10ms)  
25/05/25 10:10:02 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable  
25/05/25 10:10:05 INFO SparkContext: Running Spark version 3.5.3  
25/05/25 10:10:05 INFO SparkContext: OS info Linux, 6.8.0-56-generic, amd64  
25/05/25 10:10:05 INFO SparkContext: Java version 1.8.0_442  
25/05/25 10:10:05 INFO ResourceUtils: ======
```

```

hadoopquynhthu@quynhthu: ~ hadoopquynhthu@quynhthu: ~ + v - □ ×
25/05/25 10:11:02 INFO DAGScheduler: Job 24 finished: collectAsMap at MulticlassMetrics.scala:61, took 0.906073 s

Dữ chính xác mô hình: 96.82%
25/05/25 10:11:02 INFO deprecation: mapred.output.dir is deprecated. Instead, use mapreduce.output.fileoutputformat.outputdir
25/05/25 10:11:02 INFO HadoopMapReduceCommitProtocol: Using output committer class org.apache.hadoop.mapred.FileOutputCommitter
25/05/25 10:11:02 INFO FileOutputCommitter: File Output Committer Algorithm version is 1
25/05/25 10:11:02 INFO FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
25/05/25 10:11:02 INFO SparkContext: Starting job: runJob at SparkHadoopWriter.scala:83
25/05/25 10:11:02 INFO DAGScheduler: Got job 25 (runJob at SparkHadoopWriter.scala:83) with 1 output partitions
25/05/25 10:11:02 INFO DAGScheduler: Final stage: ResultStage 44 (runJob at SparkHadoopWriter.scala:83)
25/05/25 10:11:02 INFO MemoryStore: Block broadcast_48 stored as values in memory (estimated size 102.5 KiB, free 361.4 MiB)
25/05/25 10:11:02 INFO MemoryStore: Block broadcast_48_piece0 stored as bytes in memory (estimated size 37.3 KiB, free 361.4 MiB)
25/05/25 10:11:02 INFO BlockManagerInfo: Added broadcast_48_piece0 in memory on quynhthu-master:40903 (size: 37.3 KiB, free: 365.5 MiB)
25/05/25 10:11:02 INFO SparkContext: Created broadcast 48 from broadcast at DAGScheduler.scala:1585
25/05/25 10:11:02 INFO DAGScheduler: Submitting 1 missing tasks from ResultStage 44 (MapPartitionsRDD[114] at saveAsTextFile at ReadWrite.scala:413) (first 15 tasks are for partitions Vector(0))
25/05/25 10:11:02 INFO TaskSchedulerImpl: Adding task set 44.0 with 1 tasks resource profile 0
25/05/25 10:11:02 INFO TaskSetManager: Starting task 0.0 in stage 44.0 (TID 441) (quynhthu-master, executor driver, partition 0, PROCESS_LOCAL, 10440) by

```

```

hadoopquynhthu@quynhthu: ~ hadoopquynhthu@quynhthu: ~ + v - □ ×
in 1 ms
25/05/25 10:11:12 INFO Executor: Finished task 0.0 in stage 60.0 (TID 552). 7038 bytes result sent to driver
25/05/25 10:11:12 INFO TaskSetManager: Finished task 0.0 in stage 60.0 (TID 552) in 62 ms on quynhthu-master (executor driver) (1/1)
25/05/25 10:11:12 INFO TaskSchedulerImpl: Removed TaskSet 60.0, whose tasks have all completed, from pool
25/05/25 10:11:12 INFO DAGScheduler: ResultStage 60 ($anonfun$recordDeltaOperationInternal$1 at DatabricksLogging.scala:128) finished in 0.087 s
25/05/25 10:11:12 INFO DAGScheduler: Job 36 is finished. Cancelling potential speculative or zombie tasks for this job
25/05/25 10:11:12 INFO TaskSchedulerImpl: Killing all running tasks in stage 60: Stage finished
25/05/25 10:11:12 INFO DAGScheduler: Job 36 finished: $anonfun$recordDeltaOperationInternal$1 at DatabricksLogging.scala:128, took 0.089908 s
25/05/25 10:11:12 INFO Snapshot: [tableId=9c9e7792-7ee0-4daa-92e6-5d6ffe766871] DELTA: Done
25/05/25 10:11:12 INFO OptimisticTransaction: [tableId=84246244, txId=55003331] Committed delta #0 to hdfs://quynhthu-master:9000/lakehouse/platinum/air_quality_classified/air_quality_predictions_from_silver/_delta_log
25/05/25 10:11:12 INFO SparkContext: SparkContext is stopping with exitCode 0.
25/05/25 10:11:12 INFO SparkUI: Stopped Spark web UI at http://quynhthu-master:4040
25/05/25 10:11:12 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
25/05/25 10:11:12 INFO MemoryStore: MemoryStore cleared
25/05/25 10:11:12 INFO BlockManager: BlockManager stopped
25/05/25 10:11:12 INFO BlockManagerMaster: BlockManagerMaster stopped
25/05/25 10:11:12 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
25/05/25 10:11:12 INFO SparkContext: Successfully stopped SparkContext
25/05/25 10:11:13 INFO ShutdownHookManager: Shutdown hook called
25/05/25 10:11:13 INFO ShutdownHookManager: Deleting directory /tmp/spark-e9538a43-1452-4615-a72f-ef8181da2eb6/pyspark-35c728eb-54f3-48c6-aa80-52103adf2f65
25/05/25 10:11:13 INFO ShutdownHookManager: Deleting directory /tmp/spark-12370d98-6b64-44cd-9ffc-29c584bf25c0
25/05/25 10:11:13 INFO ShutdownHookManager: Deleting directory /tmp/spark-e9538a43-1452-4615-a72f-ef8181da2eb6
hadoopquynhthu@quynhthu-master:~$ |

```

## Kiểm tra kết quả dự đoán

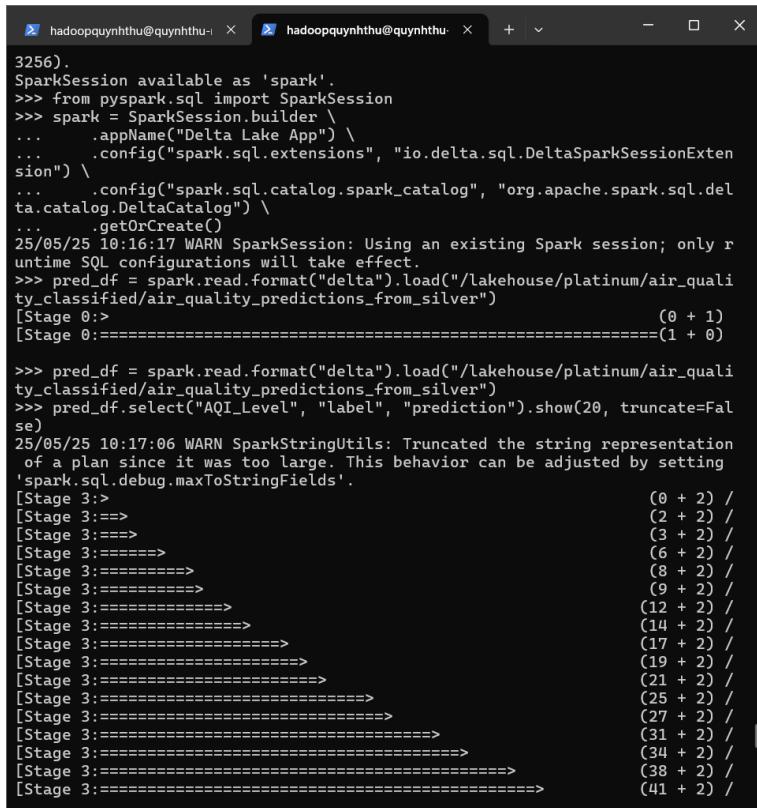
### 1. Đọc lại dữ liệu dự đoán từ Delta

Sau khi mô hình chạy xong, ta đã lưu kết quả dự đoán ở:

*/lakehouse/platinum/air\_quality\_classified/air\_quality\_predictions\_from\_silver*

Đọc lại bảng PySpark:

```
>> from pyspark.sql import SparkSession
spark = SparkSession.builder \
    .appName("Delta Lake App") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog",
"org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .getOrCreate()
>> pred_df =
spark.read.format("delta").load("/lakehouse/platinum/air_quality_classified/air_quality_predictions_from_silver")
>> pred_df.select("AQI_Level", "label", "prediction").show(20, truncate=False)
```



The screenshot shows a terminal window with two tabs. The current tab displays the PySpark session code and its execution results. The code reads a Delta table and shows the first 20 rows of the 'pred\_df' DataFrame, which includes columns 'AQI\_Level', 'label', and 'prediction'. The terminal also shows several WARN messages related to Spark session configuration and string truncation.

```
3256).
SparkSession available as 'spark'.
>>> from pyspark.sql import SparkSession
>>> spark = SparkSession.builder \
...     .appName("Delta Lake App") \
...     .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
...     .config("spark.sql.catalog.spark_catalog", "org.apache.spark.sql.delta.catalog.DeltaCatalog") \
...     .getOrCreate()
>>> pred_df = spark.read.format("delta").load("/lakehouse/platinum/air_quality_classified/air_quality_predictions_from_silver")
[Stage 0:>                                (0 + 1)
[Stage 0:=====                           (1 + 0)

>>> pred_df = spark.read.format("delta").load("/lakehouse/platinum/air_quality_classified/air_quality_predictions_from_silver")
>>> pred_df.select("AQI_Level", "label", "prediction").show(20, truncate=False)
25/05/25 10:17:06 WARN SparkStringUtils: Truncated the string representation
of a plan since it was too large. This behavior can be adjusted by setting
'spark.sql.debug.maxToStringFields'.
[Stage 3:>                                (0 + 2) /
[Stage 3:==>                            (2 + 2) /
[Stage 3:==>                            (3 + 2) /
[Stage 3:====>                        (6 + 2) /
[Stage 3:====>                        (8 + 2) /
[Stage 3:====>                        (9 + 2) /
[Stage 3:====>                        (12 + 2) /
[Stage 3:====>                        (14 + 2) /
[Stage 3:====>                        (17 + 2) /
[Stage 3:====>                        (19 + 2) /
[Stage 3:====>                        (21 + 2) /
[Stage 3:====>                        (25 + 2) /
[Stage 3:====>                        (27 + 2) /
[Stage 3:====>                        (31 + 2) /
[Stage 3:====>                        (34 + 2) /
[Stage 3:====>                        (38 + 2) /
[Stage 3:====>                        (41 + 2) /
```

```

hadoopquynhthu@quynhthu-OptiPlex-5090: ~ % hadoopquynhthu@quynhthu-OptiPlex-5090: ~ %
[Stage 3:=====> (41 + 2) /
[Stage 3:=====> (45 + 2) /
[Stage 3:=====> (49 + 1) /
[Stage 8:=====> (26 + 3) /
[Stage 8:=====> (41 + 2) /
[Stage 9:> (0 + 1)

+-----+-----+
|AQI_Level|label|prediction|
+-----+-----+
|Average |0.0  |0.0
|Unhealthy|2.0  |3.0
|Unhealthy|2.0  |3.0
|Moderate |3.0  |3.0
|Average  |0.0  |0.0
|Average  |0.0  |3.0
+-----+
only showing top 20 rows

>>> |

```

Phân tích kết quả:

AQI_Level	label	prediction	Diễn giải
Average	0	0	Đoán đúng
Unhealthy	2	3	Đoán sai, mô hình nhầm với nhãn 3.0 (tức "Moderate")
Moderate	3	3	Đoán đúng
Average	0	3	Đoán sai, mô hình nhầm "Average" là "Moderate"

## 2. Bổ sung

- Tạo confusion matrix

PySpark không hỗ trợ trực tiếp về confusion matrix, nhưng ta có thể làm theo cách: Đếm số lượng từng tổ hợp label - prediction bằng PySpark

- Đánh giá chi tiết bằng Precision, Recall, F1-score
- Kiểm tra độ quan trọng của các đặc trưng (feature importances)

Cập nhật file platinum\_classify\_aqi\_from\_silver.py

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, count
from pyspark.ml.feature import StringIndexer, VectorAssembler, StandardScaler
from pyspark.ml.classification import RandomForestClassifier
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

# 1. Khởi tạo SparkSession
spark = (
    SparkSession.builder
    .appName("MyApp")
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension")
    .config("spark.sql.catalog.spark_catalog",
"org.apache.spark.sql.delta.catalog.DeltaCatalog")
    .getOrCreate()
)

# 2. Đọc dữ liệu từ lớp Silver
df = spark.read.format("delta").load("/lakehouse/silver/air_quality_cleaned/")

# 3. Ép kiểu dữ liệu phù hợp
df = df.select(
    col("CO").cast("double"),
    col("NO2").cast("double"),
    col("PM25").cast("double"),
    col("Temperature").cast("double"),
    col("Humidity").cast("double"),
    col("AQI_Level"),
    col("date"),
    col("hour"),
    col("day"),
    col("month"),
    col("Station_No")
).dropna()

# 4. Encode nhãn AQI_Level thành số
indexer = StringIndexer(inputCol="AQI_Level", outputCol="label")
indexer_model = indexer.fit(df)
df = indexer_model.transform(df)

# 5. Vector hóa các đặc trưng đầu vào
feature_cols = ["CO", "NO2", "PM25", "Temperature", "Humidity"]
assembler = VectorAssembler(inputCols=feature_cols, outputCol="raw_features")
df_vector = assembler.transform(df)

# 6. Chuẩn hóa đặc trưng
scaler = StandardScaler(inputCol="raw_features", outputCol="features", withMean=True,
withStd=True)
scaler_model = scaler.fit(df_vector)
df_scaled = scaler_model.transform(df_vector)

# 7. Chia dữ liệu train/test
train_data, test_data = df_scaled.randomSplit([0.8, 0.2], seed=123)

# 8. Huấn luyện mô hình Random Forest
```

```

rf = RandomForestClassifier(labelCol="label", featuresCol="features", numTrees=50,
seed=123)
rf_model = rf.fit(train_data)

# 9. Dự đoán và đánh giá mô hình RF
pred_rf = rf_model.transform(test_data)

evaluator = MulticlassClassificationEvaluator(labelCol="label",
predictionCol="prediction")

accuracy = evaluator.setMetricName("accuracy").evaluate(pred_rf)
f1 = evaluator.setMetricName("f1").evaluate(pred_rf)
precision = evaluator.setMetricName("weightedPrecision").evaluate(pred_rf)
recall = evaluator.setMetricName("weightedRecall").evaluate(pred_rf)

print(f"\n\n\nRandom Forest Performance:")
print(f"Accuracy: {accuracy:.4f}")
print(f"F1 Score: {f1:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")

# Tạo ánh xạ từ chỉ số label -> tên AQI_Level
label_names = indexer_model.labels

# In ma trận nhầm lẫn với tên nhãn trực tiếp
confusion_df = pred_rf.groupBy("label", "prediction") \
    .agg(count("*").alias("count")) \
    .orderBy("label", "prediction")

confusion_data = confusion_df.collect()

print("\n\n\nConfusion Matrix - Random Forest:")
print(f"{'Label':<5} {'Label Name':<30} {'Predicted':<10} {'Prediction Name':<30} \
{'Count':<6}")
print("-" * 90)
for row in confusion_data:
    label_idx = int(row['label'])
    pred_idx = int(row['prediction'])
    label_name = label_names[label_idx]
    pred_name = label_names[pred_idx]
    count_val = row['count']
    print(f"{label_idx:<5} {label_name:<30} {pred_idx:<10} {pred_name:<30} \
{count_val:<6}")

# 10. Kiểm tra độ quan trọng của các đặc trưng (feature importances)
print("\n\n\nFeature Importances - Random Forest:")
for feature, importance in zip(feature_cols, rf_model.featureImportances):
    print(f"{feature}: {importance:.4f}")

# 11. Lưu mô hình Random Forest
rf_model.save("/lakehouse/platinum/air_quality_classified/aqi_rf_model_from_silver")

# 12. Lưu kết quả dự đoán của RF ra Delta
pred_rf.select("CO", "NO2", "PM25", "Temperature", "Humidity", "AQI_Level", "label",
"prediction") \

```

```

[hadoopquynhthu@quynhthu ~] [hadoopquynhthu@quynhthu ~] + 
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, count
from pyspark.ml.feature import StringIndexer, VectorAssembler, StandardScaler
from pyspark.ml.classification import RandomForestClassifier
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

# 1. Khởi tạo SparkSession
spark = (
    SparkSession.builder
        .appName("MyApp")
        .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension")
        .config("spark.sql.catalog.spark_catalog", "org.apache.spark.sql.delta.catalog.DeltaCatalog")
        .getOrCreate()
)

# 2. Đọc dữ liệu từ lớp Silver
df = spark.read.format("delta").load("/lakehouse/silver/air_quality_cleaned/*")

# 3. Èp kiểu dữ liệu phù hợp
df = df.select(
    col("CO").cast("double"),
    col("NO2").cast("double"),
    col("PM25").cast("double"),
    col("Temperature").cast("double"),
    col("Humidity").cast("double"),
    col("AQI_Level"),
    col("AQI_Hour"),
    col("hour"),
    col("day"),
    col("month"),
    col("Station_No")
).dropna()

# 4. Encodes mã AQI_Level thành số
indexer = StringIndexer(inputCol="AQI_Level", outputCol="label")
indexer_model = indexer.fit(df)
df = indexer_model.transform(df)

# 5. Vector hóa các đặc trưng đầu vào
feature_cols = ["PM25", "Temperature", "Humidity"]
assembler = VectorAssembler(inputCols=feature_cols, outputCol="raw_features")
df_vector = assembler.transform(df)

# 6. Chuẩn hóa đặc trưng
scaler = StandardScaler(inputCol="raw_features", outputCol="features", withMean=True, withStd=True)
scaler_model = scaler.fit(df_vector)
df_scaled = scaler_model.transform(df_vector)

# 7. Chia dữ liệu train/test
"platinum_classify_aqi_from_silver.py" 188L, 4027B
[hadoopquynhthu@quynhthu ~] [hadoopquynhthu@quynhthu ~] + 
27,1 Top

# 8. Huấn luyện mô hình Random Forest
rf = RandomForestClassifier(labelCol="label", featuresCol="features", numTrees=50, seed=123)
rf_model = rf.fit(train_data)

# 9. Dự đoán và đánh giá mô hình RF
pred_rf = rf_model.transform(test_data)

evaluator = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction")

accuracy = evaluator.setMetricName("accuracy").evaluate(pred_rf)
f1 = evaluator.setMetricName("F1").evaluate(pred_rf)
precision = evaluator.setMetricName("weightedPrecision").evaluate(pred_rf)
recall = evaluator.setMetricName("weightedRecall").evaluate(pred_rf)

print("\n\n\nRandom Forest Performance:")
print(f"Accuracy: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")

# Tạo ánh xạ từ chỉ số label → tên AQI_Level
label_names = indexer_model.labels

# In ma trận nhầm lẫn với tên nhãn trục tiếp
confusion_df = pred_rf.groupby("label", "prediction") \
    .agg(count("*").alias("count")) \
    .orderby("label", "prediction")
confusion_data = confusion_df.collect()

print("\n\n\nConfusion Matrix - Random Forest:")
print(f"{'Label':<5} {'Predicted':<10} {'Count':<6}")
print("-" * 98)
for row in confusion_data:
    label_idx = int(row["label"])
    pred_idx = int(row["prediction"])
    label_name = label_names[label_idx]
    pred_name = label_names[pred_idx]
    count_val = row["count"]
    print(f"{'label':<5} {'pred':<10} {'label_name':<30} {'pred_name':<30} {'count':<6}")
    print(f"{'label':<5} {'pred':<10} {label_name:<30} {pred_name:<30} {count_val:<6}")

# 10. Kiểm tra độ quan trọng của các đặc trưng (feature importances)
print("\n\n\nFeature Importances - Random Forest:")
for feature, importance in zip(feature_names, rf_model.featureImportances):
    print(f"{'feature':<10} {importance:.4f}")

# 11. Lưu mô hình Random Forest
rf_model.save("/lakehouse/platinum/air_quality_classified/aqi_rf_model_from_silver")
94,1 84%

```

## Xoá file kết quả:

```
hdfs dfs -rm -r /lakehouse/platinum/air_quality_classified/*
```

```

[hadoopquynhthu@quynhthu ~] [hadoopquynhthu@quynhthu ~] + 
[hadoopquynhthu@quynhthu-master:~$ hdfs dfs -rm -r /lakehouse/platinum/air_quality_classifie
d/*|]

```

## Chạy file

```
$ spark-submit --packages io.delta:delta-spark_2.12:3.0.0
platinum_classify_aqi_from_silver.py
```



```
hadoopquynhthu@quynhthu:~$ spark-submit --packages io.delta:delta-spark_2.12:3.0.0 p
latinum_classify_aqi_from_silver.py
:: loading settings :: url = jar:file:/home/hadoopquynhthu/spark/jars/ivy-2.5.1.jar!/org/apache/ivy/core/settings/ivysettings.xml
Ivy Default Cache set to: /home/hadoopquynhthu/.ivy2/cache
The jars for the packages stored in: /home/hadoopquynhthu/.ivy2/jars
io.delta#delta-spark_2.12 added as a dependency
:: resolving dependencies :: org.apache.spark#spark-submit-parent-c8571b7b-eb45-4b1d-b648-9
0a9f2d9b8be;1.0
  confs: [default]
    found io.delta#delta-spark_2.12;3.0.0 in central
    found io.delta#delta-storage;3.0.0 in central
    found org.antlr#antlr4-runtime;4.9.3 in central
:: resolution report :: resolve 232ms :: artifacts dl 11ms
  :: modules in use:
    io.delta#delta-spark_2.12;3.0.0 from central in [default]
    io.delta#delta-storage;3.0.0 from central in [default]
    org.antlr#antlr4-runtime;4.9.3 from central in [default]
-----
|       conf      |           modules           ||   artifacts   | | | | |
|               |   number| search|downloaded|evicted||   number|downloaded|
|       default   |     3   |   0   |   0   |   0   ||   3   |   0   |
-----
:: retrieving :: org.apache.spark#spark-submit-parent-c8571b7b-eb45-4b1d-b648-90a9f2d9b8be
  confs: [default]
    0 artifacts copied, 3 already retrieved (0kB/10ms)
25/05/25 10:27:35 WARN NativeCodeLoader: Unable to load native-hadoop library for your plat
form... using builtin-java classes where applicable
25/05/25 10:27:37 INFO SparkContext: Running Spark version 3.5.3
25/05/25 10:27:37 INFO SparkContext: OS info Linux, 6.8.0-56-generic, amd64
25/05/25 10:27:37 INFO SparkContext: Java version 1.8.0_442
25/05/25 10:27:38 INFO ResourceUtils: =====
25/05/25 10:27:38 INFO ResourceUtils: No custom resources configured for spark.driver.
25/05/25 10:27:38 INFO ResourceUtils: =====
=====
25/05/25 10:27:38 INFO SparkContext: Submitted application: MyApp
25/05/25 10:27:38 INFO ResourceProfile: Default ResourceProfile created, executor resources
: Map(cores -> name: cores, amount: 1, script: , vendor: , memory -> name: memory, amount:
1024, script: , vendor: , offHeap -> name: offHeap, amount: 0, script: , vendor: ), task re
sources: Map(cpus -> name: cpus, amount: 1.0)
25/05/25 10:27:38 INFO ResourceProfile: Limiting resource is cpu
25/05/25 10:27:38 INFO ResourceProfileManager: Added ResourceProfile id: 0
25/05/25 10:27:38 INFO SecurityManager: Changing view acls to: hadoopquynhthu
25/05/25 10:27:38 INFO SecurityManager: Changing modify acls to: hadoopquynhthu
25/05/25 10:27:38 INFO SecurityManager: Changing view acls groups to:
25/05/25 10:27:38 INFO SecurityManager: Changing modify acls groups to:
25/05/25 10:27:38 INFO SecurityManager: SecurityManager: authentication disabled; ui acls d
```

```

la:61, took 0.470369 s

Random Forest Performance:
Accuracy: 0.9627
F1 Score: 0.9629
Precision: 0.9658
Recall: 0.9627
25/05/25 10:28:33 INFO BlockManagerInfo: Removed broadcast_52_piece0 on quynhthu-master:355
41 in memory (size: 36.8 Kib, free: 365.6 MiB)
25/05/25 10:28:34 INFO BlockManagerInfo: Removed broadcast_54_piece0 on quynhthu-master:355
41 in memory (size: 2.8 Kib, free: 365.6 MiB)
25/05/25 10:28:34 INFO BlockManagerInfo: Removed broadcast_53_piece0 on quynhthu-master:355
41 in memory (size: 125.3 Kib, free: 365.8 MiB)
25/05/25 10:28:34 INFO PrepareDeltaScan: DELTA: Filtering files for query
25/05/25 10:28:34 INFO SparkContext: Starting job: collect at /home/hadoopquynhthu/platinum
_classify_aqi_from_silver.py:80
25/05/25 10:28:34 INFO DAGScheduler: Got job 27 (collect at /home/hadoopquynhthu/platinum_c
lassify_aqi_from_silver.py:80) with 50 output partitions
25/05/25 10:28:34 INFO DAGScheduler: Final stage: ResultStage 49 (collect at /home/hadoopqu
ynhthu/platinum_classify_aqi_from_silver.py:80)
25/05/25 10:28:34 INFO DAGScheduler: Parents of final stage: List(ShuffleMapStage 48)
25/05/25 10:28:34 INFO DAGScheduler: Missing parents: List()
25/05/25 10:28:34 INFO DAGScheduler: Submitting ResultStage 49 (MapPartitionsRDD[135] at co
llect at /home/hadoopquynhthu/platinum_classify_aqi_from_silver.py:80), which has no missin
g parents
25/05/25 10:28:34 INFO MemoryStore: Block broadcast_55 stored as values in memory (estimat
e d size 725.3 Kib, free 361.5 MiB)
25/05/25 10:28:34 INFO MemoryStore: Block broadcast_55_piece0 stored as bytes in memory (es
timated size 165.6 Kib, free 361.4 MiB)
25/05/25 10:28:34 INFO BlockManagerInfo: Added broadcast_55_piece0 in memory on quynhthu-ma
ster:35541 (size: 165.6 Kib, free: 365.6 MiB)
25/05/25 10:28:34 INFO DAGScheduler: Created broadcast 55 from broadcast at DAGScheduler.sc
ala:1585
25/05/25 10:28:34 INFO DAGScheduler: Submitting 50 missing tasks from ResultStage 49 (MapPa
rtitionsRDD[135] at collect at /home/hadoopquynhthu/platinum_classify_aqi_from_silver.py:80
) (first 15 tasks are for partitions Vector(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 1
4))
25/05/25 10:28:34 INFO TaskSchedulerImpl: Adding task set 49.0 with 50 tasks resource profi
le 0
25/05/25 10:28:34 INFO TaskSetManager: Starting task 0.0 in stage 49.0 (TID 501) (quynhthu-
master, executor driver, partition 0, PROCESS_LOCAL, 9688 bytes)
25/05/25 10:28:34 INFO TaskSetManager: Starting task 1.0 in stage 49.0 (TID 502) (quynhthu-
master, executor driver, partition 1, PROCESS_LOCAL, 9688 bytes)
25/05/25 10:28:34 INFO Executor: Running task 0.0 in stage 49.0 (TID 501)
25/05/25 10:28:34 INFO Executor: Running task 1.0 in stage 49.0 (TID 502)
25/05/25 10:28:34 INFO BlockManager: Found block rdd_18_1 locally
25/05/25 10:28:34 INFO BlockManager: Found block rdd_18_0 locally

```

## Đánh giá hiệu năng mô hình:

Metric	Giá trị	Nhận xét
Accuracy	0.9627	Mô hình phân loại đúng khoảng 96.27% các mẫu trên tập test. Đây là độ chính xác cao.
F1 Score	0.9629	Giá trị F1 Score cao phản ánh mô hình cân bằng tốt giữa Precision và Recall.
Precision	0.9658	Mô hình dự đoán ít false positives, tức là các dự đoán dương tính thường chính xác.
Recall	0.9627	Mô hình dự đoán được đa số các nhãn thực sự đúng, tức là ít bỏ sót.

Nhìn chung, các chỉ số đều rất tốt → mô hình hoạt động hiệu quả và ổn định.

```

hadoopquynhthu@quynhthu: ~      hadoopquynhthu@quynhthu: ~ + v - □ ×
25/05/25 10:28:37 INFO DAGScheduler: ResultStage 57 (collect at /home/hadoopquynhthu/platin
um_classify_aqi_from_silver.py:80) finished in 0.059 s
25/05/25 10:28:37 INFO DAGScheduler: Job 31 is finished. Cancelling potential speculative o
r zombie tasks for this job
25/05/25 10:28:37 INFO TaskSchedulerImpl: Killing all running tasks in stage 57: Stage fini
shed
25/05/25 10:28:37 INFO DAGScheduler: Job 31 finished: collect at /home/hadoopquynhthu/plati
num_classify_aqi_from_silver.py:80, took 0.065381 s

Confusion Matrix - Random Forest:
Label Label Name          Predicted  Prediction Name   Count
-----+-----+-----+-----+-----+-----+
0    Average             0           Average            6312
0    Average             2           Unhealthy          76
1    Good                1           Good              2329
1    Good                2           Unhealthy          56
2    Unhealthy           0           Average            85
2    Unhealthy           2           Unhealthy          864
2    Unhealthy           3           Moderate           2
3    Moderate            0           Average            71
3    Moderate            2           Unhealthy          104
3    Moderate            3           Moderate           660

Feature Importances - Random Forest:
CO: 0.0160
NO2: 0.0061
PM25: 0.9722
Temperature: 0.0033
Humidity: 0.0025
25/05/25 10:28:37 INFO deprecation: mapred.output.dir is deprecated. Instead, use mapreduce
.output.fileoutputformat.outputdir
25/05/25 10:28:37 INFO HadoopMapReduceCommitProtocol: Using output committer class org.apache.
hadoop.mapred.FileOutputCommitter
25/05/25 10:28:37 INFO FileOutputCommitter: File Output Committer Algorithm version is 1
25/05/25 10:28:37 INFO FileOutputCommitter: FileOutputCommitter skip cleanup _temporary fol
ders under output directory:false, ignore cleanup failures: false
25/05/25 10:28:37 INFO SparkContext: Starting job: runJob at SparkHadoopWriter.scala:83
25/05/25 10:28:37 INFO DAGScheduler: Got job 32 (runJob at SparkHadoopWriter.scala:83) with
1 output partitions
25/05/25 10:28:37 INFO DAGScheduler: Final stage: ResultStage 58 (runJob at SparkHadoopWrit
er.scala:83)
25/05/25 10:28:37 INFO DAGScheduler: Parents of final stage: List()
25/05/25 10:28:37 INFO DAGScheduler: Missing parents: List()
25/05/25 10:28:37 INFO DAGScheduler: Submitting ResultStage 58 (MapPartitionsRDD[150] at sa
veAsTextFile at ReadWrite.scala:413), which has no missing parents

```

Nhãn thực	Dự đoán đúng	Tỷ lệ đúng tương đối
0.0 ("Average")	$6312 / (6312 + 76) = 6312 / 6388 \approx 98.81\%$	Rất tốt
1.0 ("Good")	$2329 / (2329 + 56) = 2329 / 2385 \approx 97.65\%$	Rất tốt
2.0 ("Unhealthy")	$864 / (864 + 85 + 2) = 864 / 951 \approx 90.85\%$	Tốt
3.0 ("Moderate")	$660 / (660 + 71 + 104) = 660 / 835 \approx 79.04\%$	Tốt

```

hadoopquynhthu@quynhthu: ~ 25/hadoopquynhthu@quynhthu: ~ + - ×
25/05/25 10:28:37 INFO DAGScheduler: ResultStage 57 (collect at /home/hadoopquynhthu/platin
um_classify_aqi_from_silver.py:80) finished in 0.059 s
25/05/25 10:28:37 INFO DAGScheduler: Job 31 is finished. Cancelling potential speculative o
r zombie tasks for this job
25/05/25 10:28:37 INFO TaskSchedulerImpl: Killing all running tasks in stage 57: Stage fini
shed
25/05/25 10:28:37 INFO DAGScheduler: Job 31 finished: collect at /home/hadoopquynhthu/plati
num_classify_aqi_from_silver.py:80, took 0.065381 s

Confusion Matrix - Random Forest:
Label Label Name          Predicted Prediction Name      Count
-----
0    Average             0    Average                6312
0    Average             2    Unhealthy              76
1    Good               1    Good                 2329
1    Good               2    Unhealthy              56
2    Unhealthy           0    Average                85
2    Unhealthy           2    Unhealthy              864
2    Unhealthy           3    Moderate               2
3    Moderate            0    Average                71
3    Moderate            2    Unhealthy              104
3    Moderate            3    Moderate               660

Feature Importances - Random Forest:
CO: 0.0160
NO2: 0.0061
PM25: 0.9722
Temperature: 0.0033
Humidity: 0.0025
25/05/25 10:28:37 INFO deprecation: mapred.output.dir is deprecated. Instead, use mapreduce
.output.fileoutputformat.outputdir
25/05/25 10:28:37 INFO HadoopMapRedCommitProtocol: Using output committer class org.apache.
hadoop.mapred.FileOutputCommitter
25/05/25 10:28:37 INFO FileOutputCommitter: File Output Committer Algorithm version is 1
25/05/25 10:28:37 INFO FileOutputCommitter: FileOutputCommitter skip cleanup _temporary fol
ders under output directory:false, ignore cleanup failures: false
25/05/25 10:28:37 INFO SparkContext: Starting job: runJob at SparkHadoopWriter.scala:83
25/05/25 10:28:37 INFO DAGScheduler: Got job 32 (runJob at SparkHadoopWriter.scala:83) with
1 output partitions
25/05/25 10:28:37 INFO DAGScheduler: Final stage: ResultStage 58 (runJob at SparkHadoopWrit
er.scala:83)
25/05/25 10:28:37 INFO DAGScheduler: Parents of final stage: List()
25/05/25 10:28:37 INFO DAGScheduler: Missing parents: List()
25/05/25 10:28:37 INFO DAGScheduler: Submitting ResultStage 58 (MapPartitionsRDD[150] at sa
veAsTextFile at ReadWrite.scala:413), which has no missing parents

```

**PM25 Chiếm ưu thế tuyệt đối**, là yếu tố quyết định chính trong phân loại AQI → rất hợp lý vì PM2.5 là tác nhân chính ảnh hưởng đến chất lượng không khí.

```
(first 15 tasks are for partitions Vector(0))
25/05/25 10:28:48 INFO TaskSchedulerImpl: Adding task set 74.0 with 1 tasks resource profil
e 0
25/05/25 10:28:48 INFO TaskSetManager: Starting task 0.0 in stage 74.0 (TID 667) (quynhthu-
master, executor driver, partition 0, NODE_LOCAL, 9688 bytes)
25/05/25 10:28:48 INFO Executor: Running task 0.0 in stage 74.0 (TID 667)
25/05/25 10:28:48 INFO ShuffleBlockFetcherIterator: Getting 50 (4.8 KiB) non-empty blocks i
ncluding 50 (4.8 KiB) local and 0 (0.0 B) host-local and 0 (0.0 B) push-merged-local and 0
(0.0 B) remote blocks
25/05/25 10:28:48 INFO ShuffleBlockFetcherIterator: Started 0 remote fetches in 0 ms
25/05/25 10:28:48 INFO CodeGenerator: Code generated in 11.974223 ms
25/05/25 10:28:48 INFO CodeGenerator: Code generated in 13.161362 ms
25/05/25 10:28:48 INFO Executor: Finished task 0.0 in stage 74.0 (TID 667). 7038 bytes resu
lt sent to driver
25/05/25 10:28:48 INFO TaskSetManager: Finished task 0.0 in stage 74.0 (TID 667) in 90 ms o
n quynhthu-master (executor driver) (1/1)
25/05/25 10:28:48 INFO TaskSchedulerImpl: Removed TaskSet 74.0, whose tasks have all comple
ted, from pool
25/05/25 10:28:48 INFO DAGScheduler: ResultStage 74 ($anonfun$recordDeltaOperationInternal$1
at DatabricksLogging.scala:128) finished in 0.102 s
25/05/25 10:28:48 INFO DAGScheduler: Job 43 is finished. Cancelling potential speculative o
r zombie tasks for this job
25/05/25 10:28:48 INFO TaskSchedulerImpl: Killing all running tasks in stage 74: Stage fini
shed
25/05/25 10:28:48 INFO DAGScheduler: Job 43 finished: $anonfun$recordDeltaOperationInternal
$1 at DatabricksLogging.scala:128, took 0.105510 s
25/05/25 10:28:48 INFO CodeGenerator: Code generated in 22.016956 ms
25/05/25 10:28:49 INFO Snapshot: [tableId=a330766f-84b7-46fa-9d1a-fce1e5ad8f2c] DELTA: Done
25/05/25 10:28:49 INFO OptimisticTransaction: [tableId=b2127342, txId=52a95acb] Committed d
elta #0 to hdfs://quynhthu-master:9000/lakehouse/platinum/air_quality_classified/air_qualit
y_predictions_rf_from_silver/_delta_log
25/05/25 10:28:49 INFO SparkContext: SparkContext is stopping with exitCode 0.
25/05/25 10:28:49 INFO SparkUI: Stopped Spark web UI at http://quynhthu-master:4041
25/05/25 10:28:49 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopp
ed!
25/05/25 10:28:49 INFO MemoryStore: MemoryStore cleared
25/05/25 10:28:49 INFO BlockManager: BlockManager stopped
25/05/25 10:28:49 INFO BlockManagerMaster: BlockManagerMaster stopped
25/05/25 10:28:49 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommi
tCoordinator stopped!
25/05/25 10:28:49 INFO SparkContext: Successfully stopped SparkContext
25/05/25 10:28:49 INFO ShutdownHookManager: Shutdown hook called
25/05/25 10:28:49 INFO ShutdownHookManager: Deleting directory /tmp/spark-de819abd-6f63-49e
2-a6d7-45d307pd7412
25/05/25 10:28:49 INFO ShutdownHookManager: Deleting directory /tmp/spark-106a08af-d18c-45e
0-bb17-49f4be30aece
25/05/25 10:28:49 INFO ShutdownHookManager: Deleting directory /tmp/spark-106a08af-d18c-45e
0-bb17-49f4be30aece/pyspark-9664d72a-70d4-4933-b013-0db68b6a2de4
hadoopquynhthu@quynhthu-master:~$ |
```

## Kiểm tra thư mục kết quả

```
>> df_result =
spark.read.format("delta").load("/lakehouse/platinum/air_quality_classified/air_quali
ty_predictions_rf_from_silver")
>> df_result.show(5)
```

```
>>> df_result = spark.read.format("delta").load("/lakehouse/platinum/air_quality_classified/air_quality_predictions_rf_from_silver")
>>> df_result.show(5)
[Stage 40:=====] (8 + 2) /
[Stage 40:=====] (11 + 2) /
[Stage 40:=====] (15 + 2) /
[Stage 40:=====] (18 + 2) /
[Stage 40:=====] (22 + 2) /
[Stage 40:=====] (26 + 2) /
[Stage 40:=====] (29 + 2) /
[Stage 40:=====] (33 + 2) /
[Stage 40:=====] (37 + 2) /
[Stage 40:=====] (41 + 2) /
[Stage 40:=====] (45 + 2) /
+---+---+---+---+---+---+
| CO|NO2|PM25|Temperature|Humidity|AQI_Level|label|prediction|
+---+---+---+---+---+---+
| 0.0|0.0|14.29666667| 25.11|98.48666667| Average| 0.0| 0.0|
| 0.0|0.0|15.49333333| 31.165| 72.89| Average| 0.0| 0.0|
| 0.0|0.0|18.33275862|25.08103448|98.20517241| Average| 0.0| 0.0|
| 0.0|0.0| 18.415|31.14833333| 71.32833333| Average| 0.0| 0.0|
| 0.0|0.0|22.84333333| 24.94|98.60833333| Average| 0.0| 0.0|
+---+---+---+---+---+---+
only showing top 5 rows

>>> |
```

## Xuất thành file CSV cho Power BI

```
df_result.coalesce(1) \
    .write.option("header", True) \
    .mode("overwrite") \
    .csv("/lakehouse/platinum/air_quality_classified/predictions_csv_single")
```

#### Tìm và đổi tên file kết quả

```
$ hdfs dfs -ls /lakehouse/platinum/air_quality_classified/predictions_csv_single
$ hdfs dfs -mv
/lakehouse/platinum/air_quality_classified/predictions_csv_single/part-*.csv
/lakehouse/platinum/air quality classified/predictions csv single/predictions.csv
```

```
hadoopquynhthu@quynhthu:~$ hdfs dfs -ls /lakehouse/platinum/air_quality_classified/predictions_csv_single
Found 2 items
-rw-r--r-- 2 hadoopquynhthu supergroup          0 2025-05-25 10:32 /lakehouse/platinum/air_quality_classified/predictions_csv_single/_SUCCESS
-rw-r--r-- 2 hadoopquynhthu supergroup    715015 2025-05-25 10:32 /lakehouse/platinum/air_quality_classified/predictions_csv_single/part-00000-be6be873-66fa-41cd-b3c5-6f4b0aa33ab9-c000.csv
hadoopquynhthu@quynhthu:~$ hdfs dfs -mv /lakehouse/platinum/air_quality_classified/predictions_csv_single/part-*.csv /lakehouse/platinum/air_quality_classified/predictions_csv_single/predictions.csv
hadoopquynhthu@quynhthu:~$ |
```

```
$ hdfs dfs -get
/lakehouse/platinum/air_quality_classified/predictions_csv_single/predictions.csv
~/air_quality_export/predictions.csv
```

```
hadoopquynhthu@quynhthu:~$ hdfs dfs -get /lakehouse/platinum/air_quality_classified/predictions_csv_single/predictions.csv ~/air_quality_export/predictions.csv
hadoopquynhthu@quynhthu:~$ |
```

Tải về máy local

```
> scp hadoopquynhthu@192.168.40.11:~/air_quality_export/*.csv
C:\Users\Admin\Downloads\PhanTichDuLieuLons
```

```
Command Prompt
C:\Users\Admin>scp hadoopquynhthu@192.168.40.11:~/air_quality_export/*.csv C:\Users\Admin\Downloads\PhanTichDuLieuLon
hadoopquynhthu@192.168.40.11's password: predictions.csv
100%   698KB  12.4MB/s  00:00
C:\Users\Admin>
```

## 5. Phát triển mô hình dự báo SARIMA từ dữ liệu lớp Gold

Vì Spark không hỗ trợ mô hình ARIMA/SARIMA một cách trực tiếp, nên sử dụng Jupyter Notebook với Python để xây dựng mô hình dự báo. Dữ liệu được đọc từ lớp Gold trong kiến trúc Lakehouse, sử dụng định dạng Delta Lake để đảm bảo tính linh hoạt và hiệu quả khi truy xuất dữ liệu

Cài đặt jupyter:

```
$ pip install jupyter
```

```
(myenv) hadoophongtho@hongtho-master:~/project$ pip install jupyter
Collecting jupyter
  Downloading jupyter-1.1.1-py2.py3-none-any.whl.metadata (2.0 kB)
Collecting notebook (from jupyter)
  Downloading notebook-7.4.2-py3-none-any.whl.metadata (10 kB)
Collecting jupyter-console (from jupyter)
  Downloading jupyter_console-6.6.3-py3-none-any.whl.metadata (5.8 kB)
Collecting nbconvert (from jupyter)
  Downloading nbconvert-7.16.6-py3-none-any.whl.metadata (8.5 kB)
Collecting ipykernel (from jupyter)
  Downloading ipykernel-6.29.5-py3-none-any.whl.metadata (6.3 kB)
Collecting ipywidgets (from jupyter)
  Downloading ipywidgets-8.1.7-py3-none-any.whl.metadata (2.4 kB)
Collecting jupyterlab (from jupyter)
  Downloading jupyterlab-4.4.2-py3-none-any.whl.metadata (16 kB)
Collecting comm>=0.1.1 (from ipykernel->jupyter)
  Downloading comm-0.2.2-py3-none-any.whl.metadata (3.7 kB)
Collecting debugpy>=1.6.5 (from ipykernel->jupyter)
  Downloading debugpy-1.8.14-cp312-cp312-manylinux_2_5_x86_64.manylinux1_x86_
_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (1.3 kB)
Collecting ipython>=7.23.1 (from ipykernel->jupyter)
  Downloading ipython-9.2.0-py3-none-any.whl.metadata (4.4 kB)
Collecting jupyter-client>=6.1.12 (from ipykernel->jupyter)
  Downloading jupyter_client-8.6.3-py3-none-any.whl.metadata (8.3 kB)
Collecting jupyter-core!=5.0.*,>=4.12 (from ipykernel->jupyter)
  Downloading jupyter_core-5.7.2-py3-none-any.whl.metadata (3.4 kB)
Collecting matplotlib-inline>=0.1 (from ipykernel->jupyter)
  Downloading matplotlib_inline-0.1.7-py3-none-any.whl.metadata (3.9 kB)
Collecting nest-asyncio (from ipykernel->jupyter)
  Downloading nest_asyncio-1.6.0-py3-none-any.whl.metadata (2.8 kB)
Collecting packaging (from ipykernel->jupyter)
  Downloading packaging-25.0-py3-none-any.whl.metadata (3.3 kB)
Collecting psutil (from ipykernel->jupyter)
  Downloading psutil-7.0.0-cp36-abi3-manylinux_2_12_x86_64.manylinux2010_x86_
_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (22 kB)
Collecting pyzmq>=24 (from ipykernel->jupyter)
  Downloading pyzmq-26.4.0-cp312-cp312-manylinux_2_28_x86_64.whl.metadata (6
.0 kB)
```

```
$ jupyter-notebook --version
```

```
(myenv) hadoophongtho@hongtho-master:~/project$ jupyter-notebook --version
7.4.2
```

Cáu hình môi trường để truy cập từ tất cả IP:

```
$ jupyter notebook --generate-config
```

```
(myenv) hadoophongtho@hongtho-master:~/project$ jupyter notebook --generate-
config
Writing default config to: /home/hadoophongtho/.jupyter/jupyter_notebook_con-
fig.py
```

```
$ jupyter notebook password
```

```
(myenv) hadoophongtho@hongtho-master:~/project$ jupyter notebook password
Enter password:
Verify password:
[JupyterPasswordApp] Wrote hashed password to /home/hadoophongtho/.jupyter/jupyter_server_config.json
```

```
$ vim ~/.jupyter/jupyter_notebook_config.py
```

```
#                         default browser will be determined by the 'webbrowser'
#                         standard library module, which allows setting of the
#                         BROWSER environment variable to override it.
# Default: ''
# c.ServerApp.browser = ''

## The full path to an SSL/TLS certificate file.
# Default: ''
# c.ServerApp.certfile = ''

## The full path to a certificate authority certificate for SSL/TLS client
# authentication.
# Default: ''
# c.ServerApp.client_ca = ''

## Full path of a config file.
# See also: JupyterApp.config_file
# c.ServerApp.config_file = ''
vim ~/.jupyter/jupyter_notebook_config.py
## Specify a config file to load.
# See also: JupyterApp.config_file_name
# c.ServerApp.config_file_name = ''

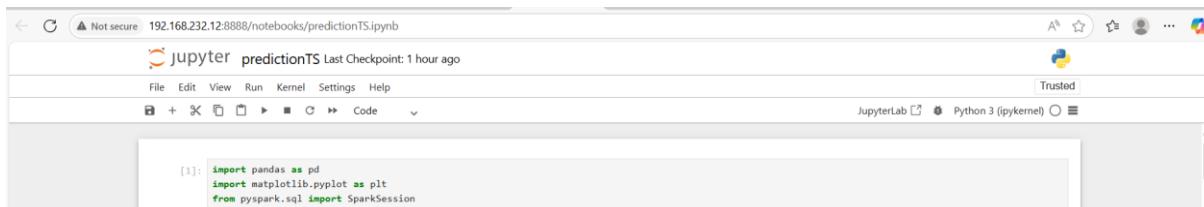
## The config manager class to use
# Default: 'jupyter_server.services.config_manager.ConfigManager'
# c.ServerApp.config_manager_class = 'jupyter_server.services.config_manager.ConfigManager'
c.NotebookApp.ip = '0.0.0.0' # Cho phép kết nối từ bất kỳ IP nào
c.NotebookApp.port = 8888 # Port mặc định
c.NotebookApp.open_browser = False
```

```
$ jupyter notebook --no-browser
```

```
(myenv) hadoophongtho@hongtho-master:~/project$ jupyter notebook --no-browser
[I 2025-05-25 15:39:51.032 ServerApp] jupyter_lsp | extension was successfully linked.
[I 2025-05-25 15:39:51.036 ServerApp] jupyter_server_terminals | extension was successfully linked.
[I 2025-05-25 15:39:51.040 ServerApp] jupyterlab | extension was successfully linked.
[W 2025-05-25 15:39:51.043 JupyterNotebookApp] 'ip' has moved from NotebookApp to ServerApp. This config will be passed to ServerApp. Be sure to update your config before our next release.
[W 2025-05-25 15:39:51.043 JupyterNotebookApp] 'port' has moved from NotebookApp to ServerApp. This config will be passed to ServerApp. Be sure to update your config before our next release.
[W 2025-05-25 15:39:51.043 JupyterNotebookApp] 'port' has moved from NotebookApp to ServerApp. This config will be passed to ServerApp. Be sure to update your config before our next release.
[I 2025-05-25 15:39:51.046 ServerApp] notebook | extension was successfully linked.
```

```
[I 2025-05-25 15:39:51.282 ServerApp] http://hongtho-master:8888/tree
[I 2025-05-25 15:39:51.282 ServerApp] http://127.0.0.1:8888/tree
[I 2025-05-25 15:39:51.282 ServerApp] use Control-C to stop this server and
shut down all kernels (twice to skip confirmation).
[I 2025-05-25 15:39:51.299 ServerApp] Skipped non-installed server(s): bash-
language-server, dockerfile-language-server-nodejs, javascript-typescript-la-
ngserver, jedi-language-server, julia-language-server, pyright, python-langu-
age-server, python-lsp-server, r-languageserver, sql-language-server, texlab
, typescript-language-server, unified-language-server, vscode-css-languagese-
rver-bin, vscode-html-languageserver-bin, vscode-json-languageserver-bin, ya-
ml-language-server
[I 2025-05-25 15:40:25.893 JupyterNotebookApp] 302 GET /tree (@192.168.232.1
) 0.64ms
[I 2025-05-25 15:40:29.188 ServerApp] User 1f8d56410f82468e8a4bf1f41d6274ea
logged in.
[I 2025-05-25 15:40:29.189 ServerApp] 302 POST /login?next=%2Ftree (1f8d5641
0f82468e8a4bf1f41d6274ea@192.168.232.1) 63.05ms
[I 2025-05-25 15:40:57.712 ServerApp] Creating new notebook in
[I 2025-05-25 15:40:57.754 ServerApp] Writing notebook-signing key to /home/
hadoophongtho/.local/share/jupyter/notebook_secret
[I 2025-05-25 15:41:03.074 ServerApp] Kernel started: 54277f5b-2d26-4cce-b64
2-bbfffca1417e3
[I 2025-05-25 15:41:03.576 ServerApp] Connecting to kernel 54277f5b-2d26-4cc
e-b642-bbfffca1417e3.
[W 2025-05-25 15:41:03.578 ServerApp] The websocket_ping_timeout (90000) can
not be longer than the websocket_ping_interval (30000).
Setting websocket_ping_timeout=30000
[I 2025-05-25 15:41:03.593 ServerApp] Starting buffering for 54277f5b-2d26-4
```

Truy cập địa chỉ để mở Jupyter trên Browser Windows:



Cài đặt spark delta để lấy data từ Layer Gold (cài đặt bản 3.3.0 phù hợp với spark 3.5.3):

```
$ pip install delta-spark==3.3.0
```

```
(myenv) hadoophongtho@hongtho-master:~$ pip install delta-spark==3.3.0
Collecting delta-spark==3.3.0
  Downloading delta_spark-3.3.0-py3-none-any.whl.metadata (2.0 kB)
Requirement already satisfied: pyspark<3.6.0,>=3.5.3 in ./spark/python (from delta-spark==3.3.0) (3.5.3)
Requirement already satisfied: importlib-metadata>=1.0.0 in ./myenv/lib/python3.12/site-packages (from delta-spark==3.3.0) (0.7.0)
Requirement already satisfied: zipp>=3.2.0 in ./myenv/lib/python3.12/site-packages (from importlib-metadata>=1.0.0->delta-spark==3.3.0) (3.21.0)
Requirement already satisfied: py4j==0.10.9.7 in ./myenv/lib/python3.12/site-packages (from pyspark<3.6.0,>=3.5.3->delta-spark==3.3.0) (0.10.9.7)
Downloading delta_spark-3.3.0-py3-none-any.whl (21 kB)
Installing collected packages: delta-spark
  Attempting uninstall: delta-spark
    Found existing installation: delta-spark 3.3.1
    Uninstalling delta-spark-3.3.1:
      Successfully uninstalled delta-spark-3.3.1
Successfully installed delta-spark-3.3.0
```

Tạo file predictionTS.py để xây dựng mô hình ARIMA dự đoán chỉ số trung bình của PM 2.5 theo chuỗi thời gian:

```
[1]: import pandas as pd
import matplotlib.pyplot as plt
from pyspark.sql import SparkSession

[2]: pip show delta-spark
```

Name: delta-spark  
Version: 3.3.0  
Summary: Python APIs for using Delta Lake with Apache Spark  
Home-page: <https://github.com/delta-io/delta/>  
Author: The Delta Lake Project Authors  
Author-email: delta-users@googlegroups.com  
License: Apache-2.0  
Location: /home/hadoophongtho/myenv/lib/python3.12/site-packages  
Requires: importlib-metadata, pyspark  
Required-by:  
Note: you may need to restart the kernel to use updated packages.

```
[3]: import os
import sys
import pyspark
from delta import *

# Thiết lập các biến môi trường
os.environ["SPARK_HOME"] = "/home/hadoophongtho/spark"
os.environ["HADOOP_CONF_DIR"] = "/etc/hadoop/conf"
```

Chỉ định đường dẫn cài đặt Apache Spark để Python/PySpark biết spark ở đâu:

```
os.environ["SPARK_HOME"] = "/home/hadoophongtho/spark"
os.environ["HADOOP_CONF_DIR"] = "/etc/hadoop/conf"
```

Tạo một Spark App theo hướng dẫn trên:

*Delta Lake, “Quickstart: Set up project”, Delta Lake Documentation, Version 3.3.1. [Online]. Available: <https://docs.delta.io/latest/quick-start.html#set-up-project>*

#### Python

To set up a Python project (for example, for unit testing), you can install Delta Lake using `pip install delta-spark==3.3.1` and then configure the `SparkSession` with the `configure_spark_with_delta_pip()` utility function in Delta Lake.

```
Python
Copy

import pyspark
from delta import *

builder = pyspark.sql.SparkSession.builder.appName("MyApp") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog", "org.apache.spark.sql.delta.catalog.DeltaCatalog")

spark = configure_spark_with_delta_pip(builder).getOrCreate()
```

```
from delta import configure_spark_with_delta_pip

builder = pyspark.sql.SparkSession.builder.appName("MyApp") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog",
"org.apache.spark.sql.delta.catalog.DeltaCatalog")

spark = configure_spark_with_delta_pip(builder).getOrCreate()
```

```
[4]: import pyspark
from delta import *
from delta import configure_spark_with_delta_pip

builder = pyspark.sql.SparkSession.builder.appName("MyApp") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog", "org.apache.spark.sql.delta.catalog.DeltaCatalog")

spark = configure_spark_with_delta_pip(builder).getOrCreate()

:: loading settings :: url = jar:file:/home/hadoophongtho/spark/jars/ivy-2.5.1.jar!/org/apache/ivy/core/settings/ivysettings.xml
Ivy Default Cache set to: /home/hadoophongtho/.ivy2/cache
The jars for the packages stored in: /home/hadoophongtho/.ivy2/jars
io.delta#delta-spark_2.12 added as a dependency
:: resolving dependencies :: org.apache.spark#spark-submit-parent-813f131e-c4fa-4fca-81e4-e4f0d0bf9fc9;1.0
  confs: [default]
    found io.delta#delta-spark_2.12;3.3.0 in central
    found io.delta#delta-storage;3.3.0 in central
    found org.antlr#antlr4-runtime;4.9.3 in local-m2-cache
:: resolution report :: resolve 149ms :: artifacts dl 6ms
  :: modules in use:
    io.delta#delta-spark_2.12;3.3.0 from central in [default]
    io.delta#delta-storage;3.3.0 from central in [default]
    org.antlr#antlr4-runtime;4.9.3 from local-m2-cache in [default]
    -----
    |           |         modules      ||   artifacts  |
    |     conf    | number| search|dwnlded|evicted|| number|dwnlded|
    -----
    |     default  |   3  |  0  |   0  |   0  ||   3  |  0  |
    -----
:: retrieving :: org.apache.spark#spark-submit-parent-813f131e-c4fa-4fca-81e4-e4f0d0bf9fc9
  confs: [default]
  0 artifacts copied, 3 already retrieved (0kB/4ms)
25/05/25 19:28:58 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-j
ava classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
25/05/25 19:28:59 WARN Utils: Service 'SparkUI' could not bind on port 4040. Attempting port 4041.
25/05/25 19:28:59 WARN Utils: Service 'SparkUI' could not bind on port 4041. Attempting port 4042.
```

Load data từ lớp Gold của Lakehouse:

```
df_spark =
spark.read.format("delta").load("hdfs://192.168.232.12:9000/lakehouse/gold/air_quality_daily_avg")
df = df_spark.toPandas()
```

```
[5]: df_spark = spark.read.format("delta").load("hdfs://192.168.232.12:9000/lakehouse/gold/air_quality_daily_avg")
df = df_spark.toPandas()
```

```
25/05/25 19:29:07 WARN SparkStringUtils: Truncated the string representation of a plan since it was too large. Thi
s behavior can be adjusted by setting 'spark.sql.debug.maxToStringFields'.
```

Set cột thời gian làm index để chuyển thành chuỗi thời gian:

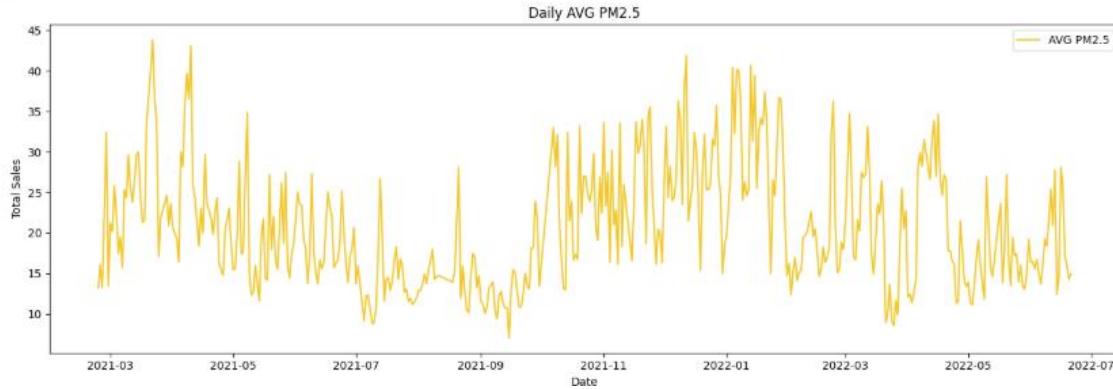
```
[6]: df['date_only'] = pd.to_datetime(df['date_only'])
df = df.sort_values('date_only')
df = df.set_index('date_only')
```

```
[7]: print(df)
```

	avg_PM25	avg_temp	avg_humidity	AQI_Level
date_only				
2021-02-23	13.235970	28.201846	64.528641	Average
2021-02-24	16.092616	27.516220	64.908390	Average
2021-02-25	13.298236	27.910777	66.225786	Average
2021-02-26	22.128216	27.740009	69.505777	Average
2021-02-27	32.433861	29.187368	69.267263	Average
...	...	...	...	...
2022-06-17	25.697171	28.910312	62.900108	Average
2022-06-18	17.123734	28.753054	63.558921	Average
2022-06-19	15.915215	27.900984	67.331003	Average
2022-06-20	14.278092	27.652727	67.054576	Average
2022-06-21	14.997616	27.911035	63.925762	Average

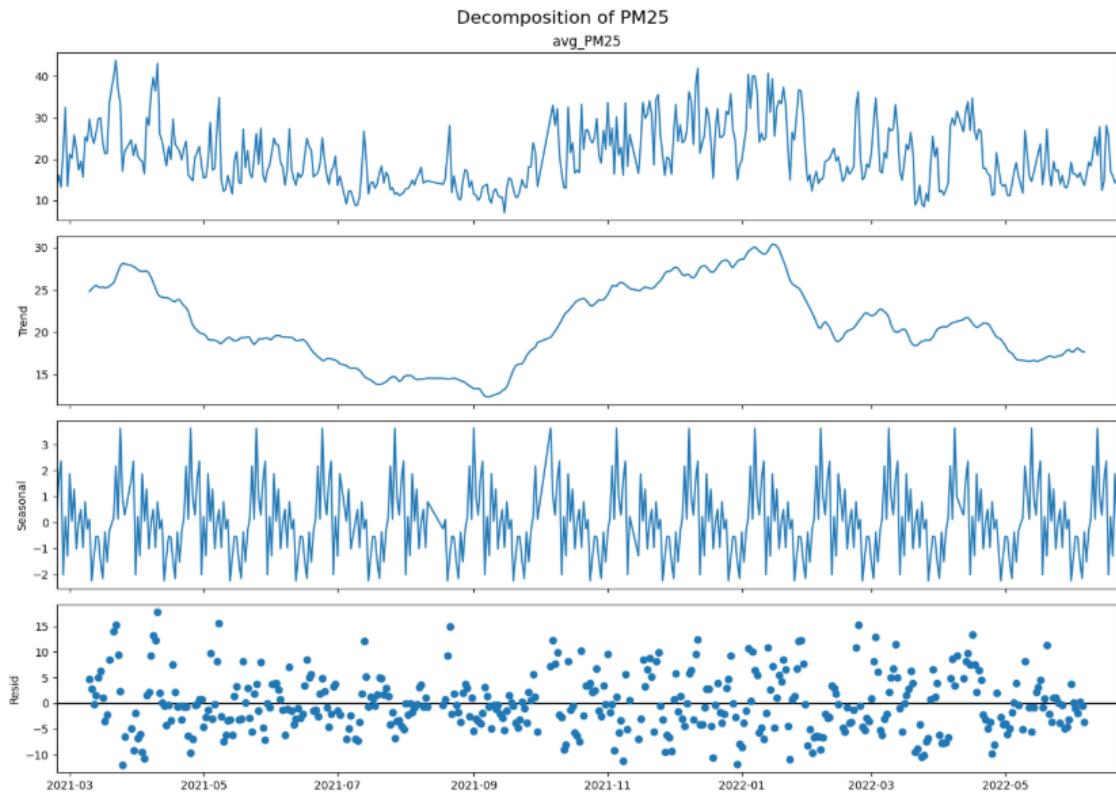
[458 rows x 4 columns]

```
[20]: plt.figure(figsize=(14, 5))
plt.plot(df['avg_PM25'], label='AVG PM2.5', color='#F3C623')
plt.title('Daily AVG PM2.5')
plt.xlabel('Date')
plt.ylabel('Total Sales')
plt.legend()
plt.grid(False)
plt.tight_layout()
plt.show()
```



Phân rã dữ liệu để xem xét xu hướng, chu kỳ, mùa vụ:

```
[21]: from statsmodels.tsa.seasonal import seasonal_decompose
# Phân tích thành phần chuỗi thời gian theo ngày (Seasonal Decompose)
# model='additive': Điều này có nghĩa là đang thực hiện phân rã theo kiểu cộng (additive decomposition)
# period=30: Giả định rằng có một chu kỳ hàng tháng trong dữ liệu (30 ngày)
decomposition = seasonal_decompose(df['avg_PM25'], model='additive', period=30)
fig = decomposition.plot()
fig.set_size_inches(14, 10)
plt.suptitle('Decomposition of PM25', fontsize=16)
plt.tight_layout()
plt.show()
```



Trung bình trượt để xem rõ xu hướng của dữ liệu:

```
[22]: # Trực quan hóa riêng cho trung bình động của dữ liệu gốc
plt.figure(figsize=(14, 6))

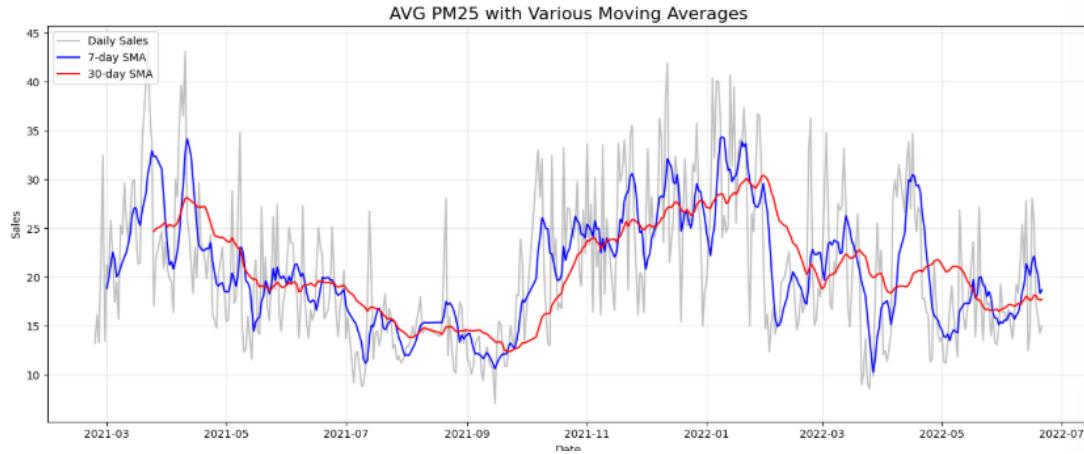
# Vẽ dữ liệu gốc
plt.plot(df['avg_PM25'], label='Daily Sales', color='gray', alpha=0.5)

# Tính và vẽ các loại trung bình động
# 1. Simple Moving Average (SMA)
sma_7 = df['avg_PM25'].rolling(window=7, center=False).mean()
sma_30 = df['avg_PM25'].rolling(window=30, center=False).mean()

# 2. Exponential Moving Average (EMA)
ema_7 = df['avg_PM25'].ewm(span=7, adjust=False).mean()
ema_30 = df['avg_PM25'].ewm(span=30, adjust=False).mean()

# Vẽ các đường trung bình động
plt.plot(sma_7, label='7-day SMA', color='blue', linewidth=1.5)
plt.plot(sma_30, label='30-day SMA', color='red', linewidth=1.5)

plt.title('AVG PM25 with Various Moving Averages', fontsize=16)
plt.xlabel('Date')
plt.ylabel('Sales')
plt.grid(True, alpha=0.3)
plt.legend(loc='upper left')
plt.tight_layout()
plt.show()
```



→ Chỉ số PM 2.5 không có xu hướng rõ ràng, nhưng có thể có tính mùa vụ, chu kỳ.

Kiểm tra tính dừng của chuỗi:

```
def check_stationarity(series):
    result = adfuller(series.dropna())
    print(f'ADF Statistic: {result[0]}')
    print(f'p-value: {result[1]}')
    if result[1] < 0.05:
        print("Chuỗi dữ liệu đã dừng (stationary)")
    else:
        print("Chuỗi dữ liệu chưa dừng (non-stationary)")

# Kiểm tra tính dừng của PM2.5
check_stationarity(df['avg_PM25'])
```

ADF Statistic: -3.6212138092152006

p-value: 0.005368178879456069

Chuỗi dữ liệu đã dừng (stationary)

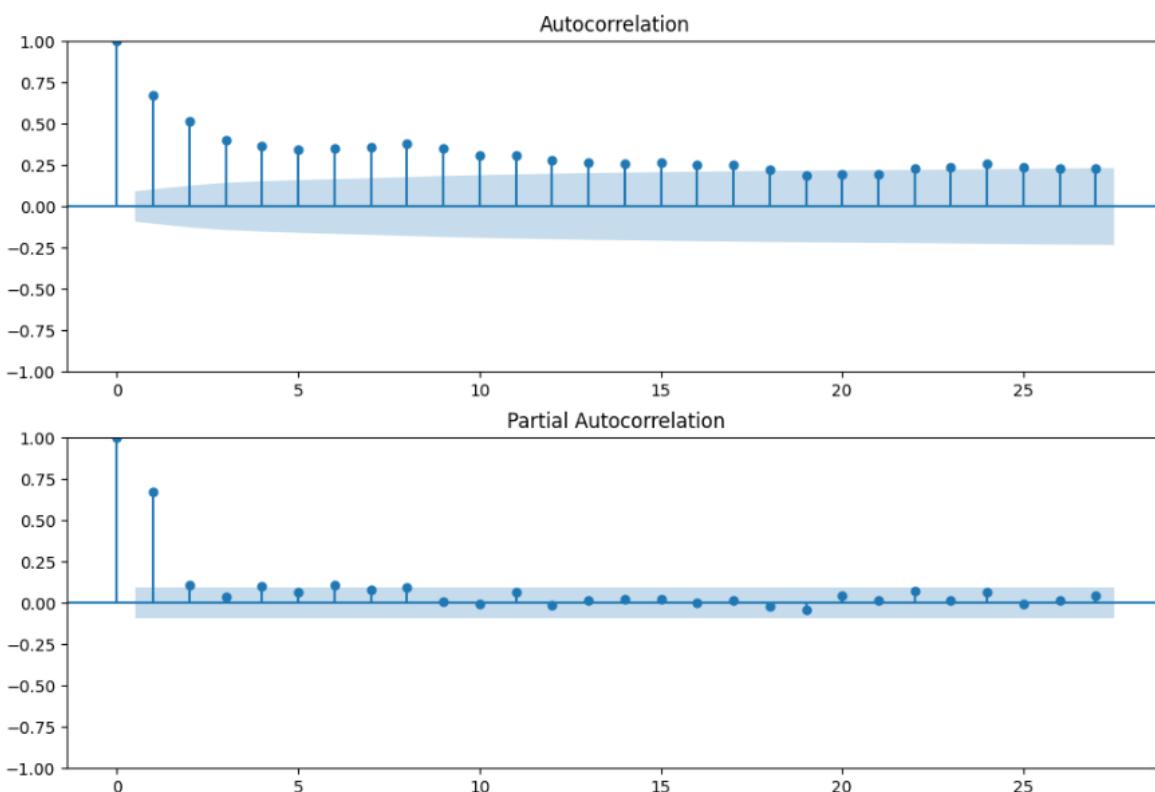
→ Chuỗi dữ liệu đã dừng. Có chu kỳ nhưng không có xu hướng và mùa vụ.

Trực quan biểu đồ ACF và PACF để xem bậc của p, q trong mô hình:

```

fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 8))
plot_acf(df['avg_PM25'].dropna(), ax=ax1)
plot_pacf(df['avg_PM25'].dropna(), ax=ax2)
fig.show()

```



Biểu đồ ACF cho thấy tại độ trễ 1 có tương quan cao nhất  $\Rightarrow q = 1$ .

Biểu đồ PACF cho thấy tại độ trễ 1 có tương quan cao nhất  $\Rightarrow p = 1$ .

Vậy mô hình ARIMA(1,0,1).

Xây dựng mô hình ARIMA dự đoán giá trị PM 2.5 theo thời gian:

```

# Tạo trực thời gian
if isinstance(df_PM25.index, pd.DatetimeIndex):
    x_train = train_data.index
    x_test = test_data.index
else:
    x_train = range(len(train_data))
    x_test = range(len(train_data), len(df_PM25))
# Tạo mô hình ARIMA với các tham số (p, d, q)
model_arima = ARIMA(y_train, order=(1,0,1))
# Huấn luyện mô hình
results_arima = model_arima.fit()
# Dự đoán trên tập kiểm tra
y_pred_arima = results_arima.forecast(steps=len(y_test))
# Đánh giá mô hình

```

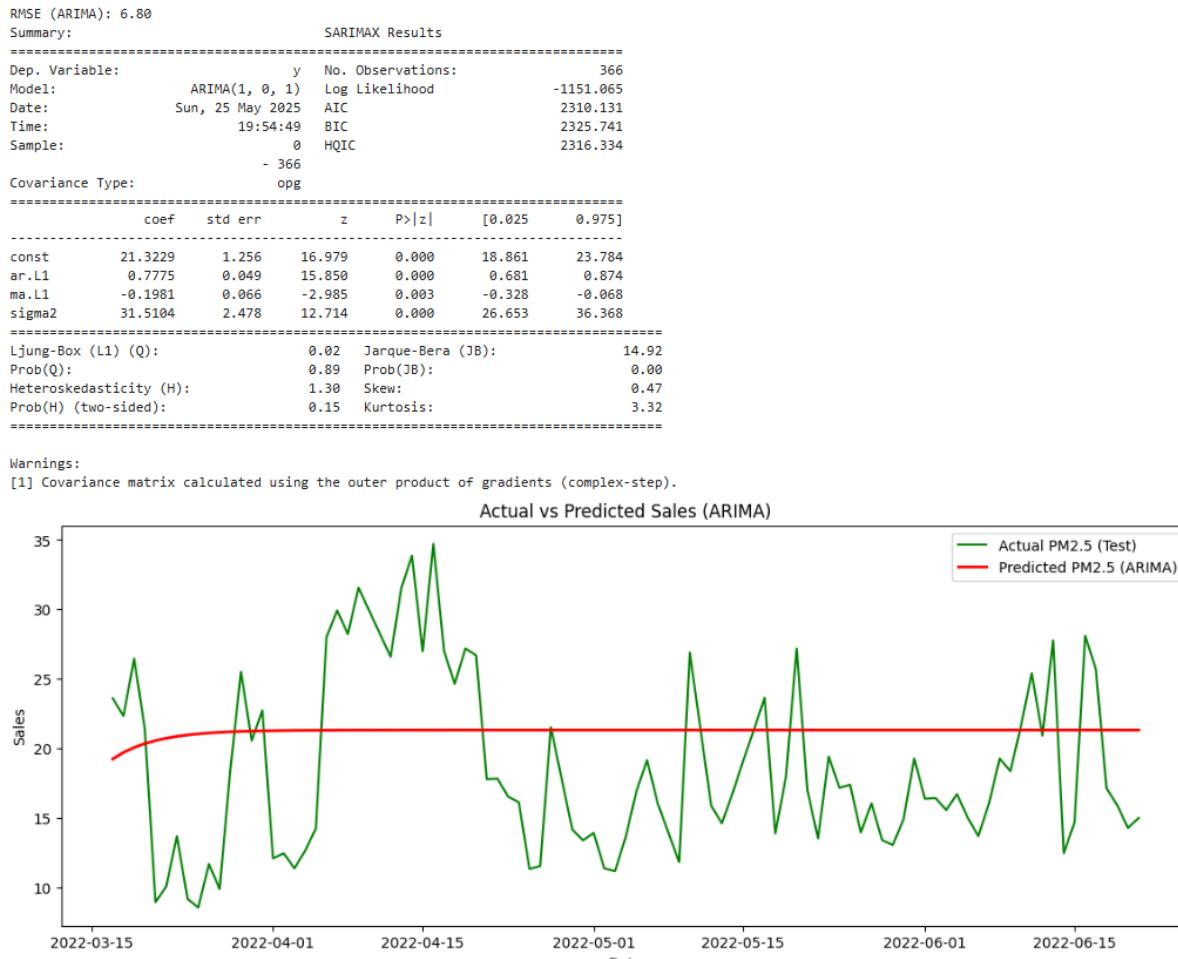
```

mse_arima = mean_squared_error(y_test, y_pred_arima)
rmse_arima = sqrt(mse_arima)
print(f"RMSE (ARIMA): {rmse_arima:.2f}")
print(f"Summary:", results_arima.summary())
# Vẽ biểu đồ so sánh thực tế và dự đoán
plt.figure(figsize=(14, 5))

plt.plot(x_test, y_test, label='Actual PM2.5 (Test)', color='green')
plt.plot(x_test, y_pred_arima, label='Predicted PM2.5 (ARIMA)', color='red',
linewidth=2)
plt.title('Actual vs Predicted Sales (ARIMA)')
plt.xlabel('Date')
plt.ylabel('Sales')
plt.legend()
plt.show()

```

Kết quả:

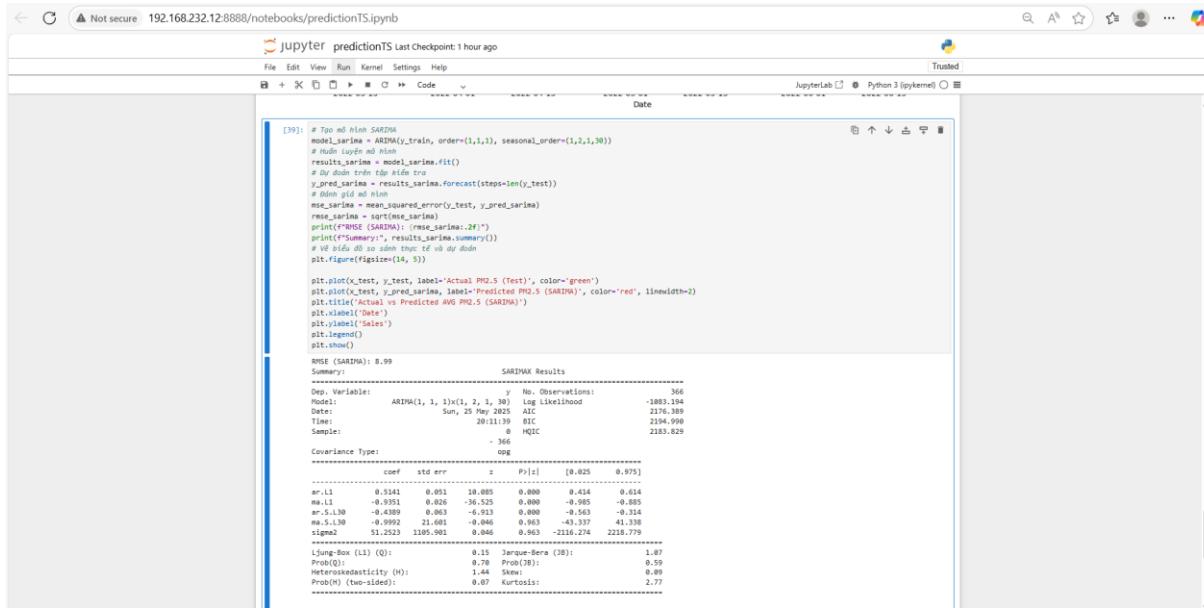


Sai số RMSE = 6.8 còn lớn, trực quan biểu đồ cho thấy mô hình chưa phát hiện được các biến động trong chuỗi thời gian. Cần cải thiện, vì PM2.5 thường có mùa vụ khoảng 6 hoặc 12 tháng, nếu chỉ có 16 tháng thì mô hình SARIMA có thể không học được tốt chu kỳ đó.

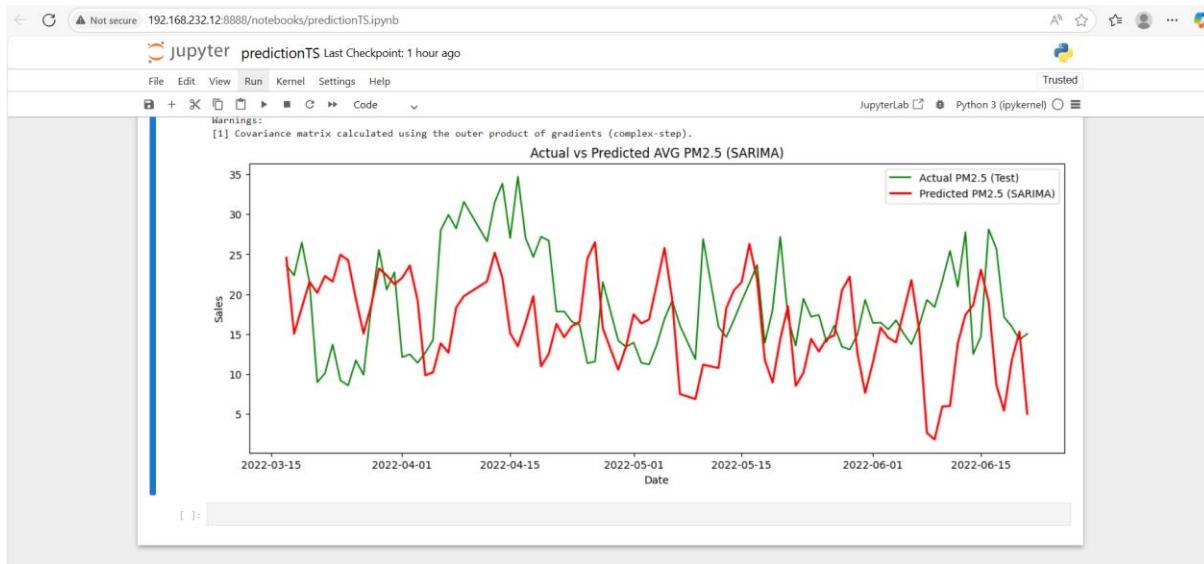
Cải thiện mô hình:

```
# Tạo mô hình SARIMA
model_sarima = ARIMA(y_train, order=(1,1,1), seasonal_order=(1,2,1,30))
# Huấn luyện mô hình
results_sarima = model_sarima.fit()
# Dự đoán trên tập kiểm tra
y_pred_sarima = results_sarima.forecast(steps=len(y_test))
# Đánh giá mô hình
mse_sarima = mean_squared_error(y_test, y_pred_sarima)
rmse_sarima = sqrt(mse_sarima)
print(f"RMSE (ARIMA): {rmse_sarima:.2f}")
print(f"Summary:", results_sarima.summary())
# Vẽ biểu đồ so sánh thực tế và dự đoán
plt.figure(figsize=(14, 5))

plt.plot(x_test, y_test, label='Actual PM2.5 (Test)', color='green')
plt.plot(x_test, y_pred_sarima, label='Predicted PM2.5 (SARIMA)', color='red',
         linewidth=2)
plt.title('Actual vs Predicted AVG PM2.5 (SARIMA)')
plt.xlabel('Date')
plt.ylabel('Sales')
plt.legend()
plt.show()
```



Biểu đồ:



SARIMA cho sai số RMSE (SARIMA): 8.99 cao hơn sao với ARIMA, nhưng AIC (2176) và BIC (2194) thấp hơn ARIMA (AIC: 2310, BIC: 2316).

→ ARIMA dự đoán ngắn hạn tốt hơn, nhưng SARIMA tổng quát hóa và khớp mô hình tốt hơn.

Dự đoán 3 tháng tiếp theo:

```
# Tạo trục thời gian
if isinstance(df_PM25.index, pd.DatetimeIndex):
    x_train = train_data.index
    x_test = test_data.index
    # Tạo trục thời gian cho 6 tháng tới
    future_dates = pd.date_range(start=x_test[-1], periods=90, freq='D')
else:
    x_train = range(len(train_data))
    x_test = range(len(train_data), len(df_PM25))
    # Tạo trục thời gian
    future_dates = range(len(df_PM25), len(df_PM25) + 90)

# Dự đoán
y_future_pred = results_sarima.forecast(steps=90)

# Vẽ biểu đồ so sánh thực tế, dự đoán và dự đoán tương lai
plt.figure(figsize=(14, 5))

# Dữ liệu thực tế
plt.plot(x_test, y_test, label='Actual PM2.5 (Test)', color='green')

# Dự đoán trên tập kiểm tra
plt.plot(x_test, y_pred_sarima, label='Predicted PM2.5 (SARIMA)', color='red',
         linewidth=2)

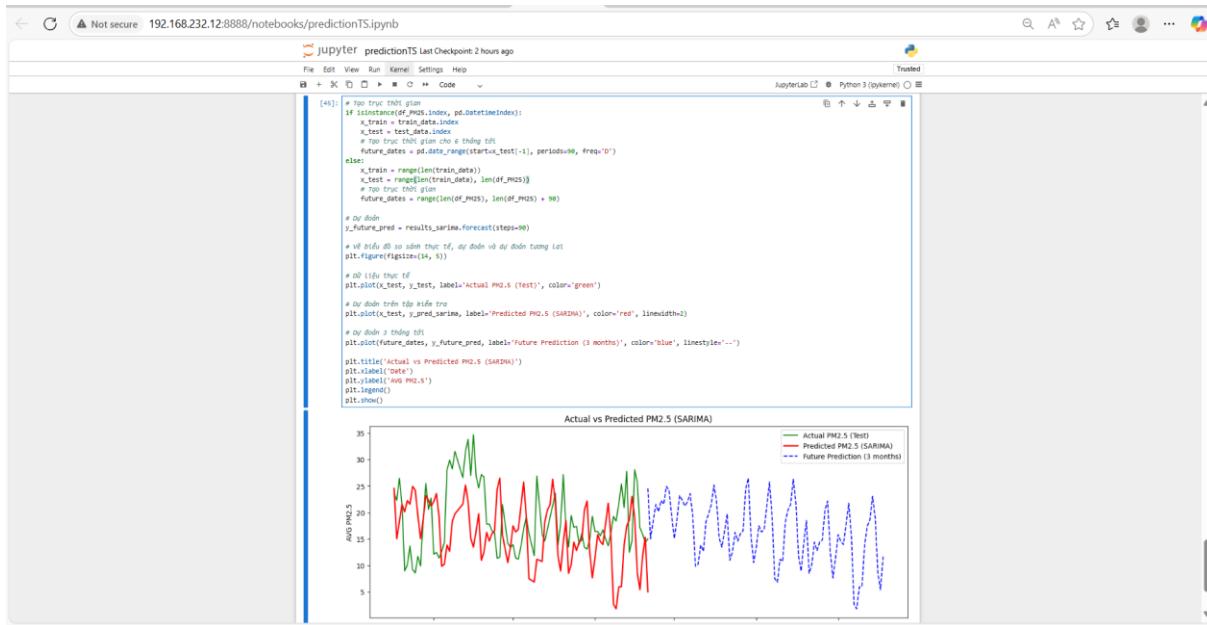
# Dự đoán 3 tháng tới
```

```

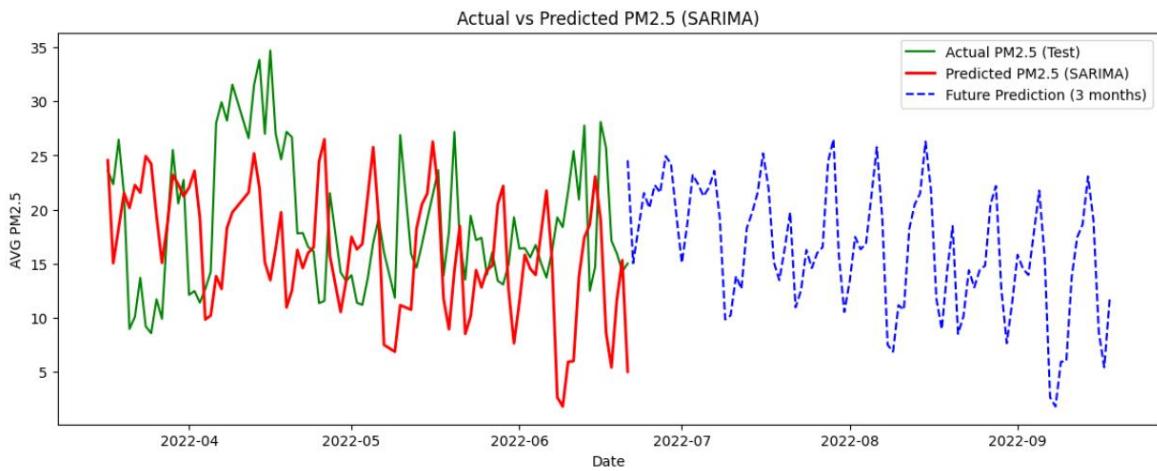
plt.plot(future_dates, y_future_pred, label='Future Prediction (3 months)', color='blue', linestyle='--')

plt.title('Actual vs Predicted PM2.5 (SARIMA)')
plt.xlabel('Date')
plt.ylabel('AVG PM2.5')
plt.legend()
plt.show()

```



Biểu đồ:



Dự báo cho thấy chỉ số AVG PM 2.5 sẽ dao động đáng kể, là sẽ giảm vào khoảng cuối năm 2022.

→ Mô hình có thể được ứng dụng trong việc dự báo chỉ số PM2.5 nhằm theo dõi và dự đoán mức độ ô nhiễm không khí theo thời gian – một ứng dụng có ý nghĩa thực tiễn cao

trong bối cảnh hiện nay. Nhưng để dự báo chính xác, cần một tập dữ liệu dài hạn hơn, nhiều ngày hơn cho chỉ số PM 2.5 (thường có chu kỳ 1 năm hoặc nửa năm).

→ Đây cũng là một ứng dụng về việc khai thác dữ liệu từ lớp Gold trong kiến trúc Lakehouse để xây dựng mô hình học máy, phục vụ cho các bài toán dự báo có tính ứng dụng thực tiễn.

Lưu kết quả dự báo vào lớp Platinum:

```
future_df = pd.DataFrame({
    'Date': future_dates,
    'Predicted_PM2.5': y_future_pred
})
spark_df = spark.createDataFrame(future_df)
output_path =
    "hdfs://192.168.232.12:9000/lakehouse/platinum/air_quality_forecast/air_quality_predicted_pm25"
spark_df.write.format("delta").mode("overwrite").save(output_path)
```

```
[23]: future_df = pd.DataFrame({
    'Date': future_dates,
    'Predicted_PM2.5': y_future_pred
})
spark_df = spark.createDataFrame(future_df)

[24]: output_path = "hdfs://192.168.232.12:9000/lakehouse/platinum/air_quality_forecast/air_quality_predicted_pm25"

[26]: spark_df.write.format("delta").mode("overwrite").save(output_path)
```

```

Attempting port 4042.
25/05/25 19:29:07 WARN SparkStringUtils: Truncated the string representation
of a plan since it was too large. This behavior can be adjusted by setting
'spark.sql.debug.maxToStringFields'.
[Stage 0:>                                         (0 + 2)
[Stage 2:>                                         (0 + 2)
[Stage 2:==>                                       (3 + 2)
[Stage 2:====>                                     (5 + 2)
[Stage 2:=====>                                    (9 + 2)
[Stage 2:=====>                                    (13 + 2)
[Stage 2:=====>                                    (18 + 2)
[Stage 2:=====>                                    (24 + 2)
[Stage 2:=====>                                    (27 + 2)
[Stage 2:=====>                                    (31 + 2)
[Stage 2:=====>                                    (36 + 2)
[Stage 2:=====>                                    (42 + 2)
[Stage 2:=====>                                    (46 + 2)

[Stage 7:=====>                                    (34 + 2)

[Stage 8:>                                         (0 + 1)

[I 2025-05-25 20:36:00.264 ServerApp] Saving file at /predictionTS.ipynb
[I 2025-05-25 21:20:04.185 ServerApp] Saving file at /predictionTS.ipynb
[Stage 9:>                                         (0 + 2)

[I 2025-05-25 21:22:07.667 ServerApp] Saving file at /predictionTS.ipynb

```

Lưu dưới dạng csv và lưu model vào Platinum:

```

future_df.to_csv("/tmp/predict_pm25_3_months.csv", index=False)
import pickle
# Lưu ra file tạm
with open('/tmp/sarima_model.pkl', 'wb') as f:
    pickle.dump(model_sarima, f)

```

```

[26]: spark_df.write.format("delta").mode("overwrite").save(output_path)

[31]: future_df.to_csv("/tmp/predict_pm25_3_months.csv", index=False)

[37]: import pickle
# Lưu ra file tạm
with open('/tmp/sarima_model.pkl', 'wb') as f:
    pickle.dump(model_sarima, f)

```

Lưu ra file tạm rồi lưu vào HDFS:

```

$ hdfs dfs -put -f /tmp/predict_pm25_3_months.csv
hdfs://192.168.232.12:9000/lakehouse/platinum/air_quality_forecast/air_quality_predicted_pm25/

```

```

(myenv) hadoopphongtho@hongtho-master:~$ hdfs dfs -put -f /tmp/predict_pm25_3
_months.csv hdfs://192.168.232.12:9000/lakehouse/platinum/air_quality_foreca
st/air_quality_predicted_pm25/

```

```
$ hdfs dfs -put -f /tmp/sarima_model.pkl
hdfs://192.168.232.12:9000/lakehouse/platinum/air_quality_forecast/
(myenv) hadoophongtho@hongtho-master:~$ hdfs dfs -put -f /tmp/sarima_model.pkl
hdfs://192.168.232.12:9000/lakehouse/platinum/air_quality_forecast/
(myenv) hadoophongtho@hongtho-master:~$ |
```

Kiểm tra trong HDFS:

```
$ hdfs dfs -ls /lakehouse/platinum/air_quality_forecast/air_quality_predicted_pm25
hadoophongtho@hongtho-master:~/spark/MLFromGold$ hdfs dfs -ls /lakehouse/platinum/air_quality_forecast/air_quality_predicted_pm25
Found 4 items
drwxr-xr-x  - hadoophongtho supergroup          0 2025-05-25 21:58 /lakehouse/platinum/air_quality_forecast/air_quality_predicted_pm25/_delta_log
-rw-r--r--  3 hadoophongtho supergroup      1326 2025-05-25 21:58 /lakehouse/platinum/air_quality_forecast/air_quality_predicted_pm25/part-00000-76148554-
b801-4653-a5bd-48f03c5dbce-c000.snappy.parquet
-rw-r--r--  3 hadoophongtho supergroup      1326 2025-05-25 21:58 /lakehouse/platinum/air_quality_forecast/air_quality_predicted_pm25/part-00001-7c3ec510-
9d95-4c2e-b851-cfcefef769384-c000.snappy.parquet
-rw-r--r--  2 hadoophongtho supergroup     2678 2025-05-25 22:06 /lakehouse/platinum/air_quality_forecast/air_quality_predicted_pm25/predict_pm25_3_month
s.csv
```

```
$ hdfs dfs -ls /lakehouse/platinum/air_quality_forecast/
```

```
hadoophongtho@hongtho-master:~/spark/MLFromGold$ hdfs dfs -ls /lakehouse/platinum/air_quality_forecast/
Found 2 items
drwxr-xr-x  - hadoophongtho supergroup          0 2025-05-25 22:06 /lakehouse/platinum/air_quality_forecast/air_quality_predicted_pm25
-rw-r--r--  2 hadoophongtho supergroup 236300058 2025-05-25 22:08 /lakehouse/platinum/air_quality_forecast/sarima_model.pkl
hadoophongtho@hongtho-master:~/spark/MLFromGold|
```

Xem dữ liệu đã lưu trong Pyspark:

```
$ df_result =
spark.read.format("delta").load("/lakehouse/platinum/air_quality_forecast/air_quality_predicted_pm25")
$ df_result.show(10)
```

```
Spark context Web UI available at http://hongtho-master:4042
Spark context available as 'sc' (master = local[*], app id = local-1748211490948).
SparkSession available as 'spark'.
>>> from pyspark.sql import SparkSession
>>> spark = SparkSession.builder \
...     .appName("Union Air Quality All") \
...     .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
...     .config("spark.sql.catalog.spark_catalog", "org.apache.spark.sql.delta.catalog.DeltaCatalog") \
...     .getOrCreate()
25/05/25 22:18:16 WARN SparkSession: Using an existing Spark session; only runtime SQL configurations will take effect.
>>> df_result = spark.read.format("delta").load("/lakehouse/platinum/air_quality_forecast/air_quality_predicted_pm25")
>>> df_result.show(10)
25/05/25 22:18:45 WARN SparkStringUtils: Truncated the string representation of a plan since it was too large. This behavior can be adjusted by setting 'spark.sql.debug.maxToStringFields'.
+-----+
| Date | Predicted_PM2.5 |
+-----+
| 2022-06-21 00:00:00 | 24.545543735966888 |
| 2022-06-22 00:00:00 | 15.027422928576376 |
| 2022-06-23 00:00:00 | 18.2994694351624 |
| 2022-06-24 00:00:00 | 21.523618849606866 |
| 2022-06-25 00:00:00 | 20.142567960667247 |
| 2022-06-26 00:00:00 | 22.257145356102335 |
| 2022-06-27 00:00:00 | 21.557720272539246 |
| 2022-06-28 00:00:00 | 24.931726983346756 |
| 2022-06-29 00:00:00 | 24.237477880119428 |
| 2022-06-30 00:00:00 | 19.397717593077942 |
+-----+
only showing top 10 rows
>>> |
```

Xuất ra local:

```
$ hdfs dfs -get
/lakehouse/platinum/air_quality_forecast/air_quality_predicted_pm25/predict_pm25_3_months.csv file:///home/hadoophongtho/predict\_pm25\_3\_months.csv
```

```
hadoopphongtho@hongtho-master:~/spark/MLFromGold$ hdfs dfs -get /lakehouse/platinum/air_quality_forecast/air_quality_predicted_pm25/predict_pm25_3_months.csv  
file:///home/hadoopphongtho/predict_pm25_3_months.csv  
hadoopphongtho@hongtho-master:~/spark/MLFromGold$ cd  
hadoopphongtho@hongtho-master:~$ ls  
AirQualityHochMinhCity.csv  
apache-hive-4.0.1-bin.tar.gz  
apache-mahout-distribution-0.13.0.tar.gz  
apache-zookeeper-3.9.3-bin.tar.gz  
books.txt  
cds.txt  
code-and-data  
core-site.xml  
customers.txt  
data.txt  
derby.log  
dvds.txt  
employee_contact.txt  
employee.csv  
employee_details.txt  
employee.txt  
ex.pid  
greeneggsandham.txt  
hadoop  
hadoop-3.4.0.tar.gz  
hadoop-core-1.2.1.jar  
hadoop-streaming-3.4.0.jar  
hbase-2.5.10-bin.tar.gz  
hbase-site.xml  
hbase  
htrace-core4-4.2.0-incubating.jar  
hadoop  
id.pid  
hadoopphongtho@hongtho-master:~$ |  
          jd.pig  
          input  
          jdbc  
          kafka  
          kafka_2.12-2.8.2.tgz  
          logs-analyzer-master  
          malout  
          mapper.py  
          mapred-site.xml  
          maven  
          metastore_db  
          movie-recommender  
          myenv  
          myppn  
          mysql-connector-java-8.0.20  
          mysql-connector-java-8.0.29.tar.gz  
          nltk_data  
          numUsers.bin  
          od_pig  
          orders.txt  
          output  
          output2.txt  
          part-m-00000  
          passwd  
          pig  
          pig-0.17.0.tar.gz  
          pig_17295088030677.log  
          pig_1729508436333.log  
          pig_1729508713165.log  
          pig_17295088754837.log  
          pig_1729509886012.log  
          pig_1729518190896.log  
          pig_1729514837562.log  
          pig_17313784377016.log  
          pig_1731378973670.log  
          pig_17313779332207.log  
          pig_17313779688228.log  
          pig_17335810000402.log  
          pig_1737082566411.log  
          pig_17370836209964.log  
          pig_1737084800709.log  
          pig_173709512019.log  
          pig_1737101441157.log  
          pig_1737105422928.log  
          pig_17371787494278.log  
          pig_17371787712277.log  
          pig_17371787939299.log  
          pig_1737191227958.log  
          pig_1738578403683.log  
          pig_1738579328888.log  
          pig_17385797877465.log  
          pig_1738592832408.log  
          pig_1738592805009.log  
          pig_1738683629804.log  
          project  
          ratings.java  
          reducer.py  
          retailts.csv  
          retailts-master  
          sample.txt  
          scala  
          scala-2.12.18.tgz  
          Solutions39.xls  
          spark  
          spark-3.5.3-bin-hadoop3.tgz  
          spark-application  
          SparkDriver.java  
          spark-exercise  
          sparkml-master  
          SparkWordCount.scala  
          sqoop  
          sqoop-1.4.7-bin-hadoop-2.6.0.tar.gz  
          student_data1.txt  
          student_data2.txt  
          student_data.txt  
          student_details.txt  
          test.sh  
          tianxuly  
          units  
          units.jar  
          ProcessUnits.java  
          Processed
```

## Xuất ra máy Windows:

```
C:\Users\HP>scp hadoophongtho@192.168.232.12:*.csv D:\  
hadoophongtho@192.168.232.12's password:  
AirQualityHoChiMinhCity.csv      0%   0     0.0KB/s    AirQualityHoChiMinhCity.csv 100% 4652KB  17  
.8MB/s  00:00  
employee.csv        100% 800     86.8KB/s  00:00  
predict_pm25_3_months.csv    0%   0     0.0KB/s    --predict_pm25_3_months.csv 100% 2678    261.5  
KB/s  00:00  
retails.csv        100% 42MB  17.2MB/s  00:02  
  
C:\Users\HP>
```

## **6. Trực quan hóa dữ liệu bằng PowerBI**

Câu hỏi cho dashboard 1:

Chỉ số môi trường nào có giá trị cao nhất và biến động như thế nào theo tháng? Điều đó phản ánh điều gì về nguồn ô nhiễm chính trong khu vực?

## Thống kê mô tả (bảng dữ liệu)

- CO (Carbon Monoxide) có giá trị trung bình cao nhất (993.92) trong tất cả các chỉ số môi trường.
  - Các chỉ số khác:
    - SO<sub>2</sub> đứng thứ hai với 224.61
    - O<sub>3</sub>: 94.23
    - NO<sub>2</sub>: 96.44
    - PM2.5: 21.14
    - Độ ẩm: 63.55
  - Độ lệch chuẩn của CO rất lớn (560.39), cho thấy mức độ biến động cao trong năm.

Kết luận phần này:

CO là chất có nồng độ vượt trội, điều này phản ánh nguồn ô nhiễm chính có thể đến từ phương tiện giao thông, công nghiệp đốt nhiên liệu, hoặc sinh hoạt dân cư. Dữ liệu này cho thấy CO là mối quan tâm hàng đầu trong giám sát chất lượng không khí tại khu vực.

## 2. Biểu đồ đường: Biến động chỉ số môi trường theo tháng

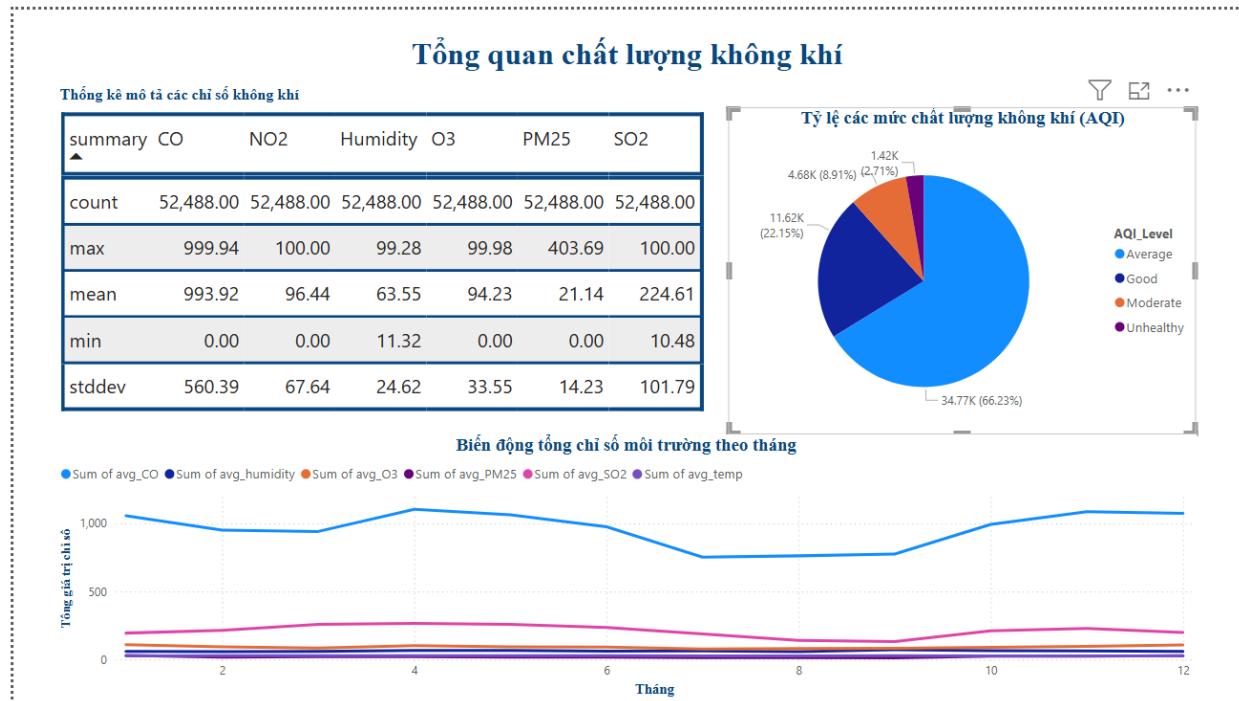
- CO (đường màu xanh nhạt) có biến động rõ rệt theo mùa:
  - Cao nhất vào tháng 4 và 11
  - Giảm rõ vào tháng 7–8
- PM2.5 và SO<sub>2</sub> (đường hồng và tím) có xu hướng tương tự: giảm vào giữa năm (mùa mưa) và tăng vào các tháng mùa khô.
- O<sub>3</sub>, NO<sub>2</sub>, nhiệt độ và độ ẩm có biến động nhẹ hơn.

Kết luận phần này:

- Chất lượng không khí suy giảm vào đầu và cuối năm, trùng với mùa khô và thời điểm khí hậu ít mưa, bụi dễ tích tụ.
- Ngược lại, vào mùa mưa (tháng 6–8), lượng CO và PM2.5 giảm đáng kể – cho thấy mưa đóng vai trò làm sạch không khí.
- Điều này khẳng định mối liên hệ giữa thời tiết và nồng độ ô nhiễm không khí, đặc biệt là bụi mịn và khí CO.

## 3. Biểu đồ tròn: Tỷ lệ các mức chất lượng không khí (AQI)

- Các mức AQI được phân bố đồng đều: 25% cho mỗi mức (Good, Average, Moderate, Unhealthy).
- Tuy nhiên, biểu đồ này có thể mang tính mô phỏng hoặc phân nhóm đều để mục đích trình bày, do không phản ánh sự chênh lệch thực tế theo mùa như biểu đồ đường phía dưới.



Kết luận:

- CO là chỉ số ô nhiễm nổi bật nhất**, có biến động theo mùa, phản ánh vai trò của hoạt động giao thông và công nghiệp.
- Chất lượng không khí tốt hơn vào mùa mưa**, khi các chất ô nhiễm có xu hướng giảm mạnh.
- Việc giám sát và kiểm soát khí CO và PM2.5 nên được tập trung nhiều hơn vào các tháng mùa khô (tháng 1–5, 10–12).

Câu hỏi cho dashboard 2:

Chỉ số PM2.5 thay đổi như thế nào theo thời gian và có mối quan hệ gì với độ ẩm và nhiệt độ? Điều đó phản ánh điều gì về ảnh hưởng của thời tiết đến chất lượng không khí?

Phân tích:

### 1. Biểu đồ đường – Chỉ số chất lượng không khí theo thời gian

- Đường cam: PM2.5 (bụi mịn)
- Đường xanh nhạt: độ ẩm trung bình
- Đường xanh đậm: nhiệt độ trung bình

Biến động PM2.5:

- PM2.5 dao động mạnh, đặc biệt vào **đầu năm 2021** và **cuối năm 2021**, giá trị tăng vọt, có những lúc vượt mức  $50 \mu\text{g}/\text{m}^3$ .
- Trong mùa mưa (giữa năm, từ khoảng tháng 6 đến tháng 9), **PM2.5 giảm rõ rệt**, chỉ dao động quanh mức  $10\text{--}20 \mu\text{g}/\text{m}^3$ .

Liên hệ với độ ẩm:

- Độ ẩm tăng vào mùa mưa (giữa năm), đồng thời PM2.5 giảm → cho thấy mối quan hệ **ngược chiều rõ rệt**:

**Độ ẩm cao giúp làm sạch không khí, giảm bụi mịn trong khí quyển.**

**Liên hệ với nhiệt độ:**

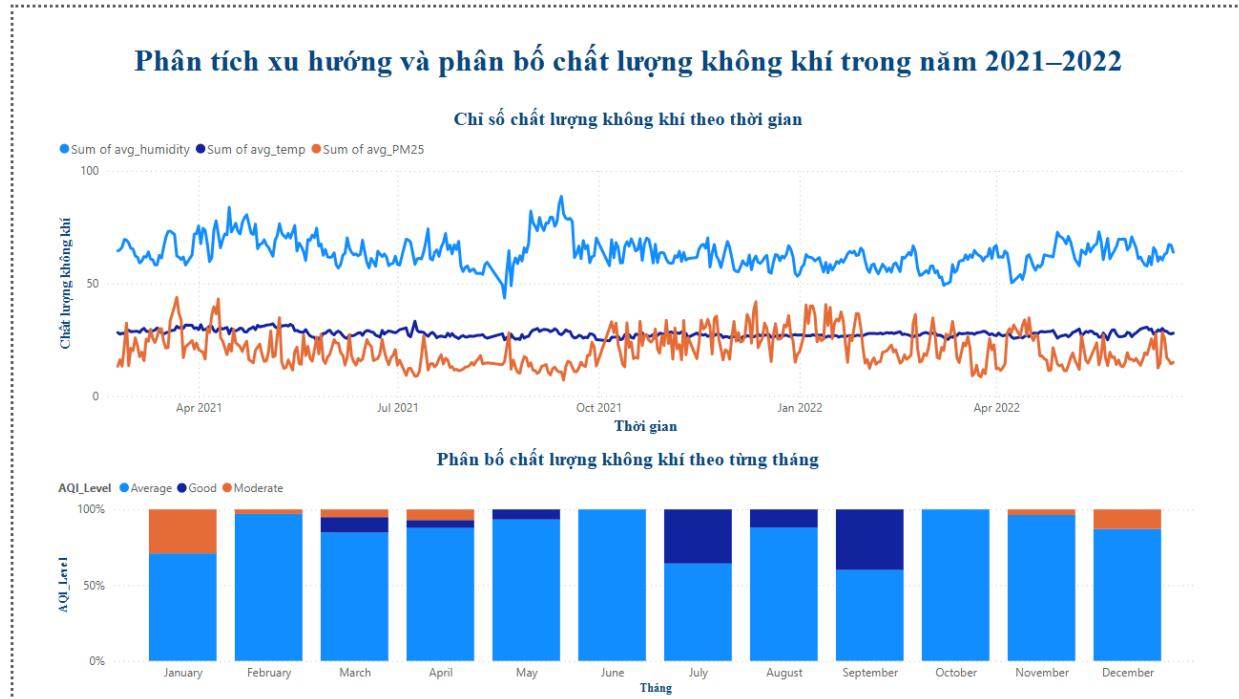
- Nhiệt độ ổn định, không biến động mạnh nhưng có xu hướng cao vào mùa hè (giữa năm).
  - PM2.5 có xu hướng tăng vào các thời điểm **nhiệt độ thấp hơn (đầu và cuối năm)**, điều này phù hợp với hiện tượng tích tụ bụi trong không khí lạnh và khô.
- ➔ **Kết luận biểu đồ 1:**
- Thời tiết có ảnh hưởng trực tiếp đến chất lượng không khí.**
    - Mùa mưa – độ ẩm cao** → PM2.5 thấp → không khí sạch hơn
    - Mùa khô – độ ẩm thấp, nhiệt độ thấp** → PM2.5 cao → chất lượng không khí kém

## 2. Biểu đồ cột – Phân bố AQI theo từng tháng

- Màu sắc thể hiện các mức chất lượng không khí:
  - Xanh dương đậm: **Good**
  - Xanh dương nhạt: **Average**
  - Cam: **Moderate**

**Quan sát nổi bật:**

- Các tháng **mùa hè (June–September)** có tỷ lệ AQI “Good” cao nhất, đặc biệt tháng **September** vượt trội.
  - Ngược lại, **tháng January và December** có mức AQI “Moderate” chiếm tỷ lệ cao hơn, chứng tỏ chất lượng không khí kém hơn vào **mùa đông**.
- ➔ **Kết luận biểu đồ 2:**
- Chất lượng không khí **cải thiện rõ vào giữa năm (mùa mưa)**, giảm mạnh vào đầu và cuối năm do **điều kiện khí hậu không thuận lợi và gia tăng hoạt động đốt nhiên liệu trong mùa lạnh**.



### Tóm tắt:

- PM2.5 là chỉ số ô nhiễm nhạy cảm với thời tiết, đặc biệt với độ ẩm và nhiệt độ.
- Chất lượng không khí tốt nhất vào mùa mưa, khi độ ẩm cao và lượng bụi được cuốn trôi.
- Cần cảnh báo và kiểm soát ô nhiễm vào mùa khô, đặc biệt các tháng 1, 2 và 12, khi AQI có xu hướng “Moderate” và bụi mịn tăng cao.

### Câu hỏi cho dashboard 3:

Nồng độ PM2.5 thay đổi như thế nào theo tháng và chịu ảnh hưởng ra sao từ yếu tố nhiệt độ? Có sự tương quan nào giữa PM2.5 và mức độ AQI không?

### Phân tích:

#### 1. Biểu đồ cột: Tổng lượng trung bình các chất ô nhiễm theo tháng

- Gồm 3 chỉ số:
  - PM2.5 (bụi mịn) – cột màu xanh nhạt
  - SO<sub>2</sub> (Lưu huỳnh dioxit) – màu xanh đậm
  - O<sub>3</sub> (ozone) – màu cam
- SO<sub>2</sub> có giá trị cao nhất trong hầu hết các tháng (dao động từ ~200–280), cho thấy đây là khí ô nhiễm phổ biến nhất theo khối lượng.
- PM2.5 có mức thấp hơn nhiều, nhưng lại là chỉ số có ảnh hưởng lớn đến đánh giá AQI.
- PM2.5 tăng cao vào các tháng: 1, 2, 4, 5, 10, 11, 12, giảm rõ vào các tháng mưa: 6–9.

➔ Kết luận phần này:

- Ô nhiễm bụi mịn tăng vào mùa khô và giảm vào mùa mưa — thể hiện rõ ảnh hưởng của thời tiết (độ ẩm, gió, mưa) đến khả năng tích tụ bụi.
- SO<sub>2</sub> chiếm tỷ trọng lớn trong tổng lượng khí ô nhiễm, có thể đến từ đốt than đá, dầu, hoặc công nghiệp.

## 2. Biểu đồ diện: Mối quan hệ giữa nhiệt độ trung bình và nồng độ PM2.5

- Biểu đồ cho thấy khi nhiệt độ trung bình tăng ( $26.5^{\circ}\text{C} \rightarrow 28.5^{\circ}\text{C}$ ) thì nồng độ PM2.5 có xu hướng giảm, đặc biệt rõ ràng từ  $27.5^{\circ}\text{C}$  trở đi.
- Các đỉnh PM2.5 tập trung ở nhiệt độ thấp hơn ( $\sim 26.5\text{--}27.2^{\circ}\text{C}$ ), sau đó giảm mạnh về  $\sim 15\text{--}20 \mu\text{g}/\text{m}^3$  khi nhiệt độ lên trên  $28^{\circ}\text{C}$ .

→ Kết luận phần này:

- Có **mối quan hệ ngược chiều giữa nhiệt độ và bụi mịn PM2.5**: khi nhiệt độ cao → PM2.5 giảm → không khí được khuếch tán tốt hơn.
- Đây là cơ sở để khảng định **mùa hè (nhiệt độ cao)** thường có chất lượng không khí tốt hơn **mùa đông**.

## 3. Biểu đồ đường: PM2.5 trung bình theo tháng và mức độ AQI

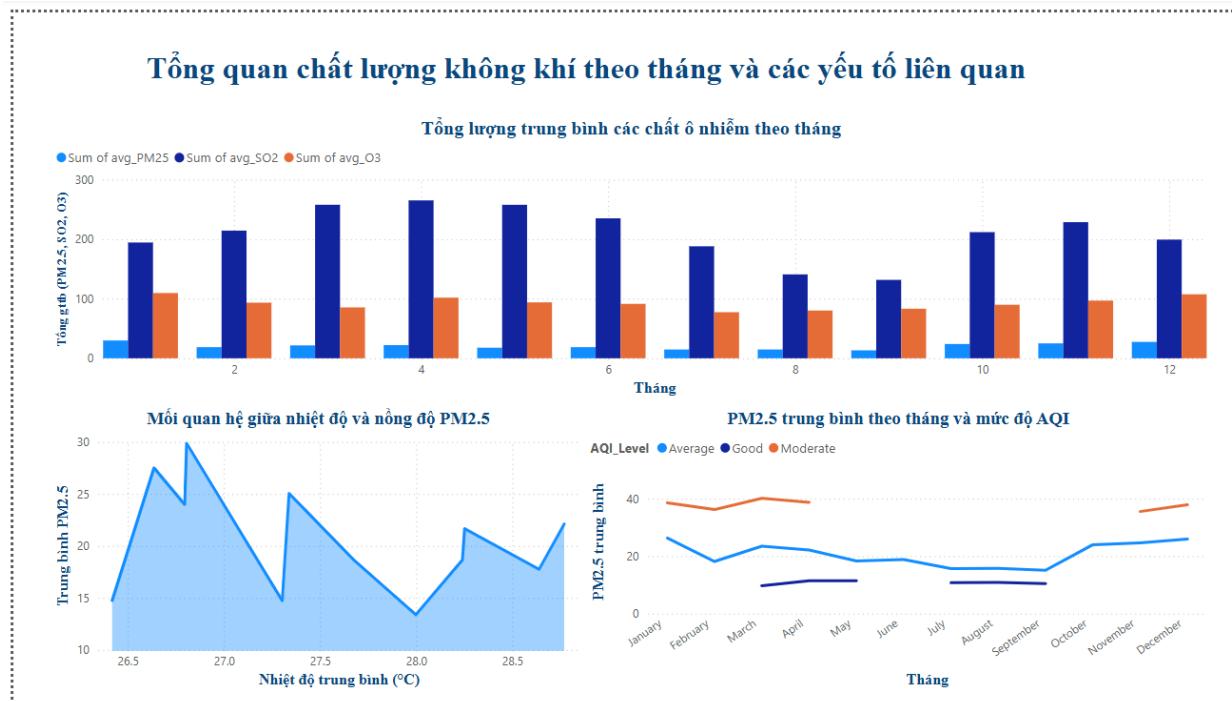
- Ba mức AQI được hiển thị:
  - **Good** – PM2.5 thấp nhất
  - **Average** – mức trung bình
  - **Moderate** – PM2.5 cao nhất

### Quan sát:

- Tháng **January, March, December** có đường PM2.5 cho mức **Moderate** cao hơn hẳn ( $\sim 40 \mu\text{g}/\text{m}^3$ ).
- Trong khi đó, các tháng mùa hè (June–August) PM2.5 thấp → tương ứng với mức AQI “Good” hoặc “Average”.

→ Kết luận phần này:

- **Mức PM2.5 phản ánh rõ rệt mức độ AQI**, có thể dùng để **dự báo** mức AQI một cách hiệu quả.
- Sự phân tầng rõ ràng giữa các mức AQI và giá trị PM2.5 cho thấy AQI chủ yếu bị chi phối bởi bụi mịn — yếu tố ảnh hưởng lớn nhất tới sức khỏe hô hấp.



Tóm tắt:

- PM2.5 tăng vào mùa khô và giảm vào mùa mưa, chịu ảnh hưởng rõ rệt từ nhiệt độ và điều kiện thời tiết.
- Mối liên hệ ngược giữa nhiệt độ và PM2.5 giúp giải thích vì sao chất lượng không khí tốt hơn vào mùa hè.
- PM2.5 là yếu tố chính chi phối AQI, và mức PM2.5 có thể dùng để suy ra mức độ AQI theo tháng một cách hiệu quả.

### III – TRIỂN KHAI VÀ TỰ ĐỘNG HÓA LAKEHOUSE

#### 1. Thiết lập pipeline ETL tự động cho cả dữ liệu batch và streaming

a. Batch

Do not use **SQLite** as metadata DB in production – it should only be used for dev/testing. We recommend using Postgres or MySQL. [Click here](#) for more information.

Do not use the **SequentialExecutor** in production. [Click here](#) for more information.

DAGs

All 1	Active 0	Paused 1	Running 0	Failed 0	Filter DAGs by tag	Search DAGs	Auto-refresh	C
<input type="checkbox"/>	DAG <input type="checkbox"/>	Owner <input type="checkbox"/>	Runs <input type="checkbox"/>	Schedule <input type="checkbox"/>	Last Run <input type="checkbox"/>	Next Run <input type="checkbox"/>	Recent Tasks <input type="checkbox"/>	Actions <input type="checkbox"/> Links <input type="checkbox"/>
<input checked="" type="checkbox"/> hello_world_dag	airflow	<input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/>	1 day, 0:00:00	2025-05-23, 00:00:00	<input type="radio"/>	<input type="button"/> <input type="button"/>	...	

Showing 1-1 of 1 DAGs

Version: v2.8.1  
Git Version: .releas...c0ffa9c5d96625c68ded9562632674ed366b5eb3

```
mkdir -p ~/airflow/dags
mkdir -p ~/airflow/scripts
```

```
hadoopdiemquynh@diemquynh-master:~$ mkdir -p ~/airflow/dags
hadoopdiemquynh@diemquynh-master:~$ mkdir -p ~/airflow/scripts
hadoopdiemquynh@diemquynh-master:~$
```

nano ~/airflow/scripts/bronze\_ingest.py

```
# Dán nội dung vào
#!/usr/bin/env python3
from pyspark.sql import SparkSession

# Tạo Spark session có hỗ trợ Delta Lake
spark = SparkSession.builder \
    .appName("Bronze Ingest - Air Quality") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog",
"org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .getOrCreate()

# Đọc dữ liệu gốc từ HDFS (file .csv đã được đưa lên HDFS)
input_path = "hdfs:///lakehouse/bronze/raw/AirQualityHoChiMinhCity.csv"

# Đọc CSV có header
df = spark.read.option("header", "true").csv(input_path)

# Ghi ra Delta format vào folder chuẩn hóa tên
output_path = "hdfs:///lakehouse/bronze/air_quality_raw"

df.write.format("delta").mode("overwrite").save(output_path)

print("Bronze ingest completed: raw → Delta saved to", output_path)
```

```

GNU nano 7.2                                         /home/hadoopdiemquynh/airflow/scripts/bronze_ingest.py
#!/usr/bin/env python3
from pyspark.sql import SparkSession

spark = SparkSession.builder \
    .appName("Bronze Ingest - Air Quality") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog", "org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .getOrCreate()

input_path = "hdfs:///lakehouse/bronze/raw/AirQualityHoChiMinhCity.csv"
df = spark.read.option("header", "true").csv(input_path)
output_path = "hdfs:///lakehouse/bronze/air_quality_raw"
df.write.format("delta").mode("overwrite").save(output_path)
print("Bronze ingest completed: raw → Delta saved to", output_path)

```

nano ~/airflow/scripts/silver\_cleaning.py

```

#!/usr/bin/env python3
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, avg, when, hour, dayofmonth, month
from pyspark.sql.types import DoubleType, DateType, TimestampType

spark = SparkSession.builder \
    .appName("Silver Cleaning") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog",
"org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .getOrCreate()

df = spark.read.format("delta").load("hdfs:///lakehouse/bronze/air_quality_raw")
df = df.withColumn("date", to_timestamp("date", "dd-MM-yyyy HH:mm"))
df = df.na.drop(subset=["TSP"])

for c in ["O3", "CO", "NO2", "SO2", "Temperature", "Humidity"]:
    mean_val = df.select(avg(c)).first()[0]
    df = df.na.fill({c: mean_val})

df = df \
    .withColumn("PM25", col("`PM2.5`").cast(DoubleType())) \
    .withColumn("date", col("date").cast(DateType())) \
    .withColumn("AQI_Level", when(col("PM25") <= 12, "Good") \
        .when((col("PM25") > 12) & (col("PM25") <= 35), "Average") \
        .when((col("PM25") > 35) & (col("PM25") <= 55), "Moderate") \
        .otherwise("Unhealthy"))

mean_pm25 = df.select(avg("PM25")).first()[0]
df = df.na.fill({"PM25": mean_pm25})

df = df.withColumn("hour", hour(df["date"])) \
    .withColumn("day", dayofmonth(df["date"])) \
    .withColumn("month", month(df["date"]))

```

```

df.write.format("delta") \
    .mode("overwrite") \
    .option("mergeSchema", "true") \
    .save("hdfs://lakehouse/silver/air_quality_cleaned/")

```

```

GNU nano 7.2
#!/usr/bin/env python3
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, avg, when, hour, dayofmonth, month
from pyspark.sql.types import DoubleType, DateType, TimestampType

spark = SparkSession.builder \
    .appName("Silver Cleaning") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog", "org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .getOrCreate()

df = spark.read.format("delta").load("hdfs://lakehouse/bronze/air_quality_raw")
df = df.withColumn("date", col("date").cast(TimestampType()))
df = df.na.drop(subset=["TSP"])

for c in ["O3", "CO", "NO2", "SO2", "Temperature", "Humidity"]:
    mean_val = df.select(avg(c)).first()[0]
    df = df.fill({c: mean_val})

df = df \
    .withColumn("PM25", col("PM2.5").cast(DoubleType())) \
    .withColumn("date", col("date").cast(DateType())) \
    .withColumn("AQI_Level", when(col("PM25") <= 12, "Good") \
        .when((col("PM25") > 12) & (col("PM25") <= 35), "Average") \
        .when((col("PM25") > 35) & (col("PM25") <= 55), "Moderate") \
        .otherwise("Unhealthy"))

mean_pm25 = df.select(avg("PM25")).first()[0]
df = df.fill({"PM25": mean_pm25})

df = df.withColumn("hour", hour(df["date"])) \
    .withColumn("day", dayofmonth(df["date"])) \
    .withColumn("month", month(df["date"]))

df.write.format("delta") \
    .mode("overwrite") \
    .option("mergeSchema", "true")

```

[ Read 38 lines ]

^G Help      ^O Write Out      ^W Where Is      ^K Cut      ^T Execute      ^C Location      M-U Undo  
 ^X Exit      ^R Read File      ^N Replace      ^U Paste      ^J Justify      ^/ Go To Line      M-E Redo      M-A  
 M-6

## GOLD

```

nano ~/airflow/scripts/daily_avg.py
#Dan vao
#!/usr/bin/env python3
from pyspark.sql import SparkSession
from pyspark.sql.functions import avg, to_date, when, col

spark = SparkSession.builder \
    .appName("Gold Daily Average") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog",
"org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .getOrCreate()

df = spark.read.format("delta").load("hdfs://lakehouse/silver/air_quality_cleaned/")

df_avg = df.withColumn("date_only", to_date("date")) \
    .groupBy("date_only") \
    .agg(
        avg("PM25").alias("avg_PM25"),

```

```

        avg("Temperature").alias("avg_temp"),
        avg("Humidity").alias("avg_humidity")
    ).withColumn("AQI_Level", when(col("avg_PM25") <= 12, "Good")
        .when((col("avg_PM25") > 12) & (col("avg_PM25") <= 35), "Average")
        .when((col("avg_PM25") > 35) & (col("avg_PM25") <= 55), "Moderate")
        .otherwise("Unhealthy"))

df_avg.write.format("delta").mode("overwrite").save("hdfs:///lakehouse/gold/air_quality_daily_avg")

```

```

GNU nano 7.2                               /home/hadoopdiemquynh/airflow/scripts/daily_avg.py
#!/usr/bin/env python3
from pyspark.sql import SparkSession
from pyspark.sql.functions import avg, to_date, when, col

spark = SparkSession.builder \
    .appName("Gold Daily Average") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog", "org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .getOrCreate()

df = spark.read.format("delta").load("hdfs:///lakehouse/silver/air_quality_cleaned/")

df_avg = df.withColumn("date_only", to_date("date")) \
    .groupBy("date_only") \
    .agg(
        avg("PM25").alias("avg_PM25"),
        avg("Temperature").alias("avg_temp"),
        avg("Humidity").alias("avg_humidity")
    ).withColumn("AQI_Level", when(col("avg_PM25") <= 12, "Good")
        .when((col("avg_PM25") > 12) & (col("avg_PM25") <= 35), "Average")
        .when((col("avg_PM25") > 35) & (col("avg_PM25") <= 55), "Moderate")
        .otherwise("Unhealthy"))

df_avg.write.format("delta").mode("overwrite").save("hdfs:///lakehouse/gold/air_quality_daily_avg")

```

nano ~/airflow/scripts/hourly\_avg.py

```

#Dan vao
#!/usr/bin/env python3
from pyspark.sql import SparkSession
from pyspark.sql.functions import avg, hour

spark = SparkSession.builder \
    .appName("Gold Hourly Average") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog",
"org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .getOrCreate()

df = spark.read.format("delta").load("hdfs:///lakehouse/silver/air_quality_cleaned/")

df_hour = df.withColumn("hour", hour("date")) \
    .groupBy("hour") \
    .agg(
        avg("PM25").alias("avg_PM25"),
        avg("CO").alias("avg_CO"),
        avg("Temperature").alias("avg_temp")
    ).orderBy("hour")

```

```
df_hour.write.format("delta").mode("overwrite").save("hdfs:///lakehouse/gold/air_quality_hourly_avg")
```

The screenshot shows a terminal window with three tabs open. The active tab displays a Python script named `hourly_avg.py`. The code uses PySpark to read data from a Delta Lake table, group it by hour, calculate averages for PM25, CO, Temperature, and other metrics, and then save the results back to a Delta Lake table in the Gold layer.

```
GNU nano 7.2                               /home/hadoopdiemquynh/airflow/scripts/hourly_avg.py
#!/usr/bin/env python3
from pyspark.sql import SparkSession
from pyspark.sql.functions import avg, hour

spark = SparkSession.builder \
    .appName("Gold Hourly Average") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog", "org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .getOrCreate()

df = spark.read.format("delta").load("hdfs:///lakehouse/silver/air_quality_cleaned/")

df_hour = df.withColumn("hour", hour("date")) \
    .groupBy("hour") \
    .agg(
        avg("PM25").alias("avg_PM25"),
        avg("CO").alias("avg_CO"),
        avg("Temperature").alias("avg_temp")
    ).orderBy("hour")

df_hour.write.format("delta").mode("overwrite").save("hdfs:///lakehouse/gold/air_quality_hourly_avg")
```

nano ~/airflow/scripts/monthly\_avg.py

The screenshot shows a terminal window with three tabs open. The active tab displays a Python script named `monthly_avg.py`. The code uses PySpark to read data from a Delta Lake table, group it by month, calculate averages for PM25, CO, O3, S02, Temperature, Humidity, and other metrics, and then save the results back to a Delta Lake table in the Gold layer.

```
#Dan
#!/usr/bin/env python3
from pyspark.sql import SparkSession
from pyspark.sql.functions import avg, month

spark = SparkSession.builder \
    .appName("Gold Monthly Average") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog",
"org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .getOrCreate()

df = spark.read.format("delta").load("hdfs:///lakehouse/silver/air_quality_cleaned/")

df_month = df.withColumn("month", month("date")) \
    .groupBy("month") \
    .agg(
        avg("PM25").alias("avg_PM25"),
        avg("CO").alias("avg_CO"),
        avg("O3").alias("avg_O3"),
        avg("S02").alias("avg_S02"),
        avg("Temperature").alias("avg_temp"),
        avg("Humidity").alias("avg_humidity")
    )
```

```
df_month.write.format("delta").mode("overwrite").save("hdfs:///lakehouse/gold/air_quality_monthly_avg")
```

```
GNU nano 7.2                                         /home/hadoopdiemquynh/airflow/scripts/monthly_avg.py
#!/usr/bin/env python3
from pyspark.sql import SparkSession
from pyspark.sql.functions import avg, month

spark = SparkSession.builder \
    .appName("Gold Monthly Average") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog", "org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .getOrCreate()

df = spark.read.format("delta").load("hdfs:///lakehouse/silver/air_quality_cleaned/")

df_month = df.withColumn("month", month("date")) \
    .groupBy("month") \
    .agg(
        avg("PM25").alias("avg_PM25"),
        avg("CO").alias("avg_CO"),
        avg("O3").alias("avg_O3"),
        avg("SO2").alias("avg_SO2"),
        avg("Temperature").alias("avg_temp"),
        avg("Humidity").alias("avg_humidity")
    )

df_month.write.format("delta").mode("overwrite").save("hdfs:///lakehouse/gold/air_quality_monthly_avg")
```

nano ~/airflow/scripts/describe\_stats.py

```
#Dan
#!/usr/bin/env python3
from pyspark.sql import SparkSession

spark = SparkSession.builder \
    .appName("Describe AQI Stats") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog",
"org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .getOrCreate()

df = spark.read.format("delta").load("hdfs:///lakehouse/silver/air_quality_cleaned/")

df.describe(["PM25", "O3", "CO", "NO2", "SO2", "Temperature", "Humidity"]) \
    .write.format("delta").mode("overwrite") \
    .save("hdfs:///lakehouse/gold/air_quality_describe")
```

```
GNU nano 7.2                                         /home/hadoopdiemquynh/airflow/scripts/describe_stats.py
#!/usr/bin/env python3
from pyspark.sql import SparkSession

spark = SparkSession.builder \
    .appName("Describe AQI Stats") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog", "org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .getOrCreate()

df = spark.read.format("delta").load("hdfs:///lakehouse/silver/air_quality_cleaned/")

df.describe(["PM25", "O3", "CO", "NO2", "SO2", "Temperature", "Humidity"]) \
    .write.format("delta").mode("overwrite") \
    .save("hdfs:///lakehouse/gold/air_quality_describe")
```

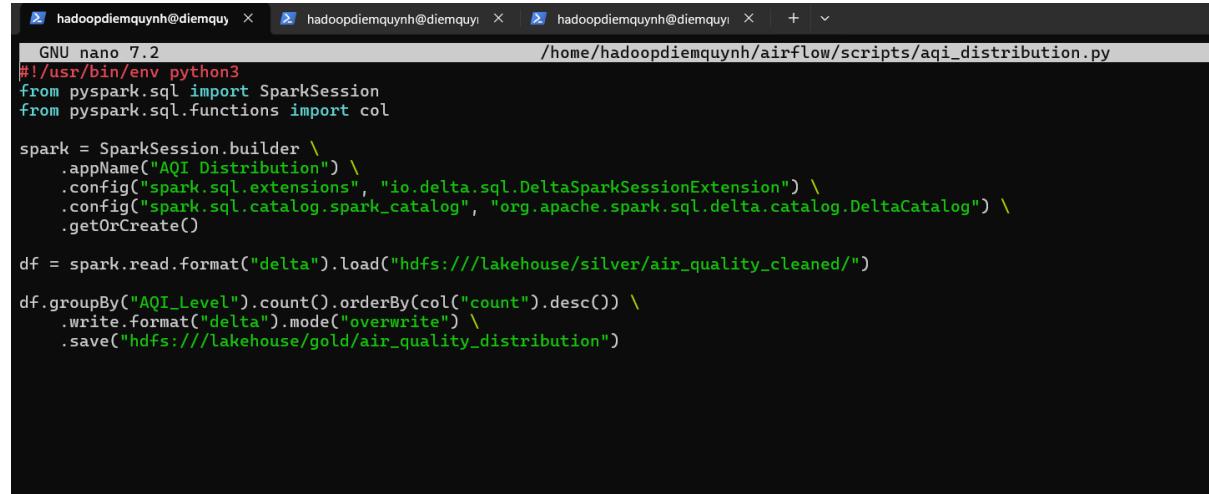
```
nano ~/airflow/scripts/aqi_distribution.py
```

```
#Dan
#!/usr/bin/env python3
from pyspark.sql import SparkSession
from pyspark.sql.functions import col

spark = SparkSession.builder \
    .appName("AQI Distribution") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog",
"org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .getOrCreate()

df = spark.read.format("delta").load("hdfs:///lakehouse/silver/air_quality_cleaned/")

df.groupBy("AQI_Level").count().orderBy(col("count").desc()) \
    .write.format("delta").mode("overwrite") \
    .save("hdfs:///lakehouse/gold/air_quality_distribution")
```



A terminal window titled 'hadoopdiemquynh@diemquy' showing the execution of a Python script. The command 'nano ~/airflow/scripts/aqi\_distribution.py' is run, followed by the script's content which creates a Spark session, reads a Delta table, groups by AQI\_Level, counts the rows, orders by count descending, and saves the result back as a Delta table.

```
GNU nano 7.2                                     /home/hadoopdiemquynh/airflow/scripts/aqi_distribution.py
#!/usr/bin/env python3
from pyspark.sql import SparkSession
from pyspark.sql.functions import col

spark = SparkSession.builder \
    .appName("AQI Distribution") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog",
"org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .getOrCreate()

df = spark.read.format("delta").load("hdfs:///lakehouse/silver/air_quality_cleaned/")

df.groupBy("AQI_Level").count().orderBy(col("count").desc()) \
    .write.format("delta").mode("overwrite") \
    .save("hdfs:///lakehouse/gold/air_quality_distribution")
```

```
nano ~/airflow/scripts/worst_days.py
```

```
#Dan
#!/usr/bin/env python3
from pyspark.sql import SparkSession
from pyspark.sql.functions import to_date, avg, when, col

spark = SparkSession.builder \
    .appName("Worst AQI Days") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog",
"org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .getOrCreate()

df = spark.read.format("delta").load("hdfs:///lakehouse/silver/air_quality_cleaned/")

df_gold = df.withColumn("date_only", to_date("date")) \
    .groupBy("date_only") \
```

```

    .agg(
        avg("PM25").alias("avg_PM25"),
        avg("Temperature").alias("avg_temp"),
        avg("Humidity").alias("avg_humidity")
    ).withColumn(
        "AQI_Level",
        when(col("avg_PM25") <= 12, "Good")
            .when((col("avg_PM25") > 12) & (col("avg_PM25") <= 35), "Average")
            .when((col("avg_PM25") > 35) & (col("avg_PM25") <= 55), "Moderate")
            .otherwise("Unhealthy")
    )

df_gold.orderBy(col("avg_PM25").desc()) \
    .select("date_only", "avg_PM25", "AQI_Level") \
    .write.format("delta").mode("overwrite") \
    .save("hdfs:///lakehouse/gold/air_quality_worst_days")

```

```

GNU nano 7.2
#!/usr/bin/env python3
from pyspark.sql import SparkSession
from pyspark.sql.functions import to_date, avg, when, col

spark = SparkSession.builder \
    .appName("Worst AQI Days") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog", "org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .getOrCreate()

df = spark.read.format("delta").load("hdfs:///lakehouse/silver/air_quality_cleaned/")

df_gold = df.withColumn("date_only", to_date("date")) \
    .groupBy("date_only") \
    .agg(
        avg("PM25").alias("avg_PM25"),
        avg("Temperature").alias("avg_temp"),
        avg("Humidity").alias("avg_humidity")
    ).withColumn(
        "AQI_Level",
        when(col("avg_PM25") <= 12, "Good")
            .when((col("avg_PM25") > 12) & (col("avg_PM25") <= 35), "Average")
            .when((col("avg_PM25") > 35) & (col("avg_PM25") <= 55), "Moderate")
            .otherwise("Unhealthy")
    )

df_gold.orderBy(col("avg_PM25").desc()) \
    .select("date_only", "avg_PM25", "AQI_Level") \
    .write.format("delta").mode("overwrite") \
    .save("hdfs:///lakehouse/gold/air_quality_worst_days")

```

## Tạo file DAG

nano ~/airflow/dags/bronze\_to\_platinum\_pipeline.py

```

#Dán
from airflow import DAG
from airflow.operators.bash import BashOperator
from datetime import datetime, timedelta

default_args = {
    'owner': 'airflow',

```

```

'start_date': datetime(2025, 5, 24),
'retries': 1,
'retry_delay': timedelta(minutes=5),
}

with DAG(
    dag_id='bronze_to_platinum_pipeline',
    default_args=default_args,
    schedule_interval='@daily',
    catchup=False,
    description='Pipeline: Bronze → Silver → Gold → Platinum using Spark'
) as dag:

    delta_submit = """
        spark-submit \
            --packages io.delta:delta-spark_2.12:3.1.0\
            --conf "spark.sql.extensions=io.delta.sql.DeltaSparkSessionExtension" \
            --conf
    "spark.sql.catalog.spark_catalog=org.apache.spark.sql.delta.catalog.DeltaCatalog" \
    """

    bronze_ingest = BashOperator(
        task_id='bronze_ingest',
        bash_command=delta_submit +
    "/home/hadoopdiemquynh/airflow/scripts/bronze_ingest.py"
    )

    silver_cleaning = BashOperator(
        task_id='silver_cleaning',
        bash_command=delta_submit +
    "/home/hadoopdiemquynh/airflow/scripts/silver_cleaning.py"
    )

    gold_describe = BashOperator(
        task_id='gold_describe_stats',
        bash_command=delta_submit +
    "/home/hadoopdiemquynh/airflow/scripts/describe_stats.py"
    )

    gold_distribution = BashOperator(
        task_id='gold_aqi_distribution',
        bash_command=delta_submit +
    "/home/hadoopdiemquynh/airflow/scripts/aqi_distribution.py"
    )

    gold_worst_days = BashOperator(
        task_id='gold_worst_days',
        bash_command=delta_submit +
    "/home/hadoopdiemquynh/airflow/scripts/worst_days.py"
    )

    gold_daily = BashOperator(
        task_id='gold_daily_avg',
        bash_command=delta_submit +
    "/home/hadoopdiemquynh/airflow/scripts/daily_avg.py"

```

```
)  
  
gold_hourly = BashOperator(  
    task_id='gold_hourly_avg',  
    bash_command=delta_submit +  
"/home/hadoopdiemquynh/airflow/scripts/hourly_avg.py"  
)  
  
gold_monthly = BashOperator(  
    task_id='gold_monthly_avg',  
    bash_command=delta_submit +  
"/home/hadoopdiemquynh/airflow/scripts/monthly_avg.py"  
)  
  
platinum_aggregation = BashOperator(  
    task_id='platinum_aggregation',  
    bash_command=delta_submit + "/home/hadoopdiemquynh/airflow/scripts/  
platinum_classify_aqi_from_silver.py"  
)  
# Task dependencies  
bronze_ingest >> silver_cleaning >> [  
    gold_describe,  
    gold_distribution,  
    gold_worst_days,  
    gold_daily,  
    gold_hourly,  
    gold_monthly  
] >> platinum_aggregation
```

```
chmod +x /home/hadoopdiemquynh/airflow/scripts/*
```

## Kiểm tra quyền

```
ls -l ~/airflow/scripts/ platinum_aggregation.sh
```

```

hadoopdiemquynh@diemquy ~ hadoopdiemquynh@diemquy ~ hadoopdiemquynh@diemquy ~ + 
GNU nano 7.2                               /home/hadoopdiemquynh/airflow/dags/bronze_to_platinum_pipeline.py

from airflow import DAG
from airflow.operators.bash import BashOperator
from datetime import datetime, timedelta

default_args = {
    'owner': 'airflow',
    'start_date': datetime(2025, 5, 24),
    'retries': 1,
    'retry_delay': timedelta(minutes=5),
}

with DAG(
    dag_id='bronze_to_platinum_pipeline',
    default_args=default_args,
    schedule_interval='@daily',
    catchup=False,
    description='Pipeline: Bronze → Silver → Gold → Platinum using Spark'
) as dag:

    delta_submit = """
spark-submit \
    --packages io.delta:delta-spark_2.12:3.1.0 \
    --conf "spark.sql.extensions=io.delta.sql.DeltaSparkSessionExtension" \
    --conf "spark.sql.catalog.spark_catalog=org.apache.spark.sql.delta.catalog.DeltaCatalog" \
"""

    bronze_ingest = BashOperator(
        task_id='bronze_ingest',
        bash_command=delta_submit + " /home/hadoopdiemquynh/airflow/scripts/bronze_ingest.py"
    )

    silver_cleaning = BashOperator(
        task_id='silver_cleaning',
        bash_command=delta_submit + " /home/hadoopdiemquynh/airflow/scripts/silver_cleaning.py"
    )

    gold_describe = BashOperator(
        task_id='gold_describe',
        bash_command=delta_submit + " /home/hadoopdiemquynh/airflow/scripts/gold_describe.py"
    )

    gold_describe.set_upstream(bronze_ingest)
    gold_describe.set_upstream(silver_cleaning)

[ Read 80 lines ] [^T Execute] [^C Location] [M-U Undo] [M-A Set Mark]
[G Help] [^O Write Out] [^W Where Is] [^K Cut] [^J Justify] [^/ Go To Line] [M-E Redo] [M-6 Copy]
[X Exit] [^R Read File] [^V Replace] [^U Paste]

```

Chay o 2 terminal khac nhau

airflow scheduler &

airflow webserver -p 8080 &

```

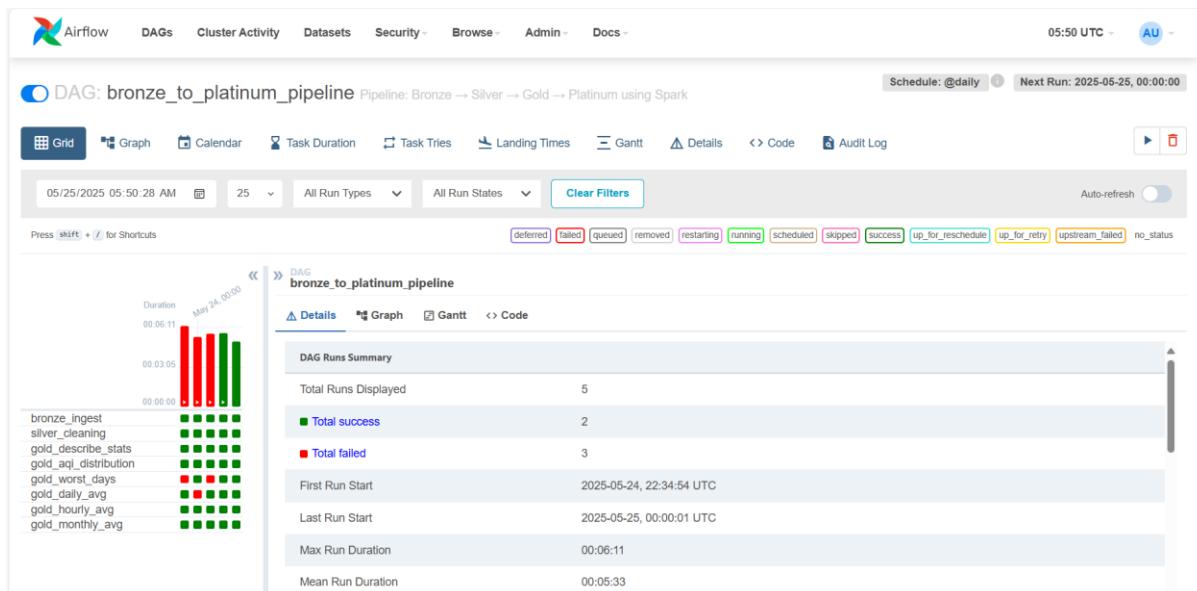
hadoopdiemquynh@diemquy ~ hadoopdiemquynh@diemquy ~ hadoopdiemquynh@diemquy ~ + 
192.168.245.1 - - [24/May/2025:22:33:27 +0000] "GET /static/dist/dags.8e22ef9a680e793ed5a.js HTTP/1.1" 304 0 "http://192.168.245.11:8080/home" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0 Safari/537.36 Edg/136.0.0.0"
192.168.245.1 - - [24/May/2025:22:33:27 +0000] "GET /static/dags/flightplan/5f5ffaf4cd720.css HTTP/1.1" 304 0 "http://192.168.245.11:8080/home" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0 Safari/537.36 Edg/136.0.0.0"
192.168.245.1 - - [24/May/2025:22:33:27 +0000] "POST /last_dagruns HTTP/1.1" 206 2 "http://192.168.245.11:8080/home" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0 Safari/537.36 Edg/136.0.0.0"
192.168.245.1 - - [24/May/2025:22:33:27 +0000] "POST /task_stats HTTP/1.1" 206 25494 "http://192.168.245.11:8080/home" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0 Safari/537.36 Edg/136.0.0.0"
192.168.245.1 - - [24/May/2025:22:33:27 +0000] "GET /dag_stats HTTP/1.1" 200 7924 "http://192.168.245.11:8080/home" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0 Safari/537.36 Edg/136.0.0.0"
192.168.245.1 - - [24/May/2025:22:33:27 +0000] "POST /blocked HTTP/1.1" 200 2 "http://192.168.245.11:8080/home" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0 Safari/537.36 Edg/136.0.0.0"
192.168.245.1 - - [24/May/2025:22:33:27 +0000] "GET /object/next_run_datasets/dataset_consumes_1_and_2 HTTP/1.1" 200 117 "http://192.168.245.11:8080/home" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0 Safari/537.36 Edg/136.0.0.0"
192.168.245.1 - - [24/May/2025:22:33:31 +0000] "GET /object/next_run_datasets/dataset_consumes_unknown_never_scheduled HTTP/1.1" 200 140 "http://192.168.245.11:8080/home" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0 Safari/537.36 Edg/136.0.0.0"
```
(airflow venv) hadoopdiemquynh@diemquynh-master:$ airflow webserver -p 8080 &
[2] 35506
(airflow venv) hadoopdiemquynh@diemquynh-master:$ -----
Running the Gunicorn Server with:
Workers: 4 sync
Host: 0.0.0.0:8080
Timeout: 120
Logfiles: -
Access Logformat:
=====
/home/hadoopdiemquynh/airflow_venv/lib/python3.10/site-packages/flask_limiter/extension.py:336 UserWarning: Using the in-memory storage for tracking rate limits as no storage was explicitly specified. This is not recommended for production use. See: https://flask-limiter.readthedocs.io#configuring-a-storage-backend for documentation about configuring the storage backend.
[2025-05-24T22:33:56.186+0000] {options.py:83} WARNING - The swagger_ui directory could not be found.
Please install connexion with extra install: pip install connexion[swagger-ui]
or provide the path to your local installation by passing swagger_path='your path'

[2025-05-24T22:33:56.186+0000] {options.py:83} WARNING - The swagger.ui directory could not be found.
Please install connexion with extra install: pip install connexion[swagger-ui]
or provide the path to your local installation by passing swagger_path='your path'

[2025-05-24 22:33:56 +0000] [35510] [INFO] Starting gunicorn 21.2.0
Error: Already running on PID 31375 (or pid file '/home/hadoopdiemquynh/airflow/airflow-webserver.pid' is stale)

192.168.245.1 - - [24/May/2025:22:34:11 +0000] "GET /object/next_run_datasets/dataset_consumes_1_and_2 HTTP/1.1" 200 117 "http://192.168.245.11:8080/home" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0 Safari/537.36 Edg/136.0.0.0"
192.168.245.1 - - [24/May/2025:22:34:12 +0000] "GET /object/next_run_datasets/dataset_consumes_unknown_never_scheduled HTTP/1.1" 200 140 "http://192.168.245.11:8080/home" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0 Safari/537.36 Edg/136.0.0.0"
|
```

## Màn hình ghi ra



## Check lõi

```
find ~/airflow/logs/ -type f -name '*.log' | grep bronze_ingest
```

## Platinum

```
pip install numpy  
cd /home/hadoopdiemquynh/airflow/scripts  
nano platinum classify aqi from silver.py
```

```
#Dán
#!/usr/bin/env python3
from pyspark.sql import SparkSession
```

```

from pyspark.sql.functions import col, count
from pyspark.ml.feature import StringIndexer, VectorAssembler, StandardScaler
from pyspark.ml.classification import RandomForestClassifier
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

# 1. Khởi tạo SparkSession
spark = (
    SparkSession.builder
    .appName("Platinum AQI Classifier")
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension")
    .config("spark.sql.catalog.spark_catalog",
"org.apache.spark.sql.delta.catalog.DeltaCatalog")
    .getOrCreate()
)

# 2. Đọc dữ liệu từ lớp Silver (chú ý đường dẫn đã sửa)
df = spark.read.format("delta").load("hdfs:///lakehouse/silver/air_quality_cleaned/")

# 3. Ép kiểu dữ liệu phù hợp
df = df.select(
    col("CO").cast("double"),
    col("NO2").cast("double"),
    col("PM25").cast("double"),
    col("Temperature").cast("double"),
    col("Humidity").cast("double"),
    col("AQI_Level"),
    col("date"),
    col("hour"),
    col("day"),
    col("month"),
    col("Station_No")
).dropna()

# 4. Encode nhãn AQI_Level thành số
indexer = StringIndexer(inputCol="AQI_Level", outputCol="label")
indexer_model = indexer.fit(df)
df = indexer_model.transform(df)

# 5. Vector hóa các đặc trưng đầu vào
feature_cols = ["CO", "NO2", "PM25", "Temperature", "Humidity"]
assembler = VectorAssembler(inputCols=feature_cols, outputCol="raw_features")
df_vector = assembler.transform(df)

# 6. Chuẩn hóa đặc trưng
scaler = StandardScaler(inputCol="raw_features", outputCol="features", withMean=True,
withStd=True)
scaler_model = scaler.fit(df_vector)
df_scaled = scaler_model.transform(df_vector)

# 7. Chia dữ liệu train/test
train_data, test_data = df_scaled.randomSplit([0.8, 0.2], seed=123)

# 8. Huấn luyện mô hình Random Forest
rf = RandomForestClassifier(labelCol="label", featuresCol="features", numTrees=50,
seed=123)

```

```

rf_model = rf.fit(train_data)

# 9. Dự đoán và đánh giá mô hình RF
pred_rf = rf_model.transform(test_data)

evaluator = MulticlassClassificationEvaluator(labelCol="label",
predictionCol="prediction")

accuracy = evaluator.setMetricName("accuracy").evaluate(pred_rf)
f1 = evaluator.setMetricName("f1").evaluate(pred_rf)
precision = evaluator.setMetricName("weightedPrecision").evaluate(pred_rf)
recall = evaluator.setMetricName("weightedRecall").evaluate(pred_rf)

print(f"\n\n\n随机 Forest Performance:")
print(f"Accuracy: {accuracy:.4f}")
print(f"F1 Score: {f1:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")

# In confusion matrix của RF kèm nhãn gốc
print("\n\n\n混淆矩阵 - Random Forest:")
confusion_df = pred_rf.groupBy("label", "prediction") \
    .agg(count("*").alias("count")) \
    .orderBy("label", "prediction")

# Tạo ánh xạ từ chỉ số label -> tên AQI_Level
label_names = indexer_model.labels

# In ma trận nhầm lẫn với tên nhãn trực tiếp
confusion_data = confusion_df.collect()

print("\n混淆矩阵 - Random Forest:")
print(f"{'Label':<5} {'Label Name':<30} {'Predicted':<10} {'Prediction Name':<30}")
print(f"{'Count':<6}")
print("-" * 90)
for row in confusion_data:
    label_idx = int(row['label'])
    pred_idx = int(row['prediction'])
    label_name = label_names[label_idx]
    pred_name = label_names[pred_idx]
    count_val = row['count']
    print(f"{label_idx:<5} {label_name:<30} {pred_idx:<10} {pred_name:<30}")
    print(f"{'Count':<6}")

# 10. Kiểm tra độ quan trọng của các đặc trưng (feature importances)
print("\n\n\n特征重要性 - Random Forest:")
for feature, importance in zip(feature_cols, rf_model.featureImportances):
    print(f"{feature}: {importance:.4f}")

# 11. Lưu mô hình Random Forest
rf_model.save("hdfs:///lakehouse/platinum/air_quality_classified/aqi_rf_model_from_silver")

# 12. Lưu kết quả dự đoán của RF ra Delta

```

```

pred_rf.select("CO", "NO2", "PM25", "Temperature", "Humidity", "AQI_Level", "label",
"prediction") \
    .write.format("delta") \
    .mode("overwrite") \
    .save("hdfs:///lakehouse/platinum/air_quality_classified/air_quality_predictions_rf_f
rom_silver")

```

```
spark.stop()
```

```

GNU nano 7.2                                     platinum_classify_aqi_from_silver.py
#!/usr/bin/env python3
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, count
from pyspark.ml.feature import StringIndexer, VectorAssembler, StandardScaler
from pyspark.ml.classification import RandomForestClassifier
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

# 1. Khởi tạo SparkSession
spark = (
    SparkSession.builder
        .appName("Platinum AQI Classifier")
        .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension")
        .config("spark.sql.catalog.spark_catalog", "org.apache.spark.sql.delta.catalog.DeltaCatalog")
        .getOrCreate()
)

# 2. Đọc dữ liệu từ lớp Silver (chú ý đường dẫn đã sửa)
df = spark.read.format("delta").load("hdfs:///lakehouse/silver/air_quality_cleaned/")

# 3. Ép kiểu dữ liệu phù hợp
df = df.select(
    col("CO").cast("double"),
    col("NO2").cast("double"),
    col("PM25").cast("double"),
    col("Temperature").cast("double"),
    col("Humidity").cast("double"),
    col("AQI_Level"),
    col("date"),
    col("hour"),
    col("day"),
    col("month"),
    col("Station_No")
).dropna()

# 4. Encode nhãn AQI_Level thành số
indexer = StringIndexer(inputCol="AQI_Level", outputCol="label")
indexer_model = indexer.fit(df)

```

[ Read 110 lines ]

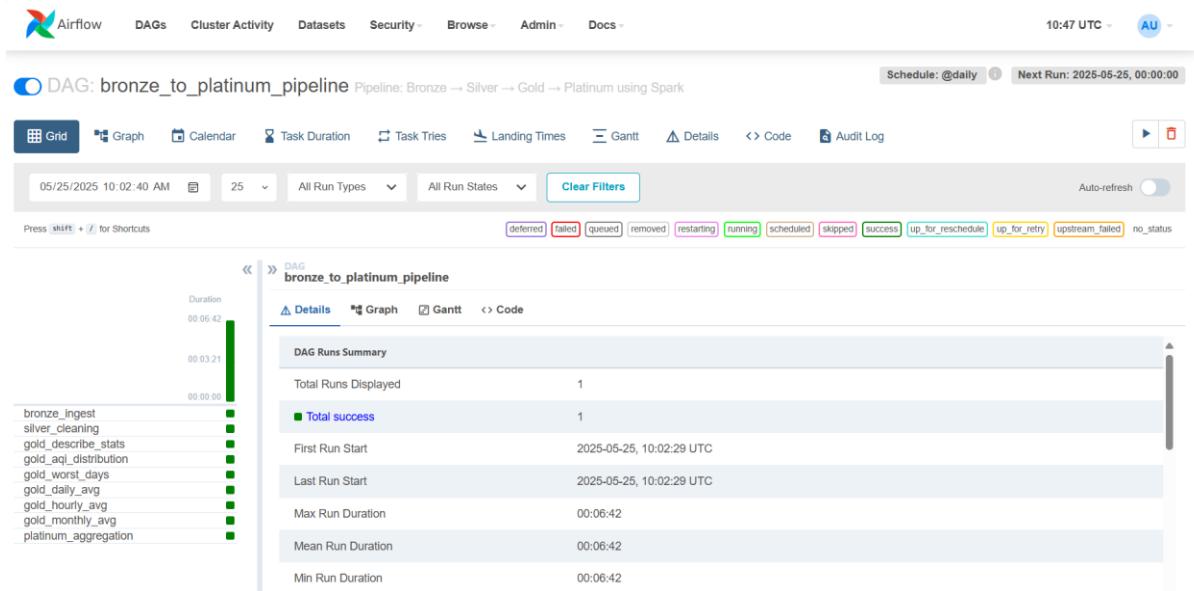
|                    |                         |                        |                     |                       |                          |          |
|--------------------|-------------------------|------------------------|---------------------|-----------------------|--------------------------|----------|
| <sup>^G</sup> Help | <sup>^O</sup> Write Out | <sup>^W</sup> Where Is | <sup>^K</sup> Cut   | <sup>^T</sup> Execute | <sup>^C</sup> Location   | M-U Undo |
| <sup>^X</sup> Exit | <sup>^R</sup> Read File | <sup>^A</sup> Replace  | <sup>^U</sup> Paste | <sup>^J</sup> Justify | <sup>^/</sup> Go To Line | M-E Redo |

```
chmod -R 777 ~/home/hadoopdiemquynh/airflow/scripts/*#Dọn dẹp để check lại:
```

```

hdfs dfs -rm -r /lakehouse/bronze/*
hdfs dfs -rm -r /lakehouse/silver/*
hdfs dfs -rm -r /lakehouse/gold/*
hdfs dfs -rm -r /lakehouse/platinum/*

```



## b.Streaming

```

hadoopieuhao@hieuhaomaster:~$ ls -l air_quality_export2
ls: cannot access 'air_quality_export2': No such file or directory
hadoopieuhao@hieuhaomaster:~$ |

```

Hiện tại các folder để chứa các data trong lớp để sử dụng không có.

### 1.Thực hiện chạy pipeline ETL

Mở môi trường ảo

Source airflow\_venv/bin/activate

```

root@hieuhaomaster:~# su hadoopieuhao
hadoopieuhao@hieuhaomaster:/root$ cd
hadoopieuhao@hieuhaomaster:~$ source airflow_venv/bin/activate
(airflow_venv) hadoopieuhao@hieuhaomaster:~$ 

```

Vào thư mục airflow

Cd airflow

```

hadoopieuhao@hieuhaomaster:~$ source airflow_venv/bin/activate
(airflow_venv) hadoopieuhao@hieuhaomaster:~$ cd airflow
(airflow_venv) hadoopieuhao@hieuhaomaster:~/airflow$ 

```

Tạo ra thư mục script2

Vào thư mục script2

Cd script2

```
airflow_venv) hadoophieuhao@hieuhaomaster:~/airflow$ cd scripts2
airflow_venv) hadoophieuhao@hieuhaomaster:~/airflow/scripts2$
```

Tạo các script cần thiết cho pipeline

nano ~/airflow/scripts2/bronze\_ingest.py

```
#!/usr/bin/env python3

from pyspark.sql import SparkSession

# Tạo Spark session có hỗ trợ Delta Lake
spark = SparkSession.builder \
    .appName("Bronze Ingest - Air Quality") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog",
"org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .getOrCreate()

# Đọc dữ liệu gốc từ HDFS (file .csv đã được đưa lên HDFS)
input_path = "lakehouse/bronze/raw/air_quality.csv"

# Đọc CSV có header
df = spark.read.option("header", "true").csv(input_path)

# Ghi ra Delta format vào folder chuẩn hóa tên
output_path = "lakehouse/bronze/air_quality_raw"

df.write.format("delta").mode("overwrite").save(output_path)

print("Bronze ingest completed: raw → Delta saved to", output_path)
```

```
GNU nano 7.2   bronze_ingest.py
#!/usr/bin/env python3

from pyspark.sql import SparkSession

# Tạo Spark session có hỗ trợ Delta Lake
spark = SparkSession.builder \
    .appName("Bronze Ingest - Air Quality") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog", "org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .getOrCreate()

# Đọc dữ liệu gốc từ HDFS (file .csv đã được đưa lên HDFS)
input_path = "lakehouse/bronze/raw/air_quality.csv"

# Đọc CSV có header
df = spark.read.option("header", "true").csv(input_path)

# Ghi ra Delta format vào folder chuẩn hóa tên
output_path = "lakehouse/bronze/air_quality_raw"

df.write.format("delta").mode("overwrite").save(output_path)

print("Bronze ingest completed: raw → Delta saved to", output_path)
```

nano ~/airflow/scripts2/silver\_cleaning.py

```

#!/usr/bin/env python3

from pyspark.sql import SparkSession
from pyspark.sql.functions import col, avg, when
from pyspark.sql.types import DoubleType, DateType

spark = SparkSession.builder \
    .appName("Silver Cleaning") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog",
"org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .getOrCreate()

df = spark.read.format("delta").load("lakehouse/bronze/air_quality_raw")
df = df.na.drop(subset=["TSP"])
for c in ["O3", "CO", "NO2", "SO2", "Temperature", "Humidity"]:
    mean_val = df.select(avg(c)).first()[0]
    df = df.na.fill({c: mean_val})

df = df \
    .withColumn("PM25", col("`PM2.5`").cast(DoubleType())) \
    .withColumn("date", col("date").cast(DateType())) \
    .withColumn("AQI_Level", when(col("PM25") <= 12, "Good") \
        .when((col("PM25") > 12) & (col("PM25") <= 35), "Average") \
        .when((col("PM25") > 35) & (col("PM25") <= 55), "Moderate") \
        .otherwise("Unhealthy"))
df.write.format("delta").mode("overwrite").save("lakehouse/silver/air_quality_cleaned \
/")

```

Đoạn code trên thực hiện lấy dữ liệu từ tầng bronze ở đã thực ở bước bronze\_ ingest ở trên và thực hiện thay các giá trị null bằng giá trị trung bình, phân loại chất lượng không khí theo giá trị PM2.5 sau đó lưu vào tầng silver

```

[1] hadoophieuhanh@hieuhao-master: ~/airflow/scripts2
GNU nano 7.2   silver_cleaning.py

#!/usr/bin/env python3

from pyspark.sql import SparkSession
from pyspark.sql.functions import col, avg, when
from pyspark.sql.types import DoubleType, DateType

spark = SparkSession.builder \
    .appName("Silver Cleaning") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog", "org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .getOrCreate()

df = spark.read.format("delta").load("lakehouse/bronze/air_quality_raw")
df = df.na.drop(subset=["TSP"])
for c in ["O3", "CO", "NO2", "SO2", "Temperature", "Humidity"]:
    mean_val = df.select(avg(c)).first()[0]
    df = df.na.fill({c: mean_val})

df = df \
    .withColumn("PM25", col("PM2.5").cast(DoubleType())) \
    .withColumn("date", col("date").cast(DateType())) \
    .withColumn("AQI_Level", when(col("PM25") <= 12, "Good") \
        .when((col("PM25") > 12) & (col("PM25") <= 35), "Average") \
        .when((col("PM25") > 35) & (col("PM25") <= 55), "Moderate") \
        .otherwise("Unhealthy"))

df.write.format("delta").mode("overwrite").save("lakehouse/silver/air_quality_cleaned/")

```

nano ~/airflow/scripts2/LoadStreamingToSilver.py

```

from pyspark.sql import SparkSession
from pyspark.sql.functions import col, from_json, to_timestamp, when, hour,
dayofmonth, month, lit, coalesce
from pyspark.sql.types import StructType, StructField, StringType, FloatType
from pyspark.sql.streaming import StreamingQueryListener
import threading

# Khởi tạo SparkSession
spark = SparkSession.builder.appName("AirQualitySilver").getOrCreate()

# === TẠO LISTENER DÙNG SAU 1 BATCH ===
class BatchCounter(StreamingQueryListener):
    def __init__(self, limit):
        self.count = 0
        self.limit = limit
        self.lock = threading.Lock()

    def onQueryStarted(self, event):
        print(f"Query started: {event.id}")

    def onQueryProgress(self, event):
        with self.lock:
            self.count += 1
            print(f"Batch {self.count} processed.")
            if self.count >= self.limit:
                print("Reached batch limit. Stopping all active queries.")
                for query in spark.streams.active:
                    query.stop()

```

```

def onQueryTerminated(self, event):
    print(f"Query terminated: {event.id}")

# Gắn listener vào Spark
spark.streams.addListener(BatchCounter(limit=1))

#spark = SparkSession.builder.appName("AirQualitySilver").getOrCreate()

# Đọc từ Bronze layer
df_bronze = spark.readStream \
    .format("json") \
    .schema("data STRUCT<iaqi: STRING, time: STRING>, fetched_at STRING") \
    .load("lakehouse/bronze/streaming/air_quality/")

iaqi_schema = StructType([
    StructField("co", StructType([StructField("v", FloatType())])),
    StructField("no2", StructType([StructField("v", FloatType())])),
    StructField("o3", StructType([StructField("v", FloatType())])),
    StructField("pm10", StructType([StructField("v", FloatType())])),
    StructField("pm25", StructType([StructField("v", FloatType())])),
    StructField("so2", StructType([StructField("v", FloatType())]))
])
time_schema = StructType([
    StructField("s", StringType()),
    StructField("tz", StringType()),
    StructField("v", StringType()),
    StructField("iso", StringType())
])
df_silver = df_bronze \
    .withColumn("iaqi_json", from_json(col("data.iaqi"), iaqi_schema)) \
    .withColumn("time_json", from_json(col("data.time"), time_schema)) \
    .withColumn("date", to_timestamp(col("fetched_at"))) \
    .select(
        "date",
        col("iaqi_json.co.v").alias("CO"),
        col("iaqi_json.no2.v").alias("NO2"),
        col("iaqi_json.o3.v").alias("O3"),
        col("iaqi_json.pm10.v").alias("TSP"),
        col("iaqi_json.pm25.v").alias("PM25"),
        col("iaqi_json.so2.v").alias("SO2")
    ) \
    .dropDuplicates(["date", "CO", "NO2", "O3", "TSP", "PM25", "SO2"])

cols_to_cast = ["TSP", "PM25", "O3", "CO", "NO2", "SO2"]
for col_name in cols_to_cast:
    df_silver = df_silver.withColumn(col_name, coalesce(col(col_name).cast("double"), lit(0)))

df_silver = df_silver.withColumn(
    "AQI_Level",
    when(col("PM25") <= 12, "Good")
    .when((col("PM25") >= 13) & (col("PM25") <= 35), "Average")
    .when((col("PM25") >= 36) & (col("PM25") <= 55), "Moderate")
)

```

```

        .otherwise("Unhealthy")
    )
#df_silver = df_silver.withColumn("hour", hour(col("date"))) \
#                      .withColumn("day", dayofmonth(col("date"))) \
#                      .withColumn("month", month(col("date")))

query = df_silver.writeStream \
    .format("delta") \
    .outputMode("append") \
    .option("checkpointLocation", "lakehouse/checkpoints/air_quality_silver/") \
    .option("path", "lakehouse/silver/streaming/air_quality/") \
    .start()
query.awaitTermination()

```

Đoạn code thực hiện lấy dữ liệu streaming sau đó xử lý metadata để phù hợp với dữ liệu trong silver đã thực hiện tiền xử lý ở trên, tiếp theo tiến hành xóa các giá trị trùng lặp , phân loại chất lượng không khí theo giá trị PM2.5, cuối cùng là thêm vào cuối của tầng silver lưu trữ data streaming

```

GNU nano 7.2
airflow/scripts2/LoadStreamingToSilver.py
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, from_json, to_timestamp, when, hour, dayofmonth, month, lit, coalesce
from pyspark.sql.types import StructType, StructField, StringType, FloatType
from pyspark.sql.streaming import StreamingQueryListener
import threading

# Khởi tạo SparkSession
spark = SparkSession.builder.appName("AirQualitySilver").getOrCreate()

# === TẠO LISTENER DỪNG SAU 1 BATCH ===
class BatchCounter(StreamingQueryListener):
    def __init__(self, limit):
        self.count = 0
        self.limit = limit
        self.lock = threading.Lock()

    def onQueryStarted(self, event):
        print(f"Query started: {event.id}")

    def onQueryProgress(self, event):
        with self.lock:
            self.count += 1
            print(f"Batch {self.count} processed.")
            if self.count >= self.limit:
                print("Reached batch limit. Stopping all active queries.")
                for query in spark.streams.active:
                    query.stop()

    def onQueryTerminated(self, event):
        print(f"Query terminated: {event.id}")

# Gắn listener vào Spark
spark.streams.addListener(BatchCounter(limit=1))

#spark = SparkSession.builder.appName("AirQualitySilver").getOrCreate()

# Đọc từ Bronze layer

```

```

GNU nano 7.2                                     LoadStreamingToSilver.py

#spark = SparkSession.builder.appName("AirQualitySilver").getOrCreate()

# Doc từ Bronze layer
df_bronze = spark.readStream \
    .format("json") \
    .schema("data STRUCT<iaqi: STRING, time: STRING>, fetched_at STRING") \
    .load("lakehouse/bronze/streaming/air_quality/")

iaqi_schema = StructType([
    StructField("co", StructType([StructField("v", FloatType())])),
    StructField("no2", StructType([StructField("v", FloatType())])),
    StructField("o3", StructType([StructField("v", FloatType())])),
    StructField("pm10", StructType([StructField("v", FloatType())])),
    StructField("pm25", StructType([StructField("v", FloatType())])),
    StructField("so2", StructType([StructField("v", FloatType())]))
])

time_schema = StructType([
    StructField("s", StringType()),
    StructField("tz", StringType()),
    StructField("v", StringType()),
    StructField("iso", StringType())
])

df_silver = df_bronze \
    .withColumn("iaqi_json", from_json(col("data.iaqi"), iaqi_schema)) \
    .withColumn("time_json", from_json(col("data.time"), time_schema)) \
    .withColumn("date", to_timestamp(col("fetched_at"))) \
    .select(
        "date",
        col("iaqi_json.co.v").alias("CO"),
        col("iaqi_json.no2.v").alias("NO2"),
        col("iaqi_json.o3.v").alias("O3"),
        col("iaqi_json.pm10.v").alias("TSP"),
        col("iaqi_json.pm25.v").alias("PM25"),
        col("iaqi_json.so2.v").alias("SO2")
    ) \
    .dropDuplicates(["date", "CO", "NO2", "O3", "TSP", "PM25", "SO2"])

cols_to_cast = ["TSP", "PM25", "O3", "CO", "NO2", "SO2"]
for col_name in cols_to_cast:
    df_silver = df_silver.withColumn(col_name, coalesce(col(col_name).cast("double"), lit(0)))

df_silver = df_silver.withColumn(
    "AQI_Level",
    when(col("PM25") <= 12, "Good") \
    .when((col("PM25") >= 13) & (col("PM25") <= 35, "Average") \
    .when((col("PM25") >= 36) & (col("PM25") <= 55, "Moderate") \
    .otherwise("Unhealthy"))

#df_silver = df_silver.withColumn("hour", hour(col("date"))) \
#    .withColumn("day", dayofmonth(col("date"))) \
#    .withColumn("month", month(col("date")))

query = df_silver.writeStream \
    .format("delta") \
    .outputMode("append") \
    .option("checkpointLocation", "lakehouse/checkpoints/air_quality_silver/") \
    .option("path", "lakehouse/silver/streaming/air_quality/") \
    .start()

query.awaitTermination()

```

```

GNU nano 7.2                                     LoadStreamingToSilver.py

.withColumn("time_json", from_json(col("data.time"), time_schema)) \
    .withColumn("date", to_timestamp(col("fetched_at"))) \
    .select(
        "date",
        col("iaqi_json.co.v").alias("CO"),
        col("iaqi_json.no2.v").alias("NO2"),
        col("iaqi_json.o3.v").alias("O3"),
        col("iaqi_json.pm10.v").alias("TSP"),
        col("iaqi_json.pm25.v").alias("PM25"),
        col("iaqi_json.so2.v").alias("SO2")
    ) \
    .dropDuplicates(["date", "CO", "NO2", "O3", "TSP", "PM25", "SO2"])

cols_to_cast = ["TSP", "PM25", "O3", "CO", "NO2", "SO2"]
for col_name in cols_to_cast:
    df_silver = df_silver.withColumn(col_name, coalesce(col(col_name).cast("double"), lit(0)))

df_silver = df_silver.withColumn(
    "AQI_Level",
    when(col("PM25") <= 12, "Good") \
    .when((col("PM25") >= 13) & (col("PM25") <= 35, "Average") \
    .when((col("PM25") >= 36) & (col("PM25") <= 55, "Moderate") \
    .otherwise("Unhealthy"))

#df_silver = df_silver.withColumn("hour", hour(col("date"))) \
#    .withColumn("day", dayofmonth(col("date"))) \
#    .withColumn("month", month(col("date")))

query = df_silver.writeStream \
    .format("delta") \
    .outputMode("append") \
    .option("checkpointLocation", "lakehouse/checkpoints/air_quality_silver/") \
    .option("path", "lakehouse/silver/streaming/air_quality/") \
    .start()

query.awaitTermination()

```

nano ~airflow/scripts2/MergeStreamingAndBatch.py

```

from pyspark.sql import SparkSession
spark = SparkSession.builder \
    .appName("Union Air Quality All") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog",
"org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .getOrCreate()
df_batch = spark.read.format("delta").load("lakehouse/silver/air_quality_cleaned/")

```

```

df_streaming =
spark.read.format("delta").load("lakehouse/silver/streaming/air_quality/")
cols = ["date", "CO", "NO2", "O3", "TSP", "PM25", "SO2", "AQI_Level"]

df_batch_sel = df_batch.select(*[c for c in cols if c in df_batch.columns])
df_streaming_sel = df_streaming.select(*[c for c in cols if c in
df_streaming.columns])
df_all = df_batch_sel.unionByName(df_streaming_sel)

df_all.write.format("delta") \
    .mode("overwrite") \
    .save("lakehouse/silver/air_quality_all/")

```

Đoạn code thực hiện việc lấy dữ liệu từ tầng silver gồm 2 phần là dữ liệu batch đã được xử lý và dữ liệu streaming đã được xử lý chọn ra các cột chung giữa 2 phần sau đó thực hiện việc hợp các dòng lại và lưu vào tầng silver

```

GNU nano 7.2                                     MergeStreamingAndBatch.py
from pyspark.sql import SparkSession
spark = SparkSession.builder \
    .appName("Union Air Quality All") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog", "org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .getOrCreate()
df_batch = spark.read.format("delta").load("lakehouse/silver/air_quality_cleaned/")
df_streaming = spark.read.format("delta").load("lakehouse/silver/streaming/air_quality/")
cols = ["date", "CO", "NO2", "O3", "TSP", "PM25", "SO2", "AQI_Level"]

df_batch_sel = df_batch.select(*[c for c in cols if c in df_batch.columns])
df_streaming_sel = df_streaming.select(*[c for c in cols if c in df_streaming.columns])
df_all = df_batch_sel.unionByName(df_streaming_sel)

df_all.write.format("delta") \
    .mode("overwrite") \
    .save("lakehouse/silver/air_quality_all/")

```

## GOLD

nano ~/airflow/scripts2/daily\_avg.py

```

#!/usr/bin/env python3
from pyspark.sql import SparkSession
from pyspark.sql.functions import avg, to_date, when, col

spark = SparkSession.builder \
    .appName("Gold Daily Average") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog",
"org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .getOrCreate()

df = spark.read.format("delta").load("lakehouse/silver/air_quality_all/")

df_avg = df.withColumn("date_only", to_date("date")) \
    .groupBy("date_only") \
    .agg(
        avg("PM25").alias("avg_PM25"),
#        avg("Temperature").alias("avg_temp"),
#        avg("Humidity").alias("avg_humidity")
    ).withColumn("AQI_Level", when(col("avg_PM25") <= 12, "Good") \
        .when((col("avg_PM25") > 12) & (col("avg_PM25") <= 35), "Average")

```

```

        .when((col("avg_PM25") > 35) & (col("avg_PM25") <= 55), "Moderate")
        .otherwise("Unhealthy"))

df_avg.write.format("delta").mode("overwrite").save("lakehouse/gold/air_quality_daily_avg")

```

đoạn code thực hiện lấy dữ liệu đã merge từ silver và tiến hành lấy dữ liệu phân loại PM25 theo ngày sau đó lưu vào lớp gold

```

GNU nano 7.2                               daily_avg.py
#!/usr/bin/env python3
from pyspark.sql import SparkSession
from pyspark.sql.functions import avg, to_date, when, col

spark = SparkSession.builder \
    .appName("Gold Daily Average") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog", "org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .getOrCreate()

df = spark.read.format("delta").load("lakehouse/silver/air_quality_all")

df_avg = df.withColumn("date_only", to_date("date")) \
    .groupBy("date_only") \
    .agg(
        avg("PM25").alias("avg_PM25"),
        #   avg("Temperature").alias("avg_temp"),
        #   avg("Humidity").alias("avg_humidity"))
    .withColumn("AQI_Level", when(col("avg_PM25") <= 12, "Good")
        .when((col("avg_PM25") > 12) & (col("avg_PM25") <= 35), "Average")
        .when((col("avg_PM25") > 35) & (col("avg_PM25") <= 55), "Moderate")
        .otherwise("Unhealthy"))

df_avg.write.format("delta").mode("overwrite").save("lakehouse/gold/air_quality_daily_avg")

```

nano ~/airflow/scripts2/hourly\_avg.py

```

#!/usr/bin/env python3
from pyspark.sql import SparkSession
from pyspark.sql.functions import avg, hour

spark = SparkSession.builder \
    .appName("Gold Hourly Average") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog",
"org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .getOrCreate()

df = spark.read.format("delta").load("lakehouse/silver/air_quality_all")

df_hour = df.withColumn("hour", hour("date")) \
    .groupBy("hour") \
    .agg(
        avg("PM25").alias("avg_PM25"),
        avg("CO").alias("avg_CO"),
        #   avg("Temperature").alias("avg_temp")
    ).orderBy("hour")

df_hour.write.format("delta").mode("overwrite").save("lakehouse/gold/air_quality_hourly_avg")

```

Đoạn code thực hiện lấy từ dữ liệu đã merge giá trị trung bình PM25 và giá trị trung bình CO theo giờ sau đó lưu vào lớp gold

```

GNU nano 7.2                                     hourly_avg.py
#!/usr/bin/env python3
from pyspark.sql import SparkSession
from pyspark.sql.functions import avg, hour

spark = SparkSession.builder \
    .appName("Gold Hourly Average") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog", "org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .getOrCreate()

df = spark.read.format("delta").load("lakehouse/silver/air_quality_all/")

df_hour = df.withColumn("hour", hour("date")) \
    .groupBy("hour") \
    .agg(
        avg("PM25").alias("avg_PM25"),
        avg("CO").alias("avg_CO"),
        #     avg("Temperature").alias("avg_temp")
    ).orderBy("hour")

df_hour.write.format("delta").mode("overwrite").save("lakehouse/gold/air_quality_hourly_avg")

```

nano ~/airflow/scripts2/monthly\_avg.py

```

#!/usr/bin/env python3
from pyspark.sql import SparkSession
from pyspark.sql.functions import avg, month

spark = SparkSession.builder \
    .appName("Gold Monthly Average") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog",
"org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .getOrCreate()

df = spark.read.format("delta").load("lakehouse/silver/air_quality_all/")

df_month = df.withColumn("month", month("date")) \
    .groupBy("month") \
    .agg(
        avg("PM25").alias("avg_PM25"),
        avg("CO").alias("avg_CO"),
        avg("O3").alias("avg_O3"),
        avg("SO2").alias("avg_SO2"),
#         avg("Temperature").alias("avg_temp"),
#         avg("Humidity").alias("avg_humidity")
    )

df_month.write.format("delta").mode("overwrite").save("lakehouse/gold/air_quality_monthly_avg")

```

Đoạn code thực hiện lấy dữ liệu đã merge ở silver các giá trị trung bình PM25, trung bình CO, trung bình O3, trung bình SO2 theo tháng

```

GNU nano 7.2   monthly_avg.py
#!/usr/bin/env python3
from pyspark.sql import SparkSession
from pyspark.sql.functions import avg, month

spark = SparkSession.builder \
    .appName("Gold Monthly Average") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog", "org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .getOrCreate()

df = spark.read.format("delta").load("lakehouse/silver/air_quality_all/")

df_month = df.withColumn("month", month("date")) \
    .groupBy("month") \
    .agg(
        avg("PM25").alias("avg_PM25"),
        avg("CO").alias("avg_CO"),
        avg("O3").alias("avg_O3"),
        avg("SO2").alias("avg_SO2"),
        #      avg("Temperature").alias("avg_temp"),
        #      avg("Humidity").alias("avg_humidity")
    )

df_month.write.format("delta").mode("overwrite").save("lakehouse/gold/air_quality_monthly_avg")

```

nano ~/airflow/scripts2/describe\_stats.py

```

#!/usr/bin/env python3
from pyspark.sql import SparkSession

spark = SparkSession.builder \
    .appName("Describe AQI Stats") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog",
"org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .getOrCreate()

df = spark.read.format("delta").load("lakehouse/silver/air_quality_all/")

df.describe(["PM25", "O3", "CO", "NO2", "SO2"]) \
    .write.format("delta").mode("overwrite") \
    .save("lakehouse/gold/air_quality_describe")

```

Đoạn code thực hiện thống kê mô tả từ dữ liệu đã merge ở silver các chỉ số PM25, O3, CO, NO2, SO2 và lưu vào lớp gold

```

GNU nano 7.2   describe_stats.py
#!/usr/bin/env python3
from pyspark.sql import SparkSession

spark = SparkSession.builder \
    .appName("Describe AQI Stats") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog", "org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .getOrCreate()

df = spark.read.format("delta").load("lakehouse/silver/air_quality_all/")

df.describe(["PM25", "O3", "CO", "NO2", "SO2"]) \
    .write.format("delta").mode("overwrite") \
    .save("lakehouse/gold/air_quality_describe")

```

nano ~/airflow/scripts2/aqi\_distribution.py

```

#!/usr/bin/env python3
from pyspark.sql import SparkSession

```

```

from pyspark.sql.functions import col

spark = SparkSession.builder \
    .appName("AQI Distribution") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog",
"org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .getOrCreate()

df = spark.read.format("delta").load("lakehouse/silver/air_quality_all/")

df.groupBy("AQI_Level").count().orderBy(col("count").desc()) \
    .write.format("delta").mode("overwrite") \
    .save("lakehouse/gold/air_quality_distribution")

```

Đoạn code thực hiện việc đếm tổng từng giá trị phân loại PM25 sắp xếp theo thứ tự giảm dần trong dữ liệu silver đã merge và lưu vào lớp gold

```

hadoopieuhao@hieu-hao:~/airflow/scripts2
GNU nano 7.2
#!/usr/bin/env python3
from pyspark.sql import SparkSession
from pyspark.sql.functions import col

spark = SparkSession.builder \
    .appName("AQI Distribution") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog", "org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .getOrCreate()

df = spark.read.format("delta").load("lakehouse/silver/air_quality_all/")

df.groupBy("AQI_Level").count().orderBy(col("count").desc()) \
    .write.format("delta").mode("overwrite") \
    .save("lakehouse/gold/air_quality_distribution")

```

nano ~/airflow/scripts2/worst\_days.py

```

#!/usr/bin/env python3
from pyspark.sql import SparkSession
from pyspark.sql.functions import to_date, avg, when, col

spark = SparkSession.builder \
    .appName("Worst AQI Days") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog",
"org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .getOrCreate()

df = spark.read.format("delta").load("lakehouse/silver/air_quality_all/")

df_gold = df.withColumn("date_only", to_date("date")) \
    .groupBy("date_only") \
    .agg(
        avg("PM25").alias("avg_PM25"),
#        avg("Temperature").alias("avg_temp"),
#        avg("Humidity").alias("avg_humidity")

```

```

    ).withColumn(
        "AQI_Level",
        when(col("avg_PM25") <= 12, "Good")
        .when((col("avg_PM25") > 12) & (col("avg_PM25") <= 35), "Average")
        .when((col("avg_PM25") > 35) & (col("avg_PM25") <= 55), "Moderate")
        .otherwise("Unhealthy")
    )

df_gold.orderBy(col("avg_PM25").desc()) \
    .select("date_only", "avg_PM25", "AQI_Level") \
    .write.format("delta").mode("overwrite") \
    .save("lakehouse/gold/air_quality_worst_days")

```

Đoạn code trên phân tích những ngày có chất lượng không khí kém nhất (dựa trên chỉ số PM2.5 trung bình mỗi ngày) và lưu kết quả vào lớp gold

```

hadoopieuhao@hieuham: ~ hadoopieuhao@hieuham: ~ hadoopieuhao@hieuham: ~ hadoopieuhao@hieuham: ~ hadoopieuhao@hieuham: ~ + worst_days.py
GNU nano 7.2
#!/usr/bin/env python3
from pyspark.sql import SparkSession
from pyspark.sql.functions import to_date, avg, when, col

spark = SparkSession.builder \
    .appName("Worst AQI Days") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog", "org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .getOrCreate()

df = spark.read.format("delta").load("lakehouse/silver/air_quality_all/")

df_gold = df.withColumn("date_only", to_date("date")) \
    .groupBy("date_only") \
    .agg(
        avg("PM25").alias("avg_PM25"),
        # avg("Temperature").alias("avg_temp"),
        # avg("Humidity").alias("avg_humidity")
    ).withColumn(
        "AQI_Level",
        when(col("avg_PM25") <= 12, "Good")
        .when((col("avg_PM25") > 12) & (col("avg_PM25") <= 35), "Average")
        .when((col("avg_PM25") > 35) & (col("avg_PM25") <= 55), "Moderate")
        .otherwise("Unhealthy")
    )

df_gold.orderBy(col("avg_PM25").desc()) \
    .select("date_only", "avg_PM25", "AQI_Level") \
    .write.format("delta").mode("overwrite") \
    .save("lakehouse/gold/air_quality_worst_days")

```

## Tạo script xuất dữ liệu về local

nano ~/airflow/scripts/export\_parquet

```

#!/bin/bash

mkdir -p
~/air_quality_export2/{daily,hourly,monthly,worst_days,distribution,describe}
chmod -R 777 ~/air_quality_export2
hdfs dfs -get -f /lakehouse/gold/air_quality_daily_avg/part-*.parquet
~/air_quality_export2/daily/
hdfs dfs -get -f /lakehouse/gold/air_quality_hourly_avg/part-*.parquet
~/air_quality_export2/hourly/
hdfs dfs -get -f /lakehouse/gold/air_quality_monthly_avg/part-*.parquet
~/air_quality_export2/monthly/

```

```

hdfs dfs -get -f /lakehouse/gold/air_quality_worst_days/part-*.parquet
~/air_quality_export2/worst_days/
hdfs dfs -get -f /lakehouse/gold/air_quality_distribution/part-*.parquet
~/air_quality_export2/distribution/
hdfs dfs -get -f /lakehouse/gold/air_quality_describe/part-*.parquet
~/air_quality_export2/describe/

```

đoạn code thực hiện lấy dữ liệu từ lớp gold trong hdfs và chuyển về trong user hadoop ở folder air\_quality\_export2

```

hadoop:hieuhaohieuhao@hieuhaohieuhao-master: ~/airflow/scripts2
GNU nano 7.2
#!/bin/bash
export_parquet

mkdir -p ~/air_quality_export2/{daily,hourly,monthly,worst_days,distribution,describe}
chmod -R 777 ~/air_quality_export2
hdfs dfs -get -f lakehouse/gold/air_quality_daily_avg/part-*.parquet ~/air_quality_export2/daily/
hdfs dfs -get -f lakehouse/gold/air_quality_hourly_avg/part-*.parquet ~/air_quality_export2/hourly/
hdfs dfs -get -f lakehouse/gold/air_quality_monthly_avg/part-*.parquet ~/air_quality_export2/monthly/

hdfs dfs -get -f lakehouse/gold/air_quality_worst_days/part-*.parquet ~/air_quality_export2/worst_days/
hdfs dfs -get -f lakehouse/gold/air_quality_distribution/part-*.parquet ~/air_quality_export2/distribution/
hdfs dfs -get -f lakehouse/gold/air_quality_describe/part-*.parquet ~/air_quality_export2/describe/

```

Cấp quyền cho tất cả thư mục

chmod -R 777 ~/airflow/scripts2

## Tạo file DAG

nano ~/airflow/dags/bronze\_to\_platinum\_pipeline2.py

```

from airflow import DAG
from airflow.operators.bash import BashOperator
from datetime import datetime, timedelta

default_args = {
    'owner': 'airflow',
    'start_date': datetime(2025, 5, 24),
    'retries': 2,
    'retry_delay': timedelta(minutes=5),
    'execution_timeout': timedelta(minutes=30),
}

with DAG(
    dag_id='bronze_to_platinum_pipeline2',
    default_args=default_args,
    schedule_interval='@daily',
    catchup=False,
    description='Pipeline: Bronze → Silver → Gold → Platinum using Spark'
) as dag:

    delta_submit = (
        "/usr/local/bin/spark-submit"

```

```
    " --packages io.delta:delta-spark_2.12:3.0.0"
    " --conf \"spark.sql.extensions=io.delta.sql.DeltaSparkSessionExtension\" "
    " --conf
\"spark.sql.catalog.spark_catalog=org.apache.spark.sql.delta.catalog.DeltaCatalog\" "
)
bronze_ingest = BashOperator(
    task_id='bronze_ingest',
    bash_command=delta_submit +
"/home/hadoophieuhaao/airflow/scripts2/bronze_ingest.py"
)

silver_cleaning = BashOperator(
    task_id='silver_cleaning',
    bash_command=delta_submit +
"/home/hadoophieuhaao/airflow/scripts2/silver_cleaning.py"
)

streaming_to_silver = BashOperator(
    task_id='streaming_to_silver',
    bash_command=delta_submit +
"/home/hadoophieuhaao/airflow/scripts2/LoadStreamingToSilver.py"
)

merge_data_silver = BashOperator(
    task_id='merge_data_silver',
    bash_command=delta_submit +
"/home/hadoophieuhaao/airflow/scripts2/MergeStreamingAndBatch.py"
)

gold_describe = BashOperator(
    task_id='gold_describe_stats',
    bash_command=delta_submit +
"/home/hadoophieuhaao/airflow/scripts2/describe_stats.py"
)

gold_distribution = BashOperator(
    task_id='gold_aqi_distribution',
    bash_command=delta_submit +
"/home/hadoophieuhaao/airflow/scripts2/aqi_distribution.py"
)

gold_worst_days = BashOperator(
    task_id='gold_worst_days',
    bash_command=delta_submit +
"/home/hadoophieuhaao/airflow/scripts2/worst_days.py"
)

gold_daily = BashOperator(
    task_id='gold_daily_avg',
    bash_command=delta_submit +
"/home/hadoophieuhaao/airflow/scripts2/daily_avg.py"
)

gold_hourly = BashOperator(
    task_id='gold_hourly_avg',
```

```

        bash_command=delta_submit +
"/home/hadoopieuhao/airflow/scripts2/hourly_avg.py"
)

gold_monthly = BashOperator(
    task_id='gold_monthly_avg',
    bash_command=delta_submit +
"/home/hadoopieuhao/airflow/scripts2/monthly_avg.py"
)

export_parquet = BashOperator(
    task_id='export_parquet_files',
    bash_command="/home/hadoopieuhao/airflow/scripts2/export_parquet"
)

# Task dependencies
[bronze_ingest,streaming_to_silver ] >> silver_cleaning
silver_cleaning >> merge_data_silver
merge_data_silver >> [
    gold_describe,
    gold_distribution,
    gold_worst_days,
    gold_daily,
    gold_hourly,
    gold_monthly
] >> export_parquet

```

Đoạn code trên hành xây dựng 1 pipeline bao gồm các task mỗi task sẽ thực hiện việc spark-submit các đoạn code ở trong thư mục scripts2 các bước thực hiện là bronze\_ingest, streaming\_to\_silver đến silver\_cleaning sau đó là merge\_data\_silver tiếp theo là các task gold và cuối cùng là export\_parquet

```
[hadoopieuhao@hieuhao-master: ~/airflow] GNU nano 7.2 dags/bronze_to_platinum_pipeline2.py
from airflow import DAG
from airflow.operators.bash import BashOperator
from datetime import datetime, timedelta

default_args = {
    'owner': 'airflow',
    'start_date': datetime(2025, 5, 24),
    'retries': 2,
    'retry_delay': timedelta(minutes=5),
    'execution_timeout': timedelta(minutes=30),
}

with DAG(
    dag_id='bronze_to_platinum_pipeline2',
    default_args=default_args,
    schedule_interval='@daily',
    catchup=False,
    description='Pipeline: Bronze → Silver → Gold → Platinum using Spark'
) as dag:

    delta_submit = (
        "/usr/local/bin/spark-submit"
        " --packages io.delta:delta-spark_2.12:3.0.0"
        " --conf \"spark.sql.extensions=io.delta.sql.DeltaSparkSessionExtension\" "
        " --conf \"spark.sql.catalog.spark_catalog=org.apache.spark.sql.delta.catalog.DeltaCatalog\" "
    )
    bronze_ingest = BashOperator(
        task_id='bronze_ingest',
        bash_command=delta_submit + " /home/hadoopieuhao/airflow/scripts2/bronze_ingest.py"
    )

    silver_cleaning = BashOperator(
        task_id='silver_cleaning',
        bash_command=delta_submit + " /home/hadoopieuhao/airflow/scripts2/silver_cleaning.py"
    )

    streaming_to_silver = BashOperator(
        task_id='streaming_to_silver',
        bash_command=delta_submit + " /home/hadoopieuhao/airflow/scripts2/LoadStreamingToSilver.py"
    )

    merge_data_silver = BashOperator(
        task_id='merge_data_silver',
        bash_command=delta_submit + " /home/hadoopieuhao/airflow/scripts2/MergeStreamingAndBatch.py"
```

```
GNU nano 7.2                                     dags/bronze_to_platinum_pipeline2.py
task_id='silver_cleaning',
bash_command=delta_submit + "/home/hadoophieuhanh/airflow/scripts2/silver_cleaning.py"
)

streaming_to_silver = BashOperator(
    task_id='streaming_to_silver',
    bash_command=delta_submit + "/home/hadoophieuhanh/airflow/scripts2/LoadStreamingToSilver.py"
)

merge_data_silver = BashOperator(
    task_id='merge_data_silver',
    bash_command=delta_submit + "/home/hadoophieuhanh/airflow/scripts2/MergeStreamingAndBatch.py"
)

gold_describe = BashOperator(
    task_id='gold_describe_stats',
    bash_command=delta_submit + "/home/hadoophieuhanh/airflow/scripts2/describe_stats.py"
)

gold_distribution = BashOperator(
    task_id='gold_aqi_distribution',
    bash_command=delta_submit + "/home/hadoophieuhanh/airflow/scripts2/aqi_distribution.py"
)

gold_worst_days = BashOperator(
    task_id='gold_worst_days',
    bash_command=delta_submit + "/home/hadoophieuhanh/airflow/scripts2/worst_days.py"
)

gold_daily = BashOperator(
    task_id='gold_daily_avg',
    bash_command=delta_submit + "/home/hadoophieuhanh/airflow/scripts2/daily_avg.py"
)

gold_hourly = BashOperator(
    task_id='gold_hourly_avg',
    bash_command=delta_submit + "/home/hadoophieuhanh/airflow/scripts2/hourly_avg.py"
)

gold_monthly = BashOperator(
    task_id='gold_monthly_avg',
    bash_command=delta_submit + "/home/hadoophieuhanh/airflow/scripts2/monthly_avg.py"
)

export_parquet = BashOperator(
```

```

GNU nano 7.2                                     dags/bronze_to_platinum_pipeline2.py
    bash_command=delta_submit + "/home/hadoophieuhao/airflow/scripts2/describe_stats.py"
)
gold_distribution = BashOperator(
    task_id='gold_aqi_distribution',
    bash_command=delta_submit + "/home/hadoophieuhao/airflow/scripts2/aqi_distribution.py"
)
gold_worst_days = BashOperator(
    task_id='gold_worst_days',
    bash_command=delta_submit + "/home/hadoophieuhao/airflow/scripts2/worst_days.py"
)
gold_daily = BashOperator(
    task_id='gold_daily_avg',
    bash_command=delta_submit + "/home/hadoophieuhao/airflow/scripts2/daily_avg.py"
)
gold_hourly = BashOperator(
    task_id='gold_hourly_avg',
    bash_command=delta_submit + "/home/hadoophieuhao/airflow/scripts2/hourly_avg.py"
)
gold_monthly = BashOperator(
    task_id='gold_monthly_avg',
    bash_command=delta_submit + "/home/hadoophieuhao/airflow/scripts2/monthly_avg.py"
)
export_parquet = BashOperator(
    task_id='export_parquet_files',
    bash_command="/home/hadoophieuhao/airflow/scripts2/export_parquet"
)
# Task dependencies
[bronze_ingest,streaming_to_silver ] >> silver_cleaning
silver_cleaning >> merge_data_silver
merge_data_silver >> [
    gold_describe,
    gold_distribution,
    gold_worst_days,
    gold_daily,
    gold_hourly,
    gold_monthly
] >> export_parquet

```

Chay o 2 terminal khac nhau

airflow scheduler &

airflow webserver -p 8080 &

## Chạy zookeeper

## Mỗi phần mở 1 terminal

```
source myenv/bin/activate
```

```
cd kafka
```

```
bin/zookeeper-server-start.sh config/zookeeper.properties
```

```
[ca] hadoophieuhanh@hieuhanh-master: ~/kafka
hadoophieuhanh@hieuhanh-master:~$ source myenv/bin/activate
(myenv) hadoophieuhanh@hieuhanh-master:~$ cd kafka/
(myenv) hadoophieuhanh@hieuhanh-master:~/kafka$ bin/zookeeper-server-start.sh config/zookeeper.properties
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/hadoop/hieuhanh/hive/lib/log4j-slf4j-impl-2.18.0.jar!/org/slf4j/impl/StaticLoggerB
inder.class]
SLF4J: Found binding in [jar:file:/home/hadoop/hieuhanh/kafka/libs/slf4j-log4j12-1.7.30.jar!/org/slf4j/impl/StaticLoggerBi
nder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
[2025-05-25 23:36:33,697] INFO Reading configuration from: config/zookeeper.properties (org.apache.zookeeper.server.quor
um.QuorumPeerConfig)
[2025-05-25 23:36:33,737] WARN config/zookeeper.properties is relative. Prepend ./ to indicate that you're sure! (org.ap
ache.zookeeper.server.quorum.QuorumPeerConfig)
[2025-05-25 23:36:33,742] INFO clientPortAddress is 0.0.0.0:2181 (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2025-05-25 23:36:33,743] INFO secureClientPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2025-05-25 23:36:33,743] INFO observerMasterPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2025-05-25 23:36:33,743] INFO metricsProvider.className is org.apache.zookeeper.metrics.impl.DefaultMetricsProvider (or
g.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2025-05-25 23:36:33,750] INFO autopurge.snapRetainCount set to 3 (org.apache.zookeeper.server.DatadirCleanupManager)
[2025-05-25 23:36:33,751] INFO autopurge.purgeInterval set to 0 (org.apache.zookeeper.server.DatadirCleanupManager)
[2025-05-25 23:36:33,752] INFO Purge task is not scheduled. (org.apache.zookeeper.server.DatadirCleanupManager)
[2025-05-25 23:36:33,752] WARN Either no config or no quorum defined in config, running in standalone mode (org.apache.z
ookeeper.server.quorum.QuorumPeerMain)
[2025-05-25 23:36:33,780] INFO Log4j 1.2 jmx support found and enabled. (org.apache.zookeeper.jmx.ManagedUtil)
[2025-05-25 23:36:33,802] DEBUG preRegister called. Server=com.sun.jmx.mbeanserver.JmxMBeanServer@d716361, name=log4j:lo
gger=root (root)
[2025-05-25 23:36:33,807] INFO Reading configuration from: config/zookeeper.properties (org.apache.zookeeper.server.quor
um.QuorumPeerConfig)
[2025-05-25 23:36:33,809] WARN config/zookeeper.properties is relative. Prepend ./ to indicate that you're sure! (org.ap
```

Chạy kafka

source myenv/bin/activate

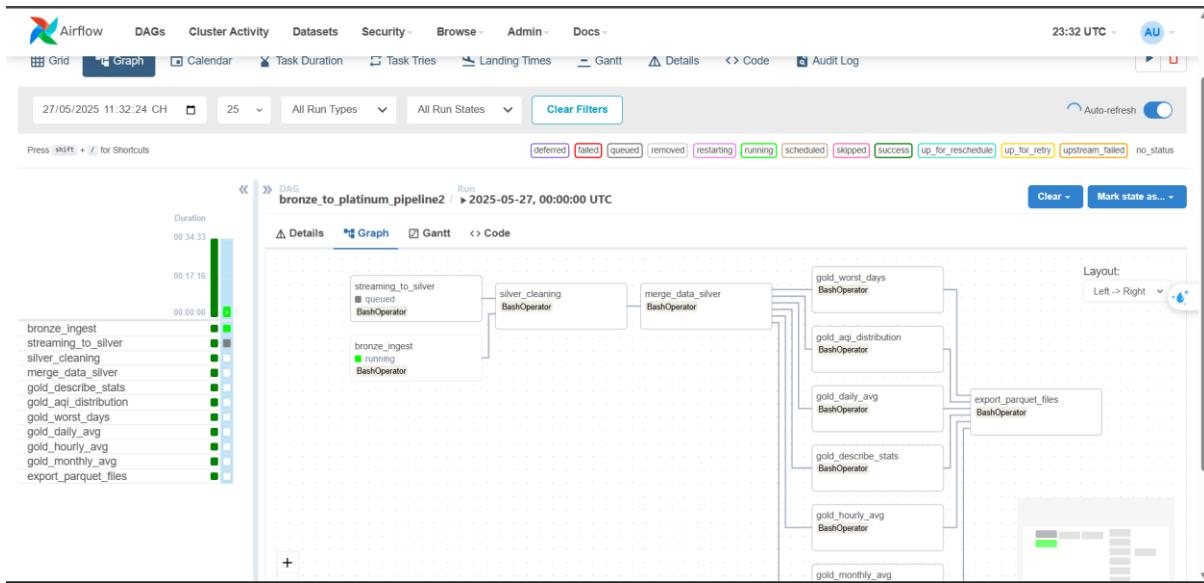
cd kafka

./run\_kafka.sh

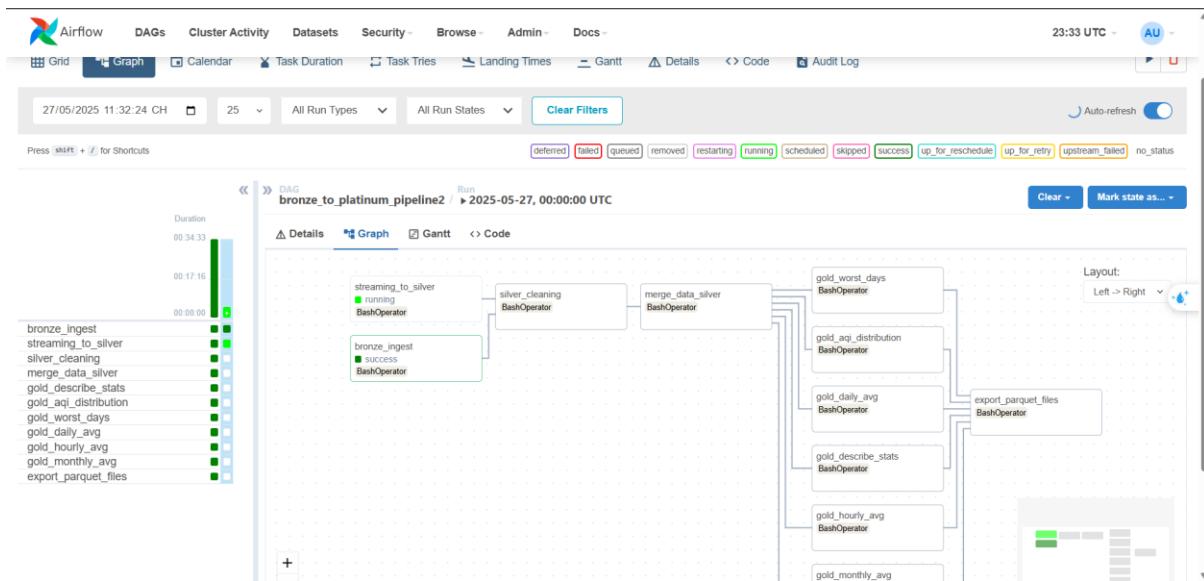
```
[ca] hadoophieuhanh@hieuhanh-master: ~/kafka
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

Last login: Sun May 25 23:34:47 2025 from 192.168.226.1
hadoophieuhanh@hieuhanh-master:~$ source myenv/bin/activate
(myenv) hadoophieuhanh@hieuhanh-master:~$ cd kafka/
(myenv) hadoophieuhanh@hieuhanh-master:~/kafka$ ./run_kafka.sh
[2025-05-25 23:36:49,779] INFO Registered kafka:type=kafka.Log4jController MBean (kafka.utils.Log4jControllerRegistratio
n$)
[2025-05-25 23:36:50,878] INFO Setting -D jdk.tls.rejectClientInitiatedRenegotiation=true to disable client-initiated TL
S renegotiation (org.apache.zookeeper.common.X509Util)
[2025-05-25 23:36:51,064] INFO Registered signal handlers for TERM, INT, HUP (org.apache.kafka.common.utils.LoggingSigna
lHandler)
[2025-05-25 23:36:51,073] INFO starting (kafka.server.KafkaServer)
[2025-05-25 23:36:51,074] INFO Connecting to zookeeper on localhost:2181 (kafka.server.KafkaServer)
[2025-05-25 23:36:51,340] INFO [ZooKeeperClient Kafka server] Initializing a new session to localhost:2181. (kafka.zooke
per.ZooKeeperClient)
[2025-05-25 23:36:51,361] INFO Client environment:zookeeper.version=3.5.9-83df9301aa5c2a5d284a9940177808c01bc35cef, buil
t on 01/06/2021 20:03 GMT (org.apache.zookeeper.ZooKeeper)
[2025-05-25 23:36:51,361] INFO Client environment:host.name=hieuhanh-master (org.apache.zookeeper.ZooKeeper)
[2025-05-25 23:36:51,361] INFO Client environment:java.version=1.8.0_452 (org.apache.zookeeper.ZooKeeper)
[2025-05-25 23:36:51,362] INFO Client environment:java.vendor=Private Build (org.apache.zookeeper.ZooKeeper)
[2025-05-25 23:36:51,362] INFO Client environment:java.home=/usr/lib/jvm/java-8-openjdk-amd64/jre (org.apache.zookeeper.ZooKeeper)
[2025-05-25 23:36:51,362] INFO Client environment:java.class.path=/home/hadoop/hieuhanh/kafka/bin/.. /libs/activation-1.1.1
.jar:/home/hadoop/hieuhanh/kafka/bin/.. /libs/aopalliance-repackaged-2.6.1.jar:/home/hadoop/hieuhanh/kafka/bin/.. /libs/argpar
se4j-0.7.0.jar:/home/hadoop/hieuhanh/kafka/bin/.. /libs/audience-annotations-0.5.0.jar:/home/hadoop/hieuhanh/kafka/bin/.. /lib
s/commons-cli-1.4.jar:/home/hadoop/hieuhanh/kafka/bin/.. /libs/commons-lang3-3.8.1.jar:/home/hadoop/hieuhanh/kafka/bin/.. /lib
s/connect-api-2.8.2.jar:/home/hadoop/hieuhanh/kafka/bin/.. /libs/connect-basic-auth-extension-2.8.2.jar:/home/hadoop/hieuhanh
```

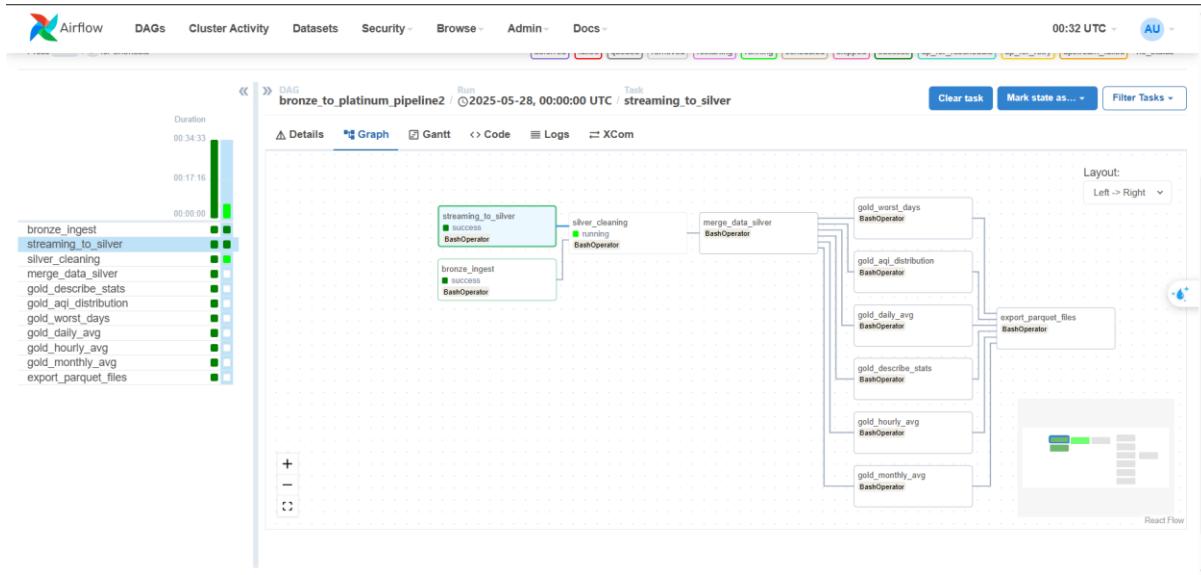
## Tiến hành chạy dag



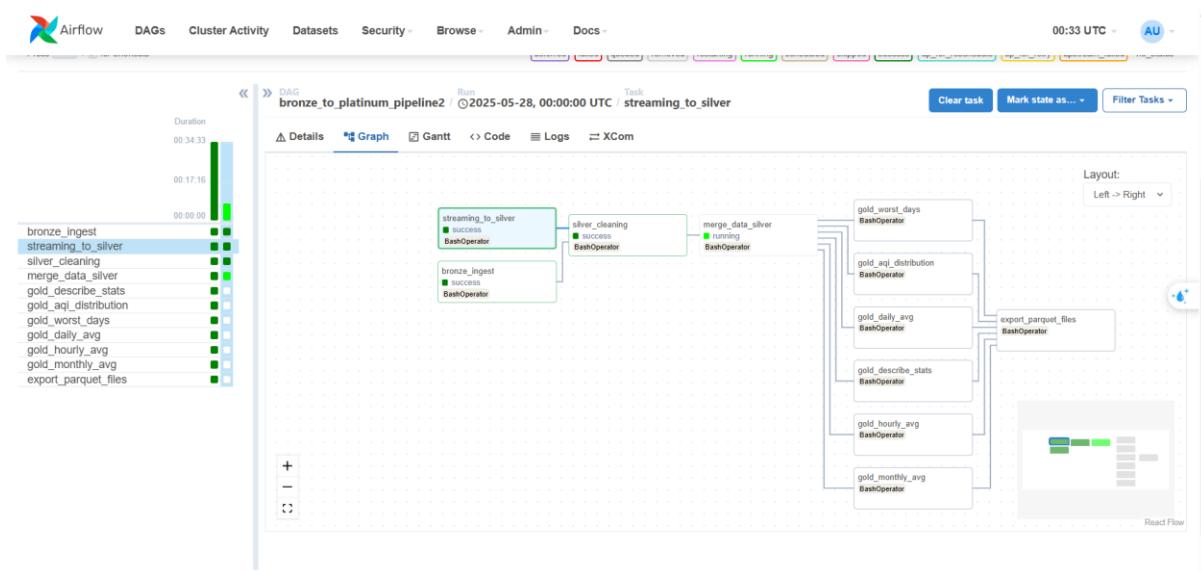
Khi đến streaming\_to\_silver sẽ bắt đầu việc lấy dữ liệu lấy xong sẽ dừng việc lấy để tránh tốn dung lượng bộ nhớ cho spark



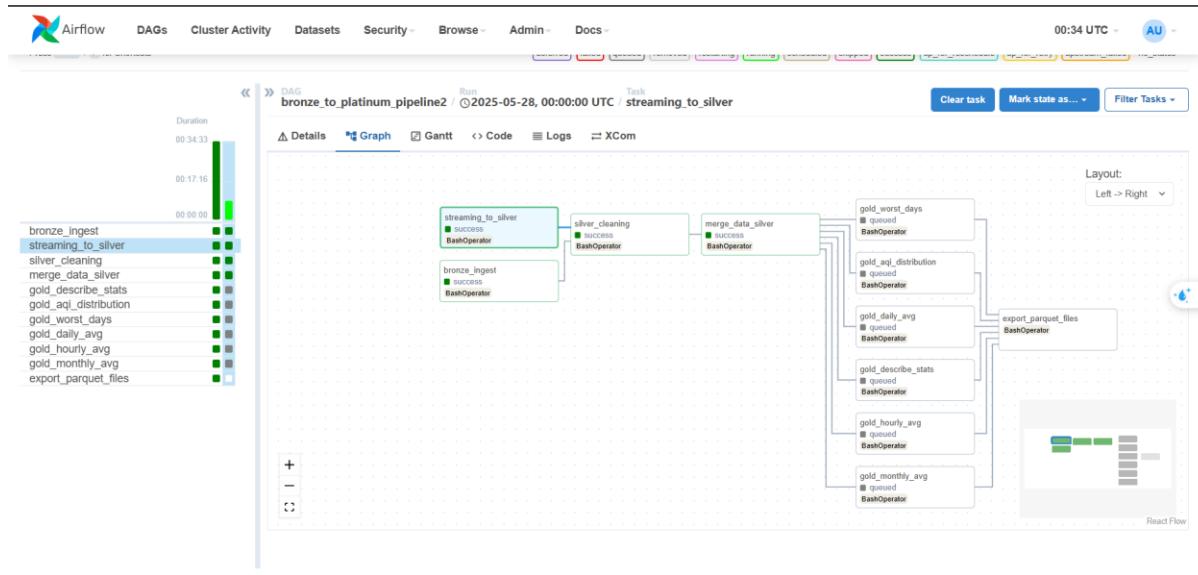
Hoàn thành việc xử lý data streaming đưa vào silver



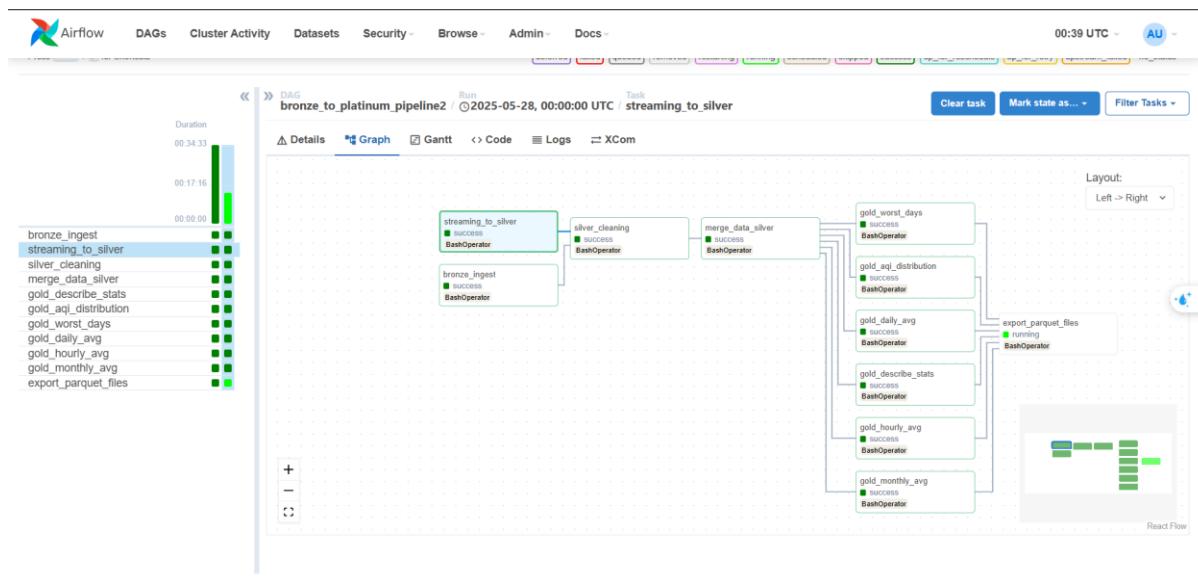
Đến phần xử lý data từ bronze vào silver



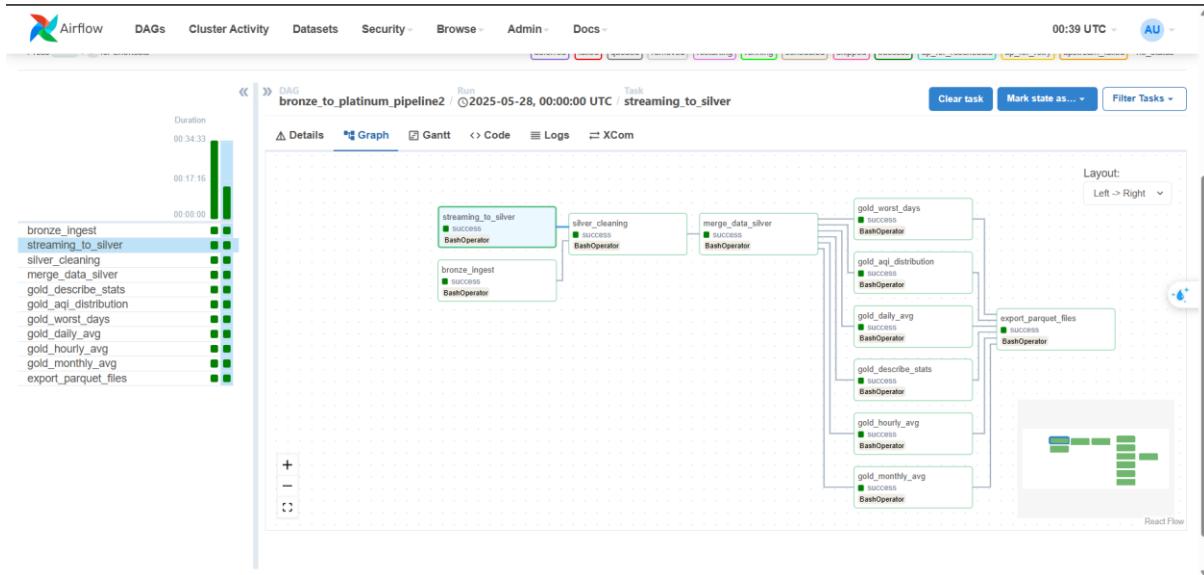
Tiếp theo merge data trong silver



Sau đó bắt đầu xử lý dữ liệu đưa vào gold



Cuối cùng đưa dữ liệu từ gold về user Hadoop



## Kiểm tra lại trong user Hadoop

```

hadoopieuhao@hieuhaomaster:~$ ls
airflow          hivel           myenv          spark-jars
airflow_venv     hive-env.xml   output          spark-rdd
air_quality.csv  hive-site.xml  passwd         spark-streaming
air_quality_export  input          Permission    spark-wordcount
air_quality_export2 json          jdbc          testdelta.py
azure            input-1.txt    pig           testminio
bronze_to_platinum_pipeline2.py json_to_delta.py process_air_quality_data_fixed.py
dataset          kafka          project        test.py
deltatable.py    kafka_2.12-2.8.2.tgz retail        test.sh
derby.log        lakehouse      sample.txt   testtmp.py
directory        Load_TienXL.py  scala-2.12.15 units
Exercise39       metastore_db  scala-2.13.16  units.jar
Gold.py          minio          scripts2    venv
hadoop          minio-bin      spark         spark-3.5.5-bin-hadoop3.tgz
hadoop-core-1.2.1.jar minio-data
hive             minio          spark
hadoopieuhao@hieuhaomaster:~$ ls -l air_quality_export2
total 24
drwxrwxrwx 2 hadoopieuhao hadoopieuhao 4096 May 28 00:39 daily
drwxrwxrwx 2 hadoopieuhao hadoopieuhao 4096 May 28 00:40 describe
drwxrwxrwx 2 hadoopieuhao hadoopieuhao 4096 May 28 00:40 distribution
drwxrwxrwx 2 hadoopieuhao hadoopieuhao 4096 May 28 00:39 hourly
drwxrwxrwx 2 hadoopieuhao hadoopieuhao 4096 May 28 00:39 monthly
drwxrwxrwx 2 hadoopieuhao hadoopieuhao 4096 May 28 00:39 worst_days
hadoopieuhao@hieuhaomaster:~$
```

## 2. Lấy dữ liệu từ API đưa về hdfs

Tiến hành việc lấy dữ liệu từ api

```

import requests
import json
import time
from kafka import KafkaProducer
from datetime import datetime

API_TOKEN = "f64cf8310728156d5ff964ce7c6ca400894535d"
API_URL =
f"https://api.waqi.info/feed/@8767/?token=f64cf8310728156d5ff964ce7c6ca400894535d"
```

```

KAFKA_BROKER = "localhost:9092"
TOPIC = "air-quality"

producer = KafkaProducer(
    bootstrap_servers=KAFKA_BROKER,
    value_serializer=lambda v: json.dumps(v).encode("utf-8")
)

def fetch_data():
    response = requests.get(API_URL)
    if response.status_code == 200:
        data = response.json()
        if data["status"] == "ok":
            data["fetched_at"] = datetime.now().isoformat()
            return data
    return None

while True:
    data = fetch_data()
    if data:
        producer.send(TOPIC, value=data)
        print("Sent data to Kafka:", data["fetched_at"])
    time.sleep(3600)

```

Đoạn code thực hiện lấy data từ API mỗi 1 tiếng

```

GNU nano 7.2
air_quality_kafka_producer.py
import requests
import json
import time
from kafka import KafkaProducer
from datetime import datetime

API_TOKEN = "f64cf8310728156d5ff964ce7c6ca400894535d"
API_URL = f"https://api.waqi.info/feed/@8767/?token=f64cf8310728156d5ff964ce7c6ca400894535d"

KAFKA_BROKER = "localhost:9092"
TOPIC = "air-quality"

producer = KafkaProducer(
    bootstrap_servers=KAFKA_BROKER,
    value_serializer=lambda v: json.dumps(v).encode("utf-8")
)

def fetch_data():
    response = requests.get(API_URL)
    if response.status_code == 200:
        data = response.json()
        if data["status"] == "ok":
            data["fetched_at"] = datetime.now().isoformat()
            return data
    return None

while True:
    data = fetch_data()
    if data:
        producer.send(TOPIC, value=data)
        print("Sent data to Kafka:", data["fetched_at"])
    time.sleep(3600)

```

[ Wrote 32 lines ]

GNU nano 7.2

^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location ^U Undo ^A Set Mark ^I To Bracket  
^X Exit ^R Read File ^N Replace ^U Paste ^J Justify ^/ Go To Line ^E Redo ^- Copy ^Q Where Was

Dùng lệnh bên dưới để thực hiện việc lấy data

python3 air\_quality\_kafka\_producer.py

```

Microsoft Windows [Version 10.0.26100.4061]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ADMIN>ssh hadoophieuao@192.168.226.11
hadoophieuao@192.168.226.11's password:
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-60-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/pro

System information as of Tue May 27 11:16:49 PM UTC 2025

System load: 0.6          Processes: 250
Usage of /: 82.3% of 17.83GB   Users logged in: 1
Memory usage: 48%           IPv4 address for ens3: 192.168.226.11
Swap usage: 20%

Expanded Security Maintenance for Applications is not enabled.

196 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

Last login: Tue May 27 22:25:38 2025 from 192.168.226.1
hadoopiehao@hieuao:~$ source myenv/bin/activate
(myenv) hadoopiehao@hieuao:~$ cd project/
(myenv) hadoopiehao@hieuao:~/project$ python3 air_quality_kafka_producer.py
Sent data to Kafka: 2025-05-27T23:19:56.204755

```

Đây dữ liệu về hdfs

```

from pyspark.sql import SparkSession
from pyspark.sql.functions import from_json, col
from pyspark.sql.types import *

spark = SparkSession.builder \
    .appName("KafkaToHDFS") \
    .config("spark.sql.extensions", "") \
    .getOrCreate()
print("spark.sql.catalog.spark_catalog =", 
spark.conf.get("spark.sql.catalog.spark_catalog", "Not Set"))

df_raw = spark.readStream \
    .format("kafka") \
    .option("kafka.bootstrap.servers", "localhost:9092") \
    .option("subscribe", "air-quality") \
    .option("startingOffsets", "latest") \
    .option("failOnDataLoss", "false") \
    .load()

df_json = df_raw.selectExpr("CAST(value AS STRING) as json_str")

schema = StructType().add("data", MapType(StringType(),
StringType())).add("fetched_at", StringType())

df_parsed = df_json.select(from_json(col("json_str"), schema).alias("parsed"))
df_result = df_parsed.select("parsed.*")

df_result.writeStream \
    .format("json") \
    .option("path", "lakehouse/bronze/streaming/air_quality/") \
    .option("checkpointLocation", "lakehouse/checkpoints/air_quality/") \
    .outputMode("append") \

```

```
.start() \
.awaitTermination()
```

```
hadoopieuhao@hieu-hao-MX700: ~/project
GNU nano 7.2   spark_kafka_to_hdfs.py
from pyspark.sql import SparkSession
from pyspark.sql.functions import from_json, col
from pyspark.sql.types import *

spark = SparkSession.builder \
    .appName("KafkaToHDFS") \
    .config("spark.sql.extensions", "") \
    .getOrCreate()
print("spark.sql.catalog.spark_catalog =", spark.conf.get("spark.sql.catalog.spark_catalog", "Not Set"))

df_raw = spark.readStream \
    .format("kafka") \
    .option("kafka.bootstrap.servers", "localhost:9092") \
    .option("subscribe", "air-quality") \
    .option("startingOffsets", "latest") \
    .option("failOnDataLoss", "false") \
    .load()

df_json = df_raw.selectExpr("CAST(value AS STRING) as json_str")

schema = StructType().add("data", MapType(StringType(), StringType())).add("fetched_at", StringType())

df_parsed = df_json.select(from_json(col("json_str"), schema).alias("parsed"))
df_result = df_parsed.select("parsed.*")

df_result.writeStream \
    .format("json") \
    .option("path", "lakehouse/bronze/streaming/air_quality/") \
    .option("checkpointLocation", "lakehouse/checkpoints/air_quality/") \
    .outputMode("append") \
    .start() \
    .awaitTermination()
```

Dùng lệnh dưới để thực hiện việc đẩy dữ liệu về hdfs

```
spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.12:3.5.3
spark_kafka_to_hdfs.py
```

```

[hadoopieuhao@hieuhanh-master:~/project]$ ./spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.12:3.5.3 spark_kafka_to_hdfs.py
:: loading settings :: url = jar:file:/home/hadoopieuhao/spark/jars/ivy-2.5.1.jar!/org/apache/ivy/core/settings/ivysettings.xml
Ivy Default Cache set to: /home/hadoopieuhao/.ivy2/cache
The jars for the packages stored in: /home/hadoopieuhao/.ivy2/jars
org.apache.spark#spark-sql-kafka-0-10_2.12 added as a dependency
:: resolving dependencies :: org.apache.spark#spark-submit-parent-dd2d4101-46ed-471e-8df0-c7b13347c5e8;1.0
  confs: [default]
    found org.apache.spark#spark-sql-kafka-0-10_2.12;3.5.3 in central
    found org.apache.spark#spark-token-provider-kafka-0-10_2.12;3.5.3 in central
    found org.apache.kafka:kafka-clients;3.4.1 in central
    found org.lz4#lz4-java;1.8.0 in central
    found org.xerial.snappy#snappy-java;1.1.10.5 in central
    found org.slf4j#slf4j-api;2.0.7 in central
    found org.apache.hadoop#hadoop-client-runtime;3.3.4 in central
    found org.apache.hadoop#hadoop-client-api;3.3.4 in central
    found commons-logging#commons-logging;1.1.3 in central
    found com.google.code.findbugs#jsr305;3.0.0 in central
    found org.apache.commons#commons-pool2;2.11.1 in central
:: resolution report :: resolve 920ms :: artifacts dl 41ms
  :: modules in use:
    com.google.code.findbugs#jsr305;3.0.0 from central in [default]
    commons-logging#commons-logging;1.1.3 from central in [default]
    org.apache.commons#commons-pool2;2.11.1 from central in [default]
    org.apache.hadoop#hadoop-client-api;3.3.4 from central in [default]
    org.apache.hadoop#hadoop-client-runtime;3.3.4 from central in [default]
    org.apache.kafka:kafka-clients;3.4.1 from central in [default]
    org.apache.spark#spark-sql-kafka-0-10_2.12;3.5.3 from central in [default]
    org.apache.spark#spark-token-provider-kafka-0-10_2.12;3.5.3 from central in [default]
    org.lz4#lz4-java;1.8.0 from central in [default]
    org.slf4j#slf4j-api;2.0.7 from central in [default]
    org.xerial.snappy#snappy-java;1.1.10.5 from central in [default]
-----
conf	modules		artifacts				
conf	number	search	downloaded	evicted		number	downloaded
default	11	0	0	0		11	0
-----
:: retrieving :: org.apache.spark#spark-submit-parent-dd2d4101-46ed-471e-8df0-c7b13347c5e8
  confs: [default]
  0 artifacts copied, 11 already retrieved (0kB/17ms)
25/05/26 03:33:19 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-Java classes where applicable

```

```

[hadoopieuhao@hieuhanh-master:~/project]$ ./spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.12:3.5.3 spark_kafka_to_hdfs.py
}
25/05/26 03:33:54 INFO CheckpointFileManager: Writing atomically to hdfs://hieuhanh-master:9000/user/hadoopieuhao/lakehouse/checkpoints/air_quality/offsets/24 using temp file hdfs://hieuhanh-master:9000/user/hadoopieuhao/lakehouse/checkpoints/air_quality/offsets/.24.b56b89e2-b7ac-42c8-a899-9ffdb4fb044a.tmp
25/05/26 03:33:54 INFO CheckpointFileManager: Renamed temp file hdfs://hieuhanh-master:9000/user/hadoopieuhao/lakehouse/checkpoints/air_quality/offsets/.24.b56b89e2-b7ac-42c8-a899-9ffdb4fb044a.tmp to hdfs://hieuhanh-master:9000/user/hadoopieuhao/lakehouse/checkpoints/air_quality/offsets/24
25/05/26 03:33:54 INFO MicroBatchExecution: Committed offsets for batch 24. Metadata OffsetSeqMetadata(0,1748230434541,Map(spark.sql.streaming.stateStore.providerClass -> org.apache.spark.execution.streaming.state.HDFSBackedStateStoreProvider, spark.sql.streaming.join.stateFormatVersion -> 2, spark.sql.streaming.stateStore.compression.codec -> lz4, spark.sql.streaming.stateStore.rocksdb.formatVersion -> 5, spark.sql.streaming.statefulOperator.useStrictDistribution -> true, spark.sql.streaming.flatMapGroupsWithState.stateFormatVersion -> 2, spark.sql.streaming.multipleWatermarkPolicy -> min, spark.sql.streaming.aggregation.stateFormatVersion -> 2, spark.sql.shuffle.partitions -> 200))
25/05/26 03:33:54 INFO KafkaOffsetReaderAdmin: Partitions added: Map()
25/05/26 03:33:54 INFO KafkaOffsetReaderAdmin: Partitions added: Map()
25/05/26 03:33:54 INFO KafkaOffsetReaderAdmin: Partitions added: Map()
25/05/26 03:33:54 INFO FileStreamSinkLog: BatchIds found from listing: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 23
25/05/26 03:33:54 INFO SparkContext: Starting job: start at NativeMethodAccessorImpl.java:0
25/05/26 03:33:54 INFO DAGScheduler: Got job 1 (start at NativeMethodAccessorImpl.java:0) with 1 output partitions
25/05/26 03:33:54 INFO DAGScheduler: Final stage: ResultStage 1 (start at NativeMethodAccessorImpl.java:0)
25/05/26 03:33:54 INFO DAGScheduler: Parents of final stage: List()
25/05/26 03:33:54 INFO DAGScheduler: Submitting ResultStage 1 (MapPartitionsRDD[7] at start at NativeMethodAccessorImpl.java:0), which has no missing parents
25/05/26 03:33:54 INFO MemoryStore: Block broadcast_1 stored as values in memory (estimated size 239.7 KiB, free 911.7 MiB)
25/05/26 03:33:54 INFO MemoryStore: Block broadcast_1_piece0 stored as bytes in memory (estimated size 87.4 KiB, free 911.7 MiB)

```

```

[hadoop@hieu-hao ~]$ hadoop@hieu-hao:~/project
25/05/26 03:33:55 INFO MicroBatchExecution: Streaming query made progress: {
  "id" : "f772130c-28c5-4776-a508-30e3793bad35",
  "runId" : "29893280-41e7-48f5-afa7-c39d74b3900d",
  "name" : null,
  "timestamp" : "2025-05-26T03:33:54.533Z",
  "batchId" : 24,
  "numInputRows" : 1,
  "inputRowsPerSecond" : 66.66666666666667,
  "processedRowsPerSecond" : 0.8278145695364238,
  "durationMs" : {
    "addBatch" : 1055,
    "commitOffsets" : 37,
    "getBatch" : 0,
    "latestOffset" : 7,
    "queryPlanning" : 47,
    "triggerExecution" : 1208,
    "walCommit" : 55
  },
  "stateOperators" : [ ],
  "sources" : [ {
    "description" : "KafkaV2[Subscribe[air-quality]]",
    "startOffset" : {
      "air-quality" : {
        "0" : 56
      }
    },
    "endOffset" : {
      "air-quality" : {
        "0" : 57
      }
    },
    "latestOffset" : {
      "air-quality" : {
        "0" : 57
      }
    },
    "numInputRows" : 1,
    "inputRowsPerSecond" : 66.66666666666667,
    "processedRowsPerSecond" : 0.8278145695364238,
    "metrics" : {
      "avgOffsetsBehindLatest" : "0.0",
      "maxOffsetsBehindLatest" : "0",
      "minOffsetsBehindLatest" : "0"
    }
  }],
  "sink" : {
    "description" : "FileSink[lakehouse/bronze/streaming/air_quality/]",
    "numOutputRows" : -1
  }
}

```

## 2. Xử lý luồng dữ liệu streaming từ API (Producer)

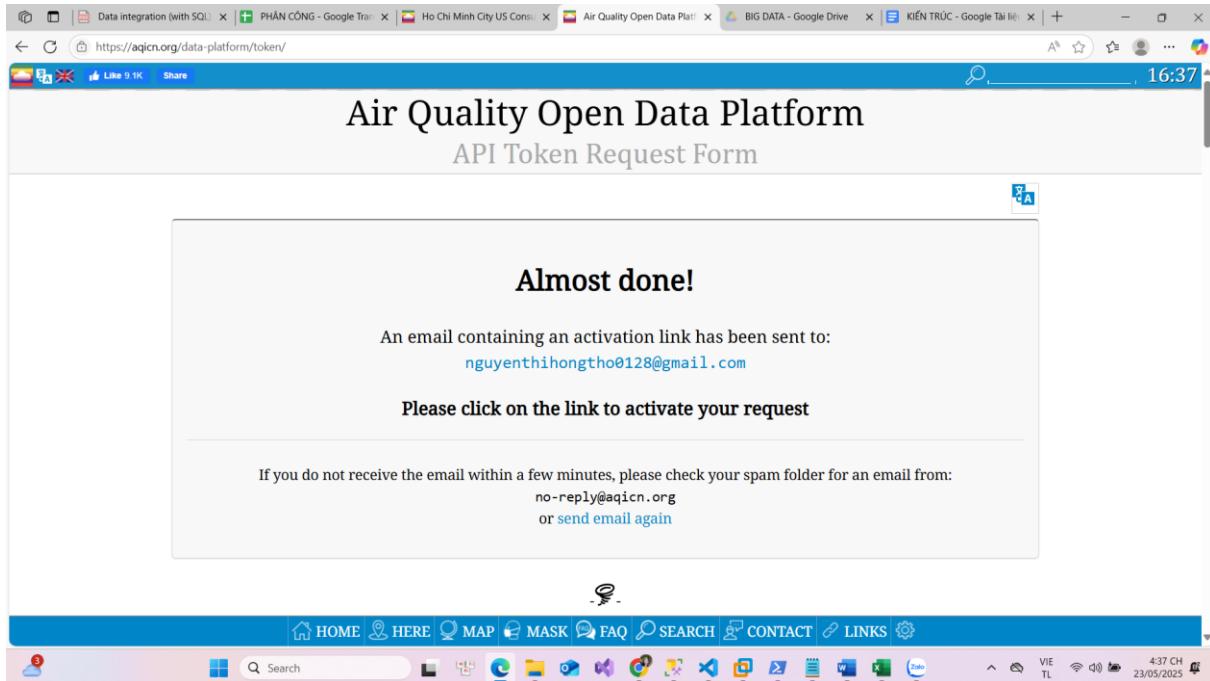
Lấy API dữ liệu AQI từ web:

<https://aqicn.org/city/vietnam/ho-chi-minh-city/us-consulate/>

Trang này hiển thị Chỉ số Chất lượng Không khí (Air Quality Index - AQI) theo thời gian thực, cụ thể là từ Khu Liên cơ quan Bộ Tài Nguyên và Môi Trường - số 20 Đ. Lý Chính Thắng (KK), Ho Chi Minh City, Vietnam.



Lấy token gửi về mail:



Lấy link URL API:

```
{
  "status": "ok",
  "data": {
    "aqi": 27,
    "idx": -476182,
    "attribution": [
      {
        "url": "http://cem.gov.vn/",
        "name": "Vietnam Center For Environmental Monitoring Portal (công thông tin quan trắc môi trường)",
        "station": "31390912357075263208060500522"
      }
    ]
  }
}
```

<https://api.waqi.info/feed/A476182/?token=3be4e2f3785c7e0aa3a2e903abaa49b1a575cd a3>

Dữ liệu JSON mẫu trả về sẽ bao gồm:

- aqi: Chỉ số chất lượng không khí tổng thể.

- iaqi: Giá trị theo từng thành phần (PM2.5, PM10, CO, NO2, SO2, nhiệt độ t, độ ẩm h, tốc độ gió w, v.v.)
- time: Thời gian ghi nhận dữ liệu.
- city: Thông tin địa điểm.

```
{
  "status": "ok",
  "data": {
    "aqi": 27,
    "idx": -476182,
    "attributions": [
      {
        "url": "http://cem.gov.vn/",
        "name": "Vietnam Center For Environmental Monitoring Portal (cổng thông tin quan trắc môi trường)",
        "station": "31390912357075263208060500522"
      },
      {
        "url": "https://waqi.info/",
        "name": "World Air Quality Index Project"
      }
    ],
    "city": {
      "geo": [
        10.7823,
        106.6834
      ],
      "name": "HCM: Khu Liên cơ quan Bộ Tài Nguyên và Môi Trường - số 20 Đ. Lý Chính Thắng (KK)",
      "url": "https://aqicn.org/station/@476182",
      "location": "Khu liên cơ quan Bộ Tài nguyên và Môi trường tại TP.HCM, 200, Đường Lý Chính Thắng, Ward 9, District 3, Ho Chi Minh City, 70001, Vietnam"
    },
    "dominentpol": "pm25",
    "iaqi": {
      "co": {
        "v": 0
      },
      "no2": {
        "v": 4
      },
      "o3": {
        "v": 22
      },
      "pm10": {
        "v": 18
      },
      "pm25": {
        "v": 27
      },
      "so2": {
        "v": 49
      }
    }
  }
}
```

```
        }
    },
    "time": {
        "s": "2025-05-25 17:00:00",
        "tz": "+07:00",
        "v": 1748167200,
        "iso": "2025-05-25T10:00:00Z"
    }
}
```

Tạo một Topic: air-quality

```
$ vim create_topic.sh
```

```
hadoop@hongtho-m: ~ % hadoop@hongtho-m: ~ % hadoop@hongtho-m: ~ %
```

```
#!/bin/bash

unset CLASSPATH
unset HADOOP_HOME
unset HIVE_HOME
unset HADOOP_CONF_DIR
unset HIVE_CONF_DIR

# Script để tạo một Kafka topic

TOPIC_NAME="air-quality"
BOOTSTRAP_SERVER="localhost:9092"
PARTITIONS=1
REPLICATION_FACTOR=1

# Tạo topic
bin/kafka-topics.sh \
--create \
--topic "$TOPIC_NAME" \
--bootstrap-server "$BOOTSTRAP_SERVER" \
--partitions "$PARTITIONS" \
--replication-factor "$REPLICATION_FACTOR"
```

```
$ ./create_topic.sh
```

```

hadoopphongtho@hongtho-master:~/kafka
[2025-05-23 10:50:00,781] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from _consumer_offsets-27 in 140 milliseconds for epoch 0, of which 140 milliseconds was spent in the scheduler. (kafka.coordinator.group.GroupMetadataManager)
[2025-05-23 10:50:00,784] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from _consumer_offsets-30 in 141 milliseconds for epoch 0, of which 141 milliseconds was spent in the scheduler. (kafka.coordinator.group.GroupMetadataManager)
[2025-05-23 10:50:00,785] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from _consumer_offsets-33 in 141 milliseconds for epoch 0, of which 141 milliseconds was spent in the scheduler. (kafka.coordinator.group.GroupMetadataManager)
[2025-05-23 10:50:00,785] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from _consumer_offsets-36 in 141 milliseconds for epoch 0, of which 141 milliseconds was spent in the scheduler. (kafka.coordinator.group.GroupMetadataManager)
[2025-05-23 10:50:00,786] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from _consumer_offsets-39 in 141 milliseconds for epoch 0, of which 141 milliseconds was spent in the scheduler. (kafka.coordinator.group.GroupMetadataManager)
[2025-05-23 10:50:00,787] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from _consumer_offsets-41 in 141 milliseconds for epoch 0, of which 141 milliseconds was spent in the scheduler. (kafka.coordinator.group.GroupMetadataManager)
[2025-05-23 10:50:00,787] INFO [GroupMetadataManager brokerId=0] Creating topic air-quality with configuration {} and initial partition assignment Map(0 -> ArrayBuffer(0)) (kafka.zk.AdminZkClient)
[2025-05-23 10:50:00,777] INFO [ReplicaFetcherManager on broker 0] Removed fetcher for partitions Set(air-quality-0) (kafka.server.ReplicaFetcherManager)
[2025-05-23 10:50:07,786] INFO [Log partition=air-quality-0, dir=/tmp/kafka-logs] Loading producer state till offset 0 with message format version 2 (kafka.log.Log)
[2025-05-23 10:50:07,790] INFO [Log partition=air-quality-0] Created log for partition air-quality-0 in /tmp/kafka-logs/air-quality-0 with properties {} (kafka.log.LogManager)
[2025-05-23 10:50:07,792] INFO [Partition air-quality-0 broker=0] No checkpointed highwatermark is found for partition air-quality-0 (kafka.cluster.Partition)
[2025-05-23 10:50:07,793] INFO [Partition air-quality-0 broker=0] Log loaded for partition air-quality-0 with initial high watermark 0 (kafka.cluster.Partition)
[2025-05-23 10:50:07,793] INFO [Partition air-quality-0 broker=0] Creating log for partition air-quality-0 with configuration {} and initial partition assignment Map(0 -> ArrayBuffer(0)) (kafka.zk.AdminZkClient)
[2025-05-23 10:50:29,293] INFO [ReplicaFetcherManager on broker 0] Removed fetcher for partitions Set(AirQuality-0) (kafka.server.ReplicaFetcherManager)
[2025-05-23 10:50:29,297] INFO [Log partition=AirQuality-0, dir=/tmp/kafka-logs] Loading producer state till offset 0 with message format version 2 (kafka.log.Log)
[2025-05-23 10:50:29,298] INFO [Log partition=AirQuality-0] Created log for partition AirQuality-0 in /tmp/kafka-logs/AirQuality-0 with properties {} (kafka.log.LogManager)
[2025-05-23 10:50:29,299] INFO [Partition AirQuality-0 broker=0] No checkpointed highwatermark is found for partition AirQuality-0 (kafka.cluster.Partition)
[2025-05-23 10:50:29,299] INFO [Partition AirQuality-0 broker=0] Log loaded for partition AirQuality-0 with initial high watermark 0 (kafka.cluster.Partition)

hadoopphongtho@hongtho-m:~/kafka$ ./r_mess.sh
hello
my kafka
hello streaming hadoopphongtho
^CProcessed a total of 3 messages
hadoopphongtho@hongtho-m:~/kafka$ vim r_mess.sh
hadoopphongtho@hongtho-m:~/kafka$ vim create_topic.sh
hadoopphongtho@hongtho-m:~/kafka$ ./create_topic.sh
WARNING: Due to limitations in metric names, topics with a period ('.') or underscore '_' could collide. To avoid issues it is best to use either, but not both.
[2025-05-23 10:49:02,141] WARN [AdminClient clientId=adminclient-1] Connect to node -1 (localhost/127.0.0.1:9092) could not be established. Broker may not be available. (org.apache.kafka.clients.NetworkClient)
[2025-05-23 10:49:02,244] WARN [AdminClient clientId=adminclient-1] Connect to node -1 (localhost/127.0.0.1:9092) could not be established. Broker may not be available. (org.apache.kafka.clients.NetworkClient)
[2025-05-23 10:49:02,445] WARN [AdminClient clientId=adminclient-1] Connect to node -1 (localhost/127.0.0.1:9092) could not be established. Broker may not be available. (org.apache.kafka.clients.NetworkClient)
[2025-05-23 10:49:02,646] WARN [AdminClient clientId=adminclient-1] Connect to node -1 (localhost/127.0.0.1:9092) could not be established. Broker may not be available. (org.apache.kafka.clients.NetworkClient)
[2025-05-23 10:49:02,847] WARN [AdminClient clientId=adminclient-1] Connect to node -1 (localhost/127.0.0.1:9092) could not be established. Broker may not be available. (org.apache.kafka.clients.NetworkClient)
[2025-05-23 10:49:03,149] WARN [AdminClient clientId=adminclient-1] Connect to node -1 (localhost/127.0.0.1:9092) could not be established. Broker may not be available. (org.apache.kafka.clients.NetworkClient)
[2025-05-23 10:49:04,153] WARN [AdminClient clientId=adminclient-1] Connect to node -1 (localhost/127.0.0.1:9092) could not be established. Broker may not be available. (org.apache.kafka.clients.NetworkClient)
[2025-05-23 10:49:05,057] WARN [AdminClient clientId=adminclient-1] Connect to node -1 (localhost/127.0.0.1:9092) could not be established. Broker may not be available. (org.apache.kafka.clients.NetworkClient)
^Chadoopphongtho@hongtho-m:~/kafka$ ./create_topic.sh
WARNING: Due to limitations in metric names, topics with a period ('.') or underscore '_' could collide. To avoid issues it is best to use either, but not both.
Created topic air-quality.
hadoopphongtho@hongtho-m:~/kafka$ vim create_topic.sh
hadoopphongtho@hongtho-m:~/kafka$ ./create_topic.sh
Created topic AirQuality.
hadoopphongtho@hongtho-m:~/kafka$ 

```

Log ghi nhận bên màn hình trái đã tạo Topic air-quality.

Cài Kafka Producer (Python):

Kafka Producer là thành phần đẩy dữ liệu vào Kafka topic. Trong luồng xử lý dữ liệu:

Nguồn dữ liệu (API) → Kafka Producer → Kafka Topic → Spark Streaming → Lakehouse (Bronze)

Cài pip để cài thư viện Kafka Producer:

```
$ sudo apt install python3-pip -y
```

```
Setting up binutils-x86-64-linux-gnu (2.42-4ubuntu2.5) ...
Setting up cpp-x86-64-linux-gnu (4:13.2.0-7ubuntu1) ...
Setting up libpython3-dev:amd64 (3.12.3-0ubuntu2) ...
Setting up cpp-13 (13.3.0-6ubuntu2~24.04) ...
Setting up gcc-13-x86-64-linux-gnu (13.3.0-6ubuntu2~24.04) ...
Setting up binutils (2.42-4ubuntu2.5) ...
Setting up dpkg-dev (1.22.6ubuntu6.1) ...
Setting up python3-dev (3.12.3-0ubuntu2) ...
Setting up gcc-13 (13.3.0-6ubuntu2~24.04) ...
Setting up cpp (4:13.2.0-7ubuntu1) ...
Setting up g++-13-x86-64-linux-gnu (13.3.0-6ubuntu2~24.04) ...
Setting up gcc-x86-64-linux-gnu (4:13.2.0-7ubuntu1) ...
Setting up gcc (4:13.2.0-7ubuntu1) ...
Setting up g++-x86-64-linux-gnu (4:13.2.0-7ubuntu1) ...
Setting up g++-13 (13.3.0-6ubuntu2~24.04) ...
Setting up g++ (4:13.2.0-7ubuntu1) ...
update-alternatives: using /usr/bin/g++ to provide /usr/bin/c++ (c++) in auto mode
Setting up build-essential (12.10ubuntu1) ...
Processing triggers for man-db (2.12.0-4build2) ...
Processing triggers for libc-bin (2.39-0ubuntu8.4) ...
Scanning processes...
Scanning candidates...
Scanning linux images...

Running kernel seems to be up-to-date.

Restarting services...

Service restarts being deferred:
/etc/needrestart/restart.d/dbus.service
systemctl restart systemd-logind.service
systemctl restart unattended-upgrades.service

No containers need to be restarted.

User sessions running outdated binaries:
hadoophongtho @ user manager service: systemd[1456]

No VM guests are running outdated hypervisor (qemu) binaries on this host.
root@hongtho-master:~#
```

```
root@hongtho-master:~# sudo apt install python3.12-venv -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  python3-pip-whl python3-setuptools-whl
The following NEW packages will be installed:
  python3-pip-whl python3-setuptools-whl python3.12-venv
0 upgraded, 3 newly installed, 0 to remove and 149 not upgraded.
Need to get 2,424 kB of archives.
After this operation, 2,771 kB of additional disk space will be used.
Get:1 http://vn.archive.ubuntu.com/ubuntu noble-updates/universe amd64 python3-pip-whl all 24.0+dfsg-1ubuntu1.1 [1,703 kB]
Get:2 http://vn.archive.ubuntu.com/ubuntu noble-updates/universe amd64 python3-setuptools-whl all 68.1.2-2ubuntu1.1 [716 kB]
Get:3 http://vn.archive.ubuntu.com/ubuntu noble-updates/universe amd64 python3.12-venv amd64 3.12.3-1ubuntu0.5 [5,678 B]
Fetched 2,424 kB in 1s (4,681 kB/s)
Selecting previously unselected package python3-pip-whl.
(Reading database ... 144046 files and directories currently installed.)
Preparing to unpack .../python3-pip-whl_24.0+dfsg-1ubuntu1.1_all.deb ...
Unpacking python3-pip-whl (24.0+dfsg-1ubuntu1.1) ...
Selecting previously unselected package python3-setuptools-whl.
Preparing to unpack .../python3-setuptools-whl_68.1.2-2ubuntu1.1_all.deb ...
Unpacking python3-setuptools-whl (68.1.2-2ubuntu1.1) ...
Selecting previously unselected package python3.12-venv.
Preparing to unpack .../python3.12-venv_3.12.3-1ubuntu0.5_amd64.deb ...
```

Ubuntu phiên bản mới không còn hỗ trợ cài đặt trực tiếp qua pip. Nên tạo môi trường ảo env để cô lập (isolate) các thư viện Python và phiên bản phụ thuộc cho dự án hiện tại, tránh xung đột với hệ thống hoặc các dự án khác:

```
$ python3 -m venv ~/myenv
```

Kích hoạt myenv:

```
$ source ~/myenv/bin/activate
```

Cài đặt thư viện client Kafka dành cho Python. Cho phép viết Producer (gửi dữ liệu vào Kafka) và Consumer (đọc dữ liệu từ Kafka):

```
$ pip install kafka-python requests
```

```
hadoophongtho@hongtho-master:/root$ cd
hadoophongtho@hongtho-master:~/myenv
hadoophongtho@hongtho-master:~/myenv/bin/activate
(myenv) hadoophongtho@hongtho-master:~$ pip install kafka-python requests
Collecting kafka-python
  Downloading kafka_python-2.2.10-py2.py3-none-any.whl.metadata (10.0 kB)
Collecting requests
  Downloading requests-2.32.3-py3-none-any.whl.metadata (4.6 kB)
Collecting charset-normalizer<4,>=2 (from requests)
  Downloading charset_normalizer-3.4.2-cp312-cp312-manylinux_2_17_x86_64.man
ylinux2014_x86_64.whl.metadata (35 kB)
Collecting idna<4,>=2.5 (from requests)
  Downloading idna-3.10-py3-none-any.whl.metadata (10 kB)
Collecting urllib3<3,>=1.21.1 (from requests)
  Downloading urllib3-2.4.0-py3-none-any.whl.metadata (6.5 kB)
Collecting certifi>=2017.4.17 (from requests)
  Downloading certifi-2025.4.26-py3-none-any.whl.metadata (2.5 kB)
Downloading kafka_python-2.2.10-py2.py3-none-any.whl (309 kB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 309.3/309.3 kB 870.8 kB/s eta 0:00:00
Downloading requests-2.32.3-py3-none-any.whl (64 kB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━ 64.9/64.9 kB 1.3 MB/s eta 0:00:00
Downloading certifi-2025.4.26-py3-none-any.whl (159 kB)
    ━━━━━━━━━━━━━━━━━━━━━━━━ 159.6/159.6 kB 1.1 MB/s eta 0:00:00
Downloading charset_normalizer-3.4.2-cp312-cp312-manylinux_2_17_x86_64.manyl
inux2014_x86_64.whl (148 kB)
    ━━━━━━━━━━━━━━━━━━━━━━ 148.6/148.6 kB 1.4 MB/s eta 0:00:00
Downloading idna-3.10-py3-none-any.whl (70 kB)
    ━━━━━━━━━━━━━━━━━━ 70.4/70.4 kB 1.4 MB/s eta 0:00:00
Downloading urllib3-2.4.0-py3-none-any.whl (128 kB)
    ━━━━━━━━━━━━━━━━ 128.7/128.7 kB 1.4 MB/s eta 0:00:00
Installing collected packages: kafka-python, urllib3, idna, charset-normaliz
er, certifi, requests
Successfully installed certifi-2025.4.26 charset-normalizer-3.4.2 idna-3.10
kafka-python-2.2.10 requests-2.32.3 urllib3-2.4.0
(myenv) hadoophongtho@hongtho-master:~$ |
```

Tạo project để tạo file air\_quality\_kafka\_producer.py:

```
$ mkdir project
$ cd project/
$ vim air_quality_kafka_producer.py
```

API\_URL: nơi lấy dữ liệu từ trạm đo không khí ở HCM.

KAFKA BROKER: địa chỉ Kafka server (ở local).

TOPIC: tên Kafka topic sẽ gửi dữ liệu đến.

Sau đó, dùng `json.dumps` để chuyển dữ liệu Python thành JSON, rồi encode thành bytes (utf-8).

## Chạy file:

```
$ python3 air_quality_kafka_producer.py
```

Thành công:

```
(myenv) hadoophongtho@hongtho-master:~/project$ python3 air_quality_kafka_producer.py
Sent data to Kafka: 2025-05-23T10:51:11.071791
```

Tạo file lấy data từ Kafka → HDFS:

```
$ cd spark
$ vim spark_kafka_to_hdfs.py
```

```
hadoopphongtho@hongtho-master:~$ cd spark
hadoopphongtho@hongtho-master:~/spark$ ls
bin  data  jars  LICENSE  logs  python  README.md  sbin  yarn
conf  examples  kubernetes  licenses  NOTICE  R  RELEASE  work
hadoopphongtho@hongtho-master:~/spark$ vim spark_kafka_to_hdfs.py
```

```
hadoopphongtho@hongtho-m ~  hadoopphongtho@hongtho-m ~  Windows PowerShell ~ + ~
from pyspark.sql import SparkSession
from pyspark.sql.functions import from_json, col
from pyspark.sql.types import *

spark = SparkSession.builder \
    .appName("KafkaToHDFS") \
    .getOrCreate()

df_raw = spark.readStream \
    .format("kafka") \
    .option("kafka.bootstrap.servers", "localhost:9092") \
    .option("subscribe", "air-quality") \
    .load()

df_json = df_raw.selectExpr("CAST(value AS STRING) as json_str")

schema = StructType().add("data", MapType(StringType(), StringType())).add("fetched_at", StringType())

df_parsed = df_json.select(from_json(col("json_str"), schema).alias("parsed"))
df_result = df_parsed.select("parsed.*")

df_result.writeStream \
    .format("json") \
    .option("path", "/Lakehouse/bronze/streaming/air_quality/") \
    .option("checkpointLocation", "/Lakehouse/checkpoints/air_quality/") \
    .outputMode("append") \
    .start() \
    .awaitTermination()
```

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import from_json, col
from pyspark.sql.types import *

spark = SparkSession.builder \
    .appName("KafkaToHDFS") \
```

```

    .getOrCreate()

df_raw = spark.readStream \
    .format("kafka") \
    .option("kafka.bootstrap.servers", "localhost:9092") \
    .option("subscribe", "air-quality") \
    .load()

df_json = df_raw.selectExpr("CAST(value AS STRING) as json_str")

schema = StructType().add("data", MapType(StringType(),
StringType())).add("fetched_at", StringType())

df_parsed = df_json.select(from_json(col("json_str"), schema).alias("parsed"))
df_result = df_parsed.select("parsed.*")

df_result.writeStream \
    .format("json") \
    .option("path", "/lakehouse/bronze/streaming/air_quality/") \
    .option("checkpointLocation", "/lakehouse/checkpoints/air_quality/") \
    .outputMode("append") \
    .start() \
    .awaitTermination()

```

File này đọc dữ liệu thời gian thực từ Kafka topic air-quality, parse JSON, và ghi vào HDFS theo định dạng JSON ở layer Bronze của Lakehouse.

```

df_raw = spark.readStream \
    .format("kafka") \
    .option("kafka.bootstrap.servers", "localhost:9092") \
    .option("subscribe", "air-quality") \
    .load()

```

Phần này đọc dữ liệu từ Kafka Stream: air-quality.

Kafka gửi dữ liệu dạng bytes, nên phải ép kiểu value thành chuỗi JSON:

```
df_json = df_raw.selectExpr("CAST(value AS STRING) as json_str")
```

Ghi dữ liệu stream ra JSON files (layer Bronze):

```

df_result.writeStream \
    .format("json") \
    .option("path", "/lakehouse/bronze/streaming/air_quality/") \
    .option("checkpointLocation", "/lakehouse/checkpoints/air_quality/") \
    .outputMode("append") \
    .start() \
    .awaitTermination()

```

Tải Jar để đảm bảo thư viện slf4j-api sẵn có cho quá trình build hoặc chạy ứng dụng Spark:

```
wget https://repo1.maven.org/maven2/org/slf4j/slf4j-api/2.0.7/slf4j-api-2.0.7.jar -P  
~/m2/repository/org/slf4j/slf4j-api/2.0.7/
```

```
hadoophongtho@hongtho-master:~/spark$ wget https://repo1.maven.org/maven2/org/slf4j/slf4j-api/2.0.7/slf4j-api-2.0.7.jar -P ~/m2/repository/org/slf4j/slf4j-api/2.0.7/  
--2025-05-23 11:18:27-- https://repo1.maven.org/maven2/org/slf4j/slf4j-api/2.0.7/slf4j-api-2.0.7.jar  
Resolving repo1.maven.org (repo1.maven.org)... 199.232.196.209, 199.232.192.209, 2a04:4e42:4c::209, ...  
Connecting to repo1.maven.org (repo1.maven.org)|199.232.196.209|:443... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 63635 (62K) [application/java-archive]  
Saving to: '/home/hadoophongtho/.m2/repository/org/slf4j/slf4j-api/2.0.7/slf4j-api-2.0.7.jar'  
  
slf4j-api-2.0.7.jar 100%[=====] 62.14K 296KB/s in 0.2s  
2025-05-23 11:18:28 (296 KB/s) - '/home/hadoophongtho/.m2/repository/org/slf4j/slf4j-api/2.0.7/slf4j-api-2.0.7.jar' saved [63635/63635]
```

Chạy lệnh spark-submit để submit một job Spark có hỗ trợ Kafka:

Lệnh này tự động tải thư viện hỗ trợ Kafka cho Spark phiên bản 3.5.3 và Scala 2.12, giúp Spark có thể đọc/ghi Kafka:

```
$ spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.12:3.5.3  
spark_kafka_to_hdfs.py
```

Thành công:

```
hadoophongtho@hongtho-master:~/spark$ spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.12:3.5.3 spark_kafka_to_hdfs.py
:: loading settings :: url = jar:file:/home/hadoophongtho/spark/jars/ivy-2.5
.1.jar!/org/apache/ivy/core/settings/ivysettings.xml
Ivy Default Cache set to: /home/hadoophongtho/.ivy2/cache
The jars for the packages stored in: /home/hadoophongtho/.ivy2/jars
org.apache.spark#spark-sql-kafka-0-10_2.12 added as a dependency
:: resolving dependencies :: org.apache.spark#spark-submit-parent-36544cff-8
66d-4d53-81da-cc6537ff494d;1.0
  confs: [default]
    found org.apache.spark#spark-sql-kafka-0-10_2.12;3.5.3 in central
    found org.apache.spark#spark-token-provider-kafka-0-10_2.12;3.5.3 in
central
      found org.apache.kafka#kafka-clients;3.4.1 in central
      found org.lz4#lz4-java;1.8.0 in local-m2-cache
      found org.xerial.snappy#snappy-java;1.1.10.5 in local-m2-cache
      found org.slf4j#slf4j-api;2.0.7 in local-m2-cache
      found org.apache.hadoop#hadoop-client-runtime;3.3.4 in local-m2-cach
e
      found org.apache.hadoop#hadoop-client-api;3.3.4 in local-m2-cache
      found commons-logging#commons-logging;1.1.3 in local-m2-cache
      found com.google.code.findbugs#jsr305;3.0.0 in local-m2-cache
      found org.apache.commons#commons-pool2;2.11.1 in central
downloading file:/home/hadoophongtho/.m2/repository/org/slf4j/slf4j-api/2.0.
7/slf4j-api-2.0.7.jar ...
  [SUCCESSFUL ] org.slf4j#slf4j-api;2.0.7!slf4j-api.jar (4ms)
:: resolution report :: resolve 368ms :: artifacts dl 15ms
  :: modules in use:
    com.google.code.findbugs#jsr305;3.0.0 from local-m2-cache in [defaul
t]
    commons-logging#commons-logging;1.1.3 from local-m2-cache in [defaul
t]
    org.apache.commons#commons-pool2;2.11.1 from central in [default]
    org.apache.hadoop#hadoop-client-api;3.3.4 from local-m2-cache in [de
fault]
    org.apache.hadoop#hadoop-client-runtime;3.3.4 from local-m2-cache in
[default]
    org.apache.kafka#kafka-clients;3.4.1 from central in [default]
    org.apache.spark#spark-sql-kafka-0-10_2.12;3.5.3 from central in [de
fault]
```

Để 2 terminal chạy song song:

```
(myenv) hadoopphongtho@hongtho-master:~/project$ python3 air_quality_kafka_producer.py
Sent data to Kafka: 2025-05-23T10:51:11.071791
Sent data to Kafka: 2025-05-23T10:52:11.571093
Sent data to Kafka: 2025-05-23T10:53:11.948922
[{"Sent data to Kafka: 2025-05-23T10:54:12.327762
Sent data to Kafka: 2025-05-23T10:55:12.719953
Sent data to Kafka: 2025-05-23T10:56:13.096686
Sent data to Kafka: 2025-05-23T10:57:13.478954
Sent data to Kafka: 2025-05-23T10:58:13.856265
Sent data to Kafka: 2025-05-23T10:59:14.234761
Sent data to Kafka: 2025-05-23T11:00:14.609525
Sent data to Kafka: 2025-05-23T11:01:14.988129
Sent data to Kafka: 2025-05-23T11:02:15.361023
Sent data to Kafka: 2025-05-23T11:03:15.752728
Sent data to Kafka: 2025-05-23T11:04:16.131674
Sent data to Kafka: 2025-05-23T11:05:16.518594
Sent data to Kafka: 2025-05-23T11:06:16.899594
Sent data to Kafka: 2025-05-23T11:07:17.281868
Sent data to Kafka: 2025-05-23T11:08:17.674263
Sent data to Kafka: 2025-05-23T11:09:18.066420
Sent data to Kafka: 2025-05-23T11:10:18.448830
Sent data to Kafka: 2025-05-23T11:11:18.838538
Sent data to Kafka: 2025-05-23T11:12:19.783797
Sent data to Kafka: 2025-05-23T11:13:20.172629
Sent data to Kafka: 2025-05-23T11:14:20.579389
Sent data to Kafka: 2025-05-23T11:15:20.958914
Sent data to Kafka: 2025-05-23T11:16:21.352146
Sent data to Kafka: 2025-05-23T11:17:21.736985
Sent data to Kafka: 2025-05-23T11:18:22.119721
Sent data to Kafka: 2025-05-23T11:19:22.503013
Sent data to Kafka: 2025-05-23T11:20:22.888121
Sent data to Kafka: 2025-05-23T11:21:23.276819
Sent data to Kafka: 2025-05-23T11:22:23.665879
Sent data to Kafka: 2025-05-23T11:23:24.044919
]
{
    "air-quality" : {
        "0" : 33
    }
},
"latestOffset" : {
    "air-quality" : {
        "0" : 33
    }
},
"numInputRows" : 1,
"inputRowsPerSecond" : 83.33333333333333,
"processedRowsPerSecond" : 1.2953367875647668,
"metrics" : [
    "avgOffsetsBehindLatest" : "0.0",
    "maxOffsetsBehindLatest" : "0",
    "minOffsetsBehindLatest" : "0"
],
},
"sink" : {
    "description" : "FileSink[/lakehouse/bronze/streaming/air_quality/]",
    "numOutputRows" : -1
}
}
]
25/05/23 11:23:34 INFO MicroBatchExecution: Streaming query has been idle and waiting for new data more than 10000 ms.
25/05/23 11:23:41 INFO NetworkClient: [AdminClient clientId=adminclient-1] Node -1 disconnected.
25/05/23 11:23:44 INFO MicroBatchExecution: Streaming query has been idle and waiting for new data more than 10000 ms.
25/05/23 11:23:49 INFO MicroBatchExecution: Streaming query has been idle and waiting for new data more than 10000 ms.
25/05/23 11:24:04 INFO MicroBatchExecution: Streaming query has been idle and waiting for new data more than 10000 ms.
25/05/23 11:24:06 INFO BlockManagerInfo: Removed broadcast_5_piece0 on hongtho-master:34397 in memory (size: 86.1 KiB, free: 366.3 MiB)
```

Xem Job trên localhost:

<http://192.168.232.12.40:4040>

| Job Id (Job Group)                       | Description                                                                                                                       | Submitted           | Duration | Stages: Succeeded/Total | Tasks (for all stages): Succeeded/Total |
|------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------|---------------------|----------|-------------------------|-----------------------------------------|
| 2 (29ee6416-2d85-4652-b022-3e584fbe8176) | id = e9cec340-7f59-43c7-9950-7927ff6984b7 runId = 29ee6416-2d85-4652-b022-3e584fbe8...<br>start at NativeMethodAccessormpl.java:0 | 2025/05/23 11:20:22 | 0.6 s    | 1/1                     | 1/1                                     |
| 1 (29ee6416-2d85-4652-b022-3e584fbe8176) | id = e9cec340-7f59-43c7-9950-7927ff6984b7 runId = 29ee6416-2d85-4652-b022-3e584fbe8...<br>start at NativeMethodAccessormpl.java:0 | 2025/05/23 11:19:22 | 1 s      | 1/1                     | 1/1                                     |
| 0 (29ee6416-2d85-4652-b022-3e584fbe8176) | id = e9cec340-7f59-43c7-9950-7927ff6984b7 runId = 29ee6416-2d85-4652-b022-3e584fbe8...<br>start at NativeMethodAccessormpl.java:0 | 2025/05/23 11:18:43 | 0.4 s    | 1/1                     | 1/1                                     |



The screenshot shows the Apache Spark 3.5.3 Structured Streaming application UI. The main title bar says "KafkaToHDFS application UI". The top navigation bar includes links for Jobs, Stages, Storage, Environment, Executors, SQL / DataFrame, and Structured Streaming. The "Structured Streaming" tab is selected. Below the navigation is a section titled "Streaming Query" with a subtitle "Active Streaming Queries (1)". A table lists one query: Name <no name>, Status RUNNING, ID e9cec340-7f59-43c7-9950-7927ff6984b7, Run ID 29ee6416-2d85-4652-b022-3e584fbe8176, Start Time 2025/05/23 11:18:40, Duration 2 minutes 12 seconds, Avg Input/sec 60.61, Avg Process/sec 0.67, and Latest Batch 2. There are pagination controls at the bottom.

### Streaming Query

Active Streaming Queries (1)

| Name      | Status  | ID                                   | Run ID                               | Start Time          | Duration             | Avg Input/sec | Avg Process/sec | Latest Batch |
|-----------|---------|--------------------------------------|--------------------------------------|---------------------|----------------------|---------------|-----------------|--------------|
| <no name> | RUNNING | e9cec340-7f59-43c7-9950-7927ff6984b7 | 29ee6416-2d85-4652-b022-3e584fbe8176 | 2025/05/23 11:18:40 | 2 minutes 12 seconds | 60.61         | 0.67            | 2            |

Page: 1 | 1 Pages. Jump to 1 | Show 100 items in a page. Go

The screenshot shows the Apache Spark 3.5.3 Structured Streaming application UI. The main title bar says "KafkaToHDFS application UI". The top navigation bar includes links for Jobs, Stages, Storage, Environment, Executors, SQL / DataFrame, and Structured Streaming. The "SQL / DataFrame" tab is selected. Below the navigation is a section titled "SQL / DataFrame" with a subtitle "Completed Queries: 3". A table lists three completed queries: ID 2 (runid = 29ee6416-2d85-4652-b022-3e584fbe8176 batch = 2), ID 1 (runid = 29ee6416-2d85-4652-b022-3e584fbe8176 batch = 1), and ID 0 (runid = 29ee6416-2d85-4652-b022-3e584fbe8176 batch = 0). Each row has a "details" link. There are pagination controls at the bottom.

### SQL / DataFrame

Completed Queries: 3

Completed Queries (3)

| ID | Description                                                                                      | Submitted           | Duration | Job IDs |
|----|--------------------------------------------------------------------------------------------------|---------------------|----------|---------|
| 2  | id = e9cec340-7f59-43c7-9950-7927ff6984b7 runid = 29ee6416-2d85-4652-b022-3e584fbe8176 batch = 2 | 2025/05/23 11:20:22 | 0.7 s    | [2]     |
| 1  | id = e9cec340-7f59-43c7-9950-7927ff6984b7 runid = 29ee6416-2d85-4652-b022-3e584fbe8176 batch = 1 | 2025/05/23 11:19:22 | 1 s      | [1]     |
| 0  | id = e9cec340-7f59-43c7-9950-7927ff6984b7 runid = 29ee6416-2d85-4652-b022-3e584fbe8176 batch = 0 | 2025/05/23 11:18:42 | 1 s      | [0]     |

Page: 1 | 1 Pages. Jump to 1 | Show 100 items in a page. Go

The screenshot shows the Apache Spark 3.5.3 Structured Streaming application UI. The main title bar says "KafkaToHDFS application UI". The top navigation bar includes links for Jobs, Stages, Storage, Environment, Executors, SQL / DataFrame, and Structured Streaming. The "SQL / DataFrame" tab is selected. Below the navigation is a section titled "SQL / DataFrame" with a subtitle "Completed Queries: 3". A table lists three completed queries: ID 2 (runid = 29ee6416-2d85-4652-b022-3e584fbe8176 batch = 2), ID 1 (runid = 29ee6416-2d85-4652-b022-3e584fbe8176 batch = 1), and ID 0 (runid = 29ee6416-2d85-4652-b022-3e584fbe8176 batch = 0). Each row has a "details" link. There are pagination controls at the bottom.

Nhận file thành công: Kiểm tra trong HDFS

```
$ hdfs dfs -ls /lakehouse/bronze/streaming/air_quality
```

```

hadoophongtho@hongtho-master:$ hdfs dfs -ls /lakehouse/bronze/streaming/air_quality
Found 45 items
drwxr-xr-x   - hadoophongtho supergroup          0 2025-05-25 13:22 /lakehouse/bronze/streaming/air_quality/_spark_metadata
-rw-r--r--   2 hadoophongtho supergroup          0 2025-05-25 10:55 /lakehouse/bronze/streaming/air_quality/part-00000-87c34a99-902b-403b-b896-910351ef7e65-
c000.json
-rw-r--r--   2 hadoophongtho supergroup         967 2025-05-25 11:03 /lakehouse/bronze/streaming/air_quality/part-00000-099a8634-58ec-47f7-b60e-d4d83c3be439-
c000.json
-rw-r--r--   2 hadoophongtho supergroup         991 2025-05-25 13:20 /lakehouse/bronze/streaming/air_quality/part-00000-0bf0c9ba-c9d8-4d0e-b27a-37ad2148e58a-
c000.json
-rw-r--r--   2 hadoophongtho supergroup         991 2025-05-25 11:15 /lakehouse/bronze/streaming/air_quality/part-00000-17ed58c8-c3d2-441c-ab3a-7b247b08c356-
c000.json
-rw-r--r--   2 hadoophongtho supergroup         991 2025-05-25 12:15 /lakehouse/bronze/streaming/air_quality/part-00000-1bc09206-4e70-4265-bef3-ff1e01e81a4b-
c000.json
-rw-r--r--   2 hadoophongtho supergroup        2423 2025-05-25 10:59 /lakehouse/bronze/streaming/air_quality/part-00000-29fb44db-021b-4805-9b25-207fcdda2c98-
c000.json
-rw-r--r--   2 hadoophongtho supergroup         967 2025-05-25 12:22 /lakehouse/bronze/streaming/air_quality/part-00000-2a335ff3-550f-4eef-a00e-dd77700ef078-
c000.json
-rw-r--r--   2 hadoophongtho supergroup         991 2025-05-25 11:07 /lakehouse/bronze/streaming/air_quality/part-00000-3509796b-0e0f-48e6-b0b5-ce2b6764ce24-
c000.json
-rw-r--r--   2 hadoophongtho supergroup         991 2025-05-25 12:19 /lakehouse/bronze/streaming/air_quality/part-00000-36d8079b-8045-4a21-b1b4-c4cf0f4b7bcl-
c000.json
-rw-r--r--   2 hadoophongtho supergroup         991 2025-05-25 10:58 /lakehouse/bronze/streaming/air_quality/part-00000-424b5b8e-7e6f-49cc-93ca-c9fbf7dd081b-
c000.json
-rw-r--r--   2 hadoophongtho supergroup         991 2025-05-25 11:04 /lakehouse/bronze/streaming/air_quality/part-00000-50926a25-6403-4d50-880d-df2af8640fb2-
c000.json
-rw-r--r--   2 hadoophongtho supergroup         967 2025-05-25 11:10 /lakehouse/bronze/streaming/air_quality/part-00000-57c08da6-3aff-42c2-85a9-b1491ac01fd3-
c000.json
-rw-r--r--   2 hadoophongtho supergroup         991 2025-05-25 12:20 /lakehouse/bronze/streaming/air_quality/part-00000-58beb8d1-3ff1-4236-8cf3-3ed50427e02b-
c000.json
-rw-r--r--   2 hadoophongtho supergroup         991 2025-05-25 12:12 /lakehouse/bronze/streaming/air_quality/part-00000-626ce3b2-c9c8-4866-b219-72d1856d6cf2-
c000.json
-rw-r--r--   2 hadoophongtho supergroup         991 2025-05-25 12:18 /lakehouse/bronze/streaming/air_quality/part-00000-6c0fcf86-f21b-4162-82b4-4cf15c30d1a-
c000.json
-rw-r--r--   2 hadoophongtho supergroup         991 2025-05-25 12:14 /lakehouse/bronze/streaming/air_quality/part-00000-6f1a9ce4-b37b-452a-8edc-3759b9777407-
c000.json
-rw-r--r--   2 hadoophongtho supergroup        2423 2025-05-25 11:00 /lakehouse/bronze/streaming/air_quality/part-00000-7224a18c-c8f5-4ded-ad7b-9f279e11acfcd-
c000.json
-rw-r--r--   2 hadoophongtho supergroup         991 2025-05-25 11:13 /lakehouse/bronze/streaming/air_quality/part-00000-7b71f0cc-bc70-4e4b-ab22-5ebcf1b32134-
c000.json

```

```
$ hdfs dfs -cat /lakehouse/bronze/streaming/air_quality/part-00000-e51d2787-d385-4f38-a54d-8ef87d9ca701-c000.json
```

```

hadoophongtho@hongtho-master:$ hdfs dfs -cat /lakehouse/bronze/streaming/air_quality/part-00000-e51d2787-d385-4f38-a54d-8ef87d9ca701-c000.json
{"data": {"aqi": "14", "idx": "-476182", "attributions": "[{\\"url\\": \"http://cem.gov.vn/\", \"name\\": \"Vietnam Center For Environmental Monitoring Portal (công thô ng tin quan trắc môi trường)\", \"station\\": \"3139991235707526320806050522\"}, {\\"url\\": \"https://waqi.info/\", \"name\\": \"World Air Quality Index Project\"]"}, "city": "{\"geo\": [10.7823, 106.6834], \"name\": \"HCM: Khu Liên cơ quan Bộ Tài Nguyên và Môi Trường - số 20 B, Lý Chính Thắng (KK)\", \"url\": \"https://aqicn.org/station/0476182\", \"location\": \"Khu liên cơ quan Bộ Tài nguyên và Môi trường tại TP.HCM, 200, Đường Lý Chính Thắng, Ward 9, District 3, Ho Chi Minh City, 70001, Vietnam\"}, \"dominantpol\": \"pm25\", \"iaqi\": {\"co\": {\"v\": 0}, \"no2\": {\"v\": 5}, \"o3\": {\"v\": 22}, \"pm10\": {\"v\": 18}, \"pm25\": {\"v\": 14}, \"so2\": {\"v\": 49}}", "time": "{\"s\": \"2025-05-25 20:00:00\", \"tz\": \"+07:00\", \"v\": 1748178000}, \"iso\": \"2025-05-25T13:00:00Z\"}", "fetched_at": \"2025-05-25T13:21:20.168776\""}
hadoophongtho@hongtho-master:$ |

```

### 3. Xử lý Data Streaming qua 3 lớp: Bronze - Silver – Gold

Chỉ lưu các thông tin chủ yếu từ Json như:

```

"iaqi": {
    "co": {
        "v": 0
    },
    "no2": {
        "v": 4
    },
    "o3": {
        "v": 22
    },
    "pm10": {
        "v": 18
    },
    "pm25": {
        "v": 27
    },
    "so2": {
        "v": 49
    }
},
}
```

Và dòng Fetch\_time:

```
"fetched_at": "2025-05-25T12:13:07.526838"}
```

Đưa các dữ liệu này vào dạng delta:

```
hadoopphongtho@hongtho-master:~/spark$ vim LoadStreamingToSilver.py |
```

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, from_json, to_timestamp, when, hour,
dayofmonth, month, lit, coalesce
from pyspark.sql.types import StructType, StructField, StringType, FloatType

spark = SparkSession.builder.appName("AirQualitySilver").getOrCreate()

# Đọc từ Bronze layer
df_bronze = spark.readStream \
    .format("json") \
    .schema("data STRUCT<iaqi: STRING, time: STRING>, fetched_at STRING") \
    .load("hdfs://lakehouse/bronze/streaming/air_quality/")

iaqi_schema = StructType([
    StructField("co", StructType([StructField("v", FloatType())])),
    StructField("no2", StructType([StructField("v", FloatType())])),
    StructField("o3", StructType([StructField("v", FloatType())])),
    StructField("pm10", StructType([StructField("v", FloatType())])),
    StructField("pm25", StructType([StructField("v", FloatType())])),
    StructField("so2", StructType([StructField("v", FloatType())]))
])

time_schema = StructType([
    StructField("s", StringType()),
    StructField("tz", StringType()),
    StructField("v", StringType()),
    StructField("iso", StringType())
])

df_silver = df_bronze \
    .withColumn("iaqi_json", from_json(col("data.iaqi"), iaqi_schema)) \
    .withColumn("time_json", from_json(col("data.time"), time_schema)) \
    .withColumn("date", to_timestamp(col("fetched_at"))) \
    .select(
        "date",
        col("iaqi_json.co.v").alias("CO"),
        col("iaqi_json.no2.v").alias("NO2"),
        col("iaqi_json.o3.v").alias("O3"),
        col("iaqi_json.pm10.v").alias("TSP"),
        col("iaqi_json.pm25.v").alias("PM25"),
        col("iaqi_json.so2.v").alias("SO2")
    ) \
    .dropDuplicates(["date", "CO", "NO2", "O3", "TSP", "PM25", "SO2"])

cols_to_cast = ["TSP", "PM25", "O3", "CO", "NO2", "SO2"]
for col_name in cols_to_cast:
    df_silver = df_silver.withColumn(col_name, coalesce(col(col_name).cast("double"),
lit(0)))

df_silver = df_silver.withColumn(
```

```

    "AQI_Level",
    when(col("PM25") <= 12, "Good")
    .when((col("PM25") >= 13) & (col("PM25") <= 35), "Average")
    .when((col("PM25") >= 36) & (col("PM25") <= 55), "Moderate")
    .otherwise("Unhealthy")
)
df_silver = df_silver.withColumn("hour", hour(col("date")))
    .withColumn("day", dayofmonth(col("date")))
    .withColumn("month", month(col("date")))

query = df_silver.writeStream \
    .format("delta") \
    .outputMode("append") \
    .option("checkpointLocation", "/lakehouse/checkpoints/air_quality_silver/") \
    .option("path", "/lakehouse/silver/streaming/air_quality/") \
    .start()

query.awaitTermination()

```

```

from pyspark.sql import SparkSession
from pyspark.sql.functions import col, from_json, to_timestamp, when, hour, dayofmonth, month, lit, coalesce
from pyspark.sql.types import StructType, StructField, StringType, FloatType

spark = SparkSession.builder.appName("AirQualitySilver").getOrCreate()

# Doc từ Bronze layer
df_bronze = spark.readStream \
    .format("json") \
    .schema("data STRUCT<iaqi: STRING, time: STRING>, fetched_at STRING") \
    .load("hdfs://lakehouse/bronze/streaming/air_quality/")

iaqi_schema = StructType([
    StructField("co", StructType([StructField("v", FloatType())])),
    StructField("no2", StructType([StructField("v", FloatType())])),
    StructField("o3", StructType([StructField("v", FloatType())])),
    StructField("pm10", StructType([StructField("v", FloatType())])),
    StructField("pm25", StructType([StructField("v", FloatType())])),
    StructField("so2", StructType([StructField("v", FloatType())]))
])

time_schema = StructType([
    StructField("s", StringType()),
    StructField("tz", StringType()),
    StructField("v", StringType()),
    StructField("iso", StringType())
])

df_silver = df_bronze \
    .withColumn("iaqi_json", from_json(col("data.iaqi"), iaqi_schema)) \
    .withColumn("time_json", from_json(col("data.time"), time_schema)) \
    .withColumn("date", to_timestamp(col("fetched_at"))) \
    .select(
        "date",
        col("iaqi_json.co.v").alias("CO"),
        col("iaqi_json.no2.v").alias("NO2"),
        col("iaqi_json.o3.v").alias("O3"),
        col("iaqi_json.pm10.v").alias("PM10"),
        col("iaqi_json.pm25.v").alias("PM25"),
        col("iaqi_json.so2.v").alias("SO2")
    )

```

1,1      Top

Chạy file bằng Spark-submit:

```

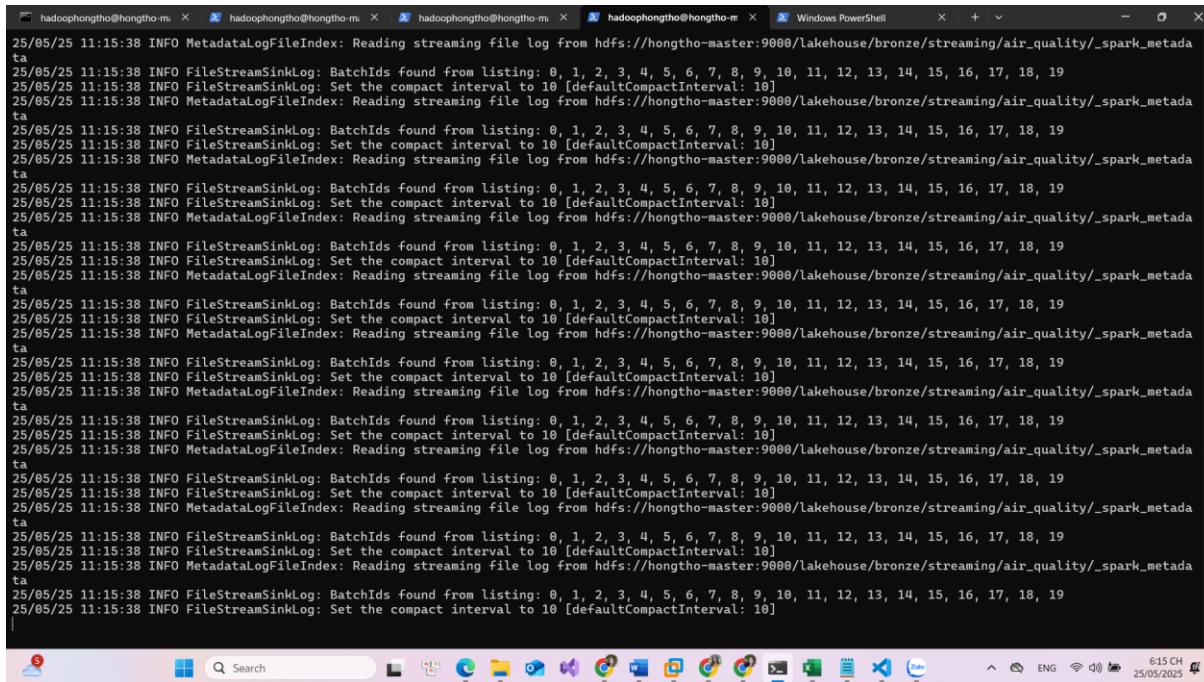
$ spark-submit --packages io.delta:delta-spark_2.12:3.3.1 --conf
spark.sql.extensions=io.delta.sql.DeltaSparkSessionExtension" --conf
"spark.sql.catalog.spark_catalog=org.apache.spark.sql.delta.catalog.DeltaCatalog
LoadStreamingToSilver.py

```

```

hadoophongtho@hongtho-master:~/spark$ spark-submit --packages io.delta:delta-spark_2.12:3.3.1 --conf "spark.sql.extensions=io.delta.sql.DeltaSparkSessionExtension" --conf "spark.sql.catalog.spark_catalog=org.apache.spark.sql.delta.catalog.DeltaCatalog" LoadStreamingToSilver.py
:: loading settings :: url = jar:file:/home/hadoophongtho/spark/jars/ivy-2.5.1.jar!/org/apache/ivy/core/settings/ivysettings.xml
ivy Default Cache set to: /home/hadoophongtho/.ivy2/cache
The jars for the packages stored in: /home/hadoophongtho/.ivy2/jars
io.delta#delta-spark_2.12 added as a dependency
:: resolving dependencies :: org.apache.spark#spark-submit-parent-9a6cf8a7-209c-412f-8a57-de659c76de93;1.0
  confs: [default]
  found io.delta#delta-spark_2.12;3.3.1 in central
  found io.delta#delta-storage;3.3.1 in central
  found org.antlr#antlr4-runtime;4.9.3 in local-m2-cache
:: resolution report :: resolve 149ms :: artifacts dl 7ms
  :: modules in use:
    io.delta#delta-spark_2.12;3.3.1 from central in [default]
    io.delta#delta-storage;3.3.1 from central in [default]
    org.antlr#antlr4-runtime;4.9.3 from local-m2-cache in [default]
-----
	modules		artifacts					
conf	number	search	dwnlded	evicted		number	dwnldn	
default	3	0	0	0	0		3	0
-----
:: retrieving :: org.apache.spark#spark-submit-parent-9a6cf8a7-209c-412f-8a57-de659c76de93
  confs: [default]
  0 artifacts copied, 3 already retrieved (0kB/7ms)
25/05/25 11:14:34 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
25/05/25 11:14:34 INFO SparkContext: Running Spark version 3.5.3
25/05/25 11:14:34 INFO SparkContext: OS info Linux, 6.8.0-57-generic, amd64
25/05/25 11:14:34 INFO SparkContext: Java version 1.8.0_452
25/05/25 11:14:34 INFO ResourceUtils: =====
25/05/25 11:14:34 INFO ResourceUtils: No custom resources configured for spark driver
25/05/25 11:14:34 INFO ResourceUtils: =====
25/05/25 11:14:34 INFO SparkContext: Submitted application: LoadStreamingToSilverApp
25/05/25 11:14:34 INFO ResourceProfile: Default ResourceProfile created, executor resources: Map(cores -> name: cores, amount: 1, script: , vendor: , memory

```



Kiểm tra lại bằng cách đọc một luồng thay đổi từ bảng Delta (Read a Stream of Changes from a Table) trực tiếp trong PySpark:

```
$ pyspark --packages io.delta:delta-spark_2.12:3.3.1 --conf  
"spark.sql.extensions=io.delta.sql.DeltaSparkSessionExtension" --conf  
"spark.sql.catalog.spark_catalog=org.apache.spark.sql.delta.catalog.DeltaCatalog"
```

Trong khi một luồng đang ghi dữ liệu vào bảng Delta, cũng có thể đọc dữ liệu từ bảng đó theo dạng streaming (liên tục). Điều này rất hữu ích để theo dõi thay đổi theo thời gian thực:

```
streamingDF = spark.readStream.format("delta") \
    .load("hdfs:///lakehouse/silver/streaming/air_quality/")
query = streamingDF.writeStream.format("console") \
```

```

.option("truncate", "false") \
.start()
query.awaitTermination()

```

```

hadoopphongtho@hongtho-m ~ hadoopphongtho@hongtho-m ~ hadoopphongtho@hongtho-m ~ hadoopphongtho@hongtho-m ~ hadoopphongtho@hongtho-m ~ + - o x
ay":25,"month":5},{"date":"2025-05-25T11:21:47.448Z","CO":0.0,"NO2":4.0,"O3":22.0,"TSP":18.0,"PM25":22.0,"SO2":49.0,"AQI_Level":"Average","hour":11,"day":25,"month":5}, {"date":null,"CO":null,"NO2":null,"O3":null,"TSP":null,"PM25":null,"SO2":null,"AQI_Level":null,"hour":null,"day":null,"month":None}))).  

>>> query = streamingDF.writeStream.format("console") \  

... .option("truncate", "false") \  

... .start()  

25/05/25 12:08:34 WARN ResolveWriteToStream: Temporary checkpoint location created which is deleted normally when the query didn't fail: /tmp/temporary-7213  

244f-506b-44c0-8dfb-e5051028722f. If it's required to delete it under any circumstances, please set spark.sql.streaming.forceDeleteTempCheckpointLocation to  

true. Important to know deleting temp checkpoint folder is best effort.  

25/05/25 12:08:34 WARN ResolveWriteToStream: spark.sql.adaptive.enabled is not supported in streaming DataFrames/Datasets and will be disabled.  

>>> query.awaitTermination()  

Batch: 0  

+-----+  

|date |CO |NO2|O3 |TSP |PM25|SO2 |AQI_Level|hour|day|month|  

+-----+  

|2025-05-25 11:16:45.015018|0.0|14.0|22.0|18.0|27.0|49.0|Average|11|25|5|  

|2025-05-25 11:08:42.067857|0.0|14.0|22.0|18.0|27.0|49.0|Average|11|25|5|  

|2025-05-25 11:15:44.677035|0.0|14.0|22.0|18.0|27.0|49.0|Average|11|25|5|  

|2025-05-25 11:03:46.192643|0.0|14.0|22.0|18.0|27.0|49.0|Average|11|25|5|  

|2025-05-25 10:57:37.334403|0.0|14.0|22.0|18.0|27.0|49.0|Average|10|25|5|  

|2025-05-25 11:05:46.926245|0.0|14.0|22.0|18.0|27.0|49.0|Average|11|25|5|  

|2025-05-25 11:21:47.448476|0.0|14.0|22.0|18.0|27.0|49.0|Average|11|25|5|  

|2025-05-25 11:13:43.935989|0.0|14.0|22.0|18.0|27.0|49.0|Average|11|25|5|  

|2025-05-25 11:02:39.76879|0.0|14.0|22.0|18.0|27.0|49.0|Average|11|25|5|  

|2025-05-25 11:06:41.308569|0.0|14.0|22.0|18.0|27.0|49.0|Average|11|25|5|  

|2025-05-25 10:58:37.785232|0.0|14.0|22.0|18.0|27.0|49.0|Average|10|25|5|  

|2025-05-25 11:34:52.385946|0.0|14.0|22.0|18.0|27.0|49.0|Average|11|25|5|  

|2025-05-25 11:07:41.692527|0.0|14.0|22.0|18.0|27.0|49.0|Average|11|25|5|  

|2025-05-25 11:19:46.566365|0.0|14.0|22.0|18.0|27.0|49.0|Average|11|25|5|  

|2025-05-25 11:23:48.189785|0.0|14.0|22.0|18.0|27.0|49.0|Average|11|25|5|  

|2025-05-25 11:11:43.193848|0.0|14.0|22.0|18.0|27.0|49.0|Average|11|25|5|  

|2025-05-25 11:18:42.811217|0.0|14.0|22.0|18.0|27.0|49.0|Average|11|25|5|  

|2025-05-25 11:04:46.561939|0.0|14.0|22.0|18.0|27.0|49.0|Average|11|25|5|  

|2025-05-25 11:09:42.440869|0.0|14.0|22.0|18.0|27.0|49.0|Average|11|25|5|  

|2025-05-25 11:17:45.423028|0.0|14.0|22.0|18.0|27.0|49.0|Average|11|25|5|  

+-----+  

only showing top 20 rows
| |

```

Qua khung giờ mới dữ liệu sẽ được cập nhật:

```

Batch: 0
-----
+-----+-----+-----+-----+-----+-----+-----+
|date |CO |NO2|O3 |TSP |PM25|SO2 |AQI_Level|hour|day|month|
+-----+-----+-----+-----+-----+-----+-----+
2025-05-25 11:16:45.045018	0.0	4.0	22.0	18.0	27.0	49.0	Average	11	25	5
2025-05-25 11:08:42.067857	0.0	4.0	22.0	18.0	27.0	49.0	Average	11	25	5
2025-05-25 11:15:44.677035	0.0	4.0	22.0	18.0	27.0	49.0	Average	11	25	5
2025-05-25 11:03:40.192643	0.0	4.0	22.0	18.0	27.0	49.0	Average	11	25	5
2025-05-25 10:57:37.334403	0.0	4.0	22.0	18.0	27.0	49.0	Average	10	25	5
2025-05-25 11:05:40.926245	0.0	4.0	22.0	18.0	27.0	49.0	Average	11	25	5
2025-05-25 11:21:47.448476	0.0	4.0	22.0	18.0	22.0	49.0	Average	11	25	5
2025-05-25 11:13:43.935989	0.0	4.0	22.0	18.0	27.0	49.0	Average	11	25	5
2025-05-25 11:02:39.76879	0.0	4.0	22.0	18.0	27.0	49.0	Average	11	25	5
2025-05-25 11:06:41.308569	0.0	4.0	22.0	18.0	27.0	49.0	Average	11	25	5
2025-05-25 10:58:37.705232	0.0	4.0	22.0	18.0	27.0	49.0	Average	10	25	5
2025-05-25 11:34:52.385946	0.0	4.0	22.0	18.0	22.0	49.0	Average	11	25	5
2025-05-25 11:07:41.692527	0.0	4.0	22.0	18.0	27.0	49.0	Average	11	25	5
2025-05-25 11:19:46.566365	0.0	4.0	22.0	18.0	22.0	49.0	Average	11	25	5
2025-05-25 11:23:48.189705	0.0	4.0	22.0	18.0	22.0	49.0	Average	11	25	5
2025-05-25 11:11:43.193048	0.0	4.0	22.0	18.0	27.0	49.0	Average	11	25	5
2025-05-25 11:10:42.811217	0.0	4.0	22.0	18.0	27.0	49.0	Average	11	25	5
2025-05-25 11:04:40.561939	0.0	4.0	22.0	18.0	27.0	49.0	Average	11	25	5
2025-05-25 11:09:42.440969	0.0	4.0	22.0	18.0	27.0	49.0	Average	11	25	5
2025-05-25 11:17:45.423028	0.0	4.0	22.0	18.0	22.0	49.0	Average	11	25	5
+-----+-----+-----+-----+-----+-----+-----+
only showing top 20 rows

```

```

Batch: 1
-----
+-----+-----+-----+-----+-----+-----+-----+
|date |CO |NO2|O3 |TSP |PM25|SO2 |AQI_Level|hour|day|month|
+-----+-----+-----+-----+-----+-----+-----+
|2025-05-25 12:09:05.953846|0.0|4.0|22.0|18.0|22.0|49.0|Average |12 |25 |5 |
+-----+-----+-----+-----+-----+-----+-----+

```

Chạy song song 5 Terminal:

```
[2025-05-25 09:23:03,544] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from __consumer_offsets-21 in 23 milliseconds f or epoch 0, of which 23 milliseconds was spent in the scheduler. (kafka.coordinator.group.GroupMetadataManager)
[2025-05-25 09:23:03,544] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from __consumer_offsets-24 in 23 milliseconds f or epoch 0, of which 23 milliseconds was spent in the scheduler. (kafka.coordinator.group.GroupMetadataManager)
[2025-05-25 09:23:03,544] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from __consumer_offsets-27 in 23 milliseconds f or epoch 0, of which 23 milliseconds was spent in the scheduler. (kafka.coordinator.group.GroupMetadataManager)
[2025-05-25 09:23:03,545] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from __consumer_offsets-30 in 23 milliseconds f or epoch 0, of which 23 milliseconds was spent in the scheduler. (kafka.coordinator.group.GroupMetadataManager)
[2025-05-25 09:23:03,545] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from __consumer_offsets-33 in 24 milliseconds f or epoch 0, of which 24 milliseconds was spent in the scheduler. (kafka.coordinator.group.GroupMetadataManager)
[2025-05-25 09:23:03,545] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from __consumer_offsets-36 in 24 milliseconds f or epoch 0, of which 24 milliseconds was spent in the scheduler. (kafka.coordinator.group.GroupMetadataManager)
[2025-05-25 09:23:03,545] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from __consumer_offsets-39 in 24 milliseconds f or epoch 0, of which 24 milliseconds was spent in the scheduler. (kafka.coordinator.group.GroupMetadataManager)
[2025-05-25 09:23:03,545] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from __consumer_offsets-42 in 24 milliseconds f or epoch 0, of which 24 milliseconds was spent in the scheduler. (kafka.coordinator.group.GroupMetadataManager)
[2025-05-25 09:23:03,546] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from __consumer_offsets-45 in 25 milliseconds f or epoch 0, of which 25 milliseconds was spent in the scheduler. (kafka.coordinator.group.GroupMetadataManager)
[2025-05-25 09:23:03,546] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from __consumer_offsets-48 in 25 milliseconds f or epoch 0, of which 25 milliseconds was spent in the scheduler. (kafka.coordinator.group.GroupMetadataManager)
[2025-05-25 10:25:02,956] INFO Creating topic air-quality_2 with configuration {} and initial partition assignment Map(0 -> ArrayBuffer(0)) (kafka.zk.AdminZ kClient)
[2025-05-25 10:25:03,031] INFO [ReplicaFetcherManager on broker 0] Removed fetcher for partitions Set(air-quality_2-0) (kafka.server.ReplicaFetcherManager)
[2025-05-25 10:25:03,035] INFO [Log partition=air-quality_2-0, dir=/tmp/kafka-logs] Loading producer state till offset 0 with message format version 2 (kafka.log.Log)
[2025-05-25 10:25:03,036] INFO Created log for partition air-quality_2-0 in /tmp/kafka-logs/air-quality_2-0 with properties {} (kafka.log.LogManager)
[2025-05-25 10:25:03,037] INFO [Partition air-quality_2-0 broker=0] No checkpointed highwatermark is found for partition air-quality_2-0 (kafka.cluster.Part ition)
[2025-05-25 10:25:03,037] INFO [Partition air-quality_2-0 broker=0] Log loaded for partition air-quality_2-0 with initial high watermark 0 (kafka.cluster.Pa rtition)
[2025-05-25 10:25:28,488] INFO Creating topic air-quality-2 with configuration {} and initial partition assignment Map(0 -> ArrayBuffer(0)) (kafka.zk.AdminZ kClient)
[2025-05-25 10:25:28,522] INFO [ReplicaFetcherManager on broker 0] Removed fetcher for partitions Set(air-quality-2-0) (kafka.server.ReplicaFetcherManager)
[2025-05-25 10:25:28,529] INFO [Log partition=air-quality-2-0, dir=/tmp/kafka-logs] Loading producer state till offset 0 with message format version 2 (kafka.log.Log)
[2025-05-25 10:25:28,531] INFO Created log for partition air-quality-2-0 in /tmp/kafka-logs/air-quality-2-0 with properties {} (kafka.log.LogManager)
[2025-05-25 10:25:28,533] INFO [Partition air-quality-2-0 broker=0] No checkpointed highwatermark is found for partition air-quality-2-0 (kafka.cluster.Part ition)
[2025-05-25 10:25:28,533] INFO [Partition air-quality-2-0 broker=0] Log loaded for partition air-quality-2-0 with initial high watermark 0 (kafka.cluster.Pa rtition)
|
```



```
KeyboardInterrupt

(myenv) hadoophongtho@hongtho-master:~/project$ vim air_quality_kafka_producer.py
(myenv) hadoophongtho@hongtho-master:~/project$ python3 air_quality_kafka_producer.py
Sent data to Kafka: 2025-05-25T10:59:41.144680
Sent data to Kafka: 2025-05-25T11:00:41.513750
Sent data to Kafka: 2025-05-25T11:01:41.881947
^CTraceback (most recent call last):
  File "/home/hadoophongtho/project/air_quality_kafka_producer.py", line 32, in <module>
    time.sleep(60)
KeyboardInterrupt

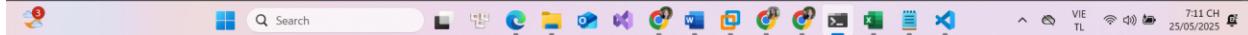
(myenv) hadoophongtho@hongtho-master:~/project$ vim air_quality_kafka_producer.py
(myenv) hadoophongtho@hongtho-master:~/project$ python3 air_quality_kafka_producer.py
Sent data to Kafka: 2025-05-25T11:02:39.768790
Sent data to Kafka: 2025-05-25T11:03:40.192643
Sent data to Kafka: 2025-05-25T11:04:40.561939
Sent data to Kafka: 2025-05-25T11:05:40.926245
Sent data to Kafka: 2025-05-25T11:06:41.308569
Sent data to Kafka: 2025-05-25T11:07:41.692527
Sent data to Kafka: 2025-05-25T11:08:42.067857
Sent data to Kafka: 2025-05-25T11:09:42.440969
Sent data to Kafka: 2025-05-25T11:10:42.811217
Sent data to Kafka: 2025-05-25T11:11:43.193048
Sent data to Kafka: 2025-05-25T11:12:43.563078
Sent data to Kafka: 2025-05-25T11:13:43.935988
Sent data to Kafka: 2025-05-25T11:14:44.389318
Sent data to Kafka: 2025-05-25T11:15:44.677035
Sent data to Kafka: 2025-05-25T11:16:45.045018
Sent data to Kafka: 2025-05-25T11:17:45.423028
Sent data to Kafka: 2025-05-25T11:18:46.177651
Sent data to Kafka: 2025-05-25T11:19:46.566365
Sent data to Kafka: 2025-05-25T11:20:47.069577
Sent data to Kafka: 2025-05-25T11:21:47.448476
Sent data to Kafka: 2025-05-25T11:22:47.824411
Sent data to Kafka: 2025-05-25T11:23:48.189765
Sent data to Kafka: 2025-05-25T11:34:52.385946
Sent data to Kafka: 2025-05-25T12:09:05.953846
Sent data to Kafka: 2025-05-25T12:10:06.349733
Sent data to Kafka: 2025-05-25T12:11:06.784205
|
```



```
"getBatch" : 0,
"latestOffset" : 4,
"queryPlanning" : 39,
"triggerExecution" : 850,
"walCommit" : 43
},
"stateOperators" : [ ],
"sources" : [ {
"description" : "KafkaV2[Subscribe[air-quality]]",
"startOffset" : {
"air-quality" : {
"0" : 80
}
},
"endOffset" : {
"air-quality" : {
"0" : 81
}
},
"latestOffset" : {
"air-quality" : {
"0" : 81
}
},
"numInputRows" : 1,
"inputRowsPerSecond" : 90.90909090999992,
"processedRowsPerSecond" : 1.1764705882352942,
"metrics" : {
"avgOffsetsBehindLatest" : "0.0",
"maxOffsetsBehindLatest" : "0",
"minOffsetsBehindLatest" : "0"
}
],
"sink" : {
"description" : "FileSink[/lakehouse/bronze/streaming/air_quality/]",
"numOutputRows" : -1
}
}
],
25/05/25 12:11:12 INFO BlockManagerInfo: Removed broadcast_5_piece0 on hongtho-master:44489 in memory (size: 86.0 Kib, free: 366.2 Mib)
25/05/25 12:11:17 INFO MicroBatchExecution: Streaming query has been idle and waiting for new data more than 10000 ms.
|
```



```
25/05/25 12:11:30 INFO MetadataLogFileIndex: Reading streaming file log from hdfs://hongtho-master:9000/lakehouse/bronze/streaming/air_quality/_spark_metadata
ta
25/05/25 12:11:30 INFO FileStreamSinkLog: BatchIds found from listing: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23,
24, 25, 26
25/05/25 12:11:30 INFO FileStreamSinkLog: Set the compact interval to 10 [defaultCompactInterval: 10]
25/05/25 12:11:30 INFO MetadataLogFileIndex: Reading streaming file log from hdfs://hongtho-master:9000/lakehouse/bronze/streaming/air_quality/_spark_metadata
ta
25/05/25 12:11:30 INFO FileStreamSinkLog: BatchIds found from listing: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23,
24, 25, 26
25/05/25 12:11:30 INFO FileStreamSinkLog: Set the compact interval to 10 [defaultCompactInterval: 10]
25/05/25 12:11:30 INFO MetadataLogFileIndex: Reading streaming file log from hdfs://hongtho-master:9000/lakehouse/bronze/streaming/air_quality/_spark_metadata
ta
25/05/25 12:11:30 INFO FileStreamSinkLog: BatchIds found from listing: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23,
24, 25, 26
25/05/25 12:11:30 INFO FileStreamSinkLog: Set the compact interval to 10 [defaultCompactInterval: 10]
25/05/25 12:11:30 INFO MetadataLogFileIndex: Reading streaming file log from hdfs://hongtho-master:9000/lakehouse/bronze/streaming/air_quality/_spark_metadata
ta
25/05/25 12:11:30 INFO FileStreamSinkLog: BatchIds found from listing: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23,
24, 25, 26
25/05/25 12:11:30 INFO FileStreamSinkLog: Set the compact interval to 10 [defaultCompactInterval: 10]
25/05/25 12:11:30 INFO MetadataLogFileIndex: Reading streaming file log from hdfs://hongtho-master:9000/lakehouse/bronze/streaming/air_quality/_spark_metadata
ta
25/05/25 12:11:30 INFO FileStreamSinkLog: BatchIds found from listing: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23,
24, 25, 26
25/05/25 12:11:30 INFO FileStreamSinkLog: Set the compact interval to 10 [defaultCompactInterval: 10]
25/05/25 12:11:30 INFO MetadataLogFileIndex: Reading streaming file log from hdfs://hongtho-master:9000/lakehouse/bronze/streaming/air_quality/_spark_metadata
ta
25/05/25 12:11:30 INFO FileStreamSinkLog: BatchIds found from listing: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23,
24, 25, 26
25/05/25 12:11:30 INFO FileStreamSinkLog: Set the compact interval to 10 [defaultCompactInterval: 10]
25/05/25 12:11:30 INFO MetadataLogFileIndex: Reading streaming file log from hdfs://hongtho-master:9000/lakehouse/bronze/streaming/air_quality/_spark_metadata
ta
25/05/25 12:11:30 INFO FileStreamSinkLog: BatchIds found from listing: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23,
24, 25, 26
25/05/25 12:11:30 INFO FileStreamSinkLog: Set the compact interval to 10 [defaultCompactInterval: 10]
25/05/25 12:11:30 INFO MetadataLogFileIndex: Reading streaming file log from hdfs://hongtho-master:9000/lakehouse/bronze/streaming/air_quality/_spark_metadata
ta
25/05/25 12:11:30 INFO FileStreamSinkLog: BatchIds found from listing: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23,
24, 25, 26
25/05/25 12:11:30 INFO FileStreamSinkLog: Set the compact interval to 10 [defaultCompactInterval: 10]
25/05/25 12:11:30 INFO MetadataLogFileIndex: Reading streaming file log from hdfs://hongtho-master:9000/lakehouse/bronze/streaming/air_quality/_spark_metadata
ta
25/05/25 12:11:30 INFO FileStreamSinkLog: BatchIds found from listing: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23,
24, 25, 26
25/05/25 12:11:30 INFO FileStreamSinkLog: Set the compact interval to 10 [defaultCompactInterval: 10]
25/05/25 12:11:30 INFO MetadataLogFileIndex: Reading streaming file log from hdfs://hongtho-master:9000/lakehouse/bronze/streaming/air_quality/_spark_metadata
ta
|
```



```
| hadoopphongtho@hongtho-mi | hadoopphongtho@hongtho-mi | hadoopphongtho@hongtho-mi | hadoopphongtho@hongtho-mi | hadoopphongtho@hongtho-mi | hadoopphongtho@hongtho-mi | + -+  
2025-05-25 10:58:37.705232	0.0	4.0	22.0	18.0	27.0	49.0	Average	10	25	5
2025-05-25 11:34:52.385946	0.0	4.0	22.0	18.0	22.0	49.0	Average	11	25	5
2025-05-25 11:07:41.692527	0.0	4.0	22.0	18.0	27.0	49.0	Average	11	25	5
2025-05-25 11:19:46.566365	0.0	4.0	22.0	18.0	22.0	49.0	Average	11	25	5
2025-05-25 11:23:48.189705	0.0	4.0	22.0	18.0	22.0	49.0	Average	11	25	5
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
only showing top 20 rows  
  
Batch: 1  
-----  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
|date |CO |NO2|O3 |TSP |PM25|SO2 |AQI_Level|hour|day|month|  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
|2025-05-25 12:09:05.958346|0.0|4.0|22.0|18.0|22.0|49.0|Average |12 |25 |5 |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
  
Batch: 2  
-----  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
|date |CO |NO2|O3 |TSP |PM25|SO2 |AQI_Level|hour|day|month|  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
|2025-05-25 12:10:06.349733|0.0|4.0|22.0|18.0|22.0|49.0|Average |12 |25 |5 |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
  
Batch: 3  
-----  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
|date |CO |NO2|O3 |TSP |PM25|SO2 |AQI_Level|hour|day|month|  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
|2025-05-25 12:11:06.784205|0.0|4.0|22.0|18.0|22.0|49.0|Average |12 |25 |5 |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
  
-----  
Batch: 3  
-----  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
|date |CO |NO2|O3 |TSP |PM25|SO2 |AQI_Level|hour|day|month|  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
|2025-05-25 13:21:20.168776|0.0|5.0|22.0|18.0|14.0|49.0|Average |13 |25 |5 |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
```

Đẩy data lên lớp Gold tương tự như data dạng Batch:

```
$ vim LoadStreamingToGold.py
```

```
hadoopphongtho@hongtho-master:~/spark$ vim LoadStreamingToGold.py
```

```

from pyspark.sql import SparkSession
from pyspark.sql.functions import avg, to_date, when, col, month, hour

spark = SparkSession.builder \
    .appName("Gold Streaming AQI Average") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog", "org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .getOrCreate()

df_silver = spark.read.format("delta").load("hdfs:///lakehouse/silver/streaming/air_quality/")

# 1. Daily average
df_daily = df_silver.withColumn("date_only", to_date("date")) \
    .groupBy("date_only") \
    .agg(
        avg("PM25").alias("avg_PM25")
    ) \
    .withColumn(
        "AQI_Level",
        when(col("avg_PM25") <= 12, "Good") \
            .when((col("avg_PM25") > 12) & (col("avg_PM25") <= 35), "Average") \
            .when((col("avg_PM25") > 35) & (col("avg_PM25") <= 55), "Moderate") \
            .otherwise("Unhealthy")
    )
df_daily.write.format("delta") \
    .mode("overwrite") \
    .save("hdfs:///lakehouse/gold/streaming/air_quality_daily_avg")

# 2. Hourly average
df_hourly = df_silver.withColumn("hour", hour("date")) \
    .groupBy("hour") \
    .agg(
        avg("PM25").alias("avg_PM25"),
        avg("CO").alias("avg_CO")
    ) \
    .orderBy("hour")
df_hourly.write.format("delta") \
    .mode("overwrite") \

```

1,24 Top

```

from pyspark.sql import SparkSession
from pyspark.sql.functions import avg, to_date, when, col, month, hour

spark = SparkSession.builder \
    .appName("Gold Streaming AQI Average") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog",
"org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .getOrCreate()

df_silver =
spark.read.format("delta").load("hdfs:///lakehouse/silver/streaming/air_quality/")

# 1. Daily average
df_daily = df_silver.withColumn("date_only", to_date("date")) \
    .groupBy("date_only") \
    .agg(
        avg("PM25").alias("avg_PM25")
    ) \
    .withColumn(
        "AQI_Level",
        when(col("avg_PM25") <= 12, "Good") \
            .when((col("avg_PM25") > 12) & (col("avg_PM25") <= 35), "Average") \
            .when((col("avg_PM25") > 35) & (col("avg_PM25") <= 55), "Moderate") \
            .otherwise("Unhealthy")
    )
df_daily.write.format("delta") \
    .mode("overwrite") \
    .save("hdfs:///lakehouse/gold/streaming/air_quality_daily_avg")

# 2. Hourly average
df_hourly = df_silver.withColumn("hour", hour("date")) \
    .groupBy("hour") \

```

```

    .agg(
        avg("PM25").alias("avg_PM25"),
        avg("CO").alias("avg_CO")
    ) \
    .orderBy("hour")

df_hourly.write.format("delta") \
    .mode("overwrite") \
    .save("hdfs://lakehouse/gold/streaming/air_quality_hourly_avg")

# 3. Monthly average
df_monthly = df_silver.withColumn("month", month("date")) \
    .groupBy("month") \
    .agg(
        avg("PM25").alias("avg_PM25"),
        avg("CO").alias("avg_CO"),
        avg("O3").alias("avg_O3"),
        avg("SO2").alias("avg_SO2")
    )

df_monthly.write.format("delta") \
    .mode("overwrite") \
    .save("hdfs://lakehouse/gold/streaming/air_quality_monthly_avg")

```

Chạy:

```

$ spark-submit --packages io.delta:delta-spark_2.12:3.3.1 --conf
spark.sql.extensions=io.delta.sql.DeltaSparkSessionExtension" --conf
"spark.sql.catalog.spark_catalog=org.apache.spark.sql.delta.catalog.DeltaCatalog
LoadStreamingToGold.py

```

```

hadoop@hongtho-master:~/spark$ spark-submit --packages io.delta:delta-spark_2.12:3.3.1 --conf spark.sql.extensions=io.delta.sql.DeltaSparkSessionExtension" --conf "spark.sql.catalog.spark_catalog=org.apache.spark.sql.delta.catalog.DeltaCatalog LoadStreamingToGold.py
:: loading settings :: url = jar:file:/home/hadoop@hongtho-spark/jars/ivy-2.5.1.jar!/org/apache/ivy/core/settings/ivysettings.xml
Ivy Default Cache set to: /home/hadoop@hongtho/.ivy2/cache
The jars for the packages stored in: /home/hadoop@hongtho/.ivy2/jars
io.delta#delta-spark_2.12 added as a dependency
:: resolving dependencies :: org.apache.spark#spark-submit-parent-9f138e14-30c6-48f9-afe6-2b57924bfa17;1.0
  confs: [default]
    found io.delta#delta-spark_2.12;3.3.1 in central
    found io.delta#delta-storage;3.3.1 in central
    found org.antlr#antlr4-runtime;4.9.3 in local-m2-cache
:: resolution report :: resolve 483ms :: artifacts dl 25ms
  :: modules in use:
    io.delta#delta-spark_2.12;3.3.1 from central in [default]
    io.delta#delta-storage;3.3.1 from central in [default]
    org.antlr#antlr4-runtime;4.9.3 from local-m2-cache in [default]
    |-----|-----|-----|-----|
    | conf | number| modules | artifacts | | | | |
|---|---|---|---|---|---|---|---|
    | default | 3 | 0 | 0 | 0 || 3 | 0 |
  :: retrieving :: org.apache.spark#spark-submit-parent-9f138e14-30c6-48f9-afe6-2b57924bfa17
  confs: [default]
  0 artifacts copied, 3 already retrieved (0kB/18ms)
25/05/25 13:17:46 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
25/05/25 13:17:48 INFO SparkContext: Running Spark version 3.5.3
25/05/25 13:17:48 INFO SparkContext: OS info Linux, 6.8.0-57-generic, amd64
25/05/25 13:17:48 INFO SparkContext: Java version 1.8.0_452
25/05/25 13:17:48 INFO ResourceUtils: =====
25/05/25 13:17:48 INFO ResourceUtils: No custom resources configured for spark.driver.
25/05/25 13:17:48 INFO ResourceUtils: =====
25/05/25 13:17:48 INFO SparkContext: Submitted application: Gold Streaming AQI Average
25/05/25 13:17:48 INFO ResourceProfile: Default ResourceProfile created, executor resources: Map(cores -> name: cores, amount: 1, script: , vendor: , memory

```

```

hadoopphongtho@hongtho:~$ hadoopphongtho@hongtho:~$ hadoopphongtho@hongtho:~$ hadoopphongtho@hongtho:~$ hadoopphongtho@hongtho:~$ hadoopphongtho@hongtho:~$ hadoopphongtho@hongtho:~$ + - 
-bc41-b5b5b2e5c182/userFiles-98ef3870-08a9-4c5b-9cc9-010bcf701dba/io.delta_delta-spark_2.12-3.3.1.jar
25/05/25 13:22:05 INFO Executor: Fetching file://home/hadoopphongtho/.ivy2/jars/io.delta_delta-storage-3.3.1.jar with timestamp 1748179322537
25/05/25 13:22:05 INFO Executor: /home/hadoopphongtho/.ivy2/jars/io.delta_delta-storage-3.3.1.jar has been previously copied to /tmp/spark-b2b1b0e5-5b53-4ca2-bc41-b5b5b2e5c182/userFiles-98ef3870-08a9-4c5b-9cc9-010bcf701dba/io.delta_delta-storage-3.3.1.jar
25/05/25 13:22:05 INFO Executor: Fetching spark://hongtho-master:34737/jars/io.delta_delta-storage-3.3.1.jar with timestamp 1748179322537
25/05/25 13:22:05 INFO TransportClientFactory: Successfully created connection to hongtho-master/192.168.232.12:34737 after 59 ms (0 ms spent in bootstraps)
25/05/25 13:22:05 INFO Executor: Fetching spark://hongtho-master:34737/jars/io.delta_delta-storage-3.3.1.jar to /tmp/spark-b2b1b0e5-5b53-4ca2-bc41-b5b5b2e5c182/userFiles-98ef3870-08a9-4c5b-9cc9-010bcf701dba/fetchFileTemp922145344294530214.tmp
25/05/25 13:22:05 INFO Utils: /tmp/spark-b2b1b0e5-5b53-4ca2-bc41-b5b5b2e5c182/userFiles-98ef3870-08a9-4c5b-9cc9-010bcf701dba/fetchFileTemp922145344294530214.tmp has been previously copied to /tmp/spark-b2b1b0e5-5b53-4ca2-bc41-b5b5b2e5c182/userFiles-98ef3870-08a9-4c5b-9cc9-010bcf701dba/io.delta_delta-storage-3.3.1.jar
25/05/25 13:22:05 INFO Executor: Adding file:/tmp/spark-b2b1b0e5-5b53-4ca2-bc41-b5b5b2e5c182/userFiles-98ef3870-08a9-4c5b-9cc9-010bcf701dba/io.delta_delta-storage-3.3.1.jar to class loader default
25/05/25 13:22:05 INFO Executor: Fetching spark://hongtho-master:34737/jars/io.delta_delta-spark_2.12-3.3.1.jar with timestamp 1748179322537
25/05/25 13:22:05 INFO Utils: /tmp/spark-b2b1b0e5-5b53-4ca2-bc41-b5b5b2e5c182/userFiles-98ef3870-08a9-4c5b-9cc9-010bcf701dba/fetchFileTemp922145344294530214.tmp has been previously copied to /tmp/spark-b2b1b0e5-5b53-4ca2-bc41-b5b5b2e5c182/userFiles-98ef3870-08a9-4c5b-9cc9-010bcf701dba/io.delta_delta-spark_2.12-3.3.1.jar
25/05/25 13:22:05 INFO Executor: Adding file:/tmp/spark-b2b1b0e5-5b53-4ca2-bc41-b5b5b2e5c182/userFiles-98ef3870-08a9-4c5b-9cc9-010bcf701dba/io.delta_delta-spark_2.12-3.3.1.jar to class loader default
25/05/25 13:22:05 INFO Executor: Fetching spark://hongtho-master:34737/jars/org.antlr.antlr4-runtime-4.9.3.jar with timestamp 1748179322537
25/05/25 13:22:05 INFO Utils: /tmp/spark-b2b1b0e5-5b53-4ca2-bc41-b5b5b2e5c182/userFiles-98ef3870-08a9-4c5b-9cc9-010bcf701dba/fetchFileTemp922145344294530214.tmp has been previously copied to /tmp/spark-b2b1b0e5-5b53-4ca2-bc41-b5b5b2e5c182/userFiles-98ef3870-08a9-4c5b-9cc9-010bcf701dba/org.antlr.antlr4-runtime-4.9.3.jar
25/05/25 13:22:05 INFO Executor: Adding file:/tmp/spark-b2b1b0e5-5b53-4ca2-bc41-b5b5b2e5c182/userFiles-98ef3870-08a9-4c5b-9cc9-010bcf701dba/org.antlr.antlr4-runtime-4.9.3.jar to class loader default
25/05/25 13:22:05 INFO Utils: Successfully started service 'org.apache.spark.network.netty.NettyBlockTransferService' on port 44805.
25/05/25 13:22:05 INFO NettyBlockTransferService: Server created on hongtho-master:44805
25/05/25 13:22:05 INFO BlockManager: Using org.apache.spark.storage.RandomBlockReplicationPolicy for block replication policy
25/05/25 13:22:05 INFO BlockManagerMaster: Registering BlockManager BlockManagerId(driver, hongtho-master, 44805, None)
25/05/25 13:22:05 INFO BlockManagerMasterEndpoint: Registering block manager hongtho-master:44805 with 366.3 MiB RAM, BlockManagerId(driver, hongtho-master, 44805, None)
25/05/25 13:22:05 INFO BlockManagerMaster: Registered BlockManager BlockManagerId(driver, hongtho-master, 44805, None)
25/05/25 13:22:05 INFO BlockManager: Initialized BlockManager: BlockManagerId(driver, hongtho-master, 44805, None)
25/05/25 13:22:06 INFO SharedState: Setting hive.metastore.warehouse.dir ('null') to the value of spark.sql.warehouse.dir.
25/05/25 13:22:06 INFO SharedState: Warehouse path is 'file:/home/hadoopphongtho/spark/spark-warehouse'.
25/05/25 13:22:35 INFO DelegatingLogStore: LogStore LogStoreAdapter(io.delta.storage.HDFSLogStore) is used for scheme hdfs

```

Dữ liệu tạo ra sẽ là các giá trị trung bình theo ngày, giờ, tháng.

Kiểm tra lại:

```
$ hdfs dfs -ls /lakehouse/gold/streaming/
```

```

hadoopphongtho@hongtho-master:~$ hdfs dfs -ls /lakehouse/gold/streaming/
Found 3 items
drwxr-xr-x  - hadoopphongtho supergroup          0 2025-05-25 13:48 /lakehouse/gold/streaming/air_quality_daily_avg
drwxr-xr-x  - hadoopphongtho supergroup          0 2025-05-25 13:48 /lakehouse/gold/streaming/air_quality_hourly_avg
drwxr-xr-x  - hadoopphongtho supergroup          0 2025-05-25 13:48 /lakehouse/gold/streaming/air_quality_monthly_avg
hadoopphongtho@hongtho-master:~|

```

Tạo thêm một tập data gộp lại từ data batch và data streaming để thuận tiện cho trực quan hóa dữ liệu:

```

hadoopphongtho@hongtho-master:~/spark$ vim MergeStreamingAndBatch.py |
```

```

from pyspark.sql import SparkSession

spark = SparkSession.builder \
    .appName("Union Air Quality All") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog", "org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .getOrCreate()

df_batch = spark.read.format("delta").load("hdfs:///lakehouse/silver/air_quality_cleaned/")
df_streaming = spark.read.format("delta").load("hdfs:///lakehouse/silver/streaming/air_quality/")
cols = ["date", "CO", "NO2", "O3", "TSP", "PM25", "SO2", "AQI_Level", "hour", "day", "month"]

df_batch_sel = df_batch.select(*[c for c in cols if c in df_batch.columns])
df_streaming_sel = df_streaming.select(*[c for c in cols if c in df_streaming.columns])
df_all = df_batch_sel.unionByName(df_streaming_sel)

df_all.write.format("delta") \
    .mode("overwrite") \
    .save("hdfs:///lakehouse/gold/air_quality_all/")
~ ~

```

```

from pyspark.sql import SparkSession
spark = SparkSession.builder \

```

```

    .appName("Union Air Quality All") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog",
"org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .getOrCreate()
df_batch =
spark.read.format("delta").load("hdfs:///lakehouse/silver/air_quality_cleaned/")
df_streaming =
spark.read.format("delta").load("hdfs:///lakehouse/silver/streaming/air_quality/")
cols = ["date", "CO", "NO2", "O3", "TSP", "PM25", "SO2", "AQI_Level", "hour", "day",
"month"]

df_batch_sel = df_batch.select(*[c for c in cols if c in df_batch.columns])
df_streaming_sel = df_streaming.select(*[c for c in cols if c in
df_streaming.columns])
df_all = df_batch_sel.unionByName(df_streaming_sel)

df_all.write.format("delta") \
    .mode("overwrite") \
    .save("hdfs:///lakehouse/gold/air_quality_all/")

```

Chạy:

```

$ spark-submit --packages io.delta:delta-spark_2.12:3.3.1 --conf
spark.sql.extensions=io.delta.sql.DeltaSparkSessionExtension" --conf
"spark.sql.catalog.spark_catalog=org.apache.spark.sql.delta.catalog.DeltaCatalog
MergeStreamingAndBatch.py

```

```

hadoopphongtho@hongtho-master:~/spark$ spark-submit --packages io.delta:delta-spark_2.12:3.3.1 --conf spark.sql.extensions=io.delta.sql.DeltaSparkSessionExte
nsion" --conf "spark.sql.catalog.spark_catalog=org.apache.spark.sql.delta.catalog.DeltaCatalog MergeStreamingAndBatch.py
:: loading settings :: url = jar:file:/home/hadoopphongtho/spark/jars/ivy-2.5.1.jar!/org/apache/ivy/core/settings/ivysettings.xml
Ivy Default Cache set to: /home/hadoopphongtho/.ivy2/cache
The jars for the packages stored in: /home/hadoopphongtho/.ivy2/jars
io.delta#delta-spark_2.12 added as a dependency
:: resolving dependencies :: org.apache.spark#spark-submit-parent-0fb8085-8467-4e24-abc7-bb53ee0e05f0;1.0
  confs: [default]
    found io.delta#delta-spark_2.12;3.3.1 in central
    found io.delta#delta-storage;3.3.1 in central
    found org.antlr#antlr4-runtime;4.9.3 in local-m2-cache
:: resolution report :: resolve 125ms :: artifacts dl 5ms
  :: modules in use:
    io.delta#delta-spark_2.12;3.3.1 from central in [default]
    io.delta#delta-storage;3.3.1 from central in [default]
    org.antlr#antlr4-runtime;4.9.3 from local-m2-cache in [default]
    | conf | modules || artifacts | | | | |
    | number| search|dwnlded|evicted|| number|dwnlded|
    | default | 3 | 0 | 0 | 0 || 3 | 0 |
:: retrieving :: org.apache.spark#spark-submit-parent-0fb8085-8467-4e24-abc7-bb53ee0e05f0
  confs: [default]
  0 artifacts copied, 3 already retrieved (0kB/4ms)
25/05/25 13:58:34 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
25/05/25 13:58:35 INFO SparkContext: Running Spark version 3.5.3
25/05/25 13:58:35 INFO SparkContext: OS info Linux, 6.8.0-57-generic, amd64
25/05/25 13:58:35 INFO SparkContext: Java version 1.8.0_452
25/05/25 13:58:35 INFO ResourceUtils: =====
25/05/25 13:58:35 INFO ResourceUtils: No custom resources configured for spark.driver.
25/05/25 13:58:35 INFO ResourceUtils: =====
25/05/25 13:58:35 INFO SparkContext: Submitted application: Union Air Quality All
25/05/25 13:58:35 INFO ResourceProfile: Default ResourceProfile created, executor resources: Map(cores -> name: cores, amount: 1, script: , vendor: , memory
-> name: memory, amount: 1024, script: , vendor: , offheap -> name: offHeap, amount: 0, script: , vendor: ), task resources: Map(cpus -> name: cpus, amount
: 1,0)
25/05/25 13:58:35 INFO ResourceProfile: Limiting resource is cpu
25/05/25 13:58:35 INFO ResourceProfileManager: Added ResourceProfile id: 0
25/05/25 13:58:35 INFO SecurityManager: Changing view acls to: hadoopphongtho

```

Kiểm tra lại trực tiếp trong pyspark:

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col
spark = SparkSession.builder.getOrCreate()
df = spark.read.format("delta").load("hdfs://lakehouse/gold/air_quality_all/")
df.orderBy(col("date").desc()).show(20, truncate=False)
```

```

>>> from pyspark.sql import SparkSession
>>> from pyspark.sql.functions import col
>>> spark = SparkSession.builder.getOrCreate()
>>> df = spark.read.format("delta").load("hdfs://lakehouse/gold/air_quality_all/")
>>> df.orderBy(col("date").desc()).show(20, truncate=False)
25/05/25 14:08:34 WARN SparkStringUtils: Truncated the string representation of a plan since it was too large. This behavior can be adjusted by setting 'spark.sql.debug.maxToStringFields'.
+-----+-----+-----+-----+-----+-----+-----+-----+
|date |CO |NO2|O3 |TSP |PM25|SO2 |AQI_Level|hour|day|month|
+-----+-----+-----+-----+-----+-----+-----+-----+
|2025-05-25 13:21:20.168776|0.0|15.0|22.0|18.0|14.0|49.0|Average|13 |25 |5 |
|2025-05-25 13:20:19.777319|0.0|15.0|22.0|18.0|14.0|49.0|Average|13 |25 |5 |
|2025-05-25 13:15:49.013093|0.0|15.0|22.0|18.0|14.0|49.0|Average|13 |25 |5 |
|2025-05-25 13:14:33.438421|0.0|15.0|22.0|18.0|14.0|49.0|Average|13 |25 |5 |
|2025-05-25 13:11:37.727498|0.0|15.0|22.0|18.0|14.0|49.0|Average|13 |25 |5 |
|2025-05-25 12:24:12.842251|0.0|14.0|22.0|18.0|14.0|49.0|Average|12 |25 |5 |
|2025-05-25 12:23:12.459001|0.0|14.0|22.0|18.0|14.0|49.0|Average|12 |25 |5 |
|2025-05-25 12:22:12.081683|0.0|14.0|22.0|18.0|14.0|49.0|Average|12 |25 |5 |
|2025-05-25 12:20:11.709855|0.0|14.0|22.0|18.0|14.0|49.0|Average|12 |25 |5 |
|2025-05-25 12:19:10.96421|0.0|14.0|22.0|18.0|14.0|49.0|Average|12 |25 |5 |
|2025-05-25 12:18:10.577139|0.0|14.0|22.0|18.0|14.0|49.0|Average|12 |25 |5 |
|2025-05-25 12:17:10.142353|0.0|14.0|22.0|18.0|12.0|49.0|Average|12 |25 |5 |
|2025-05-25 12:16:09.775549|0.0|14.0|22.0|18.0|12.0|49.0|Average|12 |25 |5 |
|2025-05-25 12:15:09.35994|0.0|14.0|22.0|18.0|12.0|49.0|Average|12 |25 |5 |
|2025-05-25 12:14:07.962286|0.0|14.0|22.0|18.0|12.0|49.0|Average|12 |25 |5 |
|2025-05-25 12:13:07.526838|0.0|14.0|22.0|18.0|12.0|49.0|Average|12 |25 |5 |
|2025-05-25 12:12:07.158723|0.0|14.0|22.0|18.0|12.0|49.0|Average|12 |25 |5 |
|2025-05-25 12:11:06.784205|0.0|14.0|22.0|18.0|12.0|49.0|Average|12 |25 |5 |
+-----+-----+-----+-----+-----+-----+-----+-----+
only showing top 20 rows

>>>

```

Dữ liệu được gộp thành công ở lớp Gold.

## V – MỞ RỘNG TRIỀN KHAI LAKEHOUSE TRÊN CLOUD (AZURE + DATABRICKS)

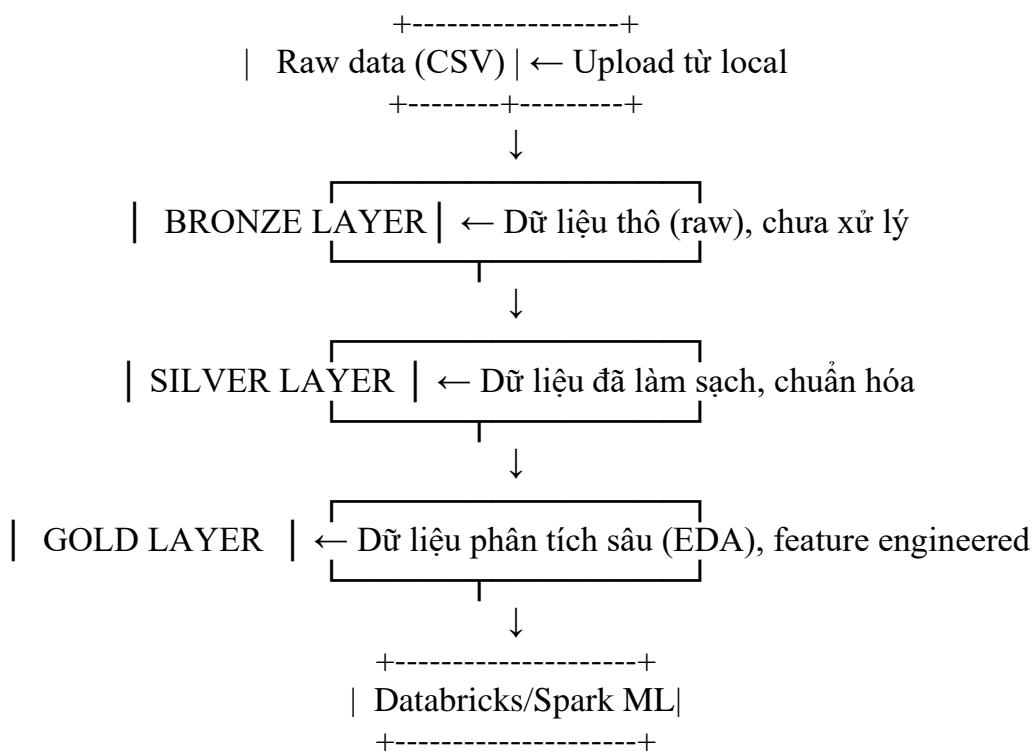
**Thiết kế một pipeline phân tầng hiện đại trên cloud, gồm:**

## **1. Mục tiêu tổng thể**

- Upload dữ liệu lên **cloud storage (Azure/AWS)**
- Thiết kế **kiến trúc 3 lớp (silver, gold, platinum)** theo chuẩn **Lakehouse**
- Dùng **Databricks (Spark)** để xử lý, phân tích, và train model
- Trực quan hóa kết quả bằng PowerBI

## **2. Quy trình triển khai theo kiến trúc 3 lớp (Bronze – Silver – Gold) trên Cloud**

Tổng quan kiến trúc:

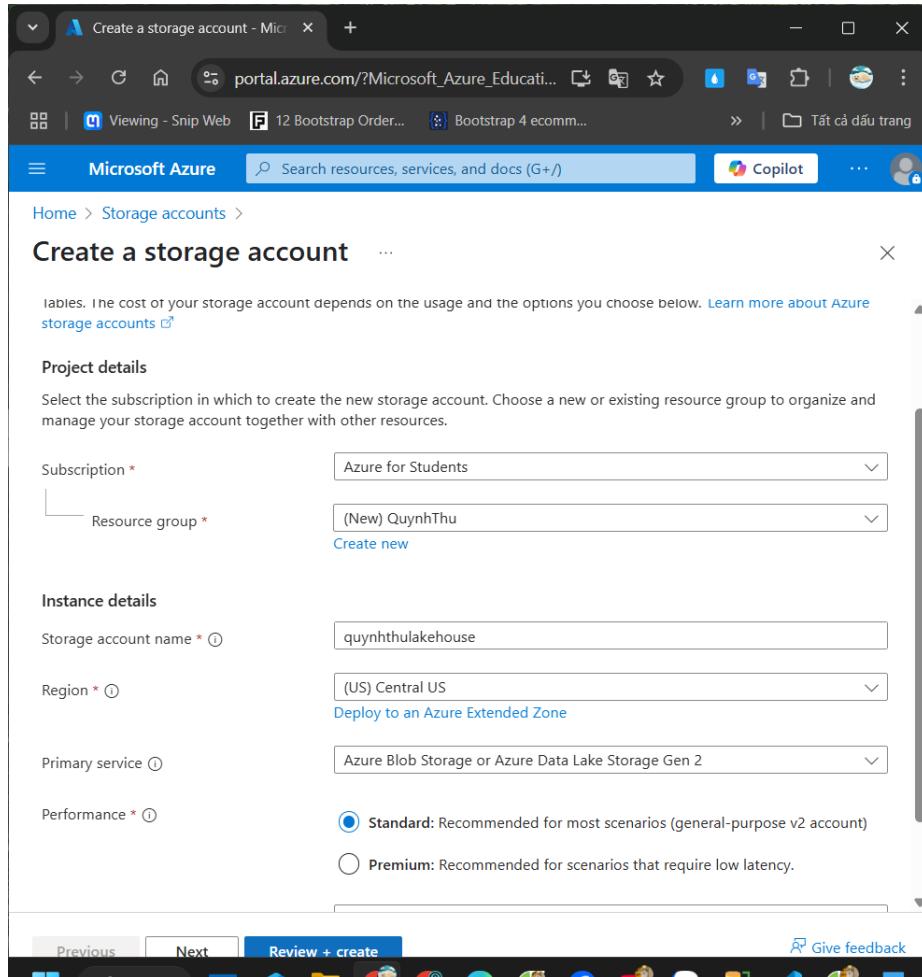


### **Bước 1: Upload dữ liệu lên cloud**

#### **1.1 Tạo Azure Storage Account (Data Lake Gen2)**

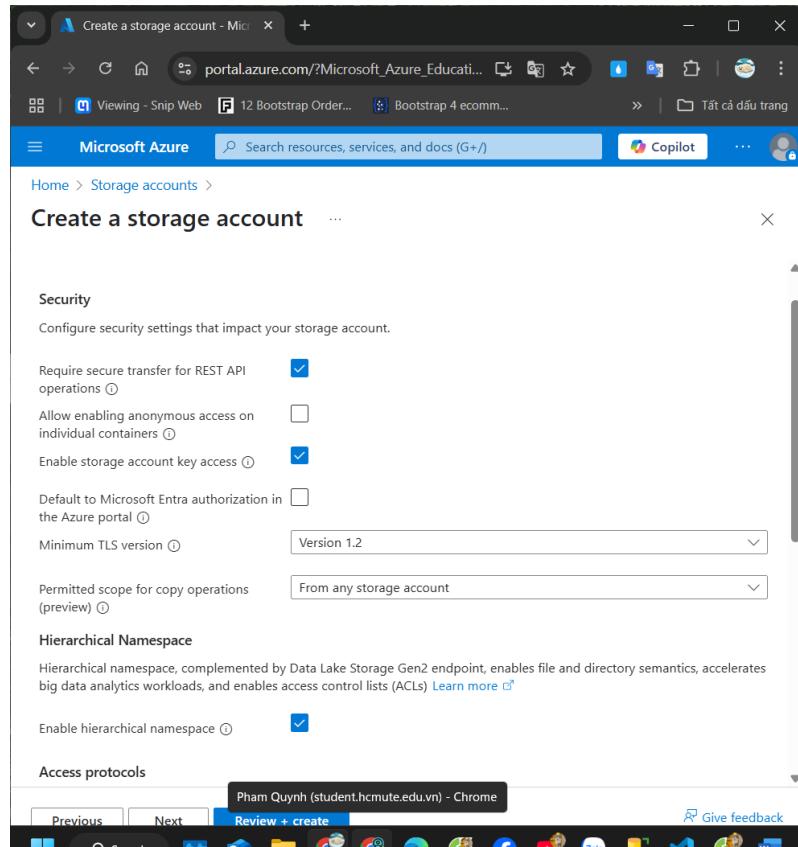
1. Truy cập: <https://portal.azure.com>
2. Tìm: "Storage accounts" → chọn "Create"
3. Cấu hình:
  - Subscription: chọn mặc định
  - Resource group: tạo mới, QuynhThu

- Storage account name: quynhthulakehouse
- Region: US



4. Bấm Next → Advanced

5. Bật: Enable Hierarchical namespace (đây là để dùng ADLS Gen2)



6. Review + Create → chờ vài phút để tạo xong

The screenshot shows the Microsoft Azure portal interface for creating a storage account. The page title is "Create a storage account". Below it, there are tabs for "Basics", "Advanced", "Networking", "Data protection", "Encryption", "Tags", and "Review + create", with "Review + create" being the active tab. The "Review + create" section displays the following configuration details:

| Setting              | Value                                               |
|----------------------|-----------------------------------------------------|
| Subscription         | Azure for Students                                  |
| Resource group       | QuynhThu                                            |
| Location             | Central US                                          |
| Storage account name | quynhthulakehouse                                   |
| Primary service      | Azure Blob Storage or Azure Data Lake Storage Gen 2 |
| Performance          | Standard                                            |
| Replication          | Read-access geo-redundant storage (RA-GRS)          |

Below this, the "Advanced" section shows the following settings:

| Setting                        | Value    |
|--------------------------------|----------|
| Enable hierarchical namespace  | Enabled  |
| Enable SFTP                    | Disabled |
| Enable network file system v3  | Disabled |
| Allow cross-tenant replication | Disabled |
| Access tier                    | Hot      |

At the bottom of the review screen, there are "Previous", "Next", and "Create" buttons, with "Create" being highlighted. A status bar at the bottom indicates "Pham Quynh (student.hcmute.edu.vn) - Chrome".

Chọn Create

Sau khi tạo xong, chờ workspace deploy.

The screenshot shows the Microsoft Azure portal's "Overview" page for the storage account "quynhthulakehouse\_1748178070704". The left sidebar has "Deployment" selected. The main content area displays the following information:

**Deployment is in progress**

| Deployment name | quynhthulakehouse_1748178070704                                                                                  |
|-----------------|------------------------------------------------------------------------------------------------------------------|
| Description     | Deployment name: quynhthulakehouse_1748178070704<br>Subscription: Azure for Students<br>Resource group: QuynhThu |
| Start time      | 5/25/2025, 8:01:59 PM                                                                                            |
| Correlation ID  | 4d7131bc-f21b-47d1-8dca-90f63a100840                                                                             |

**Deployment details**

| Resource    | Type | Status | Operation details |
|-------------|------|--------|-------------------|
| No results. |      |        |                   |

Below the table, there are links for "Give feedback" and "Tell us about your experience with deployment". To the right, there are promotional banners for "Microsoft Defender for Cloud", "Free Microsoft tutorials", and "Work with an expert".

Hoàn tất:

The screenshot shows the Microsoft Azure portal with a deployment overview for 'quynhthulakehouse\_1748178070704'. The status is 'Deployment succeeded' with a green checkmark icon. Deployment details include the name 'quynhthulakehouse\_1748178070704', subscription 'Azure for Students', and resource group 'QuynhThu'. The start time was 5/25/2025, 8:01:59 PM. A Correlation ID is also provided. On the right side, there are promotional cards for 'Cost Management', 'Microsoft Defender for Cloud', and 'Free Microsoft tutorials'. A sidebar on the left lists 'Overview', 'Inputs', 'Outputs', and 'Template'.

## 1.2 Tạo container (giống như thư mục gốc)

1. Sau khi tạo xong → Vào Storage account vừa tạo (quynhthulakehouse)
2. Menu bên trái → Chọn Containers

The screenshot shows the 'Containers' section of the Azure Storage account 'quynhthulakehouse'. It displays a table with one item: 'Logs'. The columns are 'Name' (checkbox), 'Last modified' (20/02/29 25/5/2025), 'Anonymous access level' (Private), and 'Lease state' (Available). The table has a header row with 'Edit columns' and 'only show active containers' options. The left sidebar includes sections for 'Access Control (IAM)', 'Data storage', 'Containers' (selected), 'Security + networking', 'Settings', 'Configuration', 'Endpoints', 'Help', 'Connectivity check', and 'Support + Troubleshooting'.

### 3. Bấm + Container

- o Name: lakehouse (hoặc datalake)
- o Public access level: Private (no anonymous access)

New container

Name \*  
lakehouse

Anonymous access level  
Private (no anonymous access)

The access level is set to private because anonymous access is disabled on this storage account.

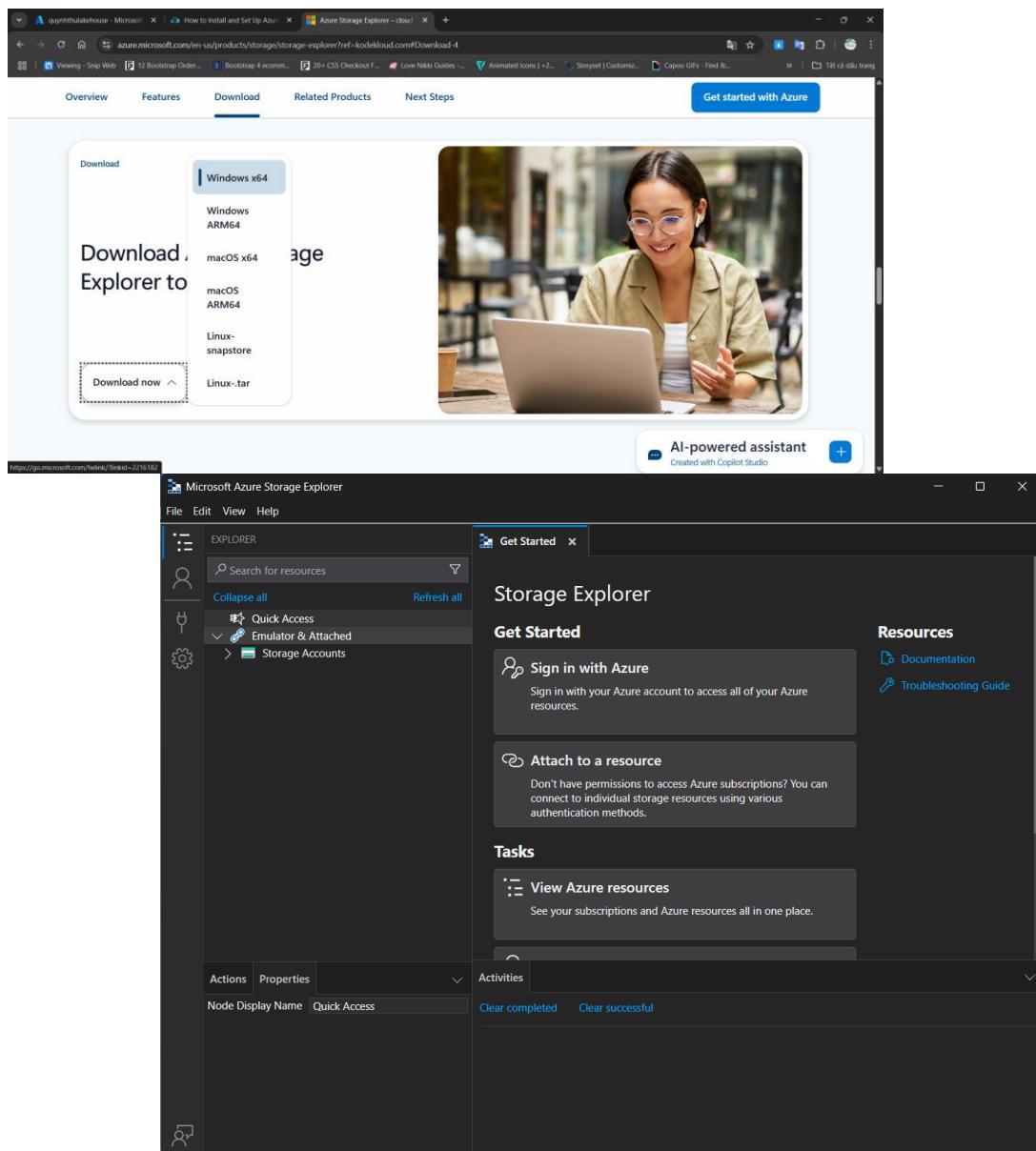
Create

o Bấm Create

| Name      | Last modified      | Anonymous access level | Lease state |
|-----------|--------------------|------------------------|-------------|
| Slogs     | 20:02:39 25/5/2025 | Private                | Available   |
| lakehouse | 20:07:10 25/5/2025 | Private                | Available   |

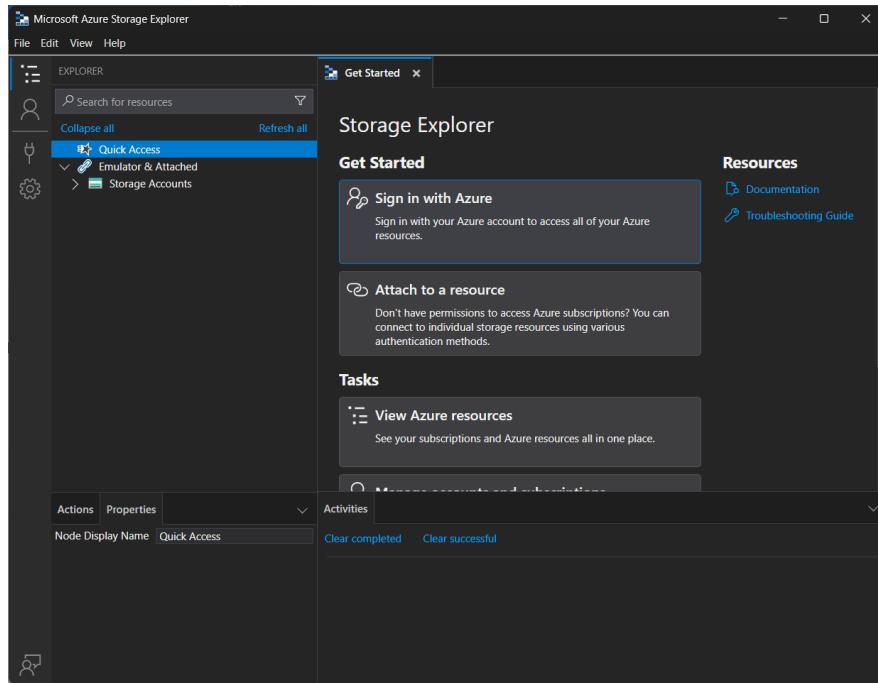
### 1.3 Cài Azure Storage Explorer để upload file

Link: <https://azure.microsoft.com/en-us/products/storage/storage-explorer?ref=kodekloud.com#Download-4>

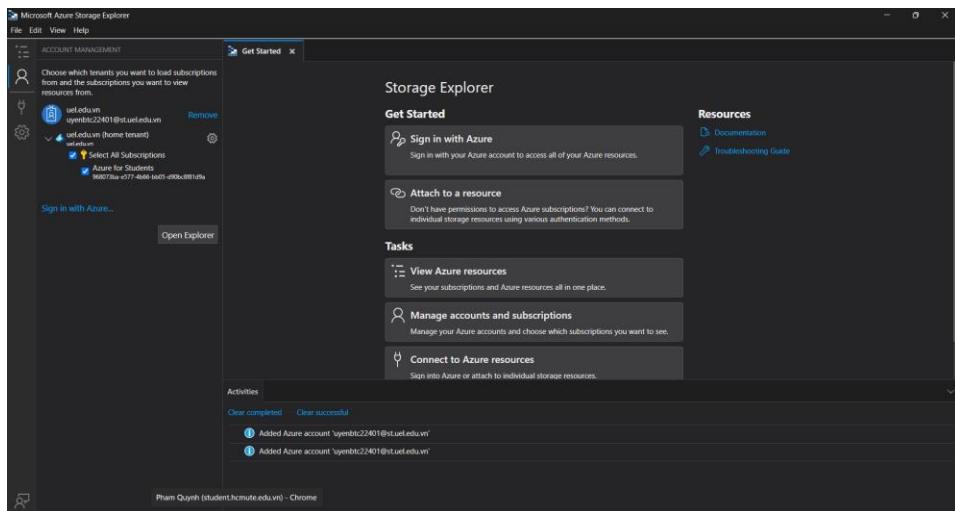


## 1.4 Kết nối Azure Storage Explorer với tài khoản Azure

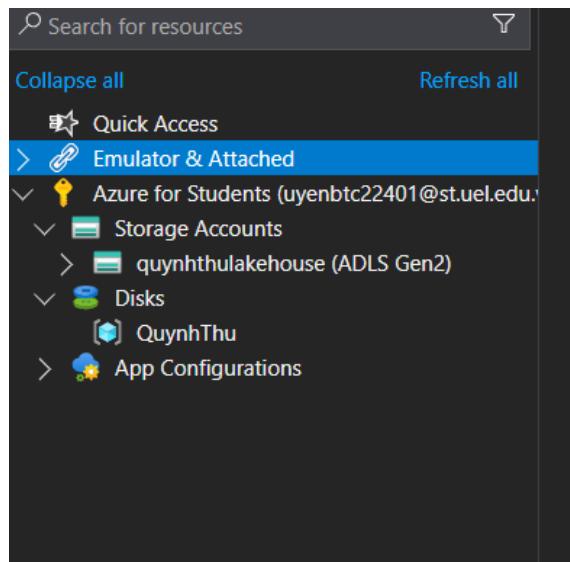
### 1. Mở Azure Storage Explorer



2. Bấm vào nút plug (Connect to Azure account)
3. Chọn "Add an Azure Account"
4. Đăng nhập bằng tài khoản Azure



5. Storage account sẽ xuất hiện trong phần Storage Accounts → [Tên tài khoản]

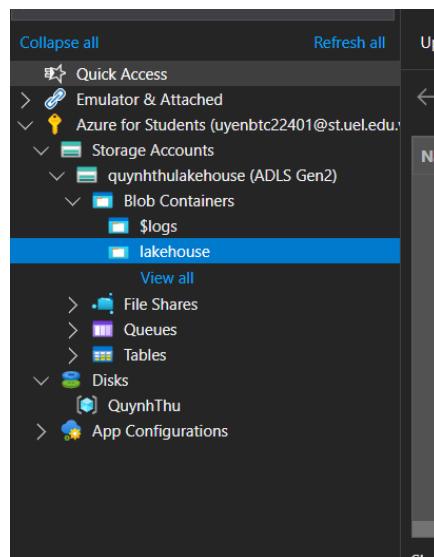


## 1.5 Upload dữ liệu vào thư mục bronze

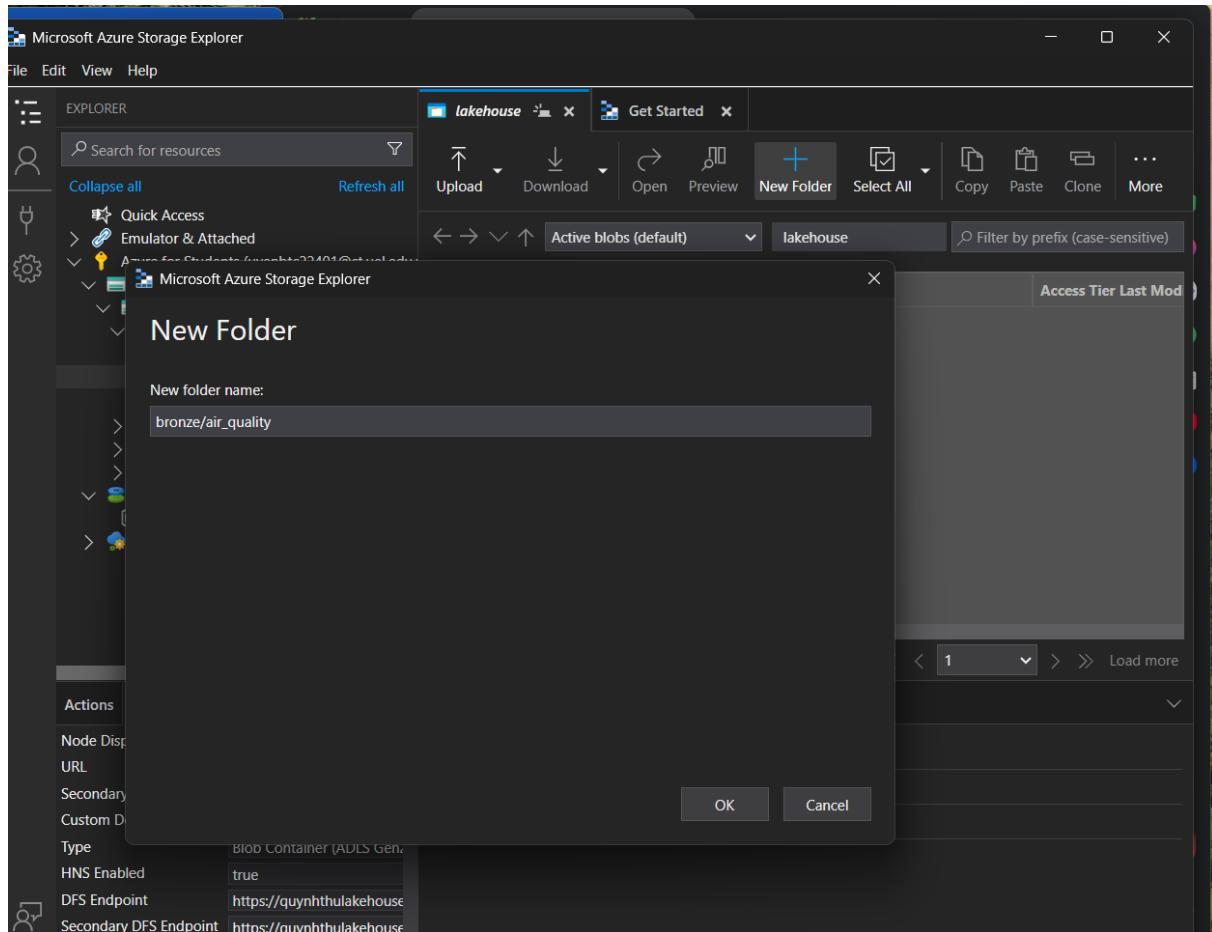
- Trong Storage Explorer:

- Điều hướng đến:

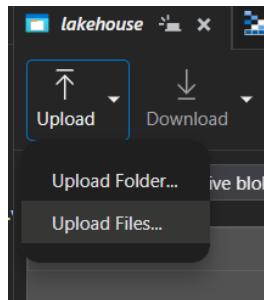
Storage Accounts → quynhthulakehouse → Blob Containers  
→ lakehouse



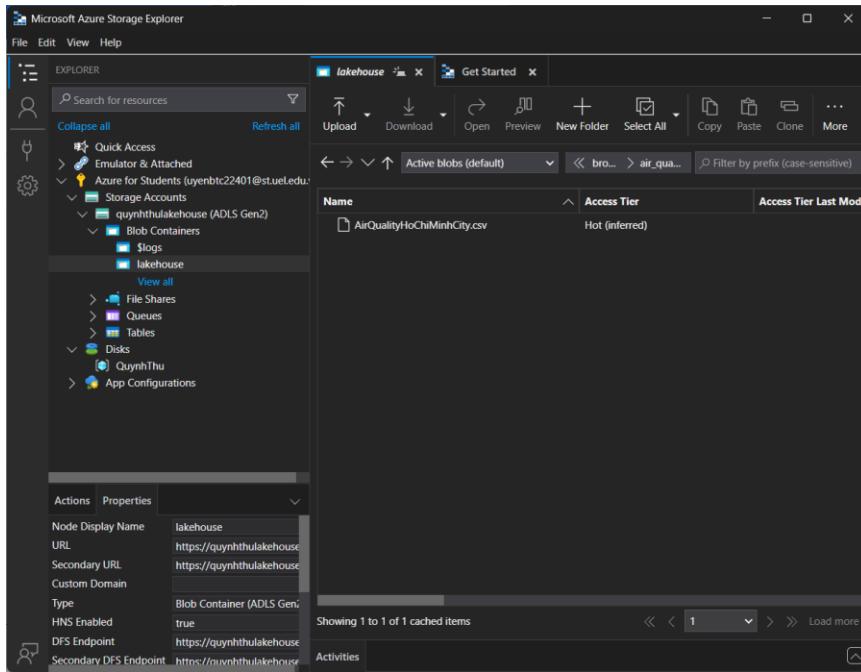
- New Folder → nhập bronze/air\_quality



- Vào folder đó → bấm Upload → Upload Files



- Chọn file predictions.csv từ máy→ Bấm Upload



Sau khi xong, dữ liệu sẽ nằm tại:

`abfss://lakehouse@quynhthulakehouse.dfs.core.windows.net/bronze/air_quality/`

`AirQualityHoChiMinhCity.csv`

## Bước 2: Kết nối cloud storage vào Databricks

### 2.1. Tạo App Registration mới

Tìm: "App registrations" → click vào

Tạo App mới

1. Click “+ New registration”
2. Điền:
  - o Name: databricks-app
  - o Supported account types: Chọn Accounts in this organizational directory only
  - o Leave Redirect URI trống

**Name**  
The user-facing display name for this application (this can be changed later).

**Supported account types**

Who can use this application or access this API?

- Accounts in this organizational directory only (ue.edu.vn only - Single tenant)
- Accounts in any organizational directory (Any Microsoft Entra ID tenant - Multitenant)
- Accounts in any organizational directory (Any Microsoft Entra ID tenant - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)
- Personal Microsoft accounts only

[Help me choose...](#)

**Redirect URI (optional)**  
We'll return the authentication response to this URI after successfully authenticating the user. Providing this now is optional and it can be  
[By proceeding, you agree to the Microsoft Platform Policies](#)

[Register](#)

### 3. Click Register

Lấy thông tin xác thực

Sau khi tạo xong, trang Overview của app xuất hiện.

**Overview**

Quickstart

Integration assistant

Diagnose and solve problems

Manage

Branding & properties

Authentication

Certificates & secrets

Token configuration

API permissions

Expose an API

App roles

Owners

Roles and administrators

Manifest

Support + Troubleshooting

Add or remove favorites by pressing **Ctrl+Shift+F**

**Essentials**

|                         |                                        |                             |                               |
|-------------------------|----------------------------------------|-----------------------------|-------------------------------|
| Display name            | : databricks-app                       | Client credentials          | : Add a certificate or secret |
| Application (client) ID | : 13947051-2f53-4b1e-b328-13ce35710f1a | Redirect URIs               | : Add a Redirect URI          |
| Object ID               | : eedcd936-55d3-4651-b17b-eb676be6e36  | Application ID URI          | : Add an Application ID URI   |
| Directory (tenant) ID   | : 07acb355-56bc-489b-b98c-8fea440460e8 | Managed application in L... | : databricks-app              |
| Supported account types | : My organization only                 |                             |                               |

**Welcome to the new and improved App registrations. Looking to learn how it's changed from App registrations (Legacy)? [Learn more](#)**

**Starting June 30th, 2020 we will no longer add any new features to Azure Active Directory Authentication Library (ADAL) and Azure Active Directory Graph. We will continue to provide technical support and security updates but we will no longer provide feature updates. Applications will need to be upgraded to Microsoft Authentication Library (MSAL) and Microsoft Graph. [Learn more](#)**

**Get Started** Documentation

**Build your application with the Microsoft identity platform**

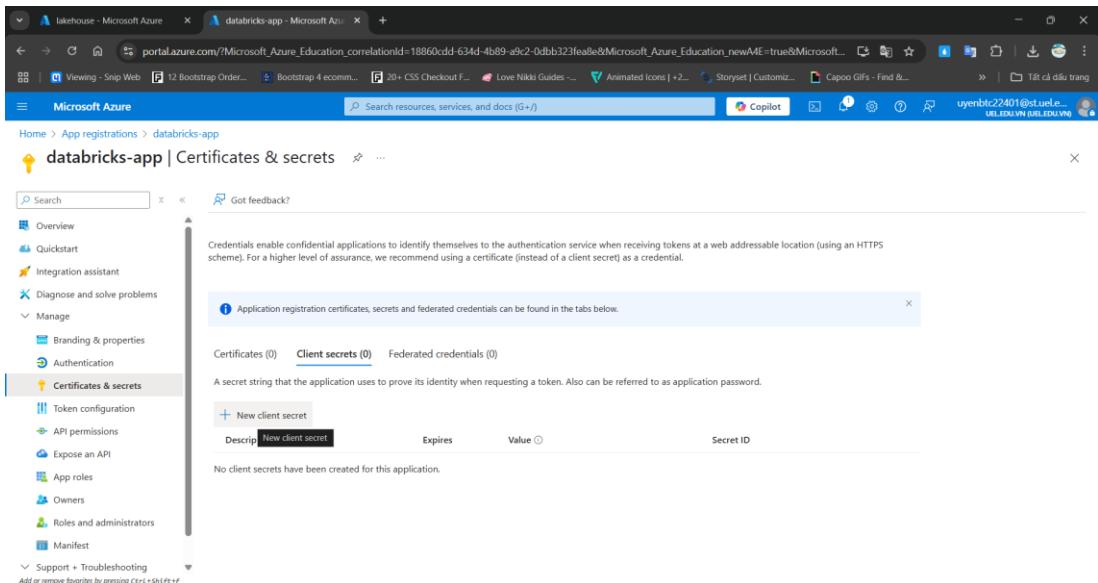
The Microsoft identity platform is an authentication service, open-source libraries, and application management tools. You can create modern,

Ghi lại các thông tin sau:

| Thông tin                      | Cách dùng                                                                 |
|--------------------------------|---------------------------------------------------------------------------|
| <b>Application (client) ID</b> | Dùng để cấu hình trong Databricks<br>13947051-2f53-4b1e-b328-13ce35710f1a |
| <b>Directory (tenant) ID</b>   | Dùng để xác thực qua OAuth<br>07acb355-56bc-489b-b98c-8fea440460e8        |

## Tạo Client Secret

## 1. Vào tab Certificates & secrets



## 2. Click + New client secret

- Description: databricks-secret
  - Expiry: 6 hoặc 12 tháng

3. Click Add

4. Copy ngay giá trị secret sau khi tạo (không xem lại được)

**Value: NpM8Q~aMRru6tQ1a1NQWQLNENeaqYjtS2c54PdlO**

**Secret ID: 312f340a-ea48-4f16-8a5d-56cd46ffc183**

## Cấp quyền truy cập Azure Data Lake

### Vào Storage Account

1. Vào Azure Portal → Storage accounts

The screenshot shows the Microsoft Azure Storage accounts page. At the top, there are navigation links for 'Home', 'Storage accounts', and 'Copilot'. Below the header, there's a message: 'You are viewing a new version of Browse experience. Some features may be missing. Click here to access the old experience.' A search bar and filter options ('Subscription equals all', 'Resource Group equals all', 'Location equals all') are present. The main table lists one storage account:

| Name              | Type            | Kind      | Resource Group | Location   |
|-------------------|-----------------|-----------|----------------|------------|
| quynhthulakehouse | Storage account | StorageV2 | QuynhThu       | Central US |

At the bottom, it says 'Showing 1 - of 1. Display count: 10' and a 'Give feedback' link.

## 2. Chọn đúng Data Lake Storage (loại StorageV2 - Hierarchical namespace)

Gán quyền

1. Vào menu Access control (IAM) bên trái
2. Click + Add → Add role assignment

The screenshot shows the Microsoft Azure Access Control (IAM) page for the storage account 'quynhthulakehouse'. The left sidebar has a tree view with 'Access Control (IAM)' selected. The main area has tabs: 'Add role assignment' (selected), 'Check access', 'View my access', 'Roles', 'Deny assignments', and 'Classic administrators'. Under 'My access', there's a 'Check access' button. The right side has three sections: 'Grant access to this resource' (with 'Add role assignment' button), 'View access to this resource' (with 'View' button), and 'View deny assignments' (with 'View' button).

### 3. Chọn role:

- o Storage Blob Data Reader
- o Storage Blob Data Contributor (nếu cần ghi)

### 4. Assign access to: User, group, or service principal

### 5. Select member: tìm App vừa tạo (databricks-app)

Selected role: Storage Blob Data Contributor

Assign access to: User, group, or service principal

| Name           | Object ID                             | Type |
|----------------|---------------------------------------|------|
| databricks-app | 1e3127c7-9d20-4cca-8d1d-e53f73d7dc... | App  |

Description: Optional

Review + assign    Previous    Next

### 6. Click Review + assign

Role: Storage Blob Data Contributor

Scope: /subscriptions/968073ba-e577-4b66-bb05-d90bc8ff1d9a/resourceGroups/QuynhThu/providers/Microsoft.Storage/storageAccounts/quynhthulakehouse

| Members | Name           | Object ID                            | Type |
|---------|----------------|--------------------------------------|------|
|         | databricks-app | 1e3127c7-9d20-4cca-8d1d-e53f73d7dcdd | App  |

Description: No description

Condition: None

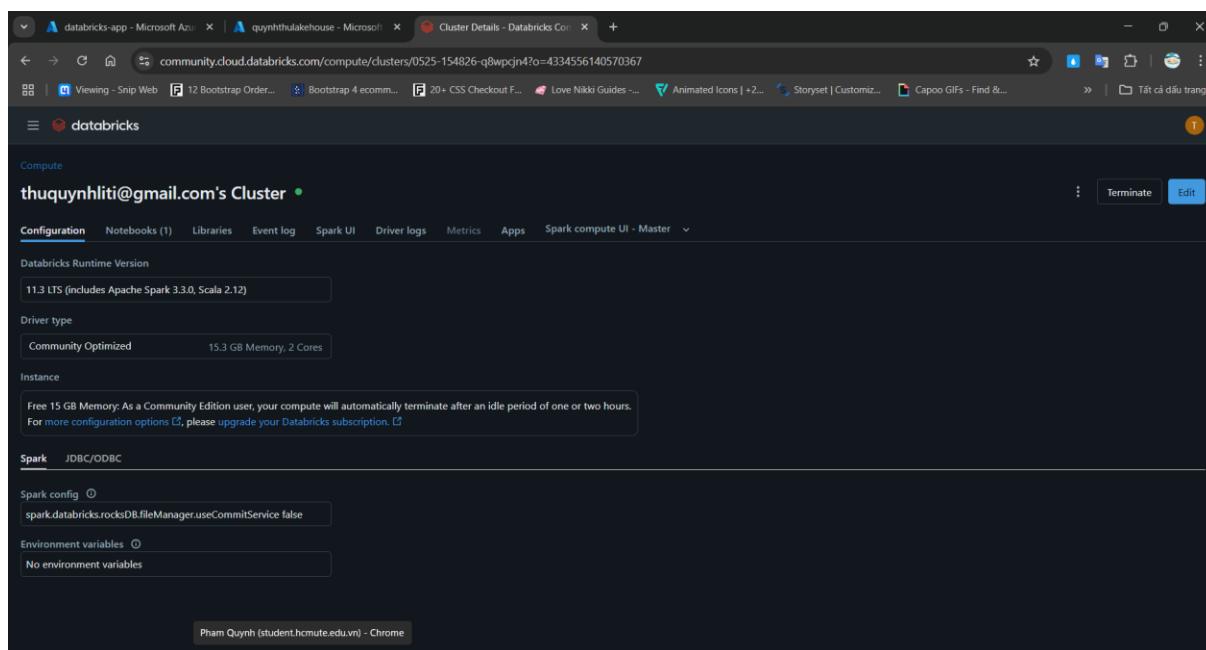
Review + assign    Previous    Next

## 2.2 – TẠO TÀI KHOẢN DATABRICKS (Community Edition – MIỄN PHÍ)

- Click "Sign up"
- Điền thông tin email, mật khẩu, họ tên
- Kiểm tra email để xác nhận
- Đăng nhập vào Databricks Community Edition
- Sau khi vào, ta sẽ có:
  - Workspace miễn phí
  - Cluster mặc định (tự tạo)

Tạo một Spark cluster mới:

- Bấm Create Compute.
- Đặt tên tùy ý.
- Runtime: chọn latest (ví dụ: 11.3 LTS (Scala 2.12, Spark 3.3.0) hoặc cao hơn).
- Nhấn Create Cluster.



## Mount Azure Data Lake vào Databricks

Cấu hình trong Databricks notebook:

```
configs = {
```

```

"fs.azure.account.auth.type": "OAuth",
"fs.azure.account.oauth.provider.type":
"org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider",
"fs.azure.account.oauth2.client.id": "<your-client-id>",
"fs.azure.account.oauth2.client.secret": "<your-client-secret>",
"fs.azure.account.oauth2.client.endpoint": "https://login.microsoftonline.com/<your-
tenant-id>/oauth2/token"
}

dbutils.fs.mount(
  source = "abfss://<filesystem-name>@<storage-account-name>.dfs.core.windows.net/",
  mount_point = "/mnt/lakehouse",
  extra_configs = configs
)

```

### Thay thế:

- <your-client-id> = Application ID *13947051-2f53-4b1e-b328-13ce35710f1a*
- <your-client-secret> = Secret vừa tạo  
*NpM8Q~aMRru6tQ1a1NQWQLNEaqYjtS2c54PdlO*
- <your-tenant-id> = Directory ID *07acb355-56bc-489b-b98c-8fea440460e8*
- <filesystem-name> = tên container đã upload dữ liệu *lakehouse*
- <storage-account-name> = tên storage account Azure *quynhthulakehouse*

```

configs = {
  "fs.azure.account.auth.type": "OAuth",
  "fs.azure.account.oauth.provider.type":
"org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider",
  "fs.azure.account.oauth2.client.id": "13947051-2f53-4b1e-b328-13ce35710f1a",
  "fs.azure.account.oauth2.client.secret": 
"NpM8Q~aMRru6tQ1a1NQWQLNEaqYjtS2c54PdlO",
  "fs.azure.account.oauth2.client.endpoint": 
"https://login.microsoftonline.com/07acb355-56bc-489b-b98c-8fea440460e8/oauth2/token"
}

dbutils.fs.mount(
  source = "abfss://lakehouse@quynhthulakehouse.dfs.core.windows.net/",
  mount_point = "/mnt/lakehouse",
  extra_configs = configs
)

```

```

configs = (
    "fs.azure.account.auth.type": "OAuth",
    "fs.azure.account.oauth.provider-type": "org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider",
    "fs.azure.account.oauth2.client.id": "13947051-2f53-4b1e-b328-13ce35710f1a",
    "fs.azure.account.oauth2.client.secret": "NpHQ0Q-aRvu6TQ1n1QWQLNEaqayjSz2c54Pd1O",
    "fs.azure.account.oauth2.client.endpoint": "https://login.microsoftonline.com/07acb355-56bc-489b-b98c-8fea440460e8/oauth2/token"
)

dbutils.fs.mount(
    source = "abfss://lakehouse@quynhthulakehouse.dfs.core.windows.net/",
    mount_point = "/mnt/lakehouse",
    extra_configs = configs
)

```

Out[2]: True

### Bước 3: Xây dựng pipeline 3 lớp trên Databricks

Bronze → Silver (làm sạch, chuyển kiểu dữ liệu)

- Ép kiểu dữ liệu cho đúng (timestamp, float)
- Loại bỏ null/NA
- Lưu dưới dạng Parquet

```

spark.conf.set(
    "fs.azure.account.key.quynhthulakehouse.dfs.core.windows.net",
    "lZBCbTiGHXUZGBuXZUu0KZ7wPdSArFQVLBst0yRfhCZEhD3hhg2KXpaUXtqv4jzn/4kIH8HTNSi0+AST
76W0XA=="
)

# Đọc dữ liệu từ bronze
bronze_df = spark.read.option("header",
    True).csv("abfss://lakehouse@quynhthulakehouse.dfs.core.windows.net/bronze/air_quality/AirQualityHoChiMinhCity.csv")

# Ép kiểu và loại bỏ các dòng null
from pyspark.sql.functions import col, to_timestamp

silver_df = bronze_df.select(
    to_timestamp(col("date"), "dd-MM-yyyy HH:mm").alias("datetime"),
    col(`PM2.5`).cast("double").alias("pm25"),
    col("PM10").cast("double").alias("pm10"),
    col("PM100").cast("double").alias("pm100"),
    col("NO2").cast("double").alias("no2"),
    col("SO2").cast("double").alias("so2"),
    col("CO").cast("double").alias("co"),
    col("O3").cast("double").alias("o3"),
    col("Rain").cast("double").alias("rain"),
    col("Temp").cast("double").alias("temp"),
    col("Humidity").cast("double").alias("humidity"),
    col("Wind").cast("double").alias("wind"),
    col("Clouds").cast("double").alias("clouds"),
    col("Pressure").cast("double").alias("pressure")
)

```

```

    col("TSP").cast("double").alias("tsp"),
    col("O3").cast("double").alias("o3"),
    col("CO").cast("double").alias("co"),
    col("NO2").cast("double").alias("no2"),
    col("SO2").cast("double").alias("so2"),
    col("Station_No").alias("station")
).na.drop()

# Ghi ra Silver layer
silver_df.write.mode("overwrite").parquet("abfss://lakehouse@quynhthulakehouse.dfs.core.windows.net/silver/air_quality/")

```

```

Untitled Notebook 2025-05-25 19:16:05 Python
File Edit View Run Help Last edit was 3 minutes ago
Bronze Layer → Silver Layer (Làm sạch và chuẩn hóa)

spark.conf.set(
    "fs.azure.account.key.quynhthulakehouse.dfs.core.windows.net",
    "12Bacbf1fG001ZGbuXZLbdKZ7wPdSArFQVL8tbyRfCZEnD3hgg2XpulRtqv4jzn/4kIDH8HTNS10+AST70h0XA=="
)
# Đọc dữ liệu từ bronze
bronze_df = spark.read.option("header", True).csv("abfss://lakehouse@quynhthulakehouse.dfs.core.windows.net/bronze/air_quality/AirQualityHoChiMinhCity.csv")

# Ép kiểu và loại bỏ các dòng null
from pyspark.sql.functions import col, to_timestamp

silver_df = bronze_df.select(
    to_timestamp(col("date"), "dd-MM-yyyy HH:mm").alias("datetime"),
    col("PM2.5").cast("double").alias("pm25"),
    col("TSP").cast("double").alias("tsp"),
    col("O3").cast("double").alias("o3"),
    col("CO").cast("double").alias("co"),
    col("NO2").cast("double").alias("no2"),
    col("SO2").cast("double").alias("so2"),
    col("Station_No").alias("station")
).na.drop()

# Ghi ra Silver layer
silver_df.write.mode("overwrite").parquet("abfss://lakehouse@quynhthulakehouse.dfs.core.windows.net/silver/air_quality/")

```

Kiểm tra:

```

display(bronze_df.limit(10))
display(silver_df.limit(10))

```

Untitled Notebook 2025-05-25 19:16:05 Python

File Edit View Run Help Last edit was 3 minutes ago

(2) Spark Jobs

Table +

|    | $\Delta_c$ date           | $\Delta_c$ Station_No | $\Delta_c$ TSP | $\Delta_c$ PM2.5 | $\Delta_c$ O3 | $\Delta_c$ CO | $\Delta_c$ NO2 | $\Delta_c$ SO2 | $\Delta_c$ Temperature | $\Delta_c$ Humidity |
|----|---------------------------|-----------------------|----------------|------------------|---------------|---------------|----------------|----------------|------------------------|---------------------|
| 1  | 23-02-2021 21:00:00+00:00 | 1                     | 32.93571429    | 15.6047619       | 55.43138095   | 1330.451429   | 112.7407619    | 393            | 28.36190476            | 63.18809524         |
| 2  | 23-02-2021 22:00:00+00:00 | 1                     | 30.93235294    | 14.59411765      | 58.19717647   | 1200.603529   | 112.3664706    | 377.5882353    | 28.32058824            | 63.77352941         |
| 3  | 23-02-2021 23:00:00+00:00 | 1                     | 27.645         | 13.43666667      | 55.02943333   | 1177.897      | 112.7004333    | 372.4766667    | 28.33666667            | 64.205              |
| 4  | 24-02-2021 00:00:00+00:00 | 1                     | 24.38          | 12.365           | 54.7677       | 1267.476      | 112.4808667    | 389.07         | 28.305                 | 64.735              |
| 5  | 24-02-2021 01:00:00+00:00 | 1                     | 22.52166667    | 11.63666667      | 53.7862       | 1322.293      | 114.3315       | 393            | 28.3                   | 65.18833333         |
| 6  | 24-02-2021 02:00:00+00:00 | 1                     | 22.14333333    | 11.53            | 55.81463333   | 1358.392      | 115.9939333    | 394.31         | 28.25333333            | 65.56166667         |
| 7  | 24-02-2021 03:00:00+00:00 | 1                     | 21.57166667    | 11.24166667      | 54.80041667   | 1309.687      | 114.1746667    | 393            | 28.21666667            | 65.925              |
| 8  | 24-02-2021 04:00:00+00:00 | 1                     | 20.98833333    | 11.13            | 53.85163333   | 1270.914      | 113.9237333    | 393            | 28.2                   | 66.29166667         |
| 9  | 24-02-2021 05:00:00+00:00 | 1                     | 20.96          | 11.06            | 52.8047       | 1263.656      | 113.0141       | 393            | 28.18833333            | 66.58333333         |
| 10 | 24-02-2021 06:00:00+00:00 | 1                     | 21.34166667    | 11.17666667      | 54.30966667   | 1322.675      | 115.3666       | 399.55         | 28.16166667            | 66.90666667         |

↓ 10 rows | 1.39s runtime

Table +

|    | $\Delta_c$ datetime           | 1.2 pm25    | 1.2 tsp     | 1.2 o3      | 1.2 co      | 1.2 no2     | 1.2 so2     | $\Delta_c$ station |
|----|-------------------------------|-------------|-------------|-------------|-------------|-------------|-------------|--------------------|
| 1  | 2021-02-23T21:00:00.000+00:00 | 15.6047619  | 32.93571429 | 55.43138095 | 1330.451429 | 112.7407619 | 393         | 1                  |
| 2  | 2021-02-23T22:00:00.000+00:00 | 14.59411765 | 30.93235294 | 58.19717647 | 1200.603529 | 112.3664706 | 377.5882353 | 1                  |
| 3  | 2021-02-23T23:00:00.000+00:00 | 13.43666667 | 27.645      | 55.02943333 | 1177.897    | 112.7004333 | 372.4766667 | 1                  |
| 4  | 2021-02-24T00:00:00.000+00:00 | 12.365      | 24.38       | 54.7677     | 1267.476    | 112.4808667 | 389.07      | 1                  |
| 5  | 2021-02-24T01:00:00.000+00:00 | 11.63666667 | 22.52166667 | 53.7862     | 1322.293    | 114.3315    | 393         | 1                  |
| 6  | 2021-02-24T02:00:00.000+00:00 | 11.53       | 22.14333333 | 55.81463333 | 1358.392    | 115.9939333 | 394.31      | 1                  |
| 7  | 2021-02-24T03:00:00.000+00:00 | 11.24166667 | 21.57166667 | 54.80041667 | 1309.687    | 114.1746667 | 393         | 1                  |
| 8  | 2021-02-24T04:00:00.000+00:00 | 11.13       | 20.98833333 | 53.85163333 | 1270.914    | 113.9237333 | 393         | 1                  |
| 9  | 2021-02-24T05:00:00.000+00:00 | 11.06       | 20.96       | 52.8047     | 1263.656    | 113.0141    | 393         | 1                  |
| 10 | 2021-02-24T06:00:00.000+00:00 | 11.17666667 | 21.34166667 | 54.30966667 | 1322.675    | 115.3666    | 399.55      | 1                  |

↓ 10 rows | 1.39s runtime

## Silver → Gold (EDA, Feature Engineering)

- Trích xuất đặc trưng thời gian: giờ, ngày, thứ, tháng
- Tính trung bình/rolling (nếu muốn)
- Chuẩn bị dữ liệu để train

```
from pyspark.sql.functions import hour, dayofweek, dayofmonth, month

# Đọc từ Silver
silver_df =
spark.read.parquet("abfss://lakehouse@quynhthulakehouse.dfs.core.windows.net/silver/air_quality/")

# Thêm các đặc trưng thời gian
gold_df = silver_df.withColumn("hour", hour("datetime")) \
```

```

        .withColumn("dayofweek", dayofweek("datetime")) \
        .withColumn("day", dayofmonth("datetime")) \
        .withColumn("month", month("datetime"))

# Ghi ra Gold layer
gold_df.write.mode("overwrite").parquet("abfss://lakehouse@quynhthulakehouse.dfs.core
.windows.net/gold/air_quality/")

```

The screenshot shows a Databricks notebook interface. The title bar reads "Untitled Notebook 2025-05-25 19:16:05 Python". The notebook content is as follows:

```

Silver Layer → Gold Layer (Feature Engineering + EDA)

from pyspark.sql.functions import hour, dayofweek, dayofmonth, month

# Đầu từ Silver
silver_df = spark.read.parquet("abfss://lakehouse@quynhthulakehouse.dfs.core.windows.net/silver/air_quality/")

# Thêm các đặc trưng thời gian
gold_df = silver_df.withColumn("hour", hour("datetime")) \
    .withColumn("dayofweek", dayofweek("datetime")) \
    .withColumn("day", dayofmonth("datetime")) \
    .withColumn("month", month("datetime"))

# Ghi ra Gold layer
gold_df.write.mode("overwrite").parquet("abfss://lakehouse@quynhthulakehouse.dfs.core.windows.net/gold/air_quality/")

(2) Spark Jobs
> gold_df: pyspark.sql.dataframe.DataFrame = [datetime: timestamp, pm25: double ... 10 more fields]
> silver_df: pyspark.sql.dataframe.DataFrame = [datetime: timestamp, pm25: double ... 6 more fields]

```

## Kiểm tra kết quả:

```
display(gold_df.limit(10))
```

Kiểm tra kết quả

```

display(gold_df.limit(10))

```

|    | datetime                      | pm25        | tsp         | so3         | co          | no2         | so2         | station | hour | dayofweek | day | month |
|----|-------------------------------|-------------|-------------|-------------|-------------|-------------|-------------|---------|------|-----------|-----|-------|
| 1  | 2021-02-23T21:00:00.000+00:00 | 15.6047619  | 32.93571429 | 55.43138095 | 1330.451429 | 112.7407619 | 393         | 1       | 21   | 3         | 23  | 2     |
| 2  | 2021-02-23T22:00:00.000+00:00 | 14.59411765 | 30.93235294 | 58.19717647 | 1200.603529 | 112.3664706 | 377.5882353 | 1       | 22   | 3         | 23  | 2     |
| 3  | 2021-02-23T23:00:00.000+00:00 | 13.43666667 | 27.645      | 55.02943333 | 1177.897    | 112.7004333 | 372.4766667 | 1       | 23   | 3         | 23  | 2     |
| 4  | 2021-02-24T00:00:00.000+00:00 | 12.365      | 24.38       | 54.7677     | 1267.476    | 112.4808667 | 389.07      | 1       | 0    | 4         | 24  | 2     |
| 5  | 2021-02-24T01:00:00.000+00:00 | 11.53666667 | 22.52166667 | 53.7862     | 1322.293    | 114.3315    | 393         | 1       | 1    | 4         | 24  | 2     |
| 6  | 2021-02-24T02:00:00.000+00:00 | 11.53       | 22.14333333 | 55.81463333 | 1358.392    | 115.9939333 | 394.31      | 1       | 2    | 4         | 24  | 2     |
| 7  | 2021-02-24T03:00:00.000+00:00 | 11.24166667 | 21.57166667 | 54.80041667 | 1309.687    | 114.1746667 | 393         | 1       | 3    | 4         | 24  | 2     |
| 8  | 2021-02-24T04:00:00.000+00:00 | 11.13       | 20.98833333 | 53.85163333 | 1270.914    | 113.9237333 | 393         | 1       | 4    | 4         | 24  | 2     |
| 9  | 2021-02-24T05:00:00.000+00:00 | 11.06       | 20.96       | 52.8047     | 1263.656    | 113.0141    | 393         | 1       | 5    | 4         | 24  | 2     |
| 10 | 2021-02-24T06:00:00.000+00:00 | 11.17666667 | 21.34166667 | 54.30966667 | 1322.675    | 115.3666    | 399.55      | 1       | 6    | 4         | 24  | 2     |

↓ 10 rows | 0.59s runtime Refreshed 4 minutes ago

## Kết quả thư mục:

/lakehouse/

- └── bronze/air\_quality/
- └── silver/air\_quality/
- └── gold/air\_quality/

- ← Dữ liệu CSV gốc
- ← Dữ liệu sạch (parquet)
- ← Dữ liệu sẵn sàng train (parquet)

## Bước 4: Train mô hình trên Databricks

### 1. Import thư viện cần thiết

```

import pandas as pd
from sklearn.model_selection import train_test_split
from catboost import CatBoostRegressor
from sklearn.metrics import mean_squared_error, r2_score
from pyspark.sql.functions import lag
from pyspark.sql.window import Window

```

```

15
  ✓ 04:34 AM (10d)
  X pip install catboost
  Python interpreter will be restarted.
  Collecting catboost
    Downloading catboost-1.2.0-cp39-cp39-manylinux2014_x86_64.whl (99.2 kB)
    Collecting pyyaml
      Downloading pyyaml-6.0-py3-none-any.whl (47 kB)
    Requirement already satisfied: numpy<3.0,>=1.16.0 in /databricks/python/lib/python3.9/site-packages (from catboost) (1.20.3)
    Requirement already satisfied: plotly in /databricks/python/lib/python3.9/site-packages (from catboost) (5.9.0)
    Requirement already satisfied: pandas in /databricks/python/lib/python3.9/site-packages (from catboost) (1.3.4)
    Requirement already satisfied: six in /databricks/python/lib/python3.9/site-packages (from catboost) (1.16.0)
    Requirement already satisfied: matplotlib in /databricks/python/lib/python3.9/site-packages (from catboost) (3.4.3)
    Requirement already satisfied: scipy in /databricks/python/lib/python3.9/site-packages (from catboost) (1.7.1)
    Requirement already satisfied: python-dateutil>=2.7.3 in /databricks/python/lib/python3.9/site-packages (from catboost) (2.28.2)
    Requirement already satisfied: pytz in /databricks/python/lib/python3.9/site-packages (from catboost) (2023.1)
    Requirement already satisfied: pillow in /databricks/python/lib/python3.9/site-packages (from catboost) (8.0.4)
    Requirement already satisfied: scikit-learn in /databricks/python/lib/python3.9/site-packages (from catboost) (1.3.1)
    Requirement already satisfied: cycle>=0.29 in /databricks/python/lib/python3.9/site-packages (from matplotlib>catboost) (0.10.0)
    Requirement already satisfied: kiwisolver>=1.0.1 in /databricks/python/lib/python3.9/site-packages (from matplotlib>catboost) (0.1.3.1)
    Requirement already satisfied: pyparsing in /databricks/python/lib/python3.9/site-packages (from matplotlib>catboost) (3.1.2)
    Requirement already satisfied: numpy-base in /databricks/python/lib/python3.9/site-packages (from matplotlib>catboost) (1.20.3)
    Installing collected packages: pyyaml, catboost
    Successfully installed catboost-1.2.0 pyyaml-6.0.20.3
    Python interpreter will be restarted.

```

```

16
  ✓ 04:37 AM (10d)
  import pandas as pd
  from sklearn.model_selection import train_test_split
  from catboost import CatBoostRegressor
  from sklearn.metrics import mean_squared_error, r2_score
  from pyspark.sql.functions import col
  from pyspark.sql.window import Window

```

## 2. Đọc dữ liệu từ Gold Layer

```

df =
spark.read.parquet("abfss://lakehouse@quynhthulakehouse.dfs.core.windows.net/gold/air_
_quality/")
df.printSchema()
df.show(5)

```

```

18
  ✓ 04:37 AM (2d)
  df = spark.read.parquet("abfss://lakehouse@quynhthulakehouse.dfs.core.windows.net/gold/air_quality")
  df.printSchema()
  df.show(5)
  > (2) Spark Jobs
  > df: pyspark.sql.dataframe.DataFrame = (datetime: timestamp, pm25: double ... 10 more fields)
  |-- o3: double (nullable = true)
  |-- co: double (nullable = true)
  |-- no2: double (nullable = true)
  |-- so2: double (nullable = true)
  |-- station: string (nullable = true)
  |-- hour: integer (nullable = true)
  |-- dayofweek: integer (nullable = true)
  |-- day: integer (nullable = true)
  |-- month: integer (nullable = true)

  +-----+-----+-----+-----+-----+-----+-----+-----+
  | datetime|pm25|tsp|o3|co|no2|so2|station|hour|dayofweek|day|month|
  +-----+-----+-----+-----+-----+-----+-----+-----+
2021-02-23 21:00:00	15.6047619	32.9357149	55.43138095	1330.451429	112.7407619	393.0	1	21	3	23	2
2021-02-23 22:00:00	14.5941765	30.93235294	58.19717647	1200.603529	112.366476	377.5882353	1	22	3	23	2
2021-02-23 23:00:00	14.43666667	27.64555.0294333	117.897	112.7004333	137.4766667	1	23	3	23	2	
2021-02-24 00:00:00	12.365	24.38	54.7677	1267.476	112.488667	389.07	1	0	4	24	2
2021-02-24 01:00:00	11.63666667	22.52166667	53.7862	1322.293	114.3315	393.0	1	1	4	24	2
  +-----+-----+-----+-----+-----+-----+-----+-----+
  only showing top 5 rows

```

## 3. Thêm đặc trưng lag để cải thiện dự đoán

Dữ liệu ở đây là thời gian, nên thêm các giá trị PM2.5 trước đó (lag features) có thể giúp mô hình học tốt hơn các mẫu thời gian:

Trong Spark, thêm PM2.5 của 1-3 giờ trước:

```
# Tạo window để tính lag theo station và datetime
w = Window.partitionBy("station").orderBy("datetime")
df = df.withColumn("pm25_lag1", lag("pm25", 1).over(w))
df = df.withColumn("pm25_lag2", lag("pm25", 2).over(w))
df = df.withColumn("pm25_lag3", lag("pm25", 3).over(w))
df = df.na.drop() # Loại bỏ hàng null do lag

# Chuyển sang Pandas
pdf = df.select("pm25", "tsp", "o3", "co", "no2", "so2", "station", "hour",
"dayofweek", "day", "month", "pm25_lag1", "pm25_lag2", "pm25_lag3").toPandas()
```

The screenshot shows a Databricks notebook interface. The notebook title is 'Untitled Notebook 2025-05-25 19:16:05'. The code cell contains the provided Python code for creating lag features and converting the DataFrame to Pandas. The code cell has a green checkmark icon indicating it has been run successfully. The notebook interface includes a header bar with file navigation, a toolbar with various icons, and a sidebar on the right.

```
only showing top 5 rows
```

**Thêm đặc trưng lag để cải thiện dự đoán**

Dữ liệu ở đây là thời gian, nên thêm các giá trị PM2.5 trước đó (lag features) có thể giúp mô hình học tốt hơn các mẫu thời gian:

Trong Spark, thêm PM2.5 của 1-3 giờ trước

```
# Tạo window để tính lag theo station và datetime
w = Window.partitionBy("station").orderBy("datetime")
df = df.withColumn("pm25_lag1", lag("pm25", 1).over(w))
df = df.withColumn("pm25_lag2", lag("pm25", 2).over(w))
df = df.withColumn("pm25_lag3", lag("pm25", 3).over(w))
df = df.na.drop() # Loại bỏ hàng null do lag

# Chuyển sang Pandas
pdf = df.select("pm25", "tsp", "o3", "co", "no2", "so2", "station", "hour",
"dayofweek", "day", "month", "pm25_lag1", "pm25_lag2", "pm25_lag3").toPandas()
```

(2) Spark Jobs

```
df: pyspark.sql.DataFrame = [datetime: timestamp, pm25: double ... 13 more fields]
```

#### 4. Chia dữ liệu train và test

```
# Tách features và target
X = pdf.drop(columns=["pm25"])
y = pdf["pm25"]

# Tách train/test (80/20), giữ thứ tự thời gian
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
shuffle=False)
```

```

Chia dữ liệu train và test

04:39 AM (1s)
# Tách features và target
X = pdf.drop(columns=["pm25"])
y = pdf["pm25"]

# Tách train/test (80/20), giữ thứ tự thời gian
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=False)

```

## 5. Huấn luyện mô hình CatBoost

```

cat_features = ["station"]
model = CatBoostRegressor(iterations=500, depth=6, learning_rate=0.1, verbose=100,
random_seed=42, task_type="CPU")
model.fit(X_train, y_train, eval_set=(X_test, y_test), cat_features=cat_features)

```

```

Huấn luyện mô hình CatBoost

04:39 AM (7s)
cat_features = ["station"]
model = CatBoostRegressor(iterations=500, depth=6, learning_rate=0.1, verbose=100, random_seed=42, task_type="CPU")
model.fit(X_train, y_train, eval_set=(X_test, y_test), cat_features=cat_features)

0: learn: 14.0937082    test: 11.7678931   best: 11.7678931 (0)   total: 18ms   remaining: 9.01s
100: learn: 4.8088405    test: 2.9980817   best: 2.9875794 (97)   total: 1.36s   remaining: 5.36s
200: learn: 4.2492994    test: 2.9454147   best: 2.9213715 (198)   total: 2.69s   remaining: 4.01s
300: learn: 3.9202563    test: 2.8699281   best: 2.8638973 (277)   total: 4s      remaining: 2.65s
400: learn: 3.6779908    test: 2.9381799   best: 2.8599819 (316)   total: 5.34s   remaining: 1.32s
499: learn: 3.4866730    test: 2.9646379   best: 2.8599819 (316)   total: 6.65s   remaining: 0us

besttest = 2.859981941
bestiteration = 316

Shrink model to first 317 iterations.
Out[13]: <catboost.core.CatBoostRegressor at 0x7fc415e2580>

```

CatBoost sử dụng RMSE làm hàm mất mát mặc định cho bài toán hồi quy. Các giá trị learn và test là RMSE trên tập huấn luyện ( $X_{train}$ ,  $y_{train}$ ) và tập kiểm tra ( $X_{test}$ ,  $y_{test}$ ).

**Hiệu suất tốt:** RMSE = 2.86 nhờ lag features,  $R^2$  dự kiến ~0.9. Mô hình CatBoost hiệu quả, thời gian chạy nhanh (6.65s).

## 6. Dự đoán và đánh giá mô hình

```
# Dự đoán trên tập test
y_pred = model.predict(X_test)

rmse = mean_squared_error(y_test, y_pred, squared=False)
r2 = r2_score(y_test, y_pred)
print(f"RMSE: {rmse:.4f}")
print(f"R²: {r2:.4f}")
```

The screenshot shows a Jupyter Notebook interface in Microsoft Azure. The notebook title is "Untitled Notebook 2025-05-25 19:16:05" and it is set to Python. The code cell contains the same prediction and evaluation code as above. The output cell shows the results: RMSE: 2.8600 and R²: 0.9471. Below the notebook, a note explains the benefit of lag features in time series modeling.

```
Shrink model to first 317 iterations.
Out[13]: <catboost.core.CatBoostRegressor at 0x7fc415e2580>
```

Dự đoán và đánh giá mô hình

```
# Dự đoán trên tập test
y_pred = model.predict(X_test)

rmse = mean_squared_error(y_test, y_pred, squared=False)
r2 = r2_score(y_test, y_pred)
print(f"RMSE: {rmse:.4f}")
print(f"R²: {r2:.4f}")

RMSE: 2.8600
R²: 0.9471
```

Tại sao hiệu quả?: Lag features giúp mô hình học các mối quan hệ thời gian (ví dụ: PM2.5 hiện tại phụ thuộc vào PM2.5 trước đó). Điều này thường cải thiện đáng kể  $R^2$ .

Tại sao hiệu quả?: Lag features giúp mô hình học các mối quan hệ thời gian (ví dụ: PM2.5 hiện tại phụ thuộc vào PM2.5 trước đó). Điều này thường cải thiện đáng kể  $R^2$ .

## 7. Lưu model và kết quả

CatBoost không ghi trực tiếp vào abfss://, vì vậy ta cần lưu mô hình vào một tệp tạm thời trên DBFS, sau đó sao chép vào Azure Data Lake Storage:

```
import os

# Lưu mô hình vào /tmp/
temp_path = "/tmp/catboost_pm25_model.cbm"
model.save_model(temp_path)

# Sao chép sang abfss://
```

```

model_save_path =
"abfss://lakehouse@quynhthulakehouse.dfs.core.windows.net/platinum/air_quality_classified/pm25_gbt_model"
dbutils.fs.cp(f"file://{temp_path}", model_save_path)

# Xóa tệp tạm
os.remove(temp_path)

```

```

Lưu model và kết quả

atBoost không ghi trực tiếp vào abfss://, vì vậy ta cần lưu mô hình vào một tệp tạm thời trên DBFS, sau đó sao chép vào Azure Data Lake Storage:

04:45 AM (1s) 30 Python

import os

# Lưu mô hình vào /tmp/
temp_path = "/tmp/catboost_pm25_model.cbm"
model.save_model(temp_path)

# Sao chép sang abfss://
model_save_path = "abfss://lakehouse@quynhthulakehouse.dfs.core.windows.net/platinum/air_quality_classified/pm25_gbt_model"
dbutils.fs.cp(f"file://{temp_path}", model_save_path)

# Xóa tệp tạm
os.remove(temp_path)

```

```

from pyspark.sql import SparkSession

predictions_df = X_test.copy()
predictions_df["pm25"] = y_test
predictions_df["prediction"] = y_pred

# Chuyển sang Spark DataFrame
spark = SparkSession.builder.getOrCreate()
spark_predictions = spark.createDataFrame(predictions_df)

# Lưu vào abfss:// dưới dạng bảng Delta
predictions_path =
"abfss://lakehouse@quynhthulakehouse.dfs.core.windows.net/platinum/air_quality_classified/pm25_predictions"
spark_predictions.write.format("delta").mode("overwrite").save(predictions_path)

```

The screenshot shows a Databricks notebook interface. The notebook title is "Untitled Notebook 2025-05-25 19:16:05" and the language is set to Python. The code cell contains the following Python code:

```
from pyspark.sql import SparkSession

predictions_df = X_test.copy()
predictions_df["pm25"] = y_test
predictions_df["prediction"] = y_pred

# Chuyển sang Spark DataFrame
spark = SparkSession.builder.getOrCreate()
spark_predictions = spark.createDataFrame(predictions_df)

# Lưu vào abfss:// dưới dạng bảng Delta
predictions_path = "abfss://lakehouse@quynhthulakehouse.dfs.core.windows.net/platinum/air_quality_classified/pm25_predictions"
spark_predictions.write.format("delta").mode("overwrite").save(predictions_path)
```

The code is intended to save the predictions DataFrame to a Delta table at the specified path.

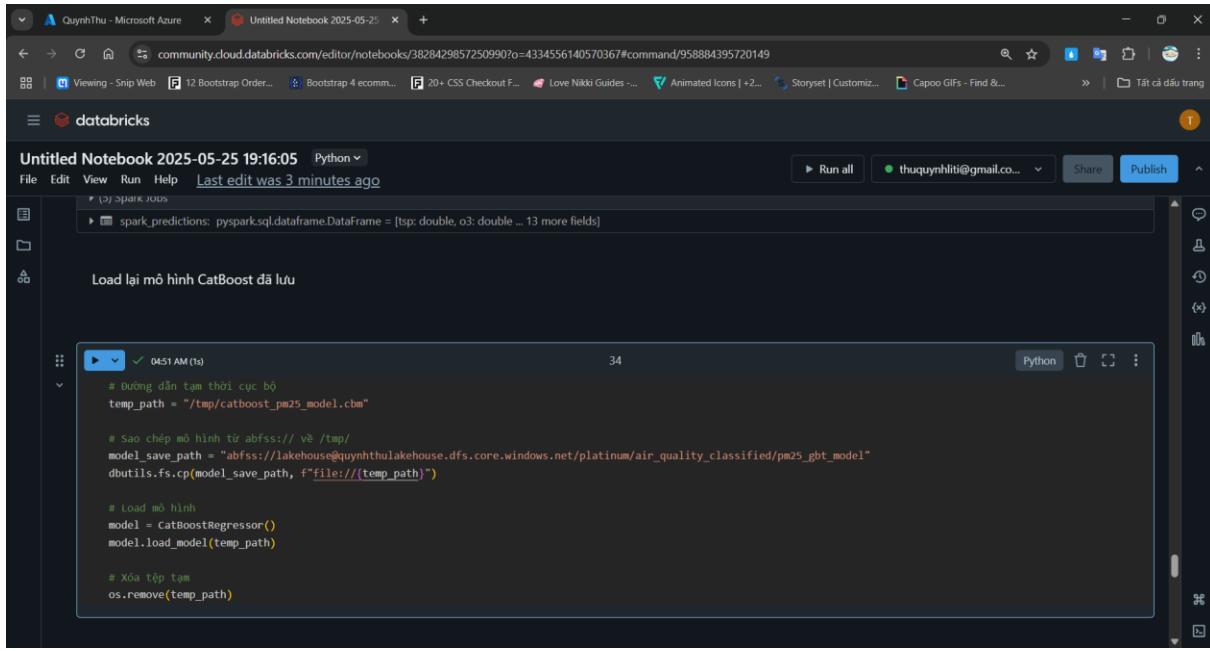
## 8. Load lại mô hình CatBoost đã lưu

```
# Đường dẫn tạm thời cục bộ
temp_path = "/tmp/catboost_pm25_model.cbm"

# Sao chép mô hình từ abfss:// về /tmp/
model_save_path =
"abfss://lakehouse@quynhthulakehouse.dfs.core.windows.net/platinum/air_quality_classified/pm25_gbt_model"
dbutils.fs.cp(model_save_path, f"file:///{temp_path}")

# Load mô hình
model = CatBoostRegressor()
model.load_model(temp_path)

# Xóa tệp tạm
os.remove(temp_path)
```



## 9. Xem lại dữ liệu dự đoán đã lưu (pm25\_predictions)

```
# Khởi tạo Spark session
spark = SparkSession.builder.getOrCreate()

# Load bảng Delta
predictions_path =
"abfss://lakehouse@quynhthulakehouse.dfs.core.windows.net/platinum/air_quality_classified/pm25_predictions"
predictions_df = spark.read.format("delta").load(predictions_path)

# Xem 5 dòng đầu tiên
predictions_df.show(5)
```

```

Xem lại dữ liệu dự đoán đã lưu (pm25_predictions)

# Khởi tạo Spark session
spark = SparkSession.builder.getOrCreate()

# Load bảng Delta
predictions_path = "abfss://lakehouse@quynhthulakehouse.dfs.core.windows.net/platinum/air_quality_classified/pm25_predictions"
predictions_df = spark.read.format("delta").load(predictions_path)

# Xem 5 dòng đầu tiên
predictions_df.show(5)

```

| tsp         | o3          | co          | no2         | so2         | station | hour | dayofweek | day | month                                                               | pm25_lag1 | pm25_lag2 | pm25_lag3 | pm25 | prediction |
|-------------|-------------|-------------|-------------|-------------|---------|------|-----------|-----|---------------------------------------------------------------------|-----------|-----------|-----------|------|------------|
| 24.4464286  | 84.68942857 | 582.0042857 | 111.8781786 | 128.6667143 | 6       | 19   | 3         | 3   | 8 13.50666667 17.13166667 17.7433333 12.78571429 12.626673610769425 |           |           |           |      |            |
| 22.2883333  | 83.68923333 | 576.326     | 111.6967    | 115.28      | 6       | 28   | 3         | 3   | 8 12.78571429 13.50666667 17.13166667 11.805 11.51923477367413      |           |           |           |      |            |
| 20.52166667 | 86.01211667 | 566.203     | 114.3942333 | 104.8       | 6       | 21   | 3         | 3   | 8 11.805 12.78571429 13.50666667 11.11666667 10.924677129315        |           |           |           |      |            |
| 21.00166667 | 82.47871667 | 555.046     | 115.2411333 | 104.8       | 6       | 22   | 3         | 3   | 8 11.11666667 11.805 12.78571429 11.14333333 10.820521362412162     |           |           |           |      |            |
| 28.635      | 78.68358333 | 556.653     | 112.4181333 | 103.49      | 6       | 23   | 3         | 3   | 8 11.14333333 11.11666667 11.805 10.955 10.991404982738851          |           |           |           |      |            |

only showing top 5 rows

## 10. Chuyển từ Delta sang Parquet để Power BI dễ đọc

```

# Đường dẫn lưu file Parquet
parquet_path =
"abfss://lakehouse@quynhthulakehouse.dfs.core.windows.net/platinum/air_quality_classified/pm25_predictions_parquet"

# Chuyển từ Delta sang Parquet
predictions_df.coalesce(1).write.format("parquet").mode("overwrite").save(parquet_path)

```

```

# Đường dẫn lưu file Parquet
parquet_path = "abfss://lakehouse@quynhthulakehouse.dfs.core.windows.net/platinum/air_quality_classified/pm25_predictions_parquet"

# Chuyển từ Delta sang Parquet
predictions_df.coalesce(1).write.format("parquet").mode("overwrite").save(parquet_path)

```

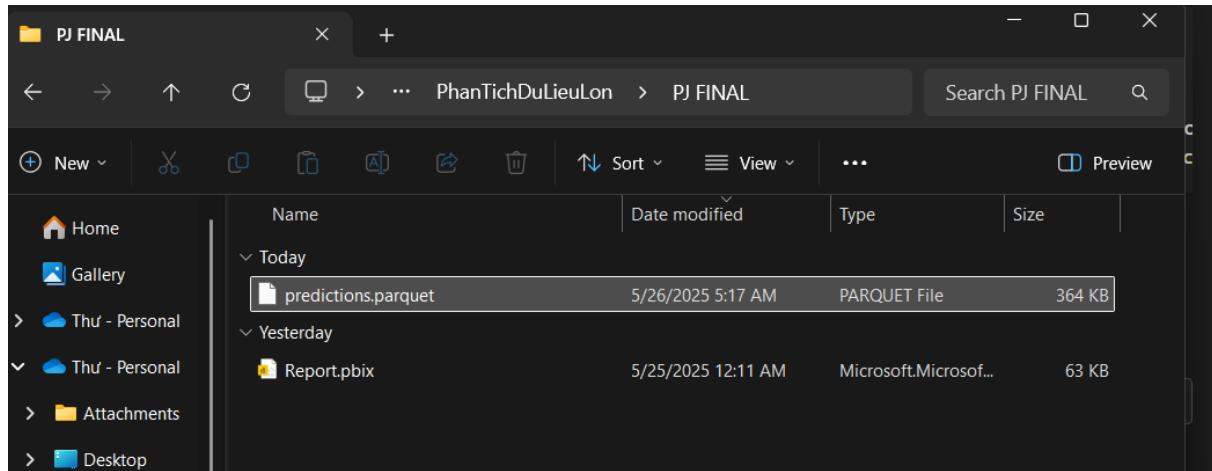
The screenshot shows a Microsoft Azure Databricks notebook interface. The notebook title is "Untitled Notebook 2025-05-25 19:16:05" and the language is Python. The code cell contains the provided Python code to convert a Delta table to Parquet format and save it to a blob storage path. The code is run successfully, indicated by the green checkmark icon.

## Bước 5: Dùng Power BI Desktop

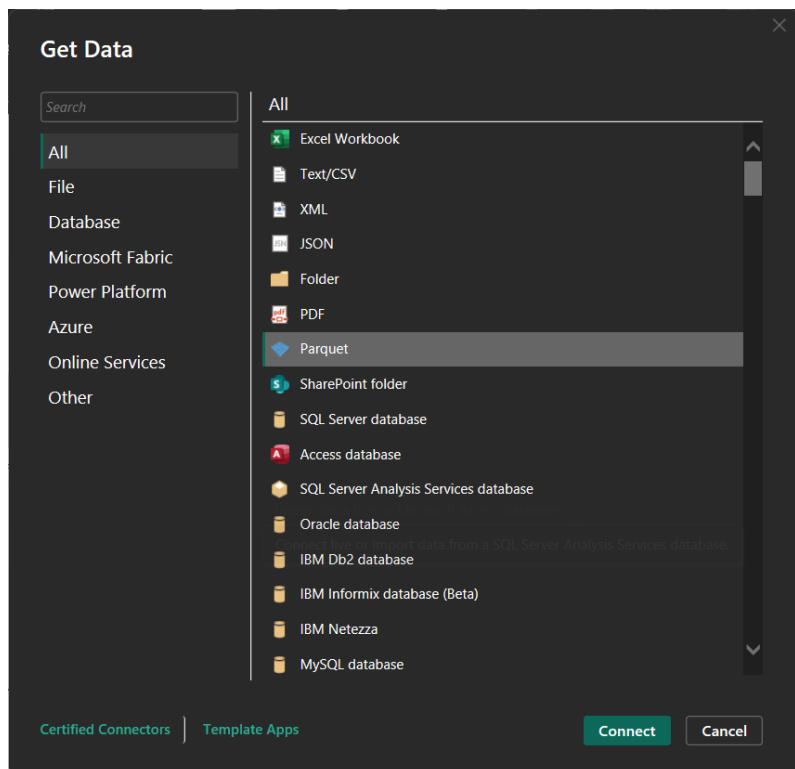
### Chuyển từ Delta sang Parquet/CSV để Power BI dễ đọc

Tải file về máy và đổi tên:

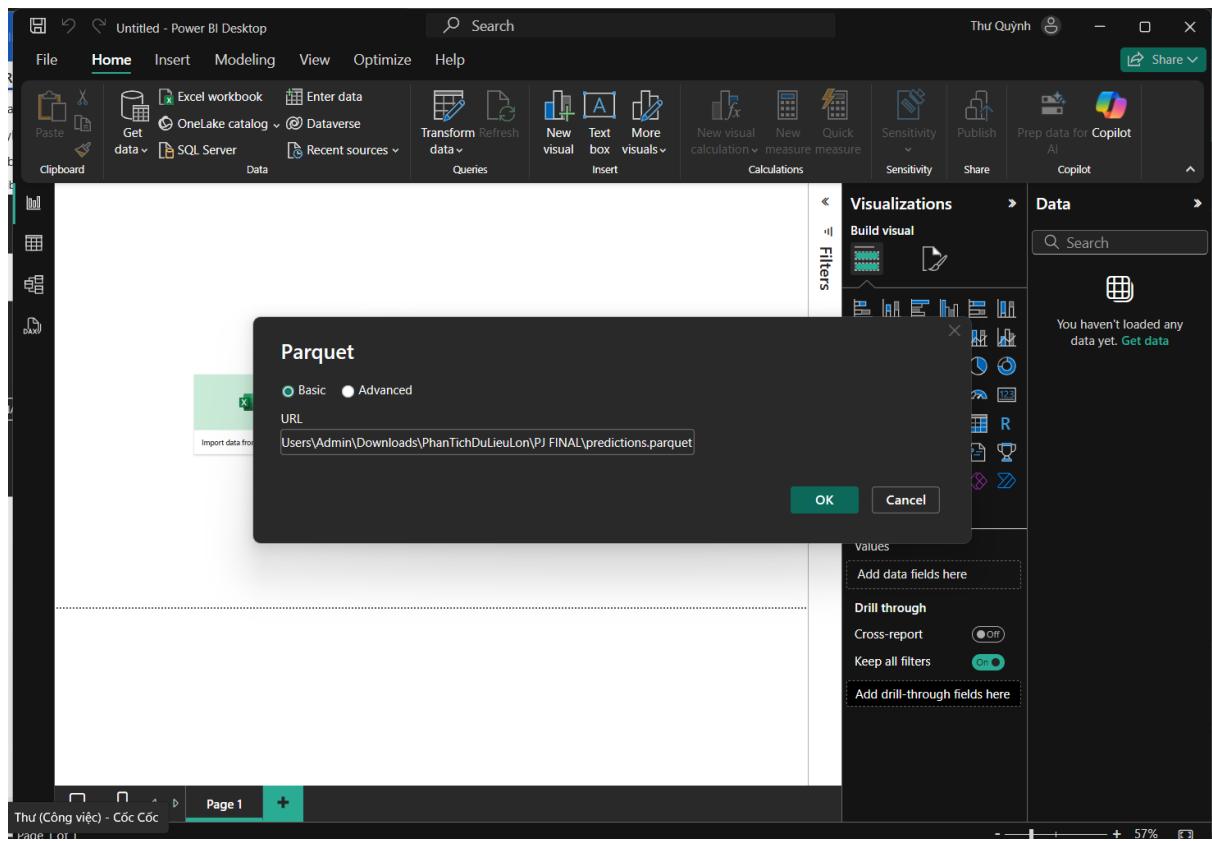
The screenshot shows the Microsoft Azure Storage Explorer interface. The left sidebar shows a tree view of storage accounts and containers. The main pane displays a list of blobs in the 'pm25\_predictions\_parquet' container, including files like '\_committed\_2721933979600306278', '\_started\_2721933979600306278', '\_SUCCESS', and 'part-00000-tid-2721933979600306278-8be8611'. A context menu is open over the '\_committed...' file, with the option 'Select folder for download' highlighted. A small modal window is visible at the bottom right, showing a file selection dialog with the path 'PhanTichDulieuLon\PI FINAL' and a message 'No items match your search.'



Sau đó, vào lại Power BI và chọn **Get Data > Parquet**



Nhập đường dẫn và nhấn OK:



Untitled - Power BI Desktop

File Home Insert Modeling View Optimize Help

Clipboard Get data

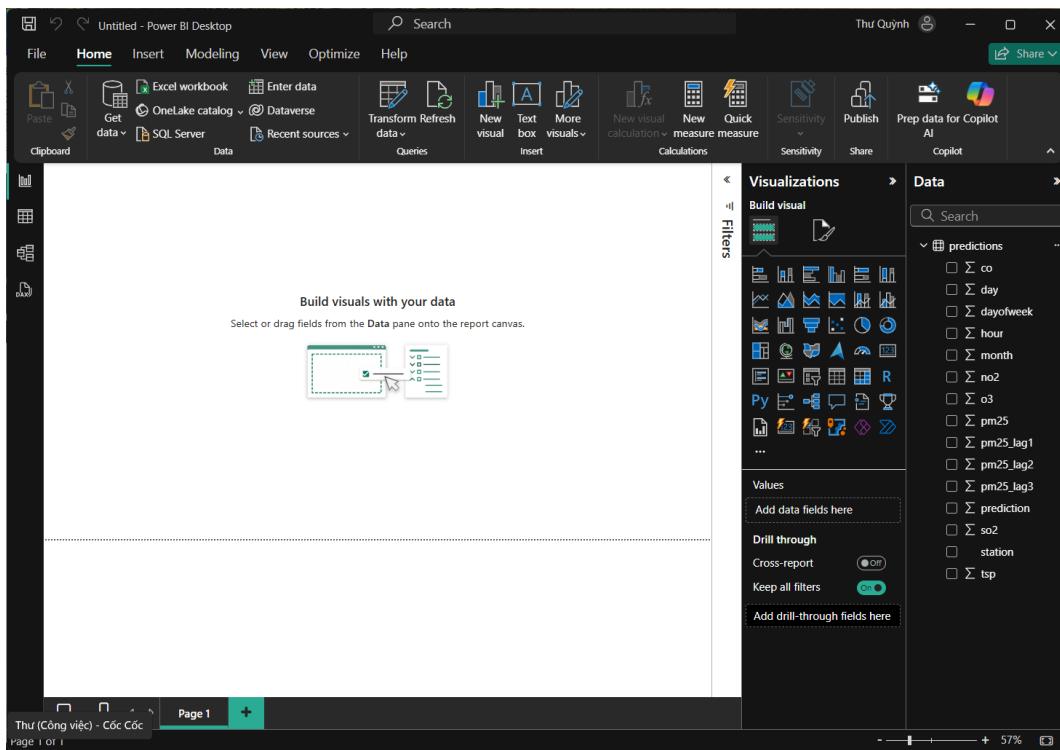
**predictions.parquet**

| tsp         | o3          | co          | no2         | so2         | station | hour | dayofweek | day | month | pm25_lag1   | pm25_lag2   |
|-------------|-------------|-------------|-------------|-------------|---------|------|-----------|-----|-------|-------------|-------------|
| 43.99       | 90.78875    | 963.786     | 144.6630667 | 152.3966667 | 6       | 3    | 6         | 24  | 12    | 23.45833333 | 20.80833333 |
| 44.67166667 | 86.00783333 | 1142.18     | 142.091     | 178.16      | 6       | 4    | 6         | 24  | 12    | 20.3        | 23.45833333 |
| 40.72666667 | 86.30656667 | 1112.575    | 139.0464333 | 155.0166667 | 6       | 5    | 6         | 24  | 12    | 19.245      | 20.3        |
| 53.00666667 | 88.46586667 | 1510.237    | 145.3217667 | 228.8133333 | 6       | 6    | 6         | 24  | 12    | 18.6        | 19.245      |
| 69.435      | 91.4758     | 358.544     | 145.4158667 | 541.03      | 6       | 7    | 6         | 24  | 12    | 23.35666667 | 18.6        |
| 75.13833333 | 112.6107667 | 3743.791    | 151.9401333 | 557.1866667 | 6       | 8    | 6         | 24  | 12    | 27.615      | 23.35666667 |
| 95.06833333 | 143.59345   | 3396.553    | 138.8916    | 475.0933333 | 6       | 9    | 6         | 24  | 12    | 28.76333333 | 27.615      |
| 95.015      | 124.8795167 | 3193.138    | 88.39126667 | 479.46      | 6       | 10   | 6         | 24  | 12    | 36.27166667 | 28.76333333 |
| 55.19333333 | 68.01795    | 1775.345    | 31.38623333 | 341.0366667 | 6       | 11   | 6         | 24  | 12    | 36.02666667 | 36.27166667 |
| 60.635      | 84.67073333 | 1357.246    | 17.6908     | 32.88       | 6       | 12   | 6         | 24  | 12    | 21.12166667 | 36.02666667 |
| 68.48333333 | 97.00491667 | 1486.362    | 8.218066667 | 396.24      | 6       | 13   | 6         | 24  | 12    | 24.90833333 | 21.12166667 |
| 79.35333333 | 175.00145   | 1375.773    | 81.74153333 | 351.08      | 6       | 14   | 6         | 24  | 12    | 27.00333333 | 24.90833333 |
| 56.27       | 200.6840333 | 1352.662    | 107.1843333 | 294.3133333 | 6       | 15   | 6         | 24  | 12    | 30.32       | 27.00333333 |
| 36.015      | 172.4822667 | 1789.097    | 160.5346    | 296.4966667 | 6       | 16   | 6         | 24  | 12    | 20.62       | 30.32       |
| 86.53666667 | 171.66435   | 3009.014    | 187.8236    | 480.77      | 6       | 17   | 6         | 24  | 12    | 15.715      | 20.62       |
| 49.36166667 | 152.62325   | 3803.192    | 191.6817    | 600.4166667 | 6       | 18   | 6         | 24  | 12    | 23.54833333 | 15.715      |
| 73.68333333 | 120.1028833 | 4304.758    | 166.3374333 | 670.72      | 6       | 20   | 6         | 24  | 12    | 21.785      | 23.54833333 |
| 57.20689655 | 104.6820517 | 1756.541379 | 139.268     | 271.937931  | 6       | 21   | 6         | 24  | 12    | 25.985      | 21.785      |
| 53.09833333 | 106.0347167 | 1511.383    | 146.1686667 | 232.3066667 | 6       | 22   | 6         | 24  | 12    | 21.66896552 | 25.985      |
| 62.305      | 108.3576    | 1810.298    | 158.5585    | 273.3533333 | 6       | 23   | 6         | 24  | 12    | 20.22666667 | 21.66896552 |

The data in the preview has been truncated due to size limits.

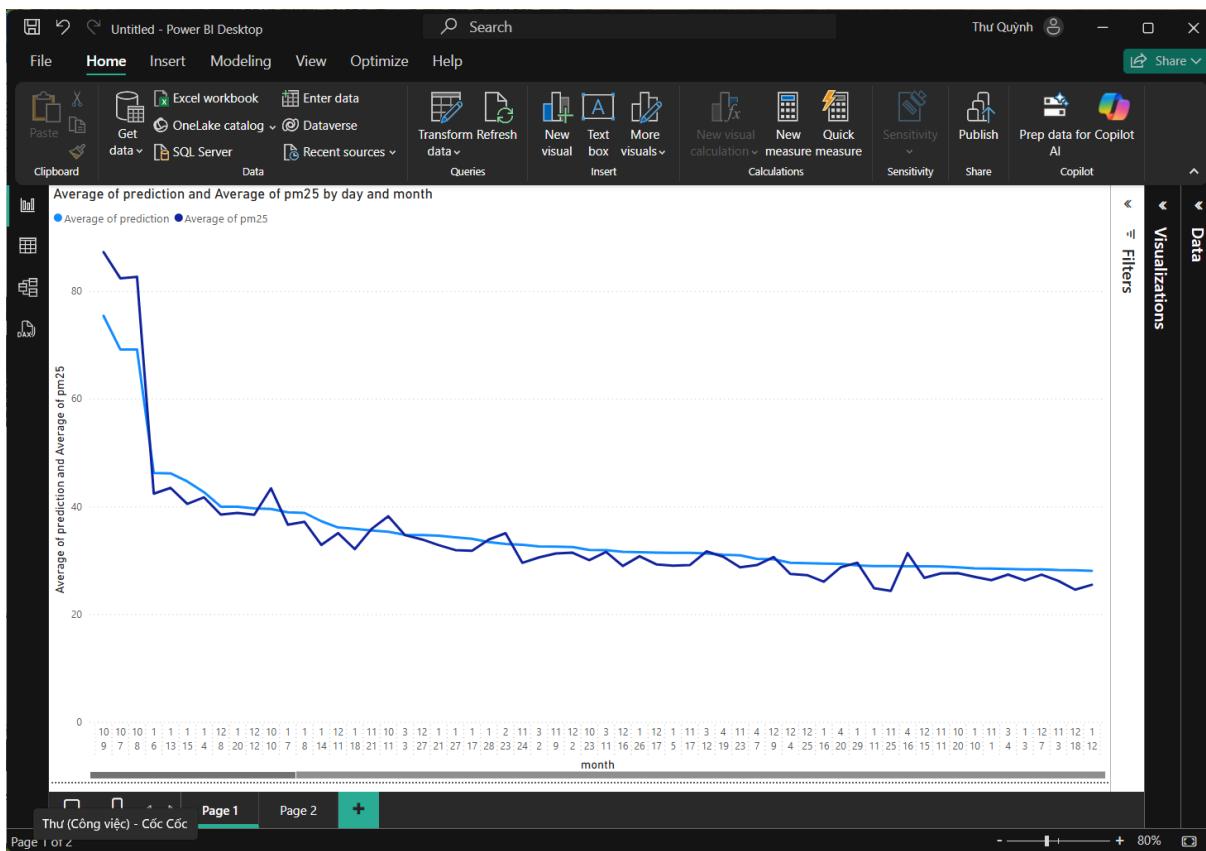
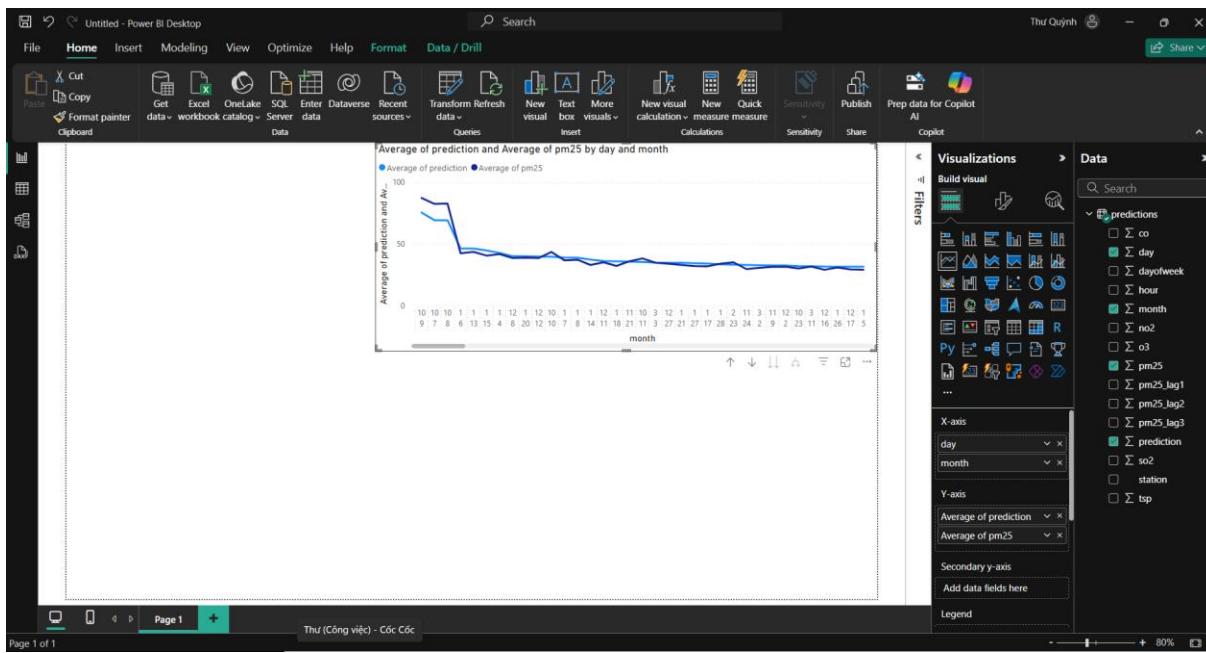
Load Transform Data Cancel

Nhấn Load data vào powerbi:



Tạo biểu đồ so sánh thực tế và dự đoán:

- Thêm một Line Chart từ mục Visualizations.
- Đặt:
  - o Axis: hour hoặc day (hoặc kết hợp month, day để tạo trục thời gian).
  - o Values: pm25 (thực tế) và prediction (dự đoán).
  - o Legend: station (nếu muốn chia theo trạm).
- Kết quả: Biểu đồ đường so sánh PM2.5 thực tế và dự đoán theo thời gian.



## KẾT LUẬN

Qua quá trình thực hiện dự án "Tìm hiểu cách tạo pipeline tự động hóa Lakehouse, tạo thành một Lakehouse thống nhất khi dùng Airflow", chúng em đã thành công trong việc xây dựng và triển khai một hệ thống xử lý dữ liệu hoàn chỉnh, từ khâu thu thập dữ liệu đầu vào đến trực quan hóa thông tin cho người dùng cuối. Dự án đã đạt được những mục tiêu chính đề ra, cụ thể:

### 1. Xây dựng thành công pipeline dữ liệu batch tự động hóa:

- Dữ liệu CSV đã được tải vào lớp Bronze trên HDFS.
- Delta Table đã được tạo và ứng dụng hiệu quả cho các lớp Bronze, Silver, Gold và Platinum, đảm bảo tính nhất quán, độ tin cậy và khả năng quản lý phiên bản dữ liệu.
- Các bước tiền xử lý (loại bỏ null, trùng lặp), biến đổi (tính toán trung bình AQI theo ngày), và chuẩn hóa dữ liệu đã được thực hiện bằng PySpark, tuần tự đưa dữ liệu qua các lớp Bronze -> Silver -> Gold -> Platinum.
- Toàn bộ quy trình được tự động hóa và điều phối bởi Apache Airflow.

### 2. Tích hợp thành công pipeline dữ liệu streaming:

- Hệ thống Kafka đã được cài đặt và cấu hình để hoạt động như một message broker.
- Producer Kafka đã được phát triển để lấy dữ liệu liên tục từ API.
- Spark Streaming Consumer đã được triển khai để tiêu thụ dữ liệu từ Kafka và ghi vào Bronze Delta Table, đảm bảo dữ liệu mới nhất luôn được cập nhật vào Lakehouse.

### 3. Kết nối và trực quan hóa dữ liệu hiệu quả:

- Đã tìm hiểu và thực hiện thành công việc kết nối Power BI với Delta Table lưu trữ trên HDFS.
- Các dashboard trực quan đã được xây dựng trên Power BI, cung cấp cái nhìn sâu sắc về dữ liệu AQI và các thông tin liên quan khác từ dữ liệu đã qua xử lý ở lớp Gold và Platinum.

**4. Nắm vững và ứng dụng các công nghệ chủ chốt:** Dự án đã mang lại kinh nghiệm thực tiễn quý báu trong việc làm việc với Apache Airflow, PySpark, Delta Lake, Apache Kafka, Spark Streaming và Power BI, cũng như hiểu rõ hơn về kiến trúc Lakehouse và lợi ích mà nó mang lại. Các minh chứng chi tiết về cấu hình và kết quả từng bước đã được ghi nhận.

#### Những bài học kinh nghiệm và hướng phát triển tương lai:

- **Tầm quan trọng của Delta Lake:** Delta Lake thực sự là một yếu tố thay đổi cuộc chơi trong việc xây dựng Data Lake đáng tin cậy, khắc phục nhiều nhược điểm của Data Lake truyền thống.
- **Sức mạnh của tự động hóa:** Airflow chứng tỏ là một công cụ điều phối mạnh mẽ, giúp quản lý các workflow phức tạp một cách hiệu quả và giảm thiểu can thiệp thủ công.
- **Thách thức của dữ liệu streaming:** Xử lý dữ liệu streaming đòi hỏi sự chú ý đặc biệt đến khả năng chịu lỗi, độ trễ và quản lý trạng thái.

Trong tương lai, hệ thống có thể được mở rộng và cải tiến thêm bằng cách:

- Tích hợp thêm các nguồn dữ liệu đa dạng khác.
- Áp dụng các thuật toán Machine Learning trên lớp Gold để đưa ra các dự đoán hoặc phân tích nâng cao.
- Tối ưu hóa hiệu suất xử lý và lưu trữ cho khối lượng dữ liệu lớn hơn.
- Triển khai các cơ chế bảo mật và quản trị dữ liệu (data governance) chặt chẽ hơn.

Tóm lại, dự án này không chỉ hoàn thành các yêu cầu kỹ thuật đặt ra mà còn cung cấp một nền tảng vững chắc và kiến thức thực tiễn về việc xây dựng các hệ thống dữ liệu hiện đại theo kiến trúc Lakehouse. Đây là một minh chứng rõ ràng về khả năng tích hợp các công nghệ mã nguồn mở mạnh mẽ để giải quyết các bài toán dữ liệu phức tạp trong thực tế.

## TÀI LIỆU THAM KHẢO

1. Apache Spark, *Structured Streaming + Kafka Integration Guide*, Version 3.5.5. [Online]. Available: <https://spark.apache.org/docs/latest/structured-streaming-kafka-integration.html>
2. Apache Kafka, *Apache Kafka Quickstart*. [Online]. Available: <https://kafka.apache.org/quickstart>
3. Conduktor, *How to Install Apache Kafka on Linux?* [Online]. Available: <https://learn.conduktor.io/kafka/how-to-install-apache-kafka-on-linux/>
4. A. H. Payberah, *Lab3 - Kafka and Spark Streaming*.
5. Databricks, "Data Warehousing Modeling Techniques and Their Implementation on the Databricks Lakehouse Platform," *Databricks Blog*, 2022, [Online]. Available: <https://www.databricks.com/blog/2022/06/24/data-warehousing-modeling-techniques-and-their-implementation-on-the-databricks-lakehouse-platform.html>. [Accessed: 10 May 2025].
6. Apache Software Foundation, "Machine Learning Library (MLlib) Guide," *Apache Spark Documentation*, 2024. [Online]. Available: <https://spark.apache.org/docs/latest/ml-guide.html>. [Accessed: 20 May 2025].
7. Databricks, "Databricks documentation - Databricks on AWS," *Databricks Documentation*. [Online]. Available: <https://docs.databricks.com/en/aws/>. [Accessed: 14 May 2025].
8. Rajnish Rakholia, Quan Le, Khue Hoang Ngoc Vu, Bang Quoc Ho, "Ricardo Simon Carbajo, *Outdoor air quality data for spatiotemporal analysis and air quality modelling in Ho Chi Minh City, Vietnam: A part of HealthyAir Project*", Data in Brief, Volume 46, February 2023, Pages 108774.
9. Delta Lake Community, "Delta Lake 3.3.0," *GitHub*, 2024. [Online]. Available: <https://github.com/delta-io/delta/releases/tag/v3.3.0>. [Accessed: 18 May 2025].