**REPORT LAB 8 + LAB 9**

Thành viên:

23673611 Phan Khánh Khoa

23667351 Trương Tuấn Binh

23644831 Đỗ Lê Vinh

**Lab 9**

Ex1 : Dowload các file từ đường link cho trước

1. Các bước thực hiện:

B1. Khởi tạo thư mục:

* Kiểm tra và tạo thư mục downloads bên trong Exercise-1 (nếu chưa có).

B2. Tải từng file:

* Duyệt danh sách URL (có sẵn trong main.py).
* Với mỗi URL, dùng requests để tải về.
* Trích lấy tên file gốc từ URL (lấy phần sau dấu / cuối cùng).

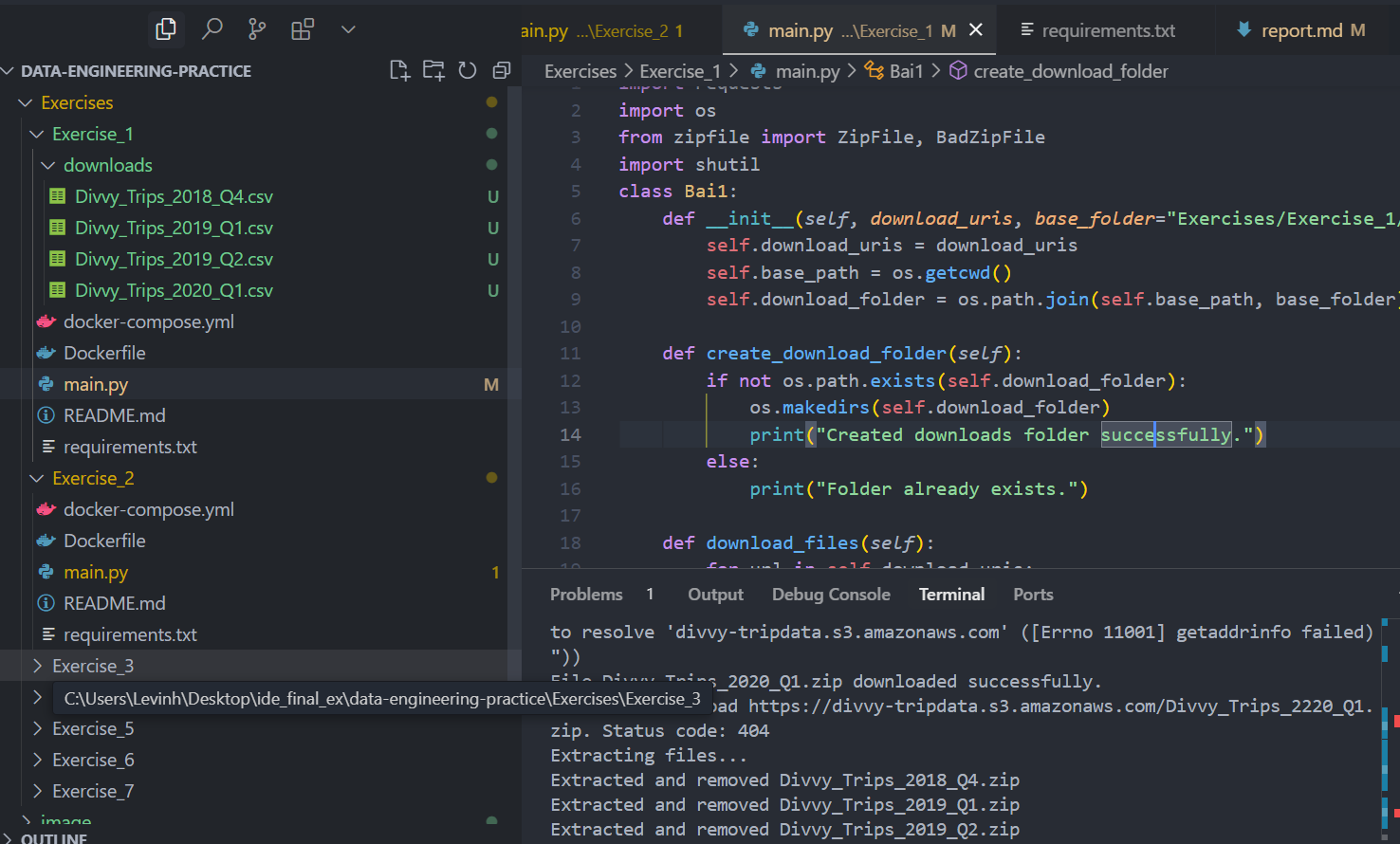
B3: Giải nén ZIP

* Mỗi file tải về là một ZIP, giải nén bằng thư viện zipfile để lấy file CSV bên trong.
* Đặt file CSV vào thư mục downloads.

B4: Dọn dẹp

* Dùng shutil xóa file ZIP sau khi đã giải nén thành công, chỉ giữ lại CSV.

1. Kết quả: Thành công tải, giải nén và xóa các file zip



Ex2: Tải file và tìm dữ liệu theo yêu cầu

1. Các bước thực hiện

B1. Khởi tạo đối tượng:

* base\_url: thư mục chứa danh sách các file
* target\_date: chuỗi ngày-giờ cần tìm, định dạng "YYYY-MM-DD HH:MM".
* output\_folder: nơi lưu file tải về
* Chuyển target\_date thành đối tượng datetime và tạo thư mục lưu trữ nếu chưa tồn tại.

B2. Lấy danh sách file từ server

* Gửi GET request tới base\_url.
* Dùng BeautifulSoup phân tích HTML trả về.

B3. Tìm file có thời gian khớp

* Duyệt tất cả các thẻ <a> trong bảng.
* Với mỗi link, lấy hàng (<tr>) chứa nó, kiểm tra xem text trong hàng có chứa đúng chuỗi target\_date không.
* Khi tìm thấy, nối base\_url với href của link để ra URL đầy đủ của file.

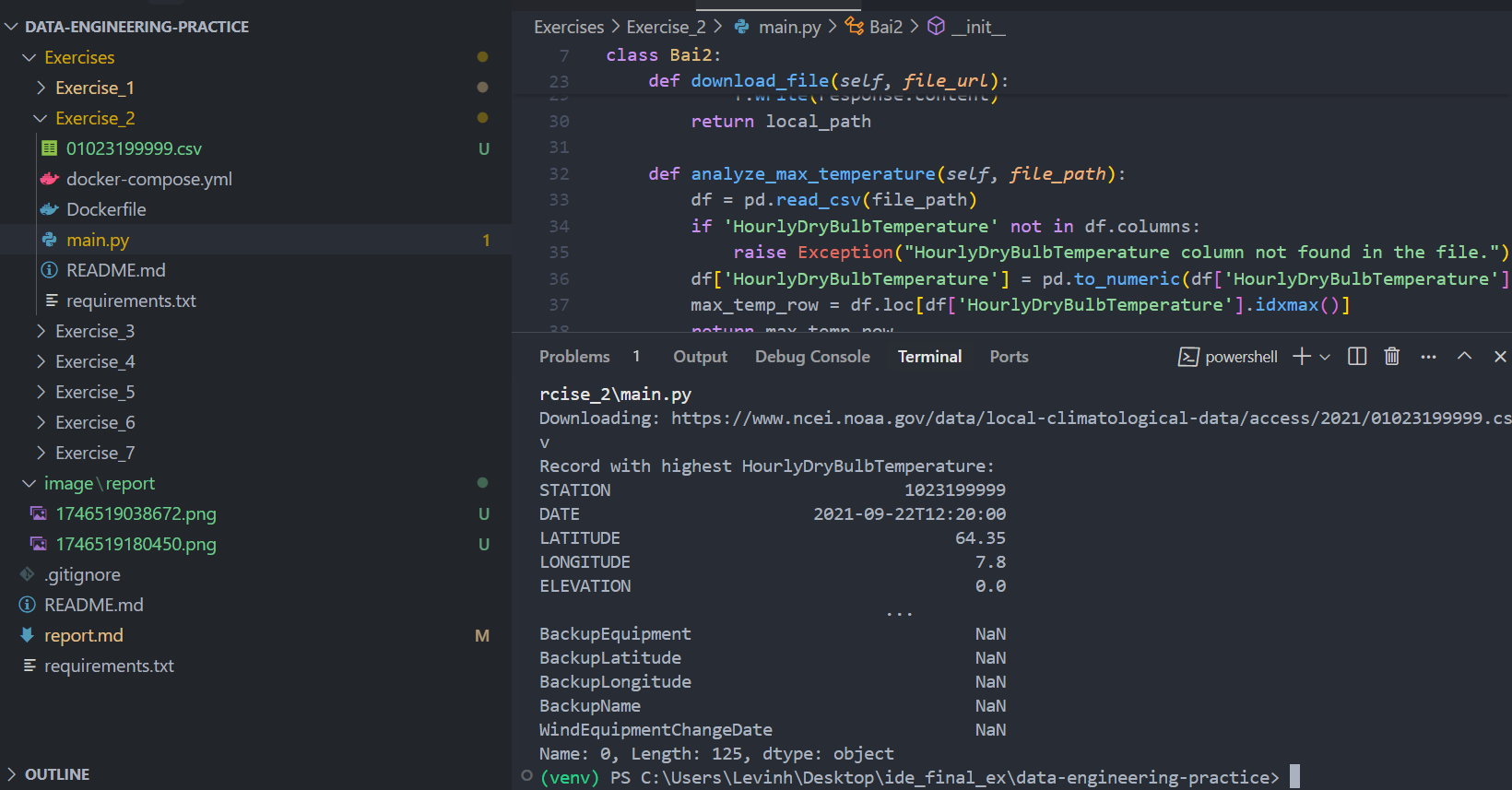
B4. Tải file về máy

* Gửi GET request tới URL file.
* Ghi nội dung nhị phân vào file trong thư mục output\_folder.
* Trả về đường dẫn file đã lưu.

B5. Phân tích dữ liệu nhiệt độ

* Đọc file CSV bằng pandas.
* Kiểm tra cột HourlyDryBulbTemperature có tồn tại không, nếu không có sẽ báo lỗi.
* Chuyển giá trị trong cột này về số thực, bỏ giá trị không chuyển được thành NaN.
* Tìm dòng có nhiệt độ cao nhất (.idxmax()) và trả về record đó.

1. Kết quả: Thành công tải file và tìm dữ liệu theo yêu cầu



Ex 3: Tải và hiển thị file dữ liệu

1. Các bước thực hiện:

B1. Xây dựng đường dẫn

* Xây dựng URL từ base\_url + path.

B2. Tải file .gz đầu tiên về

* Gửi HTTP GET (stream) với requests.
* Trả về nội dung nhị phân (bytes) của file.

B3. Giải nén nội dung .gz

* Đưa bytes vào io.BytesIO, mở bằng gzip.GzipFile.
* Đọc toàn bộ, decode UTF-8 (thay lỗi bằng ký tự thay thế) và trả về chuỗi.

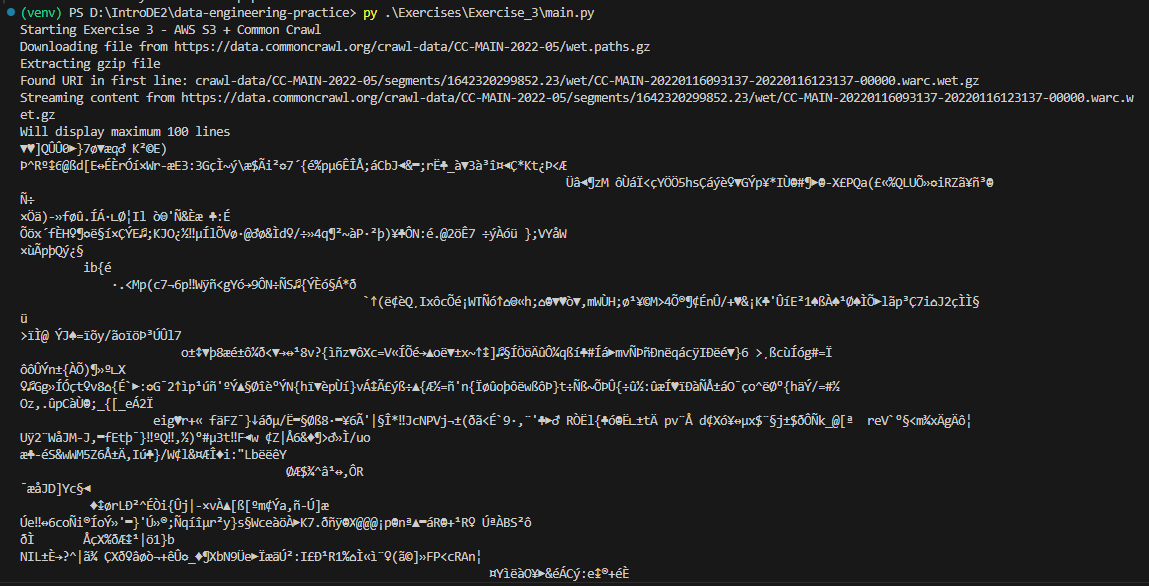
B4. Lấy URI từ dòng đầu

* Tách đoạn text theo dấu xuống dòng, lấy phần tử đầu tiên.
* Trả về URI (chuỗi) để sử dụng bước sau.

B5. Stream và in nội dung file thứ hai

* Tương tự download\_file, nhưng giữ streaming và dùng iter\_lines().
* Lặp in từng dòng (decode UTF-8, fallback sang Latin-1 nếu lỗi).
* Dừng sau max\_lines(100 dòng ) và báo tổng dòng đã xử lý.

1. Kết quả: Thành công tải và hiển thị dữ liệu



Ex4: Chuyển file json sang csv

1. Các bước thực hiện:  
   B1. Duyệt các file JSON

* Dùng glob.glob với recursive=True tìm tất cả các file .json trong thư mục Exercise\_4/data và các thư mục con.

B2. Đọc nội dung JSON

* Với mỗi file, mở và load nội dung bằng json.load().
* Nếu JSON ở dạng một object đơn (dict), chuyển thành list chứa một phần tử để đồng nhất xử lý.

B3. Làm phẳng (flatten) cấu trúc JSON lồng nhau

* Đệ quy qua các dict và list:

+ Với dict: nối tên key xuống dưới, phân tách bằng dấu \_.

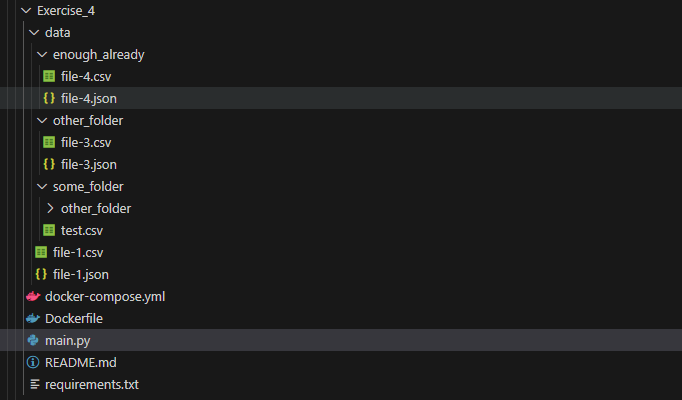
+ Với list: đánh chỉ số phần tử, gắn vào tên trường.

* Kết quả là một dict phẳng, mỗi key là đường dẫn đến giá trị gốc và value là giá trị tương ứng.

B2. Ghi ra file CSV

* Lấy tập hợp các keys từ record đầu tiên làm header.
* Dùng csv.DictWriter ghi header và các dòng dữ liệu đã được flatten.

1. Kết quả: Thành công chuyển các file jsol thành csv:



Ex5 : Tạo bảng và import dữ liệu vào database

1. Các bước thực hiện

B1. Khởi tạo kết nối và con trỏ

* Nhận đối tượng connection (psycopg2) và đường dẫn thư mục dữ liệu.
* Tạo cursor để thực thi SQL.

B2. Tạo cấu trúc bảng (nếu chưa có)

* Chạy một khối SQL gồm ba lệnh CREATE TABLE IF NOT EXISTS cho các bảng accounts, products, transactions.
* Tạo các index cần thiết để tối ưu truy vấn.

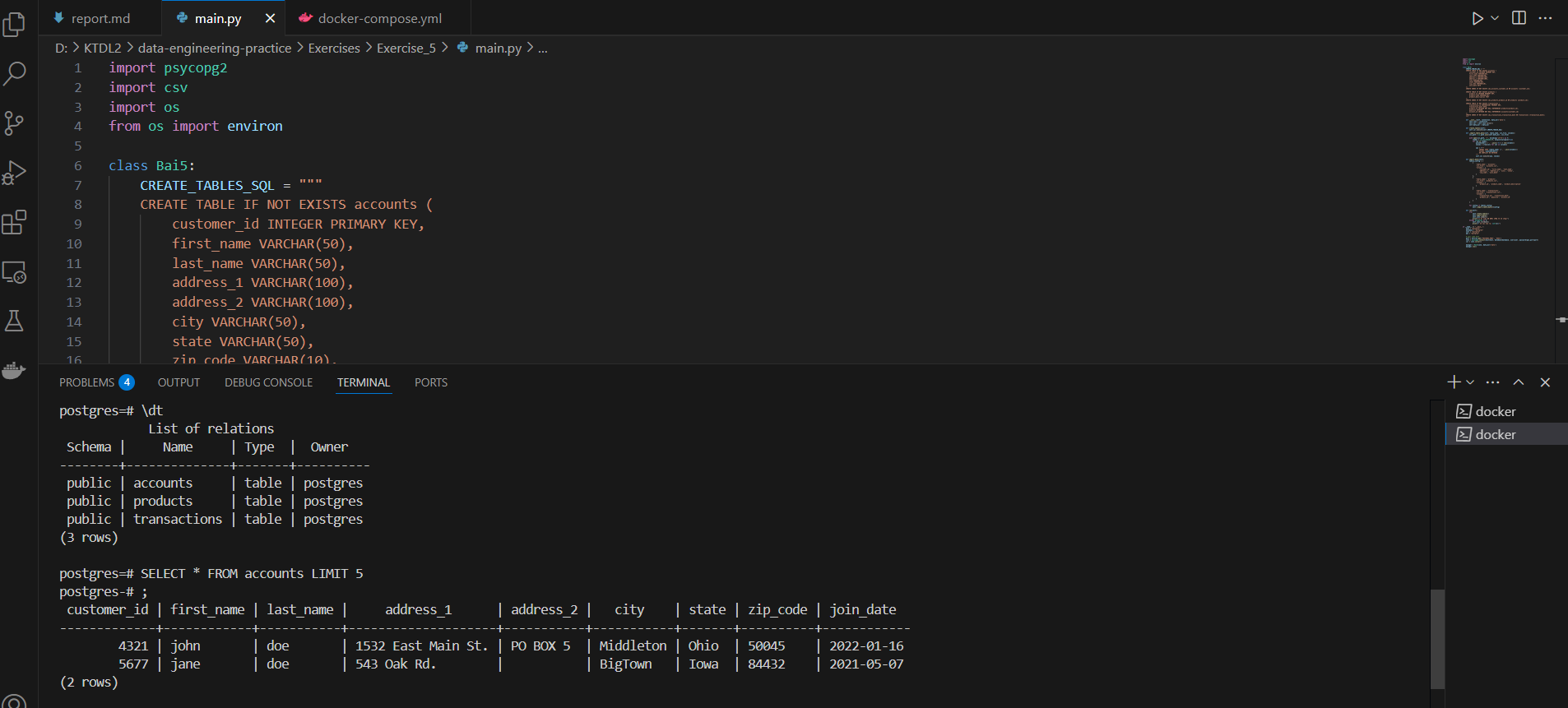
B3. Đọc và nhập dữ liệu từ CSV

* Định nghĩa cấu hình cho từng bảng: tên bảng, tên file CSV, và danh sách cột.
* Với mỗi bảng:
  + Mở file CSV tương ứng (được đặt trong thư mục data\_dir).
  + Dùng csv.DictReader đọc từng dòng, gom giá trị theo danh sách cột.
  + Chèn vào bảng bằng lệnh INSERT … ON CONFLICT DO NOTHING để tránh lỗi trùng khóa chính.

B4. Commit hoặc rollback

* Nếu toàn bộ các thao tác tạo bảng và nhập dữ liệu thành công, gọi conn.commit() và in thông báo thành công.
* Nếu có lỗi trong quá trình, gọi conn.rollback() và in lỗi ra màn hình.

1. Kết quả: Thành công tạo bảng và import vào database



Ex6: Giải nén file và tính toán dữ liệu với spark

1. Các bước thực hiện:

B1. Giải nén dữ liệu

* Duyệt tất cả file .zip trong thư mục gốc (zip\_folder).
* Giải nén từng file vào thư mục đích (extract\_to).

B2. Khởi tạo SparkSession

* Tạo một SparkSession với tên ứng dụng "Exercise6".

B3. Đọc dữ liệu CSV

* Đọc tất cả file .csv trong thư mục đã giải nén thành một DataFrame, với header và schema tự động suy đoán.

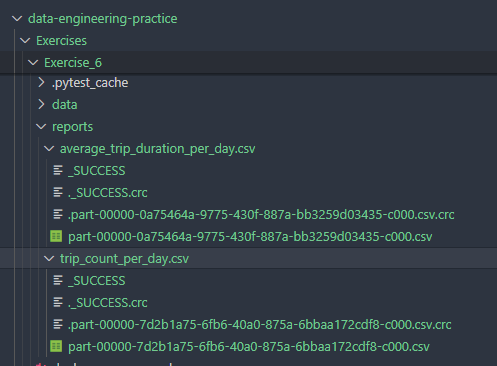
B4. Tính toán và ghi báo cáo

* Average trip duration per day: nhóm theo ngày (chuyển start\_time sang date), tính trung bình tripduration, ghi ra reports/average\_trip\_duration\_per\_day.
* Trips per day: nhóm theo ngày, đếm số chuyến, ghi ra reports/trips\_per\_day.
* Most popular start station per month: thêm cột year, month lấy từ start\_time, nhóm theo (year, month, from\_station\_name) rồi chọn station có số chuyến cao nhất mỗi tháng, ghi ra reports/most\_popular\_start\_station\_per\_month.
* Top 3 stations last two weeks: xác định 14 ngày gần nhất, đếm chuyến mỗi ngày và mỗi station, lấy top 3 theo số chuyến mỗi ngày, ghi ra reports/top\_3\_trip\_stations\_last\_two\_weeks.
* Gender trip duration comparison: nhóm theo gender, tính trung bình tripduration, ghi ra reports/gender\_trip\_duration\_comparison.
* Top 10 ages longest/shortest trips: tính tuổi từ birthyear, sắp xếp để lấy 10 tuổi có chuyến dài nhất và 10 tuổi có chuyến ngắn nhất, ghi hai báo cáo trong reports/.

B2. Dừng Spark

* 1. Gọi spark.stop() để giải phóng tài nguyên.

1. Kết quả: Thành công phân tích và xuất dữ liệu phân tích ra file csv.



Ex7: Đọc file và xữ lí dữ liệu với spark

1. Các bước thực hiện

B1. Khởi tạo SparkSession

* Tạo đối tượng SparkSession với tên ứng dụng và kích hoạt Hive support.

B2. Đọc dữ liệu từ file ZIP

* Mở file ZIP (zip\_path) và đọc nội dung của file CSV bên trong (inner\_file) dưới dạng bytes.
* Chuyển bytes thành một RDD các dòng text.
* Dùng spark.read.csv với header và inferSchema để tạo DataFrame.

B3. Gắn nhãn nguồn file

* Tạo cột mới source\_file chứa tên file gốc, để truy vết nguồn dữ liệu.

B4. Trích xuất ngày từ tên file

* Dùng biểu thức chính quy tìm pattern YYYY-MM-DD trong source\_file.
* Chuyển thành cột file\_date kiểu Date.

B5. Tách nhãn “brand” từ cột model

* Nếu model chứa dấu cách thì lấy phần trước dấu cách làm brand.
* Ngược lại gán brand = 'unknown'.

B6. Tính thứ hạng storage

* Lấy danh sách các cặp (model, capacity\_bytes) duy nhất.
* Dùng dense\_rank() trên capacity\_bytes giảm dần để tạo cột storage\_ranking (số thứ tự, không bỏ sót bậc).
* Join lại với DataFrame chính để thêm storage\_ranking.

B7. Loại bỏ bản ghi trùng lặp

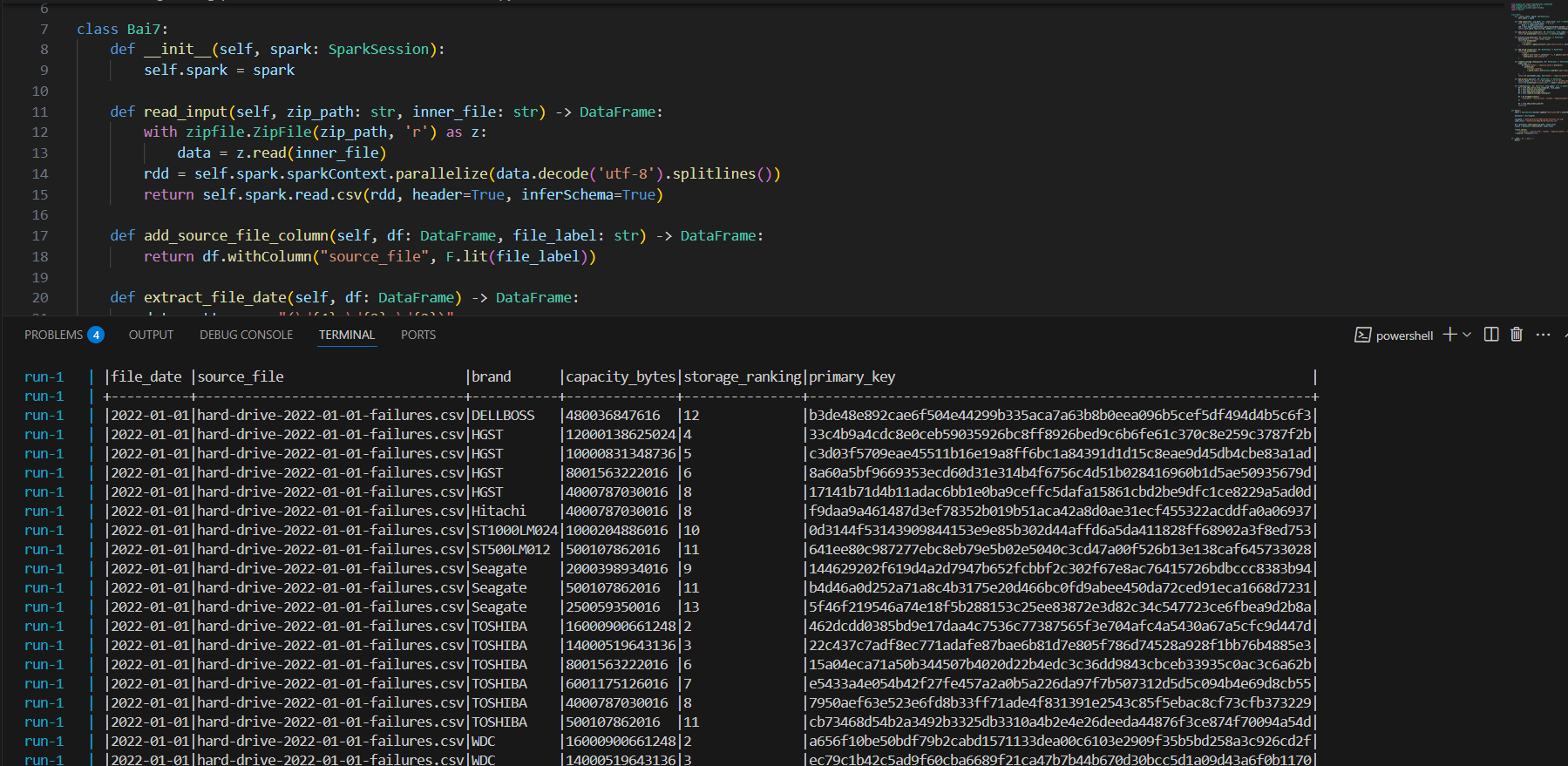
* Dùng dropDuplicates với tổ hợp các cột:  
   file\_date, source\_file, brand, capacity\_bytes, storage\_ranking.

B8. Tạo khóa chính (primary\_key)

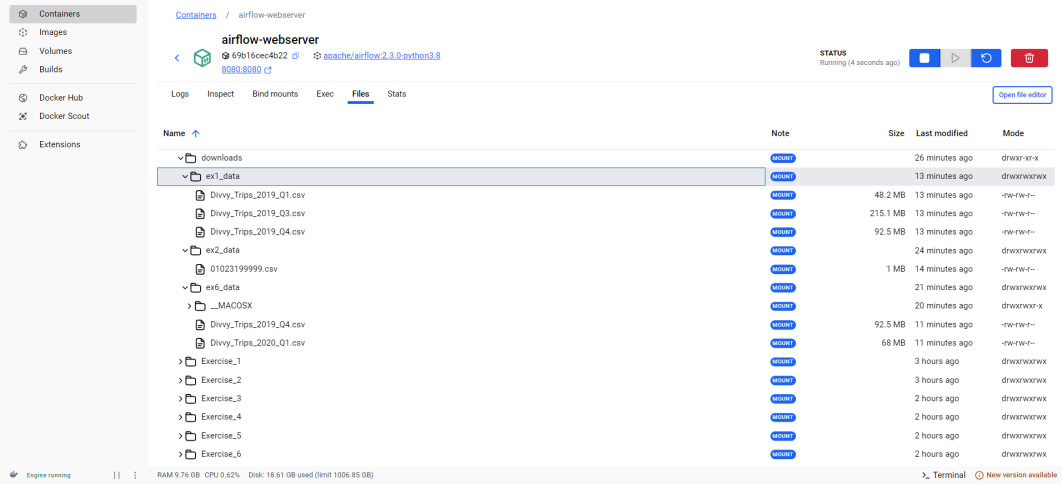
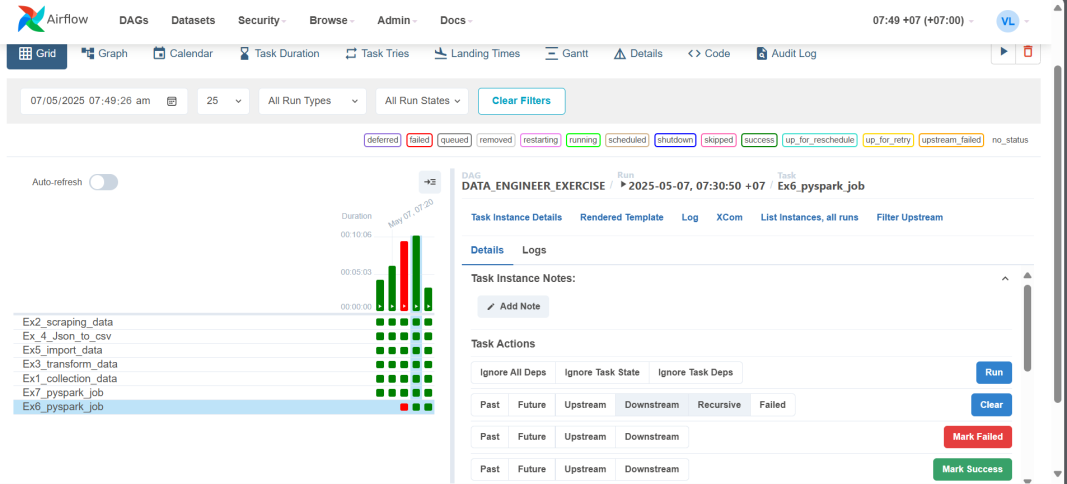
* Nối tất cả cột (ngoại trừ primary\_key) bằng ký tự phân tách '||'.
* Hash SHA-256 kết quả để ra một chuỗi băm làm primary\_key.

B9. Hiển thị kết quả

* Chọn các cột cần thiết:  
   file\_date, source\_file, brand, capacity\_bytes, storage\_ranking, primary\_key.
* Gọi .show() để in ra console 20 dòng đầu.

1. Kết quả: Thành công giải nén và xử lí dữ liệu 

Tổng hợp lại thành 1 pipeline à chạy trong AIRFLOW



**LAB 8**

**Bài 1: Tự động thực hiện quy trình ETL**

1. Các bước thực hiện

B1. Trích Xuất Dữ Liệu Từ Cơ Sở Dữ Liệu MySQL

- Sử dụng lớp MySQLOperators từ mysql\_operator.py để kết nối với cơ sở dữ liệu MySQL.

- Truy xuất dữ liệu thô từ các bảng liên quan đến thương mại điện tử.

- Sử dụng phương thức get\_data\_to\_pd từ MySQLOperators để lấy dữ liệu dưới dạng DataFrame.

- Sử dụng lớp PostgresOperators từ postgresql\_operator.py để lưu dữ liệu thô vào các bảng tạm trong schema staging của PostgreSQL, với tên bảng có tiền tố stg\_

B2. Biến Đổi Dữ Liệu Để Tạo Các Bảng Dimension

-Sử dụng phương thức get\_data\_to\_pd từ PostgresOperators (postgresql\_operator.py) để đọc dữ liệu từ các bảng staging.

-Thực hiện các thao tác làm sạch, chuẩn hóa, và tạo khóa thay thế (surrogate keys) để xây dựng các bảng dimension trong schema warehouse:

+ dim\_customers: Chuẩn hóa mã bưu điện, tên thành phố, tiểu bang, tạo khóa thay thế (customer\_key) và cột theo dõi thay đổi (effective\_date, end\_date, is\_current).

+ dim\_products: Kết hợp dữ liệu sản phẩm và danh mục, làm sạch trọng lượng/kích thước, tạo khóa thay thế (product\_key) và cột last\_updated.

+ dim\_sellers: Chuẩn hóa dữ liệu người bán, tạo khóa thay thế (seller\_key) và cột last\_updated.

+ dim\_geolocation: Loại bỏ trùng lặp dựa trên mã bưu điện, tạo khóa thay thế (geolocation\_key).

+ dim\_dates: Tạo bảng ngày từ 2016 đến 2025 với các thuộc tính thời gian.

+ dim\_payments: Chuẩn hóa dữ liệu thanh toán, tạo khóa thay thế (payment\_key).

B3. Tải Dữ Liệu Vào Bảng Fact

- Sử dụng get\_data\_to\_pd từ PostgresOperators để đọc dữ liệu từ các bảng staging (stg\_orders, stg\_order\_items, stg\_payments, stg\_customers).

- Kết hợp dữ liệu bằng pandas.merge để xây dựng bảng fact\_orders.

- Thực hiện các phép tính để tạo chỉ số kinh doanh:

+ total\_amount: Tổng giá trị đơn hàng (giá sản phẩm cộng phí vận chuyển).

+ delivery\_time: Thời gian giao hàng thực tế (tính bằng ngày).

+ estimated\_delivery\_time: Thời gian giao hàng dự kiến.

- Gắn các khóa ngoại (customer\_key, product\_key, seller\_key, geolocation\_key, payment\_key, order\_date\_key) để liên kết với các bảng dimension.

- Sử dụng save\_data\_to\_postgres từ PostgresOperators để lưu bảng fact\_orders vào schema warehouse.

B4. Quản Lý và Tự Động Hóa Quy Trình

- Sử dụng Apache Airflow để quản lý và tự động hóa quy trình ETL thông qua DAG e\_commerce\_dw\_etl.

- Quy trình được chia thành ba nhóm task:

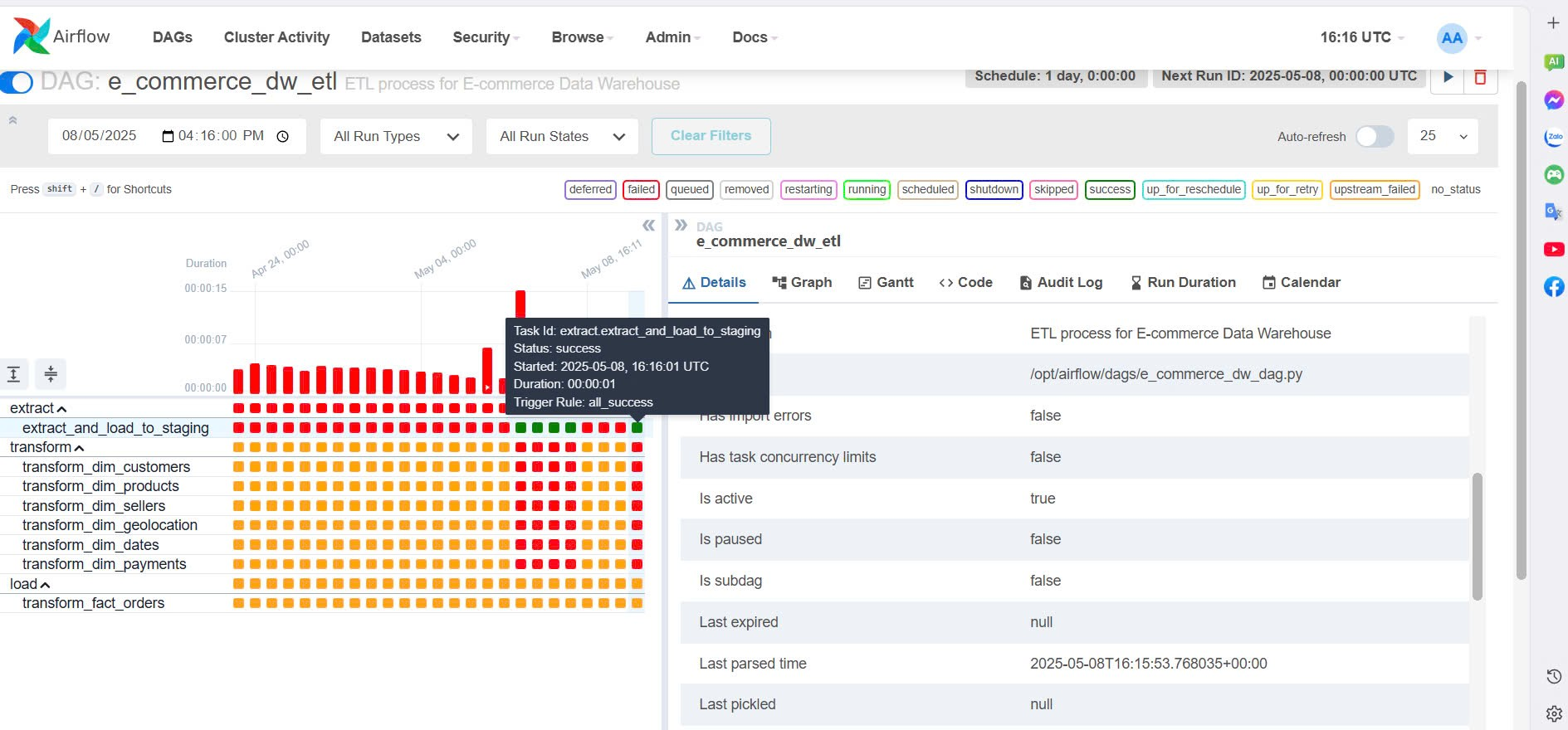
+ Extract: Trích xuất dữ liệu từ MySQL vào staging

+ Transform: Biến đổi dữ liệu để tạo các bảng dimension.

+ Load: Tải dữ liệu vào bảng fact.

- Airflow lập lịch chạy hàng ngày, đảm bảo dữ liệu trong kho luôn được cập nhật.

2.Kết quả: Thành công trích xuất dữ liệu từ MySQL và lưu vào các bảng staging trong PostgreSQL nhưng không biến đổi và tải dữ liệu được



Bài 2: Tự động cào dữ liệu và train model:

1. Các bước thực hiện

B1. Định nghĩa các hàm xử lý

Crawl data:

* Lấy to\_date từ op\_kwargs, mặc định from\_date = "2000-01-01".
* Tải JSON giá cổ phiếu từ GitHub, parse ra DataFrame pandas.
* Ghi ra file CSV tại /opt/airflow/dags/stock\_price.csv.

Train\_model:

* Đọc CSV vừa tạo, lấy cột giá (column thứ 5:6).
* Scale dữ liệu về [0,1] với MinMaxScaler.
* Tạo tập train với 60 time-steps → 1 bước dự đoán.
* Xây dựng và train mạng LSTM 4 layer (mỗi layer 50 units + Dropout 0.2).
* Lưu model ra /opt/airflow/dags/stockmodel.h5.

Gửi email():

* Đọc file CSV, encode Base64.
* Khởi tạo email qua SendGrid với nội dung HTML + logo.
* Đính kèm file CSV, gửi tới địa chỉ đích.

B2. Khởi tạo DAG

* ID: miai\_dag.
* Chạy hàng ngày (schedule\_interval=timedelta(days=1)), bắt đầu từ hôm qua.
* Default args: gửi email khi thất bại, retry 1 lần sau 5 phút.

B3. Tạo các task PythonOperator

* crawl\_data: gọi craw\_stock\_price, truyền to\_date="{{ ds }}".
* train\_model: gọi train\_model.
* email\_operator: gọi email.

B4. Đặt thứ tự thực thi

* crawl\_data → train\_model → email\_operator.

1. Kết quả: Thành công cào dữ liệu và train model nhưng không gửi được mail do không có SenGrid API

