ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA VẬT LÝ

----֎----



NGUYỄN MỸ ANH - 21002187

**BÁO CÁO BÀI TẬP LỚN CUỐI KỲ**

**MÔN HỌC THỰC TẬP TIN HỌC ỨNG DỤNG**

Kỹ thuật điện tử và tin học

**HÀ NỘI – 2024**

**Overview**

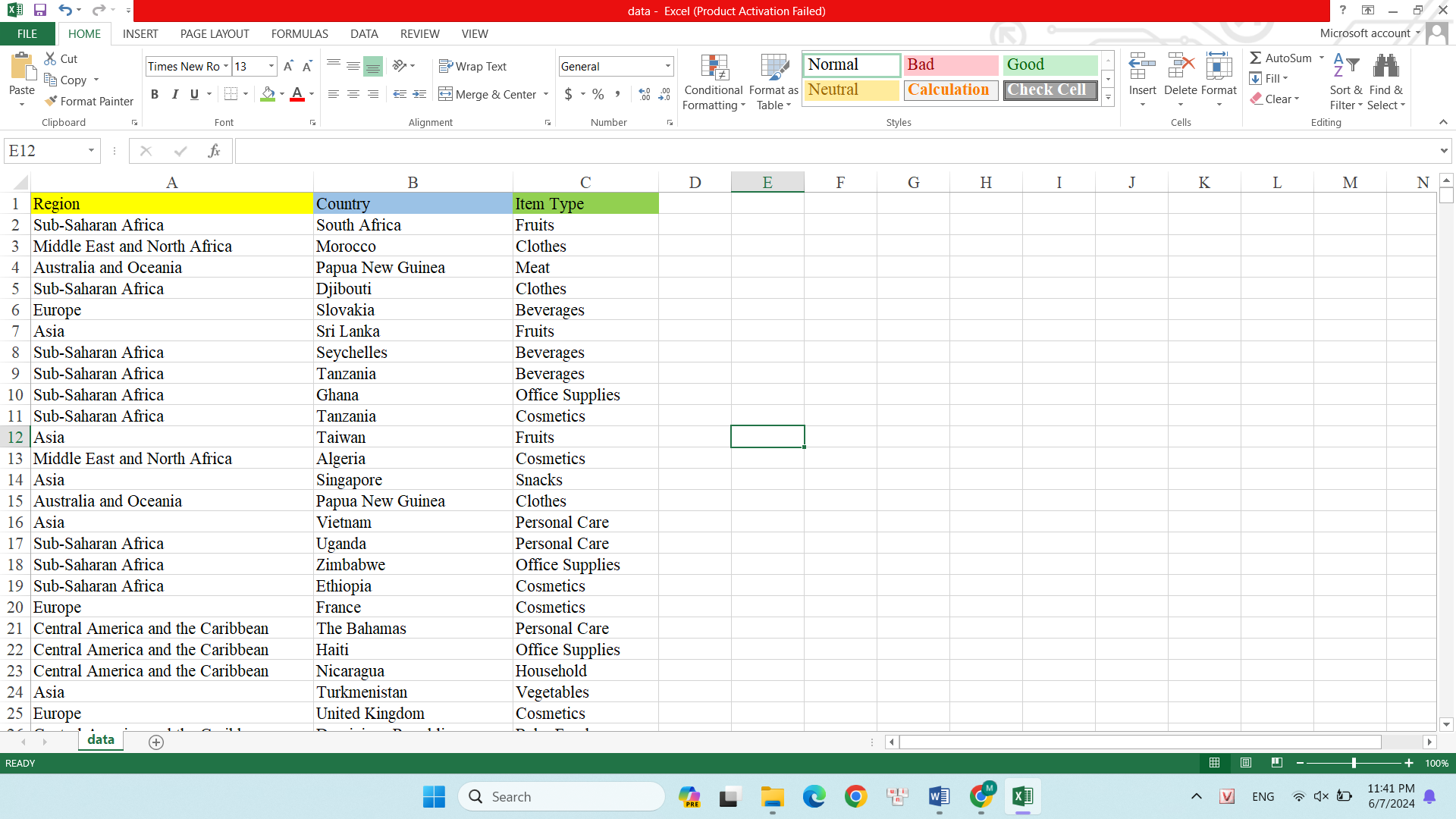
Báo cáo này cung cấp hướng dẫn chi tiết về cách thiết lập Docker container để chạy Apache Spark và PostgreSQL, kết nối SparkSQL với cơ sở dữ liệu PostgreSQL, và thực hiện các thao tác SQL bằng script Python.

**Quy trình cài đặt**

**Bước 1: Chuẩn bị các tập tin**

Đảm bảo các tập tin sau đây có trong cùng một thư mục:

1. Dockerfile: Xây dựng image Docker.
2. entrypoint.sh: Script khởi tạo PostgreSQL, tạo cơ sở dữ liệu, bảng và thiết lập Spark.
3. spark\_sql.py: Script Python để tương tác với PostgreSQL bằng SparkSQL.
4. data.csv: Tập dữ liệu mẫu có 1 triệu dòng để nhập vào PostgreSQL.

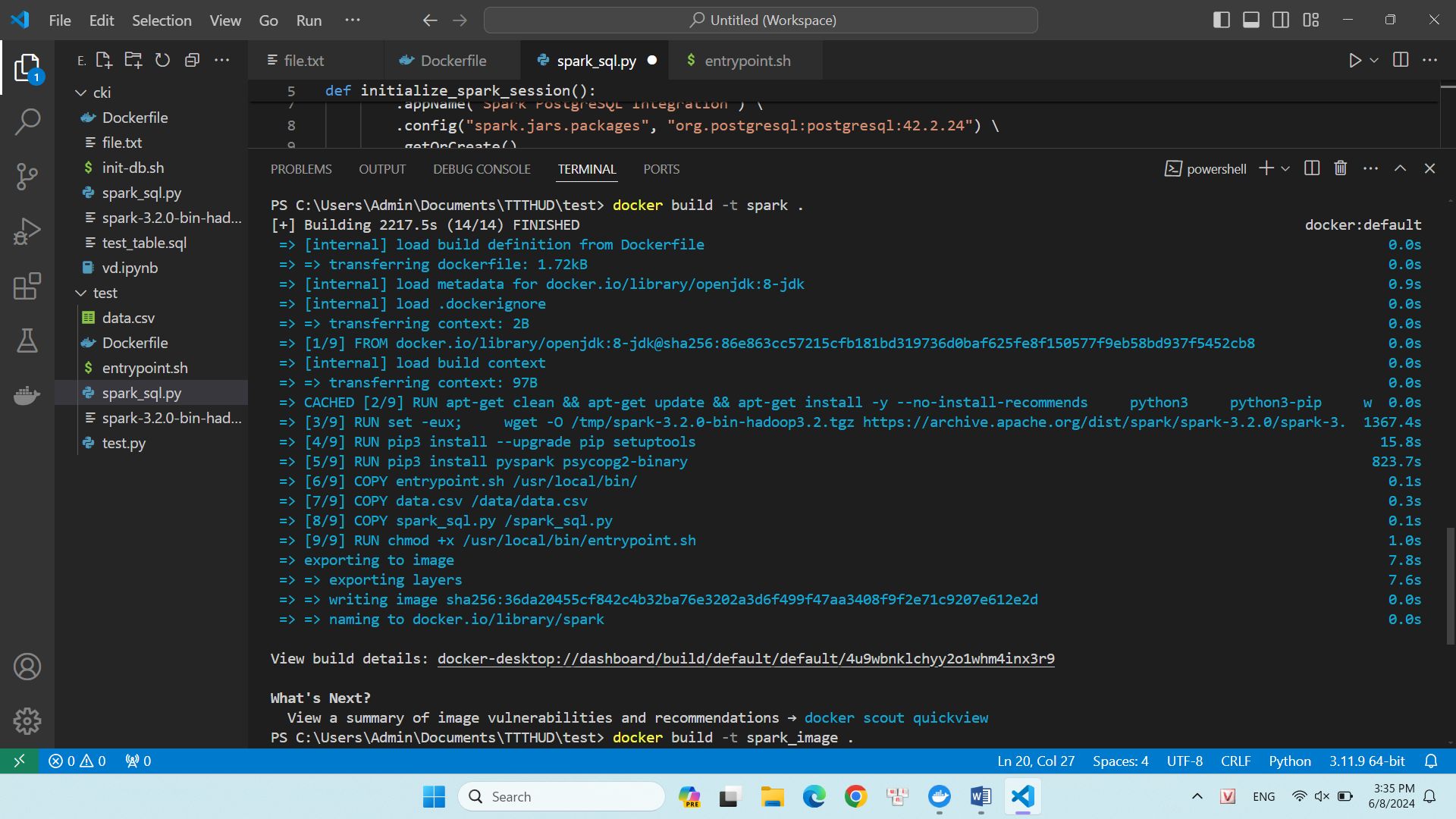


Tập dữ liệu data.csv

**Bước 2: Xây dựng Docker image**

Mở terminal trong thư mục chứa các tập tin trên và chạy lệnh sau:

**docker build -t spark .**

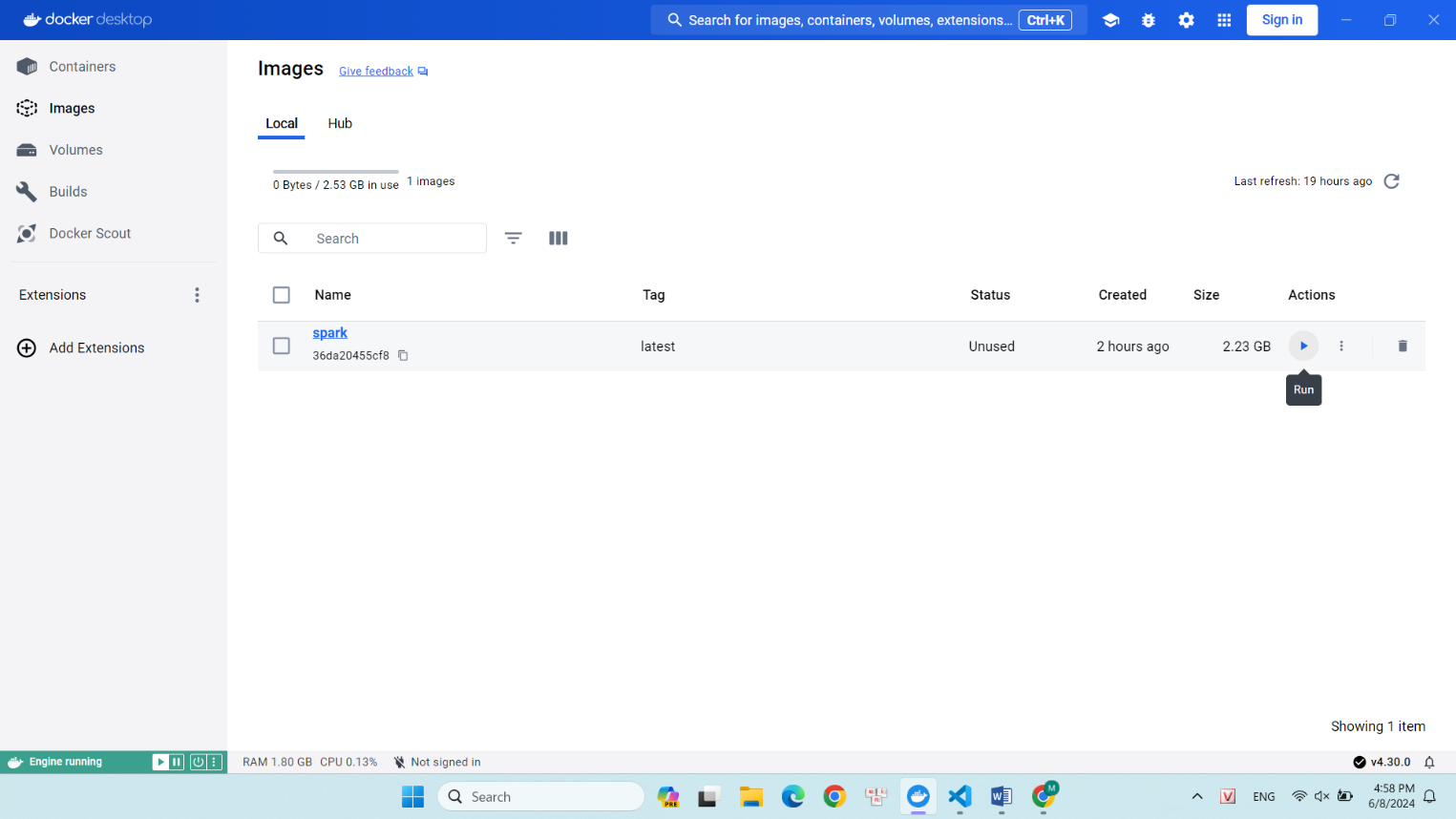


**Bước 3: Chạy Docker Container**

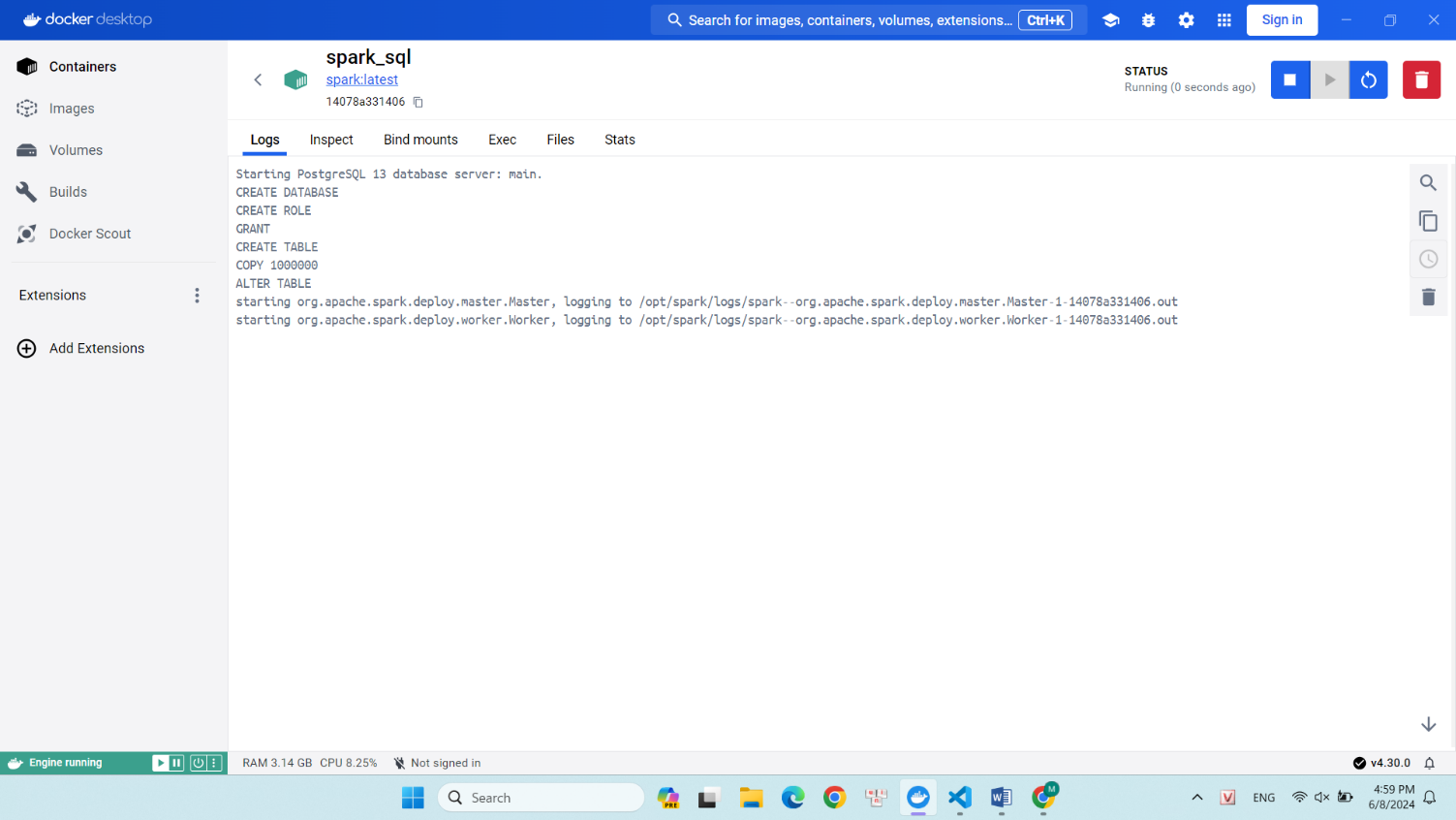
* Sau khi Docker image được xây dựng thành công, chạy container với lệnh sau:

**docker run --name spark\_sql -d spark**

Hoặc vào Docker desktop, tìm image vừa xây dựng và nhấn nút Run



Container sẽ tự động thực thi tập lệnh entrypoint.sh, thiết lập PostgreSQL và khởi tạo Spark. Kiểm tra Logs xem có lỗi không:

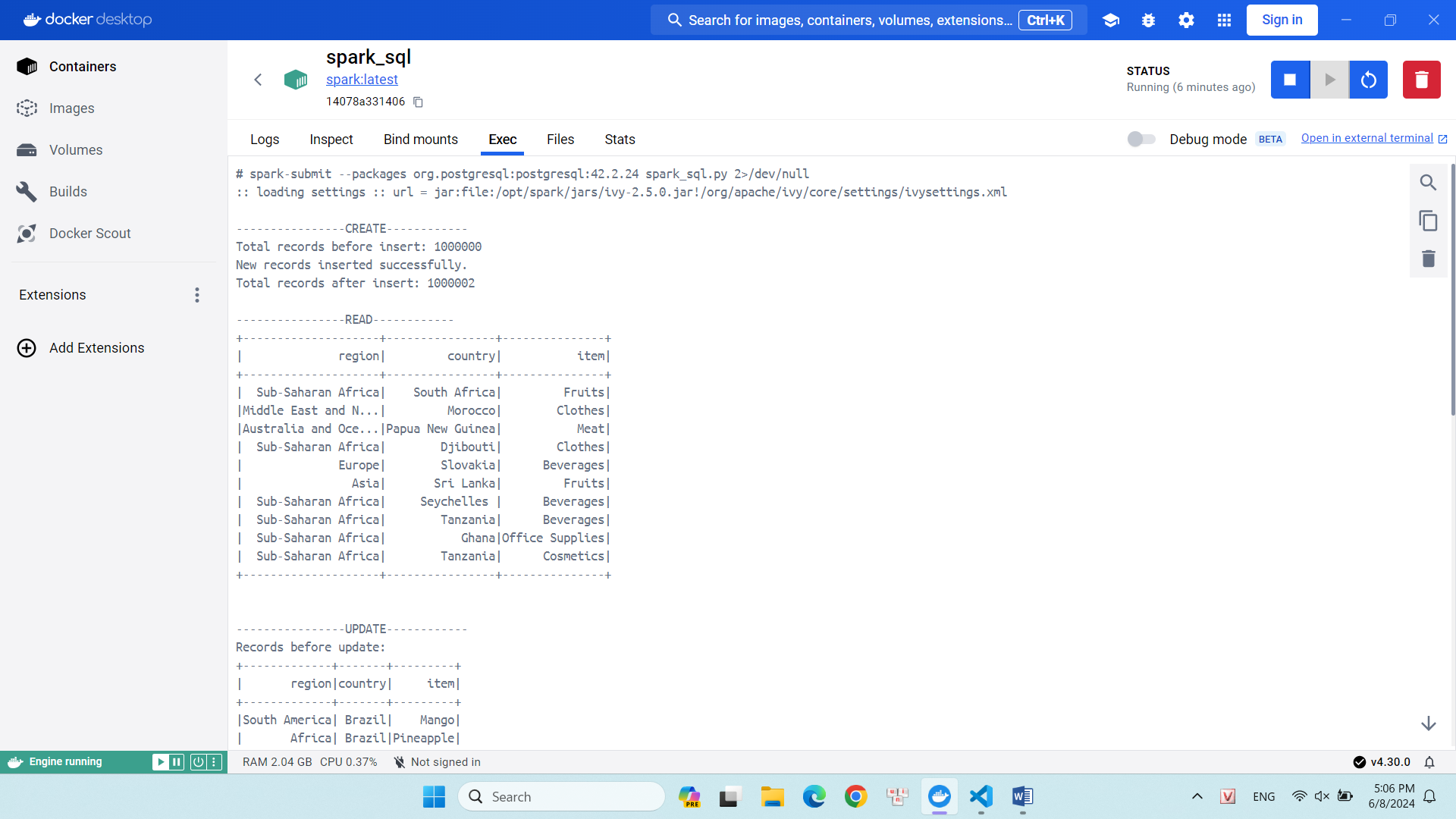


**Bước 4: Kết nối SparkSQL với cơ sở dữ liệu**

1. **Chạy script Python**

* Để chạy script spark\_sql.py, chạy lệnh sau trong Exec của container vừa tạo:

**spark-submit --packages org.postgresql:postgresql:42.2.24 spark\_sql.py 2>/dev/null**



* Bên trong container, SparkSQL đã được cấu hình sẵn để kết nối với cơ sở dữ liệu PostgreSQL sparkdb.
* Các thuộc tính kết nối được thiết lập trong spark\_sql.py:

JDBC URL: jdbc:postgresql://localhost:5432/sparkdb

Tên người dùng: sparkuser

Mật khẩu: sparkpassword

1. **Kết quả:**

Script sẽ thực hiện các thao tác CRUD (Create, Read, Update, Delete) và đo lường hiệu suất, hiển thị kết quả trên Exec.

* **Create**:

Câu lệnh này chèn dữ liệu mới từ DataFrame new\_data\_df vào bảng sample\_data trong cơ sở dữ liệu PostgreSQL. Sử dụng phương thức write.jdbc() của Spark DataFrame để ghi dữ liệu vào cơ sở dữ liệu, với chế độ "append" để thêm dữ liệu mới vào bảng.

# Creating a DataFrame to insert

data = [("South America", "Brazil", "Mango"),

            ("Africa", "Brazil", "Pineapple")]

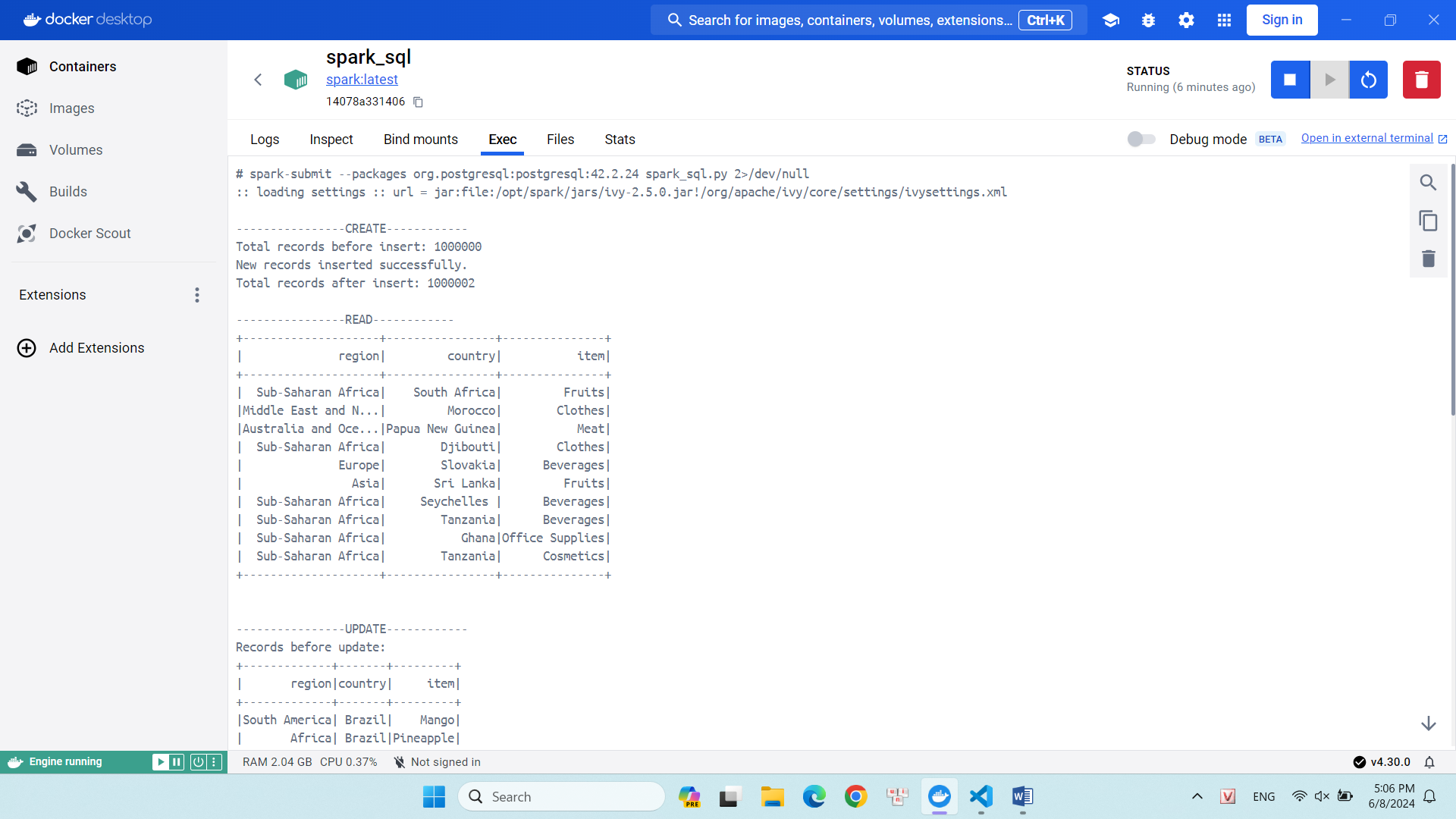
columns = ["region", "country", "item"]

new\_data\_df = spark.createDataFrame(data, columns)

# Appending new data to PostgreSQL

new\_data\_df.write.jdbc(url=jdbc\_url, table="sample\_data", mode="append", properties=properties)

Kết quả: Thêm 2 dòng dữ liệu mới, in ra tổng số dòng trước và sau khi thêm

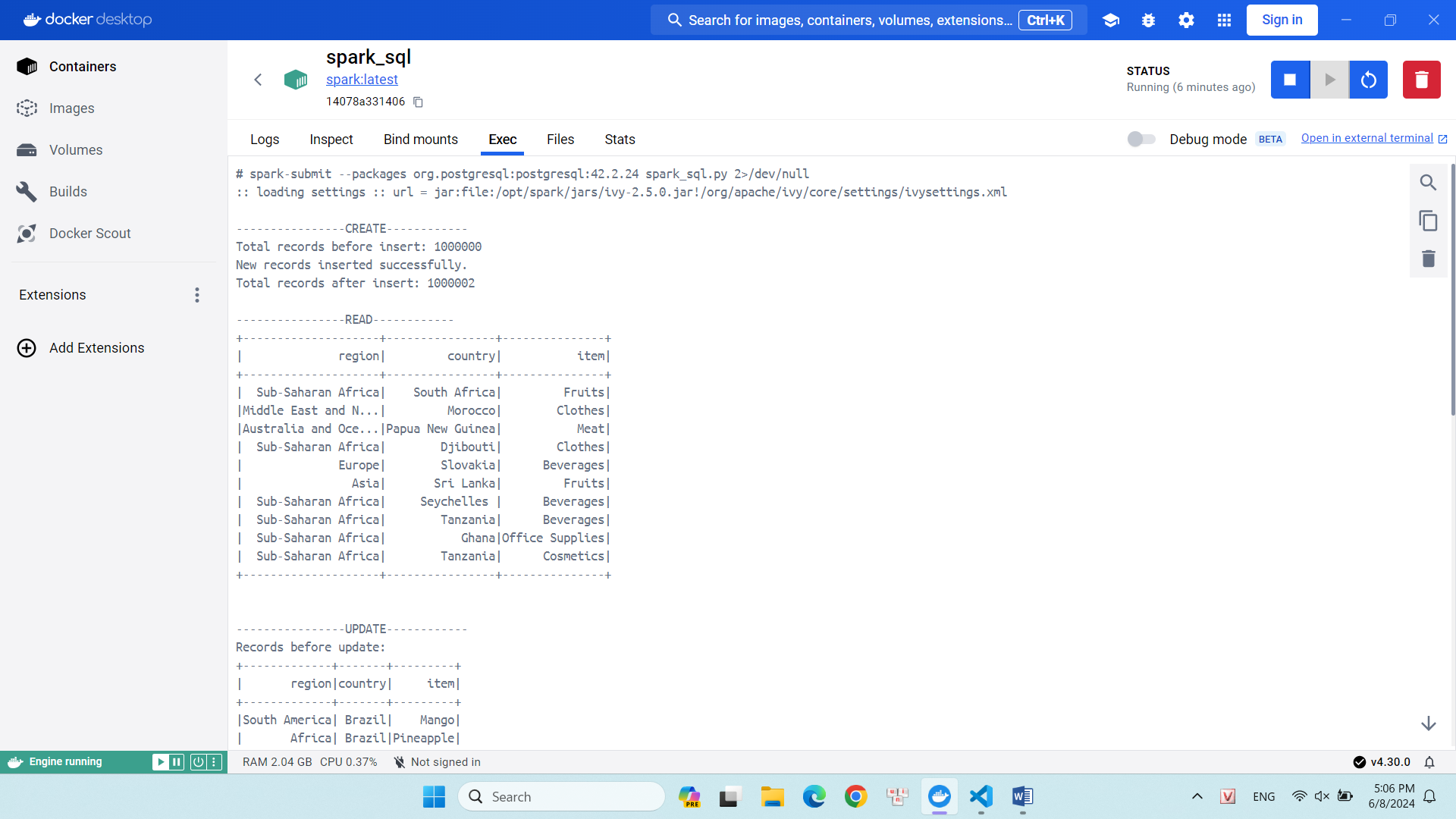


* **Read:**

Câu lệnh này sử dụng Spark SQL để chọn tất cả các cột và hàng từ bảng sample\_data và hiển thị 10 hàng đầu tiên. Nó thực hiện thao tác đọc dữ liệu từ cơ sở dữ liệu PostgreSQL và trả về một DataFrame Spark, sau đó sử dụng phương thức show() để hiển thị dữ liệu.

spark.sql("SELECT \* FROM sample\_data LIMIT 10").show()

Kết quả:



* **Update:**

Câu lệnh này cập nhật các dòng trong bảng sample\_data trong cơ sở dữ liệu PostgreSQL. Nó sử dụng câu lệnh SQL UPDATE để thay đổi giá trị của cột item thành 'Papaya' cho tất cả các dòng có giá trị country là 'Brazil'.

# Perform update

update\_query = """

        UPDATE sample\_data

        SET item = 'Papaya'

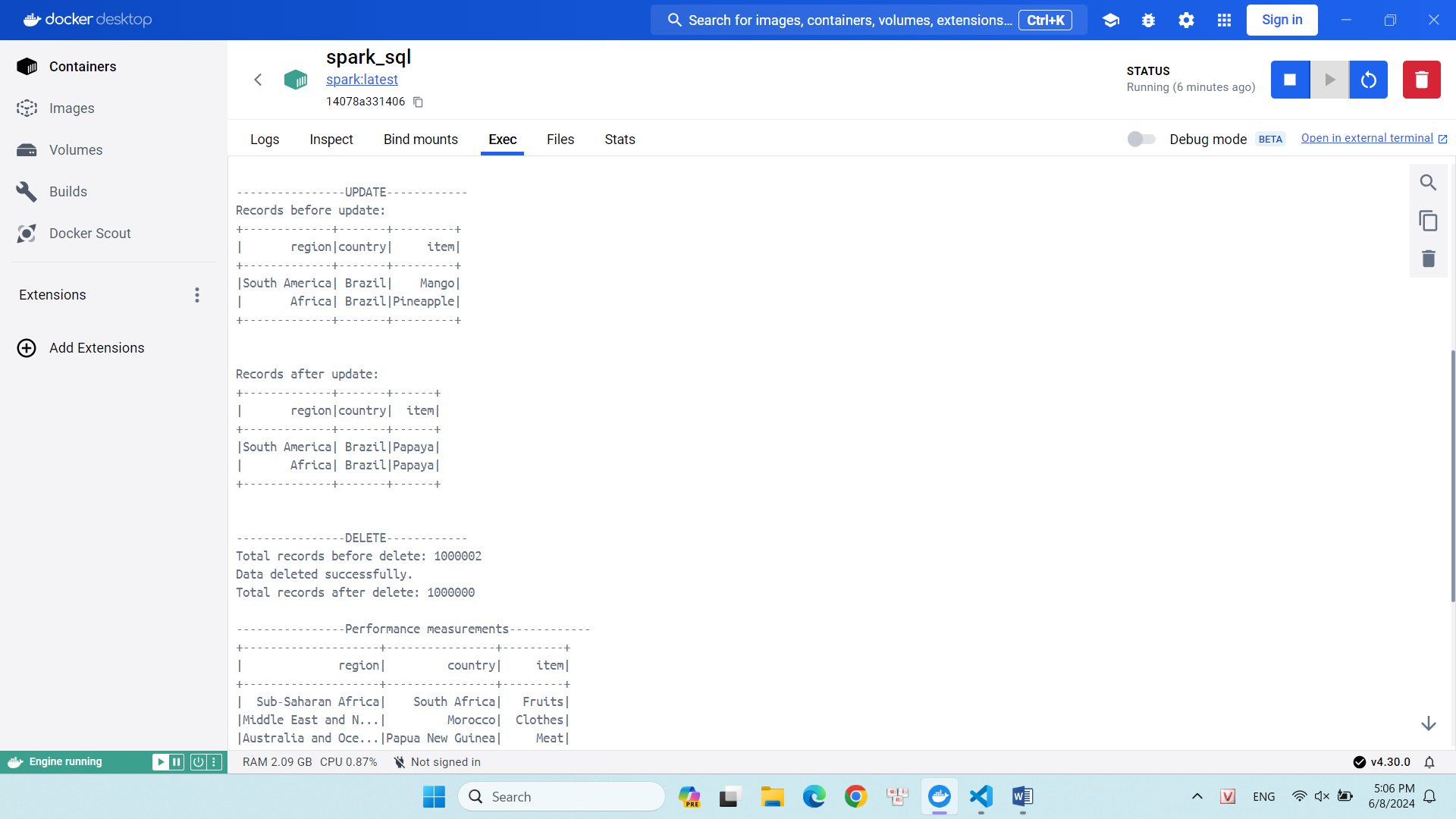
        WHERE country = 'Brazil'

"""

cursor.execute(update\_query)

conn.commit()

Kết quả: Hiện thị các dòng có giá trị country là 'Brazil' trước và sau khi cập nhật



* **Delete**: Câu lệnh này xóa các dòng từ bảng sample\_data trong cơ sở dữ liệu PostgreSQL. Nó sử dụng câu lệnh SQL DELETE để xóa tất cả các bản ghi có giá trị country là 'Brazil' khỏi bảng. Ở đây xóa 2 dòng dữ liệu vừa mới thêm ở bước Create vì trong tập dữ liệu mẫu data.csv không có dòng nào có giá trị country là Brazil

delete\_query = """

        DELETE FROM sample\_data

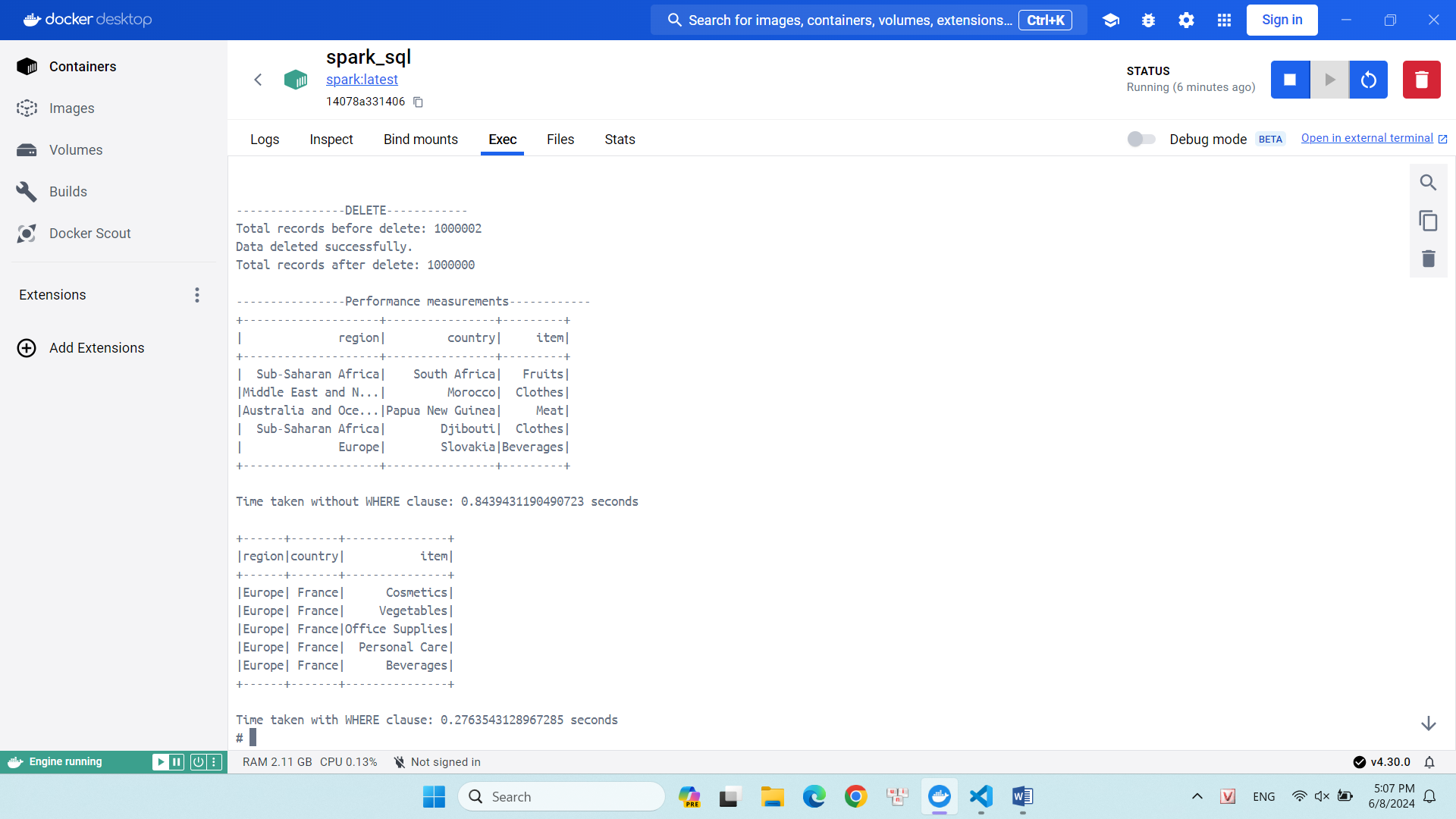
        WHERE country = 'Brazil'

"""

cursor.execute(delete\_query)

conn.commit()

Kết quả: Xóa và in ra tổng số dòng trước và sau khi xóa



* **Đánh giá hiệu suất**:

Các câu lệnh này được sử dụng để đo thời gian xử lý của các truy vấn SQL trong Spark SQL. Câu lệnh thứ nhất trả về 5 hàng đầu tiên từ bảng sample\_data, trong khi câu lệnh thứ hai chỉ trả về 5 hàng đầu tiên từ bảng sample\_data mà country là 'France'. Việc đo thời gian này giúp đánh giá hiệu suất của các truy vấn có và không có điều kiện WHERE. Dùng hàm time trong python để tính khoảng thời gian thực thi của các truy vấn

# Measure time for query without WHERE clause

spark.sql("SELECT \* FROM sample\_data LIMIT 5").show()

# Measure time for query with WHERE clause

spark.sql("SELECT \* FROM sample\_data WHERE country = 'France' LIMIT 5").show()

Kết quả: Việc sử dụng điều kiện WHERE trong câu lệnh SQL thường mang lại hiệu suất cao hơn bởi vì nó giảm bớt khối lượng dữ liệu cần xử lý, tối ưu hóa quy trình xử lý dữ liệu và tận dụng các cơ chế tối ưu hóa của hệ thống cơ sở dữ liệu.

