PROJECT KẾT THÚC MÔN HỌC

THỰC TẬP TIN HỌC ỨNG DỤNG

Họ và tên: Trần Ngọc Phúc

Lớp: K66 Kỹ thuật điện tử và tin học

MSV: 21000431

Trong project kết thúc môn học thực tập tin học ứng dụng, em thực hiện việc cấu hình và chạy SparkSQL trên Docker container, cùng với việc tải xuống một số phần mềm, thư viện cần thiết trên container để truy vấn SQL trên cơ sở dữ liệu được lưu trữ trên container.

1. Docker setup: Tạo 1 docker container

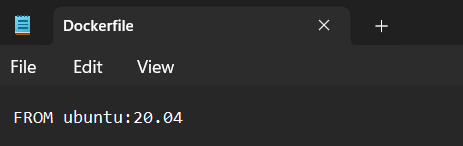
* Tải xuống docker desktop trên HĐH Windows
* Tạo một vùng làm việc mới. Ở đây ta tạo một thư mục tên là DockerProject và truy cập vào thư mục đã tạo:

$ mkdir DockerProject

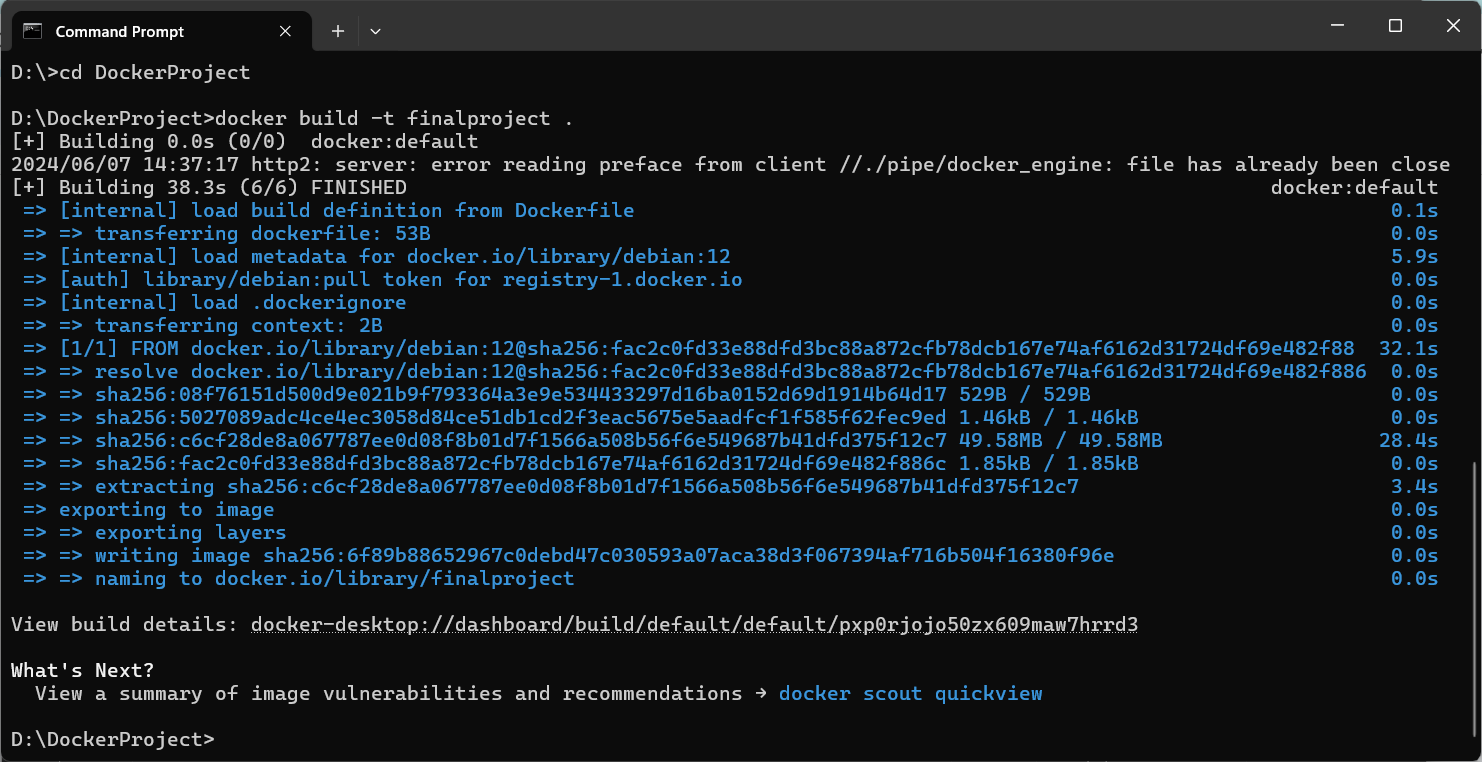
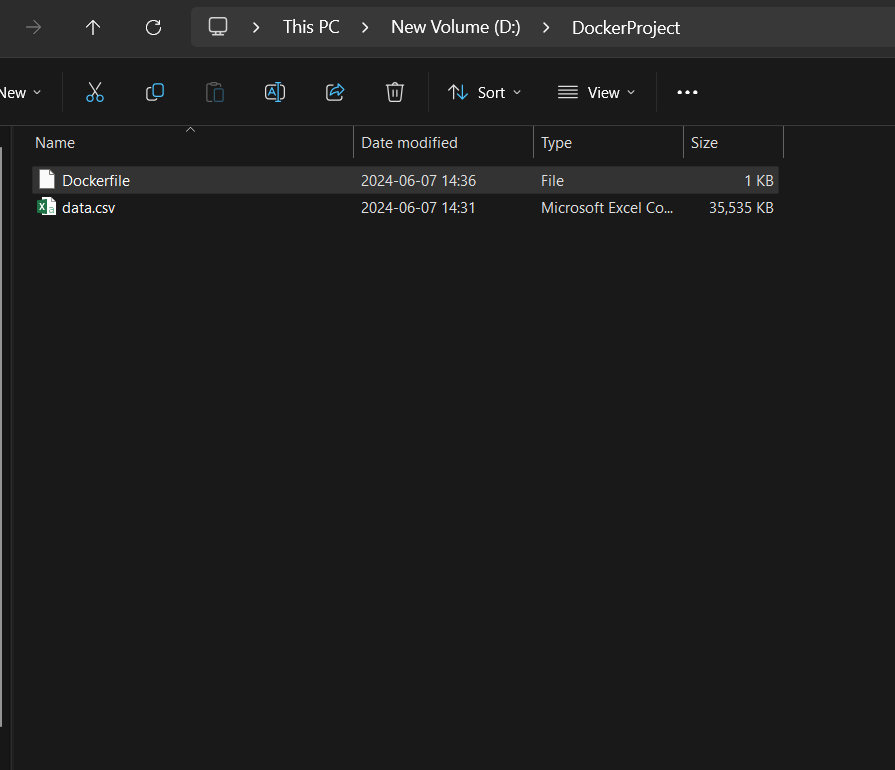
$ cd DockerProject

* Tạo 1 file với tên là Dockerfile và không có hậu tố:

$ touch Dockerfile

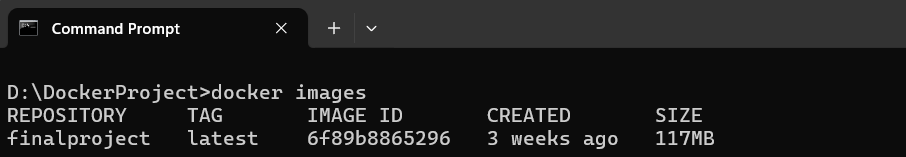
* Thực hiện ghi file Dockerfile đã tạo:

Ảnh 1: Cấu hình dockerfile

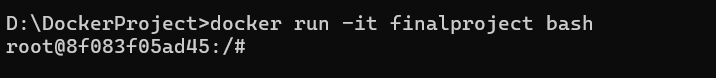
* Lưu lại file và thoát khỏi trình ghi file. Sau đó vào thư mục DockerProject và thực hiện build docker image bằng lệnh: “docker build -t finalproject .”.

Ảnh 3: Kết quả khi build image

Ảnh 2: Thư mục lưu trữ



Ảnh 4: Kiểm tra xem đã có image hay không

* Sau khi build image, tiếp tục đến bước tạo Container. Nhập lệnh docker run -it finalproject bash để chạy container.

Ảnh 5: Chạy container

1. Cài đặt Spark.

* Trước tiên Spark là một phần mềm mã nguồn mở miễn phí, nó tương thích với mọi hệ điều hành. Với image đã tạo ở trên có image base là debian phù hợp với việc cài đặt Spark và các công cụ cần thiết khác.
* Đầu tiên để chạy Spark trên container đã tạo, ta thực hiện việc cài đặt một số công cụ cần thiết. Sử dụng file Dockerfile ở phần 1và nhập các lệnh sau:

ENV DEBIAN\_FRONTEND=noninteractive

RUN apt-get update \

&& apt-get install -y curl tar wget vim \

python3 python3-pip \

&& ln -s /usr/bin/python3 /usr/bin/python \

&& apt-get clean \

&& rm -rf /var/lib/apt/lists/\*

RUN apt-get update \

&& apt-get install -y default-jdk \

&& apt-get clean \

&& rm -rf /var/lib/apt/lists/\*

RUN wget -qO- "https://downloads.apache.org/spark/spark-${SPARK\_VERSION}/spark-${SPARK\_VERSION}-bin-hadoop${HADOOP\_VERSION}.tgz" | tar -xz -C /opt/

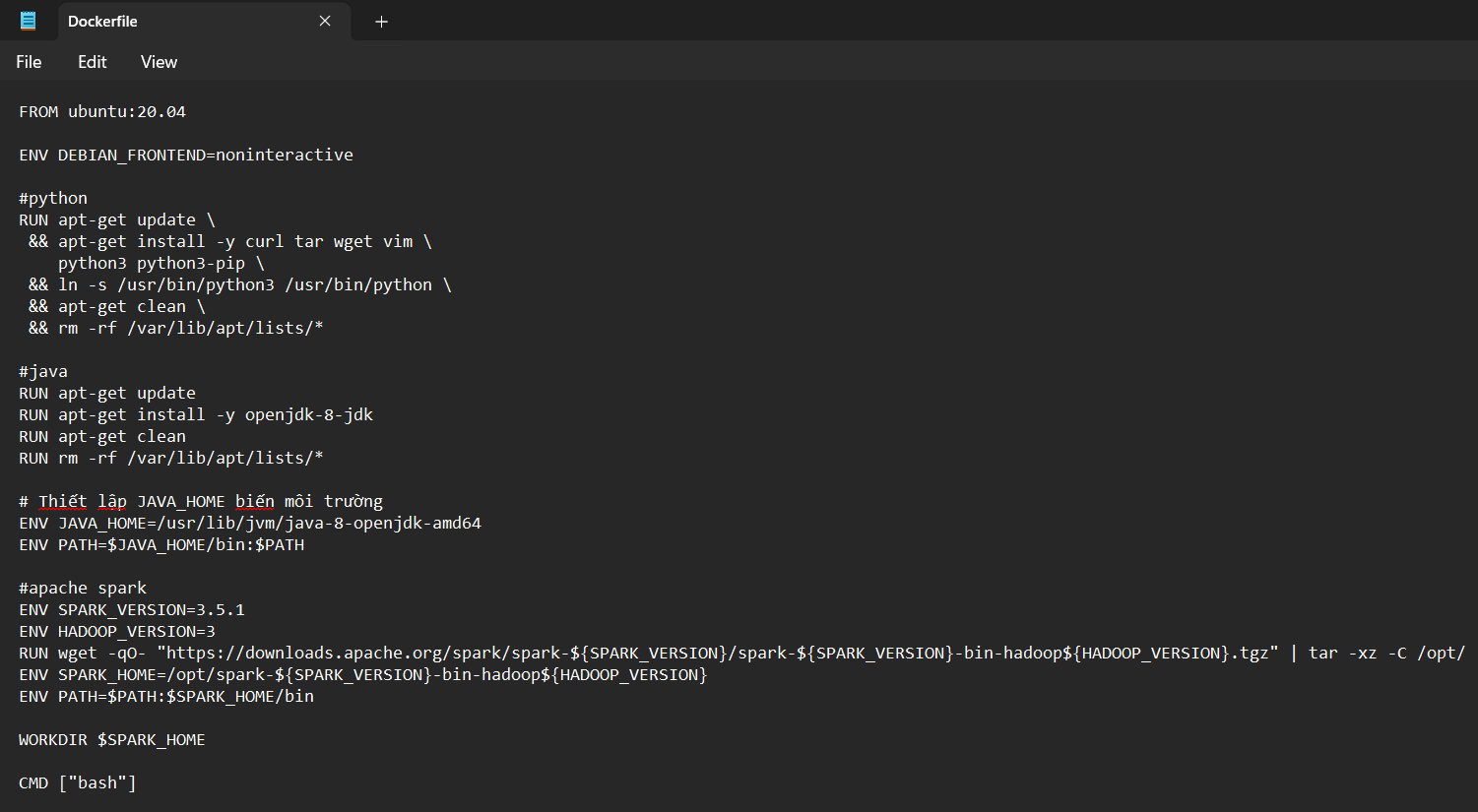
* Khai báo biến môi trường DEBIAN\_FRONTEND = noninteractive để không hỏi các câu hỏi khi cài đặt thư viện.
* Ở đây ta có các môi trường như python, java cùng với các thư viện bổ trợ như curl, tar, wget, vim, các thư viện quản lý gói thư viện như pip.
* Phiên bản Spark sử dụng trong bài này là 3.5.1 đi cùng với Hadoop phiên bản 3.0.
* Sau khi lựa chọn các gói thư viện, công cụ cần thiểt, giờ ta cấu hình đường dẫn cho Spark trong Dockerfile:

ENV JAVA\_HOME=/usr/lib/jvm/java-8-openjdk-amd64

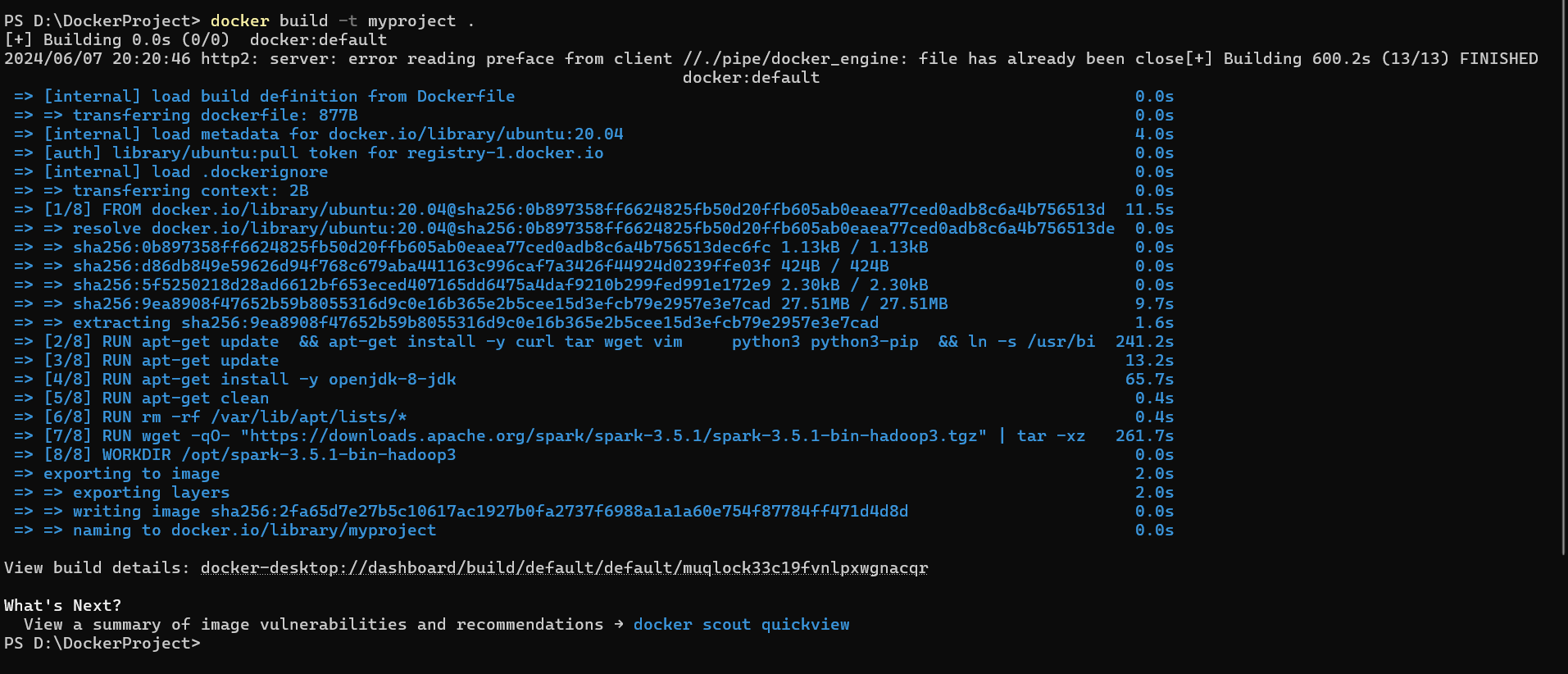
ENV PATH=$JAVA\_HOME/bin:$PATH

ENV SPARK\_HOME=/opt/spark-${SPARK\_VERSION}-bin-hadoop${HADOOP\_VERSION}

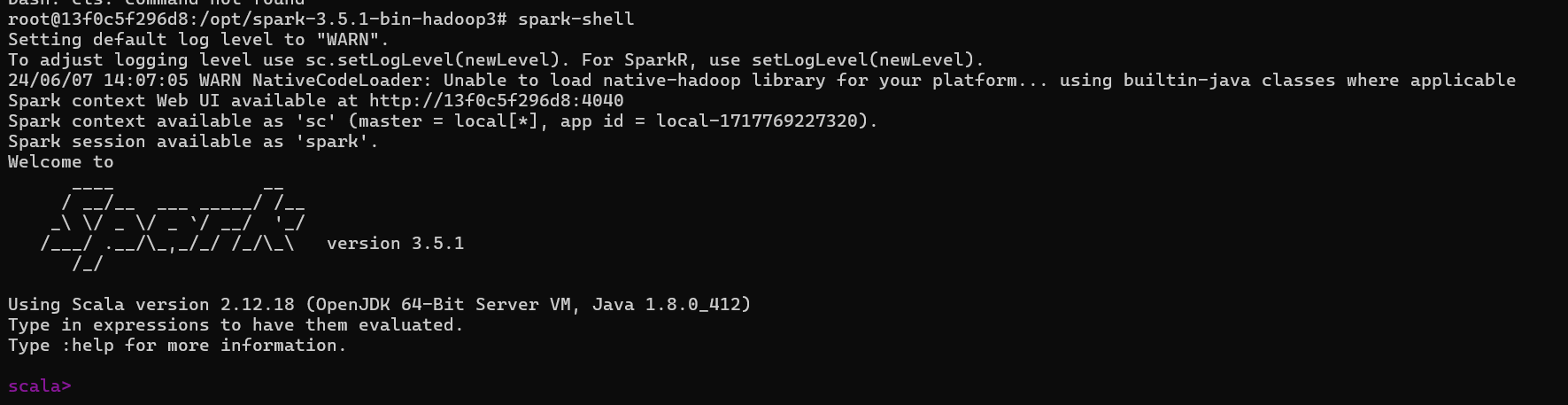
ENV PATH=$PATH:$SPARK\_HOME/bin



Ảnh 6: Cấu hình Dockerfile

* Lưu lại file Dockerfile và thực hiện build image với tên là myproject.
* Sau khi build xong image, thực hiện run container bằng việc nhập lệnh: docker run -it myproject.

Ảnh 7: Hoàn thiện việc build image

* Kiểm tra spark đã cài đặt bằng lệnh spark-shell.

Ảnh 8: Chạy container

Ảnh 9: Spark đã được cài đặt

1. Setup database

* Đầu tiên mở Dockerfile bổ sung lệnh sau để setup database:

RUN apt-get update \

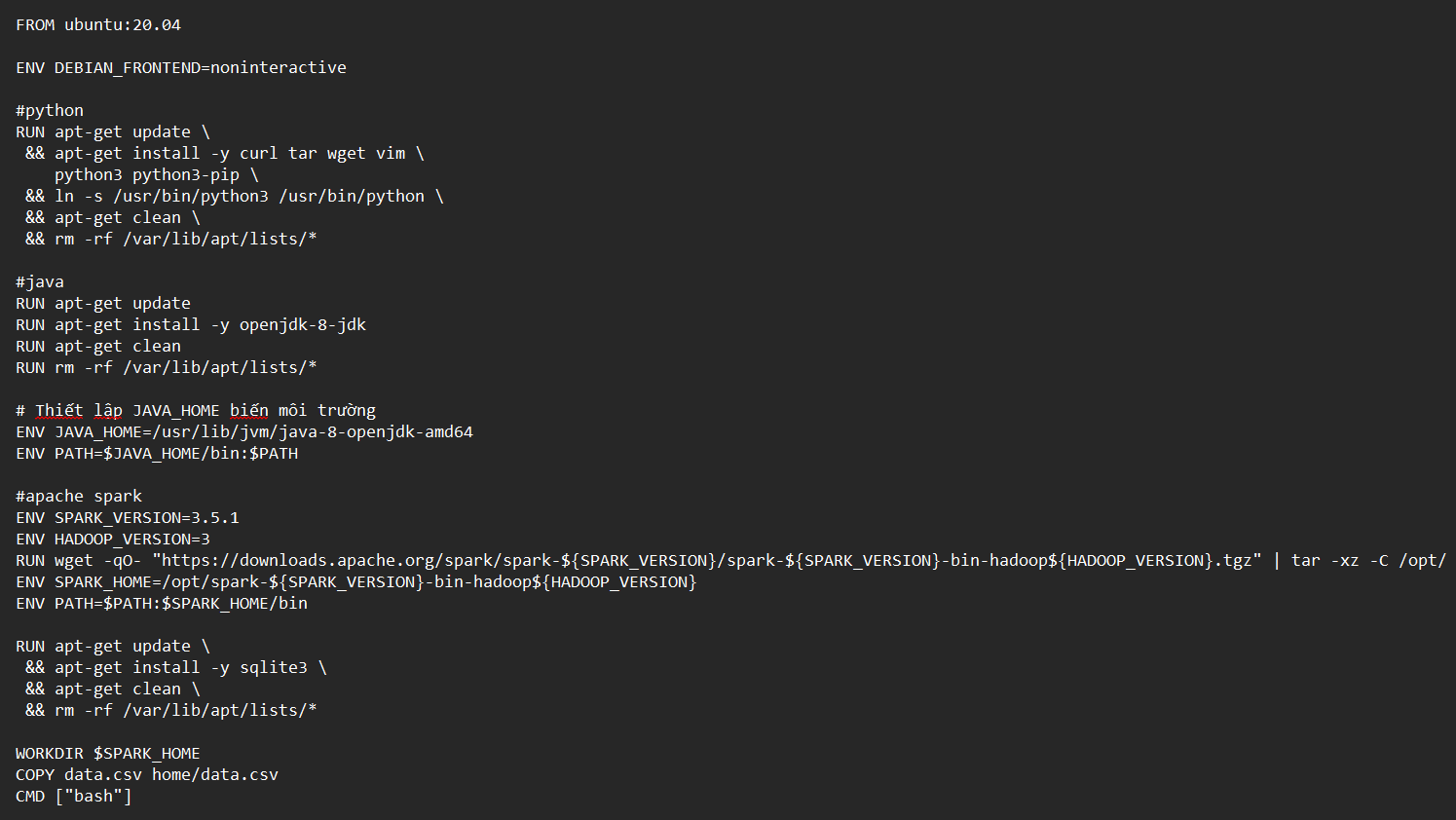
&& apt-get install -y sqlite3 \

&& apt-get clean \

&& rm -rf /var/lib/apt/lists/\*

* Tiếp đến copy file tên data.csv đã chuẩn bị trước vào thư mục home với tên là data.csv với dòng lệnh sau:

COPY data.csv home/data.csv

* Lưu lại Dockerfile và thực hiện build image với tên trước đấy là myproject.
* Sau khi build xong image, thực hiện chạy container bằng lệnh “docker run -it myproject”.

Ảnh 10: Cấu hình Dockerfile

* Sau khi chạy container, tạo và chạy cơ sở dữ liệu bằng lệnh sau: sqlite3 /home/test.db
* Giao diện dòng lệnh trong database sẽ hiện lên để tương tác với cơ sở dữ liệu. Lúc này thực hiện 2 dòng lệnh để đưa file data.csv vào bảng trong cơ sở dữ liệu:

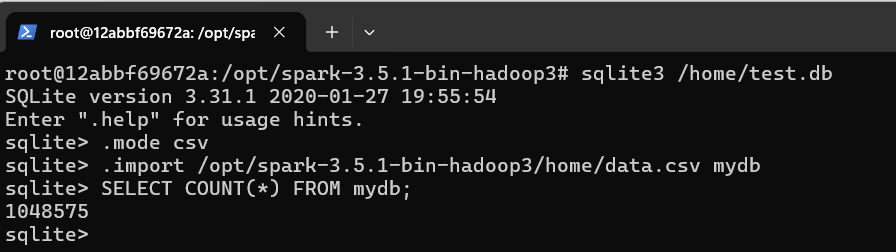
.mode csv

.import /opt/spark-3.5.1-bin-hadoop3/home/data.csv mydb

* Quá trình này thực hiện việc đưa file data.csv vào bảng có tên là mydb.
* Thực hiện kiểm tra số lượng hàng trong bảng bằng lệnh sau:

SELECT COUNT(\*) FROM mydb;

Ảnh 11: Kiểm tra số lượng hàng trong bảng

* Trong bước kiểm tra này, số lượng hàng trong bảng là hơn 1 hàng và lệnh đã thực hiện đưa file đầy đủ.

1. Dependencies:

* Ở phần này, ta cần cài thêm JDBC để kết nối với cơ sở dữ liệu và cài thêm pyspark. Để thực hiện việc trên, bổ sung thêm lệnh sau vào Dockerfile:

RUN pip install pyspark

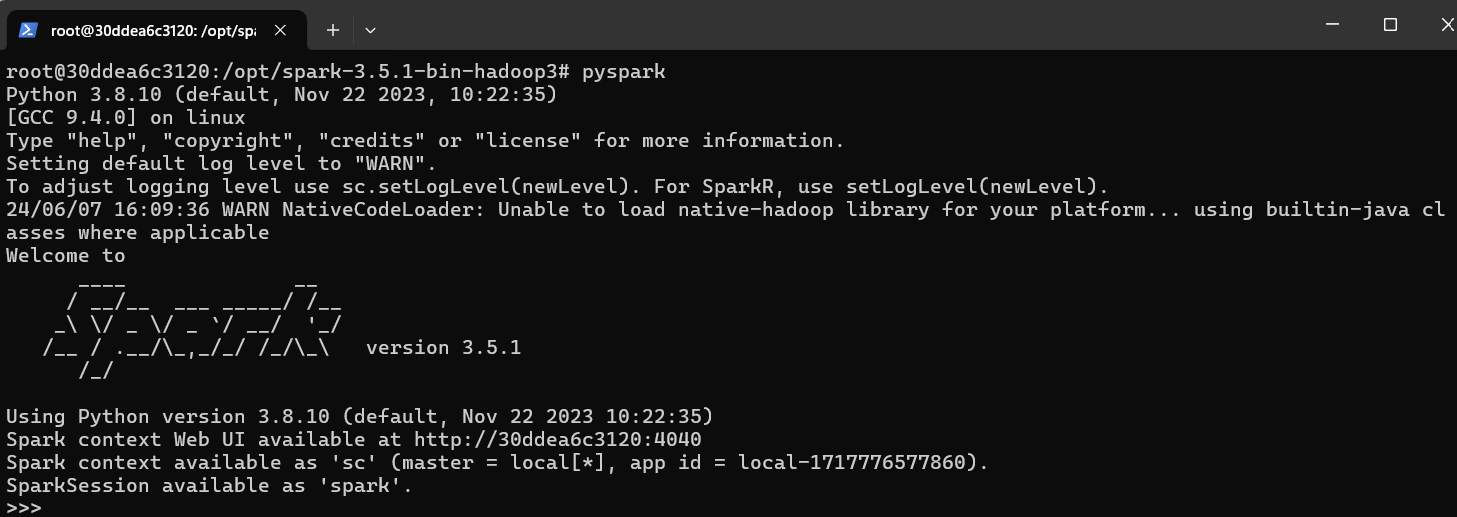
RUN wget <https://repo1.maven.org/maven2/org/xerial/sqlite-jdbc/3.34.0/sqlite-jdbc-3.34.0.jar>

* Sau đó thực hiện lại việc build image và chạy container.

1. Configuration:

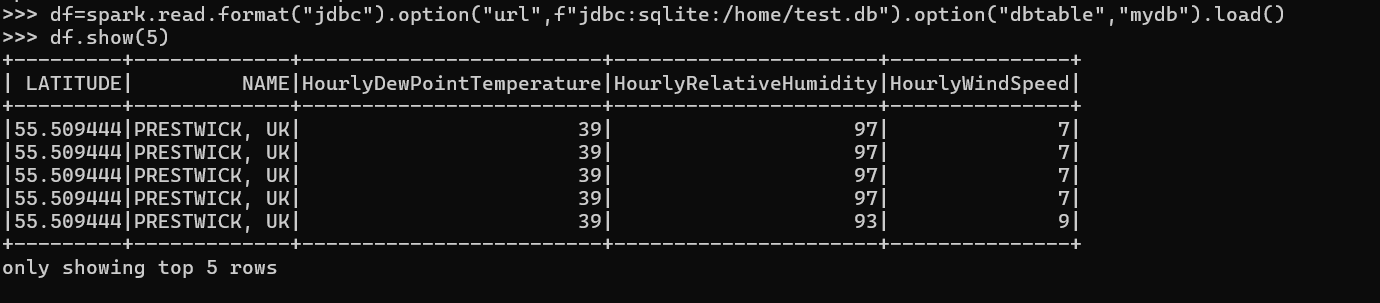
* Để Spark có thể kết nối với cơ sở dữ liệu, ta cần chuyển JDBC vào thư mục jars trong spark, thực hiện lệnh sau:

mv sqlite-jdbc-3.34.0.jar /opt/spark/jars

* Sau khi di chuyển, ta thực hiện việc kiểm tra spark có kết nối với cơ sở dữ liệu hay không.
* Sau khi chạy container, ta nhập lệnh ‘spark’ để khởi chạy. Lúc này một phiên bản spark với môi trường python, cùng với giao diện dòng lệnh bắt đầu.
* Sau đó nhập lệnh để kết nối spark tới cơ sở dữ liệu và thực hiện đọc dữ liệu bảng trong cơ sở dữ liệu:

Ảnh 12: Khởi động spark

df=spark.read.format("jdbc").option("url",f"jdbc:sqlite:/home/test.db").option("dbtable","mydb").load()

* Tiếp tục ghi lệnh “df.show(5)” để hiển thị 5 giá trị đầu tiên trong bảng.

Ảnh 13: Kết quả in ra

* Tiếp tục với việc kiểm tra SparkSQL có thể kết nối và truy vấn vào cơ sở dữ liệu như sau:

Bước 1: Trở về giao diện tương tác với container. Thực hiện cài đặt nano để chỉnh sửa văn bản bằng lệnh “apt-get install -y nano”.

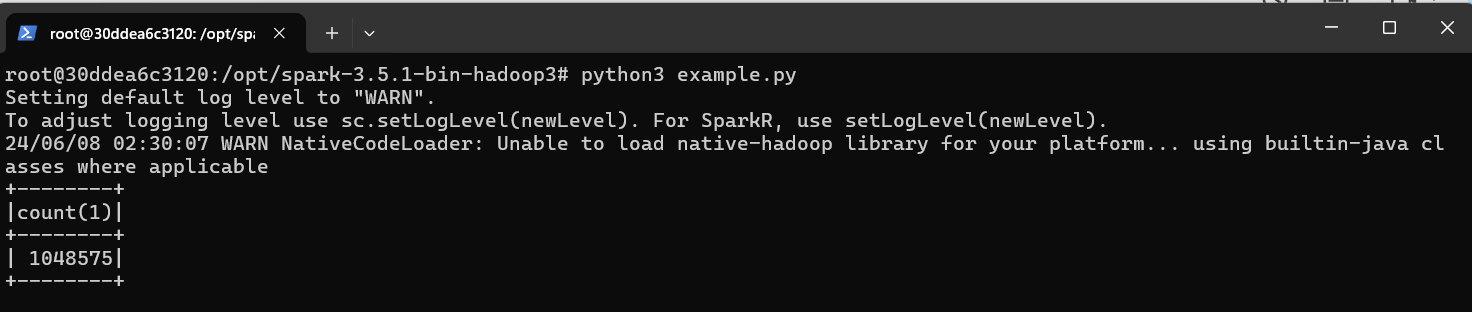
Bước 2: Tạo một file để thực hiện việc kiểm tra. Ở đây ta tạo một file có tên test.py để thử nghiệm, để tạo file ghi lệnh “touch example.py”. Sau đó ghi lệnh “nano example.py” để thực hiện ghi dòng code vào trong file .py đó.

Bước 3: Thực hiện ghi file theo như hình dưới đây:

Ảnh 14: file example.py

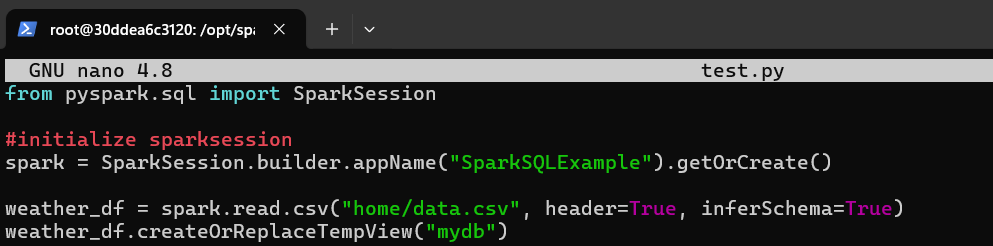
Bước 4: Lưu lại file và thoát khỏi giao diện nano. Nhiệm vụ của file example.py là thực hiện kết nối SparkSQL và thực hiện truy vấn, ở đây là thực hiện đếm số hàng trong bảng dữ liệu.

Bước 5: Thực hiện chạy file example.py bằng lệnh “python example.py” và kết quả thu được như sau:

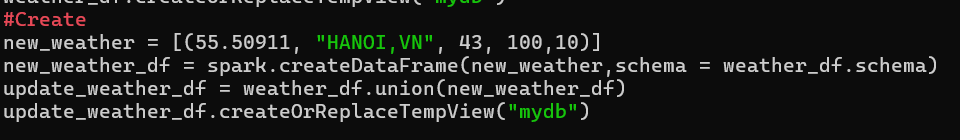
1. Running queries:

Ảnh 15: kết quả chạy file example.py

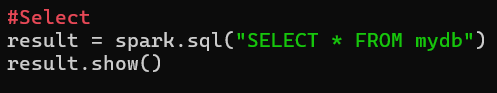
* Đầu tiên, ta sẽ viết file nội dung để thực hiện các thao tác với cơ sở dữ liệu. Tạo file test.py và thực hiện ghi dòng lệnh sau để khởi tạo Spark và đọc dữ liệu từ bảng:



Ảnh 16: Khởi tạo SparkSQL

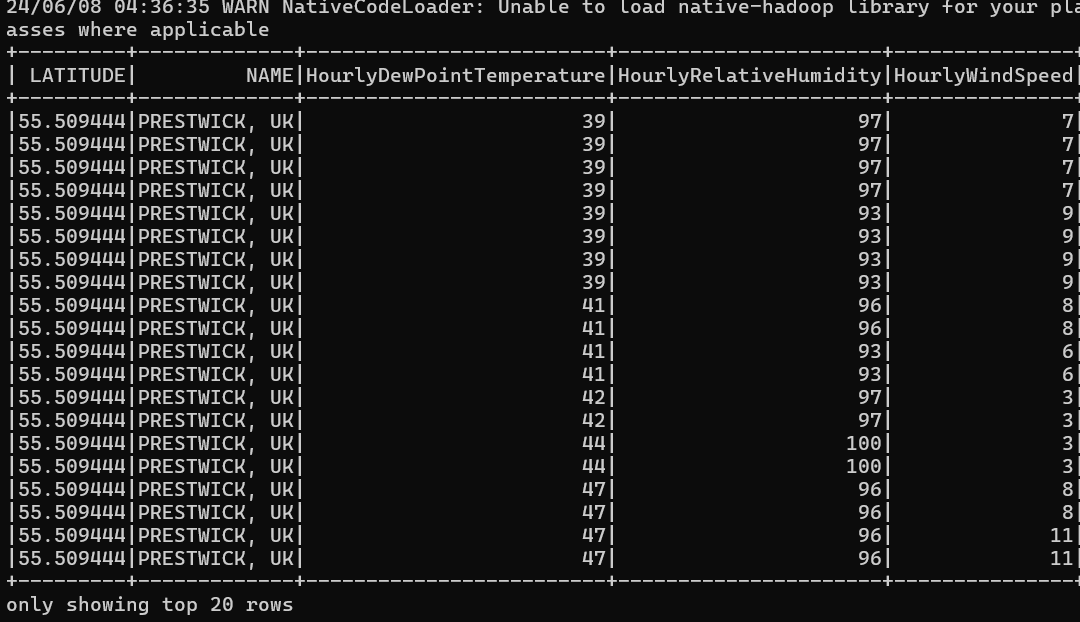
* Tiếp theo, thực hiện các thao tác cơ bản:
* Create: Ở phần này, ta thực hiện việc thêm hàng mới vào.
* Read(Select): Ở phần này, ta thực hiện chọn tất cả thành phần trong bảng và in ra.

Ảnh 17: Create operation

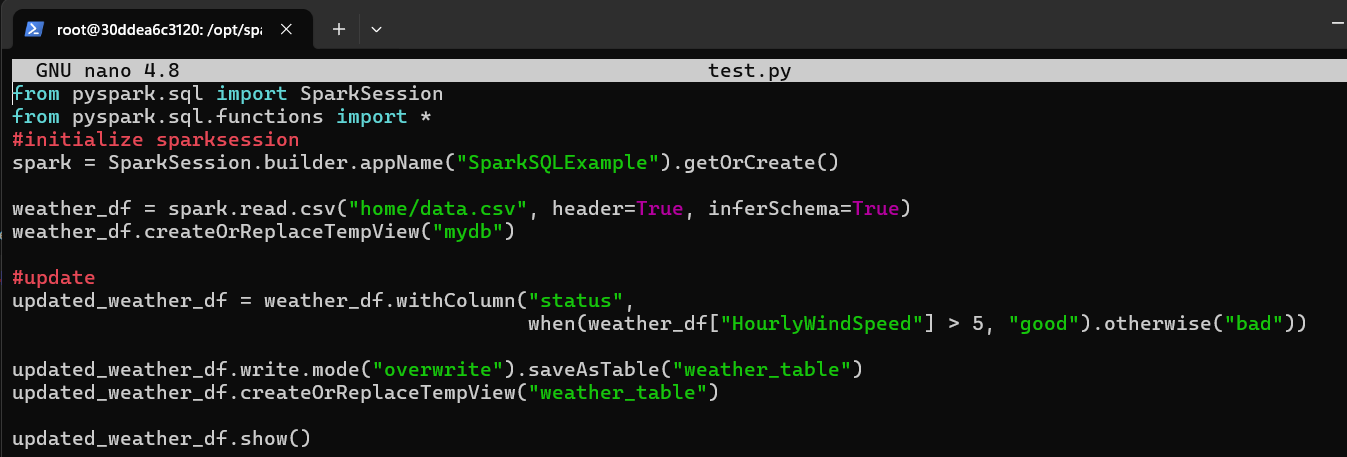


Ảnh 18: Read operation

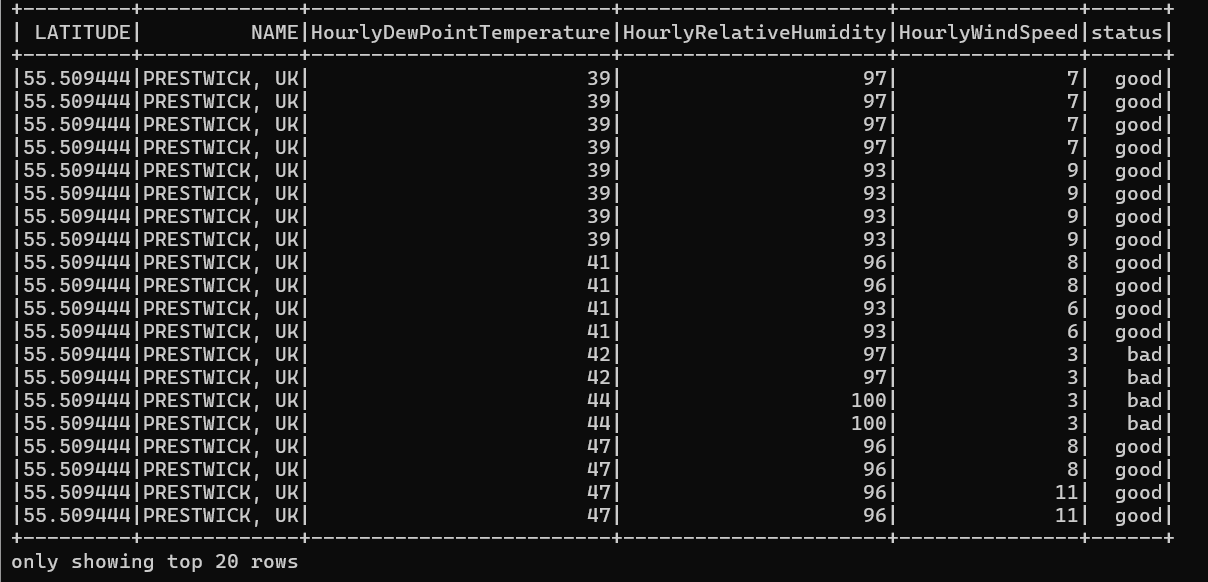
Sau khi viết xong lệnh trên, lưu lại file và thực hiện chạy file. Kết quả nhận được là:

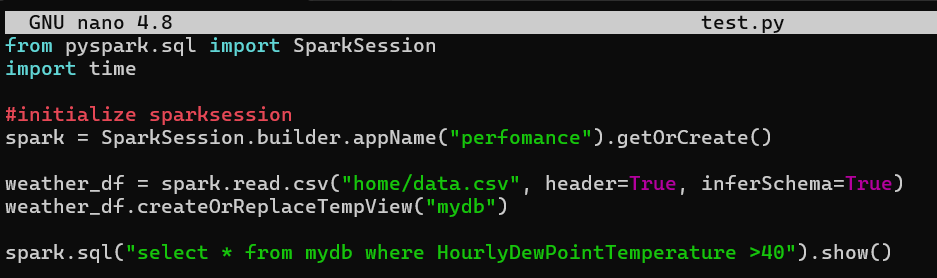
* Update: Ở phần này, ta thêm cột “status” và thực hiện update cột mới, file thực hiện được ghi dưới đây:

Ảnh 19: Kết quả từ select



Ảnh 20: Code thực hiện việc update

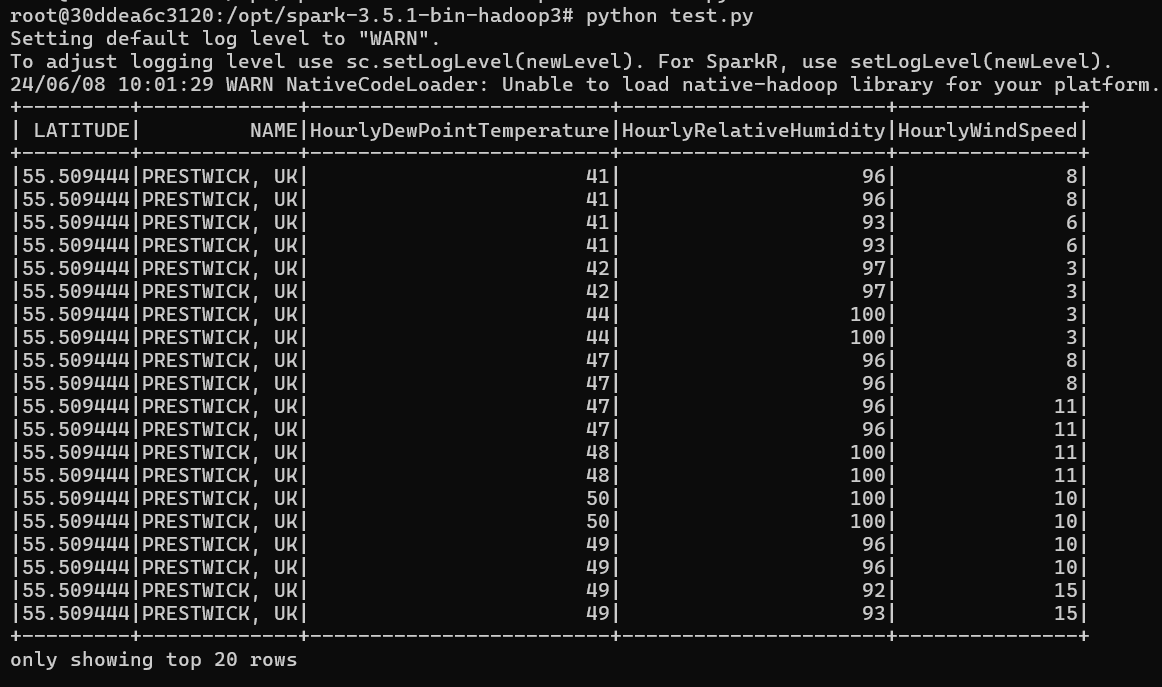
Theo tìm hiểu, do Spark.SQL không hỗ trợ cú pháp UPDATE nên việc cập nhật được thực hiện các hàm trong pyspark. Kết quả thu được như sau:

* DELETE: Ở phần này, ta sẽ thực hiện xoá dữ liệu trong bảng:

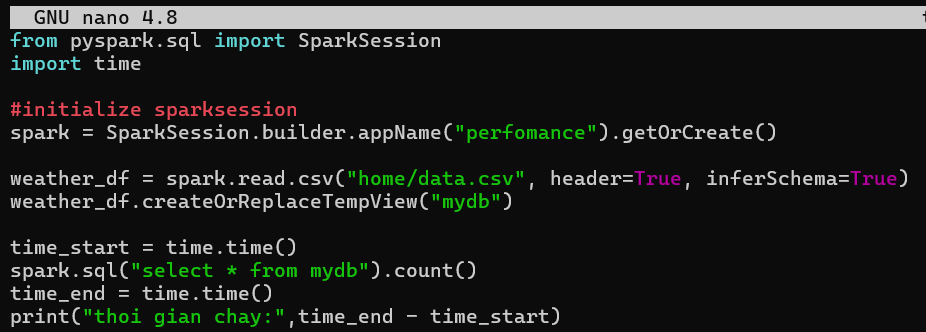
Ảnh 21: Kết quả của việc update

Ảnh 22: Code thực hiện việc xoá

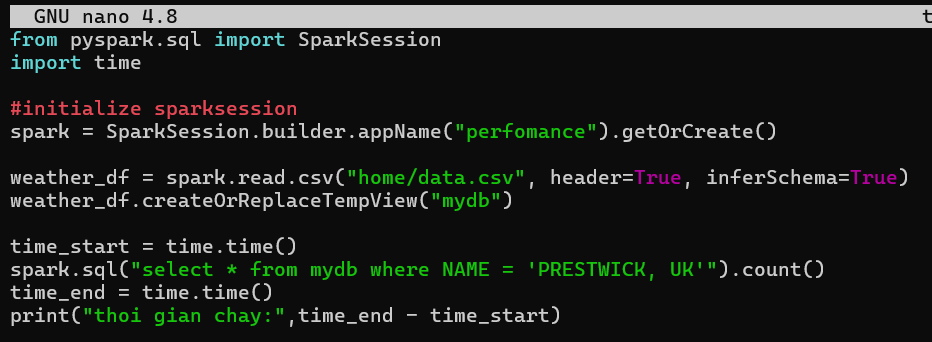
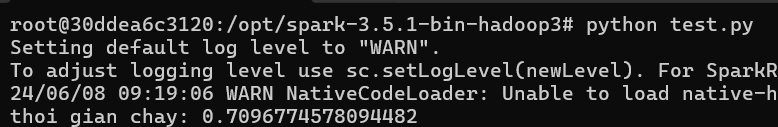
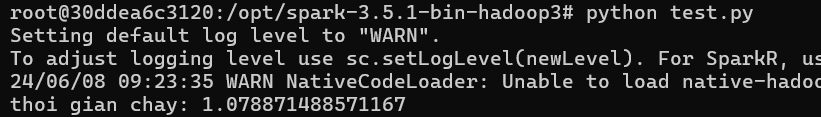
Kết quả thu được như sau:



* Tiếp đến, ta tìm hiểu xem WHERE ảnh hưởng tới hiệu năng của một số truy vấn như thế nào. Đầu tiên thử với truy vấn sau:



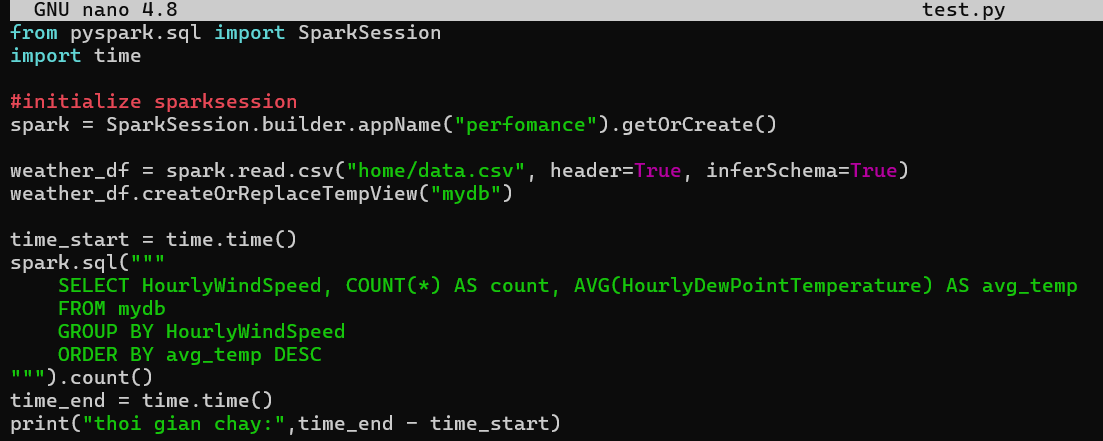
Ảnh 23: Truy vấn không sử dụng where



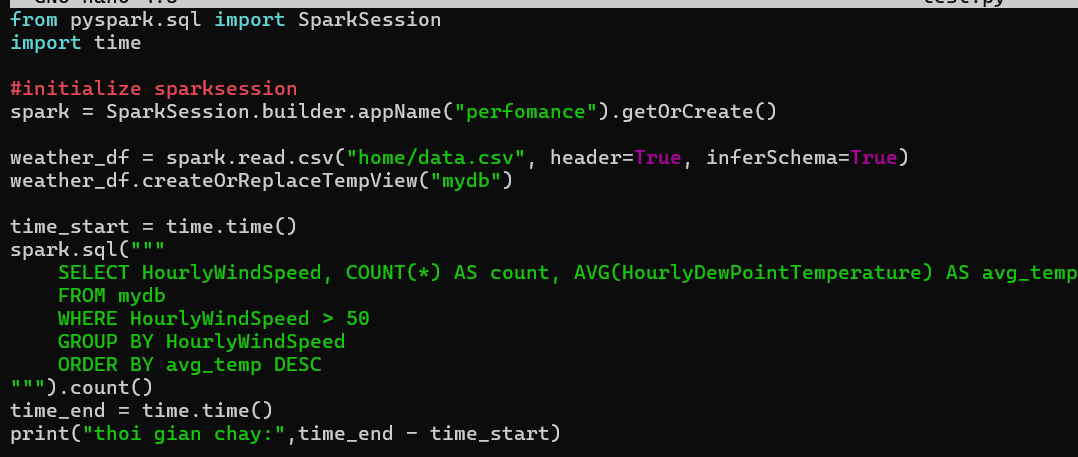
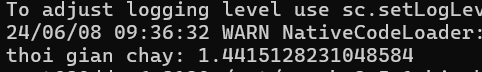
Ảnh 26: Thời gian khi sử dụng where

Ảnh 25: Truy vấn khi sử dụng where

Ảnh 24: Thời gian khi không sử dụng where

* Tiếp tục với với truy vấn khác:

Ảnh 27: Truy vấn không sử dụng where

Ảnh 28: Thời gian khi không sử dụng where

Ảnh 30: Thời gian khi sử dụng where

Ảnh 29: Truy vấn khi sử dụng where

* Qua các thử nghiệm trên, where có ảnh hưởng vào hiệu năng truy vấn, tuỳ thuộc vào các yếu tố như độ phức tạp của điều kiện, chỉ mục, các phép toán logic mà thời gian truy vấn sẽ khác nhau.

1. Kết luận:

Qua project này, ta nắm được cách xây dựng và chạy một docker container, cùng với đó là thực hiện sử dụng spark trong docker, biết cách sử dụng cơ sở dữ liệu trong docker và thực hiện một vài truy vấn, ngoài ra nhận thấy được sự ảnh hưởng của where đối với hiệu năng truy vấn.