

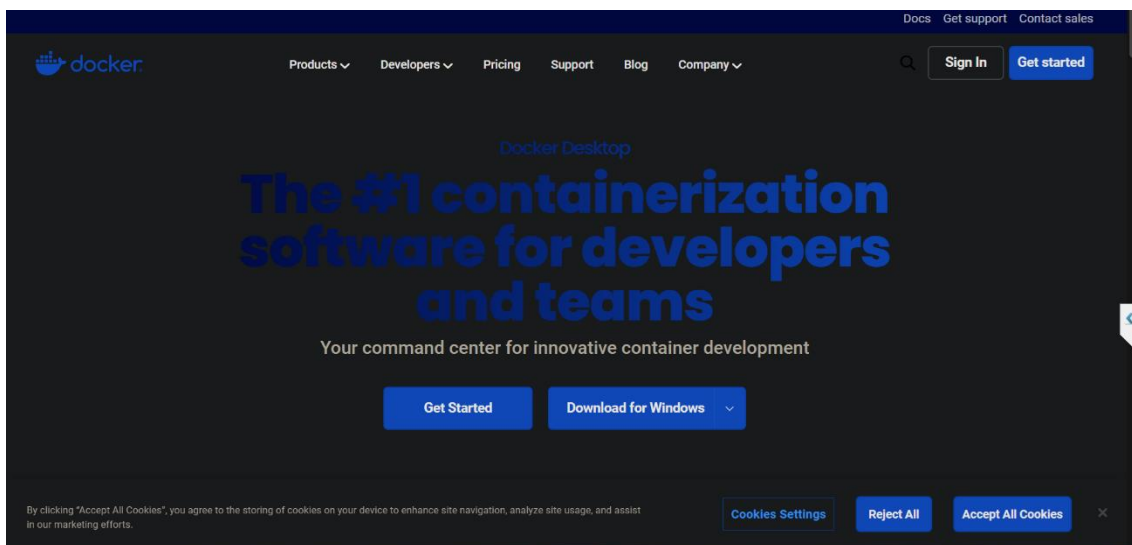
# SparkSQL in Docker Project

Mục tiêu: Mục tiêu của project này là cài đặt và chạy SparkSQL trong một Docker container. Container này sẽ lưu trữ một cơ sở dữ liệu (cụ thể là SQLite database) và cho phép thực hiện các truy vấn SQL sử dụng SparkSQL.

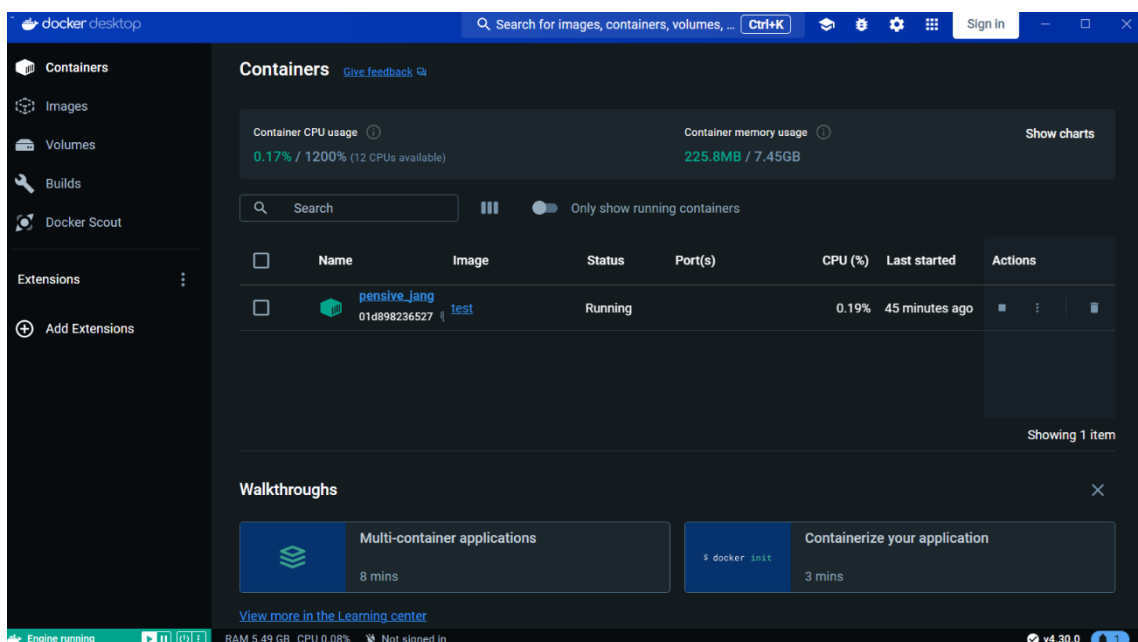
Setup:

Tải dockerdesktop tùy theo hệ điều hành của máy.

<https://www.docker.com/products/docker-desktop/>



Đây là giao diện.

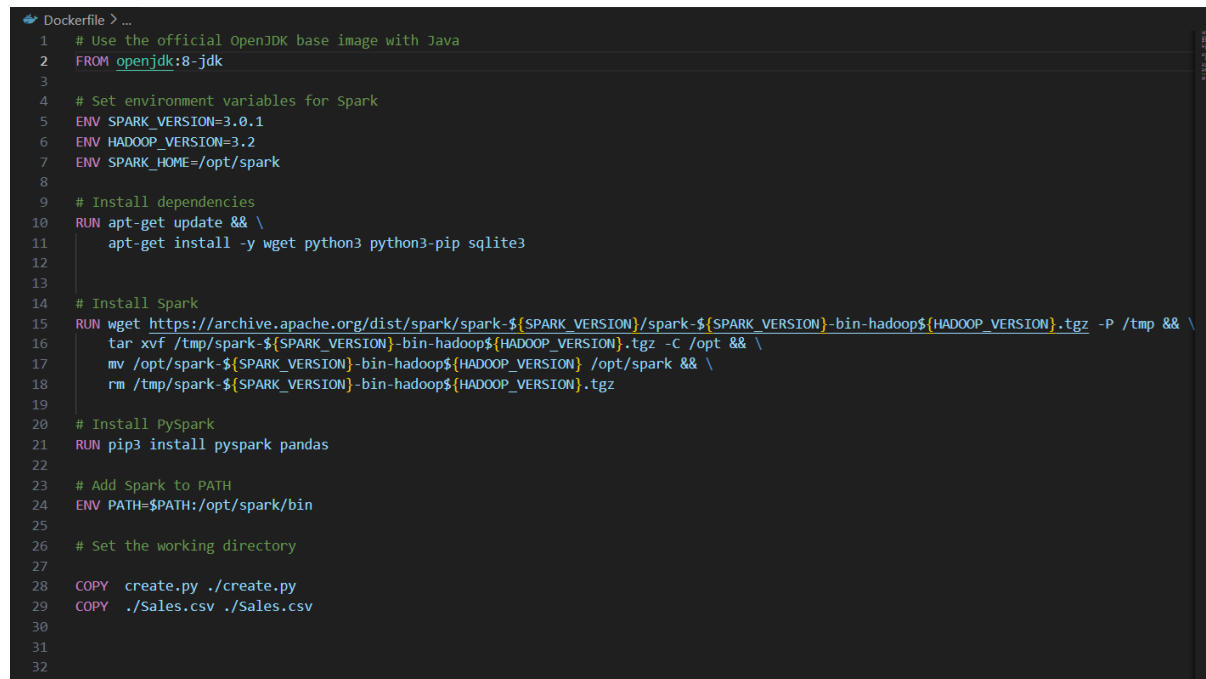


Ta tiến hành dow và cài đặt chương trình.

Sau đó em sử dụng VS Code để tạo và cài đặt cho chương trình.

## Giải nén file Sales.csv

Đầu tiên ta tạo file tên Dockerfile với nội dung như sau:



```
Dockerfile > ...
1 # Use the official OpenJDK base image with Java
2 FROM openjdk:8-jdk
3
4 # Set environment variables for Spark
5 ENV SPARK_VERSION=3.0.1
6 ENV HADOOP_VERSION=3.2
7 ENV SPARK_HOME=/opt/spark
8
9 # Install dependencies
10 RUN apt-get update && \
11     apt-get install -y wget python3 python3-pip sqlite3
12
13 # Install Spark
14 RUN wget https://archive.apache.org/dist/spark/spark-${SPARK_VERSION}/spark-${SPARK_VERSION}-bin-hadoop${HADOOP_VERSION}.tgz -P /tmp && \
15     tar xvf /tmp/spark-${SPARK_VERSION}-bin-hadoop${HADOOP_VERSION}.tgz -C /opt && \
16     mv /opt/spark-${SPARK_VERSION}-bin-hadoop${HADOOP_VERSION} /opt/spark && \
17     rm /tmp/spark-${SPARK_VERSION}-bin-hadoop${HADOOP_VERSION}.tgz
18
19 # Install PySpark
20 RUN pip3 install pyspark pandas
21
22 # Add Spark to PATH
23 ENV PATH=$PATH:/opt/spark/bin
24
25 # Set the working directory
26
27 COPY create.py ./create.py
28 COPY ./Sales.csv ./Sales.csv
29
30
31
32
```

Ở đây, đầu tiên sử dụng hình ảnh chính thức của OpenJDK với Java 8 làm cơ sở để xây dựng Image.

Sau đó thiết lập môi trường cần thiết cho spark, SPARK\_VERSION xác định phiên bản Spark mà bạn muốn cài đặt, HADOOP\_VERSION xác định phiên bản Hadoop và SPARK\_HOME là đường dẫn nơi Spark sẽ được cài đặt.

Chạy lệnh apt-get update để cập nhật danh sách các gói và sau đó cài đặt các gói cần thiết như wget, python3, python3-pip, và sqlite3. Cờ -y được sử dụng để tự động xác nhận các lời nhắc trong quá trình cài đặt.

Cài đặt spark:

Tải xuống file tar.gz của Spark từ trang lưu trữ của Apache về thư mục /tmp.

Giải nén file tar.gz vào thư mục /opt.

Đổi tên thư mục vừa giải nén thành /opt/spark để dễ dàng tham chiếu.

Xóa file tar.gz đã tải về để tiết kiệm không gian đĩa.

Sử dụng pip3 để cài đặt các gói Python cần thiết cho PySpark và pandas.

Cập nhật biến môi trường PATH để bao gồm đường dẫn đến thư mục bin của Spark. Điều này cho phép bạn gọi các lệnh của Spark từ bất kỳ đâu trong container.

Sao chép file create.py và Sales.csv từ máy của bạn vào thư mục làm việc mặc định trong container. Những file này sẽ có sẵn để sử dụng trong container.

**Sau khi setup xong dùng lệnh ‘docker build -t test .’ và ‘docker run -it test’ để chạy và cài containers và image trong dockerdestop.**

```
PS C:\Users\manhd\OneDrive\Máy tính\New folder> docker build -t test .
2024/06/12 21:22:26 http2: server: error reading preface from client //./pipe/docker_engine: file has already been closed
[+] Building 1.1s (12/12) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 900B
=> [internal] load metadata for docker.io/library/openjdk:8-jdk
=> [internal] load .dockerignore
=> transferring context: 2B
=> [1/7] FROM docker.io/library/openjdk:8-jdk@sha256:86e863cc57215cfb181bd319736d0baf625fe8f150577f9eb58bd937f5452cb8
=> [internal] load build context
=> => transferring context: 91B
=> CACHED [2/7] RUN apt-get update && apt-get install -y wget curl python3 python3-pip sqlite3
=> CACHED [3/7] RUN wget https://archive.apache.org/dist/spark/spark-3.0.1/spark-3.0.1-bin-hadoop3.2.tgz && tar xvf spark-3.0.1-bin-hadoop3.2.tgz &&
=> CACHED [4/7] RUN pip3 install pyspark pandas
=> CACHED [5/7] COPY create.py ./create.py
=> CACHED [6/7] COPY ./Sales.csv ./Sales.csv
=> CACHED [7/7] COPY codepy.py ./codepy.py
=> exporting to image
=> => exporting layers
=> => writing image sha256:1e64e7f3e1145a9bc26c055895657b2777fb0b086119c734b0e92a34459439b5
=> => naming to docker.io/library/test

What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\Users\manhd\OneDrive\Máy tính\New folder> docker run -it test
root@0c6d57368a91:/#
```

## Thực thi:

Tạo cơ sở dữ liệu:

Chuẩn bị 1 file có tên create.py giúp tạo ra một cơ sở dữ liệu SQLite bằng sqlite3.

```
Dockerfile .\ ● Dockerfile C:\...\pyspark-with-docker-main create.py X query.ipynb
create.py > ...
1 import sqlite3
2 import random
3 import string
4 import pandas as pd
5
6 data = pd.read_csv('Sales.csv')
7 # Connect to SQLite database (creates it if it doesn't exist)
8 conn = sqlite3.connect('Sales_record.db')
9 cursor = conn.cursor()
10
11 # Create a sample table
12 cursor.execute('''
13 CREATE TABLE IF NOT EXISTS Sales (
14     Region VARCHAR(255) NOT NULL,
15     Country VARCHAR(255) NOT NULL,
16     Sold FLOAT NOT NULL,
17     Price FLOAT NOT NULL,
18     Cost FLOAT NOT NULL,
19     Profit FLOAT NOT NULL
20 );
21 ''')
22
23 # Insert data from CSV into the database
24 data.to_sql('Sales', conn, if_exists='append', index=False)
25
26 # Commit and close the connection
27 conn.commit()
28 conn.close()
```

Đầu tiên tạo database tên là 'Sales\_record' sau đó tạo table tên 'Sales' bao gồm 6 cột Region, Country, Sold, Price, Cost, Profit.

Ta đọc file 'Sales.csv' bằng pandas bao gồm 1,000,000 dòng.

Để kết nối với database SQLite, sử dụng một chuẩn API để tương tác với cơ sở dữ liệu có tên là JDBC driver. Cụ thể dự án sử dụng sqlite jdbc driver 3.46.0.0.

```
What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\Users\manhd\OneDrive\Máy tính\New folder> docker run -it test
root@0c150b692ae9:/# python3 create.py
root@0c150b692ae9:/# python3
Python 3.9.2 (default, Feb 28 2021, 17:03:44)
[GCC 10.2.1 20210110] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> █
```

Sau đó dùng những câu lệnh này kể nốt nối được với table.

```
from pyspark.sql import SparkSession
import time

spark = SparkSession.builder.config('spark.jars.packages', 'org.xerial:sqlite-jdbc:3.46.0.0').getOrCreate()
df = spark.read.format('jdbc').options(driver='org.sqlite.JDBC', dbtable='Sales', url='jdbc:sqlite:/Sales_record.db').load()
df.show()
```

Do code trực tiếp lên terminal lên hơi khó quan sát.

```
>>> from pyspark.sql import SparkSession
>>> spark = SparkSession.builder.config('spark.jars.packages', 'org.xerial:sqlite-jdbc:3.46.0.0').getOrCreate()
:: loading settings :: url = jar:file:/usr/local/lib/python3.9/dist-packages/pyspark/jars/ivy-2.5.1.jar!/org/apache/ivy/core/settings/ivysettings.xml
Ivy Default Cache set to: /root/.ivy2/cache
The jars for the packages stored in: /root/.ivy2/jars
org.xerial#sqlite-jdbc added as a dependency
:: resolving dependencies :: org.apache.spark#spark-submit-parent-936a0dd6-c08c-4e09-8e64-7c34bdde8266;1.0
  confs: [default]
    found org.xerial#sqlite-jdbc;3.46.0.0 in central
    found org.slf4j#slf4j-api;1.7.36 in central
downloading https://repo1.maven.org/maven2/org/xerial/sqlite-jdbc/3.46.0.0/sqlite-jdbc-3.46.0.0.jar ...
[SUCCESSFUL ] org.xerial#sqlite-jdbc;3.46.0.0!sqlite-jdbc.jar (12286ms)
downloading https://repo1.maven.org/maven2/org/slf4j/slf4j-api/1.7.36/slf4j-api-1.7.36.jar ...
[SUCCESSFUL ] org.slf4j#slf4j-api;1.7.36!slf4j-api.jar (411ms)
:: resolution report :: resolve 6345ms :: artifacts dl 12702ms
  :: modules in use:
    org.slf4j#slf4j-api;1.7.36 from central in [default]
    org.xerial#sqlite-jdbc;3.46.0.0 from central in [default]
-----
|   conf   | modules | artifacts | | | | |
|---|---|---|---|---|---|---|
|   conf   | number | search | dwnlded | evicted | number | dwnlded |
|-----|-----|-----|-----|-----|
| default  | 2      | 2      | 2       | 0       | 2      | 2       |
-----
:: retrieving :: org.apache.spark#spark-submit-parent-936a0dd6-c08c-4e09-8e64-7c34bdde8266
  confs: [default]
  2 artifacts copied, 0 already retrieved (13336kB/31ms)
24/06/12 14:30:21 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
>>> df = spark.read.format('jdbc').options(driver='org.sqlite.JDBC', dbtable='Sales', url='jdbc:sqlite:/Sales_record.db').load()
>>> df.show()
+-----+-----+-----+-----+-----+-----+
| Region | Country | Sold | Price | Cost | Profit |
+-----+-----+-----+-----+-----+-----+
| Asia   | VN      | 1.0  | 2.0   | 3.0  | 4.0    |
| Middle East and N... | Morocco | 4611.0 | 109.28 | 165258.23 | 338631.84 |
| Australia and Oce... | Papua New Guinea | 360.0 | 421.89 | 131288.4 | 20592.0 |
| Sub-Saharan Africa | Djibouti | 562.0 | 109.28 | 20142.08 | 41273.28 |
| Europe | Slovakia | 3973.0 | 47.45 | 126301.67 | 62217.18 |
| Asia   | Sri Lanka | 1379.0 | 9.33 | 9542.68 | 3323.39 |
| Sub-Saharan Africa | Seychelles | 597.0 | 47.45 | 18978.63 | 9349.02 |
| Sub-Saharan Africa | Tanzania | 1476.0 | 47.45 | 46922.04 | 23114.16 |
| Sub-Saharan Africa | Ghana | 896.0 | 651.21 | 470364.16 | 113120.0 |
| Sub-Saharan Africa | Tanzania | 7768.0 | 437.2 | 2045547.5 | 1350622.1 |
| Asia   | Taiwan | 8034.0 | 9.33 | 55595.28 | 19361.94 |
| Middle East and N... | Algeria | 9669.0 | 437.2 | 2546137.8 | 1681149.0 |
| Asia   | Singapore | 7676.0 | 152.58 | 747949.44 | 423254.62 |
| Australia and Oce... | Papua New Guinea | 9092.0 | 109.28 | 325857.28 | 667716.5 |
| Asia   | Vietnam | 7984.0 | 81.73 | 452453.28 | 200079.05 |
| Sub-Saharan Africa | Uganda | 451.0 | 81.73 | 25558.17 | 11302.06 |
| Sub-Saharan Africa | Zimbabwe | 9623.0 | 651.21 | 5051690.0 | 1214903.8 |
| Sub-Saharan Africa | Ethiopia | 662.0 | 437.2 | 174324.45 | 115101.94 |
| Europe | France | 5758.0 | 437.2 | 1516254.1 | 1001143.44 |
| Central America a... | The Bahamas | 9137.0 | 81.73 | 517793.78 | 228973.22 |
+-----+-----+-----+-----+-----+-----+
only showing top 20 rows
```

Ta đã build xong được Image.

Ở hàng đầu tiên của dữ liệu, em thay đổi data như vậy để dễ nhìn khi thực hiện thao tác CRUD.

## Thực hiện các thao tác CRUD.

Để thực hiện các thao tác CRUD hiệu quả, dự án sử dụng DataFrame của pyspark vì DataFrame đã được xây dựng để hoạt động hiệu quả với dataset lớn hay Big Data.

### Create

```
# Create operation
from pyspark.sql import Row
# Create a new row
new_row = Row(Region = 'Asia', Country='Japane', Sold = 111111, Price = 2222222, Cost=333333, Profit=444444)
# Convert the Row into a DataFrame
new_df = spark.createDataFrame([new_row], schema=df.schema)
# Append the new DataFrame to the existing DataFrame
df = df.union(new_df)
# Show the updated DataFrame
print("Create")
df.tail()
```

### Tạo SparkSession

Tạo một dòng mới, Ở đây, Row được sử dụng để tạo một dòng mới với các giá trị được chỉ định.

Chuyển đổi Row thành DataFrame, dòng này chuyển đổi Row thành DataFrame mới với cùng schema như df.

Thêm DataFrame mới vào DataFrame hiện có, dòng này thêm new\_df vào df hiện có.

### Hiển thị DataFrame cập nhật.

```
>>> from pyspark.sql import Row
>>> new_row = Row(Region = 'Asia', Country='Japane', Sold = 1111.1 , Price = 222.2, Cost=333.3, Profit=444.4)
>>> new_df = spark.createDataFrame([new_row], schema=df.schema)
>>> df = df.union(new_df)
>>> df.tail(5)
[Row(Region='Central America and the Caribbean', Country='Panama', Sold=4068.0, Price=651.2100219726562, Cost=2135537.25, Profit=513585.0), Row(Region='Europe', Country='Norway', Sold=5266.0, Price=651.2100219726562, Cost=2764439.25, Profit=664832.5), Row(Region='Europe', Country='Montenegro', Sold=8551.0, Price=47.45000076293945, Cost=271836.28125, Profit=133908.65625), Row(Region='Central America and the Caribbean', Country='Nicaragua', Sold=7519.0, Price=421.8900146484375, Cost=2742104.0, Profit=430086.8125), Row(Region='Asia', Country='Japane', Sold=1111.0999755859375, Price=222.1999969482422, Cost=333.29998779296875, Profit=444.3999938964844)]
>>>
```

Dòng gạch đỏ là data đã được thêm.

Read.

```
# Read operation
filtered_df = df.filter(df.Sold > 1000)
# Search high Sold price.
print("read")
filtered_df.show()
print("Sold higher 1000: ", filtered_df.count())
```

```
964844)]
>>> filtered_df = df.filter(df.Sold > 1000)
>>> filtered_df.show()
+-----+-----+-----+-----+-----+-----+
|          Region|          Country| Sold| Price|      Cost|      Profit|
+-----+-----+-----+-----+-----+-----+
|Middle East and N...|          Morocco|4611.0|109.28|165258.23| 338631.84|
|          Europe|          Slovakia|3973.0| 47.45|126301.67|  62217.18|
|          Asia|          Sri Lanka|1379.0|  9.33|  9542.68|   3323.39|
|Sub-Saharan Africa|          Tanzania|1476.0| 47.45| 46922.04|  23114.16|
|Sub-Saharan Africa|          Tanzania|7768.0| 437.2|2045547.5|1350622.1|
|          Asia|          Taiwan|8034.0|  9.33| 55595.28|  19361.94|
|Middle East and N...|          Algeria|9669.0| 437.2|2546137.8|1681149.0|
|          Asia|          Singapore|7676.0|152.58|747949.44|423254.62|
|Australia and Oce...|Papua New Guinea|9092.0|109.28|325857.28|  667716.5|
|          Asia|          Vietnam|7984.0| 81.73|452453.28|200079.05|
|Sub-Saharan Africa|          Zimbabwe|9623.0|651.21|5051690.0|1214903.8|
|          Europe|          France|5758.0| 437.2|1516254.1|1001143.44|
|Central America a...|The Bahamas|9137.0| 81.73|517793.78| 228973.22|
|Central America a...|          Haiti|2052.0|651.21|1077217.9|  259065.0|
|Central America a...|          Nicaragua|7791.0|668.27|3915289.2|1291202.4|
|          Asia|          Turkmenistan|6670.0|154.06| 606503.1|  421077.1|
|          Europe|United Kingdom|1038.0| 437.2|273336.53|180477.06|
|          Asia|          China|5791.0|651.21|3040043.2| 731113.75|
|Sub-Saharan Africa|          Uganda|6031.0| 437.2|1588143.2|1048610.0|
|Middle East and N...|          Kuwait|1466.0|668.27| 736723.6| 242960.19|
+-----+-----+-----+-----+-----+-----+
only showing top 20 rows

>>> print("Sold higher 1000: ", filtered_df.count())
Sold higher 1000:  1800285
>>> █
```

Update:

```
# Update the last row column with Sold = 1
df = df.withColumn('Sold', when((df.Region == 'Asia') & \
                                (df.Country == 'VN') & \
                                (df.Price == 2) & \
                                (df.Cost == 3) & \
                                (df.Profit == 4), 10000).otherwise(df.Sold))

# Show the updated DataFrame
df.show(10)
```

```
>>> from pyspark.sql.functions import when
>>> df = df.withColumn('Sold', when((df.Region == 'Asia') & \
...                               (df.Country == 'VN') & \
...                               (df.Price == 2) & \
...                               (df.Cost == 3) & \
...                               (df.Profit == 4), 10000).otherwise(df.Sold))
>>> df.show(10)
```

Region	Country	Sold	Price	Cost	Profit
Asia	VN	10000.0	2.0	3.0	4.0
Middle East and N...	Morocco	4611.0	109.28	165258.23	338631.84
Australia and Oce...	Papua New Guinea	360.0	421.89	131288.4	20592.0
Sub-Saharan Africa	Djibouti	562.0	109.28	20142.08	41273.28
Europe	Slovakia	3973.0	47.45	126301.67	62217.18
Asia	Sri Lanka	1379.0	9.33	9542.68	3323.39
Sub-Saharan Africa	Seychelles	597.0	47.45	18978.63	9349.02
Sub-Saharan Africa	Tanzania	1476.0	47.45	46922.04	23114.16
Sub-Saharan Africa	Ghana	896.0	651.21	470364.16	113120.0
Sub-Saharan Africa	Tanzania	7768.0	437.2	2045547.5	1350622.1

only showing top 10 rows

Đã update sold từ 1.0 lên 10000.0

Delete

```
# Delete Operation
df = df.filter(~((df.Region == 'Asia') & \
                 (df.Country == 'VN') & \
                 (df.Sold == 10000) & \
                 (df.Price == 2) & \
                 (df.Cost == 3) & \
                 (df.Profit == 4))) # Try delete the first row

df.head(5)
```



```
>>> df = df.filter(~((df.Region == 'Asia') &
...                  (df.Country == 'VN') &
...                  (df.Sold == 10000) &
...                  (df.Price == 2) &
...                  (df.Cost == 3) &
...                  (df.Profit == 4))) # Try delete the first row
>>> df.show()
```

Region	Country	Sold	Price	Cost	Profit
Middle East and N...	Morocco	4611.0	109.28	165258.23	338631.84
Australia and Oce...	Papua New Guinea	360.0	421.89	131288.4	20592.0
Sub-Saharan Africa	Djibouti	562.0	109.28	20142.08	41273.28
Europe	Slovakia	3973.0	47.45	126301.67	62217.18
Asia	Sri Lanka	1379.0	9.33	9542.68	3323.39
Sub-Saharan Africa	Seychelles	597.0	47.45	18978.63	9349.02
Sub-Saharan Africa	Tanzania	1476.0	47.45	46922.04	23114.16
Sub-Saharan Africa	Ghana	896.0	651.21	470364.16	113120.0
Sub-Saharan Africa	Tanzania	7768.0	437.2	2045547.5	1350622.1
Asia	Taiwan	8034.0	9.33	55595.28	19361.94
Middle East and N...	Algeria	9669.0	437.2	2546137.8	1681149.0
Asia	Singapore	7676.0	152.58	747949.44	423254.62
Australia and Oce...	Papua New Guinea	9092.0	109.28	325857.28	667716.5
Asia	Vietnam	7984.0	81.73	452453.28	200079.05
Sub-Saharan Africa	Uganda	451.0	81.73	25558.17	11302.06
Sub-Saharan Africa	Zimbabwe	9623.0	651.21	5051690.0	1214903.8
Sub-Saharan Africa	Ethiopia	662.0	437.2	174324.45	115101.94
Europe	France	5758.0	437.2	1516254.1	1001143.44
Central America a...	The Bahamas	9137.0	81.73	517793.78	228973.22
Central America a...	Haiti	2052.0	651.21	1077217.9	259065.0

```
only showing top 20 rows

>>> 
```

Đã xóa dòng đầu.

**So sánh tốc độ thực thi câu lệnh truy vấn:**

Import thư viện thời gian.

```

import time

start_time = time.time()
df.createOrReplaceTempView("Sales")
spark.sql("SELECT * FROM Sales").show(10)
end_time = time.time()
print("execution time", end_time - start_time)

start_time1 = time.time()
spark.sql("SELECT * FROM Sales WHERE Sold > 1000").show(10)
end_time1 = time.time()
print("execution time1", end_time1 - start_time1)

start_time2 = time.time()
spark.sql("SELECT Country, SUM(Sold) as TotalValue FROM Sales WHERE Sold > 1000 GROUP BY Country").show(10)
end_time2 = time.time()
print("execution time2", end_time2 - start_time2)

start_time3 = time.time()
spark.sql("SELECT Country, SUM(Sold) as TotalValue FROM Sales WHERE Sold > 1000 GROUP BY Country HAVING TotalValue > 15000").show(10)
end_time3 = time.time()
print("execution time3", end_time3 - start_time3)

```

Khi không có WHERE, GROUP BY, HAVING.

```

>>> import time
>>> start_time = time.time()
>>> df.createOrReplaceTempView("Sales")
>>> spark.sql("SELECT * FROM Sales").show(10)
+-----+-----+-----+-----+-----+-----+
|          Region|          Country| Sold| Price|      Cost|      Profit|
+-----+-----+-----+-----+-----+-----+
|          Asia|          VN|    1.0|    2.0|        3.0|        4.0|
|Middle East and N...|Morocco|4611.0|109.28|165258.23|338631.84|
|Australia and Oce...|Papua New Guinea| 360.0|421.89| 131288.4|  20592.0|
|Sub-Saharan Africa|Djibouti| 562.0|109.28| 20142.08|  41273.28|
|          Europe|Slovakia|3973.0| 47.45|126301.67| 62217.18|
|          Asia|Sri Lanka|1379.0|  9.33|  9542.68|  3323.39|
|Sub-Saharan Africa|Seychelles| 597.0| 47.45| 18978.63|  9349.02|
|Sub-Saharan Africa|Tanzania|1476.0| 47.45| 46922.04| 23114.16|
|Sub-Saharan Africa|Ghana| 896.0|651.21|470364.16|113120.0|
|Sub-Saharan Africa|Tanzania|7768.0| 437.2|2045547.5|1350622.1|
+-----+-----+-----+-----+-----+-----+
only showing top 10 rows

>>> end_time = time.time()
>>> print("execution time", end_time - start_time)
execution time 11.623245000839233
>>> 

```

Thời gian truy vấn vẫn là 11.6 ms

```
>>> start_time1 = time.time()
>>> spark.sql("SELECT * FROM Sales WHERE Sold > 1000").show(10)
+-----+-----+-----+-----+-----+-----+
| Region | Country | Sold | Price | Cost | Profit |
+-----+-----+-----+-----+-----+-----+
| Middle East and N... | Morocco | 4611.0 | 109.28 | 165258.23 | 338631.84 |
| Europe | Slovakia | 3973.0 | 47.45 | 126301.67 | 62217.18 |
| Asia | Sri Lanka | 1379.0 | 9.33 | 9542.68 | 3323.39 |
| Sub-Saharan Africa | Tanzania | 1476.0 | 47.45 | 46922.04 | 23114.16 |
| Sub-Saharan Africa | Tanzania | 7768.0 | 437.2 | 2045547.5 | 1350622.1 |
| Asia | Taiwan | 8034.0 | 9.33 | 55595.28 | 19361.94 |
| Middle East and N... | Algeria | 9669.0 | 437.2 | 2546137.8 | 1681149.0 |
| Asia | Singapore | 7676.0 | 152.58 | 747949.44 | 423254.62 |
| Australia and Oce... | Papua New Guinea | 9092.0 | 109.28 | 325857.28 | 667716.5 |
| Asia | Vietnam | 7984.0 | 81.73 | 452453.28 | 200079.05 |
+-----+-----+-----+-----+-----+-----+
only showing top 10 rows

>>> end_time1 = time.time()
>>> print("execution time1", end_time1 - start_time1)
execution time1 5.310858964920044
>>> █
```

Khi có WHERE time là 5.3 ms. Nhanh hơn một chút so với không có Where.

```
>>> start_time2 = time.time()
>>> spark.sql("SELECT Country, SUM(Sold) as TotalValue FROM Sales WHERE Sold > 1000 GROUP BY Country").show(10)
+-----+-----+
| Country | TotalValue |
+-----+-----+
| Chad | 2.729724E7 |
| Russia | 2.6482908E7 |
| Yemen | 2.6618532E7 |
| Senegal | 2.7325484E7 |
| Sweden | 2.657155E7 |
| Kiribati | 2.6676929E7 |
| Eritrea | 2.6617653E7 |
| Philippines | 2.6588033E7 |
| Tonga | 2.6194518E7 |
| Djibouti | 2.6861152E7 |
+-----+-----+
only showing top 10 rows

>>> end_time2 = time.time()
>>> print("execution time2", end_time2 - start_time2)
execution time2 10.908334732055664
>>> █
```

Khi có GROUP BY time là 10.9 ms. Lâu hơn một chút do tính toán tổng của giá trị.

```
>>> start_time3 = time.time()
>>> spark.sql("SELECT Country, SUM(Sold) as TotalValue FROM Sales WHERE Sold > 1000 GROUP BY Country HAVING TotalValue > 15000").show(10)
+-----+-----+
| Country | TotalValue |
+-----+-----+
| Chad | 2.729724E7 |
| Russia | 2.6482908E7 |
| Yemen | 2.6618532E7 |
| Senegal | 2.7325484E7 |
| Sweden | 2.657155E7 |
| Kiribati | 2.6676929E7 |
| Eritrea | 2.6617653E7 |
| Philippines | 2.6588033E7 |
| Tonga | 2.6194518E7 |
| Djibouti | 2.6861152E7 |
+-----+-----+
only showing top 10 rows

>>> end_time3 = time.time()
>>> print("execution time3", end_time3 - start_time3)
execution time3 8.962153673171997
>>> █
```

Khi có HAVING thời gian chạy là 8.9ms. Nhanh hơn khi không có HAVING.