

## TÀI LIỆU

### DỰ ÁN CUỐI KỲ: CHẠY SQLSPARK trong DOCKER CONTAINER

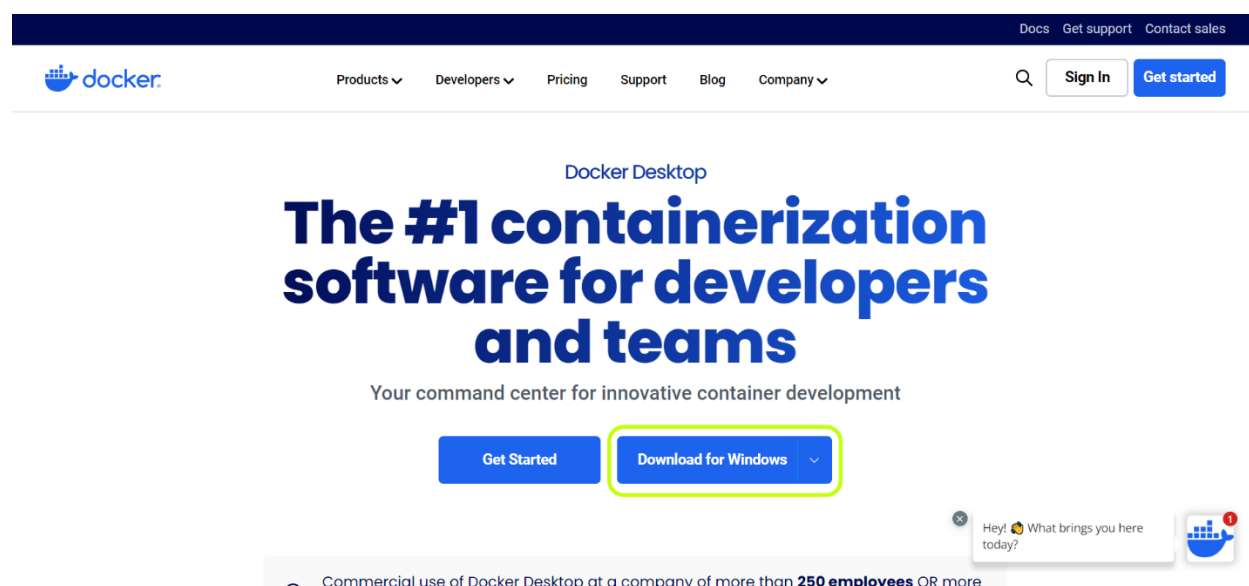
#### Mô tả

Tài liệu này nhằm mục đích hướng dẫn thiết lập và chạy SparkSQL trong Docker container. Các phần phụ thuộc cần thiết trong container được cài đặt để thực hiện các truy vấn SQL trên cơ sở dữ liệu (SQLite) cũng được lưu trữ trong cùng một container.

#### Điều kiện

Trước khi triển khai dự án, hãy đảm bảo máy tính của bạn có đủ bộ nhớ và đã cài đặt Docker (chạy hiệu quả với RAM 16GB).

Tải Docker Desktop tại [Docker Desktop](#) và chọn phiên bản phù hợp với hệ điều hành của máy.



*Giao diện website tải Docker Desktop*

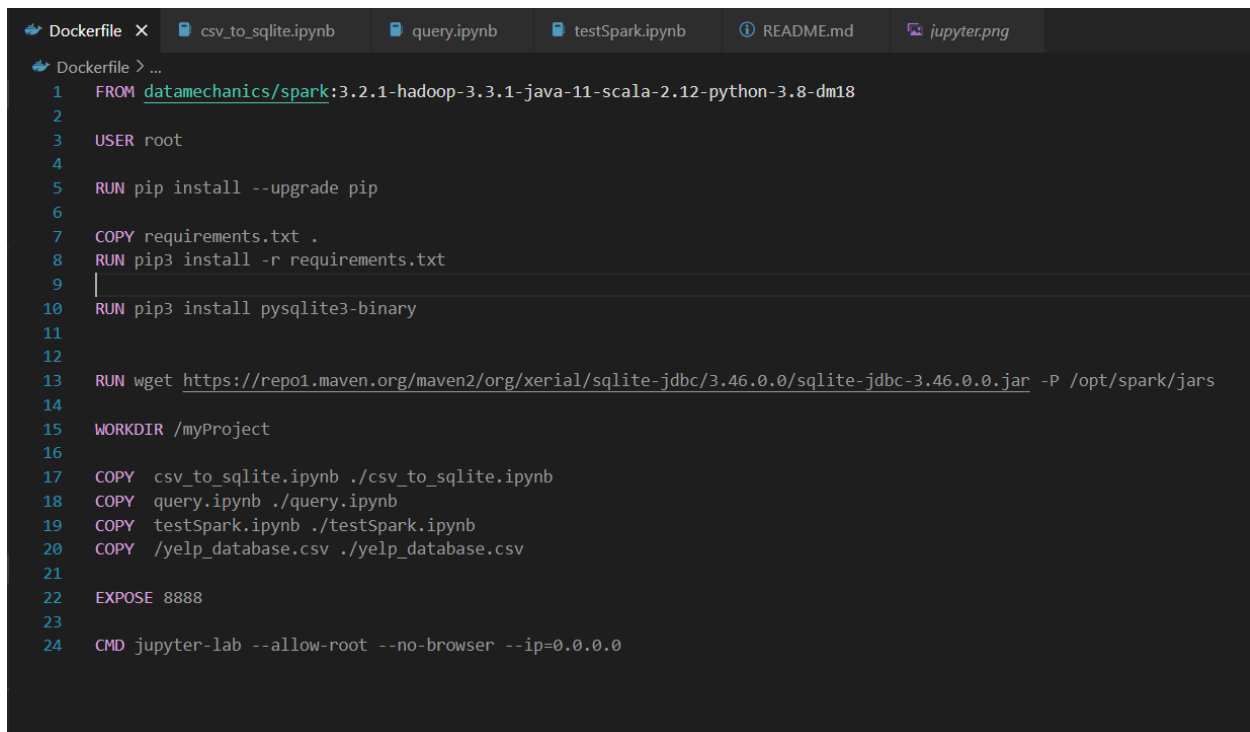
## Cài đặt docker

### Bước 1: Xây dựng Dockerfile

Để triển khai Spark với Docker, chúng tôi muốn cài đặt một số gói (connectors) để không cần cài đặt riêng. Nếu kiểm tra Docker hub, chúng tôi thấy rằng có nhiều tùy chọn có sẵn cho Spark.

Dự án đã lựa chọn image [datamechanics/spark](https://hub.docker.com/r/datamechanics/spark) vì nó cung cấp tất cả các connectors cần thiết mà chúng ta cần:

- Jupyter Notebook
- Tích hợp hỗ trợ Python & PySpark
- pip và conda (vì vậy rất dễ cài đặt các gói bổ sung)



```
Dockerfile X  csv_to_sqlite.ipynb  query.ipynb  testSpark.ipynb  README.md  jupyter.png
Dockerfile > ...
1 FROM datamechanics/spark:3.2.1-hadoop-3.3.1-java-11-scala-2.12-python-3.8-dm18
2
3 USER root
4
5 RUN pip install --upgrade pip
6
7 COPY requirements.txt .
8 RUN pip3 install -r requirements.txt
9
10 RUN pip3 install pysqlite3-binary
11
12
13 RUN wget https://repo1.maven.org/maven2/org/xerial/sqlite-jdbc/3.46.0.0/sqlite-jdbc-3.46.0.0.jar -P /opt/spark/jars
14
15 WORKDIR /myProject
16
17 COPY csv_to_sqlite.ipynb ./csv_to_sqlite.ipynb
18 COPY query.ipynb ./query.ipynb
19 COPY testSpark.ipynb ./testSpark.ipynb
20 COPY /yelp_database.csv ./yelp_database.csv
21
22 EXPOSE 8888
23
24 CMD jupyter-lab --allow-root --no-browser --ip=0.0.0.0
```

*Docker file*

Đầu tiên, nó sẽ cài đặt datamechanics spark image, sau đó

sẽ cài đặt các gói cần thiết có trong *requirements.txt*, đặt working directory và đặt cổng ra cho Jupyter Notebook là 8888. Cuối cùng dẫn đến jupyter lab.

Như vậy ta đã viết được một docker file chuẩn bị các cài đặt cơ bản cho một container.

## Bước 2: Xây dựng Docker Image

```
docker build . -t sparkhome
```

Câu lệnh khởi tạo một docker image có tên là **sparkhome** từ *Dockerfile*.

```
PS C:\Users\Admin> cd Project
PS C:\Users\Admin\Project> docker build . -t sparkhome
[+] Building 0.0s (0/0)  docker:default
2024/06/04 15:52:45 http2: server: error reading preface from client //./pipe/docker_engine: file has already been close
[+] Building 4.5s (2/3)  docker:default
[+] Building 7.6s (17/17) FINISHED  docker:default
=> [internal] load build definition from Dockerfile 0.1s
=> => transferring dockerfile: 644B 0.1s
=> [internal] load metadata for docker.io/datamechanics/spark:3.2.1-hadoop-3.3.1-java-11-scala-2.12-python-3.8-d 6.8s
=> [auth] datamechanics/spark:pull token for registry-1.docker.io 0.0s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [ 1/11] FROM docker.io/datamechanics/spark:3.2.1-hadoop-3.3.1-java-11-scala-2.12-python-3.8-dm18@sha256:945cb 0.0s
=> [internal] load build context 0.0s
=> => transferring context: 1.25kB 0.0s
=> CACHED [ 2/11] RUN pip install --upgrade pip 0.0s
=> CACHED [ 3/11] COPY requirements.txt . 0.0s
=> CACHED [ 4/11] RUN pip3 install -r requirements.txt 0.0s
=> CACHED [ 5/11] RUN pip install pysqlite3-binary 0.0s
=> CACHED [ 6/11] RUN wget https://repo1.maven.org/maven2/org/xerial/sqlite-jdbc/3.46.0.0/sqlite-jdbc-3.46.0.0.j 0.0s
=> CACHED [ 7/11] WORKDIR /myProject 0.0s
=> CACHED [ 8/11] COPY createdb.ipynb ./createdb.ipynb 0.0s
=> CACHED [ 9/11] COPY ./yelp_database.csv ./yelp_database.csv 0.0s
=> CACHED [10/11] COPY query.ipynb ./query.ipynb 0.0s
=> [11/11] COPY testSpark.ipynb ./testSpark.ipynb 0.1s
=> exporting to image 0.2s
=> => exporting layers 0.1s
=> => writing image sha256:b82da2ea951b2a1df69c3e8b183756ae8242ddf4285a6dc03bed64d704666d16 0.0s
=> => naming to docker.io/library/sparkhome 0.0s

What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\Users\Admin\Project> docker image ls
REPOSITORY TAG IMAGE ID CREATED SIZE
sparkhome latest b82da2ea951b 18 seconds ago 3.63GB
```

### Bước 3: Khởi tạo docker container

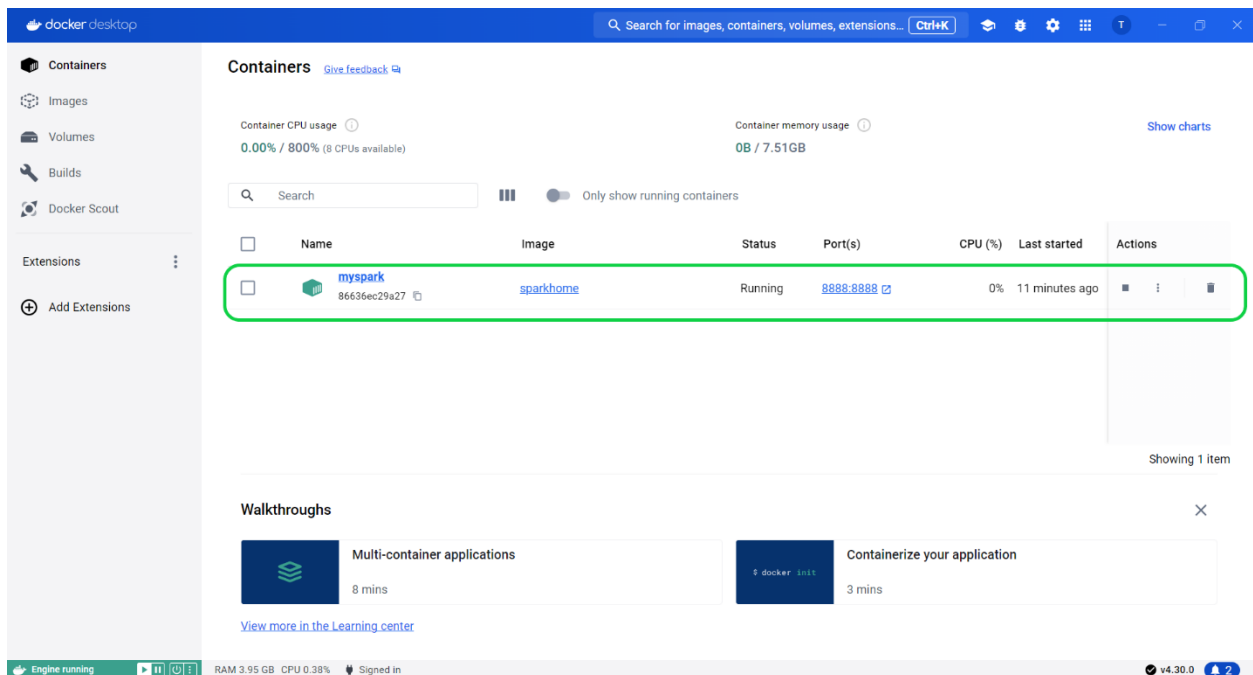
```
docker run -p 8888:8888 --name myspark -d sparkhome
```

Câu lệnh này sẽ khởi động một container Docker từ image **sparkhome**, mở cổng **8888** cho phép truy cập từ bên ngoài, đặt tên cho container là **myspark**, và chạy container ở chế độ nền.

```
PS C:\Users\Admin\Project> docker image ls
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
sparkhome     latest   98a81052f8b6   About a minute ago   3.63GB
PS C:\Users\Admin\Project> docker run -p 8888:8888 --name myspark -d sparkhome
c67f0a535e11f838908e815fee3473637c4fc2f790f21e7baabad264f03ad608
```

### Bước 4: Run container

Tại option **Container** của Docker Desktop sẽ xuất hiện một container có tên là **myspark**.



*Container trên giao diện Docker Desktop*

Trong option **logs** của container myspark sẽ có liên kết đến Jupyter lab

myspark

sparkhome

c67f0a535e11

88888888

STATUS

Running (3 minutes ago)

Logs

Inspect

Bind mounts

Exec

Files

Stats

[ 2024-06-04 15:46:43 [I 2024-06-04 08:46:43.633 ServerApp] jupyter\_server\_terminals | extension was successfully linked.

[ 2024-06-04 15:46:43 [I 2024-06-04 08:46:43.640 ServerApp] jupyterlab | extension was successfully linked.

[ 2024-06-04 15:46:43 [I 2024-06-04 08:46:43.642 ServerApp] Writing Jupyter server cookie secret to /home/sparkuser/.local/share/jupyter/runtime/jupyter\_cookie\_secret

[ 2024-06-04 15:46:43 [I 2024-06-04 08:46:43.965 ServerApp] notebook\_shim | extension was successfully linked.

[ 2024-06-04 15:46:42 Unsetting extraneous env vars (UTC): 08:46:42

[ 2024-06-04 15:46:42 Finished unsetting extraneous env vars (UTC): 08:46:42

[ 2024-06-04 15:46:42 Non-spark-on-k8s command provided, proceeding in pass-through mode...

[ 2024-06-04 15:46:44 [I 2024-06-04 08:46:44.144 ServerApp] notebook\_shim | extension was successfully loaded.

[ 2024-06-04 15:46:44 [I 2024-06-04 08:46:44.148 ServerApp] jupyter\_lsp | extension was successfully loaded.

[ 2024-06-04 15:46:44 [I 2024-06-04 08:46:44.150 ServerApp] jupyter\_server\_terminals | extension was successfully loaded.

[ 2024-06-04 15:46:44 [I 2024-06-04 08:46:44.154 LabApp] JupyterLab extension loaded from /opt/conda/lib/python3.8/site-packages/jupyterlab

[ 2024-06-04 15:46:44 [I 2024-06-04 08:46:44.154 LabApp] JupyterLab application directory is /opt/conda/share/jupyter/lab

[ 2024-06-04 15:46:44 [I 2024-06-04 08:46:44.156 LabApp] Extension Manager is 'pyp1'.

[ 2024-06-04 15:46:44 [I 2024-06-04 08:46:44.181 ServerApp] jupyterlab | extension was successfully loaded.

[ 2024-06-04 15:46:44 [I 2024-06-04 08:46:44.183 ServerApp] Serving notebooks from local directory: /myProject

[ 2024-06-04 15:46:44 [I 2024-06-04 08:46:44.184 ServerApp] Jupyter Server 2.14.1 is running at:

[ 2024-06-04 15:46:44 [I 2024-06-04 08:46:44.184 ServerApp] http://c67f0a535e11:8888/lab?token=9a585334ae27eb73cb81fa7fe1378c7d241d3054a84541db

[ 2024-06-04 15:46:44 [I 2024-06-04 08:46:44.184 ServerApp] http://127.0.0.1:8888/lab?token=9a585334ae27eb73cb81fa7fe1378c7d241d3054a84541db

[ 2024-06-04 15:46:44 [I 2024-06-04 08:46:44.184 ServerApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).

[ 2024-06-04 15:46:44 [C 2024-06-04 08:46:44.191 ServerApp]

[ 2024-06-04 15:46:44 To access the server, open this file in a browser:

[ 2024-06-04 15:46:44 file:///home/sparkuser/.local/share/jupyter/runtime/jpservice-21-open.html

[ 2024-06-04 15:46:44 Or copy and paste one of these URLs:

[ 2024-06-04 15:46:44 http://c67f0a535e11:8888/lab?token=9a585334ae27eb73cb81fa7fe1378c7d241d3054a84541db

[ 2024-06-04 15:46:44 http://127.0.0.1:8888/lab?token=9a585334ae27eb73cb81fa7fe1378c7d241d3054a84541db

[ 2024-06-04 15:46:44 [I 2024-06-04 08:46:44.249 ServerApp] Skipped non-installed server(s): bash-language-server, dockerfile-language-server-nodejs, javascript-typescript-l

[ 2024-06-04 15:46:44 angserver, jedi-language-server, julia-language-server, pyright, python-language-server, python-lsp-server, r-languageserver, sql-language-server, texlab, typescript-langu

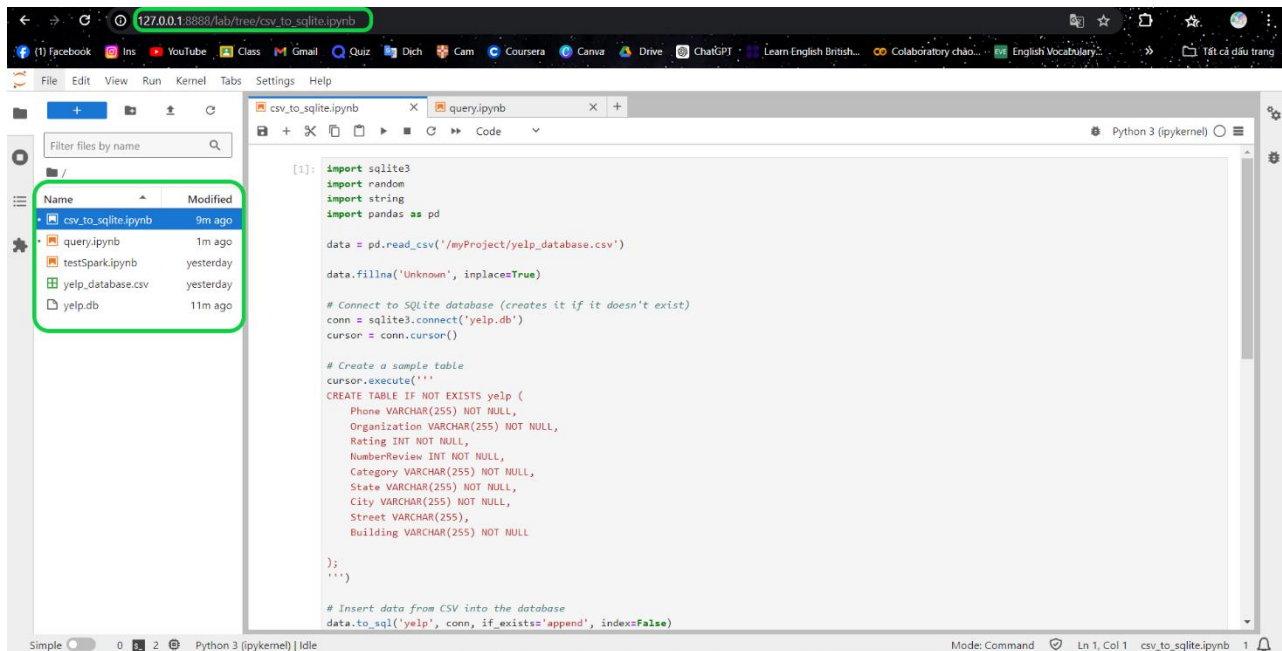
[ 2024-06-04 15:46:44 age-server, utf8-language-server, vscode-css-language-server-bin, vscode-html-language-server-bin, vscode-javascript-language-server-bin, vscode-yaml-language-server

[ 2024-06-04 15:46:52 [W 2024-06-04 08:46:52.566 LabApp] Could not determine jupyterlab build status without nodejs

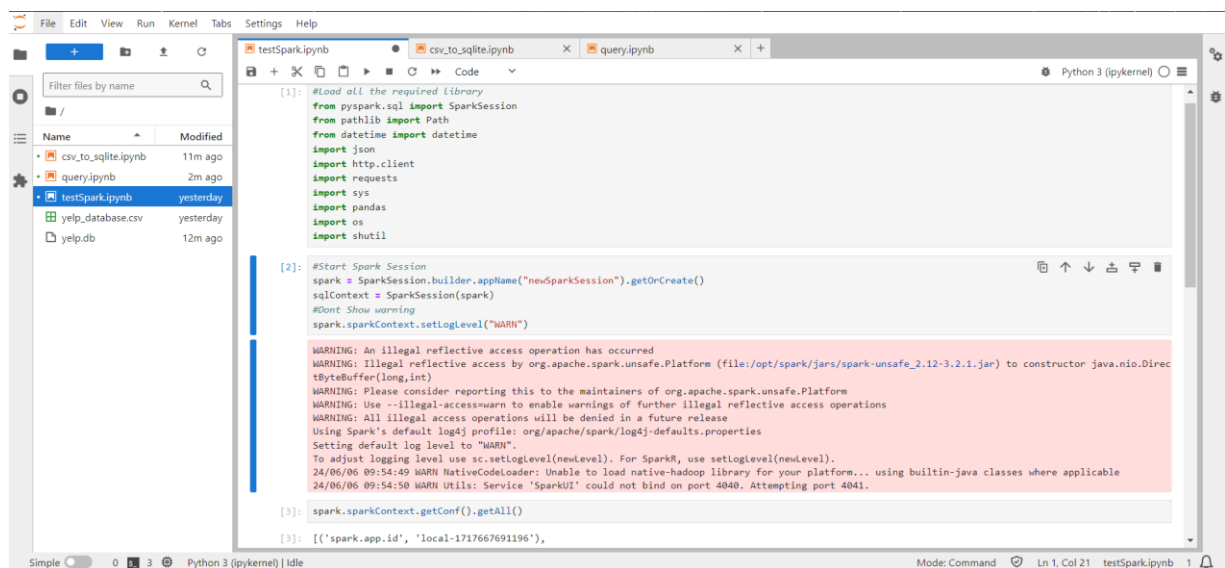
*Liên kết đến Jupyter Lab*

## Bước 5

- Sao chép và dán link vào trình duyệt để chạy phiên pyspark từ trình duyệt.



- Tại Jupyter lab chạy file `testSpark.ipynb` để kiểm tra Spark có hoạt động không.



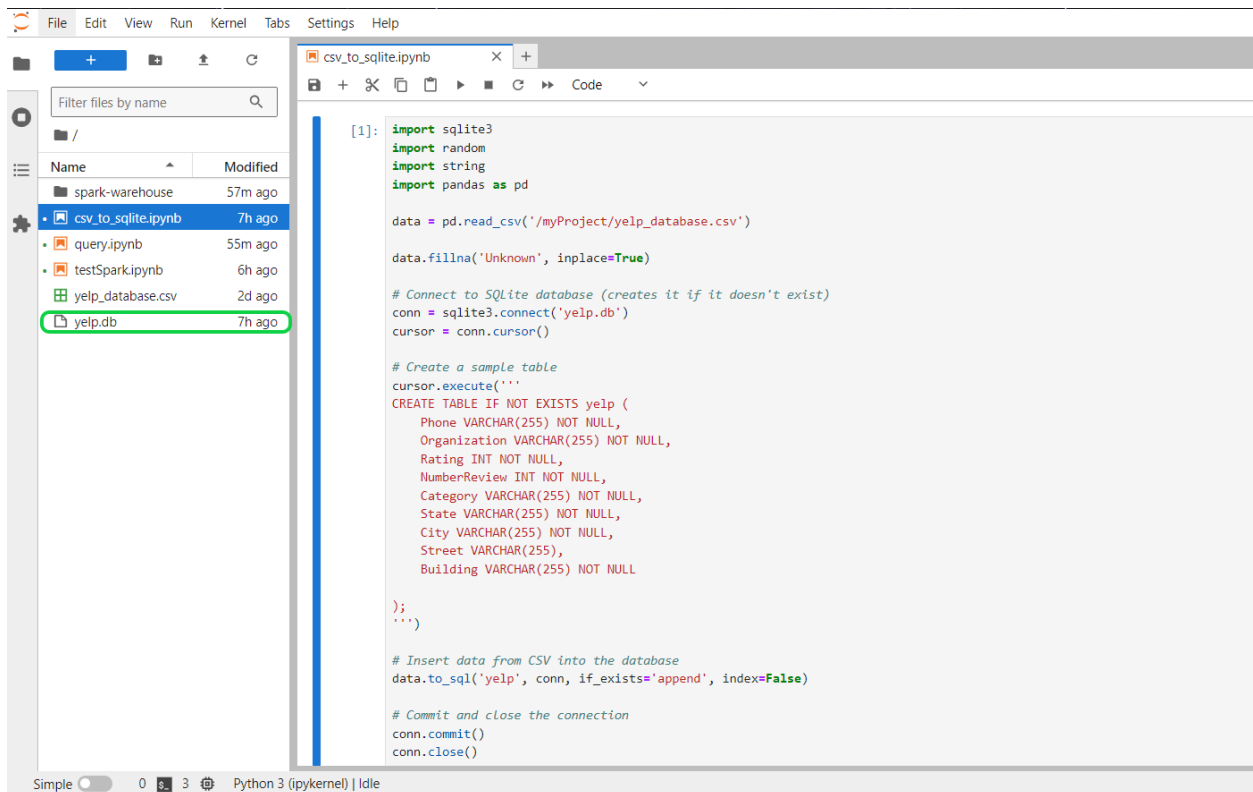
## SparkSQL

### Bước 1: Tạo cơ sở dữ liệu

File *csv-to-sqlite.ipynb* với mục đích tạo ra một cơ sở dữ liệu SQLite bằng sqlite3 và chèn dữ liệu từ một file .csv

vào cơ sở dữ liệu đó. Từ đó có thể dễ dàng thực hiện các thao tác truy vấn và phân tích dữ liệu sau này.

- Đầu tiên ta tạo một database có tên là **yelp**. Sau đó tạo một Table cũng tên là **yelp** bao gồm 9 cột: Phone, Organization, Rating, NumberReview, Category, State, City, Street, Building.
- Insert vào Table file **yelp\_database.csv** bao gồm 1000000 dòng. Đây là dataset thống kê đánh giá của dữ liệu người dùng cho các doanh nghiệp tại US.
- Chạy file **csv-to-sqlite.ipynb** để tạo tệp **yelp.db**.



The screenshot displays a Jupyter Notebook environment. On the left, a file explorer sidebar shows a directory structure with files like 'spark-warehouse', 'csv\_to\_sqlite.ipynb', 'query.ipynb', 'testSpark.ipynb', 'yelp\_database.csv', and 'yelp.db'. The 'yelp.db' file is highlighted with a green box. The main area shows the code from the 'csv\_to\_sqlite.ipynb' notebook, which is being executed. The code imports necessary libraries (sqlite3, random, string, pandas), reads a CSV file, fills missing values, connects to a SQLite database named 'yelp.db', creates a table with 9 columns (Phone, Organization, Rating, NumberReview, Category, State, City, Street, Building), and inserts data from the CSV file into this table. The status bar at the bottom indicates the environment is Python 3 (ipykernel) and the notebook is in 'Simple' mode.

```
[1]: import sqlite3
import random
import string
import pandas as pd

data = pd.read_csv('/myProject/yelp_database.csv')

data.fillna('Unknown', inplace=True)

# Connect to SQLite database (creates it if it doesn't exist)
conn = sqlite3.connect('yelp.db')
cursor = conn.cursor()

# Create a sample table
cursor.execute('''
CREATE TABLE IF NOT EXISTS yelp (
    Phone VARCHAR(255) NOT NULL,
    Organization VARCHAR(255) NOT NULL,
    Rating INT NOT NULL,
    NumberReview INT NOT NULL,
    Category VARCHAR(255) NOT NULL,
    State VARCHAR(255) NOT NULL,
    City VARCHAR(255) NOT NULL,
    Street VARCHAR(255),
    Building VARCHAR(255) NOT NULL
);
''')

# Insert data from CSV into the database
data.to_sql('yelp', conn, if_exists='append', index=False)

# Commit and close the connection
conn.commit()
conn.close()
```

## Bước 2: Kết nối với cơ sở dữ liệu

- Tạo một phiên SparkSession để làm việc với dữ liệu trong môi trường Spark.

```
query.ipynb x +
[17]: from pyspark.sql import SparkSession
      spark = SparkSession.builder \
        .appName("mySQLite") \
        .config("spark.jars.packages", "org.xerial:sqlite-jdbc:3.46.0.0") \
        .getOrCreate()
```

Đây là một phiên làm việc trong Spark, cho phép tạo DataFrame và thực hiện các truy vấn trên dữ liệu.

- Để kết nối và tương tác với các cơ sở dữ liệu từ Spark, cho phép đọc, ghi và xử lý dữ liệu từ nguồn dữ liệu SQLite, dự án sử dụng API JDBC driver 3.46.0.0

```
query.ipynb x +
[17]: from pyspark.sql import SparkSession
      spark = SparkSession.builder \
        .appName("mySQLite") \
        .config("spark.jars.packages", "org.xerial:sqlite-jdbc:3.46.0.0") \
        .getOrCreate()

[18]: # Define URL JDBC và các thuộc tính kết nối
      jdbc_url = "jdbc:sqlite:/myProject/yelp.db" # URL JDBC cho cơ sở dữ liệu SQLite
      connection_properties = {
        "driver": "org.sqlite.JDBC" # Chỉ định lớp driver cho SQLite
      }
      df = spark.read.jdbc(url=jdbc_url, table="yelp", properties=connection_properties)
```

Cơ sở dữ liệu SQLite được lưu trữ tại “/myProject/yelp.db” và bảng là **yelp**, sau đó tạo một DataFrame từ dữ liệu đó để xử lý và phân tích dữ liệu trong môi trường Spark.



df sẽ chứa dữ liệu từ bảng **yelp** trong cơ sở dữ liệu SQLite được xác định. Các truy vấn trên dữ liệu được thực hiện trong DataFrame **df**.

```
query.ipynb x +
Python 3 (ipykernel)

[20]: df.show()

+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Phone | Organization | Rating | NumberReview | Category | State | City | Street | Building |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 12562343500.0 | The Station's Best | 4 | 4 | Delivery | AL | Alexander City | Jefferson | 977 |
| 12566758004.0 | Pizza Hut | 3 | 6 | Delivery | AL | Alexander City | 4581 Hwy | 4581 |
| 12562342181.0 | Zaxby's Chicken F... | 3 | 13 | Delivery | AL | Alexander City | 4497 Highway | 4497 |
| 12564097272.0 | Papa John's Pizza | 1 | 1 | Delivery | AL | Alexander City | 2064 Cherokee | 2064 |
| 12562155510.0 | Arby's | 2 | 7 | Delivery | AL | Alexander City | 2593 Hwy | 2593 |
| 12562343500.0 | The Station's Best | 4 | 4 | Delivery | AL | Alexander City | Jefferson | 977 |
| 12566758004.0 | Pizza Hut | 3 | 6 | Delivery | AL | Alexander City | 4581 Hwy | 4581 |
| 12562342181.0 | Zaxby's Chicken F... | 3 | 13 | Delivery | AL | Alexander City | 4497 Highway | 4497 |
| 12564097272.0 | Papa John's Pizza | 1 | 1 | Delivery | AL | Alexander City | 2064 Cherokee | 2064 |
| 12562155510.0 | Arby's | 2 | 7 | Delivery | AL | Alexander City | 2593 Hwy | 2593 |
| 13342221398.0 | Happy Kitchen | 3 | 11 | Delivery | AL | Andalusia | 700 Western Gate | 700 |
| 13344271400.0 | Ophelia's Italian... | 3 | 30 | Delivery | AL | Andalusia | River Falls | 401 |
| 13342224106.0 | Pizza Hut | 1 | 3 | Delivery | AL | Andalusia | 1203 Dr Martin L... | 1203 |
| 13345828484.0 | Zaxby's Chicken F... | 3 | 12 | Delivery | AL | Andalusia | River Falls | 1000 |
| 13344278969.0 | Arby's | 2 | 5 | Delivery | AL | Andalusia | 326 Dr Mlk Jr | 326 |
| 13344278969.0 | Arby's | 1 | 3 | Delivery | AL | Andalusia | 328 W | 328 |
| 13342225005.0 | Tropical Smoothie... | 3 | 3 | Delivery | AL | Andalusia | 411 West | 411 |
| 12567707256.0 | Thai One On | 4 | 92 | Delivery | AL | Anniston | Noble | 911 |
| 12562360305.0 | House Of Chen | 4 | 10 | Delivery | AL | Anniston | E 43rd | 4 |
| 12568319981.0 | Jasmine Chinese T... | 4 | 32 | Delivery | AL | Oxford | Snow | 1225 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
only showing top 20 rows

[21]: df.count()

1000000
```

### Bước 3: Thực hiện truy vấn SQL bằng SparkSQL

- Thao tác tạo

```
Create operation

from pyspark.sql import Row

# Tạo một hàng dữ liệu mới
new_row = Row(Phone = 1284680203, Organization = 'ChickenGarden', Rating = 3.0, NumberReview = 100, Category = 'Chicken', State = 'AL',
# Tạo DataFrame mới (new_df) chỉ có new_row
new_df = spark.createDataFrame([new_row], schema=df.schema)

df = df.union(new_df)

df.tail(3)
df.count()

1000001
```

- Thao tác xóa

#### Delete Operation

```
df = df.filter(~((df.Phone == 13342221398) &
                 (df.Organization == 'Happy Kitchen') &
                 (df.Rating == 3.5) &
                 (df.NumberReview == 11) &
                 (df.Category == 'Delivery') &
                 (df.State == 'AL') &
                 (df.City == 'Andalusia') &
                 (df.Street == ' 700 Western Gate') &
                 (df.Building == '700'))))

print(df.count())
df.head(12)
#Row 12
```

999999

- Thao tác cập nhật

#### Update operation

```
[29]: from pyspark.sql.functions import when

# Update the DataFrame to change the NumberReview columns
df = df.withColumn('NumberReview', when((df.Organization == 'Pizza Hut') & (df.City == 'Alexander City') & (df.Rating == 3), 7).otherwise(df.NumberReview))

df.head(10)
```

[29]: [Row(Phone='12562343500.0', Organization='The Station's Best', Rating=4, NumberReview=4, Category='Delivery', State='AL', City='Alexander City', Street='Jefferson', Building='977'),  
Row(Phone='12566758004.0', Organization='Pizza Hut', Rating=3, NumberReview=7, Category='Delivery', State='AL', City='Alexander City', Street='4581 Highway', Building='4581'),  
Row(Phone='12562342181.0', Organization='Zaxby's Chicken Fingers & Buffalo Wings', Rating=3, NumberReview=13, Category='Delivery', State='AL', City='Alexander City', Street='4497 Highway', Building='4497'),  
Row(Phone='12564097272.0', Organization='Papa John's Pizza', Rating=1, NumberReview=1, Category='Delivery', State='AL', City='Alexander City', Street='2064 Cherokee', Building='2064'),  
Row(Phone='12562155510.0', Organization='Arby's', Rating=2, NumberReview=7, Category='Delivery', State='AL', City='Alexander City', Street='2593 Highway', Building='2593'),  
Row(Phone='12562343500.0', Organization='The Station's Best', Rating=4, NumberReview=4, Category='Delivery', State='AL', City='Alexander City', Street='Jefferson', Building='977'),  
Row(Phone='12566758004.0', Organization='Pizza Hut', Rating=3, NumberReview=7, Category='Delivery', State='AL', City='Alexander City', Street='4581 Highway', Building='4581'),  
Row(Phone='12562342181.0', Organization='Zaxby's Chicken Fingers & Buffalo Wings', Rating=3, NumberReview=13, Category='Delivery', State='AL', City='Alexander City', Street='4497 Highway', Building='4497')]

- Thao tác đọc

Read operation

```
[30]: filtered_df = df.filter(df.Rating > 4)
```

```
# Show the filtered DataFrame
filtered_df.show()
print(filtered_df.count())
```

Phone	Organization	Rating	NumberReview	Category	State	City	Street	Building
12568483131.0	Marco's Pizza	4	3	Delivery	AL	Anniston	2485 US Hwy 431	2485
12563652001.0	Jimmy John's	4	2	Delivery	AL	Jacksonville	505 Pelham Rd	505
12564357272.0	Papa John's Pizza	4	2	Delivery	AL	Jacksonville	702 South Pelham	702
12564444005.0	Ramona Js Resturant	5	6	Delivery	AL	Athens	1212 US Hwy 31	1212
12562323000.0	Papa John's Pizza	4	3	Delivery	AL	Athens	916 Us Highway 72	916
12562161099.0	Papa Murphy's	5	1	Delivery	AL	Athens	1001 Hwy 72	1001
12568581600.0	Al Shish Palace	4	56	Delivery	AL	Madison	1591 Hughes	1591
12567215527.0	Subway	4	3	Delivery	AL	Madison	8580 Highway 72	8580
12518674664.0	Arby's	5	1	Delivery	AL	Brewton	380 South	380
18505424316.0	Thai Rice	4	9	Delivery	FL	Pensacola	7175 N Davis	7175
12519900995.0	Gambino's Italian...	4	305	Delivery	AL	Fairhope	18 Laurel	18
18504774424.0	Sky's Pizza Pie	4	313	Delivery	FL	Pensacola	5559 N Davis	5559
12516256550.0	Marco's Pizza	4	20	Delivery	AL	Daphne	2004 US Hwy	2004
12514385234.0	219	4	56	Delivery	AL	Mobile	Conti	219
12512642520.0	Buster's Brick Oven	4	16	Delivery	AL	Daphne	Main	1711
12515177536.0	Sage Lebanese Cui...	5	133	Delivery	AL	Fairhope	319 Fairhope	319
12517256912.0	Iron Hand Brewing	4	27	Delivery	AL	Mobile	State	206
12516264065.0	Roll & Go Sushi	5	1	Delivery	AL	Daphne	1410 US Hwy	1410

## Bước 4: So sánh hiệu suất truy vấn liên quan đến WHERE

- Khi không có WHERE

```
[65]: %time
df.createOrReplaceTempView("yelp")
spark.sql("SELECT * FROM yelp").show(5)
```

Phone	Organization	Rating	NumberReview	Category	State	City	Street	Building
12562343500	The Station's Best	4.0	4	Delivery	AL	Alexander City	Jefferson	977
12566758004	Pizza Hut	3.0	7	Delivery	AL	Alexander City	4581 Hwy	4581
12562342181	Zaxby's Chicken F...	3.0	13	Delivery	AL	Alexander City	4497 Highway	4497
12564097272	Papa John's Pizza	1.0	1	Delivery	AL	Alexander City	2064 Cherokee	2064
12562155510	Arby's	2.0	7	Delivery	AL	Alexander City	2593 Hwy	2593

only showing top 5 rows

CPU times: user 4.54 ms, sys: 1.63 ms, total: 6.17 ms  
Wall time: 237 ms

- Khi có WHERE

```
[66]: %%time
spark.sql("SELECT * FROM yelp WHERE Rating > 4").show(5)
```

Phone	Organization	Rating	NumberReview	Category	State	City	Street	Building
12568483131	Marco's Pizza	4.5	3	Delivery	AL	Anniston	2485 US Hwy 431	2485
12563652001	Jimmy John's	4.5	2	Delivery	AL	Jacksonville	505 Pelham Rd	505
12564357272	Papa John's Pizza	4.5	2	Delivery	AL	Jacksonville	702 South Pelham	702
12564444005	Ramona Js Resturant	5.0	6	Delivery	AL	Athens	1212 US Hwy 31	1212
12562323000	Papa John's Pizza	4.5	3	Delivery	AL	Athens	916 Us Highway 72	916

only showing top 5 rows

CPU times: user 0 ns, sys: 5.01 ms, total: 5.01 ms  
Wall time: 151 ms

Mệnh đề WHERE giúp cơ sở dữ liệu làm việc hiệu quả hơn bằng cách giảm lượng dữ liệu cần xử lý, tận dụng chỉ mục và giảm tải hệ thống. Điều này dẫn đến thời gian thực thi nhanh hơn so với truy vấn không có WHERE.

- Không có WHERE, có GROUP BY

```
[69]: %%time
spark.sql("SELECT City, SUM(NumberReview) as TotalNumberReview FROM yelp GROUP BY City").show()
```

[Stage 157:=====> (8 + 1) / 9]

City	TotalNumberReview
Jemison	25
Prattville	1093
Fairbanks	2334
Rancho Bernardo	177
Santa Paula	1322
Agawam	1625
Kilauea	180
Antwerp	7
Minster	27
Bluffton	380
Hanover	9924
Grimes	672
Fredonia	264
Nahant	600
Worcester	12847
North Saint Paul	14
Palermo	56
Westampton	618
West Sand Lake	104
Rhinebeck	640

only showing top 20 rows

CPU times: user 6.51 ms, sys: 2.36 ms, total: 8.87 ms

- Có WHERE, có GROUP BY

```
67]: %%time
spark.sql("SELECT City, SUM(NumberReview) as TotalNumberReview FROM yelp WHERE Rating > 4 GROUP BY City").show()

# Hiển thị tổng số lượt đánh giá (TotalNumberReview) cho mỗi City có Rating lớn hơn 4.
```

City	TotalNumberReview
Prattville	155
Fairbanks	588
Santa Paula	524
Agawam	616
Kilauea	167
Minster	6
Grimes	32
Hanover	814
Worcester	1968
Palermo	37
Fredonia	89
Rhinebeck	306
Johnsonburg	30
Tyler	200
Saint George	303
Jemison	11
Hanceville	4
Moreland	7
Santee Cal	9
Bluffton	28

only showing top 20 rows

CPU times: user 5.12 ms, sys: 1.85 ms, total: 6.97 ms

Mệnh đề WHERE khi kết hợp với GROUP BY giúp tăng tốc độ thực thi truy vấn bằng cách giảm số lượng bản ghi cần nhóm, tối ưu hóa kế hoạch thực thi và giảm tải hệ thống.

Nếu không có mệnh đề WHERE, cơ sở dữ liệu phải xử lý tất cả các dữ liệu trong bảng trước khi thực hiện việc nhóm, điều này sẽ tốn nhiều thời gian và tài nguyên hơn, đặc biệt nếu bảng có rất nhiều dữ liệu.

- Khi có HAVING

```
[68]: %%time
spark.sql("SELECT City, SUM(NumberReview) as TotalNumberReview FROM yelp WHERE Rating > 4 GROUP BY City HAVING TotalNumberReview > 100").show()

# Thực hiện câu lệnh SQL để tính tổng giá trị (SUM(NumberReview)) của các hàng có Rating lớn hơn 4
# Nhóm kết quả theo City, và chỉ hiển thị các nhóm có SUM(NumberReview) lớn hơn 100.
```

[Stage 154:=====> (8 + 1) / 9]

City	TotalNumberReview
Prattville	155
Fairbanks	588
Santa Paula	524
Agawam	616
Kilauea	167
Hanover	814
Worcester	1968
Rhinebeck	306
Tyler	200
Saint George	303
Nahant	420
Tempe	50837
Corona	52141
Leucadia	544
Marcus Hook	110
Springfield	12637
Charleston	11240
Bowling Green	827
Truckee	2118
Oakhurst	120

only showing top 20 rows

CPU times: user 9.06 ms, sys: 0 ns, total: 9.06 ms

Khi sử dụng cả WHERE và HAVING trong một truy vấn, mệnh đề WHERE sẽ lọc các hàng dữ liệu có Rating > 4 trước, sau đó GROUP BY nhóm các dữ liệu theo City và cuối cùng, mệnh đề HAVING sẽ áp dụng điều kiện trên các nhóm này.

Việc kiểm tra điều kiện HAVING trên mỗi nhóm sau khi nhóm đã được tạo ra làm tăng thêm bước xử lý, kéo dài thời gian thực thi.

Tuy nhiên, việc có WHERE đã giúp tối ưu hiệu suất truy vấn hơn rất nhiều.