

Họ và tên : Trịnh Quang Vinh

MSV : 21002244

Lớp : K66 -Kĩ thuật điện tử và tin học

## **BỘ MÔN THỰC HÀNH TIN HỌC ỨNG DỤNG**

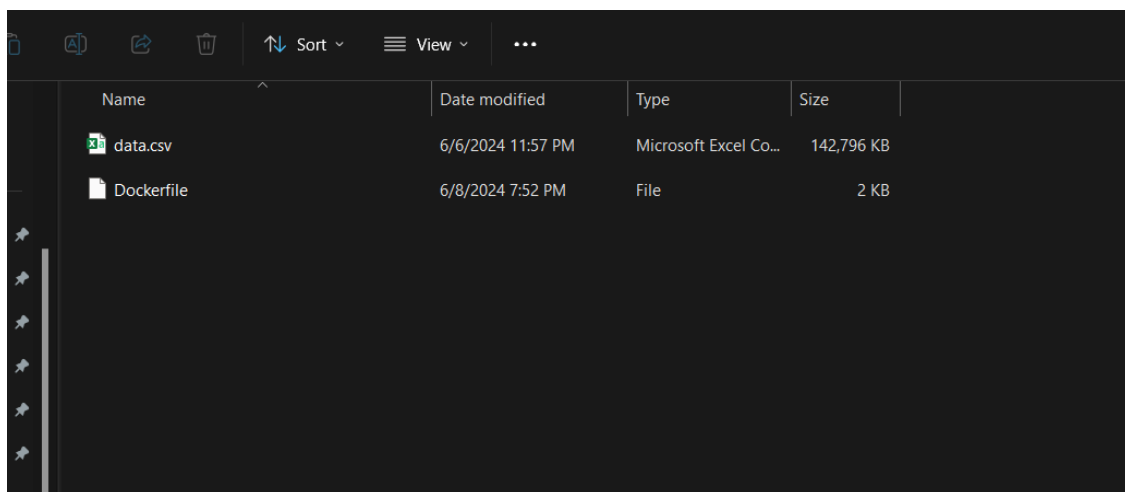
### **PROJECT HẾT HỌC PHẦN**

#### **1.Docker setup:**

- Install docker desktop lên máy tính
- Tạo 1 folder có tên là prj .
- Tải data với số cột là > 1000000 dòng lưu vào folder prj
- Khởi tạo 1 file có tên là Dockerfile , và thực hiện ghi docker file đã tạo

#Cấu hình Images gốc của container :

```
FROM ubuntu:20.04
```



Ảnh 1: Thư mục lưu trữ

- Lưu file .Sau đó mở ứng dụng “**Terminal**” trên máy tính .Đưa đường dẫn đến đúng folder của project đang làm

```

root@cee75f88c0ee: /home
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\vinh5> D:
PS D:\> cd prj
PS D:\prj>

```

Ảnh 2: Chạy trên terminal

- Tạo một images bằng lệnh: "**docker build -t spark-project .**"
- Sau khi build xong image thì ta thực hiện lệnh để chạy container : "**docker run -it lab-final bash**"

```

PS D:\prj> docker build -t spark_project .
[+] Building 2.4s (16/16) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 2.01kB
=> [internal] load metadata for docker.io/library/ubuntu:20.04
=> [auth] library/ubuntu:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load build context
=> => transferring context: 32B
=> [ 1/10] FROM docker.io/library/ubuntu:20.04@sha256:0b897358ff6624825fb50d20fffb605ab0eaea77ced0adb8c6a4b756513
=> CACHED [ 2/10] RUN apt-get update && apt-get install -y openjdk-8-jdk wget tar curl pytho
=> CACHED [ 3/10] RUN wget https://d1cdn.apache.org/spark/spark-3.5.1/spark-3.5.1-bin-hadoop3.tgz
=> CACHED [ 4/10] RUN tar xvf spark-3.5.1-bin-hadoop3.tgz
=> CACHED [ 5/10] RUN mv spark-3.5.1-bin-hadoop3 /opt/spark
=> CACHED [ 6/10] RUN rm spark-3.5.1-bin-hadoop3.tgz
=> CACHED [ 7/10] RUN wget -P /opt/spark/jars https://repo1.maven.org/maven2/org/xerial/sqlite-jdbc/3.34.0/sqlit
=> CACHED [ 8/10] RUN pip install pyspark
=> CACHED [ 9/10] COPY data.csv /home/data.csv
=> CACHED [10/10] WORKDIR /home
=> exporting to image
=> => exporting layers
=> => writing image sha256:af9b67d82cb77db35943b57f3affab084ba45ffbcd2d4eb1c216b49b38b5686e
=> => naming to docker.io/library/spark_project

View build details: docker-desktop://dashboard/build/default/default/i3is2h1bfa4rsqslcety6mwk9

What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview
PS D:\prj>

```

Ảnh 3: build image

```

cee/5f88c0ee
PS D:\prj> docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS        NAMES
00961eca6112   spark_project "bash"                 3 minutes ago Up 3 minutes             flamboyant_haibt
cee75f88c0ee   spark_project "/bin/bash"           17 hours ago  Up About an hour         keen_heisenberg
PS D:\prj>

```

Ảnh 4: check container

```
View a summary of image vulnerabilities and recommendations → docker scout quickview  
PS D:\prj> docker run -it spark_project bash  
root@00961eca6112:/home# |
```

Ảnh 5: Chạy Container

## 2. Install Spark

- 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 1 và nhập các lệnh sau:

```
ENV DEBIAN_FRONTEND=noninteractive  
RUN apt-get update && apt-get install -y \  
    openjdk-8-jdk \  
    wget \  
    tar \  
    curl \  
    python3-pip \  
    sqlite3 \  
    nano \  
&& ln -s /usr/bin/python3 /usr/bin/python \  
&& apt-get clean \  
&& rm -rf /var/lib/apt/lists/*  
# Tải xuống và cài đặt Apache Spark  
RUN wget https://dlcdn.apache.org/spark/spark-3.5.1/spark-3.5.1-bin-hadoop3.tgz  
RUN tar xvf spark-3.5.1-bin-hadoop3.tgz  
RUN mv spark-3.5.1-bin-hadoop3 /opt/spark  
RUN rm spark-3.5.1-bin-hadoop3.tgz  
# Tải xuống JDBC driver cho SQLite  
RUN wget -P $SPARK_HOME/jars  
https://repo1.maven.org/maven2/org/xerial/sqlite-jdbc/3.34.0/sqlite-jdbc-3.34.0.jar  
#pyspark  
RUN pip install pyspark
```

- Phiên bản Spark sử dụng trong bài này là 3.5.1 đi cùng với Hadoop phiên bản 3.0.  
- Sau khi đã lựa chọn các công cụ cần thiết, giờ ta cần cấu hình đường dẫn chạy cho Spark vào trong Dockerfile

```
ENV SPARK_HOME=/opt/spark  
ENV PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin:$PATH
```

```
Dockerfile • data.csv
Dockerfile > ...
1 FROM ubuntu:20.04
2 # Cài đặt các gói cần thiết
3 ENV DEBIAN_FRONTEND=noninteractive
4 RUN apt-get update && apt-get install -y \
5     openjdk-8-jdk \
6     wget \
7     tar \
8     curl \
9     python3-pip \
10    sqlite3 \
11    vim \
12    nano \
13    && ln -s /usr/bin/python3 /usr/bin/python \
14    && apt-get clean \
15    && rm -rf /var/lib/apt/lists/*
16 # Tải xuống và cài đặt Apache Spark
17 RUN wget https://d1cdn.apache.org/spark/spark-3.5.1/spark-3.5.1-bin-hadoop3.tgz
18 RUN tar xvf spark-3.5.1-bin-hadoop3.tgz
19 RUN mv spark-3.5.1-bin-hadoop3 /opt/spark
20 RUN rm spark-3.5.1-bin-hadoop3.tgz
21
22 # Thiết lập biến môi trường cho Spark
23 ENV SPARK_HOME=/opt/spark
24 ENV PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin:$PATH
25
26 # Tải xuống JDBC driver cho SQLite
27 RUN wget -P $SPARK_HOME/jars https://repo1.maven.org/maven2/org/xerial/sqlite-jdbc/3.34.0/sqlite-jdbc-3.34.0.jar
28 #pyspark
29 RUN pip install pyspark
30
31 COPY data.csv /home/data.csv
32 # Thiết lập thư mục làm việc mặc định
33 WORKDIR /home
```

ảnh 6: Cấu hình dockerfile

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

```
PS C:\Users\vinh5> docker start cee75f88c0ee
cee75f88c0ee
PS C:\Users\vinh5> docker exec -it cee75f88c0ee bash
root@cee75f88c0ee:/home# pyspark
Python 3.8.10 (default, Nov 22 2023, 10:22:35)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
24/06/08 15:23:45 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
Welcome to

  ____      _
 / ___|  __| | | |
 \___ \  | | | | | |
  ___) | | | | | | |
 |_____|_|_|_|_|_|_|

 version 3.5.1

Using Python version 3.8.10 (default, Nov 22 2023 10:22:35)
Spark context Web UI available at http://cee75f88c0ee:4040
Spark context available as 'sc' (master = local[*], app id = local-1717860226718).
SparkSession available as 'spark'.
>>>
```

ảnh 7: Khởi động spark

### 3. Database Setup

Ta đã có những câu lệnh cần thiết trong Dockerfile để setup database

```

RUN apt-get update && apt-get install -y \
  openjdk-8-jdk \
  wget \
  tar \
  curl \
  python3-pip \
  sqlite3 \
  nano \
  && ln -s /usr/bin/python3 /usr/bin/python \
  && apt-get clean \
  && rm -rf /var/lib/apt/lists/*

```

- Sau khi build xong image, thực hiện chạy container bằng lệnh “**docker run -it spark-project**”.

- Sau khi chạy container, tạo và chạy cơ sở dữ liệu bằng lệnh sau:

“**sqlite3 /home/database.db**”

- Giao diện dòng lệnh trong database sẽ hiện lên để tương tác với cơ sở dữ liệu.

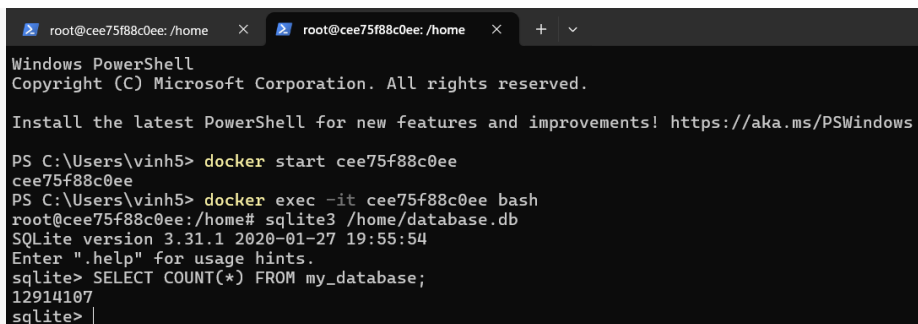
Lúc này thực hiện 2 dòng lệnh để đưa file data.csv vào bảng trong cơ sở dữ liệu:

**.mode csv**

**.import /opt/spark-3.5.1-bin-hadoop3/home/data.csv my\_database**  
(đưa file data.csv vào bảng có tên là **my\_database** )

- Thực hiện kiểm tra số lượng hàng trong bảng bằng lệnh sau:

**SELECT COUNT(\*) FROM my\_database**



```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\vinh5> docker start cee75f88c0ee
cee75f88c0ee
PS C:\Users\vinh5> docker exec -it cee75f88c0ee bash
root@cee75f88c0ee:/home# sqlite3 /home/database.db
SQLite version 3.31.1 2020-01-27 19:55:54
Enter ".help" for usage hints.
sqlite> SELECT COUNT(*) FROM my_database;
12914107
sqlite>

```

ảnh 8: Kiểm tra số dòng dữ liệu (12.914.107 dòng)

#### 4. 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>

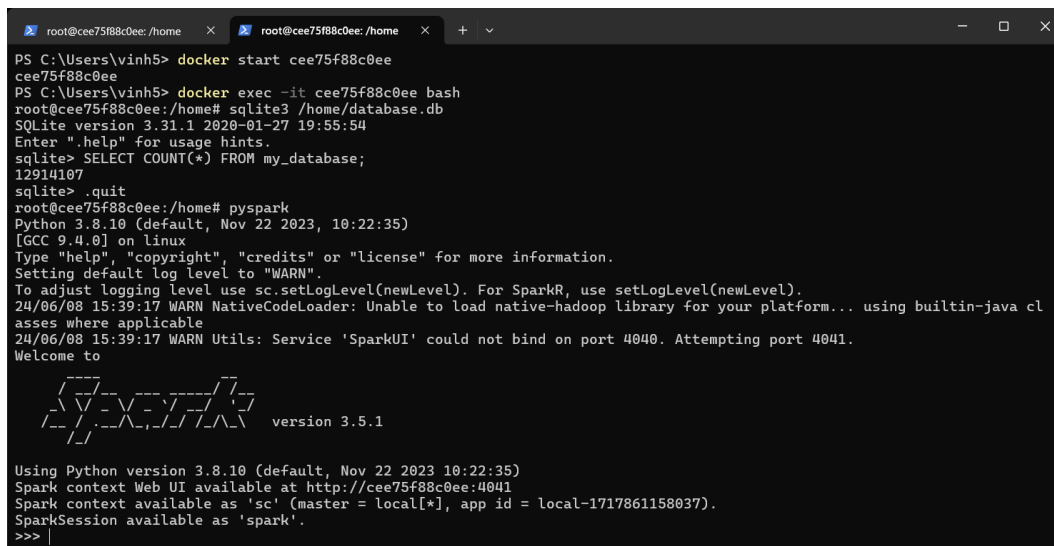
## 5.Configuration:

- Để Spark có thể kết nối với cơ sở dữ liệu, ta cần chuyển JDBC vào thư mục jars trong spark, thì ta đã thực hiện nó trong dockerfile bằng câu lệnh:

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

- Sau khi di chuyển hoàn tất, ta có thể thử truy cập và tương tác với cơ sở dữ liệu bằng PySpark

- Tại giao diện terminal, nhập "**pyspark**" để khởi chạy, lúc này một phiên làm việc của Spark thông qua giao diện dòng lệnh sẽ bắt đầu, nó cung cấp một môi trường tương tác với Spark bằng Python.



```
PS C:\Users\vinh5> docker start cee75f88c0ee
cee75f88c0ee
PS C:\Users\vinh5> docker exec -it cee75f88c0ee bash
root@cee75f88c0ee:/home# sqlite3 /home/database.db
SQLite version 3.31.1 2020-01-27 19:55:54
Enter ".help" for usage hints.
sqlite> SELECT COUNT(*) FROM my_database;
12914107
sqlite> .quit
root@cee75f88c0ee:/home# pyspark
Python 3.8.10 (default, Nov 22 2023, 10:22:35)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
24/06/08 15:39:17 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
24/06/08 15:39:17 WARN Utils: Service 'SparkUI' could not bind on port 4040. Attempting port 4041.
Welcome to

  ____      _
 / ___|  __| | | |
 \___ \  | | | | | |
  ___) | | | | | | |
 |____|_|_|_|_|_|_|_|

version 3.5.1

Using Python version 3.8.10 (default, Nov 22 2023 10:22:35)
Spark context Web UI available at http://cee75f88c0ee:4041
Spark context available as 'sc' (master = local[*], app id = local-1717861158037).
SparkSession available as 'spark'.
>>>
```

- Thử nghiệm kết nối đến cơ sở dữ liệu database.db, bảng my\_database, hiển thị 10 dòng đầu tiên. Ta sử dụng câu lệnh để kết nối spark tới cơ sở dữ liệu và thực hiện đọc dữ liệu bảng trong cơ sở dữ liệu:

```
df=spark.read.format("jdbc").option("url",f"jdbc:sqlite:/home/databas
e.db").option("dbtable","my_database").load()
df.show(10)
```

```
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
at java.lang.reflect.Method.invoke(Method.java:498)
at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)
at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:374)
at py4j.Gateway.invoke(Gateway.java:282)
at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)
at py4j.commands.CallCommand.execute(CallCommand.java:79)
at py4j.ClientServerConnection.waitForCommands(ClientServerConnection.java:182)
at py4j.ClientServerConnection.run(ClientServerConnection.java:106)
at java.lang.Thread.run(Thread.java:750)

>>> df=spark.read.format("jdbc").option("url",f"jdbc:sqlite:/home/database.db").option("dbtable","my_database").load()
>>> df.show(10)
+-----+-----+-----+-----+-----+
|anzsic06|Area|year|geo_count|ec_count|
+-----+-----+-----+-----+
|A|A100100|2023|90|160|
|A|A100200|2023|138|180|
|A|A100301|2023|6|40|
|A|A100400|2023|57|45|
|A|A100500|2023|54|95|
|A|A100600|2023|9|18|
|A|A100700|2023|12|30|
|A|A100800|2023|21|75|
|A|A100900|2023|54|25|
|A|A101000|2023|48|20|
+-----+-----+-----+-----+
only showing top 10 rows
>>> |
```

## 6. Running Queries

- Đầu tiên, ta sẽ viết file nội dung để thực hiện các thao tác với cơ sở dữ liệu. Tạo file exam1.py với câu lệnh “**nano exam1.py**” để thực hiện ghi dòng code vào trong file .py đó và thực hiện ghi dòng lệnh sau để khởi tạo Spark , đọc dữ liệu từ bảng và thực hiện các thao tác cơ bản với database:

```
GNU nano 4.8 exam1.py Modified
from pyspark.sql import SparkSession
import time

#Initializes a SparkSession object.
spark = SparkSession.builder \
    .appName("SparkSQL CRUD Operations") \
    .config("spark.some.config.option", "some-value") \
    .getOrCreate()

#Read Data from SQLite Database
df = spark.read.format("jdbc").option("url", f"jdbc:sqlite:/home/database.db").option("dbtable", "my_database").load()

#Create Temporary View and Execute Query
df.createOrReplaceTempView("my_database")

time_start = time.time()

#Executes the SQL query
result = spark.sql("select * from my_database ")

time_end = time.time()

result.show(10)

print("run time: ",time_end - time_start)

^G Get Help      ^O Write Out    ^W Where Is     ^K Cut Text     ^J Justify      ^C Cur Pos      M-U Undo        M-A Mark Text
^X Exit          ^R Read File    ^\ Replace      ^U Paste Text   ^T To Spell     ^_ Go To Line    M-E Redo        M-6 Copy Text
```

- Kết quả in ra : thời gian chạy chỉ mất 0.1s

```
root@cee75f88c0ee: /home
Traceback (most recent call last):
  File "exam1.py", line 12, in <module>
    df.CreateOrReplaceTempView(table_name)
  File "/usr/local/lib/python3.8/dist-packages/pyspark/sql/dataframe.py", line 3127, in __getattr__
    raise AttributeError(
AttributeError: 'DataFrame' object has no attribute 'CreateOrReplaceTempView'
root@cee75f88c0ee:/home# nano exam1.py
root@cee75f88c0ee:/home# python3 exam1.py
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
24/06/09 04:57:48 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
+-----+-----+-----+-----+
|anzsic06| Area|year|geo_count|ec_count|
+-----+-----+-----+-----+
|A|A100100|2023|90|160|
|A|A100200|2023|138|180|
|A|A100301|2023|6|40|
|A|A100400|2023|57|45|
|A|A100500|2023|54|95|
|A|A100600|2023|9|18|
|A|A100700|2023|12|30|
|A|A100800|2023|21|75|
|A|A100900|2023|54|25|
|A|A101000|2023|48|20|
+-----+-----+-----+-----+
only showing top 10 rows

run time: 0.10777878761291504
root@cee75f88c0ee:/home#
```

## CREATE

Trong SparkSQL, tạo dữ liệu mới có thể thực hiện bằng cách chèn các hàng mới vào DataFrame và sau đó ghi lại vào cơ sở dữ liệu.

```
GNU nano 4.8 exam1.py Modified
spark = SparkSession.builder \
    .appName("SparkSQL CRUD Operations") \
    .config("spark.some.config.option", "some-value") \
    .getOrCreate()

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

#CREATE
new_data = [("A", "B10000", 2024, 234, 232), ("C", "A10000", 2024, 214, 132)]
columns = ['anzsic06', 'Area', 'year', 'geo_count', 'ec_count']
new_df = spark.createDataFrame(new_data, columns)
df_new = df.union(new_df)

#Create Tempnew View and Execute Query
```

## READ

Ở phần này, ta thực hiện chọn tất cả thành phần trong bảng và in ra

```
time_start = time.time()
#READ
result = spark.sql("select * from my_database ")
time_end = time.time()
result.show(10)
```



## UPDATE, DELETE:

Spark không hỗ trợ UPDATE và DELETE trực tiếp trong DataFrame. Thay vào đó, có thể tạo một DataFrame mới với các giá trị đã cập nhật và ghi đè bảng cũ.

```
time_start = time.time()
#READ
result = spark.sql("select Area,year,geo_count from my_database where geo_count > 300")
time_end = time.time()
result.show()

print("run time: ",time_end - time_start)
```

```
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
24/06/09 09:34:53 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
+-----+-----+-----+
| Area|year|geo_count|
+-----+-----+-----+
|A109101|2023|306|
|A190501|2023|393|
|A192000|2023|399|
|A192500|2023|372|
|A220501|2023|351|
|A306600|2023|402|
|A358300|2023|321|
|A358400|2023|345|
|C07601|2023|1419|
|C07621|2023|1377|
|CTotal|2023|4044|
|R01|2023|4800|
|R02|2023|4044|
|R03|2023|11583|
|R04|2023|5991|
|R05|2023|1314|
|R06|2023|3147|
|R07|2023|3867|
|R08|2023|5535|
|R09|2023|2115|
+-----+-----+-----+
only showing top 20 rows
run time: 0.11569547653198242
```

Thời gian chạy : 0.1156

- Ta tìm hiểu xem WHERE ảnh hưởng tới hiệu năng của một số truy vấn như thế nào . Sự khác biệt 2 file là điều kiện **WHERE geo\_count > 300** để xem sự ảnh hưởng của **WHERE** sẽ thế nào đến truy vấn. Đầu tiên thử với truy vấn sau:

```
root@cee75f88c0ee: /home  GNU nano 4.8  exam1.py
from pyspark.sql import SparkSession
import time

#Initializes a SparkSession object.
spark = SparkSession.builder \
    .appName("SparkSQL CRUD Operations") \
    .config("spark.some.config.option", "some-value") \
    .getOrCreate()

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

time_start = time.time()
result = spark.sql("select * from my_database where geo_count > 300")
time_end = time.time()
print("run time: ",time_end - time_start)
```

[ Read 16 lines ]

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos M-U Undo M-A Mark Text  
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell ^\_ Go To Line M-E Redo M-G Copy Text

-Thời gian khi sử dụng where :

```
root@cee75f88c0ee:/home# nano exam1.py
root@cee75f88c0ee:/home# python3 exam1.py
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
24/06/09 17:10:21 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
run time:  0.10000348091125488
root@cee75f88c0ee:/home# nano exam1.py
```

```
Windows PowerShell  root@cee75f88c0ee: /home  GNU nano 4.8  exam1.py  Modified
spark = SparkSession.builder \
    .appName("SparkSQL CRUD Operations") \
    .config("spark.some.config.option", "some-value") \
    .getOrCreate()

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

#CREATE
new_data = [(("A","B10000",2024,234,232),("C","A10000",2024,214,132))]
columns = ['anzsic06', 'Area', 'year', 'geo_count','ec_count']
new_df = spark.createDataFrame(new_data, columns)
df_new = df.union(new_df)

#Create Temporary View and Execute Query
df_new.createOrReplaceTempView("my_database")

time_start = time.time()
#READ
result = spark.sql("select * from my_database ")
time_end = time.time()
result.show(10)

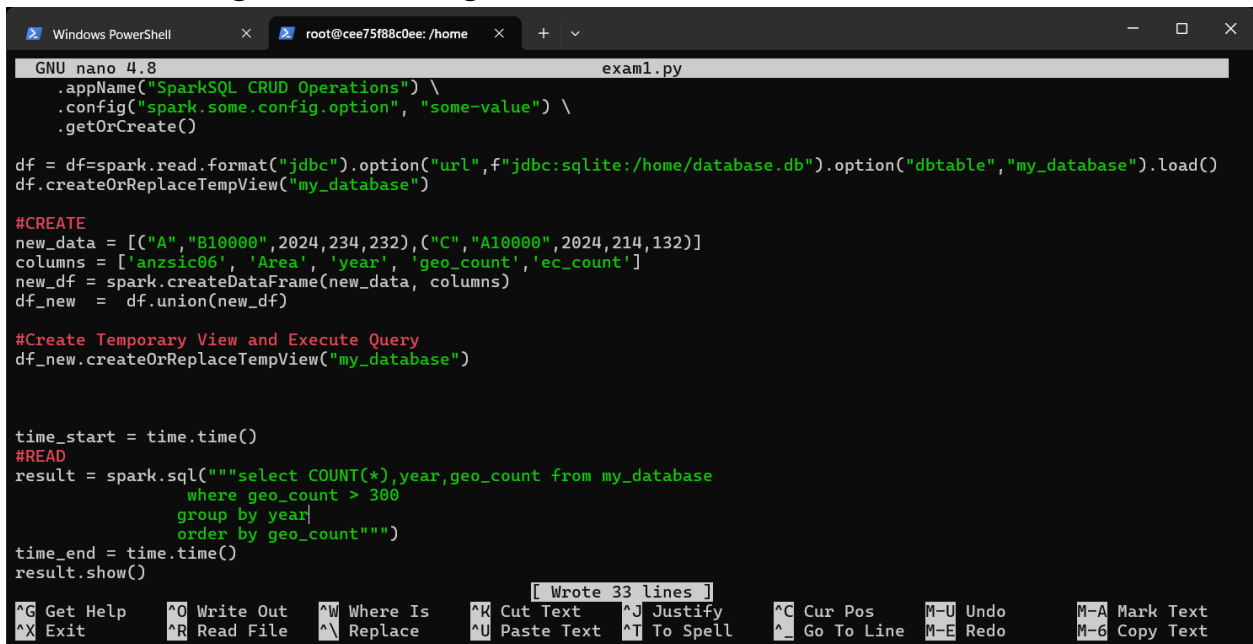
print("run time: ",time_end - time_start)
```

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos M-U Undo M-A Mark Text  
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell ^\_ Go To Line M-E Redo M-G Copy Text

Thời gian khi không có where:

```
root@cee75f88c0ee:/home# python3 exam1.py
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
24/06/09 17:11:03 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
run time:  0.16033649444580078
```

## - Thời gian khi sử dụng where



```
GNU nano 4.8 exam1.py
.appName("SparkSQL CRUD Operations") \
.config("spark.some.config.option", "some-value") \
.getOrCreate()

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

#CREATE
new_data = [(("A", "B10000", 2024, 234, 232), ("C", "A10000", 2024, 214, 132))]
columns = ['anzsic06', 'Area', 'year', 'geo_count', 'ec_count']
new_df = spark.createDataFrame(new_data, columns)
df_new = df.union(new_df)

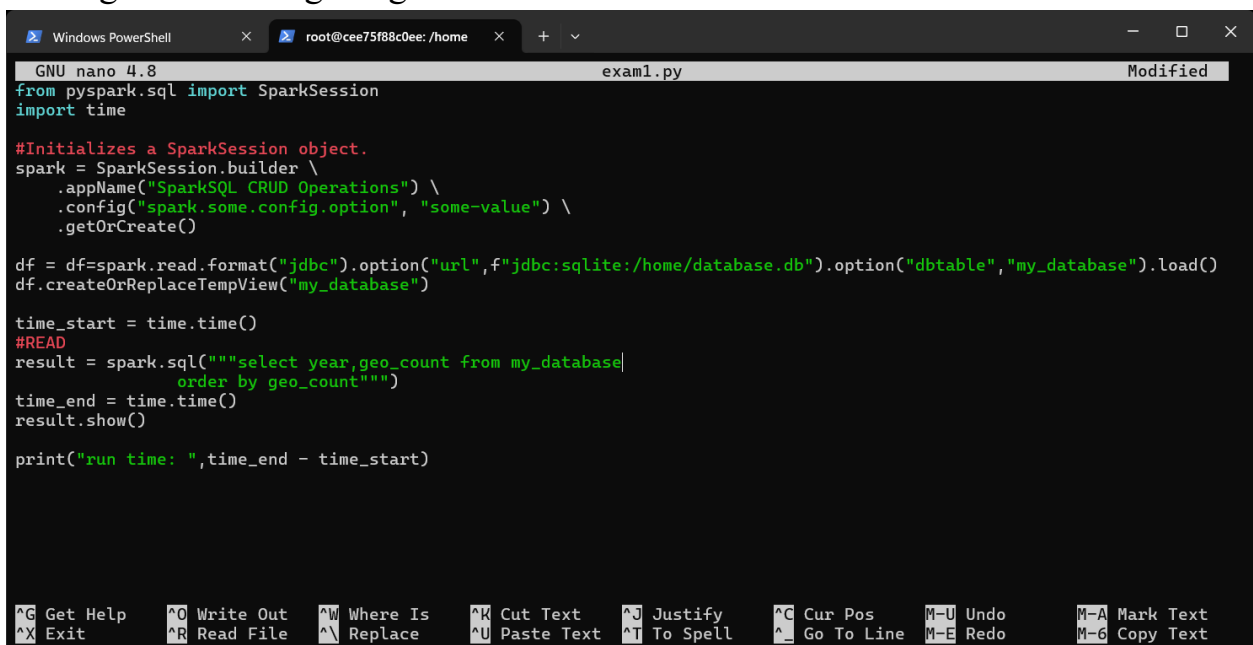
#Create Temporary View and Execute Query
df_new.createOrReplaceTempView("my_database")

time_start = time.time()
#READ
result = spark.sql("""select COUNT(*) ,year,geo_count from my_database
                        where geo_count > 300
                        group by year
                        order by geo_count""")
time_end = time.time()
result.show()

[ Wrote 33 lines ]
^G Get Help      ^O Write Out    ^W Where Is    ^K Cut Text    ^J Justify    ^C Cur Pos    M-U Undo    M-A Mark Text
^X Exit          ^R Read File    ^\ Replace     ^U Paste Text  ^T To Spell   ^_ Go To Line M-E Redo    M-6 Copy Text
```

```
To adjust logging level use sc.setLogLevel('WARN')
24/06/08 09:36:32 WARN NativeCodeLoader:
thoi gian chay: 1.4415128231048584
```

## -Thời gian khi không dung where



```
GNU nano 4.8 exam1.py Modified
from pyspark.sql import SparkSession
import time

#Initializes a SparkSession object.
spark = SparkSession.builder \
    .appName("SparkSQL CRUD Operations") \
    .config("spark.some.config.option", "some-value") \
    .getOrCreate()

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

time_start = time.time()
#READ
result = spark.sql("""select year,geo_count from my_database
                        order by geo_count""")
time_end = time.time()
result.show()

print("run time: ",time_end - time_start)
```

```
24/06/08 09:31:14 WARN NativeCodeLoader:
thoi gian chay: 1.7001993656158447
```

- Qua các thử nghiệm trên, where có ảnh hưởng vào hiệu năng truy vấn, tùy 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.

## **7. 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.