# Basic Python - Work with SQLite Database

*Hoàng-Nguyên Vũ*


1. **Mô tả: Làm quen với Cơ sở dữ liệu SQLite**

   - **SQLite** là một hệ quản trị cơ sở dữ liệu hay còn gọi là hệ thống cơ sở dữ liệu quan hệ nhỏ gọn, khác với các hệ quản trị khác như MySQL, SQL Server, Ocracle, PostgreSQL... SQLite là một thư viện phần mềm mà triển khai một SQL Database Engine truyền thống, không cần mô hình client-server nên rất nhỏ gọn. SQLite được sử dụng vào rất nhiều chương trình từ desktop đến mobile hay là website..
   - Ngoài những lý do trên thì không thể không kể đến những ưu điểm khi sử dụng SQLite, sau đây là phần ưu điểm của SQLite. **Ưu điểm của SQLite:**
     - **Nhẹ và dễ sử dụng:** SQLite không yêu cầu cài đặt máy chủ riêng biệt và có thể được nhúng trực tiếp vào ứng dụng Python của bạn.
     - **Hiệu suất cao:** SQLite cung cấp hiệu suất truy vấn nhanh và có thể xử lý lượng dữ liệu lớn.
     - **Đa nền tảng:** SQLite có sẵn trên hầu hết các nền tảng, bao gồm Windows, MacOS, Linux và Android.
     - **Miễn phí và mã nguồn mở:** SQLite là phần mềm miễn phí và mã nguồn mở, có nghĩa là bạn có thể sử dụng và sửa đổi nó cho mục đích của riêng bạn.

2. **Làm quen với các câu lệnh SQL trong SQLite Database:**

   - Ví dụ chúng ta cần quản lý thông tin khách hàng gồm các trường dữ liệu cơ bản như sau:

Bảng 1: Bảng quản lý thông tin của khách hàng

| CUSTOMER | | |
|---|---|---|
| Email | Name | Phone |

Phân tích sơ lược, chúng ta dễ thấy mỗi khách hàng chỉ có duy nhất 1 email, nên email sẽ đóng vai trò là khóa chính, nhằm dễ xác định thông tin của một khách hàng trong hệ thống. Để quản lý các thông tin trên và và truy vấn chúng ta sẽ có những thao tác chính như sau: Tạo bảng **(CREATE TABLE)**, lấy dữ liệu từ bảng **(SELECT)**, thêm mới dữ liệu **(INSERT)**, cập nhật dữ liệu **(UPDATE)**, xóa dữ liệu **(DELETE)**.

+ **Tạo bảng mới (CREATE TABLE):** Để tạo bảng trên ta sẽ sử dụng như sau:

```sql
-- Tạo Bảng CUSTOMERS gồm 3 cột email, name, phone với email là khóa chính
CREATE TABLE customers (
  email TEXT PRIMARY KEY,
  name TEXT NOT NULL,
  phone TEXT
);
```

+ **Lấy dữ liệu (SELECT):** Chúng ta có 2 cách lấy dữ liệu từ bảng trong cơ sở dữ liệu bao gồm: lấy hết dữ liệu của toàn bộ cột trong bảng và lấy dữ liệu từ các cột được chỉ định. Cách truy vấn (Query) có cấu trúc tổng quát như sau:

```sql
SELECT column1, column2, ...
FROM table_name;
```

**A. Lấy toàn bộ dữ liệu của tất cả cột trong bảng:**

```sql
-- Lấy toàn bộ dữ liệu của tất cả cột trong bảng CUSTOMERS
SELECT *
FROM CUSTOMERS;
```

**B. Lấy dữ liệu từ các cột được chỉ định trong bảng:**

```sql
-- Lấy dữ liệu từ các cột được chỉ định trong bảng CUSTOMERS
SELECT NAME, PHONE
FROM CUSTOMERS;
```

**C. Lấy dữ liệu theo điều kiện trong bảng (Filter):**

```sql
-- Lấy dữ liệu theo điều kiện email = giá trị truyền vào từ bảng CUSTOMERS
SELECT NAME, PHONE
FROM CUSTOMERS
WHERRE 1 = 1
AND EMAIL = 'aivietnam@aivietnam.edu.vn';
```

+ **Thêm dữ liệu (INSERT):** Để thêm dữ liệu vào bảng chúng ta có thể thực thi với câu lệnh có cấu trúc tổng quát như sau:

```sql
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

**Ví dụ:**

```
-- Thêm mới 1 dòng trong bảng CUSTOMERS
INSERT INTO CUSTOMERS(EMAIL, NAME, PHONE)
VALUES('nguyen@nguyen.com', 'Nguyen', '123456789');

-- Thêm mới nhiều dòng trong bảng CUSTOMERS
INSERT INTO CUSTOMERS(EMAIL, NAME, PHONE)
VALUES
    ('nguyen@aivietnam.edu.vn', 'Nguyen', '123456789'),
    ('admin@aivietnam.edu.vn', 'Vinh', '1122334455');
```

+ **Cập nhật dữ liệu (UPDATE):** Để cập nhật dữ liệu vào bảng chúng ta có thể thực thi với câu lệnh có cấu trúc tổng quát như sau:

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

**Ví dụ:** Cập nhật tên của email là nguyen@aivietnam.edu.vn thành Hoang Nguyen.

```
-- Cập nhật tên của email là nguyen@aivietnam.edu.vn thành Hoang
   Nguyen
UPDATE CUSTOMERS
SET NAME = 'Hoang Nguyen'
WHERE 1 = 1
AND EMAIL = 'nguyen@aivietnam.edu.vn';
```

+ **Xóa dữ liệu (DELETE):** Để xóa dữ liệu khỏi bảng chúng ta có thể thực thi với câu lệnh có cấu trúc tổng quát như sau:

```
DELETE FROM table_name
WHERE condition;
```

**Ví dụ:** Xóa email nguyen@aivietnam.edu.vn khỏi bảng CUSTOMERS

```
-- Xóa email: nguyen@aivietnam.edu.vn khỏi bảng CUSTOMERS
DELETE FROM CUSTOMERS
WHERE 1 = 1
AND EMAIL = 'nguyen@aivietnam.edu.vn';
```

3. **Sử dụng SQLite trong Python:**

- **Tạo kết nối đến CSDL:** Để sử dụng SQLite và tương tác với cơ sở dữ liệu SQLite trong python, chúng ta cần tạo kết nối đến CSDL bằng câu lệnh như sau:

```
import sqlite3
# Tạo kết nối tới CSDL có tên là database.sqlite
# Nếu database.sqlite chưa tồn tại trong hệ thống thì nó sẽ
# tự tạo mới
connection = sqlite3.connect('database.sqlite')
cursor = connection.cursor()
```

→ Sau khi tạo kết nối đến cơ sở dữ liệu xong, chúng ta có Object có tên *cursor* nhằm giúp chúng ta thực thi và tương tác các câu lệnh truy vấn (SELECT, INSERT, UPDATE, DELETE,...) đến cơ sở dữ liệu.

- **Tạo bảng:** Để tạo bảng mới xong cơ sở dữ liệu chúng ta sẽ thực hiện như sau:

```
# Tạo bảng CUSTOMERS:
cursor.execute("""
CREATE TABLE CUSTOMERS (
  EMAIL TEXT PRIMARY KEY,
  NAME TEXT NOT NULL,
  PHONE TEXT NOT NULL
);
""")
```

- **Thêm dữ liệu vào bảng (INSERT):** Sau khi thực hiện tạo bảng ở bước trên, lúc này bảng CUSTOMERS của chúng ta hoàn toàn không có dữ liệu, nên chúng ta sẽ thêm dữ liệu vào bảng bằng cách thực thi câu SQL như sau:

```
# Insert data mới
cursor.execute("""
INSERT INTO CUSTOMERS(EMAIL, NAME, PHONE)
VALUES
    ('nguyen@aivietnam.edu.vn', 'Nguyen', '123456789'),
    ('admin@aivietnam.edu.vn', 'Vinh', '1122334455');
""")

connection.commit()
```

→ **Mở rộng:** Khi thực hiện thao tác INSERT/UPDATE/DELETE, khi thực hiện cursor.execute() trên cơ sở dữ liệu SQLite, những thay đổi này chỉ được lưu trữ tạm thời trong bộ nhớ. Nên chúng ta cần phải thực bước commit để lưu những thay đổi này vào CSDL, đảm bảo chúng được lưu trữ vĩnh viễn.

- **Lấy dữ liệu (SELECT):** Sau khi thực hiện bước tạo dữ liệu ở trên, chúng ta sẽ kiểm tra dữ liệu vừa thêm bằng cách thực thi câu SQL như sau:

```
import pandas as pd
# Lấy tất cả data từ bảng CUSTOMER
data=pd.read_sql_query("SELECT * FROM CUSTOMERS", connection)
print(data)
```

→ **Kết quả:**

```
[5]: import pandas as pd

[7]: data=pd.read_sql_query("SELECT * FROM CUSTOMERS", connection)
     print(data)

                       EMAIL    NAME       PHONE
     0  nguyen@aivietnam.edu.vn  Nguyen   123456789
     1   admin@aivietnam.edu.vn    Vinh  1122334455
```

Hình 1: Kết quả sau khi thực hiện SELECT

- **Cập nhật dữ liệu (UPDATE):** Để thực hiện cập nhật dữ liệu trong CSDL, chúng ta sẽ thực thi câu SQL như sau:

Cập nhật tên của email : nguyen@aivietnam.edu.vn sang giá trị mới và kiểm tra kết quả sau khi cập nhật.

```
# Update name của email: nguyen@aivietnam.edu.vn
cursor.execute("""
UPDATE CUSTOMERS
SET NAME = 'Hoang Nguyen'
WHERE 1 = 1
AND EMAIL = 'nguyen@aivietnam.edu.vn';
""")

connection.commit()

data=pd.read_sql_query("SELECT * FROM CUSTOMERS", connection)
print(data)
```

→ **Kết quả:**

| | EMAIL | NAME | PHONE |
|---|---|---|---|
| 0 | nguyen@aivietnam.edu.vn | Hoang Nguyen | 123456789 |
| 1 | admin@aivietnam.edu.vn | Vinh | 1122334455 |

Hình 2: Kết quả sau khi thực hiện UPDATE

- **Xóa dữ liệu (DELETE):** Để thực hiện xóa dữ liệu trong CSDL, chúng ta sẽ thực thi câu SQL như sau:

  Xóa email : nguyen@aivietnam.edu.vn khỏi bảng CUSTOMERS và kiểm tra kết quả sau khi cập nhật.

```
# Update name của email: nguyen@aivietnam.edu.vn
cursor.execute("""
DELETE FROM CUSTOMERS
WHERE 1 = 1
AND EMAIL = 'nguyen@aivietnam.edu.vn';
""")

connection.commit()

data=pd.read_sql_query("SELECT * FROM CUSTOMERS", connection)
print(data)
```

→ **Kết quả:**

| | EMAIL | NAME | PHONE |
|---|---|---|---|
| 0 | admin@aivietnam.edu.vn | Vinh | 1122334455 |

Hình 3: Kết quả sau khi thực hiện DELETE

4. **Bài tập:**

- Hãy tạo mới bảng có tên *PRODUCT* có các cột như sau:

Bảng 2: Bảng quản lý sản phẩm

| **PRODUCT** | | |
|---|---|---|
| **Cột** | **Kiểu dữ liệu** | **Chú thích** |
| ID | INTEGER | Khóa chính |
| NAME | TEXT | Not null |
| PRICE | INTEGER | Not null |

**Câu 1:** Hãy thêm mới các dòng có giá trị như sau và kiểm tra kết quả:

| **PRODUCT** | | |
|---|---|---|
| **ID** | **NAME** | **PRICE** |
| 1 | iPhone 15 | 18000000 |
| 2 | Galaxy Z-Fold 5 | 30000000 |

→ **Kết quả:**

```
   ID             NAME     PRICE
0   1        iPhone 15  18000000
1   2  Galaxy Z-Fold 5  30000000
```

**Câu 2:** Hãy cập nhật giá mới cho Galaxy Z-Fold 5 thành 50.000.000 và kiểm tra kết quả:
→ **Kết quả:**

```
   ID             NAME     PRICE
0   1        iPhone 15  18000000
1   2  Galaxy Z-Fold 5  50000000
```

**Câu 3:** Hãy xóa iPhone 15 ra khỏi CSDL và kiểm tra kết quả:
→ **Kết quả:**

```
   ID             NAME     PRICE
0   2  Galaxy Z-Fold 5  50000000
```

# Basic Python - Work with SQLite Database

*Hoàng-Nguyên Vũ*


1. **Mô tả: Thống kê cơ bản trong SQLite**

   - **SQL** cung cấp nhiều hàm để thực hiện các phép tính thống kê cơ bản trên dữ liệu. Các hàm này được sử dụng để tính toán các giá trị như tổng, trung bình, giá trị tối đa, giá trị tối thiểu, v.v. Dưới đây là một số hàm thống kê cơ bản trong SQL:
     - **SUM():** Tính tổng của các giá trị trong một cột.
     - **AVG():** Tính trung bình của các giá trị trong một cột.
     - **MIN():** Tìm giá trị nhỏ nhất trong một cột.
     - **MAX():** Tìm giá trị lớn nhất trong một cột.
     - **COUNT():** Đếm số lượng các giá trị trong một cột.

     **Ví dụ:** Giả sử bạn có một bảng *Product* với các cột *name*, *brand* và *price*. Bạn muốn biết:

     - Tổng doanh thu của tất cả các sản phẩm.
     - Doanh thu trung bình của các sản phẩm.
     - Giá sản phẩm cao nhất.
     - Số lượng các sản phẩm khác nhau.

     Câu lệnh SQL cho các yêu cầu trên sẽ như sau:

```
# Query để lấy tổng giá bán toàn bộ sản phẩm trong bảng Product
query = """
SELECT SUM(price) AS total_revenue
FROM PRODUCT;
"""

data_sum = pd.read_sql_query(query, connection)
print(data_sum)

```

```
# Query để lấy thông tin sản phẩm có giá bán cao nhất
query = """
SELECT NAME, MAX(price) AS PRICE
FROM PRODUCT;
"""
data_max = pd.read_sql_query(query, connection)
print(data_max)
```

→ **Kết quả:**

```
data=pd.read_sql_query("SELECT * FROM PRODUCT", connection)
print('=== All data ===')
print(data)
```

```
=== All data ===
   ID               NAME    BRAND     PRICE
0   1          iPhone 15    Apple  18000000
1   2    Galaxy Z-Fold 5  Samsung  30000000
2   3             Find X     Oppo  20000000
3   4          iPhone 14    Apple  16000000
4   5      Galaxy Z-Flip  Samsung  17000000
5   6  iPhone 15 Pro Max    Apple  48000000
```

```
query = """
SELECT SUM(price) AS total_revenue
FROM PRODUCT;
"""
data_sum = pd.read_sql_query(query, connection)
print(data_sum)
```

```
   total_revenue
0      149000000
```

```
query = """
SELECT NAME, MAX(price) AS PRICE
FROM PRODUCT;
"""
data_max = pd.read_sql_query(query, connection)
print(data_max)
```

```
               NAME     PRICE
0  iPhone 15 Pro Max  48000000
```

- **Hàm thống kê với GROUP BY:**

  **GROUP BY** là một mệnh đề trong SQL được sử dụng để nhóm các hàng dựa trên các giá trị chung trong một hoặc nhiều cột và thực hiện các phép tính tổng hợp trên các nhóm đó.

  Hàm thống kê được sử dụng để tính toán các giá trị như tổng, trung bình, giá trị tối đa, giá trị tối thiểu, v.v. trên các nhóm dữ liệu.

  **Ví dụ:** Giả sử bạn có một bảng *Product* với các cột *name*, *brand* và *price*. Bạn muốn biết:

  – Doanh thu tổng cho mỗi hãng (BRAND) của sản phẩm.

– Giá bán thấp nhất của mỗi danh mục.

Cách thực thi câu lệnh SQL cho 2 yêu cầu trên sẽ như sau:

```
# Query để lấy tổng giá bán toàn bộ sản phẩm theo hãng trong bảng Product
query = """
SELECT BRAND, SUM(price) AS total_revenue
FROM PRODUCT
GROUP BY BRAND;
"""
data_sum_by_brand = pd.read_sql_query(query, connection)
print(data_sum_by_brand)

# Query để lấy thông tin sản phẩm có giá bán thấp nhất
query = """
SELECT NAME, BRAND, MIN(price) AS PRICE
FROM PRODUCT
GROUP BY BRAND;
"""
data_min_by_brand = pd.read_sql_query(query, connection)
print(data_min_by_brand)
```

→ **Kết quả:**

```
query = """
SELECT BRAND, SUM(price) AS total_revenue
FROM PRODUCT
GROUP BY BRAND;
"""
data_sum_by_brand = pd.read_sql_query(query, connection)
print(data_sum_by_brand)
```

```
     BRAND  total_revenue
0    Apple       82000000
1     Oppo       20000000
2  Samsung       47000000
```

```
query = """
SELECT NAME, BRAND, MIN(price) AS PRICE
FROM PRODUCT
GROUP BY BRAND;
"""
data_min_by_brand = pd.read_sql_query(query, connection)
print(data_min_by_brand)
```

```
            NAME    BRAND     PRICE
0      iPhone 14    Apple  16000000
1         Find X     Oppo  20000000
2  Galaxy Z-Flip  Samsung  17000000
```

2. **Bài tập:**

- Hãy tạo mới bảng có tên *STOCK* có các cột như sau:

Bảng 1: Bảng quản lý cổ phiếu

| **STOCK** | | |
|---|---|---|
| **Cột** | **Kiểu dữ liệu** | **Chú thích** |
| ID | INTEGER | Khóa chính |
| NAME | TEXT | Not null |
| BUY | INTEGER | Not null |
| INVESTOR | TEXT | Not null |

**Hãy thêm mới các dòng có giá trị như sau:**

| **STOCK** | | | |
|---|---|---|---|
| **ID** | **NAME** | **BUY** | **INVESTOR** |
| 1 | ACB | 29.45 | Nguyen |
| 2 | VIC | 44.55 | Nguyen |
| 3 | GMD | 74.30 | Nguyen |
| 4 | ACB | 28.45 | Vinh |
| 5 | VIC | 40.55 | Vinh |
| 6 | GMD | 60.30 | Vinh |

**Câu 1:** Hãy viết lệnh SQL để truy vấn và in ra kết quả thống kê tổng giá bán (BUY) của bảng STOCK:

**Kết quả:** Tổng giá bán = 277.69

**Câu 2:** Hãy viết lệnh SQL để thống kê mã cổ phiếu có giá mua (BUY) lớn nhất theo nhà đầu tư (Investor):
→ **Kết quả:**

| | INVESTOR | NAME | MAX_PRICE |
|---|---|---|---|
| 0 | Nguyen | GMD | 74.39 |
| 1 | Vinh | GMD | 60.30 |