1. Lấy tất cả dữ liệu từ một bảng:

SELECT \* FROM ten\_bang;

1. Lấy dữ liệu từ một số cột cụ thể:

SELECT cot1, cot2 FROM ten\_bang;

1. Lọc dữ liệu với điều kiện WHERE:

SELECT \* FROM ten\_bang WHERE cot = 'giatri';

1. Sắp xếp dữ liệu với ORDER BY:

SELECT \* FROM ten\_bang ORDER BY cot ASC; -- tăng dần

SELECT \* FROM ten\_bang ORDER BY cot DESC; -- giảm dần

1. Giới hạn số lượng kết quả với LIMIT:

SELECT \* FROM ten\_bang LIMIT 10;

1. Thêm dữ liệu vào bảng (INSERT):

INSERT INTO ten\_bang (cot1, cot2) VALUES ('giatri1', 'giatri2');

1. Cập nhật dữ liệu (UPDATE):

UPDATE ten\_bang SET cot1 = 'giatri\_moi' WHERE cot2 = 'giatri2';

1. Xóa dữ liệu (DELETE):

DELETE FROM ten\_bang WHERE cot = 'giatri';

**a.JOIN**

1. INNER JOIN

Kết hợp các bản ghi có giá trị chung ở cả hai bảng.

SQL

SELECT a.\*, b.\*

FROM bang\_a a

INNER JOIN bang\_b b ON a.khoa\_chinh = b.khoa\_ngoai;

2. LEFT JOIN (LEFT OUTER JOIN)

Lấy tất cả các bản ghi từ bảng bên trái và các bản ghi khớp từ bảng bên phải.

SQL

SELECT a.\*, b.\*

FROM bang\_a a

LEFT JOIN bang\_b b ON a.khoa\_chinh = b.khoa\_ngoai;

3. RIGHT JOIN (RIGHT OUTER JOIN)

Lấy tất cả bản ghi từ bảng bên phải và các bản ghi khớp từ bảng bên trái.

SQL

SELECT a.\*, b.\*

FROM bang\_a a

RIGHT JOIN bang\_b b ON a.khoa\_chinh = b.khoa\_ngoai;

4. FULL JOIN (FULL OUTER JOIN)

Lấy tất cả bản ghi khi có sự khớp ở một trong hai bảng.

SQL

SELECT a.\*, b.\*

FROM bang\_a a

FULL JOIN bang\_b b ON a.khoa\_chinh = b.khoa\_ngoai;

(Lưu ý: FULL JOIN không phải hệ quản trị CSDL nào cũng hỗ trợ, ví dụ MySQL không hỗ trợ FULL JOIN.)

5. SELF JOIN

Tự kết hợp chính bảng đó với chính nó.

SQL

SELECT a.\*, b.\*

FROM bang\_a a

JOIN bang\_a b ON a.khoa\_chinh = b.khoa\_ngoai;

**b. Aggregate Function**

1. Đếm số lượng bản ghi (COUNT)

SQL

SELECT COUNT(\*) FROM ten\_bang;

Đếm số lượng bản ghi trong bảng.

2. Tính tổng (SUM)

SQL

SELECT SUM(ten\_cot) FROM ten\_bang;

Tính tổng giá trị của một cột.

3. Tính giá trị trung bình (AVG)

SQL

SELECT AVG(ten\_cot) FROM ten\_bang;

Tính giá trị trung bình của một cột.

4. Giá trị lớn nhất (MAX)

SQL

SELECT MAX(ten\_cot) FROM ten\_bang;

Lấy giá trị lớn nhất của một cột.

5. Giá trị nhỏ nhất (MIN)

SQL

SELECT MIN(ten\_cot) FROM ten\_bang;

Lấy giá trị nhỏ nhất của một cột.

6. Kết hợp với GROUP BY

Dùng để tổng hợp dữ liệu theo nhóm.

SQL

SELECT cot\_nhom, COUNT(\*)

FROM ten\_bang

GROUP BY cot\_nhom;

Ví dụ: Đếm số sinh viên theo từng lớp:

SQL

SELECT ma\_lop, COUNT(\*) as so\_sinh\_vien

FROM sinhvien

GROUP BY ma\_lop;

7. Kết hợp với HAVING

Lọc dữ liệu sau khi đã nhóm.

SQL

SELECT ma\_lop, COUNT(\*) as so\_sinh\_vien

FROM sinhvien

GROUP BY ma\_lop

HAVING COUNT(\*) > 30;

**c. Window Function và Subquery**

1. Window Function (Hàm cửa sổ)

Window Function cho phép thực hiện các phép tính tổng hợp trên từng dòng dữ liệu mà không làm nhóm dữ liệu lại như GROUP BY. Một số hàm cửa sổ phổ biến: ROW\_NUMBER(), RANK(), DENSE\_RANK(), SUM() OVER(), AVG() OVER(),...

Ví dụ: Xếp hạng điểm của từng sinh viên trong từng lớp

SQL

SELECT

ma\_lop,

ten\_sv,

diem,

RANK() OVER (PARTITION BY ma\_lop ORDER BY diem DESC) AS xep\_hang

FROM sinhvien;

* PARTITION BY ma\_lop: Chia nhóm theo từng lớp.
* ORDER BY diem DESC: Sắp xếp điểm giảm dần trong mỗi lớp.

Ví dụ: Tính tổng điểm từng sinh viên so với tổng điểm toàn lớp

SQL

SELECT

ma\_lop,

ten\_sv,

diem,

SUM(diem) OVER (PARTITION BY ma\_lop) AS tong\_diem\_lop

FROM sinhvien;

2. Subquery (Truy vấn lồng)

Subquery là truy vấn nằm bên trong một truy vấn SQL khác.

Ví dụ 1: Lấy sinh viên có điểm cao nhất lớp

SQL

SELECT ten\_sv, diem

FROM sinhvien

WHERE diem = (

SELECT MAX(diem)

FROM sinhvien

WHERE ma\_lop = 'L01'

);

Ví dụ 2: Lấy các lớp có số sinh viên lớn hơn 30

SQL

SELECT ma\_lop

FROM lop

WHERE ma\_lop IN (

SELECT ma\_lop

FROM sinhvien

GROUP BY ma\_lop

HAVING COUNT(\*) > 30

);

Ví dụ 3: Subquery trong FROM

SQL

SELECT ma\_lop, so\_sinh\_vien

FROM (

SELECT ma\_lop, COUNT(\*) AS so\_sinh\_vien

FROM sinhvien

GROUP BY ma\_lop

) AS sub

WHERE so\_sinh\_vien > 30;

**d. Các hàm xử lý với dữ liệu dạng thời gian**

1. Lấy ngày, tháng, năm, giờ, phút, giây

MySQL & PostgreSQL:

SQL

SELECT

NOW() AS thoi\_gian\_hien\_tai, -- Lấy thời gian hiện tại

CURDATE() AS ngay\_hien\_tai, -- Lấy ngày hiện tại (MySQL)

CURRENT\_DATE AS ngay\_hien\_tai, -- Lấy ngày hiện tại (PostgreSQL)

YEAR(ngay\_sinh) AS nam,

MONTH(ngay\_sinh) AS thang,

DAY(ngay\_sinh) AS ngay,

HOUR(gio\_tao) AS gio,

MINUTE(gio\_tao) AS phut,

SECOND(gio\_tao) AS giay

FROM sinhvien;

2. Định dạng ngày tháng

MySQL:

SQL

SELECT DATE\_FORMAT(ngay\_sinh, '%d/%m/%Y') AS ngay\_thang\_nam

FROM sinhvien;

PostgreSQL:

SQL

SELECT TO\_CHAR(ngay\_sinh, 'DD/MM/YYYY') AS ngay\_thang\_nam

FROM sinhvien;

3. Tính khoảng cách giữa hai mốc thời gian

MySQL:

SQL

SELECT DATEDIFF(ngay\_het\_han, ngay\_bat\_dau) AS so\_ngay

FROM hopdong;

PostgreSQL:

SQL

SELECT (ngay\_het\_han - ngay\_bat\_dau) AS so\_ngay

FROM hopdong;

4. Cộng/trừ ngày tháng

MySQL:

SQL

SELECT DATE\_ADD(ngay\_bat\_dau, INTERVAL 7 DAY) AS sau\_7\_ngay

FROM hopdong;

PostgreSQL:

SQL

SELECT ngay\_bat\_dau + INTERVAL '7 days' AS sau\_7\_ngay

FROM hopdong;

5. Lấy các bản ghi theo điều kiện thời gian

SQL

SELECT \*

FROM donhang

WHERE ngay\_dat >= CURDATE() - INTERVAL 30 DAY; -- Đơn hàng trong 30 ngày gần nhất (MySQL)

6. Chuyển đổi giữa kiểu dữ liệu ngày/tháng

MySQL:

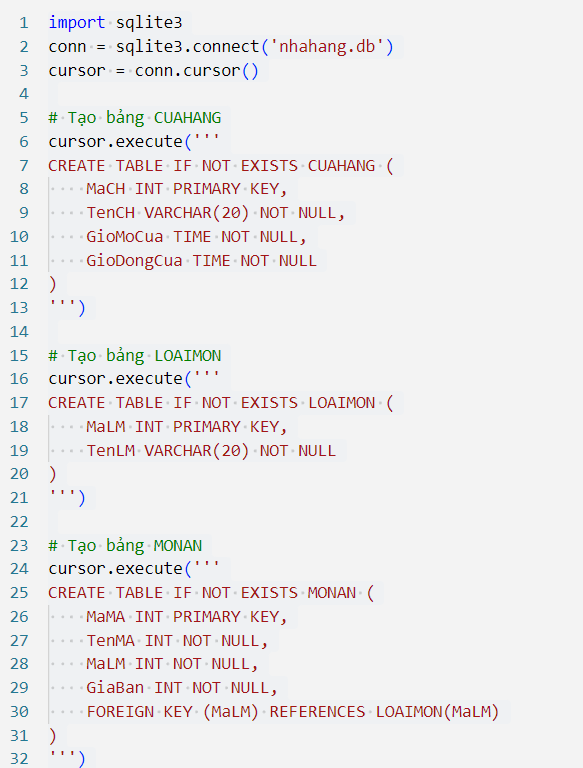
SQL

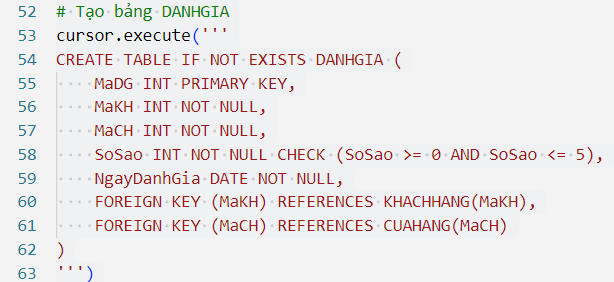
SELECT STR\_TO\_DATE('13-06-2025', '%d-%m-%Y') AS chuyen\_thanh\_date;

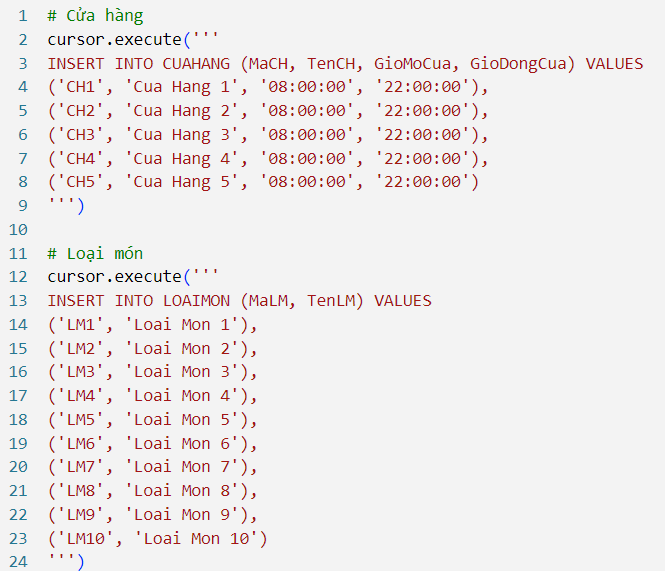
PostgreSQL:

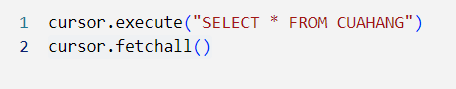
SQL

SELECT TO\_DATE('13-06-2025', 'DD-MM-YYYY') AS chuyen\_thanh\_date;









**Tạo dữ liệu tự động**

import random

from datetime import datetime, timedelta

# Tạo dữ liệu tự động cho bảng CUAHANG

cuahang\_data = [

(i, f"Cửa hàng {i}",

f"{random.randint(7, 10)}:{random.choice(['00', '30'])}",

f"{random.randint(20, 23)}:{random.choice(['00', '30'])}")

for i in range(1, 6)

]

# Tạo dữ liệu tự động cho bảng LOAIMON

loaimon\_data = [

(i, f"Loại món {i}")

for i in range(1, 11)

]

# Tạo dữ liệu tự động cho bảng MONAN

monan\_data = [

(i, f"Món {i}", random.randint(1, 10), round(random.uniform(10, 30), 2))

for i in range(1, 16)

]

# Tạo dữ liệu tự động cho bảng KHACHHANG

khachhang\_data = [

(i, f"Khách hàng {i}")

for i in range(1, 13)

]

# Tạo dữ liệu tự động cho bảng DATMON

datmon\_data = [

(i, random.randint(1, 15), random.randint(1, 5))

for i in range(1, 16)

]

# Tạo dữ liệu tự động cho bảng DANHGIA

start\_date = datetime(2025, 6, 1)

danhgia\_data = [

(

i,

random.randint(1, 12),

random.randint(1, 5),

random.randint(1, 5),

(start\_date + timedelta(days=i-1)).strftime('%Y-%m-%d')

)

for i in range(1, 13)

]

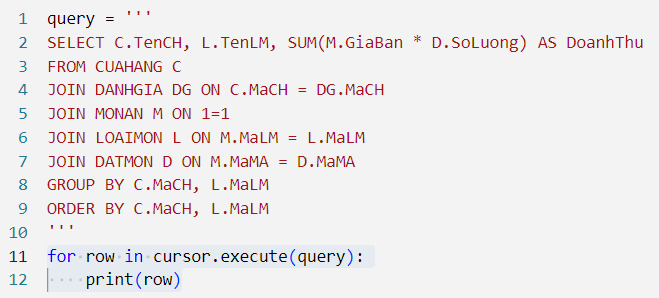
cursor.executemany(

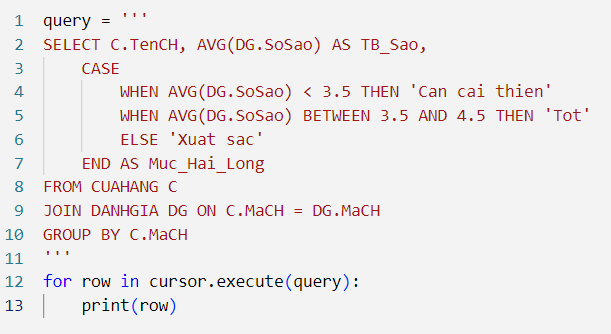
"insert into DANHGIA(MaDG, MaKH, MaCH, SoSao, NgayDanhGia) values(?,?,?,?,?)",

danhgia\_data

)

**Thống kê**

****

****

**Xử lý ngoại lai**#Tính ngưỡng ngoại lai

query = '''

WITH Q AS (

  SELECT

    (SELECT SoLuong FROM DATMON ORDER BY SoLuong LIMIT 1 OFFSET (SELECT COUNT(\*) \* 1 / 4 FROM DATMON)) AS Q1,

    (SELECT SoLuong FROM DATMON ORDER BY SoLuong LIMIT 1 OFFSET (SELECT COUNT(\*) \* 3 / 4 FROM DATMON)) AS Q3

),

Bounds AS (

  SELECT

    Q1 - 1.5 \* (Q3 - Q1) AS lower\_bound,

    Q3 + 1.5 \* (Q3 - Q1) AS upper\_bound

  FROM Q

)

SELECT lower\_bound, upper\_bound FROM Bounds;

'''

cursor.execute(query)

lower\_bound, upper\_bound = cursor.fetchone()

print(f'Ngưỡng ngoại lai: [{lower\_bound}, {upper\_bound}]')

# Lấy bản ghi ngoại lai

query = '''

WITH Q AS (

  SELECT

    (SELECT SoLuong FROM DATMON ORDER BY SoLuong LIMIT 1 OFFSET (SELECT COUNT(\*) \* 1 / 4 FROM DATMON)) AS Q1,

    (SELECT SoLuong FROM DATMON ORDER BY SoLuong LIMIT 1 OFFSET (SELECT COUNT(\*) \* 3 / 4 FROM DATMON)) AS Q3

),

Bounds AS (

  SELECT

    Q1 - 1.5 \* (Q3 - Q1) AS lower\_bound,

    Q3 + 1.5 \* (Q3 - Q1) AS upper\_bound

  FROM Q

)

SELECT \*

FROM DATMON, Bounds

WHERE SoLuong < lower\_bound OR SoLuong > upper\_bound;

'''

cursor.execute(query)

outliers = cursor.fetchall()

print('Bản ghi ngoại lai:', outliers)

# Xóa bản ghi ngoại lai

query = '''

WITH Q AS (

  SELECT

    (SELECT SoLuong FROM DATMON ORDER BY SoLuong LIMIT 1 OFFSET (SELECT COUNT(\*) \* 1 / 4 FROM DATMON)) AS Q1,

    (SELECT SoLuong FROM DATMON ORDER BY SoLuong LIMIT 1 OFFSET (SELECT COUNT(\*) \* 3 / 4 FROM DATMON)) AS Q3

),

Bounds AS (

  SELECT

    Q1 - 1.5 \* (Q3 - Q1) AS lower\_bound,

    Q3 + 1.5 \* (Q3 - Q1) AS upper\_bound

  FROM Q

)

DELETE FROM DATMON

WHERE SoLuong < (SELECT lower\_bound FROM Bounds)

   OR SoLuong > (SELECT upper\_bound FROM Bounds);

'''

cursor.execute(query)

conn.commit()

print('Đã xóa bản ghi ngoại lai')

conn.close()