

**ĐẠI HỌC BÁCH KHOA HÀ NỘI**  
**TRƯỜNG CNTT & TT**

\*\*\*



**BÁO CÁO PROJECT NHÓM MÔN**  
**THỰC HÀNH CƠ SỞ DỮ LIỆU**

Học phần : Thực hành cơ sở dữ liệu

Mã lớp học : 147775

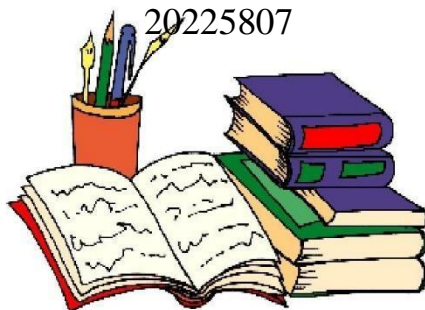
Giảng viên hướng dẫn : **TS. Vũ Tuyết Trinh**

**Nhóm sinh viên thực hiện: Nhóm 12**

Phạm Việt Anh 20225599

Mạch Ngọc Đức Anh 20225595

Đỗ Hoàng Đông 20225807



## **MỞ ĐẦU**

Đồ án môn học Thực hành Cơ sở dữ liệu do cô Vũ Tuyết Trinh hướng dẫn với chủ đề là : Xây dựng hệ thống quản lý dữ liệu cửa hàng điện thoại và phụ kiện(PSMS). Đồ án trình bày từ thực tiễn cần thiết đến thiết kế chi tiết cơ sở dữ liệu trên hệ quản trị cơ sở dữ liệu là PostgreSQL .

# Mục Lục

<b>CHƯƠNG I. PROJECT PROPOSAL .....</b>	<b>4</b>
<b>I.1. Project title .....</b>	<b>4</b>
<b>I.2. Context.....</b>	<b>4</b>
<b>I.3. Description.....</b>	<b>4</b>
<b>I.4. Requirements .....</b>	<b>4</b>
<b>CHƯƠNG II. THIẾT KẾ CƠ SỞ DỮ LIỆU.....</b>	<b>6</b>
<b>II.1. Thực thể gồm .....</b>	<b>6</b>
<b>II.2. Liên kết.....</b>	<b>6</b>
<b>II.3. Thuộc tính .....</b>	<b>6</b>
<b>II.4. Mô hình ERB .....</b>	<b>7</b>
<b>CHƯƠNG III. XÂY DỰNG CƠ SỞ DỮ LIỆU .....</b>	<b>8</b>
<b>III.1. Xây dựng các bảng .....</b>	<b>8</b>
<b>III.2. Xây dựng các hàm và TRIGGER .....</b>	<b>10</b>
<b>III.2.1. Một số hàm trong hệ thống.....</b>	<b>10</b>
<b>III.2.2. Một số TRIGGER trong hệ thống .....</b>	<b>13</b>
<b>III.3. Các câu truy vấn chạy thử CSDL .....</b>	<b>18</b>
<b>III.4. Tối ưu hóa tốc độ truy vấn dữ liệu bằng INDEX .....</b>	<b>28</b>
<b>KẾT LUẬN .....</b>	<b>36</b>

# CHƯƠNG I. PROJECT PROPOSAL

## I.1. Project title

Hệ thống quản lý dữ liệu cửa hàng điện thoại và phụ kiện(PSMS)

## I.2. Context

- Xây dựng cơ sở dữ liệu để quản lý thông tin liên quan đến sản phẩm, nhân viên trong cửa hàng, dữ liệu về khách hàng, dữ liệu vào ra của sản phẩm,... (mua bán cả offline và online)

-Mục đích: Cửa hàng áp dụng hệ thống để kiểm kho, quản lý nhân viên và doanh thu , đảm bảo về dịch vụ bảo hành được chính xác,...

## I.3. Description

Đối tượng sử dụng hệ thống cơ sở dữ liệu: nhân viên bán hàng, quản lý cửa hàng, admin staff.

1. Quản lý Sản phẩm: Hệ thống cơ sở dữ liệu giúp cửa hàng lưu trữ thông tin về sản phẩm như tên, thương hiệu, model, giá cả, và số lượng trong kho. Điều này giúp nhân viên bán hàng dễ dàng tìm kiếm và cập nhật thông tin về sản phẩm cho khách hàng.

2. Quản lý Khách hàng: Hệ thống cơ sở dữ liệu giúp ghi lại thông tin cá nhân của khách hàng bao gồm tên, địa chỉ, số điện thoại và email. Điều này giúp cửa hàng có thể tạo ra một cơ sở dữ liệu khách hàng và theo dõi lịch sử mua hàng của họ.

3. Quản lý Đơn đặt hàng: Cơ sở dữ liệu cho phép ghi lại thông tin về các đơn đặt hàng của khách hàng bao gồm sản phẩm đã mua, số lượng, giá và tổng giá trị của đơn hàng. Điều này giúp cửa hàng theo dõi các giao dịch mua bán và quản lý quá trình đặt hàng của khách hàng.

4. Quản lý tình trạng bảo hành: Cơ sở dữ liệu sẽ giúp cho phép nhân viên kiểm tra thông tin về các dịch vụ và được tình trạng bảo hành của các sản phẩm. Điều này giúp cửa hàng theo dõi được thông tin hạn bảo hành và chi phí sửa của sản phẩm.

5. Báo cáo và Phân tích: Cơ sở dữ liệu cung cấp dữ liệu cho việc tạo báo cáo và phân tích kinh doanh. Cửa hàng có thể sử dụng dữ liệu từ cơ sở dữ liệu để phân tích xu hướng mua hàng của khách hàng, đánh giá hiệu suất bán hàng và tạo ra chiến lược kinh doanh mới.

## I.4. Requirements

+) Khách hàng có thể tra cứu thông tin cơ bản của sản phẩm bằng cách nhập mã của sản phẩm

- **Đối với khách hàng mua hàng trực tiếp**

- + ) Nhân viên và quản lý có thể kiểm tra số lượng và thông tin của sản phẩm điện thoại cùng với các sản phẩm phụ kiện đi kèm còn lại ở trong kho bằng cách nhập vào mã sản phẩm
- + ) Khi khách hàng đồng ý mua sản phẩm thì nhân viên sẽ thêm thông tin của khách hàng vào cơ sở dữ liệu.
- + ) Đồng thời, nhân viên cũng thêm thông tin về đơn đặt hàng và xác nhận tình trạng thanh toán của khách hàng cho sản phẩm đã mua
- + ) Khi khách hàng muốn bảo hành, khách hàng sẽ đọc thông tin của khách hàng (số điện thoại) để nhân viên tra cứu lịch sử mua hàng (ngày tháng năm), khi nhân viên thấy sản phẩm và xác nhận khách hàng có mua sản phẩm thì nhân viên sẽ tra cứu mã sản phẩm để biết thông tin rằng sản phẩm này được bảo hành trong bao lâu để xác nhận với khách hàng có được bảo hành hay không
- + ) Nếu xác nhận bảo hành, nhân viên sẽ tạo đơn đặt hàng (ngày nhận máy, tình trạng, trạng thái) và chuyển máy cho nhân viên sửa chữa. Khi nào trả máy cho khách hàng thì cập nhật lại trạng thái cho sản phẩm

- **Đối với khách hàng mua sản phẩm online**

- + ) Khách hàng có thể tra cứu thông tin và tình trạng (còn/hết )
- + ) Nếu khách hàng muốn mua sản phẩm, thì khách hàng sẽ tự thêm thông tin về phương thức liên lạc, địa chỉ nhận hàng, số lượng (nếu số lượng mà khách hàng muốn mua vượt quá số lượng trong kho thì không thêm thông tin và yêu cầu khách hàng sửa lại số lượng), đưa ra số tiền thanh toán cho khách hàng. Sau khi khách hàng thanh toán thì nhân viên sẽ kiểm tra số tiền và xác nhận tình trạng thanh toán (nếu chưa thì nhân viên có thể gọi cho khách hàng để hỏi thêm, hướng dẫn thanh toán,...).
- + ) Nếu đã xác nhận, thì nhân viên sẽ cập nhật lại số lượng của sản phẩm trong kho và đưa thông tin khách hàng vào bảng giao hàng (thông tin của khách hàng, tình trạng giao hàng) và đóng gói sản phẩm chuyển cho dịch vụ chuyển phát
- + ) Nếu khách hàng muốn bảo hành thì khách hàng chuyển phát sản phẩm cho cửa hàng và nhân viên cũng kiểm tra hạn bảo hành như đối với khách hàng mua offline và lại chuyển phát lại cho khách hàng như lúc bán sản phẩm.

## CHƯƠNG II. THIẾT KẾ CƠ SỞ DỮ LIỆU

### II.1. Thực thể gồm

Sản phẩm, Khách hàng, Đơn hàng, Chi tiết đơn hàng, Bảo hành, Giỏ hàng.

### II.2. Liên kết

Sản phẩm (n) - Chi tiết đơn hàng(1)

Khách hàng(1) - Đơn hàng(n)

Chi tiết đơn hàng(n) - Đơn hàng(1)

Sản phẩm (1) - Bảo hành(n)

Khách hàng(1) - Bảo hành(n)

Giỏ hàng(n) - Khách hàng(1)

Giỏ hàng(n) - Sản phẩm (n)

### II.3. Thuộc tính

+) Sản phẩm (products): **product id**, product\_name, type\_product, brand, color , price, quantity\_prod, time\_warrant\_prod, seri)

+) Khách hàng (customers): **customer id**, customer\_name, phone\_number, password, date\_of\_birth, customer\_address, email, total\_order , membership\_level.

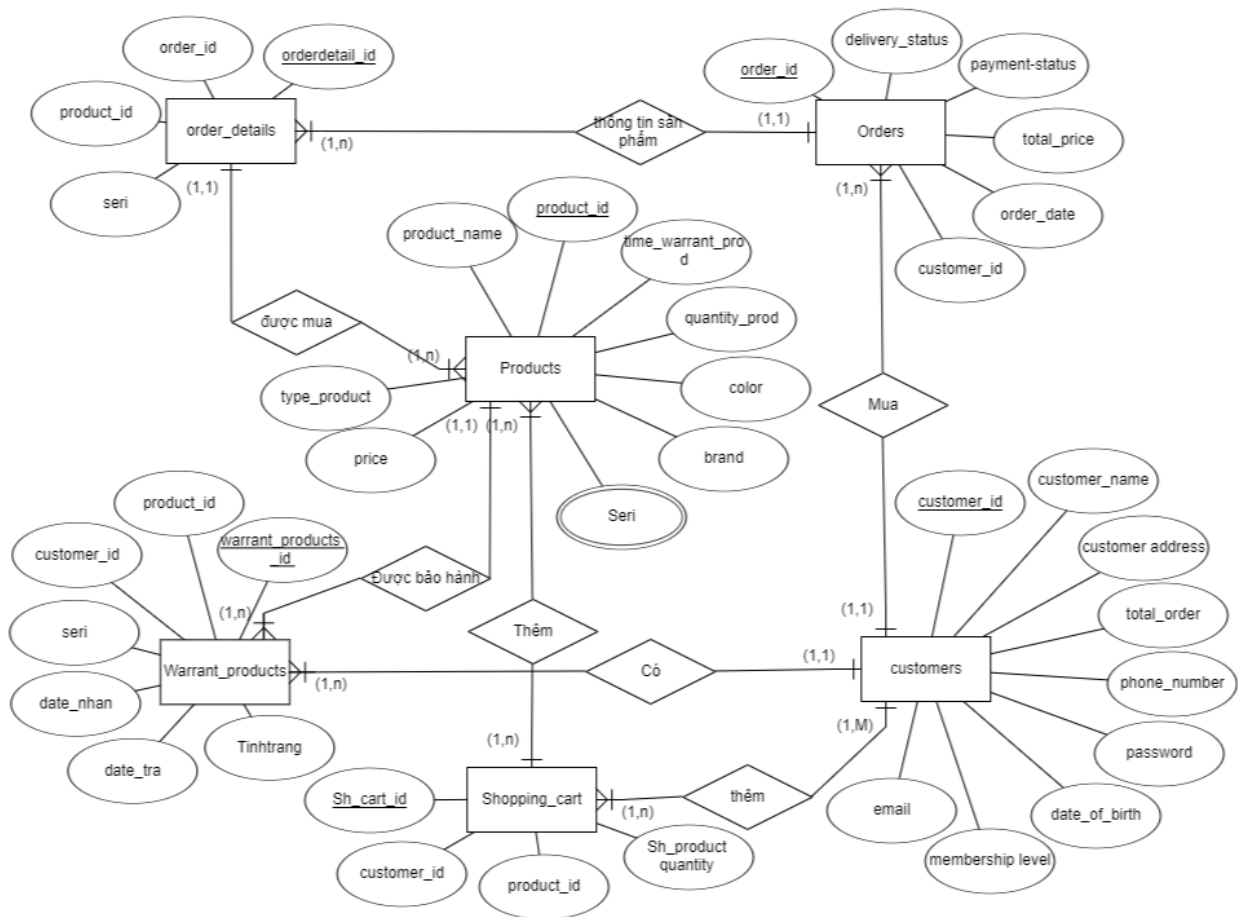
+) Đơn hàng(orders): **order id**, customer\_id , order\_date, total\_price, payment\_status , delivery\_status.

+) Chi tiết đơn hàng (order\_details): **orderdetail id**, order\_id, product\_id, seri

+) Bảo hành sản phẩm (warrant\_products): **warrant products id**, customer\_id, product\_id, seri, date\_nhan, date\_tra, tinh\_trang.

+) Giỏ Hàng (Shopping\_cart) : **Sh cart id**, customer\_id, product\_id, Sh\_product\_quantity.

## II.4. Mô hình ERB



## CHƯƠNG III. XÂY DỰNG CƠ SỞ DỮ LIỆU

### III.1. Xây dựng các bảng

#### **Bảng 1 : products**

```
CREATE TYPE loaisp AS ENUM ('dien thoai', 'phu kien');
CREATE TABLE products (
    product_id SERIAL PRIMARY KEY,
    product_name VARCHAR(50) NOT NULL ,
    type_product loaisp NOT NULL,
    brand VARCHAR(20) NOT NULL,
    color VARCHAR(20) NOT NULL,
    price NUMERIC(9) NOT NULL CHECK (price > 0),
    quantity_prod NUMERIC(5) NOT NULL CHECK (quantity_prod >= 0),
    time_warrant_prod NUMERIC(3) NOT NULL CHECK (time_warrant_prod >= 0)
);
```

- Lưu trữ các thông tin về các loại sản phẩm bao gồm điện thoại và phụ kiện có trong cửa hàng

#### **Bảng 2 : phone detail products**

```
CREATE TABLE phone_detail_products (
    detail_prod_id SERIAL PRIMARY KEY,
    product_id INT NOT NULL,
    seri VARCHAR(50) UNIQUE NOT NULL,
    FOREIGN KEY (product_id) REFERENCES products(product_id)
);
```

- Lưu trữ các thông tin về số seri của từng chiếc điện thoại mà cửa hàng đã từng nhập về

#### **Bảng 3 : customers**



```
CREATE TABLE customers (
    customer_id SERIAL PRIMARY KEY,
    customer_name VARCHAR(50) NOT NULL,
    phone_number VARCHAR(15) UNIQUE NOT NULL,
    password VARCHAR(30) NOT NULL,
    date_of_birth DATE NOT NULL,
    customer_address TEXT NOT NULL,
    email VARCHAR(60) UNIQUE NOT NULL,
    total_order INT DEFAULT 0 CHECK (total_order >= 0),
    membership_level VARCHAR(50)
);
```

- Lưu trữ các thông tin cơ bản của khách hàng đã từng mua hoặc có đăng nhập trên hệ thống web

#### **Bảng 4 : orders**

```
CREATE TYPE giaohang AS ENUM('dang xu ly', 'da gui', 'Dang van chuyen', 'da giao', 'da huy', 'da tra lai');
create table orders(
    order_id SERIAL primary key,
    customer_id int references customers(customer_id) ,
    order_date date NOT NULL DEFAULT CURRENT_DATE,
    payment_status bool NOT NULL DEFAULT FALSE,
    delivery_status giaohang NOT NULL,
    Total_price INT DEFAULT 0 CHECK (Total_price >= 0)
);
```

- Lưu trữ các thông tin về đơn hàng

#### **Bảng 5 : detail orders**

```
create table order_details(
    orderdetail_id SERIAL primary key,
    order_id int REFERENCES orders(order_id),
    product_id int REFERENCES products(product_id),
    seri VARCHAR(20) UNIQUE
);
```

- Lưu trữ các thông tin chi tiết về các đơn hàng

#### **Bảng 6 : warrant products**

```

CREATE TYPE tinh_trang_enum AS ENUM ('chưa sửa', 'đang sửa', 'đã sửa');
CREATE TABLE warrant_products(
    warrant_products_id SERIAL PRIMARY KEY,
    customer_id INT REFERENCES customers (customer_id),
    product_id INT REFERENCES products (product_id),
    seri VARCHAR(20)
    date_nhan DATE NOT NULL,
    date_tra DATE ,
    TinhTrang tinh_trang_enum NOT NULL,
    CONSTRAINT chk_date_tra CHECK (date_tra IS NULL OR date_tra >= date_nhan)
);

```

- Lưu trữ các thông tin về lịch sử bảo hành điện thoại của cửa hàng.

#### **Bảng 7 : Shopping cart**

```

CREATE TABLE Shopping_cart(
    Sh_cart_id SERIAL PRIMARY KEY,
    customer_id INT NOT NULL references customers (customer_id),
    product_id INT NOT NULL references products (product_id),
    SH_product_quantity INT NOT NULL CHECK (SH_product_quantity > 0)
);

```

- Lưu trữ các thông tin về các sản phẩm trong giỏ hàng của từng khách hàng mua online

## **III.2. Xây dựng các hàm và TRIGGER**

### **III.2.1. Một số hàm trong hệ thống**

**Hàm 1 : Sử dụng để chuyển dữ liệu từ giỏ hàng sang đơn hàng và đơn hàng chi tiết khi khách hàng xác nhận mua những sản phẩm có trong giỏ hàng**

```

CREATE OR REPLACE FUNCTION process_cart_to_order(khachhangid INT)
RETURNS VOID AS $$
DECLARE
    new_order_id INT;
    cart_item RECORD;
BEGIN
    INSERT INTO orders(customer_id, order_date, total_price, payment_status, delivery_status)
    VALUES (khachhangid, CURRENT_DATE, 0, FALSE, 'dang xu ly')
    RETURNING order_id INTO new_order_id;

    FOR cart_item IN
        SELECT s.product_id, s.sh_product_quantity
        FROM shopping_cart s
        WHERE s.customer_id = khachhangid

    LOOP
        LOOP
            INSERT INTO order_details(order_id, product_id, seri)
            VALUES ( new_order_id, cart_item.product_id, NULL);
            cart_item.sh_product_quantity := cart_item.sh_product_quantity-1;
            EXIT WHEN cart_item.sh_product_quantity <1;
        END LOOP;
    END LOOP;

    DELETE FROM Shopping_cart
    WHERE customer_id=khachhangid;
END;
$$
LANGUAGE plpgsql;

```

```

CREATE OR REPLACE FUNCTION before_remove_order(khachhangid1 INT)
RETURNS VOID AS $$
DECLARE
    carsh_customer RECORD;
    check_carsh BOOL :=TRUE;
BEGIN
    FOR carsh_customer IN
        SELECT s.product_id, s.sh_product_quantity, p.quantity_prod
        FROM shopping_cart s
        INNER JOIN products p ON s.product_id=p.product_id
        WHERE s.customer_id=khachhangid1
    LOOP
        IF carsh_customer.sh_product_quantity > carsh_customer.quantity_prod THEN
            RAISE EXCEPTION 'Số lượng sản phẩm % không đủ trong kho. Chỉ còn % sản phẩm.', carsh_customer.product_id, carsh_customer.quantity_prod ;
            check_carsh :=FALSE;
        END IF;
    END LOOP;
    IF check_carsh=TRUE THEN PERFORM process_cart_to_order(khachhangid1);
    END IF;
END;
$$
LANGUAGE plpgsql;

```

**Hàm 2 : Sử dụng để xóa một đơn hàng khi khách hàng không thanh toán hoặc xác nhận hủy đơn hàng**

```

CREATE OR REPLACE FUNCTION delete_order(delete_order_id INT)
RETURNS VOID AS $$
DECLARE
    order_customer_id INT;
    order_total_price NUMERIC;
BEGIN
    -- Lấy thông tin về khách hàng và tổng giá trị đơn hàng
    SELECT customer_id, total_price INTO order_customer_id, order_total_price
    FROM orders
    WHERE order_id = delete_order_id;

    -- Trừ giá trị đơn hàng từ total_order của khách hàng nếu payment_status là TRUE
    IF EXISTS (SELECT 1 FROM orders WHERE order_id = delete_order_id AND payment_status = TRUE) THEN
        UPDATE customers
        SET total_order = total_order - order_total_price
        WHERE customer_id = order_customer_id;
    END IF;

    -- Xóa các chi tiết đơn hàng
    DELETE FROM order_details
    WHERE order_id = delete_order_id;

    -- Xóa đơn hàng
    DELETE FROM orders
    WHERE order_id = delete_order_id;
END;
$$ LANGUAGE plpgsql;

```

**Hàm 3: Sử dụng để chèn 1 thông tin của khách hàng khi khách hàng mua offline thì nhân viên sẽ thêm còn khi khách đặt online thì khách hàng sẽ tự thêm**

```

CREATE OR REPLACE FUNCTION add_customer(
    p_customer_name VARCHAR(50),
    p_phone_number VARCHAR(15),
    p_password VARCHAR(30),
    p_date_of_birth DATE,
    p_customer_address TEXT,
    p_email VARCHAR(60)
)
RETURNS VOID AS $$
BEGIN
    INSERT INTO customers (
        customer_name,
        phone_number,
        password,
        date_of_birth,
        customer_address,
        email
    ) VALUES (
        p_customer_name,
        p_phone_number,
        p_password,
        p_date_of_birth,
        p_customer_address,
        p_email
    );
END;
$$ LANGUAGE plpgsql;

```

**Hàm 4: Dùng để tạo đơn hàng mới cho khách hàng thông qua số điện thoại**

```

CREATE OR REPLACE FUNCTION create_offline_order(
    p_phone_number VARCHAR(15)
)
RETURNS VOID AS $$
DECLARE
    v_customer_id INT;
BEGIN
    -- Lấy customer_id dựa trên số điện thoại
    SELECT customer_id INTO v_customer_id
    FROM customers
    WHERE phone_number = p_phone_number;

    -- Kiểm tra xem khách hàng có tồn tại không
    IF v_customer_id IS NULL THEN
        RAISE EXCEPTION 'Khách hàng với số điện thoại % không tồn tại', p_phone_number;
    END IF;

    -- Tạo đơn hàng mới cho khách hàng
    INSERT INTO orders (
        customer_id,
        order_date,
        total_price,
        payment_status,
        delivery_status
    ) VALUES (
        v_customer_id,
        CURRENT_DATE,
        0,
        FALSE,
        'đang xử lý'
    );
END;
$$ LANGUAGE plpgsql;

```

### III.2.2. Một số TRIGGER trong hệ thống

#### **TRIGGER 1 : Sử dụng để kiểm tra số lượng tối đa loại sản phẩm mà một khách hàng có thể thêm trong giỏ hàng**

Khách hàng chỉ được phép mua 20 loại sản phẩm ở trong giỏ hàng(online) để tránh gây lãng phí bộ nhớ. Do nếu khách hàng không mua sản phẩm mà cứ thêm rồi để đó lâu ngày thì sẽ gây lãng phí bộ nhớ

```

CREATE OR REPLACE FUNCTION check_cart_limit()
RETURNS TRIGGER AS $$
DECLARE
    max_items INT := 20; -- Giới hạn số lượng bản ghi trong giỏ hàng
    current_count INT;
BEGIN
    -- Đếm số lượng bản ghi hiện tại trong giỏ hàng của khách hàng
    SELECT COUNT(*)
    INTO current_count
    FROM Shopping_cart
    WHERE customer_id = NEW.customer_id;

    -- Kiểm tra nếu số lượng bản ghi vượt quá giới hạn
    IF current_count >= max_items THEN
        RAISE EXCEPTION 'Khách hàng này đã đạt giới hạn % sản phẩm trong giỏ hàng', max_items;
    END IF;

    RETURN NEW;
END; $$
LANGUAGE plpgsql;
CREATE TRIGGER before_insert_giohang
BEFORE INSERT ON Shopping_cart
FOR EACH ROW
EXECUTE FUNCTION check_cart_limit();

```

### **TRIGGER 2 : Sử dụng để update hạng của thành viên dựa vào tổng số tiền đã mua ở của hàng của mỗi khách hàng**

```

CREATE OR REPLACE FUNCTION update_membership_level()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.total_order < 15000000 THEN
        NEW.membership_level := 'S_new';
    ELSIF NEW.total_order >= 15000000 AND NEW.total_order < 50000000 THEN
        NEW.membership_level := 'S_mem';
    ELSE
        NEW.membership_level := 'S_vip';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER update_membership_level_trigger
BEFORE INSERT OR UPDATE ON customers
FOR EACH ROW
EXECUTE FUNCTION update_membership_level();

```

### **TRIGGER 3 : Sử dụng để tính tổng số tiền đã mua của một khách hàng tại của hàng**

Khi tình trạng thanh toán được chuyển từ FALSE sang TRUE thì hệ thống sẽ tự động cộng tổng số tiền của đơn hàng vừa thanh toán vào tổng số tiền mua của khách hàng từ đầu đến giờ

```

CREATE OR REPLACE FUNCTION update_total_order()
RETURNS TRIGGER AS $$
BEGIN
    IF (TG_OP = 'UPDATE' AND NEW.payment_status = TRUE AND OLD.payment_status = FALSE)
    THEN
        UPDATE customers
        SET total_order = total_order + NEW.total_price
        WHERE customer_id = NEW.customer_id;
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER update_total_order_trigger
AFTER INSERT OR UPDATE ON orders
FOR EACH ROW
EXECUTE FUNCTION update_total_order();

```

#### **TRIGGER 4 : Sử dụng để tính tổng tiền đơn hàng mỗi khi thêm vào bảng chi tiết đơn hàng tự động trừ số lượng sản phẩm trong kho khi bán**

Ban đầu khởi tạo giá trị của đơn hàng là 0 và khi thêm 1 sản phẩm vào đơn hàng chi tiết với mã order\_id tương ứng thì sẽ cộng dần vào giá trị của đơn hàng đồng thời số lượng sản phẩm đó trong kho sẽ trừ đi 1.

```

CREATE OR REPLACE FUNCTION update_order_and_product()
RETURNS TRIGGER AS $$
BEGIN
    UPDATE orders
    SET total_price = (
        SELECT COALESCE(SUM(p.price), 0)
        FROM order_details od
        JOIN products p ON od.product_id = p.product_id
        WHERE od.order_id = NEW.order_id )
    WHERE order_id = NEW.order_id;
    UPDATE products
    SET quantity_prod = quantity_prod - 1
    WHERE product_id = NEW.product_id;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Trigger để gọi hàm update_order_and_product

CREATE TRIGGER update_order_and_product_trigger
AFTER INSERT ON order_details
FOR EACH ROW
EXECUTE FUNCTION update_order_and_product();

```

### **TRIGGER 5 : Sử dụng để cập nhật lại số lượng sản phẩm trong kho khi hủy đơn hàng**

```
CREATE OR REPLACE FUNCTION restore_product_quantity()
RETURNS TRIGGER AS $$
BEGIN
    -- Cập nhật lại số lượng sản phẩm trong kho
    UPDATE products
    SET quantity_prod = quantity_prod + 1
    WHERE product_id = OLD.product_id;

    RETURN OLD;
END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER restore_product_quantity_trigger
AFTER DELETE ON order_details
FOR EACH ROW
EXECUTE FUNCTION restore_product_quantity();
```

### **TRIGGER 6: Sử dụng để kiểm tra số lượng sản phẩm trong kho với số lượng của khách mua online**

Kiểm tra số lượng của 1 sản phẩm có còn đủ trong kho hay không, nếu đủ thì xử lý đơn hàng trong giỏ hàng, nếu không đủ thì yêu cầu khách hàng giảm số lượng cho đến khi đủ số lượng

```
CREATE OR REPLACE FUNCTION check_product_quantity()
RETURNS TRIGGER AS $$
DECLARE
    available_quantity INT;
BEGIN
    SELECT quantity_prod
    INTO available_quantity
    FROM products
    WHERE product_id = NEW.product_id;
    IF NEW.sh_product_quantity > available_quantity THEN
        RAISE EXCEPTION 'Số lượng sản phẩm % không đủ trong kho. Chỉ còn % sản phẩm.', NEW.product_id, available_quantity;
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_stock_quantity_before_insert
BEFORE INSERT OR UPDATE ON Shopping_cart
FOR EACH ROW
EXECUTE FUNCTION check_product_quantity();
```

### **TRIGGER 7 : SỬ dụng để kiểm tra seri khi nhập vào chi tiết đơn hàng có tồn tại ở trong kho hay không**

Kiểm tra số seri mà khi nhân viên nhập vào bảng chi tiết đơn hàng phải đúng với seri đã có trong kho. Ví nếu nhân viên nhập sai sẽ dẫn tới nhiều vấn đề trong việc được bảo hành của khách hàng.



```

CREATE OR REPLACE FUNCTION check_seri_exists()
RETURNS TRIGGER AS $$
DECLARE
    serial_exists BOOLEAN;
BEGIN
    -- Check if the serial number exists in phone_detail_products
    SELECT EXISTS (
        SELECT 1
        FROM phone_detail_products
        WHERE seri = NEW.seri
    ) INTO serial_exists;
    -- Raise an exception if the serial number does not exist
    IF NOT serial_exists THEN
        RAISE EXCEPTION 'Serial number % does not exist in phone_detail_products.', NEW.seri;
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER before_insert_order_details
BEFORE UPDATE ON order_details
FOR EACH ROW
EXECUTE FUNCTION check_seri_exists();

```

### **TRIGGER 8 : Sử dụng để tự động cộng vào số lượng khi khách mua cùng 1 sản phẩm ở giỏ hàng**

```

CREATE OR REPLACE FUNCTION add_or_update_cart()
RETURNS TRIGGER AS $$
BEGIN
    -- Check if the product already exists in the customer's cart
    IF EXISTS (
        SELECT 1
        FROM Shopping_cart
        WHERE customer_id = NEW.customer_id
        AND product_id = NEW.product_id
    ) THEN
        -- If it exists, update the quantity
        UPDATE Shopping_cart
        SET SH_product_quantity = SH_product_quantity + NEW.SH_product_quantity
        WHERE customer_id = NEW.customer_id
        AND product_id = NEW.product_id;
        -- Prevent the new row from being inserted
        RETURN NULL;
    ELSE
        -- If it doesn't exist, allow the new row to be inserted
        RETURN NEW;
    END IF;
END;
$$ LANGUAGE plpgsql;

-- Trigger to call the function before insert on the Shopping_cart table
CREATE TRIGGER add_or_update_cart_trigger
BEFORE INSERT ON Shopping_cart
FOR EACH ROW
EXECUTE FUNCTION add_or_update_cart();

```

Khi khách hàng ấn mua 2 sản phẩm cùng loại thì sẽ tự động cộng thêm số lượng vào loại sản phẩm đã mua trước đó.

### **TRIGGER 9 : Sử dụng để cập nhật giảm giá cho đơn hàng phụ thuộc vào mức hạng thành viên.**

```

CREATE OR REPLACE FUNCTION apply_discount()
RETURNS TRIGGER AS $$
DECLARE
    discount_rate NUMERIC := 0;
BEGIN
    -- Kiểm tra nếu cột payment_status bị thay đổi
    IF TG_OP = 'UPDATE' AND NEW.payment_status IS DISTINCT FROM OLD.payment_status THEN
        -- Nếu chỉ cột payment_status thay đổi, không thực thi logic của trigger
        RETURN NEW;
    END IF;

    -- Kiểm tra hạng thành viên của khách hàng và xác định tỷ lệ giảm giá
    IF (SELECT membership_level FROM customers WHERE customer_id = NEW.customer_id) = 'S_vip' THEN
        discount_rate := 0.03;
    ELSIF (SELECT membership_level FROM customers WHERE customer_id = NEW.customer_id) = 'S_mem' THEN
        discount_rate := 0.01;
    END IF;

    -- Áp dụng giảm giá vào tổng giá đơn hàng
    NEW.total_price := NEW.total_price * (1 - discount_rate);

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

```

### III.3. Các câu truy vấn chạy thử CSDL

#### 1. . Viết câu truy vấn/hàm in ra thông tin của tiện thoại/phụ kiện thông qua tên

```

1 SELECT *
2 FROM products
3 WHERE product_name = 'iPhone 14';
4

```

Data Output Messages Notifications

	product_id [PK] integer	product_name character varying (50)	type_product loaisp	brand character varying (20)	color character varying (20)	price numeric (9)	quantity_prod numeric (5)	time_warrant_prod numeric (3)
1	4	iPhone 14	dien thoai	Apple	Blue	23990000	45	12
2	3	iPhone 14	dien thoai	Apple	Red	23990000	24	12
3	1	iPhone 14	dien thoai	Apple	Black	23990000	50	12
4	2	iPhone 14	dien thoai	Apple	White	23990000	39	12

#### 2. Viết câu truy vấn in ra danh sách của các loại điện thoại theo giá tiền giảm dần

Query Query History

```
1 SELECT *
2 FROM products
3 WHERE type_product = 'dien thoai'
4 ORDER BY price DESC;
5
```

Data Output Messages Notifications



	product_id [PK] integer	product_name character varying (50)	type_product loaisp	brand character varying (20)	color character varying (20)	price numeric (9)	quantity_prod numeric (5)	time_warrant_prod numeric (3)
1	35	Galaxy Z Fold 3	dien thoai	Samsung	Phantom Black	42990000	35	12
2	36	Galaxy Z Fold 3	dien thoai	Samsung	Phantom White	42990000	25	12
3	11	iPhone 14 Pro Max	dien thoai	Apple	Blue	32990000	70	12
4	9	iPhone 14 Pro Max	dien thoai	Apple	Gold	32990000	35	12
5	10	iPhone 14 Pro Max	dien thoai	Apple	White	32990000	50	12
6	34	Galaxy Note 20 Ultra	dien thoai	Samsung	Mystic White	28990000	42	12
7	6	iPhone 14 Pro	dien thoai	Apple	White	28990000	50	12
8	5	iPhone 14 Pro	dien thoai	Apple	Silver	28990000	26	12
9	7	iPhone 14 Pro	dien thoai	Apple	Red	28990000	20	12
10	8	iPhone 14 Pro	dien thoai	Apple	Blue	28990000	60	12
11	32	Galaxy S23 Ultra	dien thoai	Samsung	Phantom Silver	27990000	35	12
12	31	Galaxy S23 Ultra	dien thoai	Samsung	Phantom Navy	27990000	45	12
13	37	Galaxy Z Flip 3	dien thoai	Samsung	Cream	25990000	35	12
14	4	iPhone 14	dien thoai	Apple	Blue	23990000	45	12
15	29	Galaxy S23+	dien thoai	Samsung	Phantom Silver	23990000	20	12
16	30	Galaxy S23+	dien thoai	Samsung	Phantom Black	23990000	20	12
17	33	Galaxy Note 20	dien thoai	Samsung	Mystic Bronze	23990000	25	12
18	3	iPhone 14	dien thoai	Apple	Red	23990000	24	12

(S) > Schemas > public > Tables > products 0:00.068

Ln 5, Col 1

### 3. Viết câu truy vấn in ra danh sách của các loại phụ kiện theo giá tiền giảm dần

Query Query History

Execute script  
(F5)

```
1 SELECT *
2 FROM products
3 WHERE type_product = 'phu kien' -- hoặc giá trị tương ứng với loại phụ kiện trong bảng của bạn
4 ORDER BY price DESC;
5
6
```

Data Output Messages Notifications



	product_id [PK] integer	product_name character varying (50)	type_product loaisp	brand character varying (20)	color character varying (20)	price numeric (9)	quantity_prod numeric (5)	time_warrant_prod numeric (3)
1	55	AirPods Max	phu kien	Apple	Space Gray	12990000	50	12
2	21	Apple Watch Series 7	phu kien	Apple	Black	11990000	49	12
3	56	Apple Watch Series 6	phu kien	Apple	Silver	9990000	70	12
4	66	Galaxy Watch 3	phu kien	Samsung	Black	8490000	70	12
5	41	Galaxy Watch 4 Classic	phu kien	Samsung	Silver	8490000	50	12
6	22	Apple Watch SE	phu kien	Apple	White	7990000	60	12
7	67	Galaxy Watch Active 2	phu kien	Samsung	Silver	7490000	100	12
8	40	Galaxy Watch 4	phu kien	Samsung	Black	6490000	50	12
9	23	AirPods Pro	phu kien	Apple	White	5990000	100	12
10	42	Galaxy Buds Pro	phu kien	Samsung	Phantom Black	4490000	100	12
11	54	AirPods 2	phu kien	Apple	White	3990000	300	12
12	24	AirPods	phu kien	Apple	White	3990000	80	12
13	65	Galaxy Buds Live	phu kien	Samsung	Bronze	3790000	250	12
14	43	Galaxy Buds Live	phu kien	Samsung	Mystic Bronze	3790000	90	12
15	64	Galaxy Buds 2	phu kien	Samsung	White	3490000	300	12
16	47	Samsung Keyboard Cover	phu kien	Samsung	Gray	3490000	50	24
17	26	Magic Keyboard	phu kien	Apple	Black	3199000	70	24
18	25	Apple Pencil	phu kien	Apple	White			

✓ Successfully run. Total query runtime: 98 msec. 41 rows affected. ✕

(S) > Schemas > public > Tables > products 0:00.098

Ln 6, Col 1

#### 4. Viết câu truy vấn in ra lịch sử mua hàng của 1 khách hàng

Query Query History

```
1 SELECT o.order_id, o.order_date, o.total_price
2 FROM orders o
3 INNER JOIN customers c ON o.customer_id=c.customer_id
4 WHERE c.phone_number= '0828658596' ;
5
6
```

Data Output Messages Notifications

	order_id [PK] integer	order_date date	total_price integer
1	103	2024-06-09	59970000

MS) > Schemas > public > Tables > products 00.047 Ln 1, Col 28

#### 5. Viết câu truy vấn/hàm in thời điểm mua của 1 chiếc điện thoại thông qua số seri

```

1 SELECT
2   o.order_date AS "Thời Điểm Mua"
3 FROM
4   order_details od
5 JOIN
6   orders o ON od.order_id = o.order_id
7 WHERE
8   od.ser_i = '09WX34N07890';
9
10

```

Data Output Messages Notifications

	Thời Điểm Mua date
1	2023-06-05

✓ Successfully run. Total query runtime: 52 msec. 1 rows affected. ✕

Total rows: 1 of 1 Query complete 00:00:00.052 Ln 8, Col 7

## 6. .Viết câu truy vấn in ra những phụ kiện hoặc điện thoại vẫn đang sửa

Query Query History

```

1 SELECT
2   wp.warrant_products_id AS "Mã Sản Phẩm Bảo Hành",
3   p.product_id AS "Mã Sản Phẩm",
4   p.product_name AS "Tên Sản Phẩm",
5   p.type_product AS "Loại Sản Phẩm",
6   wp.date_nhan AS "Ngày Nhận",
7   wp.tinhtrang AS "Tình Trạng"
8 FROM
9   warrant_products wp
10 JOIN
11   products p ON wp.product_id = p.product_id
12 WHERE
13   wp.tinhtrang = 'đang sửa';
14

```

Data Output Messages Notifications

	Mã Sản Phẩm Bảo Hành integer	Mã Sản Phẩm integer	Tên Sản Phẩm character varying (50)	Loại Sản Phẩm loaisp	Ngày Nhận date	Tình Trạng tinh_trang_enum
1	11	12	iPhone 13	dien thoai	2023-07-30	đang sửa
2	18	29	Galaxy S23+	dien thoai	2023-08-06	đang sửa
3	3	2	iPhone 14	dien thoai	2023-07-22	đang sửa

Total rows: 3 of 3 Query complete 00:00:00.037 Ln 14, Col 1

## 7. .Viết câu truy vấn tính tổng doanh thu theo tháng

[Data Output](#)   [Messages](#)   [Notifications](#)

Total rows: 24 of 24      Query complete 00:00:00.045      Ln 12, Col 1

**F5**

Data Output Messages Notifications

Total rows: 1000 of 1259      Query complete 00:00:00.146      Ln 9, Col 1

## 9. Truy vấn đánh giá doanh thu từ các khách hàng VIP so với các khách hàng thông thường:

Query		Query History	
1	SELECT		
2	c.membership_level,		
3	SUM(o.total_price) AS total_revenue		
4	FROM		
5	customers c		
6	JOIN		
7	orders o ON c.customer_id = o.customer_id		
8	WHERE		
9	o.payment_status = 'TRUE'		
10	GROUP BY		
11	c.membership_level;		
12			

Data Output		Messages		Notifications	
membership_level	total_revenue				
character varying (50)	bigint				
1	S_vip				
2	S_new				
3	S_mem				

Total rows: 3 of 3    Query complete 00:00:00.384    Ln 6, Col 6

## 10. Truy vấn chi tiết đơn hàng cho một order\_id cụ thể:

Query

Query History

1

2

3

4

5

6

7

8

9

SELECT

od.orderdetail\_id, od.order\_id, p.product\_id, p.product\_name, p.brand,p.color,p.price,od.seri

FROM

order\_details od

JOIN

products p ON od.product\_id = p.product\_id

WHERE

od.order\_id = 36;

Data Output

Messages

Notifications

orderdetail\_id

integer

order\_id

integer

product\_id

integer

product\_name

character varying (50)

brand

character varying (20)

color

character varying (20)

price

numeric (9)

seri

character

1

115

36

5

iPhone 14 Pro

Apple

Silver

28990000

ahn12345

2

116

36

21

Apple Watch Series 7

Apple

Black

11990000

[null]

3

117

36

22

Apple Watch SE

Apple

White

7990000

[null]

## 11. Truy vấn danh sách các sản phẩm có số lượng trong kho nhỏ hơn 30

Query

Query History

1

SELECT

2

p.product\_id, p.product\_name, p.type\_product, p.brand, p.color, p.price, p.quantity\_prod

3

FROM

4

products p

5

WHERE

6

p.quantity\_prod < 30;

7

8

Data Output

Messages

Notifications

	product_id [PK] integer	product_name character varying (50)	type_product loaisp	brand character varying (20)	color character varying (20)	price numeric (9)	quantity_prod numeric (5)
1	3	iPhone 14	dien thoai	Apple	Red	23990000	25
2	5	iPhone 14 Pro	dien thoai	Apple	Silver	28990000	26
3	7	iPhone 14 Pro	dien thoai	Apple	Red	28990000	20
4	13	iPhone 13	dien thoai	Apple	Blue	17990000	20
Total rows: 8 of 8							

Query complete 00:00:00.202

Ln 8, Col 1

12 . Truy vấn danh sách các khách hàng có tổng số tiền đã chi tiêu cao nhất

Query

Query History

1

SELECT

2

customer\_id,

3

customer\_name,

4

phone\_number,

5

email,

6

total\_order

7

FROM

8

customers

9

ORDER BY

10

total\_order DESC

11

LIMIT 10;

Data Output

Messages

Notifications

≡

+

📄

▼

📋

▼

🗑️

📦

⬇️

📈

	customer_id [PK] integer	customer_name character varying (50)	phone_number character varying (15)	email character varying (60)	total_order integer
1	83	Trương Thị Ngọc	0925432109	truongthingoc@gmail.com	485840000
2	70	Trần Thị Hồng	0978765432	tranthihong@gmail.com	459750000
3	71	Lê Văn Đức	0967654321	levanduc@gmail.com	459750000
4	79	Nguyễn Thị Thúy	0969876543	nguyenthithuy@gmail.com	450676000
5	73	Hoàng Văn Thành	0945432109	hoangvanthanh@gmail.com	450340000
6	78	Trần Văn Dũng	0970987654	tranvandung@gmail.com	435295000

13 . Truy vấn danh sách các đơn hàng của một khách hàng dựa trên customer\_id:



Query		Query History						
1	SELECT							
2		o.order_id, o.order_date, o.total_price, o.payment_status, o.delivery_status, od.product_id,						
3		p.product_name, od.seri						
4	FROM							
5		orders o						
6	JOIN							
7		order_details od ON o.order_id = od.order_id						
8	JOIN							
9		products p ON od.product_id = p.product_id						
10	WHERE							
11		o.customer_id = '3';						

Data Output		Messages						
	order_id	order_date	total_price	payment_status	delivery_status	product_id	product_name	seri
	integer	date	integer	boolean	giaohang	integer	character varying (50)	character varying (20)
1	38	2023-01-15	31188000	true	đã giao	5	iPhone 14 Pro	ahn345678901
2	38	2023-01-15	31188000	true	đã giao	61	Galaxy S23 Ultra Case	[null]
3	38	2023-01-15	31188000	true	đã giao	62	Galaxy Note 20 Case	[null]

## 14. Truy vấn thông tin khách hàng có số đơn hàng đã đặt lớn hơn 2

Query		Query History						
1	SELECT							
2		c.customer_id, c.customer_name, c.phone_number, c.date_of_birth, c.customer_address, c.email,						
3		c.total_order, c.membership_level,						
4		COUNT(o.order_id) AS order_count						
5	FROM							
6		customers c						
7	JOIN							
8		orders o ON c.customer_id = o.customer_id						
9	GROUP BY							
10		c.customer_id, c.customer_name, c.phone_number, c.date_of_birth, c.customer_address, c.email, c.total_orc						
11	HAVING							
12		COUNT(o.order_id) > 1;						

Data Output		Messages						
	customer_id	customer_name	phone_number	date_of_birth	customer_address	email	total_orde	
	[PK] integer	character varying (50)	character varying (15)	date	text	character varying (60)	integer	
1	206	Trần Thị Kim	0958765131	1993-02-14	21010 Đường Trần Phú, Hải Phòng	tranth2ikim@gmail.com	1588	

## 15. Truy vấn sản phẩm nào được bảo hành nhiều nhất

Query

Query History

1

2

3

4

5

6

7

8

9

10

11

12

13

SELECT

p.product\_id,

p.product\_name,

COUNT(wp.product\_id) AS warrant\_count

FROM

products p

JOIN

warrant\_products wp ON p.product\_id = wp.product\_id

GROUP BY

p.product\_id, p.product\_name

ORDER BY

warrant\_count DESC

LIMIT 1;

Data Output

Messages

Notifications

product\_id

[PK] integer

product\_name

character varying (50)

warrant\_count

bigint

1

4

iPhone 14

151

## 16 . Truy vấn danh sách sản phẩm trong giỏ hàng của một khách hàng cụ thể theo customer\_id

Query

Query History

1

SELECT

2

sc.Sh\_cart\_id,

3

sc.customer\_id,

4

sc.product\_id,

5

p.product\_name,

6

p.brand,

7

p.price,

8

sc.Sh\_product\_quantity

9

FROM

10

Shopping\_cart sc

11

JOIN

12

products p ON sc.product\_id = p.product\_id

13

WHERE

14

sc.customer\_id =30;

Data Output

Messages

Notifications

	sh_cart_id integer	customer_id integer	product_id integer	product_name character varying (50)	brand character varying (20)	price numeric (9)	sh_product_quantity integer
1	259	30	3	iPhone 14	Apple	23990000	3
2	254	30	55	AirPods Max	Apple	12990000	1
3	255	30	60	Galaxy S23+ Case	Samsung	1199000	3

Total rows: 6 of 6

Query complete 00:00:00.128

Ln 1, Col 8

## 17. Truy vấn danh sách các đơn hàng trong khoảng thời gian nhất định

Query

Query History

1

SELECT

2

order\_id,

3

customer\_id,

4

order\_date,

5

total\_price,

6

payment\_status,

7

delivery\_status

8

FROM

9

orders

10

WHERE

11

-- order\_date BETWEEN '[start\_date]' AND '[end\_date]';

12

order\_date BETWEEN '1-1-2024' AND '1-8-2024';

Data Output

Messages

Notifications

	order_id [PK] integer	customer_id integer	order_date date	total_price integer	payment_status boolean	delivery_status giaohang
1	105	200	2024-06-08	0	true	da giao
2	103	207	2024-06-05	69649000	true	da giao
3	109	206	2024-06-09	0	false	da giao

Total rows: 5 of 5

Query complete 00:00:00.244

Ln 11, Col 6

Ln 11, Col 6

## 18 . Truy vấn danh sách các khách hàng sinh nhật trong tháng hiện tại

Query

Query History

1

SELECT

2

customer\_id,

3

customer\_name,

4

phone\_number,

5

date\_of\_birth,

6

customer\_address,

7

email,

8

total\_order,

9

membership\_level

10

FROM

11

customers

12

WHERE

13

EXTRACT(MONTH FROM date\_of\_birth) = EXTRACT(MONTH FROM CURRENT\_DATE);

Data Output

Messages

Notifications

customer\_id

[PK] integer

customer\_name

character varying (50)

phone\_number

character varying (15)

date\_of\_birth

date

customer\_address

text

email

character varying (60)

1

30

Oscar Price

447951123485

1983-06-25

753 Lady St, London

oscar.p@example.com

2

35

Lucas Foster

447951123490

1987-06-10

456 Viscount St, London

lucas.f@example.com

3

59

Trần Minh Quân

0962109876

1988-06-15

1111 Đường Lê Lợi, Phú Yên

tranminhquan@gmail.com

## 19 . Để truy vấn các sản phẩm được bán nhiều nhất trong tháng hiện tại

Query

Query History

```
1 SELECT | p.product_id, p.product_name,
2         COUNT(od.product_id) AS total_quantity_sold
3 FROM
4     order_details od
5 JOIN
6     orders o ON od.order_id = o.order_id
7 JOIN
8     products p ON od.product_id = p.product_id
9 WHERE
10    EXTRACT(YEAR FROM o.order_date) = EXTRACT(YEAR FROM CURRENT_DATE)
11    AND EXTRACT(MONTH FROM o.order_date) = EXTRACT(MONTH FROM CURRENT_DATE)
12    AND o.payment_status = 'TRUE'
13 GROUP BY
14    p.product_id, p.product_name
15
```

Data Output

Messages

Notifications

	product_id [PK] integer	product_name character varying (50)	total_quantity_sold bigint
1	61	Galaxy S23 Ultra Case	2
2	40	Galaxy Watch 4	2
3	63	Galaxy Z Fold 3 Case	2

## 20 . Truy vấn danh sách các sản phẩm đang trong quá trình bảo hành dựa trên số điện thoại khách hàng

Query

Query History

1

SELECT wp.warrant\_products\_id,wp.customer\_id, c.customer\_name,

2

3

wp.product\_id,p.product\_name, wp.seri, wp.date\_nhan, wp.date\_tra,wp.tinhtrang

4

FROM warrant\_products wp

5

JOIN

6

customers c ON wp.customer\_id = c.customer\_id

7

JOIN

8

products p ON wp.product\_id = p.product\_id

9

WHERE

10

c.phone\_number = '0925432109';

Data Output

Messages

Notifications

	warrant_products_id integer	customer_id integer	customer_name character varying (50)	product_id integer	product_name character varying (50)	seri character varying (20)	date_nhan date	date_tra date
1	36	83	Trương Thị Ngọc	8	iPhone 14 Pro	C8YZ12QR8945	2023-08-24	2023-12-10
2	37	83	Trương Thị Ngọc	10	iPhone 14 Pro Max	U5QR56ST5678	2023-08-25	2023-12-11
3	38	83	Trương Thị Ngọc	10	iPhone 14 Pro Max	V5ST78UV1234	2023-08-26	2023-12-12
4	114	83	Trương Thị Ngọc	8	iPhone 14 Pro	C8YZ12QR8945	2023-12-10	2023-12-11

## III.4. Tối ưu hóa tốc độ truy vấn dữ liệu bằng INDEX

### 1. Bảng products

**Nhận xét :** Bảng products hay được sử dụng để truy vấn sản phẩm và loại sản phẩm để cho khách hàng và nhân viên tìm kiếm.

**Tạo INDEX cho cột *product\_name* và *brand***

**Số dòng đánh index là 1104 dòng**









```
CREATE INDEX name_index ON products (product_name);
```

```
CREATE INDEX type_index ON products (brand);
```

**Trước khi sử dụng INDEX:**

```
EXPLAIN ANALYZE  
SELECT* FROM products  
WHERE product_name='Vertu Microphone';
```

ta Output Messages Notifications

							
QUERY PLAN							
text							
Seq Scan on products (cost=0.00..28.80 rows=1 width=54) (actual time=0.111..0.122 rows=1 loops...							
Filter: ((product_name)::text = 'Vertu Microphone'::text)							
Rows Removed by Filter: 1103							
Planning Time: 0.867 ms							
Execution Time: 0.139 ms							

⇒ Execution Time thực hiện câu lệnh là 0.139 ms với phương pháp quét tuần tự

```
EXPLAIN ANALYZE
SELECT* FROM products
WHERE brand='APPLE' ;
```

a Output Messages Notifications

	<b>QUERY PLAN</b> text
	Seq Scan on products (cost=0.00..28.80 rows=1 width=54) (actual time=0.118..0.118 rows=0 loops=...
	Filter: ((brand)::text = 'APPLE'::text)
	Rows Removed by Filter: 1104
	Planning Time: 0.594 ms
	Execution Time: 0.129 ms

⇒ Execution Time thực hiện câu lệnh là 0.129 ms với phương pháp quét tuần tự

## Sau khi tạo INDEX:

```
EXPLAIN ANALYZE
SELECT* FROM products
WHERE product_name='Vertu Microphone';
```

a Output Messages Notifications

	<b>QUERY PLAN</b> text
	Index Scan using name_index on products (cost=0.28..8.29 rows=1 width=54) (actual time=0.026..0.026 rows=1 loop=...
	Index Cond: ((product_name)::text = 'Vertu Microphone'::text)
	Planning Time: 1.444 ms
	Execution Time: 0.039 ms

⇒ Execution Time thực hiện câu lệnh là 0.039 ms với phương pháp sử dụng INDEX

⇒ Tốc độ truy vấn tăng gấp 4 lần

```
EXPLAIN ANALYZE
SELECT * FROM products
WHERE brand='APPLE' ;
```

Output Messages Notifications



QUERY PLAN	
text	
Index Scan using type_index on products (cost=0.15..8.17 rows=1 width=54) (actual time=0.030..0.031 rows=0 loops...	
Index Cond: ((brand)::text = 'APPLE'::text)	
Planning Time: 1.537 ms	
Execution Time: 0.058 ms	

- ⇒ Execution Time thực hiện câu lệnh là 0.058 ms với phương pháp sử dụng INDEX
- ⇒ Tốc độ truy vấn tăng gấp 2 lần

## 2. Bảng phone\_detail\_products

Tạo index cho thuộc tính *seri* (do khai báo seri là UNIQUE NOT NULL nên hệ thống sẽ tự tạo index cho thuộc tính *seri* để tăng tốc độ truy vấn.

**Số dòng đánh index là : 1365 dòng**

```
EXPLAIN ANALYZE
SELECT * FROM phone_detail_products
where seri='VETANH05012005';
```

Output Messages Notifications



QUERY PLAN	
text	
Index Scan using phone_detail_products_seri_key on phone_detail_products (cost=0.28..8.29 rows=1 width=21) (actual time=0.025..0.026 rows=1 loop...	
Index Cond: ((seri)::text = 'VETANH05012005'::text)	
Planning Time: 0.927 ms	
Execution Time: 0.043 ms	

### 3. Bảng customers

**Nhận xét :** Tương tự bảng *phone\_detail\_products* thì thuộc tính *phone\_number* và *email* cũng được hệ thống tự tạo index để tối ưu tốc độ truy vấn khi cần tìm kiếm thông tin khách hàng thông qua số điện thoại hay email

**Số dòng được đánh index là: 207 dòng**

```
EXPLAIN ANALYZE
SELECT * FROM customers
where phone_number='0828658596';
```

a Output Messages Notifications



QUERY PLAN	🔒
text	
Index Scan using numner_index on customers (cost=0.00..8.02 rows=1 width=115) (actual time=0.023..0.024 rows=1 loop...	
Index Cond: ((phone_number)::text = '0828658596'::text)	
Planning Time: 0.132 ms	
Execution Time: 0.042 ms	

### 4. Bảng order

**Nhận xét :** Do bảng order thường được sử dụng hàng ngày để tạo đơn hàng nên không tạo index sẽ làm chậm quá trình INSERT hay UPDATE khi thao tác trên bảng

### 5. Bảng detail orders

**Nhận xét :** Tương tự như bảng *orders* thì bảng *detail\_orders* cũng được INSERT và UPDATE hằng ngày nên ta cũng không tạo INDEX cho bảng.

### 6. Bảng warrant products

**Nhận xét :** Do bảng bảo hành không thường được INSERT cũng như UPDATE nên ta có thể thêm INDEX để tăng hiệu quả truy vấn. Tuy nhiên



```
CREATE INDEX warrant_index ON warrant_products (product_id);
CREATE INDEX idx_warrant_products_product_id_customer_id ON warrant_products (product_id, customer_id);
CREATE INDEX idx_warrant_products_customer_id_product_id ON warrant_products (customer_id, product_id);
```

```
EXPLAIN ANALYZE
SELECT *
FROM warrant_products w
INNER JOIN customers c
ON w.customer_id=c.customer_id
WHERE c.phone_number='0565432109' AND w.product_id=4;
```

ta Output Messages Notifications

QUERY PLAN	🔒
text	
Hash Join (cost=8.17..10.22 rows=1 width=139) (actual time=0.045..0.046 rows=1 loops=1)	
Hash Cond: (w.customer_id = c.customer_id)	
-> Seq Scan on warrant_products w (cost=0.00..1.98 rows=26 width=24) (actual time=0.012..0.014 rows=26 loops=1)	
Filter: (product_id = 4)	
Rows Removed by Filter: 52	
-> Hash (cost=8.16..8.16 rows=1 width=115) (actual time=0.021..0.022 rows=1 loops=1)	
Buckets: 1024 Batches: 1 Memory Usage: 9kB	
-> Index Scan using customers_phone_number_key on customers c (cost=0.14..8.16 rows=1 width=115) (actual time=0.018..0.019 rows=1 loop=1)	
Index Cond: ((phone_number)::text = '0565432109'::text)	
Planning Time: 1.625 ms	
Execution Time: 0.076 ms	

Khi tạo INDEX cho hai thuộc tính *product\_id* và *customer\_id* thì PostgreSQL không sử dụng để tối ưu hóa truy vấn và quét tuần tự (do HQTCSDL cho rằng quét tuần tự sẽ nhanh hơn do số lượng bản ghi không quá lớn) nên khi số lượng bản ghi lớn thì INDEX sẽ tối ưu truy vấn.

**Khi tăng số lượng bản ghi lên**

**Số dòng được đánh INDEX là : 537**

**Trước khi có INDEX**

Query Query History

```

1 create index ctmid on warrant_products using hash(customer_id);
2 drop index ctmid;
3 explain analyze
4 select * from warrant_products
5 where customer_id = 207;

```

Data Output Messages Notifications

QUERY PLAN

1	Seq Scan on warrant_products (cost=0.00..39.92 rows=1 width=37) (actual time=0.340..0.340 rows=0 loops=...
2	Filter: (customer_id = 207)
3	Rows Removed by Filter: 1914
4	Planning Time: 0.896 ms
5	Execution Time: 0.356 ms

Total rows: 5 of 5 Query complete 00:00:00.132 Ln 3, Col 1

⇒ Execution Time thực hiện câu lệnh là 0.356 ms với phương pháp quét tuần tự  
**Sau khi tạo INDEX**

Query Query History

```

1 create index ctmid on warrant_products using hash(customer_id);
2 drop index ctmid;
3 explain analyze
4 select * from warrant_products
5 where customer_id = 207;

```

Data Output Messages Notifications

QUERY PLAN

1	Index Scan using ctmid on warrant_products (cost=0.00..8.02 rows=1 width=37) (actual time=0.020..0.021 rows=0 loops=...
2	Index Cond: (customer_id = 207)
3	Planning Time: 1.630 ms
4	Execution Time: 0.042 ms

Query complete 00:00:00.132 Ln 3, Col 1

⇒ Execution Time thực hiện câu lệnh là 0.042 ms với phương pháp sử dụng INDEX  
 ⇒ Tốc độ truy vấn tăng gấp 9 lần

## **7. Bảng Shopping\_cart**

**Nhận xét :** Bảng Shopping\_cart thường xuyên được INSERT hoặc UPDATE và cũng không hay truy vấn trên bảng Shopping\_cart nên việc sử dụng INDEX là không cần thiết.

### **Đánh giá**

Cơ sở dữ liệu quản lý cửa hàng bán điện thoại và phụ kiện trên đáp ứng được cơ bản các thông tin để quản lý các sản phẩm trong kho, danh sách khách hàng, lịch sử mua hàng của khách hàng cũng như phục vụ được việc mua hàng online của khách hàng thông qua Internet.

Các hàm và trigger đã xây dựng sẽ giúp cho công việc nhập xuất hay kiểm tra truy xuất sẽ thuận tiện và tránh được các sai sót.

Trong trường hợp số lượng bản ghi lớn cũng có thể truy xuất nhanh chóng nhờ INDEX ở một số bảng

## KẾT LUẬN

CSDL phù hợp để thực hiện các tác vụ cơ bản để quản lý của một cửa hàng bán điện thoại và phụ kiện

Thu hoạch từ đồ án :

- + ) Biết cách thiết kế cơ sở dữ liệu cho một mô hình cụ thể trong thực tế , cụ thể ở đây là một cửa hàng bán điện thoại và phụ kiện. Và hoàn toàn bọn em có thể xây dựng được các mô hình khác trong thực tế dựa vào những kiến thức, kinh nghiệm và lời nhận xét của giảng viên trong đồ án trên.
- + ) Thành thạo ngôn ngữ SQL và sử dụng hệ quản trị cơ sở dữ liệu để xây dựng cơ sở dữ liệu dựa trên mô hình đã thiết kế. Các tác vụ như lập bảng, viết hàm, viết câu truy vấn nhóm chúng em cũng đã được rèn luyện rất nhiều thông qua đồ án môn học.
- + ) Biết cách phân tích mô hình CSDL với các khả năng, trường hợp có thể xảy ra đối với một mô hình cơ sở dữ liệu và cách giải quyết các khả năng, trường hợp đó
- + ) Nắm rõ được trình tự , các bước để xây dựng một CSDL
- + ) Biết cách đánh giá hiệu năng và sử dụng INDEX nhằm mục đích làm tăng tốc độ truy vấn. Với những cơ sở dữ liệu lớn, cần việc truy xuất dữ liệu thường xuyên thì việc đòi hỏi tối ưu thời gian là rất cần thiết, nhất là trong thời đại ngày nay. Thông qua đồ án cũng như tìm hiểu các kiến thức trên mạng, bọn em đã biết khi nào nên sử dụng INDEX, khi nào không nên sử dụng INDEX và các đánh giá hiệu quả của INDEX.
- + ) Bên cạnh đó, việc làm đồ án nhóm còn giúp bọn em cải thiện kỹ năng làm việc nhóm. Làm việc nhóm giúp bọn em học hỏi rất nhiều từ nhau và làm cho kiến thức chuyên môn trở nên vững chắc và sâu sắc hơn.

## **TÀI LIỆU THAM KHẢO**

- [1] Tài liệu: <https://www.postgresql.org/docs/13/sql-createfunction.html>
- [2] Tài liệu: <https://www.postgresql.org/docs/14/indexes.htm>
- [3] Tài liệu: <https://www.postgresql.org/docs/13/sqlcreatetrigger.html>