ĐẠI HỌC BÁCH KHOA HÀ NỘI

TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

BÁO CÁO PROJECT

ĐỀ TÀI: THIẾT KẾ CƠ SỞ DỮ LIỆU TRANG WEB BÁN GIÀY

MÔN HỌC: IT3290 – THỰC HÀNH CƠ SỞ DỮ LIỆU

GVHD: TS. NGUYỄN THỊ OANH; TS. TRẦN VĂN ĐẶNG

Nhóm	**6**		
Họ và tên	**MSSV**	**Mã lớp**	**Công việc**
Nguyễn Tài Hưng *(Leader)*	20236034	156784	Thiết kế database, vẽ ERD, vẽ RSD, viết trigger, function, view, truy vấn, làm báo cáo, thuyết trình.
Trần Việt Gia Khánh	20235756	156784	Viết trigger, function, truy vấn, làm báo cáo.
Nguyễn Trung Kiên	20235759	156784	Viết truy vấn, làm báo cáo.

A. MÔ TẢ NGHIỆP VỤ

1. Đặt vấn đề

   Trong bối cảnh thị trường thương mại điện tử ngày càng phát triển mạnh mẽ, đặc biệt là lĩnh vực thời trang và phụ kiện, việc sở hữu một trang web bán giày trực tuyến hiệu quả đã trở thành một yếu tố then chốt để các doanh nghiệp tiếp cận khách hàng tiềm năng và gia tăng doanh số. Tuy nhiên, sự đa dạng về mẫu mã, kích cỡ, nhà cung cấp và thông tin khách hàng đặt ra một thách thức không nhỏ trong việc quản lý dữ liệu một cách khoa học và hệ thống. Chính vì lẽ đó, dự án "Xây dựng cơ sở dữ liệu cho trang web bán giày" được đề xuất nhằm giải quyết bài toán này, tạo nền tảng vững chắc cho việc vận hành và phát triển trang web một cách bền vững.

2. Mô tả nghiệp vụ

   a. Quản lí sản phẩm

      - Doanh nghiệp nhập giày từ các Thương hiệu và phân loại chúng theo Danh mục.
      - Mỗi sản phẩm có thể có nhiều Biến thể khác nhau về kích cỡ và màu sắc.
      - Nhân viên có thể chọn thêm mới hoặc cập nhật sản phẩm để thêm sản phẩm mới, cập nhật thông tin, giả cả, mô tả,…
      - Hệ thống theo dõi số lượng tồn kho của từng biến thể sản phẩm để đảm bảo chỉ bán các mặt hàng có sẵn.

   b. Quản lý khách hàng

      - Thêm/sửa/xóa thông tin khách hàng.
      - Hỗ trợ cấp/đổi lại mật khẩu/thông tin khách hàng nếu cần thiết.

   c. Quản lý doanh thu, lợi nhuận

      - Hệ thống cần quản lý tổng doanh thu đơn hàng (chưa áp voucher, đã app voucher) thống kê theo từng ngày, tuần, tháng, quý, năm.
      - Hệ thống quản lý số tiền nhận được của mỗi đơn hàng, cập theo thời gian thực.

d. Quá trình Bán hàng:

- Khách hàng tiềm năng tìm kiếm và duyệt xem các sản phẩm trên trang web.
- Khách hàng lựa chọn Biến thể sản phẩm (kích cỡ, màu sắc) và thêm vào giỏ hàng.
- Khách hàng tiến hành Đặt hàng, cung cấp thông tin giao hàng và thanh toán.
- Hệ thống tạo một Đơn hàng, ghi lại các Chi tiết đơn hàng cụ thể (sản phẩm, số lượng, giá tại thời điểm đặt) và trừ số lượng tồn kho tương ứng.
- Khách hàng thực hiện Thanh toán cho đơn hàng thông qua các phương thức được hỗ trợ.

e. Xử lý và Thực hiện Đơn hàng:

- Nhân viên quản lý đơn hàng tiếp nhận các Đơn hàng mới.
- Nhân viên xác nhận tính hợp lệ của đơn hàng và thông tin thanh toán.
- Đơn hàng được chuẩn bị (nhặt hàng từ kho, đóng gói).
- Quá trình Vận chuyển được khởi tạo, bàn giao gói hàng cho đơn vị vận chuyển và cập nhật mã theo dõi.
- Trạng thái Đơn hàng được cập nhật qua các giai đoạn (đang xử lý, đã đóng gói, đang vận chuyển, đã giao...).

B. MÔ TẢ ỨNG DỤNG

1. Kịch bản sử dụng

    - Khách hàng: Khách hàng tìm kiếm, lựa chọn sản phẩm của cửa hàng, xác nhận đặt mua, thanh toán, viết đánh giá, liên hệ hỗ trợ, theo dõi tình trạng đơn hàng.
    - Nhân viên: Quản lý thông tin khách hàng, quản lý đơn hàng, tạo đơn vận chuyển, hỗ trợ khách hàng.
    - Quản lý: Quản lý thông tin khách hàng, quản lý voucher, quản lý kho, quản lý sản phẩm, thống kê doanh thu, lợi nhuận.

2. Mô tả chức năng

    a. Khách hàng

    - **Tìm kiếm và lọc sản phẩm:** Tìm kiếm sản phẩm theo từ khóa. Lọc sản phẩm theo các tiêu chí (giá, kích cỡ, màu sắc, thương hiệu, loại). Sắp xếp sản phẩm theo các tiêu chí (giá tăng/giảm, mới nhất, bán chạy).
    - **Xem sản phẩm:** Xem chi tiết sản phẩm (tên, mã, thương hiệu, loại, kích cỡ, màu sắc, chất liệu, giá, mô tả, hình ảnh). Xem review, rating của sản phẩm. Xem các sản phẩm liên quan hoặc gợi ý.
    - **Giỏ hàng:** Thêm sản phẩm vào giỏ hàng. Xem danh sách sản phẩm trong giỏ hàng. Thay đổi số lượng sản phẩm trong giỏ hàng. Xóa sản phẩm khỏi giỏ hàng. Cập nhật tổng tiền giỏ hàng.
    - **Đặt hàng:** Nhập thông tin giao hàng (tên, địa chỉ, số điện thoại). Chọn phương thức thanh toán. Xem lại thông tin đơn hàng trước khi xác nhận. Xác nhận đặt hàng. Thanh toán (nếu là phương thức dùng thẻ).
    - **Đánh giá sản phẩm:** Chọn vào danh sách các đơn hàng đã mua thành công. Chọn phần đánh giá. Viết đánh giá, thêm hình ảnh/video sản phẩm, đánh giá trên thang 5 điểm sản phẩm.
    - **Quản lý tài khoản:** Đăng ký tài khoản mới. Đăng nhập/Đăng xuất. Xem và cập nhật thông tin cá nhân. Xem lịch sử đơn hàng và trạng thái đơn hàng. Hủy đơn hàng (trong thời gian cho phép).

b. Nhân viên

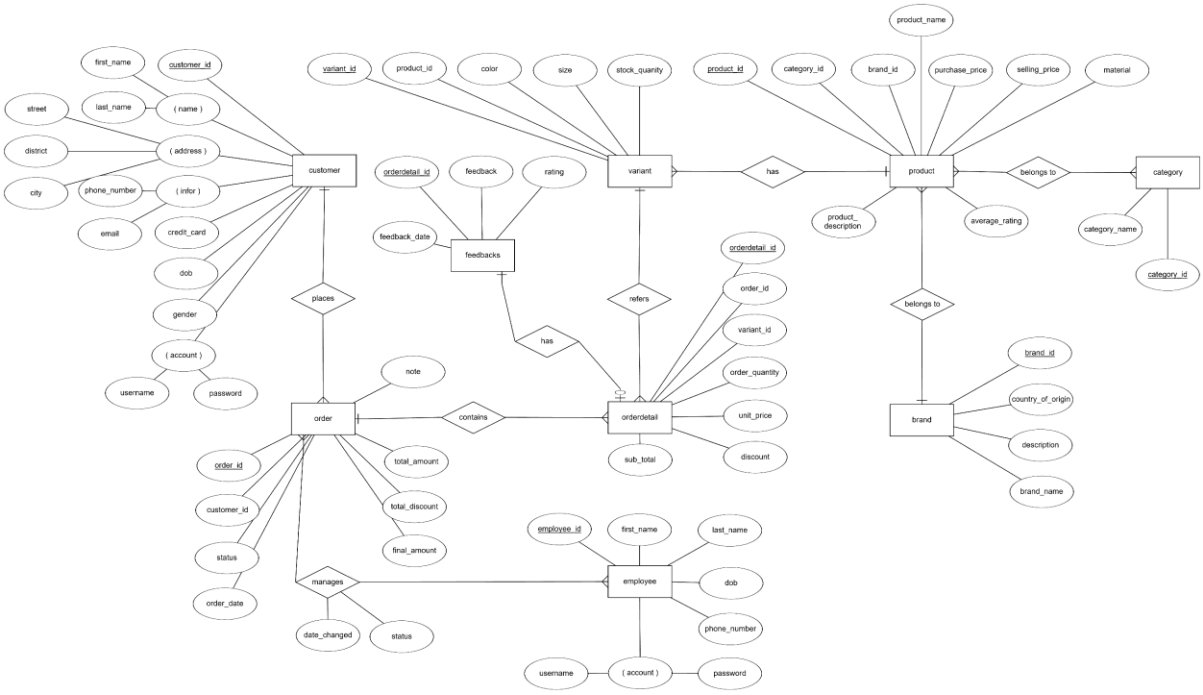
- **Quản lý sản phẩm (với quyền hạn):** Xem danh sách sản phẩm. Tìm kiếm và lọc sản phẩm. Xem chi tiết sản phẩm. Cập nhật thông tin sản phẩm (số lượng tồn kho, hình ảnh).
- **Quản lý đơn hàng:** Xem danh sách đơn hàng mới. Xem chi tiết đơn hàng. Xác nhận/ Từ chối đơn hàng. Cập nhật trạng thái đơn hàng (đã đặt hàng, đã xác nhận, đã gửi hàng, đang giao, đã giao). In hóa đơn.
- **Quản lý khách hàng:** Xem danh sách khách hàng. Xem thông tin chi tiết của khách hàng. Xem lịch sử mua hàng. Hỗ trợ khách hàng (VD: trả lời thắc mắc về đơn hàng).
- **Quản lý kho hàng (với quyền hạn):** Xem số lượng tồn kho.
- **Quảng lý tài khoản:** Đăng nhập/Đăng xuất. Xem và cập nhật thông tin cá nhân.

c. Quản lý

- **Quản lý sản phẩm:** Thêm sản phẩm mới. Cập nhật thông tin sản phẩm (toàn bộ thông tin). Xóa sản phẩm.
- **Quản lý kho hàng:** Xem số lượng tồn kho. Thêm sản phẩm mới. Điều chỉnh số lượng tồn kho. Xem báo cáo tồn kho.
- **Quản lý đơn hàng:** Xem toàn bộ danh sách đơn hàng. Xem chi tiết đơn hàng. Xác nhận đơn hàng. Hủy đơn hàng. Cập nhật trạng thái đơn hàng.
- **Quản lý người dùng:** Thêm tài khoản nhân viên mới. Sửa đổi thông tin tài khoản nhân viên. Xóa tài khoản nhân viên. Phân quyền truy cập cho nhân viên.
- **Báo cáo và thống kê:** Xem báo cáo doanh số (theo thời gian, sản phẩm, khách hàng, …). Xem thống kê sản phẩm bán chạy, sản phẩm tồn kho. Xuất dữ liệu báo cáo.

C. SƠ ĐỒ THỰC THỂ LIÊN KẾT

1. Xác định các thực thể chính
    - Sản phẩm
    - Biến thể của sản phẩm
    - Phân loại
    - Nhãn hàng
    - Đơn hàng
    - Chi tiết đơn hàng
    - Khách hàng
    - Nhân viên
2. Sơ đồ



(đính kèm file png)

D. SƠ ĐỒ QUAN HỆ

1. Xác định mối quan hệ

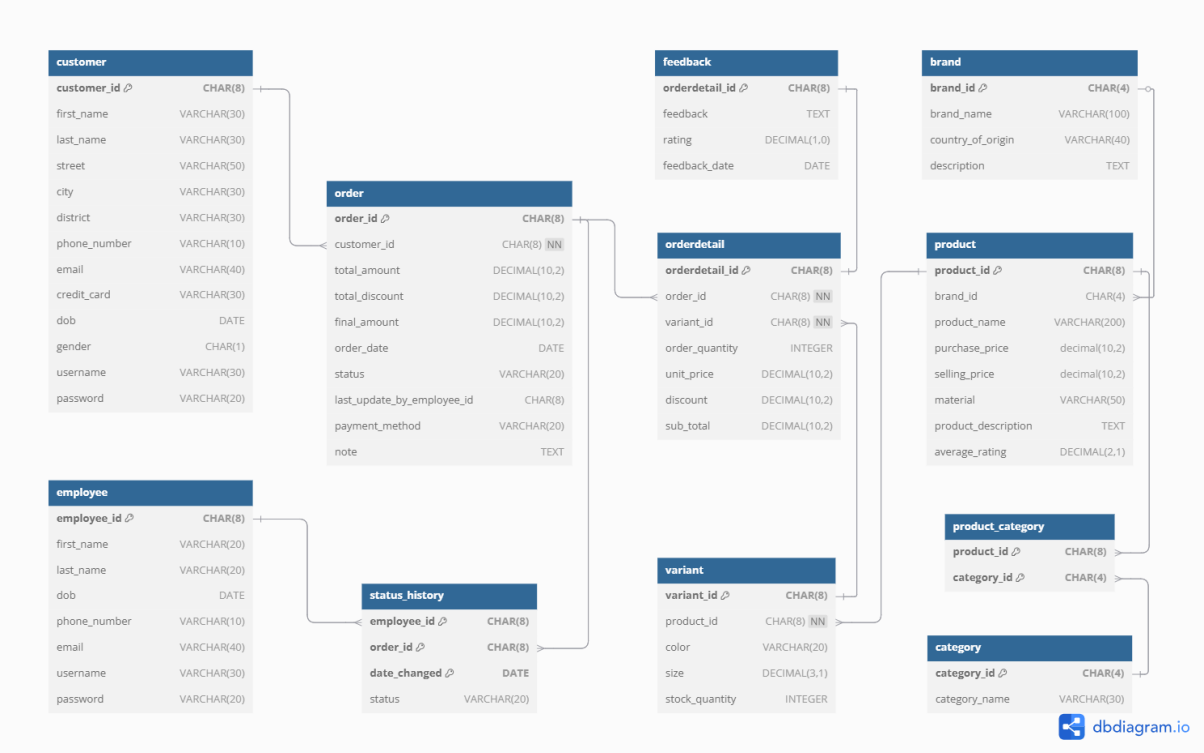
Thực thể A	Thực thể B	Quan hệ	Mô tả liên kết
Customer	Order	1 – n	Một khách hàng có thể đặt nhiều đơn hàng, mỗi đơn hàng chỉ được đặt bởi một khách hàng.
Order	Orderdetail	1 – n	Mỗi đơn hàng có thể chứa nhiều chi tiết đơn hàng, mỗi chi tiết đơn hàng chỉ thuộc về một đơn hàng.
Employee	Order	n – n	Mỗi nhân viên có thể xử lý nhiều đơn hàng, mỗi đơn hàng có thể được xử lý bởi nhiều nhân viên.
Variant	Orderdetail	1 – n	Mỗi biến thể có thể thuộc nhiều chi tiết đơn hàng, mỗi chi tiết đơn hàng chỉ có một biến thể.
Product	Variant	1 – n	Một sản phẩm có thể có nhiều tùy chọn khác nhau (biến thể khác nhau), mỗi biến thể chỉ thuộc (là) một sản phẩm.
Category	Product	n – n	Mỗi danh mục có thể có nhiều sản phẩm, mỗi sản phẩm có thể thuộc nhiều danh mục.
Brand	Product	1 – n	Một nhãn hàng có thể có nhiều sản phẩm, mỗi sản phẩm chỉ thuộc một nhãn hàng.
Feedback	Orderdetail	1 – 1 hoặc 0 – 1	Một chi tiết đơn hàng chỉ tương ứng duy nhất một đánh giá và ngược lại, hoặc không được đánh giá.

2. Xác định các bảng

- category(**category_id**, category_name)
- brand(**brand**, brand_name, country_of_origin, brand_description)
- product(**product_id**, *brand_id*, product_name, purchase_price, selling_price, material, product_description, average_rating)
- product_category(***product_id***, ***category_id***) (product_id, category_id là **FK)**
- variant(**variant_id**, *product_id*, color, size, stock_quantity)

- customer(**<u>customer_id</u>**, first_name, last_name, street, city, district, phone_number, email, credit_card, dob, gender, username, password)
- feedback(***<u>orderdetail_id</u>***, feedback, rating, feedback_date) (orderdetail_id là **FK**)
- orderdetail(**<u>orderdetail_id</u>**, *order_id*, *variant_id*, order_quantity, unit_price, discount, sub_total)
- order(**<u>order_id</u>**, *customer_id*, total_amount, total_discount, final_amount, order_date, status, payment_method, note)
- employee(**<u>employee_id</u>**, first_name, last_name, dob, phone_number, email, username, password)
- status_history(***<u>employee_id</u>***, ***<u>order_id</u>***, **<u>date_changed</u>**, status) (employee_id, order_id là **FK**)

3. Sơ đồ



E. CREATE TABLE

```
CREATE TABLE customer (
    customer_id CHAR(8) PRIMARY KEY,
    first_name VARCHAR(30) NOT NULL,
    last_name VARCHAR(30),
    street VARCHAR(50),
    city VARCHAR(30),
    district VARCHAR(30),
    phone_number VARCHAR(10) UNIQUE NOT NULL,
    email VARCHAR(40) UNIQUE,
    credit_card VARCHAR(30),
    dob DATE,
    gender CHAR(1) CHECK (gender IN ('F', 'M')),
    username VARCHAR(30) UNIQUE NOT NULL,
    password VARCHAR(20) NOT NULL
);
```

```
CREATE TABLE employee (
    employee_id CHAR(8) PRIMARY KEY,
    first_name VARCHAR(20),
    last_name VARCHAR(20),
    dob DATE,
    phone_number CHAR(10),
    email VARCHAR(40),
    username VARCHAR(30) UNIQUE NOT NULL,
    password VARCHAR(20) NOT NULL
);
```

```
CREATE TABLE category (
    category_id CHAR(4) PRIMARY KEY,
```

```
    category_name VARCHAR(30) NOT NULL
);
```

```
CREATE TABLE brand (
    brand_id CHAR(4) PRIMARY KEY,
    brand_name VARCHAR(100) NOT NULL,
    country_of_origin VARCHAR(40),
    brand_description TEXT
);
```

```
CREATE TABLE product (
    product_id CHAR(8) PRIMARY KEY,
    brand_id CHAR(4),
    product_name VARCHAR(200) NOT NULL,
    purchase_price DECIMAL(10,2) NOT NULL,
    selling_price DECIMAL(10,2) NOT NULL,
    material VARCHAR(50),
    product_description TEXT,
    average_rating DECIMAL(2,1) DEFAULT 0.0,
    CONSTRAINT fk_brand_id FOREIGN KEY (brand_id) REFERENCES brand(brand_id)
);
```

```
CREATE TABLE variant (
    variant_id CHAR(8) PRIMARY KEY,
    product_id CHAR(8) NOT NULL,
    color VARCHAR(50) NOT NULL,
    size DECIMAL(3,1) NOT NULL,
    stock_quantity INTEGER NOT NULL DEFAULT 0 CHECK (stock_quantity >= 0),
    CONSTRAINT fk_product_id FOREIGN KEY (product_id) REFERENCES product(product_id)
);
```

```
CREATE TABLE product_category (
    product_id CHAR(8) NOT NULL,
    category_id CHAR(4) NOT NULL,
    CONSTRAINT pk_product_category PRIMARY KEY (product_id, category_id),
    CONSTRAINT fk_product_id FOREIGN KEY (product_id) REFERENCES product(product_id),
    CONSTRAINT fk_category_id FOREIGN KEY (category_id) REFERENCES
category(category_id)
);
```

```
CREATE TABLE "order" (
    order_id CHAR(8) PRIMARY KEY,
    customer_id CHAR(8),
    total_amount DECIMAL(10,2) DEFAULT 0.0,
    total_discount DECIMAL(10,2) DEFAULT 0.0,
    final_amount DECIMAL(10,2) DEFAULT 0.0,
    order_date DATE DEFAULT CURRENT_DATE,
    status VARCHAR(20) CHECK
        (status IN ('PENDING', 'PACKAGING', 'ON DELIVERY', 'DELIVERED', 'CANCELLED'))
DEFAULT 'PENDING',
    last_updated_by_employee_id CHAR(8),
    payment_method VARCHAR(20),
    note TEXT,
    CONSTRAINT fk_customer_id FOREIGN KEY (customer_id) REFERENCES
customer(customer_id),
    CONSTRAINT fk_employee_id FOREIGN KEY (last_updated_by_employee_id) REFERENCES
employee(employee_id)
);
```

```
CREATE TABLE orderdetail (
    orderdetail_id CHAR(8) PRIMARY KEY,
    order_id CHAR(8) NOT NULL,
```

```
    variant_id CHAR(8) NOT NULL,
    order_quantity INTEGER NOT NULL,
    unit_price DECIMAL(10,2) DEFAULT 0.0,
    discount INTEGER DEFAULT 0,
    sub_total DECIMAL(10,2) DEFAULT 0.0,
    CONSTRAINT fk_order_id FOREIGN KEY (order_id) REFERENCES "order"(order_id),
    CONSTRAINT fk_variant_id FOREIGN KEY (variant_id) REFERENCES variant(variant_id)
);
```

```
CREATE TABLE feedback (
    orderdetail_id CHAR(8) PRIMARY KEY,
    feedback TEXT,
    rating DECIMAL(1,0),
    feedback_date DATE,
    CONSTRAINT fk_orderdetail_id FOREIGN KEY (orderdetail_id) REFERENCES
orderdetail(orderdetail_id)
);
```

```
CREATE TABLE status_history (
    employee_id CHAR(8) NOT NULL,
    order_id CHAR(8) NOT NULL,
    date_changed DATE NOT NULL DEFAULT CURRENT_DATE,
    status VARCHAR(20) CHECK
        (status IN ('PENDING', 'PACKAGING', 'ON DELIVERY', 'DELIVERED', 'CANCELLED'))
DEFAULT 'PENDING',
    CONSTRAINT pk_status_history PRIMARY KEY (employee_id, order_id, date_changed),
    CONSTRAINT fk_employee_id FOREIGN KEY (employee_id) REFERENCES
employee(employee_id),
    CONSTRAINT fk_order_id FOREIGN KEY (order_id) REFERENCES "order"(order_id)
);
```

F. VIEW

```
-- Tạo view chỉ hiện thị các các sản phẩm còn hàng
CREATE OR REPLACE VIEW available_products AS
SELECT p.*
FROM product p
WHERE
  EXISTS (
    SELECT 1
    FROM variant v
    WHERE v.product_id = p.product_id AND v.stock_quantity > 0
  );
```

G. INDEX

```
CREATE INDEX idx_order_status_date ON "order"(status, order_date ASC);
CREATE INDEX idx_order_customer_date ON "order"(customer_id, order_date DESC);
CREATE INDEX idx_variant_product_stock ON variant(product_id, stock_quantity);
CREATE INDEX idx_variant_size ON variant(size);
```

H. FUNCTION

```
-- Hàm check stock của một variant còn bao nhiêu
CREATE OR REPLACE FUNCTION fn_check_stock(p_variant_id CHAR(8))
RETURNS INTEGER AS $$
DECLARE
   v_stock_quantity INTEGER;
BEGIN
   SELECT
      stock_quantity
   INTO
      v_stock_quantity
   FROM
      variant
   WHERE
      variant_id = p_variant_id;
   RETURN COALESCE(v_stock_quantity, 0);
END;
$$ LANGUAGE plpgsql;
```

```
-- Hàm tìm kiếm sản phẩm dựa trên một chuỗi đầu vào
CREATE OR REPLACE FUNCTION fn_search_products(p_search_term TEXT)
RETURNS TABLE(product_id CHAR(8)) AS $$
BEGIN
   RETURN QUERY
   SELECT DISTINCT
      p.product_id
   FROM
      product p
   JOIN
      brand b ON p.brand_id = b.brand_id
```

```
    LEFT JOIN
        variant v ON p.product_id = v.product_id
    WHERE
        p.product_name ILIKE '%' || p_search_term || '%' OR
        b.brand_name ILIKE '%' || p_search_term || '%' OR
        p.product_description ILIKE '%' || p_search_term || '%' OR
        v.color ILIKE '%' || p_search_term || '%';
END;
$$ LANGUAGE plpgsql;
```

```
-- Hàm tạo order, đóng gói toàn bộ vào một giao dịch
CREATE OR REPLACE FUNCTION fn_create_order(
    p_customer_id CHAR(8),
    p_payment_method VARCHAR(20),
    p_note TEXT,
    p_cart_items JSONB -- '[{"variant_id": "V0000001", "quantity": 2}, ...]'
)
RETURNS CHAR(8) AS $$
DECLARE
    v_new_order_id CHAR(8);
    v_item JSONB;
    v_current_stock INTEGER;
    v_product_price DECIMAL(10, 2);
    v_total_amount DECIMAL(10, 2) := 0;
    v_final_amount DECIMAL(10, 2) := 0;
BEGIN
    FOR v_item IN SELECT * FROM jsonb_array_elements(p_cart_items)
    LOOP
        v_current_stock := fn_check_stock((v_item->>'variant_id')::CHAR(8));

        IF v_current_stock < (v_item->>'quantity')::INTEGER THEN
```

```
        RAISE EXCEPTION 'Insufficient inventory with variant %. The remaining quantity is %',
v_item->>'variant_id', v_current_stock;
    END IF;
  END LOOP;

  INSERT INTO "order" (customer_id, payment_method, note, status, order_date)
  VALUES (p_customer_id, p_payment_method, p_note, 'PENDING', NOW())
  RETURNING order_id INTO v_new_order_id;

  FOR v_item IN SELECT * FROM jsonb_array_elements(p_cart_items)
  LOOP
    SELECT p.selling_price INTO v_product_price
    FROM variant v
    JOIN product p ON v.product_id = p.product_id
    WHERE v.variant_id = (v_item->>'variant_id')::CHAR(8);

    INSERT INTO orderdetail (order_id, variant_id, order_quantity, unit_price, discount, sub_total)
    VALUES (
      v_new_order_id,
      (v_item->>'variant_id')::CHAR(8),
      (v_item->>'quantity')::INTEGER,
      v_product_price,
      0,
      v_product_price * (v_item->>'quantity')::INTEGER
    );
    UPDATE variant
    SET stock_quantity = stock_quantity - (v_item->>'quantity')::INTEGER
    WHERE variant_id = (v_item->>'variant_id')::CHAR(8);
  END LOOP;

  SELECT
```

```
    SUM(unit_price * order_quantity),
    SUM(sub_total)
  INTO
    v_total_amount, v_final_amount
  FROM orderdetail
  WHERE order_id = v_new_order_id;
  UPDATE "order"
  SET
    total_amount = v_total_amount,
    final_amount = v_final_amount
  WHERE order_id = v_new_order_id;

  RETURN v_new_order_id;
END;
$$ LANGUAGE plpgsql;
```

```
-- Hàm tính tổng doanh thu của một thương hiệu
CREATE OR REPLACE FUNCTION brand_revenue(bid CHAR(4))
RETURNS NUMERIC AS $$
DECLARE
 revenue NUMERIC;
BEGIN
 SELECT SUM(od.sub_total) INTO revenue
 FROM orderdetail od
 JOIN variant v ON od.variant_id = v.variant_id
 JOIN product p ON v.product_id = p.product_id
 WHERE p.brand_id = bid;
 RETURN COALESCE(revenue, 0);
END;
$$ LANGUAGE plpgsql;
```

I. TRIGGER

```
-- Hàm cho trigger tự động lưu lịch sử thay đổi trạng thái đơn hàng
CREATE OR REPLACE FUNCTION fn_handle_after_update_order()
RETURNS TRIGGER AS $$
BEGIN
   IF NEW.status IS DISTINCT FROM OLD.status THEN
      INSERT INTO status_history (order_id, employee_id, date_changed, status)
      VALUES (NEW.order_id, NEW.last_update_by_employee_id, NOW(), NEW.status);
   END IF;

   RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_after_update_order
AFTER UPDATE ON "order"
FOR EACH ROW
EXECUTE FUNCTION fn_handle_after_update_order();
```

```
-- Hàm cho trigger khi có một đơn hàng bị cập nhật trạng thái thành CANCELLED
CREATE OR REPLACE FUNCTION fn_handle_order_cancellation()
RETURNS TRIGGER AS $$
BEGIN
   IF NEW.status = 'CANCELLED' AND OLD.status <> 'CANCELLED' THEN
      UPDATE variant v
      SET stock_quantity = v.stock_quantity + od.order_quantity
      FROM orderdetail od
      WHERE v.variant_id = od.variant_id AND od.order_id = NEW.order_id;
   END IF;
```

```
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_handle_order_cancellation
AFTER UPDATE ON "order"
FOR EACH ROW
EXECUTE FUNCTION fn_handle_order_cancellation();
```

```
-- Hàm cho trigger sau khi có feedback mới
CREATE OR REPLACE FUNCTION fn_handle_after_insert_feedback()
RETURNS TRIGGER AS $$
DECLARE
    v_product_id CHAR(8);
    v_new_avg_rating DECIMAL(2,1);
BEGIN
    SELECT
        v.product_id INTO v_product_id
    FROM
        orderdetail od
    JOIN
        variant v ON od.variant_id = v.variant_id
    WHERE
        od.orderdetail_id = NEW.orderdetail_id;

    SELECT
        AVG(f.rating) INTO v_new_avg_rating
    FROM
        feedback f
    JOIN
        orderdetail od ON f.orderdetail_id = od.orderdetail_id
```

```
    JOIN
        variant v ON od.variant_id = v.variant_id
    WHERE
        v.product_id = v_product_id;

    UPDATE product
    SET average_rating = v_new_avg_rating
    WHERE product_id = v_product_id;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_after_insert_feedback
AFTER INSERT ON feedback
FOR EACH ROW
EXECUTE FUNCTION fn_handle_after_insert_feedback();
```

```
--Trigger tự động tính toán và thiết lập giá trị của unit_price và sub_total của orderdetail

CREATE OR REPLACE FUNCTION calculate_orderdetail_prices_trigger_function()

RETURNS TRIGGER AS $$ DECLARE v_selling_price NUMERIC(10, 2);

BEGIN

SELECT p.selling_price INTO v_selling_price

FROM product AS p JOIN variant AS v ON p.product_id = v.product_id

WHERE v.variant_id = NEW.variant_id;

IF NOT FOUND THEN RAISE EXCEPTION 'Product variant with ID % not found.',
NEW.variant_id;
END IF;
NEW.unit_price = v_selling_price;
NEW.sub_total = NEW.unit_price * NEW.order_quantity * (1 - NEW.discount / 100.00);
RETURN NEW;
```

```
END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_set_orderdetail_prices

BEFORE INSERT ON orderdetail FOR EACH ROW EXECUTE FUNCTION
calculate_orderdetail_prices_trigger_function();
```

```
-- Trigger tự động cập nhật bảng order khi orderline được cập nhật

CREATE OR REPLACE FUNCTION update_order_totals_function()

RETURNS TRIGGER AS $$

DECLARE

        v_order_id CHAR(8);

        v_total_amount NUMERIC(10, 2);

        v_total_discount_calculated NUMERIC(10, 2);

        v_final_amount_calculated NUMERIC(10, 2);

BEGIN

        IF TG_OP = 'DELETE' THEN v_order_id = OLD.order_id;

        ELSE v_order_id = NEW.order_id; END IF;

SELECT
   SUM(od.order_quantity * od.unit_price),
   SUM(od.order_quantity * od.unit_price * od.discount / 100.00),
   SUM(od.sub_total)
INTO
   v_total_amount,
   v_total_discount_calculated,
   v_final_amount_calculated
FROM
   orderdetail od
WHERE
   od.order_id = v_order_id;

UPDATE "order"
SET
   total_amount = v_total_amount,
   total_discount = v_total_discount_calculated,
```

```
    final_amount = v_final_amount_calculated
WHERE
    order_id = v_order_id;

IF TG_OP = 'DELETE' THEN
    RETURN OLD;
ELSE
    RETURN NEW;
END IF;


END;

$$ LANGUAGE plpgsql;



CREATE TRIGGER trg_after_orderdetail_change

AFTER INSERT OR UPDATE OR DELETE ON orderdetail

FOR EACH ROW

EXECUTE FUNCTION update_order_totals_function();
```

```
--Trigger quản lý số lượng tồn kho của sản phẩm trong variant

CREATE OR REPLACE FUNCTION update_stock_quantity_function()

RETURNS TRIGGER AS $$

DECLARE v_stock_quantity INT; v_quantity_change INT;

BEGIN

SELECT stock_quantity INTO v_stock_quantity

FROM variant

WHERE variant_id = COALESCE(NEW.variant_id, OLD.variant_id);

IF TG_OP = 'INSERT' THEN
    IF v_stock_quantity < NEW.order_quantity THEN
        RAISE EXCEPTION 'Không đủ số lượng tồn kho cho sản phẩm có variant_id %. Chỉ còn % sản
phẩm.', NEW.variant_id, v_stock_quantity;
    END IF;
    UPDATE variant
    SET stock_quantity = stock_quantity - NEW.order_quantity
```

```
    WHERE variant_id = NEW.variant_id;

ELSIF TG_OP = 'DELETE' THEN
    UPDATE variant
    SET stock_quantity = stock_quantity + OLD.order_quantity
    WHERE variant_id = OLD.variant_id;

ELSIF TG_OP = 'UPDATE' THEN
    v_quantity_change = NEW.order_quantity - OLD.order_quantity;

IF v_stock_quantity < v_quantity_change THEN
        RAISE EXCEPTION 'Không đủ số lượng tồn kho để tăng số lượng cho sản phẩm có variant_id
%. Chỉ còn % sản phẩm.', NEW.variant_id, v_stock_quantity;
    END IF;
    UPDATE variant
    SET stock_quantity = stock_quantity - v_quantity_change
    WHERE variant_id = NEW.variant_id;
END IF;
IF TG_OP = 'DELETE' THEN
    RETURN OLD;
ELSE
    RETURN NEW;
END IF;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_before_orderdetail_change_update_stock

BEFORE INSERT OR UPDATE OR DELETE ON orderdetail

FOR EACH ROW

EXECUTE FUNCTION update_stock_quantity_function();
```

J. QUERY

1. Truy vấn nghiệp vụ quản lí sản phẩm và kho hàng

```
-- Nguyễn Trung Kiên

-- 1. Liệt kê tất cả sản phẩm kèm tên thương hiệu và giá bán

SELECT p.product_id, p.product_name, b.brand_name, p.selling_price

FROM product p

JOIN brand b ON p.brand_id = b.brand_id

LIMIT 10;
```

```sql
-- 2. Liệt kê 10 sản phẩm có giá bán cao nhất (product_id, product_name, selling_price)
SELECT product_id, product_name, selling_price
FROM product
ORDER BY selling_price DESC
LIMIT 10;
```

```sql
-- 3. Thống kê số sản phẩm theo từng thương hiệu
SELECT b.brand_id, b.brand_name, COUNT(p.product_id) AS product_count
FROM brand b
LEFT JOIN product p ON b.brand_id = p.brand_id
GROUP BY b.brand_id, b.brand_name
ORDER BY product_count DESC;
```

```sql
-- 4. Liệt kê các sản phẩm chưa từng được bán (không xuất hiện trong orderdetail)
SELECT p.product_id, p.product_name
FROM product p
LEFT JOIN variant v ON p.product_id = v.product_id
LEFT JOIN orderdetail od ON v.variant_id = od.variant_id
WHERE od.orderdetail_id IS NULL
LIMIT 10;
```

```sql
-- 5. Thống kê số lượng sản phẩm theo từng phân loại (category)
SELECT c.category_id, c.category_name, COUNT(pc.product_id) AS product_count
FROM category c
LEFT JOIN product_category pc ON c.category_id = pc.category_id
GROUP BY c.category_id, c.category_name
ORDER BY product_count DESC;
```

```sql
-- 6. Thống kê số lượng sản phẩm theo từng nước xuất xứ của thương hiệu
```

```
SELECT b.country_of_origin, COUNT(p.product_id) AS product_count
FROM brand b
JOIN product p ON b.brand_id = p.brand_id
GROUP BY b.country_of_origin
ORDER BY product_count DESC;
```

```
-- 7. Liệt kê 3 thương hiệu có doanh thu cao nhất và số sản phẩm đã bán của mỗi thương hiệu
WITH BrandSales AS (
   SELECT
      b.brand_id,
      b.brand_name,
      SUM(od.sub_total) AS total_revenue,
      SUM(od.order_quantity) AS total_sold
   FROM brand b
   JOIN product p ON b.brand_id = p.brand_id
   JOIN variant v ON p.product_id = v.product_id
   JOIN orderdetail od ON v.variant_id = od.variant_id
   GROUP BY b.brand_id, b.brand_name
)
SELECT *
FROM BrandSales
ORDER BY total_revenue DESC
LIMIT 3;
```

```
-- 8. Liệt kê các thương hiệu có tổng số lượng tồn kho của tất cả sản phẩm lớn hơn 100
SELECT b.brand_id, b.brand_name, SUM(v.stock_quantity) AS total_stock
FROM brand b
JOIN product p ON b.brand_id = p.brand_id
JOIN variant v ON p.product_id = v.product_id
GROUP BY b.brand_id, b.brand_name
```

```
HAVING SUM(v.stock_quantity) > 100;
```

```
-- 9. Sử dụng function: Lấy trung bình đánh giá của một sản phẩm
CREATE OR REPLACE FUNCTION get_average_rating(pid CHAR(8))
RETURNS NUMERIC(2,1) AS $$
DECLARE
   avg_rating NUMERIC(2,1);
BEGIN
   SELECT average_rating INTO avg_rating FROM product WHERE product_id = pid;
   RETURN avg_rating;
END;
$$ LANGUAGE plpgsql;


SELECT product_id, product_name, get_average_rating(product_id) AS avg_rating
FROM product
ORDER BY avg_rating DESC
LIMIT 10;
```

```
-- 10. FUNCTION: Trả về tổng doanh thu của một thương hiệu theo brand_id
SELECT brand_id, brand_name, brand_revenue(brand_id) AS total_revenue
FROM brand
ORDER BY total_revenue DESC
LIMIT 5;
```

2. Truy vấn nghiệp vụ đặt hàng và quản lí đơn hàng

```
-- Nguyễn Tài Hưng
-- 1. Tìm kiếm sản phẩm theo khoảng giá
SELECT *
FROM available_products
WHERE selling_price
BETWEEN 40 AND 50
```

```
ORDER BY selling_price ASC
LIMIT 10;
```

```
-- 2. Tìm kiếm sản phẩm theo size
SELECT DISTINCT p.*
FROM available_products p
JOIN variant v
USING (product_id)
WHERE v.size = 44
ORDER BY p.selling_price ASC
LIMIT 10;
```

```
-- 3. Lấy tất cả đánh giá của một sản phẩm
SELECT v.product_id, v.variant_id, v.color, f.rating, f.feedback, f.feedback_date,
CONCAT(c.first_name, ' ', c.last_name) customer_name
FROM feedback f
JOIN orderdetail od
ON f.orderdetail_id = od.orderdetail_id
JOIN variant v
ON od.variant_id = v.variant_id
JOIN "order" o
ON od.order_id = o.order_id
JOIN customer c
ON o.customer_id = c.customer_id
WHERE v.product_id = 'PD000020';
```

```
-- 4. Tìm kiếm các sản phẩm đang được bán theo một chuỗi
SELECT * FROM fn_search_products('sport');
```

```
-- 5. Tạo một đơn hàng mới với một vài biến thể
SELECT fn_create_order(
```

```
    'CUS00001',
    'Cash',
    'hehehe!',
    '[{"variant_id": "V0000020", "quantity": 5}, {"variant_id": "V0000040", "quantity": 3}]'::JSONB
);
```

```
-- 6. Xem chi tiết một đơn hàng
SELECT o.*, p.product_name, v.color, v.size, od.order_quantity, od.unit_price
FROM "order" o
JOIN orderdetail od ON o.order_id = od.order_id
JOIN variant v ON od.variant_id = v.variant_id
JOIN product p ON v.product_id = p.product_id
WHERE o.order_id = 'ORD00308';
```

```
-- 7. Lấy lịch sử đơn hàng của một khách hàng cụ thể
SELECT order_id, order_date, final_amount, status
FROM "order"
WHERE customer_id = 'CUS00001' -- Thay bằng ID của khách hàng đang đăng nhập
ORDER BY order_date DESC;
```

```
-- 8. Lấy danh sách các đơn hàng ở trạng thái "PENDING" để nhân viên vào xử lý
SELECT order_id, customer_id, final_amount, order_date, payment_method
FROM "order"
WHERE status IN ('pending', 'processing')
ORDER BY order_date ASC; -- Ưu tiên xử lý đơn cũ trước
```

```
--9. Tìm kiếm đơn hàng theo thông tin khách hàng (SĐT hoặc Tên)
SELECT o.order_id, o.order_date, o.status, o.final_amount, c.first_name, c.last_name,
c.phone_number
FROM "order" o
JOIN customer c ON o.customer_id = c.customer_id
```

```
WHERE c.phone_number = '0987654321' OR c.first_name ILIKE '%Hung%';
```

```
--10. Lấy danh sách sản phẩm cần đóng gói cho các đơn đang xử lý

SELECT p.product_id, p.product_name, v.color, v.size, SUM(od.order_quantity) AS
total_quantity_to_pack

FROM orderdetail od

JOIN "order" o ON od.order_id = o.order_id

JOIN variant v ON od.variant_id = v.variant_id

JOIN product p ON v.product_id = p.product_id

WHERE o.status = 'PENDING'

GROUP BY p.product_id, p.product_name, v.color, v.size

ORDER BY p.product_name;
```

```
--11. Xem lịch sử thay đổi trạng thái của một đơn hàng

SELECT s.date_changed, s.status, e.first_name || ' ' || e.last_name AS employee_name

FROM status_history s

LEFT JOIN employee e ON s.employee_id = e.employee_id

WHERE s.order_id = 'OD000001'

ORDER BY s.date_changed ASC;
```

```
--12. Liệt kê các đơn hàng bị treo quá lâu

SELECT order_id, customer_id, order_date, status

FROM "order"

WHERE status = 'PENDING' AND order_date < NOW() - INTERVAL '3 days';
```

```
--13. Top 10 khách hàng chi tiêu nhiều nhất

SELECT c.customer_id, c.first_name, c.last_name, c.email, COUNT(o.order_id) AS total_orders,
SUM(o.final_amount) AS total_spent

FROM customer c

JOIN "order" o ON c.customer_id = o.customer_id

WHERE o.status = 'completed'
```

```
GROUP BY c.customer_id, c.first_name, c.last_name, c.email
ORDER BY total_spent DESC
LIMIT 10;
```

```
--14. Tính tổng giá trị của hàng tồn kho hiện tại
SELECT SUM(v.stock_quantity * p.purchase_price) AS total_inventory_value
FROM variant v
JOIN product p ON v.product_id = p.product_id
WHERE v.stock_quantity > 0;
```

3. Truy vấn về báo cáo và thống kê

```
--Trần Việt Gia Khánh
-- 1. Báo cáo doanh thu hàng ngày/tháng/năm
-- Doanh thu theo ngày
SELECT
    order_date,
    SUM(final_amount) AS daily_revenue
FROM "order"
WHERE order_date = CURRENT_DATE -- Hoặc thay bằng 'YYYY-MM-DD' cụ thể
GROUP BY order_date
ORDER BY order_date;


-- Doanh thu theo tháng
SELECT
    TO_CHAR(order_date, 'YYYY-MM') AS month,
    SUM(final_amount) AS monthly_revenue
FROM "order"
WHERE EXTRACT(YEAR FROM order_date) = EXTRACT(YEAR FROM CURRENT_DATE)
GROUP BY month
ORDER BY month;
-- Doanh thu theo năm
```

```
SELECT
    EXTRACT(YEAR FROM order_date) AS year,
    SUM(final_amount) AS yearly_revenue
FROM "order"
GROUP BY year
ORDER BY year;
```

```
-- 2. Báo cáo sản phẩm bán chạy nhất (theo số lượng hoặc doanh thu)
-- Sản phẩm bán chạy nhất theo số lượng
SELECT p.product_id, p.product_name, SUM(od.order_quantity) AS total_quantity_sold
FROM orderdetail od
JOIN variant v ON od.variant_id = v.variant_id
JOIN product p ON v.product_id = p.product_id
GROUP BY p.product_id, p.product_name
ORDER BY total_quantity_sold DESC
LIMIT 10;
-- Sản phẩm bán chạy nhất theo doanh thu
SELECT p.product_id, p.product_name, SUM(od.sub_total) AS total_revenue
FROM orderdetail od
JOIN variant v ON od.variant_id = v.variant_id
JOIN product p ON v.product_id = p.product_id
GROUP BY p.product_id, p.product_name ORDER BY total_revenue DESC LIMIT 10;
```

```
-- 3. Báo cáo doanh thu theo thương hiệu
SELECT b.brand_id, b.brand_name, SUM(od.sub_total) AS brand_revenue
FROM orderdetail od
JOIN variant v ON od.variant_id = v.variant_id
JOIN product p ON v.product_id = p.product_id
JOIN brand b ON p.brand_id = b.brand_id
GROUP BY b.brand_id, b.brand_name
```

```
ORDER BY brand_revenue DESC;
```

```
-- 4. Báo cáo doanh thu theo danh mục sản phẩm
SELECT c.category_id, c.category_name, SUM(od.sub_total) AS category_revenue
FROM orderdetail od
JOIN variant v ON od.variant_id = v.variant_id
JOIN product p ON v.product_id = p.product_id
JOIN product_category pc ON p.product_id = pc.product_id
JOIN category c ON pc.category_id = c.category_id
GROUP BY c.category_id, c.category_name
ORDER BY category_revenue DESC;
```

```
-- 5. Lịch sử mua hàng của một khách hàng cụ thể
SELECT o.order_id, o.order_date, p.product_name, v.color, v.size, od.order_quantity, od.unit_price,
od.sub_total
FROM "order" o
JOIN orderdetail od ON o.order_id = od.order_id
JOIN variant v ON od.variant_id = v.variant_id
JOIN product p ON v.product_id = p.product_id
WHERE o.customer_id = 'CUST001'
ORDER BY o.order_date DESC, o.order_id DESC;
```

```
-- 6. Danh sách khách hàng có số lượng đơn hàng cao nhất
SELECT
    c.customer_id,
    c.first_name,
    c.last_name,
    COUNT(o.order_id) AS total_orders
FROM customer c
JOIN "order" o ON c.customer_id = o.customer_id
GROUP BY c.customer_id, c.first_name, c.last_name
```

```
ORDER BY total_orders DESC
LIMIT 10;
```

```
-- 7. Danh sách khách hàng có tổng giá trị mua hàng cao nhất
SELECT
    c.customer_id,
    c.first_name,
    c.last_name,
    SUM(o.final_amount) AS total_spending
FROM customer c
JOIN "order" o ON c.customer_id = o.customer_id
GROUP BY c.customer_id, c.first_name, c.last_name
ORDER BY total_spending DESC
LIMIT 10;
```

```
-- 8. Các đơn hàng bị hủy hoặc hoàn trả
SELECT
    o.order_id,
    o.order_date,
    o.final_amount,
    o.status,
    c.first_name,
    c.last_name
FROM "order" o
JOIN customer c ON o.customer_id = c.customer_id
WHERE o.status IN ('Cancelled', 'Returned', 'Refunded')
ORDER BY o.order_date DESC;
```