

**Đại học Bách Khoa Hà Nội**  
**Trường công nghệ thông tin và truyền thông**  
\_\_\_\_\_ \*



**BÁO CÁO PROJECT CUỐI KỲ**  
**THỰC HÀNH CƠ SỞ DỮ LIỆU**

**Hệ thống quản lý và bán quần áo trực tuyến**

**GVHD : TS. Nguyễn Thị Oanh**

**Mã lớp : 156777**

**Nhóm 8:**

|                    |          |                               |
|--------------------|----------|-------------------------------|
| <b>Đỗ Xuân Quý</b> | 20235818 | Quy.DX235818@sis.hust.edu.vn  |
| Nguyễn Minh Quân   | 20235816 | Quan.NM235816@sis.hust.edu.vn |
| Trần Đăng Sinh     | 20235821 | Sinh.TD235821@sis.hust.edu.vn |

# MỤC LỤC

## Hệ thống quản lý và bán quần áo trực tuyến

|   |           |
|---|-----------|
| <b>1.Tổng quan đề tài</b> .....                                     | <b>3</b>  |
| 1.1.    Mục đích.....   | 3         |
| 1.2.    Quy trình nghiệp vụ .....                                   | 3         |
| 1.2.1.    Quy trình đăng ký tài khoản .....                         | 3         |
| 1.2.2.    Quy trình đặt hàng.....                                   | 3         |
| 1.2.3.    Quy trình thanh toán .....                                | 4         |
| 1.2.4.    Quy trình nhận mã giảm giá .....                          | 5         |
| 1.2.5.    Quy trình xử lý đơn hàng .....                            | 6         |
| 1.2.6.    Quy trình giao hàng .....                                 | 6         |
| 1.2.7.    Quy trình phản hồi, đánh giá .....                        | 6         |
| 1.2.8.    Quy trình quản lý sản phẩm, mã giảm giá của quản lý ..... | 7         |
| 1.2.9.    Quy trình thống kê báo cáo .....                          | 8         |
| <b>2.Chức năng ứng dụng</b> .....                                   | <b>9</b>  |
| 2.1.    Khách hàng .....  | 9         |
| 2.2.    Nhân viên .....   | 11        |
| 2.3.    Quản lý .....   | 12        |
| <b>3.Sơ đồ thực thể liên kết ERD</b> .....                          | <b>14</b> |
| <b>4.Mô hình quan hệ dữ liệu</b> .....                              | <b>15</b> |
| 4.1.    Thông tin cụ thể các bảng.....                              | 16        |
| <b>5.Phân tích truy vấn</b> .....                                   | <b>18</b> |
| 5.1.    10 câu truy vấn của Đỗ Xuân Quý .....                       | 18        |
| 5.2.    10 câu truy vấn của Nguyễn Minh Quân .....                  | 34        |
| 5.3.    10 câu truy vấn của Trần Đăng Sinh.....                     | 50        |
| <b>6.Xây dựng Website</b> .....                                     | <b>65</b> |
| 6.1.    Mô tả tổng quan.....  | 65        |
| 6.2.    Công nghệ sử dụng.....                                      | 65        |
| 6.3.    Demo website .....  | 65        |
| <b>7.Khó khăn khi thực hiện và cách giải quyết</b> .....            | <b>75</b> |
| <b>8.Đánh giá kết quả đạt được</b> .....                            | <b>78</b> |
| 8.1.    Ưu điểm .....   | 78        |
| 8.2.    Hạn chế.....  | 78        |
| <b>9.Nhiệm vụ các thành viên</b> .....                              | <b>79</b> |

# 1. Tổng quan đề tài

## 1.1. Mục đích

Dự án tập trung vào quản lý và bán sản phẩm, nhằm kết nối chủ shop và người mua hàng, giúp người dùng dễ dàng tìm được sản phẩm, đặt hàng, thanh toán và theo dõi đơn hàng. Với chủ cửa hàng, hệ thống cung cấp các tác vụ như đăng tải các sản phẩm, theo dõi thông tin khách hàng, xử lý các đơn hàng và kiểm tra thanh toán.

Với giao diện thân thiện, hệ thống tự động hóa quy trình bán hàng, đáp ứng nhu cầu của các cửa hàng kinh doanh quần áo và khách hàng mua sắm trực tuyến.

## 1.2. Quy trình nghiệp vụ

### 1.2.1. Quy trình đăng ký tài khoản

#### a. Khách hàng

- Truy cập trang đăng nhập → Nhập thông tin tài khoản ( first name, last name, email, phone number ) → Xác thực qua email gửi về
- Một tài khoản chỉ liên kết **1 email** và **1 số điện thoại**
- Sau khi xác thực, khách hàng được cung cấp quyền truy cập để xem sản phẩm, đặt hàng và quản lý thông tin cá nhân

#### b. Hệ thống

- Tích hợp bên thứ 3 ( Google Gmail ) để xác thực tài khoản
- Thông báo kết quả đăng ký ( thành công/thất bại )
- Lưu trữ thông tin khách hàng vào cơ sở dữ liệu, đảm bảo tính bảo mật và duy nhất của email/số điện thoại trong hệ thống.

#### c. Quản lý

- Tạo/xóa/cập nhật tài khoản nhân viên
- Phân quyền chức năng (quản lý, nhân viên )
- Xem thông tin người dùng

### 1.2.2. Quy trình đặt hàng

#### a. Khách hàng

- Khách hàng xem danh sách các sản phẩm trên website, tìm kiếm sản phẩm theo tên, filter theo danh mục, kích thước, màu sắc, khoảng giá và số sao đánh giá
- Khi chọn sản phẩm, hiển thị chi tiết sản phẩm bao gồm thông tin sản phẩm, số lượng sản phẩm còn trong kho (Nếu out of stock thì sẽ không thêm vào

giỏ hàng ), số sao đánh giá trung bình kèm những comment đánh giá.

- Với mỗi sản phẩm, khi người dùng chọn các biến thể khác nhau, giá sản phẩm không thay đổi ( ví dụ : Sơ mi HUST màu đỏ size S có giá 100k, Sơ mi HUST màu xanh size M cũng có giá 100k )
- Khách hàng chọn sản phẩm kèm số lượng, kích thước, màu sắc để thêm vào giỏ
- Giỏ hàng
  - Tăng/giảm số lượng của sản phẩm, xóa nếu như không có nhu cầu mua nữa
  - Nếu người dùng chưa đăng ký tài khoản vẫn có thể thêm vào giỏ hàng ( cart tạm thời ), sau khi đăng nhập thì cart vẫn hiển thị những sản phẩm kia.
- Sau khi checkout, khách hàng nhập thêm thông tin liên quan như địa chỉ giao kèm số điện thoại nhận,...
- **Lưu ý :**
  - Khách hàng muốn hủy/thay đổi đơn hàng sau khi thanh toán, vui lòng liên hệ với fanpage cửa hàng trong vòng 30' kể từ khi đặt.

#### b. Nhân viên

- Tiếp nhận đơn hàng → Đóng gói → Chuyển trạng thái đơn hàng từ “pending” sang “processing”
- Xử lý yêu cầu hủy/thay đổi của khách hàng (nếu thông báo lại trong vòng 30')

#### c. Hệ thống

- Cập nhật trạng thái đơn hàng theo thời gian thực : “pending” → “processing” → “shipping” → “completed”
- Khi người dùng thêm sản phẩm vào giỏ hàng, tự động kiểm tra các thuộc tính đi kèm bắt buộc ( kích thước, màu sắc ) với sản phẩm xem đã chọn chưa, nếu chưa thì thông báo cho người dùng chọn

### 1.2.3. Quy trình thanh toán

#### a. Khách hàng

- Khách hàng được chọn giữa 2 hình thức thanh toán
  - Thông qua cổng thanh toán VNPay ( bên thứ 3 cung cấp )
  - Thanh toán khi nhận hàng COD.

### **- Quy định**

- Chỉ được áp dụng 1 mã giảm giá / đơn
- Chỉ thanh toán 1 lần /đơn, không cộng dồn, không nửa thanh toán, nửa tiền mặt.
- Khi chưa thanh toán, nếu muốn thay đổi thông tin đơn hàng, có thể quay lại phần checkout để sửa thông tin. Nếu đã thanh toán, vui lòng liên hệ shop để thông báo thay đổi trong vòng 30'.
- Nếu thanh toán qua cổng thanh toán Online, giới hạn thanh toán trong vòng 15', nếu quá thời gian mà chưa thanh toán, quay về trạng thái ban đầu.

### **b. Nhân viên**

- Quản lý phân công cho nhân viên chuẩn bị đơn hàng phụ trách đơn hàng vừa được thanh toán xong, phân công cụ thể nhân viên giao hàng xử lý đơn hàng đó trên hệ thống
- Nhân viên chuẩn bị hàng nhận thông tin đơn hàng đã thanh toán, đóng gói và bàn giao cho nhân viên giao hàng.
- Tiếp nhận phản hồi từ khách hàng về việc đơn hàng có vấn đề thông qua website hệ thống
- Xử lý và nhanh chóng phản hồi tới khách hàng về thông tin đơn hàng, cách giải quyết dựa theo quy định của cửa hàng(chỉ được thay đổi thông tin đơn hàng/hủy đơn trong vòng 30' kể từ lúc thanh toán thành công)

### **c. Hệ thống**

- Cập nhật liên tục trạng thái đơn hàng theo lệnh thay đổi hợp lệ của nhân viên phụ trách đơn hàng, nhân viên giao hàng, khách hàng:
  - Người chuyển trạng thái từ “pending” → “processing” : nhân viên chuẩn bị đơn hàng
  - Người chuyển trạng thái từ “processing” → “shipping” : nhân viên chuẩn bị đơn hàng
  - Người chuyển trạng thái từ “shipping” → “completed” : nhân viên giao hàng và khách hàng đồng thời xác nhận
- Nếu thanh toán thất bại, hệ thống thông báo và tự động hủy đơn hàng sau 15' nếu khách hàng không thực hiện lại

### **1.2.4. Quy trình nhận mã giảm giá**

#### **a. Khách hàng**

- Mã giảm giá được cấp khi khách hàng thỏa mãn các điều kiện của chương trình ( mua hàng lần đầu, đạt 5 đánh giá, .... )
- Khách hàng có thể xem chi tiết mã giảm giá ( % giảm, thời hạn sử dụng, điều kiện áp dụng ) trong dashboard cá nhân

- **Lưu ý :** Mỗi mã giảm giá chỉ được sử dụng trên 1 đơn hàng duy nhất

b. **Nhân viên**

- Hỗ trợ giải đáp thắc mắc của khách hàng về mã giảm giá qua hotline, fanpage

c. **Hệ thống**

- Tự động kiểm tra điều kiện áp dụng mã giảm giá ( thời hạn chương trình, số vé sử dụng, điều kiện khác )
- Gửi thông báo đến khách hàng khi nhận mã giảm giá trên website

#### 1.2.5. **Quy trình xử lý đơn hàng**

a. **Quản lý**

- Khi thấy order mới từ khách hàng, quản lý thông báo và bàn giao cho nhân viên chuẩn bị hàng xử lý

b. **Nhân viên chuẩn bị hàng**

- Khi nhận được thông tin order từ quản lý, nhân viên chuẩn bị đơn hàng cập nhật trạng thái đơn hàng từ “pending” → “processing”
- Sau đó nhân viên bắt đầu chuẩn bị và đóng gói đơn hàng.

#### 1.2.6. **Quy trình giao hàng**

a. **Khách hàng**

- Theo dõi thông tin và trạng thái của đơn hàng trong phần My Order trên dashboard cá nhân
- Khi nhận hàng, khách hàng kí xác nhận vào biên lai giao hàng

b. **Nhân viên chuẩn bị hàng**

- Chuẩn bị và bàn giao đơn hàng cho nhân viên giao hàng
- Cập nhật trạng thái đơn hàng từ “processing” → “shipping”
- Xử lý yêu cầu hủy/thay đổi đơn hàng từ khách hàng theo quy định

c. **Nhân viên giao hàng**

- Nhận đơn hàng và thông tin từ cửa hàng từ nhân viên chuẩn bị
- Gọi điện thông báo cho khách hàng 15' trước khi giao tới nơi
- Khi hoàn thành giao hàng, kí xác nhận dành cho người giao hàng, và đưa cho khách hàng kí xác nhận đơn hàng

#### 1.2.7. **Quy trình phản hồi, đánh giá**

#### a. **Khách hàng**

- Mỗi khách hàng đều có thể đánh giá sản phẩm bao gồm số sao, đi kèm comment đánh giá ( Không yêu cầu mua hàng rồi mới đánh giá )
- Đánh giá phải tuân thủ quy định: Phải rate số sao, không sử dụng từ viết tắt, những từ ngữ vi phạm chuẩn mực cộng đồng. Nếu vi phạm, nhân viên sẽ kiểm tra và xóa bỏ comment đó.
- Đánh giá theo thang 5 sao ( MIN = 0.5 sao )
- Khách hàng có thể sửa comment bất cứ khi nào, nhưng không được xóa comment của mình.

#### b. **Nhân viên**

- Xem xét và loại bỏ những đánh giá không hợp lệ
- Phản hồi khách hàng dựa trên nội dung đánh giá

#### c. **Hệ thống**

- Hiển thị những đánh giá của người dùng bên dưới thông tin của từng sản phẩm tương ứng
- Tự động cập nhật tính số sao trung bình cho sản phẩm đó

### 1.2.8. **Quy trình quản lý sản phẩm, mã giảm giá của quản lý**

#### a. **Thêm danh mục mới**

- Quản lý nhập thông tin như : Tên danh mục, mô tả ( nếu cần )

#### b. **Thêm sản phẩm mới**

- Quản lý nhập các thông tin của sản phẩm
  - Thông tin cơ bản : Tên sản phẩm, Mô tả, Ảnh, Giá, số lượng trong kho
  - Biển thể : Thêm màu sắc, kích thước cho riêng từng sản phẩm ( Ví dụ : Áo sơ mi HUST có các màu : đỏ, xanh, đen và các kích thước : XS, S, M, L )
  - Mỗi sản phẩm chỉ thuộc **1 danh mục sản phẩm**

#### c. **Cập nhật/Xóa thông tin danh mục, sản phẩm**

- Quản lý có thể chỉnh sửa thông tin của danh mục, xóa nếu không bán loại mặt hàng đó nữa ( Khi xóa danh mục thì các sản phẩm trong danh mục cũng bị xóa ).
- Quản lý chỉnh sửa thông tin sản phẩm bao gồm tên sản phẩm, mô tả, giá , hình ảnh, các biển thể ( màu sắc, kích thước) của sản phẩm và xóa nếu như không bán sản phẩm nữa.

#### d. **Quản lý mã giảm giá**

- Quản lý thiết lập chương trình giảm giá, bao gồm

- Phần trăm giảm giá
  - Thời gian hiệu lực của mã
  - Điều kiện áp dụng : Khách hàng lần đầu mua hàng, ....
  - Số lượng mã phát hành
- Hệ thống tự động kiểm tra và cung cấp cho khách hàng những mã giảm giá  
khả dụng khi thanh toán đơn hàng.

### 1.2.9. **Quy trình thống kê báo cáo**

#### a. **Thống kê doanh thu**

- Hệ thống tổng hợp doanh thu theo ngày, tuần, tháng, năm. Hiển thị biểu đồ  
doanh thu và biểu đồ tăng trưởng ( dạng cột, đường, ) giúp quản lý đánh giá  
hiệu quả kinh doanh
- Quản lý có thể xuất báo cáo dưới dạng file Excel hoặc Pdf để tiện theo dõi

#### b. **Thống kê sản phẩm bán chạy**

- Hệ thống liệt kê các sản phẩm bán chạy nhất dựa trên số lượng đơn hàng trong  
tuần/tháng/năm.
- Quản lý sử dụng dữ liệu để điều chỉnh chiến lược kinh doanh, ưu tiên quảng  
bá sản phẩm nổi bật.

## 2. Chức năng ứng dụng

Dưới đây là bảng tổng hợp các chức năng của hệ thống quản lý và bán quần áo trực tuyến, phân loại theo từng đối tượng sử dụng ( Khách hàng, Nhân viên, Quản lý ) cùng với mô tả, đầu vào, đầu ra và các ràng buộc liên quan tới.

### 2.1. Khách hàng

| STT | Chức năng                             | Mô tả   | Đầu vào   | Đầu ra                           | Ràng buộc  |
|-----|---------------------------------------|---|---|----------------------------------|--|
| 1   | Đăng ký tài khoản                     | Khách hàng đăng ký tài khoản để sử dụng website   | first_name, last_name, email, phone               | Tài khoản khách hàng             | - Email và SDT phải duy nhất trong hệ thống<br>- Phải thực hiện xác thực qua email             |
| 2   | Đăng nhập                             | Khách hàng đăng nhập vào hệ thống   | email, password                                   | Phiên đăng nhập ( session )      | Không  |
| 3   | Xem danh sách sản phẩm                | Hiển thị danh sách sản phẩm có sẵn ở menu   | Không   | Bảng danh sách các sản phẩm      | Không  |
| 4   | Tìm kiếm và lọc sản phẩm theo nhu cầu | Tìm kiếm, lọc sản phẩm theo tên, danh mục, khoảng giá, màu sắc, kích thước, số sao đánh giá | keyword, category, price_range, color, size, star | Bảng sản phẩm phù hợp với filter | Không  |
| 5   | Xem chi tiết sản phẩm                 | Hiển thị thông tin chi tiết của sản phẩm  | product_id  | Thông tin chi tiết về sản phẩm   | Không  |
| 6   | Thêm vào giỏ hàng                     | Khách hàng thêm sản phẩm vào giỏ hàng   | product_id, quantiy, color, size                  | Giỏ hàng cập nhật                | - Sản phẩm phải còn hàng ( quantity > 0 )<br>- Phải chọn đủ thuộc tính ( màu sắc, kích thước ) |
| 7   | Xem đánh giá sản phẩm                 | Xem các đánh giá và số sao trung bình của một sản phẩm                                      | product_id  | Danh sách đánh giá               | Không  |

|    |                       |  |  |                                       |  |
|----|-----------------------|--|--|---------------------------------------|--|
| 8  | Thanh toán đơn hàng   | Khách hàng tiến hành lựa chọn hình thức thanh toán và thanh toán đơn hàng  | user_id, cart_id, shipping_address, payment_method | Đơn hàng được tạo                     | <ul style="list-style-type: none"> <li>-Chỉ được sử dụng 1 mã/đơn</li> <li>-Thanh toán trong vòng 15' (nếu qua công thanh toán)</li> </ul> |
| 9  | Theo dõi đơn hàng     | Xem trạng thái hiện tại và chi tiết đơn hàng                               | user_id, order_id                                  | Danh sách đơn hàng và trạng thái      | Không  |
| 10 | Hủy/thay đổi đơn hàng | Hủy hoặc yêu cầu thay đổi đơn hàng theo quy định của shop                  | order_id   | Thông báo hủy/thay đổi đơn hàng       | Chỉ được thực hiện trong vòng 30' sau khi thanh toán   |
| 11 | Đánh giá sản phẩm     | Khách hàng đánh giá sản phẩm đã mua  | product_id, rating, comment                        | Đánh giá                              | <ul style="list-style-type: none"> <li>-Đánh giá từ 0.5 sao đến 5 sao</li> <li>-Chỉnh sửa trong vòng 24 giờ</li> </ul>                     |
| 12 | Xem mã giảm giá       | Xem các mã giảm giá khả dụng   | promotion_id                                       | Danh sách mã giảm giá đang có         | Không  |
| 13 | Sử dụng mã giảm giá   | Áp dụng mã giảm giá khi thanh toán   | discount_code                                      | Giá đơn hàng được giảm                | <ul style="list-style-type: none"> <li>-Mỗi mã dùng 1 lần/đơn</li> <li>-Phải thỏa mãn điều kiện áp dụng (mua hàng lần đầu,...)</li> </ul>  |
| 14 | Xem lịch sử mua hàng  | Xem các đơn hàng đã mua trong quá khứ                                      | user_id, order_id                                  | Thông tin các đơn hàng đã mua         | Không  |
| 15 | Quản lý giỏ hàng      | Xem các sản phẩm trong giỏ hàng, tăng/giảm số lượng sản phẩm, xóa sản phẩm | user_id, order_id                                  | Danh sách các sản phẩm trong giỏ hàng | Không  |

## 2.2. Nhân viên

| STT | Chức năng                     | Mô tả   | Đầu vào          | Đầu ra                            | Ràng buộc   |
|-----|-------------------------------|---|------------------|-----------------------------------|---|
| 1   | Xem danh sách đơn hàng        | Xem tất cả đơn hàng đang chờ xử lý                          | Không            | Danh sách đơn hàng                | Không   |
| 2   | Xem chi tiết đơn hàng         | Xem thông tin chi tiết của 1 đơn hàng                       | order_id         | Thông tin đơn hàng                | Không   |
| 3   | Xem yêu cầu hủy/thay đổi      | Xem các yêu cầu hủy/thay đổi đơn hàng từ khách hàng         | order_id         | Danh sách yêu cầu                 | Chỉ xem những yêu cầu trong 30' sau thanh toán                    |
| 4   | Xem đánh giá sản phẩm         | Xem các đánh giá sản phẩm để kiểm tra tính hợp lệ           | product_id       | Danh sách đánh giá                | Không   |
| 5   | Cập nhật trạng thái đơn hàng  | Cập nhật trạng thái đơn hàng (“pending” sang “processing” ) | order_id, status | Trạng thái đơn hàng được cập nhật | Không   |
| 6   | Xóa các đánh giá không hợp lệ | Xóa các đánh giá vi phạm quy định                           | review_id        | Đánh giá bị xóa                   | Chỉ xóa những đánh giá không hợp lệ, vi phạm tiêu chuẩn cộng đồng |

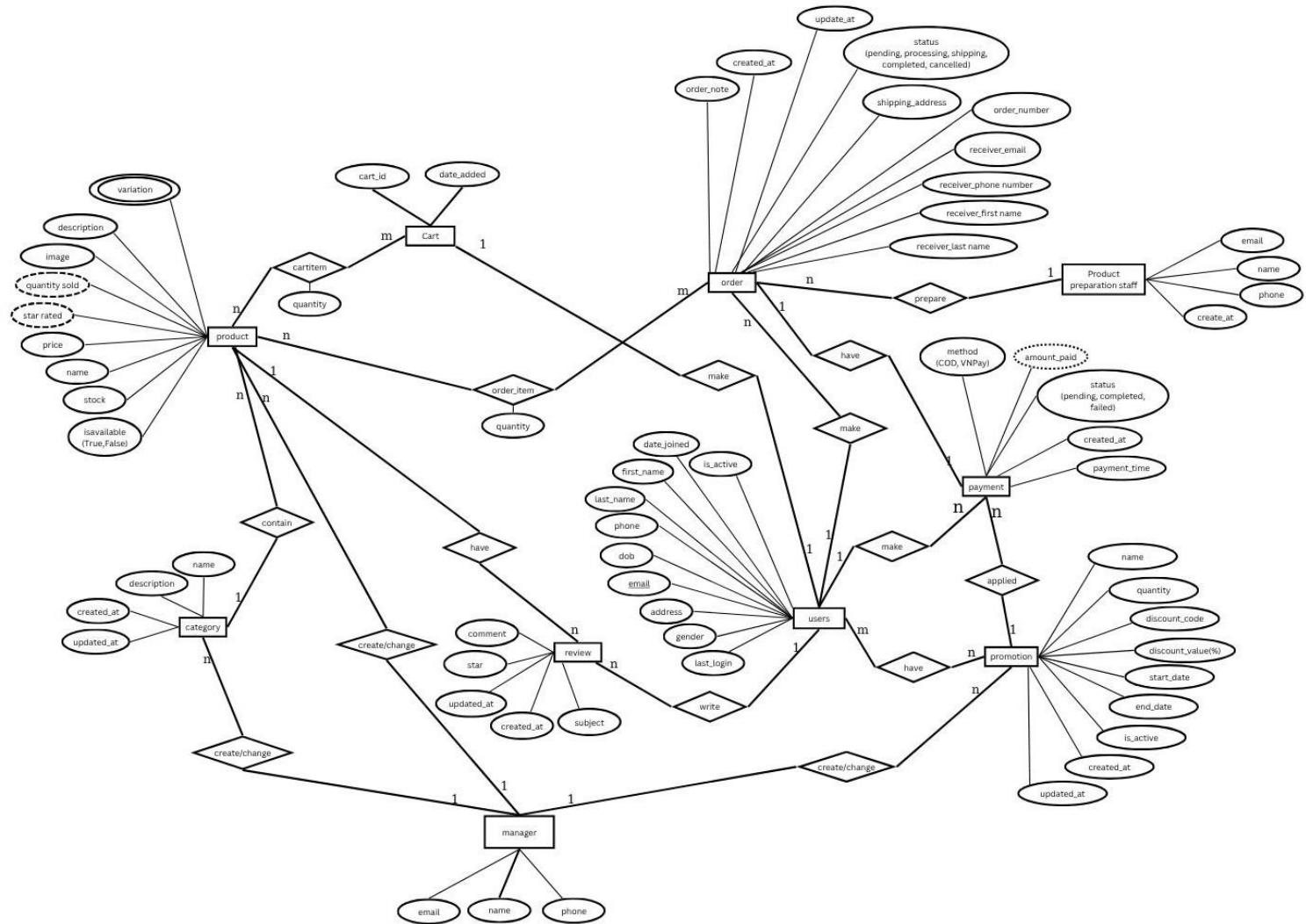
### 2.3. Quản lý

| STT | Chức năng                         | Mô tả  | Đầu vào  | Đầu ra                                       | Ràng buộc                     |
|-----|-----------------------------------|--|--|--|-------------------------------|
| 1   | Xem danh sách sản phẩm, biển thẻ  | Xem tất cả các sản phẩm kèm biển thẻ hiện có trên hệ thống | manager_id   | Danh sách sản phẩm                           | Không                         |
| 2   | Xem danh sách các danh mục        | Xem tất cả các danh mục hiện có trên hệ thống              | manager_id   | Danh sách danh mục                           | Không                         |
| 3   | Xem thông tin nhân viên           | Xem thông tin tài các tài khoản nhân viên hiện có          | staff_id   | Danh sách nhân viên                          | Không                         |
| 4   | Xem thông tin khách hàng          | Xem tất cả các thông tin tài khách hàng trong hệ thống     | user_id  | Danh sách khách hàng                         | Không                         |
| 5   | Xem sản phẩm bán chạy             | Xem danh sách các sản phẩm bán chạy theo tuần/tháng/năm    | time_period  | Danh sách sản phẩm bán chạy                  | Không                         |
| 6   | Xem thống kê doanh thu hàng tháng | Xem tổng doanh thu, lợi nhuận của tháng cụ thể             | manager_id, month  | Tổng doanh thu, lợi nhuận của 1 tháng cụ thể | Không                         |
| 7   | Xem danh sách mã giảm giá         | Xem tất cả mã giảm giá đang có trong hệ thống              | manager_id   | Danh sách mã giảm giá                        | Không                         |
| 8   | Thêm/xóa/sửa sản phẩm             | Quản lý thông tin sản phẩm                                 | product_id, product_name, price, stock_quantity, color, size | Sản phẩm được cập nhật                       | Sản phẩm chỉ thuộc 1 danh mục |

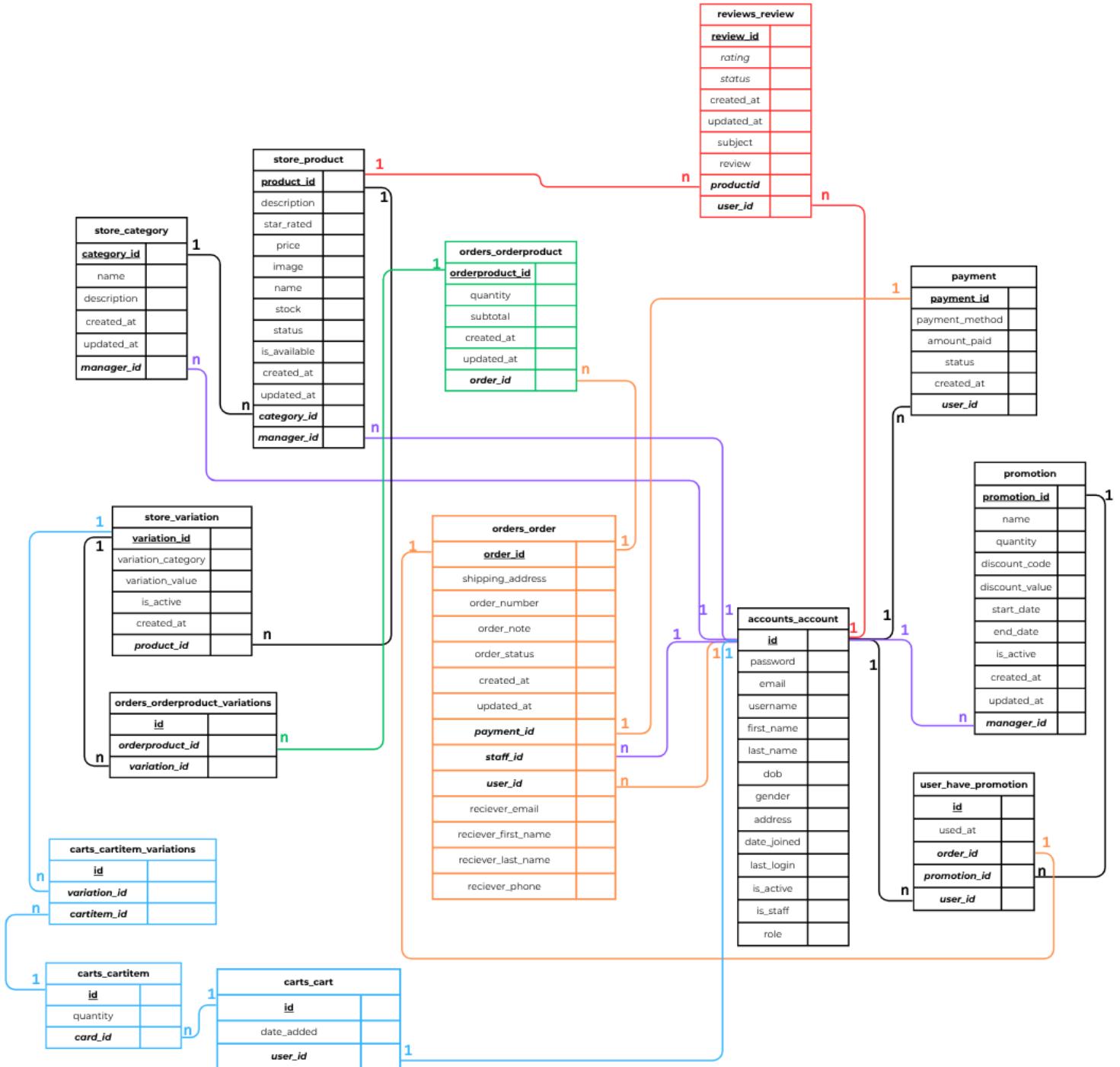
|    |  |  |   |                                   |  |
|----|--|--|---|-----------------------------------|--|
| 9  | Thêm/sửa/xóa danh mục                  | Quản lý danh mục sản phẩm                                    | category_id, category_name, description             | Danh mục sản phẩm được cập nhật   | Xóa danh mục sẽ xóa toàn bộ các sản phẩm trong nó                      |
| 10 | Thêm/xóa/sửa mã giảm giá               | Quản lý thiết lập chương trình giảm giá                      | discount_code, discount_value, start_date, end_date | Mã giảm giá được cập nhật         | Mã có thời hạn và điều kiện như lần đầu mua hàng, đạt 10 lượt đánh giá |
| 11 | Quản lý tài khoản nhân viên            | Tạo/xóa tài khoản nhân viên                                  | staff_id, email, role                               | Tài khoản nhân viên được cập nhật | Không  |
| 12 | Xuất báo cáo                           | Xuất báo cáo doanh thu hoặc sản phẩm bán chạy                | Không   | File Exel/Pdf/Ảnh                 | Không  |
| 13 | Xem tỉ lệ đơn hàng bị hủy              | Xem tỉ lệ các đơn hàng người dùng hủy                        | order_id  | Tỷ lệ bị hủy                      | Không  |
| 14 | Xem trạng thái hoàn thành của đơn hàng | Quản lý có thể xem đơn hàng đã được giao thành công hay chưa | order_id, status                                    | Bảng trạng thái đơn hàng          | Không  |

### 3. Sơ đồ thực thể liên kết ERD

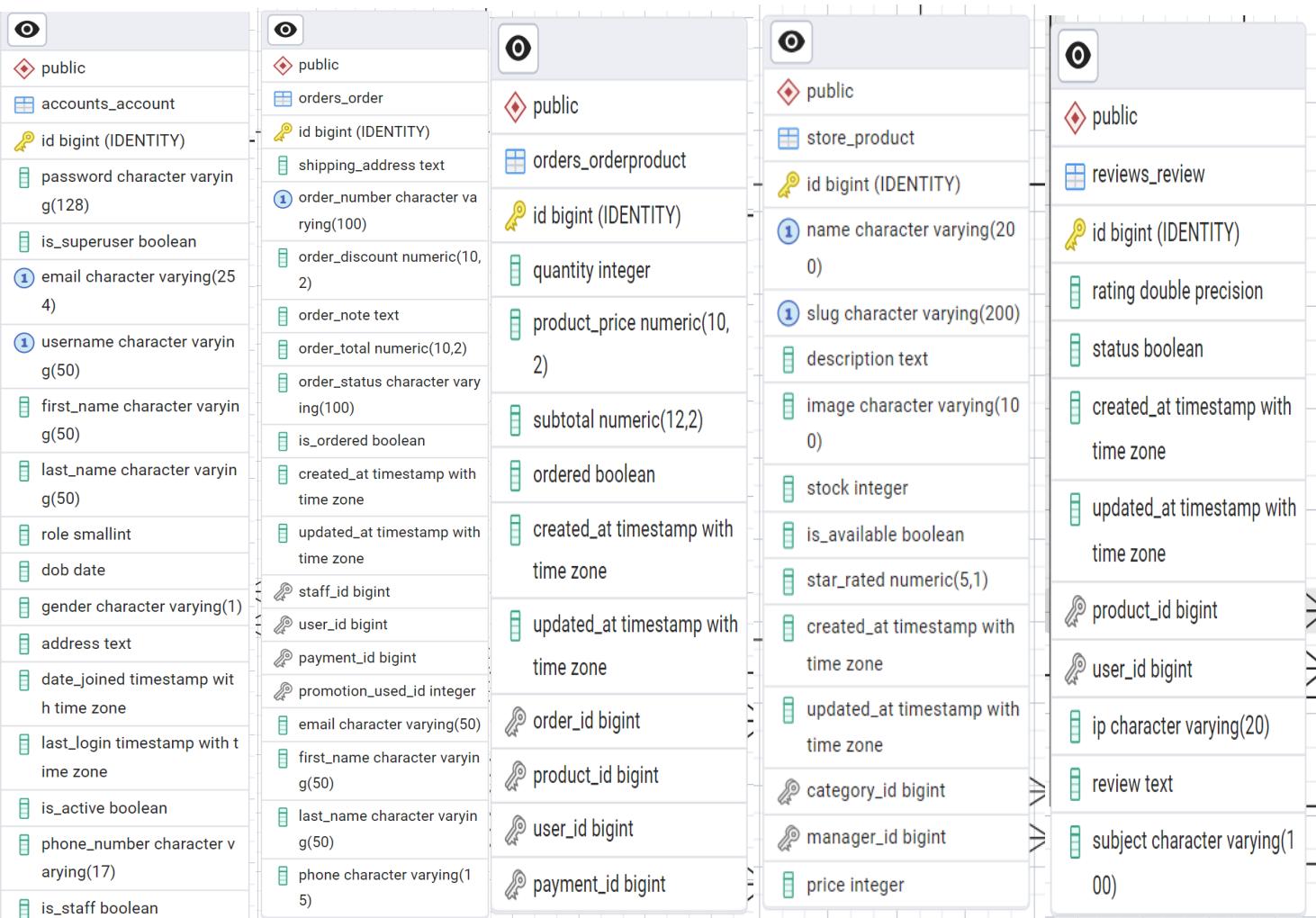
ERD có thể được xem [tại đây](#).

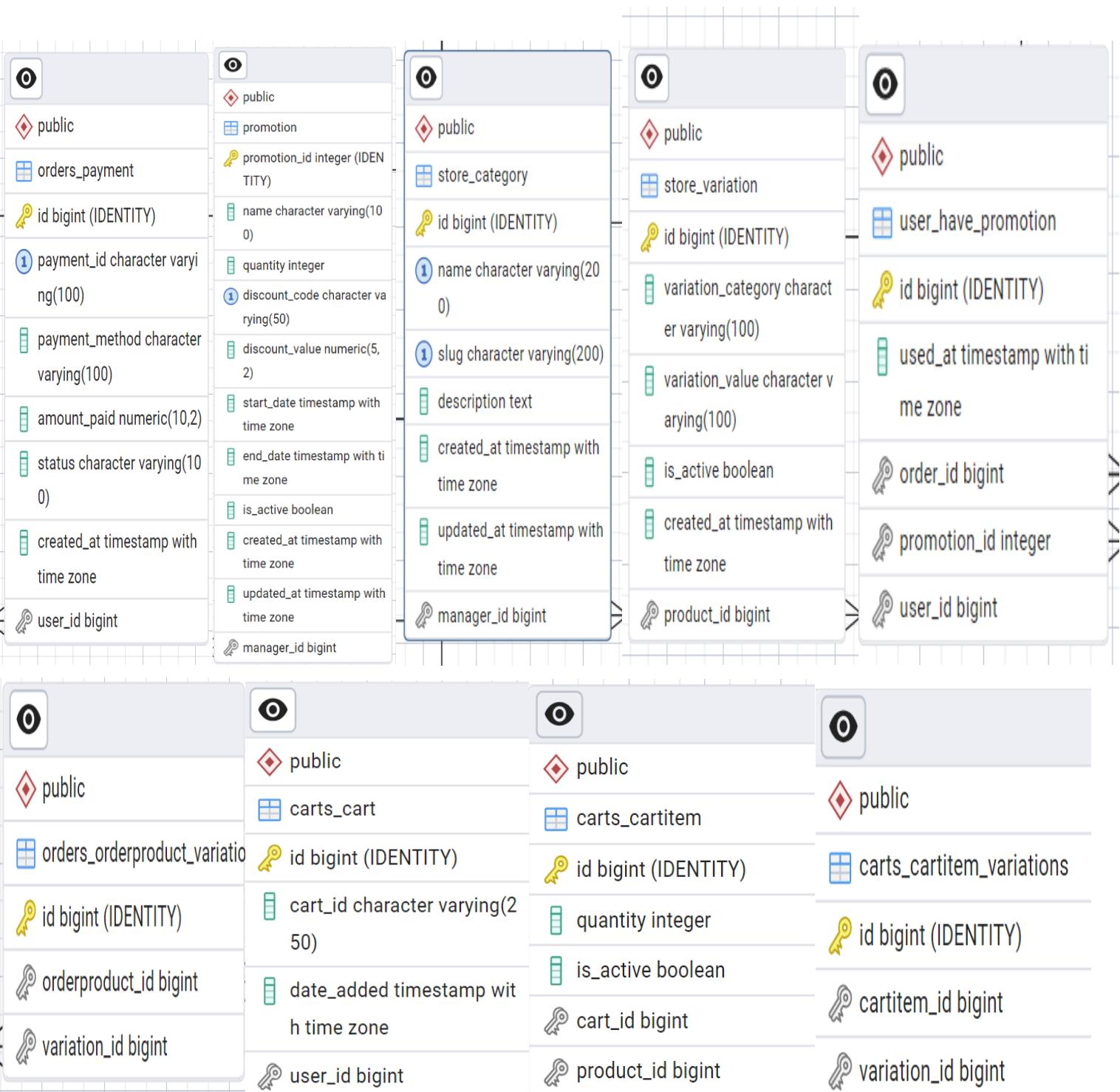


## 4. Mô hình quan hệ dữ liệu



## 4.1. Thông tin cụ thể các bảng





## 5. Phân tích truy vấn

Dữ liệu dùng trong các truy vấn bên dưới bạn em đã gửi đính kèm trong file sample\_data.sql

### 5.1. 10 câu truy vấn của Đỗ Xuân Quý

#### Câu 1: Function thống kê top sản phẩm bán chạy nhất

- Function trả về limit\_count sản phẩm tạo ra doanh thu cao nhất, được tính bằng SUM(op.quantity \* p.price) là tổng doanh thu, chỉ tính các đơn hàng đã hoàn tất (o.is\_ordered = true) và mặc định trả 5 sản phẩm doanh thu cao nhất nếu không truyền giá trị limit\_count.

```
CREATE OR REPLACE FUNCTION hustshop_top_products_by_revenue(limit_count INT DEFAULT 5)
RETURNS TABLE (
    product_id BIGINT,
    product_name TEXT,
    total_revenue BIGINT
) AS $$

BEGIN
    RETURN QUERY
    SELECT
        p.id,
        p.name::TEXT,
        SUM(op.quantity * p.price) AS total_revenue
    FROM orders_orderproduct op
    JOIN store_product p ON op.product_id = p.id
    JOIN orders_order o ON op.order_id = o.id
    WHERE o.is_ordered = true
    GROUP BY p.id, p.name
    ORDER BY total_revenue DESC
    LIMIT limit_count;
END;
$$ LANGUAGE plpgsql STABLE;

# ANALYZE:
EXPLAIN ANALYZE
SELECT
    p.id AS product_id,
    p.name::TEXT AS product_name,
    SUM(op.quantity * p.price) AS total_revenue
FROM orders_orderproduct op
JOIN store_product p ON op.product_id = p.id
JOIN orders_order o ON op.order_id = o.id
WHERE o.is_ordered = true
GROUP BY p.id, p.name
ORDER BY total_revenue DESC
LIMIT 30;
```

```

1 Limit (cost=1659.01..1659.09 rows=30 width=67) (actual time=10.216..10.221 rows=30 loops=1)
2   -> Sort (cost=1659.01..1680.80 rows=8714 width=67) (actual time=10.215..10.219 rows=30 loops=1)
3     Sort Key: (sum((op.quantity * p.price))) DESC
4     Sort Method: top-N heapsort Memory: 31kB
5     -> HashAggregate (cost=1314.51..1401.65 rows=8714 width=67) (actual time=9.263..9.801 rows=5661 loops=1)
6       Group Key: p.id
7       Batches: 1 Memory Usage: 1169kB
8       -> Hash Join (cost=992.25..1249.16 rows=8714 width=35) (actual time=5.038..7.918 rows=8714 loops=1)
9         Hash Cond: (op.order_id = o.id)
10        -> Hash Join (cost=492.33..726.36 rows=8714 width=43) (actual time=1.832..3.790 rows=8714 loops=1)
11          Hash Cond: (op.product_id = p.id)
12          -> Seq Scan on orders_orderproduct op (cost=0.00..211.14 rows=8714 width=20) (actual time=0.019..0.373 ro...
13          -> Hash (cost=374.37..374.37 rows=9437 width=31) (actual time=1.761..1.762 rows=9437 loops=1)
14            Buckets: 16384 Batches: 1 Memory Usage: 729kB
15            -> Seq Scan on store_product p (cost=0.00..374.37 rows=9437 width=31) (actual time=0.007..1.023 rows=9...
16            -> Hash (cost=383.85..383.85 rows=9285 width=8) (actual time=3.166..3.167 rows=9285 loops=1)
17            Buckets: 16384 Batches: 1 Memory Usage: 491kB
18            -> Seq Scan on orders_order o (cost=0.00..383.85 rows=9285 width=8) (actual time=0.012..2.135 rows=9285 l...
19            Filter: is_ordered
20 Planning Time: 0.359 ms
21 Execution Time: 10.540 ms

```

## # TEST

```
SELECT * FROM hustshop_top_products_by_revenue(3);
```

|   | product_id  | product_name  | total_revenue  |
|---|--|--|---|
| 1 | 3166   | Special Item 615   | 10274   |
| 2 | 4656   | Special Item 2105  | 8820  |
| 3 | 7486   | Special Item 4935  | 7999  |

## # Nhận xét:

- Trong câu lệnh có 3 lần Seq Scan tại các bảng: orders\_orderproduct, store\_product, orders\_order.
- Tuy nhiên cost của 3 lần này là rất nhỏ so với chi phí tính toán tổng thể của cả câu truy vấn và tần suất sử dụng hàm không lớn do đó không nên tạo thêm index để tối ưu thời gian chạy cho câu lệnh.

## Câu 2: Function thống kê đơn hàng của một khách hàng cụ thể

- Function trả về số đơn hàng, giá trị trung bình các đơn hàng và lần mua hàng gần nhất của một khách hàng với đầu vào là id của khách hàng đó.

```
CREATE OR REPLACE FUNCTION get_customer_stats(customer_id BIGINT)
RETURNS TABLE(
    total_orders INTEGER,
    total_spent NUMERIC(12,2),
    avg_order_value NUMERIC(10,2),
    last_order_date TIMESTAMP WITH TIME ZONE
) AS $$

BEGIN
    RETURN QUERY
    SELECT
        COUNT(*)::INTEGER as total_orders,
        COALESCE(SUM(order_total), 0) as total_spent,
        COALESCE(AVG(order_total), 0) as avg_order_value,
        MAX(created_at) as last_order_date
    FROM orders_order
    WHERE user_id = customer_id AND is_ordered = true;
END;
$$ LANGUAGE plpgsql;

# ANALYZE:
EXPLAIN ANALYZE
SELECT
    COUNT(*)::INTEGER as total_orders,
    COALESCE(SUM(order_total), 0) as total_spent,
    COALESCE(AVG(order_total), 0) as avg_order_value,
    MAX(created_at) as last_order_date
FROM orders_order
WHERE user_id = 1 AND is_ordered = true;
```

|   | QUERY PLAN   |
|---|--|
| 1 | Aggregate (cost=11.85..11.86 rows=1 width=76) (actual time=0.018..0.018 rows=1 loops=1)  |
| 2 | -> Bitmap Heap Scan on orders_order (cost=4.30..11.83 rows=2 width=14) (actual time=0.016..0.016 rows=0 loops=1)                 |
| 3 | Recheck Cond: (user_id = 1)  |
| 4 | Filter: is_ordered   |
| 5 | -> Bitmap Index Scan on orders_order_user_id_e9b59eb1 (cost=0.00..4.30 rows=2 width=0) (actual time=0.011..0.011 rows=0 loops=1) |
| 6 | Index Cond: (user_id = 1)  |
| 7 | Planning Time: 0.147 ms  |
| 8 | Execution Time: 0.037 ms   |

## # TEST:

```
SELECT * FROM get_customer_stats(159);
```

|   | total_orders | total_spent | avg_order_value      | last_order_date        |
|---|--------------|-------------|----------------------|------------------------|
| 1 | 2            | 616.74      | 308.3700000000000000 | 2024-05-26 14:29:18+07 |

## # Nhận xét:

- Không có Seq Scan trong Query Plan, tần suất sử dụng của hàm nhỏ nên không cần sử dụng

index để cải thiện hiệu suất.

### Câu 3: Trigger kiểm tra giá trị giảm giá có hợp lệ hay không của promotion

- Trigger sẽ đưa ra thông báo dữ liệu không hợp lệ nếu mục discount\_value vượt quá phạm vi (0,99.99] khi thay đổi hoặc thêm vào bảng promotion.

```
CREATE OR REPLACE FUNCTION validate_discount_value()
RETURNS TRIGGER AS $$ 
BEGIN
    IF NEW.discount_value <= 0 OR NEW.discount_value > 99.99 THEN
        RAISE EXCEPTION 'Giá trị giảm giá (%) phải nằm trong khoảng 0.01 đến 99.99 cho khuyến mãi ID: %.', 
                        NEW.discount_value, NEW.promotion_id;
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER trigger_validate_discount_value
BEFORE INSERT OR UPDATE ON public.promotion
FOR EACH ROW
EXECUTE FUNCTION validate_discount_value();

# TEST:
INSERT INTO public.promotion (name, quantity, discount_code, discount_value,
start_date, end_date, is_active, created_at, updated_at)
VALUES ('Khuyến mãi', 50, 'ERROR2025', 205.00, '2025-06-19 03:00:00+07',
'2025-06-20 03:00:00+07', true, CURRENT_TIMESTAMP, CURRENT_TIMESTAMP);
ERROR: Giá trị giảm giá (205.00) phải nằm trong khoảng 0.01 đến 99.99 cho khuyến mãi ID: 2.
CONTEXT: PL/pgSQL function validate_discount_value() line 4 at RAISE

SQL state: P0001
UPDATE public.promotion SET discount_value=-75.5
WHERE name='Khuyến mãi';
ERROR: Giá trị giảm giá (-75.50) phải nằm trong khoảng 0.01 đến 99.99 cho khuyến mãi ID: 41.
CONTEXT: PL/pgSQL function validate_discount_value() line 4 at RAISE

SQL state: P0001
UPDATE public.promotion SET discount_value=45.5
WHERE name='Khuyến mãi';
UPDATE 1

Query returned successfully in 30 msec.
```

### # Nhận xét:

- Tần suất sử dụng trigger không quá lớn, trigger chỉ kiểm tra giá trị của cột trong bảng NEW nên trigger không sử dụng nhiều tài nguyên của hệ thống và đảm bảo giá trị đưa vào đúng với quy tắc đã đặt ra.

#### Câu 4: Function tính doanh thu theo danh mục sản phẩm (category revenue) trong 1 năm

- Function tính và liệt kê doanh thu của các danh mục sản phẩm (category revenue) trong 1 năm với đầu vào là năm cần thống kê.

```
CREATE OR REPLACE FUNCTION GetCategoryRevenue(year_input INT)
RETURNS TABLE (
    category TEXT,
    revenue BIGINT
) AS $$

BEGIN
    RETURN QUERY
    SELECT
        c.name::TEXT AS category,
        SUM(op.quantity * p.price) AS revenue
    FROM orders_order o
    JOIN orders_orderproduct op ON o.id = op.order_id
    JOIN store_product p ON p.id = op.product_id
    JOIN store_category c ON c.id = p.category_id
    WHERE EXTRACT(YEAR FROM o.created_at) = year_input
    GROUP BY c.name;
END;
$$ LANGUAGE plpgsql;
```

# ANALYZE:

```
EXPLAIN ANALYZE
SELECT
    c.name::TEXT AS category,
    SUM(op.quantity * p.price) AS revenue
FROM orders_order o
JOIN orders_orderproduct op ON o.id = op.order_id
JOIN store_product p ON p.id = op.product_id
JOIN store_category c ON c.id = p.category_id
WHERE EXTRACT(YEAR FROM o.created_at) = 2025
GROUP BY c.name;
```

| QUERY PLAN |  |
|------------|--|
| text       |  |
| 1          | GroupAggregate (cost=687.19..688.05 rows=43 width=458) (actual time=5.800..5.867 rows=5 loops=1)   |
| 2          | Group Key: c.name  |
| 3          | -> Sort (cost=687.19..687.30 rows=43 width=426) (actual time=5.790..5.810 rows=739 loops=1)  |
| 4          | Sort Key: c.name   |
| 5          | Sort Method: quicksort Memory: 48kB  |
| 6          | -> Nested Loop (cost=431.29..686.03 rows=43 width=426) (actual time=2.831..5.666 rows=739 loops=1)                                       |
| 7          | -> Nested Loop (cost=431.13..684.00 rows=43 width=16) (actual time=2.818..5.469 rows=739 loops=1)  |
| 8          | -> Hash Join (cost=430.85..664.88 rows=43 width=12) (actual time=2.788..4.136 rows=739 loops=1)  |
| 9          | Hash Cond: (op.order_id = o.id)  |
| 10         | -> Seq Scan on orders_orderproduct op (cost=0.00..211.14 rows=8714 width=20) (actual time=0.045..0.428 rows=8714 loops=1)                |
| 11         | -> Hash (cost=430.27..430.27 rows=46 width=8) (actual time=2.728..2.728 rows=781 loops=1)  |
| 12         | Buckets: 1024 Batches: 1 Memory Usage: 39kB  |
| 13         | -> Seq Scan on orders_order o (cost=0.00..430.27 rows=46 width=8) (actual time=0.016..2.628 rows=781 loops=1)                            |
| 14         | Filter: (EXTRACT(year FROM created_at) = '2025'::numeric)  |
| 15         | Rows Removed by Filter: 8504   |
| 16         | -> Index Scan using store_product_pkey on store_product p (cost=0.29..0.44 rows=1 width=20) (actual time=0.002..0.002 rows=1 loops=1)    |
| 17         | Index Cond: (id = op.product_id)   |
| 18         | -> Memoize (cost=0.15..0.18 rows=1 width=426) (actual time=0.000..0.000 rows=1 loops=739)  |
| 19         | Cache Key: p.category_id   |
| 20         | Cache Mode: logical  |
| 21         | Hits: 734 Misses: 5 Evictions: 0 Overflows: 0 Memory Usage: 1kB  |
| 22         | -> Index Scan using store_category_pkey on store_category c (cost=0.14..0.17 rows=1 width=426) (actual time=0.002..0.002 rows=1 loops=1) |
| 23         | Index Cond: (id = p.category_id)   |
| 24         | Planning Time: 0.638 ms  |
| 25         | Execution Time: 5.921 ms   |

## # TEST:

```
SELECT *
FROM GetCategoryRevenue(2022);
```

|   | category | revenue |
|---|----------|---------|
| 1 | Jackets  | 65710   |
| 2 | Jeans    | 71235   |
| 3 | Shirts   | 1002007 |
| 4 | Shoes    | 69330   |
| 5 | T-Shirts | 60982   |

## # Nhận xét:

- Trong câu lệnh truy vấn có 2 lần Seq Scan tại các bảng: orders\_orderproduct, orders\_order.
- Do tần suất sử dụng của hàm không lớn và chi phí tính toán của 2 lần Seq Scan không ảnh hưởng nhiều đến chi phí tính toán chung của hàm nên không sử dụng index để tối ưu hiệu suất của hàm.

## Câu 5: Function tính thu nhập một tháng cụ thể của cửa hàng do nhân viên cụ thể phụ trách

- Function trả về tổng số tiền các đơn hàng do một nhân viên cụ thể phụ trách trong một tháng cụ thể, đầu vào là id của nhân viên và thời gian theo định dạng 'YYYY-MM'.

```
CREATE OR REPLACE FUNCTION hustshop_monthly_earning(
    IN p_staff_id bigint,
    IN p_month char(7) -- format: 'YYYY-MM'
)
RETURNS TABLE(
    total_recieved NUMERIC
) AS $$

BEGIN
    RETURN QUERY
    SELECT COALESCE(SUM(p.amount_paid), 0) as total_recieved
    -- INTO total_money
    FROM orders_order o
    JOIN orders_payment p ON p.id = o.payment_id
    WHERE o.staff_id = p_staff_id
        AND TO_CHAR(p.created_at, 'YYYY-MM') = p_month;
END;
$$ LANGUAGE plpgsql STABLE;

# ANALYZE:
EXPLAIN ANALYZE
SELECT COALESCE(SUM(p.amount_paid), 0)
FROM orders_order o
JOIN orders_payment p ON p.id = o.payment_id
WHERE o.staff_id = 123
    AND TO_CHAR(p.created_at, 'YYYY-MM') = '2025-06';
```

|   | QUERY PLAN  | text |
|---|---|------|
| 1 | Aggregate (cost=16.95..16.96 rows=1 width=32) (actual time=0.008..0.009 rows=1 loops=1)   |      |
| 2 | -> Nested Loop (cost=0.57..16.94 rows=1 width=6) (actual time=0.006..0.006 rows=0 loops=1)  |      |
| 3 | -> Index Scan using orders_order_staff_id_de0718d4 on orders_order o (cost=0.29..8.30 rows=1 width=8) (actual time=0.005..0.005 rows=0 loops=1) |      |
| 4 | Index Cond: (staff_id = 123)  |      |
| 5 | -> Index Scan using orders_payment_pkey on orders_payment p (cost=0.29..8.31 rows=1 width=14) (never executed)                                  |      |
| 6 | Index Cond: (id = o.payment_id)   |      |
| 7 | Filter: (to_char(created_at, 'YYYY-MM'::text) = '2025-06'::text)  |      |
| 8 | Planning Time: 0.321 ms   |      |
| 9 | Execution Time: 0.031 ms  |      |

## # TEST:

```
SELECT *
FROM hustshop_monthly_earning(1, '2025-01');
```

|   | total_recieved | numeric |
|---|----------------|---------|
| 1 | 144.46         |         |

## # Nhận xét:

- Trong function không tồn tại Seq Scan, tần suất sử dụng hàm không quá lớn và chi phí tính toán không quá lớn.

## Câu 6: Trigger kiểm tra logic thời gian khi thêm hoặc cập nhật các bản ghi khuyến mãi

- Trigger xác thực rằng start\_date (thời điểm bắt đầu khuyến mãi) luôn nhỏ hơn end\_date (thời điểm kết thúc khuyến mãi) khi chèn hoặc cập nhật bản ghi trong bảng public.promotion.

```

CREATE OR REPLACE FUNCTION validate_promotion_dates()
RETURNS TRIGGER AS $$ 
BEGIN
    IF NEW.start_date >= NEW.end_date THEN
        RAISE EXCEPTION 'start_date (%) phải nhỏ hơn end_date (%) cho khuyến mãi ID: .',
                        NEW.start_date, NEW.end_date, NEW.promotion_id;
    END IF;
    NEW.updated_at = CURRENT_TIMESTAMP;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER trigger_validate_promotion_dates
BEFORE INSERT OR UPDATE ON public.promotion
FOR EACH ROW
EXECUTE FUNCTION validate_promotion_dates();

# TEST:
INSERT INTO public.promotion (name, quantity, discount_code, discount_value,
start_date, end_date, is_active, created_at, updated_at)
VALUES ('Khuyến mãi lỗi', 50, 'ERROR2025', 5.00,
'2025-06-19 03:00:00+07', '2025-06-19 03:00:00+07',
true, CURRENT_TIMESTAMP, CURRENT_TIMESTAMP);
ERROR: start_date (2025-06-19 03:00:00+07) phải nhỏ hơn end_date (2025-06-19 03:00:00+07) cho khuyến mãi ID: 104.
CONTEXT: PL/pgSQL function validate_promotion_dates() line 4 at RAISE

SQL state: P0001
UPDATE public.promotion
SET start_date='3000-06-18 03:00:00+07'
WHERE promotion_id=1;
ERROR: start_date (3000-06-18 03:00:00+07) phải nhỏ hơn end_date (2025-06-18 03:00:00+07) cho khuyến mãi ID: 1.
CONTEXT: PL/pgSQL function validate_promotion_dates() line 4 at RAISE

SQL state: P0001
UPDATE public.promotion
SET start_date='2025-03-18 03:00:00+07'
WHERE promotion_id=1;
UPDATE 1

Query returned successfully in 29 msec.

```

### # Nhận xét:

- Tần suất sử dụng trigger không quá lớn, trigger chỉ kiểm tra giá trị của cột trong bảng NEW nên trigger không sử dụng nhiều tài nguyên của hệ thống và đảm bảo giá trị đưa vào đúng với quy tắc đã đặt ra.

## Câu 7: Query thống kê số khách hàng đến hiện tại đã mua theo từng sản phẩm với thứ tự là số lượng khách mua giảm dần

```
SELECT sp.name,Count(oo.id) AS CustomerNum
FROM orders_order AS oo
JOIN orders_orderproduct AS op
  ON oo.id=op.order_id
JOIN orders_orderproduct_variations AS opv
  ON opv.orderproduct_id= op.id
JOIN store_variation AS ov
  ON opv.variation_id=ov.id
JOIN store_product AS sp
  ON ov.product_id=sp.id
GROUP BY sp.id
ORDER BY CustomerNum DESC;

# ANALYZE:
EXPLAIN ANALYZE
SELECT sp.name,Count(oo.id) AS CustomerNum
FROM orders_order AS oo
JOIN orders_orderproduct AS op
  ON oo.id=op.order_id
JOIN orders_orderproduct_variations AS opv
  ON opv.orderproduct_id= op.id
JOIN store_variation AS ov
  ON opv.variation_id=ov.id
JOIN store_product AS sp
  ON ov.product_id=sp.id
GROUP BY sp.id
ORDER BY CustomerNum DESC;
```

| QUERY PLAN |   |
|------------|---|
| text       |   |
| 1          | Sort (cost=4207.90..4231.49 rows=9437 width=35) (actual time=47.159..47.377 rows=5661 loops=1)  |
| 2          | Sort Key: (count(oo.id)) DESC   |
| 3          | Sort Method: quicksort Memory: 506kB  |
| 4          | > HashAggregate (cost=3490.50..3584.87 rows=9437 width=35) (actual time=45.058..46.164 rows=5661 loops=1)                                       |
| 5          | Group Key: sp.id  |
| 6          | Batches: 1 Memory Usage: 1169kB   |
| 7          | > Hash Join (cost=2927.01..3403.36 rows=17428 width=35) (actual time=15.734..39.557 rows=17428 loops=1)   |
| 8          | Hash Cond: (ov.product_id = sp.id)  |
| 9          | > Hash Join (cost=2434.68..2865.25 rows=17428 width=16) (actual time=13.737..32.104 rows=17428 loops=1)   |
| 10         | Hash Cond: (opv.variation_id = ov.id)   |
| 11         | > Hash Join (cost=691.69..1076.51 rows=17428 width=16) (actual time=3.283..13.626 rows=17428 loops=1)   |
| 12         | Hash Cond: (op.order_id = oo.id)  |
| 13         | > Hash Join (cost=320.06..659.12 rows=17428 width=16) (actual time=2.138..8.494 rows=17428 loops=1)   |
| 14         | Hash Cond: (opv.orderproduct_id = op.id)  |
| 15         | > Seq Scan on orders_orderproduct_variations opv (cost=0.00..293.28 rows=17428 width=16) (actual time=0.029..1.261 rows=17428 loops=1)          |
| 16         | > Hash (cost=211.14..211.14 rows=8714 width=16) (actual time=2.057..2.059 rows=8714 loops=1)  |
| 17         | Buckets: 16384 Batches: 1 Memory Usage: 537kB   |
| 18         | > Seq Scan on orders_orderproduct op (cost=0.00..211.14 rows=8714 width=16) (actual time=0.011..1.066 rows=8714 loops=1)                        |
| 19         | > Hash (cost=255.56..255.56 rows=9285 width=8) (actual time=1.093..1.094 rows=9285 loops=1)   |
| 20         | Buckets: 16384 Batches: 1 Memory Usage: 491kB   |
| 21         | > Index Only Scan using orders_order_pkey on orders_order oo (cost=0.29..255.56 rows=9285 width=8) (actual time=0.020..0.601 rows=9285 loops=1) |
| 22         | Heap Fetches: 1574  |
| 23         | > Hash (cost=1035.22..1035.22 rows=56622 width=16) (actual time=10.302..10.302 rows=56622 loops=1)  |
| 24         | Buckets: 65536 Batches: 1 Memory Usage: 3167kB  |
| 25         | > Seq Scan on store_variation ov (cost=0.00..1035.22 rows=56622 width=16) (actual time=0.016..4.885 rows=56622 loops=1)                         |
| 26         | > Hash (cost=374.37..374.37 rows=9437 width=27) (actual time=1.947..1.947 rows=9437 loops=1)  |
| 27         | Buckets: 16384 Batches: 1 Memory Usage: 724kB   |
| 28         | > Seq Scan on store_product sp (cost=0.00..374.37 rows=9437 width=27) (actual time=0.025..0.874 rows=9437 loops=1)                              |
| 29         | Planning Time: 1.023 ms   |
| 30         | Execution Time: 48.671 ms   |

## # TEST:

```

SELECT sp.name,Count(oo.id) AS CustomerNum
FROM orders_order AS oo
JOIN orders_orderproduct AS op
  ON oo.id=op.order_id
JOIN orders_orderproduct_variations AS opv
  ON opv.orderproduct_id= op.id
JOIN store_variation AS ov
  ON opv.variation_id=ov.id
JOIN store_product AS sp
  ON ov.product_id=sp.id
GROUP BY sp.id
ORDER BY CustomerNum DESC
LIMIT 10;

```

## # Nhận xét:

- Trong câu lệnh truy vấn có 4 lần Seq Scan tại các bảng: orders\_orderproduct\_variations, orders\_orderproduct, store\_variation, store\_product.
- Do tần suất sử dụng của hàm không lớn và chi phí tính toán của 4 lần Seq Scan không ảnh hưởng nhiều đến chi phí tính toán chung của hàm nên không sử dụng index để tối ưu hiệu suất của hàm.

|    | name<br>character varying (200) | customernum<br>bigint |
|----|---------------------------------|-----------------------|
| 1  | Special Item 4450               | 12                    |
| 2  | Special Item 4077               | 12                    |
| 3  | Loafers Beige Max               | 12                    |
| 4  | Special Item 4896               | 12                    |
| 5  | Special Item 2583               | 12                    |
| 6  | Special Item 615                | 12                    |
| 7  | Special Item 3631               | 10                    |
| 8  | Special Item 2342               | 10                    |
| 9  | Special Item 2004               | 10                    |
| 10 | Brown Cardigan                  | 10                    |

### Câu 8 : Function đếm số lượng đã mua của từng sản phẩm

- Function trả về id, tên và tổng số lượng đã bán tính đến hiện tại của các sản phẩm trong database.

```
CREATE OR REPLACE FUNCTION hustshop_product_purchase_stats()
RETURNS TABLE (
    product_id BIGINT,
    name TEXT,
    total_quantity BIGINT
) AS $$

BEGIN
    RETURN QUERY
    SELECT
        p.id,
        p.name::TEXT,
        SUM(op.quantity)
    FROM orders_orderproduct op
    JOIN store_product p ON p.id = op.product_id
    GROUP BY p.id, p.name;
END;
$$ LANGUAGE plpgsql STABLE;

# ANALYZE:
EXPLAIN ANALYZE
SELECT
    p.id,
    p.name::TEXT,
    SUM(op.quantity) AS total_quantity
FROM orders_orderproduct op
JOIN store_product p ON p.id = op.product_id
GROUP BY p.id, p.name;
```

|    | QUERY PLAN                            | text  | 🔒                   |
|----|---------------------------------------|---|---------------------|
| 1  | HashAggregate                         | (cost=769.93..857.07 rows=8714 width=67) (actual time=6.925..7.601 rows=5661 loops=1) |                     |
| 2  | Group Key: p.id                       |   |                     |
| 3  | Batches: 1                            | Memory Usage: 1169kB  |                     |
| 4  | -> Hash Join                          | (cost=492.33..726.36 rows=8714 width=31) (actual time=1.387..4.899 rows=8714 loops=1) |                     |
| 5  | Hash Cond: (op.product_id = p.id)     |   |                     |
| 6  | -> Seq Scan on orders_orderproduct op | (cost=0.00..211.14 rows=8714 width=12) (actual time=0.022..0.532 rows=8714 loops=1)   |                     |
| 7  | -> Hash                               | (cost=374.37..374.37 rows=9437 width=27) (actual time=1.268..1.269 rows=9437 loops=1) |                     |
| 8  | Buckets: 16384                        | Batches: 1  | Memory Usage: 674kB |
| 9  | -> Seq Scan on store_product p        | (cost=0.00..374.37 rows=9437 width=27) (actual time=0.014..0.610 rows=9437 loops=1)   |                     |
| 10 | Planning Time:                        | 0.459 ms  |                     |
| 11 | Execution Time:                       | 8.259 ms  |                     |

# TEST:

```
SELECT *
FROM hustshop_product_purchase_stats()
ORDER BY total_quantity DESC
LIMIT 10;
```

|    | product_id<br>bigint | name<br>text      | total_quantity<br>bigint |
|----|----------------------|-------------------|--------------------------|
| 1  | 2249                 | Brown Cardigan    | 23                       |
| 2  | 3166                 | Special Item 615  | 22                       |
| 3  | 7447                 | Special Item 4896 | 22                       |
| 4  | 5134                 | Special Item 2583 | 20                       |
| 5  | 2246                 | Beige Windbrea... | 20                       |
| 6  | 6602                 | Special Item 4051 | 19                       |
| 7  | 3743                 | Special Item 1192 | 19                       |
| 8  | 7486                 | Special Item 4935 | 19                       |
| 9  | 5867                 | Special Item 3316 | 18                       |
| 10 | 3911                 | Special Item 1360 | 18                       |

#### # Nhận xét:

- Trong câu lệnh truy vấn có 2 lần Seq Scan tại các bảng: orders\_orderproduct, store\_product.
- Do tần suất sử dụng của hàm không lớn và chi phí tính toán của 2 lần Seq Scan không ảnh hưởng nhiều đến chi phí tính toán chung của hàm nên không sử dụng index để tối ưu hiệu suất của hàm.

### Câu 9: Query tìm các mẫu sản phẩm chưa từng được thêm vào bất kỳ đơn hàng nào

- Query trả về tên, thể loại và giá trị thể loại của các sản phẩm chưa bán được sản phẩm nào.

```
SELECT sp.name,sv.variation_category,sv.variation_value
FROM orders_orderproduct AS op
JOIN orders_orderproduct_variations AS opv
    ON opv.orderproduct_id= op.id
RIGHT JOIN store_variation AS sv
    ON opv.variation_id=sv.id
JOIN store_product AS sp
    ON sv.product_id=sp.id
WHERE op.id IS NULL;
```

#### # ANALYZE:

```
EXPLAIN ANALYZE
SELECT sp.name,sv.variation_category,sv.variation_value
FROM orders_orderproduct AS op
JOIN orders_orderproduct_variations AS opv
    ON opv.orderproduct_id= op.id
RIGHT JOIN store_variation AS sv
    ON opv.variation_id=sv.id
JOIN store_product AS sp
    ON sv.product_id=sp.id
WHERE op.id IS NULL;
```

| QUERY PLAN |  |
|------------|--|
| text       |  |
| 1          | Nested Loop (cost=2063.35..2448.18 rows=1 width=28) (actual time=16.823..87.757 rows=41624 loops=1)                                      |
| 2          | -> Hash Right Join (cost=2063.06..2447.86 rows=1 width=17) (actual time=16.794..29.319 rows=41624 loops=1)                               |
| 3          | Hash Cond: (opv.variation_id = sv.id)  |
| 4          | Filter: (op.id IS NULL)  |
| 5          | Rows Removed by Filter: 17428  |
| 6          | -> Hash Join (cost=320.06..659.12 rows=17428 width=16) (actual time=0.967..3.525 rows=17428 loops=1)                                     |
| 7          | Hash Cond: (opv.orderproduct_id = op.id)   |
| 8          | -> Seq Scan on orders_orderproduct_variations opv (cost=0.00..293.28 rows=17428 width=16) (actual time=0.016..0.588 rows=1742...         |
| 9          | -> Hash (cost=211.14..211.14 rows=8714 width=8) (actual time=0.909..0.910 rows=8714 loops=1)   |
| 10         | Buckets: 16384 Batches: 1 Memory Usage: 469kB  |
| 11         | -> Seq Scan on orders_orderproduct op (cost=0.00..211.14 rows=8714 width=8) (actual time=0.006..0.442 rows=8714 loops=1)                 |
| 12         | -> Hash (cost=1035.22..1035.22 rows=56622 width=25) (actual time=10.024..10.025 rows=56622 loops=1)                                      |
| 13         | Buckets: 65536 Batches: 1 Memory Usage: 3867kB   |
| 14         | -> Seq Scan on store_variation sv (cost=0.00..1035.22 rows=56622 width=25) (actual time=0.011..3.799 rows=56622 loops=1)                 |
| 15         | -> Index Scan using store_product_pkey on store_product sp (cost=0.29..0.32 rows=1 width=27) (actual time=0.001..0.001 rows=1 loops=4... |
| 16         | Index Cond: (id = sv.product_id)   |
| 17         | Planning Time: 0.380 ms  |
| 18         | Execution Time: 89.557 ms  |

#### # TEST:

```

SELECT sp.name,sv.variation_category,sv.variation_value
FROM orders_orderproduct AS op
JOIN orders_orderproduct_variations AS opv
  ON opv.orderproduct_id= op.id
RIGHT JOIN store_variation AS sv
  ON opv.variation_id=sv.id
JOIN store_product AS sp
  ON sv.product_id=sp.id
WHERE op.id IS NULL
LIMIT 10;

```

|    | name<br>character varying (200)  | variation_category<br>character varying (100) | variation_value<br>character varying (100) |
|----|----------------------------------|---|--|
| 1  | Special Item 3576                | size  | M  |
| 2  | Polo Shirt Collection            | size  | S  |
| 3  | Special Item 6986                | size  | XXL  |
| 4  | Vintage Black Formal White Shirt | size  | M  |
| 5  | Modern Black Running Shoes       | size  | XL   |
| 6  | Special Item 1160                | color   | Red  |
| 7  | Special Item 4060                | size  | S  |
| 8  | Premium Brown Basic Cotton T...  | color   | Red  |
| 9  | Special Item 6852                | color   | Yellow                                     |
| 10 | Special Item 5626                | color   | Green                                      |

#### # Nhận xét:

- Trong câu lệnh truy vấn có 3 lần Seq Scan tại các bảng: orders\_orderproduct\_variations, orders\_orderproduct, store\_variation.

- Do tần suất sử dụng của hàm không lớn và chi phí tính toán của 3 lần Seq Scan không ảnh hưởng nhiều đến chi phí tính toán chung của query nên không sử dụng index để tối ưu hiệu suất của query.

### Câu 10: Query truy vấn tên và doanh thu trong một khoảng thời gian của tất cả các sản phẩm và sắp xếp theo thứ tự doanh thu giảm dần

- Query trả về tên và doanh thu của các sản phẩm bán được trong khoảng thời gian cho trước [StartTime, EndTime].

```

SELECT name, SUM(quantity)*price AS Revenue
FROM orders_orderproduct AS op
JOIN orders_orderproduct_variations AS opv
  ON opv.orderproduct_id= op.id
JOIN store_variation AS ov
  ON opv.variation_id=ov.id
JOIN store_product AS sp
  ON ov.product_id=sp.id
WHERE op.updated_at>= ['StartTime'] AND op.updated_at<= ['EndTime']
GROUP BY sp.id
ORDER BY Revenue DESC;

# ANALYZE:
EXPLAIN ANALYZE
SELECT name, SUM(quantity)*price AS Revenue
FROM orders_orderproduct AS op
JOIN orders_orderproduct_variations AS opv
  ON opv.orderproduct_id= op.id
JOIN store_variation AS ov
  ON opv.variation_id=ov.id
JOIN store_product AS sp
  ON ov.product_id=sp.id
WHERE op.updated_at>='2025-01-01' AND op.updated_at<= '2025-06-20'
GROUP BY sp.id
ORDER BY Revenue DESC;

```

|    | QUERY PLAN   |
|----|--|
| 1  | text   |
| 1  | Sort (cost=1909.14..1913.23 rows=1638 width=35) (actual time=10.296..10.327 rows=779 loops=1)  |
| 2  | Sort Key: ((sum(op.quantity) * sp.price)) DESC   |
| 3  | Sort Method: quicksort Memory: 68kB  |
| 4  | -> HashAggregate (cost=1801.21..1821.69 rows=1638 width=35) (actual time=10.041..10.140 rows=779 loops=1)                              |
| 5  | Group Key: sp.id   |
| 6  | Batches: 1 Memory Usage: 193kB   |
| 7  | -> Hash Join (cost=1366.88..1793.02 rows=1638 width=35) (actual time=7.708..9.753 rows=1626 loops=1)                                   |
| 8  | Hash Cond: (sp.id = ov.product_id)   |
| 9  | -> Seq Scan on store_product sp (cost=0.00..374.37 rows=9437 width=31) (actual time=0.022..0.574 rows=9437 loops=1)                    |
| 10 | -> Hash (cost=1346.41..1346.41 rows=1638 width=12) (actual time=7.668..7.670 rows=1626 loops=1)  |
| 11 | Buckets: 2048 Batches: 1 Memory Usage: 93kB  |
| 12 | -> Nested Loop (cost=265.24..1346.41 rows=1638 width=12) (actual time=1.212..7.259 rows=1626 loops=1)                                  |
| 13 | -> Hash Join (cost=264.95..604.00 rows=1638 width=12) (actual time=1.192..3.775 rows=1626 loops=1)                                     |
| 14 | Hash Cond: (opv.orderproduct_id = op.id)   |
| 15 | -> Seq Scan on orders_orderproduct_variations opv (cost=0.00..293.28 rows=17428 width=16) (actual time=0.019..1.004 rows=17...         |
| 16 | -> Hash (cost=254.71..254.71 rows=819 width=12) (actual time=1.160..1.161 rows=813 loops=1)  |
| 17 | Buckets: 1024 Batches: 1 Memory Usage: 47kB  |
| 18 | -> Seq Scan on orders_orderproduct op (cost=0.00..254.71 rows=819 width=12) (actual time=0.018..1.058 rows=813 loops=1)                |
| 19 | Filter: ((updated_at >= '2025-01-01 00:00:00+07'::timestamp with time zone) AND (updated_at <= '2025-06-20 00:00:00+07'::t...          |
| 20 | Rows Removed by Filter: 7901   |
| 21 | -> Index Scan using store_variation_pkey on store_variation ov (cost=0.29..0.45 rows=1 width=16) (actual time=0.002..0.002 rows=1 l... |
| 22 | Index Cond: (id = opv.variation_id)  |
| 23 | Planning Time: 0.957 ms  |
| 24 | Execution Time: 10.502 ms  |

`CREATE INDEX idx_btree_OrderUpdatedTime ON orders_orderproduct(updated_at);`

`CREATE INDEX`

Query returned successfully in 42 msec.

|    | QUERY PLAN  |
|----|---|
| 1  | text  |
| 1  | Sort (cost=1811.39..1815.49 rows=1638 width=35) (actual time=4.848..4.870 rows=779 loops=1)   |
| 2  | Sort Key: ((sum(op.quantity) * sp.price)) DESC  |
| 3  | Sort Method: quicksort Memory: 68kB   |
| 4  | -> HashAggregate (cost=1703.47..1723.94 rows=1638 width=35) (actual time=4.677..4.745 rows=779 loops=1)                                     |
| 5  | Group Key: sp.id  |
| 6  | Batches: 1 Memory Usage: 193kB  |
| 7  | -> Hash Join (cost=1269.14..1695.28 rows=1638 width=35) (actual time=3.508..4.486 rows=1626 loops=1)  |
| 8  | Hash Cond: (sp.id = ov.product_id)  |
| 9  | -> Seq Scan on store_product sp (cost=0.00..374.37 rows=9437 width=31) (actual time=0.015..0.350 rows=9437 loops=1)                         |
| 10 | -> Hash (cost=1248.66..1248.66 rows=1638 width=12) (actual time=3.481..3.483 rows=1626 loops=1)   |
| 11 | Buckets: 2048 Batches: 1 Memory Usage: 93kB   |
| 12 | -> Nested Loop (cost=167.49..1248.66 rows=1638 width=12) (actual time=0.265..3.357 rows=1626 loops=1)                                       |
| 13 | -> Hash Join (cost=167.20..506.25 rows=1638 width=12) (actual time=0.257..1.714 rows=1626 loops=1)  |
| 14 | Hash Cond: (opv.orderproduct_id = op.id)  |
| 15 | -> Seq Scan on orders_orderproduct_variations opv (cost=0.00..293.28 rows=17428 width=16) (actual time=0.005..0.548 rows=17428 lo...        |
| 16 | -> Hash (cost=156.96..156.96 rows=819 width=12) (actual time=0.243..0.244 rows=813 loops=1)   |
| 17 | Buckets: 1024 Batches: 1 Memory Usage: 47kB   |
| 18 | -> Bitmap Heap Scan on orders_orderproduct op (cost=20.68..156.96 rows=819 width=12) (actual time=0.045..0.189 rows=813 loop...             |
| 19 | Recheck Cond: ((updated_at >= '2025-01-01 00:00:00+07'::timestamp with time zone) AND (updated_at <= '2025-06-20 00:00:00+0...              |
| 20 | Heap Blocks: exact=124  |
| 21 | -> Bitmap Index Scan on idx_btree_orderupdatedtime (cost=0.00..20.47 rows=819 width=0) (actual time=0.031..0.031 rows=813 l...              |
| 22 | Index Cond: ((updated_at >= '2025-01-01 00:00:00+07'::timestamp with time zone) AND (updated_at <= '2025-06-20 00:00:00+0...                |
| 23 | -> Index Scan using store_variation_pkey on store_variation ov (cost=0.29..0.45 rows=1 width=16) (actual time=0.001..0.001 rows=1 loops=... |
| 24 | Index Cond: (id = opv.variation_id)   |
| 25 | Planning Time: 0.475 ms   |
| 26 | Execution Time: 4.995 ms  |

## # TEST:

```

SELECT name, SUM(quantity)*price AS Revenue
FROM orders_orderproduct AS op
JOIN orders_orderproduct_variations AS opv
  ON opv.orderproduct_id= op.id
JOIN store_variation AS ov
  ON opv.variation_id=ov.id
JOIN store_product AS sp
  ON ov.product_id=sp.id
WHERE op.updated_at>='2025-01-01' AND op.updated_at<= '2025-06-20'
GROUP BY sp.id
ORDER BY Revenue DESC
LIMIT 10;

```

|    | name<br>character varying (200) | revenue<br>bigint |
|----|---------------------------------|-------------------|
| 1  | Special Item 7230               | 8540              |
| 2  | Special Item 3793               | 6480              |
| 3  | Special Item 4566               | 6176              |
| 4  | Modern Navy Sandals             | 6112              |
| 5  | Premium White Raw Deni...       | 5776              |
| 6  | Special Item 4334               | 5026              |
| 7  | Special Item 432                | 4980              |
| 8  | Special Item 226                | 4970              |
| 9  | Special Item 5760               | 4960              |
| 10 | Special Item 6917               | 4900              |

## # Nhận xét:

- Trong câu lệnh truy vấn có 3 lần Seq Scan tại các bảng: store\_product, orders\_orderproduct\_variations, orders\_orderproduct.
- Trong dashboard của manager, tần suất sử dụng query này ở mức trung bình, để tối ưu tốc độ truy vấn, sử dụng index dạng B-tree đối với cột updated\_at trong bảng orders\_orderproduct đã giúp query chạy nhanh gấp đôi so với trước khi sử dụng index.
- Lý do là vì B-tree cho khả năng truy vấn nhanh dữ liệu trong một khoảng xác định, bảng orders\_orderproduct có rất nhiều dữ liệu và trong query ta cần lọc ra những order có thời gian được cập nhật nằm trong khoảng thời gian cho trước.

## 5.2. 10 câu truy vấn của Nguyễn Minh Quân

### Truy vấn 1: Top khách hàng có tổng số chi tiêu từ đơn hàng lớn nhất (Nếu cùng số tiền thì đưa ra tất)

Cách 1 : SubQuery

```
SELECT a.username, a.email,
       COUNT(o.id) as order_count,
       SUM(o.order_total) as total_spent
  FROM accounts_account a
  JOIN orders_order o ON a.id = o.user_id
 WHERE o.is_ordered = true
 GROUP BY a.id, a.username, a.email
 HAVING SUM(o.order_total) >= ALL (
   SELECT SUM(o2.order_total)
   FROM accounts_account a2
   JOIN orders_order o2 ON a2.id = o2.user_id
   WHERE o2.is_ordered = true
   GROUP BY a2.id
 )
 ORDER BY total_spent DESC;
```

Cách 2: CTE ( Common Table Expression )

```
WITH tmp AS (
  SELECT
    a.username,
    a.email,
    COUNT(o.id) AS order_count,
    SUM(o.order_total) AS order_total
   FROM accounts_account a
   JOIN orders_order o ON a.id = o.user_id
  WHERE o.is_ordered = true
  GROUP BY a.id, a.username, a.email
)
SELECT *
  FROM tmp
 WHERE tmp.order_total >= ALL (
   SELECT order_total FROM tmp
 );
```

### Cách 3: CTE với Window function RANK()

```
WITH RankedAccounts AS (
    SELECT
        a.username,
        a.email,
        COUNT(o.id) AS order_count,
        SUM(o.order_total) AS total_spent,
        RANK() OVER (ORDER BY SUM(o.order_total) DESC) AS rnk
    FROM accounts_account a
    JOIN orders_order o ON a.id = o.user_id
    WHERE o.is_ordered = true
    GROUP BY a.id, a.username, a.email
)
SELECT username, email, order_count, total_spent
FROM RankedAccounts
WHERE rnk = 1
ORDER BY total_spent DESC;
```

Kết quả :

|   | username<br>character varying (50)  | email<br>character varying (254)  | order_count<br>bigint  | total_spent<br>numeric  |
|---|--|--|---|--|
| 1 | ibuckley   | ibuckley@example.com   | 6   | 2561.97  |

### Phân tích hiệu năng

#### Cách 1: SubQuery

| QUERY PLAN<br>text   |
|--|
| 4   -> HashAggregate (cost=1752.72..875370.07 rows=4673 width=80) (actual time=27.768..29.804 rows=1 loops=1)                                    |
| 5    Group Key: a.id   |
| 6    Filter: (ALL (sum(o.order_total) >= (SubPlan 1).col1))  |
| 7    Batches: 1 Memory Usage: 2961kB   |
| 8    Rows Removed by Filter: 5936  |
| 9    -> Hash Join (cost=449.67..852.67 rows=9346 width=54) (actual time=2.232..5.886 rows=9346 loops=1)  |
| 10    Hash Cond: (o.user_id = a.id)  |
| 11    -> Seq Scan on orders_order o (cost=0.00..378.46 rows=9346 width=22) (actual time=0.016..1.804 rows=9346 loops=1)                          |
| 12    Filter: is_ordered   |
| 13    -> Hash (cost=331.52..331.52 rows=9452 width=40) (actual time=2.163..2.164 rows=9452 loops=1)  |
| 14    Buckets: 16384 Batches: 1 Memory Usage: 838kB  |
| 15    -> Seq Scan on accounts_account a (cost=0.00..331.52 rows=9452 width=40) (actual time=0.007..1.064 rows=9452 loops=1)                      |
| 16    SubPlan 1  |
| 17     -> Materialize (cost=829.95..993.50 rows=9346 width=40) (actual time=0.001..0.002 rows=16 loops=5937)                                     |
| 18     -> HashAggregate (cost=829.95..946.77 rows=9346 width=40) (actual time=7.042..8.400 rows=5937 loops=1)                                    |
| 19      Group Key: a2.id   |
| 20      Batches: 1 Memory Usage: 2705kB  |
| 21      -> Hash Join (cost=380.22..783.22 rows=9346 width=14) (actual time=1.421..4.350 rows=9346 loops=1)                                       |
| 22      Hash Cond: (o2.user_id = a2.id)  |
| 23      -> Seq Scan on orders_order o2 (cost=0.00..378.46 rows=9346 width=14) (actual time=0.013..1.534 rows=9346 loops=1)                       |
| 24      Filter: is_ordered   |
| 25      -> Hash (cost=262.06..262.06 rows=9452 width=8) (actual time=1.346..1.347 rows=9452 loops=1)   |
| 26      Buckets: 16384 Batches: 1 Memory Usage: 498kB  |
| 27      -> Index Only Scan using accounts_account_pkey on accounts_account a2 (cost=0.29..262.06 rows=9452 width=8) (actual time=0.017..0.66...) |
| 28      Heap Fetches: 0  |
| 29    Planning Time: 0.689 ms  |

## Cách 2: CTE ( Common Table Expression )

| QUERY PLAN  |  |
|---|--|
| text  |  |
| CTE Scan on tmp (cost=1039.59..983911.68 rows=4673 width=674) (actual time=26.065..28.622 rows=1 loops=1)             |  |
| Filter: (ALL (order_total >= (SubPlan 2).col1))   |  |
| Rows Removed by Filter: 5936  |  |
| CTE tmp   |  |
| -> HashAggregate (cost=922.77..1039.59 rows=9346 width=80) (actual time=11.013..12.986 rows=5937 loops=1)             |  |
| Group Key: a.id   |  |
| Batches: 1 Memory Usage: 2961kB   |  |
| -> Hash Join (cost=449.67..852.67 rows=9346 width=54) (actual time=3.702..7.686 rows=9346 loops=1)                    |  |
| Hash Cond: (o.user_id = a.id)   |  |
| -> Seq Scan on orders_order o (cost=0.00..378.46 rows=9346 width=22) (actual time=0.028..2.002 rows=9346 loops=1)     |  |
| Filter: is_ordered  |  |
| -> Hash (cost=331.52..331.52 rows=9452 width=40) (actual time=3.573..3.577 rows=9452 loops=1)                         |  |
| Buckets: 16384 Batches: 1 Memory Usage: 838kB   |  |
| -> Seq Scan on accounts_account a (cost=0.00..331.52 rows=9452 width=40) (actual time=0.021..2.177 rows=9452 loops=1) |  |
| SubPlan 2   |  |
| -> CTE Scan on tmp tmp_1 (cost=0.00..186.92 rows=9346 width=32) (actual time=0.000..0.002 rows=16 loops=5937)         |  |
| Planning Time: 0.316 ms   |  |
| Execution Time: 29.980 ms   |  |

## Cách 3: CTE với RANK

|      | QUERY PLAN  |
|------|---|
| text |   |
| 1    | Subquery Scan on rankedaccounts (cost=1655.99..1936.35 rows=47 width=72) (actual time=13.883..13.889 rows=1 loops=1)  |
| 2    | Filter: (rankedaccounts.rnk = 1)  |
| 3    | -> WindowAgg (cost=1655.99..1819.52 rows=9346 width=88) (actual time=13.882..13.886 rows=1 loops=1)                   |
| 4    | Run Condition: (rank() OVER (?: <= 1))  |
| 5    | -> Sort (cost=1655.97..1679.33 rows=9346 width=80) (actual time=13.875..13.877 rows=2 loops=1)                        |
| 6    | Sort Key: (sum(o.order_total)) DESC   |
| 7    | Sort Method: quicksort Memory: 638kB  |
| 8    | -> HashAggregate (cost=922.77..1039.59 rows=9346 width=80) (actual time=9.224..12.291 rows=5937 loops=1)              |
| 9    | Group Key: a.id   |
| 10   | Batches: 1 Memory Usage: 2961kB   |
| 11   | -> Hash Join (cost=449.67..852.67 rows=9346 width=54) (actual time=1.948..5.723 rows=9346 loops=1)                    |
| 12   | Hash Cond: (o.user_id = a.id)   |
| 13   | -> Seq Scan on orders_order o (cost=0.00..378.46 rows=9346 width=22) (actual time=0.009..1.634 rows=9346 loops=1)     |
| 14   | Filter: is_ordered  |
| 15   | -> Hash (cost=331.52..331.52 rows=9452 width=40) (actual time=1.925..1.925 rows=9452 loops=1)                         |
| 16   | Buckets: 16384 Batches: 1 Memory Usage: 838kB   |
| 17   | -> Seq Scan on accounts_account a (cost=0.00..331.52 rows=9452 width=40) (actual time=0.004..0.869 rows=9452 loops=1) |
| 18   | Planning Time: 0.282 ms   |
| 19   | Execution Time: 14.318 ms   |

### Giải thích :

- Trong 3 cách này, dễ thấy cách **CTE với RANK()** đạt hiệu quả cao nhất bởi RANK() chỉ cần sắp xếp dữ liệu một lần theo total\_spent và gán thứ hạng,

WindowAgg và Sort thay thế cho việc sử dụng ALL, giúp giảm đáng kể chi phí so sánh vì không cần lặp lại subquery hoặc CTE scan nhiều lần.

- Sử dụng CTE giúp tính toán tổng order\_total một lần và lưu vào bảng tạm tmp, sau đó tái sử dụng kết quả này trong điều kiện WHERE. Điều này giảm chi phí so với subquery vì không cần tính toán lại SUM(o.order\_total) cho mỗi nhóm trong WHERE. Tuy nhiên do vẫn sử dụng ALL trong mệnh đề WHERE dẫn tới việc so sánh toàn bộ kết quả trong CTE, gây chi phí đáng kể ( loops = 5937 )
- Do việc phải so sánh người dùng có tổng order\_total lớn nhất bằng cách so sánh với tất cả các giá trị SUM (o2.order\_total) trong Subquery dẫn tới việc Subquery phải lặp lại nhiều lần, việc tính lại SUM(o2.order\_total) cho toàn bộ dữ liệu mỗi lần, không có cơ chế lưu tạm như CTE → Tốn kém chi phí

## Truy vấn 2: Danh sách khách hàng chưa mua sản phẩm nào trong tháng 3/2025

Cách 1: Sử dụng LEFT JOIN với điều kiện

```
SELECT a.*  
FROM accounts_account a  
LEFT JOIN orders_order o ON a.id = o.user_id  
    AND o.is_ordered = true  
    AND o.created_at >= '2025-03-01'  
    AND o.created_at < '2025-04-01'  
WHERE o.id IS NULL;
```

Cách 2: Sử dụng NOT EXISTS

```
SELECT a.*  
FROM accounts_account a  
WHERE NOT EXISTS (  
    SELECT 1  
    FROM orders_order o  
    WHERE o.user_id = a.id  
    AND o.is_ordered = true  
    AND o.created_at >= '2025-03-01'  
    AND o.created_at < '2025-04-01'  
);
```

- Trong 2 cách trên thì cách sử dụng NOT EXISTS sẽ tối ưu hơn là LEFT JOIN bởi NOT EXISTS không cần thực hiện JOIN giữa các bảng còn LEFT JOIN phải tạo bảng tạm chứa toàn bộ kết quả kết nối trước khi lọc. Khi có nhiều bản ghi thỏa mãn điều kiện join, thì LEFT JOIN sẽ phải tốn nhiều chi phí và bộ nhớ hơn.
- Trong cả 2 cách truy vấn trên, ta có thể tăng hiệu năng bằng cách sử dụng index

```
CREATE INDEX idx_orders_created_at ON orders_order(created_at);
```

- Bởi chúng ta chỉ đang cần lấy dữ liệu trên tháng 3, trong khi cột created\_at lại chứa rất nhiều dữ liệu. Hiểu đơn giản thì khi ta dùng index thì hệ thống sẽ nhảy tới vị trí đầu tháng 3/2025 để quét tới 31/3/2025 thay vì quét toàn bộ bảng (từ năm 2020 đến hiện tại) → Dùng index giảm chi phí đi rất nhiều

### Khi không sử dụng index với cách 1

| QUERY PLAN text |   |
|-----------------|---|
| 1               | Hash Left Join (cost=432.97..872.19 rows=1 width=145) (actual time=1.111..4.148 rows=9321 loops=1)  |
| 2               | Hash Cond: (a.id = o.user_id)   |
| 3               | Filter: (o.id IS NULL)  |
| 4               | Rows Removed by Filter: 133   |
| 5               | -> Seq Scan on accounts_account a (cost=0.00..331.52 rows=9452 width=145) (actual time=0.012..1.745 rows=9452 loops=1)  |
| 6               | -> Hash (cost=431.27..431.27 rows=136 width=16) (actual time=1.092..1.093 rows=133 loops=1)   |
| 7               | Buckets: 1024 Batches: 1 Memory Usage: 15kB   |
| 8               | -> Seq Scan on orders_order o (cost=0.00..431.27 rows=136 width=16) (actual time=0.018..1.077 rows=133 loops=1)   |
| 9               | Filter: (is_ordered AND (created_at >= '2025-03-01 00:00:00+07'::timestamp with time zone) AND (created_at < '2025-04-01 00:00:00+07'::timestamp with time zone)) |
| 10              | Rows Removed by Filter: 9152  |
| 11              | Planning Time: 1.849 ms   |
| 12              | Execution Time: 4.426 ms  |

### Khi sử dụng index với cách 1

| QUERY PLAN text |  |
|-----------------|--|
| 1               | Hash Left Join (cost=248.11..687.32 rows=1 width=145) (actual time=0.164..2.683 rows=9321 loops=1)   |
| 2               | Hash Cond: (a.id = o.user_id)  |
| 3               | Filter: (o.id IS NULL)   |
| 4               | Rows Removed by Filter: 133  |
| 5               | -> Seq Scan on accounts_account a (cost=0.00..331.52 rows=9452 width=145) (actual time=0.010..1.461 rows=9452 loops=1)                                   |
| 6               | -> Hash (cost=246.41..246.41 rows=136 width=16) (actual time=0.145..0.146 rows=133 loops=1)  |
| 7               | Buckets: 1024 Batches: 1 Memory Usage: 15kB  |
| 8               | -> Bitmap Heap Scan on orders_order o (cost=5.68..246.41 rows=136 width=16) (actual time=0.049..0.133 rows=133 loops=1)                                  |
| 9               | Recheck Cond: ((created_at >= '2025-03-01 00:00:00+07'::timestamp with time zone) AND (created_at < '2025-04-01 00:00:00+07'::timestamp with time zone)) |
| 10              | Filter: is_ordered   |
| 11              | Heap Blocks: exact=107   |
| 12              | -> Bitmap Index Scan on idx_orders_created_at (cost=0.00..5.65 rows=136 width=0) (actual time=0.036..0.036 rows=133 loops=1)                             |
| 13              | Index Cond: ((created_at >= '2025-03-01 00:00:00+07'::timestamp with time zone) AND (created_at < '2025-04-01 00:00:00+07'::timestamp with time zone))   |
| 14              | Planning Time: 1.936 ms  |
| 15              | Execution Time: 2.917 ms   |

**Nhận xét :** Ta có thể thấy cost giảm đáng kể từ 872.19 → 687.3

## Khi không sử dụng index với cách 2

| QUERY PLAN<br>text |   |
|--------------------|---|
| 1                  | Hash Anti Join (cost=432.97..883.64 rows=9316 width=145) (actual time=1.017..4.095 rows=9321 loops=1)   |
| 2                  | Hash Cond: (a.id = o.user_id)   |
| 3                  | -> Seq Scan on accounts_account a (cost=0.00..331.52 rows=9452 width=145) (actual time=0.008..1.837 rows=9452 loops=1)                                    |
| 4                  | -> Hash (cost=431.27..431.27 rows=136 width=8) (actual time=1.003..1.004 rows=133 loops=1)  |
| 5                  | Buckets: 1024 Batches: 1 Memory Usage: 14kB   |
| 6                  | -> Seq Scan on orders_order o (cost=0.00..431.27 rows=136 width=8) (actual time=0.017..0.990 rows=133 loops=1)  |
| 7                  | Filter: (is_ordered AND (created_at >= '2025-03-01 00:00:00+07':timestamp with time zone) AND (created_at < '2025-04-01 00:00:00+07':timestamp with ti... |
| 8                  | Rows Removed by Filter: 9152  |
| 9                  | Planning Time: 0.239 ms   |
| 10                 | Execution Time: 4.387 ms  |

## Khi sử dụng index với cách 2

| QUERY PLAN<br>text |  |
|--------------------|--|
| 1                  | Hash Anti Join (cost=248.11..698.77 rows=9316 width=145) (actual time=0.160..2.639 rows=9321 loops=1)  |
| 2                  | Hash Cond: (a.id = o.user_id)  |
| 3                  | -> Seq Scan on accounts_account a (cost=0.00..331.52 rows=9452 width=145) (actual time=0.009..1.545 rows=9452 loops=1)                       |
| 4                  | -> Hash (cost=246.41..246.41 rows=136 width=8) (actual time=0.144..0.146 rows=133 loops=1)   |
| 5                  | Buckets: 1024 Batches: 1 Memory Usage: 14kB  |
| 6                  | -> Bitmap Heap Scan on orders_order o (cost=5.68..246.41 rows=136 width=8) (actual time=0.049..0.134 rows=133 loops=1)                       |
| 7                  | Recheck Cond: ((created_at >= '2025-03-01 00:00:00+07':timestamp with time zone) AND (created_at < '2025-04-01 00:00:00+07':timestamp wit... |
| 8                  | Filter: is_ordered   |
| 9                  | Heap Blocks: exact=107   |
| 10                 | -> Bitmap Index Scan on idx_orders_created_at (cost=0.00..5.65 rows=136 width=0) (actual time=0.036..0.036 rows=133 loops=1)                 |
| 11                 | Index Cond: ((created_at >= '2025-03-01 00:00:00+07':timestamp with time zone) AND (created_at < '2025-04-01 00:00:00+07':timestamp wit...   |
| 12                 | Planning Time: 2.257 ms  |
| 13                 | Execution Time: 2.861 ms   |

**Nhận xét :** Ta có thể thấy cost giảm từ 883.64 → 698.77

### Truy vấn 3: Tự động cập nhật trạng thái của mã giảm giá

Mô tả : Trigger sẽ tự động cập nhật trạng thái hoạt động is\_active và thời gian cập nhật updated\_at của các chương trình khuyến mại trong bảng promotion

```
CREATE OR REPLACE FUNCTION update_promotion_status()
RETURNS TRIGGER AS $$
BEGIN

    IF NEW.end_date < CURRENT_TIMESTAMP THEN
        NEW.is_active = false;
    ELSIF NEW.start_date <= CURRENT_TIMESTAMP AND NEW.end_date >= CURRENT_TIMESTAMP THEN
        NEW.is_active = true;
    END IF;

    NEW.updated_at = CURRENT_TIMESTAMP;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_auto_update_promotion_status
BEFORE INSERT OR UPDATE ON promotion
FOR EACH ROW
EXECUTE FUNCTION update_promotion_status();
```

#### Kết quả:

UPDATE PROMOTION SET end\_date = '2025-12-16' WHERE name = 'Promo 2';
select \* from promotion where name = 'Promo 2';

| promotion_id | name    | quantity | discount_code | discount_value | start_date             | end_date               | is_active | created_at             | updated_at                    |
|--------------|---------|----------|---------------|----------------|------------------------|------------------------|-----------|------------------------|-------------------------------|
| 2            | Promo 2 | 100      | CODE2         | 25.00          | 2023-04-17 07:54:04+07 | 2025-12-16 00:00:00+07 | true      | 2022-01-25 14:11:58+07 | 2025-06-24 12:38:48.308678+07 |

#### Giải thích :

Trigger sẽ kiểm tra nếu như ngày kết thúc (end\_date) có nằm trước thời gian hiện tại hay không, nếu có thì đặt is\_active thành false. Ngược lại, nếu ngày bắt đầu (start\_date) nằm trước hiện tại và ngày kết thúc (end\_date) nằm sau hiện tại thì sẽ đặt is\_active thành true. Đồng thời cũng cập nhật luôn thời gian updated thành thời gian hiện tại. Trigger sẽ được kích hoạt trước mỗi lần INSERT hay UPDATE vào bảng promotion

## Truy vấn 4 : Xem lịch sử các đơn hàng của khách hàng

Mô tả : Function này được tạo ra để có thể tra cứu thông tin lịch sử đơn hàng của 1 người dùng cụ thể, bao gồm các thông tin chi tiết về đơn hàng

```
CREATE OR REPLACE FUNCTION get_order_history(userid bigint)
RETURNS TABLE (order_id bigint, order_number varchar(100), customer_name text, order_date timestamp with time zone,
order_total numeric(10,2), order_status varchar(100), payment_status varchar(100), promotion_used varchar(100),
promotion_discount numeric(5,2))
) AS $$
BEGIN
    RETURN QUERY
    SELECT
        o.id AS order_id, o.order_number, CONCAT(a.first_name, ' ', a.last_name) AS customer_name,
        o.created_at AS order_date, o.order_total, o.order_status, p.status AS payment_status,
        COALESCE(pr.name, 'No apply') AS promotion_used, COALESCE(pr.discount_value, 0) AS promotion_discount
    FROM orders_order o
    JOIN accounts_account a ON o.user_id = a.id
    LEFT JOIN orders_payment p ON o.payment_id = p.id
    LEFT JOIN promotion pr ON o.promotion_used_id = pr.promotion_id
    WHERE o.user_id = userid
    ORDER BY o.created_at DESC;
END;
$$ LANGUAGE plpgsql;
```

### Kết quả:

73 select \* from get\_order\_history(4);

Data Output Messages Notifications

Showing rows: 1 to 4

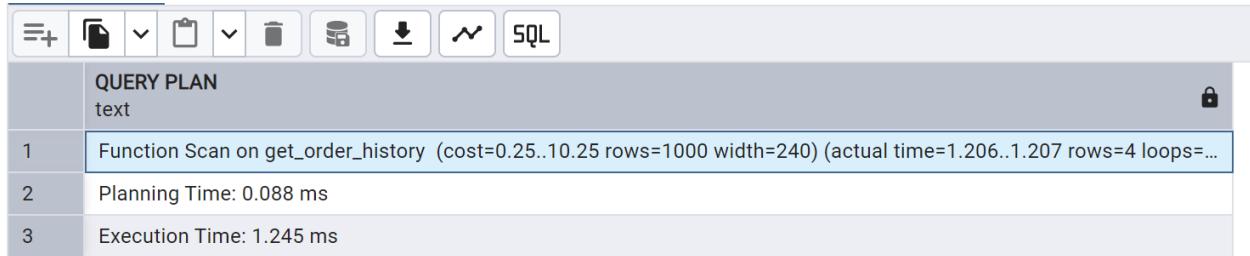
|   | order_id | order_number | customer_name | order_date             | order_total | order_status | payment_status | promotion_used | promotion_discount |
|---|----------|--------------|---------------|------------------------|-------------|--------------|----------------|----------------|--------------------|
| 1 | 8771     | ORD008771    | Taylor King   | 2024-02-19 05:24:05+07 | 200.96      | Shipping     | Failed         | Promo 235      | 6.00               |
| 2 | 8587     | ORD008587    | Taylor King   | 2022-02-23 22:38:09+07 | 509.79      | Shipping     | Pending        | No apply       | 0                  |
| 3 | 6875     | ORD006875    | Taylor King   | 2020-12-28 03:10:40+07 | 486.34      | Shipping     | Failed         | Promo 82       | 8.00               |
| 4 | 3731     | ORD003731    | Taylor King   | 2020-07-06 03:07:18+07 | 335.60      | Pending      | Paid           | No apply       | 0                  |

**Giải thích :** Hàm nhận tham số đầu vào là mã khách hàng(userid), trả về 1 bảng với các cột là thông tin của đơn hàng. Từ bảng orders\_order ta sẽ JOIN với bảng accounts\_account để lấy thông tin khác hàng. Ta sử dụng LEFT JOIN với bảng orders\_payment để lấy trạng thái thanh toán (payment\_status), vì 1 số đơn hàng chưa có thông tin thanh toán. Tương tự, LEFT JOIN với bảng promotion để lấy thông tin khuyến mại, và không phải đơn hàng nào cũng áp dụng khuyến mại. Sử dụng COALESCE để thay thế giá trị NULL của promotion\_used bằng 'No Apply' và promotion\_discount bằng 0

### Phân tích truy vấn

```
EXPLAIN ANALYZE
select * from get_order_history(4);
```

## Kết quả:



The screenshot shows a PostgreSQL query plan viewer. The top bar has icons for file operations and a 'SQL' button. Below is a table with three rows:

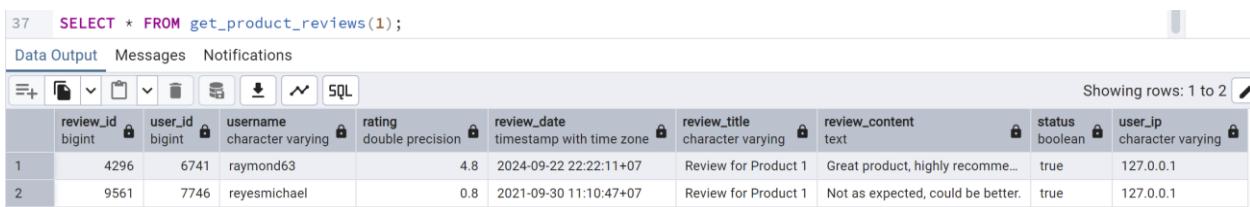
|   | QUERY PLAN  |
|---|---|
| 1 | Function Scan on get_order_history (cost=0.25..10.25 rows=1000 width=240) (actual time=1.206..1.207 rows=4 loops=...) |
| 2 | Planning Time: 0.088 ms   |
| 3 | Execution Time: 1.245 ms  |

## Truy vấn 5 : Xem các đánh giá của sản phẩm bất kì

Mô tả : Function này tạo ra để đưa ra những đánh giá hiện có của 1 sản phẩm bất kì

```
CREATE OR REPLACE FUNCTION get_product_reviews(product_id_param bigint)
RETURNS TABLE (review_id bigint, user_id bigint, username varchar(50), rating double precision,
               review_date timestamp with time zone, review_title varchar(100), review_content text, status boolean
) AS $$
BEGIN
    RETURN QUERY
    SELECT
        r.id AS review_id, u.id AS user_id,
        u.username, r.rating, r.created_at AS review_date,
        r.subject AS review_title, r.review AS review_content,
        r.status
    FROM reviews_review r
    JOIN accounts_account u ON r.user_id = u.id
    WHERE
        r.product_id = product_id_param
        AND r.status = true
    ORDER BY r.created_at DESC;
END;
$$ LANGUAGE plpgsql;
```

## Kết quả:



The screenshot shows a PostgreSQL query results viewer. The top bar has icons for file operations and a 'SQL' button. Below is a table with two rows of data:

|   | review_id | user_id | username     | rating | review_date            | review_title         | review_content                    | status | user_ip   |
|---|-----------|---------|--------------|--------|------------------------|----------------------|-----------------------------------|--------|-----------|
| 1 | 4296      | 6741    | raymond63    | 4.8    | 2024-09-22 22:22:11+07 | Review for Product 1 | Great product, highly recomme...  | true   | 127.0.0.1 |
| 2 | 9561      | 7746    | reyesmichael | 0.8    | 2021-09-30 11:10:47+07 | Review for Product 1 | Not as expected, could be better. | true   | 127.0.0.1 |

## Giải thích :

Hàm nhận tham số đầu vào là mã của sản phẩm, trả về bảng chứa các thông tin đánh giá của sản phẩm đó. Ta JOIN bảng reviews\_review với bảng accounts\_account để lấy thông tin của người dùng, và đảm bảo rằng chỉ lấy những đánh giá của người dùng hợp lệ (user\_id tồn tại trong accounts\_account). Ở điều kiện lọc thì chỉ lọc những sản phẩm có mã trùng với mã sản phẩm đầu vào và đang có trạng thái hiển thị status = true (Loại bỏ trường hợp đánh giá không hợp lệ, bị quản lý, nhân viên set status thành false )

## Phân tích truy vấn

```
EXPLAIN ANALYZE
SELECT
    r.id AS review_id, u.id AS user_id,
    u.username, r.rating, r.created_at AS review_date,
    r.subject AS review_title, r.review AS review_content,
    r.status
FROM reviews_review r
JOIN accounts_account u ON r.user_id = u.id
WHERE
    r.product_id = 1
    AND r.status = true
ORDER BY r.created_at DESC;
```

### Khi chạy không có index

|    | QUERY PLAN<br>text  |
|----|---|
| 1  | Sort (cost=324.85..324.86 rows=1 width=99) (actual time=1.218..1.219 rows=2 loops=1)  |
| 2  | Sort Key: r.created_at DESC   |
| 3  | Sort Method: quicksort Memory: 25kB   |
| 4  | -> Nested Loop (cost=0.29..324.84 rows=1 width=99) (actual time=0.542..1.210 rows=2 loops=1)  |
| 5  | -> Seq Scan on reviews_review r (cost=0.00..316.54 rows=1 width=89) (actual time=0.532..1.192 rows=2 loops=1)                               |
| 6  | Filter: (status AND (product_id = 1))   |
| 7  | Rows Removed by Filter: 8921  |
| 8  | -> Index Scan using accounts_account_pkey on accounts_account u (cost=0.29..8.30 rows=1 width=18) (actual time=0.006..0.007 rows=1 loops=1) |
| 9  | Index Cond: (id = r.user_id)  |
| 10 | Planning Time: 1.780 ms   |
| 11 | Execution Time: 1.256 ms  |

### Khi chạy với index

```
CREATE INDEX idx_reviews_product_id ON reviews_review(product_id);
```

|    | QUERY PLAN<br>text  |
|----|---|
| 1  | Sort (cost=16.62..16.62 rows=1 width=99) (actual time=0.045..0.045 rows=2 loops=1)  |
| 2  | Sort Key: r.created_at DESC   |
| 3  | Sort Method: quicksort Memory: 25kB   |
| 4  | -> Nested Loop (cost=0.57..16.61 rows=1 width=99) (actual time=0.033..0.039 rows=2 loops=1)   |
| 5  | -> Index Scan using idx_reviews_product_id on reviews_review r (cost=0.29..8.30 rows=1 width=89) (actual time=0.026..0.028 rows=2 loops=1)  |
| 6  | Index Cond: (product_id = 1)  |
| 7  | Filter: status  |
| 8  | -> Index Scan using accounts_account_pkey on accounts_account u (cost=0.29..8.30 rows=1 width=18) (actual time=0.004..0.004 rows=1 loops=1) |
| 9  | Index Cond: (id = r.user_id)  |
| 10 | Planning Time: 2.327 ms   |
| 11 | Execution Time: 0.071 ms  |

**Nhận xét :** cost giảm đáng kể từ 324.86 → 16.62

**Lí do sử dụng truy vấn :** Bởi trong hàm đang sử dụng điều kiện WHERE r.product\_id = product\_id\_param để lọc 1 sản phẩm cụ thể. Nếu như không có index, hệ thống dùng Seq Scan để quét lần lượt → Rất tốn chi phí bởi có đến hàng nghìn bản ghi. Với index này, hệ thống dùng Index Scan nhảy tới hàng có product\_id phù hợp, giảm đáng kể chi phí.

## Truy vấn 6 : Tự động cập nhật số lượng trong kho của sản phẩm

Mô tả : Trigger sẽ tự động cập nhật stock của sản phẩm dựa trên số lượng sản phẩm đã bán ra

```
CREATE OR REPLACE FUNCTION update_product_stock()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.ordered = true THEN
        UPDATE store_product
        SET stock = stock - NEW.quantity,
            updated_at = NOW(),
            is_available = (stock - NEW.quantity) > 0
        WHERE id = NEW.product_id;

        IF (SELECT stock FROM store_product WHERE id = NEW.product_id) < 0 THEN
            RAISE EXCEPTION 'Out of stock %', NEW.product_id;
        END IF;
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_update_product_stock
AFTER INSERT OR UPDATE ON orders_orderproduct
FOR EACH ROW
EXECUTE FUNCTION update_product_stock();
```

**Giải thích :** Ban đầu, hàm kiểm tra xem nếu sản phẩm trong đơn hàng đã được đánh dấu là đặt thì sẽ thực hiện việc cập nhật trên bảng store\_product với các việc như giảm stock đúng bằng số lượng đặt hàng (NEW.quantity), cập nhật lại thời gian update\_at bằng thời gian hiện tại và cập nhật is\_available thành true nếu như stock sau khi trừ vẫn lớn hơn 0. False nếu hết hàng. Bên cạnh đó còn kiểm tra nếu như stock mà nhỏ hơn 0 thì sẽ raise thông báo. Trigger sẽ được kích hoạt **sau khi** có INSERT hay UPDATE trên bảng orders\_orderproduct.

## Truy vấn 7 : Xem doanh thu trong 6 tháng gần nhất

Mô tả: View này tổng hợp doanh thu và hiệu suất bán hàng trong 6 tháng gần nhất, giúp quản lý dễ dàng phân tích xu hướng kinh doanh và điều chỉnh nếu không phù hợp chiến lược.

```
CREATE OR REPLACE VIEW report_6months_summary AS
SELECT
    TO_CHAR(DATE_TRUNC('month', o.created_at), 'YYYY-MM') AS month,
    COUNT(DISTINCT o.id) AS total_orders,
    COUNT(DISTINCT o.user_id) AS unique_customers,
    ROUND(SUM(o.order_total)::numeric, 2) AS gross_revenue,
    ROUND(SUM(o.order_total - o.order_discount)::numeric, 2) AS net_revenue,
    ROUND((SUM(o.order_total - o.order_discount) / COUNT(DISTINCT o.id))::numeric, 2) AS avg_order_value,
    ROUND((SUM(o.order_discount) * 100.0 / NULLIF(SUM(o.order_total), 0))::numeric, 2) AS discount_percentage
FROM orders_order o
WHERE
    o.is_ordered = true
    AND o.created_at >= NOW() - INTERVAL '6 months'
GROUP BY DATE_TRUNC('month', o.created_at)
ORDER BY month DESC;
```

## Kết quả :

| 21 select * from report_6months_summary; |            |                     |                         |                       |                     |                         |                             |  |
|--|------------|---------------------|-------------------------|-----------------------|---------------------|-------------------------|-----------------------------|--|
| Data Output Messages Notifications       |            |                     |                         |                       |                     |                         |                             |  |
|  | month text | total_orders bigint | unique_customers bigint | gross_revenue numeric | net_revenue numeric | avg_order_value numeric | discount_percentage numeric |  |
| 1  | 2025-06    | 117                 | 116                     | 41342.45              | 38074.68            | 325.42                  | 7.90                        |  |
| 2  | 2025-05    | 142                 | 142                     | 49401.80              | 45391.97            | 319.66                  | 8.12                        |  |
| 3  | 2025-04    | 141                 | 140                     | 51554.85              | 47631.81            | 337.81                  | 7.61                        |  |
| 4  | 2025-03    | 133                 | 131                     | 46983.75              | 43578.65            | 327.66                  | 7.25                        |  |
| 5  | 2025-02    | 109                 | 108                     | 36964.61              | 33865.60            | 310.69                  | 8.38                        |  |
| 6  | 2025-01    | 139                 | 139                     | 48274.79              | 44278.61            | 318.55                  | 8.28                        |  |
| 7  | 2024-12    | 34                  | 34                      | 12272.68              | 11403.86            | 335.41                  | 7.08                        |  |

## Giải thích :

- Ta chỉ lấy những đơn hàng đã được xác nhận(is\_ordered=true) trong 6 tháng gần nhất(NOW() – INTERVAL '6 months' ), tất cả các đơn hàng được gom chung về cùng tháng để dễ so sánh qua DATE\_TRUNC, cùng với tính các chỉ số liên quan như số lượng đơn hàng trong tháng (total\_orders), số lượng khách hàng khác nhau đã mua ( unique\_customers), tổng tiền thu về trước khi giảm giá (gross\_revenue), tổng tiền thực sau khi trừ mã giảm giá (net\_revenue), giá trị trung bình mỗi đơn hàng (avg\_order\_value = doanh thu thực / số đơn hàng ), tỷ lệ giảm giá trung bình (discount\_percentage). Ta dùng ROUND(...,2) để làm tròn về 2 chữ số thập phân và NULLIF để tránh chia cho 0. Sau đó nhóm kết quả theo tháng để tổng hợp.

## Phân tích hiệu năng:

**EXPLAIN ANALYZE**

```
select * from report_6months_summary;
```

|    | QUERY PLAN   |
|----|--|
| 1  | text   |
| 1  | Subquery Scan on report_6months_summary (cost=460.94..471.14 rows=816 width=176) (actual time=1.855..1.857 rows=7 loops=1) |
| 2  | -> Sort (cost=460.94..462.98 rows=816 width=184) (actual time=1.854..1.855 rows=7 loops=1)                                 |
| 3  | Sort Key: (to_char((date_trunc('month)::text, o.created_at), 'YYYY-MM)::text)) DESC  |
| 4  | Sort Method: quicksort Memory: 25kB  |
| 5  | -> GroupAggregate (cost=366.40..421.48 rows=816 width=184) (actual time=1.429..1.810 rows=7 loops=1)                       |
| 6  | Group Key: (date_trunc('month)::text, o.created_at)  |
| 7  | -> Sort (cost=366.40..368.44 rows=816 width=44) (actual time=1.373..1.409 rows=815 loops=1)                                |
| 8  | Sort Key: (date_trunc('month)::text, o.created_at), o.id   |
| 9  | Sort Method: quicksort Memory: 75kB  |
| 10 | -> Bitmap Heap Scan on orders_order o (cost=18.61..326.93 rows=816 width=44) (actual time=0.217..1.000 rows=81...          |
| 11 | Recheck Cond: (created_at >= (now() - '6 mons'::interval))   |
| 12 | Filter: is_ordered   |
| 13 | Heap Blocks: exact=278   |
| 14 | -> Bitmap Index Scan on idx_orders_created_at (cost=0.00..18.41 rows=816 width=0) (actual time=0.160..0.160 r...           |
| 15 | Index Cond: (created_at >= (now() - '6 mons'::interval))   |
| 16 | Planning Time: 0.935 ms  |

Truy vấn sử dụng index idx\_orders\_created\_at ( đã tạo ở Truy vấn 2 ).

## Truy vấn 8 : Lọc sản phẩm theo các biến thể và danh mục

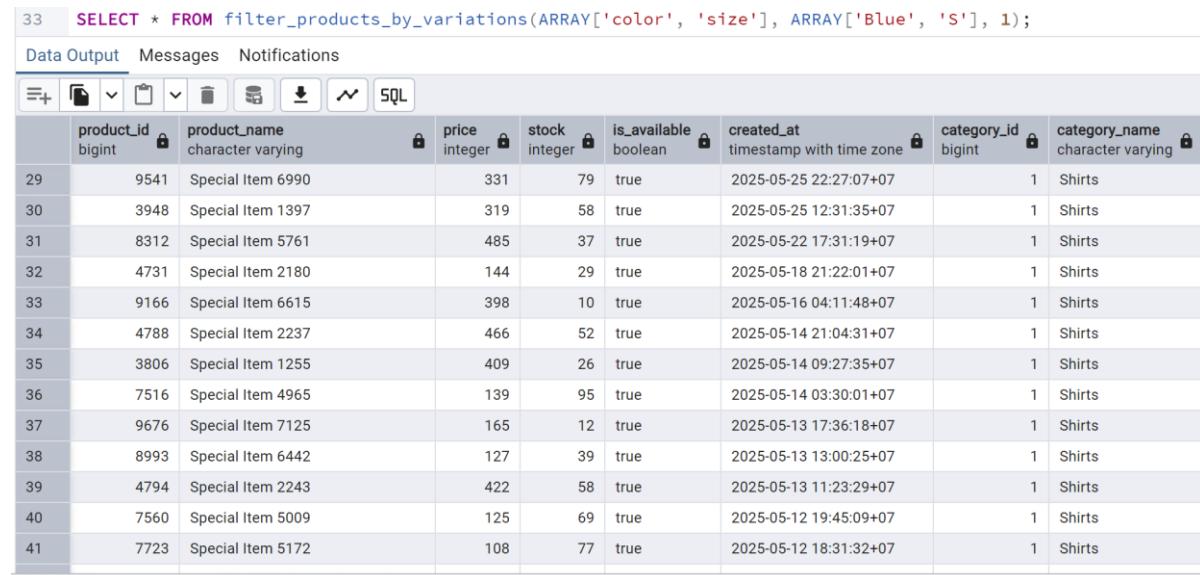
Mô tả : Function này được thiết kế để lọc sản phẩm theo các biến thể (color,size) và danh mục các sản phẩm

```
-- Cau8
CREATE OR REPLACE FUNCTION filter_products_by_variations(
    p_variation_categories varchar(100)[],
    p_variation_values varchar(100)[],
    p_category_id bigint
)
RETURNS TABLE (
    product_id bigint, product_name varchar(200), price integer,
    stock integer, is_available boolean, created_at timestamp with time zone,
    category_id bigint, category_name varchar(200)
) AS $$|
BEGIN
    RETURN QUERY
    WITH filtered_variations AS (
        SELECT v.product_id
        FROM store_variation v
        WHERE (p_variation_categories IS NULL OR v.variation_category = ANY(p_variation_categories))
        AND (p_variation_values IS NULL OR v.variation_value = ANY(p_variation_values))
        GROUP BY v.product_id
        HAVING COUNT(*) = COALESCE(array_length(p_variation_categories, 1), 0)
    )
    SELECT
        p.id AS product_id, p.name, p.price, p.stock, p.is_available, p.created_at,
        c.id AS category_id, c.name AS category_name
    FROM store_product p
    INNER JOIN filtered_variations fv ON p.id = fv.product_id
    LEFT JOIN store_category c ON p.category_id = c.id
    WHERE p.is_available = true
    AND (p_category_id IS NULL OR p.category_id = p_category_id)
    ORDER BY p.created_at DESC;
END;
$$ LANGUAGE plpgsql;
```

Kết quả :

33    `SELECT * FROM filter_products_by_variations(ARRAY['color', 'size'], ARRAY['Blue', 'S'], 1);`

Data Output    Messages    Notifications



|    | product_id | product_name      | price | stock | is_available | created_at             | category_id | category_name |
|----|------------|-------------------|-------|-------|--------------|------------------------|-------------|---------------|
| 29 | 9541       | Special Item 6990 | 331   | 79    | true         | 2025-05-25 22:27:07+07 | 1           | Shirts        |
| 30 | 3948       | Special Item 1397 | 319   | 58    | true         | 2025-05-25 12:31:35+07 | 1           | Shirts        |
| 31 | 8312       | Special Item 5761 | 485   | 37    | true         | 2025-05-22 17:31:19+07 | 1           | Shirts        |
| 32 | 4731       | Special Item 2180 | 144   | 29    | true         | 2025-05-18 21:22:01+07 | 1           | Shirts        |
| 33 | 9166       | Special Item 6615 | 398   | 10    | true         | 2025-05-16 04:11:48+07 | 1           | Shirts        |
| 34 | 4788       | Special Item 2237 | 466   | 52    | true         | 2025-05-14 21:04:31+07 | 1           | Shirts        |
| 35 | 3806       | Special Item 1255 | 409   | 26    | true         | 2025-05-14 09:27:35+07 | 1           | Shirts        |
| 36 | 7516       | Special Item 4965 | 139   | 95    | true         | 2025-05-14 03:30:01+07 | 1           | Shirts        |
| 37 | 9676       | Special Item 7125 | 165   | 12    | true         | 2025-05-13 17:36:18+07 | 1           | Shirts        |
| 38 | 8993       | Special Item 6442 | 127   | 39    | true         | 2025-05-13 13:00:25+07 | 1           | Shirts        |
| 39 | 4794       | Special Item 2243 | 422   | 58    | true         | 2025-05-13 11:23:29+07 | 1           | Shirts        |
| 40 | 7560       | Special Item 5009 | 125   | 69    | true         | 2025-05-12 19:45:09+07 | 1           | Shirts        |
| 41 | 7723       | Special Item 5172 | 108   | 77    | true         | 2025-05-12 18:31:32+07 | 1           | Shirts        |

### Giải thích :

- Hàm nhận tham số đầu vào là mảng các loại biến thể( ['color','size'] ), mảng các biến thể tương ứng ( ['Red','L'] ), ID của danh mục sản phẩm và trả về thông tin các sản phẩm thỏa mãn điều kiện. Hàm sử dụng 1 CTE (filtered\_variations) để lọc sản phẩm dựa trên tiêu chí, rồi dựa vào đó để truy vấn tiếp.
- Cụ thể thì CTE lọc các biến thể có variation\_category nằm trong mảng đầu vào ban đầu, nếu như bảng NULL thì bỏ qua. Cách lọc các biến thể trong bảng p\_variation\_categories tương tự như vậy. Ta nhóm theo product\_id để đảm bảo mỗi sản phẩm xuất hiện 1 lần.
- HAVING COUNT(\*) = COALESCE(array\_length(p\_variation\_categories, 1), 0) đảm bảo rằng số lượng biến thể phải khớp với số lượng biến thể đã cung cấp, xử lý luôn cả trường hợp p\_variation\_categories là NULL thì array\_length trả về NULL, COALESCE cũng thành 0 → Không yêu cầu biến thể nào ( Ví dụ cụ thể như cung cấp mảng đầu vào ['color','size'], thì sản phẩm phải có đúng 2 biến thể khớp thì mới thỏa mãn )

### Truy vấn 9 : Xem thông tin chi tiết của 1 sản phẩm cụ thể

Mô tả : Function được tạo ra với mục đích xem thông tin chi tiết của sản phẩm bao gồm tên, mô tả, danh mục, giá, sao đánh giá

```
CREATE OR REPLACE FUNCTION get_product_details(product_id bigint)
RETURNS TABLE (product_name varchar(200), description text, price integer,
               stock integer, star_rating numeric(5,1), category_name varchar(200)
) AS $$
BEGIN
  RETURN QUERY
  SELECT p.name AS product_name, p.description, p.price, p.stock, p.star_rated , c.name AS category_name
  FROM store_product p
  JOIN store_category c ON p.category_id = c.id
  WHERE p.id = product_id AND p.is_available = true;
END;
$$ LANGUAGE plpgsql;
```

### Kết quả :



| 15 SELECT * FROM get_product_details(123); |  |  |                         |                        |                               |  |
|--|--|--|-------------------------|------------------------|-------------------------------|--|
| Data Output                                |  | Messages   |                         | Notifications          |                               |  |
|  |  |  |                         |                        |                               |  |
| 1  | product_name<br>character varying<br>Grey Oxford Shirt | description<br>text<br>Mr agreement dark out charge fall. Else radio task suggest article lose soon. | price<br>integer<br>146 | stock<br>integer<br>52 | star_rating<br>numeric<br>4.9 | category_name<br>character varying<br>Shirts |

## Phân tích truy vấn bên trong

```
EXPLAIN ANALYZE
SELECT p.name AS product_name, p.description, p.price, p.stock, p.star_rated, c.name AS category_name
FROM store_product p
JOIN store_category c ON p.category_id = c.id
WHERE p.id = 123 AND p.is_available = true;
```

|   | QUERY PLAN  | text | 🔒 |
|---|---|------|---|
| 1 | Nested Loop (cost=0.43..16.66 rows=1 width=523) (actual time=0.017..0.018 rows=1 loops=1)   |      |   |
| 2 | -> Index Scan using store_produ_id_2abda1_idx on store_product p (cost=0.29..8.30 rows=1 width=113) (actual time=0.009..0.009 rows=1 loops=1) |      |   |
| 3 | Index Cond: (id = 123)  |      |   |
| 4 | Filter: is_available  |      |   |
| 5 | -> Index Scan using store_category_pkey on store_category c (cost=0.14..8.16 rows=1 width=426) (actual time=0.006..0.006 rows=1 loops=1)      |      |   |
| 6 | Index Cond: (id = p.category_id)  |      |   |
| 7 | Planning Time: 0.926 ms   |      |   |
| 8 | Execution Time: 0.041 ms  |      |   |

### Giải thích :

Dù chưa tạo chỉ mục nào nhưng ta cũng thấy hệ thống sử dụng những index sẵn có để quét(store\_produ\_id\_2abda1\_idx, store\_category\_pkey), những index được hệ thống tạo ra bởi chúng chính là những khóa chính của bảng.

## Truy vấn 10 : Xem các đơn hàng của người dùng có trạng thái thanh toán thất bại

Mô tả : Hàm này được tạo ra để có thể lấy ra danh sách các đơn hàng có trạng thái thanh toán thất bại của 1 người dùng

```
CREATE OR REPLACE FUNCTION get_user_failed_orders(p_user_id bigint)
RETURNS TABLE (order_id bigint, order_number varchar(100), order_total numeric(10,2), payment_method varchar(100), payment_status varchar(100),
               payment_amount numeric(10,2), order_date timestamp with time zone, customer_name text, customer_phone varchar(15))
AS $$
BEGIN
  RETURN QUERY
  SELECT o.id AS order_id, o.order_number, o.order_total, p.payment_method, p.status , p.amount_paid ,
         o.created_at AS order_date, CONCAT(o.first_name, ' ', o.last_name) AS customer_name, o.phone
  FROM orders_order o
  JOIN orders_payment p ON o.payment_id = p.id
  WHERE
    p.status = 'Failed' AND o.is_ordered = true AND o.user_id = p_user_id;
END;
$$ LANGUAGE plpgsql;
```

## Kết quả:

| 15 | SELECT * FROM get_user_failed_orders(45); |
|----|---|
|    | Data Output Messages Notifications        |
|    | SQL                                       |

Showing rows: 1 to 1

|   | order_id | order_number | order_total | payment_method | payment_status | payment_amount | order_date             | customer_name | customer_phone |
|---|----------|--------------|-------------|----------------|----------------|----------------|------------------------|---------------|----------------|
| 1 | 6335     | ORD006335    | 125.97      | VNPay          | Failed         | 138.70         | 2022-12-16 11:49:28+07 | James Payne   | 0224269283     |

## Phân tích truy vấn bên trong

```
EXPLAIN ANALYZE
SELECT o.id AS order_id, o.order_number, o.order_total, p.payment_method, p.status, p.amount_paid,
o.created_at AS order_date, CONCAT(o.first_name, ' ', o.last_name) AS customer_name, o.phone AS
FROM orders_order o
JOIN orders_payment p ON o.payment_id = p.id
WHERE
    p.status = 'Failed' AND
    o.is_ordered = true AND
    o.user_id = 123;
```

## Kết quả :

|    | QUERY PLAN  |
|----|---|
| 1  | text  |
| 1  | Nested Loop (cost=4.59..28.45 rows=1 width=91) (actual time=0.066..0.067 rows=1 loops=1)  |
| 2  | -> Bitmap Heap Scan on orders_order o (cost=4.30..11.83 rows=2 width=65) (actual time=0.039..0.043 rows=2 loops=1)                      |
| 3  | Recheck Cond: (user_id = 123)   |
| 4  | Filter: is_ordered  |
| 5  | Heap Blocks: exact=2  |
| 6  | -> Bitmap Index Scan on orders_order_user_id_e9b59eb1 (cost=0.00..4.30 rows=2 width=0) (actual time=0.013..0.014 rows=2 loops=1)        |
| 7  | Index Cond: (user_id = 123)   |
| 8  | -> Index Scan using orders_payment_pkey on orders_payment p (cost=0.29..8.30 rows=1 width=24) (actual time=0.008..0.008 rows=0 loops=1) |
| 9  | Index Cond: (id = o.payment_id)   |
| 10 | Filter: ((status)::text = 'Failed'::text)   |
| 11 | Rows Removed by Filter: 0   |
| 12 | Planning Time: 0.537 ms   |
| 13 | Execution Time: 0.117 ms  |

### 5.3. 10 câu truy vấn của Trần Đăng Sinh

#### Câu 1: Trigger kiểm tra và cảnh báo khi khách hàng có số điện thoại bị trùng

- Đảm bảo rằng mỗi số điện thoại trong bảng accounts\_userphone là duy nhất trên toàn hệ thống, qua đó ngăn chặn việc gán cùng một số điện thoại cho nhiều tài khoản khác nhau — tránh nhầm lẫn trong liên hệ, xác thực hoặc xử lý đơn hàng.
- Nếu số điện thoại NEW.phone\_number tồn tại trong bảng nhưng thuộc bản ghi khác=> raise trigger và từ chối thao tác , nếu chưa tồn tại thì cho phép thực hiện

```
CREATE OR REPLACE FUNCTION trg_check_duplicate_phone()
RETURNS TRIGGER AS $$

BEGIN
    IF EXISTS (
        SELECT 1
        FROM accounts_account
        WHERE phone_number = NEW.phone_number
        AND id <> NEW.id
    ) THEN
        RAISE EXCEPTION 'Số điện thoại % đã tồn tại.', NEW.phone_number;
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER before_insert_or_update_phone
BEFORE INSERT OR UPDATE ON accounts_account
FOR EACH ROW
EXECUTE FUNCTION trg_check_duplicate_phone();
```

Kết quả :

```

hustshoppe=# INSERT INTO accounts_account (
hustshoppe(#     id, password, is_superuser, email, username,
hustshoppe(#     first_name, last_name, role, dob, gender, address,
hustshoppe(#     date_joined, last_login, is_active, phone_number, is_staff
hustshoppe(# )
hustshoppe-# VALUES (
hustshoppe(#     10010 , '123456', false, 'user@example.com', 'user123',
hustshoppe(#     'Nguyen', 'Van A', 2, '1990-01-01', 'M', 'Hanoi',
hustshoppe(#     NOW(), NOW(), true, '0123456789', false
hustshoppe(# );
hustshoppe-# );
INSERT 0 1
hustshoppe=# INSERT INTO accounts_account (
hustshoppe(#     id, password, is_superuser, email, username,
hustshoppe(#     first_name, last_name, role, dob, gender, address,
hustshoppe(#     date_joined, last_login, is_active, phone_number, is_staff
hustshoppe(# )
hustshoppe-# VALUES (
hustshoppe(#     10011 , '123456', false, 'user@example.com', 'user123',
hustshoppe(#     'Nguyen', 'Van B', 2, '2005-12-23', 'M', 'Hanoi',
hustshoppe(#     NOW(), NOW(), true, '0123456789', false
hustshoppe(# );
ERROR:  S? di?n tho?i 0123456789 da t?n t?i.
CONTEXT:  PL/pgSQL function trg_check_duplicate_phone() line 9 at RAISE
hustshoppe=#

```

## Câu 2 : Cập nhật trạng thái tự động dựa trên thời gian

- Tự động cập nhật trạng thái (is\_active) của một mã khuyến mãi trong bảng public.promotion dựa trên thời gian hiện tại (CURRENT\_TIMESTAMP) so với các cột start\_date và end\_date

```

CREATE OR REPLACE FUNCTION update_promotion_status()
RETURNS TRIGGER AS $$$
BEGIN
    IF NEW.end_date < CURRENT_TIMESTAMP THEN
        NEW.is_active = false;
    ELSIF NEW.start_date <= CURRENT_TIMESTAMP AND NEW.end_date >= CURRENT_TIMESTAMP THEN
        NEW.is_active = true;
    END IF;
    NEW.updated_at = CURRENT_TIMESTAMP;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER update_promotion_status
BEFORE INSERT OR UPDATE ON public.promotion
FOR EACH ROW
EXECUTE FUNCTION update_promotion_status();

```

## Kết quả:

```

hustshop=# SELECT promotion_id, name, start_date, end_date, is_active, updated_at
hustshop=# FROM public.promotion
hustshop=# WHERE promotion_id = 4;
+-----+-----+-----+-----+-----+-----+
| promotion_id | name | start_date | end_date | is_active | updated_at |
+-----+-----+-----+-----+-----+-----+
| 4 | Khuy?n mai cu | 2025-06-18 03:00:00+07 | 2025-06-18 04:00:00+07 | f | 2025-06-19 04:53:50.71496+07
+-----+-----+-----+-----+-----+-----+
(1 row)

```

```

hustshop=# UPDATE public.promotion
hustshop-# SET end_date = '2025-06-28 04:00:00+07'
hustshop-# WHERE promotion_id = 4;
UPDATE 1
hustshop=# SELECT promotion_id, name, start_date, end_date, is_active, updated_at
hustshop=# FROM public.promotion
hustshop=# WHERE promotion_id = 4;
+-----+-----+-----+-----+-----+-----+
| promotion_id | name | start_date | end_date | is_active | updated_at |
+-----+-----+-----+-----+-----+-----+
| 4 | Khuy?n mai cu | 2025-06-18 03:00:00+07 | 2025-06-28 04:00:00+07 | t | 2025-06-19 04:56:06.813487+07
+-----+-----+-----+-----+-----+-----+
(1 row)

```

### Câu 3: Đánh giá trung bình của sản phẩm

- Cách 1: Gộp dữ liệu trước, JOIN sau

```

SELECT
    p.id AS product_id,
    p.name AS product_name,
    rp.average_rating,
    rp.total_reviews
FROM (
    SELECT
        product_id,
        ROUND(AVG(rating)::numeric, 2) AS average_rating,
        COUNT(*) AS total_reviews
    FROM reviews_review
    GROUP BY product_id
) rp
JOIN store_product p ON p.id = rp.product_id
ORDER BY rp.average_rating DESC;

```

- Cách 2: JOIN trước toàn bộ rồi tổng hợp

```

SELECT
    p.id AS product_id,
    p.name AS product_name,
    ROUND(AVG(r.rating)::numeric, 2) AS average_rating,
    COUNT(r.id) AS total_reviews
FROM reviews_review r
JOIN store_product p ON r.product_id = p.id
GROUP BY p.id, p.name
ORDER BY average_rating DESC;

```

- Phân tích

- 
- Cả hai cách trên có phần khá tương tự nhau, đều thực hiện mục tiêu tính điểm trung bình và số lượt đánh giá cho từng sản phẩm, rồi hiển thị thêm thông tin tên sản phẩm.
  - Tuy nhiên ở cách 1 tách phần thống kê AVG, COUNT thành một bảng tạm rồi sau đó JOIN với bảng store\_product.
  - Còn cách 2 JOIN toàn bộ bảng reviews\_review và store\_product trước, sau đó mới GROUP BY và tổng hợp dữ liệu.
- 
- Cách 1 Tổng hợp trực tiếp trên bảng đánh giá – vốn nhỏ hơn nhiều so với bảng sản phẩm và chỉ JOIN với bảng store\_product sau khi đã rút gọn dữ liệu, giảm số lượng bản ghi trung gian.
    - ⇒ Phù hợp khi bảng store\_product có nhiều cột hoặc dung lượng lớn.
  - Cách 2 JOIN toàn bộ bảng trước có thể tốn thời gian hơn, nhất là khi 1 sản phẩm có nhiều đánh giá rồi mới GROUP BY => có thể tăng thời gian xử lý và tiêu tốn bộ nhớ, dẫn tới kém hiệu quả hơn nếu bảng store\_product lớn

#### Câu 4: Tính tổng tiền cuối cùng của đơn hàng

- Cách 1:

```

CREATE OR REPLACE FUNCTION calculate_order_total(order_id_param BIGINT)
RETURNS NUMERIC(12,2) AS $$

DECLARE
    subtotal NUMERIC(12,2);
    discount_value NUMERIC(5,2) := 0;
    final_total NUMERIC(12,2);

BEGIN

    SELECT COALESCE(SUM(op.subtotal), 0) INTO subtotal
    FROM orders_orderproduct op
    WHERE op.order_id = order_id_param;

    SELECT COALESCE(p.discount_value, 0) INTO discount_value
    FROM orders_order o
    LEFT JOIN promotion p ON o.promotion_used_id = p.promotion_id
    WHERE o.id = order_id_param;

    final_total := subtotal - (subtotal * discount_value / 100);

    RETURN final_total;
END;

```

- Cách 2:

```

CREATE OR REPLACE FUNCTION calculate_order_total(order_id_param BIGINT)
RETURNS NUMERIC(12,2) AS $$

DECLARE
    final_total NUMERIC(12,2);

BEGIN
    SELECT
        COALESCE(SUM(op.subtotal), 0) * (1 - COALESCE(MAX(p.discount_value), 0) / 100.0)
    INTO final_total
    FROM orders_order o
    LEFT JOIN promotion p ON o.promotion_used_id = p.promotion_id
    LEFT JOIN orders_orderproduct op ON o.id = op.order_id
    WHERE o.id = order_id_param;

    RETURN final_total;
END;
$$ LANGUAGE plpgsql;

```

### • Phân tích:

- Cách 1 truy cập vào database 2 lần với 2 lần truy vấn SELECT, lần 1 sử dụng SUM(op.subtotal) và lần 2 dùng promotion.discount\_value, việc tách riêng 2 truy vấn sẽ giúp dễ hiểu hơn khi đọc và viết truy vấn, cũng như dễ kiểm soát hơn nhưng tốn tài nguyên hơn
- Cách 2 chỉ truy cập vào database 1 lần, gộp cả 2 truy vấn của cách 1 vào => giúp giảm số lần truy vấn, sử dụng bộ nhớ hiệu quả hơn

## Câu 5: Doanh thu của 1 nhân viên trong 1 tháng cụ thể của cửa hàng

```
CREATE OR REPLACE FUNCTION hustshop_monthly_earning(
    IN p_staff_id bigint,
    IN p_month char(7) -- format: 'YYYY-MM'
)
RETURNS void AS $$
DECLARE
    total_money numeric := 0;
BEGIN
    SELECT COALESCE(SUM(p.amount_paid), 0)
    INTO total_money
    FROM orders_order o
    JOIN orders_payment p ON p.id = o.payment_id
    WHERE o.staff_id = p.staff_id
    AND TO_CHAR(p.created_at, 'YYYY-MM') = p_month;

    RAISE NOTICE 'Total received earnings for staff % in month %: %', p_staff_id, p_month, total_money;
END;
$$ LANGUAGE plpgsql STABLE;
```

EXPLAIN ANALYZE

```
SELECT COALESCE(SUM(p.amount_paid), 0)
FROM orders_order o
JOIN orders_payment p ON p.id = o.payment_id
WHERE o.staff_id = 123
AND TO_CHAR(p.created_at, 'YYYY-MM') = '2022-01';
```

QUERY PLAN

```
Aggregate (cost=16.95..16.96 rows=1 width=32) (actual time=1.607..1.607 rows=1 loops=1)
  -> Nested Loop (cost=0.57..16.94 rows=1 width=6) (actual time=0.232..1.593 rows=11 loops=1)
      -> Index Scan using orders_order_staff_id_de0718d4 on orders_order o  (cost=0.29..8.30 rows=1 width=8) (actual time=0.008..0.255 rows=363 loops=1)
          Index Cond: (staff_id = 1)
      -> Index Scan using orders_payment_pkey on orders_payment p  (cost=0.29..8.31 rows=1 width=14) (actual time=0.003..0.003 rows=0 loops=363)
          Index Cond: (id = o.payment_id)
          Filter: (to_char(created_at, 'YYYY-MM'::text) = '2022-01'::text)
          Rows Removed by Filter: 1
Planning Time: 1.707 ms
Execution Time: 1.636 ms
(10 rows)
```

**CREATE INDEX idx\_orders\_order\_staff\_id ON orders\_order(staff\_id);**

- **Lý do thêm chỉ mục :** Btree sắp xếp các giá trị staff\_id theo thứ tự tăng dần, cho phép thực hiện tìm kiếm nhị phân để xác định các hàng khớp với p\_staff\_id. Ngoài ra, điều kiện o.staff\_id = p\_staff\_id là một phép so sánh bằng, phù hợp với đặc tính của B-tree, nơi nó hỗ trợ truy cập trực tiếp đến các nút chứa giá trị khớp.

```
QUERY PLAN
Aggregate (cost=16.95..16.96 rows=1 width=32) (actual time=0.024..0.025 rows=1 loops=1)
  -> Nested Loop (cost=0.57..16.94 rows=1 width=6) (actual time=0.022..0.022 rows=0 loops=1)
      -> Index Scan using idx_orders_order_staff_id on orders_order o  (cost=0.29..8.30 rows=1 width=8) (actual time=0.022..0.022 rows=0 loops=1)
          Index Cond: (staff_id = 123)
      -> Index Scan using orders_payment_pkey on orders_payment p  (cost=0.29..8.31 rows=1 width=14) (never executed)
          Index Cond: (id = o.payment_id)
          Filter: (to_char(created_at, 'YYYY-MM'::text) = '2022-01'::text)
Planning Time: 1.515 ms
Execution Time: 0.044 ms
(9 rows)
```

**CREATE INDEX idx\_orders\_payment\_created\_at ON orders\_payment(created\_at);**

- **Lý do thêm chỉ mục:** Btree sắp xếp các giá trị created\_at (kiểu timestamp) theo thứ tự thời gian, cho phép truy cập nhanh đến các hàng trong một phạm vi thời gian.

```

QUERY PLAN
Aggregate (cost=16.95..16.96 rows=1 width=32) (actual time=0.006..0.006 rows=1 loops=1)
  -> Nested Loop (cost=0.57..16.94 rows=1 width=6) (actual time=0.004..0.004 rows=0 loops=1)
    -> Index Scan using idx_orders_order_staff_id on orders_order o  (cost=0.29..8.30 rows=1 width=8) (actual time=0.003..0.003 rows=0 loops=1)
      Index Cond: (staff_id = 123)
    -> Index Scan using orders_payment_pkey on orders_payment p  (cost=0.29..8.31 rows=1 width=14) (never executed)
      Index Cond: (id = o.payment_id)
      Filter: (to_char(created_at, 'YYYY-MM')::text) = '2022-01'::text
Planning Time: 1.229 ms
Execution Time: 0.029 ms
(9 rows)

```

## Câu 6 : Top 5 người tiêu dung nhiều nhất

```

CREATE OR REPLACE FUNCTION hustshop_top_5_customers_by_spending()
RETURNS TABLE (
    user_id BIGINT,
    full_name TEXT,
    email TEXT,
    total_spent NUMERIC(12,2)
) AS $$

BEGIN
    RETURN QUERY
    SELECT
        a.id AS user_id,
        (a.first_name || ' ' || a.last_name)::TEXT AS full_name,
        a.email::TEXT,
        SUM(o.order_total) AS total_spent
    FROM orders_order o
    JOIN accounts_account a ON o.user_id = a.id
    WHERE o.is_ordered = true
    GROUP BY a.id, a.first_name, a.last_name, a.email
    ORDER BY total_spent DESC
    LIMIT 5;
END;
$$ LANGUAGE plpgsql STABLE;

```

```

hustshoppe=# select*from hustshop_top_5_customers_by_spending();
   user_id |      full_name      |           email           | total_spent
-----+-----+-----+-----+
      6255 | Melanie Wolfe    | murphygloria@example.com | 2573.94
        400 | Bryce Gutierrez | hamptonchad@example.com | 2538.02
      7131 | Sara Sweeney    | ulin@example.com        | 2487.38
      4088 | Michelle Tucker | brookspeter@example.com | 2294.46
      5046 | Justin Carney    | taylorjessica@example.com | 2237.39
(5 rows)

```

```

EXPLAIN ANALYZE
SELECT
  a.id AS user_id,
  (a.first_name || ' ' || a.last_name)::TEXT AS full_name,
  a.email::TEXT,
  SUM(o.order_total) AS total_spent
FROM orders_order o
JOIN accounts_account a ON o.user_id = a.id
WHERE o.is_ordered = true
GROUP BY a.id, a.first_name, a.last_name, a.email
ORDER BY total_spent DESC
LIMIT 5;

```

```

QUERY PLAN
-----
Limit  (cost=126.63..126.64 rows=5 width=139) (actual time=1.901..1.904 rows=5 loops=1)
  -> Sort  (cost=126.63..129.11 rows=991 width=139) (actual time=1.901..1.902 rows=5 loops=1)
      Sort Key: (sum(o.order_total)) DESC
      Sort Method: top-N heapsort  Memory: 26kB
      -> HashAggregate  (cost=92.83..110.17 rows=991 width=139) (actual time=1.398..1.675 rows=619 loops=1)
          Group Key: a.id
          Batches: 1  Memory Usage: 321kB
          -> Hash Join  (cost=44.30..87.86 rows=994 width=49) (actual time=0.338..0.948 rows=994 loops=1)
              Hash Cond: (o.user_id = a.id)
              -> Seq Scan on orders_order o  (cost=0.00..40.94 rows=994 width=14) (actual time=0.016..0.367 rows=994 loops=1)
                  Filter: is_ordered
              -> Hash  (cost=31.91..31.91 rows=991 width=43) (actual time=0.316..0.317 rows=991 loops=1)
                  Buckets: 1024  Batches: 1  Memory Usage: 83kB
                  -> Seq Scan on accounts_account a  (cost=0.00..31.91 rows=991 width=43) (actual time=0.006..0.160 rows=991 loops=1)
Planning Time: 0.472 ms
Execution Time: 2.423 ms
(16 rows)

```

## Câu 7 : Xem nội dung giỏ hàng

```

CREATE OR REPLACE FUNCTION hustshop_view_cart_by_user_id(p_user_id BIGINT)
RETURNS TABLE (
    cart_id BIGINT,
    product_id BIGINT,
    product_name TEXT,
    quantity INT,
    price NUMERIC(10,2),
    total_price NUMERIC(12,2)
) AS $$

BEGIN
    RETURN QUERY
    SELECT
        cart.id,
        item.product_id,
        p.name::TEXT,
        item.quantity,
        p.price::NUMERIC(10,2),
        (item.quantity * p.price)::NUMERIC(12,2)
    FROM carts_cart cart
    JOIN carts_cartitem item ON item.cart_id = cart.id
    JOIN store_product p ON item.product_id = p.id
    WHERE cart.user_id = p_user_id;
END;
$$ LANGUAGE plpgsql STABLE;

```

```

hustshoppe=# SELECT * FROM hustshop_view_cart_by_user_id(1);
 cart_id | product_id |    product_name    | quantity | price   | total_price
-----+-----+-----+-----+-----+-----+
  42401 |      1629 | Athletic Shoes Plus |        4 | 435.00 | 1740.00
(1 row)

```

```

EXPLAIN ANALYZE
SELECT
    cart.id,
    item.product_id,
    p.name::TEXT,
    item.quantity,
    p.price::NUMERIC(10,2),
    (item.quantity * p.price)::NUMERIC(12,2)
FROM carts_cart cart
JOIN carts_cartitem item ON item.cart_id = cart.id
JOIN store_product p ON item.product_id = p.id
WHERE cart.user_id = 1;

```

```

-----  

QUERY PLAN  

-----  

Nested Loop  (cost=4.88..20.97 rows=2 width=84) (actual time=0.032..0.034 rows=1 loops=1)
  -> Nested Loop  (cost=4.59..20.31 rows=2 width=20) (actual time=0.024..0.025 rows=1 loops=1)
    -> Index Scan using unique_user_cart on carts_cart cart  (cost=0.29..8.30 rows=1 width=8) (actual time=0.009..0.010 rows=1 loops=1)
      Index Cond: (user_id = 1)
    -> Bitmap Heap Scan on carts_cartitem item  (cost=4.31..11.99 rows=2 width=20) (actual time=0.012..0.013 rows=1 loops=1)
      Recheck Cond: (cart_id = cart.id)
      Heap Blocks: exact=1
    -> Bitmap Index Scan on carts_cartitem_cart_id_9cb0a756  (cost=0.00..4.31 rows=2 width=0) (actual time=0.004..0.004 rows=1 loops=1)
      Index Cond: (cart_id = cart.id)
-> Index Scan using store_product_pkey on store_product p  (cost=0.29..0.32 rows=1 width=31) (actual time=0.005..0.005 rows=1 loops=1)
  Index Cond: (id = item.product_id)
Planning Time: 2.054 ms
Execution Time: 0.071 ms
(13 rows)

```

## Câu 8 : Tìm khách hàng chưa thanh toán bất kì đơn hàng nào

```

CREATE OR REPLACE FUNCTION hustshop_unpaid_customers()
RETURNS TABLE (
    user_id BIGINT,
    username TEXT
) AS $$  

BEGIN
    RETURN QUERY
    SELECT DISTINCT a.id AS user_id, a.username::TEXT
    FROM accounts_account a
    WHERE NOT EXISTS (
        SELECT 1
        FROM orders_order o
        WHERE o.user_id = a.id
        AND o.payment_id IS NOT NULL
    );
END;
$$ LANGUAGE plpgsql STABLE;

```

Kết quả:

```
hustshoppe=# SELECT * FROM hustshop_unpaid_customers();
 user_id |      username
-----+-----
 7113 | carlajohnson
 4638 | sara31
 7709 | tapiathomas
 854 | igraham
 43 | hritter
 9141 | caitlin67
 7953 | adamsabigail
 6556 | washingtonchristy
 6004 | rebecca00
 9916 | cynthiareyes
 7330 | rebecca47
 8867 | jennifersoto
 8125 | kleblanc
 1226 | sarahmeyers
 6770 | ssullivan
 9480 | cummingsdebra
 1427 | manuel24
 2138 | nicole37
```

```
EXPLAIN ANALYZE
SELECT DISTINCT a.id, a.username
FROM accounts_account a
WHERE NOT EXISTS (
    SELECT 1
    FROM orders_order o
    WHERE o.user_id = a.id
    AND o.payment_id IS NOT NULL
);
```

```
hustshoppe=#
                                         QUERY PLAN
-----
HashAggregate  (cost=941.17..976.63 rows=3546 width=40) (actual time=7.526..7.751 rows=3547 loops=1)
  Group Key: a.id, (a.username)::text
  Batches: 1  Memory Usage: 465kB
    -> Hash Right Anti Join  (cost=439.67..923.44 rows=3546 width=40) (actual time=6.323..6.797 rows=3547 loops=1)
        Hash Cond: (o.user_id = a.id)
        -> Seq Scan on orders_order o  (cost=0.00..394.85 rows=9285 width=8) (actual time=0.025..2.379 rows=9285 loops=1)
            Filter: (payment_id IS NOT NULL)
        -> Hash  (cost=321.52..321.52 rows=9452 width=18) (actual time=2.312..2.313 rows=9453 loops=1)
            Buckets: 16384  Batches: 1  Memory Usage: 598kB
            -> Seq Scan on accounts_account a  (cost=0.00..321.52 rows=9452 width=18) (actual time=0.016..1.072 rows=9453 loops=1)
Planning Time: 0.365 ms
Execution Time: 7.975 ms
```

### CREATE INDEX idx\_orders\_user\_payment ON orders\_order(user\_id, payment\_id)

- **Lý do thêm chỉ mục :** trong truy vấn gốc, truy vấn sử dụng NOT EXISTS với điều kiện o.user\_id = a.id AND o.payment\_id IS NOT NULL => yêu cầu kiểm tra từng tài khoản trong accounts\_account để xem có đơn hàng nào trong orders\_order liên kết với nó có payment\_id khác null hay không.
- Chỉ mục đã tạo là một chỉ mục B-tree trên hai cột user\_id và payment\_id, sắp xếp dữ liệu theo thứ tự tăng dần của user\_id và trong mỗi user\_id, sắp xếp theo payment\_id=> chỉ cần lọc theo 2 cột, giảm được số hàng quét

```

QUERY PLAN
HashAggregate  (cost=941.17..976.63 rows=3546 width=40) (actual time=4.127..4.356 rows=3547 loops=1)
  Group Key: a.id, (a.username)::text
  Batches: 1  Memory Usage: 465kB
    -> Hash Right Anti Join  (cost=439.67..923.44 rows=3546 width=40) (actual time=2.928..3.270 rows=3547 loops=1)
        Hash Cond: (o.user_id = a.id)
        -> Seq Scan on orders_order o  (cost=0.00..394.85 rows=9285 width=8) (actual time=0.007..0.909 rows=9285 loops=1)
            Filter: (payment_id IS NOT NULL)
        -> Hash  (cost=321.52..321.52 rows=9452 width=18) (actual time=1.303..1.303 rows=9453 loops=1)
            Buckets: 16384  Batches: 1  Memory Usage: 598kB
              -> Seq Scan on accounts_account a  (cost=0.00..321.52 rows=9452 width=18) (actual time=0.008..0.667 rows=9453 loops=1)
Planning Time: 1.389 ms
Execution Time: 4.615 ms
(12 rows)

```

## Câu 9 : Danh sách khuyến mãi đã được dùng

```

CREATE OR REPLACE VIEW used_promotions_summary AS
SELECT
    p.promotion_id,
    p.name AS promotion_name,
    COUNT(o.id) AS number_of_orders_used,
    MIN(o.created_at) AS first_used,
    MAX(o.created_at) AS last_used
FROM promotion p
JOIN orders_order o ON o.promotion_used_id = p.promotion_id
WHERE o.is_ordered = true
GROUP BY p.promotion_id, p.name;

```

**SELECT \* FROM used\_promotions\_summary;**

```

hustshop=# SELECT * FROM used_promotions_summary;
promotion_id | promotion_name | number_of_orders_used | first_used | last_used
-----+-----+-----+-----+-----+
 42 | Promo 42 | 15 | 2025-06-19 11:54:46.527224+07 | 2025-06-19 11:54:48.185721+07
 29 | Promo 29 | 14 | 2025-06-19 11:54:46.338588+07 | 2025-06-19 11:54:48.20169+07
 4 | Promo 4 | 19 | 2025-06-19 11:54:46.368853+07 | 2025-06-19 11:54:48.24809+07
 34 | Promo 34 | 19 | 2025-06-19 11:54:46.348629+07 | 2025-06-19 11:54:48.236314+07
 41 | Promo 41 | 19 | 2025-06-19 11:54:46.314728+07 | 2025-06-19 11:54:48.260299+07
 40 | Promo 40 | 19 | 2025-06-19 11:54:46.516437+07 | 2025-06-19 11:54:48.182247+07
 46 | Promo 46 | 18 | 2025-06-19 11:54:46.353064+07 | 2025-06-19 11:54:48.144906+07
 43 | Promo 43 | 19 | 2025-06-19 11:54:46.412855+07 | 2025-06-19 11:54:48.297549+07
 32 | Promo 32 | 6 | 2025-06-19 11:54:46.434947+07 | 2025-06-19 11:54:48.193634+07
 7 | Promo 7 | 16 | 2025-06-19 11:54:46.382396+07 | 2025-06-19 11:54:48.002974+07
 9 | Promo 9 | 22 | 2025-06-19 11:54:46.337297+07 | 2025-06-19 11:54:48.183947+07
 10 | Promo 10 | 15 | 2025-06-19 11:54:46.305682+07 | 2025-06-19 11:54:48.088485+07
 35 | Promo 35 | 20 | 2025-06-19 11:54:46.662064+07 | 2025-06-19 11:54:48.281634+07
 45 | Promo 45 | 33 | 2025-06-19 11:54:46.326254+07 | 2025-06-19 11:54:48.270997+07
 38 | Promo 38 | 17 | 2025-06-19 11:54:46.621314+07 | 2025-06-19 11:54:47.833881+07
 15 | Promo 15 | 13 | 2025-06-19 11:54:46.589374+07 | 2025-06-19 11:54:48.28414+07
 6 | Promo 6 | 25 | 2025-06-19 11:54:46.354361+07 | 2025-06-19 11:54:48.30557+07
 26 | Promo 26 | 23 | 2025-06-19 11:54:46.638345+07 | 2025-06-19 11:54:48.101872+07
 12 | Promo 12 | 17 | 2025-06-19 11:54:46.365165+07 | 2025-06-19 11:54:48.15065+07
 48 | Promo 48 | 18 | 2025-06-19 11:54:46.312161+07 | 2025-06-19 11:54:48.229278+07
 39 | Promo 39 | 20 | 2025-06-19 11:54:46.395495+07 | 2025-06-19 11:54:48.215848+07
 24 | Promo 24 | 22 | 2025-06-19 11:54:46.417395+07 | 2025-06-19 11:54:48.299624+07
 19 | Promo 19 | 24 | 2025-06-19 11:54:46.398368+07 | 2025-06-19 11:54:48.303521+07
 25 | Promo 25 | 21 | 2025-06-19 11:54:46.35174+07 | 2025-06-19 11:54:48.265293+07
 36 | Promo 36 | 23 | 2025-06-19 11:54:46.319128+07 | 2025-06-19 11:54:48.301631+07
 31 | Promo 31 | 23 | 2025-06-19 11:54:46.394289+07 | 2025-06-19 11:54:48.278881+07
 30 | Promo 30 | 13 | 2025-06-19 11:54:46.65334+07 | 2025-06-19 11:54:48.222797+07
 50 | Promo 50 | 27 | 2025-06-19 11:54:46.359158+07 | 2025-06-19 11:54:48.288755+07
 21 | Promo 21 | 31 | 2025-06-19 11:54:46.317067+07 | 2025-06-19 11:54:48.252712+07
 49 | Promo 49 | 26 | 2025-06-19 11:54:46.371236+07 | 2025-06-19 11:54:48.217842+07
 47 | Promo 47 | 14 | 2025-06-19 11:54:46.449178+07 | 2025-06-19 11:54:48.168995+07
 14 | Promo 14 | 20 | 2025-06-19 11:54:46.38033+07 | 2025-06-19 11:54:48.213816+07
 3 | Promo 3 | 25 | 2025-06-19 11:54:46.400971+07 | 2025-06-19 11:54:48.269793+07
 17 | Promo 17 | 14 | 2025-06-19 11:54:46.655103+07 | 2025-06-19 11:54:48.020046+07

```

```
EXPLAIN ANALYZE
SELECT * FROM used_promotions_summary;
```

```
QUERY PLAN
-----
HashAggregate  (cost=531.05..535.88 rows=483 width=37) (actual time=3.127..3.201 rows=483 loops=1)
  Group Key: p.promotion_id
  Batches: 1  Memory Usage: 105kB
    -> Hash Join  (cost=18.87..438.20 rows=9285 width=29) (actual time=0.083..2.447 rows=4605 loops=1)
        Hash Cond: (o.promotion_used_id = p.promotion_id)
        -> Seq Scan on orders_order o  (cost=0.00..394.85 rows=9285 width=20) (actual time=0.010..1.371 rows=9285 loops=1)
            Filter: is_ordered
        -> Hash  (cost=12.83..12.83 rows=483 width=13) (actual time=0.065..0.066 rows=483 loops=1)
            Buckets: 1024  Batches: 1  Memory Usage: 30kB
            -> Seq Scan on promotion p  (cost=0.00..12.83 rows=483 width=13) (actual time=0.006..0.038 rows=483 loops=1)
Planning Time: 0.319 ms
Execution Time: 3.255 ms
(12 rows)
```

## Câu 10 : Xếp khách hàng có đơn hàng gần đây nhất

```
CREATE OR REPLACE VIEW customer_latest_orders AS
SELECT
    a.id AS user_id,
    a.username,
    MAX(o.created_at) AS last_order_date
FROM accounts_account a
JOIN orders_order o ON o.user_id = a.id
WHERE o.is_ordered = true
GROUP BY a.id, a.username
ORDER BY last_order_date DESC;
```

```
hustshoppe=# SELECT*FROM customer_latest_orders;
 user_id |      username      | last_order_date
-----+-----+-----+
  7786 | pperez           | 2025-06-24 16:27:35+07
  3904 | craig72          | 2025-06-24 06:36:00+07
  4920 | amy78             | 2025-06-24 04:52:52+07
  6858 | greenpatricia    | 2025-06-24 03:46:55+07
   537 | garciagregory    | 2025-06-24 02:20:26+07
  2733 | karen95           | 2025-06-24 01:37:11+07
  2085 | philip09          | 2025-06-23 17:23:27+07
  2059 | edwardsjerry     | 2025-06-23 13:41:40+07
  9503 | tinarodriguez    | 2025-06-23 12:44:32+07
  6148 | brownmichael      | 2025-06-23 02:57:41+07
  6045 | dunnleslie        | 2025-06-23 01:58:31+07
  8263 | sophia63          | 2025-06-22 16:56:15+07
```

```
EXPLAIN ANALYZE
SELECT*FROM customer_latest_orders;
```

```
hustshop=# EXPLAIN ANALYZE
hustshop=# SELECT*FROM customer_latest_orders;
                                         QUERY PLAN
-----
Sort  (cost=152.07..154.55 rows=991 width=26) (actual time=0.571..0.583 rows=619 loops=1)
  Sort Key: (max(o.created_at)) DESC
  Sort Method: quicksort  Memory: 52kB
    >  HashAggregate  (cost=92.85..102.76 rows=991 width=26) (actual time=0.450..0.494 rows=619 loops=1)
        Group Key: a.id
        Batches: 1  Memory Usage: 129kB
    ->  Hash Join  (cost=44.30..87.87 rows=995 width=26) (actual time=0.136..0.346 rows=995 loops=1)
        Hash Cond: (o.user_id = a.id)
          ->  Seq Scan on orders_order o  (cost=0.00..40.95 rows=995 width=16) (actual time=0.007..0.125 rows=995 loops=1)
              Filter: is_ordered
          ->  Hash  (cost=31.91..31.91 rows=991 width=18) (actual time=0.125..0.125 rows=992 loops=1)
              Buckets: 1024  Batches: 1  Memory Usage: 58kB
          ->  Seq Scan on accounts_account a  (cost=0.00..31.91 rows=991 width=18) (actual time=0.003..0.073 rows=992 loops=1)
Planning Time: 0.342 ms
Execution Time: 0.615 ms
(15 rows)
```

## Câu 11 : Sản phẩm chưa được nhận đánh giá

```
hustshoppe=# CREATE OR REPLACE VIEW customer_latest_orders AS
hustshoppe-# SELECT
hustshoppe-#     a.id AS user_id,
hustshoppe-#     a.username,
hustshoppe-#     MAX(o.created_at) AS last_order_date
hustshoppe-# FROM accounts_account a
hustshoppe-# JOIN orders_order o ON o.user_id = a.id
hustshoppe-# WHERE o.is_ordered = true
hustshoppe-# GROUP BY a.id, a.username
hustshoppe-# ORDER BY last_order_date DESC;
CREATE VIEW
hustshoppe=# SELECT*FROM customer_latest_orders;


| user_id | username         | last_order_date        |
|---------|------------------|------------------------|
| 7786    | pperez           | 2025-06-24 16:27:35+07 |
| 3904    | craig72          | 2025-06-24 06:36:00+07 |
| 4920    | amy78            | 2025-06-24 04:52:52+07 |
| 6858    | greenpatricia    | 2025-06-24 03:46:55+07 |
| 537     | garciagregory    | 2025-06-24 02:20:26+07 |
| 2733    | karen95          | 2025-06-24 01:37:11+07 |
| 2085    | philip09         | 2025-06-23 17:23:27+07 |
| 2059    | edwardsjerry     | 2025-06-23 13:41:40+07 |
| 9503    | tinarodriguez    | 2025-06-23 12:44:32+07 |
| 6148    | brownmichael     | 2025-06-23 02:57:41+07 |
| 6045    | dunnleslie       | 2025-06-23 01:58:31+07 |
| 8263    | sophia63         | 2025-06-22 16:56:15+07 |
| 8605    | justincooper     | 2025-06-22 15:20:02+07 |
| 7472    | april30          | 2025-06-22 09:39:46+07 |
| 4540    | edwardking       | 2025-06-22 08:09:15+07 |
| 3970    | moorecheryl      | 2025-06-21 17:53:30+07 |
| 1361    | dixonkelly       | 2025-06-21 11:57:15+07 |
| 4158    | againes          | 2025-06-21 09:14:21+07 |
| 8489    | harrisbecky      | 2025-06-21 07:14:59+07 |
| 5758    | hernandezderrick | 2025-06-21 04:29:46+07 |
| 3945    | qhorn            | 2025-06-21 03:07:02+07 |
| 5251    | travisprice      | 2025-06-21 01:05:49+07 |
| 4398    | michael127       | 2025-06-20 18:18:26+07 |
| 4354    | robinsonwendy    | 2025-06-20 17:19:08+07 |
| 8031    | jason37          | 2025-06-20 05:22:10+07 |


```

```
----- QUERY PLAN -----
Hash Right Join  (cost=3.12..35.69 rows=1 width=28) (actual time=0.191..0.194 rows=2 loops=1)
  Hash Cond: (r.product_id = p.id)
  Filter: (r.id IS NULL)
  Rows Removed by Filter: 978
->  Seq Scan on reviews_review r  (cost=0.00..29.78 rows=978 width=16) (actual time=0.005..0.083 rows=978 loops=1)
    ->  Hash  (cost=2.50..2.50 rows=50 width=28) (actual time=0.020..0.020 rows=52 loops=1)
        Buckets: 1024  Batches: 1  Memory Usage: 12kB
          ->  Seq Scan on store_product p  (cost=0.00..2.50 rows=50 width=28) (actual time=0.008..0.013 rows=52 loops=1)
Planning Time: 0.273 ms
Execution Time: 0.210 ms
(10 rows)
```

## 6. Xây dựng Website

### 6.1. Mô tả tổng quan

Với sự chuẩn bị về quy trình nghiệp vụ và chức năng, cũng như là database, bạn em đã dựa vào đó làm cơ sở để xây dựng một website nhỏ demo các chức năng. Website được thiết kế cho cả quản lý cửa hàng, nhân viên và người dùng.

Với người dùng, website cung cấp một giao diện thân thiện giúp người dùng tìm sản phẩm, đặt hàng, thanh toán và theo dõi đơn hàng. Bên cạnh đó, hệ thống giúp người quản lý cửa hàng dễ dàng quản lý nhân viên, người dùng, hay các chức năng như xem thống kê doanh thu, báo cáo, quản lý sản phẩm, đơn hàng,....

### 6.2. Công nghệ sử dụng

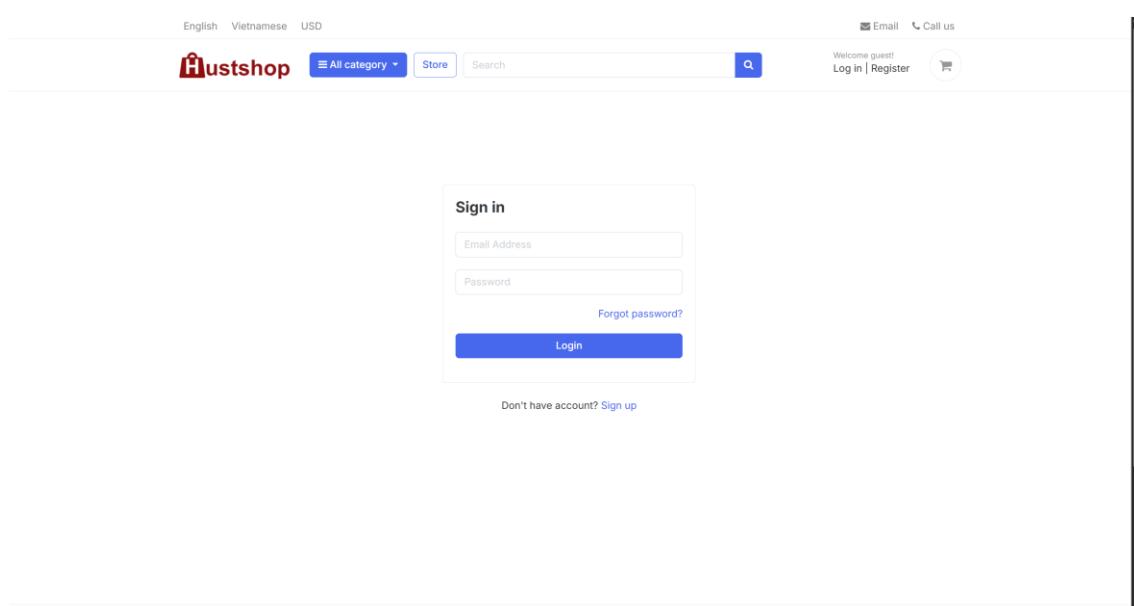
- Frontend : Html, CSS, Bootstrap
- Backend : Django ( Framework Python )
- Database : Postgre SQL

### 6.3. Demo website

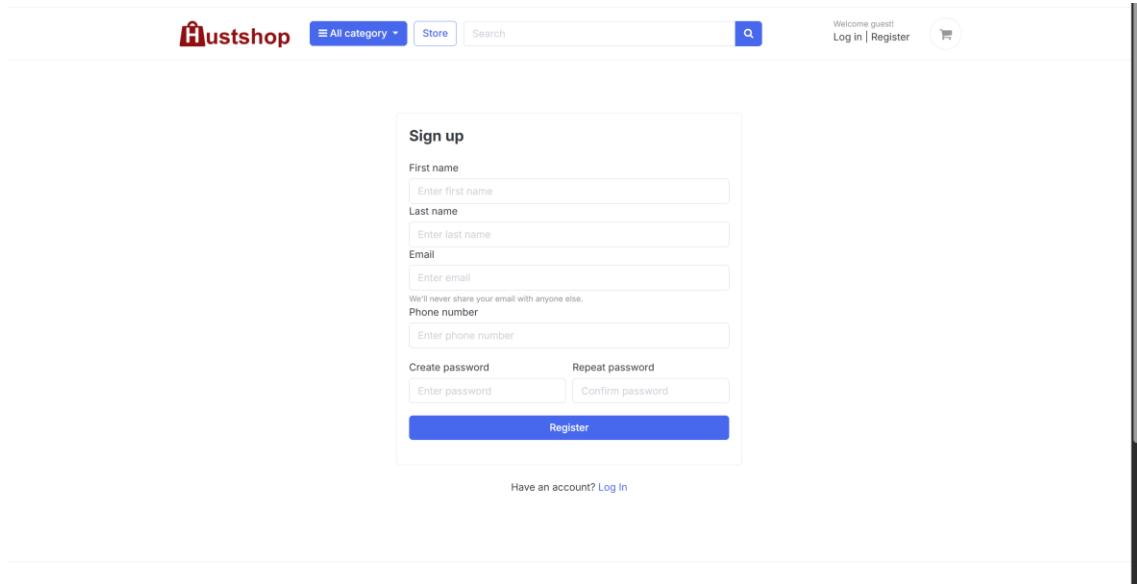
Link github dự án có thể xem [tại đây](#)

#### a. Với người dùng

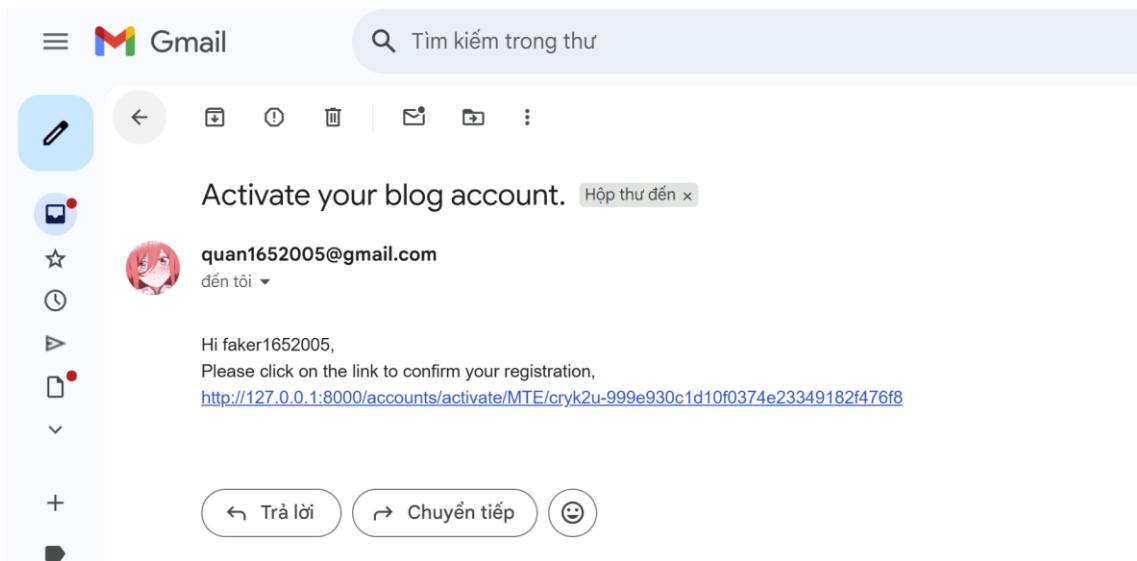
Ban đầu khi người dùng vào web, nếu như đã có tài khoản thì người dùng đăng nhập tại phần Log in



Còn nếu như người dùng chưa có tài khoản thì sẽ đăng ký tài khoản ở trang Register



Sau khi điền xong, hệ thống sẽ gửi 1 email xác thực tới cho người dùng, khi người dùng ấn vào link đó thì tài khoản mới được kích hoạt thành công



Sau khi kích hoạt thành công và đăng nhập vào trang web thì người dùng sẽ được chuyển hướng đến trang Dashboard. Ở trang này thì người dùng có thể xem các thông tin cá nhân, chỉnh sửa các thông tin ở phần Edit Profile, xem những đơn hàng của mình và đổi mật khẩu, nhưng tính năng đổi mật khẩu thì bọn em chưa phát triển a.

English Vietnamese USD

Welcome Quan! Dashboard | Logout

Hustshop All category Store Search

Dashboard My Orders Edit Profile Change Password Log out

Welcome back!

Thông tin tài khoản

Tên đầy đủ: Quan Nguyenn

Email: faker1652005@gmail.com

Số điện thoại: 0862853702

Ngày tham gia: 25/06/2025

Xem đơn hàng Sửa thông tin Tiếp tục mua sắm

Company About us Career Find a store Terms & conditions Sitemap

Help Contact us Money refund Order status Shipping info Open dispute

Account User Login User register Account Setting My Orders

Social Facebook Twitter Instagram YouTube

Sau đó người dùng ấn vào trang mua sắm chính, để tìm sản phẩm mình cần

English Vietnamese USD

Welcome Quan! Dashboard | Logout

Hustshop All category Store Search



Popular products

|   |   |  |   |
|---|---|--|---|
|  |  |  |  |
| BA Shirt<br>100000VND   | HUST Shirt<br>200000VND   | FTU Jacket<br>150000VND  | VNU Shirt<br>169000VND  |
| See all   |   |  |   |

Người dùng cũng có thể tìm kiếm bằng thanh tìm kiếm hoặc bằng những filter như danh mục, số sao trung bình, khoảng giá,...

Categories

- All Products
- Jackets
- Jeans
- Shirts
- Shoes
- T Shirt

Sizes

- L
- M
- S
- XL
- XS
- XXL

Colors

- Black
- Blue
- Green
- Grey
- Red
- White
- Yellow

Price range

- Min 0VND
- Max 500.000VND

Rating

- Any rating

View detail

BA Shirt 100000VND

HUST Shirt 200000VND

FTU Jacket 150000VND

VNU Shirt 169000VND

UEB Shoes 250000VND

AEF Shoes 125000VND

First Previous 1 2 Next Last

Apply Filters

Clear Filters

Người dùng có thể ấn vào View Detail để xem chi tiết sản phẩm, cũng như cả comment đánh giá sản phẩm. Sau đó chọn các biến thể màu sắc và kích thước để thêm vào giỏ hàng.

Hustshop All category Store Search

Welcome Guest Dashboard Logout

FTU Jacket

★★★★★ 1 reviews

150000VND

Choose Color: Blue

Select Size: S

Add to Cart

Write Your Review

How do you rate this product?

★★★★★

Review Title:

Review:

Submit Review

Customer Reviews

★★★★★ 1 reviews

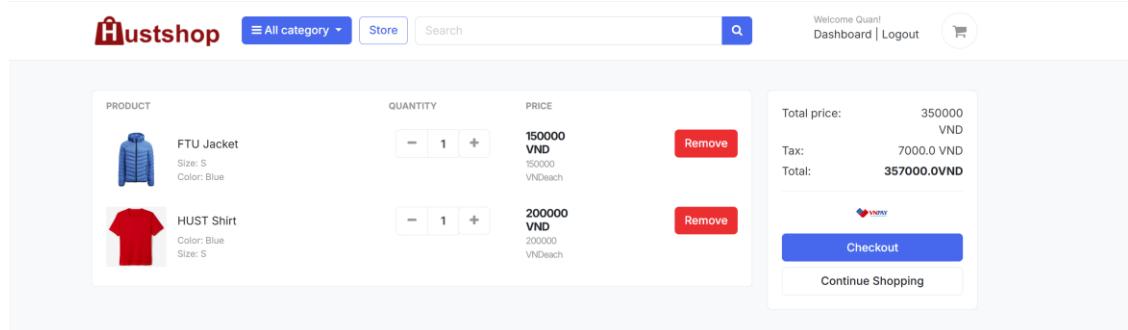
Quan Nguyen

★★★★★

cood

June 10, 2025, 5:15 p.m.

Sau đó người dùng có thể vào Giỏ hàng để kiểm tra, có thể tăng/giảm số lượng sản phẩm và xóa sản phẩm nếu không có nhu cầu



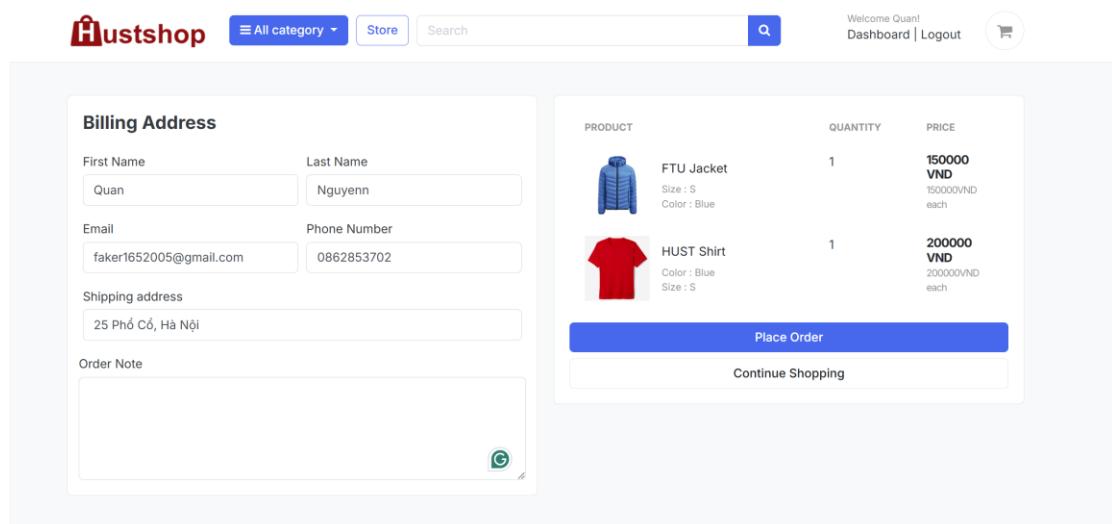
The screenshot shows the Hustshop shopping cart interface. At the top, there are navigation links for 'All category', 'Store', 'Search', and user information ('Welcome Quan! Dashboard | Logout'). The cart summary on the right indicates a total price of 350,000 VND, tax of 7,000.0 VND, and a total of 357,000.0 VND. The cart contains two items:

- FTU Jacket**: Size: S, Color: Blue. Price: 150,000 VND (150,000 VNĐ each).
- HUST Shirt**: Color: Blue, Size: S. Price: 200,000 VND (200,000 VNĐ each).

Below the cart summary are buttons for 'Checkout' and 'Continue Shopping'.

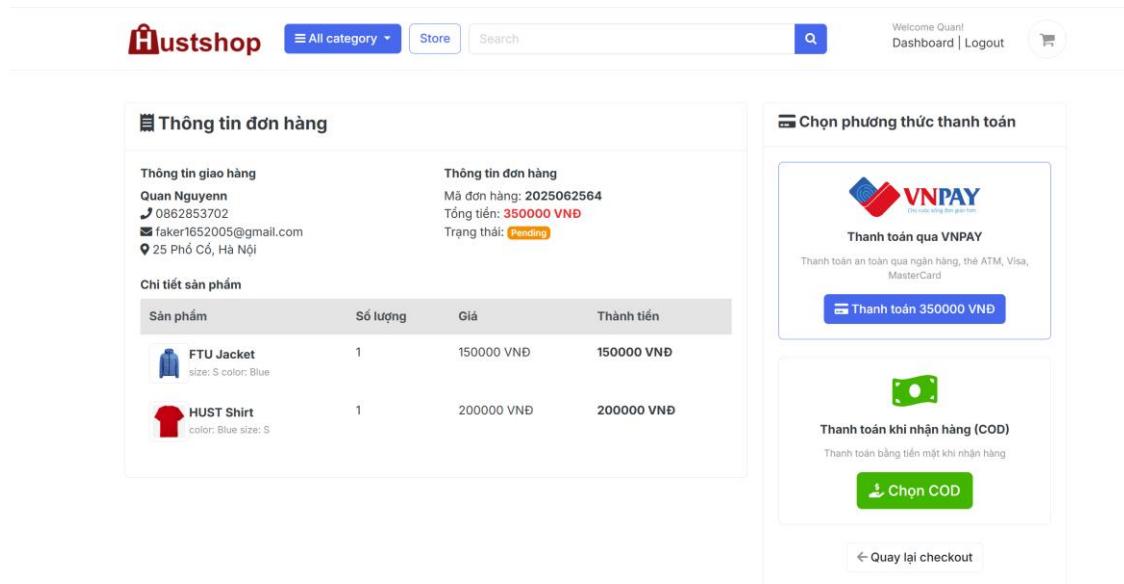
At the bottom of the page, there are links for 'Company', 'Help', 'Account', and 'Social' categories.

Sau khi ấn checkout, thì hệ thống sẽ chuyển người dùng đến trang điền thông tin nhận của đơn hàng.



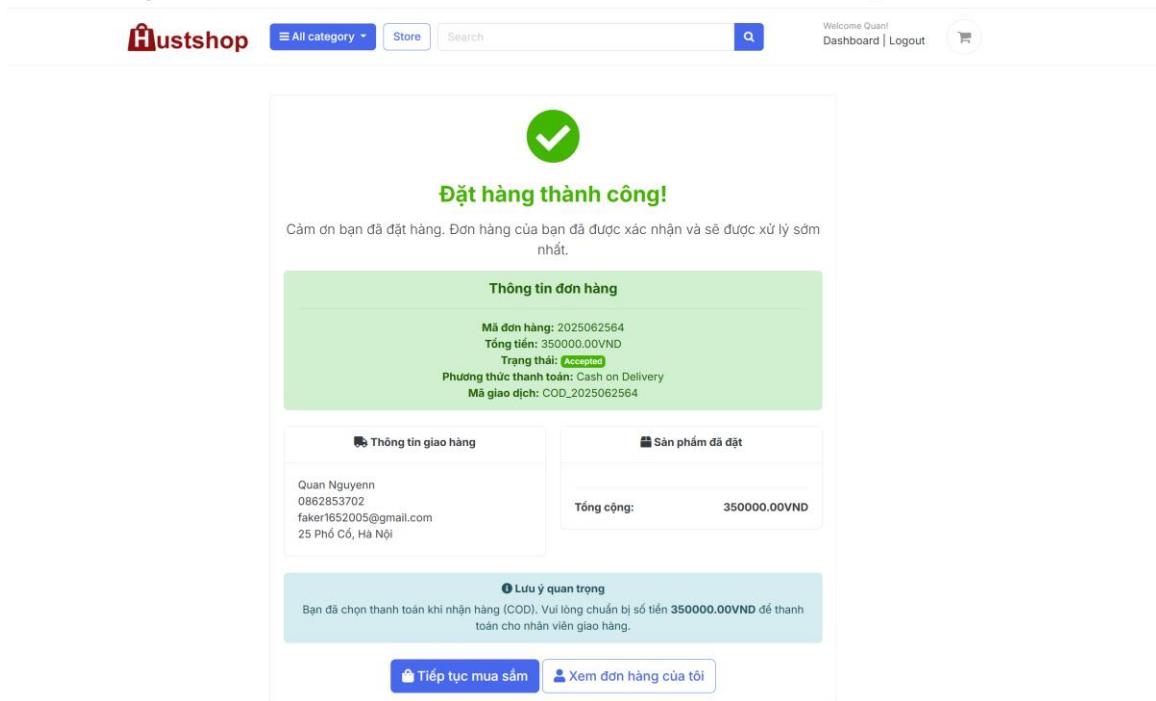
The screenshot shows the Hustshop checkout page. At the top, there are navigation links for 'All category', 'Store', 'Search', and user information ('Welcome Quan! Dashboard | Logout'). The left side features a 'Billing Address' form with fields for First Name (Quan), Last Name (Nguyenn), Email (faker1652005@gmail.com), Phone Number (0862853702), Shipping address (25 Phố Cổ, Hà Nội), and Order Note (empty). The right side shows the order summary with the same two items as the cart: FTU Jacket and HUST Shirt, both at 1 unit each. The total price is 357,000 VND. Below the summary are buttons for 'Place Order' and 'Continue Shopping'.

Khi ấn PlaceOrder, người dùng sẽ được chuyển tới trang xác nhận các thông tin về đơn hàng. Tại đây, người dùng có thể chọn 2 hình thức, 1 là thanh toán qua VNPay, 2 là COD.

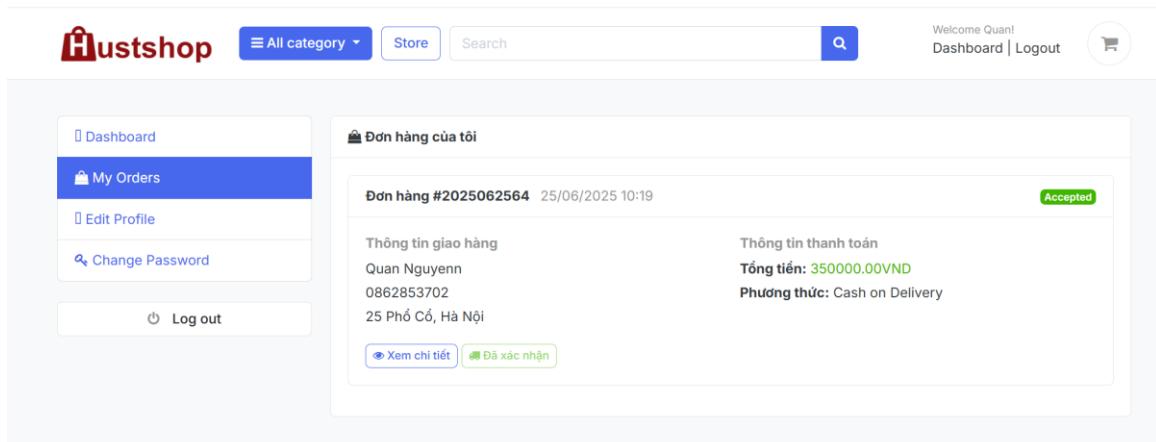


The screenshot shows a checkout page for Hustshop. On the left, there's a summary of the order items: FTU Jacket (1 unit, 150,000 VND) and HUST Shirt (1 unit, 200,000 VND). On the right, there are two payment selection boxes. The first box, 'Chọn phương thức thanh toán', offers 'Thanh toán qua VNPay' (VNPay payment method) and 'Thanh toán khi nhận hàng (COD)' (COD payment method). The second box, 'VNPay', shows a payment of 350,000 VND.

Khi người dùng ấn vào thanh toán tiền mặt COD, hệ thống sẽ thông báo thanh toán thành công



Khi ấn vào Xem đơn hàng của tôi, sẽ chuyển người dùng đến phần My Orders trong Dashboard. Ở đây sẽ hiển thị các đơn hàng người dùng đã đặt.



#### Company

About us  
Career  
Find a store  
Terms & conditions  
Sitemap

#### Help

Contact us  
Money refund  
Order status  
Shipping info  
Open dispute

#### Account

User Login  
User register  
Account Setting  
My Orders

#### Social

Facebook  
 Twitter  
 Instagram  
 Youtube

## b. Với quản lý

Với quản lý sẽ có 1 trang riêng để quản lý cửa hàng

Django administration

Site administration

ACCOUNTS

MODEL NAME ADD LINK CHANGE OR VIEW LIST LINK

Accounts [+ Add](#) [Change](#)

AUTHENTICATION AND AUTHORIZATION

MODEL NAME ADD LINK CHANGE OR VIEW LIST LINK

Groups [+ Add](#) [Change](#)

CARTS

MODEL NAME ADD LINK CHANGE OR VIEW LIST LINK

Cart items [+ Add](#) [Change](#)

Carts [+ Add](#) [Change](#)

ORDERS

MODEL NAME ADD LINK CHANGE OR VIEW LIST LINK

Order products [+ Add](#) [Change](#)

Orders [+ Add](#) [Change](#)

Payments [+ Add](#) [Change](#)

Recent actions

My actions

- Deleted: Order #2025062554 - Accepted Order
- Deleted: Order #2025062551 - Accepted Order
- Deleted: Order #2025062557 - Pending Order
- Deleted: Order #2025062555 - Accepted Order
- Deleted: Order #2025062553 - Pending Order
- Deleted: Order #2025062552 - Accepted Order
- Deleted: Order #2025062558 - Pending Order
- Deleted: Order #2025062559 - Accepted Order
- Deleted: Order #2025062556 - Pending Order
- Deleted: Order #2025062550 -

Ở đây thì quản lý có thể xem/thêm/sửa/xóa các sản phẩm, danh mục hay quản lý các đơn hàng, mã giảm giá, loại bỏ reviews không hợp lệ.

Home · Store · Products

Carts [+ Add](#)

ORDERS

MODEL NAME ADD LINK CHANGE OR VIEW LIST LINK

Order products [+ Add](#)

Orders [+ Add](#)

Payments [+ Add](#)

Promotions [+ Add](#)

User Promotion Usages [+ Add](#)

REVIEWS

MODEL NAME ADD LINK CHANGE OR VIEW LIST LINK

Reviews [+ Add](#)

Select Product to change

Action: [—](#) [Go](#) 0 of 10 selected

| NAME           | PRICE  | STOCK | CATEGORY | CREATED AT              | UPDATED AT                | IS AVAILABLE |
|----------------|--------|-------|----------|-------------------------|---------------------------|--------------|
| BA Shirt       | 100000 | 8     | Shirts   | June 9, 2025, 9:26 a.m. | June 24, 2025, 7:49 p.m.  | ✓            |
| HUST Shirt     | 200000 | 18    | Shirts   | June 4, 2025, 6:34 p.m. | June 25, 2025, 10:23 a.m. | ✓            |
| FTU Jacket     | 150000 | 16    | Jackets  | June 2, 2025, 4:28 p.m. | June 25, 2025, 10:23 a.m. | ✓            |
| VNU Shirt      | 169000 | 34    | Shirts   | June 2, 2025, 4:27 p.m. | June 24, 2025, 7:46 p.m.  | ✓            |
| UEB Shoes      | 250000 | 0     | Shoes    | June 2, 2025, 4:26 p.m. | June 11, 2025, 9:29 a.m.  | ✓            |
| AEF Shoes      | 125000 | 0     | Shoes    | June 2, 2025, 4:26 p.m. | June 11, 2025, 9:29 a.m.  | ✓            |
| UET Shoes      | 350000 | 20    | Shoes    | June 2, 2025, 4:22 p.m. | June 11, 2025, 9:30 a.m.  | ✓            |
| HUCAET T Shirt | 450000 | 50    | T Shirt  | June 2, 2025, 4:21 p.m. | June 11, 2025, 9:30 a.m.  | ✓            |
| NEU Jacket     | 499000 | 10    | Jackets  | June 2, 2025, 4:19 p.m. | June 11, 2025, 9:30 a.m.  | ✓            |
| ATX Jeans      | 420000 | 20    | Jeans    | June 2, 2025, 4:18 p.m. | June 11, 2025, 9:31 a.m.  | ✓            |

10 Products

ADD PRODUCT +

Skip to main content

Django administration

Home · Store · Products > HUST Shirt

Start typing to filter...

ACCOUNTS

| MODEL NAME | ADD LINK              | CHANGE OR VIEW LIST LINK |
|------------|-----------------------|--------------------------|
| Accounts   | <a href="#">+ Add</a> |                          |

AUTHENTICATION AND AUTHORIZATION

| MODEL NAME | ADD LINK              | CHANGE OR VIEW LIST LINK |
|------------|-----------------------|--------------------------|
| Groups     | <a href="#">+ Add</a> |                          |

CARTS

| MODEL NAME | ADD LINK              | CHANGE OR VIEW LIST LINK |
|------------|-----------------------|--------------------------|
| Cart items | <a href="#">+ Add</a> |                          |
| Carts      | <a href="#">+ Add</a> |                          |

ORDERS

| MODEL NAME | ADD | CHANGE |
|------------|-----|--------|
|------------|-----|--------|

Change Product

**HUST Shirt**

Name:

Slug:

Description:

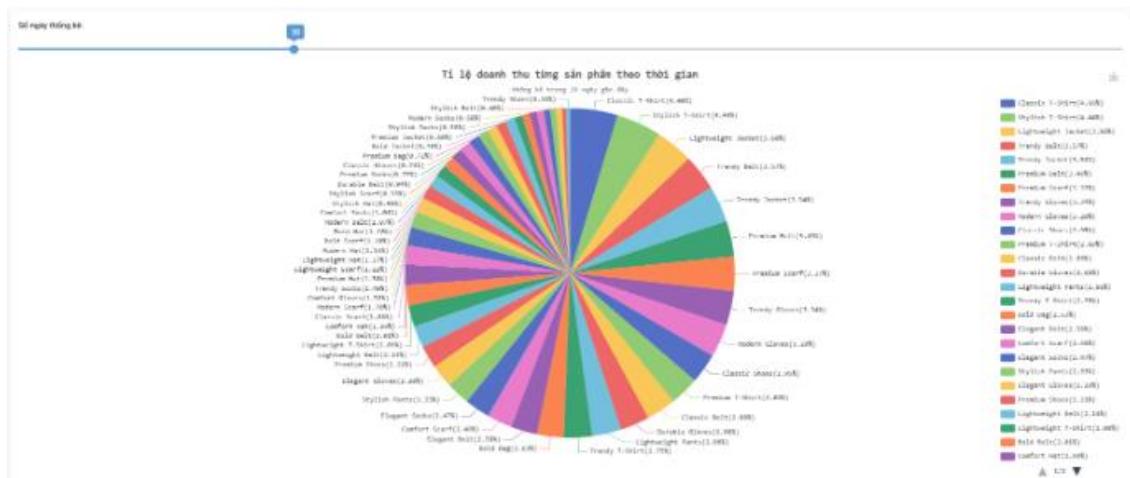
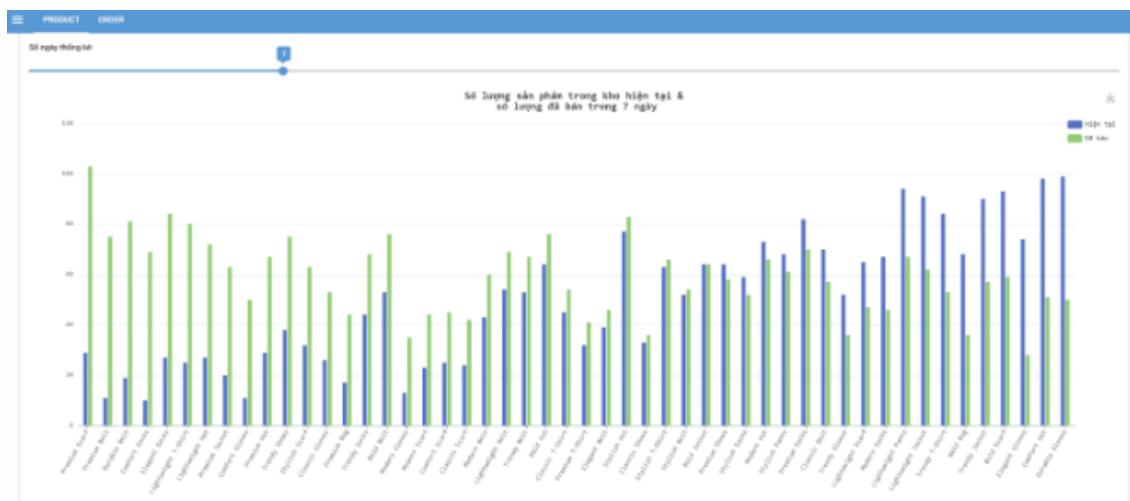
Image:    No file chosen

Price:

Stock:

[HISTORY](#)

Quản lý cũng có thể xem được doanh thu cũng như là thống kê của cửa hàng



### c. VỚI NHÂN VIÊN

Nhân viên thì sẽ được phân 1 số quyền như xem các reviews, xóa bỏ nếu như không hợp lệ, xem các đơn hàng và chuyển trạng thái đơn hàng từ “pending” → “processing”

The screenshot shows the Django admin interface for the Orders model. The top navigation bar includes 'WELCOME, HAQ@GMAIL.COM' and 'Log out'. The main content area displays a success message: 'The order "Order #2025062564 - processing" was changed successfully.' Below this, a table lists five orders with columns: ORDER NUMBER, USER, ORDER STATUS, ORDER TOTAL, and CREATED AT. The table shows the following data:

| ORDER NUMBER | USER                   | ORDER STATUS | ORDER TOTAL | CREATED AT                |
|--------------|------------------------|--------------|-------------|---------------------------|
| 2025062564   | faker1652005@gmail.com | Processing   | 350000.00   | June 25, 2025, 10:19 a.m. |
| 2025062563   | quan@gmail.com         | -            | 150000.00   | June 24, 2025, 7:52 p.m.  |
| 2025062562   | quan@gmail.com         | -            | 100000.00   | June 24, 2025, 7:49 p.m.  |
| 2025062561   | quan@gmail.com         | -            | 338000.00   | June 24, 2025, 7:46 p.m.  |
| 2025062560   | quan@gmail.com         | -            | 100000.00   | June 24, 2025, 7:43 p.m.  |

Below the table, a message indicates '5 orders' were selected. The right sidebar contains a 'FILTER' section with dropdown menus for 'Show counts', 'By order status' (All, Pending, Processing, Shipped, Completed, Cancelled), and 'By created at' (Any date, Today, Past 7 days, This month, This year).

The screenshot shows the Django admin interface for the Reviews model. The top navigation bar includes 'WELCOME, HAQ@GMAIL.COM' and 'Log out'. The main content area displays a success message: 'The review "UEB Shoes" was changed successfully.' Below this, a table lists reviews with columns: PRODUCT, USER, RATING, REVIEW, STATUS, and CREATED AT. The table shows the following data:

| PRODUCT      | USER              | RATING | REVIEW   | STATUS | CREATED AT               |
|--------------|-------------------|--------|----------|--------|--------------------------|
| UEB Shoes    | quanhao@gmail.com | 4.0    | nice     | ✓      | June 10, 2025, 5:15 p.m. |
| FTU Jacket   | quanhao@gmail.com | 5.0    | niceee   | ✓      | June 10, 2025, 5:15 p.m. |
| BA Shirt     | quanhao@gmail.com | 3.0    | not bad  | ✓      | June 10, 2025, 5:14 p.m. |
| NEU Jacket   | quanhao@gmail.com | 2.5    | 4        | ✓      | June 9, 2025, 7:44 a.m.  |
| UET Shoes    | quanhao@gmail.com | 1.5    | 2        | ✓      | June 9, 2025, 7:43 a.m.  |
| HUCE T Shirt | quanhao@gmail.com | 2.5    | 1        | ✓      | June 9, 2025, 7:43 a.m.  |
| VNU Shirt    | quanhao@gmail.com | 4.5    | 23       | ✓      | June 9, 2025, 6:39 a.m.  |
| HUST Shirt   | quanhao@gmail.com | 4.0    | 123      | ✓      | June 9, 2025, 6:38 a.m.  |
| HUST Shirt   | quan@gmail.com    | 1.5    | ádasdasd | ✓      | June 8, 2025, 7:39 a.m.  |

The right sidebar contains a 'FILTER' section with dropdown menus for 'Show counts', 'By rating' (All, 1.5, 2.5, 3.0, 4.0, 4.5, 5.0), and 'By created at' (Any date, Today, Past 7 days, This month, This year).

## 7. Khó khăn khi thực hiện và cách giải quyết

Trong quá trình thực hiện dự án, bạn em gặp 1 số các vấn đề bất trắc như sau:

### a. Về ERD và mô hình quan hệ dữ liệu

Ban đầu, khi chuyển từ ERD sang mô hình quan hệ, bạn em định chia ra cụ thể nhân viên riêng, quản lý riêng và người dùng thành các bảng riêng nhưng bạn em cảm thấy làm như vậy thì có vẻ như không được tối ưu bởi bản chất cả nhân viên, quản lý hay người dùng thì đều có những điểm chung như tên, gmail, địa chỉ, giới tính,.....

➔ **Cách giải quyết :** Bạn em đã giải quyết bằng cách chỉ giữ lại 1 bảng account và thêm 1 thuộc tính Role để phân quyền cho người dùng, nhân viên và quản lý.

Bên cạnh đó, bạn em cũng có một bài khá rắc rối liên quan tới các biến thể liên quan tới sản phẩm như kích thước và màu sắc. Ban đầu chúng em định tách riêng màu sắc, kích thước riêng để xử lý, nhưng khi làm vậy thì bạn em nhận ra làm vậy thì sẽ rất tốn thời gian, bởi tổ hợp màu sắc, kích thước là rất nhiều.

➔ **Cách giải quyết :** Bạn em đã gộp 2 thuộc tính đa trị color và size thành 1 thuộc tính đa trị là biến thể, khi ta muốn thêm các biến thể cho sản phẩm, thì ta chỉ cần vào bảng variation, chọn mục thêm là color hay là size, nhập giá trị rồi thêm.

### b. Sinh data mẫu

Để có thể kiểm tra được hiệu năng của các truy vấn, bạn em đã sử dụng 1 file python để gen data mẫu dựa vào ERD và mô hình quan hệ bạn em đã thiết kế ở trên( khoảng 500k dữ liệu ). Trong quá trình chạy file data, đã có một số lỗi nhỏ, có thể là khi gen data có sự trùng lặp khiến thất thoát khoảng 50k. Dẫu vậy thì tổng quan quá trình gen data vẫn ổn đến 90%.

Ban đầu khi chạy phiên bản đầu tiên, khi đã chạy data trên database, đến phần truy vấn em đã phát hiện 1 số trực trắc. Đó là khi em viết truy vấn liên quan đến ngày tháng, khi em viết 1 index cho câu đó rồi chạy phân tích thì thấy hệ thống không hề sử dụng cái index đó mà vẫn sử dụng Seq scan, mặc dù ở bảng đó có tới hơn 10k bản ghi. Theo lí thuyết thì nó phải dùng cái index trên cột date. Sau đó bạn em đã phát hiện ra đó chính là vấn đề nằm ở cột date, dạng

dữ liệu đang là current stamp nên khi gen ra tất cả dữ liệu trong cột date đều là ngày hôm đó, dẫn tới việc hệ thống không hề dùng tới Index Scan.

➔ **Cách giải quyết :** Bạn em đã điều chỉnh lại file python sinh dữ liệu sao cho random cả phần date ( Như trong file data mẫu, bạn em gen từ 2020 – hiện tại).

➔ **Kết quả :** Khi bạn em chạy lại phân tích trên data mới thì hệ thống đã sử dụng Index Scan trên cột date → Đúng như mong muốn.

### c. Website demo

Khi bắt đầu làm web, bạn em đã gặp khó khăn trong việc chọn ngôn ngữ lập trình cũng như là framework. Quý thì muốn dùng NiceGUI, FastAPI, Sinh thì muốn sử dụng JavaScript với NodeJS và Quân thì lại chọn Django.

➔ Nhưng sau quá trình trao đổi và bàn bạc kĩ lượng thì bạn em đã chọn Django. Bởi bạn em chỉ định làm website quy mô nhỏ và làm trong 1 thời gian ngắn nên Django là lựa chọn phù hợp nhất với cú pháp đơn giản, dễ sử dụng, hệ thống tích hợp nhiều tính năng sẵn như tích hợp bảo mật và xác thực qua CSRF,.... và sử dụng mô hình Model – View – Controller thuận tiện cho việc triển khai.

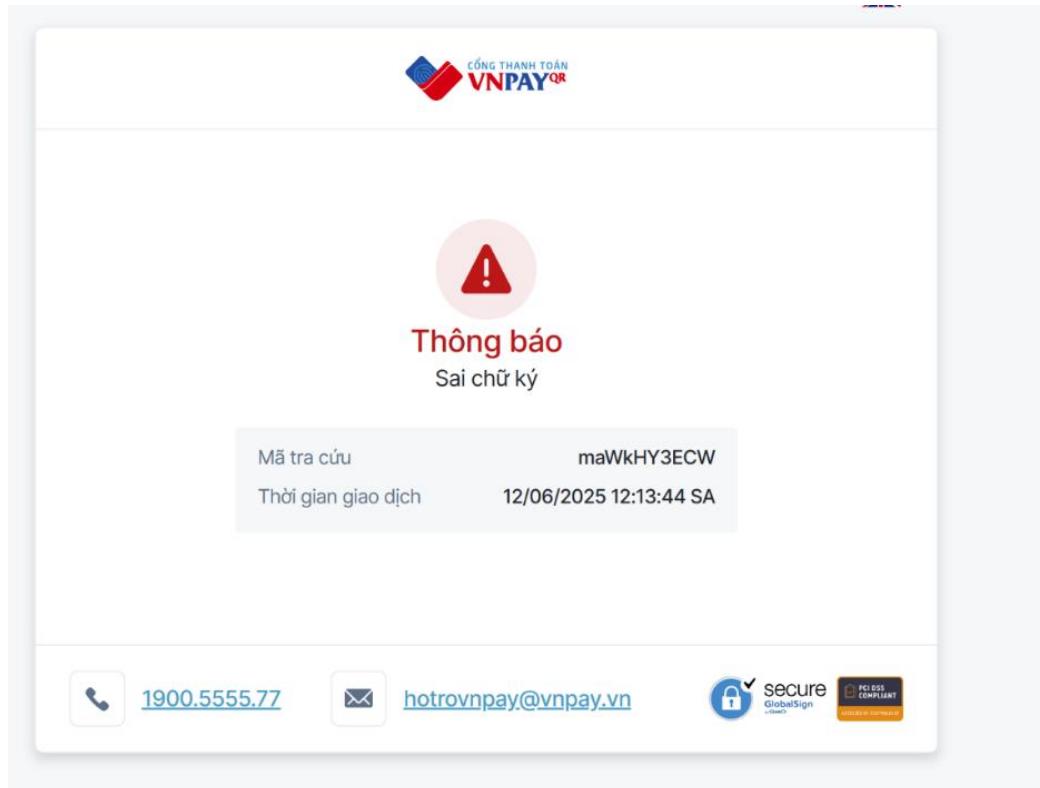
Khi bạn em thử liên kết web với database đã import dữ liệu mẫu vào thì ở trang chủ chính phải mất tới 30s mới load được web, do số lượng sản phẩm quá lớn (~10k) dẫn tới việc đôi khi bị overload.

➔ **Cách giải quyết :** Bạn em đã giải quyết bằng cách sử dụng phân trang cho phần store, tránh được việc overload và khiến người dùng cảm thấy ổn định hơn. Khi không có phân trang thì người dùng phải kéo rất lâu mới hết được trang sản phẩm nên không tối ưu.

Về các chức năng bạn em cũng gặp 1 số các khó khăn

- Do chưa triển khai trên 1 domain công khai mà chỉ dùng ở localhost, nên chức năng xác thực chỉ thực hiện được trên máy của bạn em.
- Về chức năng thanh toán, bạn em gặp một số khó khăn. Bạn em đã đang kí merchant môi trường test của VNPA Y, với đầy đủ các thông tin cấu hình như vnp\_TmnCode, vnp\_HashSecret, vnp\_URL, vnp\_IPN nhưng khi em gọi tới thì nó có chuyển tới sandbox của VNPA Y nhưng lại thông báo là lỗi chữ ký( như ảnh dưới ). Em có kiểm tra lại mã TMN cũng như HashSecret thì thấy vẫn đúng, ở phần xác thực phản hồi từ cổng thanh toán VNPA Y thông qua Hmac-Sha512 thì em nghĩ là do lỗi ở phần này. Em cũng đã thử sử dụng ngrok để tạo 1 URL công khai tạm thời cho

phản thanh toán này nhưng vẫn bị lỗi chữ ký. Xong do chưa có nhiều thời gian nghiên cứu và tìm hiểu, nhóm bạn em tạm thời bỏ qua chức năng này.



## 8. Đánh giá kết quả đạt được

### 8.1. Ưu điểm

- Nhóm đã xây dựng thành công mô hình ERD và chuyển đổi qua mô hình quan hệ dữ liệu một cách tối ưu, đảm bảo tính nhất quán với nghiệp vụ và chức năng.
- Website đã triển khai được đầy đủ các tính năng cơ bản cần có như đăng ký/đăng nhập, quản lý sản phẩm, giỏ hàng, thanh toán (COD), theo dõi đơn hàng.
- Giao diện thân thiện với người dùng, phân trang hiệu quả xử lý lượng sản phẩm lớn
- Phân quyền rõ ràng cho 3 role : khách hàng, nhân viên, quản lý
- Sinh được lượng dữ liệu đủ lớn để thử nghiệm với việc truy vấn và liên kết với web

### 8.2. Hạn chế

- Chưa xây dựng website cho nhân viên vận chuyển kiểm tra và xác nhận, hiện tại bọn em vẫn đang dùng cách thủ công là biên lai giấy xác nhận.
- Chưa tích hợp được chức năng hủy đơn hàng trên giao diện khách hàng, hiện tại bọn em vẫn đang sử dụng cách thủ công là gọi điện cho shop trong vòng 30' sau khi thanh toán.
- Chưa triển khai được chức năng đổi mật khẩu và chưa triển khai được chức năng xác thực trên domain thực tế.

## 9. Nhiệm vụ các thành viên

| Thành viên         | Nhiệm vụ  |
|--------------------|---|
| Đỗ Xuân Quý ( NT ) | Làm web thống kê cho quản lý, làm ERD, mô hình quan hệ, tạo truy vấn, tham gia làm mô tả nghiệp vụ.   |
| Nguyễn Minh Quân   | Xây dựng website, làm mô tả nghiệp vụ và chức năng, viết báo cáo, làm slide, tham gia làm ERD, mô hình quan hệ, tạo truy vấn, tham gia sinh data. |
| Trần Đăng Sinh     | Sinh data cho mô hình, tham gia làm ERD, mô hình quan hệ, tham gia làm nghiệp vụ, tạo truy vấn.   |



