

BỘ CÔNG THƯƠNG
TRƯỜNG ĐẠI HỌC CÔNG THƯƠNG TP.HCM
KHOA CÔNG NGHỆ THÔNG TIN



ĐỒ ÁN MÔN HỌC
CƠ SỞ DỮ LIỆU NOSQL

**ĐỀ TÀI: TÌM HIỂU CÁC CHỨC NĂNG CỦA CÔNG CỤ DBEAVER
ENTERPRISE EDITION ĐỂ QUẢN TRỊ VÀ KHAI THÁC CƠ SỞ DỮ LIỆU
CASSANDRA TRONG WEBSITE “BÁN HÀNG MỸ PHẨM ONLINE”**

Thành viên nhóm

GVHD: Nguyễn Thị Thu Tâm

1. Trương Lê Bảo Trân - 2001216232
2. Phan Thế Thanh - 2001216141
3. Phan Trần Minh Tâm - 2001216126
4. Nguyễn Minh Trí – 2001216237

TP. HỒ CHÍ MINH - 10/2024

Lời Cảm Ơn

Lời đầu tiên, nhóm em xin được gửi lời cảm ơn chân thành đến cô Nguyễn Thị Thu Tâm, trong quá trình học tập và tìm hiểu môn Cơ sở dữ liệu Nosql, nhóm em đã nhận được rất nhiều sự quan tâm, giúp đỡ, hướng dẫn tâm huyết và tận tình của cô. Cô đã giúp nhóm em tích lũy thêm nhiều kiến thức về môn học này để có thể hoàn thành được bài tiểu luận về đề tài Website bán hàng mỹ phẩm Moon Shop bằng hệ quản trị Cassandra

Nhóm đã cố gắng vận dụng những kiến thức đã học được và tìm tòi thêm nhiều thông tin để hoàn thành đồ án này. Trong quá trình làm bài, chắc chắn chúng em sẽ mắc phải những thiếu sót nên rất mong nhận được sự đóng góp ý kiến của cô để chúng em có cơ hội sửa sai, phát triển trong thực tế và từng bước hoàn thiện hơn nữa nội dung về đề tài: Website bán hàng mỹ phẩm Moon Shop bằng hệ quản trị cassandra.

Nhóm em xin trân trọng cảm ơn!

Bảng phân công tiến độ hàng tuần

		TRÂN	TÂM	THANH	TRÍ
Tuần 2 19/8/2024- 25/8/2024	Nghiên cứu phần cài đặt các ứng dụng cô giao trên lớp	Tìm hiểu cài đặt MongoDB	Tìm hiểu cài đặt Neo4j	Tìm hiểu cài đặt Cassandra	Tìm hiểu cài đặt Cassandra
Tuần 3 26/8/2024- 1/9/2024	Tìm hiểu về Cassandra Thiết kế giao diện ứng dụng	Giao diện Menu, Footer, Header	Giao diện Trang chủ	Giao diện thanh toán	Giao diện trang thông tin, giới thiệu
		Tìm hiểu tổng quan về Cassandra (đặc điểm, bản chất...)			
		Hợp nhóm bàn luận về kiến thức tìm hiểu được			
Tuần 4 2/9/2024- 8/9/2024	Tìm hiểu về Cassandra Thiết kế giao diện ứng dụng	Giao diện giỏ hàng	Giao diện trang hiển thị sản phẩm	Giao diện Lịch sử mua hàng	Giao diện đăng nhập, đăng ký
		Tìm hiểu về truy vấn CQL	Tìm hiểu về cách tạo bảng, key space	Tìm hiểu về cách tạo và sử dụng DBBear	Tìm hiểu và cài đặt DBBear
		Hợp nhóm bàn luận về kiến thức tìm hiểu được			
Tuần 5 9/9/2024- 15/9/2024	Thiết kế cơ sở dữ liệu Cassandra Hoàn thiện giao diện ứng dụng	Thiết kế câu truy vấn và bảng cho chức năng Giỏ hàng	Thiết kế câu truy vấn và bảng cho chức năng Giỏ hàng	Thiết kế câu truy vấn và bảng cho chức năng Giỏ hàng	Thiết kế câu truy vấn và bảng cho chức năng Giỏ hàng

Tuần 6 16/9/2024- 22/9/2024	<div> <div> Tìm hiểu và kết nối cơ sở dữ liệu cassandra </div> <div> <i>Code tính năng</i> Quản lý giỏ hàng Hiển thị menu loại, thương hiệu hiển thị sản phẩm theo thương hiệu (kèm phân trang) và sp mới tại trang chủ </div> </div>	<i>Code tính năng</i> Quản lý thương hiệu, loại hàng (kèm phân trang) Hiển thị tất cả sản phẩm tại trang cửa hàng Hiển thị sản phẩm theo loại	<i>Code tính năng</i> Quản lý đơn hàng (kèm phân trang) Đặt hàng, Hiển thị lịch sử mua hàng	<i>Code tính năng</i> Quản lý sản phẩm, người dùng (kèm phân trang, tìm kiếm, sắp xếp) Đăng nhập, đăng ký, phân quyền
Tuần 7 23/9/2024- 29/9/2024				
Tuần 8 30/9/2024- 6/10/2024	<div> <div>Hoàn thiện bản demo và báo cáo</div> <div> Trình bày word, powperpoint chương 3 </div> </div>	<div> <div>Trình bày word, powerpoint chương 1, 2</div> <div>Trình bày word, powerpoint chương 4</div> </div>	<div> <div>Trình bày word, powerpoint chương 5, 6</div> </div>	

Mục Lục

Lời Cảm Ơn.....	i
Bảng phân công tiến độ hàng tuần.....	ii
Mục Lục.....	iv
Lời mở đầu	viii
Chương 1. TỔNG QUAN ĐỀ TÀI	1
1.1. Tên Đề Tài	1
1.2. Lí Do Chọn Đề Tài	1
1.3. Mục Tiêu Của Đề Tài.....	1
1.4. Ý Nghĩa Của Đề Tài.....	1
1.5. Khảo sát hiện trạng.....	1
1.5.1. Giới thiệu về nơi khảo sát.....	1
1.5.2. Quy trình nghiệp vụ	2
1.5.3. Yêu cầu chức năng cho website.....	4
1.5.4. Yêu cầu phi chức năng cho website.....	5
Chương 2. GIỚI THIỆU VỀ CASSANDRA VÀ CÔNG NGHỆ SỬ DỤNG.....	6
2.1. Cassandra	6
2.1.1. Cassandra là gì?	6
2.1.2. Đặc điểm của Cassandra	6
2.1.3. Mô hình dữ liệu trong cassandra.....	7
2.2. Các công nghệ sử dụng trong đề án	8
2.2.1. DbeaverEE	8
2.2.1.1. DBeaverEE là gì?.....	8
2.2.1.2. Ưu và nhược điểm của DBeaver.....	8
Chương 3. PHÂN TÍCH CƠ SỞ DỮ LIỆU.....	10
3.1. Sơ đồ thiết kế cơ sở dữ liệu	10
3.2. Thiết kế câu truy vấn và các bảng.....	10

3.2.1. Bảng product.....	10
3.2.3. Bảng Product_by_brand	12
3.2.4. Bảng product_by_category.....	12
Bảng user.....	13
3.2.5. Bảng Cart_by_user	15
3.2.6. Bảng Orders.....	17
3.2.7. Bảng Order_details.....	19
3.2.8. Bảng Category	20
3.2.9. Bảng Brand.....	22
Chương 4. XÂY DỰNG WEBSITE	25
4.1. Hàm code các chức năng tương ứng có dùng CQL	25
4.1.1. Chức năng quản lý sản phẩm.....	25
4.1.1.1. Hiện thị danh sách sản phẩm	25
4.1.1.2. Thêm mới sản phẩm.....	26
4.1.1.3. Sửa sản phẩm	26
4.1.1.4. Xóa sản phẩm.....	27
4.1.1.5. Xem chi tiết sản phẩm	28
4.1.2. Quản lý người dùng	29
4.1.2.1. Hiện thị danh sách người dùng	29
4.1.2.2. Thêm người dùng mới	30
4.1.2.3. Sửa thông tin người dùng.....	31
4.1.2.4. Xóa người dùng	32
4.1.3. Chức năng hiển thị sản phẩm theo thương hiệu.....	33
4.1.4. Chức năng giỏ hàng	34
4.1.4.1. Hiện thị sản phẩm trong giỏ hàng	34
4.1.4.2. Đếm số lượng các dòng sản phẩm trong giỏ hàng.....	35
4.1.4.3. Thêm sản phẩm vào giỏ hàng	36

4.1.4.4.	Tăng số lượng 1 sản phẩm trong giỏ hàng.....	36
4.1.4.5.	Giảm số lượng sản phẩm trong giỏ hàng	37
4.1.4.6.	Xóa 1 sản phẩm trong giỏ hàng	38
4.1.4.7.	Xóa tất cả sản phẩm trong giỏ hàng.....	38
4.1.5.	Chức năng thanh toán	39
4.1.6.	Chức năng hiển thị lịch sử đơn hàng của người dùng	41
4.1.7.	Chức năng quản lý đơn hàng	42
4.1.7.1.	Hiển thị danh sách các đơn hàng (cùng với tổng doanh thu, tổng số đơn hàng ...)	42
4.1.7.2.	Cập nhật trạng thái 1 đơn hàng của 1 người dùng	44
4.1.8.	Chức năng quản lý loại sản phẩm	45
4.1.8.1.	Hiển thị danh sách các loại sản phẩm	45
4.1.8.2.	Thêm một loại sản phẩm.....	46
4.1.8.3.	Cập nhật một loại sản phẩm.....	46
4.1.8.4.	Xóa một loại sản phẩm	47
4.1.9.	Chức năng quản lý thương hiệu	49
4.1.9.1.	Hiển thị danh sách các thương hiệu	49
4.1.9.2.	Thêm một thương hiệu sản phẩm	49
4.1.9.3.	Cập nhật một thương hiệu sản phẩm	50
4.1.9.4.	Xóa một thương hiệu sản phẩm	50
Chương 5.	CÀI ĐẶT.....	51
5.1.	Cài đặt.....	51
5.2.	Kết nối python django với cơ sở dữ liệu Cassandra.....	56
Chương 6:	KẾT LUẬN – HƯỚNG PHÁT TRIỂN	58
TÀI LIỆU THAM KHẢO		59

Lời mở đầu

Ngày nay, nhu cầu sử dụng của con người trong xã hội luôn là động lực lớn thúc đẩy sự phát triển sản xuất. Như chúng ta đã biết, việc thiếu thông tin khiến cho quá trình đưa sản phẩm đến tay người tiêu dùng trở nên khó khăn và gây ra nhiều hạn chế trong việc di chuyển. Tuy nhiên, sự phổ biến của Internet đã tạo ra một bước ngoặt mới trong định hướng phát triển Công nghệ Thông tin tại Việt Nam, cùng với đó là nhu cầu sử dụng các thiết bị để trao đổi, tìm hiểu thông tin ngày càng tăng. Để đáp ứng nhu cầu này, việc cung cấp thông tin tiện lợi và hỗ trợ mua bán diễn ra nhanh chóng, hiệu quả là vô cùng quan trọng. Bán hàng qua mạng đã trở thành một giải pháp tối ưu trong việc phân phối thông tin phục vụ mục đích thương mại.

Việc phải đi mua sắm từ cửa hàng này đến cửa hàng khác vừa mất thời gian, vừa tốn công sức, thì giờ đây, bạn có thể mua sắm trực tuyến qua các website, tìm kiếm sản phẩm một cách nhanh chóng, dễ dàng và tiết kiệm. Bất kỳ loại hàng hóa nào, từ bó hoa tươi đến chiếc điện thoại, đều có thể được đặt mua qua Internet. Trong hoạt động sản xuất và kinh doanh, thương mại điện tử đã khẳng định vai trò quan trọng trong việc thúc đẩy và phát triển doanh nghiệp. Đối với một cửa hàng hoặc shop, việc quảng bá và giới thiệu các sản phẩm mới để đáp ứng nhu cầu khách hàng là vô cùng cần thiết. Và để quảng bá hiệu quả, xây dựng một website cho cửa hàng của mình là điều không thể thiếu, giúp người tiêu dùng biết đến tất cả các sản phẩm mà cửa hàng cung cấp.

Do đó, việc xây dựng “Website mua bán mỹ phẩm” là một vấn đề thực tế, ứng dụng và có tiềm năng phát triển trong tương lai, đặc biệt khi thị trường làm đẹp đang thu hút sự quan tâm của đông đảo người tiêu dùng.

Với lý do này, dưới sự hướng dẫn tận tình của cô **Nguyễn Thị Thu Tâm**, chúng em đã chọn đề tài “Website mua bán mỹ phẩm bằng hệ quản trị cơ sở dữ liệu Cassandra” để có thể vận dụng những kiến thức cô đã dạy nhằm xây dựng một trang web hoàn chỉnh, hỗ trợ lưu trữ và xử lý dữ liệu lớn hiệu quả.

Chương 1. TỔNG QUAN ĐỀ TÀI

1.1. Tên Đề Tài

Website mua bán mỹ phẩm Moon Shop

1.2. Lí Do Chọn Đề Tài

Độ phổ biến của ngành công nghiệp mỹ phẩm: Ngành công nghiệp mỹ phẩm là một trong những lĩnh vực kinh doanh phát triển nhanh chóng, với sự tăng trưởng đáng kể trong một số năm gần đây. Nghiên cứu về website mỹ phẩm có thể mang lại những thông tin quan trọng về xu hướng thị trường, đối thủ cạnh tranh, và những cơ hội mới.

Xu hướng mua sắm trực tuyến: Ngày càng nhiều người tiêu dùng chọn mua sắm mỹ phẩm trực tuyến. Nghiên cứu về các yếu tố ảnh hưởng đến quyết định mua sắm trực tuyến trong lĩnh vực này có thể làm sáng tỏ những chiến lược hiệu quả để thu hút và giữ chân khách hàng.

1.3. Mục Tiêu Của Đề Tài

Đề tài này tập trung vào việc xây dựng một hệ thống hỗ trợ bán hàng trực tuyến trên nền tảng Internet. Mục tiêu chính là tạo ra một môi trường mua bán trực tuyến nhanh chóng, chính xác và thuận tiện cho người dùng. Điều này giúp việc tối ưu hóa tốc độ giao dịch để đảm bảo hiệu suất, cung cấp thông tin chính xác và tạo trải nghiệm mua sắm dễ dàng, tiện lợi. Mục đích cuối cùng của đề tài là tạo ra một hệ thống linh hoạt và hiệu quả, hỗ trợ cả quá trình bán và mua hàng một cách thông minh và thuận tiện cho người dùng trên Internet.

1.4. Ý Nghĩa Của Đề Tài

Đề tài này nhằm tạo ra một hệ thống mua bán trực tuyến giúp khách hàng tiết kiệm thời gian và chi phí mua hàng. Đồng thời, nó cung cấp thông tin quan trọng cho chủ sở hữu để họ có thể đạt được mục tiêu kinh doanh và xây dựng chiến lược marketing linh hoạt và hiệu quả cho từng sản phẩm hay giai đoạn kinh doanh.

1.5. Khảo sát hiện trạng

1.5.1. Giới thiệu về nơi khảo sát

Moon Shop tại 287 Đ. Nguyễn Thị Thập, Quận 7, Hồ Chí Minh là một cửa hàng mỹ phẩm chuyên cung cấp các sản phẩm làm đẹp từ nhiều thương hiệu nổi tiếng trong và ngoài nước. Các mặt hàng tại đây bao gồm sản phẩm chăm sóc da, trang điểm, dưỡng tóc, và các phụ kiện làm đẹp, luôn đảm bảo chất lượng và đa dạng. Ngoài việc kinh doanh trực tiếp tại

cửa hàng, Moon Shop còn mở rộng hoạt động bán hàng online trên các sàn thương mại điện tử như website chính thức của cửa hàng, Shopee, Lazada và nhiều nền tảng khác.



Hình 1. 1 Hình ảnh nơi khảo sát

Cơ cấu tổ chức của cửa hàng:

Ban quản lý cửa hàng: Đây là cấp quản lý cao nhất, bao gồm các nhà đầu tư và điều hành hoạt động website. Họ chỉ quan tâm đến báo cáo tổng hợp, doanh thu và chiến lược phát triển, không can thiệp vào công việc hàng ngày của nhân viên.

Nhân viên kho: Kiểm tra và quản lý hàng hóa trong kho, đối chiếu số liệu thực tế với hệ thống, xử lý nhập hàng từ nhà cung cấp, sắp xếp, đóng gói và chuẩn bị hàng cho đơn vị giao hàng.

Nhân viên bán hàng: Quản lý số liệu hàng hóa, thống kê hàng tồn, liên hệ nhà cung cấp đặt hàng mới, xác nhận đơn hàng và phối hợp với nhân viên kho để xử lý và gửi hàng cho khách.

1.5.2. Quy trình nghiệp vụ

a. Tổng quan các nghiệp vụ:

Hàng tuần, nhân viên tiến hành thống kê số liệu hàng hóa để kiểm tra tình trạng thực tế tại kho. Sau quá trình kiểm kho, nếu phát hiện thiếu hụt hoặc các mặt hàng có lượng tồn kho dưới mức an toàn, nhân viên sẽ nhập hàng từ nhà cung cấp. Khi hàng được giao đến,

nhân viên kiểm tra số lượng và chất lượng sản phẩm theo yêu cầu. Nếu đạt yêu cầu, nhân viên thực hiện nhập hàng vào kho, cập nhật hàng lên hệ thống và đăng tải lên website.

Khách hàng có thể đăng ký tài khoản, tìm kiếm sản phẩm và đặt mua hàng. Nếu khách chọn thanh toán trực tuyến, họ phải hoàn tất thanh toán mới được xác nhận đơn hàng. Đầu giờ sáng hoặc đầu giờ chiều, nhân viên sẽ dựa vào các hóa đơn và phiếu xuất kho để chuẩn bị hàng. Hàng hóa được kiểm tra lần cuối, đóng gói theo đúng quy cách, và gửi đến đơn vị vận chuyển để tiến hành giao hàng đến tay khách hàng. Sau khi nhận hàng, khách hàng có thể truy cập vào tài khoản của mình để lại đánh giá về sản phẩm.

b. Quy trình xử lý nghiệp vụ:

Qua quá trình khảo sát và phân tích, nhóm xác định quy trình xử lý từng nghiệp vụ chính cho hoạt động bán mỹ phẩm online như sau:

Quy trình kiểm kho: Hàng tuần, phía nhân viên bán hàng sẽ thống kê và lập danh sách sản phẩm kèm số liệu hàng hóa trên hệ thống đưa cho nhân viên kho. Dựa vào dữ liệu trên hệ thống đó và các phiếu xuất kho trong tuần để tiến hành kiểm tra, đối soát số lượng hàng hóa thực tế. Sau khi hoàn tất kiểm kê, số liệu được nhân viên kho ghi nhận và gửi báo cáo kết quả kiểm kê đến Ban quản lý cửa hàng. Báo cáo này bao gồm tình trạng hàng hóa, số lượng, và các đề xuất xử lý để điều chỉnh số liệu tồn kho. Nhân viên kho sau đó sẽ thực hiện các biện pháp điều chỉnh cần thiết để đảm bảo độ chính xác của số liệu kho và cải thiện quy trình quản lý hàng hóa trong tương lai.

Quy trình nhập hàng từ nhà cung cấp: Nghiệp vụ nhập hàng từ nhà cung cấp bắt đầu với việc nhân viên bán hàng dựa theo kết quả kiểm kê kho, xác định nhu cầu và lập yêu cầu đặt hàng, bao gồm thông tin sản phẩm, số lượng và gửi đến nhà cung cấp. Sau đó, nhà cung cấp xác nhận đơn hàng, gửi phản hồi về thời gian giao hàng, giá cả. Khi hàng được giao đến cửa hàng, nhân viên kho sẽ kiểm tra số lượng và chất lượng để đảm bảo đúng với yêu cầu. Nếu đạt yêu cầu, hàng hóa sẽ được đưa vào kho. Nhân viên bán hàng thực hiện thanh toán hóa đơn nhập hàng theo thỏa thuận, sau đó cập nhật thông tin hàng hóa lên hệ thống và đăng tải sản phẩm lên trang web.

Quy trình đăng ký tài khoản: Khi khách hàng nhấn vào nút "Đăng ký" trên trang chủ. Khách hàng sẽ được yêu cầu điền các thông tin cá nhân như họ tên, email, số điện thoại và mật khẩu. Để đảm bảo bảo mật, hệ thống sẽ yêu cầu nhập lại mật khẩu. Sau khi nhập đầy đủ thông tin, khách hàng nhấn "Hoàn tất đăng ký". Hệ thống sẽ kiểm tra tính hợp lệ của thông tin và gửi một email chứa mã xác nhận để kích hoạt. Sau khi xác nhận thành công, tài

khoản của khách hàng sẽ được kích hoạt, cho phép họ đăng nhập và sử dụng các chức năng mua sắm, theo dõi đơn hàng, và nhận thông tin khuyến mãi.

Quy trình tìm kiếm sản phẩm: Khi khách hàng truy cập vào trang chủ và nhập từ khóa liên quan đến sản phẩm mà họ muốn tìm, chẳng hạn như tên sản phẩm, thương hiệu, hoặc đặc tính cụ thể (son, kem chống nắng,...). Hệ thống tìm kiếm sẽ tiếp nhận từ khóa và thực hiện việc tìm kiếm trong cơ sở dữ liệu, dựa trên các thông tin như tên sản phẩm, danh mục, hoặc các từ khóa liên quan. Sau đó, hệ thống sẽ hiển thị danh sách các sản phẩm phù hợp với từ khóa mà khách hàng đã nhập, bao gồm các thông tin như tên sản phẩm, hình ảnh, giá cả, và các tùy chọn mua hàng. Khách hàng có thể xem chi tiết sản phẩm, so sánh các mặt hàng khác nhau để lựa chọn sản phẩm phù hợp.

Quy trình mua hàng online của khách hàng: Quy trình mua hàng online trên website bắt đầu khi khách hàng chọn sản phẩm và thêm vào giỏ hàng. Khi khách hàng xác nhận mua hàng, họ sẽ nhập thông tin cá nhân, địa chỉ giao hàng và lựa chọn phương thức thanh toán. Nếu chọn thanh toán trực tuyến, khách hàng phải hoàn tất quy trình thanh toán. Sau đó nhân viên bán hàng sẽ xác nhận đơn hàng và in hóa đơn bán hàng, để vào đầu giờ sáng hoặc đầu giờ chiều có thể tổng hợp và gửi các đơn hàng này đến bộ phận kho nhằm thực hiện quy trình giao hàng.

Quy trình thanh toán của khách hàng: Quy trình thanh toán bắt đầu khi khách hàng xác nhận đơn hàng và chọn phương thức thanh toán như thẻ ngân hàng, ví điện tử, hoặc thanh toán khi nhận hàng. Hệ thống sau đó tính toán phí vận chuyển và hiển thị tổng số tiền cần thanh toán. Khách hàng kiểm tra lại thông tin đơn hàng và xác nhận lần cuối. Đối với phương thức thanh toán trực tuyến, khách hàng phải hoàn tất quy trình thanh toán trước mới được xác nhận đơn hàng. Cuối cùng, nhân viên xác nhận đơn hàng, khách hàng sẽ nhận được email thông báo tương ứng.

Quy trình giao hàng: Đầu giờ sáng hoặc đầu giờ chiều, nhân viên bán hàng sẽ tạo phiếu xuất kho gồm các sản phẩm kèm số lượng cần thiết và các hóa đơn để chuyển cho nhân viên kho. Nhân viên kho kiểm tra và chuẩn bị hàng hóa, đóng gói đúng tiêu chuẩn, dán nhãn và tạo thẻ vận chuyển rồi gửi đến đơn vị vận chuyển. Cuối cùng, sau khi giao hàng hoàn tất, trạng thái đơn hàng được cập nhật thành “Hoàn Thành”.

1.5.3. Yêu cầu chức năng cho website

Đăng tải sản phẩm: Cho phép quản trị viên đăng sản phẩm với thông tin chi tiết, hình ảnh, giá cả, và khuyến mãi.

Quản lý giỏ hàng: Khách hàng có thể thêm, xóa, và cập nhật số lượng sản phẩm trong giỏ hàng.

Thanh toán trực tuyến: Hỗ trợ các phương thức thanh toán như thẻ ngân hàng, ví điện tử, và thanh toán khi nhận hàng (COD).

Quản lý đơn hàng: Hệ thống lưu trữ, theo dõi, và cập nhật trạng thái đơn hàng, cho phép khách hàng kiểm tra tiến trình giao hàng.

Đăng nhập/Đăng ký: Cho phép khách hàng và quản trị viên tạo tài khoản, đăng nhập, và quản lý thông tin cá nhân.

Tìm kiếm và lọc sản phẩm: Khách hàng có thể tìm kiếm và lọc sản phẩm theo danh mục, giá cả, và đánh giá.

Quản lý quyền truy cập: Phân quyền truy cập cho nhân viên dựa trên vai trò, chức vụ, giới hạn quyền thực hiện các tác vụ của bộ phận khác.

1.5.4. Yêu cầu phi chức năng cho website

Hệ thống cần đáp ứng các yêu cầu về hoạt động, hiệu suất và bảo mật để đảm bảo tính hiệu quả và an toàn như sau:

Hệ thống cần dễ mở rộng để hỗ trợ sự tăng trưởng về số lượng sản phẩm, người dùng và giao dịch.

Bảo mật đòi hỏi mỗi tài khoản nhân viên phải được cấp quyền truy cập phù hợp với chức vụ, không thể thực hiện nhiệm vụ của bộ phận khác

Chương 2. GIỚI THIỆU VỀ CASSANDRA VÀ CÔNG NGHỆ SỬ DỤNG

2.1. Cassandra

2.1.1. Cassandra là gì?

Cassandra là NoSQL, được phát triển bởi Facebook vào năm 2007. Sau đó nó được tặng cho quỹ Apache vào 2/2010 và nâng cấp lên thành dự án hàng đầu của Apache. Ngôn ngữ phát triển Cassandra là Java.

Cassandra là hệ cơ sở dữ liệu phân tán, dữ liệu được lưu trữ trên nhiều node của nhiều máy khác nhau, dạng Column-Family theo cơ chế P2P. Cassandra cung cấp hỗ trợ cho các clusters trên data centers. Hiệu năng xử lý của hệ thống cũng tăng theo số node (nếu càng nhiều node thì càng xử lý được nhiều request). Điều đó sẽ giúp cho Cassandra dễ dàng scale theo chiều ngang.

Cassandra hiện nay được rất nhiều tổ chức công nghệ hàng đầu thế giới sử dụng như: Spotify, Netflix, ebay, Github, Discord,...

2.1.2. Đặc điểm của Cassandra

Một số đặc tính nổi bật của Cassandra

- Scalability: Khả năng mở rộng cao
- Availability: Tính sẵn sàng cao
- Reliability: Độ tin cậy cao
- Distributed: Tính phân tán
- Masterless: Không sử dụng cơ chế Master-Slave mà các node đều đóng vai trò như nhau trong xử lý dữ liệu (P2P).
- Cassandra có khả năng mở rộng đàn hồi. Cluster có thể dễ dàng thu nhỏ, mở rộng.

Các thành phần chính:

- Node: Là một thực thể xử lý và lưu trữ dữ liệu. Một server có thể cài nhiều node nhưng bất lợi vì khi server bị down sẽ gây ảnh hưởng nhiều hơn 1 node.
- Rack: Là chỉ một nhóm các node có chung nguy cơ bị lỗi. Các node này là các server cài chung một rack thì sẽ có chung nguy cơ khi rack đó bị sập (nguồn/mạng...)
- Datacenter: Là một tập hợp các node mà chứa đủ thông tin dữ liệu. Các node sẽ xử lý một phần của dữ liệu, tập các node xử lý đủ 100% dữ liệu gọi là một datacenter.

- Cluster: Là thành phần mà có thể chứa một hoặc nhiều datacenter.
- Commitlog: Là cơ chế để xử lý khôi phục sự cố (crash-recovery mechanism). Mọi thao tác ghi dữ liệu (write operation) đều được ghi vào commitlog.

2.1.3. Mô hình dữ liệu trong cassandra

- Trong Cassandra, một cơ sở dữ liệu (keyspace) được tổ chức dưới dạng cột (Column Family). Trong đó dữ liệu được lưu trữ trong các bảng (được gọi là "column families"). Mỗi bảng chứa các hàng được nhận diện bởi một khóa chính (primary key) duy nhất.

- Keyspace trong Cassandra là một thành phần quan trọng trong mô hình dữ liệu, tương đương với database trong các hệ quản trị cơ sở dữ liệu quan hệ. Khi tạo keyspace, cần định nghĩa chiến lược sao chép (replication strategy) và hệ số sao chép (replication factor).

- Khóa chính (Primary Key):

+ **Khóa phân vùng (Partition Key):** Khóa phân vùng quyết định cách dữ liệu được phân phối trên các node (nút) trong hệ thống. Nó giúp phân phối đều dữ liệu trên các node để tối ưu hóa hiệu suất.

+ **Khóa phân cụm (Clustering Key):** Khóa phân cụm xác định thứ tự lưu trữ của dữ liệu bên trong mỗi phân vùng, cho phép thực hiện các truy vấn phạm vi hiệu quả.

+ **Composite/Compound key** là key được tạo nên bởi nhiều hơn một field.

- Sao chép và phân mảnh dữ liệu (Replication and Partitioning): Dữ liệu trong Cassandra được phân mảnh và lưu trữ trên nhiều node thông qua khóa phân vùng.

- Không chuẩn hóa (Denormalization): Cassandra khuyến khích việc không chuẩn hóa dữ liệu (denormalization) để tối ưu hóa hiệu suất đọc. Dữ liệu có thể được lặp lại ở nhiều bảng để phục vụ các truy vấn khác nhau mà không cần phải nối bảng (join).

- Tối ưu hóa theo mô hình truy vấn: Cassandra được thiết kế dựa trên các mô hình truy vấn. Khi tạo mô hình dữ liệu, người thiết kế cần dựa vào các truy vấn cụ thể để đảm bảo hiệu suất đọc và ghi.

- Không hỗ trợ Join và các phép tổng hợp: Cassandra không hỗ trợ các phép nối (join) giữa các bảng hoặc các phép toán tổng hợp (aggregation). Điều này yêu cầu phải thiết kế dữ liệu cẩn thận ngay từ đầu.

- Chỉ mục và các bảng materialized view: Cassandra hỗ trợ chỉ mục phụ (secondary index) và bảng nhìn vật lý (materialized views) để tối ưu hóa cho các truy vấn nhất định.

2.2. Các công nghệ sử dụng trong đồ án

2.2.1. DbeaverEE

2.2.1.1. DBeaverEE là gì?

DBeaver là một công cụ quản lý và phát triển cơ sở dữ liệu mạnh mẽ, hỗ trợ nhiều hệ quản trị cơ sở dữ liệu như Cassandra, MySQL, MariaDB, PostgreSQL, SQLite, Redis, và nhiều hơn nữa. DBeaver cung cấp một giao diện người dùng phong phú và thân thiện, giúp việc quản lý cơ sở dữ liệu trở nên hiệu quả và tiện lợi hơn.

DBeaver là một công cụ đa nền tảng và hoạt động trên các nền tảng được Eclipse hỗ trợ (Windows, Linux, MacOS X, Solaris). DBeaver có nhiều phiên bản để phù hợp với nhiều nhu cầu khác nhau của người dùng:

- Phiên bản Community Edition (CE) là phiên bản đầu tiên của DBeaver. Phiên bản này được phát hành vào năm 2010 và trở thành mã nguồn mở (GPL) vào năm 2011. Hỗ trợ tương đối nhiều hệ quản trị cơ sở dữ liệu phổ biến nhưng hạn chế là chỉ hoạt động trong môi trường Windows.

- Phiên bản Plugin Eclipse: Phiên bản này được sử dụng bởi các lập trình viên sử dụng Eclipse IDE để phát triển phần mềm và cần một công cụ quản lý cơ sở dữ liệu ngay trong IDE của họ. Plugin Eclipse bao gồm hầu hết các tính năng của Community Edition.

- Phiên bản Enterprise Edition: DBeaver 3.x đã công bố hỗ trợ cơ sở dữ liệu NoSQL (Cassandra và MongoDB trong phiên bản đầu tiên). Kể từ đó, DBeaver đã chia thành các phiên bản Community và Enterprise. Enterprise Edition hỗ trợ cơ sở dữ liệu NoSQL, trình quản lý truy vấn liên tục và một số tính năng cấp doanh nghiệp khác. Phiên bản EE không phải là mã nguồn mở và yêu cầu phải mua giấy phép (có thể tạo giấy phép dùng thử miễn phí trong vòng 14 ngày).

2.2.1.2. Ưu và nhược điểm của DBeaver

Công dụng:

- Quản lý cơ sở dữ liệu: DBeaver cho phép người dùng kết nối và quản lý nhiều cơ sở dữ liệu khác nhau trong cùng một giao diện.

- Truy vấn dữ liệu: Hỗ trợ viết và chạy các câu lệnh SQL với gợi ý cú pháp, hỗ trợ phân tích và tối ưu hóa truy vấn.

- Thiết kế và chỉnh sửa bảng: DBeaver cung cấp công cụ thiết kế sơ đồ cơ sở dữ liệu, giúp người dùng dễ dàng chỉnh sửa cấu trúc bảng mà không cần viết mã SQL phức tạp.

- Xuất và nhập dữ liệu: Công cụ hỗ trợ xuất dữ liệu ra nhiều định dạng như CSV, Excel, HTML hoặc JSON, cũng như nhập dữ liệu từ các định dạng này vào cơ sở dữ liệu.

Ưu điểm:

- Hỗ trợ nhiều cơ sở dữ liệu: DBeaver hỗ trợ nhiều hệ quản trị cơ sở dữ liệu, làm cho nó trở thành công cụ đa năng cho các nhà phát triển.

- Thân thiện với người dùng: Cung cấp giao diện đồ họa dễ sử dụng, giúp người dùng dễ dàng thực hiện các thao tác cơ sở dữ liệu.

- Hiệu suất cao: DBeaver chú trọng tối ưu hóa hiệu suất, có thể xử lý dữ liệu quy mô lớn một cách nhanh chóng.

- Hỗ trợ đa nền tảng: DBeaver hỗ trợ các hệ điều hành Windows, macOS và Linux.

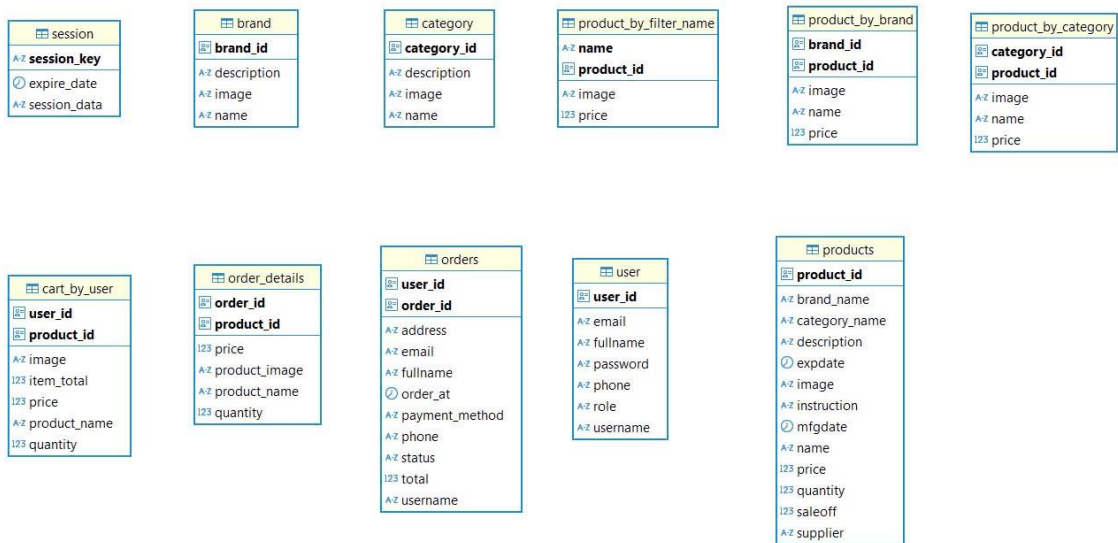
Nhược điểm:

- Tốc độ: DBeaver có thể chậm trong một số trường hợp khi xử lý lượng dữ liệu lớn.

- Tính năng hạn chế: Một số tính năng phức tạp có thể không được hỗ trợ hoặc cần thêm công cụ bổ sung khi sử dụng phiên bản cộng đồng mã nguồn mở.

Chương 3. PHÂN TÍCH CƠ SỞ DỮ LIỆU

3.1. Sơ đồ thiết kế cơ sở dữ liệu

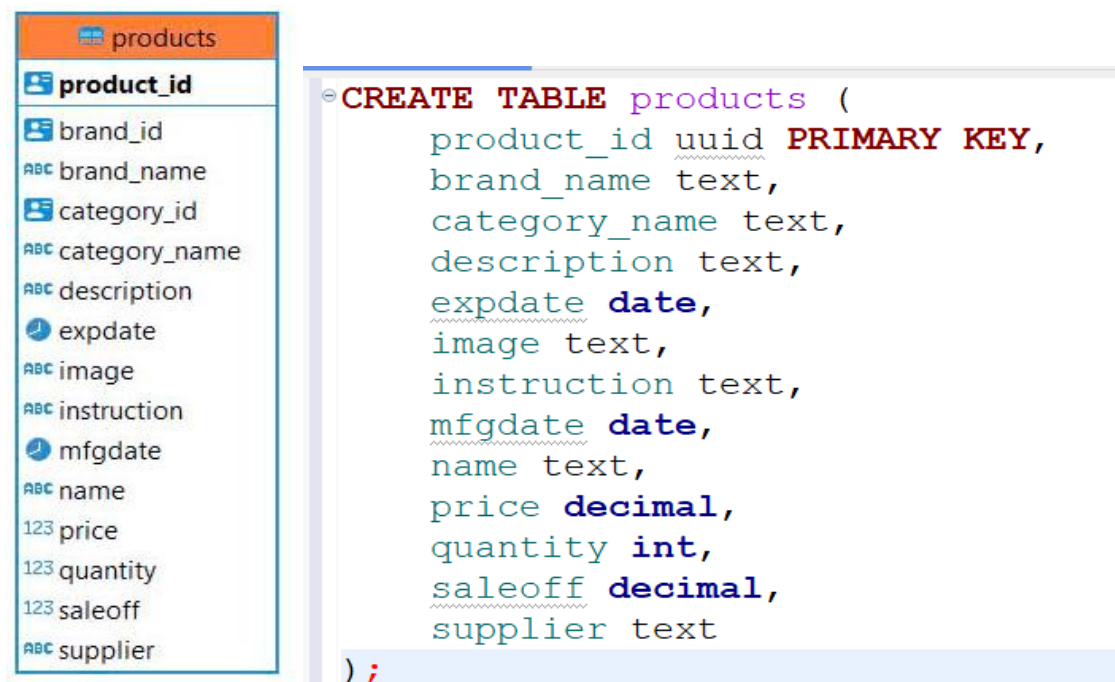


Hình 3. 1: Sơ đồ thiết kế cơ sở dữ liệu cassandra

3.2. Thiết kế câu truy vấn và các bảng

3.2.1. Bảng product

* **Câu truy vấn:** Cho biết tất cả sản phẩm tại cửa hàng



Hình 3. 2: Bảng dữ liệu sản phẩm

Khóa phân vùng là product_id

SELECT * FROM website_mypham.products;							
Results 1							
SELECT * FROM website_mypham.products							
	product_id	brand_name	category_name	description	expdate	image	instruction
1	22a68e4f-1864-4e5b-a1f8-c7302960ca9	L'Oréal	Kem dưỡng da	Dưỡng trắng, cấp ẩm cho da	2023-11-30	kem-duong-da-loreal.png	Thoa đều lên da mặt
2	96860ece-8cac-41ae-9e72-bed2358684d0	Chanel	Son môi	a	2024-11-02	/static/img/son-tab-ui-2.png	a
3	c2bfe817-c32e-4944-9927-7a927bd29459	MAC	Phấn nền	Phấn nền mịn, bám lâu	2024-08-01	phan-nen-mac.png	Dùng cơ hoặc bông phấn tán đều
4	b354c9fb-a142-407b-8224-01a5b9c3ef17	Chanel	Kem dưỡng da	Chống nắng hiệu quả, không gây nhờn	2024-05-01		Thoa lên da trước khi ra nắng
5	c0da56bc-b485-44c1-8954-720e249c688d	Chanel	Son môi	Son môi lâu trôi, mịn màng	2024-12-31	son-chanel.png	Thoa trực tiếp lên môi
6	0ba36c58-bbc2-4b73-b9e5-a3d33e224424	aa	a	a	[NULL]	[NULL]	a

Hình 3. 3. Xem tất cả sản phẩm

* Các lệnh khác:

Thêm mới một sản phẩm (Dùng cho tính năng thêm mới 1 sản phẩm)

INSERT INTO products(product_id, brand_name, category_name, description, expdate, image, instruction, m							
VALUES (uuid(), 'L'Oréal', 'Skincare', 'A hydrating face cream for daily use', '2025-12-31', 'loreal_fac							
products							
	product_id	brand_name	category_name	description	expdate	image	instruction
1	22a68e4f-1864-4e5b-a1f8-c7302960ca9	L'Oréal	Kem dưỡng da	Dưỡng trắng, cấp ẩm cho da	2023-11-30	kem-duong-da-loreal.png	Thoa đều lên da mặt
2	96860ece-8cac-41ae-9e72-bed2358684d0	Chanel	Son môi	a	2024-11-02	/static/img/son-tab-ui-2.png	a
3	c2bfe817-c32e-4944-9927-7a927bd29459	MAC	Phấn nền	Phấn nền mịn, bám lâu	2024-08-01	phan-nen-mac.png	Dùng cơ hoặc bông phấn tán đều
4	dc1043cb-ba6a-4310-8735-456c843ba3a6	L'Oréal	Skincare	A hydrating face cream for daily use	2025-12-31	loreal_facecream.jpg	Apply a small amount to face twic

Hình 3. 4: Thêm mới một sản phẩm

Cập nhật thông tin sản phẩm (Dùng cho tính năng cập nhật sản phẩm)

UPDATE products	
SET	
	price = 17.99,
	quantity = 120,
	saleoff = 0.20,
	description = 'Updated hydrating face cream with improved formula'
WHERE	product_id = dc1043cb-ba6a-4310-8735-456c843ba3a6;
Name	Value
Updated Rows	-1
Query	UPDATE products
	SET
	price = 17.99,
	quantity = 120,
	saleoff = 0.20,
	description = 'Updated hydrating face cream with improved formula'
	WHERE product_id = dc1043cb-ba6a-4310-8735-456c843ba3a6
Start time	Sun Oct 06 08:25:50 ICT 2024
Finish time	Sun Oct 06 08:25:50 ICT 2024

Hình 3. 5: Cập nhật sản phẩm

Xóa thông tin sản phẩm (Dùng cho tính năng xóa sản phẩm)

```
DELETE FROM website_mypham.products
WHERE product_id = dc1043cb-ba6a-4310-8735-456c843ba3a6;
```

Statistics 1	
Name	Value
Updated Rows	-1
Query	DELETE FROM website_mypham.products WHERE product_id = dc1043cb-ba6a-4310-8735-456c843ba3a6
Start time	Sun Oct 06 08:28:46 ICT 2024
Finish time	Sun Oct 06 08:28:46 ICT 2024

Hình 3. 6: Xóa sản phẩm

3.2.3. Bảng Product_by_brand

Câu truy vấn: Cho biết n (số lượng) sản phẩm thuộc 1 loại thương hiệu cho trước

(Dùng cho tính năng lọc sản phẩm theo cùng 1 thương hiệu ở menu trang chủ)

product_by_brand	CREATE TABLE product_by_brand (
brand_id	brand_id UUID,
product_id	product_id UUID,
image	name TEXT,
name	price DECIMAL,
price	image TEXT,
	PRIMARY KEY (brand_id, product_id)
);

Hình 3. 7: Bảng dữ liệu sản phẩm theo thương hiệu

Khóa phân vùng là brand_id

Khóa phân cụm là product_id

```
SELECT * FROM product_by_brand WHERE brand_id = 9b65230b-6d40-40e4-9f20-4849dbf15fcb LIMIT 5
```

brand_id	product_id	image	name	price
9b65230b-6d40-40e4-9f20-4849dbf15fcb	cb3df60a-697b-420e-b9c8-3ff4bf53aec5	son_moi_rouge_coco.jpg	Rouge Coco	850,000

Hình 3. 8: Lọc sản phẩm theo 1 thương hiệu trên menu

3.2.4. Bảng product_by_category

Câu truy vấn: Cho biết n (số lượng) sản phẩm thuộc 1 loại cho trước

(Dùng cho tính năng lọc sản phẩm theo cùng 1 loại ở menu trang chủ)

product_by_category	CREATE TABLE website_mypham.product_by_category (
category_id	category_id uuid,
product_id	product_id uuid,
image	image text,
name	name text,
price	price decimal,
	PRIMARY KEY (category_id, product_id)

Hình 3. 9: Bảng dữ liệu sản phẩm theo loại

Khóa phân vùng là category_id

Khóa phân cụm là product_id

SELECT * FROM product_by_category WHERE category_id = 2a14f89a-49a2-4045-b6f2-ceb11142521c LIMIT 5;	
product_by_category 1 x	
SELECT * FROM product_by_category \	Enter a SQL expression to filter results (use Ctrl+Space)
category_id	product_id
image	name
price	
1	2a14f89a-49a2-4045-b6f2-ceb1
	449e0dd6-8097-4596-a2a8-b77
	https://example.com/images/li
	Son đường môi tự nhiên
	150,000

Hình 3. 10: Lọc sản phẩm theo loại

Bảng user

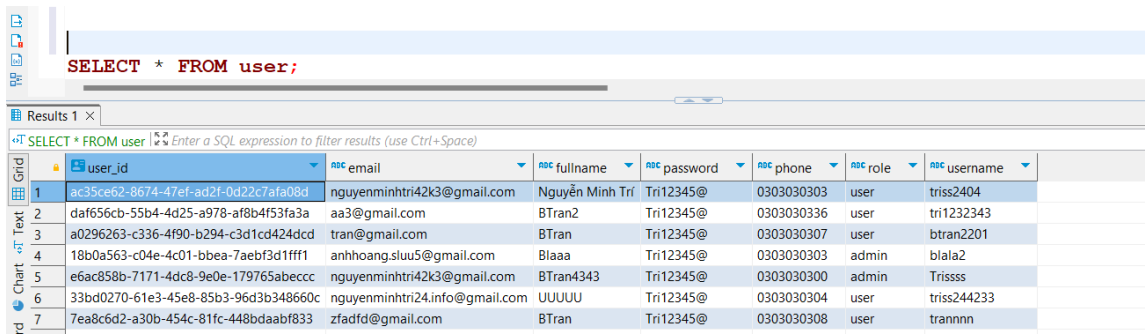
Câu truy vấn : Hiển thị danh sách user

(Dùng cho tính năng xem tất cả user)

user	CREATE TABLE user (
user_id	user_id uuid PRIMARY KEY,
email	email text,
fullname	fullname text,
password	password text,
phone	phone text,
role	role text,
username	username text
) ;

Hình 3. 11: Bảng dữ liệu người dùng

Khóa phân vùng là product_id



```
SELECT * FROM user;
```

	user_id	email	fullname	password	phone	role	username
1	ac35ce62-8674-47ef-ad2f-0d22c7afa08d	nguyenminhtri42k3@gmail.com	Nguyễn Minh Trí	Tri12345@	0303030303	user	triss2404
2	daf656cb-55b4-4d25-a978-af8b4f53fa3a	aa3@gmail.com	BTran2	Tri12345@	0303030336	user	tri1232343
3	a0296263-c336-4f90-b294-c3d1cd424dcd	tran@gmail.com	BTran	Tri12345@	0303030307	user	btran2201
4	18b0a563-c04e-4c01-bbea-7aebf3d1fff1	anhhoang.slou5@gmail.com	Blaaa	Tri12345@	0303030303	admin	blala2
5	e6ac858b-7171-4dc8-9e0e-179765abeccc	nguyenminhtri42k3@gmail.com	BTran4343	Tri12345@	0303030300	admin	Trissss
6	33bd0270-61e3-45e8-85b3-96d3b348660c	nguyenminhtri24.info@gmail.com	UUUUU	Tri12345@	0303030304	user	triss244233
7	7ea8c6d2-a30b-454c-81fc-448bdaabf833	zfadfd@gmail.com	BTran	Tri12345@	0303030308	user	trannnn

Hình 3. 12: Hiện thị danh sách người dùng

Thêm: Thêm mới một user

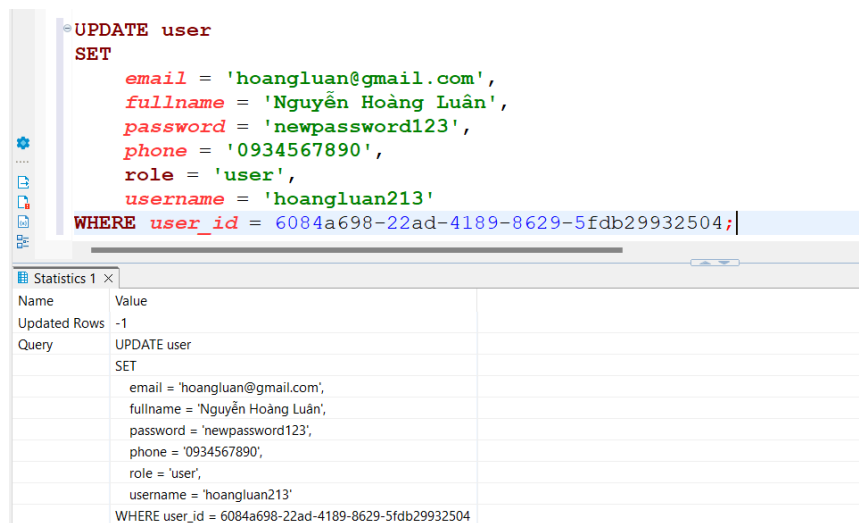
(Dùng cho tính năng thêm mới 1 user)

```
INSERT INTO user (user_id, email, fullname, password, phone, role, username)
VALUES (uuid(), 'nguyen.van.a@example.com', 'Nguyễn Văn A', 'password123', '0901234567', 'admin', 'nguye
```

Hình 3. 13: Thêm một người dùng

Cập nhật: Cập nhật thông tin user

(Dùng cho tính năng cập nhật thông tin user)



```
UPDATE user
SET
    email = 'hoangluan@gmail.com',
    fullname = 'Nguyễn Hoàng Luân',
    password = 'newpassword123',
    phone = '0934567890',
    role = 'user',
    username = 'hoangluan213'
WHERE user_id = 6084a698-22ad-4189-8629-5fdb29932504;
```

Name	Value
Updated Rows	-1
Query	UPDATE user
	SET
	email = 'hoangluan@gmail.com',
	fullname = 'Nguyễn Hoàng Luân',
	password = 'newpassword123',
	phone = '0934567890',
	role = 'user',
	username = 'hoangluan213'
	WHERE user_id = 6084a698-22ad-4189-8629-5fdb29932504

Hình 3. 14: Cập nhật thông tin người dùng

Xóa: Xóa thông tin user

(Dùng cho tính năng xóa sản phẩm)

<pre>DELETE FROM user WHERE user_id = 6084a698-22ad-4189-8629-5fdb29932504;</pre>	
Statistics 1	
Name	Value
Updated Rows	-1
Query	DELETE FROM user WHERE user_id = 6084a698-22ad-4189-8629-5fdb29932504
Start time	Sun Oct 06 08:52:23 ICT 2024
Finish time	Sun Oct 06 08:52:23 ICT 2024

Hình 3. 15: Xóa thông tin người dùng

Câu truy vấn: *Hiển thị thông tin chi tiết của một sản phẩm*

(Dùng cho tính năng xem chi tiết 1 sản phẩm)

Hình 3. 16: Hiển thị thông tin chi tiết một sản phẩm

3.2.5. Bảng Cart_by_user

Câu truy vấn: *Hiển thị tất cả sản phẩm trong giỏ hàng của 1 người dùng*

(Dùng cho tính năng hiển thị sản phẩm trong giỏ hàng khi người dùng click vào giỏ hàng)

<div> <div>cart_by_user</div> <div> <div>user_id</div> <div>product_id</div> <div>image</div> <div>item_total</div> <div>price</div> <div>product_name</div> <div>quantity</div> </div> </div>	<pre>CREATE table if not EXISTS cart_by_user (user_id UUID, product_id UUID, product_name text, image text, quantity int, price decimal, item_total decimal, PRIMARY KEY (user_id, product_id));</pre>
--	--

Hình 3. 17: Bảng dữ liệu giỏ hàng của người dùng

Khóa phân vùng là user_id

Khóa phân cụm là product_id

The screenshot shows a SQL query: `SELECT * FROM cart_by_user WHERE user_id = aaf48254-9013-4637-a965-1c8bfeb34eee`. The result is a table with columns: user_id, product_id, image, item_total, price, product_name, and quantity. It lists two items: 'kem-duong-da-loreal.png' and 'son-chanel.png'.

	user_id	product_id	image	item_total	price	product_name	quantity
1	aaf48254-9013-4637-a965-1c8bfeb34eee	a5bcb6d8-ac00-4245-92a6-deee72e376d1	kem-duong-da-loreal.png	450,000	450,000	Kem dưỡng trắng da White Perfect	1
2	aaf48254-9013-4637-a965-1c8bfeb34eee	deb98f80-6a53-4b8d-a0ee-9238642b444e	son-chanel.png	2,550,000	850,000	Son môi Rouge Coco	3

Hình 3. 18: Hiển thị tất cả sản phẩm trong giỏ hàng của người dùng

Câu truy vấn: Cho biết số lượng các dòng sản phẩm trong giỏ hàng của 1 người

(Dùng cho tính năng đếm số lượng dòng sản phẩm trong giỏ hàng → hiển thị lên biểu tượng giỏ hàng)

The screenshot shows a SQL query: `SELECT COUNT(*) FROM cart_by_user WHERE user_id = aaf48254-9013-4637-a965-1c8bfeb34eee`. The result is a table with one column 'count' and one row with the value '2'.

count
2

Hình 3. 19: Đếm số lượng dòng trong giỏ hàng của người dùng

Câu truy vấn: Tính tổng tiền tất cả các mặt hàng trong giỏ hàng của 1 người

(Dùng cho tính năng tính tổng tiền toàn bộ giỏ hàng của 1 người)

The screenshot shows a SQL query: `SELECT SUM(item_total) AS total_price FROM cart_by_user WHERE user_id = aaf48254-9013-4637-a965-1c8bfeb34eee`. The result is a table with one column 'total_price' and one row with the value '3,000,000'.

total_price
3,000,000

Hình 3. 20: Tổng tiền tất cả các mặt hàng trong giỏ hàng người dùng

Thêm: Thêm 1 sản phẩm vào giỏ hàng của 1 người dùng

(Dùng cho tính năng thêm 1 sản phẩm vào giỏ hàng)

The screenshot shows a SQL query: `INSERT INTO cart_by_user (user_id, product_id, product_name, quantity, price, image, item_total) VALUES (aaf48254-9013-4637-a965-1c8bfeb34eee, deb98f80-6a53-4b8d-a0ee-9238642b444e, 'Sữa rửa mặt Dove', 1, 80000, 'image/h1.jpg', 80000)`. Below the query, a 'Statistics' window shows the execution details.

Name	Value
Updated Rows	-1
Query	INSERT INTO cart_by_user (user_id, product_id, product_name, quantity, price, image, item_total) VALUES (aaf48254-9013-4637-a965-1c8bfeb34eee, deb98f80-6a53-4b8d-a0ee-9238642b444e, 'Sữa rửa mặt Dove', 1, 80000, 'image/h1.jpg', 80000)
Start time	Sat Oct 05 21:38:16 ICT 2024
Finish time	Sat Oct 05 21:38:16 ICT 2024

Hình 3. 21: Thêm một sản phẩm vào giỏ hàng của người dùng

Sửa: Sửa số lượng 1 sản phẩm trong giỏ hàng của 1 người dùng

(Dùng cho tính năng tăng/giảm 1 sản phẩm bất kì trong giỏ hàng)

UPDATE	cart_by_user
SET	quantity = 2, item_total = 160000
WHERE	user_id = aaf48254-9013-4637-a965-1c8bfeb34eee AND product_id = deb98f80-6a53-4b8d-a0ee-9238642b444e

Name	Value
Updated Rows	-1
Query	UPDATE cart_by_user SET quantity = 2, item_total = 160000 WHERE user_id = aaf48254-9013-4637-a965-1c8bfeb34eee AND product_id = deb98f80-6a53-4b8d-a0ee-9238642b444e
Start time	Sat Oct 05 21:40:20 ICT 2024
Finish time	Sat Oct 05 21:40:20 ICT 2024

Hình 3. 22: Sửa số lượng sản phẩm trong giỏ hàng người dùng

Xóa: Xóa 1/Xóa tất cả sản phẩm trong giỏ hàng của 1 người dùng

(Dùng cho tính năng xóa 1, xóa tất cả sản phẩm trong giỏ hàng)

DELETE FROM	cart_by_user	WHERE	user_id = aaf48254-9013-4637-a965-1c8bfeb34eee AND product_id = deb98f80-6a53-4b8d-a0ee-9238642b444e
-------------	--------------	-------	--

Name	Value
Updated Rows	-1
Query	DELETE FROM cart_by_user WHERE user_id = aaf48254-9013-4637-a965-1c8bfeb34eee AND product_id = deb98f80-6a53-4b8d-a0ee-9238642b444e
Start time	Sat Oct 05 21:41:40 ICT 2024
Finish time	Sat Oct 05 21:41:40 ICT 2024

Hình 3. 23: Xóa một sản phẩm trong giỏ hàng

DELETE FROM	cart_by_user	WHERE	user_id = aaf48254-9013-4637-a965-1c8bfeb34eee
-------------	--------------	-------	--

Name	Value
Updated Rows	-1
Query	DELETE FROM cart_by_user WHERE user_id = aaf48254-9013-4637-a965-1c8bfeb34eee
Start time	Sat Oct 05 21:42:16 ICT 2024
Finish time	Sat Oct 05 21:42:16 ICT 2024

Hình 3. 24: Xóa tất cả sản phẩm trong giỏ hàng

3.2.6. Bảng Orders

Câu truy vấn: Hiển thị lịch sử mua hàng của người dùng

(Dùng cho tính năng hiển thị lịch sử mua hàng của người dùng ở trang đơn hàng)

orders	
user_id	
order_id	
address	
email	
fullname	
order_at	
payment_method	
phone	
status	
total	
username	

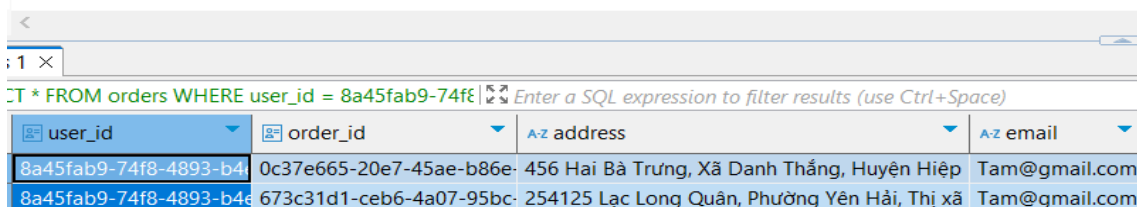

```
CREATE TABLE orders (
  order_id uuid,
  user_id uuid,
  address text,
  email text,
  fullname text,
  order_at timestamp,
  payment_method text,
  phone text,
  status text,
  total decimal,
  username text,
  PRIMARY KEY (user_id, order_id)
);
```

Hình 3. 25: Bảng dữ liệu mua hàng của người dùng

Khóa phân vùng là user_id

Khóa phân cụm là order_id

```
SELECT * FROM orders WHERE user_id = 8a45fab9-74f8-4893-b4e7-dba121cbfd8f
```



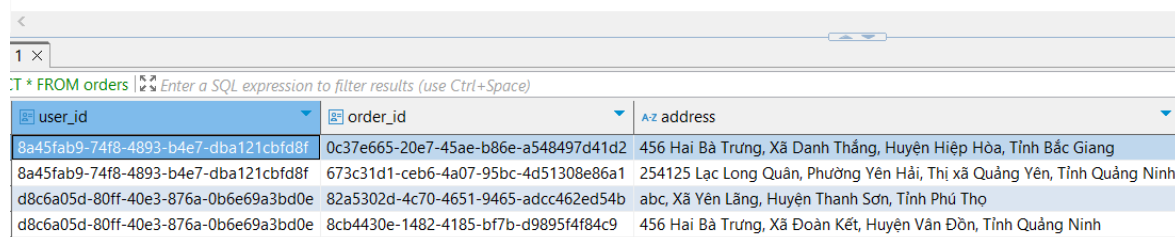
user_id	order_id	address	email
8a45fab9-74f8-4893-b4e7-dba121cbfd8f	0c37e665-20e7-45ae-b86e-a548497d41d2	456 Hai Bà Trưng, Xã Danh Thắng, Huyện Hiệp Hòa, Tỉnh Bắc Giang	Tam@gmail.com
8a45fab9-74f8-4893-b4e7-dba121cbfd8f	673c31d1-ceb6-4a07-95bc-4d51308e86a1	254125 Lạc Long Quân, Phường Yên Hải, Thị xã Quảng Yên, Tỉnh Quảng Ninh	Tam@gmail.com

Hình 3. 26: Hiển thị lịch sử mua hàng của người dùng

Câu truy vấn: Hiển thị danh sách các đơn hàng

(Dùng cho tính năng hiển thị danh sách các đơn hàng ở trang quản lý đơn hàng)

```
SELECT * FROM orders
```

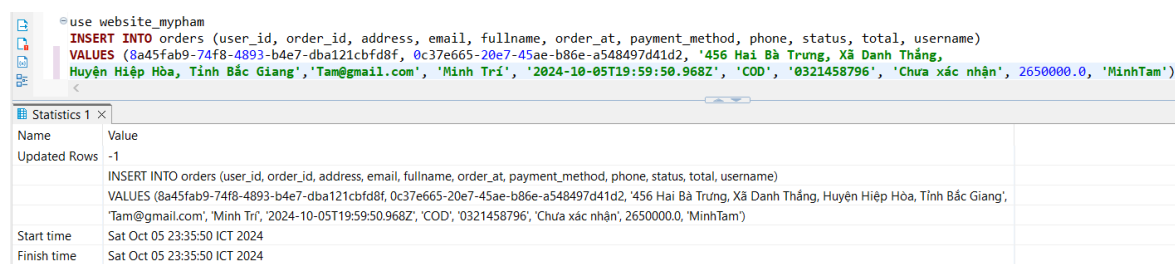


user_id	order_id	address
8a45fab9-74f8-4893-b4e7-dba121cbfd8f	0c37e665-20e7-45ae-b86e-a548497d41d2	456 Hai Bà Trưng, Xã Danh Thắng, Huyện Hiệp Hòa, Tỉnh Bắc Giang
8a45fab9-74f8-4893-b4e7-dba121cbfd8f	673c31d1-ceb6-4a07-95bc-4d51308e86a1	254125 Lạc Long Quân, Phường Yên Hải, Thị xã Quảng Yên, Tỉnh Quảng Ninh
d8c6a05d-80ff-40e3-876a-0b6e69a3bd0e	82a5302d-4c70-4651-9465-adcc462ed54b	abc, Xã Yên Lãng, Huyện Thanh Sơn, Tỉnh Phú Thọ
d8c6a05d-80ff-40e3-876a-0b6e69a3bd0e	8cb4430e-1482-4185-bf7b-d9895f4f84c9	456 Hai Bà Trưng, Xã Đoàn Kết, Huyện Văn Đồn, Tỉnh Quảng Ninh

Hình 3. 27: Hiển thị danh sách các đơn hàng

Câu truy vấn: Thêm đơn hàng của người dùng

(Dùng cho tính năng thanh toán ở trang chi tiết thanh toán)



```
INSERT INTO orders (user_id, order_id, address, email, fullname, order_at, payment_method, phone, status, total, username)
VALUES (8a45fab9-74f8-4893-b4e7-dba121cbfd8f, 0c37e665-20e7-45ae-b86e-a548497d41d2, '456 Hai Bà Trưng, Xã Danh Thắng, Huyện Hiệp Hòa, Tỉnh Bắc Giang', 'Tam@gmail.com', 'Minh Trí', '2024-10-05T19:59:50.968Z', 'COD', '0321458796', 'Chưa xác nhận', 2650000.0, 'MinhTam')
```

Name	Value
Updated Rows	-1
INSERT INTO orders (user_id, order_id, address, email, fullname, order_at, payment_method, phone, status, total, username)	
VALUES (8a45fab9-74f8-4893-b4e7-dba121cbfd8f, 0c37e665-20e7-45ae-b86e-a548497d41d2, '456 Hai Bà Trưng, Xã Danh Thắng, Huyện Hiệp Hòa, Tỉnh Bắc Giang', 'Tam@gmail.com', 'Minh Trí', '2024-10-05T19:59:50.968Z', 'COD', '0321458796', 'Chưa xác nhận', 2650000.0, 'MinhTam')	
Start time	Sat Oct 05 23:35:50 ICT 2024
Finish time	Sat Oct 05 23:35:50 ICT 2024

Hình 3. 28: Thêm đơn hàng của người dùng

Câu truy vấn: Cập nhật trạng thái đơn hàng cho người dùng

(Dùng cho tính năng thanh toán ở trang chi tiết thanh toán)

UPDATE	orders
SET	status = 'Đã xác nhận'
WHERE	order_id = 8cb4430e-1482-4185-bf7b-d9895f4f84c9 and user_id = 8a45fab9-74f8-4893-b4e7-dba121cbfd8f

Statistics 1	
Name	Value
Updated Rows	-1
Query	UPDATE orders SET status = 'Đã xác nhận' WHERE order_id = 8cb4430e-1482-4185-bf7b-d9895f4f84c9 and user_id = 8a45fab9-74f8-4893-b4e7-dba121cbfd8f
Start time	Sat Oct 05 23:58:24 ICT 2024
Finish time	Sat Oct 05 23:58:24 ICT 2024

Hình 3. 29: Cập nhật trạng thái đơn hàng

3.2.7. Bảng Order_details

Câu truy vấn: Hiển thị chi tiết 1 đơn hàng của người dùng

(Dùng cho tính năng hiển thị lịch sử mua hàng ở trang đơn hàng và dùng cho tính năng hiển thị chi tiết đơn hàng ở trang quản lý đơn hàng)

order_details	CREATE TABLE order_details (
order_id	order_id uuid,
product_id	product_id uuid,
price	price decimal,
product_image	product_image text,
product_name	product_name text,
quantity	quantity int,
	PRIMARY KEY (order_id, product_id)
);

Hình 3. 30: Bảng dữ liệu chi tiết mua hàng

Khóa phân vùng là order_id

Khóa phân cụm là product_id

SELECT * FROM order_details WHERE order_id = 8cb4430e-1482-4185-bf7b-d9895f4f84c9					
details 1					
Enter a SQL expression to filter results (use Ctrl+Space)					
order_id	product_id	price	product_image	product_name	quantity
8cb4430e-1482-4185-bf7b-d9895f4f84c9	0369b65a-7154-456d-87de-e3d97ce902ea	850,000	son-chanel.png	Son môi Rouge Coco	2
8cb4430e-1482-4185-bf7b-d9895f4f84c9	70ab11f3-0f6a-4d5d-94cd-df524d9e15a8	950,000	phan-nen-mac.png	Phấn nền MAC Studio Fix	2
8cb4430e-1482-4185-bf7b-d9895f4f84c9	8b96744d-e582-40ce-aa69-8c87b0596fad	2,100,000	nuoc-hoa-dior.png	Nước hoa Miss Dior	1

Hình 3. 31: Hiển thị chi tiết đơn hàng của người dùng

Câu truy vấn: Cho biết tổng doanh thu của các đơn hàng

(Dùng cho tính năng hiển thị doanh thu ở trang quản lý đơn hàng)

SELECT SUM(total) FROM orders	
<	
1 x	
T SUM(total) FROM orders Enter a SQ	
123 system.sum(total)	
	24,100,000

Hình 3. 32: Tổng doanh thu của đơn hàng

Câu truy vấn: Cho biết tổng số đơn hàng

(Dùng cho tính năng hiển thị tổng đơn hàng ở trang quản lý đơn hàng)

SELECT COUNT(*) FROM orders	
<	
1 x	
T COUNT(*) FROM orders Enter a	
123 count	
	5

Hình 3. 33: Tổng số đơn hàng

3.2.8. Bảng Category

Câu truy vấn: Cho biết tất cả các loại sản phẩm tại cửa hàng

(Dùng cho tính năng xem tất cả các sản phẩm)

category	CREATE TABLE website_mypham.category (category_id uuid, description text, image text, name text, PRIMARY KEY (category_id)
category_id	
A-Z description	
A-Z image	
A-Z name	

Hình 3. 34: Bảng dữ liệu loại sản phẩm

Khóa phân vùng là category_id

SELECT * FROM category

category_id	description	image	name
59312da6-2ae0-4859-b174-0e	Mặt nạ dưỡng da	mat-na.png	Mặt nạ
56434683-9d6f-4581-8282-ec9	Chăm sóc và nuôi dưỡng tóc	san-pham-toc.png	Sản phẩm tóc
792d11a3-268f-446a-9bc1-223	Giảm thâm quầng và chống nhện	kem-mat.png	Kem mắt
873c4394-9120-4db6-a554-fb8	Làm sạch da mặt hàng ngày	sua-rua-mat.png	Sữa rửa mặt
a69d0a17-0b37-4549-8dc4-20	Dưỡng môi mềm mịn	son-duong.png	Son dưỡng
63c6f7be-c0f2-4172-9263-b2d	Sản phẩm dưỡng da mềm mịn	kem-duong-da.png	Kem dưỡng da
2c52aa41-e5c7-46cf-9128-ed9	Sản phẩm làm đẹp cho đôi môi	son-moi.png	Son môi
6189d2d3-0950-4821-a3f9-f6d	Bảo vệ da khỏi tia UV	kem-chong-nang.png	Kem chống nắng
a7ed0985-f463-4abb-8d13-38c	Sản phẩm trang điểm cho da	phan-nen.png	Phấn nền

Hình 3. 35: Xem tất cả các loại sản phẩm

Thêm: Thêm mới một loại sản phẩm

(Dùng cho tính năng thêm mới 1 loại sản phẩm)

INSERT INTO category (category_id, name, description, image)
VALUES (uuid(), 'Son môi', 'Sản phẩm làm đẹp cho đôi môi dưỡng ', 'son-moi.png');

Name	Value
Updated Rows	-1
Query	INSERT INTO category (category_id, name, description, image) VALUES (uuid(), 'Son môi', 'Sản phẩm làm đẹp cho đôi môi dưỡng ', 'son-moi.png')
Start time	Sun Oct 06 14:16:23 ICT 2024
Finish time	Sun Oct 06 14:16:23 ICT 2024

Hình 3. 36: Thêm một loại sản phẩm

INSERT INTO category (category_id, name, description, image)
VALUES (uuid(), 'Son môi', 'Sản phẩm làm đẹp cho đôi môi dưỡng ', 'son-moi.png');

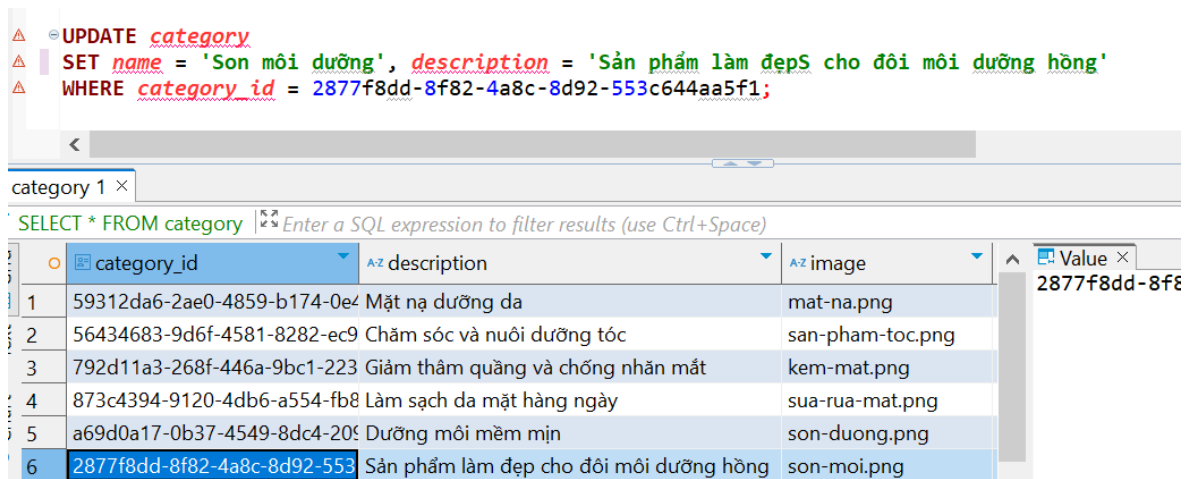
category 1 x

SELECT * FROM category

category_id	description	image	name
873c4394-9120-4db6-a554-fb82	Làm sạch da mặt hàng ngày	sua-rua-mat.png	Sữa
a69d0a17-0b37-4549-8dc4-209	Dưỡng môi mềm mịn	son-duong.png	Son
2877f8dd-8f82-4a8c-8d92-553c	Sản phẩm làm đẹp cho đôi môi dưỡng	son-moi.png	Son

Cập nhật: Cập nhật thông tin loại sản phẩm

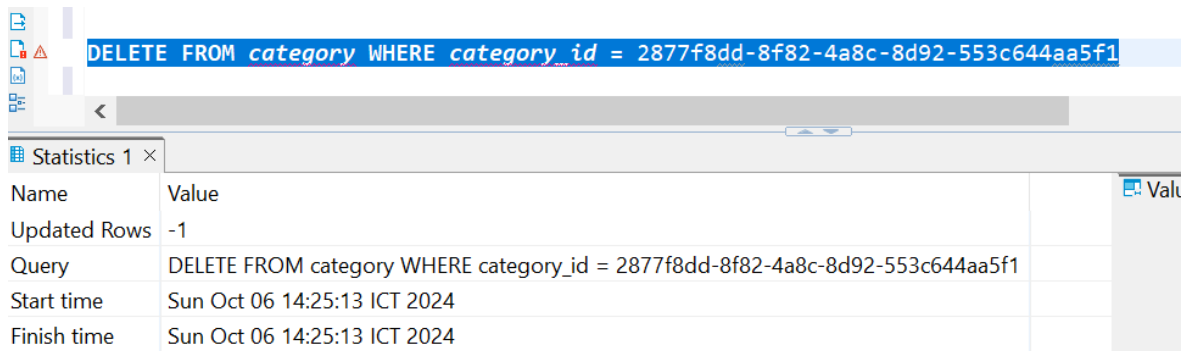
(Dùng cho tính năng cập nhật loại sản phẩm)



Hình 3. 37: Cập nhật thông tin một loại sản phẩm

Xóa: Xóa thông tin loại sản phẩm

(Dùng cho tính năng Xóa loại sản phẩm)

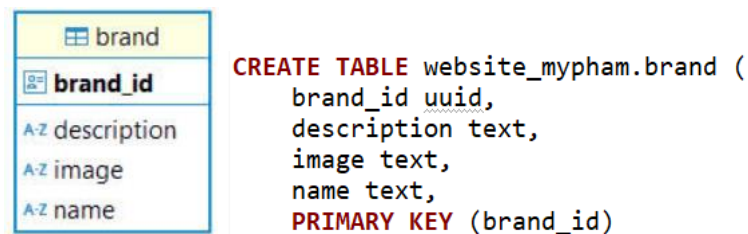


Hình 3. 38: Xóa thông tin loại sản phẩm

3.2.9. Bảng Brand

Câu truy vấn: Cho biết tất cả các thương hiệu sản phẩm

(Dùng cho tính năng xem tất cả các thương hiệu sản phẩm)



Hình 3. 39: Bảng dữ liệu thương hiệu

Khóa phân vùng là brand_id

SELECT * FROM brand

brand 1 x

Enter a SQL expression to filter results (use Ctrl+Space)

	brand_id	description	image	name
1	e6351c88-7168-4522-8a78-15f	Mỹ phẩm chăm sóc sắc đẹp cac	estee-lauder.png	Estee Lauder
2	33329eb6-028e-4613-aa37-065	Chuyên gia làm đẹp từ Pháp	lancome.png	Lancôme
3	f2bb6702-b432-400f-b5b1-5f2	Thương hiệu trang điểm chuyên mac	mac.png	MAC
4	f598d16d-f2c9-43cf-961f-0640	Thương hiệu mỹ phẩm Nhật Bản	shiseido.png	Shiseido
5	67801045-28d7-445c-a54b-64	Sản phẩm chăm sóc da không c	clinique.png	Clinique
6	0f21475d-af23-4c2b-88d9-290	Thương hiệu cao cấp đến từ Ph	chanel.png	Chanel
7	f5545dc7-8179-483a-8ac5-ac0	Mỹ phẩm thiên nhiên Hàn Quốc	innisfree.png	Innisfree
8	de154f53-4eb9-4dce-a0c4-a84	Mỹ phẩm Hàn Quốc chiết xuất t	the-face-shop.png	The Face Shop
9	1d3ce07f-c5ff-4aed-9715-00c7	Thương hiệu nổi tiếng toàn cầu	dior.png	Dior

Hình 3. 40: Xem tất cả các thương hiệu sản phẩm

Thêm: Thêm mới một thương hiệu sản phẩm

(Dùng cho tính năng thêm mới 1 thương hiệu sản phẩm)

INSERT INTO brand (brand_id, name, description, image)
VALUES (uuid(), 'Chanel cao cấp', 'Thương hiệu cao cấp đến từ đài loan', 'chanel.png');

brand 1 x

Enter a SQL expression to filter results (use Ctrl+Space)

	brand_id	description	image	name
1	e6351c88-7168-4522-8a78-15f	Mỹ phẩm chăm sóc sắc đẹp cao cấp	estee-la	Estee Lauder
2	33329eb6-028e-4613-aa37-065	Chuyên gia làm đẹp từ Pháp	lancon	Lancôme
3	f2bb6702-b432-400f-b5b1-5f2	Thương hiệu trang điểm chuyên nghiệp	mac.p	MAC
4	f598d16d-f2c9-43cf-961f-0640	Thương hiệu mỹ phẩm Nhật Bản	shiseic	Shiseido
5	67801045-28d7-445c-a54b-64	Sản phẩm chăm sóc da không gây dị ứng	cliniqu	Clinique
6	0f21475d-af23-4c2b-88d9-290	Thương hiệu cao cấp đến từ Pháp	chanel	Chanel
7	f5545dc7-8179-483a-8ac5-ac0	Mỹ phẩm thiên nhiên Hàn Quốc	innisfr	Innisfree
8	51a2491a-a231-41ae-b7e8-4af3b6af2e98	Thương hiệu cao cấp đến từ đài loan	chanel	Chanel cao cấp

Value x
Thương hiệu cao cấp

Hình 3. 41: Thêm một thương hiệu

Cập nhật: Cập nhật thông tin thương hiệu sản phẩm

(Dùng cho tính năng cập nhật thương hiệu sản phẩm)

UPDATE brand
SET name = 'Chanel cao cấp thượng hạng', description = 'Thương hiệu cao cấp đến từ đài loan nội địa'
WHERE brand_id = 51a2491a-a231-41ae-b7e8-4af3b6af2e98;

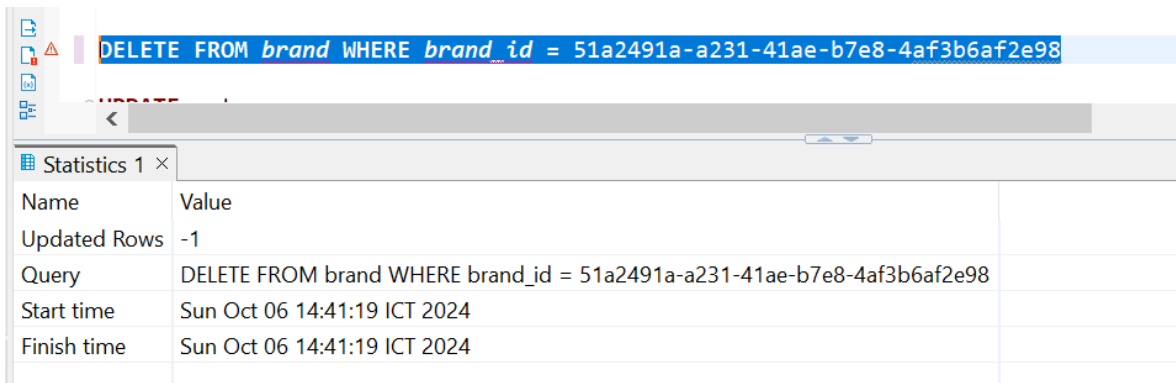
Statistics 1 x

Name	Value
Updated Rows	-1
Query	UPDATE brand SET name = 'Chanel cao cấp thượng hạng', description = 'Thương hiệu cao cấp đến từ đài loan nội địa' WHERE brand_id = 51a2491a-a231-41ae-b7e8-4af3b6af2e98
Start time	Sun Oct 06 14:39:22 ICT 2024
Finish time	Sun Oct 06 14:39:22 ICT 2024

Hình 3. 42: Cập nhật thông tin thương hiệu sản phẩm

Xóa: Xóa thông tin thương hiệu sản phẩm

(Dùng cho tính năng xóa thương hiệu sản phẩm)



The screenshot shows a database management interface. At the top, a SQL query is entered in a text field: `DELETE FROM brand WHERE brand_id = 51a2491a-a231-41ae-b7e8-4af3b6af2e98`. Below the query field, a 'Statistics 1' tab is active, displaying a table with execution details.

Name	Value
Updated Rows	-1
Query	DELETE FROM brand WHERE brand_id = 51a2491a-a231-41ae-b7e8-4af3b6af2e98
Start time	Sun Oct 06 14:41:19 ICT 2024
Finish time	Sun Oct 06 14:41:19 ICT 2024

Hình 3. 43: Xóa thương hiệu sản phẩm

Chương 4. XÂY DỰNG WEBSITE

4.1. Hàm code các chức năng tương ứng có dùng CQL

4.1.1. Chức năng quản lý sản phẩm

4.1.1.1. Hiện thị danh sách sản phẩm

```
def products(request):
    search_query = request.GET.get('search', '')
    category_query = request.GET.get('category', '')
    sort_option = request.GET.get('sort', '')

    # Lấy tất cả sản phẩm mặc định
    open_connection()

    # Truy vấn lấy tất cả sản phẩm
    query = "SELECT * FROM products"
    products = list(session.execute(query))

    # Lọc sản phẩm theo từ khóa tìm kiếm
    if search_query:
        products = [product for product in products if product.name and search_query.lower() in product.name.lower()]

    # Lọc sản phẩm theo loại sản phẩm
    if category_query:
        products = [product for product in products if product.category_name and category_query.lower() in product.category_name.lower()]

    # Sắp xếp sản phẩm theo tùy chọn
    if sort_option == 'name_asc':
        products = sorted(products, key=lambda x: x.name)
    elif sort_option == 'name_desc':
        products = sorted(products, key=lambda x: x.name, reverse=True)
    elif sort_option == 'price_asc':
        products = sorted(products, key=lambda x: x.price)
    elif sort_option == 'price_desc':
        products = sorted(products, key=lambda x: x.price, reverse=True)

    # PHÂN TRANG
    items_per_page = 5 # Số sản phẩm trên mỗi trang
    paginator = Paginator(products, items_per_page)

    page_number = request.GET.get('page', 1)
    page_obj = paginator.get_page(page_number)

    total_pages = paginator.num_pages
    previous_page = max(page_obj.number - 1, 1)
    next_page = min(page_obj.number + 1, total_pages)

    start_page = max(1, min(page_obj.number - 1, total_pages - 2))
    end_page = min(total_pages, start_page + 2)

    page_range = range(start_page, end_page + 1)

    data = {
        'DM_SanPham': page_obj,
        'search_query': search_query,
        'category_query': category_query,
        'page_range': page_range,
        'total_pages': total_pages,
        'previous_page': previous_page,
        'next_page': next_page,
    }

    close_connection()

    return render(request, 'admin/products.html', data)
```

Hình 4. 1: Code hiện thị danh sách sản phẩm

Hàm `products` xử lý yêu cầu để hiện thị danh sách sản phẩm với khả năng tìm kiếm, lọc theo loại và sắp xếp. Đầu tiên, nó mở kết nối cơ sở dữ liệu và truy vấn tất cả sản phẩm. Sau đó, nó áp dụng các bộ lọc dựa trên từ khóa tìm kiếm và loại sản phẩm, cùng với việc sắp

xếp theo tên hoặc giá nếu có yêu cầu. Cuối cùng, nó thực hiện phân trang cho danh sách sản phẩm và trả về dữ liệu đến template admin/products.html để hiển thị.

4.1.1.2. Thêm mới sản phẩm

Đầu tiên, hàm lấy danh sách danh mục và thương hiệu

```
try:
    categories_query = "SELECT category_id, name FROM category"
    brands_query = "SELECT brand_id, name FROM brand"

    categories = list(session.execute(categories_query))
    brands = list(session.execute(brands_query))
```

Sau đó, hàm lấy và xử lý dữ liệu từ form. Tiếp tục, thực hiện các truy vấn CQL để thêm sản phẩm

```
def add_product(name, price, image, quantity, supplier, mfgdate, expdate, description, instruction, brand_id, category_id):
    # Ensure the image URL uses forward slashes
    image_url = os.path.join(settings.STATIC_URL, 'img', image.name).replace('\\', '/')

    query = """
    INSERT INTO products (product_id, brand_id, category_id, name, price, saleoff, image, quantity, supplier, mfgdate, expdate, description, instruction, br
    VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)
    """
    session.execute(query, (product_id, UUID(brand_id), UUID(category_id), name, price, saleoff, image_url, quantity, supplier, mfgdate, expdate, descriptio

    queryProByCate = """
    INSERT INTO product_by_category (category_id, product_id, image, name, price)
    VALUES (%s, %s, %s, %s, %s)
    """
    session.execute(queryProByCate, (UUID(category_id), product_id, image_url, name, price))

    queryProByBrand = """
    INSERT INTO product_by_brand (brand_id, product_id, image, name, price)
    VALUES (%s, %s, %s, %s, %s)
    """
    session.execute(queryProByBrand, (UUID(brand_id), product_id, image_url, name, price))

    return redirect('/admin/products/')
```

Hình 4. 2: Code thêm mới một sản phẩm

Hàm add_product xử lý yêu cầu thêm sản phẩm mới vào cơ sở dữ liệu. Đầu tiên, nó mở kết nối và truy vấn danh sách các loại và thương hiệu sản phẩm. Nếu phương thức yêu cầu là POST và form hợp lệ, nó thu thập dữ liệu sản phẩm từ form, lưu trữ hình ảnh, và chèn dữ liệu vào bảng products cùng các bảng phân loại theo thương hiệu và loại sản phẩm. Cuối cùng, nếu không có lỗi, nó chuyển hướng về trang danh sách sản phẩm.

4.1.1.3. Sửa sản phẩm

Đầu tiên hàm thực hiện lấy dữ liệu danh mục, thương hiệu và sản phẩm hiện tại:

```

try:
    # Lấy dữ liệu từ bảng Category và Brand bằng CQL
    categories_query = "SELECT category_id, name FROM category"
    brands_query = "SELECT brand_id, name FROM brand"

    categories = list(session.execute(categories_query))
    brands = list(session.execute(brands_query))

    # Lấy dữ liệu sản phẩm hiện tại
    product_query = "SELECT * FROM products WHERE product_id = %s"
    product = session.execute(product_query, (product_id,)).one()

    if not product:
        return HttpResponseNotFound("Product not found")

```

Khi nhận được yêu cầu POST, hàm kiểm tra tính hợp lệ của biểu mẫu ProductForm. Nếu hợp lệ, thông tin sản phẩm sẽ được cập nhật.

```

# Cập nhật dữ liệu vào bảng products
query = """
UPDATE products SET category_id = %s, brand_id = %s, name = %s, price = %s, saleoff = %s, image = %s, quantity = %s, supplier = %s, mfgdate = %s, expdate = %s
WHERE product_id = %s
"""
# print("Executing query:", query)
# print("With data:", (category_id, brand_id, name, price, saleoff, image, quantity, supplier, mfgdate, expdate, description, instruction, brand_name, category_name))
session.execute(query, (UUID(category_id), UUID(brand_id), name, price, saleoff, image_url, quantity, supplier, mfgdate, expdate, description, instruction, brand_name, category_name))

```

Sau đó, hàm xóa bản ghi cũ liên quan đến sản phẩm từ các bảng product_by_category và product_by_brand, sau đó chèn bản ghi mới để cập nhật thông tin cho phù hợp với các thay đổi trong sản phẩm.

```

# delete product_by_category cũ
queryDeleteProByCate = "DELETE FROM product_by_category WHERE category_id = %s and product_id = %s"
session.execute(queryDeleteProByCate, (category_id_old, product_id))
# create product_by_category mới
queryProByCate = """
INSERT INTO product_by_category (category_id, product_id, image, name, price)
VALUES (%s, %s, %s, %s, %s)"""
session.execute(queryProByCate, (UUID(category_id), product_id, image_url, name, price))

# delete product_by_brand cũ
queryDeleteProByBrand = "DELETE FROM product_by_brand WHERE brand_id = %s and product_id = %s"
session.execute(queryDeleteProByBrand, (brand_id_old, product_id))
# create product_by_brand mới
queryProByBrand = """
INSERT INTO product_by_brand (brand_id, product_id, image, name, price)
VALUES (%s, %s, %s, %s, %s)"""
session.execute(queryProByBrand, (UUID(brand_id), product_id, image_url, name, price))

return redirect('/admin/products/')

```

Hình 4. 3 Sửa sản phẩm

4.1.1.4. Xóa sản phẩm

```

#xóa sản phẩm
def delete_product(request, product_id):
    # Mở kết nối đến Cassandra
    open_connection()

    try:
        if request.method == 'POST':
            # Xóa sản phẩm khỏi bảng products
            delete_query = "DELETE FROM products WHERE product_id = %s"
            session.execute(delete_query, (product_id,))

            category_id = request.POST.get('category_id')
            brand_id = request.POST.get('brand_id')

            # Xóa sản phẩm khỏi bảng product_by_brand
            delete_query_brand = "DELETE FROM product_by_brand WHERE brand_id = %s and product_id = %s"
            session.execute(delete_query_brand, (UUID(brand_id), product_id))

            # Xóa sản phẩm khỏi bảng product_by_category
            delete_query_cate = "DELETE FROM product_by_category WHERE category_id = %s and product_id = %s"
            session.execute(delete_query_cate, (UUID(category_id), product_id))

            return redirect('/admin/products/')

    except Exception as e:
        print(f"Lỗi khi xóa sản phẩm: {e}")
    finally:
        # Đóng kết nối
        close_connection()

    return redirect('/admin/products/')

```

Hình 4. 4: Xóa sản phẩm

Hàm `delete_product` thực hiện chức năng xóa một sản phẩm khỏi cơ sở dữ liệu Cassandra dựa trên `product_id` khi nhận được yêu cầu POST. Sau khi mở kết nối, nó sẽ xóa sản phẩm khỏi bảng `products` và các bảng liên quan `product_by_brand` và `product_by_category` sử dụng các truy vấn DELETE. Nếu có lỗi xảy ra trong quá trình xóa, lỗi sẽ được ghi lại, và sau đó hàm sẽ đóng kết nối và chuyển hướng về trang danh sách sản phẩm.

4.1.1.5. Xem chi tiết sản phẩm

```

#chi tiết sản phẩm
def detail_product(request, product_id):
    # Mở kết nối đến Cassandra
    open_connection()

    try:
        # Lấy dữ liệu sản phẩm hiện tại
        product_query = "SELECT * FROM products WHERE product_id = %s"
        product = session.execute(product_query, (product_id,)).one()

        if not product:
            return HttpResponseNotFound("Product not found")

        # Truyền dữ liệu sản phẩm vào context
        context = {
            'product': product,
        }

    except Exception as e:
        print(f"Lỗi khi lấy dữ liệu: {e}")
        import traceback
        traceback.print_exc()
    finally:
        # Đóng kết nối
        close_connection()

    return render(request, 'admin/detail_product.html', context)

```

Hình 4. 5: Xem chi tiết sản phẩm

Hàm `detail_product` lấy thông tin chi tiết về một sản phẩm cụ thể dựa trên `product_id`. Nó mở kết nối tới cơ sở dữ liệu Cassandra, thực hiện truy vấn để lấy thông tin sản phẩm và kiểm tra xem sản phẩm có tồn tại hay không. Nếu sản phẩm không được tìm thấy, hàm sẽ trả về một phản hồi "Product not found". Nếu có lỗi xảy ra trong quá trình truy vấn, lỗi sẽ được ghi lại, và sau đó hàm sẽ đóng kết nối trước khi render trang chi tiết sản phẩm.

4.1.2. Quản lý người dùng

4.1.2.1. Hiển thị danh sách người dùng

```

#danh sách người dùng
def users(request):
    session_info = is_logged_in(request)
    if not session_info['email'] or session_info['role'] != 'admin':
        return HttpResponseRedirect('/login')

    open_connection()
    query = "SELECT * FROM user"
    listUser= list(session.execute(query))

    #PHÂN TRANG
    items_per_page = 5 # Số sản phẩm trên mỗi trang
    paginator = Paginator(listUser, items_per_page)
    page_number = request.GET.get('page', 1)
    page_obj = paginator.get_page(page_number)
    total_pages = paginator.num_pages
    previous_page = max(page_obj.number - 1, 1)
    next_page = min(page_obj.number + 1, total_pages)
    start_page = max(1, min(page_obj.number - 1, total_pages - 2))
    end_page = min(total_pages, start_page + 2)
    page_range = range(start_page, end_page + 1)

    data = {
        'listUser': page_obj,
        'page_range': page_range,
        'total_pages': total_pages,
        'previous_page': previous_page,
        'next_page': next_page,
    }
    close_connection()
    return render(request, 'admin/users.html', data)

```

Hình 4. 6. Hiện thị danh sách người dùng

Hàm users kiểm tra xem người dùng có đăng nhập và có vai trò là admin hay không; nếu không, nó chuyển hướng đến trang đăng nhập. Sau đó, mở kết nối đến cơ sở dữ liệu và lấy danh sách tất cả người dùng. Hàm sử dụng phân trang để giới hạn số lượng người dùng hiển thị trên mỗi trang, với 5 người dùng mỗi trang, và tính toán các thông tin cần thiết như số trang, trang trước và sau, và phạm vi trang. Cuối cùng, nó đóng kết nối và trả về trang users.html với dữ liệu phân trang cho danh sách người dùng.

4.1.2.2. Thêm người dùng mới

```

#thêm người dùng
def add_user(request):
    session_info = is_logged_in(request)
    if not session_info['email'] or session_info['role'] != 'admin':
        return HttpResponseRedirect('/login')

    open_connection()

    if request.method == 'POST':
        form = UserForm(request.POST)
        if form.is_valid():
            user_data = form.cleaned_data
            email = user_data['email']
            fullname = user_data['fullname']
            password = user_data['password']
            phone = user_data['phone']
            role = user_data['role']
            username = user_data['username']

            # Insert user data into Cassandra
            query = """
            INSERT INTO user (user_id, email, fullname, password, phone, role, username)
            VALUES (uuid(), %s, %s, %s, %s, %s, %s)
            """
            session.execute(query, [email, fullname, password, phone, role, username])

            close_connection()
            return HttpResponseRedirect('/admin/users')
        else:
            form = UserForm()

    close_connection()
    return render(request, 'admin/add_user.html', {'form': form})

```

Hình 4. 7: Thêm người dùng mới

Hàm `add_user` kiểm tra nếu yêu cầu là phương thức POST, nó khởi tạo một biểu mẫu `UserForm` với dữ liệu gửi lên và kiểm tra tính hợp lệ. Nếu dữ liệu hợp lệ, nó trích xuất thông tin người dùng và chèn vào bảng `user` trong cơ sở dữ liệu `Cassandra`. Cuối cùng, hàm đóng kết nối và chuyển hướng về trang danh sách người dùng nếu thêm thành công, hoặc trả về biểu mẫu để thêm người dùng mới.

4.1.2.3. Sửa thông tin người dùng


```

if request.method == 'POST':
    form = UserForm(request.POST)
    if form.is_valid():
        user_data = form.cleaned_data
        email = user_data['email']
        fullname = user_data['fullname']
        password = user_data['password']
        phone = user_data['phone']
        role = user_data['role']
        username = user_data['username']

        # Update user data in Cassandra
        query = """
        UPDATE user SET email = %s, fullname = %s, password = %s, phone = %s, role = %s, username = %s
        WHERE user_id = %s
        """
        session.execute(query, [email, fullname, password, phone, role, username, user_id])

        close_connection()
        return HttpResponseRedirect('/admin/users')
    else:
        initial_data = {
            'email': user.email,
            'fullname': user.fullname,
            'password': user.password,
            'phone': user.phone,
            'role': user.role,
            'username': user.username,
        }
        form = UserForm(initial=initial_data)

close_connection()
return render(request, 'admin/edit_user.html', {'form': form, 'user_id': user_id})

```

Hình 4. 8: Sửa thông tin người dùng

Hàm `edit_user` kiểm tra xem người dùng có đăng nhập và có vai trò là admin hay không; nếu không, nó chuyển hướng đến trang đăng nhập. Sau đó, nó mở kết nối và truy vấn thông tin người dùng từ cơ sở dữ liệu Cassandra dựa trên `user_id`. Nếu tìm thấy người dùng, hàm xử lý yêu cầu POST để cập nhật thông tin người dùng bằng biểu mẫu `UserForm` nếu dữ liệu hợp lệ, hoặc khởi tạo biểu mẫu với dữ liệu hiện tại để hiển thị cho người dùng. Cuối cùng, nó đóng kết nối và trả về trang chỉnh sửa người dùng với biểu mẫu đã được cập nhật.

4.1.2.4. Xóa người dùng

```

#xóa người dùng
def delete_user(request, user_id):
    session_info = is_logged_in(request)
    if not session_info['email'] or session_info['role'] != 'admin':
        return HttpResponseRedirect('/login')

    open_connection()

    try:
        # Delete user data from Cassandra
        query = "DELETE FROM user WHERE user_id = %s"
        session.execute(query, [user_id])
    except Exception as e:
        print(f"Error deleting user: {e}")
    finally:
        close_connection()

    return HttpResponseRedirect('/admin/users')

```

Hình 4. 9: Xóa người dùng

Hàm `delete_user` kiểm tra xem người dùng đã đăng nhập và có vai trò là admin chưa; nếu không, nó chuyển hướng đến trang đăng nhập. Sau khi mở kết nối với cơ sở dữ liệu Cassandra, hàm xóa thông tin người dùng theo `user_id` thông qua một truy vấn DELETE.

4.1.3. Chức năng hiển thị sản phẩm theo thương hiệu

```

def DSSPTheoTH(request, ml):
    open_connection()
    items_per_page = 5
    page_number = int(request.GET.get('page', 1))
    try:
        query = "SELECT * FROM product_by_brand WHERE brand_id = %s LIMIT %s"
        result = session.execute(query, [ml, items_per_page])
        products = result.all()
        #Phân trang
        total_items = len(products)
        total_pages = (total_items + items_per_page - 1) // items_per_page
        start_page = max(1, page_number - 1)
        end_page = min(total_pages, start_page + 2)
        page_range = range(start_page, end_page + 1)
        previous_page = max(1, page_number - 1)
        next_page = min(total_pages, page_number + 1)
        data = {
            'dm_sp': products,
            'page_range': page_range,
            'total_pages': total_pages,
            'previous_page': previous_page,
            'next_page': next_page,
        }
    except Exception as e:
        print(f"Lỗi khi thực hiện truy vấn Cassandra: {e}")
        data = {}
    finally:
        close_connection()
    return render(request, "page/shop_ThuongHieu.html", data)

```

Hình 4. 10 Hiển thị danh sách sản phẩm theo thương hiệu

Hàm DSSPTheoTH thực hiện truy vấn CQL để lấy danh sách sản phẩm theo thương hiệu từ Cassandra với giới hạn số sản phẩm trên mỗi trang. Nó sử dụng kết quả truy vấn để phân trang và tính toán các thông số như tổng số trang, phạm vi trang, và trang trước/sau. Kết quả cuối cùng trả về là trang HTML chứa danh sách sản phẩm và thông tin phân trang.

4.1.4. Chức năng giỏ hàng

4.1.4.1. Hiển thị sản phẩm trong giỏ hàng

```

def cart(request):
    user_info = is_logged_in(request)
    if not user_info['email']:
        return redirect('login')

    user_id = UUID(user_info.get('user_id'))
    try:
        open_connection()
        # Truy vấn để lấy tất cả sản phẩm trong giỏ hàng 1 người
        query = "SELECT * FROM cart_by_user WHERE user_id = %s"
        rows = session.execute(query, [user_id])
        cart_details = []
        for item in rows:
            cart_details.append({
                'product_id': item.product_id,
                'product_name': item.product_name,
                'product_image': item.image,
                'quantity': item.quantity,
                'price': item.price,
                'item_total': item.item_total
            })
        # Truy vấn để lấy tổng tiền tất cả mặt hàng trong giỏ hàng 1 người
        total_price_query = "SELECT SUM(item_total) AS total_price FROM cart_by_user WHERE user_id = %s"
        total_price_result = session.execute(total_price_query, [user_id]).one()
        total_price = total_price_result.total_price if total_price_result and total_price_result.total_price is not None else 0
        return render(request, 'page/cart.html', {
            'cart_items': cart_details,
            'total_price': total_price,
        })
    except Exception as e:
        print(f"Lỗi khi truy vấn giỏ hàng: {e}")
        return render(request, 'page/cart.html', {'cart_items': [], 'total_price': 0})
    finally:
        close_connection()

```

Hình 4. 11 Hiển thị giỏ hàng của người dùng

Hàm sẽ truy vấn cơ sở dữ liệu để lấy tất cả sản phẩm trong giỏ hàng của người dùng và tính tổng giá trị của các mặt hàng. Kết quả sẽ được trả về trong một trang HTML, trong đó có danh sách sản phẩm trong giỏ hàng và tổng giá. Nếu có lỗi xảy ra trong quá trình truy vấn, giỏ hàng sẽ được trả về dưới dạng rỗng với tổng giá trị bằng 0.

4.1.4.2. Đếm số lượng các dòng sản phẩm trong giỏ hàng

```

def cart_item_count(request):
    total_items = 0
    user_info = is_logged_in(request)
    if user_info.get('user_id'):
        user_id = UUID(user_info['user_id'])

        try:
            open_connection()
            # Truy vấn số lượng sản phẩm trong giỏ hàng
            query = "SELECT COUNT(product_id) FROM cart_by_user WHERE user_id = %s"
            result = session.execute(query, [user_id])
            total_items = result.one()[0]

        except Exception as e:
            print(f"Lỗi khi truy vấn số lượng sản phẩm trong giỏ hàng: {e}")
        finally:
            close_connection()

    return {'cart_item_count': total_items}

```

Hình 4. 12 Đếm số lượng sản phẩm trong giỏ hàng của người dùng

Hàm thực hiện truy vấn để đếm số lượng sản phẩm trong giỏ hàng dựa trên user_id. Kết quả được lưu trữ trong biến total_items. Cuối cùng, hàm trả về một từ điển chứa số lượng sản phẩm trong giỏ hàng.

4.1.4.3. Thêm sản phẩm vào giỏ hàng

```
def addProToCart(request):
    user_info = is_logged_in(request)
    if not user_info['email']:
        return redirect('login')
    user_id = UUID(user_info.get('user_id'))

    if request.GET.get('action') == 'addcart' and 'id_product' in request.GET:
        product_id = UUID(request.GET['id_product'])

        try:
            open_connection()
            query_product = "SELECT * FROM Products WHERE product_id = %s"
            product = session.execute(query_product, [product_id]).one()
            if user_id and product:
                # Kiểm tra xem sản phẩm đã có trong giỏ hàng chưa
                query_cart = "SELECT * FROM cart_by_user WHERE user_id = %s AND product_id = %s"
                cart_item = session.execute(query_cart, [user_id, product.product_id]).one()
                if cart_item: # Đã tồn tại trong giỏ hàng ==> tăng số lượng
                    new_quantity = cart_item.quantity + 1
                    new_item_total = product.price * new_quantity
                    query_update = "UPDATE cart_by_user SET quantity = %s, item_total=%s WHERE user_id = %s AND product_id = %s"
                    session.execute(query_update, [new_quantity, new_item_total, user_id, product.product_id])
                else: # Chưa có trong giỏ => tạo mục mới
                    query_insert = "INSERT INTO cart_by_user (user_id, product_id, product_name, quantity, price, image, item_total)"
                    session.execute(query_insert, [
                        user_id, product.product_id, product.name, 1, float(product.price), product.image, float(product.price)
                    ])
            else: # Nếu không có user, chuyển hướng đăng nhập
                return redirect('login')
        except Exception as e:
            print(f"Lỗi khi thêm sản phẩm vào giỏ: {e}")
        finally:
            close_connection()
    return redirect('shop')
```

Hình 4. 13. Thêm sản phẩm vào giỏ hàng

Hàm addProToCart kiểm tra xem người dùng đã đăng nhập chưa và lấy user_id. Nếu người dùng thực hiện hành động thêm sản phẩm vào giỏ hàng (addcart), hàm sẽ truy vấn cơ sở dữ liệu để lấy thông tin sản phẩm và kiểm tra sự tồn tại của sản phẩm đó trong giỏ hàng. Nếu đã có, nó tăng số lượng và cập nhật tổng giá. Nếu chưa có, hàm thêm sản phẩm vào giỏ hàng với số lượng là 1. Cuối cùng, hàm chuyển hướng người dùng đến trang cửa hàng và xử lý lỗi nếu xảy ra.

4.1.4.4. Tăng số lượng 1 sản phẩm trong giỏ hàng

```
# tăng số lượng sản phẩm trong giỏ hàng
def increase_quantity(request, product_id):
    user_info = is_logged_in(request)
    if not user_info['email']:
        return redirect('login')

    user_id = UUID(user_info.get('user_id'))
    if user_id:
        try:
            open_connection()
            # Truy vấn lấy số lượng hiện tại của sản phẩm trong giỏ hàng
            query_select = "SELECT * FROM cart_by_user WHERE user_id = %s AND product_id = %s"
            cart_item = session.execute(query_select, [user_id, product_id]).one()

            if cart_item:
                # Tăng số lượng sản phẩm
                new_quantity = cart_item.quantity + 1
                new_total_item = cart_item.item_total + cart_item.price # Cập nhật tổng giá trị của mặt hàng
                query_update = "UPDATE cart_by_user SET quantity = %s, item_total = %s WHERE user_id = %s AND product_id = %s"
                session.execute(query_update, [new_quantity, new_total_item, user_id, product_id])
            except Exception as e:
                print(f"Lỗi khi tăng số lượng sản phẩm: {e}")
            finally:
                close_connection()
        return redirect('cart')
```

Hình 4. 14 Tăng số lượng sản phẩm trong giỏ hàng

Hàm `increase_quantity` kiểm tra xem người dùng đã đăng nhập chưa và lấy `user_id`. Nếu người dùng đã đăng nhập, hàm truy vấn để lấy số lượng hiện tại của sản phẩm trong giỏ hàng, sau đó tăng số lượng lên 1 và cập nhật tổng giá trị của mặt hàng.

4.1.4.5. Giảm số lượng sản phẩm trong giỏ hàng

```
# giảm số lượng sản phẩm trong giỏ hàng
def decrease_quantity(request, product_id):
    user_info = is_logged_in(request)
    if not user_info['email']:
        return redirect('login')

    user_id = UUID(user_info.get('user_id'))
    if user_id:
        try:
            open_connection()
            # Truy vấn lấy số lượng hiện tại và giá trị của sản phẩm trong giỏ hàng
            query_select = "SELECT * FROM cart_by_user WHERE user_id = %s AND product_id = %s"
            cart_item = session.execute(query_select, [user_id, product_id]).one() # Sử dụng one_or_none()

            if cart_item:
                if cart_item.quantity > 1: # Giảm số lượng sản phẩm
                    new_quantity = cart_item.quantity - 1
                    new_total_item = cart_item.item_total - cart_item.price # Cập nhật tổng giá trị của mặt hàng
                    query_update = "UPDATE cart_by_user SET quantity = %s, item_total = %s WHERE user_id = %s AND product_id = %s"
                    session.execute(query_update, [new_quantity, new_total_item, user_id, product_id])
                else: # Xóa sản phẩm khỏi giỏ hàng nếu số lượng là 1
                    query_delete = "DELETE FROM cart_by_user WHERE user_id = %s AND product_id = %s"
                    session.execute(query_delete, [user_id, product_id])
            else:
                print("Không tìm thấy sản phẩm trong giỏ hàng.")
            except Exception as e:
                print(f"Lỗi khi giảm số lượng sản phẩm: {e}")
            finally:
                close_connection()
        return redirect('cart')
```

Hình 4. 15 Giảm số lượng sản phẩm trong giỏ hàng

Hàm `decrease_quantity` kiểm tra người dùng đã đăng nhập hay chưa và lấy `user_id`. Sau đó, hàm truy vấn để lấy thông tin sản phẩm trong giỏ hàng. Nếu sản phẩm tồn tại và số lượng lớn hơn 1, số lượng sẽ được giảm đi 1 và cập nhật tổng giá trị của mặt hàng. Nếu số lượng chỉ là 1, sản phẩm sẽ bị xóa khỏi giỏ hàng.

4.1.4.6. Xóa 1 sản phẩm trong giỏ hàng

```
# xóa 1 sản phẩm trong giỏ hàng
def deleteProFromCart(request, product_id):
    user_info = is_logged_in(request)
    if not user_info['email']:
        return redirect('login')
    user_id = UUID(user_info.get('user_id'))

    if user_id:
        try:
            open_connection()
            query_delete_item = "DELETE FROM cart_by_user WHERE user_id = %s AND product_id = %s"
            session.execute(query_delete_item, [user_id, product_id])
        except Exception as e:
            print(f"Lỗi khi xóa sản phẩm khỏi giỏ hàng: {e}")
        finally:
            close_connection()
    return redirect('cart')
```

Hình 4. 16 Xóa 1 sản phẩm trong giỏ hàng

Hàm `deleteProFromCart` kiểm tra xem người dùng đã đăng nhập hay chưa và lấy `user_id`. Sau đó, nếu người dùng hợp lệ, hàm thực hiện truy vấn để xóa sản phẩm với `product_id` tương ứng khỏi giỏ hàng của người dùng (`cart_by_user`). Nếu xảy ra lỗi trong quá trình xóa, nó sẽ in thông báo lỗi. Cuối cùng, kết nối được đóng và người dùng được chuyển hướng về trang giỏ hàng.

4.1.4.7. Xóa tất cả sản phẩm trong giỏ hàng

```
#xóa tất cả sản phẩm giỏ hàng
def clearCart(request):
    user_info = is_logged_in(request)
    if not user_info['email']:
        return redirect('login')
    user_id = UUID(user_info.get('user_id'))

    if user_id:
        try:
            open_connection()
            query_delete_cart = "DELETE FROM cart_by_user WHERE user_id = %s"
            session.execute(query_delete_cart, [user_id]) # Xóa tất cả sản phẩm của user_id
        except Exception as e:
            print(f"Lỗi khi xóa giỏ hàng: {e}")
        finally:
            close_connection()

    return redirect('cart')
```

Hình 4. 17 Xóa tất cả sản phẩm trong giỏ hàng

Hàm `clearCart` kiểm tra xem người dùng đã đăng nhập hay chưa và lấy `user_id`. Nếu người dùng hợp lệ, hàm thực hiện truy vấn để xóa tất cả sản phẩm trong giỏ hàng của người

dùng đó từ bảng `cart_by_user`. Nếu có lỗi xảy ra trong quá trình xóa, nó sẽ in ra thông báo lỗi. Cuối cùng, kết nối được đóng và người dùng được chuyển hướng về trang giỏ hàng.

4.1.5. Chức năng thanh toán

CHI TIẾT THANH TOÁN

Họ và tên *

Họ tên

Địa chỉ *

Địa chỉ nhận hàng

Tỉnh / Thành *

Chọn tỉnh thành

Quận / Huyện *

Chọn quận huyện

Phường / Xã *

Chọn phường xã

Số điện thoại *

Phương thức thanh toán *

☒ Thanh toán tiền mặt khi nhận (COD)

☐ Chuyển khoản ngân hàng

1900 8198

HÓA ĐƠN CỦA BẠN

Sản phẩm	Tổng tiền
1. Son môi Rouge Coco Số lượng: 1	850.000 đ
2. Phấn nền MAC Studio Fix Số lượng: 2	1.900.000 đ
Thành tiền	2.750.000 đ

HOÀN TẤT THANH TOÁN

Hình 4. 18 Giao diện chức năng thanh toán

Hàm `checkout` thực hiện lấy `user_id` của người dùng đã đăng nhập để lấy danh sách sản phẩm trong giỏ hàng của người dùng. Bên cạnh đó hàm tính tổng số tiền và chuẩn bị danh sách chi tiết các sản phẩm để hiển thị trên trang.


```

def checkout(request):
    user_info = is_logged_in(request)
    if not user_info['email']:
        return redirect('login')

    user_id = UUID(user_info.get('user_id'))
    order_form = OrderForm()
    order_detail_form = OrderDetailForm()

    try:
        open_connection()

        query = "SELECT product_id, product_name, image, quantity, price FROM cart_by_user WHERE user_id = %s "
        cart_items = session.execute(query, [user_id])

        cart_details = []
        total_amount = 0
        index = 0

        # lấy thông tin các sp trong giỏ hàng để hiển thị trong template
        for item in cart_items:
            product_total = item.price * item.quantity
            total_amount += product_total
            # total_amount = item.item_total // lấy total từ cart
            index+=1

            cart_details.append({
                'index': index,
                'product_id': item.product_id,
                'product_name': item.product_name,
                'product_image': item.image,
                'price': item.price,
                'quantity': item.quantity,
                'product_total': "{:,.0f}".format(product_total).replace(",", ".")
            })

```

Nếu người dùng gửi thông tin đặt hàng (qua phương thức POST), hàm sẽ lấy thông tin địa chỉ và phương thức thanh toán từ biểu mẫu. Kiểm tra nếu thông tin nhập hợp lệ, tạo

một đơn hàng mới và lưu thông tin đơn hàng vào cơ sở dữ liệu. Cuối cùng, hàm sẽ cập nhật tổng tiền và dọn dẹp giỏ hàng.

```
if request.method == 'POST':
    order_form = OrderForm(request.POST)
    order_detail_form = OrderDetailForm(request.POST)

    if order_form.is_valid():
        recipient_name = request.POST.get('recipient_name')
        ward = request.POST.get('ward')
        district = request.POST.get('district')
        city = request.POST.get('city')
        recipient_address = request.POST.get('recipient_address')
        recipient_phone = request.POST.get('recipient_phone')
        method_payment = request.POST.get('method_payment')

        full_recipient_address = f"{recipient_address}, {ward}, {district}, {city}"

        # Tạo và lưu đơn hàng mới
        total=0
        order_at = datetime.now()
        status = 'Chưa xác nhận'
        order_id = uuid4() # Sử dụng uuid4() để tạo UUID

        queryOrder = "INSERT INTO orders (user_id, order_id, address, email, fullname, order_at, payment_method, phone, status, total, username) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)"
        session.execute(queryOrder,(user_id, order_id, full_recipient_address, user_info['email'],recipient_name, order_at, method_payment, recipient_phone, status, total, username))

        # chi tiết đơn hàng

        # chi tiết đơn hàng
        for item in cart_details:
            product_id = item['product_id']
            product_name = item['product_name']
            quantity = item['quantity']
            price = item['price']
            product_image = item['product_image']
            total_amount = total_amount + price * quantity

            # Tạo và lưu chi tiết đơn hàng
            queryOrderDetails = "INSERT INTO order_details (order_id, product_id, price, product_image, product_name, quantity) VALUES (%s, %s, %s, %s, %s, %s)"
            session.execute(queryOrderDetails,(order_id, product_id, price, product_image, product_name, quantity))

        # Cập nhật tổng số tiền đơn hàng
        queryUpdateOrder = "UPDATE orders SET total = %s WHERE order_id = %s AND user_id = %s"
        session.execute(queryUpdateOrder, [total_amount, order_id, user_id])

        # Xóa giỏ hàng sau khi đặt hàng thành công
        query_delete_cart = "DELETE FROM cart_by_user WHERE user_id = %s"
        session.execute(query_delete_cart, [user_id])

        return HttpResponseRedirect('/order_details')

finally:
    close_connection()

total_amount = "{:,0f}".format(total_amount).replace(",", ".") #format tổng tiền

return render(request, 'page/checkout.html', {'order_form': order_form,
'order_detail_form': order_detail_form, 'cart_items': cart_items, 'cart_details':cart_details,
'total_amount': total_amount})
```

Hình 4. 19 Code thực hiện chức năng thanh toán

4.1.6. Chức năng hiển thị lịch sử đơn hàng của người dùng

```

user_id = UUID(user_info.get('user_id'))

if user_id:
    try:
        open_connection()
        queryOrder = "SELECT * FROM orders WHERE user_id = %s"
        orders = session.execute(queryOrder, [user_id])
        order_list = []
        # lặp lấy các thông tin trong từng hd của user (lấy chi tiết hd)
        for item in orders:
            order_details = "SELECT * FROM order_details WHERE order_id = %s"
            order_details = session.execute(order_details, [item.order_id])

            details_list = []

            for detail in order_details: # lặp lấy các sản phẩm trong 1 hd (lấy các sp trong cthd)
                details_list.append({ # add vào list để hiển thị bên html
                    'product_name': detail.product_name,
                    'product_image': detail.product_image,
                    'quantity': detail.quantity,
                    'product_price': "{:,.0f}".format(detail.price).replace(",", "."),
                    'product_total': "{:,.0f}".format(detail.quantity * detail.price).replace(",", "."),
                })
            order_list.append({ # add từng hd vào list để hiển thị bên html
                'order_id': item.order_id,
                'order_date': item.order_at,
                'status': item.status,
                'total_price': "{:,.0f}".format(item.total).replace(",", "."),
                'details': details_list,
            })
        return render(request, 'page/order_details.html', {
            'order_list': order_list
            # 'total_price': total_price
        })
    except:
        pass

```

Hình 4. 20 Code hiển thị lịch sử tất cả đơn hàng của người dùng

Hàm `order_user` xử lý việc hiển thị thông tin đơn hàng của người dùng đã đăng nhập. Đầu tiên, hàm kiểm tra xem nếu người dùng đã đăng nhập, hàm sẽ mở kết nối cơ sở dữ liệu và truy vấn để lấy tất cả các đơn hàng của người dùng. Sau đó, cho mỗi đơn hàng, hàm sẽ lấy chi tiết sản phẩm và tạo danh sách các thông tin cần thiết để hiển thị trên trang `order_details.html`. Cuối cùng, hàm sẽ đóng kết nối và trả về danh sách đơn hàng đã được chuẩn bị cho giao diện người dùng.

4.1.7. Chức năng quản lý đơn hàng

4.1.7.1. Hiển thị danh sách các đơn hàng (cùng với tổng doanh thu, tổng số đơn hàng ...)

Tổng doanh thu 24.100.000 đ	Tổng đơn hàng đã đặt 4	Tổng số account 5	Sản phẩm hiện có 4
---------------------------------------	----------------------------------	-----------------------------	------------------------------

Đơn Đặt Hàng

Mã đơn	Địa chỉ email	Ngày đặt	Phương thức thanh toán	Tổng tiền	Trạng thái	Thao tác
0c37e665-20e7-45ae-b86e-a548497d41d2	Tam@gmail.com	05/10/2024 19:59:50	COD	2650000.0	Chưa xác nhận	Xem chi tiết
673c31d1-ceb6-4a07-95bc-4d51308e86a1	Tam@gmail.com	05/10/2024 20:44:12	Chuyển khoản	12350000.0	Đã xác nhận	Xem chi tiết
82a5302d-4c70-4651-9465-adcc462ed54b	Thanh@gmail.com	05/10/2024 21:25:44	Chuyển khoản	3400000.0	Đã xác nhận	Xem chi tiết
8cb4430e-1482-4185-bf7b-d9895f4f84c9	Thanh@gmail.com	05/10/2024 21:22:19	Chuyển khoản	5700000.0	Chưa xác nhận	Xem chi tiết

Hình 4. 21 Giao diện quản lý đơn hàng của admin

```
def orders(request):
    user_info = is_logged_in(request)
    if not user_info['email']:
        return redirect('login')
    try:
        open_connection()
        # Lấy danh sách đơn hàng
        queryOrder = "SELECT * FROM orders"
        list_orders = session.execute(queryOrder)
        # Tổng doanh thu
        queryTotalOrder = "SELECT SUM(total) FROM orders"
        total = session.execute(queryTotalOrder)
        total_revenue = total.one()[0]
        # Tổng số đơn hàng đã đặt
        queryTotalOrder = "SELECT COUNT(*) FROM orders"
        total = session.execute(queryTotalOrder)
        total_orders = total.one()[0]
        # Tổng số account
        queryTotalAccount = "SELECT COUNT(*) FROM user"
        total = session.execute(queryTotalAccount)
        total_accounts = total.one()[0]
        # Sản phẩm hiện có
        queryToTalPro = "SELECT COUNT(*) FROM products"
        total = session.execute(queryToTalPro)
        total_products = total.one()[0]
        data = {
            'listOrder': list_orders,
            'total_revenue': "{:,.0f}".format(total_revenue).replace(",", "."),
            'total_orders': total_orders,
            'total_accounts': total_accounts,
            'total_products': total_products
        }
    finally:
        close_connection()
    return render(request, 'admin/orders.html', data)
```

Hình 4. 22 Code xem danh sách đơn hàng

Hàm orders hàm mở kết nối đến cơ sở dữ liệu và thực hiện các truy vấn để lấy danh sách đơn hàng, tổng doanh thu từ các đơn hàng, tổng số đơn hàng đã đặt, tổng số tài khoản

người dùng, và số lượng sản phẩm hiện có. Tất cả dữ liệu này được định dạng và lưu vào một từ điển data, rồi trả về để hiển thị trên trang quản trị admin/orders.html.

4.1.7.2. Cập nhật trạng thái 1 đơn hàng của 1 người dùng

Danh sách đơn hàng

Chi tiết đơn hàng

Mã đơn hàng: 8cb4430e-1482-4185-bf7b-d9895f4f84c9
21:22:19

Ngày lập: 05/10/2024

Trạng
thái: Chưa xác nhận

Cập nhật trạng
thái

Mã sản phẩm	Tên sản phẩm	Số lượng	Giá trị sản phẩm
0369b65a-7154-456d-87de-e3d97ce902ea	Son môi Rouge Coco	2	850000.0 VNĐ
70ab11f3-0f6a-4d5d-94cd-df524d9e15a8	Phấn nền MAC Studio Fix	2	950000.0 VNĐ
8b96744d-e582-40ce-aa69-8c87b0596fad	Nước hoa Miss Dior	1	2100000.0 VNĐ

Tổng tiền: 5700000.0 VNĐ

Thông tin đơn hàng

Thông tin người đặt

Mã khách hàng: d8c6a05d-80ff-40e3-876a-0b6e69a3bd0e
Email: Thanh@gmail.com

Tên khách hàng: thanhhdzai
Số điện thoại: 0596321478

Thông tin người nhận

Tên người nhận: Thế thanh
Địa chỉ: 456 Hai Bà Trưng, Xã Đoàn Kết, Huyện Văn Đồn, Tỉnh Quảng
Ninh

Số điện thoại: 0596321478
Phương thức thanh toán: Chuyển khoản

Hình 4. 23 Giao diện cập nhật trạng thái của đơn hàng

```
def detail_order(request, id_order, id_user):
    user_info = is_logged_in(request)
    if not user_info['email']:
        return redirect('login')

    try:
        open_connection()
        queryOrder = "SELECT * FROM orders WHERE order_id = %s and user_id = %s"
        result_set = session.execute(queryOrder, [id_order, id_user])
        order = result_set.one()

        queryOrderDetails = "SELECT * FROM order_details WHERE order_id = %s"
        order_details = session.execute(queryOrderDetails, [id_order])

        if request.method == 'POST':
            form = UpdateOrderStatusForm(request.POST)
            if form.is_valid():
                new_status = form.cleaned_data['status']

                queryUpdateOrder = "UPDATE orders SET status = %s WHERE order_id = %s and user_id = %s"
                session.execute(queryUpdateOrder, [new_status, id_order, id_user])

                return HttpResponseRedirect('/admin/orders')
            else:
                form = UpdateOrderStatusForm(initial={'status': order.status})
        finally:
            close_connection()

    return render(request, 'admin/detail_order.html', {'order': order, 'order_details': order_details, 'form': form})
```

Hình 4. 24 Code cập nhật trạng thái đơn hàng

Hàm detail_order mở kết nối đến cơ sở dữ liệu để lấy thông tin chi tiết về đơn hàng dựa trên id_order và id_user, cũng như các sản phẩm liên quan trong đơn hàng đó. Nếu yêu

cầu là phương thức POST, hàm sẽ cập nhật trạng thái đơn hàng nếu biểu mẫu (UpdateOrderStatusForm) hợp lệ; nếu không, nó khởi tạo biểu mẫu với trạng thái hiện tại của đơn hàng. Cuối cùng, hàm đóng kết nối và trả về trang detail_order.html với thông tin đơn hàng, chi tiết đơn hàng và biểu mẫu.

4.1.8. Chức năng quản lý loại sản phẩm

4.1.8.1. Hiện thị danh sách các loại sản phẩm

Danh sách loại sản phẩm		
Thêm loại phẩm mới		
Tên Loại	Tác vụ	
Mặt nạ	Xem chi tiết	Sửa Xóa
Sản phẩm tóc	Xem chi tiết	Sửa Xóa
Kem mặt	Xem chi tiết	Sửa Xóa
Sữa rửa mặt	Xem chi tiết	Sửa Xóa
Son dưỡng	Xem chi tiết	Sửa Xóa

« 1 2 »

Hình 4. 25 Giao diện danh sách loại sản phẩm

Đầu tiên, hàm sẽ thực hiện truy vấn lấy tất cả loại.

```
# Truy vấn lấy tất cả loại
query = "SELECT * FROM category"
rows = session.execute(query)
categorys = list(rows) # Chuyển đổi kết quả truy vấn thành danh sách các thương hiệu
```

Sau đó, trả về dưới dạng mảng để hiển thị ra loại sản phẩm

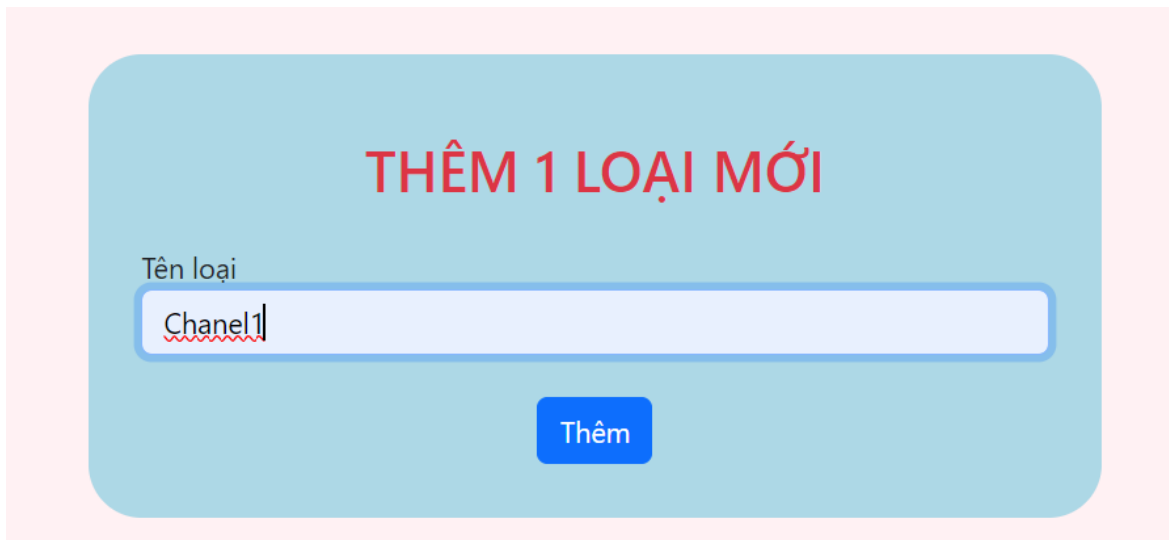
```
# Dữ liệu truyền vào template
data = {
    'listCategory': page_obj, # Danh sách thương hiệu của trang hiện tại
    'search_query': search_query, # Từ khóa tìm kiếm (để giữ lại trên giao diện)
    'sort_option': sort_option, # Tùy chọn sắp xếp (để giữ lại trên giao diện)
    'page_range': page_range, # Phạm vi các số trang hiển thị
    'total_pages': total_pages, # Tổng số trang
    'previous_page': previous_page, # Trang trước đó
    'next_page': next_page, # Trang tiếp theo
}

# Đóng kết nối Cassandra
close_connection()

return render(request, 'admin/categorys.html', data)
```

Hình 4. 26 Code hiển thị danh sách loại sản phẩm

4.1.8.2. Thêm một loại sản phẩm



Hình 4. 27 Giao diện thêm 1 loại sản phẩm

```
# Chèn dữ liệu vào bảng brand
print("Thêm loại với dữ liệu:", description, image_url, name) # Log dữ liệu
query = """
INSERT INTO category (category_id, description, image, name)
VALUES (uuid(), %s, %s, %s)
"""

session.execute(query, (description, image_url, name)) # Đảm bảo thứ tự tham số khớp với câu lệnh SQL
print("Loại đã được thêm thành công!") # Log thông báo thành công

return redirect('/admin/categories/') # Chuyển hướng về trang danh sách loại
```

Hình 4. 28 Code thực hiện thêm 1 loại sản phẩm

Hàm `add_category` thực hiện việc thêm một danh mục mới vào cơ sở dữ liệu Cassandra. Đầu tiên, nó mở kết nối và kiểm tra xem yêu cầu là phương thức POST hay không. Nếu có, nó xác thực dữ liệu từ form và xử lý ảnh (nếu có), sau đó chèn thông tin danh mục vào bảng `category`. Nếu form không hợp lệ hoặc có lỗi, thông báo lỗi sẽ được in ra, và cuối cùng, hàm sẽ đóng kết nối và trả về trang thêm danh mục.

4.1.8.3. Cập nhật một loại sản phẩm

```

try:
    # Lấy thương hiệu cần chỉnh sửa, nếu không tồn tại sẽ trả về 404
    query = "SELECT * FROM category WHERE category_id = %s"
    category = session.execute(query, (id_category,)).one()

    if category is None:
        raise Http404("Loại không tồn tại.")

    if request.method == 'POST':
        # Cập nhật thông tin thương hiệu từ form
        form = CategoryForm(request.POST) # Tạo form mới từ request.POST
        if form.is_valid():
            # Lấy dữ liệu từ form
            category_data = form.cleaned_data
            name = category_data['name']
            description = category_data['description']

            try:
                # Cập nhật dữ liệu vào bảng category
                update_query = """
                UPDATE category
                SET name = %s, description = %s
                WHERE category_id = %s
                """
                session.execute(update_query, (name, description, id_category))

                # Thêm thông báo thành công
                messages.success(request, 'Loại đã được cập nhật thành công!')
                return redirect('/admin/categorys')

            except Exception as e:
                messages.error(request, f"Có lỗi xảy ra: {str(e)}")

```

Hình 4. 29 Code thực hiện cập nhật 1 loại sản phẩm

Hàm `edit_category` cho phép người dùng chỉnh sửa thông tin của một danh mục cụ thể trong cơ sở dữ liệu Cassandra. Nó mở kết nối, kiểm tra xem danh mục có tồn tại hay không, và nếu yêu cầu là phương thức POST, nó xác thực dữ liệu từ form để cập nhật thông tin danh mục. Nếu thành công, một thông báo sẽ được hiển thị, và nếu không, thông báo lỗi sẽ được in ra. Cuối cùng, hàm đóng kết nối và render trang chỉnh sửa danh mục với dữ liệu hiện tại.

4.1.8.4. Xóa một loại sản phẩm



Hình 4. 30 Giao diện xóa loại sản phẩm

```
# # #xóa sản phẩm
def delete_category(request, id_category):
    # Mở kết nối đến Cassandra
    open_connection()

    # Lấy thương hiệu cần xóa
    query = "SELECT * FROM category WHERE category_id = %s"
    category = session.execute(query, (id_category,)).one()

    if category is None:
        # Nếu không tìm thấy loại, trả về 404
        messages.error(request, "Loại không tồn tại.")
        return redirect('/admin/categories')

    if request.method == 'POST':
        try:
            # Xóa thương hiệu khỏi bảng category
            delete_query = "DELETE FROM category WHERE category_id = %s"
            session.execute(delete_query, (id_category,))
            messages.success(request, 'Loại đã được xóa thành công!')
            return redirect('/admin/categories') # Chuyển hướng về trang danh sách loại

        except Exception as e:
            messages.error(request, f"Có lỗi xảy ra khi xóa loại: {str(e)}")

    # Truyền dữ liệu thương hiệu vào context nếu phương thức là GET
    form = {
        'name': category.name,
        'description': category.description
    }

    # Render form xác nhận xóa thương hiệu
    return render(request, 'admin/delete_category.html', {'form': form, 'category': category})
```

Hình 4. 31 Code thực hiện xóa loại sản phẩm

Hàm `edit_category` cho phép người dùng chỉnh sửa thông tin của một danh mục cụ thể trong cơ sở dữ liệu Cassandra. Nó mở kết nối, kiểm tra xem danh mục có tồn tại hay không, và nếu yêu cầu là phương thức POST, nó xác thực dữ liệu từ form để cập nhật thông tin danh mục. Nếu thành công, một thông báo sẽ được hiển thị, và nếu không, thông báo lỗi

sẽ được in ra. Cuối cùng, hàm đóng kết nối và render trang chỉnh sửa danh mục với dữ liệu hiện tại.

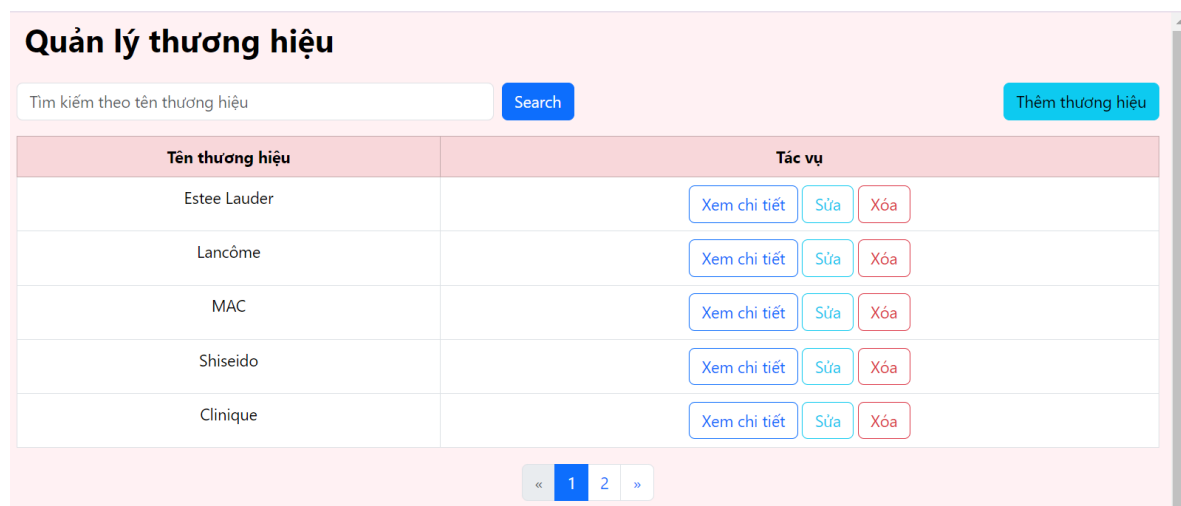
4.1.9. Chức năng quản lý thương hiệu

4.1.9.1. Hiện thị danh sách các thương hiệu

```
# Truy vấn lấy tất cả thương hiệu
query = "SELECT * FROM brand"
rows = session.execute(query)
brands = list(rows) # Chuyển đổi kết quả truy vấn thành danh sách các thương hiệu
```

Hình 4. 32 Code thực hiện lấy danh sách thương hiệu

Hàm brands lấy danh sách thương hiệu từ cơ sở dữ liệu Cassandra, cho phép tìm kiếm và sắp xếp theo tên. Nó cũng thực hiện phân trang để hiển thị một số lượng thương hiệu nhất định trên mỗi trang. Cuối cùng, hàm trả về trang brands.html với dữ liệu cần thiết cho giao diện người dùng.



Hình 4. 33 Giao diện quản lý thương hiệu

4.1.9.2. Thêm một thương hiệu sản phẩm

```
# Chèn dữ liệu vào bảng brand
print("Thêm thương hiệu với dữ liệu:", description, image_url, name) # Log dữ liệu
query = """
INSERT INTO brand (brand_id, description, image, name)
VALUES (uuid(), %s, %s, %s)
"""

session.execute(query, (description, image_url, name)) # Đảm bảo thứ tự tham số khớp với câu lệnh SQL
print("Thương hiệu đã được thêm thành công!") # Log thông báo thành công

return redirect('/admin/brands/') # Chuyển hướng về trang danh sách thương hiệu
lse:
print("Form is not valid:", form.errors)
```

Hình 4. 34 Code thực hiện thêm thương hiệu

Hàm `add_brand` xử lý việc thêm thương hiệu mới vào cơ sở dữ liệu Cassandra thông qua một form. Khi nhận được yêu cầu POST, nó xác thực form, xử lý ảnh nếu có, và chèn dữ liệu thương hiệu vào bảng. Cuối cùng, hàm chuyển hướng người dùng về trang danh sách thương hiệu và hiển thị thông báo thành công hoặc lỗi nếu có.

4.1.9.3. Cập nhật một thương hiệu sản phẩm

```
# Lấy thương hiệu cần chỉnh sửa, nếu không tồn tại sẽ trả về 404
query = "SELECT * FROM brand WHERE brand_id = %s"
brand = session.execute(query, (id_brand,)).one()

if brand is None:
    raise Http404("Thương hiệu không tồn tại.")

if request.method == 'POST':
    # Cập nhật thông tin thương hiệu từ form
    form = BrandForm(request.POST) # Tạo form mới từ request.POST
    if form.is_valid():
        # Lấy dữ liệu từ form
        brand_data = form.cleaned_data
        name = brand_data['name']
        description = brand_data['description']

        try:
            # Cập nhật dữ liệu vào bảng brand
            update_query = """
            UPDATE brand
            SET name = %s, description = %s
            WHERE brand_id = %s
            """
            session.execute(update_query, (name, description, id_brand))

            # Thêm thông báo thành công
            messages.success(request, 'Thương hiệu đã được cập nhật thành công!')
            return redirect('/admin/brands')

        except Exception as e:
            messages.error(request, f"Có lỗi xảy ra: {str(e)}")
    else:
```

Hình 4. 35 Code thực hiện cập nhật thương hiệu

Hàm `add_brand` xử lý việc thêm thương hiệu mới vào cơ sở dữ liệu Cassandra thông qua một form. Khi nhận được yêu cầu POST, nó xác thực form, xử lý ảnh nếu có, và chèn dữ liệu thương hiệu vào bảng. Cuối cùng, hàm chuyển hướng người dùng về trang danh sách thương hiệu và hiển thị thông báo thành công hoặc lỗi nếu có.

4.1.9.4. Xóa một thương hiệu sản phẩm

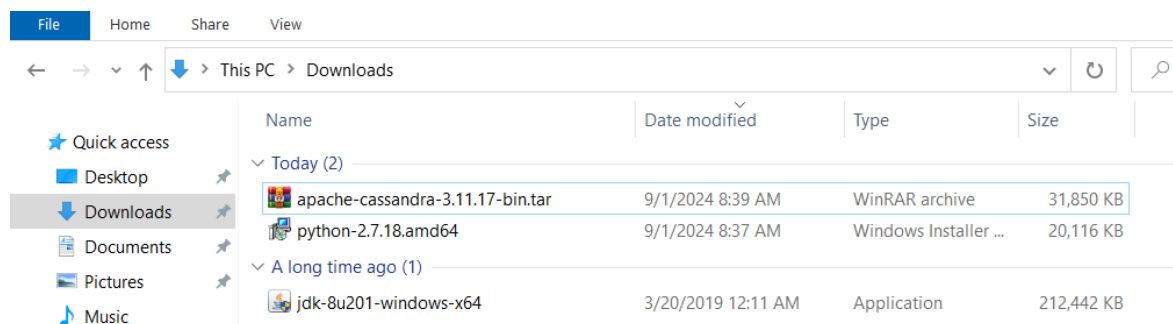
Hình 4. 36 Code thực hiện xóa thương hiệu

Chương 5. CÀI ĐẶT

5.1. Cài đặt

Bước 1: Chuẩn bị

Để cài đặt được cassandra ta cần chuẩn bị đầy đủ: python 2.7.18, java jdk8u201 và file cassandra

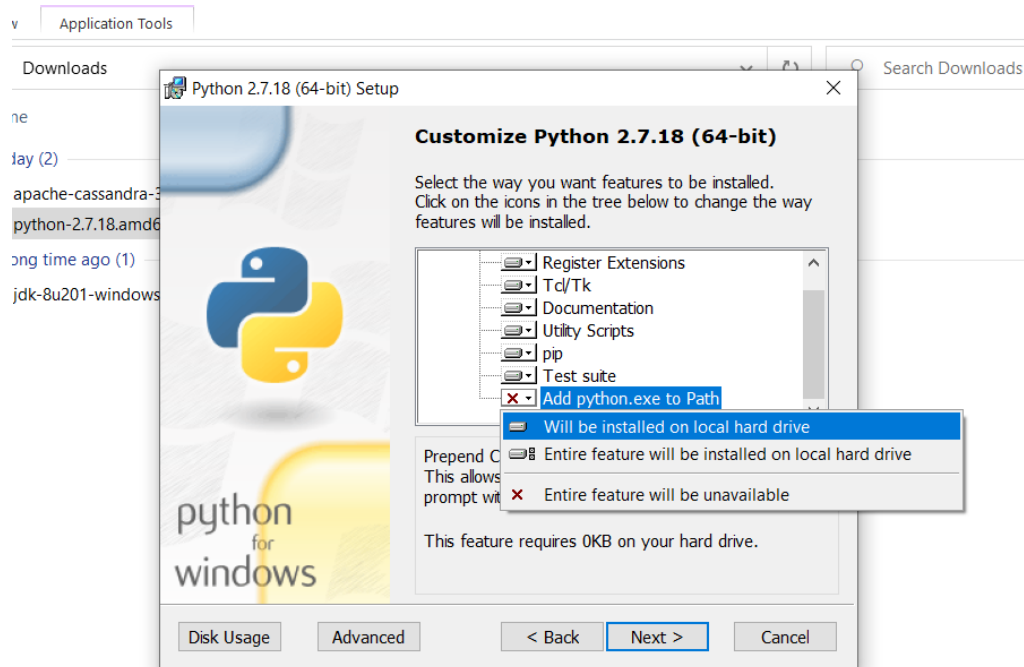


Hình 5. 1 Chuẩn bị file cài cần thiết để cài đặt Cassandra

Bước 2. Cài đặt

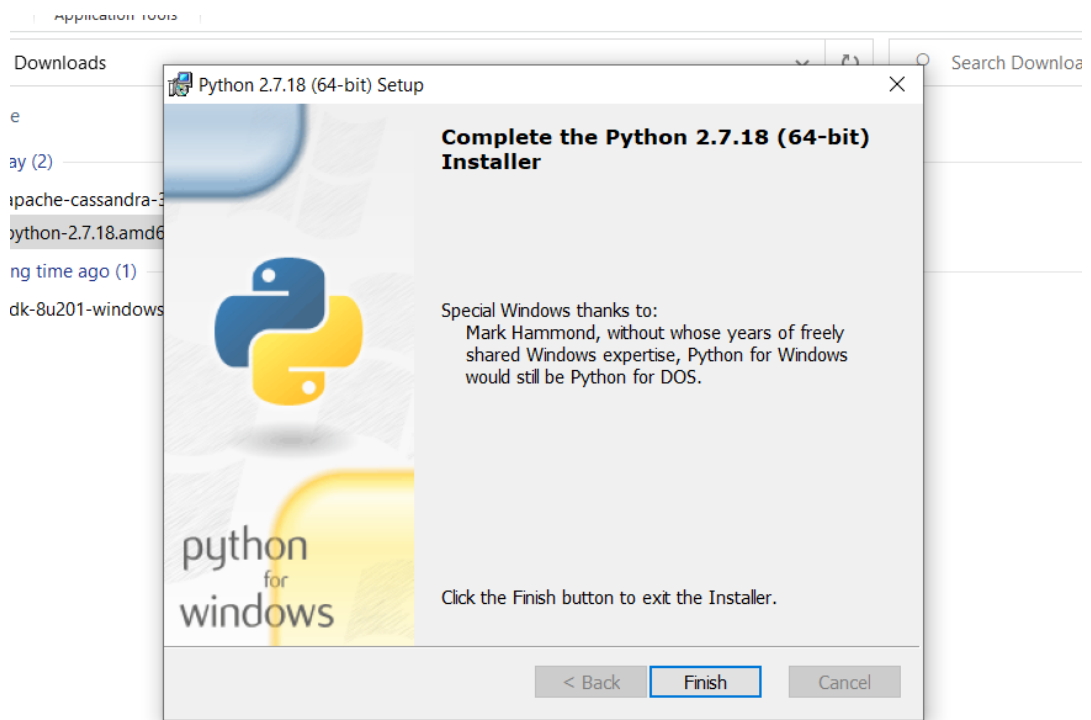
*Cài đặt python

Chạy file setup và chọn next.



Hình 5. 2 Lưu ý khi cài đặt môi trường python

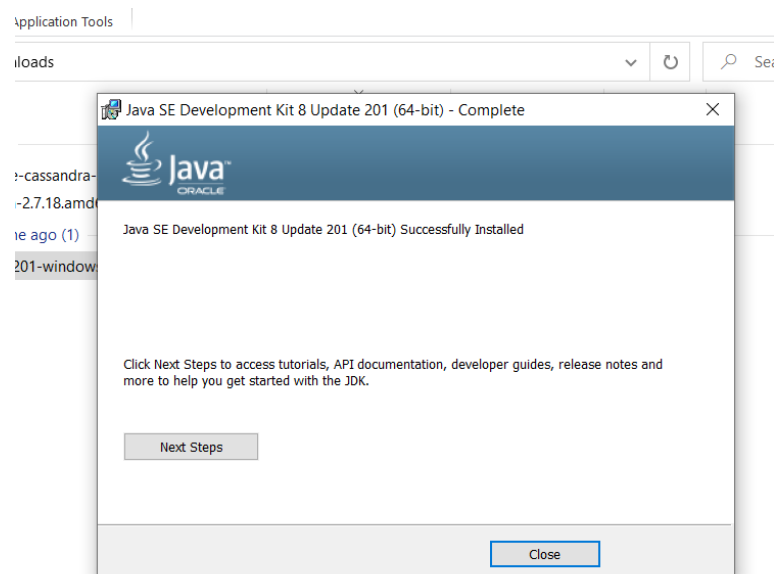
Lưu ý ở bước này chỗ Add python.exe to Path phải chọn là Will be installed on local hard drive (đăng ký biến môi trường toàn cục)



Hình 5. 3 Màn hình thông báo python đã được cài thành công

***Cài đặt java jdk**

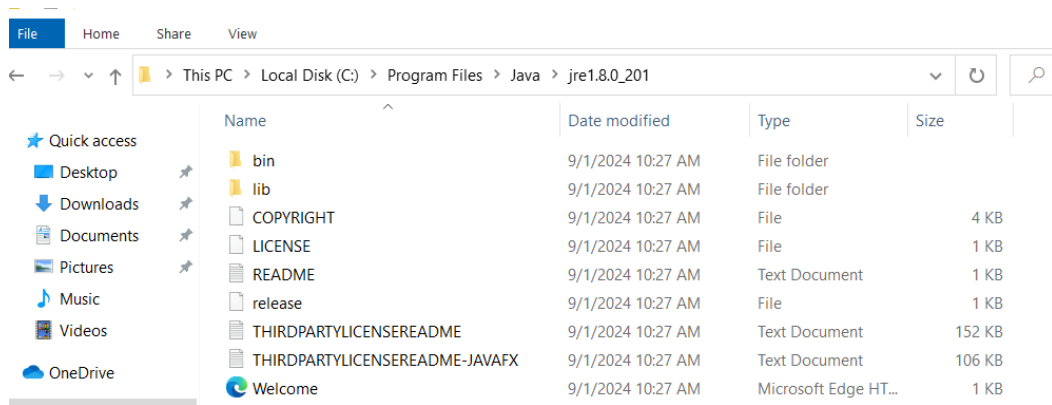
Chạy file setup và chọn Next



Hình 5. 4 Màn hình thông báo java jdk được cài thành công

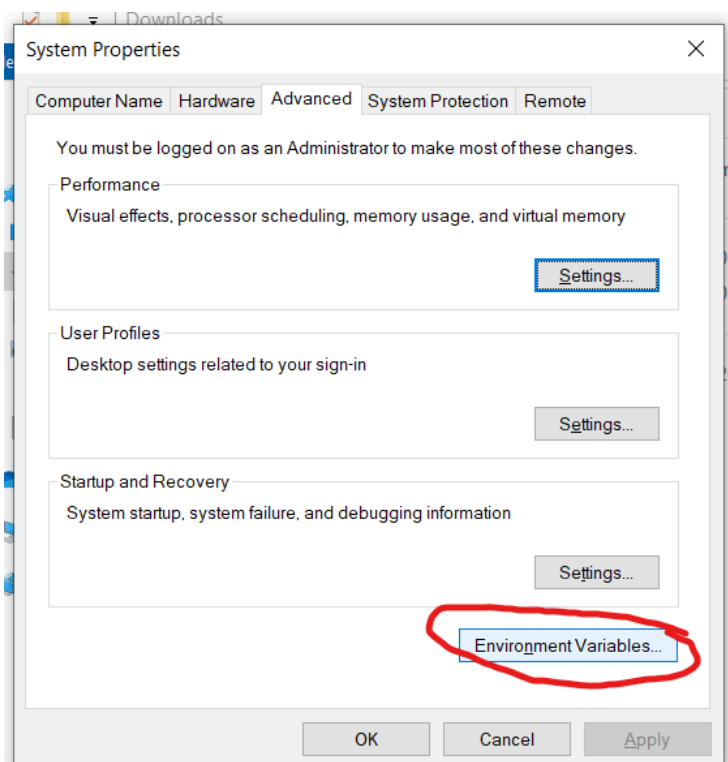
***Cài đặt biến môi trường cho java jdk**

- Vào đường dẫn đã lưu java jdk, chọn tới thư mục jre1.8.0_201 và copy đường dẫn



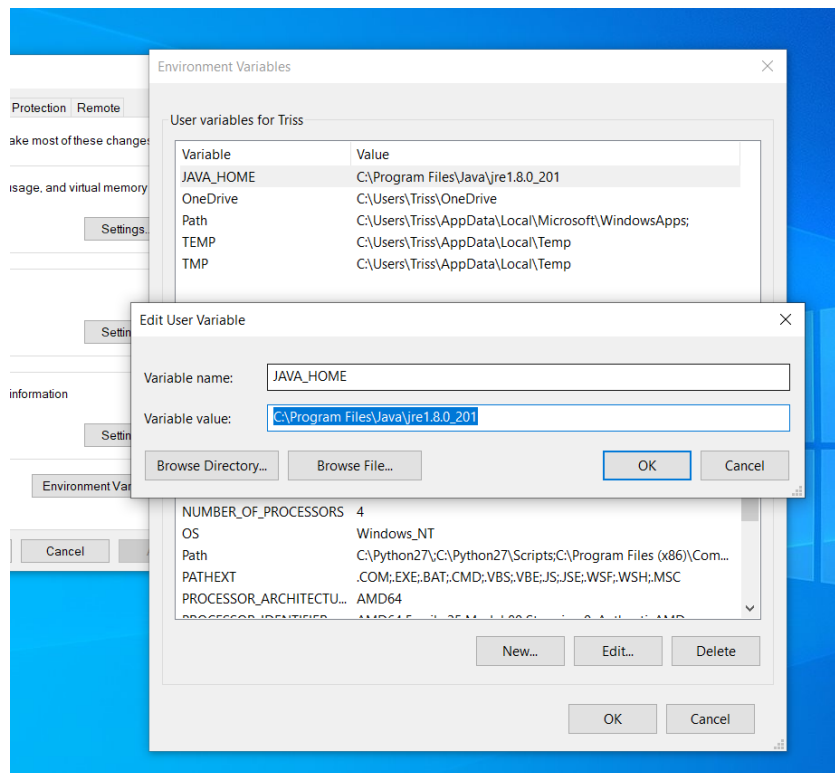
Hình 5. 5 Tìm đường dẫn đến thư mục java jre

- Vào Setting => System Properties => Advanced => Enviroment Variables



Hình 5. 6 Thiết lập môi trường trong enviroment variables

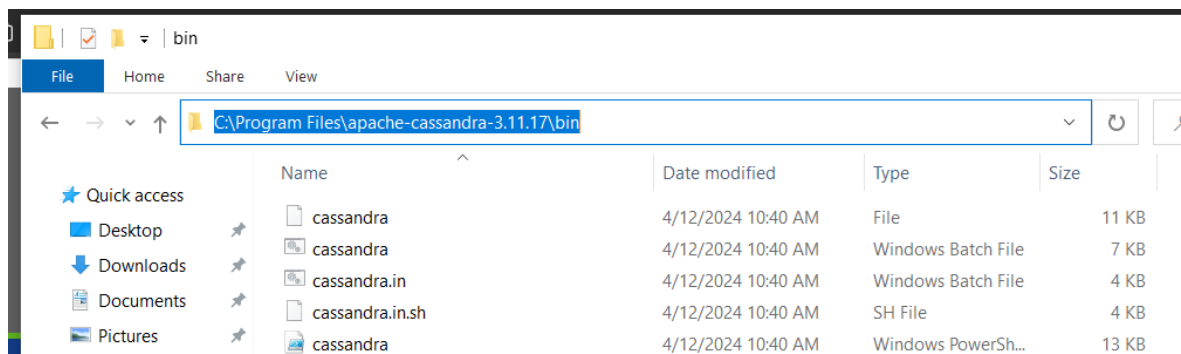
- Tạo biến môi trường JAVA_HOME



Hình 5. 7 Thiết lập môi biến môi trường cho

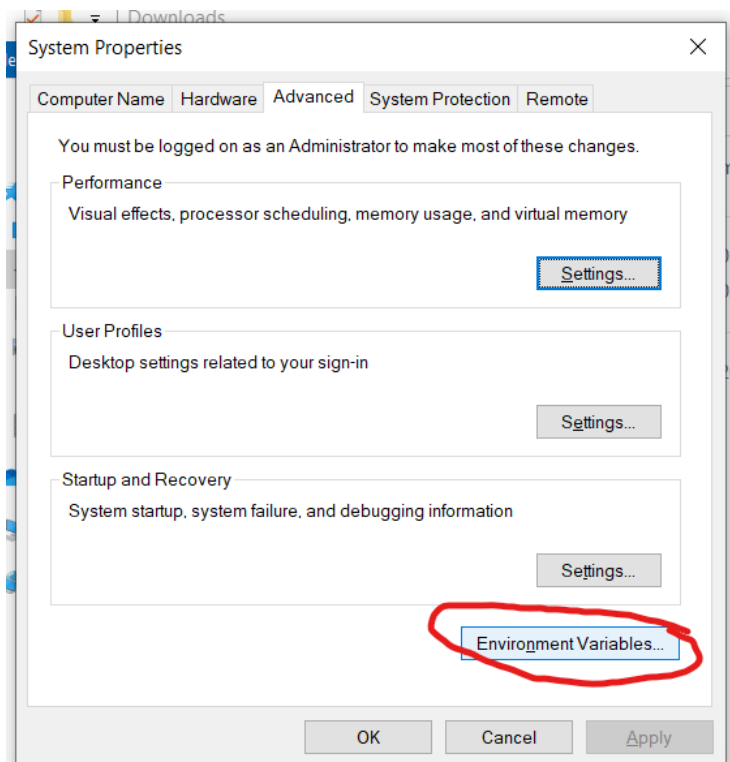
*** Cài đặt biến môi trường cho cassandra**

- Giải nén và copy thư mục apache cassandra vào ổ C
- Vào thư mục của cassandra và copy đường dẫn



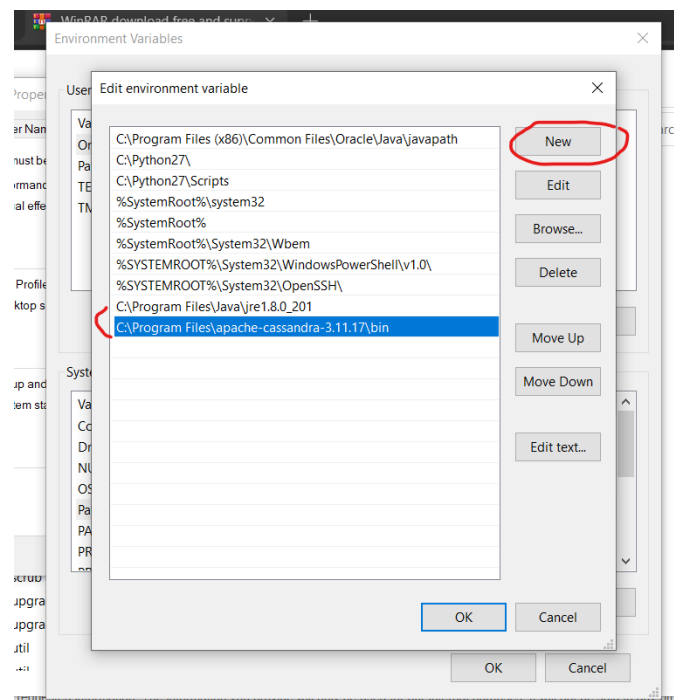
Hình 5. 8 Đường dẫn đến thư mục Cassandra

- Vào Setting => System Properties => Advanced => Enviroment Variables



Hình 5. 9 Thiết lập môi trường trong enviroment variables

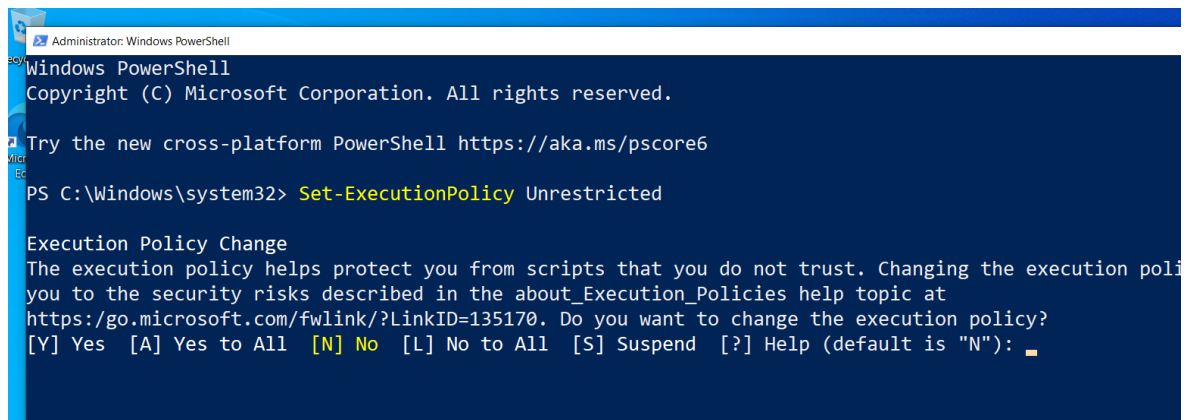
- Khi cửa sổ Path bật lên, Chọn New để thêm Path mới và copy đường dẫn lúc này vừa copy vào. Sau đó chọn OK



Hình 5. 10 Thêm path mới cho biến môi trường

* Cài đặt Policy cho PowerShell

- Tại cửa sổ của PowerShell, gõ lệnh Set-ExecutionPolicy Unrestricted, sau đó gõ Y để accept không hạn chế.



```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

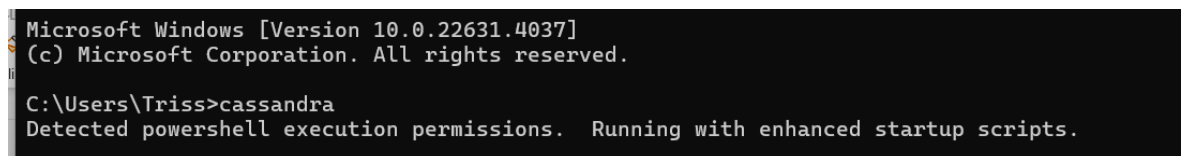
Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Windows\system32> Set-ExecutionPolicy Unrestricted

Execution Policy Change
The execution policy helps protect you from scripts that you do not trust. Changing the execution policy
you to the security risks described in the about_Execution_Policies help topic at
https://go.microsoft.com/fwlink/?LinkID=135170. Do you want to change the execution policy?
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "N"): 
```

Hình 5. 11 Cấu hình cho phép policy

* Khởi động Cassandra

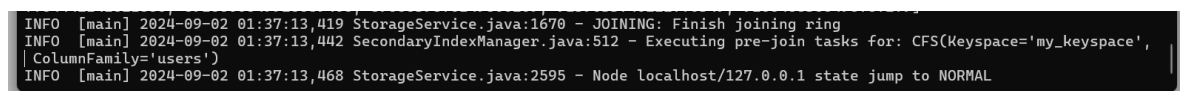


```
Microsoft Windows [Version 10.0.22631.4037]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Triss>cassandra
Detected powershell execution permissions. Running with enhanced startup scripts.
```

Hình 5. 12 Lệnh khởi động Cassandra

- Nếu có đoạn lệnh sau là server cassandra đã start thành công

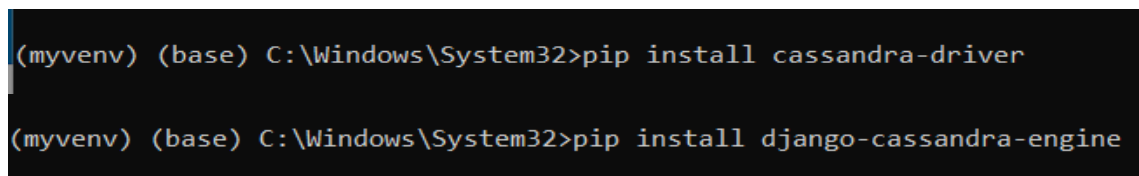


```
INFO [main] 2024-09-02 01:37:13,419 StorageService.java:1670 - JOINING: Finish joining ring
INFO [main] 2024-09-02 01:37:13,442 SecondaryIndexManager.java:512 - Executing pre-join tasks for: CFS(Keyspace='my_keyspace',
| ColumnFamily='users')
INFO [main] 2024-09-02 01:37:13,468 StorageService.java:2595 - Node localhost/127.0.0.1 state jump to NORMAL
```

Hình 5. 13 Màn hình thông báo Cassandra chạy thành công

5.2. Kết nối python django với cơ sở dữ liệu Cassandra

- Cài 2 thư viện cassandra driver và django-cassandra-engine



```
(myvenv) (base) C:\Windows\System32>pip install cassandra-driver
(myvenv) (base) C:\Windows\System32>pip install django-cassandra-engine
```

Hình 5. 14 Cài thư viện để kết nối Cassandra

- Cấu hình trong settings.py của ứng dụng web

```
INSTALLED_APPS = [  
    "django.contrib.admin",  
    "django.contrib.auth",  
    "django.contrib.contenttypes",  
    "django.contrib.sessions",  
    "django.contrib.messages",  
    "django.contrib.staticfiles",  
    "CuaHangMyPham",  
    "django_cassandra_engine",  
]
```

Hình 5. 15 Khai báo thư viện sử dụng

```
CASSANDRA_FALLBACK_ORDER_BY_PYTHON = True  
SESSION_ENGINE = 'django_cassandra_engine.sessions.backends.db'  
  
DATABASES = {  
    'default': {  
        'ENGINE': 'django_cassandra_engine',  
        'NAME': 'website_mypham',  
        'TEST_NAME': 'website_mypham',  
        'HOST': 'localhost',  
        'OPTIONS': {  
            'replication': {  
                'strategy_class': 'SimpleStrategy',  
                'replication_factor': 1  
            }  
        }  
    }  
}
```

Hình 5. 16 Khai báo cấu hình kết nối đến Cassandra

Chương 6: KẾT LUẬN – HƯỚNG PHÁT TRIỂN

Nhóm đã hoàn thành việc xây dựng website bán hàng mỹ phẩm Moon Shop với một số kết quả đáng chú ý. Website này không chỉ đáp ứng phong cách bán hàng trực tuyến mà còn tích hợp với hệ quản trị cơ sở dữ liệu Cassandra, giúp quản lý lượng dữ liệu lớn một cách hiệu quả và bảo đảm tốc độ truy cập cao.

Website đáp ứng đầy đủ các yêu cầu của một nền tảng dành cho người yêu thích làm đẹp, đặc biệt hướng tới đối tượng phái nữ với mong muốn trở nên quý phái và trẻ trung. Khách hàng có thể trải nghiệm việc xem hàng, đặt hàng, và mua hàng một cách thuận tiện, nhanh chóng. Cơ sở dữ liệu Cassandra được sử dụng để lưu trữ thông tin sản phẩm, khách hàng và đơn hàng, giúp tối ưu hóa quá trình truy xuất dữ liệu khi có khối lượng giao dịch lớn.

Ngoài ra, người quản trị của website có khả năng dễ dàng cập nhật thông tin sản phẩm, quản lý đơn hàng, khách hàng, và thực hiện các thống kê liên quan đến sản phẩm một cách chính xác nhờ vào khả năng mở rộng và phân tán của Cassandra. Điều này cho phép hệ thống hoạt động mượt mà ngay cả khi số lượng người dùng truy cập tăng đột biến.

Website có đầy đủ các chức năng cơ bản của một trang thương mại điện tử: bao gồm trang chủ, giới thiệu, sản phẩm, tin tức, thông tin liên hệ, chức năng giới thiệu sản phẩm, xem hàng, đặt hàng, mua hàng, góp ý kiến... Hệ thống cơ sở dữ liệu Cassandra đóng vai trò quan trọng trong việc lưu trữ và xử lý các giao dịch, đảm bảo tính toàn vẹn và hiệu suất cao cho hệ thống.

TÀI LIỆU THAM KHẢO

1. File bài tập thực hành môn Lập trình ngôn ngữ hiện đại Trường Đại học Công Thương TP HCM
2. <https://hiepsiit.com/>
3. <https://www.djangoproject.com/>
4. <https://www.w3schools.com/>
5. <https://getbootstrap.com/>