

ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG



BÁO CÁO

Học phần: Thực hành Cơ sở dữ liệu

Đề tài: Hệ thống quản lý khách sạn

Nhóm: Nhóm 1

Mã học phần: IT3290

Giảng viên hướng dẫn: TS. Trần Văn Đăng

Danh sách sinh viên thực hiện:

STT	Họ Tên	Mã sinh viên
1	Lê Văn Thành An	20236018
2	Lê Thành An	20235631
3	Trần Đức Nam Anh	20235655

Hà Nội, tháng 06 năm 2025

Mục lục

1	Giới thiệu đề tài	4
2	Phân công công việc	4
3	Phân tích yêu cầu hệ thống	4
3.1	Mô tả nghiệp vụ	4
3.2	Xác định thực thể	6
4	Thiết kế cơ sở dữ liệu	7
4.1	Sơ đồ thực thể liên kết (ER Diagram).....	7
4.2	Xác định các bảng trung gian	8
4.3	Xây dựng quan hệ.....	9
4.4	Relational Schema.....	10
5	Triển khai hệ thống	11
5.1	Công cụ sử dụng.....	11
5.2	Các câu lệnh SQL để tạo các quan hệ (bảng)	11
6	Truy vấn cho hệ thống	18
7	Hiệu năng	36

Lời nói đầu

Báo cáo này được thực hiện nhằm tổng kết quá trình học tập và nghiên cứu của nhóm sinh viên trong học phần **Thực hành cơ sở dữ liệu**. Trong suốt quá trình thực hiện đề tài "*Hệ thống quản lý khách sạn*", nhóm đã áp dụng những kiến thức lý thuyết đã học để phân tích, thiết kế và xây dựng hệ thống cơ sở dữ liệu thực tế.

Nhóm xin chân thành cảm ơn thầy **TS. Trần Văn Đặng** đã tận tình hướng dẫn và hỗ trợ nhóm trong suốt quá trình làm báo cáo. Mặc dù đã rất cố gắng, nhưng không thể tránh khỏi những thiếu sót. Nhóm mong nhận được sự góp ý của thầy để hoàn thiện hơn trong tương lai.

Hà Nội, tháng 06 năm 2025
Nhóm sinh viên thực hiện

1 Giới thiệu đề tài

Khách sạn là một trong những loại hình dịch vụ lưu trú phổ biến và phát triển mạnh mẽ trong xã hội hiện đại. Với nhu cầu ngày càng cao về nghỉ dưỡng, công tác và du lịch của người dân, các khách sạn không ngừng được đầu tư mở rộng cả về quy mô lẫn chất lượng dịch vụ. Tuy nhiên, để vận hành hiệu quả một khách sạn không chỉ đơn thuần là cung cấp phòng nghỉ, mà còn đòi hỏi khả năng quản lý chặt chẽ và chính xác nhiều loại thông tin như: đặt phòng, thông tin khách hàng, dịch vụ đi kèm, chương trình khuyến mãi, hóa đơn và doanh thu theo từng ngày, tuần hoặc tháng.

Trong bối cảnh đó, việc xây dựng một hệ thống quản lý khách sạn, có khả năng xử lý và lưu trữ dữ liệu lớn, là điều hết sức cần thiết. Một hệ thống như vậy không chỉ giúp tối ưu hóa quy trình vận hành, tiết kiệm thời gian và nhân lực, mà còn nâng cao trải nghiệm khách hàng thông qua các chức năng như đặt phòng trực tuyến, lựa chọn loại phòng linh hoạt, tích điểm thành viên, và tra cứu lịch sử giao dịch.

Đề tài này hướng đến việc thiết kế và xây dựng một hệ thống cơ sở dữ liệu phục vụ cho việc quản lý hoạt động của một khách sạn. Hệ thống bao gồm các chức năng chính như quản lý thông tin đặt phòng, quản lý khách hàng, theo dõi doanh thu, hỗ trợ các chương trình khuyến mãi và tích lũy điểm thưởng. Thông qua việc áp dụng các nguyên lý thiết kế cơ sở dữ liệu và sử dụng hệ quản trị cơ sở dữ liệu PostgreSQL, đề tài nhằm đảm bảo tính nhất quán, toàn vẹn dữ liệu, đồng thời mang lại hiệu năng cao khi vận hành trong thực tế.

2 Phân công công việc

Mỗi thành viên trong nhóm, ngoài đóng góp các truy vấn riêng, được phân công công việc sơ bộ như sau:

STT	Họ tên	Nội dung công việc
1	Lê Văn Thành An	Phân tích yêu cầu, nghiệp vụ hệ thống. xây dựng truy vấn liên quan đến nghiệp vụ khách hàng, tạo function, trigger hỗ trợ việc insert dữ liệu. Xây dựng demo cho dự án
2	Lê Thành An	Phân tích yêu cầu, nghiệp vụ hệ thống. Xây dựng ERD, relational schema, xây dựng truy vấn liên quan đến nghiệp vụ admin.
3	Trần Đức Nam Anh	Phân tích yêu cầu, nghiệp vụ hệ thống. Xây dựng ERD, tạo các bảng dữ liệu mẫu, xây dựng truy vấn liên quan đến nghiệp vụ admin.

3 Phân tích yêu cầu hệ thống

3.1 Mô tả nghiệp vụ

Hệ thống quản lý khách sạn được xây dựng với mục tiêu hỗ trợ đầy đủ các nghiệp vụ vận hành, đồng thời nâng cao trải nghiệm của khách hàng. Hệ thống bao gồm các chức năng dành cho cả người dùng (khách hàng) và bộ phận quản trị hệ thống.

Khách hàng có thể **tạo tài khoản, đăng nhập vào hệ thống** và thực hiện các thao tác như **đặt phòng trực tuyến, xem thông tin phòng còn trống** (số lượng, giá cả, tiện ích nổi bật như view đẹp, có tivi, có bồn tắm,...), **tra cứu các chương trình ưu đãi và quyền lợi theo hạng thành viên**. Hệ thống cho phép khách hàng sử dụng bộ lọc để tìm kiếm loại phòng phù hợp (phòng đôi, phòng có giảm giá, mới tân trang,...).

Trong quá trình đặt phòng, **khách hàng** có thể áp dụng **mã khuyến mãi** nếu đủ điều kiện. Các ưu đãi này được kiểm tra tự động dựa trên thời gian lưu trú, thời điểm áp dụng và số lượt sử dụng. Khi thanh toán, hệ thống ghi nhận đầy đủ **thông tin hóa đơn**, bao gồm: thông tin khách hàng, thời gian lưu trú, dịch vụ sử dụng, phương thức thanh toán và số tiền. Hóa đơn được lưu vào lịch sử tài khoản để khách hàng dễ dàng tra cứu.

Việc hủy phòng cũng có thể thực hiện online. Lịch sử hủy phòng được ghi nhận đầy đủ (**thời gian, lý do**), và các tài khoản có dấu hiệu lạm dụng sẽ bị giới hạn đặt phòng.

Khách hàng có thể yêu cầu **sử dụng dịch vụ** trong thời gian lưu trú (food, spa, laundry...). Hệ thống ghi nhận dịch vụ sử dụng và **tự động cộng vào hóa đơn**. Dịch vụ được quản lý theo từng lần sử dụng, thống kê mức độ phổ biến và đánh giá của khách hàng. Các chương trình ưu đãi cho dịch vụ cũng được quản lý theo thời gian và đối tượng áp dụng.

Khách hàng có thể để lại **đánh giá sau kỳ nghỉ** (số sao, bình luận), các đánh giá này được liên kết với từng lượt đặt phòng cụ thể. Hệ thống thống kê điểm trung bình theo từng loại phòng và dịch vụ.

Lễ tân thực hiện các thao tác **check-in, check-out, xác nhận đặt phòng, cập nhật trạng thái phòng** (đã đặt, đã nhận, đã trả, đã hủy,...). Khi khách trả phòng, lễ tân kiểm tra tình trạng vật dụng, dịch vụ phát sinh (như minibar, spa, giặt là), cập nhật hóa đơn và thực hiện thanh toán.

Hệ thống hỗ trợ cảnh báo khi sắp hết loại phòng nhất định hoặc có phòng trống đột xuất, đồng thời cho phép đặt phòng theo số người, yêu cầu dịch vụ kèm theo.

Nhân viên kho kiểm tra **tình trạng vật tư dự phòng** (tivi, máy sấy, ấm siêu tốc, v.v...), cập nhật tồn kho theo từng loại, xử lý kịp thời các vật dụng hỏng, báo cáo định kỳ về thiết bị sắp hết hoặc cần thay thế. Khi nhập hàng, hệ thống lưu thông tin nhà cung cấp, số lượng, giá tiền và người phụ trách.

Mỗi lần thay vật tư trong phòng (sau khi khách trả), hệ thống cập nhật số lượng tồn kho, nhân viên thay và thời gian thực hiện.

Hệ thống ghi nhận toàn bộ giao dịch thanh toán từ khách hàng, bao gồm các hình thức: tiền mặt, thẻ, ví điện tử,... Hóa đơn thể hiện chi tiết giá phòng, thuế, dịch vụ kèm theo và mã khuyến mãi áp dụng. Ngoài ra, hệ thống còn thống kê doanh thu theo ngày, tuần, tháng, năm; so sánh doanh thu theo từng loại phòng, dịch vụ; tổng hợp chi phí vận hành và lợi nhuận thực tế.

Quản trị viên hệ thống có toàn quyền thêm/sửa/xóa thông tin người dùng, thiết lập các loại phòng, dịch vụ, tiện ích, chương trình khuyến mãi và chính sách hoàn/hủy phòng. Quản trị viên có thể theo dõi báo cáo đánh giá, phản hồi khách hàng và chủ động kiểm tra các phòng/dịch vụ bị đánh giá kém. Ngoài ra, hệ thống có thể tự động ghi nhận điểm tích lũy khi giao dịch thành công, hỗ trợ nâng hạng thành

3.2 Xác định thực thể

Hệ thống quản lý khách sạn bao gồm các thực thể chính phản ánh đầy đủ hoạt động vận hành và tương tác giữa khách hàng, nhân viên và hệ thống đặt phòng:

- **Customer:** Lưu trữ thông tin chi tiết về khách hàng bao gồm họ tên, số căn cước công dân, số điện thoại, email, quốc tịch, thông tin tài khoản và mức độ thành viên. Đây là thực thể trung tâm liên kết với các hoạt động đặt phòng, đánh giá và hóa đơn.
- **Room:** Đại diện cho các phòng trong khách sạn với các thuộc tính như số phòng, loại phòng, sức chứa, giá mỗi đêm, tiện nghi đi kèm, tình trạng sử dụng (trống, đang dùng, bảo trì) và số tầng. Mỗi phòng có thể được sử dụng trong nhiều lần đặt phòng khác nhau.
- **Booking:** Ghi nhận thông tin các lượt đặt phòng từ khách hàng bao gồm thời gian nhận và trả phòng, ngày đặt, yêu cầu đặc biệt, mã khuyến mãi (nếu có), trạng thái đặt phòng (đã hủy, đã xác nhận, đang sử dụng) và lý do hủy nếu có. Booking có quan hệ với Customer, Room, và Promotion.
- **Invoice:** Đại diện cho các hóa đơn phát sinh từ các lượt đặt phòng, ghi nhận ngày lập hóa đơn, phương thức thanh toán và trạng thái thanh toán. Chỉ các đặt phòng hợp lệ (không bị hủy) mới có hóa đơn tương ứng.
- **Service:** Mô tả các dịch vụ cung cấp trong khách sạn như spa, ẩm thực, giặt là,... với thông tin về tên, mô tả, giá tiền và loại dịch vụ. Mỗi dịch vụ được gắn với một booking cụ thể để ghi nhận sử dụng.
- **Promotion:** Lưu trữ các chương trình khuyến mãi có thời gian hiệu lực cụ thể, tỷ lệ giảm giá, yêu cầu về số ngày lưu trú tối thiểu và loại phòng áp dụng. Một số chương trình có thể áp dụng cho toàn bộ hệ thống.
- **Review:** Ghi nhận đánh giá của khách hàng sau khi lưu trú, bao gồm điểm đánh giá, nhận xét, ngày đánh giá và phản hồi của khách sạn (nếu có). Chỉ các đặt phòng không bị hủy mới có thể viết đánh giá.
- **Inventory:** Quản lý thông tin các vật dụng và thiết bị trong khách sạn như gối, tivi, bàn ủi,... với thông tin về số lượng tồn kho, đơn giá, nhà cung cấp và vị trí lưu trữ.
- **Room_Inventory:** Liên kết giữa các phòng và thiết bị vật tư đi kèm, giúp theo dõi các thiết bị hiện có trong từng phòng.

3.3 Phân quyền:

- Khách hàng:
 - Tạo và quản lý tài khoản cá nhân bằng username và password, có thể cập nhật và chỉnh sửa thông tin
 - Có thể truy cập danh sách các loại phòng đang có tại khách sạn, tìm hiểu thông tin cần thiết cho quá trình đặt phòng
 - Áp dụng các mã giảm giá, ưu đãi đặc biệt, thực hiện đặt phòng và

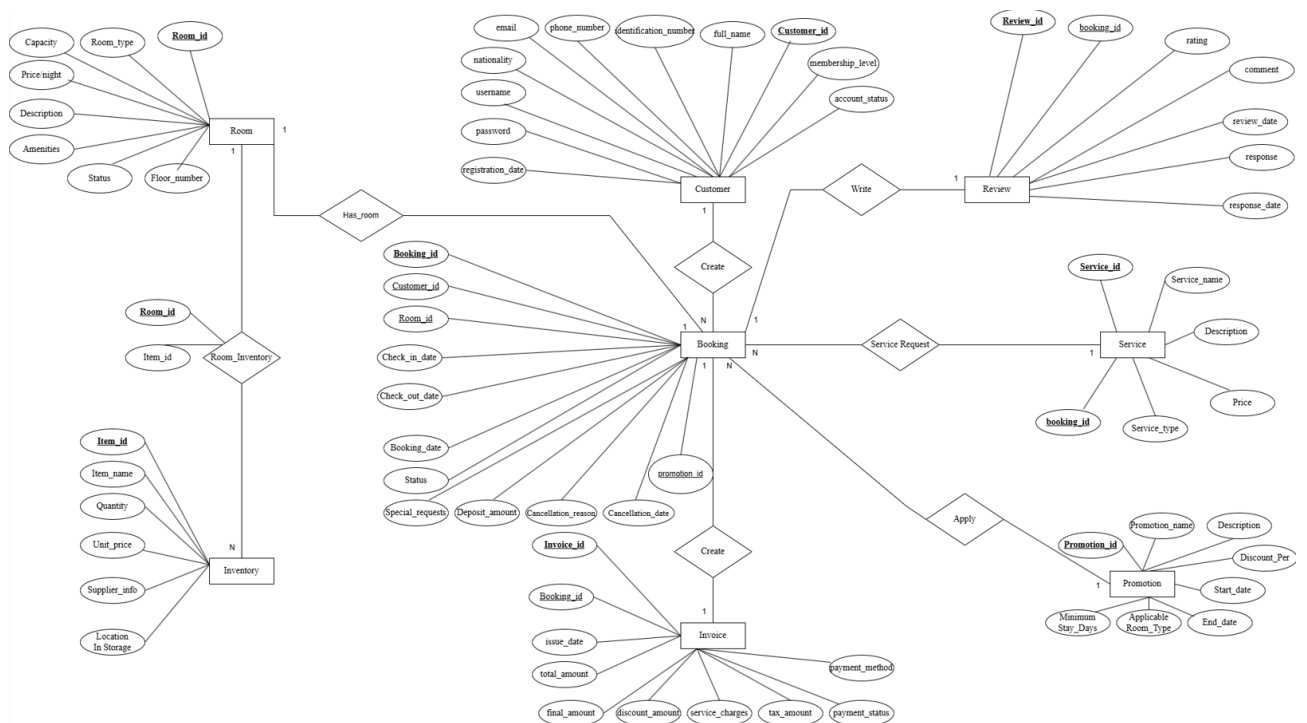
thanh toán

- Theo dõi lịch sử đặt phòng, kiểm tra trạng thái đặt phòng
 - Xem lại hoá đơn chi tiết về dịch vụ sử dụng cũng như các loại phí khác
 - Sau khi kết thúc thời gian đặt phòng, khách hàng có thể để lại đánh giá, nhận xét về dịch vụ, trải nghiệm của khách sạn
 - Khách hàng **không có quyền** truy cập hay chỉnh sửa thông tin của khách hàng khác, và không sử dụng các tính năng đặc trưng của quản lí
- Quản lí khách sạn:
 - Có thể nhìn thấy danh sách tài khoản, thông tin của khách hàng, những thông tin như mật khẩu sẽ không hiển thị
 - Truy cập và cập nhật những thông tin quan trọng liên quan đến phòng ở hay dịch vụ
 - Thay đổi trạng thái phòng sau khi khách check-out hay sau khi khách huỷ
 - Theo dõi chi tiết về vật tư, đồ dung trong phòng
 - Theo dõi và thống kê doanh thu, tỷ lệ đặt phòng, tỷ lệ khách vào từng tháng
 - Quản lí các chương trình khuyến mãi đang được áp dụng, có thể đưa ra những khuyến mãi phù hợp trong quá trình đặt phòng của khách
 - Xem và tổng hợp, phân tích các đánh giá, phản hồi từ khách
 - Tạo hoá đơn sau khi khách hàng check-out, cập nhật trạng thái thanh toán của khách

4 Thiết kế cơ sở dữ liệu

4.1 Sơ đồ thực thể liên kết (ER Diagram)

Dựa vào những thực thể đã xác định, nhóm đưa ra sơ đồ ER tương ứng như sau:



Hình 1: Sơ đồ ERD hệ thống quản lý khách sạn

4.2 Xác định các bảng trung gian

Sau khi xác định các mối quan hệ giữa các thực thể chính, hệ thống cần xây dựng các bảng trung gian để biểu diễn chính xác các ràng buộc và mối liên kết trong schema. Cụ thể:

- **Room_Inventory:** Là bảng trung gian giữa thực thể Room và Inventory, dùng để quản lý các vật dụng hoặc thiết bị cụ thể có trong từng phòng. Bảng này cho phép một vật dụng có thể được trang bị ở nhiều phòng, và mỗi phòng có thể chứa nhiều loại vật dụng khác nhau

. Các bảng trung gian này giúp chuẩn hóa dữ liệu và thể hiện rõ các mối quan hệ n-n (nhiều-nhiều) giữa các thực thể, đồng thời hỗ trợ việc quản lý thiết bị phòng một cách hiệu quả và linh hoạt

4.3 Xây dựng quan hệ

Dựa trên sơ đồ ER và thuộc tính, các bảng trung gian cũng như các quan hệ được xác định, nhóm đưa ra được các quan hệ như sau:

- **Customer(customer_id, full_name, identification_number, phone_number, email, nationality, membership_level, account_status, username, password, registration_date)**

Quản lý thông tin cá nhân, liên hệ và tài khoản của khách hàng. Mỗi khách hàng sẽ có một tài khoản riêng biệt để thực hiện đặt phòng và sử dụng dịch vụ.

- **Room(room_id, room_type, capacity, price_per_night, view_description, amenities, status, floor_number)**

Quản lý thông tin từng phòng trong khách sạn. Mỗi phòng có loại riêng, sức chứa nhất định, giá theo đêm, tiện nghi cụ thể và trạng thái hiện tại (trống, có khách, đang bảo trì)

- **RoomInventory(room_id, item_id)**
(Chữ đỏ: khoá chính, gạch chân: khoá ngoài)

Là bảng trung gian thể hiện mối quan hệ n-n giữa Room và Inventory. Một phòng có thể chứa nhiều vật dụng, và một vật dụng có thể xuất hiện trong nhiều phòng.

- **Booking(booking_id, customer_id, room_id, check_in_date, check_out_date, booking_date, status, special_requests, deposit_amount, cancellation_reason, cancellation_date)**

Ghi nhận mỗi lượt đặt phòng của khách hàng. Mỗi lượt booking gắn với một khách, một phòng và có thể áp dụng một khuyến mãi cụ thể. Trạng thái (đang sử dụng, bị huỷ) thể hiện việc sử dụng thực tế.

- **Review(review_id, booking_id, rating, comment, review_date, response, response_date)**

Ghi nhận đánh giá của khách hàng sau mỗi lượt đặt phòng thành công. Một booking chỉ có thể có một đánh giá nếu không bị huỷ.

- **Invoice(invoice_id, booking_id, issue_date, total_amount, payment_method, payment_status, tax_amount, service_charges, discount_amount, final_amount)**

Hóa đơn thanh toán được tạo sau khi booking hoàn tất và không bị huỷ. Tính toán tổng chi phí từ phòng, dịch vụ kèm theo, thuế, khuyến mãi và trạng thái thanh toán.

- **Service(service_id, booking_id, room_id, service_name, description, price, service_type)**

Danh sách các dịch vụ phát sinh trong thời gian khách ở lại. Mỗi service gắn với một booking và một phòng, mô tả dịch vụ như Spa, Ăn uống, Giặt là...

- **Promotion(promotion_id, promotion_name, description, discount_percentage, start_date, end_date, applicable_room_types, minimum_stay_days)**

Danh sách các chương trình khuyến mãi có thể áp dụng cho lượt đặt phòng. Mỗi chương trình có thời gian hiệu lực, điều kiện áp dụng và mức giảm giá khác nhau.

- **Inventory(item_id, item_name, quantity_in_stock, unit_price, supplier_info, location_in_storage)**

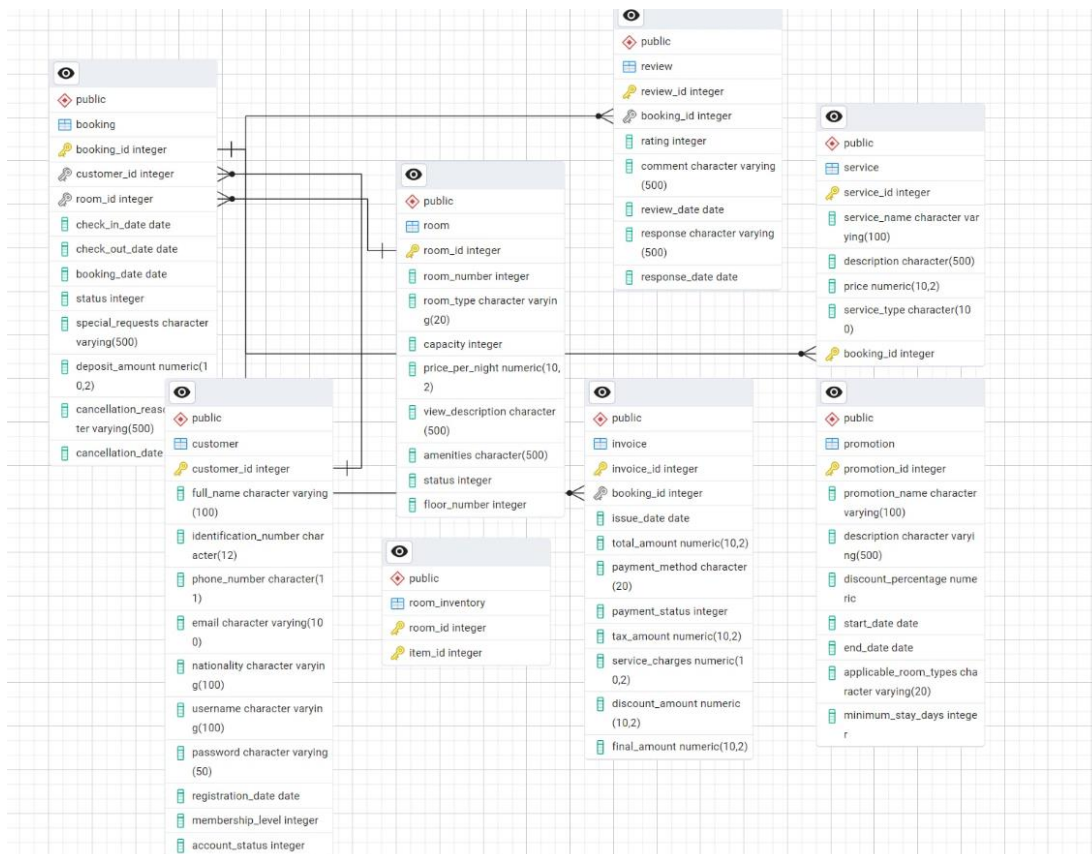
Danh mục các thiết bị, vật dụng trong khách sạn cùng thông tin tồn kho, nhà cung cấp và vị trí lưu trữ.

Loại liên kết giữa các thực thể

Thực thể A	Thực thể B	Quan hệ
Room	Inventory	N-N
Booking	Review	1-1
Booking	Invoice	1-1
Promotion	Booking	N-1
Booking	Service	N-1
Booking	Room	N-1
Customer	Booking	1-N

4.4 Relational Schema

Dựa trên các quan hệ đã xây dựng ở bước trên, tiến hành tạo relational schema tương ứng:



Lược đồ quan hệ của hệ thống quản lý khách sạn

5 Triển khai hệ thống

5.1 Công cụ sử dụng

Trong quá trình phát triển hệ thống, nhóm sử dụng **hệ quản trị cơ sở dữ liệu Post-greSQL** kết hợp với **ngôn ngữ truy vấn SQL** để thao tác và quản lý dữ liệu. Bên cạnh đó, **công cụ draw.io** được sử dụng để thiết kế sơ đồ thực thể liên kết (ERD) và relational schema.

5.2 Các câu lệnh SQL để tạo các quan hệ (bảng)

Nhằm đảm bảo tính chính xác và nhất quán, toàn vẹn giữa các bảng, cũng như hạn chế lỗi nhập liệu, các ràng buộc sau được sử dụng:

- NOT NULL: Đảm bảo không được để trống
- Ngoài ra còn cần đảm bảo ràng buộc về khóa chính, khóa ngoài giữa các bảng như ở mục 4.

Triển khai tạo quan hệ bằng SQL như sau:

Tạo bảng CUSTOMER:

```
CREATE TABLE IF NOT EXISTS public.customer
(
    customer_id integer NOT NULL,
    full_name character varying(100) COLLATE pg_catalog."default",
    identification_number character(12) COLLATE pg_catalog."default",
    phone_number character(11) COLLATE pg_catalog."default",
    email character varying(100) COLLATE pg_catalog."default",
    nationality character varying(100) COLLATE pg_catalog."default",
    username character varying(100) COLLATE pg_catalog."default",
    password character varying(50) COLLATE pg_catalog."default",
    registration_date date,
    membership_level integer DEFAULT 0,
    account_status integer,
    CONSTRAINT customer_pkey PRIMARY KEY (customer_id)
);
```

Tạo bảng INVENTORY

```
CREATE TABLE IF NOT EXISTS public.inventory
(
    item_id integer NOT NULL,
    item_name character varying(500) COLLATE pg_catalog."default",
    quantity_in_stock integer,
    unit_price numeric(10,2),
    supplier_info character varying COLLATE pg_catalog."default",
```

```

location_in_storage character varying(100) COLLATE pg_catalog."default",
CONSTRAINT inventory_pkey PRIMARY KEY (item_id)
);

```

Tạo bảng PROMOTION

```

CREATE TABLE IF NOT EXISTS public.promotion
(
    promotion_id integer NOT NULL,
    promotion_name character varying(100) COLLATE pg_catalog."default",
    description character varying(500) COLLATE pg_catalog."default",
    discount_percentage numeric,
    start_date date,
    end_date date,
    applicable_room_types character varying(20) COLLATE pg_catalog."default",
    minimum_stay_days integer,
    CONSTRAINT promotion_pkey PRIMARY KEY (promotion_id)
);

```

Tạo bảng ROOM

```

CREATE TABLE IF NOT EXISTS public.room
(
    room_id integer NOT NULL,
    room_number integer,
    room_type character varying(20) COLLATE pg_catalog."default",
    capacity integer,
    price_per_night numeric(10,2),
    view_description character(500) COLLATE pg_catalog."default",
    amenities character(500) COLLATE pg_catalog."default",
    status integer,
    floor_number integer,
    CONSTRAINT room_pkey PRIMARY KEY (room_id)
);

```

Tạo bảng BOOKING

```

CREATE TABLE IF NOT EXISTS public.booking
(
    booking_id integer NOT NULL,
    customer_id integer,
    room_id integer,
    promotion_id integer,
    check_in_date date,
    check_out_date date,
    booking_date date,
    status integer,
    special_requests character varying(500) COLLATE pg_catalog."default",
    deposit_amount numeric(10,2),

```

```

cancellation_reason character varying(500) COLLATE pg_catalog."default",
cancellation_date date,
CONSTRAINT booking_pkey PRIMARY KEY (booking_id),
CONSTRAINT booking_customer_id_fkey FOREIGN KEY (customer_id)
    REFERENCES public.customer (customer_id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID,
CONSTRAINT booking_promotion_id_fkey FOREIGN KEY (promotion_id)
    REFERENCES public.promotion (promotion_id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID,
CONSTRAINT booking_room_id_fkey FOREIGN KEY (room_id)
    REFERENCES public.room (room_id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID
);

```

Tạo bảng INVOICE

```

CREATE TABLE IF NOT EXISTS public.invoice
(
    invoice_id integer NOT NULL,
    booking_id integer,
    issue_date date,
    total_amount numeric(10,2),
    payment_method character(20) COLLATE pg_catalog."default",
    payment_status integer,
    tax_amount numeric(10,2),
    service_charges numeric(10,2),
    discount_amount numeric(10,2),
    final_amount numeric(10,2),
    CONSTRAINT invoice_pkey PRIMARY KEY (invoice_id),
    CONSTRAINT invoice_booking_id_fkey FOREIGN KEY (booking_id)
        REFERENCES public.booking (booking_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
);

```

Tạo bảng REVIEW

```

CREATE TABLE IF NOT EXISTS public.review
(
    review_id integer NOT NULL,
    booking_id integer,
    rating integer,
    comment character varying(500) COLLATE pg_catalog."default",
    review_date date,

```

```

response character varying(500) COLLATE pg_catalog."default",
response_date date,
CONSTRAINT review_pkey PRIMARY KEY (review_id),
CONSTRAINT review_booking_id_fkey FOREIGN KEY (booking_id)
REFERENCES public.booking (booking_id) MATCH SIMPLE
ON UPDATE NO ACTION
ON DELETE NO ACTION
);

```

Tạo bảng ROOM_INVENTORY

```

CREATE TABLE IF NOT EXISTS public.room_inventory
(
    room_id integer NOT NULL,
    item_id integer NOT NULL,
    CONSTRAINT room_inventory_pkey PRIMARY KEY (room_id, item_id)
);

```

Tạo bảng SERVICE

```

CREATE TABLE IF NOT EXISTS public.service
(
    service_id integer NOT NULL,
    service_name character varying(100) COLLATE pg_catalog."default",
    description character(500) COLLATE pg_catalog."default",
    booking_id integer,
    price numeric(10,2),
    service_type character(100) COLLATE pg_catalog."default",
    CONSTRAINT service_pkey PRIMARY KEY (service_id),
    CONSTRAINT booking_booking_id_fkey FOREIGN KEY (booking_id)
REFERENCES public.booking (booking_id) MATCH SIMPLE
ON UPDATE NO ACTION
ON DELETE NO ACTION
NOT VALID
)

```

Sau khi tạo bảng, nhóm thực hiện thêm dữ liệu vào hệ thống bằng các câu lệnh INSERT INTO <tên bảng> nhằm kiểm tra các truy vấn ở bước tiếp theo.

- TRIGGER và FUNCTION tính giá tiền trong invoice:

```

CREATE OR REPLACE FUNCTION calculate_invoice_amounts()
RETURNS TRIGGER AS
$$
DECLARE
    room_price_per_night NUMERIC(10, 2);
    check_in_date DATE;
    check_out_date DATE;
    stay_days INTEGER;

```

```

room_type_value VARCHAR(50);
calculated_total_amount NUMERIC(10, 2);
calculated_tax_amount NUMERIC(10, 2);
calculated_service_charges NUMERIC(10, 2);
calculated_discount_amount NUMERIC(10, 2);
calculated_final_amount NUMERIC(10, 2);
applicable_discount_rate NUMERIC(5, 2); -- phần trăm giảm giá
BEGIN
SELECT
    r.price_per_night,
    b.check_in_date,
    b.check_out_date,
    r.room_type
INTO
    room_price_per_night,
    check_in_date,
    check_out_date,
    room_type_value
FROM booking b
JOIN room r ON b.room_id = r.room_id
WHERE b.booking_id = NEW.booking_id;

stay_days := check_out_date - check_in_date;

calculated_total_amount := room_price_per_night * stay_days;

SELECT COALESCE(SUM(s.price), 0)
INTO calculated_service_charges
FROM service s
WHERE s.booking_id = NEW.booking_id;

SELECT COALESCE(MAX(p.discount_percentage), 0)
INTO applicable_discount_rate
FROM promotion p
WHERE NEW.issue_date BETWEEN p.start_date AND p.end_date
    AND (p.applicable_room_types IS NULL
        OR p.applicable_room_types = "
        OR POSITION(room_type_value IN p.applicable_room_types) > 0)
    AND stay_days >= p.minimum_stay_days;

calculated_discount_amount :=
    (calculated_total_amount + calculated_service_charges) * applicable_discount_rate / 100;

```



```

    calculated_tax_amount := calculated_total_amount * 0.10;

    calculated_final_amount :=
        calculated_total_amount + calculated_service_charges + calculated_tax_amount -
        calculated_discount_amount;

    NEW.total_amount := calculated_total_amount;
    NEW.tax_amount := calculated_tax_amount;
    NEW.service_charges := calculated_service_charges;
    NEW.discount_amount := calculated_discount_amount;
    NEW.final_amount := calculated_final_amount;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

TRIGGER:

CREATE OR REPLACE TRIGGER trigger_calculate_invoice_amounts
    BEFORE INSERT OR UPDATE ON Invoice
    FOR EACH ROW
    EXECUTE FUNCTION calculate_invoice_amounts();

```

6 Truy vấn cho hệ thống

Với các bảng được tạo ở trên, kết hợp với các chức năng cần thiết của hệ thống, nhóm đưa ra được những truy vấn SQL như sau:

Nhóm chia các câu truy vấn theo 2 phân quyền chính là khách hàng và quản lí khách sạn

- **Khách hàng:**

Trước khi sử dụng các function này, cần thiết lập các khóa chính các bảng booking, invoice, service, ta có thể làm như sau:

```
ALTER TABLE booking
ALTER COLUMN booking_id DROP DEFAULT,
ALTER COLUMN booking_id ADD GENERATED ALWAYS AS IDENTITY (START WITH ' ||
(max_booking_id + 1) || ' INCREMENT BY 1)
```

sau đó làm tương tự với các bảng còn lại.

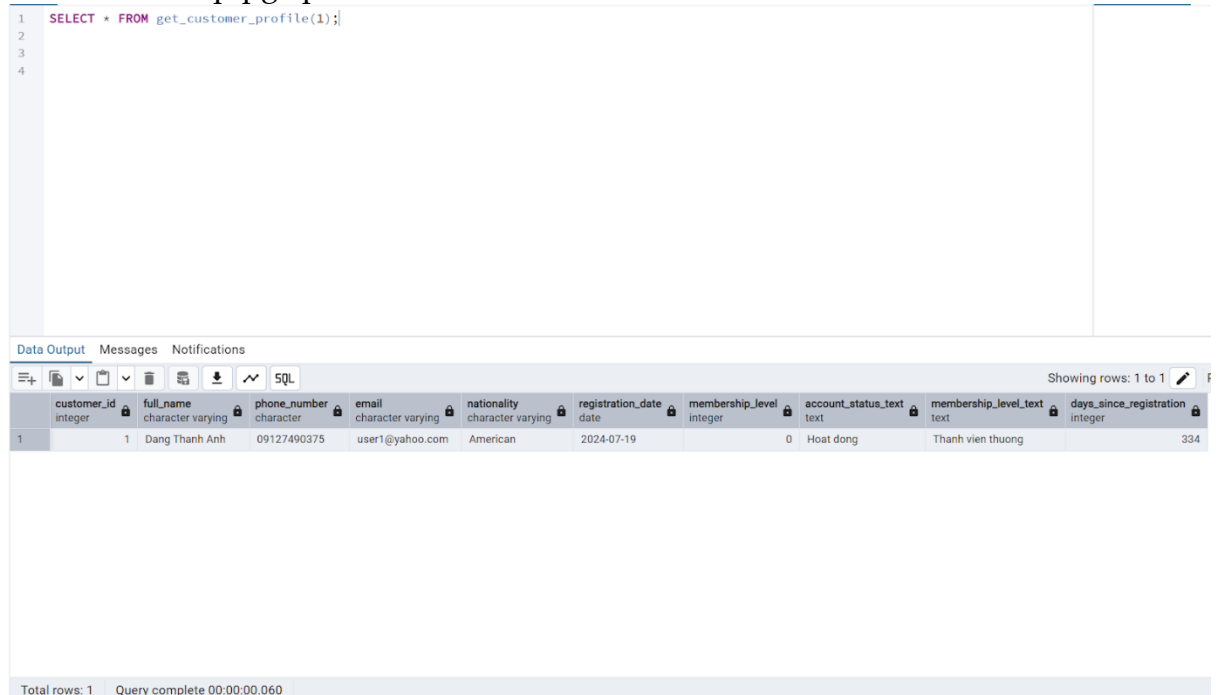
- **Tạo view để khách hàng có thể xem thông tin profile của bản thân, với view này quản lí cũng có thể xem được thông tin của các khách hàng (ngoại trừ những thông tin nhạy cảm như mật khẩu)**

```
CREATE OR REPLACE FUNCTION get_customer_profile(p_customer_id INTEGER)
RETURNS TABLE(
    customer_id INTEGER,
    full_name VARCHAR(100),
    phone_number CHAR(11),
    email VARCHAR(100),
    nationality VARCHAR(100),
    registration_date DATE,
    membership_level INTEGER,
    account_status_text TEXT,
    membership_level_text TEXT,
    days_since_registration INTEGER
) AS $$
BEGIN
    RETURN QUERY
    SELECT cp.customer_id, cp.full_name,
           cp.phone_number, cp.email, cp.nationality,
           cp.registration_date, cp.membership_level, cp.account_status_text,
           cp.membership_level_text, cp.days_since_registration
    FROM customer_profile cp
```

```

WHERE cp.customer_id = p_customer_id;
END;
$$ LANGUAGE plpgsql SECURITY DEFINER;

```



The screenshot shows a database query interface. The SQL editor contains the query: `SELECT * FROM get_customer_profile(1);`. Below the editor, the 'Data Output' tab is active, displaying a table with 10 columns and 1 row of data. The columns are: `customer_id` (integer), `full_name` (character varying), `phone_number` (character), `email` (character varying), `nationality` (character varying), `registration_date` (date), `membership_level` (integer), `account_status_text` (text), `membership_level_text` (text), and `days_since_registration` (integer). The single row of data contains the following values: 1, Dang Thanh Anh, 09127490375, user1@yahoo.com, American, 2024-07-19, 0, Hoat dong, Thanh vien thuong, 334. The interface also shows 'Total rows: 1' and 'Query complete 00:00:00.060'.

customer_id	full_name	phone_number	email	nationality	registration_date	membership_level	account_status_text	membership_level_text	days_since_registration
1	Dang Thanh Anh	09127490375	user1@yahoo.com	American	2024-07-19	0	Hoat dong	Thanh vien thuong	334

- Đăng ký tài khoản khách hàng, kiểm tra trùng username/email/CCCD

```

ALTER TABLE customer
ALTER COLUMN customer_id ADD GENERATED ALWAYS AS IDENTITY;

```

```

SELECT setval(
    pg_get_serial_sequence('customer', 'customer_id'),
    (SELECT MAX(customer_id) FROM customer)
);

```

```

CREATE OR REPLACE FUNCTION register_customer(
    p_full_name VARCHAR,
    p_identification_number CHAR(12),
    p_phone_number CHAR(11),
    p_email VARCHAR,
    p_nationality VARCHAR,
    p_username VARCHAR,
    p_password VARCHAR
) RETURNS TEXT AS $$
DECLARE
    duplicate_count INTEGER;
BEGIN
    SELECT COUNT(*) INTO duplicate_count
    FROM customer
    WHERE username = p_username OR email = p_email OR identification_number =

```

```

p_identification_number;
IF duplicate_count > 0 THEN
    RETURN 'Thông tin đăng ký đã tồn tại!';
END IF;

INSERT INTO customer (
    full_name, identification_number, phone_number, email, nationality, username,
    password, registration_date, account_status
) VALUES (
    p_full_name, p_identification_number, p_phone_number, p_email, p_nationality,
    p_username, p_password, CURRENT_DATE, 1
);
RETURN 'Đăng ký thành công!';
END;
$$ LANGUAGE plpgsql;

SELECT register_customer(
    'Le Thanh An',
    '012345678901',
    '09876543210',
    'leva@example.com',
    'Vietnam',
    'levana01',
    'matkhau123'
);

```

```

hotel_management=# SELECT register_customer(
hotel_management(#      'Le Thanh An',
hotel_management(#      '012345678901',
hotel_management(#      '09876543210',
hotel_management(#      'leva@example.com',
hotel_management(#      'Vietnam',
hotel_management(#      'levana01',
hotel_management(#      'matkhau123'
hotel_management(# );
      register_customer
-----
Dang ky thanh cong!
(1 row)

```

```

SELECT register_customer(
    'Nguyễn Văn D',
    '222222222222',
    '09222222222',

```

```

        'vand@example.com',
        'Vietnam',
        'levana01',      -- Trùng username
        'matkhau789'
    );
hotel_management=#
hotel_management=# SELECT register_customer(
hotel_management(#      'Nguyễn Văn D',
hotel_management(#      '222222222222',
hotel_management(#      '092222222222',
hotel_management(#      'vand@example.com',
hotel_management(#      'Vietnam',
hotel_management(#      'levana01',      -- Trùng username
hotel_management(#      'matkhau789'
hotel_management(# );
        register_customer
-----
Thong tin dang ky da ton tai!
(1 row)

```

- Khách hàng cập nhật thông tin cá nhân

```

CREATE OR REPLACE FUNCTION update_customer_profile(
    p_customer_id INTEGER,
    p_full_name VARCHAR(100) DEFAULT NULL, -- default null vì có thể khách hàng không
    update field này
    p_phone_number CHAR(11) DEFAULT NULL,
    p_email VARCHAR(100) DEFAULT NULL,
    p_nationality VARCHAR(100) DEFAULT NULL
)
RETURNS BOOLEAN AS $$
DECLARE
    record_count INTEGER;
BEGIN
    SELECT COUNT(*) INTO record_count -- đếm số cột bị thay đổi
    FROM customer
    WHERE customer_id = p_customer_id AND account_status >= 0;

    IF record_count = 0 THEN
        RETURN FALSE;
    END IF;

    UPDATE customer
    SET
        full_name = COALESCE(p_full_name, full_name),
        phone_number = COALESCE(p_phone_number, phone_number),
        email = COALESCE(p_email, email),
        nationality = COALESCE(p_nationality, nationality)
    WHERE customer_id = p_customer_id;

```

```

RETURN TRUE;
END;
$$ LANGUAGE plpgsql SECURITY DEFINER;

```

– SECURITY DEFINER để đảm bảo người dùng không có truy cập thẳng vào bảng customer mà vẫn có thể thay đổi được thông tin cá nhân của họ

```

SELECT update_customer_profile(1, 'Le Van Thanh An 2', '0123456789',
'levanthanhan@example.com', 'Vietnam');

```

```

hotel_management=> SELECT update_customer_profile(1, 'Le Van Thanh An 2', '0123456789', 'levanthanhan@example.com', 'Vietnam');
update_customer_profile
-----
t
(1 row)

```

```

hotel_management=> select * from get_customer_profile(1);
customer_id | full_name | phone_number | email | nationality | registration_date | membership_level | account_status_text | membership_level_text | day
s_since_registration
-----
1 | Le Van Thanh An 2 | 0123456789 | levanthanhan@example.com | Vietnam | 2024-07-19 | 0 | Hoat dong | Thanh vien thuong |
(1 row)

```

- Khách hàng xem hoá đơn

```

CREATE OR REPLACE FUNCTION get_booking_details(p_booking_id INTEGER)
RETURNS TABLE(

```

```

    booking_id INTEGER,
    customer_name VARCHAR(100),
    room_number INTEGER,
    room_type VARCHAR(20),
    check_in_date DATE,
    check_out_date DATE,
    nights INTEGER,
    room_price NUMERIC(10,2),
    promotion_name VARCHAR(100),
    total_room_amount NUMERIC(10,2),
    service_charges NUMERIC(10,2),
    tax_amount NUMERIC(10,2),
    discount_amount NUMERIC(10,2),
    final_amount NUMERIC(10,2),
    payment_status INTEGER

```

```

) AS $$

```

```

BEGIN

```

```

    RETURN QUERY

```

```

    SELECT

```

```

        b.booking_id,
        c.full_name,
        r.room_number,
        r.room_type,
        b.check_in_date,

```

```

        b.check_out_date,
        (b.check_out_date - b.check_in_date)::INTEGER AS nights,
        r.price_per_night,
        p.promotion_name,
        COALESCE(i.total_amount, 0)::NUMERIC(10,2),
        COALESCE(i.service_charges, 0)::NUMERIC(10,2),
        COALESCE(i.tax_amount, 0)::NUMERIC(10,2),
        COALESCE(i.discount_amount, 0)::NUMERIC(10,2),
        COALESCE(i.final_amount, 0)::NUMERIC(10,2),
        COALESCE(i.payment_status, 0)::INTEGER
    - COALESCE các trường này để tránh thiếu dữ liệu khi LEFT JOIN, do có 1 số trường không
      bắt buộc phải có giá trị
    FROM booking b
    JOIN customer c ON b.customer_id = c.customer_id
    JOIN room r ON b.room_id = r.room_id
    LEFT JOIN promotion p ON b.promotion_id = p.promotion_id - có thể không có promotion
      áp dụng
    LEFT JOIN invoice i ON b.booking_id = i.booking_id - có thể chưa tạo hóa đơn
    WHERE b.booking_id = p_booking_id;
END;
$$ LANGUAGE plpgsql;

SELECT * FROM get_booking_details(1505);

```

hotel_management> SELECT * FROM get_booking_details(1505);

booking_id	customer_name	room_number	room_type	check_in_date	check_out_date	nights	room_price	promotion_name	total_room_amount	service_charges	tax_amount
1595	Hoang Gia Cuong	102	Deluxe	2025-06-20	2025-06-22	2	470.00		940.00	72.86	94.00
(1 row)											

- **Khách hàng huỷ phòng:**

```

CREATE OR REPLACE FUNCTION cancel_booking_by_customer(
    p_booking_id INTEGER,
    p_customer_id INTEGER,
    p_cancellation_reason VARCHAR(500)
)
RETURNS TABLE (
    booking_id INTEGER,
    message TEXT
) AS $$
DECLARE
    v_room_id INTEGER;
BEGIN
    -- Kiểm tra booking tồn tại, đúng khách và đang hoạt động
    SELECT b.room_id
    INTO v_room_id

```

```

FROM booking b
WHERE b.booking_id = p_booking_id
      AND b.customer_id = p_customer_id
      AND b.status IN (1);

IF NOT FOUND THEN
    RETURN QUERY SELECT NULL, 'Booking không tồn tại hoặc không thuộc về khách
hàng này';
    RETURN;
END IF;

-- Cập nhật trạng thái booking
UPDATE booking b
SET status = 0,
    cancellation_reason = p_cancellation_reason,
    cancellation_date = CURRENT_DATE
WHERE b.booking_id = p_booking_id;

-- Cập nhật trạng thái phòng về trống
UPDATE room r
SET status = 0
WHERE r.room_id = v_room_id;

RETURN QUERY SELECT p_booking_id, 'Booking đã được hủy';
END;
$$ LANGUAGE plpgsql;

```

```
SELECT * FROM cancel_booking_by_customer(1505, 2, 'Khách thay đổi kế hoạch');
```

```

hotel_management=> SELECT * FROM cancel_booking_by_customer(1505, 2, 'Khách thay đổi kế
hoạch');
booking_id |      message
-----+-----
      1505 | Booking đã được hủy
(1 row)

```

Tình trạng booking cũng được chuyển đổi

```

hotel_management=> SELECT * FROM cancel_booking_by_customer(1505, 2, 'Khách thay đổi kế
hoạch');
booking_id |      message
-----+-----
      1505 | Booking đã được hủy
(1 row)

```

Tình trạng phòng trở về available


```

hotel_management=# select room_id, status from room where room_id = 2;
 room_id | status 
-----+-----
       2 |      0 
(1 row)

```

- **Quản lí ADMIN:**
- **Gợi ý khuyến mãi cho khách hàng:**

```

CREATE OR REPLACE FUNCTION suggest_promotions(
    p_room_type VARCHAR,
    p_stay_days INT,
    p_current_date DATE DEFAULT CURRENT_DATE
)
RETURNS TABLE (
    promotion_id INT,
    promotion_name VARCHAR,
    discount_percentage NUMERIC,
    start_date DATE,
    end_date DATE
) AS $$
BEGIN
    RETURN QUERY
    SELECT
        p.promotion_id,
        p.promotion_name,
        p.discount_percentage,
        p.start_date,
        p.end_date
    FROM promotion p
    WHERE p_current_date BETWEEN p.start_date AND p.end_date
        AND (p.applicable_room_types IS NULL OR p.applicable_room_types = '' OR
        p.applicable_room_types = p_room_type)
        AND p.minimum_stay_days <= p_stay_days;
END;
$$ LANGUAGE plpgsql;
SELECT * FROM suggest_promotions('Double', 3, '2025-06-20');

```

Data Output Messages Notifications					
	promotion_id integer	promotion_name character varying	discount_percentage numeric	start_date date	end_date date
1	1	Summer Splash	7	2025-06-18	2025-06-23

(Nếu khách lựa chọn phòng Double, đi trong 3 ngày từ 20-6)

- **Gợi ý phòng trống theo yêu cầu của khách hàng:**

```

CREATE OR REPLACE FUNCTION suggest_available_rooms(
    p_room_type VARCHAR,
    p_min_capacity INT,
    p_check_in_date DATE,
    p_check_out_date DATE
)
RETURNS TABLE (
    room_id INT,
    room_number INT,
    capacity INT,
    price_per_night NUMERIC(10,2),
    floor_number INT
) AS $$
BEGIN
    RETURN QUERY
    SELECT
        r.room_id,
        r.room_number,
        r.capacity,
        r.price_per_night,
        r.floor_number
    FROM room r
    WHERE r.room_type = p_room_type
        AND r.capacity >= p_min_capacity
        AND r.status != 2
        AND NOT EXISTS (
            SELECT 1 FROM booking b
            WHERE b.room_id = r.room_id
                AND b.status = 1
                AND (
                    b.check_in_date < p_check_out_date
                    AND b.check_out_date > p_check_in_date
                )
        )
    ORDER BY r.price_per_night;
END;

```

```
$$ LANGUAGE plpgsql;
SELECT * FROM suggest_available_rooms('Deluxe', 3, '2025-06-10', '2025-06-13');
```

Data Output Messages Notifications

	room_id integer	room_number integer	capacity integer	price_per_night numeric	floor_number integer
1	1	101	3	450.00	2
2	8	108	3	450.00	9
3	15	115	3	450.00	16
4	22	122	3	450.00	3
5	29	129	3	450.00	10
6	36	136	3	450.00	17
7	43	143	3	450.00	4
8	50	150	3	450.00	11
9	57	157	3	450.00	18
10	64	164	3	450.00	5
11	71	171	3	450.00	12
12	78	178	3	450.00	19
13	85	185	3	450.00	6
14	92	192	3	450.00	13
15	99	199	3	450.00	20
16	106	206	3	450.00	7
17	113	213	3	450.00	14
18	127	227	3	450.00	8
19	134	234	3	450.00	15
20	141	241	3	450.00	2

(Sẽ dựa vào khoảng thời gian khách đi cũng như loại phòng khách chọn)

- Tạo view đưa ra thông tin 10 ưu đãi mới nhất cho phòng đơn và phòng đôi

```
CREATE OR REPLACE VIEW v_top10_newest_single_double_promotions AS
SELECT *
FROM promotion
WHERE applicable_room_types IN ('Single', 'Double')
OR applicable_room_types IS NULL
ORDER BY start_date DESC
LIMIT 10;
Select * from v_top10_newest_single_double_promotions;
```

CREATE VIEW

promotion_id	promotion_name	description	discount_percentage	start_date	end_date	applicable_room_types	minimum_stay_days
20	Room Rush	Get 45% off for a limited time!	45	2025-11-05	2025-11-10		
17	Midweek Madness	Get 39% off for a limited time!	39	2025-10-15	2025-10-20	Double	
16	Limited Luxury	Get 37% off for a limited time!	37	2025-10-08	2025-10-13		
13	Sunny Days	Get 31% off for a limited time!	31	2025-09-17	2025-09-22	Double	
12	Cozy Nights	Get 29% off for a limited time!	29	2025-09-10	2025-09-15		
9	Flash Deal	Get 23% off for a limited time!	23	2025-08-20	2025-08-25	Double	
8	Holiday Bonus	Get 21% off for a limited time!	21	2025-08-13	2025-08-18		
5	Rainy Season Sale	Get 15% off for a limited time!	15	2025-07-23	2025-07-28	Double	
4	VIP Deal	Get 13% off for a limited time!	13	2025-07-16	2025-07-21		
1	Summer Splash	Get 7% off for a limited time!	7	2025-06-25	2025-06-30	Double	

(10 rows)

- Đưa ra các supplier cho item và số item họ cung cấp theo thứ tự giảm dần:

```
SELECT supplier_info, COUNT(*) AS total_items
FROM inventory
GROUP BY supplier_info
ORDER BY total_items DESC, supplier_info;
```

```
hotel_management=# SELECT supplier_info, COUNT(*) AS total_items
hotel_management=# FROM inventory
hotel_management=# GROUP BY supplier_info
hotel_management=# ORDER BY total_items DESC, supplier_info;
 supplier_info | total_items
-----+-----
BlueSky Supplies |          300
Kim Long Ltd.   |          300
Phuc An JSC     |          300
SunHouse Co.    |          300
Thanh Binh Co.  |          300
(5 rows)
```

- Đưa ra danh sách vật tư dưới 1 mốc số lượng nhất định

```
SELECT item_id, item_name, quantity_in_stock
FROM inventory
WHERE quantity_in_stock < 10
ORDER BY quantity_in_stock ASC, item_name;
```

```
hotel_management=# SELECT item_id, item_name, quantity_in_stock
hotel_management=# FROM inventory
hotel_management=# WHERE quantity_in_stock < 10
hotel_management=# ORDER BY quantity_in_stock ASC, item_name;
 item_id | item_name | quantity_in_stock
-----+-----+-----
766 | Blanket 766 | 5
1379 | Curtains 1379 | 5
40 | Hair Dryer 40 | 5
625 | Hair Dryer 625 | 5
85 | Hair Dryer 85 | 5
264 | Hanger 264 | 5
1121 | Kettle 1121 | 5
161 | Kettle 161 | 5
1258 | Mirror 1258 | 5
313 | Mirror 313 | 5
943 | Mirror 943 | 5
1035 | Pillow 1035 | 5
435 | Pillow 435 | 5
1444 | Shampoo 1444 | 5
1433 | Slippers 1433 | 5
17 | Towel 17 | 5
197 | Towel 197 | 5
1071 | TV 1071 | 5
1221 | TV 1221 | 5
246 | TV 246 | 5
1066 | Blanket 1066 | 6
192 | Desk Lamp 192 | 6
657 | Desk Lamp 657 | 6
```

- Đưa ra danh sách vật tư mà khách sạn hiện đang có và số lượng theo thứ tự giảm dần

```
SELECT item_id, item_name, quantity_in_stock
FROM inventory
ORDER BY quantity_in_stock DESC, item_name;
```

```
hotel_management=# SELECT item_id, item_name, quantity_in_stock
hotel_management=# FROM inventory
hotel_management=# ORDER BY quantity_in_stock DESC, item_name;
```

item_id	item_name	quantity_in_stock
1422	Desk Lamp 1422	100
402	Desk Lamp 402	100
459	Hanger 459	100
1076	Kettle 1076	100
367	Minibar 367	100
712	Minibar 712	100
538	Mirror 538	100
808	Mirror 808	100
1309	Shampoo 1309	100
1310	Soap Bar 1310	100
243	Toothbrush Set 243	100
723	Toothbrush Set 723	100
797	Towel 797	100
741	TV 741	100
166	Blanket 166	99
691	Blanket 691	99
777	Desk Lamp 777	99
1284	Hanger 1284	99
1374	Hanger 1374	99
487	Minibar 487	99
388	Mirror 388	99
780	Pillow 780	99
1052	Towel 1052	99
377	Towel 377	99
156	TV 156	99
1304	Curtains 1304	98
102	Desk Lamp 102	98
1345	Hair Dryer 1345	98
190	Hair Dryer 190	98
580	Hair Dryer 580	98
1209	Hanger 1209	98
465	Pillow 465	98
630	Pillow 630	98
645	Pillow 645	98
620	Soap Bar 620	98
6	TV 6	98
224	Curtains 224	97
42	Desk Lamp 42	97
1391	Kettle 1391	97
180	Pillow 180	97
600	Pillow 600	97
720	Pillow 720	97
575	Soap Bar 575	97
1323	Toothbrush Set 1323	97
1476	TV 1476	97
1306	Blanket 1306	96
631	Blanket 631	96
974	Curtains 974	96
207	Desk Lamp 207	96
760	Hair Dryer 760	96

- Quản lý kiểm tra các vật tư có số lượng ít:

```
CREATE OR REPLACE FUNCTION check_low_inventory()
RETURNS TABLE(
    item_id INTEGER,
    item_name VARCHAR(500),
    current_stock INTEGER,
    unit_price NUMERIC(10,2),
    supplier_info VARCHAR,
    location VARCHAR(100),
    warning_message TEXT
```

```

) AS $$
BEGIN
    RETURN QUERY
    SELECT
        i.item_id,
        i.item_name,
        i.quantity_in_stock,
        i.unit_price,
        i.supplier_info,
        i.location_in_storage,
        CONCAT('CANH BAO: ', i.item_name, ' chi con ', i.quantity_in_stock)
        AS warning_message
    FROM Inventory i
    WHERE i.quantity_in_stock < 20;
END;
$$ LANGUAGE plpgsql;

```

Query Query History Scratch Pad X

```

1 SELECT * FROM check_low_inventory();
2

```

Data Output Messages Notifications

Showing rows: 1 to 248 Page No: 1 of 1

	Item_id integer	Item_name character varying	current_stock integer	unit_price numeric	supplier_info character varying	location character varying	warning_message text
1	17	Towel 17	19	107.35	BlueSky Supplies	Shelf 18 - Zone C	CANH BAO: Towel 17 chi con 19
2	22	Minibar 22	8	64.52	BlueSky Supplies	Shelf 3 - Zone C	CANH BAO: Minibar 22 chi con 8
3	25	Hair Dryer 25	18	39.94	Thanh Binh Co.	Shelf 6 - Zone A	CANH BAO: Hair Dryer 25 chi con 18
4	27	Desk Lamp 27	16	72.89	BlueSky Supplies	Shelf 8 - Zone C	CANH BAO: Desk Lamp 27 chi con 16
5	29	Curtains 29	9	29.84	Phuc An JSC	Shelf 10 - Zone E	CANH BAO: Curtains 29 chi con 9
6	31	Blanket 31	10	104.70	Kim Long Ltd.	Shelf 12 - Zone B	CANH BAO: Blanket 31 chi con 10
7	33	Toothbrush Set 33	14	177.27	SunHouse Co.	Shelf 14 - Zone D	CANH BAO: Toothbrush Set 33 chi con 14
8	34	Shampoo 34	19	156.51	Phuc An JSC	Shelf 15 - Zone E	CANH BAO: Shampoo 34 chi con 19
9	37	Minibar 37	18	85.82	BlueSky Supplies	Shelf 18 - Zone C	CANH BAO: Minibar 37 chi con 18
10	41	Kettle 41	16	146.21	Kim Long Ltd.	Shelf 2 - Zone B	CANH BAO: Kettle 41 chi con 16
11	46	Blanket 46	12	168.57	Kim Long Ltd.	Shelf 7 - Zone B	CANH BAO: Blanket 46 chi con 12
12	60	Pillow 60	6	19.45	Thanh Binh Co.	Shelf 1 - Zone A	CANH BAO: Pillow 60 chi con 6
13	76	Blanket 76	9	79.90	Kim Long Ltd.	Shelf 17 - Zone B	CANH BAO: Blanket 76 chi con 9

Total rows: 248 Query complete 00:00:00.082

Successfully run. Total query runtime: 82 msec. 248 rows affected. X

CRLF Ln 1, Col 37

- **Thống kê tổng tiền dịch vụ theo các loại trong mùa du lịch**

```

CREATE OR REPLACE FUNCTION services_revenue_by_type_summer_2025()
RETURNS TABLE (
    service_type CHAR(100),
    total_revenue NUMERIC(10,2),
    number_of_services BIGINT
) AS $$
BEGIN
    RETURN QUERY
    SELECT
        s.service_type,
        SUM(s.price)::NUMERIC(10,2) AS total_revenue,
        COUNT(*) AS number_of_services

```

```

FROM service s
JOIN booking b ON s.booking_id = b.booking_id
WHERE EXTRACT(YEAR FROM b.check_out_date) = 2025
      AND EXTRACT(MONTH FROM b.check_out_date) BETWEEN 6 AND 8
GROUP BY s.service_type
ORDER BY total_revenue DESC;
END;
$$ LANGUAGE plpgsql;
SELECT * FROM services_revenue_by_type_summer_2025();

```

Data Output Messages Notifications

	service_type character	total_revenue numeric	number_of_services bigint
1	Spa	1000.90	16
2	Laundry	924.85	15
3	Food	902.95	15

(tháng 6,7,8 năm 2025)

- Truy vấn tỉ lệ sử dụng dịch vụ của khách hàng

```

SELECT
  TO_CHAR(b.check_out_date, 'YYYY-MM') AS month,
  s.service_type,
  COUNT(*) AS usage_count,
  ROUND(100.0 * COUNT(*) / SUM(COUNT(*) OVER (PARTITION BY TO_CHAR(b.check_out_date,
    'YYYY-MM')), 2) AS percentage
FROM service s
JOIN booking b ON s.booking_id = b.booking_id
WHERE EXTRACT(YEAR FROM b.check_out_date) = 2025
      AND EXTRACT(MONTH FROM b.check_out_date) BETWEEN 5 AND 8
GROUP BY month, s.service_type
ORDER BY month, percentage DESC;

```

	month text	service_type character (100)	usage_count bigint	percentage numeric
1	2025-05	Food	6	37.50
2	2025-05	Laundry	5	31.25
3	2025-05	Spa	5	31.25
4	2025-06	Food	5	33.33
5	2025-06	Laundry	5	33.33
6	2025-06	Spa	5	33.33
7	2025-07	Spa	6	37.50
8	2025-07	Food	5	31.25
9	2025-07	Laundry	5	31.25
10	2025-08	Food	5	33.33
11	2025-08	Laundry	5	33.33
12	2025-08	Spa	5	33.33

- **Thống kê top 5 khách hàng có lượng chi tiêu nhiều nhất năm:**

```
CREATE OR REPLACE FUNCTION top_spending_customers(year_input INT)
RETURNS TABLE (
    customer_id INT,
    full_name VARCHAR(100),
    total_spent NUMERIC(10,2)
) AS $$
BEGIN
    RETURN QUERY
    SELECT
        c.customer_id,
        c.full_name,
        SUM(i.final_amount)::NUMERIC(10,2)
    FROM customer c
    JOIN booking b ON c.customer_id = b.customer_id
    JOIN invoice i ON b.booking_id = i.booking_id
    WHERE EXTRACT(YEAR FROM i.issue_date) = year_input
        AND i.payment_status = 1
    GROUP BY c.customer_id, c.full_name
    ORDER BY SUM(i.final_amount) DESC
    LIMIT 5;
END;
$$ LANGUAGE plpgsql;
SELECT * FROM top_spending_customers(2025);
```

Data Output Messages Notifications

	customer_id integer	full_name character varying	total_spent numeric
1	40	Huynh Van Giang	3933.43
2	5	Hoang Van Binh	3930.77
3	75	Nguyen Duc Phuong	3928.36
4	110	Vo Thu Khanh	3899.09
5	19	Pham Thu Phuong	3124.63

- **Tìm các khách hàng chưa thanh toán hoá đơn:**

```
SELECT
    i.invoice_id,
    c.full_name AS customer_name,
    i.issue_date,
    i.final_amount,
    i.payment_method,
    CURRENT_DATE - i.issue_date AS days_overdue
FROM (
```



```

SELECT invoice_id, booking_id, issue_date, final_amount, payment_method
FROM invoice
WHERE payment_status = 0 AND CURRENT_DATE > issue_date
) AS i
JOIN booking b ON i.booking_id = b.booking_id
JOIN customer c ON b.customer_id = c.customer_id
ORDER BY i.issue_date ASC;

```

Query Query History

```

1 SELECT
2     i.invoice_id,
3     c.full_name,
4     i.issue_date,
5     i.final_amount,
6     i.payment_method,
7     CURRENT_DATE - i.issue_date AS days_overdue
8 FROM invoice i
9 JOIN booking b ON i.booking_id = b.booking_id
10 JOIN customer_profile c ON b.customer_id = c.customer_id
11 WHERE i.payment_status = 0 AND CURRENT_DATE > i.issue_date
12 ORDER BY i.issue_date ASC;
13

```

Data Output Messages Notifications

	invoice_id integer	full_name character varying (100)	issue_date date	final_amount numeric (10,2)	payment_method character (20)	days_overdue integer
1	5	Pham Thu Phuong	2025-01-13	1145.04	Transfer	163
2	15	Le Huu Binh	2025-02-04	1348.88	Cash	141
3	25	Huynh Quoc Hoa	2025-02-22	1576.03	Card	123
4	35	Bui Thanh Nam	2025-03-16	2034.26	Transfer	101
5	45	Tran Thu Binh	2025-04-03	620.08	Cash	83
6	55	Tran Thi Giang	2025-04-25	2137.34	Card	61
7	65	Bui Minh Nam	2025-05-13	868.25	Transfer	43
8	75	Tran Thanh Binh	2025-06-04	2201.51	Cash	21
9	85	Bui Minh Khanh	2025-06-22	694.39	Card	3

- **Doanh thu theo tháng và theo phòng**

```

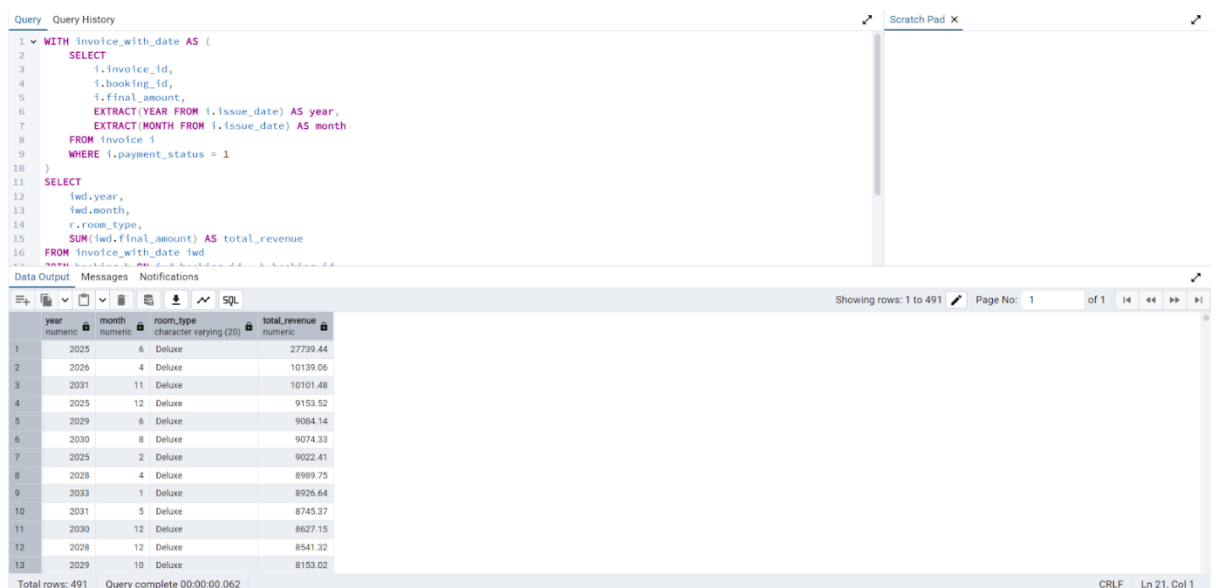
WITH invoice_with_date AS (
SELECT
    i.invoice_id,
    i.booking_id,
    i.final_amount,
    EXTRACT(YEAR FROM i.issue_date) AS year,
    EXTRACT(MONTH FROM i.issue_date) AS month
FROM invoice i
WHERE i.payment_status = 1
)
SELECT
    iwd.year,
    iwd.month,
    r.room_type,
    SUM(iwd.final_amount) AS total_revenue
FROM invoice_with_date iwd

```

```

JOIN booking b ON iwd.booking_id = b.booking_id
JOIN room r ON b.room_id = r.room_id
GROUP BY iwd.year, iwd.month, r.room_type
ORDER BY total_revenue DESC, iwd.year ASC, iwd.month ASC;

```



The screenshot shows a SQL query editor with a query window on the left and a data output window on the right. The query is a complex SQL statement using a CTE to filter invoices and then aggregate them by year, month, and room type to calculate total revenue. The data output window displays the results of this query as a table with 4 columns: year, month, room_type, and total_revenue. The table contains 13 rows of data, sorted by total_revenue in descending order. The status bar at the bottom indicates that the query is complete and has returned 491 rows.

year	month	room_type	total_revenue
2025	6	Deluxe	27739.44
2026	4	Deluxe	10139.06
2031	11	Deluxe	10101.48
2025	12	Deluxe	9153.52
2029	6	Deluxe	9084.14
2030	8	Deluxe	9074.33
2025	2	Deluxe	9022.41
2028	4	Deluxe	8989.75
2032	1	Deluxe	8926.64
2031	5	Deluxe	8745.37
2030	12	Deluxe	8627.15
2028	12	Deluxe	8541.32
2029	10	Deluxe	8153.02

- **Tổng hợp feedback sau khi đặt phòng theo đơn của khách hàng:**

```

CREATE OR REPLACE VIEW v_booking_review_summary AS
SELECT
    b.booking_id,
    c.customer_id,
    c.full_name AS customer_name,
    r.room_number,
    b.check_in_date,
    b.check_out_date,
    rv.review_id,
    rv.rating,
    rv.comment,
    rv.review_date,
    rv.response,
    rv.response_date
FROM booking b
JOIN customer c ON b.customer_id = c.customer_id
JOIN room r ON b.room_id = r.room_id
LEFT JOIN review rv ON b.booking_id = rv.booking_id
ORDER BY c.customer_id, b.booking_id;

```

```

SELECT * FROM v_booking_review_summary;

```

hotel_management=# SELECT * FROM v_booking_review_summary;										
booking_id	customer_id	customer_name	room_number	check_in_date	check_out_date	review_id	rating	comment	review_date	r
1500	1	Bui Minh Dung	101	2033-03-20	2033-03-21					
1	2	Vo Huu Hoa	114	2025-01-03	2025-01-05					
2	3	Dang Ngoc Anh	127	2025-01-05	2025-01-08	2	5	Comment sample 3	2025-01-10	
3	4	Bui Thanh Hoa	140	2025-01-07	2025-01-11					
4	5	Huynh Thanh Cuong	153	2025-01-09	2025-01-10					
5	6	Dang Thanh Hoa	166	2025-01-11	2025-01-13					
6	7	Tran Quoc Hoa	179	2025-01-13	2025-01-16					
7	8	Vo Gia Phuong	192	2025-01-15	2025-01-19	7	5	Comment sample 8	2025-01-19	
8	9	Bui Thu Linh	205	2025-01-17	2025-01-18					
9	10	Huynh Ngoc Phuong	218	2025-01-19	2025-01-21	9	5	Comment sample 10	2025-01-23	
10	11	Tran Ngoc Hoa	231	2025-01-21	2025-01-24					

6	7	Tran Quoc Hoa	179	2025-01-13	2025-01-16					
7	8	Vo Gia Phuong	192	2025-01-15	2025-01-19	7	5	Comment sample 8	2025-01-19	
8	9	Bui Thu Linh	205	2025-01-17	2025-01-18					
9	10	Huynh Ngoc Phuong	218	2025-01-19	2025-01-21	9	5	Comment sample 10	2025-01-23	
10	11	Tran Ngoc Hoa	231	2025-01-21	2025-01-24					
11	12	Do Gia Phuong	244	2025-01-23	2025-01-27					
12	13	Le Gia Dung	257	2025-01-25	2025-01-26	11	5	Comment sample 13	2025-01-31	--
13	14	Hoang Minh Linh	270	2025-01-27	2025-01-29					
14	15	Huynh Quoc Cuong	283	2025-01-29	2025-02-01					
15	16	Vo Ngoc Dung	296	2025-01-31	2025-02-04	14	5	Comment sample 16	2025-02-05	
16	17	Nguyen Huu Phuong	309	2025-02-02	2025-02-03					
17	18	Bui Thanh Giang	322	2025-02-04	2025-02-06					
18	19	Pham Thanh Khanh	335	2025-02-06	2025-02-09					
19	20	Dang Quoc Binh	348	2025-02-08	2025-02-12					
20	21	Huynh Thanh Khanh	361	2025-02-10	2025-02-11					

7 Hiệu năng

Hệ thống quản lý khách sạn được xây dựng trên nền tảng hệ quản trị cơ sở dữ liệu PostgreSQL, nổi bật với khả năng xử lý truy vấn mạnh mẽ và tính mở rộng cao. Cơ sở dữ liệu được chuẩn hóa đến dạng ba (3NF) nhằm loại bỏ dư thừa dữ liệu và đảm bảo tính toàn vẹn.

PostgreSQL cung cấp nhiều công cụ hỗ trợ tối ưu hóa hiệu năng như EXPLAIN và ANALYZE, cho phép phân tích kế hoạch thực thi truy vấn và điều chỉnh các truy vấn chậm. Trong quá trình phát triển hệ thống, nhóm đã sử dụng các công cụ này để tối ưu các truy vấn phức tạp như: đưa ra danh sách vật tư theo từng yêu cầu (với việc bảng inventory có số lượng bản ghi lớn), hay doanh thu theo tháng và theo loại phòng, tìm các khách hàng chưa thanh toán hoá đơn, ...)

- **Khi thao tác tìm khách hàng chưa thanh toán hoá đơn**

EXPLAIN ANALYZE

SELECT

```
    i.invoice_id,
    c.full_name AS customer_name,
    i.issue_date,
    i.final_amount,
    i.payment_method,
    CURRENT_DATE - i.issue_date AS days_overdue
```

FROM (

```
    SELECT invoice_id, booking_id, issue_date, final_amount, payment_method
    FROM invoice
```

```
    WHERE payment_status = 0 AND CURRENT_DATE > issue_date
```

) AS i

JOIN booking b ON i.booking_id = b.booking_id

JOIN customer c ON b.customer_id = c.customer_id

ORDER BY i.issue_date ASC;

Planning Time: 20.774 ms

Execution Time: 2.514 ms

Ta tối ưu như sau:

EXPLAIN ANALYZE

SELECT

```
    i.invoice_id,
    c.full_name,
```

```

        i.issue_date,
        i.final_amount,
        i.payment_method,
        CURRENT_DATE - i.issue_date AS days_overdue
FROM invoice i
JOIN booking b ON i.booking_id = b.booking_id
JOIN customer_profile c ON b.customer_id = c.customer_id
WHERE i.payment_status = 0 AND CURRENT_DATE > i.issue_date
ORDER BY i.issue_date ASC;

```

Việc không sử dụng subquery sẽ nhanh hơn, do trong subquery của câu ban đầu chứa nhiều điều kiện và DBMS sẽ cần phải chạy subquery trước rồi mới thực hiện phép join. Nó làm tăng planning time của câu truy vấn

Cụ thể kết quả trước khi có index

Planning Time: 0.668 ms

Execution Time: 0.433 ms

```

CREATE INDEX IF NOT EXISTS idx_invoice_status_date ON invoice (payment_status, issue_date);
CREATE INDEX idx_invoice_booking_id ON invoice (booking_id);
CREATE INDEX idx_booking_customer_id ON booking (customer_id);

```

sau khi chạy truy vấn, ta thấy truy vấn sử dụng: Bitmap Index Scan on idx_invoice_status_date (nó sẽ truy cập theo từng group, không cần tìm từng dòng), Index Scan using idx_customer_id on customer và Index Cond: (customer_id = b.customer_id). Và câu lệnh cũng cho thấy Postgres đã sử dụng Nested Loop, nó sẽ nhanh với dữ liệu nhỏ và có index

Kết quả sau khi dùng index

Planning Time: 0.410 ms

Execution Time: 0.341 ms

- **Khi thao tác tính doanh thu:**

```

EXPLAIN ANALYZE
WITH invoice_with_date AS (
SELECT
        i.invoice_id,
        i.booking_id,
        i.final_amount,
        EXTRACT(YEAR FROM i.issue_date) AS year,
        EXTRACT(MONTH FROM i.issue_date) AS month
FROM invoice i
WHERE i.payment_status = 1

```

```

)
SELECT
    iwd.year,
    iwd.month,
    r.room_type,
    SUM(iwd.final_amount) AS total_revenue
FROM invoice_with_date iwd
JOIN booking b ON iwd.booking_id = b.booking_id
JOIN room r ON b.room_id = r.room_id
GROUP BY iwd.year, iwd.month, r.room_type
ORDER BY total_revenue DESC, iwd.year ASC, iwd.month ASC;

```

Ta có thể tách CTE để làm giảm độ phức tạp của truy vấn chính, và cũng có thể sử dụng lại CTE này cho các truy vấn khác (nếu cần). Ngoài ra, ta tránh gọi `EXTRACT()` nhiều lần, do DBMS không ghi nhớ kết quả `EXTRACT` trong lệnh `GROUP BY`. Vậy, cách tốt nhất là tạo CTE là một bảng tạm thời, tồn tại trong phạm vi của một câu truy vấn.

Planning Time: 0.358 ms

Execution Time: 2.895 ms

- Khi thao tác tìm kiếm các loại vật tư theo thứ tự giảm dần:

```

hotel_management=# EXPLAIN ANALYZE
hotel_management=# SELECT item_id, item_name, quantity_in_stock
hotel_management=# FROM inventory
hotel_management=# ORDER BY quantity_in_stock DESC, item_name;
               QUERY PLAN
-----
Sort  (cost=112.13..115.88 rows=1500 width=20) (actual time=1.686..1.736 rows=1500 loops=1)
  Sort Key: quantity_in_stock DESC, item_name
  Sort Method: quicksort  Memory: 107kB
  -> Seq Scan on inventory  (cost=0.00..33.00 rows=1500 width=20) (actual time=0.018..0.211 rows=1500 loops=1)
Planning Time: 0.085 ms
Execution Time: 1.845 ms
(6 rows)

```

`CREATE INDEX idx_inventory_quantity_itemname ON inventory(quantity_in_stock DESC, item_name);`

```

               QUERY PLAN
-----
Sort  (cost=112.13..115.88 rows=1500 width=20) (actual time=1.511..1.561 rows=1500 loops=1)
  Sort Key: quantity_in_stock DESC, item_name
  Sort Method: quicksort  Memory: 107kB
  -> Seq Scan on inventory  (cost=0.00..33.00 rows=1500 width=20) (actual time=0.012..0.124 rows=1500 loops=1)
Planning Time: 0.096 ms
Execution Time: 1.622 ms
(6 rows)

```

- Đưa ra số lượng vật tư dưới 1 mốc nhất định

```

               QUERY PLAN
-----
Sort  (cost=26.66..26.87 rows=84 width=20) (actual time=0.301..0.306 rows=84 loops=1)
  Sort Key: quantity_in_stock, item_name
  Sort Method: quicksort  Memory: 28kB
  -> Bitmap Heap Scan on inventory  (cost=4.93..23.98 rows=84 width=20) (actual time=0.094..0.128 rows=84 loops=1)
        Recheck Cond: (quantity_in_stock < 10)
        Heap Blocks: exact=17
  -> Bitmap Index Scan on idx_inventory_quantity_itemname  (cost=0.00..4.91 rows=84 width=0) (actual time=0.077..0.077 rows=84 loops=1)
        Index Cond: (quantity_in_stock < 10)
Planning Time: 0.126 ms
Execution Time: 0.344 ms
(10 rows)

```

```
CREATE INDEX idx_inventory_quantityname_asc ON inventory(quantity_in_stock ASC,  
item_name);
```

```
EXPLAIN ANALYZE  
SELECT item_id, item_name, quantity_in_stock  
FROM inventory  
WHERE quantity_in_stock < 10  
ORDER BY quantity_in_stock ASC, item_name;
```

```
QUERY PLAN  
-----  
Sort (cost=26.66..26.87 rows=84 width=20) (actual time=0.226..0.232 rows=84 loops=1)  
  Sort Key: quantity_in_stock, item_name  
  Sort Method: quicksort Memory: 28kB  
    -> Bitmap Heap Scan on inventory (cost=4.93..23.98 rows=84 width=20) (actual time=0.055..0.090 rows=84 loops=1)  
        Recheck Cond: (quantity_in_stock < 10)  
        Heap Blocks: exact=17  
        -> Bitmap Index Scan on idx_inventory_quantityname_asc (cost=0.00..4.91 rows=84 width=0) (actual time=0.040..0.041 rows=84 loops=1)  
            Index Cond: (quantity_in_stock < 10)  
Planning Time: 1.045 ms  
Execution Time: 0.256 ms  
(10 rows)
```

Kết luận

Qua quá trình thực hiện đề tài “*Hệ thống quản lý khách sạn*”, nhóm đã có cơ hội áp dụng kiến thức về cơ sở dữ liệu vào một bài toán thực tế, từ khâu phân tích yêu cầu, thiết kế mô hình dữ liệu đến hiện thực hoá bằng các bảng quan hệ và câu lệnh SQL.

Đề tài giúp nhóm nâng cao khả năng làm việc nhóm, tư duy logic, và kỹ năng sử dụng hệ quản trị cơ sở dữ liệu. Nhóm hy vọng báo cáo này phần nào thể hiện được sự cố gắng và kiến thức đã tích lũy được trong học phần.

Nhóm sinh viên thực hiện

