ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH

Hệ cơ sở dữ liệu (TN) (CO2014)

Báo cáo bài tập lớn

Hệ thống đặt vé xem phim

GVHD: Dương Huỳnh Anh Đức

Lớp: L05

Sinh viên: Lư Chấn Vũ - 2313955 *(Nhóm 7)*
Nguyễn Phú Vinh - 2313922 *(Nhóm 7)*
Huỳnh Xuân Quốc Việt - 2313891 *(Nhóm 7)*
Lê Minh Khoa - 2311593 *(Nhóm 7)*
Lê Minh Trí - 2313593 *(Nhóm 7, **Leader**)*

TP. HỒ CHÍ MINH, 09/2025

Mục lục

Danh sách hình ảnh

Danh sách bảng biểu

Danh sách thành viên và nhiệm vụ

STT	Họ và tên	MSSV	Nhiệm vụ	% hoàn thành
1	Lư Chấn Vũ	2313955	- Phần 1.3 - Phần 2, 3	100%
2	Nguyễn Phú Vinh	2313922	- Phần 1.2.2 - Phần 2, 3	100%
3	Huỳnh Xuân Quốc Việt	2313891	- Phần 1.2.1 - Phần 2, 3	100%
4	Lê Minh Khoa	2311593	- Phần 1.2.2 - Phần 2, 3	100%
5	Lê Minh Trí	2313593	- Phần 1.1 - Phần 2, 3	100%

Bảng 1: Danh sách thành viên và nhiệm vụ

Lời mở đầu

Trong kỷ nguyên số hiện nay, dữ liệu được xem như là tài sản quý giá của mọi tổ chức và doanh nghiệp. Việc thu thập, quản lý và khai thác dữ liệu một cách hiệu quả đóng vai trò quan trọng trong quá trình vận hành, phát triển dịch vụ cũng như ra quyết định chiến lược. Cơ sở dữ liệu (CSDL) chính là nền tảng cốt lõi giúp đảm bảo cho các hệ thống thông tin hoạt động ổn định, chính xác và an toàn. Một cơ sở dữ liệu được thiết kế khoa học không chỉ hỗ trợ lưu trữ và xử lý khối lượng dữ liệu lớn mà còn giúp tối ưu hiệu suất, duy trì tính toàn vẹn và tạo điều kiện mở rộng hệ thống trong tương lai.

Trong thực tế, nhiều lĩnh vực ứng dụng công nghệ thông tin đều cần đến các hệ thống cơ sở dữ liệu, điển hình như thương mại điện tử, ngân hàng, giáo dục, và đặc biệt là lĩnh vực giải trí. Một trong những dịch vụ giải trí phổ biến hiện nay là rạp chiếu phim, nơi nhu cầu đặt vé trực tuyến ngày càng gia tăng. Các ứng dụng đặt vé xem phim không chỉ giúp khách hàng dễ dàng lựa chọn phim, suất chiếu, vị trí ghế ngồi và thanh toán trực tuyến, mà còn hỗ trợ nhà quản lý rạp kiểm soát lịch chiếu, doanh thu, khuyến mãi cũng như tình trạng đặt vé theo thời gian thực. Tất cả những chức năng này đều được xây dựng và vận hành dựa trên một hệ thống cơ sở dữ liệu được thiết kế bài bản.

Trong khuôn khổ môn *Hệ cơ sở dữ liệu* tại Trường Đại học Bách Khoa – ĐHQG TP.HCM, nhóm chúng em thực hiện đề tài **"Xây dựng cơ sở dữ liệu cho hệ thống đặt vé xem phim"**. Mục tiêu của đề tài là phân tích yêu cầu nghiệp vụ của hệ thống, xây dựng sơ đồ EERD (Enhanced Entity-Relationship Diagram) để mô hình hóa các thực thể và mối quan hệ, sau đó tiến hành ánh xạ sang mô hình quan hệ để triển khai dưới dạng các bảng trong hệ quản trị cơ sở dữ liệu. Quá trình này không chỉ giúp nhóm củng cố và vận dụng các kiến thức lý thuyết đã học, mà còn rèn luyện kỹ năng phân tích, mô hình hóa và triển khai một cơ sở dữ liệu trong tình huống thực tế.

Thông qua báo cáo này, nhóm mong muốn trình bày một cách có hệ thống các bước thiết kế cơ sở dữ liệu cho một ứng dụng đặt vé xem phim, từ khâu phân tích đến triển khai. Đây sẽ là nền tảng quan trọng giúp sinh viên tiếp cận gần hơn với thực tiễn, đồng thời làm quen với các yêu cầu về chất lượng, tính chính xác và khả năng mở rộng của một hệ thống cơ sở dữ liệu trong bối cảnh công nghệ ngày nay.

1 Phân tích và mô tả yêu cầu dữ liệu

1.1 Tìm hiểu ứng dụng/hệ thống

1.1.1 Tên ứng dụng/hệ thống

Để khảo sát và tham khảo cho việc xây dựng cơ sở dữ liệu ứng dụng đặt vé xem phim, nhóm lựa chọn hệ thống CGV Cinemas Việt Nam làm đối tượng nghiên cứu chính. CGV hiện là một trong những chuỗi rạp chiếu phim lớn tại Việt Nam, cung cấp ứng dụng di động (CGV Cinemas trên App Store và Google Play) cũng như website chính thức tại địa chỉ: *CGV*.

Ứng dụng cho phép người dùng dễ dàng tra cứu thông tin phim đang chiếu, xem trailer, lịch chiếu, chọn rạp, chọn suất chiếu và ghế ngồi trực tiếp trên giao diện. Sau khi đặt vé, khách hàng có thể thanh toán qua nhiều phương thức khác nhau (thẻ ngân hàng, ví điện tử, thẻ thành viên) và nhận vé điện tử dưới dạng mã QR để quét khi vào rạp. Ngoài ra, ứng dụng còn tích hợp hệ thống hội viên với tính năng tích điểm, đổi quà và sử dụng các ưu đãi, khuyến mãi đi kèm. Những chức năng này phản ánh luồng nghiệp vụ cốt lõi của một ứng dụng đặt vé hiện đại, bao gồm: quản lý phim, lịch chiếu, rạp chiếu, ghế ngồi, đơn đặt vé, thanh toán và ưu đãi khách hàng. Việc nghiên cứu CGV giúp định hình rõ các yêu cầu nghiệp vụ cần thiết cho hệ thống đặt vé, từ đó xây dựng mô hình dữ liệu phù hợp.

1.1.2 Phân tích nghiệp vụ

Dựa trên khảo sát hệ thống CGV Cinemas, các chức năng chính của ứng dụng đặt vé xem phim có thể phân thành các nhóm: chức năng cho khách hàng, chức năng cho nhân viên, chức năng cho quản trị rạp và các luồng nghiệp vụ cốt lõi.

Chức năng chính:

Đối với khách hàng, hệ thống cung cấp các tính năng như đăng ký/đăng nhập tài khoản, tìm kiếm phim, xem lịch chiếu, đặt vé trực tuyến, thanh toán, quản lý vé và theo dõi khuyến mãi. Với nhân viên, hệ thống hỗ trợ tra cứu thông tin suất chiếu, xác nhận và in vé, quản lý đặt chỗ tại quầy, cũng như hỗ trợ khách hàng trong quá trình sử dụng dịch vụ. Nhóm chức năng dành cho quản trị rạp bao gồm quản lý phim, suất chiếu, phòng chiếu, ghế ngồi, giá vé, chương trình khuyến mãi, cùng với việc thống kê doanh thu và theo dõi tình trạng hoạt động của rạp. Các nhóm chức năng này kết hợp với nhau tạo nên một hệ thống đồng bộ, đáp ứng đầy đủ nhu cầu từ người dùng cuối đến công tác vận hành và quản lý rạp chiếu phim.

Luồng nghiệp vụ cốt lõi:

1. Khách hàng mở ứng dụng → xem danh sách phim → chọn phim.
2. Chọn rạp → chọn ngày giờ → chọn ghế → chọn combo (nếu có).
3. Thực hiện thanh toán → nhận vé điện tử (QR code).
4. Đến rạp → quét QR code tại cổng → vào phòng chiếu.
5. Sau khi xem phim → hệ thống lưu lịch sử, cộng điểm hội viên.

1.2 Mô tả hệ thống đề xuất

1.2.1 Mô tả người dùng và chức năng chính của hệ thống

Ứng dụng đặt vé xem phim được thiết kế để phục vụ nhu cầu mua vé nhanh, chọn ghế trực quan và quản lý vận hành rạp hiệu quả. Hệ thống gồm các vai trò chính, mỗi vai trò có tập chức năng riêng phục vụ cả trải nghiệm khách hàng và nghiệp vụ rạp.

Người dùng của hệ thống

- **Khách hàng (User, End-user)**: là người dùng cuối truy cập app/web để tra cứu phim, đặt vé và sử dụng dịch vụ tại rạp. Khách hàng có thể là người dùng chưa đăng ký (khách vãng lai) hoặc thành viên (có tài khoản, tích điểm, nhận ưu đãi). Nhu cầu chính: thông tin phim rõ ràng, tìm rạp nhanh, chọn ghế trực quan, thanh toán an toàn, nhận vé điện tử, theo dõi lịch sử và ưu đãi cá nhân.
- **Nhân viên rạp (Staff / Box Office / Gate Concession)**: là nhân viên trực tiếp vận hành tại rạp: bán vé tại quầy, xác thực vé QR ở cổng, xử lý đổi/hủy, quản lý tình trạng ghế và hỗ trợ khách. Họ cần giao diện đơn giản để check-in, huỷ/đổi vé, khóa ghế tạm thời, và truy xuất thông tin đơn hàng nhanh.
- **Quản trị hệ thống (Admin / Manager)**:là người quản lý cấp cao của rạp hoặc chuỗi rạp, chịu trách nhiệm cấu hình hệ thống: thêm phim, lập lịch chiếu, điều chỉnh giá, tạo khuyến mãi, xem báo cáo doanh thu, quản lý tài khoản nhân viên và phân quyền. Họ cần công cụ báo cáo, audit log và cấu hình tích hợp (cổng thanh toán, POS).

Chức năng chính của hệ thống

- **Chức năng dành cho khách hàng:** Khách hàng có thể: đăng ký/đăng nhập và quản lý hồ sơ cá nhân; xem danh sách phim, xem chi tiết phim (tóm tắt, thời lượng, thể loại, độ tuổi), và xem trailer; tìm kiếm và lọc phim theo rạp/ngày/thể loại; chọn rạp và suất chiếu; chọn ghế trực quan trên sơ đồ phòng (hiển thị ghế trống/đã bán/không sử dụng); thêm combo đồ ăn/đồ uống vào đơn; áp dụng mã khuyến mãi hoặc sử dụng điểm thành viên khi đặt vé; thanh toán trực tuyến qua thẻ/QR/ví điện tử hoặc thanh toán tại quầy; nhận vé điện tử kèm mã QR và email/xác nhận; xem lịch sử đặt vé, in lại/e-ticket và yêu cầu đổi/hủy theo chính sách; nhận thông báo đẩy về khuyến mãi, thay đổi suất chiếu hoặc nhắc lịch; và đánh giá phim/để lại phản hồi. Mỗi chức năng phải rõ trạng thái (thành công/thất bại) và có thông báo lỗi dễ hiểu.
- **Chức năng dành cho nhân viên rạp:** Nhân viên có thể: check-in khách bằng quét mã QR hoặc nhập mã thủ công; xác thực và huỷ mã QR; thực hiện bán vé trực tiếp tại quầy (tạo đơn, chọn ghế, in vé giấy); quản lý sơ đồ ghế trong ca (khóa/giải phóng ghế, đánh dấu ghế hỏng); xử lý yêu cầu đổi/hủy theo quy định (hoàn tiền, đổi suất); quản lý đơn hàng combo/kho hàng quầy; xem danh sách suất chiếu trong ca và số ghế còn trống; hỗ trợ in lại vé hoặc gửi lại vé điện tử cho khách; và báo cáo vấn đề kỹ thuật hoặc yêu cầu hỗ trợ lên admin. Giao diện dành cho nhân viên cần thao tác nhanh, ít bước, và có kiểm tra phân quyền.
- **Chức năng dành cho quản trị hệ thống:** Admin thực hiện: quản lý phim (thêm, sửa, ngưng chiếu, upload poster/trailer), quản lý rạp và phòng chiếu (thêm rạp, cấu hình phòng,

sơ đồ ghế, loại phòng), lập lịch chiếu và điều chỉnh suất (cập nhật giá, format, thời lượng); thiết lập chính sách giá (giá theo loại ghế, khung giờ, ưu đãi), tạo/quản lý chương trình khuyến mãi và mã giảm giá; quản lý người dùng và phân quyền (tạo tài khoản nhân viên, cấp/thu quyền); giám sát đơn hàng và quản lý tài chính (đối chiếu giao dịch, hoàn tiền, quản lý phiếu thu); báo cáo và phân tích (doanh thu theo rạp/phim/suất, tỉ lệ lấp ghế, báo cáo theo khoảng thời gian); cấu hình tích hợp (cổng thanh toán, hệ thống POS, email/SMS gateway); theo dõi nhật ký hoạt động (audit log), backup dữ liệu và cài đặt bảo mật; và quản lý hệ thống (cấu hình bản thử nghiệm, release, monitor). Tất cả hành động quản trị cần có cơ chế kiểm tra truy vết và phân tầng quyền để tránh thao tác trái phép.

1.2.2 Mô tả các kiểu thực thể, các thuộc tính, mối liên kết

Hệ thống cơ sở dữ liệu đặt vé xem phim quản lý và lưu trữ các thông tin về Người dùng (User), Phim (Movie), Rạp chiếu (Cinema), Phòng chiếu (Room), Suất chiếu (Showtime), Ghế (Seat), Đặt vé (Booking), Thanh toán (Payment), Khuyến mãi (Voucher), Nhân viên (Employee) và các dịch vụ về đồ ăn và thức uống.

Hệ thống đặt vé xem phim cho phép quản lý nhiều rạp cùng lúc. Mỗi rạp được gán mã định danh riêng (CinemaID) lưu kèm theo tên (Name) và địa chỉ (Address). Mỗi rạp được cấp một số hotline riêng (Hotline) cho phép khách hàng gọi đặt vé online.

Mỗi rạp quản lý một số lượng phòng chiếu nhất định. Mỗi phòng chiếu được quản lý bằng mã phòng (RoomID) riêng biệt giữa các rạp. Mỗi phòng được đánh tên riêng (RoomName) và được chia thành nhiều loại phòng (RoomType) với các sức chứa khác nhau (Capacity).

Sức chứa của các phòng là khác nhau nên từng ghế trong mỗi phòng chiếu được quản lý bằng mã riêng biệt (SeatInfo) đảm bảo phân biệt các ghế trong cùng phòng. Mỗi ghế gắn kèm thông tin về hạng ghế (SeatType) cho khách hàng có nhiều sự lựa chọn đa dạng.

Hệ thống cơ sở dữ liệu quản lý và lưu trữ thông tin của khách hàng/người dùng đặt vé. Mỗi người dùng được cấp một mã số định danh riêng (UserID) và yêu cầu thông tin về họ và tên (gồm phần tên (FirstName) và phần họ (LastName)), địa chỉ email duy nhất (Email), mật khẩu tài khoản (Password), số điện thoại (Phone) của người dùng. Ngoài ra, hệ thống cũng lưu trữ thông tin về ngày sinh (Birthday), giới tính (Gender) cũng như ngày tạo tài khoản (CreatedAt) để đáp ứng các dịch vụ mở rộng của hệ thống.

Người dùng đã có thông tin có thể đặt vé xem phim trực tuyến kết hợp sử dụng các mã khuyến mãi tích lũy được từ hệ thống. Mỗi vé được quản lý bằng mã số riêng (BookingID) và ghi nhận thời gian đặt (BookingTime). Một đơn vé bao gồm thông tin về suất chiếu phim đã đặt (ShowtimeID), thông tin về vị trí ghế đã chọn (SeatID). Tổng giá tiền (TotalPrice) của đơn vé sẽ được tính toán dựa trên các thông tin của vé sau khi đã áp dụng các mã khuyến mãi từ người dùng. Tất nhiên, mỗi người dùng có thể đặt bao nhiêu vé tùy ý.

Mỗi suất chiếu được quản lý bằng mã định danh riêng không trùng lặp (ShowtimeID). Suất chiếu chứa thông tin về thời gian chiếu (StartTime và EndTime) cũng như giá cơ bản cho một suất (BasePrice). Mỗi bộ phim có thể có nhiều suất chiếu khác nhau trong ngày và mỗi suất chiếu được quy định một phòng chiếu cụ thể.

Thông tin về các phim được chiếu được hệ thống lưu trữ chi tiết. Cụ thể, mỗi phim được đánh một mã số riêng để quản lý (MovieID), cũng như bao gồm tên phim (Title), thể loại (Genre), thời lượng chiếu (Duration), ngôn ngữ hỗ trợ (Language), độ tuổi giới hạn (AgeRating), mô tả cơ bản về nội dung phim (Description). Ngoài ra, mỗi phim còn lưu thêm thông tin về các diễn

viên trong phim (Actors), đạo diễn bộ phim (Director), ngày khởi chiếu (ReleaseDate) kèm theo poster (Poster) và trailer khởi chiếu (Trailer). Mỗi bộ phim có thể là phần trước hoặc phần sau của một series phim nào đó.

Hệ thống cũng triển khai các chương trình khuyến mãi nhằm thu hút khách hàng. Mỗi chiến dịch khuyến mãi tung ra hàng ngàn voucher khuyến mãi khác nhau. Các voucher được quản lý bằng mã voucher nhận thưởng (Promo Code). Mỗi voucher có kiểu khuyến mãi riêng (DiscountType) như giảm theo phần trăm hay giảm theo số tiền cùng với giá trị được giảm của vé (DiscountValue), đồng thời cũng yêu cầu các điều kiện áp dụng khác nhau cho từng mã (Condition). Mỗi đơn vé được đặt có thể áp dụng nhiều mã khuyến mãi khác nhau (nếu có) để giảm giá vé, đồng thời mỗi mã có thể được sử dụng nhiều lần bởi cho nhiều vé khác nhau. Mỗi mã cũng cần có thời gian bắt đầu (StartDate) và kết thúc (EndDate) để đảm bảo mã không bị lạm dụng.

Hệ thống cung cấp các dịch vụ đồ ăn thức uống đi kèm nhằm nâng cao trải nghiệm khách hàng. Mỗi món ăn thức uống được quản lý bằng mã riêng (FB ID) và bao gồm tên món (FBName), giá tiền (Price) và số lượng mua (Quantity) cho từng món. Mỗi món ăn thức uống có thể được đặt kèm trong mỗi đơn vé (nếu có) và mỗi đơn vé có thể đặt nhiều món khác nhau bao gồm chỉ số quản lý số lượng món đã đặt (Num of Items).

Hệ thống đặt vé xem phim được vận hành bởi đội ngũ nhân viên chuyên nghiệp. Trong đó, mọi nhân viên được cấp một mã số nhân viên riêng (EmployeeID), được phân bổ vị trí làm việc tại từng rạp riêng biệt (CinemaID). Hệ thống có thể kiểm soát được số lượng nhân viên (Num of Employee) đang làm việc tại mỗi rạp. Hệ thống cũng cần lưu thông tin về tên nhân viên (Name), ca làm việc (Shift) và lương thưởng (Salary) của từng nhân viên. Nhân viên được chia thành hai loại chính là quản lý (Manager) và nhân viên (Staff). Với nhân viên, hệ thống lưu trữ về vị trí làm việc (Position) để phân biệt các vai trò khác nhau trong rạp. Với quản lý, hệ thống lưu trữ số lượng các nhân viên (Num of staff) mà họ quản lý trực tiếp.

Các bộ phim được chiếu tại rạp được kiểm soát bởi quản lý rạp. Mỗi bộ phim được quản lý trực tiếp bởi quản lý hoặc được hệ thống xử lý không can thiệp. Do sức mạnh hệ thống xử lý tự động còn hạn chế, quản lý đôi khi phải đảm nhận nhiều bộ phim. Trong quá trình quản lý bộ phim, hệ thống cần lưu thông tin cho biết hành động của quản lý rạp với bộ phim (ActionType) đó bao gồm thêm, sửa, dừng chiếu hay các cập nhật khác, đồng thời ghi nhận thời gian thực hiện hành động (ActionDate). Tương tự với bộ phim, mỗi suất chiếu trong rạp cũng phải được quản lý bởi một nhân viên quản lý trong khi mỗi quản lý có thể quản lý nhiều suất chiếu khác nhau. Suất chiếu được tạo ra và kiểm duyệt dựa trên các thông tin mà quản lý rạp cung cấp bao gồm lịch dự kiến (Schedule Date), thời lượng dự kiến (Duration Adjust), hình thức suất chiếu (Format) và giá vé điều chỉnh (PriceAdjust). Quản lý rạp cũng chịu trách nhiệm quản lý các phòng chiếu trong rạp. Mỗi phòng chiếu được quản lý bởi một quản lý và mỗi quản lý có thể quản lý nhiều phòng chiếu khác nhau. Hệ thống cần lưu thông tin trong quá trình quản lý các phòng chiếu cho biết hành động của quản lý với phòng chiếu (ActionType) đó bao gồm quản lý cấu trúc phòng, loại phòng và sơ đồ bố trí ghế, đồng thời ghi nhận thời gian thực hiện hành động (ActionDate).

Bên cạnh việc đặt vé trực tuyến, khách hàng cũng có thể đến trực tiếp quầy bán vé của rạp để mua vé. Mỗi quầy bán vé được định danh một mã số riêng (CounterID) cho phép một nhân viên bán vé vận hành quầy. Trong quá trình bán vé, hệ thống cần lưu thông tin về thời gian bán vé (SellTime) và trạng thái xuất vé (PrintStatus) để đảm bảo vé không bị làm giả. Đối với khách hàng làm mất vé, nhân viên có thể hỗ trợ in lại vé nếu khách hàng cung cấp đầy đủ thông tin về mã vé (BookingID) và thông tin cá nhân đã đăng ký. Ngoài việc bán vé, nhân viên còn chịu trách nhiệm quản lý sơ đồ ghế. Một nhân viên được phân công quản lý một dãy các ghế và đồng

thời mỗi dãy ghế cũng được kiểm tra bởi nhiều nhân viên khác nhau. Hệ thống cho phép nhân viên cập nhật tình trạng của ghế (SeatlockStatus) bao gồm khóa ghế, mở khóa ghế hay đánh dấu ghế hỏng, đồng thời ghi nhận thời gian thực hiện cập nhật (UpdateTime). Bên cạnh đó, nhân viên còn chịu trách nhiệm xử lý các yêu cầu đổi/hủy giao dịch tại quầy. Mỗi yêu cầu giao dịch được xử lý bởi một nhân viên và mỗi nhân viên có thể xử lý nhiều yêu cầu khác nhau. Hệ thống cần lưu thông tin về loại yêu cầu (RequestType) gồm đổi, hủy các giao dịch đã thực hiện kèm với số tiền hoàn trả (RefundAmount) và thời điểm các yêu cầu được xử lý (ProcessDate).

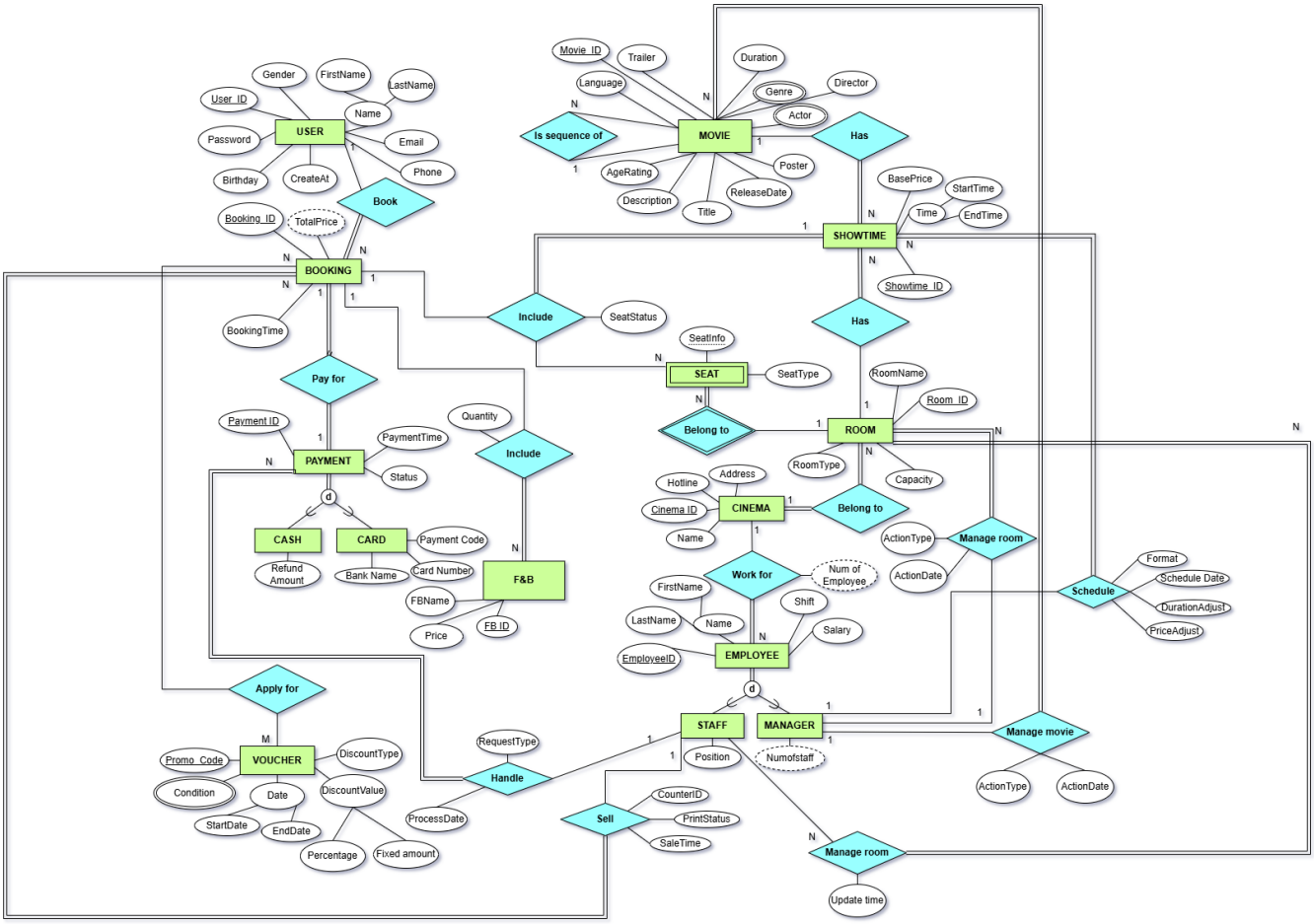
Hệ thống cho phép người dùng thanh toán bằng nhiều hình thức khác nhau và lưu trữ thông tin thanh toán. Mỗi thanh toán/hóa đơn có mã hóa đơn riêng (PaymentID) và thuộc về một vé duy nhất (BookingID). Mỗi thanh toán có thể là giao dịch qua thẻ tín dụng, ví điện tử hay trực tiếp bằng tiền mặt. Đối với các thanh toán trực tuyến, hệ thống lưu trữ thông tin về mã giao dịch (PaymentCode) do bên thứ ba cung cấp và thông tin về số tài khoản (Card Number) và ngân hàng xử lý giao dịch (Bank Name). Đối với các thanh toán trực tiếp tại quầy, hệ thống lưu trữ thông tin về tên nhân viên thu ngân (StaffName). Trạng thái thanh toán (Status) cũng cần được lưu cho phép hệ thống kiểm tra giao dịch hoàn tất hay chưa. Khi giao dịch hoàn tất, hệ thống cũng cần ghi nhận thời gian thực hiện giao dịch (PaymentTime), đảm bảo tính minh bạch cho hệ thống, nâng cao sự tín nhiệm với khách hàng.

1.3 Mô tả các ràng buộc ngữ nghĩa

- Sức chứa (Capacity) của mỗi phòng chiếu (Room) phải có giá trị dương.
- Thời lượng chiếu (Duration) của mỗi bộ phim (Movie) phải có giá trị dương.
- Thời gian bắt đầu chiếu (StartTime) của mỗi suất chiếu (Showtime) phải sớm hơn thời gian kết thúc của nó (EndTime).
- Ngày công chiếu (ReleaseDate) của mỗi bộ phim (Movie) phải sớm hơn ngày khởi chiếu (Showtime.StartTime).
- Giá cơ bản của mỗi suất (BasePrice) phải có giá trị dương.
- Hai suất chiếu (Showtime) cùng một phòng chiếu (Room) không được trùng lắp nhau.
- Mật khẩu đăng nhập (Password) của người dùng (User) phải là chuỗi có ít nhất 8 kí tự, trong đó phải có ít nhất 1 kí tự chữ hoa, ít nhất 1 kí tự số, ít nhất 1 kí tự đặc biệt thuộc tập @, $, #, %, !, &.
- Ngày sinh (Birthday) của người dùng (User) phải sớm hơn ngày hiện tại.
- Số điện thoại (Phone) của người dùng (User) phải là chuỗi có 10 chữ số.
- Thời gian đặt vé (BookingTime) phải sớm hơn thời gian bắt đầu của suất chiếu (Showtime.StartTime).
- Số lượng nhân viên (Num of Employee) đang làm việc tại mỗi rạp được tính bằng tổng số nhân viên có EmployeeID tương ứng trong bảng Employee.
- Giới tính (Gender) của người dùng (User) phải thuộc tập "Male", "Female".
- Giá tiền (Price) của mỗi món ăn thức uống phải có giá trị dương.

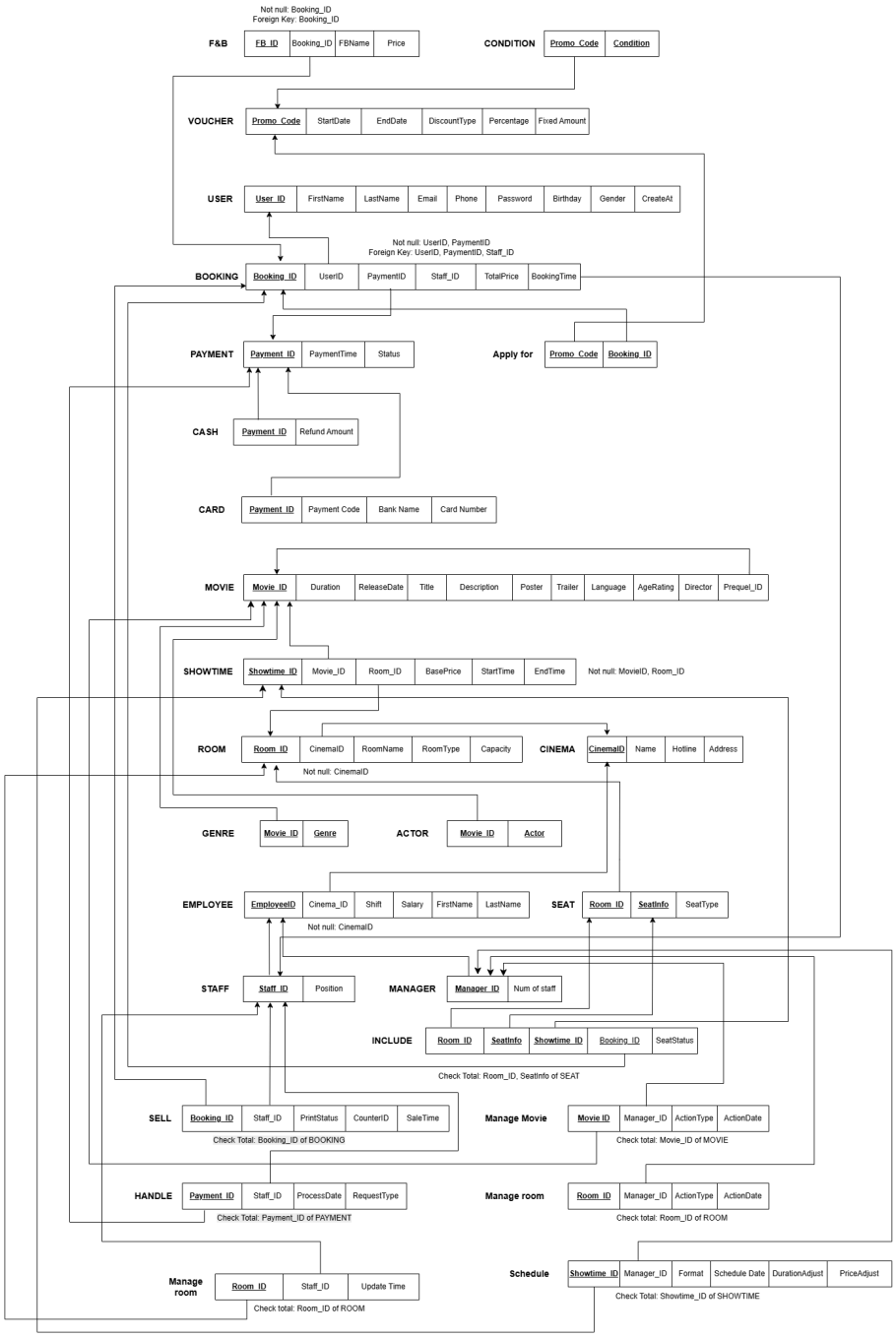
- Số lượng món đã đặt (Num of Items) được tính bằng tổng số lượng món (Quantity) ứng với các sản phẩm được đặt trong bảng F&B.
- Hạng ghế (SeatType) phải thuộc tập "Standard", "VIP", "Couple", "Deluxe".
- Thời gian bắt đầu (Discount.StartDate) của mỗi voucher khuyến mãi phải sớm hơn (Discount.EndDate).
- Ca làm việc (Shift) của nhân viên (Employee) thuộc tập "Morning", "Afternoon", "Evening".
- Lương thưởng (Salary) của nhân viên (Employee) phải có giá trị dương.
- TotalPrice = tổng các Showtime.BasePrice + tổng các F&B.Price - Voucher.DiscountValue.
- Thời gian thực hiện giao dịch (PaymentTime) phải trễ hơn thời gian đặt (BookingTime).

2 Thiết kế EERD



Hình 1: Sơ đồ EERD cho hệ thống đặt vé xem phim

3 Ánh xạ lược đồ EERD sang lược đồ CSDL



Hình 2: Ánh xạ lược đồ EERD sang lược đồ CSDL

4 Đường dẫn truy cập các sơ đồ

Đường dẫn truy cập các sơ đồ như sau: *Link*

5 Tạo bảng và dữ liệu mẫu

Dựa trên thiết kế EERD và lược đồ quan hệ đã phân tích ở Bài tập lớn 1, nhóm tiến hành hiện thực hóa cơ sở dữ liệu trên hệ quản trị SQL Server. Phần này trình bày các câu lệnh DDL (Data Definition Language) để tạo bảng, thiết lập các ràng buộc toàn vẹn và DML (Data Manipulation Language) để thêm dữ liệu mẫu.

5.1 Tạo bảng và các ràng buộc (Constraints)

Các bảng dữ liệu được tạo theo thứ tự để đảm bảo tính toàn vẹn tham chiếu (tạo bảng cha trước, bảng con sau). Các ràng buộc miền giá trị (Domain Constraints) được cài đặt trực tiếp bằng từ khóa `CHECK`.

5.1.1 Bảng Rạp chiếu (CINEMA) và Phòng chiếu (ROOM)

Bảng `CINEMA` lưu trữ thông tin các cụm rạp. Bảng `ROOM` lưu trữ thông tin phòng chiếu, có ràng buộc sức chứa (`Capacity`) phải lớn hơn 0 và loại phòng (`RoomType`) thuộc tập quy định.

```
CREATE TABLE CINEMA (
    CinemaID VARCHAR(10) PRIMARY KEY,
    Name NVARCHAR(100) NOT NULL,
    Hotline VARCHAR(15),
    Address NVARCHAR(200)
);

CREATE TABLE ROOM (
    Room_ID VARCHAR(10) PRIMARY KEY,
    CinemaID VARCHAR(10) NOT NULL,
    RoomName NVARCHAR(50) NOT NULL,
    RoomType NVARCHAR(20),
    Capacity INT NOT NULL,

    FOREIGN KEY (CinemaID) REFERENCES CINEMA(CinemaID),
    CONSTRAINT CK_Room_Capacity CHECK (Capacity > 0),
    CONSTRAINT CK_Room_Type CHECK (RoomType IN ('2D', '3D', 'IMAX', '4DX', 'Gold
    Class', 'Sweetbox', 'VIP')),
);
```

5.1.2 Bảng Ghế (SEAT)

Bảng `SEAT` lưu trữ thông tin ghế trong phòng chiếu, có ràng buộc hạng ghế (`SeatType`) thuộc tập quy định và thông tin ghế (`SeatInfo`) phải có định dạng gồm một chữ cái và các số.

```
  CREATE TABLE SEAT (
    Room_ID VARCHAR(10),
```

```
    SeatInfo VARCHAR(10),
    SeatType NVARCHAR(20) NOT NULL,

    PRIMARY KEY (Room_ID, SeatInfo),
    FOREIGN KEY (Room_ID) REFERENCES ROOM(Room_ID),
    CONSTRAINT CK_Seat_Type CHECK (SeatType IN ('Standard', 'VIP', 'Couple', '
    Deluxe')),
    CONSTRAINT CK_Seat_SeatInfo CHECK (SeatInfo LIKE '[A-Z]%' AND ISNUMERIC(
    SUBSTRING(SeatInfo, 2, LEN(SeatInfo) - 1)) = 1)
);
```

5.1.3 Bảng Khách hàng (USER) và Phim (MOVIE)

Bảng USER có các ràng buộc quan trọng về định dạng số điện thoại, mật khẩu phức tạp và độ tuổi. Bảng MOVIE kiểm tra thời lượng phim phải là số dương và giới hạn độ tuổi hợp lệ.

```
CREATE TABLE [USER] (
    UserID VARCHAR(10) PRIMARY KEY,
    FirstName NVARCHAR(50) NOT NULL,
    LastName NVARCHAR(50) NOT NULL,
    Email VARCHAR(100) UNIQUE NOT NULL,
    Phone VARCHAR(15) NOT NULL,
    Password VARCHAR(50) NOT NULL,
    Birthday DATE,
    Gender NVARCHAR(10),
    CreatedAt DATETIME DEFAULT GETDATE(),

    CONSTRAINT CK_User_Phone CHECK (LEN(Phone) = 10 AND ISNUMERIC(Phone) = 1),
    CONSTRAINT CK_User_Gender CHECK (Gender IN ('Male', 'Female')),
    CONSTRAINT CK_User_Birthday CHECK (Birthday < GETDATE()),
    CONSTRAINT CK_User_Password CHECK (
        LEN(Password) >= 8 AND
        Password COLLATE Latin1_General_BIN LIKE '%[A-Z]%' AND
        Password COLLATE Latin1_General_BIN LIKE '%[0-9]%' AND
        Password COLLATE Latin1_General_BIN LIKE '%[@$#%!&]%'
    )
);

CREATE TABLE MOVIE (
    Movie_ID VARCHAR(10) PRIMARY KEY,
    Title NVARCHAR(100) NOT NULL,
    Duration INT NOT NULL,
    ReleaseDate DATE NOT NULL,
    Description NVARCHAR(MAX),
    Poster VARCHAR(200),
    Trailer VARCHAR(200),
    Language NVARCHAR(30),
    AgeRating VARCHAR(5),
    Director NVARCHAR(50),
    Prequel_ID VARCHAR(10),

    FOREIGN KEY (Prequel_ID) REFERENCES MOVIE(Movie_ID),
    CONSTRAINT CK_Movie_Duration CHECK (Duration > 0),
    CONSTRAINT CK_Movie_AgeRating CHECK (AgeRating IN ('P', 'T13', 'T16', 'T18', '
    K') )
);
```

5.1.4 Bảng Thể loại phim (Genre) và Diễn viên (Actor)

Bảng `Genre` và bảng `Actor` lưu trữ thông tin thể loại phim và diễn viên, có ràng buộc thể loại thuộc tập quy định.

```
CREATE TABLE GENRE (
    Movie_ID VARCHAR(10),
    Genre NVARCHAR(50),
    PRIMARY KEY (Movie_ID, Genre),
    FOREIGN KEY (Movie_ID) REFERENCES MOVIE(Movie_ID),
    CONSTRAINT CK_Genre_Type CHECK (Genre IN ('Action', 'Comedy', 'Drama', 'Horror
    ', 'Romance', 'Sci-Fi', 'Documentary', 'Animation', 'Thriller', 'Fantasy', '
    Adventure', 'Musical', 'Biography', 'Crime', 'Family', 'Mystery', 'War', '
    Western', 'Sport', 'History', 'Mysterious', 'Mythical') )
);

CREATE TABLE ACTOR (
    Movie_ID VARCHAR(10),
    Actor NVARCHAR(50),
    PRIMARY KEY (Movie_ID, Actor),
    FOREIGN KEY (Movie_ID) REFERENCES MOVIE(Movie_ID)
);
```

5.1.5 Bảng Suất chiếu (SHOWTIME) và Thanh toán (PAYMENT)

Bảng `SHOWTIME` chứa thông tin suất chiếu và các ràng buộc quan trọng: Giờ bắt đầu phải nhỏ hơn giờ kết thúc và giá vé phải dương và suất chiếu không được trùng lấp trong cùng một phòng và thời gian suất chiếu (EndTime - StartTime) phải lớn hơn hoặc bằng thời lượng gốc của bộ phim (sẽ được kiểm tra bằng Trigger). Bảng `PAYMENT` quản lý thông tin thanh toán với ràng buộc trạng thái thanh toán `PaymentStatus` phải thuộc tập quy định và thời gian thanh toán phải sau thời gian đặt vé và thời gian thanh toán phải trong vòng 24h kể từ thời gian đặt vé (sẽ được kiểm tra bằng Trigger).

```
CREATE TABLE SHOWTIME (
    Showtime_ID VARCHAR(10) PRIMARY KEY,
    Movie_ID VARCHAR(10) NOT NULL,
    Room_ID VARCHAR(10) NOT NULL,
    BasePrice DECIMAL(18, 2) NOT NULL,
    StartTime DATETIME NOT NULL,
    EndTime DATETIME NOT NULL,

    FOREIGN KEY (Movie_ID) REFERENCES MOVIE(Movie_ID),
    FOREIGN KEY (Room_ID) REFERENCES ROOM(Room_ID),
    CONSTRAINT CK_Showtime_Price CHECK (BasePrice > 0),
    CONSTRAINT CK_Showtime_Time CHECK (StartTime < EndTime)
);

CREATE TABLE PAYMENT (
    Payment_ID VARCHAR(10) PRIMARY KEY,
    PaymentTime DATETIME,
    Status NVARCHAR(20) -- Success, Pending, Failed
    CONSTRAINT CK_Payment_Status CHECK (Status IN ('Success', 'Pending', 'Failed')
     )
);
```

5.1.6 Bảng Nhân viên tổng quát (EMPLOYEE), Nhân viên (STAFF) và Quản lý (MANAGER)

Bảng `EMPLOYEE` lưu trữ thông tin chung của nhân viên. Bảng `STAFF` và bảng `MANAGER` kế thừa từ bảng `EMPLOYEE` để lưu trữ thông tin riêng biệt của từng loại nhân viên, có ràng buộc lương thưởng phải dương và ca làm việc thuộc tập quy định. Bảng `MANAGER` có ràng buộc số lượng nhân viên quản lý phải lớn hơn hoặc bằng 0. Bảng `STAFF` có ràng buộc vị trí làm việc thuộc tập quy định.

```
CREATE TABLE EMPLOYEE (
    EmployeeID VARCHAR(10) PRIMARY KEY,
    Cinema_ID VARCHAR(10) NOT NULL,
    FirstName NVARCHAR(50),
    LastName NVARCHAR(50),
    Shift NVARCHAR(20),
    Salary DECIMAL(18, 2),

    FOREIGN KEY (Cinema_ID) REFERENCES CINEMA(CinemaID),
    CONSTRAINT CK_Employee_Salary CHECK (Salary > 0),
    CONSTRAINT CK_Employee_Shift CHECK (Shift IN ('Morning', 'Afternoon', 'Evening
    '))
);

CREATE TABLE STAFF (
    Staff_ID VARCHAR(10) PRIMARY KEY,
    Position NVARCHAR(50),
    FOREIGN KEY (Staff_ID) REFERENCES EMPLOYEE(EmployeeID),
    CONSTRAINT CK_Staff_Position CHECK (Position IN ('Cashier', 'Usher', 'Cleaner'
    , 'Technician', 'FoodService'))
);

CREATE TABLE MANAGER (
    Manager_ID VARCHAR(10) PRIMARY KEY,
    Num_of_staff INT,
    FOREIGN KEY (Manager_ID) REFERENCES EMPLOYEE(EmployeeID),
    CONSTRAINT CK_Manage_Staff CHECK (Num_of_staff >= 0)
);
```

5.1.7 Bảng Đặt vé (BOOKING)

Bảng `BOOKING` lưu trữ thông tin đặt vé, có ràng buộc thời gian đặt vé phải trước thời gian bắt đầu suất chiếu (sẽ được kiểm tra bằng Trigger).

```
CREATE TABLE BOOKING (
    Booking_ID VARCHAR(10) PRIMARY KEY,
    UserID VARCHAR(10) NOT NULL,
    PaymentID VARCHAR(10) UNIQUE, -- 1-1 Relationship
    Staff_ID VARCHAR(10),
    TotalPrice DECIMAL(18, 2),
    BookingTime DATETIME DEFAULT GETDATE(),

    FOREIGN KEY (UserID) REFERENCES [USER](UserID),
    FOREIGN KEY (PaymentID) REFERENCES PAYMENT(Payment_ID),
    FOREIGN KEY (Staff_ID) REFERENCES STAFF(Staff_ID)
);
```

5.1.8 Bảng Khuyến mãi (VOUCHER), Điều kiện áp dụng voucher (CONDITION), Áp dụng voucher (APPLY_FOR)

Bảng VOUCHER lưu trữ thông tin voucher khuyến mãi với các ràng buộc về ngày bắt đầu và kết thúc và thuộc tính DiscountType phải thuộc tập quy định. Bảng CONDITION lưu trữ các điều kiện áp dụng voucher. Bảng APPLY_FOR thể hiện mối quan hệ nhiều-nhiều giữa bảng BOOKING và bảng VOUCHER.

```
CREATE TABLE VOUCHER (
    Promo_Code VARCHAR(20) PRIMARY KEY,
    StartDate DATETIME NOT NULL,
    EndDate DATETIME NOT NULL,
    DiscountType NVARCHAR(20), -- Percent, Fixed
    Percentage FLOAT,
    FixedAmount DECIMAL(18, 2),
    CONSTRAINT CK_Voucher_Date CHECK (StartDate < EndDate),
    CONSTRAINT CK_Voucher_DiscountType CHECK (DiscountType IN ('Percent', 'Fixed')
    )
);

CREATE TABLE CONDITION (
    Promo_Code VARCHAR(20),
    Condition NVARCHAR(200),
    PRIMARY KEY (Promo_Code, Condition),
    FOREIGN KEY (Promo_Code) REFERENCES VOUCHER(Promo_Code)
);

CREATE TABLE APPLY_FOR (
    Promo_Code VARCHAR(20),
    Booking_ID VARCHAR(10),
    PRIMARY KEY (Promo_Code, Booking_ID),
    FOREIGN KEY (Promo_Code) REFERENCES VOUCHER(Promo_Code),
    FOREIGN KEY (Booking_ID) REFERENCES BOOKING(Booking_ID)
);
```

5.1.9 Bảng Đồ ăn & Nước uống (F&B) và Chi tiết đặt ghế (INCLUDE_SEAT)

Bảng F&B lưu trữ thông tin đồ ăn và nước uống với ràng buộc giá tiền phải dương. Bảng INCLUDE_SEAT thể hiện mối quan hệ 1 - 1 - nhiều giữa bảng BOOKING, bảng SHOWTIME và bảng SEAT với ràng buộc SeatStatus phải thuộc tập quy định.

```
CREATE TABLE F_B (
    FB_ID VARCHAR(10),
    Booking_ID VARCHAR(10),
    FBName NVARCHAR(50),
    Price DECIMAL(18, 2),

    PRIMARY KEY (FB_ID, Booking_ID),
    FOREIGN KEY (Booking_ID) REFERENCES BOOKING(Booking_ID),
    CONSTRAINT CK_FB_Price CHECK (Price > 0)
);

CREATE TABLE INCLUDE_SEAT (
  Room_ID VARCHAR(10),
    SeatInfo VARCHAR(10),
    Showtime_ID VARCHAR(10),
    Booking_ID VARCHAR(10),
```

```
    SeatStatus NVARCHAR(20), -- Booked, Sold

    PRIMARY KEY (Room_ID, SeatInfo, Showtime_ID),
    FOREIGN KEY (Room_ID, SeatInfo) REFERENCES SEAT(Room_ID, SeatInfo),
    FOREIGN KEY (Showtime_ID) REFERENCES SHOWTIME(Showtime_ID),
    FOREIGN KEY (Booking_ID) REFERENCES BOOKING(Booking_ID),
    CONSTRAINT CK_IncludeSeat_Status CHECK (SeatStatus IN ('Booked', 'Sold'))
  );
```

5.1.10 Bảng Quản lý phim (MANAGE_MOVIE), Quản lý suất chiếu (SCHEDULE) và Quản lý phòng chiếu (MANAGE_ROOM)

Bảng `MANAGE_MOVIE`, bảng `SCHEDULE` và bảng `MANAGE_ROOM` lưu trữ thông tin quản lý phim, suất chiếu và phòng chiếu tương ứng. Được thực hiện bởi các quản lý rạp với các ràng buộc về loại hành động (`ActionType`) và loại điều chỉnh (`Format`) phải thuộc tập quy định.

```
CREATE TABLE MANAGE_MOVIE (
    Movie_ID VARCHAR(10),
    Manager_ID VARCHAR(10),
    ActionType NVARCHAR(50),
    ActionDate DATETIME,

    PRIMARY KEY (Movie_ID, Manager_ID, ActionDate),
    FOREIGN KEY (Movie_ID) REFERENCES MOVIE(Movie_ID),
    FOREIGN KEY (Manager_ID) REFERENCES MANAGER(Manager_ID),
    CONSTRAINT CK_ManageMovie_ActionType CHECK (ActionType IN ('Add', 'Update', '
    Remove'))
);

CREATE TABLE MANAGE_ROOM (
    Room_ID VARCHAR(10),
    Manager_ID VARCHAR(10),
    ActionType NVARCHAR(50),
    ActionDate DATETIME,

    PRIMARY KEY (Room_ID, Manager_ID, ActionDate),
    FOREIGN KEY (Room_ID) REFERENCES ROOM(Room_ID),
    FOREIGN KEY (Manager_ID) REFERENCES MANAGER(Manager_ID),
    CONSTRAINT CK_ManageRoom_ActionType CHECK (ActionType IN ('Add', 'Update', '
    Remove'))
);

CREATE TABLE SCHEDULE (
    Showtime_ID VARCHAR(10) PRIMARY KEY,
    Manager_ID VARCHAR(10),
    Format NVARCHAR(20),
    ScheduleDate DATE,
    DurationAdjust INT,
    PriceAdjust DECIMAL(18, 2),

    FOREIGN KEY (Showtime_ID) REFERENCES SHOWTIME(Showtime_ID),
    FOREIGN KEY (Manager_ID) REFERENCES MANAGER(Manager_ID)
);
```

5.1.11 Bảng Bán vé (SELL)

Bảng SELL lưu trữ thông tin bán vé (thời điểm bán vé, trạng thái in của vé).

```
CREATE TABLE SELL (
    Booking_ID VARCHAR(10) PRIMARY KEY,
    Staff_ID VARCHAR(10),
    PrintStatus BIT,
    SaleTime DATETIME,

    FOREIGN KEY (Booking_ID) REFERENCES BOOKING(Booking_ID),
    FOREIGN KEY (Staff_ID) REFERENCES STAFF(Staff_ID)
);
```

5.1.12 Bảng Xử lý (HANDLE)

Bảng HANDLE lưu trữ thông tin xử lý thanh toán.

```
CREATE TABLE HANDLE (
    Payment_ID VARCHAR(10),
    Staff_ID VARCHAR(10),
    ProcessDate DATETIME,

    PRIMARY KEY (Payment_ID, Staff_ID),
    FOREIGN KEY (Payment_ID) REFERENCES PAYMENT(Payment_ID),
    FOREIGN KEY (Staff_ID) REFERENCES STAFF(Staff_ID),
);
```

5.1.13 Bảng Xử lý (MANAGE_ROOM_STATUS)

Bảng MANAGE_ROOM_STATUS lưu trữ thông tin phân công nhân viên quản lý phòng chiếu.

```
CREATE TABLE MANAGE_ROOM_STATUS (
    Room_ID VARCHAR(10),
    Staff_ID VARCHAR(10),
    UpdateTime DATETIME,

    PRIMARY KEY (Room_ID, Staff_ID, UpdateTime),
    FOREIGN KEY (Room_ID) REFERENCES ROOM(Room_ID),
    FOREIGN KEY (Staff_ID) REFERENCES STAFF(Staff_ID)
);
```

5.2 Tạo dữ liệu mẫu

Để kiểm chứng tính đúng đắn của thiết kế CSDL và hoạt động của các ràng buộc toàn vẹn, nhóm đã xây dựng một bộ dữ liệu mẫu mô phỏng hệ thống rạp phim CGV thực tế tại TP.HCM.

5.3 Mô tả dữ liệu chi tiết

5.3.1 Hệ thống Rạp và Phòng chiếu

Hệ thống bao gồm 10 cụm rạp CGV lớn tại TP.HCM. Mỗi rạp có các loại phòng chiếu đa dạng (IMAX, 4DX, Sweetbox, Gold Class).

```
-- Insert Rap CGV
INSERT INTO CINEMA (CinemaID, Name, Hotline, Address) VALUES
('CN001', N'CGV Vincom Dong Khoi', '19006017', N'72 Le Thanh Ton...'),
('CN002', N'CGV Landmark 81', '19006017', N'772 Dien Bien Phu...');

-- Insert Phong chieu voi suc chua va loai phong cu the
INSERT INTO ROOM (Room_ID, CinemaID, RoomName, RoomType, Capacity) VALUES
('R001', 'CN001', N'Cinema 1', 'IMAX', 250),
('R005', 'CN003', N'Cinema 1', 'Gold Class', 80);
```

Listing 1: Dữ liệu Rạp và Phòng chiếu

5.3.2 Người dùng và Nhân sự

- **Khách hàng (User):** Bao gồm 23 tài khoản, trong đó có tích hợp thông tin của các thành viên trong nhóm thực hiện đề tài (US019 - US023) để phục vụ demo.
- **Nhân viên (Employee):** Phân chia rõ ràng giữa cấp Quản lý (Manager) và Nhân viên (Staff) với các ca làm việc (Shift) và mức lương khác nhau.

```
-- Insert Thanh vien nhom
INSERT INTO [USER] (UserID, FirstName, LastName, Email...) VALUES
('US019', N'Nguyen', N'Phu Vinh', 'svPV@gmail.com'... ),
('US023', N'Le', N'Minh Tri', 'svLMT@gmail.com'... );

-- Phan cap Nhan vien va Quan ly
INSERT INTO EMPLOYEE (EmployeeID, Shift, Salary...) VALUES ('EM001', 'Morning',
    5000000...);
INSERT INTO MANAGER (Manager_ID, Num_of_staff) VALUES ('EM001', 10);
INSERT INTO STAFF (Staff_ID, Position) VALUES ('EM002', 'Cashier');
```

5.3.3 Phim và Lịch chiếu

Dữ liệu phim bao gồm đầy đủ thông tin về Đạo diễn, Diễn viên, Thể loại và giới hạn độ tuổi. Đặc biệt có Series phim "Lật Mặt" từ phần 1 đến phần 8 để kiểm tra truy vấn phim theo series (Prequel).

```
INSERT INTO MOVIE (Movie_ID, Title, Duration, AgeRating...) VALUES
('MV001', N'Mai', 131, 'T18'...),
('MV002', N'Dune: Part Two', 166, 'T16'...),
('MV018', N'Lat Mat 7: Mot Dieu Uoc', 105, 'P'...);

INSERT INTO GENRE (Movie_ID, Genre) VALUES
('MV001', 'Drama'),
('MV002', 'Sci-Fi'),
('MV018', 'Action');
```

```
10
11 INSERT INTO ACTOR (Movie_ID, Actor) VALUES
12 ('MV001', N'Tran Nghia'),
13 ('MV002', N'Timothee Chalamet'),
14 ('MV018', N'Thanh Loc');
15
16 INSERT INTO SHOWTIME (Showtime_ID, Movie_ID, Room_ID, BasePrice, StartTime,
       EndTime) VALUES
17 ('ST001', 'MV001', 'R001', 120000, '2024-07-01 14:00', '2024-07-01 16:11'),
18 ('ST002', 'MV002', 'R002', 150000, '2024-07-01 17:00', '2024-07-01 19:46');
```

5.3.4 Giao dịch Đặt vé (Transaction Flow)

Đây là phần phức tạp nhất, mô phỏng quy trình nghiệp vụ trọn vẹn:

1. Khách hàng chọn Suất chiếu (`SHOWTIME`).
2. Chọn Ghế (`INCLUDE_SEAT`) và Combo bắp nước (`F_B`).
3. Áp dụng mã giảm giá (`VOUCHER` - ví dụ: SUMMER2024, TET2024).
4. Thanh toán (`PAYMENT` - Tiền mặt hoặc Thẻ).

```
1  -- 1. Tao Giao dich Payment truoc
2  INSERT INTO PAYMENT (Payment_ID, Status) VALUES ('PM001', 'Success');
3
4  INSERT INTO CASH (Payment_ID, RefundAmount) VALUES
5  ('PM001', 0), ('PM002', 0), ('PM003', 0), ('PM004', 0), ('PM009', 0);
6
7  INSERT INTO CARD (Payment_ID, PaymentCode, BankName, CardNumber) VALUES
8  ('PM005', 'TXN005', 'ACB', '9704123456790'),
9  ('PM006', 'TXN006', 'Vietcombank', '9704123456789'),
10
11 -- 2. Tao Booking ket noi User va Payment
12 INSERT INTO BOOKING (Booking_ID, UserID, PaymentID...) VALUES
13 ('BK001', 'US001', 'PM001'...);
14
15 -- 3. Chi tiet Ghe va Do an
16 INSERT INTO INCLUDE_SEAT (Room_ID, SeatInfo, Booking_ID...) VALUES
17 ('R001', 'A1', 'BK001'...);
18 INSERT INTO F_B (FB_ID, Booking_ID, FBName...) VALUES
19 ('CB01', 'BK001', N'Combo Popcorn Solo'...);
20
21 -- 4. Ap dung Voucher
22 INSERT INTO APPLY_FOR (Promo_Code, Booking_ID) VALUES ('TET2024', 'BK001');
```

5.3.5 Các hoạt động của MANAGER

Dữ liệu ghi nhận các hoạt động quản lý của các quản lý rạp, bao gồm thêm, sửa, xóa phim và phòng chiếu, cũng như điều chỉnh lịch chiếu.

```
1 --Bang Manage_Movie
2 INSERT INTO MANAGE_MOVIE (Movie_ID, Manager_ID, ActionType, ActionDate) VALUES
3 ('MV001', 'EM004', 'Add', '2024-01-20 08:00:00'),
```

```
('MV002', 'EM004', 'Add', '2024-02-15 09:00:00'),
('MV003', 'EM006', 'Add', '2024-02-20 10:00:00'),
('MV004', 'EM004', 'Update', '2024-03-16 11:00:00'),

--Bang Manage_Room
INSERT INTO MANAGE_ROOM (Room_ID, Manager_ID, ActionType, ActionDate) VALUES
('R001', 'EM001', 'Add', '2021-12-01 08:00:00'),
('R002', 'EM001', 'Add', '2021-12-01 09:00:00'),
('R003', 'EM004', 'Add', '2021-01-10 10:00:00'),
('R005', 'EM006', 'Add', '2020-11-15 08:00:00'),


--Bang Schedule
INSERT INTO SCHEDULE (Showtime_ID, Manager_ID, Format, ScheduleDate,
    DurationAdjust, PriceAdjust) VALUES
('SH001', 'EM001', '2D', '2024-02-10', 0, 0),
('SH002', 'EM001', '2D', '2024-02-10', 0, 10000),
('SH003', 'EM001', 'IMAX', '2024-02-25', 0, 20000),
('SH004', 'EM004', '3D', '2024-03-05', 0, 0),
```

5.3.6 Các hoạt động của Nhân viên

Dữ liệu ghi nhận các hoạt động của nhân viên trong việc xử lý thanh toán và bán vé cho khách hàng. Đồng thời, quản lý phòng chiếu.

```
INSERT INTO SELL (Booking_ID, Staff_ID, PrintStatus, SaleTime) VALUES
('BK001', 'EM003', 1, '2024-02-14 16:55:00'),
('BK002', 'EM003', 1, '2024-02-14 20:25:00'),
('BK003', 'EM003', 1, '2024-03-02 18:25:00'),
('BK004', 'EM002', 1, '2024-03-09 09:25:00'),
('BK009', 'EM007', 1, '2024-07-27 18:05:00');

INSERT INTO HANDLE (Payment_ID, Staff_ID, ProcessDate) VALUES
('PM001', 'EM003', '2024-02-14 17:00:00'),
('PM002', 'EM003', '2024-02-14 20:30:00'),
('PM003', 'EM003', '2024-03-02 18:30:00'),
('PM004', 'EM002', '2024-03-09 09:30:00'),
('PM009', 'EM007', '2024-07-27 19:30:00');

INSERT INTO MANAGE_ROOM_STATUS (Room_ID, Staff_ID, UpdateTime) VALUES
('R001', 'EM003', '2024-02-14 16:00:00'),
('R001', 'EM003', '2024-02-14 23:30:00'),
('R002', 'EM003', '2024-03-02 22:00:00'),
('R003', 'EM002', '2024-03-09 12:00:00'),
('R004', 'EM002', '2024-03-17 01:30:00'),
('R005', 'EM005', '2024-03-30 17:15:00'),
```

6 Viết các thủ tục, trigger và hàm

6.1 Viết thủ tục cho bảng USER

Trong phần này, nhóm chúng em sẽ trình bày các thủ tục để thêm (insert), sửa (update) và xóa (delete) dữ liệu vào bảng `USER` trong cơ sở dữ liệu. Các thủ tục này đều bao gồm việc kiểm tra dữ liệu hợp lệ (validate) để đảm bảo rằng dữ liệu được nhập vào bảng đáp ứng các ràng buộc

cụ thể. Đồng thời, các thông báo lỗi cụ thể cũng được xuất ra khi có lỗi, giúp người dùng dễ dàng xác định vấn đề.

6.1.1 Thủ tục sp_InsertUser

```
CREATE PROCEDURE sp_InsertUser
@UserID VARCHAR(10),
@FirstName NVARCHAR(50),
@LastName NVARCHAR(50),
@Email VARCHAR(100),
@Phone VARCHAR(15),
@Password VARCHAR(50),
@Birthday DATE,
@Gender NVARCHAR(10)
AS
BEGIN
SET NOCOUNT ON;
-- 1. Kiem tra trung khoa chinh (UserID)
\begin{lstlisting}
IF EXISTS (SELECT 1 FROM [USER] WHERE UserID = @UserID)
BEGIN
    ;THROW 50001, N'Loi: UserID nay da ton tai trong he thong.', 1;
    RETURN;
END
-- 2. Kiem tra trung Email
IF EXISTS (SELECT 1 FROM [USER] WHERE Email = @Email)
BEGIN
    ;THROW 50002, N'Loi: Email nay da duoc su dung boi tai khoan khac.', 1;
    RETURN;
END
-- 3. Validate So dien thoai (Phai la so va dai 10 ky tu)
IF (LEN(@Phone) <> 10 OR ISNUMERIC(@Phone) = 0)
BEGIN
    ;THROW 50003, N'Loi: So dien thoai khong hop le (Phai bao gom dung 10 chu so).
    ', 1;
    RETURN;
END

-- 4. Validate Mat khau (>= 8 ky tu, co chu hoa, so, ky tu dac biet)
IF (LEN(@Password) < 8
    OR @Password COLLATE Latin1_General_BIN NOT LIKE '%[A-Z]%'
    OR @Password COLLATE Latin1_General_BIN NOT LIKE '%[0-9]%'
    OR @Password COLLATE Latin1_General_BIN NOT LIKE '%[@$#%!&]%')
BEGIN
    ;THROW 50004, N'Loi: Mat khau yeu. Phai co it nhat 8 ky tu, bao gom chu hoa,
    so va ky tu dac biet (@$#%!&).', 1;
    RETURN;
END

-- 5. Validate Tuoi (Khach hang > 13 tuoi va ngay sinh < thoi diem hien tai)
IF (@Birthday >= GETDATE())
BEGIN
    ;THROW 50005, N'Loi: Ngay sinh khong duoc lon hon ngay hien tai.', 1;
    RETURN;
END

-- 6. Validate Gioi tinh
IF (@Gender NOT IN ('Male', 'Female'))
BEGIN
```

```
    ;THROW 50007, N'Loi: Gioi tinh khong hop le (Chi chap nhan "Male" hoac "Female
    ").', 1;
    RETURN;
END

-- Neu tat ca hop le, thuc hien INSERT
INSERT INTO [USER] (UserID, FirstName, LastName, Email, Phone, Password, Birthday,
     Gender, CreatedAt)
VALUES (@UserID, @FirstName, @LastName, @Email, @Phone, @Password, @Birthday,
    @Gender, GETDATE());

PRINT N'Them nguoi dung moi thanh cong!';

END;
```

Giải thích thủ tục:

`sp_InsertUser` là thủ tục dùng để thêm người dùng mới vào bảng `USER`.

Các kiểm tra đầu tiên đảm bảo rằng `UserID` và `Email` không bị trùng lặp trong hệ thống, đảm bảo tính duy nhất của các trường này.

Thủ tục cũng kiểm tra tính hợp lệ của số điện thoại (phải là số và dài 10 ký tự), mật khẩu (ít nhất 8 ký tự, có chữ hoa, số, và ký tự đặc biệt), tuổi của người dùng (ngày sinh không được lớn hơn ngày hiện tại), và giới tính (phải là "Male" hoặc "Female").

Nếu tất cả các điều kiện đều hợp lệ, thủ tục thực hiện thêm thông tin người dùng vào bảng `USER`.

Testing:

```
-- Test 1: Them user voi mat khau yeu (Se bao loi 50004)
EXEC sp_InsertUser 'US_TEST', N'Test', N'User', 'test@mail.com', '0901234567', '
    12345', '2000-01-01', 'Male';

-- Test 2: Them user hop le
EXEC sp_InsertUser 'US_TEST', N'Test', N'User', 'test@mail.com', '0901234567', '
    Pass@123', '2000-01-01', 'Male';
```

Kết quả:

```
Msg 50004, Level 16, State 1, Procedure sp_InsertUser, Line 41
Lỗi: Mật khẩu yếu. Phải có ít nhất 8 ký tự, bao gồm chữ hoa, số và ký tự đặc biệt (@$#!&).
```

Hình 3: Kết quả test 1

Thêm người dùng mới thành công!

	UserID	FirstNa...	LastName	Email	Phone	Password	Birthday	Gender	CreatedAt
1	US_TEST	Test	User	test@mail.com	0901234567	Pass@123	2000-01-01	Male	2025-11-27 16:28:40.423
2	US001	Nguyen	Van A	vana@gmail.com	0901234567	PassWord@1	1995-01-01	Male	2025-11-26 16:20:47.927
3	US002	Tran	Thi B	thib@gmail.com	0902345678	Secure#2024	1998-05-15	Female	2025-11-26 16:20:47.927
4	US003	Le	Van C	vanc@yahoo.com	0903456789	MyPass!123	2000-10-20	Male	2025-11-26 16:20:47.927

Hình 4: Kết quả test 2 và bảng USER sau khi thêm user thành công

6.1.2 Thủ tục `sp_UpdateUser`

```sql
CREATE PROCEDURE sp_UpdateUser
@UserID VARCHAR(10),
@NewEmail VARCHAR(100),
@NewPhone VARCHAR(15),
@NewPassword VARCHAR(50)
AS
BEGIN
SET NOCOUNT ON;
-- 1. Kiem tra UserID co ton tai khong
IF NOT EXISTS (SELECT 1 FROM [USER] WHERE UserID = @UserID)
BEGIN
    ;THROW 50008, N'Loi: Khong tim thay UserID can cap nhat.', 1;
    RETURN;
END

-- 2. Validate Email moi: Khong duoc trung voi user khac (nhung duoc trung voi
    chinh minh)
IF EXISTS (SELECT 1 FROM [USER] WHERE Email = @NewEmail AND UserID <> @UserID)
BEGIN
    ;THROW 50009, N'Loi: Email moi da thuoc ve mot tai khoan khac.', 1;
    RETURN;
END

-- 3. Validate Phone moi
IF (LEN(@NewPhone) <> 10 OR ISNUMERIC(@NewPhone) = 0)
BEGIN
    ;THROW 50010, N'Loi: So dien thoai moi khong hop le.', 1;
    RETURN;
END

-- 4. Validate Password moi
IF (LEN(@NewPassword) < 8
    OR @NewPassword COLLATE Latin1_General_BIN NOT LIKE '%[A-Z]%'
    OR @NewPassword COLLATE Latin1_General_BIN NOT LIKE '%[0-9]%'
    OR @NewPassword COLLATE Latin1_General_BIN NOT LIKE '%[@$#%!&]%')
BEGIN
    ;THROW 50011, N'Loi: Mat khau moi khong du manh.', 1;
    RETURN;
END

-- Thuc hien UPDATE
UPDATE [USER]
SET Email = @NewEmail,
    Phone = @NewPhone,
    Password = @NewPassword
WHERE UserID = @UserID;

PRINT N'Cap nhat thong tin nguoi dung thanh cong!';

END;
```

Giải thích thủ tục:

`sp_UpdateUser` là thủ tục dùng để cập nhật thông tin người dùng.

Thủ tục kiểm tra sự tồn tại của `UserID`, đảm bảo rằng người dùng cần cập nhật phải tồn tại trong cơ sở dữ liệu.

Sau đó, thủ tục kiểm tra tính hợp lệ của `Email` (không được trùng với tài khoản khác), `Phone` (phải là số và đủ 10 chữ số) và `Password` (phải đủ mạnh, bao gồm chữ hoa, số và ký tự đặc biệt).

Nếu tất cả các kiểm tra hợp lệ, thủ tục thực hiện cập nhật thông tin của người dùng trong bảng USER.

Testing:

```
-- Test 3: Cap nhat thong tin cua userID khong ton tai (That bai)
EXEC sp_UpdateUser 'US_100', 'abc@gmail.com', '0901122334', 'StrongPass1@';

-- Test 4: Cap nhat thong tin hop le
EXEC sp_UpdateUser 'US_TEST', 'newemail@mail.com', '0908765432', 'NewPass@123';
```

Kết quả:

```
Msg 50008, Level 16, State 1, Procedure sp_UpdateUser, Line 13
Lỗi: Không tìm thấy UserID cần cập nhật.
```

Hình 5: Kết quả test 3

Cập nhật thông tin người dùng thành công!

	UserID	FirstNa...	LastName	Email	Phone	Password	Birthday	Gender	CreatedAt
1	US_TEST	Test	User	newemail@mail.com	0908765432	NewPass@123	2000-01-01	Male	2025-11-27 16:28:40.423

Hình 6: Kết quả test 4 và UserID: "US_TEST" sau khi cập nhật thành công

6.1.3 Thủ tục sp_DeleteUser

```
CREATE PROCEDURE sp_DeleteUser
@UserID VARCHAR(10)
AS
BEGIN
SET NOCOUNT ON;
-- 1. Kiem tra UserID co ton tai khong
IF NOT EXISTS (SELECT 1 FROM [USER] WHERE UserID = @UserID)
BEGIN
    ;THROW 50012, N'Loi: UserID khong ton tai de xoa.', 1;
    RETURN;
END

-- 2. Kiem tra rang buoc du lieu (Neu User da tung dat ve)
-- Neu User da tung dat ve (ton tai trong bang BOOKING), KHONG DUOC XOA.
IF EXISTS (SELECT 1 FROM BOOKING WHERE UserID = @UserID)
BEGIN
    ;THROW 50013, N'Loi: Khong the xoa nguoi dung nay vi ho da co lich su giao
    dich (Booking). Hay vo hieu hoa tai khoan nay thay vi xoa.', 1;
    RETURN;
END

-- Neu thoa man dieu kien (chua tung mua ve), thuc hien DELETE
DELETE FROM [USER]
WHERE UserID = @UserID;

PRINT N'Xoa nguoi dung thanh cong!';

END;
```

Giải thích thủ tục:

`sp_DeleteUser` là thủ tục dùng để xóa người dùng khỏi bảng `USER`.

Thủ tục đầu tiên kiểm tra sự tồn tại của `UserID`.

Nếu người dùng đã từng thực hiện giao dịch (có lịch sử trong bảng `BOOKING`), thủ tục không cho phép xóa và xuất ra thông báo lỗi.

Nếu người dùng chưa thực hiện giao dịch nào, thủ tục thực hiện xóa người dùng khỏi bảng `USER`.

Testing:

```sql
-- Test 5: Xoa user US001 (That bai vi US001 da co Booking trong du lieu mau truoc do)
EXEC sp_DeleteUser 'US001';

-- Test 6: Xoa user vua tao (Thanh cong vi chua mua ve)
EXEC sp_DeleteUser 'US_TEST';
```

Kết quả:

```
Msg 50013, Level 16, State 1, Procedure sp_DeleteUser, Line 18
Lỗi: Không thể xóa người dùng này vì họ đã có lịch sử giao dịch (Booking). Hãy vô hiệu hóa tài khoản thay vì xóa.
```

Hình 7: Kết quả test 5

Xóa người dùng thành công!

	UserID	FirstNa...	LastName	Email	Phone	Password	Birthday	Gender	CreatedAt
1	US001	Nguyen	Van A	vana@gmail.com	0901234567	PassWord@1	1995-01-01	Male	2025-11-26 16:20:47.927
2	US002	Tran	Thi B	thib@gmail.com	0902345678	Secure#2024	1998-05-15	Female	2025-11-26 16:20:47.927
3	US003	Le	Van C	vanc@yahoo.com	0903456789	MyPass!123	2000-10-20	Male	2025-11-26 16:20:47.927

Hình 8: Kết quả test 6 và bảng USER sau khi xóa user thành công

Mục đích và Tại sao cần kiểm tra?

Mục đích của thủ tục: Các thủ tục này nhằm đảm bảo tính toàn vẹn của dữ liệu trong bảng `USER`, đồng thời đảm bảo rằng các thông tin người dùng được nhập vào hệ thống phải hợp lệ và đáp ứng các yêu cầu bảo mật như mật khẩu mạnh và thông tin liên lạc chính xác.

Kiểm tra dữ liệu: Việc kiểm tra dữ liệu nhằm phát hiện sớm các lỗi nhập liệu, tránh việc nhập vào các thông tin sai hoặc không hợp lệ, như `Email` trùng lặp, số điện thoại sai định dạng, hoặc mật khẩu yếu. Các kiểm tra này cũng giúp bảo vệ tính bảo mật và bảo vệ quyền lợi của người dùng trong hệ thống.

Lý do cần xóa dữ liệu: Thủ tục xóa người dùng chỉ thực hiện khi người dùng không có giao dịch lịch sử. Việc này nhằm đảm bảo tính toàn vẹn của dữ liệu và không làm mất các thông tin liên quan đến các giao dịch đã thực hiện.

6.2 Viết các Trigger

6.2.1 Trigger kiểm tra ngày khởi chiếu hợp lệ

- **1. Ràng buộc nghiệp vụ:** Ngày bắt đầu suất chiếu (`StartTime`) phải diễn ra sau hoặc cùng ngày với ngày khởi chiếu chính thức (`ReleaseDate`) của bộ phim.
- **2. Thao tác DML vi phạm:** `INSERT`, `UPDATE` trên bảng `SHOWTIME`.
- **3. Mã lệnh Trigger:**

```
CREATE TRIGGER trg_Check_ReleaseDate
ON SHOWTIME
AFTER INSERT, UPDATE
AS
BEGIN
    IF EXISTS (
        SELECT 1 FROM INSERTED i
        JOIN MOVIE m ON i.Movie_ID = m.Movie_ID
        WHERE m.ReleaseDate > i.StartTime
    )
    BEGIN
        RAISERROR (N'Loi: Ngay khoi chieu phim phai som hon thoi gian bat dau suat chieu.', 16, 1);
        ROLLBACK TRANSACTION;
    END
END;
GO
```

- **4. Dữ liệu minh họa (Test Case):**

```
-- Phim MV001 cong chieu 2026. Tao suat chieu nam 2025 -> LOI
INSERT INTO SHOWTIME (Showtime_ID, Movie_ID, Room_ID, BasePrice, StartTime, EndTime)
VALUES ('ST_TEST_1', 'MV001', 'R001', 100000, '2025-12-31 18:00', '2025-12-31 20:00');
```

```
Msg 50000, Level 16, State 1, Procedure trg_Check_Showtime_Duration_Valid, Line 16
Loi: Thoi gian suat chieu (EndTime - StartTime) ngan hon thoi luong goc cua bo phim.
Msg 3609, Level 16, State 1, Line 1
The transaction ended in the trigger. The batch has been aborted.
```

6.2.2 Trigger chống trùng lịch chiếu (Overlap)

- **1. Ràng buộc nghiệp vụ:** Trong cùng một phòng chiếu (`Room_ID`), không được phép có hai suất chiếu chồng chéo thời gian lên nhau.
- **2. Thao tác DML vi phạm:** `INSERT`, `UPDATE` trên bảng `SHOWTIME`.
- **3. Mã lệnh Trigger:**

```sql
CREATE TRIGGER trg_Check_Showtime_Overlap
ON SHOWTIME
AFTER INSERT, UPDATE
AS
BEGIN
    IF EXISTS (
        SELECT 1 FROM INSERTED i
        JOIN SHOWTIME s ON i.Room_ID = s.Room_ID
        WHERE i.Showtime_ID <> s.Showtime_ID -- Khong so sanh voi chinh no
          AND (
              (i.StartTime >= s.StartTime AND i.StartTime < s.EndTime)
              OR (i.EndTime > s.StartTime AND i.EndTime <= s.EndTime)
              OR (i.StartTime <= s.StartTime AND i.EndTime >= s.EndTime)
          )
    )
    BEGIN
        RAISERROR (N'Loi: Suat chieu bi trung thoi gian trong cung mot phong.', 16, 1);
        ROLLBACK TRANSACTION;
    END
END;
GO
```

- **4. Dữ liệu minh họa (Test Case):**

```sql
-- Tao suat chieu 19:00 - 21:00 chong len suat 18:00 - 20:00 da co -> LOI
INSERT INTO SHOWTIME (Showtime_ID, Movie_ID, Room_ID, BasePrice, StartTime, EndTime)
VALUES ('ST_ERR', 'MV001', 'R001', 100000, '2024-02-14 19:00', '2024-02-14 21:00');
```

```
Msg 50000, Level 16, State 1, Procedure trg_Check_Showtime_Overlap, Line 22
Loi: Suat chieu bi trung thoi gian trong cung mot phong.
Msg 3609, Level 16, State 1, Line 2
The transaction ended in the trigger. The batch has been aborted.
```

6.2.3 Trigger kiểm tra thời gian đặt vé

- **1. Ràng buộc nghiệp vụ:** Thời gian đặt vé (`BookingTime`) phải sớm hơn thời gian bắt đầu suất chiếu.
- **2. Thao tác DML vi phạm:** `INSERT`, `UPDATE` trên bảng `INCLUDE_SEAT`.
- **3. Mã lệnh Trigger:**

```sql
CREATE TRIGGER trg_Check_BookingTime
ON INCLUDE_SEAT
AFTER INSERT, UPDATE
AS
BEGIN
    IF EXISTS (
        SELECT 1 FROM INSERTED i
        JOIN BOOKING b ON i.Booking_ID = b.Booking_ID
        JOIN SHOWTIME s ON i.Showtime_ID = s.Showtime_ID
```

```
        WHERE b.BookingTime >= s.StartTime
    )
    BEGIN
        RAISERROR (N'Loi: Thoi gian dat ve phai som hon suat chieu.', 16, 1);
        ROLLBACK TRANSACTION;
    END
END;
GO
```

- **4. Ví dụ minh họa:**

```
INSERT INTO BOOKING(BOOKING_ID, USERID,BOOKINGTIME) VALUES ('BK_TEST_1','
    US001','2025-11-28 18:00:00');
INSERT INTO INCLUDE_SEAT(ROOM_ID,SEATINFO,SHOWTIME_ID,BOOKING_ID,SEATSTATUS)
    VALUES ('R010','B2','SH019','BK_TEST_1','Booked');
```

```
Msg 50000, Level 16, State 1, Procedure trg_Check_BookingTime, Line 16
Loi: Thoi gian dat ve phai som hon thoi gian bat dau suat chieu.
Msg 3609, Level 16, State 1, Line 1
The transaction ended in the trigger. The batch has been aborted.
```

6.2.4 Trigger kiểm tra thời gian thanh toán

- **1. Ràng buộc nghiệp vụ:** Thời gian thanh toán phải diễn ra sau khi đã tạo đơn đặt vé.
- **2. Thao tác DML vi phạm:** `INSERT`, `UPDATE` trên bảng `PAYMENT`.
- **3. Mã lệnh Trigger:**

```
CREATE TRIGGER trg_Check_PaymentTime
ON BOOKING
AFTER INSERT, UPDATE
AS
BEGIN
    IF EXISTS (
        SELECT 1
        FROM INSERTED p
        JOIN PAYMENT b ON b.Payment_ID = p.PaymentID
        WHERE p.BookingTime >= b.PaymentTime
    )
    BEGIN
        RAISERROR ('Loi: Thoi gian thanh toan phai sau thoi gian dat ve.',
    16, 1);
        ROLLBACK TRANSACTION;
    END
END;
GO
```

- **4. Ví dụ minh họa:**

```
INSERT INTO BOOKING(BOOKING_ID, USERID, PaymentID,BOOKINGTIME) VALUES ('
    BK_TEST_1','US001','PMT_TEST_1','2025-11-28 18:00:00');
INSERT INTO PAYMENT(Payment_ID, PaymentTime) VALUES ('PMT_TEST_1','2025-11-28
     17:00:00');
```

```
Msg 50000, Level 16, State 1, Procedure trg_Check_PaymentTime, Line 14
Loi: Thoi gian thanh toan phai sau thoi gian dat ve.
Msg 3609, Level 16, State 1, Line 1
The transaction ended in the trigger. The batch has been aborted.
```

6.2.5 Trigger kiểm tra hạn thanh toán trong 24h

- **1. Ràng buộc nghiệp vụ:** Thanh toán phải thực hiện trong vòng 24 giờ kể từ khi đặt vé.
- **2. Thao tác DML vi phạm:** `INSERT`, `UPDATE` trên bảng `PAYMENT`.
- **3. Mã lệnh Trigger:**

```
CREATE TRIGGER trg_Check_Payment_Within_24Hours
ON BOOKING
AFTER INSERT, UPDATE
AS
BEGIN
    IF EXISTS (
        SELECT 1
        FROM INSERTED p
        JOIN PAYMENT b ON p.PaymentID = b.Payment_ID
        WHERE DATEDIFF(HOUR, p.BookingTime, b.PaymentTime) > 24
    )
    BEGIN
        RAISERROR ('Loi: Thoi gian thanh toan phai trong vong 24 gio ke tu
    thoi gian dat ve.', 16, 1);
        ROLLBACK TRANSACTION;
    END
END;
GO
```

- **4. Ví dụ minh họa:**

```
INSERT INTO PAYMENT (Payment_ID, PaymentTime)
VALUES ('PM_ERR', '2025-01-10 09:00:00');
INSERT INTO BOOKING (Booking_ID, UserID, PaymentID, BookingTime)
VALUES ('BK_ERR', 'US001', 'PM_ERR', '2025-01-08 10:00:00');
```

```
Msg 50000, Level 16, State 1, Procedure trg_Check_Payment_Within_24Hours,
    Line 13
Loi: Thoi gian thanh toan phai trong vong 24 gio ke tu thoi gian dat ve.
Msg 3609, Level 16, State 1, Line 1
The transaction ended in the trigger. The batch has been aborted.
```

6.2.6 Trigger kiểm tra thời lượng suất chiếu

- **1. Ràng buộc nghiệp vụ:** Thời gian chiếu (`EndTime - StartTime`) phải đủ dài để chiếu hết bộ phim.
- **2. Thao tác DML vi phạm:** `INSERT`, `UPDATE` trên bảng `SHOWTIME`.

- **3. Mã lệnh Trigger:**

```
CREATE TRIGGER trg_Check_Showtime_Duration_Valid
ON SHOWTIME
AFTER INSERT, UPDATE
AS
BEGIN
    IF EXISTS (
        SELECT 1 FROM INSERTED i
        JOIN MOVIE m ON i.Movie_ID = m.Movie_ID
        WHERE DATEDIFF(MINUTE, i.StartTime, i.EndTime) < m.Duration
    )
    BEGIN
        RAISERROR (N'Loi: Thoi gian suat chieu ngan hon phim.', 16, 1);
        ROLLBACK TRANSACTION;
    END
END;
GO
```

- **4. Ví dụ minh họa:**

```
INSERT INTO SHOWTIME (Showtime_ID, Movie_ID, Room_ID, BasePrice, StartTime,
    EndTime)
VALUES ('ST_TEST_2', 'MV001', 'R001', 100000, '2026-01-01 18:00', '2026-01-01
     19:00');
```

```
Msg 50000, Level 16, State 1, Procedure trg_Check_Showtime_Duration_Valid,
     Line 16
Loi: Thoi gian suat chieu (EndTime - StartTime) ngan hon thoi luong goc
    cua bo phim.
Msg 3609, Level 16, State 1, Line 1
The transaction ended in the trigger. The batch has been aborted.
```

6.2.7 Trigger tính thuộc tính dẫn xuất TotalPrice trong BOOKING

- **1. Chọn thuộc tính dẫn xuất:** `TotalPrice` (Tổng tiền cuối cùng) trong bảng `BOOKING`.
  Công thức tính:

$$TotalPrice = (\sum TienVe + \sum TienFB) - Voucher$$

- **2. Các thao tác DML làm thay đổi giá trị:** Giá trị `TotalPrice` phụ thuộc vào dữ liệu của 3 bảng khác nhau, do đó cần bắt sự kiện trên cả 3 bảng:
  - Bảng `INCLUDE_SEAT`: `INSERT`, `UPDATE`, `DELETE` (Thay đổi số lượng ghế).
  - Bảng `F_B`: `INSERT`, `UPDATE`, `DELETE` (Thay đổi dịch vụ ăn uống).
  - Bảng `APPLY_FOR`: `INSERT`, `DELETE` (Áp dụng hoặc gỡ bỏ mã giảm giá).
- **3. Mã lệnh (Stored Procedure & Trigger):**
  Giải pháp: Do yêu cầu phải tính tổng tiền hàng (Vé + F&B) trước thì mới tính được mức giảm giá của Voucher (ví dụ giảm 10% trên tổng đơn), nên nhóm sử dụng một **Stored Procedure** trung tâm để xử lý tuần tự logic này, sau đó các Trigger sẽ gọi Procedure đó.
  A. Stored Procedure (Bộ xử lý trung tâm):

```sql
CREATE PROCEDURE sp_Calculate_Booking_Total
    @BookingID VARCHAR(10)
AS
BEGIN
    SET NOCOUNT ON;
    DECLARE @TotalTicket DECIMAL(18, 2) = 0;
    DECLARE @TotalFB DECIMAL(18, 2) = 0;
    DECLARE @Discount DECIMAL(18, 2) = 0;
    DECLARE @FinalPrice DECIMAL(18, 2) = 0;

    -- BUOC 1: Tinh tong tien ve (Cong don tung ve trong INCLUDE_SEAT)
    SELECT @TotalTicket = ISNULL(SUM(ST.BasePrice), 0)
    FROM INCLUDE_SEAT INS JOIN SHOWTIME ST ON INS.Showtime_ID = ST.
    Showtime_ID
    WHERE INS.Booking_ID = @BookingID;

    -- BUOC 2: Tinh tong tien F&B (Cong don tung mon trong F_B)
    SELECT @TotalFB = ISNULL(SUM(Price), 0) FROM F_B WHERE Booking_ID =
    @BookingID;

    -- Tong tam tinh (Subtotal) truoc khi giam gia
    DECLARE @SubTotal DECIMAL(18, 2) = @TotalTicket + @TotalFB;

    -- BUOC 3: Xu ly Voucher (Kiem tra logic Apply Voucher)
    IF EXISTS (SELECT 1 FROM APPLY_FOR WHERE Booking_ID = @BookingID)
    BEGIN
        DECLARE @DiscountType NVARCHAR(20), @Percentage FLOAT, @FixedAmount
    DECIMAL(18, 2), @EndDate DATETIME;
        SELECT TOP 1
            @DiscountType = V.DiscountType,
            @Percentage = V.Percentage,
            @FixedAmount = V.FixedAmount,
            @EndDate = V.EndDate
        FROM APPLY_FOR AF JOIN VOUCHER V ON AF.Promo_Code = V.Promo_Code
        WHERE AF.Booking_ID = @BookingID;

        IF GETDATE() <= @EndDate
        BEGIN
            IF @DiscountType = 'Percent' SET @Discount = @SubTotal * (
    @Percentage / 100.0);
            ELSE IF @DiscountType = 'Fixed' SET @Discount = @FixedAmount;
        END
    END

    -- BUOC 4: Chot gia cuoi cung
    SET @FinalPrice = @SubTotal - @Discount;
    IF @FinalPrice < 0 SET @FinalPrice = 0;

    UPDATE BOOKING SET TotalPrice = @FinalPrice WHERE Booking_ID = @BookingID
    ;
END;
GO
```

B. Các Trigger kích hoạt (Event Listeners):

Trigger 1: Cập nhật khi thay đổi Ghế (INCLUDE_SEAT)

- **Mục đích:** Tự động tính toán và cập nhật lại tổng tiền (TotalPrice) trong bảng Booking bất cứ khi nào có thay đổi liên quan đến danh sách ghế ngồi danh sách ghế ngồi của đơn đặt vé.

```sql
CREATE TRIGGER trg_Update_Total_On_Seat
ON INCLUDE_SEAT
AFTER INSERT, DELETE, UPDATE
AS
BEGIN
    SET NOCOUNT ON;
    DECLARE @BookingList TABLE (ID VARCHAR(10));
    INSERT INTO @BookingList SELECT Booking_ID FROM Inserted UNION SELECT
    Booking_ID FROM Deleted;

    DECLARE @BookingID VARCHAR(10);
    DECLARE cur CURSOR FOR SELECT ID FROM @BookingList;
    OPEN cur; FETCH NEXT FROM cur INTO @BookingID;
    WHILE @@FETCH_STATUS = 0 BEGIN
        EXEC sp_Calculate_Booking_Total @BookingID;
        FETCH NEXT FROM cur INTO @BookingID;
    END
    CLOSE cur; DEALLOCATE cur;
END;
GO
```

Trigger 2: Cập nhật khi thay đổi F&B (F_B)

- **Mục đích:** Tự động tính toán và cập nhật lại tổng tiền (TotalPrice) trong bảng Booking bất cứ khi nào có thay đổi liên quan đến dịch vụ ăn uống (F&B) của đơn đặt vé.

```sql
CREATE TRIGGER trg_Update_Total_On_FB
ON F_B
AFTER INSERT, DELETE, UPDATE
AS
BEGIN
    SET NOCOUNT ON;
    DECLARE @BookingList TABLE (ID VARCHAR(10));
    INSERT INTO @BookingList SELECT Booking_ID FROM Inserted UNION SELECT
    Booking_ID FROM Deleted;

    DECLARE @BookingID VARCHAR(10);
    DECLARE cur CURSOR FOR SELECT ID FROM @BookingList;
    OPEN cur; FETCH NEXT FROM cur INTO @BookingID;
    WHILE @@FETCH_STATUS = 0 BEGIN
        EXEC sp_Calculate_Booking_Total @BookingID;
        FETCH NEXT FROM cur INTO @BookingID;
    END
    CLOSE cur; DEALLOCATE cur;
END;
GO
```

Trigger 3: Cập nhật khi áp dụng Voucher (APPLY_FOR)

- **Mục đích:** Tự động tính toán và cập nhật lại tổng tiền (TotalPrice) trong bảng Booking bất cứ khi nào có thay đổi liên quan đến việc áp dụng hoặc gỡ bỏ mã giảm giá (Voucher) cho đơn đặt vé.

```sql
CREATE TRIGGER trg_Update_Total_On_Voucher
ON APPLY_FOR
AFTER INSERT, DELETE
AS
BEGIN
    SET NOCOUNT ON;
```

```
    DECLARE @BookingList TABLE (ID VARCHAR(10));
    INSERT INTO @BookingList SELECT Booking_ID FROM Inserted UNION SELECT
    Booking_ID FROM Deleted;

    DECLARE @BookingID VARCHAR(10);
    DECLARE cur CURSOR FOR SELECT ID FROM @BookingList;
    OPEN cur; FETCH NEXT FROM cur INTO @BookingID;
    WHILE @@FETCH_STATUS = 0 BEGIN
        EXEC sp_Calculate_Booking_Total @BookingID;
        FETCH NEXT FROM cur INTO @BookingID;
    END
    CLOSE cur; DEALLOCATE cur;
END;
GO
```

- **4. Dữ liệu minh họa (Test Case):**

```
-- 1. Tao mot ghe gia trong phong R001 de test
IF NOT EXISTS (SELECT * FROM SEAT WHERE Room_ID = 'R001' AND SeatInfo = 'T1')
    INSERT INTO SEAT (Room_ID, SeatInfo, SeatType) VALUES ('R001', 'T1', '
    Standard');

IF NOT EXISTS (SELECT * FROM SEAT WHERE Room_ID = 'R001' AND SeatInfo = 'T2')
    INSERT INTO SEAT (Room_ID, SeatInfo, SeatType) VALUES ('R001', 'T2', '
    Standard');

-- 2. Tao mot Voucher test (Giam 10000)
IF NOT EXISTS (SELECT * FROM VOUCHER WHERE Promo_Code = 'TEST_10PERCENT')
    INSERT INTO VOUCHER (Promo_Code, StartDate, EndDate, DiscountType,
    FixedAmount)
    VALUES ('TEST_10PERCENT', '2020-01-01', '2030-12-31', 'Fixed', 10000);
```

```
-- Tao moi
INSERT INTO BOOKING (Booking_ID, UserID, TotalPrice, BookingTime)
VALUES ('BK_TEST', 'US001', 0, '2024-02-14 14:00:00');
-- Them ghe 1
INSERT INTO INCLUDE_SEAT (Room_ID, SeatInfo, Showtime_ID, Booking_ID,
    SeatStatus)
VALUES ('R001', 'T1', 'SH001', 'BK_TEST', 'Booked');

-- Them ghe 2
INSERT INTO INCLUDE_SEAT (Room_ID, SeatInfo, Showtime_ID, Booking_ID,
    SeatStatus)
VALUES ('R001', 'T2', 'SH001', 'BK_TEST', 'Booked');

-- Kiem tra gia (Ky vong: 2 ve * 100k = 200,000)
SELECT Booking_ID, TotalPrice AS [Gia Buoc 2 (200k)] FROM BOOKING WHERE
    Booking_ID = 'BK_TEST';
```

	Booking...	Giá Bướ...
1	BK_TEST	200000.00

Hình 9: Kết quả sau bước 2: TotalPrice = 200,000

```
-- Them 1 Combo gia 50,000
INSERT INTO F_B (FB_ID, Booking_ID, FBName, Price)
```

```
VALUES ('TEST_FB', 'BK_TEST', N'Combo Test', 50000);

-- Kiem tra gia (Ky vong: 200k ve + 50k nuoc = 250,000)
SELECT Booking_ID, TotalPrice AS [Gia Buoc 3 (250k)] FROM BOOKING WHERE
    Booking_ID = 'BK_TEST';
```

	Booking...	Giá Bướ...
1	BK_TEST	250000.00

Hình 10: Kết quả sau bước 3: TotalPrice = 250,000

```
-- Ap dung ma giam 10000
INSERT INTO APPLY_FOR (Promo_Code, Booking_ID)
VALUES ('TEST_10PERCENT', 'BK_TEST');

-- Kiem tra gia (Ky vong: 250k - 10000 = 240,000)
SELECT Booking_ID, TotalPrice AS [Gia Buoc 4 (240k)] FROM BOOKING WHERE
    Booking_ID = 'BK_TEST';
```

	Booking...	Giá Bướ...
1	BK_TEST	240000.00

Hình 11: Kết quả sau bước 4: TotalPrice = 240,000

6.3 Thủ tục truy vấn

6.3.1 Thủ tục sp_ReportUserSpendingByDate

```
CREATE PROCEDURE sp_ReportUserSpendingByDate
    @StartDate DATETIME,
    @EndDate DATETIME,
    @MinSpending DECIMAL(18, 2)
AS
BEGIN
    SET NOCOUNT ON;

    IF @StartDate IS NULL OR @EndDate IS NULL
    BEGIN
        ;THROW 51005, N'Loi: Cac tham so (@StartDate, @EndDate) la bat buoc.', 1;
    END

    IF @StartDate > @EndDate OR @StartDate > GETDATE() OR @EndDate > GETDATE()
    BEGIN
        ;THROW 51002, N'Loi: Thoi gian tra cuu khong hop le.', 1;
    END

    IF @MinSpending < 0
    BEGIN
        ;THROW 51001, N'Loi: Muc chi tieu toi thieu (@MinSpending) khong hop le.',
     1;
```

```
    END

    PRINT N'Bat dau tao bao cao chi tieu...';

    SELECT
        U.UserID,
        U.FirstName + ' ' + U.LastName AS "Ho Ten",
        U.Phone,
        U.Email,
        COUNT(B.Booking_ID) AS "Tong So Don Hang",
        SUM(B.TotalPrice) AS "Tong Chi Tieu"
    FROM
        [USER] U
    JOIN
        BOOKING B ON U.UserID = B.UserID
    JOIN
        PAYMENT P ON B.PaymentID = P.Payment_ID
    WHERE
        (B.BookingTime BETWEEN @StartDate AND @EndDate)
        AND P.Status = 'Success'
    GROUP BY
        U.UserID, U.FirstName, U.LastName, U.Phone, U.Email
    HAVING
        SUM(B.TotalPrice) >= ISNULL(@MinSpending, 0)
    ORDER BY
        "Tong Chi Tieu" DESC;

    DECLARE @RowFound INT = @@ROWCOUNT;

    IF @RowFound = 0
    BEGIN
        PRINT N'Thong bao: Khong tim thay du lieu.';
    END
    ELSE
    BEGIN
        PRINT N'Thanh cong: Da tim thay ' + CAST(@RowFound AS NVARCHAR(10)) + N'
    khach hang.';
    END
END;
```

Giải thích thủ tục:

`sp_ReportUserSpendingByDate` là thủ tục dùng để lọc ra danh sách những khách hàng đã chi tiêu trong một khoảng thời gian nhất định, đồng thời đạt mức chi tiêu tối thiểu.

Thủ tục yêu cầu 3 thông tin:

- @StartDate, @EndDate: Khoảng thời gian cần thu thập.
- @MinSpending: Mức chi tiêu tối thiểu.

Thủ tục tiến hành kết nối các bảng USER, BOOKING, PAYMENT với điều kiện các đơn hàng phải thuộc trong khoảng thời gian cần thu thập. Từ đó kết quả trả về sẽ bao gồm UserID, Họ tên USER, Số điện thoại, Email và các thuộc tính dẫn xuất như Tổng số đơn hàng và Tổng chi tiêu.

Từ danh sách kết quả trả về, thủ tục tiến hành gộp nhóm theo thông tin cá nhân của khách hàng, đồng thời tính toán các thông tin về Tổng số đơn hàng và Tổng chi tiêu của khách hàng.

Sau khi đã thực hiện gộp nhóm và tính toán các thông số dẫn xuất, thủ tục tiến hành lọc các khách hàng có tổng chi tiêu thỏa yêu cầu về mức chi tối thiểu.

Cuối cùng, thủ tục trả về danh sách các kết quả thỏa mãn các bộ lọc, thông qua thao tác sắp xếp mức chi tiêu giảm dần để dễ dàng nhận biết các khách hàng "VIP" của hệ thống.

Dữ liệu trả về có thể được sử dụng để áp dụng các chương trình tri ân khách hàng đặc biệt dựa trên mức chi tiêu mà khách hàng đã thực hiện, đồng thời cũng là công cụ hữu hiệu đối với các cấp quản lý khi đưa ra các quyết định quan trọng của hệ thống,

Xử lý ngoại lệ

- Nếu người dùng không truyền tham số @StartDate, @EndDate, thủ tục trả về lỗi và không thực thi truy vấn.
- Nếu các tham số @StartDate, @EndDate không hợp lệ (Vd: @StartDate > @EndDate), thủ tục trả về lỗi và không thực thi truy vấn.
- Nếu tham số @MinSpending được truyền vào không hợp lệ (Vd: số ấm), thủ tục trả về lỗi và không thực thi truy vấn.

Testing

```
-- Test 1: Truy suat thong tin hop le
EXEC sp_ReportUserSpendingByDate
    @StartDate = '2024-01-01',
    @EndDate = '2024-12-31',
    @MinSpending = 200000;

-- Test 2: Truyen tham so khong hop le
EXEC sp_ReportUserSpendingByDate
    @StartDate = NULL,
    @EndDate = NULL,
    @MinSpending = NULL;

-- Test 3: Tham so @MinSpending khong hop le
EXEC sp_ReportUserSpendingByDate
    @StartDate = '2024-01-01',
    @EndDate = '2024-12-31',
    @MinSpending = -1;

-- Test 4: Tham so @MinSpending khong duoc truyen, gia tri mac dinh la 0
EXEC sp_ReportUserSpendingByDate
    @StartDate = '2024-01-01',
    @EndDate = '2024-12-31',
    @MinSpending = NULL;
```

Kết quả

	UserID	Ho Ten	Phone	Email	Tong So...	Tong Ch...
1	US003	Le Van C	0903456789	vanc@yahoo.com	1	404595.00
2	US001	Nguyen Van A	0901234567	vana@gmail.com	1	329000.00
3	US002	Tran Thi B	0902345678	thib@gmail.com	1	259000.00
4	US009	Do Van I	0909012345	vani@gmail.com	1	250000.00
5	US020	Huynh Xuan Quoc Viet	0919123456	svVIT@gmail.com	1	243512.00

```
Started executing query at  Line 68
Bắt đầu tạo báo cáo chi tiêu...
Thành công: Đã tìm thấy 5 khách hàng.
Total execution time: 00:00:00.053
```

Hình 12: Kết quả Test 1

```
Started executing query at  Line 73
Msg 51005, Level 16, State 1, Procedure sp_ReportUserSpendingByDate, Line 12
Lỗi: Các tham số (@StartDate, @EndDate) là bắt buộc.
Total execution time: 00:00:00.047
```

Hình 13: Kết quả Test 2

```
Started executing query at  Line 78
Msg 51001, Level 16, State 1, Procedure sp_ReportUserSpendingByDate, Line 22
Lỗi: Mức chi tiêu tối thiểu (@MinSpending) không hợp lệ.
Total execution time: 00:00:00.046
```

Hình 14: Kết quả Test 3

	UserID	Ho Ten	Phone	Email	Tong So...	Tong Ch...
1	US003	Le Van C	0903456789	vanc@yahoo.com	1	404595.00
2	US001	Nguyen Van A	0901234567	vana@gmail.com	1	329000.00
3	US002	Tran Thi B	0902345678	thib@gmail.com	1	259000.00
4	US009	Do Van I	0909012345	vani@gmail.com	1	250000.00
5	US020	Huynh Xuan Quoc Viet	0919123456	svVIT@gmail.com	1	243512.00
6	US006	Vo Thi F	0906789012	thif@gmail.com	1	159000.00
7	US008	Bui Thi H	0908901234	thih@gmail.com	1	149000.00
8	US007	Dang Van G	0907890123	vang@hotmail.com	1	145000.00
9	US005	Hoang Van E	0905678901	vane@gmail.com	1	140000.00
10	US010	Ngo Thi K	0910123456	thik@gmail.com	1	120000.00

```
Started executing query at  Line 83
Bắt đầu tạo báo cáo chi tiêu...
Thành công: Đã tìm thấy 10 khách hàng.
Total execution time: 00:00:00.061
```

Hình 15: Kết quả Test 4

6.3.2 Thủ tục sp_SearchShowtime

```
CREATE PROCEDURE sp_SearchShowtime
    @MovieTitle NVARCHAR(100),
    @SearchDate DATE
AS
BEGIN
    SET NOCOUNT ON;

    IF @MovieTitle IS NULL OR LTRIM(RTRIM(@MovieTitle)) = '' OR @SearchDate IS
    NULL
    BEGIN
        ;THROW 51005, N'Loi: Cac tham so (@MovieTitle, @SearchDate) la bat buoc.',
     1;
    END

    IF @SearchDate > GETDATE()
    BEGIN
        ;THROW 51003, N'Loi: Ngay tim kiem (@SearchDate) khong hop le.', 1;
    END

    PRINT N'Tu khoa phim: ' + @MovieTitle;
    PRINT N'Thoi gian tu ngay: ' + CONVERT(NVARCHAR, @SearchDate, 103);
    PRINT N'Dang tim kiem lich chieu...';

    SELECT
```

```
        M.Title AS "Ten Phim",
        C.Name AS "Ten Rap",
        R.RoomName AS "Phong Chieu",
        R.RoomType AS "Loai Phong",
        S.StartTime AS "Gio Chieu",
        S.BasePrice AS "Gia Ve Co Ban"
    FROM
        SHOWTIME S
    JOIN
        MOVIE M ON S.Movie_ID = M.Movie_ID
    JOIN
        ROOM R ON S.Room_ID = R.Room_ID
    JOIN
        CINEMA C ON R.CinemaID = C.CinemaID
    WHERE
        M.Title LIKE '%' + @MovieTitle + '%'
        AND CAST(S.StartTime AS DATE) >= @SearchDate
    ORDER BY
        M.Title ASC,
        S.StartTime ASC,
        C.Name ASC;

    DECLARE @RowFound INT = @@ROWCOUNT;

    IF @RowFound = 0
    BEGIN
        PRINT N'Thong bao: Khong tim thay suat chieu nao phu hop.';
    END
    ELSE
    BEGIN
        PRINT N'Thanh cong: Tim thay ' + CAST(@RowFound AS NVARCHAR(10)) + N' suat
     chieu.';
    END
END;
```

Giải thích thủ tục:

`sp_SearchShowtime` là thủ tục được thiết kế để tìm kiếm lịch chiếu phim trong hệ thống. Nó cho phép người dùng (hoặc ứng dụng) tìm các suất chiếu dựa trên tên phim (gần đúng) và ngày bắt đầu xem.

Thủ tục yêu cầu 2 thông tin:

- @MovieTitle: Tên phim cần tìm (hỗ trợ tiếng Việt có dấu)
- @SearchDate: Mốc thời gian tìm kiếm

Thủ tục tiến hành kết nối các bảng MOVIE, ROOM, CINEMA vào SHOWTIME. Từ đó, kết quả trả về sẽ bao gồm Tên phim, Tên rạp, Phòng chiếu, Loại phòng, Giờ chiếu và Giá vé.

Từ danh sách đã kết nối, thủ tục tiến hành các bộ lọc:

- Lọc tên phim: Thủ tục áp dụng phép tính gần đúng để tìm tất cả các phim có chứa tên phim được yêu cầu.
- Lọc thời gian tìm kiếm: Sau khi lọc theo tên phim, thủ tục tiếp tục lọc các suất chiếu từ ngày được yêu cầu trở về sau. Điều này giúp hệ thống gợi ý thêm cho người dùng nếu phim người dùng mong muốn không có suất chiếu trong ngày hôm đó.

Cuối cùng, thủ tục trả về danh sách các kết quả thỏa mãn các bộ lọc, thông qua thao tác sắp xếp với mức ưu tiên cao nhất cho tên phim, tiếp theo là thời gian suất chiếu và cuối cùng theo tên rạp.

Xử lý ngoại lệ

- Nếu người dùng không truyền tham số @MovieTitle, @SearchDate hoặc truyền dữ liệu rỗng, thủ tục trả về lỗi và không thực thi truy vấn.
- Nếu tham số @SearchDate không hợp lệ (Vd: @SearchDate > GETDATE()), thủ tục trả về lỗi và không thực thi truy vấn.

Testing

```
-- Test 1: Truy xuat thong tin hop le
EXEC sp_SearchShowtime @MovieTitle = N'Lat Mat', @SearchDate = '2024-01-01';

-- Test 2: Truyen tham so khong hop le
EXEC sp_SearchShowtime @MovieTitle = N'', @SearchDate = '';

-- Test 3: Thoi gian truy xuat khong hop le
EXEC sp_SearchShowtime @MovieTitle = N'Mai', @SearchDate = '2026-01-01';
```

Kết quả

	Ten Phim	Ten Rap	Phong ...	Loai Ph...	Gio Chieu	Gia Ve C...
1	Lật Mặt	CGV Vincom Đồng Khởi	Cinema 1	IMAX	2024-05-01 17:00:00.000	80000.00
2	Lật Mặt 2: Phim Trường	CGV Vincom Đồng Khởi	Cinema 2	3D	2024-05-02 19:00:00.000	85000.00
3	Lật Mặt 3: Ba Chàng Khuyết	CGV Landmark 81	Cinema 1	4DX	2024-05-03 15:00:00.000	90000.00
4	Lật Mặt 4: Nhà Có Khách	CGV Landmark 81	Cinema 2	2D	2024-05-04 20:00:00.000	95000.00
5	Lật Mặt 5: 48h	CGV SC VivoCity	Cinema 1	Gold Class	2024-05-05 18:00:00.000	100000.00
6	Lật Mặt 6: Tấm Vé Định Mệnh	CGV SC VivoCity	Cinema 2	3D	2024-05-06 21:00:00.000	105000.00
7	Lật Mặt 7: Một Điều Ước	CGV Liberty Citypoint	Cinema 1	VIP	2024-05-07 14:00:00.000	110000.00
8	Lật Mặt 8: Vòng Tay Nắng	CGV Liberty Citypoint	Cinema 2	2D	2025-05-01 16:00:00.000	115000.00

```
Started executing query at  Line 65
Từ khóa phim: Lật Mặt
Thời gian từ ngày: 01/01/2024
Đang tìm kiếm lịch chiếu...
Thành công: Tìm thấy 8 suất chiếu.
Total execution time: 00:00:00.075
```

Hình 16: Kết quả Test 1

```
Started executing query at  Line 67
Msg 51005, Level 16, State 1, Procedure sp_SearchShowtime, Line 12
Lỗi: Các tham số (@MovieTitle, @SearchDate) là bắt buộc.
Total execution time: 00:00:00.050
```

Hình 17: Kết quả Test 2

```
Started executing query at  Line 69
Msg 51003, Level 16, State 1, Procedure sp_SearchShowtime, Line 17
Lỗi: Ngày tìm kiếm (@SearchDate) không hợp lệ.
Total execution time: 00:00:00.050
```

Hình 18: Kết quả Test 3

6.3.3 Thủ tục sp_SearchUsersByBookingCount

```sql
CREATE PROCEDURE sp_SearchUsersByBookingCount
    @Keyword NVARCHAR(100),
    @MaxBookingCount INT
AS
BEGIN
    SET NOCOUNT ON;
    IF @MaxBookingCount < 0
    BEGIN
        ;THROW 51001, N'Loi: So luong don hang toi da khong duoc am.', 1;
    END

    SELECT
        U.UserID,
        ISNULL(U.FirstName, '') + ' ' + ISNULL(U.LastName, '') AS FullName,
        U.Email,
        U.Phone,
        COUNT(B.Booking_ID) AS TotalBookings,
        ISNULL(SUM(B.TotalPrice), 0) AS TotalSpent
    FROM
        [USER] U
    LEFT JOIN
        BOOKING B ON U.UserID = B.UserID
    LEFT JOIN
        PAYMENT P ON B.PaymentID = P.Payment_ID
    WHERE
        (@Keyword IS NULL OR U.Email LIKE '%' + @Keyword + '%' OR U.FirstName LIKE
     '%' + @Keyword + '%')
    GROUP BY
        U.UserID, U.FirstName, U.LastName, U.Email, U.Phone
    HAVING
        COUNT(B.Booking_ID) <= ISNULL(@MaxBookingCount, 999999)
    ORDER BY
        TotalBookings ASC, TotalSpent DESC;

    DECLARE @RowFound INT = @@ROWCOUNT;
    IF @RowFound = 0
```

```
        PRINT N'Khong tim thay khach hang nao.';
    ELSE
        PRINT N'Tim thay ' + CAST(@RowFound AS NVARCHAR(10)) + N' khach hang.';
END;
```

Giải thích thủ tục:

`sp_SearchUsersByBookingCount` là thủ tục được thiết kế truy xuất danh sách người dùng ít hoạt động. Nó giúp quản lý thu thập thông tin phục vụ các chiến dịch Marketing, kích cầu tiêu dùng và ưu đãi cho các khách hàng đã dần lãng quên hệ thống.

Thủ tục yêu cầu 2 thông tin:

- @Keyword: Từ khóa tìm kiếm theo tên (có thể để trống).
- @MaxBookingCount: Số lượng đơn hàng tối đa của một user.

Thủ tục tiến hành kết nối các bảng USER và BOOKING, PAYMENT. Từ đó, kết quả trả về sẽ bao gồm Thông tin cá nhân, Tổng số đơn hàng và tổng chi tiêu của từng cá nhân.

Từ danh sách đã kết nối, thủ tục tiến hành các bộ lọc:

- Lọc tên USER: Thủ tục áp dụng phép tính gần đúng để tìm tất cả các USER có tên gần giống với tham số truyền vào.
- Lọc theo tổng số đơn đã đặt: Thủ tục tiến hành gộp nhóm theo thông tin cá nhân và tính toán tổng số đơn hàng đã đặt, lọc bỏ theo số lượng đơn hàng tối đa từ tham số truyền vào.

Cuối cùng, thủ tục trả về danh sách các kết quả thỏa mãn các bộ lọc, thông qua thao tác sắp xếp tổng số đơn hàng tăng dần và tổng chi tiêu giảm dần.

Xử lý ngoại lệ

- Nếu người dùng không truyền tham số @Keyword, thủ tục sẽ bỏ qua không lọc và vẫn tiếp tục truy vấn.
- Nếu tham số @MaxBookingCount âm, thủ tục sẽ báo lỗi và ngừng truy vấn ngay lập tức.
- Nếu tham số @MaxBookingCount không được truyền vào, thủ tục sẽ sử dụng giá trị mặc định là 99999 và tiếp tục truy vấn.

Testing

```
-- Test 1: Truy suat thong tin hop le
EXEC sp_SearchUsersByBookingCount
    @Keyword="Le",
    @MaxBookingCount=10;
-- Test 2: Khong truyen tham so
EXEC sp_SearchUsersByBookingCount
    @Keyword=NULL,
    @MaxBookingCount=NULL
-- Test 3: Tham so @MaxBookingCount khong hop le
EXEC sp_SearchUsersByBookingCount
    @Keyword="V",
    @MaxBookingCount=-2
```

Kết quả

	UserID	FullName	Email	Phone	TotalBo...	TotalSp...
1	US022	Le Minh Khoa	svLMK@gmail.com	0919345678	0	0.00
2	US023	Le Minh Tri	svLMT@gmail.com	0919456789	0	0.00
3	US003	Le Van C	vanc@yahoo.com	0903456789	1	404595.00

```
Tìm thấy 3 khách hàng.
Total execution time: 00:00:00.083
```

Hình 19: Kết quả Test 1

	UserID	FullName	Email	Phone	TotalBo...	TotalSp...
1	US004	Pham Thi D	thid@outlook.com	0904567890	0	0.00
2	US011	Chu Van L	vanl@gmail.com	0911234567	0	0.00
3	US012	Ly Thi M	thim@gmail.com	0912345678	0	0.00
4	US013	Phan Van N	vann@gmail.com	0913456789	0	0.00
5	US014	Dinh Thi O	thio@gmail.com	0914567890	0	0.00
6	US015	Thu Van P	vanp@gmail.com	0915678901	0	0.00
7	US016	To Thi Q	thiq@gmail.com	0916789012	0	0.00
8	US017	Ha Van R	vanr@gmail.com	0917890123	0	0.00
9	US018	Quach Thi S	this@gmail.com	0918901234	0	0.00
10	US019	Nguyen Phu Vinh	svPV@gmail.com	0919012345	0	0.00
11	US021	Lu Chan Vu	svLCV@gmail.com	0919234567	0	0.00
12	US022	Le Minh Khoa	svLMK@gmail.com	0919345678	0	0.00
13	US023	Le Minh Tri	svLMT@gmail.com	0919456789	0	0.00
14	US024	Huynh Phu Vinh	phuvinh@gmail.com	0123456789	0	0.00
15	US003	Le Van C	vanc@yahoo.com	0903456789	1	404595.00
16	US002	Tran Thi B	thib@gmail.com	0902345678	1	259000.00
17	US009	Do Van I	vani@gmail.com	0909012345	1	250000.00
18	US020	Huynh Xuan Quoc Viet	svVIT@gmail.com	0919123456	1	243512.00
19	US006	Vo Thi F	thif@gmail.com	0906789012	1	159000.00
20	US008	Bui Thi H	thih@gmail.com	0908901234	1	149000.00
21	US007	Dang Van G	vang@hotmail.com	0907890123	1	145000.00
22	US005	Hoang Van E	vane@gmail.com	0905678901	1	140000.00
23	US010	Ngo Thi K	thik@gmail.com	0910123456	1	120000.00
24	US001	Nguyen Van A	vana@gmail.com	0901234567	2	569000.00

```
Tìm thấy 24 khách hàng.
Total execution time: 00:00:00.081
```

Hình 20: Kết quả Test 2

```
Started executing query at  Line 55
Msg 51001, Level 16, State 1, Procedure sp_SearchUsersByBookingCount, Line 10
Lỗi: Số lượng đơn hàng tối đa không được âm.
Total execution time: 00:00:00.064
```

Hình 21: Kết quả Test 3

6.4 Hàm

6.4.1 Hàm fn_GetMovieRevenue

```sql
CREATE OR ALTER FUNCTION fn_GetMovieRevenue
(
    @MovieID VARCHAR(10)
)
RETURNS NVARCHAR(MAX)
AS
BEGIN
    DECLARE @Result NVARCHAR(MAX) = N'';
    DECLARE @MovieTitle NVARCHAR(100);
    DECLARE @TotalRevenue DECIMAL(18, 2) = 0;
    DECLARE @ShowtimeCount INT = 0;

    IF @MovieID IS NULL
        RETURN N'Lop: MovieID khong duoc NULL';

    SELECT @MovieTitle = Title FROM MOVIE WHERE Movie_ID = @MovieID;
    IF @MovieTitle IS NULL
        RETURN N'Loi: Khong tim thay phim voi ID: ' + @MovieID;

    SET @Result = N'Phim: ' + @MovieTitle + CHAR(13) + CHAR(10);

    DECLARE @ShowtimeID VARCHAR(10);
    DECLARE @StartTime DATETIME;
    DECLARE @Revenue DECIMAL(18, 2);

    DECLARE showtime_cursor CURSOR FOR
    SELECT S.Showtime_ID, S.StartTime
    FROM SHOWTIME S
    WHERE S.Movie_ID = @MovieID
    ORDER BY S.StartTime;

    OPEN showtime_cursor;
    FETCH NEXT FROM showtime_cursor INTO @ShowtimeID, @StartTime;

    WHILE @@FETCH_STATUS = 0
    BEGIN
        SELECT @Revenue = ISNULL(SUM(B.TotalPrice), 0)
        FROM BOOKING B
        JOIN INCLUDE_SEAT I ON B.Booking_ID = I.Booking_ID
        WHERE I.Showtime_ID = @ShowtimeID;

        SET @TotalRevenue = @TotalRevenue + @Revenue;
        SET @ShowtimeCount = @ShowtimeCount + 1;

        SET @Result = @Result +
            N'  Suat ' + CAST(@ShowtimeCount AS NVARCHAR) +
            N' (' + CONVERT(NVARCHAR, @StartTime, 120) + N'): ' +
            FORMAT(@Revenue, 'NO') + N' VND' + CHAR(13) + CHAR(10);
```

```
49
50          FETCH NEXT FROM showtime_cursor INTO @ShowtimeID, @StartTime;
51      END
52
53      CLOSE showtime_cursor;
54      DEALLOCATE showtime_cursor;
55
56      SET @Result = @Result +
57          N'---' + CHAR(13) + CHAR(10) +
58          N'Tong doanh thu: ' + FORMAT(@TotalRevenue, 'NO') + N' VND' + CHAR(13) +
    CHAR(10) +
59          N'Tong so suat chieu: ' + CAST(@ShowtimeCount AS NVARCHAR);
60
61      RETURN @Result;
62  END;
```

Giải thích hàm:

Hàm `fn_GetMovieRevenue` được thiết kế để tính toán doanh thu chi tiết của một bộ phim dựa trên tất cả các suất chiếu của phim đó. Không chỉ trả về tổng doanh thu, hàm còn cung cấp báo cáo chi tiết từng suất chiếu dưới dạng chuỗi văn bản có cấu trúc.

Hàm yêu cầu 1 tham số duy nhất: @MovieID là mã định danh duy nhất của bộ phim cần phân tích

Hàm truy vấn thông tin cơ bản của phim từ bảng MOVIE. Sau đó khởi tạo con trỏ, truy vấn danh sách tất cả suất chiếu của phim (sắp xếp theo thời gian) và sử dụng vòng lặp WHILE để duyệt qua từng suất chiếu, thực hiện:

- Tính doanh thu suất chiếu: Truy vấn tổng giá trị các vé đã bán cho suất chiếu hiện tại.
- Cộng dồn tổng doanh thu: Tích lũy vào biến tổng doanh thu của phim.
- Định dạng thông tin: Chuyển đổi dữ liệu thành chuỗi mô tả có cấu trúc.
- Xây dựng báo cáo chi tiết: Thêm thông tin suất chiếu vào chuỗi kết quả.

Xử lý ngoại lệ:

- Nếu người dùng không truyền tham số @MovieID hoặc không thể tìm thấy phim với @MovieID tương ứng, hàm sẽ trả về thông báo lỗi cụ thể.

Testing

```
1  -- Test 1: MovieID hop le
2  SELECT dbo.fn_GetMovieRevenue('MV001') AS MovieRevenueDetail;
3
4  -- Test 2: MovieID khong ton tai
5  SELECT dbo.fn_GetMovieRevenue('MV999') AS MovieRevenueDetail;
6
7  -- Test 3: MovieID NULL
8  SELECT dbo.fn_GetMovieRevenue(NULL) AS MovieRevenueDetail;
```

Kết quả

```
Phim: Mai
  Suat 1 (14/02/2024): 809,000 VND
  Suat 2 (14/02/2024): 259,000 VND
---
Tong doanh thu: 1,068,000 VND
Tong so suat chieu: 2
```

Hình 22: Kết quả Test 1

```
Loi: Khong tim thay phim voi ID: MV999
```

Hình 23: Kết quả Test 2

```
Loi: MovieID khong duoc NULL
```

Hình 24: Kết quả Test 3

6.4.2 Hàm `fn_GetRoomOccupancyRate`

```sql
    CREATE OR ALTER FUNCTION fn_GetRoomOccupancyRate
(
    @RoomID VARCHAR(10),
    @StartDate DATETIME,
    @EndDate DATETIME
)
RETURNS DECIMAL(5, 2)
AS
BEGIN
    DECLARE @OccupancyRate DECIMAL(5, 2) = 0;
    DECLARE @TotalSeats INT;
    DECLARE @TotalShowtimes INT = 0;
    DECLARE @TotalBookedSeats INT = 0;

    DECLARE @ShowtimeID VARCHAR(10);
    DECLARE @BookedSeats INT;

    IF @RoomID IS NULL OR LTRIM(RTRIM(@RoomID)) = ''
    BEGIN
        RETURN -1;
    END

    IF @StartDate IS NULL OR @EndDate IS NULL
    BEGIN
        RETURN -2;
```

```
    END

    IF @StartDate > @EndDate
    BEGIN
        RETURN -3;
    END

    IF NOT EXISTS (SELECT 1 FROM ROOM WHERE Room_ID = @RoomID)
    BEGIN
        RETURN -4;
    END

    SELECT @TotalSeats = Capacity
    FROM ROOM
    WHERE Room_ID = @RoomID;

    IF @TotalSeats = 0
    BEGIN
        RETURN 0;
    END

    DECLARE showtime_cursor CURSOR FOR
    SELECT Showtime_ID
    FROM SHOWTIME
    WHERE Room_ID = @RoomID
        AND StartTime BETWEEN @StartDate AND @EndDate
    ORDER BY StartTime;

    OPEN showtime_cursor;
    FETCH NEXT FROM showtime_cursor INTO @ShowtimeID;

    WHILE @@FETCH_STATUS = 0
    BEGIN
        SET @TotalShowtimes = @TotalShowtimes + 1;

        SELECT @BookedSeats = COUNT(*)
        FROM INCLUDE_SEAT
        WHERE Showtime_ID = @ShowtimeID
            AND SeatStatus IN ('Booked', 'Sold');

        SET @TotalBookedSeats = @TotalBookedSeats + @BookedSeats;

        FETCH NEXT FROM showtime_cursor INTO @ShowtimeID;
    END

    CLOSE showtime_cursor;
    DEALLOCATE showtime_cursor;

    IF @TotalShowtimes > 0
    BEGIN
        SET @OccupancyRate = (CAST(@TotalBookedSeats AS DECIMAL(18, 2)) /
                              (CAST(@TotalSeats AS DECIMAL(18, 2)) *
    @TotalShowtimes)) * 100;
    END
    ELSE
    BEGIN
        SET @OccupancyRate = 0;
    END

    RETURN @OccupancyRate;
END;
```

Giải thích hàm:

`fn_GetRoomOccupancyRate` là hàm được thiết kế để tính toán tỷ lệ lấp đầy ghế của một phòng chiếu phim trong một khoảng thời gian cụ thể. Tỷ lệ này là chỉ số quan trọng đánh giá hiệu quả sử dụng không gian và khả năng thu hút khán giả của phòng chiếu.

Hàm yêu cầu 3 tham số:

- @RoomID: Mã định danh của phòng chiếu cần phân tích.
- @StartDate: Ngày bắt đầu khoảng thời gian phân tích.
- @EndDate: Ngày kết thúc khoảng thời gian phân tích.

Hàm truy vấn tổng số ghế (Capacity) từ bảng ROOM và trả về 0 ngay lập tức nếu phòng không có ghế (Capacity = 0).

Hàm sử dụng con trỏ và vòng lặp WHILE để duyệt tuần tự qua tất cả các suất chiếu của phòng:

- Tăng bộ đếm suất chiếu: Theo dõi tổng số suất chiếu trong khoảng thời gian.
- Đếm ghế đã đặt: Truy vấn số ghế có trạng thái 'Booked' hoặc 'Sold' cho mỗi suất chiếu.
- Cộng dồn số ghế: Tích lũy tổng số ghế đã đặt qua tất cả suất chiếu.

Sau khi thu thập đầy đủ dữ liệu hàm tính toán tỉ lệ lấp đầy theo công thức:

Occupancy Rate = (Tổng số ghế đã đặt / (Sức chứa phòng × Tổng số suất chiếu)) × 100

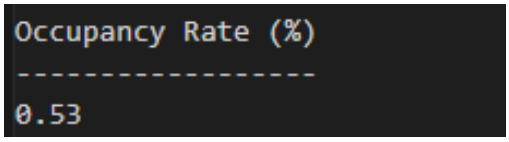
Xử lý ngoại lệ:

- Nếu người dùng không truyền tham số @RoomID, @StartDate hoặc @EndDate, hàm sẽ trả về mã lỗi tương ứng (-1 cho RoomID và -2 cho StartDate và EndDate).
- Nếu EndDate bé hơn StartDate thì hàm sẽ trả về mã lỗi là -3.
- Nếu RoomID không tồn tại thì hàm sẽ trả về mã lỗi là -4.

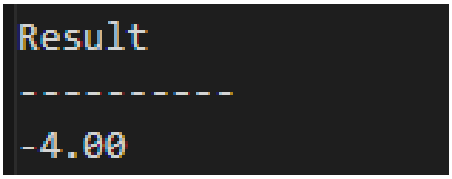
Testing

```
-- Test 1: Tinh ti le lap day cho phong hop le
DECLARE @Rate DECIMAL(5, 2);
SET @Rate = dbo.fn_GetRoomOccupancyRate('R001', '2024-01-01', '2024-12-31');
SELECT @Rate AS [Occupancy Rate (%)];

-- Test 2: RoomID khong ton tai
SELECT dbo.fn_GetRoomOccupancyRate('R999', '2024-01-01', '2024-12-31') AS [Result
    ];

-- Test 3: Ngay khong hop le
SELECT dbo.fn_GetRoomOccupancyRate('R001', '2024-12-31', '2024-01-01') AS [Result
    ];
```

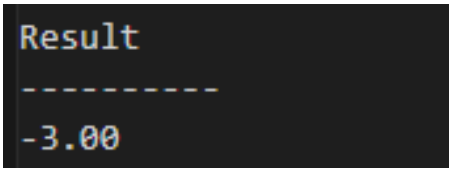
Kết quả



Hình 25: Kết quả Test 1



Hình 26: Kết quả Test 2



Hình 27: Kết quả Test 3

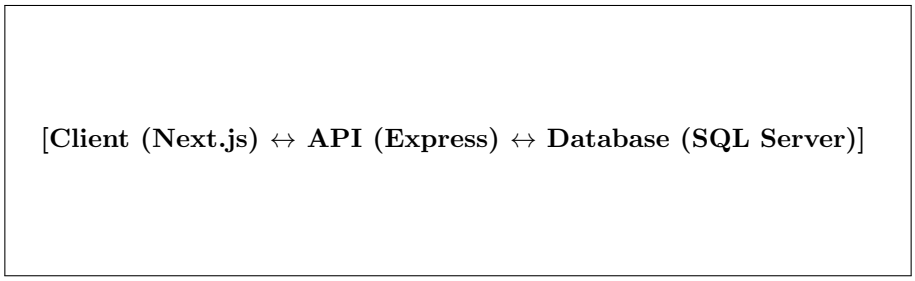
7 Hiện thực ứng dụng

Nhóm đã phát triển một ứng dụng web quản lý rạp chiếu phim để minh họa việc kết nối và tương tác với cơ sở dữ liệu SQL Server. Ứng dụng tập trung vào các thao tác dữ liệu thông qua Stored Procedures và xử lý các ràng buộc nghiệp vụ.

7.1 Kiến trúc hệ thống

Hệ thống tuân theo mô hình Client-Server 3 lớp:

- **Database Layer (SQL Server):** Lưu trữ dữ liệu, đảm bảo toàn vẹn dữ liệu thông qua Constraints, Triggers và cung cấp logic nghiệp vụ qua Stored Procedures.
- **Backend Layer (Node.js + Express):** Cung cấp RESTful API. Sử dụng thư viện `mssql` để kết nối Database, thực thi các thủ tục lưu trữ và xử lý logic trung gian (Authentication, Validation).
- **Frontend Layer (Next.js):** Giao diện người dùng (UI) được xây dựng bằng React, tương tác với Backend qua HTTP Requests (Axios/Fetch).



Hình 28: Mô hình kiến trúc ứng dụng

7.2 Kết nối Cơ sở dữ liệu

Kết nối được thiết lập trong file `db.js` sử dụng biến môi trường để bảo mật.

```
import sql from "mssql";

const sqlConfig = {
    user: process.env.DB_USER,
    password: process.env.DB_PASSWORD,
    database: "BTL_DB",
    server: process.env.DB_SERVER,
    port: +process.env.DB_PORT,
    options: {
        encrypt: true,
        trustServerCertificate: false
    }
};

const connectDB = async () => {
    try {
        await sql.connect(sqlConfig);
        console.log("Microsoft SQL server connected");
    } catch (error) {
        console.error("Connection failed:", error);
    }
};
export default { connectDB };
```

7.3 Hiện thực Backend và Frontend

7.3.1 Quản lý Người dùng (User Management)

Đây là chức năng chính minh họa cho việc Thêm/Xóa/Sửa dữ liệu (Câu 3.1) và Hiển thị danh sách/Xử lý lỗi (Câu 3.2).

1. Backend API (Controller)

File `userController.js` xử lý các request từ client và gọi các Stored Procedure tương ứng: `sp_InsertUser`, `sp_UpdateUser`, `sp_DeleteUser`.

```
// 1. Them nguoi dung (Goi sp_InsertUser)
export const createUser = async (req, res) => {
```

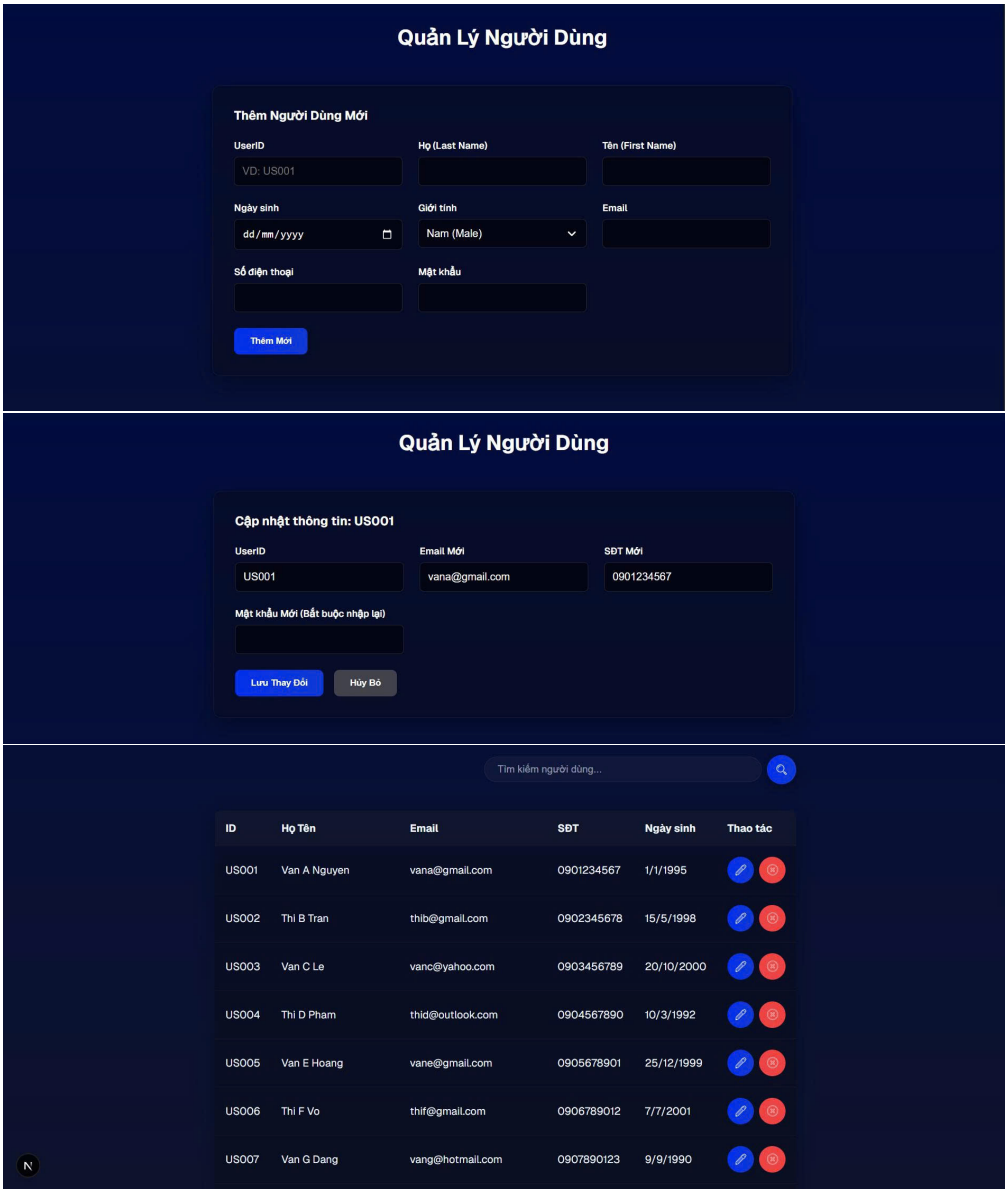
```
    const { UserID, FirstName, LastName, Email, Phone, Password, ... } = req.body;
    try {
        const request = new sql.Request();
        request.input("UserID", sql.VarChar(10), UserID);
        // ... mapping cac tham so khac

        // Thuc thi thu tuc
        await request.execute("sp_InsertUser");
        res.status(201).json({ message: "Them nguoi dung thanh cong!" });
    } catch (error) {
        // Bat loi vi pham rang buoc (VD: UserID da ton tai)
        res.status(400).json({ message: error.message });
    }
};
// 2. Cap nhat nguoi dung (Goi sp_UpdateUser)
export const updateUser = async (req, res) => {
    const { id } = req.params;
    const { NewEmail, NewPhone, NewPassword } = req.body;

    try {
        const request = new sql.Request();
        request.input("UserID", sql.VarChar(10), id);
        request.input("NewEmail", sql.VarChar(100), NewEmail);
        request.input("NewPhone", sql.VarChar(15), NewPhone);
        request.input("NewPassword", sql.VarChar(50), NewPassword);

        await request.execute("sp_UpdateUser");

        res.status(200).json({ message: "Cap nhat thanh cong!" });
    } catch (error) {
        console.error("SQL Error:", error.message);
        res.status(400).json({ message: error.message });
    }
};
// 3. Xoa nguoi dung (Goi sp_DeleteUser)
export const deleteUser = async (req, res) => {
    const { id } = req.params;
    try {
        const request = new sql.Request();
        request.input("UserID", sql.VarChar(10), id);

        await request.execute("sp_DeleteUser");
        res.status(200).json({ message: "Xoa thanh cong!" });
    } catch (error) {
        // Bat loi khoa ngoai (VD: User da mua ve)
        res.status(400).json({ message: error.message });
    }
};
```

2. Frontend (Giao diện và Logic)

Giao diện `/users` được thiết kế để đáp ứng các yêu cầu:

- **Danh sách dữ liệu:** Hiển thị bảng User với đầy đủ thông tin.
- **Thêm/Sửa trên cùng giao diện:** Form tự động chuyển đổi trạng thái. Khi sửa, các trường không được phép thay đổi (như UserID, Tên) sẽ bị khóa hoặc ẩn đi, tuân thủ logic của `sp_UpdateUser`.
- **Xử lý lỗi logic:** Các lỗi từ Database (như nhập mật khẩu $<$ 8 ký tự, xóa user đang có

vé) được hiển thị rõ ràng cho người dùng thông qua thông báo (Toast/Alert).



Hình 29: Giao diện Quản lý Người dùng

7.3.2 Thống kê và Quản lý Khách hàng (User Statistics & Management)

Chức năng này minh họa cho yêu cầu Tìm kiếm nâng cao và Thống kê tổng hợp. Hệ thống cho phép lọc người dùng dựa trên tên và ố lượng đơn hàng đã đặt, đồng thời tính toán tổng chi tiêu (sử dụng kết hợp `COUNT`, `SUM` và `HAVING`).

1. Backend API (Controller)

File `procedureController.js` xử lý logic tìm kiếm thông qua phương thức `countBooking`. API nhận tham số từ Query String, xử lý thành tham số đầu vào của thủ tục và tiến hành thực thi thủ tục:

- **Xử lý tham số:** Biến `q` tương đương tham số `@Keyword`, biến `max` tương đương tham số `@MaxBookingCount`.
- **Chuẩn hóa dữ liệu:** Xử lý các trường hợp mặc định (tham số `null`) và ép kiểu số nguyên trước khi truyền vào `sql.Request`.
- **Trả về kết quả:** Kết quả thực thi (`data.recordset`) được trả về trực tiếp dưới dạng JSON để Frontend hiển thị.

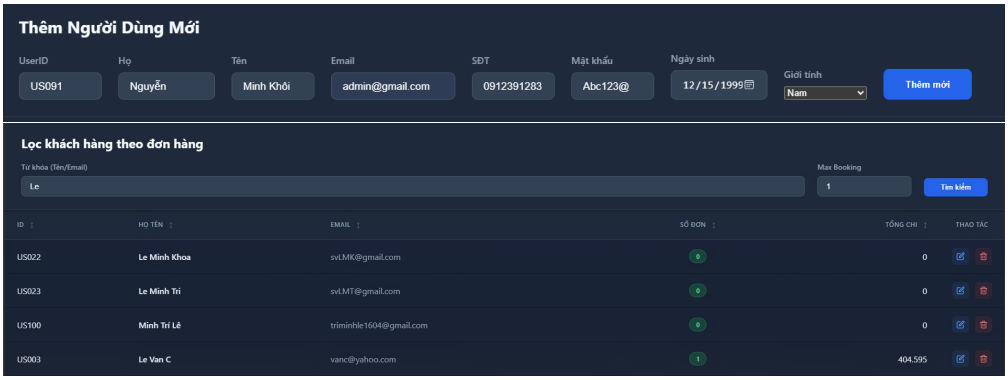
```
const ProcedureController = {
    countBooking: async (req, res) => {
        try {
            const { q: keyword = null, max: limit = null } = req.query;
            const request = new sql.Request();
            request.input("Keyword", sql.NVarChar, keyword);
            request.input("MaxBookingCount", sql.Int, limit ? +limit : null);
            const data = await request.execute("sp_SearchUsersByBookingCount");
            res.status(200).json({ message: "Query success", data: data.recordset
    });
        } catch (error) {
            console.log(error);
            res.status(500).json({ message: "Server error", error: error.message
    });
        }
    },
};
```

2. Frontend (Giao diện và Logic)

Giao diện `/admin/customer-filter` được thiết kế tập trung trên một màn hình duy nhất với 3 khu vực chức năng chính để tối ưu thao tác người dùng:

- **Form thêm người dùng nhanh:** Cung cấp các trường nhập liệu chi tiết (UserID, Tên, Email, Mật khẩu...). Hệ thống tự động kiểm tra dữ liệu đầu vào và hiển thị thông báo trạng thái (thành công hoặc lỗi từ server) ngay phía dưới form để người dùng dễ dàng nhận biết.
- **Bộ lọc Thống kê:** Đây là công cụ chính của người quản trị hệ thống, cho phép tìm kiếm nâng cao dựa trên từ khóa và giới hạn số lượng đơn hàng (Max Booking). Dữ liệu trả về không chỉ có thông tin cá nhân mà còn bao gồm các chỉ số thống kê tổng hợp (Tổng số đơn, Tổng chi tiêu).
- **Bảng dữ liệu tương tác:** Đây là giao diện chính mà người quản trị hệ thống tương tác để đưa ra các chiến lược đối quan trọng thông qua nguồn thông tin dữ liệu trả về (kết quả trả về là quá trình thực thi thủ tục `sp_SearchUsersByBookingCount` với các tham số là dữ liệu từ bộ lọc bên trên)
  - *Sắp xếp (Sorting):* Người dùng có thể nhấp vào tiêu đề cột để sắp xếp dữ liệu tăng/giảm dần ngay lập tức mà không cần tải lại trang.

– *Xử lý xóa an toàn:* Thao tác xóa được đảm bảo bằng hộp thoại xác nhận (Modal Popup). Nếu xảy ra lỗi, thông báo lỗi cụ thể sẽ hiển thị ngay trong hộp thoại để cảnh báo người dùng.



Hình 30: Giao diện Thống kê và Quản lý Khách hàng

7.3.3 Thống kê doanh thu phim (Movie's revenue statistics)

1. Backend API (Controller)

File `revenueController.js` là module chịu trách nhiệm xử lý các yêu cầu liên quan đến doanh thu phim, bao gồm:

- Hàm `revenueDetail`: Lấy chi tiết doanh thu của một phim theo ID, dưới dạng chuỗi mô tả đầy đủ được trả về từ hàm SQL `fn_GetMovieRevenue`.
- Hàm `moviesRevenue`: Lấy doanh thu của toàn bộ phim và chuyển đổi sang dạng JSON chuẩn cho frontend.

```
const RevenueController = {
    revenueDetail: async (req, res) => {
        try {
            const movieId = req.params.id;
            const request = new sql.Request();
            const result = await request.input("MovieID", sql.VarChar(10), movieId
    ).query(`SELECT dbo.fn_GetMovieRevenue(@MovieID) AS Revenue`);
            res.status(200).json({ detail: result.recordset[0].Revenue });
        } catch (error) {
            console.log(error);
            res.status(500).json({ message: "Server error", error: error.message
    });
        }
    },

    moviesRevenue: async (req, res) => {
        try {
            const request = new sql.Request();
            const result = await request.query(`SELECT
                M.Movie_ID AS id,
                dbo.fn_GetMovieRevenue(M.Movie_ID) AS revenueDetail
```

```
            FROM MOVIE M
            ORDER BY M.Movie_ID;
            `)
            res.status(200).json({ data: result.recordset.map(parseRevenueDetail)
    }); // ham parseRevenueDetail chuyen chuoi tra ve tu fn_GetMovieRevenue thanh
    JSON cho fe
        } catch (error) {
            console.log(error);
            res.status(500).json({ message: "Server error", error: error.message
    });
        }
    },
};
```

2. Frontend (Giao diện và Logic)

Giao diện `/admin/revenue` được thiết kế theo phong cách tương tự hai màn hình quản lý người dùng, nhưng tập trung hoàn toàn vào việc trực quan hóa doanh thu phim. Màn hình này giúp giảng viên có thể kiểm tra trực tiếp kết quả hàm `fn_GetMovieRevenue` và hai API `moviesRevenue`, `revenueDetail` mà không cần thao tác trên SSMS.

Về bố cục, trang được chia thành hai khu vực chính:

- **Thanh tổng quan doanh thu (Summary Bar):** Nằm ngay phía trên đầu trang, hiển thị *Tổng doanh thu của toàn bộ phim* được tính từ dữ liệu trả về của API `moviesRevenue`. Khi danh sách phim thay đổi, giá trị tổng cũng được cập nhật lại tương ứng, giúp người quản trị có cái nhìn nhanh về tình hình doanh thu chung của toàn hệ thống.
- **Bảng danh sách phim (Movie Revenue Table):** Khu vực trung tâm của màn hình là một bảng dữ liệu hiển thị từng bộ phim, bao gồm các cột cơ bản như:
  - `Movie_ID`: mã phim.
  - `Title`: tên phim.
  - `TotalRevenue`: doanh thu tổng của phim (đã được Backend bóc tách từ chuỗi trả về của `fn_GetMovieRevenue` và chuyển sang dạng số để frontend định dạng `VND`).

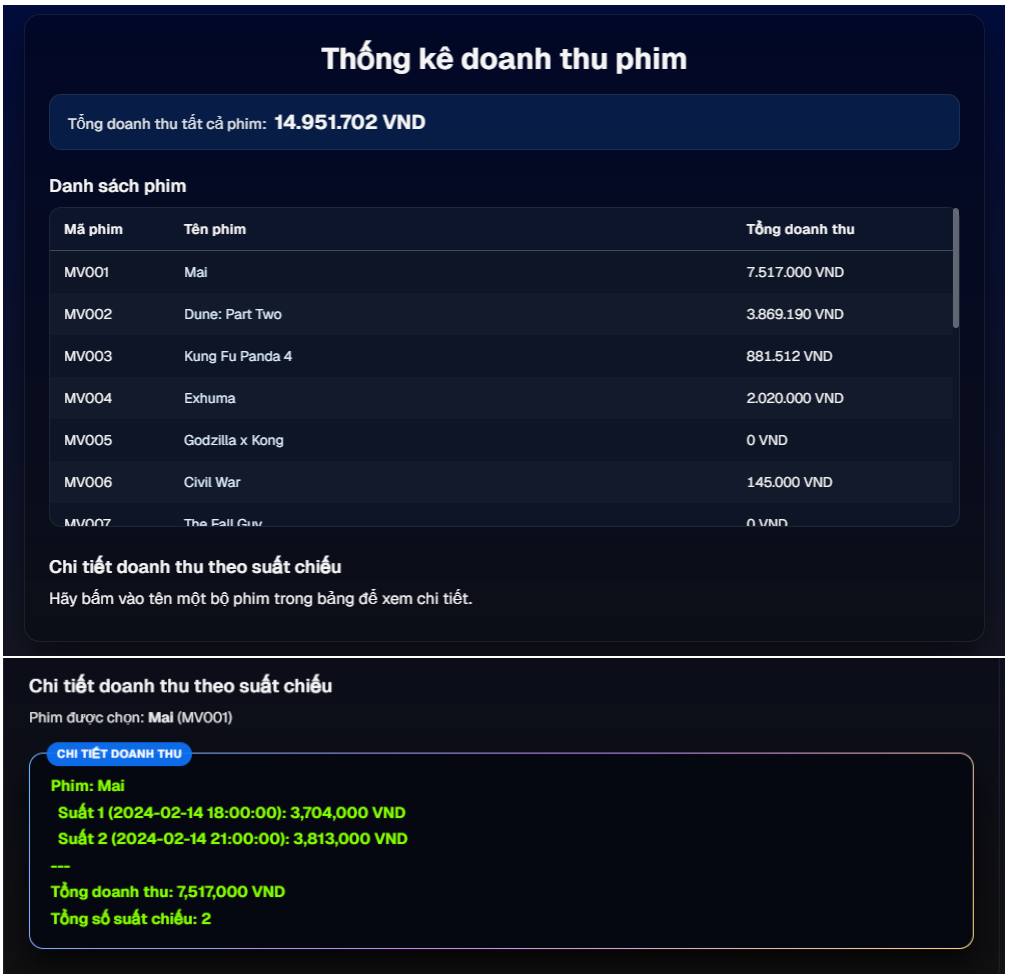
  Bảng được đặt trong một khung cuộn (*scrollable container*) để có thể hiển thị danh sách dài mà không phá vỡ bố cục trang. Các hàng được tô màu khi rê chuột, giúp người dùng dễ nhận biết dòng đang thao tác. Tên phim được hiển thị dạng `button/link`; khi người dùng bấm vào, ứng dụng gửi yêu cầu tới API `/api/movies/:id/revenue-detail` và nạp dữ liệu chi tiết xuống khu vực bên dưới.

Khu vực thứ hai trên trang là **panel hiển thị chi tiết doanh thu phim**. Mặc định, khi chưa chọn phim nào, panel chỉ hiển thị một dòng hướng dẫn ngắn gọn. Sau khi người dùng bấm vào tên một bộ phim trong bảng, UI sẽ:

- Ghi nhận `MovieID` và `Title` đang được chọn, hiển thị phía trên panel theo dạng: *"Phim được chọn: [Tên phim] ([MovieID])"*.
- Hiển thị trạng thái tải dữ liệu (*"Dang tải chi tiết doanh thu..."*) trong khi chờ API trả về.
- Sau khi nhận phản hồi từ Backend, phần thân panel hiển thị nội dung ở dạng `<pre>` để giữ nguyên format dòng, xuống hàng và căn lề giống kết quả chạy hàm `fn_GetMovieRevenue` trong SSMS (bao gồm tiêu đề phim, từng suất chiếu, tổng doanh thu và tổng số suất chiếu).

Giao diện cũng xử lý rõ ràng các trường hợp lỗi nghiệp vụ từ hàm SQL:

- Nếu API trả về chuỗi bắt đầu bằng `"Loi:"` hoặc `"Lop:"` (ví dụ: `"Loi: Khong tim thay phim voi ID: MV999"` hoặc `"Lop: MovieID khong duoc NULL"`), frontend vẫn hiển thị nguyên văn nội dung trong panel chi tiết nhưng dùng màu chữ cảnh báo để người dùng dễ nhận biết đó là thông báo lỗi từ hệ thống.
- Nếu lỗi xảy ra ở tầng API (ví dụ mất kết nối, lỗi máy chủ), một thông báo lỗi riêng (`toast/alert`) sẽ xuất hiện phía trên bảng dữ liệu, đồng thời panel chi tiết giữ nguyên dữ liệu lần xem gần nhất để tránh gây khó chịu cho người thao tác.



Hình 31: Giao diện Thống kê doanh thu phim