BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC ĐẠI NAM

BÀI TẬP LỚN

TÊN HỌC PHẦN: HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU

ĐỀ TÀI: THIẾT KẾ CƠ SỞ DỮ LIỆU HỆ THỐNG QUẢN LÝ VÉ XEM PHIM

Giảng viên hướng dẫn: ThS. Trần Thị Thanh Nhàn

Sinh viên thực hiện:

STT	Mã Sinh Viên	Họ và Tên	Ngày Sinh	Lớp
1	1771020308	Trần Khắc Hồng	02/10/2005	CNTT 17-04
2	1771020194	Nguyễn Văn Được	22/12/2005	CNTT 17-04
3	1771020189	Lê Tuấn Dũng	28/11/2005	CNTT 17-04

Hà Nội, năm 2025

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC ĐẠI NAM

BÀI TẬP LỚN

TÊN HỌC PHẦN: HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU

ĐỀ TÀI: THIẾT KẾ CƠ SỞ DỮ LIỆU HỆ THỐNG QUẢN LÝ VÉ XEM PHIM

Giảng viên hướng dẫn: ThS. Trần Thị Thanh Nhàn

Sinh viên thực hiện:

STT	Mã Sinh Viên	Họ và Tên	Ngày Sinh	Điểm	
				Bằng Số	Bằng Chữ
1	1771020308	Trần Khắc Hồng	02/10/2005		
2	1771020194	Nguyễn Văn Được	22/12/2005		
3	1771020189	Lê Tuấn Dũng	28/11/2005		

CÁN BỘ CHẤM THI 1 **CÁN BỘ CHẤM THI 2**

Hà Nội, năm 2025

LỜI NÓI ĐẦU

Trong bối cảnh ngành giải trí và điện ảnh không ngừng phát triển, nhu cầu của khán giả ngày càng đa dạng và yêu cầu cao hơn về chất lượng dịch vụ. Việc quản lý hoạt động đặt vé hiệu quả đóng vai trò quan trọng trong việc nâng cao trải nghiệm khách hàng, từ đó gia tăng tính cạnh tranh cho các rạp chiếu phim. Hệ thống quản lý vé không chỉ đơn thuần là công cụ hỗ trợ công việc mà còn là yếu tố quyết định trong việc tối ưu hóa quy trình vận hành, đảm bảo tính chính xác và chuyên nghiệp trong mọi giao dịch.

Một hệ thống quản lý vé xem phim hiện đại cần đáp ứng được nhiều yêu cầu như lưu trữ thông tin khách hàng, quản lý lịch chiếu, tình trạng ghế ngồi, cũng như tích hợp các dịch vụ bổ sung như đặt vé trực tuyến, thanh toán điện tử hay các chương trình khuyến mãi. Chính vì vậy, cơ sở dữ liệu (CSDL) đóng vai trò then chốt trong việc tổ chức và lưu trữ dữ liệu một cách hiệu quả. Thiết kế CSDL hợp lý giúp đảm bảo tính nhất quán, khả năng mở rộng và an toàn của hệ thống, từ đó giúp rạp chiếu phim vận hành trơn tru và đáp ứng nhanh chóng nhu cầu thay đổi của khách hàng.

Thông qua việc áp dụng các nguyên tắc thiết kế cơ sở dữ liệu hiện đại và các công nghệ lập trình tiên tiến, hệ thống quản lý vé xem phim sẽ giúp nâng cao hiệu suất làm việc của nhân viên rạp, giảm thiểu sai sót trong quy trình quản lý và cải thiện đáng kể trải nghiệm của khách hàng. Điều này không chỉ tối ưu hóa nguồn lực mà còn mang lại lợi thế cạnh tranh cho các rạp chiếu phim trong việc phát triển dịch vụ và mở rộng kinh doanh bền vững.

MỤC LỤC

CHƯƠNG 1. TỔNG QUAN VỀ ĐỀ TÀI

1.1 Giới thiệu đề tài

Trong thời đại công nghệ số phát triển mạnh mẽ, nhu cầu giải trí của con người ngày càng gia tăng, đặc biệt là việc xem phim tại rạp chiếu. Hệ thống quản lý rạp chiếu phim đóng vai trò quan trọng trong việc tổ chức, quản lý lịch chiếu phim, đặt vé, kiểm soát khách hàng, nhân viên và doanh thu. Chính vì vậy, việc xây dựng một hệ thống quản lý rạp phim là vô cùng cần thiết để giúp tối ưu hóa hoạt động kinh doanh, nâng cao trải nghiệm khách hàng và giảm thiểu sai sót trong quá trình vận hành.

Việc đặt vé truyền thống có thể gặp nhiều bất tiện như mất thời gian xếp hàng, khó khăn trong việc chọn ghế hoặc đôi khi xảy ra lỗi trong hệ thống kiểm soát vé. Hệ thống quản lý rạp phim giúp số hóa các quy trình này, từ đó tăng cường tính chuyên nghiệp và tiện lợi trong quá trình đặt vé, kiểm tra thông tin và quản lý dữ liệu khách hàng. Với sự phát triển của thương mại điện tử, các rạp phim cũng cần tích hợp các phương thức thanh toán trực tuyến, đặt vé qua ứng dụng để nâng cao chất lượng phục vụ.

1.2 Mục tiêu nghiên cứu

Mục tiêu chính của đề tài là thiết kế và xây dựng một hệ thống quản lý rạp phim giúp tự động hóa các quy trình nghiệp vụ, nâng cao hiệu suất làm việc và cải thiện trải nghiệm người dùng. Cụ thể, hệ thống cần đạt được các mục tiêu sau:

- Hỗ trợ khách hàng đặt vé trực tuyến nhanh chóng và thuận tiện: Giúp khách hàng có thể dễ dàng lựa chọn phim, thời gian chiếu, chỗ ngồi, thanh toán trực tuyến và nhận vé điện tử mà không cần xếp hàng chờ mua vé tại rạp.
- Quản lý thông tin suất chiếu, phòng chiếu và ghế ngồi hiệu quả: Đảm bảo thông tin về các suất chiếu luôn được cập nhật chính xác, giúp khách hàng dễ dàng tra cứu và lựa chọn phù hợp.
- Kiểm soát doanh thu từ việc bán vé và các chương trình khuyến mãi: Hệ thống cần có chức năng báo cáo doanh thu theo ngày, tuần, tháng để nhà quản lý có thể theo dõi và điều chỉnh chiến lược kinh doanh kịp thời.

- Cung cấp báo cáo và thống kê giúp nhà quản lý ra quyết định: Báo cáo chi tiết về doanh thu, lượng vé bán ra, suất chiếu đông khách giúp đưa ra những chiến lược marketing hiệu quả hơn.

Hệ thống này không chỉ giúp quản lý dữ liệu hiệu quả hơn mà còn mang lại lợi ích lớn cho khách hàng, giúp họ dễ dàng tìm kiếm và đặt vé mà không cần tốn quá nhiều thời gian. Ngoài ra, hệ thống cũng giúp nhà quản lý đưa ra quyết định tốt hơn dựa trên dữ liệu thu thập từ người dùng.

1.3 Phạm vi nghiên cứu

- Hệ thống tập trung vào các chức năng chính của một hệ thống quản lý rạp phim, bao gồm:

- Quản lý thông tin khách hàng: Lưu trữ thông tin cá nhân, lịch sử đặt vé, tạo hồ sơ khách hàng thân thiết.
- Quản lý vé: Đặt vé, hủy vé, kiểm tra vé, tích hợp với hệ thống thanh toán điện tử.
- Quản lý phim và lịch chiếu: Lưu trữ danh sách phim, thể loại, đạo diễn, suất chiếu, theo dõi các bộ phim sắp ra mắt.
- Quản lý phòng chiếu: Số ghế, tên phòng, rạp chiếu, tình trạng ghế trống.
- Quản lý nhân viên: Thông tin nhân viên, lịch làm việc, tiền lương, theo dõi hiệu suất làm việc.
- Báo cáo và thống kê**:** Tạo báo cáo doanh thu, lượng khách hàng theo thời gian, phân tích xu hướng.

Ngoài ra, hệ thống có thể mở rộng để hỗ trợ tính năng đánh giá phim, gợi ý phim dựa trên sở thích của khách hàng.

1.4 Phương pháp nghiên cứu

- Để thực hiện đề tài, nhóm đã nghiên cứu áp dụng các phương pháp sau:

- Phân tích yêu cầu hệ thống: Thu thập thông tin về các nghiệp vụ cần thiết cho hệ thống quản lý rạp phim từ các rạp chiếu thực tế và ý kiến của khách hàng.

- Thiết kế mô hình dữ liệu: Xây dựng mô hình ERD để thể hiện mối quan hệ giữa các thực thể trong hệ thống, đảm bảo dữ liệu được tổ chức hợp lý.
- Triển khai cơ sở dữ liệu: Sử dụng SQL để tạo các bảng, ràng buộc và xử lý dữ liệu t rong hệ thống nhằm đảm bảo tính toàn vẹn dữ liệu.
- Xây dựng chương trình: Phát triển phần mềm dựa trên mô hình đã thiết kế, sử dụng ngôn ngữ lập trình phù hợp như Python, Java hoặc PHP kết hợp với giao diện web hoặc ứng dụng di động.
- Kiểm thử và đánh giá: Chạy thử nghiệm hệ thống với dữ liệu thực tế, phát hiện và sửa lỗi, đánh giá hiệu suất hệ thống và mức độ tiện lợi cho người dùng.

CHƯƠNG 2. MÔ HÌNH CƠ SỞ DỮ LIỆU QUAN HỆ

2.1. Xác định các thực thể, thuộc tính và ràng buộc

STT	Tên thực thể (danh sách thuộc tính)	Diễn giải tiếng Việt
1	CUSTOMER (customerID, fullName, address, phoneNumber, email, dateOfBirth)	Khách hàng (mã khách hàng, tên khách hàng, địa chỉ, số điện thoại, email, ngày sinh)
2	EMPLOYEE (employeeID, fullName, gender, dateOfBirth, address, phoneNumber, email, salary)	Nhân viên (mã nhân viên, tên nhân viên, giới tính, ngày sinh, địa chỉ, số điện thoại, email, lương)
3	TICKET (ticketID, customerID, showtimeID, seatID, price, purchaseDate)	Vé xem phim (mã vé, mã khách hàng, mã suất chiếu, mã ghế, giá vé, ngày mua)
4	SHOWTIME (showtimeID, roomID, movieID, showDate, startTime, endTime)	Suất chiếu (mã suất chiếu, mã phòng, mã phim, ngày chiếu, giờ bắt đầu, giờ kết thúc)
5	SCREENROOM (roomID, roomName, seatCount)	Phòng chiếu (mã phòng, tên phòng, số ghế)
6	SEAT (seatID, roomID, seatNumber, seatType, status)	Ghế (mã ghế, mã phòng, số ghế, loại ghế, trạng thái)
7	MOVIE (movieID, movieTitle, genre, director, duration, releaseYear)	Phim (mã phim, tên phim, thể loại, đạo diễn, thời lượng, năm sản xuất)
8	INVOICE (invoiceID, customerID, employeeID, issueDate, totalAmount)	Hóa đơn (mã hóa đơn, mã khách hàng, mã nhân viên, ngày lập, tổng tiền)

9	PROMOTION (promotionID, promotionName, description, startDate, endDate)	Khuyến mãi (mã khuyến mãi, tên khuyến mãi, mô tả, thời gian bắt đầu, thời gian kết thúc)

2.2. Xây dựng các bảng

2.2.1. Bảng Customers: Lưu trữ thông tin khách hàng

TT	Tên thuộc tính	Kiểu dữ liệu	Ràng buộc	Ghi chú
1	iCustomerID	INT IDENTITY(1,1)	PK	Mã khách hàng
2	sFullName	NVARCHAR(100)	NOT NULL	Họ và tên
3	sAddress	NVARCHAR(255)	NULL	Địa chỉ
4	dDateOfBirth	DATE	NULL	Ngày sinh
5	iPhoneNumber	NVARCHAR(20)	NULL	Số điện thoại
6	sEmail	NVARCHAR(100)	NULL	Email

2.2.2. Bảng Employees: Lưu trữ thông tin nhân viên

TT	Tên thuộc tính	Kiểu dữ liệu	Ràng buộc	Ghi chú
1	iEmployeeID	INT IDENTITY(1,1)	PK	Mã nhân viên
2	sFullName	NVARCHAR(100)	NOT NULL	Họ và tên
3	sAddress	NVARCHAR(255)	NULL	Địa chỉ
4	dDateOfBirth	DATE	NULL	Ngày sinh
5	iPhoneNumber	NVARCHAR(20)	NULL	Số điện thoại
6	dSalary	DECIMAL(10,2)	NULL	Lương

2.2.3. Bảng Movies: Lưu trữ thông tin phim

TT	Tên thuộc tính	Kiểu dữ liệu	Ràng buộc	Ghi chú
1	iMovieID	INT IDENTITY(1,1)	PK	Mã phim
2	sNameMovie	NVARCHAR(100)	NOT NULL	Tên phim

3	sDirector	NVARCHAR(255)	NULL	Đạo diễn
4	iMovieGenreID	INT	NULL	Mã thể loại phim

2.2.4. Bảng MovieGenres: Thể loại phim

TT	Tên thuộc tính	Kiểu dữ liệu	Ràng buộc	Ghi chú
1	iMovieGenreID	INT IDENTITY(1,1)	PK	Mã thể loại phim
2	sNameGenre	NVARCHAR(50)	NOT NULL	Tên thể loại

2.2.5. Bảng ShowTimes: Lưu trữ thông tin suất chiếu

TT	Tên thuộc tính	Kiểu dữ liệu	Ràng buộc	Ghi chú
1	iShowTimeID	INT IDENTITY(1,1)	PK	Mã suất chiếu
2	sMovieID	INT	FK	Mã phim
3	dBeginTime	DATETIME	NOT NULL	Thời gian bắt đầu
4	dEndTime	DATETIME	NOT NULL	Thời gian kết thúc
5	iCapacityUtilization	DECIMAL(5,2)	NULL	Hệ số sử dụng phòng

2.2.6. Bảng ScreenRooms: Lưu trữ thông tin phòng chiếu

TT	Tên thuộc tính	Kiểu dữ liệu	Ràng buộc	Ghi chú
1	iScreenRoomID	INT IDENTITY(1,1)	PK	Mã phòng chiếu
2	sNameRoom	NVARCHAR(50)	NOT NULL	Tên phòng
3	iTotalSeat	INT	NULL	Tổng số ghế

2.2.7. Bảng Tickets: Lưu trữ thông tin vé

TT	Tên thuộc tính	Kiểu dữ liệu	Ràng buộc	Ghi chú
1	iTicketID	INT IDENTITY(1,1)	PK	Mã vé
2	dPrice	DECIMAL(10,2)	NOT NULL	Giá vé
3	iEmployeeID	INT	FK	Mã nhân viên
4	iCustomerID	INT	FK	Mã khách hàng

5	iSeatID	INT	FK	Mã ghế ngồi
6	iMovieID	INT	FK	Mã phim
7	iScreenRoomID	INT	FK	Mã phòng chiếu
8	iShowTimeID	INT	FK	Mã suất chiếu
9	iPromotionID	INT	FK	Mã khuyến mại
10	sTicketType	NVARCHAR(50)	NULL	Loại vé
11	dTicketSaleDate	DATE	NULL	Ngày bán

2.2.8. Bảng Counters: Lưu trữ thông tin quầy bán vé.

TT	Tên thuộc tính	Kiểu dữ liệu	Ràng buộc	Ghi chú
1	iCounterID	INT	PK	Mã suất chiếu
2	iTicketID	INT	FK	Mã vé
3	iEmployeeID	INT	FK	Mã nhân viên
4	sNameCounter	NVARCHAR(50)	NOT NULL	Tên quầy

2.2.9. Bảng Seats: Lưu trữ thông tin về ghế ngồi

TT	Tên thuộc tính	Kiểu dữ liệu	Ràng buộc	Ghi chú
1	iSeatID	INT IDENTITY(1,1)	PK	Mã ghế
2	iScreenRoomID	INT	FK	Mã phòng chiếu
3	sSeatNumber	NVARCHAR(10)	NOT NULL	Số ghế
4	sStatus	NVARCHAR(10)	DEFAULT ‘Trống’	Trạng thái (‘Trống’, ‘Đã đặt’, ‘Đã bán’)

2.2.10. Bảng Promotions: Lưu trữ thông tin về các khuyến mãi

TT	Tên thuộc tính	Kiểu dữ liệu	Ràng buộc	Ghi chú
1	iPromotionID	INT IDENTITY(1,1)	PK	Mã khuyến mại
2	sPromotionCode	NVARCHAR(100)	NOT NULL	Mã khuyến mại
3	iDiscountPercentage	DECIMAL(5,2)	NULL	Giảm giá (%)

4	dStartDate	DATE	NULL	Ngày bắt đầu
5	dEndDate	DATE	NULL	Ngày kết thúc
6	sApplicableTicketType	NVARCHAR(50)	NULL	Loại vé áp dụng

2.2.11. Bảng Invoices: Lưu trữ thông tin về hóa đơn

TT	Tên thuộc tính	Kiểu dữ liệu	Ràng buộc	Ghi chú
1	iInvoiceID	INT IDENTITY(1,1)	PK	Mã khuyến mại
2	iTicketID	INT IDENTITY(1,1)	FK	Mã vé
3	iCustomerID	INT IDENTITY(1,1)	FK	Mã khách hàng
4	dAmount	DECIMAL(10,2)	NOT NULL	Số tiền thanh toán
5	sPaymentMethod	NVARCHAR(20)	Check (PaymentMethod IN (N'Tiền mặt', N'Thẻ', N'Ví điện tử'))	Phương thức thanh toán
6	dInvoiceDate	DATETIME	NOT NULL	Ngày lập hóa đơn

CHƯƠNG 3: TẠO CƠ SỞ DỮ LIỆU

3.1. Tạo Database

Cơ sở dữ liệu là một phần quan trọng trong hệ thống "Quản lý rạp phim", giúp lưu trữ và xử lý thông tin về khách hàng, vé xem phim, suất chiếu, phòng chiếu, nhân viên, hóa đơn và các khía cạnh khác liên quan đến hoạt động của rạp. Việc thiết kế cơ sở dữ liệu cần đảm bảo tính toàn vẹn dữ liệu, hiệu suất cao và khả năng mở rộng trong tương lai.

- Hệ thống cơ sở dữ liệu này nhằm mục đích:

  - Quản lý thông tin phim, khách hàng, nhân viên, phòng chiếu, suất chiếu.
  - Đảm bảo tính nhất quán và toàn vẹn dữ liệu.
  - Hỗ trợ thao tác truy vấn dữ liệu nhanh chóng và hiệu quả.
  - Cung cấp khả năng mở rộng để tích hợp với các tính năng mới trong tương lai.

3.1.1. Database Diagram

- Dựa trên file SQL và mô hình ERD, sơ đồ cơ sở dữ liệu hệ thống "Quản lý rạp phim" bao gồm các thực thể chính sau:

  - MovieGenre (Thể loại phim): Quản lý các thể loại phim với ID duy nhất.
  - Movie (Phim): Lưu thông tin chi tiết về phim như tên phim, đạo diễn, thể loại.
  - ScreenRoom (Phòng chiếu): Quản lý thông tin phòng chiếu, số lượng ghế.
  - Seats (Ghế): Lưu trạng thái ghế của từng phòng chiếu (trống, đã đặt, đã bán).
  - ShowTime (Suất chiếu): Lưu thông tin về thời gian chiếu phim, thời gian bắt đầu và kết thúc.
  - Customers (Khách hàng): Quản lý thông tin khách hàng, bao gồm tên, địa chỉ, số điện thoại, email.

- Employees (Nhân viên): Lưu trữ thông tin nhân viên của rạp, bao gồm lương và ngày sinh.
- Tickets (Vé): Lưu thông tin về vé, liên kết với khách hàng, suất chiếu và ghế.
- Invoices (Hóa đơn): Lưu thông tin hóa đơn, phương thức thanh toán.
- Promotions (Khuyến mãi): Lưu thông tin về các chương trình giảm giá.
- Counter (Quầy vé): Quản lý quầy bán vé và nhân viên thực hiện giao dịch.

3.1.2. Câu lệnh SQL tạo cơ sở dữ liệu

Dưới đây là các câu lệnh SQL dùng để tạo cơ sở dữ liệu và các bảng chính trong hệ thống "Quản lý rạp phim":

1. Tạo Database và sử dụng Database:

```
CREATE DATABASE QLHTBVXP;

USE QLHTBVXP;
```

2. Tạo bảng MovieGenre (Thể loại phim):

```
CREATE TABLE MovieGenre (
    MovieGenreID INT IDENTITY(1,1) PRIMARY KEY,
    NameGenre NVARCHAR(50)
);
```

3. Tạo bảng Movie (Phim):

```
CREATE TABLE Movie (
    MovieID INT IDENTITY(1,1) PRIMARY KEY,
    NameMovie NVARCHAR(100),
    Director NVARCHAR(100),
    MovieGenreID INT,
    FOREIGN KEY (MovieGenreID) REFERENCES MovieGenre(MovieGenreID)
);
```

4. Tạo bảng ScreenRoom (Phòng chiếu):

```
CREATE TABLE ScreenRoom (
    ScreenRoomID INT IDENTITY(1,1) PRIMARY KEY,
    NameRoom NVARCHAR(50),
    TotalSeat INT
);
```

5. Tạo bảng Seats (Ghế):

```
CREATE TABLE Seats (
    SeatID INT IDENTITY(1,1) PRIMARY KEY,
    ScreenRoomID INT,
    SeatNumber NVARCHAR(10),
    Status NVARCHAR(10) DEFAULT N'Trống',
    FOREIGN KEY (ScreenRoomID) REFERENCES ScreenRoom(ScreenRoomID),
    CONSTRAINT CHK_SeatStatus CHECK (Status IN (N'Trống', N'Đã đặt', N'Đã bán'))
);
```

6. Tạo bảng ShowTime (Suất chiếu):

```
CREATE TABLE ShowTime (
    ShowTimeID INT IDENTITY(1,1) PRIMARY KEY,
    BeginTime DATETIME,
    EndTime DATETIME,
    CapacityUtilization DECIMAL(5, 2)
);
```

7. Tạo bảng Promotions (Khuyến mãi):

```
CREATE TABLE Promotions (
    PromotionID INT IDENTITY(1,1) PRIMARY KEY,
    PromotionCode NVARCHAR(20),
    DiscountPercentage DECIMAL(5, 2),
    StartDate DATE,
    EndDate DATE,
    ApplicableTicketType NVARCHAR(50)
);
```

8. Tạo bảng Employees (Nhân viên):

```
CREATE TABLE Employees (
    EmployeeID INT IDENTITY(1,1) PRIMARY KEY,
    FullName NVARCHAR(100),
    Address NVARCHAR(255),
    DateOfBirth DATE,
    PhoneNumber NVARCHAR(20),
    Salary DECIMAL(10, 2)
);
```

9. Tạo bảng Customers (Khách hàng):

```
CREATE TABLE Customers (
    CustomerID INT IDENTITY(1,1) PRIMARY KEY,
    FullName NVARCHAR(100),
    Address NVARCHAR(255),
    DateOfBirth DATE,
    PhoneNumber NVARCHAR(20),
    Email NVARCHAR(100)
);
```

10. Tạo bảng Tickets (Vé):

```
CREATE TABLE Tickets (
    TicketID INT IDENTITY(1,1) PRIMARY KEY,
    Price DECIMAL(10, 2),
    EmployeeID INT,
    CustomerID INT,
    SeatID INT,
    MovieID INT,
    RoomID INT,
    ShowTimeID INT,
    TicketType NVARCHAR(50),
    TicketSaleDate DATE,
    PromotionID INT,
    FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID),
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),
    FOREIGN KEY (SeatID) REFERENCES Seats(SeatID),
    FOREIGN KEY (MovieID) REFERENCES Movie(MovieID),
    FOREIGN KEY (RoomID) REFERENCES ScreenRoom(ScreenRoomID),
    FOREIGN KEY (ShowTimeID) REFERENCES ShowTime(ShowTimeID),
    FOREIGN KEY (PromotionID) REFERENCES Promotions(PromotionID)
);
```

11. Tạo bảng Counter (Quầy vé):

```
CREATE TABLE Counter (
    CounterID INT IDENTITY(1,1) PRIMARY KEY,
    TicketID INT,
    EmployeeID INT,
    NameCounter NVARCHAR(50),
    FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID),
    FOREIGN KEY (TicketID) REFERENCES Tickets(TicketID)
);
```

12. Tạo bảng Invoices (Hóa đơn):

```
CREATE TABLE Invoices (
    InvoiceID INT IDENTITY(1,1) PRIMARY KEY,
    TicketID INT,
    CustomerID INT,
    Amount DECIMAL(10, 2),
    PaymentMethod NVARCHAR(20),
    InvoiceDate DATETIME,
    FOREIGN KEY (TicketID) REFERENCES Tickets(TicketID),
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),
    CONSTRAINT CHK_PaymentMethod CHECK (PaymentMethod IN (N'Tiền mặt', N'Thẻ', N'Ví điện tử'))
);
```

3.2. Chèn dữ liệu

3.2.1. Mục tiêu

Chèn dữ liệu mẫu vào hệ thống quản lý vé xem phim nhằm kiểm thử hoạt động của hệ thống trước khi đưa vào sử dụng thực tế. Dữ liệu bao gồm phim, khách hàng, vé, phòng chiếu, suất chiếu, nhân viên, khuyến mãi**...**

3.2.2. Phương pháp

Dữ liệu được chèn vào SQL Server thông qua Python sử dụng thư viện pyodbc. Chương trình Python kết nối với SQL Server, tạo dữ liệu giả lập và chèn vào các bảng theo đúng thứ tự.

- Công cụ sử dụng:

- SQL Server 2019
- Python 3.11.9
- Thư viện **pyodbc** để kết nối SQL Server
- Dữ liệu giả lập tạo bằng Python

3.2.3. Kết nối SQL Server từ python

```python
import pyodbc
import random
from datetime import datetime, timedelta
import re

Tabnine | Edit | Test | Explain | Document
def remove_accents(input_str):
    accents_map = str.maketrans(
        "àáảạãăằắẳặẵâầấẩậẫèéẻẹẽêềếểệễìíỉịĩòóỏọõôồốổộỗơờớởợỡùúủụũưừứửựữỳýỷỵỹđ",
        "aaaaaaaaaaaaaaaaaeeeeeeeeeeeiiiiioooooooooooooooooouuuuuuuuuuuyyyyyd"
    )
    return input_str.translate(accents_map).lower()

try:
    conn = pyodbc.connect(
        'DRIVER={ODBC Driver 17 for SQL Server};'
        'SERVER=TYANZUQ-2811\\SQLEXPRESS;'
        'DATABASE=QLHTBVXP;'
        'UID=sa;'
        'PWD=123456',
        autocommit=True
    )
    print("Kết nối thành công!")
except pyodbc.Error as e:
    print("Lỗi kết nối:", e)
    exit()
```

Trước tiên, các thư viện cần thiết như **pyodbc, random, datetime, re** được nhập vào. Sau đó, một hàm **remove_accents(input_str)** được định nghĩa để loại bỏ dấu tiếng Việt bằng cách sử dụng **str.maketrans()** và **translate()**, giúp chuyển đổi các ký tự có dấu thành không dấu. Tiếp theo, đoạn mã thiết lập kết nối với SQL Server bằng **pyodbc.connect()**, sử dụng thông tin:

- Driver: ODBC Driver 17 for SQL Server
- Server: TYANZUQ-2811\\SQLEXPRESS
- Database: QLHTBVXP
- User: sa
- Mật khẩu: 123456

Cờ **autocommit=True** được bật để tự động thực thi các thay đổi mà không cần **commit()**. Nếu kết nối thành công, chương trình in ra **"Kết nối thành công!"**. Nếu xảy ra lỗi, chương trình sẽ bắt ngoại lệ và in thông báo **"Lỗi kết nối"** kèm theo lỗi cụ thể, sau đó thoát chương trình bằng **exit().** Cuối cùng, con trỏ cur được khởi tạo từ kết nối để thực hiện các truy vấn SQL sau này.

3.2.4. Dữ liệu mẫu và chèn dữ liệu

1. Chuẩn bị dữ liệu mẫu:

```
ten_phim = [
    "Avengers: Endgame", "Titanic", "The Dark Knight", "Inception", "The Lion King",
    "Spider-Man: No Way Home", "Jurassic Park", "The Godfather", "Star Wars: Episode IV"
    "Frozen", "Parasite", "Interstellar", "The Matrix", "Forrest Gump", "Harry Potter",
    "Avatar", "Black Panther", "Joker", "Pulp Fiction", "The Shawshank Redemption",
    "Mad Max: Fury Road", "The Avengers", "Gladiator", "The Departed", "Toy Story",
    "The Empire Strikes Back", "Back to the Future", "The Silence of the Lambs",
    "Finding Nemo", "The Incredibles", "Shrek", "Die Hard", "Terminator 2",
    "Lord of the Rings", "The Hobbit", "Pirates of the Caribbean", "Deadpool",
    "Wonder Woman", "Skyfall", "The Wolf of Wall Street", "La La Land", "Coco",
    "Inside Out", "Up", "WALL-E", "The Grand Budapest Hotel", "Memento",
    "Fight Club", "Se7en", "No Country for Old Men"
]

the_loai_phim = [
    "Hành động", "Kinh dị", "Tình cảm", "Hài", "Khoa học viễn tưởng",
    "Phiêu lưu", "Hoạt hình", "Tội phạm", "Drama", "Giả tưởng",
    "Chiến tranh", "Lịch sử", "Thể thao", "Âm nhạc", "Bí ẩn"
]

ten_phong_chieu = [
    "Phòng 1", "Phòng 2", "Phòng 3", "Phòng 4", "Phòng 5", "Phòng 6", "Phòng 7",
    "Phòng VIP 1", "Phòng VIP 2", "Phòng VIP 3", "Phòng 3D 1", "Phòng 3D 2",
    "Phòng 4DX 1", "Phòng 4DX 2", "Phòng IMAX"
]

ho = ["Nguyễn", "Trần", "Lê", "Phạm", "Vũ", "Đặng", "Hoàng", "Hồ", "Ngô", "Dương"]
ten_dem = ["Quốc", "Anh", "Thế", "Đức", "Minh", "Văn", "Huy", "Khánh", "Trường", "Bảo"]
ten = ["Hải", "Tùng", "Thành", "Long", "Minh", "Quang", "Duy", "Khôi", "Hưng", "Vũ"]

ma_khuyen_mai = ["KM10", "KM20", "KM30", "KM50", "KMTET", "KMVIP", "KMWEEKEND",
                 "KMNEW", "KMFRI", "KMSUN", "KMSTUDENT", "KMGROUP"]
loai_ve = ["Thường", "VIP", "Couple", "Student", "3D", "4DX"]
```

2. Sinh dữ liệu:

```
def random_name():
    return random.choice(ho) + " " + random.choice(ten_dem) + " " + random.choice(ten)

Tabnine | Edit | Test | Explain | Document
def generate_email(name):
    name_no_accents = remove_accents(name)
    email_base = re.sub(r'\s+', '', name_no_accents)
    email = f"{email_base}{random.randint(1, 999)}@gmail.com"
    return email

Tabnine | Edit | Test | Explain | Document
def random_date(start_date, end_date):
    delta = end_date - start_date
    random_days = random.randint(0, delta.days)
    return start_date + timedelta(days=random_days)

Tabnine | Edit | Test | Explain | Document
def random_time():
    hour = random.randint(8, 22)
    minute = random.choice([0, 15, 30, 45])
    return f"{hour:02}:{minute:02}:00"
```

- **random_name():** Tạo một tên đầy đủ ngẫu nhiên bằng cách chọn ngẫu nhiên một họ từ danh sách **ho**, một tên đệm từ **ten_dem**, và một tên từ **ten**, sau đó kết hợp chúng thành chuỗi có khoảng trắng.
- **generate_email(name):** Tạo địa chỉ email ngẫu nhiên từ một tên cho trước bằng cách loại bỏ dấu tiếng Việt với **remove_accents**, xóa hết khoảng trắng bằng **re.sub**, thêm một số ngẫu nhiên từ 1 đến 999, rồi nối với **"@gmail.com"**.
- **random_date(start_date, end_date):** Tạo một ngày ngẫu nhiên nằm trong khoảng thời gian từ **start_date** đến **end_date**, tính khoảng cách bằng **delta**, chọn số ngày ngẫu nhiên với **random.randint**, rồi cộng vào **start_date**.
- **random_time():** Tạo thời gian ngẫu nhiên trong ngày, với giờ được chọn ngẫu nhiên từ 8 đến 22 bằng **random.randint**, phút chọn từ danh sách [0, 15, 30, 45], định dạng chuỗi thành **"HH:MM:00".**

3. Bảng Movie (phim)

```
def insert_movie():
    for i, movie in enumerate(ten_phim):
        genre_id = random.randint(1, len(the_loai_phim))
        director = random_name()
        sql = f'''
            INSERT INTO Movie (NameMovie, Director, MovieGenreID)
            VALUES (N'{movie}', N'{director}', {genre_id})
        '''
        cur.execute(sql)
    conn.commit()
```

Đoạn mã trên giúp tự động chèn danh sách phim vào cơ sở dữ liệu SQL Server bằng Python. Mỗi bộ phim được gán thể loại ngẫu nhiên và đạo diễn ngẫu nhiên trước khi được lưu vào bảng Movie. Câu lệnh **INSERT INTO** sử dụng **Unicode (**N'...'**)** để hỗ trợ tiếng Việt và ký tự đặc biệt. Sau khi thực thi lệnh SQL, **commit()** được gọi để lưu thay đổi. Nhờ đó, quá trình nhập dữ liệu được tự động hóa, tiết kiệm thời gian và giảm sai sót, đồng thời có thể mở rộng với dữ liệu từ API hoặc file CSV.

4. Bảng MovieGenre (thể loại phim)

```
def insert_movie_genre():
    for genre in the_loai_phim:
        sql = f'''
            INSERT INTO MovieGenre (NameGenre)
            VALUES (N'{genre}')
        '''
        cur.execute(sql)
    conn.commit()
```

Đoạn mã trên thực hiện chèn danh sách thể loại phim vào bảng MovieGenre trong SQL Server bằng Python. Chương trình sử dụng vòng lặp để duyệt qua từng thể loại phim trong danh sách **the_loai_phim**, sau đó tạo câu lệnh SQL **INSERT INTO** để thêm thể loại vào cơ sở dữ liệu. Ký tự **N'...'** được dùng để hỗ trợ Unicode, giúp lưu trữ chính xác tên thể loại phim có dấu. Cuối cùng, lệnh **commit()** được gọi để xác nhận thay đổi. Nhờ đó, dữ liệu thể loại phim được tự động nhập nhanh chóng, giảm sai sót và tiết kiệm thời gian so với việc nhập thủ công.

5. Bảng ScreenRoom (phòng chiếu)

```
def insert_screen_room():
    for i, room in enumerate(ten_phong_chieu):
        total_seat = random.randint(50, 150)
        sql = f'''
            INSERT INTO ScreenRoom (NameRoom, TotalSeat)
            VALUES (N'{room}', {total_seat})
        '''
        cur.execute(sql)
    conn.commit()
```

Đoạn mã trên giúp tự động chèn danh sách phòng chiếu vào bảng ScreenRoom trong SQL Server. Mỗi phòng chiếu trong danh sách **ten_phong_chieu** sẽ được gán một số ghế ngẫu nhiên trong khoảng từ 50 đến 150. Câu lệnh SQL **INSERT INTO** đảm bảo dữ liệu được lưu trữ chính xác, trong đó **N'{room}'** hỗ trợ Unicode để lưu tên phòng có dấu. Sau khi thực thi lệnh, **commit()** được gọi để xác nhận thay đổi. Nhờ đó, việc nhập dữ liệu phòng chiếu trở nên nhanh chóng, chính xác và giảm thiểu sai sót**.**

6. Bảng Seats (ghế ngồi)

```
def insert_seats():
    cur.execute("SELECT ScreenRoomID, TotalSeat FROM ScreenRoom")
    rooms = cur.fetchall()
    seat_id = 1
    for room in rooms:
        room_id, total_seat = room
        for seat_num in range(1, total_seat + 1):
            seat_number = f"{chr(65 + (seat_num - 1) // 10)}{(seat_num - 1) % 10 + 1}"
            sql = f'''
                INSERT INTO Seats (ScreenRoomID, SeatNumber, Status)
                VALUES ({room_id}, N'{seat_number}', 'Trống')
            '''
            cur.execute(sql)
            seat_id += 1
    conn.commit()
```

Đoạn mã trên giúp tự động tạo và chèn danh sách ghế ngồi cho từng phòng chiếu vào bảng Seats trong SQL Server. Mỗi phòng chiếu được lấy từ bảng ScreenRoom, sau đó ghế được đánh số theo hàng **(A, B, C,...)** và cột **(1-10)** để mô phỏng cách bố trí ghế trong rạp. Trạng thái ghế mặc định là **"Trống"**.

7. Bảng ShowTimes (thời gian chiếu)

```
def insert_show_time():
    start_date = datetime(2025, 3, 1)
    end_date = datetime(2025, 12, 31)
    for i in range(150):
        show_date = random_date(start_date, end_date)
        begin_time = random_time()
        end_time = (datetime.strptime(begin_time, "%H:%M:%S") + timedelta(hours=2)).strftime("%H:%M:%S")
        capacity = 0
        sql = f'''
            INSERT INTO ShowTime (BeginTime, EndTime, CapacityUtilization)
            VALUES ('{show_date.strftime('%Y-%m-%d')} {begin_time}',
                    '{show_date.strftime('%Y-%m-%d')} {end_time}', {capacity})
        '''
        cur.execute(sql)
    conn.commit()
```

Đoạn mã trên tự động tạo và chèn lịch chiếu phim vào bảng ShowTime trong SQL Server. Mỗi suất chiếu có thời gian kết thúc sau 2 giờ và **CapacityUtilization** mặc định là 0. Câu lệnh SQL **INSERT INTO** ShowTime giúp lưu dữ liệu vào cơ sở dữ liệu.

8. Bảng Employees và Customers (nhân viên và khách hàng)

```
def insert_employees():
    for i in range(50):
        name = random_name()
        address = f"Số {random.randint(1, 100)} Đường {random.choice(ten)}"
        dob = random_date(datetime(1980, 1, 1), datetime(2000, 12, 31)).strftime("%Y-%m-%d")
        phone = f"09{random.randint(10000000, 99999999)}"
        salary = random.randint(5000000, 20000000)
        sql = f'''
            INSERT INTO Employees (FullName, Address, DateOfBirth, PhoneNumber, Salary)
            VALUES (N'{name}', N'{address}', '{dob}', N'{phone}', {salary})
        '''
        cur.execute(sql)
    conn.commit()

Tabnine | Edit | Test | Explain | Document
def insert_customers():
    for i in range(500):
        name = random_name()
        address = f"Số {random.randint(1, 100)} Đường {random.choice(ten)}"
        dob = random_date(datetime(1980, 1, 1), datetime(2005, 12, 31)).strftime("%Y-%m-%d")
        phone = f"09{random.randint(10000000, 99999999)}"
        email = generate_email(name)
        sql = f'''
            INSERT INTO Customers (FullName, Address, DateOfBirth, PhoneNumber, Email)
            VALUES (N'{name}', N'{address}', '{dob}', N'{phone}', N'{email}')
        '''
        cur.execute(sql)
    conn.commit()
```

Đoạn mã trên được thiết kế để tạo dữ liệu ngẫu nhiên cho nhân viên và khách hàng trong hệ thống quản lý rạp phim. Trước tiên, hàm random_**name()** sinh ra tên đầy đủ bằng cách kết hợp ngẫu nhiên họ, tên đệm và tên từ danh sách có sẵn. Sau đó, hàm **generate_email()** sẽ tạo địa chỉ email bằng cách loại bỏ dấu trong tên và thêm số ngẫu nhiên để đảm bảo tính duy nhất.

Tiếp theo, hàm **insert_employees()** tạo thông tin cho 50 nhân viên với các thuộc tính như họ tên, địa chỉ, ngày sinh, số điện thoại và mức lương, sau đó thực hiện chèn dữ liệu vào bảng Employees. Tương tự, hàm **insert_customers()** thực hiện việc tạo và chèn dữ liệu cho khách hàng, nhưng có thêm thông tin về email. Các giá trị như địa chỉ và số điện thoại cũng được sinh ngẫu nhiên để mô phỏng dữ liệu thực tế.

9. Bảng Promotions (mã khuyến mại)

```
def insert_promotions():
    start_date = datetime(2025, 1, 1)
    end_date = datetime(2025, 12, 31)
    for i, code in enumerate(ma_khuyen_mai):
        discount = random.randint(10, 50)
        promo_start = random_date(start_date, end_date)
        promo_end = promo_start + timedelta(days=random.randint(7, 30))
        ticket_type = random.choice(loai_ve)
        sql = f'''
            INSERT INTO Promotions (PromotionCode, DiscountPercentage, StartDate, EndDate, ApplicableTicketType)
            VALUES (N'{code}', {discount}, '{promo_start.strftime('%Y-%m-%d')}',
                '{promo_end.strftime('%Y-%m-%d')}', N'{ticket_type}')
        '''
        cur.execute(sql)
    conn.commit()
```

Đoạn mã trên thực hiện tạo dữ liệu khuyến mãi ngẫu nhiên cho hệ thống quản lý vé xem phim. Danh sách **ma_khuyen_mai** chứa các mã khuyến mãi khác nhau. Hàm **insert_promotions()** sẽ tạo dữ liệu khuyến mãi với các thông tin như mã khuyến mãi, phần trăm giảm giá, ngày bắt đầu, ngày kết thúc và loại vé áp dụng**.**

Trong hàm**,** ngày bắt đầu được lấy ngẫu nhiên từ khoảng thời gian quy định, còn ngày kết thúc sẽ được tính dựa trên ngày bắt đầu cộng thêm từ 7 đến 30 ngày. Mức giảm giá được chọn ngẫu nhiên trong khoảng từ 10% đến 50%. Cuối cùng, loại vé áp dụng chương trình khuyến mãi được lấy từ danh sách có sẵn **(loai_ve)**.

10. Bảng Tickets và Invoices (vé phim và hóa đơn)

```
def insert_tickets_and_invoices():
    cur.execute("SELECT SeatID, ScreenRoomID FROM Seats WHERE Status = 'Trống'")
    seats = cur.fetchall()
    cur.execute("SELECT ShowTimeID FROM ShowTime")
    showtimes = cur.fetchall()
    cur.execute("SELECT MovieID FROM Movie")
    movies = cur.fetchall()
    cur.execute("SELECT CustomerID FROM Customers")
    customers = cur.fetchall()
    cur.execute("SELECT EmployeeID FROM Employees")
    employees = cur.fetchall()
    cur.execute("SELECT PromotionID FROM Promotions")
    promotions = cur.fetchall()

    customer_tickets = {cust[0]: 0 for cust in customers}
    max_tickets_per_customer = 5

    for i in range(1000):
        if not seats or all(customer_tickets[cust_id] >= max_tickets_per_customer for cust_id in customer_tickets):
            break
        seat = random.choice(seats)
        seats.remove(seat)
        showtime = random.choice(showtimes)
        movie = random.choice(movies)
        customer = random.choice([c for c in customers if customer_tickets[c[0]] < max_tickets_per_customer])
        customer_tickets[customer[0]] += 1
        employee = random.choice(employees)
        promotion = random.choice(promotions) if random.random() > 0.5 else None
        ticket_type = random.choice(loai_ve)
        price = random.randint(50000, 200000)
        if promotion:
            cur.execute(f"SELECT DiscountPercentage FROM Promotions WHERE PromotionID = {promotion[0]}")
            discount = cur.fetchone()[0]
            price = price * (1 - discount / 100)
        sale_date = random_date(datetime(2025, 3, 1), datetime(2025, 12, 31)).strftime("%Y-%m-%d")

        sql_ticket = f'''
            INSERT INTO Tickets (Price, EmployeeID, CustomerID, SeatID, MovieID, RoomID, ShowTimeID,
                                 TicketType, TicketSaleDate, PromotionID)
            VALUES ({price}, {employee[0]}, {customer[0]}, {seat[0]}, {movie[0]}, {seat[1]},
                    {showtime[0]}, N'{ticket_type}', '{sale_date}', {promotion[0] if promotion else 'NULL'})
        '''
        cur.execute(sql_ticket)

        cur.execute(f"UPDATE Seats SET Status = 'Đã bán' WHERE SeatID = {seat[0]}")
        cur.execute(f"SELECT COUNT(*) FROM Tickets WHERE ShowTimeID = {showtime[0]}")
        ticket_count = cur.fetchone()[0]
        cur.execute(f"SELECT TotalSeat FROM ScreenRoom WHERE ScreenRoomID = {seat[1]}")
        total_seat = cur.fetchone()[0]
        capacity = (ticket_count / total_seat) * 100 if total_seat > 0 else 0
        cur.execute(f"UPDATE ShowTime SET CapacityUtilization = {capacity} WHERE ShowTimeID = {showtime[0]}")
        payment_method = random.choice(['Tiền mặt', 'Thẻ', 'Ví điện tử'])
        sql_invoice = f'''
            INSERT INTO Invoices (TicketID, CustomerID, Amount, PaymentMethod, InvoiceDate)
            VALUES ({i + 1}, {customer[0]}, {price}, N'{payment_method}', '{sale_date}')
        '''
        cur.execute(sql_invoice)

        if random.random() > 0.5:
            sql_counter = f'''
                INSERT INTO Counter (TicketID, EmployeeID, NameCounter)
                VALUES ({i + 1}, {employee[0]}, N'Quầy {random.randint(1, 5)}')
            '''
            cur.execute(sql_counter)
    conn.commit()
```

Trước tiên, các danh sách dữ liệu cần thiết như ghế trống, suất chiếu, phim, khách hàng, nhân viên và khuyến mãi được truy vấn từ cơ sở dữ liệu. Tiếp theo, một từ điển **customer_tickets** được tạo để kiểm soát số lượng vé tối đa mỗi khách hàng có thể mua, với giá trị mặc định là 5 vé/người.

Trong vòng lặp chính, hệ thống sẽ thực hiện:

- Chọn một ghế ngẫu nhiên từ danh sách ghế trống, sau đó xóa khỏi danh sách để tránh trùng lặp.
- Chọn suất chiếu ngẫu nhiên, đảm bảo không vượt quá số vé giới hạn cho mỗi khách hàng.
- Xác định giá vé, nhân viên bán vé và khách hàng mua vé.
- Áp dụng khuyến mãi ngẫu nhiên (nếu có) bằng cách kiểm tra bảng Promotions.
- Thực hiện chèn dữ liệu vào bảng Tickets, đánh dấu ghế đã được đặt.
- Cập nhật số lượng ghế còn trống trong suất chiếu để đảm bảo tính toàn vẹn dữ liệu.
- Tạo hóa đơn tương ứng cho vé vừa đặt, bao gồm phương thức thanh toán và ngày mua.

Cuối cùng, dữ liệu quầy bán vé (**Counter**) cũng được chèn vào để ghi nhận nhân viên thực hiện giao dịch. Mã này giúp tự động hóa việc tạo dữ liệu thử nghiệm, hỗ trợ kiểm thử chức năng mua vé và thanh toán trong hệ thống.

3.3. In bảng dữ liệu

1. Dữ liệu bảng phim

	MovieID	NameMovie	Director	MovieGenreID
1	1	Avengers: Endgame	Lê Văn Long	5
2	2	Titanic	Trần Quốc Long	6
3	3	The Dark Knight	Nguyễn Minh Vũ	3
4	4	Inception	Hồ Anh Vũ	10
5	5	The Lion King	Vũ Quốc Khôi	12
6	6	Spider-Man: No Way Home	Hoàng Đức Quang	4
7	7	Jurassic Park	Ngô Quốc Thành	15
8	8	The Godfather	Lê Đức Thành	7
9	9	Star Wars: Episode IV	Hồ Minh Hưng	2
10	10	Frozen	Lê Trường Duy	7
11	11	Parasite	Vũ Thế Vũ	9
12	12	Interstellar	Hồ Trường Minh	9
13	13	The Matrix	Ngô Huy Long	14
14	14	Forrest Gump	Đặng Đức Minh	1
15	15	Harry Potter	Hoàng Khánh Long	1
16	16	Avatar	Hồ Anh Khôi	11
17	17	Black Panther	Trần Anh Minh	13
18	18	Joker	Dương Khánh Duy	3
19	19	Pulp Fiction	Lê Huy Thành	12
20	20	The Shawshank Redempt...	Dương Anh Quang	10

2. Dữ liệu bảng thể loại phim

	MovieGenreID	NameGenre
1	1	Hành động
2	2	Kinh dị
3	3	Tình cảm
4	4	Hài
5	5	Khoa học viễn tưởng
6	6	Phiêu lưu
7	7	Hoạt hình
8	8	Tội phạm
9	9	Drama
10	10	Giả tưởng
11	11	Chiến tranh
12	12	Lịch sử
13	13	Thể thao
14	14	Âm nhạc
15	15	Bí ẩn
16	16	Hành động
17	17	Kinh dị

3. Dữ liệu bảng phòng chiếu

	ScreenRoomID	NameRoom	TotalSeat
1	1	Phòng 1	94
2	2	Phòng 2	124
3	3	Phòng 3	61
4	4	Phòng 4	79
5	5	Phòng 5	76
6	6	Phòng 6	52
7	7	Phòng 7	79
8	8	Phòng VIP 1	133
9	9	Phòng VIP 2	72
10	10	Phòng VIP 3	117
11	11	Phòng 3D 1	137
12	12	Phòng 3D 2	95
13	13	Phòng 4DX 1	150
14	14	Phòng 4DX 2	94
15	15	Phòng IMAX	149
16	16	Phòng 1	138
17	17	Phòng 2	73

4. Dữ liệu bảng ghế ngồi

	SeatID	ScreenRoomID	SeatNumber	Status
1	1	1	A1	Đã bán
2	2	1	A2	Đã bán
3	3	1	A3	Đã bán
4	4	1	A4	Đã bán
5	5	1	A5	Đã bán

5. Dữ liệu bảng thời gian chiếu

	ShowTimeID	BeginTime	EndTime	CapacityUtilization
1	1	2025-06-18 13:00:00.000	2025-06-18 15:00:00.000	85.00
2	2	2025-08-13 13:15:00.000	2025-08-13 15:15:00.000	13.89
3	3	2025-09-09 16:00:00.000	2025-09-09 18:00:00.000	6.71
4	4	2025-09-12 12:30:00.000	2025-09-12 14:30:00.000	7.69
5	5	2025-06-10 19:15:00.000	2025-06-10 21:15:00.000	9.56
6	6	2025-12-08 11:45:00.000	2025-12-08 13:45:00.000	4.79
7	7	2025-11-15 08:45:00.000	2025-11-15 10:45:00.000	4.67
8	8	2025-05-09 09:00:00.000	2025-05-09 11:00:00.000	5.48
9	9	2025-10-17 14:00:00.000	2025-10-17 16:00:00.000	12.50
10	10	2025-11-10 17:15:00.000	2025-11-10 19:15:00.000	8.72
11	11	2025-11-30 21:15:00.000	2025-11-30 23:15:00.000	13.64
12	12	2025-09-25 18:15:00.000	2025-09-25 20:15:00.000	6.84
13	13	2025-03-19 21:45:00.000	2025-03-19 23:45:00.000	6.62
14	14	2025-07-11 20:45:00.000	2025-07-11 22:45:00.000	9.80
15	15	2025-04-09 19:45:00.000	2025-04-09 21:45:00.000	7.38
16	16	2025-09-27 22:30:00.000	2025-09-27 00:30:00.000	7.97

6. Dữ liệu bảng khuyến mãi

	PromotionID	PromotionCode	DiscountPercentage	StartDate	EndDate	ApplicableTicketType
1	1	KM10	23.00	2025-02-25	2025-04-08	3D
2	2	KM20	11.00	2025-08-19	2025-09-10	Couple
3	3	KM30	32.00	2025-10-09	2025-10-29	Thường
4	4	KM50	28.00	2025-03-17	2025-04-10	VIP
5	5	KMTET	50.00	2025-07-18	2025-07-29	4DX
6	6	KMVIP	17.00	2025-03-29	2025-04-08	VIP
7	7	KMWEEKEND	34.00	2025-06-05	2025-06-23	Couple
8	8	KMNEW	46.00	2025-07-28	2025-08-25	Couple
9	9	KMFRI	49.00	2025-07-15	2025-08-05	VIP
10	10	KMSUN	45.00	2025-12-25	2026-01-21	3D
11	11	KMSTUDENT	48.00	2025-11-12	2025-12-08	4DX
12	12	KMGROUP	46.00	2025-10-19	2025-11-05	VIP

7. Dữ liệu bảng nhân viên

	EmployeeID	FullName	Address	DateOfBirth	PhoneNumber	Salary
1	1	Phạm Trường Duy	Số 31 Đường Khôi	1985-08-15	0912155699	14748141.00
2	2	Hoàng Thế Minh	Số 12 Đường Vũ	1987-02-25	0968920424	8978285.00
3	3	Lê Đức Thành	Số 91 Đường Long	1983-07-03	0986980455	10489085.00
4	4	Hoàng Huy Quang	Số 40 Đường Hưng	1980-05-26	0980117047	10917408.00
5	5	Ngô Huy Long	Số 12 Đường Hải	1991-07-27	0936463187	7040254.00
6	6	Ngô Quốc Long	Số 70 Đường Vũ	1997-04-30	0994754946	13798088.00
7	7	Trần Huy Minh	Số 97 Đường Tùng	1997-09-16	0974714917	14269147.00
8	8	Vũ Bảo Vũ	Số 58 Đường Minh	1980-06-10	0983263848	13176281.00
9	9	Hồ Bảo Hải	Số 96 Đường Hưng	1982-11-29	0981452833	17086481.00
10	10	Hồ Trường Quang	Số 68 Đường Tùng	2000-02-29	0983384923	6480432.00
11	11	Nguyễn Thế Khôi	Số 91 Đường Quang	1996-10-02	0915423332	15311544.00
12	12	Dương Khánh Thành	Số 13 Đường Tùng	1981-03-17	0983870175	11226027.00
13	13	Vũ Thế Duy	Số 99 Đường Minh	1980-08-18	0973356636	17325135.00
14	14	Đặng Anh Minh	Số 84 Đường Thành	1982-01-24	0954108663	11430103.00
15	15	Phạm Khánh Hải	Số 72 Đường Minh	1990-09-19	0956252204	18412665.00
16	16	Nguyễn Khánh Hải	Số 59 Đường Long	1990-07-12	0967117485	6883697.00
17	17	Hoàng Huy Tùng	Số 6 Đường Vũ	1987-09-22	0973204056	6431538.00
18	18	Hồ Thế Hải	Số 100 Đường Hưng	2000-04-29	0960633245	15009534.00

8. Dữ liệu bảng khách hàng

	CustomerID	FullName	Address	DateOfBirth	PhoneNumber	Email
1	1	Hồ Văn Minh	Số 77 Đường Vũ	1989-02-13	0938666257	hovanminh768@gmail.com
2	2	Phạm Minh Thành	Số 53 Đường Hưng	1993-04-12	0982180444	phamminhthanh514@gmail.com
3	3	Nguyễn Khánh Long	Số 84 Đường Hưng	1991-07-04	0936076797	nguyenkhanhlong352@gmail.com
4	4	Dương Anh Duy	Số 59 Đường Hải	1996-04-25	0916926252	duonganhduy901@gmail.com
5	5	Dương Bảo Tùng	Số 83 Đường Long	1982-08-26	0933427403	duongbaotung638@gmail.com
6	6	Trần Minh Long	Số 62 Đường Vũ	1990-01-06	0925273939	tranminhlong892@gmail.com
7	7	Lê Trường Duy	Số 45 Đường Vũ	2003-01-22	0923966878	letruongduy538@gmail.com
8	8	Lê Quốc Long	Số 87 Đường Minh	1980-02-04	0917526472	lequoclong874@gmail.com
9	9	Ngô Bảo Quang	Số 75 Đường Quang	1989-10-28	0999223334	ngobaoquang316@gmail.com
10	10	Phạm Quốc Tùng	Số 92 Đường Tùng	1983-10-16	0973378714	phamquoctung687@gmail.com
11	11	Đặng Minh Duy	Số 99 Đường Khôi	1987-07-08	0963568814	đangminhduy663@gmail.com
12	12	Hoàng Trường Hải	Số 95 Đường Vũ	1981-08-17	0982594642	hoangtruonghai247@gmail.com
13	13	Vũ Thế Minh	Số 79 Đường Quang	1994-03-11	0970843784	vutheminh912@gmail.com
14	14	Dương Trường Khôi	Số 21 Đường Long	1990-07-31	0990063949	duongtruongkhoi76@gmail.com
15	15	Hồ Đức Khôi	Số 7 Đường Hải	1992-02-06	0987603427	hoduckhoi857@gmail.com
16	16	Nguyễn Bảo Vũ	Số 37 Đường Thành	1982-09-30	0938343654	nguyenbaovu111@gmail.com
17	17	Nguyễn Đức Quang	Số 60 Đường Tùng	1992-06-12	0945302362	nguyenđucquang148@gmail.com
18	18	Phạm Anh Hải	Số 87 Đường Long	2004-11-21	0946767653	phamanhhai369@gmail.com

9. Dữ liệu bảng vé phim

	TicketID	Price	EmployeeID	CustomerID	SeatID	MovieID	RoomID	ShowTimeID	TicketType	TicketSaleDate	PromotionID
1	1	66606.17	8	1	1220	19	13	82	VIP	2025-04-16	1
2	2	130000.00	28	146	1337	3	14	116	Student	2025-07-01	NULL
3	3	140000.00	7	208	480	35	6	119	VIP	2025-12-05	NULL
4	4	59039.00	42	9	800	19	10	102	Thường	2025-06-20	NULL
5	5	143155.61	22	207	569	10	8	145	3D	2025-11-27	2
6	6	94539.21	19	361	1474	34	15	25	4DX	2025-10-09	9
7	7	35130.48	8	79	761	30	9	59	3D	2025-06-27	7
8	8	151486.00	15	313	184	37	2	41	Thường	2025-10-09	NULL
9	9	132604.78	26	273	663	25	8	94	4DX	2025-08-27	1
10	10	132887.00	20	94	1274	34	14	84	VIP	2025-10-24	NULL
11	11	98352.00	39	351	1145	5	13	116	4DX	2025-12-22	5
12	12	141402.00	23	76	234	3	3	52	Couple	2025-08-18	NULL

10. Bảng dữ liệu quầy bán vé

	CounterID	TicketID	EmployeeID	NameCounter
1	1	2	28	Quầy 3
2	2	3	7	Quầy 4
3	3	5	22	Quầy 5
4	4	6	19	Quầy 1
5	5	7	8	Quầy 4
6	6	9	26	Quầy 5
7	7	13	27	Quầy 1
8	8	14	46	Quầy 3
9	9	17	17	Quầy 2
10	10	19	4	Quầy 2

11. Bảng dữ liệu hóa đơn

	InvoiceID	TicketID	CustomerID	Amount	PaymentMethod	InvoiceDate
1	1	1	1	86501.52	Ví điện tử	2025-04-16 00:00:00.000
2	3	3	208	140000.00	Thẻ	2025-12-05 00:00:00.000
3	4	4	9	59039.00	Thẻ	2025-06-20 00:00:00.000
4	5	5	207	143155.61	Thẻ	2025-11-27 00:00:00.000
5	6	6	361	94539.21	Thẻ	2025-10-09 00:00:00.000
6	7	7	79	35130.48	Thẻ	2025-06-27 00:00:00.000
7	8	8	313	151486.00	Tiền mặt	2025-10-09 00:00:00.000
8	9	9	273	132604.78	Tiền mặt	2025-08-27 00:00:00.000
9	10	10	94	132887.00	Ví điện tử	2025-10-24 00:00:00.000
10	11	11	351	98352.00	Tiền mặt	2025-12-22 00:00:00.000
11	12	12	76	141402.00	Tiền mặt	2025-08-18 00:00:00.000
12	13	13	117	185305.00	Ví điện tử	2025-12-27 00:00:00.000

CHƯƠNG 4. XÂY DỰNG CÁC VIEW

4.1. View 1: MovieWithGenre

- Mục đích: Liệt kê tất cả phim cùng với thể loại tương ứng, giúp người dùng dễ dàng tìm kiếm phim theo thể loại.

```
CREATE VIEW MovieWithGenre AS
SELECT m.NameMovie, mg.NameGenre
FROM Movie m
JOIN MovieGenre mg ON m.MovieGenreID = mg.MovieGenreID;
SELECT * FROM MovieWithGenre;
```

- Kết quả

	NameMovie	NameGenre
1	Avengers: Endgame	Khoa học viễn tưởng
2	Titanic	Phiêu lưu
3	The Dark Knight	Tình cảm
4	Inception	Giả tưởng
5	The Lion King	Lịch sử
6	Spider-Man: No Way Home	Hài
7	Jurassic Park	Bí ẩn
8	The Godfather	Hoạt hình
9	Star Wars: Episode IV	Kinh dị
10	Frozen	Hoạt hình
11	Parasite	Drama
12	Interstellar	Drama
13	The Matrix	Âm nhạc
14	Forrest Gump	Hành động
15	Harry Potter	Hành động
16	Avatar	Chiến tranh
17	Black Panther	Thể thao
18	Joker	Tình cảm

4.2. View 2: RoomWithSeats

- Mục đích: Hiển thị thông tin cơ bản về phòng chiếu (tên phòng và tổng số ghế), hỗ trợ quản lý cơ sở vật chất.

```
CREATE VIEW RoomWithSeats AS
SELECT NameRoom, TotalSeat
FROM ScreenRoom;

SELECT * FROM RoomWithSeats
```

- Kết quả

	NameRoom	TotalSe
1	Phòng 1	94
2	Phòng 2	124
3	Phòng 3	61
4	Phòng 4	79
5	Phòng 5	76
6	Phòng 6	52
7	Phòng 7	79
8	Phòng VIP 1	133
9	Phòng VIP 2	72
10	Phòng VIP 3	117
11	Phòng 3D 1	137
12	Phòng 3D 2	95
13	Phòng 4DX 1	150
14	Phòng 4DX 2	94
15	Phòng IMAX	149
16	Phòng 1	138

4.3. View 3: SeatStatus

- Mục đích: Cung cấp trạng thái của từng ghế trong từng phòng (trống, đã đặt, đã bán), hỗ trợ bán vé và kiểm tra chỗ ngồi.

```
CREATE VIEW SeatStatus AS
SELECT s.SeatNumber, sr.NameRoom, s.Status
FROM Seats s
JOIN ScreenRoom sr ON s.ScreenRoomID = sr.ScreenRoomID;

SELECT * FROM SeatStatus
```

- Kết quả

46	E6	Phòng 1	Đã bán
47	E7	Phòng 1	Đã bán
48	E8	Phòng 1	Đã bán
49	E9	Phòng 1	Đã bán
50	E10	Phòng 1	Đã bán
51	F1	Phòng 1	Đã bán
52	F2	Phòng 1	Đã bán
53	F3	Phòng 1	Đã bán
54	F4	Phòng 1	Đã bán
55	F5	Phòng 1	Đã bán
56	F6	Phòng 1	Đã bán
57	F7	Phòng 1	Đã bán
58	F8	Phòng 1	Đã bán

4.4. View 4: ShowTimesWithMovieAndRoom

- Mục đích: Liệt kê lịch chiếu với thông tin phim và phòng chiếu, dùng để hiển thị lịch chiếu cho khách hàng hoặc lập kế hoạch.

```
CREATE VIEW ShowTimesWithMovieAndRoom AS
SELECT DISTINCT st.BeginTime, st.EndTime, m.NameMovie, sr.NameRoom
FROM ShowTime st
JOIN Tickets t ON st.ShowTimeID = t.ShowTimeID
JOIN Movie m ON t.MovieID = m.MovieID
JOIN ScreenRoom sr ON t.RoomID = sr.ScreenRoomID;

SELECT * FROM ShowTimesWithMovieAndRoom
```

- Kết quả

	BeginTime	EndTime	NameMovie	NameRoom
1	2025-03-02 14:15:00.000	2025-03-02 16:15:00.000	Coco	Phòng 4DX 2
2	2025-03-02 14:15:00.000	2025-03-02 16:15:00.000	Coco	Phòng IMAX
3	2025-03-02 14:15:00.000	2025-03-02 16:15:00.000	Finding Nemo	Phòng IMAX
4	2025-03-02 14:15:00.000	2025-03-02 16:15:00.000	Joker	Phòng VIP 3
5	2025-03-02 14:15:00.000	2025-03-02 16:15:00.000	Lord of the Rings	Phòng VIP 3
6	2025-03-02 14:15:00.000	2025-03-02 16:15:00.000	Se7en	Phòng VIP 1
7	2025-03-02 14:15:00.000	2025-03-02 16:15:00.000	The Dark Knight	Phòng 3
8	2025-03-02 14:15:00.000	2025-03-02 16:15:00.000	The Godfather	Phòng 2
9	2025-03-02 14:15:00.000	2025-03-02 16:15:00.000	The Grand Budapest Hotel	Phòng 1
10	2025-03-02 14:15:00.000	2025-03-02 16:15:00.000	The Silence of the Lambs	Phòng 1
11	2025-03-02 14:15:00.000	2025-03-02 16:15:00.000	The Wolf of Wall Street	Phòng 4
12	2025-03-02 14:15:00.000	2025-03-02 16:15:00.000	Titanic	Phòng 3D 1

4.5. View 5: ActivePromotions

- Mục đích: Liệt kê các chương trình khuyến mãi đang hoạt động dựa trên ngày hiện tại, hỗ trợ quản lý và áp dụng khuyến mãi.

```
CREATE VIEW ActivePromotions AS
SELECT *
FROM Promotions
WHERE GETDATE() BETWEEN StartDate AND EndDate;

SELECT * FROM ActivePromotions
```

- Kết quả

	PromotionID	PromotionCode	DiscountPercentage	StartDate	EndDate	ApplicableTicketType
1	1	KM10	23.00	2025-02-25	2025-04-08	3D
2	25	KMNEW2025	15.00	2025-03-15	2025-04-15	Thường

4.6. View 6: TicketDetails

- Mục đích: Hiển thị thông tin chi tiết của từng vé đã bán (khách hàng, phim, suất chiếu, ghế), hỗ trợ kiểm tra và dịch vụ khách hàng.

```
CREATE VIEW TicketDetails AS
SELECT t.TicketID, c.FullName as CustomerName, m.NameMovie, st.BeginTime as ShowTime, s.SeatNumber
FROM Tickets t
JOIN Customers c ON t.CustomerID = c.CustomerID
JOIN Movie m ON t.MovieID = m.MovieID
JOIN ShowTime st ON t.ShowTimeID = t.ShowTimeID
JOIN Seats s ON t.SeatID = s.SeatID;

SELECT * FROM TicketDetails
```

- Kết quả

	TicketID	CustomerName	NameMovie	ShowTime	SeatNumber
1	2001	Hồ Văn Minh	Avengers: Endgame	2025-06-18 13:00:00.000	A2
2	2001	Hồ Văn Minh	Avengers: Endgame	2025-08-13 13:15:00.000	A2
3	2001	Hồ Văn Minh	Avengers: Endgame	2025-09-09 16:00:00.000	A2
4	2001	Hồ Văn Minh	Avengers: Endgame	2025-09-12 12:30:00.000	A2
5	2001	Hồ Văn Minh	Avengers: Endgame	2025-06-10 19:15:00.000	A2
6	2001	Hồ Văn Minh	Avengers: Endgame	2025-12-08 11:45:00.000	A2
7	2001	Hồ Văn Minh	Avengers: Endgame	2025-11-15 08:45:00.000	A2
8	2001	Hồ Văn Minh	Avengers: Endgame	2025-05-09 09:00:00.000	A2
9	2001	Hồ Văn Minh	Avengers: Endgame	2025-10-17 14:00:00.000	A2
10	2001	Hồ Văn Minh	Avengers: Endgame	2025-11-10 17:15:00.000	A2
11	2001	Hồ Văn Minh	Avengers: Endgame	2025-11-30 21:15:00.000	A2
12	2001	Hồ Văn Minh	Avengers: Endgame	2025-09-25 18:15:00.000	A2

4.7. View 7: RevenuePerMovie

- Mục đích: Tính tổng doanh thu từ vé bán cho từng phim, hỗ trợ phân tích tài chính và đánh giá hiệu quả phim.

```
CREATE VIEW RevenuePerMovie AS
SELECT m.NameMovie, SUM(t.Price) as TotalRevenue
FROM Movie m
JOIN Tickets t ON m.MovieID = t.MovieID
GROUP BY m.NameMovie;

SELECT * FROM RevenuePerMovie
```

- Kết quả

	NameMovie	TotalRevenue
1	Avatar	4391065.14
2	Avengers: Endgame	4873594.97
3	Back to the Future	3887659.75
4	Black Panther	3009767.40
5	Coco	6247124.08
6	Deadpool	4002105.56
7	Die Hard	4377616.46
8	Fight Club	3888615.37
9	Finding Nemo	3349157.76
10	Forrest Gump	3625103.22
11	Frozen	3250239.74
12	Gladiator	3966617.46
13	Harry Potter	2692601.22
14	Inception	4075862.06
15	Inside Out	3222015.12
16	Interstellar	5343438.32

4.8. View 8: TicketsPerShowTime

- Mục đích: Đếm số lượng vé bán cho mỗi suất chiếu, bao gồm cả suất không có vé (số 0), giúp đánh giá hiệu suất suất chiếu.

```
CREATE VIEW TicketsPerShowTime AS
SELECT st.ShowTimeID, COUNT(t.TicketID) as TicketCount
FROM ShowTime st
LEFT JOIN Tickets t ON st.ShowTimeID = t.ShowTimeID
GROUP BY st.ShowTimeID;

SELECT * FROM
```

- Kết quả

	ShowTimeID	TicketCount
1	1	8
2	2	11
3	3	10
4	4	9
5	5	13
6	6	7
7	7	7
8	8	8
9	9	9
10	10	13
11	11	12
12	12	8

4.9. View 9: RecentCustomers

- Mục đích: Liệt kê thông tin khách hàng đã mua vé trong vòng 1 tháng gần nhất, hỗ trợ quản lý khách hàng thân thiết và khuyến mãi.

```
CREATE VIEW RecentCustomers AS
SELECT c.*
FROM Customers c
JOIN Tickets t ON c.CustomerID = t.CustomerID
WHERE t.TicketSaleDate >= DATEADD(MONTH, -1, GETDATE());

SELECT * FROM RecentCustomers
```

- Kết quả

	CustomerID	FullName	Address	DateOfBirth	PhoneNumber	Email
1	1	Hồ Văn Minh	Số 77 Đường Vũ	1989-02-13	0938666257	hovanminh768@gmail.com
2	1	Hồ Văn Minh	Số 77 Đường Vũ	1989-02-13	0938666257	hovanminh768@gmail.com
3	1	Hồ Văn Minh	Số 77 Đường Vũ	1989-02-13	0938666257	hovanminh768@gmail.com
4	1	Hồ Văn Minh	Số 77 Đường Vũ	1989-02-13	0938666257	hovanminh768@gmail.com
5	1	Hồ Văn Minh	Số 77 Đường Vũ	1989-02-13	0938666257	hovanminh768@gmail.com
6	2	Phạm Minh Thành	Số 53 Đường Hưng	1993-04-12	0982180444	phamminhthanh514@gmail.com
7	2	Phạm Minh Thành	Số 53 Đường Hưng	1993-04-12	0982180444	phamminhthanh514@gmail.com
8	2	Phạm Minh Thành	Số 53 Đường Hưng	1993-04-12	0982180444	phamminhthanh514@gmail.com
9	2	Phạm Minh Thành	Số 53 Đường Hưng	1993-04-12	0982180444	phamminhthanh514@gmail.com
10	3	Nguyễn Khánh Long	Số 84 Đường Hưng	1991-07-04	0936076797	nguyenkhanhlong352@gmail.com

4.10. View 10: ShowTimeDetails

- Mục đích: Cung cấp chi tiết về mỗi suất chiếu (phim, phòng, tổng số ghế, số vé đã bán), hỗ trợ phân tích hiệu suất và lập kế hoạch.

```
CREATE VIEW ShowTimeDetails AS
SELECT st.ShowTimeID, m.NameMovie, sr.NameRoom, sr.TotalSeat, COUNT(t.TicketID) as TicketsSold
FROM ShowTime st
JOIN Tickets t ON st.ShowTimeID = t.ShowTimeID
JOIN Movie m ON t.MovieID = m.MovieID
JOIN ScreenRoom sr ON t.RoomID = sr.ScreenRoomID
GROUP BY st.ShowTimeID, m.NameMovie, sr.NameRoom, sr.TotalSeat;

SELECT * FROM ShowTimeDetails
```

- Kết quả

	ShowTimeID	NameMovie	NameRoom	TotalSeat	TicketsSold
1	1	Avengers: Endgame	Phòng 1	94	3
2	1	Joker	Phòng VIP 1	130	1
3	1	Jurassic Park	Phòng 3	147	1
4	1	Star Wars: Episode IV	Phòng VIP 3	117	1
5	1	The Lion King	Phòng 4	79	1
6	1	Wonder Woman	Phòng 4	79	1
7	2	Avatar	Phòng 4DX 2	94	1
8	2	Die Hard	Phòng 3D 1	137	1
9	2	Inside Out	Phòng 4DX 1	72	1
10	2	No Country for Old Men	Phòng 3D 1	137	1
11	2	Parasite	Phòng 4DX 1	150	1
12	2	Pulp Fiction	Phòng 3	61	1
13	2	The Godfather	Phòng 4	79	1
14	2	The Godfather	Phòng 4DX 1	150	1
15	2	The Incredibles	Phòng 3D 2	95	1
16	2	The Matrix	Phòng 4	79	1
17	2	Titanic	Phòng 2	124	1
18	3	Coco	Phòng 2	124	1

CHƯƠNG 5. XÂY DỰNG CÁC PROCEDURE

5.1. Procedure 1: Thêm khách hàng mới và kiểm tra email hợp lệ

- Mục đích: Stored Procedure sp_AddCustomer dùng để thêm khách hàng vào bảng Customers, kiểm tra định dạng email hợp lệ trước khi chèn dữ liệu. Nếu email sai, sẽ báo lỗi và dừng lại, nếu đúng thì thêm khách hàng và hiển thị thông báo thành công

```
CREATE PROCEDURE sp_AddCustomer
    @FullName NVARCHAR(100),
    @Address NVARCHAR(255),
    @DateOfBirth DATE,
    @PhoneNumber NVARCHAR(20),
    @Email NVARCHAR(100)
AS
BEGIN
    IF @Email NOT LIKE '%_@__%.__%' -- Kiểm tra định dạng email cơ bản
    BEGIN
        RAISERROR ('Email không hợp lệ!', 16, 1);
        RETURN;
    END
    INSERT INTO Customers (FullName, Address, DateOfBirth, PhoneNumber, Email)
    VALUES (@FullName, @Address, @DateOfBirth, @PhoneNumber, @Email);
    PRINT 'Thêm khách hàng thành công!';
END;

EXEC sp_AddCustomer
    @FullName = N'Nguyễn Quốc Phong',
    @Address = N'Số 99 Đường Nguyễn Trãi',
    @DateOfBirth = '1995-08-20',
    @PhoneNumber = '0941234567',
    @Email = 'nguyenquocphong@gmail.com';
```

- Kết quả

```
(1 row affected)
Thêm khách hàng thành công!

Completion time: 2025-03-15T12:22:37.3983919+07:00
```

5.2. Procedure 2: Cập nhật trạng thái ghế sau khi đặt vé

- Mục đích: Stored Procedure này dùng để cập nhật trạng thái ghế trong rạp chiếu phim khi khách hàng đặt vé hoặc hủy vé. Nó nhận vào SeatID và Status để thay đổi trạng thái ghế. Nếu SeatID không tồn tại trong hệ thống, thủ tục sẽ báo lỗi để tránh cập nhật sai dữ liệu. Nếu cập nhật thành công, hiển thị thông báo xác nhận.

```
CREATE PROCEDURE sp_UpdateSeatStatus
    @SeatID INT,
    @Status VARCHAR(10)
AS
BEGIN
    UPDATE Seats
    SET Status = @Status
    WHERE SeatID = @SeatID;
    IF @@ROWCOUNT = 0
        RAISERROR ('Không tìm thấy ghế với SeatID đã cho!', 16, 1);
    ELSE
        PRINT 'Cập nhật trạng thái ghế thành công!';
END;

EXEC sp_UpdateSeatStatus
    @SeatID = 1,
    @Status = 'Đã đặt';
```

- Kết quả

```
(1 row affected)
Cập nhật trạng thái ghế thành công!
```

5.3. Procedure 3: Tính tổng doanh thu theo ngày

- Mục đích: Thủ tục này giúp tính tổng doanh thu của rạp chiếu phim trong một ngày cụ thể. Nó truy vấn bảng Invoices để lấy tổng số tiền (Amount) từ các hóa đơn có ngày lập trùng với ngày được truyền vào (SaleDate). Điều này giúp quản lý rạp theo dõi doanh thu hàng ngày một cách chính xác.

```
CREATE PROCEDURE sp_GetRevenueByDate
    @SaleDate DATE
AS
BEGIN
    SELECT SUM(Amount) AS TotalRevenue
    FROM Invoices
    WHERE CAST(InvoiceDate AS DATE) = @SaleDate;
END;

EXEC sp_GetRevenueByDate
@SaleDate = '2025-03-15';
```

- Kết quả

	TotalRevenue
1	103540.58

5.4. Procedure 4: Áp dụng khuyến mại cho vé dựa trên loại vé

- Mục đích: Thủ tục này áp dụng mã khuyến mãi cho vé xem phim. Khi được gọi, nó kiểm tra xem mã khuyến mãi có hợp lệ hay không (còn hạn sử dụng, đúng loại vé). Nếu hợp lệ,

thủ tục tính toán giá vé sau khi giảm và cập nhật vào bảng Tickets. Nếu mã không hợp lệ hoặc không áp dụng được cho loại vé đó, hệ thống sẽ báo lỗi.

```
CREATE PROCEDURE sp_ApplyPromotion
    @TicketID INT,
    @PromotionCode NVARCHAR(20)
AS
BEGIN
    DECLARE @PromotionID INT, @Discount DECIMAL(5, 2), @TicketType NVARCHAR(50), @Price DECIMAL(10, 2);
    SELECT @PromotionID = PromotionID, @Discount = DiscountPercentage, @TicketType = ApplicableTicketType
    FROM Promotions
    WHERE PromotionCode = @PromotionCode
    AND GETDATE() BETWEEN StartDate AND EndDate;

    IF @PromotionID IS NULL
    BEGIN
        RAISERROR ('Mã khuyến mãi không hợp lệ hoặc đã hết hạn!', 16, 1);
        RETURN;
    END
    SELECT @Price = Price, @TicketType = TicketType
    FROM Tickets
    WHERE TicketID = @TicketID;
    IF @TicketType != @TicketType
    BEGIN
        RAISERROR ('Loại vé không áp dụng được khuyến mãi này!', 16, 1);
        RETURN;
    END
    UPDATE Tickets
    SET Price = @Price * (1 - @Discount / 100),
        PromotionID = @PromotionID
    WHERE TicketID = @TicketID;
    PRINT 'Áp dụng khuyến mãi thành công!';
END;

EXEC sp_ApplyPromotion
    @TicketID = 1,
    @PromotionCode = 'KM10';
```

- Kết quả

```
Msg 50000, Level 16, State 1, Procedure sp_ApplyPromotion, Line 14 [Batch Start Line 331]
Mã khuyến mãi không hợp lệ hoặc đã hết hạn!

Completion time: 2025-03-15T13:08:45.1167915+07:00
```

5.5. Procedure 5: Tìm ghế trống trong một phòng chiếu cho suất chiếu cụ thể

- Mục đích**:** Stored Procedure này giúp lấy danh sách ghế trống trong một phòng chiếu cho một suất chiếu cụ thể. Nó lọc ra các ghế thuộc phòng chiếu và có trạng thái "Trống", đồng thời loại bỏ các ghế đã được đặt trong bảng Tickets. Thủ tục này hỗ trợ hệ thống bán vé hiển thị danh sách ghế trống cho khách hàng lựa chọn.

```sql
CREATE PROCEDURE sp_GetAvailableSeats
    @ShowTimeID INT,
    @RoomID INT
AS
BEGIN
    SELECT s.SeatID, s.SeatNumber
    FROM Seats s
    WHERE s.ScreenRoomID = @RoomID
    AND s.Status = 'Trống'
    AND s.SeatID NOT IN (
        SELECT t.SeatID
        FROM Tickets t
        WHERE t.ShowTimeID = @ShowTimeID
        AND t.RoomID = @RoomID
    );
END;

EXEC sp_GetAvailableSeats
    @ShowTimeID = 1,
    @RoomID = 1;
```

- Kết quả

	SeatID	SeatNumber
1	6	A6
2	8	A8
3	9	A9
4	10	A10
5	14	B4
6	15	B5
7	34	D4
8	39	D9
9	41	E1
10	46	E6

5.6. Procedure 6: Thống kê phim có doanh thu cao nhất trong tháng

- Mục đích: Tìm bộ phim có doanh thu cao nhất trong một tháng và năm cụ thể. Tính tổng doanh thu từ các vé bán ra, nhóm theo tên phim, sắp xếp giảm dần và lấy phim có doanh thu cao nhất. Hỗ trợ quản lý rạp theo dõi phim bán chạy để tối ưu lịch chiếu và chiến lược quảng bá.

```sql
CREATE PROCEDURE sp_GetTopRevenueMovieByMonth
    @Month INT,
    @Year INT
AS
BEGIN
    SELECT TOP 1
        m.NameMovie,
        SUM(i.Amount) AS TotalRevenue
    FROM Movie m
    JOIN Tickets t ON m.MovieID = t.MovieID
    JOIN Invoices i ON t.TicketID = i.TicketID
    WHERE MONTH(i.InvoiceDate) = @Month
    AND YEAR(i.InvoiceDate) = @Year
    GROUP BY m.NameMovie
    ORDER BY TotalRevenue DESC;
END;
GO

EXEC sp_GetTopRevenueMovieByMonth
    @Month = 3,
    @Year = 2025;
```

- Kết quả

	NameMovie	TotalRevenue
1	Die Hard	876173.34

5.7. Procedure 7: Cập nhật tỷ lệ sử dụng suất chiếu

- Mục đích: Stored Procedure sp_UpdateCapacityUtilization được sử dụng để tính toán và cập nhật tỷ lệ sử dụng suất chiếu trong rạp chiếu phim. Thủ tục này lấy tổng số ghế trong phòng chiếu, đếm số ghế đã được bán cho suất chiếu cụ thể, sau đó tính phần trăm số ghế đã bán so với tổng số ghế. Kết quả sẽ được cập nhật vào bảng ShowTime, giúp theo dõi mức độ lấp đầy của từng suất chiếu. Điều này hỗ trợ quản lý suất chiếu hiệu quả, tối ưu hóa lịch trình và cải thiện chiến lược bán vé.

```
CREATE PROCEDURE sp_UpdateCapacityUtilization
    @ShowTimeID INT
AS
BEGIN
    DECLARE @RoomID INT, @TotalSeats INT, @SoldSeats INT, @Capacity DECIMAL(5, 2)
    SELECT @RoomID = RoomID
    FROM Tickets
    WHERE ShowTimeID = @ShowTimeID
    GROUP BY RoomID;
    SELECT @TotalSeats = TotalSeat
    FROM ScreenRoom
    WHERE ScreenRoomID = @RoomID;
    SELECT @SoldSeats = COUNT(*)
    FROM Tickets
    WHERE ShowTimeID = @ShowTimeID
    AND RoomID = @RoomID;
    SET @Capacity = (@SoldSeats * 100.0) / @TotalSeats;
    UPDATE ShowTime
    SET CapacityUtilization = @Capacity
    WHERE ShowTimeID = @ShowTimeID;
    PRINT N'Cập nhật tỷ lệ sử dụng suất chiếu thành công!';
END;
Drop procedure sp_UpdateCapacityUtilization
EXEC sp_UpdateCapacityUtilization
    @ShowTimeID = 1;
```

- Kết quả

```
(1 row affected)
Cập nhật tỷ lệ sử dụng suất chiếu thành công!

Completion time: 2025-03-15T13:19:35.2844242+07:00
```

5.8. Procedure 8: Tạo hóa đơn tự động khi bán vé

- Mục đích**:** Stored Procedure sp_CreateInvoice được sử dụng để tạo hóa đơn mới cho khách hàng khi họ mua vé xem phim. Thủ tục này lấy thông tin giá vé từ bảng Tickets dựa trên TicketID, sau đó chèn dữ liệu vào bảng Invoices, bao gồm mã vé, mã khách hàng, số tiền, phương thức thanh toán và ngày lập hóa đơn. Sau khi hóa đơn được tạo thành công, hệ thống hiển thị thông báo xác nhận. Điều này giúp tự động hóa quy trình thanh toán, đảm bảo tính chính xác và quản lý hiệu quả doanh thu từ việc bán vé.

```
CREATE PROCEDURE sp_CreateInvoice
    @TicketID INT,
    @CustomerID INT,
    @PaymentMethod NVARCHAR(20)
AS
BEGIN
    DECLARE @Amount DECIMAL(10, 2);
    SELECT @Amount = Price
    FROM Tickets
    WHERE TicketID = @TicketID;
    INSERT INTO Invoices (TicketID, CustomerID, Amount, PaymentMethod, InvoiceDate)
    VALUES (@TicketID, @CustomerID, @Amount, @PaymentMethod, GETDATE());
    PRINT 'Tạo hóa đơn thành công!';
END;

EXEC sp_CreateInvoice
    @TicketID = 3,
    @CustomerID = 3,
    @PaymentMethod = N'Thẻ';
```

- Kết quả

```
(1 row affected)
Tạo hóa đơn thành công!

Completion time: 2025-03-15T13:24:00.2646005+07:00
```

5.9. Procedure 9: Tìm khách hàng mua nhiều vé nhất trong khoảng thời gian

- Mục đích**:** Stored Procedure này dùng để tìm khách hàng mua nhiều vé nhất trong khoảng thời gian được chỉ định. Hệ thống sẽ đếm số vé mà mỗi khách hàng đã mua, sau đó chọn ra khách hàng có số lượng vé cao nhất và hiển thị thông tin của họ. Điều này giúp quản lý dễ dàng xác định khách hàng trung thành để áp dụng ưu đãi phù hợp.

```
CREATE PROCEDURE sp_GetTopCustomerByTicketCount
    @StartDate DATE,
    @EndDate DATE
AS
BEGIN
    SELECT TOP 1
        c.CustomerID,
        c.FullName,
        COUNT(t.TicketID) AS TotalTickets
    FROM Customers c
    JOIN Tickets t ON c.CustomerID = t.CustomerID
    WHERE t.TicketSaleDate BETWEEN @StartDate AND @EndDate
    GROUP BY c.CustomerID, c.FullName
    ORDER BY TotalTickets DESC;
END;

EXEC sp_GetTopCustomerByTicketCount
    @StartDate = '2025-03-01',
    @EndDate = '2025-03-31';
```

- Kết quả

	CustomerID	FullName	TotalTickets
1	65	Lê Văn Duy	2

5.10. Procedure 10: Tự động gia hạn khuyến mãi nếu chưa hết hạn và đạt điều kiện

- Mục đích: Stored Procedure này dùng để gia hạn chương trình khuyến mãi nếu số lượng vé bán ra đạt mức tối thiểu yêu cầu. Nếu đủ điều kiện, hệ thống sẽ tự động kéo dài thời gian khuyến mãi theo số ngày được chỉ định. Ngược lại, nếu vé bán ra không đủ, hệ thống sẽ báo lỗi và không gia hạn. Điều này giúp quản lý chương trình khuyến mãi linh hoạt và hiệu quả hơn.

```
CREATE PROCEDURE sp_ExtendPromotion
    @PromotionID INT,
    @ExtensionDays INT,
    @MinTicketSold INT
AS
BEGIN
    DECLARE @TicketCount INT;
    SELECT @TicketCount = COUNT(*)
    FROM Tickets
    WHERE PromotionID = @PromotionID;
    IF @TicketCount >= @MinTicketSold
    BEGIN
        UPDATE Promotions
        SET EndDate = DATEADD(DAY, @ExtensionDays, EndDate)
        WHERE PromotionID = @PromotionID
        AND EndDate >= GETDATE();
        PRINT N'Gia hạn khuyến mãi thành công!';
    END
    ELSE
    BEGIN
        RAISERROR (N'Không đủ số vé để gia hạn khuyến mãi!', 16, 1);
    END
END;
drop procedure sp_ExtendPromotion
EXEC sp_ExtendPromotion
    @PromotionID = 1,
    @ExtensionDays = 15,
    @MinTicketSold = 2;
```

- Kết quả

```
(1 row affected)
Gia hạn khuyến mãi thành công!

Completion time: 2025-03-15T13:26:40.5161713+07:00
```

CHƯƠNG 6. XÂY DỰNG CÁC TRIGGER

6.1. Trigger 1: Tự động cập nhật trạng thái ghế thành “Đã đặt” khi thêm vé mới

- Mục đích: Tự động cập nhật trạng thái ghế thành "Đã đặt" khi thêm vé mới vào hệ thống. Điều này giúp đảm bảo rằng ghế đã được đặt không thể bị người khác đặt trùng, tránh sai sót trong việc quản lý chỗ ngồi.

```
CREATE TRIGGER tr_AfterInsertTicket
ON Tickets
AFTER INSERT
AS
BEGIN
    UPDATE Seats
    SET Status = 'Đã đặt'
    WHERE SeatID IN (SELECT SeatID FROM inserted);
    PRINT N'Trigger tr_AfterInsertTicket đã cập nhật trạng thái ghế thành Đã đặt!';
END;

drop trigger tr_AfterInsertTicket
INSERT INTO Tickets (Price, EmployeeID, CustomerID, SeatID, MovieID, RoomID, ShowTimeID, TicketType, TicketSaleDate, PromotionID)
VALUES (120000.00, 1, 1, 2, 1, 1, 1, N'Thường', '2025-03-14', NULL);
```

- Kết quả

```
(1 row affected)
Trigger tr_AfterInsertTicket đã cập nhật trạng thái ghế thành Đã đặt!

(1 row affected)

Completion time: 2025-03-15T13:50:12.1351860+07:00
```

6.2. Trigger 2: Ngăn chặn xóa khách hàng nếu đã mua vé

- Mục đích: Ngăn chặn xóa khách hàng nếu họ đã mua vé trước đó. Điều này nhằm bảo vệ dữ liệu giao dịch, tránh trường hợp mất thông tin quan trọng khi cần kiểm tra lịch sử mua vé của khách hàng.

```
CREATE TRIGGER tr_PreventDeleteCustomer
ON Customers
INSTEAD OF DELETE
AS
BEGIN
    IF EXISTS (SELECT 1 FROM deleted d JOIN Tickets t ON d.CustomerID = t.CustomerID)
    BEGIN
        RAISERROR (N'Không thể xóa khách hàng đã mua vé!', 16, 1);
        ROLLBACK;
    END
    ELSE
    BEGIN
        DELETE FROM Customers WHERE CustomerID IN (SELECT CustomerID FROM deleted);
        PRINT N'Xóa khách hàng thành công!';
    END
END;

DELETE FROM Customers WHERE CustomerID = 1;
```

- Kết quả

```
Msg 50000, Level 16, State 1, Procedure tr_PreventDeleteCustomer, Line 8 [Batch Start Line 515]
Không thể xóa khách hàng đã mua vé!
Msg 3609, Level 16, State 1, Line 519
The transaction ended in the trigger. The batch has been aborted.

Completion time: 2025-03-15T13:51:56.2969324+07:00
```

6.3. Trigger 3: Ghi log khi cập nhật giá vé

- Mục đích: Ghi log khi cập nhật giá vé nhằm theo dõi các thay đổi trong hệ thống. Việc này giúp quản lý có thể kiểm tra lịch sử thay đổi giá, tránh các trường hợp sai lệch hoặc gian lận trong việc cập nhật giá vé.

```
CREATE TRIGGER tr_LogTicketUpdate
ON Tickets
AFTER UPDATE
AS
BEGIN
    INSERT INTO TicketUpdateLog (TicketID, OldPrice, NewPrice, UpdateDate)
    SELECT i.TicketID, d.Price, i.Price, GETDATE()
    FROM inserted i
    JOIN deleted d ON i.TicketID = d.TicketID
    WHERE i.Price != d.Price;
    PRINT N'Trigger tr_LogTicketUpdate đã ghi log cập nhật giá vé!';
END;
GO
drop trigger tr_LogTicketUpdate
CREATE TABLE TicketUpdateLog (
    LogID INT IDENTITY(1,1) PRIMARY KEY,
    TicketID INT,
    OldPrice DECIMAL(10, 2),
    NewPrice DECIMAL(10, 2),
    UpdateDate DATETIME
);
GO

UPDATE Tickets SET Price = 130000.00 WHERE TicketID = 2;
```

- Kết quả

```
(0 rows affected)
Trigger tr_LogTicketUpdate đã ghi log cập nhật giá vé!

(1 row affected)

Completion time: 2025-03-15T13:54:25.6553225+07:00
```

6.4. Trigger 4: Ngăn chặn đặt vé nếu suất chiếu đã đầy

- Mục đích: Ngăn chặn việc đặt vé nếu suất chiếu đã đạt giới hạn số ghế tối đa. Trigger này giúp đảm bảo rằng không có tình trạng bán quá số ghế có sẵn trong rạp, tránh ảnh hưởng đến trải nghiệm khách hàng.

```
CREATE TRIGGER tr_PreventOverCapacity
ON Tickets
INSTEAD OF INSERT
AS
BEGIN
    IF EXISTS (
        SELECT 1
        FROM inserted i
        JOIN ShowTime s ON i.ShowTimeID = s.ShowTimeID
        JOIN ScreenRoom sr ON i.RoomID = sr.ScreenRoomID
        JOIN (SELECT ShowTimeID, COUNT(*) as SeatCount FROM Tickets GROUP BY ShowTimeID) t
            ON i.ShowTimeID = t.ShowTimeID
        WHERE t.SeatCount >= sr.TotalSeat
    )
    BEGIN
        RAISERROR ('Suất chiếu đã đầy, không thể đặt thêm vé!', 16, 1);
        ROLLBACK;
    END
    ELSE
    BEGIN
        INSERT INTO Tickets (Price, EmployeeID, CustomerID, SeatID, MovieID, RoomID, ShowTimeID, TicketType, TicketSaleDate, PromotionID)
        SELECT Price, EmployeeID, CustomerID, SeatID, MovieID, RoomID, ShowTimeID, TicketType, TicketSaleDate, PromotionID
        FROM inserted;
        PRINT 'Trigger tr_PreventOverCapacity đã cho phép đặt vé!';
    END
END;

INSERT INTO Tickets (Price, EmployeeID, CustomerID, SeatID, MovieID, RoomID, ShowTimeID, TicketType, TicketSaleDate, PromotionID)
VALUES (120000.00, 1, 2, 3, 1, 1, 1, N'Thường', '2025-03-14', NULL);
```

- Kết quả

```
(1 row affected)
Trigger tr_AfterInsertTicket đã cập nhật trạng thái ghế thành Đã đặt!

(1 row affected)
Trigger tr_PreventOverCapacity đã cho phép đ?t vé!

(1 row affected)

Completion time: 2025-03-15T13:55:43.8171638+07:00
```

6.5. Trigger 5: Tự động cập nhật hóa đơn khi giá vé thay đổi

- Mục đích: Khi giá vé thay đổi, trigger này sẽ tự động cập nhật lại số tiền trong hóa đơn tương ứng. Điều này giúp hệ thống kế toán luôn chính xác, tránh tình trạng hóa đơn không khớp với giá vé thực tế.

```
CREATE TRIGGER tr_UpdateInvoiceOnTicketPriceChange
ON Tickets
AFTER UPDATE
AS
BEGIN
    IF UPDATE(Price)
    BEGIN
        UPDATE i
        SET i.Amount = t.Price
        FROM Invoices i
        JOIN inserted t ON i.TicketID = t.TicketID;
        PRINT 'Trigger tr_UpdateInvoiceOnTicketPriceChange đã cập nhật hóa đơn!';
    END
END;

UPDATE Tickets SET Price = 140000.00 WHERE TicketID = 3;
```

- Kết quả

```
(1 row affected)
Trigger tr_LogTicketUpdate đã ghi log cập nhật giá vé!

(3 rows affected)
Trigger tr_UpdateInvoiceOnTicketPriceChange đã c?p nh?t hóa don!

(1 row affected)

Completion time: 2025-03-15T13:56:52.7954252+07:00
```

6.6. Trigger 6: Tự động gửi thông báo khi thêm khuyến mại mới

- Mục đích: Khi có chương trình khuyến mãi mới, trigger này sẽ tự động gửi thông báo về chi tiết khuyến mãi. Điều này giúp nhân viên và khách hàng nắm bắt thông tin kịp thời, góp phần tăng hiệu quả marketing.

```
CREATE TRIGGER tr_NotifyNewPromotion
ON Promotions
AFTER INSERT
AS
BEGIN
    DECLARE @PromotionCode NVARCHAR(20), @Discount DECIMAL(5, 2);
    SELECT @PromotionCode = PromotionCode, @Discount = DiscountPercentage FROM inserted;
    PRINT N'Khuyến mãi mới: Mã ' + @PromotionCode + ' với giảm giá ' + CAST(@Discount AS NVARCHAR(5)) + '% đã được thêm!';
END;
drop trigger tr_NotifyNewPromotion
INSERT INTO Promotions (PromotionCode, DiscountPercentage, StartDate, EndDate, ApplicableTicketType)
VALUES (N'KMNEW2025', 15.00, '2025-03-15', '2025-04-15', N'Thường');
```

- Kết quả

```
Khuyến mãi mới: Mã KMNEW2025 với giảm giá 15.00% đã được thêm!

(1 row affected)

Completion time: 2025-03-15T14:00:30.7857184+07:00
```

6.7. Trigger 7: Tự động cập nhật lương nhân viên khi bán vé thành công

- Mục đích: Mỗi khi nhân viên bán được một vé, trigger này sẽ tự động tăng lương cho nhân viên đó. Cơ chế này giúp khuyến khích nhân viên làm việc hiệu quả hơn và phản ánh doanh số bán hàng vào thu nhập của họ.

```
CREATE TRIGGER tr_UpdateEmployeeSalary
ON Tickets
AFTER INSERT
AS
BEGIN
    UPDATE Employees
    SET Salary = Salary + 5000 -- Tăng 5000 cho mỗi vé bán
    WHERE EmployeeID IN (SELECT EmployeeID FROM inserted);
    PRINT 'Trigger tr_UpdateEmployeeSalary đã cập nhật lương nhân viên!';
END;

INSERT INTO Tickets (Price, EmployeeID, CustomerID, SeatID, MovieID, RoomID, ShowTimeID, TicketType, TicketSaleDate, PromotionID)
VALUES (120000.00, 1, 3, 4, 1, 1, 1, N'Thường', '2025-03-14', NULL);
```

- Kết quả

```
(1 row affected)
Trigger tr_AfterInsertTicket đã cập nhật trạng thái ghế thành Đã đặt!

(1 row affected)
Trigger tr_UpdateEmployeeSalary đã c?p nh?t luong nhân viên!

(1 row affected)
Trigger tr_PreventOverCapacity đã cho phép d?t vé!

(1 row affected)

Completion time: 2025-03-15T14:01:19.2720536+07:00
```

6.8. Trigger 8: Ngăn chặn cập nhật sai ngày bán vé

- Mục đích: Ngăn chặn việc cập nhật ngày bán vé trong tương lai, đảm bảo dữ liệu luôn chính xác. Điều này giúp hệ thống tránh được những sai sót hoặc hành vi gian lận trong việc đặt vé trước ngày hiện tại.

```
CREATE TRIGGER tr_PreventInvalidTicketSaleDate
ON Tickets
INSTEAD OF UPDATE
AS
BEGIN
    IF UPDATE(TicketSaleDate)
    BEGIN
        IF EXISTS (
            SELECT 1 FROM inserted i
            WHERE i.TicketSaleDate > GETDATE()
        )
        BEGIN
            RAISERROR ('Ngày bán vé không thể là tương lai!', 16, 1);
            ROLLBACK;
        END
        ELSE
        BEGIN
            UPDATE Tickets
            SET TicketSaleDate = i.TicketSaleDate
            FROM Tickets t
            JOIN inserted i ON t.TicketID = i.TicketID;
            PRINT 'Trigger tr_PreventInvalidTicketSaleDate đã cập nhật ngày bán vé!';
        END
    END
END;

UPDATE Tickets SET TicketSaleDate = '2025-04-01' WHERE TicketID = 3;
```

- Kết quả

```
Msg 50000, Level 16, State 1, Procedure tr_PreventInvalidTicketSaleDate, Line 13 [Batch Start Line 698]
Ngày bán vé không thể là tương lai!
Msg 3609, Level 16, State 1, Line 699
The transaction ended in the trigger. The batch has been aborted.

Completion time: 2025-03-15T14:02:24.5139036+07:00
```

6.9. Trigger 9: Tự động tạo hóa đơn khi thêm vé

- Mục đích: Tự động tạo hóa đơn ngay khi khách hàng mua vé. Điều này giúp hệ thống vận hành nhanh chóng, giảm công việc nhập liệu thủ công của nhân viên và đảm bảo không có giao dịch nào bị bỏ sót.

```
CREATE TRIGGER tr_AutoCreateInvoice
ON Tickets
AFTER INSERT
AS
BEGIN
    INSERT INTO Invoices (TicketID, CustomerID, Amount, PaymentMethod, InvoiceDate)
    SELECT i.TicketID, i.CustomerID, i.Price, N'Tiền mặt', GETDATE()
    FROM inserted i;
    PRINT 'Trigger tr_AutoCreateInvoice đã tạo hóa đơn tự động!';
END;

INSERT INTO Tickets (Price, EmployeeID, CustomerID, SeatID, MovieID, RoomID, ShowTimeID, TicketType, TicketSaleDate, PromotionID)
VALUES (130000.00, 2, 3, 5, 2, 2, 2, N'VIP', '2025-03-14', NULL);
```

- Kết quả

```
(1 row affected)
Trigger tr_AfterInsertTicket đã cập nhật trạng thái ghế thành Đã đặt!

(1 row affected)
Trigger tr_UpdateEmployeeSalary đã c?p nh?t luong nhân viên!

(1 row affected)
Trigger tr_AutoCreateInvoice đã t?o hóa don t? d?ng!

(1 row affected)
Trigger tr_PreventOverCapacity đã cho phép d?t vé!

(1 row affected)

Completion time: 2025-03-15T14:03:27.1275288+07:00
```

6.10. Trigger 10: Cảnh báo tỉ lệ sử dụng suất chiếu vượt 80%.

- Mục đích: Khi tỷ lệ sử dụng suất chiếu vượt quá 80%, trigger này sẽ cảnh báo cho quản lý. Điều này giúp đội ngũ vận hành có kế hoạch mở thêm suất chiếu hoặc tăng cường nhân sự phục vụ.

```
CREATE TRIGGER tr_WarnHighCapacityUtilization
ON ShowTime
AFTER UPDATE
AS
BEGIN
    IF EXISTS (
        SELECT 1 FROM inserted i
        WHERE i.CapacityUtilization > 80.0
    )
    BEGIN
        PRINT N'CẢNH BÁO: Tỷ lệ sử dụng suất chiếu vượt 80%!';
    END
END;
GO
drop trigger tr_WarnHighCapacityUtilization
UPDATE ShowTime SET CapacityUtilization = 85.00 WHERE ShowTimeID = 1;
```

- Kết quả

```
CANH BAO: Tỷ lệ sử dụng suất chiếu vượt 80%!

(1 row affected)

Completion time: 2025-03-15T14:04:44.1067292+07:00
```

6.11. Trigger 11. Ngăn chặn đặt vé trùng ghế trong cùng suất chiếu.

- Trigger tr_CheckCapacityAndDuplicateSeat được tạo ra nhằm ngăn chặn việc đặt vé trùng ghế trong cùng một suất chiếu và đảm bảo suất chiếu không vượt quá số lượng ghế tối đa của phòng chiếu. Khi có yêu cầu thêm vé, trigger sẽ kiểm tra xem suất chiếu đã đầy hay chưa và ghế đã được đặt hay chưa. Nếu vi phạm, hệ thống sẽ báo lỗi và hủy thao tác, giúp duy trì tính toàn vẹn dữ liệu và tránh xung đột đặt vé.

```
CREATE TRIGGER tr_CheckCapacityAndDuplicateSeat
ON Tickets
INSTEAD OF INSERT
AS
BEGIN
    IF EXISTS (
        SELECT 1
        FROM inserted i
        JOIN ShowTime s ON i.ShowTimeID = s.ShowTimeID
        JOIN ScreenRoom sr ON i.RoomID = sr.ScreenRoomID
        JOIN (SELECT ShowTimeID, COUNT(*) as SeatCount FROM Tickets GROUP BY ShowTimeID) t
            ON i.ShowTimeID = t.ShowTimeID
        WHERE t.SeatCount >= sr.TotalSeat
    )
    BEGIN
        RAISERROR ('Suất chiếu đã đầy, không thể đặt thêm vé!', 16, 1);
        ROLLBACK;
        RETURN;
    END
    IF EXISTS (
        SELECT 1
        FROM inserted i
        JOIN Tickets t ON i.SeatID = t.SeatID AND i.ShowTimeID = t.ShowTimeID AND i.RoomID = t.RoomID
    )
    BEGIN
        RAISERROR ('Ghế đã được đặt trong suất chiếu này!', 16, 1);
        ROLLBACK;
        RETURN;
    END
    INSERT INTO Tickets (Price, EmployeeID, CustomerID, SeatID, MovieID, RoomID, ShowTimeID, TicketType, TicketSaleDate, PromotionID)
    SELECT Price, EmployeeID, CustomerID, SeatID, MovieID, RoomID, ShowTimeID, TicketType, TicketSaleDate, PromotionID
    FROM inserted;
    PRINT 'Trigger tr_CheckCapacityAndDuplicateSeat đã cho phép đặt vé!';
END;

DROP TRIGGER IF EXISTS tr_CheckCapacityAndDuplicateSeat;
INSERT INTO Tickets (Price, EmployeeID, CustomerID, SeatID, MovieID, RoomID, ShowTimeID, TicketType, TicketSaleDate, PromotionID)
VALUES (120000.00, 1, 2, 2, 1, 1, 1, N'Thường', '2025-03-14', NULL);
```

- Kết quả

```
Msg 50000, Level 16, State 1, Procedure tr_CheckCapacityAndDuplicateSeat, Line 26 [Batch Start Line 614]
Ghế đã được đặt trong suất chiếu này!
Msg 3609, Level 16, State 1, Line 615
The transaction ended in the trigger. The batch has been aborted.
```

CHƯƠNG 7. PHÂN QUYỀN VÀ BẢO VỆ CƠ SỞ DỮ LIỆU

7.1. Tạo Login cho các tài khoản

```
IF NOT EXISTS (SELECT * FROM sys.server_principals WHERE name = 'ManagerLogin')
    CREATE LOGIN ManagerLogin WITH PASSWORD = 'Manager@123455', DEFAULT_DATABASE = QLHTBVXP, CHECK_POLICY = ON;
IF NOT EXISTS (SELECT * FROM sys.server_principals WHERE name = 'EmployeeLogin')
    CREATE LOGIN EmployeeLogin WITH PASSWORD = 'Employee@123455', DEFAULT_DATABASE = QLHTBVXP, CHECK_POLICY = ON;
IF NOT EXISTS (SELECT * FROM sys.server_principals WHERE name = 'CustomerLogin')
    CREATE LOGIN CustomerLogin WITH PASSWORD = 'Customer@123455', DEFAULT_DATABASE = QLHTBVXP, CHECK_POLICY = ON;
```

7.2. Tạo User và ánh xạ với Login

```
IF NOT EXISTS (SELECT * FROM sys.database_principals WHERE name = 'ManagerUser')
    CREATE USER ManagerUser FOR LOGIN ManagerLogin;
IF NOT EXISTS (SELECT * FROM sys.database_principals WHERE name = 'EmployeeUser')
    CREATE USER EmployeeUser FOR LOGIN EmployeeLogin;
IF NOT EXISTS (SELECT * FROM sys.database_principals WHERE name = 'CustomerUser')
    CREATE USER CustomerUser FOR LOGIN CustomerLogin;
```

7.3. Tạo Role

```
IF NOT EXISTS (SELECT * FROM sys.database_principals WHERE name = 'ManagerRole')
    CREATE ROLE ManagerRole;
IF NOT EXISTS (SELECT * FROM sys.database_principals WHERE name = 'EmployeeRole')
    CREATE ROLE EmployeeRole;
IF NOT EXISTS (SELECT * FROM sys.database_principals WHERE name = 'CustomerRole')
    CREATE ROLE CustomerRole;
```

7.4. Gán User vào Role

```
ALTER ROLE ManagerRole ADD MEMBER ManagerUser;
ALTER ROLE EmployeeRole ADD MEMBER EmployeeUser;
ALTER ROLE CustomerRole ADD MEMBER CustomerUser;
```

7.5. Tạo bảng Log để bảo vệ (ghi lại hành động)

```
CREATE TABLE AuditLog (
    LogID INT IDENTITY(1,1) PRIMARY KEY,
    UserName NVARCHAR(128),
    ActionType NVARCHAR(50),
    TableName NVARCHAR(128),
    RecordID INT,
    ActionDate DATETIME DEFAULT GETDATE()
);
```

7.6. Tạo trigger để ghi log khi chèn vé

```
CREATE TRIGGER tr_LogTicketInsert
ON Tickets
AFTER INSERT
AS
BEGIN
    INSERT INTO AuditLog (UserName, ActionType, TableName, RecordID)
    SELECT SUSER_SNAME(), 'INSERT', 'Tickets', i.TicketID
    FROM inserted i;
END;
```

7.7. Phân quyền từng vai trò

7.7.1. Quyền cho tài khoản quản lý (ManangerRole)

```
ALTER ROLE db_owner ADD MEMBER ManagerUser;
GRANT SELECT, INSERT, UPDATE, DELETE, EXECUTE ON SCHEMA::dbo TO ManagerRole;
GRANT CONTROL ON DATABASE::QLHTBVXP TO ManagerRole;
GRANT SELECT, INSERT, UPDATE, DELETE ON AuditLog TO ManagerRole;
```

7.7.2. Quyền cho Tài khoản Nhân viên (EmployeeRole)

```
GRANT SELECT, INSERT, UPDATE ON Tickets TO EmployeeRole;
GRANT SELECT, UPDATE ON Seats TO EmployeeRole;
GRANT SELECT, INSERT ON Invoices TO EmployeeRole;
GRANT EXECUTE ON sp_UpdateSeatStatus TO EmployeeRole;
GRANT EXECUTE ON sp_CreateInvoice TO EmployeeRole;
DENY SELECT ON Employees TO EmployeeRole;
DENY SELECT ON Promotions TO EmployeeRole;
```

7.7.3. Quyền cho Tài khoản Khách hàng (CustomerRole)

```
GRANT SELECT ON Tickets TO CustomerRole;
GRANT SELECT ON ShowTime TO CustomerRole;
GRANT EXECUTE ON sp_GetShowTimesByMovieAndDate TO CustomerRole;
DENY INSERT, UPDATE, DELETE ON Tickets TO CustomerRole;
DENY INSERT, UPDATE, DELETE ON ShowTime TO CustomerRole;
```

7.7.4. Tạo view với tham số CustomerID và cấp quyền thực thi Stored Procedure

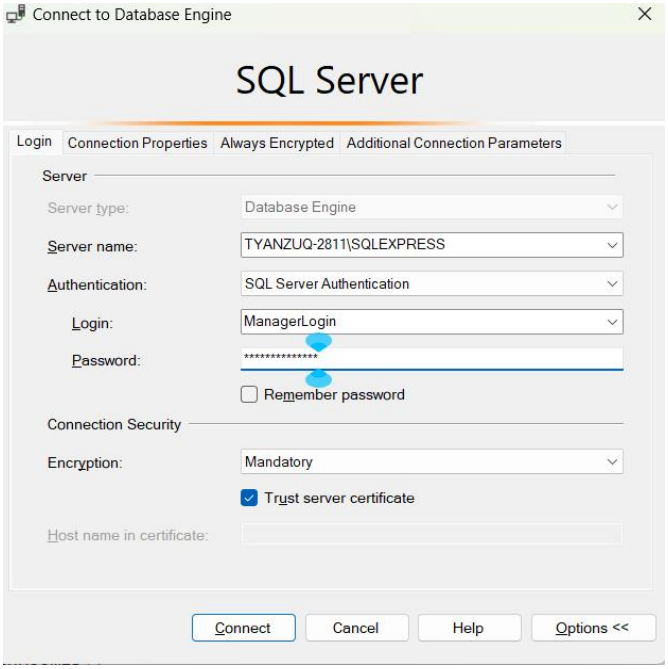
```
CREATE OR ALTER PROCEDURE sp_GetCustomerTickets
    @CustomerID INT
AS
BEGIN
    SELECT t.*
    FROM Tickets t
    WHERE t.CustomerID = @CustomerID;
END;

GRANT EXECUTE ON sp_GetCustomerTickets TO CustomerRole;
```

7.8. Test phân quyền các tài khoản

7.8.1. Test quyền của Tài khoản Quản lý (ManagerLogin/ManagerUser)

- Đăng nhập Tài Khoản Quản lý:

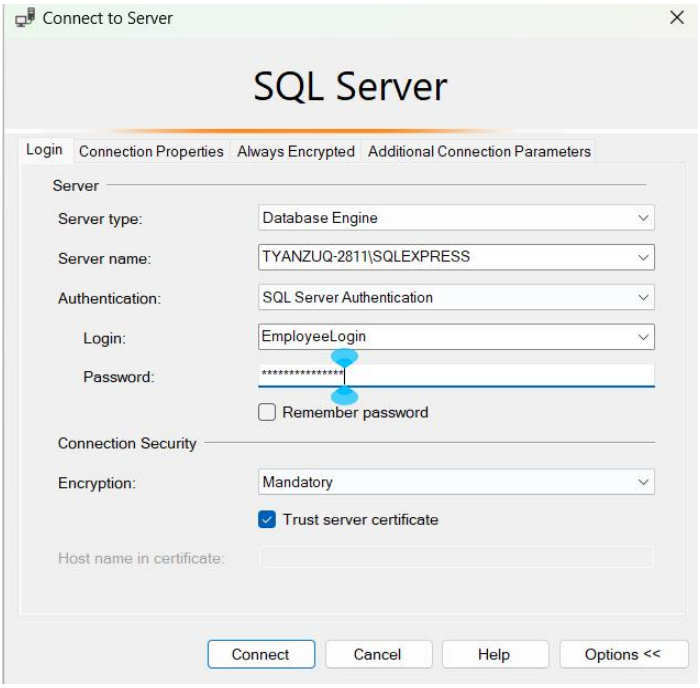


- Test phân quyền:

```
SELECT * FROM Tickets; -- Thành công
UPDATE Tickets SET Price = 150000 WHERE TicketID = 1; -- Thành công
EXEC sp_UpdateSeatStatus @SeatID = 1, @Status = 'Đã bán'; -- Thành công
DROP TABLE Tickets; -- Thành công
SELECT * FROM AuditLog; -- Thành công (xem log)
```

7.8.2. Test quyền của Tài khoản Nhân viên (EmployeeLogin/EmployeeUser)

- Đăng nhập Tài khoản Nhân Viên:

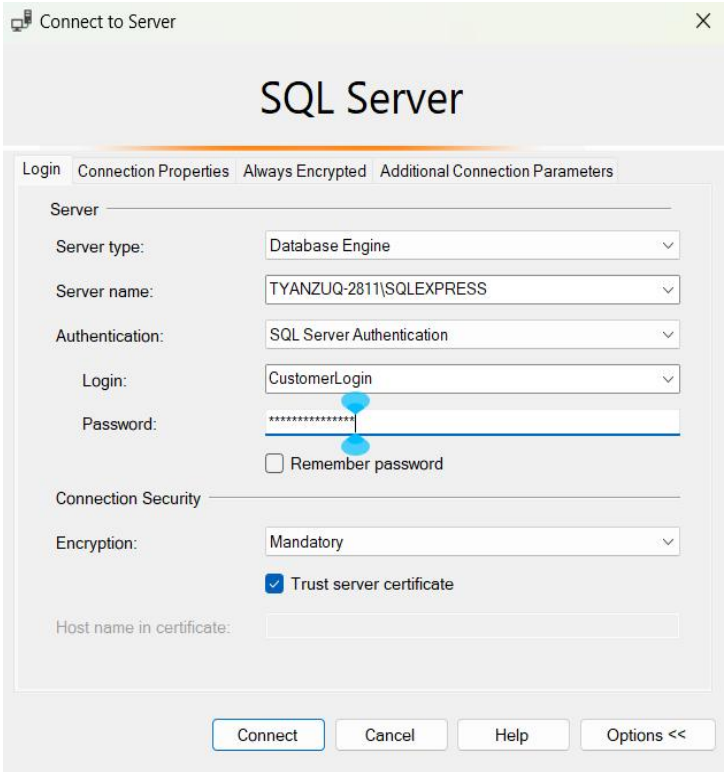


- Test các phân quyền:

```
SELECT * FROM Tickets; -- Thành công
UPDATE Seats SET Status = 'Đã đặt' WHERE SeatID = 1; -- Thành công
EXEC sp_UpdateSeatStatus @SeatID = 1, @Status = 'Đã đặt'; -- Thành công
SELECT * FROM Employees; -- Bị từ chối
INSERT INTO Promotions VALUES ('NEWPROMO', 10, '2025-03-01', '2025-03-31', 'Thường'); -- Bị từ chối
INSERT INTO Tickets VALUES (120000, 1, 1, 1, 1, 1, 'Thường', '2025-03-14', NULL); -- Thành công
SELECT * FROM AuditLog; -- Bị từ chối (chỉ Quản lý xem log)
```

7.8.3. Test quyền của Tài khoản Khách hàng (CustomerLogin/CustomerUser)

- Đăng nhập Tài Khoản Khách hàng:



- Test phân quyền:

```
EXEC sp_GetCustomerTickets @CustomerID = 1; -- Thành công (chỉ thấy vé của CustomerID = 1)
EXEC sp_GetShowTimesByMovieAndDate @MovieID = 1, @ShowDate = '2025-03-14'; -- Thành công
UPDATE Tickets SET Price = 150000 WHERE TicketID = 1; -- Bị từ chối
DELETE FROM Tickets WHERE TicketID = 1; -- Bị từ chối
SELECT * FROM Employees; -- Bị từ chối
```

KẾT LUẬN

Hệ thống quản lý vé xem phim mang lại nhiều lợi ích trong việc tự động hóa quy trình bán vé, quản lý phòng chiếu và khách hàng. Với việc sử dụng SQL Server để lưu trữ dữ liệu và Stored Procedure để xử lý giao dịch, hệ thống đảm bảo hiệu suất cao, bảo mật tốt và khả năng mở rộng dễ dàng. Người dùng có thể đặt vé nhanh chóng, kiểm tra tình trạng ghế trống và nhận thông tin khuyến mãi mà không cần đến trực tiếp rạp chiếu. Đồng thời, ban quản lý rạp có thể theo dõi doanh thu, suất chiếu và hiệu suất hoạt động của từng phòng chiếu một cách chi tiết.

Mặc dù hệ thống mang lại nhiều lợi ích, nhưng vẫn tồn tại một số nhược điểm. Một trong những vấn đề chính là khả năng xử lý giao dịch đồng thời khi có lượng lớn người dùng đặt vé cùng lúc, có thể gây quá tải hệ thống. Ngoài ra, việc xử lý hoàn vé hoặc đổi vé có thể gặp khó khăn nếu không có cơ chế kiểm soát chặt chẽ. Một hạn chế khác là phụ thuộc vào cơ sở hạ tầng mạng và máy chủ, nếu hệ thống gặp sự cố hoặc mất kết nối Internet, việc đặt vé trực tuyến sẽ bị gián đoạn.

Trong tương lai, hệ thống có thể nâng cấp lên nền tảng điện toán đám mây (Cloud) để tăng khả năng mở rộng và đảm bảo hiệu suất ổn định ngay cả khi có nhiều người truy cập. Ngoài ra, việc tích hợp trí tuệ nhân tạo (AI) có thể giúp đề xuất phim phù hợp với từng khách hàng dựa trên sở thích và lịch sử đặt vé. Một hướng phát triển quan trọng khác là tích hợp thanh toán điện tử đa dạng hơn như ví Momo, ZaloPay hoặc tiền điện tử, giúp người dùng có nhiều lựa chọn thanh toán tiện lợi hơn.

DANH MỤC TÀI LIỆU THAM KHẢO

[1]. Trần Thị Thanh Nhàn (2025), *Giáo trình Hệ quản trị CSDL*, Đại học Đại Nam.

[2]. Nguyễn Hải Đăng (2024), *Phát triển ứng dụng quản lý vé xem phim với SQL Server và Python*, Nhà xuất bản Khoa học & Kỹ thuật.

[3]. Nguyễn Văn Cường (2023), *Giáo trình Lập trình Python kết nối SQL Server*, Nhà xuất bản Thống Kê.