ASSIGNMENT REPORT

DBI202-DATABASE OF MOVIE TICKET MANAGEMENT SYSTEM

TRẦN HUỲNH PHÚC TẤN - DE 170727

NGUYỄN BÌNH TÚ - DE 170690

LÊ THÀNH NHÂN - DE 180929

PHAN THÁI KHÁNH NHI - DE180931

NGUYỄN LÊ QUANG HUY - DE 170716

I/ PROBLEMS:

This database is designed to manage information related to a movie theater, covering various aspects such as movies, tickets, customers, staff, orders, screens, genres, and ticket types. The database schema includes several interconnected tables, each serving a specific purpose to organize and store relevant data efficiently.

II/ SYSTEM DESCRIPTION:

This system is designed to support and optimize key operations within a movie theater. Here is a detailed description of how the system performs its primary functions:

1. Movie Information Management (MOVIE):

The system stores detailed information about movies, including the movie name, genre, release date, and other details. Managing genres helps categorize and search for movies easily.

2. Ticket Sales Management (TICKET):

The system helps manage the ticket sales process, including information about ticket types, orders, the currently playing movie, the theater screen, seat numbers, showtimes, and release dates. This facilitates tracking ticket sales, customer information, and timely management.

3. Movie Genre Information Management (MOVIEGENRE):

The system holds information about different movie genres, aiding in the organization and classification of movies by genre. This provides a foundation for event organization and promotion.

4. Customer Information Management (CUSTOMER):

The system stores personal information about customers, including names, addresses, phone numbers, and birthdates. This assists in creating and managing customer accounts, as well as tracking their ticket purchase history.

5. Staff Information Management (STAFF):

The system tracks information about staff members, including names, addresses, phone numbers, birthdates, and genders. Managing staff information helps in human resource management, scheduling, and tracking employee activities.

6. Ticket Type Management (TICKETTYPE):

The system maintains information about various ticket types and their prices. This helps in managing ticket prices, categorizing customers, and easily tracking revenue from each ticket type.

7. Theater Screen Information Management (SCREEN):

The system tracks information about the different movie screens in the theater, aiding in organizing and scheduling movie showings. This information is crucial for classifying movie events and managing theater resources.

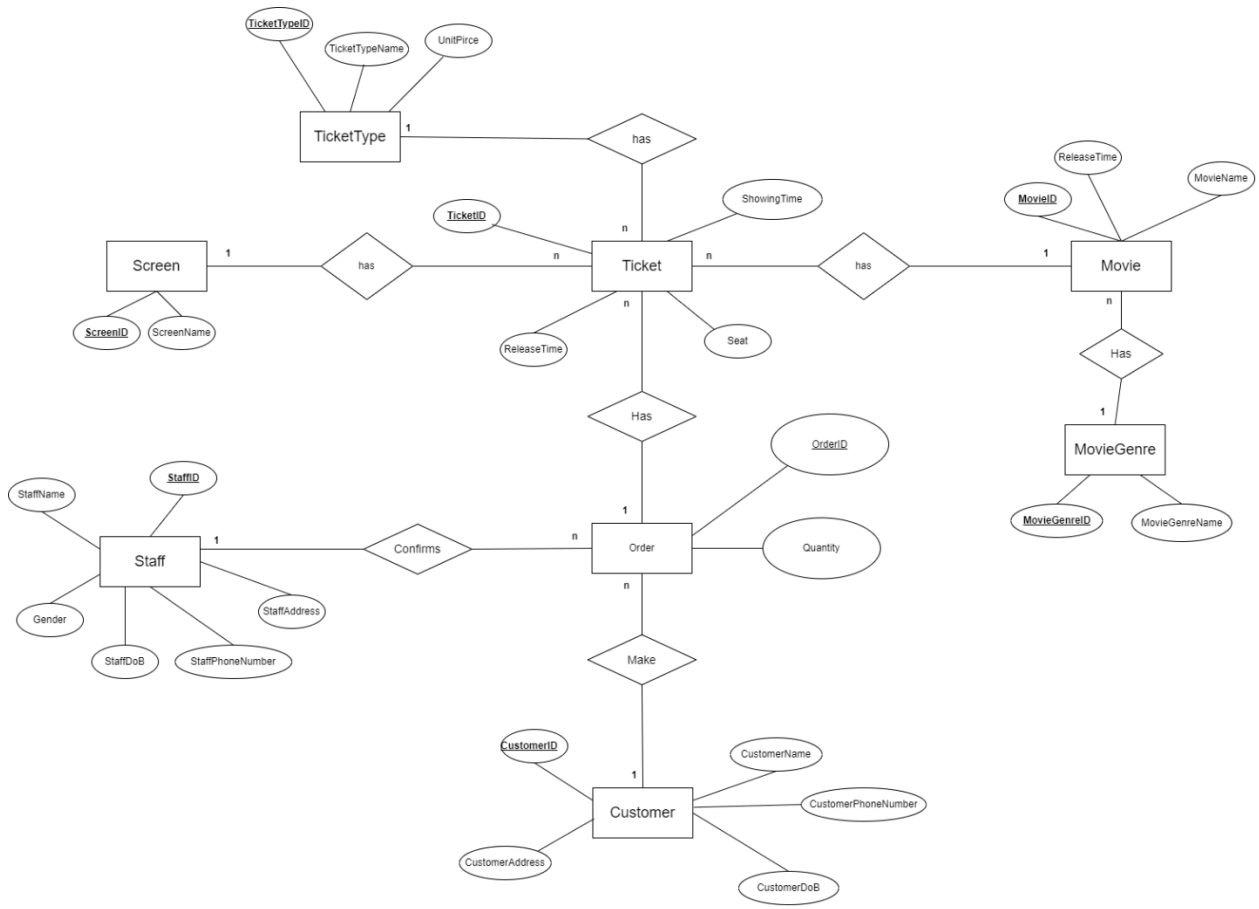
8. Order Detail Management (ORDERDETAIL):

The system stores information about orders, including the quantity of tickets, customer information, and the staff member handling the order. The addition of the Total column helps track the total value of each order.

The system provides a tightly integrated database with relationships between tables to ensure data consistency and efficiency in managing information. For example, the MovieGenreID in the MOVIE table references the MovieGenreID in the MOVIEGENRE table, establishing a link between movies and their genres.

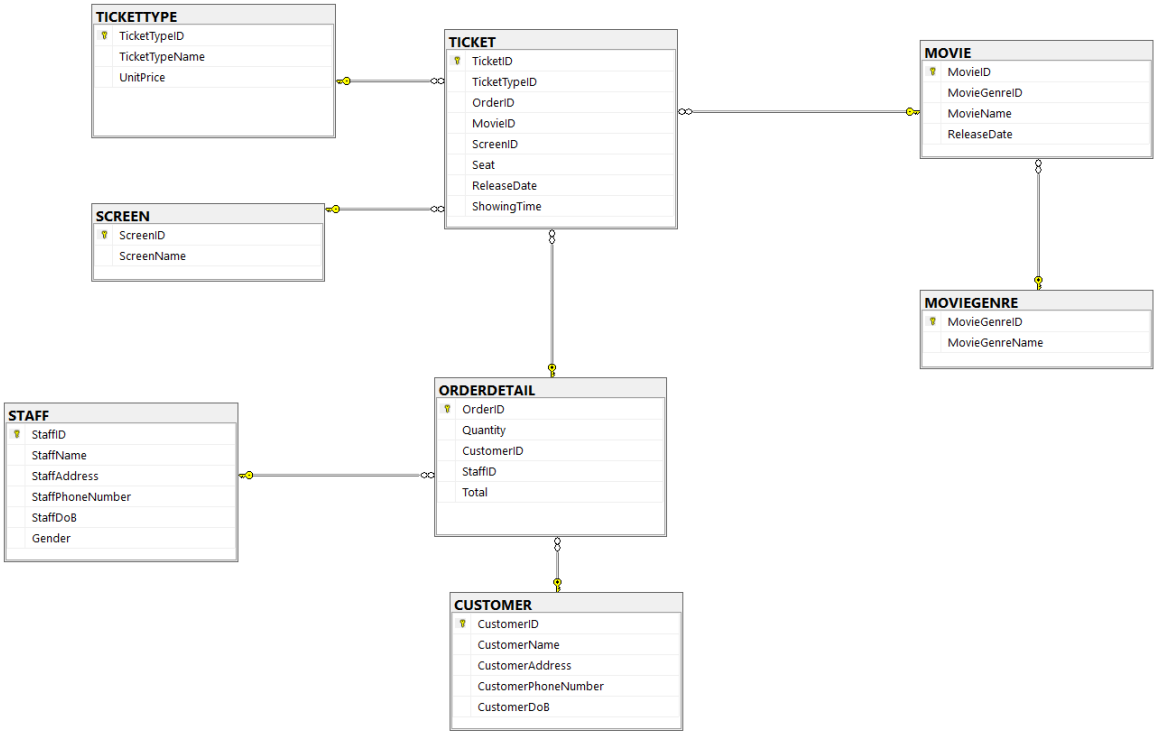
Furthermore, the use of foreign key constraints ensures data integrity, and the sample data illustrates the practical use of the database in a movie theater setting. The system is capable of supporting functions such as searching, tracking revenue, and efficiently managing customer and staff information.

III/ ER DIAGRAM



-

IV/ RELATIONAL MODEL



V/ DATA DESCRIPTION AND DETAILED CONSTRAINTS

Data Element	Description	Composition or Data Type	Length	Values
MovieID	Unique identifier for each movie	CHAR(10)	10 characters	Alphanumeric, 10 characters.
MovieGenreID	Foreign key referencing MovieGenre table, indicating the genre of the movie.	CHAR(10)	10 characters	Alphanumeric, 10 characters.
MovieName	Name of the movie	NVARCHAR(100)	Up to 100 characters.	Alphanumeric, up to 100 characters.
ReleaseDate	Date when the movie was released	DATETIME	N/A	Date and time
TicketID	Unique identifier for each ticket	CHAR(10)	10 characters	Alphanumeric, 10 characters.

TicketTypeID	Foreign key referencing TicketType table, indicating the type of the ticket.	CHAR(10)	10 characters	Alphanumeric, 10 characters.
OrderID	Foreign key referencing OrderDetail table, indicating the order associated with the ticket.	CHAR(10)	10 characters	Alphanumeric, 10 characters.
MovieID	Foreign key referencing OrderDetail table, indicating the order associated with the ticket.	CHAR(10)	10 characters	Alphanumeric, 10 characters.
ScreenID	Foreign key referencing OrderDetail table, indicating the order associated with the ticket.	CHAR(10)	10 characters	Alphanumeric, 10 characters.
Seat	Seat number for the ticket.	INT	N/A	Date and time
ReleaseDate	Date when the ticket is released	DATETIME	N/A	Date and time

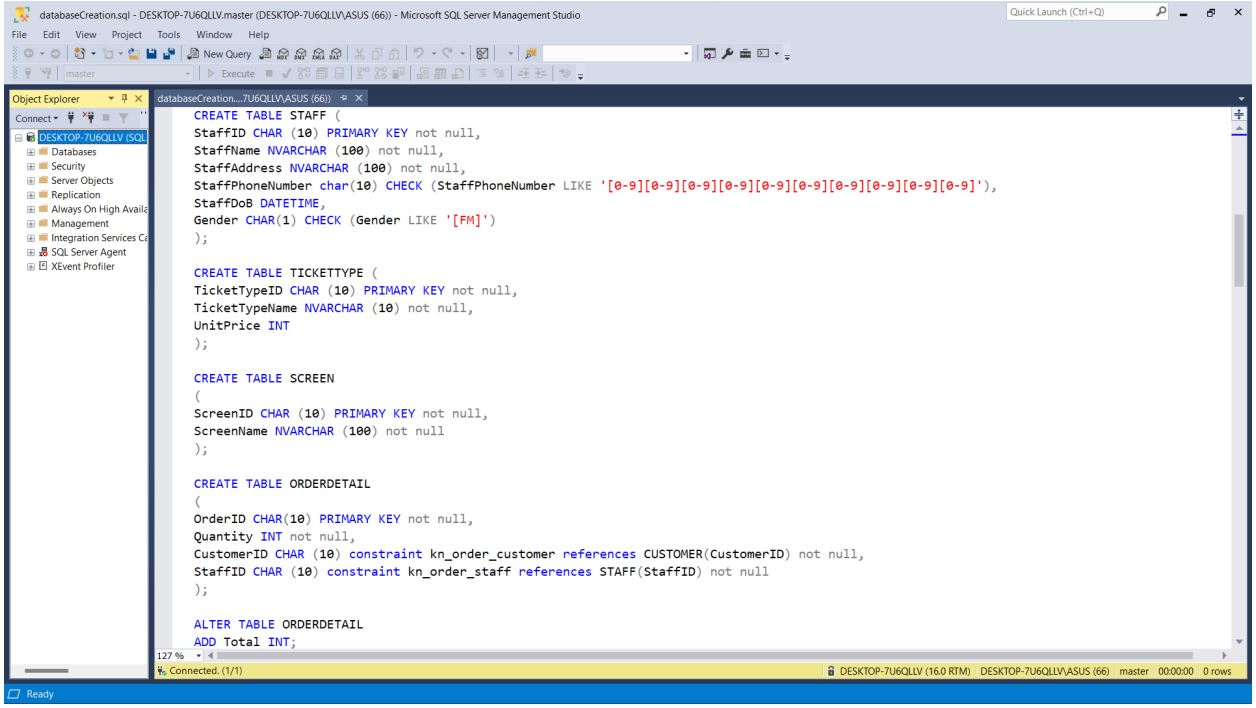
ShowingTime	Date and time when the movie associated with the ticket is scheduled to be shown.	DATETIME	N/A	Date and time
MovieGenreID	Unique identifier for each movie genre	CHAR(10)	10 characters	Alphanumeric, 10 characters
MovieGenreName	Name of the movie genre	NVARCHAR(100)	Up to 100 characters	Alphanumeric, up to 100 characters
CustomerID	Unique identifier for each customer.	CHAR(10)	10 characters	Alphanumeric, 10 characters
CustomerName	Name of the customer.	NVARCHAR(100)	Up to 100 characters	Alphanumeric, up to 100 characters
CustomerAddress	Address of the customer.	NVARCHAR(100)	Up to 100 characters	Alphanumeric, up to 100 characters
CustomerPhoneNumber	Phone number of the customer.	CHAR(10)	10 characters	Alphanumeric, 10 characters
CustomerDoB	Date of Birth of the customer.	DATETIME	N/A	Date
StaffID	Unique identifier for each staff member.	CHAR(10)	10 characters	Alphanumeric, 10 characters

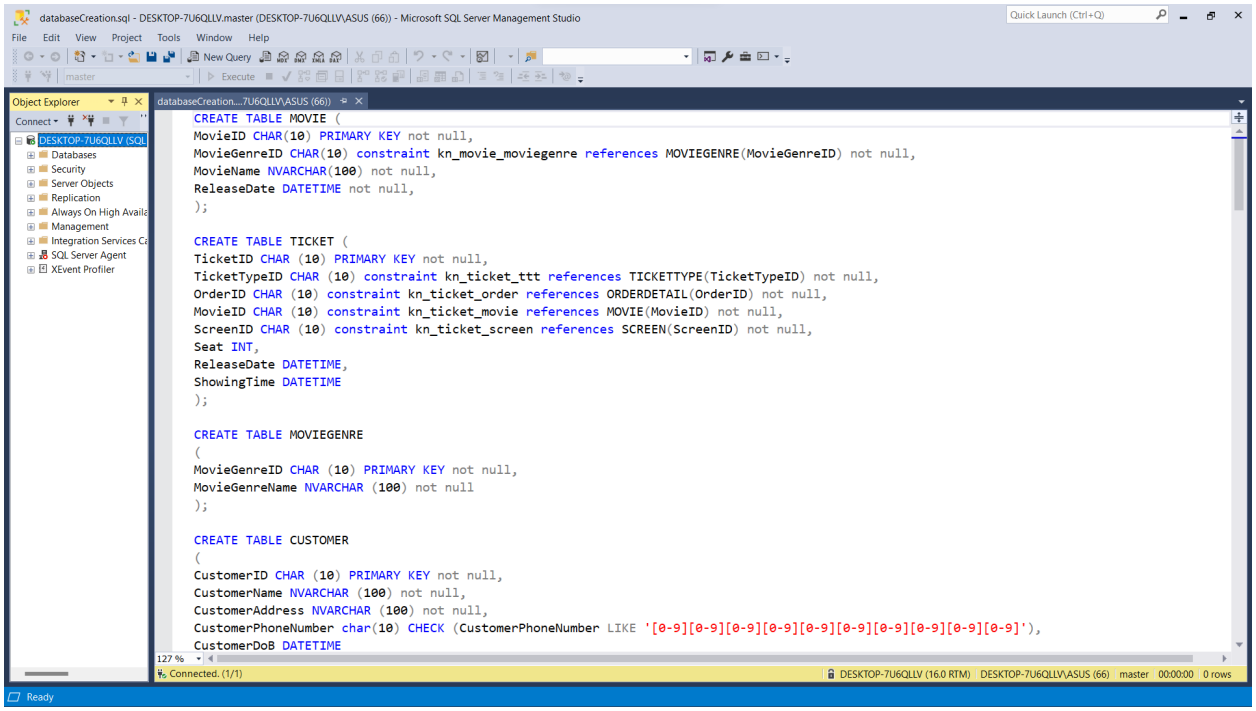
StaffName	Name of the staff member.	NVARCHAR(100)	Up to 100 characters	Alphanumeric, up to 100 characters
StaffAddress	Address of the staff member.	NVARCHAR(100)	Up to 100 characters	Alphanumeric, up to 100 characters
StaffPhoneNumber	Phone number of the staff member.	CHAR(10)	10 characters	Alphanumeric, 10 characters
StaffDoB	Date of Birth of the staff member.	DATETIME	N/A	Date
Gender	Gender of the staff member (M for Male, F for Female).	CHAR(1)	1character	'M' or 'F'
TicketTypeID	Unique identifier for each ticket type.	CHAR(10)	10 characters	Alphanumeric, 10 characters
TicketTypeName	Name of the ticket type.	NVARCHAR(100)	Up to 100 characters	Alphanumeric, up to 100 characters
UnitPrice	The price of a single ticket for this type.	INT	N/A	Integer
ScreenID	Unique identifier for each screen.	CHAR(10)	10 characters	Alphanumeric, 10 characters

ScreenName	Name or identifier of the screen.	NVARCHAR(100)	Up to 100 characters	Alphanumeric, up to 100 characters
OrderID	Unique identifier for each order.	CHAR(10)	10 characters	Alphanumeric, 10 characters
Quantity	The quantity of tickets in the order.	INT	N/A	Integer
CustomerID	Foreign key referencing Customer table, indicating the customer associated with the order.	CHAR(10)	10 characters	Alphanumeric, 10 characters
StaffID	Foreign key referencing Staff table, indicating the staff member associated with the order.	CHAR(10)	10 characters	Alphanumeric, 10 characters
Total	The total cost of the order.	INT	N/A	Integer

VI/ INSTALLATION

1. CREATE TABLE, INSERT DATA





databaseCreation.sql - DESKTOP-7U6QLLV.master (DESKTOP-7U6QLLV\ASUS (66)) - Microsoft SQL Server Management Studio

```sql
    ADD Total INT;

    -- Insert data into the MOVIEGENRE table
    INSERT INTO MOVIEGENRE (MovieGenreID, MovieGenreName)
    VALUES
      ('Genre1', 'Action'),
      ('Genre2', 'Comedy'),
      ('Genre3', 'Drama'),
      ('Genre4', 'Horror'),
      ('Genre5', 'Science Fiction'),
      ('Genre6', 'Romance'),
      ('Genre7', 'Thriller'),
      ('Genre8', 'Adventure'),
      ('Genre9', 'Animation'),
      ('Genre10', 'Fantasy');

    -- Insert data into the CUSTOMER table
    INSERT INTO CUSTOMER (CustomerID, CustomerName, CustomerAddress, CustomerPhoneNumber, CustomerDoB)
    VALUES
      ('C1', 'John Doe', '123 Main St, City', '1234567890', '1990-01-15'),
      ('C2', 'Jane Smith', '456 Elm St, City', '9876543210', '1985-07-22'),
      ('C3', 'Michael Johnson', '789 Oak St, City', '5551112222', '1978-03-10'),
      ('C4', 'Sarah Brown', '234 Pine St, City', '9998887777', '2000-05-30'),
      ('C5', 'David Lee', '567 Maple St, City', '3334445555', '1995-11-18'),
      ('C6', 'Emily Wilson', '890 Birch St, City', '7776665555', '1980-09-05'),
      ('C7', 'Daniel Davis', '123 Cedar St, City', '1112223333', '1992-12-20'),
      ('C8', 'Olivia White', '345 Redwood St, City', '4443332222', '1987-04-27'),
      ('C9', 'William Hall', '678 Sequoia St, City', '6665554444', '2002-08-14'),
      ('C10', 'Sophia Miller', '901 Hemlock St, City', '2227778888', '1975-06-02');

    -- Insert data into the STAFF table
```

Connected. (1/1) | DESKTOP-7U6QLLV (16.0 RTM) | DESKTOP-7U6QLLV\ASUS (66) | master | 00:00:00 | 0 rows

Ready

databaseCreation.sql - DESKTOP-7U6QLLV.master (DESKTOP-7U6QLLV\ASUS (66)) - Microsoft SQL Server Management Studio

```sql
      ('C9', 'William Hall', '678 Sequoia St, City', '6665554444', '2002-08-14'),
      ('C10', 'Sophia Miller', '901 Hemlock St, City', '2227778888', '1975-06-02');

    -- Insert data into the STAFF table
    INSERT INTO STAFF (StaffID, StaffName, StaffAddress, StaffPhoneNumber, StaffDoB, Gender)
    VALUES
      ('Staff1', 'Robert Johnson', '111 Staff Rd, City', '5555555555', '1988-02-12', 'M'),
      ('Staff2', 'Maria Garcia', '222 Staff Rd, City', '6666666666', '1991-09-25', 'F'),
      ('Staff3', 'Michael Smith', '333 Staff Rd, City', '7777777777', '1980-04-07', 'M'),
      ('Staff4', 'Jennifer Davis', '444 Staff Rd, City', '8888888888', '1995-12-03', 'F'),
      ('Staff5', 'David Martinez', '555 Staff Rd, City', '9999999999', '1979-07-15', 'M'),
      ('Staff6', 'Linda Brown', '666 Staff Rd, City', '1111111111', '1983-06-18', 'F'),
      ('Staff7', 'John Wilson', '777 Staff Rd, City', '2222222222', '1993-10-29', 'M'),
      ('Staff8', 'Susan Lee', '888 Staff Rd, City', '3333333333', '1986-03-22', 'F'),
      ('Staff9', 'Daniel Jackson', '999 Staff Rd, City', '4444444444', '1990-11-12', 'M'),
      ('Staff10', 'Emily Taylor', '101 Staff Rd, City', '5555555555', '1987-08-05', 'F');

    -- Insert data into the MOVIE table
    INSERT INTO MOVIE (MovieID, MovieGenreID, MovieName, ReleaseDate)
    VALUES
      ('M1', 'Genre1', 'Action Movie 1', '2023-01-15'),
      ('M2', 'Genre2', 'Comedy Movie 1', '2022-12-10'),
      ('M3', 'Genre3', 'Drama Movie 1', '2023-02-20'),
      ('M4', 'Genre4', 'Horror Movie 1', '2022-11-05'),
      ('M5', 'Genre5', 'Sci-Fi Movie 1', '2022-09-30'),
      ('M6', 'Genre6', 'Romantic Movie 1', '2023-03-25'),
      ('M7', 'Genre7', 'Thriller Movie 1', '2023-04-08'),
      ('M8', 'Genre8', 'Adventure Movie 1', '2022-10-15'),
      ('M9', 'Genre9', 'Animation Movie 1', '2023-05-12'),
      ('M10', 'Genre10', 'Fantasy Movie 1', '2022-08-01');
```

Connected. (1/1) | DESKTOP-7U6QLLV (16.0 RTM) | DESKTOP-7U6QLLV\ASUS (66) | master | 00:00:00 | 0 rows

Ready

databaseCreation.sql - DESKTOP-7U6QLLV.master (DESKTOP-7U6QLLV\ASUS (66)) - Microsoft SQL Server Management Studio

```sql
-- Insert data into the SCREEN table
INSERT INTO SCREEN (ScreenID, ScreenName)
VALUES
  ('S1', 'Screen 1'),
  ('S2', 'Screen 2'),
  ('S3', 'Screen 3'),
  ('S4', 'Screen 4'),
  ('S5', 'Screen 5'),
  ('S6', 'Screen 6'),
  ('S7', 'Screen 7'),
  ('S8', 'Screen 8'),
  ('S9', 'Screen 9'),
  ('S10', 'Screen 10');

-- Insert data into the TICKETTYPE table
INSERT INTO TICKETTYPE (TicketTypeID, TicketTypeName, UnitPrice)
VALUES
  ('TT1', 'Standard', 10),
  ('TT2', 'VIP', 20),
  ('TT3', 'Child', 5),
  ('TT4', 'Senior', 7),
  ('TT5', 'Student', 8),
  ('TT6', 'Matinee', 6),
  ('TT7', 'Group', 15),
  ('TT8', 'Family', 25),
  ('TT9', 'Combo', 12),
  ('TT10', 'Special', 18);

-- Insert data into the ORDERDETAIL table
INSERT INTO ORDERDETAIL (OrderID, CustomerID, StaffID, Quantity)
```

Connected. (1/1) | DESKTOP-7U6QLLV (16.0 RTM) | DESKTOP-7U6QLLV\ASUS (66) | master | 00:00:00 | 0 rows

Ready

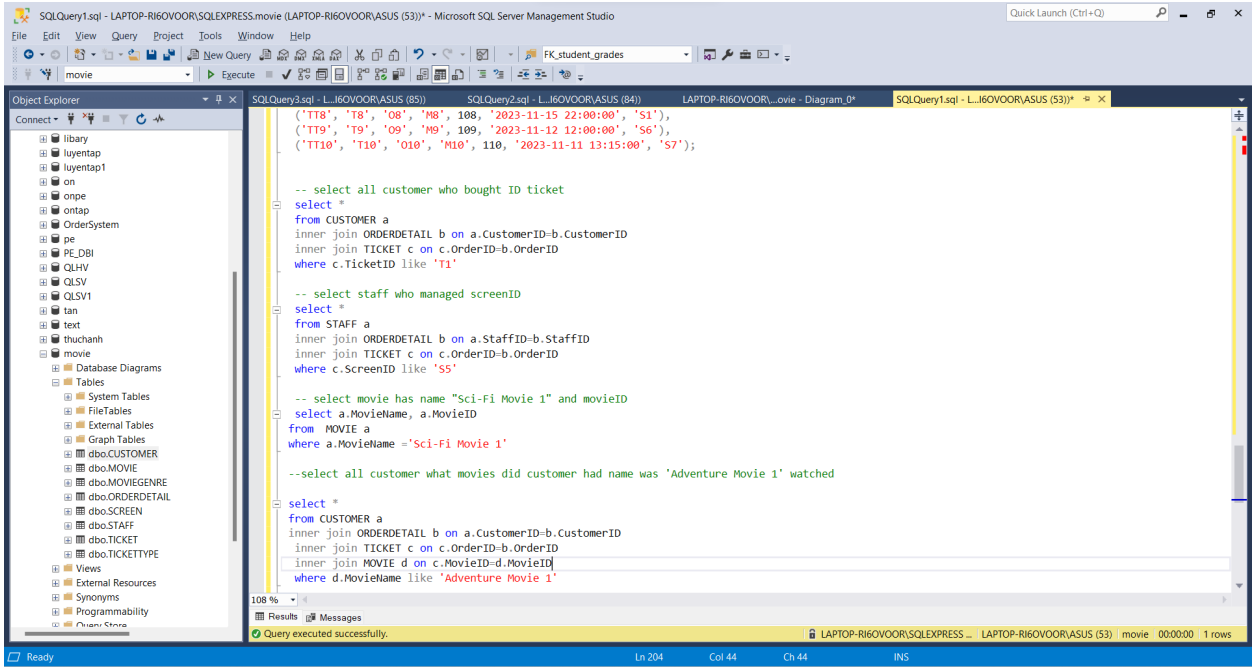
databaseCreation.sql - DESKTOP-7U6QLLV.master (DESKTOP-7U6QLLV\ASUS (66)) - Microsoft SQL Server Management Studio

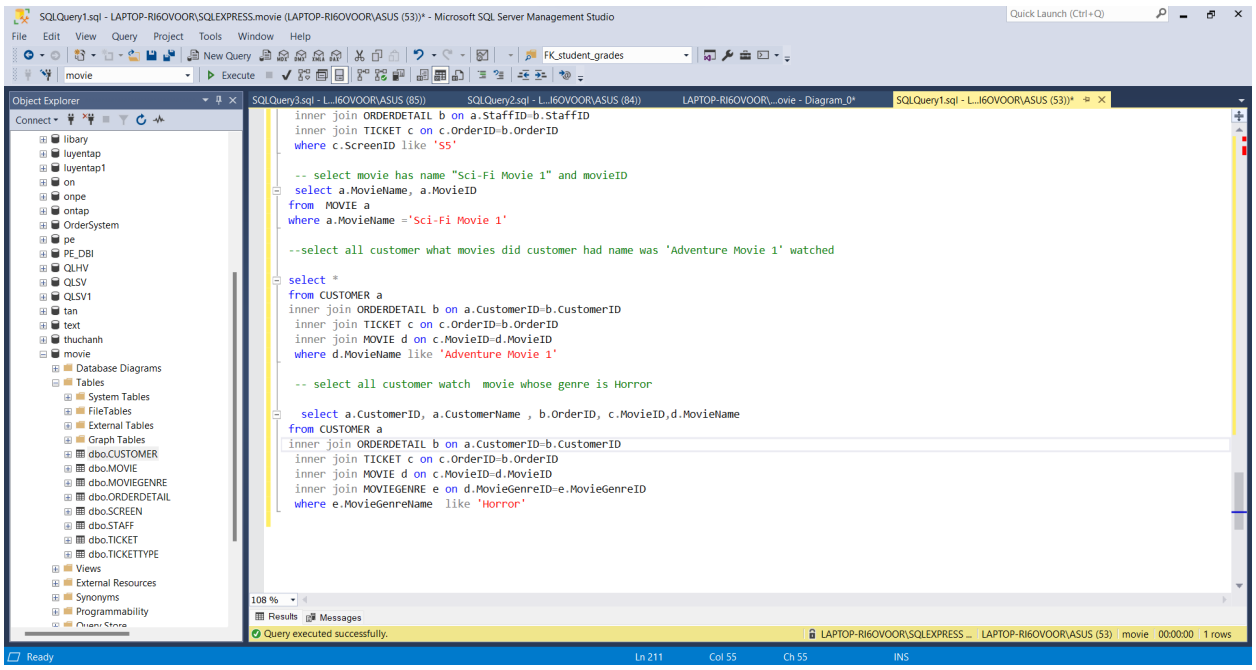
```sql
  ('TT10', 'Special', 18);

-- Insert data into the ORDERDETAIL table
INSERT INTO ORDERDETAIL (OrderID, CustomerID, StaffID, Quantity)
VALUES
  ('O1', 'C1', 'Staff1', 0),
  ('O2', 'C1', 'Staff2', 0),
  ('O3', 'C3', 'Staff3', 0),
  ('O4', 'C2', 'Staff4', 0),
  ('O5', 'C8', 'Staff5', 0),
  ('O6', 'C2', 'Staff6', 0),
  ('O7', 'C1', 'Staff7', 0),
  ('O8', 'C5', 'Staff8', 0),
  ('O9', 'C7', 'Staff9', 0),
  ('O10', 'C1', 'Staff10', 0);

-- Insert data into the TICKET table
INSERT INTO TICKET (TicketTypeID, TicketID, OrderID, MovieID, Seat, ShowingTime, ScreenID)
VALUES
  ('TT1', 'T1', 'O1', 'M1', 101, '2023-11-15 12:30:00', 'S1'),
  ('TT2', 'T2', 'O2', 'M2', 102, '2023-11-10 14:00:00', 'S2'),
  ('TT3', 'T3', 'O3', 'M3', 103, '2023-12-20 15:45:00', 'S3'),
  ('TT4', 'T4', 'O4', 'M4', 104, '2023-11-15 16:30:00', 'S4'),
  ('TT5', 'T5', 'O1', 'M5', 105, '2023-11-30 18:15:00', 'S5'),
  ('TT6', 'T6', 'O6', 'M6', 106, '2023-11-25 19:30:00', 'S2'),
  ('TT7', 'T7', 'O1', 'M7', 107, '2023-11-18 20:45:00', 'S5'),
  ('TT8', 'T8', 'O8', 'M8', 108, '2023-11-15 22:00:00', 'S1'),
  ('TT9', 'T9', 'O9', 'M9', 109, '2023-11-12 12:00:00', 'S6'),
  ('TT10', 'T10', 'O10', 'M10', 110, '2023-11-11 13:15:00', 'S7');
```

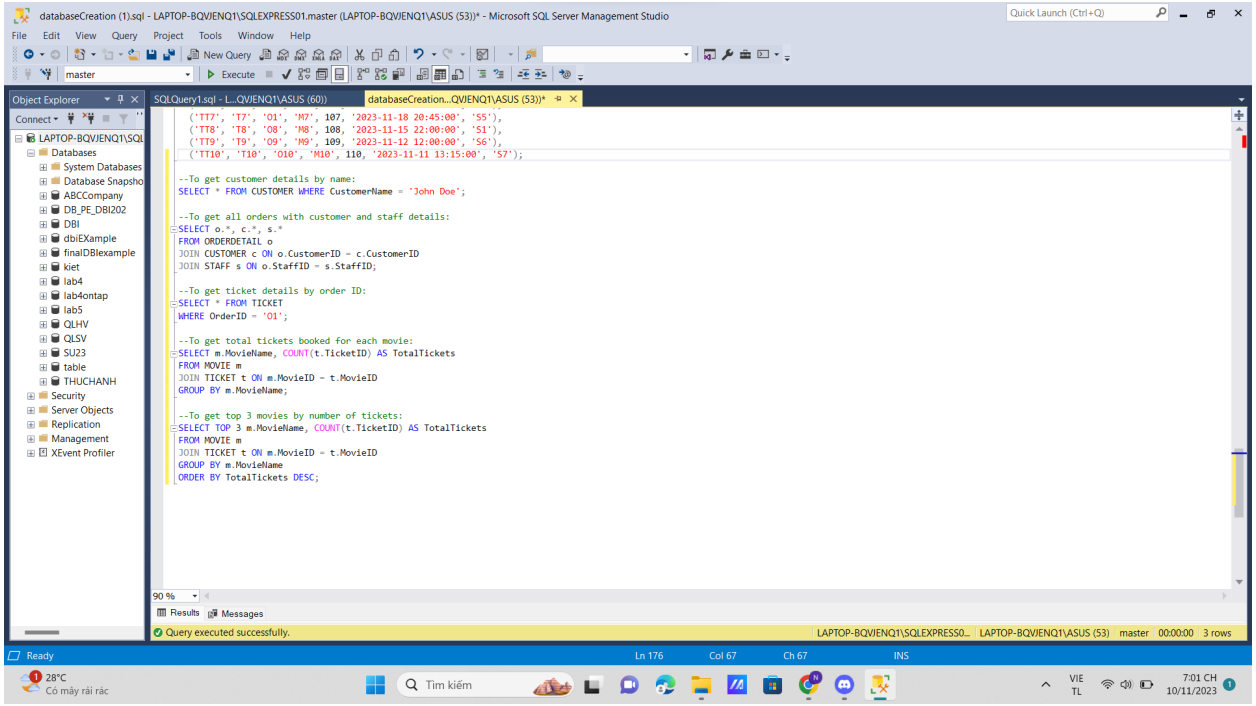
Connected. (1/1) | DESKTOP-7U6QLLV (16.0 RTM) | DESKTOP-7U6QLLV\ASUS (66) | master | 00:00:00 | 0 rows

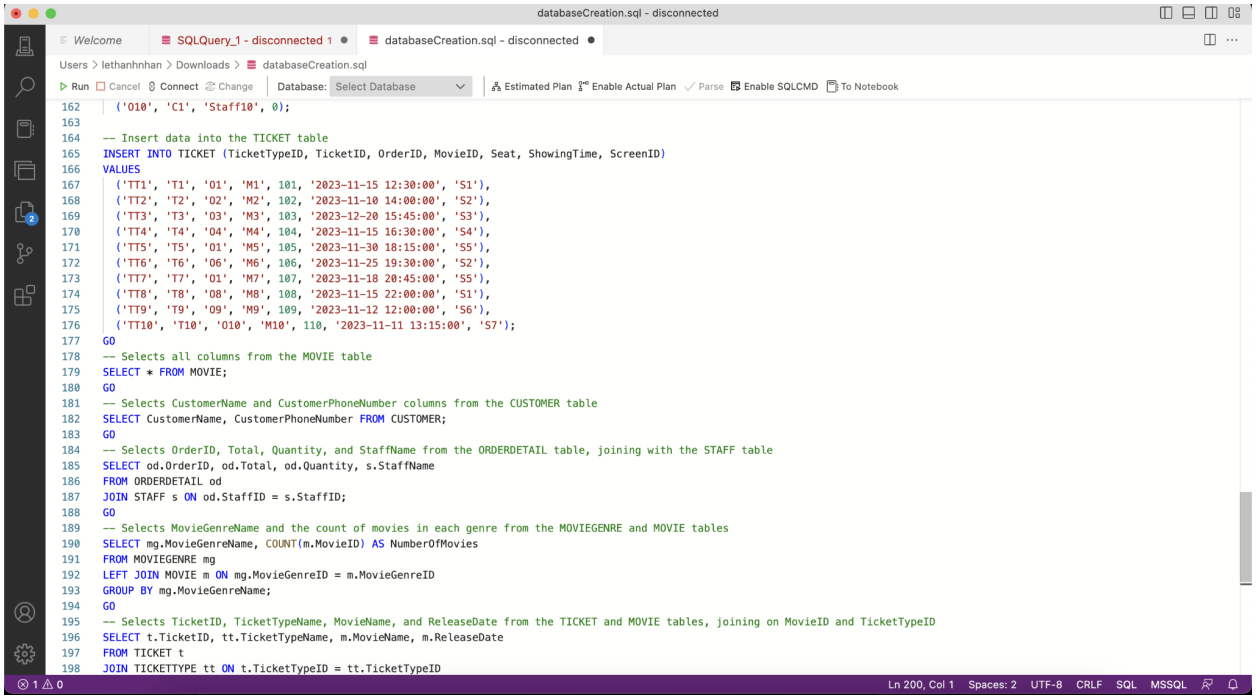
Ready

2. SELECT

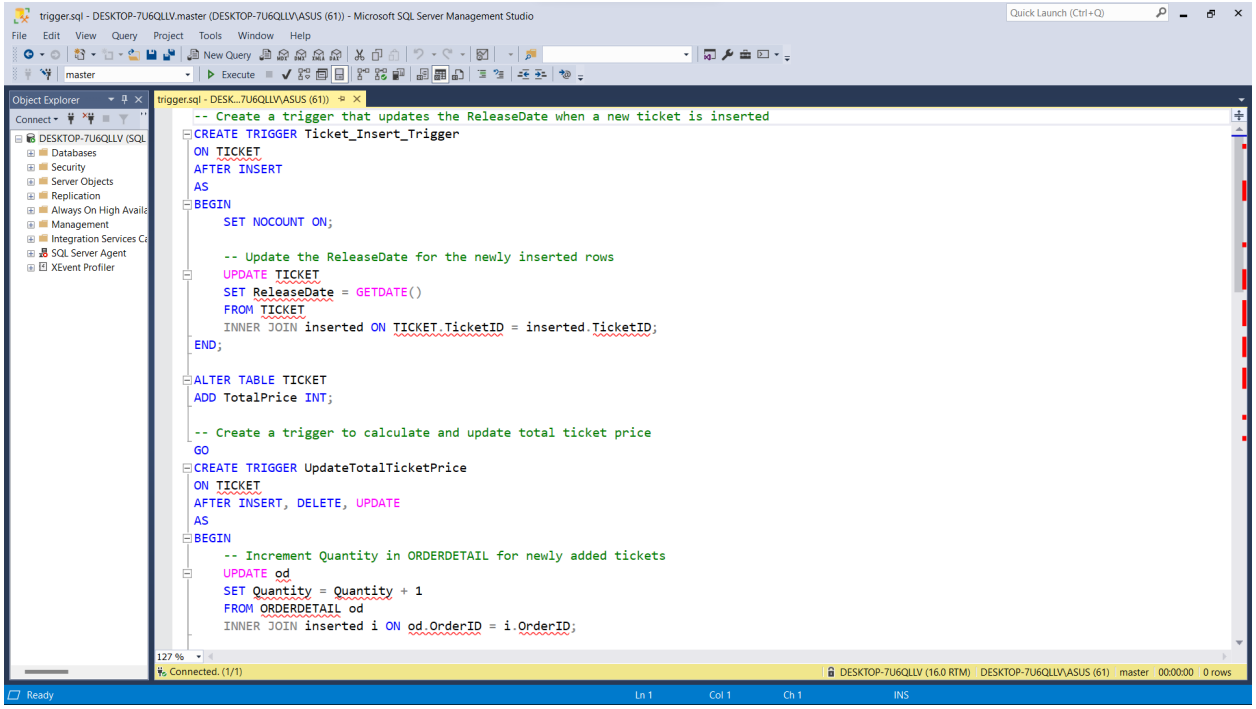

SQLQuery1.sql - LAPTOP-RI6OVOOR\SQLEXPRESS.movie (LAPTOP-RI6OVOOR\ASUS (53))* - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query
FK_student_grades
movie
Execute
Object Explorer
Connect
SQLQuery3.sql - L...I6OVOOR\ASUS (85))
SQLQuery2.sql - L...I6OVOOR\ASUS (84))
LAPTOP-RI6OVOOR\...ovie - Diagram_0*
SQLQuery1.sql - L...I6OVOOR\ASUS (53))*
('TT8', 'T8', 'O8', 'M8', 108, '2023-11-15 22:00:00', 'S1'),
('TT9', 'T9', 'O9', 'M9', 109, '2023-11-12 12:00:00', 'S6'),
('TT10', 'T10', 'O10', 'M10', 110, '2023-11-11 13:15:00', 'S7');
-- select all customer who bought ID ticket
select *
from CUSTOMER a
inner join ORDERDETAIL b on a.CustomerID=b.CustomerID
inner join TICKET c on c.OrderID=b.OrderID
where c.TicketID like 'T1'
-- select staff who managed screenID
select *
from STAFF a
inner join ORDERDETAIL b on a.StaffID=b.StaffID
inner join TICKET c on c.OrderID=b.OrderID
where c.ScreenID like 'S5'
-- select movie has name "Sci-Fi Movie 1" and movieID
select a.MovieName, a.MovieID
from MOVIE a
where a.MovieName ='Sci-Fi Movie 1'
--select all customer what movies did customer had name was 'Adventure Movie 1' watched
select *
from CUSTOMER a
inner join ORDERDETAIL b on a.CustomerID=b.CustomerID
inner join TICKET c on c.OrderID=b.OrderID
inner join MOVIE d on c.MovieID=d.MovieID
where d.MovieName like 'Adventure Movie 1'
Results
Messages
Query executed successfully.
Ready
Ln 204
Col 44
Ch 44
INS

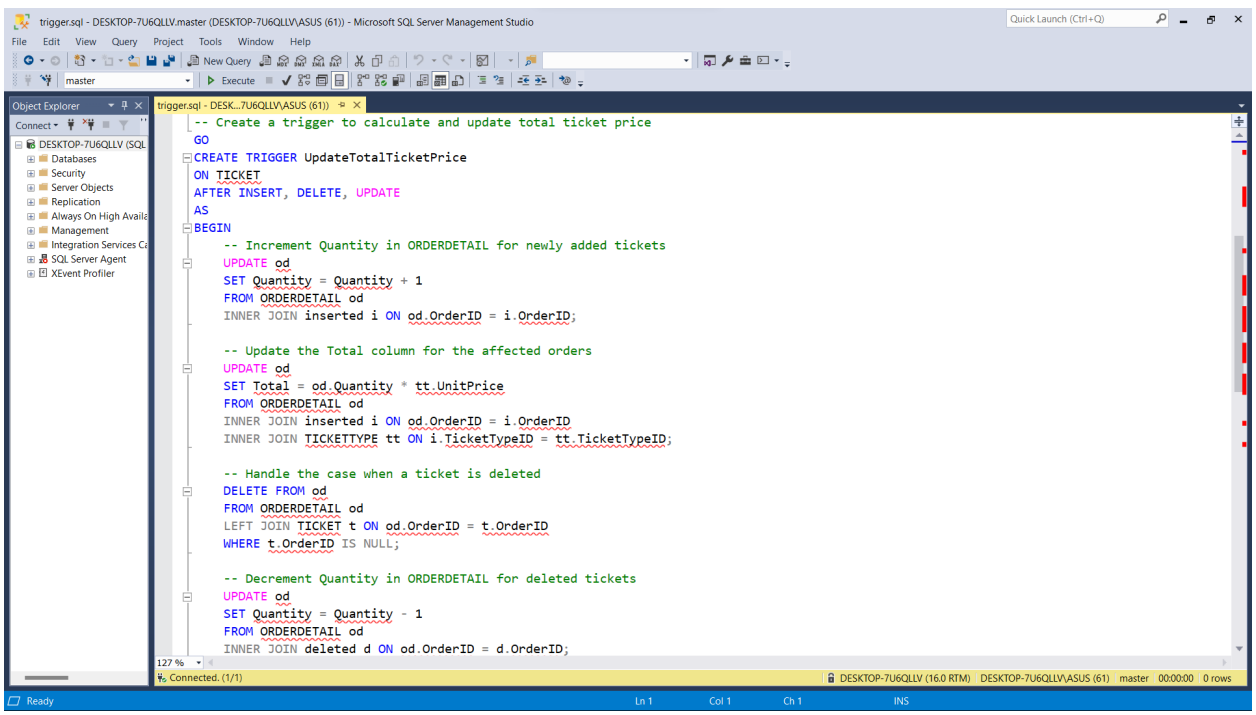

SQLQuery1.sql - LAPTOP-RI6OVOOR\SQLEXPRESS.movie (LAPTOP-RI6OVOOR\ASUS (53))* - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query
FK_student_grades
movie
Execute
Object Explorer
Connect
SQLQuery3.sql - L...I6OVOOR\ASUS (85))
SQLQuery2.sql - L...I6OVOOR\ASUS (84))
LAPTOP-RI6OVOOR\...ovie - Diagram_0*
SQLQuery1.sql - L...I6OVOOR\ASUS (53))*
inner join ORDERDETAIL b on a.StaffID=b.StaffID
inner join TICKET c on c.OrderID=b.OrderID
where c.ScreenID like 'S5'
-- select movie has name "Sci-Fi Movie 1" and movieID
select a.MovieName, a.MovieID
from MOVIE a
where a.MovieName ='Sci-Fi Movie 1'
--select all customer what movies did customer had name was 'Adventure Movie 1' watched
select *
from CUSTOMER a
inner join ORDERDETAIL b on a.CustomerID=b.CustomerID
inner join TICKET c on c.OrderID=b.OrderID
inner join MOVIE d on c.MovieID=d.MovieID
where d.MovieName like 'Adventure Movie 1'
-- select all customer watch movie whose genre is Horror
select a.CustomerID, a.CustomerName , b.OrderID, c.MovieID,d.MovieName
from CUSTOMER a
inner join ORDERDETAIL b on a.CustomerID=b.CustomerID
inner join TICKET c on c.OrderID=b.OrderID
inner join MOVIE d on c.MovieID=d.MovieID
inner join MOVIEGENRE e on d.MovieGenreID=e.MovieGenreID
where e.MovieGenreName like 'Horror'
Results
Messages
Query executed successfully.
Ready
Ln 211
Col 55
Ch 55
INS

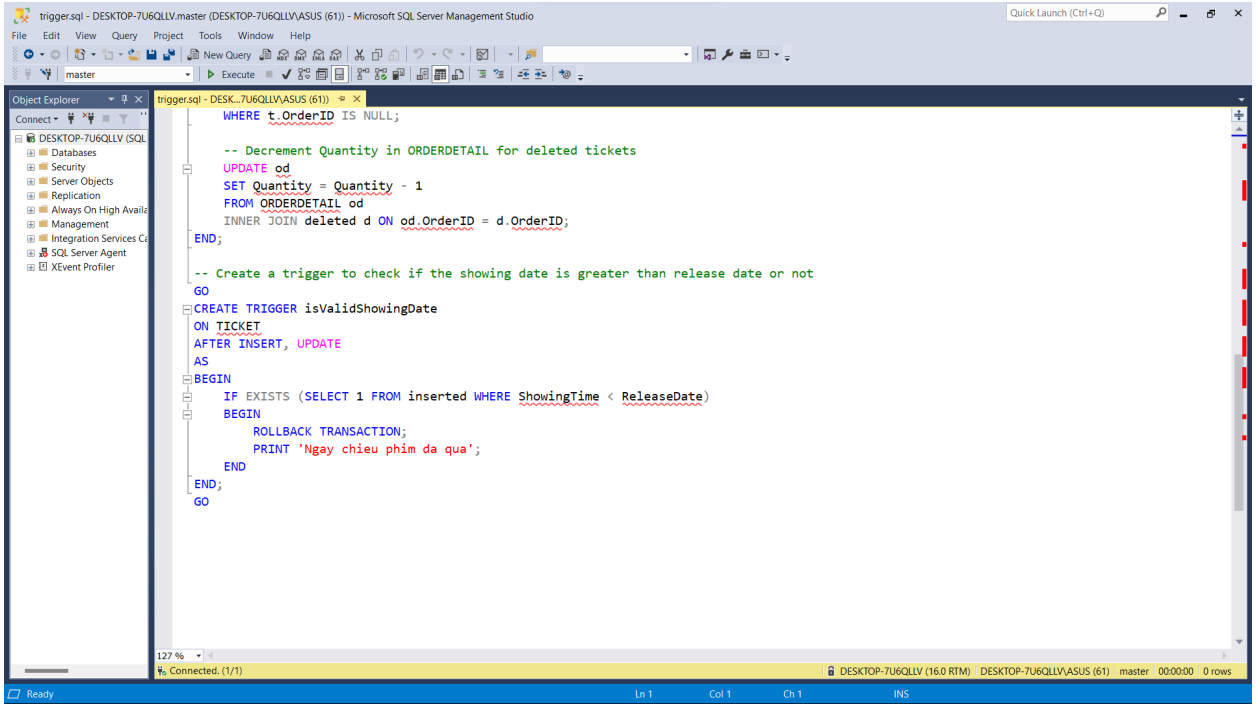

databaseCreation (1).sql - LAPTOP-BQVJENQ1\SQLEXPRESS01.master (LAPTOP-BQVJENQ1\ASUS (53))* - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query
Execute
master
Object Explorer
Connect
LAPTOP-BQVJENQ1\SQL
Databases
System Databases
Database Snapsho
ABCCompany
DB_PE_DBI202
DBI
dbiEXample
finalDBIexample
kiet
lab4
lab4ontap
lab5
QLHV
QLSV
SU23
table
THUCHANH
Security
Server Objects
Replication
Management
XEvent Profiler
SQLQuery1.sql - L...QVJENQ1\ASUS (60))
databaseCreation...QVJENQ1\ASUS (53))*
('TT7', 'T7', 'O1', 'M7', 107, '2023-11-18 20:45:00', 'S5'),
('TT8', 'T8', 'O8', 'M8', 108, '2023-11-15 22:00:00', 'S1'),
('TT9', 'T9', 'O9', 'M9', 109, '2023-11-12 12:00:00', 'S6'),
('TT10', 'T10', 'O10', 'M10', 110, '2023-11-11 13:15:00', 'S7');
--To get customer details by name:
SELECT * FROM CUSTOMER WHERE CustomerName = 'John Doe';
--To get all orders with customer and staff details:
SELECT o.*, c.*, s.*
FROM ORDERDETAIL o
JOIN CUSTOMER c ON o.CustomerID = c.CustomerID
JOIN STAFF s ON o.StaffID = s.StaffID;
--To get ticket details by order ID:
SELECT * FROM TICKET
WHERE OrderID = 'O1';
--To get total tickets booked for each movie:
SELECT m.MovieName, COUNT(t.TicketID) AS TotalTickets
FROM MOVIE m
JOIN TICKET t ON m.MovieID = t.MovieID
GROUP BY m.MovieName;
--To get top 3 movies by number of tickets:
SELECT TOP 3 m.MovieName, COUNT(t.TicketID) AS TotalTickets
FROM MOVIE m
JOIN TICKET t ON m.MovieID = t.MovieID
GROUP BY m.MovieName
ORDER BY TotalTickets DESC;
90 %
Results
Messages
Query executed successfully.
LAPTOP-BQVJENQ1\SQLEXPRESS0...
LAPTOP-BQVJENQ1\ASUS (53)
master
00:00:00
3 rows
Ready
Ln 176
Col 67
Ch 67
INS
28°C
Có mây rải rác
Tìm kiếm
VIE
TL
7:01 CH
10/11/2023


databaseCreation.sql - disconnected
Welcome
SQLQuery_1 - disconnected 1
databaseCreation.sql - disconnected
Users > lethanhnhan > Downloads > databaseCreation.sql
Run
Cancel
Connect
Change
Database: Select Database
Estimated Plan
Enable Actual Plan
Parse
Enable SQLCMD
To Notebook
('O10', 'C1', 'Staff10', 0);
-- Insert data into the TICKET table
INSERT INTO TICKET (TicketTypeID, TicketID, OrderID, MovieID, Seat, ShowingTime, ScreenID)
VALUES
('TT1', 'T1', 'O1', 'M1', 101, '2023-11-15 12:30:00', 'S1'),
('TT2', 'T2', 'O2', 'M2', 102, '2023-11-10 14:00:00', 'S2'),
('TT3', 'T3', 'O3', 'M3', 103, '2023-12-20 15:45:00', 'S3'),
('TT4', 'T4', 'O4', 'M4', 104, '2023-11-15 16:30:00', 'S4'),
('TT5', 'T5', 'O1', 'M5', 105, '2023-11-30 18:15:00', 'S5'),
('TT6', 'T6', 'O6', 'M6', 106, '2023-11-25 19:30:00', 'S2'),
('TT7', 'T7', 'O1', 'M7', 107, '2023-11-18 20:45:00', 'S5'),
('TT8', 'T8', 'O8', 'M8', 108, '2023-11-15 22:00:00', 'S1'),
('TT9', 'T9', 'O9', 'M9', 109, '2023-11-12 12:00:00', 'S6'),
('TT10', 'T10', 'O10', 'M10', 110, '2023-11-11 13:15:00', 'S7');
GO
-- Selects all columns from the MOVIE table
SELECT * FROM MOVIE;
GO
-- Selects CustomerName and CustomerPhoneNumber columns from the CUSTOMER table
SELECT CustomerName, CustomerPhoneNumber FROM CUSTOMER;
GO
-- Selects OrderID, Total, Quantity, and StaffName from the ORDERDETAIL table, joining with the STAFF table
SELECT od.OrderID, od.Total, od.Quantity, s.StaffName
FROM ORDERDETAIL od
JOIN STAFF s ON od.StaffID = s.StaffID;
GO
-- Selects MovieGenreName and the count of movies in each genre from the MOVIEGENRE and MOVIE tables
SELECT mg.MovieGenreName, COUNT(m.MovieID) AS NumberOfMovies
FROM MOVIEGENRE mg
LEFT JOIN MOVIE m ON mg.MovieGenreID = m.MovieGenreID
GROUP BY mg.MovieGenreName;
GO
-- Selects TicketID, TicketTypeName, MovieName, and ReleaseDate from the TICKET and MOVIE tables, joining on MovieID and TicketTypeID
SELECT t.TicketID, tt.TicketTypeName, m.MovieName, m.ReleaseDate
FROM TICKET t
JOIN TICKETTYPE tt ON t.TicketTypeID = tt.TicketTypeID
1 0
Ln 200, Col 1
Spaces: 2
UTF-8
CRLF
SQL
MSSQL

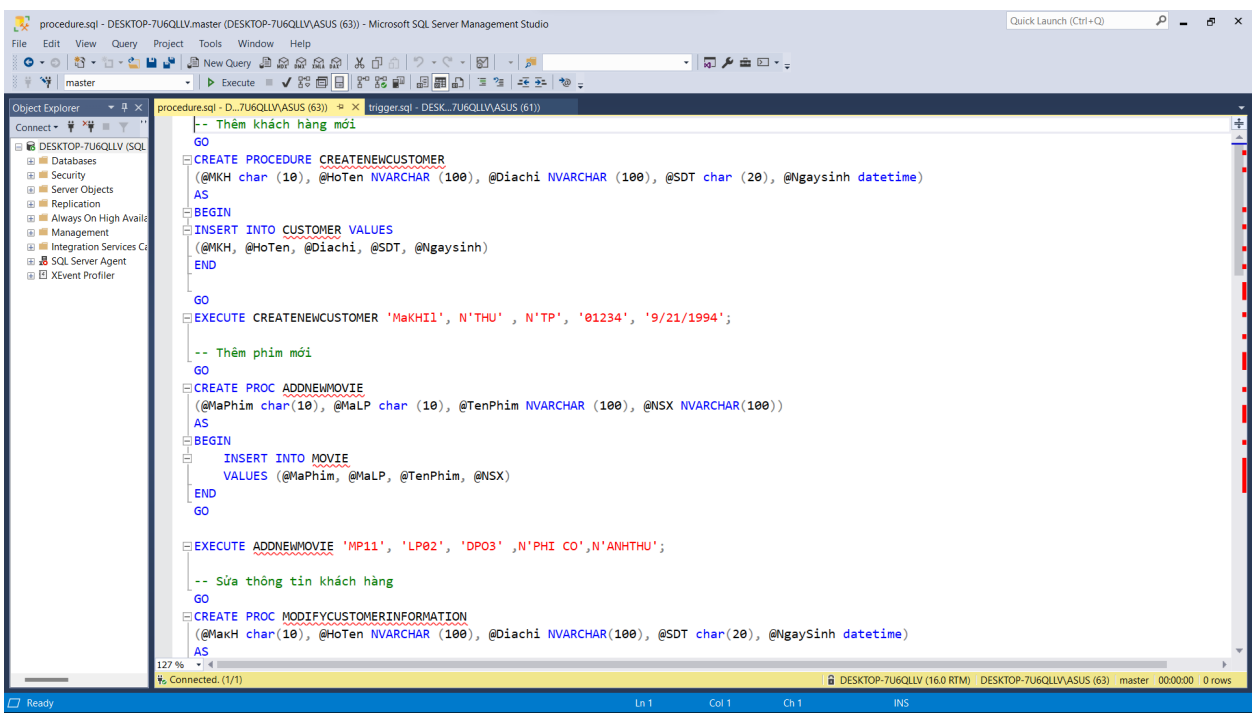
3. TRIGGER


trigger.sql - DESKTOP-7U6QLLV.master (DESKTOP-7U6QLLV\ASUS (61)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query
Execute
master
Object Explorer
Connect
DESKTOP-7U6QLLV (SQL
Databases
Security
Server Objects
Replication
Always On High Availa
Management
Integration Services Ca
SQL Server Agent
XEvent Profiler
trigger.sql - DESK...7U6QLLV\ASUS (61))
-- Create a trigger that updates the ReleaseDate when a new ticket is inserted
CREATE TRIGGER Ticket_Insert_Trigger
ON TICKET
AFTER INSERT
AS
BEGIN
SET NOCOUNT ON;
-- Update the ReleaseDate for the newly inserted rows
UPDATE TICKET
SET ReleaseDate = GETDATE()
FROM TICKET
INNER JOIN inserted ON TICKET.TicketID = inserted.TicketID;
END;
ALTER TABLE TICKET
ADD TotalPrice INT;
-- Create a trigger to calculate and update total ticket price
GO
CREATE TRIGGER UpdateTotalTicketPrice
ON TICKET
AFTER INSERT, DELETE, UPDATE
AS
BEGIN
-- Increment Quantity in ORDERDETAIL for newly added tickets
UPDATE od
SET Quantity = Quantity + 1
FROM ORDERDETAIL od
INNER JOIN inserted i ON od.OrderID = i.OrderID;
127 %
Connected. (1/1)
DESKTOP-7U6QLLV (16.0 RTM)
DESKTOP-7U6QLLV\ASUS (61)
master
00:00:00
0 rows
Ready
Ln 1
Col 1
Ch 1
INS


trigger.sql - DESKTOP-7U6QLLV.master (DESKTOP-7U6QLLV\ASUS (61)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query
Execute
master
Object Explorer
Connect
DESKTOP-7U6QLLV (SQL
Databases
Security
Server Objects
Replication
Always On High Availa
Management
Integration Services Ca
SQL Server Agent
XEvent Profiler
trigger.sql - DESK...7U6QLLV\ASUS (61))
-- Create a trigger to calculate and update total ticket price
GO
CREATE TRIGGER UpdateTotalTicketPrice
ON TICKET
AFTER INSERT, DELETE, UPDATE
AS
BEGIN
-- Increment Quantity in ORDERDETAIL for newly added tickets
UPDATE od
SET Quantity = Quantity + 1
FROM ORDERDETAIL od
INNER JOIN inserted i ON od.OrderID = i.OrderID;
-- Update the Total column for the affected orders
UPDATE od
SET Total = od.Quantity * tt.UnitPrice
FROM ORDERDETAIL od
INNER JOIN inserted i ON od.OrderID = i.OrderID
INNER JOIN TICKETTYPE tt ON i.TicketTypeID = tt.TicketTypeID;
-- Handle the case when a ticket is deleted
DELETE FROM od
FROM ORDERDETAIL od
LEFT JOIN TICKET t ON od.OrderID = t.OrderID
WHERE t.OrderID IS NULL;
-- Decrement Quantity in ORDERDETAIL for deleted tickets
UPDATE od
SET Quantity = Quantity - 1
FROM ORDERDETAIL od
INNER JOIN deleted d ON od.OrderID = d.OrderID;
127 %
Connected. (1/1)
DESKTOP-7U6QLLV (16.0 RTM)
DESKTOP-7U6QLLV\ASUS (61)
master
00:00:00
0 rows
Ready
Ln 1
Col 1
Ch 1
INS


trigger.sql - DESKTOP-7U6QLLV.master (DESKTOP-7U6QLLV\ASUS (61)) - Microsoft SQL Server Management Studio
trigger.sql - DESK...7U6QLLV\ASUS (61))
WHERE t.OrderID IS NULL;
-- Decrement Quantity in ORDERDETAIL for deleted tickets
UPDATE od
SET Quantity = Quantity - 1
FROM ORDERDETAIL od
INNER JOIN deleted d ON od.OrderID = d.OrderID;
END;
-- Create a trigger to check if the showing date is greater than release date or not
GO
CREATE TRIGGER isValidShowingDate
ON TICKET
AFTER INSERT, UPDATE
AS
BEGIN
IF EXISTS (SELECT 1 FROM inserted WHERE ShowingTime < ReleaseDate)
BEGIN
ROLLBACK TRANSACTION;
PRINT 'Ngay chieu phim da qua';
END
END;
GO

4. PROCEDURE


procedure.sql - DESKTOP-7U6QLLV.master (DESKTOP-7U6QLLV\ASUS (63)) - Microsoft SQL Server Management Studio
procedure.sql - D...7U6QLLV\ASUS (63))
trigger.sql - DESK...7U6QLLV\ASUS (61))
-- Thêm khách hàng mới
GO
CREATE PROCEDURE CREATENEWCUSTOMER
(@MKH char (10), @HoTen NVARCHAR (100), @Diachi NVARCHAR (100), @SDT char (20), @Ngaysinh datetime)
AS
BEGIN
INSERT INTO CUSTOMER VALUES
(@MKH, @HoTen, @Diachi, @SDT, @Ngaysinh)
END
GO
EXECUTE CREATENEWCUSTOMER 'MaKHIl', N'THU' , N'TP', '01234', '9/21/1994';
-- Thêm phim mới
GO
CREATE PROC ADDNEWMOVIE
(@MaPhim char(10), @MaLP char (10), @TenPhim NVARCHAR (100), @NSX NVARCHAR(100))
AS
BEGIN
INSERT INTO MOVIE
VALUES (@MaPhim, @MaLP, @TenPhim, @NSX)
END
GO
EXECUTE ADDNEWMOVIE 'MP11', 'LP02', 'DPO3' ,N'PHI CO',N'ANHTHU';
-- Sửa thông tin khách hàng
GO
CREATE PROC MODIFYCUSTOMERINFORMATION
(@MaKH char(10), @HoTen NVARCHAR (100), @Diachi NVARCHAR(100), @SDT char(20), @NgaySinh datetime)
AS

```sql
EXECUTE ADDNEWMOVIE 'MP11', 'LP02', 'DPO3' ,N'PHI CO',N'ANHTHU';

-- Sửa thông tin khách hàng
GO
CREATE PROC MODIFYCUSTOMERINFORMATION
(@MaKH char(10), @HoTen NVARCHAR (100), @Diachi NVARCHAR(100), @SDT char(20), @NgaySinh datetime)
AS
BEGIN
    UPDATE CUSTOMER
    SET CustomerName = @HoTen, CustomerAddress = @DiaChi, CustomerPhoneNumber = @SDT, CustomerDoB = @NgaySinh
    FROM CUSTOMER
    WHERE CustomerID = @MaKH
END
GO

EXECUTE SUAKH @MaKH='MKH09', CHoTen= N'HUỲNH THU',
@Diachi-N'TP', @SDT='01234', @NgaySinh=19/21/19941

-- Sửa thông tin nhân viên
GO
CREATE PROC MODIFYSTAFFINFORMATION
(@MaNV char (10), @HoTen NVARCHAR (100), @Diachi NVARCHAR (100), @SDT char (20), @Ngaysinh datetime, @Gt NVARCHAR (10))
AS
BEGIN
    UPDATE STAFF
    SET StaffID = @HoTen, StaffAddress = @Diachi, StaffPhoneNumber = @SDT, StaffDoB = @Ngaysinh, Gender = @Gt
    FROM STAFF
    WHERE StaffID = @MaNV
END
GO
```

```sql
    WHERE StaffID = @MaNV
END
GO

-- Sửa thông tin phim
CREATE PROC MODIFYMOVIEINFORMATION
(@MaPhim char(10), @MALP char(10), @TenPhim NVARCHAR (100), @NSX NVARCHAR(100))
AS
BEGIN
    UPDATE MOVIE
    SET MovieID = @MaPhim, MovieGenreID = @MALP, MovieName = @TenPhim, ReleaseDate = @NSX
    FROM MOVIE
    WHERE MovieID = @MaPhim
END

-- Liệt kê số lượt xem của 1 bộ phim
GO
CREATE PROC LXP
(@tenphim NVARCHAR (100))
AS
BEGIN
    SELECT MovieName, COUNT(TicketID) As 'Luot xem'
    FROM ORDERDETAIL a
    INNER JOIN T
    ON  T.OrderID = a.OrderID
    INNER JOIN MOVIE b
    ON T.MovieID = b.MovieID
    WHERE MovieName = @tenphim
    GROUP BY MovieName
END;
```

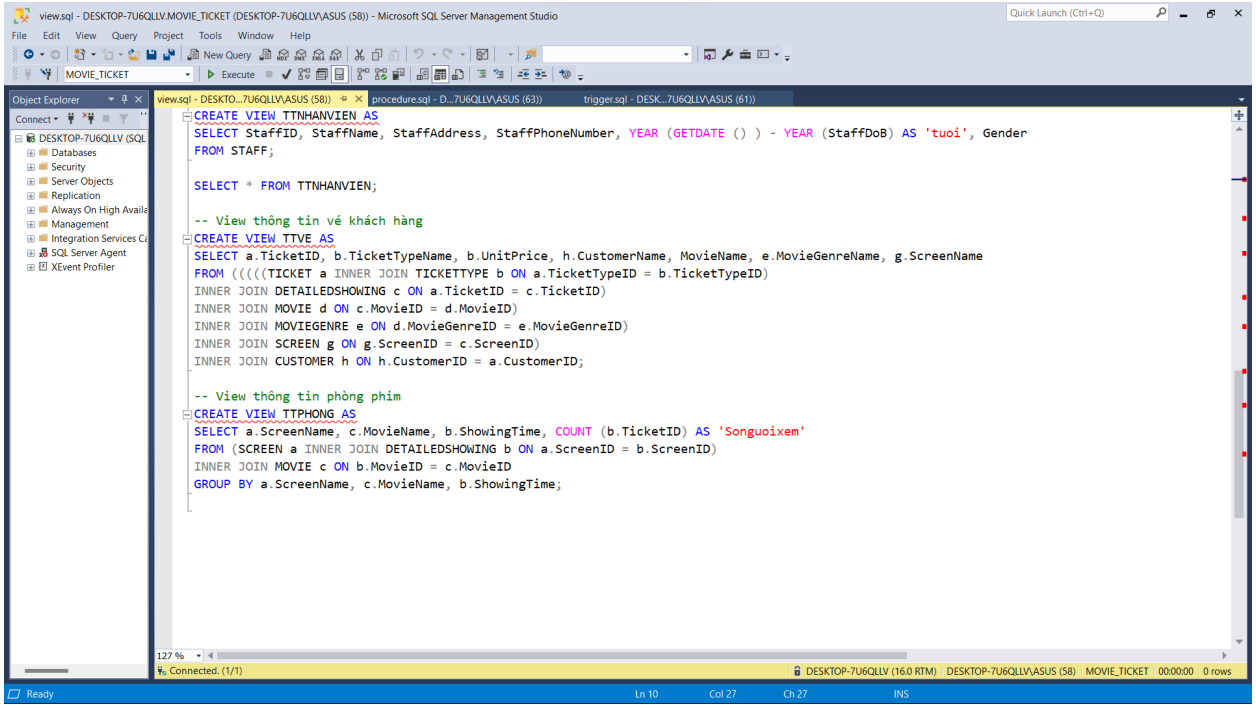
5. VIEW

view.sql - DESKTOP-7U6QLLV.MOVIE_TICKET (DESKTOP-7U6QLLV\ASUS (58)) - Microsoft SQL Server Management Studio

```sql
-- View thông tin doanh thu của từng loại vé
CREATE VIEW DOANHTHU AS
SELECT a.TicketTypeID, COUNT (a.TicketID) AS 'Soluong' , SUM (b.UnitPrice) AS 'tongthu'
FROM TICKET a INNER JOIN TICKETTYPE b
ON a.TicketTypeID = b.TicketTypeID
GROUP BY a.TicketID, a.TicketTypeID;

-- View thông tin khách hàng
GO
CREATE VIEW TTKHACHHANG AS
SELECT a.CustomerID, a.CustomerName, a.CustomerAddress, a.CustomerPhoneNumber, YEAR (GETDATE () ) - YEAR (CustomerDoB) AS 'Tuoi', b.TicketID,
FROM (((CUSTOMER a INNER JOIN TICKET b ON a.CustomerID = b.CustomerID)
INNER JOIN DETAILEDSHOWING c ON b.TicketID = c.TicketID)
INNER JOIN MOVIE d ON c.MovieID = d.MovieID)
INNER JOIN SCREEN e ON c.ScreenID = e.ScreenID;

-- View thông tin các phim
CREATE VIEW TTPHIM AS
SELECT DISTINCT a.MovieID, a.MovieName, b.MovieGenreName, a.ReleaseDate
FROM (MOVIE a INNER JOIN MOVIEGENRE b ON a.MovieGenreID = b.MovieGenreID);

SELECT * FROM TTPHIM;

-- VIEW thông tin phim đã được chiếu
CREATE VIEW TTPHIMDACHIEU AS
SELECT DISTINCT a.MovieID, a.MovieName, b.MovieGenreName, a.ReleaseDate, d.ShowingTime, e.ScreenName
FROM (((MOVIE a INNER JOIN MOVIEGENRE b ON a.MovieGenreID = b.MovieGenreID)
    INNER JOIN DETAILEDSHOWING d ON a.MovieID = d.MovieID)
    INNER JOIN SCREEN e ON e.ScreenID = d.ScreenID);

SELECT * FROM TTPHIMDACHIEU;
```

view.sql - DESKTOP-7U6QLLV.MOVIE_TICKET (DESKTOP-7U6QLLV\ASUS (58)) - Microsoft SQL Server Management Studio

```sql
SELECT * FROM TTPHIMDACHIEU;

-- View thông tin nhân viên bán được vé
CREATE VIEW TTNVBANVE AS
SELECT a.StaffID, a.StaffName, a.StaffAddress, a.StaffPhoneNumber, YEAR (GETDATE () ) - YEAR (StaffDoB) AS 'Tuoi', COUNT (b.TicketID) AS 'sol
FROM STAFF a INNER JOIN TICKET b ON a.StaffID = b.StaffID
GROUP BY a.StaffID, a.StaffName, a.StaffAddress, a.StaffPhoneNumber, a.StaffDoB;

-- View thông tin lượt xem phim
CREATE VIEW LUOTXEMPHIM AS
SELECT b.MovieID, b.MovieName, c.MovieGenreName, COUNT (a.TicketID) AS 'soluotxem'
FROM ( (DETAILEDSHOWING a INNER JOIN MOVIE b ON a.MovieID = b.MovieID)
INNER JOIN MOVIEGENRE c ON b.MovieGenreID = C.MovieGenreID)
GROUP BY b.MovieID, b.MovieName, c.MovieGenreName;

SELECT * FROM LUOTXEMPHIM;

-- VIEW thong tin tất ca các nhân viên
CREATE VIEW TTNHANVIEN AS
SELECT StaffID, StaffName, StaffAddress, StaffPhoneNumber, YEAR (GETDATE () ) - YEAR (StaffDoB) AS 'tuoi', Gender
FROM STAFF;

SELECT * FROM TTNHANVIEN;

-- View thông tin vé khách hàng
CREATE VIEW TTVE AS
SELECT a.TicketID, b.TicketTypeName, b.UnitPrice, h.CustomerName, MovieName, e.MovieGenreName, g.ScreenName
FROM (((((TICKET a INNER JOIN TICKETTYPE b ON a.TicketTypeID = b.TicketTypeID)
INNER JOIN DETAILEDSHOWING c ON a.TicketID = c.TicketID)
INNER JOIN MOVIE d ON c.MovieID = d.MovieID)
INNER JOIN MOVIEGENRE e ON d.MovieGenreID = e.MovieGenreID)
```



VII/ CONCLUSION

- In summary, the movie theater management system, with its well-defined tables, relationships, and constraints, provides a solid foundation for effective and organized data handling in a dynamic cinema setting. Its structured design enables seamless management of movies, tickets, customers, staff, orders, screens, genres, and ticket types, contributing to a streamlined and efficient movie-going experience for both customers and cinema staff.