Offline Cinema Ticket Booking

Group 1

Names	:	Kheyral Sutan Dumas
		Zaky Al Fitra
		Rizqika Nurrahma Faira
Class	:	1CC7

CEP CCIT

FAKULTAS TEKNIK UNIVERSITAS INDONESIA

2023

PROJECT ON

Offline Cinema Ticket Booking

Developed by

1. **Kheyral Sutan Dumas**
2. **Zaky Al Fitra**
3. **Rizqika Nurrahma Faira**

Offline Cinema Ticket Booking

Batch Code	: 1CC7
Start Date	: December 18th, 2023
End Date	: December 27th, 2023
Name of Faculty	: Bayu Citra Nur Aulia, S.T., M.TI.

Names of Developer :

1. Kheyral Sutan Dumas
2. Zaky Al Fitra
3. Rizqika Nurrahma Faira

Date of Submission: December 27 th, 2023

NIIT

CERTIFICATE

This is to certify that this report titled “Cinema Ticket Booking” embodies the original work done by Kheyral Sutan Dumas, Zaky Al Fitra and Rizqika Nurrahma Faira. Project in partial fulfillment of their course requirement at NIIT.

Coordinator:

Bayu Citra Nur Aulia, S.T, M.TI.

ACKNOWLEDGEMENT

Thanks to God Almighty's abundant compassion and grace, the writer was able to finish this job on time, both in terms of the paper and the presentations. We would like to extend a special thank you to the professors, Mrs. Bayu Citra Nur Aulia, S.T, M.TI. and other professors, who continuously assist and direct the authors in completing this project paper. We appreciate the support from our other students and your participation in this education at CCIT-FTUI.

The writers of the project paper "Offline Cinema Ticket Booking" suggest it as the third project for the year 2023's work needs. The author hopes that the reader will find this material beneficial and that it will bring knowledge and insight to their life. The author is aware that this essay is not flawless.

Therefore, in order to enhance this article, the authors welcome any and all helpful comments and recommendations from the reader. Finally, it is hoped that the readers of this work will get a variety of advantages.

Depok, December 27st 2023

TABLE OF CONTENT

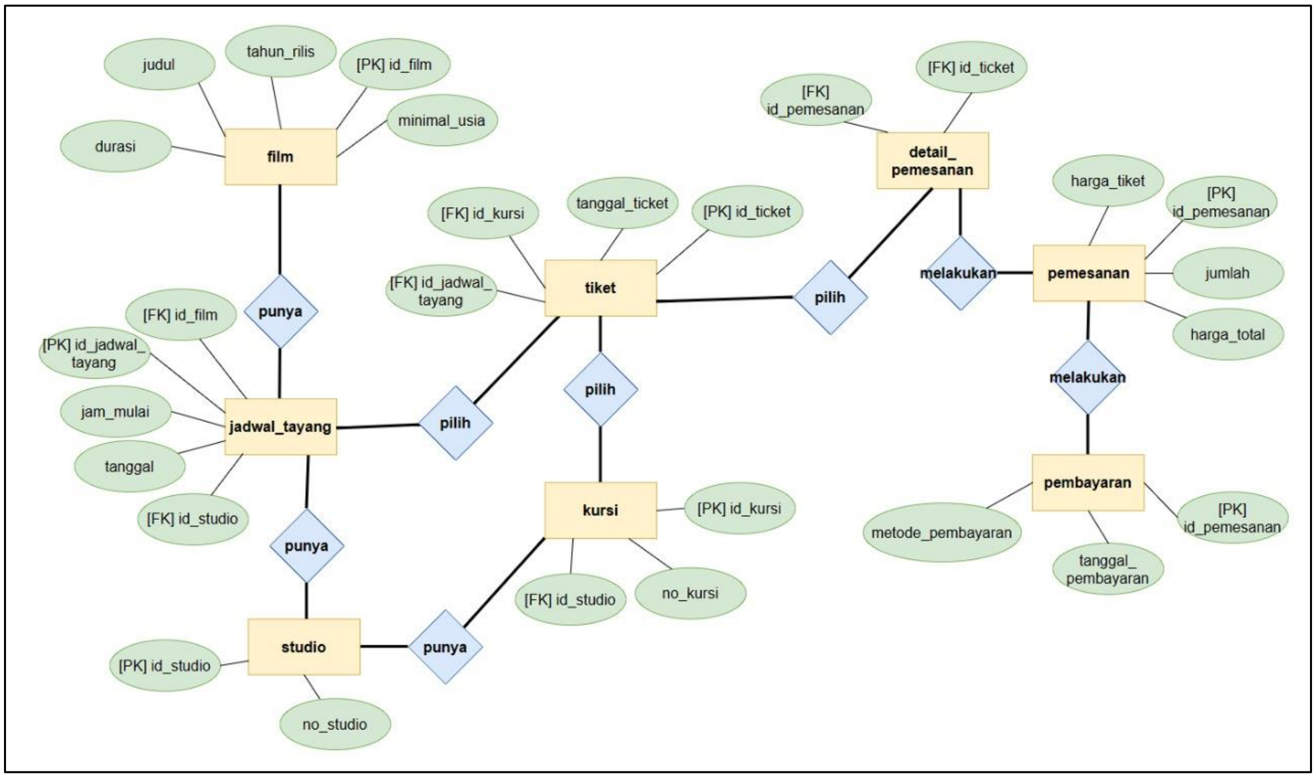
SYSTEM ANALYSIS

System Summary :

The offline movie ticket booking system is a conventional approach in which customers can go directly to the physical cinema to buy tickets in person. Although the online system is becoming increasingly popular, booking movies tickets offline still has certain advantages. Users who come directly to the movie theater locations can check the movie schedules live, see seat availability, and get detailed information about specific offers or discounts.

This process provides a unique and more personal interactive experience for customers, especially those who prefer to plan a movie theater visit spontaneously. Nonetheless, the weaknesses of the off-line system involve the potential for delays and time constraints, as well as the limitations to monitor the schedules and the capacity of the cinemas efficiently. In this analysis, it is important to consider customer preferences and operational challenges to optimize the offline movie ticket booking experience.

ENTITY RELATIONSHIP DIAGRAM



DATABASE DESIGN

Database Name: Cinema Ticket Booking

Number of Procedure: **5**

1. Procedure for Inserting New Movies
2. Procedure Add New Ticket
3. Procedure New Booking
4. Procedure Detail Order
5. Procedure Payment

Number of Trigger: 1

1. Trigger Automatic Update Ticket Price and Calculate IT in Booking

Number of Tables: 8 Table Names:

1. Movies
2. Ticket
3. Booking
4. Studio
5. Schedule
6. Seat
7. Payment
8. Order Detail

SCHEMATIC DIAGRAM

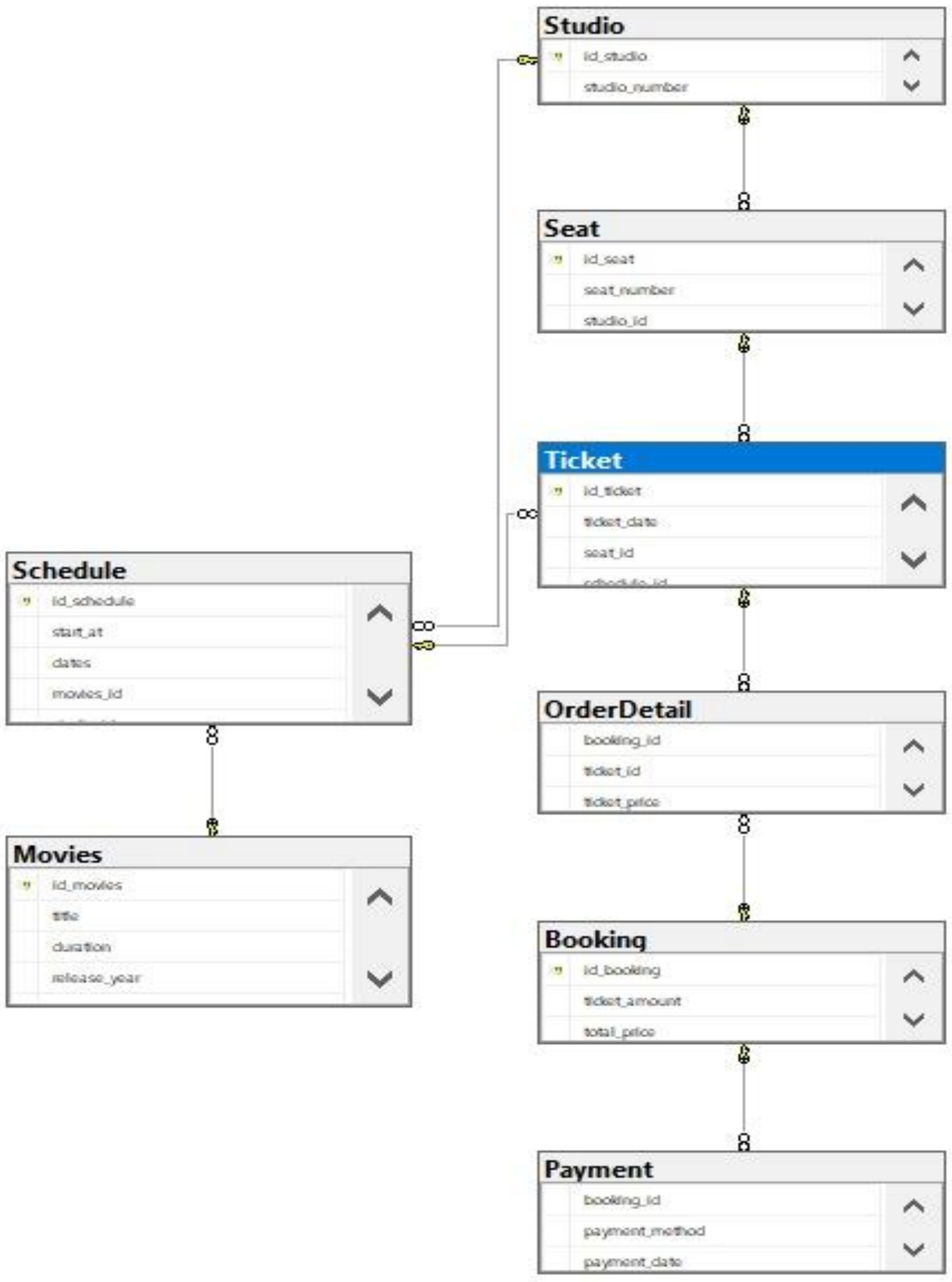
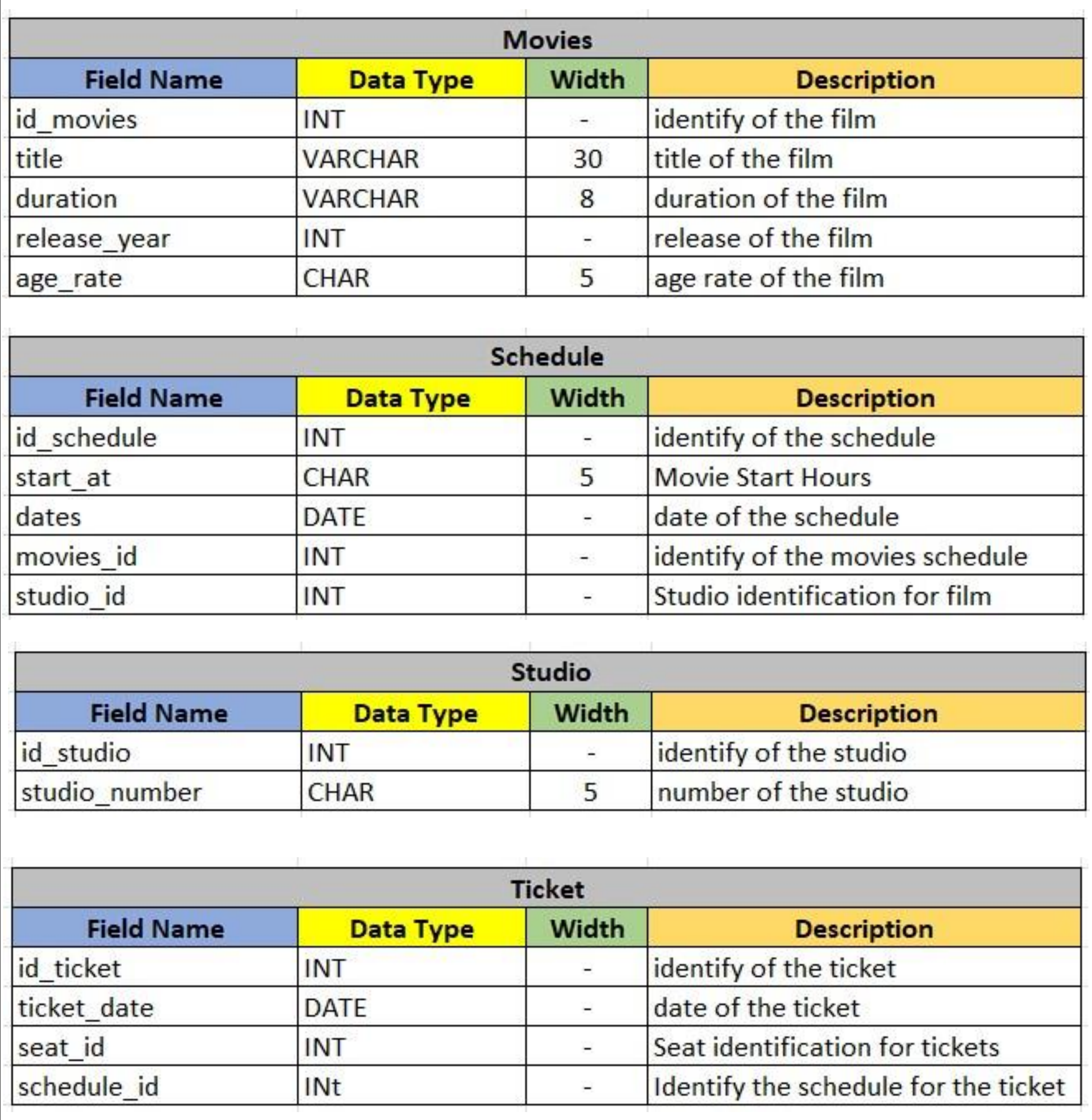


TABLE DESIGN



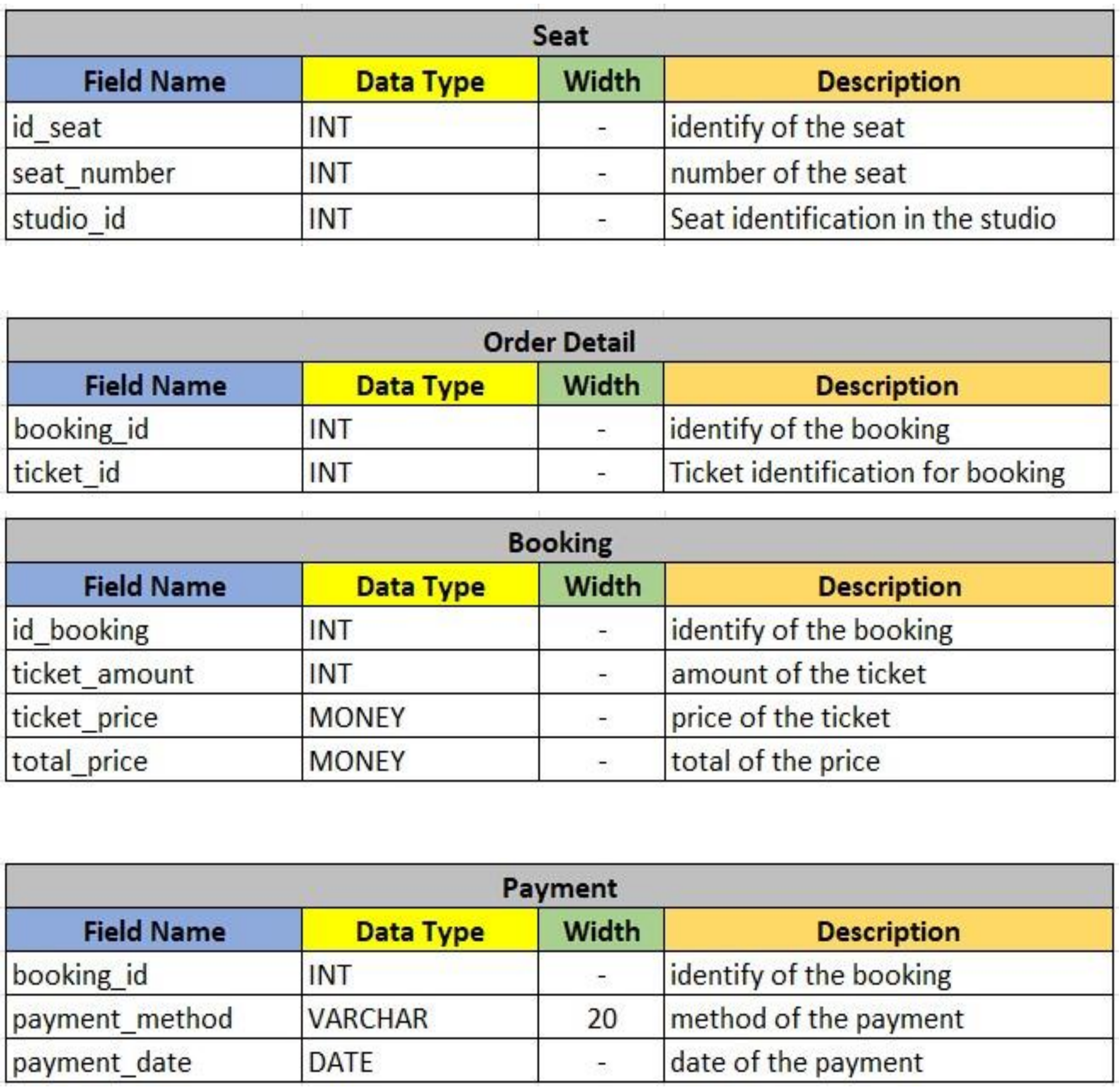
Movies			
Field Name	Data Type	Width	Description
id_movies	INT	-	identify of the film
title	VARCHAR	30	title of the film
duration	VARCHAR	8	duration of the film
release_year	INT	-	release of the film
age_rate	CHAR	5	age rate of the film

Schedule			
Field Name	Data Type	Width	Description
id_schedule	INT	-	identify of the schedule
start_at	CHAR	5	Movie Start Hours
dates	DATE	-	date of the schedule
movies_id	INT	-	identify of the movies schedule
studio_id	INT	-	Studio identification for film

Studio			
Field Name	Data Type	Width	Description
id_studio	INT	-	identify of the studio
studio_number	CHAR	5	number of the studio

Ticket			
Field Name	Data Type	Width	Description
id_ticket	INT	-	identify of the ticket
ticket_date	DATE	-	date of the ticket
seat_id	INT	-	Seat identification for tickets
schedule_id	INt	-	Identify the schedule for the ticket

TABLE DESIGN



Seat			
Field Name	**Data Type**	**Width**	**Description**
id_seat	INT	-	identify of the seat
seat_number	INT	-	number of the seat
studio_id	INT	-	Seat identification in the studio

Order Detail			
Field Name	**Data Type**	**Width**	**Description**
booking_id	INT	-	identify of the booking
ticket_id	INT	-	Ticket identification for booking

Booking			
Field Name	**Data Type**	**Width**	**Description**
id_booking	INT	-	identify of the booking
ticket_amount	INT	-	amount of the ticket
ticket_price	MONEY	-	price of the ticket
total_price	MONEY	-	total of the price

Payment			
Field Name	**Data Type**	**Width**	**Description**
booking_id	INT	-	identify of the booking
payment_method	VARCHAR	20	method of the payment
payment_date	DATE	-	date of the payment

SCRIPT

```
------------------- C R E A T I N G  D A T A B A S E -------------------

CREATE DATABASE OfflineBooking
USE OfflineBooking

------------------- C R E A T I N G  T A B L E -------------------

-- CREATING TABLE FOR MOVIES
CREATE TABLE Movies(
id_movies INT PRIMARY KEY NOT NULL IDENTITY (1, 1),
title VARCHAR (30) NOT NULL,
duration VARCHAR (8) NOT NULL,
release_year INT NOT NULL,
age_rate CHAR (5) NOT NULL)

-- CREATING TABLE FOR STUDIO
CREATE TABLE Studio (
id_studio INT PRIMARY KEY NOT NULL IDENTITY (1, 1),
studio_number CHAR (5) UNIQUE NOT NULL)

-- CREATING TABLE FOR SCHEDULE
CREATE TABLE Schedule (
id_schedule INT PRIMARY KEY NOT NULL IDENTITY (1, 1),
start_at CHAR (5) NOT NULL,
dates DATE NOT NULL,
movies_id INT FOREIGN KEY (movies_id) REFERENCES Movies(id_movies) NOT NULL,
studio_id INT FOREIGN KEY (studio_id) REFERENCES Studio (id_studio) NOT NULL)

-- CREATING TABLE FOR SEAT
CREATE TABLE Seat (
id_seat INT PRIMARY KEY NOT NULL IDENTITY (1, 1),
seat_number INT NOT NULL,
studio_id INT FOREIGN KEY (studio_id) REFERENCES Studio(id_studio)NOT NULL)

-- CREATING TABLE FOR TICKET
CREATE TABLE Ticket (
id_ticket INT PRIMARY KEY NOT NULL IDENTITY (1, 1),
ticket_date DATE NOT NULL,
seat_id INT FOREIGN KEY (seat_id) REFERENCES Seat (id_seat) NOT NULL,
schedule_id INT FOREIGN KEY (schedule_id) REFERENCES Schedule(id_schedule) NOT NULL)

-- CREATING TABLE FOR BOOKING
CREATE TABLE Booking (
id_booking INT PRIMARY KEY IDENTITY (1, 1),
ticket_amount INT,
ticket_price MONEY,
total_price MONEY)
```

```
-- CREATING TABLE FOR ORDER DETAIL
CREATE TABLE OrderDetail (
booking_id INT FOREIGN KEY (booking_id) REFERENCES Booking (id_booking),
ticket_id INT FOREIGN KEY (ticket_id) REFERENCES Ticket (id_ticket))


-- CREATING TABLE FOR PAYMENT
CREATE TABLE Payment (
booking_id INT FOREIGN KEY (booking_id) REFERENCES Booking (id_booking),
payment_method VARCHAR (20),
payment_date DATE,
CONSTRAINT PayMthd CHECK ( payment_method IN ('CASH', 'E-WALLET', 'TRANSFER')))


------------------- I N S E R T I N G  D A T A -------------------

-- INSERTING DATA TO TABLE MOVIES
INSERT INTO Movies (title, release_year, duration, age_rate)
VALUES
    ('Inception', 2010, '2:28:00', 'PG'),
    ('Oppenheimer', 2023, '3:00:00', 'R'),
    ('The Dark Knight', 2008, '2:32:00', 'PG'),
    ('Interstellar', 2004, '2:49:00', 'PG'),
    ('Dunkirk', 1994, '1:46:00', 'G')


-- INSERTING DATA TO TABLE STUDIO
INSERT INTO Studio(studio_number)
VALUES
    ('ST1'),
       ('ST2'),
       ('ST3')


-- INSERTING DATA TO TABLE SCHEDULE
INSERT INTO Schedule(start_at, dates, movies_id, studio_id)
VALUES
    ('13:00', '12-29-2023', 1, 1),
       ('10:00', '12-29-2023', 2, 1),
       ('16:00', '12-30-2023', 3, 2),
       ('20:00', '12-30-2023', 4, 2),
       ('13:00', '12-31-2023', 5, 3)


-- INSERTING DATA INTO SEAT TABLE
INSERT INTO Seat (seat_number, studio_id)
VALUES
    (101, 1), -- Seat 101 in Studio 1
    (102, 1), -- Seat 102 in Studio 1
    (103, 1), -- Seat 103 in Studio 1
    (104, 1), -- Seat 104 in Studio 1
    (105, 1), -- Seat 105 in Studio 1
    (101, 2), -- Seat 106 in Studio 2
    (102, 2), -- Seat 107 in Studio 2
    (103, 2), -- Seat 108 in Studio 2
    (104, 2), -- Seat 109 in Studio 2
    (105, 2), -- Seat 110 in Studio 2
    (101, 3), -- Seat 101 in Studio 3
    (102, 3), -- Seat 102 in Studio 3
    (103, 3), -- Seat 103 in Studio 3
    (104, 3), -- Seat 104 in Studio 3
    (105, 3) -- Seat 105 in Studio 3


-- INSERTING DATA INTO TICKET TABLE
INSERT INTO Ticket (ticket_date, seat_id, schedule_id)
VALUES
       (GETDATE(), 1, 2),
       (GETDATE(), 2, 2),
       (GETDATE(), 6, 4),
       (GETDATE(), 7, 4),
       (GETDATE(), 14, 5),
       (GETDATE(), 15, 5),
       (GETDATE(), 9, 3),
       (GETDATE(), 10, 3)

------------------- P R O C E D U R E -------------------

-- PROCEDURE FOR INSERTING NEW MOVIES
CREATE PROCEDURE addNewMovies
    @title VARCHAR(50),
    @duration VARCHAR(50),
    @year INT,
    @age VARCHAR(50),
    @start VARCHAR(19),
    @date DATE,
       @movieID INT,
    @studnum INT
AS
BEGIN
    -- Insert into Movies table
    INSERT INTO Movies (title, duration, release_year, age_rate)
    VALUES (@title, @duration, @year, @age);

    -- Insert into Schedule table
    INSERT INTO Schedule (start_at, dates, movies_id, studio_id)
    VALUES (@start, @date, @movieId, @studnum);

       PRINT 'Movies data inserted successfully'
       PRINT 'Age Rate Note : (PARENTAL GUIDANCE), (GENERAL), (RESTRICTED)'

END


-- PROCEDURE ADD NEW TICKET
CREATE PROCEDURE newTicket
       @date DATE,
       @seat INT,
       @schedule INT
AS
BEGIN
       -- Insert into Ticket table
       INSERT INTO Ticket (ticket_date, seat_id, schedule_id)
       VALUES (@date, @seat, @schedule)
       PRINT 'New Ticket Successfully Added'
END
```

```
-- PROCEDURE NEW BOOKING
CREATE PROCEDURE newBooking
       @amount INT
AS
BEGIN
       INSERT INTO Booking (ticket_amount)
       VALUES (@amount)
       PRINT 'New Order Inserted'
END


-- PROCEDURE DETAIL ORDER
-- view ticket table fisrt before inserting new detail order
CREATE PROCEDURE newDetailOrder
       @booking INT,
       @ticket INT
AS
BEGIN
       INSERT INTO OrderDetail (booking_id, ticket_id)
       VALUES (@booking, @ticket)
END


-- PROCEDURE PAYMENT
CREATE PROCEDURE newPayment
       @bookid INT,
       @date DATE,
       @paymethod VARCHAR (20)
AS
BEGIN
       INSERT INTO Payment (booking_id, payment_date, payment_method)
       VALUES (@bookid, @date, @paymethod)
       PRINT 'Payment success'
END


------------------- T R I G G E R  -------------------

-- TRIGGER AUTOMATIC UPDATE TICKET PRICE AND CALCULATE IT IN BOOKING
CREATE TRIGGER bookingPrices
ON Booking
AFTER INSERT
AS
BEGIN
    -- Set fixed ticket_price
    UPDATE b
    SET ticket_price = 5.25
    FROM Booking b
    INNER JOIN inserted i ON b.id_booking = i.id_booking;

    -- Calculate total price based on ticket amount and ticket price
    UPDATE b
    SET total_price = i.ticket_amount * b.ticket_price
    FROM Booking b
    INNER JOIN inserted i ON b.id_booking = i.id_booking;
END;
```

```
------------------- V I E W   -------------------

-- VIEWING ALL TABLES
SELECT * FROM Movies
SELECT * FROM Schedule
SELECT * FROM Studio
SELECT * FROM Seat
SELECT * FROM  Ticket
SELECT * FROM Booking
SELECT * FROM OrderDetail
SELECT * FROM Payment
```

CONFIGURATION

Hardware :

Asus Vivobook M1603QA-AMD RYZEN 5000 Series,

Operating System :

Microsoft Windows 11 Home

Software:

Microsoft SQL Server 2019, Microsoft Office 2021