	NATIONAL UNIVERSITY OF TECHNOLOGY **Main I.J.P Road, Sector I-12 Islamabad**	

National University of Technology, Islamabad.

Computer Science Department

Semester Spring-2024
Course: DBS (CS-23-B)
Course Code: CS160

"DBS Project Report"

Submitted To:	**Submitted by:**
Mam Amna Ikram Date:03-06-2024	Sonia (F23605098) Ali Ahmed Khoso (F23605066) Arjia Ashraf (F23605088) Afnan Ahmed Gulzar (F23605087) Um-E-Aiman (F23605065)

Table of Contents

Introduction 1

Project Overview 1

Purpose 1

Scope 1

Software Requirements 2

System Design 3

Entity Relationship Diagram 3

Data Anomalies 4

Cinema Table 4

Seat Table 6

Final Schema 8

Entity Relationship Diagram 8

RELATIONSHIP, CARDINALITY AND CARDINALITY LIMITS BETWEEN ENTITIES 9

..... 10

Relational Diagram 13

SQL Queries 14

Front End 30

ONLINE MOVIE TICKET BOOKING SYSTEM

Introduction

Project Overview

The Movie Booking System is a database project designed to facilitate the booking of movie tickets through efficient digital platform. This system utilizes SQL for managing and storing data in a relational database. The data is stored in the form of tables, which include information about movies, screenings, users, bookings, and payments.

Purpose

The primary purpose of the Movie Booking System is to provide users with a convenient and user-friendly interface to browse movies, check show times, select seats, and book tickets online. For cinema administrators, it offers an efficient way to manage movie schedules, monitor bookings, and handle customer data. The system aims to automate and simplify the booking process, reduce manual errors, and enhance the overall customer experience.

Scope

The scope of the project includes the development of a comprehensive database that supports the following functionalities:

User Management:

- User registration and authentication.
- Profile management for users and administrators.

Movie Management:

- Adding, updating, and deleting movie information.
- Managing movie genres, durations, and descriptions.

Showtime and Screening Management:

- Scheduling movie screenings with dates and times.
- Managing different cinema halls and their seating capacities.

Booking System:

- Browsing available movies and show times.
- Generating booking confirmations.

Software Requirements

To develop and implement the Movie Ticket Booking System, the following software tools and platforms are used:

Database Design and Management Tool (Draw.io)

A diagramming tool used to create Entity-Relationship (ER) diagrams, which helps in visually representing the database structure. It is used for designing the database schema, defining tables, relationships, and constraints.

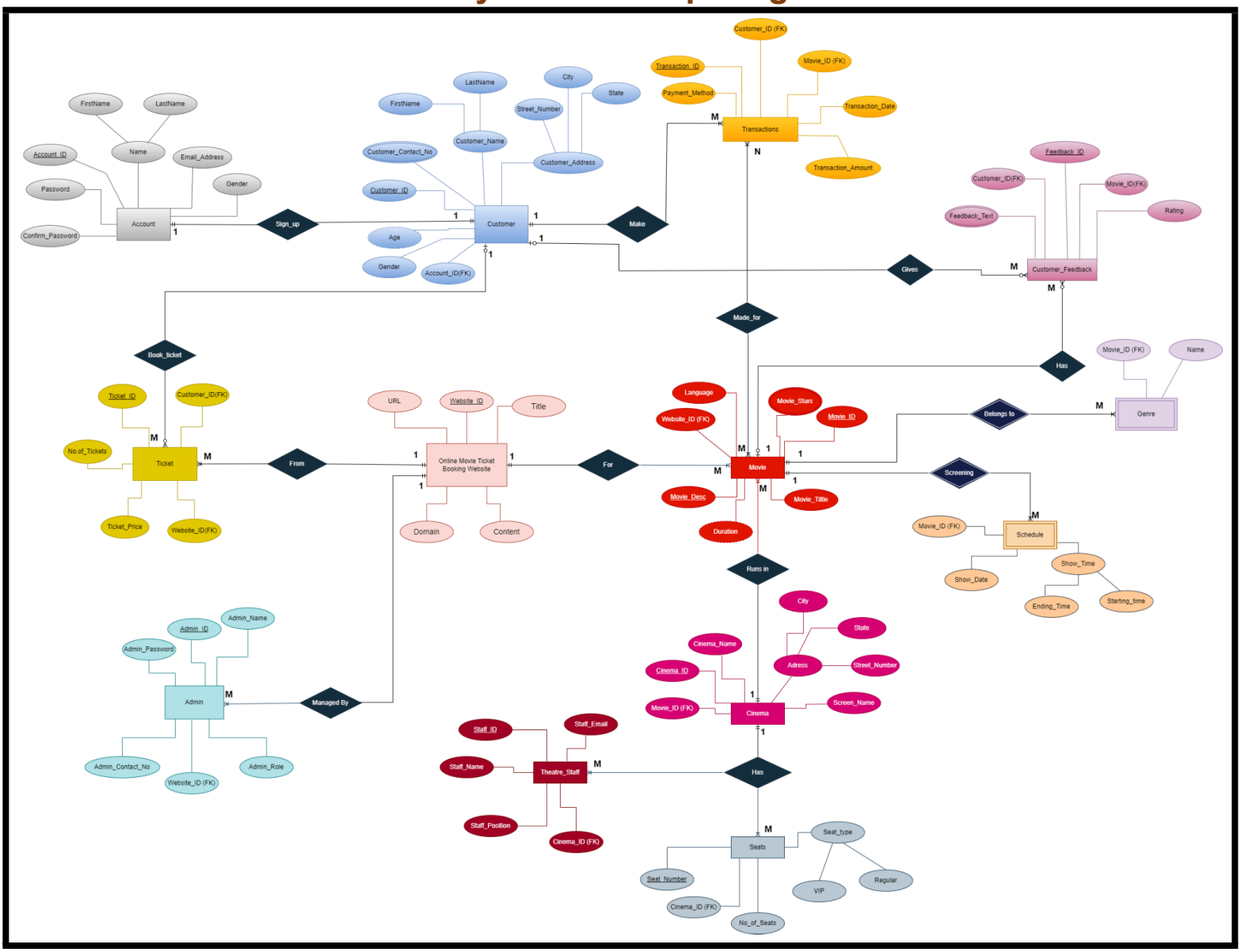
Database Management (XAMPP):

XAMPP is an open-source cross-platform web server solution package that includes:

- **Apache Server**: To run the server-side of the application.
- **MySQL**: For managing the relational database.
- **PHP**: To handle server-side scripting.
- **PhpMyAdmin**: A web-based interface to manage MySQL databases and tables easily.

System Design

Entity Relationship Diagram



Data Anomalies

Cinema Table

	Cinema_ID	Cinema_Name	City	State	Street_Number	Movie_ID
1	C1	Cinepax Lahore	Lahore	Punjab	street#4	M001
2	C2	Nueplex Cinema Karachi	Karachi	Sindh	DHA Phase 8	M002
3	C3	Cinegold Plex Islamabad	Islamabad	Islamabad Capital Territory	Bahria Enclave	M003
4	C4	Cine Star Cinema Multan	Multan	Punjab	Nishtar Road	M004
5	C5	Cinepax Rawalpindi	Rawalpindi	Punjab	1st Floor, Giga Mall	M005

Insertion Anomaly

If a new cinema is being added but no movie is currently scheduled, we cannot insert a new cinema without a valid Movie_ID it can lead to issues.

Update Anomaly

If a movie's details change, such as the `Movie_ID` in the `Movie` table we must update all instances of that `Movie_ID` in the `Cinemaa` table. This can lead to inconsistencies if one of the updates is missed.

Deletion Anomaly

If a movie is deleted from the `Movie` table, any cinema associated with that movie will have an invalid `Movie_ID`, breaking the foreign key constraint and potentially leading to data inconsistency.

Normalized Schema

To resolve these anomalies, we can normalize the table by splitting it into two tables: one for Cinema and another for the movie assignment.

Cinema Entity

	Cinema_ID	Cinema_Name	City	State	Street_Number
1	C1	Cinepax Lahore	Lahore	Punjab	street#4
2	C2	Nueplex Cinema Karachi	Karachi	Sindh	DHA Phase 8
3	C3	Cinegold Plex Islamabad	Islamabad	Islamabad Capital Territory	Bahria Enclave
4	C4	Cine Star Cinema Multan	Multan	Punjab	Nishtar Road
5	C5	Cinepax Rawalpindi	Rawalpindi	Punjab	1st Floor, Giga Mall

Cinema Assignment Entity

	Cinema_ID	Movie_ID
1	C1	M001
2	C2	M002
3	C3	M003
4	C4	M004
5	C5	M005

Seat Table

	Seat_Num	VIP	Regular	No_of_Seats	Cinema_ID
1	S1	VIP	Not regular	8	C1
2	S2	Not VIP	Regular	9	C2
3	S3	Not VIP	Regular	8	C3
4	S4	Not VIP	Regular	12	C4
5	S5	VIP	Not regular	5	C5
6	S6	Not VIP	Regular	3	C1
7	S7	VIP	Not regular	5	C2

Update Anomaly

The VIP and Regular columns represent redundant data. If the definition of VIP or Regular changes, each row containing these attributes must be updated, which could lead to inconsistencies.

Deletion Anomaly

If a row is deleted from the table, it could result in the loss of information about the cinema's seating arrangement. For example, if a cinema with ID 'C1' is closed, all its corresponding rows in the Seats table would be deleted, causing the loss of data about those seats even if they are used in other cinemas.

Insertion Anomaly

If we want to add information about a new cinema but don't currently have any seats allocated for it, we can't insert data into the Seats table because the primary key Seat_Num cannot be null.

Normalized Schema

To resolve these anomalies, we can normalize the table by splitting it into two tables: one for seat types and another for the seating arrangement in each cinema.

Seat Type Entity

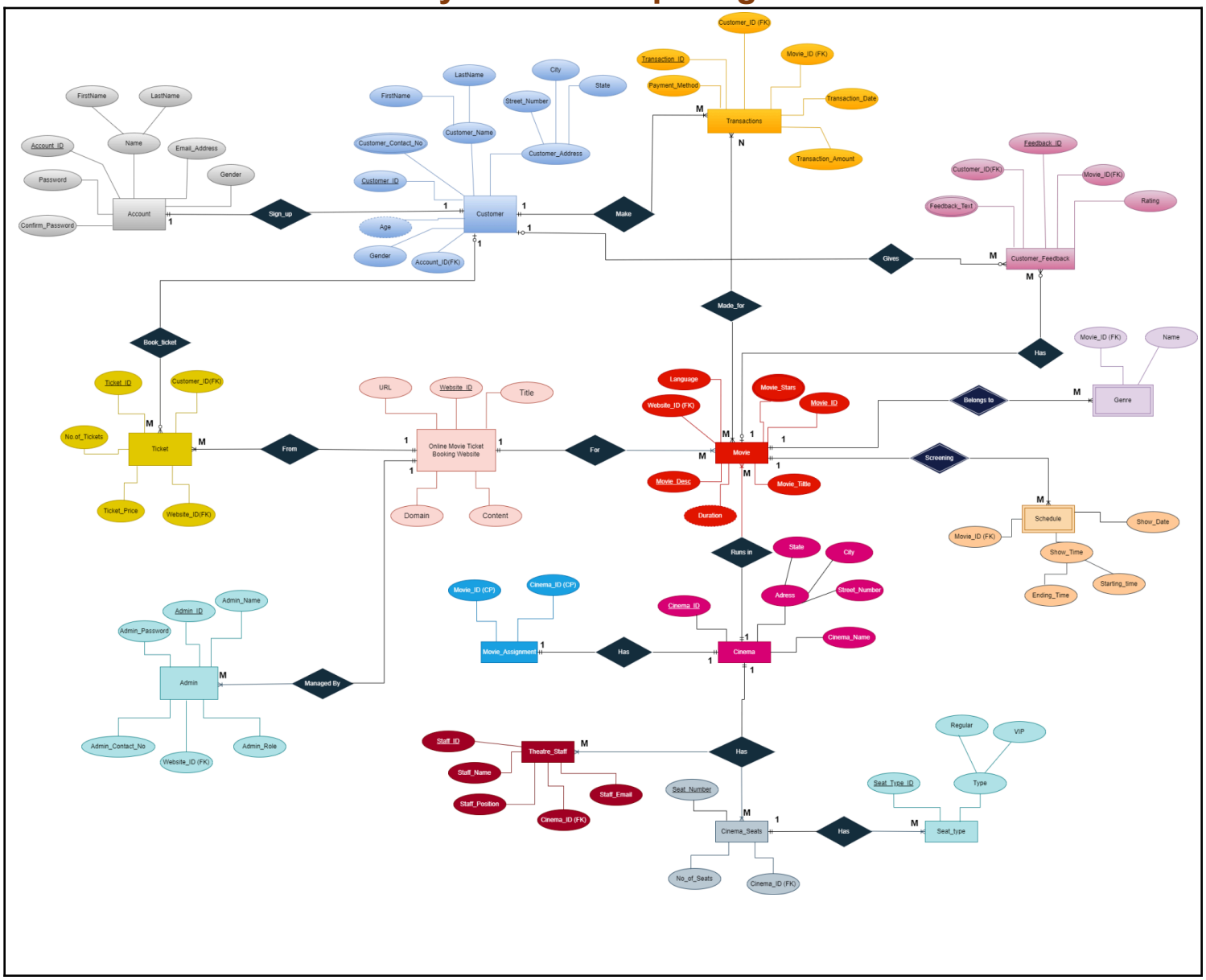
	Seat_Type_ID	Type
1	1	VIP
2	2	Regular

Cinema Seat Entity

	Seat_Num	No_of_Seats	Cinema_ID	Seat_Type_ID
1	S1	8	C1	1
2	S2	9	C2	2
3	S3	8	C3	2
4	S4	12	C4	2
5	S5	5	C5	1
6	S6	3	C1	2
7	S7	5	C2	1

Final Schema

Entity Relationship Diagram



RELATIONSHIP, CARDINALITY AND CARDINALITY LIMITS BETWEEN ENTITIES:

. Customer and Account

- **Relationship**: One-to-One
- **Cardinality**: One customer can have only one account.
- **Cardinality Limits**: 1 customer ↔ (0...1) account
- **Explanation**: Each customer can have one unique account, and each account belongs to one customer.

Customer and Transactions

- **Relationship**: One-to-Many
- **Cardinality**: One customer can have zero or many transactions.
- **Cardinality Limits**: 1 customer ↔ (0...N) transactions
- **Explanation**: A single customer can perform multiple transactions, but each transaction is performed by one customer.

Customer and Customer Feedback

- **Relationship**: One-to-Many
- **Cardinality**: One customer can provide zero or many feedbacks.
- **Cardinality Limits**: 1 customer ↔ (0...N) feedbacks
- **Explanation**: A single customer can give multiple feedbacks, but each feedback is given by one customer.

Transactions and Movie

- **Relationship**: Many-to-One
- **Cardinality**: Many transactions can be for one movie.
- **Cardinality Limits**: N transactions ↔ 1 movie
- **Explanation**: Multiple transactions can involve the same movie, but each transaction is for one specific movie.

Movie and Customer Feedback

- **Relationship**: One-to-Many
- **Cardinality**: One movie can have zero or many feedbacks.
- **Cardinality Limits**: 1 movie ↔ (0...N) feedbacks
- **Explanation**: A single movie can receive feedback from multiple customers, but each feedback is for one specific movie.

Cinema and Movie

- **Relationship**: Many-to-Many
- **Cardinality**: One cinema can show many movies, and one movie can be shown in many cinemas.
- **Cardinality Limits**: N cinemas ↔ M movies
- **Explanation**: Multiple cinemas can show the same movie, and each cinema can show multiple movies.

Admin and Online Movie Ticket Booking Website

- **Relationship**: Many-to-One
- **Cardinality**: One admin manages one online movie ticket booking website.
- **Cardinality Limits**: M admin ↔ 1…N website
- **Explanation**: Each website is managed by one or many admin.

Cinema and Theatre Staff

- **Relationship**: One-to-Many
- **Cardinality**: One cinema employs zero or many theatre staff.
- **Cardinality Limits**: 1 cinema ↔ (1...N) staff
- **Explanation**: A single cinema can have multiple theatre staff members, but each staff member works for one cinema.

Movie and Genre

- **Relationship**: Many-to-Many
- **Cardinality**: One movie can belong to many genres, and one genre can apply to many movies.
- **Cardinality Limits**: N movies ↔1… M genres
- **Explanation**: A movie can fall into multiple genres, and each genre can include multiple movies

Movie and Schedule

- **Relationship**: One-to-Many
- **Cardinality**: One movie can have zero or many schedules.
- **Cardinality Limits**: 1 movie ↔ (0...N) schedules
- **Explanation**: A single movie can have multiple schedules, but each schedule is for one specific movie.

Customer and Ticket

- **Relationship**: One-to-Many
- **Cardinality**: One customer can buy zero or many tickets.
- **Cardinality Limits**: 1 customer ↔ (0...N) tickets
- **Explanation**: A single customer can purchase multiple tickets, but each ticket is bought by one customer.

Ticket and Online Movie Ticket Booking Website

- **Relationship**: Many-to-One
- **Cardinality**: Many tickets can be booked through one online movie ticket booking website.
- **Cardinality Limits**: N tickets ↔ 1 website
- **Explanation**: Multiple tickets can be booked through the same website, but each ticket booking is made through one specific website.

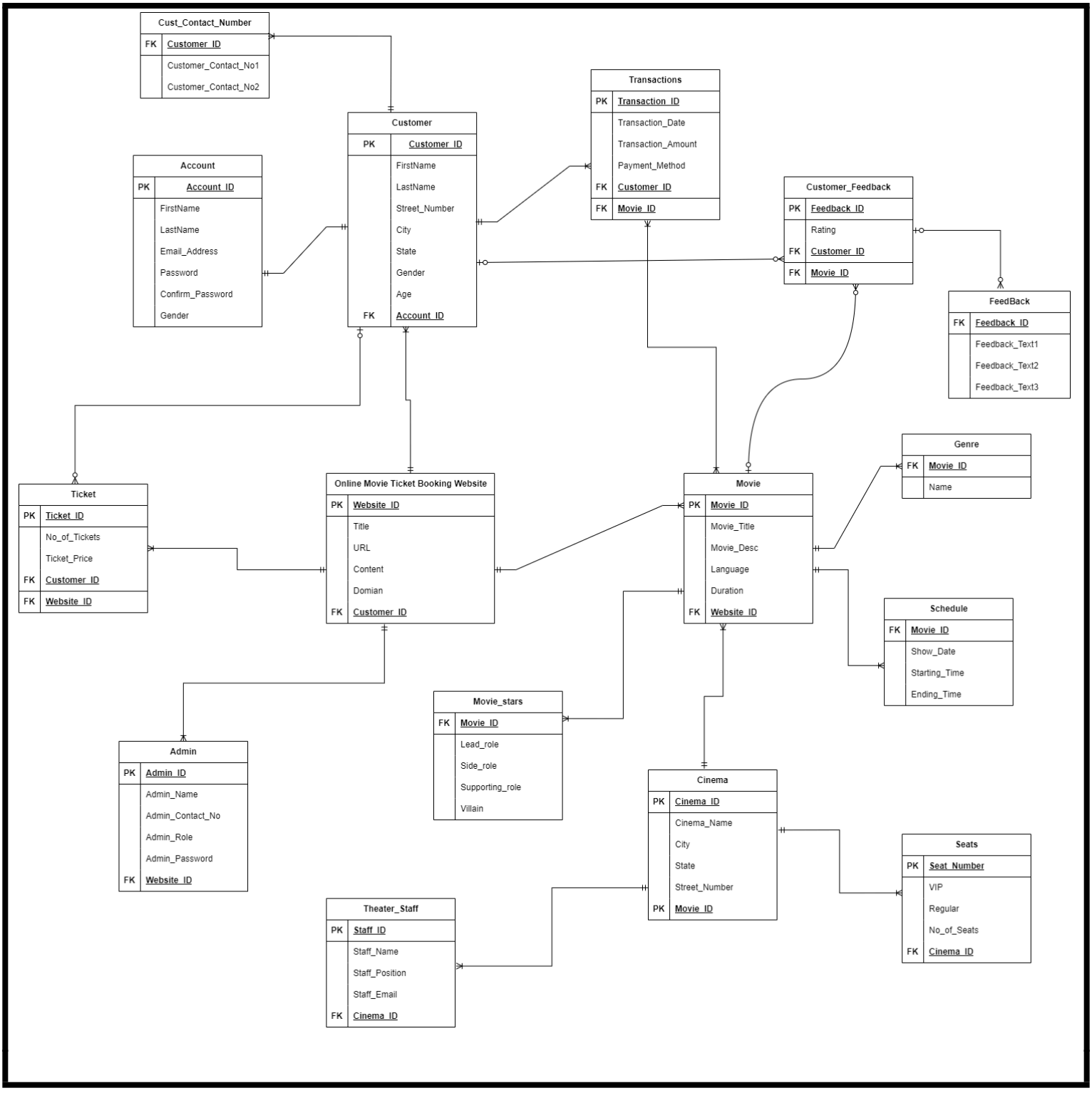
Movie and Online Movie Ticket Booking Website

- **Relationship**: Many-to-Many
- **Cardinality**: One movie can be listed on much online movie ticket booking websites, and one website can list many movies.
- **Cardinality Limits**: N movies ↔ M websites
- **Explanation**: A movie can be available on multiple ticket booking websites, and each website can list multiple movies.

Cinema and Seats

- **Relationship**: One-to-Many
- **Cardinality**: One cinema can have many seats.
- **Cardinality Limits**: 1 cinema ↔ (1...N) seats
- **Explanation**: A cinema contains multiple seats, but each seat belongs to one cinema.

Relational Diagram



SQL Queries

Query For Customer Table

```sql
USE Movie_Theatre_DBS_Project;
--Create Account table
CREATE TABLE Account
(
  Account_ID VARCHAR(20) PRIMARY KEY,
  FirstName VARCHAR(20) NOT NULL,
  LastName VARCHAR(20) ,
  Email_Address VARCHAR(50),
  Password VARCHAR(30) NOT NULL,
  Confirm_Password VARCHAR(30) NOT NULL,
  Gender VARCHAR(10),
);
-- Insert into account table
INSERT INTO Account VALUES
('A001','Danish','Khan','danish@gmail.com','F23','F23','Male'),
('A002', 'Fatima', 'Ahmed', 'fatima@gmail.com', 'password2', 'password2', 'Female'),
('A003', 'Muhammad', 'Hussain', 'hussain@gmail.com', 'password3', 'password3', 'Male'),
('A004', 'Ayesha', 'Malik', 'ayesha@gmail.com', 'password4', 'password4', 'Female'),
('A005', 'Ahmed', 'Khan', 'ahmed@gmail.com', 'password5', 'password5', 'Male'),
('A006', 'Sana', 'Ali', 'sana@gmail.com', 'password6', 'password6', 'Female'),
('A007', 'Hassan', 'Raza', 'hassan@gmail.com', 'password7', 'password7', 'Male'),
('A008', 'Zainab', 'Khan', 'zainab@gmail.com', 'password8', 'password8', 'Female'),
('A009', 'Bilal', 'Akhtar', 'bilal@gmail.com', 'password9', 'password9', 'Male'),
('A010', 'Nimra', 'Shah', 'nimra@gmail.com', 'password10', 'password10', 'Female'),
('A011','Saad','Raza','saad@gmail.com','password11','password11','Male'),
('A012','Fatima','Khan','fatima@gmail.com','password12','password12','Female'),
('A013','Sana','Javed','sana@gmail.com','password13','password13','Female');

--Display
```

100 %

Results | Messages

	Account_ID	FirstName	LastName	Email_Address	Password	Confirm_Password	Gender
1	A001	Danish	Khan	danish@gmail.com	F23	F23	Male
2	A002	Fatima	Ahmed	fatima@gmail.com	password2	password2	Female
3	A003	Muhammad	Hussain	hussain@gmail.com	password3	password3	Male
4	A004	Ayesha	Malik	ayesha@gmail.com	password4	password4	Female
5	A005	Ahmed	Khan	ahmed@gmail.com	password5	password5	Male

Query For Admin Table

```sql
USE Movie_Theatre_DBS_Project;

-- Create Admin table
CREATE TABLE Admin
(
   Admin_ID INT PRIMARY KEY,
   Admin_Name VARCHAR(50),
   Admin_Contact_No VARCHAR(15),
   Admin_Role VARCHAR(50),
   Admin_Pasword VARCHAR(30),
   Website_ID INT,
   FOREIGN KEY (Website_ID ) REFERENCES Online_Movie_Ticket_Booking_Website(Website_ID
);

--Insert into admin table
INSERT INTO Admin VALUES
('1', 'Ali Ahmed', '923001234567', 'Website Administrator', 'password123','1');

--Display
SELECT*FROM Admin;
```

Query For Cinema Table

```sql
USE NORMALISATION;
--Split cinema table into two tables

--Create normalise cinema table
CREATE TABLE Cinema_Normalised
(
  Cinema_ID VARCHAR(15) PRIMARY KEY,
  Cinema_Name VARCHAR(100),
  City VARCHAR(30),
  State VARCHAR(30),
  Street_Number VARCHAR(30)
);
--Insert values
INSERT INTO Cinema_Normalised VALUES
('C1', 'Cinepax Lahore', 'Lahore', 'Punjab', 'street#4'),
('C2', 'Nueplex Cinema Karachi', 'Karachi', 'Sindh', 'DHA Phase 8'),
('C3', 'Cinegold Plex Islamabad', 'Islamabad', 'Islamabad Capital Territory', 'Bahria Enclave'),
('C4', 'Cine Star Cinema Multan', 'Multan', 'Punjab', 'Nishtar Road'),
('C5', 'Cinepax Rawalpindi', 'Rawalpindi', 'Punjab', '1st Floor, Giga Mall');
SELECT*FROM Cinema_Normalised;

--create normalise movie table
CREATE TABLE Movie_Normalised
(
  Cinema_ID VARCHAR(15),
  Movie_ID VARCHAR(20),
  PRIMARY KEY (Cinema_ID, Movie_ID),
  FOREIGN KEY (Cinema_ID) REFERENCES Cinema_Normalised(Cinema_ID),
  FOREIGN KEY (Movie_ID) REFERENCES Movie(Movie_ID)
);
--Insert values
INSERT INTO Movie_Normalised VALUES
('C1','M001'),
('C2','M002'),
('C3','M003'),
('C4','M004'),
('C5','M005');
```

Query For Customer Contact Table

```
USE Movie_Theatre_DBS_Project;

-- Create Customer_Contact_No table
CREATE TABLE Cust_Contact_Numberr
(
  Customer_ID VARCHAR(10),
  FOREIGN KEY(Customer_ID) REFERENCES Customer(Customer_ID),
  Customer_Contact_No1 VARCHAR(30),
  Customer_Contact_No2 VARCHAR(30),
);

--Insert into customer table
INSERT INTO Cust_Contact_Numberr VALUES
('C001', '923001234567','6762821881'),
('C002', '923451234567','0918625378'),
('C003', '923001112233','08166521811'),
('C004', '923001234568','03324578901'),
('C005', '923003456789','8289189819'),
('C006','097432482910','03341672829'),
('C007','033245699191','03316161829'),
('C008','033241516919','04217189291'),
('C009','07625149291','08178288171'),
('C010','033224561891','05561781921');

--Display
```

100 %

Results | Messages

	Customer_ID	Customer_Contact_No1	Customer_Contact_No2
1	C001	923001234567	6762821881
2	C002	923451234567	0918625378
3	C003	923001112233	08166521811
4	C004	923001234568	03324578901
5	C005	923003456789	8289189819
6	C006	097432482910	03341672829
7	C007	033245699191	03316161829
8	C008	033241516919	04217189291
9	C009	07625149291	08178288171
10	C010	033224561891	05561781921

Query For Customer Table

```sql
USE Movie_Theatre_DBS_Project;

-- Create Customer table
CREATE TABLE Customer (
    Customer_ID VARCHAR(10) PRIMARY KEY,
    FirstName VARCHAR(100),
    LastName VARCHAR(100),
    Street_Number VARCHAR(30),
    City VARCHAR(30),
    State VARCHAR(30),
    Gender VARCHAR(30),
    Age INT,
    Account_ID VARCHAR(20) ,
    FOREIGN KEY (Account_ID) REFERENCES Account(Account_ID),
);
-- Insert into cutomer table
INSERT INTO Customer VALUES
('C001', 'Danish', 'Khan', '123', 'Karachi', 'Sindh', 'Male', 30,'A001'),
('C002', 'Fatima', 'Ahmed', '456', 'Lahore', 'Punjab', 'Female', 25, 'A002'),
('C003', 'Muhammad', 'Hussain', '789', 'Islamabad', 'Islamabad Capital Territory', 'Male', 35, 'A003'),
('C004', 'Ayesha', 'Malik', '1011', 'Rawalpindi', 'Punjab', 'Female', 28, 'A004'),
('C005', 'Ahmed', 'Khan', '1213', 'Faisalabad', 'Punjab', 'Male', 40, 'A005'),
('C006', 'Sana', 'Ali', '1210', 'Lahore', 'Punjab', 'Male', 40, 'A006'),
('C007','Hassan', 'Raza','15','Rawalpindi', 'Punjab','Male',27,'A007'),
('C008','Zainab', 'Khan','34','Karachi','Sindh','Female',22,'A008'),
('C009','Bilal', 'Akhtar','10','Lahore','Punjab','Male',30,'A009'),
('C010','Nimra', 'Shah','17','Rawalpindi','Punjab','Female',25,'A010');

--Display
SELECT*FROM Customer;
```

100 %

Results | Messages

	Customer_ID	FirstName	LastName	Street_Number	City	State	Gender	Age	Account_ID
1	C001	Danish	Khan	123	Karachi	Sindh	Male	30	A001
2	C002	Fatima	Ahmed	456	Lahore	Punjab	Female	25	A002
3	C003	Muhammad	Hussain	789	Islamabad	Islamabad Capital Territory	Male	35	A003
4	C004	Ayesha	Malik	1011	Rawalpindi	Punjab	Female	28	A004
5	C005	Ahmed	Khan	1213	Faisalabad	Punjab	Male	40	A005

Query For Customer Feedback Table

```sql
USE Movie_Theatre_DBS_Project;

-- Create customer feedback table
CREATE TABLE Customer_Feedback (
    Feedback_ID INT PRIMARY KEY,
    Rating INT,
    Customer_ID VARCHAR(10),
    FOREIGN KEY (Customer_ID) REFERENCES Customer(Customer_ID),
    Movie_ID VARCHAR(20),
    FOREIGN KEY (Movie_ID) REFERENCES Movie(Movie_ID)
);

--Insert values
INSERT INTO Customer_Feedback VALUES
(1, 4,'C001','M001'),
(2, 5,'C002','M002'),
(3, 1,'C003','M003'),
(4, 4,'C004','M004'),
(5, 3,'C005','M005');

--Display
SELECT*FROM Customer_Feedback;
```

100 %

Results | Messages

	Feedback_ID	Rating	Customer_ID	Movie_ID
1	1	4	C001	M001
2	2	5	C002	M002
3	3	1	C003	M003
4	4	4	C004	M004
5	5	3	C005	M005

Query For Customer Feedback Text Table

```
USE Movie_Theatre_DBS_Project;

-- Create customer feedback text table(multivalued)
CREATE TABLE Feedback_text
(
  Feedback_ID INT,
  FOREIGN KEY(Feedback_ID ) REFERENCES Customer_Feedback(Feedback_ID ),
  Feedback_Text1 VARCHAR(70),
  Feedback_Text2 VARCHAR(70),
  Feedback_Text3 VARCHAR(70),
);
--Insert values
INSERT INTO Feedback_text VALUES
(1, 'Great experience! Loved the movie selection.', 'The ticket booking process was smooth', 'The theater ambiance was fantastic.'),
(2, 'Movie was okay, could have been better.', 'The snack options could be improved.', 'Overall, a good experience.'),
(3, 'Enjoyed the movie night with friends.', 'The staff was friendly and helpful.', 'Looking forward to visiting again.'),
(4, 'The seating was comfortable.', 'The sound system needs an upgrade.', 'Overall, a satisfactory experience.'),
(5, 'Great movie! Really enjoyed it.','The cinema experience was bad','overall, ok');

--Display
SELECT*FROM Feedback_text;
```

100 %

Results | Messages

	Feedback_ID	Feedback_Text1	Feedback_Text2	Feedback_Text3
1	1	Great experience! Loved the movie selection.	The ticket booking process was smooth	The theater ambiance was fantastic.
2	2	Movie was okay, could have been better.	The snack options could be improved.	Overall, a good experience.
3	3	Enjoyed the movie night with friends.	The staff was friendly and helpful.	Looking forward to visiting again.
4	4	The seating was comfortable.	The sound system needs an upgrade.	Overall, a satisfactory experience.
5	5	Great movie! Really enjoyed it.	The cinema experience was bad	overall, ok

Query For Customer Feedback Text Table

```
USE Movie_Theatre_DBS_Project;

-- Create Genre table
CREATE TABLE Genre
(
  Name VARCHAR(30),
  Movie_ID VARCHAR(20),
  FOREIGN KEY( Movie_ID) REFERENCES Movie( Movie_ID)
);

--Insert values
INSERT INTO Genre VALUES
('Action','M001'),
('Comedy','M002'),
('Mystery','M003'),
('Action','M004'),
('Fantasy','M005');

--Display
SELECT*FROM Genre;
```

100 %

Results | Messages

	Name	Movie_ID
1	Action	M001
2	Comedy	M002
3	Mystery	M003
4	Action	M004
5	Fantasy	M005

Query For Movie Stars Table

```sql
USE Movie_Theatre_DBS_Project;

-- Create Movie_Stars table (multivlued)
CREATE TABLE Movie_Stars
(
  Movie_ID VARCHAR(20),
  FOREIGN KEY(Movie_ID) REFERENCES Movie(Movie_ID),
  Lead_role VARCHAR(25),
  Side_role VARCHAR(20),
  Supporting_role VARCHAR(30),
  Villain VARCHAR(25),
);

INSERT INTO Movie_Stars VALUES
('M001', 'Leonardo DiCaprio', 'Joseph Gordon-Levitt', 'Ellen Page', 'Cillian Murphy'),
('M002', 'Christian Bale', 'Heath Ledger', 'Aaron Eckhart', 'Tom Hardy'),
('M003', 'Bruce Willis','NULL','NULL','Alan Rickman'),
('M004', 'Song Kang-ho','NULL','Cho Yeo-jeong', 'Lee Sun-kyun'),
('M005', 'Aamir Khan','NULL', 'Sakshi Tanwar','NULL');

--Display
SELECT*FROM Movie_Stars;
```

100 %

Results | Messages

	Movie_ID	Lead_role	Side_role	Supporting_role	Villain
1	M001	Leonardo DiCaprio	Joseph Gordon-Levitt	Ellen Page	Cillian Murphy
2	M002	Christian Bale	Heath Ledger	Aaron Eckhart	Tom Hardy
3	M003	Bruce Willis	NULL	NULL	Alan Rickman
4	M004	Song Kang-ho	NULL	Cho Yeo-jeong	Lee Sun-kyun
5	M005	Aamir Khan	NULL	Sakshi Tanwar	NULL

Query For Movie Table

```
USE Movie_Theatre_DBS_Project;

-- Create Movie table
CREATE TABLE Movie
(
  Movie_ID VARCHAR(20) PRIMARY KEY,
  Move_Title VARCHAR(50),
  Duration VARCHAR(40),
  Movie_Desc VARCHAR(20),
  Language VARCHAR(30),
  Website_ID INT,
  FOREIGN KEY(Website_ID) REFERENCES Online_Movie_Ticket_Booking_Website(Website_ID),
);

--insert into movie table
INSERT INTO Movie  VALUES
('M001', 'Inception', '2h 28m', 'thriller', 'English', 1),
('M002', 'The Dark Knight','3h 15m', 'Batman Begins', 'Spanish',1),
('M003', 'Die Hard','3h 4m','Action-packed ','Turkey',1),
('M004', 'Parasite', '2h 12m', 'dark comedy', 'Korean', 1),
('M005', 'Dangal', '2h 41m', 'sports biographical', 'Hindi',1);

--Display
SELECT*FROM Movie ;
```

100 %

Results | Messages

	Movie_ID	Move_Title	Duration	Movie_Desc	Language	Website_ID
1	M001	Inception	2h 28m	thriller	English	1
2	M002	The Dark Knight	3h 15m	Batman Begins	Spanish	1
3	M003	Die Hard	3h 4m	Action-packed	Turkey	1
4	M004	Parasite	2h 12m	dark comedy	Korean	1
5	M005	Dangal	2h 41m	sports biographical	Hindi	1

Query For Schedule Table

```
USE Movie_Theatre_DBS_Project;

-- Create Schedule table
CREATE TABLE Schedule
(
   Show_Date Date,
   Starting_Time VARCHAR(20),
   Ending_Time VARCHAR(20),
    Movie_ID VARCHAR(20),
   FOREIGN KEY( Movie_ID) REFERENCES Movie( Movie_ID)
);

--Insert values
INSERT INTO Schedule VALUES
('2024-05-20', '13:00', '15:30', 'M001'),
('2024-05-20', '16:00', '18:30', 'M002'),
('2024-05-21', '14:00', '16:30', 'M003'),
('2024-05-21', '18:00', '20:30', 'M004'),
('2024-05-22', '12:00', '14:30', 'M005');

--Display
SELECT*FROM Schedule;
```

100 %

Results | Messages

	Show_Date	Starting_Time	Ending_Time	Movie_ID
1	2024-05-20	13:00	15:30	M001
2	2024-05-20	16:00	18:30	M002
3	2024-05-21	14:00	16:30	M003
4	2024-05-21	18:00	20:30	M004
5	2024-05-22	12:00	14:30	M005

Query For Theater staff Table

```
Genre table.sql - D...P-SL5V2VI\dell (64))    schedule table.sql...P-SL5V2VI\dell (71))    cinema table.sql -...P

USE Movie_Theatre_DBS_Project;

-- Create Theater_Staff table
CREATE TABLE Theatre_Staff
(
   Staff_ID VARCHAR(15) PRIMARY KEY,
   Staff_Name VARCHAR(100),
   Staff_Position VARCHAR(30),
   Staff_Email VARCHAR(50),
   Cinema_ID VARCHAR(15),
   FOREIGN KEY (Cinema_ID) REFERENCES Cinemaa(Cinema_ID)
);

--Insert values
INSERT INTO Theatre_Staff VALUES
('F23605066', 'Ali Ahmed', 'Manager', 'ali@example.com','C1'),
('F23605088', 'Sonia', 'Ticketing Staff', 'sonia@example.com','C2'),
('F23605098', 'Arjia Ashraf', 'Ticketing Staff2', 'arjia@example.com','C3'),
('F23605065', 'Ume Aimen', 'Concession Staff', 'aimen@example.com','C4'),
('F23605087', 'Afnan Ahmed', 'Projectionist', 'afnan@example.com','C5');

--Display
SELECT*FROM Theatre_Staff ;
```

100 %

Results | Messages

	Staff_ID	Staff_Name	Staff_Position	Staff_Email	Cinema_ID
1	F23605065	Ume Aimen	Concession Staff	aimen@example.com	C4
2	F23605066	Ali Ahmed	Manager	ali@example.com	C1
3	F23605087	Afnan Ahmed	Projectionist	afnan@example.com	C5
4	F23605088	Sonia	Ticketing Staff	sonia@example.com	C2
5	F23605098	Arjia Ashraf	Ticketing Staff2	arjia@example.com	C3

Query For Tickets Table

```sql
USE Movie_Theatre_DBS_Project;

CREATE TABLE Tickets (
    TicketID INT PRIMARY KEY,
    NoOfTickets INT ,
    TicketPrice DECIMAL(10, 2) NOT NULL,
    Customer_ID VARCHAR(10),
    Website_ID INT,
    FOREIGN KEY (Customer_ID) REFERENCES customer( Customer_ID),
    FOREIGN KEY (Website_ID) REFERENCES Online_Movie_Ticket_Booking_Website(Website_ID)
);

INSERT INTO Tickets VALUES
(1, 2, 250.99, 'C001', 1),
(2, 4, 150.50,'C002', 1),
(3,5,198.5,'C003',1),
(4,4,145.7,'C004',1),
(5,8,345.7,'C005',1);

--Display
SELECT*FROM Tickets;
```

100 %

Results | Messages

	TicketID	NoOfTickets	TicketPrice	Customer_ID	Website_ID
1	1	2	250.99	C001	1
2	2	4	150.50	C002	1
3	3	5	198.50	C003	1
4	4	4	145.70	C004	1
5	5	8	345.70	C005	1

Query For Tickets Table

```sql
USE Movie_Theatre_DBS_Project;

-- Create transaction table
CREATE TABLE Transactions(
    Transaction_ID INT PRIMARY KEY,
    Transaction_Date DATE,
    Transaction_Amount DECIMAL(10,2),
    Payment_Method VARCHAR(30),
    Customer_ID VARCHAR(10),
    FOREIGN KEY (Customer_ID) REFERENCES Customer(Customer_ID),
    Movie_ID VARCHAR(20),
    FOREIGN KEY (Movie_ID) REFERENCES Movie(Movie_ID)
);

--Insert values
INSERT INTO Transactions  VALUES
(1, '2024-05-20', 2550, 'Credit Card','C001','M001'),
(2, '2024-05-20', 3000, 'Debit Card','C002','M002'),
(3, '2024-05-21', 2068, 'Cash','C003','M003'),
(4, '2024-05-21', 2520, 'Credit Card','C004','M004'),
(5, '2024-05-22', 1800, 'Debit Card','C005','M005');

SELECT*FROM Transactions;
```

100 %

Results | Messages

	Transaction_ID	Transaction_Date	Transaction_Amount	Payment_Method	Customer_ID	Movie_ID
1	2	2024-05-20	3000.00	Debit Card	C002	M002
2	3	2024-05-21	2068.00	Cash	C003	M003
3	4	2024-05-21	2520.00	Credit Card	C004	M004
4	5	2024-05-22	1800.00	Debit Card	C005	M005
5	91	2024-05-20	2550.00	Credit Card	C001	M001

Query For Website Table

```sql
USE Movie_Theatre_DBS_Project;

-- Create Website table
CREATE TABLE Online_Movie_Ticket_Booking_Website (
    Website_ID INT PRIMARY KEY,
    Title VARCHAR(100),
    Domian VARCHAR(15),
    URL VARCHAR(25),
    Content VARCHAR(30),
    Customer_ID VARCHAR(10),
    FOREIGN KEY (Customer_ID) REFERENCES Customer(Customer_ID)
);

--insert into website table
INSERT INTO Online_Movie_Ticket_Booking_Website VALUES
('1', 'MovieBooker', 'moviebooker.com', 'www.moviebooker.com', 'Movie ticket booking platform', 'C001');

--Display
SELECT*FROM Online_Movie_Ticket_Booking_Website;
```

100 %

Results | Messages

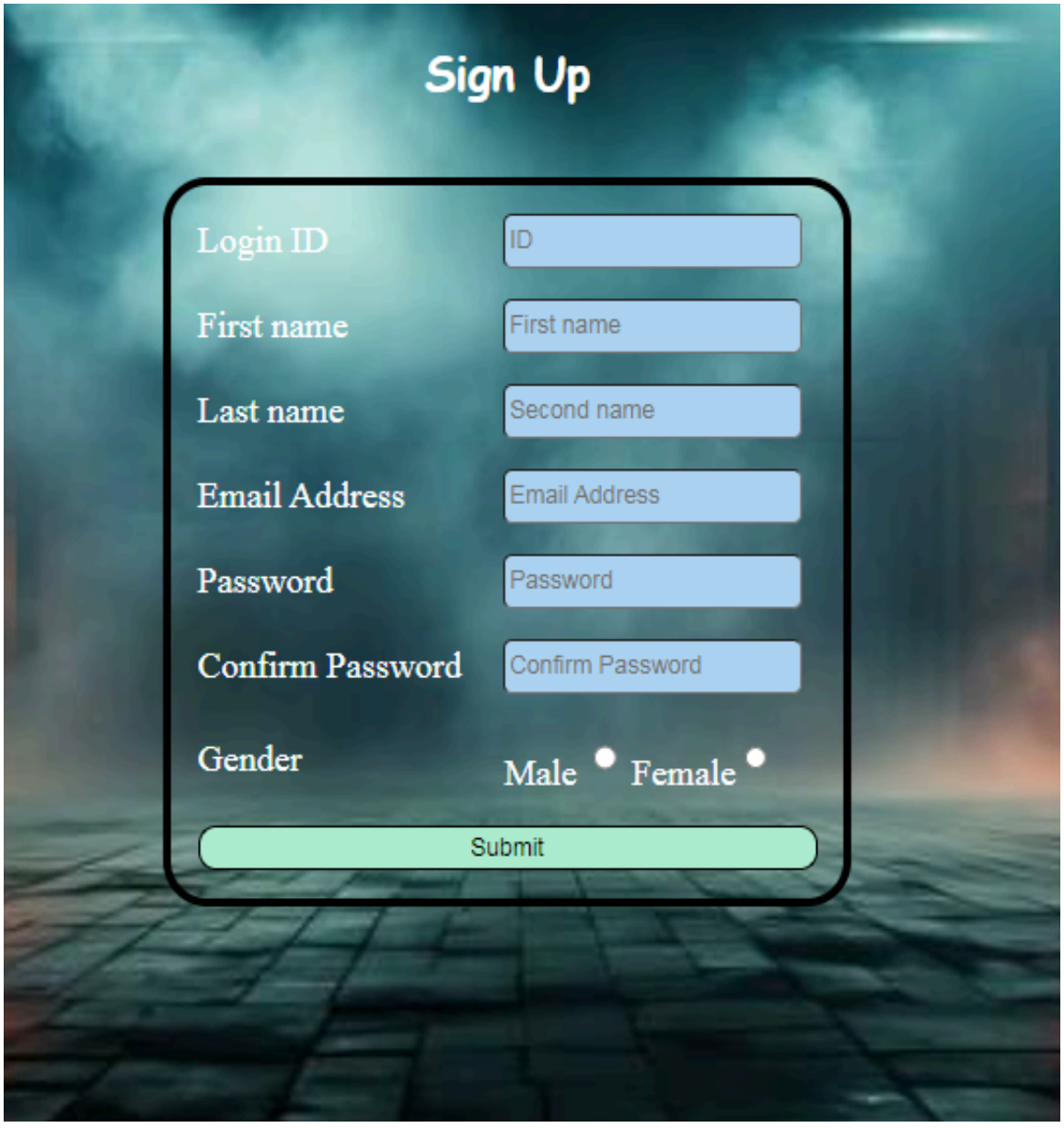
	Website_ID	Title	Domian	URL	Content	Customer_ID
1	1	MovieBooker	moviebooker.com	www.moviebooker.com	Movie ticket booking platform	C001

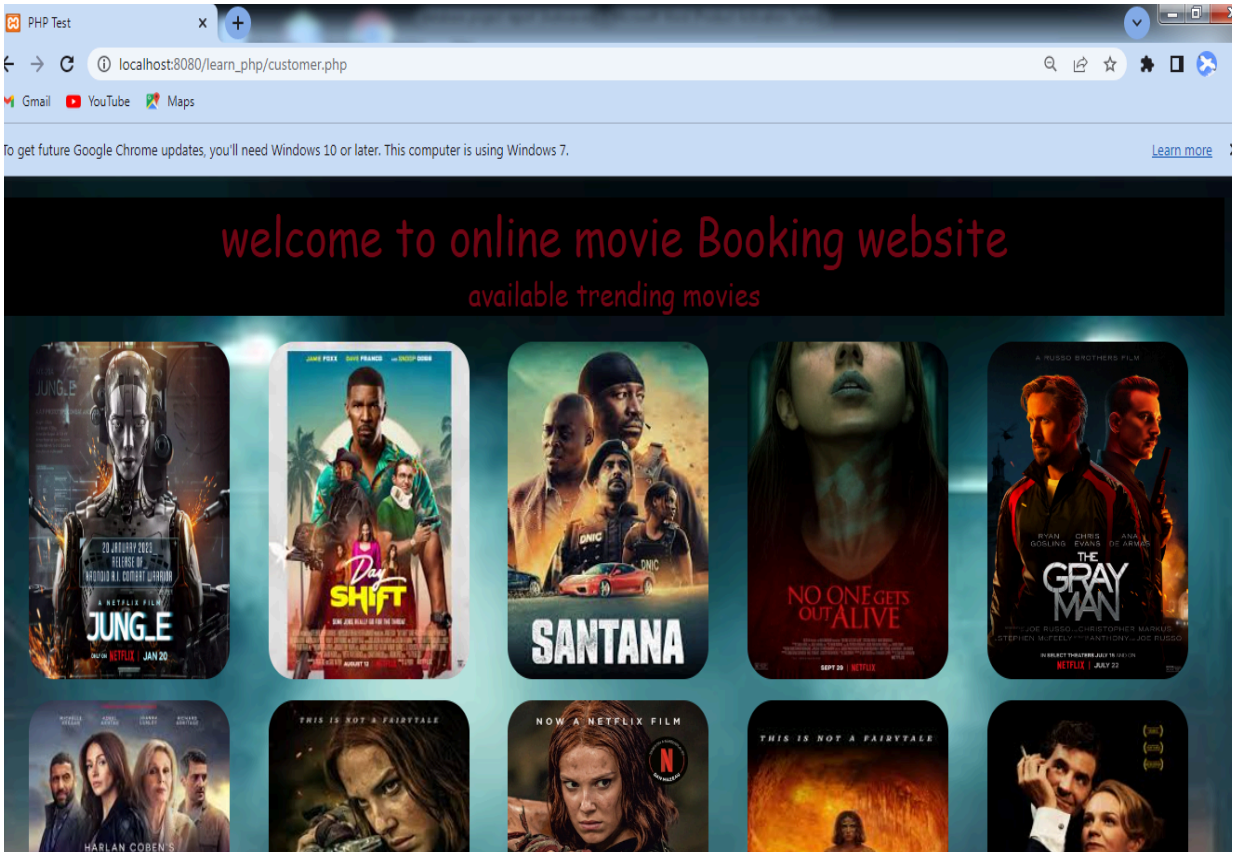
Query For Cinema Seat and Seat Type Table

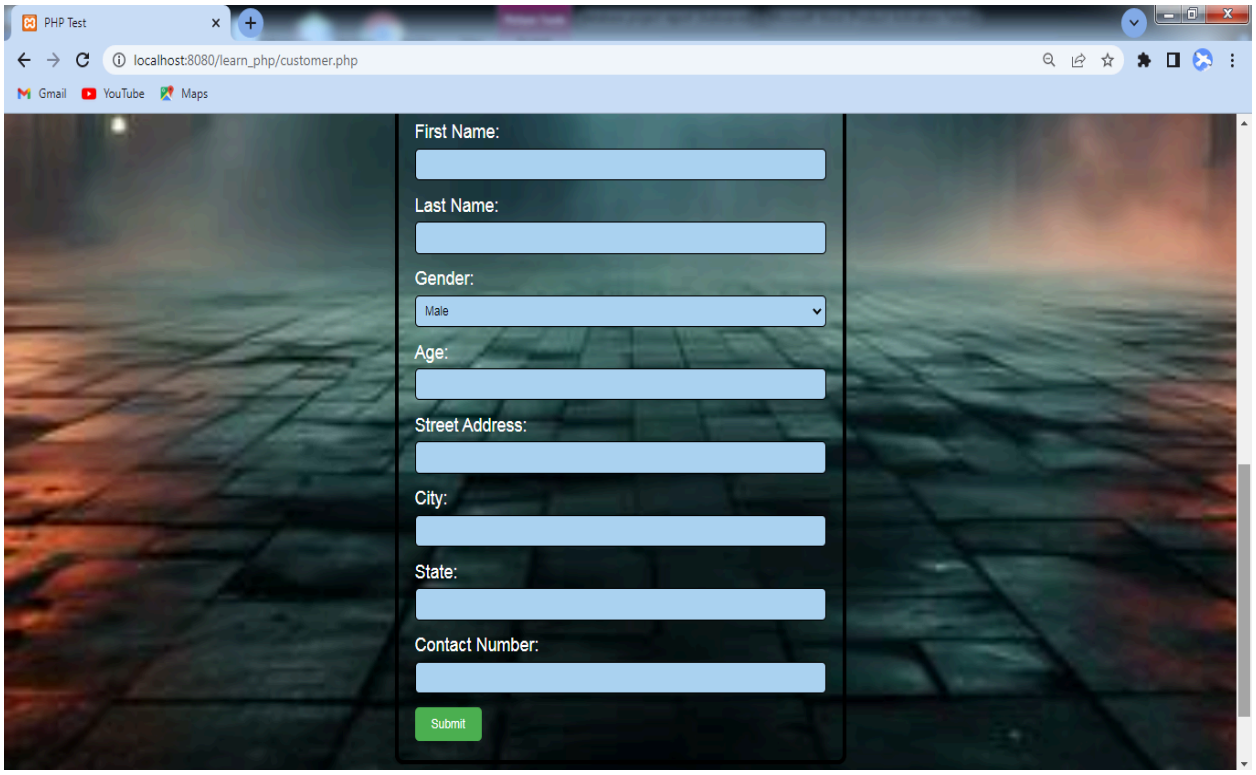
```sql
USE NORMALISATION;
--Split seat table into two table

--Create seat type
CREATE TABLE Seat_Types (
    Seat_Type_ID INT PRIMARY KEY,
    Type VARCHAR(20) NOT NULL -- 'VIP' or 'Regular'
);
--Insert values
INSERT INTO Seat_Types (Seat_Type_ID, Type) VALUES
(1, 'VIP'),
(2, 'Regular');

--create Cinema seat
CREATE TABLE Cinema_Seats (
    Seat_Num VARCHAR(10) PRIMARY KEY,
    No_of_Seats INT,
    Cinema_ID VARCHAR(15),
    Seat_Type_ID INT,
    FOREIGN KEY (Cinema_ID) REFERENCES Cinema_Normalised(Cinema_ID),
    FOREIGN KEY (Seat_Type_ID) REFERENCES Seat_Types(Seat_Type_ID)
);
--Insert values
INSERT INTO Cinema_Seats (Seat_Num, No_of_Seats, Cinema_ID, Seat_Type_ID) VALUES
('S1', 8, 'C1', 1),
('S2', 9, 'C2', 2),
('S3', 8, 'C3', 2),
('S4', 12, 'C4', 2),
('S5', 5, 'C5', 1),
('S6', 3, 'C1', 2),
('S7', 5, 'C2', 1);
```

Front End

Online Movie Ticket Booking Website




PHP Test
localhost:8080/learn_php/customer.php
Gmail
YouTube
Maps
To get future Google Chrome updates, you'll need Windows 10 or later. This computer is using Windows 7.
Learn more
welcome to online movie Booking website
available trending movies
JUNG_E
Day SHIFT
SANTANA
NO ONE GETS OUT ALIVE
THE GRAY MAN
NOW A NETFLIX FILM
THIS IS NOT A FAIRYTALE


PHP Test
localhost:8080/learn_php/customer.php
Gmail
YouTube
Maps
First Name:
Last Name:
Gender:
Male
Age:
Street Address:
City:
State:
Contact Number:
Submit