

**School of Technology & Management Engineering, Navi Mumbai**

**Department of Computer Science**

**Kharghar, Navi Mumbai- 410210**

A Report on

**Movie Ticket Booking System**



**Course:** DBMS

Submitted By: **Prapti Gupta**

**Tanisha Shaha**

Roll Nos: **A-180, A-175**

**SVKM’s NMIMS**

**School of Technology Management & Engineering, Navi Mumbai**

**A.Y. 2023 – 24**

**Course:** Database Management Systems

**Project Report**

|  |  |  |
| --- | --- | --- |
| **Program:** | B-Tech CE | |
| **Semester:** | IV | |
| **Name of the Project:** | Movie Ticket Booking System | |
|  | | |
| **Details of Project Members:** |  |  |
| **Batch** | **Roll No.** | **Name** |
| B1 | A175 | Tanisha Shaha |
| B1 | A180 | Prapti Gupta |
|  |  |  |
| **Date of Submission:** 27/03/2024 | | |

**Contribution of each project Members:**

|  |  |  |
| --- | --- | --- |
| **Roll No.** | **Name:** | **Contribution** |
| A175 | Tanisha Shaha | Equal |
| A180 | Prapti Gupta | Equal |

**GitHub Link:** <https://github.com/Prapti-gupta/movie-ticket-booking-db-queries>

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Sr no.** | **Topic** | **Page no.** |
| **1** | Storyline | 4 |
| **2** | Components of Database Design | 4-6 |
| **3** | Entity Relationship Diagram | 7 |
| **4** | Relational Model | 7 |
| **5** | Normalization | 8 |
| **6** | SQL Queries | 9-25 |
| **8** | Project Demonstration | 26 |
| **9** | Self-learning beyond classroom | 26 |
| **10** | Learning from the project | 26 |
| **11** | Challenges faced | 27 |
| **12** | Conclusion | 27 |

**I. Storyline**

In a busy city, Max and his friends wanted to create a movie platform where booking tickets would be easy for everyone. So, they made Cinemania, a friendly place where movie lovers can easily book their favorite shows. Cinemania needed a solid foundation - a database to manage everything from user profiles to showtimes and payments. This includes storing user profiles, managing movie details like titles, genres, and showtimes, handling ticket bookings and reservations, processing payments securely, and providing administrative controls. This database would allow users to browse movies, choose showtimes, and book seats effortlessly, all while keeping their information safe and secure. With everything organized, Cinemania assured to make movies easy and reliable for everyone's enjoyment.

**II. Components of Database Design**

There are a total 10 entities in this database, listed as follows:

* **Users**:

**Attributes**: LoginID (Primary Key), Name, Age, Gender, Phone Number, Email, Ticket No. (Foreign Key)

**Cardinality**: One user can make many bookings. (one to many)

* **Bookings**:

**Attributes**: BookingID (Primary Key), Booking Date, Total Price, Number of People, Login ID (Foreign Key)

* **Movie**:

**Attributes**: Movie ID (Primary Key), Movie Name, Genre, Language, Movie rating

**Cardinality**: One movie can have many shows. (one to many)

* **Showtimes**:

**Attributes**: Show ID (Primary Key), Show Date, Show Time, Screen No, Movie ID (Foreign Key)

* **Manager**:

**Attributes:** M\_Name, M\_Age, M\_Gender, Theater ID (Foreign Key)

**Cardinality**: One manager can manage many movie shows. (one to many)

* **Tickets**:

**Attributes**: Ticket No (Primary Key), Screen No, Movie Name, show Time, show date, Price, Seat ID(Foreign Key), Movie ID(Foreign Key)

* **Payments**:

**Attributes**: Payment Type, Amount, Login ID (Foreign Key)

* **Login**:

**Attributes**: Username, Password, Login ID (Foreign Key)

* **Seat**:

**Attributes**: Seat ID (Primary Key), Seat type, Row number, Theater ID (Foreign Key)

* **Theater:**

**Attributes**: Theater ID (Primary Key), Theater Name, Location, No. of screens

**Relationships:**

* **Users - Logins**:

Relationship: One-to-One

Description: Each user has one login, and each login is associated with only one user.

* **Movies - Shows:**

Relationship: One-to-Many

Description: Each movie can have multiple shows, but each show belongs to only one movie.

* **Bookings - Users:**

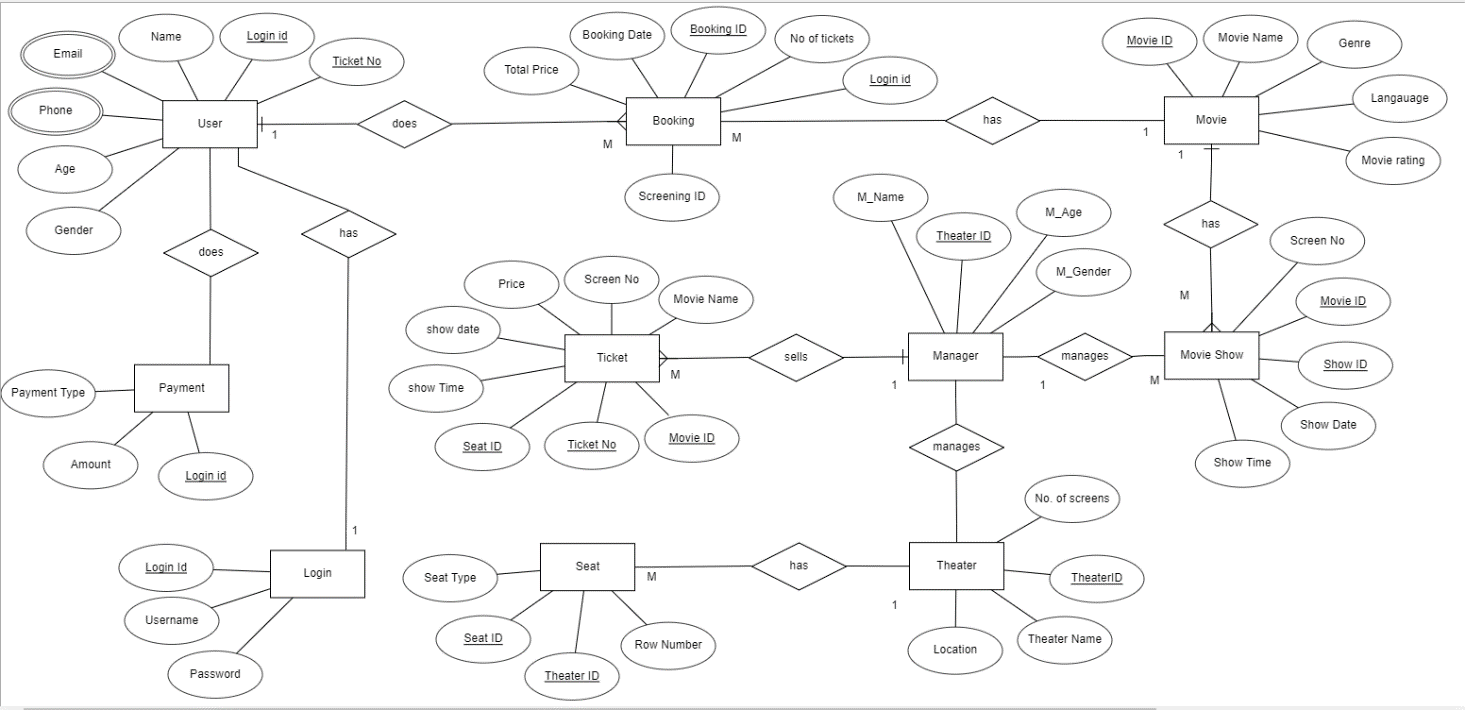
Relationship: Many-to-One

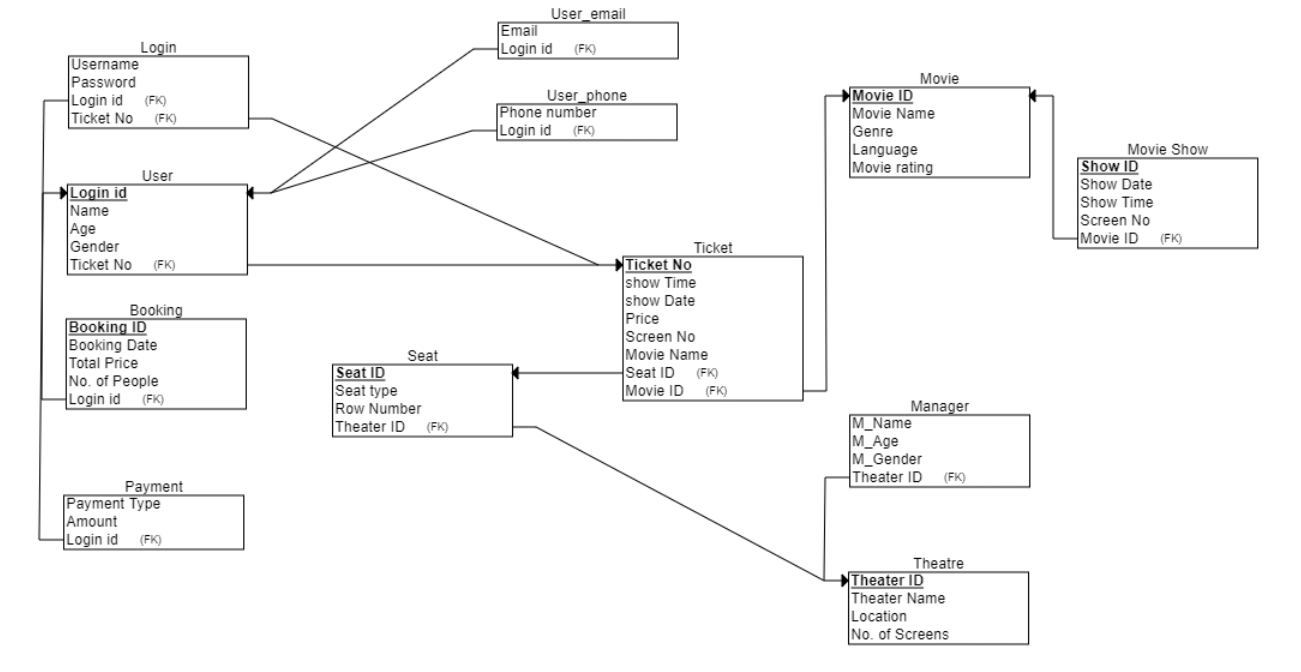
Description: Multiple bookings can be made by one user, but each booking is made by only one user.

* **Manager – Movie Show:**

Relationship: One-to-Many

Description: Each admin can manage multiple movie shows, but each movie show can be managed by only one manager.

**III. Entity Relationship Diagram**

**IV. Relational Model**

**V. Normalization**

In this section, we have applied normalization to the Movie Ticket Booking System database to eliminate data redundancy and improve data integrity. We went through 1NF, 2NF, 3NF, and BCNF to ensure the Movie Ticket Booking System database was properly normalized.

**First Normal Form (1NF):**

Our current database is already in the First Normal Form (1NF) because it meets the requirements of having single values in each cell i.e. the property of Atomicity and avoiding repeated sets of columns.

**Second Normal Form (2NF):**

A relation is in 2NF if it is in 1NF and no partial dependency exists. On checking our existing tables we observe that all the non-key attributes are dependent on the entire primary key which indicates that our tables are already in 2NF.

**Third Normal Form (3NF):**

A relation is in 3NF if it is in 2NF and no transitive dependency exists which means that there should be no cases where a non-key attribute determines another non-key attribute only through another non-key attribute. After checking our tables, we find that transitive dependencies does not exist, which means our tables are already in 3NF.

**BCNF (Boyce-Codd Normal Form):**

A relation is in BCNF if no non-key attribute determines another non-key attribute. This means that every non-trivial functional dependency in the relation must have a superkey as its determinant. Since in our database every determinant is a candidate key and the table is in 3NF, our tables are already in BCNF.

**VI. SQL Queries**

* Table Creation and feeding meaningful data into it:

Create database Movie\_Booking;

create table User (

Name varchar(30) not NULL,

Login\_id numeric,

Password varchar(15),

Email varchar(30),

Phone\_no numeric,

Gender char,

Age numeric,

Ticket\_no numeric,

constraint pk\_User primary key(Login\_id)

);

INSERT into User(Name,Login\_id,Password,Email,Phone\_no,Gender,Age,Ticket\_no) values('Prapti Gupta',1,"abc123",'prapti@gmail.com',9876543210,'F',19,201);

INSERT into User(Name,Login\_id,Password,Email,Phone\_no,Gender,Age,Ticket\_no) values('Tanisha Shaha',2,"def456",'tanisha@gmail.com',0123456789,'F',19,202);

INSERT into User(Name,Login\_id,Password,Email,Phone\_no,Gender,Age,Ticket\_no) values('Mahek Makhija',3,"hello987",'mahek@gmail.com',8763425608,'F',22,203);

INSERT into User(Name,Login\_id,Password,Email,Phone\_no,Gender,Age,Ticket\_no) values('Agam Singh',4,"hi987",'agam@gmail.com',9871239747,'M',29,204);

INSERT into User(Name,Login\_id,Password,Email,Phone\_no,Gender,Age,Ticket\_no) values('Neelam Gupta',5,"xyz123",'neelam@gmail.com',8357290966,'F',47,205);

INSERT into User(Name,Login\_id,Password,Email,Phone\_no,Gender,Age,Ticket\_no) values('Krish Surti',6,"kkr143",'krish@gmail.com',9863638211,'M',36,206);

INSERT into User(Name,Login\_id,Password,Email,Phone\_no,Gender,Age,Ticket\_no) values('Aadit Khanolkar',7,"123pqr",'aadit@gmail.com',9820963210,'M',65,207);

INSERT into User(Name,Login\_id,Password,Email,Phone\_no,Gender,Age,Ticket\_no) values('Yash Malhotra',8,"a03a",'yash@gmail.com',9098763210,'M',19,208);

INSERT into User(Name,Login\_id,Password,Email,Phone\_no,Gender,Age,Ticket\_no) values('Vaishnavi Awashti',9,"qwert87",'vaishnavi@gmail.com',1234509876,'F',32,209);

INSERT into User(Name,Login\_id,Password,Email,Phone\_no,Gender,Age,Ticket\_no) values('Lokendra Pandey',10,"lpr123",'lokendra@gmail.com',9877073518,'M',52,'210');

select\*from User;

create table Login (

Login\_id numeric,

username varchar(30) Not Null,

password varchar(15),

constraint fk\_Login foreign key(Login\_id) references User(Login\_id)

);

INSERT into Login(Login\_id, username,password) values (1,"me\_prapti","abc123");

INSERT into Login(Login\_id, username,password) values (2,"me\_tanisha","def456");

INSERT into Login(Login\_id, username,password) values (3,"me\_mahek","hello987");

INSERT into Login(Login\_id, username,password) values (4,"me\_agam","hi987");

INSERT into Login(Login\_id, username,password) values (5,"me\_neelam","xyz12");

INSERT into Login(Login\_id, username,password) values (6,"me\_krish","kkr143");

INSERT into Login(Login\_id, username,password) values (7,"me\_aadit","123pqr");

INSERT into Login(Login\_id, username,password) values (8,"me\_yash","a03a");

INSERT into Login(Login\_id, username,password) values (9,"me\_vaishnavi","qwert87");

INSERT into Login(Login\_id, username,password) values (10,"me\_lokendra","lpr123");

select\*from Login;

create table Payment (

Login\_id numeric,

Payment\_Type varchar(15) Not Null,

Amount numeric,

constraint fk\_Payment foreign key(Login\_id) references User(Login\_id)

);

INSERT into Payment(Login\_id,Payment\_Type,Amount) values (1, "Credit Card", 350);

INSERT into Payment(Login\_id,Payment\_Type,Amount) values (2, "Net Banking", 700);

INSERT into Payment(Login\_id,Payment\_Type,Amount) values (3, "UPI", 1400);

INSERT into Payment(Login\_id,Payment\_Type,Amount) values (4, "Credit Card", 1050);

INSERT into Payment(Login\_id,Payment\_Type,Amount) values (5, "Credit Card", 350);

INSERT into Payment(Login\_id,Payment\_Type,Amount) values (6, "UPI", 350);

INSERT into Payment(Login\_id,Payment\_Type,Amount) values (7, "UPI", 1400);

INSERT into Payment(Login\_id,Payment\_Type,Amount) values (8, "Credit Card", 700);

INSERT into Payment(Login\_id,Payment\_Type,Amount) values (9, "Debit Card", 350);

INSERT into Payment(Login\_id,Payment\_Type,Amount) values (10, "Net Banking", 1050);

select\*from Payment;

create table Booking (

Login\_id numeric,

No\_of\_Tickets numeric,

Price numeric,

Booking\_id varchar(10),

Booking\_Date date,

constraint fk\_Booking foreign key(Login\_id) references User(Login\_id)

);

INSERT into Booking(Login\_id,No\_of\_Tickets,Price,Booking\_id,Booking\_Date) values(1,1,350,"#1234","2024-03-26");

INSERT into Booking(Login\_id,No\_of\_Tickets,Price,Booking\_id,Booking\_Date) values(2,2,700,"#5678","2024-03-21");

INSERT into Booking(Login\_id,No\_of\_Tickets,Price,Booking\_id,Booking\_Date) values(3,4,1400,"#3746","2024-02-26");

INSERT into Booking(Login\_id,No\_of\_Tickets,Price,Booking\_id,Booking\_Date) values(4,3,1050,"#2937","2024-03-30");

INSERT into Booking(Login\_id,No\_of\_Tickets,Price,Booking\_id,Booking\_Date) values(5,1,350,"#9736","2024-03-31");

INSERT into Booking(Login\_id,No\_of\_Tickets,Price,Booking\_id,Booking\_Date) values(6,1,350,"#0924","2024-03-5");

INSERT into Booking(Login\_id,No\_of\_Tickets,Price,Booking\_id,Booking\_Date) values(7,4,1400,"#1952","2024-03-27");

INSERT into Booking(Login\_id,No\_of\_Tickets,Price,Booking\_id,Booking\_Date) values(8,2,700,"#0176","2024-03-16");

INSERT into Booking(Login\_id,No\_of\_Tickets,Price,Booking\_id,Booking\_Date) values(9,1,350,"#1298","2024-03-8");

INSERT into Booking(Login\_id,No\_of\_Tickets,Price,Booking\_id,Booking\_Date) values(10,3,1050,"#1271","2024-03-19");

select\*from Booking;

create table Movie (

Movie\_id numeric,

Movie\_Name varchar(50),

Genre varchar(15),

Language varchar(15),

Movie\_Rating decimal(10,1),

constraint pk\_Movie primary key(Movie\_id)

);

INSERT into Movie(Movie\_id,Movie\_Name,Genre,Language,Movie\_Rating)

values(16, 'Dangal', 'Biography', 'Hindi',4.2),

(17, 'Baahubali: The Beginning', 'Action', 'Telugu',4),

(18, '3 Idiots', 'Comedy', 'Hindi',4.8),

(19, 'Drishyam', 'Thriller', 'Malayalam',4.8),

(20, 'PK', 'Comedy-Drama', 'Hindi',4.6),

(21, 'Kabir Singh', 'Romance', 'Hindi',3.7),

(22, 'Rang De Basanti', 'Drama', 'Hindi',3.2),

(23, 'Queen', 'Drama-Comedy', 'Hindi',3.5),

(24, 'Ustad Hotel', 'Drama', 'Malayalam',2.8),

(25, 'Gully Boy', 'Drama', 'Hindi',3.0);

select\*from Movie;

create table Movie\_Show(

Movie\_id numeric,

Show\_id numeric,

Show\_Date Date,

Show\_Time time,

Screen numeric,

constraint fk\_Show foreign key(Movie\_id) references Movie(Movie\_id)

);

INSERT INTO Movie\_Show (Movie\_id, Show\_id, Show\_Date, Show\_Time,Screen)

VALUES

(16, 1, '2024-03-27', '15:00',3),

(17, 2, '2024-03-28', '18:30',6),

(18, 3, '2024-03-29', '21:00',5),

(19, 4, '2024-03-27', '17:30',4),

(20, 5, '2024-03-28', '20:00',3),

(21, 6, '2024-03-29', '13:00',4),

(22, 7, '2024-03-27', '16:30',2),

(23, 8, '2024-03-28', '19:45',5),

(24, 9, '2024-03-29', '22:15',3),

(25, 10, '2024-03-27', '14:00',4);

select\*from Movie\_Show;

create table Theater (

Theatre\_id numeric,

Theatre\_Name varchar(50),

Location varchar(15),

Screen numeric,

constraint pk\_Theater primary key(Theatre\_id)

);

INSERT INTO Theater (Theatre\_id, Theatre\_Name, Location, Screen)

VALUES

(101, 'Regal Cinemas', 'Colaba', 3),

(102, 'PVR Cinemas', 'Andheri', 6),

(103, 'INOX Cinemas', 'Bandra', 5),

(104, 'Cinepolis', 'Malad', 4),

(105, 'Miraj Cinemas', 'Chembur', 3),

(106, 'Carnival Cinemas', 'Borivali', 4),

(107, 'Metro Cinema', 'Marine Lines', 2),

(108, 'MAX Cinemas', 'Goregaon', 5),

(109, 'Movietime Cinemas', 'Kandivali', 3),

(110, 'Fame Cinemas', 'Vashi', 4);

select\*from Theater;

create table Seat (

Seat\_id numeric,

Seat\_Type varchar(50),

Row\_no varchar(5),

constraint pk\_Seat primary key(Seat\_id)

);

INSERT INTO Seat (Seat\_id, Seat\_Type, Row\_no)

VALUES

(1, 'Standard', "J"),

(2, 'Standard', "I"),

(3, 'Standard', "H"),

(4, 'Standard', "G"),

(5, 'VIP', "D"),

(6, 'VIP', "E"),

(7, 'VIP', "F"),

(8, 'Premium', "A"),

(9, 'Premium', "B"),

(10, 'Premium', "C");

select\*from Seat;

create table Manager(

Theatre\_id numeric,

M\_Name varchar(20),

M\_Gender char,

M\_Age numeric,

constraint fk\_Manager foreign key(Theatre\_id) references Theater(Theatre\_id)

);

INSERT INTO Manager (Theatre\_id, M\_Name, M\_Gender, M\_Age)

VALUES

(101, 'Priya Sharma', 'F', 32),

(102,'Rahul Desai', 'M', 29),

(103,'Anjali Patel', 'F', 35),

(104,'Amit Kumar', 'M', 40),

(105, 'Neha Gupta', 'F', 28),

(106,'Rajesh Singh', 'M', 45),

(107,'Pooja Joshi', 'F', 33),

(108,'Vikram Sharma', 'M', 38),

(109,'Sunita Reddy', 'F', 30),

(110,'Alok Verma', 'M', 42);

select\*from Manager;

CREATE INDEX idx\_show\_time\_date ON Movie\_Show (Show\_Time, Show\_Date);

CREATE TABLE ticket (

Movie\_id numeric,

Ticket\_no numeric,

Seat\_id numeric,

Show\_Time time,

Show\_Date date,

Price DECIMAL(10, 2),

Screen VARCHAR(50),

Movie\_Name VARCHAR(100),

FOREIGN KEY (Movie\_id) REFERENCES Movie(Movie\_id),

FOREIGN KEY (Seat\_id) REFERENCES Seat(Seat\_id),

FOREIGN KEY (Show\_Time, Show\_Date) REFERENCES Movie\_Show(Show\_Time, Show\_Date)

);

INSERT INTO ticket (Movie\_id, Ticket\_no, Seat\_id, Show\_Time, Show\_Date, Price, Screen, Movie\_Name)

VALUES

(16, 201, 1, '15:00', '2024-03-27', 300, 3, 'Dangal'),

(17, 202, 2, '18:30', '2024-03-28', 700, 6, 'Baahubali: The Beginning'),

(18, 203, 3, '21:00', '2024-03-29', 1400, 5, '3 Idioits'),

(16, 204, 4, '15:00', '2024-03-27', 1050, 4, 'Dangal'),

(25, 205, 5, '14:00', '2024-03-27', 700, 4, 'Gully Boy'),

(22, 206, 6, '16:30:00', '2024-03-27', 350, 2, 'Rand De Basanti'),

(19, 207, 7, '17:30', '2024-03-27', 350, 4, 'Drishyam'),

(24, 208, 8, '22:15', '2024-03-29', 1050, 3, 'Ustad Hotel'),

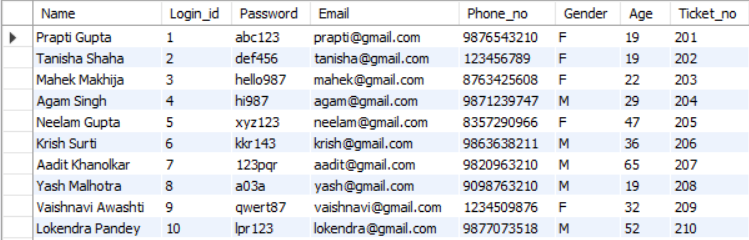
(23, 209, 9, '19:45', '2024-03-28', 1400, 5, 'Queen'),

(25, 210, 10, '14:00', '2024-03-27', 700, 4, 'Gully Boy');

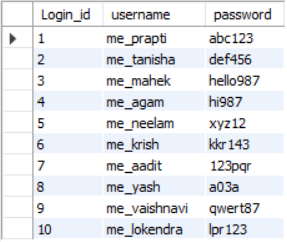
select\*from Ticket;

Snapshots of Tables Created:

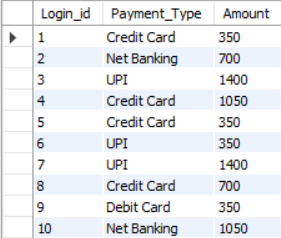
**User:**



**Login:**



**Payment:**

****

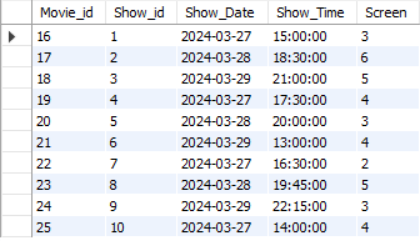
**Booking:**

****

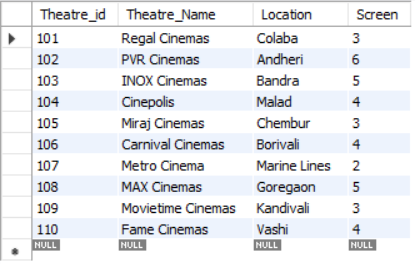
**Movie:**

****

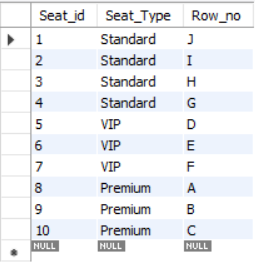
**Movie\_Show:**

****

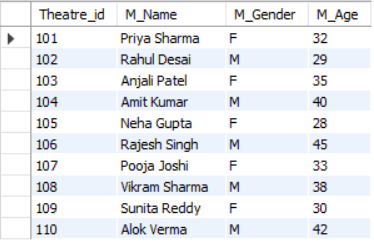
**Theater:**

****

**Seat:**

****

**Manager:**

****

**Ticket:**

****

**Queries:**

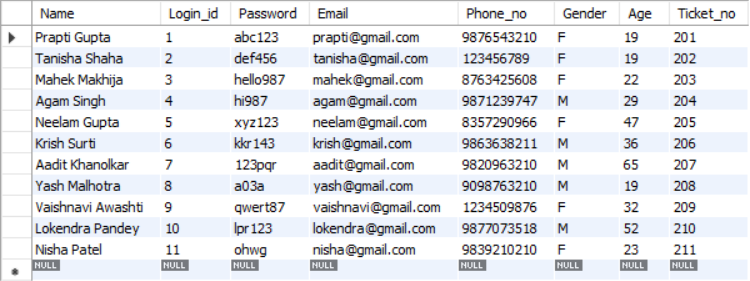
1. Add a user.

Ans:

INSERT into User(Name,Login\_id,Password,Email,Phone\_no,Gender,Age,Ticket\_no) values('Nisha Patel',11,"ohwg",'nisha@gmail.com',9839210210,'F',23,'211');

select\*from User;

Output:



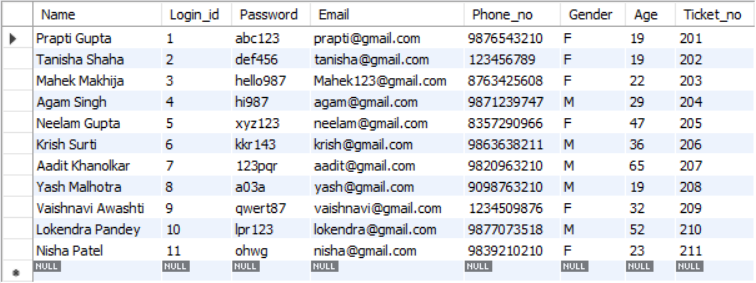
1. Update email id:

Ans:

UPDATE user set Email="Mahek123@gmail.com" where Login\_id=3;

select\*from User;

Output:



1. Update the amount in the payments table to add taxes.

Ans:

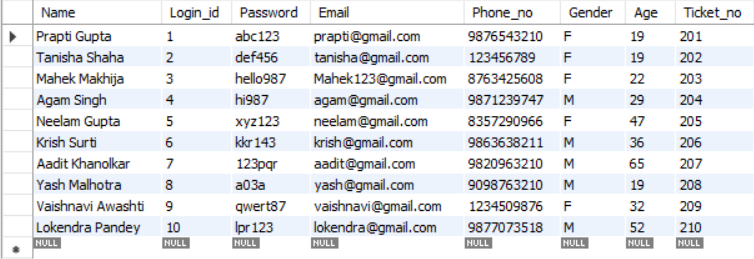
Update payment

SET amount= amount+(0.15\*amount);

SELECT\*from payment;

SELECT\*from User;

Output:



1. Add 10% festive discount

Ans:

SELECT\*price-(price\*0.1) from Booking;

Output:

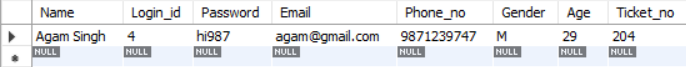


1. Query to find details of a customer whose name contains "Ag" string

Ans:

SELECT \*FROM User WHERE Name LIKE '%Ag%';

Output:



1. Query to calculate the total earning from the site (sum of Price)

Ans:

SELECT SUM(Price) AS Total\_Earning FROM Booking;

Output:



1. Query to get the count of number of male and female managers

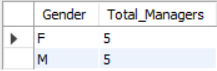
Ans:

SELECT M\_Gender AS Gender,

COUNT(\*) AS Total\_Managers

FROM Manager GROUP BY M\_Gender;

Output:



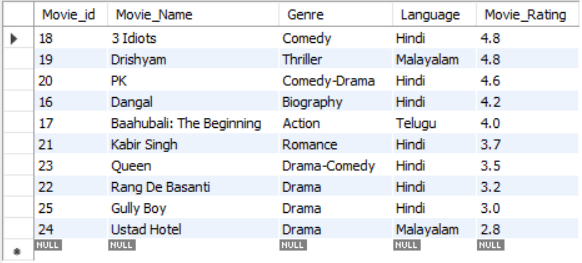
1. Query to order movies by ratings in descending order

Ans:

SELECT Movie\_id,Movie\_Name, Genre,Language,Movie\_Rating FROM Movie

ORDER BY Movie\_Rating DESC;

Output:



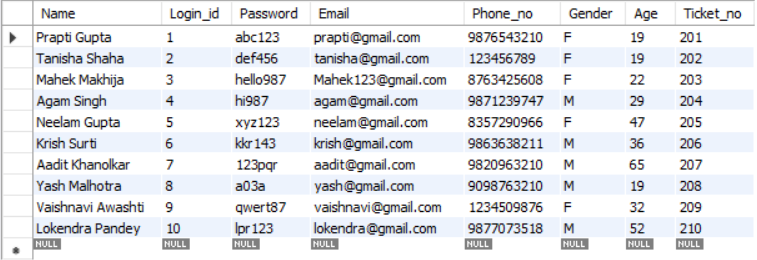
1. Query to delete an account (entry) from the User table

Ans:

DELETE FROM User WHERE Login\_id = 11;

SELECT\* from user;

Output:



1. Query to Search movie by movie by genre and language

Ans:

SELECT \* FROM Movie WHERE Genre = 'Comedy' and language="Hindi";

Output:

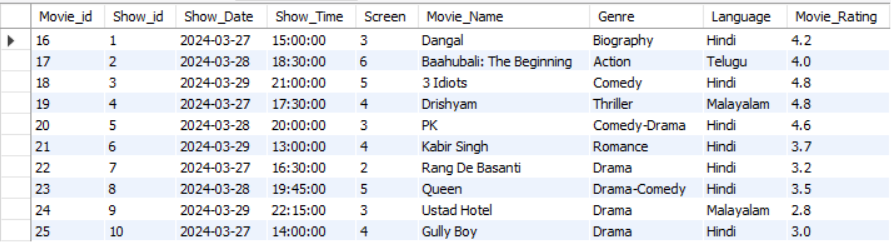


1. Query to display all the data about the movie with its show timings and other details.

Ans:

Select \* from movie\_Show NATURAL JOIN Movie;

Output:



1. Query to search for a manager working in the particular theater

Ans:

SELECT \* FROM Manager WHERE Theatre\_id = 101;

Output:

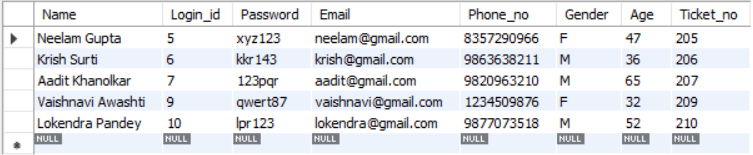


1. Query to get details about users under the age 30

Ans:

SELECT \* from User where age>=30;

Output:



1. Count the total number of seats by seat type (Standard, VIP, Premium) and display them in a descending order based on the number of seats for each type:

Ans:

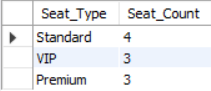
SELECT Seat.Seat\_Type,

COUNT(\*) as Seat\_Count FROM Seat

GROUP BY Seat.Seat\_Type

ORDER BY Seat\_Count DESC;

Output:



1. Retrieve the names and ages of managers who oversee theaters with at least 5 screens..

Ans:

SELECT M\_Name, M\_Age

FROM Manager

WHERE Theatre\_id IN (

SELECT Theatre\_id

FROM Theater

WHERE Screen >= 5);

Output:



1. Retrieve Movie Names and Show Dates for Movies Scheduled on March 27, 2024:

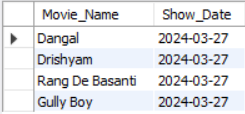
Ans:

SELECT m.Movie\_Name, ms.Show\_Date

FROM Movie\_Show ms

JOIN Movie m ON ms.Movie\_id = m.Movie\_id

WHERE ms.Show\_Date = '2024-03-27';

Output:

1. Create a view to show total amount spent by each user

Ans:

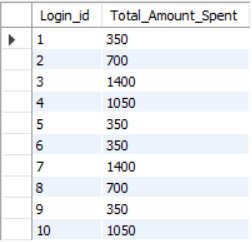
CREATE VIEW TotalAmountSpent AS

SELECT Login\_id, SUM(Price) AS Total\_Amount\_Spent FROM Booking

GROUP BY Login\_id;

SELECT \* FROM TotalAmountSpent;

Output:



1. Modifying table by adding a column Salary in the manager

Ans:

ALTER TABLE Manager

ADD COLUMN Salary numeric(10, 2);

UPDATE Manager SET Salary = 50000.00 WHERE Theatre\_id = 101;

UPDATE Manager SET Salary = 60000.00 WHERE Theatre\_id = 102;

UPDATE Manager SET Salary = 70000.00 WHERE Theatre\_id = 103;

UPDATE Manager SET Salary = 80000.00 WHERE Theatre\_id = 104;

UPDATE Manager SET Salary = 90000.00 WHERE Theatre\_id = 105;

UPDATE Manager SET Salary = 100000.00 WHERE Theatre\_id = 106;

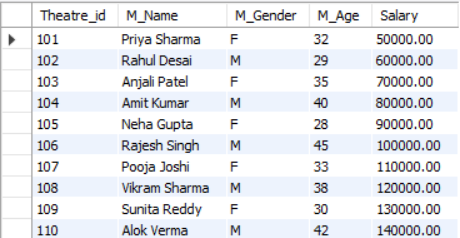
UPDATE Manager SET Salary = 110000.00 WHERE Theatre\_id = 107;

UPDATE Manager SET Salary = 120000.00 WHERE Theatre\_id = 108;

UPDATE Manager SET Salary = 130000.00 WHERE Theatre\_id = 109;

UPDATE Manager SET Salary = 140000.00 WHERE Theatre\_id = 110;

select \* from Manager;

Output:

1. Display the names of managers whose salray is between 50000 and 70000

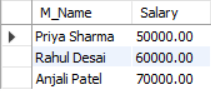
Ans:

SELECT M\_Name, Salary

FROM Manager

WHERE Salary BETWEEN 50000.00 AND 70000.00;

Output:



1. Providing special offers for female users on occasion of women's day

Ans:

UPDATE Ticket

Price = Price \* 0.9

WHERE Show\_Date = '2024-03-28' -- Women's Day

AND Ticket.Ticket\_no IN (

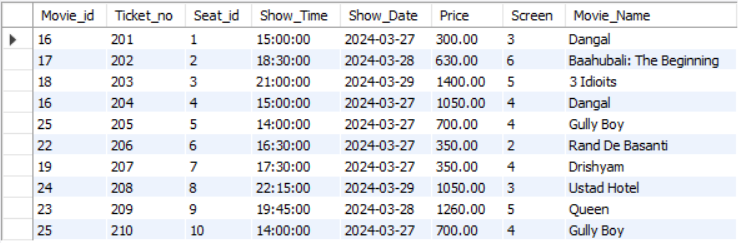
SELECT User.Ticket\_no

FROM User

WHERE User.Gender = 'F');

SELECT\*from Ticket;

Output:



**VI. Project demonstration**

For this project, the following tools are used:

* **MySQL Database Management System:** Used for storing and managing data related to users, movies, showtimes, bookings, and payments.
* **MySQL Workbench:** Used as the primary database management system.
* **ERD Plus:** Used to create Entity-relationship models and Relational Schema

**VII. Self -Learning beyond classroom:**

We discovered how to include multiple foreign keys in a table, each of which isn't theprimary key of another table. To achieve this, we created an index, which helped organize and speed up the retrieval of data related to these foreign keys. This approach enabled us to create connections between various tables within the database.

**VIII. Learning from the Project**

This project helped us learn a lot about managing and designing databases using MySQL Workbench. We figured out how to organize data better. Learning things like normalization and indexing helped us understand databases even more.

Additionally, creating Entity-Relationship (ER) models and relational schemas improved our ability to organize data logically. Overall, this project helped us get better at writing SQL queries, managing databases, and organizing data. It gave us a good understanding of how to handle databases effectively.

**IX. Challenges Faced**

The main challenge we faced while working on this project was that in order to make our searches more effective and meaningful, we constantly had to introduce new attributes into our database tables. Each time we did this, we also had to adjust and update the Entity-Relationship (ER) model and relational schema accordingly.

**X. Conclusion**

We learned to effectively structure and optimize databases for better performance and data integrity.