<u>CineBooker</u>

Final Project For Sql Module

By Sanket Kambli

➢ **Description** :

CineBooker is an innovative Movie Management System (MMS), designed to enhance the movie-going experience. This comprehensive case study introduces a carefully designed approach to movie management, using basic techniques and tools, leveraging the power of relational database management systems (RDBMS). It efficiently tracks movie data, user preferences, and theater information, providing a seamless platform for both movie enthusiasts and theater operators.

This system offers accurate tracking of movie screenings,timings, user bookings, and theater capacities, and also shows available seats for the next user ensuring optimal resource utilization. It simplifies billing processes by using various payment methods, maintains customer records, and analyzes movie ratings, Helping theaters provide great movie experiences while making operations smoother.

Benefits :

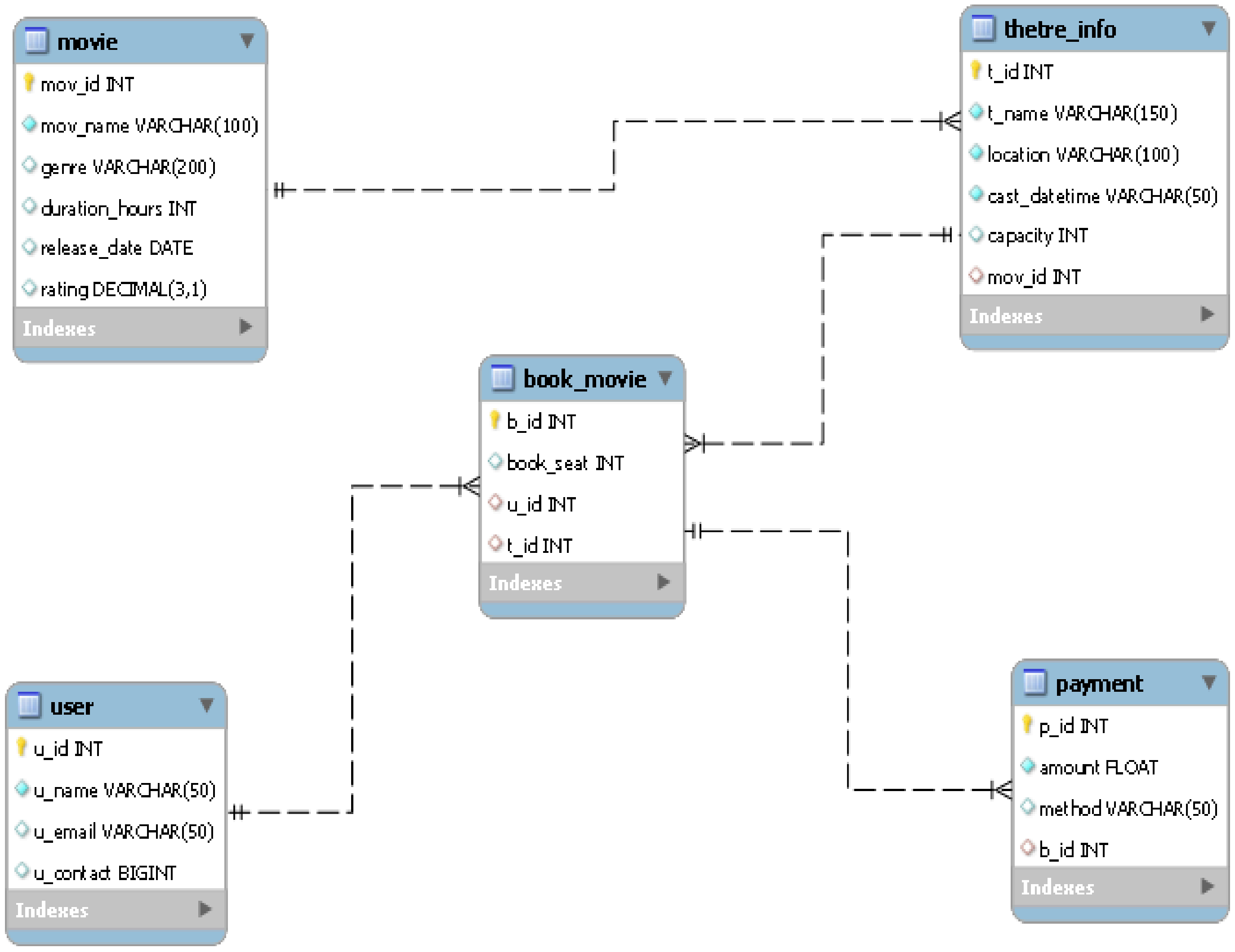
➔ Users can easily browse movie listings, check showtimes, and book tickets with just a few clicks, enhancing convenience and saving time.
➔ Theater owners can gain valuable insights into customer preferences and booking trends.
➔ CineBooker supports various payment methods, ensuring a seamless and secure transaction process for users.
➔ By efficiently managing seat bookings and optimizing theater capacities, cinema owners can maximize revenue potential and ensure an enjoyable movie experience for customers .

➢ **Project Overview** :

The Database Structure of CineBooker :-

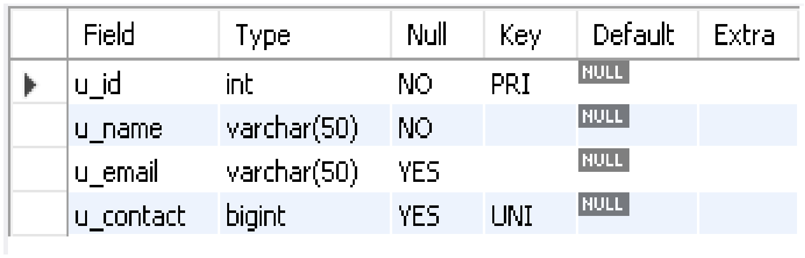
- User Table:
  - ➢ u_id (PK)
  - ➢ u_name
  - ➢ u_email
  - ➢ u_contact
- Movie Table:
  - ➢ mov_id (Primary Key)
  - ➢ mov_name
  - ➢ genre
  - ➢ duration_hours
  - ➢ release_date
  - ➢ rating
- Theater-info Table:
  - ➢ t_id (PK)
  - ➢ t_name
  - ➢ location
  - ➢ cast_datetime
  - ➢ capacity
  - ➢ movie_id (FK-Movie_Table)
- Booking Information Table (Booking):
  - ➢ b_id (PK)
  - ➢ book_seat
  - ➢ user_id (FK-User_Table)
  - ➢ theater_id (FK-Theater_info_Table)
- Payment Table:
  - ➢ p_id (PK)
  - ➢ b_id (FK-Booking_Table)
  - ➢ amount
  - ➢ Payment_method

➢ **<u>ER Diagram Of CineBooker</u> :**



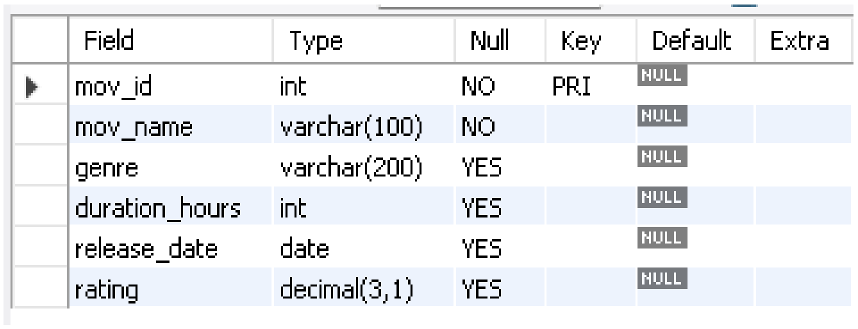
➢ **Table Description :**

- **user :**



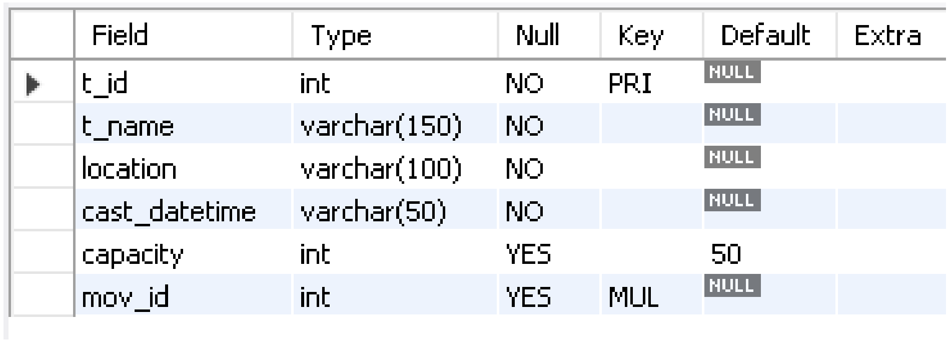
	Field	Type	Null	Key	Default	Extra
▶	u_id	int	NO	PRI	NULL	
	u_name	varchar(50)	NO		NULL	
	u_email	varchar(50)	YES		NULL	
	u_contact	bigint	YES	UNI	NULL	

- **movie:**



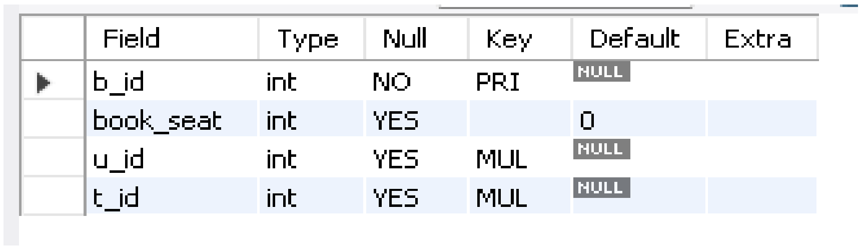
	Field	Type	Null	Key	Default	Extra
▶	mov_id	int	NO	PRI	NULL	
	mov_name	varchar(100)	NO		NULL	
	genre	varchar(200)	YES		NULL	
	duration_hours	int	YES		NULL	
	release_date	date	YES		NULL	
	rating	decimal(3,1)	YES		NULL	

- **thetre_info:**



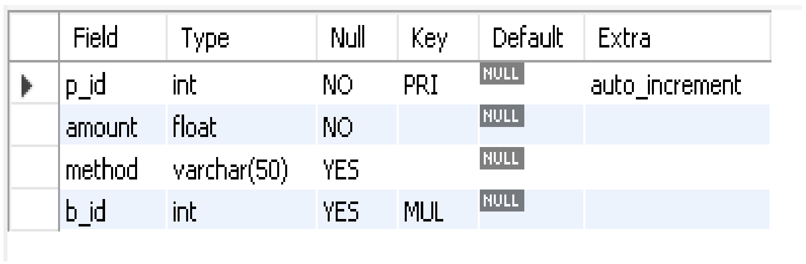
	Field	Type	Null	Key	Default	Extra
▶	t_id	int	NO	PRI	NULL	
	t_name	varchar(150)	NO		NULL	
	location	varchar(100)	NO		NULL	
	cast_datetime	varchar(50)	NO		NULL	
	capacity	int	YES		50	
	mov_id	int	YES	MUL	NULL	

- **book_movie:**



	Field	Type	Null	Key	Default	Extra
▶	b_id	int	NO	PRI	NULL	
	book_seat	int	YES		0	
	u_id	int	YES	MUL	NULL	
	t_id	int	YES	MUL	NULL	

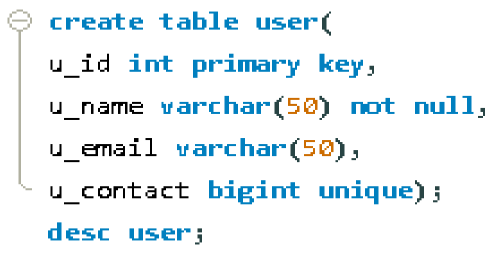
- **payment:**



	Field	Type	Null	Key	Default	Extra
▶	p_id	int	NO	PRI	NULL	auto_increment
	amount	float	NO		NULL	
	method	varchar(50)	YES		NULL	
	b_id	int	YES	MUL	NULL	

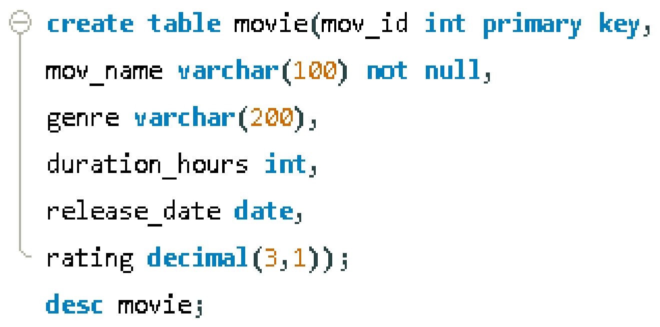
➢ <u>**DDL Commands (Creation of tables)**</u>**:**

- **Creating User Table :**



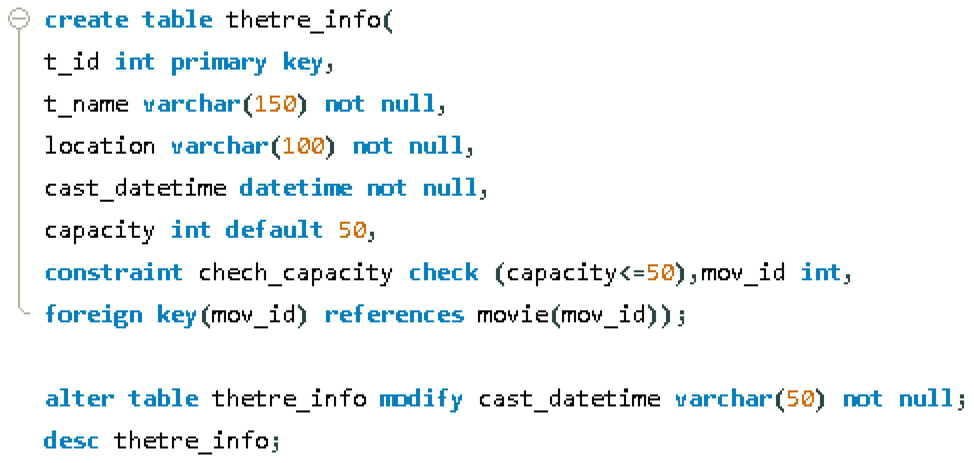
```
create table user(
u_id int primary key,
u_name varchar(50) not null,
u_email varchar(50),
u_contact bigint unique);
desc user;
```

- **Creating Movie Table :**



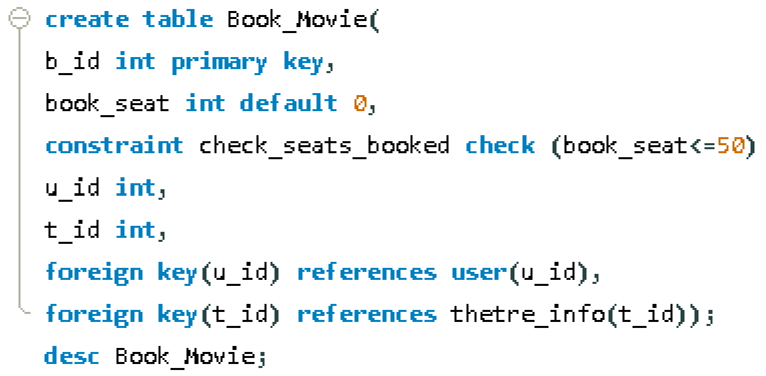
```
create table movie(mov_id int primary key,
mov_name varchar(100) not null,
genre varchar(200),
duration_hours int,
release_date date,
rating decimal(3,1));
desc movie;
```

- **Creating Thetre_Info Table :**



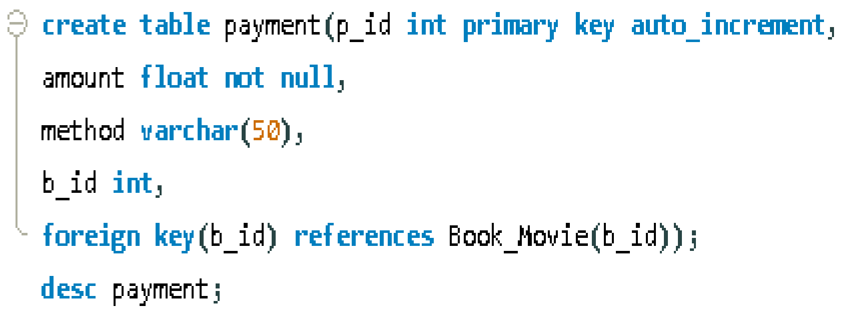
```
create table thetre_info(
t_id int primary key,
t_name varchar(150) not null,
location varchar(100) not null,
cast_datetime datetime not null,
capacity int default 50,
constraint chech_capacity check (capacity<=50),mov_id int,
foreign key(mov_id) references movie(mov_id));

alter table thetre_info modify cast_datetime varchar(50) not null;
desc thetre_info;
```

- **Creating Book_Movie Table:**



```
create table Book_Movie(
b_id int primary key,
book_seat int default 0,
constraint check_seats_booked check (book_seat<=50)
u_id int,
t_id int,
foreign key(u_id) references user(u_id),
foreign key(t_id) references thetre_info(t_id));
desc Book_Movie;
```

- **Creating Payment Table :**



```
create table payment(p_id int primary key auto_increment,
amount float not null,
method varchar(50),
b_id int,
foreign key(b_id) references Book_Movie(b_id));
desc payment;
```

➢ DML Commands (Insertion Values):

- **User :**

```
insert into user(u_id,u_name,u_email,u_contact) values
(1,'Sanket','sanketkambli04082001@gmail.com',9898765678),
(2,'Krushna','krushna@gmail.com',4548765678),
(3,'Rajit','rajit@gmail.com',7878765678),
(4,'Rohan','rohan@gmail.com',9888765678),
(5,'Shruti','shruti@gmail.com',9888765777),
(6,'Pragati','pragati@gmail.com',9999765678),
(7,'Ashish','ashish@gmail.com',6668765678),
(8,'Sahil','sahil@gmail.com',9885655678),
(9,'Tejas','tejas@gmail.com',7808765678),
(10,'Pratik','pratik@gmail.com',2228765678),
(11,'Suyog','suyog@gmail.com',9800065678),
(12,'Suyash','suyash@gmail.com',4488765678),
(13,'Hiten','hiten@gmail.com',1188765678),
(14,'Mandar','mandar@gmail.com',1888765678),
(15,'Soham','soham@gmail.com',3188765678);
```

- **Movie :**

```
insert into movie(mov_id,mov_name,genre,duration_hours,release_date,rating) values
(1,'Interstellar','Sci-fi/Adventur',3,'2023-04-08',9.9),
(2,'Free_Guy','Comedy/Action',2,'2024-05-08',7.5),
(3,'Godzilla','Action/Sci-fi',3,'2023-09-07',8.5),
(4,'Iron_Man','Action/Sci-fi',3,'2021-10-07',9.9),
(5,'Avatar','Action/Sci-fi',4,'2022-05-01',9.5),
(6,'The_Avengers','Action/Sci-fi',2,'2024-11-05',9),
(7,'Shogun','Drama/Action',4,'2024-07-04',8.7),
(8,'Minions','Comedy',3,'2019-05-01',9),
(9,'Natasamrat','Family/Drama',3,'2019-05-01',9.9),
(10,'Mission:Impossible_Fallout','Action/Thriller',2,'2020-09-07',9.8),
(11,'The_Matrix',' Action/Sci-fi',3,'2000-08-07',9.9),
(12,'Big_HEro_6','Action/Comedy',3,'2024-09-01',9),
(13,'The_Dark_Knight','Thriller/Crime',4,'2021-09-01',9),
(14,'Man_Of_Steel','Action/Thriller',3,'2022-09-01',8),
(15,'Joker','Thriller/Crime',3,'2025-10-10',9.9);
```

- **Thetre_info:**

```
insert into thetre_info(t_id,t_name,location,cast_datetime,capacity,mov_id) values
(1,'INOX:Metro_Mall','Kalyan','2024-03-25 12:30',default,1),
(2,'Carnival_Cinemas','Wadala','2024-03-25 4:30 PM',40,13),
(3,'INOX_R-city','Ghatkopar','2024-03-26 1:00 PM',30,9),
(4,'EROS_Cinemas','Churchgate','2024-03-26 8:00 AM',50,2),
(5,'Regal_Cinemas','Colaba','2024-03-26 10:00 AM',50,6),
(6,'PVR_Cinemas','Parel','2024-04-27 11:30 AM',20,7),
(7,'PVR_Phoenix_Marketcity','Krula','2024-04-30 9:30 PM',45,8),
(8,'PVR_Infinity_Mall','Anderi','2024-04-30 11:00 PM',50,9),
(9,'Cinepolis_Viviana_Mall','Thane','2024-05-12 1:00 PM',30,5),
(10,'PVR_Oberoi','Goregaon','2024-06-30 11:00 PM',30,10),
(11,'Plaza_Cinema','Dadar','2024-07-01 12:00 PM',50,15),
(12,'Paradise_Cinema','Mahim','2024-07-12 08:00 PM',40,14),
(13,'Citylight_cinema','Mahim','2024-08-30 12:00 PM',30,15),
(14,'KK_Cinema','Kamothe','2024-09-12 11:30 AM',10,9),
(15,'Woodland_Cinema','Virar','2024-10-10 09:00 PM',50,12);
```

- **Book_Movie :**

```
insert into Book_Movie(b_id,book_seat,u_id,t_id) values
(1,5,1,15),
(2,10,2,14),
(3,2,3,12),
(4,1,4,7),
(5,11,5,9),
(6,15,6,10),
(7,9,7,10),
(8,3,8,10),
(9,10,9,1),
(10,10,10,3),
(11,10,11,5),
(12,10,12,7),
(13,10,13,9),
(14,10,14,4),
(15,10,15,3);
```

- **Payment:**

```
insert into payment (p_id,amount,method,b_id) values
(1,100,'upi',1),
(2,100,'cash',2),
(3,100,'credit',3),
(4,100,'upi',4),
(5,100,'upi',5),
(6,100,'upi',6),
(7,100,'cash',7),
(8,100,'cash',8),
(9,100,'credit',9),
(10,100,'upi',10),
(11,100,'upi',11),
(12,100,'cash',12),
(13,100,'cash',13),
(14,100,'gpay',14),
(15,100,'credit',15);
```

➢ DQL Commands:

Joins and Clauses

- Inner Join to fetch which user have booked for which movie in which theater

```
select u.u_name as User_Name ,u.u_email User_EmailID,t.t_name as Theter_Name,
t.location as Theter_Location,t.cast_datetime as Movie_Schedule,
m.mov_name as Movie_Title,m.release_date as Movie_ReleaseDate,
b.book_seat as Number_of_SeatsBooked
from Book_Movie b
inner join user u on b.u_id = u.u_id
inner join thetre_info t on b.t_id = t.t_id
inner join movie m on t.mov_id = m.mov_id;
```

- Using join and group by clause to show max number of users who booked for a particular movie

```
select m.mov_name as Movie_name ,count(distinct(u.u_id)) as Total_Booking
from book_movie b
inner join user u on b.u_id =u.u_id
inner join thetre_info t on b.t_id = t.t_id
inner join movie m on t.mov_id = m.mov_id
group by m.mov_name
order by Total_Booking desc;
```

- which users have have used different methods for payment

```
select u.u_name ,p.method
from book_movie b
inner join user u on b.u_id = u.u_id
inner join payment p on b.b_id = p.b_id
group by u.u_name,p.method;
```

- Which areas have highest theaters

```
select t.location,count(t.location) as thetre_Count
from thetre_info t
group by t.location
order by thetre_count desc;
```

- Which movies have the highest ratings

```
select distinct(m.mov_name),m.rating
from movie as m
order by m.rating desc;
```

- The average rating of movies for each genre.

```
select m.genre ,avg(m.rating) as Average_Rating
from movie as m
group by m.genre
order by Average_Rating desc;
```

- List of users who booked Total number of seats

```
select u.u_name, b.book_seat
from user u
left join Book_Movie b ON u.u_id = b.u_id;
```

- Calculate the available seats in theater for the next user

```
select t.t_name,t.capacity,b.book_seat, (t.capacity-b.book_seat) as Avaaible_seats
from thetre_info t
inner join book_movie b
on t.t_id = b.t_id;
```

- Inner Join to show all the data in the systematic format and also the total amount they had to pay for each seats

```
select u.u_name as User_Name ,u.u_email User_EmailID,t.t_name as Theter_Name,
t.location as Theter_Location,t.cast_datetime as Movie_Schedule,
m.mov_name as Movie_Title,m.release_date as Movie_ReleaseDate,
b.book_seat as Number_of_SeatsBooked,p.method as Payment_method,
p.amount as Ammount,p.amount*b.book_seat as Total_Ammount
from Book_Movie b
inner join user u on b.u_id = u.u_id
inner join thetre_info t on b.t_id = t.t_id
inner join movie m on t.mov_id = m.mov_id
inner join payment p on b.b_id = p.b_id;
```

Subqueries

- Movie title whose rating is higher than avg rating.

```
select m.mov_name , m.rating
from movie m
where m.rating > (select avg(m.rating) from movie m) limit 1;
```

- select the user who has the maximum number of seats booked.

```
select u.u_name,u.u_contact,u.u_email
from user u
inner join book_movie b
on u.u_id = b.u_id
where u.u_id = (select b.u_id from book_movie b order by book_seat desc limit 1);
```

- Select the users who have booked seats in the cinema 'PVR_Oberoi'.

```
select u.u_name,t.t_name,b.book_seat
from book_movie b
inner join user u on b.u_id = u.u_id
inner join thetre_info t on b.t_id = t.t_id
inner join movie m on m.mov_id = t.mov_id
where u.u_id in (
select u.u_id from book_movie b where t.t_id in (
select t.t_id from thetre_info t where t.t_name = 'PVR_Oberoi')
)
```