JSS MAHAVIDYAPEETHA

SRI JAYACHAMARAJENDRA COLLEGE OF ENGINEERING

JSS SCIENCE AND TECHNOLOGY UNIVERSITY

JSS TECHNICAL INSTITUTIONS CAMPUS

MYSURU - 570 006

DATABASE MANAGEMENT SYSTEM (20CS510)

Event-2

THEATRE BOOKING MANAGEMENT SYSTEM

CONTENTS

INTRODUCTION

In the world of entertainment, the theater industry stands as a pillar of cultural enrichment and artistic expression. Managing a theater, whether large or small, involves a myriad of intricate tasks and responsibilities. From scheduling performances and handling ticket sales to coordinating with artists and ensuring the comfort of the audience, the complexities of theater management are numerous. In the digital age, a robust and efficient database management system is not just a luxury but a necessity to streamline these operations and enhance the overall theater-going experience.

FEATURES OF THE PROJECT

Here are the features of our project:

1.Movie Schedule Management: Movie schedule management is an essential aspect of running a movie theater or cinema. It involves planning, organizing, and maintaining a structured timetable for screening movies to ensure a smooth and efficient operation. Create and manage movie showtimes, including start times, duration, and screen assignments.

2.Seat Booking: Seat booking, also known as seat reservation or ticket booking, is the process of allowing individuals or groups to select and secure specific seats within a theatre for movie screening in advance.

3.Movie Repository:Storing information about available movies, including details such as title, genre, language, screening time and duration is a fundamental aspect of any movie-related database or content management system.

4.Showtime Selection:Showtime selection in a movie database is a crucial feature that allows users to choose from available screening times for a particular movie. It is an integral part of any online movie booking or cinema management system.

LIMITATIONS

Limitations of our website:

1. There will be only 4 movies shown on one day.

One screen will show only one movie throughout the day

2. Only the movies playing today will be shown and customer can book before the time slot on that particular day itself.

3.One each day admin/employee will insert the screen table with the movies that will play today and their information along with movie order

DATABASE SCHEMA

1.Ticket : This table describes all the components that a movie ticket can possibly have. It consists of the Ticket-ID (Primary key), Movie-name, Screen-ID, Customer-ID the class of the seat (Silver, Gold, Platinum), Time of the show, Snacks ordered and the price of the ticket.

2.Customer : This table describes the information that the customer provides when registering to the system which will later be used for the ticket. This table consists of the Customer-ID (Primary key), Customer-Name, Ticket-ID, Email, Phone number,Movie selected, slot of the show selected, Snacks selected, Class selected and Membership of the customer.

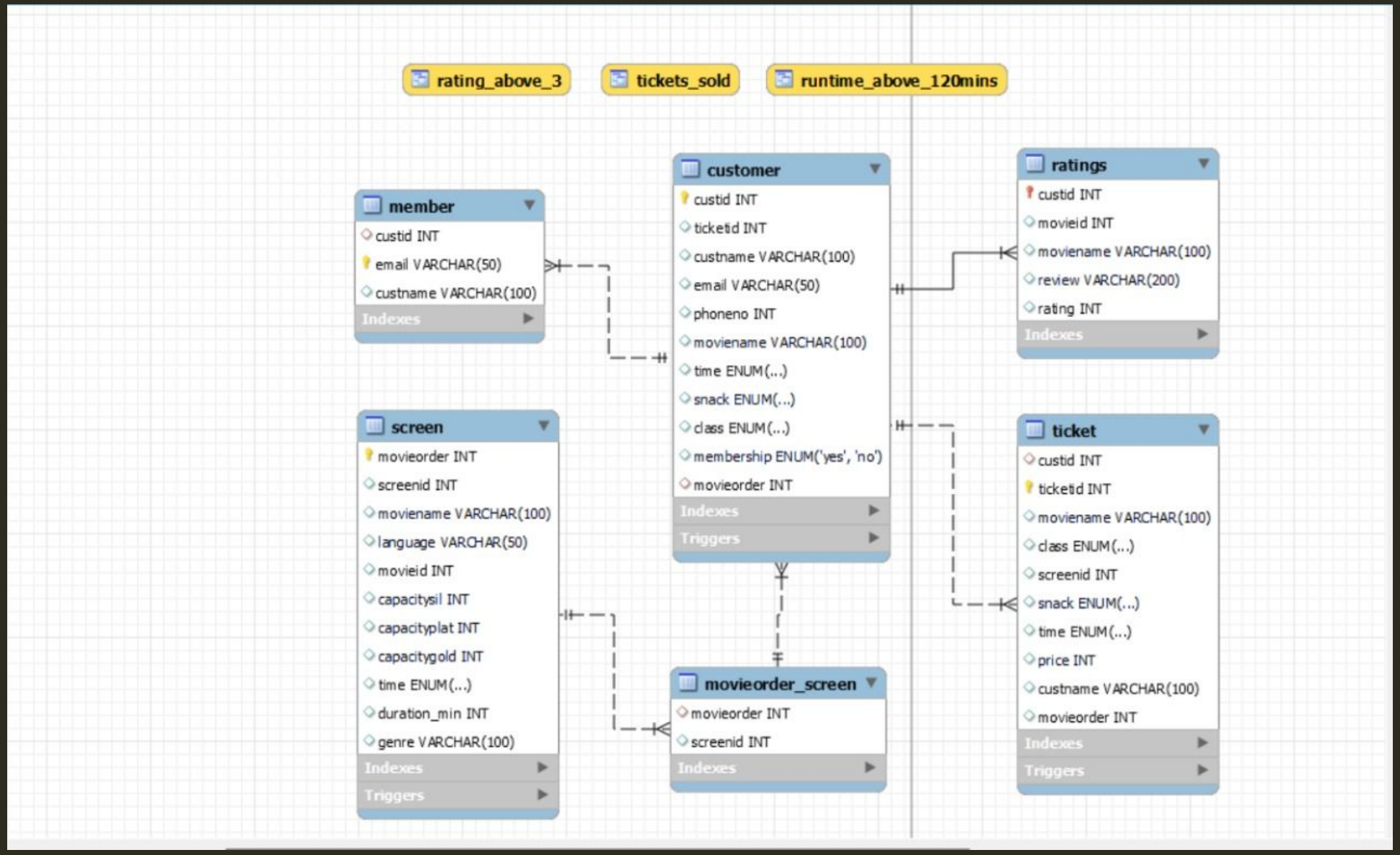
3.Screen : This table describes the different screens present in the theatre having different screen capacities. The attributes are, Screen-ID(Primary Key), Movie-ID, Movie-Name, Time and the Capacity of the screen.

4.MovieOrder_Screen : This table has attributes as movieorder and Screen-ID

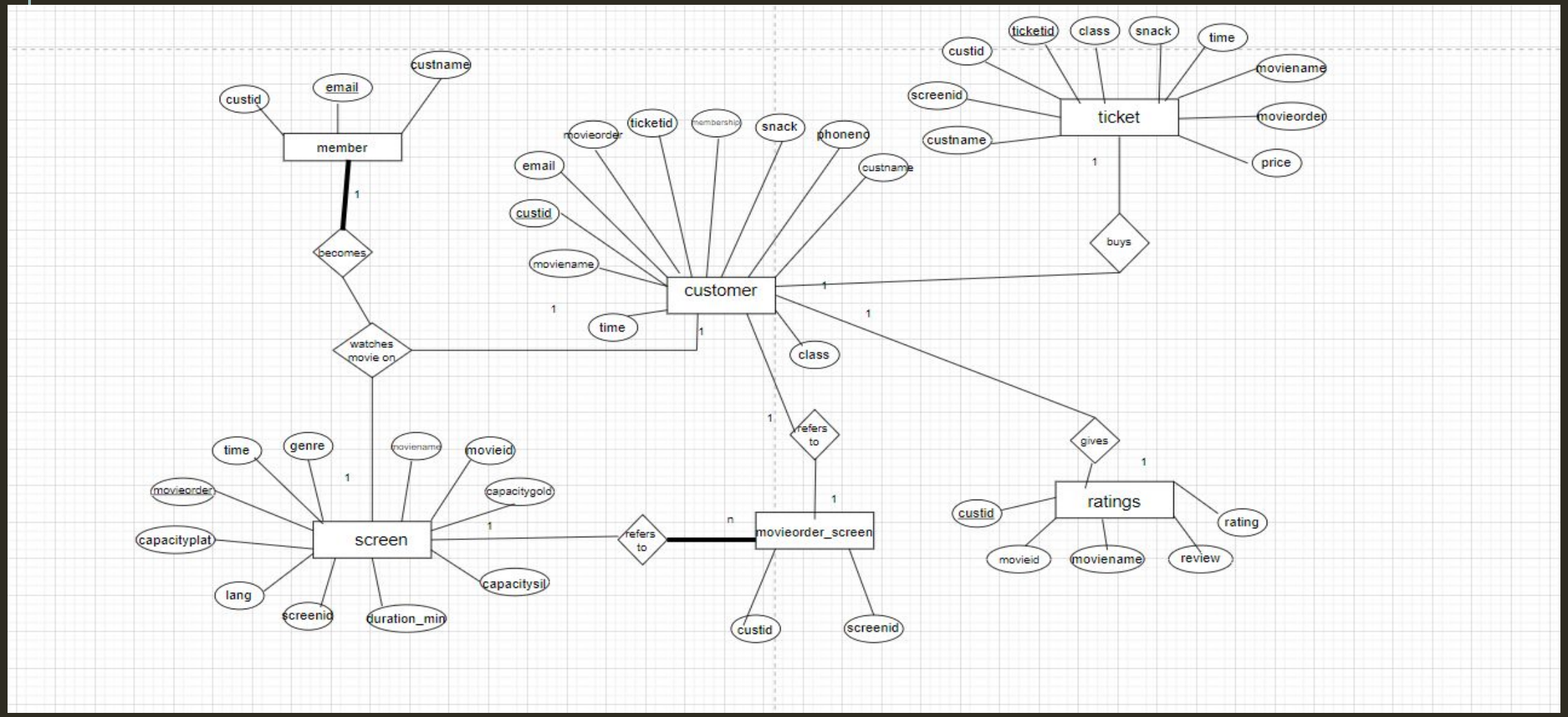
5.Membership: This table has details of the customer who has a membership. It contains Email-ID(Primary Key) and Customer name.

6.Ratings: This table describes the ratings and reviews given by IMdb. It contains Movie-ID, Customer-ID, Movie Name, Review and Rating.

SCHEMA DIAGRAM



Entity Relationship Diagram(ERD)



IMPLEMENTATION

CREATION

```sql
create table screen(movieorder int(2) primary key,screenid int(10),moviename varchar(100),language varchar(50),movieid int(10),capacitysil
int(3),capacityplat int(3),capacitygold int(3),time enum("10 am","3 pm","7 pm"),duration_min int(3),genre varchar(100));

create table movieorder_screen(movieorder int(2),screenid int(10),foreign key (movieorder) references screen(movieorder));

create table customer(custid int(10) auto_increment,ticketid int(10) ,custname varchar(100),email varchar(50),phoneno int(12),moviename varchar(100),
time ENUM("10 am","3 pm","7 pm"),snack ENUM("popcorn","nachos","coca cola"),class ENUM("silver","gold","platinum"),membership ENUM("yes","no"),movieorder int(2),
foreign key (movieorder) references movieorder_screen(movieorder),primary key (custid));

create table ticket(custid int(10) ,ticketid int(10) auto_increment,moviename varchar(100),class ENUM("silver","gold","platinum"),screenid int(10),
snack ENUM("popcorn","nachos","coca cola"),time ENUM("10 am","3 pm","7 pm"),price int(5) default 100,custname varchar(100),movieorder int(2),
foreign key (custid) references customer(custid) on delete cascade,primary key (ticketid));

CREATE TABLE ratings(custid int(10) primary key,
movieid INT NULL,
moviename VARCHAR(100) NULL,
review VARCHAR(200) NULL,
rating INT(3) NULL DEFAULT 0,foreign key (custid) references customer(custid));

create table member(custid int(10),email varchar(50) primary key,custname varchar(100),foreign key (custid) references customer(custid));
```

1.Insertion

Inserting into Customer table

```
insert into customer(ticketid,custname,email,phoneno,moviename,time,snack,class,membership,movieorder) values
(100,"Taneeshka","tan@gmail.com",935927534,"Barbie",'10 am','popcorn','silver','yes',1),
(101,"Natalie","nat@gmail.com",935927567,"Oppenheimer",'3 pm','nachos','gold','yes',5),
(102,"Matt","matt@gmail.com",935923467,"Phir Hera Pheri",'3 pm','coca cola','platinum','no',8),
(103,"John","john@gmail.com",934927567,"Barbie",'7 pm','nachos','silver','yes',3),
(104,"Meghan","meg@gmail.com",944927567,"Train To Busan",'10 am','popcorn','gold','yes',10);
```

Inserting into Screen table

```
insert into screen(movieorder,screenid,moviename,language,movieid,capacitysil,capacityplat,capacitygold,time,duration_min,genre
(1,1,"Barbie","Eng/Hindi",200,50,50,50,"10 am",120,"Fantasy"),
(2,1,"Barbie","Eng/Hindi",200,50,50,50,"3 pm",120,"Fantasy"),
(3,1,"Barbie","Eng/Hindi",200,50,50,50,"7 pm",120,"Fantasy"),
(4,2,"Oppenheimer","Eng",201,50,50,50,"10 am",150,"Fantasy"),
(5,2,"Oppenheimer","Eng",201,50,50,50,"3 pm",150,"Fantasy"),
(6,2,"Oppenheimer","Eng",201,50,50,50,"7 pm",150,"Fantasy"),
(7,3,"Phir Hera Pheri","Hindi",202,50,50,50,"10 am",90,"Fantasy"),
(8,3,"Phir Hera Pheri","Hindi",202,50,50,50,"3 pm",90,"Fantasy"),
(9,3,"Phir Hera Pheri","Hindi",202,50,50,50,"7 pm",90,"Fantasy"),
(10,4,"Train To Busan","Korean",203,50,50,50,"10 am",130,"Fantasy"),
(11,4,"Train To Busan","Korean",203,50,50,50,"3 pm",130,"Fantasy"),
(12,4,"Train To Busan","Korean",203,50,50,50,"7 pm",130,"Fantasy");
```

Inserting into Ratings table

```
insert into ratings(custid,movieid,moviename,review,rating) values
(1,200,"Barbie","Fantastic",4),
(4,203,"Train To Busan","Amazing",5),
(2,201,"Oppenheimer","Thrilling",4);
select *from ratings;
```

2.Showing Tables

Screen Table:

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

movieorder	screenid	moviename	language	movieid	capacitysil	capacityplat	capacitygold	time	duration_min	genre
1	1	Barbie	Eng/Hindi	200	50	50	50	10 am	120	Fantasy
2	1	Barbie	Eng/Hindi	200	50	50	50	3 pm	120	Fantasy
3	1	Barbie	Eng/Hindi	200	50	50	50	7 pm	120	Fantasy
4	2	Oppenheimer	Eng	201	50	50	50	10 am	150	Fantasy
5	2	Oppenheimer	Eng	201	50	50	50	3 pm	150	Fantasy
6	2	Oppenheimer	Eng	201	50	50	50	7 pm	150	Fantasy
7	3	Phir Hera Pheri	Hindi	202	50	50	50	10 am	90	Fantasy
8	3	Phir Hera Pheri	Hindi	202	50	50	50	3 pm	90	Fantasy
9	3	Phir Hera Pheri	Hindi	202	50	50	50	7 pm	90	Fantasy
10	4	Train To Busan	Korean	203	50	50	50	10 am	130	Fantasy
11	4	Train To Busan	Korean	203	50	50	50	3 pm	130	Fantasy
12	4	Train To Busan	Korean	203	50	50	50	7 pm	130	Fantasy

Ticket Table:

	custid	ticketid	moviename	class	screenid	snack	time	price	custname	movieorder
▶	1	100	Barbie	silver	1	popcorn	10 am	170	Taneeshka	1
	2	101	Oppenheimer	gold	2	nachos	3 pm	250	Natalie	5
	3	102	Phir Hera Pheri	platinum	3	coca cola	3 pm	250	Matt	8
	4	103	Barbie	silver	1	nachos	7 pm	200	John	3
	5	104	Train To Busan	gold	4	popcorn	10 am	220	Meghan	10
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Customer Table:

	custid	ticketid	custname	email	phoneno	moviename	time	snack	class	membership	movieorder
▶	1	100	Taneeshka	tan@gmail.com	935927534	Barbie	10 am	popcorn	silver	yes	1
	2	101	Natalie	nat@gmail.com	935927567	Oppenheimer	3 pm	nachos	gold	yes	5
	3	102	Matt	matt@gmail.com	935923467	Phir Hera Pheri	3 pm	coca cola	platinum	no	8
	4	103	John	john@gmail.com	934927567	Barbie	7 pm	nachos	silver	yes	3
	5	104	Meghan	meg@gmail.com	944927567	Train To Busan	10 am	popcorn	gold	yes	10
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Movieorder_screen Table:

movieorder	screenid
1	1
2	1
3	1
4	2
5	2
6	2
7	3
8	3
9	3
10	4
11	4
12	4

Member Table:

custid	email	custname
4	john@gmail.com	John
5	meg@gmail.com	Meghan
2	nat@gmail.com	Natalie
1	tan@gmail.com	Taneeshka
NULL	NULL	NULL

Ratings Table:

custid	movieid	moviename	review	rating
1	200	Barbie	Fantastic	4
2	201	Oppenheimer	Thrilling	4
4	203	Train To Busan	Amazing	5
NULL	NULL	NULL	NULL	NULL

3.Updation

Update the ticket price in the ticket table if the customer already has a membership

```sql
UPDATE ticket
INNER JOIN member ON ticket.custname = member.custname
SET ticket.price=ticket.price-25
WHERE ticket.custname = member.custname;
select *from ticket;
```

custid	ticketid	moviename	class	screenid	snack	time	price	custname	movieorder
1	100	Barbie	silver	1	popcorn	10 am	145	Taneeshka	1
2	101	Oppenheimer	gold	2	nachos	3 pm	225	Natalie	5
3	102	Phir Hera Pheri	platinum	3	coca cola	3 pm	250	Matt	8
4	103	Barbie	silver	1	nachos	7 pm	175	John	3
5	104	Train To Busan	gold	4	popcorn	10 am	195	Meghan	10
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

4.Deletion

Deletion of the ticket booked by a particular customer

```
delete from ticket where custid=5;
select *from ticket;
```

custid	ticketid	moviename	class	screenid	snack	time	price	custname	movieorder
1	100	Barbie	silver	1	popcorn	10 am	95	Taneeshka	1
2	101	Oppenheimer	gold	2	nachos	3 pm	175	Natalie	5
3	102	Phir Hera Pheri	platinum	3	coca cola	3 pm	250	Matt	8
4	103	Barbie	silver	1	nachos	7 pm	125	John	3
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

5.Views

Movies with runtime above 120 minutes

```
CREATE VIEW runtime_above_120mins AS
SELECT DISTINCT movieid, moviename, language, genre FROM screen WHERE duration_min>=120;
select* from runtime_above_120mins;
```

movieid	moviename	language	genre
200	Barbie	Eng/Hindi	Fantasy
201	Oppenheimer	Eng	Fantasy
203	Train To Busan	Korean	Fantasy

No of tickets that got sold per movie by the end of the day

```
CREATE VIEW tickets_sold AS
SELECT moviename,COUNT(moviename) as tickets_sold from ticket
GROUP BY moviename;
select *from tickets_sold;
```

moviename	tickets_sold
Barbie	2
Oppenheimer	1
Phir Hera Pheri	1
Train To Busan	1

Movies with rating above 3

```
CREATE VIEW RATING_ABOVE_3 AS
SELECT DISTINCT s.moviename,s.language,s.genre,r.review
FROM screen s ,ratings r
WHERE s.moviename=r.moviename AND r.rating>3;
select *from rating_above_3;
```

moviename	language	genre	review
Barbie	Eng/Hindi	Fantasy	Fantastic
Oppenheimer	Eng	Fantasy	Thrilling
Train To Busan	Korean	Fantasy	Amazing

6.Triggers

ticket_insert:To insert ticket into ticket table as soon as tuple is inserted into customer table

```
Delimiter //
create trigger
ticket_insert
after insert
on customer
for each row
begin
insert into ticket(custid,ticketid,moviename,class,snack,time,custname,movieorder)
values (new.custid,new.ticketid,new.moviename,new.class,new.snack,new.time,new.custname,new.movieorder);
end //
delimiter ;
```

movie_order_insert:Insert the movie name into the movieorder_screen table as the movies played get updated each day.

```
Delimiter //
create trigger
movie_order_insert
after insert
on screen
for each row
begin
insert into movieorder_screen(movieorder,screenid)
values (new.movieorder,new.screenid);
end //
delimiter ;
```

member_insert:To insert customer details into member table as soon as he opts membership

```
Delimiter //
create trigger
member_insert
after insert
on customer
for each row
begin
IF NEW.membership = 'yes' THEN
insert into member(custid,email,custname)
values (new.custid,new.email,new.custname);
END IF;
end //
delimiter ;
```

trigger_class_price:To update the ticket price according to class selected

```
DELIMITER \\
CREATE TRIGGER trigger_class_price
BEFORE INSERT
ON ticket
FOR EACH ROW
BEGIN
  IF NEW.class = 'platinum' THEN
    SET NEW.price = NEW.price + 100;
  ELSEIF new.class='silver' THEN
    SET NEW.price = NEW.price + 0;
  ELSEIF new.class='gold' THEN
    SET NEW.price = NEW.price + 50;
  END IF;
END \\
DELIMITER ;
```

trigger_snack_price:To update the ticket price according to snack selected

```
DELIMITER \\
CREATE TRIGGER trigger_snack_price
BEFORE INSERT
ON ticket
FOR EACH ROW
BEGIN
  IF NEW.snack = 'popcorn' THEN
    SET NEW.price = NEW.price + 70;
  ELSEIF new.snack='coca cola' THEN
    SET NEW.price = NEW.price + 50;
  ELSEIF new.snack='nachos' THEN
    SET NEW.price = NEW.price + 100;
  END IF;
END \\
DELIMITER ;
```

Update capacity gold,silver and platinum: To decrease the capacity of seats in the screen as tickets get booked

```
create trigger update_capacity_gold
before insert on customer
for each row
UPDATE screen
SET capacitygold=capacitygold-1
where new.class='gold' and time=new.time and moviename=new.moviename;
```

```
create trigger update_capacity_silver
before insert on customer
for each row
UPDATE screen
SET capacitysil=capacitysil-1
where new.class='silver' and time=new.time and moviename=new.moviename;
```

```
create trigger update_capacity_platinum
before insert on customer
for each row
UPDATE screen
SET capacityplat=capacityplat-1
where new.class='platinum' and time=new.time and moviename=new.moviename;
```

Increase capacity gold,silver,platinum: To increase capacity of seats in the screens as soon as the tickets get cancelled

```
create trigger update_capacity_gold_after_delete
after delete on ticket
for each row
UPDATE screen
SET capacitygold=capacitygold+1
where old.class='gold' and time=old.time and moviename=old.moviename;
```

```
create trigger update_capacity_silver_after_delete
after delete on ticket
for each row
UPDATE screen
SET capacitysil=capacitysil+1
where old.class='silver' and time=old.time and moviename=old.moviename;
```

```
create trigger update_capacity_platinum_after_delete
after delete on ticket
for each row
UPDATE screen
SET capacityplat=capacityplat+1
where old.class='platinum' and time=old.time and moviename=old.moviename;
```

Creating a trigger if the capacity is full

```
/*create trigger if capacity is full*/

Delimiter //
create trigger capacity_full
after update
on screen
for each row
begin
 IF NEW.capacitygold=0 || NEW.capacitysil=0|| NEW.capacityplat=0 THEN
 SIGNAL SQLSTATE '45000'
 SET MESSAGE_TEXT="booking complete for this session cannot book anymore";
END IF;
end //
delimiter ;
drop trigger capacity_full;
```

CONCLUSION

In summary, the Theatre Database Management System project effectively fulfills the requirements for efficient data management and retrieval in the theater industry. This comprehensive system optimizes the management of theatrical productions, cast and crew information, scheduling, and ticket sales while also enhancing the user experience for theater administrators and patrons. With its user-friendly interface and robust database structure, this project has the potential to transform the way theaters operate and engage with their audience. As technology advances, the implementation of this system represents a significant stride toward a more organized, efficient, and enjoyable theater experience for all stakeholders.

By

Sukruti S Kadagadakai (01JCE21CS108)

Taneeshka Naganath Reddy

(01JCE21CS116)

Shaik Rihan Mehnaz (01JST21CS183)

Sudhiksha B A (01JCE21CS106)

Under the guidance of

K S MAHESH
Assistant Professor
Dept of CS & E
SJCE, JSSSTU
Mysuru.