



MOVIE TICKET BOOKING SYSTEM

MOVIE TICKET BOOKING SYSTEM



SUBMITTED BY:

NAME	Pragathya .B
SRN	PES1UG20CS286
SEMESTER	V
SECTION	E

Short Description and Scope of the Project

Open source Online Movie Ticket Booking Script (Theatre booking system) is a website to provide the customers facility to book tickets for a movie online and to gather information about the movies and theaters. It uses the ACID, CRUD properties.

The project objective is to book cinema tickets in online. The Ticket Reservation System is an Internet based application that can be accessed throughout the Net and can be accessed by anyone who has a net connection. This application will reserve the tickets and store it in the database. This online ticket booking system provides a website where any user of internet can access it. This project provides complete information regarding currently running movies on all the screens with details of show timings, available seats. Users are required to login and select their preference of movie, theatre, show and book their ticket.

ER Diagram

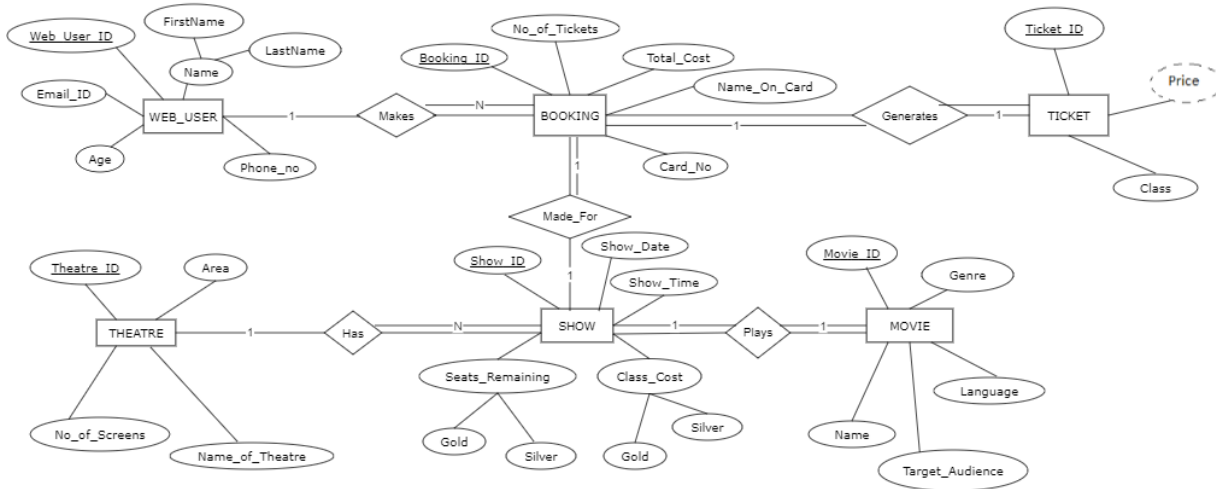


Figure 1: ER Diagram

Relational Schema

WEB USER

Web-User-ID	First-Name/Last-Name	Email-ID	Age	Phone.No
-------------	----------------------	----------	-----	----------

MOVIE

Movie-ID	Name	Language	Genre	Target-Audience
----------	------	----------	-------	-----------------

THEATRE

Theatre-ID	Name-Of-Theatre	NO-of-Screens	Area	M-ID
------------	-----------------	---------------	------	------

SHOW

Show-ID	Show-Date	Show-Time	Seats-Remaining-Gold	Class-Cost-Gold
			Seats-Remaining-Silver	Class-Cost-Silver
			T-ID	M-ID

BOOKING

Booking-ID	Name-on-card	Card-no.	NO-of-Tickets	Total-Cost	U-ID	S-ID
------------	--------------	----------	---------------	------------	------	------

TICKET

Ticket-ID	B-ID	class	Price
-----------	------	-------	-------

Figure 2: Schema Diagram

DDL statements - Building the database:

```
1 • CREATE Table Web_user(  
2   Web_User_ID varchar(5),  
3   First_Name varchar(15),  
4   Last_Name varchar(20),  
5   Email_ID varchar(30),  
6   Age int,  
7   Phone_Number varchar(10) NOT NULL,  
8   Primary Key(Web_User_ID));
```

Table 1 Creating Table "Web_User"

```
10 • Create Table Movie(  
11   Movie_ID varchar(5),  
12   Name varchar(30) NOT NULL,  
13   Language varchar(10),  
14   Genre varchar(20),  
15   Target_Audience varchar(5),  
16   Primary Key(Movie_ID));
```

Table 2 Creating Table "Movie"

```
18 • Create Table Theatre(  
19   Theatre_ID varchar(5),  
20   Name_of_Theatre varchar(30) NOT NULL,  
21   No_of_Screens int,  
22   Area varchar(30),  
23   M_ID varchar(10),  
24   Foreign Key (M_ID) REFERENCES Movie (Movie_ID) ON DELETE CASCADE ON UPDATE CASCADE,  
25   Primary Key(Theatre_ID));
```

Table 3 Creating Table "Theatre"

```

27 • CREATE Table Shows(
28     Show_ID varchar(10),
29     Show_Time time NOT NULL,
30     Show_Date date NOT NULL,
31     Seats_Remaining_Gold int NOT NULL CHECK (Seats_Remaining_Gold >= 0),
32     Seats_Remaining_Silver int NOT NULL CHECK (Seats_Remaining_Silver >= 0),
33     Class_Cost_Gold int NOT NULL,
34     Class_Cost_Silver int NOT NULL,
35     M_ID varchar(5) NOT NULL,
36     T_ID varchar(5) NOT NULL,
37     Foreign Key (M_ID) REFERENCES Movie(Movie_ID) ON DELETE CASCADE ON UPDATE CASCADE,
38     Foreign Key (T_ID) REFERENCES Theatre(Theatre_ID) ON DELETE CASCADE ON UPDATE CASCADE,
39     Primary Key(Show_ID));

```

Table 4 Creating Table "Shows"

```

41 • CREATE Table Booking(
42     Booking_ID varchar(10),
43     No_of_Tickets int NOT NULL,
44     Total_Cost int NOT NULL,
45     Card_Number varchar(19),
46     Name_on_card varchar(21),
47     U_ID varchar(5),
48     S_ID varchar(10),
49     Foreign Key (U_ID) REFERENCES Web_User (Web_User_ID) ON DELETE CASCADE ON UPDATE CASCADE,
50     Foreign Key (S_ID) REFERENCES Shows(Show_ID) ON DELETE CASCADE ON UPDATE CASCADE,
51     Primary Key(Booking_ID));

```

Table 5 Creating Table "Booking"

```

53 • CREATE Table Ticket(
54     Ticket_ID varchar(20),
55     B_ID varchar(10),
56     Class varchar(3) NOT NULL,
57     Price int NOT NULL,
58     Foreign Key(B_ID) REFERENCES Booking(Booking_ID) ON DELETE CASCADE,
59     Primary Key(Ticket_ID));

```

Table 6 Creating Table "Ticket"

OUTPUT

Output				
Action Output				
#	Time	Action	Message	Duration / Fetch
✓ 1	22:29:18	CREATE Table Web_user(Web_User_ID varchar(5), First_Name varchar(15), Last_Name varc...	0 row(s) affected	0.031 sec
✓ 2	22:29:18	Create Table Movie(Movie_ID varchar(5), Name varchar(30) NOT NULL, Language varchar(10),...	0 row(s) affected	0.016 sec
✓ 3	22:29:18	Create Table Theatre(Theatre_ID varchar(5), Name_of_Theatre varchar(30) NOT NULL, No_of...	0 row(s) affected	0.031 sec
✓ 4	22:29:18	CREATE Table Shows(Show_ID varchar(10), Show_Time time NOT NULL, Show_Date date N...	0 row(s) affected	0.031 sec
✓ 5	22:29:18	CREATE Table Booking(Booking_ID varchar(10), No_of_Tickets int NOT NULL, Total_Cost int ...	0 row(s) affected	0.032 sec
✓ 6	22:29:18	CREATE Table Ticket(Ticket_ID varchar(20), B_ID varchar(10), Class varchar(3) NOT NULL, Pr...	0 row(s) affected	0.031 sec

Figure 3 Output of the tables created

Populating the database:

```

2 • INSERT INTO web_user VALUES(
3   ('U01', 'Tara', 'Frank', 'tr@gmail.com', 19, 9876543210),
4   ('U02', 'Lily', 'Mars', 'lm@yahoo.com' , 30, 6543217890),
5   ('U03', 'Shawn', 'Murphy', 'sm@outlook.com' , 24 , 5647382910),
6   ('U04', 'Robert', 'Goldberg', 'rgr@gmail.com' ,38, 8763450912),
7   ('U05', 'Alex', 'Smith', 'als@yahoo.com', 50, 7654000123),
8   ('U06', 'Komal', 'Agarwal', 'komal.agarwal87@gmail.com', 41, 9345687654),
9   ('U07', 'Rahul', 'Raghav', 'rahulrag@gmail.com', 23, 9591990037),
10  ('U08', 'Janice', 'Fernandes', 'janicefernandes@gmail.com', 38, 9325880785),
11  ('U09', 'Nusaiba', 'Mehrunisa', 'nusaibafatima@gmail.com', 24, 8731990037),
12  ('U10', 'Mona', 'Mohammed', 'monaraisa@gmail.com', 23, 9591670037),
13  ('U11', 'Ann', 'Joseph', 'aj@gmail.com', 21, 6574389012));

```

Figure 4 Populating the “web_user” table

	Web_User_ID	First_Name	Last_Name	Email_ID	age	Phone_Number
▶	U01	Tara	Frank	tr@gmail.com	19	9876543210
	U02	Lily	Mars	lm@yahoo.com	30	6543217890
	U03	Shawn	Murphy	sm@outlook.com	24	5647382910
	U04	Robert	Goldberg	rgr@gmail.com	38	8763450912
	U05	Alex	Smith	as@yahoo.com	50	7654000123
	U06	Komal	Agarwal	komal.agarwal87@gmail.com	41	9345687654
	U07	Rahul	Raghav	rahulrag@gmail.com	23	9591990037
	U08	Janice	Fernandes	janicefernandes@gmail.com	38	9325880785
	U09	Nusaiba	Mehrunisa	nusaibafatima@gmail.com	24	8731990037
	U10	Mona	Mohammed	monaraisa@gmail.com	23	9591670037
	U11	Ann	Joseph	aj@gmail.com	21	6574389012

Table 7 Inserted values in the “web_user” table

```

15 • Insert into Movie values(
16  ('001', 'Hichki', 'Hindi', 'Drama/Comedy', 'U/A'),
17  ('002', 'Pacific Rim Uprising', 'English', 'Fantasy/SciFi', 'U/A'),
18  ('003', 'Strangers : Prey at night', 'English', 'Horror', 'U/A'),
19  ('004', 'Tomb Raider', 'English', 'Fantasy/Action', 'A'),
20  ('005', 'Black Panther', 'English', 'Fantasy/SciFi', 'U/A'),
21  ('006', 'Peter Rabbit', 'English', 'Fantasy/Adventure', 'U/A'),
22  ('007', 'Maze Runner: The Death Cure', 'English', 'Fantasy/SciFi', 'U/A'));

```

Figure 5 Populating the “Movie” table

	Movie_ID	Name	Language	Genre	Target_Audience
▶	001	Hichki	Hindi	Drama/Comedy	U/A
	002	Pacific Rim Uprising	English	Fantasy/SciFi	U/A
	003	Strangers : Prey at night	English	Horror	U/A
	004	Tomb Raider	English	Fantasy/Action	A
	005	Black Panther	English	Fantasy/SciFi	U/A
	006	Peter Rabbit	English	Fantasy/Adventure	U/A
	007	Maze Runner: The Death Cure	English	Fantasy/SciFi	U/A
★	NULL	NULL	NULL	NULL	NULL

Table 8 Inserted values in the "Movie" table

```

24 • ☯ Insert into Theatre values(
25     ('T01', 'PVR Cinemas', 4, 'Koramangala, Bangalore'),
26     ('T02', 'INOX Movies', 4, 'Malleshwaram, Bangalore'),
27     ('T03', 'Cinepolis', 3, 'Malleshwaram, Bangalore'),
28     ('T04', 'INOX Movies', 2, 'Rajajinagar, Bangalore'),
29     ('T05', 'PVR Cinemas', 1, 'Indranagar, Bangalore'),
30     ('T06', 'INOX Movies', 2, 'Rajajinagar, Bangalore'),
31     ('T07', 'Cinepolis', 1, ' Whitefield, Bangalore'),
32     ('T08', 'PVR Cinemas', 1, 'Indranagar, Bangalore'));

```

Figure 6 Populating the "Theatre" table

	Theatre_ID	Name_of_Theatre	No_of_Screens	Area	M_ID
▶	T01	PVR Cinemas	4	Koramangala, Bangalore	001
	T02	INOX Movies	4	Malleshwaram, Bangalore	002
	T03	Cinepolis	3	Malleshwaram, Bangalore	003
	T04	INOX Movies	2	Rajajinagar, Bangalore	001
	T05	PVR Cinemas	1	Indranagar, Bangalore	002
	T06	INOX Movies	2	Rajaji Nagar, Bangalore	003
	T07	Cinepolis	1	Whitefield, Bangalore	004
	T08	PVR Cinemas	1	Indranagar, Bangalore	005
★	NULL	NULL	NULL	NULL	NULL

Table 9 Inserted values in the "Theatre" table

```

34 • Insert into shows values(
35     ('S001T01', '09:00:00', '2022-11-28', 20, 60, 400, 350, 'T01', '001'),
36     ('S001T04', '15:00:00', '2022-11-28', 20, 60, 400, 350, 'T04', '001'),
37     ('S002T02', '10:00:00', '2022-11-28', 20, 60, 400, 350, 'T02', '002'),
38     ('S002T05', '19:00:00', '2022-11-28', 20, 60, 400, 350, 'T05', '002'),
39     ('S003T03', '06:00:00', '2022-11-28', 22, 64, 395, 325, 'T03', '003'),
40     ('S003T06', '22:00:00', '2022-11-28', 22, 64, 395, 325, 'T06', '003'),
41     ('S004T07', '10:00:00', '2022-11-28', 20, 60, 400, 350, 'T07', '004'),
42     ('S005T08', '13:00:00', '2022-11-28', 20, 60, 400, 350, 'T08', '005'),
43     ('S101T04', '10:00:00', '2022-11-28', 10, 14, 300, 250, 'T04', '001'),
44     ('S202T05', '07:00:00', '2022-11-28', 10, 14, 300, 250, 'T05', '002'),
45     ('S212T05', '15:00:00', '2022-11-28', 4, 9, 250, 200, 'T05', '002'));

```

Figure 7 Populating the “shows” table

	Show_ID	Show_Time	Show_Date	Seats_Remaining_Gold	Seats_Remaining_Silver	Class_Cost_Gold	Class_Cost_Silver	T_ID	M_ID
▶	S001T01	09:00:00	2022-11-28	20	60	400	350	T01	001
	S001T04	15:00:00	2022-11-28	20	60	400	350	T04	001
	S002T02	10:00:00	2022-11-28	20	60	400	350	T02	002
	S002T05	19:00:00	2022-11-28	20	60	400	350	T05	002
	S003T03	06:00:00	2022-11-28	22	64	395	325	T03	003
	S003T06	22:00:00	2022-11-28	22	64	395	325	T06	003
	S004T07	10:00:00	2022-11-28	20	60	400	350	T07	004
	S005T08	13:00:00	2022-11-28	20	60	400	350	T08	005
	S101T04	10:00:00	2022-11-28	10	14	300	250	T04	001
	S202T05	07:00:00	2022-11-28	10	14	300	250	T05	002
★	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Table 10 Inserted values in the “shows” table

```

47 • Insert into booking values(
48     ('B01', 1, 400, '45654', 'Tara', 'U01', 'S001T01'),
49     ('B02', 1, 325, '45654', 'Tara', 'U01', 'S003T06'),
50     ('B03', 2, 800, '31654', 'Shawn', 'U03', 'S004T07'),
51     ('B04', 2, 500, '99154', 'Alex', 'U05', 'S202T05'),
52     ('B05', 1, 300, '31112', 'Rahul', 'U07', 'S202T05'),
53     ('B06', 2, 800, '54112', 'Mona', 'U10', 'S001T01'),
54     ('B07', 8, 2400, '54112', 'Mona', 'U10', 'S101T04'),
55     ('B08', 4, 1600, '54112', 'Mona', 'U10', 'S005T08'));

```

Figure 8 Populating the “booking” table

	Booking_ID	No_of_Tickets	Total_Cost	Card_Number	Name_on_card	U_ID	S_ID
▶	B01	1	400	45654	Tara	U01	S001T01
	B02	1	325	45654	Tara	U01	S003T06
	B03	2	800	31654	Shawn	U03	S004T07
	B04	2	500	99154	Alex	U05	S202T05
	B05	1	300	31112	Rahul	U07	S202T05
	B06	2	800	54112	Mona	U10	S001T01
	B07	8	2400	54112	Mona	U10	S101T04
	B08	4	1600	54112	Mona	U10	S005T08
✱	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Table 11 Inserted values in the “booking” table

```

57 • ○ insert into ticket values(
58     ('TB1','B01','Gold',400),
59     ('TB2','B02','Gold',325),
60     ('TB3','B03','Gold',800),
61     ('TB4','B04','Silver',500),
62     ('TB5','B05','Gold',300),
63     ('TB6','B06','Gold',800),
64     ('TB7','B07','Silver',2400),
65     ('TB8','B08','Gold',1600));

```

Figure 9 Populating the “ticket” table

	Ticket_ID	B_ID	class	Price
▶	TB1	B01	Gold	400
	TB2	B02	Gold	325
	TB3	B03	Gold	800
	TB4	B04	Silver	500
	TB5	B05	Gold	300
	TB6	B06	Gold	800
	TB7	B07	Silver	2400
	TB8	B08	Gold	1600
✱	NULL	NULL	NULL	NULL

Table 12 Inserted values in the “ticket” table

Join Queries:

1. Natural join to retrieve the theater details in which the movie 'Pacific Rim Uprising' is running'.

```
mysql> select Theatre_ID,Name_of_Theatre,No_of_Screens
-> from Theatre JOIN Movie ON Movie_ID = M_ID
-> where Name='Pacific Rim Uprising';
```

Theatre_ID	Name_of_Theatre	No_of_Screens
T02	INOX Movies	4
T05	PVR Cinemas	1

2 rows in set (0.00 sec)

Table 13 Natural Join

2. Left join returns all customers and only bookings owed by customers.

```
mysql> SELECT COUNT(DISTINCT(w.Web_User_ID)), COUNT(b.U_ID) FROM web_user w LEFT JOIN booking b ON w.Web_User_ID = b.U_ID;
```

COUNT(DISTINCT(w.Web_User_ID))	COUNT(b.U_ID)
11	8

1 row in set (0.00 sec)

Table 14 Left Join

3. Right join returns only customers having bookings and all bookings.

```
mysql> SELECT COUNT(DISTINCT(w.Web_User_ID)) FROM web_user w RIGHT JOIN booking b ON w.Web_User_ID = b.U_ID;
```

COUNT(DISTINCT(w.Web_User_ID))
5

1 row in set (0.00 sec)

Table 15 Right Join

4. Cross join returns list of id's of different theatre's where the movie is running.

```
mysql> SELECT Name AS 'Film', Theatre_ID AS 'Theatre' FROM Movie CROSS JOIN Theater on Movie_ID = M_ID;
```

Film	Theatre
Hichki	T01
Hichki	T04
Pacific Rim Uprising	T02
Pacific Rim Uprising	T05
Strangers : Prey at night	T03
Strangers : Prey at night	T06
Tomb Raider	T07
Black Panther	T08

8 rows in set (0.00 sec)

Table 16 Cross Join

Aggregate Functions:

1. Display the total cost of various users who spent more the 400 rupees on booking tickets.

```
mysql> Select U_ID,sum(Total_Cost) from booking group by U_ID having sum(Total_Cost)>400;
+-----+-----+
| U_ID | sum(Total_Cost) |
+-----+-----+
| U01 |          725 |
| U03 |          800 |
| U05 |          500 |
| U10 |          800 |
+-----+-----+
4 rows in set (0.00 sec)
```

Table 17 Sum function

2. Display the Movie ID which Is allotted minimum number of silver seats.

```
mysql> select Movie_ID,min(Seats_Remaining_Silver) from shows;
+-----+-----+
| Movie_ID | min(Seats_Remaining_Silver) |
+-----+-----+
| 001 | 14 |
+-----+-----+
1 row in set (0.00 sec)
```

Table 18 Min function

3. Display the number of times a user booked a ticket.

```
mysql> SELECT COUNT(DISTINCT(U_ID)) FROM booking;
+-----+
| COUNT(DISTINCT(U_ID)) |
+-----+
| 5 |
+-----+
1 row in set (0.00 sec)
```

Table 19 Count function

4. Display the average number of screens allotted per movie in a theatre.

```
mysql> select Theatre_ID,M_ID, avg(No_of_Screens) from theater group by M_ID;
+-----+-----+-----+
| Theatre_ID | M_ID | avg(No_of_Screens) |
+-----+-----+-----+
| T01 | 001 | 3.0000 |
| T02 | 002 | 2.5000 |
| T03 | 003 | 2.5000 |
| T07 | 004 | 1.0000 |
| T08 | 005 | 1.0000 |
+-----+-----+-----+
5 rows in set (0.00 sec)
```

Table 20 Average function

Set Operations:

1. The following SQL statement returns the Movie_ID='002' from both the "movie" and the "Theater" table.

```
mysql> SELECT Movie_ID FROM movie
-> WHERE Movie_ID='002'
-> UNION ALL
-> SELECT M_ID FROM Theater
-> WHERE M_ID='002'
-> ORDER BY Movie_ID;
+-----+
| Movie_ID |
+-----+
| 002      |
| 002      |
| 002      |
+-----+
3 rows in set (0.00 sec)
```

Table 21 Union ALL Operation

2. Display the Users who haven't booked a ticket for a movie yet.
(MINUS OPEATOR)

```
mysql> select Web_User_ID, First_Name from web_user
-> where Web_User_ID
-> NOT IN
-> (select U_ID from booking);
+-----+-----+
| Web_User_ID | First_Name |
+-----+-----+
| U02         | Lily       |
| U04         | Robert    |
| U06         | Komal     |
| U08         | Janice    |
| U09         | Nusaiba   |
| U11         | Ann       |
| U13         | Neeta     |
| U14         | Jeniffer  |
+-----+-----+
8 rows in set (0.00 sec)
```

Table 22 Minus Operation

3. Display the movies running in a theatre out of total movies in the 'Movie' table.(INTERSECT OPERATOR)

```
mysql> select Show_ID,Movie_ID from shows where Show_ID IN (select S1_ID from booking);
+-----+-----+
| Show_ID | Movie_ID |
+-----+-----+
| S001T01 | 001      |
| S003T06 | 003      |
| S004T07 | 004      |
| S005T08 | 005      |
| S101T04 | 001      |
| S202T05 | 002      |
+-----+-----+
6 rows in set (0.00 sec)
```

Table 23 Intersection operation

4. Display the list of theaters in the Theater and the Show table.

```
mysql> select Theatre_ID from theater union select T_ID from shows;
+-----+
| Theatre_ID |
+-----+
| T01        |
| T04        |
| T02        |
| T05        |
| T03        |
| T06        |
| T07        |
| T08        |
+-----+
8 rows in set (0.00 sec)
```

Table 24 Union operation

Functions and Procedures:

1.Functions:

This functions checks and outputs if a user can purchase a ticket or not. If a user books more than 10 tickets in the same month then it displays that ="Your Limit is limit is over. Retry Post One Month"
Else displays "Can Purchase ticket"

```

1 • USE movie_booking_tool;
2 DELIMITER $$
3 • CREATE FUNCTION count_booking(ticket int)
4 RETURNS VARCHAR(50)
5 DETERMINISTIC
6 BEGIN
7 DECLARE VALUE varchar(50);
8 IF ticket > 10 then
9 set VALUE="Your Limit is limit is over. Retry Post One Month";
10 ELSE
11 set VALUE ="Can Purchase ticket";
12 end if;
13 return value;
14 END$$
15 DELIMITER ;
16
17 • with t as (Select U_ID,sum(No_of_Tickets) as count from booking group by U_ID )
18 select U_ID, count_booking(count) as Validate,count as ticket_purchased from t;

```

Figure 10 Function to display the limit of tickets that can be purchased by a user

OUTPUT

Result Grid Filter Rows: Export: Wrap Cell Content:			
	U_ID	Validate	ticket_purchased
▶	U01	Can Purchase ticket	2
	U03	Can Purchase ticket	2
	U05	Can Purchase ticket	2
	U07	Can Purchase ticket	1
	U10	Your Limit is limit is over. Retry Post One Month	14

1 18:15:30 with t as (Select U_ID,sum(No_of_Tickets) as count from booking group by U_ID)... 5 row(s) returned

Table 25 Shows if a user can purchase a ticket or not

2.Procedures:

This Procedure removes the movies from the movie table whose dates are past the current dates.

```

1 • USE movie_booking_tool;
2 • SET FOREIGN_KEY_CHECKS=0;
3   DELIMITER //
4 • CREATE PROCEDURE delete_old_date()
5   BEGIN
6     declare curdate date;
7     set curdate=curdate();
8     DELETE FROM shows
9     WHERE datediff>Show_Date,curdate)<0;
10  END //
11  DELIMITER ;
12 • select * from shows;
13 • CALL delete_old_date;
14 • select * from shows;

```

Figure 11 Procedure to remove the shows if date of the show is greater than the current date

OUTPUT

✓	4	21:21:38	select * from shows LIMIT 0, 1000	11 row(s) returned	0.000 sec / 0.000 sec
✓	5	21:21:38	CALL delete_old_date	1 row(s) affected	0.000 sec
✓	6	21:21:38	select * from shows LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec

Figure 12 Output Execution

	Show_ID	Show_Time	Show_Date	Seats_Remaining_Gold	Seats_Remaining_Silver	Class_Cost_Gold	Class_Cost_Silver	T_ID	Movie_ID
▶	S001T01	09:00:00	2022-11-28	20	60	400	350	T01	001
	S001T04	15:00:00	2022-11-28	20	60	400	350	T04	001
	S002T02	10:00:00	2022-11-28	20	60	400	350	T02	002
	S002T05	19:00:00	2022-11-28	20	60	400	350	T05	002
	S003T03	06:00:00	2022-11-28	22	64	395	325	T03	003
	S003T06	22:00:00	2022-11-28	22	64	395	325	T06	003
	S004T07	10:00:00	2022-11-28	20	60	400	350	T07	004
	S005T08	13:00:00	2022-11-28	20	60	400	350	T08	005
	S101T04	10:00:00	2022-11-28	10	14	300	250	T04	001
	S202T05	07:00:00	2022-11-28	10	14	300	250	T05	002
	S212T05	15:00:00	2022-11-11	4	9	250	200	T05	002
•	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Table 26 "Shows" table before calling the procedure (observe the last row Show_Date > currdate)

	Show_ID	Show_Time	Show_Date	Seats_Remaining_Gold	Seats_Remaining_Silver	Class_Cost_Gold	Class_Cost_Silver	T_ID	Movie_ID
▶	S001T01	09:00:00	2022-11-28	20	60	400	350	T01	001
	S001T04	15:00:00	2022-11-28	20	60	400	350	T04	001
	S002T02	10:00:00	2022-11-28	20	60	400	350	T02	002
	S002T05	19:00:00	2022-11-28	20	60	400	350	T05	002
	S003T03	06:00:00	2022-11-28	22	64	395	325	T03	003
	S003T06	22:00:00	2022-11-28	22	64	395	325	T06	003
	S004T07	10:00:00	2022-11-28	20	60	400	350	T07	004
	S005T08	13:00:00	2022-11-28	20	60	400	350	T08	005
	S101T04	10:00:00	2022-11-28	10	14	300	250	T04	001
	S202T05	07:00:00	2022-11-28	10	14	300	250	T05	002
✱	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Table 27 "Shows" table after calling the procedure (last row is removed)

Triggers and cursors:

1.Triggers

The following Trigger outputs whether a user is under-age(age below 14), if under age it doesn't insert the values in the user_table else it inserts the value in the user table

```

1 • USE movie_booking_tool;
2 • SET FOREIGN_KEY_CHECKS=0;
3   DELIMITER $$
4 • CREATE TRIGGER check_age
5   AFTER INSERT
6   ON web_user FOR EACH ROW
7   BEGIN
8     DECLARE error_msg VARCHAR(200);
9     declare count int;
10    SET error_msg = ('Under age');
11    IF (select Age from web_user where Web_User_ID = new.Web_User_ID) < '14' THEN
12      SIGNAL SQLSTATE '45000'
13      SET MESSAGE_TEXT = error_msg;
14    END IF;
15  END $$
16  DELIMITER ;
17 • insert into web_user values ('U12','Mariam','Wilson','mws@gmail.com','11','9999389012');
18 • insert into web_user values ('U13','Neeta','Bose','nbs@gmail.com','55','6374387713');
19 • select * from web_user;

```

Figure 13 Trigger Code to check the user is underage or not after insert

OUTPUT

✖ 4 20:49:15 insert into web_user values ('U12','Mariam','Wilson','mws@gmail.com','11','999938... Error Code: 1644. Under age

Figure 14 Negative case: Doesn't insert 'U12' because age<14

Positive case: Inserts 'U12' because age>14

✔ 5 20:49:19 insert into web_user values ('U13','Neeta','Bose','nbs@gmail.com','55','63743877... 1 row(s) affected

Figure 15 Positive case: Inserts 'U12' because age>14

	Web_User_ID	First_Name	Last_Name	Email_ID	age	Phone_Number
▶	U01	Tara	Frank	tr@gmail.com	19	9876543210
	U02	Lily	Mars	lm@yahoo.com	30	6543217890
	U03	Shawn	Murphy	sm@outlook.com	24	5647382910
	U04	Robert	Goldberg	rgr@gmail.com	38	8763450912
	U05	Alex	Smith	as@yahoo.com	50	7654000123
	U06	Komal	Agarwal	komal.agarwal87@gmail.com	41	9345687654
	U07	Rahul	Raghav	rahulrag@gmail.com	23	9591990037
	U08	Janice	Fernandes	janicefernandes@gmail.com	38	9325880785
	U09	Nusaiba	Mehrunisa	nusaibafatima@gmail.com	24	8731990037
	U10	Mona	Mohammed	monaraisa@gmail.com	23	9591670037
	U11	Ann	Joseph	aj@gmail.com	21	6574389012
	U13	Neeta	Bose	nbs@gmail.com	55	6374387713
*	NULL	NULL	NULL	NULL	NULL	NULL

Table 28 Web_user table where 'U12' is not inserted where 'U13' is inserted.

2.Cursors:

The following cursor retrieves the list of email-id's of users in the web-user table.

```

1  DELIMITER $$
2  • CREATE PROCEDURE createEmailList_ (INOUT emailList varchar(4000))
3  BEGIN
4      DECLARE finished INTEGER DEFAULT 0;
5      DECLARE emailAddress varchar(100) DEFAULT "";
6      DECLARE curEmail
7          CURSOR FOR
8              SELECT Email_ID FROM web_user;
9      DECLARE CONTINUE HANDLER
10         FOR NOT FOUND SET finished = 1;
11     OPEN curEmail;
12     getEmail: LOOP
13         FETCH curEmail INTO emailAddress;
14         IF finished = 1 THEN
15             LEAVE getEmail;
16         END IF;
17         SET emailList = CONCAT(emailAddress,";",emailList);
18     END LOOP getEmail;
19     CLOSE curEmail;
20 END$$
21 DELIMITER ;
22 • SET @emailList = "";
23 • CALL createEmailList_(@emailList);
24 • SELECT @emailList;
```

Figure 16 Cursor code to list the email-id's of all users

OUTPUT

@emailList
nbs@gmail.com;aj@gmail.com;monaraisa@gmail.com;nusaibafatima@gmail.com;janicefernandes@gmail.com;rahulrag@gmail.com;komal.agarwal87@gmail.com;as@yahoo.co...

Table 29 List of email-id's of users

✓	14	22:43:53	use movie_booking_tool	0 row(s) affected	0.000 sec
✓	15	22:43:53	CREATE PROCEDURE createEmailList_ (INOUT emailList varchar(4000)) BEGIN DECL...	0 row(s) affected	0.016 sec
✓	16	22:43:59	SET @emailList = ""	0 row(s) affected	0.016 sec
✓	17	22:43:59	CALL createEmailList_(@emailList)	0 row(s) affected	0.000 sec
✓	18	22:43:59	SELECT @emailList LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

Figure 17 Output of the execution step

Developing a Front-end:

1. Addition, Modification and Deletion of records from any chosen table

OVERVIEW OF MY FRONTEND

Figure 18 Insert login details into the web_user table

Figure 19 Select a movie from the movies table

Figure 20 Displays the details of the selected movie

Movie Ticket Booking App

Check for theatres

Select a Theatre

☐ T02

☒ T05

Theater Name :

PVR Cinemas

No of screens Available:

1

Location :

Indiranagar, Bangalore

Figure 21 The corresponding theaters for the selected movie are shown. Select theater of your choice which displays the details of the theaters.

Movie Ticket Booking App

Select Your Show

Select a Show

☐ S002T06

☒ S202T06

Date:

2022-11-28

Show Time :

7:00:00

Remaining Gold Seats :

10

Remaining Silver Seats :

14

Cost of Gold Seat:

300

Cost of silver Seat:

250

Figure 22 Displays "Shows" running in the theater that you selected, Select one of the shows and check its details

Movie Ticket Booking App

Book Your Ticket

Booking ID:

B09

Select

Gold Class

Enter your Card No.:

43875

Ticket count:

2

Enter your name on card:

Jennifer

Bill:

800

Book

Successfully Booked Your Ticket

Figure 23 Book the ticket by choosing Gold/Silver Class, Enter the number of tickets which displays the total cost and book the tickets by clicking the "Book " button.

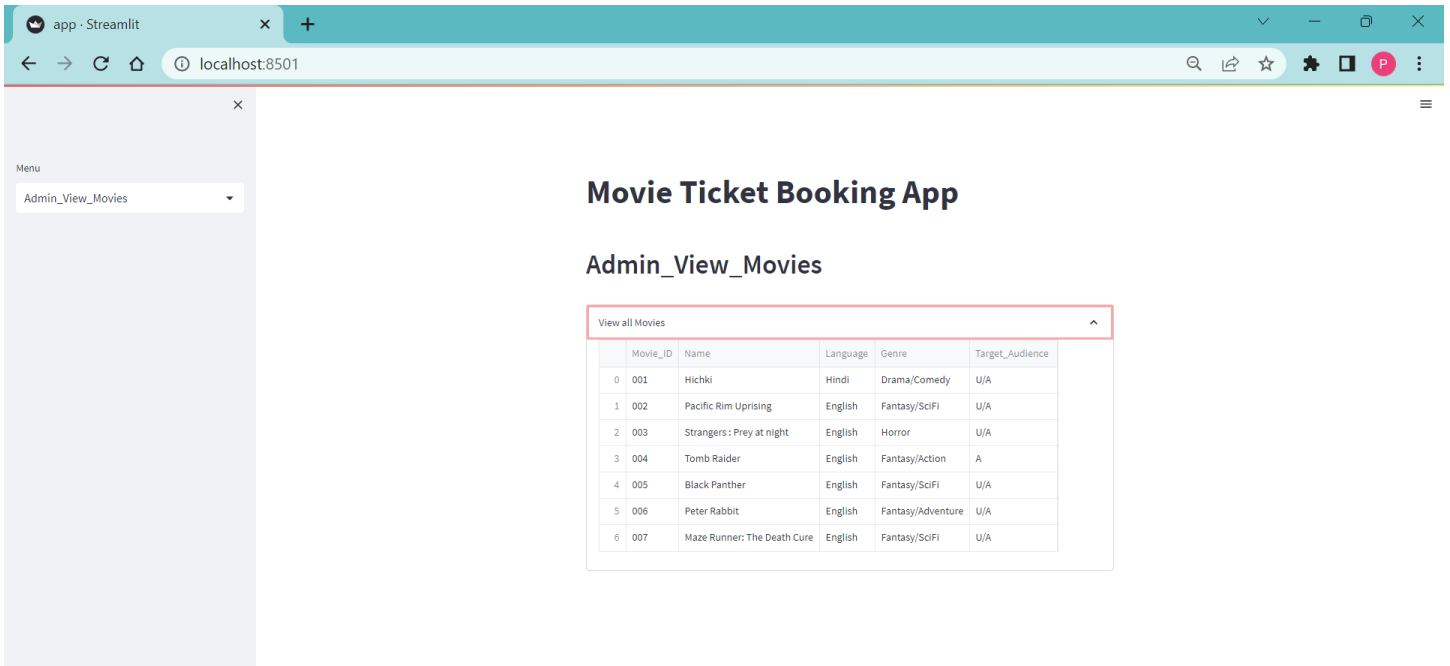


Figure 24 Admin end: Lists the Movies and their details in the database

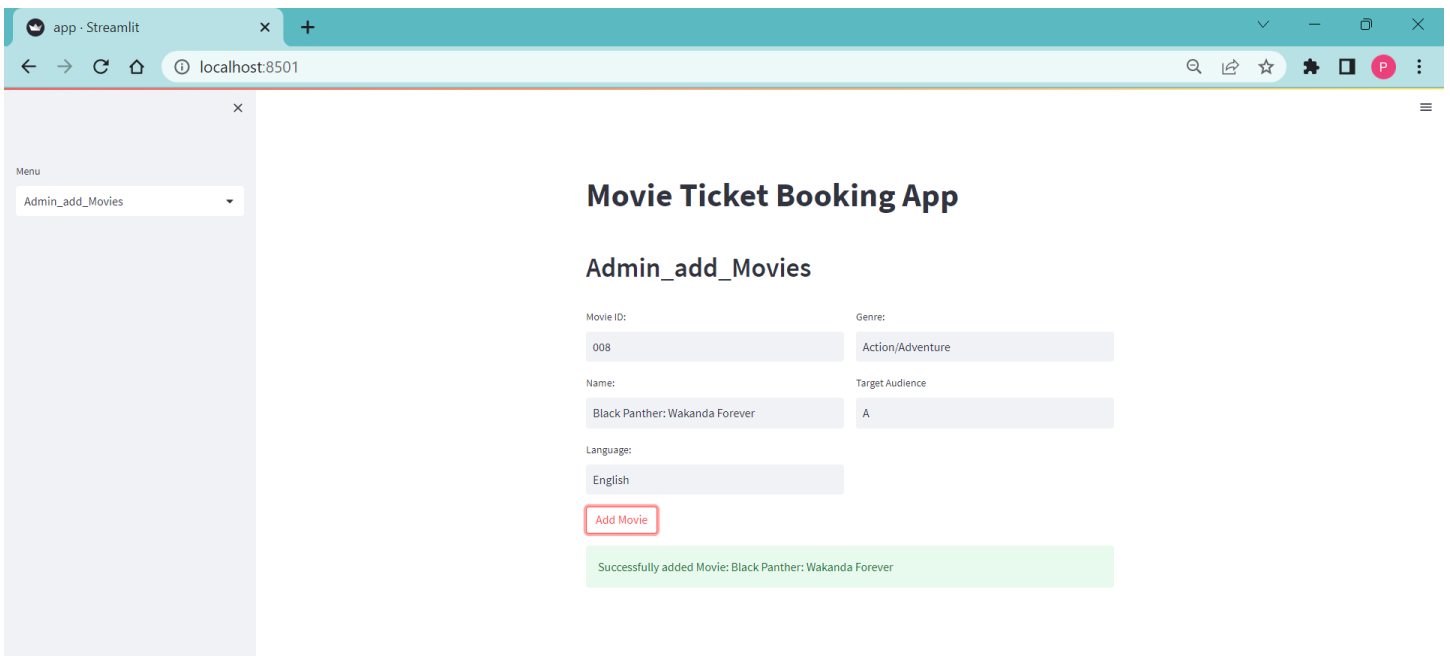
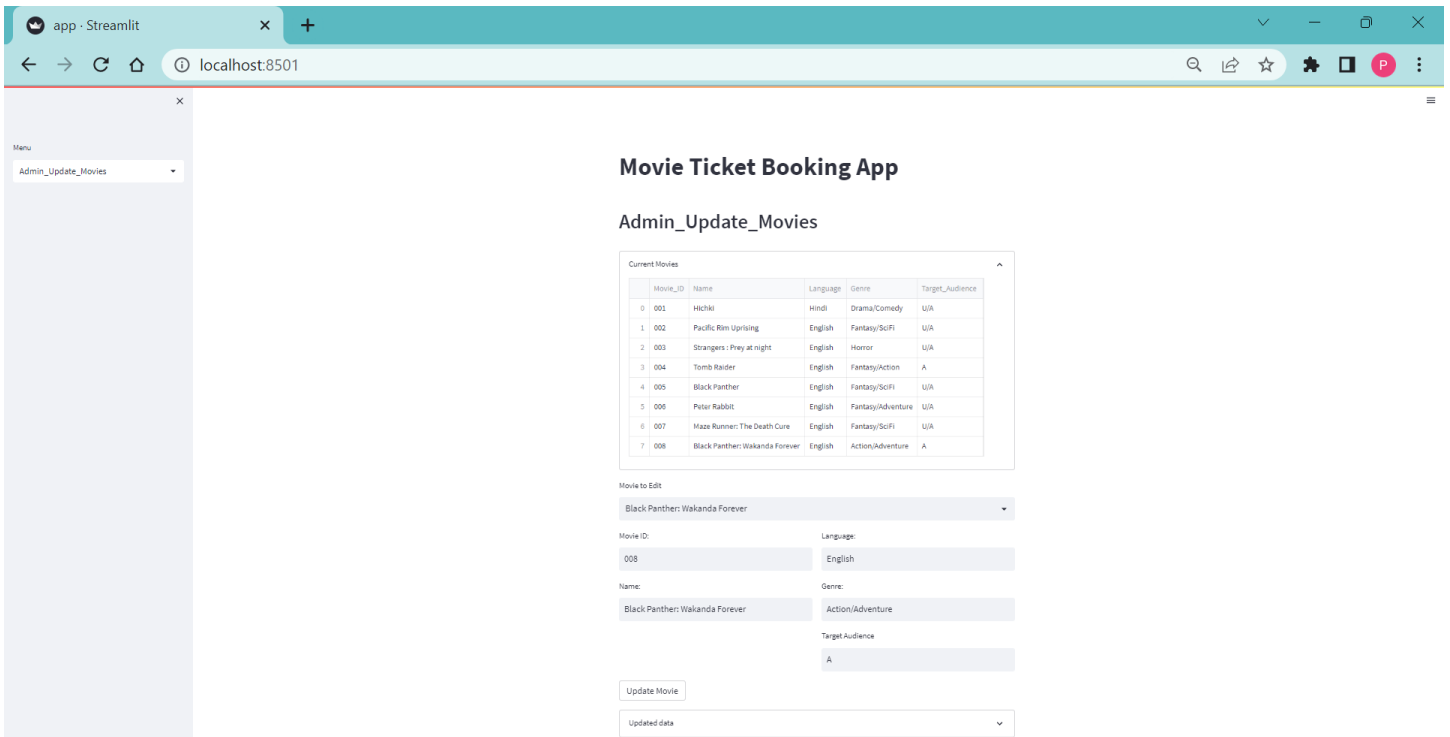


Figure 25 Admin's access to add a movie into the "Movie" Table



Movie Ticket Booking App

Admin_Update_Movies

Current Movies

Movie_ID	Name	Language	Genre	Target_Audience
0 001	Hichki	Hindi	Drama/Comedy	U/A
1 002	Pacific Rim Uprising	English	Fantasy/SciFi	U/A
2 003	Strangers : Prey at night	English	Horror	U/A
3 004	Tomb Raider	English	Fantasy/Action	A
4 005	Black Panther	English	Fantasy/SciFi	U/A
5 006	Peter Rabbit	English	Fantasy/Adventure	U/A
6 007	Maze Runner: The Death Cure	English	Fantasy/SciFi	U/A
7 008	Black Panther: Wakanda Forever	English	Action/Adventure	A

Movie to Edit: Black Panther: Wakanda Forever

Movie ID: 008 Language: English

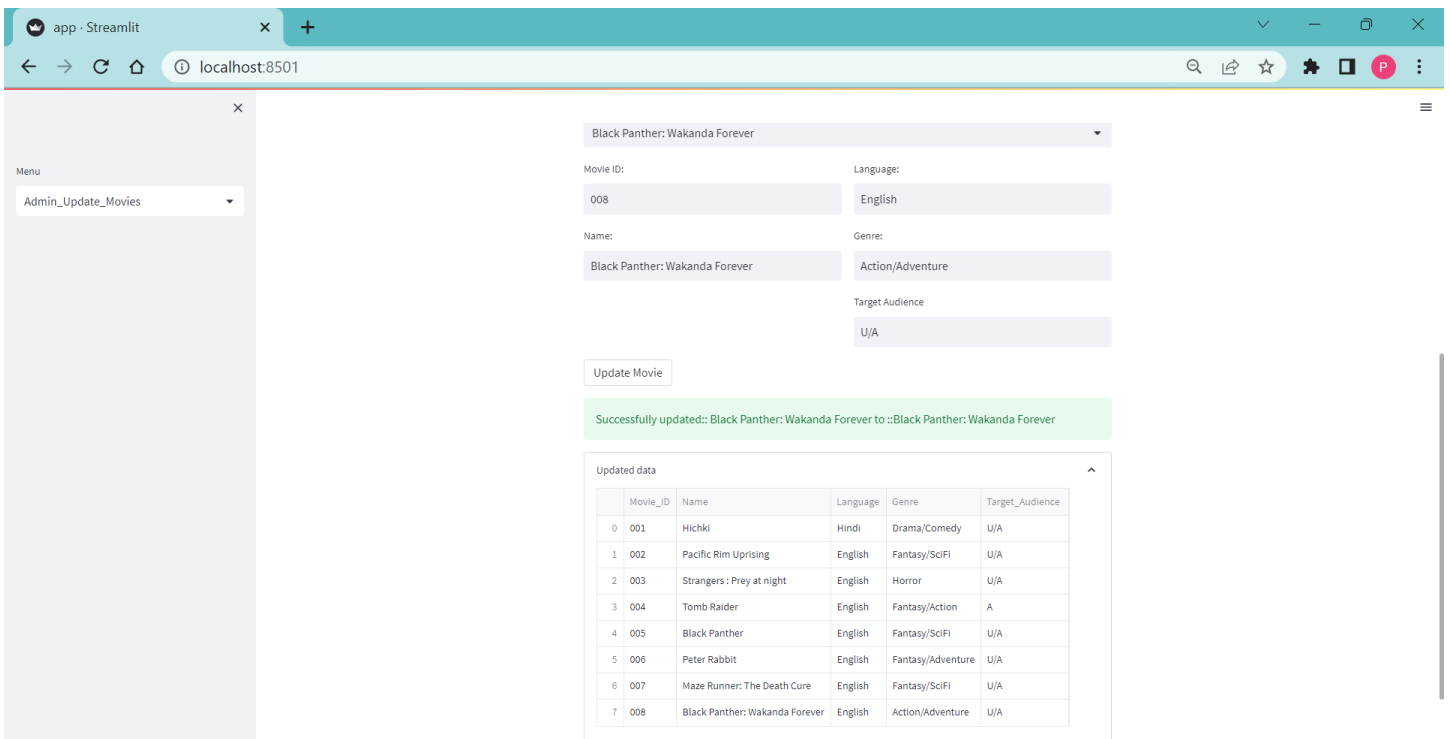
Name: Black Panther: Wakanda Forever Genre: Action/Adventure

Target Audience: A

Update Movie

Updated data

Figure 26 Admin edits the movie table



Movie Ticket Booking App

Admin_Update_Movies

Black Panther: Wakanda Forever

Movie ID: 008 Language: English

Name: Black Panther: Wakanda Forever Genre: Action/Adventure

Target Audience: U/A

Update Movie

Successfully updated: Black Panther: Wakanda Forever to ::Black Panther: Wakanda Forever

Updated data

Movie_ID	Name	Language	Genre	Target_Audience
0 001	Hichki	Hindi	Drama/Comedy	U/A
1 002	Pacific Rim Uprising	English	Fantasy/SciFi	U/A
2 003	Strangers : Prey at night	English	Horror	U/A
3 004	Tomb Raider	English	Fantasy/Action	A
4 005	Black Panther	English	Fantasy/SciFi	U/A
5 006	Peter Rabbit	English	Fantasy/Adventure	U/A
6 007	Maze Runner: The Death Cure	English	Fantasy/SciFi	U/A
7 008	Black Panther: Wakanda Forever	English	Action/Adventure	U/A

Figure 27 Target_Audience edited from A to U/A

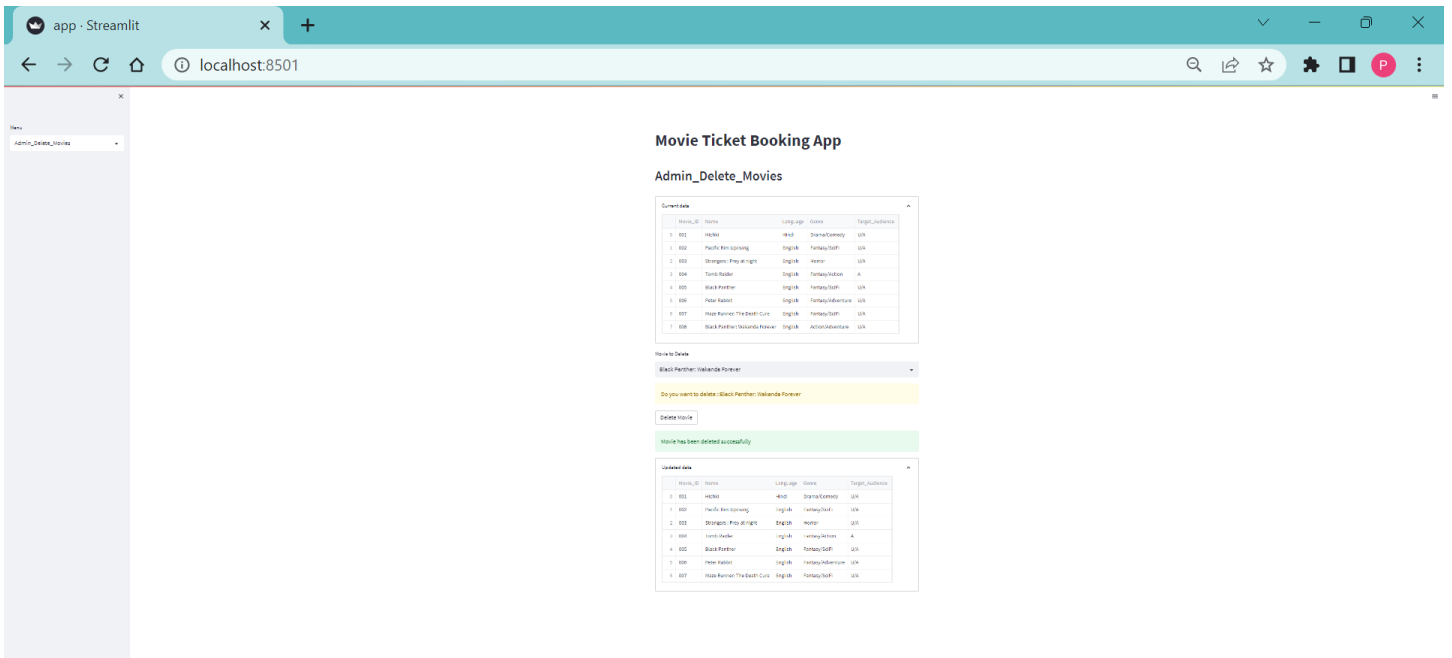


Figure 28 Admin's access to delete a movie from the "Movies" table

2. There should be a window to accept and run any SQL statement and display the result.

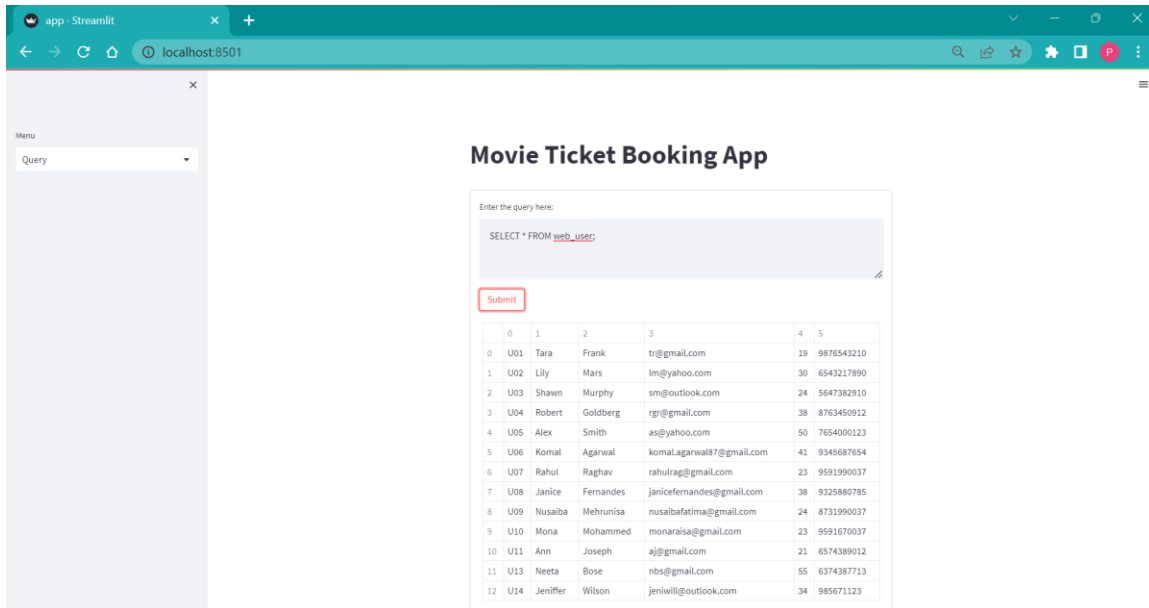


Figure 29 Query tab

