

Project Report on
Online Movie Ticket Booking System

Developed by

**Twieny Korat (IT-123) Department of IT, DD
University**
**Vaishnavi Wadhwa (IT-135) Department of IT, DD
University**

Guided By
Internal Guide:
Mukesh M Goswami
**Department of Information Technology Faculty of
Technology DD University**



**Department of Information Technology Faculty of
Technology, Dharmsinh**

DHARMSINH DESAI UNIVERSITY
NADIAD-387001, GUJARAT



I . CERTIFICATE

This is to certify that the project entitled “**Online Movie Ticket Booking System**” is bonafied report of the work carried out by

1) **Ms. Twiency Korat**, Student ID No : **18ITUON125**

2) **Ms. Vaishnavi Wadhwa**, Student ID No : **18ITUOS083**

of Department of Information Technology, semester V, under the guidance and supervision for the subject Database Management System. They were involved in Project training during academic year 2020-2021.

Prof. Mukesh M Goswami

(Project Guide)

Department of Information Technology,

Faculty of Technology,

Dharmsinh Desai University, Nadiad Date:

Prof. Vipul Dabhi

Head, Department of Information Technology,

Faculty of Technology,

Dharmsinh Desai University, Nadiad Date:

II . ACKNOWLEDGEMENT

We would like to give our sincere acknowledgement to everybody responsible for the successful completion of our project “Online Movie Ticket Booking System”.

The success and final outcome of this project required a lot of guidance and assistance from many people and we are extremely privileged to have got this all along the completion of this project.

We owe our deep gratitude to our project guide Prof. Mukesh M Goswami, who took been interest on our project work and guided us all along till the completion of our project work by providing all the necessary help for developing a good Database System.

We would also like to thank all our lecturers.

Finally, we convey our acknowledgement to all our friends and family members who directly or indirectly associated with us in the successful completion of the project. We thank one and all.

Thanking You,

Twieny Korat (IT-123)

Vaishnavi Wadhwa (IT-135)

TABLE OF CONTENTS

I. Certificate	2
II. Acknowledgement	3
1. SYSTEM OVERVIEW	5
1.1 Current system.....	5
1.2 Objectives of the Proposed System	5
1.3 Advantages of the Proposed system (over current)	5
2. E-R DIAGRAM.....	6
3. DATA DICTIONARY	7
4. SCHEMA DIAGRAM.....	11
5. DATABASE IMPLEMENTATION	12
5.1 Create Schema	12
5.2 Insert Data values	12
5.3 Queries (Based on functions, group by, having, joins, sub query etc.)	21
5.4 PL/SQL Block (Procedures and Exception Handling).....	26
5.5 Function	28
5.6 Triggers	29
5.7 Cursors	32
6. FUTURE ENHANCEMENTS OF THE SYSTEM.....	33
7.BIBLIOGRAPHY.....	34

1. SYSTEM OVERVIEW

1.1 Current system:

Online movie ticket 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 theatres.

This system is basically aimed to provide the customer the complete information of the movie, according to which the customer can book the tickets. Customer needs to register with the details mentioned at the site to book the movie tickets. After selecting the show, the user is presented a seating layout so that he can select seats of choice. Thereafter he is directed to payment gateway for making a transaction. User can update his profile, take a print out of the tickets and also view his booking history.

1.2 Objectives:

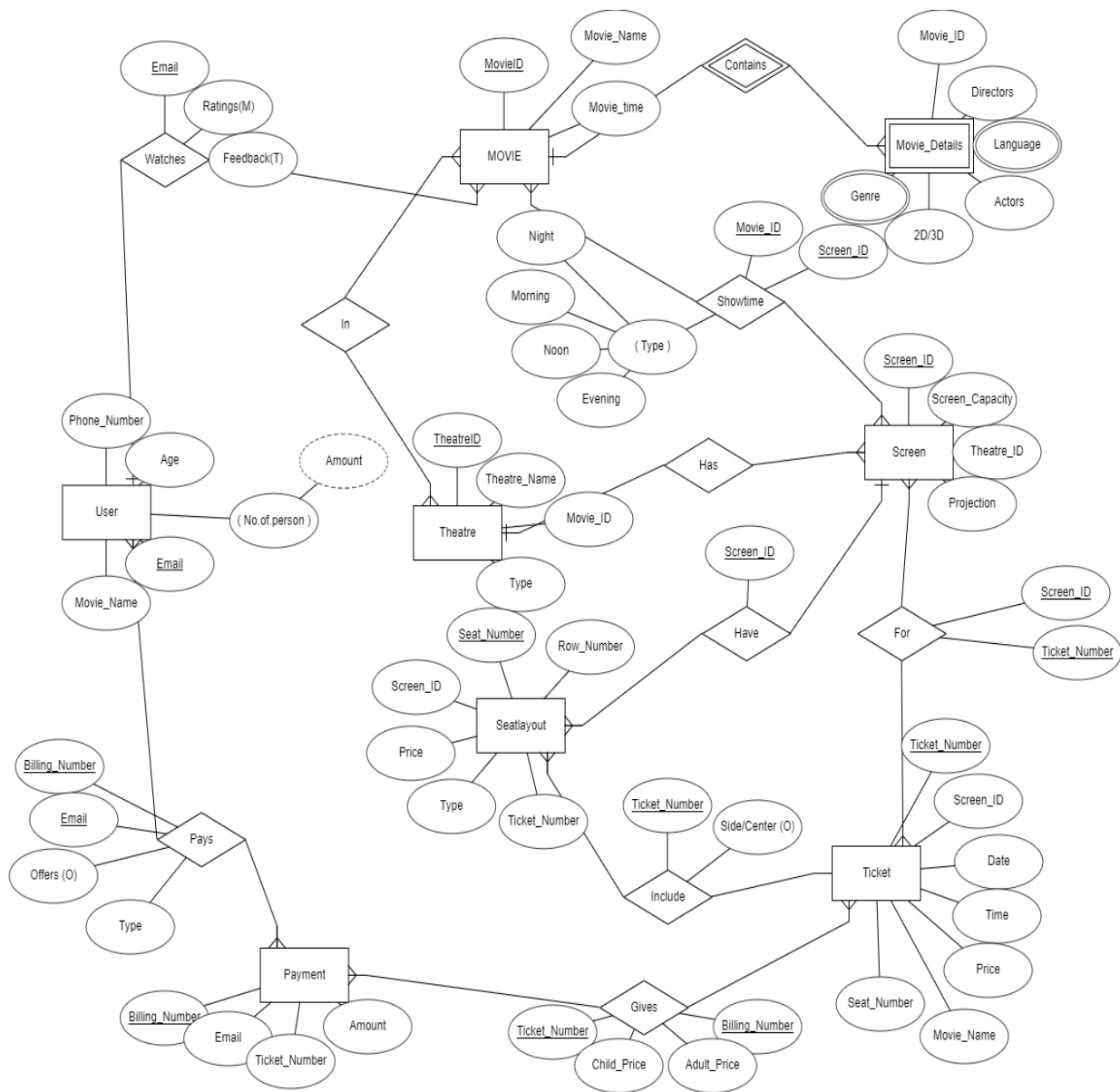
The main objective of our website is to provide the best facility to our customers so that they do not need to stand in the long queues at the box offices for their own choice of seats.

- It is a web-based application which can be accessed from anywhere. Its content is easily understandable by a normal person and doesn't need any elaboration
- This is a web application for buying movie tickets online. Customers can buy tickets 24*7.
- This system is developed keeping in view of current multiplex working pattern. Schedule for many screens can be programmed in this application.

1.3 Advantages:


1. They Work 24/7.
2. Hassle-free Management of bookings.
3. Get smarter insights into your business.
4. The number of no-shows decrease.
5. Payment are easier.
6. Cut your workload.
7. You can maximise reservation.

2. E R MODEL





3. DATA DICTIONARY

THEATRE

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	TheatreID 	int(11)			No	None		
2	Theatre_Name	varchar(20)	utf8mb4_general_ci		No	None		
3	Movie_ID	int(11)			No	None		
4	Type	varchar(20)	utf8mb4_general_ci		No	None		

SCREEN



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	Screen_ID 	int(11)			No	None		
2	Screen_Capacity	int(11)			No	None		
3	Theatre_ID	int(11)			No	None		
4	Projection	varchar(15)	utf8mb4_general_ci		No	None		
5	TheatreID 	int(11)			No	None		

TICKET


#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	Ticket_Number 	int(11)			No	None		
2	Screen_ID	int(11)			No	None		
3	MDate	date			No	None		
4	MTime	timestamp			No	current_timestamp()	ON UPDATE CURRENT_TIMESTAMP()	
5	Movie_Name	varchar(30)	utf8mb4_general_ci		No	None		
6	Price	decimal(5,0)			No	None		
7	Seat_Number	varchar(5)	utf8mb4_general_ci		No	None		

SEATLAYOUT


Online Movie Ticket Booking System

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	Seat_Number 	varchar(5)	utf8mb4_general_ci		No	None		
2	Row_Number	char(2)	utf8mb4_general_ci		No	None		
3	Ticket_Number	int(11)			No	None		
4	Type	varchar(15)	utf8mb4_general_ci		No	None		
5	Price	decimal(5,0)			No	None		
6	Screen_ID 	int(11)			No	None		



PAYMENT

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	Billing_Number 	int(11)			No	None		
2	Email	varchar(30)	utf8mb4_general_ci		No	None		
3	Type	varchar(10)	utf8mb4_general_ci		No	None		
4	Amount	int(11)			No	None		
5	Offers	varchar(10)	utf8mb4_general_ci		No	None		
6	Ticket_Number	int(11)			No	None		



CUSTO

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	Phone_Number	decimal(10,0)			No	None		
2	Age	int(11)			No	None		
3	Amount	int(11)			No	None		
4	Email 	varchar(30)	utf8mb4_general_ci		No	None		
5	Movie_Name	varchar(30)	utf8mb4_general_ci		No	None		
6	Billing_Number	int(11)			No	None		



SCREEN TICKET

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	Screen_ID 	int(11)			No	None		
2	Ticket_Number 	int(11)			No	None		

GIVES

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	Child_Price	decimal(5,0)			No	None		
2	Adult_Price	decimal(5,0)			No	None		
3	Ticket_Number 	int(11)			No	None		
4	Billing_Number 	int(11)			No	None		


USERPAYS

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	Offers	varchar(20)	utf8mb4_general_ci		Yes	NULL		
2	Type	varchar(20)	utf8mb4_general_ci		No	None		
3	Email 	varchar(30)	utf8mb4_general_ci		No	None		
4	Billing_Number 	int(11)			No	None		

MOVIE

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	MovieID 	int(11)			No	None		
2	Movie_Name	varchar(30)	utf8mb4_general_ci		No	None		
3	Movie_time	timestamp			No	current_timestamp()	ON UPDATE CURRENT_TIMESTAMP()	
4	Email 	varchar(30)	utf8mb4_general_ci		No	None		

MOVIE DETAILS

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	Directors	varchar(20)	utf8mb4_general_ci		No	None		
2	Actors	varchar(20)	utf8mb4_general_ci		No	None		
3	Dim	char(2)	utf8mb4_general_ci		No	None		
4	MovieID 	int(11)			No	None		

THEA MOVIE

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	RatingsM	int(11)			No	None		
2	FeedbackT	varchar(30)	utf8mb4_general_ci		No	None		
3	MovieID 🔑	int(11)			No	None		
4	TheatreID 🔑 🔒	int(11)			No	None		

SHOWTIME

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	Type	varchar(10)	utf8mb4_general_ci		No	None		
2	MovieID 🔑	int(11)			No	None		
3	Screen_ID 🔑	int(11)			No	None		

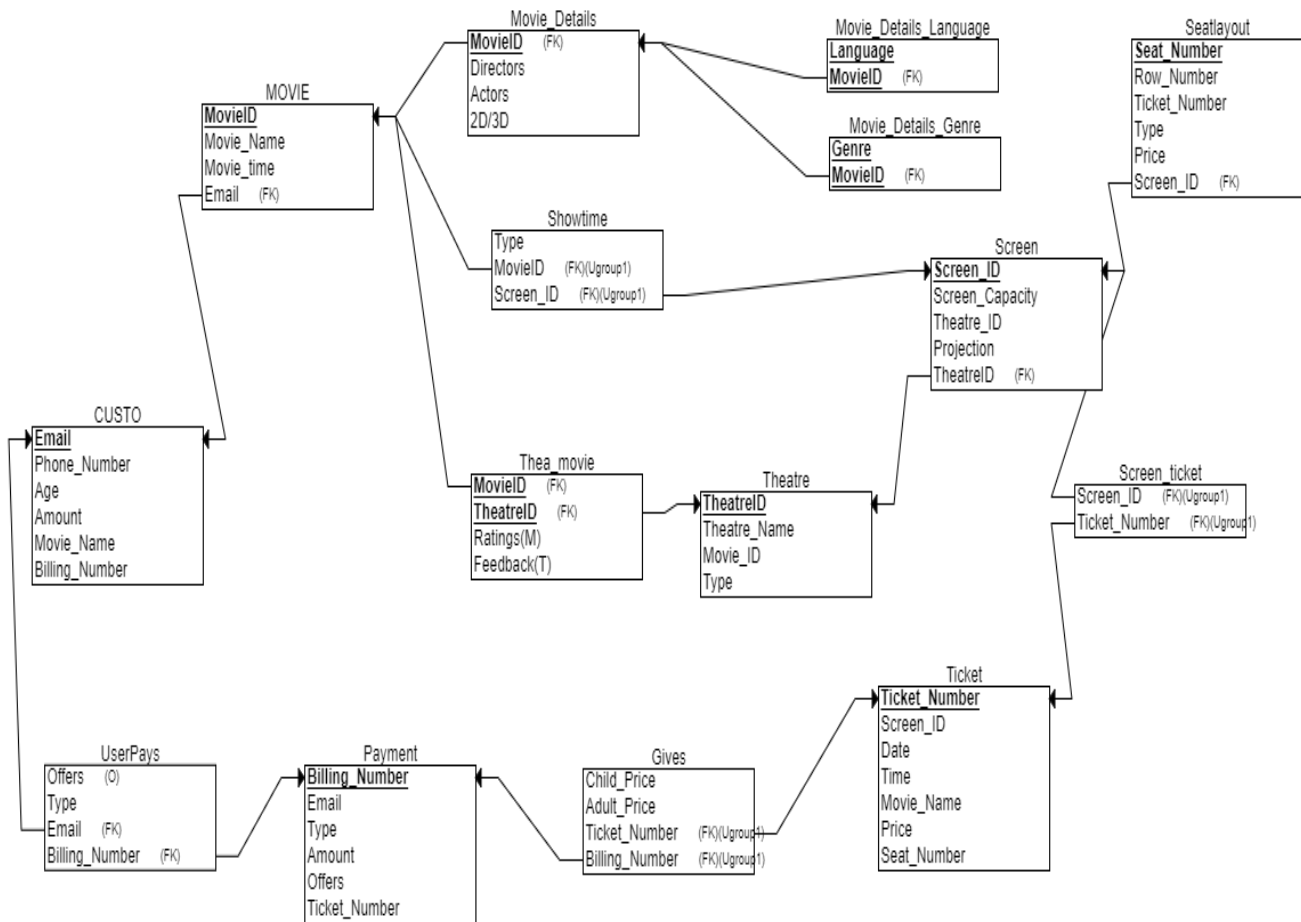
MOVIE DETAILS LANGUAGE

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	Language 🔑	varchar(20)	utf8mb4_general_ci		No	None		
2	MovieID 🔑 🔒	int(11)			No	None		

MOVIE DETAILS GENRE

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	Genre 🔑	varchar(20)	utf8mb4_general_ci		No	None		
2	MovieID 🔑 🔒	int(11)			No	None		

4. SCHEMA DIAGRAM:



5. DATABASE IMPLEMENTATION

5.1 CREATION OF SPECIFIED TABLES:

5.2 INSERT DATA VALUES:

THEATRE

```
❖ CREATE TABLE Theatre
(
    TheatreID INT NOT NULL,
    Theatre_Name VARCHAR(20) NOT NULL,
    Movie_ID INT NOT NULL,
    Type VARCHAR(20) NOT NULL,
    PRIMARY KEY (TheatreID)
);

INSERT INTO Theatre VALUES(1,'Gold', 100,'Single screen');
INSERT INTO Theatre VALUES (2,'PVR', 101,'Multiplex');
INSERT INTO Theatre VALUES (3,'Inox', 102,'Single screen');
INSERT INTO Theatre VALUES (4,'7 seas', 103,'Multiplex');
INSERT INTO Theatre VALUES (5,'Imax', 104,'Multiplex');
```

```
SQL> SELECT * FROM THEATRE;
```

THEATREID	THEATRE_NAME	MOVIE_ID	TYPE
1	Gold	100	Single screen
2	PVR	101	Multiplex
3	Inox	102	Single screen
4	7 seas	103	Multiplex
5	Imax	104	Multiplex

```
SQL>
```

MOVIE

```
❖ CREATE TABLE MOVIE
(
    MovieID INT NOT NULL,
    Movie_Name VARCHAR(30) NOT NULL,
    Movie_time TIMESTAMP NOT NULL,
    Email VARCHAR(30) NOT NULL,
    PRIMARY KEY (MovieID),
    FOREIGN KEY (Email) REFERENCES CUSTO(Email)
);

INSERT INTO MOVIE VALUES (100,' Serious men',TO_DATE('2001/01/01 00:09:30','YYYY/MM/DD HH24:MI,SS'),'tohego2805@dmeproject.com');
INSERT INTO MOVIE VALUES (101,' Serious men', TO_DATE('2001/01/11 00:12:30','YYYY/MM/DD HH24:MI,SS'),'curveted@panterra.com');
```

Online Movie Ticket Booking System

```
INSERT INTO MOVIES VALUES (102,' Serious men', TO_DATE('2001/11/11 00:06:30','YYYY/MM/DD
HH24:MI,SS'),'shahaksh23@gmail.com');
INSERT INTO MOVIES VALUES (103,' Serious men', TO_DATE('2001/10/21 00:12:30','YYYY/MM/DD
HH24:MI,SS'),'simp29@yahoo.com');
INSERT INTO MOVIES VALUES (104,' Serious men', TO_DATE('2001/01/06 00:21:30','YYYY/MM/DD
HH24:MI,SS'),'Zhen876@gmail.com');
```

MOVIEID	MOVIE_NAME	EMAIL	TIME
100	Serious men	tohego2805@dmeproject.com	01 JANUARY 01 12:09:30 AM
101	3 idiots	curveted@panterrna.com	11 JANUARY 01 12:12:30 AM
103	Panipat	simp29@yahoo.com	21 OCTOBER 01 12:12:30 AM
104	MS Dhoni	Zhen876@gmail.com	06 JANUARY 01 12:21:30 AM
102	Dabbang	Shahaksh23@gmail.com	11 NOVEMBER 01 12:06:30 AM

SCREEN

```
❖ CREATE TABLE Screen
(
  Screen_ID INT NOT NULL,
  Screen_Capacity INT NOT NULL,
  Theatre_ID INT NOT NULL,
  Projection VARCHAR(15) NOT NULL,
  TheatreID INT NOT NULL,
  PRIMARY KEY (Screen_ID),
  FOREIGN KEY (TheatreID) REFERENCES Theatre(TheatreID)
);
```

```
INSERT INTO Screen VALUES ('S1',100, 1,'Lazer');
INSERT INTO Screen VALUES ('S2', 80, 2,'Lazer');
INSERT INTO Screen VALUES ('D3',200,3,'Lazer');
INSERT INTO Screen VALUES ('Z2',250, 4,'digital');
INSERT INTO Screen VALUES ('R3', 200, 5,'digital');
```

```
SQL> select * from Screen;
```

SCREEN_ID	SCREEN_CAPACITY	THEATRE_ID	PROJECTION	THEATREID
1	100	1	Lazer	1
2	250	4	digital	2
3	200	3	Lazer	3
4	250	4	digital	4
5	200	5	digital	5

TICKETS

```
❖ CREATE TABLE Tickets
(
  Ticket_Number INT NOT NULL,
  Screen_ID INT NOT NULL,
  MDate_Time DATE NOT NULL,
  Movie_Name VARCHAR(30) NOT NULL,
  Price NUMERIC(5) NOT NULL,
  Seat_Number VARCHAR(5) NOT NULL,
  PRIMARY KEY (Ticket_Number)
);
```

```

INSERT INTO Tickets VALUES (123456,1,TO_DATE('13/10/2020 21:30:00','DD/MM/YYYY HH24:MI:SS'),
'Serious men', 150, 'R18');
INSERT INTO Tickets VALUES (234567,2, TO_DATE('07/09/2020 00:30:00','DD/MM/YYYY HH24:MI:SS'), '3
idiots', 120, 'S20');
INSERT INTO Tickets VALUES (345678,3, TO_DATE('12/10/2020 15:30:00','DD/MM/YYYY
HH24:MI:SS'),'Dabbang', 100, 'T07');
INSERT INTO Tickets VALUES (45678,4, TO_DATE('08/09/2020 18:30:00','DD/MM/YYYY HH24:MI:SS'),
'Panipat', 200, 'D02');
INSERT INTO Tickets VALUES (567890,5, TO_DATE('11/10/2020 21:30:00','DD/MM/YYYY HH24:MI:SS'), 'MS
dhoni', 250, 'V15');

```

```

SQL> select * from tickets;

```

TICKET_NUMBER	SCREEN_ID	MDATE_TIM	MOVIE_NAME	PRICE	SEAT_
123456	1	13-OCT-20	Serious men	150	R18
234567	2	07-SEP-20	3 idiots	120	S20
345678	3	12-OCT-20	Dabbang	100	T07
456789	4	08-SEP-20	Panipat	200	D02
567890	5	11-OCT-20	MS dhoni	250	V15

SEATLAYOUT

❖ CREATE TABLE Seatlayout

```

(
  Seat_Number VARCHAR(5) NOT NULL,
  Row_Number CHAR(2) NOT NULL,
  Ticket_Number INT NOT NULL,
  Type VARCHAR(15) NOT NULL,
  Price NUMERIC(5) NOT NULL,
  Screen_ID INT NOT NULL,
  PRIMARY KEY (Seat_Number),
  FOREIGN KEY (Screen_ID) REFERENCES Screen(Screen_ID)
);

```

```

INSERT INTO Seatlayout VALUES ('R18',' R', 123456, 'Gold',150, 1);
INSERT INTO Seatlayout VALUES ('S20',' S', 234567, 'Silver', 120,2);
INSERT INTO Seatlayout VALUES ('T07',' T', 345678, 'Silver', 100, 3);
INSERT INTO Seatlayout VALUES ('D02',' D', 456789, 'Diamond', 200, 4);
INSERT INTO Seatlayout VALUES ('V15',' V', 567890, 'Recliner', 250,5);

```

```

SQL> select * from Seatlayout;

```

SEAT_	RO	TICKET_NUMBER	TYPE	PRICE	SCREEN_ID
S20	S	234567	Silver	120	2
T07	T	345678	Silver	100	3
D02	D	456789	Diamond	200	2
V15	V	567890	Recliner	250	3

PAYMENT

❖ CREATE TABLE Payment

```

(
  Billing_Number INT NOT NULL,
  Email VARCHAR(30) NOT NULL,
  Type VARCHAR(10) NOT NULL,
  Amount INT NOT NULL,

```

Online Movie Ticket Booking System

```
Offers VARCHAR(10) NOT NULL,  
Ticket_Number INT NOT NULL,  
PRIMARY KEY (Billing_Number)  
);
```

```
INSERT INTO Payment VALUES (98765432,' tohego2805@dmeproject.com ', 'Credit card', 160, '5%',  
123456);  
INSERT INTO Payment VALUES (3456789,' curveted@panterrra.com ', 'Debit card', 170, 'null', 234567);  
INSERT INTO Payment VALUES (09876543,' Shahaksh23@gmail.com ', 'Google pay', 100, '20%', 345678);  
INSERT INTO Payment VALUES (76543209,' simp29@yahoo.com ', 'Paytm',250, 'null', '456789');  
INSERT INTO Payment VALUES (23456098,' Zhen876@gmail.com ', 'Phone pe', 280, '10%', '567890');
```

```
SQL> select * from payment;
```

BILLING_NUMBER	EMAIL	TYPE	AMOUNT	OFFERS	TICKET_NUMBER
34567892	curveted@panterrra.com	Debit card	170	null	234567
9876543	Shahaksh23@gmail.com	Google pay	100	20%	345678
76543209	simp29@yahoo.com	Paytm	250	null	456789
23456098	Zhen876@gmail.com	Phone pe	280	10%	567890
98765432	tohego2805@dmeproject.com	Creditcard	160	5%	123456

CUSTO

```
❖ CREATE TABLE CUSTO  
(  
Phone_Number NUMERIC(10) NOT NULL,  
Age INT NOT NULL,  
Amount INT NOT NULL,  
Email VARCHAR(30) NOT NULL,  
Movie_Name VARCHAR(30) NOT NULL,  
Billing_Number INT NOT NULL,  
PRIMARY KEY (Email)  
);
```

```
INSERT INTO CUSTO VALUES (8780655138, 19,160, 'tohego2805@dmeproject.com', 'Serious man',  
98765432);  
INSERT INTO CUSTO VALUES (8400303448, 18, 170, 'curveted@panterrra.com', '3 idiots', 34567892);  
INSERT INTO CUSTO VALUES (7016327917, 18, 100, 'Shahaksh23@gmail.com', 'Dabbang', 09876543);  
INSERT INTO CUSTO VALUES (9998343509, 19, 250, 'simp29@yahoo.com', 'Panipat', 76543209);  
INSERT INTO CUSTO VALUES (9824361601, 45, 280, 'Zhen876@gmail.com', 'MS dhoni', 23456098);
```

```
SQL> select * from custo;
```

PHONE_NUMBER	AGE	AMOUNT	EMAIL	MOVIE_NAME	BILLING_NUMBER
8780655138	19	160	tohego2805@dmeproject.com	Serious man	98765432
8400303448	18	170	curveted@panterrra.com	3 idiots	34567892
7016327917	18	100	Shahaksh23@gmail.com	Dabbang	
9998343509	19	250	simp29@yahoo.com	Panipat	76543209
9824361601	45	280	Zhen876@gmail.com	MS dhoni	23456098

SCREEN_TICKETS

```
❖ CREATE TABLE Screen_tickets
(
  Screen_ID INT NOT NULL,
  Ticket_Number INT NOT NULL,
  FOREIGN KEY (Screen_ID) REFERENCES Screen(Screen_ID),
  FOREIGN KEY (Ticket_Number) REFERENCES Tickets(Ticket_Number),
  UNIQUE (Screen_ID, Ticket_Number)
);
```

```
INSERT INTO Screen_tickets VALUES (1, 123456);
INSERT INTO Screen_tickets VALUES (2, 234567);
INSERT INTO Screen_tickets VALUES (3, 345678);
INSERT INTO Screen_tickets VALUES (4, 456789);
INSERT INTO Screen_tickets VALUES (5, 567890);
```

```
SQL> select * from screen_tickets;
```

SCREEN_ID	TICKET_NUMBER
1	123456
2	234567
3	345678
4	456789
5	567890

PAYTICKET

```
❖ CREATE TABLE Payticket
(
  Child_Price NUMERIC(5) NOT NULL,
  Adult_Price NUMERIC(5) NOT NULL,
  Ticket_Number INT NOT NULL,
  Billing_Number INT NOT NULL,
  FOREIGN KEY (Ticket_Number) REFERENCES Tickets(Ticket_Number),
  FOREIGN KEY (Billing_Number) REFERENCES Payment(Billing_Number),
  UNIQUE (Ticket_Number, Billing_Number)
);
```

```
INSERT INTO Payticket VALUES (140, 150, 123456, 98765432);
INSERT INTO Payticket VALUES (130, 120, 234567, 34567892);
INSERT INTO Payticket VALUES (90, 100, 345678, 09876543);
INSERT INTO Payticket VALUES (190, 200, 456789, 76543209);
INSERT INTO Payticket VALUES (240, 250, 567890, 23456098);
```

```
SQL> select * from Payticket;
```

CHILD_PRICE	ADULT_PRICE	TICKET_NUMBER	BILLING_NUMBER
140	150	123456	98765432
130	120	234567	34567892
190	200	456789	76543209
240	250	567890	23456098

```
SQL>
```


USERPAYS

❖ CREATE TABLE UserPays

```
(
  Offers VARCHAR(20),
  Type VARCHAR(20) NOT NULL,
  Email VARCHAR(30) NOT NULL,
  Billing_Number INT NOT NULL,
  FOREIGN KEY (Email) REFERENCES CUSTO(Email),
  FOREIGN KEY (Billing_Number) REFERENCES Payment(Billing_Number)
);
```

```
INSERT INTO UserPays VALUES ('5%', 'Credit Card', 'tohego2805@dmeproject.com', 9876543);
```

```
INSERT INTO UserPays VALUES ('Null', 'Debit Card', 'curveted@panterra.com', 34567892);
```

```
INSERT INTO UserPays VALUES ('20%', 'Google pay', 'Shahaksh23@gmail.com', 09876543);
```

```
INSERT INTO UserPays VALUES ('Null', 'Paytm', 'simp29@yahoo.com', 76543209);
```

```
INSERT INTO UserPays VALUES ('10%', 'Phone pe', 'Zhen876@gmail.com', 23456098);
```

```
SQL> select * from userpays;
```

OFFERS	TYPE	EMAIL	BILLING_NUMBER
5%	Credit Card	tohego2805@dmeproject.com	9876543
Null	Debit Card	curveted@panterra.com	34567892
20%	Google pay	Shahaksh23@gmail.com	9876543
Null	Paytm	simp29@yahoo.com	76543209
10%	Phone pe	Zhen876@gmail.com	23456098

MOVIE_DETAILS

❖ CREATE TABLE Movies_Details

```
(
  Directors VARCHAR(20) NOT NULL,
  Actors VARCHAR(20) NOT NULL,
  Dim CHAR(2) NOT NULL,
  MovieID INT NOT NULL,
  PRIMARY KEY (MovieID),
  FOREIGN KEY (MovieID) REFERENCES MOVIES(MovieID)
);
```

```
INSERT INTO Movies_Details VALUES ('Shankar','S Khan','2D','100');
```

```
INSERT INTO Movies_Details VALUES ('Yash chopra','Yang yang','2D','101');
```

```
INSERT INTO Movies_Details VALUES ('Vishal Bhardwaj','Zheng sheng','3D','102');
```

```
INSERT INTO Movies_Details VALUES ('Neeraj pandey','Mike angelo','2D','103');
```

```
INSERT INTO Movies_Details VALUES ('Shoojit sircar','K kaif','2D','104');
```

```
SQL>
SQL> select * from movies_details;

DIRECTORS      ACTORS      DI      MOVIEID
-----
Shankar        S Khan      2D      100
Yash chopra    Yang yang   2D      101
Vishal Bhardwaj Zheng sheng  3D      102
Neeraj pandey  Mike angelo 2D      103
Shoojit sircar K kaif      2D      104

SQL>
```

THEATRE_MOVIE

```
❖ CREATE TABLE Theatre_movie
(
    RatingsM INT NOT NULL,
    FeedbackT VARCHAR(30) NOT NULL,
    MovieID INT NOT NULL,
    TheatreID INT NOT NULL,
    PRIMARY KEY (MovieID, TheatreID),
    FOREIGN KEY (MovieID) REFERENCES MOVIES(MovieID),
    FOREIGN KEY (TheatreID) REFERENCES Theatre(TheatreID)
);
```

```
INSERT INTO Theatre_movie VALUES (1,'Good',100,1);
INSERT INTO Theatre_movie VALUES (5,'Fab',101,2);
INSERT INTO Theatre_movie VALUES (4,'Bad',102,3);
INSERT INTO Theatre_movie VALUES (3,'Best',103,4);
INSERT INTO Theatre_movie VALUES (2,'Better',104,5);
```

```
SQL>
SQL> select * from theatre_movie;

RATINGSM FEEDBACKT      MOVIEID  THEATREID
-----
1 Good      100      1
5 Fab       101      2
4 Bad       102      3
3 Best      103      4
2 Better    104      5
```

SHOWTIME

```
❖ CREATE TABLE Showtim
(
    Type VARCHAR(10) NOT NULL,
    MovieID INT NOT NULL,
    Screen_ID INT NOT NULL,
    FOREIGN KEY (MovieID) REFERENCES MOVIES(MovieID),
    FOREIGN KEY (Screen_ID) REFERENCES Screen(Screen_ID),
    UNIQUE (MovieID, Screen_ID)
);
```

```
INSERT INTO Showtim VALUES ('Gold',100,1);
INSERT INTO Showtim VALUES ('Silver',101,2);
INSERT INTO Showtim VALUES ('Silver',102,3);
INSERT INTO Showtim VALUES ('Diamond',103,4);
INSERT INTO Showtim VALUES ('Recliner',104,5);
```

```
SQL> SELECT * FROM SHOWTIME
2 ;
```

TYPE	MOVIEID	SCREEN_ID
Gold	100	1
Silver	101	2
Silver	102	3
Diamond	103	4
Recliner	104	5

```
SQL>
```

MOVIES_DETAILS_LANGUAGE

- ❖ CREATE TABLE Movies_Details_Language
(
Language VARCHAR(20) NOT NULL,
MovieID INT NOT NULL,
PRIMARY KEY (Language, MovieID),
FOREIGN KEY (MovieID) REFERENCES Movies_Details(MovieID)
);

INSERT INTO Movies_Details_Language VALUES ('Tamil',100);
INSERT INTO Movies_Details_Language VALUES ('English',101);
INSERT INTO Movies_Details_Language VALUES ('Hindi',102);
INSERT INTO Movies_Details_Language VALUES ('Gujarati',103);
INSERT INTO Movies_Details_Language VALUES ('Hindi',104);

```
SQL>
SQL> select * from movies_details_language;
```

LANGUAGE	MOVIEID
Tamil	100
English	101
Hindi	102
Gujarati	103
Hindi	104

```
SQL>
```

MOVIES_DETAILS_GENRE

- ❖ CREATE TABLE Movies_Details_Genre
(
Genre VARCHAR(20) NOT NULL,
MovieID INT NOT NULL,
PRIMARY KEY (Genre, MovieID),
FOREIGN KEY (MovieID) REFERENCES Movies_Details(MovieID)
);

INSERT INTO Movies_Details_Genre VALUES ('Horror',100);
INSERT INTO Movies_Details_Genre VALUES ('Sci fi',101);
INSERT INTO Movies_Details_Genre VALUES ('Action',102);
INSERT INTO Movies_Details_Genre VALUES ('Romance',103);
INSERT INTO Movies_Details_Genre VALUES ('Comedy',104);

Online Movie Ticket Booking System

```
SQL> select * from movies_details_genre;
```

GENRE	MOVIEID
Horror	100
Sci fi	101
Action	102
Romance	103
Comedy	104

5.3 QUERIES:

1. Create a query to get a Max price of the tickets.

```
SQL>
SQL> select max(price) from seatlayout;

MAX(PRICE)
-----
          250
```

2. Create a query to get average price of tickets.

```
SQL> select avg(price) from seatlayout;

AVG(PRICE)
-----
        167.5
```

3. Create a query to get a minimum price of tickets.

```
SQL> select min(price) from seatlayout;

MIN(PRICE)
-----
         100
```

4. Update theatroid=4 whose screen_id =4.

```
1      100      1 Lazer      1
2      250      4 digital     4
3      200      3 Lazer      3
4      250      4 digital     4
5      200      5 digital     5

SQL> update Screen set theatroid=2 where screen_id=2;

1 row updated.

SQL> select * from Screen;

SCREEN_ID SCREEN_CAPACITY THEATRE_ID PROJECTION  THEATREID
-----
1          100          1 Lazer      1
2          250          4 digital     2
3          200          3 Lazer      3
4          250          4 digital     4
5          200          5 digital     5
```

5. Create a query to count number of customers in theatre.

```
SQL> select count(email) from custo;

COUNT(EMAIL)
-----
5
```

6. Create a query which displays the screen capacity of the particular theatres which have digital projection.

```
SQL> select screen_capacity from screen where projection='digital';

SCREEN_CAPACITY
-----
250
250
200
```

7. Create a query which shows types of projection and types of seats in particular theatre.

```
SQL> select screen.projection,seatlayout.type from screen inner join seatlayout on screen.screen_id=seatlayout.screen_id;
```

PROJECTION	TYPE
digital	Silver
Lazer	Silver
digital	Diamond
Lazer	Recliner

8. Create a query which can identify movie in a particular theatre.

```
SQL> select theatre.theatre_name,showtim.movieid from theatre inner join showtim on theatre.movie_id=showtim.movieid;
```

THEATRE_NAME	MOVIEID
Gold	100
PVR	101
Inox	102
7 seas	103
Imax	104

9. Create a query which displays the complete information about which customers pay in what way.

```
SQL> select * from custo FULL outer join payment on CUSTO.EMAIL=payment.EMAIL;
```

PHONE_NUMBER	AGE	AMOUNT	EMAIL	TYPE	AMOUNT	OFFERS	MOVIE_NAME	TICKET_NUMBER	BILLING_NUMBER	BILLING_NUMBER
			curveted@panterrra.com	Debit card	170	null	234567		34567892	
			Shahaksh23@gmail.com	Google pay	100	20%	345678		9876543	
			simp29@yahoo.com	Paytm	250	null	456789		76543209	
			Zhen876@gmail.com	Phone pe	280	10%	567890		23456098	
			tohego2805@dmeproject.com	Creditcard	160	5%	123456		98765432	
9824361601	45	280	Zhen876@gmail.com				MS dhoni		23456098	
7016327917	18	100	Shahaksh23@gmail.com				Dabbang		9876543	
8780655138	19	160	tohego2805@dmeproject.com				Serious man		98765432	
9998343509	19	250	simp29@yahoo.com				Panipat		76543209	
8400303448	18	170	curveted@panterrra.com				3 idiots		34567892	

10 rows selected.

10. Create a query which gives average screen capacity of those theatre for each type of projection.

```
SQL> SELECT AVG(SCREEN_CAPACITY),PROJECTION FROM SCREEN GROUP BY PROJECTION;
```

AVG(SCREEN_CAPACITY)	PROJECTION
150	Lazer
233.333333	digital

11. Create a query which displays count for each type of different seats available in a theatre.

```
SQL> SELECT COUNT(PRICE),TYPE FROM SEATLAYOUT GROUP BY TYPE;
```

COUNT(PRICE)	TYPE
1	Diamond
1	Recliner
2	Silver

12. Write a query which displays full information of customer and displays age in descending order

```
SQL> SELECT * FROM CUSTO
2 ORDER BY AGE DESC;
```

PHONE_NUMBER	AGE	AMOUNT	EMAIL	MOVIE_NAME	BILLING_NUMBER
9824361601	45	280	Zhen876@gmail.com	MS dhoni	23456098
9998343509	19	250	simp29@yahoo.com	Panipat	76543209
8780655138	19	160	tohego2805@dmeproject.com	Serious man	98765432
8400303448	18	170	curveted@panterrria.com	3 idiots	34567892
7016327917	18	100	Shahaksh23@gmail.com	Dabbang	9876543

13. Write a query which displays full information about customer whose age is 19 yrs and had brought a ticket of total amount 160rs

```
SQL> SELECT * FROM CUSTO WHERE Age=19 AND Amount=160;
```

PHONE_NUMBER	AGE	AMOUNT	EMAIL	MOVIE_NAME	BILLING_NUMBER
8780655138	19	160	tohego2805@dmeproject.com	Serious man	98765432

14. Create a query which displays full information of Seat layout for those seats whose price is greater than 100

```
SQL> SELECT * FROM Seatlayout WHERE Seat_Number IN (SELECT Seat_Number FROM Seatlayout WHERE Price > 100) ;
```

SEAT_	RO	TICKET_NUMBER	TYPE	PRICE	SCREEN_ID
S20	S	234567	Silver	120	2
D02	D	456789	Diamond	200	2
V15	V	567890	Recliner	250	3

15. Create a query which shows which movie is running on which theatre

```
SQL> select movies.movie_name,theatre.theatre_name from movies inner join theatre on movies.movieid=theatre.movie_id;
```

MOVIE_NAME	THEATRE_NAME
Serious men	Gold
3 idiots	PVR
Dabbang	Inox
Panipat	7 seas
MS Dhoni	Imax

5.4 PROCEDURE:

1. Create procedure which displays a message after inserting new entry values of customer that they have successfully booked their tickets.

create or replace procedure details is

Ticket_Number INT := 1029348;

Movie_Name VARCHAR(30) := 'Robot2.0';

MDate DATE := (to_date('13/07/2020','dd/mm/yyyy'));

begin

DBMS_OUTPUT.PUT_LINE('Your ticket of ticket number :' || Ticket_Number || 'of movie:' ||

Movie_Name || 'has been booked at date:' || MDate);

end details;

```
create or replace procedure details is
Ticket_Number INT := 1029348;
Movie_Name VARCHAR(30) := 'Robot2.0';
MDate DATE := (to_date('13/07/2020','dd/mm/yyyy'));
begin
DBMS_OUTPUT.PUT_LINE( 'Your ticket of ticket number :' || Ticket_Number || 'of movie:' || Movie_Name || 'has been booked at date:' || MD
end details;
```

Procedure created.

After executing procedure

EXEC details

EXEC details

Statement processed.

Your ticket of ticket number :1029348of movie:Robot2.0has been booked at date:13-JUL-20

2. Create procedure which displays a message after inserting new entry values of customer that they have successfully booked their tickets.

create or replace procedure NO_OF_PERSON is

no_of_persons numeric(2) := 2;

tick_no INT ;

movie VARCHAR(30);

pri numeric(5);

total_amt decimal(10,2);

begin

Select Movie_Name, price,Ticket_Number into movie, pri,tick_no from ticket;

total_amt := no_of_persons * pri;

DBMS_OUTPUT.PUT_LINE('Your ticket of ticket number :' || tick_no || 'of movie:' || movie ||

total amount'|| total_amt ||'has been booked ');

end NO_OF_PERSON;

Online Movie Ticket Booking System

```
1 create or replace procedure NO_OF_PERSON is
2 no_of_persons numeric(2) := 2;
3 tick_no INT ;
4 movie VARCHAR(30);
5 pri numeric(5);
6 total_amt decimal(10,2);
7 begin
8 Select PRICE into pri from ticket WHERE TICKET_NUMBER=123456;
9 total_amt := no_of_persons * pri;
10 DBMS_OUTPUT.PUT_LINE( 'Your ticket of total amount ' || total_amt || ' has been booked ' );
11 end NO_OF_PERSON;
12 EXEC NO_OF_PERSON;
13 Select * from ticket;
```

Statement processed.

Your ticket of total amount 300 has been booked

5.5 FUNCTION:

1. Write a function which counts total number of payments done till and displays same

```
CREATE OR REPLACE FUNCTION total_payments
RETURN number IS
co number(3) := 0;
BEGIN
SELECT count (*) into co FROM payment;
RETURN co;
END;
```

```
DECLARE
C number(3);
BEGIN
C := total_payments();
Dbms_output.putline ( ' total no. of payments: ' || C );
END
```

OUTPUT:

SQL Worksheet

```

1 CREATE OR REPLACE FUNCTION total_payments
2 RETURN number IS
3 co number(3) := 0;
4 Billing_Number INT;
5 TYPE1 VARCHAR (20);
6 Offers VARCHAR(10);
7 Amount INT;
8 BEGIN
9 SELECT count ( Billing_Number) into co FROM payment;
10 RETURN co;
11 END;
```

Function created.

```

DECLARE
Cc number(3);
BEGIN
Cc := total_payments();
DBMS_OUTPUT.PUT_LINE( ' total number of payments: ' || Cc );
END;
```

Statement processed.
total number of payments: 5

5.6 TRIGGER:

1. Create a trigger which shows how the prices of tickets has been changed and compared them by showing old and new price simultaneously

```
CREATE OR REPLACE TRIGGER display_PRICES_changes
BEFORE DELETE OR INSERT OR UPDATE ON Tickets
FOR EACH ROW
WHEN (NEW.price > 0)
DECLARE
    ticket_diff number;
BEGIN
    ticket_diff := :NEW.Price - :OLD.Price;
    dbms_output.put_line('Old Price: ' || :OLD.Price);
    dbms_output.put_line('New Price: ' || :NEW.Price);
    dbms_output.put_line('Price difference: ' || ticket_diff);
END;
/
```

```
INSERT INTO Tickets VALUES (120000,1,TO_DATE('10/10/2020 21:30:00','DD/MM/YYYY
HH24:MI:SS'), 'Serious men', 200, 'R20');
```

```
UPDATE Tickets
SET Price = Price + 50
WHERE Ticket_Number = 120000;
```

OUTPUT:

The screenshot shows an SQL Worksheet interface. The top bar has a 'Clear' button. The main area contains SQL code with line numbers 26 to 40. The code defines a trigger named 'display_PRICES_changes' that fires before delete, insert, or update on the 'Tickets' table. It declares a variable 'ticket_diff' and calculates the price difference between the new and old values. The trigger body includes three dbms_output.put_line statements to display the old price, new price, and the difference. Below the code, the execution results are shown: '1 row(s) updated.', 'Old Price: 150', 'New Price: 200', and 'Price difference: 50'.

```
26 FOR EACH ROW
27 WHEN (NEW.price > 0)
28 DECLARE
29     ticket_diff number;
30 BEGIN
31     ticket_diff := :NEW.Price - :OLD.Price;
32     dbms_output.put_line('Old Price: ' || :OLD.Price);
33     dbms_output.put_line('New Price: ' || :NEW.Price);
34     dbms_output.put_line('Price difference: ' || ticket_diff);
35 END;
36 /
37 update price = 160 where screen_id=1;
38 UPDATE Ticket
39 SET Price = Price + 50
40 WHERE Ticket_Number = 123456
```

1 row(s) updated.
 Old Price: 150
 New Price: 200
 Price difference: 50

2. Create a trigger which generates automatic subsequent billing number starting from 1000001 while new data entry of customer is entered.

```
CREATE TABLE CUSTO
(
  Phone_Number NUMERIC(10),
  Age INT ,
  Amount INT ,
  Email VARCHAR(30) NOT NULL,
  Movie_Name VARCHAR(30) ,
  Billing_Number INT ,
  PRIMARY KEY (Email)
)
```

```
INSERT INTO CUSTO VALUES (8780655138, 19,160,'tohego2805@dmeproject.com','Serious man',
98765432)
```

```
INSERT INTO CUSTO VALUES (8400303448, 18, 170,'curveted@panterra.com','3 idiots', 34567892)
```

```
INSERT INTO CUSTO VALUES (7016327917, 18, 100,'Shahaksh23@gmail.com','Dabbang', 09876543)
```

```
INSERT INTO CUSTO VALUES (9998343509, 19, 250,'simp29@yahoo.com','Panipat', 76543209)
```

```
INSERT INTO CUSTO VALUES (9824361601, 45, 280,'Zhen876@gmail.com','MS dhoni', 23456098)
```

```
create sequence bill_seq start with 1000001;
CREATE TABLE CUSTO
(
  Phone_Number NUMERIC(10),
  Age INT ,
  Amount INT ,
  Email VARCHAR(30) NOT NULL,
  Movie_Name VARCHAR(30) ,
  Billing_Number INT ,
  PRIMARY KEY (Email)
);
INSERT INTO CUSTO VALUES (8780655138, 19,160,'tohego2805@dmeproject.com','Serious man', 98765432);
INSERT INTO CUSTO VALUES (8400303448, 18, 170,'curveted@panterra.com','3 idiots', 34567892);
INSERT INTO CUSTO VALUES (7016327917, 18, 100,'Shahaksh23@gmail.com','Dabbang', 09876543);
INSERT INTO CUSTO VALUES (9998343509, 19, 250,'simp29@yahoo.com','Panipat', 76543209);
INSERT INTO CUSTO VALUES (9824361601, 45, 280,'Zhen876@gmail.com','MS dhoni', 23456098);
select * from custo;
```

Output before trigger:

SQL Worksheet

PHONE_NUMBER	AGE	AMOUNT	EMAIL	MOVIE_NAME	BILLING_NUMBER
8780655138	19	160	tohego2805@dmeproject.com	Serious man	98765432
8400303448	18	170	curveted@panterra.com	3 idiots	34567892
7016327917	18	100	Shahaksh23@gmail.com	Dabbang	9876543
9998343509	19	250	simp29@yahoo.com	Panipat	76543209
9824361601	45	280	Zhen876@gmail.com	MS dhoni	23456098

Download CSV

5 rows selected.

create or replace trigger tr_custo
before insert or update on custo
for each row

Online Movie Ticket Booking System

```
begin
  if inserting and :new.Billing_Number is null then
    :new.Billing_Number := bill_seq.nextval;
  end if;
end;
```

```
insert into custo (Phone_Number, Age, Amount, Email) values (1234567890, 20, 400, 'vash5547@outlook.com')
select * from custo
```

```
create or replace trigger tr_custo
  before insert or update on custo
  for each row
begin
  if inserting and :new.Billing_Number is null then
    :new.Billing_Number := bill_seq.nextval;
  end if;
end;
/
insert into custo (Phone_Number, Age, Amount, Email) values (1234567890, 20, 400, 'vash5547@outlook.com');
```

Output after trigger:

SQL Worksheet					
PHONE_NUMBER	AGE	AMOUNT	EMAIL	MOVIE_NAME	BILLING_NUMBER
8780655138	19	160	tohego2805@dmeproject.com	Serious man	98765432
8400303448	18	170	curveted@panterrna.com	3 idiots	34567892
7016327917	18	100	Shahaksh23@gmail.com	Dabbang	9876543
9998343509	19	250	simp29@yahoo.com	Panipat	76543209
9824361601	45	280	Zhen876@gmail.com	MS dhoni	23456098
1234567890	20	400	vash5547@outlook.com	-	10000001

[Download CSV](#)
6 rows selected.

5.7 CURSOR:

1. This Cursor Gives The Information About the updates increase in price when admin does any changes.

```

DECLARE
    total_rows number(2);
BEGIN
    UPDATE SeatLayout
    SET Price = Price + 500;
    IF sql%notfound THEN
        dbms_output.put_line('no customers selected');
    ELSIF sql%found THEN
        total_rows := sql%rowcount;
        dbms_output.put_line( total_rows || ' customers selected ');
    END IF;
END;
/

```

OUTPUT:

```

SQL>
SQL> DECLARE
2   total_rows number(2);
3   BEGIN
4   UPDATE SeatLayout
5   SET Price = Price + 500;
6   IF sql%notfound THEN
7       dbms_output.put_line('no customers selected');
8   ELSIF sql%found THEN
9       total_rows := sql%rowcount;
10      dbms_output.put_line( total_rows || ' customers selected ');
11  END IF;
12  END;
13  /

PL/SQL procedure successfully completed.

SQL> select * from seatlayout
2  ;

```

SEAT_	RO	TICKET_NUMBER	TYPE	PRICE	SCREEN_ID
S20	S	234567	Silver	620	2
T07	T	345678	Silver	600	3
D02	D	456789	Diamond	700	2
V15	V	567890	Recliner	750	3

```

SQL>

```


6. FUTURE ENHANCEMENTS OF THE SYSTEM

- We've used localhost/phpMyAdmin to create the Back-end Design and we also can use Python or PHP Languages in Future.
- For security purpose New Entries is done using OTP.
- New facilities can be shared via SMS to customers.
- In future we can add discounts, payment methods and more features.
- We will make database more consistent and we are making this database efficient and easy to implement with huge data capacity. Methods and user data input will be lot easy after the implement of GUI.
- We will also add some extra features so that the users can get answer for their complaints as fast as possible.

7.BIBLIOGRAPHY

- For the successful implementation of this Online Movie Ticket Booking System project we referred to many websites and books.
- We created the ER Diagram on “erdplus.com” and Schema Diagram on “localhost/phpMyAdmin”.
- Mostly we referred the online material for syntax of procedures, triggers, Exception and cursors.

Reference Book:

Books: Database System Concepts

By: Henry F. Korth

PL/SQL Programming

By: Ivan Bayross

Reference Website:

1. <https://www.w3schools.com/sql>
2. <https://www.tutorialspoint.com/plsql>
3. <https://www.mysqltutorial.org/>