ONLINE MOVIE TICKET BOOKING SYSTEM

REVIEW-REPORT

Submitted by

Vansh Aggarwal (19BCE0987)

Prepared For

DATABASE MANAGEMENT SYSTEM (CSE2004)

PROJECT COMPONENT

Submitted To

Dr. Nithya S

Associate Professor

School of Computer Science and Engineering

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Table of Content

Chapter

Abstract

1. Introduction

    1.1 Background

    1.2 Objective

    1.3 Motivation

    1.4 Contributions of the Project

    1.5 Organization of the Project

2. Project Resource Requirements

    2.1 Software Requirements

    2.2 Hardware Requirements

3. Literature Survey

4. Design of the Project

    4.1 ER Diagram

    4.3 ER to Relational Mapping (Schema Diagram)

    4.4 Normalization

    4.5 Tables and Constraints

5. Implementation

    5.1 Introduction

    5.2 DDL & DML Queries

    5.3 SQL Queries

    5.4 PL/SQL

6. Screenshot (front end-with explanation)

7. Conclusion and Future Work

    7.1 Conclusion

    7.2 Future Work

ONLINE MOVIE TICKET BOOKING SYSTEM

Team Member Registration Number	Name	Work Assigned
19BCE0631	JANESH PATEL	IMPLEMENTATION OF QUERIES(DDL,DML,SQL,PL/SQL)
19BCE2002	KARTIK GOEL	DESIGN OF PROJECT(ER SCHEMA,NORMALISATION)
19BCE0987	VANSH AGGARWAL	TABLES & CONSTRAINTS,CONCLUSION,FRONT-END

ABASTRACT

This project is aimed at developing an online movie ticket booking system website for customers.Online movie ticket booking system is a project developed for booking movie ticket online.This project saves lots of time and reduces the work of the customer.In online movie ticket booking system booking the movie ticket can be done from anywhere and at any time(24*7).some features provided to the users are new registration,login in,see movies by category,compare ticket price and timing,Customer can book ticket online without registration but if he/she registers then he/she will get different types of special offers,e-newsletters,movie updates and lots more The user can also cancel or update their order.

INTRODUCTION

Online movie ticket booking system is the process whereby consumer directly book the ticket online. online movie ticket booking system is the process of booking ticket from merchants who sell on internet.

Since the emergence of the World Wide Web, merchants have sought to sell their tickets to people who surf the internet. customer can visit web movie ticket theatre from the comfort of their home and book as they sit in front of the computer. Consumer book a verity of movies and relatedprice from online movie theatre. In fact, people can booked the movies just about anything from the companies that provide their tickets consumer can book from an online movie theatre.

Many people choose to conduct book online because of the convenience. For example, when a person book a ticket through movie theatre, he/she has to drive to the movie theatre find a parking place, and walk throughout the movie theatre until she locates the theatres he/she needs. Online movie ticket booking theatre very helpful or useful for the old people.

METHODOLOGY

The methodology that will be used throughout the development process will be variation on the Iterative waterfall life cycle. The basic idea behind this metod is to develop a system through repeated cycles (Iterative) and in smaller portions at a time (incremental), allowing software developers to take advantage of what was learned during development of earlier parts of versions of the system. Learning comes from both the development and use of the system, where possible key steps in the process start with a simple implementation of a subset of the software requirements and iteratively enhance the evolving versions until the full system is implemented.At each Iteration,design modifications are made and new functional capabilities are added. The goal for this initial implementation is to create a product to which the user can react. It includes such items as new features to be implemented and areas of redesign of the existing solution. The control list is constantly being revised as a result of the analysis phase. The analysis of iteration is based upon user feedback, and the program analysis facilities available.

FEASIBILITY STUDY

A feasibility study is carried out to select the best system that meets performance requirements .The main aim of the feasibility study activity is to determine whether it would be financially and technically feasible to develop the product. The feasibility study activity involves the analysis of the problem and collection of all relevant information relating to the product such as the different data items which would be input to the system, the processing required to be carried out on these data, the output data required to be produced by the system as well as various constraints on the behavior of the system.

Technical Feasibility:

This is concerned with specifying equipment and software that will successfully satisfy the user requirements. The technical needs of the system may vary considerably, but might include:

The facility to produce outputs in a given time.

Response time under certain conditions.

Ability to process a certain volume of transaction at a particular speed.

Facility to communicate data to distant locations.

In examining technical feasibility, configuration of the system is given more importance than the actual make of hardware. The configuration should give the complete picture about the system's requirements.

How many workstations are required, how these units are interconnected so that they could operate and communicate smoothly?

What speeds of input and output should be achieved at particular quality of printing.

Economic Feasibility:

Economic analysis is the most frequently used technique for evaluating the effectiveness of a proposed system. More commonly known as cost/Benefit analysis, the procedure is to determine the benefits and savings that are expected from a proposed system and compare them with costs. If benefits outweight costs, a decision is taken to design and implement the system. Otherwise, further justification or alternative in the proposed system will have to be made if it is to have a chance of being approved. This is an outgoing effort that improves in accuracy at each phase of the system life cycle.

Operational Feasibility:

This is mainly related to human organizational and political aspects. The points to be considered are:

What changes will be brought with the system?

What organizational structure is disturbed?

What new skills will be required? Do the existing staff members have these skills? If not, can they be trained in due course of time?

This feasibility study is carried out by a small group of people who are familiar with information system technique and are skilled in system analysis and design process.

Proposed projects are beneficial only if they can be turned into information system that will meet the operating requirements of the organization. This test of feasibility asks if the system will work when it is developed and installed.

PROJECT REQUIREMENTS

❖Hardware Requirement:

- Intel Pentium 3 processor or higher
- RAM 256 MB
- 40 GB HDD (Hard Disk)
- CD Drive 16X or higher

❖Software Requirement:

- Front-end Tool :- Microsoft Visual Studio 2012
- Back-end Tool :- Microsoft SQL Server 2008
- Web Browser :- Google Chrome
- Documentation :- MS Word 2007
- Project presentation :- MS PowerPoint 2007

❖Platform:

- Windows XP or later

Entity Relationship Diagram (ERD)

An entity-relationship model (ER model) describe inter-related things of interest in a specific domain of knowledge. An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is a component of data. In other words, ER diagram illustrate the logical structure of databases. At first glance an entity relationship diagram looks very much like a flowchart. It is the specialized symbols, and the meanings of those symbols, that make it unique.

Entity Relationship Diagram (ERD) Symbols:

- Entities ,which are represented by rectangles. An entity is an object or concept which you want to store information.

- Relationship, which are represented by diamond shapes, show how two entities share information in the database.

- A data flow symbol represents the data flow occurring between two processes or between an External entity and a process in the direction of the data flow arrow.

- Attributes, which are represented by ovals. A key attribute is the unique, distinguishing characteristic of the entity.

- A derived attribute is based on another attribute.

- A multi valued attribute can have more then one value.

- Cardinality specifies how many instances of an entity relate to one instance of another entity.

  - ✓ One to one – 1 : 1
  - ✓ One to many – 1 : M
  - ✓ Many to one – M : M
  - ✓ Many to one – M : 1
  - ✓ One or more – 1, n

ONLINE MOVIE TICKET BOOKING SYSTEM

Entity Relationship Diagram (ERD)

TESTING

TYPES OF TESTING:

- Unit testing
- Integration testing
- System testing
- Smoke testing
- Regression testing
- Beta testing
- Acceptance testing

1. UNIT TESTING:

It focuses on smallest unit of software design. In this we test an individual unit or group of inter related units. It is often done by programmer by using sample input and observing its corresponding outputs.

2. INTEGRATION TESTING:

Testing is a very important process which is done before implementation of any to make an accurate user friendly, client requirement satisfying software. Testing is a process of executing a program with the aim of finding error. To make our software perform well it should be error free. If testing is done

successfully it will remove all the errors from the software. The objective is to take unit tested components and build a program structure that has been dictated by design. Integration testing is testing in which a group of components are combined to produce output.

Integration testing is any type of software testing that seeks to verify the interfaces between components against a software design. Software components may be integrated in an iterative way or all together ("big bang"). Normally the former is considered a better practice since it allows interface issues to be located more quickly and fixed.

Integration testing works to expose defects in the interfaces and interaction between integrated components (modules). Progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a system.[42]

3. SYSTEM TESTING:

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black-box testing. and as such, should require no knowledge of the inner design of the code or logic.

4. SMOKE TESTING:

In software testing, smoke testing is preliminary testing to reveal simple failures severe enough to, for example, reject a prospective software release. Smoke tests are a subset of test cases that cover the most important functionality of a component or system, used to aid assessment if main functions of the software appear to work correctly. When used to determine if a computer program should be subjected to further, more fine-grained testing.

Alternately, it is a set of tests run on each new build of a product to verify that the build is testable before the build is released into the hands of the test team.

5. Regression testing:

Regression testing is the process of testing changes to computer programs to make sure that the older programming still works with the new changes. Regression testing is a normal part of the program development process and, in larger companies, is done by code testing specialists. Test department coders develop code test scenarios and exercises that will test new units of code after they have been written. These test cases form what becomes the test bucket. Before a new version of a software product is released, the old test cases are run against the new version to make sure that all the old capabilities still work. The reason they might not work is because changing or adding new code to a program can easily introduce errors into code that is not intended to be changed.

6. Beta testing:

Beta Testing is one of the Acceptance Testing types, which adds value to the product as the end-user (intended real user) validates the product for functionality, usability, reliability, and compatibility.

Since Beta Testing happens at the end user's side, it cannot be the controlled activity.

7. Acceptance testing:

Acceptance testing, a testing technique performed to determine whether or not the software system has met the requirement specifications. The main purpose of this test is to evaluate the system's compliance with the business requirements and verify if it is has met the required criteria for delivery to end users.

Tables and Constraints:

THEATRES		
ATTRIBUTES	**DATA-TYPES**	**CONSTRAINTS**
th_id	varchar(10)	PRIMARY KEY
t_name	varchar(50)	NOT NULL
locations	varchar(50)	NOT NULL
admin_id	varchar(50)	FOREIGN KEY

TICKETS		
ATTRIBUTES	**DATA-TYPES**	**CONSTRAINTS**
tid	varchar(10)	PRIMARY KEY
show_i	varchar(10)	NOT NULL
seat_no	varchar(10)	NOT NULL
show_date	DATETIME	NOT NULL
screen no	int	NOT NULL

Special_Screen		
ATTRIBUTES	**DATA-TYPES**	**CONSTRAINTS**
Ss_id	varchar(10)	PRIMARY KEY
type	char(10)	NOT NULL
Ss_name	varchar(50)	NOT NULL
s_id	varchar(10)	FOREIGN KEY

SHOW		
ATTRIBUTES	**DATA-TYPES**	**CONSTRAINTS**
s_id	varchar(10)	PRIMARY KEY
Lang	varchar(20)	NOT NULL
st_time	DATETIME	NOT NULL
end_time	DATETIME	NOT NULL
m_id	varchar(10)	FOREIGN KEY

ADMIN		
ATTRIBUTES	**DATA-TYPES**	**CONSTRAINTS**
admin_id	char(10)	PRIMARY KEY
password	varchar(50)	NOT NULL,CHECK

DISCOUNTS		
ATTRIBUTES	**DATA-TYPES**	**CONSTRAINTS**
offer_id	varchar(10)	PRIMARY KEY
m_name	varchar(50)	FOREIGN KEY
PRICE	DOUBLE	NOT NULL

CUSTOMER		
ATTRIBUTES	**DATA-TYPES**	**CONSTRAINTS**
c_idd	varchar(10)	PRIMARY KEY
f_name	varchar(50)	NOT NULL
l_name	varchar(50)	NOT NULL
password	varchar(50)	NOT NULL
gender	char(1)	CHECK
DOB	DATE	NOT NULL
r_id	varchar(10)	FOREIGN KEY
email_id	varchar(50)	NOT NULL

RECEIPT		
ATTRIBUTES	**DATA-TYPES**	**CONSTRAINTS**
r_id	varchar(10)	PRIMARY KEY
Date	DATETIME	NOT NULL
amount	DOUBLE	NOT NULL

MOVIE		
ATTRIBUTES	**DATA-TYPES**	**CONSTRAINTS**
m_id	varchar(10)	PRIMARY KEY
duration	time	NOT NULL
Rating	FLOAT	NOT NULL
release_date	DATETIME	NOT NULL
m_name	varchar(50)	NOT NULL

SEATS		
ATTRIBUTES	**DATA-TYPES**	**CONSTRAINTS**
seat_id	varchar(10)	PRIMARY KEY
seat_name	varchar(10)	NOT NULL
No.of seats	INT	NOT NULL

RELATIONAL SCHEMA DIAGRAM

ADMIN

```sql
CREATE TABLE Admin
(

  Admin_id VARCHAR2(30) NOT NULL,
  Password VARCHAR2(50) NOT NULL,
  PRIMARY KEY (Admin_id)
  CHECK (LENGTH(password) >= 8)

);


INSERT INTO Admin VALUES ('JANESH007','0ABCDEFGH$');
INSERT INTO Admin VALUES ('ANY007','@1XYZABC0');
INSERT INTO Admin VALUES ('SDADS0','12345678');
INSERT INTO Admin VALUES ('HIHELLO8','ABCD2132');
INSERT INTO Admin VALUES ('DBMS400','1XYZABC0');
SELECT * FROM Admin;
```

```
JANESH007|0ABCDEFGH$
ANY007|@1XYZABC0
SDADS0|12345678
HIHELLO8|ABCD2132
DBMS400|1XYZABC0
```

CUSTOMER

```sql
CREATE TABLE Customer
(
  email VARCHAR(50) NOT NULL,
  C_iid INT NOT NULL,
  f_name VARCHAR(50) NOT NULL,
  l_name VARCHAR(50) NOT NULL,
  password VARCHAR(50) NOT NULL,
  gender CHAR(1) NOT NULL,
  DOB DATE NOT NULL,
  p_id VARCHAR(10) NOT NULL,
  PRIMARY KEY (C_iid),
  FOREIGN KEY (p_id) REFERENCES payment(p_id)
  CHECK (LENGTH(password) >= 8)
  check(gender in ('male', 'female', 'unknown'))
);

INSERT INTO Customer VALUES ('janesh@gmail.com','001','Janesh','Patel','JAN12345678','male','05-AUG-2001','F001JAN');
INSERT INTO Customer VALUES ('narendra@gmail.com','002','Narendra','Modi','NAR12345678','male','05-AUG-1980','F002NAR');
INSERT INTO Customer VALUES ('kartik@gmail.com','003','Kartik','Goel','KAR12345678','male','06-JAN-2001','F003KAR');
INSERT INTO Customer VALUES ('hello@gmail.com','004','Hello','Kapoor','HEL12345678','female','15-AUG-1991','F004HEL');
INSERT INTO Customer VALUES ('yes@gmail.com','005','Yes','Pal','YES12345678','male','05-SEP-2000','F005YES');

SELECT * FROM Customer;
```

input

```
janesh@gmail.com|1|Janesh|Patel|JAN12345678|male|05-AUG-2001|F001JAN
narendra@gmail.com|2|Narendra|Modi|NAR12345678|male|05-AUG-1980|F002NAR
kartik@gmail.com|3|Kartik|Goel|KAR12345678|male|06-JAN-2001|F003KAR
hello@gmail.com|4|Hello|Kapoor|HEL12345678|female|15-AUG-1991|F004HEL
yes@gmail.com|5|Yes|Pal|YES12345678|male|05-SEP-2000|F005YES
```

MOVIES

```sql
CREATE TABLE Movies
(
  m_id VARCHAR(10) NOT NULL,
  mname VARCHAR(50) NOT NULL,
  Release_date DATE NOT NULL,
  Review FLOAT NOT NULL,
  PRIMARY KEY (m_id)
);

INSERT INTO Movies VALUES ('001','KICK','31-AUG-2020','6.2');
INSERT INTO Movies VALUES ('002','KABALI','30-AUG-2020','8.5');
INSERT INTO Movies VALUES ('003','LOCKOUT','29-AUG-2020','7.1');
INSERT INTO Movies VALUES ('004','DEADPOOL','28-AUG-2020','8.4');
INSERT INTO Movies VALUES ('005','RAID','27-AUG-2020','8.1');

SELECT * FROM Movies;
```

```
001|KICK|31-AUG-2020|6.2
002|KABALI|30-AUG-2020|8.5
003|LOCKOUT|29-AUG-2020|7.1
004|DEADPOOL|28-AUG-2020|8.4
005|RAID|27-AUG-2020|8.1
```

SEATS

```sql
CREATE TABLE Seats
(
  Seat_id VARCHAR(10) NOT NULL,
  Seat_name VARCHAR(10) NOT NULL,
  No_of_seat INT NOT NULL,
  PRIMARY KEY (Seat_id)
);

INSERT INTO Seats VALUES ('11F','CORNER GOLD','1');
INSERT INTO Seats VALUES ('12F','CORNER GOLD','1');
INSERT INTO Seats VALUES ('13F','MIDDLE GOLD','1');
INSERT INTO Seats VALUES ('14F','MIDDLE GOLD','1');
INSERT INTO Seats VALUES ('15F','MIDDLE GOLD','1');


SELECT * FROM Seats;
```

```
11F|CORNER GOLD|1
12F|CORNER GOLD|1
13F|MIDDLE GOLD|1
14F|MIDDLE GOLD|1
15F|MIDDLE GOLD|1
```

DISCOUNT

```sql
CREATE TABLE Discount
(
  offer_id INT NOT NULL,
  M_name VARCHAR(50) NOT NULL,
  Price FLOAT NOT NULL,
  PRIMARY KEY (offer_id)
);

INSERT INTO Discount VALUES ('1001','KICK','100');
INSERT INTO Discount VALUES ('2001','KABALI','100');
INSERT INTO Discount VALUES ('3001','LOCKOUT','100');
INSERT INTO Discount VALUES ('4001','DEADPOOL','100');
INSERT INTO Discount VALUES ('5001','RAID','100');

SELECT * FROM Discount;
```

```
1001|KICK|100.0
2001|KABALI|100.0
3001|LOCKOUT|100.0
4001|DEADPOOL|100.0
5001|RAID|100.0
```

RECEIPT

```sql
CREATE TABLE Receipt

(
  r_id INT NOT NULL,
  Date DATE NOT NULL,
  Amount INT NOT NULL,
  PRIMARY KEY (r_id)

);

INSERT INTO Receipt VALUES ('123001','31-AUG-2020','400')
INSERT INTO Receipt VALUES ('123002','31-AUG-2020','400')
INSERT INTO Receipt VALUES ('123003','31-AUG-2020','400')
INSERT INTO Receipt VALUES ('123004','31-AUG-2020','400')
INSERT INTO Receipt VALUES ('123005','31-AUG-2020','400')



SELECT * FROM Receipt;
```

```
123001|31-AUG-2020|400
123002|31-AUG-2020|400
123003|31-AUG-2020|400
123004|31-AUG-2020|400
123005|31-AUG-2020|400
```

SHOWS

```sql
CREATE TABLE Show
(
  Lang VARCHAR(20) NOT NULL,
  s_id VARCHAR(10) NOT NULL,
  st_time DATE NOT NULL,
  end_time DATE NOT NULL,
  m_id VARCHAR(10) NOT NULL,
  PRIMARY KEY (s_id),
  FOREIGN KEY (m_id) REFERENCES Movies(m_id)
);

INSERT INTO Show VALUES ('HINDI','12AM001','00:00','2:40','001');
INSERT INTO Show VALUES ('TAMIL','12AM002','00:00','2:25','002');
INSERT INTO Show VALUES ('ENGLISH','12AM003','00:00','3;00','003');
INSERT INTO Show VALUES ('ENGLISH','12AM004','00:00','2:35','004');
INSERT INTO Show VALUES ('HINDI','12AM005','00:00','2:45','005');



SELECT * FROM Show;
```

```
HINDI|12AM001|00:00|2:40|001
TAMIL|12AM002|00:00|2:25|002
ENGLISH|12AM003|00:00|3;00|003
ENGLISH|12AM004|00:00|2:35|004
HINDI|12AM005|00:00|2:45|005
```

SPECIAL SCREENING

```sql
CREATE TABLE Special_screen
(
  Ss_type CHAR(10) NOT NULL,
  Ss_id INT NOT NULL,
  Ss_name VARCHAR(50) NOT NULL,
  s_id VARCHAR(10) NOT NULL,
  PRIMARY KEY (Ss_id),
  FOREIGN KEY (s_id) REFERENCES Show(s_id)
);

INSERT INTO Special_screen VALUES ('LIVE MATCH','001','CRICKET FINAL','12PM001');
INSERT INTO Special_screen VALUES ('STANDUP','002','COMEDY STANDUP','12PM002');
INSERT INTO Special_screen VALUES ('SEMINAR','003','EDUCATION','12PM003');
INSERT INTO Special_screen VALUES ('MUSIC','004','CARDI B','12PM004');
INSERT INTO Special_screen VALUES ('DANCE','005','MJ 5','12PM005');



SELECT * FROM Special_screen;
```

input

```
LIVE MATCH|1|CRICKET FINAL|12PM001
STANDUP|2|COMEDY STANDUP|12PM002
SEMINAR|3|EDUCATION|12PM003
MUSIC|4|CARDI B|12PM004
DANCE|5|MJ 5|12PM005
```

THEATRE

```sql
CREATE TABLE Theatre
(
  th_id VARCHAR(10) NOT NULL,
  locations VARCHAR(50) NOT NULL,
  t_name VARCHAR(50) NOT NULL,
  Admin_id VARCHAR(30) NOT NULL,
  PRIMARY KEY (th_id),
  FOREIGN KEY (Admin_id) REFERENCES Admin(Admin_id)
);

INSERT INTO Theatre VALUES ('001','MG HIGHWAY','SUN MOVIES','JANESH007');
INSERT INTO Theatre VALUES ('002','SP HIGHWAY','PVR CINEMA','ANY007');
INSERT INTO Theatre VALUES ('003','GOLDEN PALACE','SUR MANDIR','SDADS0');
INSERT INTO Theatre VALUES ('004','DMART','BIG CINEMAS','HIHELLO8');
INSERT INTO Theatre VALUES ('005','INORBIT','VIJAY PICTURES','DBMS400');


SELECT * FROM Theatre;
```

input

```
001|MG HIGHWAY|SUN MOVIES|JANESH007
002|SP HIGHWAY|PVR CINEMA|ANY007
003|GOLDEN PALACE|SUR MANDIR|SDADS0
004|DMART|BIG CINEMAS|HIHELLO8
005|INORBIT|VIJAY PICTURES|DBMS400
```

TICKETS

```sql
CREATE TABLE Tickets
(
  t_id VARCHAR2(10) NOT NULL,
  show_id VARCHAR2(10) NOT NULL,
  seat_no VARCHAR2(10) NOT NULL,
  show_date DATE NOT NULL,
  screen_no INT NOT NULL,
  PRIMARY KEY (t_id)
);

INSERT INTO Tickets VALUES ('120A','12AM','11F','31-aug-2020','1');
INSERT INTO Tickets VALUES ('121A','12AM','12F','31-aug-2020','1');
INSERT INTO Tickets VALUES ('122A','12AM','13F','31-aug-2020','1');
INSERT INTO Tickets VALUES ('123A','12AM','14F','31-aug-2020','1');
INSERT INTO Tickets VALUES ('124A','12AM','15F','31-aug-2020','1');

SELECT * FROM Tickets;
```

input

```
120A|12AM|11F|31-aug-2020|1
121A|12AM|12F|31-aug-2020|1
122A|12AM|13F|31-aug-2020|1
123A|12AM|14F|31-aug-2020|1
124A|12AM|15F|31-aug-2020|1
```

CODES:

ADMIN

```
CREATE TABLE Admin
(

  Admin_id VARCHAR2(30) NOT NULL,
  Password VARCHAR2(50) NOT NULL,
  PRIMARY KEY (Admin_id)
  CHECK (LENGTH(password) >= 8)

);


INSERT INTO Admin VALUES ('JANESH007','0ABCDEFGH$');
INSERT INTO Admin VALUES ('ANY007','@1XYZABC0');
INSERT INTO Admin VALUES ('SDADS0','12345678');
INSERT INTO Admin VALUES ('HIHELLO8','ABCD2132');
INSERT INTO Admin VALUES ('DBMS400','1XYZABC0');


SELECT * FROM Admin;
```

CUSTOMER

```
CREATE TABLE Customer
(
  email VARCHAR(50) NOT NULL,
  C_iid INT NOT NULL,
  f_name VARCHAR(50) NOT NULL,
  l_name VARCHAR(50) NOT NULL,
  password VARCHAR(50) NOT NULL,
  gender CHAR(1) NOT NULL,
  DOB DATE NOT NULL,
  r_id VARCHAR(10) NOT NULL,
  PRIMARY KEY (C_iid),
  FOREIGN KEY (r_id) REFERENCES receipt(r_id)
 CHECK (LENGTH(password) >= 8)
  check(gender in ('male', 'female', 'unknown'))
);

INSERT INTO Customer VALUES
('janesh@gmail.com','001','Janesh','Patel','JAN12345678','male','05-AUG-
2001','F001JAN');
INSERT INTO Customer VALUES
('narendra@gmail.com','002','Narendra','Modi','NAR12345678','male','05-AUG-
1980','F002NAR');
INSERT INTO Customer VALUES
('kartik@gmail.com','003','Kartik','Goel','KAR12345678','male','06-JAN-
2001','F003KAR');
INSERT INTO Customer VALUES
('hello@gmail.com','004','Hello','Kapoor','HEL12345678','female','15-AUG-
1991','F004HEL');
INSERT INTO Customer VALUES
('yes@gmail.com','005','Yes','Pal','YES12345678','male','05-SEP-2000','F005YES');

SELECT * FROM Customer;
```

DISCOUNT

```
CREATE TABLE Discount
(
  offer_id INT NOT NULL,
  M_name VARCHAR(50) NOT NULL,
  Price FLOAT NOT NULL,
  PRIMARY KEY (offer_id)
);

INSERT INTO Discount VALUES ('1001','KICK','100');
INSERT INTO Discount VALUES ('2001','KABALI','100');
INSERT INTO Discount VALUES ('3001','LOCKOUT','100');
INSERT INTO Discount VALUES ('4001','DEADPOOL','100');
INSERT INTO Discount VALUES ('5001','RAID','100');

SELECT * FROM Discount;
```

MOVIES

```
CREATE TABLE Movies
(
  m_id VARCHAR(10) NOT NULL,
  mname VARCHAR(50) NOT NULL,
  Release_date DATE NOT NULL,
  Review FLOAT NOT NULL,
  PRIMARY KEY (m_id)
);

INSERT INTO Movies VALUES ('001','KICK','31-AUG-2020','6.2');
INSERT INTO Movies VALUES ('002','KABALI','30-AUG-2020','8.5');
INSERT INTO Movies VALUES ('003','LOCKOUT','29-AUG-2020','7.1');
INSERT INTO Movies VALUES ('004','DEADPOOL','28-AUG-2020','8.4');
INSERT INTO Movies VALUES ('005','RAID','27-AUG-2020','8.1');

SELECT  FROM Movies;
```

RECEIPT

```
CREATE TABLE Receipt

(
  r_id INT NOT NULL,
  Date DATE NOT NULL,
  Amount INT NOT NULL,
  PRIMARY KEY (r_id)

);

INSERT INTO Receipt VALUES ('123001','31-AUG-2020','400');
INSERT INTO Receipt VALUES ('123002','31-AUG-2020','400');
INSERT INTO Receipt VALUES ('123003','31-AUG-2020','400');
INSERT INTO Receipt VALUES ('123004','31-AUG-2020','400');
INSERT INTO Receipt VALUES ('123005','31-AUG-2020','400');


SELECT * FROM Receipt;
```

SEATS

```
CREATE TABLE Seats
(
  Seat_id VARCHAR(10) NOT NULL,
  Seat_name VARCHAR(10) NOT NULL,
  No_of_seat INT NOT NULL,
  PRIMARY KEY (Seat_id)
);

INSERT INTO Seats VALUES ('11F','CORNER GOLD','1');
INSERT INTO Seats VALUES ('12F','CORNER GOLD','1');
INSERT INTO Seats VALUES ('13F','MIDDLE GOLD','1');
INSERT INTO Seats VALUES ('14F','MIDDLE GOLD','1');
INSERT INTO Seats VALUES ('15F','MIDDLE GOLD','1');


SELECT * FROM Seats;
```

SHOWS

```
CREATE TABLE Show
(
  Lang VARCHAR(20) NOT NULL,
  s_id VARCHAR(10) NOT NULL,
  st_time DATE NOT NULL,
  end_time DATE NOT NULL,
  m_id VARCHAR(10) NOT NULL,
  PRIMARY KEY (s_id),
  FOREIGN KEY (m_id) REFERENCES Movies(m_id)
);

INSERT INTO Show VALUES ('HINDI','12AM001','00:00','2:40','001');
INSERT INTO Show VALUES ('TAMIL','12AM002','00:00','2:25','002');
INSERT INTO Show VALUES ('ENGLISH','12AM003','00:00','3;00','003');
INSERT INTO Show VALUES ('ENGLISH','12AM004','00:00','2:35','004');
INSERT INTO Show VALUES ('HINDI','12AM005','00:00','2:45','005');


SELECT * FROM Show;
```

SPECIAL SCREENING

```
CREATE TABLE Special_screen
(
  Ss_type CHAR(10) NOT NULL,
  Ss_id INT NOT NULL,
  Ss_name VARCHAR(50) NOT NULL,
  s_id VARCHAR(10) NOT NULL,
  PRIMARY KEY (Ss_id),
  FOREIGN KEY (s_id) REFERENCES Show(s_id)
);

INSERT INTO Special_screen VALUES ('LIVE MATCH','001','CRICKET
FINAL','12PM001');
INSERT INTO Special_screen VALUES ('STANDUP','002','COMEDY
STANDUP','12PM002');
INSERT INTO Special_screen VALUES ('SEMINAR','003','EDUCATION','12PM003');
INSERT INTO Special_screen VALUES ('MUSIC','004','CARDI B','12PM004');
INSERT INTO Special_screen VALUES ('DANCE','005','MJ 5','12PM005');


SELECT * FROM Special_screen;
```

THEATRE

```
CREATE TABLE Theatre
(
  th_id VARCHAR(10) NOT NULL,
  locations VARCHAR(50) NOT NULL,
  t_name VARCHAR(50) NOT NULL,
  Admin_id VARCHAR(30) NOT NULL,
  PRIMARY KEY (th_id),
  FOREIGN KEY (Admin_id) REFERENCES Admin(Admin_id)
);

INSERT INTO Theatre VALUES ('001','MG HIGHWAY','SUN
MOVIES','JANESH007');
INSERT INTO Theatre VALUES ('002','SP HIGHWAY','PVR CINEMA','ANY007');
INSERT INTO Theatre VALUES ('003','GOLDEN PALACE','SUR
MANDIR','SDADS0');
INSERT INTO Theatre VALUES ('004','DMART','BIG CINEMAS','HIHELLO8');
INSERT INTO Theatre VALUES ('005','INORBIT','VIJAY PICTURES','DBMS400');


SELECT * FROM Theatre;
```

TICKETS

```
CREATE TABLE Tickets
(
  t_id VARCHAR2(10) NOT NULL,
  show_id VARCHAR2(10) NOT NULL,
  seat_no VARCHAR2(10) NOT NULL,
  show_date DATE NOT NULL,
  screen_no INT NOT NULL,
  PRIMARY KEY (t_id)
);

INSERT INTO Tickets VALUES ('120A','12AM','11F','31-aug-2020','1');
INSERT INTO Tickets VALUES ('121A','12AM','12F','31-aug-2020','1');
INSERT INTO Tickets VALUES ('122A','12AM','13F','31-aug-2020','1');
INSERT INTO Tickets VALUES ('123A','12AM','14F','31-aug-2020','1');
INSERT INTO Tickets VALUES ('124A','12AM','15F','31-aug-2020','1');

SELECT * FROM Tickets;
```

NORMALIZATION

First Normal Form (1NF)

- As per the rule of first normal form, an attribute (column) of a table cannot hold multiple values. It should hold only atomic values.
- So here tables were in 1NF already.

Second Normal Form (2NF)

- A table is said to be in 2NF if both the following conditions hold:
- Table is in 1NF (First normal form)
- No non-prime attribute is dependent on the proper subset of any candidate key of table.
- An attribute that is not part of any candidate key is known as non-prime attribute.
- So here tables were in 2NF already.

Third Normal Form (3NF)

- A table design is said to be in 3NF if both the following conditions hold:
- Table must be in 2NF
- Transitive functional dependency of non-prime attribute on any super key should be removed.
- An attribute that is not part of any candidate key is known as non-prime attribute.
- Here tables were already in 3NF.

Boyce Codd Normal Form (BCNF)

- It is an advance version of 3NF that's why it is also referred as 3.5NF. BCNF is stricter than 3NF. A table complies with BCNF if it is in 3NF and for every functional dependency X->Y, X should be the super key of the table.
- So here tables were already in BCNF.

Fourth Normal Form (4NF)

- Fourth Normal Form is related to Multi-value Dependency. Under fourth normal form, a record type should not contain two or more independent multi-value facts about an entity. In addition the record must satisfy third normal form.
- A multi-value dependency exists when
- There are at least three attributes A, B and C in a relation.
- For each value of A there is a well-defined set of values for B, and a well-defined set of values for C.
- The set of values of B is independent of set C.
- If a table in 4NF then -
- All attributes must be dependent on the primary key, but they must be independent of each other.
- No row may contain two or more multivalued facts about an entity.

Fifth Normal Form (5NF)

- The 5NF (Fifth Normal Form) is also known as project-join normal form. A relation is in Fifth Normal Form (5NF), if it is in 4NF, and won't have lossless decomposition into smaller tables.
- You can also consider that a relation is in 5NF, if the candidate key implies every join dependency in it.

NORMALIZED TABLES

NORMALIZATION

1) Customer

C_id	e_mail	password	c_name	Dob	gender

f_name L_name

Since c_name is a composite attribute therefore it is not in 1NF

gender	C_id	e_mail	password	f_name	L_name	Dob

Functional Dependencies are

C_id → gender, e_mail; password, f_name, L_name, Dob

Since there is no transitive & partial dependency

∴ it is ~~not in~~ in 2NF, 3NF

Since there is no dependency that derives the superkey ∴ it is in BCNF

② Tickets

t_id	theatre_id	offer_id	show_id	Customer_id	seat_id
	(f.k)	(f.k)	(f.k)	(f.k)	(f.k)

t_id → theatre_id, offer_id, show_id, customer_id, seat_id

Here t_id identifies theatre id, offer_id, show_id Customer id & seat_id

∴ it satisfies 1NF, 2NF, 3NF, BCNF as there is no partial dependencies, transitive dependencies and neither of attribute uniquely identifies superkey

③ Seat

Seat_id	theatre_id (f.k)	seat_type	seat_no

functional dependencies

seat_id → theatre_id seat_type seat_no

Here Seat_id uniquely identifies all the attributes but there is no partial, transitive dependencies present here

∴ it is in 1NF, 2NF, 3NF & BCNF

(4) Payment

P_id	ticket_id	amount	date
(p.k)	(f.k)		

P_id → ticket_id, amount, date

No ~~fu~~ partial or transitive dependency present ∴ it is in 1NF, 2NF, 3NF, BCNF

(5) Theatre.

Theatre_id (p.k)	th_name	th_type	city	pincode	total_seats

functional dependencies

Theatre_id → th_name, th_type, city, pincode, total_seats

pincode → city

Here transitive dependency present ∴ it is in not in 3NF form
We will split the tables

Theatre

theatre_id	th_name	th_type	city	Pincode	total_seats

(P.K) (F.K)

Locations

pincode	city

(P.K)

6) Show

Show_id	theatre_id	language	st_time	end_time

Show_id → theatre_id, language, st_time, end_time

No partial & ~~fu~~ transitive dependency present
∴ it is in 1NF, 2NF, 3NF, BCNF

(7)

Movies

m_id	m_name	release date	show_id	rating
			(f.k)	

m_id → m_name, release date, show_id, rating

no transitive & functional dependency present here

∴ it is in 1NF, 2NF, 3NF & BCNF

(8)

Watch

c_idd	m_id
(f.k)	(f.k)

it is in 1NF, 2NF, 3NF & BCNF form

hasa

9)

th_id	m_id
(f.k)	(f.k)

it is in 1NF, 2NF, 3NF & BCN form

10) Discount

offer_id	discount
(P.K)	

offer_id → discount

∴ it is in 1NF, 2NF, 3NF & BCNF

11) offers_available

offer_id	t_id
(F.K)	(F.K)

∴ it is in 1NF, 2NF, 3NF & BCNF

12) Special_Screening

ss_id	ss_name	ss_type	show_id
			(F.K)

ss_id → ss_name, ss_type, show_id

∴ it follows 1NF, 2NF, 3NF & BCNF form

Before Normalization

1) Tickets
2) Theatres
3) movies
4) show
5) seats
6) payments
7) customer
8) Discount
9) special_screen
10) watch
11) has_a
12) offer_available

After normalization

1) Tickets
2) Theatre
3) locations (derive from Theatre)
4) show
5) seats
6) payments
7) customer
8) Discount
9) special screen
10) watch
11) has_a
12) offer_available

THEATRES		
ATTRIBUTES	DATA-TYPES	CONSTRAINTS
th_id	varchar(10)	PRIMARY KEY
t_name	varchar(50)	NOT NULL
locations	varchar(50)	NOT NULL
admin_id	varchar(50)	FOREIGN KEY

TICKETS		
ATTRIBUTES	DATA-TYPES	CONSTRAINTS
tid	varchar(10)	PRIMARY KEY
show_i	varchar(10)	NOT NULL
seat_no	varchar(10)	NOT NULL
show_date	DATETIME	NOT NULL
screen no	int	NOT NULL

Special_Screen		
ATTRIBUTES	**DATA-TYPES**	**CONSTRAINTS**
Ss_id	varchar(10)	PRIMARY KEY
type	char(10)	NOT NULL
Ss_name	varchar(50)	NOT NULL
s_id	varchar(10)	FOREIGN KEY

SHOW		
ATTRIBUTES	**DATA-TYPES**	**CONSTRAINTS**
s_id	varchar(10)	PRIMARY KEY
Lang	varchar(20)	NOT NULL
st_time	DATETIME	NOT NULL
end_time	DATETIME	NOT NULL
m_id	varchar(10)	FOREIGN KEY

ADMIN		
ATTRIBUTES	**DATA-TYPES**	**CONSTRAINTS**
admin_id	char(10)	PRIMARY KEY
password	varchar(50)	NOT NULL,CHECK

DISCOUNTS		
ATTRIBUTES	**DATA-TYPES**	**CONSTRAINTS**
offer_id	varchar(10)	PRIMARY KEY
m_name	varchar(50)	FOREIGN KEY
PRICE	DOUBLE	NOT NULL

CUSTOMER		
ATTRIBUTES	DATA-TYPES	CONSTRAINTS
c_idd	varchar(10)	PRIMARY KEY
f_name	varchar(50)	NOT NULL
l_name	varchar(50)	NOT NULL
password	varchar(50)	NOT NULL
gender	char(1)	CHECK
DOB	DATE	NOT NULL
p_id	varchar(10)	FOREIGN KEY
email_id	varchar(50)	NOT NULL

RECEIPT		
ATTRIBUTES	DATA-TYPES	CONSTRAINTS
r_id	varchar(10)	PRIMARY KEY
Date	DATETIME	NOT NULL
amount	DOUBLE	NOT NULL

MOVIE		
ATTRIBUTES	DATA-TYPES	CONSTRAINTS
m_id	varchar(10)	PRIMARY KEY
duration	time	NOT NULL
Rating	FLOAT	NOT NULL
release_date	DATETIME	NOT NULL
m_name	varchar(50)	NOT NULL

SEATS		
ATTRIBUTES	DATA-TYPES	CONSTRAINTS
seat_id	varchar(10)	PRIMARY KEY
seat_name	varchar(10)	NOT NULL
No.of seats	INT	NOT NULL

WATCHES		
ATTRIBUTES	DATA-TYPES	CONSTRAINTS
C_id	varchar(10)	FOREIGN KEY
m_id	varchar(10)	FOREIGN KEY

HAS A		
ATTRIBUTES	DATA-TYPES	CONSTRAINTS
th_id	varchar(10)	FOREIGN KEY
m_id	varchar(10)	FOREIGN KEY

QUERIES

1.FIND THE DETAILS OF ALL CUSTOMERS.

```
select*from Customer
```

	email	C_iid	f_name	l_name	password	gender	DOB	p_id
1	janesh@gmail.com	1	JANESH	PATEL	jn@1234	m	1995-11-04	F001JAN
2	narendra@gmail.com	2	Narendra	Modi	nrab123	m	1999-08-21	F002JAN
3	kg@gmail.com	3	KARTIK	Goel	kr@1234	m	2000-11-16	FOO3JAN
4	vani@gmail.com	4	Vani	Kapoor	vank1234	f	1996-02-12	F004JAN
5	ps@gmail.com	5	Payal	sharma	ps@1234	f	1994-03-12	FOO5JAN

2.FIND THE DETAILS OF MOVIES.

```
select*from Movies
```

Results | Messages

	m_id	mname	Release_date	Review
1	001	KICK	2020-08-31	6.2
2	002	KABALI	2020-08-30	8.5
3	003	LOCKOUT	2020-08-30	7.1
4	004	DEADPOOL	2020-08-27	8.4
5	005	RAID	2020-08-27	8.1

3.LIST THE MOVIES IN LANGUAGE HINDI AND ENGLISH.

```
select Movies.m_id,Movies.mname from  Movies JOIN Show on Movies.m_id=Show.m_id
where Show.Lang='HINDI' OR Show.Lang='ENGLISH';
```

Results | Messages

	m_id	mname
1	001	KICK
2	003	LOCKOUT
3	004	DEADPOOL
4	005	RAID

4.FIND THE CUSTOMER HAVING DOB : 21-08-1999

```
select*from Customer where DOB='21-AUG-1999'
```

Results | Messages

	email	C_iid	f_name	l_name	password	gender	DOB	p_id
1	narendra@gmail.com	2	Narendra	Modi	nrab123	m	1999-08-21	F002JAN

5.FIND THE NAME OF THE CUSTOMERS WHO HAS PAID AMOUNT IN BETWEEN 200 TO 500.

```
select*from payment where amt between 200 and 500
```

Results | Messages

	ticket_id	p_id	amt	dates
1	T101	F001JAN	200	2001-08-05 00:00:00.000
2	T102	F002JAN	300	2019-11-07 00:00:00.000
3	T103	FOO3JAN	500	2020-12-08 00:00:00.000

6.FIND THE TICKET DETAILS FOR THE T_id =“T101”.

```
select*from Tickets where t_id='T101'
```

Results | Messages

	t_id	show_id	seat_no	show_date	screen_no
1	T101	12AM001	200	2001-08-05	1

7.DISPLAY THE CUSTOMER DETAILS WITH GENDER= “f”.

```
select*from Customer where gender='f'
```

Results | Messages

	email	C_iid	f_name	l_name	password	gender	DOB	p_id
1	vani@gmail.com	4	Vani	Kapoor	vank1234	f	1996-02-12	F004JAN
2	ps@gmail.com	5	Payal	sharma	ps@1234	f	1994-03-12	FOO5JAN

8.FIND THE C-id WHOSE GENDER IS NOT FEMALE AND EMAIL = narendra@gmail.com .

```
select*from Customer where gender!='f' and email='narendra@gmail.com'
```

Results | Messages

	email	C_iid	f_name	l_name	password	gender	DOB	p_id
1	narendra@gmail.com	2	Narendra	Modi	nrab123	m	1999-08-21	F002JAN

9. FIND THE PAYMENT DETAILS WHICH ARE MADE ON DATE 12-08-2020.

```
select*from payment where dates='12-08-2020'
```

Results | Messages

	ticket_id	p_id	amt	dates
1	T103	FOO3JAN	500	2020-12-08 00:00:00.000

10.DELETE THE MOVIE NAMES WHICH ARE IN TAMIL LANGUAGE.

```
delete from Show WHERE Lang='TAMIL'
```

	Lang	s_id	st_time	end_time	m_id
1	HINDI	12AM001	12:00:00.0000000	02:40:00.0000000	001
2	ENGLISH	12AM003	00:00:00.0000000	00:00:00.0000000	003
3	ENGLISH	12AM004	00:00:00.0000000	00:00:00.0000000	004
4	HINDI	12AM005	00:00:00.0000000	00:00:00.0000000	005

11.CHANGE THE MOVIE NAME = “KNIVES OUT” FOR THE MOVIE ID= “001”

```
update Movies set mname='KNIVES OUT' where m_id=001
```

	m_id	mname	Release_date	Review
1	001	KNIVES OUT	2020-08-31	6.2
2	002	KABALI	2020-08-30	8.5
3	003	LOCKOUT	2020-08-30	7.1
4	004	DEADPOOL	2020-08-27	8.4
5	005	RAID	2020-08-27	8.1

12.DELETE THE THEATRE NAME WHERE LOCATION= “INORBIT”.

```
delete from Theatre where locations='INORBIT'
```

Results | Messages

	th_id	locations	t_name	Admin_id
1	001	MG HIGHWAY	SUNMOVIES	JANESH007
2	002	SP HIGHWAY	PVR CINEMA	ANY007
3	003	GOLDEN PALACE	SURMANDIR	SDADS0
4	004	DMART	BIG CINEMAS	HIHELLO8

13.CHANGE THE AMOUNT TO 500 FOR PAYMENT ON DATE 12-08-2020.

```
update payment set amt=1000 where dates='12-08-2020'
```

	ticket_id	p_id	amt	dates
1	T101	F001JAN	200	2001-08-05 00:00:00.000
2	T104	F004JAN	700	2016-01-03 00:00:00.000
3	T103	FOO3JAN	1000	2020-12-08 00:00:00.000
4	T105	FOO5JAN	900	2019-11-07 00:00:00.000

14.MODIFY THE START_TIME AS “09:00” AND END_TIME AS “12:30” FOR MOVIES WITH LANGUAGE ENGLISH.

```
update Show set st_time='9:00',end_time='12:30' where Lang='ENGLISH'
```

Results | Messages

	Lang	s_id	st_time	end_time	m_id
1	HINDI	12AM001	12:00:00.0000000	02:40:00.0000000	001
2	ENGLISH	12AM003	09:00:00.0000000	12:30:00.0000000	003
3	ENGLISH	12AM004	09:00:00.0000000	12:30:00.0000000	004
4	HINDI	12AM005	00:00:00.0000000	00:00:00.0000000	005

15.DELETE THE T-id WITH SHOW DATE = “12-08-2020”.

```
delete from Tickets where show_id='12-08-2020'
```

Results | Messages

	t_id	show_id	seat_no	show_date	screen_no
1	T101	12AM001	200	2001-08-05	1
2	T103	12AM003	500	2020-12-08	3
3	T104	12AM004	700	2016-01-03	4
4	T105	12AM005	900	2019-11-07	2

16.FIND WHICH MOVIE IS BEING WATCHED BY WHICH CUSTOMER.

<u>ANS</u> : Select Movies.mname,Customer.f_name,Customer.l_name from Movies JOIN watch on watch.m_id=Movies.m_id JOIN Customer on watch.C_idd=Customer.C_iid

<u>OUTPUT</u> :

RESULTS

	mname	f_name	l_name
1	KICK	JANESH	PATEL
2	KICK	Narendra	Modi
3	KABALI	KARTIK	Goel
4	LOCKOUT	Vani	Kapoor
5	DEADPOOL	Payal	sharma

17. FIND THE NAME OF THE CUSTOMER WHO PAID THE HIGHEST AMOUNT .

<u>ANS</u> : select CONCAT(Customer.f_name,Customer.l_name),m.p_id, Customer.DOB from Customer,(select p_id from payment where amt in(select max(amt) from payment))m where m.p_id=Customer.p_id

<u>OUTPUT</u> :

RESULTS

	names	p_id	DOB
1	Payalsharma	FOO5JAN	1994-03-12

18. FIND THE MOVIE HAVING THE MAXIMUM DISCOUNT.

<u>ANS</u>: select M_name,Price Discounts from Discount where offer_id in(select offer_id from Discount where Price in(select max(Price) from Discount))

<u>OUTPUT</u>:

Results | Messages

	offer_id	M_name	Discounts
1	1001	KICK	100
2	2001	KABALI	100

19. FIND THE THEATRES SHOWING MOVIES IN HINDI LANGUAGE.

<u>ANS</u>: select* from Theatre where th_id in(select th_id from hasas where m_id in(select m_id from Movies where m_id in(select m_id from Show where Lang='HINDI')))

<u>OUTPUT</u>:

	th_id	locations	t_name	Admin_id
1	001	MG HIGHWAY	SUNMOVIES	JANESH007
2	002	SP HIGHWAY	PVR CINEMA	ANY007
3	005	INORBIT	VIJAY PICTURES	DBMS400

20. FIND THE DETAILS OF THE THEATRES THAT PLAY MOVIE FOR 02:40 hrs.

<u>ANS</u>: select *from Theatre where th_id in(select th_id from hasas where m_id in(SELECT m_id from Show where CONCAT(FORMAT(DATEADD(ss,DATEDIFF(ss,st_time, end_time),0),'hh:mm') ,':00')='02:40:00'))

<u>OUTPUT</u>:

Results | Messages

	th_id	locations	t_name	Admin_id
1	001	MG HIGHWAY	SUNMOVIES	JANESH007
2	002	SP HIGHWAY	PVR CINEMA	ANY007

21.FIND THE NAME OF ALL CUSTOMERS WATCHING MOVIE “KNIVES OUT”.

```
select C_idd,CONCAT(f_name,l_name) names ,email,mname,Release_date from Customer
JOIN watch on Customer.C_iid=watch.C_idd Join Movies on watch.m_id=Movies.m_id
where mname='KNIVES OUT'
```

Results | Messages

	C_idd	names	email	mname	Release_date
1	1	JANESHPATEL	janesh@gmail.com	KNIVES OUT	2020-08-31

22.FIND THE CUSTOMER NAMES WATCHING MOVIE OF RATING 8 and above

```
select C_idd,CONCAT(f_name,l_name) names ,email,mname,Release_date,Movies.Review
from Customer JOIN watch on Customer.C_iid=watch.C_idd Join Movies on
watch.m_id=Movies.m_id where Review>=8
```

•

Results | Messages

	C_idd	names	email	mname	Release_date	Review
1	3	KARTIKGoel	kg@gmail.com	KABALI	2020-08-30	8.5
2	5	Payalsharma	ps@gmail.com	DEADPOOL	2020-08-27	8.4

23.To Display the Details of the Male and Female Costumer who paid the highest amt

```
select *from Customer JOIN payment on Customer.p_id=payment.p_id and payment.amt
in(select max(payment.amt) from Customer Join payment on
Customer.p_id=payment.p_id group by gender)
```

	email	C_iid	f_name	l_name	password	gender	DOB	p_id	ticket_id	p_id	amt	dates
1	kg@gmail.com	3	KARTIK	Goel	kr@1234	m	2000-11-16	FOO3JAN	T103	FOO3JAN	1000	2020-12-08 00:00:00.000
2	ps@gmail.com	5	Payal	sharma	ps@1234	f	1994-03-12	FOO5JAN	T105	FOO5JAN	900	2019-11-07 00:00:00.000

24.FIND NO. OF CUSTOMERS PAYING AMOUNT BETWEEN 900 AND 1000

```
select count(amt) count from payment where amt between 900 and 1001
```

	count
1	2

25.LIST THE CUSTOMER DETAILS WHO PAID THE 2nd LOWEST AMOUNT ALONG WITH THE MOVIE NAME

```
select*from payment where payment.amt in(select MIN(amt) FROM payment WHERE
payment.amt>(SELECT MIN(amt) FROM payment))
```

Results | Messages

	ticket_id	p_id	amt	dates
1	T104	F004JAN	700	2016-01-03 00:00:00.000

26.FIND THE MOVIE WATCHED BY THE CUSTOMERS HAVING NAME STARTING WITH “J” OR “K”.

```
select *from Customer Join watch on Customer.C_iid=watch.C_idd Join Movies on
Movies.m_id=watch.m_id and Customer.f_name like 'J%'
```

110 %

Results | Messages | Execution plan

	email	C_iid	f_name	l_name	password	gender	DOB	p_id	C_idd	m_id	m_id	mname	Release_date	Review
1	janesh@gmail.com	1	JANESH	PATEL	jn@1234	m	1995-11-04	F001JAN	1	001	001	KNIVES OUT	2020-08-31	6.2

27.FIND THE Details of Theatre name WHOSE F-NAME IS 5 LETTERS LONG.

```
select*from Theatre where th_id in ( select th_id from hasas where m_id in (Select
m_id from watch where C_idd in (select C_iid from Customer where Customer.f_name
like '_____')))
```

Results | Messages | Execution plan

	th_id	locations	t_name	Admin_id
1	004	DMART	BIG CINEMAS	HIHELLO8

28.DISPLAY CUSTOMER L-NAME IN DESCENDING ORDER.

```
select *from Customer order by l_name desc
```

Results | Messages | Execution plan

	email	C_iid	f_name	l_name	password	gender	DOB	p_id
1	ps@gmail.com	5	Payal	sharma	ps@1234	f	1994-03-12	FOO5JAN
2	janesh@gmail.com	1	JANESH	PATEL	jn@1234	m	1995-11-04	F001JAN
3	vani@gmail.com	4	Vani	Kapoor	vank1234	f	1996-02-12	F004JAN
4	kg@gmail.com	3	KARTIK	Goel	kr@1234	m	2000-11-16	FOO3JAN

29.GET THE GENDER WISE COUNT OF CUSTOMER (MALE & FEMALE).

```
select count(gender) count ,gender from Customer group by gender
```

Results | Messages | Execution plan

	count	gender
1	2	f
2	2	m

30.GET THE TOTAL NO. OF MOVIES.

```
select  count(mname) no_of_movies from Movies
```

100 %

Results | Messages | Executior

	no_of_movies
1	5

31.FIND THE CURRENT AGE OF ALL THE CUSTOMERS

```
SELECT Customer.C_iid,Customer.f_name,Cast(DOB as date)  dob,FLOOR(DATEDIFF(DAY,
Customer.DOB, GetDate()) / 365.25)  age from Customer
```

Results | Messages | Execution plan

	C_iid	f_name	dob	age
1	1	JANESH	1995-11-04	24
2	3	KARTIK	2000-11-16	19
3	4	Vani	1996-02-12	24
4	5	Payal	1994-03-12	26

32. FIND THE COS OF ALL AMOUNTS.

```
select payment.amt,COS(payment.amt) cos_of_amt from payment
```

Results | Messages

	amt	cos_of_amt
1	200	0.487187675007006
2	700	-0.839104325880742
3	1000	0.562379076290703
4	900	0.0662467022031581

33.DISPLAY THE MOVIE NAMES IN REVERSE ORDER.

```
select mname,REVERSE(mname) as Rev_Name from Movies
```

100 %

Results | Messages | Execution plan

	mname	Rev_Name
1	KNIVES OUT	TUO SEVINK
2	KABALI	ILABAK
3	LOCKOUT	TUOKCOL
4	DEADPOOL	LOOPDAED
5	RAID	DIAR

34.DISPLAY CUSTOMER F-NAMES IN UPPER LATTERS.

```
SELECT UPPER(Customer.f_name) upper_names FROM Customer
```

100 %

Results | Messages | Execution pla

	upper_names
1	JANESH
2	KARTIK
3	VANI
4	PAYAL

35.FIND THE DATE AFTER ADDING 1 MONTH TO THE EXISTING ONE FOR ID= “T101”.

```
SELECT payment.ticket_id,dates,DATEADD(month, 1, payment.dates) AS DateAdd from
payment where payment.ticket_id='T101'
```

	ticket_id	dates	DateAdd
1	T101	2001-08-05 00:00:00.000	2001-09-05 00:00:00.000

36.FIND WHICH MOVIE IS BEING WATCHED BY CUSTOMER HAVING DOB 11-04-1995.

```
select*from Customer Join watch on Customer.C_iid=watch.C_idd Join Movies on
watch.m_id=Movies.m_id where Customer.DOB='11-04-1995'
```

Results | Messages

	email	C_iid	f_name	l_name	password	gender	DOB	p_id	C_idd	m_id	m_id	mname	Release_date	Review
1	janesh@gmail.com	1	JANESH	PATEL	jn@1234	m	1995-11-04	F001JAN	1	001	001	KNIVES OUT	2020-08-31	6.2

37.FIND NO. OF CUSTOMERS PAYING AMOUNT BETWEEN 850 AND 1000

```
select count(amt) count from payment where amt between 850 and 1001
```

	count
1	2

38. THE MOVIE(s) HAVING THE MAXIMUM DISCOUNT.

<u>ANS</u>: select M_name,Price Discounts from Discount where offer_id in(select offer_id from Discount where Price in(select max(Price) from Discount))

<u>OUTPUT</u>:

Results | Messages

	offer_id	M_name	Discounts
1	1001	KICK	100
2	2001	KABALI	100

39.GET THE COUNT OF CUSTOMER MALE & FEMALE.

```
select count(gender) count ,gender from Customer group by gender
```

Results | Messages | Execution plan

	count	gender
1	2	f
2	2	m

40. DETAILS OF THE CUSTOMER WHO PAID THE HIGHEST AMOUNT.(NAME , P_ID AND DOB)

ANS : select CONCAT(Customer.f_name,Customer.l_name),m.p_id, Customer.DOB from Customer,(select p_id from payment where amt in(select max(amt) from payment))m where m.p_id=Customer.p_id

OUTPUT :

RESULTS

	names	p_id	DOB
1	Payalsharma	FOO5JAN	1994-03-12

PL-SQL

1. COUNT THE NUMBER OF THE MOVIES THAT ARE AVAILABLE TO WATCH.

ANS:

```
create procedure spmov(        @@count int
out)asbegin     select*from Movies        select
@@count=@@ROWCOUNTend
```

OUTPUT:

Results | Messages

	m_id	mname	Release_date	Review
1	001	KNIVES OUT	2020-08-31	6.2
2	002	KABALI	2020-08-30	8.5
3	003	LOCKOUT	2020-08-30	7.1
4	004	DEADPOOL	2020-08-27	8.4
5	005	RAID	2020-08-27	8.1

	@@count
1	5

2. FIND THE CUSTOMERS' DETAILS WHO ARE CURRENTLY WATCHING THE SAME MOVIE.

ANS:

```
CREATE FUNCTION mov_w (    @@m_id
varchar(max))RETURNS TABLEASRETURN    SELECT
        Customer.C_iid,Customer.f_name,Customer.l_na
me,Movies.mname    FROM        Customer join watch
on Customer.C_iid=watch.C_idd Join Movies on
Movies.m_id=watch.m_id    WHERE
watch.m_id=@@m_id ;          SELECT * FROM
mov_w('001');
```

OUTPUT:

	C_iid	f_name	l_name	mname
1	1	JANESH	PATEL	KNIVES OUT
2	3	KARTIK	Goel	KNIVES OUT

3. PROVIDE THE DISCOUNT OF RS.50 FROM THE TOTAL AMOUNT FOR THE CUSTOMERS HAVING THE AGE MORE THAN 20 YEARS.

ANS:

```
eclare @@amt float;declare @@p_id varchar(max);Declare
cursor_seniorcitizens Cursor for select p_id, amt from
paymentopen cursor_seniorcitizensfetch next from
cursor_seniorcitizens into @@p_id,@@amt      declare @@age
floatwhile(@@FETCH_STATUS = 0)BEGINselect
@@age=FLOOR(DATEDIFF(DAY, Cast(Customer.Dob as
date),GetDate()) / 365.25) from Customer where
Customer.p_id=@@p_idPRINT @@ageif @@age>20UPDATE
payment SET amt =(amt-50) WHERE p_id = @@p_idFETCH
NEXT FROM cursor_seniorcitizens INTO
@@p_id,@@amtendCLOSE cursor_seniorcitizens ;DEALLOCATE
cursor_seniorcitizens;select*from payment
```

OUTPUT:

	ticket_id	p_id	amt	dates
1	T101	F001JAN	50	2001-08-05 00:00:00.000
2	T104	F004JAN	550	2016-01-03 00:00:00.000
3	T103	FOO3JAN	1000	2020-12-08 00:00:00.000
4	T105	FOO5JAN	750	2019-11-07 00:00:00.000

4. Find all the theatres where the movie is being showed provided the movie id as a parameter.

ANS:

```
alter proc spth
(
	@@m_id varchar(max)
)
	as
	begin
	declare @@st varchar(max);
	set @@st='';

	select @@st=@@st+Movies.mname+ 'showing in
'+Theatre.locations+char(10)
	from Movies JOIN hasas on Movies.m_id=hasas.m_id Join
Theatre on Theatre.th_id=hasas.th_id where
Movies.m_id=@@m_id;
		print @@st;
end
```

OUTPUT:

```
KABALIshowing in SP HIGHWAY
KABALIshowing in GOLDEN PALACE

(1 row affected)

Completion time: 2020-10-30T23:54:28.2218850+05:30
```

5. Find all the theatres where the movie is being showed provided the movie id as a parameter.

ANS:

```
create trigger tr_agess
On Customer
instead of insert
as
begin
Declare @@dob datetime
Declare @@c_iid varchar(max)
Declare @@f_name varchar(max)
Declare @@l_name varchar(max)
Declare @@e_mail varchar(max)
Declare @@p_id varchar(max)
Declare @@gender char(1)
Declare @@password varchar(max)
Declare @@age float
select @@dob=DOB from inserted
select @@f_name= f_name from inserted
select @@password=password from inserted
select @@l_name= l_name from inserted
select @@c_iid=C_iid from inserted
select @@e_mail=email from inserted
select @@gender=gender from inserted
set @@age=FLOOR(DATEDIFF(DAY, Cast(@@dob as date),GetDate())
/ 365.25)
insert into Customer
([email],[C_iid],[f_name],[l_name],[password],[gender],[DOB],[age]
)
values
(@@e_mail,@@c_iid,@@f_name,@@l_name,@@password,@@gender,
@@DOB,@@age)
end
```

OUTPUT:

	email	C_iid	f_name	l_name	password	gender	DOB	age
1	janesh@gmail.com	1	JANESH	PATEL	jn@1234	m	1995-11-04	NULL
2	kg@gmail.com	3	KARTIK	Goel	kr@1234	m	2000-11-16	NULL
3	vani@gmail.com	4	Vani	Kapoor	vank1234	f	1996-02-12	NULL
4	ps@gmail.com	5	Payal	sharma	ps@1234	f	1994-03-12	NULL
5	va@gmail.com	6	vansh	aggarwal	va@gmail.com	m	2000-07-11	20
6	ram@gmail.com	7	ram	shah	ram@gmail.com	m	2000-09-20	20

FRONT-END

HOME PAGE

File | C:/Users/Vansh%20Aggarwal/Desktop/dbms%20web/landing.html#
Apps
New Tab
Downloads
Book Your Show
About
Contact us
TOM CRUISE
OBLIVION
Sign Up
Login
file:///C:/Users/Vansh Aggarwal/Desktop/dbms web/landing.html#carouselExampleControls
Type here to search
ENG
21:13
30-10-2020

LOGIN PAGE

File | C:/Users/Vansh%20Aggarwal/Desktop/dbms%20web/movielogin.html
Apps
New Tab
Downloads
Book Your Show
About
Contact us
Log In
Email
email id
Password
enter password
Login
Type here to search
ENG
21:13
30-10-2020

SIGN-UP PAGE

File | C:/Users/Vansh%20Aggarwal/Desktop/dbms%20web/moviesignup.html
Book Your Show
About
Contact us
Sign Up
First Name: Barry
Last Name: Allen
Male
Female
Others
Email email id
Password enter password
DOB: Jan 1 1980
I agree to terms and conditions
Submit
Type here to search
ENG
21:13
30-10-2020

BOOKING PAGE

BOOKING PAGE

BOOKING PAGE

SEAT SELECTION

PAYMENT PAGE

PAYMENT CONFIRMATION

CONCLUSION

Nowadays, the traditional book cinema tickets dying methods. This technology has dominated human life, where is the new era. Software and technical equipment, exceptions are reduced and even eliminated. Moreover, for every part of their lives easy, fast and secure way to make the choice. This project is a cinema ticket booking system is designed to meet the requirements. It has been developed in PHP and database keeping in mind the specifications of the system has been created in My SQL server. Our project: ticket system with the cinema; Cinema companies can meet customer comfort. Cinema manager, employee, and customer relations to complete the process of ticket satisfied a good communication. With this platform we developed, we avoid wasting time reduce misunderstandings, easy data flow, customer happiness, and are expected to provide less difficult task. We accomplish our goals and we are satisfied with the developed code that believes.

FUTURE WORK

Any specification-untraced errors will be concentrated in the coming versions, which are planned to be developed in near future. The system at present does not take care of the money payment methods, as the consolidated constructs need SSL standards and are critically to be initiated in the first face, the application of the credit card transactions is applied as a developmental phase in the coming days. The system needs more elaborative technicality for its inception and evolution.

REFERENCES:

[1] Elmasri and Navathe, “Fundamentals of Database Systems”, 6/e, Addison - Wesley, 2010.

[2]A Silberschaltz, H.F. Korth, and S sudarshan, “Database System Concepts”, 3/e, Tata Mcgraw Hill.

[3] Thomas M. Connolly, Carolyn E. Begg, “Database Systems & Practical Approach to Design Implementation and Management”, 4/e, Addison – Wesley, 2005.

[4] . Advertising allocation and impact of advertising on event ticket sales: Which product, where, and when - MY Kang - International Journal of Market Research