Table of Contents

1.PROJECT DESCRIPTION

Cinema-going is one of the most popular out-of-home cultural activities, affecting a serious of social, economic and cultural phenomena in modern societies. Cinemas are considered to be an integral part of cities and they contribute to the definition of a local geography and identity. They also contribute to the preservation of the collective memory, since they constitute a significant social and cultural practice linked to a specific place, which acts as a common reference or landmark for many individuals.

This database can prove to be a comprehensive solution for ticket booking in multiplexes and can be used in Theatre management system, an online ticket selling software that is easy to understand, easy to use and offers the simplicity of fast point-and-click service to the customers.

The system provides broad overview of underlying operational factors that influence cinema management.

Each cinema is identified by its cinema ID.

The details of various cinemas are added in the database. The details include id, name, address, number of screens etc.

The number of movies a cinema can show at any given instant is dependent upon the number of screens. Each screen is uniquely identifiable by its screen id.

The movies have a fixed screen number and each movie has various attributes such as name, length, ratings, age restrictions, starting and ending date etc. The movies are identified by movie ID.

When a customer books a ticket, his/her details like phone number, name, email and membership status are stored. Each customer is identified by its customer ID and the bookings are identified by booing ID.

A record of sale of tickets is also kept in a separate table by the staff of the cinema. Only the staff members can access or edit this table. Essential details like sale Id, payment mode, movie name (against which the ticket is purchased) are stored in the sales table.

The staff members can be offered various positions and can have permissions according to their permission. The staff member is identified by its unique staff ID.

2.ENTITIES

CINEMA

CinemaID	Cinema Name	Cinema Address	CinemaPhoneNo	BranchRef	NoOf Screens

Reviewer

MovieName	ReviewerID	PriorityRating	Starts

Staff

StaffID	StaffName	StaffPosition

Customer

Customer ID	CustomerName	CustomerPhoneNo	Customer Email	Membership

Movie

MovieID	Movie Rating	RunTime	StartDate	EndDate	AgeRating	MovieName

Screen

ScreenNo	ScreenCapacity	CinemaID

Performances

PerformanceID	StartTime	EndTime	ScreenNo	MovieID

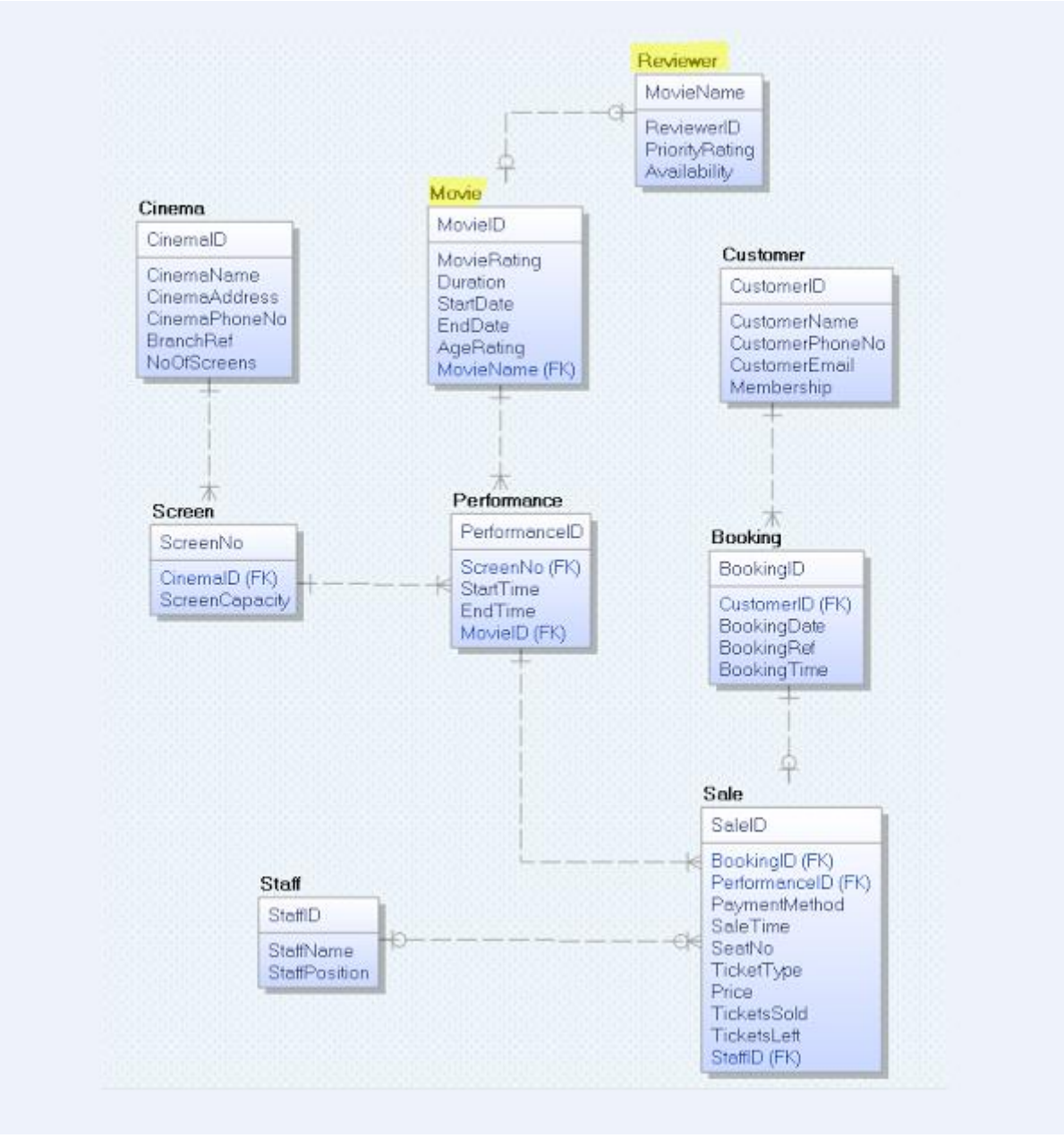
Booking

BookingID	BookingDate	BookingReference	BookingTime	CustomerID

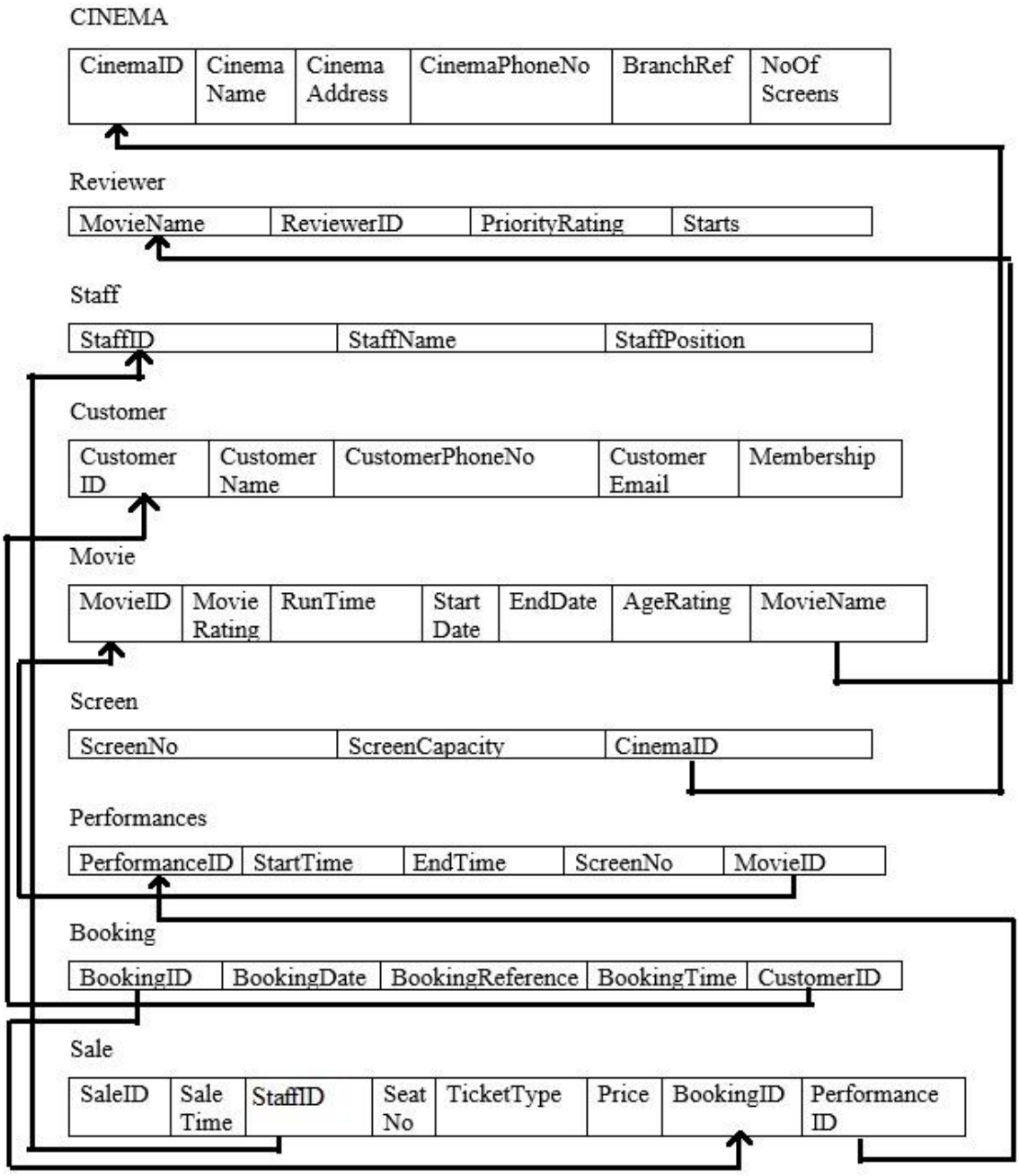
Sale

SaleID	Sale Time	PaymentMethod	SeatNo	TicketType	Price	BookingID	Performance ID

3.ER DIAGRAM


Reviewer
MovieName
ReviewerID
PriorityRating
Availability
Movie
MovieID
MovieRating
Duration
StartDate
EndDate
AgeRating
MovieName (FK)
Cinema
CinemaID
CinemaName
CinemaAddress
CinemaPhoneNo
BranchRef
NoOfScreens
Customer
CustomerID
CustomerName
CustomerPhoneNo
CustomerEmail
Membership
Screen
ScreenNo
CinemaID (FK)
ScreenCapacity
Performance
PerformanceID
ScreenNo (FK)
StartTime
EndTime
MovieID (FK)
Booking
BookingID
CustomerID (FK)
BookingDate
BookingRef
BookingTime
Sale
SaleID
BookingID (FK)
PerformanceID (FK)
PaymentMethod
SaleTime
SeatNo
TicketType
Price
TicketsSold
TicketsLeft
StaffID (FK)
Staff
StaffID
StaffName
StaffPosition

4.RELATIONAL SCHEMA



5.SQL

-- Cinema System SQL to create tables

/* Drop Table statements are included for each table to ensure that

when you first create the tables none with the same name exist. */

```
DROP TABLE Cinema CASCADE CONSTRAINTS PURGE;
DROP TABLE Reviewer CASCADE CONSTRAINTS PURGE;
DROP TABLE Staff CASCADE CONSTRAINTS PURGE;
DROP TABLE Customer CASCADE CONSTRAINTS PURGE;
DROP TABLE Movie CASCADE CONSTRAINTS PURGE;
DROP TABLE Screen CASCADE CONSTRAINTS PURGE;
DROP TABLE Performances CASCADE CONSTRAINTS PURGE;
DROP TABLE Booking CASCADE CONSTRAINTS PURGE;
DROP TABLE Sale CASCADE CONSTRAINTS PURGE;
```

/* Create table statements

Tables are created in 3 layers following a given order:

1st: Those with no foreign keys are created first

2nd: Tables depending only on these tables i.e. have foreign key relationships to 1st layer

3rd: Tables depending on 2nd layer or combination of 1st and 2nd layer

***/**

-- Create table Cinema- holds data on the different branches for the company

```
CREATE TABLE Cinema
(
CinemaID NUMBER(6) NOT NULL,
CinemaName VARCHAR2(30) NULL,
CinemaAddress VARCHAR2(30) NULL,
CinemaPhoneNo VARCHAR2(10) NULL,
BranchRef Number(2) NULL,
NoOfScreens Number(2) NULL,
CONSTRAINT Cinema_PK PRIMARY KEY (CinemaID)
);

CREATE TABLE Reviewer
(
MovieName VARCHAR2(50) NOT NULL,
ReviewerID NUMBER(6) NOT NULL,
PriorityRating VARCHAR2(2) NULL,
Starts DATE NULL,
CONSTRAINT Reviewer_PK PRIMARY KEY (MovieName)
);

CREATE TABLE Staff
(
```

```
StaffID NUMBER(6) NOT NULL,
StaffName VARCHAR2(30) NULL,
StaffPosition VARCHAR2(20) NULL,
CONSTRAINT Staff_PK PRIMARY KEY (StaffID)
);
```

-- Create table Customer - holds data on the cinemas customers

```
CREATE TABLE Customer
(
CustomerID NUMBER(6) NOT NULL,
CustomerName VARCHAR2(30) NULL,
CustomerPhoneNo VARCHAR2(10) NULL,
CustomerEmail VARCHAR2(30) NULL,
Membership VARCHAR2(1) NULL,
CONSTRAINT Customer_PK PRIMARY KEY (CustomerID)
);
```

-- Create table Movies - holds details of all movies that the cinema is showing

```
CREATE TABLE Movie
(
MovieID NUMBER(6) NOT NULL,
MovieRating VARCHAR2(10) NULL,
RunTime NUMBER(3) NULL,
StartDate DATE NULL,
EndDate DATE NULL,
```

```
AgeRating VARCHAR2(4) NULL,
MovieName VARCHAR2(50) NULL,
CONSTRAINT Movie_PK PRIMARY KEY (MovieID),
CONSTRAINT Movie_Reviewer_FK FOREIGN KEY (MovieName) REFERENCES
Reviewer (MovieName)
);
```

-- Create table Screen - holds details on each screen in each cinema

```
CREATE TABLE Screen
(
ScreenNo NUMBER(2) NOT NULL,
ScreenCapacity NUMBER(3) NOT NULL,
CinemaID NUMBER(6) NOT NULL,
CONSTRAINT Screen_PK PRIMARY KEY (ScreenNo),
CONSTRAINT Cinema_Screen_FK FOREIGN KEY (CinemaID) REFERENCES Cinema
(CinemaID)
);
```

-- Create table Performance - holds details of all performances each week in the cinema

```
CREATE TABLE Performances
(
PerformanceID NUMBER(6) NOT NULL,
StartTime VARCHAR2(10) NULL,
EndTime VARCHAR2(10) NULL,
```

ScreenNo NUMBER(2) NULL,

MovieID NUMBER(6) NOT NULL,

CONSTRAINT Performances_PK PRIMARY KEY (PerformanceID),

CONSTRAINT Screen_Performances_FK FOREIGN KEY (ScreenNo) REFERENCES

Screen (ScreenNo),

CONSTRAINT Movie_Performances_FK FOREIGN KEY (MovieID) REFERENCES Movie

(MovieID)

);

-- Create table Bookings - holds details of all bookings made over the internet/phone

CREATE TABLE Booking

(

BookingID NUMBER(6) NOT NULL,

BookingDate DATE NULL,

BookingReference NUMBER(6) NOT NULL,

BookingTime VARCHAR2(10) NULL,

CustomerID NUMBER(6) NULL,

CONSTRAINT Booking_PK PRIMARY KEY (BookingID),

CONSTRAINT Customer_Booking_FK FOREIGN KEY (CustomerID) REFERENCES

Customer (CustomerID)

);

CREATE TABLE Sale

(

SaleID NUMBER(6) NOT NULL,

SaleTime VARCHAR2(10) NULL,

PaymentMethod VARCHAR2(4) NULL,

SeatNo VARCHAR2(4) NULL,

TicketType VARCHAR2(10) NULL,

Price NUMBER(6,2) NULL,

BookingID NUMBER(6) NOT NULL,

PerformanceID NUMBER(6) NOT NULL,

CONSTRAINT Sale_PK PRIMARY KEY (SaleID),

CONSTRAINT Booking_Sale_FK FOREIGN KEY (BookingID) REFERENCES Booking (BookingID),

CONSTRAINT Performances_Sale_FK FOREIGN KEY (PerformanceID) REFERENCES Performances (PerformanceID)

);

-- Insert statements to populate the tables

INSERT INTO Cinema (CinemaID, CinemaName, CinemaAddress, CinemaPhoneNo, BranchRef, NoOfScreens)

VALUES (1000, 'Pvr Cinema', 'Ldh', 9458568954, 4, 10);

CINEMAID	CINEMANAME	CINEMAADDRESS	CINEMAPHONENO	BRANCHREF	NOOFSCREENS
1000	Pvr Cinema	Ldh	9458568954	4	10

--Reviewer Table

```
INSERT INTO Reviewer (MovieName, ReviewerID, PriorityRating, Starts)
VALUES ('War', 14, 4,TO_DATE('2019-10-02', 'yyyy-mm-dd'));
INSERT INTO Reviewer (MovieName, ReviewerID, PriorityRating, Starts)
VALUES ('Avengers', 18, 8,TO_DATE('2019-04-11', 'yyyy-mm-dd'));
INSERT INTO Reviewer (MovieName, ReviewerID, PriorityRating, Starts)
VALUES ('Bharat', 18, 9,TO_DATE( '2015-09-19', 'yyyy-mm-dd'));
INSERT INTO Reviewer (MovieName, ReviewerID, PriorityRating, Starts)
VALUES ('Thor', 15, 10,TO_DATE('2019-10-24', 'yyyy-mm-dd'));
INSERT INTO Reviewer (MovieName, ReviewerID, PriorityRating, Starts)
VALUES ('Captain America', 58, 6,TO_DATE('2018-09-18', 'yyyy-mm-dd'));
```

MOVIENAME	REVIEWERID	PRIORITYRATING	STARTS
War	14	4	10/02/2019
Avengers	18	8	04/11/2019
Bharat	18	9	09/19/2015
Thor	15	10	10/24/2019
Captain America	58	6	09/18/2018

--Staff Table

```
INSERT INTO Staff (StaffID , StaffName, StaffPosition)
VALUES (1, 'John' , 'Cinema Operative');
INSERT INTO Staff (StaffID , StaffName, StaffPosition)
VALUES (2, 'James' , 'Cinema Operative');
INSERT INTO Staff (StaffID , StaffName, StaffPosition)
VALUES (3, 'Jennie' , 'Cinema Operative');
```

INSERT INTO Staff (StaffID , StaffName, StaffPosition)

VALUES (4, 'Jack' , 'cinema Manager');

STAFFID	STAFFNAME	STAFFPOSITION
1	John	Cinema Operative
2	James	Cinema Operative
3	Jennie	Cinema Operative
4	Jack	cinema Manager

--Customer Table

INSERT INTO Customer (CustomerID, CustomerName, CustomerPhoneNo,

CustomerEmail, Membership)

VALUES (100, 'Abhishek', '9456875478', 'Abhishek@gmail.com', '1');

INSERT INTO Customer (CustomerID, CustomerName, CustomerPhoneNo,

CustomerEmail, Membership)

VALUES (101, 'Abhi', '9658547895', 'abhi@gmail.com', '0');

INSERT INTO Customer (CustomerID, CustomerName, CustomerPhoneNo,

CustomerEmail, Membership)

VALUES (102, 'Varun', '9865412578', 'varun@gmail.com', '0');

INSERT INTO Customer (CustomerID, CustomerName, CustomerPhoneNo,

CustomerEmail, Membership)

VALUES (103, 'Sahil', '8956247851', 'sahil@gmail.com', '0');

INSERT INTO Customer (CustomerID, CustomerName, CustomerPhoneNo,

CustomerEmail, Membership)

VALUES (104, 'Tejeshwar', '9568742548', 'tejeshwar@gmail.com', '0');

CUSTOMERID	CUSTOMERNAME	CUSTOMERPHONENO	CUSTOMEREMAIL	MEMBERSHIP
100	Abhishek	9456875478	Abhishek@gmail.com	1
101	Abhi	9658547895	abhi@gmail.com	0
102	Varun	9865412578	varun@gmail.com	0
103	Sahil	8956247851	sahil@gmail.com	0
104	Tejeshwar	9568742548	tejeshwar@gmail.com	0

--Movie Table

INSERT INTO Movie (MovieID, MovieRating, RunTime, StartDate, EndDate, MovieName)

VALUES (201, '1E', 111,TO_DATE('2019-10-02', 'yyyy-mm-dd'),TO_DATE('2019-11-02', 'yyyy-mm-dd'), 'War');

INSERT INTO Movie (MovieID, MovieRating, RunTime, StartDate, EndDate, MovieName)

VALUES (202, 'D', 141,TO_DATE('2019-04-11', 'yyyy-mm-dd'),TO_DATE('2019-05-11', 'yyyy-mm-dd'), 'Avengers');

INSERT INTO Movie (MovieID, MovieRating, RunTime, StartDate, EndDate, MovieName)

VALUES (203, 'C', 121,TO_DATE('2015-09-19', 'yyyy-mm-dd'),TO_DATE('2015-10-19', 'yyyy-mm-dd') , 'Bharat');

INSERT INTO Movie (MovieID, MovieRating, RunTime, StartDate, EndDate, MovieName)

VALUES (204, 'B', 94,TO_DATE('2019-10-24', 'yyyy-mm-dd') ,TO_DATE('2019-11-24', 'yyyy-mm-dd'), 'Thor');

INSERT INTO Movie (MovieID, MovieRating, RunTime, StartDate, EndDate, MovieName)

VALUES (205, '10A', 106,TO_DATE('2018-09-18', 'yyyy-mm-dd') ,TO_DATE('2018-10-18', 'yyyy-mm-dd'), 'Captain America');

MOVIEID	MOVIERATING	RUNTIME	STARTDATE	ENDDATE	AGERATING	MOVIENAME
201	1E	111	10/02/2019	11/02/2019	-	War
202	D	141	04/11/2019	05/11/2019	-	Avengers
203	C	121	09/19/2015	10/19/2015	-	Bharat
204	B	94	10/24/2019	11/24/2019	-	Thor
205	10A	106	09/18/2018	10/18/2018	-	Captain America

--Screen table

```
INSERT INTO Screen (ScreenNo, CinemaId, ScreenCapacity)

VALUES (1, 1000, 250);

INSERT INTO Screen (ScreenNo, CinemaId, ScreenCapacity)

VALUES (2, 1000, 150);

INSERT INTO Screen (ScreenNo, CinemaId, ScreenCapacity)

VALUES (3, 1000, 75);

INSERT INTO Screen (ScreenNo, CinemaId, ScreenCapacity)

VALUES (4, 1000, 200);

INSERT INTO Screen (ScreenNo, CinemaId, ScreenCapacity)

VALUES (5, 1000, 60);
```

SCREENNO	SCREENCAPACITY	CINEMAID
1	250	1000
2	150	1000
3	75	1000
4	200	1000
5	60	1000

--Performance Table

INSERT INTO Performances (PerformanceID, StartTime, EndTime, ScreenNo, MovieID)

VALUES (301, '14:00:00', '15:55:00', 1, 201);

INSERT INTO Performances (PerformanceID, StartTime, EndTime, ScreenNo, MovieID)

VALUES (302, '16:00:00', '18:14:00', 2, 202);

INSERT INTO Performances (PerformanceID, StartTime, EndTime, ScreenNo, MovieID)

VALUES (303, '11:00:00', '13:01:00', 3, 203);

INSERT INTO Performances (PerformanceID, StartTime, EndTime, ScreenNo, MovieID)

VALUES (304, '17:00:00', '19:42:00', 4, 204);

INSERT INTO Performances (PerformanceID, StartTime, EndTime, ScreenNo, MovieID)

VALUES (305, '15:00:00', '17:00:00', 5, 205);

PERFORMANCEID	STARTTIME	ENDTIME	SCREENNO	MOVIEID
301	14:00:00	15:55:00	1	201
302	16:00:00	18:14:00	2	202
303	11:00:00	13:01:00	3	203
304	17:00:00	19:42:00	4	204
305	15:00:00	17:00:00	5	205

--Booking Table

INSERT INTO Booking (BookingID, BookingDate, BookingReference, BookingTime, CustomerID)

VALUES (1000,TO_DATE('2019-10-05', 'yyyy-mm-dd') , 225,'17:49:00', 101);

INSERT INTO Booking (BookingID, BookingDate, BookingReference, BookingTime, CustomerID)

VALUES (1001,TO_DATE('2019-04-15', 'yyyy-mm-dd'), 226, '12:00:00', 103);

INSERT INTO Booking (BookingID, BookingDate, BookingReference, BookingTime, CustomerID)

VALUES (1002,TO_DATE('2015-09-24', 'yyyy-mm-dd') , 227, '13:10:00', 100);

INSERT INTO Booking (BookingID, BookingDate, BookingReference, BookingTime, CustomerID)

VALUES (1003,TO_DATE('2019-10-29', 'yyyy-mm-dd') , 228, '13:11:00', 102);

INSERT INTO Booking (BookingID, BookingDate, BookingReference, BookingTime, CustomerID)

VALUES (1004,TO_DATE('2018-09-23', 'yyyy-mm-dd'), 229, '18:30:00', 104);

BOOKINGID	BOOKINGDATE	BOOKINGREFEERENCE	BOOKINGTIME	CUSTOMERID
1000	10/05/2019	225	17:49:00	101
1001	04/15/2019	226	12:00:00	103
1002	09/24/2015	227	13:10:00	100
1003	10/29/2019	228	13:11:00	102
1004	09/23/2018	229	18:30:00	104

--Sales table

INSERT INTO Sale (SaleID, SaleTime, PaymentMethod, SeatNo, TicketType, Price, BookingID, PerformanceID)

VALUES (1000, '13:24:00', 'cash', 'M1', 'Standard', 150, 1000, 301);

INSERT INTO Sale (SaleID, SaleTime, PaymentMethod, SeatNo, TicketType, Price, BookingID, PerformanceID)

VALUES (1001, '17:49:00', 'card', 'N1', 'Standard', 150, 1000, 302);

INSERT INTO Sale (SaleID, SaleTime, PaymentMethod, SeatNo, TicketType, Price,

BookingID, PerformanceID)

VALUES (1002, '12:00:00', 'card', ' L5', 'Standard', 125, 1001, 302);

INSERT INTO Sale (SaleID, SaleTime, PaymentMethod, SeatNo, TicketType, Price,

BookingID, PerformanceID)

VALUES (1004, '13:10:00', 'card', ' P15', 'Standard', 175, 1002, 303);

INSERT INTO Sale (SaleID, SaleTime, PaymentMethod, SeatNo, TicketType, Price,

BookingID, PerformanceID)

VALUES (1005, '13:11:00', 'card', ' K14', 'Premier', 200, 1003, 304);

INSERT INTO Sale (SaleID, SaleTime, PaymentMethod, SeatNo, TicketType, Price,

BookingID, PerformanceID)

VALUES (1006, '18:30:00', 'card', ' M4', 'Standard', 150, 1004, 305);

SALEID	SALETIME	PAYMENTMETHOD	SEATNO	TICKETTYPE	PRICE	BOOKINGID	PERFORMANCEID
1000	13:24:00	cash	M1	Standard	150	1000	301
1001	17:49:00	card	N1	Standard	150	1000	302
1002	12:00:00	card	L5	Standard	125	1001	302
1005	13:11:00	card	K14	Premier	200	1003	304
1004	13:10:00	card	P15	Standard	175	1002	303
1006	18:30:00	card	M4	Standard	150	1004	305

--Grants

GRANT DELETE, INSERT ON Movie TO sparsh;

GRANT DELETE, UPDATE, INSERT ON Performances TO sparsh;

GRANT INSERT ON Reviewer TO sahil;

GRANT DELETE, UPDATE ON Performances TO sahil;

6.PL/SQL

STORED PROCEDURE

-- Procedure that adds Priority Rating to the Movie

-- that hasn't been reviewed in the MOVIE TABLE

```
create or replace procedure "ADD_PRIORITY_RATING"
(mv_name IN REVIEWER.MOVIENAME%TYPE,
priority_rating REVIEWER.PRIORITYRATING%TYPE)
is
begin
BEGIN
  IF movie_name_availability(mv_name) THEN
     INSERT INTO Reviewer (MovieName,PriorityRating)
     VALUES(mv_name, priority_rating);
    ELSE
     DBMS_OUTPUT.PUT_LINE('No movie with such a name');
  END IF;
  EXCEPTION
  WHEN others THEN
  DBMS_OUTPUT.PUT_LINE('An Error occurred.');
END;
end;
```

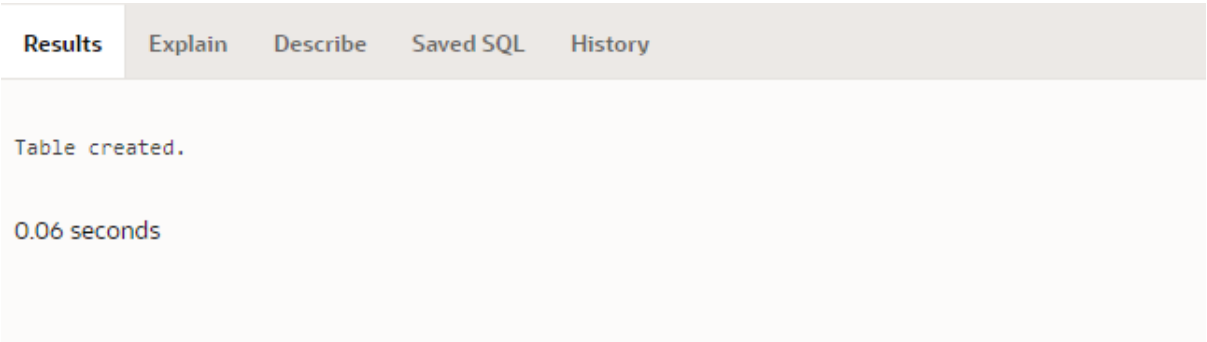
PL/SQL code successfully compiled (11:46:01)

```
create or replace procedure "ADD_PRIORITY_RATING"
(mv_name IN REVIEWER.MOVIENAME%TYPE,
priority_rating REVIEWER.PRIORITYRATING%TYPE)
is
begin
BEGIN
  IF movie_name_availability(mv_name) THEN
      INSERT INTO Reviewer (MovieName,PriorityRating)
      VALUES(mv_name, priority_rating);
    ELSE
      DBMS_OUTPUT.PUT_LINE('No movie with such a name');
  END IF;
  EXCEPTION
  WHEN others THEN
  DBMS_OUTPUT.PUT_LINE('An Error occurred.');
END;
end;
```

TRIGGERS

-- Log Table

```
CREATE TABLE Reviewer_Log (
  Table_name varchar(20),
  ReviewerID number(6),
  MovieName varchar2(50),
  Rev char(3) CHECK (Rev IN ('INS','UPD','DEL')),
  s_date DATE
)
```



-- Audit Trigger

```
create or replace trigger "REVIEW_AI"
AFTER
insert on "REVIEWER"
for each row
begin
```

INSERT INTO Reviewer_Log (Table_name,ReviewerID,MovieName,Rev,s_date) values('REVIEWS', 'INS', :new.ReviewerID, :new.MovieName, SYSDATE);

end;

-- Constraint Trigger

CREATE OR REPLACE TRIGGER Review_AI_2

BEFORE INSERT ON Reviewer

FOR EACH ROW

begin

BEGIN

DBMS_OUTPUT.PUT_LINE('Executing Review_AI_2');

END;

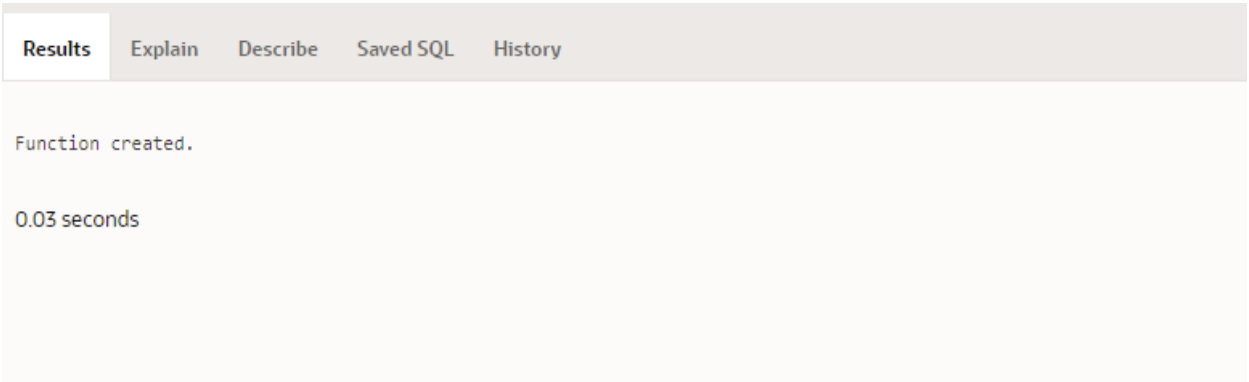
end

FUNCTION

-- Function for the Reviewer who checks if the

-- Movie is available in the cinema (Movie Table) or not

```
CREATE OR REPLACE FUNCTION movie_name_availability
 (
   movie_name MOVIE.MOVIENAME%TYPE
 )
 RETURN BOOLEAN
IS
 movie_id MOVIE.MOVIEID%TYPE;
BEGIN
 SELECT MOVIEID INTO movie_id FROM MOVIE WHERE MOVIENAME =
movie_name;
 RETURN TRUE;
EXCEPTION
WHEN NO_DATA_FOUND THEN
 RETURN FALSE;
END movie_name_availability;
```



QUERIES EXECUTED

-- Selecting and Subquery and Difference - Shows the movies that have a higher PriorityRating than the movie Pan

```
SELECT MovieName FROM Reviewer

WHERE PriorityRating >

(SELECT PriorityRating FROM Reviewer

WHERE MovieName='War');
```

MOVIENAME
Avengers
Bharat
Captain America

-- Intersection - Movie names that have been reviewed by the Reviewer and put into the Movie Table

```
SELECT MovieName

FROM Reviewer

INTERSECT

SELECT MovieName

FROM Movie;
```

MOVIENAME
Avengers
Bharat
Captain America
Thor
War

-- Union - The number of each screen that has a Movie being played

```
SELECT ScreenNo FROM Screen

UNION

SELECT ScreenNo FROM Performances

ORDER BY ScreenNo;
```

	SCREENNO
1	
2	
3	
4	
5	

-- Inner Join - Shows the rating of each Movie

```
SELECT  MovieName, MovieRating

FROM Movie

INNER JOIN Performances

ON Movie.MovieID = Performances.MovieID;
```

MOVIENAME	MOVIERATING
War	1E
Avengers	D
Bharat	C
Thor	B
Captain America	10A

-- Outer Join - Shows the total number of screens the cinema has

```
SELECT Cinema.CinemaID, Screen.ScreenNo

FROM Cinema
```

```
FULL OUTER JOIN Screen

ON Cinema.CinemaID=Screen.ScreenNo

ORDER BY Cinema.CinemaID;
```

CINEMAID	SCREENNO
1000	-
-	2
-	4
-	5
-	1
-	3

-- Aggregation - The current money that the cinema has earned

```
SELECT SUM(Price) AS CurrentMoney FROM Sale;
```

CURRENTMONEY
950