A database for

Movies

project

By Anudeep vattikuti

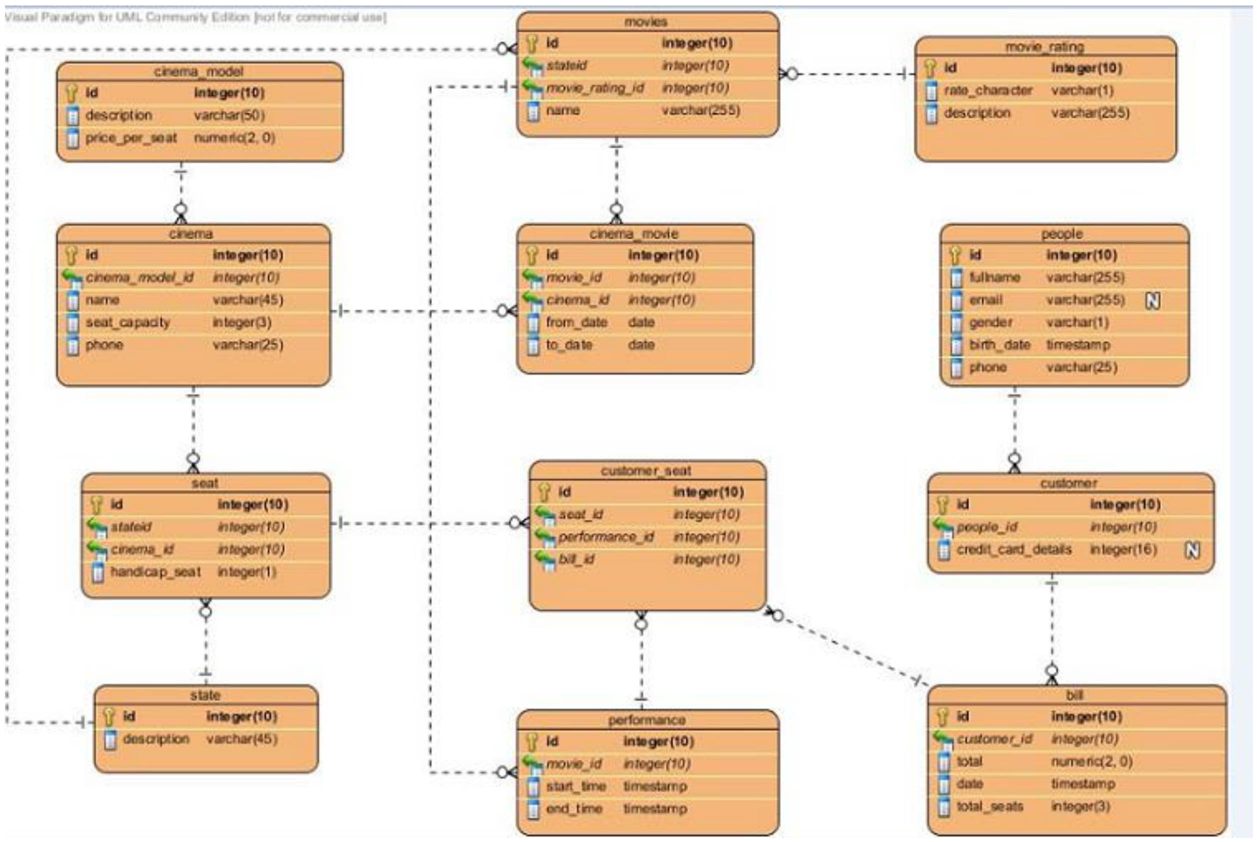
Table of contents

Summary

The document describes the database for movies includes all the type of data regarding to the entities. This is a way to see the way of playing movies, choosing seats, performance, ratings. And the design having variety of seats capacity like gold, diamond, silver, with the certain range of limits. The people who are entering into the theatre and we must have certain requirements like checking email id, mobile numbers and seat id.

And the entity relationship diagram shows the relation between all the tables depending upon their SQL code, functional dependencies and sample data. Here triggers and stored procedures are shown. Finally security, known problems and future improvements.

Entity relationship Diagram



Tables

Bill

Purpose:

To count the number of tickets sold and to check the revenue on day to day basis.

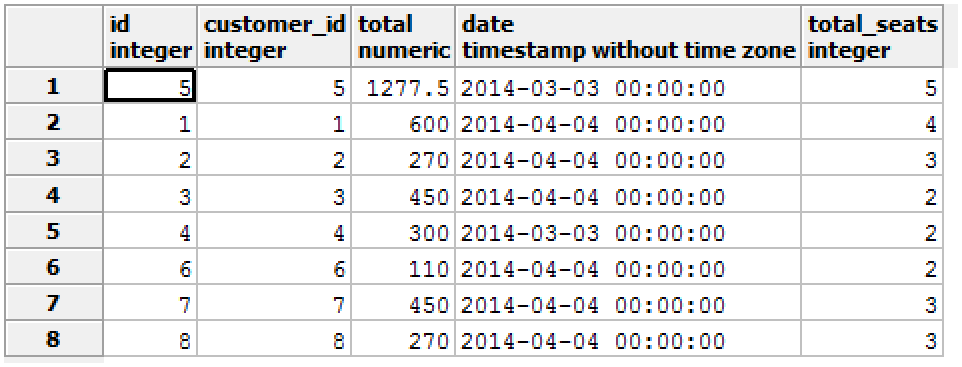
Create statement:

```
CREATE TABLE bill
(
  id integer NOT NULL,
  customer_id integer NOT NULL,
  total numeric NOT NULL DEFAULT 0,
  date timestamp without time zone NOT NULL,
  total_seats integer NOT NULL,
  CONSTRAINT pk_bill_id PRIMARY KEY (id),
  CONSTRAINT fk_customer_id FOREIGN KEY (customer_id)
      REFERENCES customer (id)
)
```

Functional dependencies:

id $\longrightarrow$ customer_id , total , date , total_seats

Sample data:



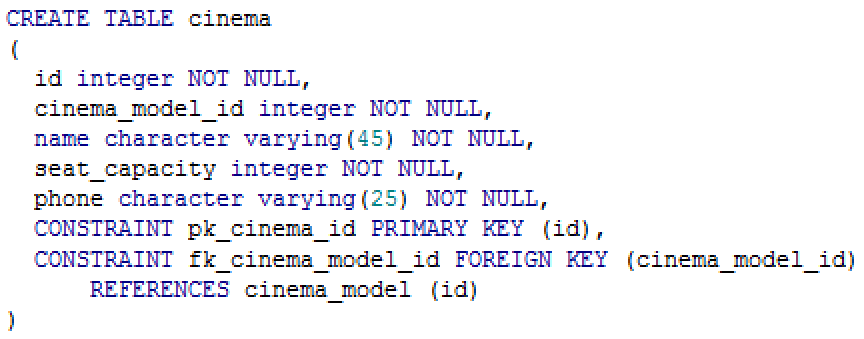
	id integer	customer_id integer	total numeric	date timestamp without time zone	total_seats integer
1	5	5	1277.5	2014-03-03 00:00:00	5
2	1	1	600	2014-04-04 00:00:00	4
3	2	2	270	2014-04-04 00:00:00	3
4	3	3	450	2014-04-04 00:00:00	2
5	4	4	300	2014-03-03 00:00:00	2
6	6	6	110	2014-04-04 00:00:00	2
7	7	7	450	2014-04-04 00:00:00	3
8	8	8	270	2014-04-04 00:00:00	3

Cinema

Purpose:

Cinema is the art of moving images; a visual medium that tells stories and exposes reality and the type of seating capacity.

Create statement:

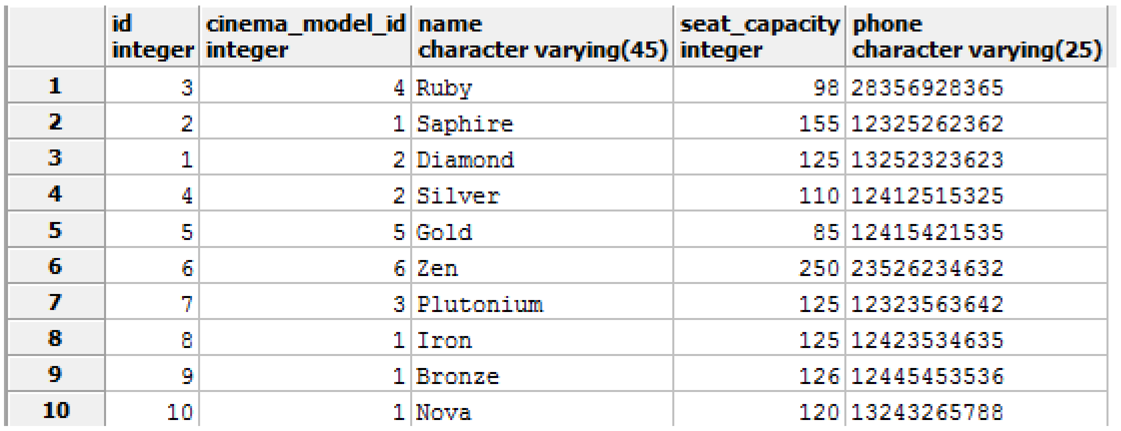


```
CREATE TABLE cinema
(
  id integer NOT NULL,
  cinema_model_id integer NOT NULL,
  name character varying(45) NOT NULL,
  seat_capacity integer NOT NULL,
  phone character varying(25) NOT NULL,
  CONSTRAINT pk_cinema_id PRIMARY KEY (id),
  CONSTRAINT fk_cinema_model_id FOREIGN KEY (cinema_model_id)
      REFERENCES cinema_model (id)
)
```

Functional dependencies:

id $\longrightarrow$ cinema_model_id , name , seat_capacity , phone

Sample data:



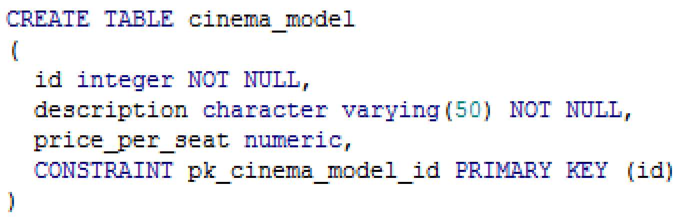
	id integer	cinema_model_id integer	name character varying(45)	seat_capacity integer	phone character varying(25)
1	3	4	Ruby	98	28356928365
2	2	1	Saphire	155	12325262362
3	1	2	Diamond	125	13252323623
4	4	2	Silver	110	12412515325
5	5	5	Gold	85	12415421535
6	6	6	Zen	250	23526234632
7	7	3	Plutonium	125	12323563642
8	8	1	Iron	125	12423534635
9	9	1	Bronze	126	12445453536
10	10	1	Nova	120	13243265788

Cinema_model

Purpose:

Cinema model is the way of displaying the movie in different images on the screen.

Create statement:

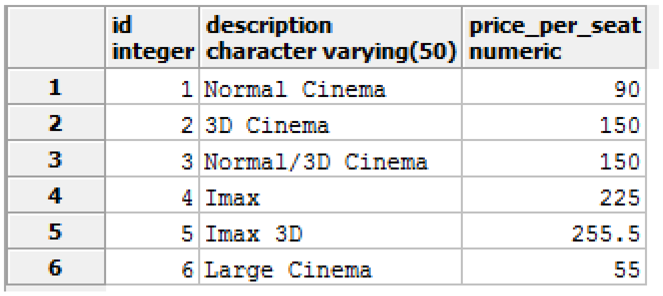


```
CREATE TABLE cinema_model
(
  id integer NOT NULL,
  description character varying(50) NOT NULL,
  price_per_seat numeric,
  CONSTRAINT pk_cinema_model_id PRIMARY KEY (id)
)
```

Functional dependencies:

id $\longrightarrow$ description , price_per_seat

Sample data:



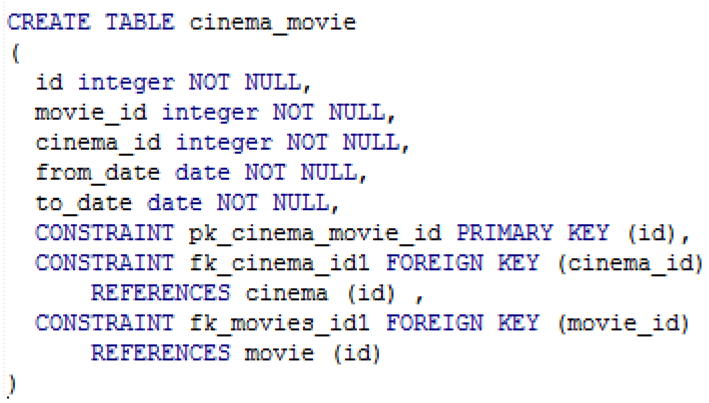
	id integer	description character varying(50)	price_per_seat numeric
1	1	Normal Cinema	90
2	2	3D Cinema	150
3	3	Normal/3D Cinema	150
4	4	Imax	225
5	5	Imax 3D	255.5
6	6	Large Cinema	55

Cinema_movie

<u>Purpose:</u>

Cinema movie describes the type of movies from release date to end date of the movie.

<u>Create statement:</u>

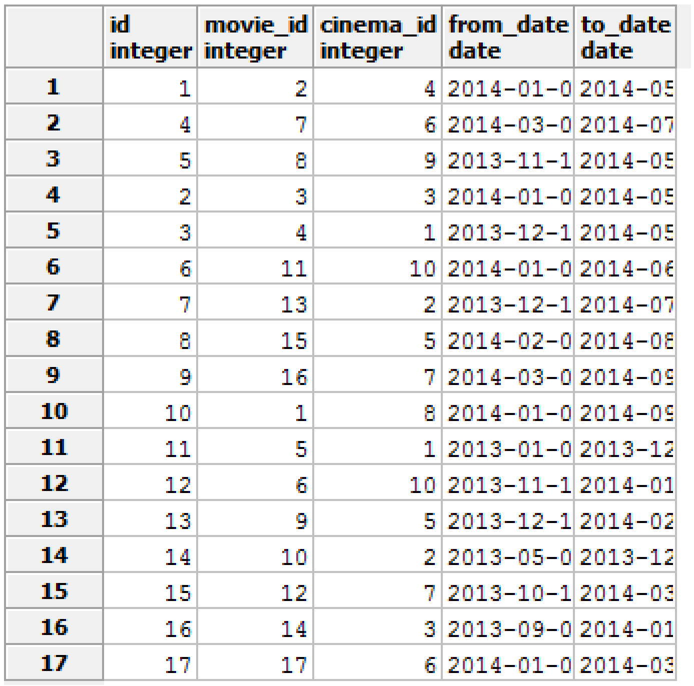


```
CREATE TABLE cinema_movie
(
  id integer NOT NULL,
  movie_id integer NOT NULL,
  cinema_id integer NOT NULL,
  from_date date NOT NULL,
  to_date date NOT NULL,
  CONSTRAINT pk_cinema_movie_id PRIMARY KEY (id),
  CONSTRAINT fk_cinema_id1 FOREIGN KEY (cinema_id)
      REFERENCES cinema (id) ,
  CONSTRAINT fk_movies_id1 FOREIGN KEY (movie_id)
      REFERENCES movie (id)
)
```

<u>Functional dependencies:</u>

id $\longrightarrow$ movie_id , cinema_id , from_date , to_date

<u>Sample data:</u>



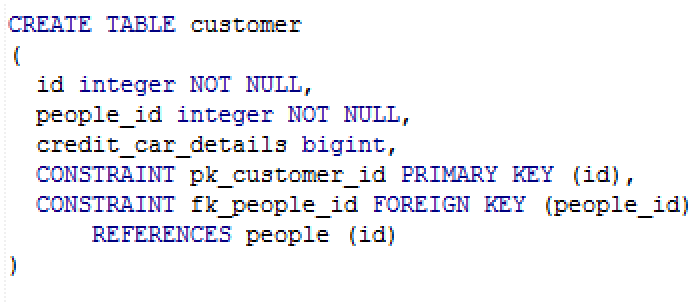
	id integer	movie_id integer	cinema_id integer	from_date date	to_date date
1	1	2	4	2014-01-0	2014-05
2	4	7	6	2014-03-0	2014-07
3	5	8	9	2013-11-1	2014-05
4	2	3	3	2014-01-0	2014-05
5	3	4	1	2013-12-1	2014-05
6	6	11	10	2014-01-0	2014-06
7	7	13	2	2013-12-1	2014-07
8	8	15	5	2014-02-0	2014-08
9	9	16	7	2014-03-0	2014-09
10	10	1	8	2014-01-0	2014-09
11	11	5	1	2013-01-0	2013-12
12	12	6	10	2013-11-1	2014-01
13	13	9	5	2013-12-1	2014-02
14	14	10	2	2013-05-0	2013-12
15	15	12	7	2013-10-1	2014-03
16	16	14	3	2013-09-0	2014-01
17	17	17	6	2014-01-0	2014-03

Customer

Purpose:

Customer is a person who intends to buy a ticket using credit details.

Create statement:

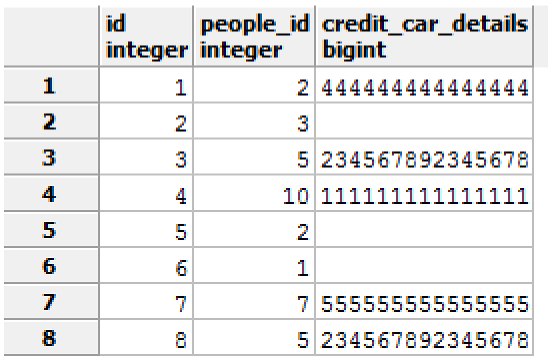


```
CREATE TABLE customer
(
  id integer NOT NULL,
  people_id integer NOT NULL,
  credit_car_details bigint,
  CONSTRAINT pk_customer_id PRIMARY KEY (id),
  CONSTRAINT fk_people_id FOREIGN KEY (people_id)
      REFERENCES people (id)
)
```

Functional dependencies:

id $\longrightarrow$ people_id , credit_car_details

Sample data:



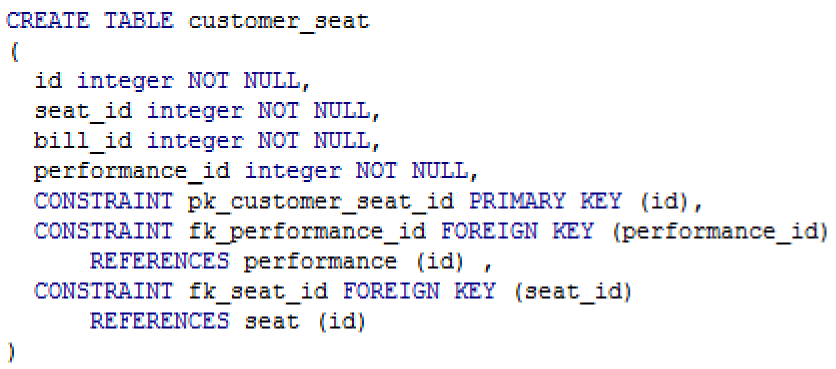
	id integer	people_id integer	credit_car_details bigint
1	1	2	444444444444444
2	2	3	
3	3	5	234567892345678
4	4	10	111111111111111
5	5	2	
6	6	1	
7	7	7	555555555555555
8	8	5	234567892345678

Customer_seat

Purpose:

The customer choose the seat depends upon the seat number, bill id , performance id.

Create statement:

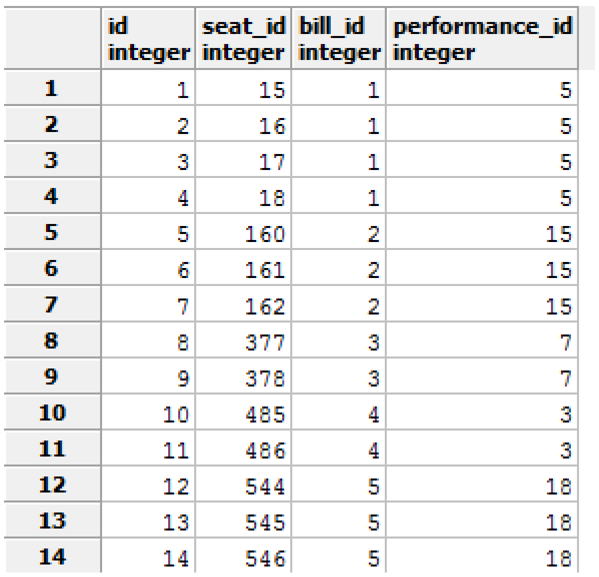


```
CREATE TABLE customer_seat
(
  id integer NOT NULL,
  seat_id integer NOT NULL,
  bill_id integer NOT NULL,
  performance_id integer NOT NULL,
  CONSTRAINT pk_customer_seat_id PRIMARY KEY (id),
  CONSTRAINT fk_performance_id FOREIGN KEY (performance_id)
      REFERENCES performance (id) ,
  CONSTRAINT fk_seat_id FOREIGN KEY (seat_id)
      REFERENCES seat (id)
)
```

Functional dependencies:

id $\rightarrow$ seat_id , bill_id , performance_id

Sample data:



	id integer	seat_id integer	bill_id integer	performance_id integer
1	1	15	1	5
2	2	16	1	5
3	3	17	1	5
4	4	18	1	5
5	5	160	2	15
6	6	161	2	15
7	7	162	2	15
8	8	377	3	7
9	9	378	3	7
10	10	485	4	3
11	11	486	4	3
12	12	544	5	18
13	13	545	5	18
14	14	546	5	18

Movie

Purpose:

Movie tells us about the movie names with rating and movie duration.

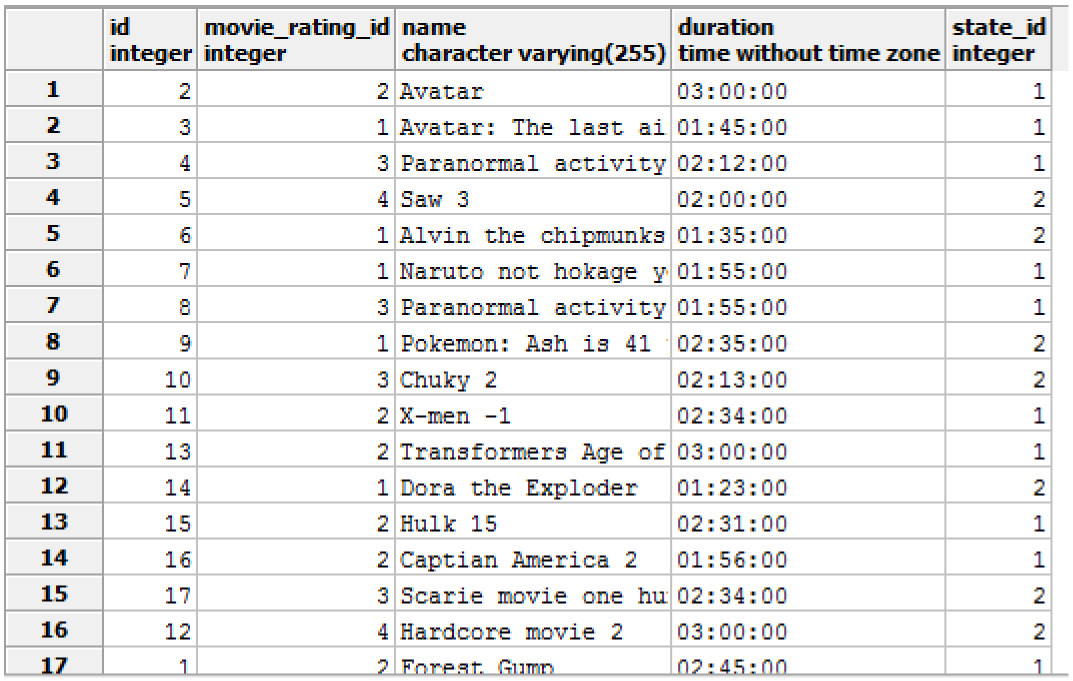
Create statement:

```
CREATE TABLE movie
(
  id integer NOT NULL,
  movie_rating_id integer NOT NULL,
  name character varying(255) NOT NULL,
  duration time without time zone NOT NULL,
  state_id integer NOT NULL,
  CONSTRAINT pk_movie_id PRIMARY KEY (id),
  CONSTRAINT fk_movie_rating_id FOREIGN KEY (movie_rating_id)
      REFERENCES movie_rating (id) ,
  CONSTRAINT fk_state_id FOREIGN KEY (state_id)
      REFERENCES state (id)
)
```

Functional dependencies:

id $\longrightarrow$ movie_rating_id , name , duration , state_id

Sample data:



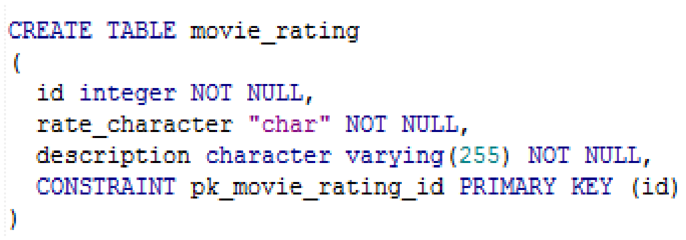
	id integer	movie_rating_id integer	name character varying(255)	duration time without time zone	state_id integer
1	2	2	Avatar	03:00:00	1
2	3	1	Avatar: The last ai	01:45:00	1
3	4	3	Paranormal activity	02:12:00	1
4	5	4	Saw 3	02:00:00	2
5	6	1	Alvin the chipmunks	01:35:00	2
6	7	1	Naruto not hokage y	01:55:00	1
7	8	3	Paranormal activity	01:55:00	1
8	9	1	Pokemon: Ash is 41	02:35:00	2
9	10	3	Chuky 2	02:13:00	2
10	11	2	X-men -1	02:34:00	1
11	13	2	Transformers Age of	03:00:00	1
12	14	1	Dora the Exploder	01:23:00	2
13	15	2	Hulk 15	02:31:00	1
14	16	2	Captian America 2	01:56:00	1
15	17	3	Scarie movie one hu	02:34:00	2
16	12	4	Hardcore movie 2	03:00:00	2
17	1	2	Forest Gump	02:45:00	1

Movie_rating

Purpose:

It describes about the category of the movie like for everyone, for everyone and child, for +17.

Create statement:

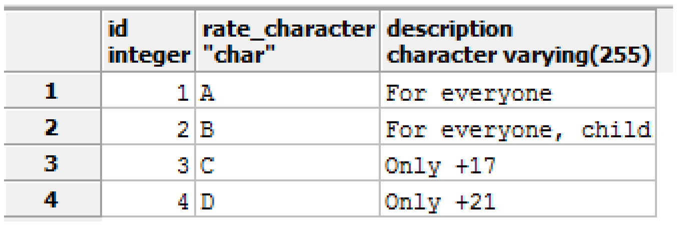


```
CREATE TABLE movie_rating
(
  id integer NOT NULL,
  rate_character "char" NOT NULL,
  description character varying(255) NOT NULL,
  CONSTRAINT pk_movie_rating_id PRIMARY KEY (id)
)
```

Functional dependencies:

id $\longrightarrow$ rate_character , description

Sample data:



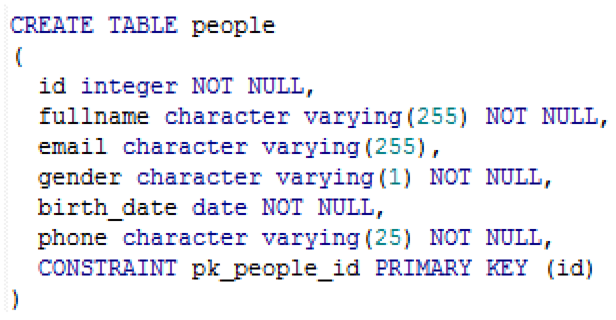
	id integer	rate_character "char"	description character varying(255)
1	1	A	For everyone
2	2	B	For everyone, child
3	3	C	Only +17
4	4	D	Only +21

People

Purpose:

People gives the information about the customers who watching movie such as email, gender.

Create statement:

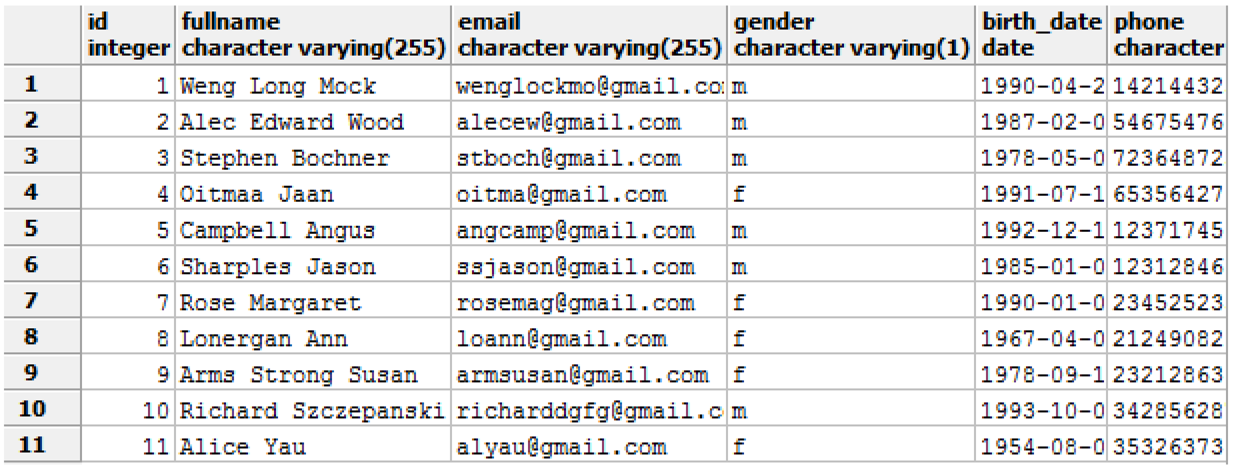


```
CREATE TABLE people
(
  id integer NOT NULL,
  fullname character varying(255) NOT NULL,
  email character varying(255),
  gender character varying(1) NOT NULL,
  birth_date date NOT NULL,
  phone character varying(25) NOT NULL,
  CONSTRAINT pk_people_id PRIMARY KEY (id)
)
```

Functional dependencies:

id $\longrightarrow$ fullname , email , gender , birth_date , phone

Sample data:



	id integer	fullname character varying(255)	email character varying(255)	gender character varying(1)	birth_date date	phone character
1	1	Weng Long Mock	wenglockmo@gmail.co	m	1990-04-2	14214432
2	2	Alec Edward Wood	alecew@gmail.com	m	1987-02-0	54675476
3	3	Stephen Bochner	stboch@gmail.com	m	1978-05-0	72364872
4	4	Oitmaa Jaan	oitma@gmail.com	f	1991-07-1	65356427
5	5	Campbell Angus	angcamp@gmail.com	m	1992-12-1	12371745
6	6	Sharples Jason	ssjason@gmail.com	m	1985-01-0	12312846
7	7	Rose Margaret	rosemag@gmail.com	f	1990-01-0	23452523
8	8	Lonergan Ann	loann@gmail.com	f	1967-04-0	21249082
9	9	Arms Strong Susan	armsusan@gmail.com	f	1978-09-1	23212863
10	10	Richard Szczepanski	richarddgfg@gmail.c	m	1993-10-0	34285628
11	11	Alice Yau	alyau@gmail.com	f	1954-08-0	35326373

Performance

Purpose:

It gives the information about the show timings of the movies in the theater.

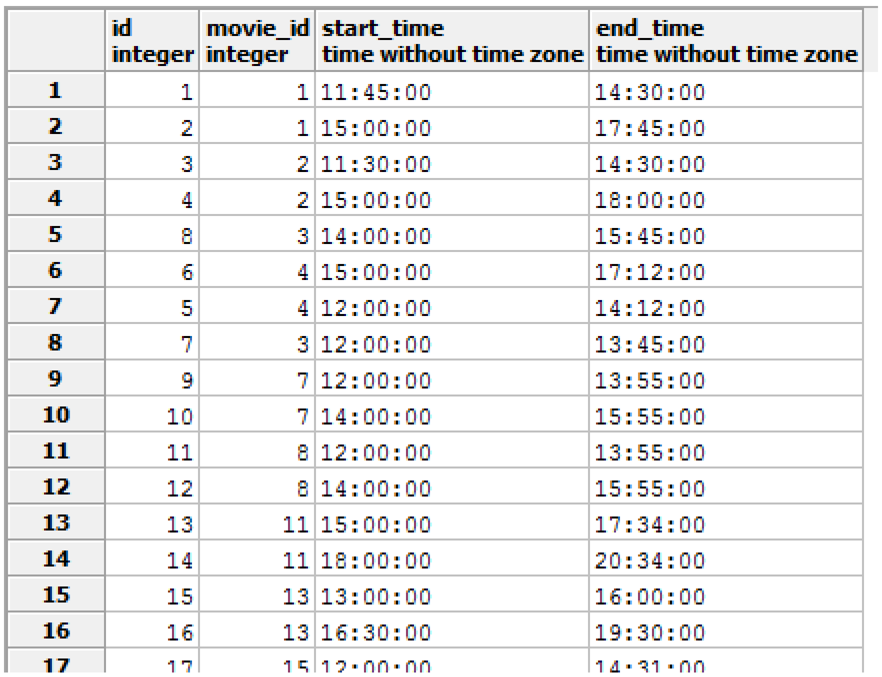
Create statement:

```
CREATE TABLE performance
(
  id integer NOT NULL,
  movie_id integer NOT NULL,
  start_time time without time zone NOT NULL,
  end_time time without time zone NOT NULL,
  CONSTRAINT pk_performance_id PRIMARY KEY (id),
  CONSTRAINT fk_movie_id2 FOREIGN KEY (movie_id)
      REFERENCES movie (id)
)
```

Functional dependencies:

id $\longrightarrow$ movie_id , start_time , end_time

Sample data:



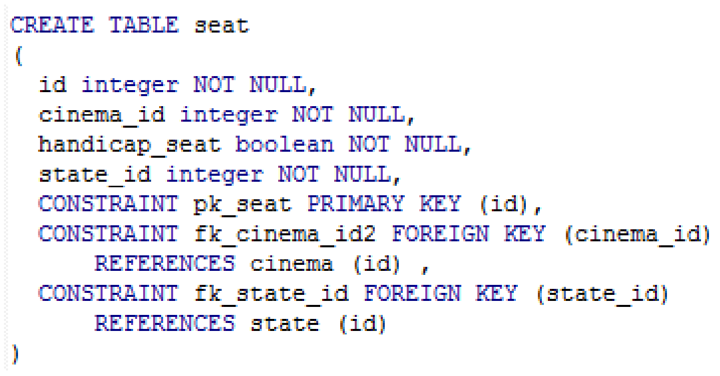
	id integer	movie_id integer	start_time time without time zone	end_time time without time zone
1	1	1	11:45:00	14:30:00
2	2	1	15:00:00	17:45:00
3	3	2	11:30:00	14:30:00
4	4	2	15:00:00	18:00:00
5	8	3	14:00:00	15:45:00
6	6	4	15:00:00	17:12:00
7	5	4	12:00:00	14:12:00
8	7	3	12:00:00	13:45:00
9	9	7	12:00:00	13:55:00
10	10	7	14:00:00	15:55:00
11	11	8	12:00:00	13:55:00
12	12	8	14:00:00	15:55:00
13	13	11	15:00:00	17:34:00
14	14	11	18:00:00	20:34:00
15	15	13	13:00:00	16:00:00
16	16	13	16:30:00	19:30:00
17	17	15	12:00:00	14:31:00

Seat

Purpose:

The person must choose the seat with respective to the cinema id, state id.

Create statement:

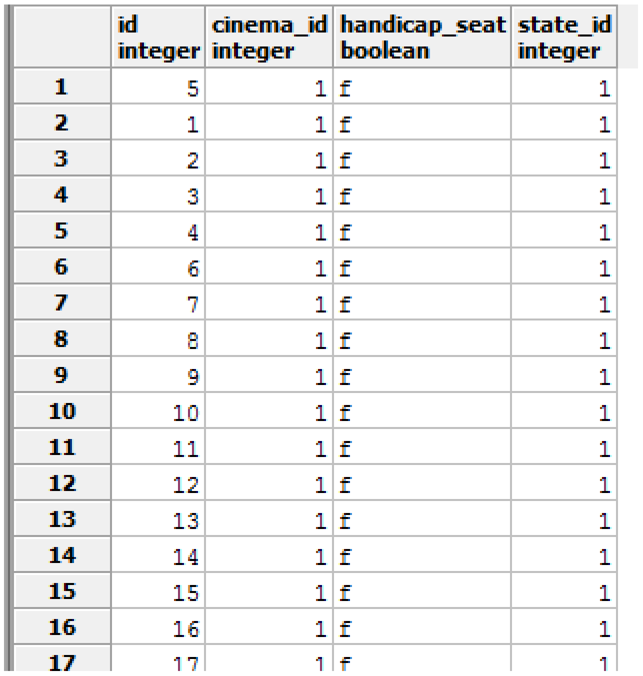


```
CREATE TABLE seat
(
  id integer NOT NULL,
  cinema_id integer NOT NULL,
  handicap_seat boolean NOT NULL,
  state_id integer NOT NULL,
  CONSTRAINT pk_seat PRIMARY KEY (id),
  CONSTRAINT fk_cinema_id2 FOREIGN KEY (cinema_id)
      REFERENCES cinema (id) ,
  CONSTRAINT fk_state_id FOREIGN KEY (state_id)
      REFERENCES state (id)
)
```

Functional dependencies:

id $\longrightarrow$ cinema_id , handicap_seat , state_id

Sample data:



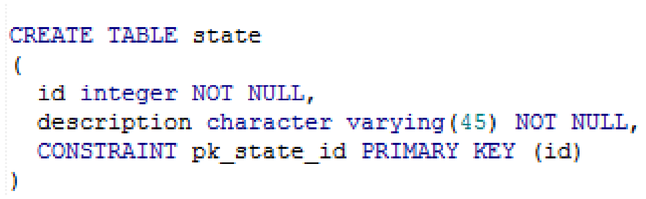
	id integer	cinema_id integer	handicap_seat boolean	state_id integer
1	5	1	f	1
2	1	1	f	1
3	2	1	f	1
4	3	1	f	1
5	4	1	f	1
6	6	1	f	1
7	7	1	f	1
8	8	1	f	1
9	9	1	f	1
10	10	1	f	1
11	11	1	f	1
12	12	1	f	1
13	13	1	f	1
14	14	1	f	1
15	15	1	f	1
16	16	1	f	1
17	17	1	f	1

State

Purpose:

Statement shows that movie is available in respective state or not.

Create statement:

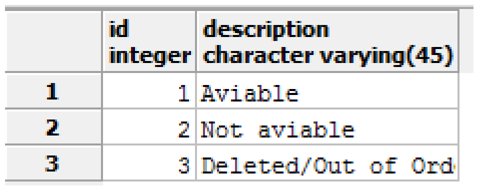


```
CREATE TABLE state
(
  id integer NOT NULL,
  description character varying(45) NOT NULL,
  CONSTRAINT pk_state_id PRIMARY KEY (id)
)
```

Functional dependencies:

id $\rightarrow$ description

Sample data:



	id integer	description character varying(45)
1	1	Aviable
2	2	Not aviable
3	3	Deleted/Out of Ord

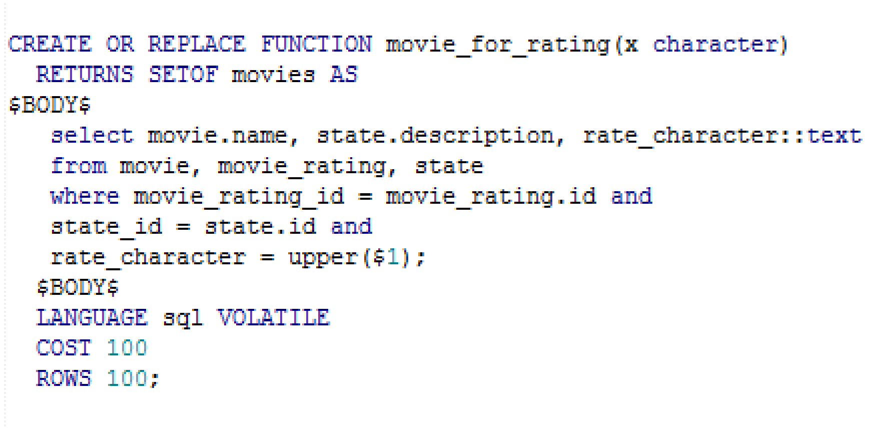
Stored procedures

Movie for rating

Purpose:

Movie for rating describes the type of movie playing in the respective states with good collections.

Query:



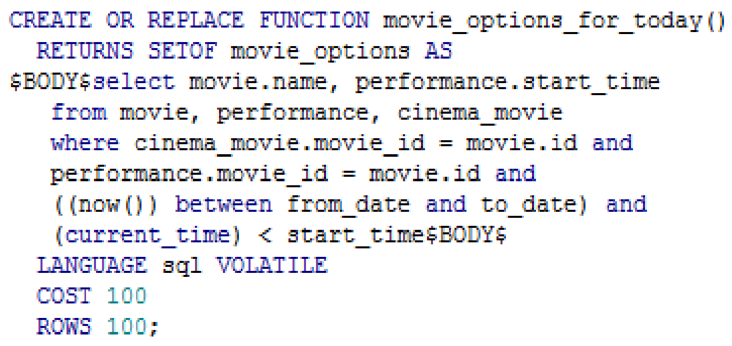
```
CREATE OR REPLACE FUNCTION movie_for_rating(x character)
  RETURNS SETOF movies AS
$BODY$
   select movie.name, state.description, rate_character::text
   from movie, movie_rating, state
   where movie_rating_id = movie_rating.id and
   state_id = state.id and
   rate_character = upper($1);
  $BODY$
  LANGUAGE sql VOLATILE
  COST 100
  ROWS 100;
```

Movies options for today

Purpose:

It describes the movie options for playing today in respective screens with respective movies in the theater

Query:

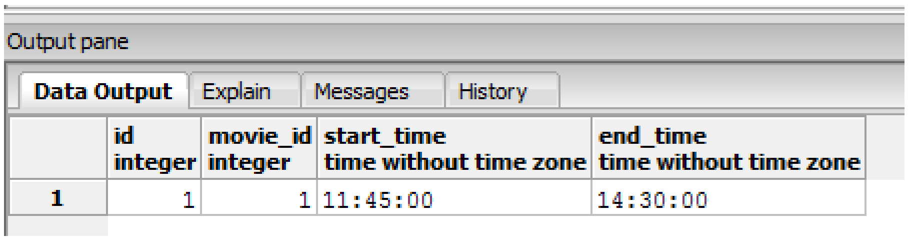


```
CREATE OR REPLACE FUNCTION movie_options_for_today()
  RETURNS SETOF movie_options AS
$BODY$select movie.name, performance.start_time
   from movie, performance, cinema_movie
   where cinema_movie.movie_id = movie.id and
   performance.movie_id = movie.id and
   ((now()) between from_date and to_date) and
   (current_time) < start_time$BODY$
  LANGUAGE sql VOLATILE
  COST 100
  ROWS 100;
```

Queries

```
select * from performance
where id in (select s.id
from seat s
join customer_seat cs
on s.id=cs.id and cs.seat_id=15 )
```

Output



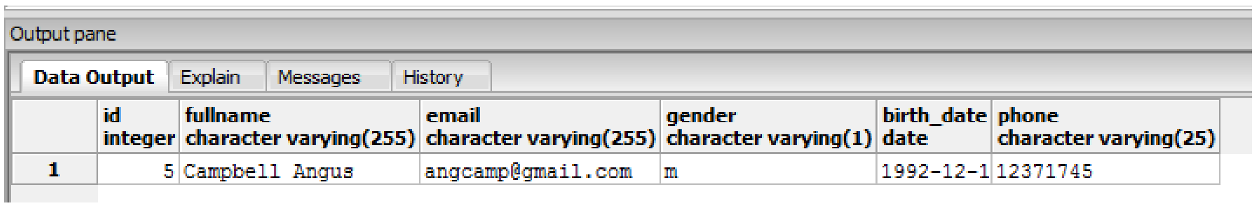
Output pane

Data Output | Explain | Messages | History

	id integer	movie_id integer	start_time time without time zone	end_time time without time zone
1	1	1	11:45:00	14:30:00

```
select * from people
where id in (select c.id
from customer c
join bill b
on c.id=b.id and b.customer_id=5 )
```

Output



Output pane

Data Output | Explain | Messages | History

	id integer	fullname character varying(255)	email character varying(255)	gender character varying(1)	birth_date date	phone character varying(25)
1	5	Campbell Angus	angcamp@gmail.com	m	1992-12-1	12371745

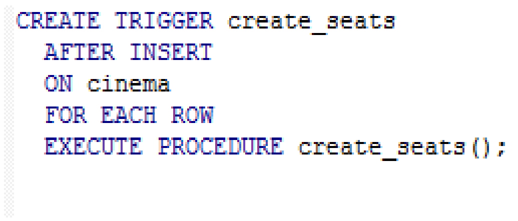
Triggers

Create_seats on cinema

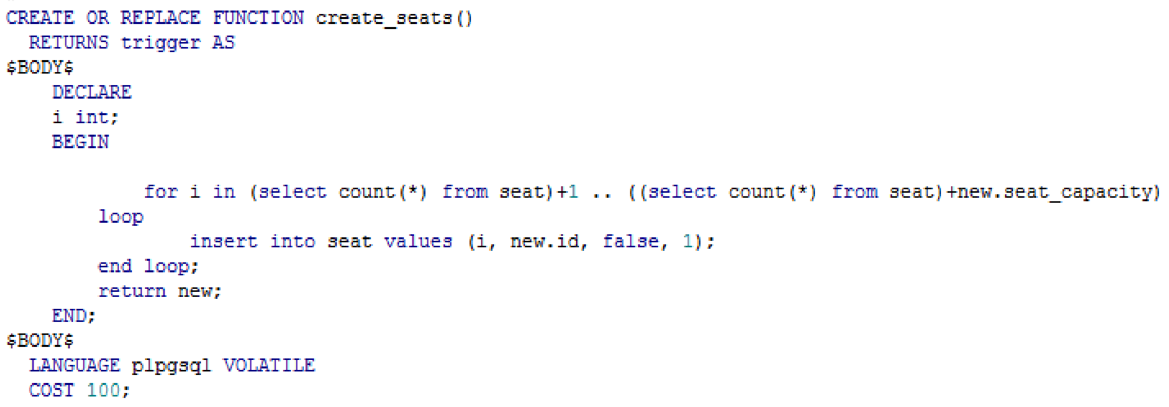
Purpose:

In the create seats query the trigger is used to create the respective seats depends on the capacity of seats in the theater.

Query:



```
CREATE TRIGGER create_seats
  AFTER INSERT
  ON cinema
  FOR EACH ROW
  EXECUTE PROCEDURE create_seats();
```



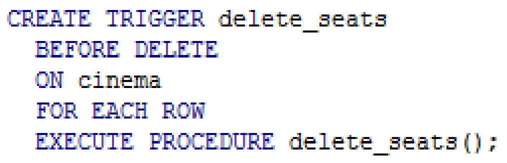
```
CREATE OR REPLACE FUNCTION create_seats()
  RETURNS trigger AS
$BODY$
    DECLARE
    i int;
    BEGIN

            for i in (select count(*) from seat)+1 .. ((select count(*) from seat)+new.seat_capacity)
        loop
                insert into seat values (i, new.id, false, 1);
        end loop;
        return new;
    END;
$BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
```

Delete_seats on cinema

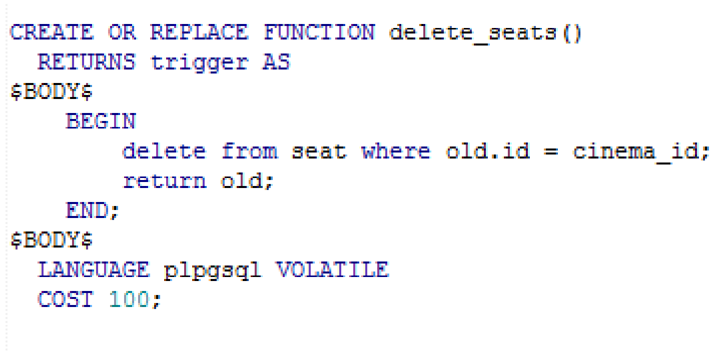
Purpose:

The delete seats trigger describes the seats deleted for the cinema means reducing the no of persons.

Query:



```
CREATE TRIGGER delete_seats
  BEFORE DELETE
  ON cinema
  FOR EACH ROW
  EXECUTE PROCEDURE delete_seats();
```



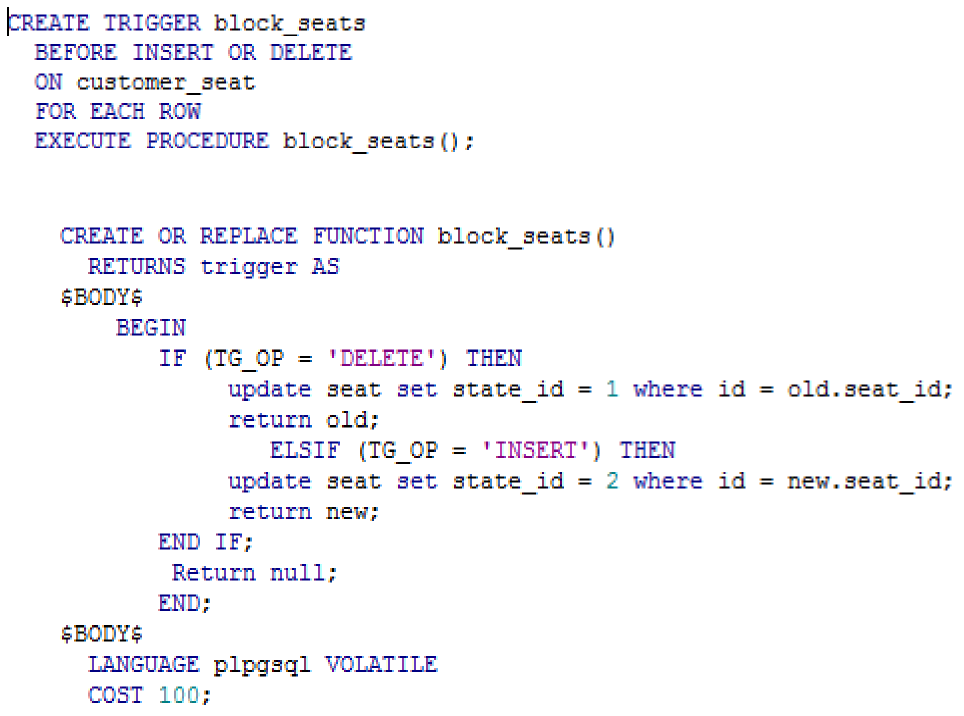
```
CREATE OR REPLACE FUNCTION delete_seats()
  RETURNS trigger AS
$BODY$
    BEGIN
        delete from seat where old.id = cinema_id;
        return old;
    END;
$BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
```

Block_seats

Purpose:

In the block seats describes the removal of old and replacing with the new seats means one type of update.

Query:



```
CREATE TRIGGER block_seats
  BEFORE INSERT OR DELETE
  ON customer_seat
  FOR EACH ROW
  EXECUTE PROCEDURE block_seats();


    CREATE OR REPLACE FUNCTION block_seats()
      RETURNS trigger AS
    $BODY$
        BEGIN
           IF (TG_OP = 'DELETE') THEN
                update seat set state_id = 1 where id = old.seat_id;
                return old;
                   ELSIF (TG_OP = 'INSERT') THEN
                update seat set state_id = 2 where id = new.seat_id;
                return new;
           END IF;
            Return null;
           END;
    $BODY$
      LANGUAGE plpgsql VOLATILE
      COST 100;
```

Security

Create role manager

Grant update, insert, select

On all tables in schema public

To manager

Known problems

a. The movie which hit the screens not only depend on movie rating , we also consider the movie reviews
b. From now we have un security regarding to the people coming for cinema by just checking their mobile number.
c. The seating capacity depending upon the type of variety like gold, silver, diamond. Here our income will reduces

Future improvements

a. We must maintain the average seating capacity for all the range of variety like gold, diamond, silver and earn more income.
b. The picture we are choosing from movies not entirely depends on movie ratings, we must consider all the factors.
c. The people who are coming for watching cinema they must bring their id like driving license, ssn number and state id for more security.