A Project Report on
Theatre Management System

Developed By:

IT155 Kashyap Sonraj

IT156 Deep Sutariya

IT170 Mehul Zala

Guided By

Internal Guide:

Prof. Archana N. Vyas

Department of Information Technology

Faculty of Technology

DD University

DHARMSINH DESAI UNIVERSITY

Department of Information Technology Faculty of Technology,

Dharmsinh Desai University College Road, Nadiad-387001

October-2022

DHARMSINH DESAI UNIVERSITY

NADIAD-387001, GUJARAT

CERTIFICATE

This is to certify that the project entitled “Theatre management System” is a bonafide report of the work carried out by

1) Kashyap Sonraj Student ID No: 21ITUBD006

2) Deep Sutariya Student ID No: 20ITUOS133

3) Mehul Zala Student ID No: 20ITUBS144

of Department of Information Technology, semester V, under the guidance and supervision for the subject Database Management System. They were involved in Project training during the academic year 2022-2023.

Prof. Archana N. Vyas
Project Guide, Department of Information Technology,
Faculty of Technology,
Dharmsinh Desai University, Nadiad
Date: 15/10/2022

Prof. Vipul Dabhi
Head, Department of Information Technology

COMMENDATION

We would like to express our heartfelt gratitude to everyone who contributed to the successful completion of our project "Theatre Management System"

The success and ultimate conclusion of this project necessitated a great deal of advice and support from a large number of individuals, and we are incredibly fortunate to have received it all along with the project's completion.

We owe a debt of appreciation to Prof. Archana N. Vyas, our project guide,who took an interest in our project work and directed us through it till it was completed by giving all of the required assistance for creating a solid Database System.

We'd also want to express our gratitude to all of our speakers. Finally, we express our gratitude to all of our friends and colleagues

INDEX

I. Certificate……………………………………………………………………………………………..I

II. Commendation………………………………………………………………………………………II

1. SYSTEM OVERVIEW……………………………………………………………………………..5

1.1 Current system……………………………………………………………………………………..5

1.2 Objectives of the Proposed System…………………………………………………………….6

1.3 Advantages of the Proposed system (over current)……………………………………….6

2. E-R DIAGRAM………………………………………………………………………………………..7

3. SCHEMA DIAGRAM…………………………………………………………………………………8

4. DATA DICTIONARY. ………………………………………………………………………………9

5. DATABASE IMPLEMENTATION………………………………………………………………..14

5.1 Create Schema……………………………………………………………………………………..14

5.2 Insert Data values…………………………………………………………………………………..19

5.3 Queries (Based on basic DBMS constructs ,Joins & Sub-Queries)…………………..23

5.4 PL/SQL Blocks (Views)……………………………………………………………………………27

5.5 Functions & Triggers……………………………………………………………………………….28

5.6 Cursors……………………………………………………………………………………………….30

6. FUTURE ENHANCEMENTS OF THE SYSTEM…………………………………………………31

1 SYSYTEM OVERVIEW

1.1 CUREENT SYSTEM

Theatre Management System is to manage the details of Shows, Booking, Payment,Movie, Customer. It manages all the information about Shows, Seats, Customer, Shows. The project is totally built at administrative end and thus only the administrator is guaranteed the access. The purpose of the project is to build an application program to reduce the manual work for managing the Shows, Booking,Seats, Payment. It tracks all the details about the Payment,Movie, Customer.

1.2 OBJECTIVE OF PROPOSED SYSTEM

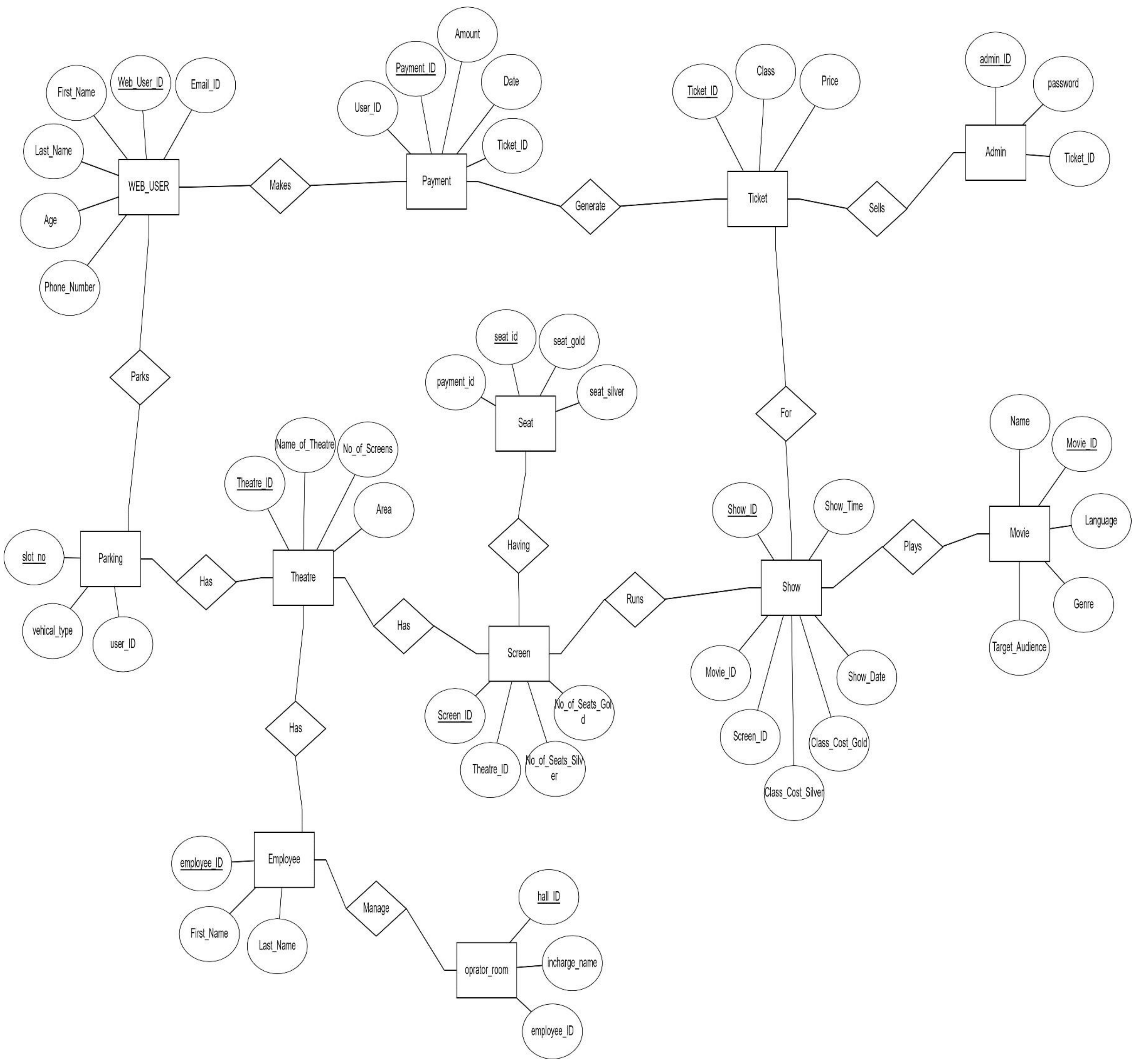
In current tech era most of things happen using internet so with this system user can able to book ticket show seat ,parking slots and can able to book tivket for specific seat and specific movie.

ADVANTAGE OF THE PROPOSED SYSTEM:

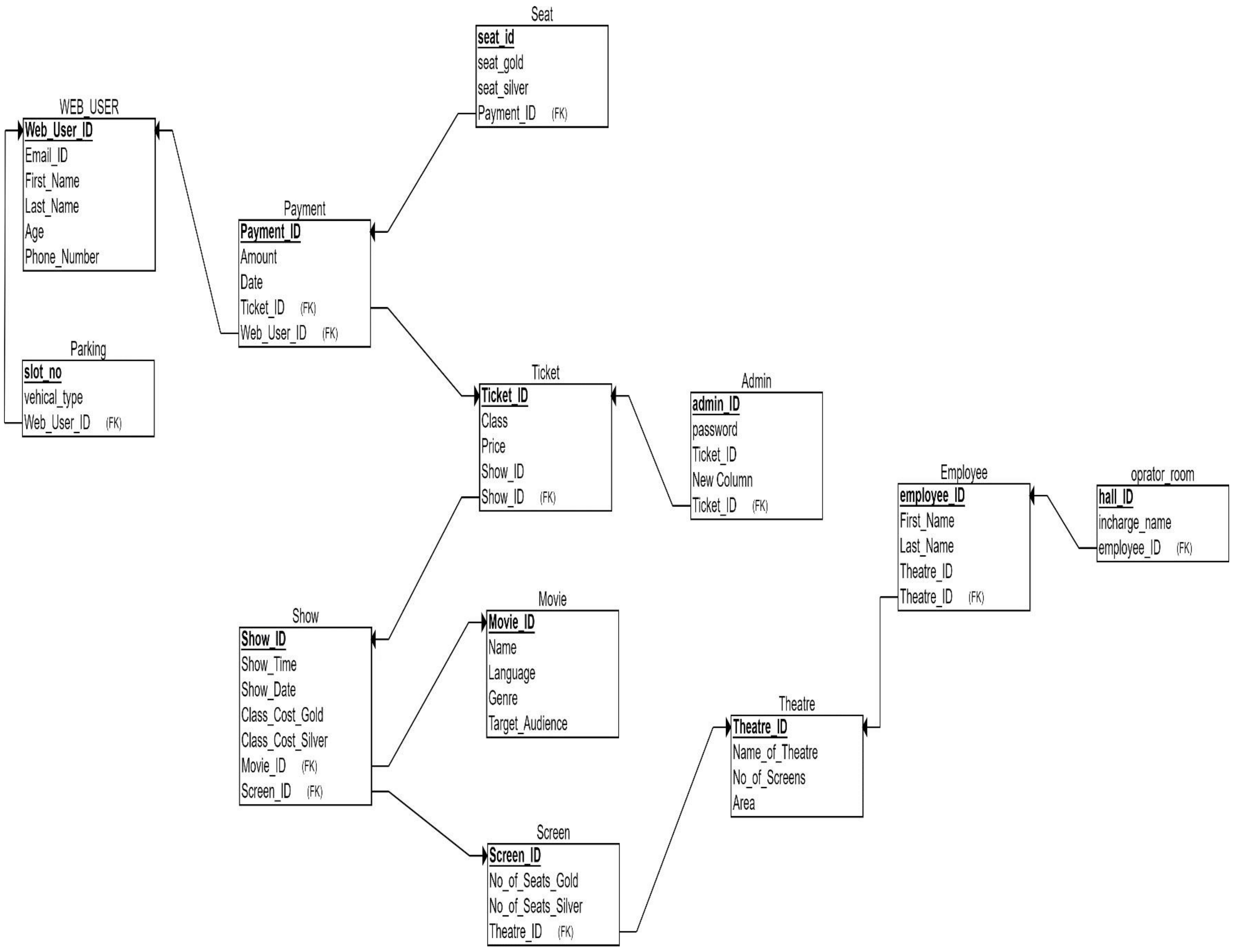
- User can able to book tickets online on theatre can able to buy gold silver tickets and also able to check parking slot is available or not.

- Admin can able to analyze how many users come at a day. Which user visits how many times. Admin can able to do all analization of data with this system.

2. ENTITY-RELATIONSHIP MODEL

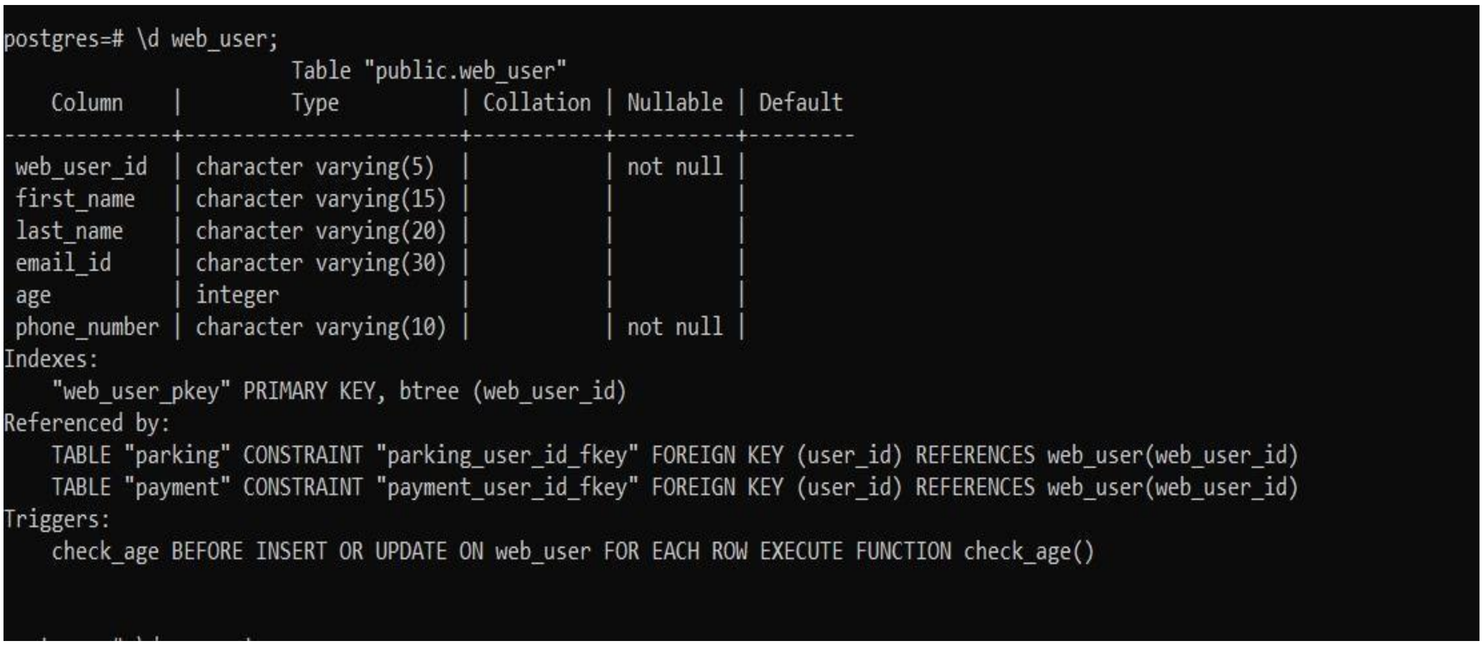


3. RELATIONAL SCHEMA



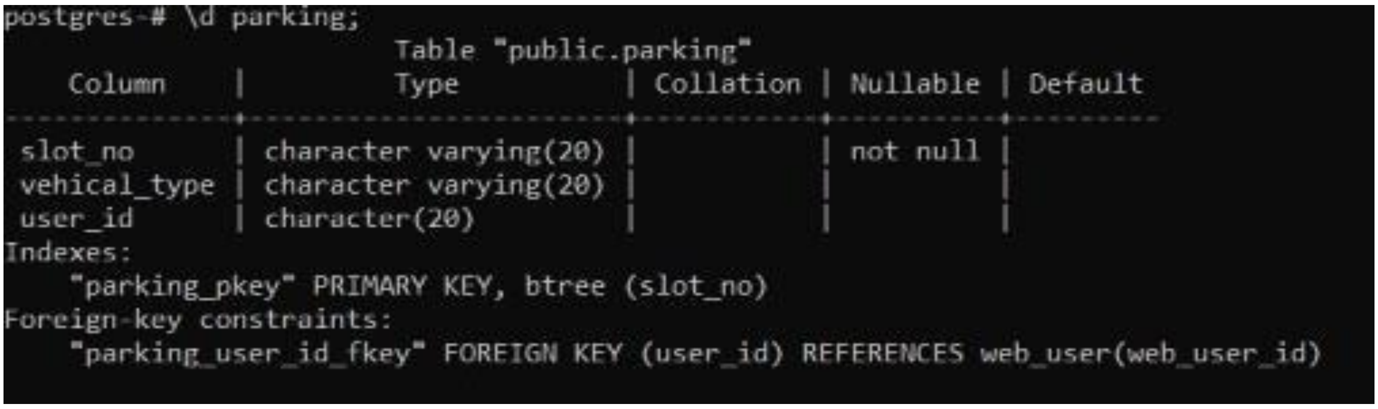
4.DATA DICTIONARY

4.1 web_user



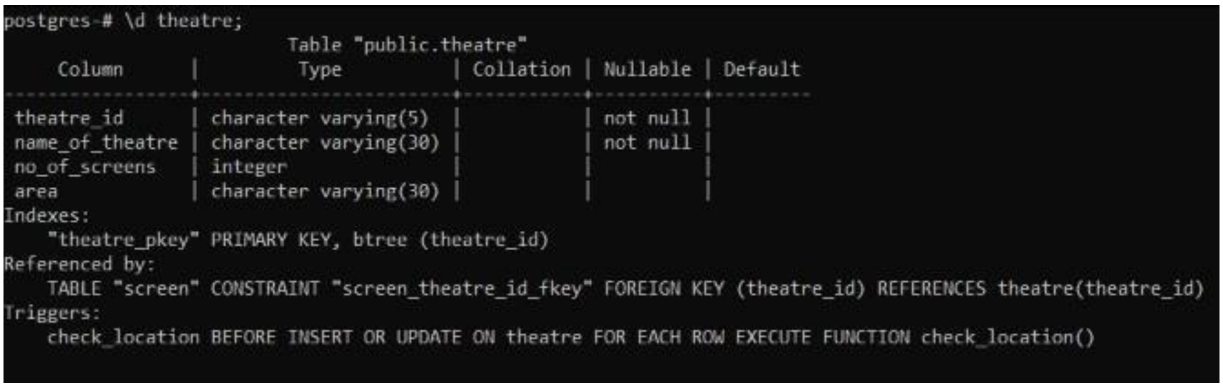
```
postgres=# \d web_user;
                        Table "public.web_user"
    Column    |         Type          | Collation | Nullable | Default
--------------+-----------------------+-----------+----------+---------
 web_user_id  | character varying(5)  |           | not null |
 first_name   | character varying(15) |           |          |
 last_name    | character varying(20) |           |          |
 email_id     | character varying(30) |           |          |
 age          | integer               |           |          |
 phone_number | character varying(10) |           | not null |
Indexes:
    "web_user_pkey" PRIMARY KEY, btree (web_user_id)
Referenced by:
    TABLE "parking" CONSTRAINT "parking_user_id_fkey" FOREIGN KEY (user_id) REFERENCES web_user(web_user_id)
    TABLE "payment" CONSTRAINT "payment_user_id_fkey" FOREIGN KEY (user_id) REFERENCES web_user(web_user_id)
Triggers:
    check_age BEFORE INSERT OR UPDATE ON web_user FOR EACH ROW EXECUTE FUNCTION check_age()
```

4.2 parking



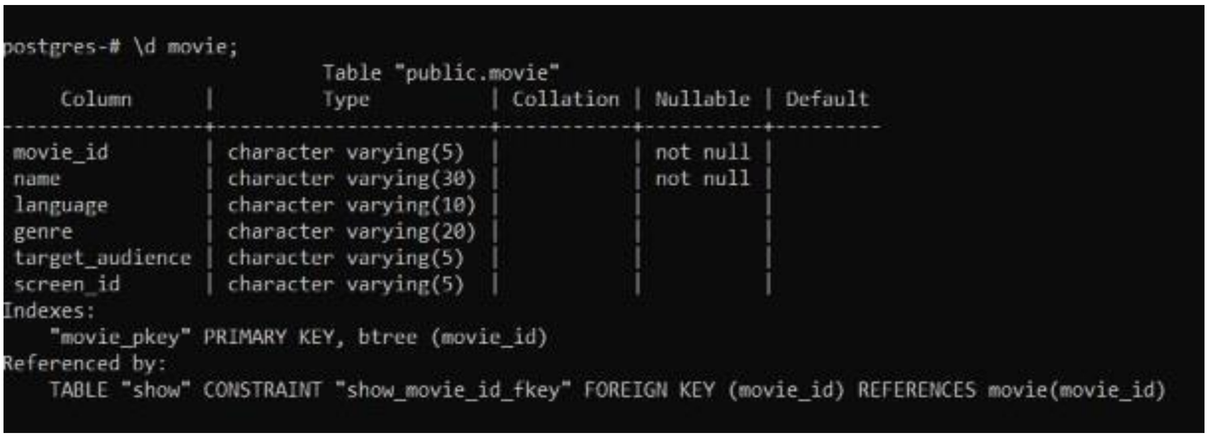
```
postgres-# \d parking;
                       Table "public.parking"
    Column    |         Type          | Collation | Nullable | Default
--------------+-----------------------+-----------+----------+---------
 slot_no      | character varying(20) |           | not null |
 vehical_type | character varying(20) |           |          |
 user_id      | character(20)         |           |          |
Indexes:
    "parking_pkey" PRIMARY KEY, btree (slot_no)
Foreign-key constraints:
    "parking_user_id_fkey" FOREIGN KEY (user_id) REFERENCES web_user(web_user_id)
```

4.3 theatre



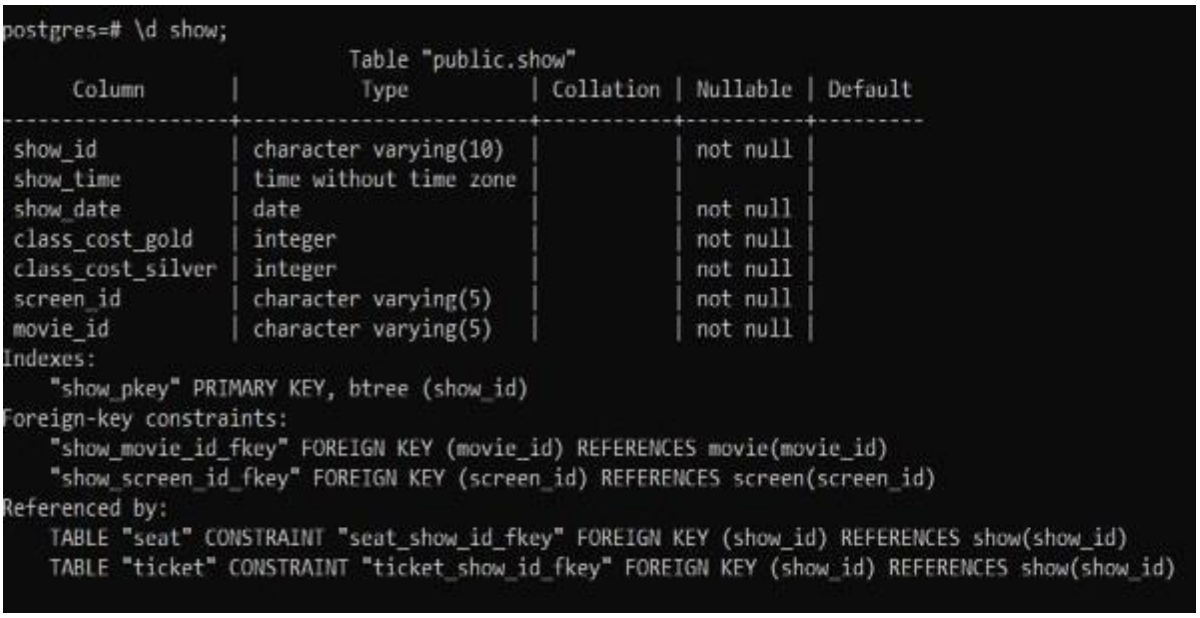
```
postgres-# \d theatre;
                        Table "public.theatre"
    Column      |         Type          | Collation | Nullable | Default
----------------+-----------------------+-----------+----------+---------
 theatre_id     | character varying(5)  |           | not null |
 name_of_theatre | character varying(30) |           | not null |
 no_of_screens  | integer               |           |          |
 area           | character varying(30) |           |          |
Indexes:
    "theatre_pkey" PRIMARY KEY, btree (theatre_id)
Referenced by:
    TABLE "screen" CONSTRAINT "screen_theatre_id_fkey" FOREIGN KEY (theatre_id) REFERENCES theatre(theatre_id)
Triggers:
    check_location BEFORE INSERT OR UPDATE ON theatre FOR EACH ROW EXECUTE FUNCTION check_location()
```

4.4 movie



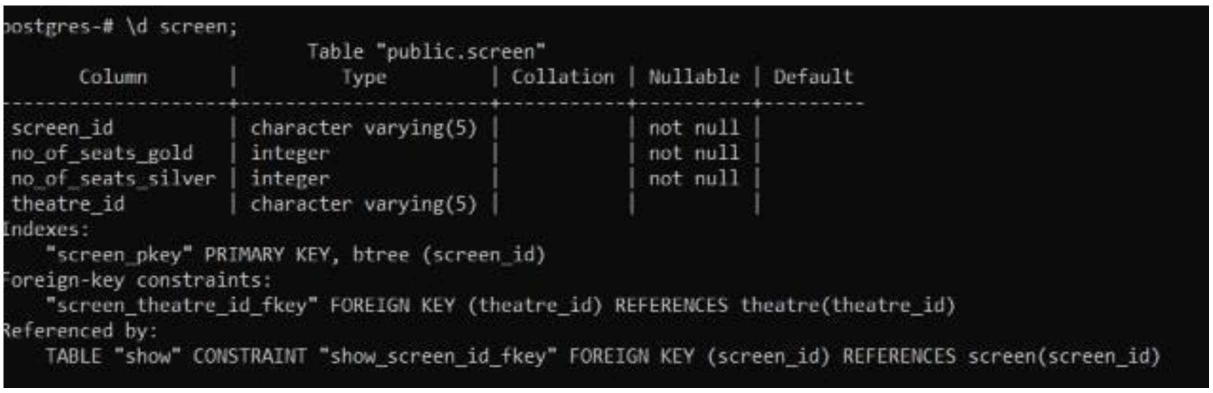
```
postgres-# \d movie;
                        Table "public.movie"
     Column      |         Type          | Collation | Nullable | Default
-----------------+-----------------------+-----------+----------+---------
 movie_id        | character varying(5)  |           | not null |
 name            | character varying(30) |           | not null |
 language        | character varying(10) |           |          |
 genre           | character varying(20) |           |          |
 target_audience | character varying(5)  |           |          |
 screen_id       | character varying(5)  |           |          |
Indexes:
    "movie_pkey" PRIMARY KEY, btree (movie_id)
Referenced by:
    TABLE "show" CONSTRAINT "show_movie_id_fkey" FOREIGN KEY (movie_id) REFERENCES movie(movie_id)
```

4.5 show



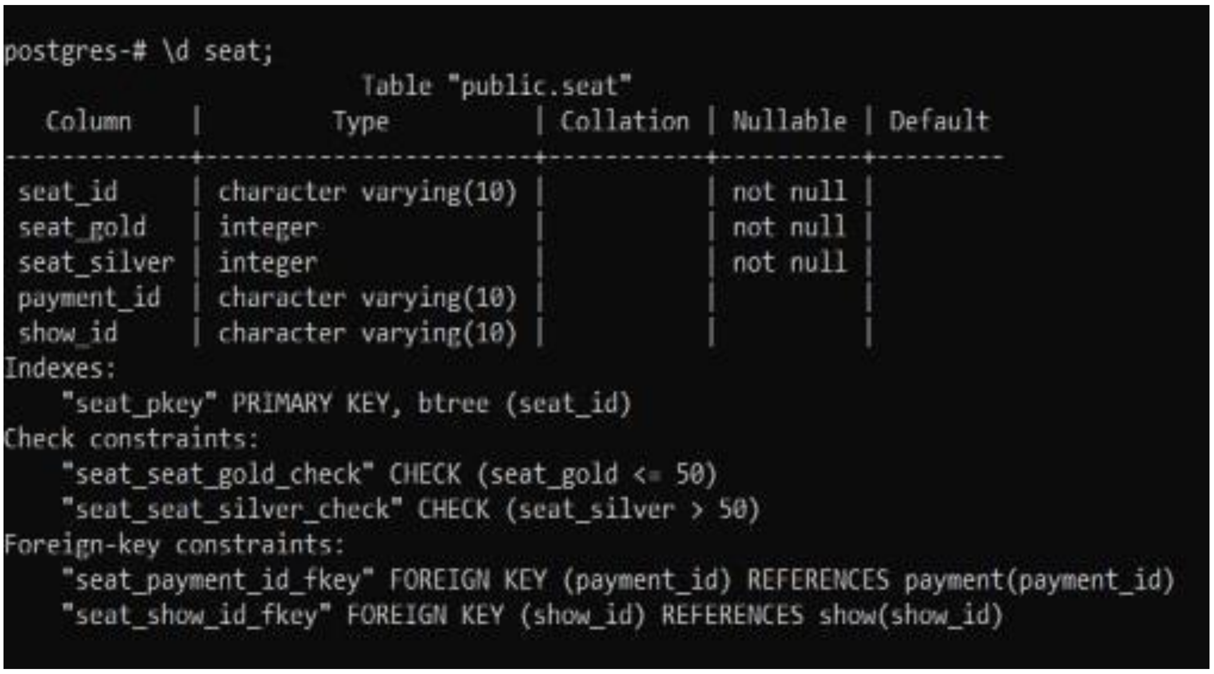
```
postgres=# \d show;
                        Table "public.show"
      Column       |          Type          | Collation | Nullable | Default
-------------------+------------------------+-----------+----------+---------
 show_id           | character varying(10)  |           | not null |
 show_time         | time without time zone |           |          |
 show_date         | date                   |           | not null |
 class_cost_gold   | integer                |           | not null |
 class_cost_silver | integer                |           | not null |
 screen_id         | character varying(5)   |           | not null |
 movie_id          | character varying(5)   |           | not null |
Indexes:
    "show_pkey" PRIMARY KEY, btree (show_id)
Foreign-key constraints:
    "show_movie_id_fkey" FOREIGN KEY (movie_id) REFERENCES movie(movie_id)
    "show_screen_id_fkey" FOREIGN KEY (screen_id) REFERENCES screen(screen_id)
Referenced by:
    TABLE "seat" CONSTRAINT "seat_show_id_fkey" FOREIGN KEY (show_id) REFERENCES show(show_id)
    TABLE "ticket" CONSTRAINT "ticket_show_id_fkey" FOREIGN KEY (show_id) REFERENCES show(show_id)
```

4.6 screen



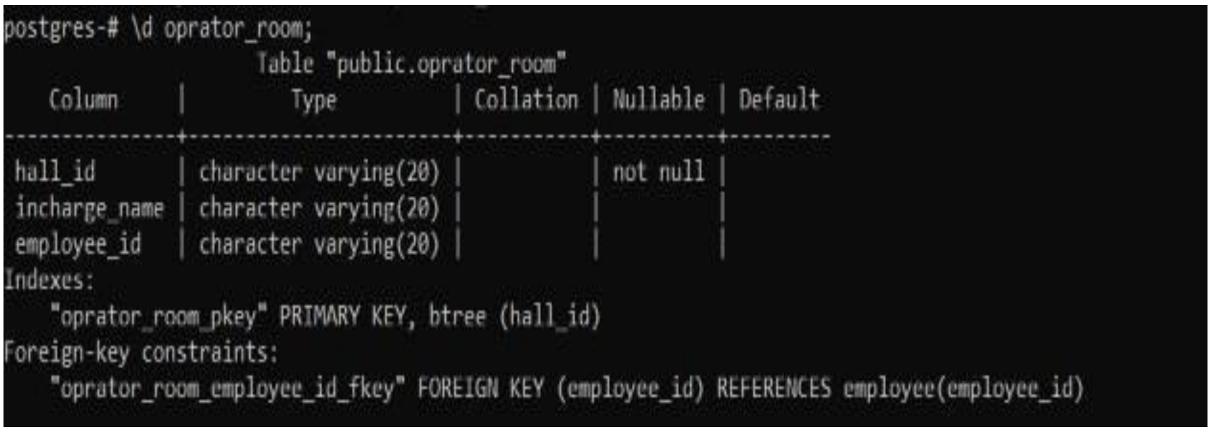
```
postgres-# \d screen;
                         Table "public.screen"
      Column       |         Type         | Collation | Nullable | Default
-------------------+----------------------+-----------+----------+---------
 screen_id         | character varying(5) |           | not null |
 no_of_seats_gold  | integer              |           | not null |
 no_of_seats_silver | integer             |           | not null |
 theatre_id        | character varying(5) |           |          |
Indexes:
    "screen_pkey" PRIMARY KEY, btree (screen_id)
Foreign-key constraints:
    "screen_theatre_id_fkey" FOREIGN KEY (theatre_id) REFERENCES theatre(theatre_id)
Referenced by:
    TABLE "show" CONSTRAINT "show_screen_id_fkey" FOREIGN KEY (screen_id) REFERENCES screen(screen_id)
```

4.7 seat



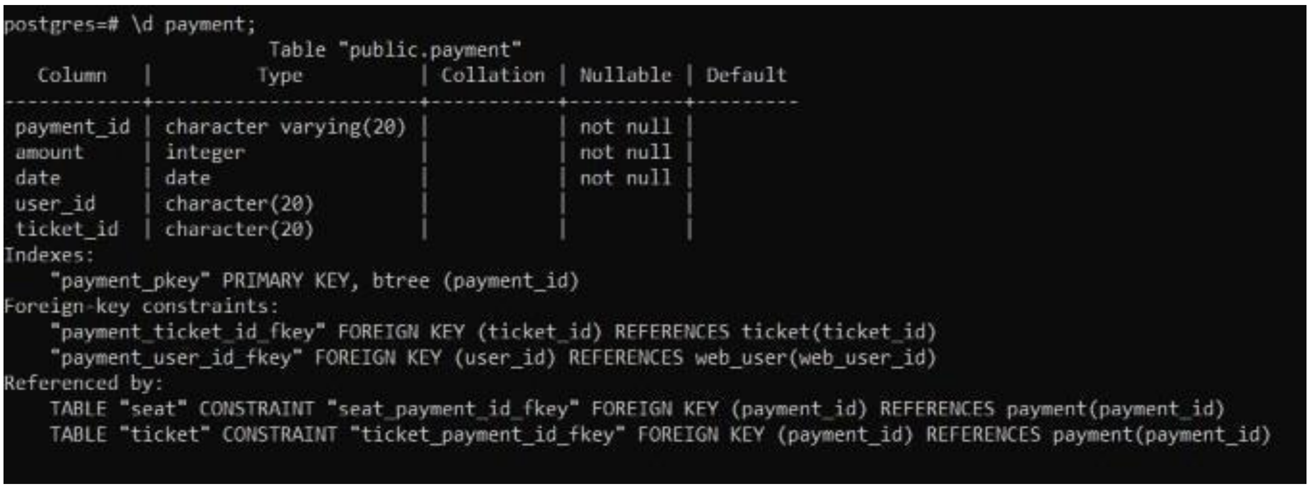
```
postgres-# \d seat;
                       Table "public.seat"
   Column    |         Type          | Collation | Nullable | Default
-------------+-----------------------+-----------+----------+---------
 seat_id     | character varying(10) |           | not null |
 seat_gold   | integer               |           | not null |
 seat_silver | integer               |           | not null |
 payment_id  | character varying(10) |           |          |
 show_id     | character varying(10) |           |          |
Indexes:
    "seat_pkey" PRIMARY KEY, btree (seat_id)
Check constraints:
    "seat_seat_gold_check" CHECK (seat_gold <= 50)
    "seat_seat_silver_check" CHECK (seat_silver > 50)
Foreign-key constraints:
    "seat_payment_id_fkey" FOREIGN KEY (payment_id) REFERENCES payment(payment_id)
    "seat_show_id_fkey" FOREIGN KEY (show_id) REFERENCES show(show_id)
```

4.8 oprator_room



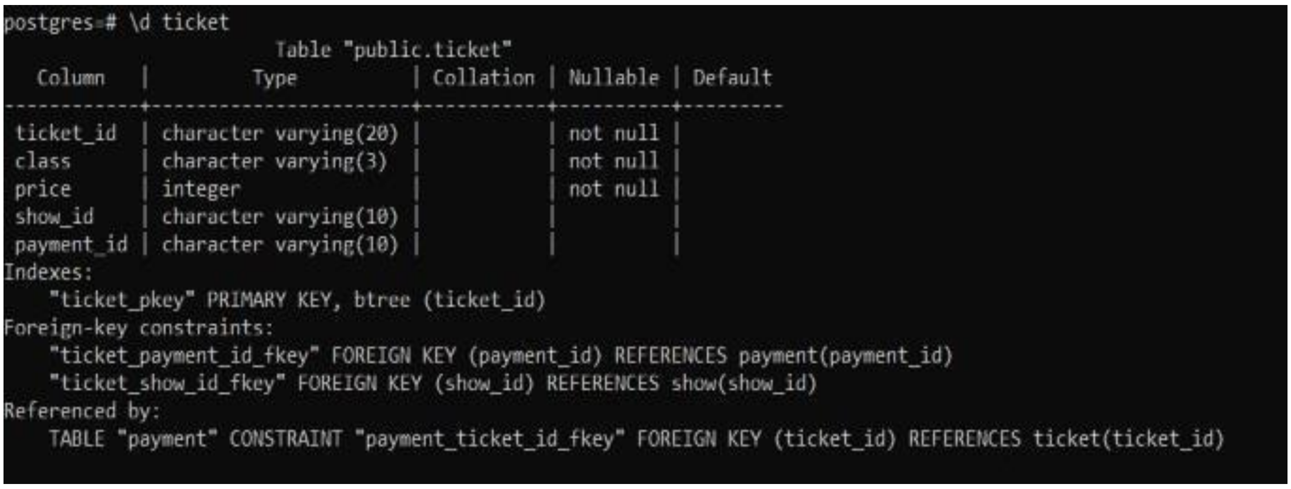
```
postgres-# \d oprator_room;
                     Table "public.oprator_room"
    Column     |         Type          | Collation | Nullable | Default
---------------+-----------------------+-----------+----------+---------
 hall_id       | character varying(20) |           | not null |
 incharge_name | character varying(20) |           |          |
 employee_id   | character varying(20) |           |          |
Indexes:
    "oprator_room_pkey" PRIMARY KEY, btree (hall_id)
Foreign-key constraints:
    "oprator_room_employee_id_fkey" FOREIGN KEY (employee_id) REFERENCES employee(employee_id)
```

4.9 payment



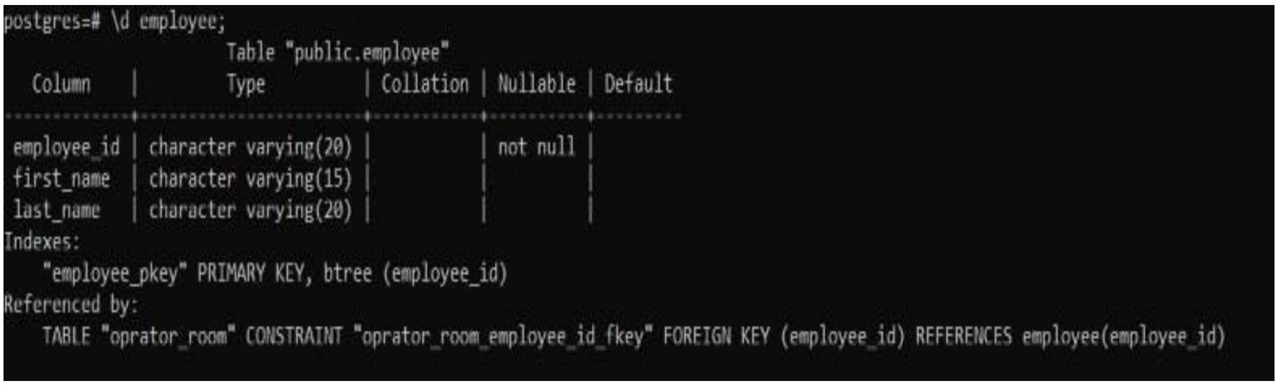
```
postgres=# \d payment;
                    Table "public.payment"
   Column   |         Type          | Collation | Nullable | Default
------------+-----------------------+-----------+----------+---------
 payment_id | character varying(20) |           | not null |
 amount     | integer               |           | not null |
 date       | date                  |           | not null |
 user_id    | character(20)         |           |          |
 ticket_id  | character(20)         |           |          |
Indexes:
    "payment_pkey" PRIMARY KEY, btree (payment_id)
Foreign-key constraints:
    "payment_ticket_id_fkey" FOREIGN KEY (ticket_id) REFERENCES ticket(ticket_id)
    "payment_user_id_fkey" FOREIGN KEY (user_id) REFERENCES web_user(web_user_id)
Referenced by:
    TABLE "seat" CONSTRAINT "seat_payment_id_fkey" FOREIGN KEY (payment_id) REFERENCES payment(payment_id)
    TABLE "ticket" CONSTRAINT "ticket_payment_id_fkey" FOREIGN KEY (payment_id) REFERENCES payment(payment_id)
```

4.10 ticket



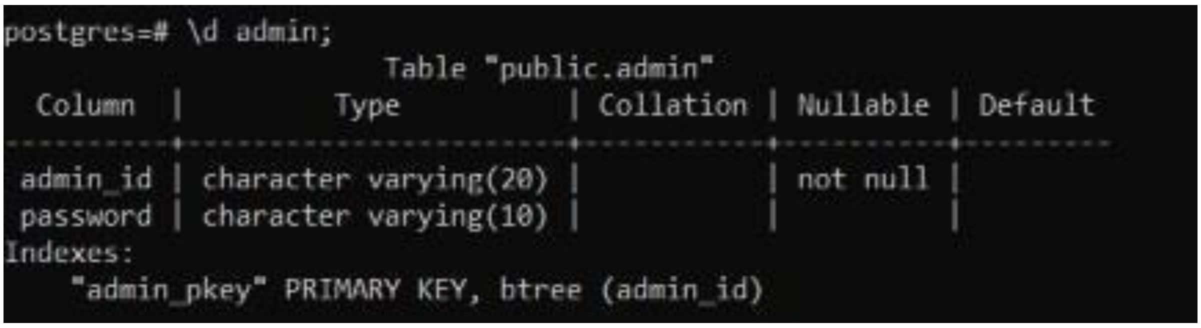
```
postgres=# \d ticket
                    Table "public.ticket"
   Column   |         Type          | Collation | Nullable | Default
------------+-----------------------+-----------+----------+---------
 ticket_id  | character varying(20) |           | not null |
 class      | character varying(3)  |           | not null |
 price      | integer               |           | not null |
 show_id    | character varying(10) |           |          |
 payment_id | character varying(10) |           |          |
Indexes:
    "ticket_pkey" PRIMARY KEY, btree (ticket_id)
Foreign-key constraints:
    "ticket_payment_id_fkey" FOREIGN KEY (payment_id) REFERENCES payment(payment_id)
    "ticket_show_id_fkey" FOREIGN KEY (show_id) REFERENCES show(show_id)
Referenced by:
    TABLE "payment" CONSTRAINT "payment_ticket_id_fkey" FOREIGN KEY (ticket_id) REFERENCES ticket(ticket_id)
```

4.11 employee



```
postgres=# \d employee;
                    Table "public.employee"
   Column    |         Type          | Collation | Nullable | Default
-------------+-----------------------+-----------+----------+---------
 employee_id | character varying(20) |           | not null |
 first_name  | character varying(15) |           |          |
 last_name   | character varying(20) |           |          |
Indexes:
    "employee_pkey" PRIMARY KEY, btree (employee_id)
Referenced by:
    TABLE "oprator_room" CONSTRAINT "oprator_room_employee_id_fkey" FOREIGN KEY (employee_id) REFERENCES employee(employee_id)
```

4.12 admin



```
postgres=# \d admin;
                    Table "public.admin"
  Column  |         Type          | Collation | Nullable | Default
----------+-----------------------+-----------+----------+---------
 admin_id | character varying(20) |           | not null |
 password | character varying(10) |           |          |
Indexes:
    "admin_pkey" PRIMARY KEY, btree (admin_id)
```

5. DATA IMPLEMENTATION

5.1 SCHEMA

5.1.1 web_user

```
CREATE Table Web_user(
Web_User_ID varchar(5),
First_Name varchar(15),
Last_Name varchar(20),
Email_ID varchar(30),
Phone_Number varchar(10) NOT NULL,
Primary Key(Web_User_ID));
```

5.1.2 parking

```
CREATE Table parking(
slot_no varchar(20),
vehical_type varchar(20),
user_ID  char(20),
primary Key(slot_no),
Foreign Key (User_ID) REFERENCES Web_User (Web_User_ID));
```

5.1.3 theatre

```
Create Table Theatre(
Theatre_ID varchar(5),
Name_of_Theatre varchar(30) NOT NULL,
No_of_Screens int,
Area varchar(30),
Primary Key(Theatre_ID));
```

5.1.4 <u>movie</u>

```
Create Table Movie(
Movie_ID varchar(5),
Name varchar(30) NOT NULL,
Language varchar(10),
Genre varchar(20),
Target_Audience varchar(5),
Primary Key(Movie_ID));
```

5.1.5 <u>show</u>

```
CREATE Table Show(
Show_ID varchar(10),
Show_Time TIME,
Show_Date date NOT NULL,
Class_Cost_Gold int NOT NULL,
Class_Cost_Silver int NOT NULL,
Screen_ID varchar(5) NOT NULL,
Movie_ID varchar(5) NOT NULL,
Primary Key(Show_ID),
Foreign Key (Screen_ID) REFERENCES Screen(Screen_ID),
Foreign Key (Movie_ID) REFERENCES Movie(Movie_ID));
```

5.1.6 <u>screen</u>

```
CREATE TABLE Screen(
Screen_ID varchar(5),
No_of_Seats_Gold int NOT NULL,
No_of_Seats_Silver int NOT NULL,
Theatre_ID varchar(5),
Primary Key(Screen_ID),
Foreign Key(Theatre_ID) REFERENCES Theatre(Theatre_ID));
```

5.1.7 <u>SEAT</u>

```
CREATE Table SEAT(
seat_id vachar(10),
seat_gold int NOT NULL CHECK(seat_gold <=50),
seat_silver int NOT NULL CHECK(seat_silver >50),
payment_id varchar(10),
primary key(seat_id),
Foreign key(payment_id) REFERENCES PAYMENT(payment_id));
```

5.1.8 <u>oprator_room</u>

```
CREATE Table oprator_room(
hall_ID varchar(20)
,incharge_name varchar(20),
employee_ID varchar(20),
primary Key(hall_ID),
Foreign Key (employee_ID) REFERENCES employee (employee_ID));
```

5.1.9 <u>payment</u>

```
CREATE Table Payment(
Payment_ID varchar(20),
Amount int NOT NULL,
Date DATE NOT NULL,
User_ID char(20),
Ticket_ID  char(20),
Primary Key(Payment_ID),
Foreign Key (User_ID) REFERENCES Web_User (Web_User_ID),
Foreign Key(Ticket_ID) REFERENCES Ticket(Ticket_ID));
```

5.1.10 <u>ticket</u>

```
CREATE Table Ticket(
Ticket_ID varchar(20),
Class varchar(3) NOT NULL,
Price int NOT NULL,
Primary Key(Ticket_ID));
```

5.1.11 <u>employee</u>

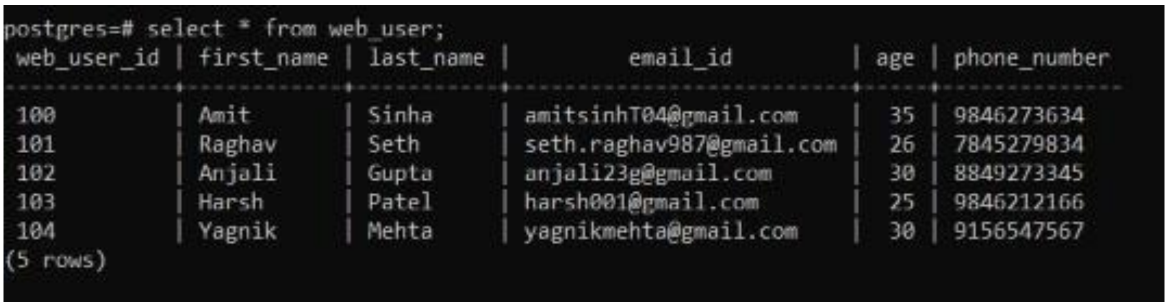
```
CREATE Table employee(
employee_ID varchar(20),
First_Name varchar(15),
Last_Name varchar(20),
primary Key(employee_ID));
```

5.1.12 <u>admin</u>

```
CREATE Table admin(
admin_ID varchar(20),
password varchar(10),
primary Key(admin_ID));
```

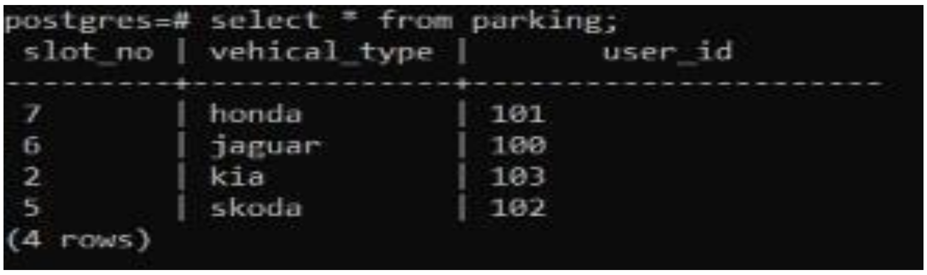
5.2 INSERTING DATA VALUES

5.2.1 web_user:



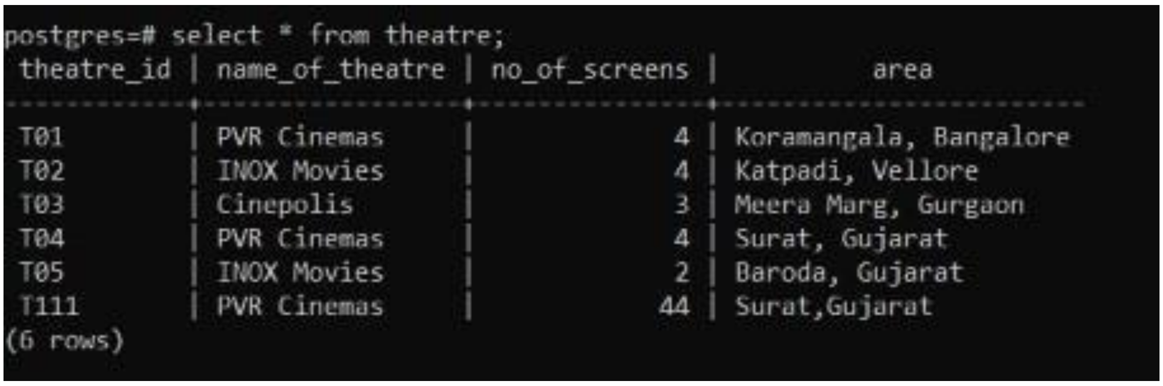
```
postgres=# select * from web_user;
 web_user_id | first_name | last_name |         email_id          | age | phone_number
-------------+------------+-----------+---------------------------+-----+--------------
 100         | Amit       | Sinha     | amitsinhT04@gmail.com     |  35 | 9846273634
 101         | Raghav     | Seth      | seth.raghav987@gmail.com  |  26 | 7845279834
 102         | Anjali     | Gupta     | anjali23g@gmail.com       |  30 | 8849273345
 103         | Harsh      | Patel     | harsh001@gmail.com        |  25 | 9846212166
 104         | Yagnik     | Mehta     | yagnikmehta@gmail.com     |  30 | 9156547567
(5 rows)
```

5.2.2 parking:



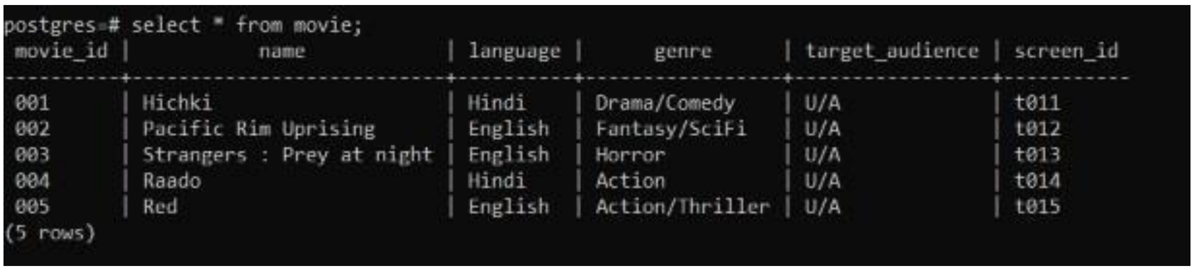
```
postgres=# select * from parking;
 slot_no | vehical_type |       user_id
---------+--------------+----------------------
 7       | honda        | 101
 6       | jaguar       | 100
 2       | kia          | 103
 5       | skoda        | 102
(4 rows)
```

5.2.3 theatre:



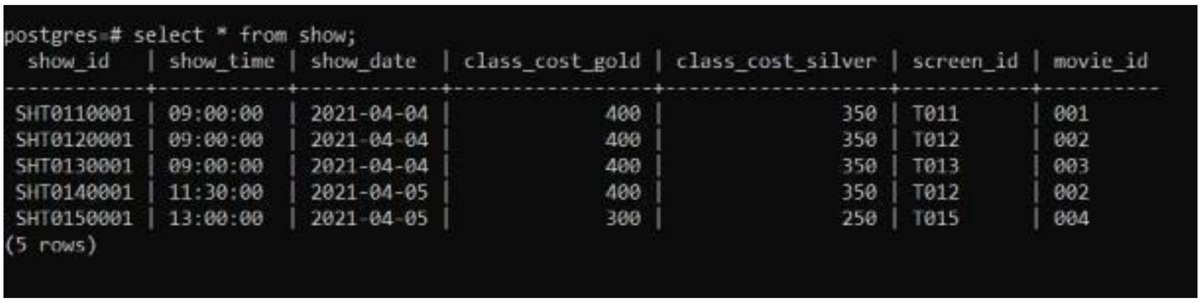
```
postgres=# select * from theatre;
 theatre_id | name_of_theatre | no_of_screens |          area
------------+-----------------+---------------+-------------------------
 T01        | PVR Cinemas     |             4 | Koramangala, Bangalore
 T02        | INOX Movies     |             4 | Katpadi, Vellore
 T03        | Cinepolis       |             3 | Meera Marg, Gurgaon
 T04        | PVR Cinemas     |             4 | Surat, Gujarat
 T05        | INOX Movies     |             2 | Baroda, Gujarat
 T111       | PVR Cinemas     |            44 | Surat,Gujarat
(6 rows)
```

5.2.4 <u>movie:</u>



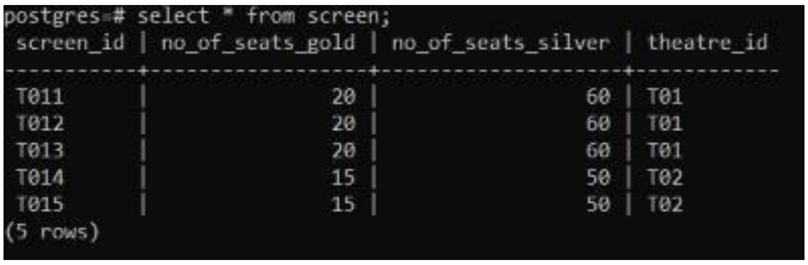
```
postgres=# select * from movie;
 movie_id |           name           | language |      genre      | target_audience | screen_id
----------+--------------------------+----------+-----------------+-----------------+-----------
 001      | Hichki                   | Hindi    | Drama/Comedy    | U/A             | t011
 002      | Pacific Rim Uprising     | English  | Fantasy/SciFi   | U/A             | t012
 003      | Strangers : Prey at night | English  | Horror          | U/A             | t013
 004      | Raado                    | Hindi    | Action          | U/A             | t014
 005      | Red                      | English  | Action/Thriller | U/A             | t015
(5 rows)
```

5.2.5 <u>show:</u>



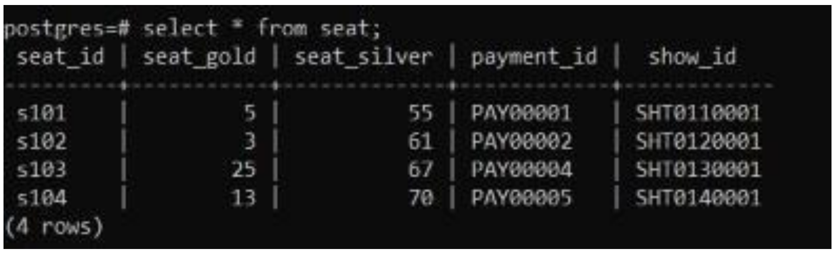
```
postgres=# select * from show;
  show_id    | show_time | show_date  | class_cost_gold | class_cost_silver | screen_id | movie_id
-------------+-----------+------------+-----------------+-------------------+-----------+----------
 SHT0110001 | 09:00:00  | 2021-04-04 |             400 |               350 | T011      | 001
 SHT0120001 | 09:00:00  | 2021-04-04 |             400 |               350 | T012      | 002
 SHT0130001 | 09:00:00  | 2021-04-04 |             400 |               350 | T013      | 003
 SHT0140001 | 11:30:00  | 2021-04-05 |             400 |               350 | T012      | 002
 SHT0150001 | 13:00:00  | 2021-04-05 |             300 |               250 | T015      | 004
(5 rows)
```

5.2.6 <u>screen:</u>



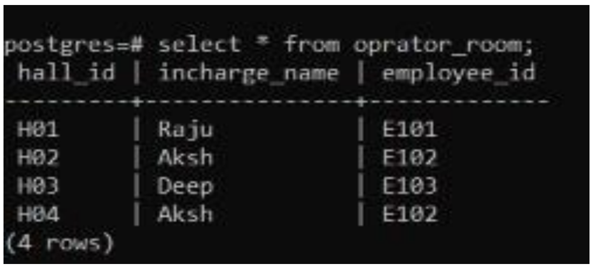
```
postgres=# select * from screen;
 screen_id | no_of_seats_gold | no_of_seats_silver | theatre_id
-----------+------------------+--------------------+------------
 T011      |               20 |                 60 | T01
 T012      |               20 |                 60 | T01
 T013      |               20 |                 60 | T01
 T014      |               15 |                 50 | T02
 T015      |               15 |                 50 | T02
(5 rows)
```

5.2.7 <u>seat:</u>



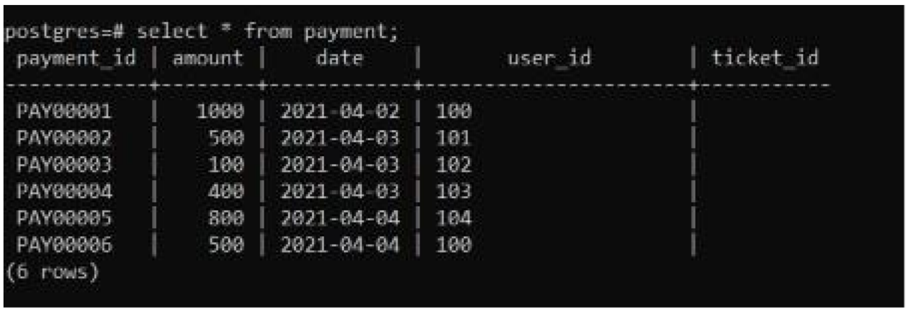
```
postgres=# select * from seat;
 seat_id | seat_gold | seat_silver | payment_id |  show_id
---------+-----------+-------------+------------+------------
 s101    |         5 |          55 | PAY00001   | SHT0110001
 s102    |         3 |          61 | PAY00002   | SHT0120001
 s103    |        25 |          67 | PAY00004   | SHT0130001
 s104    |        13 |          70 | PAY00005   | SHT0140001
(4 rows)
```

5.2.8 opprator_room:



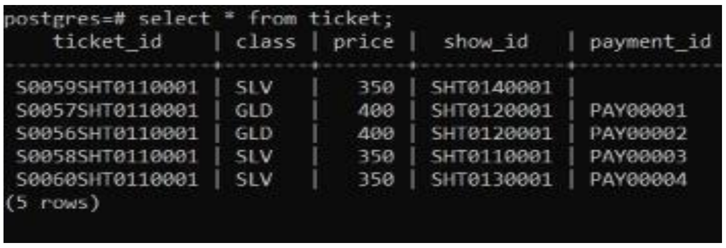
```
postgres=# select * from oprator_room;
 hall_id | incharge_name | employee_id
---------+---------------+-------------
 H01     | Raju          | E101
 H02     | Aksh          | E102
 H03     | Deep          | E103
 H04     | Aksh          | E102
(4 rows)
```

5.2.9 payment:



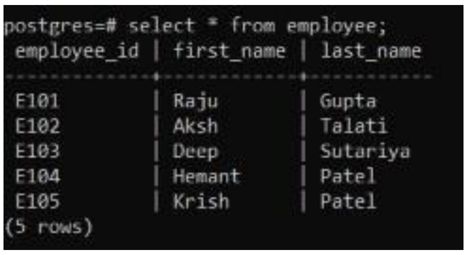
```
postgres=# select * from payment;
 payment_id | amount |    date    |        user_id        | ticket_id
------------+--------+------------+-----------------------+-----------
 PAY00001   |   1000 | 2021-04-02 | 100                   |
 PAY00002   |    500 | 2021-04-03 | 101                   |
 PAY00003   |    100 | 2021-04-03 | 102                   |
 PAY00004   |    400 | 2021-04-03 | 103                   |
 PAY00005   |    800 | 2021-04-04 | 104                   |
 PAY00006   |    500 | 2021-04-04 | 100                   |
(6 rows)
```

5.2.10 ticket:



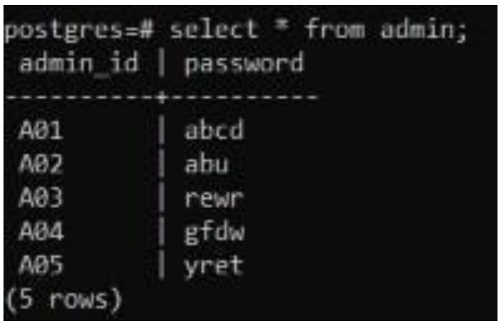
```
postgres=# select * from ticket;
    ticket_id     | class | price |   show_id   | payment_id
------------------+-------+-------+-------------+------------
 S0059SHT0110001 | SLV   |   350 | SHT0140001 |
 S0057SHT0110001 | GLD   |   400 | SHT0120001 | PAY00001
 S0056SHT0110001 | GLD   |   400 | SHT0120001 | PAY00002
 S0058SHT0110001 | SLV   |   350 | SHT0110001 | PAY00003
 S0060SHT0110001 | SLV   |   350 | SHT0130001 | PAY00004
(5 rows)
```

5.2.11 employee:



```
postgres=# select * from employee;
 employee_id | first_name | last_name
-------------+------------+-----------
 E101        | Raju       | Gupta
 E102        | Aksh       | Talati
 E103        | Deep       | Sutariya
 E104        | Hemant     | Patel
 E105        | Krish      | Patel
(5 rows)
```

5.2.12 admin:

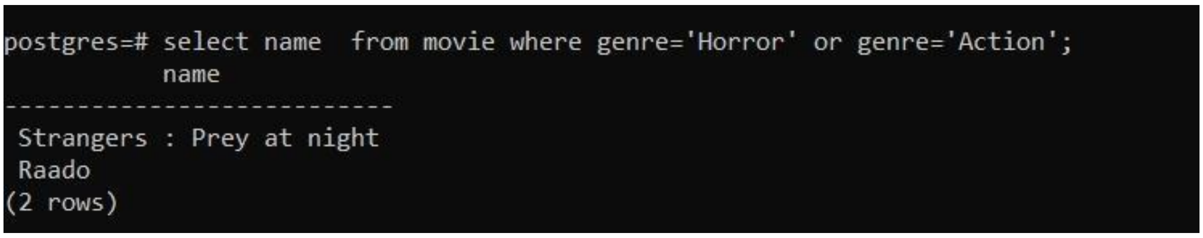


```
postgres=# select * from admin;
 admin_id | password
----------+----------
 A01      | abcd
 A02      | abu
 A03      | rewr
 A04      | gfdw
 A05      | yret
(5 rows)
```

5.3 QUERIES USING BASIC DBMS CONSTRUCTS.JOIN AND SUBQUERIES:

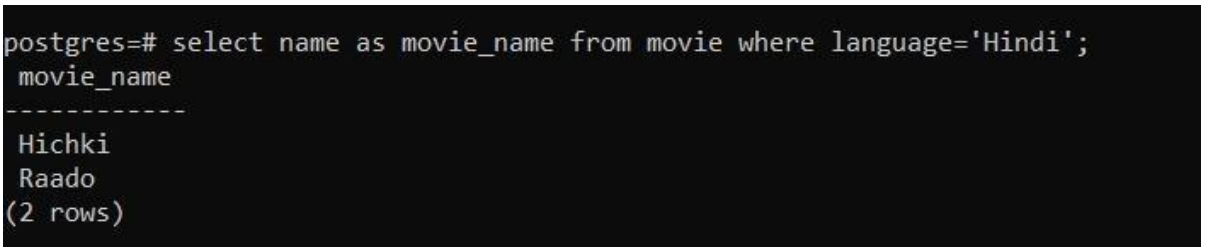
Queries :

5.3.1 Display name of movie where genre is horror or action.



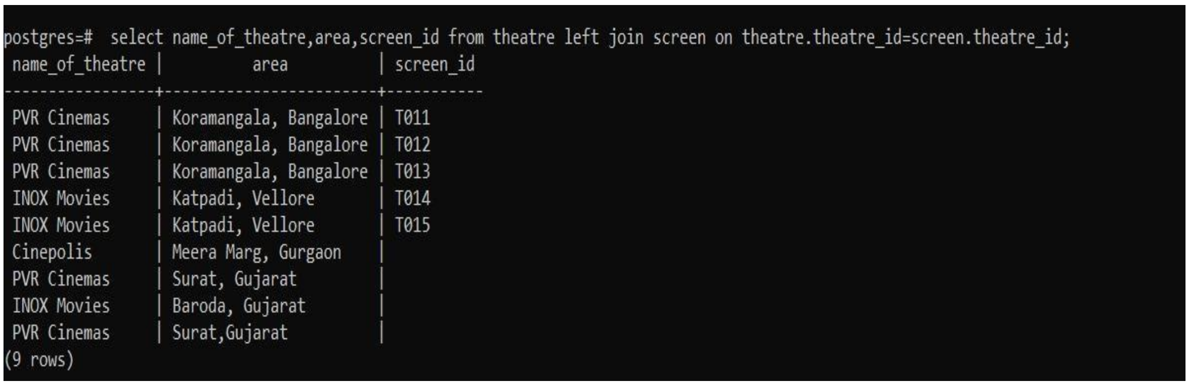
```
postgres=# select name  from movie where genre='Horror' or genre='Action';
          name
---------------------------
 Strangers : Prey at night
 Raado
(2 rows)
```

5.3.2 Display name of movie where language is hindi.



```
postgres=# select name as movie_name from movie where language='Hindi';
 movie_name
------------
 Hichki
 Raado
(2 rows)
```

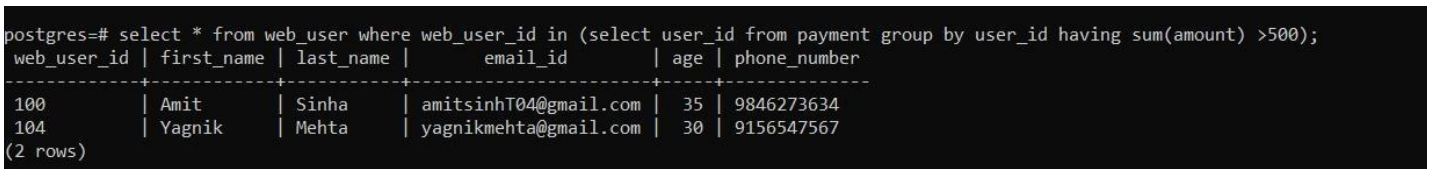
5.3.3 Display name of theatre, area and screen id where theatre id is same.



```
postgres=#  select name_of_theatre,area,screen_id from theatre left join screen on theatre.theatre_id=screen.theatre_id;
 name_of_theatre |         area          | screen_id
-----------------+-----------------------+-----------
 PVR Cinemas     | Koramangala, Bangalore | T011
 PVR Cinemas     | Koramangala, Bangalore | T012
 PVR Cinemas     | Koramangala, Bangalore | T013
 INOX Movies     | Katpadi, Vellore       | T014
 INOX Movies     | Katpadi, Vellore       | T015
 Cinepolis       | Meera Marg, Gurgaon    |
 PVR Cinemas     | Surat, Gujarat         |
 INOX Movies     | Baroda, Gujarat        |
 PVR Cinemas     | Surat,Gujarat          |
(9 rows)
```

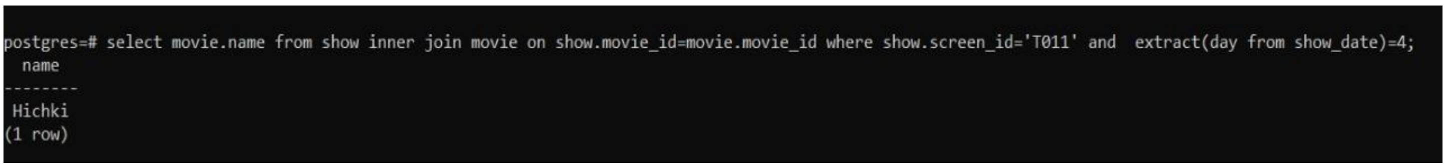
JOIN Queries:

5.3.4 Display all details of user where user paid more than 500 amount.



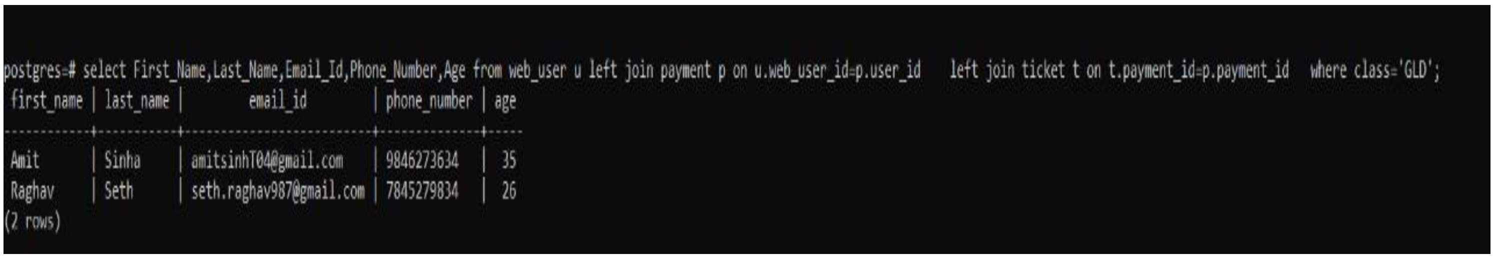
```
postgres=# select * from web_user where web_user_id in (select user_id from payment group by user_id having sum(amount) >500);
 web_user_id | first_name | last_name |       email_id        | age | phone_number
-------------+------------+-----------+-----------------------+-----+--------------
 100         | Amit       | Sinha     | amitsinhT04@gmail.com |  35 | 9846273634
 104         | Yagnik     | Mehta     | yagnikmehta@gmail.com |  30 | 9156547567
(2 rows)
```

5.3.5 Display name of movie which shown on screen id 1 on date 4.



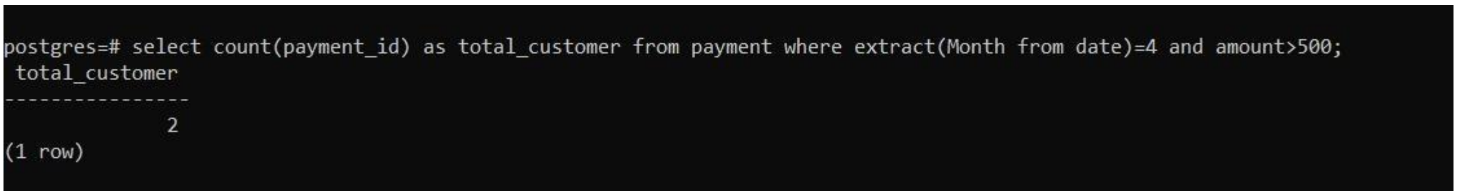
```
postgres=# select movie.name from show inner join movie on show.movie_id=movie.movie_id where show.screen_id='T011' and  extract(day from show_date)=4;
  name
--------
 Hichki
(1 row)
```

5.3.6 Display details of user where class is gold .



```
postgres=# select First_Name,Last_Name,Email_Id,Phone_Number,Age from web_user u left join payment p on u.web_user_id=p.user_id    left join ticket t on t.payment_id=p.payment_id   where class='GLD';
 first_name | last_name |        email_id        | phone_number | age
------------+-----------+------------------------+--------------+-----
 Amit       | Sinha     | amitsinhT04@gmail.com  | 9846273634   |  35
 Raghav     | Seth      | seth.raghav987@gmail.com | 7845279834 |  26
(2 rows)
```

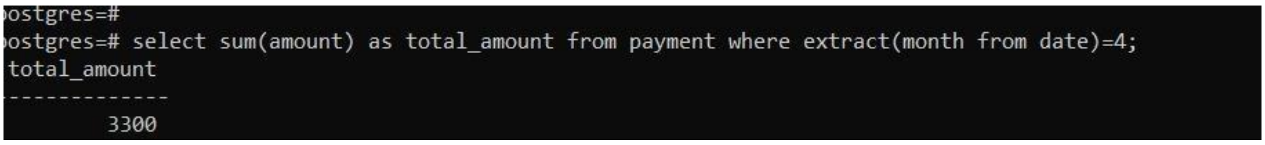
5.3.7 Display count of customer who paid more than 500 payment on date 4.



```
postgres=# select count(payment_id) as total_customer from payment where extract(Month from date)=4 and amount>500;
 total_customer
----------------
              2
(1 row)
```

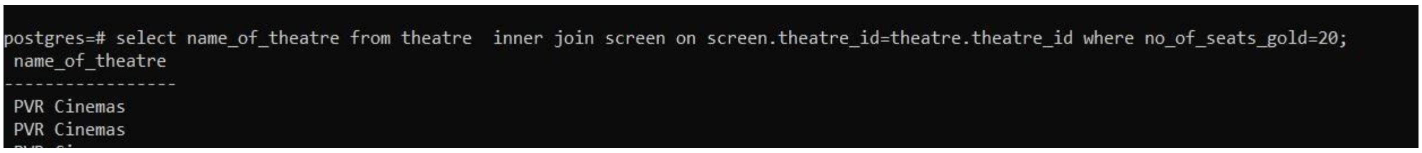
Basic queries:

5.3.8 Display total amount collected on month april.



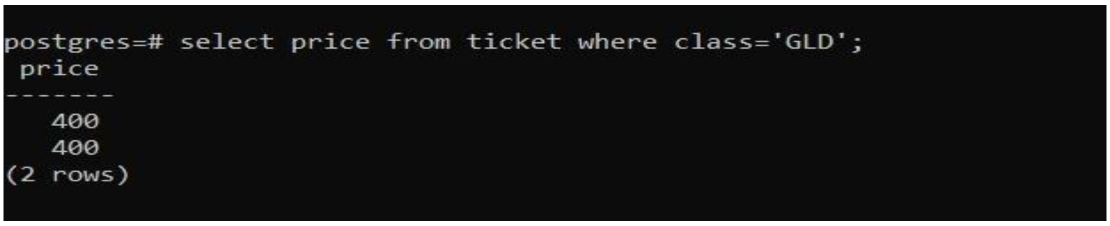
```
ostgres=#
ostgres=# select sum(amount) as total_amount from payment where extract(month from date)=4;
 total_amount
--------------
         3300
```

5.3.9 Display theatre name of screen which has no of seats gold is 20.



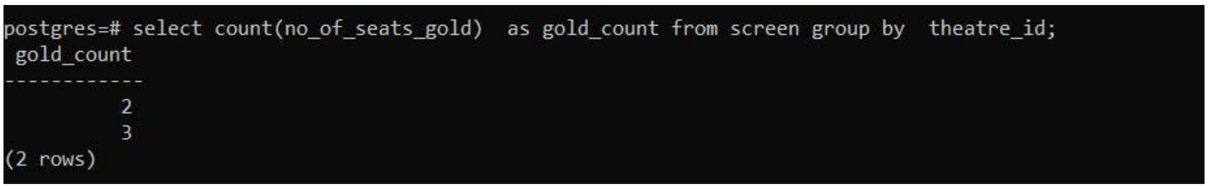
```
postgres=# select name_of_theatre from theatre  inner join screen on screen.theatre_id=theatre.theatre_id where no_of_seats_gold=20;
 name_of_theatre
-----------------
 PVR Cinemas
 PVR Cinemas
```

5.3.10 Display ticket price where class is gold.



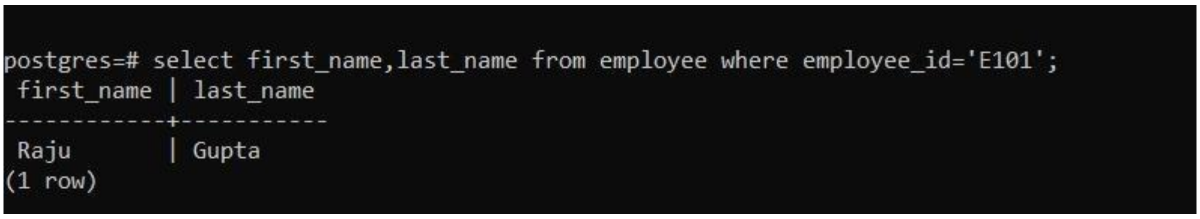
```
postgres=# select price from ticket where class='GLD';
 price
-------
   400
   400
(2 rows)
```

5.3.11 Display name of movie according to its id in descending orders.



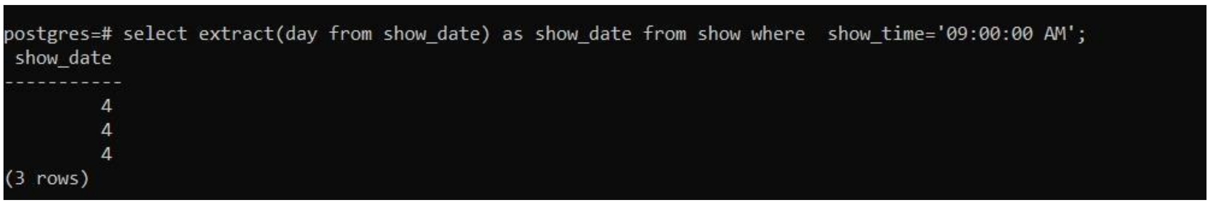
```
postgres=# select count(no_of_seats_gold)  as gold_count from screen group by  theatre_id;
 gold_count
------------
          2
          3
(2 rows)
```

5.3.12 Display count of gold_count of screens according grouping of theatre id.



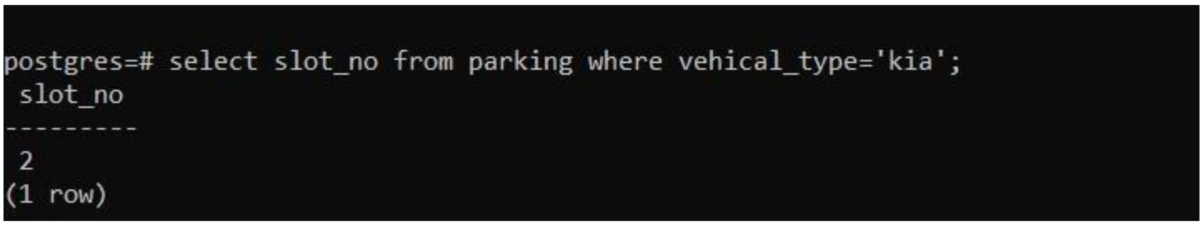
```
postgres=# select first_name,last_name from employee where employee_id='E101';
 first_name | last_name
------------+-----------
 Raju       | Gupta
(1 row)
```

5.3.13 Display first name and last name of employee whose id is E101.



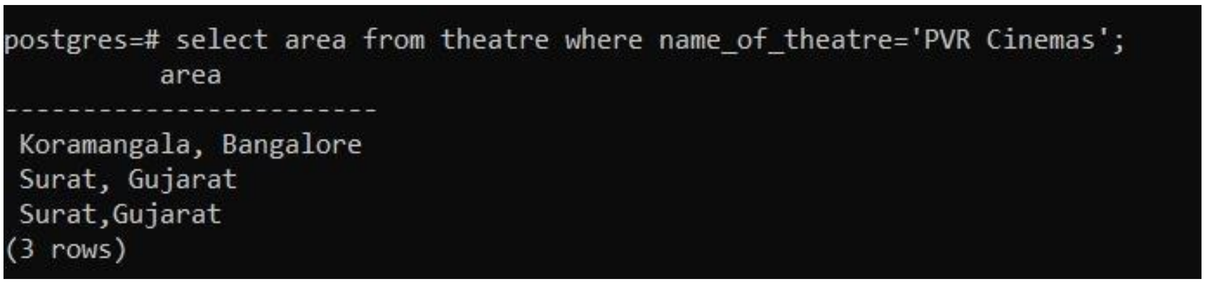
```
postgres=# select extract(day from show_date) as show_date from show where  show_time='09:00:00 AM';
 show_date
-----------
         4
         4
         4
(3 rows)
```

5.3.14 Display finding day of show where show time is 9 am.



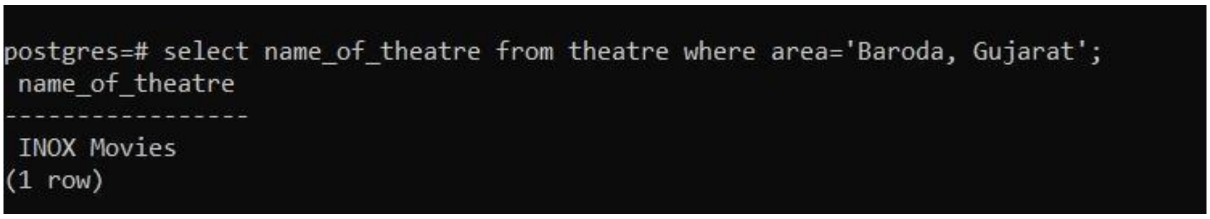
```
postgres=# select slot_no from parking where vehical_type='kia';
 slot_no
---------
 2
(1 row)
```

5.3.15 Display in which slot kia car is parked.



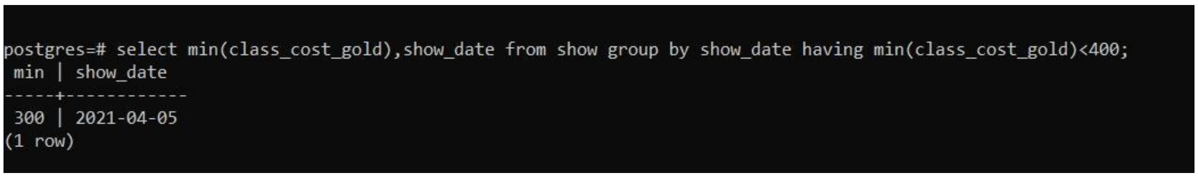
```
postgres=# select area from theatre where name_of_theatre='PVR Cinemas';
          area
------------------------
 Koramangala, Bangalore
 Surat, Gujarat
 Surat,Gujarat
(3 rows)
```

5.3.16 Display areas where pvr cinema is located



```
postgres=# select name_of_theatre from theatre where area='Baroda, Gujarat';
 name_of_theatre
-----------------
 INOX Movies
(1 row)
```

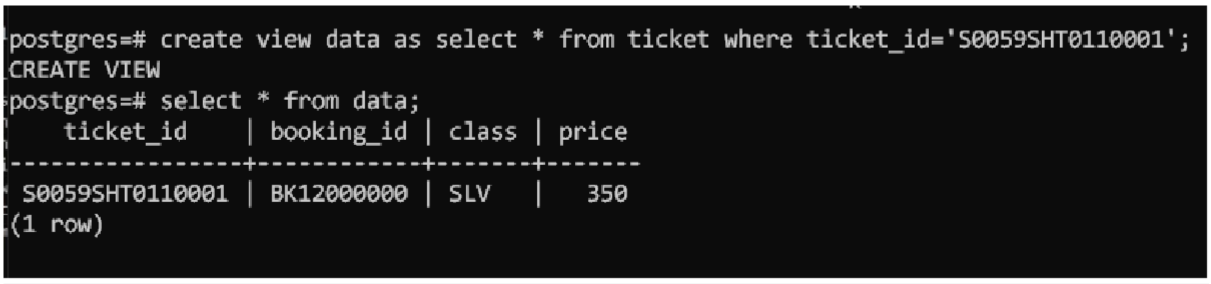
5.3.17 Display name of theatres located in baroda.



```
postgres=# select min(class_cost_gold),show_date from show group by show_date having min(class_cost_gold)<400;
 min | show_date
-----+------------
 300 | 2021-04-05
(1 row)
```

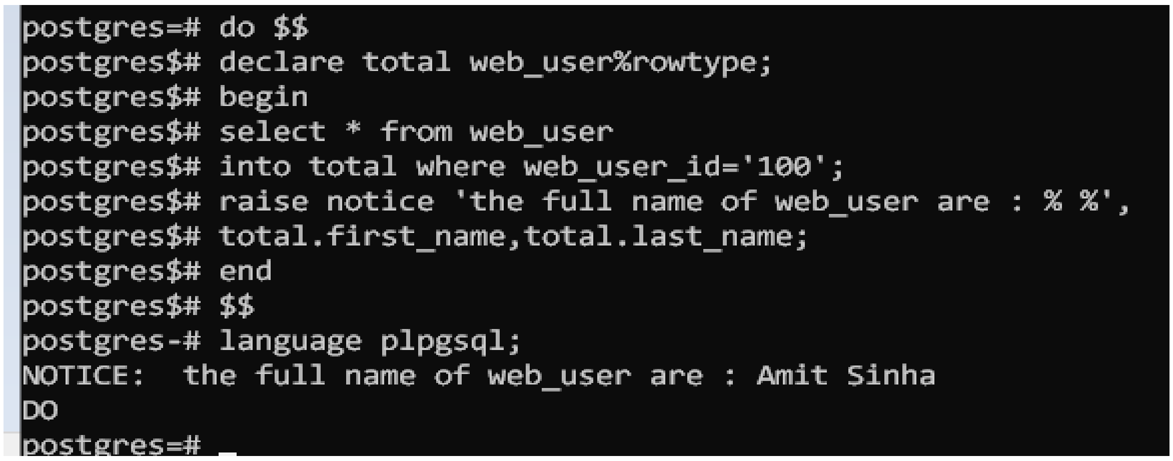
5.4 PL/SQL

VIEW:



```
postgres=# create view data as select * from ticket where ticket_id='S0059SHT0110001';
CREATE VIEW
postgres=# select * from data;
    ticket_id    | booking_id | class | price
-----------------+------------+-------+-------
 S0059SHT0110001 | BK12000000 | SLV   |   350
(1 row)
```

ROWTYPE:



```
postgres=# do $$
postgres$# declare total web_user%rowtype;
postgres$# begin
postgres$# select * from web_user
postgres$# into total where web_user_id='100';
postgres$# raise notice 'the full name of web_user are : % %',
postgres$# total.first_name,total.last_name;
postgres$# end
postgres$# $$
postgres-# language plpgsql;
NOTICE:  the full name of web_user are : Amit Sinha
DO
postgres=# _
```

5.5 FUNCTION & TRIGGERS:

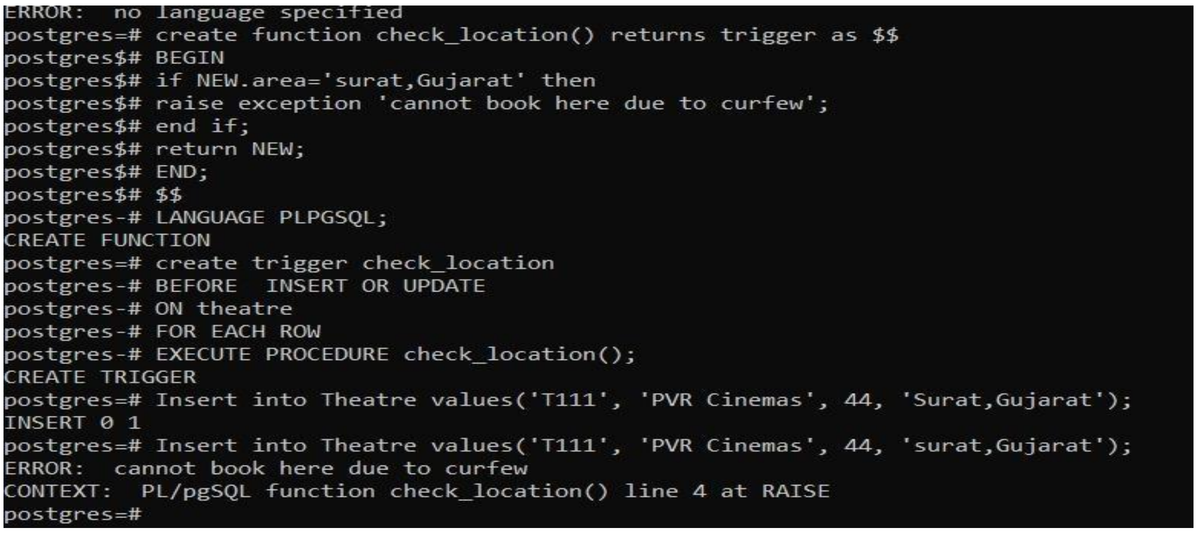
5.5.1 get error when user try to book theatre in Surat, Gujarat area

Function :

```
create function check_location() returns trigger as $$
BEGIN
 if NEW.area='surat,Gujarat' then
 raise exception 'cannot book here due to curfew';
end if;
return NEW;
END;
 $$ LANGUAGE PLPGSQL;
```

Trigger:

```
create trigger check_location
 BEFORE  INSERT OR UPDATE
 ON  theatre
 FOR EACH ROW
 EXECUTE PROCEDURE check_location();
```



```
ERROR:  no language specified
postgres=# create function check_location() returns trigger as $$
postgres$# BEGIN
postgres$# if NEW.area='surat,Gujarat' then
postgres$# raise exception 'cannot book here due to curfew';
postgres$# end if;
postgres$# return NEW;
postgres$# END;
postgres$# $$
postgres-# LANGUAGE PLPGSQL;
CREATE FUNCTION
postgres=# create trigger check_location
postgres-# BEFORE  INSERT OR UPDATE
postgres-# ON theatre
postgres-# FOR EACH ROW
postgres-# EXECUTE PROCEDURE check_location();
CREATE TRIGGER
postgres=# Insert into Theatre values('T111', 'PVR Cinemas', 44, 'Surat,Gujarat');
INSERT 0 1
postgres=# Insert into Theatre values('T111', 'PVR Cinemas', 44, 'surat,Gujarat');
ERROR:  cannot book here due to curfew
CONTEXT:  PL/pgSQL function check_location() line 4 at RAISE
postgres=#
```

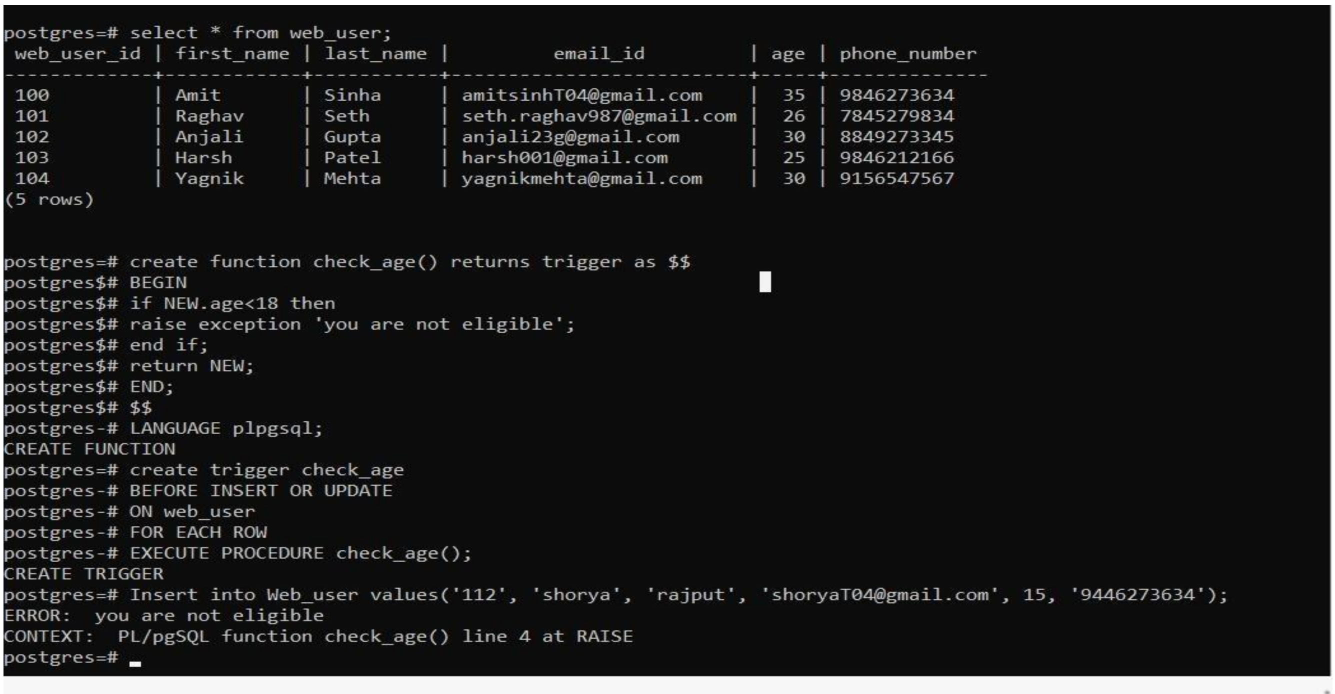
5.5.2 get error when under 18 aged person try to book ticket

Function:

```
create function check_age() returns trigger as $$
 BEGIN
 if NEW.age<18 then
 raise exception 'you are not eligible';
 end if;
return NEW;
 END;
 $$  LANGUAGE plpgsql;
```

Trigger

```
create trigger check_age
 BEFORE INSERT OR UPDATE
 ON web_user
 FOR EACH ROW
 EXECUTE PROCEDURE check_age();
```

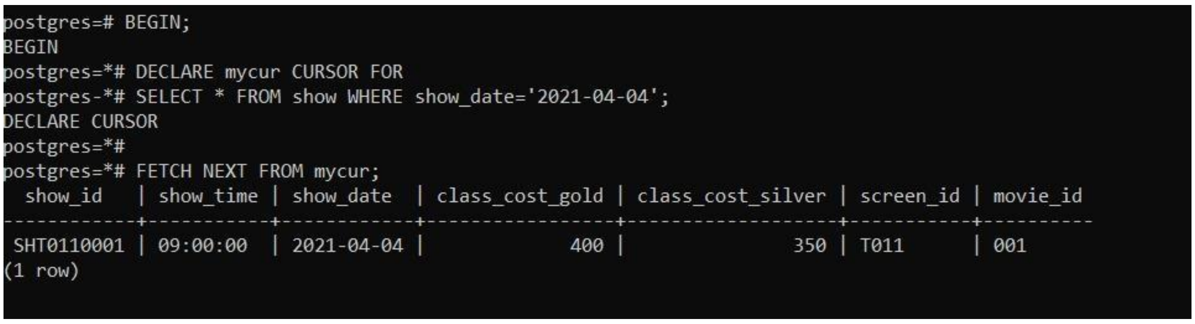


```
postgres=# select * from web_user;
 web_user_id | first_name | last_name |         email_id          | age | phone_number
-------------+------------+-----------+---------------------------+-----+--------------
 100         | Amit       | Sinha     | amitsinhT04@gmail.com     |  35 | 9846273634
 101         | Raghav     | Seth      | seth.raghav987@gmail.com  |  26 | 7845279834
 102         | Anjali     | Gupta     | anjali23g@gmail.com       |  30 | 8849273345
 103         | Harsh      | Patel     | harsh001@gmail.com        |  25 | 9846212166
 104         | Yagnik     | Mehta     | yagnikmehta@gmail.com     |  30 | 9156547567
(5 rows)


postgres=# create function check_age() returns trigger as $$
postgres$# BEGIN
postgres$# if NEW.age<18 then
postgres$# raise exception 'you are not eligible';
postgres$# end if;
postgres$# return NEW;
postgres$# END;
postgres$# $$
postgres-# LANGUAGE plpgsql;
CREATE FUNCTION
postgres=# create trigger check_age
postgres-# BEFORE INSERT OR UPDATE
postgres-# ON web_user
postgres-# FOR EACH ROW
postgres-# EXECUTE PROCEDURE check_age();
CREATE TRIGGER
postgres=# Insert into Web_user values('112', 'shorya', 'rajput', 'shoryaT04@gmail.com', 15, '9446273634');
ERROR:  you are not eligible
CONTEXT:  PL/pgSQL function check_age() line 4 at RAISE
postgres=#
```

5.6 CURSOR:

Cursor:

```
BEGIN;

DECLARE mycur CURSOR FOR

SELECT * FROM show WHERE show_date='2021-04-04';

FETCH NEXT FROM mycur;

FETCH PRIOR FROM mycur;

CLOSE mycur;

end;
```



```
postgres=# BEGIN;
BEGIN
postgres=*# DECLARE mycur CURSOR FOR
postgres-*# SELECT * FROM show WHERE show_date='2021-04-04';
DECLARE CURSOR
postgres=*#
postgres=*# FETCH NEXT FROM mycur;
  show_id   | show_time | show_date  | class_cost_gold | class_cost_silver | screen_id | movie_id
------------+-----------+------------+-----------------+-------------------+-----------+----------
 SHT0110001 | 09:00:00  | 2021-04-04 |             400 |               350 | T011      | 001
(1 row)
```

6 FUTURE ENHANCEMENT:

- We will apply a Front end and Back end and make this system to real working base system.

- We will implement proper user interface and gui so system become more reliable.

- We will make our database on cloud so work become more easy and fast.