DBMS ASSIGNMENT – 3

UE19CS301

PROJECT TITLE: ONLINE MOVIE TICKET BOOKING MANAGEMENT SYSTEM

SEMESTER: 5

SECTION: B

SRN	Team Members
PES1UG19CS080	Apoorva BS
PES1UG19CS068	Ankita V
PES1UG19CS079	Anvika D Shriyan

CONNECTING TO POSTGRE SQL:

```
dell@dell-Inspiron-7559:~$ cd Desktop
dell@dell-Inspiron-7559:~/Desktop$ cd DBMS_Assignment
dell@dell-Inspiron-7559:~/Desktop/DBMS_Assignment$ sudo -i -u postgres
[sudo] password for dell:
postgres@dell-Inspiron-7559:~$ psql -U postgres -f /home/dell/Desktop/DBMS_Assignment/cs068_079_080.sql
DROP DATABASE
CREATE DATABASE
You are now connected to database "cs068_079_080" as user "postgres".
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
ALTER TABLE
ALTER TABLE
postgres@dell-Inspiron-7559:~$ psql -U postgres -f /home/dell/Desktop/DBMS_Assignment/cs068_079_080_insert.sql
You are now connected to database "cs068_079_080" as user "postgres".
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
```

```
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
postgres@dell-Inspiron-7559:~$
```

EXECUTION OF THE QUERIES: (WITH SCREENSHOTS)

A) SIMPLE QUERIES

1. Select the movies that are Animated.

```
postgres=# \c cs068_079_080
You are now connected to database "cs068_079_080" as user "postgres".
cs068_079_080=#
cs068_079_080=#
cs068_079_080=# select movie_name from movies where genre = 'ANIMATED';
 movie_name
------------
 UP
 LIONKING
(2 rows)
```

2. How many screens does PVR have?

```
cs068_079_080=# select theatre_name, no_of_screens
from theatre
where theatre_name = 'PVR';
 theatre_name | no_of_screens
--------------+---------------
 PVR          |             6
(1 row)
```

3. Retrieve the tickets with a price greater than 200

```
cs068_079_080=# select ticket_id, price from ticket where price > 200;
 ticket_id | price
-----------+-------
 BSK60116  |   400
 JPN59980  |   300
 JPN56604  |   400
 MGR99769  |   300
 JPN577304 |   400
(5 rows)
```

4. Retrieve the user IDs of users older than 30 years

```
cs068_079_080=# select user_id from users where age > 30;
 user_id
----------
 1FGH6539
 8GRF4377
 3RFP5564
 7DXK8945
(4 rows)
```

5. List the Show IDs and Screen IDs of all the running shows.

```
cs068_079_080=# select show_id, screen_id from show;
 show_id | screen_id
---------+-----------
 AKL123I | 1D
 GHJ5OOC | 5C
 IKLD23Q | 1A
 HJK2217 | 4C
 CVNB09Z | 3B
 SD67FRT | 7S
 UI77PLM | 6A
 WTTYH89 | 3A
(8 rows)
```

B) COMPLEX QUERIES

1. Get all show_id and theatre names in all regions

```
cs068_079_080=# select show_id, theatre_name, city
cs068_079_080-# from show, theatre, region
cs068_079_080-# where show.movie_id = theatre.movie_id and theatre.theatre_id = region.theatre_id;
 show_id |   theatre_name   |    city
---------+------------------+------------
 HJK2217 | INOX MOVIES      | TOKYO
 SD67FRT | PVR              | BANGALORE
 HJK2217 | INOX MOVIES      | MYSORE
 CVNB09Z | HALLMARK         | DELHI
 SD67FRT | REX THEATRE      | MUMBAI
 UI77PLM | CINEPOLIS        | RAYLEIGH
 AKL123I | SRINIVAS THEATRE | DUSSELDORF
 UI77PLM | CINEPOLIS        | MADRID
(8 rows)
```

2. List the name of the movies, their genre, and languages that were released after the 16th of September 2021.

```
cs068_079_080=# select M.movie_name, M.genre, L.language_name
cs068_079_080-# from movies AS M, languages AS L
cs068_079_080-# where L.movie_id = M.movie_id and M.release_date > '2021-09-16'
cs068_079_080-# ORDER BY genre;
 movie_name |  genre   | language_name
------------+----------+---------------
 UP         | ANIMATED | TELUGU
 CHOPSTICKS | COMEDY   | MALAYALAM
 RUSHHOUR   | COMEDY   | JAPANESE
 PARASITE   | THRILLER | TAMIL
(4 rows)
```

3. Retrieve the Cost of the shows running on 12th September 2021.

```
cs068_079_080=# select S.show_id, S.show_date, B.cost
cs068_079_080-# from show AS S, booking AS B
cs068_079_080-# where S.show_id = B.show_id and show_date = '2021-09-12';
 show_id | show_date  | cost
---------+------------+------
 GHJ5OOC | 2021-09-12 |  900
(1 row)
```

4. Get all show_id for movies with genre 'animated'.

```
cs068_079_080=# select show_id, movie_name
cs068_079_080-# from show, movies
cs068_079_080-# where show.movie_id = movies.movie_id and genre = 'ANIMATED';
 show_id | movie_name
---------+------------
 GHJ5OOC | UP
 IKLD23Q | LIONKING
(2 rows)
```

5. List the booking ID, the number of tickets, and mode of payment of users who used Debit Card for the payment.

```
cs068_079_080=# select B.booking_id, B.no_of_tickets, M.payment
cs068_079_080-# from booking AS B, make_booking AS M
cs068_079_080-# where B.booking_id = M.bid and payment = 'DEBITCARD'
cs068_079_080-# ORDER BY no_of_tickets;
 booking_id | no_of_tickets |  payment
------------+---------------+-----------
 BTM36080   |             1 | DEBITCARD
 BSK60106   |             2 | DEBITCARD
(2 rows)
```

C) NESTED QUERIES

1. What movies are being shown after 5 pm?

```
cs068_079_080=# select M.movie_name, M.release_date, S.time
cs068_079_080-# from movies AS M, show AS S
cs068_079_080-# where M.movie_id IN (select S.movie_id from show where S.time > 1700);
 movie_name | release_date | time
------------+--------------+------
 LIONKING   | 2021-09-12   | 1900
 RUSHHOUR   | 2021-09-18   | 2100
(2 rows)
```

2. Currently how many movies are being shown across all theatres?

```
cs068_079_080=# select COUNT(*)
cs068_079_080-# from (select DISTINCT movie_name from movies)
cs068_079_080-# AS count_distinct_movies;
 count
-------
     8
(1 row)
```

3. List the users who have booked more than 3 tickets.

```
cs068_079_080=# select name
cs068_079_080-# from users
cs068_079_080-# where user_id IN (select user_id from booking where no_of_tickets > 3);
  name
---------
 ANIRUDH
 MAYA
(2 rows)
```

4. List the theatres in a region with Pin code as 448822.

```
cs068_079_080=# select theatre_name
cs068_079_080-# from theatre
cs068_079_080-# where theatre_id IN (select theatre_id from region where pincode = '448822');
   theatre_name
-----------------
 SRINIVAS THEATRE
(1 row)
```

5. Which city has the maximum number of theatres? Return the city name along with its Pin code.

```
cs068_079_080=# select city, pincode
cs068_079_080-# from region
cs068_079_080-# where no_of_theatres = (select MAX(no_of_theatres) from region);
  city   | pincode
---------+---------
 MUMBAI |  400002
(1 row)
```

D) TRANSACTIONS

Initiating concurrent Transactions and demonstrating the concurrency control for the conflicting actions.

```
cs068_079_080=# begin;
BEGIN
cs068_079_080=# create table food_counter_1 (Ticket_ID varchar(15), food_name varchar(10), food_price int);
CREATE TABLE
cs068_079_080=# insert into food_counter_1 values('1QWE34', 'Popcorn', 100);
INSERT 0 1
cs068_079_080=# insert into food_counter_1 values('2ASKL9', 'Juice', 50);
INSERT 0 1
cs068_079_080=# insert into food_counter_1 values('0YTHGR', 'Chips', 150);
INSERT 0 1
cs068_079_080=# insert into food_counter_1 values('JKLMN6', 'Chips', 150);
INSERT 0 1
cs068_079_080=# insert into food_counter_1 values('997CFG', 'Popcorn', 100);
INSERT 0 1
cs068_079_080=# insert into food_counter_1 values('UI7655', 'Juice', 60);
INSERT 0 1
cs068_079_080=# insert into food_counter_1 values('ZXTI8P', 'Juice', 60);
INSERT 0 1
cs068_079_080=# insert into food_counter_1 values('BNMRT7', 'Popcorn', 100);
INSERT 0 1
cs068_079_080=# commit;
COMMIT
cs068_079_080=# select * from food_counter_1;
 ticket_id | food_name | food_price
-----------+-----------+------------
 1QWE34    | Popcorn   |        100
 2ASKL9    | Juice     |         50
 0YTHGR    | Chips     |        150
 JKLMN6    | Chips     |        150
 997CFG    | Popcorn   |        100
 UI7655    | Juice     |         60
 ZXTI8P    | Juice     |         60
 BNMRT7    | Popcorn   |        100
(8 rows)
```

```
cs068_079_080=# begin;
BEGIN
cs068_079_080=# create table food_counter_2 (Ticket_ID varchar(15), food_name varchar(10), food_price int);
CREATE TABLE
cs068_079_080=# insert into food_counter_2 values('1QWE34', 'Popcorn', 100);
INSERT 0 1
cs068_079_080=# insert into food_counter_2 values('ZXTI8P', 'Juice', 60);
INSERT 0 1
cs068_079_080=# insert into food_counter_2 values('JKLMN6', 'Chips', 150);
INSERT 0 1
cs068_079_080=# insert into food_counter_2 values('BNMRT7', 'Popcorn', 100);
INSERT 0 1
cs068_079_080=# insert into food_counter_2 values('UI7655', 'Juice', 60);
INSERT 0 1
cs068_079_080=# rollback;
ROLLBACK
cs068_079_080=# select * from food_counter_2;
ERROR:  relation "food_counter_2" does not exist
LINE 1: select * from food_counter_2;
                      ^
```

(There is an error in one of the screenshots, that is to show that rollback has to be used with savepoint only, thus the error is deliberately there.)

```
cs068_079_080=# begin;
BEGIN
cs068_079_080=# create table food_counter_3 (Ticket_ID varchar(15), food_name varchar(10), food_price int);
CREATE TABLE
cs068_079_080=# insert into food_counter_3 values('1QWE34', 'Popcorn', 100);
INSERT 0 1
cs068_079_080=# insert into food_counter_3 values('2ASKL9', 'Juice', 50);
INSERT 0 1
cs068_079_080=# insert into food_counter_3 values('0YTHGR', 'Chips', 150);
INSERT 0 1
cs068_079_080=# insert into food_counter_3 values('JKLMN6', 'Chips', 150);
INSERT 0 1
cs068_079_080=# savepoint check_pt;
SAVEPOINT
cs068_079_080=# insert into food_counter_3 values('997CFG', 'Popcorn', 100);
INSERT 0 1
cs068_079_080=# insert into food_counter_3 values('UI7655', 'Juice', 60);
INSERT 0 1
cs068_079_080=# insert into food_counter_3 values('ZXTI8P', 'Juice', 60);
INSERT 0 1
cs068_079_080=# insert into food_counter_3 values('BNMRT7', 'Popcorn', 100);
INSERT 0 1
cs068_079_080=# rollback to check_pt;
ROLLBACK
cs068_079_080=# select * from food_counter_3;
 ticket_id | food_name | food_price
-----------+-----------+------------
 1QWE34    | Popcorn   |        100
 2ASKL9    | Juice     |         50
 0YTHGR    | Chips     |        150
 JKLMN6    | Chips     |        150
(4 rows)
```

E) CREATING USERS

Multiple users with different access privilege levels for different parts of the database are created.

```
cs068_079_080=# create user USER1 with password '334569' createdb;
CREATE ROLE
cs068_079_080=# create user USER2 with password '120087' createdb;
CREATE ROLE
cs068_079_080=# create user USER3 with password '998223' createdb;
CREATE ROLE
cs068_079_080=# create user USER4 with password '007754' createdb;
CREATE ROLE
cs068_079_080=# create user USER5 with password '265431' createdb;
CREATE ROLE
```

F) GRANTS GIVEN TO USERS:

User 1: Select grants given for Movies and Regions tables.

```
cs068_079_080=# grant select on Movies to USER1;
GRANT
cs068_079_080=# grant select on Region to USER1;
GRANT
```

User 2: All grants are given for the Users table.

```
cs068_079_080=# grant insert on Users to USER2;
GRANT
```

User 3: All grants are given for Ticket and Booking table.

```
cs068_079_080=# grant all on Ticket to USER3;
GRANT
cs068_079_080=# grant all on Booking to USER3;
GRANT
```

User 4: Delete grants given for Languages table while delete and update grants are given for Show table.

```
cs068_079_080=# grant delete on Languages to USER4;
GRANT
cs068_079_080=# grant delete, update on Show to USER4;
GRANT
```

User 5: All grants given to Screen table.

```
cs068_079_080=# grant all on Screen to USER5;
GRANT
```

INDIVIDUAL CONTRIBUTIONS OF MEMBERS:

Ankita	PES1UG19CS068	Simple, Nested, Complex Queries, Adding screenshots and formatting report	2 hours
Apoorva	PES1UG19CS080	Simple, Complex Queries, User Access, and Transaction Queries	2 hours
Anvika	PES1UG19CS079	Simple, Nested, Complex Queries	2 hours