UE19CS301 – DBMS Assignment 3

B R Pratheek – PES1UG19CS101
Arvind Krishna – PES1UG19CS090
Anurag Khanra – PES1UG19CS072

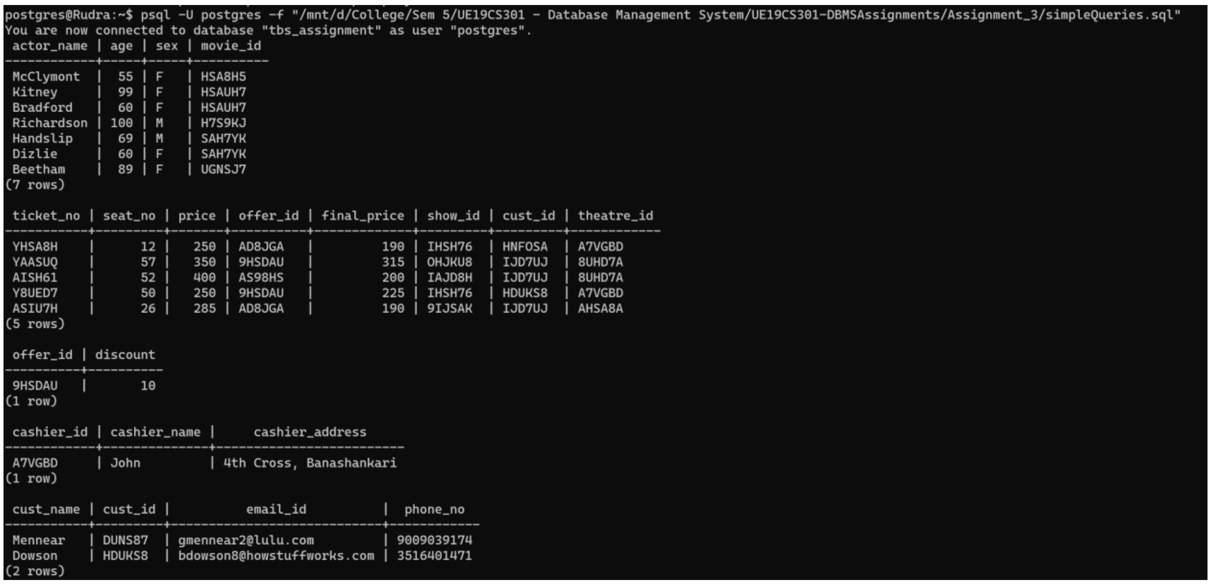
Ticket Booking System

Problem statement

Our DBMS application is designed to tackle the ticket booking system in theatres and cinemas.

Simple Queries:

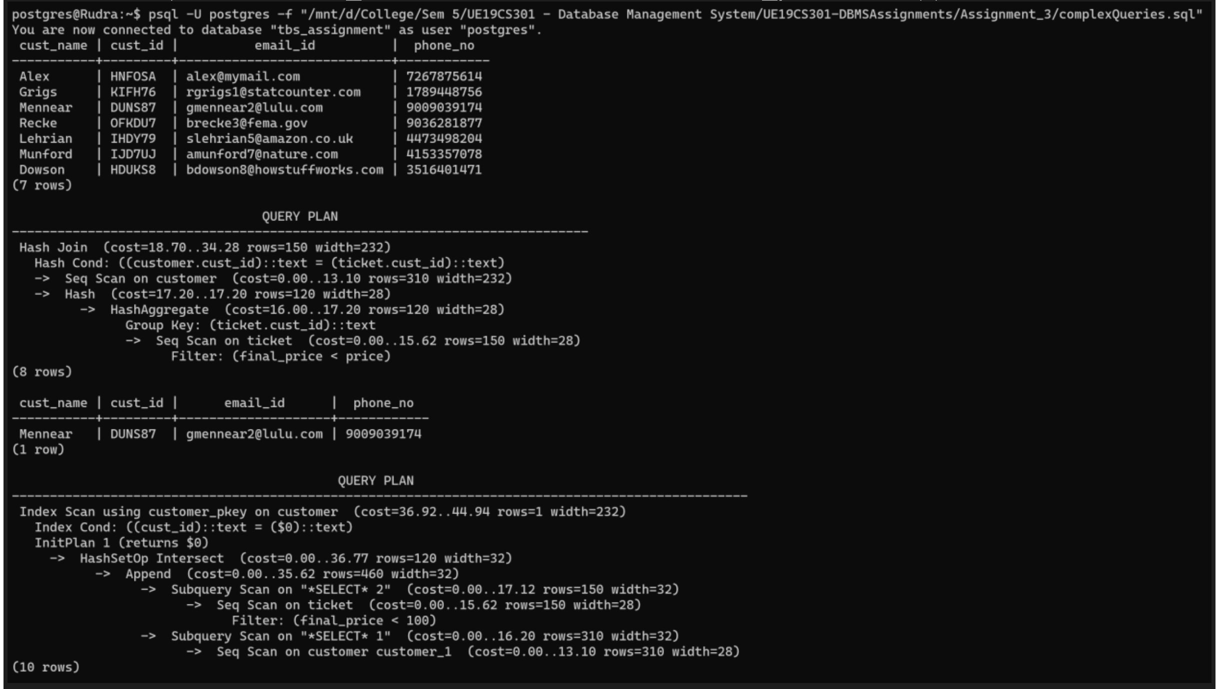
```sql
\c tbs_assignment

SELECT * from actors WHERE AGE > 50;
SELECT * from ticket WHERE price > 200;
SELECT * from offer where discount < 15;
SELECT * FROM cashier where cashier_name = 'John';
SELECT * FROM customer where (phone_no = '9009039174' or phone_no =
'3516401471');
```



```
postgres@Rudra:~$ psql -U postgres -f "/mnt/d/College/Sem 5/UE19CS301 - Database Management System/UE19CS301-DBMSAssignments/Assignment_3/simpleQueries.sql"
You are now connected to database "tbs_assignment" as user "postgres".
 actor_name | age | sex | movie_id
------------+-----+-----+----------
 McClymont  |  55 | F   | HSA8H5
 Kitney     |  99 | F   | HSAUH7
 Bradford   |  60 | F   | HSAUH7
 Richardson | 100 | M   | H7S9KJ
 Handslip   |  69 | M   | SAH7YK
 Dizlie     |  60 | F   | SAH7YK
 Beetham    |  89 | F   | UGNSJ7
(7 rows)

 ticket_no | seat_no | price | offer_id | final_price | show_id | cust_id | theatre_id
-----------+---------+-------+----------+-------------+---------+---------+------------
 YHSA8H    |      12 |   250 | AD8JGA   |         190 | IHSH76  | HNFOSA  | A7VGBD
 YAASUQ    |      57 |   350 | 9HSDAU   |         315 | OHJKU8  | IJD7UJ  | 8UHD7A
 AISH61    |      52 |   400 | AS98HS   |         200 | IAJD8H  | IJD7UJ  | 8UHD7A
 Y8UED7    |      50 |   250 | 9HSDAU   |         225 | IHSH76  | HDUKS8  | A7VGBD
 ASIU7H    |      26 |   285 | AD8JGA   |         190 | 9IJSAK  | IJD7UJ  | AHSA8A
(5 rows)

 offer_id | discount
----------+----------
 9HSDAU   |       10
(1 row)

 cashier_id | cashier_name |     cashier_address
------------+--------------+------------------------
 A7VGBD     | John         | 4th Cross, Banashankari
(1 row)

 cust_name | cust_id |          email_id            |  phone_no
-----------+---------+------------------------------+------------
 Mennear   | DUNS87  | gmennear2@lulu.com           | 9009039174
 Dowson    | HDUKS8  | bdowson8@howstuffworks.com   | 3516401471
(2 rows)
```

Complex Queries and Performance Analysis using “Explain” clause:

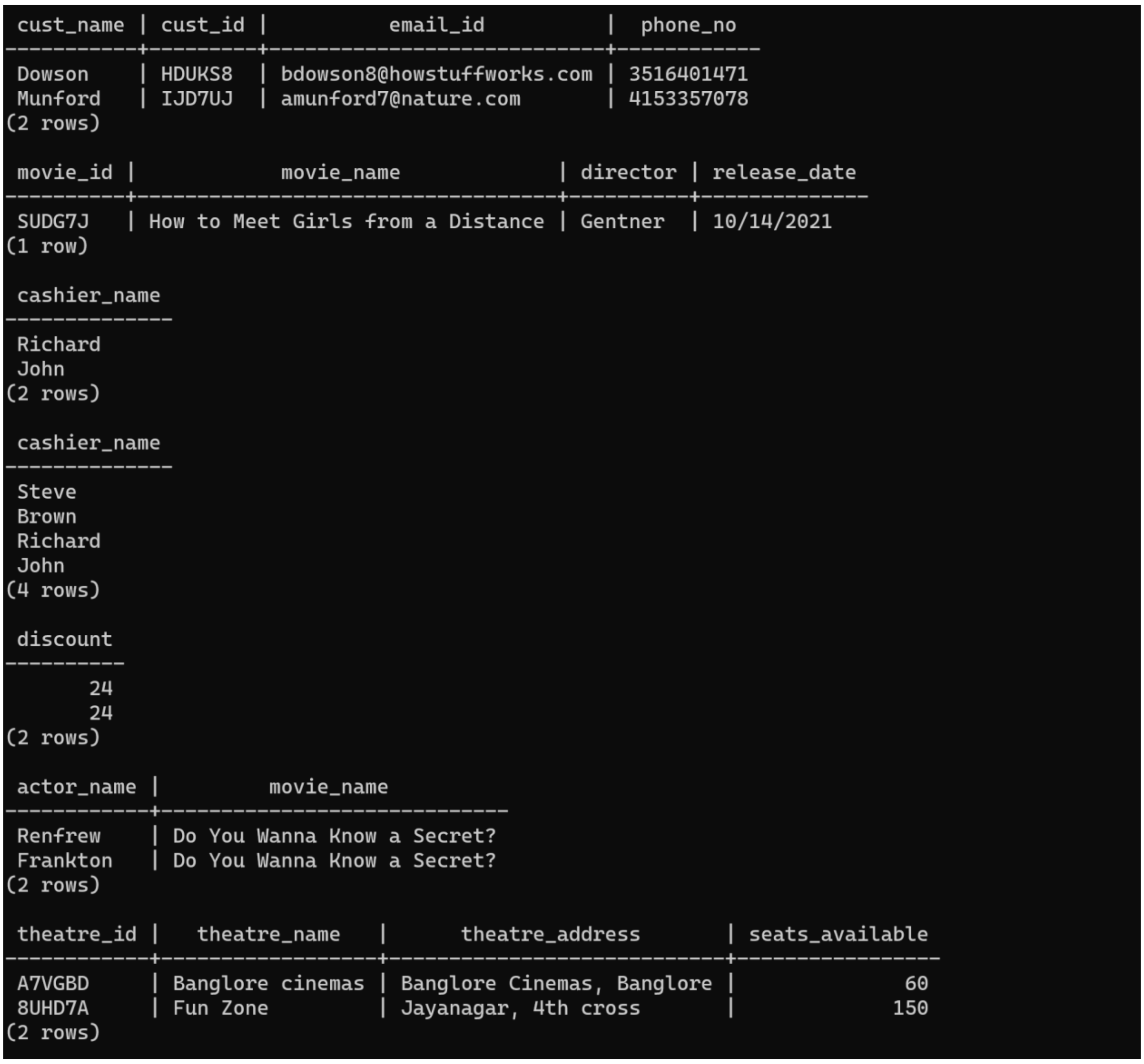
```
\c tbs_assignment

--Some more sorta simple queries
-- SELECT * from customer, booking where (customer.cust_id = booking.cust_id);

-- SELECT seat_no, final_price, screen_no from ticket, shows where
((ticket.show_id = shows.show_id) and (shows.screen_no = 3));

--select all customer whose price was more than $100 after applying offer
select * from customer where cust_id in (select cust_ID from ticket where
ticket.final_price < ticket.price);
explain select * from customer where cust_id in (select cust_ID from ticket
where ticket.final_price < ticket.price);

-- select all customer whose price was less than $100
select * from customer where cust_id=((select cust_id from customer) intersect
(select cust_id from ticket where final_price<100));
explain select * from customer where cust_id=((select cust_id from customer)
intersect (select cust_id from ticket where final_price<100));
```



```
postgres@Rudra:~$ psql -U postgres -f "/mnt/d/College/Sem 5/UE19CS301 - Database Management System/UE19CS301-DBMSAssignments/Assignment_3/complexQueries.sql"
You are now connected to database "tbs_assignment" as user "postgres".
 cust_name | cust_id |          email_id          |  phone_no
-----------+---------+----------------------------+------------
 Alex      | HNFOSA  | alex@mymail.com            | 7267875614
 Grigs     | KIFH76  | rgrigs1@statcounter.com    | 1789448756
 Mennear   | DUNS87  | gmennear2@lulu.com         | 9009039174
 Recke     | OFKDU7  | brecke3@fema.gov           | 9036281877
 Lehrian   | IHDY79  | slehrian5@amazon.co.uk     | 4473498204
 Munford   | IJD7UJ  | amunford7@nature.com       | 4153357078
 Dowson    | HDUKS8  | bdowson8@howstuffworks.com | 3516401471
(7 rows)

                                   QUERY PLAN
-----------------------------------------------------------------------------
 Hash Join  (cost=18.70..34.28 rows=150 width=232)
   Hash Cond: ((customer.cust_id)::text = (ticket.cust_id)::text)
   ->  Seq Scan on customer  (cost=0.00..13.10 rows=310 width=232)
   ->  Hash  (cost=17.20..17.20 rows=120 width=28)
         ->  HashAggregate  (cost=16.00..17.20 rows=120 width=28)
               Group Key: (ticket.cust_id)::text
               ->  Seq Scan on ticket  (cost=0.00..15.62 rows=150 width=28)
                     Filter: (final_price < price)
(8 rows)

 cust_name | cust_id |      email_id      |  phone_no
-----------+---------+--------------------+------------
 Mennear   | DUNS87  | gmennear2@lulu.com | 9009039174
(1 row)

                                              QUERY PLAN
---------------------------------------------------------------------------------------------------
 Index Scan using customer_pkey on customer  (cost=36.92..44.94 rows=1 width=232)
   Index Cond: ((cust_id)::text = ($0)::text)
   InitPlan 1 (returns $0)
     ->  HashSetOp Intersect  (cost=0.00..36.77 rows=120 width=32)
           ->  Append  (cost=0.00..35.62 rows=460 width=32)
                 ->  Subquery Scan on "*SELECT* 2"  (cost=0.00..17.12 rows=150 width=32)
                       ->  Seq Scan on ticket  (cost=0.00..15.62 rows=150 width=28)
                             Filter: (final_price < 100)
                 ->  Subquery Scan on "*SELECT* 1"  (cost=0.00..16.20 rows=310 width=32)
                       ->  Seq Scan on customer customer_1  (cost=0.00..13.10 rows=310 width=28)
(10 rows)
```

```
-- Customer queries:
-- select customer who have availed 10% off of their ticket
select * from customer where cust_ID in (select cust_ID from ticket where
offer_ID in (select offer_ID from offer where discount = 10));

-- select the maximum amount paid for a ticket and give details who paid it
and for what movie
select * from movie where movie_id in (select movie_id from shows as S inner
join (select show_id from ticket where final_price=(select max(final_price)
from ticket)) as Q on Q.show_id=S.show_id);
```

```sql
-- Cashier queries:
-- select the cashier who has sold the ticket to a particular person
select cashier_name from cashier as C,sale as S where
C.cashier_id=S.cashier_id and ticket_no in (select ticket_no from ticket as T
where T.cust_id in (select cust_id from customer where cust_name='Alex'));

-- select the cashier who has sold the ticket for a particular movie
select C.cashier_name from cashier as C where cashier_id in(select
S.cashier_id from sale as S where S.ticket_no in (select ticket_no from ticket
as T where T.show_id in ((select show_id from movie as M ,shows as S where
movie_name='Fame' and M.movie_id=S.movie_id) union (select show_id from movie
as M ,shows as S where movie_name='Flicker' and M.movie_id=S.movie_id))));

-- check the discount offered on the ticket of a particular customer
select discount from offer as O inner join (select offer_id from ticket as T
inner join customer as C on C.cust_id=T.cust_id and C.cust_name='Alex') as Q
on Q.offer_id=O.offer_id ;

-- Actor queries:
-- select the actors acting in a given movie
select actors.Actor_name, movie.movie_name from actors,movie where
actors.movie_id=movie.movie_id and movie.movie_name = 'Do You Wanna Know a
Secret?';

--Theatre queries:
--return the theatre details that runs a movie directed by some specific
director
select * from theatre where theatre_id in (select theatre_id from shows where
movie_id=((select movie_id from movie where director='Gentner') intersect
(select movie_id from shows)));
```

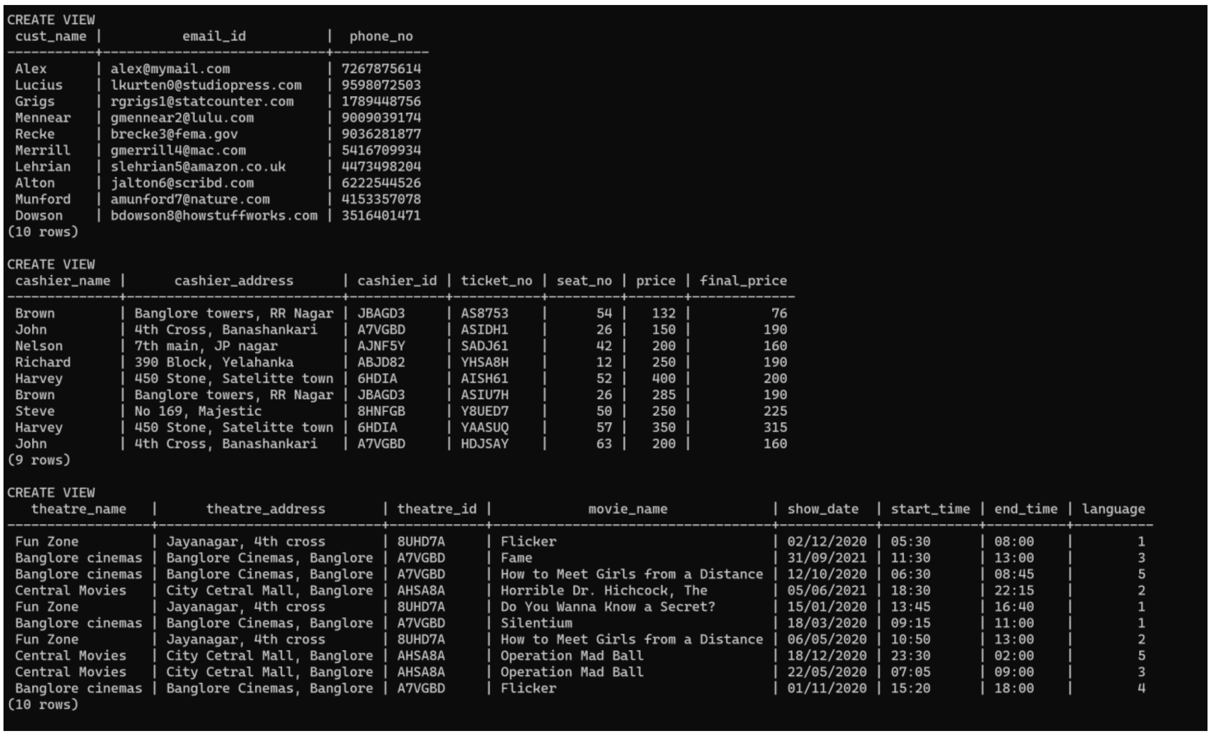


```
 cust_name | cust_id |          email_id           |  phone_no
-----------+---------+-----------------------------+------------
 Dowson    | HDUKS8  | bdowson8@howstuffworks.com | 3516401471
 Munford   | IJD7UJ  | amunford7@nature.com        | 4153357078
(2 rows)

 movie_id |            movie_name             | director | release_date
----------+-----------------------------------+----------+--------------
 SUDG7J   | How to Meet Girls from a Distance | Gentner  | 10/14/2021
(1 row)

 cashier_name
--------------
 Richard
 John
(2 rows)

 cashier_name
--------------
 Steve
 Brown
 Richard
 John
(4 rows)

 discount
----------
       24
       24
(2 rows)

 actor_name |         movie_name
------------+----------------------------
 Renfrew    | Do You Wanna Know a Secret?
 Frankton   | Do You Wanna Know a Secret?
(2 rows)

 theatre_id |   theatre_name    |      theatre_address        | seats_available
------------+-------------------+-----------------------------+-----------------
 A7VGBD     | Banglore cinemas | Banglore Cinemas, Banglore |              60
 8UHD7A     | Fun Zone          | Jayanagar, 4th cross        |             150
(2 rows)
```

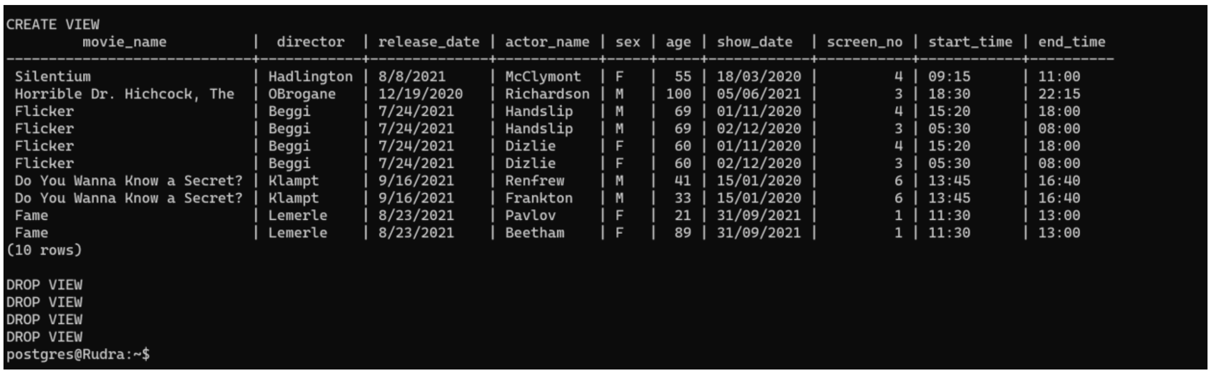
Creating different Views:

```
\c tbs_assignment

--create multiple views of the database

 drop view IF EXISTS *;
--customer view
create view customer_view as select cust_name, email_id, phone_no from
customer;
select * from customer_view;

--sale view
create view cashiers_sale_view as select cashier_name, cashier_address,
cashier.cashier_ID, ticket.ticket_no, seat_no, price, final_price from
cashier, sale, ticket where (sale.cashier_ID = cashier.cashier_ID and
sale.ticket_no = ticket.ticket_no);
select * from cashiers_sale_view;
```

```sql
--theatre view
create view theatre_view as select theatre_name, theatre_address,
theatre.theatre_ID, movie_name, show_date, start_time, end_time, language from
theatre, shows, movie where (theatre.theatre_ID = shows.theatre_ID and
shows.movie_ID = movie.movie_ID);
select * from theatre_view;

--movie view
create view movie_view as select movie_name, director, release_date,
Actor_name, Sex, Age, show_date, screen_no, start_time, end_time from movie,
actors, shows where (movie.movie_ID = shows.movie_ID and actors.movie_ID =
movie.movie_ID);
select * from movie_view;

drop view customer_view;
drop view cashiers_sale_view;
drop view theatre_view;
drop view movie_view;
```



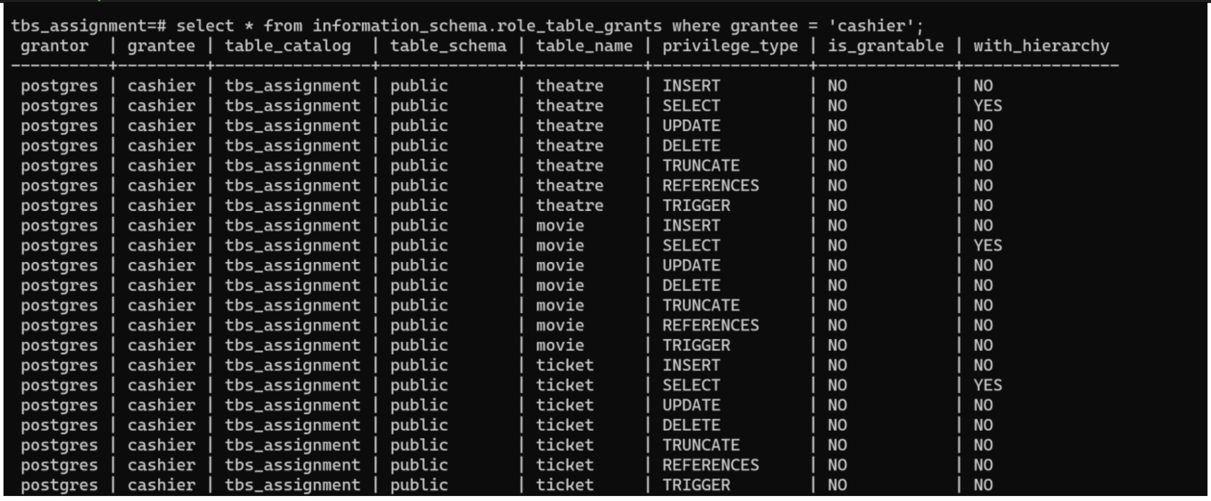
```
CREATE VIEW
 cust_name |           email_id           |  phone_no
-----------+------------------------------+------------
 Alex      | alex@mymail.com              | 7267875614
 Lucius    | lkurten0@studiopress.com     | 9598072503
 Grigs     | rgrigs1@statcounter.com      | 1789448756
 Mennear   | gmennear2@lulu.com           | 9009039174
 Recke     | brecke3@fema.gov             | 9036281877
 Merrill   | gmerrill4@mac.com            | 5416709934
 Lehrian   | slehrian5@amazon.co.uk       | 4473498204
 Alton     | jalton6@scribd.com           | 6222544526
 Munford   | amunford7@nature.com         | 4153357078
 Dowson    | bdowson8@howstuffworks.com   | 3516401471
(10 rows)

CREATE VIEW
 cashier_name |      cashier_address       | cashier_id | ticket_no | seat_no | price | final_price
--------------+----------------------------+------------+-----------+---------+-------+-------------
 Brown        | Banglore towers, RR Nagar  | JBAGD3     | AS8753    |      54 |   132 |          76
 John         | 4th Cross, Banashankari    | A7VGBD     | ASIDH1    |      26 |   150 |         190
 Nelson       | 7th main, JP nagar         | AJNF5Y     | SADJ61    |      42 |   200 |         160
 Richard      | 390 Block, Yelahanka       | ABJD82     | YHSA8H    |      12 |   250 |         190
 Harvey       | 450 Stone, Satelitte town  | 6HDIA      | AISH61    |      52 |   400 |         200
 Brown        | Banglore towers, RR Nagar  | JBAGD3     | ASIU7H    |      26 |   285 |         190
 Steve        | No 169, Majestic           | 8HNFGB     | Y8UED7    |      50 |   250 |         225
 Harvey       | 450 Stone, Satelitte town  | 6HDIA      | YAASUQ    |      57 |   350 |         315
 John         | 4th Cross, Banashankari    | A7VGBD     | HDJSAY    |      63 |   200 |         160
(9 rows)

CREATE VIEW
   theatre_name   |      theatre_address       | theatre_id |             movie_name             | show_date  | start_time | end_time | language
------------------+----------------------------+------------+------------------------------------+------------+------------+----------+----------
 Fun Zone         | Jayanagar, 4th cross       | 8UHD7A     | Flicker                            | 02/12/2020 | 05:30      | 08:00    |        1
 Banglore cinemas | Banglore Cinemas, Banglore | A7VGBD     | Fame                               | 31/09/2021 | 11:30      | 13:00    |        3
 Banglore cinemas | Banglore Cinemas, Banglore | A7VGBD     | How to Meet Girls from a Distance  | 12/10/2020 | 06:30      | 08:45    |        5
 Central Movies   | City Cetral Mall, Banglore | AHSA8A     | Horrible Dr. Hichcock, The         | 05/06/2021 | 18:30      | 22:15    |        2
 Fun Zone         | Jayanagar, 4th cross       | 8UHD7A     | Do You Wanna Know a Secret?        | 15/01/2020 | 13:45      | 16:40    |        1
 Banglore cinemas | Banglore Cinemas, Banglore | A7VGBD     | Silentium                          | 18/03/2020 | 09:15      | 11:00    |        1
 Fun Zone         | Jayanagar, 4th cross       | 8UHD7A     | How to Meet Girls from a Distance  | 06/05/2020 | 10:50      | 13:00    |        2
 Central Movies   | City Cetral Mall, Banglore | AHSA8A     | Operation Mad Ball                 | 18/12/2020 | 23:30      | 02:00    |        5
 Central Movies   | City Cetral Mall, Banglore | AHSA8A     | Operation Mad Ball                 | 22/05/2020 | 07:05      | 09:00    |        3
 Banglore cinemas | Banglore Cinemas, Banglore | A7VGBD     | Flicker                            | 01/11/2020 | 15:20      | 18:00    |        4
(10 rows)
```



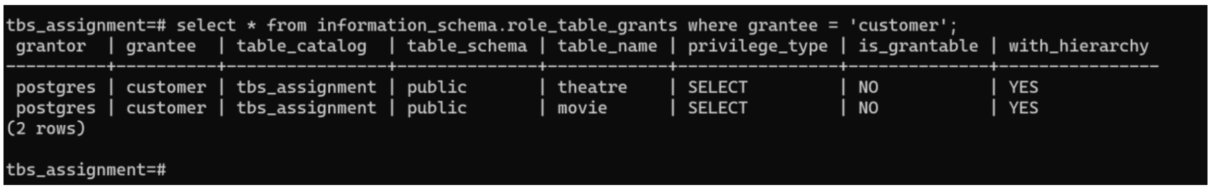
```
CREATE VIEW
         movie_name          |  director  | release_date | actor_name | sex | age | show_date  | screen_no | start_time | end_time
-----------------------------+------------+--------------+------------+-----+-----+------------+-----------+------------+----------
 Silentium                   | Hadlington | 8/8/2021     | McClymont  | F   |  55 | 18/03/2020 |         4 | 09:15      | 11:00
 Horrible Dr. Hichcock, The  | OBrogane   | 12/19/2020   | Richardson | M   | 100 | 05/06/2021 |         3 | 18:30      | 22:15
 Flicker                     | Beggi      | 7/24/2021    | Handslip   | M   |  69 | 01/11/2020 |         4 | 15:20      | 18:00
 Flicker                     | Beggi      | 7/24/2021    | Handslip   | M   |  69 | 02/12/2020 |         3 | 05:30      | 08:00
 Flicker                     | Beggi      | 7/24/2021    | Dizlie     | F   |  60 | 01/11/2020 |         4 | 15:20      | 18:00
 Flicker                     | Beggi      | 7/24/2021    | Dizlie     | F   |  60 | 02/12/2020 |         3 | 05:30      | 08:00
 Do You Wanna Know a Secret? | Klampt     | 9/16/2021    | Renfrew    | M   |  41 | 15/01/2020 |         6 | 13:45      | 16:40
 Do You Wanna Know a Secret? | Klampt     | 9/16/2021    | Frankton   | M   |  33 | 15/01/2020 |         6 | 13:45      | 16:40
 Fame                        | Lemerle    | 8/23/2021    | Pavlov     | F   |  21 | 31/09/2021 |         1 | 11:30      | 13:00
 Fame                        | Lemerle    | 8/23/2021    | Beetham    | F   |  89 | 31/09/2021 |         1 | 11:30      | 13:00
(10 rows)

DROP VIEW
DROP VIEW
DROP VIEW
DROP VIEW
postgres@Rudra:~$
```

Creating Roles/Users:

```
\c tbs_assignment

-- revoke all on database tbs_assignment from cashier;
-- revoke all on database tbs_assignment from cashier;
-- drop user  cashier;
-- drop user  customer;

create user cashier with encrypted password 'cashier';
create user customer with encrypted password 'customer';

grant all on theatre, movie, ticket, shows, sale, booking to cashier;
grant select on theatre, movie, shows to customer;

--Uncomment to see all access privilages to the different users

select * from information_schema.role_table_grants where grantee = 'cashier';
select * from information_schema.role_table_grants where grantee = 'customer';

-- drop user cashier
-- drop user customer
```



```
tbs_assignment=# select * from information_schema.role_table_grants where grantee = 'cashier';
 grantor  | grantee | table_catalog  | table_schema | table_name | privilege_type | is_grantable | with_hierarchy
----------+---------+----------------+--------------+------------+----------------+--------------+----------------
 postgres | cashier | tbs_assignment | public       | theatre    | INSERT         | NO           | NO
 postgres | cashier | tbs_assignment | public       | theatre    | SELECT         | NO           | YES
 postgres | cashier | tbs_assignment | public       | theatre    | UPDATE         | NO           | NO
 postgres | cashier | tbs_assignment | public       | theatre    | DELETE         | NO           | NO
 postgres | cashier | tbs_assignment | public       | theatre    | TRUNCATE       | NO           | NO
 postgres | cashier | tbs_assignment | public       | theatre    | REFERENCES     | NO           | NO
 postgres | cashier | tbs_assignment | public       | theatre    | TRIGGER        | NO           | NO
 postgres | cashier | tbs_assignment | public       | movie      | INSERT         | NO           | NO
 postgres | cashier | tbs_assignment | public       | movie      | SELECT         | NO           | YES
 postgres | cashier | tbs_assignment | public       | movie      | UPDATE         | NO           | NO
 postgres | cashier | tbs_assignment | public       | movie      | DELETE         | NO           | NO
 postgres | cashier | tbs_assignment | public       | movie      | TRUNCATE       | NO           | NO
 postgres | cashier | tbs_assignment | public       | movie      | REFERENCES     | NO           | NO
 postgres | cashier | tbs_assignment | public       | movie      | TRIGGER        | NO           | NO
 postgres | cashier | tbs_assignment | public       | ticket     | INSERT         | NO           | NO
 postgres | cashier | tbs_assignment | public       | ticket     | SELECT         | NO           | YES
 postgres | cashier | tbs_assignment | public       | ticket     | UPDATE         | NO           | NO
 postgres | cashier | tbs_assignment | public       | ticket     | DELETE         | NO           | NO
 postgres | cashier | tbs_assignment | public       | ticket     | TRUNCATE       | NO           | NO
 postgres | cashier | tbs_assignment | public       | ticket     | REFERENCES     | NO           | NO
 postgres | cashier | tbs_assignment | public       | ticket     | TRIGGER        | NO           | NO
```



```
tbs_assignment=# select * from information_schema.role_table_grants where grantee = 'customer';
 grantor  | grantee  | table_catalog  | table_schema | table_name | privilege_type | is_grantable | with_hierarchy
----------+----------+----------------+--------------+------------+----------------+--------------+----------------
 postgres | customer | tbs_assignment | public       | theatre    | SELECT         | NO           | YES
 postgres | customer | tbs_assignment | public       | movie      | SELECT         | NO           | YES
(2 rows)

tbs_assignment=#
```

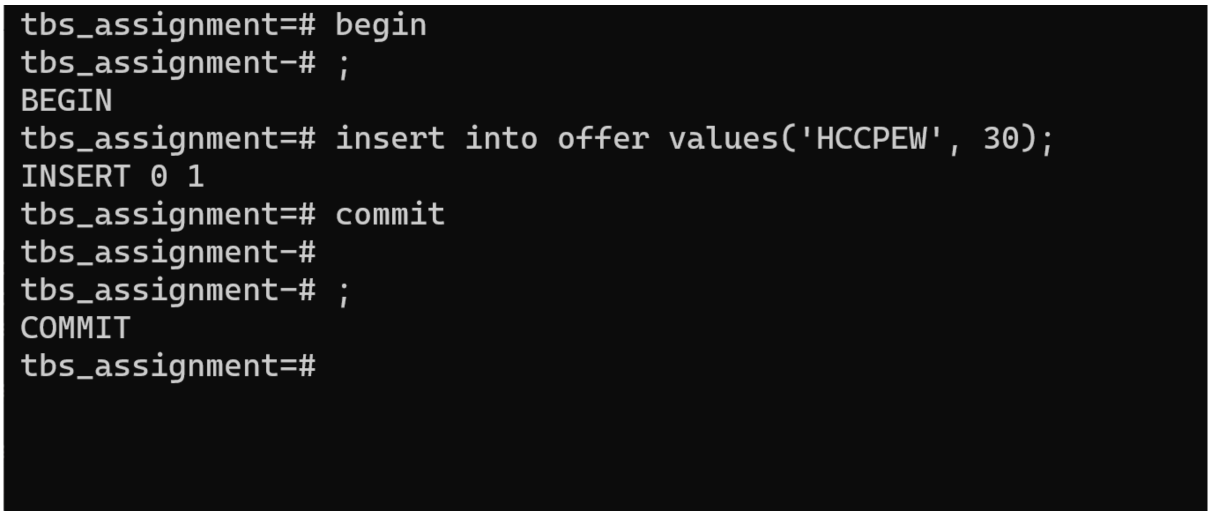
Concurrency Control:

Read Committed Isolation Level (Default) - A statement can only see rows committed before it began.

- Open two terminals and connect to the tbs_assignment database on both.
- The first select statement is used to view the inventory before the insert statement.
- Insert values into the inventory on Terminal 2
- Before commit statement is executed in Terminal 2, we see that the changes aren't reflected (second select statement in Terminal 1)
- After commit statement is executed in Terminal 2, we can see that the changes have now been reflected (third select statement in Terminal 1)

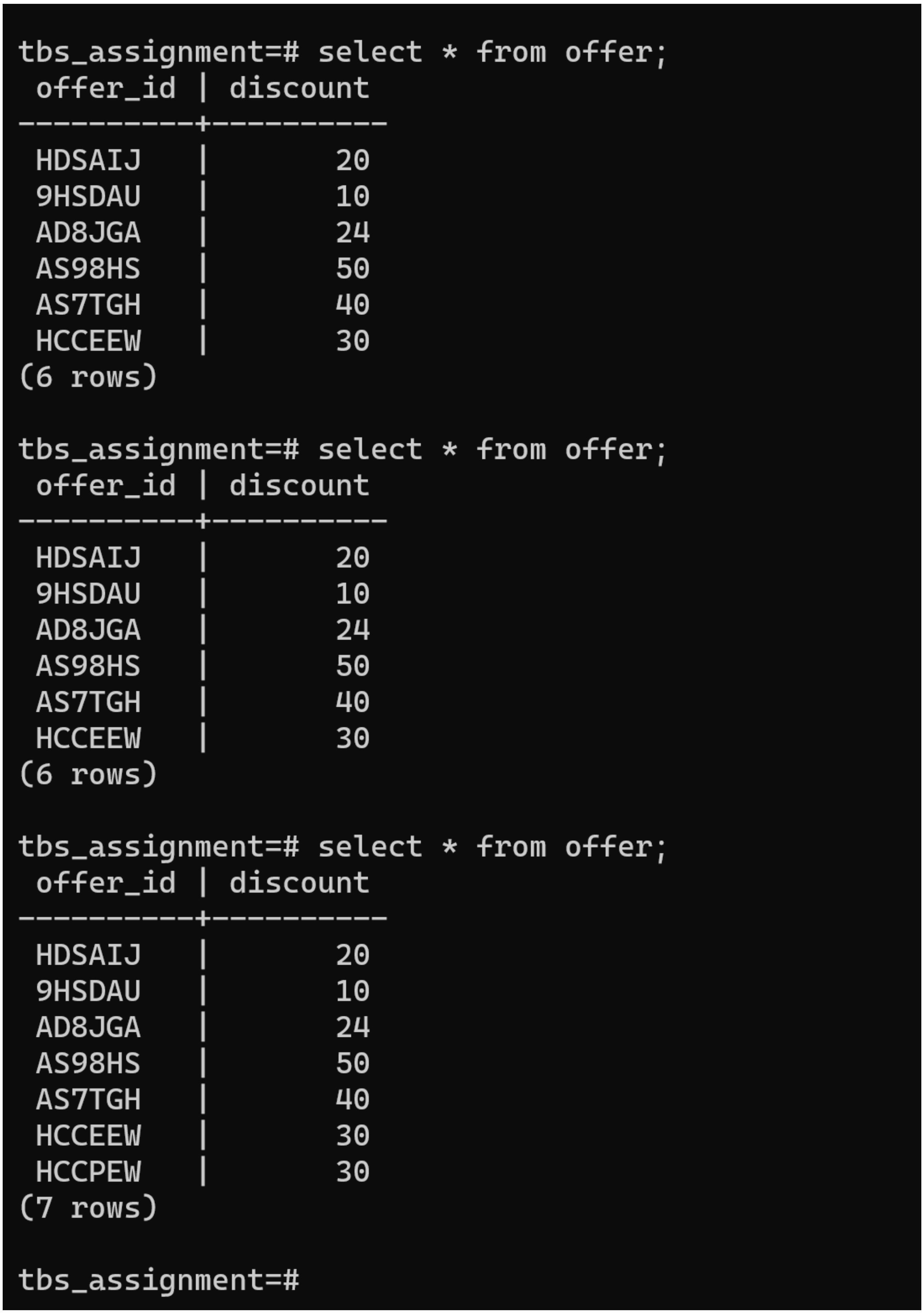
Terminal 2

`



```
tbs_assignment=# begin
tbs_assignment-# ;
BEGIN
tbs_assignment=# insert into offer values('HCCPEW', 30);
INSERT 0 1
tbs_assignment=# commit
tbs_assignment-#
tbs_assignment-# ;
COMMIT
tbs_assignment=#
```

Terminal 1

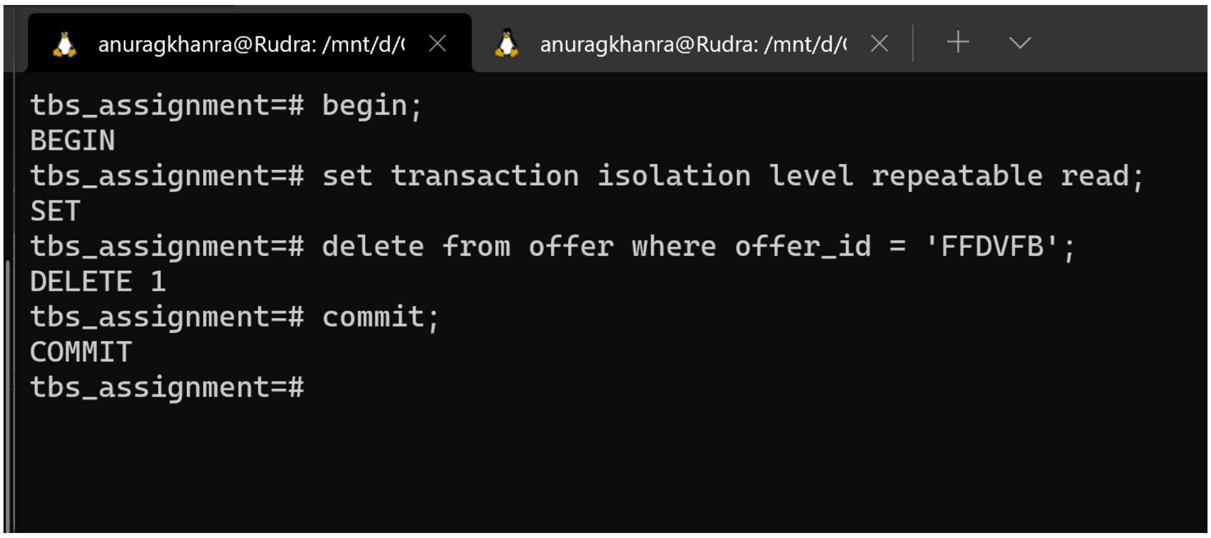


```
tbs_assignment=# select * from offer;
 offer_id | discount
----------+----------
 HDSAIJ   |       20
 9HSDAU   |       10
 AD8JGA   |       24
 AS98HS   |       50
 AS7TGH   |       40
 HCCEEW   |       30
(6 rows)

tbs_assignment=# select * from offer;
 offer_id | discount
----------+----------
 HDSAIJ   |       20
 9HSDAU   |       10
 AD8JGA   |       24
 AS98HS   |       50
 AS7TGH   |       40
 HCCEEW   |       30
(6 rows)

tbs_assignment=# select * from offer;
 offer_id | discount
----------+----------
 HDSAIJ   |       20
 9HSDAU   |       10
 AD8JGA   |       24
 AS98HS   |       50
 AS7TGH   |       40
 HCCEEW   |       30
 HCCPEW   |       30
(7 rows)

tbs_assignment=#
```

Repeatable Read Isolation Level - All statements of the current transaction can only see rows committed before the first query or data-modification statement was executed in this transaction

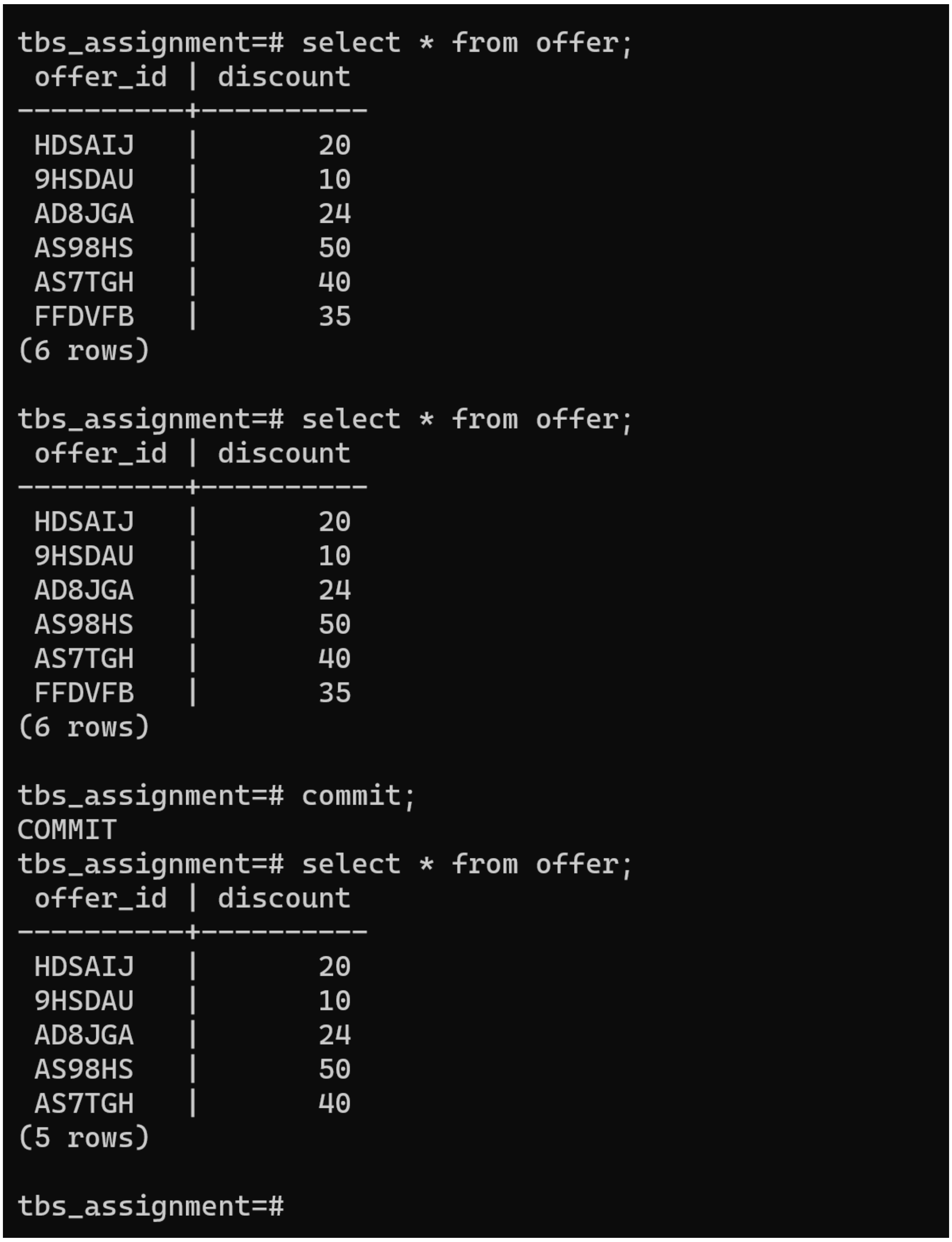
- Open two terminals and connect to the tbs_assignment database on both.
- Set Transaction Isolation Level to Repeatable Read.
- The first select statement in Terminal 1 shows the inventory after the delete statement is executed in Terminal 2. (No change reflected)
- The second select statement in Terminal 1 shows the inventory after commit statement is executed in Terminal 2. (No change reflected)
- The third select statement shows the inventory after the commit statement is executed on Terminal 1. Here we see that the delete operation has finally been reflected.

Terminal 2



```
tbs_assignment=# begin;
BEGIN
tbs_assignment=# set transaction isolation level repeatable read;
SET
tbs_assignment=# delete from offer where offer_id = 'FFDVFB';
DELETE 1
tbs_assignment=# commit;
COMMIT
tbs_assignment=#
```

Terminal 1



```
tbs_assignment=# select * from offer;
 offer_id | discount
----------+----------
 HDSAIJ   |       20
 9HSDAU   |       10
 AD8JGA   |       24
 AS98HS   |       50
 AS7TGH   |       40
 FFDVFB   |       35
(6 rows)

tbs_assignment=# select * from offer;
 offer_id | discount
----------+----------
 HDSAIJ   |       20
 9HSDAU   |       10
 AD8JGA   |       24
 AS98HS   |       50
 AS7TGH   |       40
 FFDVFB   |       35
(6 rows)

tbs_assignment=# commit;
COMMIT
tbs_assignment=# select * from offer;
 offer_id | discount
----------+----------
 HDSAIJ   |       20
 9HSDAU   |       10
 AD8JGA   |       24
 AS98HS   |       50
 AS7TGH   |       40
(5 rows)

tbs_assignment=#
```