CS3120 Database Management Systems Laboratory

Midterm Assignment

Mayank Singla
111901030

Using VSCode to write the queries in a file and then execute that file from the terminal using "`\i file.sql`"

Q1. Add a column genre in the film table and its value can't be null and text length should not be more than 100. [Hint: Not Null constraint will be added after inserting values.]

solution.sql U

dbms-lab > labwork > midterm_assignment > solution.sql > ...

Run on active connection | Select block

```sql
/**********************************************************************/
/* Q1 */
```

Run SQL

```sql
ALTER TABLE film
ADD COLUMN genre VARCHAR(100);
```

Run SQL

```sql
UPDATE film
SET genre = 'Genre';
```

Run SQL

```sql
ALTER TABLE film
ADD CONSTRAINT genre_not_null CHECK (genre IS NOT NULL);
```

```
dvdrental=# \i solution.sql
ALTER TABLE
UPDATE 1000
ALTER TABLE
dvdrental=# \d film
```

```
                                        Table "public.film"
     Column      |            Type             | Collation | Nullable |                 Default
-----------------+-----------------------------+-----------+----------+-----------------------------------
 film_id         | integer                     |           | not null | nextval('film_film_id_seq'::regclass)
 title           | character varying(255)      |           | not null |
 description     | text                        |           |          |
 release_year    | year                        |           |          |
 language_id     | smallint                    |           | not null |
 rental_duration | smallint                    |           | not null | 3
 rental_rate     | numeric(4,2)                |           | not null | 4.99
 length          | smallint                    |           |          |
 replacement_cost | numeric(5,2)               |           | not null | 19.99
 rating          | mpaa_rating                 |           |          | 'G'::mpaa_rating
 last_update     | timestamp without time zone |           | not null | now()
 special_features | text[]                     |           |          |
 fulltext        | tsvector                    |           | not null |
 genre           | character varying(100)      |           |          |
Indexes:
    "film_pkey" PRIMARY KEY, btree (film_id)
    "film_fulltext_idx" gist (fulltext)
    "idx_fk_language_id" btree (language_id)
    "idx_title" btree (title)
Check constraints:
    "genre_not_null" CHECK (genre IS NOT NULL)
```

```
dvdrental=# SELECT film_id, title, genre
dvdrental-# FROM film
dvdrental-# LIMIT 10;
 film_id |       title        | genre
---------+--------------------+-------
     133 | Chamber Italian    | Genre
     384 | Grosse Wonderful   | Genre
       8 | Airport Pollock    | Genre
      98 | Bright Encounters  | Genre
       1 | Academy Dinosaur   | Genre
       2 | Ace Goldfinger     | Genre
       3 | Adaptation Holes   | Genre
       4 | Affair Prejudice   | Genre
       5 | African Egg        | Genre
       6 | Agent Truman       | Genre
(10 rows)
```

Q2. Insert values inside the genre column from existing film category data.

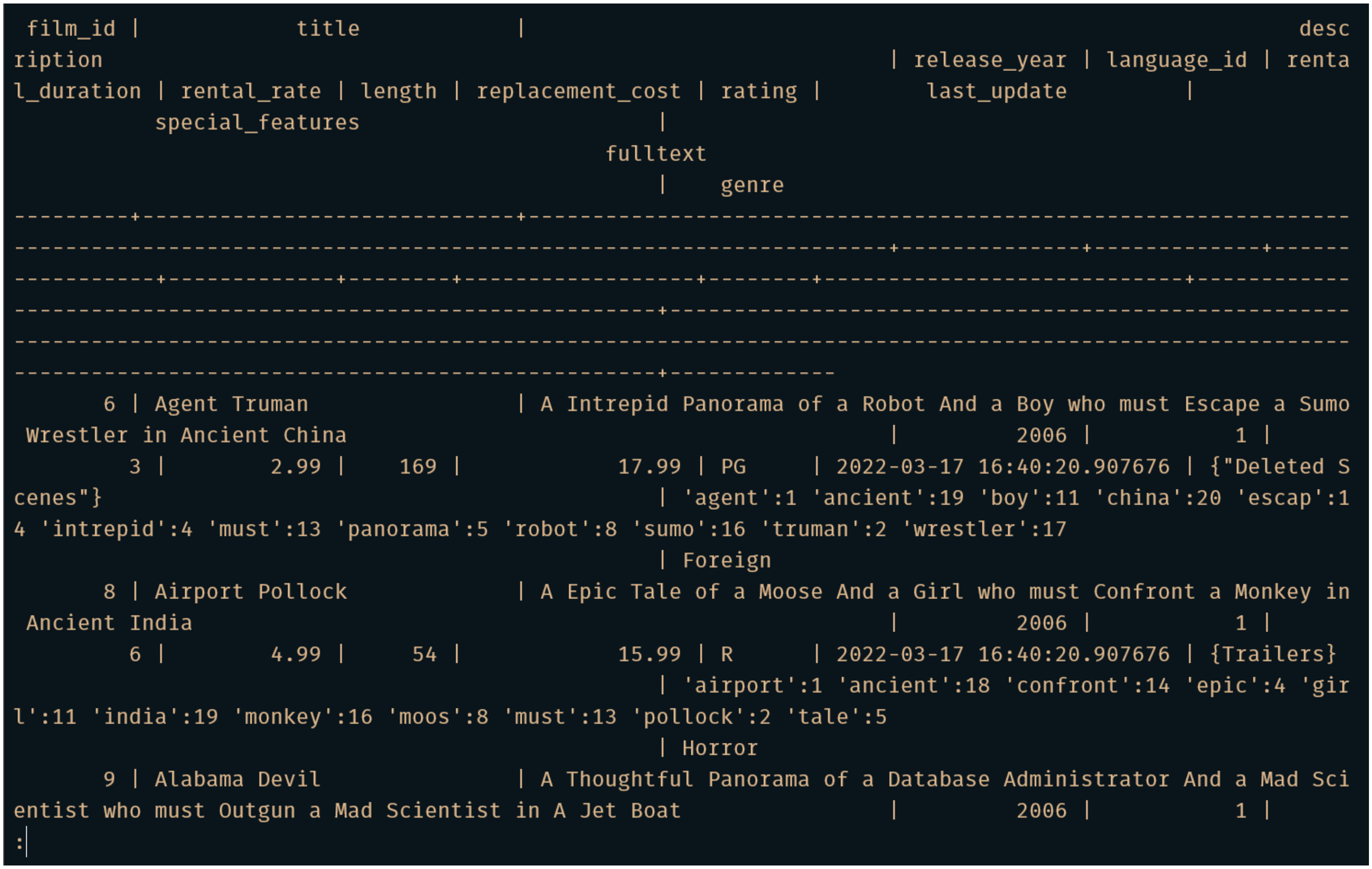
solution.sql U ×

dbms-lab > labwork > midterm_assignment > solution.sql > ...

```sql
/**********************************************************************/
/* Q2 */
▶ Run SQL
UPDATE film
SET genre = category.name
FROM film_category,
    category
WHERE film.film_id = film_category.film_id
    AND film_category.category_id = category.
    category_id;
```

```
dvdrental=# \i solution.sql
UPDATE 1000
dvdrental=# SELECT film_id, title, genre
FROM film
LIMIT 10;
 film_id |      title       |  genre
---------+------------------+----------
       6 | Agent Truman     | Foreign
       8 | Airport Pollock  | Horror
       9 | Alabama Devil    | Horror
      10 | Aladdin Calendar | Sports
      11 | Alamo Videotape  | Foreign
      12 | Alaska Phantom   | Music
      13 | Ali Forever      | Horror
      14 | Alice Fantasia   | Classics
      15 | Alien Center     | Foreign
      16 | Alley Evolution  | Foreign
(10 rows)
```

```
dvdrental=# SELECT * FROM film;
```



```
 film_id |            title            |                                                            desc
ription                                                                 | release_year | language_id | renta
l_duration | rental_rate | length | replacement_cost | rating |        last_update         |
           special_features                      |
                                              fulltext
                                                  |    genre
---------+-----------------------------+---------------------------------------------------------------
-------------------------------------------------------------------+--------------+-------------+------
-----------+-------------+--------+------------------+--------+----------------------------+-----------
---------------------------------------------------+----------------------------------------------------
--------------------------------------------------------------------------------------------------------
---------------------------------------------------+-------------
       6 | Agent Truman                | A Intrepid Panorama of a Robot And a Boy who must Escape a Sumo
 Wrestler in Ancient China                                              |         2006 |           1 |
         3 |        2.99 |    169 |            17.99 | PG     | 2022-03-17 16:40:20.907676 | {"Deleted S
cenes"}                                          | 'agent':1 'ancient':19 'boy':11 'china':20 'escap':1
4 'intrepid':4 'must':13 'panorama':5 'robot':8 'sumo':16 'truman':2 'wrestler':17
                                                  | Foreign
       8 | Airport Pollock             | A Epic Tale of a Moose And a Girl who must Confront a Monkey in
 Ancient India                                                          |         2006 |           1 |
         6 |        4.99 |     54 |            15.99 | R      | 2022-03-17 16:40:20.907676 | {Trailers}
                                                  | 'airport':1 'ancient':18 'confront':14 'epic':4 'gir
l':11 'india':19 'monkey':16 'moos':8 'must':13 'pollock':2 'tale':5
                                                  | Horror
       9 | Alabama Devil               | A Thoughtful Panorama of a Database Administrator And a Mad Sci
entist who must Outgun a Mad Scientist in A Jet Boat                    |         2006 |           1 |
:
```

Q3. Write a query to find the title, category name, and last name of at most 3 cast members of the movies with their title starting with a vowel. (The cast member names should be appended in a single cell as comma-separated values. Hint: you may want to use the 'Limit' command)

solution.sql U

dbms-lab > labwork > midterm_assignment > solution.sql > ...

```sql
/* Q3 */
▶ Run SQL
SELECT film.title,
    film.genre as category,
    (
        SELECT STRING_AGG(
                lname,
                ', '
                ORDER BY lname ASC
            )
        FROM (
                SELECT actor.last_name AS lname
                FROM film_actor,
                    actor
                WHERE film.film_id = film_actor.film_id
                    AND film_actor.actor_id = actor.actor_id
                LIMIT 3
            ) AS top_3_actors
    ) as cast
FROM film
WHERE film.title SIMILAR TO '[aAeEiIoOuU]%';
```

```
dvdrental=# \i solution.sql
```

```
           title              |  category   |             cast
-----------------------------+-------------+------------------------------
 Agent Truman                | Foreign     | Kilmer, Neeson, Paltrow
 Airport Pollock             | Horror      | Davis, Kilmer, Willis
 Alabama Devil               | Horror      | Crawford, Gable, Marx
 Aladdin Calendar            | Sports      | Bolger, Dean, Wayne
 Alamo Videotape             | Foreign     | Cage, Damon, Guiness
 Alaska Phantom              | Music       | Bolger, Crowe, Posey
 Ali Forever                 | Horror      | Berry, Mcconaughey, Torn
 Alice Fantasia              | Classics    | Hoffman, Williams, Zellweger
 Alien Center                | Foreign     | Crowe, Dukakis, Paltrow
 Alley Evolution             | Foreign     | Berry, Cruise, Johansson
 Alone Trip                  | Music       | Berry, Chase, Wood
 Alter Victory               | Animation   | Kilmer, Ryder, Witherspoon
 Amadeus Holy                | Action      | Bolger, Lollobrigida, Mcqueen
 Amelie Hellfighters         | Music       | Hunt, Mansfield, Torn
 American Circus             | Action      | Bloom, Crawford, Crowe
 Amistad Midsummer           | New         | Mcconaughey, Nolte, Wahlberg
 Anaconda Confessions        | Animation   | Davis, Guiness, Marx
 Analyze Hoosiers            | Horror      | Bailey, Mckellen, Miranda
 Angels Life                 | New         | Davis, Guiness, Mostel
 Annie Identity              | Sci-Fi      | Grant, Keitel, Mcqueen
 Anonymous Human             | Sports      | Kilmer, Mostel, Mostel
 Anthem Luke                 | Comedy      | Keitel, Kilmer
 Antitrust Tomatoes          | Action      | Crowe, Nicholson, Wood
 Anything Savannah           | Horror      | Monroe, Swank, West
 Apache Divine               | Family      | Cronyn, Olivier, Wahlberg
:
```

Q4. Display the name and total payment of customers who have made the top 5 total payments in descending order of the total payment made.

solution.sql U ×

dbms-lab > labwork > midterm_assignment > solution.sql > ...

```sql
/* Q4 */
▶ Run SQL
SELECT first_name,
    last_name,
    SUM(amount) as total_payment
FROM customer,
    payment
WHERE customer.customer_id = payment.customer_id
GROUP BY customer.customer_id
ORDER BY total_payment DESC
LIMIT 5;
```

```
dvdrental=# \i solution.sql
 first_name | last_name | total_payment
------------+-----------+---------------
 Eleanor    | Hunt      |        211.55
 Karl       | Seal      |        208.58
 Marion     | Snyder    |        194.61
 Rhonda     | Kennedy   |        191.62
 Clara      | Shaw      |        189.60
(5 rows)

dvdrental=#
```

Q5. Create two user groups called customer and cashier. The customer can only view the movie titles, category names, and cast of the movies. The cashier should only be able to view payment details of customers with their name, no other details of customers should be accessible to the cashier other than the name and payment details. Add a few users in both groups. [Hint: Granting select to particular columns on each table may not always help, instead you can create views and permit views]

solution.sql U ×

dbms-lab > labwork > midterm_assignment > solution.sql > ...

```sql
/* Q5 */
▶ Run SQL
CREATE GROUP customer;
▶ Run SQL
CREATE GROUP cashier;
```

```
dvdrental=# \i solution.sql
CREATE ROLE
CREATE ROLE
dvdrental=# \du
                                   List of roles
 Role name |                         Attributes                         | Member of
-----------+------------------------------------------------------------+-----------
 cashier   | Cannot login                                               | {}
 customer  | Cannot login                                               | {}
 mayank    | Superuser                                                  | {}
 postgres  | Superuser, Create role, Create DB, Replication, Bypass RLS | {}
```

Creating views for both groups for required data.

solution.sql U ×

dbms-lab > labwork > midterm_assignment > solution.sql > ...

▶ Run SQL

```sql
CREATE VIEW customer_view AS
SELECT film.title,
    film.genre as category,
    STRING_AGG(
        actor.first_name || ' ' || actor.last_name,
        ', '
    ) as cast_names
FROM film,
    film_actor,
    actor
WHERE film.film_id = film_actor.film_id
    AND film_actor.actor_id = actor.actor_id
GROUP BY film.film_id;
```

```
dvdrental=# \i solution.sql
CREATE VIEW
dvdrental=# SELECT * FROM customer_view;
```

```
           title            |  category   |
             cast_names
-----------------------------+-------------+-----------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------
 Academy Dinosaur            | Documentary | Rock Dukakis, Mary Keitel, Johnny Cage, Penelope Guiness, Sandra Peck, Christian Gable, O
prah Kilmer, Warren Nolte, Lucille Tracy, Mena Temple
 Ace Goldfinger              | Horror      | Minnie Zellweger, Chris Depp, Bob Fawcett, Sean Guiness
 Adaptation Holes            | Documentary | Cameron Streep, Bob Fawcett, Nick Wahlberg, Ray Johansson, Julianne Dench
 Affair Prejudice            | Horror      | Jodie Degeneres, Kenneth Pesci, Fay Winslet, Oprah Kilmer, Scarlett Damon
 African Egg                 | Family      | Dustin Tautou, Matthew Leigh, Gary Phoenix, Matthew Carrey, Thora Temple
 Agent Truman                | Foreign     | Warren Nolte, Sandra Kilmer, Jayne Neeson, Morgan Williams, Kirsten Paltrow, Kenneth Hoff
man, Reese West
 Airplane Sierra             | Comedy      | Mena Hopper, Jim Mostel, Michael Bolger, Oprah Kilmer, Richard Penn
 Airport Pollock             | Horror      | Lucille Dee, Susan Davis, Fay Kilmer, Gene Willis
 Alabama Devil               | Horror      | William Hackman, Rip Crawford, Rip Winslet, Greta Keitel, Christian Gable, Mena Temple, M
eryl Allen, Warren Nolte, Elvis Marx
 Aladdin Calendar            | Sports      | Greta Malden, Rock Dukakis, Ray Johansson, Renee Tracy, Val Bolger, Judy Dean, Jada Ryder
, Alec Wayne
 Alamo Videotape             | Foreign     | Michael Bening, Johnny Cage, Scarlett Damon, Sean Guiness
 Alaska Phantom              | Music       | Val Bolger, Burt Posey, Gene Mckellen, Jeff Silverstone, Sylvester Dern, Albert Johansson
, Sidney Crowe
 Ali Forever                 | Horror      | Christopher Berry, Kenneth Torn, Cary Mcconaughey, Jon Chase, Morgan Mcdormand
 Alice Fantasia              | Classics    | Woody Hoffman, Rock Dukakis, Minnie Zellweger, Morgan Williams
 Alien Center                | Foreign     | Sidney Crowe, Kenneth Paltrow, Humphrey Willis, Renee Tracy, Burt Dukakis, Mena Hopper
 Alley Evolution             | Foreign     | Gregory Gooding, Jude Cruise, John Suvari, Karl Berry, Albert Johansson
 Alone Trip                  | Music       | Renee Ball, Ed Chase, Spencer Depp, Karl Berry, Laurence Bullock, Woody Jolie, Chris Depp
, Uma Wood
:
```

solution.sql U

dbms-lab > labwork > midterm_assignment > solution.sql > ...

```sql
CREATE VIEW cashier_view AS
SELECT customer.first_name || ' ' || customer.last_name AS customer_name,
    payment.amount,
    payment.payment_date,
    payment.transaction_id,
    payment.payment_info
FROM customer,
    payment
WHERE customer.customer_id = payment.customer_id;
```

```
dvdrental=# \i solution.sql
CREATE VIEW
dvdrental=# SELECT * FROM cashier_view;
```

```
     customer_name      | amount |        payment_date        |            transaction_id            |

                        payment_info

-----------------------+--------+----------------------------+--------------------------------------+--------------------------------
--------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
 Harold Martino        |   5.99 | 2007-02-20 02:11:44.996577 | acf9416b-e353-4435-9f45-9e61f02395e4 | {"intent": "sale", "payer": { "p
ayment_method": "paypal" }, "redirect_urls": { "return_url": "http://return.url", "cancel_url": "http://cancel.url" }, "transactions":
 [{ "item_list": { "items": [{ "name": "item", "sku": "item", "price": "1.00", "currency": "USD", "quantity": "1" }]}, "amount": { "cu
rrency": "USD", "total": "1.00" }, "description": "This is the payment information." }] }
 Harold Martino        |   2.99 | 2007-02-20 13:57:39.996577 | 20d95f27-f636-4c3d-b230-4f8276e1e820 | {"intent": "sale", "payer": { "p
ayment_method": "paypal" }, "redirect_urls": { "return_url": "http://return.url", "cancel_url": "http://cancel.url" }, "transactions":
 [{ "item_list": { "items": [{ "name": "item", "sku": "item", "price": "1.00", "currency": "USD", "quantity": "1" }]}, "amount": { "cu
rrency": "USD", "total": "1.00" }, "description": "This is the payment information." }] }
 Douglas Graf          |   4.99 | 2007-02-16 00:10:50.996577 | 80222e33-8776-4ed3-9c72-c4a0948dcfa8 | {"intent": "sale", "payer": { "p
ayment_method": "paypal" }, "redirect_urls": { "return_url": "http://return.url", "cancel_url": "http://cancel.url" }, "transactions":
 [{ "item_list": { "items": [{ "name": "item", "sku": "item", "price": "1.00", "currency": "USD", "quantity": "1" }]}, "amount": { "cu
rrency": "USD", "total": "1.00" }, "description": "This is the payment information." }] }
 Douglas Graf          |   6.99 | 2007-02-16 01:15:33.996577 | 2a328b64-1cd7-402b-8153-e1633db520c5 | {"intent": "sale", "payer": { "p
ayment_method": "paypal" }, "redirect_urls": { "return_url": "http://return.url", "cancel_url": "http://cancel.url" }, "transactions":
 [{ "item_list": { "items": [{ "name": "item", "sku": "item", "price": "1.00", "currency": "USD", "quantity": "1" }]}, "amount": { "cu
rrency": "USD", "total": "1.00" }, "description": "This is the payment information." }] }
 Douglas Graf          |   0.99 | 2007-02-17 01:26:00.996577 | 7f2d5128-9663-4340-ac9e-6ca14f26032f | {"intent": "sale", "payer": { "p
ayment_method": "paypal" }, "redirect_urls": { "return_url": "http://return.url", "cancel_url": "http://cancel.url" }, "transactions":
 [{ "item_list": { "items": [{ "name": "item", "sku": "item", "price": "1.00", "currency": "USD", "quantity": "1" }]}, "amount": { "cu
:
```

Granting privileges to both groups.

```
solution.sql U ×
dbms-lab > labwork > midterm_assignment > solution.sql > ...
    ▶ Run SQL
83  GRANT SELECT ON customer_view TO customer;
    ▶ Run SQL
84  GRANT SELECT ON cashier_view TO cashier;
85
```

```
dvdrental=# \i solution.sql
GRANT
GRANT
dvdrental=# \z customer_view
                                   Access privileges
 Schema |     Name      | Type |   Access privileges   | Column privileges | Policies
--------+---------------+------+-----------------------+-------------------+----------
 public | customer_view | view | mayank=arwdDxt/mayank+|                   |
        |               |      | customer=r/mayank     |                   |
(1 row)

dvdrental=# \z cashier_view
                                   Access privileges
 Schema |     Name     | Type |   Access privileges   | Column privileges | Policies
--------+--------------+------+-----------------------+-------------------+----------
 public | cashier_view | view | mayank=arwdDxt/mayank+|                   |
        |              |      | cashier=r/mayank      |                   |
(1 row)
```

Adding a few users to each group

solution.sql U ×

dbms-lab > labwork > midterm_assignment > solution.sql > ...

```sql
CREATE USER aditya IN GROUP customer ENCRYPTED PASSWORD 'aditya';
▶ Run SQL
CREATE USER satyam IN GROUP customer ENCRYPTED PASSWORD 'satyam';
▶ Run SQL
CREATE USER neel IN GROUP customer ENCRYPTED PASSWORD 'neel';
▶ Run SQL
CREATE USER amish IN GROUP cashier ENCRYPTED PASSWORD 'amish';
▶ Run SQL
CREATE USER harsh IN GROUP cashier ENCRYPTED PASSWORD 'harsh';
▶ Run SQL
CREATE USER jerry IN GROUP cashier ENCRYPTED PASSWORD 'jerry';
```

```
dvdrental=# \i solution.sql
CREATE ROLE
CREATE ROLE
CREATE ROLE
CREATE ROLE
CREATE ROLE
CREATE ROLE
dvdrental=# \du
                                   List of roles
 Role name |                         Attributes                         | Member of
-----------+------------------------------------------------------------+------------
 aditya    |                                                            | {customer}
 amish     |                                                            | {cashier}
 cashier   | Cannot login                                               | {}
 customer  | Cannot login                                               | {}
 harsh     |                                                            | {cashier}
 jerry     |                                                            | {cashier}
 mayank    | Superuser                                                  | {}
 neel      |                                                            | {customer}
 postgres  | Superuser, Create role, Create DB, Replication, Bypass RLS | {}
 satyam    |                                                            | {customer}
```

Logging as different users and trying to access the information.

```
└─ psql -U satyam -d dvdrental
Password for user satyam:
psql (14.2 (Ubuntu 14.2-1.pgdg20.04+1))
Type "help" for help.

dvdrental=> select * from customer;
ERROR:  permission denied for table customer
dvdrental=> select title, category from customer_view limit 2;
      title       |  category
------------------+-------------
 Academy Dinosaur | Documentary
 Ace Goldfinger   | Horror
(2 rows)

dvdrental=>
```

```
└─ psql -U jerry -d dvdrental
Password for user jerry:
psql (14.2 (Ubuntu 14.2-1.pgdg20.04+1))
Type "help" for help.

dvdrental=> select * from payment;
ERROR:  permission denied for table payment
dvdrental=> select customer_name, amount from cashier_view limit 3;
 customer_name  | amount
----------------+--------
 Harold Martino |   5.99
 Harold Martino |   2.99
 Douglas Graf   |   4.99
(3 rows)

dvdrental=>
```

Q6. Create a function to calculate cashback offers on total payments done by each customer. The function should take the customer id as an argument and calculate their cashback on total payments they made. Top 10% of customers based on the total payment made will get a 2% cash back else 1% cash back. The function should return the customer's name and cashback amount.

solution.sql U ×

dbms-lab > labwork > midterm_assignment > solution.sql > ...

```sql
CREATE OR REPLACE FUNCTION cashback_offer(cid INT)
RETURNS TABLE(
    customer_name TEXT,
    cashback_amount NUMERIC
)
LANGUAGE PLPGSQL
AS $$
DECLARE
    top_10_cid SMALLINT;
```

▶ Run SQL

```sql
BEGIN

    -- Check if customer id is in top 10% of customers based on total payment (SELECT-INTO)
    SELECT customer_id
    INTO top_10_cid
    FROM (
            SELECT customer_id
            FROM payment
            GROUP BY payment.customer_id
            ORDER BY SUM(amount) DESC
            LIMIT 0.1 * (
                    SELECT COUNT(DISTINCT payment.customer_id)
                    FROM payment
                )
        ) AS top_10_precent_cusomters
    WHERE customer_id = cid;
```

solution.sql U ×

dbms-lab > labwork > midterm_assignment > solution.sql > ...

```sql
    -- If customer id is not in top 10% list, then returning 1% cashback
    ▶ Run SQL
    IF NOT FOUND THEN
        RETURN QUERY
            SELECT customer.first_name || ' ' || customer.last_name AS customer_name,
                0.01 * SUM(amount) AS cashback_amount
            FROM payment,
                customer
            WHERE payment.customer_id = customer.customer_id
            GROUP BY payment.customer_id,
                customer.first_name,
                customer.last_name
            HAVING payment.customer_id = cid;
```

solution.sql U ×

dbms-lab > labwork > midterm_assignment > solution.sql > ...

```sql
    ▶ Run SQL
    ELSE
        -- If customer id is in top 10% list, then returning 2% cashback
        RETURN QUERY
            SELECT customer.first_name || ' ' || customer.last_name AS customer_name,
                0.02 * SUM(amount) AS cashback_amount
            FROM payment,
                customer
            WHERE payment.customer_id = customer.customer_id
            GROUP BY payment.customer_id,
                customer.first_name,
                customer.last_name
            HAVING payment.customer_id = cid;
    ▶ Run SQL
    END IF;
▶ Run SQL
END;
▶ Run SQL
$$;
```

```
dvdrental=# \i solution.sql
CREATE FUNCTION
dvdrental=#
```

To get testing data for the function for both cases.

solution.sql U temp.sql U

dbms-lab > labwork > midterm_assignment > temp.sql > ...

Run on active connection | Select block

```sql
-- See top 10% customer list
```

Run SQL

```sql
SELECT customer_id,
    SUM(amount) AS amount
FROM payment
GROUP BY payment.customer_id
ORDER BY SUM(amount) DESC
LIMIT 0.1 * (
        SELECT COUNT(DISTINCT payment.customer_id)
        FROM payment
    );
```

```
dvdrental=# \i temp.sql
```

```
 347 | 142.70
 390 | 142.69
 267 | 142.67
 257 | 142.66
  39 | 141.71
  78 | 141.69
 362 | 140.69
 363 | 139.72
 368 | 139.69
 119 | 139.69
  66 | 139.67
 237 | 138.69
  29 | 138.65
  87 | 137.72
  80 | 137.70
 439 | 137.67
 558 | 135.72
 494 | 135.70
 576 | 135.68
 591 | 134.73
 467 | 134.72
 479 | 134.71
 210 | 134.70
 120 | 134.70
 204 | 134.69
(60 rows)

(END)
```

solution.sql U temp.sql U ×

dbms-lab > labwork > midterm_assignment > temp.sql > ...

Run on active connection | Select block

```sql
-- See all customer list
```

Run SQL

```sql
SELECT customer_id,
    SUM(amount) AS amount
FROM payment
GROUP BY payment.customer_id
ORDER BY SUM(amount) DESC;
```

```
dvdrental=# \i temp.sql
```

```
 313 |  63.84
 548 |  63.84
 350 |  63.79
 557 |  63.78
 492 |  62.85
 549 |  62.84
 401 |  62.81
 330 |  60.82
 136 |  59.86
 162 |  59.83
 159 |  58.83
  97 |  58.82
 252 |  58.80
  61 |  57.87
 124 |  57.86
 395 |  57.81
 271 |  56.84
 250 |  54.85
 288 |  52.81
 586 |  50.83
 110 |  49.88
 320 |  47.85
 248 |  37.87
 281 |  32.90
 318 |  27.93
(599 rows)

(END)
```

```
dvdrental=# SELECT * FROM cashback_offer(204);
  customer_name   | cashback_amount
------------------+-----------------
 Rosemary Schmidt |          2.6938
(1 row)

dvdrental=# SELECT * FROM cashback_offer(318);
 customer_name | cashback_amount
---------------+-----------------
 Brian Wyman   |          0.2793
(1 row)

dvdrental=# -- These are the last ids taken
dvdrental=# -- from both the above lists shown
```