Lab₃

User-defined functions:

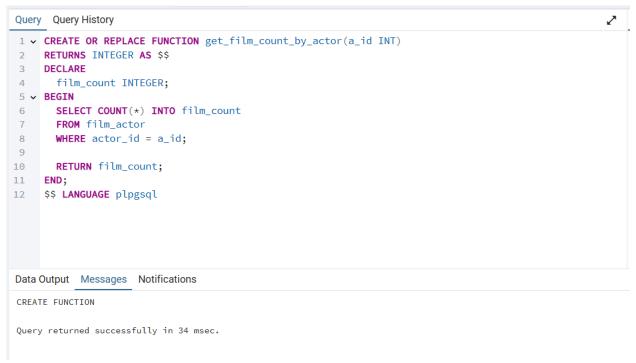
1. Create a function that takes category name and returns the number of films in this category.

```
∠
Query Query History
1 - CREATE OR REPLACE FUNCTION get_film_count_by_category(cat_name TEXT)
    RETURNS INTEGER AS $$
    DECLARE
     count_result INTEGER;
5 V BEGIN
     SELECT COUNT(*) INTO count_result
     FROM film
7
     JOIN film_category USING (film_id)
8
     JOIN category USING (category_id)
9
     WHERE category.name = cat_name;
10
11
12
     RETURN count_result;
   END;
13
    $$ LANGUAGE plpgsql
14
Data Output Messages Notifications
CREATE FUNCTION
Query returned successfully in 45 msec.
```

2. Create a function that takes customer_id and returns full name (first_name last_name)



3. Create a function that's takes actor_id and returns his/her film count.



4. Create a function that returns the most recent rental date for a given customer.



5. Function overload question

- a. Create two versions of a function get_film_info:
 - i. One takes film_id and returns title
 - ii. One takes title and returns film_id

```
Query Query History
1 \rightarrow CREATE OR REPLACE FUNCTION get_film_info(f_id INT)
    RETURNS TEXT AS $$
3 DECLARE
      film_title TEXT;
5 V BEGIN
      SELECT title INTO film_title
     FROM film
    WHERE film_id = f_id;
8
9
    RETURN film_title;
10
11 END;
12 $$ LANGUAGE plpgsql;
13
14 • CREATE OR REPLACE FUNCTION get_film_info(f_title TEXT)
15 RETURNS INT AS $$
16 DECLARE
     film_id_result INT;
17
18 V BEGIN
19
    SELECT film_id INTO film_id_result
20 FROM film
21 WHERE title = f_title;
22
23
   RETURN film_id_result;
24 END;
25 $$ LANGUAGE plpgsql;
Data Output Messages Notifications
CREATE FUNCTION
Query returned successfully in 32 msec.
```

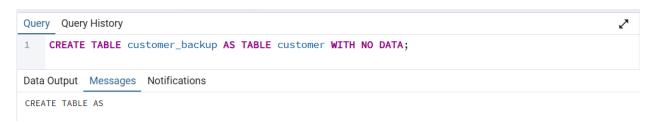
Trigger

6. Create a trigger to prevent setting a negative value for rental_rate column in the film table.

```
Query Query History
                                                                                        ~
1 • CREATE OR REPLACE FUNCTION prevent_negative_rental_rate()
2 RETURNS TRIGGER AS $$
3 BEGIN
4 IF NEW.rental_rate < 0 THEN
5
     RAISE EXCEPTION 'Rental rate cannot be negative';
6 END IF;
7 RETURN NEW;
8 END;
9 $$ LANGUAGE plpgsql;
10
11 • CREATE TRIGGER check_rental_rate
12 BEFORE INSERT OR UPDATE ON film
13 FOR EACH ROW
Data Output Messages Notifications
CREATE TRIGGER
Query returned successfully in 36 msec.
```

7. Create a trigger to save a backup for newly inserted customers. (create customer_backup table first).

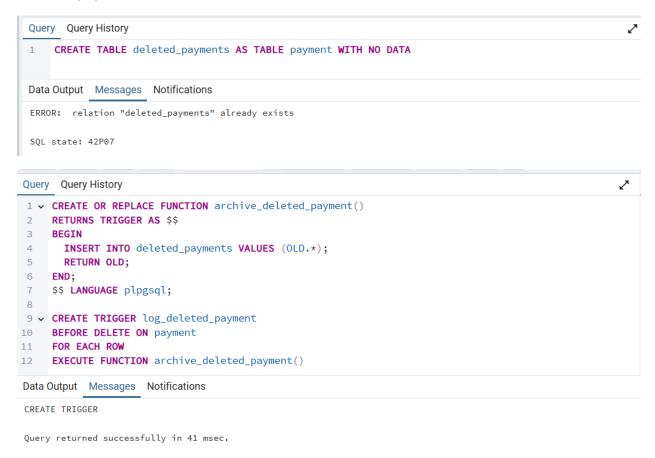
Creating customer_backup table first



Then trigger function

```
Query Query History
1 ➤ CREATE OR REPLACE FUNCTION backup_new_customer()
    RETURNS TRIGGER AS $$
3 BEGIN
4
    INSERT INTO customer_backup VALUES (NEW.*);
5 RETURN NEW;
6 END;
7 $$ LANGUAGE plpgsql;
8
9 ▼ CREATE TRIGGER backup_customer_insert
10 AFTER INSERT ON customer
11 FOR EACH ROW
12 EXECUTE FUNCTION backup_new_customer();
Data Output Messages Notifications
CREATE TRIGGER
Query returned successfully in 32 msec.
```

8. Create a trigger on payment table that copies deleted rows into a new table deleted_payments



9. Track when a film's title is updated, and log the old and new titles along with the time of the change in a new table (film_audit).

