

Ahmed Ramadan Elsayed

PostgreSQL – Lab3

Query Query History

```
61
62 create function get_category(cat_name text)
63 returns int
64 language plpgsql
65 as
66 $$
67 begin
68 return (
69 select count(*) from category c
70 inner join film_category fc
71 on c.category_id=fc.category_id
72 where c.name=cat_name
73 );
74 end;
75 $$;
76 select get_category('Comedy');
77
78
```

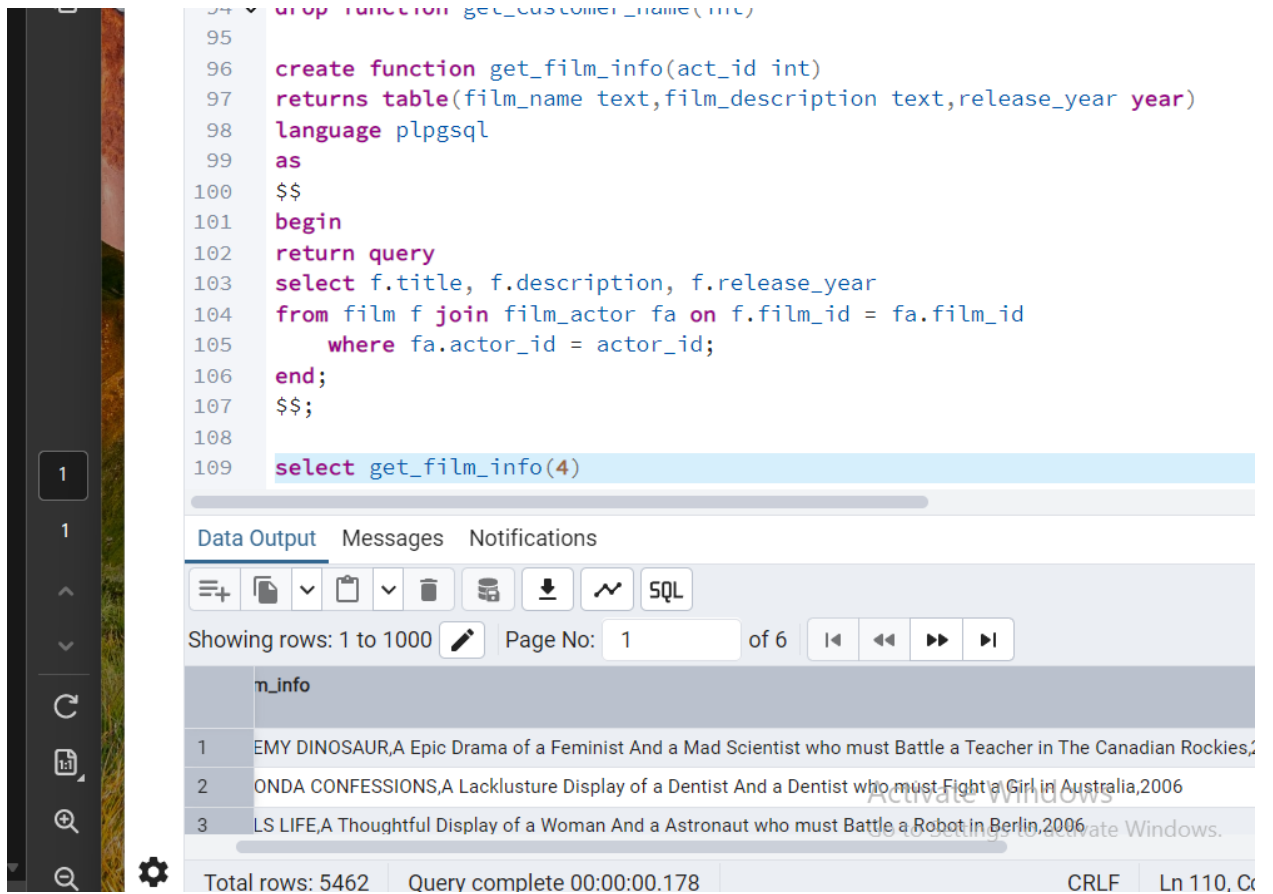
Data Output Messages Notifications

Showing rows: 1 to 1 Page No: 1 of 1

	get_category integer
1	58

1. Act

3.



```

94 drop function get_customer_name(int);
95
96 create function get_film_info(actor_id int)
97 returns table(film_name text, film_description text, release_year year)
98 language plpgsql
99 as
100 $$
101 begin
102 return query
103 select f.title, f.description, f.release_year
104 from film f join film_actor fa on f.film_id = fa.film_id
105 where fa.actor_id = actor_id;
106 end;
107 $$;
108
109 select get_film_info(4)

```

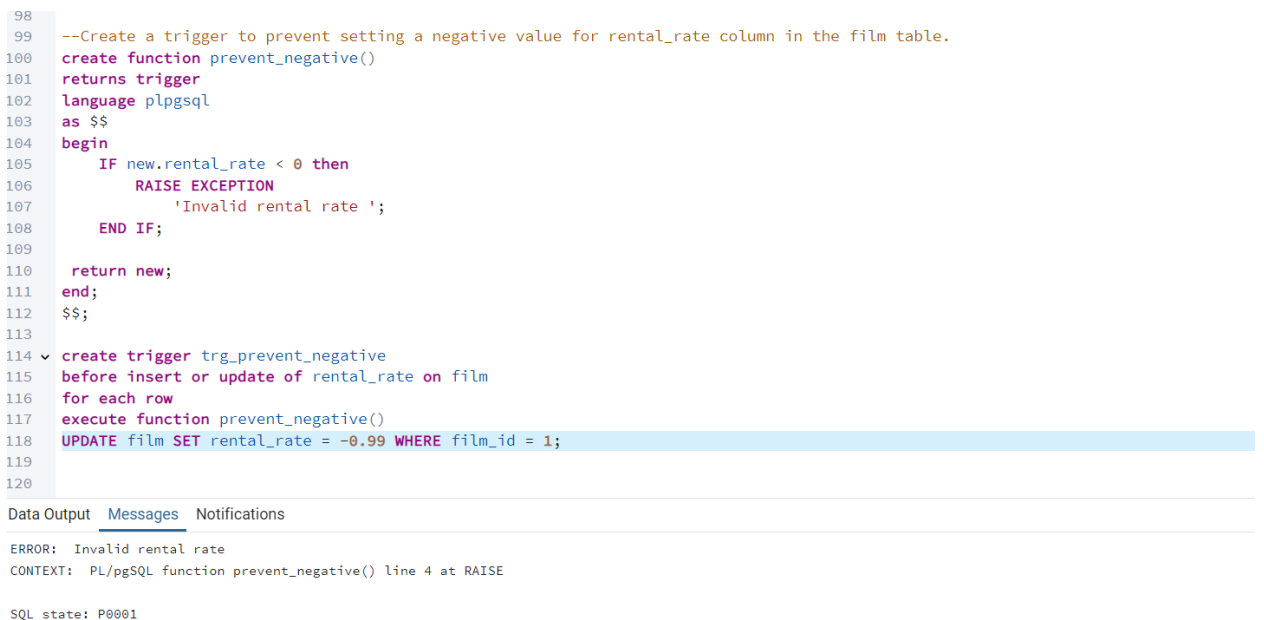
Data Output Messages Notifications

Showing rows: 1 to 1000 Page No: 1 of 6

	n_info
1	EMY DINOSAUR,A Epic Drama of a Feminist And a Mad Scientist who must Battle a Teacher in The Canadian Rockies,
2	ONDA CONFESSIONS,A Lacklusture Display of a Dentist And a Dentist who must Fight a Girl in Australia,2006
3	LS LIFE,A Thoughtful Display of a Woman And a Astronaut who must Battle a Robot in Berlin,2006

Total rows: 5462 Query complete 00:00:00.178 CRLF Ln 110, C

4.



```

98
99 --Create a trigger to prevent setting a negative value for rental_rate column in the film table.
100 create function prevent_negative()
101 returns trigger
102 language plpgsql
103 as $$
104 begin
105 IF new.rental_rate < 0 then
106 RAISE EXCEPTION
107 'Invalid rental rate ';
108 END IF;
109
110 return new;
111 end;
112 $$;
113
114 create trigger trg_prevent_negative
115 before insert or update of rental_rate on film
116 for each row
117 execute function prevent_negative()
118 UPDATE film SET rental_rate = -0.99 WHERE film_id = 1;
119
120

```

Data Output Messages Notifications

ERROR: Invalid rental rate
CONTEXT: PL/pgSQL function prevent_negative() line 4 at RAISE

SQL state: P0001

```

create table customer_backup(backup_id serial,customer_id int,name text,email text,address_id int)

create function make_customer_backup()
returns trigger
language plpgsql
as $$
begin
insert into customer_backup(customer_id,name,email,address_id)
values(new.customer_id,new.first_name || ' ' || new.last_name,
new.email,new.address_id);
return new;
end;
$$;

create trigger backup_trigger
after insert on customer
for each row
execute function make_customer_backup();
select * from customer_backup

```

ata Output Messages Notifications

backup_id	customer_id	name	email	address_id
integer	integer	text	text	integer
1	90000	ahmed mohamed	ahmed@gmail.com	3

5.

```

--Create a trigger on payment table that copies deleted rows into a new table deleted_payments
select * from payment order by payment_id desc;
create table deleted_payments(payment_id int, customer_id int,staff_id int,rental_id int, amount numeric,payment_date timestamp)
create function make_copy_deleted_payments()
returns trigger
language plpgsql
as $$
begin
insert into deleted_payments(payment_id , customer_id , staff_id , rental_id , amount , payment_date )
values(old.payment_id,old.customer_id ,old.staff_id ,old.rental_id , old.amount ,old.payment_date);
return old;
end;
$$;
create trigger delete_payments_trigger
after delete on payment
for each row
execute function make_copy_deleted_payments();
delete from payment where payment_id=32097
select * from deleted_payments

```

6.

```

7
15 --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).
16 select * from film
17 create table film_audit(oid_title text,new_title text,change_time timestamp);
18 v create function changed_film_name()
19 returns trigger
20 language plpgsql
21 as $$
22 begin
23     IF OLD.title <> NEW.title THEN
24         insert into film_audit
25         values(oid.title,new.title,now());
26     END IF;
27
28     return new;
29 end;
30 $$;
31
32 v create trigger changed_film_name
33 after update on film
34 for each row
35 execute function changed_film_name()
36 UPDATE film SET title = 'NEW TITLE' WHERE film_id = 1;
37 SELECT * FROM film_audit

```

7.