**SQL ASSIGNMENT**

Chapter1:

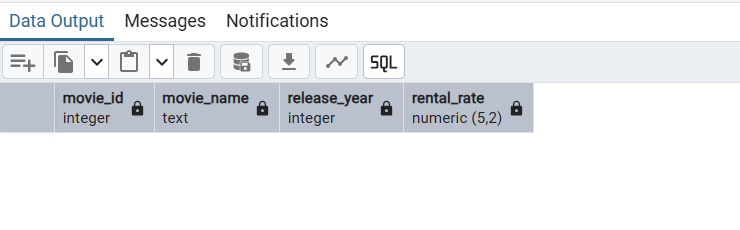
1. Write a query to create a simple table Movies including columns movie\_id , movie\_name, release\_year, rental\_rate.

Query:

Create Table movies (movie\_id Integer, movie\_name Text, release\_year Integer, rental\_rate Numeric (5,2));

Select \* From movies;

Output:



1. Write a query to insert 5 records with your own value into the table Movies against each

column. The release\_year must be 2016, 2017, 2022, 2025, 2023.

Query:

Insert into movies (movie\_id, movie\_name, release\_year, rental\_rate)

Values (1, 'interstellar',2016, 500),

(2,'lion king', 2017, 700),

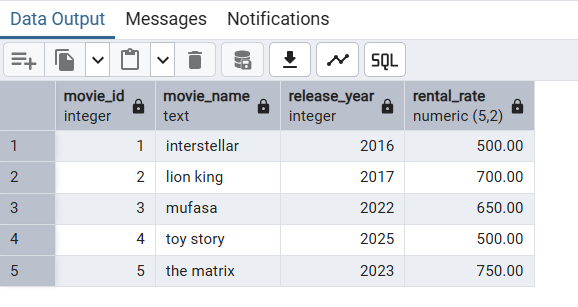
(3,'mufasa', 2022, 650),

(4,'toy story', 2025, 500),

(5,'the matrix', 2023, 750);

Select \* From movies;

Output:



1. Write a query to insert rows from film table to Movies table.

Query:

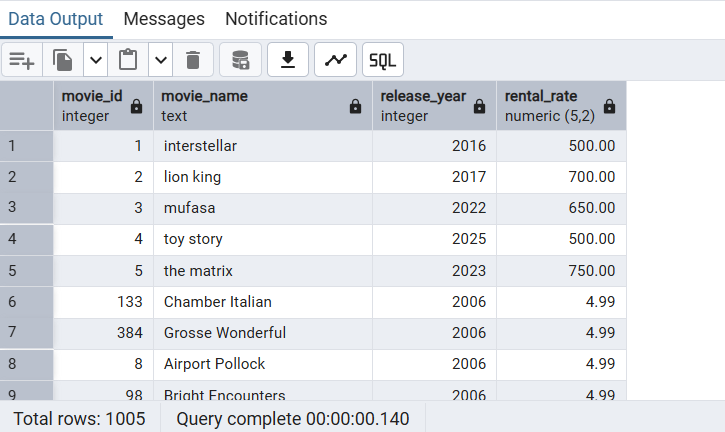
Insert into movies (movie\_id, movie\_name, release\_year, rental\_rate)

Select film\_id, title, release\_year, rental\_rate

from film;

Select \* from movies;

Output:



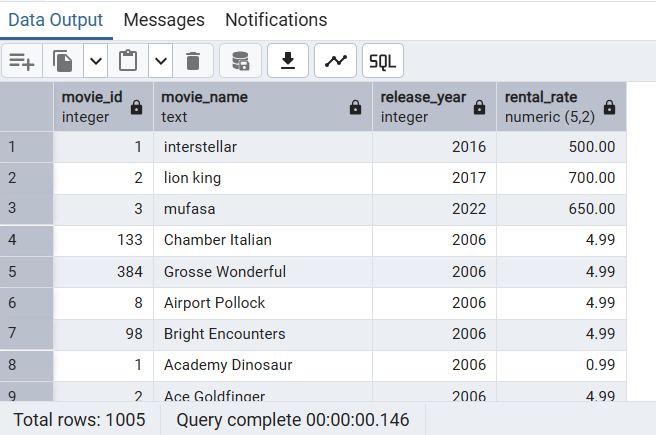
1. Write a query to update the rental\_rate to 999 for those Movies whose release\_year is greater than 2022.

Query:

Update movies Set rental\_rate='999' where release\_year>2022;

Select \* from movies;

Output:



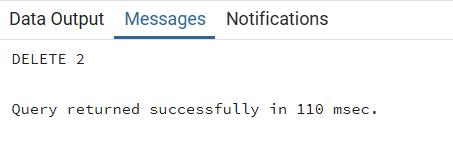
1. Delete the rows from the Movies table where the rental\_rate is 999.

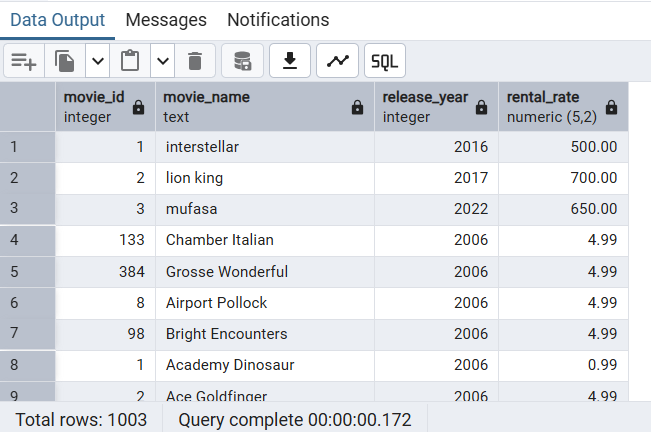
Query:

Delete from movies where rental\_rate = 999;

Select \* from movies;

Output:



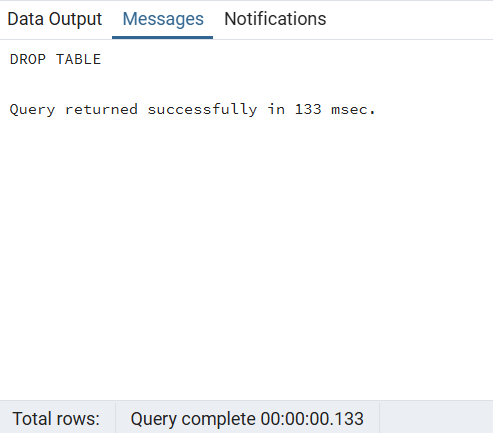


1. Drop the table Movies.

Query:

Drop table movies;

Output:



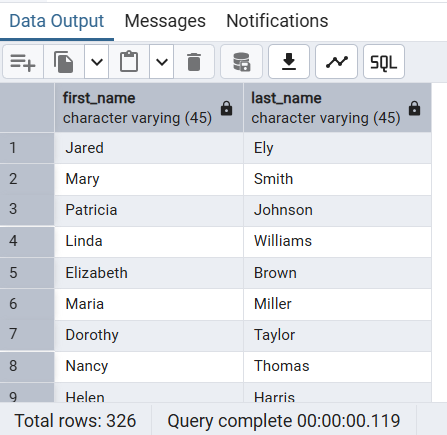
Chapter 2

1. Display all the names of the customers with store\_id = 1.

Query:

Select first\_name, last\_name from customer where store\_id = 1;

Output:

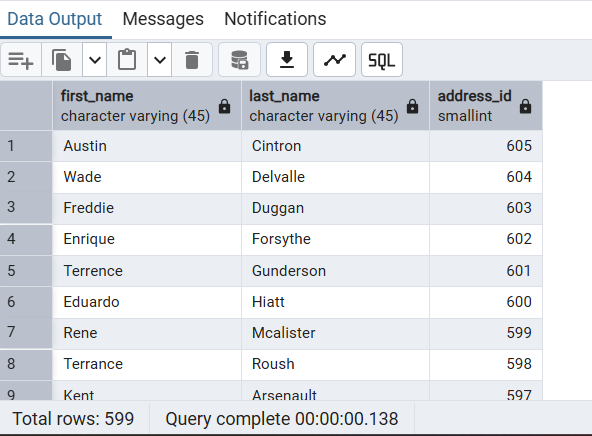


1. Display all the customer names ordered in descending order of their address.

Query:

Select first\_name, last\_name, address\_id from customer order by address\_id desc;

Output:

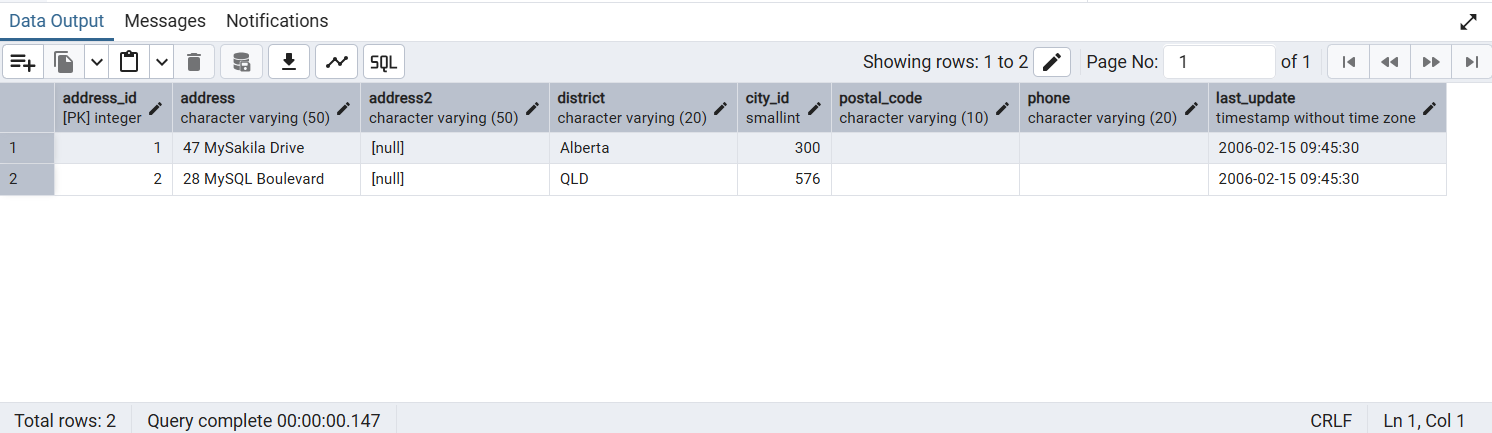


1. Display the addresses whose phone number is empty.

Query:

Select \* from address where phone = '';

Output:

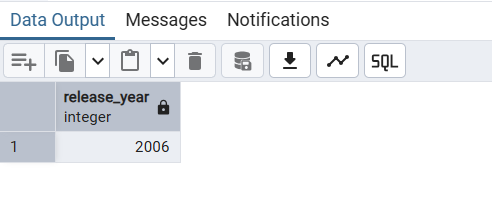


1. Display all the distinct years of the movie released from the films.

Query:

Select distinct release\_year from film;

Output:



5. Display the names of the customers whose first names start with A and last name

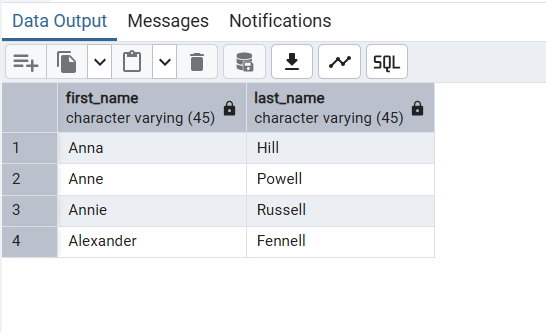
ends with ‘l’.

Query:

Select first\_name, last\_name from customer

where first\_name like 'A%' and last\_name like '%l';

Output:

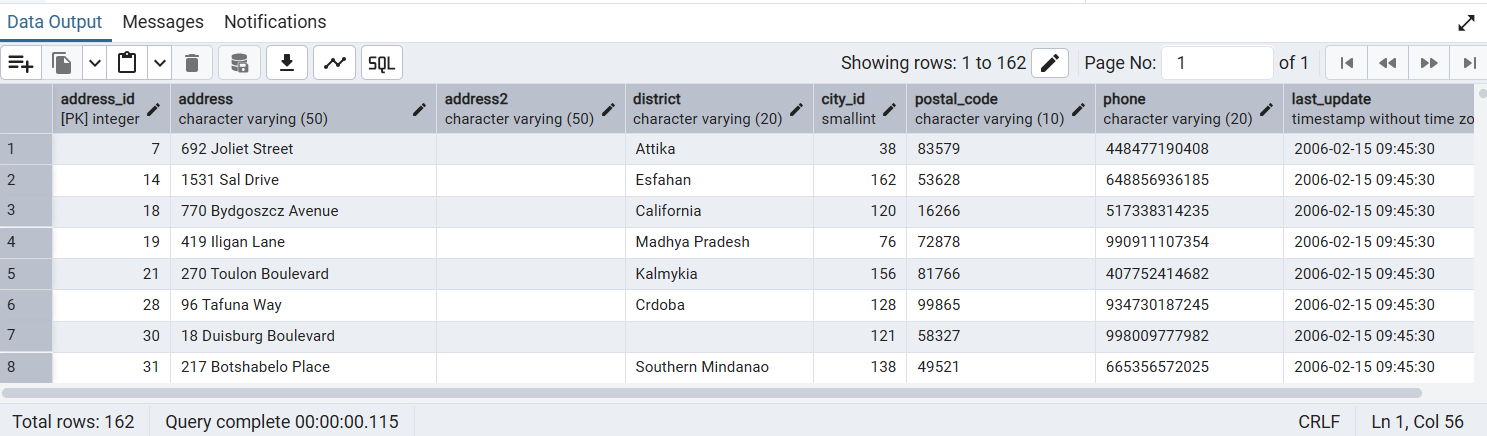


6. Display the address whose city id is in between 30 and 190.

Query:

Select \* from address where city\_id between 30 and 190;

Output:



7. Display the address whose city id is in between 30 and 190 and whose district is

Texas.

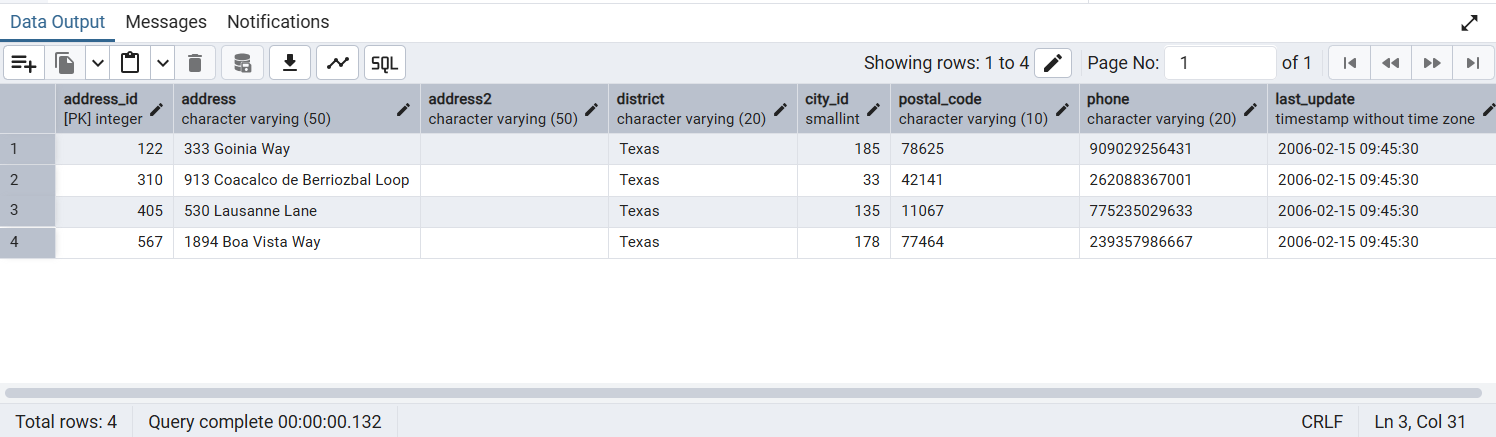
Query:

Select \* from address

where city\_id between 30 and 190

and lower(district) = 'texas';

Output:

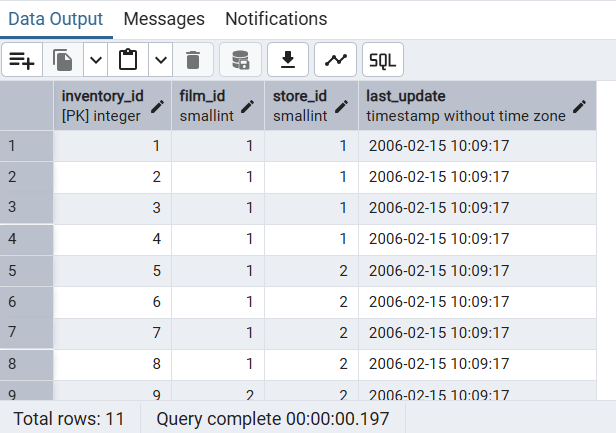


8. Display all the inventories whose film id is 1 or 2.

Query:

Select \* from inventory where film\_id = 1 or film\_id = 2;

Output:



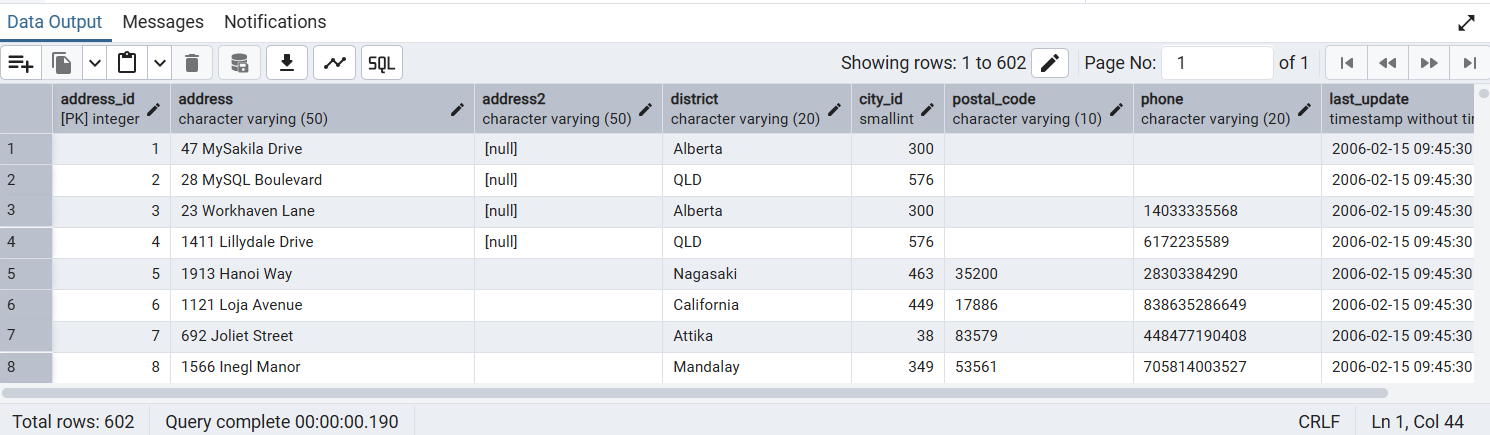
9. Display all the records whose city\_id is not 200.

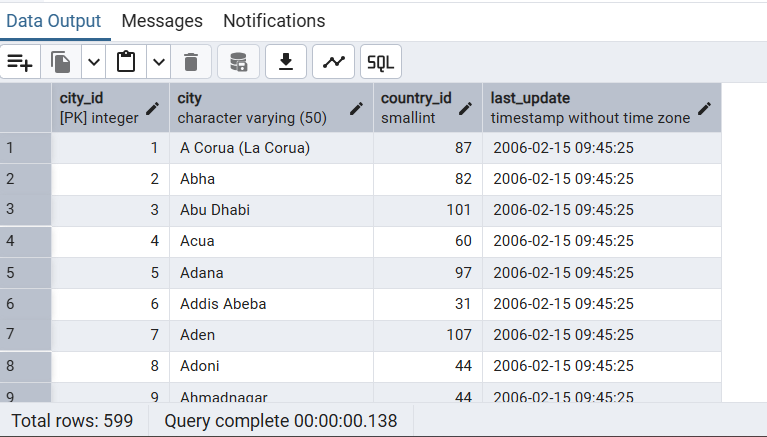
Query:

Select \* from address where city\_id != 200;

Select \* from city where city\_id != 200;

Output:





10. Write a query to create a table Movies and set NOT NULL and PRIMARY KEY

constraints for movie\_name and movie\_id .

Query:

Create table movies

(movie\_id Integer not null primary key ,

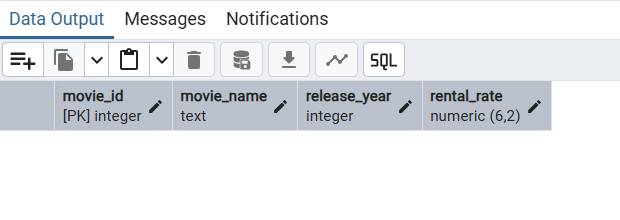
movie\_name Text not null ,

release\_year Integer,

rental\_rate Numeric(6,2));

Select \* from movies;

Output:



Chapter 3

1. What is the movie(s) that was rented the most.

Query: (Joined rental, inventory and film tables)

SELECT f.film\_id, f.title, COUNT(\*) AS rental\_count

FROM rental r

JOIN inventory i ON r.inventory\_id = i.inventory\_id

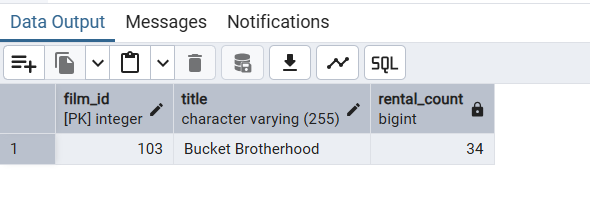
JOIN film f ON i.film\_id = f.film\_id

GROUP BY f.film\_id, f.title

ORDER BY rental\_count DESC

LIMIT 1;

Output:



1. Display each movie and the number of times it got rented.

Query:

Select f.film\_id, f.title, Count(\*) As rental\_count

From rental r

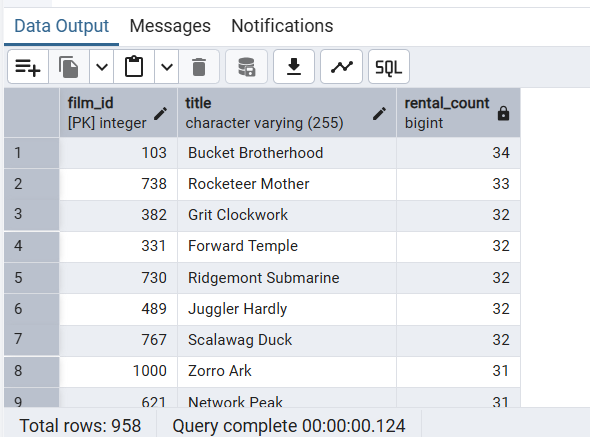
Join inventory i On r.inventory\_id = i.inventory\_id

Join film f On i.film\_id = f.film\_id

Group by f.film\_id, f.title

Order by rental\_count Desc;

Output:



1. Show the number of movies each actor acted in.

Query: (Joined actor, film\_actor and film tables)

Select a.actor\_id, a.first\_name , a.last\_name ,

count(\*)as film\_count

from actor a

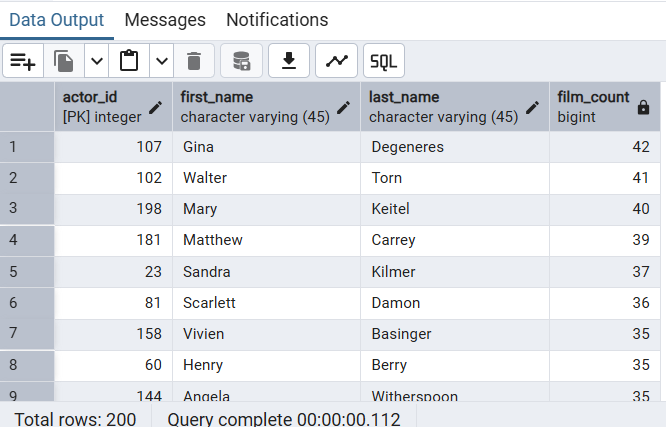
join film\_actor fa on a.actor\_id = fa.actor\_id

join film f on fa.film\_id = f.film\_id

group by a.actor\_id, a.first\_name , a.last\_name

order by film\_count Desc;

Output:



1. Display the names of the actors that acted in more than 20 movies.

Query:

Select a.actor\_id, a.first\_name , a.last\_name ,

count(\*)as film\_count

from actor a

join film\_actor fa on a.actor\_id = fa.actor\_id

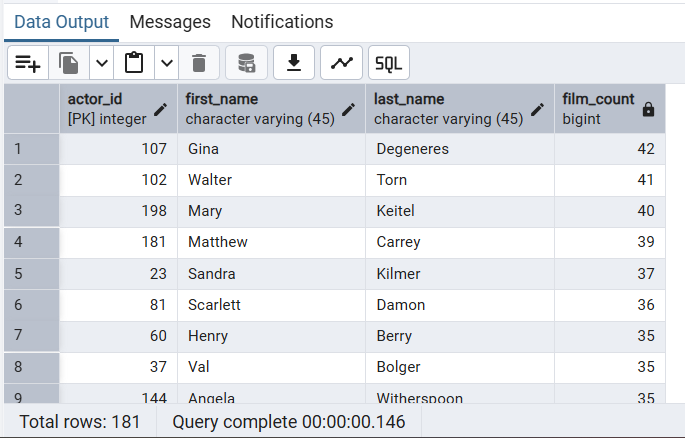
join film f on fa.film\_id = f.film\_id

group by a.actor\_id, a.first\_name , a.last\_name

Having count(\*) > 20

order by film\_count desc;

Output:



1. For each store, display the number of customers that are members of that
2. store.

Query: (Joined store and customer tables)

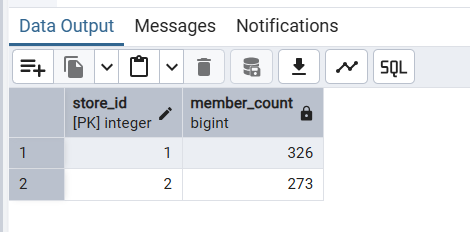
Select s.store\_id , count(c.customer\_id) as member\_count

from store s

join customer c on s.store\_id = c.store\_id

group by s.store\_id;

Output:



1. What is the highest total\_payment done.

Query:

Select customer\_id,sum(amount)as Total\_Highest\_Payment

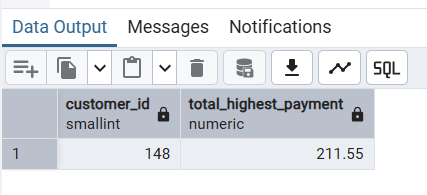
from payment

group by customer\_id

order by Total\_Highest\_payment desc

Limit 1;

Output:



1. What is the name of the customer who made the highest total payments.

Query:

Select c.customer\_id, c.first\_name, c.last\_name, sum(p.amount)as Total\_Highest\_Payment

from payment p

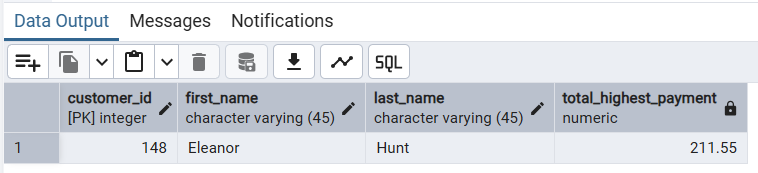
join customer c on p.customer\_id = c.customer\_id

group by c.customer\_id, c.first\_name, c.last\_name

order by Total\_Highest\_payment desc

Limit 1;

Output:



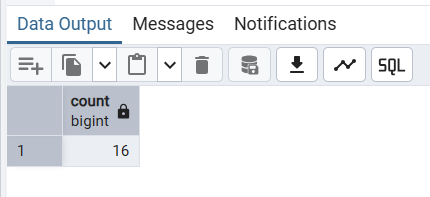
1. How many actors have 8 letters only in their first\_names.

Query:

Select count(\*) from actor

where length(first\_name)=8;

Output:



1. Display the movies offered for rent in store\_id 1 and not offered in
2. store\_id 2.

Query: (Joined film and inventory tables)

Select f.film\_id, f.title, i.store\_id

from film f

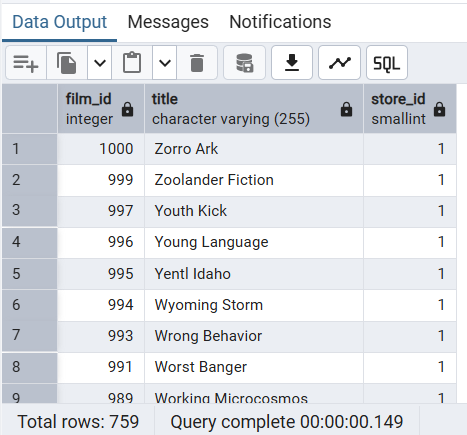
join inventory i on f.film\_id = i.film\_id

group by f.film\_id, f.title, i.store\_id

having i.store\_id = 1

order by f.film\_id desc;

Output:



1. Display the movie title for the most rented movie in the store with
2. store\_id 1.

Query: ( Joined rental, film and inventory table)

Select f.film\_id, f.title, i.store\_id, Count(\*) As rental\_count

From rental r

Join inventory i On r.inventory\_id = i.inventory\_id

Join film f On i.film\_id = f.film\_id

Group by f.film\_id, f.title ,i.store\_id

Having i.store\_id = 1

Order by rental\_count Desc

Limit 1;

Output:



Chapter 4

1. Find the names of the customers had bought DVD for rent for more than 5

days?

Query:

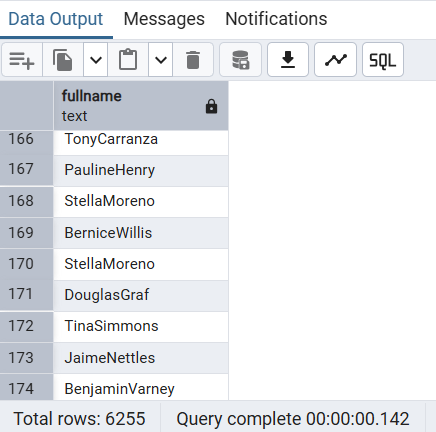
Select Concat(c.first\_name,'',c.last\_name)as Fullname

from Customer c

Join rental r on c.customer\_id = r.customer\_id

where Extract (Day From r.return\_date - r.rental\_date) > 5;

Output:



2. Find the city with maximum number of Staff ?

Query:

Select c.city\_id, c.city, count(s.staff\_id)as StaffCount

from city c

join address a on c.city\_id = a.city\_id

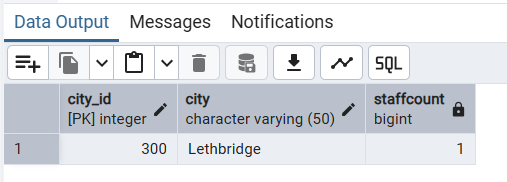
join staff s on s.address\_id = a.address\_id

group by c.city\_id, c.city

order by StaffCount desc

Limit 1;

Output:



3. Find the Staff Names in a city "Barcelona"?

Query:

Select s.staff\_id, concat(s.first\_name,' ',s.last\_name),c.city

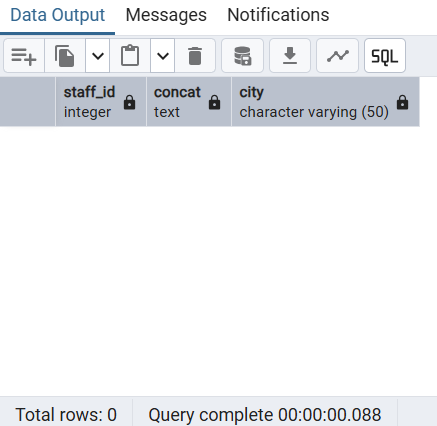
from staff s

join address a on s.address\_id = a.address\_id

join city c on c.city\_id = a.city\_id

where c.city = 'Barcelona';

Output: The city Barcelona has no staff.



4. List all the stores with their address.

Query:

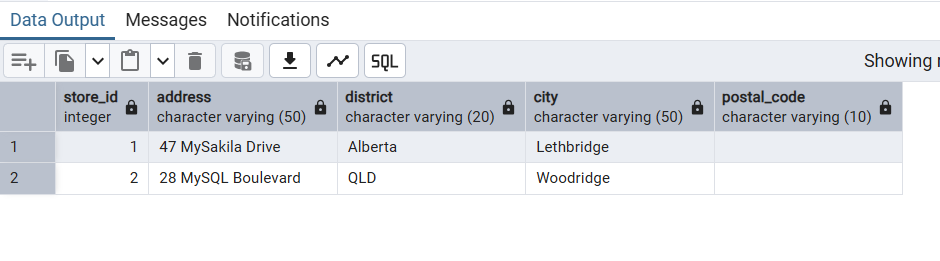
Select s.store\_id, a.address, a.district, c.city, a.postal\_code

from city c

join address a on c.city\_id = a.city\_id

join store s on a.address\_id = s.address\_id;

Output:



5. Find the films which were not rented ?

Query:

select f.film\_id, f.title

from film f

where f.film\_id not in (

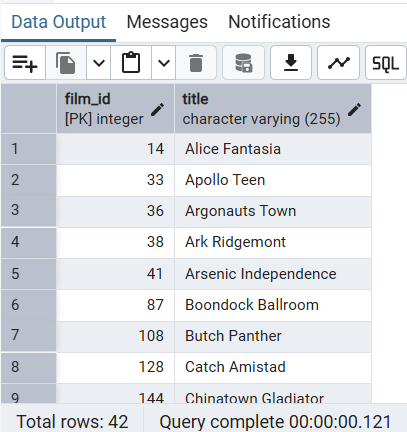
select i.film\_id

from inventory i

join rental r on r.inventory\_id = i.inventory\_id

);

Output:



6. Find the film which has maximum number of inventory ?

Query:

select f.film\_id, f.title, count(i.inventory\_id)as inventoryCount

from film f

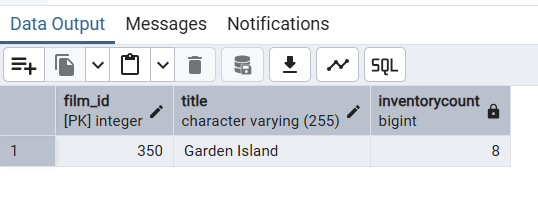
join inventory i on f.film\_id = i.film\_id

group by f.film\_id,f.title

order by inventoryCount desc

limit 1;

Output:



7. Find the name of the store which has maximum inventory ?

Query: (Joined 4 tables here)

Select s.store\_id,concat(a.address ,' ', a.district,' ',c.city)as StoreLocation,

count(i.inventory\_id)as inventoryCount

from store s

join inventory i on s.store\_id = i.store\_id

join address a on s.address\_id = a.address\_id

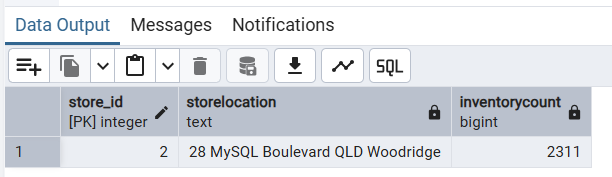
join city c on c.city\_id = a.city\_id

group by s.store\_id, concat(a.address ,' ', a.district,' ',c.city)

order by inventoryCount desc

Limit 1;

Output:



8. Find the actors who have not acted in a film ?

Query:

select actor\_id, concat(first\_name,' ',last\_name)as ActorName

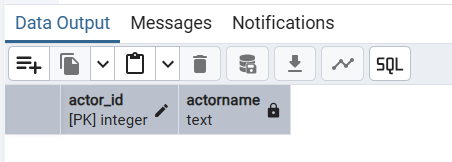
from actor

where actor\_id not in (

select actor\_id from film\_actor

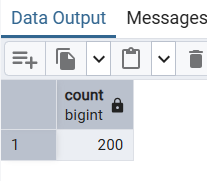
);

Output:

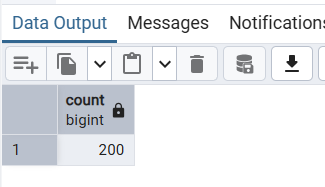


It means all the actors have acted in alteast one film. To verify that I have verified the actor’s count in both the tables.

Select count(distinct(actor\_id)) from actor;



Select count(distinct(actor\_id)) from film\_actor;



9. Show the number of rented movies under each rating.

Query:

Select f.rating , Count(r.rental\_id)as rentalCount

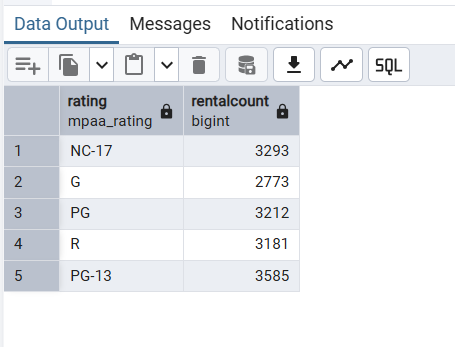
from rental r

join inventory i on r.inventory\_id = i.inventory\_id

join film f on f.film\_id = i.film\_id

group by f.rating;

Output:



Chapter 5:

1. Create view on table film on columns film\_id and title.

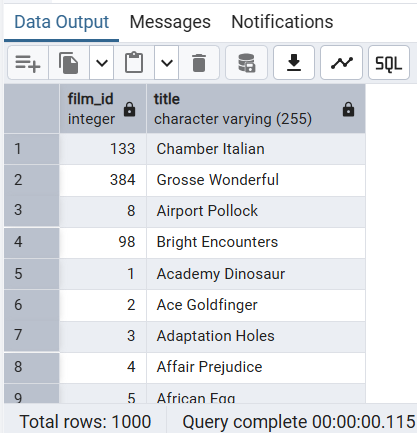
Query:

Create view film\_view as

select film\_id , title from film;

Select \* from film\_view;

Output:



2. Create a view to locate the rental\_rate is 4.99.

Query:

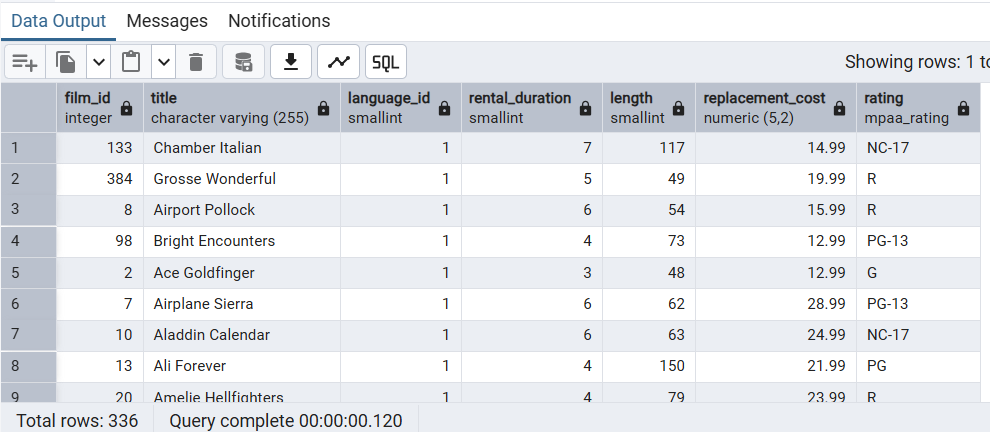
Create view rental\_rate\_view as

Select film\_id, title, language\_id,rental\_duration,length, replacement\_cost,rating

from film where rental\_rate = 4.99;

Select \* from rental\_rate\_view;

Output:

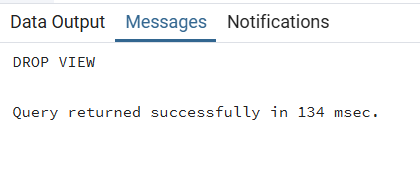


3. Drop the view for the table film.

Query:

Drop view if exists film\_view;

Output:



Chapter 6:

1. Create index on ‘film’ table.

Query:

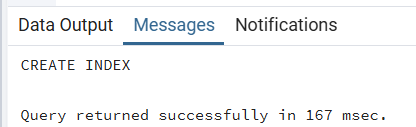
Create index film\_index on film(title);

To list the Index:

Select indexname , indexdef from pg\_indexes

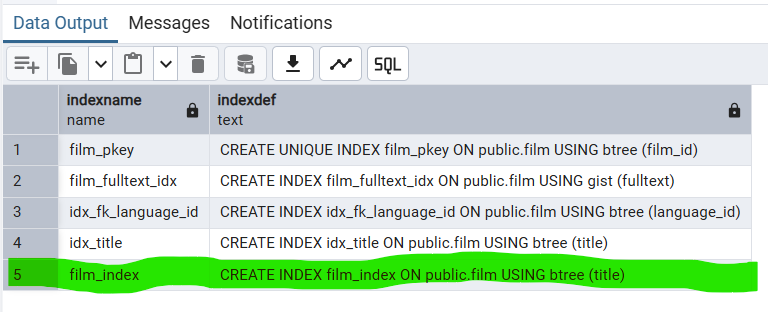
Where tablename = 'film';

Output 1:



Output 2: Listing the index

Latest film\_index that I have created.



2. Create index on the on the ‘customer’ table using the first\_name and the

last\_name.

Query:

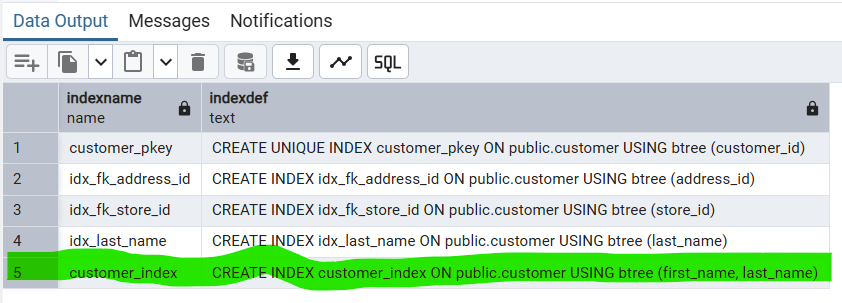
Create index customer\_index on customer(first\_name ,last\_name);

Select indexname , indexdef

from pg\_indexes

Where tablename = 'customer';

Output:



3. Write a Query to drop the indexes.

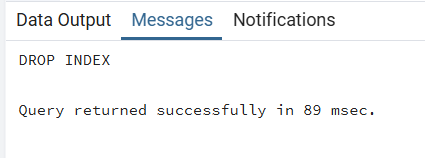
Query:

Drop index film\_index;

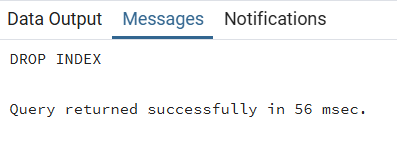
Drop index customer\_index;

Output:

Dropping film\_index



Dropping customer index



Chapter 7:

1. Create a trigger function while performing insert on the ‘film’ table.

Query:

-- Crating an audit table--

Create Table film\_insert\_log (

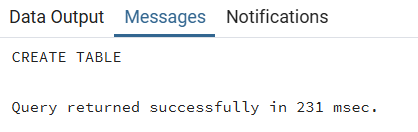
film\_id int,

title text,

inserted\_at timestamp Default Current\_Timestamp

);

Output:



Query:

--Creating Trigger Function--

Create or Replace Function log\_film\_insertion()

Returns Trigger as $$

Begin

Insert into film\_insert\_log(film\_id, title)

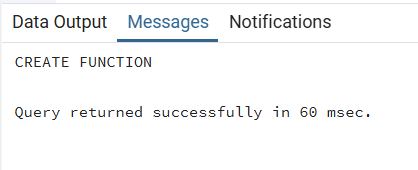
Values (NEW.film\_id, NEW.title);

Return New;

End;

$$ Language plpgsql;

Output:



Query:

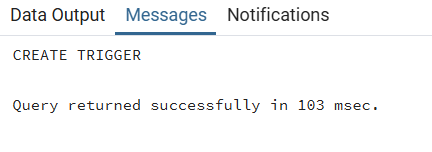
Create Trigger trg\_log\_film\_insert

After Insert On film

For Each Row

Execute Function log\_film\_insertion();

Output:



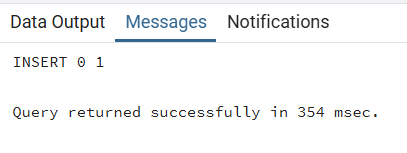
Query:

--Inserting New row—

INSERT INTO film (title, description, release\_year, language\_id, rental\_duration, rental\_rate, replacement\_cost, rating)

VALUES ('Modern World', 'Fancy', 2025, 1, 5, 4.99, 19.99, 'PG');

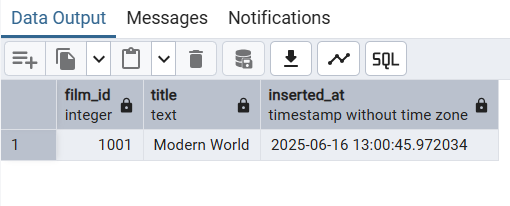
Output:



Query: Checking the audit table

Select \* From film\_insert\_log;

Output:

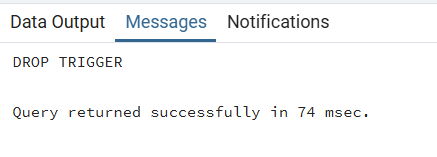


1. Delete the trigger.

Query:

Drop Trigger trg\_log\_film\_insert on film;

Output:



Chapter 8:

1. Create a table. Write a stored procedure to insert data in to the created table.

Query: Table Creation

Create table workers(

Id serial Primary key,

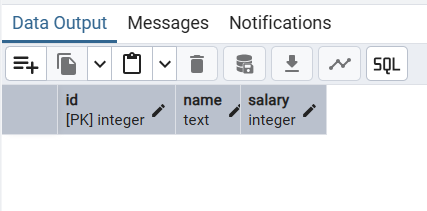
Name text,

Salary integer

);

Select \* from workers;

Output:



Query: Creating Procedure

Create or Replace Procedure insert\_worker\_val(

P\_name text,

P\_salary integer

)

Language plpgsql as $$

Begin

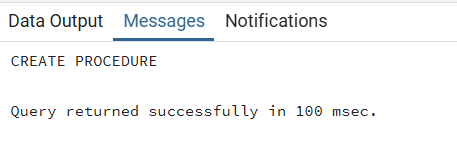
Insert into workers(Name , Salary)

values(P\_name, P\_salary);

End;

$$

Output:



Query: Calling the Procedure with Parameters

Call insert\_worker\_val('Suganya', 50000);

Call insert\_worker\_val('Mala',40000);

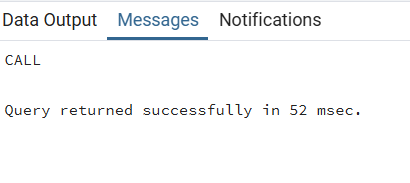
Call insert\_worker\_val('Nila',45000);

Call insert\_worker\_val('Mugil',43000);

Call insert\_worker\_val('Malar',30000);

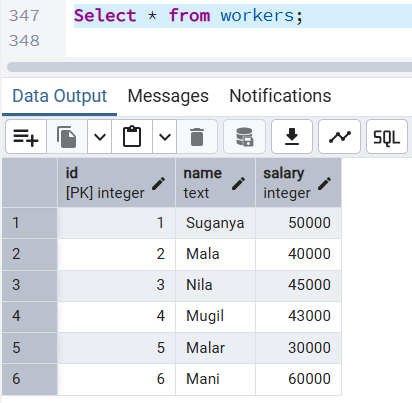
Call insert\_worker\_val('Mani',60000);

Output:



Final table view:

Select \* from workers;



2. Write a stored procedure to select the customers who rented from store\_id 2.

Query:

Create or Replace Procedure get\_customers\_from\_store\_2()

Language plpgsql

As $$

Begin

-- Select customers who rented from store\_id 2

Raise Notice 'Customer List from Store 2:';

Perform

c.customer\_id, c.first\_name, c.last\_name

From customer c

Join rental r On c.customer\_id = r.customer\_id

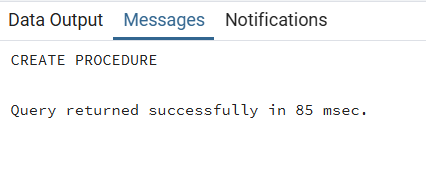
Join inventory i On r.inventory\_id = i.inventory\_id

Where i.store\_id = 2;

End;

$$;

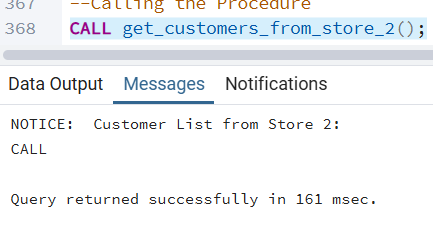
Output:



Query: To call the procedure to view Customer details

CALL get\_customers\_from\_store\_2();

Output:



In Stored Procedure we cannot return the results, we can perform only the particular action and raise notice. If we would like to view or display the output we need to use Functions instead.

Query: Same query in Functions

Create or Replace Function get\_customers\_from\_store(p\_store\_id INT)

Returns Table (customer\_id INT, first\_name TEXT, last\_name TEXT)

Language sql

As $$

Select Distinct c.customer\_id, c.first\_name, c.last\_name

From customer c

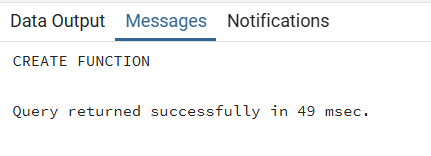
Join rental r On c.customer\_id = r.customer\_id

Join inventory i On r.inventory\_id = i.inventory\_id

Where i.store\_id = p\_store\_id;

$$;

Output:



Query: Calling the Function

Select \* From get\_customers\_from\_store(2);

Output:

