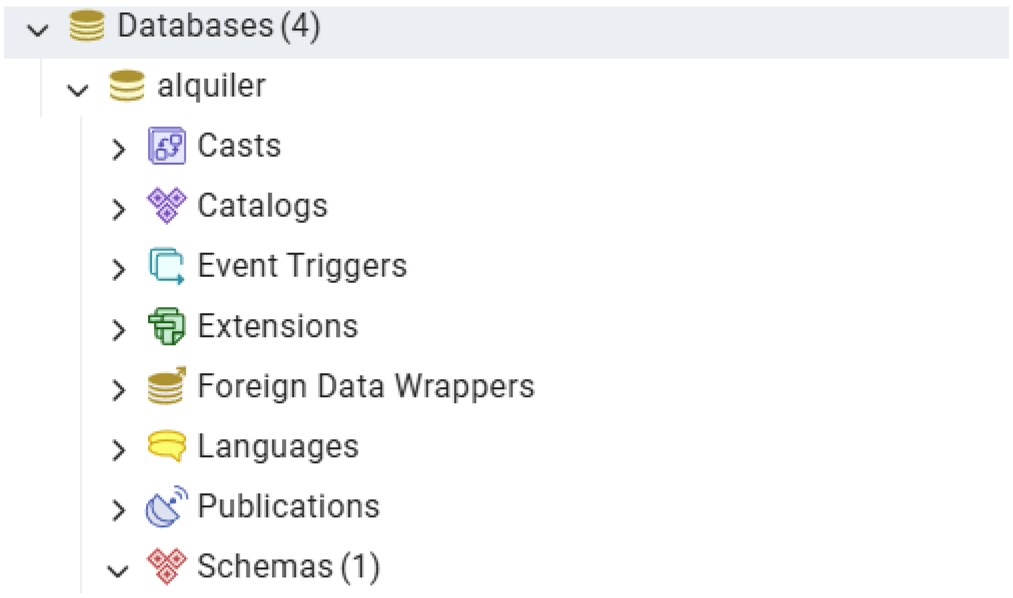
Práctica N°5

Juan David Duran - Angel Andres Boada

1. Restauración de la Base de Datos

Se hizo utilización del ".tar" que se nos Brindó, en nuestro caso utilizamos el pgAdmin4 para hacer un restore y así evitarnos problemas con los permisos al tener todo instalado en la máquina local.

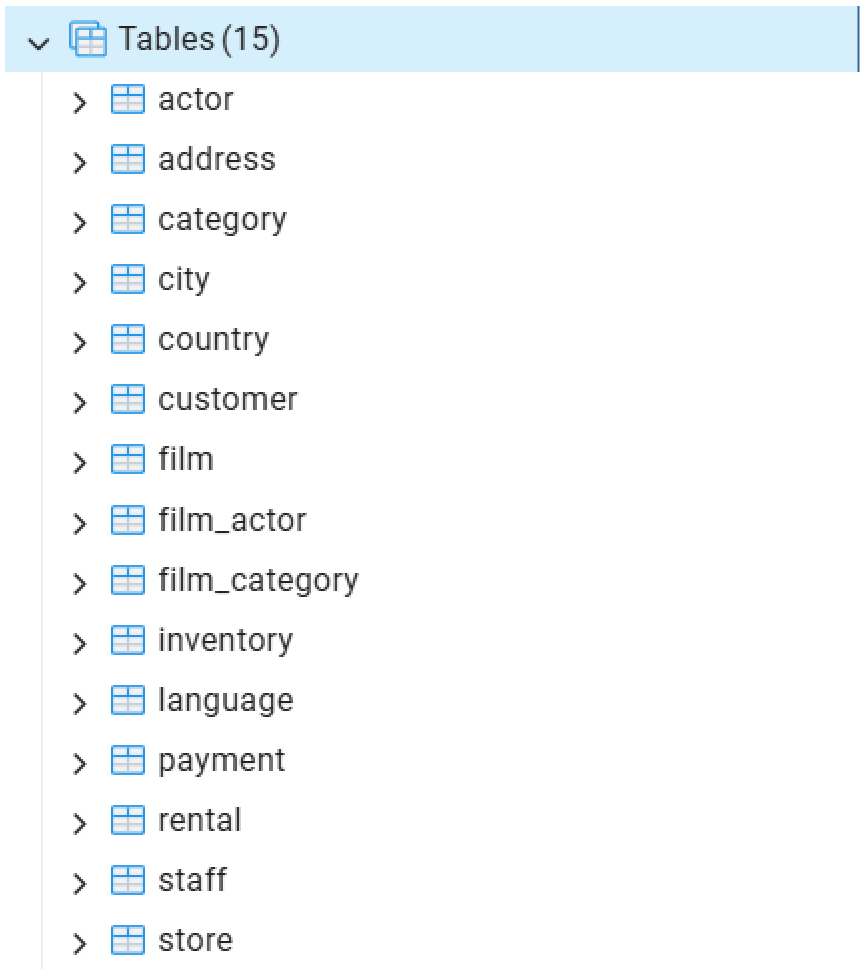


2. Reconocimiento de Tablas

Para obtener la estructura general de la base de datos se emplearon los siguientes comandos dentro de PostgreSQL, además de visualizarlo en el Panel.

- \dt -- Muestra las tablas
- \dv -- Muestra las vistas
- \ds -- Muestra las secuencias

Teniendo en cuenta como las más Importantes las Tablas: film, actor, category, rental, payment, inventory, store, staff y customer.

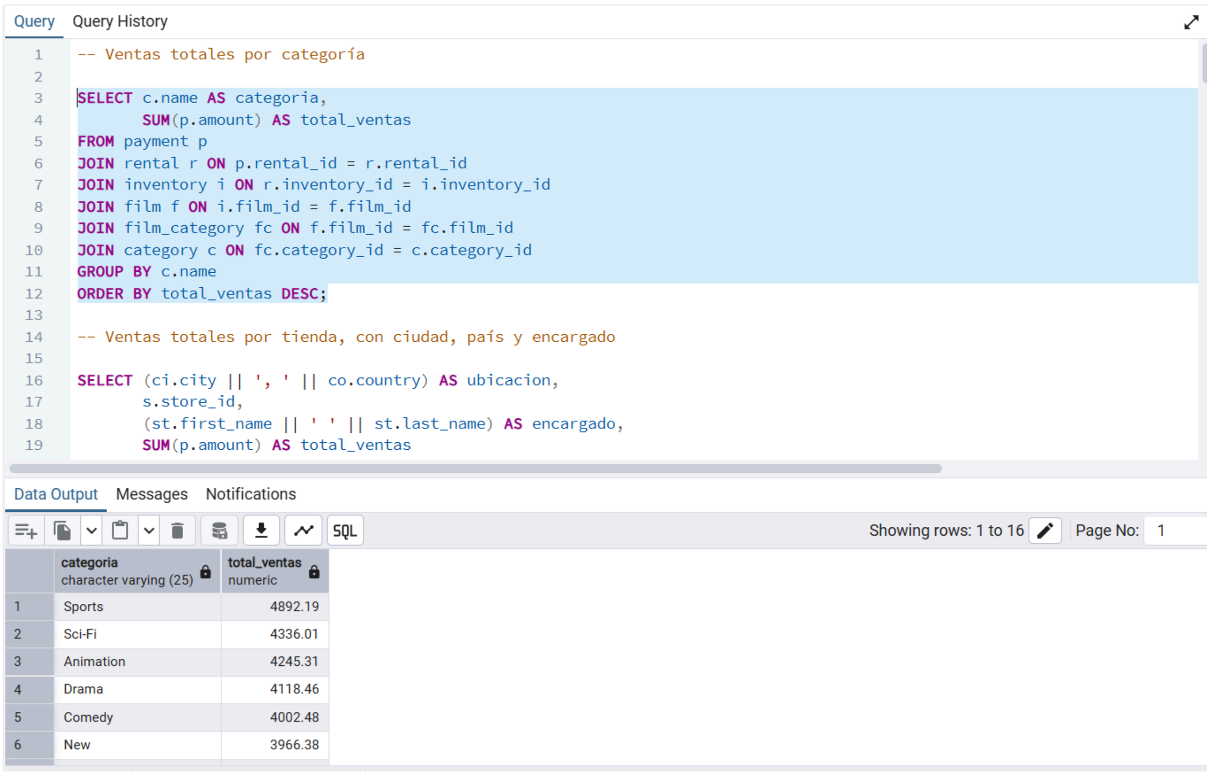

Tables (15)
actor
address
category
city
country
customer
film
film_actor
film_category
inventory
language
payment
rental
staff
store

3. Elementos Claves

Tabla	Descripción	Clave Primaria	Campos Relevantes
film	Películas disponibles para alquiler	film_id	title, description, rental_rate, length
actor	Actores participantes en películas	actor_id	first_name, last_name
category	Categorías de películas	category_id	name
rental	Registro de alquileres realizados	rental_id	rental_date, return_date
payment	Pagos efectuados por los clientes	payment_id	amount, payment_date
store	Tiendas de la empresa	store_id	address_id, manager_staff_id

staff	Empleados de las tiendas	staff_id	first_name, last_name
customer	Clientes registrados	customer_id	store_id, active, last_update

4. Consultas SQL



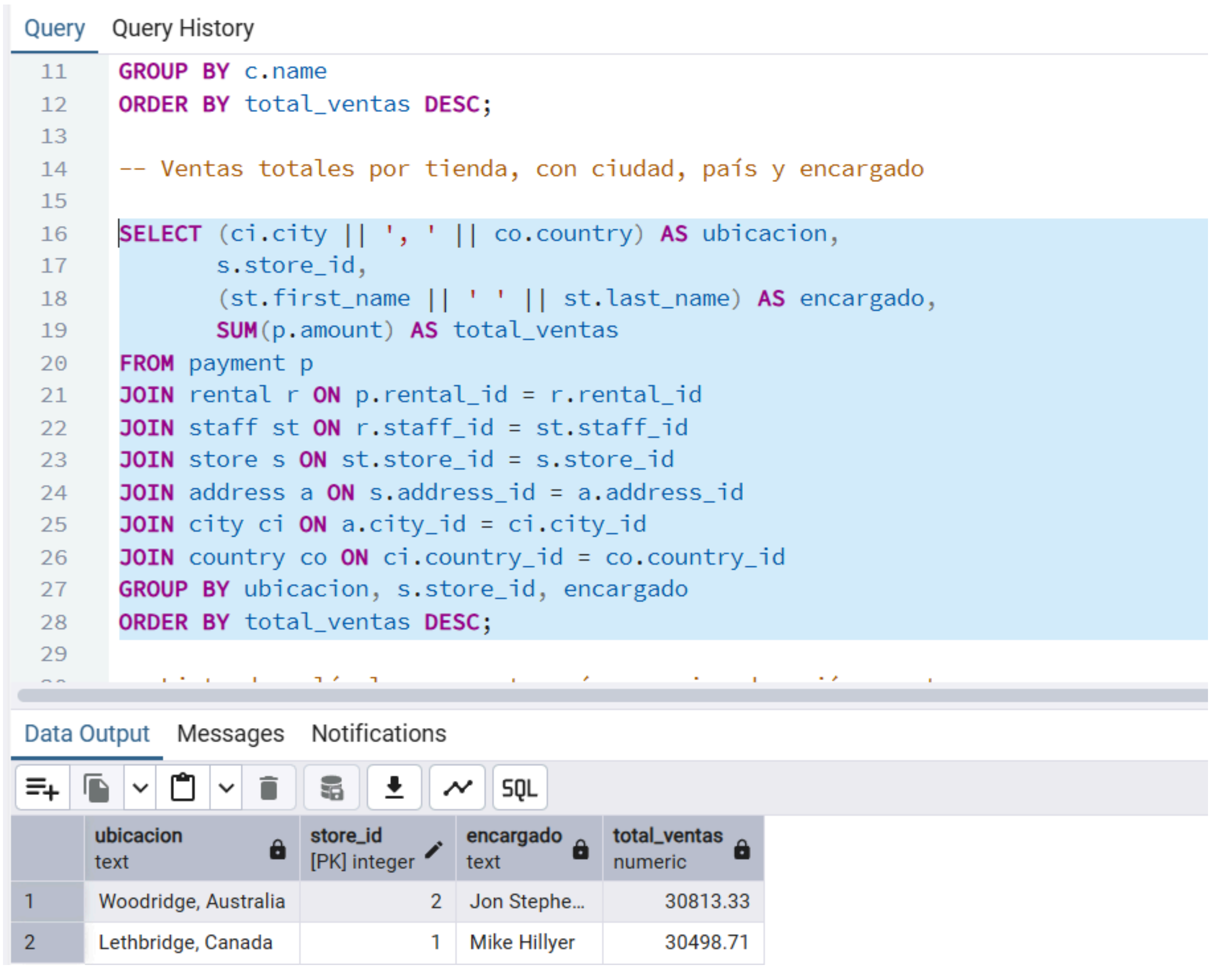
Query Query History

```sql
-- Ventas totales por categoría

SELECT c.name AS categoria,
       SUM(p.amount) AS total_ventas
FROM payment p
JOIN rental r ON p.rental_id = r.rental_id
JOIN inventory i ON r.inventory_id = i.inventory_id
JOIN film f ON i.film_id = f.film_id
JOIN film_category fc ON f.film_id = fc.film_id
JOIN category c ON fc.category_id = c.category_id
GROUP BY c.name
ORDER BY total_ventas DESC;

-- Ventas totales por tienda, con ciudad, país y encargado

SELECT (ci.city || ', ' || co.country) AS ubicacion,
       s.store_id,
       (st.first_name || ' ' || st.last_name) AS encargado,
       SUM(p.amount) AS total_ventas
```

Data Output Messages Notifications

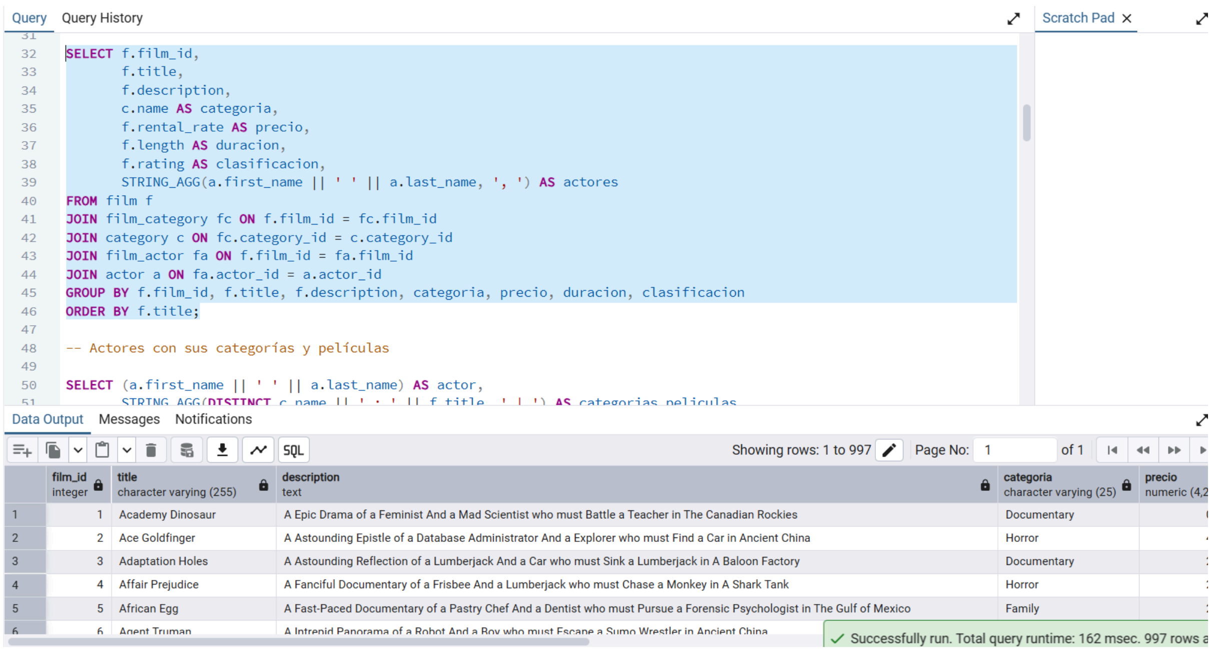
Showing rows: 1 to 16 Page No: 1

	categoria character varying (25)	total_ventas numeric
1	Sports	4892.19
2	Sci-Fi	4336.01
3	Animation	4245.31
4	Drama	4118.46
5	Comedy	4002.48
6	New	3966.38



```sql
GROUP BY c.name
ORDER BY total_ventas DESC;

-- Ventas totales por tienda, con ciudad, país y encargado

SELECT (ci.city || ', ' || co.country) AS ubicacion,
       s.store_id,
       (st.first_name || ' ' || st.last_name) AS encargado,
       SUM(p.amount) AS total_ventas
FROM payment p
JOIN rental r ON p.rental_id = r.rental_id
JOIN staff st ON r.staff_id = st.staff_id
JOIN store s ON st.store_id = s.store_id
JOIN address a ON s.address_id = a.address_id
JOIN city ci ON a.city_id = ci.city_id
JOIN country co ON ci.country_id = co.country_id
GROUP BY ubicacion, s.store_id, encargado
ORDER BY total_ventas DESC;
```

	ubicacion text	store_id [PK] integer	encargado text	total_ventas numeric
1	Woodridge, Australia	2	Jon Stephe...	30813.33
2	Lethbridge, Canada	1	Mike Hillyer	30498.71

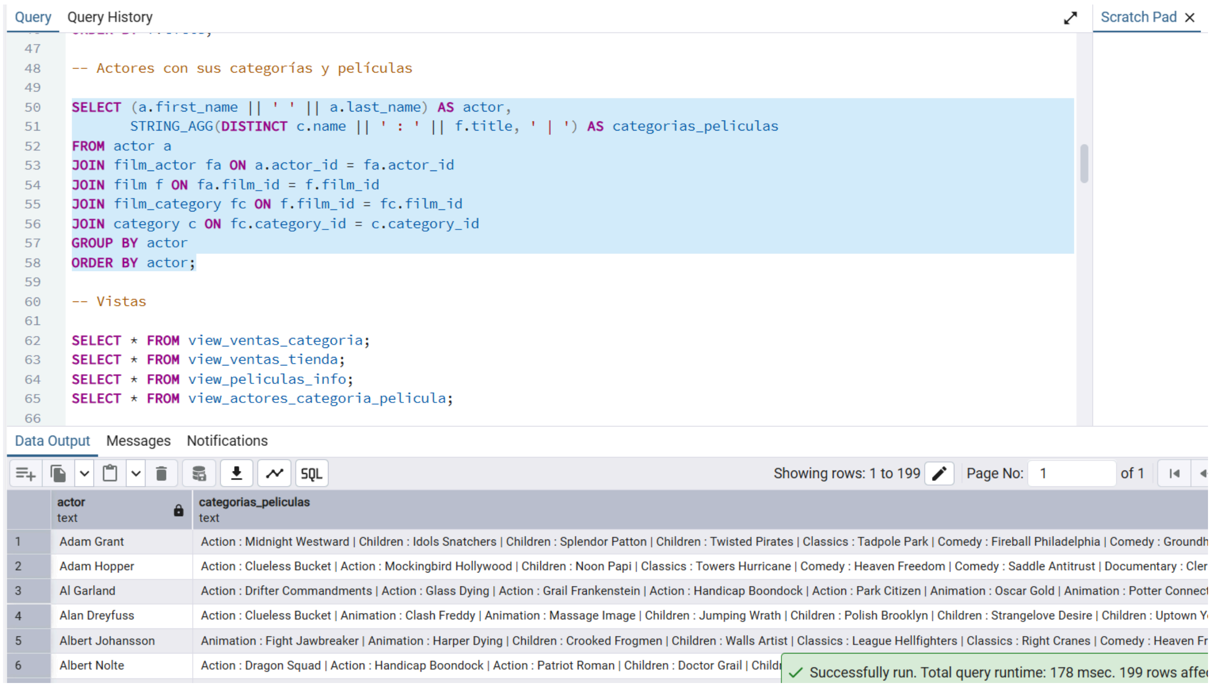


```sql
SELECT f.film_id,
       f.title,
       f.description,
       c.name AS categoria,
       f.rental_rate AS precio,
       f.length AS duracion,
       f.rating AS clasificacion,
       STRING_AGG(a.first_name || ' ' || a.last_name, ', ') AS actores
FROM film f
JOIN film_category fc ON f.film_id = fc.film_id
JOIN category c ON fc.category_id = c.category_id
JOIN film_actor fa ON f.film_id = fa.film_id
JOIN actor a ON fa.actor_id = a.actor_id
GROUP BY f.film_id, f.title, f.description, categoria, precio, duracion, clasificacion
ORDER BY f.title;

-- Actores con sus categorías y películas

SELECT (a.first_name || ' ' || a.last_name) AS actor,
       STRING_AGG(DISTINCT c.name || ': ' || f.title, ' | ') AS categorias_peliculas
```

Showing rows: 1 to 997 Page No: 1 of 1

	film_id integer	title character varying (255)	description text	categoria character varying (25)	precio numeric (4,2
1	1	Academy Dinosaur	A Epic Drama of a Feminist And a Mad Scientist who must Battle a Teacher in The Canadian Rockies	Documentary	
2	2	Ace Goldfinger	A Astounding Epistle of a Database Administrator And a Explorer who must Find a Car in Ancient China	Horror	
3	3	Adaptation Holes	A Astounding Reflection of a Lumberjack And a Car who must Sink a Lumberjack in A Baloon Factory	Documentary	
4	4	Affair Prejudice	A Fanciful Documentary of a Frisbee And a Lumberjack who must Chase a Monkey in A Shark Tank	Horror	
5	5	African Egg	A Fast-Paced Documentary of a Pastry Chef And a Dentist who must Pursue a Forensic Psychologist in The Gulf of Mexico	Family	
6	6	Agent Truman	A Intrepid Panorama of a Robot And a Boy who must Escape a Sumo Wrestler in Ancient China		

✓ Successfully run. Total query runtime: 162 msec. 997 rows a



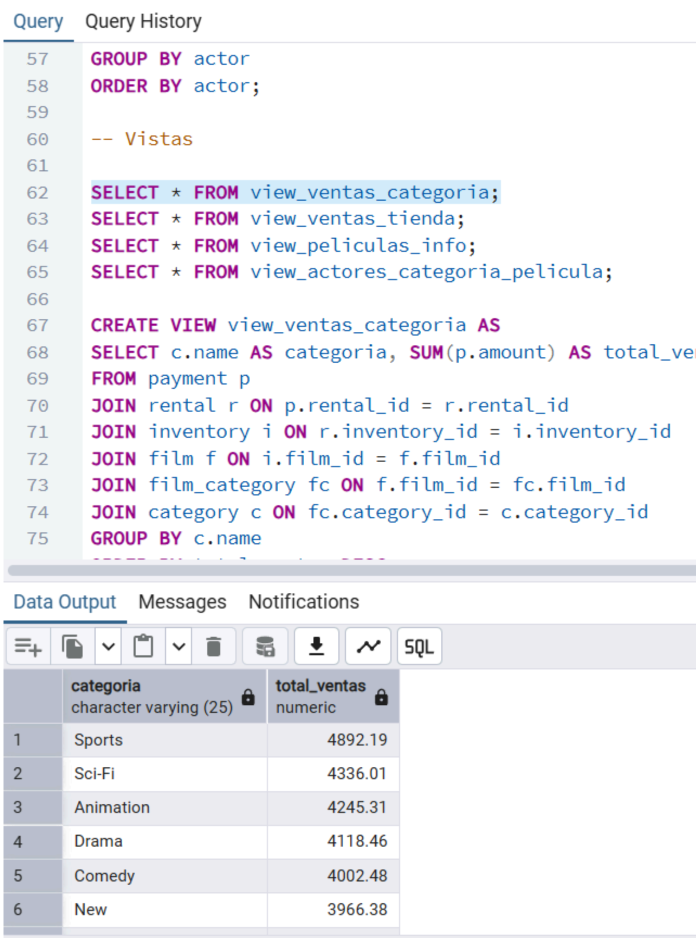
```sql
-- Actores con sus categorías y películas

SELECT (a.first_name || ' ' || a.last_name) AS actor,
       STRING_AGG(DISTINCT c.name || ' : ' || f.title, ' | ') AS categorias_peliculas
FROM actor a
JOIN film_actor fa ON a.actor_id = fa.actor_id
JOIN film f ON fa.film_id = f.film_id
JOIN film_category fc ON f.film_id = fc.film_id
JOIN category c ON fc.category_id = c.category_id
GROUP BY actor
ORDER BY actor;

-- Vistas

SELECT * FROM view_ventas_categoria;
SELECT * FROM view_ventas_tienda;
SELECT * FROM view_peliculas_info;
SELECT * FROM view_actores_categoria_pelicula;
```

Showing rows: 1 to 199 Page No: 1 of 1

	actor text	categorias_peliculas text
1	Adam Grant	Action : Midnight Westward | Children : Idols Snatchers | Children : Splendor Patton | Children : Twisted Pirates | Classics : Tadpole Park | Comedy : Fireball Philadelphia | Comedy : Groundh
2	Adam Hopper	Action : Clueless Bucket | Action : Mockingbird Hollywood | Children : Noon Papi | Classics : Towers Hurricane | Comedy : Heaven Freedom | Comedy : Saddle Antitrust | Documentary : Cler
3	Al Garland	Action : Drifter Commandments | Action : Glass Dying | Action : Grail Frankenstein | Action : Handicap Boondock | Action : Park Citizen | Animation : Oscar Gold | Animation : Potter Connect
4	Alan Dreyfuss	Action : Clueless Bucket | Animation : Clash Freddy | Animation : Massage Image | Children : Jumping Wrath | Children : Polish Brooklyn | Children : Strangelove Desire | Children : Uptown Y
5	Albert Johansson	Animation : Fight Jawbreaker | Animation : Harper Dying | Children : Crooked Frogmen | Children : Walls Artist | Classics : League Hellfighters | Classics : Right Cranes | Comedy : Heaven Fr
6	Albert Nolte	Action : Dragon Squad | Action : Handicap Boondock | Action : Patriot Roman | Children : Doctor Grail | Childr

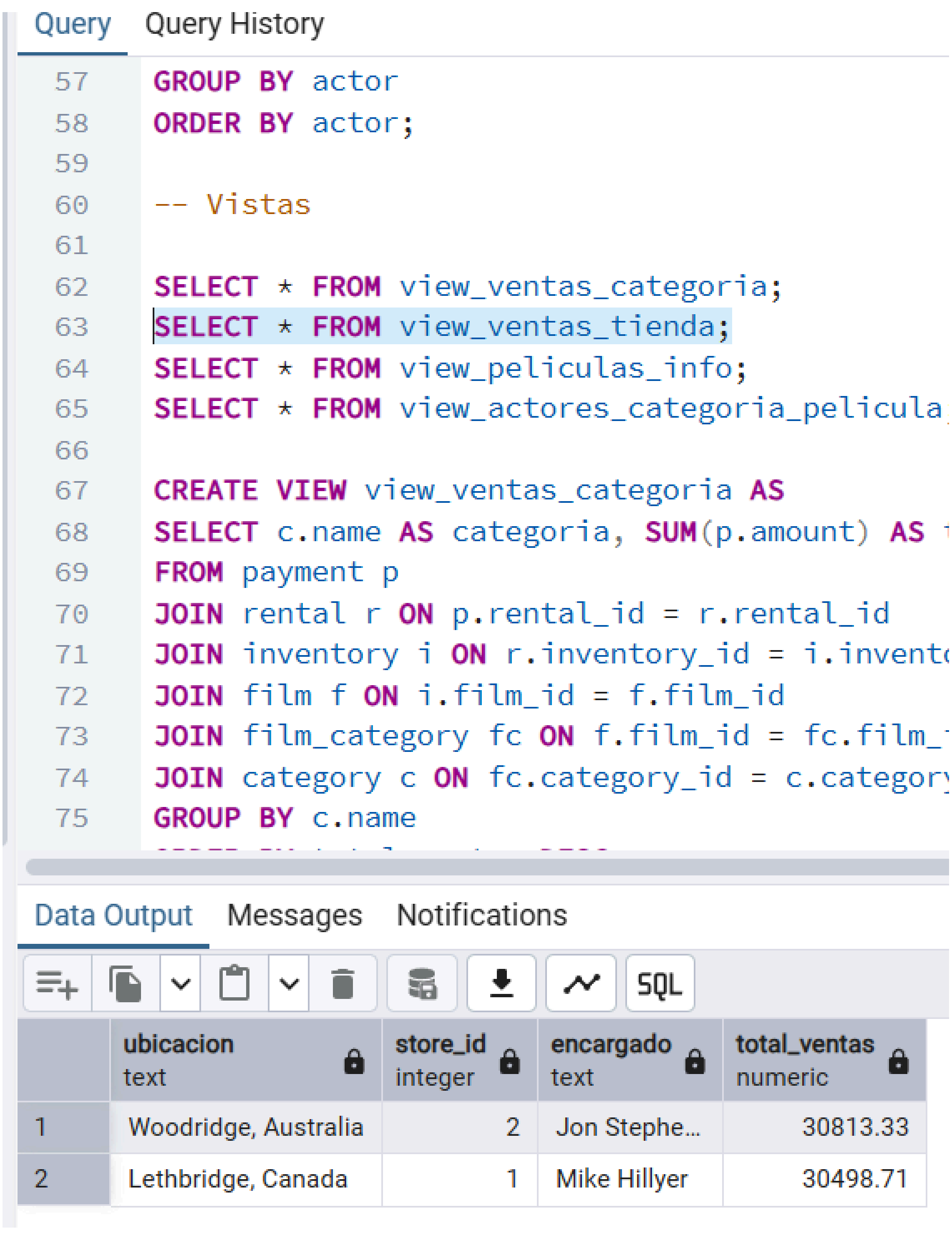
✓ Successfully run. Total query runtime: 178 msec. 199 rows affe

5. Creación de Vistas



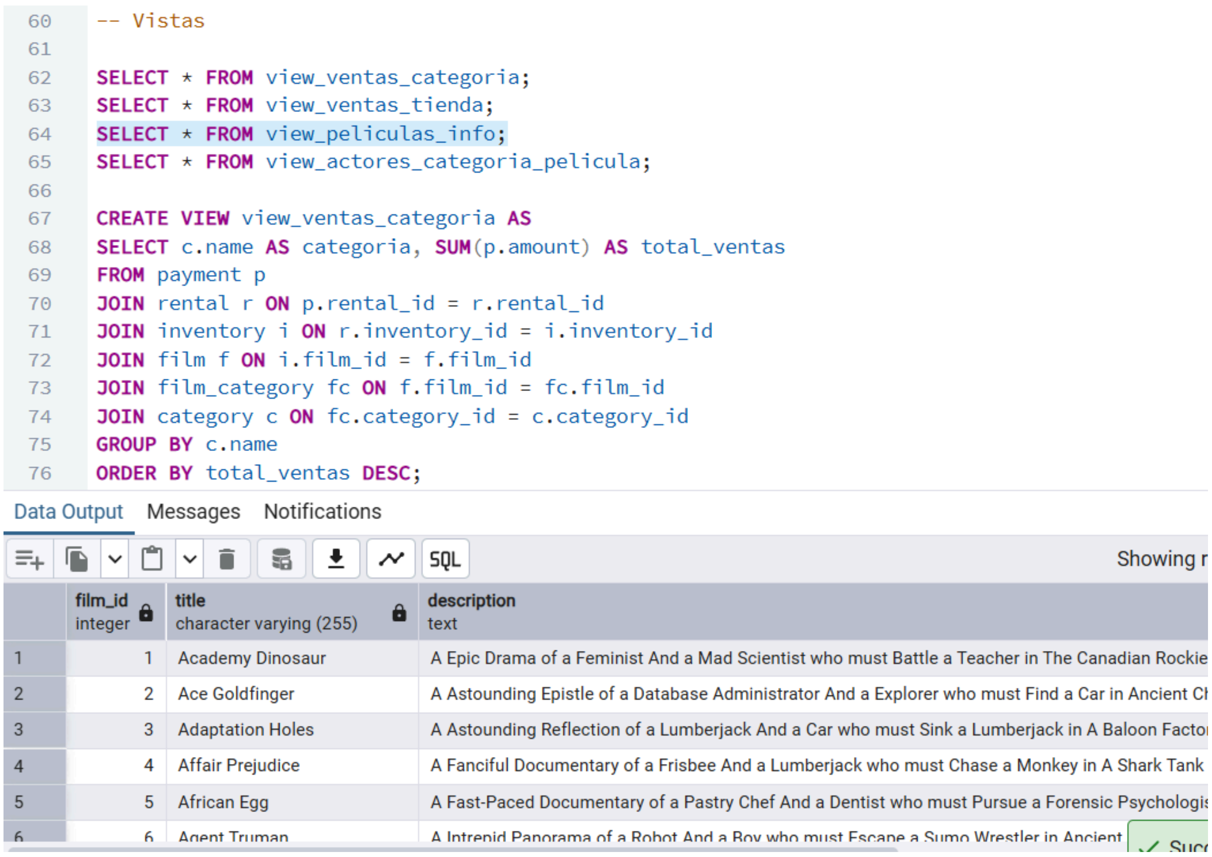
```sql
GROUP BY actor
ORDER BY actor;

-- Vistas

SELECT * FROM view_ventas_categoria;
SELECT * FROM view_ventas_tienda;
SELECT * FROM view_peliculas_info;
SELECT * FROM view_actores_categoria_pelicula;

CREATE VIEW view_ventas_categoria AS
SELECT c.name AS categoria, SUM(p.amount) AS total_ve
FROM payment p
JOIN rental r ON p.rental_id = r.rental_id
JOIN inventory i ON r.inventory_id = i.inventory_id
JOIN film f ON i.film_id = f.film_id
JOIN film_category fc ON f.film_id = fc.film_id
JOIN category c ON fc.category_id = c.category_id
GROUP BY c.name
```

	categoria character varying (25)	total_ventas numeric
1	Sports	4892.19
2	Sci-Fi	4336.01
3	Animation	4245.31
4	Drama	4118.46
5	Comedy	4002.48
6	New	3966.38



```
GROUP BY actor
ORDER BY actor;

-- Vistas

SELECT * FROM view_ventas_categoria;
SELECT * FROM view_ventas_tienda;
SELECT * FROM view_peliculas_info;
SELECT * FROM view_actores_categoria_pelicula

CREATE VIEW view_ventas_categoria AS
SELECT c.name AS categoria, SUM(p.amount) AS
FROM payment p
JOIN rental r ON p.rental_id = r.rental_id
JOIN inventory i ON r.inventory_id = i.invent
JOIN film f ON i.film_id = f.film_id
JOIN film_category fc ON f.film_id = fc.film_
JOIN category c ON fc.category_id = c.categor
GROUP BY c.name
```

	ubicacion text	store_id integer	encargado text	total_ventas numeric
1	Woodridge, Australia	2	Jon Stephe...	30813.33
2	Lethbridge, Canada	1	Mike Hillyer	30498.71

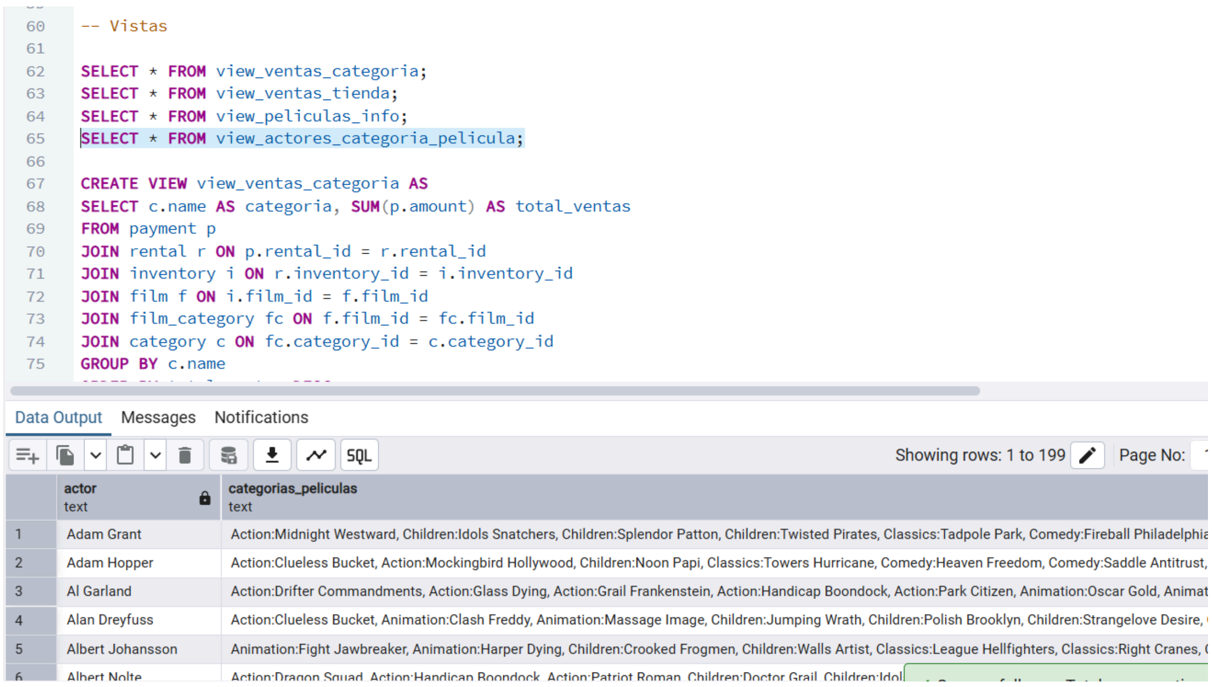


```sql
-- Vistas

SELECT * FROM view_ventas_categoria;
SELECT * FROM view_ventas_tienda;
SELECT * FROM view_peliculas_info;
SELECT * FROM view_actores_categoria_pelicula;

CREATE VIEW view_ventas_categoria AS
SELECT c.name AS categoria, SUM(p.amount) AS total_ventas
FROM payment p
JOIN rental r ON p.rental_id = r.rental_id
JOIN inventory i ON r.inventory_id = i.inventory_id
JOIN film f ON i.film_id = f.film_id
JOIN film_category fc ON f.film_id = fc.film_id
JOIN category c ON fc.category_id = c.category_id
GROUP BY c.name
ORDER BY total_ventas DESC;
```

Data Output Messages Notifications

Showing r

	film_id integer	title character varying (255)	description text
1	1	Academy Dinosaur	A Epic Drama of a Feminist And a Mad Scientist who must Battle a Teacher in The Canadian Rockie
2	2	Ace Goldfinger	A Astounding Epistle of a Database Administrator And a Explorer who must Find a Car in Ancient Ch
3	3	Adaptation Holes	A Astounding Reflection of a Lumberjack And a Car who must Sink a Lumberjack in A Baloon Factor
4	4	Affair Prejudice	A Fanciful Documentary of a Frisbee And a Lumberjack who must Chase a Monkey in A Shark Tank
5	5	African Egg	A Fast-Paced Documentary of a Pastry Chef And a Dentist who must Pursue a Forensic Psychologis
6	6	Agent Truman	A Intrepid Panorama of a Robot And a Boy who must Escape a Sumo Wrestler in Ancient



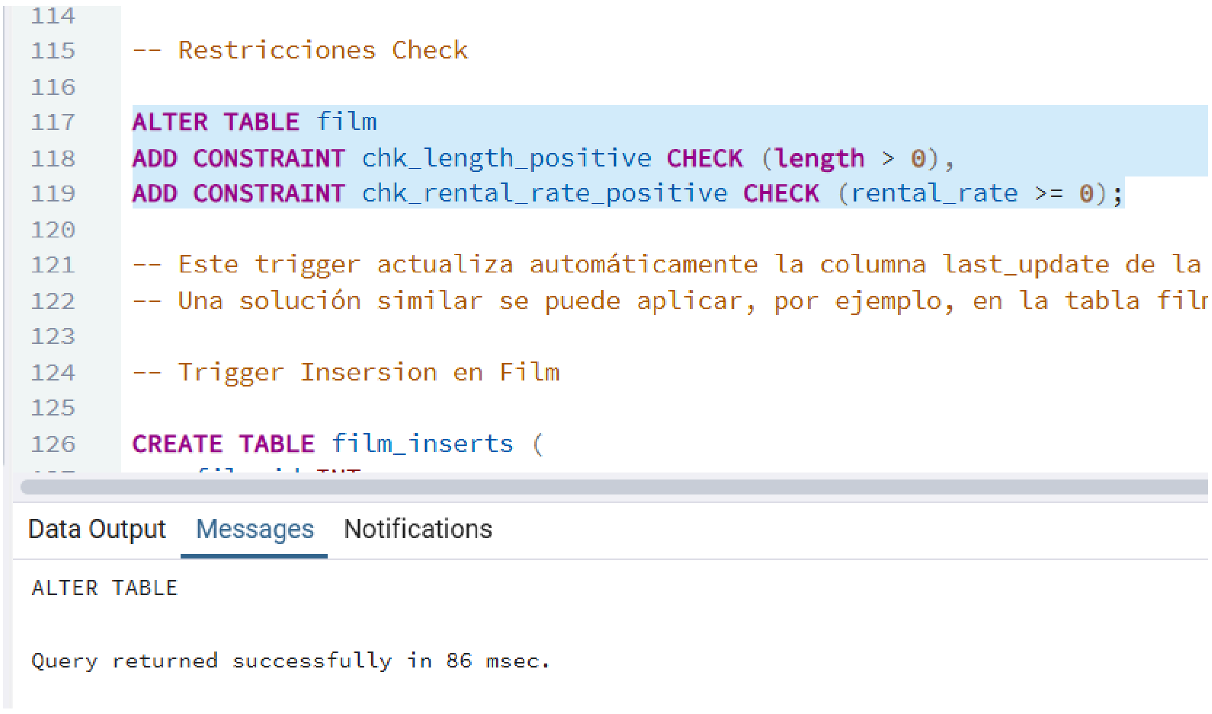
```sql
-- Vistas

SELECT * FROM view_ventas_categoria;
SELECT * FROM view_ventas_tienda;
SELECT * FROM view_peliculas_info;
SELECT * FROM view_actores_categoria_pelicula;

CREATE VIEW view_ventas_categoria AS
SELECT c.name AS categoria, SUM(p.amount) AS total_ventas
FROM payment p
JOIN rental r ON p.rental_id = r.rental_id
JOIN inventory i ON r.inventory_id = i.inventory_id
JOIN film f ON i.film_id = f.film_id
JOIN film_category fc ON f.film_id = fc.film_id
JOIN category c ON fc.category_id = c.category_id
GROUP BY c.name
```

Data Output Messages Notifications

Showing rows: 1 to 199 Page No:

	actor text	categorias_peliculas text
1	Adam Grant	Action:Midnight Westward, Children:Idols Snatchers, Children:Splendor Patton, Children:Twisted Pirates, Classics:Tadpole Park, Comedy:Fireball Philadelphia
2	Adam Hopper	Action:Clueless Bucket, Action:Mockingbird Hollywood, Children:Noon Papi, Classics:Towers Hurricane, Comedy:Heaven Freedom, Comedy:Saddle Antitrust,
3	Al Garland	Action:Drifter Commandments, Action:Glass Dying, Action:Grail Frankenstein, Action:Handicap Boondock, Action:Park Citizen, Animation:Oscar Gold, Animat
4	Alan Dreyfuss	Action:Clueless Bucket, Animation:Clash Freddy, Animation:Massage Image, Children:Jumping Wrath, Children:Polish Brooklyn, Children:Strangelove Desire,
5	Albert Johansson	Animation:Fight Jawbreaker, Animation:Harper Dying, Children:Crooked Frogmen, Children:Walls Artist, Classics:League Hellfighters, Classics:Right Cranes,
6	Albert Nolte	Action:Dragon Squad, Action:Handicap Boondock, Action:Patriot Roman, Children:Doctor Grail, Children:Idol

6. Restricciones Check



```
-- Restricciones Check

ALTER TABLE film
ADD CONSTRAINT chk_length_positive CHECK (length > 0),
ADD CONSTRAINT chk_rental_rate_positive CHECK (rental_rate >= 0);

-- Este trigger actualiza automáticamente la columna last_update de la
-- Una solución similar se puede aplicar, por ejemplo, en la tabla fil

-- Trigger Insersion en Film

CREATE TABLE film_inserts (
```

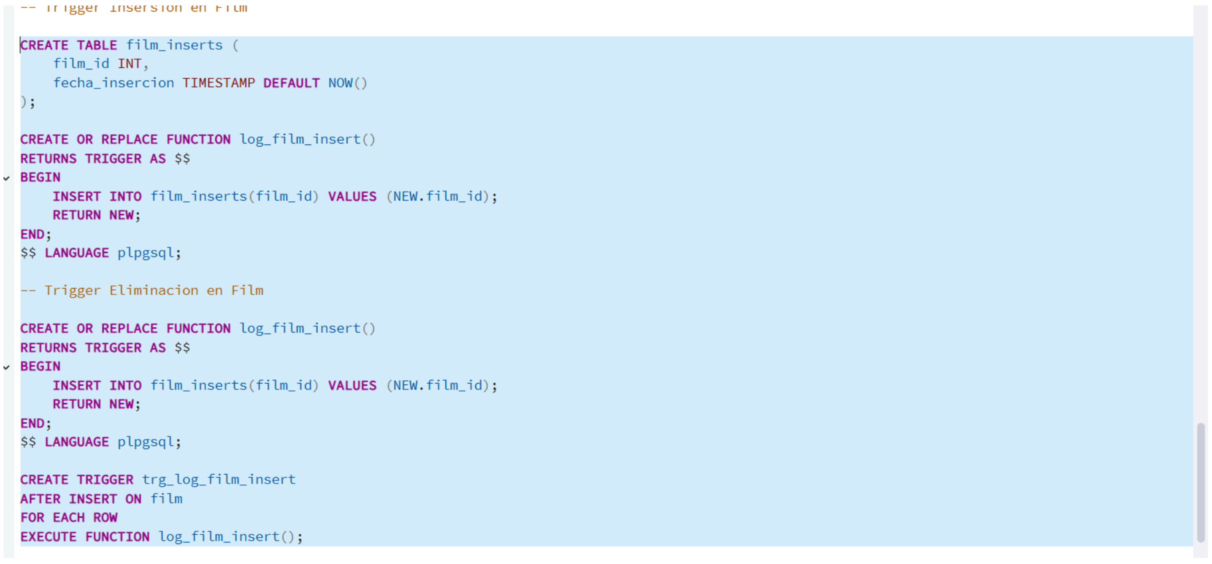
Data Output | Messages | Notifications

```
ALTER TABLE

Query returned successfully in 86 msec.
```

7. ¿Qué hace el Trigger ya creado?

Este trigger actualiza automáticamente la columna last_update de la tabla customer cada vez que se modifica un registro. Una solución similar se puede aplicar, por ejemplo, en la tabla film, inventory o staff, donde queramos mantener el registro de la última modificación.

8. Triggers y Disparadores



```
-- Trigger Insersion en Film

CREATE TABLE film_inserts (
    film_id INT,
    fecha_insercion TIMESTAMP DEFAULT NOW()
);

CREATE OR REPLACE FUNCTION log_film_insert()
RETURNS TRIGGER AS $$
BEGIN
    INSERT INTO film_inserts(film_id) VALUES (NEW.film_id);
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Trigger Eliminacion en Film

CREATE OR REPLACE FUNCTION log_film_insert()
RETURNS TRIGGER AS $$
BEGIN
    INSERT INTO film_inserts(film_id) VALUES (NEW.film_id);
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_log_film_insert
AFTER INSERT ON film
FOR EACH ROW
EXECUTE FUNCTION log_film_insert();
```

9. Significado y Relevancia de Secuencias

En PostgreSQL, las secuencias se usan para generar identificadores únicos automáticamente, normalmente asociados a columnas con SERIAL o GENERATED. Por ejemplo, customer_id o film_id usan secuencias para garantizar valores consecutivos sin conflictos.