

CASO PRÁCTICO 1 - POSTGRESQL

Cristóbal Suárez Abad

ADMINISTRACIÓN DE SISTEMAS GESTORES DE BASES DE DATOS - 2º ASIR

Índice

Evidencias	3
Actividad 1 – Creación de usuarios y roles	4
Actividad 2 - Creación de vistas personalizadas	12
Creación de tablas y datos base	12
Asignación de permisos a usuarios	18
Ampliación	22
Actividad 3 – Sinónimos y alias	23
Creación de sinónimos simulados	23
Permisos y pruebas de acceso	24
Actividad 4 – Gestión de privilegios (criterios d–g)	27
Agrupación de privilegios y rol de solo lectura	29
Privilegios sobre esquemas	34
Auditoría final de roles y privilegios	35

Evidencias

- *Será necesario añadir las sentencias ejecutadas así como consultas de comprobación.*

La empresa TechData S.L., dedicada a la venta y soporte de equipos informáticos, ha decidido implantar una base de datos en PostgreSQL para gestionar la información de clientes y pedidos.

Como administrador de bases de datos, tu tarea consiste en configurar los usuarios, roles y privilegios del sistema, creando vistas personalizadas y aplicando medidas de seguridad y control de acceso.

El objetivo es que cada perfil (administrador de ventas, empleado y auditor) acceda solo a la información necesaria, garantizando la protección de datos y el principio de mínimo privilegio.

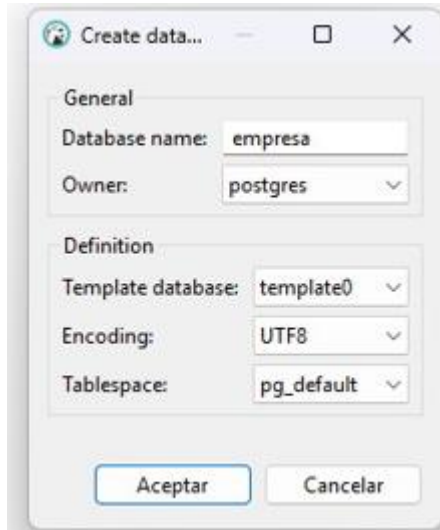
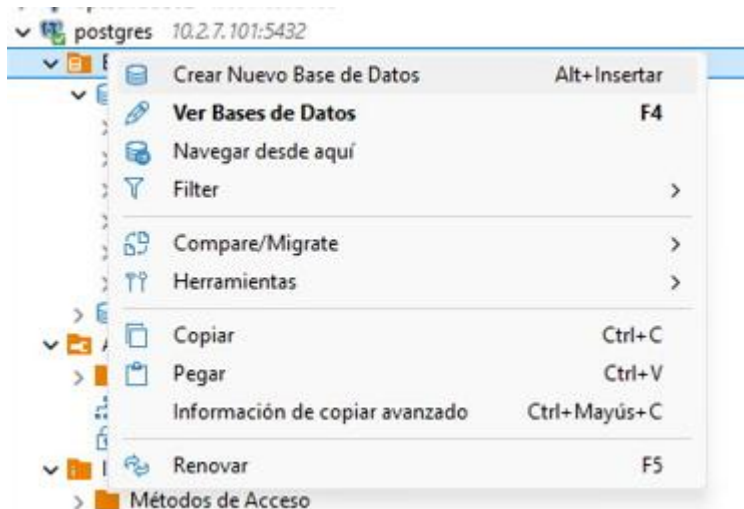
Actividad 1 – Creación de usuarios y roles

1.- Crea una base de datos llamada **empresa**.

Podemos crearlo desde el terminal:

CREATE DATABASE empresa;

O desde DBeaver:

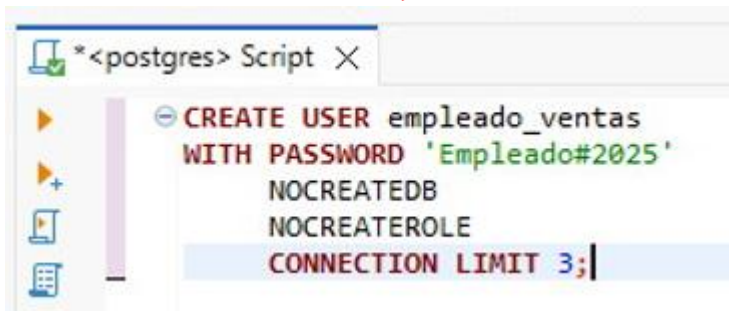


En DBeaver, una vez que lo creamos con la actual conexión recomiendo crear una nueva conexión solo para la nueva base de datos, para evitar confusiones.

2.- Crea tres usuarios con las opciones definidas:

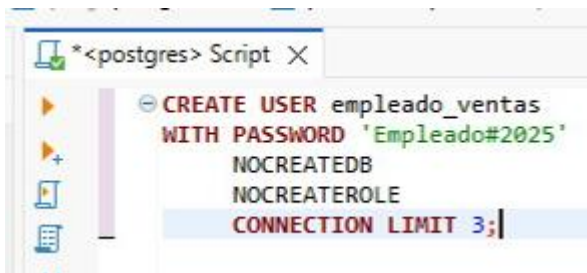
- **admin_ventas**
 - El usuario puede iniciar sesión con la contraseña 'Audit\$2025'
 - Puede crear bases de datos
 - Puede crear y gestionar roles
 - Hereda privilegios de roles asignados
 - Su cuenta expira el 31/12/2026

```
CREATE USER admin_ventas  
WITH PASSWORD 'Audit$2025'  
CREATEDB  
CREATEROLE  
INHERIT  
VALID UNTIL '2026-12-31';
```



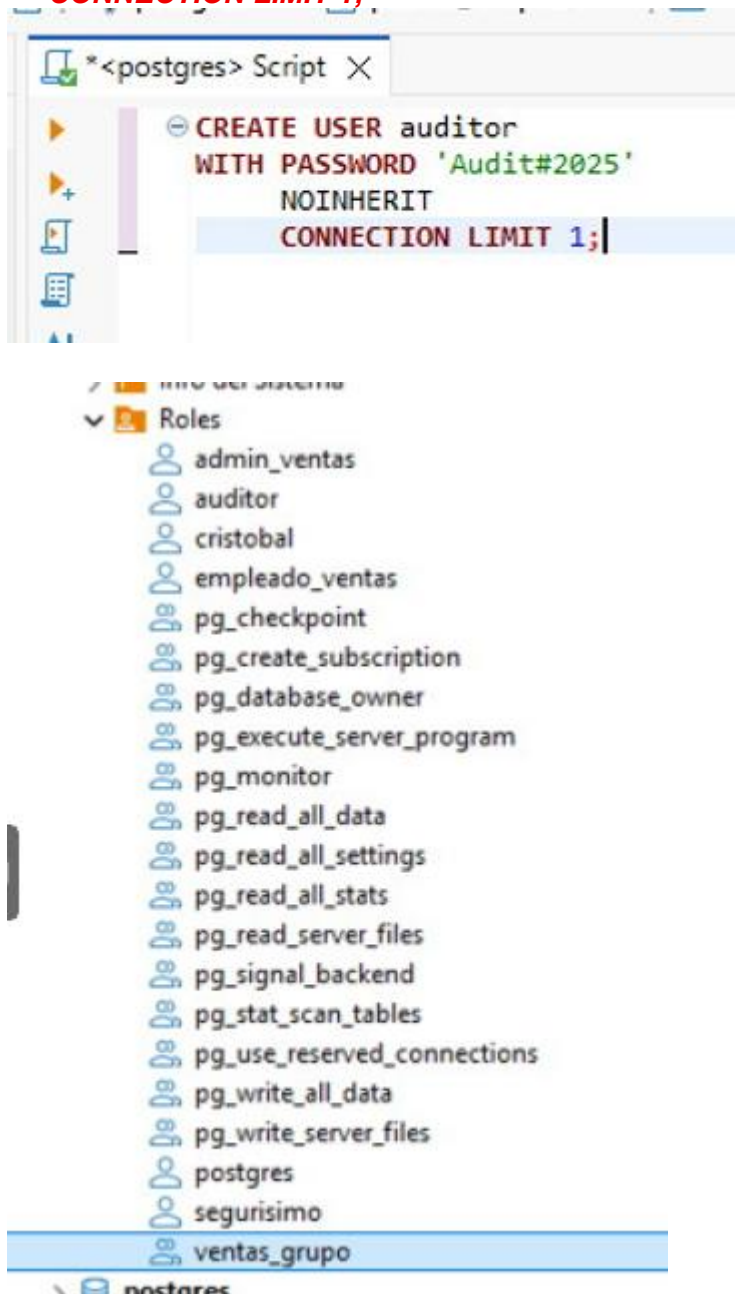
- **empleado_ventas**
 - El usuario puede iniciar sesión con la contraseña 'Empleado#2025'
 - Tiene un límite de 3 conexiones simultáneas
 - No puede crear roles ni bases de datos

```
CREATE USER empleado_ventas  
WITH PASSWORD 'Empleado#2025'  
NOCREATEDB  
NOCREATEROLE  
CONNECTION LIMIT 3;
```



- auditor
 - El usuario puede iniciar sesión con la contraseña 'Audit#2025'
 - No hereda permisos de otros roles
 - Solo puede tener una sesión activa

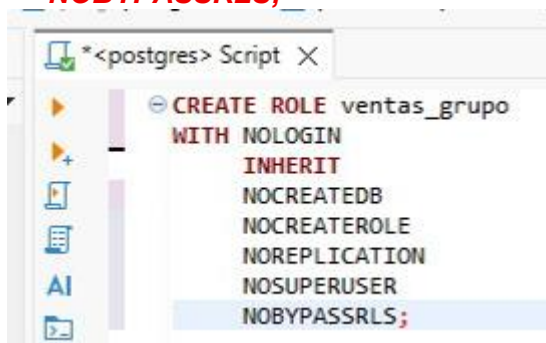
CREATE USER auditor
WITH PASSWORD 'Audit#2025'
NOINHERIT
CONNECTION LIMIT 1;



3.- Crea un rol llamado **ventas_grupo**.

- No puede iniciar sesión
- Puede heredar privilegios
- No puede crear bases de datos ni roles
- No es superusuario ni tiene permisos de replicación
- No puede omitir políticas de seguridad por filas

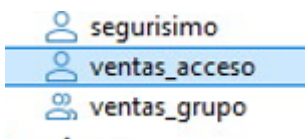
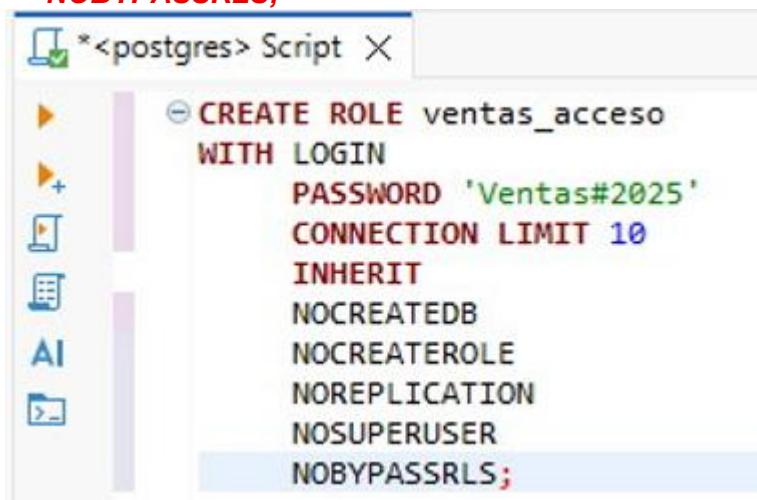
```
CREATE ROLE ventas_grupo  
WITH NOLOGIN  
INHERIT  
NOCREATEDB  
NOCREATEROLE  
NOREPLICATION  
NOSUPERUSER  
NOBYPASSRLS;
```



4.- Crea un rol llamado **ventas_acceso**

- Puede **iniciar sesión**
- Tiene una **contraseña cifrada** 'Ventas#2025'
- Puede realizar hasta **10 conexiones simultáneas**
- **Hereda privilegios** de otros roles
- No puede crear bases de datos ni roles
- No es superusuario ni tiene permisos de replicación
- No puede omitir políticas de seguridad

```
CREATE ROLE ventas_acceso  
WITH LOGIN  
PASSWORD 'Ventas#2025'  
CONNECTION LIMIT 10  
INHERIT  
NOCREATEDB  
NOCREATEROLE  
NOREPLICATION  
NOSUPERUSER  
NOBYPASSRLS;
```



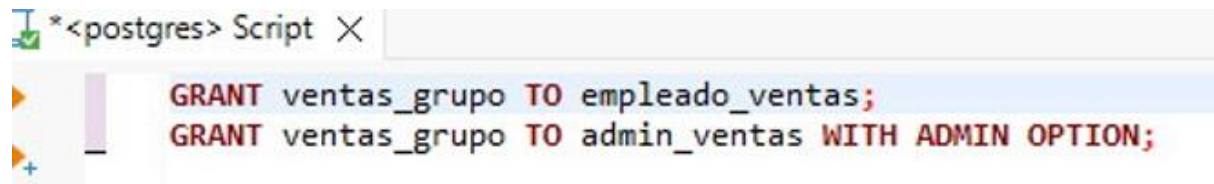
5.- Asocia `empleado_ventas` y `admin_ventas` al rol `ventas_grupo`. `admin_ventas` tendrá permisos de revocación y asignación al role, `empleado_ventas` no.

- Asignación para `empleado_ventas` (sin capacidad de administrar el rol)

GRANT ventas_grupo TO empleado_ventas;

- Asignación para `admin_ventas` (con capacidad de administrar el rol)

GRANT ventas_grupo TO admin_ventas WITH ADMIN OPTION;

A screenshot of a PostgreSQL script editor window titled "*<postgres> Script". The editor contains two SQL statements: "GRANT ventas_grupo TO empleado_ventas;" and "GRANT ventas_grupo TO admin_ventas WITH ADMIN OPTION;". The first statement is highlighted with a blue background, and the second statement is highlighted with a red background. The editor has a standard toolbar with icons for saving, undo, redo, and other editing functions.

```
GRANT ventas_grupo TO empleado_ventas;  
GRANT ventas_grupo TO admin_ventas WITH ADMIN OPTION;
```

6.- Crea una tabla con el usuario **empleado_ventas**, y luego intenta eliminar el usuario. ¿Qué ocurre? ¿Cómo podrías eliminar el usuario? ¿Qué consecuencias tendría?

Previamente, desde un usuario administrador debemos darle permisos de creación de tablas en esquema public al usuario, porque si no, no podrá crear la tabla:

GRANT CREATE ON SCHEMA public TO empleado_ventas;

```

You are now connected to database empresa as user postgres
empresa=# GRANT CREATE ON SCHEMA public TO empleado_ventas;
GRANT
empresa=# |

```

Luego el usuario podrá crear la tabla. En este caso la creamos desde una sesión de DBeaver del empleado_ventas.

```

CREATE TABLE public.menu (
    id varchar NOT NULL,
    plato varchar NULL,
    precio varchar NULL,
    CONSTRAINT restaurante_pk PRIMARY KEY (id)
);

```



Ahora intentamos borrar al usuario desde una cuenta de administrador.

DROP ROLE empleado_ventas;

```

postgres=# DROP ROLE empleado_ventas;
ERROR:  role "empleado_ventas" cannot be dropped because some objects depend on it
DETALLE:  privileges for schema public
2 objects in database empresa
postgres=# |

```

No podemos, porque hay algunos objetos que dependen de él. Hay que conectarse a todas las bases de datos donde tenga algo el usuario y luego ejecutar los siguientes comandos para quitar cualquier privilegio:

REVOKE ALL ON ALL TABLES IN SCHEMA public FROM empleado_ventas;

REVOKE ALL ON ALL SEQUENCES IN SCHEMA public FROM empleado_ventas;

REVOKE ALL ON ALL FUNCTIONS IN SCHEMA public FROM empleado_ventas;

REVOKE ALL ON SCHEMA public FROM empleado_ventas;

Ahora eliminamos al usuario

DROP USER empleado_ventas;

En este caso lo hemos ejecutado tanto en “**postgres**” como en “**empresa**”.

```

root@ubuntumysqlsuarez: /h  X  +  v

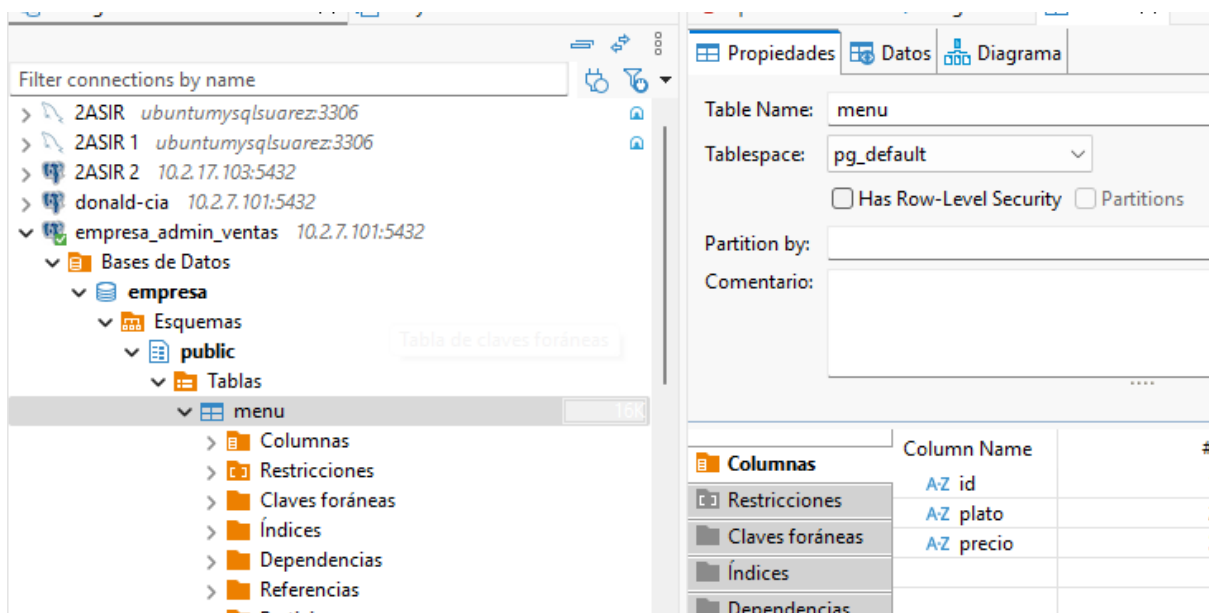
postgres=# REVOKE ALL ON ALL TABLES IN SCHEMA public FROM empleado_ventas;
REVOKE
postgres=# REVOKE ALL ON ALL SEQUENCES IN SCHEMA public FROM empleado_ventas;
REVOKE
postgres=# REVOKE ALL ON ALL FUNCTIONS IN SCHEMA public FROM empleado_ventas;
REVOKE
postgres=# REVOKE ALL ON SCHEMA public FROM empleado_ventas;
REVOKE
postgres=# DROP USER empleado_ventas;
DROP ROLE
postgres=# \c empresa;

```

Consecuencias:

No podemos “logearnos” con sus credenciales, porque ya no existe.

Pero la tabla que creó sigue existiendo: Nos conectamos desde la cuenta de admin_ventas



Actividad 2 - Creación de vistas personalizadas

Creación de tablas y datos base

1. Crea una base de datos para el área de ventas llamada ventas_db

CREATE DATABASE ventas_db;

Y nos conectamos a ella: \c ventas_db;

```
postgres=# CREATE DATABASE ventas_db;
CREATE DATABASE
postgres=# \c ventas_db;
You are now connected to database "ventas_db" as user "postgres".
ventas_db=# |
```

2. Crea las tablas e inserta algunos registros de ejemplo:
 - clientes (id, nombre, dni, telefono, email, saldo)

CREATE TABLE clientes (
id SERIAL PRIMARY KEY,
nombre VARCHAR(100) NOT NULL,
dni VARCHAR(15) UNIQUE,
telefono VARCHAR(20),
email VARCHAR(100),
saldo NUMERIC(10, 2) DEFAULT 0.00
);

```
ventas_db=# CREATE TABLE clientes (
    id SERIAL PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    dni VARCHAR(15) UNIQUE,
    telefono VARCHAR(20),
    email VARCHAR(100),
    saldo NUMERIC(10, 2) DEFAULT 0.00
);
CREATE TABLE
ventas_db=# |
```

- pedidos (id, id_cliente, fecha, total, estado)

```
CREATE TABLE pedidos (  

  id SERIAL PRIMARY KEY,  

  id_cliente INTEGER REFERENCES clientes(id),  

  fecha DATE DEFAULT CURRENT_DATE,  

  total NUMERIC(10, 2) NOT NULL,  

  estado VARCHAR(50)  

);
```

```
ventas_db=# CREATE TABLE pedidos (  

  id SERIAL PRIMARY KEY,  

  id_cliente INTEGER REFERENCES clientes(id),  

  fecha DATE DEFAULT CURRENT_DATE,  

  total NUMERIC(10, 2) NOT NULL,  

  estado VARCHAR(50)  

);  

CREATE TABLE  

ventas_db=# |
```

3. Inserta algunos registros de ejemplo:

-- Inserción en clientes

```
INSERT INTO clientes (nombre, dni, telefono, email, saldo) VALUES  

('Perro Sanchez', '12345678A', '600111222', 'perrete@psoe.com', -9999999.00),  

('Donald Trumpino', '98765432B', '600333444', 'trump@usa.com', 5000000.00),  

('Vladimiro Putinino', '11223344C', '600555666', 'ervladi@ruski.com', 1.00);
```

```
ventas_db=# INSERT INTO clientes (nombre, dni, telefono, email, saldo) VALUES  

('Perro Sanchez', '12345678A', '600111222', 'perrete@psoe.com', -9999999.00),  

('Donald Trumpino', '98765432B', '600333444', 'trump@usa.com', 5000000.00),  

('Vladimiro Putinino', '11223344C', '600555666', 'ervladi@ruski.com', 1.00);  

INSERT 0 3  

ventas_db=# |
```

-- Inserción en pedidos

```
INSERT INTO pedidos (id_cliente, total, estado) VALUES  

(1, 150.00, 'Entregado'),  

(1, 350.50, 'En Proceso'),  

(2, 75.25, 'Pendiente'),  

(3, 1200.00, 'Entregado');
```

```

INSERT 0 3
ventas_db=# INSERT INTO pedidos (id_cliente, total, estado) VALUES
(1, 150.00, 'Entregado'),
(1, 350.50, 'En Proceso'),
(2, 75.25, 'Pendiente'),
(3, 1200.00, 'Entregado');
INSERT 0 4

```

4. Verifica el contenido:

```
SELECT * FROM clientes;
```

```
SELECT * FROM pedidos;
```

```

ventas_db=# SELECT * FROM clientes;
 id |      nombre      | dni   | telefono |      email      |      saldo
-----+-----+-----+-----+-----+-----
  1 | Perro Sanchez    | 12345678A | 600111222 | perrete@psoe.com | -9999999.00
  2 | Donald Trumpino  | 98765432B | 600333444 | trump@usa.com    | 5000000.00
  3 | Vladimiro Putinino | 11223344C | 600555666 | ervladi@ruski.com |      1.00
(3 rows)

ventas_db=# SELECT * FROM pedidos;
 id | id_cliente | fecha      | total  | estado
-----+-----+-----+-----+-----
  1 |          1 | 2025-11-13 | 150.00 | Entregado
  2 |          1 | 2025-11-13 | 350.50 | En Proceso
  3 |          2 | 2025-11-13 |  75.25 | Pendiente
  4 |          3 | 2025-11-13 | 1200.00 | Entregado
(4 rows)

```

Creación de vistas personalizadas

- Los administradores deben tener acceso total a los datos de clientes y pedidos, con el número de pedidos y total de todos los pedidos

```
CREATE VIEW vista_admin_ventas AS
SELECT
  c.id AS cliente_id,
  c.nombre,
  c.dni,
  c.telefono,
  c.email,
  c.saldo,
  COUNT(p.id) AS numero_pedidos,
  COALESCE(SUM(p.total), 0) AS total_gastado
FROM
  clientes c
LEFT JOIN
  pedidos p ON c.id = p.id_cliente
GROUP BY
  c.id, c.nombre, c.dni, c.telefono, c.email, c.saldo;
```

```
ventas_db=# CREATE VIEW vista_admin_ventas AS
SELECT
  c.id AS cliente_id,
  c.nombre,
  c.dni,
  c.telefono,
  c.email,
  c.saldo,
  COUNT(p.id) AS numero_pedidos,
  COALESCE(SUM(p.total), 0) AS total_gastado
FROM
  clientes c
LEFT JOIN
  pedidos p ON c.id = p.id_cliente
GROUP BY
  c.id, c.nombre, c.dni, c.telefono, c.email, c.saldo;
CREATE VIEW
ventas_db=# |
```

clientes 1									
SELECT c.id AS cliente_id, c.nombre, c.dni, c.telefono, c.en Enter a SQL expression to filter results (use Ctrl+Space)									
Grilla	123 cliente_id	AZ nombre	AZ dni	AZ telefono	AZ email	123 saldo	123 numero_pedidos	123 total_gastado	
1	3	Vladimiro Putinino	11223344C	600555666	ervladi@ruski.com	1	1	1,200	
2	2	Donaldo Trumpino	98765432B	600333444	trump@usa.com	5,000.000	1	75,25	
3	1	Perro Sanchez	12345678A	600111222	perrete@psoe.com	-9,999.999	2	500,5	

6. Los empleados solo deben ver información de contacto y saldo, sin DNI ni email.

CREATE VIEW vista_empleado_ventas AS

SELECT

id,

nombre,

telefono,

saldo

FROM

clientes;

```
ventas_db=# CREATE VIEW vista_empleado_ventas AS
SELECT
    id,
    nombre,
    telefono,
    saldo
FROM
    clientes;
CREATE VIEW
```

SELECT id, nombre, telefono, saldo FROM clientes | Enter a SQL expression to filter results

	Grilla	123 id	A-Z nombre	A-Z telefono	123 saldo
	1	1	Perro Sanchez	600111222	-9.999.999
	2	2	Donaldo Trumpino	600333444	5.000.000
	3	3	Vladimiro Putinino	600555666	1

- El auditor puede consultar datos, pero sin información personal identificable.

```
CREATE VIEW vista_auditor AS
SELECT
  c.id AS cliente_id,
  c.saldo,
  p.id AS pedido_id,
  p.fecha,
  p.total,
  p.estado
FROM
  clientes c
LEFT JOIN
  pedidos p ON c.id = p.id_cliente;
```

```
ventas_db=# CREATE VIEW vista_auditor AS
SELECT
  c.id AS cliente_id,
  c.saldo,
  p.id AS pedido_id,
  p.fecha,
  p.total,
  p.estado
FROM
  clientes c
LEFT JOIN
  pedidos p ON c.id = p.id_cliente;
CREATE VIEW
```

SELECT c.id AS cliente_id, c.saldo, p.id AS pedido_id, p.fecha, p.total, p.estado

	123 cliente_id	123 saldo	123 pedido_id	fecha	123 total	A-Z estado
1	1	-9.999.999	1	2025-11-13	150	Entregado
2	1	-9.999.999	2	2025-11-13	350,5	En Proceso
3	2	5.000.000	3	2025-11-13	75,25	Pendiente
4	3	1	4	2025-11-13	1.200	Entregado

Asignación de permisos a usuarios

8. Concede permisos de lectura sobre las vistas a cada role o usuarios:

Asignar la vista de administración al usuario admin_ventas

GRANT SELECT ON vista_admin_ventas TO admin_ventas;

```
ventas_db=# GRANT SELECT ON vista_admin_ventas TO admin_ventas;
GRANT
ventas_db=# |
```

Asignar la vista de empleados al usuario empleado_ventas.

GRANT SELECT ON vista_empleado_ventas TO empleado_ventas;

```
ventas_db=# GRANT SELECT ON vista_empleado_ventas TO empleado_ventas;
GRANT
ventas_db=# |
```

Asignar la vista de auditoría al usuario auditor

GRANT SELECT ON vista_auditor TO auditor;

```
ventas_db=# GRANT SELECT ON vista_auditor TO auditor;
GRANT
ventas_db=# |
```

9. Revoca permisos directos sobre las tablas base, para que solo puedan acceder a través de las vistas

REVOKE ALL ON ALL TABLES IN SCHEMA public FROM PUBLIC;

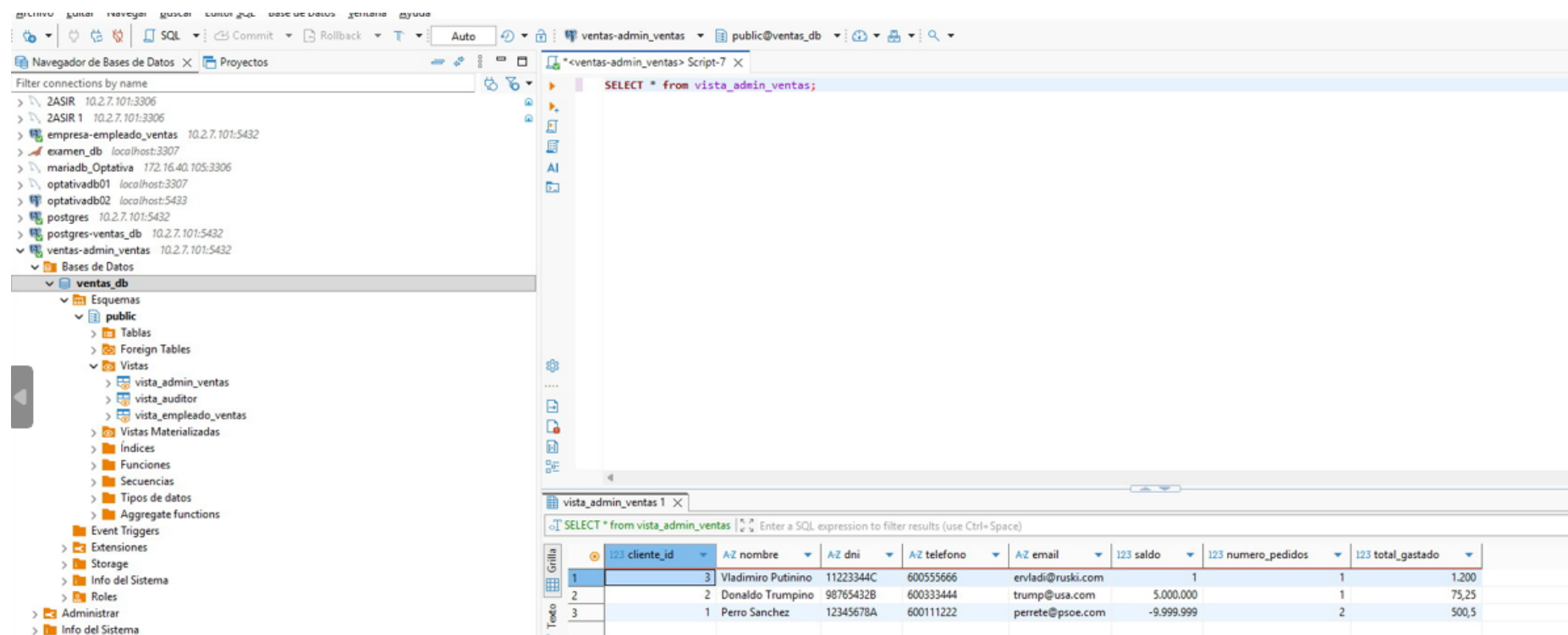
```
ventas_db=# REVOKE ALL ON ALL TABLES IN SCHEMA public FROM PUBLIC;
REVOKE
ventas_db=# |
```

Por si acaso, también podemos usar:

```
REVOKE ALL ON clientes FROM ventas_grupo;
REVOKE ALL ON pedidos FROM ventas_grupo;
REVOKE ALL ON clientes FROM auditor;
REVOKE ALL ON pedidos FROM auditor;
```

10. Comprueba que cada usuario solo puede acceder a su vista correspondiente:

“admin_ventas”: **SELECT * from vista_admin_ventas;**



The screenshot shows a database management tool interface. On the left, a tree view displays the database structure, including a schema named 'ventas_db' with a public schema containing several views. The main window shows a SQL query: `SELECT * from vista_admin_ventas;`. Below the query, the results are displayed in a table with 9 columns: cliente_id, nombre, dni, telefono, email, saldo, numero_pedidos, total_gastado, and an additional column labeled '123'. The results show three rows of data.

	123 cliente_id	AZ nombre	AZ dni	AZ telefono	AZ email	123 saldo	123 numero_pedidos	123 total_gastado
1	3	Vladimiro Putinino	11223344C	600555666	ervladi@ruski.com	1	1	1.200
2	2	Donaldo Trumpino	98765432B	600333444	trump@usa.com	5.000.000	1	75,25
3	1	Perro Sanchez	12345678A	600111222	perrete@psoe.com	-9.999.999	2	500,5

“auditor”: **select * from vista_auditor;**

The screenshot shows a database management interface with a left sidebar for the database explorer and a main area for SQL queries and results.

Database Explorer (Left Sidebar):

- Navegador de Bases de Datos
- Proyectos
- Filter connections by name
- 2ASIR 10.2.7.101:3306
- 2ASIR 1 10.2.7.101:3306
- empresa-empleado_ventas 10.2.7.101:5432
- examen_db localhost:3307
- mariadb_Optativa 172.16.40.105:3306
- optativadb01 localhost:3307
- optativadb02 localhost:5433
- postgres 10.2.7.101:5432
- postgres-ventas_db 10.2.7.101:5432
- ventas_db-admin_ventas 10.2.7.101:5432
- ventas_db-auditor 10.2.7.101:5432**
 - Bases de Datos
 - ventas_db
 - Esquemas
 - public
 - Tablas
 - clientes 40K
 - pedidos 24K
 - Foreign Tables
 - Vistas
 - Vistas Materializadas
 - Índices
 - Funciones
 - Secuencias
 - Tipos de datos
 - Aggregate functions
 - Event Triggers
 - Extensiones
 - Storage
 - Info del Sistema
 - Roles
 - Administrar
 - Info del Sistema

SQL Editor (Main Area):

Script-8

```
select * from vista_auditor;
```

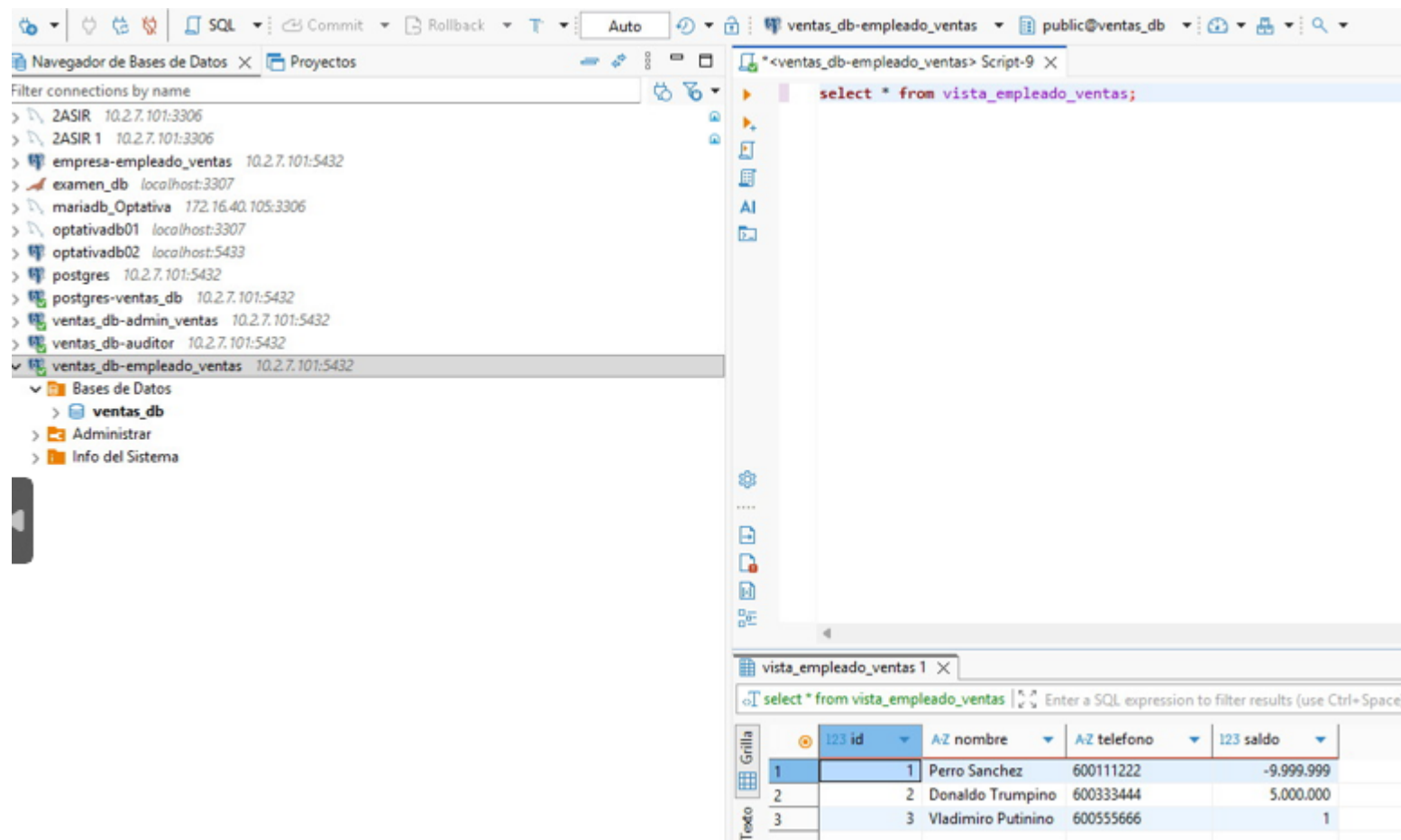
Results (Bottom Panel):

vista_auditor 1

select * from vista_auditor

	cliente_id	saldo	pedido_id	fecha	total	estado
1	1	-9.999.999	1	2025-11-13	150	Entregado
2	1	-9.999.999	2	2025-11-13	350,5	En Proceso
3	2	5.000.000	3	2025-11-13	75,25	Pendiente
4	3	1	4	2025-11-13	1.200	Entregado

“empleado_ventas”: **select * from vista_empleado_ventas;**



The screenshot shows a database management interface. On the left, a tree view displays the database structure, with 'ventas_db-empleado_ventas' selected. The main window shows a SQL script with the query: `select * from vista_empleado_ventas;`. Below the script, the results are displayed in a table view. The table has four columns: 'id', 'nombre', 'telefono', and 'saldo'. The results show three rows of data.

	id	nombre	telefono	saldo
1	1	Perro Sanchez	600111222	-9.999.999
2	2	Donaldo Trumpino	600333444	5.000.000
3	3	Vladimiro Putinino	600555666	1

Ampliación

11. Crea una nueva vista vista_clientes_negativos que muestre solo clientes con saldo menor que 0.

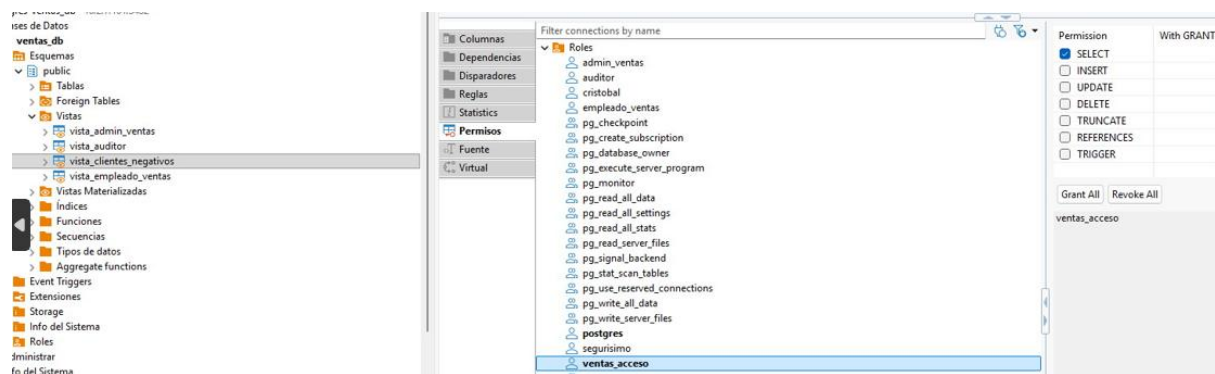
```
CREATE VIEW vista_clientes_negativos AS
SELECT
  id,
  nombre,
  telefono,
  saldo
FROM
  clientes
WHERE
  saldo < 0;
```

```
ventas_db=# CREATE VIEW vista_clientes_negativos AS
SELECT
  id,
  nombre,
  telefono,
  saldo
FROM
  clientes
WHERE
  saldo < 0;
CREATE VIEW
```

12. Asigna esta vista al rol ventas_acceso para que todos los usuarios de ventas puedan consultarla.

GRANT SELECT ON vista_clientes_negativos TO ventas_acceso;

```
ventas_db=# GRANT SELECT ON vista_clientes_negativos TO ventas_acceso;
GRANT
ventas_db=#
```



Actividad 3 – Sinónimos y alias

Creación de sinónimos simulados

Recuerda: PostgreSQL no tiene el comando **CREATE SYNONYM** (como Oracle o SQL Server).

- Crea una vista simple que actúe como alias **vista_clientes_admin** de la tabla **clientes**:

```
CREATE VIEW vista_clientes_admin AS
SELECT *
FROM public.clientes;
```

```
ventas_db=# CREATE VIEW vista_clientes_admin AS
SELECT *
FROM public.clientes;
CREATE VIEW
ventas_db=# |
```

- Crea otra vista que funcione como sinónimo de la vista **vista_clientes_admin**:

```
CREATE VIEW vca_sinonimo AS
SELECT *
FROM vista_clientes_admin;
```

```
ventas_db=# CREATE VIEW vca_sinonimo AS
SELECT *
FROM vista_clientes_admin;
CREATE VIEW
```

- Crea un alias más complejo para la vista de empleados, que renombre columnas

```
CREATE VIEW ventas_contacto AS
SELECT
  id AS cliente_id,
  nombre AS nombre_cliente,
  telefono AS contacto_principal,
  saldo AS deuda_o_credito
FROM
  vista_empleado_ventas;
```

```
ventas_db=# CREATE VIEW ventas_contacto AS
SELECT
  id AS cliente_id,
  nombre AS nombre_cliente,
  telefono AS contacto_principal,
  saldo AS deuda_o_credito
FROM
  vista_empleado_ventas;
CREATE VIEW
```

Permisos y pruebas de acceso

- Concede permisos de lectura sobre los alias a los usuarios o roles que consideres.

GRANT SELECT ON vista_clientes_admin TO admin_ventas;

GRANT SELECT ON vca_sinonimo TO admin_ventas;

GRANT SELECT ON ventas_contacto TO ventas_grupo;

```
CREATE VIEW
ventas_db=# GRANT SELECT ON vista_clientes_admin TO admin_ventas;
GRANT
ventas_db=# GRANT SELECT ON vca_sinonimo TO admin_ventas;
GRANT
ventas_db=# GRANT SELECT ON ventas_contacto TO ventas_grupo;
GRANT
```

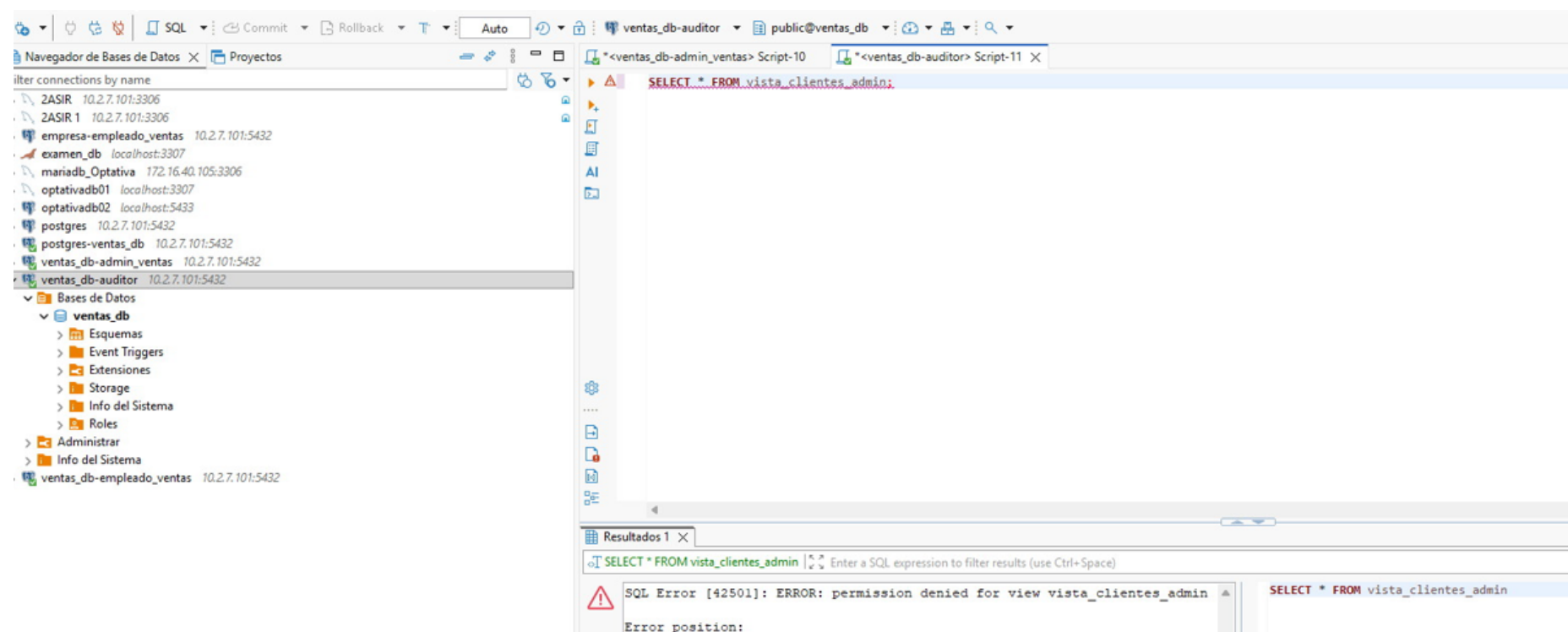
- Comprueba la diferencia entre un usuario con permisos y sin permisos

Si lo hacemos desde el “admin_ventas” funciona.

The screenshot shows a database management interface. On the left, a tree view displays the database structure, including the 'ventas_db' database and its 'public' schema. The 'vistas' (views) folder is expanded, showing several views, including 'vista_clientes_admin'. The main window displays a SQL script titled '<ventas_db-admin_ventas> Script-10' with the query: `SELECT * FROM vista_clientes_admin;`. Below the script, the results of the query are shown in a table format. The table has columns: 'id', 'nombre', 'dni', 'telefono', 'email', and 'saldo'. The results show three rows of data.

	123 id	AZ nombre	AZ dni	AZ telefono	AZ email	123 saldo
1	1	Perro Sanchez	12345678A	600111222	perrete@poe.com	-9.999.999
2	2	Donaldo Trumpino	98765432B	600333444	trump@usa.com	5.000.000
3	3	Vladimiro Putinino	11223344C	600555666	ervladi@ruski.com	1

Si lo intentamos desde del “auditor” no funciona.



Actividad 4 – Gestión de privilegios (criterios d–g)

Comprobación previa

- Lista los privilegios actuales sobre todas las tablas y vistas:

\dp *.*

```
ventas_db=# \dp *.*|
```

Schema	Name	Access privileges	
Column privileges	Policies	Type	Access privileges
information_schema	_pg_foreign_data_wrappers	view	
information_schema	_pg_foreign_servers	view	
information_schema	_pg_foreign_table_columns	view	
information_schema	_pg_foreign_tables	view	
information_schema	_pg_user_mappings	view	
information_schema	administrable_role_authorizations	view	postgres=arwdDxt/postgres +
			=r/postgres
information_schema	applicable_roles	view	postgres=arwdDxt/postgres +
			=r/postgres
information_schema	attributes	view	postgres=arwdDxt/postgres +
			=r/postgres
information_schema	character_sets	view	postgres=arwdDxt/postgres +
			=r/postgres
information_schema	check_constraint_routine_usage	view	postgres=arwdDxt/postgres +

- Comprueba también los roles y sus pertenencias:

\du

```
ventas_db=# \du
```

Role name	Attributes	
admin_ventas	Create role, Create DB	+
	Password valid until 2026-12-31 00:00:00+01	
auditor	No inheritance	+
	15 connections	
cristobal		
empleado_ventas	3 connections	
postgres	Superuser, Create role, Create DB, Replication, Bypass RLS	
segurísimo	Superuser, Create role, Create DB, Replication, Bypass RLS	
ventas_acceso	10 connections	
ventas_grupo	Cannot login	

Asignación y prueba de privilegios

- Sobre la tabla clientes
 - Concede privilegios de manipulación a **ventas_grupo** sobre **clientes**

GRANT SELECT, INSERT, UPDATE, DELETE ON public.clientes TO ventas_grupo;

```
ventas_db=# GRANT INSERT, UPDATE, DELETE ON public.clientes TO ventas_grupo;
GRANT
ventas_db=# REVOKE DELETE ON public.clientes FROM empleado_ventas;
```

- Da permisos de eliminación **solo** al usuario **admin_ventas**:

GRANT DELETE ON public.clientes TO admin_ventas;

```
GRANT
ventas_db=# REVOKE DELETE ON public.clientes FROM empleado_ventas;
REVOKE
ventas_db=# GRANT DELETE ON public.clientes TO admin_ventas;
GRANT
```

- Revoca explícitamente el permiso de eliminación a **empleado_ventas**:

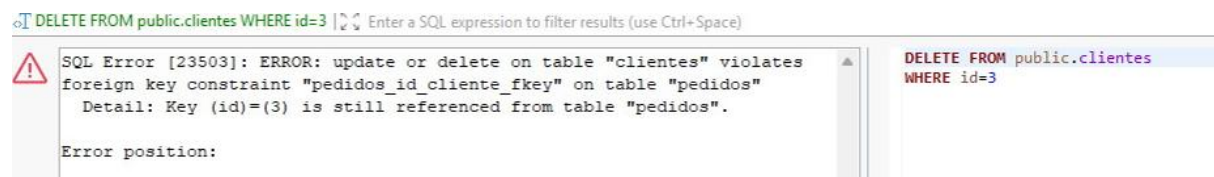
REVOKE DELETE ON public.clientes TO empleado_ventas;

- Verifica el efecto práctico:

Desde el usuario **admin_ventas**:

**DELETE FROM public.clientes
WHERE id=3;**

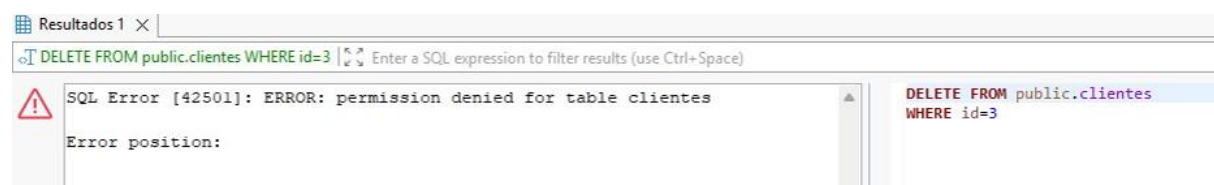
En teoría podemos borrarlo, pero no nos deja porque está siendo usada en otra tabla.



Desde el usuario **empleado_ventas**:

**DELETE FROM public.clientes
WHERE id=3;**

A este no le deja porque no tiene permiso.



Agrupación de privilegios y rol de solo lectura

- Crea un rol de solo lectura:

CREATE ROLE solo_lectura NOLOGIN;

```
ventas_db=# CREATE ROLE solo_lectura NOLOGIN;  
CREATE ROLE
```

- Concede privilegios de lectura sobre todas las tablas y vistas actuales:

GRANT SELECT ON ALL TABLES IN SCHEMA public TO solo_lectura;

Para poder acceder a los objetos también hay que darle este permiso:

GRANT USAGE ON SCHEMA public TO solo_lectura;

```
ventas_db=# GRANT SELECT ON ALL TABLES IN SCHEMA public TO solo_lectura;  
GRANT  
ventas_db=# GRANT USAGE ON SCHEMA public TO solo_lectura;  
GRANT
```

- Haz que los **nuevos objetos creados** también sean legibles por este rol:

ALTER DEFAULT PRIVILEGES IN SCHEMA public GRANT SELECT ON TABLES TO solo_lectura;

```
ventas_db=# ALTER DEFAULT PRIVILEGES IN SCHEMA public GRANT SELECT ON TABLES TO solo_lectura;  
ALTER DEFAULT PRIVILEGES
```

- Asigna el rol al usuario **auditor**:

GRANT solo_lectura TO auditor;

```
ventas_db=# GRANT solo_lectura TO auditor;  
GRANT ROLE
```

- Comprueba que el usuario **auditor** hereda los permisos:

- Inicia sesión como **auditor** y prueba:

Desde la sesión del auditor, ejecutamos:

SELECT * FROM public.clientes;

The screenshot shows a database management interface. On the left, a tree view displays the database structure, including a connection named 'ventas_db-auditor'. The main window shows the SQL query 'SELECT * FROM public.clientes;' being executed. Below the query editor, the results are displayed in a table format.

	id	nombre	dni	telefono	email	saldo
1	1	Perro Sanchez	12345678A	600111222	perrete@psoe.com	-9.999.999
2	2	Donaldo Trumpino	98765432B	600333444	trump@usa.com	5.000.000
3	3	Vladimiro Putinino	11223344C	600555666	envladi@ruski.com	1

SELECT * FROM pedidos;

The screenshot shows a database management interface. On the left, a tree view displays the database structure for 'ventas_db', including schemas, event triggers, extensions, storage, system info, roles, and administrators. The main window shows a SQL query editor with the query 'SELECT * FROM pedidos;'. Below the editor, a table titled 'pedidos 1' displays the results of the query. The table has columns for 'id', 'id_cliente', 'fecha', 'total', and 'estado'. The results show four rows of data.

	id	id_cliente	fecha	total	estado
1	1	1	2025-11-13	150	Entregado
2	2	1	2025-11-13	350,5	En Proceso
3	3	2	2025-11-13	75,25	Pendiente
4	4	3	2025-11-13	1.200	Entregado

Gestión dinámica de privilegios

Sobre la tabla clientes:

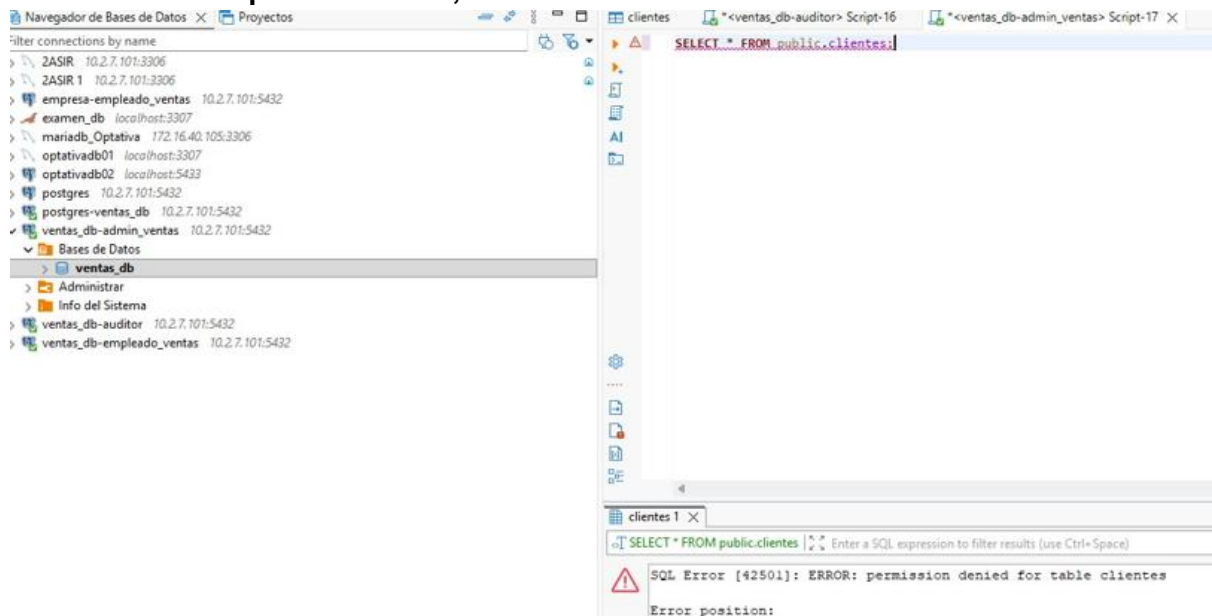
- Elimina los permisos del grupo de ventas:

REVOKE ALL ON public.clientes FROM ventas_grupo;

```
ventas_db=# REVOKE ALL ON public.clientes FROM ventas_grupo;
REVOKE
ventas_db=# |
```

- Observa el efecto

SELECT * FROM public.clientes;



No tiene permiso.

- Añade privilegios de consulta

GRANT SELECT ON public.clientes TO ventas_grupo;

```
ventas_db=# GRANT SELECT ON public.clientes TO ventas_grupo;
GRANT
ventas_db=# |
```

- Comprueba si el permiso vuelve a estar disponible.

Funciona

Navegador de Bases de Datos X
 Proyectos

tr connections by name

- \ 2ASIR 10.2.7.101:3306
- \ 2ASIR 1 10.2.7.101:3306
- \ empresa-empleado_ventas 10.2.7.101:5432
- \ examen_db localhost:3307
- \ mariadb_Optativa 172.16.40.105:3306
- \ optativadb01 localhost:3307
- \ optativadb02 localhost:5433
- \ postgres 10.2.7.101:5432
- \ postgres-ventas_db 10.2.7.101:5432
- \ ventas_db-admin_ventas 10.2.7.101:5432
- ✓ Bases de Datos
 - ventas_db
 - Administrar
 - Info del Sistema
 - ventas_db-auditor 10.2.7.101:5432
 - ventas_db-empleado_ventas 10.2.7.101:5432

ventas_db-admin_ventas public@ventas_db

clientes *<ventas_db-auditor> Script-16 *<ventas_db-admin_ventas> Script-17 X

SELECT * FROM public.clientes;

clientes 1 X

SELECT * FROM public.clientes | Enter a SQL expression to filter results (use Ctrl+Space)

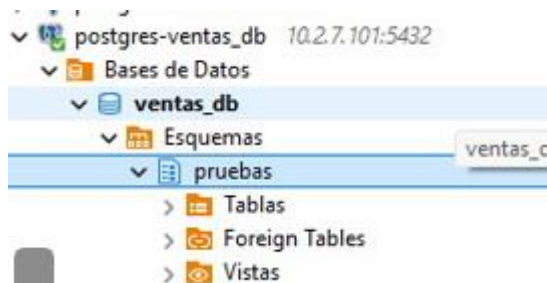
	id	nombre	dni	telefono	email	saldo
1	1	Perro Sanchez	12345678A	600111222	perrete@psoc.com	-9.999.999
2	2	Donaldo Trumpino	98765432B	600333444	trump@usa.com	5.000.000
3	3	Vladimiro Putinino	11223344C	600555666	envladi@ruski.com	1

Privilegios sobre esquemas

- Crea un nuevo esquema llamado pruebas:

CREATE SCHEMA pruebas;

```
GRANT
ventas_db=# CREATE SCHEMA pruebas;
CREATE SCHEMA
```



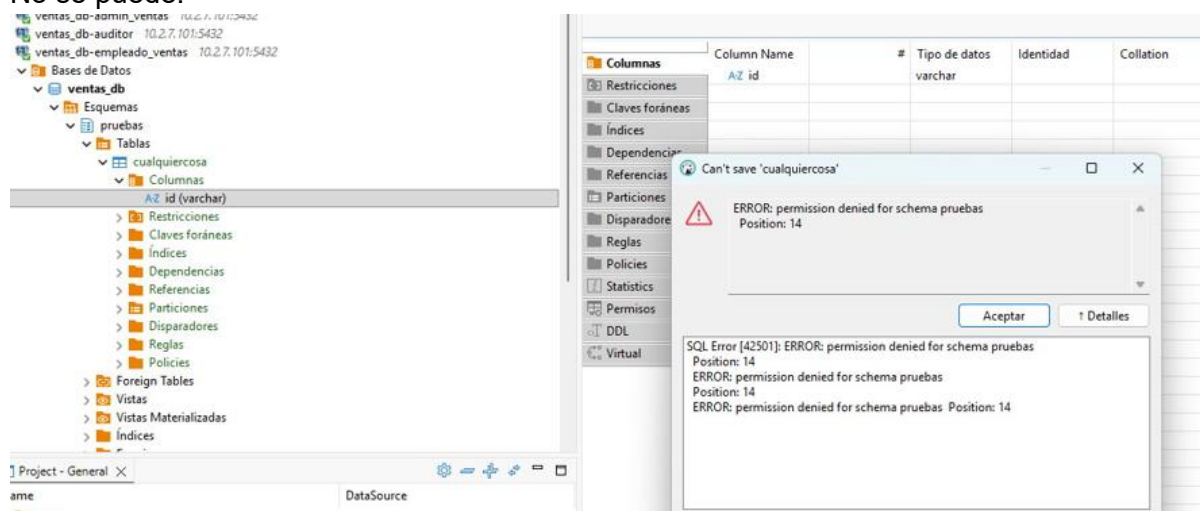
- Concede a **ventas_grupo** permiso para usar el esquema pero no para crear objetos

GRANT USAGE ON SCHEMA pruebas TO ventas_grupo;

```
ventas_db=# GRANT USAGE ON SCHEMA pruebas TO ventas_grupo;
GRANT
```

- Intenta crear una tabla dentro del esquema con **empleado_ventas** y verifica el resultado.

No se puede:



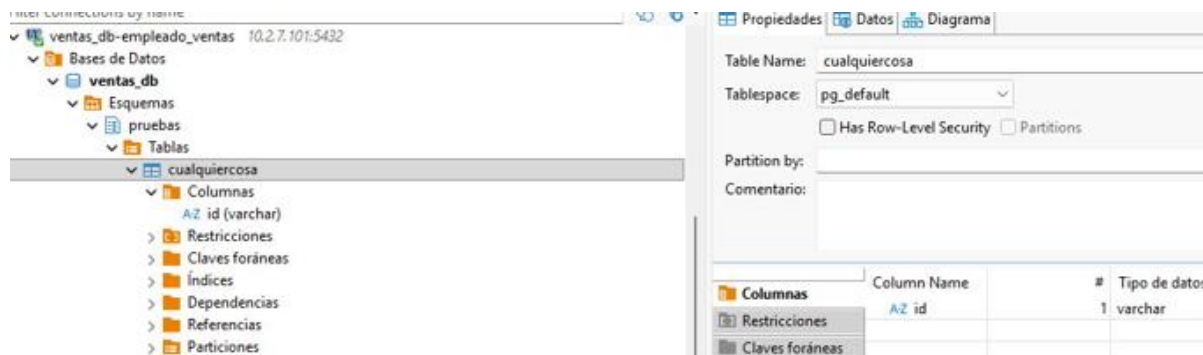
- Da permisos para crear

GRANT CREATE ON SCHEMA pruebas TO ventas_grupo;

```
ventas_db=# GRANT CREATE ON SCHEMA pruebas TO ventas_grupo;
GRANT
ventas_db=#
```

- Vuelve a probar y observa la diferencia.

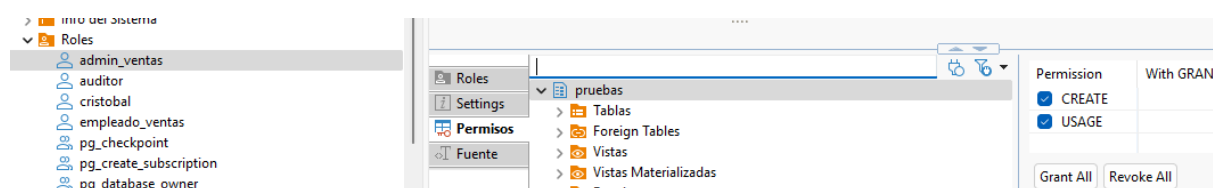
Ahora ya se puede.



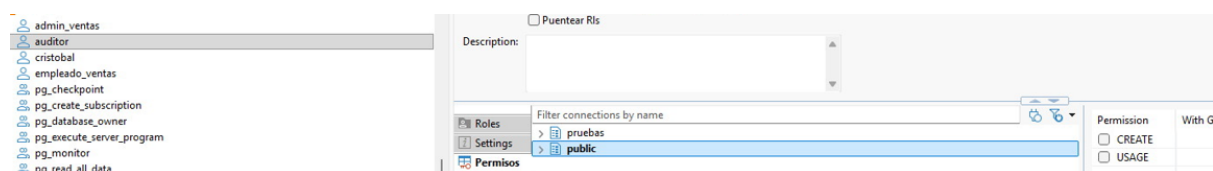
Auditoría final de roles y privilegios

- Consulta todos los privilegios otorgados a cada usuario:

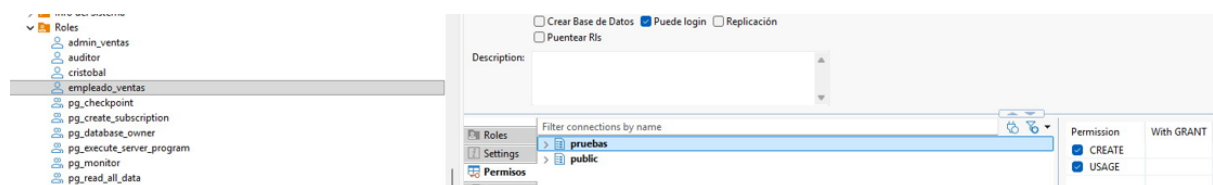
“admin_ventas”



“auditor”



“empleado_ventas”



Actividad 5 – Seguridad y cumplimiento

1. Configuración de políticas de seguridad

- Activa el registro (`logging_collector = on`) y revisa el archivo `postgresql.conf`.

Modificamos el archivo “`postgresql.conf`”. Debemos poner:

`logging_collector = on`

Y activar también las opciones:

`log_connections = on`

`log_disconnections = on`

```
# This is used when logging to stderr:
logging_collector = on # Enable capturing of stderr, jsonlog,
# and csvlog into log files. Required
#log_checkpoints = on
log_connections = on
log_disconnections = on
#log_duration = off
```

- Cambia la política de autenticación de `md5` a `scram-sha-256` en `pg_hba.conf`.

```
# IPv4 local connections:
host all all 127.0.0.1/32 scram-sha-256
host all segurísimo 0.0.0.0/0 scram-sha-256
# IPv6 local connections:
host all all ::1/128 scram-sha-256
# Allow replication connections from localhost, by a user with the
# replication privilege.
local replication all peer
host replication all 127.0.0.1/32 scram-sha-256
host replication all ::1/128 scram-sha-256
host all all 0.0.0.0/0 scram-sha-256
```

2. Auditoría de accesos

- Conéctate con distintos usuarios y revisa los logs.

Para saber dónde están los logs:

Te metes en postgresql

Encuentra el Directorio de Datos (PGDATA): **SHOW data_directory;**

Verifica el Directorio de Logs: **SHOW log_directory;**

Determina el Nombre del Archivo de Log Actual (si el servidor está en ejecución): **SELECT pg_current_logfile();**

```
postgres=# SHOW data_directory;
data_directory
-----
/var/lib/postgresql/16/main
(1 row)

postgres=# SHOW log_directory;
log_directory
-----
log
(1 row)

postgres=# SELECT pg_current_logfile();
pg_current_logfile
-----
log/postgresql-2025-11-14_095314.log
(1 row)
```

“Logeo” exitoso:

```
2025-11-14 10:04:38.632 CET [2748] auditor@ventas_db LOG: connection authorized: user=auditor database=ventas_db SSL enabled (protocol=TLSv1.3, cipher=TLS_AES_256_GCM_SHA384, bits=256)
2025-11-14 10:04:38.655 CET [2749] [unknown]@[unknown] LOG: connection received: host=172.16.40.105 port=63715
2025-11-14 10:04:38.672 CET [2749] auditor@ventas_db LOG: connection authenticated: identity="auditor" method=scram-sha-256 (/etc/postgresql/16/main/pg_hba.conf:150)
2025-11-14 10:04:38.672 CET [2749] auditor@ventas_db LOG: connection authorized: user=auditor database=ventas_db SSL enabled (protocol=TLSv1.3, cipher=TLS_AES_256_GCM_SHA384, bits=256)
2025-11-14 10:04:42.530 CET [2750] [unknown]@[unknown] LOG: connection received: host=172.16.40.105 port=63718
2025-11-14 10:04:42.548 CET [2750] empleado_ventas@ventas_db LOG: connection authenticated: identity="empleado_ventas" method=scram-sha-256 (/etc/postgresql/16/main/pg_hba.conf:150)
2025-11-14 10:04:42.548 CET [2750] empleado_ventas@ventas_db LOG: connection authorized: user=empleado_ventas database=ventas_db SSL enabled (protocol=TLSv1.3, cipher=TLS_AES_256_GCM_SHA384, bits=256)
2025-11-14 10:04:42.577 CET [2751] [unknown]@[unknown] LOG: connection received: host=172.16.40.105 port=63719
2025-11-14 10:04:42.595 CET [2751] empleado_ventas@ventas_db LOG: connection authenticated: identity="empleado_ventas" method=scram-sha-256 (/etc/postgresql/16/main/pg_hba.conf:150)
2025-11-14 10:04:42.595 CET [2751] empleado_ventas@ventas_db LOG: connection authorized: user=empleado_ventas database=ventas_db SSL enabled (protocol=TLSv1.3, cipher=TLS_AES_256_GCM_SHA384, bits=256)
root@buntumysqlsuarez:/home/cristobal#
```

- Identifica intentos fallidos de conexión. Explica que ves

“Logeo” fallido:

```
2025-11-14 10:07:42.661 CET [2829] perro_sanchez@ventas_db FATAL: password authentication failed for user "perro_sanchez"
2025-11-14 10:07:42.661 CET [2829] perro_sanchez@ventas_db DETAIL: Role "perro_sanchez" does not exist.
Connection matched file "/etc/postgresql/16/main/pg_hba.conf" line 150: "host all 0.0.0.0/0 scram-sha-256"
2025-11-14 10:07:44.899 CET [2830] [unknown]@[unknown] LOG: connection received: host=172.16.40.105 port=63765
2025-11-14 10:07:44.916 CET [2830] perro_sanchez@ventas_db FATAL: password authentication failed for user "perro_sanchez"
2025-11-14 10:07:44.916 CET [2830] perro_sanchez@ventas_db DETAIL: Role "perro_sanchez" does not exist.
Connection matched file "/etc/postgresql/16/main/pg_hba.conf" line 150: "host all 0.0.0.0/0 scram-sha-256"
2025-11-14 10:07:46.558 CET [2832] [unknown]@[unknown] LOG: connection received: host=172.16.40.105 port=63766
2025-11-14 10:07:46.575 CET [2832] perro_sanchez@ventas_db FATAL: password authentication failed for user "perro_sanchez"
2025-11-14 10:07:46.575 CET [2832] perro_sanchez@ventas_db DETAIL: Role "perro_sanchez" does not exist.
Connection matched file "/etc/postgresql/16/main/pg_hba.conf" line 150: "host all 0.0.0.0/0 scram-sha-256"
root@buntumysqlsuarez:/home/cristobal#
```

Me indica hora, IP y nombre del usuario que intenta conectarse, también la conexión configurada en “pg_hba.conf” que se le aplicaría.

3. Comprobación de seguridad

- Verifica los privilegios de cada usuario (`\du` y `\z`).

```
postgres=# \du
```

List of roles		
Role name	Attributes	
admin_ventas	Create role, Create DB Password valid until 2026-12-31 00:00:00+01	+
auditor	No inheritance 15 connections	+
cristobal		
empleado_ventas	3 connections	
postgres	Superuser, Create role, Create DB, Replication, Bypass RLS	
segurísimo	Superuser, Create role, Create DB, Replication, Bypass RLS	
solo_lectura	Cannot login	
ventas_acceso	10 connections	
ventas_grupo	Cannot login	

```
postgres=# \z
```

Access privileges					
Schema	Name	Type	Access privileges	Column privileges	Policies
public	clientes	table			
public	clientes_id_seq	sequence			
public	pedidos	table			
public	pedidos_id_seq	sequence			
public	productos	foreign table	postgres=arwdDxt/postgres		

```
(5 rows)
```

- Elabora un informe final indicando qué medidas garantizan la seguridad.
- **Vistas personalizadas:** Cada usuario solo accede a las columnas y datos esenciales para su función, ocultando información sensible (DNI, Email).
- **Revocación de Permisos Directos:** Se fuerza a los usuarios a usar las vistas, asegurando que las políticas de filtrado y exposición de datos se cumplan.
- **Restricción de Conexiones:** Previene el abuso o el uso excesivo de recursos por parte de cuentas individuales.
- **Asignación de Permisos de Manipulación Detallada:** Solo admin_ventas tiene el poder de eliminar registros, mientras que empleado_ventas solo puede insertar/actualizar.
- **Rol solo_lectura:** Centraliza y simplifica la concesión de acceso de solo lectura para la auditoría.
- **Autenticación scram-sha-256:** Mejora la seguridad de las contraseñas almacenadas, haciéndolas menos susceptibles a ataques que la política md5.
- **Registro de Actividad:** Permite detectar y rastrear intentos fallidos, accesos no autorizados y actividades sospechosas, garantizando el cumplimiento.