

Evidencias

El alumno deberá entregar: **CRISTÓBAL SUÁREZ ABAD**

- Sentencias SQL completas y resultado obtenido
- Evidencia con consultas de verificación

Actividad 1 — Gestión de Usuarios y Roles (2 puntos)

1. Crea una base de datos llamada **sistema_ventas** y conéctate a ella.

```
CREATE DATABASE sistema_ventas;
```

```
postgres=# CREATE DATABASE sistema_ventas;
CREATE DATABASE
postgres=# |
```

Nos posicionamos en ella para seguir con la tarea:

```
\c sistema_ventas;
```

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

2. Crea los siguientes usuarios:

Usuario	Requisitos
supervisor_ventas	Contraseña Sup3r#2025, puede crear roles, no puede crear BD, hereda privilegios
asistente_ventas	Contraseña Asist\$2025, límite de 2 sesiones, no puede crear roles ni BD
inspector	Contraseña Inspect#2025, no hereda privilegios, solo 1 sesión

supervisor_ventas:

```
CREATE USER
supervisor_ventas
WITH PASSWORD
'Sup3r#2025'
CREATEROLE
NOCREATEDB
INHERIT;
```

```
sistema_ventas=# CREATE USER supervisor_ventas
WITH PASSWORD 'Sup3r#2025'
    CREATEROLE
    NOCREATEDB
    INHERIT;
CREATE ROLE
sistema_ventas=# |
```

asistente_ventas:

```
CREATE USER asistente_ventas
```

```
WITH PASSWORD 'Asist$2025'
```

```
    CONNECTION LIMIT 2
```

```
    NOCREATEDB
```

```
    NOCREATEROLE;
```

```
CREATE ROLE
sistema_ventas=# CREATE USER asistente_ventas
WITH PASSWORD 'Asist$2025'
    CONNECTION LIMIT 2
    NOCREATEDB
    NOCREATEROLE;
CREATE ROLE
sistema_ventas=#
```

inspector:

```
CREATE USER inspector
```

```
WITH PASSWORD 'Inspect#2025'
```

```
    NOINHERIT
```

```
    CONNECTION LIMIT 1;
```

```
sistema_ventas=# CREATE USER inspector
WITH PASSWORD 'Inspect#2025'
    NOINHERIT
    CONNECTION LIMIT 1;
CREATE ROLE
sistema_ventas=#
```

3. Crea un rol **ventas_equipo** con los siguientes requisitos:

- No inicia sesión
- Hereda permisos
- No crea BD ni roles
- No es superusuario
- No tiene permisos de replicación
- No puede omitir políticas

```
CREATE ROLE ventas_equipo
```

```
WITH NOLOGIN
    INHERIT
    NOCREATEDB
    NOCREATEROLE
    NOREPLICATION
    NOSUPERUSER
    NOBYPASSRLS;
```

```
sistema_ventas=# CREATE ROLE ventas_equipo
WITH NOLOGIN
    INHERIT
    NOCREATEDB
    NOCREATEROLE
    NOREPLICATION
    NOSUPERUSER
    NOBYPASSRLS;
CREATE ROLE
sistema_ventas=# |
```

4. Asocia **supervisor_ventas** y **asistente_ventas** al rol **ventas_equipo**.
El supervisor debe poder administrar permisos del rol; el asistente no.

```
GRANT ventas_equipo TO supervisor_ventas WITH ADMIN OPTION;
```

```
GRANT ventas_equipo TO asistente_ventas;
```

```
sistema_ventas=# GRANT ventas_equipo TO supervisor_ventas WITH ADMIN OPTION;
GRANT ROLE
sistema_ventas=# GRANT ventas_equipo TO asistente_ventas;
GRANT ROLE
sistema_ventas=# |
```

5. Demuestra la pertenencia a los roles con consultas.

6. Con el usuario **asistente_ventas**, crea una tabla llamada **productos** y luego intenta eliminar el usuario desde otro perfil.

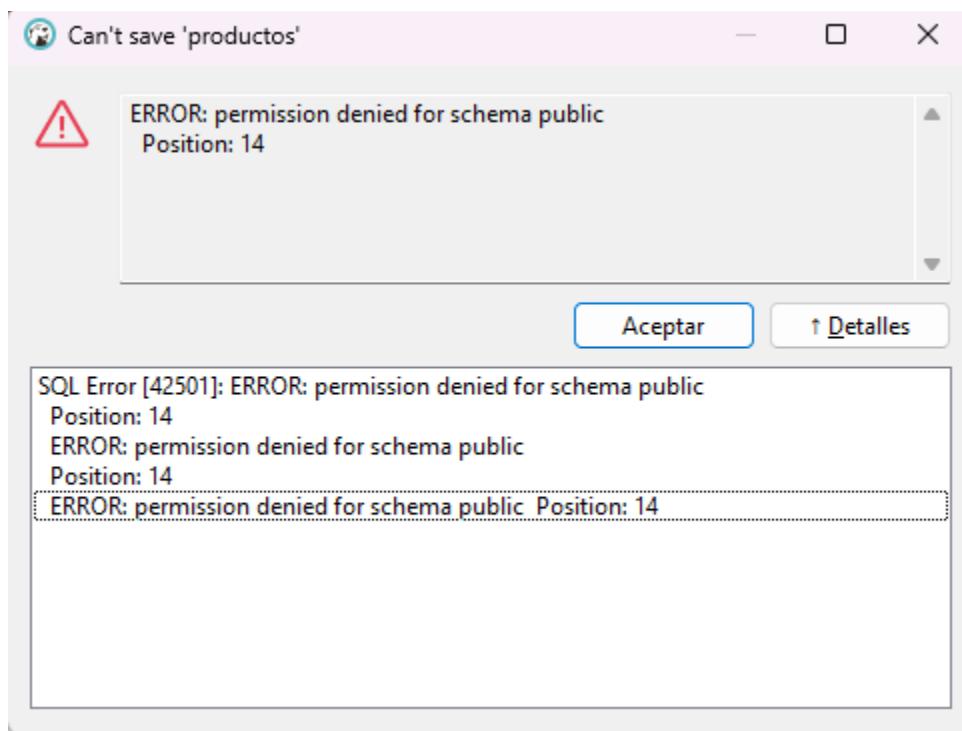
Desde DBeaver nos creamos una conexión con las credenciales del usuario.

```

7. CREATE TABLE public.productos (
8.     id varchar NOT NULL,
9.     nombre varchar NULL,
10.    CONSTRAINT productos_pk PRIMARY KEY (id)
11. );

```

De principio no podemos crear la tabla, porque el usuario no tiene permisos para crear tablas en el esquema público.



Para solucionarlo.

```
GRANT CREATE ON SCHEMA public TO asistente_ventas;
```

```
sistema_ventas=# GRANT CREATE ON SCHEMA public TO asistente_ventas;
GRANT
sistema_ventas=# |
```

Ahora si Podemos:

```
sistema_ventas_ASISTENTE 10.2.7.101:5432
  Bases de Datos
    sistema_ventas
      Esquemas
        public
          Tablas
            productos
              Columnas
                id (varchar)
```

```
sistema_ventas=# \dt
      List of relations
 Schema |     Name      | Type |      Owner
-----+--------------+-----+--------------
 public | productos   | table | asistente_ventas
(1 row)

sistema_ventas=# |
```

Intentamos eliminarlo:

```
DROP ROLE asistente_ventas;
```

- ¿Qué sucede y por qué?

No podemos: Algunos elementos (objetos) dependen de él. Es dueño de algunas tablas

```
sistema_ventas=# DROP ROLE asistente_ventas;
ERROR: role "asistente_ventas" cannot be dropped because some objects depend on it
DETALLE: privileges for schema public
owner of table productos
sistema_ventas=# |
```

- ¿Qué pasos serían necesarios para eliminarlo correctamente?

Quitarle de todo:

```
REVOKE ALL ON ALL TABLES IN SCHEMA public FROM asistente_ventas;
```

```
REVOKE ALL ON ALL SEQUENCES IN SCHEMA public FROM asistente_ventas;
```

```
REVOKE ALL ON ALL FUNCTIONS IN SCHEMA public FROM asistente_ventas;
```

```
REVOKE ALL ON SCHEMA public FROM asistente_ventas;
```

```
sistema_ventas=# REVOKE ALL ON ALL TABLES IN SCHEMA public FROM asistente_ventas;
REVOKE
sistema_ventas=# REVOKE ALL ON ALL SEQUENCES IN SCHEMA public FROM asistente_ventas;
REVOKE
sistema_ventas=# REVOKE ALL ON ALL FUNCTIONS IN SCHEMA public FROM asistente_ventas;
REVOKE
sistema_ventas=# REVOKE ALL ON SCHEMA public FROM asistente_ventas;
REVOKE
sistema_ventas=# |
```

Nos puede seguir danto error porque es dueño de una tabla

```
sistema_ventas=# DROP ROLE asistente_ventas;
ERROR:  role "asistente_ventas" cannot be dropped because some objects depend on it
DETAIL:  owner of table productos
```

Para ello cambiamos el propietario de la tabla:

```
ALTER TABLE public.productos OWNER TO supervisor_ventas;
```

Ahora si nos deja borrar al usuario.

```
sistema_ventas=# ALTER TABLE public.productos OWNER TO supervisor_ventas;
ALTER TABLE
sistema_ventas=# DROP ROLE asistente_ventas;
DROP ROLE
sistema_ventas=# |
```

- ¿Qué riesgos implicaría forzar su eliminación?

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

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

```
DROP ROLE
sistema_ventas=# \dt
      List of relations
 Schema |     Name      |   Type   |      Owner
-----+---------------+----------+--------------
 public | productos    | table    | supervisor_ventas
(1 row)

sistema_ventas=# |
```

Actividad 2 — Vistas y Accesos Controlados (2 puntos)

1. Crea las tablas e inserta datos:

LO VAMOS A HACER DESDE EL USUARIO “postgres”.

clientes(id, nombre, telefono, email, saldo, vip boolean)

```
CREATE TABLE clientes (
    id SERIAL PRIMARY KEY,
    nombre VARCHAR(100),
    telefono VARCHAR(20),
    email VARCHAR(100),
    saldo NUMERIC(10, 2) DEFAULT 0.00,
    vip BOOLEAN DEFAULT FALSE
);
```

```
sistema_ventas=# CREATE TABLE clientes (
    id SERIAL PRIMARY KEY,
    nombre VARCHAR(100),
    telefono VARCHAR(20),
    email VARCHAR(100),
    saldo NUMERIC(10, 2) DEFAULT 0.00,
    vip BOOLEAN DEFAULT FALSE
);
CREATE TABLE
sistema_ventas=# \DT
sistema_ventas=# \dt
              List of relations
 Schema |     Name      | Type |  Owner
-----+---------------+-----+---------
 public | clientes    | table| postgres
 public | productos   | table| supervisor_ventas
(2 rows)

sistema_ventas=# |
```

```
ventas(id, id_cliente, fecha, total, estado)
```

Para poner foreing key: lo del INTEGER REFERENCES

<https://www.postgresql.org/docs/current/tutorial-fk.html>

```
CREATE TABLE ventas (
    id SERIAL PRIMARY KEY,
    id_cliente INTEGER REFERENCES clientes(id),
    fecha DATE DEFAULT CURRENT_DATE,
    total NUMERIC(10, 2),
    estado VARCHAR(50)
);
```

```
sistema_ventas=# CREATE TABLE ventas (
    id SERIAL PRIMARY KEY,
    id_cliente INTEGER REFERENCES clientes(id),
    fecha DATE DEFAULT CURRENT_DATE,
    total NUMERIC(10, 2),
    estado VARCHAR(50)
);
CREATE TABLE
sistema_ventas=# \dt
      List of relations
 Schema |   Name    | Type  |          Owner
-----+-----------+-----+-----
 public | clientes | table | postgres
 public | productos | table | supervisor_ventas
 public | ventas   | table | postgres
(3 rows)

sistema_ventas=# |
```

2. Inserta al menos 3 clientes y 3 ventas de ejemplo.

```
INSERT INTO clientes (id, nombre, telefono, email, saldo, vip) VALUES
('12345677', 'Perro Sanchez', '600111222', 'perrete@psoe.com', -9999999.00, TRUE),
('12345678', 'Donaldinho Trompino', '600333444', 'trump@usa.com', 5000000.00,
FALSE),
('12345679', 'Vladimiro Putinesco', '600555666', 'ervladi@ruski.com', 1.00, FALSE);
```

```
sistema_ventas=# INSERT INTO clientes (id, nombre, telefono, email, saldo, vip) VALUES
('12345677', 'Perro Sanchez', '600111222', 'perrete@psoe.com', -9999999.00, TRUE),
('12345678', 'Donaldinho Trompino', '600333444', 'trump@usa.com', 5000000.00, FALSE),
('12345679', 'Vladimiro Putinesco', '600555666', 'ervladi@ruski.com', 1.00, FALSE);
INSERT 0 3
sistema_ventas=# SELECT * FROM clientes;
```

```
sistema_ventas=# SELECT * FROM clientes;
+----+-----+-----+-----+-----+-----+
| id | nombre | telefono | email | saldo | vip |
+----+-----+-----+-----+-----+
| 12345677 | Perro Sanchez | 600111222 | perrete@psoe.com | -9999999.00 | t |
| 12345678 | Donaldinho Trompino | 600333444 | trump@usa.com | 5000000.00 | f |
| 12345679 | Vladimiro Putinesco | 600555666 | ervladi@ruski.com | 1.00 | f |
+----+-----+-----+-----+-----+
(3 rows)

sistema_ventas=# |
```

```
INSERT INTO ventas (id, id_cliente, total, estado) VALUES
(1, 12345677, 300.00, 'Entregado'),
(2, 12345678, 50.50, 'Pendiente'),
(3, 12345677, 150.00, 'Entregado'),
(4, 12345679, 400.00, 'Entregado');
```

```
sistema_ventas=# INSERT INTO ventas (id, id_cliente, total, estado) VALUES
(1, 12345677, 300.00, 'Entregado'),
(2, 12345678, 50.50, 'Pendiente'),
(3, 12345677, 150.00, 'Entregado'),
(4, 12345679, 400.00, 'Entregado');
INSERT 0 4
sistema_ventas=# |
```

```
INSERT 0 4
sistema_ventas=# SELECT * FROM ventas;
+----+-----+-----+-----+-----+
| id | id_cliente | fecha | total | estado |
+----+-----+-----+-----+
| 1 | 12345677 | 2025-11-28 | 300.00 | Entregado |
| 2 | 12345678 | 2025-11-28 | 50.50 | Pendiente |
| 3 | 12345677 | 2025-11-28 | 150.00 | Entregado |
| 4 | 12345679 | 2025-11-28 | 400.00 | Entregado |
+----+-----+-----+-----+
(4 rows)

sistema_ventas=# |
```

- Crea las siguientes vistas:

Vista	Contenido	Quién accede
vista_admin_clientes	Todos los datos + suma total gastado por cliente	supervisor_ventas
vista_asistente_clientes	Nombre, teléfono, saldo, vip (sin email)	asistente_ventas
vista_inspector_anonima	Solo totales por cliente sin información personal	inspector

vista_admin_clientes:

```
CREATE VIEW vista_admin_clientes AS
SELECT c.*,
COALESCE(SUM(v.total),0.00)
AS total_gastado FROM clientes c LEFT JOIN ventas v ON c.id = v.id_cliente
GROUP BY c.id
ORDER BY c.id;
```

```
sistema_ventas=# SELECT c.*,
COALESCE(SUM(v.total),0.00)
AS total_gastado FROM clientes c LEFT JOIN ventas v ON c.id = v.id_cliente
GROUP BY c.id
ORDER BY c.id;
+-----+-----+-----+-----+-----+-----+-----+
| id   | nombre | telefono | email    | saldo   | vip   | total_gastado |
+-----+-----+-----+-----+-----+-----+-----+
| 12345677 | Perro Sanchez | 600111222 | perrete@psoe.com | -9999999.00 | t | 450.00
| 12345678 | Donaldinho Trompino | 600333444 | trump@usa.com | 5000000.00 | f | 50.50
| 12345679 | Vladimiro Putinesco | 600555666 | ervladi@ruski.com | 1.00 | f | 400.00
(3 rows)
```

```
sistema_ventas=# CREATE VIEW vista_admin_clientes AS
SELECT c.*,
COALESCE(SUM(v.total),0.00)
AS total_gastado FROM clientes c LEFT JOIN ventas v ON c.id = v.id_cliente
GROUP BY c.id
ORDER BY c.id;
CREATE VIEW
sistema_ventas=# |
```

vista_asistente_clientes:

```
CREATE VIEW vista_asistente_clientes AS
SELECT nombre,
telefono,
saldo,
vip FROM clientes;
```

```
sistema_ventas=# CREATE VIEW vista_asistente_clientes AS
    SELECT nombre,
    telefono,
    saldo,
    vip FROM clientes;
CREATE VIEW
sistema_ventas=# |
```

```
CREATE VIEW
sistema_ventas=#           SELECT nombre,
    telefono,
    saldo,
    vip FROM clientes;
    nombre      | telefono | saldo   | vip
-----+-----+-----+-----+
 Perro Sanchez | 600111222 | -9999999.00 | t
 Donaldinho Trompino | 600333444 | 5000000.00 | f
 Vladimiro Putinesco | 600555666 |       1.00 | f
(3 rows)

sistema_ventas=# |
```

vista_inspector_anonima:

```
CREATE VIEW vista_inspector_anonima AS
    SELECT id_cliente,
    SUM(total) AS total_compras FROM ventas GROUP BY id_cliente;
```

```
sistema_ventas=# CREATE VIEW vista_inspector_anonima AS
    SELECT id_cliente,
    SUM(total) AS total_compras FROM ventas GROUP BY id_cliente;
CREATE VIEW
sistema_ventas=# |
```

```
sistema_ventas=#           SELECT id_cliente,
    SUM(total) AS total_compras FROM ventas GROUP BY id_cliente;
    id_cliente | total_compras
-----+-----
 12345677 |      450.00
 12345679 |      400.00
 12345678 |      50.50
(3 rows)
```

3. Revoca permisos directos sobre las tablas base a todos los usuarios excepto el superusuario.

Concede permisos solamente a través de vistas.

QUITAR PERMISOS:

```
REVOKE ALL ON clientes FROM supervisor_ventas;  
REVOKE ALL ON ventas FROM supervisor_ventas;  
REVOKE ALL ON clientes FROM inspector;  
REVOKE ALL ON ventas FROM inspector;
```

asistente_ventas UNA VEZ QUE SE BORRÓ NO HA SIDO CREADO DE NUEVO. DE TODAS MANERAS:

```
REVOKE ALL ON clientes FROM asistente_ventas;  
REVOKE ALL ON ventas FROM asistente_ventas;
```

```
sistema_ventas=# REVOKE ALL ON clientes FROM supervisor_ventas;  
REVOKE  
sistema_ventas=# REVOKE ALL ON pedidos FROM supervisor_ventas;  
ERROR: relation "pedidos" does not exist  
sistema_ventas=# \DT  
invalid command \DT  
Try \? for help.  
sistema_ventas=# \dt  
      List of relations  
 Schema |   Name    | Type | Owner  
-----+-----+-----+  
 public | clientes | table | postgres  
 public | productos | table | supervisor_ventas  
 public | ventas | table | postgres  
(3 rows)  
  
sistema_ventas=# REVOKE ALL ON ventas FROM supervisor_ventas;  
REVOKE  
sistema_ventas=# REVOKE ALL ON clientes FROM inspector;  
REVOKE  
sistema_ventas=# REVOKE ALL ON ventas FROM inspector;  
REVOKE  
sistema_ventas=# REVOKE ALL ON clientes FROM asistente_ventas;  
REVOKE  
sistema_ventas=# REVOKE ALL ON ventas FROM asistente_ventas;  
REVOKE  
sistema_ventas=# |
```

CONCEDER PERMISOS:

```
GRANT SELECT ON vista_admin_clientes TO supervisor_ventas;  
GRANT SELECT ON vista_asistente_clientes TO asistente_ventas;  
GRANT SELECT ON vista_inspector_anonima TO inspector;
```

```

sistema_ventas=# GRANT SELECT ON vista_admin_clientes TO supervisor_ventas;
GRANT
sistema_ventas=# GRANT SELECT ON vista_asistente_clientes TO asistente_ventas;
GRANT
sistema_ventas=# GRANT SELECT ON vista_inspector_anonima TO inspector;
GRANT
sistema_ventas=#

```

4. Demuestra las diferencias al consultar con cada usuario.

Desde asistente_ventas

SELECT * from vista_asistente_clientes;

	AZ nombre	AZ telefono	123 saldo	vip
1	Perro Sanchez	600111222	-9.999.999	[v]
2	Donaldinho Trompino	600333444	5.000.000	[]
3	Vladimiro Putinesco	600555666	1	[]

Desde supervisor_ventas:

select * from vista_admin_clientes;

123 id	AZ nombre	AZ telefono	AZ email	123 saldo	vip	123 total_gastado	
1	12.345.677	Perro Sanchez	600111222	perrete@psoe.com	-9.999.999	[v]	450
2	12.345.678	Donaldinho Trompino	600333444	trump@usa.com	5.000.000	[]	50,5
3	12.345.679	Vladimiro Putinesco	600555666	ervladi@ruski.com	1	[]	400

Desde inspector:

Puede que haya un problema con el límites de conexiones de inspector al usar DBeaver:

```

ALTER ROLE
sistema_ventas=# ALTER USER inspector CONNECTION LIMIT 8;
ALTER ROLE
sistema_ventas=#

```

```
select * from vista_inspector_anonima;
```

id_cliente	total_compras
12345.677	450
12345.679	400
12345.678	50,5

Actividad 3 — Auditoría, Seguridad y Políticas (2 puntos)

1. Crea un rol adicional llamado **solo_lectura_global** y:

- Dale acceso de *solo lectura* a todas las tablas y vistas existentes.

```
CREATE ROLE solo_lectura_global NOLOGIN;
```

```
ALTER ROLE
sistema_ventas=# CREATE ROLE solo_lectura_global NOLOGIN;
CREATE ROLE
sistema_ventas=#

```

GRANT SELECT ON ALL TABLES IN SCHEMA public TO solo_lectura_global;

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

GRANT USAGE ON SCHEMA public TO solo_lectura_global;

```
CREATE ROLE
sistema_ventas=# GRANT SELECT ON ALL TABLES IN SCHEMA public TO solo_lectura_global;
GRANT
sistema_ventas=# GRANT USAGE ON SCHEMA public TO solo_lectura_global;
GRANT
sistema_ventas=#

```

- Haz que el usuario inspector herede este rol.

¿Te acuerdas que inspector tiene noinherit?

```
ALTER USER inspector INHERIT;
```

```
GRANT solo_lectura_global TO inspector;
```

```
GRANT
sistema_ventas=# GRANT solo_lectura_global TO inspector;
GRANT ROLE
sistema_ventas=# |
```

- Asegura que cualquier objeto futuro creado en el esquema sea accesible en lectura por este rol.

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

```
sistema_ventas=# ALTER DEFAULT PRIVILEGES IN SCHEMA public GRANT SELECT ON TABLES TO solo_lectura_global;
ALTER DEFAULT PRIVILEGES
sistema_ventas=# |
```

Actividad 4 — Práctica de Revocación y Cambios de Privilegios (2 puntos)

1. Sobre la tabla clientes:

- Concede a **ventas_equipo** permisos de INSERT y SELECT.

GRANT SELECT ON public.clientes TO ventas_equipo;

GRANT INSERT ON public.clientes TO ventas_equipo;

```
root@ubuntumysqlsuarez:/home/ + ^
sistema_ventas=# GRANT SELECT ON public.clientes TO ventas_equipo;
GRANT
sistema_ventas=# GRANT INSERT ON public.clientes TO ventas_equipo;
GRANT
sistema_ventas=# |
```

- Concede a **supervisor_ventas** permiso de DELETE.

GRANT DELETE ON public.clientes TO supervisor_ventas;

```
sistema_ventas=# GRANT DELETE ON public.clientes TO supervisor_ventas;
GRANT
sistema_ventas=# |
```

- Revoca explícitamente DELETE a **asistente_ventas**.

REVOKE DELETE ON public.clientes FROM **asistente_ventas**;

```
sistema_ventas=# REVOKE DELETE ON public.clientes FROM asistente_ventas;  
REVOKE  
sistema_ventas=# |
```

2. Luego:

- Revoca TODOS los permisos del rol **ventas_equipo** sobre clientes.

REVOKE ALL ON clientes FROM **ventas_equipo**;

```
REVOKE  
sistema_ventas=# REVOKE ALL ON clientes FROM ventas_equipo;  
REVOKE  
sistema_ventas=# |
```

- Vuelve a conceder solo SELECT.

GRANT SELECT ON public.clientes TO **ventas_equipo**;

```
sistema_ventas=# GRANT SELECT ON public.clientes TO ventas_equipo;  
GRANT  
sistema_ventas=# |
```

Actividad 5 — Informe de Seguridad (2 puntos)

1. Modifica autenticación en pg_hba.conf para usar **scram-sha-256**.
Activa el log y registra intentos de conexión fallidos.

nano /etc/postgresql/16/main/pg_hba.conf

```
# "local" is for Unix domain socket connections only
local  all      all                                     peer
# IPv4 local connections:
host   all      all          127.0.0.1/32            scram-sha-256
host   all      segurisimo  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
```

nano /etc/postgresql/16/main/postgresql.conf

Activamos loggin_collector

```
# This is used when logging to stderr:
logging_collector = on                                # Enable capturing of stderr, jsonlog,
                                                       # and csvlog into log files. Required
                                                       # to be on for csvlogs and jsonlogs
```

Y después

```
#log_checkpoints = on
log_connections = on
log_disconnections = on
#log_directory = 'log'
```

Algunos compañeros han dicho que han tenido que “descomentar” las siguientes líneas para que les funcione el registro. A mi no me ha hecho falta.

```
# These are only used if logging_collector is on:
#log_directory = 'log'                                # directory where log files are written,
                                                       # can be absolute or relative to PGDATA
#log_filename = 'postgresql-%Y-%m-%d_%H%M%S.log'      # log file name pattern,
                                                       # can include strftime() escapes
#log_min_messages = 'warning'
```

Entrega:

- Captura de logs con fallos:

Usuario sin credenciales intenta “logearse”.

```
ge=0.002 s; distance=16 kB, estimate=755 kB; lsn=0/42103D0, redo lsn=0/4210398
2025-11-28 11:10:37.643 CET [6507] [unknown]@[unknown] LOG: connection received: host=172.16.40.105 port=64815
2025-11-28 11:10:37.663 CET [6507] andres_el_hacker@sistema_ventas FATAL: password authentication failed for user "andres_el_hacker"
2025-11-28 11:10:37.663 CET [6507] andres_el_hacker@sistema_ventas DETAIL: Role "andres_el_hacker" does not exist.
      Connection matched file "/etc/postgresql/16/main/pg_hba.conf" line 150: "host      all          all      0.0.0.0/0           scram-sha-256"
root@ubuntuvm:suarez:/home/cristobal#
```

- Verifica los privilegios de cada usuario

En la tabla clientes:

```
sistema_ventas=# \dp *.*  
sistema_ventas=# \dp clientes;  
                                         Access privileges  
Schema |   Name    | Type |          Access privileges          | Column privileges | Policies  
-----+-----+-----+-----+-----+-----+-----+  
public | clientes | table | postgres=arwdDxt/postgres      +|                   |  
           |           |       | solo_lectura_global=r/postgres+|                   |  
           |           |       | supervisor_ventas=d/postgres +|                   |  
           |           |       | ventas_equipo=r/postgres    |                   |  
(1 row)
```

En la tabla ventas:

```
sistema_ventas=# \dp ventas;
                                         Access privileges
Schema |   Name   | Type  |          Access privileges          | Column privileges | Policies
-----+-----+-----+-----+
public | ventas | table | postgres=arwdDxt/postgres      +|                   |
                  |           |             solo_lectura_global=r/postgres |                   |
(1 row)

sistema_ventas=#
```

Desde DBeaver:

El usuario supervisor_ventas: Solo tiene permisos en clientes y es para "Borrar" del

The screenshot shows the DBeaver interface with the following details:

- Left Panel (Roles):** A tree view of PostgreSQL roles. The 'supervisor_ventas' role is highlighted with a gray bar at the bottom.
- Central Panel (Permissions):** The 'Permisos' tab is selected. The 'public' schema is expanded, showing the 'Tablas' (Tables) node. The 'clientes' table is selected and highlighted with a blue bar.
- Right Panel (Permissions List):** A table listing permissions for the selected table ('public.clientes').

Permission	With GRANT	With Hierarchy
<input type="checkbox"/> SELECT		
<input type="checkbox"/> INSERT		
<input type="checkbox"/> UPDATE		
<input checked="" type="checkbox"/> DELETE		
<input type="checkbox"/> TRUNCATE		
<input type="checkbox"/> REFERENCES		
<input type="checkbox"/> TRIGGER		
<input type="checkbox"/> MAINTAIN		
- Bottom Panel:** Buttons for 'Grant All' and 'Revoke All' and the text 'public.clientes'.

asistente_ventas no tiene ningún privilegio:

The screenshot shows the pgAdmin interface for managing PostgreSQL roles. On the left, a tree view lists various database objects and roles. The 'Roles' section is expanded, showing 'admin_ventas' and 'asistente_ventas'. The 'asistente_ventas' node is selected and highlighted in grey. To the right, a detailed configuration window is open for this role.

Role Configuration (Top Panel):

- Super Usuario
- Heredar
- Crear Rol
- Crear Base de Datos
- Puede login
- Replicación
- Puenteear Rls

Description: [Empty text area]

Permissions (Left Panel):

- Roles
- Settings
- Permisos** (Selected)
- Fuente

Filter connections by name

- public
 - Tablas
 - clientes
 - productos
 - ventas** (Selected)
 - Foreign Tables
 - Vistas
 - Vistas Materializadas
 - Funciones
 - Secuencias

Permissions (Right Panel):

Permission	With GI
<input type="checkbox"/> SELECT	
<input type="checkbox"/> INSERT	
<input type="checkbox"/> UPDATE	
<input type="checkbox"/> DELETE	
<input type="checkbox"/> TRUNCATE	
<input type="checkbox"/> REFERENCES	
<input type="checkbox"/> TRIGGER	
<input type="checkbox"/> MAINTAIN	

Buttons:

- Grant All
- Revoke All

public.ventas

inspector: no tiene ningún privilegio.

The screenshot shows the pgAdmin interface for managing database permissions. On the left, a sidebar lists various PostgreSQL roles. The 'inspector' role is selected and highlighted in grey. The main panel displays the 'Permisos' (Permissions) tab for the 'public' schema. A tree view shows the 'ventas' table under the 'Tablas' (Tables) node. To the right, a detailed permissions grid is shown for the 'ventas' table, listing various privileges like SELECT, INSERT, UPDATE, etc., with checkboxes for granting or revoking them. Buttons for 'Grant All' and 'Revoke All' are also present.

Permission	With GRAN
<input type="checkbox"/> SELECT	
<input type="checkbox"/> INSERT	
<input type="checkbox"/> UPDATE	
<input type="checkbox"/> DELETE	
<input type="checkbox"/> TRUNCATE	
<input type="checkbox"/> REFERENCES	
<input type="checkbox"/> TRIGGER	
<input type="checkbox"/> MAINTAIN	

Grant All Revoke All

public.ventas

Rol ventas_equipo: Puede hacer Select en “clientes”.

The screenshot shows the pgAdmin interface for managing database roles. On the left, a sidebar lists various PostgreSQL system roles and user-defined roles. The role 'ventas_equipo' is selected and highlighted with a blue background. The main pane displays the permissions assigned to this role. Under the 'Permisos' tab, the 'public' schema is selected, and within it, the 'Tablas' (Tables) node is expanded. The table 'clientes' is selected, and its permissions are shown in the list on the right. The 'Permission' column contains checkboxes, and the 'With G' column indicates which grants apply to the global scope. The 'clients' table has the 'SELECT' permission granted.

Permission	With G
<input checked="" type="checkbox"/> SELECT	
<input type="checkbox"/> INSERT	
<input type="checkbox"/> UPDATE	
<input type="checkbox"/> DELETE	
<input type="checkbox"/> TRUNCATE	
<input type="checkbox"/> REFERENCES	
<input type="checkbox"/> TRIGGER	
<input type="checkbox"/> MAINTAIN	

Grant All | Revoke All
public.clientes

solo_lectura_global: Tiene Select en todas las tablas.

The screenshot shows the pgAdmin interface for managing database permissions. On the left, a sidebar lists various database objects and roles. The 'solo_lectura_global' role is selected and highlighted in grey. The main pane displays the 'public' schema's permission settings. Under the 'Tablas' section, the 'clientes', 'productos', and 'ventas' tables are listed. Under the 'Vistas' section, three views are listed: 'vista_admin_clientes', 'vista_asistente_clientes', and 'vista_inspector_anonima'. A 'Foreign Tables' section is also visible. On the right, a detailed permissions table is shown, with the 'SELECT' checkbox checked for all objects. Other checkboxes for INSERT, UPDATE, DELETE, TRUNCATE, REFERENCES, TRIGGER, and MAINTAIN are unselected. Buttons for 'Grant All' and 'Revoke All' are at the bottom of the table.

Permission	With GRANT	With Hierarchy
<input checked="" type="checkbox"/> SELECT		
<input type="checkbox"/> INSERT		
<input type="checkbox"/> UPDATE		
<input type="checkbox"/> DELETE		
<input type="checkbox"/> TRUNCATE		
<input type="checkbox"/> REFERENCES		
<input type="checkbox"/> TRIGGER		
<input type="checkbox"/> MAINTAIN		