**PostgretSQL**

1. **Estructura básica de trabajo**

BEGIN;

CREATE TABLE IF NOT EXISTS public.categorias

(

idcategoria integer NOT NULL DEFAULT nextval('"categorias\_idCategoria\_seq"'::regclass),

nombre character(150) COLLATE pg\_catalog."default" NOT NULL,

estado character(1) COLLATE pg\_catalog."default" NOT NULL DEFAULT 'A'::bpchar,

CONSTRAINT pk\_categorias\_idcategoria PRIMARY KEY (idcategoria),

CONSTRAINT "UC\_categorias\_nombre" UNIQUE (nombre)

);

CREATE TABLE IF NOT EXISTS public.productos

(

"idProd" serial NOT NULL,

nombre character(200) COLLATE pg\_catalog."default" NOT NULL,

stock numeric(5, 0) NOT NULL DEFAULT 0,

precio numeric(10, 2) NOT NULL DEFAULT 0,

estado character(1) COLLATE pg\_catalog."default" NOT NULL DEFAULT 'A'::bpchar,

idcategoria integer NOT NULL,

"valorTotal" numeric(10, 2),

CONSTRAINT pk\_productos\_idprod PRIMARY KEY ("idProd"),

CONSTRAINT "UC\_productos\_nombre" UNIQUE (nombre)

);

ALTER TABLE IF EXISTS public.productos

ADD CONSTRAINT fk\_productos\_idcategoria FOREIGN KEY (idcategoria)

REFERENCES public.categorias (idcategoria) MATCH SIMPLE

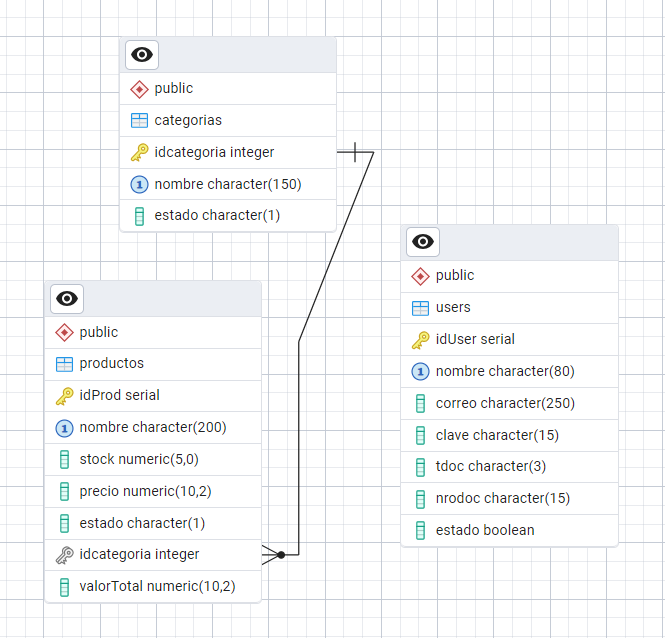
ON UPDATE NO ACTION

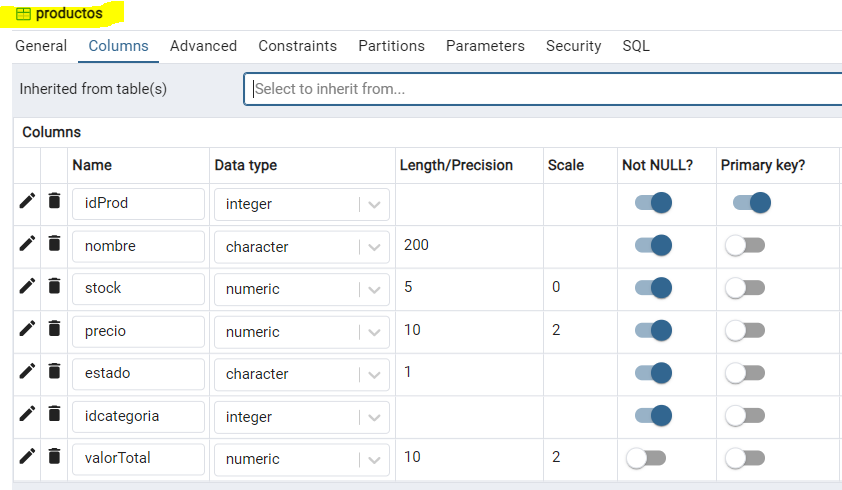
ON DELETE NO ACTION

NOT VALID;

END;

**Modelo Lógico de la BD PostgreSQL**

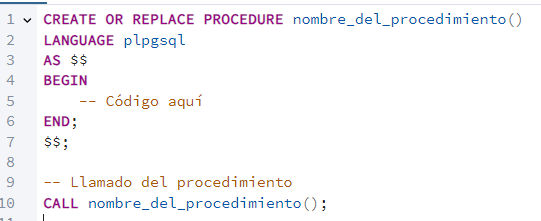




Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

1. **Sintax de Procedimiento Almacenado**



1. **Sintax de la Función**

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

1. **PostgreSQL diferencia entre un Procedimiento Almacenado y una Función**

**Procedimiento Almacenado**

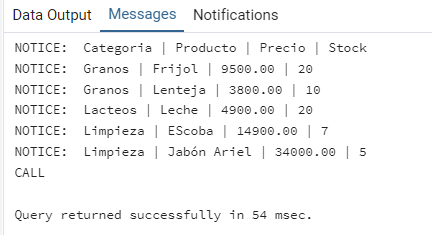
* **Propósito:** Los procedimientos se utilizan generalmente para realizar operaciones de cambio o procesamiento de datos sin la intención de devolver un valor.
* **Uso**: Son adecuados para tareas como actualización de datos, eliminación de registros, procesamiento en segundo plano, o incluso interacciones complejas entre varias tablas.
* **Resultado:** No devuelven un valor directamente (aunque pueden usar OUT o cursores para manejar datos), y no se pueden usar en una consulta SQL como parte de un SELECT.

**Funciones**

* **Propósito:** Las funciones se diseñan para calcular y devolver un valor (o un conjunto de resultados), y se pueden utilizar directamente en consultas SQL como parte de un SELECT.
* **Uso:** Son adecuadas para tareas donde se necesite devolver un resultado, realizar cálculos, transformar datos o hacer consultas repetitivas.
* **Resultado:** Devuelven un valor o un conjunto de resultados que se puede utilizar dentro de una consulta, y pueden ser invocadas en cualquier parte de una sentencia SQL.

1. **Ejercicio:** Realiza un procedimiento almacenado que muestre por Categoría, los productos asociados, mostrando el nombre de la categoría, el nombre del producto, el precio y el stock, para productos y categorías activos





**Encabezado con RAISE NOTICE**:

Imprime un encabezado en la consola para organizar los resultados (Categoría | Producto | Precio | Stock).

**Bucle FOR ... IN**:

Ejecuta una consulta SQL para seleccionar las categorías, productos, precios, y stock e Itera sobre cada fila del resultado, asignando los valores a las variables locales categoria\_nombre, producto\_nombre, precio, y stock.

El bucle FOR ... IN ... LOOP en PL/pgSQL recorre cada fila resultante de la consulta SQL indicada en el SELECT. Por cada fila, asigna los valores de las columnas seleccionadas a las variables especificadas, y luego ejecuta el código dentro del bloque LOOP.

**Salida de Datos con RAISE NOTICE**:

Imprime cada fila en formato legible (% | % | % | %), sustituyendo los valores de las variables en la cadena.

% es un marcador de posición, y las variables (categoria\_nombre, producto\_nombre, precio, stock) se insertan en el orden indicado.

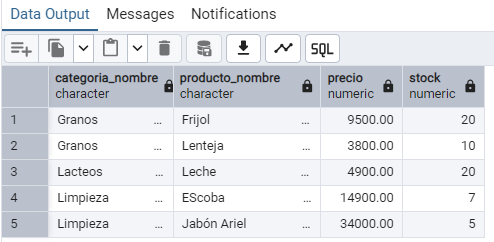
NOTA: Sin embargo, esta no es la forma más presentable, por lo que se recomienda evaluar a través de las funciones.

**Haciendo uso de Funtion**

Texto

El contenido generado por IA puede ser incorrecto.

Resultado mostrado:



1. **Ejercicio:** crear un procedimiento almacenado, donde tenga como parámetro de entrada el nombre de la categoría y un porcentaje de aumento, y cambie el precio de los productos que pertenecen a esa categoría teniendo en cuenta el precio registrado + el porcentaje de aumento, y me muestre los cambios realizados, nombre del producto, precio inicial, monto de aumento, precio cambiado.

Datos de entrada = Categoria %aumento (5)

Precio = precio





Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

El tipo RECORD es un tipo de datos especial que puede contener una fila de cualquier tipo, lo que significa que no se especifica una estructura fija de columnas al momento de la declaración. Esto hace que sea flexible y útil cuando no se sabe de antemano qué columnas se devolverán en una consulta o cuando la consulta tiene una estructura dinámica.

1. **Realizar el ejercicio anterior haciendo uso de function**.

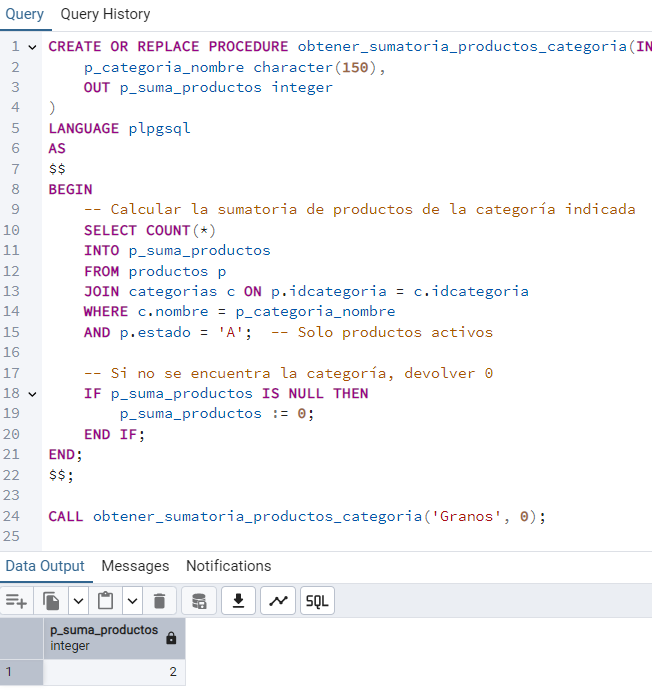




Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

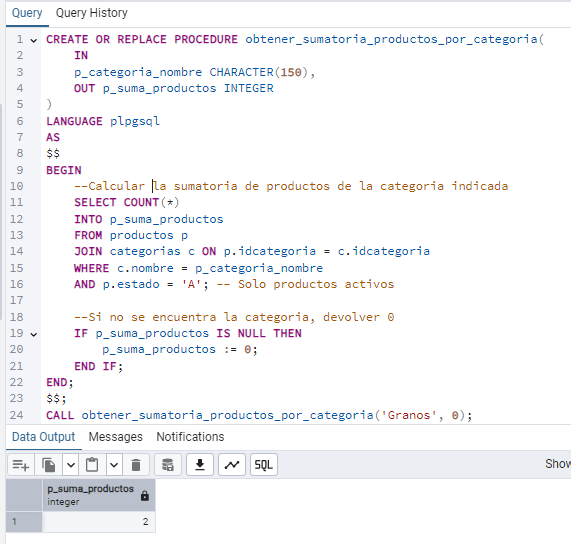
El contenido generado por IA puede ser incorrecto.

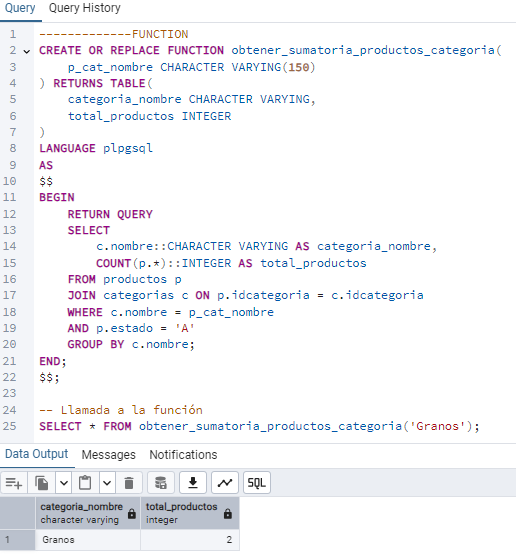
1. **Ejercicio:** crear un procedimiento almacenado en PostgreSQL que reciba el nombre de una categoría como parámetro y devuelva la sumatoria de productos de esa categoría, puedes usar el siguiente código. Este procedimiento sumará la cantidad de productos que pertenecen a la categoría especificada.



1. **Realiza el ejercicio anterior haciendo uso de Function**

**Este es el 8**



**Y este el 9 en FUNCTION  
  
**

NOTA: Se tuvo que utilizar el character varying y el integer en el

*SELECT*

*c.nombre::CHARACTER VARYING AS categoria\_nombre,*

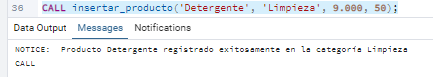
*COUNT(p.\*)::INTEGER AS total\_productos*

porque daba error en el tipo de dato a la hora de darle en el

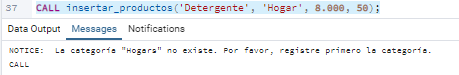
*select \* from obtener\_sumatoria\_productos\_categoria(‘Granos’);*

1. **Crea un procedimiento almacenado que inserte un producto, donde los parámetros de entrada son el nombre, nombre de categoría, precio, stock, y de un mensaje por pantalla. Si el producto a registrar la categoría es artículo de limpieza, no lo registre y de el mensaje por pantalla.**

**Se hicieron dos procedimientos distintos por la razón de que no se comprendió bien la pregunta , 1) si alguna categoría con el nombre “artículo de limpieza” no se registre o 2) si los artículos de la categoría “Limpieza” no se pueden registrar.**

1. **Artículo de limpieza, este primero si permitiría registrar la categoría “Limpieza”  
   ** ****
2. **Limpieza**

****

****

TRIGGERS

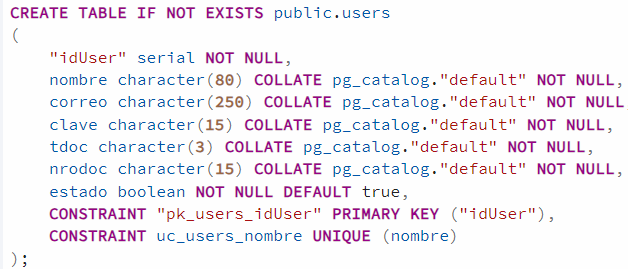
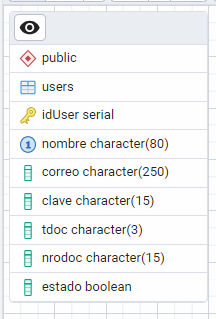
En PostgreSQL, los **Triggers** (disparadores) son funciones especiales que se ejecutan automáticamente en respuesta a ciertos eventos que ocurren en una tabla o vista. Estos eventos pueden ser operaciones como **INSERT**, **UPDATE**, **DELETE**, e incluso eventos relacionados con tablas temporales o vistas materializadas.

**¿Qué hacen los triggers?**

* Automatizan tareas repetitivas o complejas.
* Validan o transforman datos antes o después de realizar una operación.
* Mantienen la integridad de los datos.
* Realizan operaciones en otras tablas relacionadas.

1. **Ejercicio:**  se quiere realizar un trigger de auditoría sobre cualquier cambio (update y delete) de información que se realice en la tabla users, y se desea que se guarden dichos cambios en una tabla de datos auditados.

**Tabla Users**

Interfaz de usuario gráfica, Aplicación, Teams

El contenido generado por IA puede ser incorrecto.

* 1. Para iniciar se debe crear la tabla que contendrá los datos de auditoria.

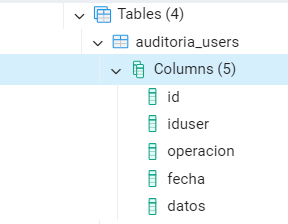
Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

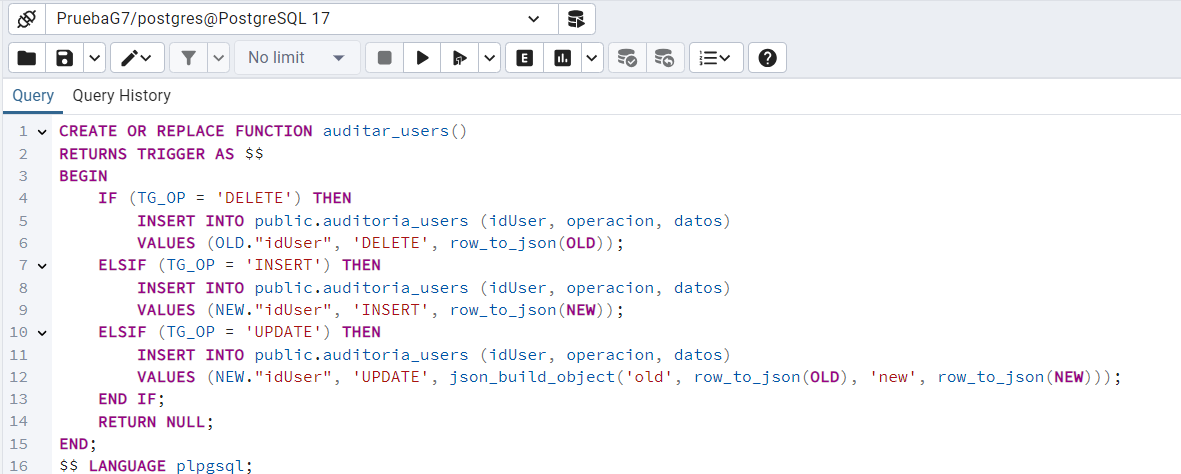
idUser: id del usuario afectado de la operación

operación: que acción se realizó (insert, update, delete)

datos: Datos de la fila afectada, en formato JSON



* 1. A continuación, creemos la función que se utilizará para Tigger



La función asociada al trigger manejará las operaciones de auditoría. En esta función se usará TG\_OP para determinar si la operación fue un INSERT, UPDATE o DELETE.

TG\_OP: Es una variable especial en triggers que indica la operación actual que se esta realizando (INSERT, UPDATE o DELETE).

También se utilizarán OLD y NEW para capturar los datos de la fila afectada.

OLD y NEW: Contienen los datos de la fila antes y después de la operación.

row\_to\_json: Convierte las filas afectadas en formato JSON para un almacenamiento detallado.

json\_build\_object: En el caso de actualizaciones (UPDATE), guarda tanto los datos antiguos (OLD) como los nuevos (NEW).

* 1. Crea el Tigger

Texto

El contenido generado por IA puede ser incorrecto.

AFTER INSERT OR UPDATE OR DELETE: Define que el trigger se ejecutará después de estas operaciones.

FOR EACH ROW: Indica que se ejecutará una vez por cada fila afectada.

TG\_OP: Es una variable especial en triggers que indica la operación actual (INSERT, UPDATE o DELETE).

* 1. Hagamos las pruebas del Tigger que acabamos de crear

Realiza operaciones en la tabla users y verifica la tabla de auditoría:

Ejemplo:

INSERT INTO users (nombre, correo, clave, tdoc, nrodoc) VALUES ('Juan Pérez', 'juan@gmail.com', 'clave123', 'CC', '12345678');

UPDATE users SET correo = 'juanperez@gmail.com' WHERE nombre = 'Juan Pérez';

DELETE FROM users WHERE nombre = 'Juan Pérez';

* 1. Una vez realizadas las operaciones, consultemos la tabla de auditoría:

La estructura que observamos en el Pbject Explorer:

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Imagen que contiene Logotipo

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

**ADICIONES: Para eliminar un Tigger creado y sus tablas:**

1ro. Elimina el Tigger

DROP TRIGGER IF EXISTS trigger\_auditoria\_users ON users;

2do. Eimina la función

DROP FUNCTION IF EXISTS auditar\_users();

3ro. Elimina la tabla de auditoria

DROP TABLE IF EXISTS auditoria\_users;

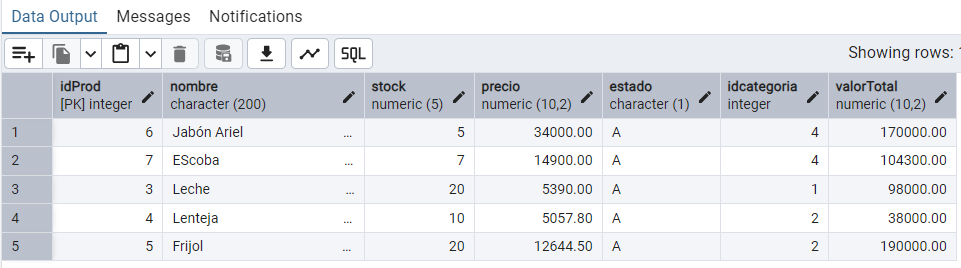
Si deseeas conservar el historial de auditoría antes de eliminar la tabla, realiza un respaldo:

COPY auditoria\_users TO '/ruta/auditoria\_users\_backup.csv' CSV HEADER;

1. Tengo la siguiente estructura de productos y categorías:

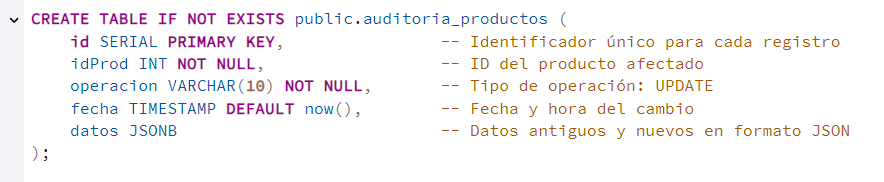
Imagen que contiene Diagrama

El contenido generado por IA puede ser incorrecto.



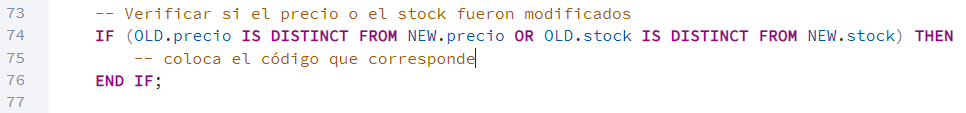
Se desea crear un Tigger que guarde la auditoria de los cambios de la función

1. Crea la tabla de la auditoria



1. Crear la Función del Trigger, para ello ten en cuenta las siguientes tareas:

* Verificar si el precio o el stock fueron modificados.

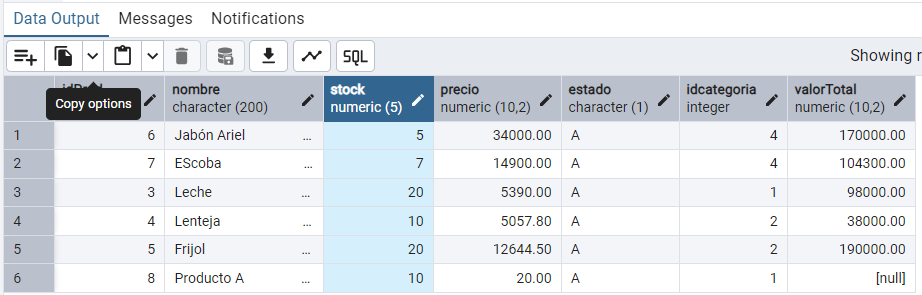


* Actualizar el campo valorTotal calculado como stock \* precio.
* Registrar los datos antiguos y nuevos en la tabla de auditoría.

1. Inserta el siguiente producto de prueba:

INSERT INTO public.productos (nombre, stock, precio, idcategoria)

VALUES ('Producto A', 10, 20.00, 1);

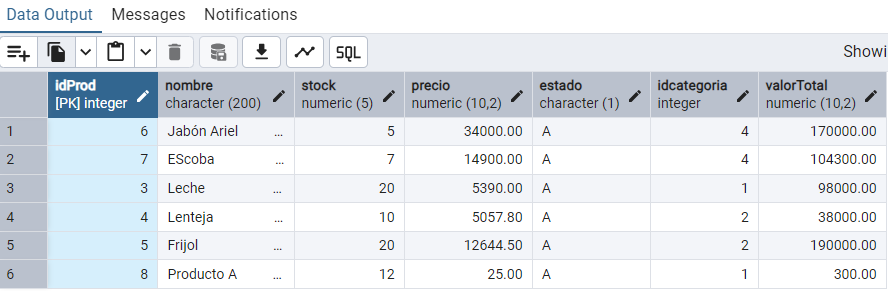


Luego, realiza un update:

UPDATE public.productos

SET precio = 25.00, stock = 12

WHERE "idProd" = 8;



1. Prueba el Tigger realizado

SELECT \* FROM public.auditoria\_productos;

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

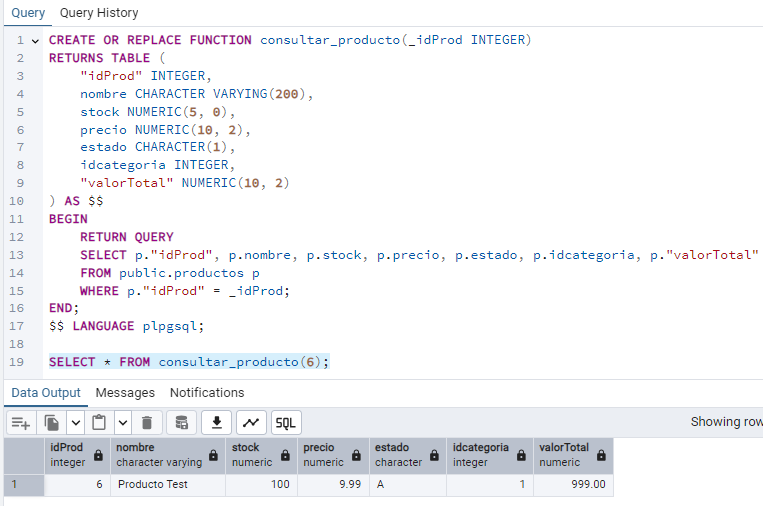
1. Crea una función por cada una de las funcionalidades de un CRUD, para productos, es decir:

* Función para insertar un producto, que reciba los parámetros de entrada para crear un producto.

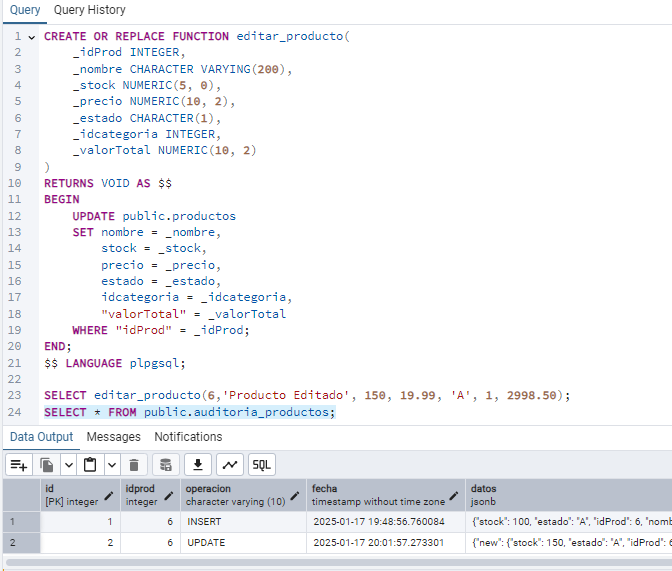
Nota como ya se tiene creado el tigger de auditoria al crear o modificar un producto en el que se actualiza el valor total, este se ejecutará cada vez que inserte un producto en la base de datos.

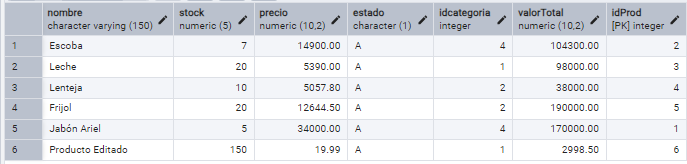


* Función para consultar un producto, donde el parámetro de entrada sea el id del producto, y retorne los datos del producto.

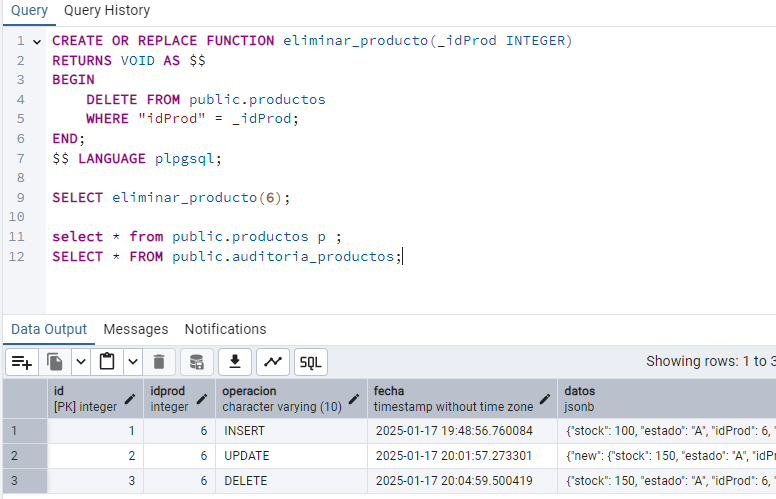


* Función para editar un producto, donde los parámetros de entrada sea el id del producto, y todos los datos a actualizar. Nota como ya se tiene creado el tigger de auditoría, que aplica para crear o modificar un producto en el que se actualizara el valor total, este se ejecutará cada vez que inserte un producto en la base de datos.

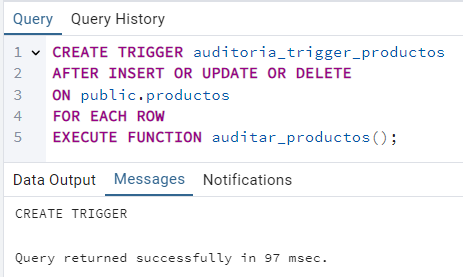


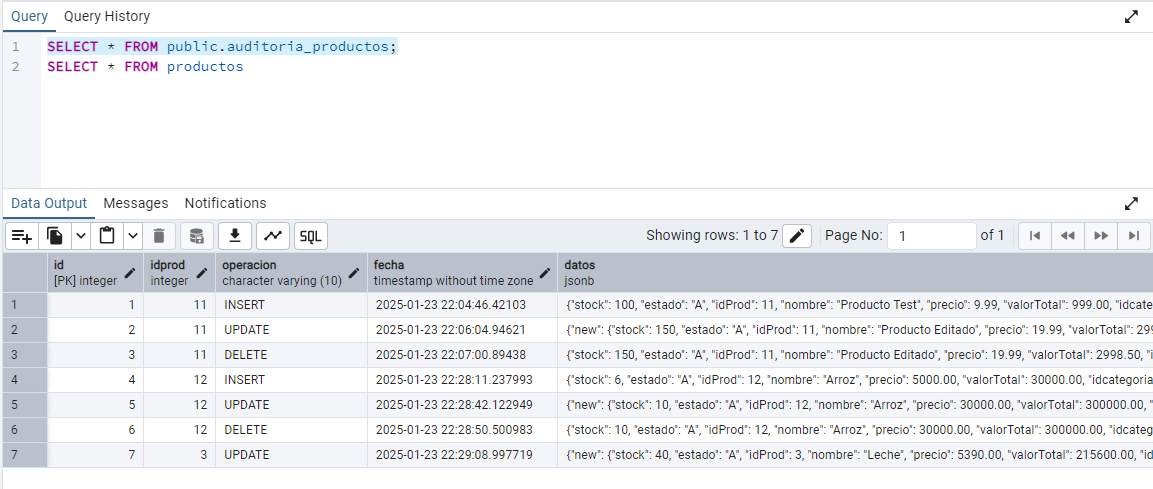


* Función para eliminar un producto, donde el parámetro de entrada sea el id del producto.



1. Completar el tigger necesario para hacer la auditoria cada vez que se realice una función del CRUD a la tabla de productos.  
   





1. Al proyecto realizado en formación el NODE-MGDB sustituir las consultas de sql por las funciones desarrolladas.
2. Elaborar un video mostrando el trabajo funcional del punto 15, tato en el aplicativo y código, como la estructura de base de datos de postgresql.
3. Subir la información a github y entregar el link como entrega de la actividad.



**¡No olvides Nunca, Ser Feliz!**

**¡Todo forma parte de la vida que formará tu carácter!**

Ánimo!. Magno.