ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

Tema:

Sistema de Gestión para Supermercado

Integrantes:

Ayol Nayely

Pérez Alessia

Docente:

Ing. Yadira Franco

Fecha:

03/08/2025

ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

ÍNDICE

1. DESCRIPCIÓN DEL SISTEMA

Se desarrollo un sistema de gestión para un supermercado. Las características principales de este son seguridad, integridad y control de acceso a la base de datos. Dentro del sistema se gestionan los datos sensibles de los clientes, empleados, productos, ventas, etc., asegurando su integridad al controlar el acceso de diferentes usuarios a través de restricciones de visualización o manipulación de datos.

El problema que se resuelve con este sistema es la protección de datos sensibles de los diferentes usuarios en la base de datos. Con esto se réplica el método de trabajo que suele llevarse a cabo en entornos reales. Es decir, la práctica de implementar varias capas de seguridad y validación de datos tiene un objetivo, el cual es evitar la filtración de información confidencial al dejar la base mal protegida.

2. OBJETIVOS

2.1 Objetivo General

Desarrollar un sistema seguro en SQL Server para la gestión de un supermercado.

2.2 Objetivos específicos

- Implementar mecanismo para la protección y validación de datos a través de cifrado y control de accesos.
- Registrar acciones importantes dentro de la base de datos a través de triggers y bitácoras.

3. ANÁLISIS DEL PROBLEMA

Muchas veces, las bases de datos de diferentes sistemas (en especial los de empresas pequeñas) no cuentan con las validaciones necesarias para controlar y mantener los datos seguros. Lo cual, pone en riesgo la integridad de la base de datos, así como también la confianza que los clientes depositan en la empresa.

El ingreso de datos inválidos, acceso no autorizado o la alteración de registros son algunos de los problemas que se busca contrarrestas con la implementación de seguridad en el sistema. Esto para evitar un mal uso de la información que se encuentra dentro de la base.

4. JUSTIFICACIÓN

Toda base de datos debe cumplir con diferentes normativas de protección de datos personales. Con el diseño e implementación del sistema se busca mantener la confiabilidad e integridad de los datos que contienen información crítica del negocio.

5. REQUISITOS DEL SISTEMA

- Implementación de tablas que registren las diferentes acciones dentro de la base de datos.
- Validación de campos para evitar errores o inconsistencia en los datos.
- Encriptación de datos sensibles para mejorar la seguridad en la base.
- Creación y gestión de roles y usuarios para controlar el acceso a la base de datos.
- Configuración de mensajes de error para señalar las acciones que se pueden y no se pueden realizar.

6. TECNOLOGÍAS USADAS

SQL Server

- Usado para la implementación de la base de datos para un supermercado.

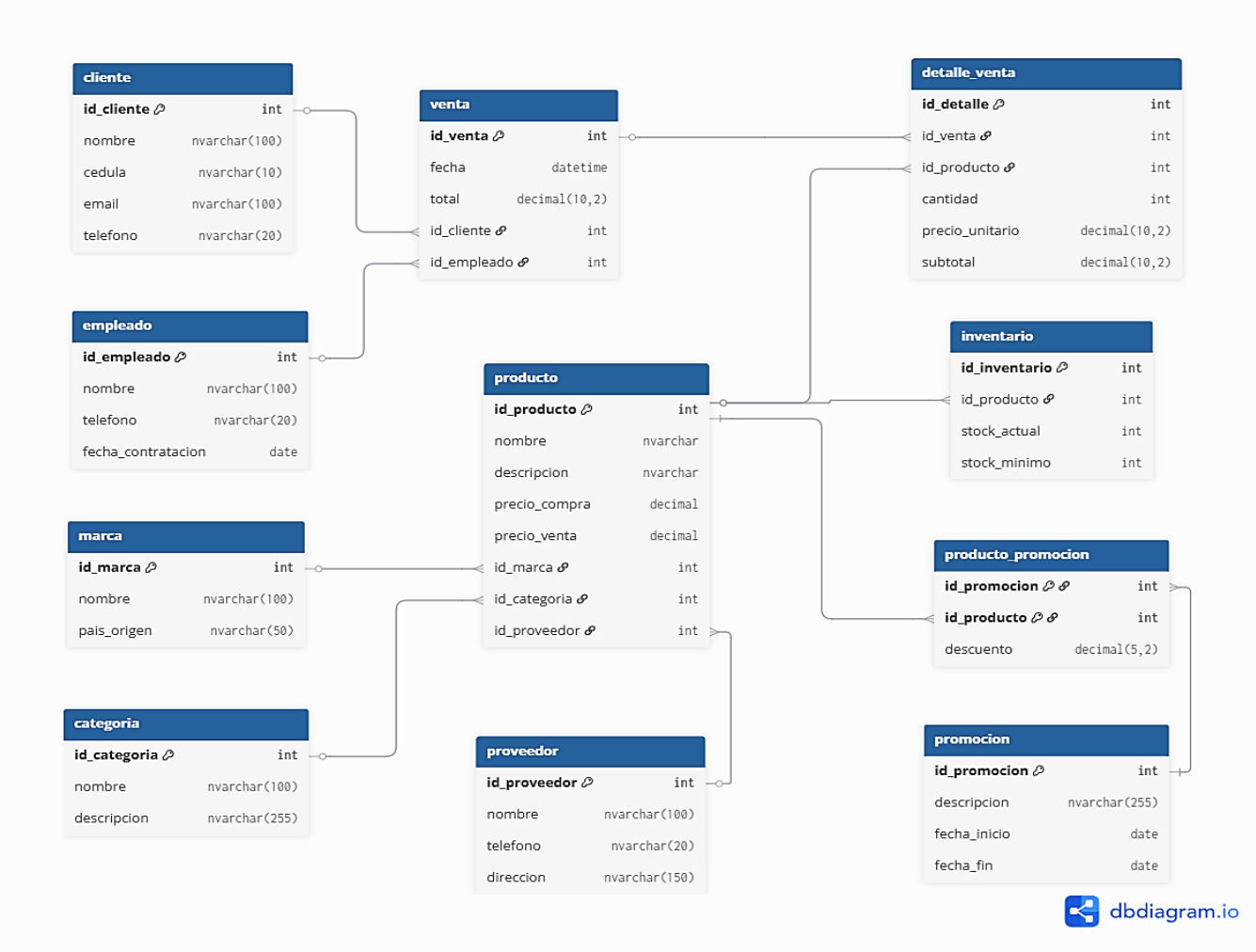
Cifrado y hashing

- Se uso ENCRYPTBYPASSPHRASE y HASHBYTES para la encriptación de cedula, email y teléfono de los datos del cliente.

Services.msc

- Utilizado para detener el servicio de SQL Server desde el panel de servicios de Windows durante el backup en frío, permitiendo copiar los archivos físicos .mdf y .ldf de la base de datos.

7. DIAGRAMA DEL SISTEMA



Link para visualizar el diagrama

- https://dbdiagram.io/d/Proyecto_Supermercado-6886ba24cca18e685cf35ae1

Link del GitHub

- https://github.com/NayelyAyol/PROYECTO_BD_SUPERMERCADO

7.1 Estructura de la base de datos

La base cuanta con un total de 11 tablas:

Cliente. - Para almacenar los datos privados de los clientes.

Empelado. – Se guarda la información de los empleados que tienen acceso al sistema.

Producto. – Contiene los datos de cada producto en el supermercado

Venta. – Se registran las ventas realizadas.

Detalle venta. – Se especifica que producto fue vendido.

Pago. – Se controlan los pagos realizados por los clientes.

Devolución. – Se registran las devoluciones que han realizado los clientes.

Inventario. – Contiene el estado y la cantidad de los productos.

Promoción. – Se guardan las promociones que son aplicadas a ciertos productos.

Producto promoción. – Tabla que representa la relación de muchos a muchos entre producto y promoción.

Bitácora. – Se registran las acciones relevantes realizadas en el sistema.

7.2 Relaciones entre tablas

Cliente → Venta

Relación: 1: N

Un cliente puede tener muchas ventas.

Empleado → Venta

Relación: 1: N

Un empleado puede registrar muchas ventas.

Venta → detalle_venta

Relación: 1: N

Una venta puede tener múltiples productos.

Producto → detalle_venta

Relación: 1: N

Un producto puede ser parte de varias ventas.

Producto → Inventario

Relación: 1: N

Cada producto es registrado en el inventario.

Marca → Producto

Relación: 1: N

Varios productos pueden pertenecer a una marca.

Categoría → Producto

Relación: 1: N

Varios productos pueden pertenecer a una categoría.

Proveedor → Producto

Relación: 1: N

Un proveedor puede proveer varios productos.

Producto → Promoción

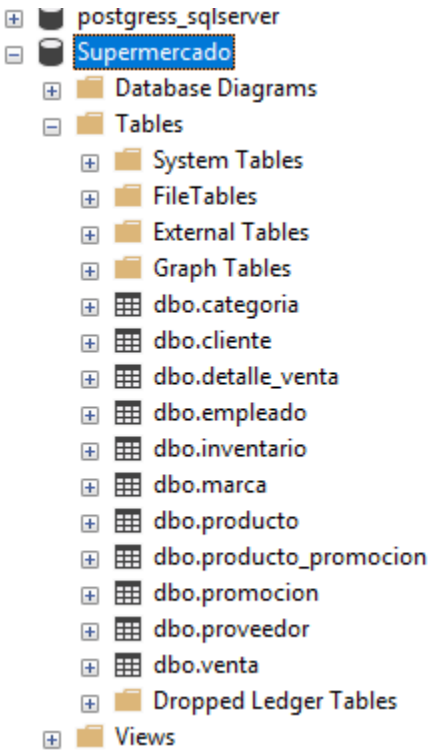
Relación: N: M

Un producto puede tener varias promociones.

Una promoción puede ser aplicada a varios productos.

8. INTEGRIDAD Y RESTRICCIONES

Se usaron restricciones: NOT NULL, UNIQUE, CHECK, DEFAULT, AUTO_INCREMENT o SERIAL. Y para las relaciones entre tablas se usaron claves foráneas. A su vez, se aplicó el ON DELETE CASCADE, SET NULL.



1. Tabla cliente

```
-- Tabla 1: cliente
CREATE TABLE cliente (
    id_cliente INT PRIMARY KEY IDENTITY(1,1),
    nombre NVARCHAR(100) NOT NULL,
    cedula NVARCHAR(10) NOT NULL UNIQUE CHECK (LEN(cedula) = 10),
    email NVARCHAR(100) UNIQUE,
    telefono NVARCHAR(20)
);
```

2. Tabla empleado

```
-- Tabla 2: empleado
CREATE TABLE empleado (
    id_empleado INT PRIMARY KEY IDENTITY(1,1),
    nombre NVARCHAR(100) NOT NULL,
    telefono NVARCHAR(20),
    fecha_contratacion DATE DEFAULT GETDATE()
);
```

3. Tabla marca

```
-- Tabla 3: marca
CREATE TABLE marca (
    id_marca INT PRIMARY KEY IDENTITY(1,1),
    nombre NVARCHAR(100) NOT NULL UNIQUE,
    pais_origen NVARCHAR(50)
);
```

4. Tabla categoría

```
-- Tabla 4: categoria
CREATE TABLE categoria (
    id_categoria INT PRIMARY KEY IDENTITY(1,1),
    nombre NVARCHAR(100) NOT NULL UNIQUE,
    descripcion NVARCHAR(255)
);
```

5. Tabla proveedor

```
-- Tabla 4: categoria
CREATE TABLE categoria (
    id_categoria INT PRIMARY KEY IDENTITY(1,1),
    nombre NVARCHAR(100) NOT NULL UNIQUE,
    descripcion NVARCHAR(255)
);
```

6. Tabla producto

```sql
-- Tabla 6: producto
CREATE TABLE producto (
    id_producto INT PRIMARY KEY IDENTITY(1,1),
    nombre NVARCHAR(100) NOT NULL,
    descripcion NVARCHAR(255),
    precio_compra DECIMAL(10,2) NOT NULL CHECK (precio_compra > 0),
    precio_venta DECIMAL(10,2) NOT NULL CHECK (precio_venta > 0),
    id_marca INT,
    id_categoria INT NOT NULL,
    id_proveedor INT NOT NULL,

    FOREIGN KEY (id_marca) REFERENCES marca(id_marca) ON DELETE SET NULL,
    FOREIGN KEY (id_categoria) REFERENCES categoria(id_categoria),
    FOREIGN KEY (id_proveedor) REFERENCES proveedor(id_proveedor)
);
```

7. Tabla inventario

```sql
-- Tabla 7: inventario
CREATE TABLE inventario (
    id_inventario INT PRIMARY KEY IDENTITY(1,1),
    id_producto INT NOT NULL UNIQUE,
    stock_actual INT NOT NULL CHECK (stock_actual >= 0) DEFAULT 0,
    stock_minimo INT NOT NULL CHECK (stock_minimo >= 0) DEFAULT 10,
    FOREIGN KEY (id_producto) REFERENCES producto(id_producto) ON DELETE CASCADE
);
```

8. Tabla venta

```sql
-- Tabla 8: venta
CREATE TABLE venta (
    id_venta INT PRIMARY KEY IDENTITY(1,1),
    fecha DATETIME NOT NULL DEFAULT GETDATE(),
    total DECIMAL(10,2) NOT NULL CHECK (total >= 0),
    id_cliente INT,
    id_empleado INT NOT NULL,
    FOREIGN KEY (id_cliente) REFERENCES cliente(id_cliente) ON DELETE SET NULL,
    FOREIGN KEY (id_empleado) REFERENCES empleado(id_empleado)
);
```

9. Tabla detalle venta

```sql
-- Tabla 9: detalle_venta
CREATE TABLE detalle_venta (
    id_detalle INT PRIMARY KEY IDENTITY(1,1),
    id_venta INT NOT NULL,
    id_producto INT NOT NULL,
    cantidad INT NOT NULL CHECK (cantidad > 0),
    precio_unitario DECIMAL(10,2) NOT NULL CHECK (precio_unitario > 0),
    subtotal DECIMAL(10,2) NOT NULL CHECK (subtotal >= 0),
    FOREIGN KEY (id_venta) REFERENCES venta(id_venta) ON DELETE CASCADE,
    FOREIGN KEY (id_producto) REFERENCES producto(id_producto)
);
```

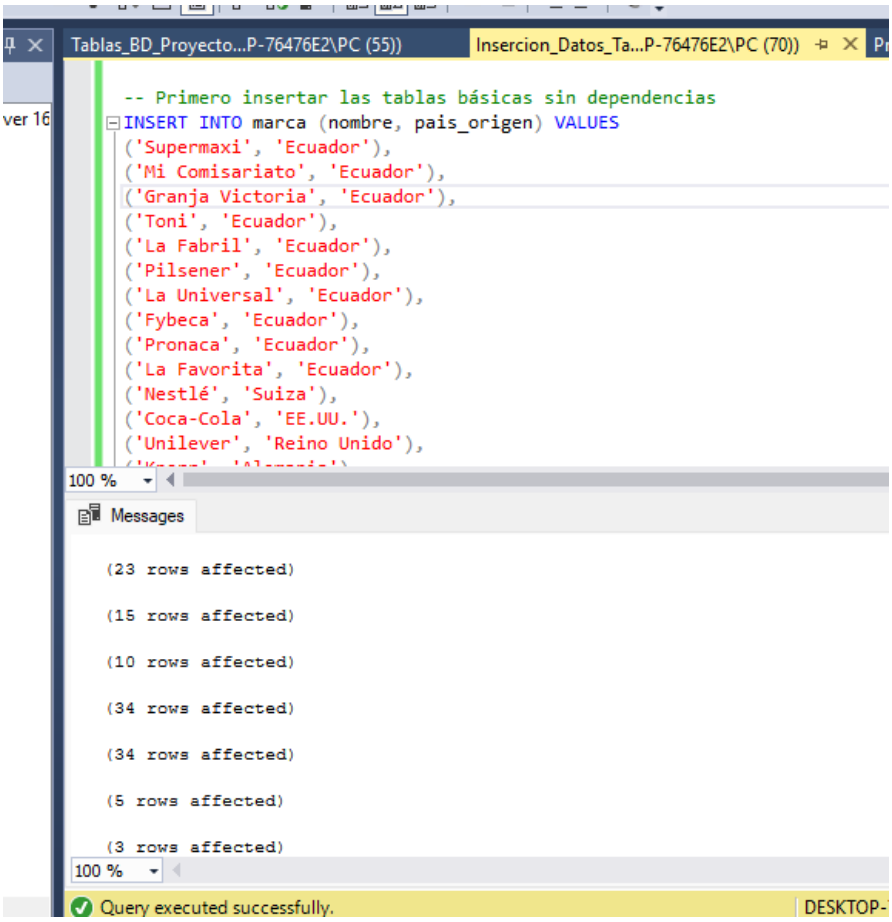
10. Tabla promoción

```sql
-- Tabla 10: promocion
CREATE TABLE promocion (
    id_promocion INT PRIMARY KEY IDENTITY(1,1),
    descripcion NVARCHAR(255),
    fecha_inicio DATE NOT NULL,
    fecha_fin DATE NOT NULL,
    CHECK (fecha_fin > fecha_inicio)
);
```

11. Tabla producto promoción

```sql
-- Tabla 11: producto_promocion
CREATE TABLE producto_promocion (
    id_promocion INT NOT NULL,
    id_producto INT NOT NULL,
    descuento DECIMAL(5,2) NOT NULL CHECK (descuento BETWEEN 0 AND 100),
    PRIMARY KEY (id_promocion, id_producto),
    FOREIGN KEY (id_promocion) REFERENCES promocion(id_promocion) ON DELETE CASCADE,
    FOREIGN KEY (id_producto) REFERENCES producto(id_producto) ON DELETE CASCADE
);
```

9. INSERCIÓN DE DATOS



10. CONSULTAS

10.1Joins

1. Productos con información completa.

```sql
-- Productos con informacion completa
SELECT p.id_producto, p.nombre AS Producto, m.nombre AS Marca,
       c.nombre AS Categoria, pr.nombre AS Proveedor,
       p.precio_venta, i.stock_actual
FROM producto p
JOIN marca m ON p.id_marca = m.id_marca
JOIN categoria c ON p.id_categoria = c.id_categoria
JOIN proveedor pr ON p.id_proveedor = pr.id_proveedor
JOIN inventario i ON p.id_producto = i.id_producto;
```

100 %

Results | Messages

	id_producto	Producto	Marca	Categoria	Proveedor	precio_venta	stock_actual
1	1	Leche Entera Vita	Toni	Lácteos	Lácteos del Valle	1.20	50
2	2	Yogurt Natural L...	La F...	Lácteos	Lácteos del Valle	0.75	40
3	3	Queso Fresco P...	Pron...	Lácteos	Lácteos del Valle	3.50	30
4	4	Mantequilla La ...	La F...	Lácteos	Lácteos del Valle	1.80	60
5	5	Leche Deslacto...	Toni	Lácteos	Lácteos del Valle	1.40	45
6	6	Agua Mineral T...	La F...	Bebidas	Distribuidora A...	0.50	35
7	7	Gaseosa Coca-...	Coc...	Bebidas	Distribuidora A...	1.00	25
8	8	Jugo de Naranj...	Nestlé	Bebidas	Distribuidora A...	1.80	40
9	9	Arroz Supremo	La F...	Abarrotes	Alimentos Ecua...	1.20	30
10	10	Aceite La Fabril	La F...	Abarrotes	Alimentos Ecua...	2.00	50
11	11	Pechuga de Pol...	Pron...	Carnes	Carnes Premiu...	4.90	45
12	12	Carne Molida La...	La F...	Carnes	Carnes Premiu...	5.80	55
13	15	Detergente Ace	Unil...	Limpieza	Importadora El...	3.50	40

2. Clientes con más compras realizadas

```sql
-- 2. Clientes con más compras realizadas.

SELECT c.id_cliente, c.nombre AS cliente, COUNT(v.id_venta) AS total_compras
FROM cliente c
JOIN venta v ON c.id_cliente = v.id_cliente
GROUP BY c.id_cliente, c.nombre
ORDER BY total_compras DESC;
```

100 %

Results | Messages

	id_cliente	cliente	total_compras
1	1	Juan Pérez	1
2	2	María Gómez	1
3	3	Carlos Andrade	1
4	4	Ana Beltrán	1

3. Productos en promoción.

```
-- 3.   Productos en promoción.

SELECT p.nombre as Producto, p.precio_venta as Precio_Inicial, prom.descuento,
       p.precio_venta * (1-prom.descuento/100) as Precio_Final
FROM producto p
JOIN producto_promocion prom ON p.id_producto = prom.id_producto
JOIN promocion pr ON prom.id_promocion = pr.id_promocion
ORDER BY prom.descuento DESC;
```

100 %

Results | Messages

	Producto	Precio_Inicial	descuento	Precio_Final
1	Agua Mineral Tesalia	0.50	50.00	0.25000000
2	Gaseosa Coca-Cola	1.00	40.00	0.60000000
3	Jugo de Naranja Tang	1.80	30.00	1.26000000
4	Detergente Ace	3.50	30.00	2.45000000
5	Cloro Magia Blanca	1.20	25.00	0.90000000
6	Pan Integral Modema	2.20	20.00	1.76000000
7	Pechuga de Pollo Pronaca	4.90	20.00	3.92000000
8	Carne Molida La Favorita	5.80	15.00	4.93000000
9	Pan de Avena San Francisco	2.50	15.00	2.12500000
10	Leche Entera Vita	1.20	15.00	1.02000000
11	Queso Fresco Pronaca	3.50	12.00	3.08000000
12	Yogurt Natural La Favorita	0.75	10.00	0.67500000

4. Productos con stock bajo.

```
-- 4.   Productos con stock bajo.

SELECT p.nombre as Producto, i.stock_actual, i.stock_minimo,
       pr.nombre as Proveedor, pr.telefono as Contacto_Proveedor
FROM inventario i
JOIN producto p ON i.id_producto = p.id_producto
JOIN proveedor pr ON p.id_proveedor = pr.id_proveedor
WHERE i.stock_actual < i.stock_minimo;


--
```

100 %

Results | Messages

Producto	stock_actual	stock_minimo	Proveedor	Contacto_Proveedor

5. Ventas con sus clientes, productos y categorías.

```sql
-- 5.   Ventas con sus clientes, productos y categorías (solo de los clientes asociados).

SELECT v.id_venta, v.fecha, c.nombre as Cliente, p.nombre as Producto,
        ct.nombre as Categoria
FROM venta v
JOIN cliente c ON v.id_cliente = c.id_cliente
JOIN detalle_venta dv ON v.id_venta = dv.id_venta
JOIN producto p ON dv.id_producto = p.id_producto
JOIN categoria ct ON p.id_categoria = ct.id_categoria;
```

100 %

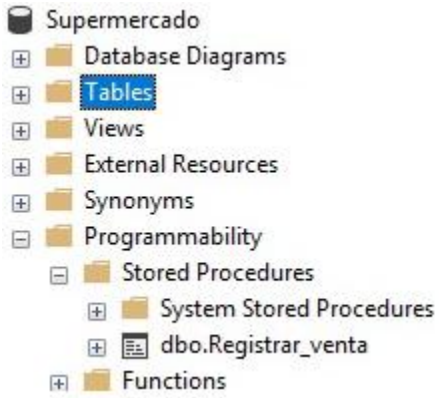
Results | Messages

	id_venta	fecha	Cliente	Producto	Categoria
1	1	2023-01-07 09:00:00.000	Juan Pérez	Leche Entera Vita	Lácteos
2	1	2023-01-07 09:00:00.000	Juan Pérez	Agua Mineral Tesalia	Bebidas
3	1	2023-01-07 09:00:00.000	Juan Pérez	Pechuga de Pollo Pronaca	Carnes
4	2	2023-01-07 10:15:00.000	María Gómez	Yogurt Natural La Favorita	Lácteos
5	2	2023-01-07 10:15:00.000	María Gómez	Gaseosa Coca-Cola	Bebidas
6	3	2023-01-07 11:30:00.000	Carlos Andrade	Queso Fresco Pronaca	Lácteos
7	3	2023-01-07 11:30:00.000	Carlos Andrade	Jugo de Naranja Tang	Bebidas
8	3	2023-01-07 11:30:00.000	Carlos Andrade	Manzanas Delicia	Frutas/Verduras
9	5	2023-01-07 16:30:00.000	Ana Beltrán	Leche Deslactosada Toni	Lácteos
10	5	2023-01-07 16:30:00.000	Ana Beltrán	Aceite La Fabril	Abarrotes
11	5	2023-01-07 16:30:00.000	Ana Beltrán	Detergente Ace	Limpieza

11. PROCEDIMIENTOS

1. Insertar una venta si el cliente y el empleado existen previamente.

```sql
-- PROCEDIMIENTOS ALMACENADOS
-- 1) Insertar una venta si el cliente y el empleado existe previamente
create procedure Registrar_venta
    @fecha datetime,
    @id_cliente int,
    @id_empleado int,
    @total decimal(10,2)
as
begin
-- validar que al fecha sea correcta y no sea futura
    if @fecha > GETDATE()
    begin
        print 'Error: No se puede registrar una venta con una fecha futura';
        return;
    end
-- Validamos si el cliente existe
    if not exists (select 1 from cliente where id_cliente=@id_cliente)
    begin
        print 'El cliente no existe';
        return;
    end
-- Verificamos si el empleado existe
if not exists (select 1 from empleado where id_empleado=@id_empleado)
    begin
        print 'El empleado no existe';
        return;
    end
-- Insertamos la venta si todo está correcto
    insert into venta(fecha, total, id_cliente,id_empleado) values
    (@fecha, @total, @id_cliente, @id_empleado);
end;
go
```



Prueba:

```
-- Verificando el funcionamiento
exec Registrar_venta
     @fecha = '2025-07-31',
     @id_cliente = 1,
     @id_empleado = 2,
     @total = 30.50;
```

6 %

Messages

```
(1 row affected)

Completion time: 2025-08-01T20:02:47.4717959-05:00
```

2. Aumentar el stock mínimo de un producto por categoría.

```
-- Actualizaciones masivas por condición.
-- 2) Aumentar el stock minimo de un producto por categoría
create procedure ActualizarStockMinimoCategoria
    @id_categoria int,
    @nuevo_stock_minimo int
as
begin
    -- Validar que el nuevo stck mínimo no sea menos a 0
    if @nuevo_stock_minimo <0
        begin
            print 'El stock mínimo no puede ser negativo';
            return;
        end
    -- Actualizar el stock mínimo de todos los datos depensidendo del id que se da
        update inventario
        set stock_minimo=@nuevo_stock_minimo
        where id_producto in(
            select id_producto from producto
            where id_categoria=@id_categoria);
        print 'Actualización masiva realizada de manera correcta';
end;
go
-- Para verificar
select i.id_producto, i.stock_minimo, p.nombre, p.id_categoria
from inventario i
join producto p on i.id_producto = p.id_producto
where p.id_categoria=2;
go
```

Prueba:

```
exec ActualizarStockMinimoCategoria @id_categoria =2, @nuevo_stock_minimo= 7;
```

112 %

Messages

```
(3 rows affected)
Actualización masiva realizada de manera correcta

Completion time: 2025-08-01T20:35:50.0241771-05:00
```

```
-- Para verificar
exec ActualizarStockMinimoCategoria @id_categoria =2, @nuevo_stock_minimo= 7;
go

select i.id_producto, i.stock_minimo, p.nombre, p.id_categoria
from inventario i
join producto p on i.id_producto = p.id_producto
where p.id_categoria=2;
go
```

112 %

Results Messages

	id_producto	stock_minimo	nombre	id_categoria
1	6	7	Agua Mineral Tesalia	2
2	7	7	Gaseosa Coca-Cola	2
3	8	7	Jugo de Naranja Tang	2

3. Eliminación de un cliente.

```
-- Eliminación segura
-- 3) Eliminacion de un cliente
create procedure Eliminar_Cliente
    @id_cliente int
as
begin
-- Verificar si el cliente existe
    if not exists(select 1 from cliente where id_cliente=@id_cliente)
        begin
            print 'El cleinte no existe';
            return;
        end
-- Verificar si el cliente tiene una venta registrada
    if not exists(select 1 from venta where id_cliente=@id_cliente)
        begin
            print 'El cliente no se puede eliminar, tiene ventas registradas';
            return;
        end
-- Eliminamos si todo está correcto
    delete from cliente where id_cliente=@id_cliente
    print 'El cliente ha sido eliminado correctamente';
end;
go
```

Prueba:

```
-- Verificar el funcionamiento
exec Eliminar_Cliente @id_cliente=3;
go

select * from cliente where id_cliente=3;
```

112 %

Results | Messages

id_cliente	nombre	cedula	email	telefono

4. Reporte de ventas por precio.

```
-- Generación de reportes por período.
-- 4) Reporte de ventas por perido
create procedure Reporte_Ventas_Periodo
    @fecha_inicio date,
    @fecha_fin date
as
begin
-- Validar que las fechas tengan sentido
    if @fecha_fin < @fecha_inicio
    begin
        print 'La fecha final no pued ser menor que la fecha de inicio';
        return;
    end
-- Mostar el reporte de ventas por periodo
    select id_venta, fecha, total from venta
    where fecha between @fecha_inicio and @fecha_fin
    order by fecha;
end;
go
```

Prueba:

```
-- Verificando el funcionamiento
exec Reporte_Ventas_Periodo
    @fecha_inicio = '2025-01-01',
    @fecha_fin='2025-12-31';
```

112 %

Results | Messages

	id_venta	fecha	total
1	6	2025-07-31 00:00:00.000	30.50

5. Factura automática.

```
-- Facturación automática
create procedure Facturacion_simple
    @id_cliente int,
    @id_empleado int,
    @id_producto int,
    @cantidad int,
    @fecha datetime
as
begin
-- validar que el stock sea suficiente
    if not exists(select 1 from inventario where id_producto= @id_producto
    and stock_actual>= @cantidad)
    begin
        print'Producto sin stock, o inexistente';
        return;
    end
    declare @precio_unitario decimal(10,2);
    declare @subtotal decimal(10,2);

    select @precio_unitario= precio_venta from producto
    where id_producto=@id_producto;
    set @subtotal=@precio_unitario*@cantidad;
-- Se inserta la venta principal
    declare @id_venta int;
    insert into venta(fecha, total, id_cliente, id_empleado)
    values (@fecha, @subtotal, @id_cliente, @id_empleado);
    set @id_venta= SCOPE_IDENTITY();
-- Insertamos el detalle de ventas
    insert into detalle_venta(id_venta, id_producto, cantidad, precio_unitario, subtotal)
    values(@id_venta, @id_producto, @cantidad, @precio_unitario, @subtotal);
-- Se descuenta a cantidad vendida del inventario
    update inventario
    set stock_actual=stock_actual - @cantidad
    where id_producto= @id_producto;
    print'Venta registrada exitosamente';
end;
```

Prueba:

```
-- Verificando
exec Facturacion_simple
    @id_cliente = 1,
    @id_empleado = 2,
    @id_producto = 3,
    @cantidad = 4,
    @fecha = '2025-08-01';
```

```
112 %
Messages

(1 row affected)

(1 row affected)

(1 row affected)
Venta registrada exitosamente
```

6. Facturación controlada.

```
-- Transacciónn controlada
create procedure Facturacion_Transaccion
    @id_cliente int,
    @id_empleado int,
    @id_producto int,
    @cantidad int,
    @fecha datetime
as
begin
    begin tran; -- Aquí inicializamos la transaccion
-- Verificar si hay el stock suficiente
    if not exists(select 1 from inventario
    where id_producto= @id_producto and stock_actual>= @cantidad)
    begin
        print'Stock insuficiente o producto inexistente';
        rollback;
        return;
    end
    declare @precio_unitario decimal(10,2);
    declare @subtotal decimal(10,2);
    declare @id_venta int;
-- Calcular el precio y subtotal
    select @precio_unitario=precio_venta from producto
    where id_producto=@id_producto;
    set @subtotal=@precio_unitario*@cantidad;
-- Insertar venta
    insert into venta(fecha, total, id_cliente, id_empleado)
    values(@fecha, @subtotal, @id_cliente, @id_empleado);
    set @id_venta= SCOPE_IDENTITY();
-- Insertar un detalle de venta
    insert into detalle_venta(id_venta, id_producto, cantidad, precio_unitario, subtotal)
    values(@id_venta, @id_producto, @cantidad, @precio_unitario, @subtotal);
-- Actualizar el inventario
    update inventario
    set  stock_actual=stock_actual-@cantidad
    where id_producto= @id_producto;
    commit;
    print'Venta realizada correctamente';
end;
go
```

Prueba:

```
-- Verificando
exec Facturacion_Transaccion
    @id_cliente = 1,
    @id_empleado = 2,
    @id_producto = 3,
    @cantidad = 2,
    @fecha = '2025-08-01';
```

```
(1 row affected)

(1 row affected)

(1 row affected)
Venta realizada correctamente

Completion time: 2025-08-02T01:00:50.6495951-05:00
```

```
exec Facturacion_Transaccion
    @id_cliente = 1,
    @id_empleado = 2,
    @id_producto = 3,
    @cantidad = 9999,
    @fecha = '2025-08-02';
go
```

```
Stock insuficiente o producto inexistente

Completion time: 2025-08-02T01:02:33.8062640-05:00
```

12. FUNCIONES

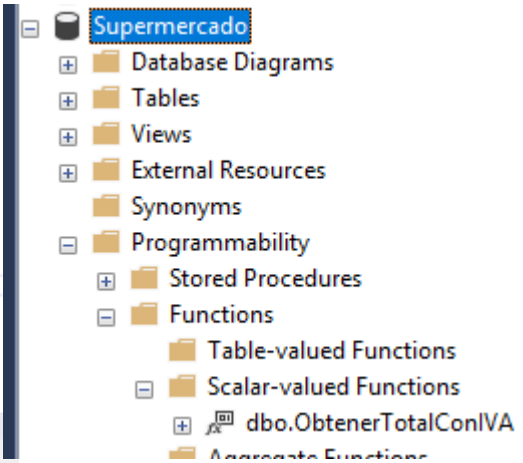
1. Función para calcular el IVA de una compra.

```
-- 1.   Función para calcular el IVA de una compra.

CREATE FUNCTION dbo.ObtenerTotalConIVA( @id_cliente INT,
    @porcentaje_iva DECIMAL(5,2))
RETURNS DECIMAL(10,2)
AS
BEGIN
    DECLARE @total_con_iva DECIMAL(10,2);
    IF @porcentaje_iva < 0 OR @porcentaje_iva > 100
    BEGIN
        RETURN 0;
    END
    SELECT @total_con_iva = SUM(v.total) * (1 + (@porcentaje_iva/100))
    FROM venta v
    WHERE v.id_cliente = @id_cliente;
    RETURN ISNULL(@total_con_iva, 0);
END;
```

83 %

Messages

```
Commands completed successfully.

Completion time: 2025-07-27T21:57:27.9889360-05:00
```




```
-- Uso de la función dbo.ObtenerTotalConIVA
SELECT dbo.ObtenerTotalConIVA(2, 12.00) AS Total_Con_IVA;
```

83 %

Results | Messages

	Total_Con_IVA
1	8.85

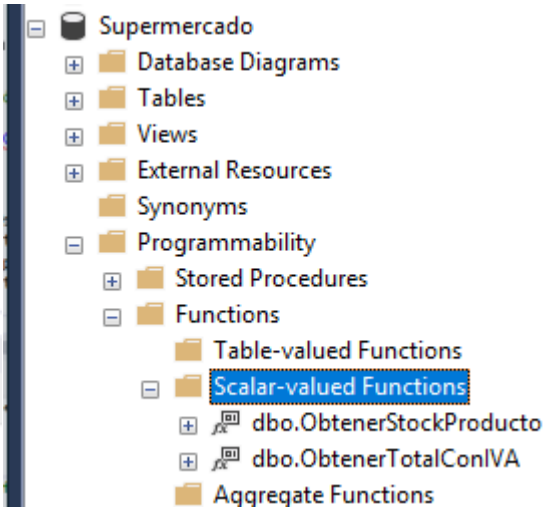
2. Función para consultar el stock del producto.

```
-- 2.   Función para consultar el stock del producto.

CREATE FUNCTION dbo.ObtenerStockProducto(@id_producto INT)
RETURNS INT
AS
BEGIN
    DECLARE @stock INT
    SELECT @stock = stock_actual FROM inventario
    WHERE id_producto = @id_producto
    RETURN @stock
END;
```

83 %

Messages

```
Commands completed successfully.

Completion time: 2025-07-27T22:14:41.9618110-05:00
```



```sql
-- Uso de la función ObtenerStockProducto

SELECT dbo.ObtenerStockProducto(8) AS Stock_Producto;
```

83 %

Results | Messages

	Stock_Producto
1	40

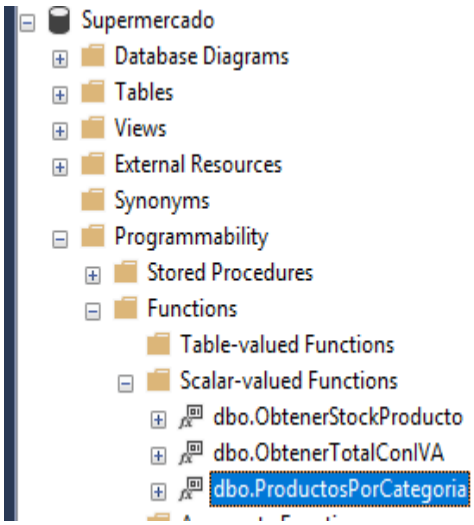
3. Función para obtener el total de productos por categoría.

```sql
-- 3.   Función para obtener el total de productos por categoría.

CREATE FUNCTION dbo.ProductosPorCategoria(@id_categoria INT)
RETURNS INT
AS
BEGIN
    DECLARE @cantidad INT
    SELECT @cantidad = COUNT(*)
    FROM producto
    WHERE id_categoria = @id_categoria
    RETURN @cantidad
END;
```

83 %

Messages

```
Commands completed successfully.

Completion time: 2025-07-27T22:23:01.1273401-05:00
```




```sql
-- Uso de la función dbo.ProductosPorCategoria

SELECT dbo.ProductosPorCategoria(10) AS Total_Productos;
```

Results | Messages

	Total_Productos
1	2

13. **TRIGGERS**

1. Creación de una tabla para auditorias.

```
-- TRIGGERS

-- Creacion de una tabla para auditorias
CREATE TABLE log_auditoria_productos (
    id_log INT IDENTITY(1,1) PRIMARY KEY,
    id_producto INT NOT NULL,
    accion VARCHAR(10) NOT NULL,
    fecha DATETIME NOT NULL DEFAULT GETDATE(),
    usuario VARCHAR(100) NOT NULL DEFAULT SYSTEM_USER,
    datos_anteriores NVARCHAR(MAX),
    datos_nuevos NVARCHAR(MAX)
);
```

83 %

Messages

```
Commands completed successfully.

Completion time: 2025-07-27T22:39:24.3720972-05:00
```

2. Trigger para registrar todas las acciones realizadas en la tabla producto

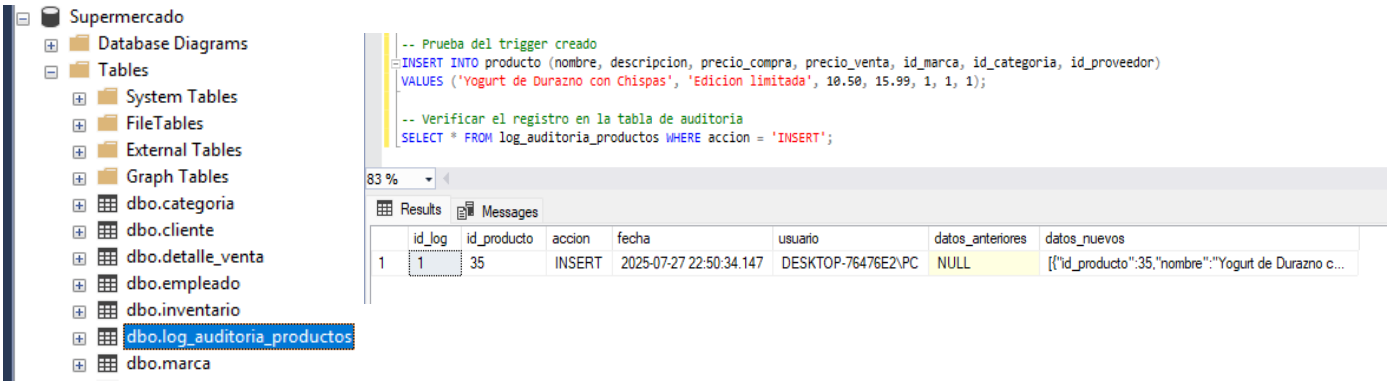
Insercion_Datos_Ta...P-76476E2\PC (63))* | Tablas_BD_Proyecto...P-76476E2\PC (55)) | Proinsert.sq

```
-- 2.  Trigger para registrar todas las acciones realizadas en la tabla producto

CREATE TRIGGER tr_auditoria_productos ON producto
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
    SET NOCOUNT ON;

    -- Para Actualizaciones
    IF EXISTS (SELECT * FROM inserted) AND EXISTS (SELECT * FROM deleted)
    BEGIN
        INSERT INTO log_auditoria_productos (id_producto, accion, datos_anteriores,
            datos_nuevos)
        SELECT d.id_producto,'UPDATE', (SELECT d.* FOR JSON PATH), (SELECT i.* FOR JSON PATH)
        FROM deleted d
        JOIN inserted i ON d.id_producto = i.id_producto;
    END

    -- Para Inserciones
    ELSE IF EXISTS (SELECT * FROM inserted)
    BEGIN
        INSERT INTO log_auditoria_productos (id_producto, accion, datos_nuevos)
        SELECT id_producto, 'INSERT', (SELECT i.* FOR JSON PATH)
        FROM inserted i;
    END

    -- Para Eliminaciones
    ELSE IF EXISTS (SELECT * FROM deleted)
    BEGIN
```

83 %

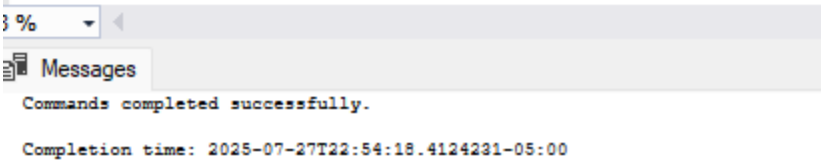
Messages

```
Commands completed successfully.

Completion time: 2025-07-27T22:42:26.0134167-05:00
```

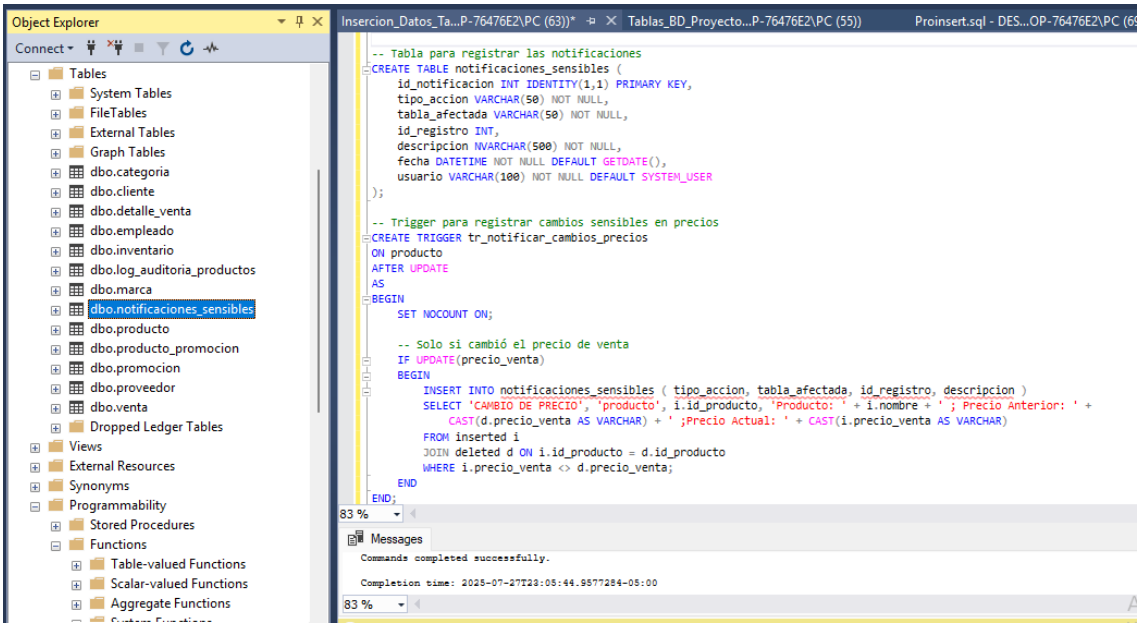


3. Trigger para controlar el stock.

```sql
-- Trigger para controlar el stock

CREATE TRIGGER tr_actualizar_inventario ON detalle_venta
AFTER INSERT
AS
BEGIN
    SET NOCOUNT ON;
    UPDATE i
    SET stock_actual = i.stock_actual - d.cantidad
    FROM inventario i
    JOIN inserted d ON i.id_producto = d.id_producto;

    -- Validacion para que no haya stock negativo
    IF EXISTS (
        SELECT 1 FROM inventario i
        JOIN inserted d ON i.id_producto = d.id_producto
        WHERE i.stock_actual < 0
    )
    BEGIN
        RAISERROR('Stock insuficiente para la venta.', 16, 1);
        ROLLBACK TRANSACTION;
    END
END;
```



4. Trigger para notificaciones sencibles




```
-- Prueba del Trigger
UPDATE producto
SET precio_venta = precio_venta * 1.2
WHERE id_producto = 14;

SELECT * FROM notificaciones_sensibles;
```

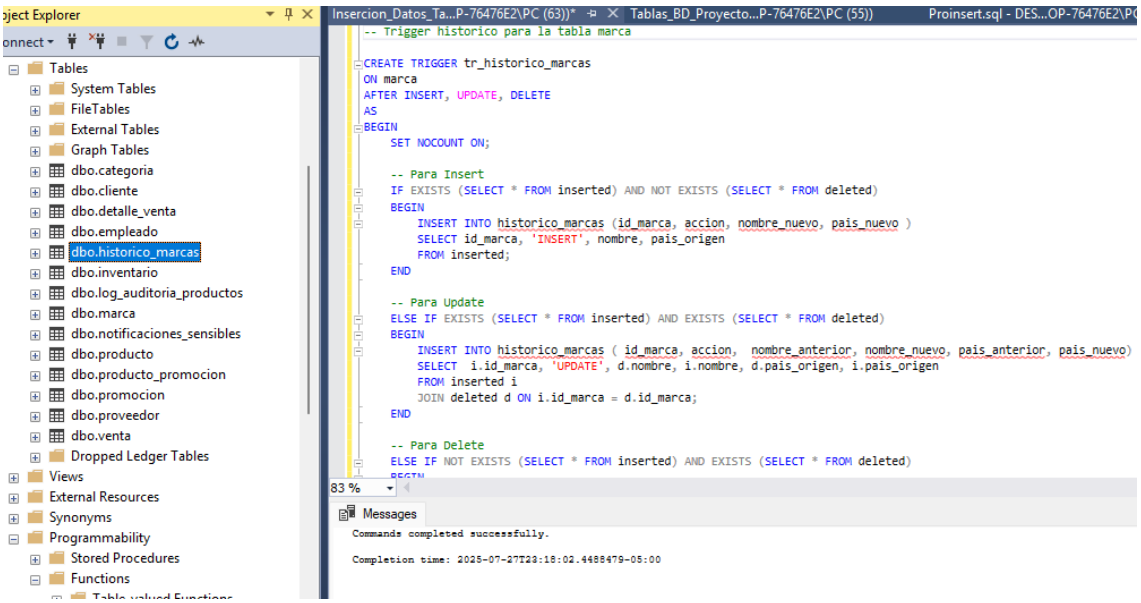
83 %

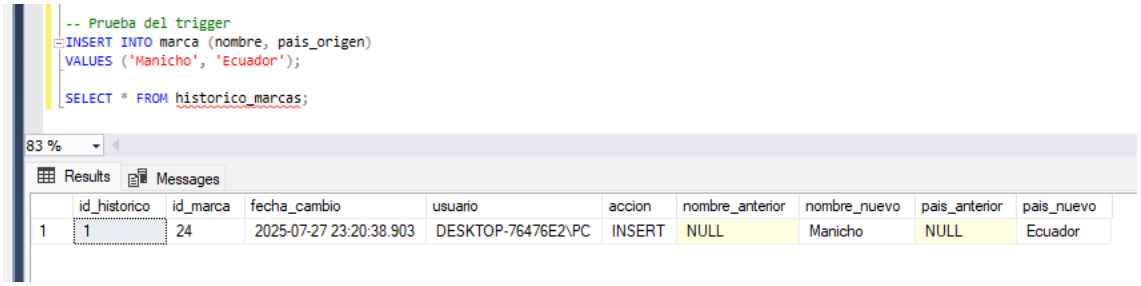
Results | Messages

	id_notificacion	tipo_accion	tabla_afectada	id_registro	descripcion	fecha	usuario
1	1	CAMBIO DE PRECIO	producto	14	Producto: Lechuga Fresca ; Precio Anterior: 0.9...	2025-07-27 23:06:57.257	DESKTOP-76476E2\PC

5. Trigger histórico para la tabla marcas.

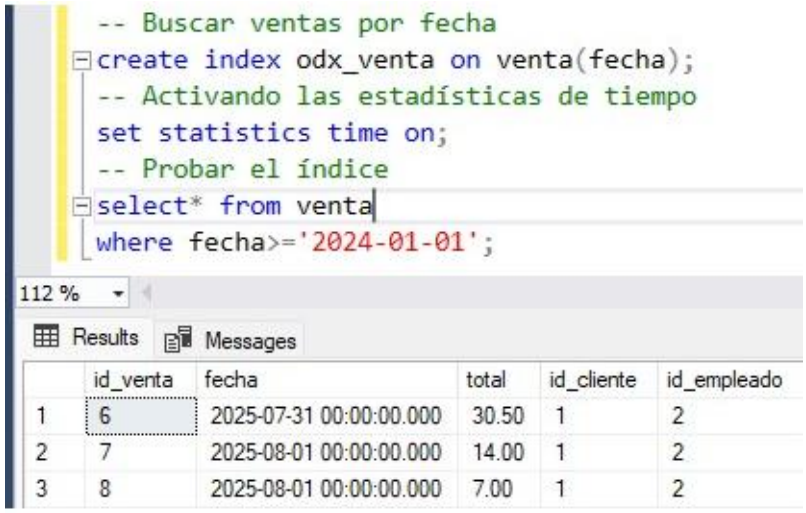
```
-- Creacion de la tabla para el registro
CREATE TABLE historico_marcas (
    id_historico INT IDENTITY(1,1) PRIMARY KEY,
    id_marca INT NOT NULL,
    fecha_cambio DATETIME NOT NULL DEFAULT GETDATE(),
    usuario VARCHAR(100) NOT NULL DEFAULT SYSTEM_USER,
    accion VARCHAR(10) NOT NULL, -- 'INSERT', 'UPDATE', 'DELETE'
    nombre_anterior VARCHAR(100),
    nombre_nuevo VARCHAR(100),
    pais_anterior VARCHAR(50),
    pais_nuevo VARCHAR(50)
);
```





14. ÍNDICES Y OPTIMIZACIÓN

1. Buscar ventas por fecha.



Índice para Joins

2. Índice para buscar productos.

```
-- INDICE PARA JOIN
-- Joins por producto
create index idx_producto on detalle_venta(id_producto);
-- Activando las estadísticas de tiempo
set statistics time on;
-- Consulta que hace JOIN usando id_producto
select dv.*, p.nombre
from detalle_venta dv
join producto p on dv.id_producto = p.id_producto;
```

112 %

Results | Messages

	id_detalle	id_venta	id_producto	cantidad	precio_unitario	subtotal	nombre
1	1	1	1	2	1.20	2.40	Leche Entera Vita
2	2	1	6	1	0.50	0.50	Agua Mineral Tesalia
3	3	1	11	1	4.90	4.90	Pechuga de Pollo Pronaca
4	4	2	2	1	0.75	0.75	Yogurt Natural La Favorita
5	5	2	7	2	1.00	2.00	Gaseosa Coca-Cola
6	6	3	3	1	3.50	3.50	Queso Fresco Pronaca
7	7	3	8	1	1.80	1.80	Jugo de Naranja Tang

Índice para order by

3. Ordenar por precio.

```
-- INDICE PARA ORDER BY
--Ordenar por precio
create index idx_precio on producto(precio_venta);
-- Activando las estadísticas de tiempo
set statistics time on;
-- Consulta que ordena por precio de venta
select * from producto
order by precio_venta desc;
```

Results | Messages

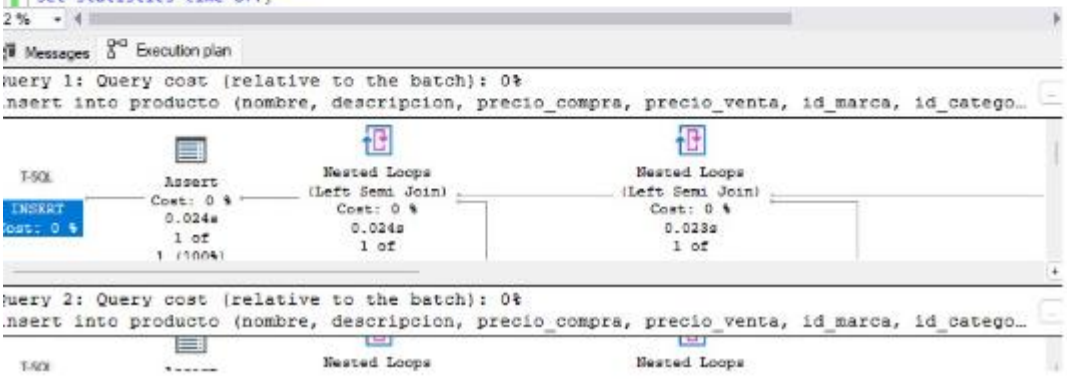
	id_producto	nombre	descripcion	precio_compra	precio_venta	id_m
1	34	Licuadora Oster	Licuadora 600W	25.00	35.00	10
2	33	Ventilador de Mesa	Ventilador 16"	15.00	22.00	19
3	29	Pañales Huggies	Pañales etapa 3 30un	8.50	12.00	11
4	30	Toallitas Huggies	Toallitas húmedas 80un	7.80	11.00	11
5	31	Croquetas Dog Chow	Alimento para perro 3kg	6.50	9.00	11
6	32	Alimento Cat Chow	Alimento para gato 2kg	5.80	8.20	16
7	21	Pizza Congelada Polar	Pizza de pepperoni 400g	4.20	6.00	8
8	12	Carne Molida La Favorita	Carne molida de res 500g	4.20	5.80	10
9	22	Helado Toni	Helado de vainilla 1L	3.80	5.40	4
10	17	Shampoo Pantene	Shampoo reparación 400ml	3.50	4.90	15
11	11	Pechuga de Pollo Pronaca	Pechuga fresca por kg	3.50	4.90	9
12	10	Jabón Líquido Dove	Jabón líquido 250ml	2.80	3.90	16

ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

Simulación de carga con 500+ registros

```
-- 4) Simular carga con 500+ registros y medir tiempos antes/después de los índices.
declare @i int = 1;
while @i <= 500
begin
    insert into producto (nombre, descripcion, precio_compra, precio_venta, id_marca, id_categoria, id_p
    values (
        CONCAT('ProductoPrueba', @i),
        'Descripción genérica',
        1.00,
        5.00 + (@i % 10),
        1,
        1,
        1
    );
    set @i = @i + 1;
end;

-- Eliminamos el indice que creamos anteriormente
drop index if exists idx_precio on producto;

-- Medimos el tiempo de la consulta que usa precio_venta
set statistics time on;
select * from producto where precio_venta>10;
set statistics time off;

-- Creamos de nuevo el indice
create index idx_precio_producto on producto(precio_venta);

-- Medimos el tiempo con índice
set statistics time on;
select * from producto where precio_venta>10;
set statistics time off;
```



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

Medición de tiempos antes y después de los índices

```
-- Medimos el tiempo de la consulta que usa precio_venta
set statistics time on;
select * from producto where precio_venta>10;
set statistics time off;
```

23 %

Results | Messages | Execution plan

	id_producto	nombre	descripcion	precio_compra	precio_venta	id_marca	id_categoria	id_proveedor
1	29	Pañales Huggies	Pañales etapa 3 30un	8.50	12.00	11	13	6
2	30	Toallitas Huggies	Toallitas húmedas 80un	7.80	11.00	11	13	6
3	33	Ventilador de Mesa	Ventilador 16"	15.00	22.00	19	15	9
4	34	Licuadora Oster	Licuadora 600W	25.00	35.00	18	15	9
5	40	ProductoPrueba6	Descripción genérica	1.00	11.00	1	1	1
6	41	ProductoPrueba7	Descripción genérica	1.00	12.00	1	1	1

```
-- Medimos el tiempo de la consulta que usa precio_venta
set statistics time on;
select * from producto where precio_venta>10;
set statistics time off;
```

123 %

Results | Messages | Execution plan

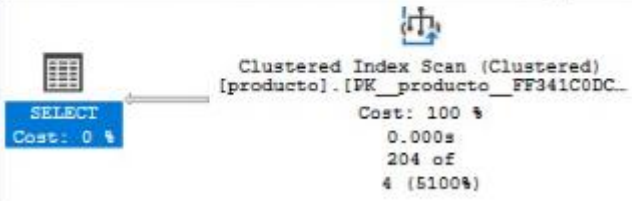
```
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.

(204 rows affected)

(1 row affected)

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 142 ms.

Completion time: 2025-08-02T13:37:36.2884407-05:00
```

```
-- Medimos el tiempo de la consulta que usa precio_venta
set statistics time on;
select * from producto where precio_venta>10;
set statistics time off;
```

123 %

Results | Messages | Execution plan

```
Query 1: Query cost (relative to the batch): 100%
SELECT * FROM [producto] WHERE [precio_venta]>@1
```



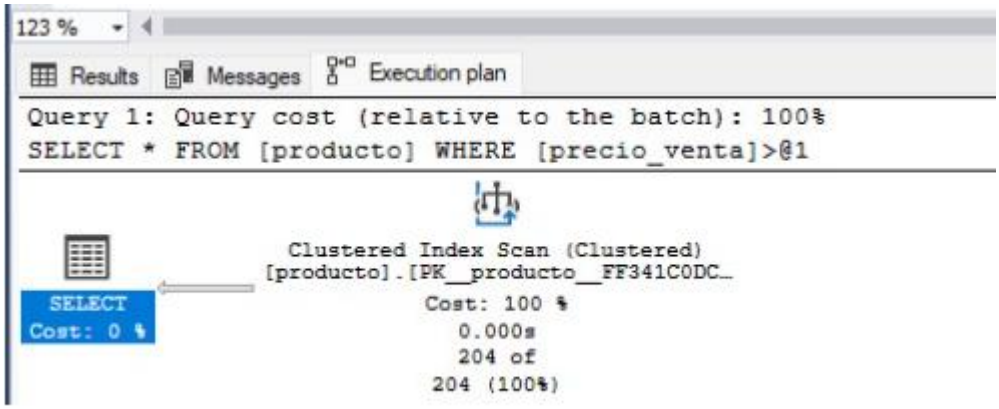
Medición del tiempo con índice

```
-- Medimos el tiempo con índice
set statistics time on;
select * from producto where precio_venta>10;
set statistics time off;
```

23 %

Results | Messages | Execution plan

```
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 1 ms.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.

(204 rows affected)

(1 row affected)

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 134 ms.

Completion time: 2025-08-02T13:40:29.2910560-05:00
```



4. Índices compuestos para combinar comunes

```
-- 2) Crear índices compuestos para combinaciones comunes.
-- Búsquedas por id_venta + id_producto
create index idx_detalle_venta_compu on detalle_venta(id_venta, id_producto);
-- Activando las estadísticas de tiempo
set statistics time on;
-- Consulta para la verificación de el procedimiento
select * from detalle_venta
where id_venta=5 and id_producto=10;
```

```
Messages
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.

Completion time: 2025-08-02T12:27:33.8056231-05:00
```

Results | Messages

	id_detalle	id_venta	id_producto	cantidad	precio_unitario	subtotal
1	12	5	10	1	2.00	2.00

- Consultar ventas por fecha.

```
-- Consultar ventas por fecha y total
create index idx_venta_fecha_total on venta(fecha, total);
-- Activando las estadísticas de tiempo
set statistics time on;
-- Verificando el funcionamiento del procedimiento
select * from venta
where fecha>='2025-08-01' order by total desc;
```

```
Messages
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.

Completion time: 2025-08-02T12:31:02.5994885-05:00
```

112 %

Results Messages

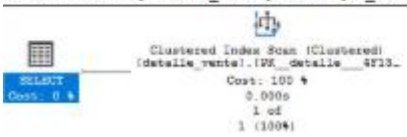
	id_venta	fecha	total	id_cliente	id_empleado
1	7	2025-08-01 00:00:00.000	14.00	1	2
2	8	2025-08-01 00:00:00.000	7.00	1	2

Analizar rendimiento con EXPLAIN

```
-- 3) Analizar rendimiento con EXPLAIN o herramientas gráficas según el motor.
select * from detalle_venta
where id_venta=5 and id_producto=10;
```

Results Messages Execution plan

```
Query 1: Query cost (relative to the batch): 100%
SELECT * FROM [detalle_venta] WHERE [id_venta]=@1 AND [id_producto]=@2
```



15. SEGURIDAD Y ROLES

1. Creación de logins.

```
-- Creacion de logins a nivel de servidor
CREATE LOGIN usuario_admin WITH PASSWORD = 'admin@123';
CREATE LOGIN usuario_auditor WITH PASSWORD = 'auditor@123';
CREATE LOGIN usuario_operador WITH PASSWORD = 'operador@123';
CREATE LOGIN usuario_cliente WITH PASSWORD = 'cliente@123';
CREATE LOGIN usuario_proveedor WITH PASSWORD = 'proveedor@123';
```

```
99 %
Messages
Commands completed successfully.

Completion time: 2025-08-02T23:24:46.3323265-05:00
```

2. Creación de usuarios.

```
-- Creacion de usuarios
CREATE USER usuario_admin FOR LOGIN usuario_admin;
CREATE USER usuario_auditor FOR LOGIN usuario_auditor;
CREATE USER usuario_operador FOR LOGIN usuario_operador;
CREATE USER usuario_cliente FOR LOGIN usuario_cliente;
CREATE USER usuario_proveedor FOR LOGIN usuario_proveedor;
GO

-- Roles con permisos personalizados
-- Administrador
```

```
82 %
Messages
Commands completed successfully.

Completion time: 2025-08-03T02:54:04.3101240-05:00
```

3. Asignación de permisos a los diferentes roles.

Administrador

```
-- Roles con permisos personalizados
-- Administrador
CREATE ROLE rol_administrador;
GRANT CONTROL ON DATABASE::Supermercado TO rol_administrador;
GO
```

Auditor

```
-- Auditor
CREATE ROLE rol_auditor;
GRANT SELECT ON SCHEMA::dbo TO rol_auditor;
GRANT SELECT ON log_auditoria_productos TO rol_auditor;
GRANT SELECT ON notificaciones_sensibles TO rol_auditor;
GRANT SELECT ON historico_marcas TO rol_auditor;
GO
```

Operador

```
-- Operador
CREATE ROLE rol_operador;
GRANT SELECT, INSERT, UPDATE ON producto TO rol_operador;
GRANT SELECT, INSERT, UPDATE ON inventario TO rol_operador;
GRANT SELECT, INSERT ON venta TO rol_operador;
GRANT SELECT, INSERT ON detalle_venta TO rol_operador;
GRANT SELECT ON cliente TO rol_operador;
GRANT SELECT ON empleado TO rol_operador;
GO
```

Cliente

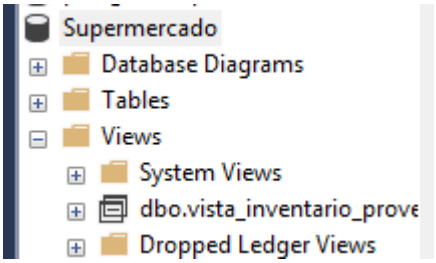
```
-- Cliente
CREATE ROLE rol_cliente;
GRANT SELECT ON producto TO rol_cliente;
GRANT SELECT ON promocion TO rol_cliente;
GRANT SELECT ON producto_promocion TO rol_cliente;
GO
```

Proveedor

```
-- Proveedor
CREATE ROLE rol_proveedor;
GO
```

- **Creación de vista para reducir la información a la que tiene acceso el proveedor**

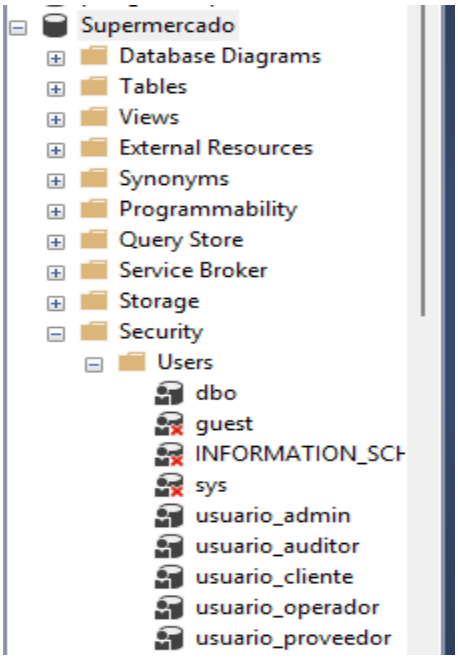
```
-- Vista para limitar el acceso del proveedor
CREATE VIEW vista_inventario_proveedor AS
SELECT p.id_producto, p.nombre, i.stock_actual
FROM producto p
JOIN inventario i ON p.id_producto = i.id_producto
WHERE p.nombre = SYSTEM_USER;
GO
```



- **Permisos.**

```
GRANT SELECT ON vista_inventario_proveedor TO rol_proveedor;
GO
```

Usuarios completos



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

Asignación de roles

```
-- Asignacion de usuarios a roles
ALTER ROLE rol_administrador ADD MEMBER usuario_admin;
ALTER ROLE rol_auditor ADD MEMBER usuario_auditor;
ALTER ROLE rol_operador ADD MEMBER usuario_operador;
ALTER ROLE rol_cliente ADD MEMBER usuario_cliente;
ALTER ROLE rol_proveedor ADD MEMBER usuario_proveedor;
GO
```

```
-- Usuario final
CREATE ROLE rol_usuario_final;
GRANT SELECT ON producto TO rol_usuario_final;
GRANT SELECT ON promocion TO rol_usuario_final;
GO
```

82 %

Messages

```
Commands completed successfully.

Completion time: 2025-08-03T03:03:39.6635829-05:00
```

- **Para mayor seguridad**

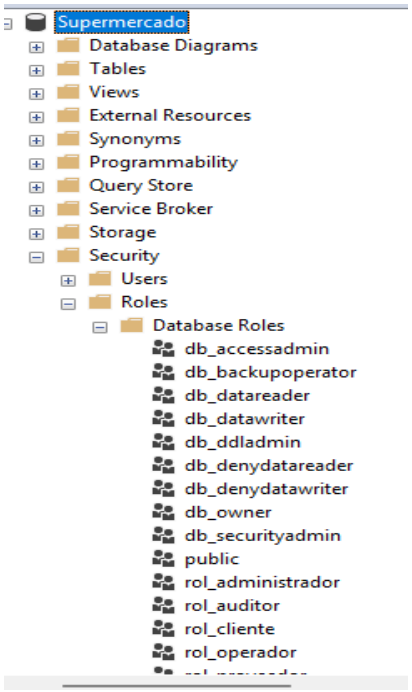
```
-- Restriccion para mayor seguridad
DENY VIEW DEFINITION TO usuario_cliente, usuario_proveedor
GO
```

2 %

Messages

```
Commands completed successfully.

Completion time: 2025-08-03T03:04:26.7906683-05:00
```

Roles completos



16. ENCRIPTACION DE DATOS

- **Definición de clave maestra.**

```
-- ENCRIPTACION DE DATOS
-- Clave para la base de datos
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'Super@123';
```

82 %

Messages

```
Commands completed successfully.

Completion time: 2025-08-03T03:12:36.4244063-05:00
```

- **Alteración de la tabla cliente**

```
-- Se agregan columnas en la tabla de clientes actual para visualizar la encriptacion de los campos
ALTER TABLE cliente
ADD cedula_cifrada VARBINARY(MAX), email_cifrado VARBINARY(MAX), telefono_cifrado VARBINARY(MAX);
GO

UPDATE cliente
```

82 %

Messages

```
Commands completed successfully.

Completion time: 2025-08-03T03:13:09.1183761-05:00
```

- **Prueba de encriptación.**

```
UPDATE cliente
SET
    cedula_cifrada = ENCRYPTBYPASSPHRASE('Super@123', cedula),
    email_cifrado = ENCRYPTBYPASSPHRASE('Super@123', email),
    telefono_cifrado = ENCRYPTBYPASSPHRASE('Super@123', telefono);
GO
```

82 %

Messages

```
(5 rows affected)

Completion time: 2025-08-03T03:14:15.1394998-05:00
```

```
-- Prueba de la encriptacion
select * from cliente;
```

Results | Messages

	id_cliente	nombre	cedula	email	telefono	cedula_cifrada	email_cifrado	telefono_cifrado
1	1	Juan Pérez	1701234567	juan.perez@gmail.com	0991234567	0x020000006ED66E64CDBB521F1D88734EA4CAB991E269D8...	0x0200000092B6062ACA61C15DDFD76E00726A26D68BDCF4...	0x020000001FD8B1EF60D111
2	2	María Gómez	1702345678	maria.gomez@outlook.com	0992345678	0x020000008B78EDD7BFD5E954F7EB32C823C19A3A4323077...	0x02000000F205FB74CC9D935A2EFA5C8CFAF5973C4EC698D...	0x02000000CF1AFCCB69E20D
3	3	Carlos Andrade	1703456789	carlos.andrade@gmail.com	0993456789	0x020000000C5166D105604928E02F84B7294EFFC78F1324B7B...	0x02000000BA5AF57BAA7C55CF6F59C70EB1B1C5D7BFB63B4...	0x020000006997B14760C2E8A
4	4	Ana Beltrán	1704567890	ana.beltran@yahoo.com	0994567890	0x020000001BAF775852E78FD5EF02CD53BF9B5298C8DD897...	0x02000000168F32074055DE153A832333A7FC58D939A0297D...	0x020000004283AEE03BD4DD
5	5	Luis Vásquez	1705678901	luis.vasquez@hotmail.com	0995678901	0x02000000CD139C123F7FCA921D8B442DE66B8ADAC29601...	0x02000000BF39E13B0E83332AF7C20168DBAE1792F68A9317...	0x02000000C6B685A784A5D5

17. DESCIFRADO DE DATOS

- Los campos cedula, email y teléfono de la tabla cliente que antes fueron encriptados, ahora son descifrados.

```
-- Descifrado de datos
SELECT id_cliente, nombre,
CONVERT(nvarchar, DECRYPTBYPASSPHRASE('Super@123', cedula_cifrada)) AS cedula,
CONVERT(nvarchar, DECRYPTBYPASSPHRASE('Super@123', email_cifrado)) AS email,
CONVERT(nvarchar, DECRYPTBYPASSPHRASE('Super@123', telefono_cifrado)) AS telefono
FROM cliente;
GO
```

82 %

Results | Messages

	id_cliente	nombre	cedula	email	telefono
1	1	Juan Pérez	1701234567	juan.perez@gmail.com	0991234567
2	2	María Gómez	1702345678	maria.gomez@outlook.com	0992345678
3	3	Carlos Andrade	1703456789	carlos.andrade@gmail.com	0993456789
4	4	Ana Beltrán	1704567890	ana.beltran@yahoo.com	0994567890
5	5	Luis Vásquez	1705678901	luis.vasquez@hotmail.com	0995678901

18. SIMULACIÓN DE CONEXIÓN SEGURA

```
-- Se habilitan las opciones avanzadas que permitiran la simulacion de conexion segura
EXEC sp_configure 'show advanced options', 1;
RECONFIGURE;
GO
```

```
Messages
Se ha cambiado la opción de configuración 'show advanced options' de 1 a 1. Ejecute la instrucci

Completion time: 2025-08-03T03:18:30.3993695-05:00
```

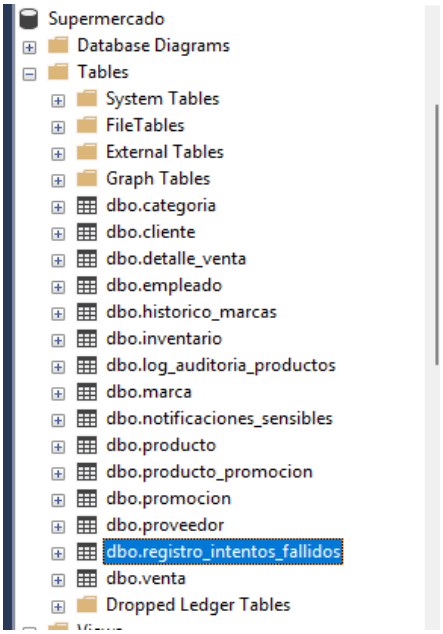
- Para conseguir que funcione la simulación de conexión segura se deben realizar una serie de configuraciones dentro de

```
-- SIMULACIÓN DE CONEXIÓN SEGURA (SSL/TLS)
-- EXEC sp_configure 'force protocol encryption', 1;
-- RECONFIGURE;
-- GO
```

Registros de intentos fallidos

```
--  REGISTRO DE INTENTOS FALLIDOS O SOSPECHOSOS
CREATE TABLE registro_intentos_fallidos (
    id INT IDENTITY(1,1) PRIMARY KEY,
    usuario NVARCHAR(100),
    evento NVARCHAR(255),
    fecha DATETIME DEFAULT GETDATE()
);
GO

-- Ejemplo de inserción simulada
INSERT INTO registro_intentos_fallidos (usuario, evento
```

```
Messages
Commands completed successfully.

Completion time: 2025-08-03T03:24:08.9804605-05:00
```



- **Inserción simulada**

```
-- Ejemplo de inserción simulada
INSERT INTO registro_intentos_fallidos (usuario, evento)
VALUES ('usuario_cliente', 'Intento de acceso a tabla inventario sin privilegios');
```

```
Messages

(1 row affected)

Completion time: 2025-08-03T03:25:51.6166552-05:00
```

- **Prueba**

```
select * from registro_intentos_fallidos;
```

Results | Messages

	id	usuario	evento	fecha
1	1	usuario_cliente	Intento de acceso a tabla inventario sin privile...	2025-08-03 03:25:51.607

19. AUDITORÍA

```
-- AUDITORÍA DE PRIVILEGIOS ACTIVOS Y ROLES
SELECT r.name AS rol, m.name AS usuario
FROM sys.database_principals r
JOIN sys.database_role_members rm ON r.principal_id = rm.role_principal_id
JOIN sys.database_principals m ON rm.member_principal_id = m.principal_id;
```

Results | Messages

	rol	usuario
1	rol_administrador	usuario_admin
2	rol_auditor	usuario_auditor
3	rol_operador	usuario_operador
4	rol_cliente	usuario_cliente
5	rol_proveedor	usuario_proveedor
6	db_owner	dbo

- Tabla para la auditoría.

```
-- AUDITORÍA
-- Tabla Log
CREATE TABLE log_acciones (
    id_log INT PRIMARY KEY IDENTITY(1,1),
    usuario NVARCHAR(100),
    ip_origen NVARCHAR(100),
    fecha DATETIME DEFAULT GETDATE(),
    accion NVARCHAR(20),
    tabla NVARCHAR(100),
    id_afectado INT,
    rol_activo NVARCHAR(100),
    transaccion NVARCHAR(MAX),
    hash_cambio AS CONVERT(VARCHAR(64), HASHBYTES('SHA2_256',
    CONCAT(usuario, accion, tabla, id_afectado,
    CONVERT(VARCHAR, fecha, 120))), 2) PERSISTED
);
GO
```

```
68 %
Messages
Commands completed successfully.

Completion time: 2025-08-03T03:29:54.3333884-05:00
```

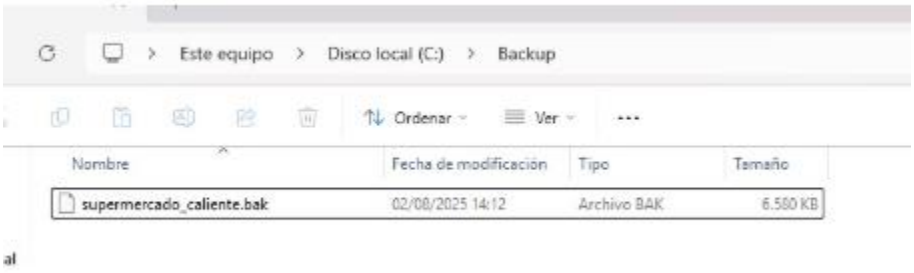
20. RESPALDO Y RESTAURACIÓN

BACKUP EN CALIENTE

```
-- BACKUP EN CALIENTE
use master;
go

-- Realiza un backup de la base de datos Supermercado
backup database Supermercado
to disk = 'C:\Backup\supermercado_caliente.bak'
with format, name = 'Backup en caliente de Supermercado';
```

```
123 %
Messages
Processed 808 pages for database 'Supermercado', file 'Supermercado' on file 1.
Processed 2 pages for database 'Supermercado', file 'Supermercado_log' on file 1.
BACKUP DATABASE successfully processed 810 pages in 1.097 seconds (5.765 MB/sec).

Completion time: 2025-08-02T14:12:50.4130759-05:00
```

- Backup guardado.

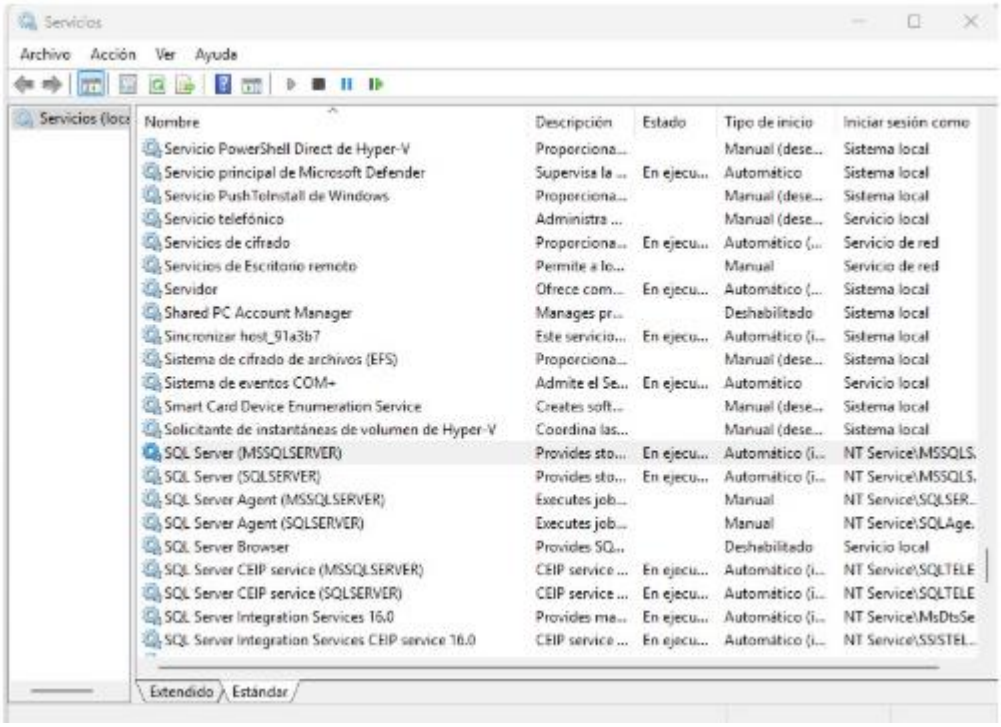


Restauración de backup en caliente

```
-- PARA LA RESTAURACIÓN EN CALIENTE
use master;
go
-- Debemos cerrar las conexiones a la base original para evitar problemas
alter database Supermercado
set single_user with rollback immediate;
-- Restauramos el backup como una nueva base, otro nombre
restore database Supermercado_Restaurada
from disk = 'C:\BackupCaliente\supermercado_caliente.bak'
with move 'Supermercado' to 'C:\BackupCaliente\Supermercado_Restaurada.mdf',
     move 'Supermercado_log' to 'C:\BackupCaliente\Supermercado_Restaurada_log.ldf';
```

```
Messages
Processed 808 pages for database 'Supermercado_Restaurada', file 'Supermercado' on file 1.
Processed 2 pages for database 'Supermercado_Restaurada', file 'Supermercado_log' on file 1.
RESTORE DATABASE successfully processed 810 pages in 0.289 seconds (21.883 MB/sec).

Completion time: 2025-08-02T19:11:12.3338339-05:00
```

- Se cierra la conexión



Confirmación de la restauración

```
-- Para confirmar su correcta restauración
-- Mostrar la lista de bases de datos con sus archivos físicos y rutas
print'La base de datos Supermercado_Restaurada se restauró correctamente.';
select
    db.name as NombreBaseDatos,
    mf.physical_name as RutaArchivo,
    mf.type_desc as TipoArchivo
from
    sys.master_files mf
inner join
    sys.databases db on mf.database_id = db.database_id
where
    db.name = 'Supermercado_Restaurada';
```

23 %

Results | Messages

	NombreBaseDatos	RutaArchivo	TipoArchivo
1	Supermercado_Restaurada	C:\BackupCaliente\Supermercado_Restaurada.mdf	ROWS
2	Supermercado_Restaurada	C:\BackupCaliente\Supermercado_Restaurada_log.ldf	LOG

%

Results | Messages

```
La base de datos Supermercado_Restaurada se restauró correctamente.

(2 rows affected)

Completion time: 2025-08-02T19:17:05.0397295-05:00
```

ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

Visualización del contenido de la base de datos restaurada

```
-- Vemos que nomás tiene nuestra base de datos restaurada
use Supermercado_Restaurada;
go
select top 10 * from cliente;
```

23 %

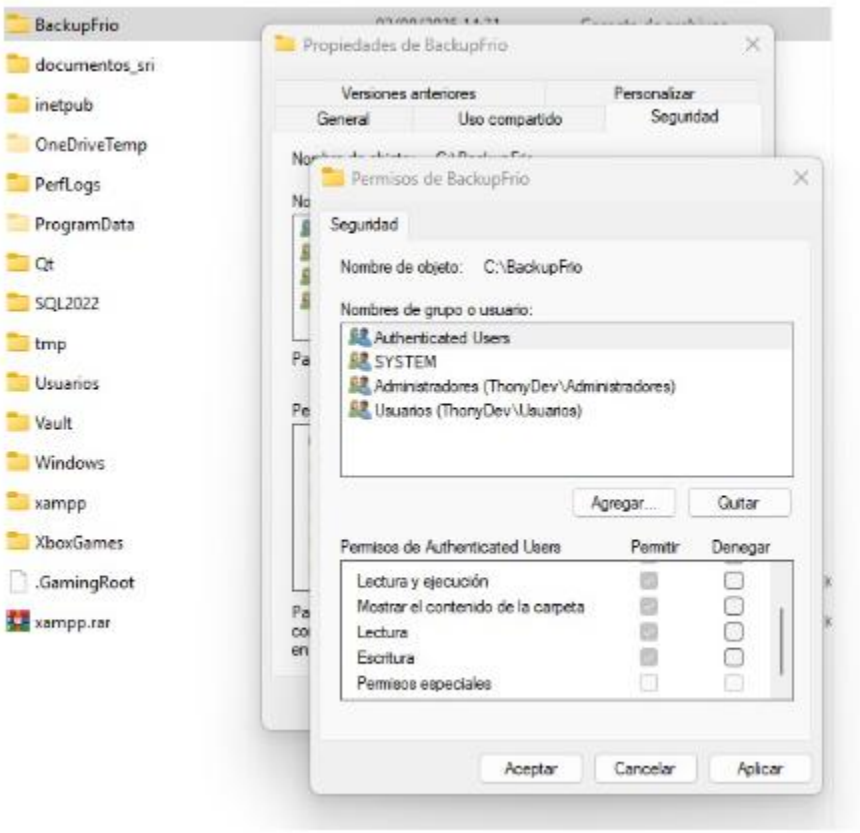
Results | Messages

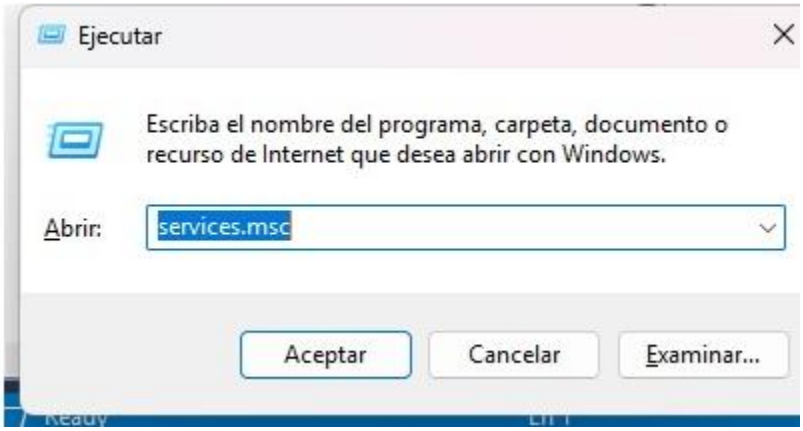
	id_cliente	nombre	cedula	email	telefono
1	1	Juan Pérez	1701234567	juan.perez@gmail.com	0991234567
2	2	María Gómez	1702345678	maria.gomez@outlook.com	0992345678
3	4	Ana Beltrán	1704567890	ana.beltran@yahoo.com	0994567890
4	5	Luis Vásquez	1705678901	luis.vasquez@hotmail.com	0995678901

ESCUELA POLITÉCNICA NACIONAL

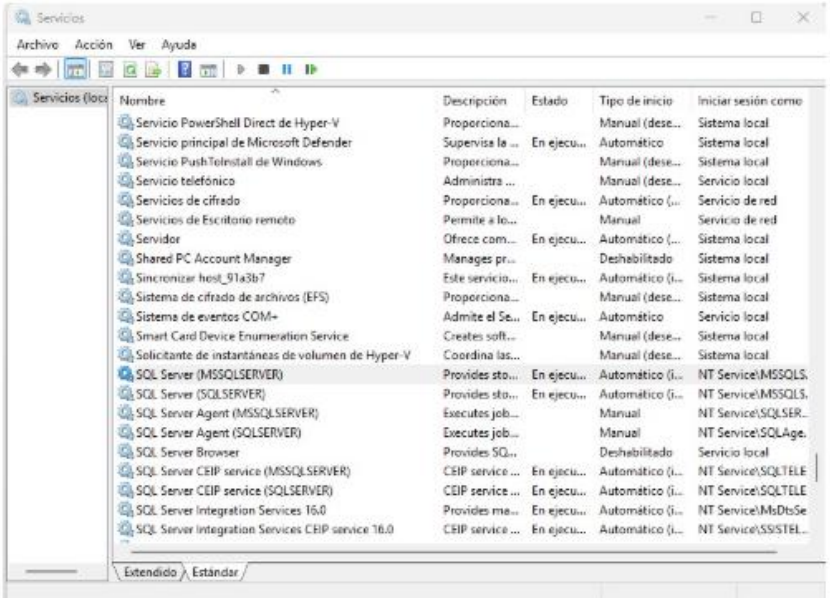
ESCUELA DE FORMACIÓN DE TECNÓLOGOS

BACKUP EN FRÍO

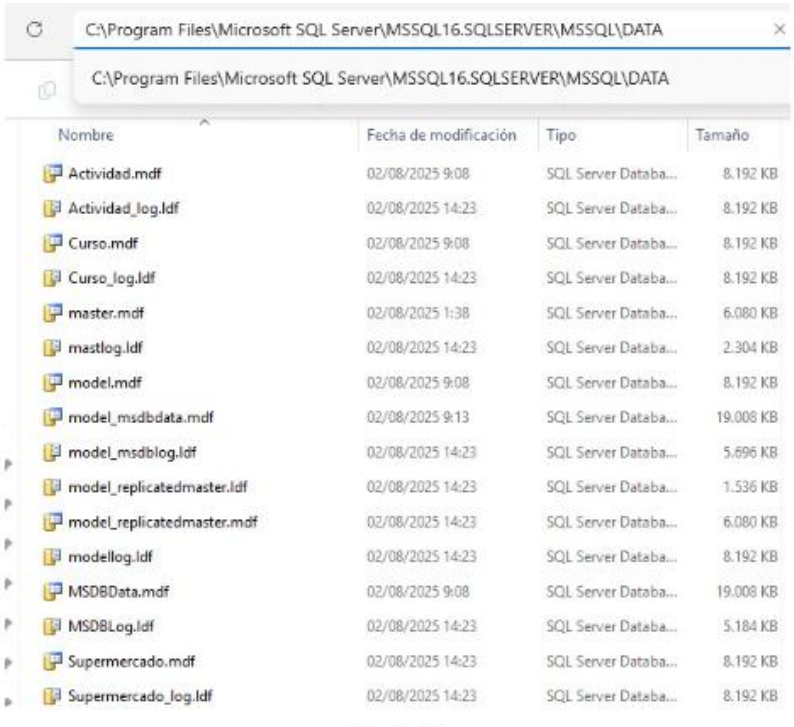




- **Se reactiva la conexión.**



- **Acedemos al lugar donde están guardadas todas las bases de datos.**



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

Nombre	Fecha de modificación	Tipo	Tamaño
Supermercado.mdf	02/08/2025 14:30	SQL Server Databa...	8.192 K
Supermercado_log.ldf	02/08/2025 14:30	SQL Server Databa...	8.192 K

Para reactivar la conexión a la base de datos

```
-- Para reactivar la conexión de mi base de datos
alter database Supermercado
set multi_user;
go

-- Para comprobar que si se reactivo correctamente
use Supermercado;
go
select top 5 * from cliente;
```

123 %

Results | Messages

	id_cliente	nombre	cedula	email	telefono
1	1	Juan Pérez	1701234567	juan.perez@gmail.com	0991234567
2	2	María Gómez	1702345678	maria.gomez@outlook.com	0992345678
3	4	Ana Beltrán	1704567890	ana.beltran@yahoo.com	0994567890
4	5	Luis Vásquez	1705678901	luis.vasquez@hotmail.com	0995678901

21. TRIGGERS PARA OPERACIONES CLAVE

Tabla: Cliente

- **Trigger para las diferentes acciones que curran en la tabla, este es INSERT.**

```
-- Triggers para las operaciones clave

-- Tabla cliente
-- Trigger para INSERT
CREATE TRIGGER trg_insertar_cliente ON cliente
AFTER INSERT
AS
BEGIN
    INSERT INTO log_acciones (usuario, ip_origen, accion, tabla, id_afectado, rol_activo, transaccion)
    SELECT SYSTEM_USER, HOST_NAME(), 'INSERT', 'cliente', id_cliente, USER_NAME(), 'INSERT INTO cliente'
    FROM inserted;
END;
GO

-- Trigger para UPDATE
```

```
8 %
Messages
Commands completed successfully.

Completion time: 2025-05-03T03:31:40.6546620-05:00
```

- **UPDATE**

```
-- Trigger para UPDATE
CREATE TRIGGER trg_actualizar_cliente ON cliente
AFTER UPDATE
AS
BEGIN
    INSERT INTO log_acciones (usuario, ip_origen, accion, tabla, id_afectado, rol_activo, transaccion)
    SELECT SYSTEM_USER, HOST_NAME(), 'UPDATE', 'cliente', id_cliente, USER_NAME(), 'UPDATE cliente'
    FROM inserted;
END;
GO
```

```
58 %
Messages
Commands completed successfully.

Completion time: 2025-05-03T03:34:00.3919537-05:00
```

- **DELETE**

```
-- Trigger para DELETE
CREATE TRIGGER trg_eliminar_cliente ON cliente
AFTER DELETE
AS
BEGIN
    INSERT INTO log_acciones(usuario, ip_origen, accion, tabla, id_afectado, rol_activo, transaccion)
    SELECT SYSTEM_USER, HOST_NAME(), 'DELETE', 'cliente', id_cliente, USER_NAME(), 'DELETE cliente'
    FROM inserted;
END;
GO
```

```
68 %
Messages
Commands completed successfully.

Completion time: 2025-08-03T03:34:36.2556926-05:00
```

22. REPORTES

Por usuario

```
-- REPORTE POR USUARIO, ACCION, MODULO O FECHA
-- Reporte por usuario
CREATE VIEW reporte_accion_usuario AS
SELECT usuario, accion, tabla, COUNT(*) AS total, MIN(fecha) AS primera_accion,
MAX(fecha) AS ultima_accion FROM log_acciones
GROUP BY usuario, accion, tabla;
GO
```

```
68 %
Messages
Commands completed successfully.

Completion time: 2025-08-03T03:36:21.7520179-05:00
```

Control de versiones con hash

```
-- Control de versiones con hash
CREATE TABLE bitacora_precio_producto(
    id_bitacora INT PRIMARY KEY IDENTITY(1,1),
    id_producto INT,
    precio_anterior DECIMAL(10,2),
    precio_nuevo DECIMAL(10,2),
    fecha DATETIME DEFAULT GETDATE(),
    usuario NVARCHAR(100)
);
GO
```

```
68 %
Messages
Commands completed successfully.

Completion time: 2025-08-03T03:36:59.3233149-05:00
```

Creación de trigger para registrar cambios.

```
-- Trigger para registrar los cambios
CREATE TRIGGER trg_actualizar_precio_producto ON producto
AFTER UPDATE
AS
BEGIN
    INSERT INTO bitacora_precio_producto(id_producto, precio_anterior, precio_nuevo, usuario)
    SELECT i.id_producto, d.precio_venta, i.precio_venta, SYSTEM_USER
    FROM inserted i
    JOIN deleted d ON i.id_producto=d.id_producto
    WHERE i.precio_venta <> d.precio_venta;
END;
GO

CREATE TABLE usuarios (
    id_usuario INT PRIMARY KEY IDENTITY(1,1),
```

```
68 %
Messages
Commands completed successfully.

Completion time: 2025-08-03T03:37:59.1515033-05:00
```

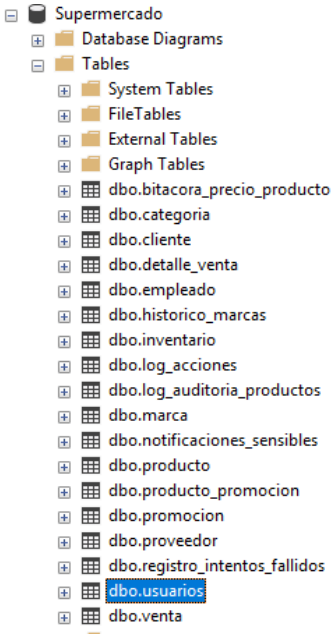
- Implementación de una nueva tabla denominada usuarios.

```
CREATE TABLE usuarios (
    id_usuario INT PRIMARY KEY IDENTITY(1,1),
    id_empleado INT NOT NULL,
    usuario NVARCHAR(100) UNIQUE NOT NULL,
    clave VARBINARY(64) NOT NULL,
    FOREIGN KEY (id_empleado) REFERENCES empleado(id_empleado)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
GO
```

68 %

Messages

```
Commands completed successfully.
```



- **Prueba**

```
-- Ejemplo de uso
INSERT INTO usuarios (id_empleado, usuario, clave)
VALUES (
    1,
    'carlos',
    HASHBYTES('SHA2_256', CONVERT(NVARCHAR(100), '12345'))
);
```

68 %

Messages

```
(1 row affected)

Completion time: 2025-05-03T03:40:12.8262315-05:00
```

```
-- Ejemplo de uso
INSERT INTO usuarios (id_empleado, usuario, clave)
VALUES (
    1,
    'carlos',
    HASHBYTES('SHA2_256', CONVERT(NVARCHAR(100), '12345'))
);

select * from usuarios;


-- Seguridad ante SQL Injection
```

68 %

Results | Messages

	id_usuario	id_empleado	usuario	clave
1	1	1	carlos	0x1445217533E7D4D0CFFD9109C4EDB60D62A02C0F0DE953.

- Seguridad para combatir un SQL Injection
- Simulación de formulario vulnerable.

```
-- Seguridad ante SQL Injection
-- Simulación de formulario vulnerable
DECLARE @usuario NVARCHAR(100) = ''' OR ''1''=''1'
DECLARE @sql NVARCHAR(MAX)

SET @sql = 'SELECT * FROM usuarios WHERE usuario = ''' + @usuario + ''''
EXEC(@sql)
GO

-- Protección con consultas
```

81 %

Results | Messages

	id_usuario	id_empleado	usuario	clave
1	1	1	carlos	0x1445217533E7D4D0CFFD9109C4EDB60D62A02C0F0DE953

- Protección con consultas.

```
-- Protección con consultas
DECLARE @usuario NVARCHAR(100) = 'admin'
DECLARE @sql NVARCHAR(MAX)

SET @sql = 'SELECT * FROM usuarios WHERE usuario =  @usr'
EXEC sp_executesql @sql, N'@usr NVARCHAR(50)', @usr = @usuario
GO
```

81 %

Results | Messages

id_usuario	id_empleado	usuario	clave

23. VALIDACIONES PREVIAS

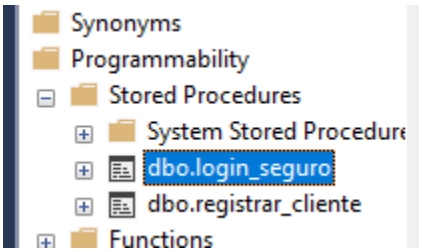
- Procedimiento para validaciones previas.

```
-
-- Validaciones previas
CREATE PROCEDURE login_seguro
    @usuario NVARCHAR(100),
    @clave NVARCHAR(100)
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @sql NVARCHAR(MAX);

    SET @sql = '
        SELECT u.usuario, e.nombre AS nombre_empleado
        FROM usuarios u
        JOIN empleado e ON u.id_empleado = e.id_empleado
        WHERE u.usuario = @userInput
          AND u.clave = HASHBYTES(''SHA2_256'', @passInput);
    ';

    EXEC sp_executesql
        @sql,
        N'@userInput NVARCHAR(100), @passInput NVARCHAR(100)',
        @userInput = @usuario,
        @passInput = @clave;
END;
GO
```

81 %

Messages

```
Commands completed successfully.

Completion time: 2025-08-03T12:21:22.6433978-05:00
```



25. MONITOREO Y RENDIMIENTO

```
use Supermercado;
go

-- MONITOREO Y RENDIMIENTO

-- Tamaño de las tablas y sus índices en la base de datos Supermercado
EXEC sp_MSforeachtable
    @command1 = 'EXEC sp_spaceused ''?''';
```

123 %

Results | Messages

	name	rows	reserved	data	index_size	unused
1	notificaciones_sensibles	1	72 KB	8 KB	0 KB	56 KB

	name	rows	reserved	data	index_size	unused
1	historico_marcas	1	72 KB	8 KB	0 KB	56 KB

	name	rows	reserved	data	index_size	unused
1	registro_intentos_fallidos	1	72 KB	8 KB	0 KB	56 KB

	name	rows	reserved	data	index_size	unused
1	log_acciones	0	0 KB	0 KB	0 KB	0 KB

	name	rows	reserved	data	index_size	unused
1	bitacora_precio_producto	0	0 KB	0 KB	0 KB	0 KB

	name	rows	reserved	data	index_size	unused
1	usuarios	1	144 KB	8 KB	24 KB	112 KB

	name	rows	reserved	data	index_size	unused
1	cliente	4	216 KB	8 KB	40 KB	168 KB

	name	rows	reserved	data	index_size	unused
1	empleado	3	72 KB	8 KB	8 KB	56 KB

	name	rows	reserved	data	index_size	unused
1	marca	24	144 KB	8 KB	24 KB	112 KB

Registros creados por semana en la tabla venta

```
-- Registros creados por semana en la tabla venta
SELECT
    DATEPART(YEAR, fecha) AS Año,
    DATEPART(WEEK, fecha) AS Semana,
    COUNT(*) AS Cantidad_ventas
FROM venta
GROUP BY DATEPART(YEAR, fecha), DATEPART(WEEK, fecha)
ORDER BY Año DESC. Semana DESC:
```

123 %

Results | Messages

	Año	Semana	Cantidad_ventas
1	2025	31	3
2	2023	26	5

ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

Consultas lentas

```sql
-- Consultas más lentas en el plan de ejecución

SELECT TOP 10
    qs.total_elapsed_time / qs.execution_count AS avg_elapsed_time_ms,
    qs.execution_count,
    qs.total_elapsed_time,
    SUBSTRING(st.text, (qs.statement_start_offset/2) + 1,
        ((CASE qs.statement_end_offset
            WHEN -1 THEN DATALENGTH(st.text)
            ELSE qs.statement_end_offset END
            - qs.statement_start_offset)/2) + 1) AS query_text,
    qp.query_plan
FROM sys.dm_exec_query_stats qs
CROSS APPLY sys.dm_exec_sql_text(qs.sql_handle) st
CROSS APPLY sys.dm_exec_query_plan(qs.plan_handle) qp
ORDER BY avg_elapsed_time_ms DESC;
```

23 %

Results | Messages

	avg_elapsed_time_ms	execution_count	total_elapsed_time	query_text	query_plan
1	1103489	1	1103489	SELECT CASE WHEN name like "\\radiopox.dll\"...	<ShowPlanXML xmlns="http://schemas.microsoft.com...
2	806415	5	4032079	SELECT db.name AS [Name], CAST(0 AS bit) AS [isFabric...	<ShowPlanXML xmlns="http://schemas.microsoft.com...
3	390525	5	1564625	insert into #tmp_sp_get_sqlagent_properties(auto_start, me...	<ShowPlanXML xmlns="http://schemas.microsoft.com...
4	183742	1	183742	SELECT db_id() AS database_id, c.system_type_id...	<ShowPlanXML xmlns="http://schemas.microsoft.com...
5	175676	1	175676	SELECT db_id() AS database_id, c.system_type_id...	<ShowPlanXML xmlns="http://schemas.microsoft.com...
6	173261	1	173261	SELECT db_id() AS database_id, c.system_type_id...	<ShowPlanXML xmlns="http://schemas.microsoft.com...
7	126366	32	4043713	SELECT db.name AS [Name], CAST(0 AS bit) AS [isFabric...	<ShowPlanXML xmlns="http://schemas.microsoft.com...
8	84004	6	504026	SELECT SCHEMA_NAME(sdf.schema_id) AS [Schema], u...	<ShowPlanXML xmlns="http://schemas.microsoft.com...
9	79987	6	479925	SELECT SCHEMA_NAME(so.schema_id) AS [Schema], so...	<ShowPlanXML xmlns="http://schemas.microsoft.com...
10	75748	1	75748	SELECT ISNULL(sum(memory_allocated_for_table_kb...	<ShowPlanXML xmlns="http://schemas.microsoft.com...

Registro de uso

```sql
-- Registro del uso de funciones, procedimientos y recursos
SELECT
    OBJECT_NAME(st.objectid) AS nombre_procedimiento,
    COUNT(*) AS ejecuciones
FROM sys.dm_exec_cached_plans cp
CROSS APPLY sys.dm_exec_query_stats qs -- sin parámetro porque es una vista
CROSS APPLY sys.dm_exec_sql_text(cp.plan_handle) AS st -- función con parámetro
WHERE cp.cacheobjtype = 'Compiled Plan'
AND st.dbid = DB_ID('Supermercado')
GROUP BY st.objectid
ORDER BY ejecuciones DESC;
```

123 %

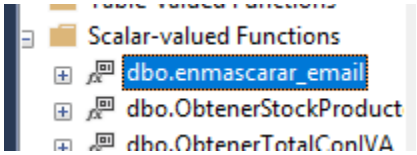
Results | Messages

	nombre_procedimiento	ejecuciones
1	NULL	60705
2	ObtenerTotalConIVA	639
3	tr_auditoria_productos	639
4	tr_notificar_cambios_precios	639
5	tr_historico_marcas	639
6	enmascarar_email	639

26. PROTECCIÓN DE DATOS

- Función para el enmascaramiento del email.

```
-- Protección de Datos y Gestión Crítica
-- Simulación de anonimización y enmascaramiento.
CREATE FUNCTION enmascarar_email (@email NVARCHAR(100))
RETURNS NVARCHAR(100)
AS
BEGIN
    DECLARE @arroba INT = CHARINDEX('@', @email)
    IF @arroba <= 3
        RETURN 'Email inválido'

    RETURN LEFT(@email, 3) + '***' + SUBSTRING(@email, @arroba, LEN(@email))
END;
GO
```




```
Messages
Commands completed successfully.

Completion time: 2025-08-03T12:26:03.1349142-05:00
```

Prueba**:**

```
-- Prueba del Enmascaramiento
SELECT nombre, dbo.enmascarar_email(email) AS email_mask
FROM cliente;
GO

CREATE FUNCTION enmascarar_cedula (
```

Results | Messages

	nombre	email_mask
1	Juan Pérez	jua***@gmail.com
2	María Gómez	mar***@outlook.com
3	Carlos Andrade	car***@gmail.com
4	Ana Beltrán	ana***@yahoo.com
5	Luis Vásquez	lui***@hotmail.com

- Función para el enmascaramiento de la cédula.

```
-- Funcion para enmascarar la cedula
CREATE FUNCTION enmascarar_cedula (
    @cedula NVARCHAR(10)
)
RETURNS NVARCHAR(10)
AS
BEGIN
    IF LEN(@cedula) != 10
        RETURN 'Cédula inválida'

    RETURN LEFT(@cedula, 2) + '*****' + RIGHT(@cedula, 3)
END
GO
```

81 %

Messages

```
Commands completed successfully.

Completion time: 2025-08-03T12:33:02.6782990-05:00
```

Prueba:

```
-- Prueba del Enmascaramiento de la cedula
SELECT nombre, dbo.enmascarar_cedula(cedula) AS cedula_mask
FROM cliente;
GO
```

81 %

Results | Messages

	nombre	cedula_mask
1	Juan Pérez	17*****567
2	María Gómez	17*****678
3	Carlos Andrade	17*****789
4	Ana Beltrán	17*****890
5	Luis Vásquez	17*****901

27. SIMULACIÓN DE PERFILES PROFESIONALES

Administrador DB

```
-- SIMULACION DE PERFILES PROFESIONALES
-- Administrador de bd
--Mantenimiento de índices
-- 1. CREACIÓN DE ÍNDICES PARA MEJORAR CONSULTAS
CREATE NONCLUSTERED INDEX idx_producto_nombre ON producto(nombre);
CREATE NONCLUSTERED INDEX idx_cliente_cedula ON cliente(cedula);
CREATE NONCLUSTERED INDEX idx_venta_fecha ON venta(fecha);
GO

-- Reorganizar (menos costoso)
EXEC sp_MSforeachtable 'ALTER INDEX ALL ON ? REORGANIZE';
```

```
123 %
Messages
Commands completed successfully.

Completion time: 2025-08-03T03:37:14.9835384-05:00
```

- **Monitoreo del sistema**

```
-- Monitoreo del sistema
EXEC sp_MSforeachtable 'EXEC sp_spaceused [?]';
GO

-- Últimas conexiones
SELECT
    login_name,
    host_name,
    program_name,
    connect_time
FROM sys.dm_exec_sessions
WHERE is_user_process = 1;
GO

-- Consultas más costosas (top 5 por uso de CPU)
SELECT TOP 5
    total_worker_time/1000.0 AS CPU_ms,
    execution_count,
    total_elapsed_time/1000.0 AS ElapsedTime_ms,
    query_hash,
```

- Últimas conexiones.

```
GO

-- Últimas conexiones
SELECT
    login_name,
    host_name,
    program_name,
    login_time
FROM sys.dm_exec_sessions
WHERE is_user_process = 1;
GO
```

123 %

Results | Messages

	login_name	host_name	program_name	login_time
1	NT SERVICE\SQLTELEMETRY$SQLSERVER	THONYDEV	SQLServerCEIP	2025-08-03 03:39:34.497
2	ThonyDev\aless	THONYDEV	Microsoft SQL Server Management Studio	2025-08-02 21:02:52.097
3	ThonyDev\aless	THONYDEV	Microsoft SQL Server Management Studio - Query	2025-08-03 01:11:03.807
4	ThonyDev\aless	THONYDEV	Microsoft SQL Server Management Studio - Query	2025-08-03 02:18:47.130
5	ThonyDev\aless	THONYDEV	Microsoft SQL Server Management Studio - Query	2025-08-03 01:13:11.880
6	ThonyDev\aless	THONYDEV	Microsoft SQL Server Management Studio - Query	2025-08-03 03:22:06.960
7	ThonyDev\aless	THONYDEV	Microsoft SQL Server Management Studio - Query	2025-08-03 01:29:34.560
8	ThonyDev\aless	THONYDEV	Microsoft SQL Server Management Studio - Query	2025-08-03 01:55:39.347
9	ThonyDev\aless	THONYDEV	Microsoft SQL Server Management Studio - Query	2025-08-03 02:07:46.263
10	ThonyDev\aless	THONYDEV	Microsoft SQL Server Management Studio - Query	2025-08-03 02:09:42.227
11	ThonyDev\aless	THONYDEV	Microsoft SQL Server Management Studio - Transa...	2025-08-03 03:44:07.317

- Consultas más costosas.

```
-- Consultas más costosas (top 5 por uso de CPU)
SELECT TOP 5
    total_worker_time/1000.0 AS CPU_ms,
    execution_count,
    total_elapsed_time/1000.0 AS ElapsedTime_ms,
    query_hash,
    SUBSTRING(qt.text, (qs.statement_start_offset/2)+1,
        ((CASE qs.statement_end_offset
          WHEN -1 THEN DATALENGTH(qt.text)
          ELSE qs.statement_end_offset END
          - qs.statement_start_offset)/2)+1) AS query_text
FROM sys.dm_exec_query_stats qs
CROSS APPLY sys.dm_exec_sql_text(qs.sql_handle) qt
ORDER BY total_worker_time DESC;
```

Results | Messages

CPU_ms	execution_count	ElapsedTime_ms	query_hash	query_text
75.746000	6	84.319000	0xD774502DEB22116D	SELECT i.name AS [Name], CAST(i.index_id AS int) AS [ID]...
46.210000	1	105.776000	0x52B3B6CEDE6FC57A	SELECT target_data FROM sys.dm_xe_session_targ...
0.237000	1	0.878000	0x1B1B559B08C771CC	IF EXISTS (SELECT * FROM sys.databases WHERE DB_...
0.186000	1	0.186000	0x1B1B559B08C771CC	IF EXISTS (SELECT * FROM sys.databases WHERE DB_...
0.175000	1	0.175000	0x954C4B6BCBEAADC5	IF EXISTS (SELECT * FROM sys.change_tracking_databa...

ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

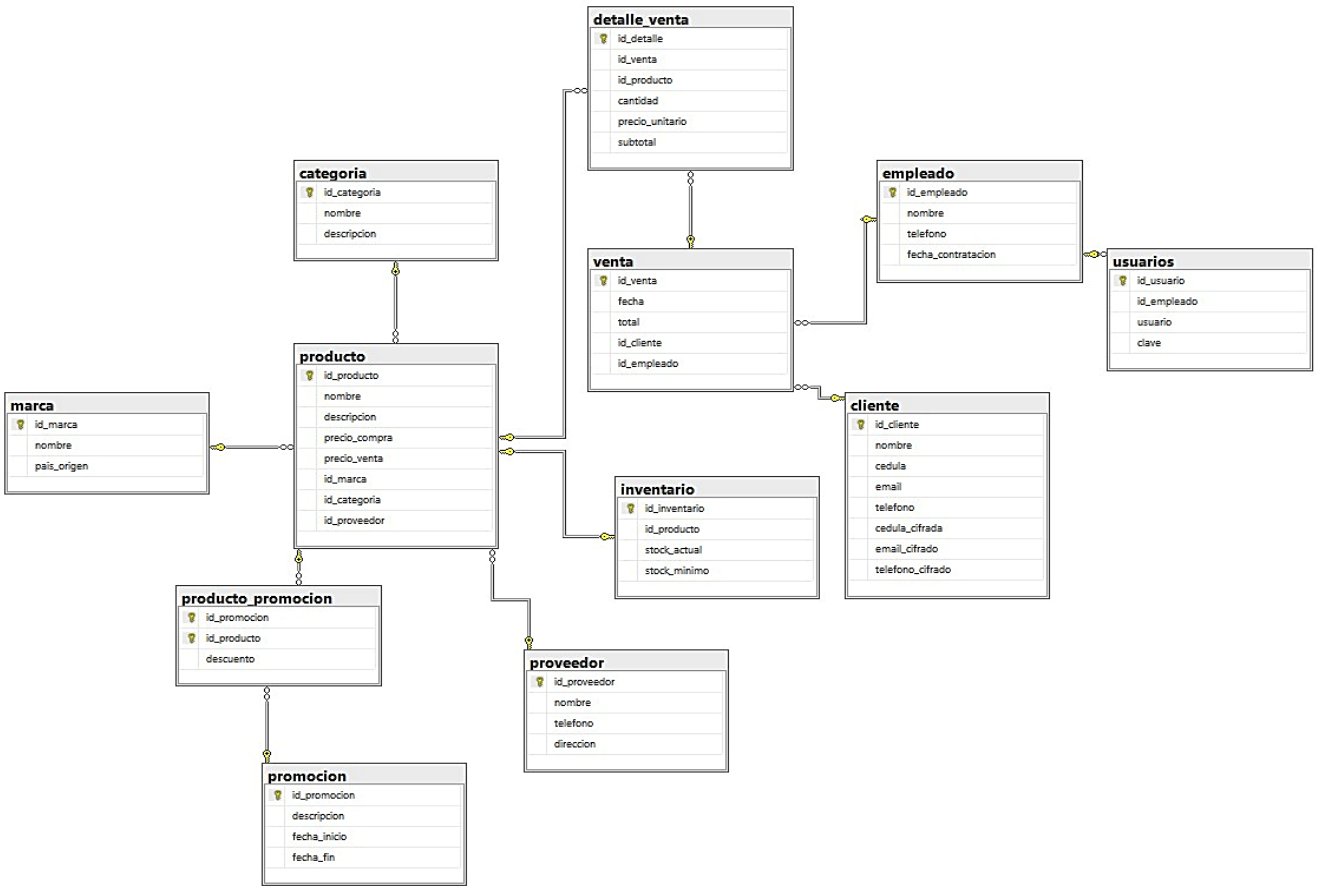
Arquitectura de DB

```sql
-- ARQUITECTO DE BD
-- Vista del diseño lógico
SELECT
    t.name AS tabla,
    c.name AS columna,
    ty.name AS tipo_dato,
    c.max_length AS longitud,
    c.is_nullable AS permite_nulo,
    i.is_primary_key AS clave_primaria
FROM sys.tables t
JOIN sys.columns c ON t.object_id = c.object_id
JOIN sys.types ty ON c.user_type_id = ty.user_type_id
LEFT JOIN sys.index_columns ic ON t.object_id = ic.object_id AND c.column_id = ic.column_id
LEFT JOIN sys.indexes i ON ic.object_id = i.object_id AND ic.index_id = i.index_id;
```

Results | Messages

	tabla	columna	tipo_dato	longitud	permite_nulo	clave_primaria
1	notificaciones_sensibles	id_notificacion	int	4	0	1
2	notificaciones_sensibles	tipo_accion	varchar	50	0	NULL
3	notificaciones_sensibles	tabla_afectada	varchar	50	0	NULL
4	notificaciones_sensibles	id_registro	int	4	1	NULL
5	notificaciones_sensibles	descripcion	nvarchar	1000	0	NULL
6	notificaciones_sensibles	fecha	datetime	8	0	NULL

Visualización



```sql
-- Detectar tablas que no sigan el prefijo estándar
SELECT name AS tabla_sin_prefijo
FROM sys.tables
WHERE name NOT LIKE 'tbl_%';
```

123 %

Results | Messages

	tabla_sin_prefijo
1	notificaciones_sensibles
2	historico_marcas
3	registro_intentos_fallidos
4	log_acciones
5	bitacora_precio_producto
6	usuarios
7	sysdiagrams
8	cliente
9	empleado

- Validación de las PK

```sql
-- Validar si las PK cumplen el estándar de nombre 'id_<tabla>'
SELECT
    t.name AS tabla,
    c.name AS columna_pk,
    CASE
        WHEN c.name = 'id_' + t.name THEN 'Cumple'
        ELSE 'No cumple'
    END AS cumple_estandar
FROM sys.tables t
JOIN sys.indexes i ON t.object_id = i.object_id AND i.is_primary_key = 1
JOIN sys.index_columns ic ON i.object_id = ic.object_id AND i.index_id = ic.index_id
JOIN sys.columns c ON t.object_id = c.object_id AND ic.column_id = c.column_id;
```

%

Results | Messages

tabla	columna_pk	cumple_estandar
notificaciones_sensibles	id_notificacion	No cumple
historico_marcas	id_historico	No cumple
registro_intentos_fallidos	id	No cumple
log_acciones	id_log	No cumple
bitacora_precio_producto	id_bitacora	No cumple

- Proyección de escalabilidad.

```
-- Proyeccion de escalabilidad
SELECT
    t.name AS tabla,
    SUM(p.rows) AS total_filas
FROM sys.tables t
JOIN sys.partitions p ON t.object_id = p.object_id
WHERE p.index_id IN (0,1)
GROUP BY t.name
ORDER BY total_filas DESC;
GO
```

Results | Messages

	tabla	total_filas
	producto	535
2	inventario	34
3	marca	24
4	categoria	15
5	detalle_venta	15
6	producto_promocion	12
7	proveedor	10

```
WHERE p.index_id IN (0,1)
GROUP BY t.name
ORDER BY total_filas DESC;
GO

EXEC sp_MSforeachtable 'EXEC sp_spaceused ''?''';
```

Results | Messages

	name	rows	reserved	data	index_size	unused
1	notificaciones_sensibles	1	72 KB	8 KB	8 KB	56 KB

	name	rows	reserved	data	index_size	unused
1	historico_marcas	1	72 KB	8 KB	8 KB	56 KB

	name	rows	reserved	data	index_size	unused
1	registro_intentos_fallidos	1	72 KB	8 KB	8 KB	56 KB

	name	rows	reserved	data	index_size	unused
1	log_acciones	0	0 KB	0 KB	0 KB	0 KB

	name	rows	reserved	data	index_size	unused
1	bitacora_precio_producto	0	0 KB	0 KB	0 KB	0 KB

	name	rows	reserved	data	index_size	unused
1	usuarios	1	144 KB	8 KB	24 KB	112 KB

	name	rows	reserved	data	index_size	unused
1	sysdiagrams	1	280 KB	104 KB	24 KB	152 KB

	name	rows	reserved	data	index_size	unused
1	cliente	4	288 KB	8 KB	56 KB	224 KB

	name	rows	reserved	data	index_size	unused
1	empleado	3	72 KB	8 KB	8 KB	56 KB

ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

Oficial de seguridad

Revisión de integridad

```sql
-- Revisión de integridad
SELECT
    f.name AS restriccion,
    OBJECT_NAME(f.parent_object_id) AS tabla_hija,
    COL_NAME(fc.parent_object_id, fc.parent_column_id) AS columna_hija,
    OBJECT_NAME(f.referenced_object_id) AS tabla_padre,
    COL_NAME(fc.referenced_object_id, fc.referenced_column_id) AS columna_padre
FROM sys.foreign_keys AS f
INNER JOIN sys.foreign_key_columns AS fc ON f.object_id = fc.constraint_object_id;
```

23 %

Results | Messages

	restriccion	tabla_hija	columna_hija	tabla_padre	columna_padre
1	FK__venta__id_client__571DF1D5	venta	id_cliente	cliente	id_cliente
2	FK__usuarios__id_emp__29221CFB	usuarios	id_empleado	empleado	id_empleado
3	FK__venta__id_emplea__5812160E	venta	id_empleado	empleado	id_empleado
4	FK__producto__id_mar__48CFD27E	producto	id_marca	marca	id_marca
5	FK__producto__id_cat__49C3F6B7	producto	id_categoria	categoria	id_categoria
6	FK__producto__id_pro__4AB81AF0	producto	id_proveedor	proveedor	id_proveedor
7	FK__inventari__id_pr__52593CB8	inventario	id_producto	producto	id_producto
8	FK__detalle_v__id_pr__5EBF139D	detalle_venta	id_producto	producto	id_producto
9	FK__producto___id_pr__66603565	producto_promocion	id_producto	producto	id_producto
10	FK__detalle_v__id_ve__5DCAEF64	detalle_venta	id_venta	venta	id_venta
11	FK				

```sql
-- Insertar producto para prueba
INSERT INTO producto (nombre, descripcion, precio_compra, precio_venta, id_categ
VALUES ('Producto Test', 'Descripcion', 10, 15, 1, 1);

-- Actualizar producto
UPDATE producto SET precio_venta = 20 WHERE nombre = 'Producto Test';

-- Eliminar producto
DELETE FROM producto WHERE nombre = 'Producto Test';

-- Consultar logs
SELECT
    l.id_log,
    l.id_producto,
    p.nombre AS nombre_producto,
    l.accion,
```

Results | Messages

id_log	id_producto	nombre_producto	accion	usuario	fecha
3	536	NULL	DELETE	NULL	2025-08-03 10:22:13.660
2	536	NULL	UPDATE	NULL	2025-08-03 10:22:13.650
1	536	NULL	INSERT	NULL	2025-08-03 10:22:13.647

Revisión de los logs

```
-- Insertar producto para prueba
INSERT INTO producto (nombre, descripcion, precio_compra, precio_venta, id_categ
VALUES ('Producto Test', 'Descripcion', 10, 15, 1, 1);

-- Actualizar producto
UPDATE producto SET precio_venta = 20 WHERE nombre = 'Producto Test';

-- Eliminar producto
DELETE FROM producto WHERE nombre = 'Producto Test';

-- Consultar logs
SELECT
    l.id_log,
    l.id_producto,
    p.nombre AS nombre_producto,
    l.accion,
```

Results | Messages

id_log	id_producto	nombre_producto	accion	usuario	fecha
3	536	NULL	DELETE	NULL	2025-08-03 10:22:13.660
2	536	NULL	UPDATE	NULL	2025-08-03 10:22:13.650
1	536	NULL	INSERT	NULL	2025-08-03 10:22:13.647

Pruebas de vulnerabilidad

```
-- Pruebas de vulnerabilidad
--  Usuarios con roles y permisos
SELECT dp.name AS Usuario, r.name AS Rol
 FROM sys.database_principals dp
 LEFT JOIN sys.database_role_members drm ON dp.principal_id = drm.member_principal_id
 LEFT JOIN sys.database_principals r ON drm.role_principal_id = r.principal_id
 WHERE dp.type NOT IN ('R', 'C')
 ORDER BY Usuario;
```

Results | Messages

Usuario	Rol
dbo	db_owner
guest	NULL
INFORMATION_SCHEMA	NULL
nombre_admin	rol_admin
nombre_usuario	rol_ventas
sys	NULL
usuario_admin	rol_admin

ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

Permisos peligrosos asignados

```
-- Permisos peligrosos asignados (como CONTROL o ALTER)
SELECT princ.name, perm.permission_name, OBJECT_NAME(perm.major_id) AS Objeto
FROM sys.database_permissions perm
JOIN sys.database_principals princ ON perm.grantee_principal_id = princ.principal_id
WHERE perm.permission_name IN ('CONTROL', 'ALTER')
ORDER BY princ.name;
```

123 %

Results | Messages

	name	permission_name	Objeto
1	rol_admin	CONTROL	NULL
2	rol_administrador	CONTROL	NULL

Conexiones activas para detectar accesos peligrosos

```
-- Conexiones activas para detectar accesos extraños
SELECT session_id, login_name, status, host_name, program_name
FROM sys.dm_exec_sessions
WHERE is_user_process = 1;
```

123 %

Results | Messages

	session_id	login_name	status	host_name	program_name
1	57	ThonyDev\aless	sleeping	THONYDEV	Microsoft SQL Server Management Studio - Query
2	58	ThonyDev\aless	running	THONYDEV	Microsoft SQL Server Management Studio - Query
3	63	ThonyDev\aless	sleeping	THONYDEV	Microsoft SQL Server Management Studio - Query
4	64	ThonyDev\aless	sleeping	THONYDEV	Microsoft SQL Server Management Studio - Query
5	66	ThonyDev\aless	sleeping	THONYDEV	Microsoft SQL Server Management Studio - Query
6	67	NT SERVICE\SQLTELEMETRY$SQLSERVER	sleeping	THONYDEV	SQLServerCEIP
7	68	ThonyDev\aless	sleeping	THONYDEV	Microsoft SQL Server Management Studio

Revisión de logs

```
-- Revisión rápida de logs de cambios (si tienes trigger)
SELECT TOP 10 * FROM log_cambios_producto ORDER BY fecha DESC;
```

123 %

Results | Messages

	id_log	id_producto	accion	fecha	usuario
1	3	536	DELETE	2025-08-03 10:22:13.660	NULL
2	2	536	UPDATE	2025-08-03 10:22:13.650	NULL
3	1	536	INSERT	2025-08-03 10:22:13.647	NULL

Desarrollador de consultas

```
-- Insertar datos de prueba
INSERT INTO cliente (nombre, cedula, email, telefono)
VALUES ('Juanito Perez', '1234567890', 'juan.perez@mail.com', '0999999999');
GO

-- Consultar como usuario normal
SELECT id_cliente, nombre, cedula, email FROM cliente;
GO

-- Consulta simulada como usuario sin permisos
EXECUTE AS USER = 'guest';  -- 'guest' es un usuario típico sin permisos especiales
SELECT id_cliente, nombre, cedula, email FROM cliente;
REVERT;
GO
```

23 % — Results | Messages

	id_cliente	nombre	cedula	email
1	1	Juan Pérez	1701234567	juan.perez@gmail.com
2	2	María Gómez	1702345678	maria.gomez@outlook.com
3	4	Ana Beltrán	1704567890	ana.beltran@yahoo.com
4	5	Luis Vásquez	1705678901	luis.vasquez@hotmail.com
5	6	Juanito Perez	1234567890	juan.perez@mail.com

- Consulta simulada.

```
-- Consulta simulada con el usuario de prueba
EXECUTE AS USER = 'prueba_masking';
SELECT id_cliente, nombre, cedula, email FROM cliente;
REVERT;
GO
```

123 % — Results | Messages

	id_cliente	nombre	cedula	email
1	1	Juan Pérez	XXXXXXX567	jXXX@XXXX.com
2	2	María Gómez	XXXXXXX678	mXXX@XXXX.com
3	4	Ana Beltrán	XXXXXXX890	aXXX@XXXX.com
4	5	Luis Vásquez	XXXXXXX901	lXXX@XXXX.com
5	6	Juanito Perez	XXXXXXX890	jXXX@XXXX.com

```sql
-- Probando
SELECT * FROM vw_ventas_detalle;
```

	id_venta	fecha	cliente	empleado	producto	cantidad	precio_unitario	subtotal
1	1	2023-07-01 09:00:00.000	Juan Pérez	Mónica Patricia Zurita	Leche Entera Vita	2	1.20	2.40

- Ventas de un cliente especifico

```sql
-- ventas de un cliente especifico
SELECT * FROM vw_ventas_detalle WHERE cliente = 'Juan Pérez';
-- O ventas en una fecha especifica
SELECT * FROM vw_ventas_detalle WHERE fecha >= '2025-01-01';
```

	id_venta	fecha	cliente	empleado	producto	cantidad	precio_unitario	subtotal
1	1	2023-07-01 09:00:00.000	Juan Pérez	Mónica Patricia Zurita	Leche Entera Vita	2	1.20	2.40
2	1	2023-07-01 09:00:00.000	Juan Pérez	Mónica Patricia Zurita	Agua Mineral Tesalia	1	0.50	0.50
3	1	2023-07-01 09:00:00.000	Juan Pérez	Mónica Patricia Zurita	Pechuga de Pollo Pronaca	1	4.90	4.90
4	7	2025-08-01 00:00:00.000	Juan Pérez	Ricardo Andrés Zambrano	Queso Fresco Pronaca	4	3.50	14.00
5	8	2025-08-01 00:00:00.000	Juan Pérez	Ricardo Andrés Zambrano	Queso Fresco Pronaca	2	3.50	7.00

```sql
-- O ventas en una fecha especifica
SELECT * FROM vw_ventas_detalle WHERE fecha >= '2025-01-01';
```

	id_venta	fecha	cliente	empleado	producto	cantidad	precio_unitario	subtotal
1	7	2025-08-01 00:00:00.000	Juan Pérez	Ricardo Andrés Zambrano	Queso Fresco Pronaca	4	3.50	14.00
2	8	2025-08-01 00:00:00.000	Juan Pérez	Ricardo Andrés Zambrano	Queso Fresco Pronaca	2	3.50	7.00

Reportes

```sql
-- Reporte complejo
SELECT
    e.nombre AS empleado,
    p.nombre AS producto,
    SUM(dv.cantidad) AS total_cantidad,
    SUM(dv.subtotal) AS total_ventas
FROM venta v
JOIN detalle_venta dv ON v.id_venta = dv.id_venta
JOIN empleado e ON v.id_empleado = e.id_empleado
JOIN producto p ON dv.id_producto = p.id_producto
WHERE v.fecha >= DATEADD(MONTH, -1, GETDATE())
GROUP BY e.nombre, p.nombre
ORDER BY total_ventas DESC;
```

empleado	producto	total_cantidad	total_ventas
Ricardo Andrés Zambrano	Queso Fresco Pronaca	6	21.00

ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

Procedimientos almacenados

```
-- Procedimiento almacenado reutilizable
CREATE PROCEDURE sp_ventas_por_cliente
    @id_cliente INT,
    @fecha_inicio DATE,
    @fecha_fin DATE
AS
BEGIN
    SELECT v.id_venta, v.fecha, v.total
    FROM venta v
    WHERE v.id_cliente = @id_cliente
      AND v.fecha BETWEEN @fecha_inicio AND @fecha_fin
    ORDER BY v.fecha;
END;
--Probando
EXEC sp_ventas_por_cliente @id_cliente = 1, @fecha_inicio = '2025-01-01', @fecha_fin = '2025
```

Results Messages

	id_venta	fecha	total
1	6	2025-07-31 00:00:00.000	30.50
2	8	2025-08-01 00:00:00.000	7.00
3	7	2025-08-01 00:00:00.000	14.00

Pruebas

```
-- Probando
-- Asegúrate de tener un producto y una venta
SELECT TOP 5 id_venta, id_producto FROM detalle_venta WHERE id_producto = 1;
SELECT dbo.fn_descuento_producto(1, 1) AS descuento_aplicado;
```

Results Messages

	descuento_aplicado
1	15.00

```
-- Otra prueba
EXECUTE AS USER = 'usuario_cliente';
SELECT id_venta, cliente, producto, cantidad, subtotal
FROM vw_ventas_detalle
WHERE fecha > '2025-01-01';
REVERT;
```

Results Messages

	id_venta	cliente	producto	cantidad	subtotal
1	7	Juan Pérez	Queso Fresco Pronaca	4	14.00
2	8	Juan Pérez	Queso Fresco Pronaca	2	7.00

LINK DEL VIDEO:

https://youtu.be/UeBhBLj7d1Q

28. CONCLUSIONES

La implementación de medidas de seguridad en una base de datos es uno de los requisitos obligatorios si deseamos gestionar la información de forma correcta. En este caso, n Supermercado suele ser una entidad que maneja un gran numero de productos, clientes y empleados. Así que, con mas razón se deben realizar las diferentes tareas de manera eficiente.

La implementación de mecanismos de protección y validación de datos es sumamente importante, si no se realiza adecuadamente, los datos personales de los usuarios serian robados y usados para fines desconocidos. Por tal razón, es necesario que se construya la base de datos con todos los parámetros y medidas para mantener la BD escalable, confiable, disponible y segura.

Aun cuando su desarrollo y configuración es complicado y muchas veces se deben seguir estándares rigurosos. Por ejemplo, registrar todas las acciones importantes que sucedan dentro del sistema. Las grandes y pequeñas empresas se esfuerzan por cumplir al pie de la letra los reglamentos que involucran este tema. Esto por dos razones, evitar demanda por parte de los usuarios y multas por parte de las autoridades pertinentes.

29. RECOMENDACIONES

- Verificar la configuración de SQL Server porque existen funciones que no funcionan correctamente sin una configuración previa, por ejemplo, la creación de backups.
- Tener un alto nivel de conocimiento acerca de las funciones disponibles en SQL Server, esto para ser más eficiente durante el desarrollo.
- Organizar los diferentes procedimientos, funciones, vistas, índices, etc., de tal manera que sea fácil de encontrar lo que se necesite inmediatamente.