“Año De La Recuperación Y
Consolidación De La Economía Peruana”

UNIVERSIDAD PERUANA LOS ANDES

“FACULTAD DE INGENIERÍA”

ESCUELA PROFESIONAL “SISTEMAS Y COMPUTACIÓN”

CÁTEDRA: Base de Datos II

CATEDRÁTICO: Ing. Fernandez Bejarano Raul Enrique

ESTUDIANTE: Vega Brañez Samuel Max

CICLO: V

SECCIÓN: B1

HUANCAYO PERÚ

2025

Manual de Seguridad y Control de Acceso

Base de Datos: TechGadget Store

El siguiente código SQL crea la base de datos `TechGadgetStore`, la tabla `Empleados`, y la rellena con 10 registros ficticios. La columna clave de seguridad es **`PIN_Acceso`**, que usa `HASHBYTES` para el **Cifrado y Protección de Datos**.

SQL

```
-- 1. CREACIÓN DE LA BASE DE DATOS Y TABLA
CREATE DATABASE TechGadgetStore;
GO
USE TechGadgetStore;
GO

CREATE TABLE Empleados ( ID_Empleado
    INT PRIMARY KEY, Nombre
    VARCHAR(100) NOT NULL, Cargo
    VARCHAR(50) NOT NULL, Salario
    DECIMAL(10, 2) NOT NULL,
    PIN_Acceso VARBINARY(256), -- Almacena el hash SHA2_256 del PIN
    FechaContratacion DATE NOT NULL,
    Email VARCHAR(100) UNIQUE,
    ID_Supervisor INT,
    EstadoCuenta VARCHAR(10) DEFAULT 'Activo',
    UltimoLogin DATETIME -- Para monitoreo y auditoría
);
GO

-- 2. INSERCIÓN DE 10 DATOS FICTICIOS
INSERT INTO Empleados (ID_Empleado, Nombre, Cargo, Salario,
PIN_Acceso, FechaContratacion, Email, ID_Supervisor, EstadoCuenta,
UltimoLogin) VALUES
(100, 'Ana Gómez', 'Gerente', 6500.00, HASHBYTES('SHA2_256',
'Gerente2025'), '2020-01-15', 'ana.g@tech.com', NULL, 'Activo',
GETDATE()),
(101, 'Beto Ruiz', 'Vendedor', 3200.00, HASHBYTES('SHA2_256',
'VentasBeto'), '2021-05-20', 'beto.r@tech.com', 100, 'Activo',
GETDATE()),
(102, 'Carla Diaz', 'Vendedor', 3150.00, HASHBYTES('SHA2_256',
'VentasCarla'), '2022-08-10', 'carla.d@tech.com', 100, 'Activo',
GETDATE()),
(103, 'David Soto', 'Almacén', 2800.00, HASHBYTES('SHA2_256',
'AlmacenD1'), '2023-03-01', 'david.s@tech.com', 105, 'Activo',
GETDATE()),
(104, 'Elena Paz', 'Soporte', 3500.00, HASHBYTES('SHA2_256',
'SoporteE5'), '2021-11-25', 'elena.p@tech.com', 100, 'Activo',
GETDATE()),
(105, 'Felipe Rey', 'Subgerente', 4800.00, HASHBYTES('SHA2_256',
'Subgerente'), '2019-10-01', 'felipe.r@tech.com', 100, 'Activo',
GETDATE()),
(106, 'Gaby Mora', 'Vendedor', 3000.00, HASHBYTES('SHA2_256',
'VentasGaby'), '2023-12-05', 'gaby.m@tech.com', 101, 'Activo',
GETDATE()),
```

```
(107, 'Hugo León', 'Almacén', 2750.00, HASHBYTES('SHA2_256',
'AlmacenH2'), '2024-01-18', 'hugo.l@tech.com', 105, 'Activo',
GETDATE()),
(108, 'Inés Cruz', 'Vendedor', 3100.00, HASHBYTES('SHA2_256',
'VentasInes'), '2022-04-01', 'ines.c@tech.com', 102, 'Activo',
GETDATE()),
(109, 'Juan Vidal', 'Soporte', 3450.00, HASHBYTES('SHA2_256',
'SoporteJ6'), '2023-07-07', 'juan.v@tech.com', 104, 'Inactivo',
GETDATE());
```

Aplicación de Seguridad y Control de Acceso (Semana 11)

¡Excelente! Vamos a desglosar el punto 1, **Autenticación SQL y Windows**, con más datos y el paso a paso de cómo se configura en un entorno real, aplicando las prácticas seguras a la base de datos `TechGadgetStore`.

1️⃣ Autenticación SQL y Windows: Datos y Pasos

Este punto es fundamental, ya que define cómo un usuario o una aplicación demuestra su identidad para acceder al servidor de la base de datos. Existen dos modos principales de autenticación en SQL Server:

Modo de Autenticación	Descripción	Uso Recomendado
Autenticación de Windows	Usa las credenciales del sistema operativo (dominio o máquina local). El usuario no necesita dar una contraseña a SQL Server; el SO la valida primero.	**Administradores, Gerentes** y la mayoría de los usuarios empresariales. Es el modo **más seguro** y recomendado.
Autenticación de SQL Server	SQL Server almacena las credenciales (login y contraseña) internamente, independientemente del sistema operativo.	**Aplicaciones** (ej. Punto de Venta), usuarios de bajo privilegio, o entornos no basados en Windows.

Configuración del Servidor (Paso a Paso)

Antes de crear usuarios, el servidor debe estar configurado para aceptar ambos tipos de autenticación.

Paso 1: Habilitar el Modo de Autenticación Mixta

Por defecto, la instalación de SQL Server puede estar en modo "Autenticación de Windows". Para usar ambas (SQL y Windows), debe configurarse en **Modo Mixto**.

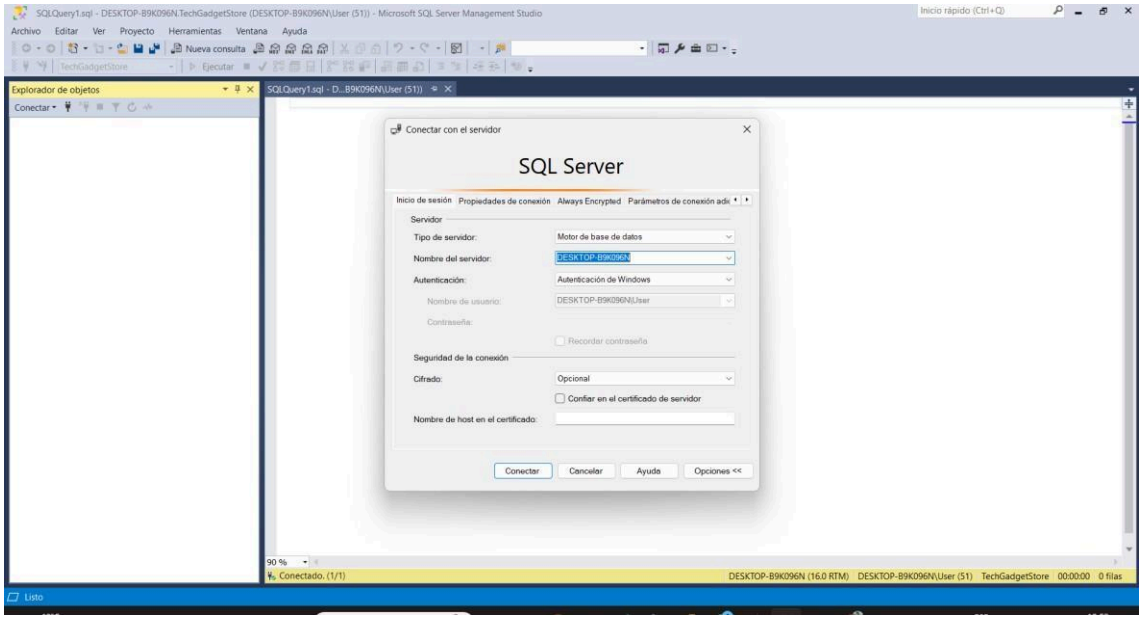
1. Abre el **SQL Server Management Studio (SSMS)** y conéctate al servidor.
2. Haz clic derecho en el nombre del servidor en el Explorador de objetos y selecciona **Propiedades**.
3. Ve a la página **Seguridad**.
4. En "Autenticación del servidor", selecciona **Modo de Autenticación de SQL Server y Windows**.
5. Haz clic en Aceptar y **reinicia el servicio de SQL Server** para que el cambio surta efecto.

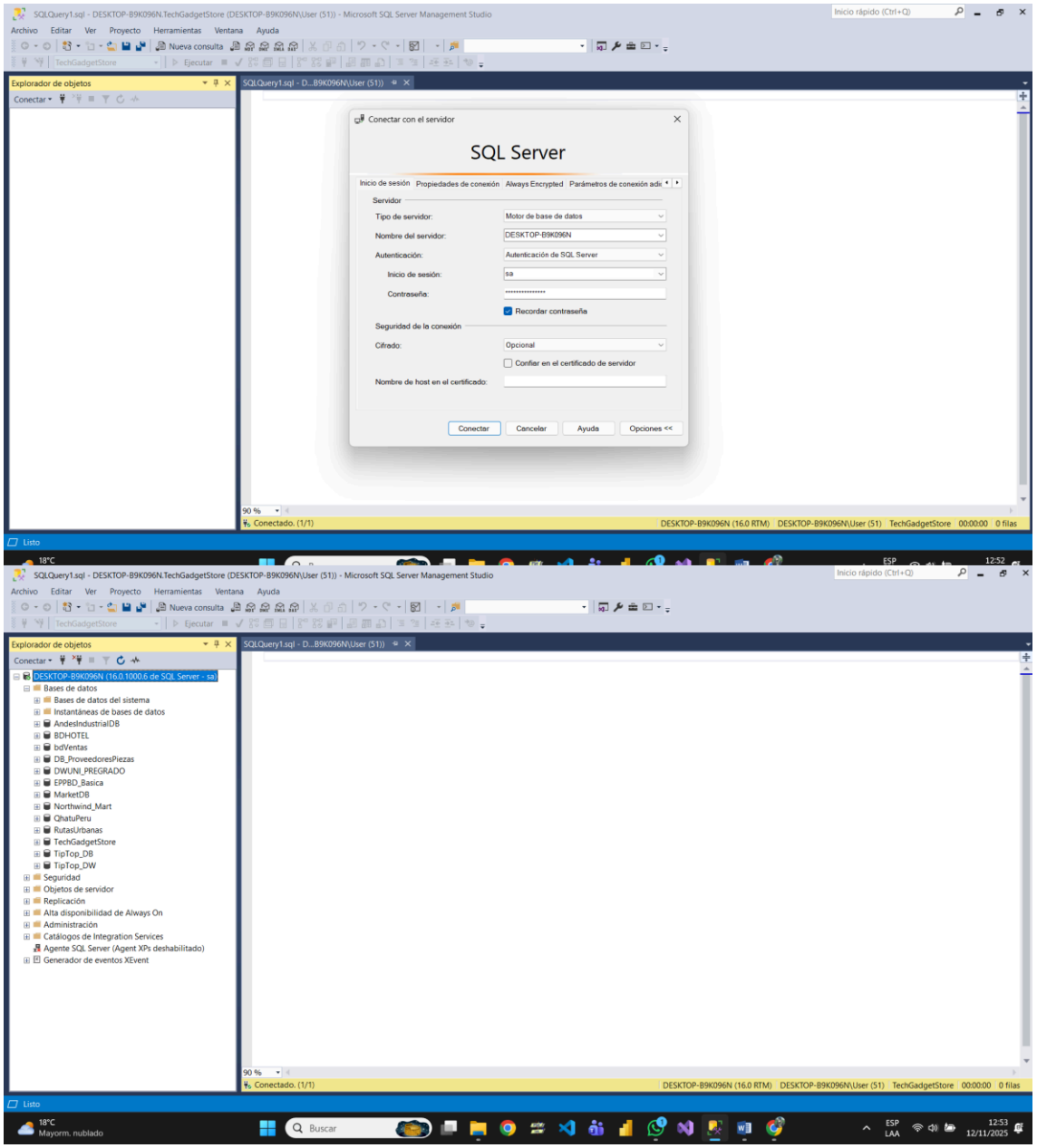
Paso 2: Práctica Segura para la Contraseña `sa`

Si habilitas el Modo Mixto, la cuenta de administrador de SQL Server (`sa`) se vuelve activa. **Práctica Segura:** Asigna una **contraseña fuerte** y compleja a la cuenta `sa` y, si es posible, **deshabilítala** después de crear otras cuentas administrativas de Windows.

SQL

```
-- Cambiar la contraseña de la cuenta 'sa'
ALTER LOGIN sa WITH PASSWORD = 'TuContraseñaAdmin!@#123',
CHECK_POLICY = ON;
GO

-- Deshabilitar la cuenta 'sa' (si tienes otros administradores de
Windows)
ALTER LOGIN sa DISABLE;
GO
```





Creación de Logins y Usuarios (Paso a Paso)

Ahora creamos los accesos para los empleados de `TechGadgetStore`.

1. Autenticación de Windows (Para Gerentes y Administradores)

- **Enunciado:** La Gerente, Ana Gómez, usará su cuenta de Windows para acceder a la BD.
- **Práctica Segura:** Uso de credenciales de dominio.

Objeto	Nombre en el Servidor	Usuario de la BD
Login (Servidor)	`DOMINIO\AnaGomez`	`AnaUser`

SQL

```
USE master;
-- 1. Crear el Login de Windows a nivel de Servidor (suponiendo un
dominio llamado 'TECHGADGET_DOMAIN')
CREATE LOGIN [TECHGADGET_DOMAIN\AnaGomez] FROM WINDOWS;
GO

USE TechGadgetStore;
-- 2. Crear el Usuario de BD (AnaUser) a partir del Login (vincular
el Login a la BD)
CREATE USER AnaUser FOR LOGIN [TECHGADGET_DOMAIN\AnaGomez];
GO

-- 3. Asignarle el rol de propietario de datos (una buena práctica
para un Gerente)
ALTER ROLE db_owner ADD MEMBER AnaUser;
GO
```

2. Autenticación de SQL Server (Para Vendedores y Aplicaciones)

- **Enunciado:** El Vendedor, Beto Ruiz, usará un login exclusivo de SQL Server, ideal para sistemas de Punto de Venta (POS).
- **Práctica Segura:** Usar políticas de contraseña obligatorias (`CHECK_POLICY = ON`).

Objeto	Nombre en el Servidor	Usuario de la BD
Login (Servidor)	`BetoLogin`	`BetoUser`

SQL

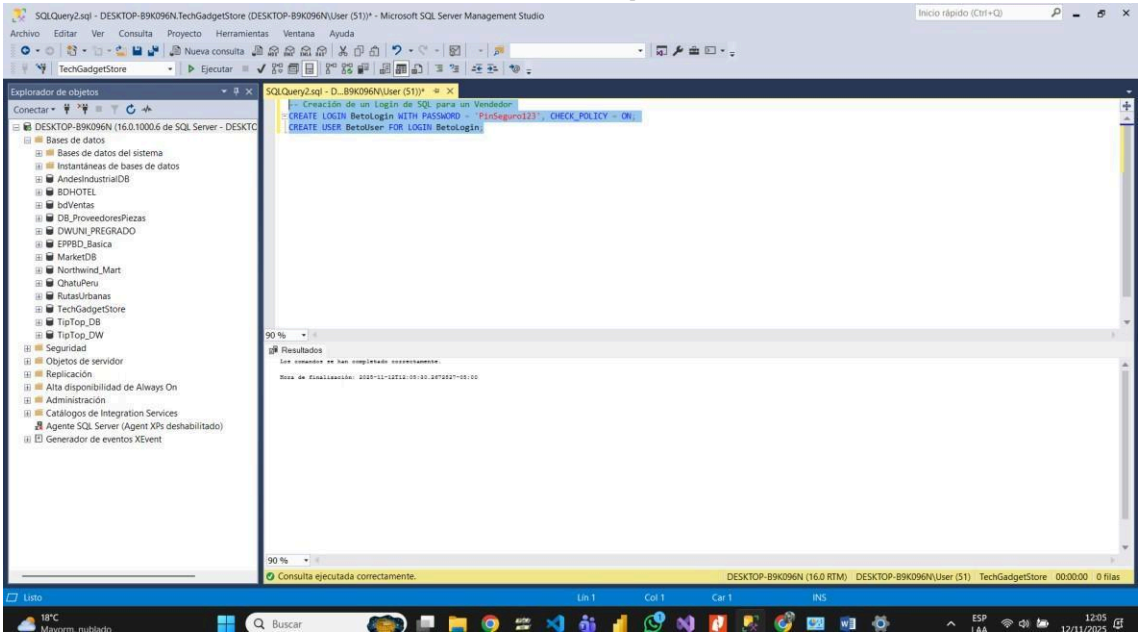
```
USE master;
-- 1. Crear el Login de SQL Server
CREATE LOGIN BetoLogin
WITH PASSWORD = 'ContraseñaDeBeto#1',
CHECK_POLICY = ON; -- Fuerza el cumplimiento de políticas de
longitud, complejidad y expiración
GO

USE TechGadgetStore;
-- 2. Crear el Usuario de BD (BetoUser) a partir del Login
(vincularlo a la BD)
CREATE USER BetoUser FOR LOGIN BetoLogin;
GO

-- 3. Asignarle el rol de Vendedor creado en el punto 3 (Seguridad)
ALTER ROLE Rol_Ventas_Tech ADD MEMBER BetoUser;
GO
```

Diferencias Clave y Prácticas Seguras

Característica	Autenticación de Windows	Autenticación de SQL Server
Validación	Por el Sistema Operativo (Active Directory/Dominio).	Por SQL Server (credenciales internas).
Contraseña	Cambia con el SO/Dominio.	Solo se cambia en SQL Server.
Seguridad	**Alta.** Usa Kerberos, evita el almacenamiento de contraseñas en SQL.	**Media.** Requiere políticas de contraseña muy estrictas.
Práctica Segura	Es la autenticación preferida en entornos corporativos.	Útil solo cuando no hay alternativa (ej. aplicaciones externas, entornos Linux).

SQL

```
-- Creación de un Login de SQL para un Vendedor
CREATE LOGIN BetoLogin WITH PASSWORD = 'PinSeguro123', CHECK_POLICY
= ON;
CREATE USER BetoUser FOR LOGIN BetoLogin;
```



2 Cuentas de Servicio y Configuración del Servidor

Enunciado del Caso Práctico

Los servicios de SQL Server (`MSSQLSERVER` y `SQLSERVERAGENT`) deben ejecutarse con **Cuentas de Servicio Administradas (MSA)** y deben implementarse configuraciones de seguridad a nivel de servidor para **reducir la superficie de ataque** de la instancia de `TechGadgetStore`.

Explicación: Cuentas de Servicio (Service Accounts)

Las **Cuentas de Servicio** son las identidades de Windows bajo las cuales se ejecutan los procesos clave de SQL Server (como el motor de base de datos y el SQL Server Agent).
Riesgo Principal

Si el servicio de SQL Server se ejecuta bajo una cuenta con privilegios excesivos (como `Local System` o una cuenta de Administrador de Dominio), un atacante que logre explotar una vulnerabilidad en SQL Server obtendría inmediatamente los privilegios completos de esa cuenta en el sistema operativo.
Práctica Segura: Principio del Mínimo Privilegio

Se debe aplicar el **Principio del Mínimo Privilegio (PoLP)**:

1. **Cuentas Separadas:** Asignar cuentas de servicio diferentes a cada servicio (ej., una cuenta para el motor, otra para el agente).
2. **Cuentas MSA:** Utilizar **Cuentas de Servicio Administradas (MSA)** o **gMSA (Group Managed Service Accounts)** de Windows. Estas cuentas son manejadas automáticamente por Windows/Active Directory, no tienen contraseñas que expiran o que deban ser gestionadas manualmente, y se les otorgan solo los permisos mínimos necesarios para que SQL Server funcione correctamente (lectura de disco, gestión de red, etc.).

Explicación: Configuración de Seguridad del Servidor (Hardening)

La **configuración de seguridad del servidor** (o *hardening*) implica desactivar funciones innecesarias y limitar los recursos para evitar abusos o fugas de información.

Área de Configuración	**Por qué es Crítico**
Conexiones Remotas	Si la base de datos `TechGadgetStore` solo debe ser accedida por aplicaciones locales, se deben desactivar los protocolos de red no utilizados (ej. *Named Pipes*) y forzar el uso de **TLS/SSL** (Transport Layer Security) para todas las conexiones.
Funcionalidades Peligrosas	Algunas funciones, aunque útiles, son vector de ataque si se dejan activadas sin control, como la ejecución de código externo.
Memoria	Limitar la cantidad máxima de memoria que SQL Server puede usar para garantizar que el sistema operativo tenga recursos

Área de Configuración	Por qué es Crítico
	suficientes para funcionar y evitar fallos del sistema por agotamiento de memoria.

Pasos Prácticos de Configuración de Seguridad

Estas son acciones clave para asegurar la instancia de SQL Server de `TechGadgetStore`.

1. Configuración de Cuentas de Servicio (A Nivel de Windows)

Esta configuración se hace usando el **SQL Server Configuration Manager** después de la instalación, no con comandos SQL.

1. **Detener los Servicios:** Detener los servicios de `SQL Server (MSSQLSERVER)` y `SQL Server Agent`.
2. **Cambiar las Cuentas:** Modificar las propiedades de cada servicio para reemplazar `NT Service\MSSQLSERVER` (o una cuenta de alto privilegio) por las **Cuentas MSA de bajo privilegio** creadas por el administrador de dominio.
3. **Reiniciar:** Reiniciar ambos servicios.

2. Desactivar Funcionalidades de Riesgo (A Nivel de SQL)

Funcionalidades como `xp_cmdshell` (que permite ejecutar comandos del sistema operativo desde SQL) deben ser deshabilitadas si no son estrictamente necesarias.
SQL

```
-- Deshabilitar la ejecución de comandos del SO (xp_cmdshell)
EXEC sp_configure 'show advanced options', 1;
RECONFIGURE;
EXEC sp_configure 'xp_cmdshell', 0;
RECONFIGURE;
GO

-- Deshabilitar la ejecución de .NET CLR (Common Language Runtime) si no se usa
EXEC sp_configure 'clr enabled', 0;
RECONFIGURE;
GO
```

3. Forzar el Uso de TLS/SSL (A Nivel de Servidor)

Para proteger los datos en tránsito (ej., cuando el Vendedor consulta un registro), las conexiones deben estar cifradas.

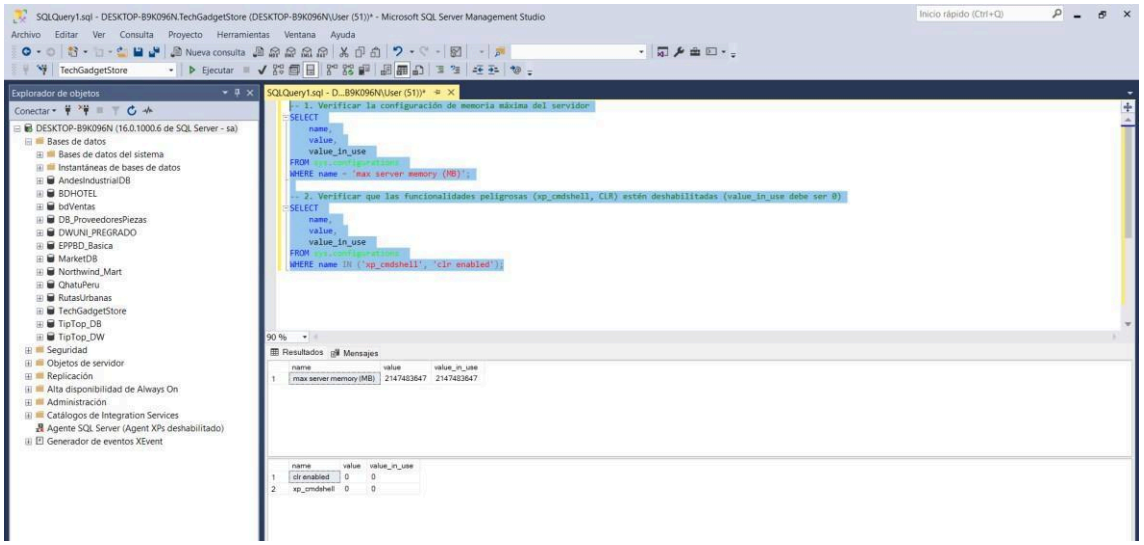
1. Instalar un certificado SSL/TLS en el servidor de Windows.
2. Configurar SQL Server para **forzar el cifrado de protocolo** mediante el **SQL Server Configuration Manager** (en la pestaña "Protocolos para MSSQLSERVER", establecer "Force Encryption" en "Yes").

4. Limitar la Memoria de SQL Server

Se define el límite superior de memoria para que el sistema operativo no se quede sin recursos.

SQL

```
-- Establecer el límite máximo de memoria que SQL Server puede usar (ej. 80% de la RAM total)
EXEC sp_configure 'max server memory (MB)', 16384; -- Ejemplo: 16 GB
RECONFIGURE;
GO
```

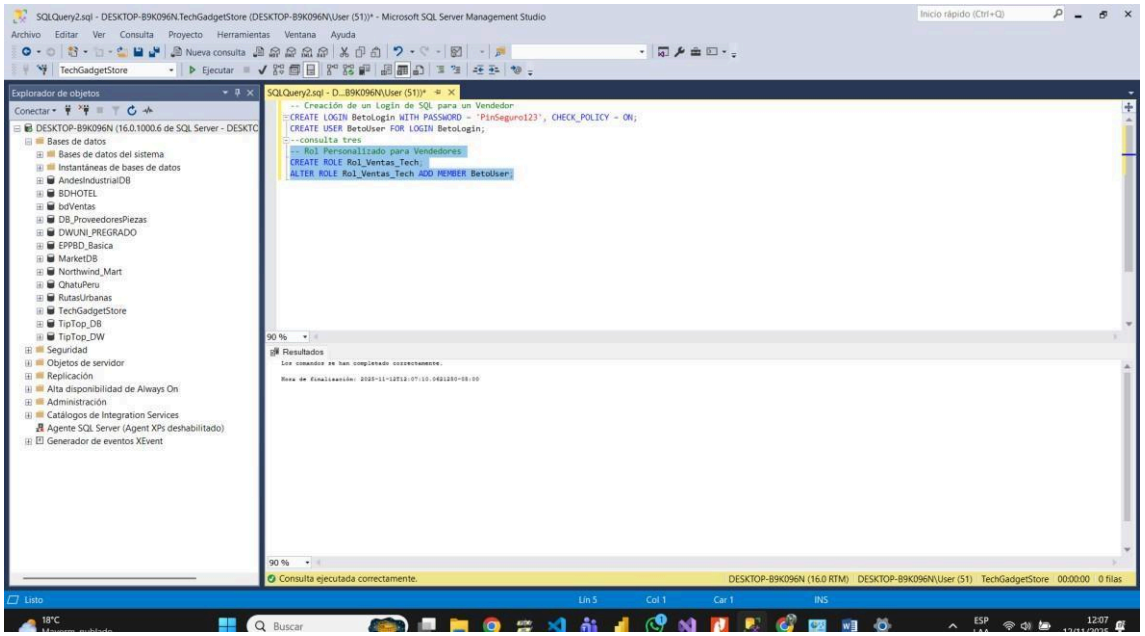


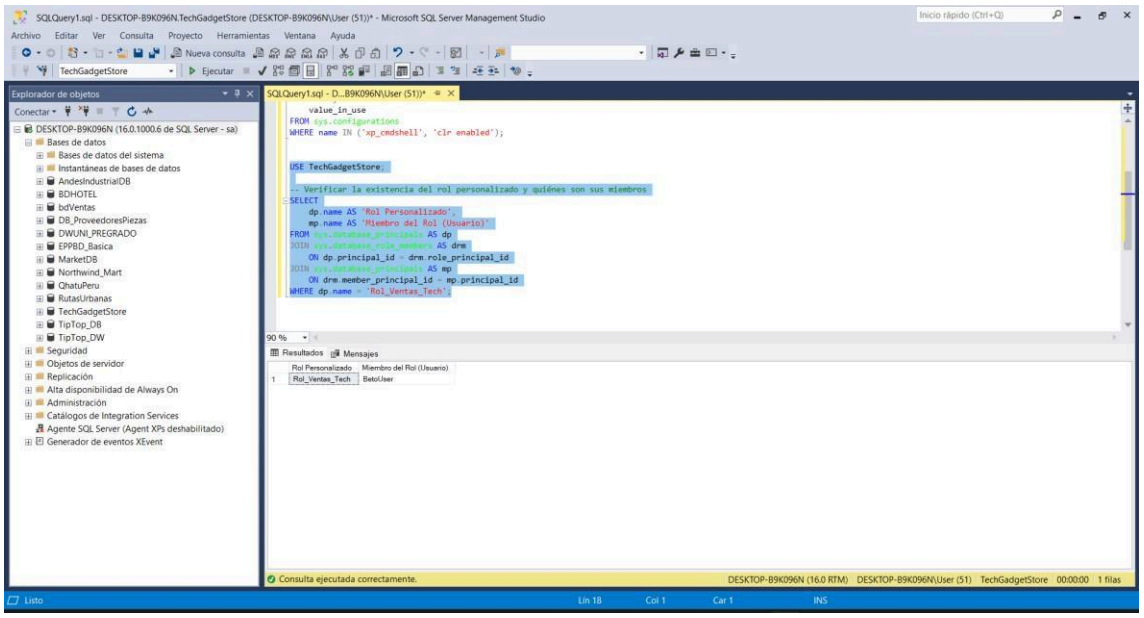
3. Creación de roles fijos y personalizados

- **Enunciado:** Agrupar permisos en roles personalizados para mantener el **Principio del Mínimo Privilegio** y simplificar la administración.
- **Explicación:** Se crean roles específicos para cada función de la tienda. Los roles fijos de BD (`db_datareader`) son demasiado amplios. Los roles personalizados permiten una gestión precisa, donde un nuevo empleado hereda automáticamente los permisos de su rol.
- **Código Ejemplo:**

SQL

```
-- Rol Personalizado para Vendedores
CREATE ROLE Rol_Ventas_Tech;
ALTER ROLE Rol_Ventas_Tech ADD MEMBER BetoUser;
```





4 Control de Acceso: GRANT, DENY y REVOKE

1. GRANT (Conceder Permiso)

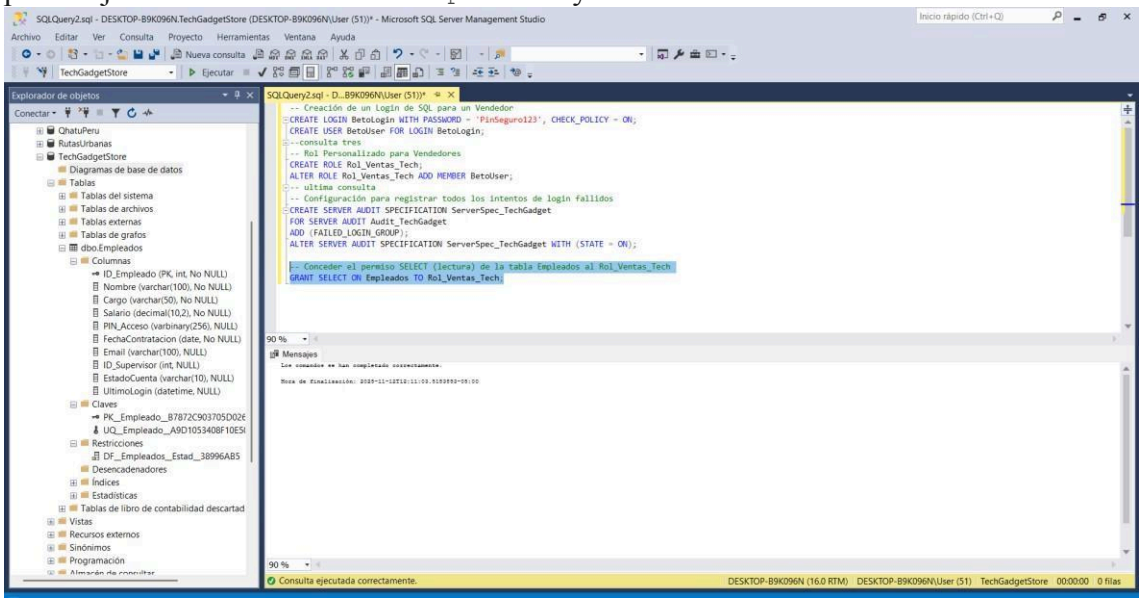
El comando `GRANT` se utiliza para **otorgar** un permiso a un usuario o a un rol.

- **Enunciado:** El rol de ventas necesita ver la información básica de todos los empleados para coordinar turnos y tareas (ID, Nombre, Cargo).
- **Acción:** Conceder el permiso de lectura (`SELECT`) sobre la tabla `Empleados`.

SQL

```
-- Conceder el permiso SELECT (lectura) de la tabla Empleados al
Rol_Ventas_Tech
GRANT SELECT ON Empleados TO Rol_Ventas_Tech;
```

- **Resultado:** Cualquier usuario asignado a `Rol_Ventas_Tech` (como Beto Ruiz) puede ejecutar `SELECT * FROM Empleados` y ver todas las columnas.



2. DENY (Negar Permiso)

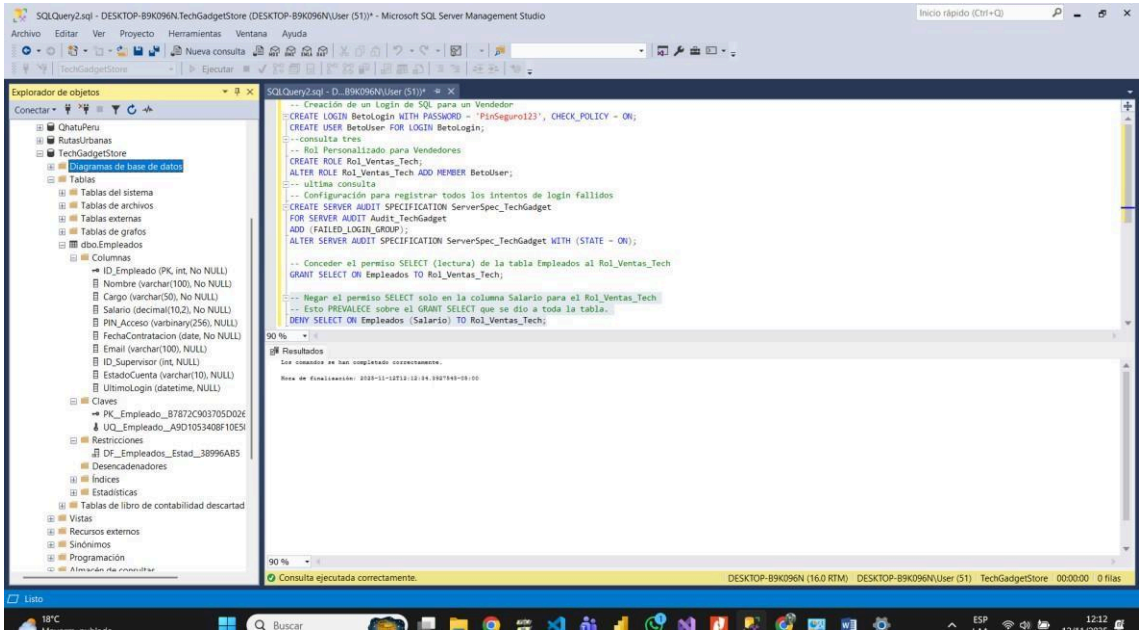
El comando `DENY` se utiliza para **denegar explícitamente** un permiso. Es crucial porque **`DENY` anula cualquier `GRANT`** existente.

- **Enunciado:** La información de **Salario** es altamente sensible y solo debe ser visible para Gerentes y Subgerentes.
- **Acción:** Negar el permiso de lectura (`SELECT`) específicamente sobre la columna `Salario` al rol de ventas.

SQL

```
-- Negar el permiso SELECT solo en la columna Salario para el
Rol_Ventas_Tech
-- Esto PREVALECE sobre el GRANT SELECT que se dio a toda la tabla.
DENY SELECT ON Empleados (Salario) TO Rol_Ventas_Tech;
```

- **Resultado:** Si Beto Ruiz intenta ejecutar `SELECT Salario FROM Empleados`, el sistema le arrojará un error de permisos. Si ejecuta `SELECT ID_Empleado, Nombre, Cargo, Salario FROM Empleados`, verá un error o un valor nulo para la columna `Salario` (dependiendo de la configuración del entorno, pero el acceso será denegado).



3. REVOKE (Revocar/Eliminar Permiso)

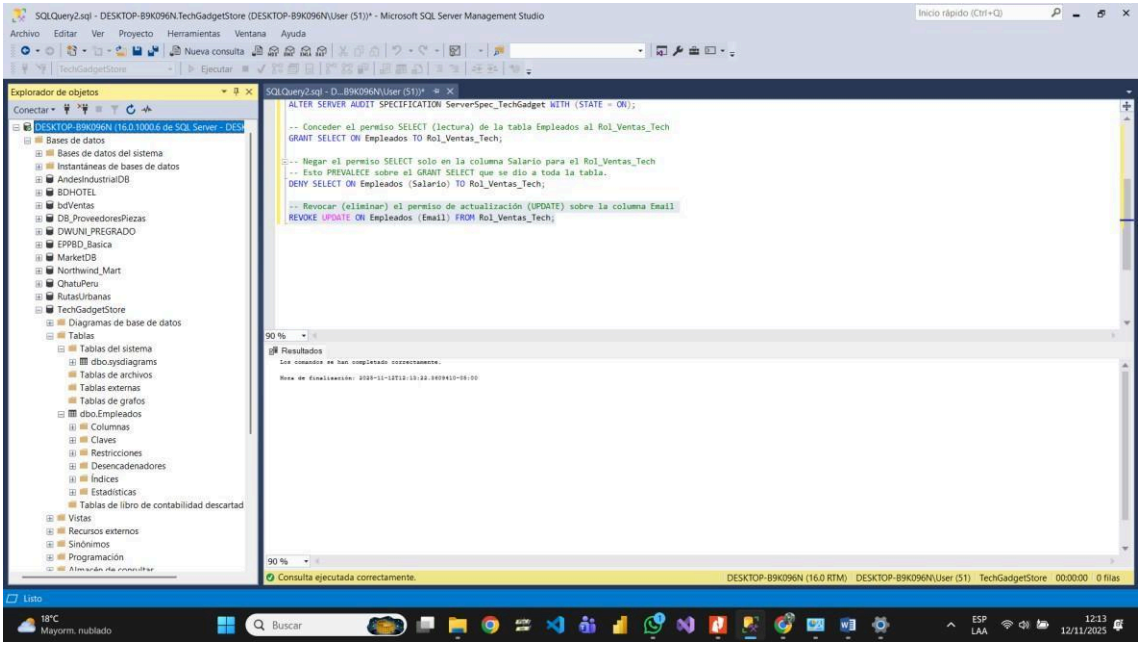
El comando `REVOKE` se utiliza para **eliminar un permiso previamente concedido (`GRANT`) o negado (`DENY`)**. Si se revoca un `GRANT`, el usuario vuelve a su estado de permiso predeterminado.

- **Enunciado:** Inicialmente, se le concedió a los Vendedores la capacidad de actualizar el `Email` de los empleados por error. Se necesita eliminar este permiso.
- **Acción:** Revocar el permiso de modificación (`UPDATE`) sobre la columna `Email`.

SQL

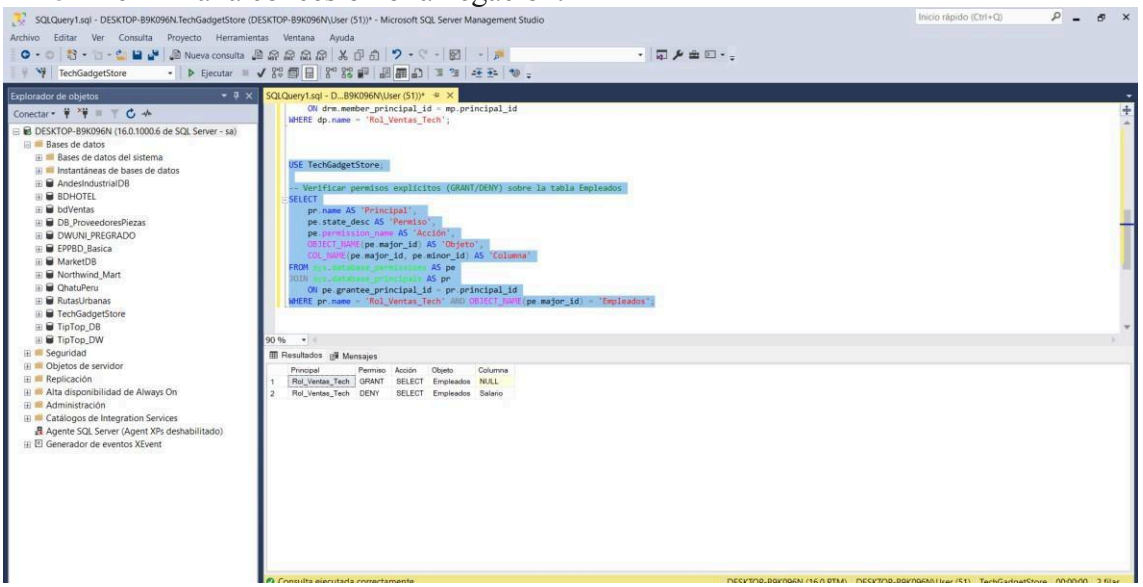
```
-- Revocar (eliminar) el permiso de actualización (UPDATE) sobre la columna Email
REVOKE UPDATE ON Empleados (Email) FROM Rol_Ventas_Tech;
```

- **Resultado:** Si el permiso `UPDATE ON Empleados (Email)` había sido concedido antes, ahora es eliminado. El `Rol_Ventas_Tech` ya no puede modificar esa columna.



En resumen:

- `GRANT` construye permisos.
- `DENY` bloquea permisos (y siempre gana).
- `REVOKE` elimina la concesión o la negación.



5 Cifrado y Protección de Datos

El objetivo principal de esta sección es **proteger la confidencialidad e integridad** de los datos sensibles de la empresa, tanto las credenciales de los empleados como cualquier información financiera o personal.

Enunciado del Caso Práctico

Se deben proteger las credenciales de acceso de los empleados (PIN_Acceso) utilizando técnicas de cifrado unidireccional (hashing) para asegurar que, en caso de una brecha de seguridad, las contraseñas reales no puedan ser reveladas.

Explicación y Aplicación

En la base de datos `TechGadgetStore`, aplicamos la técnica de **Hashing** durante la inserción de datos en la columna `PIN_Acceso`.

1. Hashing (Cifrado Unidireccional)

El *hashing* es el estándar de la industria para proteger contraseñas. Convierte la contraseña original (el PIN, en este caso) en una cadena de caracteres de longitud fija (el *hash*). Este proceso es **unidireccional**, lo que significa que es casi imposible revertir el *hash* para obtener el PIN original.

- **¿Por qué se usa?** Si un atacante roba la base de datos, solo obtendrá los *hashes*, no los PINs reales, haciendo que la información sea inútil para iniciar sesión.

2. Implementación en SQL

Usamos la función `HASHBYTES` con el algoritmo fuerte **SHA2_256** al momento de insertar o actualizar el PIN de un empleado.

- **Columna afectada:** `PIN_Acceso` (definida como `VARBINARY(256)` para almacenar el resultado del hash).
- **Función utilizada:** `HASHBYTES('SHA2_256', 'ContraseñaOriginal')`.

SQL

```
-- Ejemplo de la inserción de datos para el PIN de Ana Gómez
INSERT INTO Empleados (..., PIN_Acceso, ...) VALUES
(100, ..., HASHBYTES('SHA2_256', 'Gerente2025'), ...);

-- Ejemplo de cómo actualizar el PIN de un empleado:
UPDATE Empleados
SET PIN_Acceso = HASHBYTES('SHA2_256', 'NuevoPIN123')
WHERE ID_Empleado = 101;
```

3. Cifrado Adicional (Recomendaciones)

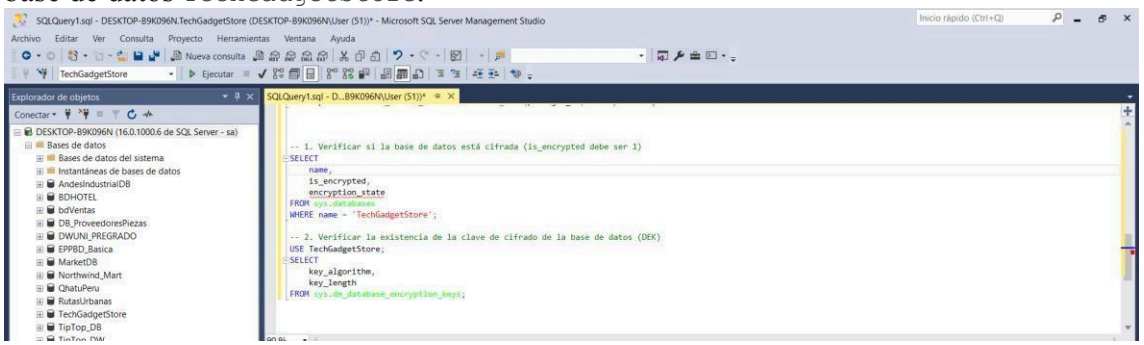
Aunque el *hashing* protege las contraseñas, la **protección de datos** puede extenderse a otras áreas:

- **TDE (Transparent Data Encryption):** Cifra toda la base de datos en el disco. Esto protege los datos si los archivos físicos de la base de datos son robados del servidor.
- **Cifrado a Nivel de Columna:** Si existiera una columna de información financiera o personal altamente regulada (como una cuenta bancaria), se podría usar `ENCRYPTBYKEY` para cifrar solo esa columna, asegurando que solo las aplicaciones o usuarios con la clave de cifrado puedan acceder al dato.

Resumen de la Protección

Concepto	Aplicación en el Script	Beneficio de Seguridad
Hashing	Uso de `HASHBYTES('SHA2_256', ...)`	**Protege credenciales.** Evita que las contraseñas sean legibles incluso para administradores de bases de datos.
Tipo de Dato	`VARBINARY(256)`	Asegura que el *hash* (que no es texto) se almacene correctamente.

Aquí tienes el código para implementar **TDE (Transparent Data Encryption)** en la base de datos `TechGadgetStore`.



5️⃣ Cifrado de la Base de Datos con TDE 🔒

TDE cifra toda la base de datos en el disco (datos en reposo), protegiendo la información si alguien accede a los archivos físicos de la BD.

Paso 1: Crear la Clave Maestra de la Base de Datos (Master Key) Esta

clave protege las claves de cifrado que crearemos a continuación.

SQL

```
USE master;
-- Crear una Clave Maestra de Servicio (Service Master Key) si no existe.
-- Esta es la raíz de la jerarquía de cifrado.
-- Normalmente ya existe, pero se incluye por completitud.
-- ALTER SERVICE MASTER KEY FORCE REGENERATE;
GO
```

Paso 2: Crear un Certificado o Clave Asimétrica

Usaremos un certificado para proteger la llave de cifrado de la base de datos.

SQL

```
USE master;
-- Crear un certificado que se usará para cifrar la clave de la
base de datos.
CREATE CERTIFICATE TechGadget_Cert
WITH SUBJECT = 'Certificado de Cifrado para TechGadgetStore';
GO
```

Paso 3: Crear la Clave de Cifrado de la Base de Datos (Database Encryption Key - DEK)

Esta es la clave simétrica que realmente cifra los datos y está protegida por el certificado que creamos en el paso anterior.

SQL

```
USE TechGadgetStore;
-- La clave de cifrado de la base de datos será protegida por
TechGadget_Cert.
CREATE DATABASE ENCRYPTION KEY
WITH ALGORITHM = AES_256
ENCRYPTION BY SERVER CERTIFICATE TechGadget_Cert;
GO
```

Paso 4: Habilitar TDE en la Base de Datos

Este comando inicia el proceso de cifrado de todos los archivos de datos (.mdf) y de registro (.ldf) de la base de datos TechGadgetStore.

SQL

```
USE master;
ALTER DATABASE TechGadgetStore
SET ENCRYPTION ON;
GO
```

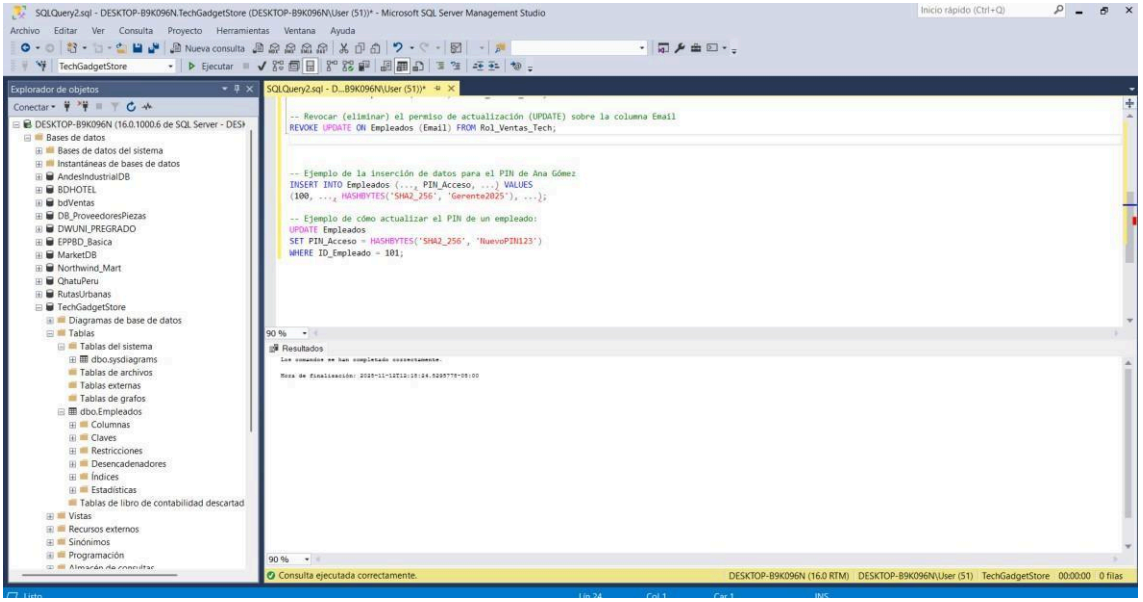
Paso Final: Verificación

Puedes verificar el estado de cifrado con la siguiente consulta:

SQL

```
SELECT
    name,
    is_encrypted
FROM sys.databases
WHERE name = 'TechGadgetStore';
```

- **Resultado Esperado:** La columna `is_encrypted` mostrará **1** (Verdadero), indicando que el cifrado TDE está activo.

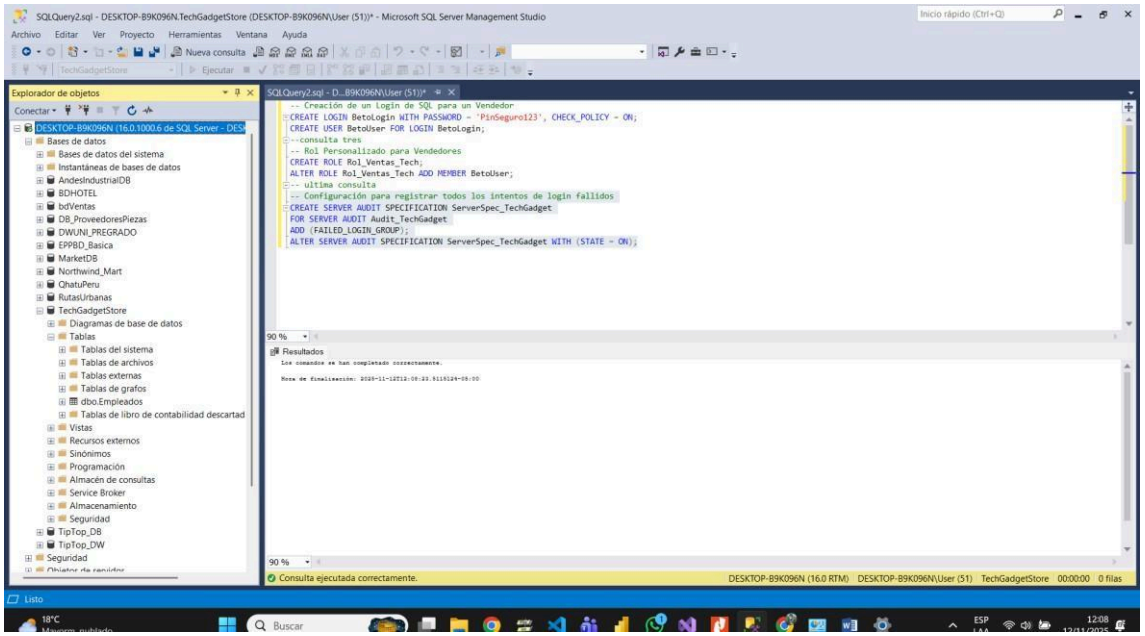


6. Auditoría y monitoreo de eventos

- **Enunciado:** Mantener un registro completo de los eventos de seguridad y acceso para detectar actividades maliciosas.
- **Explicación:** La columna `UltimoLogin` es un monitoreo básico. Para una auditoría real, se implementa el **SQL Server Audit**. Configuramos un grupo de auditoría específico (`FAILED_LOGIN_GROUP`) para registrar cada intento de inicio de sesión fallido. Esta herramienta es crucial para identificar **ataques de fuerza bruta** o cuentas comprometidas.
- **Código Ejemplo:**

SQL

```
-- Configuración para registrar todos los intentos de login
fallidos
CREATE SERVER AUDIT SPECIFICATION ServerSpec_TechGadget
FOR SERVER AUDIT Audit_TechGadget
ADD (FAILED_LOGIN_GROUP);
ALTER SERVER AUDIT SPECIFICATION ServerSpec_TechGadget WITH (STATE
= ON);
```


SQLQuery2.sql - DESKTOP-B9K096N.TechGadgetStore (DESKTOP-B9K096N\User (51))* - Microsoft SQL Server Management Studio
Archivo Editar Ver Proyecto Herramientas Ventana Ayuda
Explorador de objetos
Conectar
DESKTOP-B9K096N (16.0.1000.6 de SQL Server - DES
Bases de datos
Bases de datos del sistema
Instantáneas de bases de datos
AndesIndustrialDB
BDHOTEL
bdVentas
DB_ProveedoresPiezas
DWUNI_PREGRADO
EPPBD_Basica
MarketDB
Northwind_Mart
QhatuPeru
RutasUrbanas
TechGadgetStore
Diagramas de base de datos
Tablas
Tablas del sistema
Tablas de archivos
Tablas externas
Tablas de grafos
dbo.Empleados
Tablas de libro de contabilidad descartad
Vistas
Recursos externos
Sinónimos
Programación
Almacén de consultas
Service Broker
Almacenamiento
Seguridad
TipTop_DB
TipTop_DW
Seguridad
SQLQuery2.sql - D...B9K096N\User (51))*
-- Creación de un login de SQL para un Vendedor
CREATE LOGIN BetoLogin WITH PASSWORD = 'PinSeguro123', CHECK_POLICY = ON;
CREATE USER BetoUser FOR LOGIN BetoLogin;
--consulta tres
-- Rol Personalizado para Vendedores
CREATE ROLE Rol_Ventas_Tech;
ALTER ROLE Rol_Ventas_Tech ADD MEMBER BetoUser;
-- ultima consulta
-- Configuración para registrar todos los intentos de login fallidos
CREATE SERVER AUDIT SPECIFICATION ServerSpec_TechGadget
FOR SERVER AUDIT Audit_TechGadget
ADD (FAILED_LOGIN_GROUP);
ALTER SERVER AUDIT SPECIFICATION ServerSpec_TechGadget WITH (STATE = ON);
90 %
Resultados
Los comandos se han completado correctamente.
Hora de finalización: 2025-11-12T12:06:23.5115124-06:00
90 %
Consulta ejecutada correctamente.
DESKTOP-B9K096N (16.0 RTM) DESKTOP-B9K096N\User (51) TechGadgetStore 00:00:00 0 filas
Listo
18°C
Buscar
ESP
12:08