UNIVERSIDAD PERUANA LOS ANDES FACULTAD DE INGENIERÍA

Asignatura: Base de Datos II

Docente: Dr. Raúl Enrique Fernández Bejarano

Alumno: SORIANO TIMOTEO Joel Kevin

Ciclo: V

huancayo-Perú-2025

Manual de Seguridad y Control de Acceso

Base de Datos: TechGadget Store

El siguiente bloque de código SQL crea la base de datos **TechGadgetStore**, define la tabla **Empleados**, e inserta 10 registros de ejemplo.
La columna de seguridad principal es **PIN_Acceso**, protegida mediante el uso de **HASHBYTES (SHA2_256)** para asegurar los datos sensibles.

```
-- 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),   -- Guarda el hash SHA2_256
del PIN

    FechaContratacion DATE NOT NULL,

    Email VARCHAR(100) UNIQUE,

    ID_Supervisor INT,
```

```
    EstadoCuenta VARCHAR(10) DEFAULT 'Activo',
    UltimoLogin DATETIME         -- Auditoría y monitoreo
);
GO

-- 2. INSERCIÓN DE 10 REGISTROS DEMOSTRATIVOS
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)

A continuación, se desarrolla el punto **1: Autenticación SQL y Windows**, manteniendo su estructura original pero expresado con un texto diferente y más detallado.

1️⃣ Autenticación SQL y Windows: Datos y Pasos

Este apartado define cómo un usuario valida su identidad para ingresar al servidor SQL.
 SQL Server permite dos esquemas principales de autenticación:

Modo de Autenticación	Descripción	Uso Recomendado
Autenticación de Windows	El proceso de verificación lo realiza el sistema operativo. SQL Server confía en esa autenticación y no solicita una contraseña adicional.	Recomendado para administradores, gerentes y la mayoría de usuarios corporativos, ya que es el método más seguro.
Autenticación de SQL Server	SQL Server administra sus propias cuentas, almacenando internamente los usuarios y contraseñas.	Ideal para aplicaciones (como sistemas POS), usuarios con permisos limitados o ambientes que no dependen de Windows.

Configuración del Servidor (Paso a Paso)

Antes de registrar usuarios, es necesario ajustar la configuración del servidor para permitir ambos métodos de autenticación.

Paso 1: Activar el Modo de Autenticación Mixta

En muchas instalaciones, SQL Server viene configurado únicamente con **Autenticación de Windows**.
Para trabajar con ambos modos se debe habilitar el **Modo Mixto**:

1. Abrir *SQL Server Management Studio (SSMS)*.
2. Conectarse al servidor SQL.
3. Clic derecho sobre el nombre del servidor → **Propiedades**.
4. Entrar a la pestaña **Seguridad**.
5. Seleccionar **Autenticación de SQL Server y Windows**.
6. Aceptar los cambios.
7. Reiniciar el servicio SQL Server para aplicar la nueva configuración.

Paso 2: Buenas Prácticas para la Cuenta *sa*

Al activar el modo mixto, la cuenta **sa** se habilita automáticamente.

Medidas recomendadas:

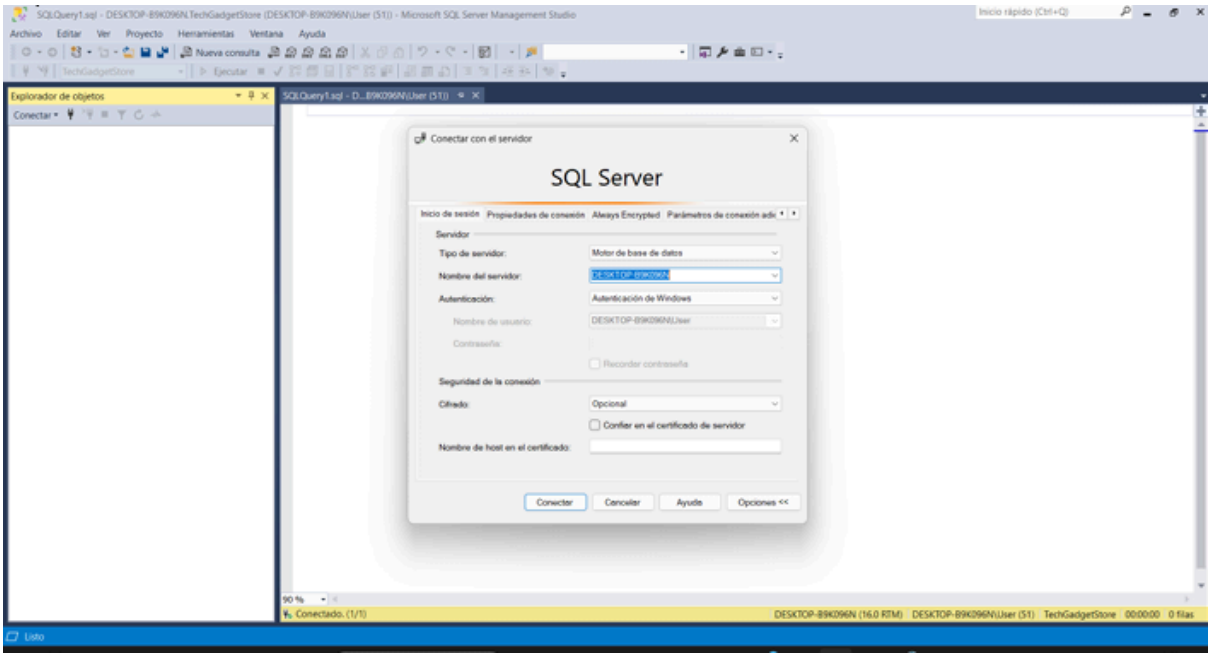
- Definir una contraseña robusta (mayúsculas, minúsculas, símbolos y longitud adecuada).
- Deshabilitar la cuenta sa si el entorno ya tiene administradores de Windows.

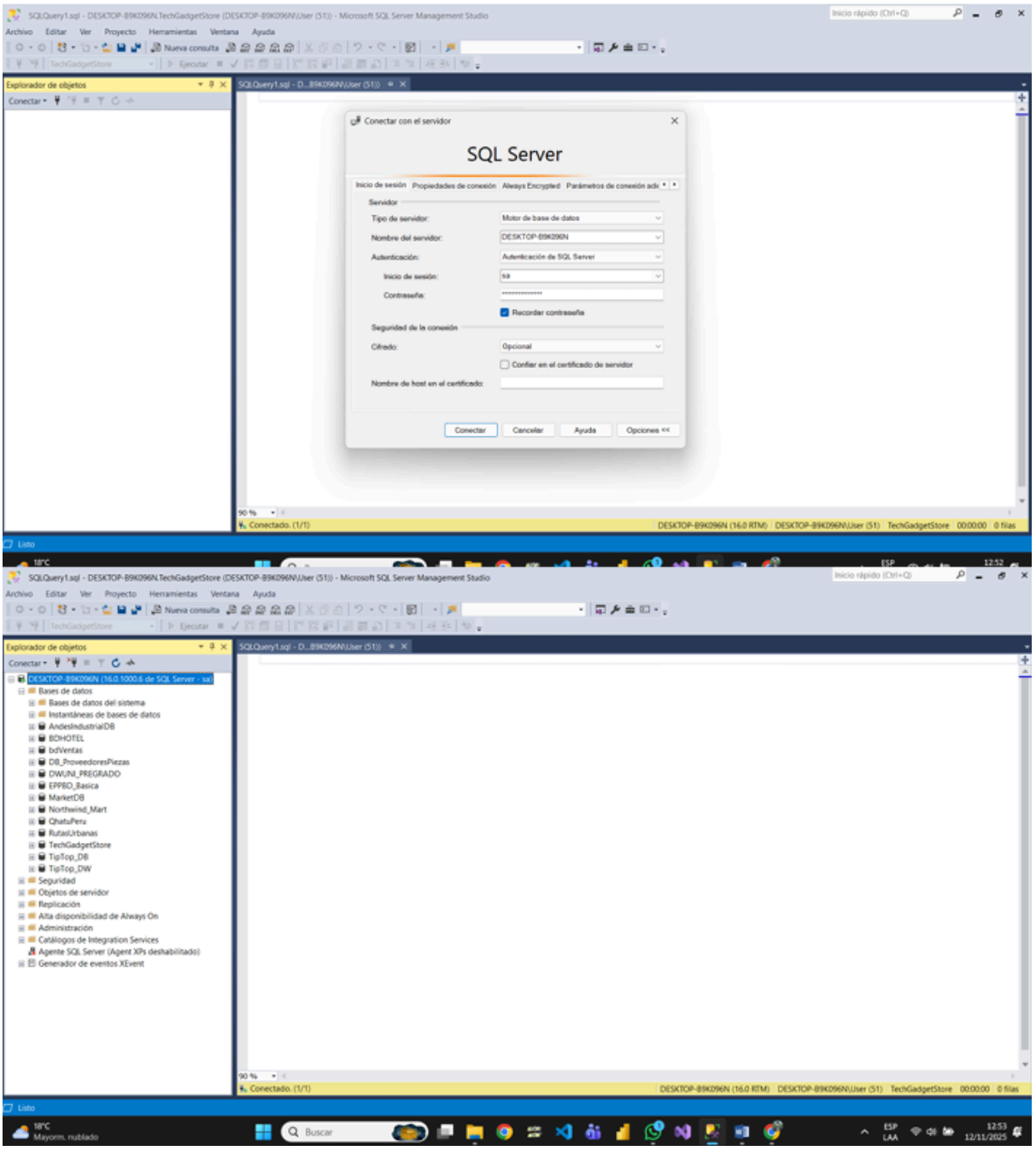
```
-- Asignar una contraseña fuerte al login 'sa'
ALTER LOGIN sa
WITH PASSWORD = 'TuContraseñaAdmin!@#123',
CHECK_POLICY = ON;
GO
```

```
-- Deshabilitar la cuenta 'sa' si existen otros
administradores

ALTER LOGIN sa DISABLE;

GO
```





Creación de Logins y Usuarios (Paso a Paso)

En esta sección se configuran los accesos que utilizarán los colaboradores de **TechGadgetStore** para entrar al servidor y a la base de datos.

1. Autenticación de Windows (Para Gerentes y Administradores)

- **Enunciado:** La gerente **Ana Gómez** accederá mediante su cuenta de Windows.
- **Práctica Segura:** Utilizar siempre credenciales del dominio, ya que la verificación la realiza el sistema operativo.

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

```
USE master;


-- 1. Crear el Login basado en Windows (dominio ficticio TECHGADGET_DOMAIN)
CREATE LOGIN [TECHGADGET_DOMAIN\AnaGomez] FROM WINDOWS;
GO


USE TechGadgetStore;


-- 2. Crear el usuario dentro de la base de datos (AnaUser) enlazado al Login
CREATE USER AnaUser FOR LOGIN [TECHGADGET_DOMAIN\AnaGomez];
GO
```

```
-- 3. Otorgar el rol db_owner, adecuado para un perfil
gerencial

ALTER ROLE db_owner ADD MEMBER AnaUser;

GO
```

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

- **Enunciado:** El vendedor **Beto Ruiz** utilizará un Login nativo de SQL Server, apropiado para sistemas POS o cuentas independientes del dominio.
- **Práctica Segura:** habilitar políticas de contraseñas para reforzar seguridad (**CHECK_POLICY = ON**).

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

```
USE master;


-- 1. Crear Login local de SQL Server para el vendedor

CREATE LOGIN BetoLogin

WITH PASSWORD = 'ContraseñaDeBeto#1',

CHECK_POLICY = ON;       -- Fuerza normas de seguridad en la
contraseña

GO
```

```
USE TechGadgetStore;


-- 2. Crear el usuario BetoUser asociado al Login recién creado

CREATE USER BetoUser FOR LOGIN BetoLogin;

GO


-- 3. Incorporar al usuario dentro del rol de Vendedores

ALTER ROLE Rol_Ventas_Tech ADD MEMBER BetoUser;

GO
```

Diferencias Clave y Prácticas Seguras

Característica	Autenticación de Windows	Autenticación de SQL Server
Validación	Realizada por el sistema operativo (Dominio/AD).	Verificación hecha por SQL Server.
Contraseña	Gestionada por el dominio o SO.	Administrada exclusivamente en SQL Server.
Seguridad	Muy alta; emplea Kerberos y evita guardar	Media; exige contraseñas robustas

	contraseñas dentro del servidor SQL.	y políticas estrictas.
Práctica Recomendada	Preferida en organizaciones por seguridad e integración.	Adecuada para apps externas o cuando no se dispone de dominio.

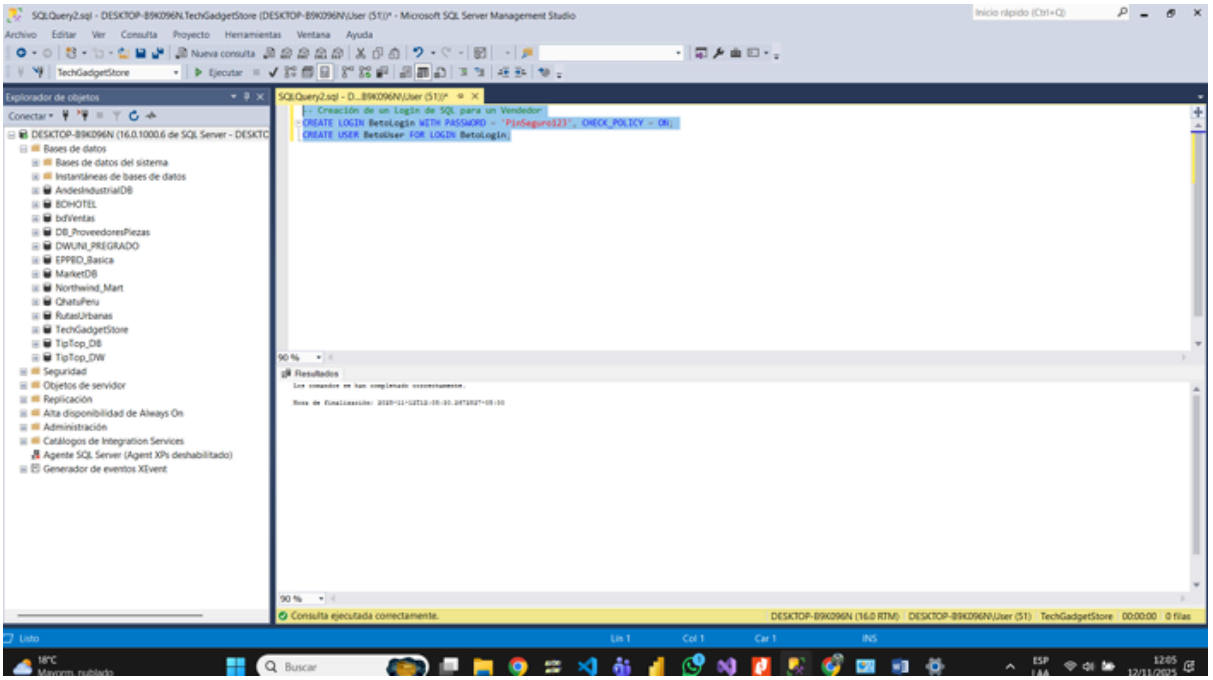
```
-- Ejemplo adicional de creación de Login SQL para un vendedor

CREATE LOGIN BetoLogin

WITH PASSWORD = 'PinSeguro123',

CHECK_POLICY = ON;


CREATE USER BetoUser FOR LOGIN BetoLogin;
```



Cuentas de Servicio y Configuración del Servidor (Paso a Paso)

Caso Práctico – TechGadgetStore

Los servicios principales de SQL Server de la empresa –**MSSQLSERVER** (motor de base de datos) y **SQLSERVERAGENT** (tareas programadas)– deben ejecutarse utilizando **Cuentas de Servicio Administradas (MSA/gMSA)** y deben configurarse opciones de seguridad para reducir la superficie de ataque.

A. Explicación Fundamental

¿Qué son las Cuentas de Servicio?

Son identidades de Windows bajo las cuales se ejecutan los servicios del motor SQL Server.
Ejemplos:

- MSSQLSERVER → Motor de BD
- SQLSERVERAGENT → Agendador de tareas

Riesgo si no se configuran correctamente

Si SQL Server se ejecuta con cuentas muy privilegiadas como:

- **Local System**
- **Administrador de Dominio**

→ Un atacante podría obtener control total del servidor Windows.

B. Práctica Segura: Principio del Mínimo Privilegio (PoLP)

1. **Cuentas distintas** para cada servicio (Motor, Agent).

2. **Cuentas MSA o gMSA**
    - Sin manejo manual de contraseñas
    - Permisos mínimos
    - Administración automática desde Active Directory

C. Hardening del Servidor SQL (Explicación)

El hardening consiste en **desactivar funciones innecesarias** y **establecer límites** para evitar abusos o fugas de información.

Área	Por qué es Crítico
Conexiones Remotas	Reducir protocolos no usados y forzar cifrado.
Funciones Peligrosas	Evitar ejecución de código externo desde SQL.
Memoria del Servidor	Evitar que SQL consuma toda la RAM del sistema.

D. Pasos Prácticos

1️⃣ Configuración de Cuentas de Servicio (a nivel Windows)

📌 *Este paso NO se realiza con SQL, sino con SQL Server Configuration Manager.*

Pasos:

1. **Abrir SQL Server Configuration Manager**
2. **Detener los servicios**
    - SQL Server (MSSQLSERVER)
    - SQL Server Agent
3. **Cambiar cuenta de servicio**
    - Antes: NT SERVICE\MSSQLSERVER
    - Después: TECHGADGET_DOMAIN\SQL_Motor$ (MSA)
    - Para Agent: TECHGADGET_DOMAIN\SQL_Agent$
4. **Reiniciar ambos servicios**

2️⃣ Desactivar Funciones de Riesgo (a nivel SQL)

Desactivar xp_cmdshell

Permite ejecutar comandos del sistema → es peligroso si no es necesario.

```
EXEC sp_configure 'show advanced options', 1;
RECONFIGURE;


EXEC sp_configure 'xp_cmdshell', 0;
RECONFIGURE;
GO
```

Desactivar CLR si no se usa

```
EXEC sp_configure 'clr enabled', 0;
RECONFIGURE;
GO
```

3️⃣ Forzar Uso de TLS/SSL (a nivel Servidor)

Pasos (no SQL):

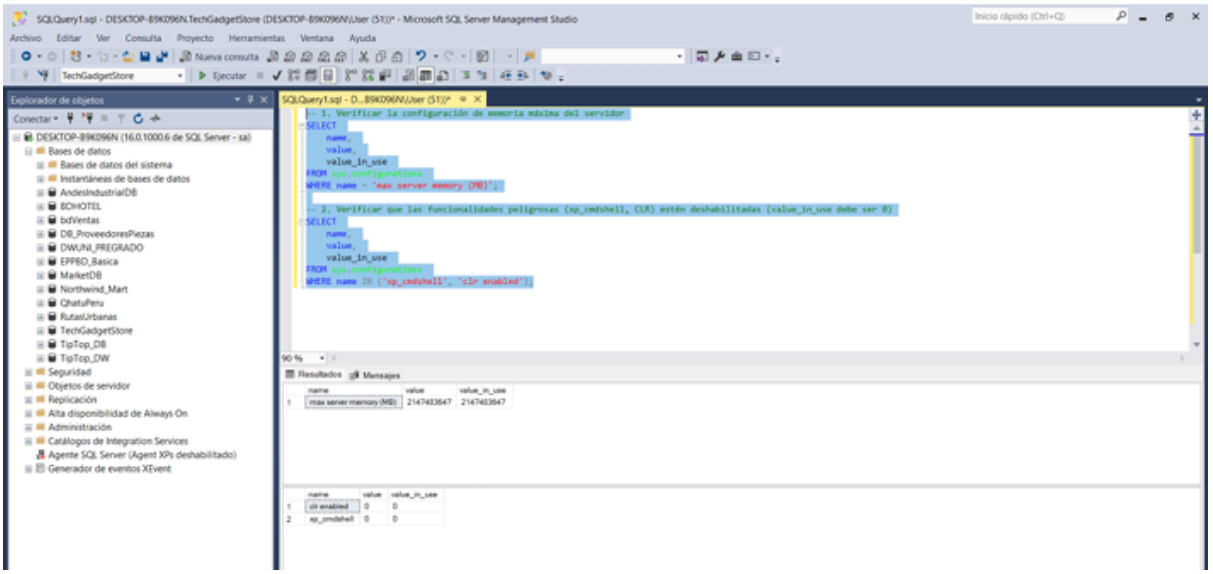
1. Instalar un certificado SSL/TLS válido en Windows Server.
2. Abrir SQL Server Configuration Manager.
3. Ir a **Protocols for MSSQLSERVER**.
4. Configurar:
    - **Force Encryption → Yes**
    - Seleccionar el certificado instalado.

Esto cifra todas las conexiones del vendedor, gerente y aplicaciones.

4️⃣ Limitar la Memoria de SQL Server

Ejemplo: Servidor con 32 GB RAM → SQL solo usará **16 GB (50%)**

```
EXEC sp_configure 'max server memory (MB)', 16384;
RECONFIGURE;
GO
```

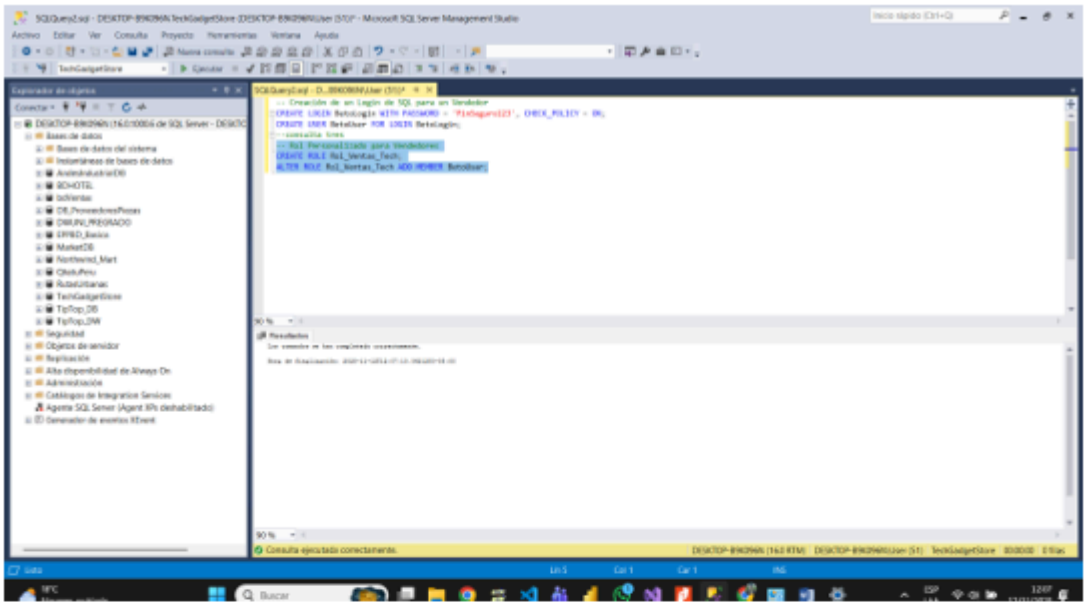


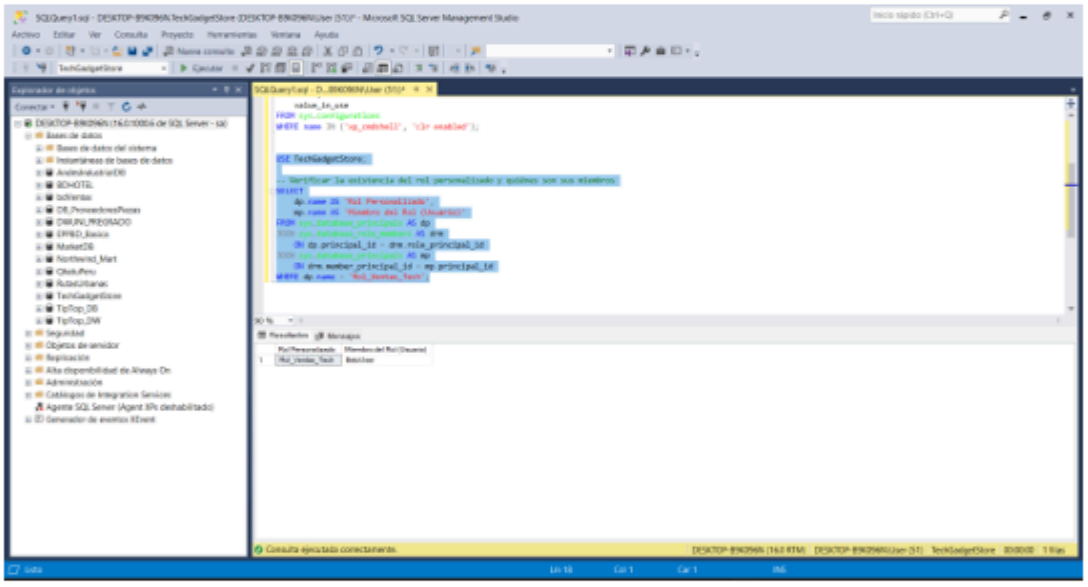
3. Creación de roles fijos y personalizados

- **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)

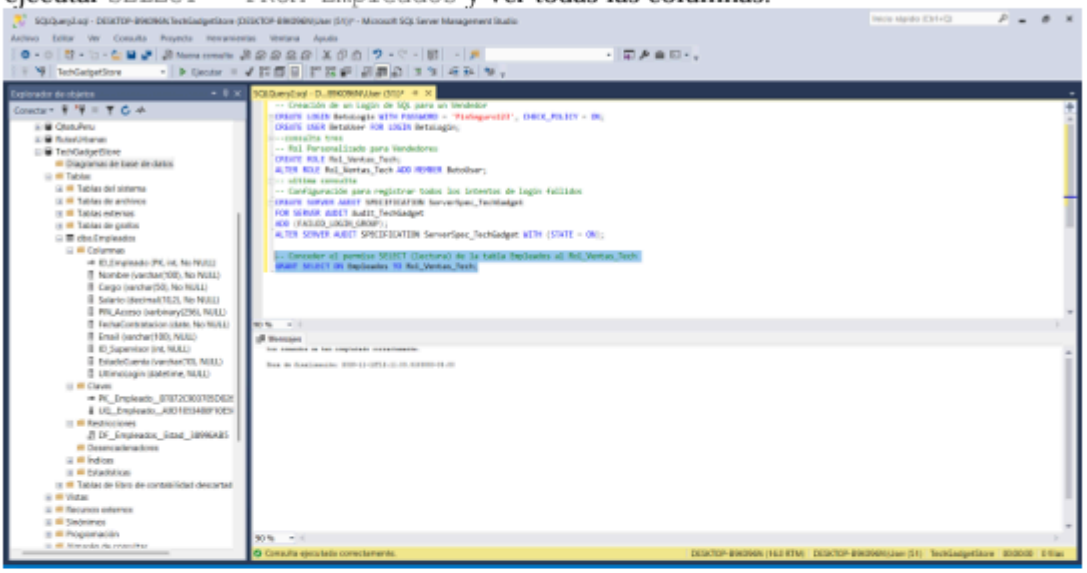
El comando `GRANT` se utiliza para **otorgar** un permiso a un usuario o a un rol.

- **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)

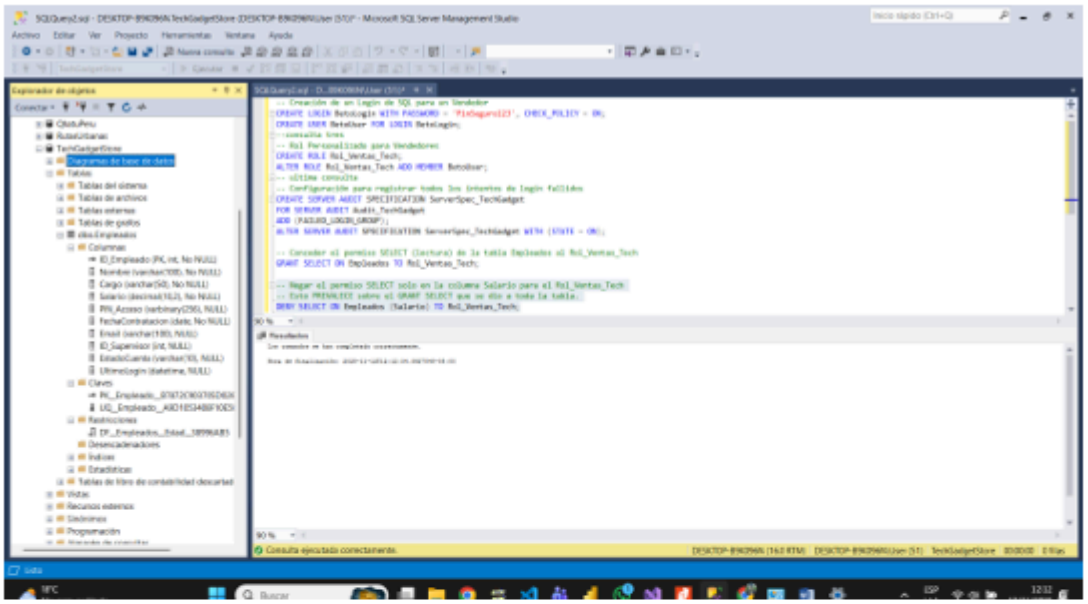
El comando `DENY` se utiliza para **denegar explícitamente** un permiso. Es crucial porque `DENY` **anula cualquier** `GRANT` existente.

- **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).



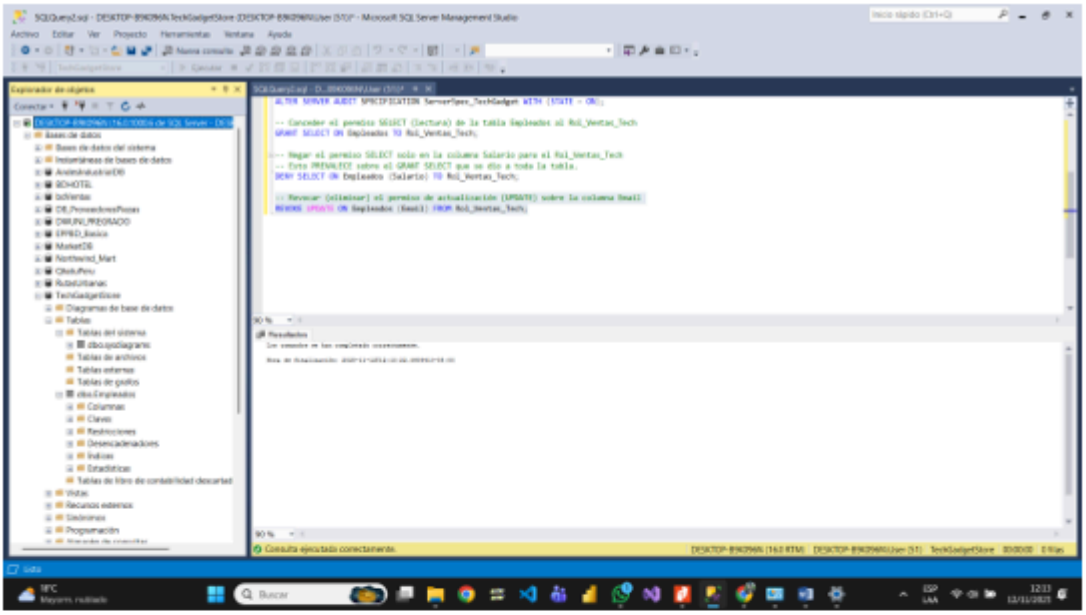
3. REVOKE (Revocar/Eliminar Permiso)

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

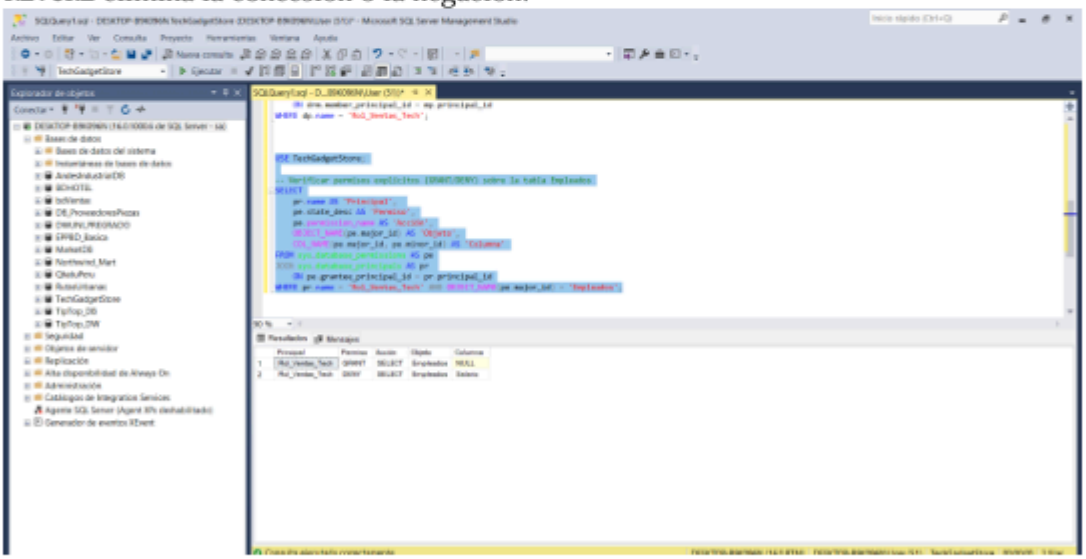
```
-- Revocar (eliminar) el permiso de actualización (UPDATE) sobre la columna Email
REVOKE UPDATE ON Empleados (Email) FROM Rol_Ventas_Tech;
```

- **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

```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)

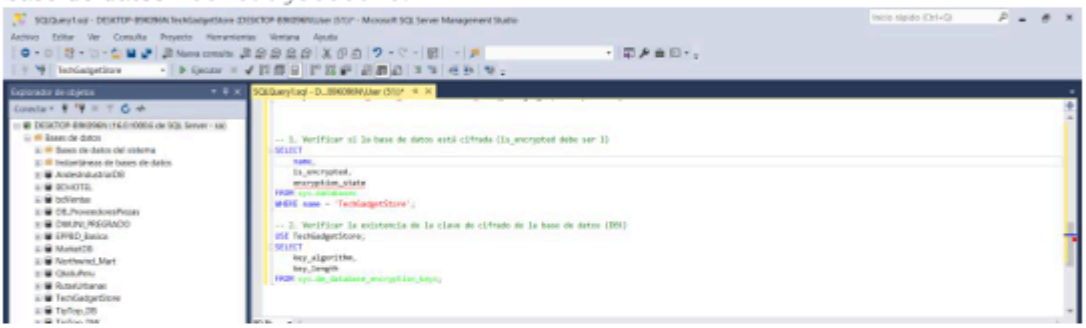
Aunque el *hashing* protege las contraseñas, la **protección de datos** puede extenderse a otras áreas:

- **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`.



5 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
```

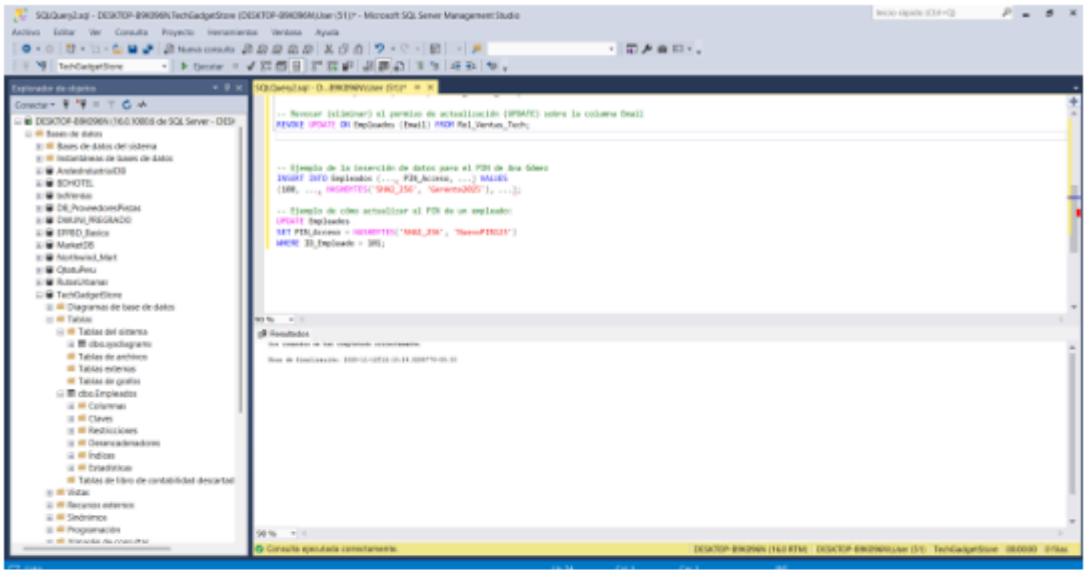
Paso Final: Verificación

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.

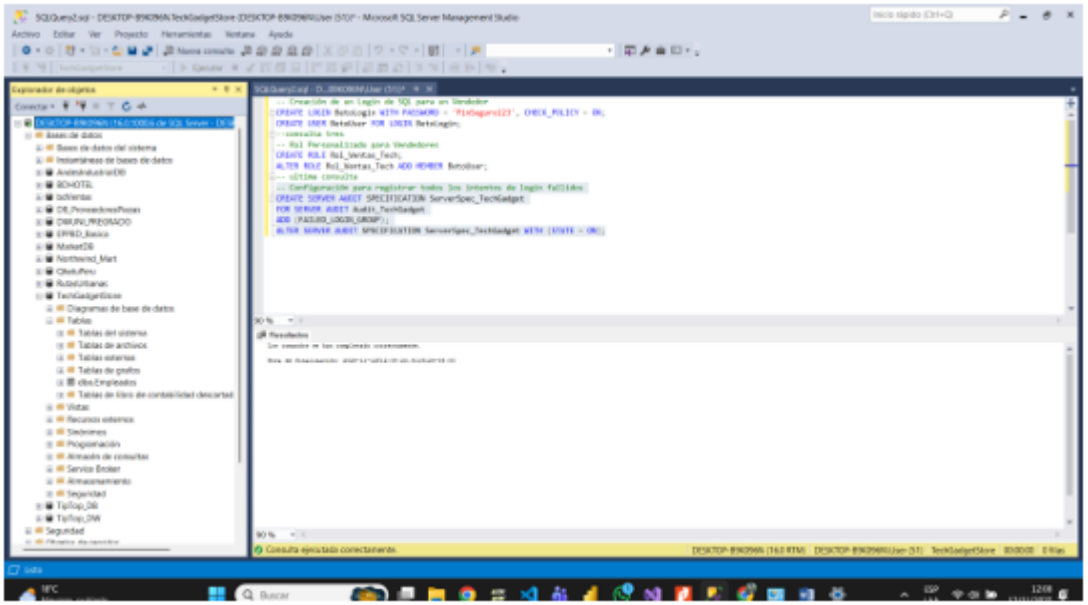


6. Auditoría y monitoreo de eventos

- **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);
```


-- Creación de un Login de SQL para un Vendedor
CREATE LOGIN BetoLogin WITH PASSWORD = 'Pi4Seguro123', 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);
Consulta ejecutada correctamente.