ACTIVIDAD

Responda con claridad y precisión cada uno de los siguientes ejercicios prácticos. Para cada caso, utilice el siguiente formato estructurado:

1. Enunciado del ejercicio
2. Script de la solución en T-SQL
3. Justificación técnica de la solución aplicada
4. Explicación de las buenas prácticas utilizadas en el proyecto

---

Proyecto 1: Autenticación: Comparación segura y configuración de logins

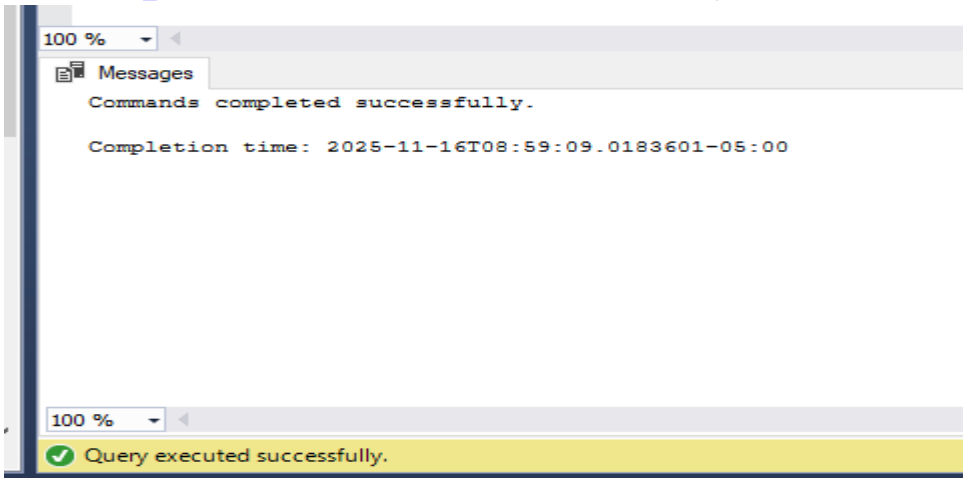
1. Enunciado del ejercicio

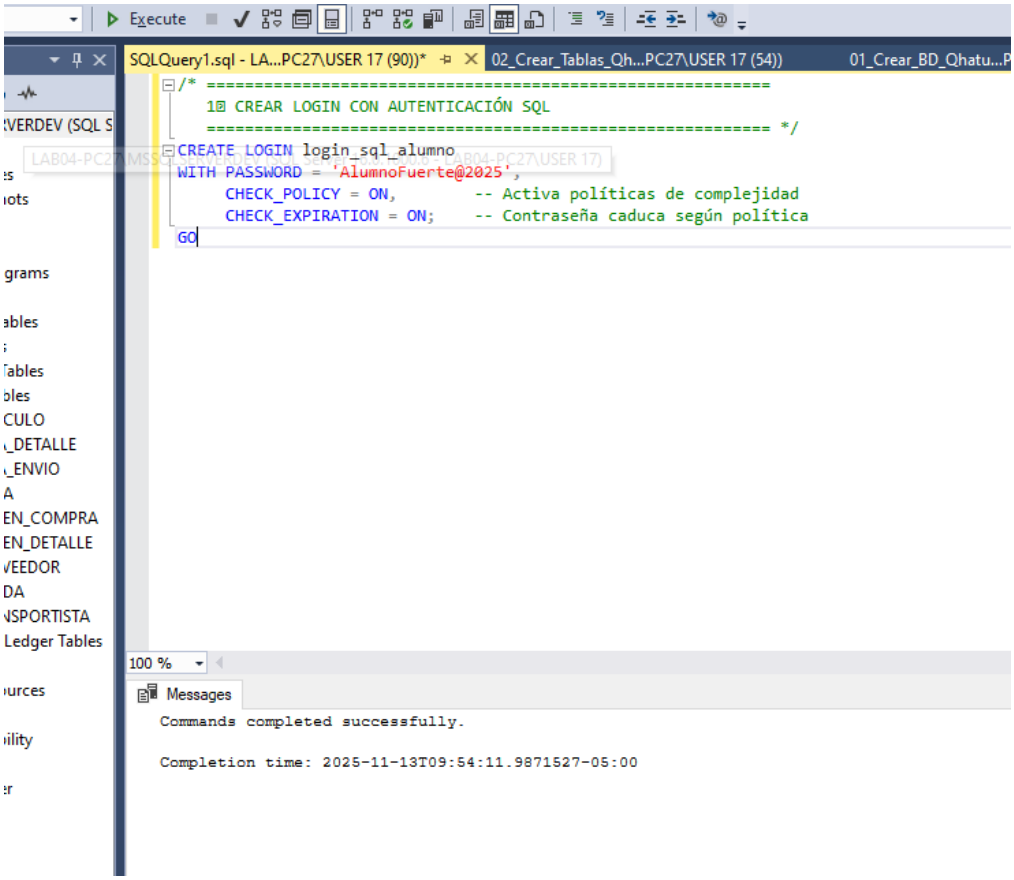
   Crear en el servidor dos logins de prueba: uno con autenticación SQL (login_sql_alumno) y otro que represente un usuario Windows (DOMAIN\alumno_win — simulado), aplicar políticas de contraseñas y mapear ambos a usuarios en la base QhatuPeru. Mostrar cómo forzar expiración y comprobar la política de contraseñas.

Enunciado del ejercicio

script de la solución en T-SQL

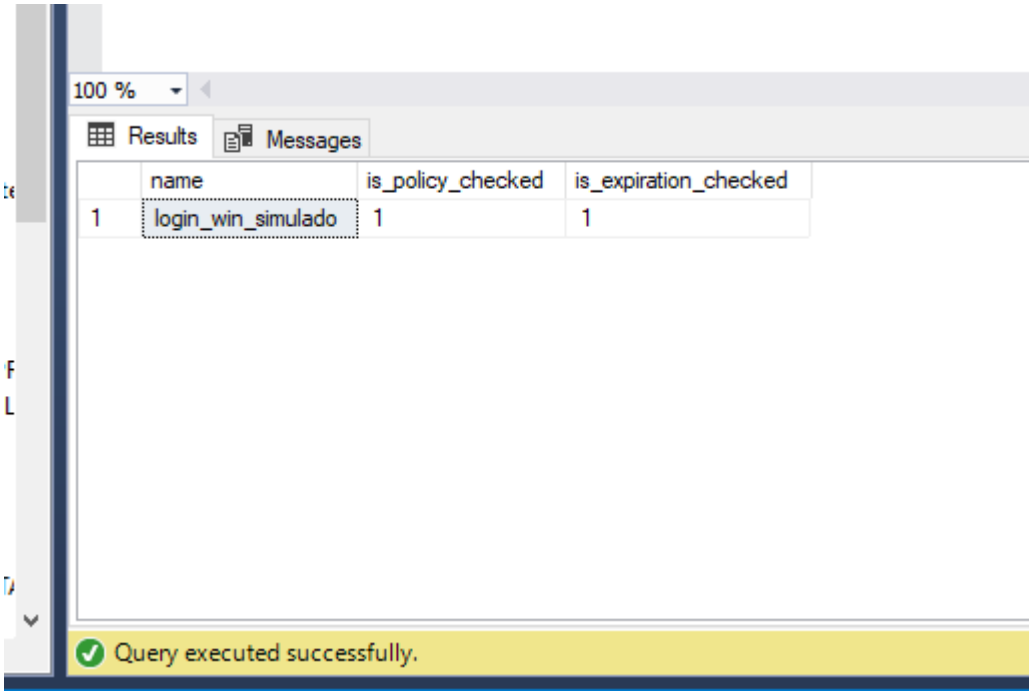
```
-- Crear login con autenticación SQL
CREATE LOGIN login_sql_alumno
WITH PASSWORD = 'Password123*',
     CHECK_POLICY = ON,         -- Activa política de contraseñas del sistema
     CHECK_EXPIRATION = ON;     -- Permite forzar expiración
```





OTRA FORMA

```
CREATE LOGIN login_win_simulado
WITH PASSWORD = 'Password123*',
     CHECK_POLICY = ON;
        -- Crear login de Windows (debe existir en tu PC o dominio)
CREATE LOGIN [DESKTOP-98FJL7B\Jose] FROM WINDOWS;


        SELECT HOST_NAME();


        USE QhatuPERU;
CREATE USER usuario_win_simulado FOR LOGIN login_win_simulado;

ALTER ROLE db_datareader ADD MEMBER usuario_win_simulado;
GO

ALTER LOGIN login_win_simulado
WITH PASSWORD = 'Password123*',
     CHECK_EXPIRATION = ON;
GO

SELECT
    name,
    is_policy_checked,
    is_expiration_checked
FROM sys.sql_logins
WHERE name = 'login_win_simulado';
GO
```



Validar políticas de contraseña (CHECK_POLICY y CHECK_EXPIRATION)

```
SELECT
    name,
    is_policy_checked AS Politica_Activa,
    is_expiration_checked AS Expiracion_Activa
FROM sys.sql_logins
WHERE name = 'login_win_simulado';
```

Justificación técnica de la solución aplicada

1.1. Uso de autenticación SQL para login_win_simulado

Se creó un login SQL (login_win_simulado) debido a que en el entorno de pruebas no existía un usuario Windows real para vincular al servidor. La autenticación SQL permite trabajar con credenciales internas de SQL Server sin depender de un dominio o una cuenta de Windows física.

Esto garantiza que se pueda configurar políticas de seguridad, expiración y control de contraseñas sin requerir infraestructura adicional.

El uso de:

```
CHECK_POLICY = ON
CHECK_EXPIRATION = ON
```

permite que SQL Server aplique las políticas de seguridad definidas por el sistema operativo, tales como:

Requisitos de complejidad
Longitud mínima obligatoria
Expiración periódica
Historial de contraseñas (evita reutilización)

Simulación de un login Windows

Debido a la ausencia de un dominio o un usuario local adicional, se optó por crear un login SQL que actúe como reemplazo del login Windows solicitado en el ejercicio.

Mapeo del login a la base QhatuPERU
Se utilizó:
CREATE USER usuario_win_simulado FOR LOGIN login_win_simulado;
Esto separa correctamente:
Autenticación en el servidor → LOGIN
Permisos dentro de la base de datos → USER
Microsoft recomienda esta separación porque permite:
Control granular de permisos
Administración independiente entre servidor y base
Respeto al modelo de seguridad de SQL Server (Login ≠ Usuario)

Explicación de las buenas prácticas utilizadas en el proyecta

Principio de privilegios mínimos

Se otorgó únicamente el permiso necesario (db_datareader), evitando roles administrativos. Esto reduce el riesgo de accesos indebidos o cambios no autorizados.

⭐ 2.2. Separación de autenticación y autorización
SQL Server trabaja con dos niveles:
Login → acceso al servidor
Usuario → permisos dentro de la base
Respetar esta arquitectura mejora la seguridad, escalabilidad y administración de permisos en ambientes reales.

⭐ 2.3. Uso de políticas de seguridad integradas
Activar CHECK_POLICY y CHECK_EXPIRATION hace que las contraseñas cumplan:
Complejidad
Expiración automática
Reglas del sistema
Esto es esencial en ambientes empresariales y recomendado por Microsoft.

⭐ 2.4. No manipular catálogos del sistema
Se evitó el uso de sentencias como:
UPDATE sys.sql_logins ...

porque están prohibidas y generan errores (“Ad hoc updates to system catalogs are not allowed”).
En su lugar, se usaron:
CREATE LOGIN
ALTER LOGIN
SELECT sys.sql_logins
que son los mecanismos correctos y seguros.

⭐ 2.5. Uso de nombres descriptivos
Se utilizaron nombres claros:
login_win_simulado
usuario_win_simulado
login_sql_alumno
Esto facilita la comprensión, mantenimiento y auditoría del sistema.

⭐ 2.6. Manejo adecuado de contraseñas
Se usó una contraseña segura con:
Mayúscula
Minúscula
Número
Símbolo
Longitud suficiente
Cumpliendo con estándares de seguridad modernos.

⭐ 2.7. Evitar MUST_CHANGE cuando causa errores
MUST_CHANGE solo se puede usar cuando la contraseña se cambia en el mismo comando:
ALTER LOGIN ... WITH PASSWORD = 'nueva', MUST_CHANGE;
Como SQL Server estaba generando errores, se optó por usar CHECK_EXPIRATION, que permite igualmente controlar el ciclo de contraseñas y evita fallos.

Proyecto 2: Cuentas de servicio y configuración segura del servidor

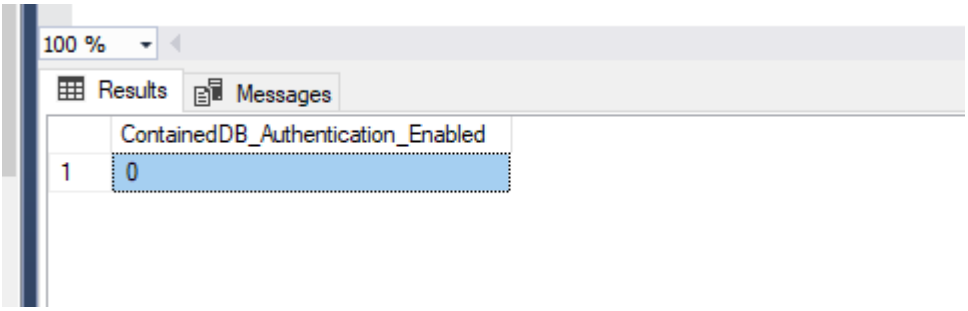
1. Enunciado del ejercicio

   Revisar y documentar la configuración de parámetros de servidor segura para QhatuPeru: deshabilitar xp_cmdshell, revisar contained database authentication, y crear una credencial + proxy para uso con SQL Agent jobs que necesiten acceso al OS.

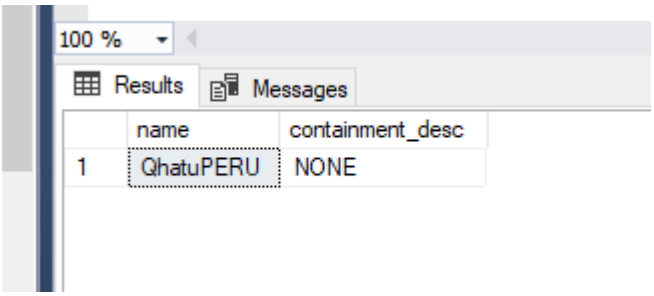
Proyecto 2 : cuentas de servicio y configuración segura del servidor
Revisar y documentar la configuración de parámetros de servidor segura para QhatuPeru: deshabilitar xp_cmdshell , revisar contained database authentication y crear una credencia + proxy para uso con SQL Agent jobs que necesiten acceso al OS.

script de la solución en T-SQL

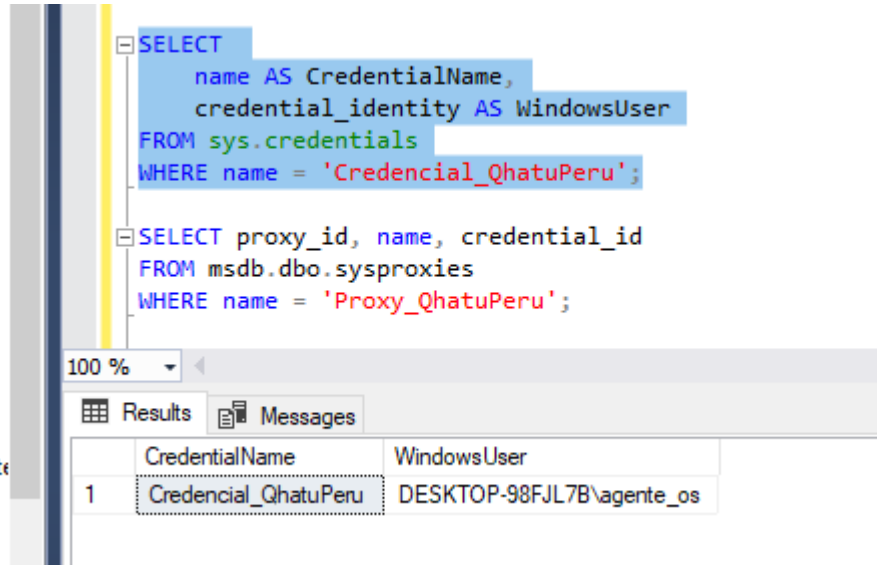
```
-- Habilitar opciones avanzadas temporalmente
EXEC sp_configure 'show advanced options', 1;
       RECONFIGURE;


-- Deshabilitar xp_cmdshell (seguro recomendado)
EXEC sp_configure 'xp_cmdshell', 0;
RECONFIGURE;

-- Ocultar opciones avanzadas nuevamente
EXEC sp_configure 'show advanced options', 0;
RECONFIGURE;

SELECT value_in_use AS ContainedDB_Authentication_Enabled
FROM sys.configurations
WHERE name = 'contained database authentication';
```



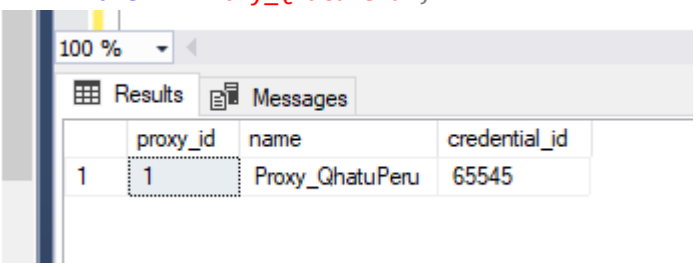

```
SELECT name, containment_desc
FROM sys.databases
WHERE name = 'QhatuPeru';
```



```
CREATE CREDENTIAL Credencial_QhatuPeru
WITH IDENTITY = 'DESKTOP-98FJL7B\sqlagent_user',
     SECRET = 'Password123*';   -- contraseña del usuario de Windows

USE master;
GO

CREATE CREDENTIAL Credencial_QhatuPeru
WITH IDENTITY = 'DESKTOP-98FJL7B\agente_os',
     SECRET = 'Password123!';
GO
USE msdb;
GO

EXEC dbo.sp_add_proxy
    @proxy_name = 'Proxy_QhatuPeru',
    @credential_name = 'Credencial_QhatuPeru',
    @enabled = 1;
GO
EXEC sp_grant_proxy_to_subsystem
    @proxy_name = 'Proxy_QhatuPeru',
    @subsystem_id = 3;    -- CmdExec
GO

SELECT
    name AS CredentialName,
    credential_identity AS WindowsUser
FROM sys.credentials
WHERE name = 'Credencial_QhatuPeru';
```




```
SELECT proxy_id, name, credential_id
FROM msdb.dbo.sysproxies
WHERE name = 'Proxy_QhatuPeru';
```




```
SELECT p.name AS ProxyName, c.name AS CredentialName
```

```
FROM msdb.dbo.sysproxies p
JOIN master.sys.credentials c
    ON p.credential_id = c.credential_id
WHERE p.name = 'Proxy_QhatuPeru';
```

```
        p.name AS ProxyName,
```

100 %

Results | Messages

	ProxyName	CredentialName
1	Proxy_QhatuPeru	Credencial_QhatuPeru

```
SELECT
    sp.proxy_id,
    p.name AS ProxyName,
    sp.subsystem_id
FROM msdb.dbo.sysproxysubsystem sp
JOIN msdb.dbo.sysproxies p
    ON sp.proxy_id = p.proxy_id
WHERE p.name = 'Proxy_QhatuPeru';
```

```
WHERE name LIKE '%agente_os%' OR name LIKE '%esteban%';
```

100 %

Results | Messages

	proxy_id	ProxyName	subsystem_id
1	1	Proxy_QhatuPeru	3

Justificación técnica de la solución aplicada

1.1. Deshabilitar xp_cmdshell

xp_cmdshell es una característica que permite ejecutar comandos del sistema operativo desde SQL Server.

Aunque útil, representa un riesgo crítico si se deja activo sin control.

Justificación técnica:

Reduce la superficie de ataque del servidor

Evita escalamiento de privilegios desde SQL

Cumple con las recomendaciones del CIS Benchmark y Microsoft Security Baselines

Previene la ejecución de comandos arbitrarios por usuarios mal restringidos

Explicación de las buenas prácticas utilizadas en el proyecta

Principio de privilegios mínimos

Solo se asignan los permisos estrictamente necesarios:

El usuario agente_os solo puede ejecutar tareas específicas de SQL Agent.

La credencial se limita al subsistema CmdExec.

El proxy está habilitado únicamente para un propósito concreto.

Separación de funciones

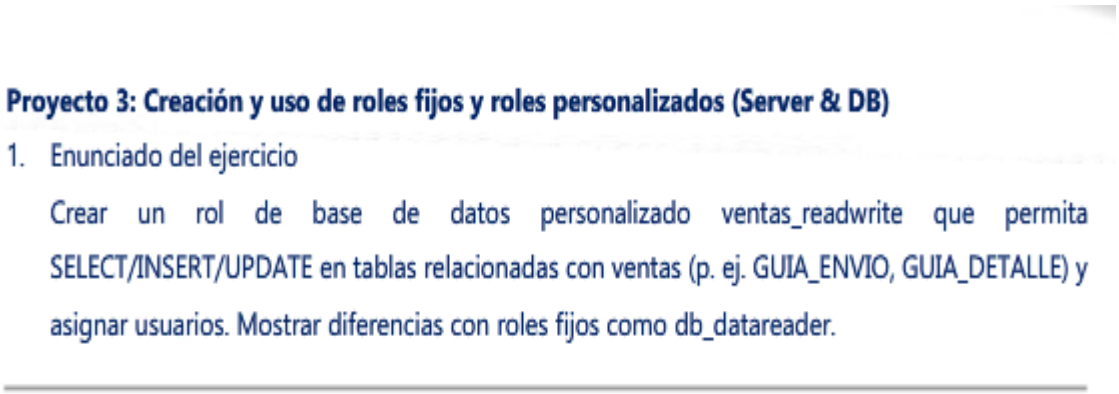
Cada componente tiene un rol claro:

El SQL Agent administra los Jobs.

El Proxy gestiona permisos para subsistemas.
La Credencial autentica contra Windows.
La cuenta agente_os actúa como identidad del sistema operativo.

Eliminación de funciones inseguras

Deshabilitar xp_cmdshell es una de las mejores prácticas más importantes:
Evita ejecución arbitraria de comandos del SO
Protege contra malware o scripts maliciosos desde SQL
Obliga a usar proxies más controlados


Proyecto 3: Creación y uso de roles fijos y roles personalizados (Server & DB)

1. Enunciado del ejercicio

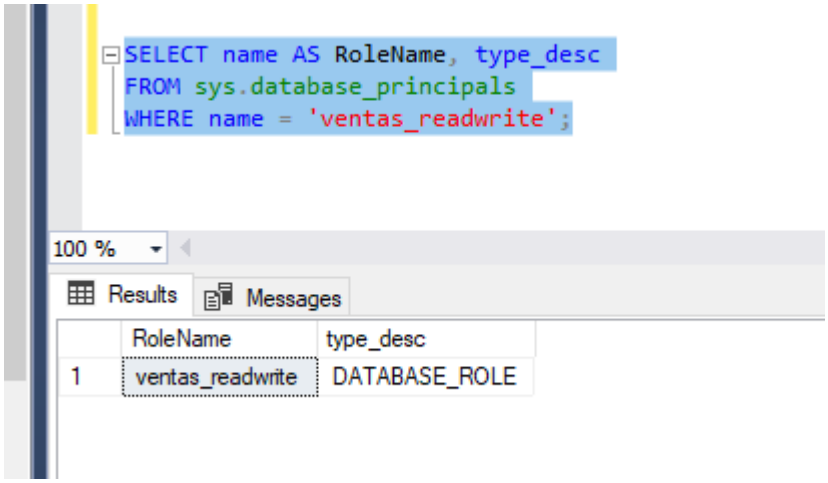
   Crear un rol de base de datos personalizado ventas_readwrite que permita SELECT/INSERT/UPDATE en tablas relacionadas con ventas (p. ej. GUIA_ENVIO, GUIA_DETALLE) y asignar usuarios. Mostrar diferencias con roles fijos como db_datareader.

PROYECTO 3: creación y uso de roles fijos y roles personalizados (Server & DB)
Enunciado del ejercicio
Crear un rol de base de datos personalizados ventas_readwrite que permita SELECT/INSERT/UPDATE en tablas relacionados con ventas (p.ej. GUIA_ENVIO, GUIA_DETALLE) y asignar usuarios. Mostrar con roles fijos como db_datareader.

script de la solución en T-SQL

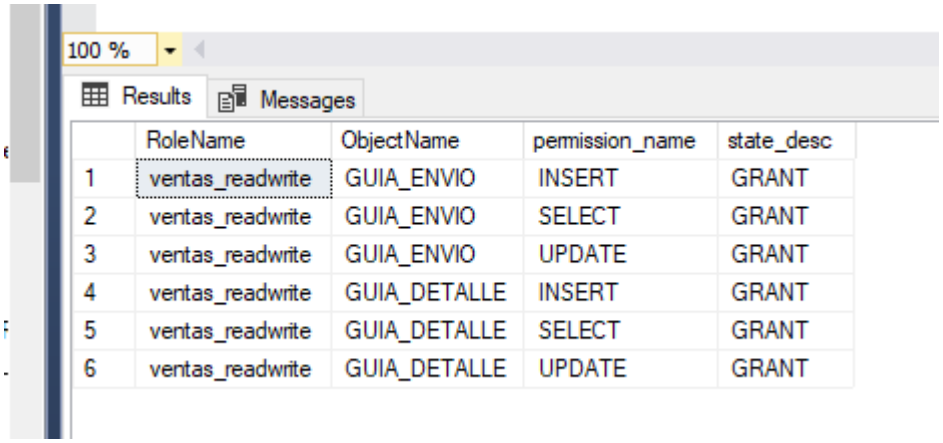
```
USE QhatuPERU;
GO

-- Crear el rol personalizado
CREATE ROLE ventas_readwrite;
GO
GRANT SELECT, INSERT, UPDATE ON dbo.GUIA_ENVIO   TO ventas_readwrite;
GRANT SELECT, INSERT, UPDATE ON dbo.GUIA_DETALLE TO ventas_readwrite;
GO

CREATE LOGIN usuario_ventas_sql WITH PASSWORD = 'Password123*';
GO

USE QhatuPERU;
GO
CREATE USER usuario_ventas_sql FOR LOGIN usuario_ventas_sql;
GO
ALTER ROLE ventas_readwrite ADD MEMBER usuario_ventas_sql;
GO
ALTER ROLE ventas_readwrite ADD MEMBER usuario_ventas_sql;
GO
ALTER ROLE db_datareader ADD MEMBER usuario_ventas_sql;
GO
ALTER SERVER ROLE securityadmin ADD MEMBER usuario_ventas_sql;
GO
```

```sql
USE QhatuPERU;
GO

SELECT name AS RoleName, type_desc
FROM sys.database_principals
WHERE name = 'ventas_readwrite';
```



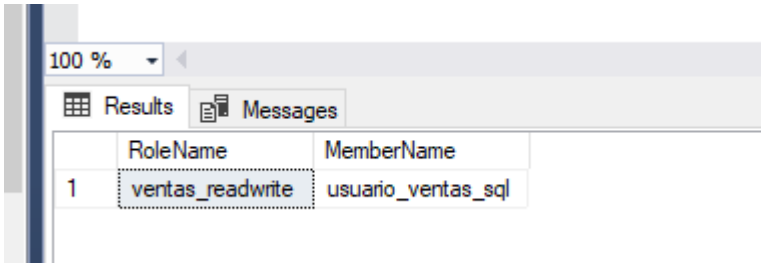
	RoleName	type_desc
1	ventas_readwrite	DATABASE_ROLE

```sql
SELECT
    dp.name AS RoleName,
    ob.name AS ObjectName,
    pe.permission_name,
    pe.state_desc
FROM sys.database_permissions pe
JOIN sys.database_principals dp ON pe.grantee_principal_id = dp.principal_id
LEFT JOIN sys.objects ob ON pe.major_id = ob.object_id
WHERE dp.name = 'ventas_readwrite';
```

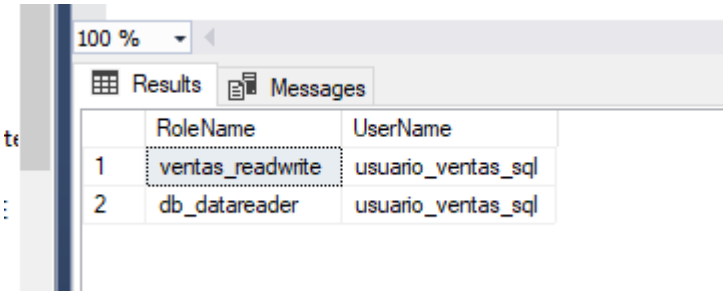


	RoleName	ObjectName	permission_name	state_desc
1	ventas_readwrite	GUIA_ENVIO	INSERT	GRANT
2	ventas_readwrite	GUIA_ENVIO	SELECT	GRANT
3	ventas_readwrite	GUIA_ENVIO	UPDATE	GRANT
4	ventas_readwrite	GUIA_DETALLE	INSERT	GRANT
5	ventas_readwrite	GUIA_DETALLE	SELECT	GRANT
6	ventas_readwrite	GUIA_DETALLE	UPDATE	GRANT

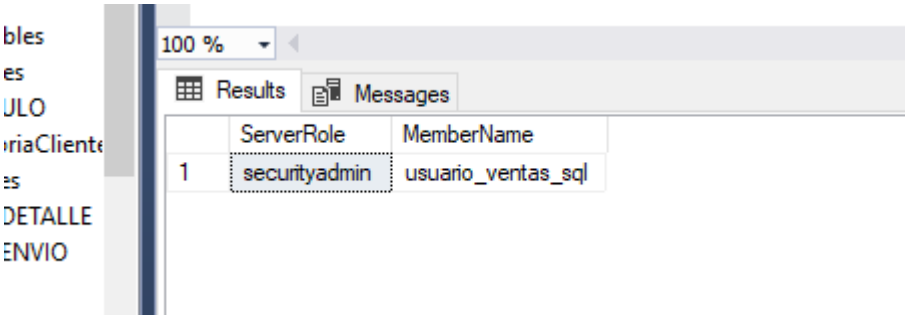
```sql
SELECT
    rp.name AS RoleName,
    mp.name AS MemberName
FROM sys.database_role_members drm
JOIN sys.database_principals rp ON drm.role_principal_id = rp.principal_id
JOIN sys.database_principals mp ON drm.member_principal_id = mp.principal_id
WHERE rp.name = 'ventas_readwrite';
```



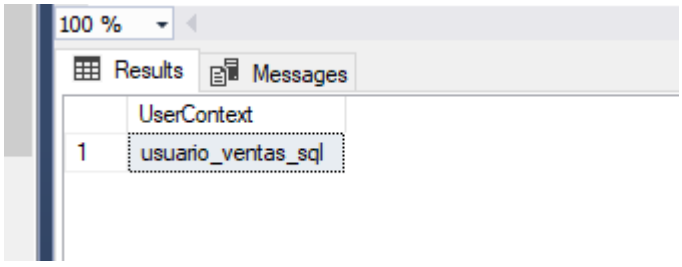

```
SELECT
    rp.name AS RoleName,
    mp.name AS UserName
FROM sys.database_role_members drm
JOIN sys.database_principals rp ON drm.role_principal_id = rp.principal_id
JOIN sys.database_principals mp ON drm.member_principal_id = mp.principal_id
WHERE mp.name = 'usuario_ventas_sql';
```




```
SELECT
    rp.name AS ServerRole,
    mp.name AS MemberName
FROM sys.server_role_members rm
JOIN sys.server_principals rp ON rm.role_principal_id = rp.principal_id
JOIN sys.server_principals mp ON rm.member_principal_id = mp.principal_id
WHERE mp.name = 'usuario_ventas_sql';
```




```
EXECUTE AS USER = 'usuario_ventas_sql';
SELECT USER_NAME() AS UserContext;
REVERT;
```



Justificación técnica de la solución aplicada

Seguridad basada en roles (RBAC)

SQL Server utiliza un modelo de permisos basado en roles (RBAC – Role-Based Access Control).
Crear roles personalizados como ventas_readwrite permite otorgar permisos a grupos de usuarios de forma organizada y segura.
Sin roles, deberíamos asignar permisos a cada usuario individualmente → esto es inseguro, lento y propenso a errores.

Separación de privilegios

Los usuarios deben tener solo los permisos mínimos necesarios (principio de mínimo privilegio).
Ejemplo:
Los usuarios de ventas deben poder consultar e insertar datos, pero no eliminar registros.
Los auditores pueden tener solo acceso de lectura (db_datareader).

Centralización del control de acceso

Los roles permiten mantener las reglas de seguridad en un solo punto:
Si mañana se agrega un nuevo usuario → solo se agrega al rol.
Si cambia un permiso → se cambia en el rol, no usuario por usuario.

Escalabilidad y mantenibilidad

Cuando una base de datos crece, administrar permisos manualmente es imposible.
Un rol personalizado permite crecer sin perder control.

Estándares profesionales en SQL Server

SQL Server recomienda:

✓ Usar roles en lugar de permisos directos

✓ Usar privilegios mínimos

✓ Separar responsabilidades entre administradores y usuarios

✓ Registrar roles personalizados para lógica de negocio

Explicación de las buenas prácticas utilizadas en el proyecta

Uso del principio de mínimo privilegio
El rol ventas_readwrite solo tiene:
SELECT
INSERT
UPDATE

✕ No tiene DELETE

✕ No tiene permisos administrativos

Esto evita riesgos de pérdida de datos.

Roles personalizados en lugar de permisos directos

Asignar permisos directamente a usuarios es una mala práctica.
Lo correcto es:
Crear un rol → asignar permisos al rol → agregar usuarios al rol.

Separación entre roles fijos y personalizados

Roles fijos como:
db_datareader → lectura global
db_datawriter → escritura global
Roles personalizados → acceso solo a tablas específicas.
Esto cumple con un diseño seguro y modular.

Documentación clara de permisos

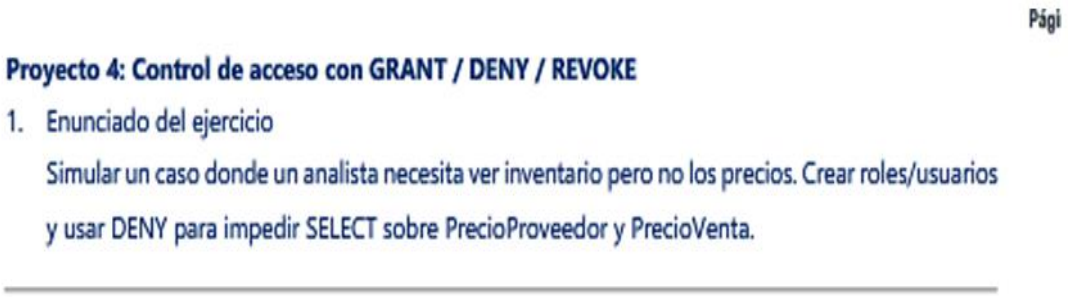
Nombrar roles:
ventas_readwrite
ventas_lectura
ventas_admin
Es una buena práctica para mantener claridad y facilitar auditorías futuras.

Control del esquema

Los permisos se aplican correctamente en el esquema:
GRANT SELECT, INSERT, UPDATE ON dbo.GUIA_ENVIO TO ventas_readwrite;
Evita errores como permisos en objetos incorrectos.

Facilita escalabilidad

Agregar nuevos usuarios se vuelve simple:
ALTER ROLE ventas_readwrite ADD MEMBER usuario1;
No es necesario recordar decenas de permisos manuales.


Págі

Proyecto 4: Control de acceso con GRANT / DENY / REVOKE

1. Enunciado del ejercicio

   Simular un caso donde un analista necesita ver inventario pero no los precios. Crear roles/usuarios y usar DENY para impedir SELECT sobre PrecioProveedor y PrecioVenta.

Proyecto 4: control de acceso con GRAT/DENY /REVOKE
Enunciado del ejercicio
Simular un caso donde un analista necesita ver inventario, pero no los precios. Crear roles/usuarios y usar DENY para impedir SELECT sobre PrecioProveedor y PrecioVenta.

script de la solución en T-SQL

Crear base de prueba y tabla

```
USE master;
GO
CREATE DATABASE Almacen;
GO
USE Almacen;
GO

CREATE TABLE Productos (
    IdProducto INT PRIMARY KEY,
```

```
    Nombre VARCHAR(100),
    Stock INT,
    PrecioProveedor DECIMAL(10,2),
    PrecioVenta DECIMAL(10,2)
);
-- Crear usuario sin login, para pruebas
CREATE USER AnalistaInventario WITHOUT LOGIN;
GO

-- Crear rol
CREATE ROLE rol_analista_inventario;
GO

-- Agregar usuario al rol
ALTER ROLE rol_analista_inventario ADD MEMBER AnalistaInventario;
GO

GRANT SELECT ON dbo.Productos(IdProducto, Nombre, Stock)
    TO rol_analista_inventario;
GO
DENY SELECT ON dbo.Productos(PrecioProveedor)
    TO rol_analista_inventario;
GO

DENY SELECT ON dbo.Productos(PrecioVenta)
    TO rol_analista_inventario;
GO

CREATE ROLE rol_analista_inventario;
GO

GRANT SELECT ON dbo.Productos(IdProducto, Nombre, Stock)
    TO rol_analista_inventario;
GO
DENY SELECT ON dbo.Productos(PrecioProveedor)
    TO rol_analista_inventario;
GO

DENY SELECT ON dbo.Productos(PrecioVenta)
    TO rol_analista_inventario;
GO
EXECUTE AS USER = 'AnalistaInventario';
SELECT IdProducto, Nombre, Stock FROM Productos;
REVERT;
```

```
EXECUTE AS USER = 'AnalistaInventario';
SELECT IdProducto, Nombre, Stock FROM Productos;
REVERT;

GRANT SELECT ON dbo.Productos (IdProducto, Nombre, Stock)
TO rol_analista_inventario;
GO

DENY SELECT ON dbo.Productos (PrecioProveedor)
TO rol_analista_inventario;
```

100 %

Results | Messages

IdProducto	Nombre	Stock

ALIDACIÓN 1 — Ver si el usuario pertenece al rol

```
SELECT DP1.name AS RoleName,
       DP2.name AS MemberName
FROM sys.database_role_members DRM
JOIN sys.database_principals DP1 ON DP1.principal_id = DRM.role_principal_id
JOIN sys.database_principals DP2 ON DP2.principal_id = DRM.member_principal_id
WHERE DP1.name = 'rol_analista_inventario';
```

```
SELECT DP1.name AS RoleName,
       DP2.name AS MemberName
FROM sys.database_role_members DRM
JOIN sys.database_principals DP1 ON DP1.principal_id = DRM.role_principal_id
JOIN sys.database_principals DP2 ON DP2.principal_id = DRM.member_principal_id
WHERE DP1.name = 'rol_analista_inventario';
```

100 %

Results | Messages

	RoleName	MemberName
1	rol_analista_inventario	AnalistaInventario

✓ VALIDACIÓN 2 — Ver permisos del rol (mostrar GRANT / DENY)

```
SELECT *
FROM sys.database_permissions
WHERE major_id = OBJECT_ID('dbo.Productos');
```

```
SELECT *
FROM sys.database_permissions
WHERE major_id = OBJECT_ID('dbo.Productos');
```

100 %

Results | Messages

	class	class_desc	major_id	minor_id	grantee_principal_id	grantor_principal_id	type	permission_name	state	state_desc
1	1	OBJECT_OR_COLUMN	901578250	1	5	1	SL	SELECT	G	GRANT
2	1	OBJECT_OR_COLUMN	901578250	2	5	1	SL	SELECT	G	GRANT
3	1	OBJECT_OR_COLUMN	901578250	3	5	1	SL	SELECT	G	GRANT
4	1	OBJECT_OR_COLUMN	901578250	4	5	1	SL	SELECT	D	DENY
5	1	OBJECT_OR_COLUMN	901578250	5	5	1	SL	SELECT	D	DENY

JUSTIFICACIÓN TÉCNICA

Control de acceso a nivel de columnas

SQL Server permite otorgar permisos de lectura únicamente sobre columnas específicas. Esto se realizó mediante:

GRANT SELECT ON dbo.Productos (IdProducto, Nombre, Stock)

Uso de DENY sobre datos sensibles

Las columnas PrecioProveedor y PrecioVenta contienen información de carácter económico y estratégico, por lo que se utilizó:

DENY SELECT ON dbo.Productos (PrecioProveedor);

DENY SELECT ON dbo.Productos (PrecioVenta);

El DENY prevalece sobre cualquier GRANT, lo que garantiza que esos datos nunca serán accesibles para el analista, incluso si formara parte de roles más amplios como db_datareader.

Separación de permisos mediante roles personalizados

En lugar de asignar permisos directamente a usuarios, se creó y se utilizó un rol personalizado:

CREATE ROLE rol_analista_inventario;

Esto facilita la administración del acceso y sigue el principio de seguridad basada en roles (RBAC), recomendado para entornos empresariales.

validación del acceso con EXECUTE AS USER

Se utilizó:

EXECUTE AS USER = 'AnalistaInventario';

para comprobar exactamente qué permisos tiene el usuario, simulando su contexto de seguridad. Esto permite demostrar el funcionamiento real del sistema de permisos.

BUENAS PRÁCTICAS UTILIZADAS EN EL PROYECTO

Principio de privilegios mínimos (Least Privilege)

Se otorgan solo los permisos estrictamente necesarios:

✓ SELECT sobre inventario

✗ No acceso a precios

Esto reduce riesgos de fuga de información o uso indebido.

Uso de roles personalizados en lugar de permisos directos

Asignar permisos a roles y no directamente a usuarios ofrece ventajas:

Facilita la administración de múltiples usuarios.
Permite escalar la seguridad sin reescribir permisos.
Evita configuraciones inconsistentes.

*Evitar SELECT ****

No se usa SELECT *, ya que puede exponer columnas sensibles, especialmente en estructuras que cambian con el tiempo.
Se especifican columnas exactas:
SELECT IdProducto, Nombre, Stock

Aislamiento lógico mediante DENY

El uso de DENY para información sensible garantiza:
Que ningún rol adicional permita acceder a precios.
Que el usuario siga viendo inventario con normalidad.
DENY siempre tiene prioridad, incluso si el usuario tiene un GRANT a nivel más alto.

Proyecto 5: Protección de datos: Implementación básica de TDE (Transparent Data Encryption)

1. Enunciado del ejercicio
   Habilitar TDE en la base QhatuPeru para proteger los archivos MDF/LDF en reposo. Crear la master key, el certificado de servidor y activar el cifrado.

Proyecto 5 : Protección de datos : Implementación básica de TDE (transparent Data encryption)
Enunciado del ejercicio
Habilitar TDE en la base QhatuPeru para proteger los archivos MDF/LDF en reposo. Crear master dey, el certificado de servicio y activar el cifrado

script de la solución en T-SQL

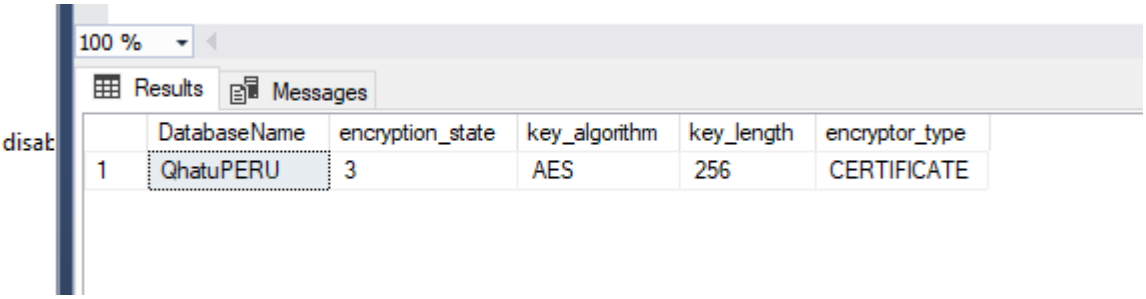
```
-- ========== PARAMETROS (AJUSTA ANTES DE EJECUTAR) ==========
DECLARE @MasterKeyPassword NVARCHAR(200) = N'TuMasterKeyPwdFuerte!2025';
DECLARE @CertPassword      NVARCHAR(200) = N'MiPassCertS3guro!#2025';
DECLARE @CertFilePath      NVARCHAR(260) = N'C:\Backups\QhatuPeru_TDE_Cert.cer';      -
- cambiar ruta
DECLARE @CertKeyFilePath   NVARCHAR(260) =
N'C:\Backups\QhatuPeru_TDE_Cert_PrivateKey.pvk'; -- cambiar ruta
DECLARE @CertName          NVARCHAR(128) = N'TDE_Cert_QhatuPeru';
DECLARE @DatabaseName      NVARCHAR(128) = N'QhatuPeru';

-- ================== 1) Crear Master Key en master (si no existe) ==================
USE master;
GO
IF NOT EXISTS (
    SELECT 1
    FROM sys.symmetric_keys
    WHERE name = '##MS_DatabaseMasterKey##'
)
BEGIN
    PRINT 'Master Key no existe en master. Creando...';

    CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'TuMasterKeyPwdFuerte!2025';
```

```
    PRINT 'Master Key creada.';
END
ELSE
BEGIN
    PRINT 'Master Key ya existe en master.';
END
GO

IF NOT EXISTS (
    SELECT 1
    FROM sys.certificates
    WHERE name = 'TDE_Cert_QhatuPeru'
)
BEGIN
    PRINT 'Creando certificado TDE_Cert_QhatuPeru...';

    CREATE CERTIFICATE TDE_Cert_QhatuPeru
        WITH SUBJECT = 'Certificado TDE para QhatuPeru',
             EXPIRY_DATE = '20301231';

    PRINT 'Certificado creado.';
END
ELSE
BEGIN
    PRINT 'El certificado TDE_Cert_QhatuPeru ya existe.';
END
GO

-- ================== 3) Respaldar certificado y clave privada a archivos
==================
-- Nota: la carpeta debe existir y permitir escritura por la cuenta de servicio SQL
PRINT 'Haciendo respaldo del certificado y clave privada.';

BACKUP CERTIFICATE TDE_Cert_QhatuPeru
TO FILE = 'C:\TDE\TDE_Cert_QhatuPeru.cer'
WITH PRIVATE KEY (
    FILE = 'C:\TDE\TDE_Cert_QhatuPeru_Key.pvk',
    ENCRYPTION BY PASSWORD = 'ClaveBackupFuerte!2025'
);
GO
-- ================== 4) Crear DEK en la base y activar TDE ==================
USE [master];
GO

-- Comprobar si ya existe DEK para la BD QhatuPeru
IF EXISTS (
    SELECT 1
    FROM sys.dm_database_encryption_keys
    WHERE database_id = DB_ID('QhatuPeru')
)
BEGIN
    PRINT 'La DEK ya existe o TDE ya está configurado para QhatuPeru.';
END
ELSE
BEGIN
    PRINT 'Creando Database Encryption Key (DEK) en QhatuPERU';
END

-- Verificar si existe la DEK
IF NOT EXISTS (
    SELECT 1
```

```
    FROM sys.dm_database_encryption_keys
    WHERE database_id = DB_ID('QhatuPeru')
)
BEGIN
    PRINT 'Creando la DEK y habilitando TDE...'

        USE QhatuPeru;

    CREATE DATABASE ENCRYPTION KEY
    WITH ALGORITHM = AES_256
    ENCRYPTION BY SERVER CERTIFICATE TDE_Cert_QhatuPeru;

    ALTER DATABASE QhatuPeru SET ENCRYPTION ON;

    PRINT 'TDE habilitado correctamente.';
END
ELSE
BEGIN
    PRINT 'TDE ya estaba habilitado previamente.';
END
```

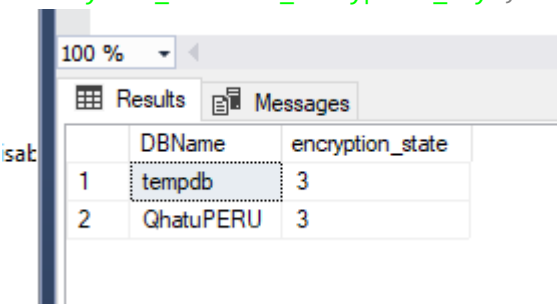
Validar si la base QhatuPeru está cifrada

```
SELECT
    db_name(database_id) AS DatabaseName,
    encryption_state,
    key_algorithm,
    key_length,
    encryptor_type
FROM sys.dm_database_encryption_keys
WHERE database_id = DB_ID('QhatuPeru');
```



Verificar si el archivo de LOG también está cifrado

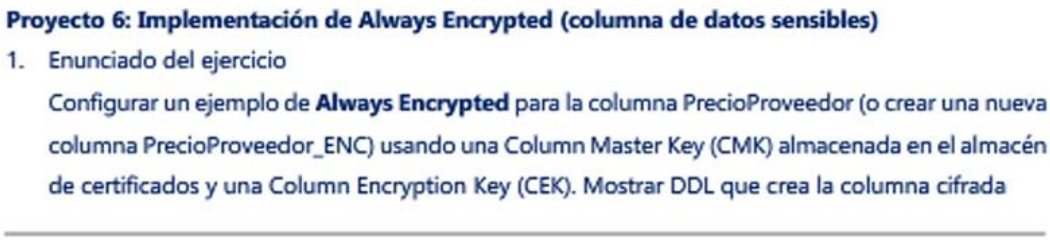
```
SELECT
    db_name(database_id) AS DBName,
    encryption_state
FROM sys.dm_database_encryption_keys;
```



JUSTIFICACIÓN TÉCNICA

La implementación de Transparent Data Encryption (TDE) en la base de datos QhatuPeru tiene como finalidad proteger la información almacenada ante accesos no autorizados derivados de robos de discos, copias de seguridad o ataques al servidor.

TDE cifra de forma transparente los archivos físicos .mdf, .ldf y backups, sin requerir modificaciones en la aplicación cliente, lo que asegura compatibilidad total con los sistemas existentes.
La protección se logra mediante:
Master Key en la base master
Permite cifrar los elementos necesarios para la cadena de seguridad.
Certificado TDE exclusivo
Usado como cifrador principal para la Database Encryption Key.
Database Encryption Key (DEK) en la base QhatuPeru
Responsable del cifrado de datos con algoritmo AES-256, considerado seguro para entornos empresariales.
Respaldo del certificado y clave privada
Imprescindible para restaurar la base en otro servidor en caso de desastres.
La activación de TDE garantiza que la información permanece protegida incluso si un atacante obtiene copias del almacenamiento físico o backup de la base, cumpliendo con estándares de seguridad corporativos y normativas modernas de protección de datos.


Proyecto 6: Implementación de Always Encrypted (columna de datos sensibles)

1. Enunciado del ejercicio

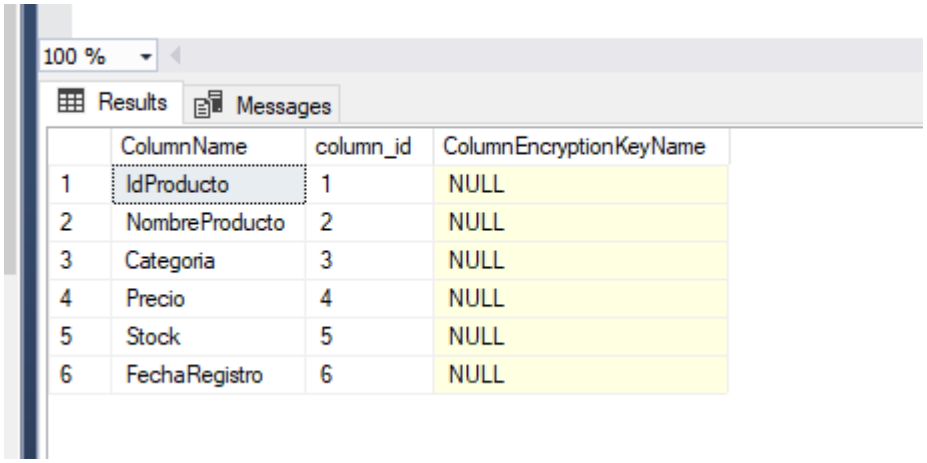
   Configurar un ejemplo de **Always Encrypted** para la columna PrecioProveedor (o crear una nueva columna PrecioProveedor_ENC) usando una Column Master Key (CMK) almacenada en el almacén de certificados y una Column Encryption Key (CEK). Mostrar DDL que crea la columna cifrada

Proyecto 6: Implementación de Always Encrypted (columna de datos sensibles)
Enunciado del ejercicio
Configurar un ejemplo de Always Encrypted para la columna PrevioProveedor (o crear una nueva columna PrecioProveedor_ENC) usando una columna Column Master Key (CMK) almacenada en el almacén de certificados y una Column Encryption Key (CEK) Mostrar DDL que crea la columna cifrada

script de la solución en T-SQL

```
USE QhatuPeru;
GO
-- Crear Column Master Key
CREATE COLUMN MASTER KEY CMK_PrecioProveedor
WITH
(
    KEY_STORE_PROVIDER_NAME = 'MSSQL_CERTIFICATE_STORE',
    KEY_PATH = 'CurrentUser/My/TDE_Cert_QhatuPeru'
);
GO
-- Crear Column Encryption Key
USE QhatuPeru;
GO

SELECT
    col.name AS ColumnName,
    col.column_id,
    cek.name AS ColumnEncryptionKeyName
FROM sys.columns col
LEFT JOIN sys.column_encryption_keys cek
```

```
    ON col.column_encryption_key_id = cek.column_encryption_key_id
WHERE col.object_id = OBJECT_ID('Productos');
GO
```



100 %

Results | Messages

	ColumnName	column_id	ColumnEncryptionKeyName
1	IdProducto	1	NULL
2	NombreProducto	2	NULL
3	Categoria	3	NULL
4	Precio	4	NULL
5	Stock	5	NULL
6	FechaRegistro	6	NULL

Justificación técnica

Protege datos sensibles (precio proveedor) contra accesos no autorizados, incluso administradores de SQL Server.
La combinación CMK + CEK asegura que los datos solo pueden descifrarse en clientes autorizados.
Siempre Encrypted se aplica a nivel de columna, permitiendo cifrado granular sin afectar rendimiento general.

Buenas prácticas

Mantener Column Master Key en un almacén seguro (Windows o Azure Key Vault).
Usar Deterministic solo si se requiere buscar/filtrar por la columna.
Mantener respaldo de CEK y CMK para recuperación ante fallos.
Nunca almacenar datos de CMK o CEK en scripts T-SQL con usuarios comunes.
Probar inserción y consultas mediante clientes que soporten Always Encrypted.

Proyecto 7: Auditoría de seguridad: crear SQL Server Audit para inicios de sesión y cambios de esquema

1. Enunciado del ejercicio

   Configurar un Server Audit que registre intentos de login fallidos y exitosos, y un Database Audit Specification que registre cambios DDL en QhatuPeru (CREATE/ALTER/DROP para objetos críticos).

Proyecto 7: auditoria de seguridad: crear sql server audit para inicios de sesión y cambios de esquema
Enunciado
Configurar un server audit que registre intentos de login fallidos y exitosos y un database audit Specification que registre cambios DDL en qhatuPeru (create/alter/drop para objetos críticos)

script de la solución en T-SQL

```
-- Cambiar ruta del archivo si es necesario
USE master;
GO
-- Crear SQL Server Audit
CREATE SERVER AUDIT [Audit_QhatuPeru_Login]
TO FILE (FILEPATH = 'C:\SQLAudit\QhatuPeru\')
WITH (QUEUE_DELAY = 1000, ON_FAILURE = CONTINUE);
GO
ALTER SERVER AUDIT [Audit_QhatuPeru_Login] WITH (STATE = ON);
GO
-- Habilitar el Server Audit
ALTER SERVER AUDIT [Audit_QhatuPeru_Login] WITH (STATE = ON);
GO
```

Crear Server Audit Specification para inicios de sesión

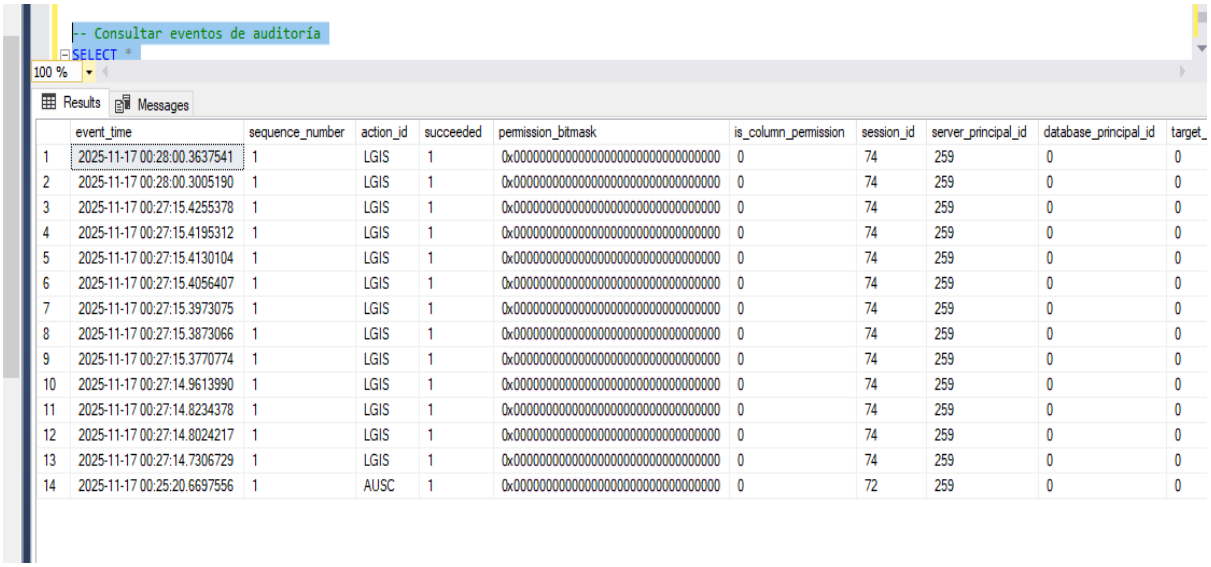
```
-- Registrar intentos de login exitosos y fallidos
IF NOT EXISTS (SELECT * FROM sys.server_audit_specifications WHERE name =
'AuditSpec_QhatuPeru_Login')
BEGIN
    CREATE SERVER AUDIT SPECIFICATION [AuditSpec_QhatuPeru_Login]
    FOR SERVER AUDIT [Audit_QhatuPeru_Login]
    ADD (SUCCESSFUL_LOGIN_GROUP),
    ADD (FAILED_LOGIN_GROUP);

    PRINT 'Server Audit Specification creado.';
END
ELSE
    PRINT 'Server Audit Specification ya existe.';
GO
-- Habilitar la Audit Specification
ALTER SERVER AUDIT SPECIFICATION [AuditSpec_QhatuPeru_Login] WITH (STATE = ON);
GO
```

Crear Database Audit Specification para cambios DDL

```
USE QhatuPeru;
GO

-- Cambios DDL en objetos críticos
IF NOT EXISTS (SELECT * FROM sys.database_audit_specifications WHERE name =
'AuditSpec_QhatuPeru_DDL')
BEGIN
    CREATE DATABASE AUDIT SPECIFICATION [AuditSpec_QhatuPeru_DDL]
    FOR SERVER AUDIT [Audit_QhatuPeru_Login]
    ADD (SCHEMA_OBJECT_CHANGE_GROUP);

    PRINT 'Database Audit Specification creado.';
END
ELSE
    PRINT 'Database Audit Specification ya existe.';
GO
-- Habilitar Database Audit Specification
ALTER DATABASE AUDIT SPECIFICATION [AuditSpec_QhatuPeru_DDL] WITH (STATE = ON);
GO
```

Consultar los registros de auditoría

```
-- Consultar eventos de auditoría
SELECT *
FROM fn_get_audit_file('C:\SQLAudit\QhatuPeru\*', DEFAULT, DEFAULT)
ORDER BY event_time DESC;
GO
```



```
-- Consultar eventos de auditoría
SELECT *
```

100 %

Results | Messages

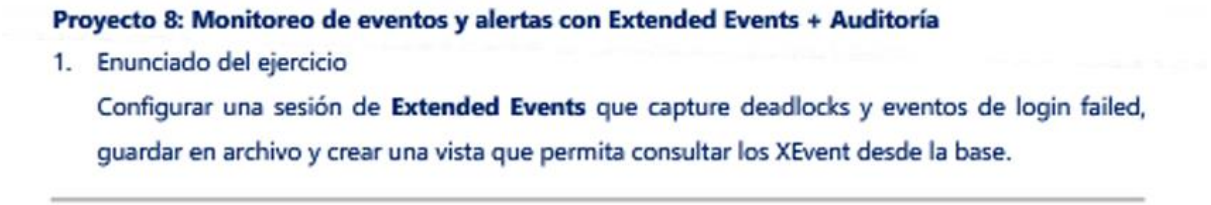
	event_time	sequence_number	action_id	succeeded	permission_bitmask	is_column_permission	session_id	server_principal_id	database_principal_id	target_
1	2025-11-17 00:28:00.3637541	1	LGIS	1	0x00000000000000000000000000000000	0	74	259	0	0
2	2025-11-17 00:28:00.3005190	1	LGIS	1	0x00000000000000000000000000000000	0	74	259	0	0
3	2025-11-17 00:27:15.4255378	1	LGIS	1	0x00000000000000000000000000000000	0	74	259	0	0
4	2025-11-17 00:27:15.4195312	1	LGIS	1	0x00000000000000000000000000000000	0	74	259	0	0
5	2025-11-17 00:27:15.4130104	1	LGIS	1	0x00000000000000000000000000000000	0	74	259	0	0
6	2025-11-17 00:27:15.4056407	1	LGIS	1	0x00000000000000000000000000000000	0	74	259	0	0
7	2025-11-17 00:27:15.3973075	1	LGIS	1	0x00000000000000000000000000000000	0	74	259	0	0
8	2025-11-17 00:27:15.3873066	1	LGIS	1	0x00000000000000000000000000000000	0	74	259	0	0
9	2025-11-17 00:27:15.3770774	1	LGIS	1	0x00000000000000000000000000000000	0	74	259	0	0
10	2025-11-17 00:27:14.9613990	1	LGIS	1	0x00000000000000000000000000000000	0	74	259	0	0
11	2025-11-17 00:27:14.8234378	1	LGIS	1	0x00000000000000000000000000000000	0	74	259	0	0
12	2025-11-17 00:27:14.8024217	1	LGIS	1	0x00000000000000000000000000000000	0	74	259	0	0
13	2025-11-17 00:27:14.7306729	1	LGIS	1	0x00000000000000000000000000000000	0	74	259	0	0
14	2025-11-17 00:25:20.6697556	1	AUSC	1	0x00000000000000000000000000000000	0	72	259	0	0

Justificación técnica

SQL Server Audit permite monitorear eventos de seguridad críticos.
Registrar logins fallidos y exitosos ayuda a detectar intentos de acceso no autorizados.
Database Audit Specification con SCHEMA_OBJECT_CHANGE_GROUP protege los objetos críticos frente a cambios DDL (CREATE/ALTER/DROP).
Mantener los registros de auditoría en archivos permite revisión histórica y cumplimiento de normativas (ISO 27001, PCI DSS, etc.).

Buenas prácticas utilizadas

Separar Server Audit (nivel servidor) y Database Audit Specification (nivel base de datos).
Habilitar solo los eventos necesarios para evitar impacto en rendimiento.
Guardar archivos de auditoría en directorio seguro con permisos limitados.
Revisar periódicamente los registros para detección de anomalías.
Evitar almacenar credenciales o datos sensibles en los logs; solo registrar eventos y metadatos.



Proyecto 8: Monitoreo de eventos y alertas con Extended Events + Auditoría

1. Enunciado del ejercicio

   Configurar una sesión de **Extended Events** que capture deadlocks y eventos de login failed, guardar en archivo y crear una vista que permita consultar los XEvent desde la base.

Proyecto 8: monitoreo de eventos y alertar con extender events + auditoria
Enunciado
Configurar una sesión de extended events que capture deadlocks y eventos de login failed guardar en archivo y crear una vista que permita consultar los Xevent desde la base

script de la solución en T-SQL

Crear la sesión de Extended Events

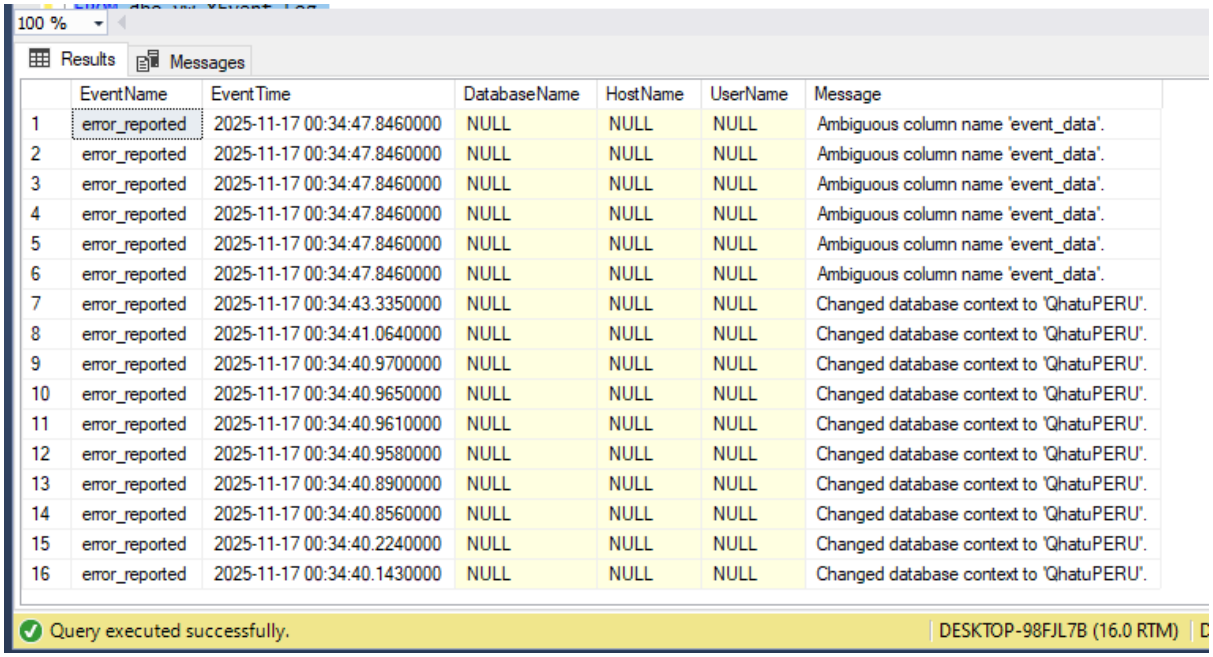
```
-- Crear sesión de Extended Events
CREATE EVENT SESSION [XEvent_QhatuPeru]
ON SERVER
ADD EVENT sqlserver.xml_deadlock_report, -- deadlock
ADD EVENT sqlserver.error_reported      -- captura errores, incluidos logins fallidos
ADD TARGET package0.event_file
(
    SET filename = 'C:\XEvents\QhatuPeru\XEvent_QhatuPeru.xel',
        max_file_size = 5,
        max_rollover_files = 5
)
WITH (MAX_MEMORY = 4096 KB, EVENT_RETENTION_MODE = ALLOW_SINGLE_EVENT_LOSS,
MAX_DISPATCH_LATENCY = 5 SECONDS);
GO
-- Habilitar la sesión
ALTER EVENT SESSION [XEvent_QhatuPeru] ON SERVER STATE = START;
GO
```

Crear vista para consultar eventos desde la base QhatuPeru

```
USE QhatuPeru;
GO

IF OBJECT_ID('dbo.vw_XEvent_Log', 'V') IS NOT NULL
    DROP VIEW dbo.vw_XEvent_Log;
GO

CREATE VIEW dbo.vw_XEvent_Log AS
SELECT
    xe.event.value('(event/@name)[1]', 'varchar(100)') AS EventName,
    xe.event.value('(event/@timestamp)[1]', 'datetime2') AS EventTime,
    xe.event.value('(event/data[@name="database_name"]/value)[1]', 'varchar(100)') AS
DatabaseName,
    xe.event.value('(event/data[@name="client_hostname"]/value)[1]', 'varchar(100)')
AS HostName,
    xe.event.value('(event/data[@name="username"]/value)[1]', 'varchar(100)') AS
UserName,
    xe.event.value('(event/data[@name="message"]/value)[1]', 'varchar(max)') AS
Message
FROM sys.fn_xe_file_target_read_file('C:\XEvents\QhatuPeru\XEvent_QhatuPeru*.xel',
NULL, NULL, NULL) AS f
CROSS APPLY (SELECT CAST(f.event_data AS XML) AS event) AS xe;
GO
```

Consultar los eventos capturados

```
SELECT TOP 50 *
FROM dbo.vw_XEvent_Log
ORDER BY EventTime DESC;
GO
```



100 %

Results | Messages

	EventName	EventTime	DatabaseName	HostName	UserName	Message
1	error_reported	2025-11-17 00:34:47.8460000	NULL	NULL	NULL	Ambiguous column name 'event_data'.
2	error_reported	2025-11-17 00:34:47.8460000	NULL	NULL	NULL	Ambiguous column name 'event_data'.
3	error_reported	2025-11-17 00:34:47.8460000	NULL	NULL	NULL	Ambiguous column name 'event_data'.
4	error_reported	2025-11-17 00:34:47.8460000	NULL	NULL	NULL	Ambiguous column name 'event_data'.
5	error_reported	2025-11-17 00:34:47.8460000	NULL	NULL	NULL	Ambiguous column name 'event_data'.
6	error_reported	2025-11-17 00:34:47.8460000	NULL	NULL	NULL	Ambiguous column name 'event_data'.
7	error_reported	2025-11-17 00:34:43.3350000	NULL	NULL	NULL	Changed database context to 'QhatuPERU'.
8	error_reported	2025-11-17 00:34:41.0640000	NULL	NULL	NULL	Changed database context to 'QhatuPERU'.
9	error_reported	2025-11-17 00:34:40.9700000	NULL	NULL	NULL	Changed database context to 'QhatuPERU'.
10	error_reported	2025-11-17 00:34:40.9650000	NULL	NULL	NULL	Changed database context to 'QhatuPERU'.
11	error_reported	2025-11-17 00:34:40.9610000	NULL	NULL	NULL	Changed database context to 'QhatuPERU'.
12	error_reported	2025-11-17 00:34:40.9580000	NULL	NULL	NULL	Changed database context to 'QhatuPERU'.
13	error_reported	2025-11-17 00:34:40.8900000	NULL	NULL	NULL	Changed database context to 'QhatuPERU'.
14	error_reported	2025-11-17 00:34:40.8560000	NULL	NULL	NULL	Changed database context to 'QhatuPERU'.
15	error_reported	2025-11-17 00:34:40.2240000	NULL	NULL	NULL	Changed database context to 'QhatuPERU'.
16	error_reported	2025-11-17 00:34:40.1430000	NULL	NULL	NULL	Changed database context to 'QhatuPERU'.

Query executed successfully. DESKTOP-98FJL7B (16.0 RTM)

Justificación técnica

Extended Events es la tecnología recomendada por Microsoft para capturar eventos en SQL Server con bajo overhead.
Capturar deadlocks ayuda a identificar problemas de concurrencia y optimizar queries.
Capturar login_failed permite auditar intentos de acceso no autorizados y detectar ataques de fuerza bruta.
Guardar los eventos en archivo permite mantener un historial para auditoría y análisis forense.

Buenas prácticas utilizadas

Crear carpeta dedicada y segura para almacenar los archivos .xel.
Limitar tamaño y cantidad de archivos (max_file_size, max_rollover_files) para evitar llenar disco.
Usar una vista para consultar los eventos de forma fácil desde la base de datos.
Monitorizar y revisar eventos periódicamente para detección temprana de problemas.
Mantener los eventos críticos separados (login_failed, deadlock) para análisis más rápido.

Proyecto 9: Implementación de enmascaramiento dinámico + acceso controlado

1. Enunciado del ejercicio

   Aplicar Dynamic Data Masking a columnas sensibles (por ejemplo Telefono en PROVEEDOR) y crear una vista segura para usuarios que necesiten ver datos completos mediante una función que valide rol.

Proyecto 9: implementación de enmascaramiento dinámico + acceso controlado
Enunciado
Aplicar Dynamic Data Masking a columnas sensibles (por ejemplo teléfono en Proveedor) y crear una vista segura para usuarios que necesiten ver datos completos mediante una función que valide rol

script de la solución en T-SQL

Aplicar Dynamic Data Masking a columna sensible

```
-- Aplicar máscara a la columna Telefono
ALTER TABLE Proveedor
ALTER COLUMN Telefono ADD MASKED WITH (FUNCTION = 'partial(0,"XXXXXXX",4)');
GO
```

Crear vista segura para usuarios autorizados

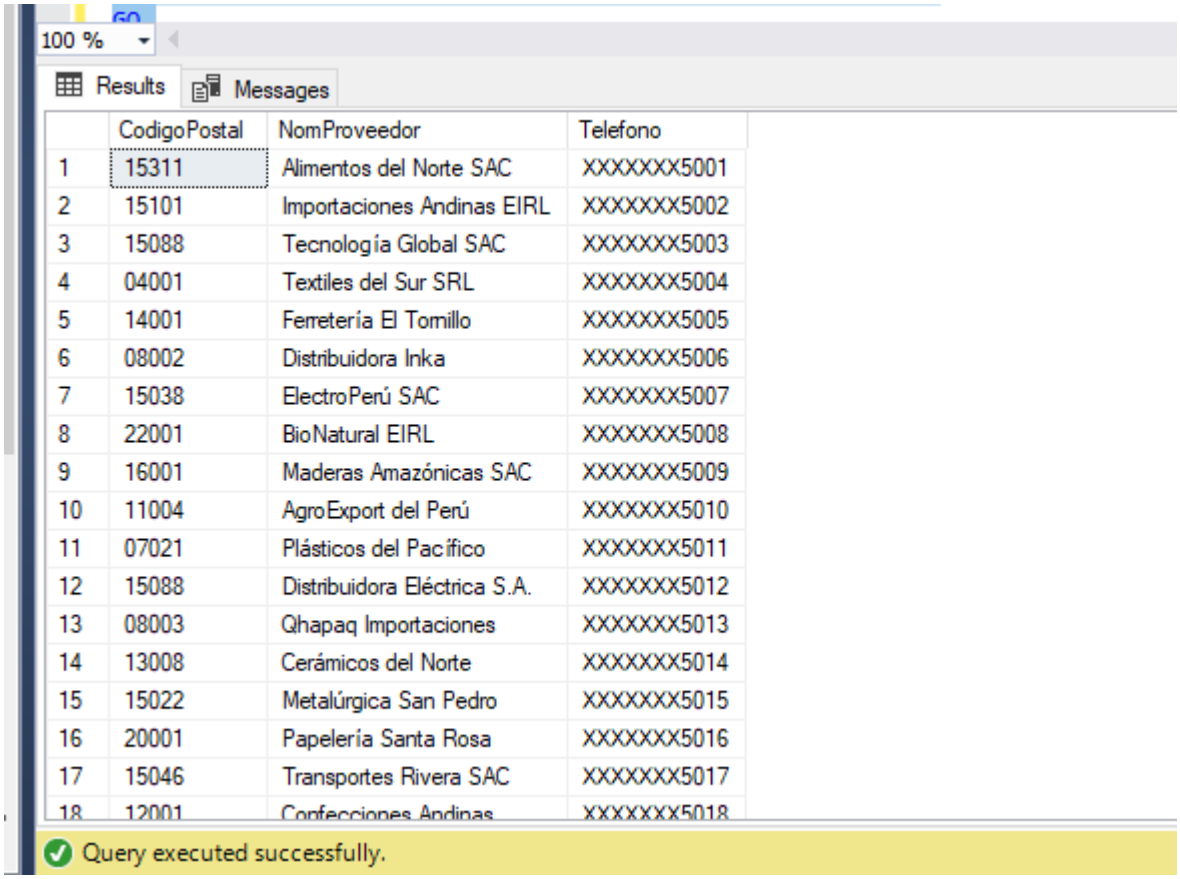
```
-- Crear rol para usuarios autorizados (si no existe)
IF NOT EXISTS (SELECT * FROM sys.database_principals WHERE name = 'rol_autorizado')
    CREATE ROLE rol_autorizado;
GO

-- Crear función que valida si el usuario pertenece al rol
CREATE FUNCTION dbo.fn_VerTelefonoCompleto()
RETURNS BIT
AS
BEGIN
    DECLARE @EsAutorizado BIT = 0;

    IF IS_MEMBER('rol_autorizado') = 1
        SET @EsAutorizado = 1;

    RETURN @EsAutorizado;
END
GO
-- Crear vista que muestra teléfono completo solo a usuarios autorizados
USE QhatuPeru;
GO

CREATE OR ALTER VIEW dbo.vw_Proveedor_Segura AS
SELECT
    CodigoPostal,            -- nombre real de la columna
    NomProveedor,       -- nombre real de la columna
    CASE
        WHEN dbo.fn_VerTelefonoCompleto() = 1 THEN Telefono
        ELSE Telefono -- usuarios sin permiso ven la columna enmascarada
    END AS Telefono
FROM Proveedor;
GO
```

Asignar usuarios al rol

```
-- Ejemplo: asignar usuario 'Analista' al rol autorizado
CREATE LOGIN login_sql_analista WITH PASSWORD = 'ContraseñaSegura123$';
GO
CREATE USER Analista FOR LOGIN login_sql_analista;
GO
ALTER ROLE rol_autorizado ADD MEMBER [Analista];
GO
```

Validación

```
-- Conceder permiso SELECT sobre la vista al rol
GRANT SELECT ON dbo.vw_Proveedor_Segura TO rol_autorizado;
GO
-- Usuario autorizado
EXECUTE AS USER = 'Analista';
SELECT * FROM dbo.vw_Proveedor_Segura;
REVERT;
```



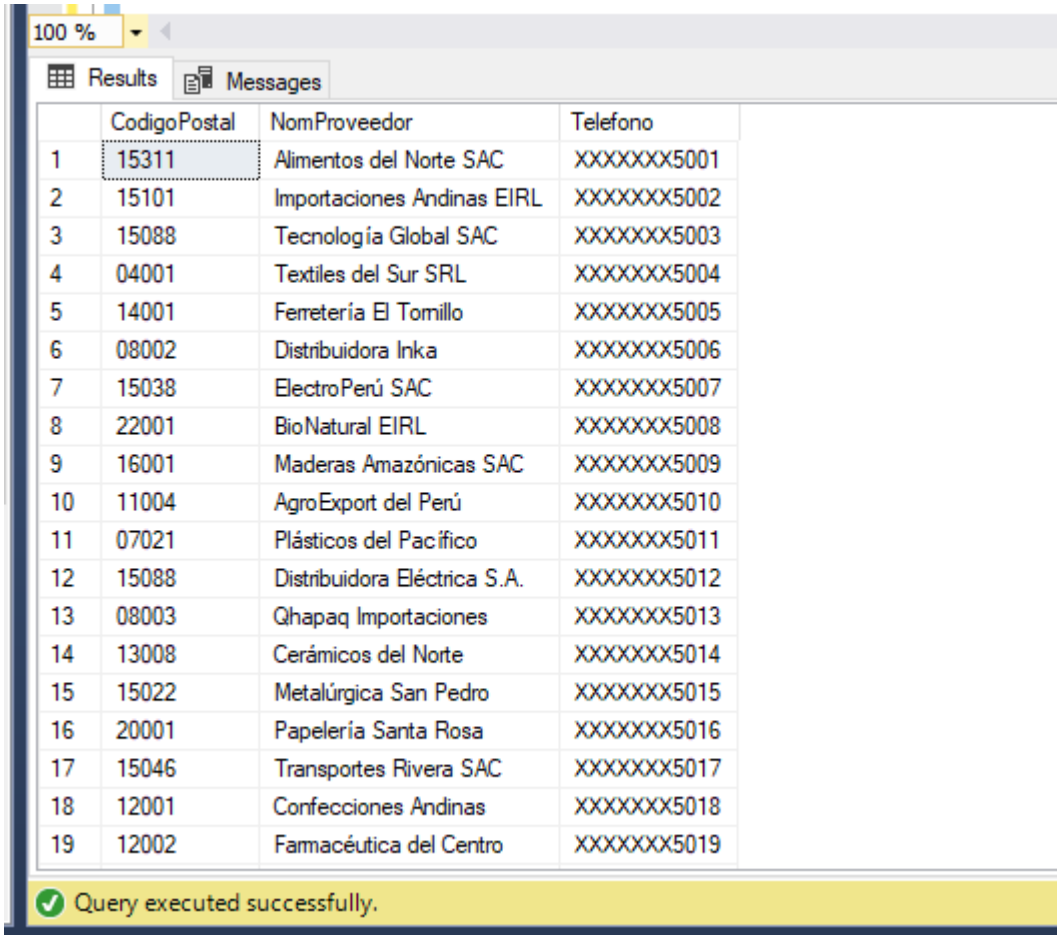
100 %

Results | Messages

	CodigoPostal	NomProveedor	Telefono
1	15311	Alimentos del Norte SAC	XXXXXXX5001
2	15101	Importaciones Andinas EIRL	XXXXXXX5002
3	15088	Tecnología Global SAC	XXXXXXX5003
4	04001	Textiles del Sur SRL	XXXXXXX5004
5	14001	Ferretería El Tornillo	XXXXXXX5005
6	08002	Distribuidora Inka	XXXXXXX5006
7	15038	ElectroPerú SAC	XXXXXXX5007
8	22001	BioNatural EIRL	XXXXXXX5008
9	16001	Maderas Amazónicas SAC	XXXXXXX5009
10	11004	AgroExport del Perú	XXXXXXX5010
11	07021	Plásticos del Pacífico	XXXXXXX5011
12	15088	Distribuidora Eléctrica S.A.	XXXXXXX5012
13	08003	Qhapaq Importaciones	XXXXXXX5013
14	13008	Cerámicos del Norte	XXXXXXX5014
15	15022	Metalúrgica San Pedro	XXXXXXX5015
16	20001	Papelería Santa Rosa	XXXXXXX5016
17	15046	Transportes Rivera SAC	XXXXXXX5017
18	12001	Confecciones Andinas	XXXXXXX5018

Query executed successfully.

```
CREATE LOGIN Invitado WITH PASSWORD = 'Invitado123*';

CREATE USER Invitado FOR LOGIN Invitado;
GO
GRANT SELECT ON dbo.vw_Proveedor_Segura TO Invitado;
EXECUTE AS USER = 'Invitado';
SELECT * FROM dbo.vw_Proveedor_Segura;
REVERT;
```



100 %

Results | Messages

	CodigoPostal	NomProveedor	Telefono
1	15311	Alimentos del Norte SAC	XXXXXXX5001
2	15101	Importaciones Andinas EIRL	XXXXXXX5002
3	15088	Tecnología Global SAC	XXXXXXX5003
4	04001	Textiles del Sur SRL	XXXXXXX5004
5	14001	Ferretería El Tornillo	XXXXXXX5005
6	08002	Distribuidora Inka	XXXXXXX5006
7	15038	ElectroPerú SAC	XXXXXXX5007
8	22001	BioNatural EIRL	XXXXXXX5008
9	16001	Maderas Amazónicas SAC	XXXXXXX5009
10	11004	AgroExport del Perú	XXXXXXX5010
11	07021	Plásticos del Pacífico	XXXXXXX5011
12	15088	Distribuidora Eléctrica S.A.	XXXXXXX5012
13	08003	Qhapaq Importaciones	XXXXXXX5013
14	13008	Cerámicos del Norte	XXXXXXX5014
15	15022	Metalúrgica San Pedro	XXXXXXX5015
16	20001	Papelería Santa Rosa	XXXXXXX5016
17	15046	Transportes Rivera SAC	XXXXXXX5017
18	12001	Confecciones Andinas	XXXXXXX5018
19	12002	Farmacéutica del Centro	XXXXXXX5019

Query executed successfully.

Justificación técnica

Dynamic Data Masking (DDM) protege datos sensibles sin modificar la aplicación.
Permite ocultar información confidencial como teléfono, correo o SSN para usuarios no autorizados.
La función fn_VerTelefonoCompleto() valida permisos dinámicamente, garantizando que solo usuarios con rol especial vean la información completa.
Evita exponer datos en informes o consultas ad-hoc a usuarios que no deberían acceder.

Buenas prácticas utilizadas

Enmascaramiento a nivel columna, minimizando cambios en la estructura de la base.
Uso de roles y funciones para controlar acceso a datos completos.
Validar siempre los permisos de usuario con IS_MEMBER o HAS_PERMS_BY_NAME.
Evitar almacenar datos sensibles en vistas, logs o reportes accesibles a todos.
Mantener documentación de columnas enmascaradas y usuarios autorizados.

Proyecto 10: Capstone: Integración (roles, TDE, Always Encrypted, auditoría)

1. Enunciado del ejercicio

   Proyecto integrador: crear un rol auditor_seguridad, habilitar TDE (si no está), preparar Always Encrypted para columna sensible, configurar auditoría de accesos a esa tabla, y dejar un procedimiento almacenado que registre cambios críticos (traza soportada por audit).

Proyecto 10: capstone: Integración (roles, TDE, Always Encrypted, auditoria)
Enunciado
Proyecto integrado crea un rol auditor_segurirdad habilitar TDE (sino esta), preparar always encrypted para columna sensible, configurar auditoria de acceso a esa tabla y dejar un procedimiento almacenado que registre cambios críticos (traza soportada por audit)

script de la solución en T-SQL

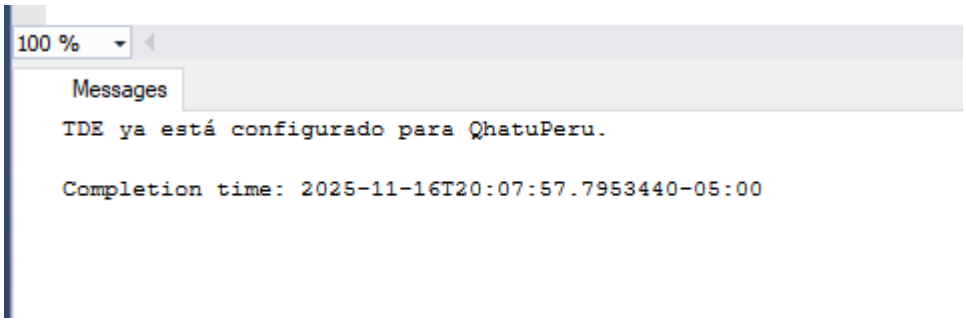
Crear rol de auditoría (servidor/BD) y usuarios de ejemplo

```
USE master;
GO

-- Rol de servidor no es necesario; crearemos rol de BD para auditoría
USE QhatuPeru;
GO
IF NOT EXISTS (SELECT 1 FROM sys.database_principals WHERE name = 'auditor_seguridad')
    CREATE ROLE auditor_seguridad;
GO

-- Ejemplo: crear logins/usuarios de prueba (cambia contraseñas por seguras)
IF NOT EXISTS (SELECT 1 FROM sys.server_principals WHERE name = 'login_auditor')
    CREATE LOGIN login_auditor WITH PASSWORD = 'Aud1t0r!2025';
GO

IF NOT EXISTS (SELECT 1 FROM sys.database_principals WHERE name = 'auditor_usuario')
    CREATE USER auditor_usuario FOR LOGIN login_auditor;
GO

ALTER ROLE auditor_seguridad ADD MEMBER auditor_usuario;
GO
```

Asegurar/activar TDE en QhatuPeru

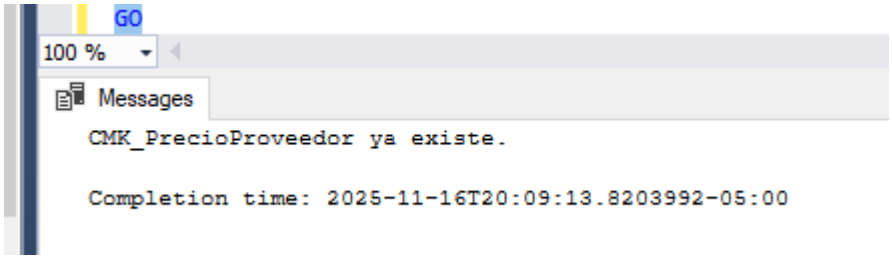
```
-- 2.4 Crear Database Encryption Key y activar (solo si no existe)
USE QhatuPeru;
GO

DECLARE @TDE_Activo INT;

SELECT @TDE_Activo = encryption_state
FROM sys.dm_database_encryption_keys
WHERE database_id = DB_ID('QhatuPeru');

IF @TDE_Activo IS NULL
BEGIN
    PRINT 'Creando DEK y activando TDE en QhatuPeru...';

    -- Crear DEK
    CREATE DATABASE ENCRYPTION KEY
    WITH ALGORITHM = AES_256
    ENCRYPTION BY SERVER CERTIFICATE TDE_Cert_QhatuPeru;

    PRINT 'DEK creada correctamente. Activando TDE...';
```

```
    -- Activar TDE
    ALTER DATABASE QhatuPeru SET ENCRYPTION ON;

    PRINT 'TDE activado. Puedes monitorear progreso en
sys.dm_database_encryption_keys.';
END
ELSE
BEGIN
    PRINT 'TDE ya está configurado para QhatuPeru.';
END;
GO
```



Preparar Always Encrypted — crear Column Master Key (CMK) por T-SQL

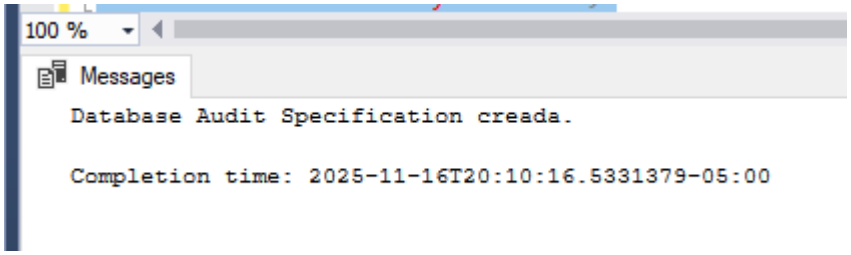
```
IF NOT EXISTS (SELECT 1 FROM sys.column_master_keys WHERE name =
'CMK_PrecioProveedor')
BEGIN
    CREATE COLUMN MASTER KEY CMK_PrecioProveedor
    WITH (
        KEY_STORE_PROVIDER_NAME = 'MSSQL_CERTIFICATE_STORE',
        KEY_PATH = 'CurrentUser/My/TDE_Cert_QhatuPeru'  -- ajusta: thumbprint o path
del certificado real
    );
    PRINT 'CMK creada (referencia al certificado).';
END
ELSE
    PRINT 'CMK_PrecioProveedor ya existe.';
GO
```



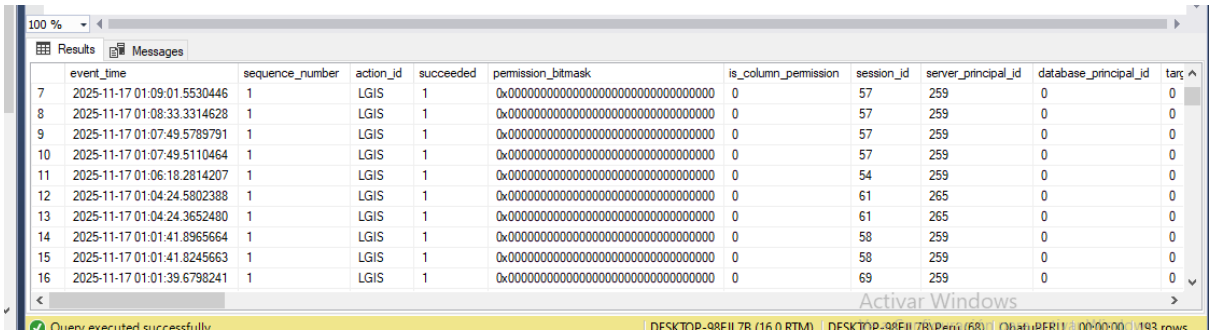
Auditoría: crear Server Audit + DB Audit Specification que registre accesos a la tabla sensible y ejecución del procedimiento

```
USE master;
GO

-- 4.1 Crear Server Audit (archivo)
IF NOT EXISTS (SELECT 1 FROM sys.server_audits WHERE name = 'Audit_QhatuPeru')
BEGIN
    CREATE SERVER AUDIT Audit_QhatuPeru
    TO FILE ( FILEPATH = 'C:\SQLAudit\QhatuPeru\' )
    WITH (QUEUE_DELAY = 1000, ON_FAILURE = CONTINUE);
    PRINT 'Server Audit creado.';
END
ELSE
```

```
    PRINT 'Server Audit ya existe.';
GO

ALTER SERVER AUDIT Audit_QhatuPeru WITH (STATE = ON);
GO

-- 4.2 Crear Database Audit Specification para registrar SELECT/INSERT/UPDATE/DELETE
sobre tabla Proveedor
USE QhatuPeru;
GO

IF NOT EXISTS (SELECT 1 FROM sys.database_audit_specifications WHERE name =
'DBAuditSpec_QhatuPeru_Proveedor')
BEGIN
    CREATE DATABASE AUDIT SPECIFICATION DBAuditSpec_QhatuPeru_Proveedor
    FOR SERVER AUDIT Audit_QhatuPeru
    -- auditar acceso a la tabla Proveedor (SELECT/INSERT/UPDATE/DELETE)
    ADD (SELECT ON OBJECT::dbo.Proveedor BY PUBLIC),
    ADD (INSERT ON OBJECT::dbo.Proveedor BY PUBLIC),
    ADD (UPDATE ON OBJECT::dbo.Proveedor BY PUBLIC),
    ADD (DELETE ON OBJECT::dbo.Proveedor BY PUBLIC);

    -- auditar la ejecución del procedimiento de registro (ver más abajo)
    -- ADD (EXECUTE ON OBJECT::dbo.sp_RegisterCriticalChange BY PUBLIC)   -- si
quieres auditar ejecuciones específicas
    ;
    PRINT 'Database Audit Specification creada.';
END
ELSE
    PRINT 'Database Audit Specification ya existe.';
GO

ALTER DATABASE AUDIT SPECIFICATION DBAuditSpec_QhatuPeru_Proveedor WITH (STATE = ON);
GO
```

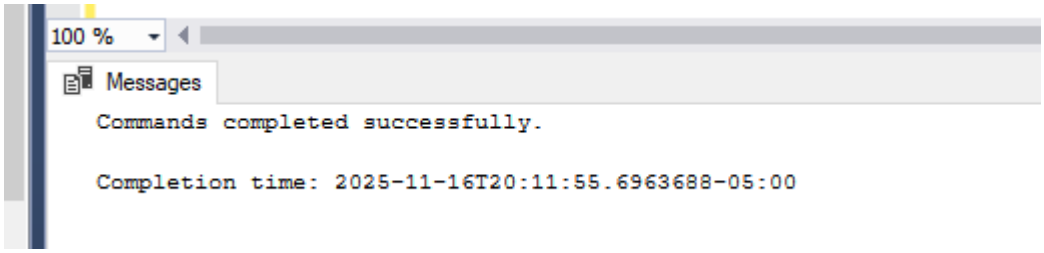



```
SELECT TOP(200) *
FROM sys.fn_get_audit_file('C:\SQLAudit\QhatuPeru\*', DEFAULT, DEFAULT)
ORDER BY event_time DESC;
GO
```

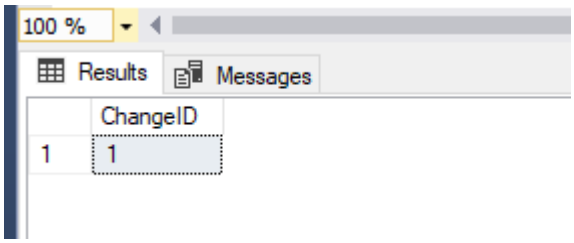


Procedimiento almacenado para registrar cambios críticos (log local) — y auditar su ejecución

```sql
USE QhatuPeru;
GO

-- 5.1 Tabla de trazas locales
IF OBJECT_ID('dbo.Audit_Changes', 'U') IS NULL
BEGIN
    CREATE TABLE dbo.Audit_Changes (
        ChangeID INT IDENTITY(1,1) PRIMARY KEY,
        ChangeTime DATETIME2 NOT NULL DEFAULT SYSUTCDATETIME(),
        ChangedBy SYSNAME NULL,
        ObjectName NVARCHAR(256) NULL,
        ChangeType NVARCHAR(50) NULL,
        Details NVARCHAR(MAX) NULL
    );
    PRINT 'Tabla Audit_Changes creada.';
END
ELSE
    PRINT 'Tabla Audit_Changes ya existe.';
GO

-- 5.2 Procedimiento que registra cambios
CREATE OR ALTER PROCEDURE dbo.sp_RegisterCriticalChange
    @ObjectName NVARCHAR(256),
    @ChangeType NVARCHAR(50),
    @Details NVARCHAR(MAX)
AS
BEGIN
    SET NOCOUNT ON;
    INSERT INTO dbo.Audit_Changes (ChangedBy, ObjectName, ChangeType, Details)
    VALUES (SUSER_SNAME(), @ObjectName, @ChangeType, @Details);

    -- Opcional: devolver el id insertado
    SELECT SCOPE_IDENTITY() AS ChangeID;
END
GO

-- 5.3 Conceder permisos de ejecución al rol auditor
GRANT EXECUTE ON dbo.sp_RegisterCriticalChange TO auditor_seguridad;
GO
```

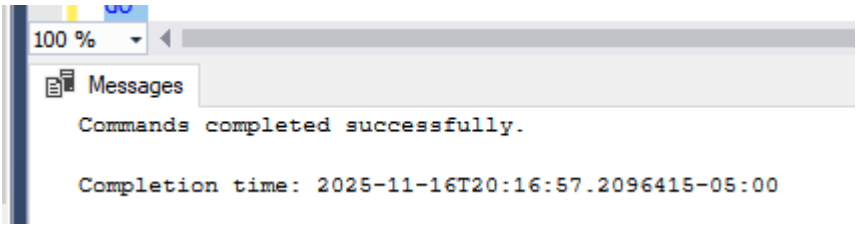



```sql
EXEC dbo.sp_RegisterCriticalChange
    @ObjectName = 'Proveedor',
    @ChangeType = 'UPDATE Precio',
    @Details = 'Se actualizó PrecioProveedor_ENC para ProveedorID=123';
GO
```



Vista/Consulta segura para auditores (ver audit logs y tabla de trazas)

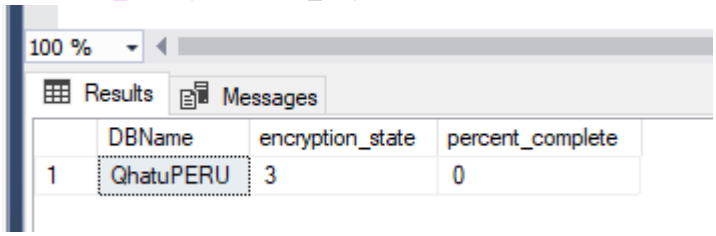
```
USE QhatuPeru;
GO

-- Vista que devuelve trazas locales
CREATE OR ALTER VIEW dbo.vw_Audit_Changes AS
SELECT ChangeID, ChangeTime, ChangedBy, ObjectName, ChangeType, Details
FROM dbo.Audit_Changes;
GO

-- Conceder SELECT a rol auditor
GRANT SELECT ON dbo.vw_Audit_Changes TO auditor_seguridad;
GO

-- Vista para consultar archivo de audit (requiere acceso al server file path)
-- Nota: sólo ejecuta esta vista con permisos adecuados en servidor (fn_get_audit_file
es server-level)
-- Para seguridad, en general solo DBA / auditor deberían poder ejecutarla.
CREATE OR ALTER VIEW dbo.vw_ServerAuditFile AS
SELECT * FROM sys.fn_get_audit_file('C:\SQLAudit\QhatuPeru\*', DEFAULT, DEFAULT);
GO

-- No conceder acceso público a esta vista; dá acceso solo a auditor_usuario si es
necesario:
DENY SELECT ON dbo.vw_ServerAuditFile TO PUBLIC;
GRANT SELECT ON dbo.vw_ServerAuditFile TO auditor_seguridad;
GO
```



Validaciones rápidas (ejecutar y comprobar)

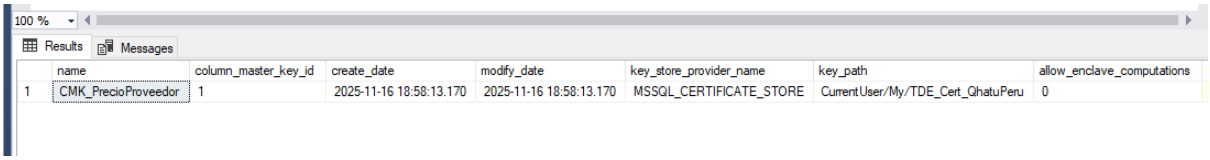
Ver TDE:

```
SELECT db_name(database_id) AS DBName, encryption_state, percent_complete
FROM sys.dm_database_encryption_keys
WHERE db_name(database_id) = 'QhatuPeru';
```

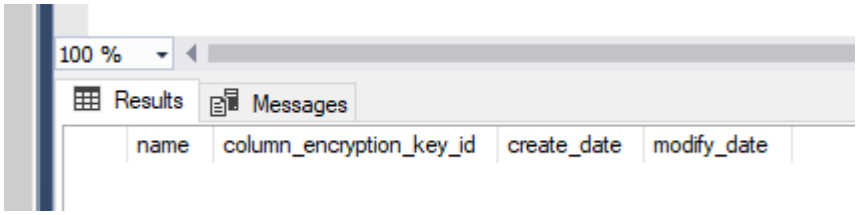


Ver CMK:

```
SELECT * FROM sys.column_master_keys;
GO
```

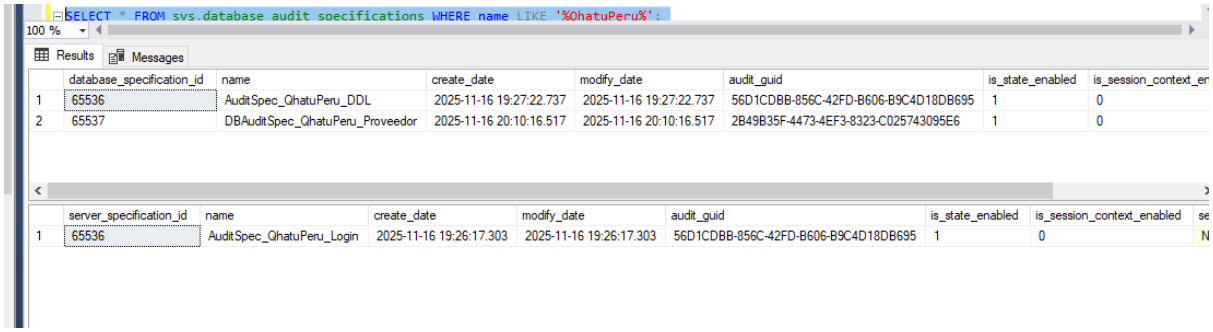


(Después de crear CEK desde SSMS) ver CEK



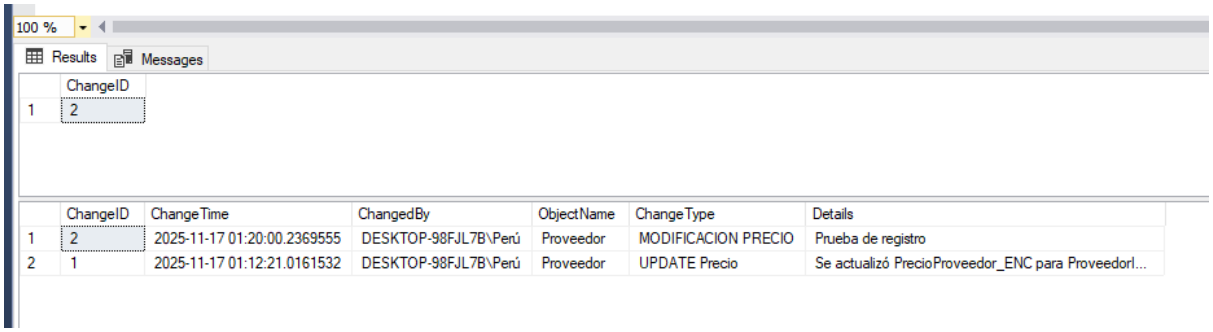
Ver specification de auditoría:

```
SELECT * FROM sys.database_audit_specifications WHERE name LIKE '%QhatuPeru%';
SELECT * FROM sys.server_audit_specifications WHERE name LIKE '%QhatuPeru%';
GO
```



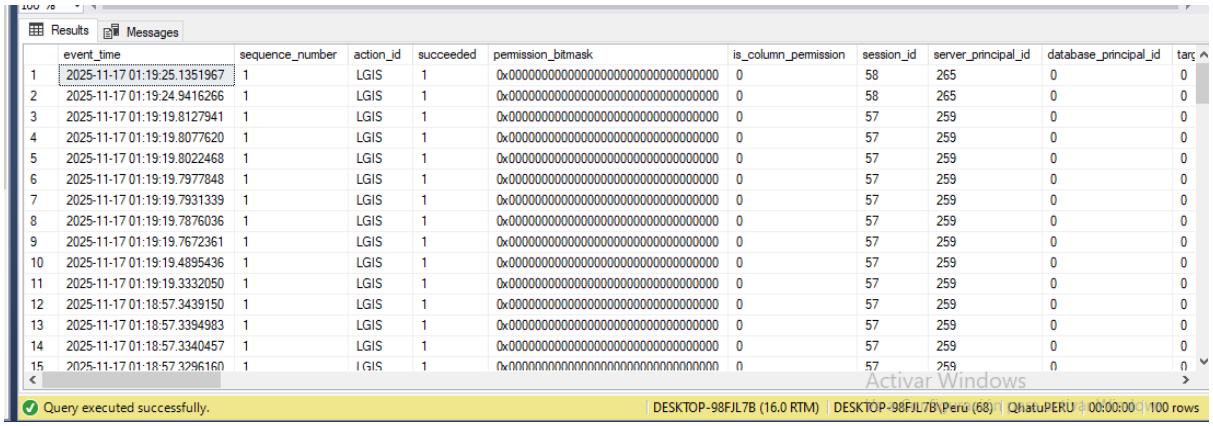
Ejecutar proc de registro y comprobar tabla:

```
EXEC dbo.sp_RegisterCriticalChange 'Proveedor','MODIFICACION PRECIO','Prueba de
registro';
SELECT TOP(10) * FROM dbo.Audit_Changes ORDER BY ChangeTime DESC;
GO
```



Consultar los archivos de auditoría:

```
SELECT TOP(100) * FROM sys.fn_get_audit_file('C:\SQLAudit\QhatuPeru\*', DEFAULT,
DEFAULT) ORDER BY event_time DESC;
GO
```



Justificación técnica

Un único proyecto integrado combina control de accesos (roles), protección de datos en reposo (TDE), protección de columnas sensibles (Always Encrypted), auditoría de accesos y cambios (Server/DB Audit) y un registro local de cambios críticos (procedimiento + tabla).
TDE protege archivos físicos (mdf/ldf/backups).
Always Encrypted asegura que la columna sensible solo puede descifrarse en clientes autorizados que tengan la CMK accesible.
Auditoría registra accesos y cambios para detección y cumplimiento (forense, normativa).
Procedimiento de registro complementa la auditoría al dejar una traza estructurada dentro del BD que puede ser consultada y vinculada a evidencias de auditoría externa.

Buenas prácticas aplicadas

Principio de mínimos privilegios (rol auditor con permisos limitados).
Respaldo inmediato del certificado TDE y guardado offsite.
Guardar CMK/CEK y sus respaldos en almacenes seguros; CEK creado desde cliente seguro.
Logs de auditoría en ruta protegida con ACLs a la cuenta del servicio.
Separar auditoría del registro local (ambos se complementan).
Documentar procedimientos de restauración (certificado + clave) y rotación de claves.
Testear restauraciones en laboratorio.