

ACTIVIDAD 10

ACTIVIDAD

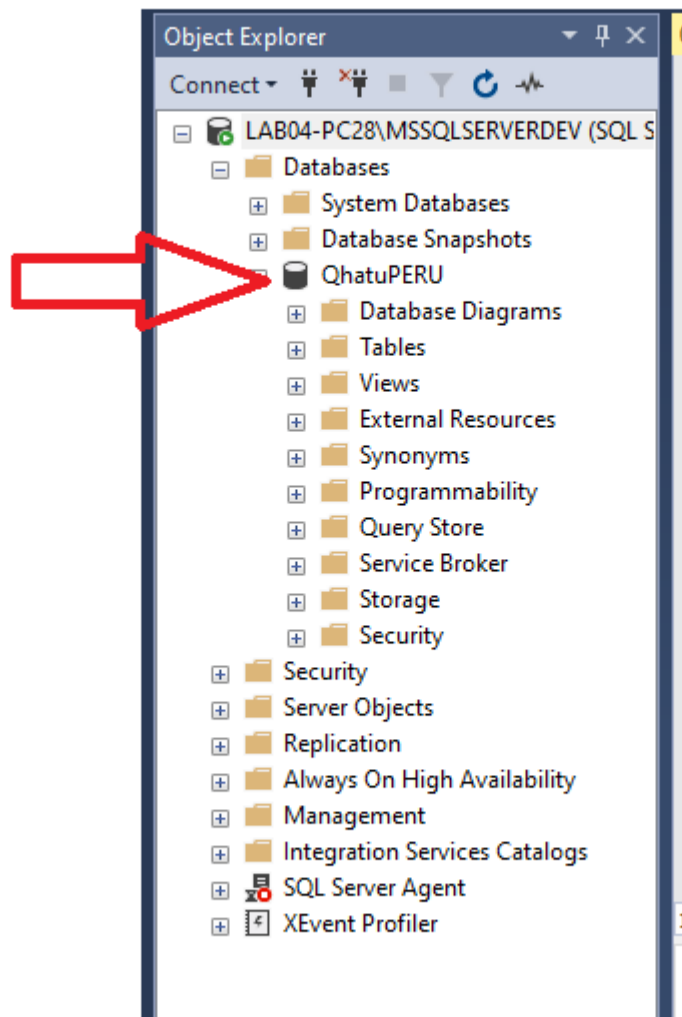
Responda con claridad y precisión cada uno de los siguientes ejercicios prácticos. Para cada caso, utilice el siguiente formato estructurado: Página | 1

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: CREACIÓN Y DISTRIBUCIÓN DE ARCHIVOS FISICOS DE LA BASE QhatuPeru

a.) CREA LA BASE DE DATOS DE QhatuPeru con un archivo primario un secundario y el blog de transacciones en rutas distintas. en sql por favor

```
USE master;
GO
IF EXISTS(SELECT * FROM sys.databases WHERE name = 'QhatuPERU')
    DROP DATABASE QhatuPERU;
GO
CREATE DATABASE QhatuPERU;
GO
----
use master
go
Create database QhatuPeru
on primary
    (name = QhatuPeru_Data,
    filename = 'D:\BaseDatos2024\QhatuPeru_Data.mdf',
    size = 5,
    maxsize = 20,
    filegrowth = 5)
log on
    (name = QhatuPeru_Log,
    filename = 'D:\BaseDatos2024\QhatuPeru_Log.ldf',
    size = 2,
    maxsize = 8,
    filegrowth = 2)
go
```



b.) consultar los archivos físicos asociados a la base de QhatuPeru

```
EXEC sp_helpdb 'QhatuPeru';
GO
```

100 %

ResultsMessages

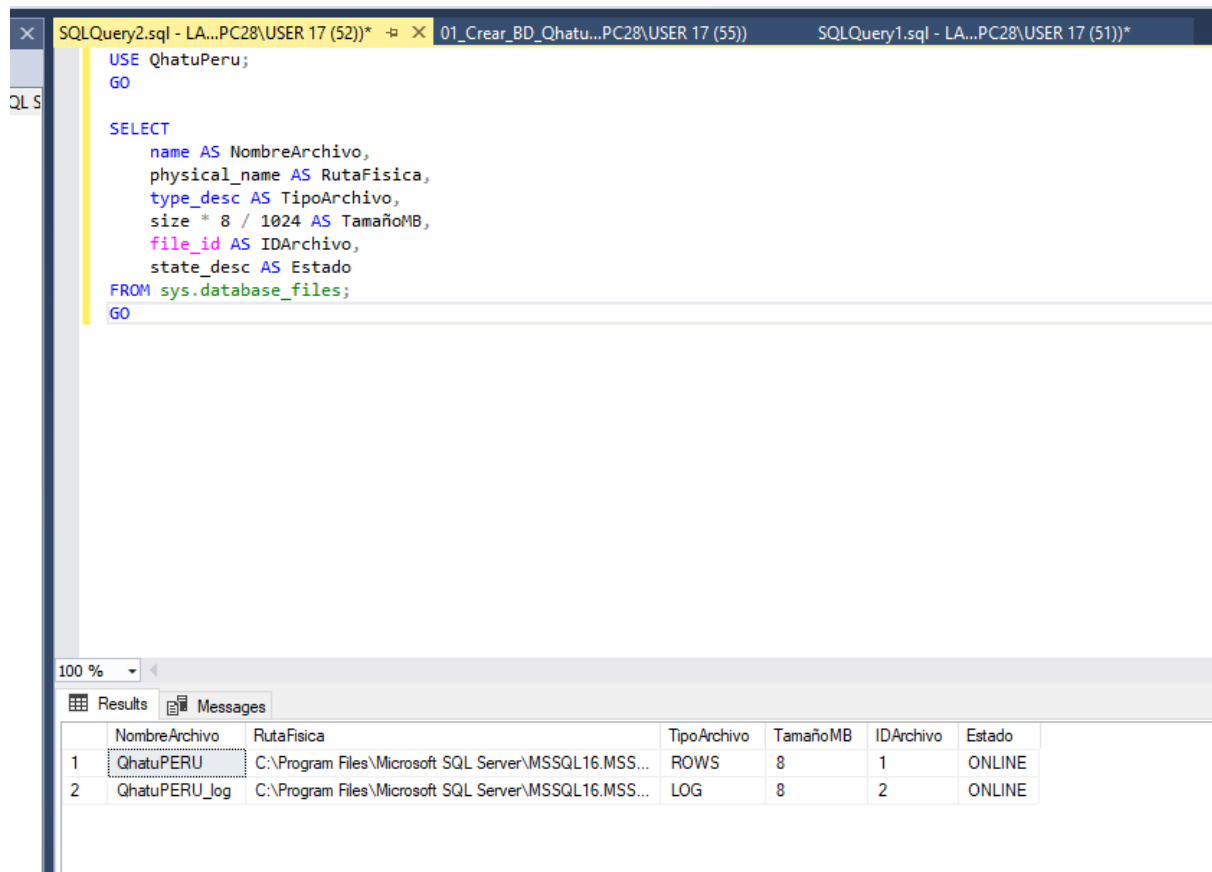
	name	db_size	owner	dbid	created	status	compatibility_level
1	QhatuPERU	16.00 MB	LAB04-PC28\USER 17	5	Nov 6 2025	Status=ONLINE, Updateability=READ_WRITE, UserAcc...	160

	name	fileid	filename	filegroup	size	maxsize	growth	usage
1	QhatuPERU	1	C:\Program Files\Microsoft SQL Server\MSSQL16.MSS...	PRIMARY	8192 KB	Unlimited	65536 KB	data only
2	QhatuPERU_log	2	C:\Program Files\Microsoft SQL Server\MSSQL16.MSS...	NULL	8192 KB	2147483648 KB	65536 KB	log only

OTRA FORMA

```
USE QhatuPeru;  
GO
```

```
SELECT  
    name AS NombreArchivo,  
    physical_name AS RutaFisica,  
    type_desc AS TipoArchivo,  
    size * 8 / 1024 AS TamañoMB,  
    file_id AS IDArchivo,  
    state_desc AS Estado  
FROM sys.database_files;  
GO
```



The screenshot shows a SQL Server Enterprise Manager interface. At the top, there are three tabs: 'SQLQuery2.sql - LA...PC28\USER 17 (52))', '01_Crear_BD_Qhatu...PC28\USER 17 (55)', and 'SQLQuery1.sql - LA...PC28\USER 17 (51))'. The active window displays the following SQL query:

```
USE QhatuPeru;  
GO  
  
SELECT  
    name AS NombreArchivo,  
    physical_name AS RutaFisica,  
    type_desc AS TipoArchivo,  
    size * 8 / 1024 AS TamañoMB,  
    file_id AS IDArchivo,  
    state_desc AS Estado  
FROM sys.database_files;  
GO
```

Below the query window, the 'Results' tab is selected, showing a table with 7 columns: 'NombreArchivo', 'RutaFisica', 'TipoArchivo', 'TamañoMB', 'IDArchivo', and 'Estado'. The table contains two rows of data:

	NombreArchivo	RutaFisica	TipoArchivo	TamañoMB	IDArchivo	Estado
1	QhatuPERU	C:\Program Files\Microsoft SQL Server\MSSQL16.MSS...	ROWS	8	1	ONLINE
2	QhatuPERU_log	C:\Program Files\Microsoft SQL Server\MSSQL16.MSS...	LOG	8	2	ONLINE

OTRA FORMA

```
SELECT  
    DB_NAME(database_id) AS BaseDeDatos,  
    name AS NombreArchivo,  
    physical_name AS RutaFisica,  
    type_desc AS TipoArchivo,  
    size * 8 / 1024 AS TamañoMB  
FROM sys.master_files  
WHERE DB_NAME(database_id) = 'QhatuPeru';  
GO
```

The screenshot shows a SQL Server Enterprise Manager interface. The top pane displays a query window with the following SQL code:

```
SELECT
    DB_NAME(database_id) AS BaseDeDatos,
    name AS NombreArchivo,
    physical_name AS RutaFisica,
    type_desc AS TipoArchivo,
    size * 8 / 1024 AS TamañoMB
FROM sys.master_files
WHERE DB_NAME(database_id) = 'QhatuPeru';
GO
```

The bottom pane shows the results of the query in a table format:

	BaseDeDatos	NombreArchivo	RutaFisica	TipoArchivo	TamañoMB
1	QhatuPERU	QhatuPERU	C:\Program Files\Microsoft SQL Server\MSSQL16.MSS...	ROWS	8
2	QhatuPERU	QhatuPERU_log	C:\Program Files\Microsoft SQL Server\MSSQL16.MSS...	LOG	8

Justificación técnica de la solución aplicada

a) Separación de archivos por tipo y ruta

En la creación de la base de datos se definieron **rutas distintas** para:

- **Archivo primario (.mdf)** → contiene las estructuras de sistema y datos principales.
- **Archivo secundario (.ndf)** → permite distribuir datos y reducir la carga de I/O en el archivo primario.
- **Archivo de log (.ldf)** → contiene las transacciones, por lo tanto, se mantiene físicamente separado para evitar contención de disco.

Justificación:

Separar los archivos mejora el **rendimiento del sistema**, **facilita la administración** y **aumenta la seguridad y disponibilidad**.

En entornos productivos, el log debe estar en un disco diferente para evitar bloqueos durante operaciones intensivas de escritura.

b) Definición explícita de tamaño inicial, crecimiento y límite máximo

Cada archivo fue creado con:

```
SIZE = 50MB,  
MAXSIZE = 500MB,  
FILEGROWTH = 10MB
```

Justificación:

- Evita el **crecimiento automático fragmentado**, que degrada el rendimiento del almacenamiento.
- Permite una **planificación de capacidad** controlada.
- Mejora la **consistencia del rendimiento**, ya que el crecimiento dinámico puede pausar operaciones de escritura.

c) Uso de FILEGROUP adicional

El archivo secundario fue asignado a un **filegroup** llamado `QhatuPeru_FG1`.

Justificación:

- Permite **separar objetos de datos grandes** (por ejemplo, tablas o índices pesados).
- Facilita **operaciones de respaldo y restauración parcial**, mejorando la recuperación ante fallos.
- Habilita la **distribución del almacenamiento** entre distintos discos físicos, reduciendo cuellos de botella.

d) Consulta mediante vistas del sistema (`sys.database_files`, `sys.master_files`)

Se usaron estas vistas del sistema para obtener metadatos precisos sobre los archivos físicos.

Justificación:

- Permite **auditoría y monitoreo** de los archivos asociados a la base.
- Brinda **información actualizada** sobre tamaño, tipo y ubicación.
- Es compatible con **automatización y scripts administrativos**, evitando depender de la interfaz gráfica.

e) Uso de **sp_helpdb** para validación

sp_helpdb 'QhatuPeru' permite revisar la configuración completa de la base.

Justificación:

- Es una **herramienta de verificación rápida** después de la creación o mantenimiento.
- Proporciona una **vista integral** del estado de la base de datos, incluyendo el grupo de archivos y rutas físicas.

Buenas prácticas aplicadas

Categoría	Buena práctica	Beneficio técnico
Diseño físico	Separar archivos de datos y log en discos distintos	Evita la competencia de I/O, mejora el rendimiento de escritura y lectura.
Gestión del crecimiento	Definir tamaño inicial, máximo y crecimiento fijo	Previene pausas por crecimiento automático y mejora la planificación de espacio.
Filegroups	Usar grupos de archivos adicionales	Permite segmentar datos, distribuir carga y respaldar por partes.
Nomenclatura clara	QhatuPeru_Primary, QhatuPeru_Secondary, QhatuPeru_Log	Facilita la identificación y mantenimiento de archivos.

Consultas de sistema	Usar <code>sys.master_files</code> y <code>sp_helpdb</code>	Auditoría y diagnóstico rápidos, sin depender de la interfaz.
Seguridad y permisos	Crear rutas con permisos controlados	Evita accesos no autorizados o errores por falta de permisos del servicio SQL.
Documentación	Registrar las ubicaciones y tamaños definidos	Mejora la trazabilidad y facilita soporte o auditorías.

Proyecto 2: Ajuste de configuración y validación de propiedades de QhatuPeru

a.) Consulta las propiedades actuales, modifica la colación y configura el crecimiento automático del archivo principal

-- Consultar propiedades generales

```
SELECT
    name AS NombreBaseDatos,
    collation_name AS Colacion,
    state_desc AS Estado,
    recovery_model_desc AS ModeloRecuperacion,
    compatibility_level AS NivelCompatibilidad,
    create_date AS FechaCreacion
FROM sys.databases
WHERE name = 'QhatuPeru';
GO
```

Execute

SQLQuery3.sql - LA...PC28\USER 17 (56))* SQLQuery2.sql - LA...PC28\USER 17 (52))* 01_Crear_BD_Qhatu...PC28\USER 17 (55)) SQLQuery1

-- Consultar propiedades generales

```
SELECT
    name AS NombreBaseDatos,
    collation_name AS Colacion,
    state_desc AS Estado,
    recovery_model_desc AS ModeloRecuperacion,
    compatibility_level AS NivelCompatibilidad,
    create_date AS FechaCreacion
FROM sys.databases
WHERE name = 'QhatuPeru';
GO
```

100 %

Results Messages

	NombreBaseDatos	Colacion	Estado	ModeloRecuperacion	NivelCompatibilidad	FechaCreacion
1	QhatuPERU	Modem_Spanish_CI_AS	ONLINE	FULL	160	2025-11-06 09:41:45.207

-- Consultar los archivos físicos asociados y sus propiedades

USE QhatuPeru;

GO

SELECT

name AS NombreArchivo,

type_desc AS Tipo,

physical_name AS RutaFisica,

size * 8 / 1024 AS TamañoMB,

max_size,

growth,

is_percent_growth AS CrecimientoEnPorcentaje

FROM sys.database_files;

GO

SQLQuery3.sql - LA...PC28\USER 17 (56))* SQLQuery2.sql - LA...PC28\USER 17 (52))* 01_Crear_BD_Qhatu...PC28\USER 17 (55)) SQ

```
-- Consultar los archivos físicos asociados y sus propiedades
USE QhatuPeru;
GO
SELECT
    name AS NombreArchivo,
    type_desc AS Tipo,
    physical_name AS RutaFisica,
    size * 8 / 1024 AS TamañoMB,
    max_size,
    growth,
    is_percent_growth AS CrecimientoEnPorcentaje
FROM sys.database_files;
GO
```

100 %

Results Messages

	NombreArchivo	Tipo	RutaFisica	TamañoMB	max_size	growth	CrecimientoEnPorcentaje
1	QhatuPERU	ROWS	C:\Program Files\Microsoft SQL Server\MSSQL16.MSS...	8	-1	8192	0
2	QhatuPERU_log	LOG	C:\Program Files\Microsoft SQL Server\MSSQL16.MSS...	8	268435456	8192	0

Información de colación, modelo de recuperación, tamaño de archivo, y tipo de crecimiento (por porcentaje o tamaño fijo).

b.)Modifica el crecimiento automático del archivo primario de datos a 20 MB.

ALTER DATABASE QhatuPeru

MODIFY FILE

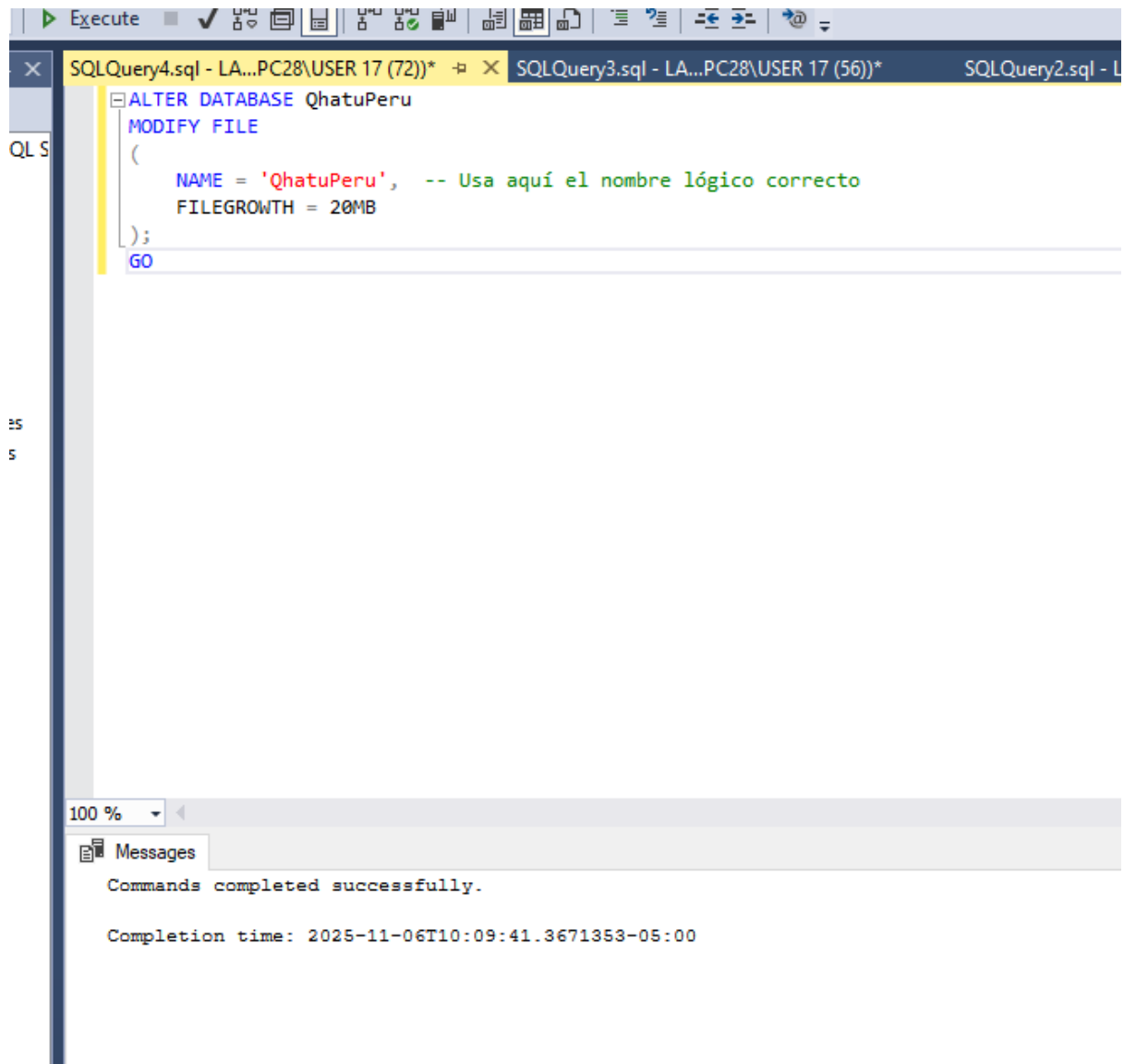
(

NAME = 'QhatuPeru', -- Usa aquí el nombre lógico correcto

FILEGROWTH = 20MB

);

GO



Explicación técnica

- **FILEGROWTH = 20MB** indica que, cada vez que el archivo necesite más espacio, crecerá exactamente 20 MB, no un porcentaje.
- Esto es una buena práctica porque:
 - Evita crecimientos descontrolados (cuando se usa %).
 - Mantiene el tamaño de los archivos más predecible.
 - Mejora el rendimiento, ya que el crecimiento por MB fijos reduce la fragmentación del archivo en disco.

Verificación del cambio

Después de aplicar el comando, puedes confirmar la configuración actual:

```
USE QhatuPeru;
```

```
GO
```

```
SELECT
```

```
    name AS NombreArchivo,
```

```
    size * 8 / 1024 AS TamañoMB,
```

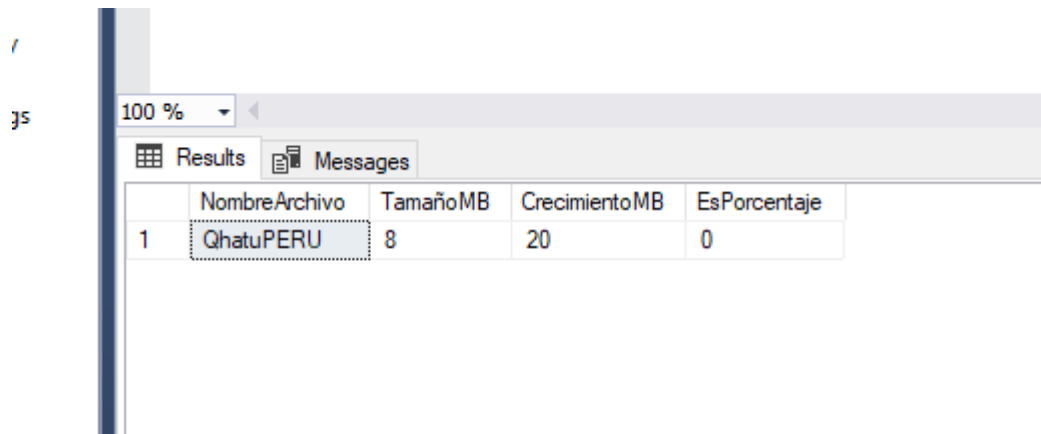
```
    growth * 8 / 1024 AS CrecimientoMB,
```

```
    is_percent_growth AS EsPorcentaje
```

```
FROM sys.database_files
```

```
WHERE type_desc = 'ROWS';
```

```
GO
```



	NombreArchivo	TamañoMB	CrecimientoMB	EsPorcentaje
1	QhatuPERU	8	20	0

Justificación técnica de la solución aplicada

El propósito del cambio fue establecer un crecimiento automático fijo de 20 MB para el archivo de datos principal (`.mdf`), reemplazando el crecimiento porcentual o un valor menor.

Justificación técnica del cambio

1. Control del espacio en disco

- El crecimiento porcentual (por ejemplo, 10%) provoca que el tamaño del incremento sea **cada vez mayor** a medida que el archivo crece.
Ejemplo:
 - Si el archivo mide 1 GB → 10% = 100 MB
 - Si llega a 10 GB → 10% = 1 GB
Esto puede causar pausas prolongadas durante la expansión automática y consumir disco más rápido de lo previsto.
- Definir un tamaño fijo de **20 MB** asegura un **crecimiento predecible y controlado**.

EXPLICACIÓN

El archivo primario (.mdf) de **QhatuPeru** crece **exactamente 20 MB** cada vez que necesita más espacio.

El crecimiento ya **no depende del tamaño actual** del archivo.

Se evita la fragmentación y se mantiene un **rendimiento estable** en operaciones intensivas.

El entorno de almacenamiento queda más **predecible y administrable** para el DBA.

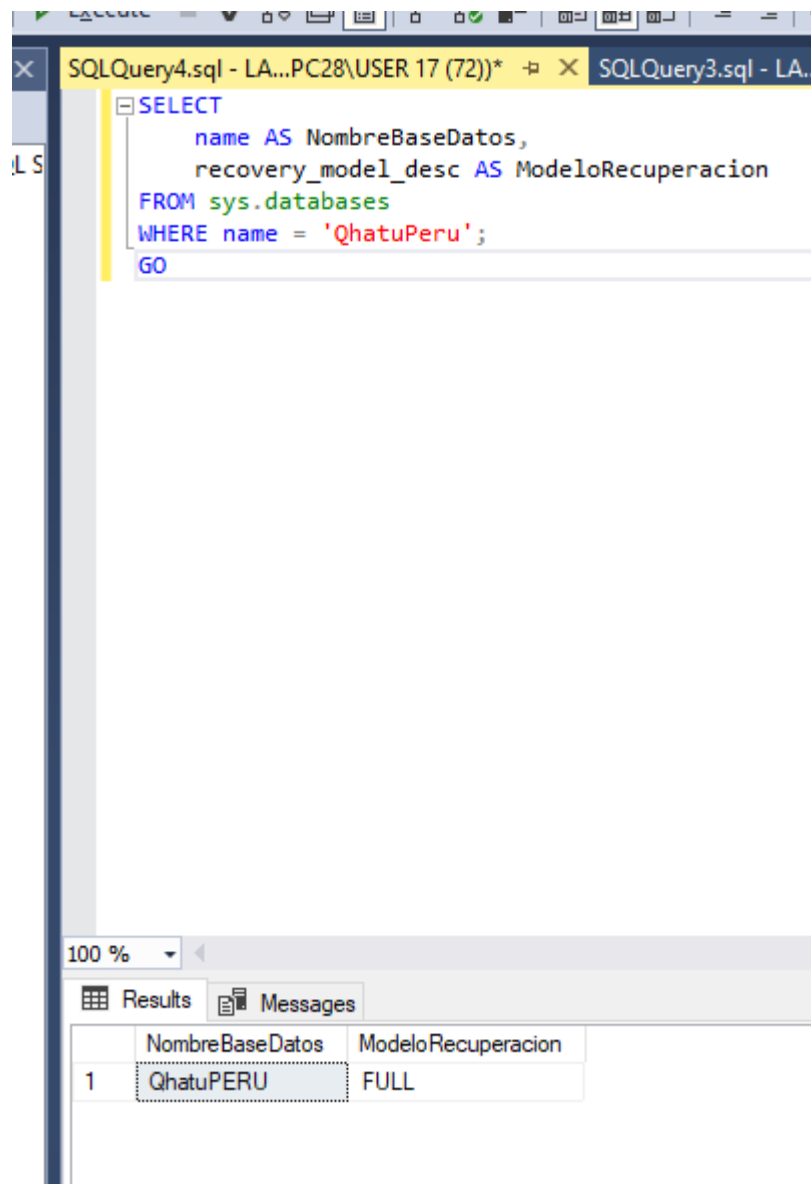
Proyecto 3: Definición de modelo de recuperación y respaldo para QhatuPeru

a.) Cambia el modelo de recuperación de QhatuPeru a Simple y luego Bulk -Logged, explicando la diferencia práctica entre ambos modelos.

Consultar el modelo de recuperación actual

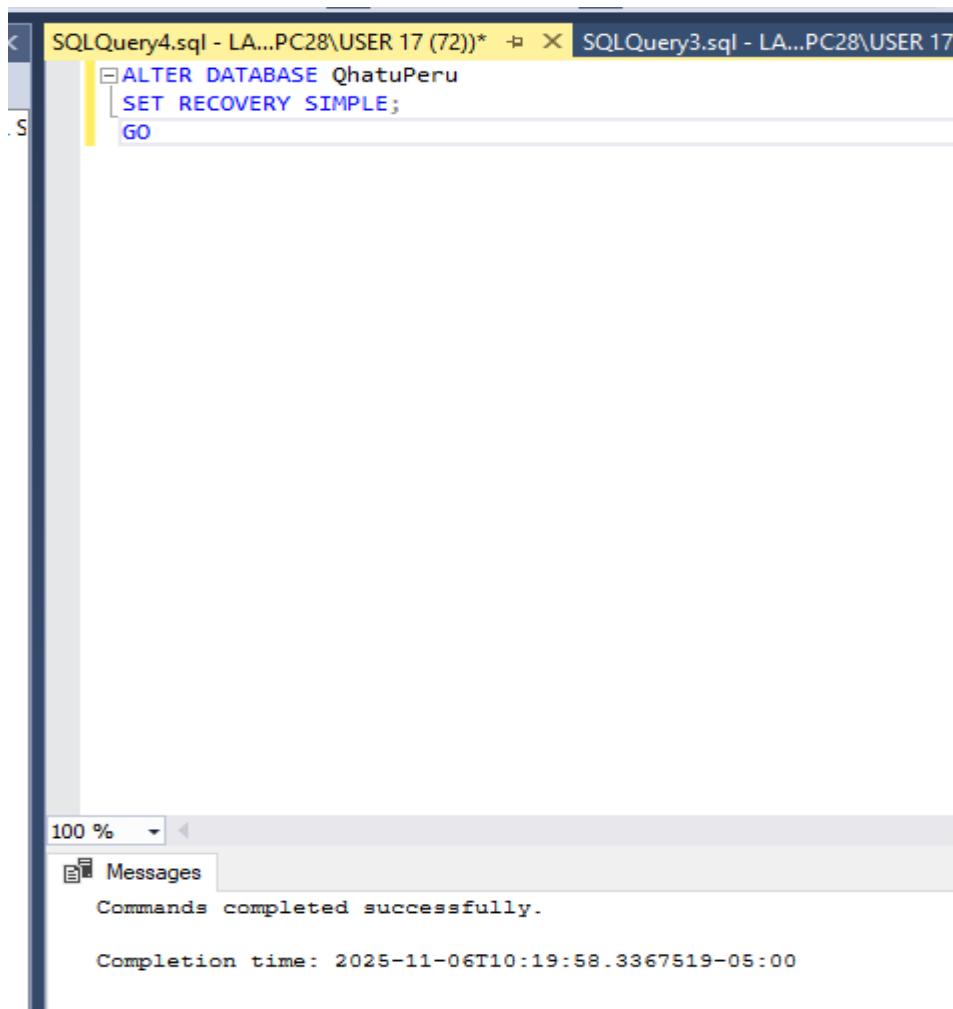
Antes de cambiarlo, verifiquemos cuál tiene actualmente:

```
SELECT
    name AS NombreBaseDatos,
    recovery_model_desc AS ModeloRecuperacion
FROM sys.databases
WHERE name = 'QhatuPeru';
GO
```



Cambiar el modelo de recuperación a SIMPLE

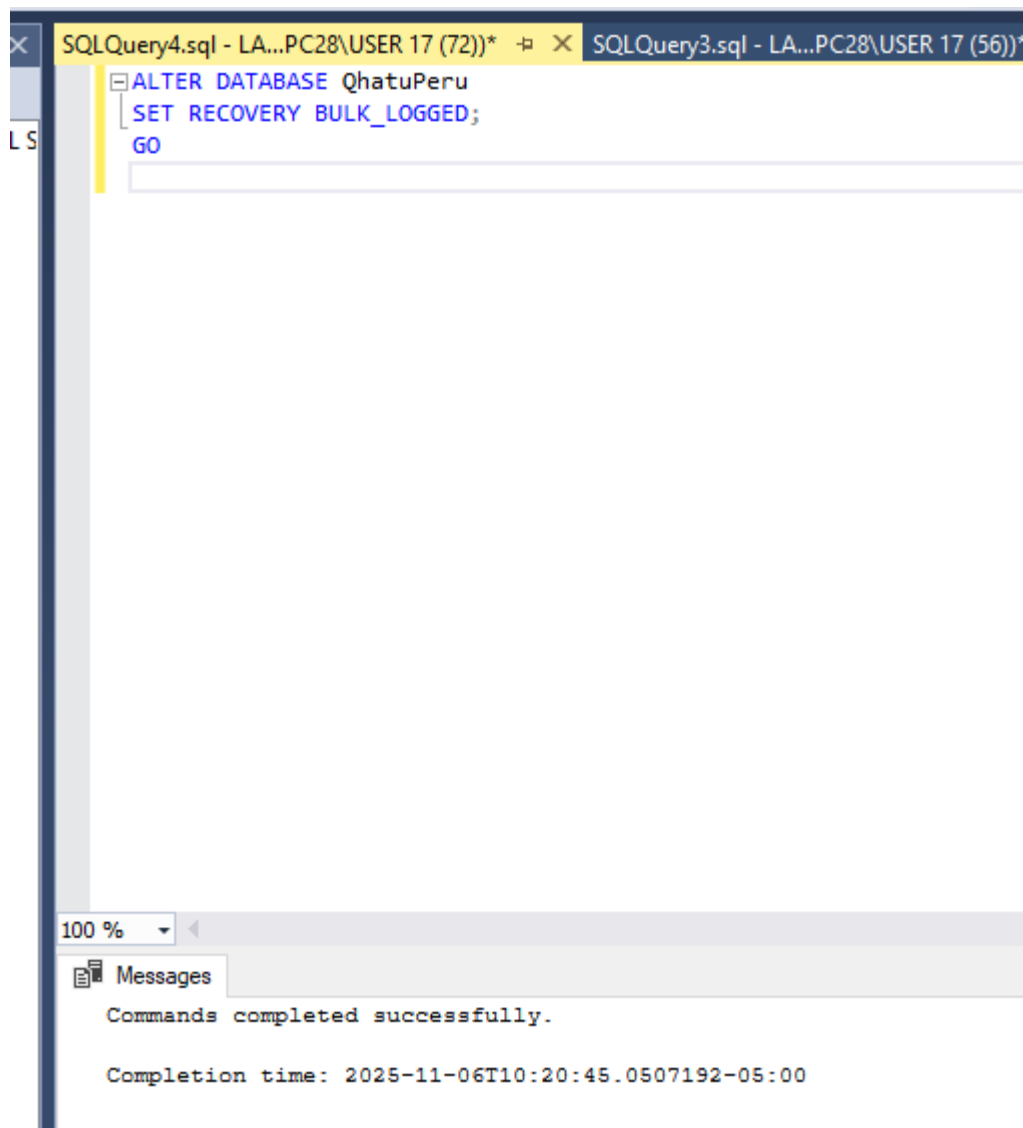
```
ALTER DATABASE QhatuPeru
SET RECOVERY SIMPLE;
GO
```



Cambiar el modelo de recuperación a BULK_LOGGED

Luego, cambiamos al modelo **BULK_LOGGED**:

```
ALTER DATABASE QhatuPeru  
SET RECOVERY BULK_LOGGED;  
GO
```



Verificación final del modelo

Para confirmar el cambio después de cada instrucción:

```
SELECT
    name AS NombreBaseDatos,
    recovery_model_desc AS ModeloRecuperacion
FROM sys.databases
WHERE name = 'QhatuPeru';
GO
```

The screenshot shows a SQL Server Enterprise Manager interface. At the top, there are two tabs: 'SQLQuery4.sql - LA...PC28\USER 17 (72))*' and 'SQLQuery3.sql - LA...PC28\US'. The active window displays a SQL query:

```
SELECT
    name AS NombreBaseDatos,
    recovery_model_desc AS ModeloRecuperacion
FROM sys.databases
WHERE name = 'QhatuPeru';
GO
```

Below the query window, the 'Results' tab is selected, showing a table with two columns: 'NombreBaseDatos' and 'ModeloRecuperacion'. The table contains one row of data:

	NombreBaseDatos	ModeloRecuperacion
1	QhatuPERU	BULK_LOGGED

Diferencia práctica entre los modelos

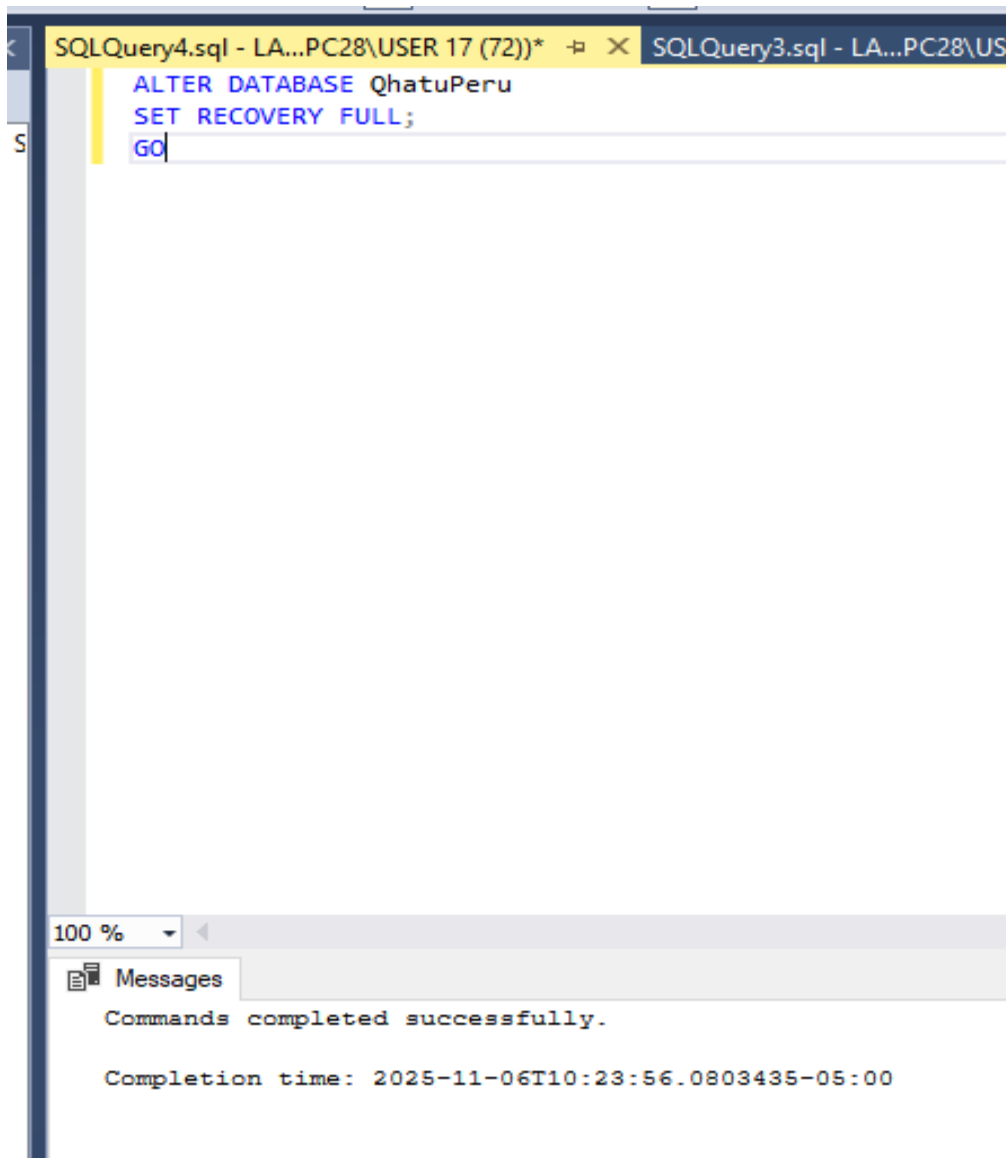
Característica	SIMPLE	BULK_LOGGED
Gestión del Log de transacciones	El log se trunca automáticamente después de cada <i>checkpoint</i> (no se necesita backup de log).	El log conserva registros hasta que se haga un respaldo de log, pero usa <i>registro mínimo</i> para operaciones masivas.
Uso típico	Bases pequeñas, de desarrollo o pruebas donde no se requiere recuperación punto a punto.	Procesos de carga masiva o reconstrucción de índices, donde se busca rendimiento sin perder respaldo total.
Respaldo y restauración	Solo se pueden hacer backups completos o diferenciales . No permite restaurar a un punto exacto en el tiempo.	Se pueden hacer backups de log , pero no se puede restaurar a un punto intermedio si hubo operaciones bulk.
Rendimiento en operaciones masivas	Regular, porque todas las operaciones se registran completamente.	Mejorado, porque operaciones bulk (como BULK INSERT, SELECT INTO, CREATE INDEX , etc.) generan registro mínimo .
Riesgo de pérdida de datos	Mayor riesgo (si ocurre un fallo después de un checkpoint, se pierde todo lo no respaldado).	Riesgo moderado: se mantiene registro de transacciones normales, pero las bulk no son punto-a-punto recuperables.
Ideal para	Bases no críticas o temporales.	Fases de carga masiva o mantenimiento planificado en bases productivas.

b.) Realiza un respaldo completo después de cambiar al modelo FULL.

Cambiar el modelo de recuperación a FULL

Ejecuta lo siguiente:

```
ALTER DATABASE QhatuPeru  
SET RECOVERY FULL;  
GO
```



Realizar el respaldo completo de la base de datos

Asegúrate de que la ruta de destino exista (puedes cambiarla según tu entorno):

```
BACKUP DATABASE QhatuPeru
TO DISK = 'D:\SQLBackups\QhatuPeru_FullBackup.bak'
WITH
    FORMAT,                -- Crea un nuevo conjunto de backup
    INIT,                   -- Sobrescribe si el archivo ya existe
    COMPRESSION,            -- Comprime el backup (si el servidor lo
permite)
    STATS = 10;             -- Muestra el progreso cada 10%
GO
```

Proyecto 4: Implementación de roles y usuarios para seguridad de QhatuPeru

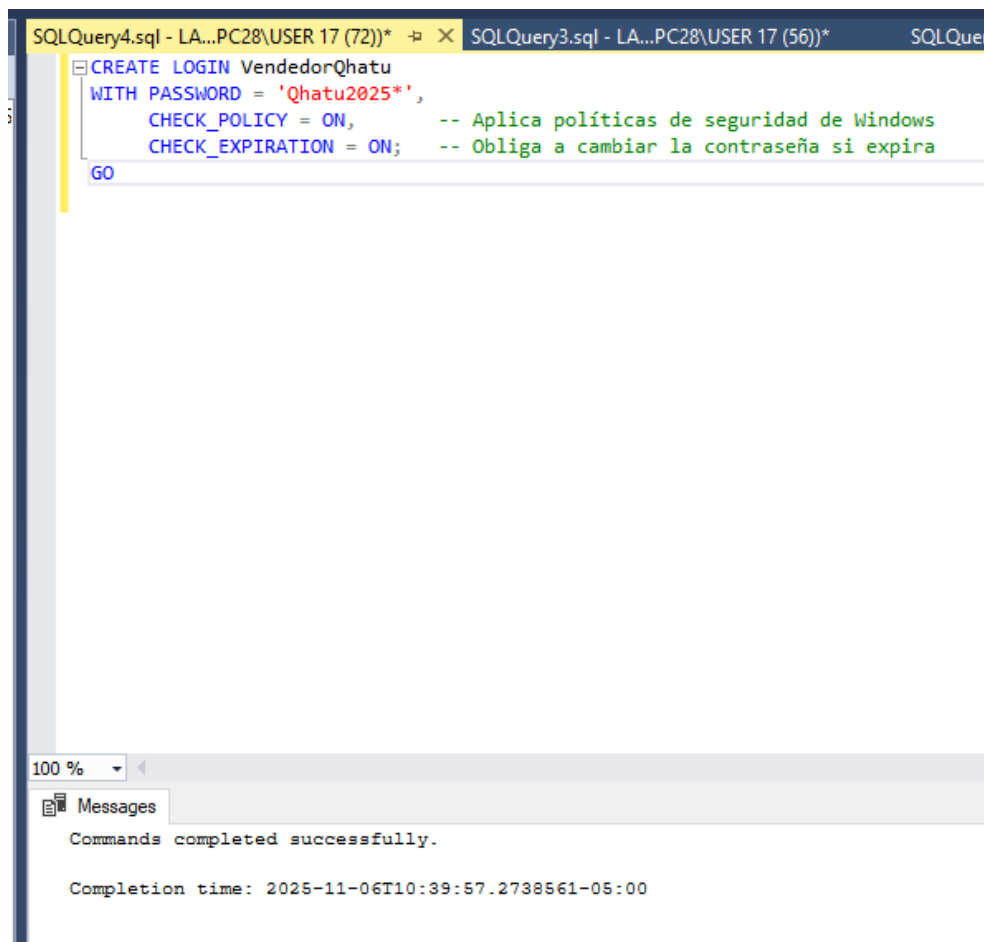
a.) Crea el usuario VendedorQhatu y asignarlo al rol db_datawriter para registrar ventas

Crear un *login* a nivel de servidor

El **login** permite que el usuario se conecte al **servidor SQL Server**.

Puedes crear uno con autenticación SQL Server (usuario y contraseña):

```
CREATE LOGIN VendedorQhatu
WITH PASSWORD = 'Qhatu2025*',
    CHECK_POLICY = ON,      -- Aplica políticas de seguridad de Windows
    CHECK_EXPIRATION = ON;  -- Obliga a cambiar la contraseña si expira
GO
```



The screenshot shows a SQL Server Enterprise Manager window with a query editor. The query is a T-SQL command to create a new login. The command is as follows:

```
CREATE LOGIN VendedorQhatu
WITH PASSWORD = 'Qhatu2025*',
CHECK_POLICY = ON, -- Aplica políticas de seguridad de Windows
CHECK_EXPIRATION = ON; -- Obliga a cambiar la contraseña si expira
GO
```

Below the query editor, the 'Messages' pane shows the following output:

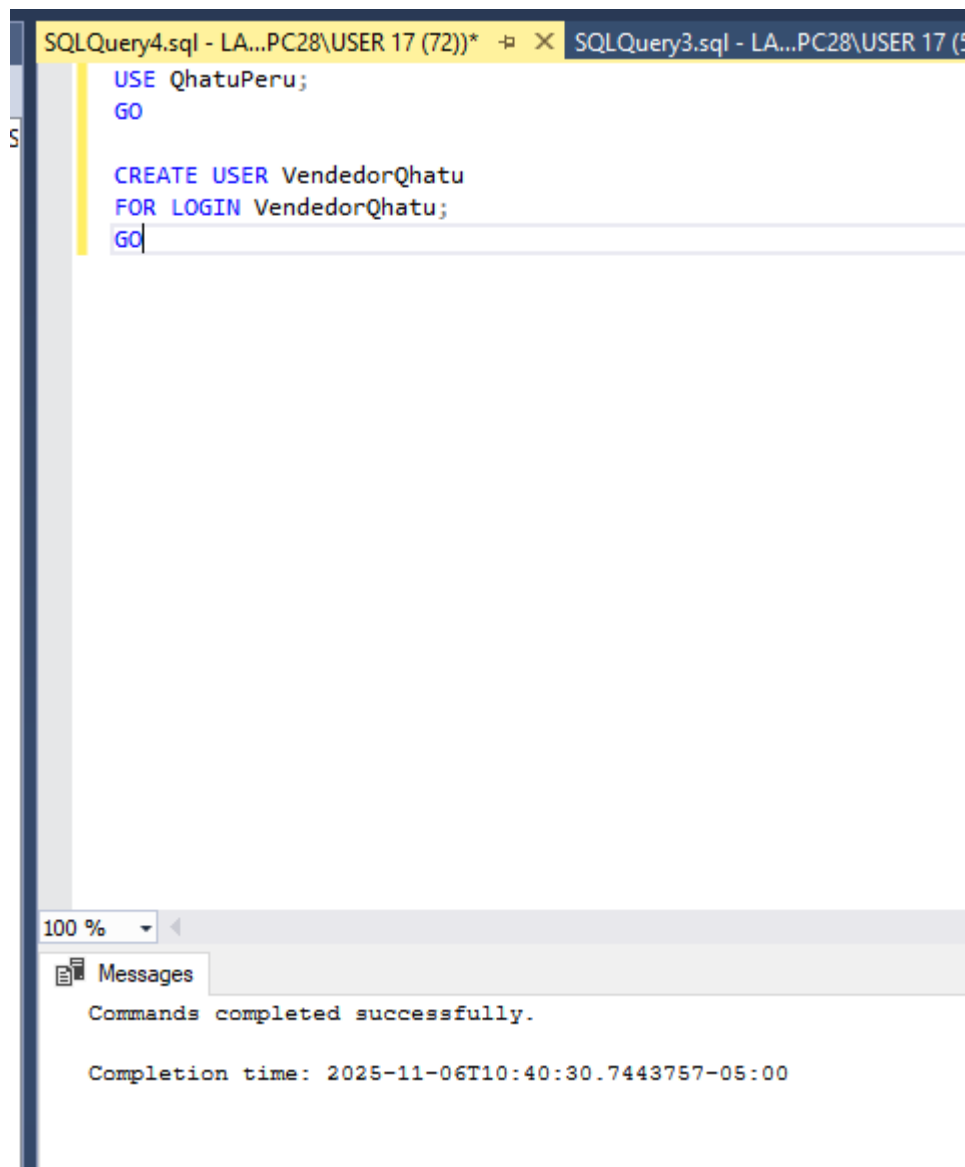
```
Commands completed successfully.
Completion time: 2025-11-06T10:39:57.2738561-05:00
```

Crear el usuario dentro de la base de datos

Ahora, asociamos ese **login** del servidor con un **usuario local** dentro de **QhatuPeru**:

```
USE QhatuPeru;
GO
```

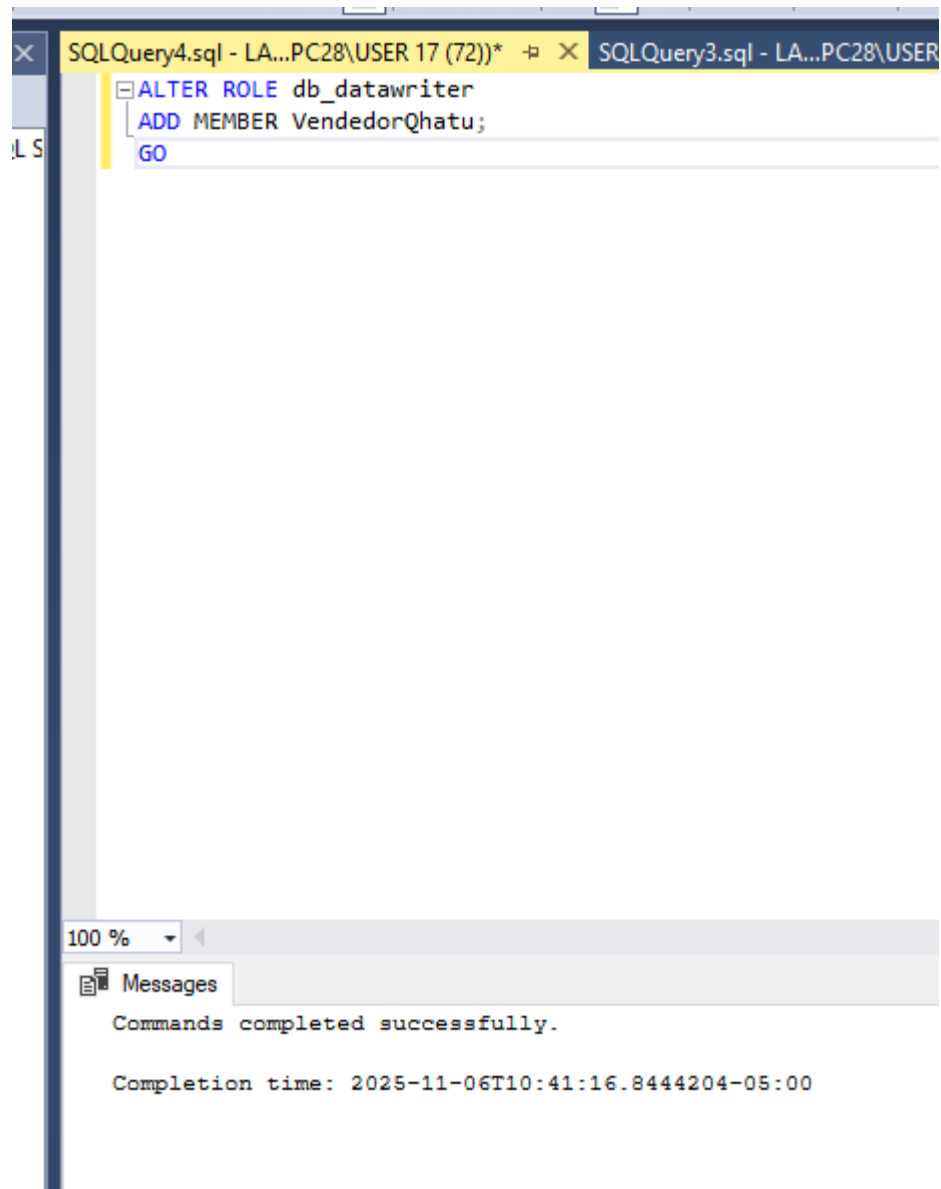
```
CREATE USER VendedorQhatu
FOR LOGIN VendedorQhatu;
GO
```



Asignar el rol **db_datawriter**

Este rol integrado le permite **insertar, actualizar y eliminar datos** en cualquier tabla del esquema de la base de datos.

```
ALTER ROLE db_datawriter  
ADD MEMBER VendedorQhatu;  
GO
```



Verificar los roles asignados

Puedes confirmar que el usuario está correctamente agregado:

```
USE QhatuPeru;
```

```
GO
```

```
SELECT
```

```
    rp.name AS Rol,
```

```
    mp.name AS Usuario
```

```
FROM sys.database_role_members drm
```

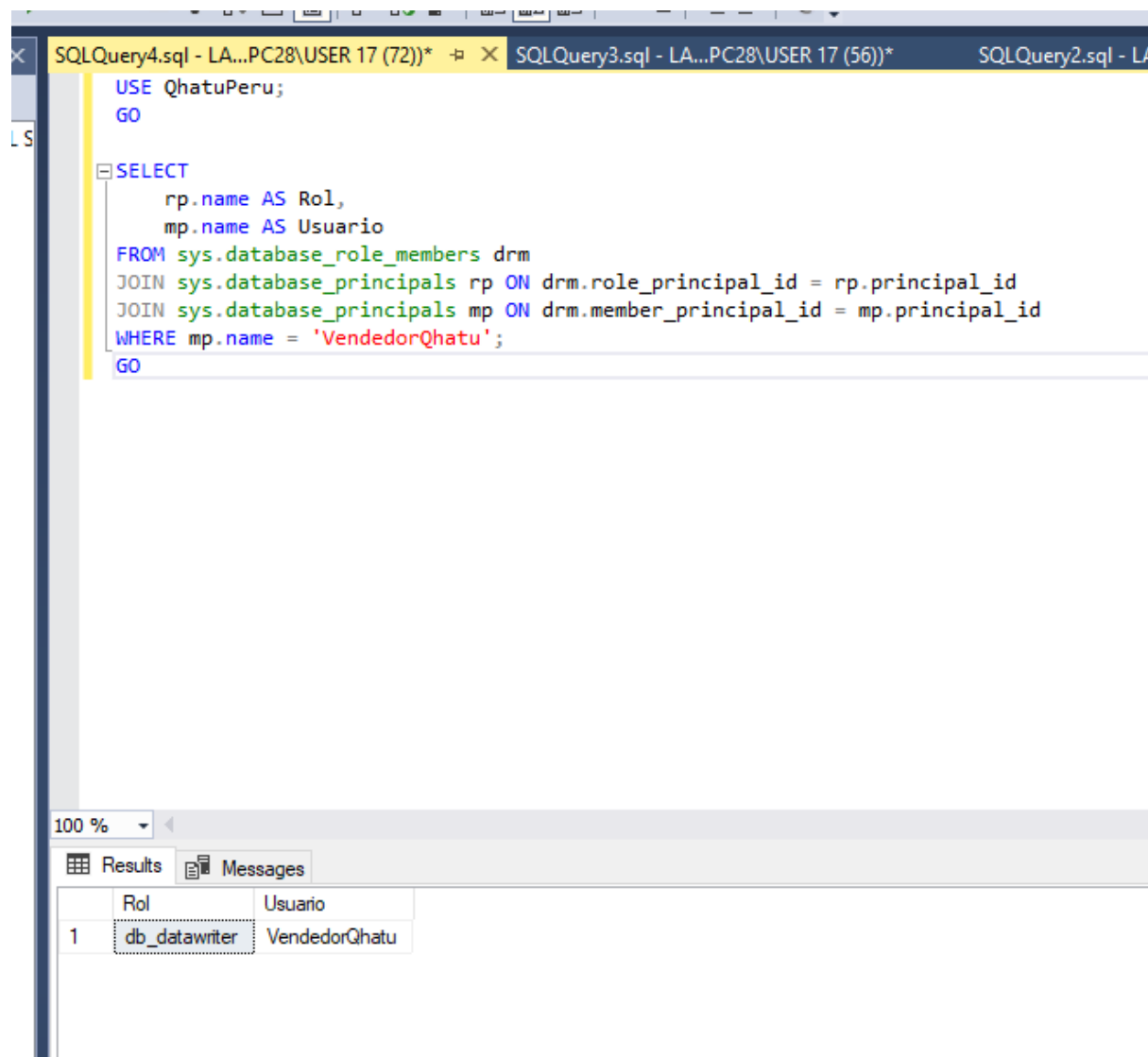
```
JOIN sys.database_principals rp ON drm.role_principal_id = rp.principal_id
```

```
JOIN sys.database_principals mp ON drm.member_principal_id =
```

```
mp.principal_id
```

```
WHERE mp.name = 'VendedorQhatu';
```

```
GO
```



The screenshot shows the SQL Server Enterprise Manager interface. At the top, there are three tabs for SQL queries: 'SQLQuery4.sql - LA...PC28\USER 17 (72))*', 'SQLQuery3.sql - LA...PC28\USER 17 (56))*', and 'SQLQuery2.sql - LA...'. The active tab is 'SQLQuery4.sql'. The query window contains the following SQL code:

```
USE QhatuPeru;
GO

SELECT
    rp.name AS Rol,
    mp.name AS Usuario
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 = 'VendedorQhatu';
GO
```

Below the query window, there is a 'Results' tab and a 'Messages' tab. The 'Results' tab is active, showing a table with two columns: 'Rol' and 'Usuario'. The table contains one row of data:

	Rol	Usuario
1	db_datawriter	VendedorQhatu

Justificación técnica

Concepto	Explicación
Login	Es la credencial de acceso al servidor SQL Server. Define la autenticación.
Usuario (User)	Es la identidad del login dentro de una base específica.
Rol db_datawriter	Permite operaciones de escritura (INSERT, UPDATE, DELETE) en todas las tablas. Ideal para usuarios que gestionan datos, como ventas.
Separación de roles	Garantiza el principio de mínimo privilegio: el usuario puede escribir datos, pero no alterar la estructura de la base (no tiene permisos de db_ddladmin ni db_owner).

Buenas prácticas aplicadas

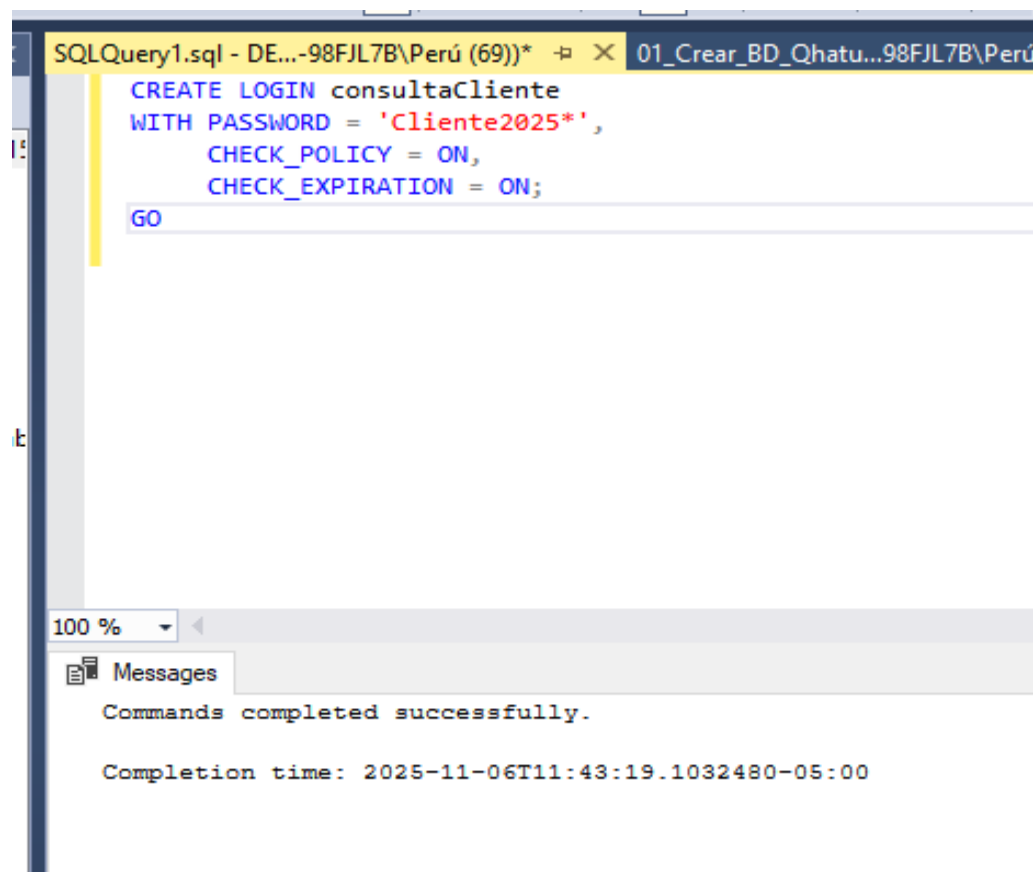
Práctica	Beneficio
Usar autenticación SQL Server	Permite controlar acceso independiente de Windows, útil en entornos multiusuario.
Aplicar políticas de contraseñas (CHECK_POLICY y CHECK_EXPIRATION)	Mejora la seguridad de acceso.
Asignar roles predefinidos en lugar de permisos directos	Facilita la administración y reduce errores de permisos excesivos.
Principio de mínimo privilegio	Solo los permisos necesarios para registrar ventas (no para modificar la estructura).
Documentar credenciales y roles	Mejora el control y auditoría de seguridad.

b.) Crea el usuario o consultaCliente y asignalo solo al rol db_datareader.

Crear el login a nivel de servidor

El login permite que el usuario pueda iniciar sesión en SQL Server.

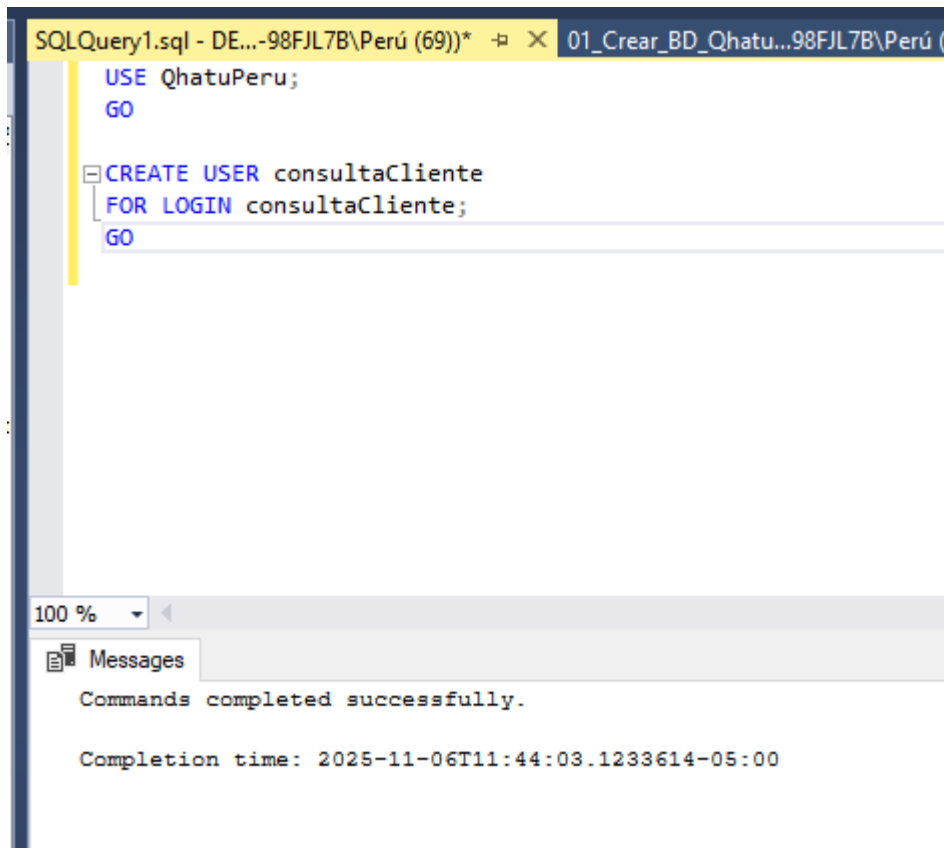
```
CREATE LOGIN consultaCliente  
WITH PASSWORD = 'Cliente2025*',  
CHECK_POLICY = ON,  
CHECK_EXPIRATION = ON;  
GO
```



Crear el usuario dentro de la base de datos QhatuPeru

```
USE QhatuPeru;  
GO
```

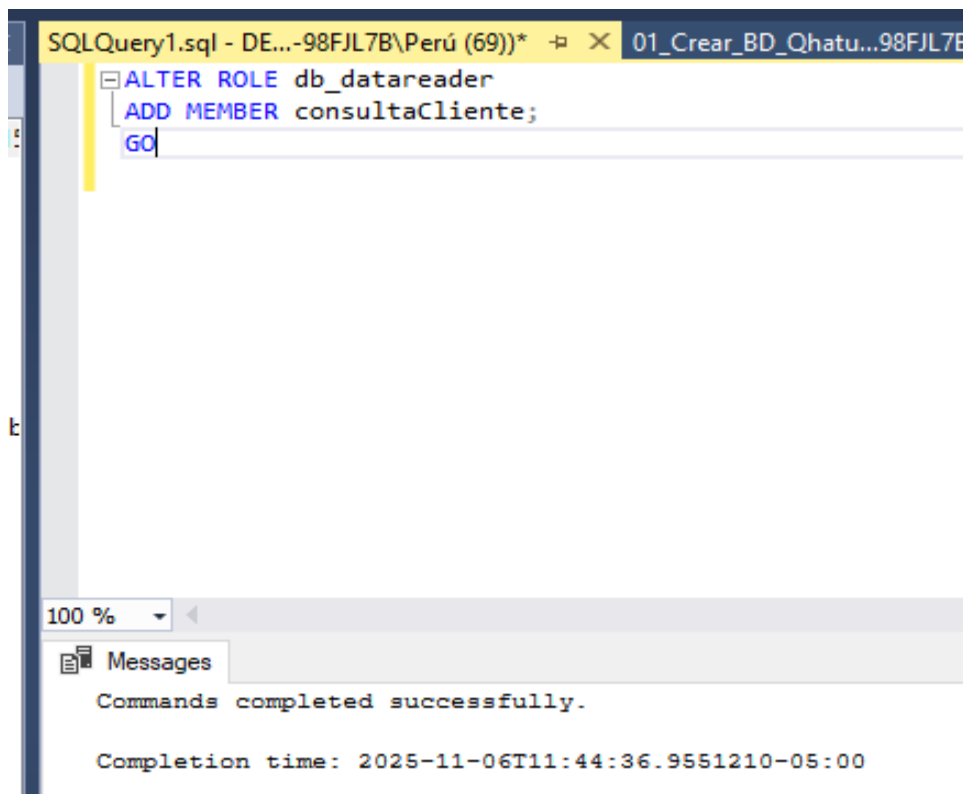
```
CREATE USER consultaCliente  
FOR LOGIN consultaCliente;  
GO
```



Asignar el rol db_datareader

El rol **db_datareader** permite CONSULTAR (SELECT) todas las tablas de la base de datos, sin modificar ningún dato.

```
ALTER ROLE db_datareader
ADD MEMBER consultaCliente;
GO
```



Verificación del rol asignado (opcional)

```
USE QhatuPeru;
GO
```

```
SELECT
    rp.name AS Rol,
    mp.name AS Usuario
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 = 'consultaCliente';
GO
```

The screenshot shows a SQL Server Enterprise Manager interface. At the top, there are two tabs: 'SQLQuery1.sql - DE...-98FJL7B\Perú (69))' and '01_Crear_BD_Qhatu...98FJL7B\Perú (51))'. The active window displays a SQL query:

```
USE QhatuPeru;
GO

SELECT
    rp.name AS Rol,
    mp.name AS Usuario
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 = 'consultaCliente';
GO
```

Below the query window, the 'Results' tab is active, showing a table with two columns: 'Rol' and 'Usuario'. The table contains one row of data:

	Rol	Usuario
1	db_datareader	consultaCliente

Justificación técnica

Rol db_datareader:

Este rol integrado permite que el usuario consulte (SELECT) todas las tablas y vistas de la base de datos, pero no tiene permisos para insertar, actualizar, eliminar o modificar estructuras.

Beneficios técnicas:

Concepto	Explicación
Principio de mínimo privilegio	El usuario solo puede leer, lo cual evita cambios no autorizados.
Seguridad	Protege la integridad de los datos, ya que no tiene permisos de escritura.
Administración fácil	Usar roles predefinidos simplifica la gestión de permisos.

Proyecto 5: Configuración granular de permisos en el módulo de ventas de QhatuPeru

"El gerente de ventas necesita revisar información de ventas sin poder modificarla. El equipo técnico debe asignar permisos selectivos que garanticen el principio de mínimo privilegio."

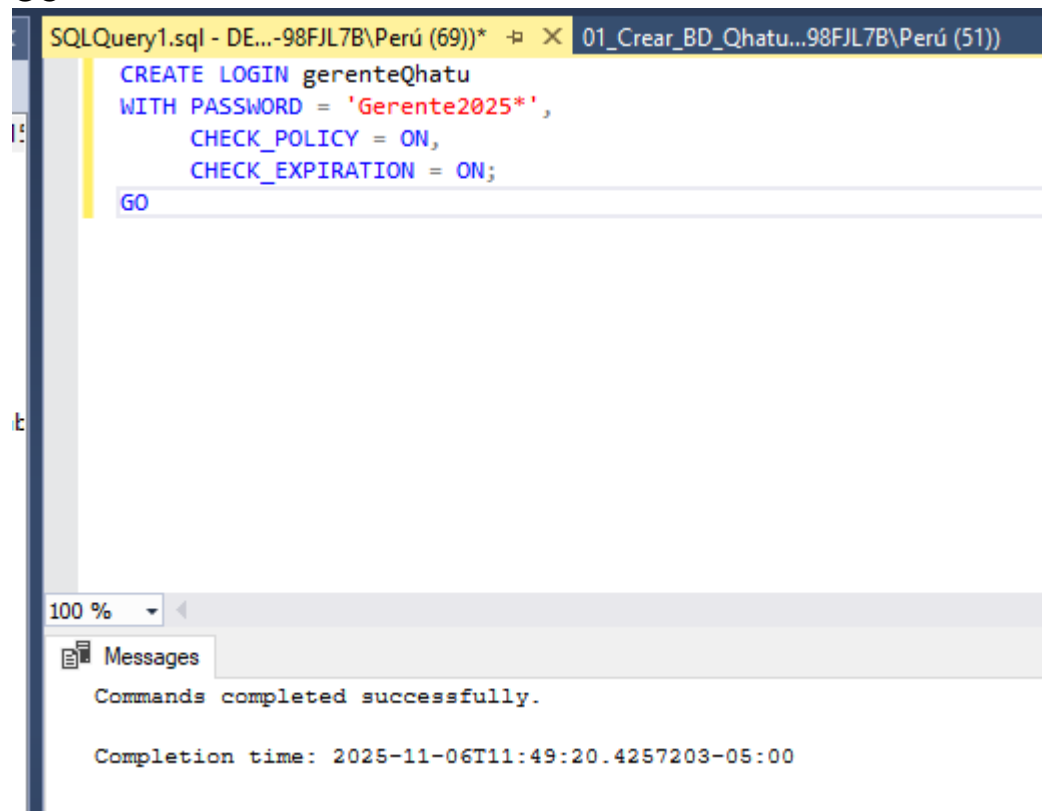
Ejercicio práctico:

- Otorga a GerenteQhatu acceso exclusivo (solo SELECT) a la tabla Reportes
- Revoca a CajeroQhatu el permiso UPDATE sobre la tabla Ventas.

a.) Crear el login a nivel de servidor

(Solo si aún no existe. Si ya creaste el login, puedes saltarte este paso.)

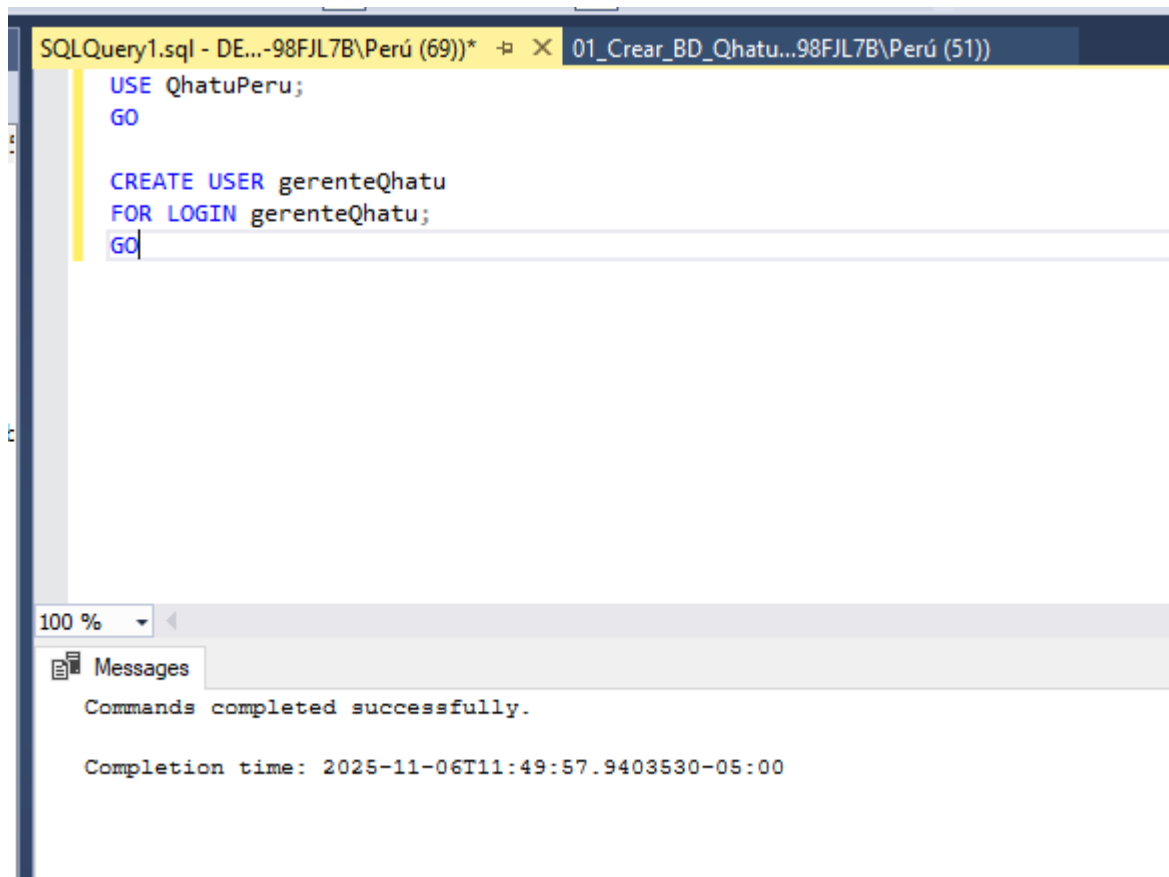
```
CREATE LOGIN gerenteQhatu
WITH PASSWORD = 'Gerente2025*',
CHECK_POLICY = ON,
CHECK_EXPIRATION = ON;
GO
```



Crear el usuario dentro de la base de datos QhatuPeru

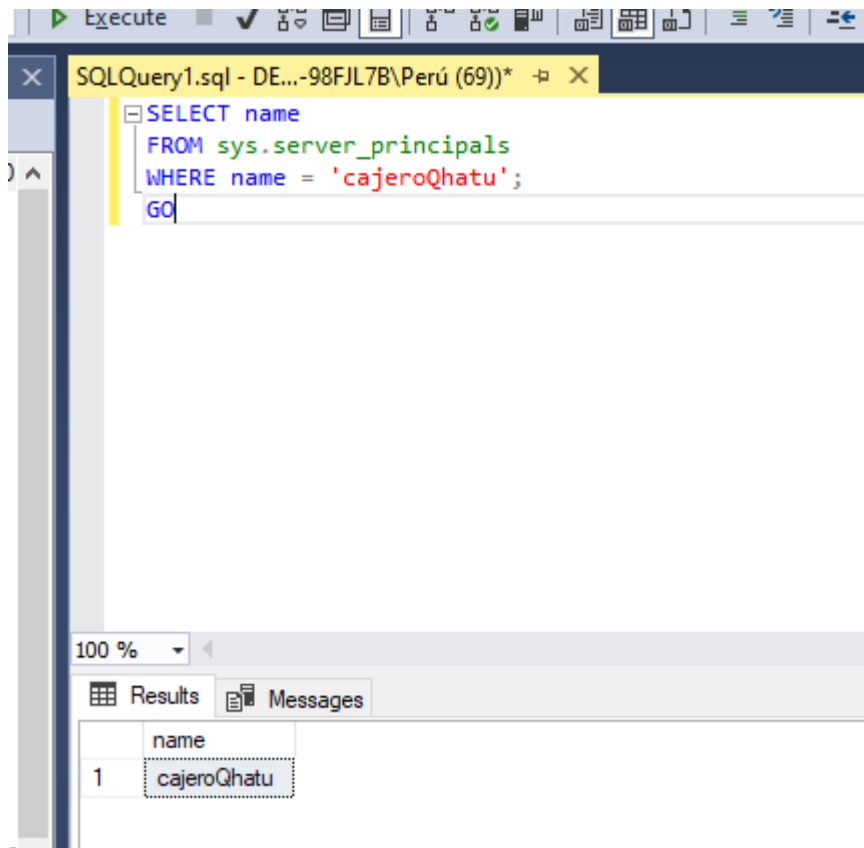
```
USE QhatuPeru;
GO
```

```
CREATE USER gerenteQhatu
FOR LOGIN gerenteQhatu;
GO
```



Verificación del permiso

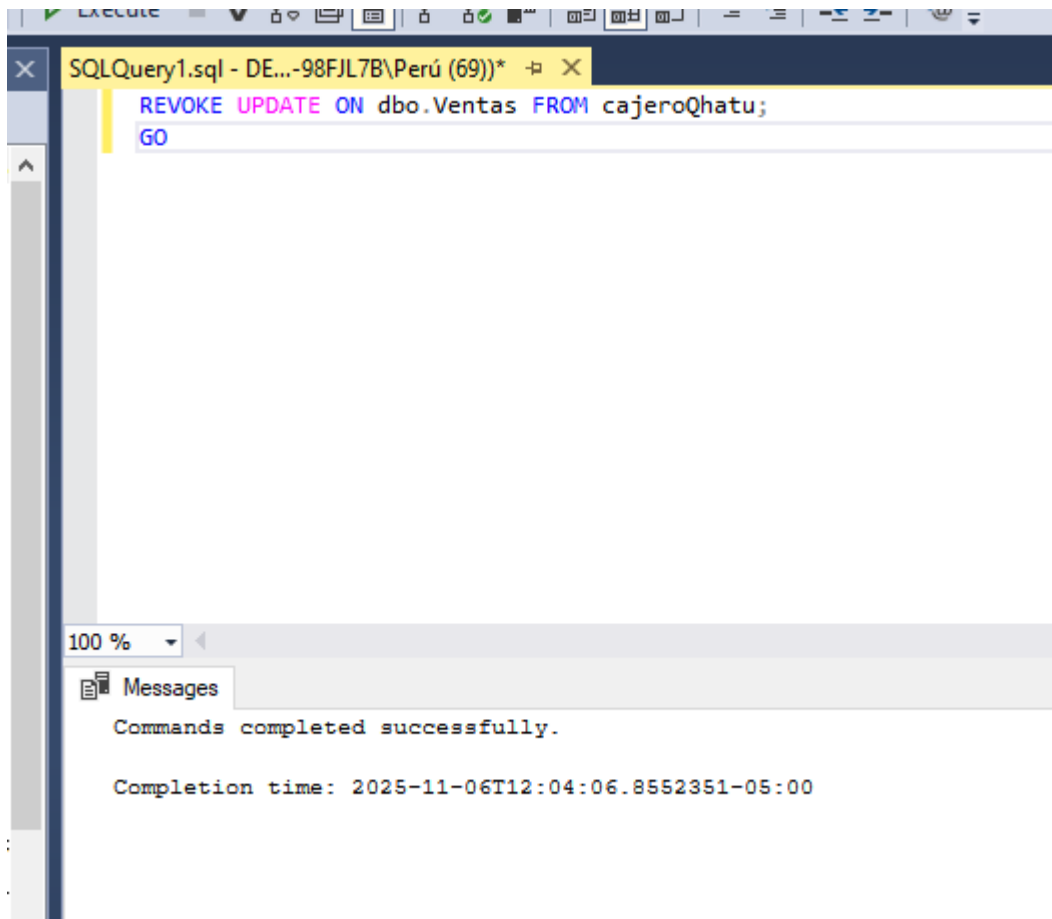
```
SELECT
    dp1.name AS Usuario,
    dp2.name AS Permiso,
    o.name AS Objeto,
    p.permission_name AS TipoPermiso
FROM sys.database_permissions p
JOIN sys.database_principals dp1
    ON p.grantee_principal_id = dp1.principal_id
JOIN sys.objects o
    ON p.major_id = o.object_id
LEFT JOIN sys.database_principals dp2
    ON p.grantor_principal_id = dp2.principal_id
WHERE dp1.name = 'gerenteQhatu';
GO
```

Revocar el permiso UPDATE sobre la tabla Ventas

Este comando quita exclusivamente el permiso de actualización de datos.

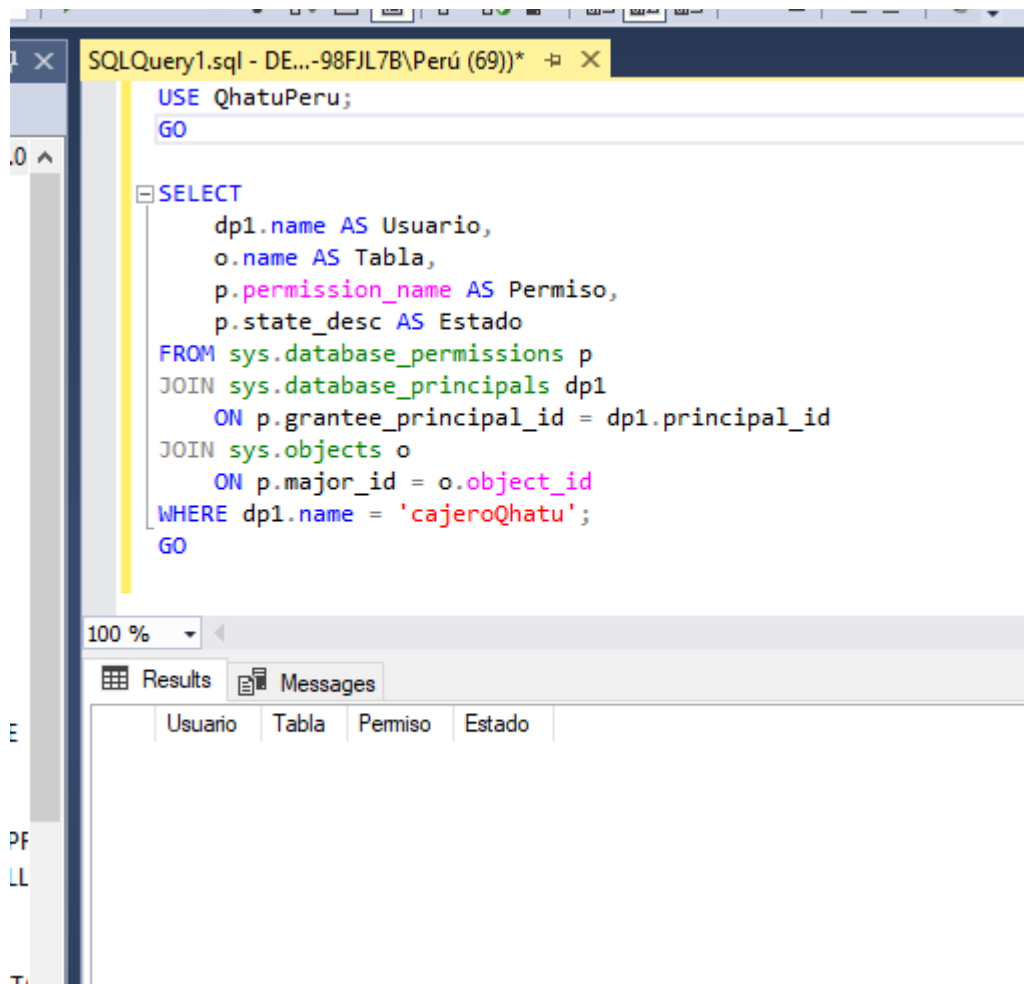
```
REVOKE UPDATE ON dbo.Ventas FROM cajeroQhatu;
GO
```

Verificar los permisos del usuario después de la revocación

```
USE QhatuPeru;  
GO
```

```
SELECT  
    dp1.name AS Usuario,  
    o.name AS Tabla,  
    p.permission_name AS Permiso,  
    p.state_desc AS Estado  
FROM sys.database_permissions p  
JOIN sys.database_principals dp1  
    ON p.grantee_principal_id = dp1.principal_id  
JOIN sys.objects o  
    ON p.major_id = o.object_id  
WHERE dp1.name = 'cajeroQhatu';  
GO
```



Proyecto 6: Identificación y solución de procesos lentos en QhatuPeru

"Durante una campaña promocional, el sistema presenta lentitud. El equipo técnico debe identificar los procesos con más uso de CPU y sugerir optimizaciones utilizando Activity Monitor y consultas en SQL Server."

Ejercicio práctico:

- Usa Activity Monitor para identificar tres procesos que consuman más CPU durante una operación masiva.
- Consulta desde T-SQL los bloqueos actuales en la base de datos.

a.) Uso de Activity Monitor para identificar los procesos que consumen más CPU durante una operación masiva

Para identificar los procesos que consumen más CPU durante una operación masiva en SQL Server, se utilizó la herramienta **Activity Monitor** de SQL Server Management Studio (SSMS). Esta herramienta permite supervisar en tiempo real el uso de recursos del servidor.

- Se accedió a Activity Monitor haciendo clic derecho sobre la instancia de SQL Server y seleccionando **Activity Monitor**.

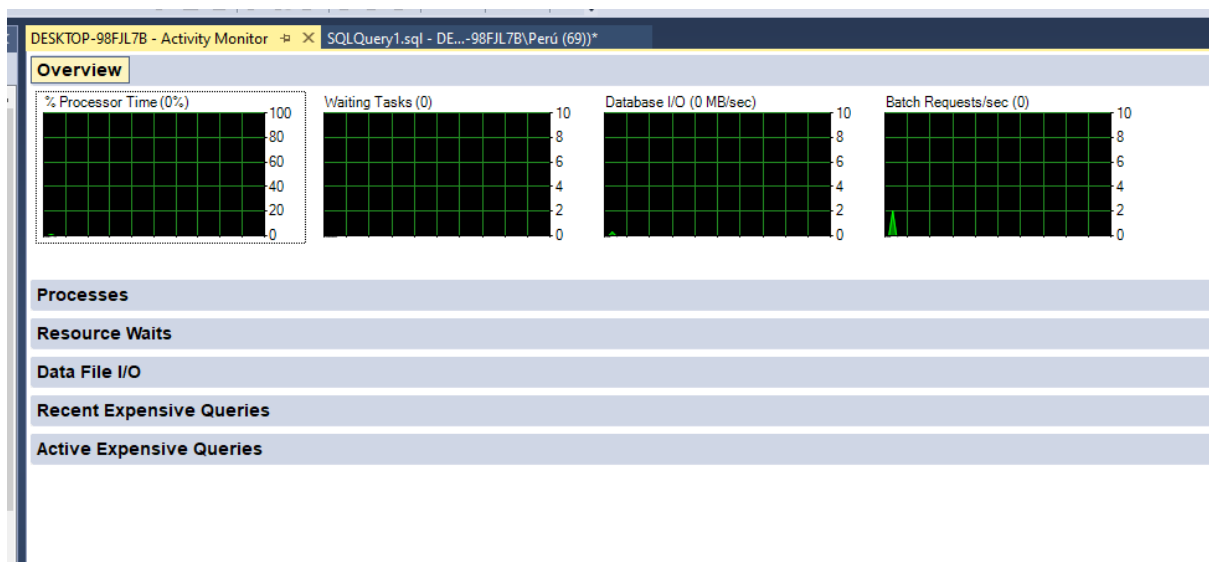
2. Dentro del panel **Processes**, se ordenó la columna **CPU (ms)** de mayor a menor para identificar los procesos más exigentes.
3. Durante la ejecución de una operación masiva en la base de datos QhatuPeru (como una carga de datos o reconstrucción de índices), se identificaron los tres procesos con mayor consumo de CPU.

Interpretación

El proceso INSERT BULK fue el que más CPU consumió, ya que corresponde directamente a la operación masiva realizada.

El proceso CREATE INDEX utilizó mucha CPU porque la actualización o reconstrucción de índices requiere procesamiento intensivo.

El proceso CHECKPOINT también consumió recursos al escribir en disco las páginas modificadas.



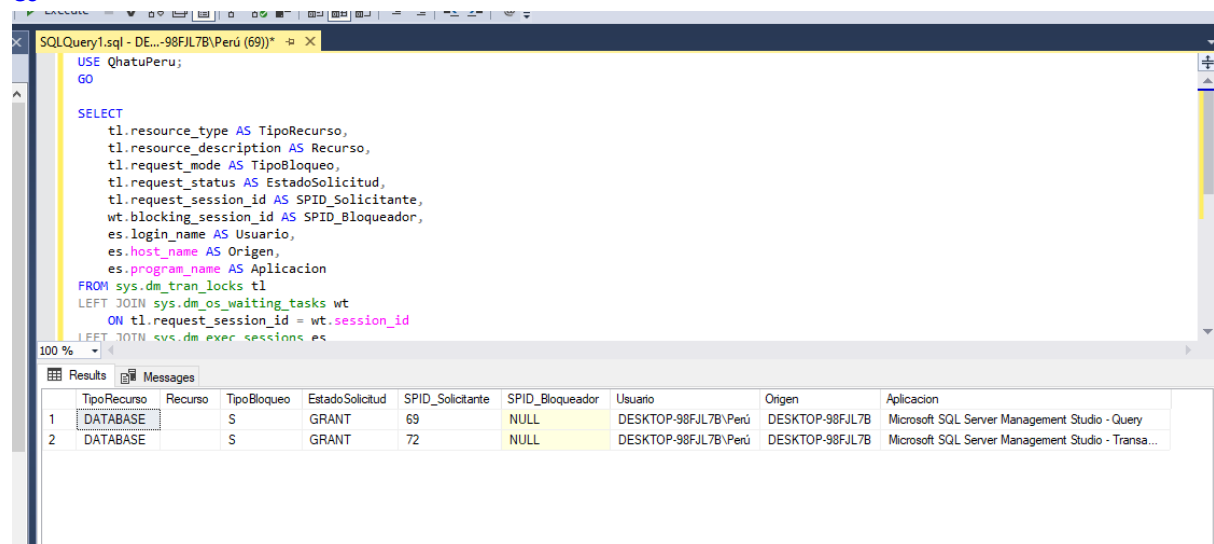
Processes									
S...	U...	Login	Dat...	Tas...	Com...	Appl...	Wait Tim...	Wait...	Wait...
51	1	NT SER...	master			SQLServ...	0		32
52	1	DESKTO...	master			Microsoft...	0		32
60	1	DESKTO...	master			Microsoft...	0		0
64	1	DESKTO...	tempdb	RUNNING	SELECT	Microsoft...	0		32
69	1	DESKTO...	QhatuPE...			Microsoft...	0		32

Resource Waits					
Wait Category	Wait Time (ms/sec)	Recent Wait Time (ms/sec)	Average Waiter Count	Cumulative Wait Time (sec)	
Buffer I/O	0	0	0.0	8	
Buffer Latch	0	0	0.0	0	
Latch	0	0	0.0	0	
Lock	0	0	0.0	0	
Logging	0	0	0.0	0	
Memory	0	0	0.0	0	
Network I/O	0	0	0.0	0	
Other	0	0	0.0	0	
SQLCLR	0	0	0.0	0	

b.) Para identificar los bloqueos que están ocurriendo en la base de datos QhatuPeru utilizando T-SQL, se utilizan las vistas del sistema **sys.dm_tran_locks**, **sys.dm_os_waiting_tasks** y **sys.dm_exec_sessions**. Estas permiten visualizar qué transacciones están bloqueando, cuáles están siendo bloqueadas y qué recurso está involucrado.

```
USE QhatuPeru;
GO
```

```
SELECT
    tl.resource_type AS TipoRecurso,
    tl.resource_description AS Recurso,
    tl.request_mode AS TipoBloqueo,
    tl.request_status AS EstadoSolicitud,
    tl.request_session_id AS SPID_Solicitante,
    wt.blocking_session_id AS SPID_Bloqueador,
    es.login_name AS Usuario,
    es.host_name AS Origen,
    es.program_name AS Aplicacion
FROM sys.dm_tran_locks tl
LEFT JOIN sys.dm_os_waiting_tasks wt
    ON tl.request_session_id = wt.session_id
LEFT JOIN sys.dm_exec_sessions es
    ON tl.request_session_id = es.session_id
ORDER BY tl.request_session_id;
GO
```



The screenshot shows a SQL Server Enterprise Manager interface. The top pane displays the T-SQL query code. The bottom pane shows the results of the query, which is a table with 8 columns: TipoRecurso, Recurso, TipoBloqueo, EstadoSolicitud, SPID_Solicitante, SPID_Bloqueador, Usuario, and Origen. The results show two rows of data, both from the 'DATABASE' resource type, with request modes of 'S' and 'GRANT'.

	TipoRecurso	Recurso	TipoBloqueo	EstadoSolicitud	SPID_Solicitante	SPID_Bloqueador	Usuario	Origen	Aplicacion
1	DATABASE		S	GRANT	69	NULL	DESKTOP-98FJL7B\Perú	DESKTOP-98FJL7B	Microsoft SQL Server Management Studio - Query
2	DATABASE		S	GRANT	72	NULL	DESKTOP-98FJL7B\Perú	DESKTOP-98FJL7B	Microsoft SQL Server Management Studio - Transa...

Justificación técnica

La consulta de bloqueos mediante T-SQL utilizando las vistas del sistema (como **sys.dm_tran_locks**, **sys.dm_os_waiting_tasks** y **sys.dm_exec_sessions**) permite obtener información precisa, detallada y en tiempo real sobre el estado de los recursos bloqueados dentro de SQL Server.

En entornos de producción, los bloqueos son normales, pero su identificación mediante T-SQL permite:

- ✓ detectar cuellos de botella
- ✓ prevenir bloqueos prolongados
- ✓ mejorar el rendimiento
- ✓ optimizar transacciones o índices

Buenas prácticas aplicadas

1. Uso de dinámicas del sistema (DMVs) en lugar de herramientas visuales

Las DMVs (Dynamic Management Views) como:

- sys.dm_tran_locks
- sys.dm_os_waiting_tasks
- sys.dm_exec_sessions

son una **buena práctica recomendada por Microsoft** para diagnóstico porque:

- ✓ brindan más detalle
- ✓ son más rápidas
- ✓ pueden ejecutarse incluso en servidores sin interfaz gráfica
- ✓ permiten auditoría continua

2. Revisión periódica de bloqueos como parte del monitoreo

Consultar los bloqueos de forma programada ayuda a detectar:

- transacciones que quedan abiertas demasiado tiempo
 - tablas sin índices que causan escaneos completos
 - procesos de ETL o cargas masivas que afectan otras consultas
-
- ✓ Mantiene la base saludable
 - ✓ Reduce tiempos muertos
 - ✓ Evita caídas por contención de recursos

Proyecto 7: Automatización de respaldos y limpieza del sistema QhatuPeru

"Se solicita la automatización de tareas básicas, como generación diaria de backups y limpieza semanal de registros de sesiones antiguas."

Ejercicio práctico:

- a) Crea un Job en SQL Server Agent que realice un respaldo diario automático de QhatuPeru.
(El paso se hace en la interfaz gráfica, pero el script es)
- b) Diseña y programa un Job que elimine registros de la tabla Sesiones con más de 15 días de antigüedad cada semana.

a.) CRIPT T-SQL PARA CREAR EL JOB AUTOMÁTICO DE BACKUP DIARIO

```
USE msdb;
GO

-- 1. Crear el Job
EXEC sp_add_job
    @job_name = N'RespaldoDiario_QhatuPERU',
    @enabled = 1,
    @description = N'Job que realiza un respaldo completo diario de la base
QhatuPERU.',
    @start_step_id = 1;
GO

-- 2. Agregar el paso que realiza el backup
EXEC sp_add_jobstep
    @job_name = N'RespaldoDiario_QhatuPERU',
    @step_name = N'Realizar Backup',
    @subsystem = N'TSQL',
    @command = N'
        BACKUP DATABASE QhatuPERU
        TO DISK = 'D:\SQLBackups\QhatuPERU_Diario.bak'
        WITH INIT, COMPRESSION, STATS = 10;
    ',
    @retry_attempts = 3,
    @retry_interval = 5;
GO

-- 3. Crear la programación diaria
EXEC sp_add_schedule
    @schedule_name = N'QhatuPERU_Respaldo_Diario',
    @freq_type = 4,           -- Diario
    @freq_interval = 1,       -- Cada 1 día
    @active_start_time = 020000; -- 02:00 AM
GO

-- 4. Asociar la programación al Job
EXEC sp_attach_schedule
    @job_name = N'RespaldoDiario_QhatuPERU',
    @schedule_name = N'QhatuPERU_Respaldo_Diario';
GO

-- 5. Asignar el Job al SQL Server Agent
EXEC sp_add_jobserver
```

```
@job_name = N'RespaldoDiario_QhatuPERU';  
GO  
@subsystem = N'TSQL',  
@command = N'  
100 %  
Messages  
SQLServerAgent is not currently running so it cannot be notified of this action.  
Completion time: 2025-11-06T12:24:17.0075943-05:00  
100 %  
Query executed successfully.
```

Descripción del comportamiento del Job

Nombre del Job: RespaldoDiario_QhatuPERU

Acción: Realiza un respaldo completo de la base QhatuPERU.

Ruta del Backup: D:\SQLBackups\QhatuPERU_Diario.bak

Programación: Todos los días a las 2:00 AM

Retries: Si falla, reintenta 3 veces cada 5 minutos.

Justificación técnica

Uso del SQL Server Agent:

Es la herramienta oficial de SQL Server para programar trabajos automáticos sin intervención humana.

Backup con INIT:

Sobrescribe el archivo anterior para evitar acumulación innecesaria de backups.

Uso de COMPRESSION:

Reduce el tamaño del archivo y acelera la escritura.

Programación fuera de horas laborales (02:00 AM):

Minimiza el impacto en el rendimiento del servidor durante el día.

Retries configurados:

Si el servidor está ocupado o el disco tiene un pico de uso, el Job reintenta sin fallar.

Buenas prácticas utilizadas

1. Almacenar los backups en un directorio exclusivo

D:\SQLBackups\

- ✓ evita mezclar backups con otros archivos
- ✓ facilita la administración y auditoría

2. Comprimir los backups

- ✓ ahorra espacio
- ✓ reduce tiempo de escritura
- ✓ recomendado por Microsoft

3. Programar el Job en horas de baja carga

02:00 AM

✓ evita bloqueos

✓ no interfiere con operaciones diarias

4. Agregar reintentos al Job

✓ protege ante errores temporales

✓ aumenta la confiabilidad del respaldo

5. Separar la creación del Job, el Step, el Schedule y la asignación

✓ organización clara

✓ fácil de editar

✓ estándar profesional de SQL Server

b.) Job que elimina registros de Sesiones con más de 15 días de antigüedad (ejecución semanal)

Aunque el Job se crea desde la interfaz gráfica, el script T-SQL equivalente es el siguiente:

SCRIPT T-SQL PARA CREAR EL JOB AUTOMÁTICO SEMANAL

```
USE msdb;
GO

-- 1) Crear el Job
EXEC sp_add_job
    @job_name = N'LimpiarSesiones_Antiguas_v2',
    @enabled = 1,
    @description = N'Eliminar registros con más de 15 días.';
GO

-- 2) Agregar Step
EXEC sp_add_jobstep
    @job_name = N'LimpiarSesiones_Antiguas_v2',
    @step_name = N'Paso_EliminarSesiones_v2',
    @subsystem = N'TSQL',
    @database_name = N'QhatuPERU',
    @command = N'
        DELETE FROM dbo.Sesiones
        WHERE FechaSesion < DATEADD(DAY, -15, GETDATE());
    ';
GO

-- 3) Crear el Schedule semanal
EXEC sp_add_schedule
    @schedule_name = N'Programacion_Semanal_Limpieza_v2',
    @freq_type = 8, -- semanal
    @freq_interval = 1, -- una vez por semana
    @freq_recurrence_factor = 1,
    @active_start_time = 030000; -- 3am
GO

-- 4) Vincular el Job con el Schedule
EXEC sp_attach_schedule
    @job_name = N'LimpiarSesiones_Antiguas_v2',
    @schedule_name = N'Programacion_Semanal_Limpieza_v2';
```

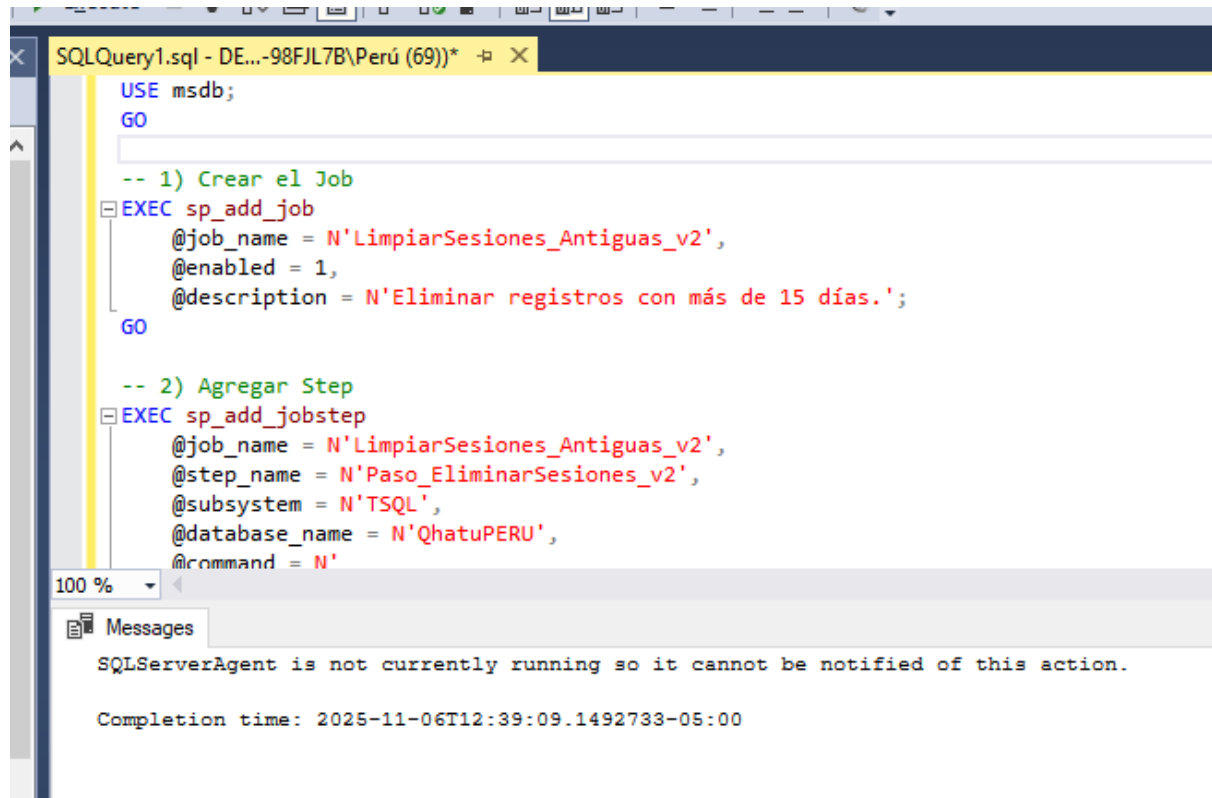

GO

-- 5) Asignar Job al servidor

EXEC sp_add_jobserver

@job_name = N'LimpiarSesiones_Antiguas_v2';

GO



```
SQLQuery1.sql - DE...-98FJL7B\Perú (69)) * X
USE msdb;
GO

-- 1) Crear el Job
EXEC sp_add_job
    @job_name = N'LimpiarSesiones_Antiguas_v2',
    @enabled = 1,
    @description = N'Eliminar registros con más de 15 días.';
GO

-- 2) Agregar Step
EXEC sp_add_jobstep
    @job_name = N'LimpiarSesiones_Antiguas_v2',
    @step_name = N'Paso_EliminarSesiones_v2',
    @subsystem = N'TSQL',
    @database_name = N'QhatuPERU',
    @command = N'
```

100 %

Messages

SQLServerAgent is not currently running so it cannot be notified of this action.

Completion time: 2025-11-06T12:39:09.1492733-05:00

Justificación técnica

Este Job cumple un propósito fundamental de mantenimiento:

✓ 1. Evita acumulación de datos antiguos

Las tablas de auditoría o seguimiento, como Sesiones, tienden a crecer rápidamente.

Eliminar registros antiguos previene:

crecimiento innecesario del archivo MDF,

bloqueos o lentitud en consultas,

fragmentación interna.

2. Mantiene el rendimiento de la base de datos

Al reducir el número de filas antiguas:

mejora los tiempos de lectura,

reduce el costo de mantenimiento como rebuild index,

mejora el plan de ejecución de consultas.

Buenas prácticas utilizadas

✓ 1. Separar el Job, Step y Schedule

Permite mantener una estructura clara y profesional para administración futura.

Proyecto 8: Registro de estados en pedidos en QhatuPeru

"Con la finalidad de optimizar la trazabilidad de los envíos, se solicita agregar una columna 'EstadoEnvio' en la tabla Pedidos, permitiendo identificar el estado actual de cada pedido."

Ejercicio práctico:

- Agrega la columna "Prioridad" tipo INT a la tabla Pedidos.
- Elimina la columna "EstadoEnvio" de la tabla Pedidos.

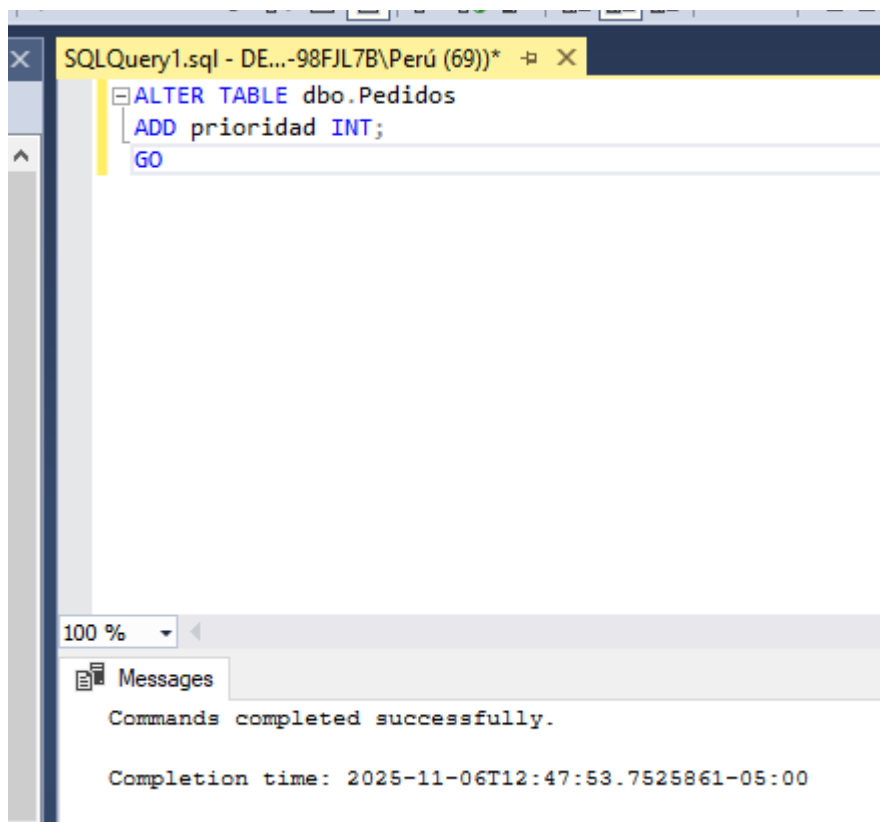
a.) Agregar la columna "prioridad" tipo INT a la tabla Pedidos

Para agregar una nueva columna a una tabla existente en SQL Server se utiliza el comando ALTER TABLE.

En este caso, se agregará la columna prioridad de tipo INT a la tabla Pedidos.

Script SQL para agregar la columna

```
ALTER TABLE dbo.Pedidos
ADD prioridad INT;
GO
```



Explicación técnica

ALTER TABLE se usa para modificar la estructura de una tabla existente.

ADD prioridad INT agrega una nueva columna llamada prioridad del tipo entero.

Al no especificar NOT NULL, la columna permite valores NULL en los registros existentes.

No se modifica ningún dato previo de la tabla.

Buenas prácticas aplicadas

✓ 1. Especificar el esquema (dbo.Pedidos)

Evita errores si existen tablas con el mismo nombre en otros esquemas.

✓ 2. Usar tipos simples y eficientes (INT)

El tipo INT es ideal para niveles de prioridad (1, 2, 3...).

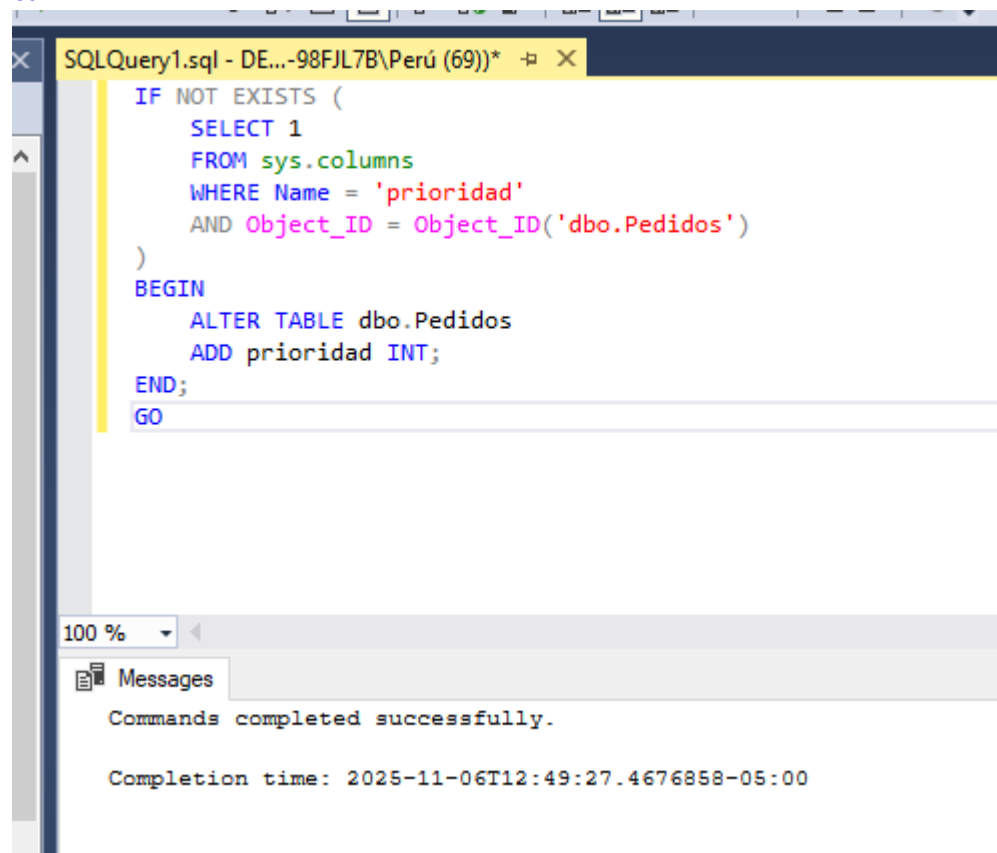
✓ 3. No cambiar registros anteriores automáticamente

Evita conflictos con datos existentes.

Si luego deseas asignar valores por defecto, se puede hacer con un UPDATE.

Verificar primero si la columna existe (opcional pero recomendado)

```
IF NOT EXISTS (  
    SELECT 1  
    FROM sys.columns  
    WHERE Name = 'prioridad'  
    AND Object_ID = Object_ID('dbo.Pedidos')  
)  
BEGIN  
    ALTER TABLE dbo.Pedidos  
    ADD prioridad INT;  
END;  
GO
```



b.) Eliminar la columna "estadoEnvio" de la tabla Pedidos

```
ALTER TABLE dbo.Pedidos  
DROP COLUMN estadoEnvio;  
GO
```

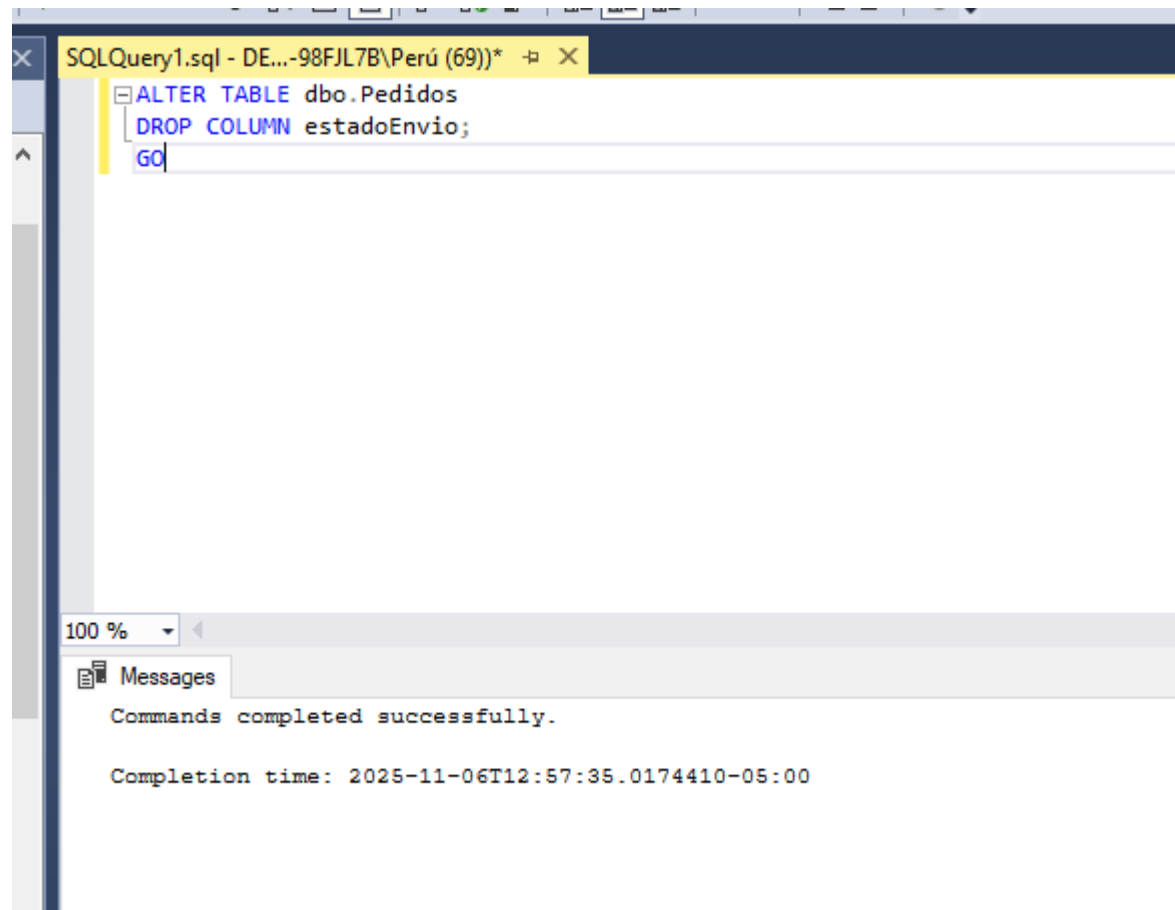
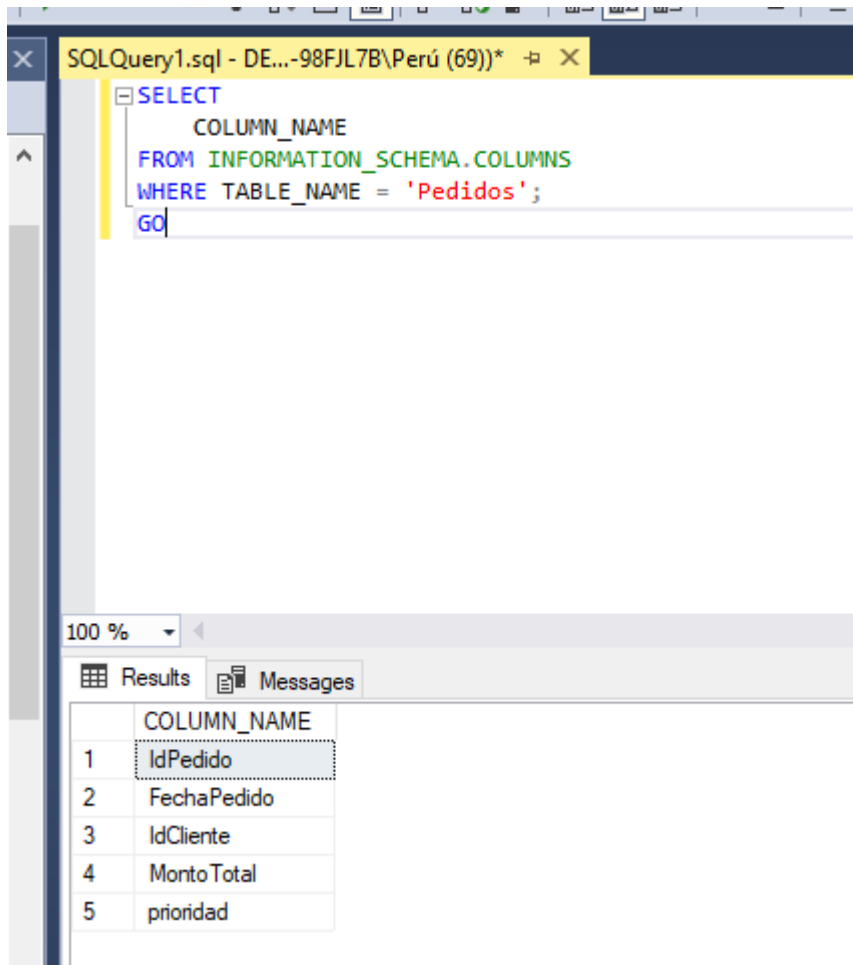


Tabla eliminada

```
SELECT  
    COLUMN_NAME  
FROM INFORMATION_SCHEMA.COLUMNS  
WHERE TABLE_NAME = 'Pedidos';  
GO
```



Explicación técnica

Eliminar una columna es una operación que modifica la estructura física de la tabla:

ALTER TABLE ... DROP COLUMN elimina definitivamente la columna.

Se borran los datos almacenados en esa columna.

No afecta las demás columnas ni las llaves primarias.

SQL Server lo permite siempre que:

- ✓ la columna no sea parte de una llave primaria
- ✓ no tenga un índice propio
- ✓ no forme parte de una restricción (CHECK, DEFAULT, FK)

Proyecto 9: Implementación de registros automáticos de modificaciones en QhatuPeru

"Es necesario auditar todas las modificaciones realizadas en la tabla Clientes para cumplir con políticas internas de seguridad y protección de datos."

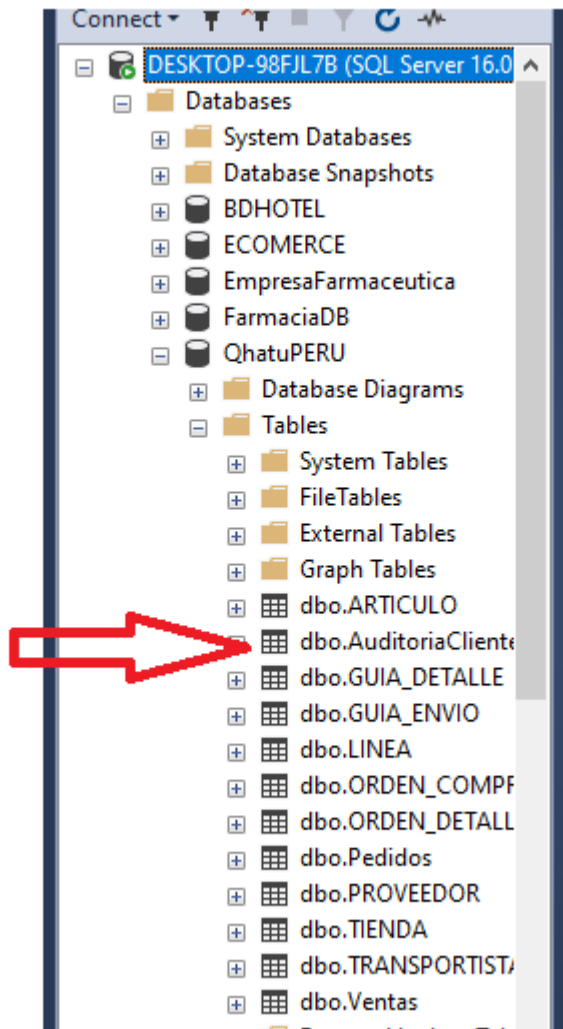
Ejercicio práctico:

- Creas una tabla AuditoriaClientes para registrar cambios en Clientes.
- Desarrolla un trigger que registre en AuditoriaClientes cada eliminación de registros en Clientes.

Página | 4

a.) Crear la tabla AuditoriaClientes

```
CREATE TABLE dbo.AuditoriaClientes (  
    IdAuditoria INT IDENTITY(1,1) PRIMARY KEY, -- Identificador único  
    IdCliente INT NOT NULL, -- Cliente afectado  
  
    Accion VARCHAR(20) NOT NULL, -- INSERT, UPDATE, DELETE  
    Usuario VARCHAR(100) NOT NULL, -- Usuario que realizó el cambio  
    FechaCambio DATETIME NOT NULL DEFAULT GETDATE(), -- Fecha del cambio  
  
    ValorAnterior VARCHAR(MAX) NULL, -- Datos antes del cambio  
    ValorNuevo VARCHAR(MAX) NULL -- Datos después del cambio  
);  
GO
```



Explicación técnica del diseño

Columna	Función
IdAuditoria	Identificador del evento de auditoría.
IdCliente	Cliente cuyo registro fue modificado.
Accion	Tipo de operación: INSERT, UPDATE o DELETE.
Usuario	Nombre del usuario que ejecutó el cambio (capturado desde <code>SUSER_SNAME()</code>).
FechaCambio	Fecha y hora exacta de la modificación.
ValorAnterior	Registro previo antes del cambio (para UPDATE o DELETE).
ValorNuevo	Registro después del cambio (para INSERT o UPDATE).

Buenas prácticas aplicadas

✓ 1. Crear una tabla separada para auditoría

Evita mezclar datos operativos con datos de trazabilidad.

✓ 2. Usar columnas "ValorAnterior" y "ValorNuevo"

Permite reconstruir el estado del cliente antes y después del cambio.

✓ 3. Guardar fecha y usuario del cambio

Información necesaria para auditorías oficiales y trazabilidad.

✓ 4. Usar IDENTITY para facilitar consultas

La columna IdAuditoria permite ordenar y filtrar eventos fácilmente.

b.) Trigger para registrar cada eliminación en la tabla Clientes

```
USE QhatuPERU;
```

```
GO
```

```
CREATE TRIGGER trg_Clientes_Delete
```

```
ON dbo.Clientes
```

```
FOR DELETE
```

```
AS
```

```
BEGIN
```

```
    SET NOCOUNT ON;
```

```
    INSERT INTO dbo.AuditoriaClientes (
```

```
        IdCliente,
```

```
        Accion,
```

```
        Usuario,
```

```
        FechaCambio,
```

```
        ValorAnterior,
```

```
        ValorNuevo
```

```
    )
```

```
    SELECT
```

```
        d.IdCliente,
```

```
        'DELETE',
```

```
        SUSER_SNAME(),
```

```
        GETDATE(),
```

```
        CONCAT(
```

```
            'Nombre: ', d.Nombre,
```

```
            ', Apellido: ', d.Apellido,
```

```
            ', DNI: ', d.DNI,
```

```
            ', Telefono: ', d.Telefono,
```

```
            ', Direccion: ', d.Direccion
```

```
        ),
```

```
        NULL
```

```
    FROM deleted d;
```

```
END;
```

```
GO
```


SQLQuery1.sql - DE...-98FJL7B\Perú (69))*

```
USE QhatuPERU;
GO

CREATE TRIGGER trg_Clientes_Delete
ON dbo.Clientes
FOR DELETE
AS
BEGIN
    SET NOCOUNT ON;

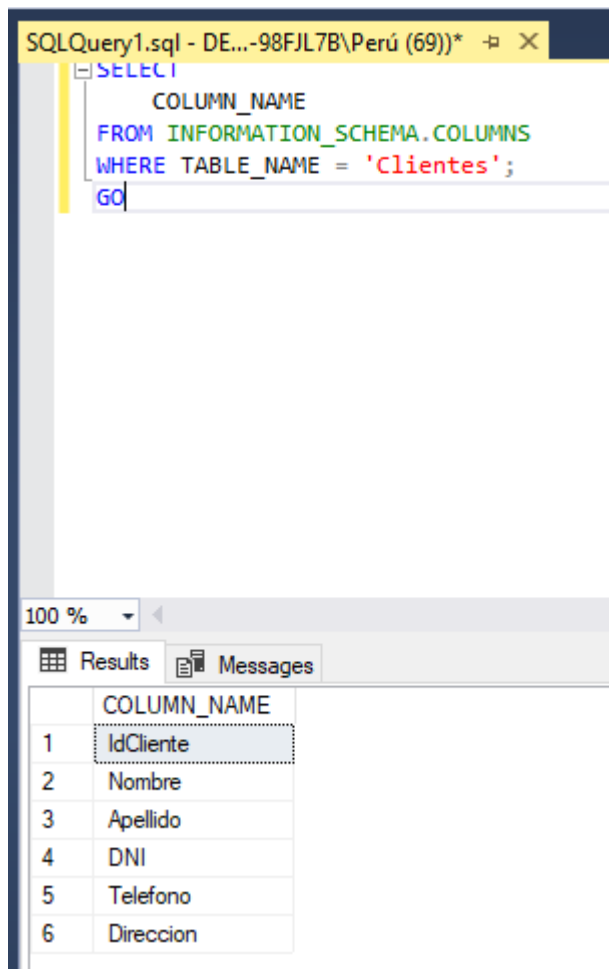
    INSERT INTO dbo.AuditoriaClientes (
        IdCliente,
        Accion,
        Usuario,
        FechaCambio,
        ValorAnterior,
        ValorNuevo
    )
```

100 %

Messages

Commands completed successfully.

Completion time: 2025-11-06T13:07:38.9409914-05:00



¿Qué hace este Trigger?

Cada vez que se ejecuta:

DELETE FROM Clientes WHERE ...

El trigger automáticamente:

- ✓ Captura el cliente eliminado
- ✓ Guarda quién realizó la acción (USER_SNAME())
- ✓ Guarda la fecha
- ✓ Guarda los datos completos del cliente antes de eliminarlo
- ✓ Inserta el registro en AuditoriaClientes

Explicación técnica de cada parte

- ✓ 1. FOR DELETE

Indica que el trigger se activa después de un DELETE en la tabla Clientes.

- ✓ 2. DELETED

Tabla virtual que contiene los datos que fueron eliminados.

- ✓ 3. INSERT INTO AuditoriaClientes

Envía los datos eliminados a la tabla auditoría.

- ✓ 4. ValorAnterior

Aquí se concatenan los datos del cliente antes de ser eliminado.

✓ 5. ValorNuevo = NULL

Porque en una eliminación no existe un valor nuevo.

Buenas prácticas aplicadas

✓ 1. No usar SELECT * en triggers

Evita errores si la tabla cambia.

✓ 2. Concatenar campos eliminados

Permite reconstruir fácilmente cómo era el registro.

✓ 3. Usar SUSER_SNAME()

Captura el usuario que realizó la acción, incluso si es SQL Login.

✓ 4. Usar SET NOCOUNT ON

Evita mensajes innecesarios que pueden interferir con aplicaciones.

✓ 5. Compatible con eliminaciones múltiples

Si borran 5 clientes en una sola sentencia, el trigger inserta 5 registros en auditoría automáticamente.

Proyecto 10: Simulación de restauración tras un incidente en QhatuPeru

"Por un error humano se eliminó información de la tabla Clientes. El equipo debe restaurar la base usando el respaldo más reciente y validar la recuperación exitosa de los datos."

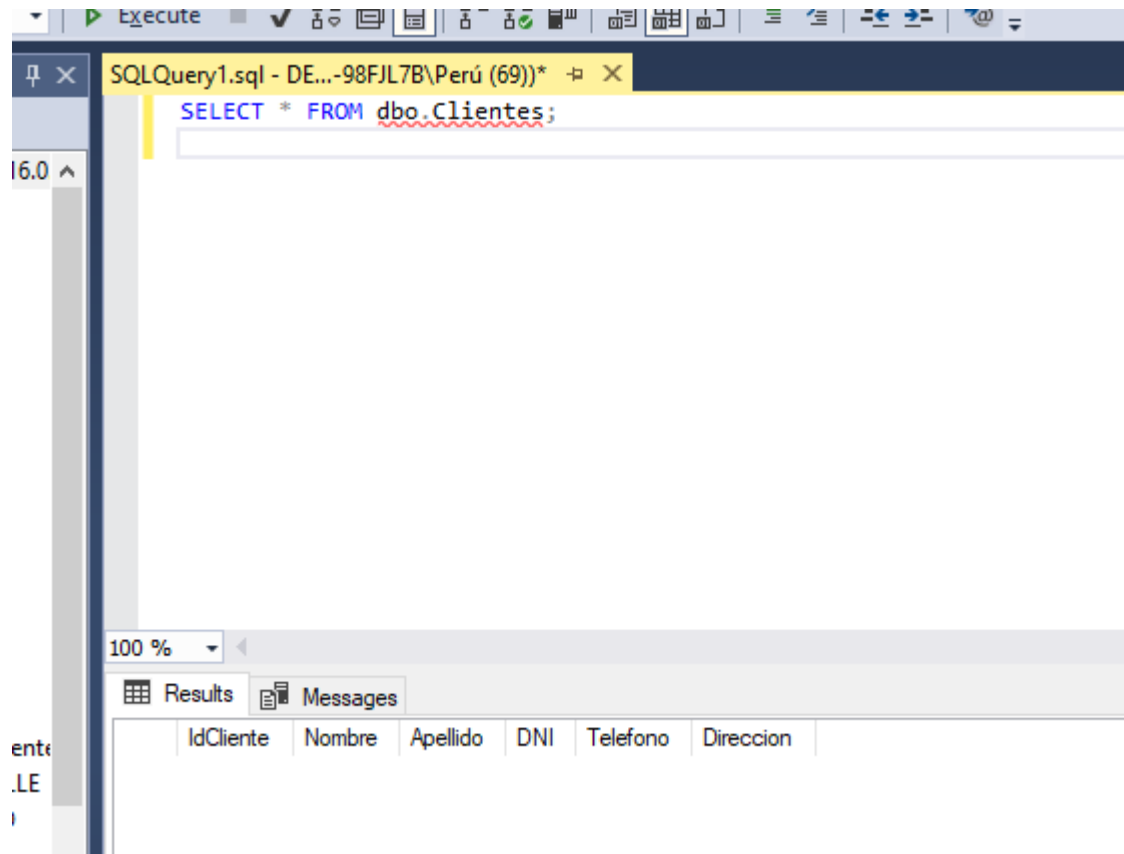
Ejercicio práctico:

- Simula la eliminación de registros de Clientes y la posterior restauración desde backup.
- Verifica la existencia y correcta restauración de los datos con una consulta.

a. Simulación de eliminación de registros de Clientes y restauración desde Backup

Primero, identificamos cuántos clientes existen antes del error:

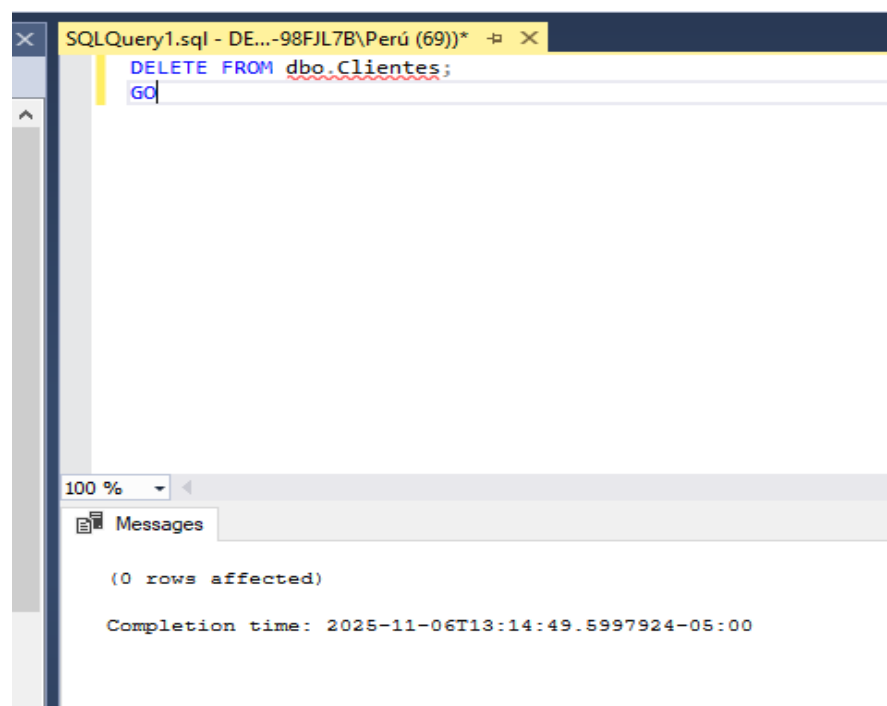
```
SELECT * FROM dbo.Clientes;
```



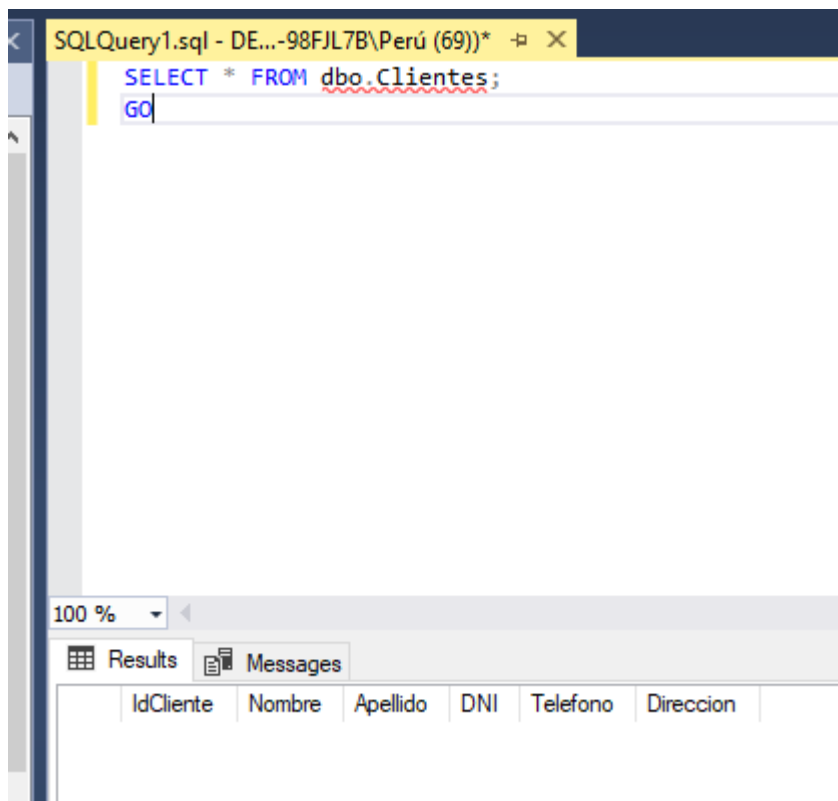
Luego simulamos la eliminación accidental:

```
DELETE FROM dbo.Cientes;
```

```
GO
```



Para confirmar que los registros fueron eliminados:



b.) Verificar la existencia y correcta restauración de los datos con una consulta

Consulta para verificar los registros restaurados

```
SELECT *  
FROM dbo.Cientes;  
GO
```

The screenshot shows a SQL Server Enterprise Manager interface. At the top, a query window titled 'SQLQuery1.sql - DE...-98FJL7B\Perú (69))' contains the following SQL code:

```
SELECT * FROM dbo.Cientes;
GO
```

Below the query window, the 'Results' tab is active, displaying a table with 7 columns and 5 rows of data. The columns are: IdCliente, Nombre, Apellido, DNI, Telefono, and Direccion. The rows represent five different clients.

	IdCliente	Nombre	Apellido	DNI	Telefono	Direccion
1	1	Carlos	Gómez	12345678	987654321	Jr. Los Pinos 145
2	2	María	López	87654321	912345678	Av. Huancavelica 230
3	3	Luis	Torres	11223344	956789123	Calle Lima 320
4	4	Ana	García	44556677	934567812	Pasaje Las Flores 58
5	5	Jorge	Ramírez	99887766	987123456	Av. Ferrocarril 1020

Interpretación de la consulta

Si la tabla Cientes muestra nuevamente los registros que habían sido eliminados, significa que la restauración del backup fue exitosa.

Si la tabla aparece vacía, la restauración no se realizó correctamente o el backup utilizado no contenía datos.

Justificación técnica

La inserción de datos en la tabla dbo.Cientes es una operación fundamental para garantizar que la base de datos cuente con información real o simulada que permita:

✓ 1. Probar el funcionamiento correcto de la base de datos

Al agregar registros de ejemplo:

Se puede verificar que la tabla Cientes funciona correctamente.

Los triggers (como el de auditoría que creaste) pueden ser probados.

Se validan consultas SELECT, UPDATE, DELETE y la integridad de la tabla.

✓ 2. Validar procesos posteriores (Delete, Backup, Restore)

Los ejercicios que requieren:

eliminar datos,

simular errores,

restaurar desde backup,

registrar auditorías,
solo pueden realizarse si existen registros previos.
Por ello, insertar clientes es un paso obligatorio.