UNIVERSIDAD PERUANA LOS ANDES

FACULTAD INGENIERIA INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

MANUAL DE LA PRACTICA DE MONITOREO Y RENDIMIENTO EN SQL SERVER

ASIGNATURA: BASE DE DATOS II

DOCENTE: MG. HERBERTH ARANDA ROJAS

ESTUDIANTE: Bonifacio Hilario Erick

CÓDIGO: S01238F

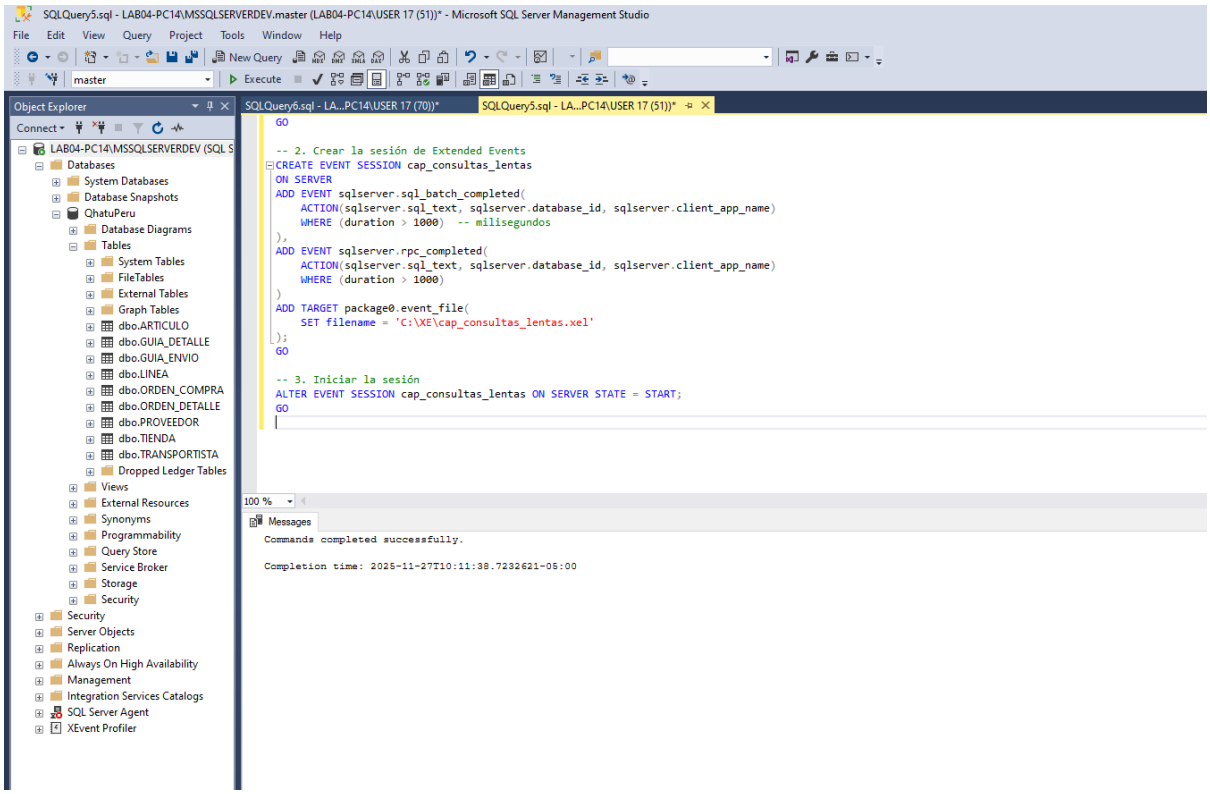
HUANCAYO-2025

Proyecto 1 — Captura de consultas lentas con Extended Events

1. **Enunciado**
   Crear una sesión de Extended Events que capture consultas que tarden más de **1 segundo** en QhatuPeru, guardando en archivo `.xel`.
2. **Script T-SQL**

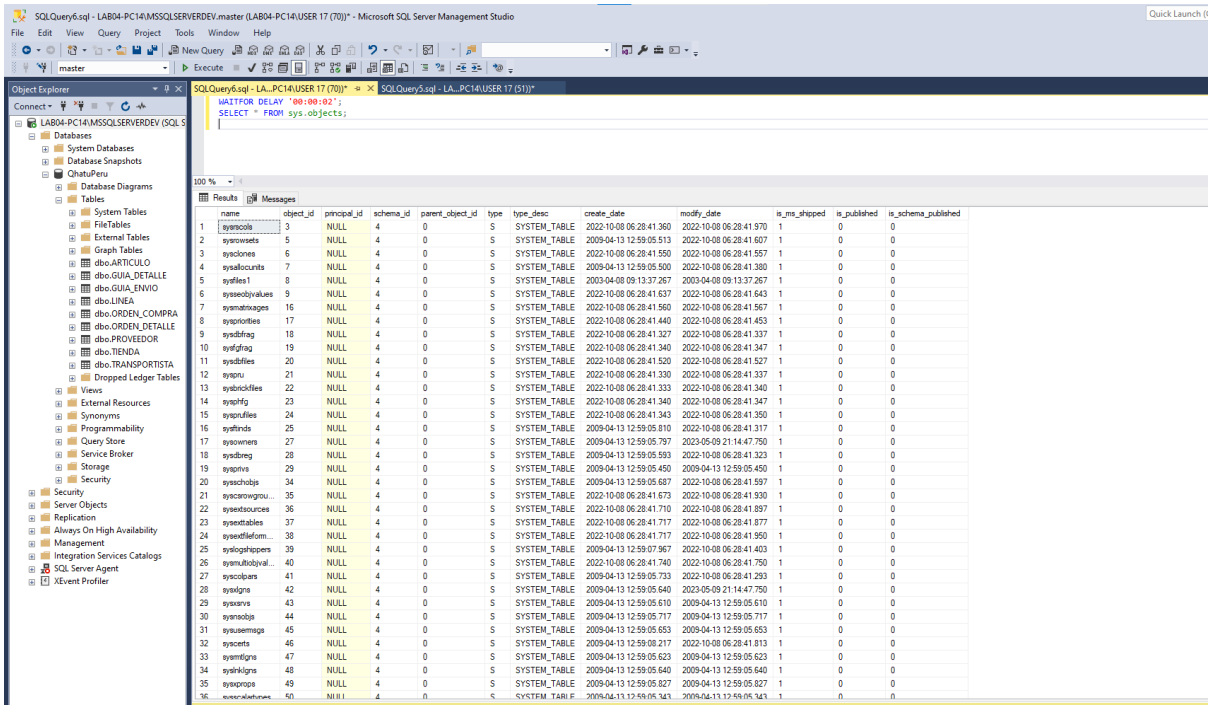
```
USE master;
GO

DECLARE @path NVARCHAR(400) = N'C:\XE\QhatuPeru_SlowQueries.xel'; --
ajusta ruta
DECLARE @dbname SYSNAME = N'QhatuPeru';
DECLARE @dbid INT = DB_ID(@dbname);
DECLARE @sessionname SYSNAME = N'XE_QhatuPeru_SlowQueries';
DECLARE @sql NVARCHAR(MAX);

-- Elimina sesión si existe
IF EXISTS (SELECT 1 FROM sys.server_event_sessions WHERE name =
@sessionname)
BEGIN
    ALTER EVENT SESSION [XE_QhatuPeru_SlowQueries] ON SERVER STATE =
STOP;
    DROP EVENT SESSION [XE_QhatuPeru_SlowQueries] ON SERVER;
END

-- Construir CREATE EVENT SESSION con valores embebidos
SET @sql = N'
CREATE EVENT SESSION ' + QUOTENAME(@sessionname) + N' ON SERVER
ADD EVENT sqlserver.sql_statement_completed(
    ACTION(sqlserver.sql_text, sqlserver.session_id,
sqlserver.database_id, sqlserver.client_app_name,
sqlserver.username)
    WHERE (sqlserver.database_id = ' + CAST(@dbid AS NVARCHAR(10)) +
N' AND duration > 1000000)
)
ADD TARGET package0.event_file(SET filename = N''' +
REPLACE(@path,'''','''''') + N''', max_file_size=(100),
max_rollover_files=(5))
WITH (MAX_MEMORY=4096 KB,
```

```
EVENT_RETENTION_MODE=ALLOW_SINGLE_EVENT_LOSS, MAX_DISPATCH_LATENCY=1
SECONDS);
';

EXEC sp_executesql @sql;

-- Iniciar sesión XE
ALTER EVENT SESSION [XE_QhatuPeru_SlowQueries] ON SERVER STATE =
START;
GO
```



3. Justificación técnica

- `duration` está en microsegundos: `> 1,000,000` = 1s.
- Uso SQL dinámico porque `CREATE EVENT SESSION` no acepta variables T-SQL directas para el `filename` ni para expresiones en `WHERE`.
- Se filtra por `database_id` para limitar captura.



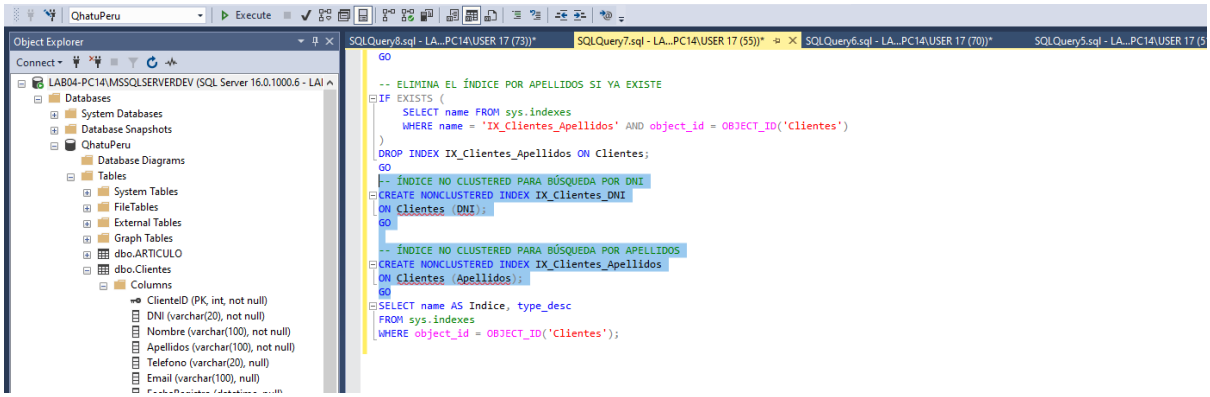
4. **Buenas prácticas**

- Ajustar ruta a disco con espacio suficiente, usar rollover.
- Ejecutar con permisos adecuados y supervisar tamaño de archivos `.xel`.
- No capturar todo el `sql_text` si hay mucho volumen (podrías capturar solo query_hash en escenarios muy verbosos).

Proyecto 2 — Índices para búsqueda por DNI y Apellidos

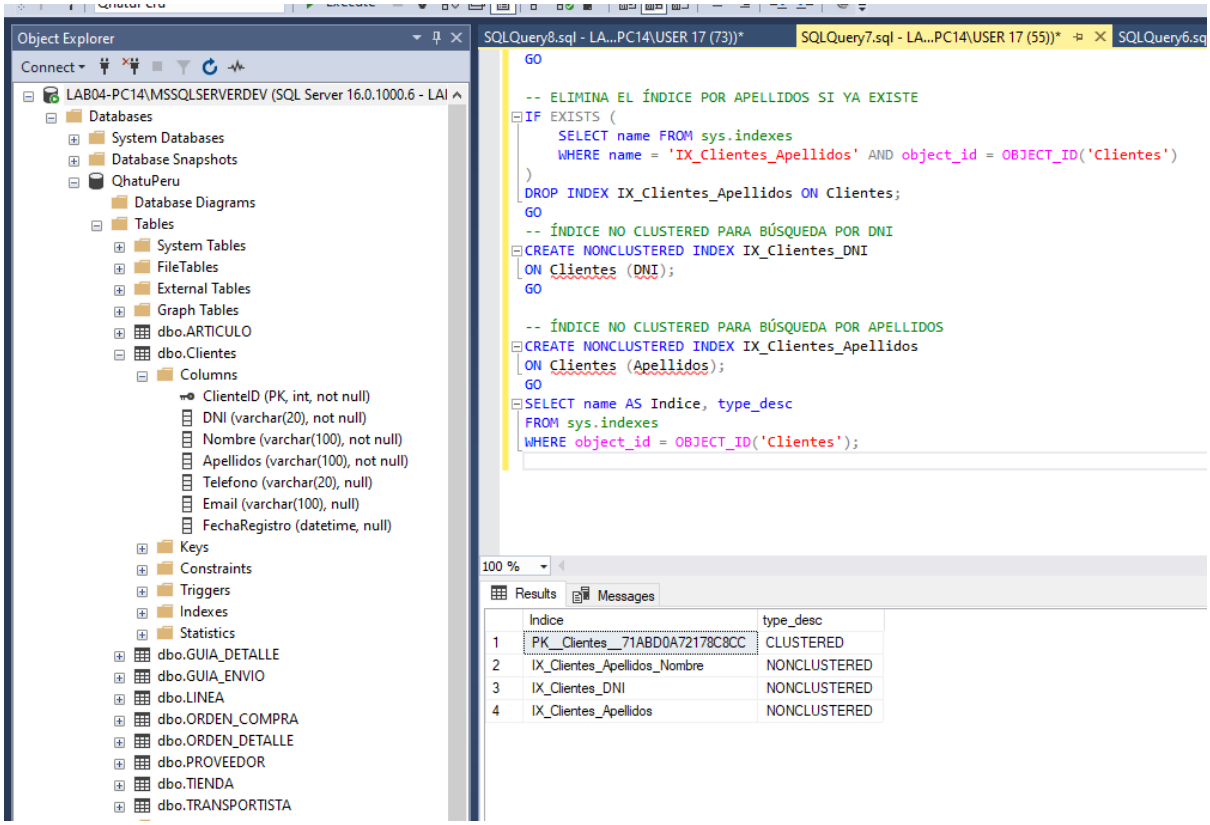
1. **Enunciado**
   Crear índices para mejorar búsquedas por `DNI` y por `Apellidos` en `dbo.Clientes`.
2. **Script T-SQL**

```
-- ÍNDICE NO CLUSTERED PARA BÚSQUEDA POR DNI CREATE NONCLUSTERED
INDEX IX_Clientes_DNI ON Clientes (DNI); GO

-- ÍNDICE NO CLUSTERED PARA BÚSQUEDA POR APELLIDOS CREATE
NONCLUSTERED INDEX IX_Clientes_Apellidos ON Clientes (Apellidos); GO
```



3. Justificación técnica

- `DNI` suele ser búsqueda exacta → índice único.
- `(Apellidos, Nombre)` mejora `LIKE 'García%'` y `ORDER BY Apellidos, Nombre`.
- `INCLUDE` evita lookups en consultas comunes.



4. Buenas prácticas

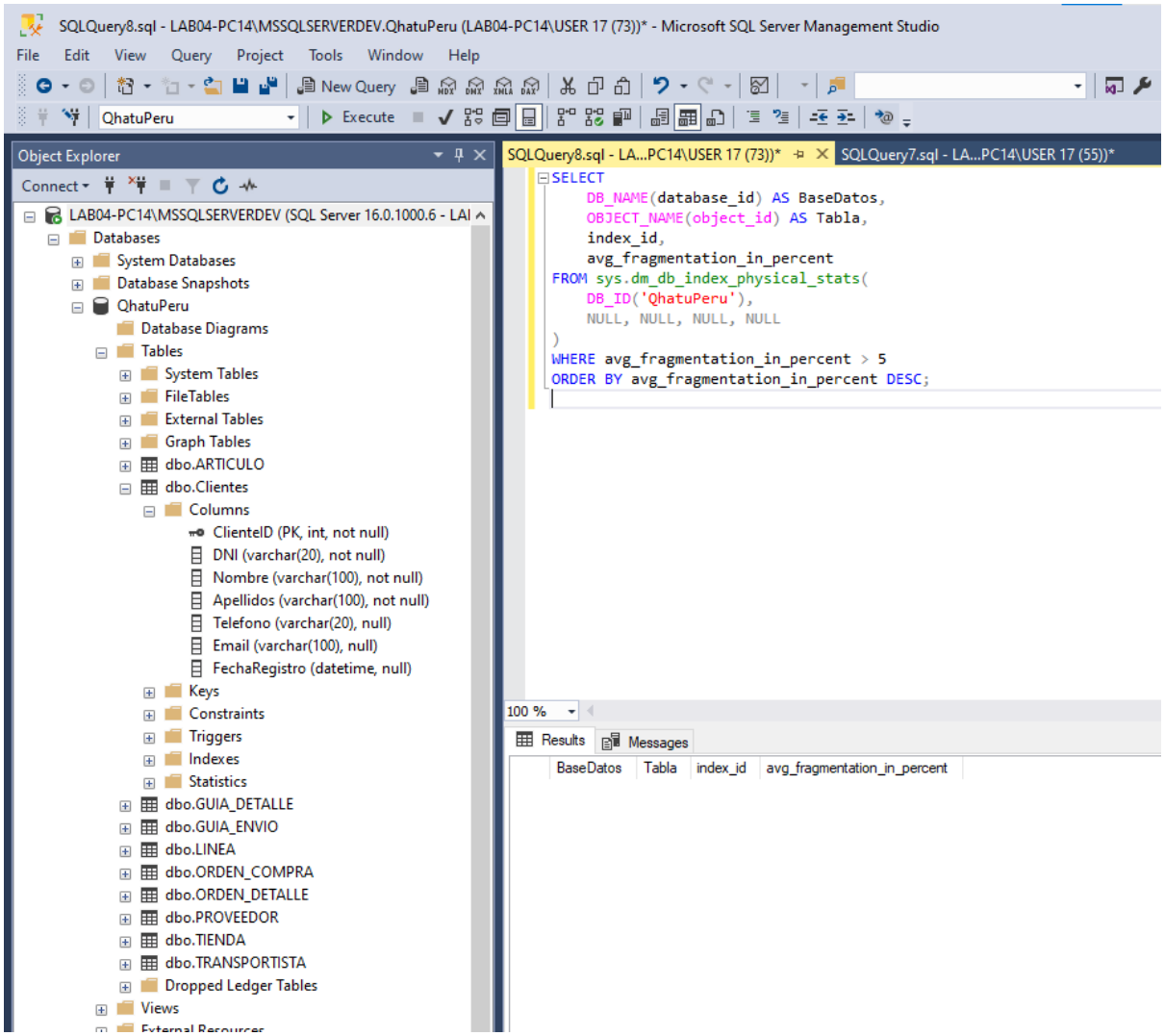
- Revisar cardinalidad antes de crear índices.
- Evitar índices redundantes; balance lectura/escritura.

Proyecto 3 — Detectar fragmentación y mantenimiento de índices

1. **Enunciado**
   Evaluar fragmentación en QhatuPeru y REORGANIZE/REBUILD según porcentaje.
2. **Script T-SQL**

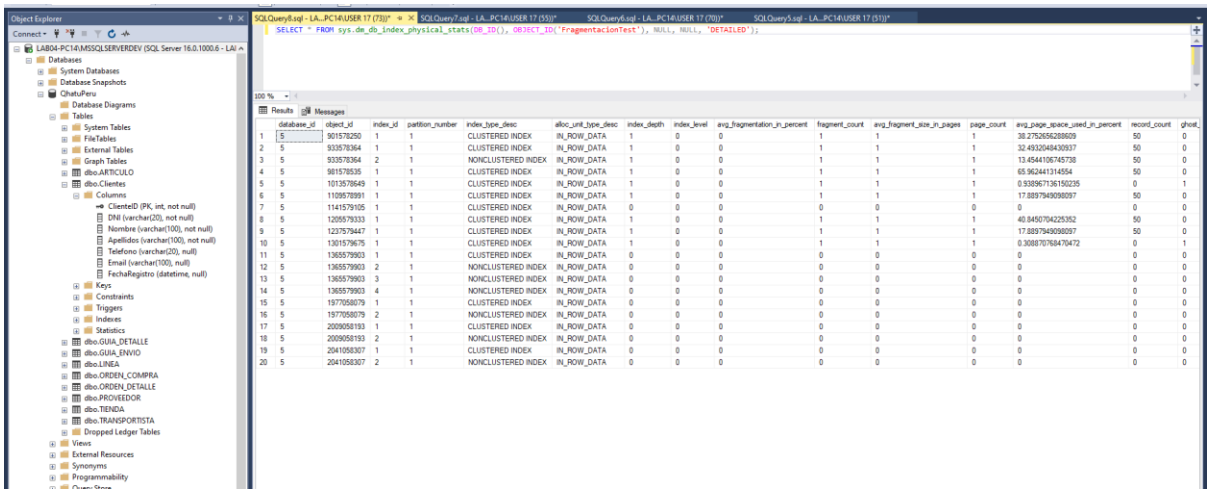
2.1 Detectar fragmentación

```
SELECT
    DB_NAME(database_id) AS BaseDatos,
    OBJECT_NAME(object_id) AS Tabla,
    index_id,
    avg_fragmentation_in_percent
FROM sys.dm_db_index_physical_stats(
    DB_ID('QhatuPeru'),
    NULL, NULL, NULL, NULL
)
WHERE avg_fragmentation_in_percent > 5
ORDER BY avg_fragmentation_in_percent DESC;
```



3. **Justificación técnica**

- `avg_fragmentation_in_percent` y `page_count` definen la acción.
- `REORGANIZE` menos costoso; `REBUILD` más efectivo.
- Construcción dinámica de `ALTER INDEX` evita errores de concatenación.



4. **Buenas prácticas**

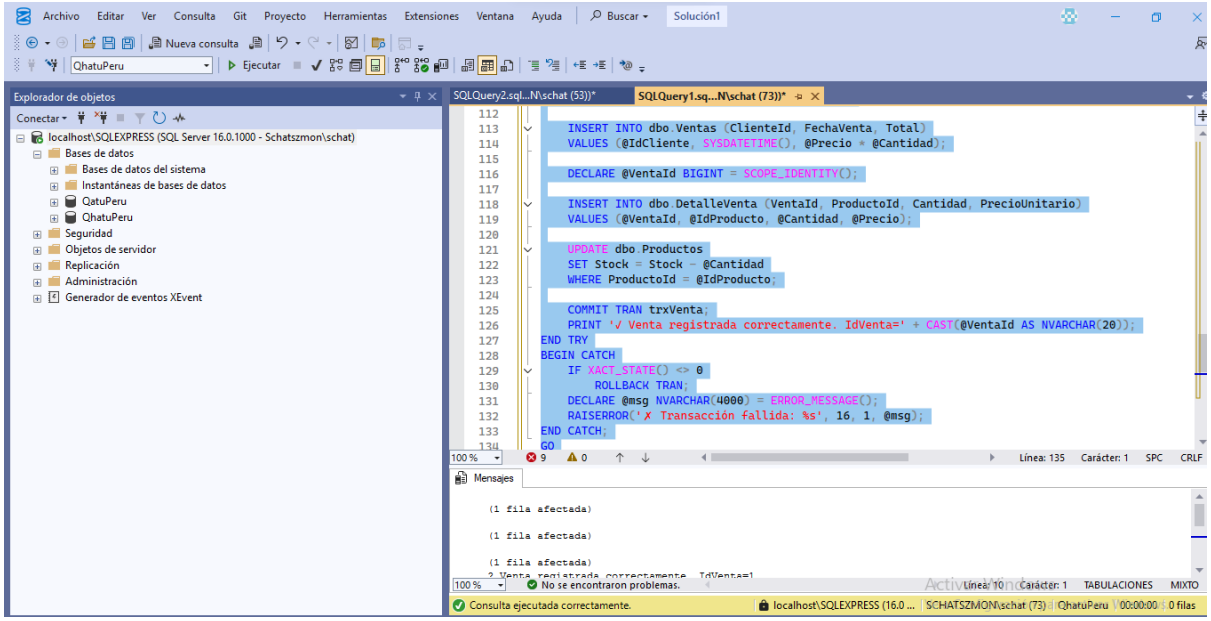
- Programar en ventanas de baja carga.
- Monitorear espacio tempdb/archivo de transacción (REBUILD puede ser intensivo).
- Log de operaciones.

Proyecto 4 — Transacción de venta (consistencia)

1. **Enunciado**
   Simular transacción con `INSERT` en `Ventas` y `UPDATE` en `Productos.Stock`, garantizando consistencia.
2. **Script T-SQL**

```
BEGIN TRY
    SET XACT_ABORT ON;
    BEGIN TRAN trxVenta;

    DECLARE @IdProducto INT = 101;
    DECLARE @Cantidad INT = 2;
    DECLARE @IdCliente INT = 2001;
    DECLARE @Precio DECIMAL(18,2) = (SELECT PrecioVenta FROM
dbo.Productos WHERE ProductoId = @IdProducto);

    IF @Precio IS NULL
        THROW 50002, 'Producto no existe.', 1;

    IF (SELECT Stock FROM dbo.Productos WHERE ProductoId =
@IdProducto) < @Cantidad
```

```
        THROW 50001, 'Stock insuficiente.', 1;

    INSERT INTO dbo.Ventas (ClienteId, FechaVenta, Total)
    VALUES (@IdCliente, SYSDATETIME(), @Precio * @Cantidad);

    DECLARE @VentaId BIGINT = SCOPE_IDENTITY();

    INSERT INTO dbo.DetalleVenta (VentaId, ProductoId, Cantidad,
PrecioUnitario)
    VALUES (@VentaId, @IdProducto, @Cantidad, @Precio);

    UPDATE dbo.Productos
    SET Stock = Stock - @Cantidad
    WHERE ProductoId = @IdProducto;

    COMMIT TRAN trxVenta;
    PRINT '✓ Venta registrada correctamente. IdVenta=' +
CAST(@VentaId AS NVARCHAR(20));
END TRY
BEGIN CATCH
    IF XACT_STATE() <> 0
        ROLLBACK TRAN;
    DECLARE @msg NVARCHAR(4000) = ERROR_MESSAGE();
    RAISERROR('✗ Transacción fallida: %s', 16, 1, @msg);
END CATCH;
GO
```

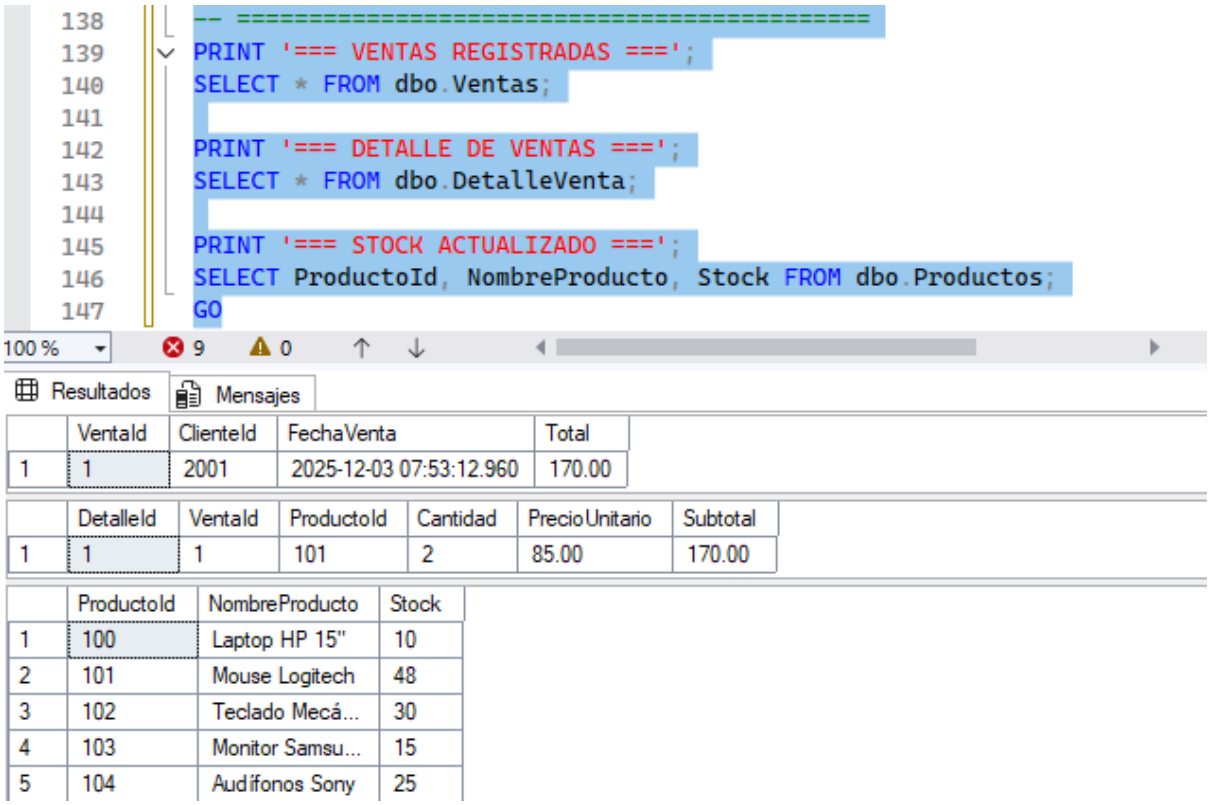


VERIFICAR RESULTADOS

PRINT '=== VENTAS REGISTRADAS ==='; SELECT * FROM dbo.Ventas;

PRINT '=== DETALLE DE VENTAS ==='; SELECT * FROM dbo.DetalleVenta;

PRINT '=== STOCK ACTUALIZADO ==='; SELECT ProductoId, NombreProducto, Stock FROM dbo.Productos; GO



```
138  -- ============================================
139  PRINT '=== VENTAS REGISTRADAS ===';
140  SELECT * FROM dbo.Ventas;
141
142  PRINT '=== DETALLE DE VENTAS ===';
143  SELECT * FROM dbo.DetalleVenta;
144
145  PRINT '=== STOCK ACTUALIZADO ===';
146  SELECT ProductoId, NombreProducto, Stock FROM dbo.Productos;
147  GO
```

100 % 9 0

Resultados Mensajes

	VentaId	ClienteId	FechaVenta	Total
1	1	2001	2025-12-03 07:53:12.960	170.00

	DetalleId	VentaId	ProductoId	Cantidad	PrecioUnitario	Subtotal
1	1	1	101	2	85.00	170.00

	ProductoId	NombreProducto	Stock
1	100	Laptop HP 15"	10
2	101	Mouse Logitech	48
3	102	Teclado Mecá...	30
4	103	Monitor Samsu...	15
5	104	Audífonos Sony	25

3. **Justificación técnica**

- `XACT_ABORT ON` garantiza rollback ante errores severos.
- Validaciones previas evitan iniciar transacciones innecesarias.
- `SCOPE_IDENTITY()` para obtener id de la venta.

4. **Buenas prácticas**

- Mantener transacciones cortas.
- Control de concurrencia (optimistic/pessimistic según necesidad).
- Registrar auditoría si falla.

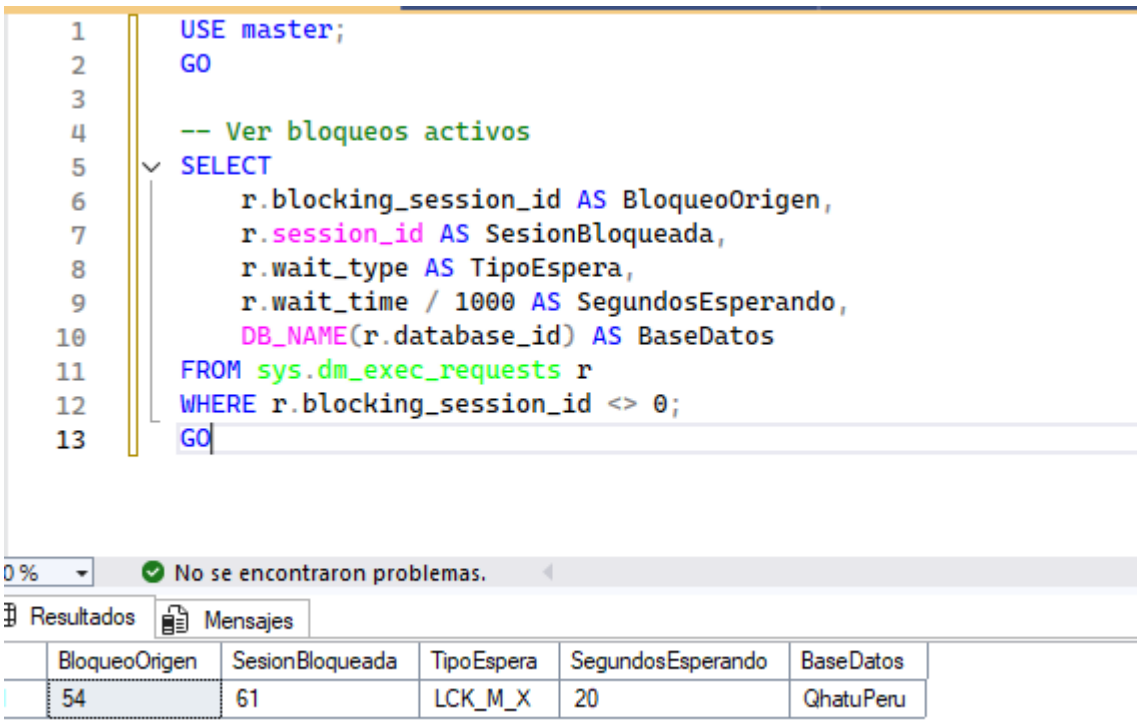
Proyecto 5 — Identificar bloqueos activos

1. **Enunciado**

   Detectar sesiones que bloquean o están bloqueadas en el servidor.

2. **Script T-SQL**

```
USE master;
GO
-- Ver bloqueos activos
SELECT
    r.blocking_session_id AS BloqueoOrigen,
    r.session_id AS SesionBloqueada,
    r.wait_type AS TipoEspera,
    r.wait_time / 1000 AS SegundosEsperando,
    DB_NAME(r.database_id) AS BaseDatos
FROM sys.dm_exec_requests r
WHERE r.blocking_session_id <> 0;
```



3. **Justificación técnica**

- `sys.dm_exec_requests` y `sys.dm_os_waiting_tasks` permiten identificar SPIDs implicados y SQL ejecutado.
- Extrae SQL para investigar la causa.

4. **Buenas prácticas**

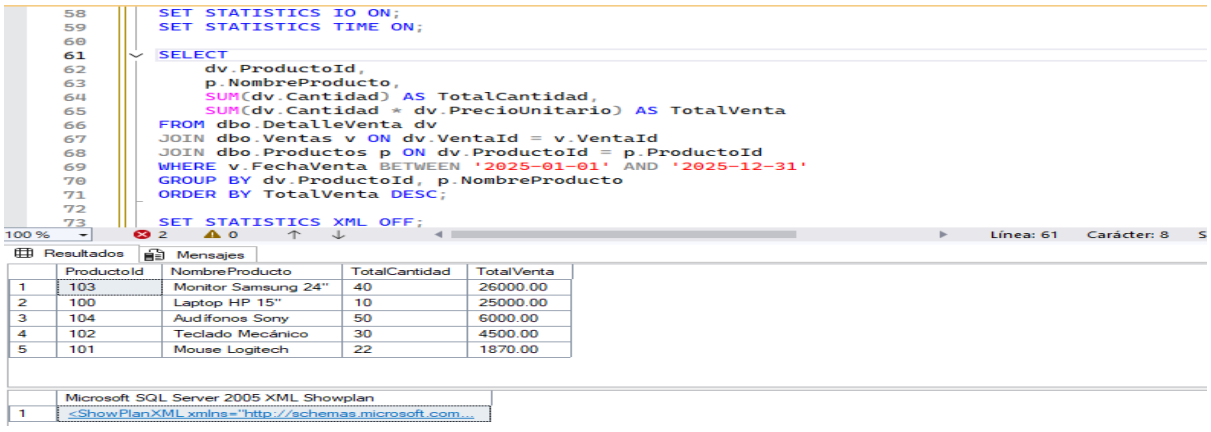
- No matar sesiones sin analizar; primero identificar transacción y usuario.

- Implementar alertas si bloqueo > umbral.

Proyecto 6 — Analizar plan de ejecución de consulta lenta

1. **Enunciado**
   Analizar plan de ejecución de consulta que devuelve ventas por producto.
2. **Script T-SQL**

```
USE QhatuPeru;
GO

-- Obtener plan, IO y tiempo
SET STATISTICS XML ON;
SET STATISTICS IO ON;
SET STATISTICS TIME ON;

SELECT dv.ProductoId, SUM(dv.Cantidad) AS TotalCantidad,
SUM(dv.Cantidad * dv.PrecioUnitario) AS TotalVenta
FROM dbo.DetalleVenta dv
JOIN dbo.Ventas v ON dv.VentaId = v.VentaId
WHERE v.FechaVenta BETWEEN @f1 AND @f2 -- define variables o
reemplaza por fechas
GROUP BY dv.ProductoId
ORDER BY TotalVenta DESC;

SET STATISTICS XML OFF;
SET STATISTICS IO OFF;
SET STATISTICS TIME OFF;
GO
```



3. **Justificación técnica**

- `SET STATISTICS XML ON` ofrece plan con estimaciones vs reales, spills, operaciones costosas.
- `STATISTICS IO/TIME` ayudan a cuantificar I/O y CPU.

4. **Buenas prácticas**

- Revisar mismatch estimado/real (posible falta de estadísticas).
- Probar índices/estadísticas en staging.

Proyecto 7 — Optimización con índices compuestos para filtrado por fecha y cliente

1. **Enunciado**
   Optimizar consulta que filtra `Ventas` por `FechaVenta` y `ClienteId`.
2. **Script T-SQL**

```
USE QhatuPeru;
GO

IF NOT EXISTS (SELECT 1 FROM sys.indexes WHERE name =
'IX_Ventas_Cliente_Fecha' AND object_id = OBJECT_ID('dbo.Ventas'))
BEGIN
    CREATE NONCLUSTERED INDEX IX_Ventas_Cliente_Fecha
    ON dbo.Ventas (ClienteId, FechaVenta)
    INCLUDE (Total); -- cubrir SUM(Total)
END
GO
```

```sql
-- ============================================
-- 2. CREAR EL ÍNDICE OPTIMIZADO
-- ============================================
IF NOT EXISTS (SELECT 1 FROM sys.indexes WHERE name = 'IX_Ventas_Cliente_Fecha' AND object_id = O
BEGIN
    CREATE NONCLUSTERED INDEX IX_Ventas_Cliente_Fecha
    ON dbo.Ventas (ClienteId, FechaVenta)
    INCLUDE (Total);

    PRINT '√ Índice IX_Ventas_Cliente_Fecha creado';
END
ELSE
    PRINT 'El índice ya existe';
GO

-- ============================================
-- 3. VERIFICAR QUE SE CREÓ
```

Línea: 24 Carácter: 1 SPC LF

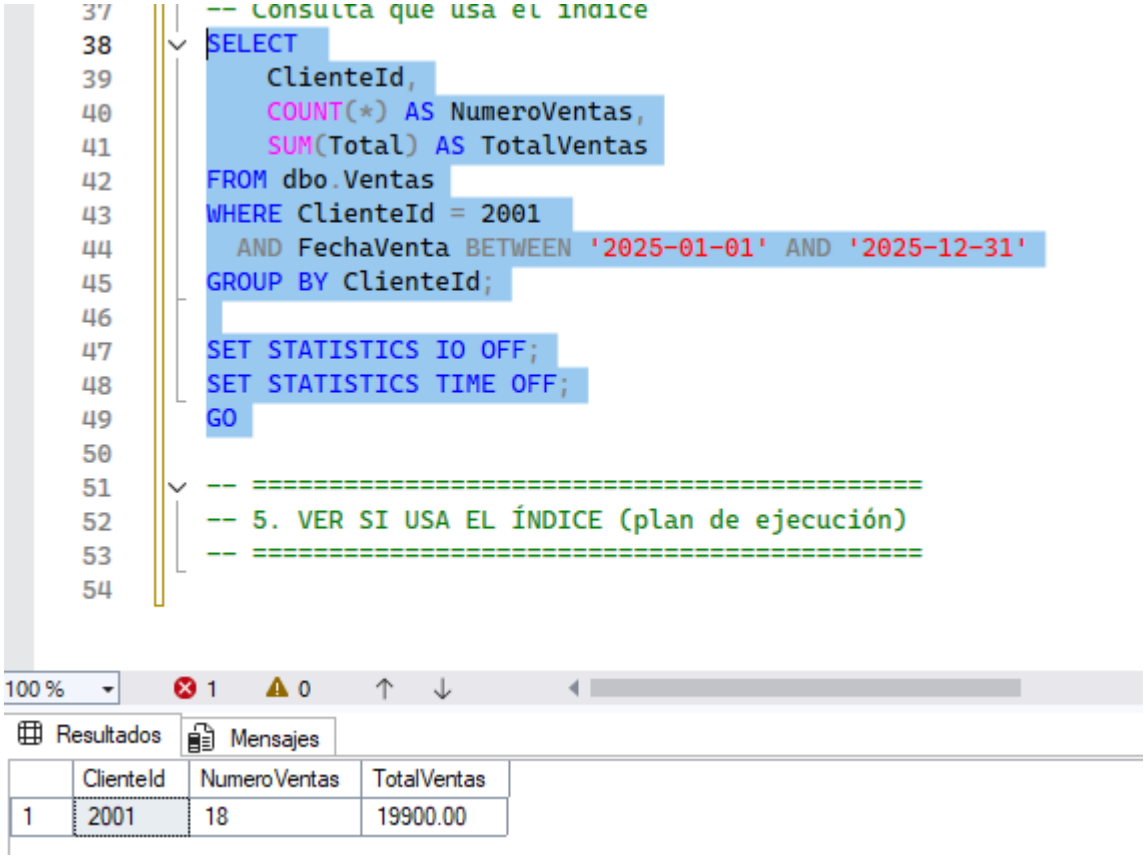
Mensajes

```
? Índice IX_Ventas_Cliente_Fecha creado

Hora de finalización: 2025-12-03T08:21:19.7799711-05:00
```

```sql
-- ============================================
EXEC sp_helpindex 'dbo.Ventas';
GO

-- ============================================
-- 4. PROBAR CONSULTA OPTIMIZADA
-- ============================================
SET STATISTICS IO ON;
SET STATISTICS TIME ON;
```

Resultados | Mensajes

index_name	index_description	index_keys
IX_Ventas_Cliente_Fecha	nonclustered located on PRIMARY	ClienteId, FechaVenta
PK__Ventas__5B4150AC027544C4	clustered, unique, primary key located on PRIMARY	VentaId



3. **Justificación técnica**

- Orden (`ClienteId, FechaVenta`) favorece `WHERE ClienteId = X AND FechaVenta BETWEEN ...`.
- `INCLUDE (Total)` evita lookups para la agregación.

4. **Buenas prácticas**

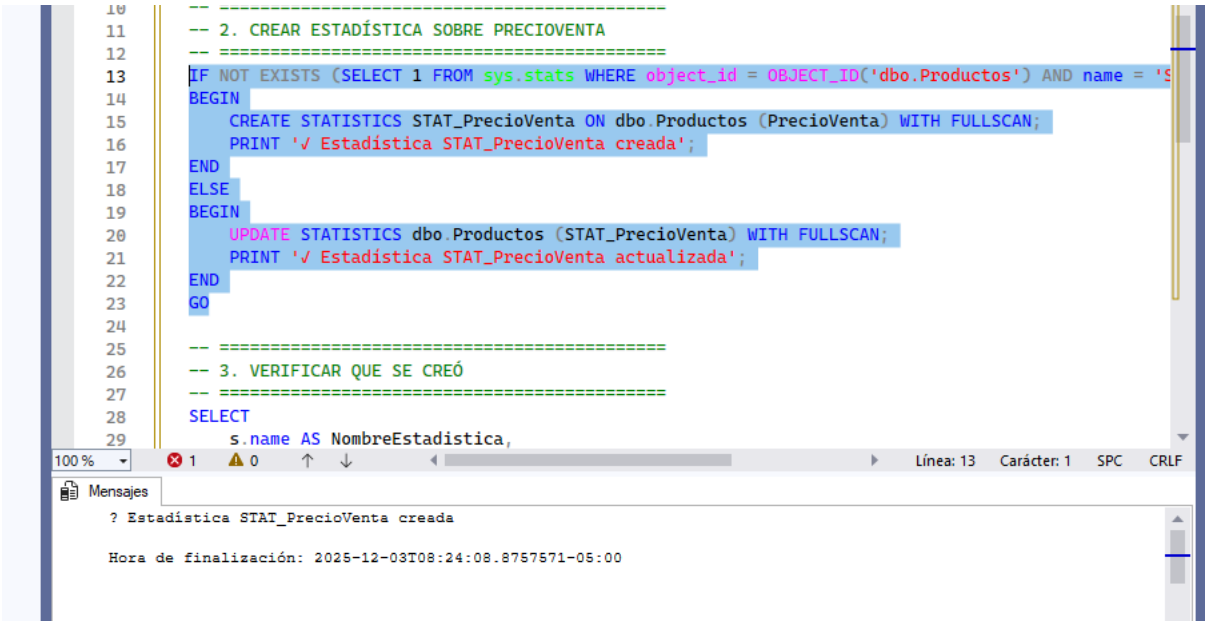
- Verificar patrón de consulta real; si la mayoría filtra por fecha primero, cambiar orden de columnas.

Proyecto 8 — Crear estadísticas manuales sobre Precio

1. **Enunciado**
   Crear estadística manual sobre `Precio` en `Productos` para mejorar consultas rango.
2. **Script T-SQL**

```
USE QhatuPeru;
GO
```

```
-- Crear o actualizar STAT sobre Precio
IF NOT EXISTS (SELECT 1 FROM sys.stats WHERE object_id =
OBJECT_ID('dbo.Productos') AND name = 'STAT_Precio')
BEGIN
    CREATE STATISTICS STAT_Precio ON dbo.Productos (Precio) WITH
FULLSCAN;
END
ELSE
BEGIN
    UPDATE STATISTICS dbo.Productos (STAT_Precio) WITH FULLSCAN;
END
GO
```



```sql
-- ============================================
-- 3. VERIFICAR QUE SE CREÓ
-- ============================================
SELECT
    s.name AS NombreEstadistica,
    COL_NAME(s.object_id, sc.column_id) AS Columna,
    s.auto_created AS AutoCreada,
    s.user_created AS CreadaPorUsuario,
    STATS_DATE(s.object_id, s.stats_id) AS UltimaActualizacion
FROM sys.stats s
JOIN sys.stats_columns sc ON s.stats_id = sc.stats_id AND s.object_id = sc.object_id
WHERE s.object_id = OBJECT_ID('dbo.Productos');
GO

-- ============================================
-- 4. VER DETALLE DE LA ESTADÍSTICA
-- ============================================
```

Línea: 28 Carácter: 1 SPC

Resultados | Mensajes

NombreEstadistica	Columna	AutoCreada	CreadaPorUsuario	UltimaActualizacion
PK__Producto__A430AEA3043F79F0	ProductoId	0	0	2025-12-03 07:53:12.903
STAT_PrecioVenta	PrecioVenta	0	1	2025-12-03 08:24:08.863

```sql
-- ============================================
-- 4. VER DETALLE DE LA ESTADÍSTICA
-- ============================================
DBCC SHOW_STATISTICS ('dbo.Productos', 'STAT_PrecioVenta');
GO

-- ============================================
-- 5. PROBAR CONSULTA POR RANGO DE PRECIO
-- ============================================
SET STATISTICS IO ON;

SELECT ProductoId, NombreProducto, PrecioVenta
FROM dbo.Productos
WHERE PrecioVenta BETWEEN 100 AND 500;

SET STATISTICS IO OFF;
GO
```

100 % Línea: 42 Carácter: 1 SPC CRLF

Resultados | Mensajes

3	650.00	0	1	0	1
4	2500.00	0	1	0	1

	ProductoId	NombreProducto	PrecioVenta
1	102	Teclado Mecánico	150.00
2	104	Audífonos Sony	120.00

3. **Justificación técnica**

- Estadísticas precisas permiten mejores estimaciones de cardinalidad para rangos (BETWEEN, >, <).
- FULLSCAN máxima precisión tras cargas masivas.

4. **Buenas prácticas**

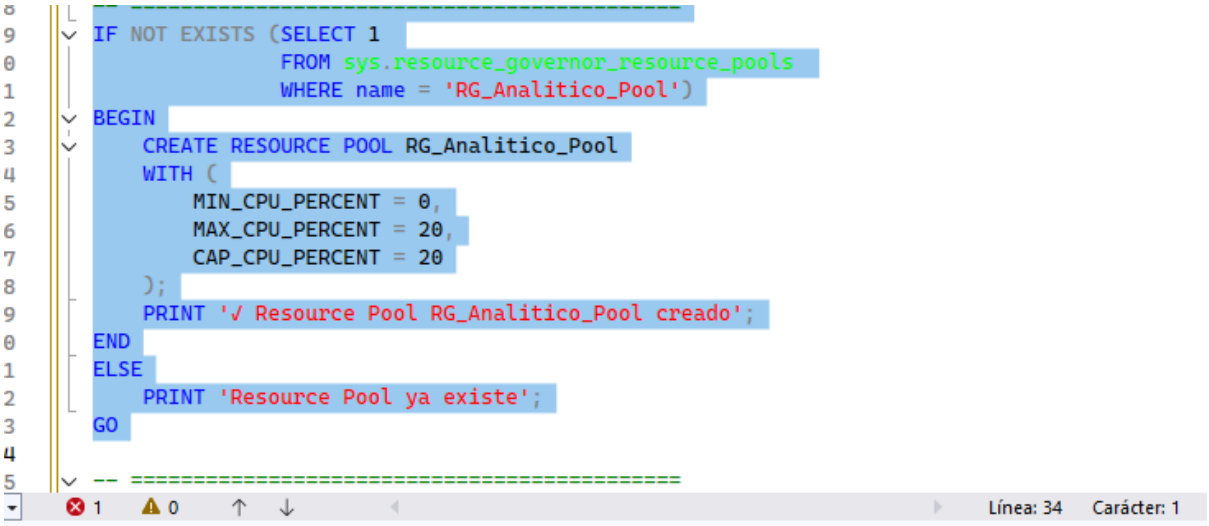
- No ejecutar FULLSCAN demasiado seguido; hacerlo tras cargas importantes.
- Monitorear `auto_update_statistics` y evaluar si conviene mantener automático.

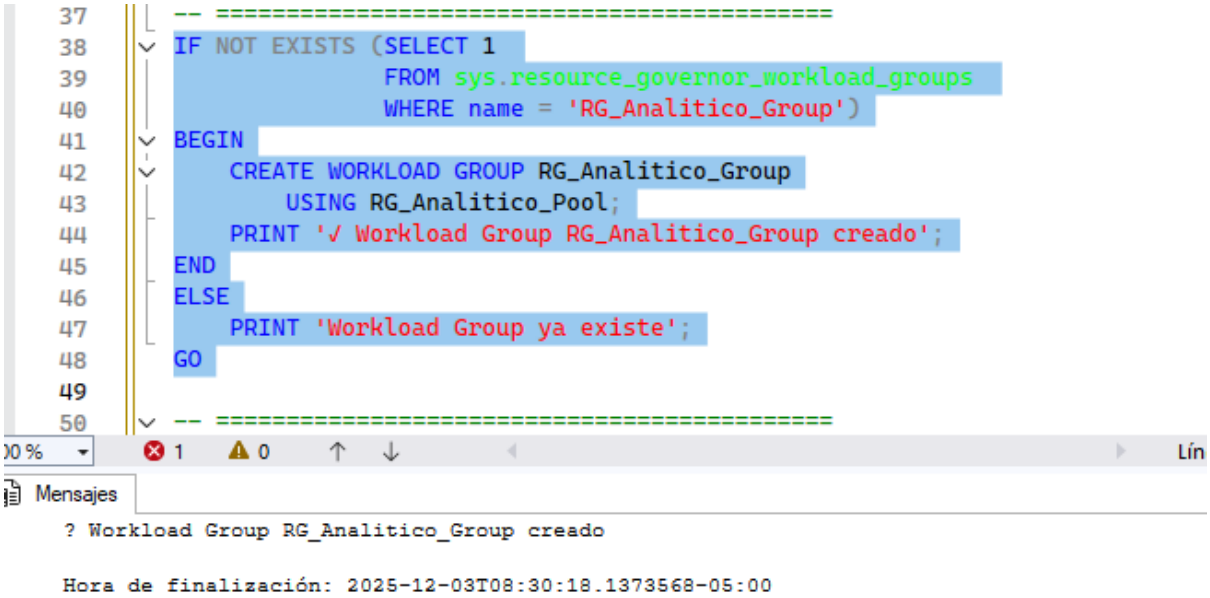
Proyecto 9 — Configuración de Resource Pool (limitación CPU 20%)

1. **Enunciado**
   Crear Resource Pool que limite el uso de CPU al 20% para consultas analíticas.
2. **Script T-SQL**

```
-- Requiere sysadmin
USE master;
GO

-- Crear pool y grupo
IF NOT EXISTS (SELECT 1 FROM sys.resource_governor_resource_pools
WHERE name = 'RG_Analitico_Pool')
BEGIN
    CREATE RESOURCE POOL RG_Analitico_Pool
    WITH (MIN_CPU_PERCENT = 0, MAX_CPU_PERCENT = 20, CAP_CPU_PERCENT
= 20);
END
GO

IF NOT EXISTS (SELECT 1 FROM sys.resource_governor_workload_groups
WHERE name = 'RG_Analitico_Group')
BEGIN
    CREATE WORKLOAD GROUP RG_Analitico_Group
    USING RG_Analitico_Pool;
END
GO

-- Función classifier simple
IF OBJECT_ID('dbo.RGClassifier', 'FN') IS NOT NULL
    DROP FUNCTION dbo.RGClassifier;
GO

CREATE FUNCTION dbo.RGClassifier()
RETURNS sysname
WITH SCHEMABINDING
AS
BEGIN
    DECLARE @grp sysname = 'default';
    IF APP_NAME() = 'AnalisisBatch' OR SUSER_SNAME() = 'analista'
```

```
        SET @grp = 'RG_Analitico_Group';
    RETURN @grp;
END;
GO

ALTER RESOURCE GOVERNOR WITH (CLASSIFIER_FUNCTION =
dbo.RGClassifier);
ALTER RESOURCE GOVERNOR RECONFIGURE;
GO
```





```sql
IF OBJECT_ID('dbo.RGClassifier', 'FN') IS NOT NULL
    DROP FUNCTION dbo.RGClassifier;
GO

CREATE FUNCTION dbo.RGClassifier()
RETURNS sysname
WITH SCHEMABINDING
AS
BEGIN
    DECLARE @grp sysname = 'default';

    -- Si la app se llama 'AnalisisBatch' o el usuario es 'analista'
    IF APP_NAME() = 'AnalisisBatch'
       OR SUSER_SNAME() = 'analista'
        SET @grp = 'RG_Analitico_Group';

    RETURN @grp;
END;
GO

PRINT '✓ Función clasificadora creada';
```

00 % 1 0 Línea: 74 Carácter: 1

Mensajes

```
? Función clasificadora creada

Hora de finalización: 2025-12-03T08:30:39.4044160-05:00
```

```sql
SELECT
    name AS NombrePool,
    min_cpu_percent AS MinCPU,
    max_cpu_percent AS MaxCPU,
    cap_cpu_percent AS LimiteCPU
FROM sys.resource_governor_resource_pools;
GO

SELECT
    name AS NombreGrupo,
    pool_id AS PoolId
FROM sys.resource_governor_workload_groups;
GO
```

00 % 1 0

Resultados Mensajes

	NombrePool	MinCPU	MaxCPU	LimiteCPU
1	internal	0	100	100
2	default	0	100	100
3	RG_Analitico_Pool	0	20	20

	NombreGrupo	PoolId
1	internal	1
2	default	2
3	RG_Analitic...	256

3. Justificación técnica

- Resource Governor controla recursos por pool y grupo; CAP_CPU_PERCENT limita CPU.

- Classifier enruta sesiones analíticas.

4. **Buenas prácticas**

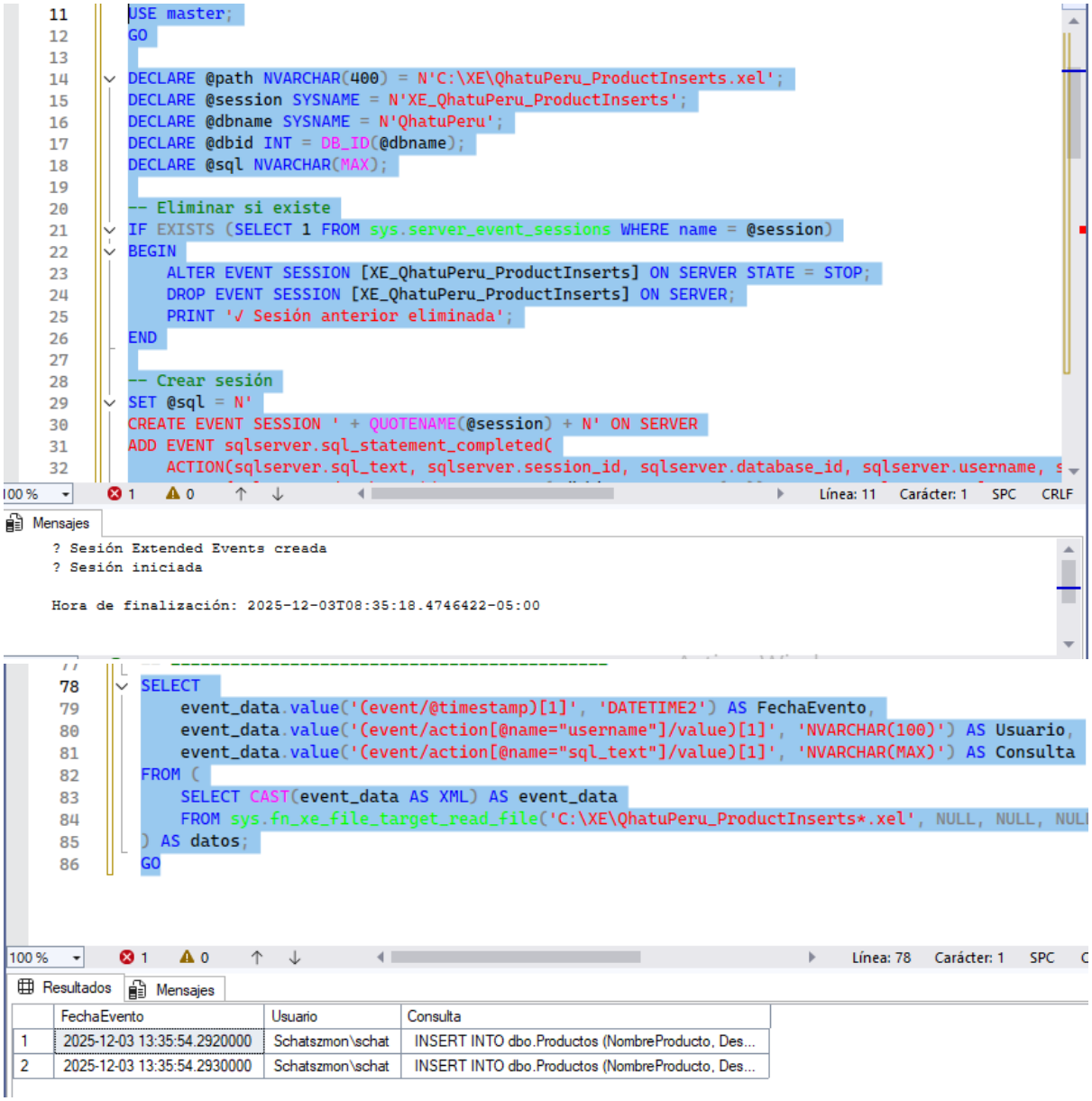
- Probar en staging; función classifier debe ser ligera.
- Monitorear impactos y ajustar porcentajes.

Proyecto 10 — Auditoría ligera de inserciones en Productos con Extended Events

1. **Enunciado**
   Auditar inserciones en `Productos` usando Extended Events sin afectar rendimiento.
2. **Script T-SQL (dinámico para ruta)**

```
USE master;
GO

DECLARE @path NVARCHAR(400) = N'C:\XE\QhatuPeru_ProductInserts.xel';
-- ajusta
DECLARE @session SYSNAME = N'XE_QhatuPeru_ProductInserts';
DECLARE @dbname SYSNAME = N'QhatuPeru';
DECLARE @dbid INT = DB_ID(@dbname);
DECLARE @sql NVARCHAR(MAX);

-- Eliminar si existe
IF EXISTS (SELECT 1 FROM sys.server_event_sessions WHERE name =
@session)
BEGIN
    ALTER EVENT SESSION [XE_QhatuPeru_ProductInserts] ON SERVER
STATE = STOP;
    DROP EVENT SESSION [XE_QhatuPeru_ProductInserts] ON SERVER;
END

-- Crear sesión con filtro por database y por texto (mínimo)
SET @sql = N'
CREATE EVENT SESSION ' + QUOTENAME(@session) + N' ON SERVER
ADD EVENT sqlserver.sql_statement_completed(
    ACTION(sqlserver.sql_text, sqlserver.session_id,
sqlserver.database_id, sqlserver.username,
sqlserver.client_app_name)
    WHERE (sqlserver.database_id = ' + CAST(@dbid AS NVARCHAR(10)) +
```

```
N' AND sqlserver.sql_text LIKE ''%INSERT%INTO%Productos%'')
)
ADD TARGET package0.event_file(SET filename = N''' +
REPLACE(@path,'''','''''') + N''', max_file_size=(50),
max_rollover_files=(10))
WITH (MAX_MEMORY=2048 KB,
EVENT_RETENTION_MODE=ALLOW_SINGLE_EVENT_LOSS, MAX_DISPATCH_LATENCY=1
SECONDS);
';

EXEC sp_executesql @sql;

ALTER EVENT SESSION [XE_QhatuPeru_ProductInserts] ON SERVER STATE =
START;
GO
```



```sql
USE master;
GO

DECLARE @path NVARCHAR(400) = N'C:\XE\QhatuPeru_ProductInserts.xel';
DECLARE @session SYSNAME = N'XE_QhatuPeru_ProductInserts';
DECLARE @dbname SYSNAME = N'QhatuPeru';
DECLARE @dbid INT = DB_ID(@dbname);
DECLARE @sql NVARCHAR(MAX);

-- Eliminar si existe
IF EXISTS (SELECT 1 FROM sys.server_event_sessions WHERE name = @session)
BEGIN
    ALTER EVENT SESSION [XE_QhatuPeru_ProductInserts] ON SERVER STATE = STOP;
    DROP EVENT SESSION [XE_QhatuPeru_ProductInserts] ON SERVER;
    PRINT '√ Sesión anterior eliminada';
END

-- Crear sesión
SET @sql = N'
CREATE EVENT SESSION ' + QUOTENAME(@session) + N' ON SERVER
ADD EVENT sqlserver.sql_statement_completed(
    ACTION(sqlserver.sql_text, sqlserver.session_id, sqlserver.database_id, sqlserver.username, s
```

100 % 1 0 Línea: 11 Carácter: 1 SPC CRLF

Mensajes

```
? Sesión Extended Events creada
? Sesión iniciada

Hora de finalización: 2025-12-03T08:35:18.4746422-05:00
```

```sql
SELECT
    event_data.value('(event/@timestamp)[1]', 'DATETIME2') AS FechaEvento,
    event_data.value('(event/action[@name="username"]/value)[1]', 'NVARCHAR(100)') AS Usuario,
    event_data.value('(event/action[@name="sql_text"]/value)[1]', 'NVARCHAR(MAX)') AS Consulta
FROM (
    SELECT CAST(event_data AS XML) AS event_data
    FROM sys.fn_xe_file_target_read_file('C:\XE\QhatuPeru_ProductInserts*.xel', NULL, NULL, NULL
) AS datos;
GO
```

100 % 1 0 Línea: 78 Carácter: 1 SPC

Resultados Mensajes

	FechaEvento	Usuario	Consulta
1	2025-12-03 13:35:54.2920000	Schatszmon\schat	INSERT INTO dbo.Productos (NombreProducto, Des...
2	2025-12-03 13:35:54.2930000	Schatszmon\schat	INSERT INTO dbo.Productos (NombreProducto, Des...

3. **Justificación técnica**

- XE es menos intrusivo que triggers T-SQL pesados.
- Se filtra por `database_id` y por patrón de texto para reducir ruido.

4. **Buenas prácticas**

- Si necesitas capturar valores antes/después considera CDC o trigger mínimo que escriba a una cola asincrónica.
- Revisar frecuencia de archivos `.xel` y archivarlos.