

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

```

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists the database structure for 'LAB04-PC14\MSSQLSERVERDEV'. The central pane contains T-SQL code for creating an event session and starting it. The bottom pane shows the 'Messages' window indicating the command completed successfully.

```

GO
-- 2. Crear la sesión de Extended Events
CREATE EVENT SESSION cap_consultas_lentas
ON SERVER
ADD EVENT sqlserver.sql_batch_completed(
    ACTION(sqlserver.sql_text, sqlserver.database_id, sqlserver.client_app_name)
    WHERE (duration > 1000) -- milisegundos
),
ADD EVENT sqlserver.rpc_completed(
    ACTION(sqlserver.sql_text, sqlserver.database_id, sqlserver.client_app_name)
    WHERE (duration > 1000)
)
ADD TARGET package0.event_file(
    SET filename = 'C:\XE\cap_consultas_lentas.xel'
);
GO
-- 3. Iniciar la sesión
ALTER EVENT SESSION cap_consultas_lentas ON SERVER STATE = START;
GO

```

Messages

Commands completed successfully.
Completion time: 2025-11-27T10:11:38.7232621-05:00

3. Justificación técnica

- `duration` está en microsegundos: $> 1,000,000 = 1\text{s}$.
- 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.

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the database 'LAB04-PC14\MSQLSERVERDEV' is selected, showing various system and user objects like 'syscolumns', 'sysconstraints', etc. In the center pane, a query window titled 'SQLQuery5.sql - LA..PC14\USER 17 (51)' is open with the following T-SQL code:

```
WAITFOR DELAY '00:00:02';
SELECT * FROM sys.objects;
```

The results grid displays a large list of system objects from the 'sys' schema, including their names, object IDs, principal IDs, schema IDs, parent object IDs, types, descriptions, creation dates, modification dates, and publication status. The table has columns such as name, object_id, principal_id, schema_id, parent_object_id, type, type_desc, create_date, modify_date, is_ms_shipped, is_published, and is_schema_published.

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
```

```

GO
-- ELIMINA EL ÍNDICE POR APELLIDOS SI YA EXISTE
IF EXISTS (
    SELECT name FROM sys.indexes
    WHERE name = 'IX_Clientes_Apellidos' AND object_id = OBJECT_ID('Clientes')
)
DROP INDEX IX_Cientes_Apellidos ON Clientes;
GO
-- ÍNDICE NO CLUSTERED PARA BÚSQUEDA POR DNI
CREATE NONCLUSTERED INDEX IX_Cientes_DNI
ON Clientes (DNI);
GO
-- ÍNDICE NO CLUSTERED PARA BÚSQUEDA POR APELLIDOS
CREATE NONCLUSTERED INDEX IX_Cientes_Apellidos
ON Clientes (Apellido);
GO
SELECT name AS Indice, type_desc
FROM sys.indexes
WHERE object_id = OBJECT_ID('Clientes');

```

3. Justificación técnica

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

```

GO
-- ELIMINA EL ÍNDICE POR APELLIDOS SI YA EXISTE
IF EXISTS (
    SELECT name FROM sys.indexes
    WHERE name = 'IX_Clientes_Apellidos' AND object_id = OBJECT_ID('Clientes')
)
DROP INDEX IX_Cientes_Apellidos ON Clientes;
GO
-- ÍNDICE NO CLUSTERED PARA BÚSQUEDA POR DNI
CREATE NONCLUSTERED INDEX IX_Cientes_DNI
ON Clientes (DNI);
GO
-- ÍNDICE NO CLUSTERED PARA BÚSQUEDA POR APELLIDOS
CREATE NONCLUSTERED INDEX IX_Cientes_Apellidos
ON Clientes (Apellido);
GO
SELECT name AS Indice, type_desc
FROM sys.indexes
WHERE object_id = OBJECT_ID('Clientes');

```

Indice	type_desc
PK_Cientes_71ABD0A72178C8CC	CLUSTERED
IX_Cientes_Apellidos_Nombre	NONCLUSTERED
IX_Cientes_DNI	NONCLUSTERED
IX_Cientes_Apellidos	NONCLUSTERED

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;
```

The screenshot shows the Microsoft SQL Server Management Studio (SSMS) interface. The Object Explorer on the left displays the database structure for 'LAB04-PC14\MSSQLSERVERDEV'. Under the 'QhatuPeru' database, the 'Tables' node is expanded, showing tables like 'dbo.ARTICULO', 'dbo.Clientes', and 'dbo.CIENCIAS'. The 'dbo.Clientes' table is further expanded to show columns such as 'ClienteID', 'DNI', 'Nombre', 'Apellidos', 'Telefono', 'Email', and 'FechaRegistro'. The 'Results' tab in the bottom right shows the output of a SQL query that retrieves fragmentation statistics for tables in the 'QhatuPeru' database.

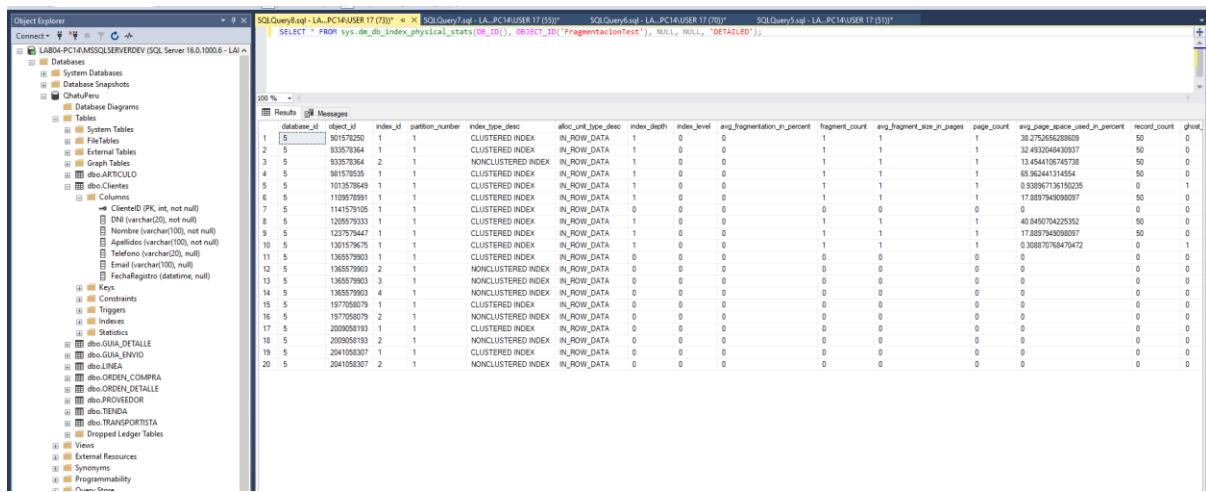
```

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.



database_id	object_id	index_id	partition_number	index_type_desc	index_type	index_depth	index_level	avg_fragmentation_in_percent	fragment_count	avg_fragment_size_in_pages	page_count	avg_page_size_kb_avg_in_percent	record_count	ghost
1	5	901572050	1	CLUSTERED INDEX	IN_ROW_DATA	1	0	0	1	1	1	38.275265220809	50	0
2		933578364	1	1	CLUSTERED INDEX	IN_ROW_DATA	1	0	1	1	1	32.49304843937	50	0
3	5	933578364	2	1	NONCLUSTERED INDEX	IN_ROW_DATA	1	0	0	1	1	13.454105745738	50	0
4	5	98157855	1	1	CLUSTERED INDEX	IN_ROW_DATA	1	0	1	1	1	65.962441514554	50	0
5	5	101578649	1	1	CLUSTERED INDEX	IN_ROW_DATA	1	0	0	1	1	0.3938717315025	0	1
6	5	1098578991	1	1	CLUSTERED INDEX	IN_ROW_DATA	1	0	0	1	1	17.893794309097	50	0
7	5	1157578991	1	1	CLUSTERED INDEX	IN_ROW_DATA	1	0	0	0	0	0	0	0
8	5	1208579333	1	1	CLUSTERED INDEX	IN_ROW_DATA	1	0	0	1	1	46.84070423532	50	0
9	5	123579447	1	1	CLUSTERED INDEX	IN_ROW_DATA	1	0	1	1	1	17.893794309097	50	0
10	5	1301579675	1	1	CLUSTERED INDEX	IN_ROW_DATA	1	0	0	1	1	0.30087078470472	0	1
11	5	1365579803	1	1	CLUSTERED INDEX	IN_ROW_DATA	0	0	0	0	0	0	0	0
12	5	1365579903	2	1	NONCLUSTERED INDEX	IN_ROW_DATA	0	0	0	0	0	0	0	0
13	5	1365579903	3	1	NONCLUSTERED INDEX	IN_ROW_DATA	0	0	0	0	0	0	0	0
14	5	1365579903	4	1	NONCLUSTERED INDEX	IN_ROW_DATA	0	0	0	0	0	0	0	0
15	5	197755079	1	1	CLUSTERED INDEX	IN_ROW_DATA	0	0	0	0	0	0	0	0
16	5	197755079	2	1	NONCLUSTERED INDEX	IN_ROW_DATA	0	0	0	0	0	0	0	0
17	5	2009508193	1	1	CLUSTERED INDEX	IN_ROW_DATA	0	0	0	0	0	0	0	0
18	5	2009508193	2	1	NONCLUSTERED INDEX	IN_ROW_DATA	0	0	0	0	0	0	0	0
19	5	2041585307	1	1	CLUSTERED INDEX	IN_ROW_DATA	0	0	0	0	0	0	0	0
20	5	2041585307	2	1	NONCLUSTERED INDEX	IN_ROW_DATA	0	0	0	0	0	0	0	0

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('X 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 ProductId, 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 ProductId, NombreProducto, Stock FROM dbo.Productos;
147  GO

```

Resultados

	Ventald	Clientld	FechaVenta	Total
1	1	2001	2025-12-03 07:53:12.960	170.00

Mensajes

	Detalleld	Ventald	ProductId	Cantidad	PrecioUnitario	Subtotal
1	1	1	101	2	85.00	170.00

	ProductId	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	Audifonos 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;
```

```
1 USE master;
2 GO
3
4 -- Ver bloqueos activos
5 v SELECT
6     r.blocking_session_id AS BloqueoOrigen,
7     r.session_id AS SesionBloqueada,
8     r.wait_type AS TipoEspera,
9     r.wait_time / 1000 AS SegundosEsperando,
10    DB_NAME(r.database_id) AS BaseDatos
11   FROM sys.dm_exec_requests r
12 WHERE r.blocking_session_id <> 0;
13 GO
```

0 % No se encontraron problemas.

Resultados Mensajes

	BloqueoOrigen	SesionBloqueada	TipoEspera	SegundosEsperando	BaseDatos
	54	61	LCK_M_X	20	QhatuPeru

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
```

ProductoId	Nombre Producto	TotalCantidad	TotalVenta
1	Monitor Samsung 24"	40	26000.00
2	Laptop HP 15"	10	25000.00
3	Audifonos Sony	50	6000.00
4	Tecclado Mecánico	30	4500.00
5	Mouse Logitech	22	1870.00

Microsoft SQL Server 2005 XML Showplan
1 <ShowPlanXML xmlns="http://schemas.microsoft.com...>

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
```

```
-- =====
10  -- 2. CREAR EL ÍNDICE OPTIMIZADO
11  -----
12  IF NOT EXISTS (SELECT 1 FROM sys.indexes WHERE name = 'IX_Ventas_Cliente_Fecha' AND object_id = OBJECT_ID('dbo.Ventas'))
13  BEGIN
14      CREATE NONCLUSTERED INDEX IX_Ventas_Cliente_Fecha
15          ON dbo.Ventas (ClienteId, FechaVenta)
16              INCLUDE (Total);
17
18      PRINT 'Índice IX_Ventas_Cliente_Fecha creado';
19
20  END
21  ELSE
22      PRINT 'El índice ya existe';
23
24  GO
25
26  -----
```

1% ▾ 1 ▲ 0 ↑ ↓ → Línea: 24 Carácter: 1 SPC LF

| Mensajes

? Índice IX_Ventas_Cliente_Fecha creado

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

```
-- =====
27 EXEC sp_helpindex 'dbo.Ventas';
28 GO
29
30
31  -----
```

32 4. PROBAR CONSULTA OPTIMIZADA

```
33  -----
34 SET STATISTICS IO ON;
35 SET STATISTICS TIME ON;
```

% ▾ 1 ▲ 0 ↑ ↓ ← → Línea: 2

| 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	VentId

```

37    -- Consulta que usa el índice
38    SELECT
39        ClienteId,
40        COUNT(*) AS NumeroVentas,
41        SUM(Total) AS TotalVentas
42    FROM dbo.Ventas
43    WHERE ClienteId = 2001
44        AND FechaVenta BETWEEN '2025-01-01' AND '2025-12-31'
45    GROUP BY ClienteId;
46
47    SET STATISTICS IO OFF;
48    SET STATISTICS TIME OFF;
49    GO
50
51    --- =====
52    -- 5. VER SI USA EL ÍNDICE (plan de ejecución)
53    --- =====

```

100 % ✖ 1 ⚠ 0 ↑ ↓ ↶

	ClienteId	NumeroVentas	TotalVentas
1	2001	18	19900.00

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

```

```

10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
-----2. CREAR ESTADÍSTICA SOBRE PRECIOVENTA-----
-----=====
13 IF NOT EXISTS (SELECT 1 FROM sys.stats WHERE object_id = OBJECT_ID('dbo.Productos') AND name = 'S
BEGIN
    CREATE STATISTICS STAT_PrecioVenta ON dbo.Productos (PrecioVenta) WITH FULLSCAN;
    PRINT 'Estadística STAT_PrecioVenta creada';
END
ELSE
BEGIN
    UPDATE STATISTICS dbo.Productos (STAT_PrecioVenta) WITH FULLSCAN;
    PRINT 'Estadística STAT_PrecioVenta actualizada';
END
GO
-----3. VERIFICAR QUE SE CREÓ-----
-----=====
SELECT
    s.name AS NombreEstadistica,
    ...
Mensajes
? Estadística STAT_PrecioVenta creada
Hora de finalización: 2025-12-03T08:24:08.8757571-05:00

```

The screenshot shows a SQL script being run in SSMS. The script creates or updates a statistics object named STAT_Precio on the Precio column of the dbo.Productos table. It includes a conditional block to check if the statistic exists and either creates it or updates it. A message is printed to the output window confirming the creation of the statistic. The status bar at the bottom right indicates the current line (Línea: 13), character (Carácter: 1), and file endings (SPC, CRLF).

```

25  --- =====
26  -- 3. VERIFICAR QUE SE CREÓ
27  --- =====
28  SELECT
29      s.name AS NombreEstadistica,
30      COL_NAME(s.object_id, sc.column_id) AS Columna,
31      s.auto_created AS AutoCreada,
32      s.user_created AS CreadaPorUsuario,
33      STATS_DATE(s.object_id, s.stats_id) AS UltimaActualizacion
34  FROM sys.stats s
35  JOIN sys.stats_columns sc ON s.stats_id = sc.stats_id AND s.object_id = sc.object_id
36  WHERE s.object_id = OBJECT_ID('dbo.Productos');
37  GO
38
39  --- =====
40  -- 4. VER DETALLE DE LA ESTADÍSTICA
41  --- =====

```

Línea: 28 Carácter: 1 SPC C

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

```

38
39  --- =====
40  -- 4. VER DETALLE DE LA ESTADÍSTICA
41  --- =====
42  DBCC SHOW_STATISTICS ('dbo.Productos', 'STAT_PrecioVenta');
43  GO
44
45  --- =====
46  -- 5. PROBAR CONSULTA POR RANGO DE PRECIO
47  --- =====
48  SET STATISTICS IO ON;
49
50  SELECT ProductoId, NombreProducto, PrecioVenta
51  FROM dbo.Productos
52  WHERE PrecioVenta BETWEEN 100 AND 500;
53
54  SET STATISTICS IO OFF;
55  GO

```

Línea: 42 Carácter: 1 SPC CRLF

Resultados Mensajes

	650.00	0	1	0	1
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

Activar Windows

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

```

The screenshot shows the SQL query being run in a query editor. The code creates a resource pool if it does not exist, with specific CPU usage parameters. A message is printed to the output window upon successful creation.

```

9  IF NOT EXISTS (SELECT 1
0   FROM sys.resource_governor_resource_pools
1   WHERE name = 'RG_Analitico_Pool')
2  BEGIN
3      CREATE RESOURCE POOL RG_Analitico_Pool
4      WITH (
5          MIN_CPU_PERCENT = 0,
6          MAX_CPU_PERCENT = 20,
7          CAP_CPU_PERCENT = 20
8      );
9      PRINT '/ Resource Pool RG_Analitico_Pool creado';
0  END
1  ELSE
2      PRINT 'Resource Pool ya existe';
3  GO
4
5  =====

```

Mensajes

```

Resource Pool RG_Analitico_Pool creado

```

Hora de finalización: 2025-12-03T08:29:55.9043218-05:00

The screenshot shows the SQL query being run in a query editor. The code creates a workload group if it does not exist, using the previously created resource pool. A message is printed to the output window upon successful creation.

```

37  =====
38  IF NOT EXISTS (SELECT 1
39   FROM sys.resource_governor_workload_groups
40   WHERE name = 'RG_Analitico_Group')
41  BEGIN
42      CREATE WORKLOAD GROUP RG_Analitico_Group
43          USING RG_Analitico_Pool;
44      PRINT '/ Workload Group RG_Analitico_Group creado';
45  END
46  ELSE
47      PRINT 'Workload Group ya existe';
48  GO
49
50  =====

```

Mensajes

```

? Workload Group RG_Analitico_Group creado

```

Hora de finalización: 2025-12-03T08:30:18.1373568-05:00

```

53  IF OBJECT_ID('dbo.RGClassifier', 'FN') IS NOT NULL
54      DROP FUNCTION dbo.RGClassifier;
55  GO
56
57  CREATE FUNCTION dbo.RGClassifier()
58  RETURNS sysname
59  WITH SCHEMABINDING
60  AS
61  BEGIN
62      DECLARE @grp sysname = 'default';
63
64      -- Si la app se llama 'AnalisisBatch' o el usuario es 'analista'
65      IF APP_NAME() = 'AnalisisBatch'
66          OR SUSER_SNAME() = 'analista'
67          SET @grp = 'RG_Analitico_Group';
68
69      RETURN @grp;
70  END;
71  GO
72
73  PRINT '/ Función clasificadora creada';
74

```

00 % □ 1 ▲ 0 ↑ ↓ ▶ Línea: 74 Carácter: 1

Mensajes

? Función clasificadora creada

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

```

-- 89
90  SELECT
91      name AS NombrePool,
92      min_cpu_percent AS MinCPU,
93      max_cpu_percent AS MaxCPU,
94      cap_cpu_percent AS LimiteCPU
95  FROM sys.resource_governor_resource_pools;
96
97  SELECT
98      name AS NombreGrupo,
99      pool_id AS PoolId
100
101
102

```

00 % □ 1 ▲ 0 ↑ ↓ ▶

Resultados

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

	NombreGrupo	Poollid
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
```

```

11 USE master;
12 GO
13
14 DECLARE @path NVARCHAR(400) = N'C:\XE\QhatuPeru_ProductInserts.xel';
15 DECLARE @session SYSNAME = N'XE_QhatuPeru_ProductInserts';
16 DECLARE @dbname SYSNAME = N'QhatuPeru';
17 DECLARE @dbid INT = DB_ID(@dbname);
18 DECLARE @sql NVARCHAR(MAX);
19
20 -- Eliminar si existe
21 IF EXISTS (SELECT 1 FROM sys.server_event_sessions WHERE name = @session)
22 BEGIN
23     ALTER EVENT SESSION [XE_QhatuPeru_ProductInserts] ON SERVER STATE = STOP;
24     DROP EVENT SESSION [XE_QhatuPeru_ProductInserts] ON SERVER;
25     PRINT '✓ Sesión anterior eliminada';
26 END
27
28 -- Crear sesión
29 SET @sql = N'
30 CREATE EVENT SESSION ' + QUOTENAME(@session) + N' ON SERVER
31 ADD EVENT sqlserver.sql_statement_completed(
32     ACTION(sqlserver.sql_text, sqlserver.session_id, sqlserver.database_id, sqlserver.username, s

```

Mensajes

- ? Sesión Extended Events creada
- ? Sesión iniciada

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


```

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

```

Resultados

	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.