Automatización sobre Northwind usando Azure SQL

Curso: Bases de Datos
Grupo: 3 estudiantes

Contexto del problema

La empresa **Northwind Traders** requiere un rediseño avanzado de procesos internos en su base de datos. Ustedes, como equipo de desarrolladores, deben construir soluciones automatizadas para análisis, gestión y mantenimiento de datos usando procedimientos almacenados, funciones y triggers avanzados.

Cada uno de los elementos debe aportar lógica empresarial real y estar correctamente diseñado, optimizado y documentado.

Requisitos generales

- Restauren la base de datos Northwind en su Azure SQL.
- Revisen las tablas principales: Products, Orders, Order Details, Customers, Employees, Suppliers.
- Todo el trabajo debe cumplir con buenas prácticas, manejo de errores y transacciones donde corresponda.

1. Procedimientos Almacenados
Construir procedimientos almacenados para llevar a cabo en la base de datos:

1.1 RegistrarNuevoPedido

- Inserta un nuevo pedido (Orders) y sus detalles (Order Details).
- Valida disponibilidad de stock y actualiza UnitsInStock.
- Usa transacción para asegurar atomicidad.

1.2 ActualizarEstadoPedido

- Permite actualizar el estado (ShippedDate, ShipVia) de un pedido.
- Solo permite cambios si el pedido no está aún enviado.

1.3 GenerarReporteVentasPorProducto

- Devuelve un resumen de ventas totales (cantidad, ingresos) por producto para un rango de fechas dado.

1.4 ListarTopClientes

- Devuelve los 5 clientes con mayores compras (en valor) en los últimos 12 meses.

1.5 CalcularBonificacionesEmpleados

- Calcula e inserta en una tabla EmployeeBonuses (deben crearla) bonificaciones para empleados basadas en el número de pedidos gestionados en el mes.

2. Funciones

Construir funciones para llevar a cabo en la base de datos:

2.1 CalcularValorTotalPedido(@OrderID)

- Devuelve el valor total del pedido (considerando descuentos).

2.2 ObtenerPromedioVentasPorProducto(@ProductID)

- Devuelve el promedio de ventas mensuales de un producto en el último año.

2.3 ObtenerPedidosPorEmpleado(@EmployeeID)

- Devuelve una tabla con los pedidos gestionados por un empleado, incluyendo cliente, fecha y valor total.

2.4 CalcularMargenProducto(@ProductID)

- Devuelve el margen de ganancia estimado (precio unitario vs. costo) para un producto.

2.5 ObtenerHistorialCambiosPrecio(@ProductID)

- Devuelve un historial de cambios de precio registrados en PriceChangeLog (deben crearla).

3. Triggers

Construir triggers para llevar a cabo en la base de datos:

3.1 Trigger en Products – Cambio de Precio

- Cuando se actualice el UnitPrice, registrar cambio en PriceChangeLog con precio anterior, nuevo, fecha, usuario.

3.2 Trigger en Orders – Eliminación de Pedido

- Cuando se elimine un pedido, eliminar también sus detalles en Order Details y registrar en OrderDeletionLog.

3.3 Trigger en Customers – Cambio de Categoría

- Cuando se actualice la categoría de un cliente (campo a crear, por ejemplo, CustomerCategory), guardar registro del cambio en CustomerCategoryLog.

3.4 Trigger en Suppliers – Inactivación de Proveedor

- Cuando se marque un proveedor como inactivo (campo a crear, por ejemplo, IsActive), generar log en SupplierStatusLog.

3.5 Trigger en EmployeeBonuses – Inserción Nueva Bonificación

- Cuando se inserte un nuevo registro de bonificación, validar que no exista duplicado para el mismo mes y empleado. Si existe, cancelar la operación.

4. Documentación y entregables

- Archivo .sql con todo el código, comentado claramente.
- Documento .pdf explicando:
    - Qué hace cada procedimiento, función y trigger.
    - Pruebas realizadas (incluyendo ejemplos, resultados).
    - Roles de cada integrante del equipo.

Notas finales

- El proyecto debe ser realista: no se aceptarán implementaciones triviales.
- El equipo debe poder explicar cada parte del código (se harán preguntas aleatorias en clase).
- Usen **transacciones, TRY-CATCH, validaciones** donde aplique.
- El trabajo es grupal y debe reflejar colaboración.