Informe del Proyecto Integrador Profesional en Bases de Datos

1. Datos Generales del Proyecto

Nombre del Proyecto: Sistema Integral de Gestión para Entornos Profesionales con Bases de Datos Relacionales

Temática Asignada: Restaurante

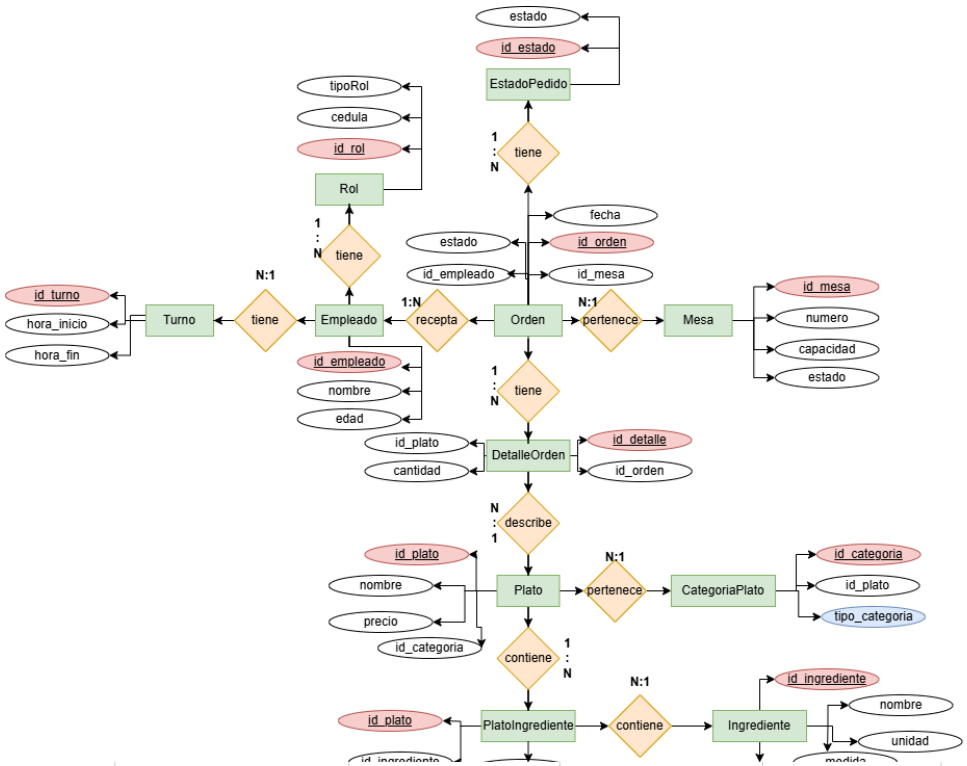
Nombres: Andrés Panchi y Mercy PerugacHI

2. Objetivo General

Diseñar e implementar un sistema profesional de base de datos, utilizando SQL Server, que abarque desde el modelado lógico hasta la protección contra ataques, aplicando buenas prácticas de rendimiento, seguridad, respaldo, programación avanzada y roles profesionales reales en una temática asignada.

3. Requerimientos Técnicos Específicos

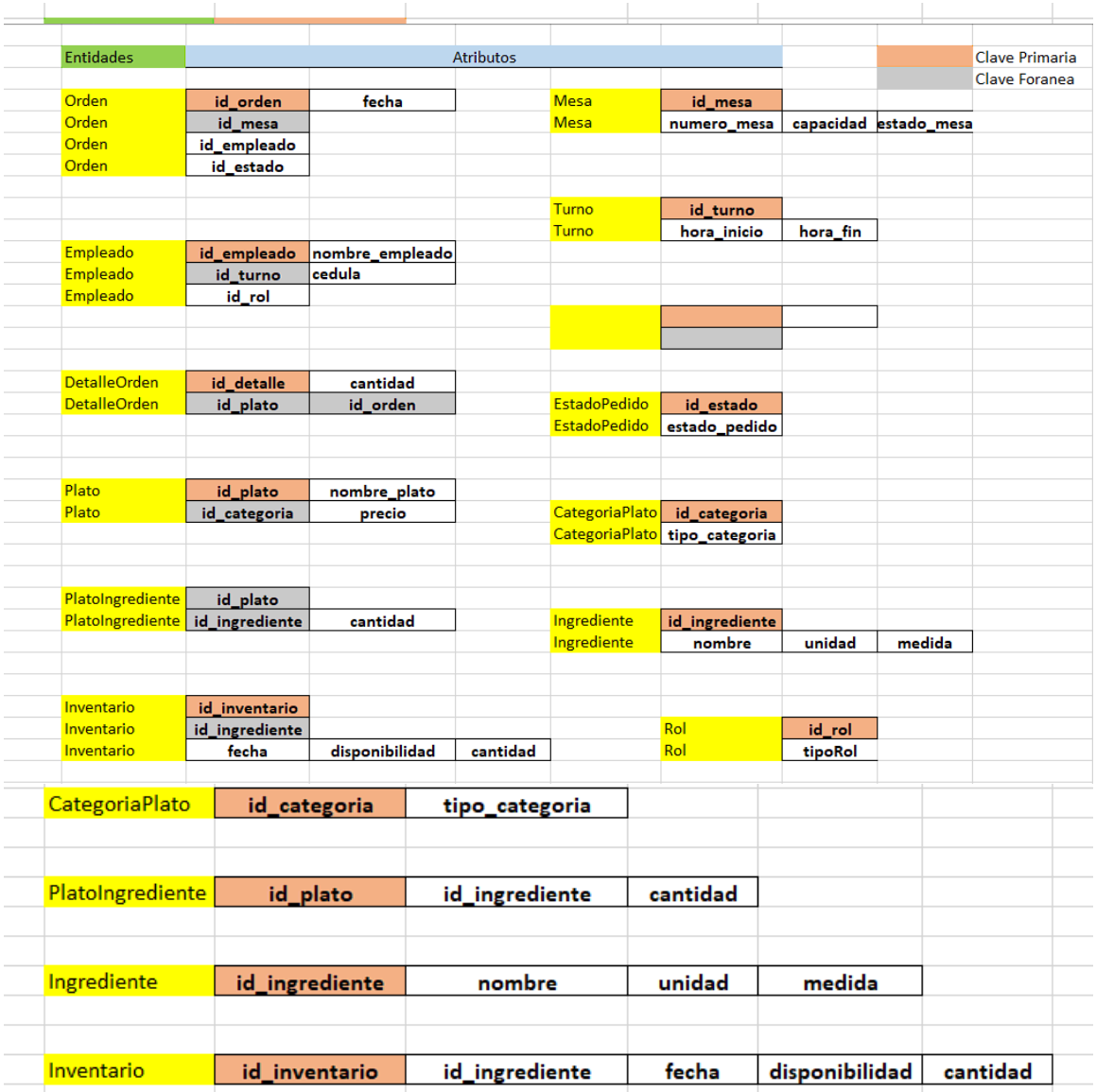
- Modelado y Normalización: Modelo entidad-relación en 3FN, integridad referencial, relaciones adecuadas.
  MODELO ENTIDAD RELACION



Entidades	Atributos				
Orden	**id_orden**	**fecha**	**id_mesa**	**id_empleado**	**id_estado**
Mesa	**id_mesa**	**numero_mesa**	**capacidad**	**estado_mesa**	
Empleado	**id_empleado**	**nombre_empleado**	**id_turno**	**id_rol**	
Turno	**id_turno**	**hora_inicio**	**hora_fin**		
Empleado	**id_rol**	**nombre_rol**			
EstadoPedido	**id_estado**	**estado_pedido**			
DetalleOrden	**id_orden**	**id_plato**	**cantidad**		
Plato	**id_plato**	**nombre_plato**	**id_categoria**		
CategoriaPlato	**id_categoria**	**tipo_categoria**			
PlatoIngrediente	**id_plato**	**id_ingrediente**	**cantidad**		
Ingrediente	**id_ingrediente**	**nombre**	**unidad**	**medida**	
Inventario	**id_inventario**	**id_ingrediente**	**fecha**	**disponibilidad**	**cantidad**

Primera Forma Normal 1FN

Segunda Forma Normal 2FN



Entidades	Atributos						Clave Primaria
							Clave Foranea
Orden	id_orden	fecha		Mesa	id_mesa		
Orden	id_mesa			Mesa	numero_mesa	capacidad	estado_mesa
Orden	id_empleado						
Orden	id_estado						
				Turno	id_turno		
				Turno	hora_inicio	hora_fin	
Empleado	id_empleado	nombre_empleado					
Empleado	id_turno	cedula					
Empleado	id_rol						
DetalleOrden	id_detalle	cantidad					
DetalleOrden	id_plato	id_orden		EstadoPedido	id_estado		
				EstadoPedido	estado_pedido		
Plato	id_plato	nombre_plato					
Plato	id_categoria	precio		CategoriaPlato	id_categoria		
				CategoriaPlato	tipo_categoria		
PlatoIngrediente	id_plato						
PlatoIngrediente	id_ingrediente	cantidad		Ingrediente	id_ingrediente		
				Ingrediente	nombre	unidad	medida
Inventario	id_inventario						
Inventario	id_ingrediente			Rol	id_rol		
Inventario	fecha	disponibilidad	cantidad	Rol	tipoRol		

CategoriaPlato	id_categoria	tipo_categoria

PlatoIngrediente	id_plato	id_ingrediente	cantidad

Ingrediente	id_ingrediente	nombre	unidad	medida

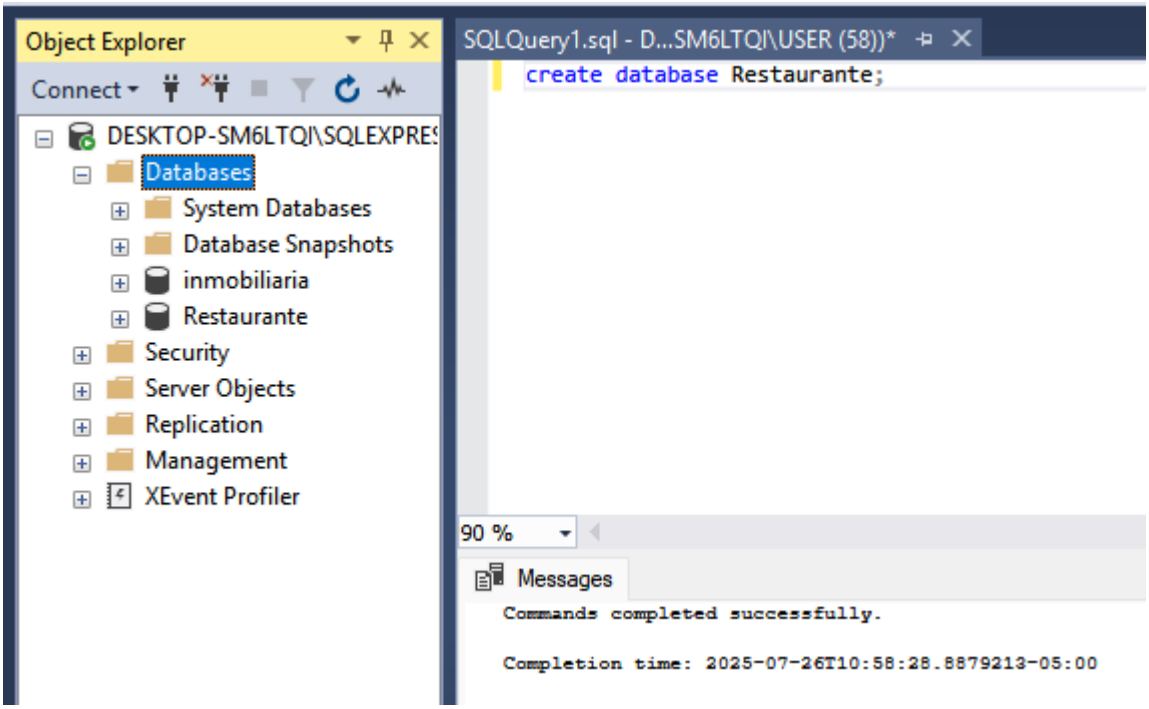
Inventario	id_inventario	id_ingrediente	fecha	disponibilidad	cantidad

Tercer Forma Normal 3FN

- Integridad y Restricciones: Aplicación estricta de restricciones y claves foráneas con ON DELETE/UPDATE.

- Definición de Estructura Base: Mínimo 10 tablas funcionales, campos obligatorios, catálogos independientes.

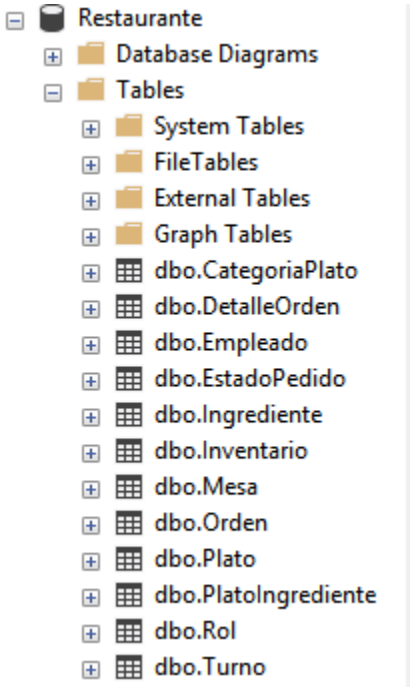
Creación de la base de datos Restaurante



Creación de las tablas

```sql
-- CREACION DE LAS TABLAS
create table Mesa (id_mesa INT IDENTITY(1,1) NOT NULL,
    numero_mesa int not null unique,
    capacidad int null,
    estado_mesa varchar(10),
    constraint PK_Mesa primary key (id_mesa)
);

create table Turno (id_turno int identity(1,1) not null,
    hora_inicio time not null,
    hora_fin time not null,
    constraint PK_Turno primary key (id_turno)
);

create table EstadoPedido(id_estado int identity(1,1) not null,
    estado_pedido varchar(15),
    constraint PK_EstadoPedido primary key (id_estado));


create table CategoriaPlato(id_categoria int identity(1,1) not null,
    tipo_categoria varchar(50) not null unique,
    constraint PK_CategoriaPlato primary key (id_categoria));

create table Ingrediente(id_ingrediente int identity(1,1) not null,
    nombre varchar(30) not null unique,
    unidad int not null,
    medida varchar(5) not null,
    constraint PK_Ingrediente primary key(id_ingrediente));

create table Rol (id_rol int identity(1,1)not null,
    tipoRol varchar(30),
    constraint PK_Rol primary key (id_rol));

create table Plato(id_plato int identity (1,1) not null,
    nombre_plato varchar(50),
    precio decimal(10,2)not null,
    id_categoria int not null,
    constraint PK_Plato primary key(id_plato),
    foreign key(id_categoria) references CategoriaPlato (id_categoria));


create table Orden(id_orden int identity(1,1) not null,
    fecha datetime not null default getdate(),
    id_mesa int not null,
    id_empleado int not null,
    id_estado int not null,
    constraint PK_Orden primary key(id_orden),
    foreign key(id_mesa)references Mesa(id_mesa),
    foreign key(id_empleado)references Empleado (id_empleado),
    foreign key(id_estado)references EstadoPedido(id_estado));
```

```sql
create table DetalleOrden(id_detalle int identity(1,1)not null,
    cantidad  int not null,
    id_plato int not null,
    id_orden int not null,
    constraint PK_DetalleOrden primary key(id_detalle),
    foreign key(id_plato) references Plato (id_plato),
    foreign key(id_orden) references Orden (id_orden));

create table PlatoIngrediente(id_plato int not null,
    id_ingrediente int not null,
    cantidad int not null,
    foreign key(id_plato)references Plato(id_plato),
    foreign key(id_ingrediente)references Ingrediente(id_ingrediente));

create table Inventario(id_inventario int identity(1,1)not null,
    fecha date not null default getdate(),
    disponibilidad varchar(50) not null,
    cantidad int not null,
    id_ingrediente int not null,
    constraint PK_Inventario primary key (id_inventario),
    foreign key(id_ingrediente)references Ingrediente(id_ingrediente));
create table Empleado(id_empleado int identity(1,1) not null,
    nombre_empleado varchar(100) not null,
    cedula int not null unique,
    id_rol int not null,
    id_turno int not null,
    constraint PK_Empleado primary key(id_empleado),
    foreign key(id_rol)references Rol (id_rol),
    foreign key(id_turno)references Turno(id_turno));
```



- Consultas: Mínimo 5, uso de JOINS.

-- 1. Obtener todas las órdenes con el nombre del empleado que la tomó y el estado del pedido.

SELECT

O.id_orden,

O.fecha,

M.numero_mesa,

E.nombre_empleado,

EP.estado_pedido

FROM Orden AS O

JOIN Mesa AS M ON O.id_mesa = M.id_mesa

JOIN Empleado AS E ON O.id_empleado = E.id_empleado

JOIN EstadoPedido AS EP ON O.id_estado = EP.id_estado;

```sql
-- 1. Obtener todas las órdenes con el nombre del empleado que la tomó y el estado del pedido.
SELECT
    O.id_orden,
    O.fecha,
    M.numero_mesa,
    E.nombre_empleado,
    EP.estado_pedido
FROM Orden AS O
JOIN Mesa AS M ON O.id_mesa = M.id_mesa
JOIN Empleado AS E ON O.id_empleado = E.id_empleado
JOIN EstadoPedido AS EP ON O.id_estado = EP.id_estado;
```

Results Messages

	id_orden	fecha	numero_mesa	nombre_empleado	estado_pedido
1	59	2025-04-08 19:41:21.183	11	Pedro González	Pendiente
2	87	2025-02-05 19:41:21.240	5	Juan Sánchez	Listo
3	116	2025-03-11 19:41:21.283	440	Juan Sánchez	Pendiente
4	142	2024-09-14 19:41:21.313	379	Juan Sánchez	Listo
5	283	2025-01-18 19:41:21.573	265	Juan Sánchez	Pagado
6	439	2025-03-08 19:41:21.853	199	Juan Sánchez	Cancelado
7	339	2024-12-03 19:41:21.693	406	Pedro Torres	Cancelado
8	281	2024-10-08 19:41:21.570	99	Pedro López	Servido
9	311	2025-01-16 19:41:21.633	396	Pedro López	Servido
10	389	2025-01-25 19:41:21.780	59	Pedro López	Listo
11	410	2025-04-14 19:41:21.807	387	Pedro López	Cancelado

Query executed successfully. DESKTOP-SM6L

-- 2. Listar los platos y su categoría.

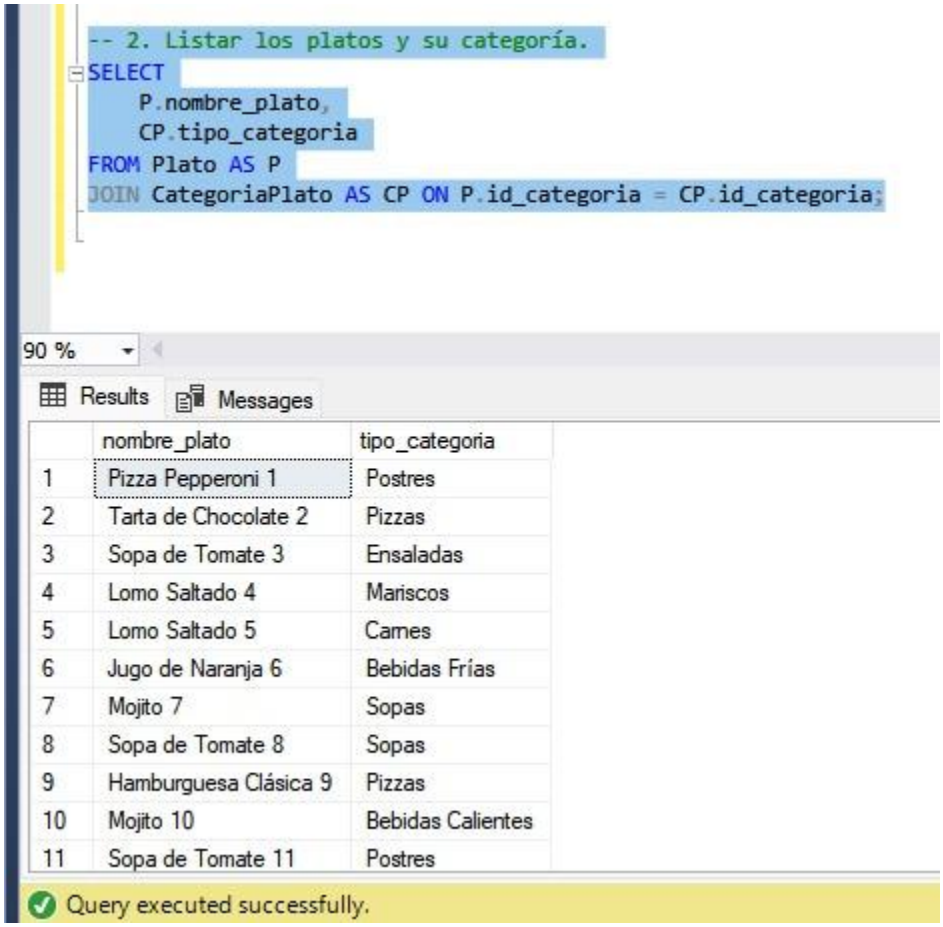
SELECT

P.nombre_plato,

CP.tipo_categoria

FROM Plato AS P

JOIN CategoriaPlato AS CP ON P.id_categoria = CP.id_categoria;



```
-- 2. Listar los platos y su categoría.
SELECT
    P.nombre_plato,
    CP.tipo_categoria
FROM Plato AS P
JOIN CategoriaPlato AS CP ON P.id_categoria = CP.id_categoria;
```

90 %

Results | Messages

	nombre_plato	tipo_categoria
1	Pizza Pepperoni 1	Postres
2	Tarta de Chocolate 2	Pizzas
3	Sopa de Tomate 3	Ensaladas
4	Lomo Saltado 4	Mariscos
5	Lomo Saltado 5	Carnes
6	Jugo de Naranja 6	Bebidas Frías
7	Mojito 7	Sopas
8	Sopa de Tomate 8	Sopas
9	Hamburguesa Clásica 9	Pizzas
10	Mojito 10	Bebidas Calientes
11	Sopa de Tomate 11	Postres

Query executed successfully.

-- 3. Ver el detalle de una orden específica, incluyendo los nombres de los platos y sus cantidades.

SELECT

O.id_orden,

P.nombre_plato,

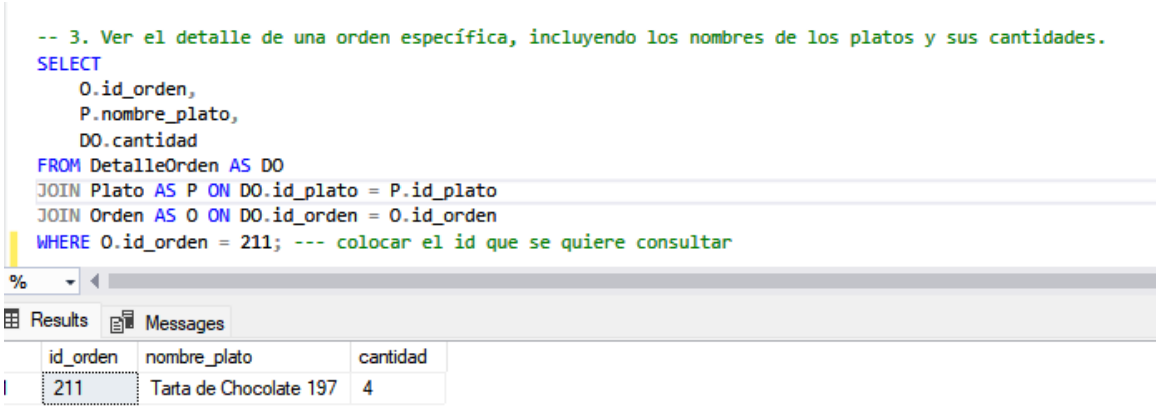
DO.cantidad

FROM DetalleOrden AS DO

JOIN Plato AS P ON DO.id_plato = P.id_plato

JOIN Orden AS O ON DO.id_orden = O.id_orden

WHERE O.id_orden = 1; -- Cambia 1 por el ID de la orden que quieras consultar



```sql
-- 3. Ver el detalle de una orden específica, incluyendo los nombres de los platos y sus cantidades.
SELECT
    O.id_orden,
    P.nombre_plato,
    DO.cantidad
FROM DetalleOrden AS DO
JOIN Plato AS P ON DO.id_plato = P.id_plato
JOIN Orden AS O ON DO.id_orden = O.id_orden
WHERE O.id_orden = 211; --- colocar el id que se quiere consultar
```

%

Results | Messages

id_orden	nombre_plato	cantidad
211	Tarta de Chocolate 197	4

-- 4. Obtener el inventario actual de los ingredientes, incluyendo el nombre del ingrediente.

SELECT

I.nombre AS nombre_ingrediente,

IV.cantidad,

IV.disponibilidad,

IV.fecha

FROM Inventario AS IV

JOIN Ingrediente AS I ON IV.id_ingrediente = I.id_ingrediente;

```sql
-- 4. Obtener el inventario actual de los ingredientes, incluyendo el nombre del ingrediente.
SELECT
    I.nombre AS nombre_ingrediente,
    IV.cantidad,
    IV.disponibilidad,
    IV.fecha
FROM Inventario AS IV
JOIN Ingrediente AS I ON IV.id_ingrediente = I.id_ingrediente;
```

0 %

Results | Messages

	nombre_ingrediente	cantidad	disponibilidad	fecha
1	Aceite2	53	Agotado	2025-07-05
2	Ajo4	380	En Stock	2025-07-12
3	Ajo4	387	Pedido	2025-07-01
4	Pasta6	305	Bajo Stock	2025-07-02
5	Tomate7	428	Pedido	2025-06-28
6	Tomate7	66	Agotado	2025-06-28
7	Cebolla8	330	En Stock	2025-07-22
8	Cebolla8	427	Agotado	2025-07-21
9	Ajo9	317	Pedido	2025-07-05
10	Plátano10	80	Bajo Stock	2025-07-22
11	Mantequilla11	184	Bajo Stock	2025-07-23
12	Pimienta13	96	Bajo Stock	2025-07-05
13	Pimienta13	415	Agotado	2025-07-18

Query executed successfully. DESKTOP-SM6LTC

-- 5. Mostrar los empleados y el rol que desempeñan, junto con su horario de turno.

```sql
SELECT
   E.nombre_empleado,
   R.tipoRol,
   T.hora_inicio,
   T.hora_fin
FROM Empleado AS E
JOIN Rol AS R ON E.id_rol = R.id_rol
JOIN Turno AS T ON E.id_turno = T.id_turno;
```

```sql
-- 5. Mostrar los empleados y el rol que desempeñan, junto con su horario de turno.
SELECT
    E.nombre_empleado,
    R.tipoRol,
    T.hora_inicio,
    T.hora_fin
FROM Empleado AS E
JOIN Rol AS R ON E.id_rol = R.id_rol
JOIN Turno AS T ON E.id_turno = T.id_turno;
```

90 %

Results | Messages

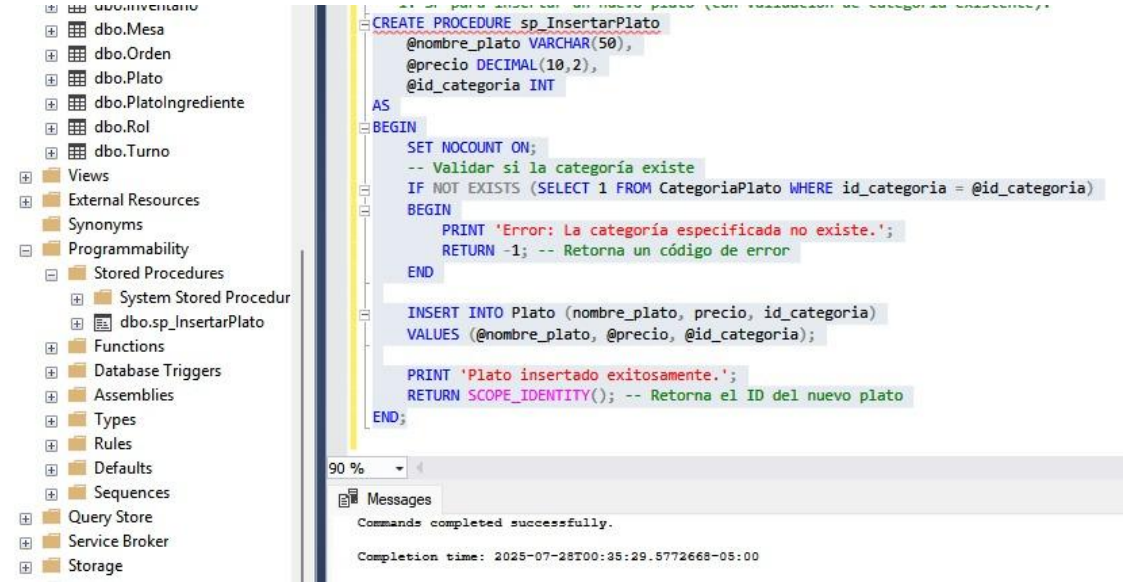
	nombre_empleado	tipoRol	hora_inicio	hora_fin
1	Pedro González	Mesero	07:32:00.0000000	16:32:00.0000000
2	Diego Torres	Lavaplatos	04:27:00.0000000	13:27:00.0000000
3	María Martínez	Gerente	16:21:00.0000000	01:21:00.0000000
4	Juan Sánchez	Hostess	09:27:00.0000000	20:27:00.0000000
5	Pedro García	Ayudante de Cocina	23:39:00.0000000	07:39:00.0000000
6	Pedro Torres	Lavaplatos	05:00:00.0000000	14:00:00.0000000
7	María Flores	Supervisor	18:06:00.0000000	02:06:00.0000000
8	Pedro López	Supervisor	11:36:00.0000000	20:36:00.0000000
9	Pedro Martínez	Mesero	03:06:00.0000000	08:06:00.0000000
10	Carlos López	Supervisor	06:50:00.0000000	13:50:00.0000000
11	Ana López	Cocinero	08:56:00.0000000	18:56:00.0000000
12	Laura Martínez	Supervisor	14:57:00.0000000	00:57:00.0000000
13	Diego Pérez	Cocinero	07:46:00.0000000	16:46:00.0000000

Query executed successfully.

- Procedimientos Almacenados: Mínimo 5, incluyendo validaciones, actualizaciones, reportes y transacciones.

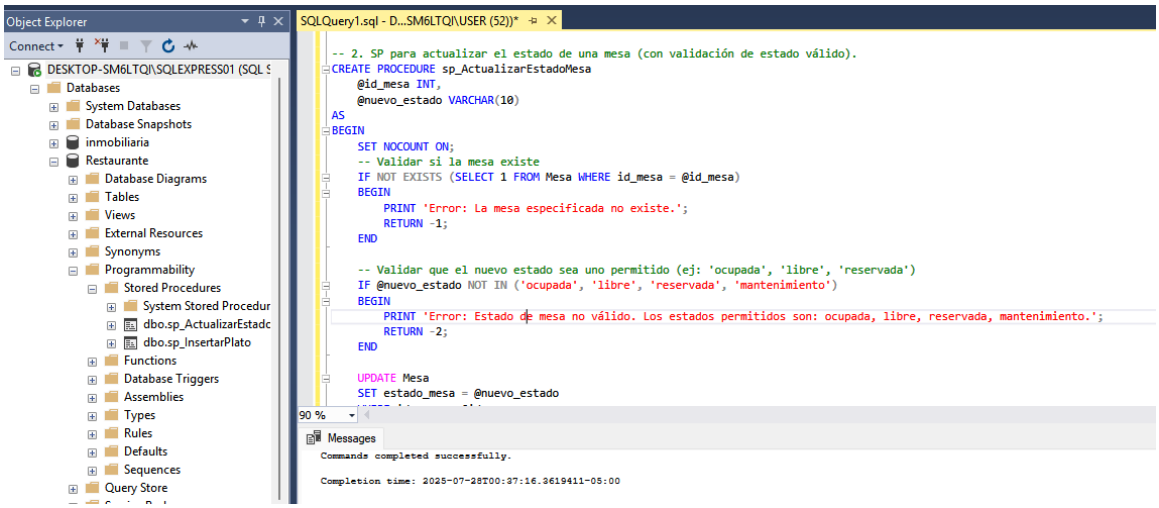
```sql
-- 1. SP para insertar un nuevo plato (con validación de categoría existente).
CREATE PROCEDURE sp_InsertarPlato
    @nombre_plato VARCHAR(50),
    @precio DECIMAL(10,2),
    @id_categoria INT
AS
BEGIN
    SET NOCOUNT ON;
    -- Validar si la categoría existe
```

```
    IF NOT EXISTS (SELECT 1 FROM CategoriaPlato WHERE id_categoria = @id_categoria)
    BEGIN
        PRINT 'Error: La categoría especificada no existe.';
        RETURN -1; -- Retorna un código de error
    END

    INSERT INTO Plato (nombre_plato, precio, id_categoria)
    VALUES (@nombre_plato, @precio, @id_categoria);

    PRINT 'Plato insertado exitosamente.';
    RETURN SCOPE_IDENTITY(); -- Retorna el ID del nuevo plato
END;
```

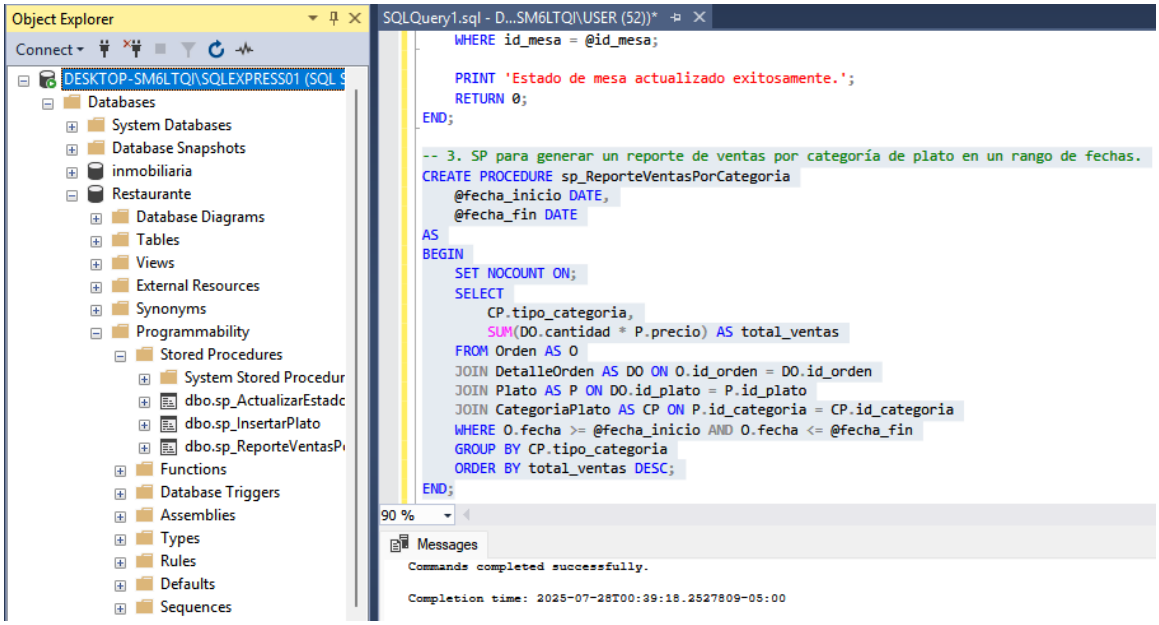



```
-- 2. SP para actualizar el estado de una mesa (con validación de estado válido).
CREATE PROCEDURE sp_ActualizarEstadoMesa
    @id_mesa INT,
    @nuevo_estado VARCHAR(10)
AS
BEGIN
    SET NOCOUNT ON;
```

```
    -- Validar si la mesa existe
    IF NOT EXISTS (SELECT 1 FROM Mesa WHERE id_mesa = @id_mesa)
    BEGIN
        PRINT 'Error: La mesa especificada no existe.';
        RETURN -1;
    END

    -- Validar que el nuevo estado sea uno permitido (ej: 'ocupada', 'libre', 'reservada')
    IF @nuevo_estado NOT IN ('ocupada', 'libre', 'reservada', 'mantenimiento')
    BEGIN
        PRINT 'Error: Estado de mesa no válido. Los estados permitidos son: ocupada, libre,
reservada, mantenimiento.';
        RETURN -2;
    END

    UPDATE Mesa
    SET estado_mesa = @nuevo_estado
    WHERE id_mesa = @id_mesa;

    PRINT 'Estado de mesa actualizado exitosamente.';
    RETURN 0;
END;
```




```
-- 3. SP para generar un reporte de ventas por categoría de plato en un rango de fechas.
CREATE PROCEDURE sp_ReporteVentasPorCategoria
    @fecha_inicio DATE,
    @fecha_fin DATE
AS
BEGIN
    SET NOCOUNT ON;
    SELECT
        CP.tipo_categoria,
        SUM(DO.cantidad * P.precio) AS total_ventas
    FROM Orden AS O
    JOIN DetalleOrden AS DO ON O.id_orden = DO.id_orden
    JOIN Plato AS P ON DO.id_plato = P.id_plato
    JOIN CategoriaPlato AS CP ON P.id_categoria = CP.id_categoria
    WHERE O.fecha >= @fecha_inicio AND O.fecha <= @fecha_fin
    GROUP BY CP.tipo_categoria
    ORDER BY total_ventas DESC;
END;
```



-- 4. SP para registrar una nueva orden y sus detalles (con transacción).

CREATE PROCEDURE sp_CrearOrdenConDetalles

@id_mesa INT,

@id_empleado INT,

@detalles_orden NVARCHAR(MAX) -- JSON o XML para los detalles: Ejemplo: '[{"id_plato":1, "cantidad":2}, {"id_plato":3, "cantidad":1}]'

AS

BEGIN

SET NOCOUNT ON;

SET XACT_ABORT ON; -- Abortar la transacción si ocurre un error

DECLARE @id_orden INT;

BEGIN TRY

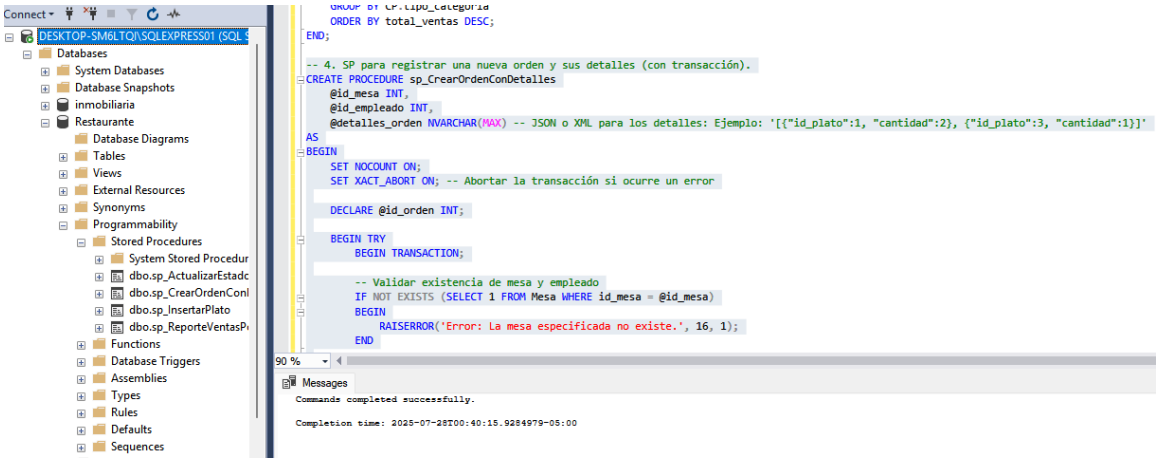
BEGIN TRANSACTION;

-- Validar existencia de mesa y empleado

IF NOT EXISTS (SELECT 1 FROM Mesa WHERE id_mesa = @id_mesa)

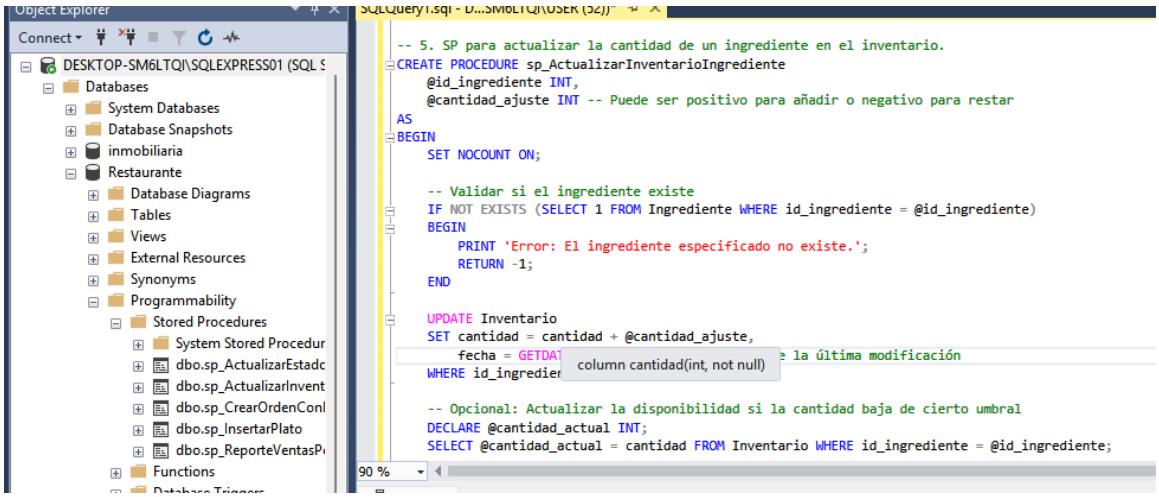
```
    BEGIN
        RAISERROR('Error: La mesa especificada no existe.', 16, 1);
    END

    IF NOT EXISTS (SELECT 1 FROM Empleado WHERE id_empleado = @id_empleado)
    BEGIN
        RAISERROR('Error: El empleado especificado no existe.', 16, 1);
    END

    -- Insertar la orden
    INSERT INTO Orden (id_mesa, id_empleado, id_estado)
    VALUES (@id_mesa, @id_empleado, 1); -- Asume 1 como 'Pendiente' en EstadoPedido
    SET @id_orden = SCOPE_IDENTITY();

    -- Insertar los detalles de la orden (ejemplo asumiendo JSON)
    INSERT INTO DetalleOrden (id_orden, id_plato, cantidad)
    SELECT
        @id_orden,
        JSON_VALUE(value, '$.id_plato'),
        JSON_VALUE(value, '$.cantidad')
    FROM OPENJSON(@detalles_orden);

    COMMIT TRANSACTION;
    PRINT 'Orden creada exitosamente con ID: ' + CAST(@id_orden AS VARCHAR);
    RETURN @id_orden;

END TRY
BEGIN CATCH
    ROLLBACK TRANSACTION;
```

```
    DECLARE @ErrorMessage NVARCHAR(MAX), @ErrorSeverity INT, @ErrorState INT;

    SELECT @ErrorMessage = ERROR_MESSAGE(), @ErrorSeverity = ERROR_SEVERITY(),
@ErrorState = ERROR_STATE();

    RAISERROR(@ErrorMessage, @ErrorSeverity, @ErrorState);

    RETURN -1;

  END CATCH;

END;
```



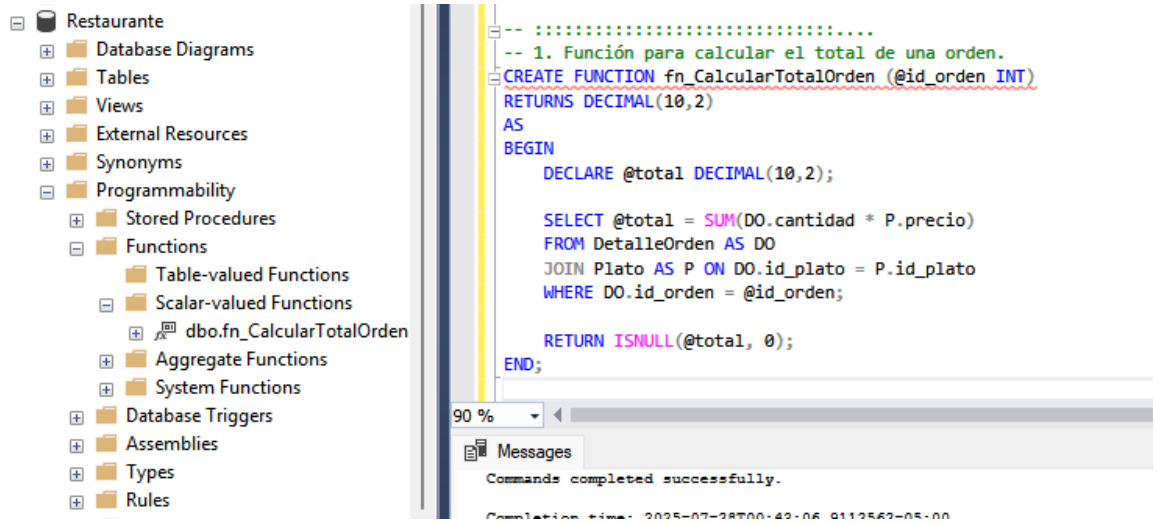

```
-- 5. SP para actualizar la cantidad de un ingrediente en el inventario.

CREATE PROCEDURE sp_ActualizarInventarioIngrediente

  @id_ingrediente INT,

  @cantidad_ajuste INT -- Puede ser positivo para añadir o negativo para restar

AS

BEGIN

  SET NOCOUNT ON;


  -- Validar si el ingrediente existe

  IF NOT EXISTS (SELECT 1 FROM Ingrediente WHERE id_ingrediente = @id_ingrediente)

  BEGIN

    PRINT 'Error: El ingrediente especificado no existe.';

    RETURN -1;

  END
```

```
    UPDATE Inventario
    SET cantidad = cantidad + @cantidad_ajuste,
        fecha = GETDATE() -- Actualizar la fecha de la última modificación
    WHERE id_ingrediente = @id_ingrediente;

    -- Opcional: Actualizar la disponibilidad si la cantidad baja de cierto umbral
    DECLARE @cantidad_actual INT;
    SELECT @cantidad_actual = cantidad FROM Inventario WHERE id_ingrediente =
@id_ingrediente;

    IF @cantidad_actual <= 0
    BEGIN
        UPDATE Inventario SET disponibilidad = 'Agotado' WHERE id_ingrediente =
@id_ingrediente;
    END
    ELSE IF @cantidad_actual < 10 -- Umbral de baja disponibilidad
    BEGIN
        UPDATE Inventario SET disponibilidad = 'Bajo Stock' WHERE id_ingrediente =
@id_ingrediente;
    END
    ELSE
    BEGIN
        UPDATE Inventario SET disponibilidad = 'Disponible' WHERE id_ingrediente =
@id_ingrediente;
    END

    PRINT 'Inventario del ingrediente actualizado exitosamente.';
    RETURN 0;
END;
```



- Funciones Definidas por el Usuario: Mínimo 3, con lógica SQL y cálculos personalizados.

```
-- 1. Función para calcular el total de una orden.
CREATE FUNCTION fn_CalcularTotalOrden (@id_orden INT)
RETURNS DECIMAL(10,2)
AS
BEGIN
   DECLARE @total DECIMAL(10,2);

   SELECT @total = SUM(DO.cantidad * P.precio)
   FROM DetalleOrden AS DO
   JOIN Plato AS P ON DO.id_plato = P.id_plato
   WHERE DO.id_orden = @id_orden;

   RETURN ISNULL(@total, 0);
END;
```



-- 2. Función para obtener el número de platos en una categoría.

CREATE FUNCTION fn_ContarPlatosPorCategoria (@id_categoria INT)

RETURNS INT

AS

BEGIN

DECLARE @conteo INT;

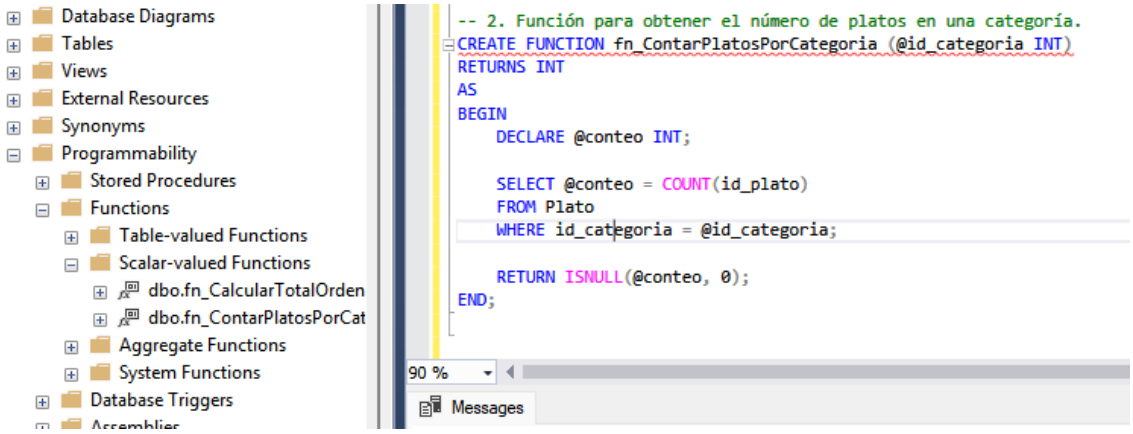
SELECT @conteo = COUNT(id_plato)

FROM Plato

WHERE id_categoria = @id_categoria;

RETURN ISNULL(@conteo, 0);

END;



-- 3. Función para determinar la duración de un turno en horas.

CREATE FUNCTION fn_CalcularDuracionTurnoHoras (@id_turno INT)

RETURNS DECIMAL(4,2)

AS

BEGIN

DECLARE @duracion_horas DECIMAL(4,2);

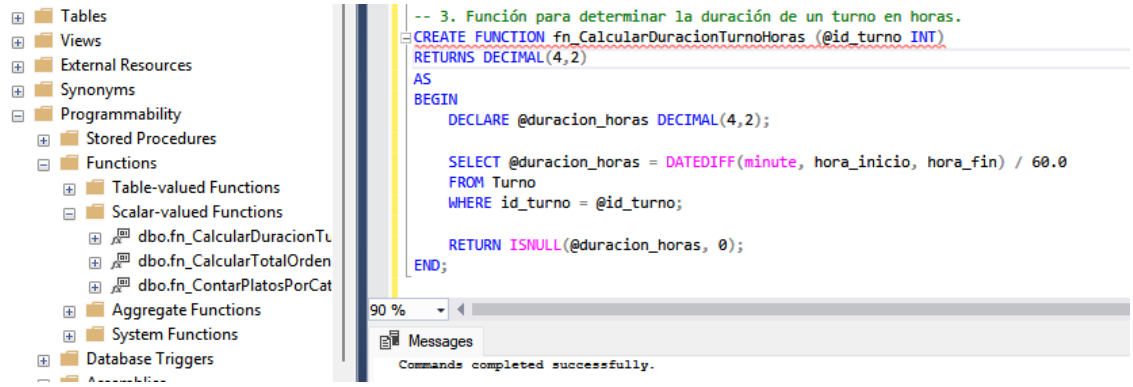
SELECT @duracion_horas = DATEDIFF(minute, hora_inicio, hora_fin) / 60.0

FROM Turno

WHERE id_turno = @id_turno;

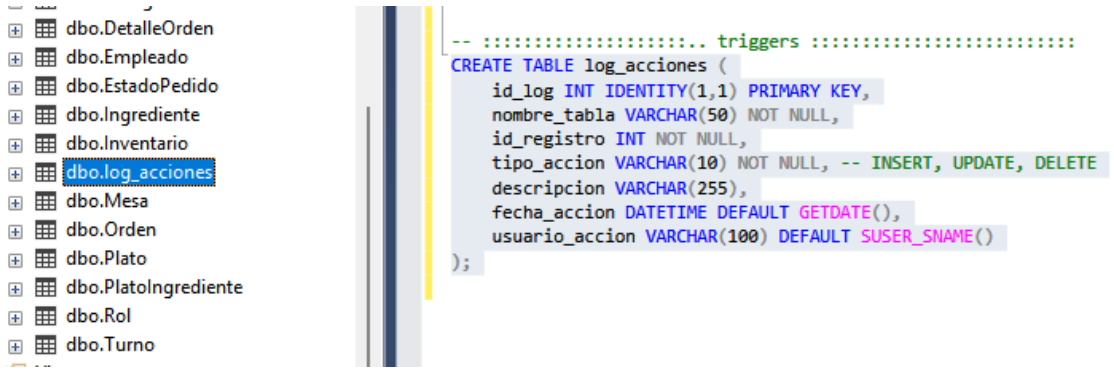
RETURN ISNULL(@duracion_horas, 0);

END;



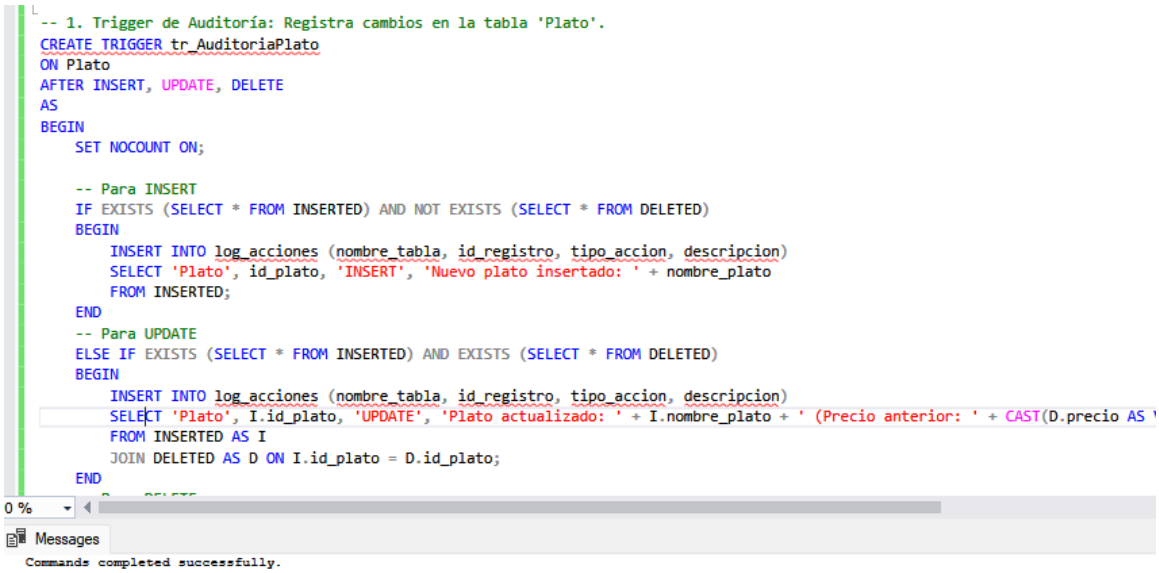
- Triggers: Mínimo 3, para auditoría, validaciones automáticas y simulación de notificaciones.

```
-- Tabla para auditoría
CREATE TABLE log_acciones (
    id_log INT IDENTITY(1,1) PRIMARY KEY,
    nombre_tabla VARCHAR(50) NOT NULL,
    id_registro INT NOT NULL,
    tipo_accion VARCHAR(10) NOT NULL, -- INSERT, UPDATE, DELETE
    descripcion VARCHAR(255),
    fecha_accion DATETIME DEFAULT GETDATE(),
    usuario_accion VARCHAR(100) DEFAULT SUSER_SNAME()
);
```




```
-- 1. Trigger de Auditoría: Registra cambios en la tabla 'Plato'.
CREATE TRIGGER tr_AuditoriaPlato
ON Plato
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
    SET NOCOUNT ON;

    -- Para INSERT
    IF EXISTS (SELECT * FROM INSERTED) AND NOT EXISTS (SELECT * FROM DELETED)
    BEGIN
```

```
    INSERT INTO log_acciones (nombre_tabla, id_registro, tipo_accion, descripcion)
    SELECT 'Plato', id_plato, 'INSERT', 'Nuevo plato insertado: ' + nombre_plato
    FROM INSERTED;
  END
  -- Para UPDATE
  ELSE IF EXISTS (SELECT * FROM INSERTED) AND EXISTS (SELECT * FROM DELETED)
  BEGIN
    INSERT INTO log_acciones (nombre_tabla, id_registro, tipo_accion, descripcion)
    SELECT 'Plato', I.id_plato, 'UPDATE', 'Plato actualizado: ' + I.nombre_plato + ' (Precio
anterior: ' + CAST(D.precio AS VARCHAR) + ', Precio nuevo: ' + CAST(I.precio AS VARCHAR) + ')'
    FROM INSERTED AS I
    JOIN DELETED AS D ON I.id_plato = D.id_plato;
  END
  -- Para DELETE
  ELSE IF EXISTS (SELECT * FROM DELETED) AND NOT EXISTS (SELECT * FROM INSERTED)
  BEGIN
    INSERT INTO log_acciones (nombre_tabla, id_registro, tipo_accion, descripcion)
    SELECT 'Plato', id_plato, 'DELETE', 'Plato eliminado: ' + nombre_plato
    FROM DELETED;
  END
END;
```



```
-- 1. Trigger de Auditoría: Registra cambios en la tabla 'Plato'.
CREATE TRIGGER tr_AuditoriaPlato
ON Plato
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
    SET NOCOUNT ON;

    -- Para INSERT
    IF EXISTS (SELECT * FROM INSERTED) AND NOT EXISTS (SELECT * FROM DELETED)
    BEGIN
        INSERT INTO log_acciones (nombre_tabla, id_registro, tipo_accion, descripcion)
        SELECT 'Plato', id_plato, 'INSERT', 'Nuevo plato insertado: ' + nombre_plato
        FROM INSERTED;
    END
    -- Para UPDATE
    ELSE IF EXISTS (SELECT * FROM INSERTED) AND EXISTS (SELECT * FROM DELETED)
    BEGIN
        INSERT INTO log_acciones (nombre_tabla, id_registro, tipo_accion, descripcion)
        SELECT 'Plato', I.id_plato, 'UPDATE', 'Plato actualizado: ' + I.nombre_plato + ' (Precio anterior: ' + CAST(D.precio AS
        FROM INSERTED AS I
        JOIN DELETED AS D ON I.id_plato = D.id_plato;
    END
```

0 %

Messages

Commands completed successfully.

```
-- 2. Trigger de Validación Automática: Evita la eliminación de un plato si está en alguna orden.
-- Este trigger se activa ANTES de que se intente eliminar un registro de la tabla Plato.
CREATE TRIGGER tr_ValidarEliminarPlato
ON Plato
INSTEAD OF DELETE
AS
BEGIN
    SET NOCOUNT ON;

    -- Declaramos una variable para almacenar el ID del plato que se intenta eliminar
    DECLARE @id_plato_a_eliminar INT;
    SELECT @id_plato_a_eliminar = id_plato FROM DELETED;

    -- Verificamos si el plato que se intenta eliminar existe en la tabla DetalleOrden
    IF EXISTS (SELECT 1 FROM DetalleOrden WHERE id_plato = @id_plato_a_eliminar)
    BEGIN
        -- Si el plato está en DetalleOrden, levantamos un error y no permitimos la eliminación
        RAISERROR('No se puede eliminar el plato con ID %d porque está asociado a una o más órdenes. Elimine primero los detalles de orden relacionados.', 16, 1, @id_plato_a_eliminar);
    END
    ELSE
    BEGIN
        -- Si el plato no está en ninguna DetalleOrden, permitimos la eliminación
        DELETE FROM Plato
        WHERE id_plato = @id_plato_a_eliminar;

        PRINT 'Plato con ID ' + CAST(@id_plato_a_eliminar AS VARCHAR) + ' eliminado exitosamente.';
    END
END;
```

GO

```sql
-- 2. Trigger de Validación Automática: Evita la eliminación de un plato si está en alguna orden.
-- Este trigger se activa ANTES de que se intente eliminar un registro de la tabla Plato.
CREATE TRIGGER tr_ValidarEliminarPlato
ON Plato
INSTEAD OF DELETE
AS
BEGIN
    SET NOCOUNT ON;

    -- Declaramos una variable para almacenar el ID del plato que se intenta eliminar
    DECLARE @id_plato_a_eliminar INT;
    SELECT @id_plato_a_eliminar = id_plato FROM DELETED;

    -- Verificamos si el plato que se intenta eliminar existe en la tabla DetalleOrden
    IF EXISTS (SELECT 1 FROM DetalleOrden WHERE id_plato = @id_plato_a_eliminar)
    BEGIN
        -- Si el plato está en DetalleOrden, levantamos un error y no permitimos la eliminación
        RAISERROR('No se puede eliminar el plato con ID %d porque está asociado a una o más órdenes. Elimine primero los detalles
    END
    ELSE
    BEGIN
        -- Si el plato no está en ninguna DetalleOrden, permitimos la eliminación
        DELETE FROM Plato
        WHERE id_plato = @id_plato_a_eliminar;
```

-- 3. Trigger de Simulación de Notificación: Cuando se actualiza el estado de un pedido a 'Completado', simula una notificación.

CREATE TRIGGER tr_NotificacionPedidoCompletado

ON Orden

AFTER UPDATE

AS

BEGIN

SET NOCOUNT ON;

DECLARE @old_estado INT;

DECLARE @new_estado INT;

DECLARE @id_orden INT;

DECLARE @numero_mesa INT;

SELECT @old_estado = D.id_estado, @new_estado = I.id_estado, @id_orden = I.id_orden

FROM INSERTED AS I

JOIN DELETED AS D ON I.id_orden = D.id_orden;

SELECT @numero_mesa = M.numero_mesa

FROM Orden AS O

JOIN Mesa AS M ON O.id_mesa = M.id_mesa

WHERE O.id_orden = @id_orden;

-- Asumiendo que el id_estado para 'Completado' es 2 (deberías verificarlo en tu tabla EstadoPedido)

IF @old_estado <> @new_estado AND @new_estado = (SELECT id_estado FROM EstadoPedido WHERE estado_pedido = 'Completado')

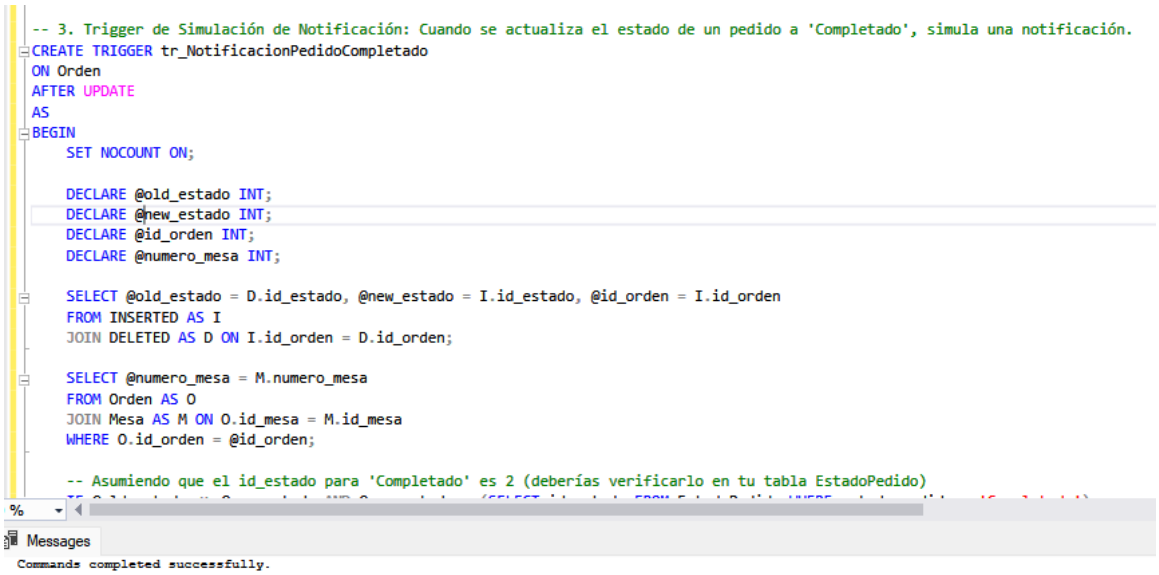
BEGIN

PRINT 'Notificación: ¡La Orden ' + CAST(@id_orden AS VARCHAR) + ' para la Mesa ' + CAST(@numero_mesa AS VARCHAR) + ' ha sido completada!';

-- Aquí podrías añadir lógica para enviar un mensaje a una cola de mensajes, un correo, etc.

END

END;



- Índices y Optimización: Índices simples y compuestos, análisis de rendimiento.

-- Índices simples:

-- Mejoran el rendimiento de las consultas que buscan por estas columnas.

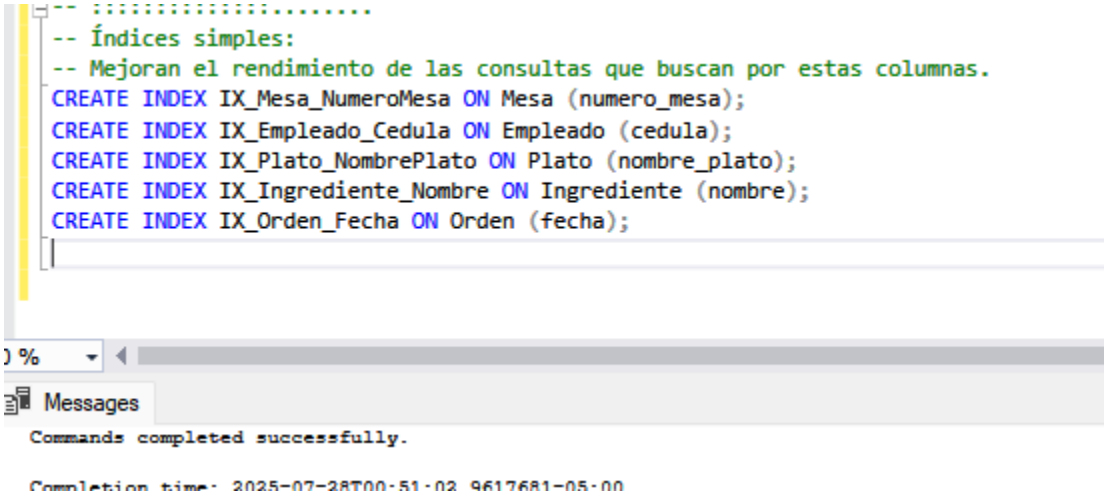
CREATE INDEX IX_Mesa_NumeroMesa ON Mesa (numero_mesa);

CREATE INDEX IX_Empleado_Cedula ON Empleado (cedula);

CREATE INDEX IX_Plato_NombrePlato ON Plato (nombre_plato);

CREATE INDEX IX_Ingrediente_Nombre ON Ingrediente (nombre);

CREATE INDEX IX_Orden_Fecha ON Orden (fecha);



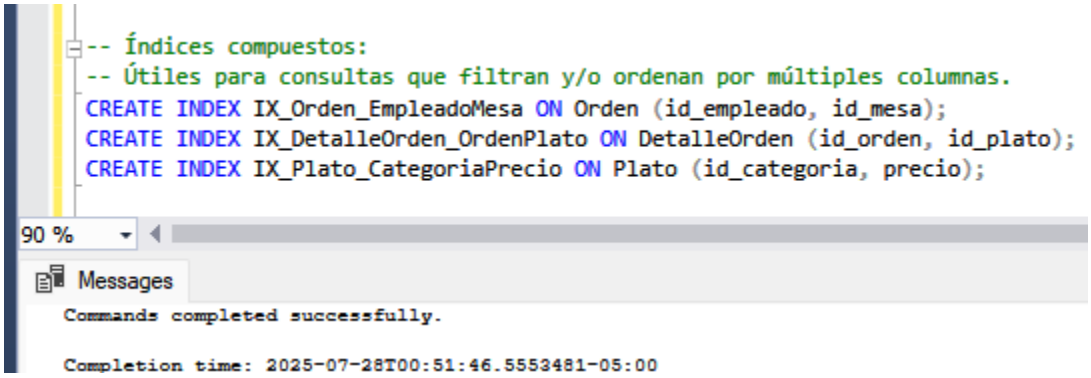
-- Índices compuestos:

-- Útiles para consultas que filtran y/o ordenan por múltiples columnas.

CREATE INDEX IX_Orden_EmpleadoMesa ON Orden (id_empleado, id_mesa);

CREATE INDEX IX_DetalleOrden_OrdenPlato ON DetalleOrden (id_orden, id_plato);

CREATE INDEX IX_Plato_CategoriaPrecio ON Plato (id_categoria, precio);



-- Análisis de rendimiento (Ejemplo de uso de EXPLAIN PLAN en SQL Server)

-- No se ejecuta, solo muestra cómo se analizaría:

-- SELECT * FROM sys.dm_exec_query_stats;

-- DBCC SHOW_STATISTICS ('Mesa', 'IX_Mesa_NumeroMesa');

-- Para ver el plan de ejecución de una consulta, puedes usar:

-- SET SHOWPLAN_ALL ON;

-- GO

-- SELECT O.id_orden, E.nombre_empleado FROM Orden AS O JOIN Empleado AS E ON O.id_empleado = E.id_empleado;

-- GO

-- SET SHOWPLAN_ALL OFF;

-- GO

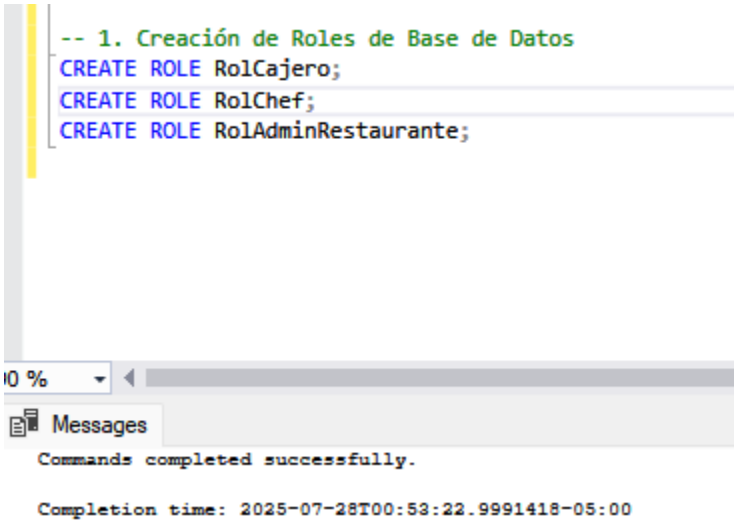
- Seguridad y Roles: Creación de roles, privilegios, encriptación, validaciones y simulación SSL.

-- 1. Creación de Roles de Base de Datos

CREATE ROLE RolCajero;

CREATE ROLE RolChef;

CREATE ROLE RolAdminRestaurante;



-- 2. Asignación de Privilegios a Roles

-- RolCajero: Puede ver mesas, órdenes, detalles de orden, empleados. Puede crear y actualizar órdenes.

GRANT SELECT ON Mesa TO RolCajero;

GRANT SELECT ON Orden TO RolCajero;

GRANT INSERT, UPDATE ON Orden TO RolCajero;

GRANT SELECT ON DetalleOrden TO RolCajero;

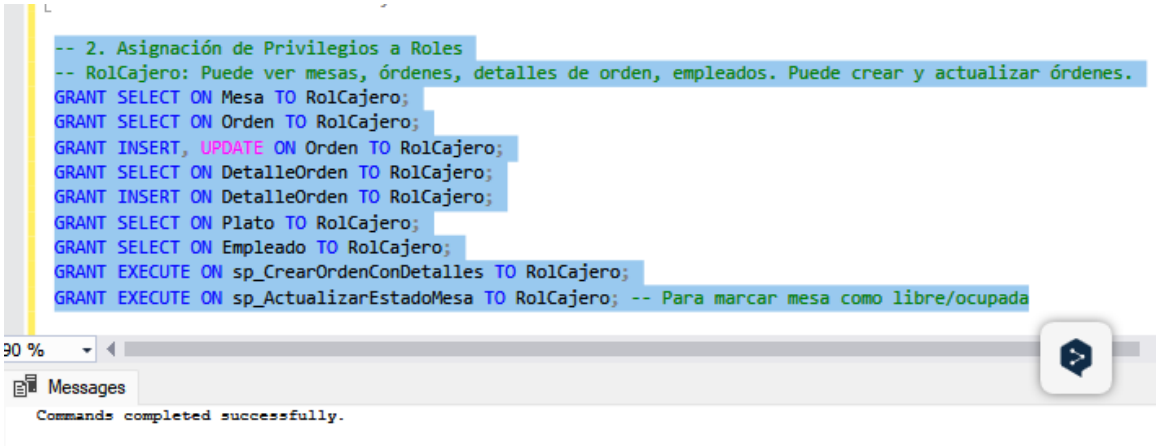
GRANT INSERT ON DetalleOrden TO RolCajero;

GRANT SELECT ON Plato TO RolCajero;

GRANT SELECT ON Empleado TO RolCajero;

GRANT EXECUTE ON sp_CrearOrdenConDetalles TO RolCajero;

GRANT EXECUTE ON sp_ActualizarEstadoMesa TO RolCajero; -- Para marcar mesa como libre/ocupada



-- RolChef: Puede ver platos, ingredientes, inventario, estados de pedido, y actualizar estados de pedido.

GRANT SELECT ON Plato TO RolChef;

GRANT SELECT ON Ingrediente TO RolChef;

GRANT SELECT ON Inventario TO RolChef;

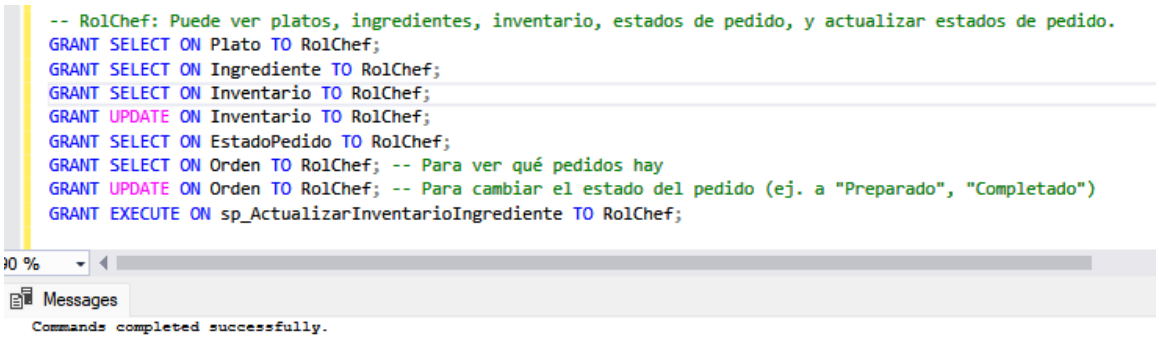
GRANT UPDATE ON Inventario TO RolChef;

GRANT SELECT ON EstadoPedido TO RolChef;

GRANT SELECT ON Orden TO RolChef; -- Para ver qué pedidos hay

GRANT UPDATE ON Orden TO RolChef; -- Para cambiar el estado del pedido (ej. a "Preparado", "Completado")

GRANT EXECUTE ON sp_ActualizarInventarioIngrediente TO RolChef;



-- RolAdminRestaurante: Control total (puede ser sysadmin o db_owner, pero mejor crear un rol específico con control granular)

-- Por simplicidad, aquí se le otorgan permisos más amplios sobre las tablas clave.

GRANT SELECT, INSERT, UPDATE, DELETE ON Mesa TO RolAdminRestaurante;

GRANT SELECT, INSERT, UPDATE, DELETE ON Turno TO RolAdminRestaurante;

GRANT SELECT, INSERT, UPDATE, DELETE ON EstadoPedido TO RolAdminRestaurante;

GRANT SELECT, INSERT, UPDATE, DELETE ON CategoriaPlato TO RolAdminRestaurante;

GRANT SELECT, INSERT, UPDATE, DELETE ON Ingrediente TO RolAdminRestaurante;

GRANT SELECT, INSERT, UPDATE, DELETE ON Rol TO RolAdminRestaurante;

GRANT SELECT, INSERT, UPDATE, DELETE ON Plato TO RolAdminRestaurante;

GRANT SELECT, INSERT, UPDATE, DELETE ON Orden TO RolAdminRestaurante;

GRANT SELECT, INSERT, UPDATE, DELETE ON DetalleOrden TO RolAdminRestaurante;

GRANT SELECT, INSERT, UPDATE, DELETE ON PlatoIngrediente TO RolAdminRestaurante;

GRANT SELECT, INSERT, UPDATE, DELETE ON Inventario TO RolAdminRestaurante;

GRANT SELECT, INSERT, UPDATE, DELETE ON Empleado TO RolAdminRestaurante;

GRANT EXECUTE TO RolAdminRestaurante; -- Permitir ejecutar todos los SPs y funciones.

```
-- RolAdminRestaurante: Control total (puede ser sysadmin o db_owner, pero mejor crear un rol específico con control granular)
-- Por simplicidad, aquí se le otorgan permisos más amplios sobre las tablas clave.
GRANT SELECT, INSERT, UPDATE, DELETE ON Mesa TO RolAdminRestaurante;
GRANT SELECT, INSERT, UPDATE, DELETE ON Turno TO RolAdminRestaurante;
GRANT SELECT, INSERT, UPDATE, DELETE ON EstadoPedido TO RolAdminRestaurante;
GRANT SELECT, INSERT, UPDATE, DELETE ON CategoriaPlato TO RolAdminRestaurante;
GRANT SELECT, INSERT, UPDATE, DELETE ON Ingrediente TO RolAdminRestaurante;
GRANT SELECT, INSERT, UPDATE, DELETE ON Rol TO RolAdminRestaurante;
GRANT SELECT, INSERT, UPDATE, DELETE ON Plato TO RolAdminRestaurante;
GRANT SELECT, INSERT, UPDATE, DELETE ON Orden TO RolAdminRestaurante;
GRANT SELECT, INSERT, UPDATE, DELETE ON DetalleOrden TO RolAdminRestaurante;
GRANT SELECT, INSERT, UPDATE, DELETE ON PlatoIngrediente TO RolAdminRestaurante;
GRANT SELECT, INSERT, UPDATE, DELETE ON Inventario TO RolAdminRestaurante;
GRANT SELECT, INSERT, UPDATE, DELETE ON Empleado TO RolAdminRestaurante;
GRANT EXECUTE TO RolAdminRestaurante; -- Permitir ejecutar todos los SPs y funciones.
```

```
%
Messages
Commands completed successfully.

Completion time: 2025-07-28T00:55:07.6803758-05:00
```

-- 3. Creación de Usuarios y Asignación a Roles (Ejemplos)

-- CREATE LOGIN usuario_cajero WITH PASSWORD = 'PasswordSeguro123!', CHECK_POLICY = ON;

-- CREATE USER usuario_cajero FOR LOGIN usuario_cajero;

-- ALTER ROLE RolCajero ADD MEMBER usuario_cajero;

```
-- 3. Creación de Usuarios y Asignación a Roles (Ejemplos)
CREATE LOGIN usuario_cajero WITH PASSWORD = '123', CHECK_POLICY = ON;
CREATE USER usuario_cajero FOR LOGIN usuario_cajero;
ALTER ROLE RolCajero ADD MEMBER usuario_cajero;

--  verificar
SELECT name, type_desc, is_disabled
FROM sys.server_principals
WHERE name = 'usuario_cajero';
```

%

Results | Messages

name	type_desc	is_disabled
usuario_cajero	SQL_LOGIN	0

```sql
-- verificar el usuario
SELECT name, type_desc
FROM sys.database_principals
WHERE name = 'usuario_cajero';
```

90 %

Results | Messages

	name	type_desc
1	usuario_cajero	SQL_USER

```sql
-- verificar si el usuario está en rol
SELECT r.name AS Rol, m.name AS Miembro
FROM sys.database_role_members rm
JOIN sys.database_principals r ON rm.role_principal_id = r.principal_id
JOIN sys.database_principals m ON rm.member_principal_id = m.principal_id
WHERE r.name = 'RolCajero' AND m.name = 'usuario_cajero';
```

90 %

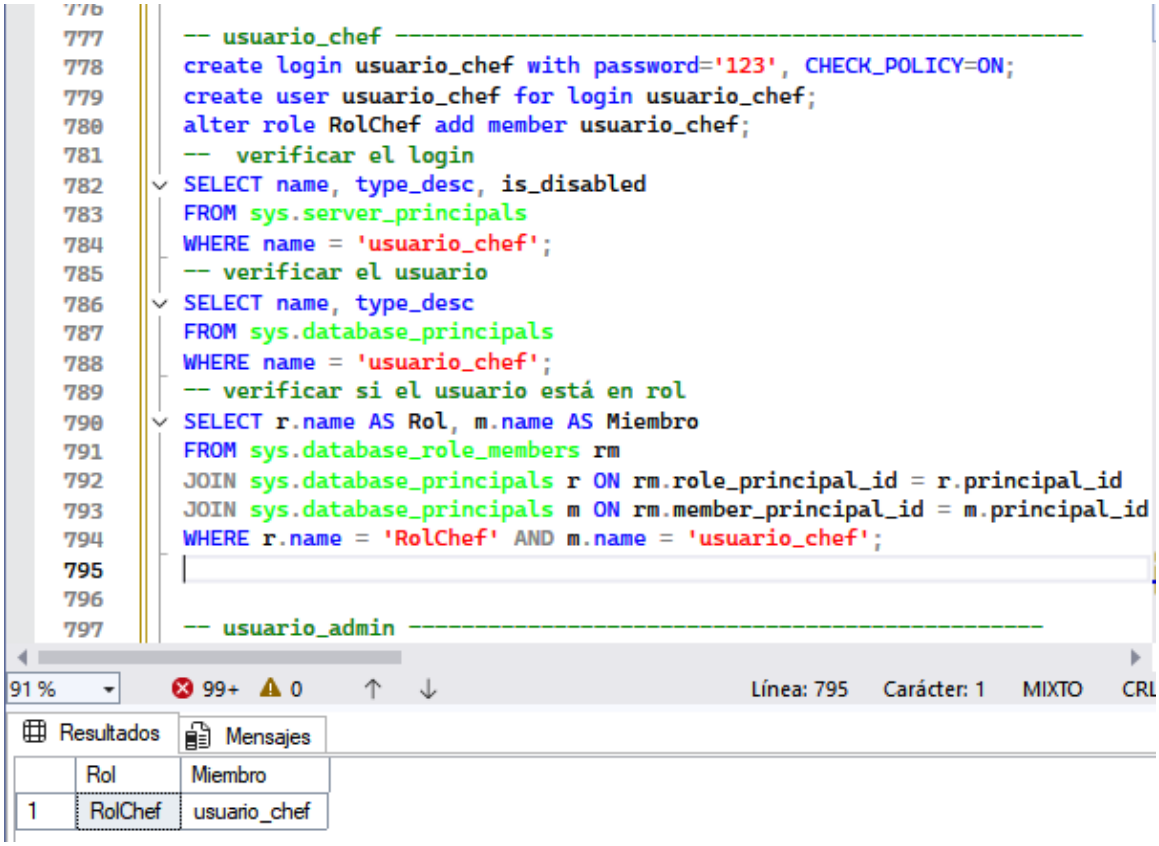
Results | Messages

	Rol	Miembro
1	RolCajero	usuario_cajero

-- CREATE LOGIN usuario_chef WITH PASSWORD = 'PasswordSeguro456!', CHECK_POLICY = ON;

-- CREATE USER usuario_chef FOR LOGIN usuario_chef;

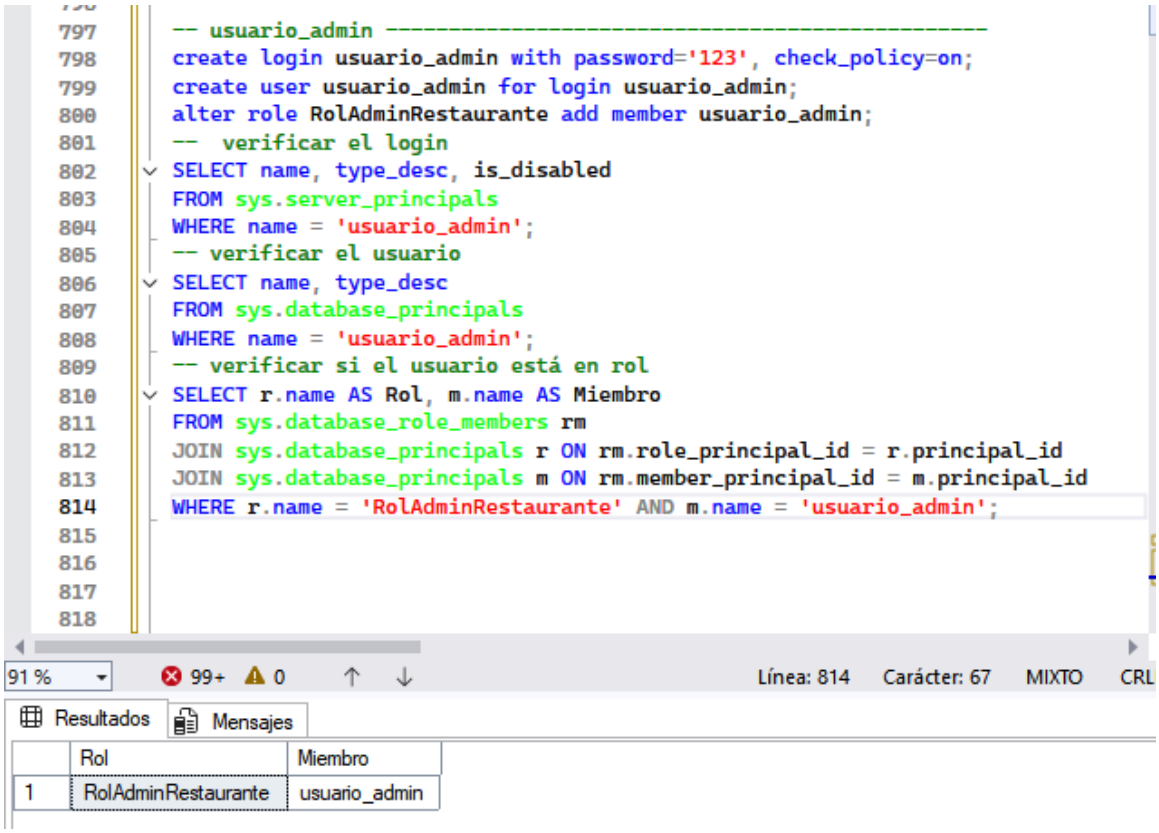
-- ALTER ROLE RolChef ADD MEMBER usuario_chef;



-- CREATE LOGIN usuario_admin WITH PASSWORD = 'PasswordSeguro789!', CHECK_POLICY = ON;

-- CREATE USER usuario_admin FOR LOGIN usuario_admin;

-- ALTER ROLE RolAdminRestaurante ADD MEMBER usuario_admin;



```
-- usuario_admin ----------------------------------------------------
create login usuario_admin with password='123', check_policy=on;
create user usuario_admin for login usuario_admin;
alter role RolAdminRestaurante add member usuario_admin;
--  verificar el login
SELECT name, type_desc, is_disabled
FROM sys.server_principals
WHERE name = 'usuario_admin';
-- verificar el usuario
SELECT name, type_desc
FROM sys.database_principals
WHERE name = 'usuario_admin';
-- verificar si el usuario está en rol
SELECT r.name AS Rol, m.name AS Miembro
FROM sys.database_role_members rm
JOIN sys.database_principals r ON rm.role_principal_id = r.principal_id
JOIN sys.database_principals m ON rm.member_principal_id = m.principal_id
WHERE r.name = 'RolAdminRestaurante' AND m.name = 'usuario_admin';
```

91 % 99+ 0 Línea: 814 Carácter: 67 MIXTO CRL

Resultados Mensajes

	Rol	Miembro
1	RolAdminRestaurante	usuario_admin

-- 4. Encriptación (Ejemplo de Always Encrypted para datos sensibles como la cédula del empleado)

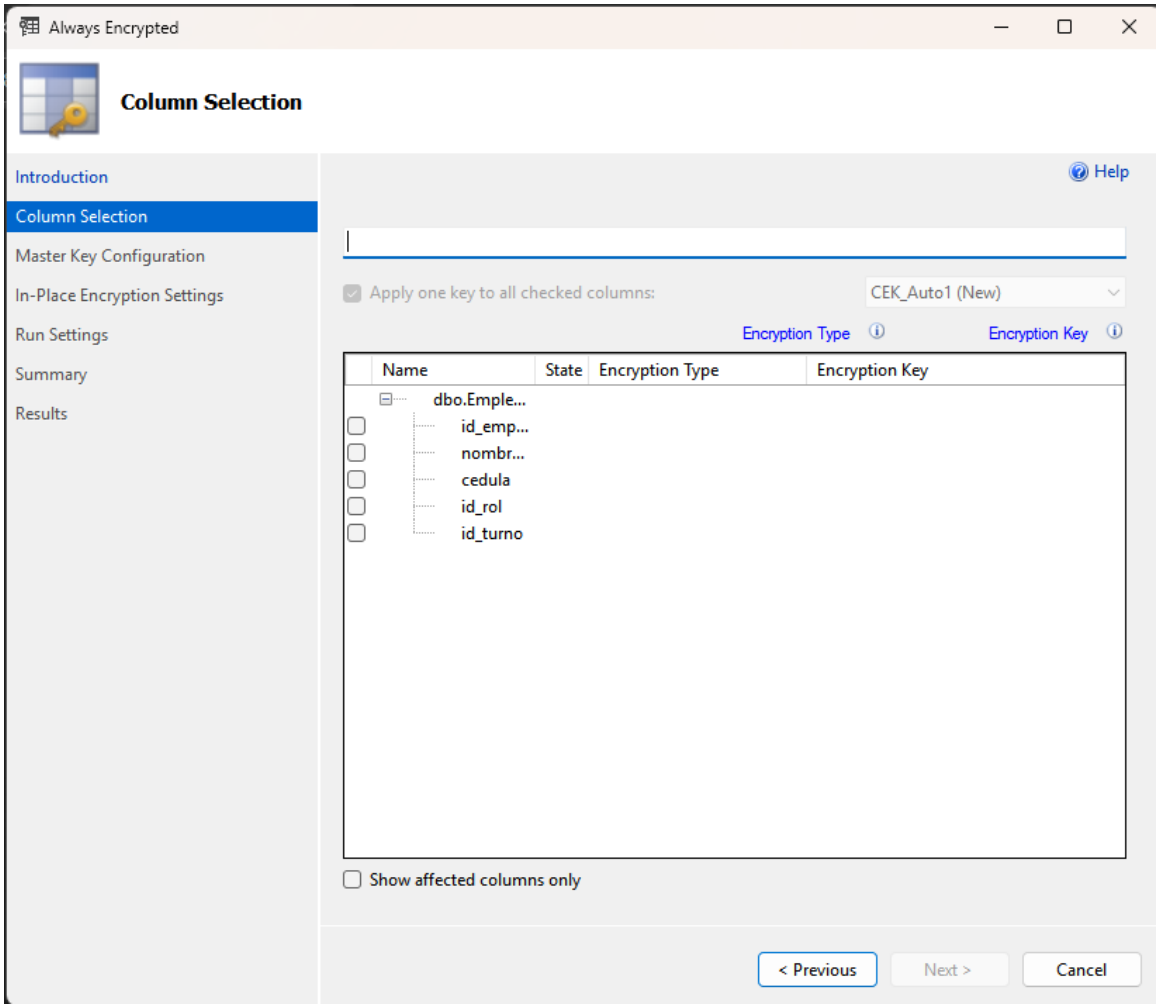
-- NOTA: Always Encrypted requiere configuración adicional fuera de este script (Keys, Column Encryption Wizard).

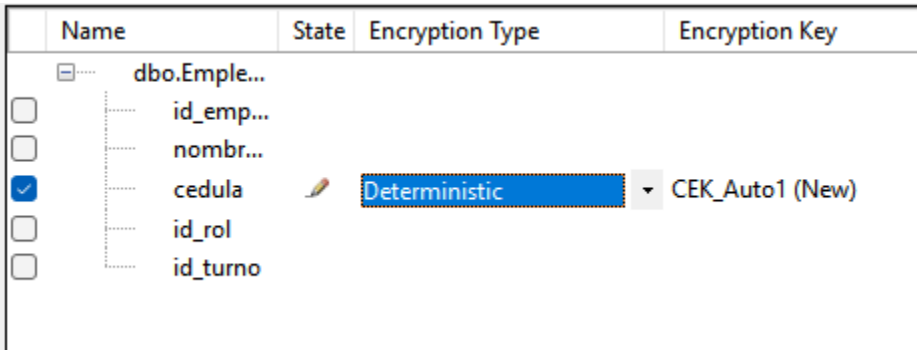
-- Esto es solo una simulación del DDL para una columna encriptada.

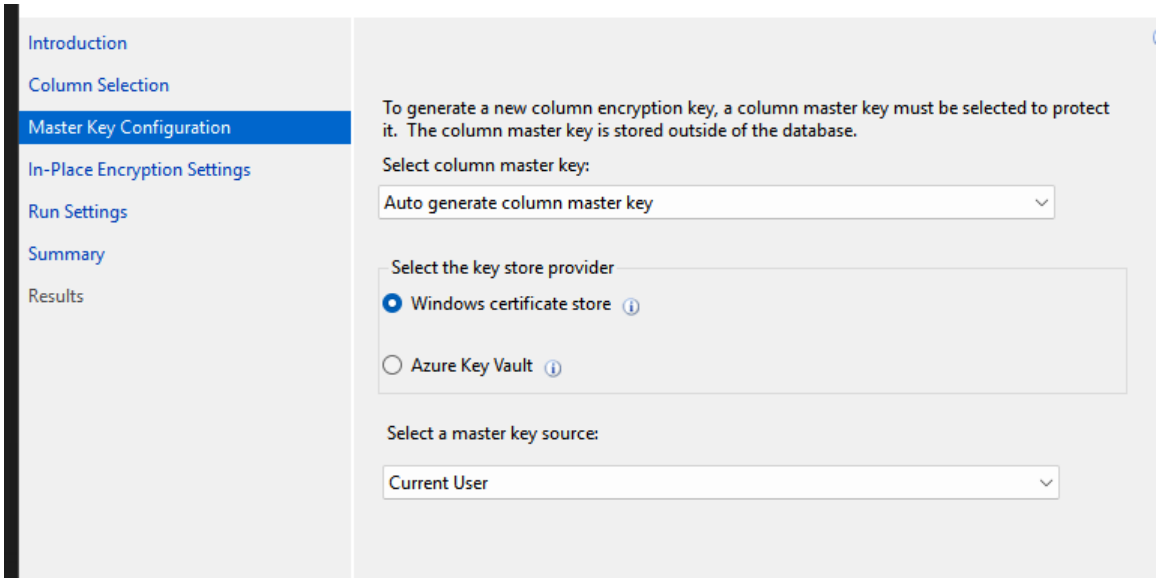
-- ALTER TABLE Empleado

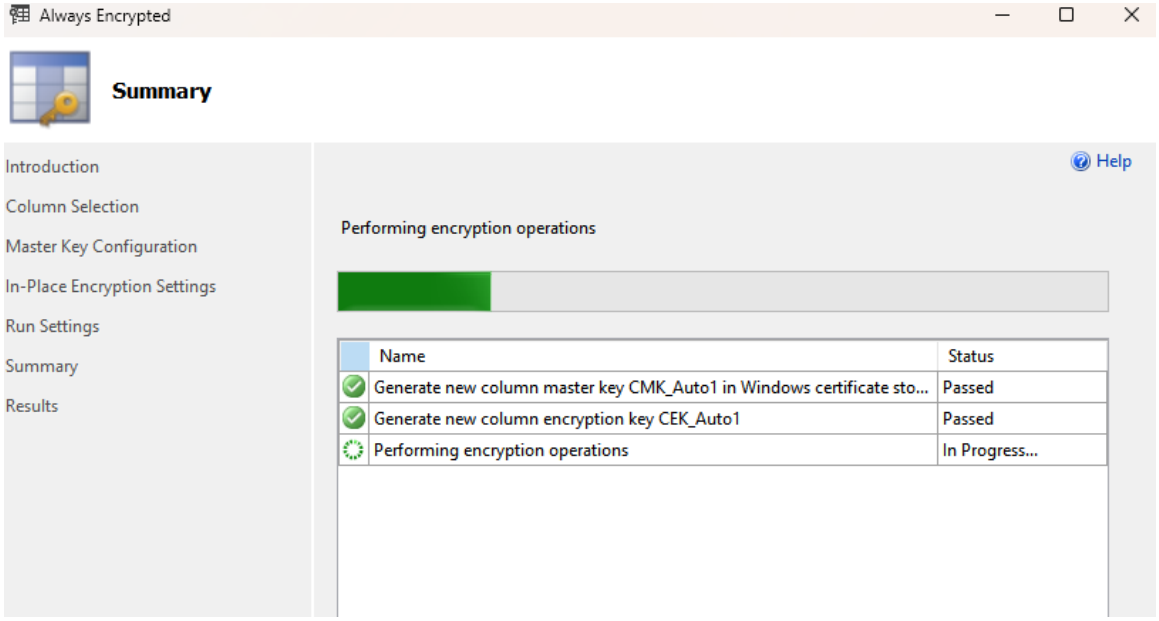
-- ALTER COLUMN cedula VARBINARY(256) ENCRYPTED WITH (COLUMN_ENCRYPTION_KEY = [CEK_Auto1], ENCRYPTION_TYPE = DETERMINISTIC, ALGORITHM = 'AEAD_AES_256_CBC_HMAC_SHA256');

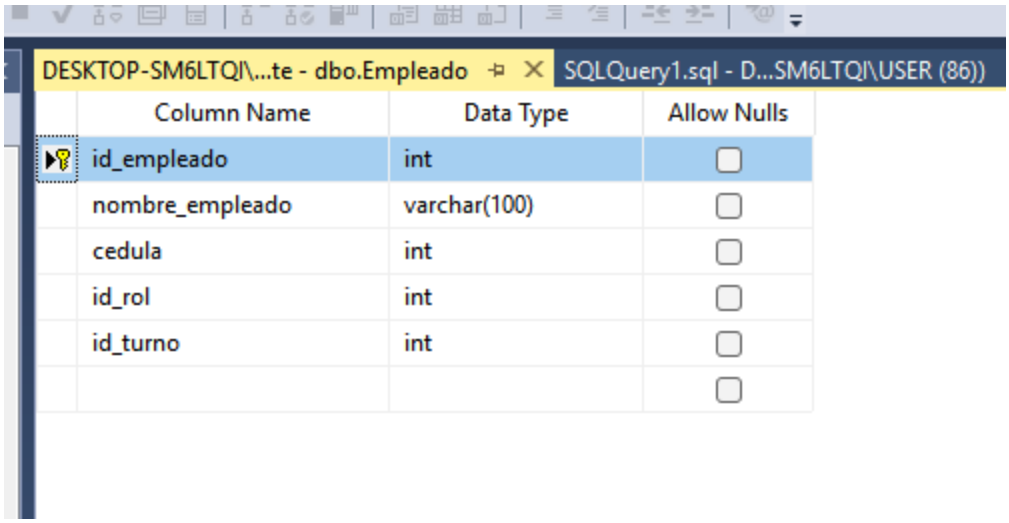
-- (Necesitarías una Master Key y Column Encryption Key creadas previamente)

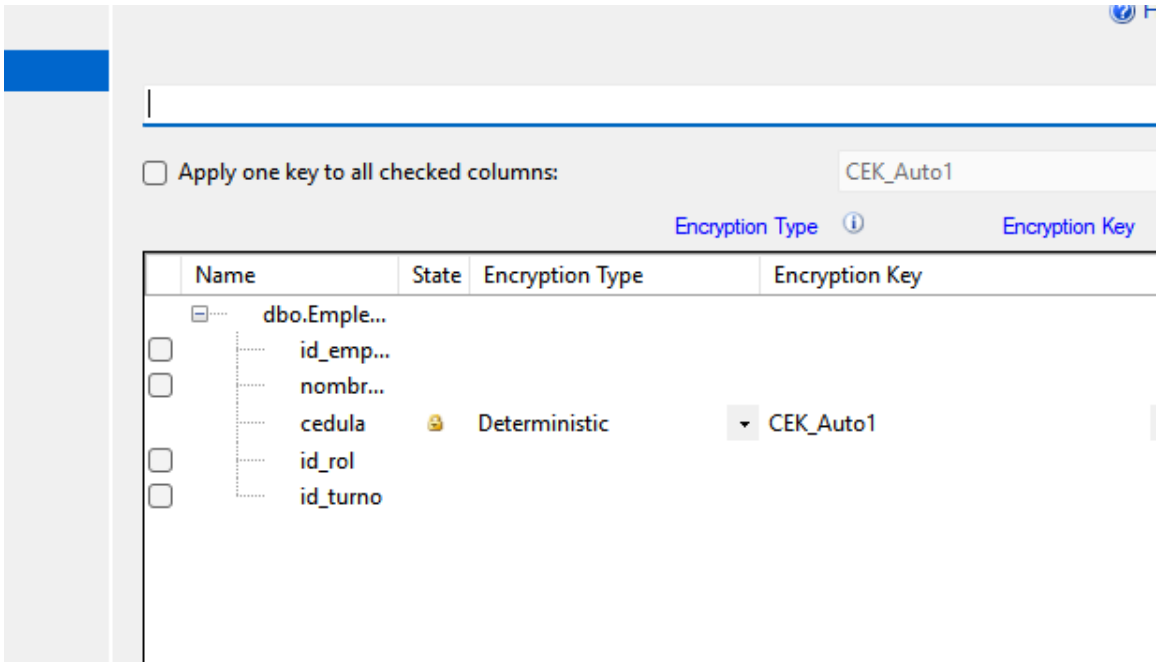

Always Encrypted
Column Selection
Introduction
Column Selection
Master Key Configuration
In-Place Encryption Settings
Run Settings
Summary
Results
Help
Apply one key to all checked columns:
CEK_Auto1 (New)
Encryption Type
Encryption Key
Name
State
Encryption Type
Encryption Key
dbo.Emple...
id_emp...
nombr...
cedula
id_rol
id_turno
Show affected columns only
< Previous
Next >
Cancel


Name
State
Encryption Type
Encryption Key
dbo.Emple...
id_emp...
nombr...
cedula
Deterministic
CEK_Auto1 (New)
id_rol
id_turno


Introduction
Column Selection
Master Key Configuration
In-Place Encryption Settings
Run Settings
Summary
Results
To generate a new column encryption key, a column master key must be selected to protect it. The column master key is stored outside of the database.
Select column master key:
Auto generate column master key
Select the key store provider
Windows certificate store
Azure Key Vault
Select a master key source:
Current User


Always Encrypted
Summary
Introduction
Column Selection
Master Key Configuration
In-Place Encryption Settings
Run Settings
Summary
Results
Help
Performing encryption operations
Name
Status
Generate new column master key CMK_Auto1 in Windows certificate sto...
Passed
Generate new column encryption key CEK_Auto1
Passed
Performing encryption operations
In Progress...





-- 5. Validaciones (Manejadas por SPs y Triggers, como se vio anteriormente)

-- Ej. validación de plato en tr _ValidarEliminarPlato

Resultado fallido:

```
842  EXEC sp_InsertarPlato
843      @nombre_plato = 'Plato de Prueba',
844      @precio = 10.00,
845      @id_categoria = 1;
```

99+ 0

Mensajes

```
Plato insertado exitosamente.

Hora de finalización: 2025-08-02T16:35:29.9530589-05:00
```

```
847  -- buscar el plato
848  select id_plato from Plato where nombre_plato= 'Plato de Prueba';
849
850  --eliminar
851  DELETE FROM Plato WHERE id_plato = 508;
```

90 % 99+ 0

Resultados Mensajes

	id_plato
1	509

```
850  --eliminar
851  DELETE FROM Plato WHERE id_plato = 509;
```

90 % 99+ 0

Mensajes

```
Plato con ID 509 eliminado exitosamente.

(1 fila afectada)

Hora de finalización: 2025-08-02T16:37:02.8591189-05:00
```

Resultado exitoso:

```
855  DELETE FROM Plato WHERE id_plato = 434;
856  -- verificar si el plato todavía existe
```

99+ 0 Línea: 860

Mensajes

```
Mensaje 50000, nivel 16, estado 1, procedimiento tr_ValidarEliminarPlato, línea 18 [línea de inicio de lote 854]
No se puede eliminar el plato con ID 434 porque está asociado a una o más órdenes. Elimine primero los detalles de orden relacionados.

(1 fila afectada)

Hora de finalización: 2025-08-02T16:46:16.8460788-05:00
```

```
856  -- verificar si el plato todavía existe
857  SELECT id_plato, nombre_plato, precio FROM Plato WHERE id_plato = 434;
858  GO
859
```

90 % 99+ 0

Resultados | Mensajes

	id_plato	nombre_plato	precio
1	434	Arroz con Pollo 434	43.03

-- Ej. validación de existencia de categoría en sp_InsertarPlato.

Resultado exitoso:

```
831  --  Probar la Validación del SP sp_InsertarPlato
832  SELECT TOP 1 id_categoria, tipo_categoria FROM CategoriaPlato;
833
834  EXEC sp_InsertarPlato
835      @nombre_plato = 'Limonada Rosa con hierba buena',
836      @precio = 15.75,
837      @id_categoria = 2;
838
839
840
```

91 % 99+ 0 Línea: 836 Carácter: 21 MIXTO

Mensajes

```
Plato insertado exitosamente.

Hora de finalización: 2025-08-02T13:19:38.9097741-05:00
```

Resultado fallido:

```
840  -- resultado fallido
841  EXEC sp_InsertarPlato
842      @nombre_plato = 'Postre Exótico',
843      @precio = 22.50,
844      @id_categoria = 9999; -- id inexistente
```

99+ 0 Línea: 843

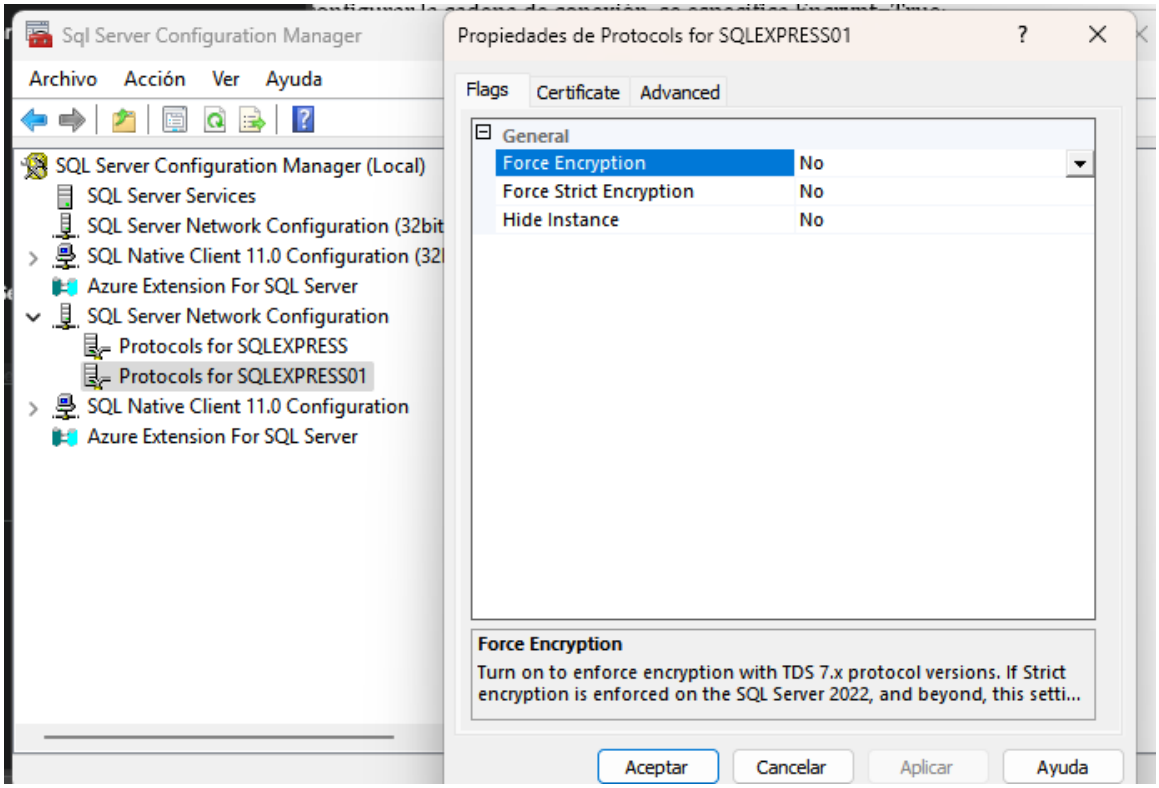
Mensajes

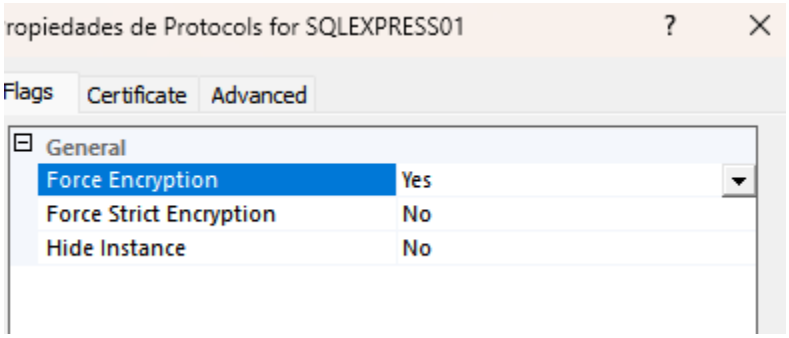
```
Error: La categoría especificada no existe.

Hora de finalización: 2025-08-02T13:22:52.5478392-05:00
```

-- 6. Simulación SSL (a nivel de conexión con SQL Server)

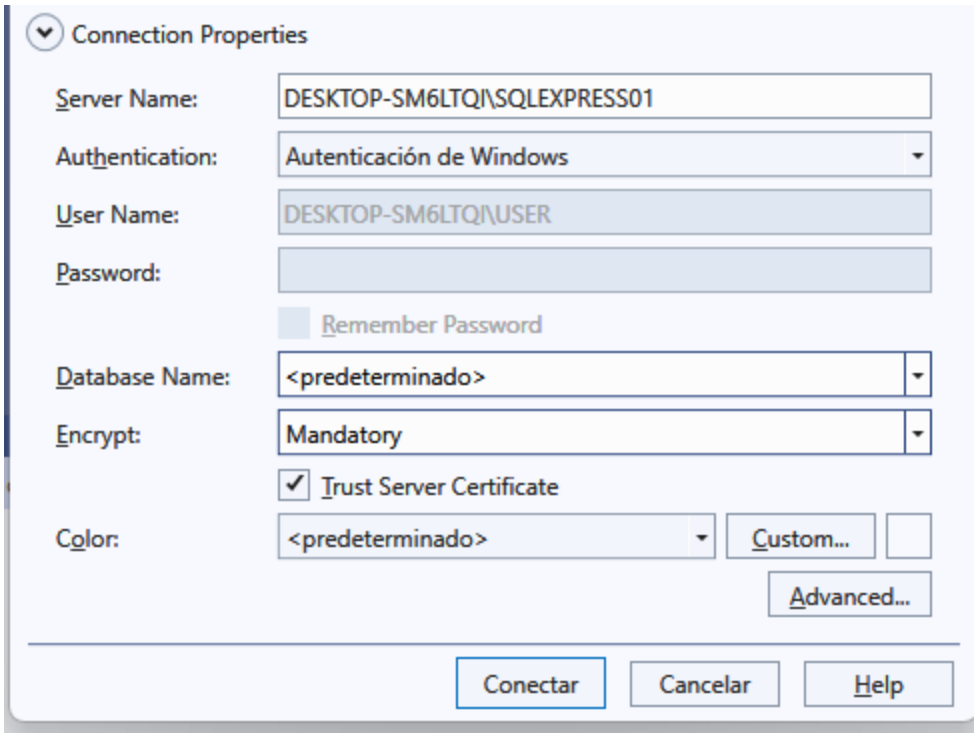
-- Esto no es un script de SQL, sino una configuración del servidor y los clientes.

-- Para forzar SSL en SQL Server, debes ir a SQL Server Configuration Manager -> SQL Server Network Configuration -> Protocols for MSSQLSERVER -> Force Encryption -> Yes.





-- En la aplicación cliente, al configurar la cadena de conexión, se especifica Encrypt=True; TrustServerCertificate=False; (si tienes un certificado válido) o TrustServerCertificate=True; (para certificados autofirmados, no recomendado en producción).



- Auditoría: Tabla log_acciones, triggers conectados, reportes y bitácora centralizada.

-- 1. Tabla log_acciones (Ya creada arriba junto con los triggers)

-- CREATE TABLE log_acciones (...);

```
879  -- 1. Tabla log_acciones
880  SELECT
881      id_log,
882      nombre_tabla,
883      id_registro,
884      tipo_accion,
885      descripcion,
886      fecha_accion,
887      usuario_accion
888  FROM
889      log_acciones
890  ORDER BY
891      fecha_accion DESC; -- ordena desde la fecha mas reciente
892  GO
```

90 % 99+ 0

Resultados Mensajes

	id_log	nombre_tabla	id_registro	tipo_accion	descripcion	fecha_accion	usuario_accion
2	16	Plato	509	INSERT	Nuevo plato insertado: Plato d...	2025-08-02 16:35:29.940	DESKTOP-SM6LTQI\USER
3	15	Plato	508	DELETE	Plato eliminado: Plato de Prue...	2025-08-02 16:31:14.583	DESKTOP-SM6LTQI\USER
4	14	Plato	507	DELETE	Plato eliminado: Plato de Prue...	2025-08-02 16:31:08.910	DESKTOP-SM6LTQI\USER
5	13	Plato	506	DELETE	Plato eliminado: Plato de Prue...	2025-08-02 16:30:59.947	DESKTOP-SM6LTQI\USER
6	12	Plato	505	DELETE	Plato eliminado: Plato de Prue...	2025-08-02 16:30:53.530	DESKTOP-SM6LTQI\USER
7	11	Plato	504	DELETE	Plato eliminado: Plato de Prue...	2025-08-02 16:30:47.183	DESKTOP-SM6LTQI\USER
8	10	Plato	503	DELETE	Plato eliminado: Plato de Prueba	2025-08-02 16:30:40.047	DESKTOP-SM6LTQI\USER
9	9	Plato	502	DELETE	Plato eliminado: Plato de Prueba	2025-08-02 16:30:06.400	DESKTOP-SM6LTQI\USER
10	8	Plato	508	INSERT	Nuevo plato insertado: Plato d...	2025-08-02 16:28:09.377	DESKTOP-SM6LTQI\USER
11	7	Plato	507	INSERT	Nuevo plato insertado: Plato d...	2025-08-02 16:28:03.893	DESKTOP-SM6LTQI\USER

-- 2. Triggers conectados (Ya creados: tr_AuditoriaPlato y cualquier otro trigger que inserte en log_acciones)

```
-- 2. Triggers conectados
SELECT
    id_log,
    id_registro,
    tipo_accion,
    descripcion,
    fecha_accion,
    usuario_accion
FROM
    log_acciones
WHERE
    nombre_tabla = 'Plato' -- Filtrar la tabla por el nombre
ORDER BY
    fecha_accion DESC;
GO
```

99+ 0

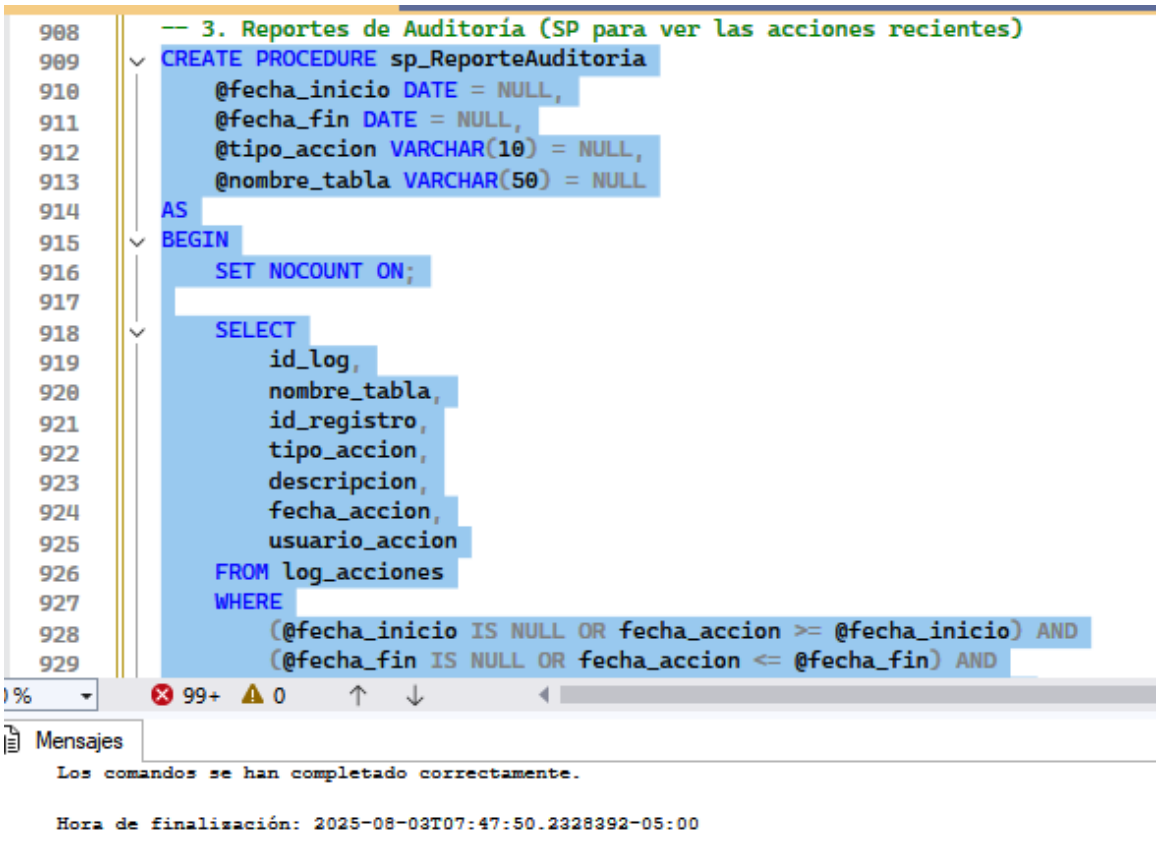
Resultados | Mensajes

id_log	id_registro	tipo_accion	descripcion	fecha_accion	usuario_accion
17	509	DELETE	Plato eliminado: Plato de Prueba	2025-08-02 16:37:02.847	DESKTOP-SM6LTQI\USER
16	509	INSERT	Nuevo plato insertado: Plato de Prueba	2025-08-02 16:35:29.940	DESKTOP-SM6LTQI\USER
15	508	DELETE	Plato eliminado: Plato de Prueba02	2025-08-02 16:31:14.583	DESKTOP-SM6LTQI\USER
14	507	DELETE	Plato eliminado: Plato de Prueba02	2025-08-02 16:31:08.910	DESKTOP-SM6LTQI\USER
13	506	DELETE	Plato eliminado: Plato de Prueba2	2025-08-02 16:30:59.947	DESKTOP-SM6LTQI\USER
12	505	DELETE	Plato eliminado: Plato de Prueba2	2025-08-02 16:30:53.530	DESKTOP-SM6LTQI\USER
11	504	DELETE	Plato eliminado: Plato de Prueba para Eliminar	2025-08-02 16:30:47.183	DESKTOP-SM6LTQI\USER
10	503	DELETE	Plato eliminado: Plato de Prueba	2025-08-02 16:30:40.047	DESKTOP-SM6LTQI\USER

-- 3. Reportes de Auditoría (Ejemplo: SP para ver las acciones recientes)

```
CREATE PROCEDURE sp_ReporteAuditoria
    @fecha_inicio DATE = NULL,
    @fecha_fin DATE = NULL,
    @tipo_accion VARCHAR(10) = NULL,
    @nombre_tabla VARCHAR(50) = NULL
AS
BEGIN
    SET NOCOUNT ON;

    SELECT
        id_log,
        nombre_tabla,
        id_registro,
```

```
    tipo_accion,
    descripcion,
    fecha_accion,
    usuario_accion
  FROM log_acciones
  WHERE
    (@fecha_inicio IS NULL OR fecha_accion >= @fecha_inicio) AND
    (@fecha_fin IS NULL OR fecha_accion <= @fecha_fin) AND
    (@tipo_accion IS NULL OR tipo_accion = @tipo_accion) AND
    (@nombre_tabla IS NULL OR nombre_tabla = @nombre_tabla)
  ORDER BY fecha_accion DESC;
END;
```



```
935  -- ver todos los registros de auditoria (sin filtros)
936  EXEC sp_ReporteAuditoria;
```

Resultados | Mensajes

	id_log	nombre_tabla	id_registro	tipo_accion	descripcion	fecha_accion	usuario_accion
1	18	Plato	510	INSERT	Nuevo plato insertado: Limonada Rosa con hierba ...	2025-08-03 07:49:19.827	DESKTOP-SM6LTQI\USER
2	17	Plato	509	DELETE	Plato eliminado: Plato de Prueba	2025-08-02 16:37:02.847	DESKTOP-SM6LTQI\USER
3	16	Plato	509	INSERT	Nuevo plato insertado: Plato de Prueba	2025-08-02 16:35:29.940	DESKTOP-SM6LTQI\USER
4	15	Plato	508	DELETE	Plato eliminado: Plato de Prueba02	2025-08-02 16:31:14.583	DESKTOP-SM6LTQI\USER
5	14	Plato	507	DELETE	Plato eliminado: Plato de Prueba02	2025-08-02 16:31:08.910	DESKTOP-SM6LTQI\USER
6	13	Plato	506	DELETE	Plato eliminado: Plato de Prueba2	2025-08-02 16:30:59.947	DESKTOP-SM6LTQI\USER
7	12	Plato	505	DELETE	Plato eliminado: Plato de Prueba2	2025-08-02 16:30:53.530	DESKTOP-SM6LTQI\USER
8	11	Plato	504	DELETE	Plato eliminado: Plato de Prueba para Eliminar	2025-08-02 16:30:47.183	DESKTOP-SM6LTQI\USER
9	10	Plato	503	DELETE	Plato eliminado: Plato de Prueba	2025-08-02 16:30:40.047	DESKTOP-SM6LTQI\USER
10	9	Plato	502	DELETE	Plato eliminado: Plato de Prueba	2025-08-02 16:30:06.400	DESKTOP-SM6LTQI\USER
11	8	Plato	508	INSERT	Nuevo plato insertado: Plato de Prueba02	2025-08-02 16:28:09.377	DESKTOP-SM6LTQI\USER
12	7	Plato	507	INSERT	Nuevo plato insertado: Plato de Prueba02	2025-08-02 16:28:03.893	DESKTOP-SM6LTQI\USER

```
937  -- ver INSERT
938  EXEC sp_ReporteAuditoria @tipo_accion = 'INSERT';
939
```

Resultados | Mensajes

	id_log	nombre_tabla	id_registro	tipo_accion	descripcion	fecha_accion	usuario_accion
1	18	Plato	510	INSERT	Nuevo plato insertado: Limonada Rosa con hierba ...	2025-08-03 07:49:19.827	DESKTOP-SM6LTQI\USER
2	16	Plato	509	INSERT	Nuevo plato insertado: Plato de Prueba	2025-08-02 16:35:29.940	DESKTOP-SM6LTQI\USER
3	8	Plato	508	INSERT	Nuevo plato insertado: Plato de Prueba02	2025-08-02 16:28:09.377	DESKTOP-SM6LTQI\USER
4	7	Plato	507	INSERT	Nuevo plato insertado: Plato de Prueba02	2025-08-02 16:28:03.893	DESKTOP-SM6LTQI\USER
5	6	Plato	506	INSERT	Nuevo plato insertado: Plato de Prueba2	2025-08-02 16:27:40.257	DESKTOP-SM6LTQI\USER
6	5	Plato	505	INSERT	Nuevo plato insertado: Plato de Prueba2	2025-08-02 16:26:31.420	DESKTOP-SM6LTQI\USER
7	4	Plato	504	INSERT	Nuevo plato insertado: Plato de Prueba para Eliminar	2025-08-02 16:24:09.640	DESKTOP-SM6LTQI\USER

```
939  -- ver todas las acciones en la tabla 'plato'
940  EXEC sp_ReporteAuditoria @nombre_tabla = 'Plato';
```

Resultados | Mensajes

	id_log	nombre_tabla	id_registro	tipo_accion	descripcion	fecha_accion	usuario_accion
1	18	Plato	510	INSERT	Nuevo plato insertado: Limonada Rosa con hierba ...	2025-08-03 07:49:19.827	DESKTOP-SM6LTQI\USER
2	17	Plato	509	DELETE	Plato eliminado: Plato de Prueba	2025-08-02 16:37:02.847	DESKTOP-SM6LTQI\USER
3	16	Plato	509	INSERT	Nuevo plato insertado: Plato de Prueba	2025-08-02 16:35:29.940	DESKTOP-SM6LTQI\USER
4	15	Plato	508	DELETE	Plato eliminado: Plato de Prueba02	2025-08-02 16:31:14.583	DESKTOP-SM6LTQI\USER
5	14	Plato	507	DELETE	Plato eliminado: Plato de Prueba02	2025-08-02 16:31:08.910	DESKTOP-SM6LTQI\USER
6	13	Plato	506	DELETE	Plato eliminado: Plato de Prueba2	2025-08-02 16:30:59.947	DESKTOP-SM6LTQI\USER
7	12	Plato	505	DELETE	Plato eliminado: Plato de Prueba2	2025-08-02 16:30:53.530	DESKTOP-SM6LTQI\USER
8	11	Plato	504	DELETE	Plato eliminado: Plato de Prueba para Eliminar	2025-08-02 16:30:47.183	DESKTOP-SM6LTQI\USER

-- 4. Bitácora Centralizada (El log_acciones actúa como una bitácora centralizada para la aplicación)

-- Todas las acciones importantes (INSERT/UPDATE/DELETE en tablas críticas, intentos de acceso fallidos si se implementan triggers a nivel de servidor, etc.)

-- pueden ser registradas en esta tabla para un seguimiento unificado.

```sql
-- 4. Bitácora Centralizada (El log_acciones actúa como una bitácora centralizada para la
-- Todas las acciones importantes (INSERT/UPDATE/DELETE en tablas críticas, intentos de a
-- pueden ser registradas en esta tabla para un seguimiento unificado.
-- Consulta para ver todos los registros en tu bitácora centralizada
SELECT
    id_log,
    nombre_tabla,
    id_registro,
    tipo_accion,
    descripcion,
    fecha_accion,
    usuario_accion
FROM
    log_acciones
ORDER BY
    fecha_accion DESC;
GO
```

99+ 0 Línea: 958 Carácter: 24 MIXTO CRLF

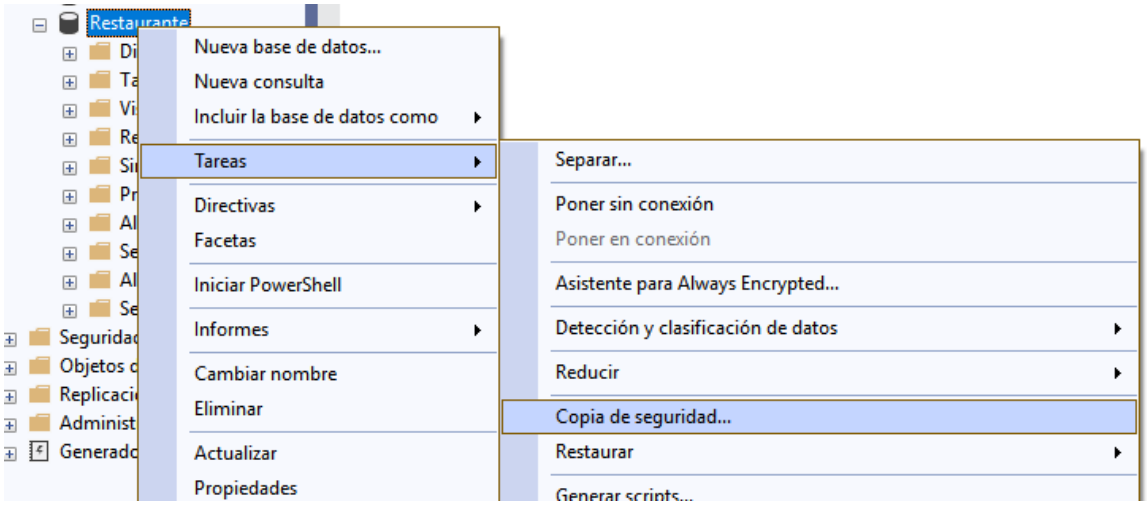
Resultados Mensajes

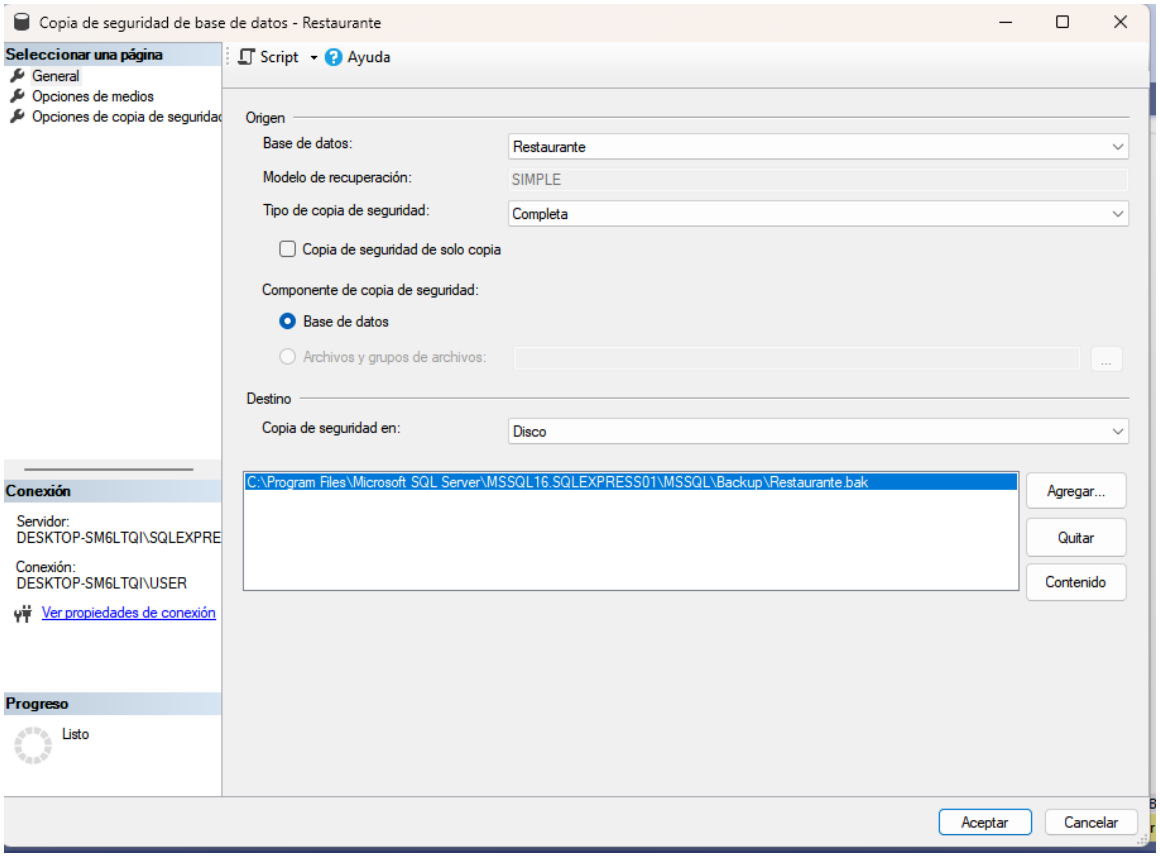
id_log	nombre_tabla	id_registro	tipo_accion	descripcion	fecha_accion
18	Plato	510	INSERT	Nuevo plato insertado: Limonada Rosa con hierba ...	2025-08-03 0
17	Plato	509	DELETE	Plato eliminado: Plato de Prueba	2025-08-02 1

- Respaldo y Restauración: Backup en caliente y frío, restauración completa o parcial.

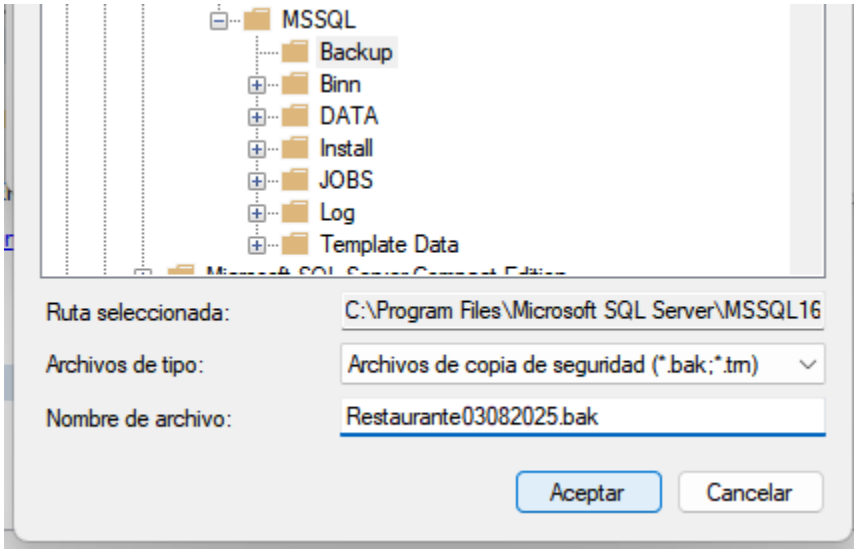
  RESPALDO

  Crear el backup

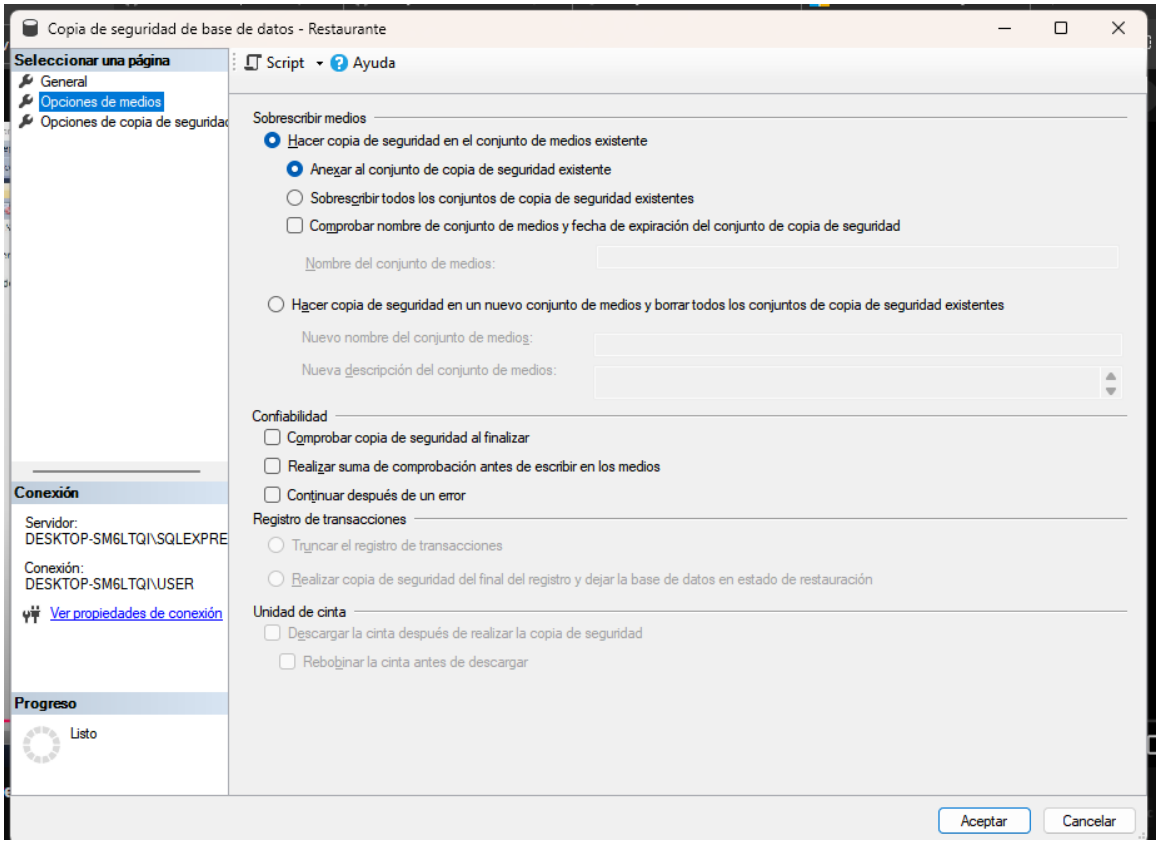


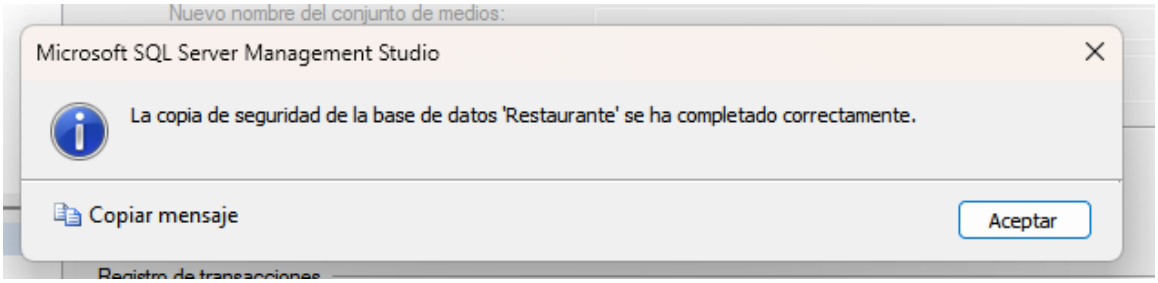


En este caso se va a modificar el nombre del bak

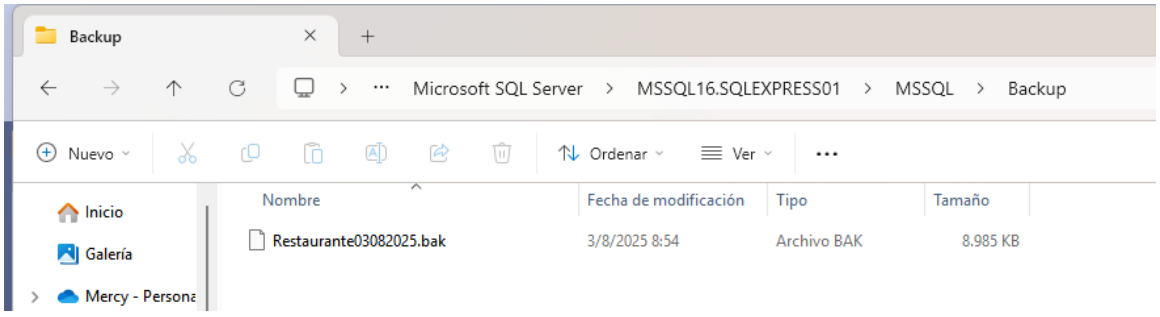


Ir a opciones de medio y aceptar

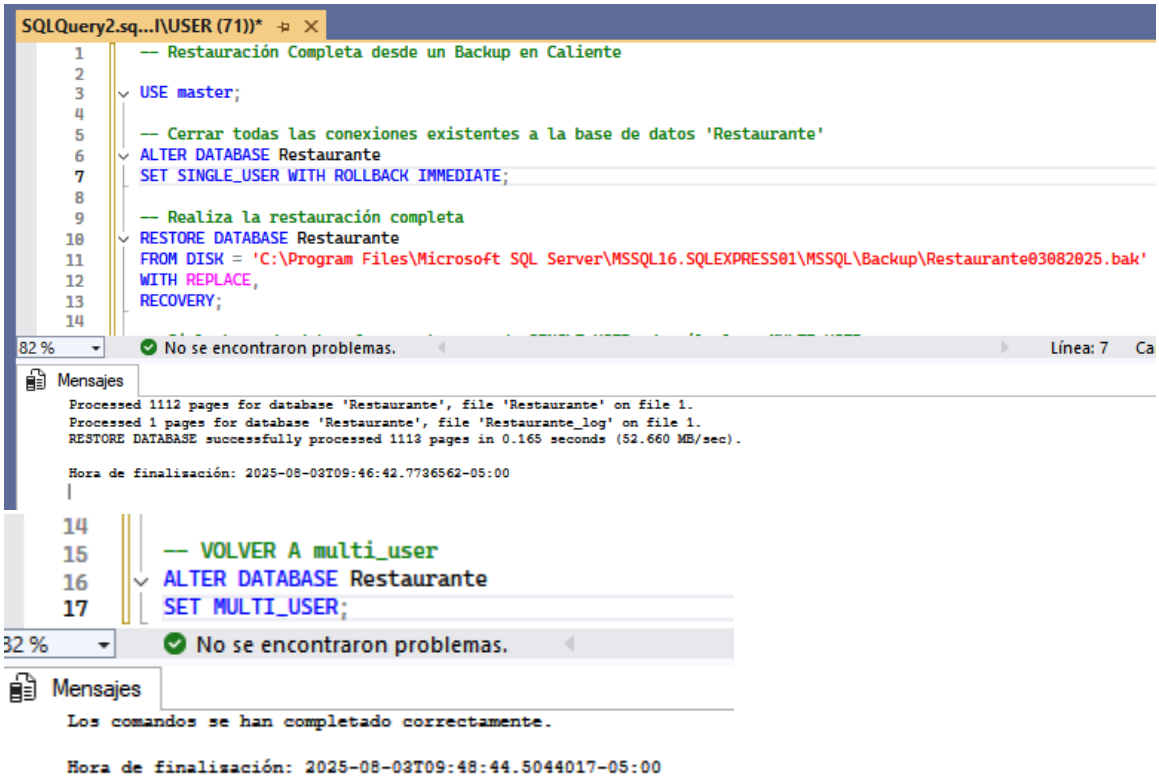




Verificar que se creó

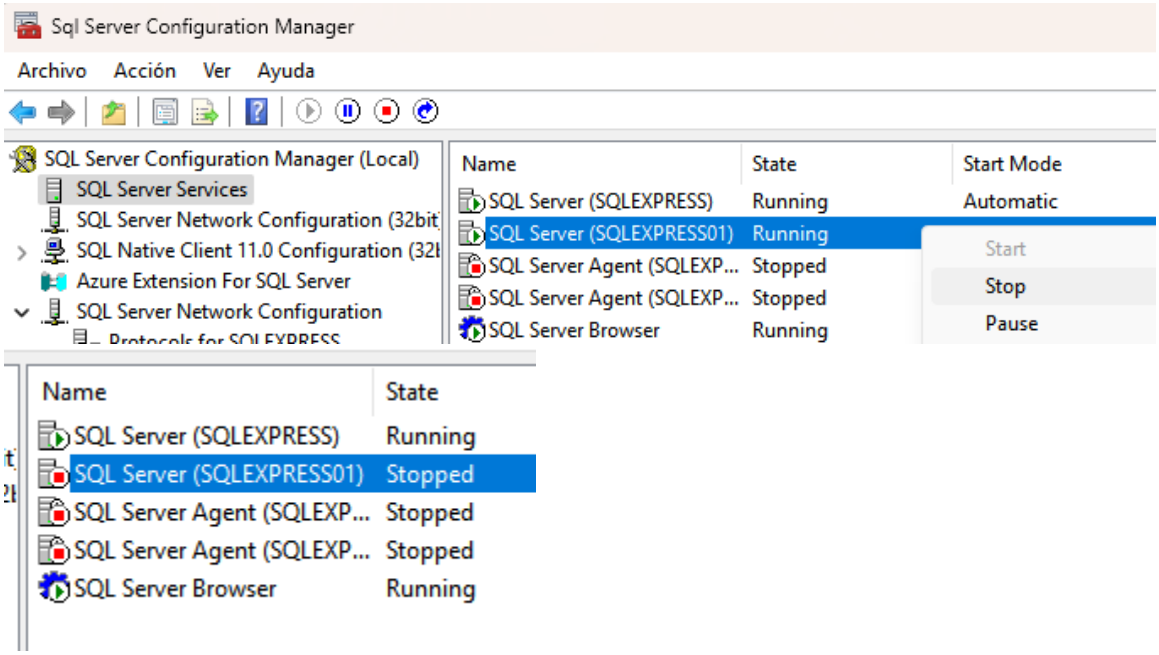


RESTAURACIÓN BACKUP CALIENTE

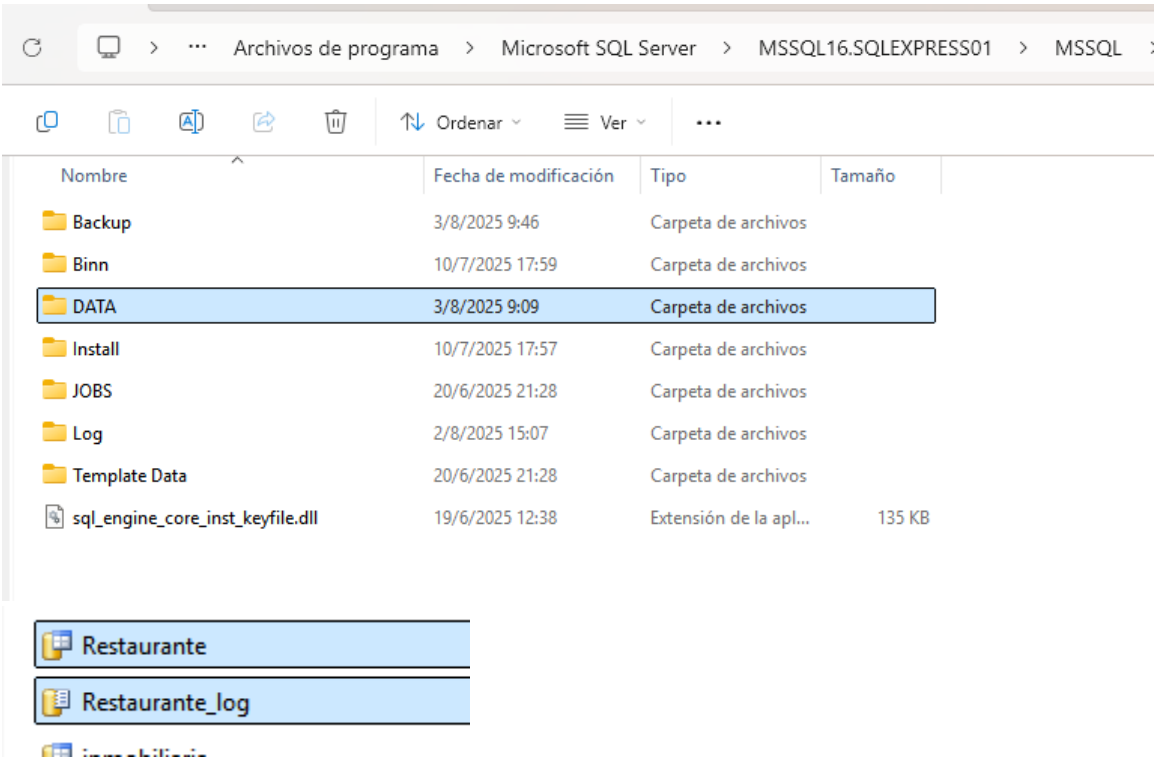


BACKUP FRIO

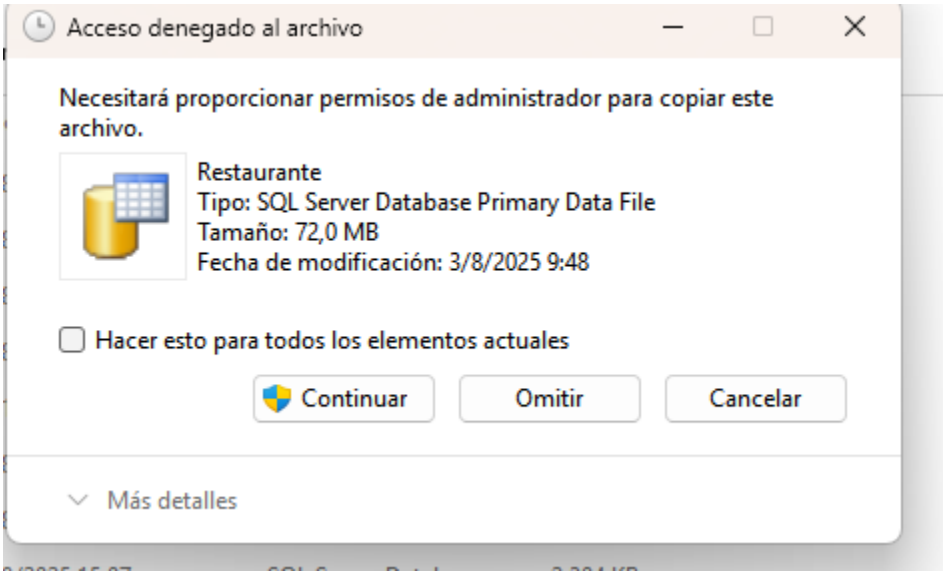
Detener el servicio

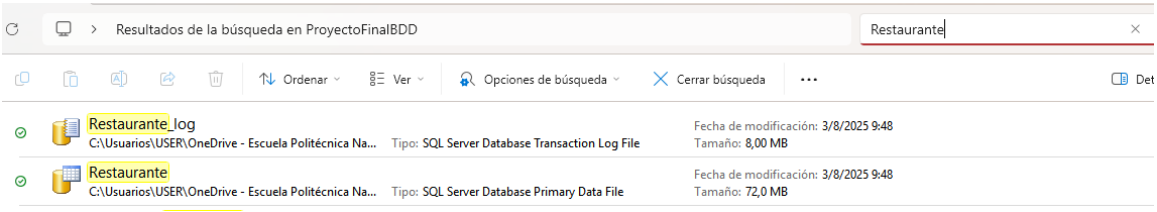


Dirigirnos a los archivos de la base de datos

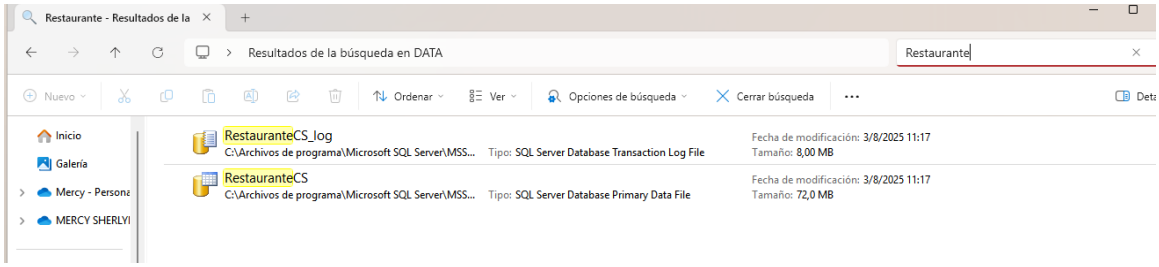


Copiamos los archivos en una ruta segura

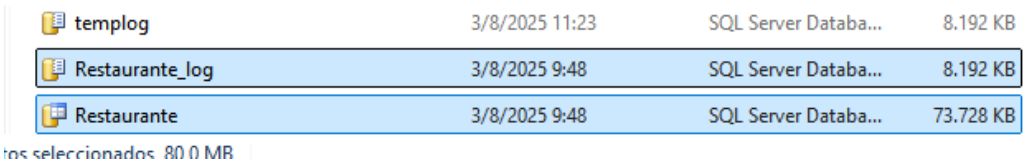




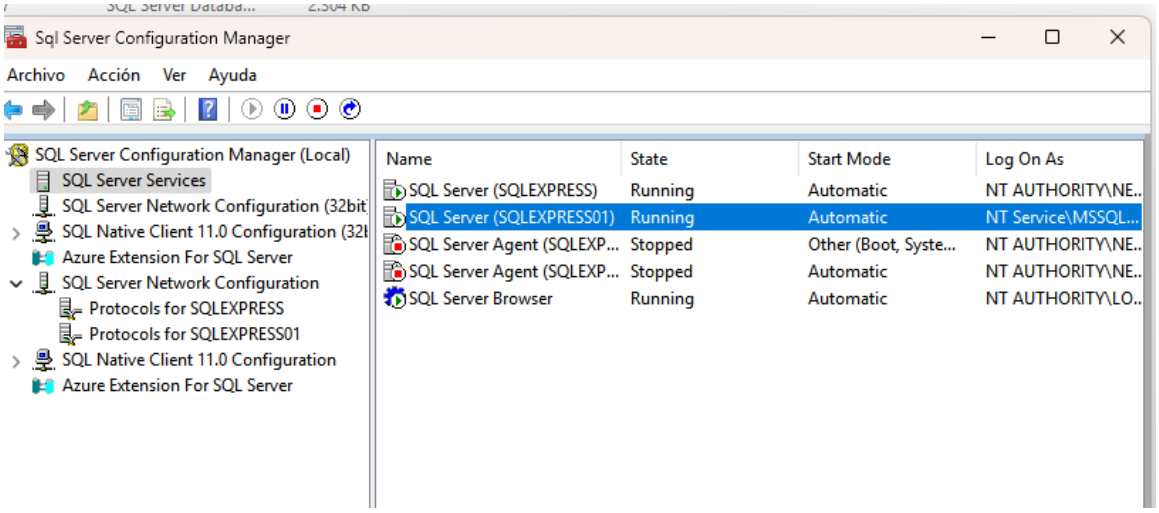
Eliminar los archivos del Restaurante ‘dañado’



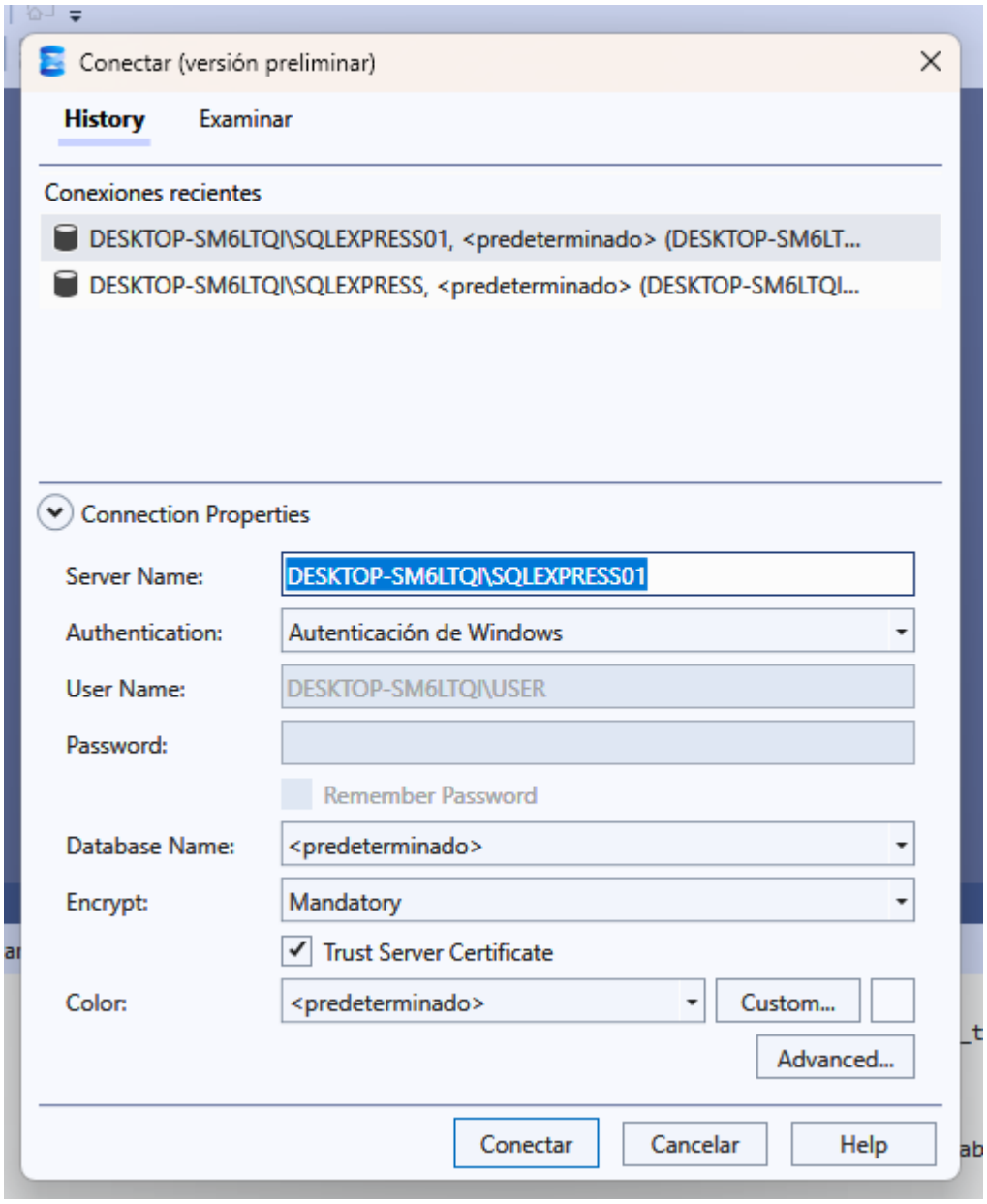
Copia los archivos a la data

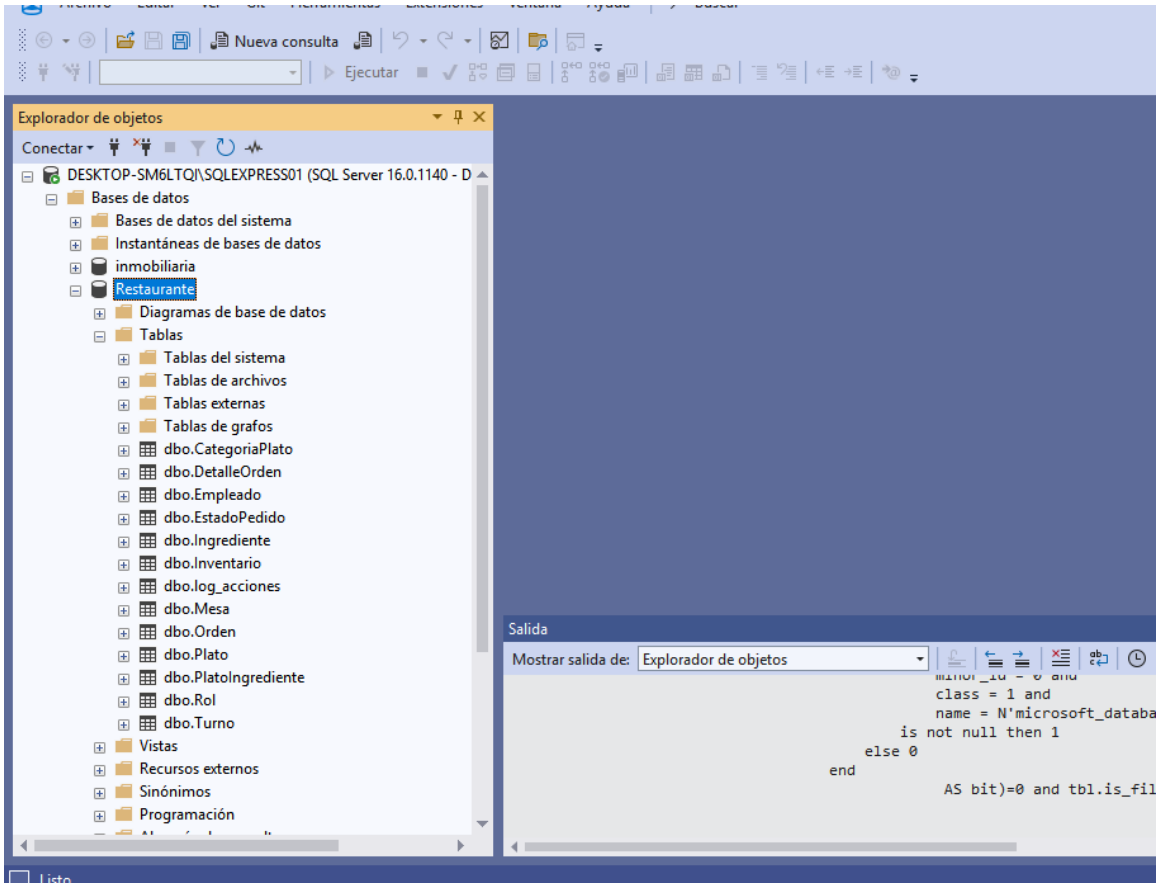


volver a iniciar los servicios de sql sever



Conectar al servidor


Conectar (versión preliminar)
History
Examinar
Conexiones recientes
DESKTOP-SM6LTQI\SQLEXPRESS01, <predeterminado> (DESKTOP-SM6LT...
DESKTOP-SM6LTQI\SQLEXPRESS, <predeterminado> (DESKTOP-SM6LTQI...
Connection Properties
Server Name:
DESKTOP-SM6LTQI\SQLEXPRESS01
Authentication:
Autenticación de Windows
User Name:
DESKTOP-SM6LTQI\USER
Password:
Remember Password
Database Name:
<predeterminado>
Encrypt:
Mandatory
Trust Server Certificate
Color:
<predeterminado>
Custom...
Advanced...
Conectar
Cancelar
Help



- Seguridad ante SQL Injection: Simulación y protección con consultas preparadas.

```
USE Restaurante;

-- Simulación de una consulta VULNERABLE a SQL Injection
-- NO USAR ESTE MÉTODO EN APLICACIONES REALES.
-- Esta es solo una demostración de cómo NO construir consultas.

-- Escenario normal (entrada esperada)
DECLARE @nombre_buscado_normal VARCHAR(100) = 'Juan García';
DECLARE @sql_vulnerable_normal NVARCHAR(MAX);

SET @sql_vulnerable_normal = 'SELECT id_empleado, nombre_empleado, cedula FROM Empleado WHERE nombre_empleado = ''' + @nombre_buscado_normal + '''';

PRINT 'Consulta normal (vulnerable):';
PRINT @sql_vulnerable_normal;
-- prueba
EXEC sp_executesql @sql_vulnerable_normal; -- Si la ejecutaras, buscaría 'Juan García'
```

81 % No se encontraron problemas. Línea: 14 Carácter: 30 TABULACIONES

Resultados Mensajes

	id_empleado	nombre_empleado	cedula
1	66	Juan García	0x01761FA1A0556A05D50528549A2EBD27389B466943D4BD0...
2	441	Juan García	0x01D31D816558DA95E9C3BD8A2AF79F0D32B2E42FF0EA22...

Datos vulnerados

```sql
-- Escenario de ATAQUE (entrada maliciosa)
DECLARE @nombre_buscado_ataque VARCHAR(100) = ''' OR 1=1 --'; -- la condición WHERE siempre será verdadera, devolviendo TODOS los empleados.
DECLARE @sql_vulnerable_ataque NVARCHAR(MAX);

SET @sql_vulnerable_ataque = 'SELECT id_empleado, nombre_empleado, cedula FROM Empleado WHERE nombre_empleado = ''' + @nombre_buscado_ataque + '''';

PRINT CHAR(13) + CHAR(10) + 'Consulta con INTENTO de ATAQUE (vulnerable):';
PRINT @sql_vulnerable_ataque;

EXEC sp_executesql @sql_vulnerable_ataque; -- codigo de ejecución
```

No se encontraron problemas. Línea: 28 Carácter: 66 TABULACIONE

Resultados | Mensajes

id_empleado	nombre_empleado	cedula
55	Sofía Martínez	0x012205B9C696256D827F49CE7E0D98C43BD1F956B05058D...
56	Pedro Flores	0x017365ED5DE295424F678F233A8C0C6C145F21201E4F57A4...
57	Luis Sánchez	0x018A6E35822D2BFC95FBFBA0BBCE565EE1F30C1B4702EF1...
58	Sofía Torres	0x01F4755DEC422A61BB1D289A36F087166DA46F9CCFEDDE8...
59	María Pérez	0x0108EFFD64FA61E9B37BB81C2897E0230D21C27060B65FD...
60	Carlos Martínez	0x0149674A87CD214C1850C711AD7F2C7352D8324185AD34A...
61	Luis González	0x0133FA76F2DE9A17D67BB3D9389869D2C741D625303B938...
62	María Rodríguez	0x019ACDABFCD1B06BAAB50C1CF3B9A6A67140D1F416597B...
63	Laura Sánchez	0x0135AB449B1DC1D13F7B3365353AB846B823B9B940CCD16...
64	Carlos González	0x01791180BD05809B647146D7B2B8724F49389CF6340D5AD4...
65	Pedro González	0x01A79709F42723F9D5FCC09DA10B10907409B9F1B7E52F69...
66	Juan García	0x01761FA1A0556A05D50528549A2EBD27389B466943D4BD0...

Protección

```sql
USE Restaurante;

-- Protección contra SQL Injection con Consultas Preparadas (sp_executesql)
-- 1. Escenario normal (entrada esperada)
DECLARE @nombre_buscado_seguro VARCHAR(100) = 'Juan García';
DECLARE @sql_seguro NVARCHAR(MAX);
DECLARE @parametros NVARCHAR(MAX);

-- Define la consulta SQL con un marcador de posición para el parámetro (@pNombre)
SET @sql_seguro = N'SELECT id_empleado, nombre_empleado, cedula FROM Empleado WHERE nombre_empleado = @pNombre;';

SET @parametros = N'@pNombre VARCHAR(100)'; -- Definir los parámetros para la consulta

PRINT 'Consulta segura (normal):';
PRINT @sql_seguro;
EXEC sp_executesql @sql_seguro, @parametros, @pNombre = @nombre_buscado_seguro; -- Ejecuta la consulta pasando el valor como un parámetro
PRINT CHAR(13) + CHAR(10) + 'Resultados de la consulta segura (normal):';

-- 2. Escenario de ATAQUE (entrada maliciosa)
```

74 % 3 0 Línea: 12

Resultados | Mensajes

	id_empleado	nombre_empleado	cedula
1	66	Juan García	0x01761FA1A0556A05D50528549A2EBD27389B466943D4BD0...
2	441	Juan García	0x01D31D816558DA95E9C3BD8A2AF79F0D32B2E42FF0EA22...

```sql
-- 2. Escenario de ATAQUE (entrada maliciosa)
-- La misma entrada maliciosa del ejemplo vulnerable: ' OR 1=1 --
DECLARE @sql_seguro NVARCHAR(MAX);
DECLARE @parametros NVARCHAR(MAX);

DECLARE @nombre_buscado_ataque_seguro VARCHAR(100) = ''' OR 1=1 --';
PRINT CHAR(13) + CHAR(10) + 'Consulta segura con INTENTO de ATAQUE:';
PRINT @sql_seguro;
-- Ejecuta la consulta pasando la entrada maliciosa como un parámetro
EXEC sp_executesql @sql_seguro, @parametros, @pNombre = @nombre_buscado_ataque_seguro;
PRINT CHAR(13) + CHAR(10) + 'Resultados de la consulta segura (con ataque):';
```

6 3 0

Mensajes

```
Consulta segura con INTENTO de ATAQUE:


Resultados de la consulta segura (con ataque):

Hora de finalización: 2025-08-03T12:08:54.8447668-05:00
```

```sql
-- Protección contra SQL Injection con Consultas Preparadas (sp_executesql)
-- 1. Escenario normal (entrada esperada)
DECLARE @nombre_buscado_seguro VARCHAR(100) = 'Juan García';
DECLARE @sql_seguro NVARCHAR(MAX);
DECLARE @parametros NVARCHAR(MAX);

-- Define la consulta SQL con un marcador de posición para el parámetro (@pNombre)
SET @sql_seguro = N'SELECT id_empleado, nombre_empleado, cedula FROM Empleado WHERE nombre_empleado = @pNombre;';

SET @parametros = N'@pNombre VARCHAR(100)'; -- Definir los parámetros para la consulta

PRINT 'Consulta segura (normal):';
PRINT @sql_seguro;
EXEC sp_executesql @sql_seguro, @parametros, @pNombre = @nombre_buscado_seguro; -- Ejecuta la consulta pasando el valor como un parámetro
PRINT CHAR(13) + CHAR(10) + 'Resultados de la consulta segura (normal):';

-- 2. Escenario de ATAQUE (entrada maliciosa)
-- Codigo malicioso ' OR 1=1 --
DECLARE @nombre_buscado_ataque_seguro VARCHAR(100) = ''' OR 1=1 --';
PRINT CHAR(13) + CHAR(10) + 'Consulta segura con INTENTO de ATAQUE:';
PRINT @sql_seguro;
-- Ejecuta la consulta pasando la entrada maliciosa como un parámetro
EXEC sp_executesql @sql_seguro, @parametros, @pNombre = @nombre_buscado_ataque_seguro;
PRINT CHAR(13) + CHAR(10) + 'Resultados de la consulta segura (con ataque):';
```

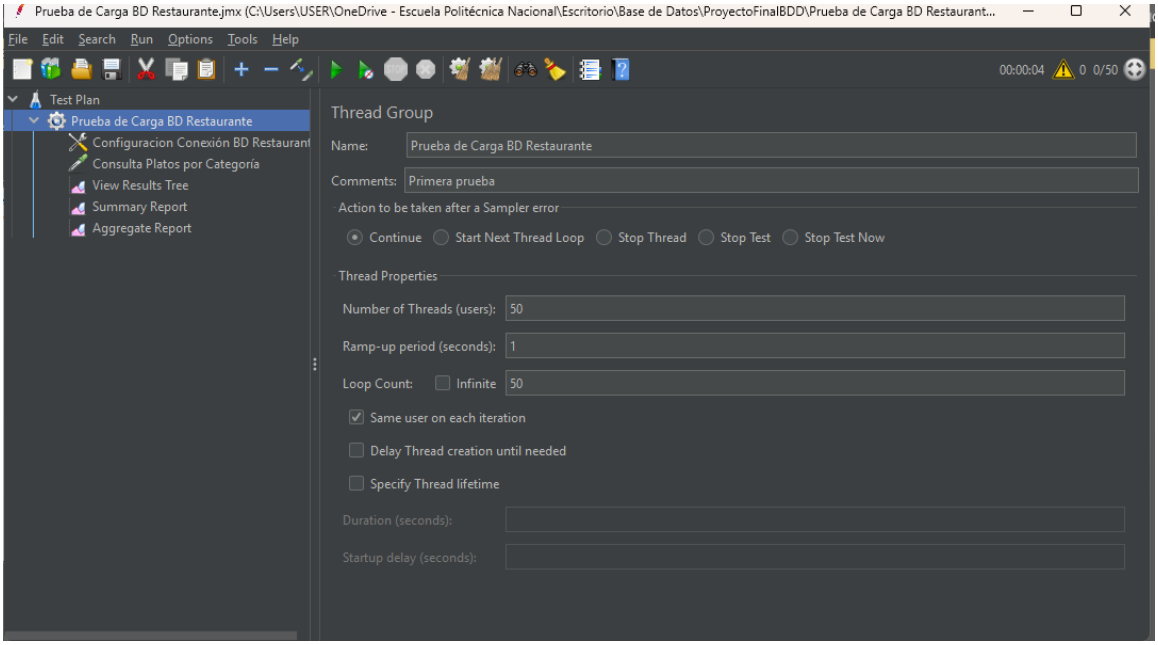
4 % 3 0 Línea: 24 Carácte

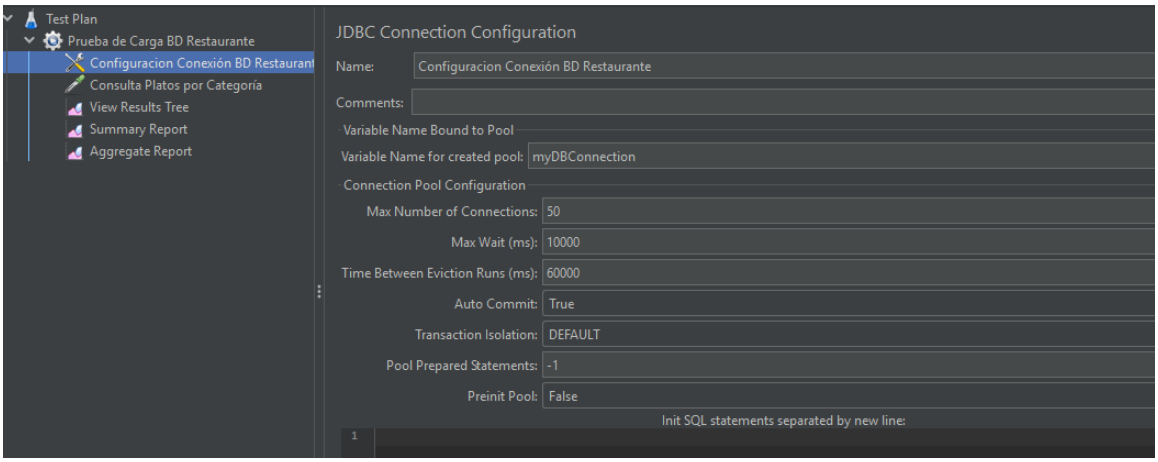
Resultados Mensajes

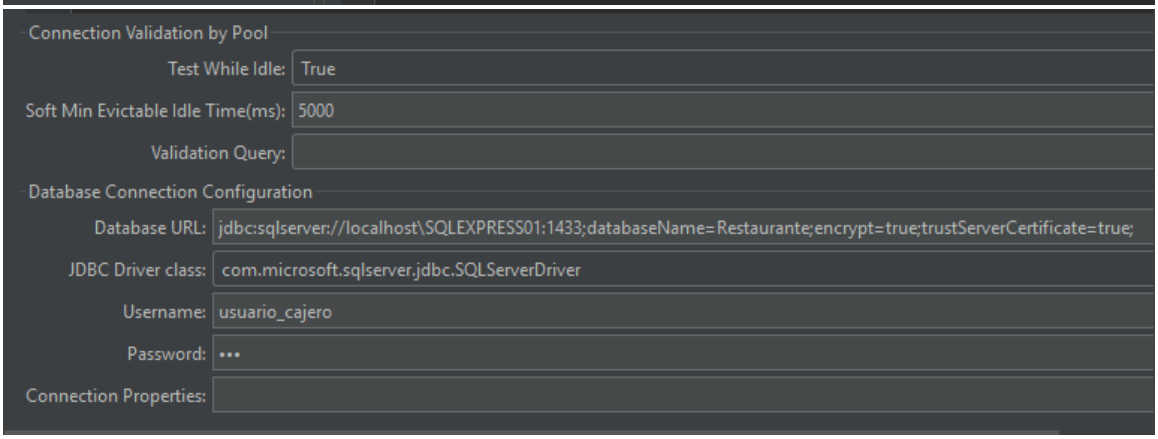
	id_empleado	nombre_empleado	cedula
1	66	Juan García	0x01761FA1A0556A05D50528549A2EBD27389B466943D4BD0...
2	441	Juan García	0x01D31D816558DA95E9C3BD8A2AF79F0D32B2E42FF0EA22...

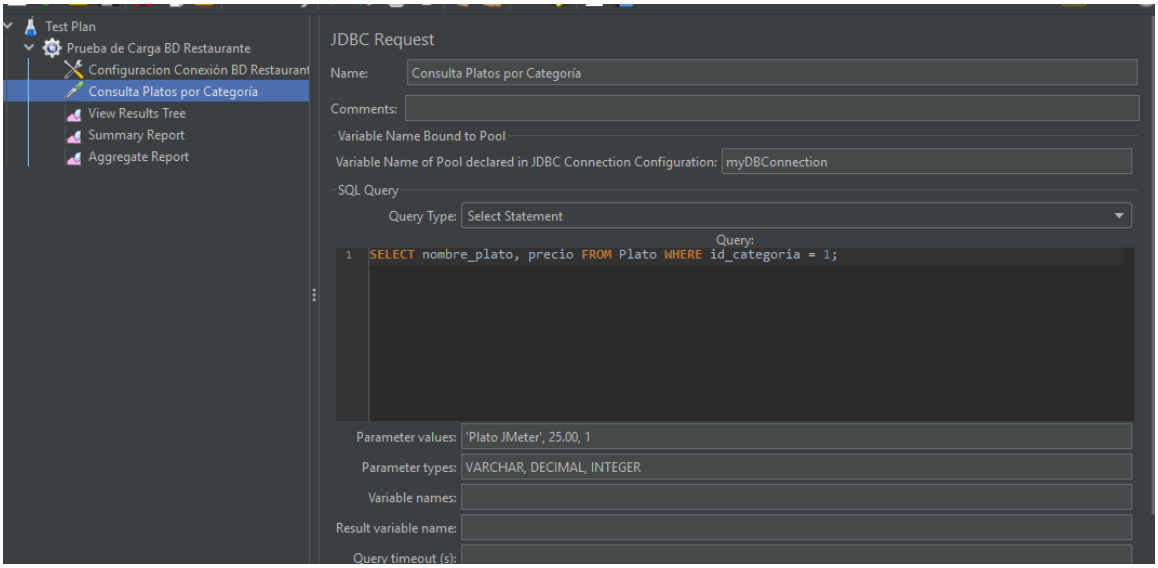
id_empleado	nombre_empleado	cedula

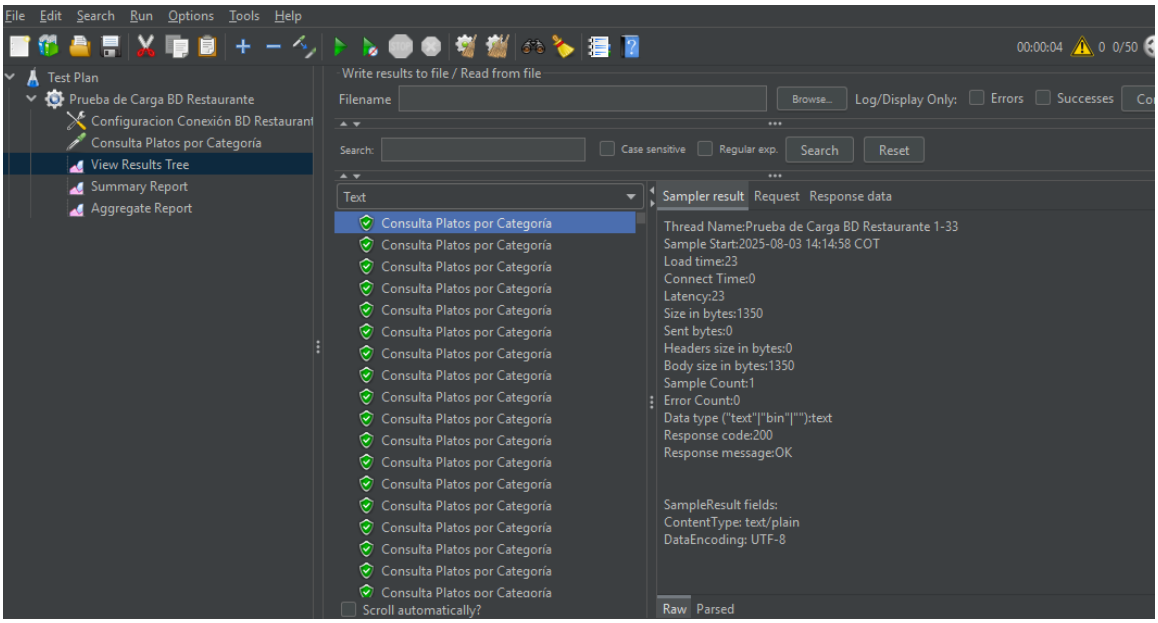
- Monitoreo y Rendimiento: Evaluación de consultas, crecimiento de registros, uso de recursos.

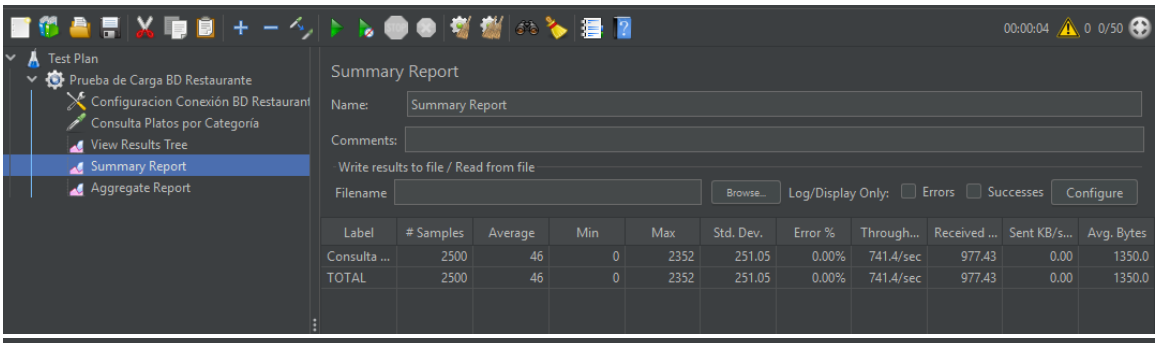


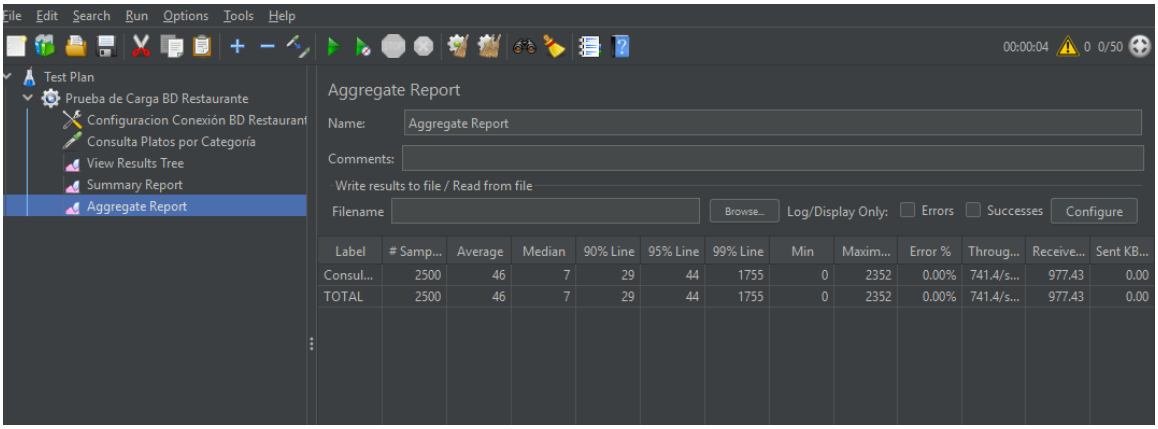

Test Plan
Prueba de Carga BD Restaurante
Configuracion Conexión BD Restaurant
Consulta Platos por Categoría
View Results Tree
Summary Report
Aggregate Report
JDBC Connection Configuration
Name: Configuracion Conexión BD Restaurante
Comments:
Variable Name Bound to Pool
Variable Name for created pool: myDBConnection
Connection Pool Configuration
Max Number of Connections: 50
Max Wait (ms): 10000
Time Between Eviction Runs (ms): 60000
Auto Commit: True
Transaction Isolation: DEFAULT
Pool Prepared Statements: -1
Preinit Pool: False
Init SQL statements separated by new line:
1


Connection Validation by Pool
Test While Idle: True
Soft Min Evictable Idle Time(ms): 5000
Validation Query:
Database Connection Configuration
Database URL: jdbc:sqlserver://localhost\SQLEXPRESS01:1433;databaseName=Restaurante;encrypt=true;trustServerCertificate=true;
JDBC Driver class: com.microsoft.sqlserver.jdbc.SQLServerDriver
Username: usuario_cajero
Password: •••
Connection Properties:


Test Plan
Prueba de Carga BD Restaurante
Configuracion Conexión BD Restaurant
Consulta Platos por Categoría
View Results Tree
Summary Report
Aggregate Report
JDBC Request
Name: Consulta Platos por Categoría
Comments:
Variable Name Bound to Pool
Variable Name of Pool declared in JDBC Connection Configuration: myDBConnection
SQL Query
Query Type: Select Statement
Query:
1 SELECT nombre_plato, precio FROM Plato WHERE id_categoria = 1;
Parameter values: 'Plato JMeter', 25.00, 1
Parameter types: VARCHAR, DECIMAL, INTEGER
Variable names:
Result variable name:
Query timeout (s):







- Protección de Datos y Gestión Crítica: Cifrado, anonimización, integridad lógica.
  Cifrado, ya se hizo

  Anonimizacion

```sql
-- La función 'default()' oculta el nombre completo para VARCHAR/NVARCHAR.
ALTER TABLE Empleado
ALTER COLUMN nombre_empleado ADD MASKED WITH (FUNCTION = 'default()');

-- Crear un usuario de prueba para la demostracion
IF NOT EXISTS (SELECT name FROM sys.database_principals WHERE name = 'UsuarioPrueba')
BEGIN
    CREATE USER UsuarioPrueba WITHOUT LOGIN;
END
GRANT SELECT ON Empleado TO UsuarioPrueba; -- otorgar permisos

-- Demostrar el enmascaramiento:
-- Para ver los datos enmascarados, ejecuta las siguientes líneas en SSMS
-- (asegúrate de que tu usuario actual tenga permisos para ejecutar AS).
PRINT 'Consulta como UsuarioPrueba';
EXECUTE AS USER = 'UsuarioPrueba';
SELECT id_empleado, nombre_empleado, cedula FROM Empleado;
REVERT;
```

79 % — No se encontraron problemas.

Resultados | Mensajes

	id_empleado	nombre_empleado	cedula
1	1	xxxx	0x0167F09191DAF48D2204034FDE8C2E1AAD2CF267D448B36...
2	2	xxxx	0x017CFDA7A020F31F7D306160C19F137E4F9E3898B62CFCD4...
3	3	xxxx	0x0106021898051643365FC0641908E12229846DA312BDD127...
4	4	xxxx	0x01FB1C7C4989E45F2C6C32A14971CE0FA2AA98304D06B5F6...
5	5	xxxx	0x018180D276EF242C9E5C1BFC4E5CD9D8406C52AA343F7B4...
6	6	xxxx	0x01D2923EDF5A22AB0F9C4663462E3C7458D04722AE544477...
7	7	xxxx	0x011839EE10D7AF2E943464EC632FA581003A61EAB5A1F40B...
8	8	xxxx	0x01DA0271B824E523187B895F03A7EC5167C8D4B3CE395590...
9	9	xxxx	0x019FFEB5CF33E049770CC37DD3EDDCCB6EB4C26F34CE76...
10	10	xxxx	0x01CD0A4A6EED27FA3E4C9FFB8798C43C169B39BD9F5458F...
11	11	xxxx	0x01230AC163BADFA8520A1527874EC26333557A68A4D17F94...
12	12	xxxx	0x0194CE97D5955092CE26D61EB12C03BC1952A98181A625D...
13	13	xxxx	0x015D9E016A80720A2081D6264400873404452824CDF66D41...

Consulta ejecutada correctamente. DESKTOP-SM6LTQI\SQLEX

```sql
REVERT;
PRINT 'Consulta como usuario original '; -- se podrá visualizar el nombre completo, sin ningún enmascaramiento
SELECT id_empleado, nombre_empleado, cedula FROM Empleado;
```

79 % — No se encontraron problemas.

Resultados | Mensajes

	id_empleado	nombre_empleado	cedula
1	1	Pedro González	0x0167F09191DAF48D2204034FDE8C2E1AAD2CF267D448B36...
2	2	Diego Torres	0x017CFDA7A020F31F7D306160C19F137E4F9E3898B62CFCD4...
3	3	María Martínez	0x0106021898051643365FC0641908E12229846DA312BDD127...
4	4	Juan Sánchez	0x01FB1C7C4989E45F2C6C32A14971CE0FA2AA98304D06B5F6...
5	5	Pedro García	0x018180D276EF242C9E5C1BFC4E5CD9D8406C52AA343F7B4...
6	6	Pedro Torres	0x01D2923EDF5A22AB0F9C4663462E3C7458D04722AE544477...
7	7	María Flores	0x011839EE10D7AF2E943464EC632FA581003A61EAB5A1F40B...
8	8	Pedro López	0x01DA0271B824E523187B895F03A7EC5167C8D4B3CE395590...
9	9	Pedro Martínez	0x019FFEB5CF33E049770CC37DD3EDDCCB6EB4C26F34CE76...
10	10	Carlos López	0x01CD0A4A6EED27FA3E4C9FFB8798C43C169B39BD9F5458F...
11	11	Ana López	0x01230AC163BADFA8520A1527874EC26333557A68A4D17F94...
12	12	Laura Martínez	0x0194CE97D5955092CE26D61EB12C03BC1952A98181A625D...
13	13	Diego Pérez	0x015D9E016A80720A2081D6264400873404452824CDF66D41...
14	14	Diego Pérez	0x011B9D03EE1D2629D1F045890F4BF0889256B7783F1DCF7E...
15	15	María Torres	0x01A19E5D66660FA73AA5C2B84AD435A5564771DD21866AF...

Integridad lógica

```sql
USE Restaurante;

-- Integridad Lógica de Datos con Restricciones CHECK: asegura que los datos cumplan ciertas reglas de negocio.

-- Añadir una restricción CHECK a la tabla Plato: el precio debe ser positivo
ALTER TABLE Plato
ADD CONSTRAINT CK_Plato_PrecioPositivo CHECK (precio > 0);

-- Probar, esto debería fallar
PRINT 'Intentando insertar un plato con precio negativo (debería fallar)';
BEGIN TRY
    INSERT INTO Plato (nombre_plato, precio, id_categoria) VALUES ('Plato Negativo', -5.00, 1);
END TRY
BEGIN CATCH
    PRINT ERROR_MESSAGE();
END CATCH;
```

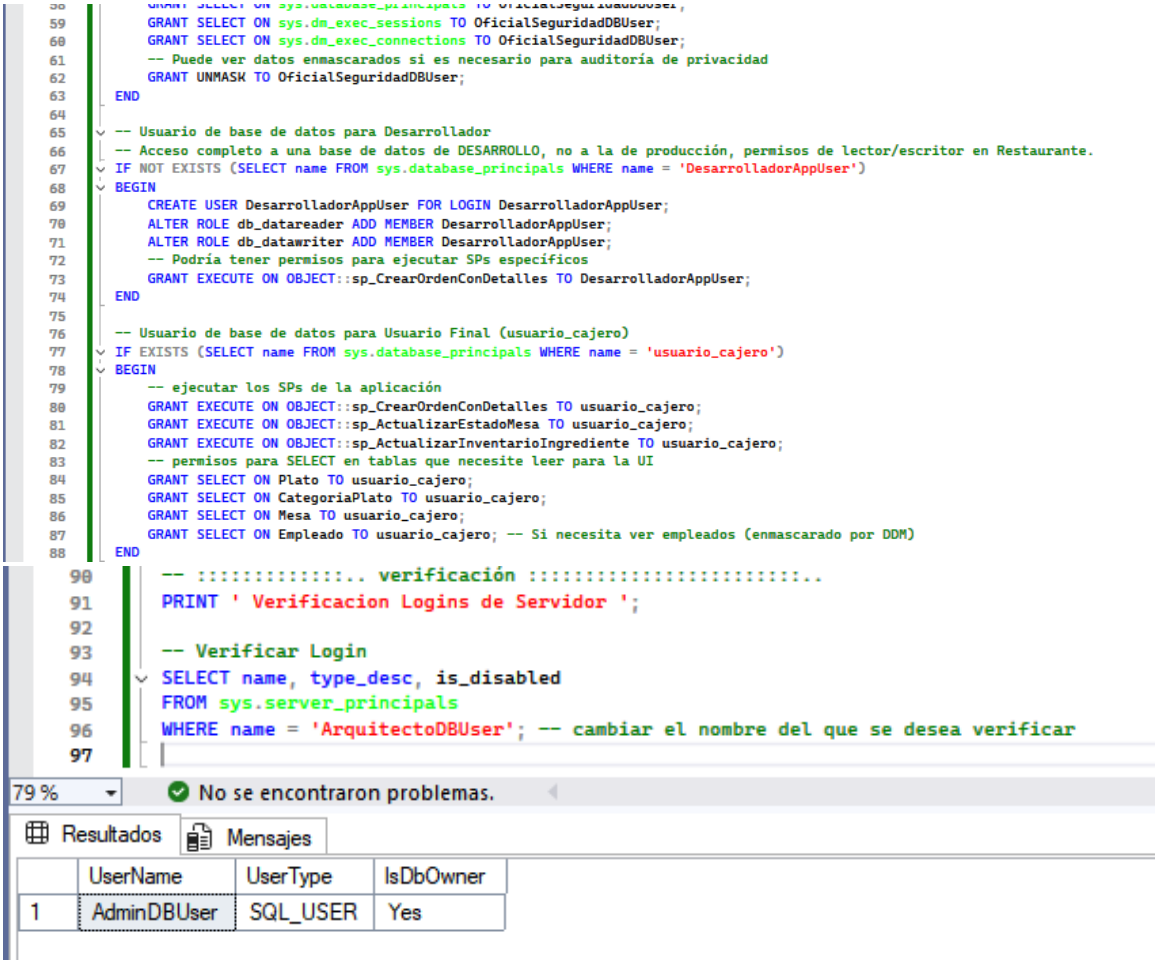
79 % — No se encontraron problemas.

Mensajes

```
Intentando insertar un plato con precio negativo (debería fallar)

(0 filas afectadas)
The INSERT statement conflicted with the CHECK constraint "CK_Plato_PrecioPositivo". The conflict occurred in database "Restaurante", table "dbo.Plato", column 'precio'.

Hora de finalización: 2025-08-03T15:51:12.4653663-05:00
```

- Simulación de Perfiles Profesionales: Roles como administrador, arquitecto, oficial de seguridad, desarrollador y usuario final.

```sql
USE master;

-- 1. Crear Logins de SQL Server para cada perfil

-- Login para Administrador de Base de Datos
IF NOT EXISTS (SELECT name FROM sys.server_principals WHERE name = 'AdminDBUser')
BEGIN
    CREATE LOGIN AdminDBUser WITH PASSWORD = '123', CHECK_EXPIRATION = OFF, CHECK_POLICY = OFF;
END

-- Login para Arquitecto de Base de Datos
IF NOT EXISTS (SELECT name FROM sys.server_principals WHERE name = 'ArquitectoDBUser')
BEGIN
    CREATE LOGIN ArquitectoDBUser WITH PASSWORD = 'PasswordArquitecto123', CHECK_EXPIRATION = OFF, CHECK_POLICY = OFF;
END

-- Login para Oficial de Seguridad de Base de Datos
IF NOT EXISTS (SELECT name FROM sys.server_principals WHERE name = 'OficialSeguridadDBUser')
BEGIN
    CREATE LOGIN OficialSeguridadDBUser WITH PASSWORD = 'PasswordSeguridad123', CHECK_EXPIRATION = OFF, CHECK_POLICY = OFI
END

-- Login para Desarrollador de Aplicaciones (acceso a DB de desarrollo, no producción)
IF NOT EXISTS (SELECT name FROM sys.server_principals WHERE name = 'DesarrolladorAppUser')
BEGIN
    CREATE LOGIN DesarrolladorAppUser WITH PASSWORD = 'PasswordDesarrollador123', CHECK_EXPIRATION = OFF, CHECK_POLICY = (
END

-- usuario final: usuario_cajero (ya creado)

-- :::::::::::::::: Crear Usuarios de Base de Datos y Asignar Roles/Permisos ::::::::::::::::::::::::::::::::::::

-- Usuario de base de datos para Administrador
IF NOT EXISTS (SELECT name FROM sys.database_principals WHERE name = 'AdminDBUser')
BEGIN
    CREATE USER AdminDBUser FOR LOGIN AdminDBUser;
    ALTER ROLE db_owner ADD MEMBER AdminDBUser;
END

-- Usuario de base de datos para Arquitecto: puede ver la estructura, crear objetos, pero no eliminar la DB.
IF NOT EXISTS (SELECT name FROM sys.database_principals WHERE name = 'ArquitectoDBUser')
BEGIN
    CREATE USER ArquitectoDBUser FOR LOGIN ArquitectoDBUser;
    ALTER ROLE db_ddladmin ADD MEMBER ArquitectoDBUser; -- Puede crear, modificar, eliminar objetos
    ALTER ROLE db_datareader ADD MEMBER ArquitectoDBUser; -- Puede leer datos
    ALTER ROLE db_datawriter ADD MEMBER ArquitectoDBUser; -- Puede escribir datos
END

-- Usuario de base de datos para Oficial de Seguridad: puede ver configuraciones de seguridad, logs, pero no modificar datos o estructura.
IF NOT EXISTS (SELECT name FROM sys.database_principals WHERE name = 'OficialSeguridadDBUser')
BEGIN
    CREATE USER OficialSeguridadDBUser FOR LOGIN OficialSeguridadDBUser;
    -- Permisos específicos para seguridad y auditoría
    GRANT VIEW SERVER STATE TO OficialSeguridadDBUser;
    GRANT VIEW ANY DATABASE TO OficialSeguridadDBUser;
    GRANT VIEW DEFINITION TO OficialSeguridadDBUser;
    GRANT SELECT ON sys.server_principals TO OficialSeguridadDBUser;
    GRANT SELECT ON sys.database_principals TO OficialSeguridadDBUser;
```



```
        GRANT SELECT ON sys.dm_exec_sessions TO OficialSeguridadDBUser;
        GRANT SELECT ON sys.dm_exec_connections TO OficialSeguridadDBUser;
        -- Puede ver datos enmascarados si es necesario para auditoría de privacidad
        GRANT UNMASK TO OficialSeguridadDBUser;
    END

    -- Usuario de base de datos para Desarrollador
    -- Acceso completo a una base de datos de DESARROLLO, no a la de producción, permisos de lector/escritor en Restaurante.
    IF NOT EXISTS (SELECT name FROM sys.database_principals WHERE name = 'DesarrolladorAppUser')
    BEGIN
        CREATE USER DesarrolladorAppUser FOR LOGIN DesarrolladorAppUser;
        ALTER ROLE db_datareader ADD MEMBER DesarrolladorAppUser;
        ALTER ROLE db_datawriter ADD MEMBER DesarrolladorAppUser;
        -- Podría tener permisos para ejecutar SPs específicos
        GRANT EXECUTE ON OBJECT::sp_CrearOrdenConDetalles TO DesarrolladorAppUser;
    END

    -- Usuario de base de datos para Usuario Final (usuario_cajero)
    IF EXISTS (SELECT name FROM sys.database_principals WHERE name = 'usuario_cajero')
    BEGIN
        -- ejecutar los SPs de la aplicación
        GRANT EXECUTE ON OBJECT::sp_CrearOrdenConDetalles TO usuario_cajero;
        GRANT EXECUTE ON OBJECT::sp_ActualizarEstadoMesa TO usuario_cajero;
        GRANT EXECUTE ON OBJECT::sp_ActualizarInventarioIngrediente TO usuario_cajero;
        -- permisos para SELECT en tablas que necesite leer para la UI
        GRANT SELECT ON Plato TO usuario_cajero;
        GRANT SELECT ON CategoriaPlato TO usuario_cajero;
        GRANT SELECT ON Mesa TO usuario_cajero;
        GRANT SELECT ON Empleado TO usuario_cajero; -- Si necesita ver empleados (enmascarado por DDM)
    END
    -- ::::::::::::::.. verificación ::::::::::::::::::::::::::..
    PRINT ' Verificacion Logins de Servidor ';

    -- Verificar Login
    SELECT name, type_desc, is_disabled
    FROM sys.server_principals
    WHERE name = 'ArquitectoDBUser'; -- cambiar el nombre del que se desea verificar
```

79 % No se encontraron problemas.

Resultados Mensajes

	UserName	UserType	IsDbOwner
1	AdminDBUser	SQL_USER	Yes

- Validación Final del Proyecto: Checklist, pruebas de rendimiento, seguridad, documentación y defensa técnica.

4. Descripción del Proyecto: Restaurante

- Menú con platos, ingredientes, precio y categoría.
- Registro de comandas con asignación de mesa y comensales.
- Seguimiento del estado del pedido: solicitado, en cocina, servido.
- Cierre de cuenta automático con cálculo de propina.
- Control de disponibilidad de ingredientes por día.
- Reportes: consumo diario, platos más pedidos, ventas por turno.
- Vistas para mesero, cocina, gerente.