UNEMI
UNIVERSIDAD ESTATAL DE MILAGRO

FACULTAD DE CIENCIAS E INGENIERÍA
CARRERA DE INGENIERÍA DE SOFTWARE

TEMA:

Formatos y vistas, Procedimientos almacenados, Funciones

Auditoría de bases de datos, Disparadores

AUTORES:

Jean Pierre Vivanco Arreaga

Emiro Josue Gomez Lavayen

Ariel Fabricio Pilamunga Aucapiña

Elvis Jair Martillo Pacheco

Matias Gabriel Murcia Basante

FECHA DE ENTREGA:

2025 - 1 - 10

PERIODO:

Abril 2024 a Agosto 2025

MILAGRO-ECUADOR

Requisitos: • Base de datos creada previamente en la unidad 1.

Procedimientos almacenados

Crear un procedimiento almacenado que me permita crear un medicamento.

1. Se debe enviar todos los campos a excepción del id, debe enviar el nombre del tipo de medicamento.

2. El id debe generarlo automáticamente dentro del procedimiento.

3. Si el tipo de medicamento no existe entonces debe seleccionar uno aleatorio.

4. No se puede crear un medicamento si el nombre se encuentra registrado o si el stock es <= 0.

5. Si el precio es menor o igual 0.00 debe generar uno aleatorio, generar 2 a 5 si el genérico caso contrario 6 a 10.

```
CREATE OR ALTER PROCEDURE CP_CREAR_MEDICAMENTO
    (
    @NOMBRETIPO VARCHAR(100),
    @NOMBRE VARCHAR(100),
    @PRECIO REAL,
    @GENERICO BIT,
    @STOCK INT
)
AS
BEGIN
    DECLARE @ID INT
    DECLARE @IDTIPO INT
    SELECT @IDTIPO = IDTIPO
    FROM dbo.TIPO
```

```
    WHERE NOMBRE = @NOMBRETIPO

    SELECT @ID = ISNULL(MAX(IDMEDICAMENTO), 0) + 1

    FROM dbo.MEDICAMENTO;

    IF EXISTS(SELECT NOMBRE

        FROM dbo.MEDICAMENTO

        WHERE NOMBRE = @NOMBRE) OR @STOCK <= 0

        BEGIN

        PRINT('NOMBRE EXISTENTE O STOCK NULO')

    END

    ELSE

    BEGIN

        IF @IDTIPO IS NULL

        BEGIN

            SELECT TOP 1

                @IDTIPO = IDTIPO

            FROM dbo.TIPO

            ORDER BY NEWID()


              SELECT TOP 1 @NOMBRETIPO = NOMBRE FROM dbo.TIPO WHERE IDTIPO =
@IDTIPO

        END

        IF @PRECIO <= 0.00

        BEGIN

            IF @NOMBRETIPO = 'Genérico'

            BEGIN

                SET @PRECIO = RAND() * (5 -2 + 1) + 2

            END

            ELSE
```

```
            BEGIN

                SET @PRECIO = RAND() * (10 - 6 + 1) + 6

            END

        END

                                        INSERT      INTO      DBO.MEDICAMENTO
(IDMEDICAMENTO,IDTIPO,NOMBRE,PRECIO,GENERICO,STOCK)                    VALUES
(@ID,@IDTIPO,@NOMBRE,@PRECIO,@GENERICO,@STOCK);

    END

END
```

```
53    EXEC CP_CREAR_MEDICAMENTO 'Pastillas','Parasetamol',5.0,0,10
54    EXEC CP_CREAR_MEDICAMENTO 'Pastillas','Parasetamol',5.0,0,10
55    select * from dbo.MEDICAMENTO
56
57    GO
58
```

Messages

```
12:14:34 PM    Started executing query at Line 54
               NOMBRE EXISTENTE O STOCK NULO
               Total execution time: 00:00:00.023
```

Crear un procedimiento almacenado de salida que reciba como parámetro el nombre de un medicamento y un año para retornar el total de ventas obtenido. Si el año es nulo debe tomar el año actual.

```
CREATE OR ALTER PROCEDURE CP_VENTAS_MEDICAMENTOS

(

    @NOMBREMEDICAMENTO VARCHAR(100),

    @ANIO VARCHAR(4) = NULL

)

AS

BEGIN

    DECLARE @VENTAS INT
```

```
    IF @ANIO IS NULL
    BEGIN
        SET @ANIO = YEAR(GETDATE())
    END
    SELECT
        M.NOMBRE AS MEDICAMENTO,
        COUNT(DV.IDVENTA) AS VENTAS
    FROM dbo.VENTADETALLE AS DV
    INNER JOIN dbo.VENTA V
    ON DV.IDVENTA = V.IDVENTA
    INNER JOIN dbo.MEDICAMENTO M
    ON DV.IDMEDICAMENTO = M.IDMEDICAMENTO
    WHERE M.NOMBRE = @NOMBREMEDICAMENTO AND YEAR(V.FECHA_REGISTRO) = @ANIO
    GROUP BY M.NOMBRE
    ORDER BY VENTAS
END
GO
```

```
84
85  EXEC CP_VENTAS_MEDICAMENTOS 'Acetaminofén'
86
```

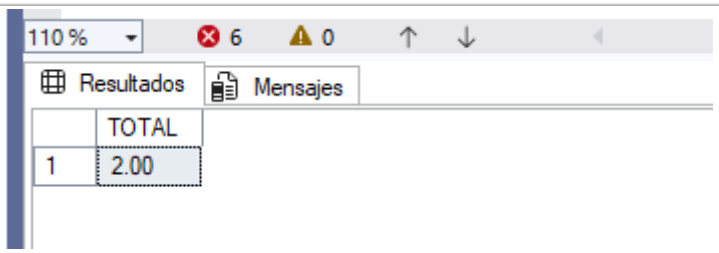
Results Messages

	MEDICAMENTO	VENTAS
1	Acetaminofén	1

Funciones

Crear una función que me permita obtener el total recaudado de un tipo de medicamento de un envío como parámetro, solo tomar en cuentas las ventas pagadas.

```
----FUNCIONES----

---TOTAL RECAUDADO---

CREATE OR ALTER FUNCTION TOTAL_RECAUDADO(

	@IDTIPO INT

)

RETURNS DECIMAL(10, 2)

AS

BEGIN

	DECLARE @TOTAL DECIMAL (10, 2) = 0

	SELECT @TOTAL = COALESCE(SUM(VD.CANT * VD.PRECIO), 0)

	FROM VENTA V JOIN VENTADETALLE VD

	ON V.IDVENTA = VD.IDVENTA JOIN MEDICAMENTO M

	ON VD.IDMEDICAMENTO = M.IDMEDICAMENTO JOIN TIPO T

	ON M.IDTIPO = T.IDTIPO

	WHERE T.IDTIPO = @IDTIPO

	AND V.PAGADO = 1


	RETURN @TOTAL

END

GO

SELECT DBO.TOTAL_RECAUDADO (1) AS [TOTAL]

GO
```



Crear una función que me permita obtener el total de dinero que adeuda un cliente, debe enviar el número de cédula

```
CREATE FUNCTION FN_TOTAL_DEUDA (@CEDULA VARCHAR(10))
RETURNS DECIMAL(10,2)
AS
BEGIN
    DECLARE @TOTAL_DEUDA DECIMAL(10,2);

    SELECT @TOTAL_DEUDA = ISNULL(SUM(V.TOTAL), 0)
    FROM CLIENTE C
    INNER JOIN VENTA V ON C.IDCLIENTE = V.IDCLIENTE
    WHERE C.CEDULA = @CEDULA
      AND V.PAGADO = 0;

    RETURN @TOTAL_DEUDA;
END;
```

```sql
use FARMACIA
GO
CREATE FUNCTION FN_TOTAL_DEUDA (@CEDULA VARCHAR(10))
RETURNS DECIMAL(10,2)
AS
BEGIN
    DECLARE @TOTAL_DEUDA DECIMAL(10,2);

    SELECT @TOTAL_DEUDA = ISNULL(SUM(V.TOTAL), 0)
    FROM CLIENTE C
    INNER JOIN VENTA V ON C.IDCLIENTE = V.IDCLIENTE
    WHERE C.CEDULA = @CEDULA
      AND V.PAGADO = 0;

    RETURN @TOTAL_DEUDA;
END;
GO
SELECT dbo.FN_TOTAL_DEUDA('0104050708') AS DEUDA_CLIENTE
GO

SELECT * FROM CLIENTE
```

Results Messages

	DEUDA_CLIENTE
1	200.00

Disparadores

Crear un trigger para poder eliminar en cascada todo lo relacionado a un tipo de medicamento.

```sql
CREATE TRIGGER trg_DeleteIDTIOPO
on TIPO
INSTEAD OF DELETE
AS
BEGIN
  SET NOCOUNT ON;
```

```
        DELETE VENTADETALLE

        FROM VENTADETALLE

        INNER JOIN MEDICAMENTO M ON VENTADETALLE.IDMEDICAMENTO =
M.IDMEDICAMENTO

        INNER JOIN deleted D ON M.IDTIPO =D.IDTIPO




        DELETE M

        FROM MEDICAMENTO M

        INNER JOIN DELETED D ON M.IDTIPO = D.IDTIPO;




        DELETE T

        FROM TIPO T

        INNER JOIN DELETED D ON T.IDTIPO = D.IDTIPO;

END;

SELECT  * FROM VENTADETALLE;

SELECT  * FROM MEDICAMENTO;

SELECT  * FROM TIPO;
DELETE FROM TIPO WHERE IDTIPO = 4
```

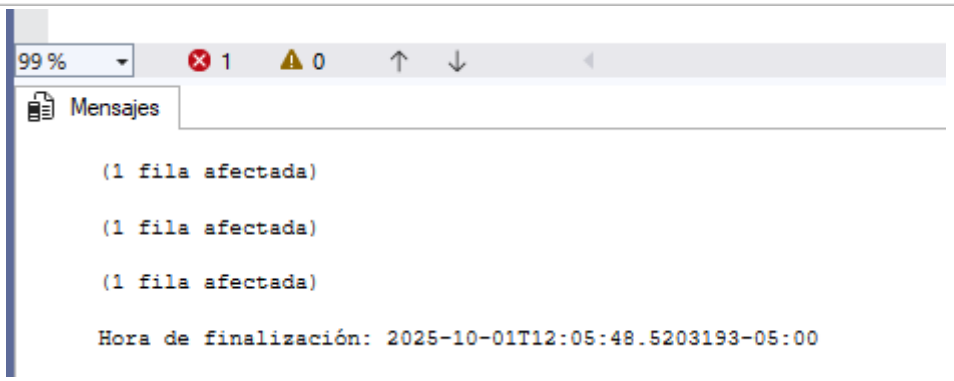
Crear un trigger para registrar en una tabla de auditoría todas las acciones que se realizan en la tabla cliente, medicamento y tipo.

```
CREATE TABLE dbo.Audit_CLIENTE (
    Idcliente INT,
    Codigo VARCHAR(100),
    Nombres VARCHAR(100),
    Cedula VARCHAR(10),
    Telefono VARCHAR(10),
    Fecha_nac DATE,
    Genero INT,
    Fecha DATE,
    Hora TIME,
    Servidor VARCHAR(50),
    Usuario VARCHAR(50),
    Accion CHAR(1)
);
GO
CREATE TABLE dbo.Audit_MEDICAMENTO (
    Idmedicamento INT,
    Idtipo INT,
    Nombre VARCHAR(100),
    Precio DECIMAL(10, 2),
    Generico BIT,
    Stock INT,
    Fecha DATE,
    Hora TIME,
    Servidor VARCHAR(50),
    Usuario VARCHAR(50),
    Accion CHAR(1)
);
GO
CREATE TABLE dbo.Audit_TIPO (
    Idtipo INT,
    Nombre VARCHAR(100),
    Fecha DATE,
    Hora TIME,
    Servidor VARCHAR(50),
    Usuario VARCHAR(50),
    Accion CHAR(1) -- I = Insertar, E = Eliminar, A = Antes de Update, N =
Después de Update
);
GO




go
CREATE OR ALTER TRIGGER tr_audit_cliente
ON CLIENTE
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
    SET NOCOUNT ON;

    -- INSERT
    IF EXISTS (SELECT * FROM inserted) AND NOT EXISTS (SELECT * FROM
deleted)
    BEGIN
        INSERT INTO Audit_CLIENTE
        (Idcliente, Codigo, Nombres, Cedula, Telefono, Fecha_nac, Genero,
Fecha, Hora, Servidor, Usuario, Accion)
        SELECT
            IDCLIENTE, CODIGO, NOMBRES, CEDULA, TELEFONO, FECHA_NAC, GENERO,
```

```
            CAST(GETDATE() AS DATE), CAST(GETDATE() AS TIME),
            HOST_NAME(), SYSTEM_USER, 'I'
        FROM inserted;
    END

    -- DELETE
    IF NOT EXISTS (SELECT * FROM inserted) AND EXISTS (SELECT * FROM
deleted)
    BEGIN
        INSERT INTO Audit_CLIENTE
        (Idcliente, Codigo, Nombres, Cedula, Telefono, Fecha_nac, Genero,
Fecha, Hora, Servidor, Usuario, Accion)
        SELECT
            IDCLIENTE, CODIGO, NOMBRES, CEDULA, TELEFONO, FECHA_NAC, GENERO,
            CAST(GETDATE() AS DATE), CAST(GETDATE() AS TIME),
            HOST_NAME(), SYSTEM_USER, 'E'
        FROM deleted;
    END

    -- UPDATE
    IF EXISTS (SELECT * FROM inserted) AND EXISTS (SELECT * FROM deleted)
    BEGIN
        -- Registro de datos ANTERIORES
        INSERT INTO Audit_CLIENTE
        (Idcliente, Codigo, Nombres, Cedula, Telefono, Fecha_nac, Genero,
Fecha, Hora, Servidor, Usuario, Accion)
        SELECT
            IDCLIENTE, CODIGO, NOMBRES, CEDULA, TELEFONO, FECHA_NAC, GENERO,
            CAST(GETDATE() AS DATE), CAST(GETDATE() AS TIME),
            HOST_NAME(), SYSTEM_USER, 'A'  -- A: Antes de la actualización
        FROM deleted;

        -- Registro de datos NUEVOS
        INSERT INTO Audit_CLIENTE
        (Idcliente, Codigo, Nombres, Cedula, Telefono, Fecha_nac, Genero,
Fecha, Hora, Servidor, Usuario, Accion)
        SELECT
            IDCLIENTE, CODIGO, NOMBRES, CEDULA, TELEFONO, FECHA_NAC, GENERO,
            CAST(GETDATE() AS DATE), CAST(GETDATE() AS TIME),
            HOST_NAME(), SYSTEM_USER, 'N'  -- N: Nuevos después de la
actualización
        FROM inserted;
    END
END
GO
CREATE OR ALTER TRIGGER tr_audit_medicamento
ON MEDICAMENTO
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
    SET NOCOUNT ON;

    -- INSERT
    IF EXISTS (SELECT * FROM inserted) AND NOT EXISTS (SELECT * FROM
deleted)
    BEGIN
        INSERT INTO Audit_MEDICAMENTO
        (Idmedicamento, Idtipo, Nombre, Precio, Generico, Stock, Fecha,
Hora, Servidor, Usuario, Accion)
        SELECT
            IDMEDICAMENTO, IDTIPO, NOMBRE, PRECIO, GENERICO, STOCK,
```

```
            CAST(GETDATE() AS DATE), CAST(GETDATE() AS TIME),
            HOST_NAME(), SYSTEM_USER, 'I'
        FROM inserted;
    END

    -- DELETE
    IF NOT EXISTS (SELECT * FROM inserted) AND EXISTS (SELECT * FROM
deleted)
    BEGIN
        INSERT INTO Audit_MEDICAMENTO
        (Idmedicamento, Idtipo, Nombre, Precio, Generico, Stock, Fecha,
Hora, Servidor, Usuario, Accion)
        SELECT
            IDMEDICAMENTO, IDTIPO, NOMBRE, PRECIO, GENERICO, STOCK,
            CAST(GETDATE() AS DATE), CAST(GETDATE() AS TIME),
            HOST_NAME(), SYSTEM_USER, 'E'
        FROM deleted;
    END

    -- UPDATE
    IF EXISTS (SELECT * FROM inserted) AND EXISTS (SELECT * FROM deleted)
    BEGIN
        -- Antes de la actualización
        INSERT INTO Audit_MEDICAMENTO
        (Idmedicamento, Idtipo, Nombre, Precio, Generico, Stock, Fecha,
Hora, Servidor, Usuario, Accion)
        SELECT
            IDMEDICAMENTO, IDTIPO, NOMBRE, PRECIO, GENERICO, STOCK,
            CAST(GETDATE() AS DATE), CAST(GETDATE() AS TIME),
            HOST_NAME(), SYSTEM_USER, 'A'
        FROM deleted;

        -- Después de la actualización
        INSERT INTO Audit_MEDICAMENTO
        (Idmedicamento, Idtipo, Nombre, Precio, Generico, Stock, Fecha,
Hora, Servidor, Usuario, Accion)
        SELECT
            IDMEDICAMENTO, IDTIPO, NOMBRE, PRECIO, GENERICO, STOCK,
            CAST(GETDATE() AS DATE), CAST(GETDATE() AS TIME),
            HOST_NAME(), SYSTEM_USER, 'N'
        FROM inserted;
    END
END
CREATE OR ALTER TRIGGER tr_audit_tipo
ON TIPO
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
    SET NOCOUNT ON;

    -- INSERT
    IF EXISTS (SELECT * FROM inserted) AND NOT EXISTS (SELECT * FROM
deleted)
    BEGIN
        INSERT INTO Audit_TIPO
        (Idtipo, Nombre, Fecha, Hora, Servidor, Usuario, Accion)
        SELECT
            IDTIPO, NOMBRE,
            CAST(GETDATE() AS DATE), CAST(GETDATE() AS TIME),
            HOST_NAME(), SYSTEM_USER, 'I'
        FROM inserted;
```

```
    END

    -- DELETE
    IF NOT EXISTS (SELECT * FROM inserted) AND EXISTS (SELECT * FROM
deleted)
    BEGIN
        INSERT INTO Audit_TIPO
        (Idtipo, Nombre, Fecha, Hora, Servidor, Usuario, Accion)
        SELECT
            IDTIPO, NOMBRE,
            CAST(GETDATE() AS DATE), CAST(GETDATE() AS TIME),
            HOST_NAME(), SYSTEM_USER, 'E'
        FROM deleted;
    END

    -- UPDATE
    IF EXISTS (SELECT * FROM inserted) AND EXISTS (SELECT * FROM deleted)
    BEGIN
        -- Antes de la actualización
        INSERT INTO Audit_TIPO
        (Idtipo, Nombre, Fecha, Hora, Servidor, Usuario, Accion)
        SELECT
            IDTIPO, NOMBRE,
            CAST(GETDATE() AS DATE), CAST(GETDATE() AS TIME),
            HOST_NAME(), SYSTEM_USER, 'A'
        FROM deleted;

        -- Después de la actualización
        INSERT INTO Audit_TIPO
        (Idtipo, Nombre, Fecha, Hora, Servidor, Usuario, Accion)
        SELECT
            IDTIPO, NOMBRE,
            CAST(GETDATE() AS DATE), CAST(GETDATE() AS TIME),
            HOST_NAME(), SYSTEM_USER, 'N'
        FROM inserted;
    END
END


GO
-- Actualizar un cliente
UPDATE CLIENTE
SET TELEFONO = '0999999999'
WHERE IDCLIENTE = 1;

-- Actualizar un medicamento
UPDATE MEDICAMENTO
SET PRECIO = 3.00
WHERE IDMEDICAMENTO = 1;

-- Actualizar un tipo
UPDATE TIPO
SET NOMBRE = 'Analgésico Común'
WHERE IDTIPO = 1;
```



Archivo sql

Practica 2