****

**Tecnicatura Universitaria en Programación**

**Base de Datos II**

**Equipo # 50**

**Integrantes:**

**Martínez, Nahuel Pablo - 25998**

**Calzolari, Malena Beatriz - 27553**

Explicación del Sistema

Nuestro sistema será una aplicación de escritorio diseñada para la gestión de inventario de una cafetería. La base de datos almacenará información detallada de los artículos (incluyendo su categoría y marca asociadas) y de las distintas sucursales. Los usuarios podrán registrar, modificar y eliminar tanto artículos como sucursales. El sistema permitirá consultar el stock actual en cada una de las sucursales y en la bodega central. Además, facilitará la realización de transacciones de artículos (como transferencias o ajustes), actualizando automáticamente los niveles de stock tanto en la bodega como en las sucursales.

### Funcionalidades principales

* **Gestión de artículos:** Permite registrar, modificar y eliminar artículos del inventario, incluyendo detalles como su categoría y marca asociada.
* **Gestión de sucursales:** Facilita el registro, la modificación y la eliminación de las distintas sucursales de la cafetería.
* **Consulta de stock:** Los usuarios pueden visualizar el stock actual de artículos disponible en cada una de las sucursales y en la bodega central.
* **Registro de transacciones de artículos:** Permite realizar y registrar movimientos de artículos, tales como transferencias entre sucursales/bodega o ajustes de inventario.
* **Actualización automática de stock:** El sistema actualiza de forma automática los niveles de stock en la bodega central y en las sucursales correspondientes tras cada transacción de artículos.

El sistema está construido con una base de datos relacional que almacena toda la información necesaria para gestionar estas funcionalidades.

## Diagrama de Entidad Relación

Una captura de pantalla de un videojuego

El contenido generado por IA puede ser incorrecto.

## Objetos de Base de Datos clave en el sistema

### Vista – Stock por Sucursal

Se utiliza la vista **VW\_StockPorSucursal** que muestra el stock de los articulos por sucursal. La misma cuenta con la siguiente información:

ID de la Sucursal, Nombre de la Sucursal, Código de Articulo, Nombre de Articulo y Cantidad.

USE CafeteriaStockDB

GO

CREATE VIEW VW\_StockPorSucursal AS

SELECT

    s.IdSucursal,

    s.Nombre AS Sucursal,

    a.Codigo AS Código,

    a.Nombre AS Articulo,

    st.Cantidad

FROM Stock st

JOIN Articulo a ON st.IDArticulo = a.IDArticulo

JOIN Sucursal s ON st.IDSucursal = s.IDSucursal

WHERE a.Estado = 1;

### Vista – Stock Total por Articulo

Se utiliza la vista **VW\_StockTotalPorArticulo** que muestra el stock general por artículo. La cual cuenta con la siguiente información:

Código, Nombre del artículo y Stock Total.

USE CafeteriaStockDB

GO

CREATE VIEW VW\_StockTotalPorArticulo AS

SELECT

    a.Codigo AS Código,

    a.Nombre AS Articulo,

    SUM(st.Cantidad) AS StockTotal

FROM Articulo a

JOIN Stock st ON a.IDArticulo = st.IDArticulo

WHERE a.Estado = 1

GROUP BY a.Codigo, a.Nombre;

### Procedimiento Almacenado – Listar Artículos

Este procedimiento almacenado llamado **SP\_ListarArticulos** lista todos los artículos de la cafetería, incluyendo detalles de su marca y categoría, mostrando por defecto solo los activos y mostrando cualquier error que ocurra durante la consulta.

USE CafeteriaStockDB;

GO

CREATE PROCEDURE SP\_ListarArticulos

AS

BEGIN

    BEGIN TRY

        SELECT

            A.IdArticulo,

            A.Codigo,

            A.Nombre,

            A.PrecioUnidad,

            A.Estado AS EstadoA,

            M.IdMarca,

            M.Descripcion AS DescripcionMarca,

            C.IdCategoria,

            C.Descripcion AS DescripcionCategoria

        FROM

            Articulo AS A

        INNER JOIN

            Marca AS M ON A.IdMarca = M.IdMarca

        INNER JOIN

            Categoria AS C ON A.IdCategoria = C.IdCategoria

        WHERE A.Estado = 1;

    END TRY

    BEGIN CATCH

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

        SELECT

            @ErrorMessage = ERROR\_MESSAGE(),

            @ErrorSeverity = ERROR\_SEVERITY(),

            @ErrorState = ERROR\_STATE();

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

        PRINT 'Error al listar los artículos.';

    END CATCH;

END;

GO

### Procedimiento Almacenado – Insertar Artículo

Este procedimiento almacenado llamado **SP\_InsertarArticulo** inserta un nuevo artículo en la base de datos de la cafetería, registrando su código, nombre, marca, categoría, precio y estado, y cuenta con manejo de errores. En el caso de que ocurra algún error se realiza un ROLLBACK y muestra la información del error.

USE CafeteriaStockDB;

GO

CREATE PROCEDURE SP\_InsertarArticulo

    @Codigo VARCHAR(50),

    @Nombre VARCHAR(150),

    @IdMarca INT,

    @IdCategoria INT,

    @PrecioUnidad DECIMAL(10, 2),

    @Estado BIT = 1

AS

BEGIN

    BEGIN TRY

        BEGIN TRANSACTION;

        INSERT INTO Articulo (Codigo, Nombre, IdMarca, IdCategoria, PrecioUnidad, Estado)

        VALUES (@Codigo, @Nombre, @IdMarca, @IdCategoria, @PrecioUnidad, @Estado);

        COMMIT TRANSACTION;

        SELECT

            A.IdArticulo,

            A.Codigo,

            A.Nombre,

            A.PrecioUnidad,

            A.Estado,

            M.Descripcion AS Marca,

            C.Descripcion AS Categoria

        FROM

            Articulo AS A

        INNER JOIN

            Marca AS M ON A.IdMarca = M.IdMarca

        INNER JOIN

            Categoria AS C ON A.IdCategoria = C.IdCategoria

        WHERE

            A.Codigo = @Codigo;

        PRINT 'Artículo insertado exitosamente.';

    END TRY

    BEGIN CATCH

        -- Si ocurre algún error, se hace ROLLBACK de la transacción

        IF @@TRANCOUNT > 0

            ROLLBACK TRANSACTION;

        -- Muestra información del error

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

        SELECT

            @ErrorMessage = ERROR\_MESSAGE(),

            @ErrorSeverity = ERROR\_SEVERITY(),

            @ErrorState = ERROR\_STATE();

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

        PRINT 'Error al insertar el artículo.';

    END CATCH;

END;

GO

### Procedimiento Almacenado – Actualizar o Inserta Stock

Este procedimiento almacenado llamado **SP\_ActualizarOInsertarStock** permite ingresarle una cantidad de stock a un artículo. Si el articulo existe en la tabla Stock, se realiza un UPDATE sumándole las cantidades ingresadas por el usuario. Si el articulo no existe en la tabla Stock, se realiza un INSERT creando un nuevo registro con la cantidad ingresada por el usuario. En el caso de que ocurra algún error muestra la información del mismo.

USE CafeteriaStockDB;

GO

CREATE PROCEDURE SP\_ActualizarOInsertarStock

    @IdArticulo INT,

    @IdSucursal INT,

    @Cantidad INT

AS

BEGIN

    BEGIN TRY

        MERGE Stock AS Target

        USING (SELECT @IdArticulo AS IdArticulo, @IdSucursal AS IdSucursal, @Cantidad AS Cantidad)

AS Source

        ON Target.IdArticulo = Source.IdArticulo AND Target.IdSucursal = Source.IdSucursal

        WHEN MATCHED THEN

            -- Si el registro existe, actualiza la cantidad sumando el valor

            UPDATE SET Target.Cantidad = Target.Cantidad + Source.Cantidad

        WHEN NOT MATCHED BY TARGET THEN

            -- Si el registro no existe, inserta uno nuevo

            INSERT (IdArticulo, IdSucursal, Cantidad)

            VALUES (Source.IdArticulo, Source.IdSucursal, Source.Cantidad);

        PRINT 'Operación de stock (actualizar o insertar) completada exitosamente.';

    END TRY

    BEGIN CATCH

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

        SELECT

            @ErrorMessage = ERROR\_MESSAGE(),

            @ErrorSeverity = ERROR\_SEVERITY(),

            @ErrorState = ERROR\_STATE();

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

        PRINT 'Error al realizar la operación de stock.';

    END CATCH;

END;

GO

### Procedimiento Almacenado – Reducir Stock

Este procedimiento almacenado llamado **SP\_ReducirStock** permite restar una cantidad de stock a una Sucursal. Valida que la cantidad a restar no sea mayor que el stock actual. En el caso de que la cantidad ingresada sea mayor que el stock actual arroja un mensaje de error y no se resta el stock.

USE CafeteriaStockDB;

GO

CREATE PROCEDURE SP\_ReducirStock

    @IdArticulo INT,

    @IdSucursal INT,

    @Cantidad INT

AS

BEGIN

    BEGIN TRY

        DECLARE @StockActual INT;

        SELECT @StockActual = Cantidad

        FROM Stock

        WHERE IdArticulo = @IdArticulo AND IdSucursal = @IdSucursal;

        -- Validar si el registro de stock existe

        IF @StockActual IS NULL

        BEGIN

            RAISERROR('No existe stock para el artículo y sucursal especificados.', 16, 1);

            RETURN;

        END

        -- Validar si la cantidad a reducir es mayor que el stock actual

        IF @Cantidad > @StockActual

        BEGIN

            RAISERROR('La cantidad a dar de baja (%d) es mayor que el stock actual (%d).', 16, 1,

@Cantidad, @StockActual);

            RETURN;

        END

        -- Restar del stock

        UPDATE Stock

        SET Cantidad = Cantidad - @Cantidad

        WHERE IdArticulo = @IdArticulo AND IdSucursal = @IdSucursal;

        PRINT 'Stock reducido exitosamente.';

    END TRY

    BEGIN CATCH

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

        SELECT

            @ErrorMessage = ERROR\_MESSAGE(),

            @ErrorSeverity = ERROR\_SEVERITY(),

            @ErrorState = ERROR\_STATE();

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

    END CATCH;

END;

### Procedimiento Almacenado – Mover Stock entre Sucursales

Este procedimiento almacenado llamado **SP\_MoverStockEntreSucursales** permite mover una cantidad de stock desde una sucursal a otra.

Valida que las sucursales de “origen” y “destino” no sean la misma y también que la cantidad a mover sea positiva, en el caso de que no se cumplan las condiciones, arroja un mensaje de error y cancela el movimiento.

Verifica si la cantidad a mover es mayor que el stock actual en origen. En el caso de que no se cumpla la condición, arroja un mensaje de error y realiza un ROLLBACK TRANSACTION.

En el caso de que las condiciones se cumplan resta la cantidad ingresada en la sucursal de “origen”.

Si ya existe un registro con el mismo IdArticulo para la sucursal de “destino” en la tabla Stock, realiza un UPDATE y aumenta el stock de ese articulo en esa sucursal. En el caso de que no exista registro del articulo en la sucursal de “destino”, realiza un INSERT con las cantidades ingresadas.

USE CafeteriaStockDB;

GO

CREATE PROCEDURE SP\_MoverStockEntreSucursales

    @IdArticulo INT,

    @IdSucursalOrigen INT,

    @IdSucursalDestino INT,

    @CantidadAMover INT

AS

BEGIN

    BEGIN TRY

        -- Validar que la cantidad a mover sea positiva

        IF @CantidadAMover <= 0

        BEGIN

            RAISERROR('La cantidad a mover debe ser un número positivo.', 16, 1);

            RETURN;

        END

        -- Validar que las sucursales de origen y destino no sean la misma

        IF @IdSucursalOrigen = @IdSucursalDestino

        BEGIN

            RAISERROR('La sucursal de origen y la de destino no pueden ser la misma.', 16, 1);

            RETURN;

        END

        BEGIN TRANSACTION;

        DECLARE @StockActualOrigen INT;

        -- Obtener el stock actual del artículo en la sucursal de origen

        SELECT @StockActualOrigen = Cantidad

        FROM Stock

        WHERE IdArticulo = @IdArticulo AND IdSucursal = @IdSucursalOrigen;

        -- Validar si la sucursal de origen tiene stock para el artículo

        IF @StockActualOrigen IS NULL

        BEGIN

            RAISERROR('No existe stock para el artículo seleccionado en la sucursal de origen.', 16, 1);

            ROLLBACK TRANSACTION;

            RETURN;

        END

        -- Validar si la cantidad a mover es mayor que el stock actual en origen

        IF @CantidadAMover > @StockActualOrigen

        BEGIN

            RAISERROR('La sucursal de origen no tiene stock suficiente (%d unidades) para mover. Stock

actual: %d.', 16, 1, @CantidadAMover, @StockActualOrigen);

            ROLLBACK TRANSACTION;

            RETURN;

        END

        -- 1. Reducir el stock en la sucursal de origen

        UPDATE Stock

        SET Cantidad = Cantidad - @CantidadAMover

        WHERE IdArticulo = @IdArticulo AND IdSucursal = @IdSucursalOrigen;

        -- 2. Aumentar el stock en la sucursal de destino (o insertarlo si no existe)

        MERGE Stock AS Target

        USING (SELECT @IdArticulo AS IdArticulo, @IdSucursalDestino AS IdSucursal, @CantidadAMover AS

Cantidad) AS Source

        ON Target.IdArticulo = Source.IdArticulo AND Target.IdSucursal = Source.IdSucursal

        WHEN MATCHED THEN

            -- Si el registro existe, actualiza la cantidad sumando el valor

            UPDATE SET Target.Cantidad = Target.Cantidad + Source.Cantidad

        WHEN NOT MATCHED BY TARGET THEN

            -- Si el registro no existe, inserta uno nuevo

            INSERT (IdArticulo, IdSucursal, Cantidad)

            VALUES (Source.IdArticulo, Source.IdSucursal, Source.Cantidad);

        COMMIT TRANSACTION;

        PRINT 'Stock movido exitosamente';

    END TRY

    BEGIN CATCH

        IF @@TRANCOUNT > 0

            ROLLBACK TRANSACTION;

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

        SELECT

            @ErrorMessage = ERROR\_MESSAGE(),

            @ErrorSeverity = ERROR\_SEVERITY(),

            @ErrorState = ERROR\_STATE();

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

    END CATCH;

END;

### Trigger – Eliminar Articulo

Este trigger **TR\_EliminarArticulo** se ejecuta en caso de que se quiera realizar un DELETE en la tabla Articulo (Instead of DELETE). Verifica si el artículo a eliminar tiene stock activo. Si tiene stock activo, impide la eliminación y emite un mensaje de error. Si no tiene stock activo (Cantidad = 0 o no hay registros) elimina los registros de stock relacionados y luego elimina el artículo.

USE CafeteriaStockDB;

GO

CREATE TRIGGER TR\_EliminarArticulo

ON Articulo

INSTEAD OF DELETE

AS

BEGIN

    DECLARE @IdArticuloAEliminar INT;

    DECLARE @TieneStockActivo BIT;

    SELECT @IdArticuloAEliminar = IdArticulo FROM DELETED;

    BEGIN TRY

        SET @TieneStockActivo = 0;

        --Verificar si el artículo tiene stock activo (Cantidad > 0)

        IF EXISTS (SELECT 1 FROM Stock WHERE IdArticulo = @IdArticuloAEliminar AND Cantidad > 0)

        BEGIN

            SET @TieneStockActivo = 1;

        END

        IF @TieneStockActivo = 1

        BEGIN

            -- Si el artículo tiene stock activo, impedir la eliminación

            RAISERROR('No se puede eliminar el artículo ya que el mismo posee stock activo.', 16, 1);

        END

        ELSE

        BEGIN

            -- Si el artículo NO tiene stock activo (Cant = 0 o no hay registros),

            -- procedemos a eliminar cualquier registro de ese artículo en la tabla Stock.

            DELETE FROM Stock

            WHERE IdArticulo = @IdArticuloAEliminar;

            --Si ya no hay referencias en Stock, podemos eliminar el artículo de la tabla Articulo.

            DELETE FROM Articulo

            WHERE IdArticulo = @IdArticuloAEliminar;

            PRINT 'Artículo eliminado exitosamente ya que no poseía stock activo.';

        END

    END TRY

    BEGIN CATCH

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

        SELECT

            @ErrorMessage = ERROR\_MESSAGE(),

            @ErrorSeverity = ERROR\_SEVERITY(),

            @ErrorState = ERROR\_STATE();

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

    END CATCH;

END;

### Trigger – Eliminar Sucursal

Este trigger **TR\_EliminarSucursal** se ejecuta en caso de que se quiera realizar un DELETE en la tabla Sucursal (Instead of DELETE). Redirige todo el stock de la sucursal que se intenta eliminar al depósito (IDSucursal = 1), luego ELIMINA ese stock de la sucursal original, y finalmente elimina la sucursal.

USE CafeteriaStockDB;

GO

CREATE TRIGGER TR\_EliminarSucursal

ON Sucursal

INSTEAD OF DELETE

AS

BEGIN

    -- ID del DEPOSITO

    DECLARE @DepositoId INT = 1;

    DECLARE @IdSucursalAEliminar INT = (SELECT IdSucursal FROM DELETED);

    BEGIN TRY

        -- Validar que la sucursal a eliminar no sea el propio depósito

        IF @IdSucursalAEliminar = @DepositoId

        BEGIN

            RAISERROR('No se puede eliminar la sucursal de depósito (ID = 1).', 16, 1);

            RETURN;

        END

        --Mover el stock de la sucursal que se va a eliminar al depósito

        MERGE Stock AS Target

        USING (

            SELECT S.IdArticulo, S.Cantidad

            FROM Stock AS S

            WHERE S.IdSucursal = @IdSucursalAEliminar

        ) AS Source (IdArticulo, Cantidad)

        ON Target.IdArticulo = Source.IdArticulo AND Target.IdSucursal = @DepositoId

        WHEN MATCHED THEN

            UPDATE SET Target.Cantidad = Target.Cantidad + Source.Cantidad

        WHEN NOT MATCHED BY TARGET THEN

            INSERT (IdArticulo, IdSucursal, Cantidad)

            VALUES (Source.IdArticulo, @DepositoId, Source.Cantidad);

        --Eliminar el stock de la sucursal que se va a eliminar

        DELETE FROM Stock

        WHERE IdSucursal = @IdSucursalAEliminar;

        --Después de transferir y limpiar el stock, eliminar la sucursal

        DELETE FROM Sucursal

        WHERE IdSucursal = @IdSucursalAEliminar;

        PRINT 'Operación de eliminación de sucursal completada. Stock transferido al depósito.';

    END TRY

    BEGIN CATCH

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

        SELECT

            @ErrorMessage = ERROR\_MESSAGE(),

            @ErrorSeverity = ERROR\_SEVERITY(),

            @ErrorState = ERROR\_STATE();

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

    END CATCH;

END;

## Links a los recursos

[**github.com/nahuelpmartinez/TPI-BD2**](https://github.com/nahuelpmartinez/TPI-BD2)

**Script de creación de base de datos con datos**

CREATE DATABASE CafeteriaStockDB;

GO

USE CafeteriaStockDB;

GO

CREATE TABLE Categoria (

    IdCategoria INT PRIMARY KEY IDENTITY(1,1),

    Descripcion VARCHAR(255) NOT NULL,

    Estado BIT NOT NULL DEFAULT 1

);

GO

CREATE TABLE Marca (

    IdMarca INT PRIMARY KEY IDENTITY(1,1),

    Descripcion VARCHAR(255) NOT NULL,

    Estado BIT NOT NULL DEFAULT 1

);

GO

CREATE TABLE Sucursal (

    IdSucursal INT PRIMARY KEY IDENTITY(1,1),

    Nombre VARCHAR(100) NOT NULL,

    Direccion VARCHAR(255),

    FechaCreacion DATETIME DEFAULT GETDATE(),

    Estado BIT NOT NULL DEFAULT 1

);

GO

CREATE TABLE Articulo (

    IdArticulo INT PRIMARY KEY IDENTITY(1,1),

    Nombre VARCHAR(150) NOT NULL,

    Codigo VARCHAR(50) NOT NULL UNIQUE,

    PrecioUnidad DECIMAL(10, 2) NOT NULL,

    IdMarca INT NOT NULL,

    IdCategoria INT NOT NULL,

    Estado BIT NOT NULL DEFAULT 1,

    FOREIGN KEY (IdMarca) REFERENCES Marca(IdMarca),

    FOREIGN KEY (IdCategoria) REFERENCES Categoria(IdCategoria)

);

GO

CREATE TABLE Stock (

    IdArticulo INT NOT NULL,

    IdSucursal INT NOT NULL,

    Cantidad INT NOT NULL,

    PRIMARY KEY (IdArticulo, IdSucursal),

    FOREIGN KEY (IdArticulo) REFERENCES Articulo(IdArticulo),

    FOREIGN KEY (IdSucursal) REFERENCES Sucursal(IdSucursal),

    CONSTRAINT CHK\_Stock\_Cantidad CHECK (Cantidad >= 0)

);

GO

-- 1. Insertar datos en la tabla Categoria

PRINT 'Insertando datos en Categoria...';

INSERT INTO Categoria (Descripcion, Estado) VALUES

('Bebidas Calientes', 1),

('Bebidas Frias', 1),

('Panaderia', 1),

('Pasteleria', 1),

('Insumos Generales', 1),

('Snacks', 1);

GO

-- 2. Insertar datos en la tabla Marca

PRINT 'Insertando datos en Marca...';

INSERT INTO Marca (Descripcion, Estado) VALUES

('Cafe Supreme', 1),

('Lacteos Frescos', 1),

('Dulce Tentacion', 1),

('Pan Artesanal', 1),

('Global Supplies', 1),

('Agua Pura', 1),

('Snack Delicioso', 1);

GO

-- 3. Insertar datos en la tabla Sucursal

PRINT 'Insertando datos en Sucursal...';

INSERT INTO Sucursal (Nombre, Direccion, Estado) VALUES

('Deposito', 'Av. Siempre Viva 742, Springfield', 1),

('Sucursal Norte', 'Calle Falsa 123, Sector Industrial', 1),

('Sucursal Sur', 'Blvd. de los Sueños Rotos 45, Zona Residencial', 1);

GO

-- 4. Insertar datos en la tabla Articulo

PRINT 'Insertando datos en Articulo...';

INSERT INTO Articulo (Nombre, Codigo, PrecioUnidad, IdMarca, IdCategoria, Estado) VALUES

-- Bebidas Calientes

('Cafe Americano', 'CAFE001', 2.50, (SELECT IdMarca FROM Marca WHERE Descripcion = 'Cafe Supreme'), (SELECT IdCategoria FROM Categoria WHERE Descripcion = 'Bebidas Calientes'), 1),

('Espresso', 'ESP001', 2.00, (SELECT IdMarca FROM Marca WHERE Descripcion = 'Cafe Supreme'), (SELECT IdCategoria FROM Categoria WHERE Descripcion = 'Bebidas Calientes'), 1),

('Latte', 'LATTE001', 3.00, (SELECT IdMarca FROM Marca WHERE Descripcion = 'Cafe Supreme'), (SELECT IdCategoria FROM Categoria WHERE Descripcion = 'Bebidas Calientes'), 1),

('Te Negro', 'TEN001', 2.20, (SELECT IdMarca FROM Marca WHERE Descripcion = 'Global Supplies'), (SELECT IdCategoria FROM Categoria WHERE Descripcion = 'Bebidas Calientes'), 1),

-- Bebidas Frias

('Jugo de Naranja', 'JUGON001', 3.50, (SELECT IdMarca FROM Marca WHERE Descripcion = 'Lacteos Frescos'), (SELECT IdCategoria FROM Categoria WHERE Descripcion = 'Bebidas Frias'), 1),

('Agua Embotellada 500ml', 'AGUA001', 1.50, (SELECT IdMarca FROM Marca WHERE Descripcion = 'Agua Pura'), (SELECT IdCategoria FROM Categoria WHERE Descripcion = 'Bebidas Frias'), 1),

('Gaseosa Cola', 'GASCOLA001', 2.80, (SELECT IdMarca FROM Marca WHERE Descripcion = 'Global Supplies'), (SELECT IdCategoria FROM Categoria WHERE Descripcion = 'Bebidas Frias'), 1),

-- Panaderia

('Croissant Simple', 'CROIS001', 1.80, (SELECT IdMarca FROM Marca WHERE Descripcion = 'Pan Artesanal'), (SELECT IdCategoria FROM Categoria WHERE Descripcion = 'Panaderia'), 1),

('Medialuna de Manteca', 'MEDIAL001', 1.60, (SELECT IdMarca FROM Marca WHERE Descripcion = 'Pan Artesanal'), (SELECT IdCategoria FROM Categoria WHERE Descripcion = 'Panaderia'), 1),

('Pan Integral Rebanado', 'PANINT001', 4.00, (SELECT IdMarca FROM Marca WHERE Descripcion = 'Pan Artesanal'), (SELECT IdCategoria FROM Categoria WHERE Descripcion = 'Panaderia'), 1),

-- Pasteleria

('Muffin de Chocolate', 'MUFFCHOC001', 2.90, (SELECT IdMarca FROM Marca WHERE Descripcion = 'Dulce Tentacion'), (SELECT IdCategoria FROM Categoria WHERE Descripcion = 'Pasteleria'), 1),

('Cheesecake porcion', 'CHEESECAKE001', 4.50, (SELECT IdMarca FROM Marca WHERE Descripcion = 'Dulce Tentacion'), (SELECT IdCategoria FROM Categoria WHERE Descripcion = 'Pasteleria'), 1),

-- Insumos Generales

('Azucar (kg)', 'AZUCAR001', 1.20, (SELECT IdMarca FROM Marca WHERE Descripcion = 'Global Supplies'), (SELECT IdCategoria FROM Categoria WHERE Descripcion = 'Insumos Generales'), 1),

('Leche Entera (Litro)', 'LECHE001', 1.90, (SELECT IdMarca FROM Marca WHERE Marca.Descripcion = 'Lacteos Frescos'), (SELECT IdCategoria FROM Categoria WHERE Descripcion = 'Insumos Generales'), 1),

-- Snacks

('Barra de Cereal', 'BARRAC001', 1.75, (SELECT IdMarca FROM Marca WHERE Descripcion = 'Snack Delicioso'), (SELECT IdCategoria FROM Categoria WHERE Descripcion = 'Snacks'), 1),

('Papas Fritas 50g', 'PAPAS001', 2.10, (SELECT IdMarca FROM Marca WHERE Descripcion = 'Snack Delicioso'), (SELECT IdCategoria FROM Categoria WHERE Descripcion = 'Snacks'), 1);

GO

-- 5. Insertar datos en la tabla Stock

PRINT 'Insertando datos en Stock...';

-- Deposito

INSERT INTO Stock (IdArticulo, IdSucursal, Cantidad) VALUES

((SELECT IdArticulo FROM Articulo WHERE Codigo = 'CAFE001'), (SELECT IdSucursal FROM Sucursal WHERE Nombre = 'Deposito'), 150),

((SELECT IdArticulo FROM Articulo WHERE Codigo = 'ESP001'), (SELECT IdSucursal FROM Sucursal WHERE Nombre = 'Deposito'), 100),

((SELECT IdArticulo FROM Articulo WHERE Codigo = 'LATTE001'), (SELECT IdSucursal FROM Sucursal WHERE Nombre = 'Deposito'), 120),

((SELECT IdArticulo FROM Articulo WHERE Codigo = 'TEN001'), (SELECT IdSucursal FROM Sucursal WHERE Nombre = 'Deposito'), 80),

((SELECT IdArticulo FROM Articulo WHERE Codigo = 'JUGON001'), (SELECT IdSucursal FROM Sucursal WHERE Nombre = 'Deposito'), 70),

((SELECT IdArticulo FROM Articulo WHERE Codigo = 'AGUA001'), (SELECT IdSucursal FROM Sucursal WHERE Nombre = 'Deposito'), 200),

((SELECT IdArticulo FROM Articulo WHERE Codigo = 'CROIS001'), (SELECT IdSucursal FROM Sucursal WHERE Nombre = 'Deposito'), 90),

((SELECT IdArticulo FROM Articulo WHERE Codigo = 'MUFFCHOC001'), (SELECT IdSucursal FROM Sucursal WHERE Nombre = 'Deposito'), 60),

((SELECT IdArticulo FROM Articulo WHERE Codigo = 'AZUCAR001'), (SELECT IdSucursal FROM Sucursal WHERE Nombre = 'Deposito'), 50),

((SELECT IdArticulo FROM Articulo WHERE Codigo = 'LECHE001'), (SELECT IdSucursal FROM Sucursal WHERE Nombre = 'Deposito'), 40),

((SELECT IdArticulo FROM Articulo WHERE Codigo = 'BARRAC001'), (SELECT IdSucursal FROM Sucursal WHERE Nombre = 'Deposito'), 110);

-- Sucursal Norte

INSERT INTO Stock (IdArticulo, IdSucursal, Cantidad) VALUES

((SELECT IdArticulo FROM Articulo WHERE Codigo = 'CAFE001'), (SELECT IdSucursal FROM Sucursal WHERE Nombre = 'Sucursal Norte'), 100),

((SELECT IdArticulo FROM Articulo WHERE Codigo = 'LATTE001'), (SELECT IdSucursal FROM Sucursal WHERE Nombre = 'Sucursal Norte'), 80),

((SELECT IdArticulo FROM Articulo WHERE Codigo = 'JUGON001'), (SELECT IdSucursal FROM Sucursal WHERE Nombre = 'Sucursal Norte'), 50),

((SELECT IdArticulo FROM Articulo WHERE Codigo = 'AGUA001'), (SELECT IdSucursal FROM Sucursal WHERE Nombre = 'Sucursal Norte'), 150),

((SELECT IdArticulo FROM Articulo WHERE Codigo = 'MEDIAL001'), (SELECT IdSucursal FROM Sucursal WHERE Nombre = 'Sucursal Norte'), 75),

((SELECT IdArticulo FROM Articulo WHERE Codigo = 'CHEESECAKE001'), (SELECT IdSucursal FROM Sucursal WHERE Nombre = 'Sucursal Norte'), 30),

((SELECT IdArticulo FROM Articulo WHERE Codigo = 'AZUCAR001'), (SELECT IdSucursal FROM Sucursal WHERE Nombre = 'Sucursal Norte'), 30),

((SELECT IdArticulo FROM Articulo WHERE Codigo = 'LECHE001'), (SELECT IdSucursal FROM Sucursal WHERE Nombre = 'Sucursal Norte'), 25),

((SELECT IdArticulo FROM Articulo WHERE Codigo = 'PAPAS001'), (SELECT IdSucursal FROM Sucursal WHERE Nombre = 'Sucursal Norte'), 85);

-- Sucursal Sur

INSERT INTO Stock (IdArticulo, IdSucursal, Cantidad) VALUES

((SELECT IdArticulo FROM Articulo WHERE Codigo = 'CAFE001'), (SELECT IdSucursal FROM Sucursal WHERE Nombre = 'Sucursal Sur'), 90),

((SELECT IdArticulo FROM Articulo WHERE Codigo = 'ESP001'), (SELECT IdSucursal FROM Sucursal WHERE Nombre = 'Sucursal Sur'), 60),

((SELECT IdArticulo FROM Articulo WHERE Codigo = 'TEN001'), (SELECT IdSucursal FROM Sucursal WHERE Nombre = 'Sucursal Sur'), 70),

((SELECT IdArticulo FROM Articulo WHERE Codigo = 'GASCOLA001'), (SELECT IdSucursal FROM Sucursal WHERE Nombre = 'Sucursal Sur'), 100),

((SELECT IdArticulo FROM Articulo WHERE Codigo = 'PANINT001'), (SELECT IdSucursal FROM Sucursal WHERE Nombre = 'Sucursal Sur'), 40),

((SELECT IdArticulo FROM Articulo WHERE Codigo = 'MUFFCHOC001'), (SELECT IdSucursal FROM Sucursal WHERE Nombre = 'Sucursal Sur'), 50),

((SELECT IdArticulo FROM Articulo WHERE Codigo = 'LECHE001'), (SELECT IdSucursal FROM Sucursal WHERE Nombre = 'Sucursal Sur'), 35);

GO

PRINT 'Datos de ejemplo insertados exitosamente en CafeteriaStockDB.';

**Video demo del sistema (hasta 25 minutos)**