Se diseñó la base de datos `BD\_AMCDrinks` para gestionar información relacionada con precios, compras, inventarios y ventas de bebidas, garantizando la integridad de los datos mediante claves foráneas.

Se utilizó `BULK INSERT` para cargar eficientemente grandes volúmenes de datos desde archivos CSV, y se implementó un procedimiento almacenado para insertar compras, evitando duplicados y simplificando el proceso. Este enfoque permite una gestión organizada y efectiva de la información en el negocio.

Para crear una \*\*automatización de carga de datos\*\* en una base de datos con las tablas mencionadas, puedes usar diferentes enfoques, como scripts SQL de inserción automatizada o procesos ETL (Extract, Transform, Load) más avanzados si los datos provienen de diferentes fuentes. A continuación te muestro cómo podrías hacer un proceso básico de automatización para cargar datos en estas tablas:

### Pasos para la automatización de la carga de datos:

#### 1. \*\*Creación de Scripts de Inserción\*\*

- Para insertar datos en cada tabla, puedes crear scripts SQL que inserten datos desde archivos CSV o cualquier otra fuente externa. SQL Server te permite hacer esto de forma automática usando la función `BULK INSERT` o a través de un script de importación por lotes (`INSERT INTO`).

#### 2. \*\*Uso de un Archivo CSV para Automatizar la Carga\*\*

Si los datos están en archivos CSV, puedes utilizar el siguiente enfoque para insertar datos en las tablas de la base de datos:

- Primero, asegúrate de que los datos estén preparados y formateados correctamente en archivos CSV.

- Usa la siguiente estructura de código para realizar la inserción en las tablas desde un archivo CSV.

```sql

BULK INSERT BD\_AMCDrinks.dbo.Precios

FROM 'C:\ruta\del\archivo\precios.csv'

WITH (

FIELDTERMINATOR = ',', -- Delimitador de campos

ROWTERMINATOR = '\n', -- Delimitador de filas

FIRSTROW = 2, -- Ignorar la fila de encabezado

TABLOCK

);

```

Este comando cargará datos directamente en la tabla \*\*Precios\*\* desde un archivo CSV.

#### 3. \*\*Validación y Carga Incremental\*\*

Puedes configurar la automatización para hacer validaciones antes de cargar los datos, como evitar duplicados o verificar si los datos ya existen en las tablas.

- Utiliza consultas `IF EXISTS` para comprobar si los datos ya están en la base de datos antes de insertarlos. Ejemplo para evitar duplicados en la tabla `Compras`:

```sql

IF NOT EXISTS (SELECT 1 FROM Compras WHERE PONumber = @PONumber)

BEGIN

INSERT INTO Compras (InventoryId, Store, Brand, ...)

VALUES (@InventoryId, @Store, @Brand, ...);

END;

```

#### 4. \*\*Scripts de Inserción en Cadena\*\*

Crea un script que se ejecute en secuencia para cargar los datos en todas las tablas relacionadas, respetando las restricciones de claves foráneas.

```sql

-- Insertar en la tabla de Precios

INSERT INTO Precios (Brand, Description, Price, Size, Volume, Classification, PurchasePrice, VendorNumber, VendorName)

VALUES (1, 'Coca Cola', 12.50, '500ml', 500, 1, 10.50, 101, 'Proveedor 1');

-- Insertar en la tabla de OrdenesCompra

INSERT INTO OrdenesCompra (VendorNumber, VendorName, InvoiceDate, PONumber, PODate, PayDate, Quantity, Dollars, Freight, Approval)

VALUES (101, 'Proveedor 1', '2023-10-01', 1001, '2023-10-01', NULL, 100, 1250.00, 50.00, 'Approved');

```

#### 5. \*\*Automatización Completa con Stored Procedures\*\*

Puedes encapsular los procesos de carga de datos en \*\*procedimientos almacenados\*\* que se ejecuten automáticamente a intervalos regulares o bajo demanda. Los procedimientos almacenados (Stored Procedures) en SQL Server son útiles para organizar la carga de datos de manera secuencial, gestionar errores, y garantizar la integridad de los datos.

Ejemplo de un \*\*Stored Procedure\*\* para la carga de datos:

```sql

CREATE PROCEDURE InsertDataIntoCompras

@InventoryId VARCHAR(50),

@Store INT,

@Brand INT,

@Description VARCHAR(200),

@Size VARCHAR(200),

@VendorNumber INT,

@VendorName VARCHAR(200),

@PONumber INT,

@PODate DATE,

@ReceivingDate DATE,

@InvoiceDate DATE,

@PayDate DATE,

@PurchasePrice DECIMAL(10,2),

@Quantity INT,

@Dollars DECIMAL(10,2)

AS

BEGIN

-- Verificar si ya existe la compra

IF NOT EXISTS (SELECT 1 FROM Compras WHERE InventoryId = @InventoryId AND PONumber = @PONumber)

BEGIN

-- Insertar la nueva compra

INSERT INTO Compras (InventoryId, Store, Brand, Description, Size, VendorNumber, VendorName, PONumber, PODate, ReceivingDate, InvoiceDate, PayDate, PurchasePrice, Quantity, Dollars)

VALUES (@InventoryId, @Store, @Brand, @Description, @Size, @VendorNumber, @VendorName, @PONumber, @PODate, @ReceivingDate, @InvoiceDate, @PayDate, @PurchasePrice, @Quantity, @Dollars);

END;

END;

```

Puedes programar la ejecución de este procedimiento almacenado en intervalos de tiempo específicos usando el \*\*SQL Server Agent\*\* o algún mecanismo de automatización.

#### 6. \*\*Manejo de Dependencias de Claves Foráneas\*\*

- Dado que hay claves foráneas entre las tablas (por ejemplo, entre \*\*Compras\*\* y \*\*OrdenesCompra\*\*), asegúrate de que los datos se carguen en el orden correcto, primero las tablas que no tienen claves foráneas y luego las que dependen de ellas.

#### 7. \*\*Automatización con Herramientas ETL\*\*

Para procesos más avanzados de carga y transformación de datos, puedes utilizar herramientas como \*\*SQL Server Integration Services (SSIS)\*\*, que permiten crear paquetes ETL para cargar datos desde múltiples fuentes, transformarlos y cargarlos en la base de datos de manera automatizada.

### Conclusión:

Este proceso de automatización de la carga de datos incluye tanto métodos básicos usando `BULK INSERT` como procedimientos almacenados para garantizar la correcta inserción y validación de los datos. Dependiendo del origen de los datos y la complejidad del sistema, puedes optar por soluciones más avanzadas como \*\*ETL\*\* para una automatización robusta y escalable.