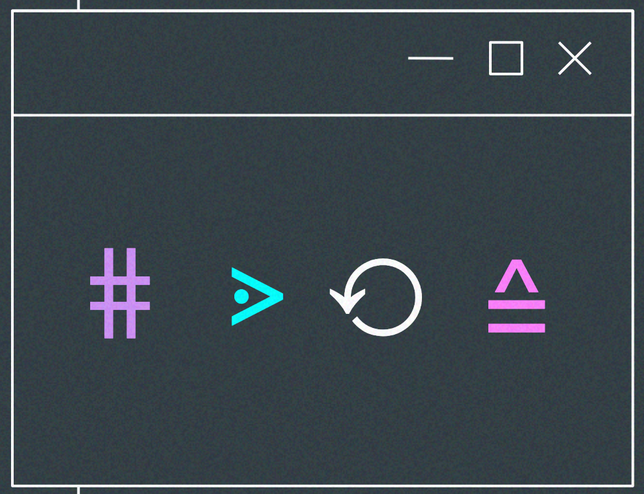
Injeção de SQL: Riscos e Medidas de Segurança



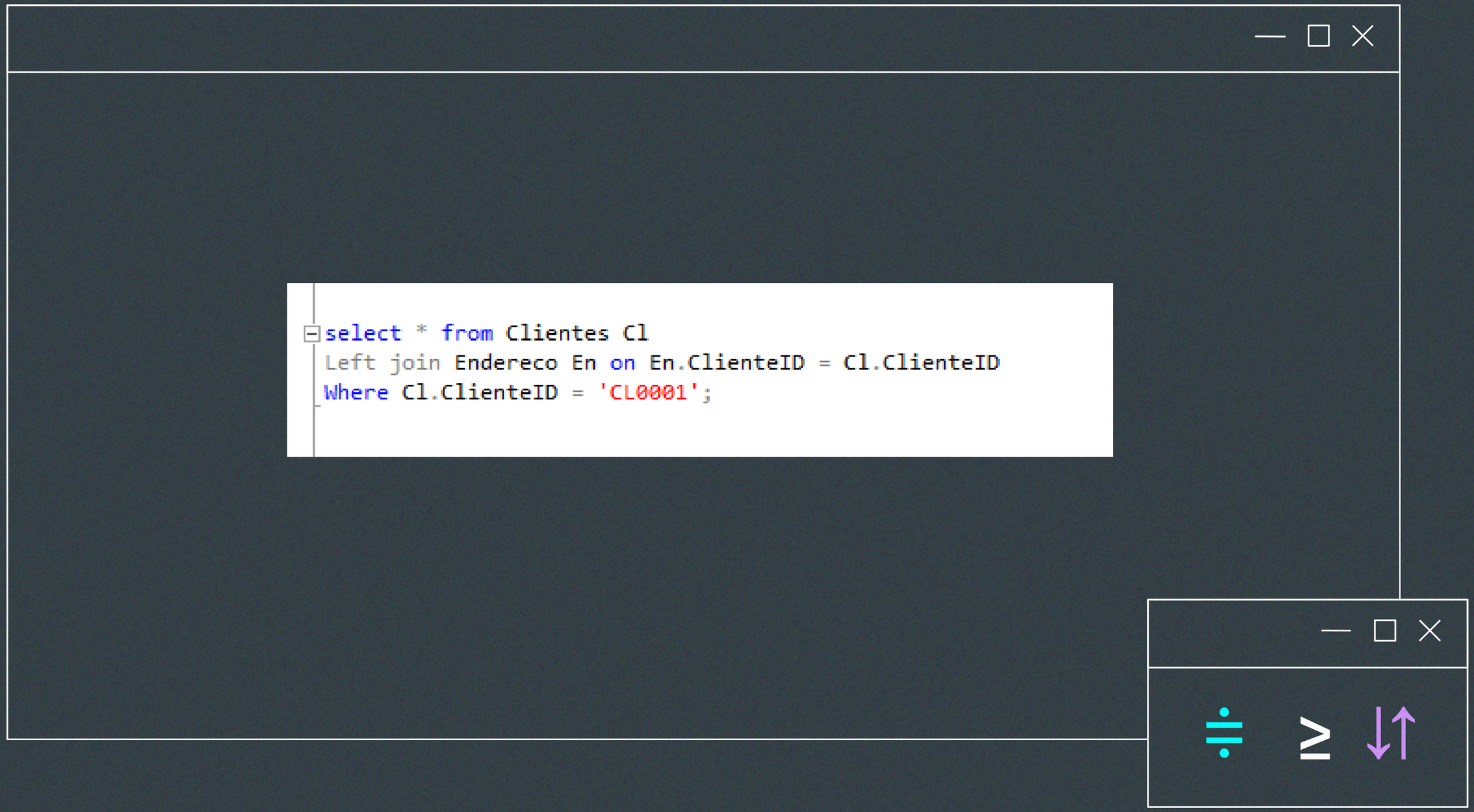
Introdução

A **injeção de SQL** é uma das vulnerabilidades mais comuns em aplicações web, permitindo a exploração de falhas para **acesso não autorizado** ao banco de dados. Este tipo de ataque pode resultar em **roubo de dados sensíveis** e comprometimento da integridade do sistema.

O que é Injeção de SQL?

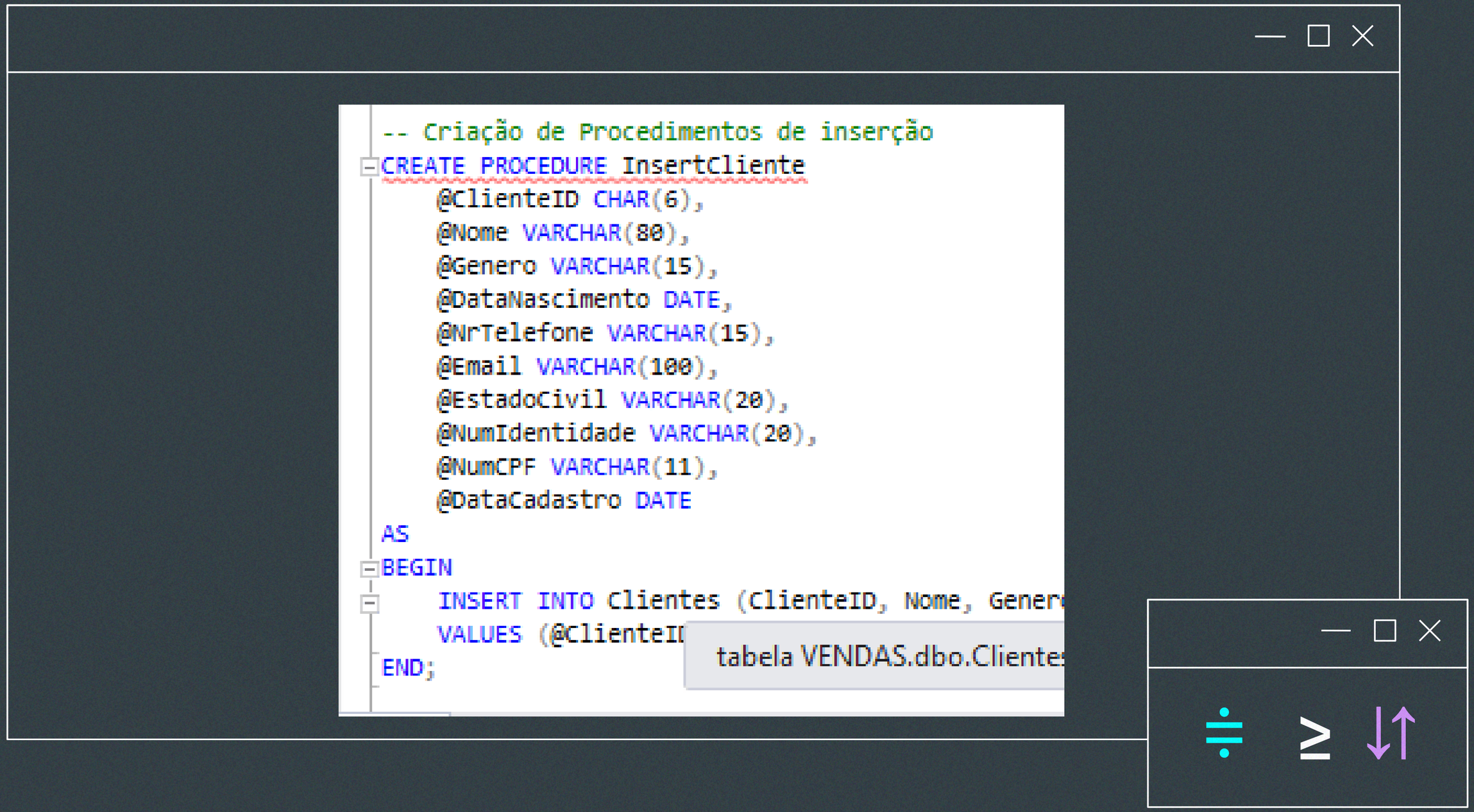
A **injeção de SQL** é uma técnica de ataque que explora falhas de segurança em aplicações web, permitindo que um invasor execute comandos **SQL maliciosos** no banco de dados. Esses ataques podem resultar em **vazamento de dados** e **corrupção de informações**.

```sql
--Tabela de controle
CREATE TABLE LogAuditoria (
    LogID INT IDENTITY(1,1) PRIMARY KEY,
    Acao NVARCHAR(50),
    NomeTabela NVARCHAR(50),
    IDGravado NVARCHAR(50),
    AlteradoPor NVARCHAR(50),
    DataAlterada DATETIME DEFAULT GETDATE()
);
```


select * from Clientes Cl
Left join Endereco En on En.ClienteID = Cl.ClienteID
Where Cl.ClienteID = 'CL0001';

```sql
-- Listando todas as compras do cliente cujo código é "CL0001"
Select Cl.ClienteID  as Codigo,
       Cl.Nome       as Nome,
       Cl.nrTelefone as Telefone,
       Ve.VendaID    as NumeroVenda,
       Ve.DataVenda  as Emissao,
       Ve.FormaPG    as FormaPagto,
       Iv.Produto    as Descricao,
       Iv.Total      as TotalItem,
       Ve.Total      as TotalVenda

--Aplicação de left join para saber o total de vendas e quais itens feitas ao cliente com id "CL0001"

From Clientes Cl
Left join Vendas     Ve on Ve.ClienteID = Cl.ClienteID
Left join ItemVendas Iv on Iv.VendaID   = Ve.VendaID
Where Cl.ClienteID = 'CL0001'
and   Ve.DataVenda between '20240601' and '20240630';
```

```sql
-- Criação de views
    CREATE VIEW vw_cliente AS
   SELECT  ClienteID, Nome, nrTelefone, Email ,EstadoCivil ,DataCadastro
     FROM Clientes

-- SELECT na View para ver o resultado
     select * from vw_cliente
```

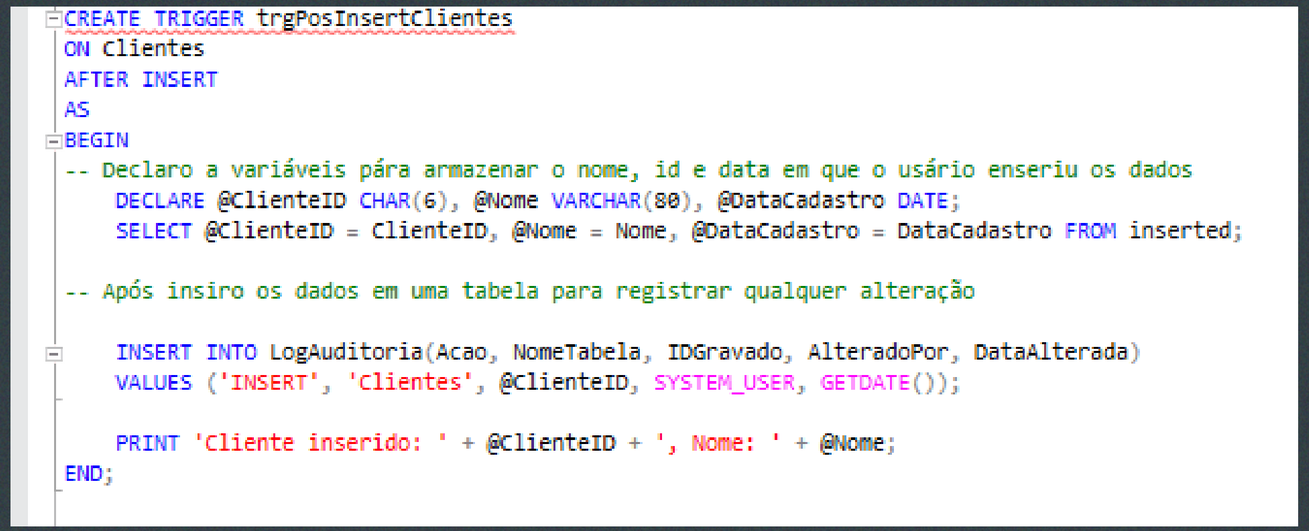

-- Criação de Procedimentos de inserção
CREATE PROCEDURE InsertCliente
@ClienteID CHAR(6),
@Nome VARCHAR(80),
@Genero VARCHAR(15),
@DataNascimento DATE,
@NrTelefone VARCHAR(15),
@Email VARCHAR(100),
@EstadoCivil VARCHAR(20),
@NumIdentidade VARCHAR(20),
@NumCPF VARCHAR(11),
@DataCadastro DATE
AS
BEGIN
INSERT INTO Clientes (ClienteID, Nome, Genero
VALUES (@ClienteID
END;
tabela VENDAS.dbo.Clientes

```sql
CREATE PROCEDURE InsertVendas
    @VendaID   CHAR(6),
    @ClienteID CHAR(6),
    @DataVenda DATE,
    @FormaPG   Varchar(30),
    @Total     DECIMAL(10, 2)

AS
BEGIN
    INSERT INTO Vendas(VendaID, ClienteID, DataVenda, FormaPG, Total)
    VALUES (@VendaID, @ClienteID, @DataVenda, @FormaPG, @Total);
END;
```




CREATE PROCEDURE InsertEndereco
@IdEndereco INT,
@ClienteID CHAR(6),
@Endereco VARCHAR(100),
@Cidade VARCHAR(50),
@Bairro VARCHAR(50),
@CEP VARCHAR(10),
@Estado VARCHAR(50)
AS
BEGIN
INSERT INTO Endereco (ClienteID, Endereco, Cidade, Bairro, CEP, Estado)
VALUES (@ClienteID, @Endereco, @Cidade, @Bairro, @CEP , @Estado);
END;

```sql
CREATE PROCEDURE InsertItemVendas
    @VendaID    CHAR(6),
    @ItemVenda  CHAR(3),
    @Produto    VARCHAR(30),
    @DataVenda  DATE,
    @Quantidade INT,
    @ValorUnit  DECIMAL(10,2),
    @Desconto   DECIMAL(10,2),
    @Total      DECIMAL(10,2)

AS
BEGIN
    INSERT INTO Vendas(VendaID, ItemVenda, Produto, DataVenda, Quantidade, ValorUnit, Desconto, Total)
    VALUES (@VendaID, @ItemVenda, @Produto, @DataVenda, @Quantidade, @ValorUnit, @Desconto, @Total);
END;
```

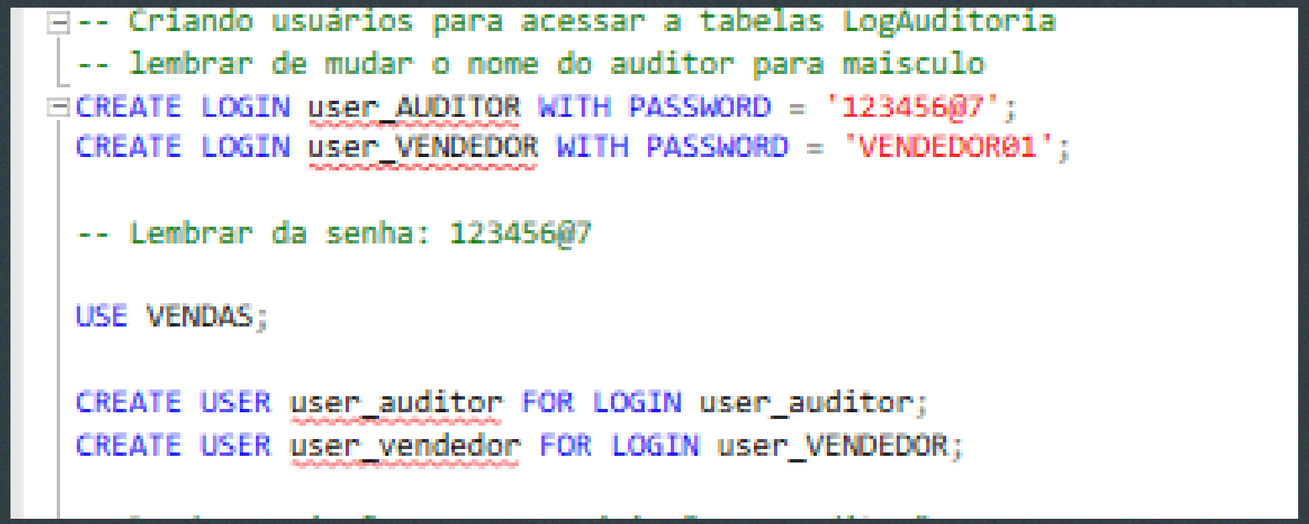


```sql
CREATE TRIGGER trgPosInsertClientes
ON Clientes
AFTER INSERT
AS
BEGIN
-- Declaro a variáveis pára armazenar o nome, id e data em que o usário enseriu os dados
    DECLARE @ClienteID CHAR(6), @Nome VARCHAR(80), @DataCadastro DATE;
    SELECT @ClienteID = ClienteID, @Nome = Nome, @DataCadastro = DataCadastro FROM inserted;

-- Após insiro os dados em uma tabela para registrar qualquer alteração

    INSERT INTO LogAuditoria(Acao, NomeTabela, IDGravado, AlteradoPor, DataAlterada)
    VALUES ('INSERT', 'Clientes', @ClienteID, SYSTEM_USER, GETDATE());

    PRINT 'Cliente inserido: ' + @ClienteID + ', Nome: ' + @Nome;
END;
```

```sql
-- Criação de Trigger para InsertVendas
CREATE TRIGGER trgPosInsertVendas
ON Vendas
AFTER INSERT
AS
BEGIN
    DECLARE @VendaID   CHAR(6),
            @ClienteID CHAR(6),
            @DataVenda DATE,
            @FormaPG   Varchar(30),
            @Total     DECIMAL(10, 2);

    SELECT  @VendaID = VendaID,
            @ClienteID = ClienteID,
            @DataVenda = DataVenda,
            @FormaPG = FormaPG,
            @Total = Total
            FROM inserted;

    -- Inserindo os dados na tabela de log
    INSERT INTO LogAuditoria (Acao, NomeTabela, IDGravado, AlteradoPor, DataAlterada)
    VALUES ('INSERT', 'Vendas', @VendaID, SYSTEM_USER, GETDATE());


END;
```

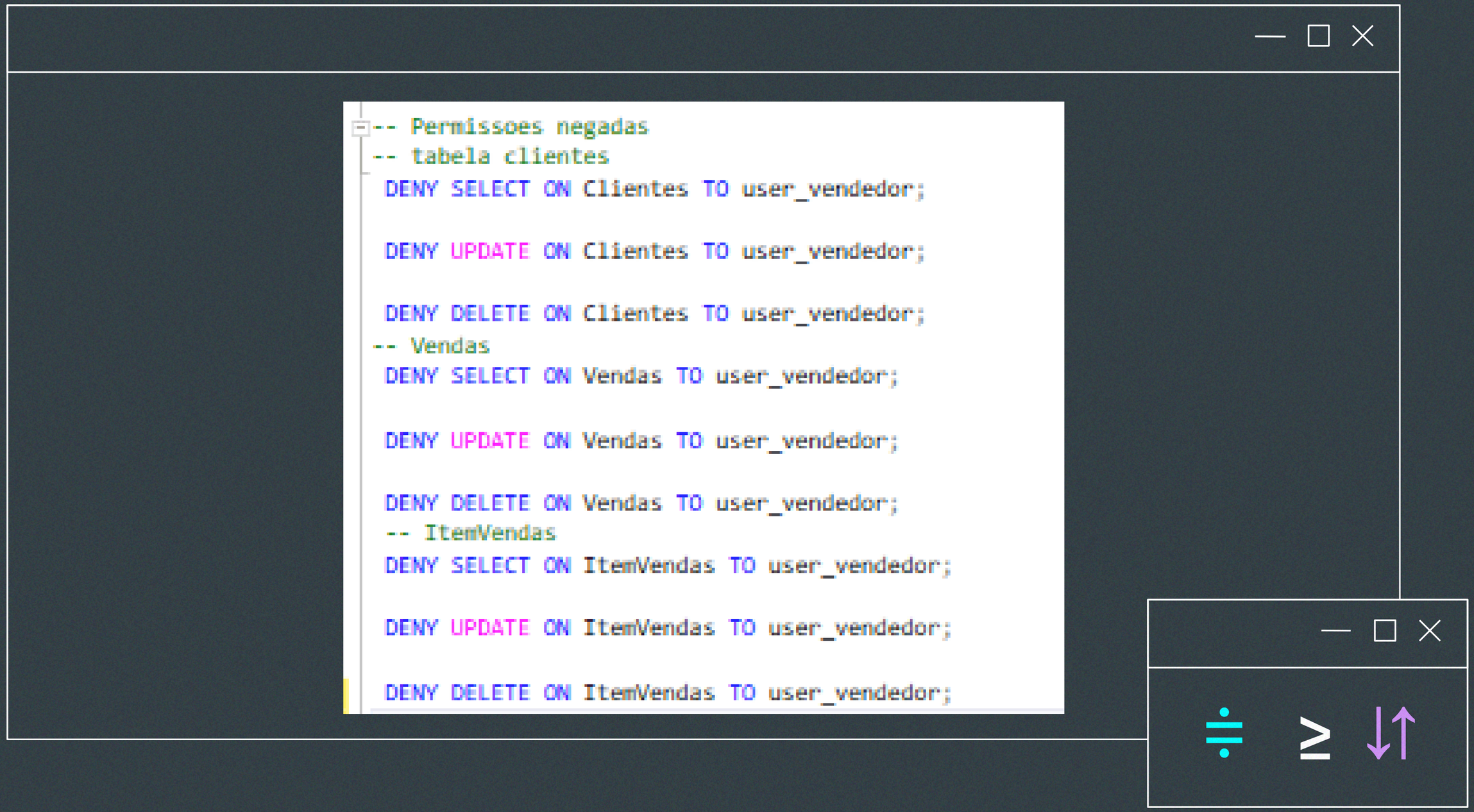
```sql
-- Criação de Trigger para InsertEndereco
CREATE TRIGGER trgPosInsertEndereco
ON Endereco
AFTER INSERT
AS
BEGIN
    DECLARE @IdEndereco  INT,
            @ClienteID   CHAR(6),
            @Endereco    VARCHAR(100),
            @Cidade      VARCHAR(50),
            @Bairro      VARCHAR(50),
            @CEP      VARCHAR(10),
            @Estado   VARCHAR(50)

    SELECT  @IdEndereco = IdEndereco,
            @ClienteID = ClienteID,
            @Endereco = Endereco,
            @Cidade = Cidade,
            @Bairro = Bairro,
            @CEP = CEP,
            @Estado = Estado
            FROM inserted;

    -- Inserindo os dados na tabela de log
    INSERT INTO LogAuditoria (Acao, NomeTabela, IDGravado, AlteradoPor, DataAlterada)
    VALUES ('INSERT', 'Endereco', @IdEndereco, SYSTEM_USER, GETDATE());


END;
```

```sql
-- Criação de Trigger para InsertItemVendas
CREATE TRIGGER trgPosInsertItemVendas
ON ItemVendas
AFTER INSERT
AS
BEGIN
    DECLARE @VendaID    CHAR(6),
            @ItemVenda  CHAR(3),
            @Produto    VARCHAR(30),
            @DataVenda  DATE,
            @Quantidade INT,
            @ValorUnit  DECIMAL(10,2),
            @Desconto   DECIMAL(10,2),
            @Total      DECIMAL(10,2)

    SELECT  @VendaID = VendaID,
            @ItemVenda = ItemVenda,
            @Produto = Produto,
            @DataVenda = DataVenda,
            @Quantidade = Quantidade,
            @ValorUnit = ValorUnit,
            @Desconto = Desconto,
            @Total = Total
            FROM inserted;

    -- Inserindo os dados na tabela de log
    INSERT INTO LogAuditoria (Acao, NomeTabela, IDGravado, AlteradoPor, DataAlterada)
    VALUES ('INSERT', 'ItemVendas', @ItemVenda, SYSTEM_USER, GETDATE());


END;
```



```sql
-- Criando usuários para acessar a tabelas LogAuditoria
-- lembrar de mudar o nome do auditor para maisculo
CREATE LOGIN user_AUDITOR WITH PASSWORD = '123456@7';
CREATE LOGIN user_VENDEDOR WITH PASSWORD = 'VENDEDOR01';

-- Lembrar da senha: 123456@7

USE VENDAS;

CREATE USER user_auditor FOR LOGIN user_auditor;
CREATE USER user_vendedor FOR LOGIN user_VENDEDOR;
```

```sql
-- Dando permissão para o usuário "user_auditor"
GRANT SELECT ON LogAuditoria TO user_auditor;

GRANT INSERT ON LogAuditoria TO user_auditor;

GRANT UPDATE ON LogAuditoria TO user_auditor;

GRANT DELETE ON LogAuditoria TO user_auditor;

-- Permissoes do vendedor
GRANT SELECT ON vw_cliente TO user_vendedor;

GRANT INSERT ON vw_cliente TO user_vendedor;

GRANT UPDATE ON vw_cliente TO user_vendedor;

GRANT DELETE ON vw_cliente TO user_vendedor;
```


-- Permissoes negadas
-- tabela clientes
DENY SELECT ON Clientes TO user_vendedor;
DENY UPDATE ON Clientes TO user_vendedor;
DENY DELETE ON Clientes TO user_vendedor;
-- Vendas
DENY SELECT ON Vendas TO user_vendedor;
DENY UPDATE ON Vendas TO user_vendedor;
DENY DELETE ON Vendas TO user_vendedor;
-- ItemVendas
DENY SELECT ON ItemVendas TO user_vendedor;
DENY UPDATE ON ItemVendas TO user_vendedor;
DENY DELETE ON ItemVendas TO user_vendedor;

```sql
-- LogAuditoria
 DENY SELECT ON LogAuditoria TO user_vendedor;

 DENY UPDATE ON LogAuditoria TO user_vendedor;

 DENY DELETE ON LogAuditoria TO user_vendedor;



-- Verificar permissões do usuário user_auditor
-- COMANDO DE DEBUG APAGAR DEPOIS DO USO
EXECUTE AS USER = 'user_auditor';
SELECT * FROM fn_my_permissions('LogAuditoria', 'OBJECT');
REVERT;

EXECUTE AS USER = 'user_vendedor';
SELECT * FROM fn_my_permissions('LogAuditoria', 'OBJECT');
REVERT;
```

```sql
-- EXECUTANDO COMANDO DELETE COMO O "use_vendedor"
EXECUTE AS USER = 'user_vendedor';
DELETE LogAuditoria;
REVERT;
```

Thanks!

O código na integra estará disponível no link: git hub