



Campus Virtual (SIA)

Desenvolvimento Full Stack

Nível 2: Vamos Manter as Informações

Turma EAD | 3º Semestre

Integrante: **Gabriel B Mathias**

Repositório: <https://github.com/GabrielBM-git/Missao-Pratica-Nivel-2-Mundo-3>

Título da Prática

Missão Prática | Nível 2 | Mundo 3

Objetivos da Prática

1. Identificar os requisitos de um sistema e transformá-los no modelo adequado.
2. Utilizar ferramentas de modelagem para bases de dados relacionais.
3. Explorar a sintaxe SQL na criação das estruturas do banco (DDL).
4. Explorar a sintaxe SQL na consulta e manipulação de dados (DML).
5. No final do exercício, o aluno terá vivenciado a experiência de modelar a base de dados para um sistema simples, além de implementá-la, através da sintaxe SQL, na plataforma do SQL Server.

Implementação

1º Procedimento | Criando o Banco de Dados

Banco: Loja

```
-- Criando o banco de dados
CREATE DATABASE Loja;
GO
```

Tabela: Usuario

```
USE Loja;
GO

-- Tabela de Usuários
CREATE TABLE Usuario (
    id INT IDENTITY(1,1) PRIMARY KEY,
    username NVARCHAR(50) NOT NULL UNIQUE,
    senha NVARCHAR(255) NOT NULL
);
GO
```

Tabela: Pessoa

```
USE Loja;
GO

-- Criando a sequence para gerar IDs de Pessoa
CREATE SEQUENCE pessoa_seq START WITH 1 INCREMENT BY 1;
GO

-- Tabela de Pessoas (Base para PessoaFisica e PessoaJuridica)
CREATE TABLE Pessoa (
    id INT PRIMARY KEY DEFAULT NEXT VALUE FOR pessoa_seq,
    nome NVARCHAR(100) NOT NULL,
    endereco NVARCHAR(255) NOT NULL,
    cidade NVARCHAR(50) NOT NULL,
    estado NVARCHAR(2) NOT NULL,
    telefone NVARCHAR(20) NULL,
    email NVARCHAR(100) NULL,
    tipo CHAR(1) NOT NULL CHECK (tipo IN ('F', 'J')) -- 'F' para Pessoa Física, 'J' para Jurídica
);
GO
```

Tabela: PessoaFisica

```
USE Loja;
GO

-- Tabela de Pessoa Física
CREATE TABLE PessoaFisica (
    id INT PRIMARY KEY,
    cpf CHAR(11) UNIQUE NOT NULL,
    CONSTRAINT FK_PessoaFisica FOREIGN KEY (id) REFERENCES Pessoa(id) ON DELETE CASCADE
);
GO
```

Tabela: PessoaJuridica

```
USE Loja;
GO

-- Tabela de Pessoa Jurídica
CREATE TABLE PessoaJuridica (
    id INT PRIMARY KEY,
    cnpj CHAR(14) UNIQUE NOT NULL,
    CONSTRAINT FK_PessoaJuridica FOREIGN KEY (id) REFERENCES Pessoa(id) ON DELETE CASCADE
);
GO
```

Tabela: Produto

```
USE Loja;
GO

-- Tabela de Produtos
CREATE TABLE Produto (
    id INT IDENTITY(1,1) PRIMARY KEY,
    nome NVARCHAR(100) NOT NULL UNIQUE,
    quantidade INT NOT NULL CHECK (quantidade >= 0),
    precoVenda DECIMAL(10,2) NOT NULL CHECK (precoVenda >= 0)
);
GO
```

Tabela: Compra

```
USE Loja;
GO

-- Tabela de Compras (Movimentação de entrada)
CREATE TABLE Compra (
    id INT IDENTITY(1,1) PRIMARY KEY,
    usuario_id INT NOT NULL,
    produto_id INT NOT NULL,
    fornecedor_id INT NOT NULL, -- Sempre uma Pessoa Jurídica
    quantidade INT NOT NULL CHECK (quantidade > 0),
    precoUnitario DECIMAL(10,2) NOT NULL CHECK (precoUnitario >= 0),
    dataCompra DATETIME DEFAULT GETDATE(),

    CONSTRAINT FK_Compra_Usuario FOREIGN KEY (usuario_id) REFERENCES Usuario(id),
    CONSTRAINT FK_Compra_Produto FOREIGN KEY (produto_id) REFERENCES Produto(id),
    CONSTRAINT FK_Compra_Fornecedor FOREIGN KEY (fornecedor_id) REFERENCES
PessoaJuridica(id)
);
GO
```

Tabela: Venda

```
USE Loja;
GO

-- Tabela de Vendas (Movimentação de saída)
CREATE TABLE Venda (
    id INT IDENTITY(1,1) PRIMARY KEY,
    usuario_id INT NOT NULL,
    produto_id INT NOT NULL,
    cliente_id INT NOT NULL, -- Sempre uma Pessoa Física
    quantidade INT NOT NULL CHECK (quantidade > 0),
    precoUnitario DECIMAL(10,2) NOT NULL CHECK (precoUnitario >= 0),
    dataVenda DATETIME DEFAULT GETDATE(),

    CONSTRAINT FK_Venda_Usuario FOREIGN KEY (usuario_id) REFERENCES Usuario(id),
    CONSTRAINT FK_Venda_Produto FOREIGN KEY (produto_id) REFERENCES Produto(id),
    CONSTRAINT FK_Venda_Cliente FOREIGN KEY (cliente_id) REFERENCES PessoaFisica(id)
);
GO
```

Execução dos Códigos

Script: dbLoja.sql

```
-- Criando o banco de dados
CREATE DATABASE Loja;
GO

USE Loja;
GO

-- Criando a sequence para gerar IDs de Pessoa
CREATE SEQUENCE pessoa_seq START WITH 1 INCREMENT BY 1;
GO

-- Tabela de Usuários
CREATE TABLE Usuario (
    id INT IDENTITY(1,1) PRIMARY KEY,
    username NVARCHAR(50) NOT NULL UNIQUE,
    senha NVARCHAR(255) NOT NULL
);
GO

-- Tabela de Pessoas (Base para PessoaFisica e PessoaJuridica)
CREATE TABLE Pessoa (
    id INT PRIMARY KEY DEFAULT NEXT VALUE FOR pessoa_seq,
    nome NVARCHAR(100) NOT NULL,
    endereco NVARCHAR(255) NOT NULL,
    cidade NVARCHAR(50) NOT NULL,
    estado NVARCHAR(2) NOT NULL,
    telefone NVARCHAR(20) NULL,
    email NVARCHAR(100) NULL,
    tipo CHAR(1) NOT NULL CHECK (tipo IN ('F', 'J')) -- 'F' para Pessoa Física, 'J' para Jurídica
);
GO

-- Tabela de Pessoa Física
CREATE TABLE PessoaFisica (
    id INT PRIMARY KEY,
    cpf CHAR(11) UNIQUE NOT NULL,
    CONSTRAINT FK_PessoaFisica FOREIGN KEY (id) REFERENCES Pessoa(id) ON DELETE CASCADE
);
GO
```

```

-- Tabela de Pessoa Jurídica
CREATE TABLE PessoaJuridica (
    id INT PRIMARY KEY,
    cnpj CHAR(14) UNIQUE NOT NULL,
    CONSTRAINT FK_PessoaJuridica FOREIGN KEY (id) REFERENCES Pessoa(id) ON DELETE CASCADE
);
GO

-- Tabela de Produtos
CREATE TABLE Produto (
    id INT IDENTITY(1,1) PRIMARY KEY,
    nome NVARCHAR(100) NOT NULL UNIQUE,
    quantidade INT NOT NULL CHECK (quantidade >= 0),
    precoVenda DECIMAL(10,2) NOT NULL CHECK (precoVenda >= 0)
);
GO

-- Tabela de Compras (Movimentação de entrada)
CREATE TABLE Compra (
    id INT IDENTITY(1,1) PRIMARY KEY,
    usuario_id INT NOT NULL,
    produto_id INT NOT NULL,
    fornecedor_id INT NOT NULL, -- Sempre uma Pessoa Jurídica
    quantidade INT NOT NULL CHECK (quantidade > 0),
    precoUnitario DECIMAL(10,2) NOT NULL CHECK (precoUnitario >= 0),
    dataCompra DATETIME DEFAULT GETDATE(),

    CONSTRAINT FK_Compra_Usuario FOREIGN KEY (usuario_id) REFERENCES Usuario(id),
    CONSTRAINT FK_Compra_Produto FOREIGN KEY (produto_id) REFERENCES Produto(id),
    CONSTRAINT FK_Compra_Fornecedor FOREIGN KEY (fornecedor_id) REFERENCES
PessoaJuridica(id)
);
GO

-- Tabela de Vendas (Movimentação de saída)
CREATE TABLE Venda (
    id INT IDENTITY(1,1) PRIMARY KEY,
    usuario_id INT NOT NULL,
    produto_id INT NOT NULL,
    cliente_id INT NOT NULL, -- Sempre uma Pessoa Física
    quantidade INT NOT NULL CHECK (quantidade > 0),
    precoUnitario DECIMAL(10,2) NOT NULL CHECK (precoUnitario >= 0),
    dataVenda DATETIME DEFAULT GETDATE(),

    CONSTRAINT FK_Venda_Usuario FOREIGN KEY (usuario_id) REFERENCES Usuario(id),
    CONSTRAINT FK_Venda_Produto FOREIGN KEY (produto_id) REFERENCES Produto(id),

```

```
CONSTRAINT FK_Venda_Cliente FOREIGN KEY (cliente_id) REFERENCES PessoaFisica(id)
);
GO
```

Resultado dos Códigos

Script: dbLoja.sql

Comandos concluídos com êxito.

Horário de conclusão: 2025-04-05T11:07:25.5023531-03:00

Análise e Conclusão

- A. Como são implementadas as diferentes cardinalidades, basicamente 1X1, 1XN ou NxN, em um banco de dados relacional?

1x1 (Um para Um):

- Uma tabela tem relação com **uma** linha de outra.
- Usa-se uma **chave estrangeira única**.

1xN (Um para Muitos):

- Uma linha da tabela A se relaciona com **várias** da tabela B.
- Tabela B tem **chave estrangeira** para A.

NxN (Muitos para Muitos):

- Ambas as tabelas se relacionam com **várias linhas** da outra.

- Usa-se uma **tabela intermediária (de junção)** com FKs.
- B. Que tipo de relacionamento deve ser utilizado para representar o uso de herança em bancos de dados relacionais?

Tabela Única (Single Table):

- Tudo em uma tabela, com campo que indica o tipo.

Tabela por Subclasse:

- Uma tabela para a classe base e outras para subclasses.

Tabelas por Classe Concreta:

- Cada subclasse vira uma tabela com todos os campos.

- C. Como o SQL Server Management Studio permite a melhoria da produtividade nas tarefas relacionadas ao gerenciamento do banco de dados?

- **Interface gráfica** facilita o uso sem precisar escrever tudo em SQL.
- **IntelliSense** ajuda com autocompletar e reduz erros.
- **Templates prontos** aceleram tarefas comuns.
- **Gerenciamento fácil** de usuários, permissões e backups.
- **Monitoramento e agendamento** de tarefas automatizadas.
- **Diagramas visuais** ajudam a entender o modelo de dados.

Implementação

2º Procedimento | Alimentando a Base

Tabela: Usuario

```
USE [Loja]
GO

INSERT INTO [dbo].[Usuario]
    ([username]
    ,[senha])
VALUES
    ('op1'
    ,'op1')
GO

INSERT INTO [dbo].[Usuario]
    ([username]
    ,[senha])
VALUES
    ('op2'
    ,'op2')
GO
```

Tabela: Produto

```
USE [Loja]
GO

INSERT INTO [dbo].[Produto]
    ([nome]
    ,[quantidade]
    ,[precoVenda])
VALUES
    ('Banana'
    ,100
    ,5.00)
GO

INSERT INTO [dbo].[Produto]
    ([nome]
    ,[quantidade]
    ,[precoVenda])
```

```
VALUES
    ('Laranja'
     ,500
     ,2.00)
GO

INSERT INTO [dbo].[Produto]
    ([nome]
     ,[quantidade]
     ,[precoVenda])
VALUES
    ('Manga'
     ,800
     ,4.00)
GO
```

Tabela: Pessoa (Física)

```
USE [Loja]
GO

DECLARE @PessoaFisicald INT = NEXT VALUE FOR pessoa_seq;

INSERT INTO [dbo].[Pessoa]
    ([id]
     ,[nome]
     ,[endereco]
     ,[cidade]
     ,[estado]
     ,[telefone]
     ,[email]
     ,[tipo])
VALUES
    (@PessoaFisicald
     ,Joao'
     ,Rua 12, casa 3, Quitanda'
     ,Riacho do Sul'
     ,PA'
     ,1111-1111'
     ,joao@riacho.com'
     ,F') -- 'F' para Pessoa Física

INSERT INTO [dbo].[PessoaFisica]
    ([id]
```

```

        ,[cpf])
VALUES
    (@PessoaFisicald
    ,'111.111.111-11')
GO

```

Tabela: Pessoa (Juridica)

```

USE [Loja]
GO

DECLARE @PessoaJuridicald INT = NEXT VALUE FOR pessoa_seq;

INSERT INTO [dbo].[Pessoa]
    ([id]
    ,[nome]
    ,[endereco]
    ,[cidade]
    ,[estado]
    ,[telefone]
    ,[email]
    ,[tipo])
VALUES
    (@PessoaJuridicald
    ,'JJC'
    ,'Rua 11, Centro'
    ,'Riacho do Norte'
    ,'PA'
    ,'1212-1212'
    ,'jjc@riacho.com'
    ,'J') -- 'J' para Jurídica

INSERT INTO [dbo].[PessoaJuridica]
    ([id]
    ,[cnpj])
VALUES
    (@PessoaJuridicald
    ,'22.222.222/2222-22')
GO

```

Tabela: Compra

```

USE [Loja]
GO

INSERT INTO [dbo].[Compra]

```

```
        ([usuario_id]
        ,[produto_id]
        ,[fornecedor_id]
        ,[quantidade]
        ,[precoUnitario]
        ,[dataCompra])
VALUES
    (1
    ,1
    ,2
    ,100
    ,2.00
    ,GETDATE())
GO

INSERT INTO [dbo].[Compra]
    ([usuario_id]
    ,[produto_id]
    ,[fornecedor_id]
    ,[quantidade]
    ,[precoUnitario]
    ,[dataCompra])
VALUES
    (1
    ,2
    ,2
    ,500
    ,1.00
    ,GETDATE())
GO

INSERT INTO [dbo].[Compra]
    ([usuario_id]
    ,[produto_id]
    ,[fornecedor_id]
    ,[quantidade]
    ,[precoUnitario]
    ,[dataCompra])
VALUES
    (1
    ,3
    ,2
    ,800
    ,2.00
    ,GETDATE())
GO
```

Tabela: Venda

```
USE [Loja]
GO

INSERT INTO [dbo].[Venda]
    ([usuario_id]
    ,[produto_id]
    ,[cliente_id]
    ,[quantidade]
    ,[precoUnitario]
    ,[dataVenda])
VALUES
    (2
    ,1
    ,1
    ,10
    ,5.00
    ,GETDATE())
GO

INSERT INTO [dbo].[Venda]
    ([usuario_id]
    ,[produto_id]
    ,[cliente_id]
    ,[quantidade]
    ,[precoUnitario]
    ,[dataVenda])
VALUES
    (2
    ,2
    ,1
    ,10
    ,2.00
    ,GETDATE())
GO

INSERT INTO [dbo].[Venda]
    ([usuario_id]
    ,[produto_id]
    ,[cliente_id]
    ,[quantidade]
    ,[precoUnitario]
    ,[dataVenda])
VALUES
    (2
```

```
,3
,1
,10
,4.00
,GETDATE())
GO
```

Execução dos Códigos

Script: dbLoja-carga_inicial.sql

```
USE [Loja]
GO

INSERT INTO [dbo].[Usuario]
    ([username]
    ,[senha])
VALUES
    ('op1'
    ,'op1')
GO

INSERT INTO [dbo].[Usuario]
    ([username]
    ,[senha])
VALUES
    ('op2'
    ,'op2')
GO
```

```
-----

INSERT INTO [dbo].[Produto]
    ([nome]
    ,[quantidade]
    ,[precoVenda])
VALUES
    ('Banana'
    ,100
    ,5.00)
GO
```

```
INSERT INTO [dbo].[Produto]
    ([nome]
```

```

        ,[quantidade]
        ,[precoVenda])
VALUES
    ('Laranja'
    ,500
    ,2.00)
GO

INSERT INTO [dbo].[Produto]
    ([nome]
    ,[quantidade]
    ,[precoVenda])
VALUES
    ('Manga'
    ,800
    ,4.00)
GO

-----

DECLARE @PessoaFisicald INT = NEXT VALUE FOR pessoa_seq;

INSERT INTO [dbo].[Pessoa]
    ([id]
    ,[nome]
    ,[endereco]
    ,[cidade]
    ,[estado]
    ,[telefone]
    ,[email]
    ,[tipo])
VALUES
    (@PessoaFisicald
    ,'Joao'
    ,'Rua 12, casa 3, Quitanda'
    ,'Riacho do Sul'
    ,'PA'
    ,'1111-1111'
    ,'joao@riacho.com'
    ,'F') -- 'F' para Pessoa Física

INSERT INTO [dbo].[PessoaFisica]
    ([id]
    ,[cpf])
VALUES
    (@PessoaFisicald

```

```

        , '111.111.111-11')
GO

-----

DECLARE @PessoaJuridicald INT = NEXT VALUE FOR pessoa_seq;

INSERT INTO [dbo].[Pessoa]
    ([id]
    , [nome]
    , [endereco]
    , [cidade]
    , [estado]
    , [telefone]
    , [email]
    , [tipo])
VALUES
    (@PessoaJuridicald
    , 'JJC'
    , 'Rua 11, Centro'
    , 'Riacho do Norte'
    , 'PA'
    , '1212-1212'
    , 'jjc@riacho.com'
    , 'J') -- 'J' para Jurídica

INSERT INTO [dbo].[PessoaJuridica]
    ([id]
    , [cnpj])
VALUES
    (@PessoaJuridicald
    , '22.222.222/2222-22')
GO

-----

INSERT INTO [dbo].[Compra]
    ([usuario_id]
    , [produto_id]
    , [fornecedor_id]
    , [quantidade]
    , [precoUnitario]
    , [dataCompra])
VALUES
    (1
    , 1

```



```
,2
,100
,2.00
,GETDATE())
GO
```

```
INSERT INTO [dbo].[Compra]
([usuario_id]
,[produto_id]
,[fornecedor_id]
,[quantidade]
,[precoUnitario]
,[dataCompra])
VALUES
(1
,2
,2
,500
,1.00
,GETDATE())
GO
```

```
INSERT INTO [dbo].[Compra]
([usuario_id]
,[produto_id]
,[fornecedor_id]
,[quantidade]
,[precoUnitario]
,[dataCompra])
VALUES
(1
,3
,2
,800
,2.00
,GETDATE())
GO
```

```
INSERT INTO [dbo].[Venda]
([usuario_id]
,[produto_id]
,[cliente_id]
,[quantidade]
,[precoUnitario]
```

```
        ,[dataVenda])
VALUES
    (2
    ,1
    ,1
    ,10
    ,5.00
    ,GETDATE())
GO

INSERT INTO [dbo].[Venda]
    ([usuario_id]
    ,[produto_id]
    ,[cliente_id]
    ,[quantidade]
    ,[precoUnitario]
    ,[dataVenda])
VALUES
    (2
    ,2
    ,1
    ,10
    ,2.00
    ,GETDATE())
GO

INSERT INTO [dbo].[Venda]
    ([usuario_id]
    ,[produto_id]
    ,[cliente_id]
    ,[quantidade]
    ,[precoUnitario]
    ,[dataVenda])
VALUES
    (2
    ,3
    ,1
    ,10
    ,4.00
    ,GETDATE())
GO
```

Script: dbLoja-consultas.sql

```
/* Consultas sobre os dados inseridos */

-- a.) Dados completos de pessoas físicas

SELECT
    p.id,
    p.nome,
    p.endereco,
    p.cidade,
    p.estado,
    p.telefone,
    p.email,
    pf.cpf
FROM Pessoa p
JOIN PessoaFisica pf ON p.id = pf.id;

-- b.) Dados completos de pessoas jurídicas

SELECT
    p.id,
    p.nome,
    p.endereco,
    p.cidade,
    p.estado,
    p.telefone,
    p.email,
    pj.cnpj
FROM Pessoa p
JOIN PessoaJuridica pj ON p.id = pj.id;

-- c.) Movimentações de entrada (compras)

SELECT
    pr.nome AS Produto,
    p.nome AS Fornecedor,
    c.quantidade,
    c.precoUnitario,
    c.quantidade * c.precoUnitario AS ValorTotal
FROM Compra c
JOIN Produto pr ON pr.id = c.produto_id
JOIN PessoaJuridica pj ON pj.id = c.fornecedor_id
JOIN Pessoa p ON p.id = pj.id;

-- d.) Movimentações de saída (vendas)
```

```

SELECT
    pr.nome AS Produto,
    p.nome AS Comprador,
    v.quantidade,
    v.precoUnitario,
    v.quantidade * v.precoUnitario AS ValorTotal
FROM Venda v
JOIN Produto pr ON pr.id = v.produto_id
JOIN PessoaFisica pf ON pf.id = v.cliente_id
JOIN Pessoa p ON p.id = pf.id;

-- e.) Valor total das entradas agrupadas por produto

SELECT
    pr.nome AS Produto,
    SUM(c.quantidade * c.precoUnitario) AS ValorTotalEntradas
FROM Compra c
JOIN Produto pr ON pr.id = c.produto_id
GROUP BY pr.nome;

-- f.) Valor total das saídas agrupadas por produto

SELECT
    pr.nome AS Produto,
    SUM(v.quantidade * v.precoUnitario) AS ValorTotalSaidas
FROM Venda v
JOIN Produto pr ON pr.id = v.produto_id
GROUP BY pr.nome;

-- g.) Operadores que não efetuaram movimentações de entrada (compra)

SELECT
    u.id,
    u.username
FROM Usuario u
WHERE NOT EXISTS (
    SELECT 1
    FROM Compra c
    WHERE c.usuario_id = u.id
);

-- h.) Valor total de entrada, agrupado por operador

SELECT
    u.username,

```

```
SUM(c.quantidade * c.precoUnitario) AS TotalEntrada
FROM Compra c
JOIN Usuario u ON u.id = c.usuario_id
GROUP BY u.username;

-- i.) Valor total de saída, agrupado por operador

SELECT
    u.username,
    SUM(v.quantidade * v.precoUnitario) AS TotalSaida
FROM Venda v
JOIN Usuario u ON u.id = v.usuario_id
GROUP BY u.username;

-- j.) Valor médio de venda por produto, utilizando média ponderada
-- (Média ponderada = soma(qtd * preço) / soma(qtd))

SELECT
    pr.nome AS Produto,
    CAST(SUM(v.quantidade * v.precoUnitario) AS DECIMAL(10,2)) /
    NULLIF(SUM(v.quantidade), 0) AS MediaPonderadaVenda
FROM Venda v
JOIN Produto pr ON pr.id = v.produto_id
GROUP BY pr.nome;
```

Resultado dos Códigos

Script: dbLoja-carga_inicial.sql

(1 linha afetada)

(1 linha afetada)

(1 linha afetada)

(1 linha afetada)

(1 linha afetada)

(1 linha afetada)

(1 linha afetada)

(1 linha afetada)

(1 linha afetada)

(1 linha afetada)

(1 linha afetada)

(1 linha afetada)

(1 linha afetada)

(1 linha afetada)

(1 linha afetada)

Horário de conclusão: 2025-04-05T13:55:31.1251731-03:00

Script: dbLoja-consultas.sql

Resultado das consultas sobre os dados inseridos +

Resultados		Mensagens						
	id	nome	endereço	cidade	estado	telefone	email	cpf
1	1	Joao	Rua 12, casa 3, Quitanda	Riacho do Sul	PA	1111-1111	joao@riacho.com	111.111.111-11

	id	nome	endereço	cidade	estado	telefone	email	cnpj
1	2	JJC	Rua 11, Centro	Riacho do Norte	PA	1212-1212	jjc@riacho.com	22.222.222/2222-22

	Produto	Fornecedor	quantidade	preço Unitário	Valor Total
1	Banana	JJC	100	2.00	200.00
2	Laranja	JJC	500	1.00	500.00
3	Manga	JJC	800	2.00	1600.00

	Produto	Comprador	quantidade	preço Unitário	Valor Total
1	Banana	Joao	10	5.00	50.00
2	Laranja	Joao	10	2.00	20.00
3	Manga	Joao	10	4.00	40.00

	Produto	Valor Total Entradas
1	Banana	200.00
2	Laranja	500.00
3	Manga	1600.00

	Produto	Valor Total Sidas
1	Banana	50.00
2	Laranja	20.00
3	Manga	40.00

	id	username
1	2	op2

	username	Total Entrada
1	op1	2300.00

	username	Total Saida
1	op2	110.00

	Produto	Média Ponderada Venda
1	Banana	5.00000000000000
2	Laranja	2.00000000000000
3	Manga	4.00000000000000



Análise e Conclusão

A. Quais as diferenças no uso de sequence e identity?

As **principais diferenças entre SEQUENCE e IDENTITY** estão no modo como os valores de chave primária (ou outras colunas auto-incrementáveis) são gerados e controlados nos bancos de dados relacionais.

B. Qual a importância das chaves estrangeiras para a consistência do banco?

As **chaves estrangeiras (foreign keys)** são fundamentais para garantir a **consistência referencial** em um banco de dados relacional. Elas impõem regras que asseguram que os dados estejam sempre **relacionados corretamente** entre as tabelas.

C. Quais operadores do SQL pertencem à álgebra relacional e quais são definidos no cálculo relacional?

- **Álgebra Relacional:** foca no *como fazer* (operacional)
- **Cálculo Relacional:** foca no *que se quer obter* (declarativo)

D. Como é feito o agrupamento em consultas, e qual requisito é obrigatório?

O agrupamento é feito usando a cláusula **GROUP BY**, que serve para **agrupar registros com base em uma ou mais colunas**, permitindo realizar **funções de agregação** (como **SUM, COUNT, AVG, MAX, MIN**) em cada grupo.

