Aluno: Altair Silva Filho
Matrícula: 202403254409
Unidade: Palhoça – SC
Curso: Desenvolvimento Full Stack
Disciplina: Vamos manter as informações?
Período: 2025.1

Relatório da Missão Prática – Nível 2 – Mundo 3

Vamos Manter as Informações?

2º Procedimento | Alimentando a Base

Objetivo

Utilizar o SQL Server Management Studio para alimentar as tabelas com dados básicos do sistema. Depois, usar os comandos do SQL para efetuar diversas consultas.

Códigos do Projeto

Scripts para a inserção de Dados

Inserção de Usuários (Operadores)

```
INSERT INTO Usuario (IdUsuario, Login, senha)
VALUES
(1, 'op1', 'op1'),
(2, 'op2', 'op2');
```

Inserção de Produtos

```
INSERT INTO Produto (IdProduto, nome, quantidade, precoVenda)
VALUES
(1, 'Banana', 100, 5.00),
(2, 'Laranja', 200, 3.00),
(3, 'Manga', 400, 6.00);
```

Inserção de Pessoas

```
CREATE SEQUENCE Seq_Id_TablePessoa
AS INT
START WITH 1
INCREMENT BY 1;
```

```
INSERT INTO Pessoa (idPessoa, nome_razaoSocial, logradouro, cidade, estado, telefone, email)
VALUES
(NEXT VALUE FOR Seq_Id_TablePessoa, 'Maria Pereira', 'Rua São José', 'São José', 'SC', '48988880011',
'mariap@estacio.br');

INSERT INTO PessoaFisica (idPessoaFisica, Pessoa_IdPessoa, cpf)
VALUES (1, 1, '11122244485');

INSERT INTO Pessoa (idPessoa, nome_razaoSocial, logradouro, cidade, estado, telefone, email)
VALUES
(NEXT VALUE FOR Seq_Id_TablePessoa, 'XP Sol', 'Rua 25', 'Curitiba', 'PR', '41988880011',
'matriz@xpsol.br');

INSERT INTO PessoaJuridica (idPessoaJuridica, Pessoa_IdPessoa, cnpj)
VALUES (1, 6, '0011222000135');
```

Inserção de Movimentação

```
INSERT INTO Movimento (idMovimento, Usuario_idUsuario, Produto_idProduto, Pessoa_idPessoa,
quantidade, tipo, valorUnitario)
VALUES
(1, 1, 2, 1, 100, 'E', 6.00),
(2, 1, 2, 3, 20, 'S', 3.00),
(3, 2, 1, 1, 100, 'S', 5.00),
(4, 2, 3, 6, 10, 'S', 6.00),
(5, 1, 1, 1, 100, 'E', 3.00);
```

Resultados da Consulta dos Dados Inseridos.

As janelas dos resultados das consultas dos dados inseridos, conforme solicitação na missão, são mostradas a seguir:

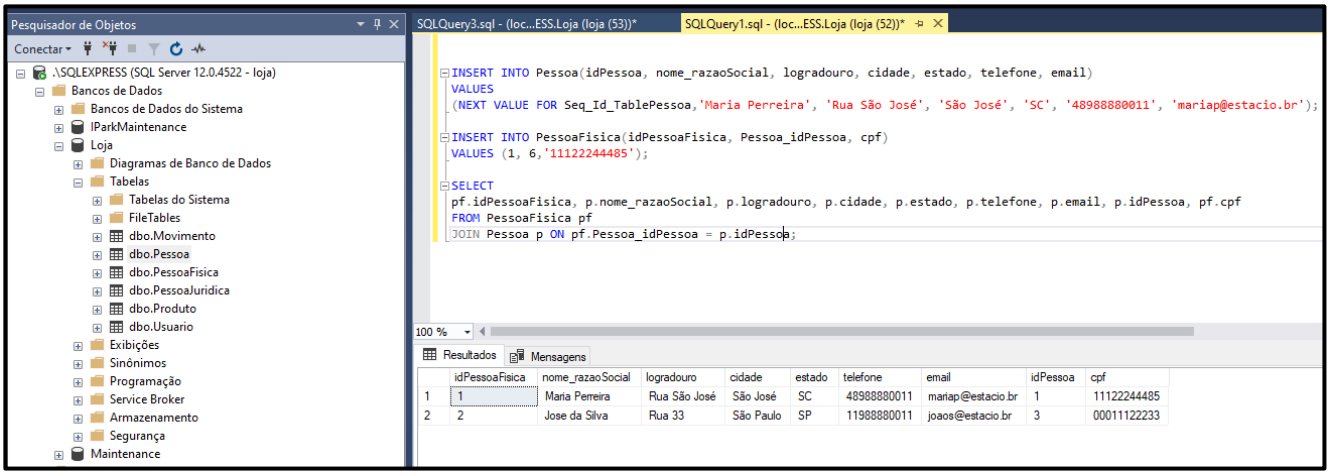

Figura 1 - Dados completos de pessoas físicas

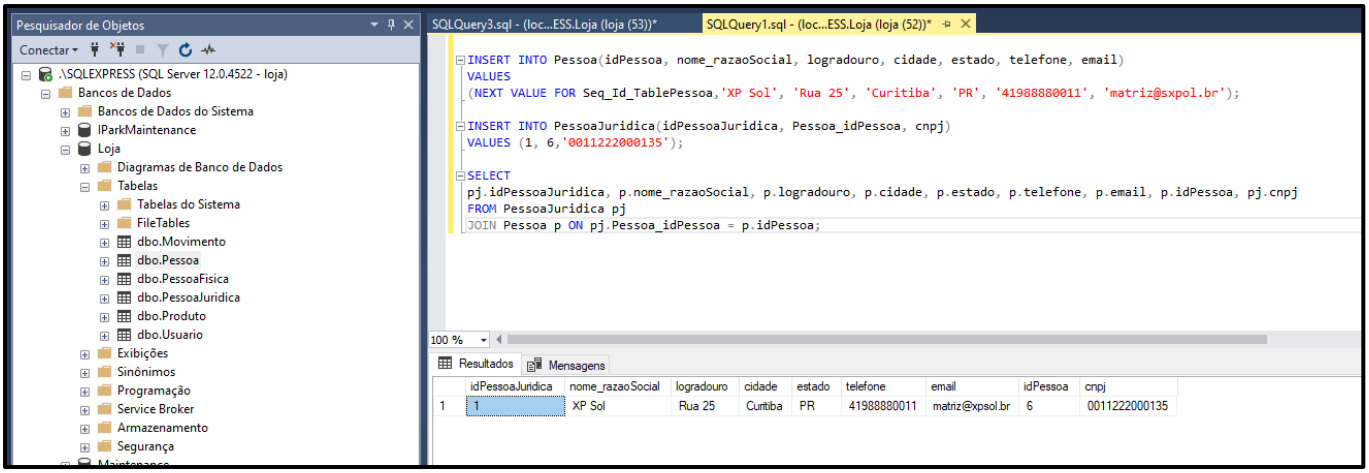

Figura 2 - Dados completos de pessoas jurídicas

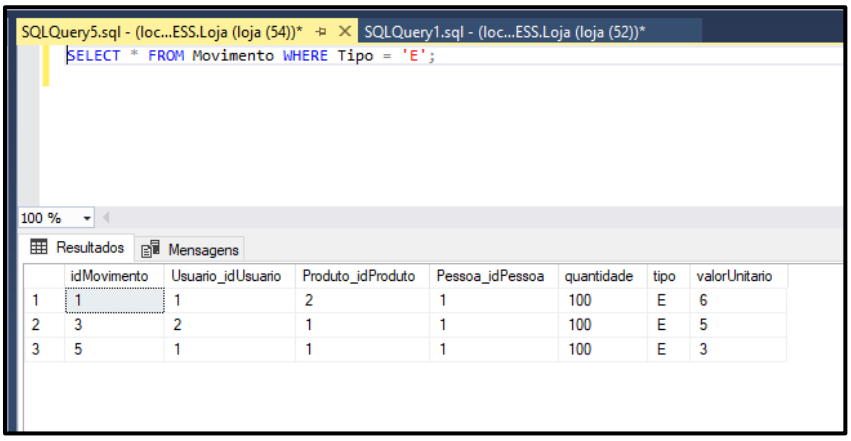

Figura 3 - Dados da movimentação de entrada

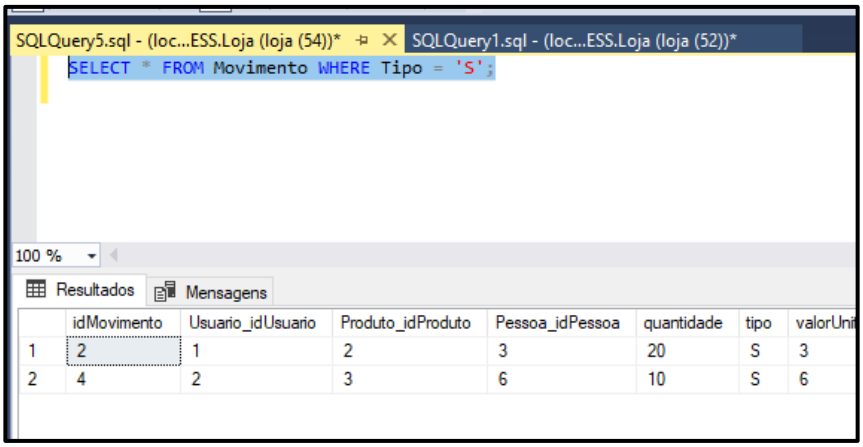

Figura 4 - Dados da movimentação de saída

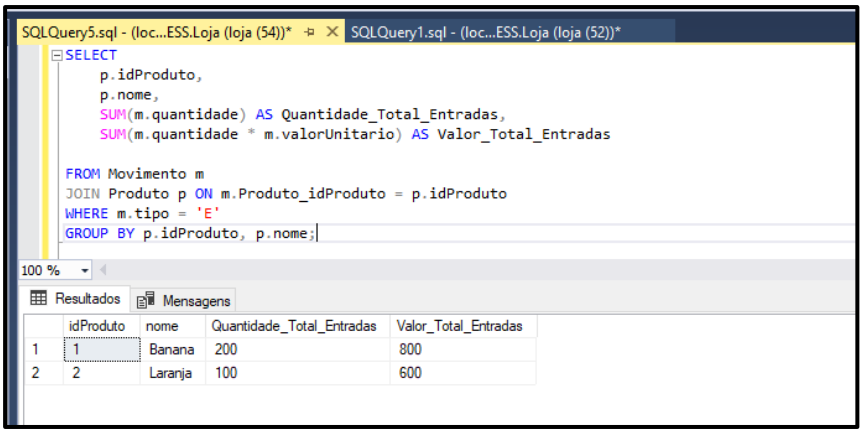

Figura 5 - Valor total das entradas agrupadas por produto

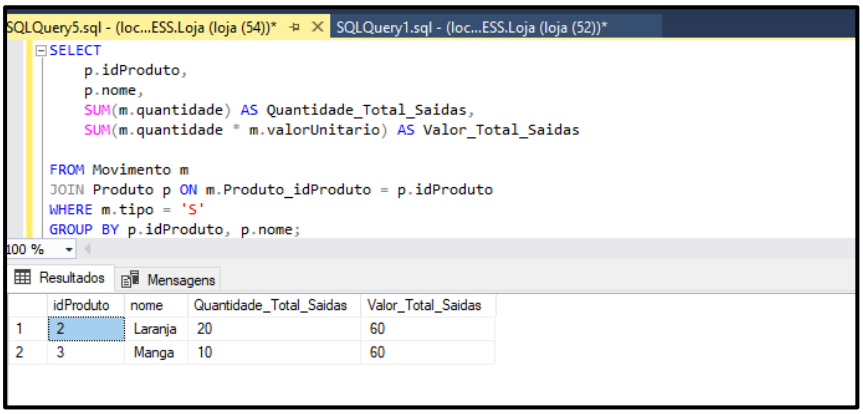

Figura 6 - Valor total das saídas agrupadas por produto

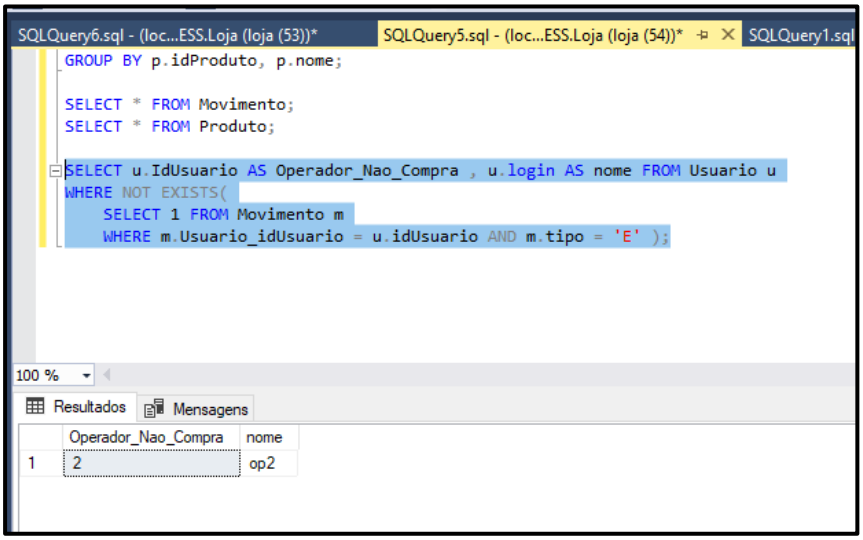

Figura 7 - Operador que não efetuou movimentações de entrada (compra)

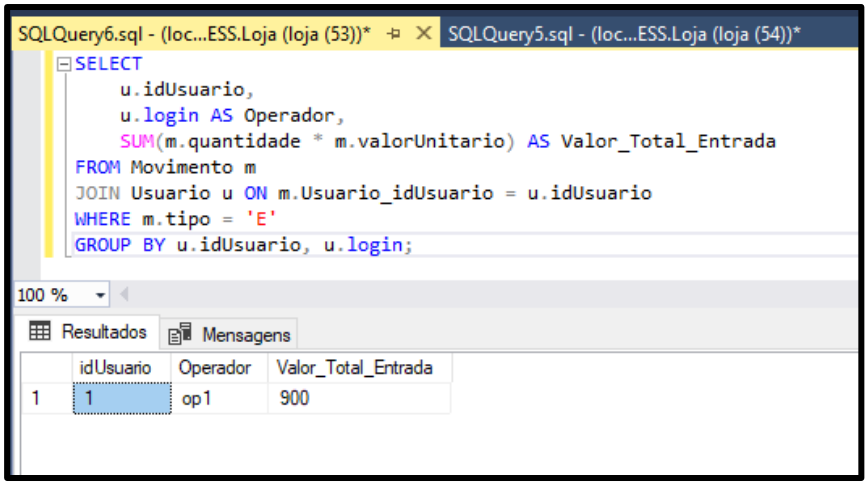

Figura 8 - Valor total de entrada, agrupado por operador

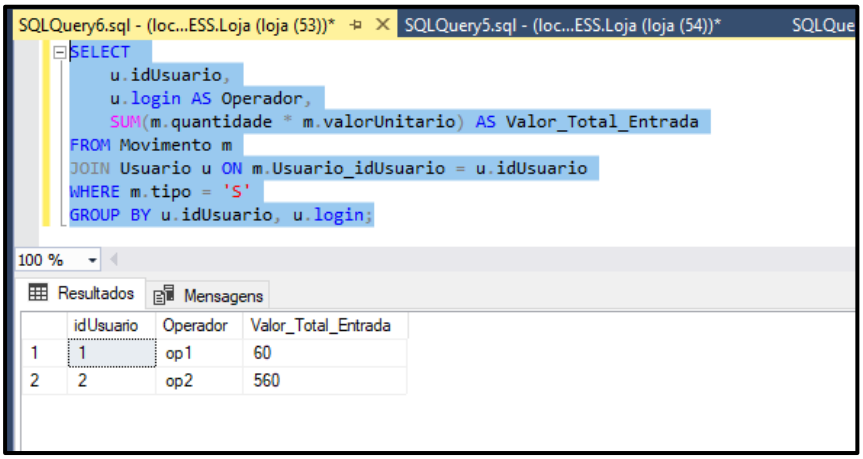

Figura 9 - Valor total de saída, agrupado por operador

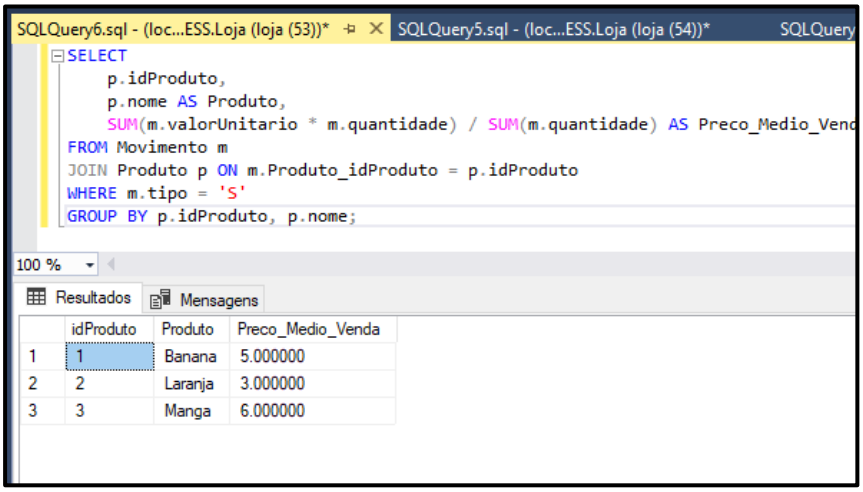

Figura 10 - Valor médio de venda por produto

Análise e Conclusão

A missão mostrou como se pode usar propriedades para a geração de números automáticos que podem ser usados para identificar campos nas tabelas. Foi feito uso da propriedade “Sequence” para gerar os “id’s” da tabela Pessoa. Esta propriedade permite a criação de valores, mesmo antes da

inserção na tabela, e pode ser usada em mais de uma tabela. Esta propriedade difere da “Identitiy”, que está restrita à tabela onde está definida.

Outra conclusão importante mostrada no projeto foi a importância das chaves estrangeiras, como elas são fundamentais para garantir a consistência e a integridade em um banco de dados relacional. Elas garantem que um valor em uma tabela corresponda a um valor específico em outra. Um exemplo pode ser visto na relação entra as tabelas “Movimento” e “Produto”. Uma transação na primeira tabela só pode apontar para um produto que realmente exista na segunda tabela.

Também, exploramos os fundamentos da linguagem SQL usada para a execução dos comandos no banco de dados. A linguagem combina operadores de Álgebra Relacional e do Cálculo Relacional. A Álgebra Relacional fornece a base teórica para a linguagem e tem os seguintes operadores: SELECT, WHERE, UNION, EXCEPT, JOIN e AS. O Cálculo Relacional, no entanto, é outro modelo de consulta que pode ser utilizado nas operações.

Finalmente, a linguagem SQL permite agrupar a consulta. Vimos no projeto, para isto, o uso da cláusula “GROUP BY” que permitiu, por exemplo, agrupar as vendas por operador. Um requisito obrigatório no uso dessa cláusula é que todo campo no SELECT que não está em uma função de agregação deve obrigatoriamente estar dentro do “GROUP BY”.

Armazenado no Github: https://github.com/altairsf/MissaoN2M3.git