

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:

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Pesquisador de Objetos' (Object Explorer) displays the database structure, including tables like 'Pessoa', 'PessoaFisica', and 'Movimento'. The main window shows a SQL query window with the following code:

```
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, 6, '11122244485');

SELECT
pf.idPessoaFisica, p.nome_razaoSocial, p.logradouro, p.cidade, p.estado, p.telefone, p.email, p.idPessoa, pf.cpf
FROM PessoaFisica pf
JOIN Pessoa p ON pf.Pessoa_idPessoa = p.idPessoa;
```

Below the query window, the 'Resultados' (Results) pane shows a grid with the following data:

	idPessoaFisica	nome_razaoSocial	logradouro	cidade	estado	telefone	email	idPessoa	cpf
1	1	Maria Pereira	Rua São José	São José	SC	48988880011	mariap@estacio.br	1	11122244485
2	2	Jose da Silva	Rua 33	São Paulo	SP	11988880011	joaos@estacio.br	3	00011122233

Figura 1 - Dados completos de pessoas físicas

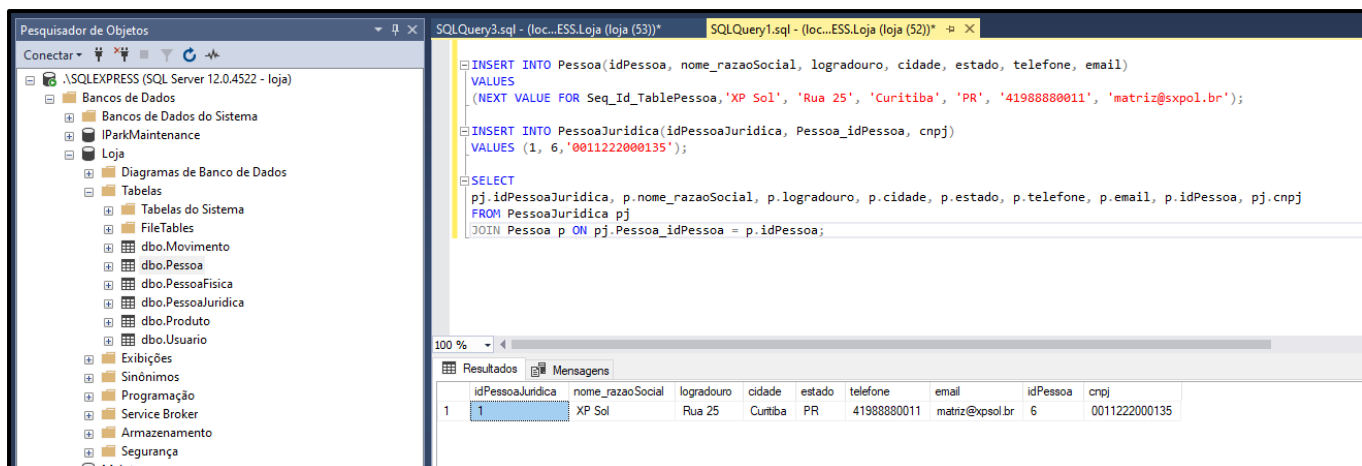


Figura 2 - Dados completos de pessoas jurídicas

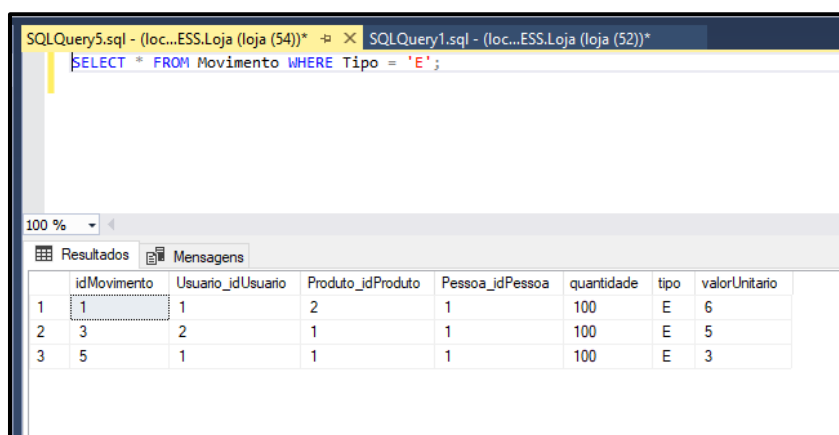


Figura 3 - Dados da movimentação de entrada

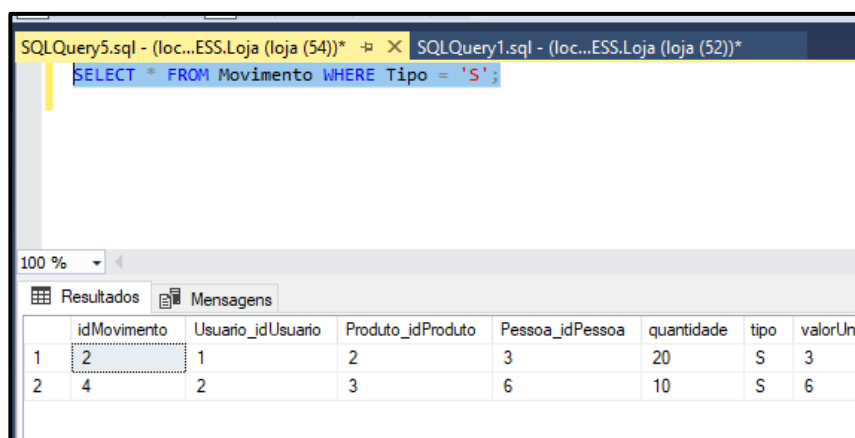


Figura 4 - Dados da movimentação de saída

SQLQuery5.sql - (loc...ESS.Loja (loja (54))) * SQLQuery1.sql - (loc...ESS.Loja (loja (52))) *

```

SELECT
    p.idProduto,
    p.nome,
    SUM(m.quantidade) AS Quantidade_Total_Entradas,
    SUM(m.quantidade * m.valorUnitario) AS Valor_Total_Entradas

FROM Movimento m
JOIN Produto p ON m.Produto_idProduto = p.idProduto
WHERE m.tipo = 'E'
GROUP BY p.idProduto, p.nome;

```

100 %

Resultados Mensagens

	idProduto	nome	Quantidade_Total_Entradas	Valor_Total_Entradas
1	1	Banana	200	800
2	2	Laranja	100	600

Figura 5 - Valor total das entradas agrupadas por produto

SQLQuery5.sql - (loc...ESS.Loja (loja (54))) * SQLQuery1.sql - (loc...ESS.Loja (loja (52))) *

```

SELECT
    p.idProduto,
    p.nome,
    SUM(m.quantidade) AS Quantidade_Total_Saidas,
    SUM(m.quantidade * m.valorUnitario) AS Valor_Total_Saidas

FROM Movimento m
JOIN Produto p ON m.Produto_idProduto = p.idProduto
WHERE m.tipo = 'S'
GROUP BY p.idProduto, p.nome;

```

100 %

Resultados Mensagens

	idProduto	nome	Quantidade_Total_Saidas	Valor_Total_Saidas
1	2	Laranja	20	60
2	3	Manga	10	60

Figura 6 - Valor total das saídas agrupadas por produto

SQLQuery6.sql - (loc...ESS.Loja (loja (53))) * SQLQuery5.sql - (loc...ESS.Loja (loja (54))) * SQLQuery1.sql

```

GROUP BY p.idProduto, p.nome;

SELECT * FROM Movimento;
SELECT * FROM Produto;

SELECT u.IdUsuario AS Operador_Nao_Compra , u.login AS nome FROM Usuario u
WHERE NOT EXISTS(
    SELECT 1 FROM Movimento m
    WHERE m.Usuario_idUsuario = u.idUsuario AND m.tipo = 'E' );

```

100 %

Resultados Mensagens

	Operador_Nao_Compra	nome
1	2	op2

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

SQLQuery6.sql - (loc...ESS.Loja (loja (53))) * SQLQuery5.sql - (loc...ESS.Loja (loja (54))) *

```

SELECT
    u.idUsuario,
    u.login AS Operador,
    SUM(m.quantidade * m.valorUnitario) AS Valor_Total_Entrada
FROM Movimento m
JOIN Usuario u ON m.Usuario_idUsuario = u.idUsuario
WHERE m.tipo = 'E'
GROUP BY u.idUsuario, u.login;

```

100 %

Resultados Mensagens

	idUsuario	Operador	Valor_Total_Entrada
1	1	op1	900

Figura 8 - Valor total de entrada, agrupado por operador

SQLQuery6.sql - (loc...ESS.Loja (loja (53))) * SQLQuery5.sql - (loc...ESS.Loja (loja (54))) * SQLQue

```

SELECT
    u.idUsuario,
    u.login AS Operador,
    SUM(m.quantidade * m.valorUnitario) AS Valor_Total_Entrada
FROM Movimento m
JOIN Usuario u ON m.Usuario_idUsuario = u.idUsuario
WHERE m.tipo = 'S'
GROUP BY u.idUsuario, u.login;

```

100 %

Resultados Mensagens

	idUsuario	Operador	Valor_Total_Entrada
1	1	op1	60
2	2	op2	560

Figura 9 - Valor total de saída, agrupado por operador

SQLQuery6.sql - (loc...ESS.Loja (loja (53))) * SQLQuery5.sql - (loc...ESS.Loja (loja (54))) * SQLQuery

```

SELECT
    p.idProduto,
    p.nome AS Produto,
    SUM(m.valorUnitario * m.quantidade) / SUM(m.quantidade) AS Preco_Medio_Venda
FROM Movimento m
JOIN Produto p ON m.Produto_idProduto = p.idProduto
WHERE m.tipo = 'S'
GROUP BY p.idProduto, p.nome;

```

100 %

Resultados Mensagens

	idProduto	Produto	Preco_Medio_Venda
1	1	Banana	5.000000
2	2	Laranja	3.000000
3	3	Manga	6.000000

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 “Identity”, 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 entre 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>