Campus: Polo n° 3366 Centro - Garopaba - SC
Curso: Tecnólogo em Desenvolvimento Full Stack
Disciplina: Vamos manter as informações?
Turma: 9001
Nome do Aluno: Luiz Evaldo Pereira
Endereço:

1. **Título**
   Alimentando a base

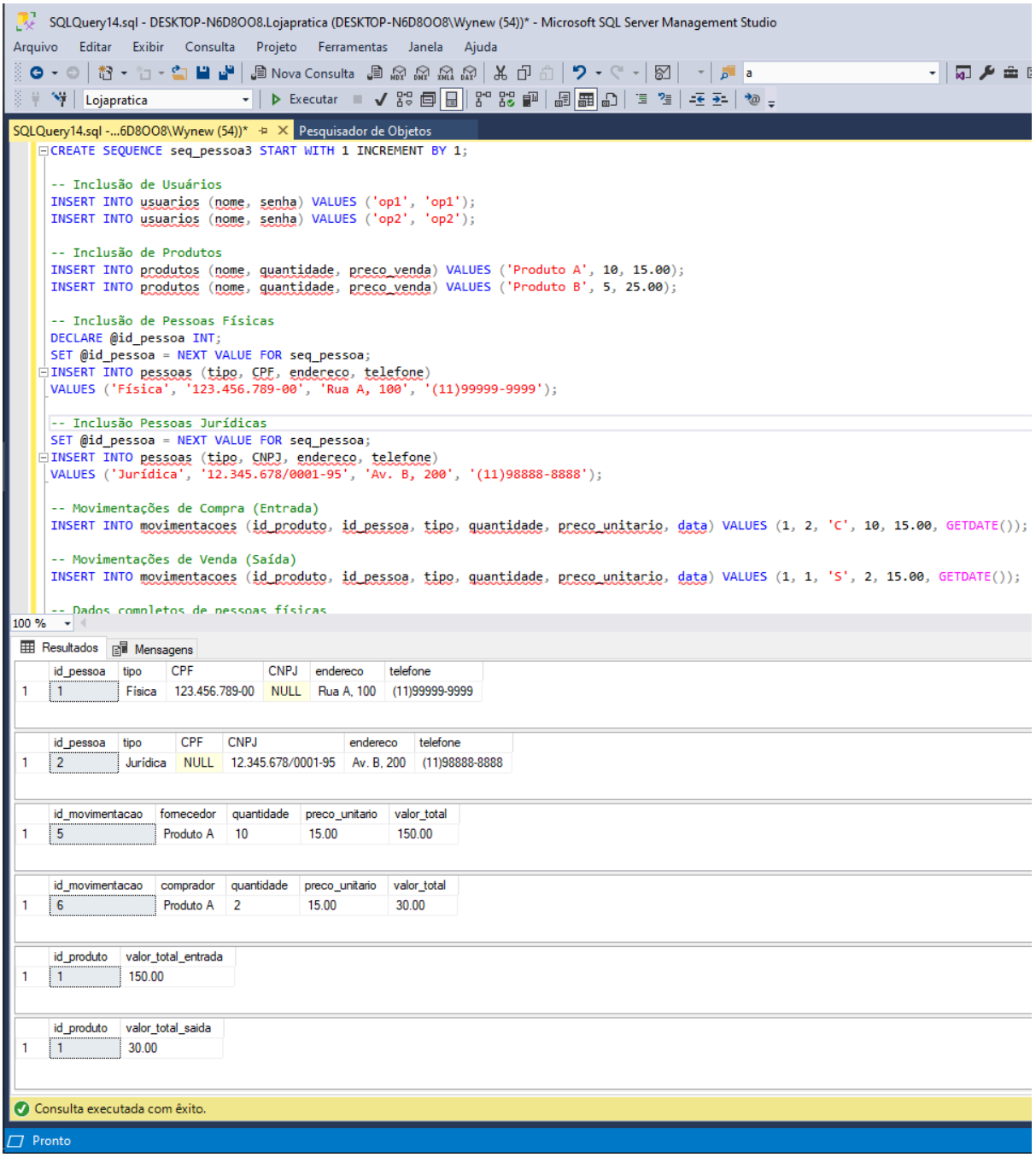
2. **Objetivo**
   O presente trabalho busca implementar um sistema básico de gestão de estoque e movimentações comerciais utilizando banco de dados relacional. A sequência (`seq_pessoa3`) automatiza a geração de IDs para registros de pessoas, garantindo unicidade sem conflitos manuais.

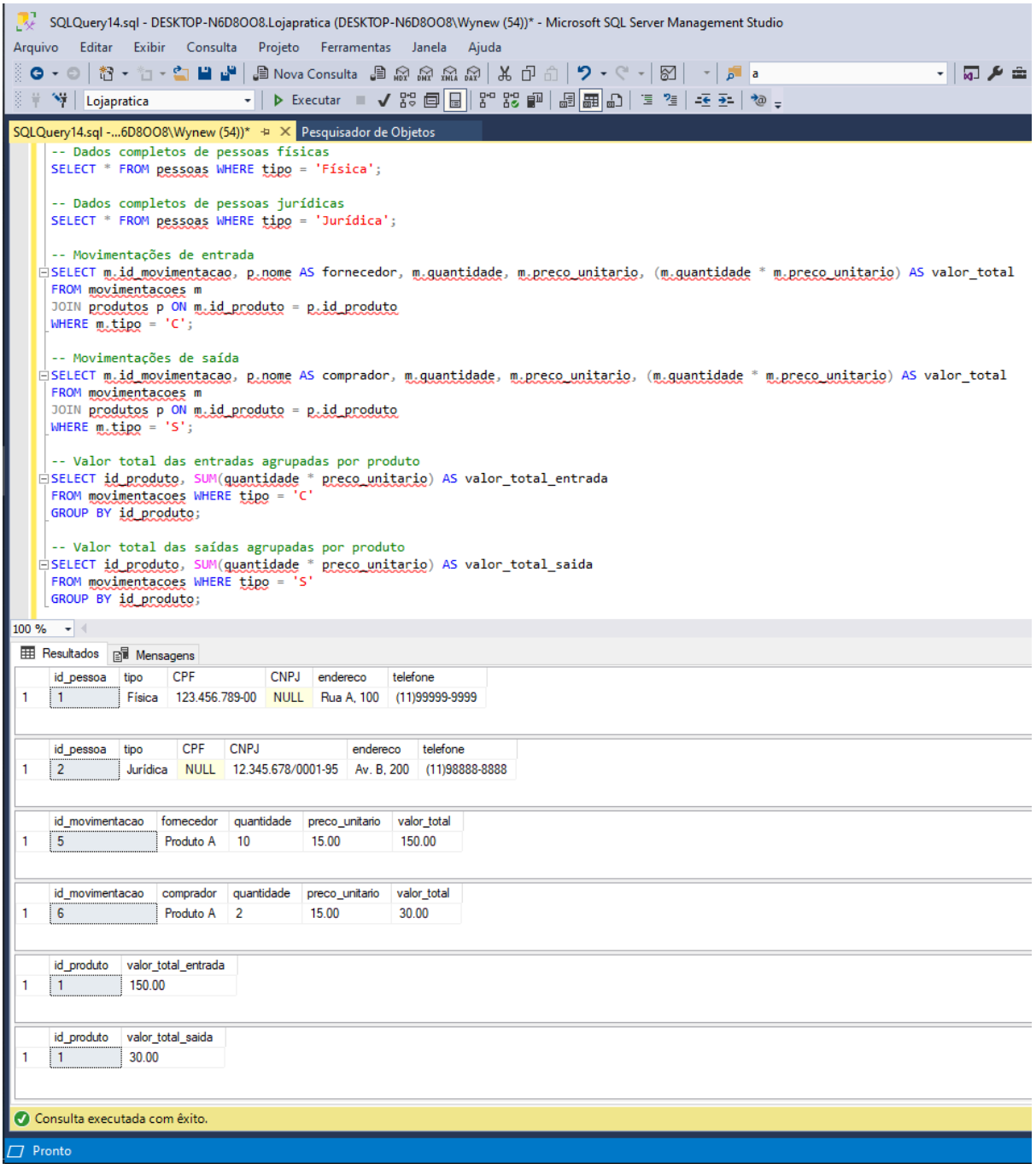
3. **Códigos**
   3.1 Modelo sequence

```
CREATE SEQUENCE seq_pessoa3 START WITH 1 INCREMENT BY 1;

-- Inclusão de usuários
INSERT INTO usuarios (nome, senha) VALUES ('op1', 'op1');
INSERT INTO usuarios (nome, senha) VALUES ('op2', 'op2');

-- Inclusão de Produtos
INSERT INTO produtos (nome, quantidade, preco_venda) VALUES ('Produto A', 10, 15.00);
INSERT INTO produtos (nome, quantidade, preco_venda) VALUES ('Produto B', 5, 25.00);

-- Inclusão de Pessoas físicas
DECLARE @id_pessoa INT;
SET @id_pessoa = NEXT VALUE FOR seq_pessoa;
INSERT INTO pessoas (tipo, CPF, endereco, telefone)
VALUES ('Física', '12.345.678/0001-95', 'Av. B, 200', '(11)99999-99999');

-- Inclusão de Pessoas Jurídicas
SET @id_pessoa = NEXT VALUE FOR seq_pessoa;
INSERT INTO pessoas (tipo, CNPJ, endereco, telefone)
VALUES ('Jurídica', '12.345.678/0001-95', 'Av. B, 200', '(11)98888,8888');

INSERT INTO movimentacoes (id_produto, id_pessoa, tipo, quantidade, preco_unitario, data_movimentacao)
VALUES (1, 2, 'C', 10, 15.00, GETDATE());
```

```
INSERT INTO movimentacoes (id_produto, id_pessoa, tipo, quantidade, preco_unitario, data_movimentacao)
VALUES (1, 1, 'S', 2, 15.00, GETDATE());

-- Consultas corrigidas com nomes de colunas corretos

-- Dados completos de pessoas físicas
SELECT * FROM pessoas WHERE tipo = 'Física';

-- Dados completos de pessoas jurídicas
SELECT * FROM pessoas WHERE tipo = 'Jurídica';

-- Movimentações de entrada com junção correta
SELECT m.id_movimentacao, p.nome AS produto, m.quantidade, m.preco_unitario,
       (m.quantidade * m.preco_unitario) AS valor_total
FROM movimentacoes m
JOIN produtos p ON m.id_produto = p.id_produto
WHERE m.tipo = 'C';

-- Movimentações de saída com junção correta
SELECT m.id_movimentacao, p.nome AS produto, m.quantidade, m.preco_unitario,
       (m.quantidade * m.preco_unitario) AS valor_total
FROM movimentacoes m
JOIN produtos p ON m.id_produto = p.id_produto
WHERE m.tipo = 'S';

-- Valor total das entradas agrupadas por produto
SELECT id_produto, SUM(quantidade * preco_unitario) AS valor_total_entrada
FROM movimentacoes
WHERE tipo = 'C'
GROUP BY id_produto;

-- Valor total das saídas agrupadas por produto
SELECT id_produto, SUM(quantidade * preco_unitario) AS valor_total_saida
FROM movimentacoes
WHERE tipo = 'S'
GROUP BY id_produto;
```

4. **Resultados**


SQLQuery14.sql - DESKTOP-N6D8OO8.Lojapratica (DESKTOP-N6D8OO8\Wynew (54))* - Microsoft SQL Server Management Studio
Arquivo Editar Exibir Consulta Projeto Ferramentas Janela Ajuda
Nova Consulta
Lojapratica
Executar
SQLQuery14.sql -...6D8OO8\Wynew (54))*
Pesquisador de Objetos
CREATE SEQUENCE seq_pessoa3 START WITH 1 INCREMENT BY 1;
-- Inclusão de Usuários
INSERT INTO usuarios (nome, senha) VALUES ('op1', 'op1');
INSERT INTO usuarios (nome, senha) VALUES ('op2', 'op2');
-- Inclusão de Produtos
INSERT INTO produtos (nome, quantidade, preco_venda) VALUES ('Produto A', 10, 15.00);
INSERT INTO produtos (nome, quantidade, preco_venda) VALUES ('Produto B', 5, 25.00);
-- Inclusão de Pessoas Físicas
DECLARE @id_pessoa INT;
SET @id_pessoa = NEXT VALUE FOR seq_pessoa;
INSERT INTO pessoas (tipo, CPF, endereco, telefone)
VALUES ('Física', '123.456.789-00', 'Rua A, 100', '(11)99999-9999');
-- Inclusão Pessoas Jurídicas
SET @id_pessoa = NEXT VALUE FOR seq_pessoa;
INSERT INTO pessoas (tipo, CNPJ, endereco, telefone)
VALUES ('Jurídica', '12.345.678/0001-95', 'Av. B, 200', '(11)98888-8888');
-- Movimentações de Compra (Entrada)
INSERT INTO movimentacoes (id_produto, id_pessoa, tipo, quantidade, preco_unitario, data) VALUES (1, 2, 'C', 10, 15.00, GETDATE());
-- Movimentações de Venda (Saída)
INSERT INTO movimentacoes (id_produto, id_pessoa, tipo, quantidade, preco_unitario, data) VALUES (1, 1, 'S', 2, 15.00, GETDATE());
-- Dados completos de pessoas físicas
100 %
Resultados
Mensagens
id_pessoa tipo CPF CNPJ endereco telefone
1 1 Física 123.456.789-00 NULL Rua A, 100 (11)99999-9999
id_pessoa tipo CPF CNPJ endereco telefone
1 2 Jurídica NULL 12.345.678/0001-95 Av. B, 200 (11)98888-8888
id_movimentacao fornecedor quantidade preco_unitario valor_total
1 5 Produto A 10 15.00 150.00
id_movimentacao comprador quantidade preco_unitario valor_total
1 6 Produto A 2 15.00 30.00
id_produto valor_total_entrada
1 1 150.00
id_produto valor_total_saida
1 1 30.00
Consulta executada com êxito.
Pronto



5. Análise e conclusão:

a) Quais as diferenças no uso de *sequence* e *identify?*

- **Sequence**: Gera números únicos e incrementais, que podem ser consultados e têm maior flexibilidade.
- **Identity**: Números são gerados automaticamente durante a inserção de linhas, sendo menos flexível que a *sequence*.

b) Qual a importância das chaves estrangeiras para a consistência do banco?

As chaves estrangeiras garantem a integridade referencial no banco de dados. Elas asseguram que os relacionamentos entre as tabelas sejam mantidos de maneira consistente, evitando dados órfãos.

c) Quais operadores do SQL pertencem à álgebra relacional e quais são definidos no cálculo relacional?

- **Álgebra Relacional**: SELECT, PROJECT, JOIN, UNION.
- **Cálculo Relacional**: WHERE, HAVING.

d) Como é feito o agrupamento em consultas, e qual requisito é obrigatório?

O agrupamento é feito utilizando a cláusula `GROUP BY`. Um requisito obrigatório é que todas as colunas não agregadas na seleção devem estar listadas na cláusula `GROUP BY`.