Campus: 1197 - POLO CENTRO - SÃO LOURENÇO DA MATA - PE
Curso: Desenvolvimento Full Stack - Graduação Tecnóloga
Disciplina: RPG0014 - Iniciando o caminho pelo Java
Turma: 9001 Semestre: 2024.1 Matrícula: 2023.01.53256-6
Aluno: Gilvan Pereira de Oliveira
Repositório GitHub: GilvanPOliveira (Gilvan P Oliveira) (github.com)

Relatório discente de acompanhamento

1º Procedimento | Criando o Banco de Dados

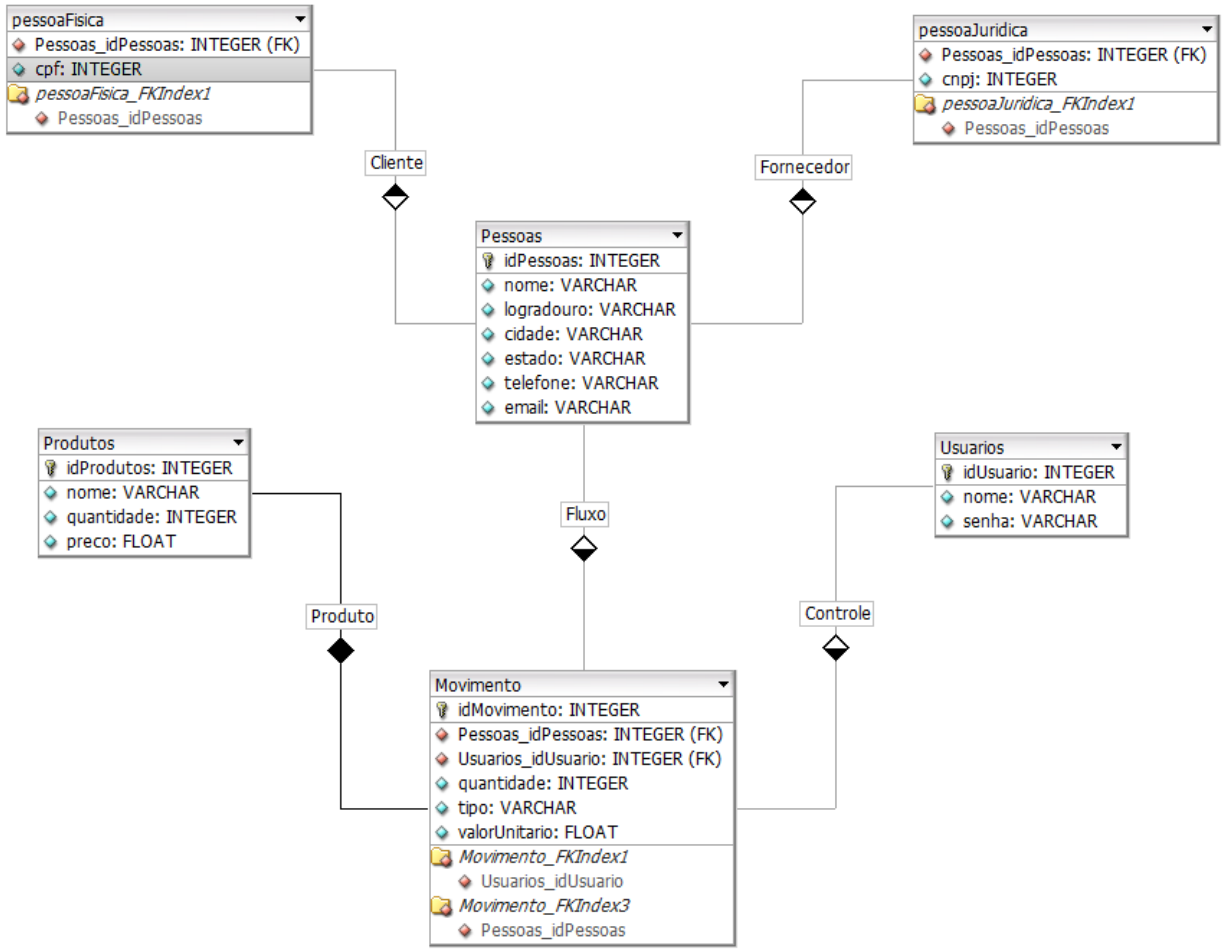
Objetivo da prática:

Desenvolver um banco de dados do zero, identificando os tipos de relacionamentos necessários por meio da modelagem e implementação, visando a criação de um sistema de vendas, englobando usuários, clientes, produtos e transações de compra e venda.

Prática:

- Utilizar o DBDesigner Fork para definir o modelo de dados do banco e seus relacionamentos;
- Utilizar o SQL Management Studio para criar o banco de dados.

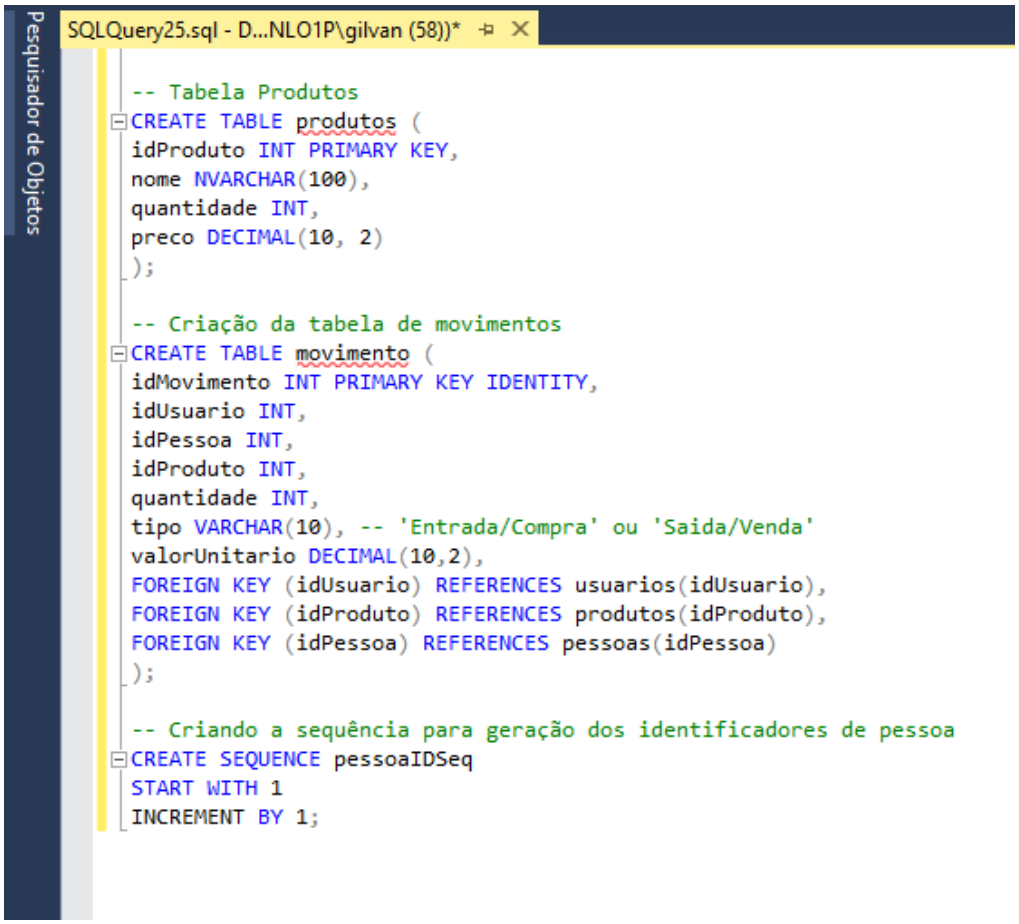
Códigos e resultados obtidos:



Pesquisador de Objetos

SQLQuery25.sql - D...NLO1P\gilvan (58))*

```sql
-- Criando o login "Loja" com senha "loja" e concedendo permissões necessárias
USE master;
GO
CREATE LOGIN Loja WITH PASSWORD = 'loja';
GO
ALTER SERVER ROLE sysadmin ADD MEMBER Loja;
GO

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

-- Criando as tabelas
-- Tabela Usuários
CREATE TABLE usuarios (
idUsuario INT PRIMARY KEY,
nome NVARCHAR(100),
senha NVARCHAR(50)
);

-- Tabela Pessoas
CREATE TABLE pessoas (
idPessoa INT PRIMARY KEY IDENTITY,
nome NVARCHAR(100),
logradouro NVARCHAR(200),
cidade NVARCHAR(50),
estado NVARCHAR(50),
telefone NVARCHAR(50),
email NVARCHAR(50)
);

-- Tabela PessoaFisica
CREATE TABLE pessoaFisica (
idPessoa INT PRIMARY KEY,
cpf NVARCHAR(14),
FOREIGN KEY (idPessoa) REFERENCES pessoas(idPessoa)
);

-- Tabela PessoaJuridica
CREATE TABLE pessoaJuridica (
idPessoa INT PRIMARY KEY,
cnpj NVARCHAR(18),
FOREIGN KEY (idPessoa) REFERENCES pessoas(idPessoa)
);
```



Conclusão:

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

Para 1x1, uma tabela contém uma chave estrangeira que referência a chave primária de outra tabela. Para 1xN, uma tabela contém uma chave estrangeira que referência a chave primária de outra tabela, permitindo que um registro se relacione com vários registros em outra tabela. E para NxN, é necessária uma tabela intermediária que contém pares de chaves estrangeiras para relacionar registros entre duas tabelas.

b) Que tipo de relacionamento deve ser utilizado para representar o uso de herança em bancos de dados relacionais?

Para representar herança em bancos de dados relacionais, usa-se o modelo de tabela única (ou tabela por classe). Uma tabela central contém atributos comuns a todas as entidades, enquanto tabelas secundárias (ou por subclasse) têm atributos específicos. Essas tabelas estão conectadas por chaves primárias e estrangeiras, garantindo integridade e coesão.

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

Gerenciando o banco de dados através de sua interface gráfica intuitiva, integrando outras ferramentas da Microsoft como visual Studio, possui um editor SQL robusto, dentre outras qualidades

2º Procedimento | Alimentando a Base

Objetivo da prática:

Utilizar o SQL Server Management Studio para operar em um banco de dados, abrangendo desde inserção de dados e criação de entidades até consultas mais complexas, alimentando tabelas e executando scripts.

Prática:

- Inserir dados básicos do sistema no banco de dados;
- Criar movimentações na base de dados;
- Efetuar consultas sobre os dados inseridos.

Códigos e resultados obtidos:

SQLQuery25.sql - D...NLO1P\gilvan (58))*

```sql
-- Inserção de usuários
INSERT INTO usuarios (idUsuario, nome, senha) VALUES
(1, 'op1', 'op1'),
(2, 'op2', 'op2');

-- Inserção de produtos
INSERT INTO produtos (idProduto, nome, quantidade, preco) VALUES
(1, 'Banana', 100, 5.00),
(3, 'Laranja', 500, 2.00),
(4, 'Manga', 800, 4.00);

-- Inserção de pessoas físicas
INSERT INTO pessoas (nome, logradouro, cidade, estado, telefone, email) VALUES
('Joao', 'Rua 12, casa 3, Quitanda', 'Riacho do Sul', 'PA', '1111-1111', 'joao@riacho.com');
INSERT INTO pessoaFisica (idPessoa, cpf) VALUES
(SCOPE_IDENTITY(), '11111111111');

-- Inserção de pessoas jurídicas
INSERT INTO pessoas (nome, logradouro, cidade, estado, telefone, email) VALUES
('JJC', 'Rua 11, Centro', 'Riacho do Norte', 'PA', '1212-1212', 'jjc@riacho.com');
INSERT INTO pessoaJuridica (idPessoa, cnpj) VALUES
(SCOPE_IDENTITY(), '22222222222222');

-- Inserção de movimentações
INSERT INTO movimento (idUsuario, idPessoa, idProduto, quantidade, tipo, valorUnitario) VALUES
(1, 1, 1, 20, 'S', 4.00),  -- Venda de 20 bananas pelo usuário 1 à pessoa física 1
(1, 1, 3, 15, 'S', 2.00),  -- Venda de 15 laranjas pelo usuário 1 à pessoa física 1
(2, 2, 3, 10, 'S', 3.00),  -- Venda de 10 laranjas pelo usuário 2 à pessoa jurídica 2
(1, 2, 3, 15, 'E', 5.00),  -- Compra de 15 laranjas pelo usuário 1 da pessoa jurídica 2
(1, 2, 4, 20, 'E', 4.00);  -- Compra de 20 mangas pelo usuário 1 da pessoa jurídica 2
```

DESKTOP-6TNLO1P.Loja - dbo.usuarios

idUsuario	nome	senha
1	op1	op1
2	op2	op2
NULL	*NULL*	*NULL*

Pesquisador de Objetos

DESKTOP-6TNLO1P.Loja - dbo.produtos | DESKTOP-6TNLO1P.Loja - dbo.usuarios

	idProduto	nome	quantidade	preco
▶	1	Banana	100	5,00
	3	Laranja	500	2,00
	4	Manga	800	4,00
*	NULL	NULL	NULL	NULL

Pesquisador de Objetos

DESKTOP-6TNLO1P.L...dbo.pessoaFisica | DES

	idPessoa	cpf
▶	1	11111111111
*	NULL	NULL

Pesquisador de Objetos

DESKTOP-6TNLO1P.L...bo.pessoaJuridica | DESK

	idPessoa	cnpj
▶	2	22222222222222
*	NULL	NULL

Pesquisador de Objetos

DESKTOP-6TNLO1P.Loja - dbo.pessoas | DESKTOP-6TNLO1P.L...bo.pessoaJuridica | DESKTOP-6TNLO1P.L...dbo.pessoaFisica

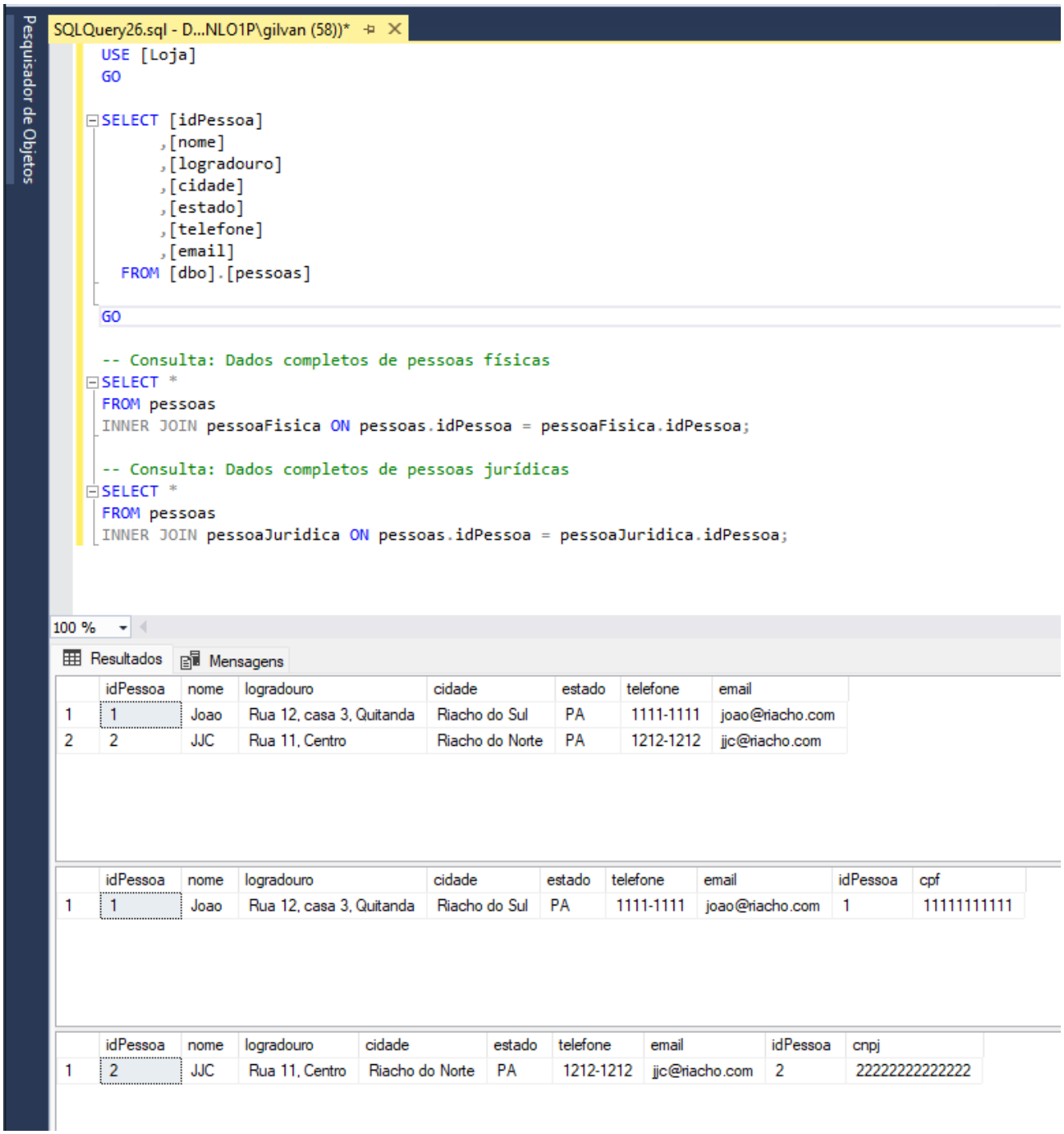
	idPessoa	nome	logradouro	cidade	estado	telefone	email
▶	1	Joao	Rua 12, casa 3, Quitanda	Riacho do Sul	PA	1111-1111	joao@riacho.com
	2	JJC	Rua 11, Centro	Riacho do Norte	PA	1212-1212	jjc@riacho.com
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Pesquisador de Objetos

DESKTOP-6TNLO1P....- dbo.movimento

	idMovimento	idUsuario	idPessoa	idProduto	quantidade	tipo	valorUnitario
▶	2	1	1	1	20	S	4,00
	3	1	1	3	15	S	2,00
	4	2	2	3	10	S	3,00
	5	1	2	3	15	E	5,00
	6	1	2	4	20	E	4,00
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Consultas:


Pesquisador de Objetos
SQLQuery26.sql - D...NLO1P\gilvan (58))*
USE [Loja]
GO
SELECT [idPessoa]
,[nome]
,[logradouro]
,[cidade]
,[estado]
,[telefone]
,[email]
FROM [dbo].[pessoas]
GO
-- Consulta: Dados completos de pessoas físicas
SELECT *
FROM pessoas
INNER JOIN pessoaFisica ON pessoas.idPessoa = pessoaFisica.idPessoa;
-- Consulta: Dados completos de pessoas jurídicas
SELECT *
FROM pessoas
INNER JOIN pessoaJuridica ON pessoas.idPessoa = pessoaJuridica.idPessoa;
100 %
Resultados
Mensagens
idPessoa nome logradouro cidade estado telefone email
1 1 Joao Rua 12, casa 3, Quitanda Riacho do Sul PA 1111-1111 joao@riacho.com
2 2 JJC Rua 11, Centro Riacho do Norte PA 1212-1212 jjc@riacho.com
idPessoa nome logradouro cidade estado telefone email idPessoa cpf
1 1 Joao Rua 12, casa 3, Quitanda Riacho do Sul PA 1111-1111 joao@riacho.com 1 11111111111
idPessoa nome logradouro cidade estado telefone email idPessoa cnpj
1 2 JJC Rua 11, Centro Riacho do Norte PA 1212-1212 jjc@riacho.com 2 22222222222222

Pesquisador de Objetos

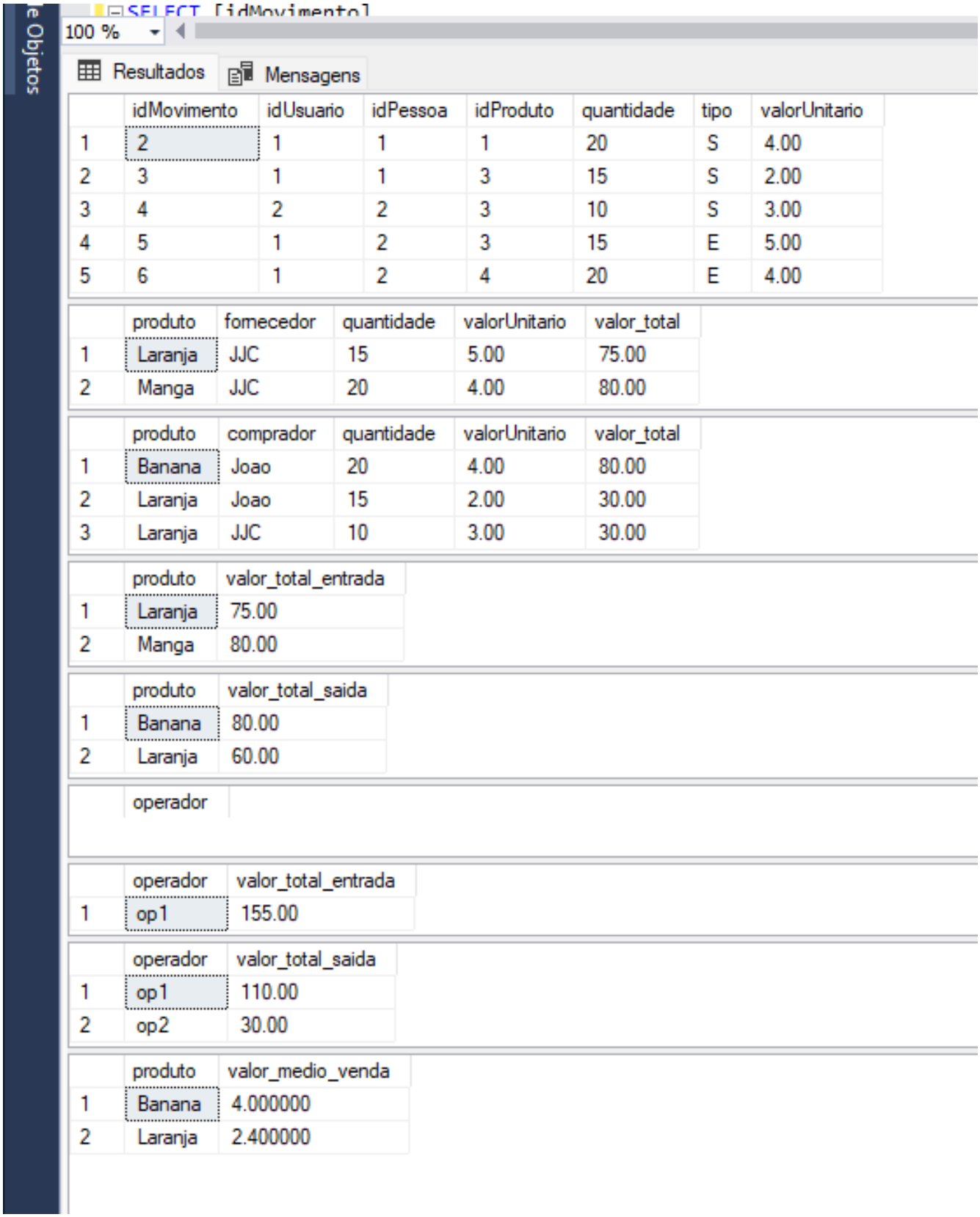
SQLQuery27.sql - D...NLO1P\gilvan (58))*

```sql
USE [Loja]
GO

SELECT [idMovimento]
      ,[idUsuario]
      ,[idPessoa]
      ,[idProduto]
      ,[quantidade]
      ,[tipo]
      ,[valorUnitario]
  FROM [dbo].[movimento]

GO

-- Consulta: Movimentações de entrada, com produto, fornecedor, quantidade, preço unitário e valor total
SELECT p.nome AS produto, pf.nome AS fornecedor, m.quantidade, m.valorUnitario, (m.quantidade * m.valorUnitario) AS valor_total
FROM movimento m
JOIN produtos p ON m.idProduto = p.idProduto
JOIN pessoas pf ON m.idPessoa = pf.idPessoa
WHERE m.tipo = 'E';

-- Consulta: Movimentações de saída, com produto, comprador, quantidade, preço unitário e valor total
SELECT p.nome AS produto, pc.nome AS comprador, m.quantidade, m.valorUnitario, (m.quantidade * m.valorUnitario) AS valor_total
FROM movimento m
JOIN produtos p ON m.idProduto = p.idProduto
JOIN pessoas pc ON m.idPessoa = pc.idPessoa
WHERE m.tipo = 'S';

-- Consulta: Valor total das entradas agrupadas por produto
SELECT p.nome AS produto, SUM(m.quantidade * m.valorUnitario) AS valor_total_entrada
FROM movimento m
JOIN produtos p ON m.idProduto = p.idProduto
WHERE m.tipo = 'E'
GROUP BY p.nome;

-- Consulta: Valor total das saídas agrupadas por produto
SELECT p.nome AS produto, SUM(m.quantidade * m.valorUnitario) AS valor_total_saida
FROM movimento m
JOIN produtos p ON m.idProduto = p.idProduto
WHERE m.tipo = 'S'
GROUP BY p.nome;

-- Consulta: Operadores que não efetuaram movimentações de entrada (compra)
SELECT u.nome AS operador
FROM usuarios u
LEFT JOIN movimento m ON u.idUsuario = m.idUsuario
WHERE m.idMovimento IS NULL;
```

Pesquisador de Objetos

SQLQuery27.sql - D...NLO1P\gilvan (58))*

```sql
-- Consulta: Valor total de entrada, agrupado por operador
SELECT u.nome AS operador, SUM(m.quantidade * m.valorUnitario) AS valor_total_entrada
FROM movimento m
JOIN usuarios u ON m.idUsuario = u.idUsuario
WHERE m.tipo = 'E'
GROUP BY u.nome;

-- Consulta: Valor total de saída, agrupado por operador
SELECT u.nome AS operador, SUM(m.quantidade * m.valorUnitario) AS valor_total_saida
FROM movimento m
JOIN usuarios u ON m.idUsuario = u.idUsuario
WHERE m.tipo = 'S'
GROUP BY u.nome;

-- Consulta: Valor médio de venda por produto, utilizando média ponderada
SELECT p.nome AS produto, SUM(m.quantidade * m.valorUnitario) / SUM(m.quantidade) AS valor_medio_venda
FROM movimento m
JOIN produtos p ON m.idProduto = p.idProduto
WHERE m.tipo = 'S'
GROUP BY p.nome;
```


SELECT [idMovimento]

100 %

Resultados | Mensagens

	idMovimento	idUsuario	idPessoa	idProduto	quantidade	tipo	valorUnitario
1	2	1	1	1	20	S	4.00
2	3	1	1	3	15	S	2.00
3	4	2	2	3	10	S	3.00
4	5	1	2	3	15	E	5.00
5	6	1	2	4	20	E	4.00

	produto	fornecedor	quantidade	valorUnitario	valor_total
1	Laranja	JJC	15	5.00	75.00
2	Manga	JJC	20	4.00	80.00

	produto	comprador	quantidade	valorUnitario	valor_total
1	Banana	Joao	20	4.00	80.00
2	Laranja	Joao	15	2.00	30.00
3	Laranja	JJC	10	3.00	30.00

	produto	valor_total_entrada
1	Laranja	75.00
2	Manga	80.00

	produto	valor_total_saida
1	Banana	80.00
2	Laranja	60.00

	operador

	operador	valor_total_entrada
1	op1	155.00

	operador	valor_total_saida
1	op1	110.00
2	op2	30.00

	produto	valor_medio_venda
1	Banana	4.000000
2	Laranja	2.400000

Conclusão:

a) Quais as diferenças no uso de sequence e identity?
   Sequence é usada para gerar valores únicos automaticamente em uma coluna, enquanto **identity** é um atributo de coluna usados para gerar automaticamente valores únicos para nova linha inserida.

b) Qual a importância das chaves estrangerias para a consistência do banco?
   Chaves estrangeiras garantem a integridade referencial entre tabelas, mantendo a consistência dos dados, evitando referências a registros inexistentes.

c) Quais operadores do SQL pertencem à álgebra relacional e quais são definidos no cálculo relacional?
   Operadores como **SELECT**, **PROJECT**, **JOIN**, **UNION**, **INTERSECT**, **DIFFERENCE** pertencem à álgebra relacional, enquanto operadores como **FORALL**, **EXISTS**, **IN** são definidos no cálculo relacional.

d) Como é feito o agrupamento em consultas, e qual requisito é obrigatório?
   O agrupamento em consultas é feito usando a cláusula **GROUP BY**, onde os resultados são agrupados com base nos valores de uma ou mais colunas. É obrigatório incluir uma função de agregação, como **SUM**, **COUNT**, **AVG**, etc., em colunas que não estão incluídas na cláusula **GROUP BY**.