 Estácio	<p align="center"> Universidade Estácio de Sá Desenvolvimento Full Stack Nível 2 – Vamos manter as informações! Turma 2022.3 – 3º Semestre </p>
Nome:	Luan Augusto Vieira Bandeira
Repositório:	https://github.com/luanguto/nivel-2-mundo-3

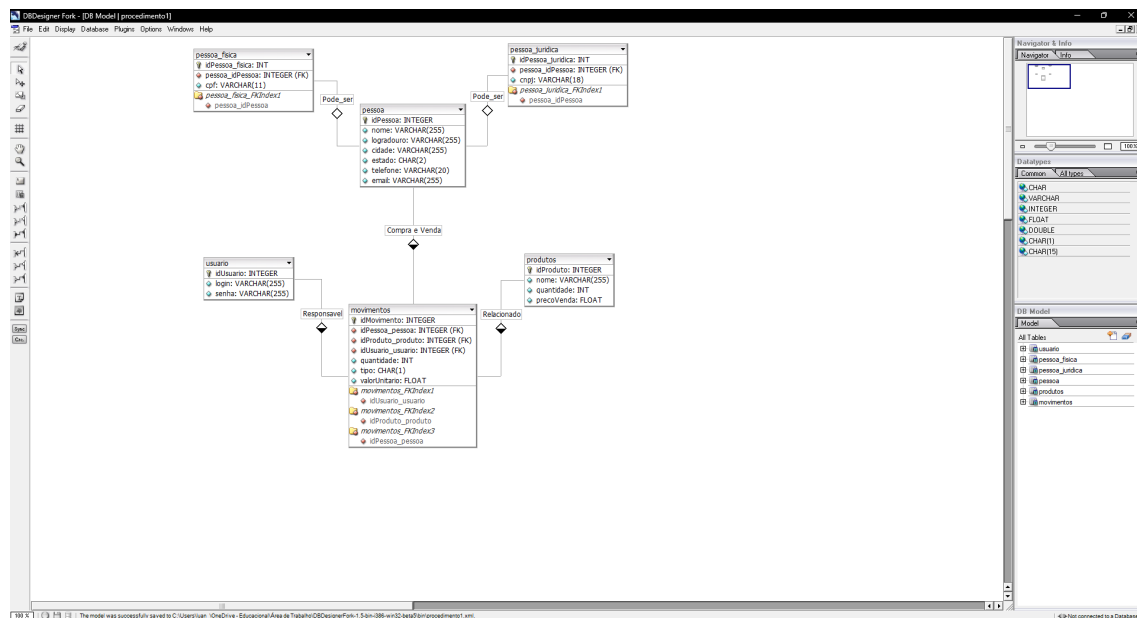
Modelagem e implementação de um banco de dados simples, utilizando como base o SQL Server.

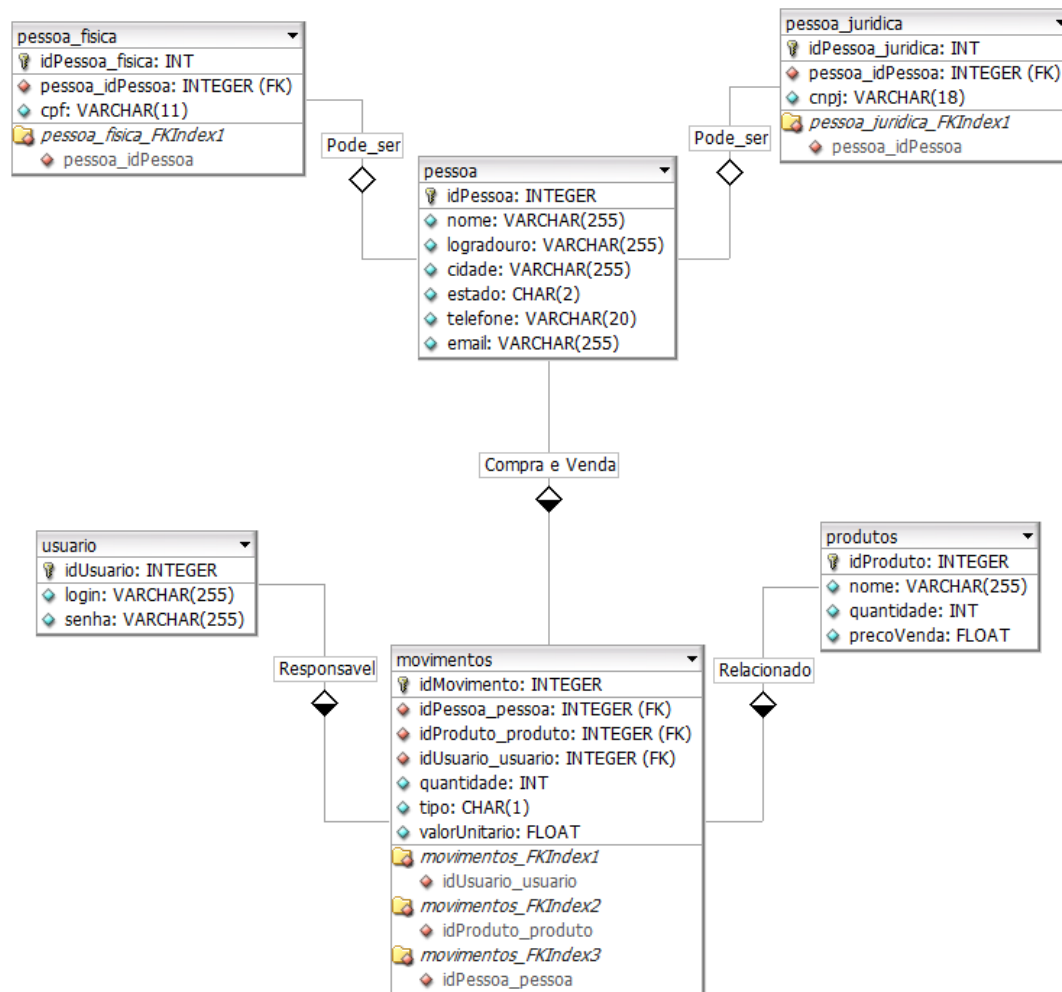
Objetivos da prática

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.

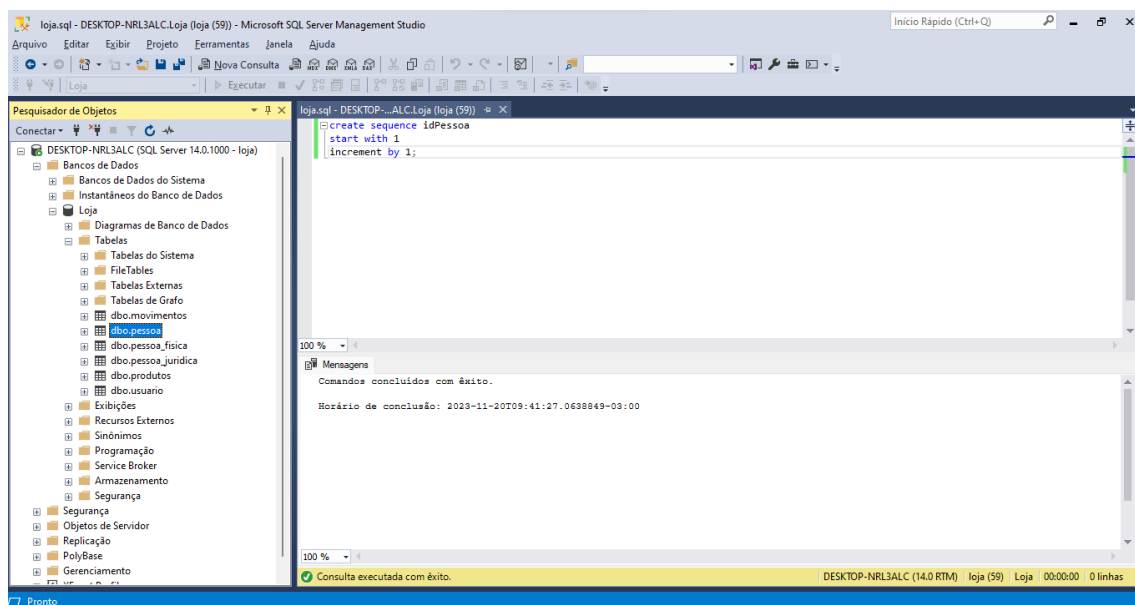
1º Procedimento – Criando o Banco de Dados

2. Definir o modelo de dados para um sistema com as características apresentadas nos tópicos seguintes:

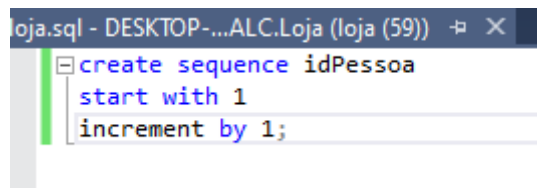




3. Utilizar o SQL Server Management Studio para criar a base de dados modelada no t3pico anterior:

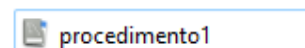


d. Definir uma sequence para geração dos identificadores de pessoa, dado o relacionamento 1x1 com pessoa física ou jurídica.



```
loja.sql - DESKTOP-...ALC.Loja (loja (59))
create sequence idPessoa
start with 1
increment by 1;
```

e. Salvar o script completo para criação do banco de dados em um arquivo com extensão .sql



```
CREATE TABLE produtos (
  idProduto INTEGER NOT NULL ,
  nome VARCHAR(255) NOT NULL ,
  quantidade INT NOT NULL ,
  precoVenda FLOAT NOT NULL ,
  PRIMARY KEY(idProduto));
```

```
CREATE TABLE pessoa (
  idPessoa INTEGER NOT NULL ,
  nome VARCHAR(255) ,
  logradouro VARCHAR(255) ,
  cidade VARCHAR(255) ,
  estado CHAR(2) ,
  telefone VARCHAR(20) ,
  email VARCHAR(255) ,
  PRIMARY KEY(idPessoa));
```

```
CREATE TABLE usuario (
  idUsuario INTEGER NOT NULL ,
  login VARCHAR(255) ,
  senha VARCHAR(255) ,
  PRIMARY KEY(idUsuario));
```

```
CREATE TABLE pessoa_juridica (
  idPessoa_juridica INT NOT NULL ,
  pessoa_idPessoa INTEGER NOT NULL ,
```

```
    cnpj VARCHAR(18) ,  
    PRIMARY KEY(idPessoa_juridica) ,  
    FOREIGN KEY(pessoa_idPessoa)  
        REFERENCES pessoa(idPessoa));
```

```
CREATE INDEX pessoa_juridica_FKIndex1 ON pessoa_juridica (pessoa_idPessoa);
```

```
CREATE INDEX IFK_Pode_ser ON pessoa_juridica (pessoa_idPessoa);
```

```
CREATE TABLE pessoa_fisica (  
    idPessoa_fisica INT NOT NULL ,  
    pessoa_idPessoa INTEGER NOT NULL ,  
    cpf VARCHAR(11) NOT NULL ,  
    PRIMARY KEY(idPessoa_fisica) ,  
    FOREIGN KEY(pessoa_idPessoa)  
        REFERENCES pessoa(idPessoa));
```

```
CREATE INDEX pessoa_fisica_FKIndex1 ON pessoa_fisica (pessoa_idPessoa);
```

```
CREATE INDEX IFK_Pode_ser ON pessoa_fisica (pessoa_idPessoa);
```

```
CREATE TABLE movimentos (  
    idMovimento INTEGER NOT NULL ,  
    idPessoa_pessoa INTEGER NOT NULL ,  
    idProduto_produto INTEGER NOT NULL ,  
    idUsuario_usuario INTEGER NOT NULL ,  
    quantidade INT ,  
    tipo CHAR(1) ,  
    valorUnitario FLOAT ,  
    PRIMARY KEY(idMovimento) ,  
    FOREIGN KEY(idUsuario_usuario)  
        REFERENCES usuario(idUsuario),  
    FOREIGN KEY(idProduto_produto)  
        REFERENCES produtos(idProduto),  
    FOREIGN KEY(idPessoa_pessoa)  
        REFERENCES pessoa(idPessoa));
```

```
CREATE INDEX movimentos_FKIndex1 ON movimentos (idUsuario_usuario);
```

```
CREATE INDEX movimentos_FKIndex2 ON movimentos (idProduto_produto);
```

```
CREATE INDEX movimentos_FKIndex3 ON movimentos (idPessoa_pessoa);
```

```
CREATE INDEX IFK_Responsavel ON movimentos (idUsuario_usuario);
```

```
CREATE INDEX IFK_Relacionado ON movimentos (idProduto_produto);
```

```
CREATE INDEX IFK_Compra e Venda ON movimentos (idPessoa_pessoa);
```

Análise e Conclusão:

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

1x1: Chave estrangeira que também é chave primária ou única.

1xN: Chave estrangeira em uma tabela 'muitos' apontando para uma chave primária em uma tabela 'um'.

NxN: Tabela de junção com chaves estrangeiras apontando para as chaves primárias das tabelas que estão sendo relacionadas.

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

Simulada usando uma tabela base para campos comuns e tabelas separadas para subclasses, ou uma única tabela com uma coluna discriminadora.

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

Interface gráfica para gerenciamento fácil, ferramentas de diagnóstico, suporte a scripts, e funcionalidades para gerenciamento de segurança e dados.

2º Procedimento – Alimentando a base

1-b:

	idUsuario	login	senha
1	1	op1	op1
2	2	op2	op2

1-c:

	idProduto	nome	quantidade	precoVenda
1	1	Banana	100	5
2	2	Laranja	500	2
3	3	Manga	800	4
4	4	Uva	200	5

2-c:

idPessoa_fisica	pessoa_idPessoa	cpf	idPessoa	nome	logradouro	cidade	estado	telefone	email
1	1	1500447897	1	Luan	Rua José Lucio 1	Tanguá	RJ	11111111	Luan@estacio.br

2-d:

idPessoa_juridica	pessoa_idPessoa	cnpj	idPessoa	nome	logradouro	cidade	estado	telefone	email
2	6	156894235400018	2	Lana	Rua José Lucio 1	Tanguá	RJ	11111111	Lana@estacio.br

3:

	idMovimento	idPessoa_pessoa	idProduto_produto	idUsuario_usuario	quantidade	tipo	valorUnitario
1	1	7	1	1	20	S	4
2	4	7	3	1	15	S	2
3	5	7	3	2	10	S	3
4	7	6	3	1	15	E	5
5	8	6	4	1	20	E	4

4. Efetuar as seguintes consultas sobre os dados inseridos:

Dados completos de pessoas físicas.

	idPessoa_fisica	pessoa_idPessoa	cpf
1	1	1	1500447897
2	2	2	1544861285
3	3	3	1567489432
4	4	4	4156489432

Dados completos de pessoas jurídicas.

	idPessoa_juridica	pessoa_idPessoa	cnpj
1	1	5	156156894200019
2	2	6	156894235400018
3	3	7	98749842100019

Movimentações de entrada, com produto, fornecedor, quantidade, preço unitário e valor total.

	idMovimento	idPessoa_pessoa	idProduto_produto	idUsuario_usuario	quantidade	tipo	valorUnitario	Produto	Fornecedor	quantidade	valorUnitario	ValorTotal
1	7	6	3	1	15	E	5	Manga	Fernando	15	5	75
2	8	6	4	1	20	E	4	Uva	Fernando	20	4	80

Movimentações de saída, com produto, comprador, quantidade, preço unitário e valor total.

	idMovimento	idPessoa_pessoa	idProduto_produto	idUsuario_usuario	quantidade	tipo	valorUnitario	Produto	Comprador	quantidade	valorUnitario	ValorTotal
1	1	7	1	1	20	S	4	Banana	Vieira	20	4	80
2	4	7	3	1	15	S	2	Manga	Vieira	15	2	30
3	5	7	3	2	10	S	3	Manga	Vieira	10	3	30

Valor total das entradas agrupadas por produto.

	Produto	ValorTotal
1	Manga	75
2	Uva	80

Valor total das saídas agrupadas por produto.

Resultados		Mensagens	
	Produto	ValorTotal	
1	Banana	80	
2	Manga	60	

Operadores que não efetuaram movimentações de entrada (compra).

Resultados		Mensagens	
	idUsuario	login	senha
1	2	op2	op2

Valor total de entrada, agrupado por operador.

Resultados		Mensagens	
	idUsuario	login	ValorTotal
1	1	op1	155

Valor total de saída, agrupado por operador.

	Resultados	Mensagens	
	idUsuario	login	ValorTotal
1	1	op1	110
2	2	op2	30

Valor médio de venda por produto, utilizando média ponderada.

	Resultados	Mensagens	
	Produto	ValorMedio	
1	Banana	4	
2	Manga	2,4	

Códigos solicitados:

```
create database Loja
```

```
CREATE TABLE produtos (  
    idProduto INTEGER NOT NULL ,  
    nome VARCHAR(255) NOT NULL ,  
    quantidade INT NOT NULL ,  
    precoVenda FLOAT NOT NULL ,  
    PRIMARY KEY(idProduto));
```

```
CREATE TABLE pessoa (  
    idPessoa INTEGER NOT NULL ,  
    nome VARCHAR(255) ,  
    logradouro VARCHAR(255) ,  
    cidade VARCHAR(255) ,  
    estado CHAR(2) ,  
    telefone VARCHAR(20) ,  
    email VARCHAR(255) ,  
    PRIMARY KEY(idPessoa));
```

```
CREATE TABLE usuario (  
    idUsuario INTEGER NOT NULL ,  
    login VARCHAR(255) ,  
    senha VARCHAR(255) ,  
    PRIMARY KEY(idUsuario));
```

```
CREATE TABLE pessoa_juridica (  
    idPessoa_juridica INT NOT NULL ,  
    pessoa_idPessoa INTEGER NOT NULL ,  
    cnpj VARCHAR(18) ,  
    PRIMARY KEY(idPessoa_juridica) ,  
    FOREIGN KEY(pessoa_idPessoa)  
        REFERENCES pessoa(idPessoa));
```

```
CREATE INDEX pessoa_juridica_FKIndex1 ON pessoa_juridica (pessoa_idPessoa);
```

```
CREATE INDEX IFK_Pode_ser ON pessoa_juridica (pessoa_idPessoa);
```

```
CREATE TABLE pessoa_fisica (  
    idPessoa_fisica INT NOT NULL ,  
    pessoa_idPessoa INTEGER NOT NULL ,  
    cpf VARCHAR(11) NOT NULL ,  
    PRIMARY KEY(idPessoa_fisica) ,  
    FOREIGN KEY(pessoa_idPessoa)  
        REFERENCES pessoa(idPessoa));
```

```
CREATE INDEX pessoa_fisica_FKIndex1 ON pessoa_fisica (pessoa_idPessoa);
```

```
CREATE INDEX IFK_Pode_ser ON pessoa_fisica (pessoa_idPessoa);
```

```
CREATE TABLE movimentos (  
    idMovimento INTEGER NOT NULL ,  
    idPessoa_pessoa INTEGER NOT NULL ,  
    idProduto_produto INTEGER NOT NULL ,  
    idUsuario_usuario INTEGER NOT NULL ,  
    quantidade INT ,  
    tipo CHAR(1) ,  
    valorUnitario FLOAT ,  
    PRIMARY KEY(idMovimento) ,  
    FOREIGN KEY(idUsuario_usuario)  
        REFERENCES usuario(idUsuario),  
    FOREIGN KEY(idProduto_produto)  
        REFERENCES produtos(idProduto),  
    FOREIGN KEY(idPessoa_pessoa)  
        REFERENCES pessoa(idPessoa));
```

```
CREATE INDEX movimentos_FKIndex1 ON movimentos (idUsuario_usuario);  
CREATE INDEX movimentos_FKIndex2 ON movimentos (idProduto_produto);  
CREATE INDEX movimentos_FKIndex3 ON movimentos (idPessoa_pessoa);
```

```
CREATE INDEX IFK_Responsavel ON movimentos (idUsuario_usuario);  
CREATE INDEX IFK_Relacionado ON movimentos (idProduto_produto);  
CREATE INDEX IFK_Compra e Venda ON movimentos (idPessoa_pessoa);
```

```
select * from usuario
```

```
insert into usuario (idUsuario, login, senha)  
values (1, 'op1', 'op1'),  
(2, 'op2', 'op2');
```

```
select * from produtos
```

```
INSERT INTO produtos(idProduto, nome, quantidade, precoVenda) VALUES  
(1, 'Banana', 100, 5.00);  
INSERT INTO produtos(idProduto, nome, quantidade, precoVenda) VALUES  
(2, 'Laranja', 500, 2.00);  
INSERT INTO produtos(idProduto, nome, quantidade, precoVenda) VALUES  
(3, 'Manga', 800, 4.00);  
INSERT INTO produtos(idProduto, nome, quantidade, precoVenda) VALUES  
(4, 'Uva', 200, 5.00);
```

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

```
select * from pessoa, pessoa_fisica
```

```
insert into pessoa (idPessoa, nome, logradouro, cidade, estado, telefone, email)  
values  
( '1' , 'Luan' , 'Rua José Lucio 1 ' , 'Tanguá' , 'RJ' , '11111111' ,  
'Luan@estacio.br'),  
( '2' , 'Lana' , 'Rua José Lucio 1 ' , 'Tanguá' , 'RJ' , '11111111' ,  
'Lana@estacio.br'),
```

```
( '3' , 'Bandeira' , 'Rua José Lucio 1 ' , 'Tanguá' , 'RJ' , '11111111' ,
'Bandeira@estacio.br'),
( '4' , 'Gabriel' , 'Rua José Lucio 21 ' , 'Tanguá' , 'RJ' , '11111111' ,
'Gabriel@estacio.br'),
( '5' , 'Felipe' , 'Rua José Lucio 31 ' , 'Tanguá' , 'RJ' , '11111111' ,
'Felipe@estacio.br'),
( '6' , 'Fernando' , 'Rua José Lucio 41 ' , 'Tanguá' , 'RJ' , '11111111' ,
'Fernando@estacio.br'),
( '7' , 'Vieira' , 'Rua José Lucio 51 ' , 'Tanguá' , 'RJ' , '11111111' ,
'Vieira@estacio.br');
```

```
select * from pessoa_fisica, pessoa
insert into pessoa_fisica (idPessoa_fisica, pessoa_idPessoa, cpf)
values ( '1' , '1' , '1500447897'),
( '2' , '2' , '1544861285'),
( '3' , '3' , '1567489432'),
( '4' , '4' , '4156489432');
```

```
select * from pessoa_juridica,pessoa
insert into pessoa_juridica (idPessoa_juridica, pessoa_idPessoa, cnpj)
values ( '1' , '5' , '156156894200019'),
( '2' , '6' , '156894235400018'),
( '3' , '7' , '98749842100019');
```

```
select * from movimentos
insert into movimentos (idMovimento, idPessoa_pessoa, idProduto_produto,
idUsuario_usuario,quantidade,tipo,valorUnitario)
values (1,7,1,1,20,'S',4.00),
(4,7,3,1,15,'S',2.00),
(5,7,3,2,10,'S',3.00),
(7,6,3,1,15,'E',5),
(8,6,4,1,20,'E',4.00);
```

```
-- Dados completos de pessoas físicas:
SELECT * FROM pessoa_fisica;
```

```
--Dados completos de pessoas jurídicas:
SELECT * FROM pessoa_juridica;
```

```
--Movimentações de entrada (com detalhes do produto e fornecedor):
SELECT m.*, p.nome AS Produto, pf.nome AS Fornecedor, m.quantidade,
m.valorUnitario, (m.quantidade * m.valorUnitario) AS ValorTotal
FROM movimentos m
JOIN produtos p ON m.idProduto_produto = p.idProduto
JOIN pessoa pf ON m.idPessoa_pessoa = pf.idPessoa
WHERE m.tipo = 'E';
```

```
--Movimentações de saída (com detalhes do produto e comprador):
SELECT m.*, p.nome AS Produto, pf.nome AS Comprador, m.quantidade,
m.valorUnitario, (m.quantidade * m.valorUnitario) AS ValorTotal
FROM movimentos m
JOIN produtos p ON m.idProduto_produto = p.idProduto
JOIN pessoa pf ON m.idPessoa_pessoa = pf.idPessoa
WHERE m.tipo = 'S';
```

```
--Valor total das entradas agrupadas por produto:
SELECT p.nome AS Produto, SUM(m.quantidade * m.valorUnitario) AS ValorTotal
FROM movimentos m
JOIN produtos p ON m.idProduto_produto = p.idProduto
WHERE m.tipo = 'E'
```

```
GROUP BY p.nome;
```

--Valor total das saídas agrupadas por produto:

```
SELECT p.nome AS Produto, SUM(m.quantidade * m.valorUnitario) AS ValorTotal
FROM movimentos m
JOIN produtos p ON m.idProduto_produto = p.idProduto
WHERE m.tipo = 'S'
GROUP BY p.nome;
```

--Operadores que não efetuaram movimentações de entrada (compra):

```
SELECT u.*
FROM usuario u
WHERE NOT EXISTS (
    SELECT 1
    FROM movimentos m
    WHERE m.idUsuario_usuario = u.idUsuario AND m.tipo = 'E'
);
```

--Valor total de entrada, agrupado por operador:

```
SELECT u.idUsuario, u.login, SUM(m.quantidade * m.valorUnitario) AS ValorTotal
FROM movimentos m
JOIN usuario u ON m.idUsuario_usuario = u.idUsuario
WHERE m.tipo = 'E'
GROUP BY u.idUsuario, u.login;
```

--Valor total de saída, agrupado por operador:

```
SELECT u.idUsuario, u.login, SUM(m.quantidade * m.valorUnitario) AS ValorTotal
FROM movimentos m
JOIN usuario u ON m.idUsuario_usuario = u.idUsuario
WHERE m.tipo = 'S'
GROUP BY u.idUsuario, u.login;
```

--Valor médio de venda por produto, utilizando média ponderada:

```
SELECT p.nome AS Produto, SUM(m.quantidade * m.valorUnitario) / SUM(m.quantidade)
AS ValorMedio
FROM movimentos m
JOIN produtos p ON m.idProduto_produto = p.idProduto
WHERE m.tipo = 'S'
GROUP BY p.nome;
```

Análise e Conclusão:

Quais as diferenças no uso de sequence e identity?

Sequence: Objeto separado, gera números sequenciais, pode ser usado em várias tabelas.

Identity: Propriedade de uma coluna, gera números automaticamente, vinculada a uma tabela específica.

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

Mantêm a integridade referencial, garantindo relações consistentes entre tabelas e evitando dados órfãos.

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

Álgebra Relacional: Operadores como JOIN, SELECT, UNION.

Cálculo Relacional: Baseado em expressões lógicas e variáveis.

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

Feito com GROUP BY.

Todas as colunas na cláusula SELECT que não são funções de agregação devem estar no GROUP BY.