Vamos manter as informações!

Maiara Accacio Machado - 202204268183

Polo Downtown – Barra da Tijuca - RJ
Inserir aqui o nome da Disciplina – Número da Turma – Semestre Letivo

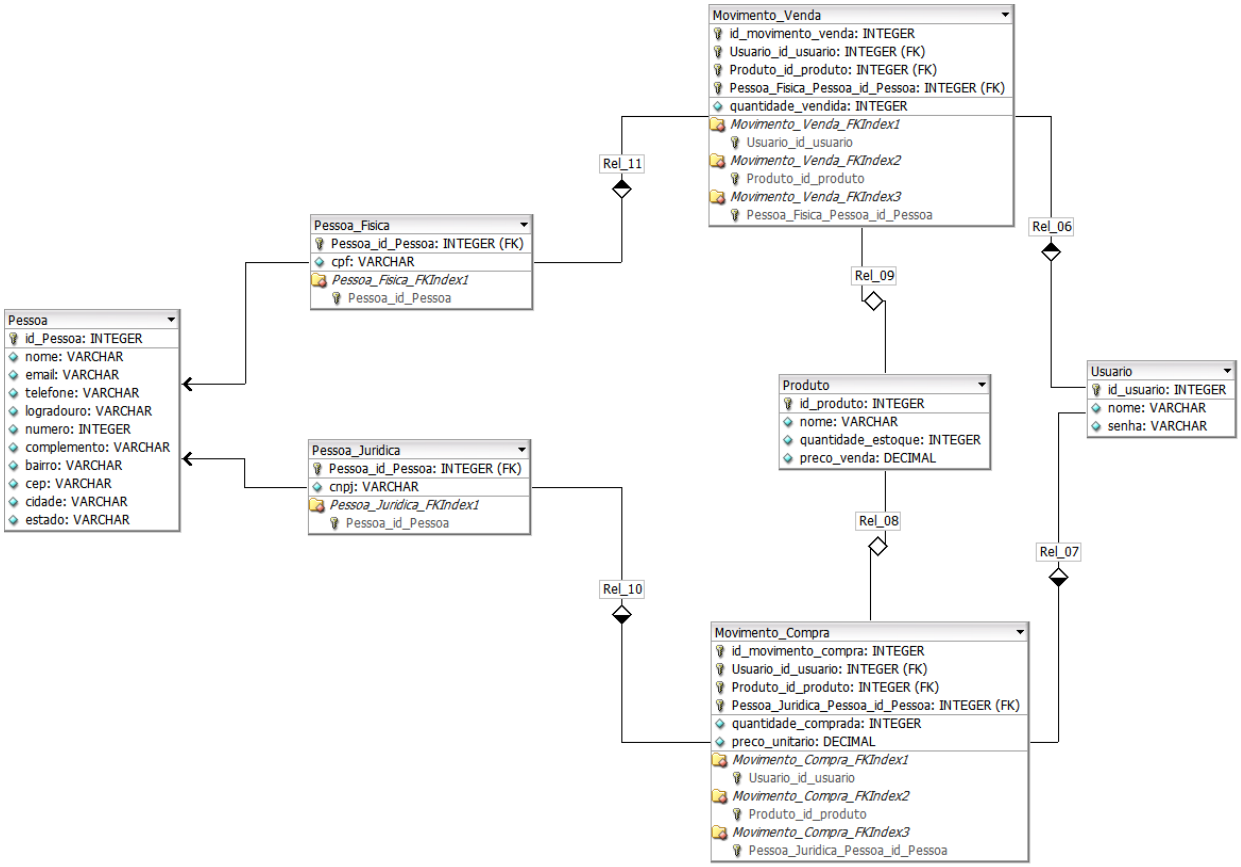
Objetivo da Prática

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)

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

Modelagem:



Código SQL:

```
create database loja;

use loja;

CREATE SEQUENCE PessoaSequence START WITH 1 INCREMENT BY 1;

CREATE TABLE Pessoa(
    id_Pessoa INTEGER PRIMARY KEY NOT NULL DEFAULT (NEXT VALUE FOR
PessoaSequence),
    nome VARCHAR(100) NOT NULL,
    logradouro VARCHAR(100) NOT NULL,
    numero INTEGER,
    complemento VARCHAR(50),
    bairro VARCHAR(30) NOT NULL,
    cidade VARCHAR(30) NOT NULL,
    estado VARCHAR(2) NOT NULL,
    email VARCHAR(50) NOT NULL,
    telefone VARCHAR(15) NOT NULL
);

CREATE TABLE Pessoa_Juridica(
    id_Pessoa INTEGER PRIMARY KEY,
    cnpj VARCHAR(25) NOT NULL,
    FOREIGN KEY(id_Pessoa) REFERENCES Pessoa(id_Pessoa)
);


CREATE TABLE Pessoa_Fisica(
    id_Pessoa INTEGER PRIMARY KEY,
```

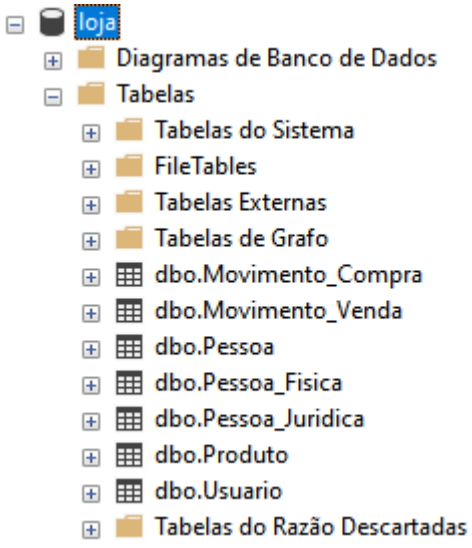
```
    cpf VARCHAR(15) NOT NULL,
    FOREIGN KEY(id_Pessoa) REFERENCES Pessoa(id_Pessoa)
);

CREATE TABLE Produto(
    id_Produto INTEGER PRIMARY KEY NOT NULL IDENTITY(1,1),
    nome VARCHAR(50) NOT NULL,
    quantidade_estoque INTEGER NOT NULL,
    preco_venda MONEY NOT NULL
);

CREATE TABLE Usuario(
    id_Usuario INTEGER PRIMARY KEY NOT NULL IDENTITY(1,1),
    login VARCHAR(50) NOT NULL,
    senha VARCHAR(10) NOT NULL
);

CREATE TABLE Movimento_Compra(
    id_mov_compra INTEGER PRIMARY KEY NOT NULL IDENTITY(1,1),
    id_Usuario INTEGER NOT NULL,
    id_Produto INTEGER NOT NULL,
    id_Pessoa INTEGER NOT NULL,
    quantidade_comprada INTEGER NOT NULL,
    preco_unitario MONEY NOT NULL,
    FOREIGN KEY(id_Usuario) REFERENCES Usuario(id_Usuario),
    FOREIGN KEY(id_Produto) REFERENCES Produto(id_Produto),
    FOREIGN KEY(id_Pessoa) REFERENCES Pessoa_Juridica(id_Pessoa)
);

CREATE TABLE Movimento_Venda(
    id_mov_venda INTEGER PRIMARY KEY NOT NULL IDENTITY(1,1),
    id_Usuario INTEGER NOT NULL,
    id_Produto INTEGER NOT NULL,
    id_Pessoa INTEGER NOT NULL,
    quantidade_vendida INTEGER NOT NULL,
    FOREIGN KEY(id_Usuario) REFERENCES Usuario(id_Usuario),
    FOREIGN KEY(id_Produto) REFERENCES Produto(id_Produto),
    FOREIGN KEY(id_Pessoa) REFERENCES Pessoa_Fisica(id_Pessoa)
);
```



Análise e Conclusão:

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

Resposta:

1X1 – Cada registro de uma tabela está relacionado a um único registro da outra tabela.

1XN – Cada registro de uma tabela pode estar relacionado a vários registros da outra tabela.

NXN – relação de muitos para muitos que significa que qualquer registro da tabela pode estar relacionado a vários registros da outra tabela e vice-versa.

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

Resposta: O tipo indicado é o relacionamento de generalização/Especialização.

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

Resposta: A interface é mais amigável para o usuário do que a interação via prompt, tornando o uso mais fácil e consequentemente mais produtivo. Além disso, oferece um assistente capaz de realizar toda configuração possível de ser feita em SQL. Um exemplo é a possibilidade de criação de bancos, tabelas e o próprio script.

2º Procedimento – Alimentando a Base

1. Utilizar o SQL Server Management Studio para alimentar as tabelas com dados básicos do sistema:

   a. Logar – OK

   b. Utilizar o editor de SQL para incluir dados na tabela de usuários

```
/*Incluindo dados de Usuário  */

INSERT INTO USUARIO (login, senha) VALUES ('op1', 'op1');
INSERT INTO USUARIO (login, senha) VALUES ('op2', 'op2');
INSERT INTO USUARIO (login, senha) VALUES ('op3', 'op3');

/*Consultando dados de Usuário  */

SELECT * FROM USUARIO
```

Resultados | Mensagens

	id_Usuario	login	senha
1	1	op1	op1
2	2	op2	op2
3	3	op3	op3

   c.Inserir alguns produtos na base de dados:

```
/*Incluindo dados de Produto  */

INSERT INTO Produto(nome, quantidade_estoque, preco_venda) VALUES ('Banana',
'100', '5.00');
INSERT INTO Produto(nome, quantidade_estoque, preco_venda) VALUES ('Laranja',
'500', '2.00');
INSERT INTO Produto(nome, quantidade_estoque, preco_venda) VALUES ('Manga',
'800', '4.00');
INSERT INTO Produto(nome, quantidade_estoque, preco_venda) VALUES ('Kiwi',
'1000', '8.00');
INSERT INTO Produto(nome, quantidade_estoque, preco_venda) VALUES ('Pessego',
'600', '7.00');


/*Consultando dados de Produto  */

SELECT * FROM Produto;
```

Resultados | Mensagens

	id_Produto	nome	quantidade_estoque	preco_venda
1	1	Banana	100	5,00
2	2	Laranja	500	2,00
3	3	Manga	800	4,00
4	4	Kiwi	1000	8,00
5	5	Pessego	600	7,00

2. Criar pessoas físicas e jurídicas na base de dados:

    a. Obter o próximo id de pessoa a partir da sequence

```
/*Incluindo dados de Pessoa*/

INSERT INTO Pessoa(id_Pessoa, nome, logradouro, numero, complemento, bairro,
cidade, estado, email, telefone)
VALUES
    (NEXT VALUE FOR PessoaSequence, 'Joao Silva', 'Rua Orucum', '12',
'fundos','Bangu','Rio de Janeiro', 'RJ','joaosilva@riacho.com', '1111-1111'),
    (NEXT VALUE FOR PessoaSequence, 'Maria Souza', 'Rua Banguense', '84', 'Apto
102','Bangu','Rio de Janeiro', 'RJ','maria.souza@riacho.com', '2222-2222'),
    (NEXT VALUE FOR PessoaSequence, 'Janete Brito', 'Rua Flores Brancas', '20',
'','Bangu','Rio de Janeiro', 'RJ','janetebrito@riacho.com', '3333-3333'),
    (NEXT VALUE FOR PessoaSequence, 'JJC', 'Av. Rio Branco', '135',
'','Centro','Rio de Janeiro', 'RJ','jjc@riacho.com', '2222-1111'),
    (NEXT VALUE FOR PessoaSequence, 'ABC', 'Av. Alm. Pedrosa', '356', 'sala
115','Centro','Rio de Janeiro', 'RJ','abc@riacho.com', '3333-2222'),
    (NEXT VALUE FOR PessoaSequence, 'Frutalicia LTDA', 'Rua Paz', '26',
'','Botafogo','Rio de Janeiro', 'RJ','Frutalicia@riacho.com', '4444-1111');
```

Resultados | Mensagens

	id_Pessoa	nome	logradouro	numero	complemento	bairro	cidade	estado	email	telefone
1	1	Joao Silva	Rua Orucum	12	fundos	Bangu	Rio de Janeiro	RJ	joaosilva@riacho.com	1111-1111
2	2	Maria Souza	Rua Banguense	84	Apto 102	Bangu	Rio de Janeiro	RJ	maria.souza@riacho.com	2222-2222
3	3	Janete Brito	Rua Flores Brancas	20		Bangu	Rio de Janeiro	RJ	janetebrito@riacho.com	3333-3333
4	4	JJC	Av. Rio Branco	135		Centro	Rio de Janeiro	RJ	jjc@riacho.com	2222-1111
5	5	ABC	Av. Alm. Pedrosa	356	sala 115	Centro	Rio de Janeiro	RJ	abc@riacho.com	3333-2222
6	6	Frutalicia LTDA	Rua Paz	26		Botafogo	Rio de Janeiro	RJ	Frutalicia@riacho.com	4444-1111

```
/*Incluindo dados de Pessoa Física*/
INSERT INTO Pessoa_Fisica(id_Pessoa, cpf) SELECT p.id_Pessoa, '11111111111'
FROM Pessoa p WHERE p.nome = 'Joao Silva';
INSERT INTO Pessoa_Fisica(id_Pessoa, cpf) SELECT p.id_Pessoa, '22222222222'
FROM Pessoa p WHERE p.nome = 'Maria Souza';
INSERT INTO Pessoa_Fisica(id_Pessoa, cpf) SELECT p.id_Pessoa, '33333333333'
FROM Pessoa p WHERE p.nome = 'Janete Brito';


/*Incluindo dados de Pessoa Jurídica*/
INSERT INTO Pessoa_Juridica(id_Pessoa, cnpj) SELECT p.id_Pessoa,
'11.111.111/0001-00'
FROM Pessoa p WHERE p.nome = 'JJC';
INSERT INTO Pessoa_Juridica(id_Pessoa, cnpj) SELECT p.id_Pessoa,
'22.111.122/0001-00'
FROM Pessoa p WHERE p.nome = 'ABC';
INSERT INTO Pessoa_Juridica(id_Pessoa, cnpj) SELECT p.id_Pessoa,
'11.352.168/0001-00'
       FROM Pessoa p WHERE p.nome = 'Frutalicia LTDA';

/*Consultando dados de Pessoa */
SELECT
    Pessoa.id_Pessoa,
nome,
logradouro,
numero,
```

```
complemento,
bairro,
cidade,
estado,
email,
telefone,
Pessoa_Fisica.cpf AS documento
FROM
    Pessoa JOIN Pessoa_Fisica ON Pessoa.id_Pessoa = Pessoa_Fisica.id_Pessoa
UNION ALL
SELECT
 Pessoa.id_Pessoa,
nome,
logradouro,
numero,
complemento,
bairro,
cidade,
estado,
email,
telefone,
Pessoa_Juridica.cnpj AS tipo
FROM
    Pessoa JOIN Pessoa_Juridica ON Pessoa.id_Pessoa = Pessoa_Juridica.id_Pessoa
```

Resultados | Mensagens

	id_Pessoa	nome	logradouro	numero	complemento	bairro	cidade	estado	email	telefone	documento
1	1	Joao Silva	Rua Orucum	12	fundos	Bangu	Rio de Janeiro	RJ	joaosilva@riacho.com	1111-1111	11111111111
2	2	Maria Souza	Rua Banguense	84	Apto 102	Bangu	Rio de Janeiro	RJ	maria.souza@riacho.com	2222-2222	22222222222
3	3	Janete Brito	Rua Flores Brancas	20		Bangu	Rio de Janeiro	RJ	janetebrito@riacho.com	3333-3333	33333333333
4	4	JJC	Av. Rio Branco	135		Centro	Rio de Janeiro	RJ	jjc@riacho.com	2222-1111	11.111.111/0001-00
5	5	ABC	Av. Alm. Pedrosa	356	sala 115	Centro	Rio de Janeiro	RJ	abc@riacho.com	3333-2222	22.111.122/0001-00
6	6	Frutalicia LTDA	Rua Paz	26		Botafogo	Rio de Janeiro	RJ	Frutalicia@riacho.com	4444-1111	11.352.168/0001-00

3. Criar algumas movimentações na base de dados

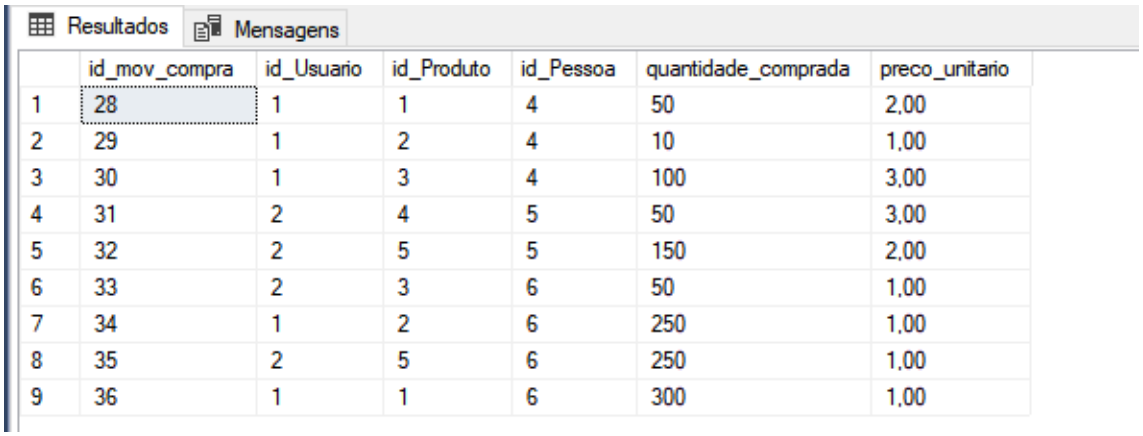
- Movimento de Compra:

```
/*Incluindo Movimento de Compra */

INSERT INTO Movimento_Compra(id_Usuario, id_Produto, id_Pessoa,
quantidade_comprada, preco_unitario) VALUES (1, 1, 4, 50, 2.00);
INSERT INTO Movimento_Compra(id_Usuario, id_Produto, id_Pessoa,
quantidade_comprada, preco_unitario) VALUES (1, 2, 4, 10, 1.00);
INSERT INTO Movimento_Compra(id_Usuario, id_Produto, id_Pessoa,
quantidade_comprada, preco_unitario) VALUES (1, 3, 4, 100, 3.00);
INSERT INTO Movimento_Compra(id_Usuario, id_Produto, id_Pessoa,
quantidade_comprada, preco_unitario) VALUES (2, 4, 5, 50, 3.00);
INSERT INTO Movimento_Compra(id_Usuario, id_Produto, id_Pessoa,
quantidade_comprada, preco_unitario) VALUES (2, 5, 5, 150, 2.00);
INSERT INTO Movimento_Compra(id_Usuario, id_Produto, id_Pessoa,
quantidade_comprada, preco_unitario) VALUES (2, 3, 6, 50, 1.00);
INSERT INTO Movimento_Compra(id_Usuario, id_Produto, id_Pessoa,
quantidade_comprada, preco_unitario) VALUES (1, 2, 6, 250, 1.00);
INSERT INTO Movimento_Compra(id_Usuario, id_Produto, id_Pessoa,
quantidade_comprada, preco_unitario) VALUES (2, 5, 6, 250, 1.00);
INSERT INTO Movimento_Compra(id_Usuario, id_Produto, id_Pessoa,
quantidade_comprada, preco_unitario) VALUES (1, 1, 6, 300, 1.00);
```

```
/*Consultando Movimento de Compra  */

select * from Movimento_Compra
```



Resultados | Mensagens

	id_mov_compra	id_Usuario	id_Produto	id_Pessoa	quantidade_comprada	preco_unitario
1	28	1	1	4	50	2,00
2	29	1	2	4	10	1,00
3	30	1	3	4	100	3,00
4	31	2	4	5	50	3,00
5	32	2	5	5	150	2,00
6	33	2	3	6	50	1,00
7	34	1	2	6	250	1,00
8	35	2	5	6	250	1,00
9	36	1	1	6	300	1,00

- Movimento de venda:

```
/*Incluindo Movimento de Venda */

INSERT INTO Movimento_Venda(id_Usuario, id_Produto, id_Pessoa,
quantidade_vendida) VALUES (1, 1, 1, 50);
INSERT INTO Movimento_Venda(id_Usuario, id_Produto, id_Pessoa,
quantidade_vendida) VALUES (1, 2, 1, 10);
INSERT INTO Movimento_Venda(id_Usuario, id_Produto, id_Pessoa,
quantidade_vendida) VALUES (1, 3, 3, 100);
INSERT INTO Movimento_Venda(id_Usuario, id_Produto, id_Pessoa,
quantidade_vendida) VALUES (2, 4, 2, 50);
INSERT INTO Movimento_Venda(id_Usuario, id_Produto, id_Pessoa,
quantidade_vendida) VALUES (2, 5, 2, 150);
INSERT INTO Movimento_Venda(id_Usuario, id_Produto, id_Pessoa,
quantidade_vendida) VALUES (2, 1, 3, 50);
INSERT INTO Movimento_Venda(id_Usuario, id_Produto, id_Pessoa,
quantidade_vendida) VALUES (1, 4, 1, 10);
INSERT INTO Movimento_Venda(id_Usuario, id_Produto, id_Pessoa,
quantidade_vendida) VALUES (1, 5, 3, 100);
INSERT INTO Movimento_Venda(id_Usuario, id_Produto, id_Pessoa,
quantidade_vendida) VALUES (2, 3, 2, 50);
INSERT INTO Movimento_Venda(id_Usuario, id_Produto, id_Pessoa,
quantidade_vendida) VALUES (2, 2, 2, 150);

/*Consultando Movimento de Venda */

select * from Movimento_Venda
```

Resultados | Mensagens

	id_mov_venda	id_Usuario	id_Produto	id_Pessoa	quantidade_vendida
1	1	1	1	1	50
2	2	1	2	1	10
3	3	1	3	3	100
4	4	2	4	2	50
5	5	2	5	2	150
6	6	2	1	3	50
7	7	1	4	1	10
8	8	1	5	3	100
9	9	2	3	2	50
10	10	2	2	2	150

- Visão geral de movimentos:

```
/*Consultando Todos os movimentos - Venda e Compra*/

SELECT
    id_mov_venda AS id_movimento,
    id_Usuario,
id_Pessoa,
    Movimento_Venda.id_Produto,
quantidade_vendida AS quantidade,
'S' AS tipo,
produto.preco_venda AS ValorUnitario

FROM
    Movimento_Venda JOIN Produto ON Movimento_Venda.id_Produto =
Produto.id_Produto
UNION ALL
SELECT
    id_mov_compra AS id_movimento,
    id_Usuario,
id_Pessoa,
    Movimento_Compra.id_Produto,
quantidade_comprada AS quantidade,
'E' AS tipo,
preco_unitario AS valorUnitario
FROM

    Movimento_Compra;
```

Resultados | Mensagens

	id_movimento	id_Usuario	id_Pessoa	id_Produto	quantidade	tipo	ValorUnitario
1	1	1	1	1	50	S	5,00
2	2	1	1	2	10	S	2,00
3	3	1	3	3	100	S	4,00
4	4	2	2	4	50	S	8,00
5	5	2	2	5	150	S	7,00
6	6	2	3	1	50	S	5,00
7	7	1	1	4	10	S	8,00
8	8	1	3	5	100	S	7,00
9	9	2	2	3	50	S	4,00
10	10	2	2	2	150	S	2,00
11	28	1	4	1	50	E	2,00
12	29	1	4	2	10	E	1,00
13	30	1	4	3	100	E	3,00
14	31	2	5	4	50	E	3,00
15	32	2	5	5	150	E	2,00
16	33	2	6	3	50	E	1,00
17	34	1	6	2	250	E	1,00
18	35	2	6	5	250	E	1,00
19	36	1	6	1	300	E	1,00

4. Efetuar as seguintes consultas sobre os dados inseridos:
   a. Dados completos de pessoas físicas.

```
SELECT * FROM Pessoa_Fisica;
```

Resultados | Mensagens

	id_Pessoa	cpf
1	1	11111111111
2	2	22222222222
3	3	33333333333

   b.Dados completos de pessoas jurídicas.

```
SELECT * FROM Pessoa_Juridica;
```

Resultados | Mensagens

	id_Pessoa	cnpj
1	4	11.111.111/0001-00
2	5	22.111.122/0001-00
3	6	11.352.168/0001-00

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

```
SELECT id_Produto, id_Pessoa AS FORNECEDOR,
quantidade_comprada AS QUANTIDADE,
preco_unitario AS PRECO_UNITARIO,
preco_unitario * quantidade_comprada AS TOTAL
FROM Movimento_Compra
```

Resultados | Mensagens

	id_Produto	FORNECEDOR	QUANTIDADE	PRECO_UNITARIO	TOTAL
1	1	4	50	2,00	100,00
2	2	4	10	1,00	10,00
3	3	4	100	3,00	300,00
4	4	5	50	3,00	150,00
5	5	5	150	2,00	300,00
6	3	6	50	1,00	50,00
7	2	6	250	1,00	250,00
8	5	6	250	1,00	250,00
9	1	6	300	1,00	300,00

- Alterando ids por nome:

```
SELECT Produto.nome AS PRODUTO,
Pessoa.nome AS FORNECEDOR,
quantidade_comprada AS QUANTIDADE,
preco_unitario AS PRECO_UNITARIO,
preco_unitario * quantidade_comprada  AS TOTAL
FROM Movimento_Compra
JOIN Produto ON Movimento_Compra.id_Produto = Produto.id_Produto
JOIN Pessoa ON Movimento_Compra.id_Pessoa = Pessoa.id_Pessoa
```

Resultados | Mensagens

	PRODUTO	FORNECEDOR	QUANTIDADE	PRECO_UNITARIO	TOTAL
1	Banana	JJC	50	2,00	100,00
2	Laranja	JJC	10	1,00	10,00
3	Manga	JJC	100	3,00	300,00
4	Kiwi	ABC	50	3,00	150,00
5	Pessego	ABC	150	2,00	300,00
6	Manga	Frutalicia LTDA	50	1,00	50,00
7	Laranja	Frutalicia LTDA	250	1,00	250,00
8	Pessego	Frutalicia LTDA	250	1,00	250,00
9	Banana	Frutalicia LTDA	300	1,00	300,00

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

```
SELECT Movimento_Venda.id_Produto, id_Pessoa AS COMPRADOR,
quantidade_vendida AS QUANTIDADE,
Produto.preco_venda AS PRECO_UNITARIO, Produto.preco_venda * quantidade_vendida
AS TOTAL
FROM Movimento_Venda
JOIN PRODUTO ON Movimento_Venda.id_Produto = Produto.id_Produto
```

Resultados | Mensagens

	id_Produto	COMPRADOR	QUANTIDADE	PRECO_UNITARIO	TOTAL
1	1	1	50	5,00	250,00
2	2	1	10	2,00	20,00
3	3	3	100	4,00	400,00
4	4	2	50	8,00	400,00
5	5	2	150	7,00	1050,00
6	1	3	50	5,00	250,00
7	4	1	10	8,00	80,00
8	5	3	100	7,00	700,00
9	3	2	50	4,00	200,00
10	2	2	150	2,00	300,00

- Alterando ids por nomes:

```
SELECT produto.nome, pessoa.nome AS COMPRADOR,
quantidade_vendida AS QUANTIDADE,
Produto.preco_venda AS PRECO_UNITARIO,
Produto.preco_venda * quantidade_vendida AS TOTAL
FROM Movimento_Venda
JOIN PRODUTO ON Movimento_Venda.id_Produto = Produto.id_Produto
JOIN Pessoa ON Movimento_venda.id_Pessoa = Pessoa.id_pessoa
```

Resultados | Mensagens

	nome	COMPRADOR	QUANTIDADE	PRECO_UNITARIO	TOTAL
1	Banana	Joao Silva	50	5,00	250,00
2	Laranja	Joao Silva	10	2,00	20,00
3	Manga	Janete Brito	100	4,00	400,00
4	Kiwi	Maria Souza	50	8,00	400,00
5	Pessego	Maria Souza	150	7,00	1050,00
6	Banana	Janete Brito	50	5,00	250,00
7	Kiwi	Joao Silva	10	8,00	80,00
8	Pessego	Janete Brito	100	7,00	700,00
9	Manga	Maria Souza	50	4,00	200,00
10	Laranja	Maria Souza	150	2,00	300,00

e.Valor total das entradas agrupadas por produto.

```
SELECT Produto.nome,
SUM(quantidade_comprada) AS QUANTIDADE,
Produto.preco_venda AS PRECO_UNITARIO,
SUM(Produto.preco_venda) * SUM(quantidade_comprada) AS TOTAL
FROM Movimento_Compra
JOIN Produto ON Movimento_Compra.id_Produto = Produto.id_Produto

GROUP BY Produto.nome,  Produto.preco_venda;
```

Resultados | Mensagens

	nome	QUANTIDADE	PRECO_UNITARIO	TOTAL
1	Laranja	260	2,00	1040,00
2	Manga	150	4,00	1200,00
3	Banana	350	5,00	3500,00
4	Pessego	400	7,00	5600,00
5	Kiwi	50	8,00	400,00

f.Valor total das saídas agrupadas por produto.

```
SELECT  Produto.nome,
SUM(quantidade_vendida) AS QUANTIDADE,
Produto.preco_venda AS PRECO_UNITARIO,
SUM(Produto.preco_venda) * SUM(quantidade_vendida) AS TOTAL
FROM Movimento_Venda
JOIN Produto ON Movimento_Venda.id_Produto = Produto.id_Produto
GROUP BY  Produto.nome, Produto.preco_venda;
```

Resultados | Mensagens

	nome	QUANTIDADE	PRECO_UNITARIO	TOTAL
1	Laranja	160	2,00	640,00
2	Manga	150	4,00	1200,00
3	Banana	100	5,00	1000,00
4	Pessego	250	7,00	3500,00
5	Kiwi	60	8,00	960,00

g.Operadores que não efetuaram movimentações de entrada (compra).

```
SELECT login FROM Usuario
WHERE id_Usuario NOT IN (SELECT id_Usuario FROM Movimento_Compra)
```

Resultados | Mensagens

	login
1	op3

h.Valor total de entrada, agrupado por operador.

```
SELECT Usuario.login AS OPERADOR, sum(preco_unitario) * sum(quantidade_comprada)
AS TOTAL
FROM Movimento_Compra
JOIN Usuario ON Movimento_Compra.id_Usuario = Usuario.id_Usuario
GROUP BY Usuario.login;
```

Resultados | Mensagens

	OPERADOR	TOTAL
1	op1	5680,00
2	op2	3500,00

i.Valor total de saída, agrupado por operador.

```
SELECT Usuario.login AS OPERADOR, sum(Produto.preco_venda) *
sum(quantidade_vendida) AS TOTAL
FROM Movimento_Venda
JOIN Usuario ON Movimento_Venda.id_Usuario = Usuario.id_Usuario
JOIN Produto ON Movimento_Venda.id_Produto = Produto.id_Produto
GROUP BY Usuario.login;
```

Resultados | Mensagens

	OPERADOR	TOTAL
1	op1	7020,00
2	op2	11700,00

j. Valor médio de venda por produto, utilizando média ponderada.

```
SELECT Produto.nome AS PRODUTO,
AVG(Produto.preco_venda * quantidade_vendida) AS Valor_Medio
FROM Movimento_Venda
JOIN Produto ON Movimento_Venda.id_Produto = Produto.id_Produto
GROUP BY Produto.nome;
```

Resultados | Mensagens

	PRODUTO	Valor_Medio
1	Banana	250,00
2	Kiwi	240,00
3	Laranja	160,00
4	Manga	300,00
5	Pessego	875,00

Análise e Conclusão:

a) Quais as diferenças no uso de sequence e identity?

   Resposta: Ambas são utilizadas para gerar valores sequenciais de forma automática, mas Identity é específica do SQL Server. Identity é definida por coluna e limitada ao escopo da tabela, enquanto Sequence pode ser definida por tabela ou esquema. Sequence são indicadas para gerar valores não ligados diretamente a uma coluna específica, já Identity é indicada para valores de chave primária.

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

   Resposta: estabelecem as relações entre tabelas garantindo a integridade referencial, ou seja, evitam que ocorram operações que possam levar a resultados incorretos ou ambíguos.

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

   Resposta: pertecem à álgebra relacional seleção, projeção, união, interseção, diferença, produto cartesiano, junção e divisão.

   São definidos no cálculo relacional seleção, projeção, junção, renomeação, união, interseção, exceção e divisão.

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

   Resposta: O agrupamento é realizado pela cláusula GROUP BY. O requisito obrigatório é a utilização de uma função de agregação que

resume os valores dentro de cada grupo como o SUM e AVG que foram usados nessa tarefa.

Conclusão

Durante esta atividade prática, pude mergulhar em uma série de conceitos essenciais relacionados a bancos de dados relacionais, SQL e o sistema de gerenciamento de banco de dados SQL Server. Durante o trabalhho, tive a oportunidade de aplicar efetivamente conceitos como relacionamentos entre tabelas, operações de agrupamento e junção de tabelas.

Uma das realizações mais satisfatórias para mim foi a capacidade de elaborar consultas que transcendem a simples obtenção de dados, permitindo-me incluir informações mais relevantes, como substituir os identificadores numéricos por nomes significativos de produtos. Essa prática proporcionou uma visão mais rica e contextualizada dos dados, tornando as consultas e os resultados mais compreensíveis.

Ao longo do processo de análise e conclusões, fui incentivada a aprofundar meu entendimento sobre cada conceito, pesquisando mais profundamente e compreendendo os princípios subjacentes. Isso resultou em uma compreensão mais profunda da teoria por trás de cada conceito enriquecendo meu aprendizado. Um exemplo foi o entendimento da diferença entre a utilização da funcionalidade Identity, específica do SQL Server, e da Sequence, e como escolher a abordagem mais apropriada com base nas necessidades e características do banco de dados. Essa compreensão refinada me proporcionou maior flexibilidade para atender às demandas específicas do projeto, utilizando a sequence onde foi pedido e aplicando Identity onde acreditei ser interessante.

No geral, essa missão prática me forneceu uma plataforma valiosa para aplicar, explorar e aprofundar meus conhecimentos em bancos de dados relacionais e SQL Server. Ela me capacitou a manipular dados explorando diferentes abordagens para resolver problemas e, ao mesmo tempo, iniciar a construção de uma base sólida para futuras explorações e desafios, mantendo total compreensão de que o aprendizado adquirido durante o trabalho é apenas o primeiro passo de uma longa caminhada.