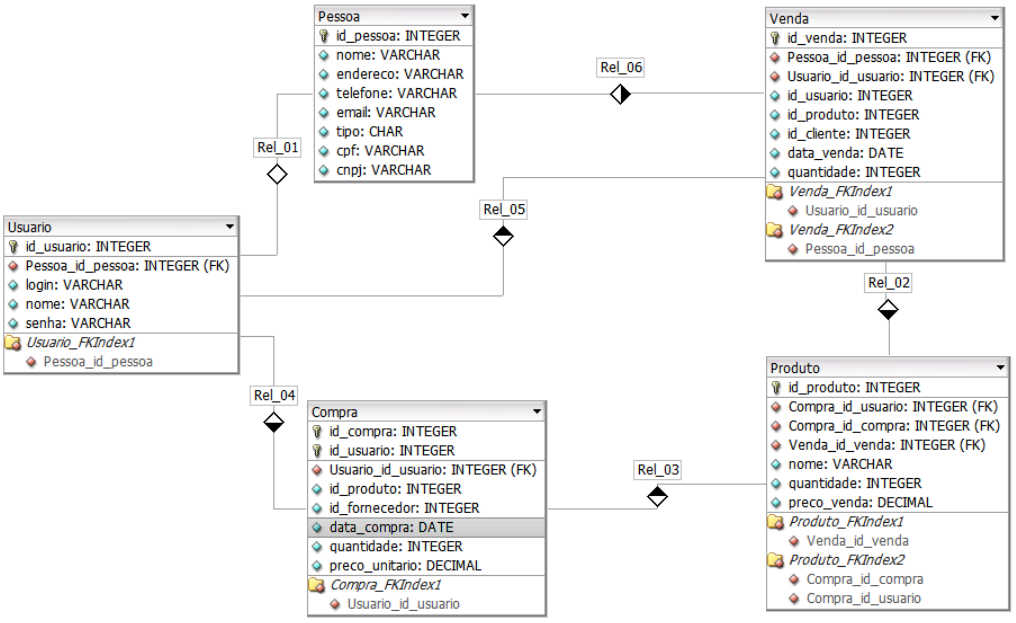
Mundo 3
Missão Prática Nível 2

Objetivos da prática

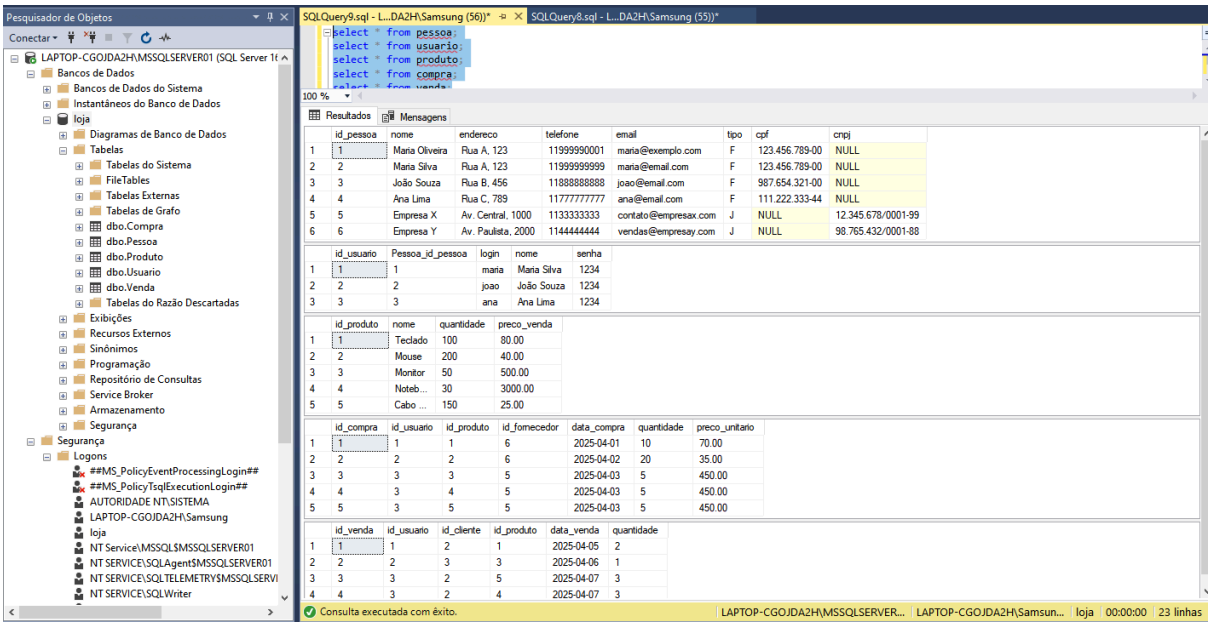
- Identificar os requisitos de um sistema e transformá-los no modelo adequado.
- Utilizar ferramentas de modelagem para bases de dados relacionais.
- Explorar a sintaxe SQL na criação das estruturas do banco (DDL).
- Explorar a sintaxe SQL na consulta e manipulação de dados (DML)
- No final do exercício, 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



Banco de dados



2º Procedimento | Alimentando a Base

Consultas

Pessoa

```sql
--a) Dados completos de pessoas físicas.
SELECT *
FROM Pessoa
WHERE tipo = 'F';

--b)Dados completos de pessoas jurídicas.
SELECT *
FROM Pessoa
WHERE tipo = 'J';
```

100 %

Resultados | Mensagens

1	1	Maria Oliveira	Rua A, 123	11999990001	maria@exemplo.com	F	123.456.789-00	NULL
2	2	Maria Silva	Rua A, 123	11999999999	maria@email.com	F	123.456.789-00	NULL
3	3	João Souza	Rua B, 456	11888888888	joao@email.com	F	987.654.321-00	NULL
4	4	Ana Lima	Rua C, 789	11777777777	ana@email.com	F	111.222.333-44	NULL

	id_pessoa	nome	endereco	telefone	email	tipo	cpf	cnpj
1	5	Empresa X	Av. Central, 1000	1133333333	contato@empresax.com	J	NULL	12.345.678/0001-99
2	6	Empresa Y	Av. Paulista, 2000	1144444444	vendas@empresay.com	J	NULL	98.765.432/0001-88

Produtos

```sql
--c)Movimentações de entrada, com produto, fornecedor, quantidade, preço unitário e valor total.
SELECT
    p.nome AS Fornecedor,
    pr.nome AS Produto,
    c.quantidade,
    c.preco_unitario,
    c.quantidade * c.preco_unitario AS Valor_Total
FROM Compra c
JOIN Pessoa p ON c.id_fornecedor = p.id_pessoa
JOIN Produto pr ON c.id_produto = pr.id_produto;

--d)Movimentações de saída, com produto, comprador, quantidade, preço unitário e valor total.
SELECT
    p.nome AS Cliente,
    pr.nome AS Produto,
    v.quantidade,
    c.preco_unitario,
    v.quantidade * c.preco_unitario AS Valor_Total
FROM Venda v
JOIN Pessoa p ON v.id_cliente = p.id_pessoa
JOIN Produto pr ON v.id_produto = pr.id_produto
JOIN Compra c ON c.id_produto = pr.id_produto;
```

%

Resultados | Mensagens

Fornecedor	Produto	quantidade	preco_unitario	Valor_Total
Empresa Y	Teclado	10	70.00	700.00
Empresa Y	Mouse	20	35.00	700.00
Empresa X	Monitor	5	450.00	2250.00
Empresa X	Notebook	5	450.00	2250.00
Empresa X	Cabo HDMI	5	450.00	2250.00

Cliente	Produto	quantidade	preco_unitario	Valor_Total
Maria Silva	Teclado	2	70.00	140.00
João Souza	Monitor	1	450.00	450.00
Maria Silva	Notebook	3	450.00	1350.00
Maria Silva	Cabo HDMI	3	450.00	1350.00

```sql
--e)Valor total das entradas agrupadas por produto.
SELECT
    pr.nome AS Produto,
    SUM(c.quantidade * c.preco_unitario) AS Total_Entrada
FROM Compra c
JOIN Produto pr ON c.id_produto = pr.id_produto
GROUP BY pr.nome;

--f)Valor total das saídas agrupadas por produto.
SELECT
    pr.nome AS Produto,
    SUM(v.quantidade * c.preco_unitario) AS Total_Saida
FROM Venda v
JOIN Produto pr ON v.id_produto = pr.id_produto
JOIN Compra c ON c.id_produto = pr.id_produto
GROUP BY pr.nome;
```

100 %

Resultados | Mensagens

	Produto	Total_Entrada
1	Cabo HDMI	2250.00
2	Monitor	2250.00
3	Mouse	700.00
4	Notebook	2250.00
5	Teclado	700.00

	Produto	Total_Saida
1	Cabo HDMI	1350.00
2	Monitor	450.00
3	Notebook	1350.00
4	Teclado	140.00

```sql
--g)Operadores que não efetuaram movimentações de entrada (compra).
SELECT u.nome AS Operador
FROM Usuario u
WHERE NOT EXISTS (
    SELECT 1
    FROM Compra c
    WHERE c.id_usuario = u.id_usuario
);

--h)Valor total de entrada, agrupado por operador.
SELECT
    u.nome AS Operador,
    SUM(c.quantidade * c.preco_unitario) AS Total_Entrada
FROM Compra c
JOIN Usuario u ON c.id_usuario = u.id_usuario
GROUP BY u.nome;
```

) %

Resultados | Mensagens

Operador
Ana Carolina

Operador	Total_Entrada
Ana Lima	6750.00
João Souza	700.00
Maria Silva	700.00

```sql
--i)Valor total de saída, agrupado por operador.
SELECT
    u.nome AS Operador,
    SUM(v.quantidade * c.preco_unitario) AS Total_Saida
FROM Venda v
JOIN Usuario u ON v.id_usuario = u.id_usuario
JOIN Compra c ON c.id_produto = v.id_produto
GROUP BY u.nome;

--j)Valor médio de venda por produto, utilizando média ponderada.
SELECT
    pr.nome AS Produto,
    SUM(v.quantidade * c.preco_unitario) * 1.0 / NULLIF(SUM(v.quantidade), 0) AS Media_Ponderada
FROM Venda v
JOIN Produto pr ON v.id_produto = pr.id_produto
JOIN Compra c ON c.id_produto = pr.id_produto
GROUP BY pr.nome;
```

00 %

Resultados | Mensagens

	Operador	Total_Saida
1	Ana Lima	2700.00
2	João Souza	450.00
3	Maria Silva	140.00

	Produto	Media_Ponderada
1	Cabo HDMI	450.000000
2	Monitor	450.000000
3	Notebook	450.000000
4	Teclado	70.000000

Script para criação do banco de dados

```
-- Criar a tabela Pessoa
CREATE TABLE Pessoa (
  id_pessoa INT NOT NULL PRIMARY KEY,
  nome VARCHAR(100),
  endereco VARCHAR(200),
  telefone VARCHAR(20),
  email VARCHAR(100),
  tipo CHAR(1) CHECK (tipo IN ('F', 'J')),
  cpf VARCHAR(14),
  cnpj VARCHAR(18)
);

CREATE SEQUENCE seq_id_pessoa
    START WITH 1
    INCREMENT BY 1;

-- Criar a tabela Usuario
CREATE TABLE Usuario (
  id_usuario INT NOT NULL PRIMARY KEY,
  Pessoa_id_pessoa INT NOT NULL,
  login VARCHAR(50) NOT NULL,
  nome VARCHAR(100) NOT NULL,
  senha VARCHAR(100) NOT NULL,
  FOREIGN KEY (Pessoa_id_pessoa) REFERENCES Pessoa(id_pessoa)
);

CREATE TABLE Compra (
  id_compra INT NOT NULL PRIMARY KEY,
  id_usuario INT NOT NULL,
  id_produto INT,
  id_fornecedor INT,
  data_compra DATE,
  quantidade INT,
  preco_unitario DECIMAL(10, 2),
  FOREIGN KEY (id_usuario) REFERENCES Usuario(id_usuario)
);

CREATE TABLE Venda (
  id_venda INT NOT NULL PRIMARY KEY,
  id_usuario INT NOT NULL,         -- vendedor
  id_cliente INT NOT NULL,         -- cliente (referência a Pessoa)
  id_produto INT NOT NULL,         -- produto vendido
  data_venda DATE,
  quantidade INT,
  FOREIGN KEY (id_usuario) REFERENCES Usuario(id_usuario),
  FOREIGN KEY (id_cliente) REFERENCES Pessoa(id_pessoa),
  FOREIGN KEY (id_produto) REFERENCES Produto(id_produto)
);

CREATE TABLE Produto (
  id_produto INT NOT NULL PRIMARY KEY,
  nome VARCHAR(100) NOT NULL,
  quantidade INT,
  preco_venda DECIMAL(10, 2)
);
```

Script para inserção de dados nas tabelas do banco de dados

```
-- Inserir pessoas (3 físicas e 2 jurídicas)
INSERT INTO Pessoa (id_pessoa, nome, endereco, telefone, email, tipo, cpf, cnpj)
VALUES
(NEXT VALUE FOR seq_id_pessoa, 'Maria Silva', 'Rua A, 123', '11999999999', 'maria@email.com', 'F', '123.456.789-00', NULL),
(NEXT VALUE FOR seq_id_pessoa, 'João Souza', 'Rua B, 456', '11888888888', 'joao@email.com', 'F', '987.654.321-00', NULL),
(NEXT VALUE FOR seq_id_pessoa, 'Ana Lima', 'Rua C, 789', '11777777777', 'ana@email.com', 'F', '111.222.333-44', NULL),
(NEXT VALUE FOR seq_id_pessoa, 'Ana Carolina', 'Rua D, 123', '11666666666', 'ana@email.com', 'F', '333.555.666-55', NULL),
(NEXT VALUE FOR seq_id_pessoa, 'Empresa X', 'Av. Central, 1000', '1133333333', 'contato@empresax.com', 'J', NULL, '12.345.678/0001-99'),
(NEXT VALUE FOR seq_id_pessoa, 'Empresa Y', 'Av. Paulista, 2000', '1144444444', 'vendas@empresay.com', 'J', NULL, '98.765.432/0001-88');

-- Obter os IDs das pessoas para relacionamento
-- Isso é um exemplo ilustrativo; no SQL Server Management Studio você precisará verificar com SELECT os IDs atribuídos

-- Inserir usuários vinculados às pessoas físicas (usuários do sistema)
INSERT INTO Usuario (id_usuario, Pessoa_id_pessoa, login, nome, senha)
VALUES
(1, 1, 'maria', 'Maria Silva', '1234'),
(2, 2, 'joao', 'João Souza', '1234'),
(4, 7, 'carolina', 'Ana Carolina', '1234'),
(3, 3, 'ana', 'Ana Lima', '1234');

-- Inserir produtos
INSERT INTO Produto (id_produto, nome, quantidade, preco_venda)
VALUES
(1, 'Teclado', 100, 80.00),
(2, 'Mouse', 200, 40.00),
(3, 'Monitor', 50, 500.00),
(4, 'Notebook', 30, 3000.00),
(5, 'Cabo HDMI', 150, 25.00);

-- Inserir compras feitas por usuários (de fornecedores)
INSERT INTO Compra (id_compra, id_usuario, id_produto, id_fornecedor, data_compra, quantidade, preco_unitario)
VALUES
(1, 1, 1, 4, '2025-04-01', 10, 70.00),
(2, 2, 2, 4, '2025-04-02', 20, 35.00),
(3, 3, 3, 5, '2025-04-03', 5, 450.00),
(4, 3, 4, 5, '2025-04-03', 5, 450.00),
(5, 3, 5, 5, '2025-04-03', 5, 450.00);


-- Inserir vendas feitas para pessoas (clientes)
INSERT INTO Venda (id_venda, id_usuario, id_cliente, id_produto, data_venda, quantidade)
VALUES
(1, 1, 2, 1,  '2025-04-05', 2),
(2, 2, 3, 3, '2025-04-06', 1),
(3, 3, 1, 5, '2025-04-07', 3),
(4, 3, 1, 4, '2025-04-07', 3);
```

Consultas realizadas dos dados inseridos no banco de dados

```
--a) Dados completos de pessoas físicas.
SELECT *
FROM Pessoa
WHERE tipo = 'F';

--b)Dados completos de pessoas jurídicas.
SELECT *
FROM Pessoa
WHERE tipo = 'J';

--c)Movimentações de entrada, com produto, fornecedor, quantidade, preço unitário e valor total.
SELECT
    p.nome AS Fornecedor,
    pr.nome AS Produto,
    c.quantidade,
    c.preco_unitario,
    c.quantidade * c.preco_unitario AS Valor_Total
FROM Compra c
JOIN Pessoa p ON c.id_fornecedor = p.id_pessoa
JOIN Produto pr ON c.id_produto = pr.id_produto;

--d)Movimentações de saída, com produto, comprador, quantidade, preço unitário e valor total.
SELECT
    p.nome AS Cliente,
    pr.nome AS Produto,
    v.quantidade,
    c.preco_unitario,
    v.quantidade * c.preco_unitario AS Valor_Total
FROM Venda v
JOIN Pessoa p ON v.id_cliente = p.id_pessoa
JOIN Produto pr ON v.id_produto = pr.id_produto
JOIN Compra c ON c.id_produto = pr.id_produto;

--e)Valor total das entradas agrupadas por produto.
SELECT
    pr.nome AS Produto,
    SUM(c.quantidade * c.preco_unitario) AS Total_Entrada
FROM Compra c
JOIN Produto pr ON c.id_produto = pr.id_produto
GROUP BY pr.nome;

--f)Valor total das saídas agrupadas por produto.
SELECT
    pr.nome AS Produto,
    SUM(v.quantidade * c.preco_unitario) AS Total_Saida
FROM Venda v
JOIN Produto pr ON v.id_produto = pr.id_produto
JOIN Compra c ON c.id_produto = pr.id_produto
GROUP BY pr.nome;

--g)Operadores que não efetuaram movimentações de entrada (compra).
SELECT u.nome AS Operador
FROM Usuario u
WHERE NOT EXISTS (
    SELECT 1
    FROM Compra c
    WHERE c.id_usuario = u.id_usuario
);

--h)Valor total de entrada, agrupado por operador.
SELECT
    u.nome AS Operador,
```

```
    SUM(c.quantidade * c.preco_unitario) AS Total_Entrada
FROM Compra c
JOIN Usuario u ON c.id_usuario = u.id_usuario
GROUP BY u.nome;

--i)Valor total de saída, agrupado por operador.
SELECT
    u.nome AS Operador,
    SUM(v.quantidade * c.preco_unitario) AS Total_Saida
FROM Venda v
JOIN Usuario u ON v.id_usuario = u.id_usuario
JOIN Compra c ON c.id_produto = v.id_produto
GROUP BY u.nome;

--j)Valor médio de venda por produto, utilizando média ponderada.
SELECT
    pr.nome AS Produto,
    SUM(v.quantidade * c.preco_unitario) * 1.0 / NULLIF(SUM(v.quantidade), 0) AS Media_Ponderada
FROM Venda v
JOIN Produto pr ON v.id_produto = pr.id_produto
JOIN Compra c ON c.id_produto = pr.id_produto
GROUP BY pr.nome;
```

Análise e Conclusão

1) Como são implementadas as diferentes cardinalidades em um banco de dados relacional?

- 1x1 (Um para Um):
  Usa-se uma chave primária em uma tabela que é também chave estrangeira na outra.
- 1xN (Um para Muitos):
  A tabela do lado "muitos" recebe uma chave estrangeira que aponta para a chave primária da tabela do lado "um".
- NxN (Muitos para Muitos):
  Cria-se uma tabela intermediária (ou de associação) com chaves estrangeiras para as duas tabelas envolvidas, e geralmente uma chave primária composta.

2) Que tipo de relacionamento representa o uso de herança em bancos relacionais?

A herança é representada por generalização/especialização, e pode ser modelada de três formas:

- Tabela única (Table per hierarchy): todos os atributos em uma tabela com uma coluna discriminadora.
- Tabelas separadas (Table per subclass): uma tabela para a superclasse e outras para subclasses, relacionadas pela chave primária.
- Tabelas completas (Table per concrete class): cada subclasse tem uma tabela independente, repetindo os dados da superclasse.

A mais comum e recomendada em bancos relacionais é a segunda (tabelas separadas).

3) Como o SQL Server Management Studio (SSMS) melhora a produtividade no gerenciamento de banco?

- Interface gráfica facilita criação de tabelas, relacionamentos, procedures, etc.
- Permite execução de scripts SQL com sugestões e histórico.
- Traz ferramentas para backup, restore, tuning e análise de performance.
- Possui diagramas visuais para modelagem e análise de relacionamentos.
- Integração com jobs, agendamentos e permissões, agilizando a administração.

4) Diferenças entre `SEQUENCE` e `IDENTITY`

Característica	SEQUENCE	IDENTITY
Definição	Objeto independente	Propriedade da coluna
Reutilizável	Sim (pode ser usado por várias tabelas)	Não (vinculado à tabela)
Controle de valor	Manual (via `NEXT VALUE FOR`)	Automático ao inserir
Restauração	Mais flexível para reiniciar	Menos controle

5) Importância das chaves estrangeiras para a consistência do banco

- Garantem integridade referencial, impedindo registros órfãos.
- Controlam dependência entre tabelas, evitando inserções e deleções indevidas.
- Mantêm o relacionamento lógico entre entidades, refletindo as regras do negócio.
- Podem ser usadas com ON DELETE e ON UPDATE para ações automáticas.

6) Operadores da Álgebra Relacional e do Cálculo Relacional

Álgebra Relacional (conjuntista e operacional):

SELEÇÃO (σ) → WHERE
PROJEÇÃO (π) → SELECT
UNIÃO (∪) → UNION
INTERSEÇÃO (∩) → INTERSECT
DIFERENÇA (-) → EXCEPT
JUNÇÃO (⋈) → JOIN
RENOMEAÇÃO (ρ) → AS

Cálculo Relacional (baseado em lógica):

- Usa variáveis e expressões lógicas (ex: ∀, ∃)
- É mais declarativo, parecido com subconsultas (EXISTS, ALL, ANY)

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

Feito com a cláusula GROUP BY, que agrupa linhas com base em valores comuns de uma ou mais colunas.

Requisito obrigatório:
Toda coluna no SELECT que não usa função agregadora (SUM, AVG, etc.) deve estar presente no GROUP BY.