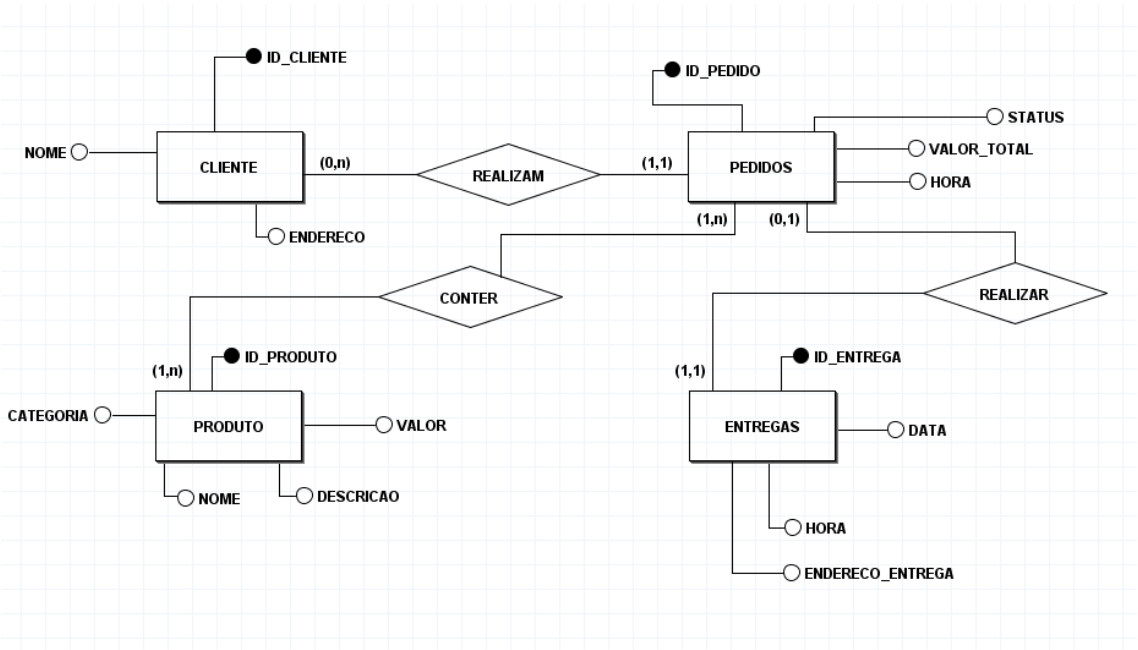
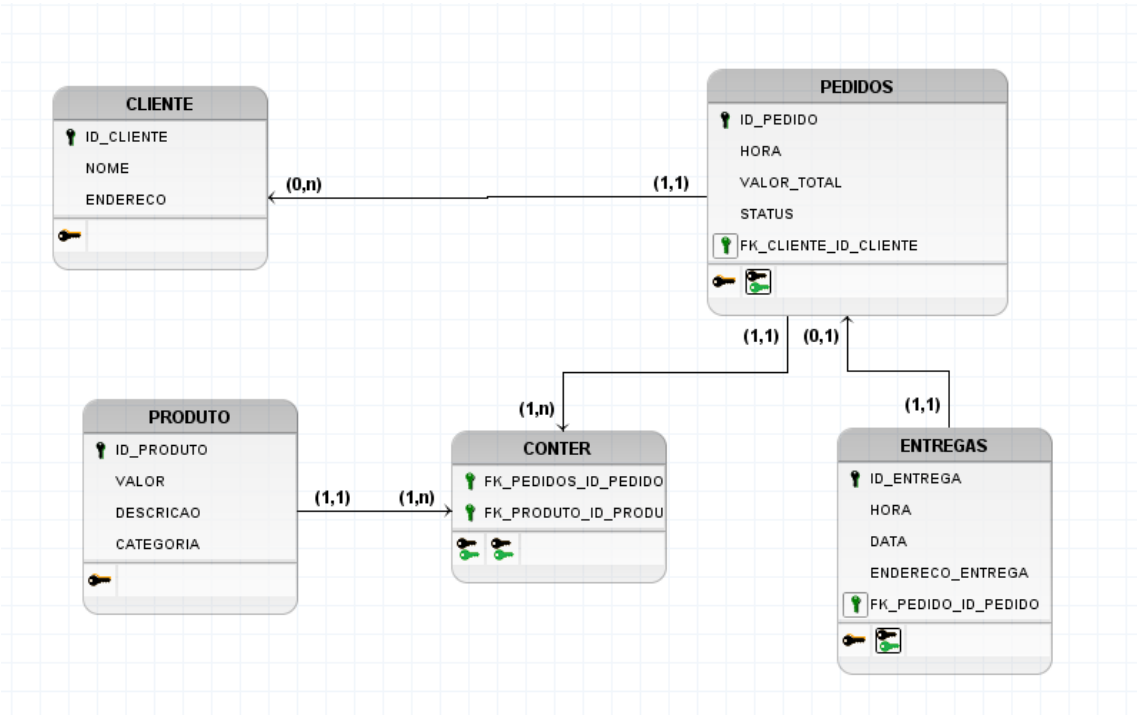
- TABELA CONCEITUAL -



- TABELA LÓGICA –



CREATE DATABASE cafeteria;

```
USE cafeteria;

-- 1. Tabela CLIENTE
CREATE TABLE CLIENTE (
  ID_CLIENTE INT AUTO_INCREMENT PRIMARY KEY,
  NOME VARCHAR(100) NOT NULL,
  ENDERECO VARCHAR(255) NOT NULL
);

-- 2. Tabela PRODUTO
CREATE TABLE PRODUTO (
  ID_PRODUTO INT AUTO_INCREMENT PRIMARY KEY,
  NOME VARCHAR(100) NOT NULL,
  DESCRICAO VARCHAR(255),
  VALOR DECIMAL(10, 2) NOT NULL,
  CATEGORIA VARCHAR(50) NOT NULL
);

-- 3. Tabela PEDIDOS (depende de CLIENTE)
CREATE TABLE PEDIDOS (
  ID_PEDIDO INT AUTO_INCREMENT PRIMARY KEY,
  HORA TIME NOT NULL,
  VALOR_TOTAL DECIMAL(10, 2) NOT NULL,
  STATUS VARCHAR(50) NOT NULL, -- Ex: 'Em Processamento', 'Pronto', 'Entregue', 'Cancelado'
  FK_ID_CLIENTE INT NOT NULL,
  FOREIGN KEY (FK_ID_CLIENTE) REFERENCES CLIENTE(ID_CLIENTE)
);
```

```
-- 4. Tabela ENTREGAS (depende de PEDIDOS)
CREATE TABLE ENTREGAS (
  ID_ENTREGA INT AUTO_INCREMENT PRIMARY KEY,
  DATA DATE NOT NULL,
  HORA TIME NOT NULL,
  ENDERECO_ENTREGA VARCHAR(255) NOT NULL,
  FK_ID_PEDIDO INT NOT NULL,
  FOREIGN KEY (FK_ID_PEDIDO) REFERENCES PEDIDOS(ID_PEDIDO)
);

-- 5. Tabela CONTER (Tabela Associativa PEDIDO_PRODUTO - depende de PEDIDOS e PRODUTO)
CREATE TABLE CONTER (
  FK_PEDIDOS_ID_PEDIDO INT NOT NULL,
  FK_PRODUTO_ID_PRODUTO INT NOT NULL,
  QUANTIDADE INT NOT NULL,
  PRIMARY KEY (FK_PEDIDOS_ID_PEDIDO, FK_PRODUTO_ID_PRODUTO), -- Chave Primária Composta
  FOREIGN KEY (FK_PEDIDOS_ID_PEDIDO) REFERENCES PEDIDOS(ID_PEDIDO),
  FOREIGN KEY (FK_PRODUTO_ID_PRODUTO) REFERENCES PRODUTO(ID_PRODUTO)
);

-- Comandos para Inserção de Dados

-- Inserindo dados na tabela CLIENTE (min 10 registros)
INSERT INTO CLIENTE (NOME, ENDERECO) VALUES
('Ana Silva', 'Rua das Flores, 123 - Centro'),
```

('Bruno Costa', 'Av. Principal, 45 - Bairro Novo'),

('Carla Dias', 'Rua da Paz, 678 - Vila Feliz'),

('Daniel Rocha', 'Travessa dos Sonhos, 90 - Cidade Velha'),

('Eduarda Lima', 'Praça da Liberdade, 10 - Bela Vista'),

('Felipe Mendes', 'Rua do Comércio, 202 - Centro'),

('Gabriela Nunes', 'Av. do Sol, 303 - Alto da Colina'),

('Henrique Pereira', 'Rua do Porto, 404 - Beira Mar'),

('Isabela Santos', 'Alameda das Árvores, 505 - Jardim Botânico'),

('João Oliveira', 'Rua da Saudade, 606 - Campo Grande'),

('Karen Souza', 'Rua das Palmeiras, 707 - Horto'),

('Luiz Fernandes', 'Av. dos Anjos, 808 - Céu Azul');

-- Inserindo dados na tabela PRODUTO (min 10 registros)

INSERT INTO PRODUTO (NOME, DESCRICAO, VALOR, CATEGORIA) VALUES

('Café Expresso', 'Café puro e forte', 5.50, 'Bebida Quente'),

('Capuccino', 'Café com leite vaporizado e espuma', 8.00, 'Bebida Quente'),

('Pão de Queijo', 'Tradicional pão de queijo mineiro', 4.00, 'Salgado'),

('Bolo de Cenoura', 'Fatia de bolo de cenoura com cobertura de chocolate', 12.00, 'Doce'),

('Suco de Laranja', 'Suco natural de laranja', 9.00, 'Bebida Gelada'),

('Croissant', 'Croissant amanteigado simples', 7.00, 'Salgado'),

('Latte', 'Café com leite e uma fina camada de espuma', 8.50, 'Bebida Quente'),

('Muffin de Blueberry', 'Muffin com mirtilos frescos', 10.00, 'Doce'),

('Água Mineral', 'Garrafa de água sem gás', 4.00, 'Bebida Gelada'),

('Torta de Limão', 'Fatia de torta de limão com merengue', 15.00, 'Doce'),

('Brigadeiro', 'Doce de chocolate tradicional brasileiro', 3.00, 'Doce'),

('Baguete na Chapa', 'Pão baguete na chapa com manteiga', 6.00, 'Salgado');

-- Inserindo dados na tabela PEDIDOS (min 10 registros, usando FK_ID_CLIENTE válidos)

INSERT INTO PEDIDOS (HORA, VALOR_TOTAL, STATUS, FK_ID_CLIENTE) VALUES

('10:00:00', 20.50, 'Em Processamento', 1), -- Ana

('10:15:00', 12.00, 'Pronto', 2), -- Bruno

('10:30:00', 30.00, 'Entregue', 3), -- Carla

('10:45:00', 8.50, 'Em Processamento', 4), -- Daniel

('11:00:00', 18.00, 'Pronto', 5), -- Eduarda

('11:15:00', 7.00, 'Cancelado', 6), -- Felipe

('11:30:00', 25.50, 'Entregue', 7), -- Gabriela

('11:45:00', 11.50, 'Em Processamento', 8), -- Henrique

('12:00:00', 40.00, 'Pronto', 9), -- Isabela

('12:15:00', 16.00, 'Entregue', 10), -- João

('12:30:00', 9.50, 'Em Processamento', 1), -- Ana (novo pedido)

('12:45:00', 22.00, 'Pronto', 2); -- Bruno (novo pedido)

-- Inserindo dados na tabela ENTREGAS (min 10 registros, usando FK_ID_PEDIDO válidos de pedidos entregues ou a serem entregues)

-- Note que nem todo pedido precisa ter uma entrega, de acordo com o modelo (0,1)

INSERT INTO ENTREGAS (DATA, HORA, ENDERECO_ENTREGA, FK_ID_PEDIDO) VALUES

('2025-07-29', '10:45:00', 'Rua das Flores, 123', 1), -- Pedido 1 (Ana)

('2025-07-29', '11:00:00', 'Av. Principal, 45', 2), -- Pedido 2 (Bruno)

('2025-07-29', '11:15:00', 'Rua da Paz, 678', 3), -- Pedido 3 (Carla)

('2025-07-29', '11:30:00', 'Travessa dos Sonhos, 90', 4), -- Pedido 4 (Daniel)

('2025-07-29', '11:45:00', 'Praça da Liberdade, 10', 5), -- Pedido 5 (Eduarda)

('2025-07-29', '12:00:00', 'Av. do Sol, 303', 7), -- Pedido 7 (Gabriela)

('2025-07-29', '12:15:00', 'Rua do Porto, 404', 8), -- Pedido 8 (Henrique)

('2025-07-29', '12:30:00', 'Alameda das Árvores, 505', 9), -- Pedido 9 (Isabela)

('2025-07-29', '12:45:00', 'Rua da Saudade, 606', 10), -- Pedido 10 (João)

('2025-07-29', '13:00:00', 'Rua das Palmeiras, 707', 11), -- Pedido 11 (Ana - novo pedido)

('2025-07-29', '13:15:00', 'Av. dos Anjos, 808', 12); -- Pedido 12 (Bruno - novo pedido)

-- Inserindo dados na tabela CONTER (min 10 registros, usando FK_PEDIDOS_ID_PEDIDO e FK_PRODUTO_ID_PRODUTO válidos)

INSERT INTO CONTER (FK_PEDIDOS_ID_PEDIDO, FK_PRODUTO_ID_PRODUTO, QUANTIDADE) VALUES

(1, 1, 2), -- Pedido 1: 2 Cafés Expressos

(1, 4, 1), -- Pedido 1: 1 Bolo de Cenoura

(2, 2, 1), -- Pedido 2: 1 Capuccino

(2, 3, 2), -- Pedido 2: 2 Pães de Queijo

(3, 5, 3), -- Pedido 3: 3 Sucos de Laranja

(4, 7, 1), -- Pedido 4: 1 Latte

(5, 8, 1), -- Pedido 5: 1 Muffin de Blueberry

(6, 6, 1), -- Pedido 6: 1 Croissant (pedido cancelado, mas registro existe)

(7, 10, 2), -- Pedido 7: 2 Tortas de Limão

(8, 1, 1), -- Pedido 8: 1 Café Expresso

(9, 2, 2), -- Pedido 9: 2 Capuccinos

(9, 11, 4), -- Pedido 9: 4 Brigadeiros

(10, 12, 1), -- Pedido 10: 1 Baguete na Chapa

(11, 1, 1); -- Pedido 11: 1 Café Expresso (novo pedido)

```
-- Consulta 1.1: Seleciona o nome e endereço de clientes com 'Silva' no nome.
SELECT NOME, ENDERECO
FROM CLIENTE
WHERE NOME LIKE '%Silva%';

-- Consulta 1.2: Seleciona todos os pedidos que estão com o status 'Em Processamento'.
SELECT *
FROM PEDIDOS
WHERE STATUS = 'Em Processamento';

-- Consulta 2.1: Conta quantos pedidos cada cliente fez, ordenando pelo cliente que fez mais pedidos.
SELECT C.NOME, COUNT(P.ID_PEDIDO) AS TotalPedidos
FROM CLIENTE AS C
JOIN PEDIDOS AS P ON C.ID_CLIENTE = P.FK_ID_CLIENTE
GROUP BY C.NOME
ORDER BY TotalPedidos DESC;

-- Consulta 2.2: Calcula o valor total de pedidos por status, ordenando pelo status.
SELECT STATUS, SUM(VALOR_TOTAL) AS SomaTotalPedidos
FROM PEDIDOS
GROUP BY STATUS
ORDER BY STATUS ASC;

-- Consulta 3.1: Calcula o valor de cada produto com um aumento de 10% (simulando reajuste).
SELECT NOME, VALOR AS ValorOriginal, (VALOR * 1.10) AS ValorComAumento10PorCento
```

```
FROM PRODUTO;


-- Consulta 3.2: Calcula o lucro potencial de cada item em um pedido (considerando um custo fixo de R$ 2,00 por item), se aplicável (requer join com CONTER para ter QUANTIDADE).

-- Este exemplo calcula o valor total do item no pedido menos um custo simulado por unidade.

-- Consulta 3.2: Calcula o lucro potencial de cada item em um pedido (considerando um custo fixo de R$ 2,00 por item), se aplicável.

SELECT

  Pr.NOME AS NomeProduto, -- CORRIGIDO: Era P.NOME, agora é Pr.NOME

  C.QUANTIDADE,

  (Pr.VALOR * C.QUANTIDADE) AS ValorTotalDoItem,

  ((Pr.VALOR - 2.00) * C.QUANTIDADE) AS LucroPotencialEstimado

FROM CONTER AS C

JOIN PRODUTO AS Pr ON C.FK_PRODUTO_ID_PRODUTO = Pr.ID_PRODUTO

JOIN PEDIDOS AS P ON C.FK_PEDIDOS_ID_PEDIDO = P.ID_PEDIDO

LIMIT 10;


-- Consulta 4.1: Seleciona produtos cujo valor é exatamente R$ 8.00.

SELECT NOME, VALOR

FROM PRODUTO

WHERE VALOR = 8.00;


-- Consulta 4.2: Seleciona pedidos com valor total maior que R$ 20.00.

SELECT ID_PEDIDO, VALOR_TOTAL, STATUS

FROM PEDIDOS

WHERE VALOR_TOTAL > 20.00;
```

```
-- Consulta 4.3: Seleciona clientes cujo ID é diferente de 5.

SELECT NOME, ID_CLIENTE

FROM CLIENTE

WHERE ID_CLIENTE != 5;


-- Consulta 5.1: Seleciona produtos que são 'Doce' E têm valor maior que R$ 10.00.

SELECT NOME, CATEGORIA, VALOR

FROM PRODUTO

WHERE CATEGORIA = 'Doce' AND VALOR > 10.00;


-- Consulta 5.2: Seleciona pedidos que estão 'Pronto' OU 'Entregue'.

SELECT ID_PEDIDO, STATUS, VALOR_TOTAL

FROM PEDIDOS

WHERE STATUS = 'Pronto' OR STATUS = 'Entregue';


-- Consulta 5.3: Seleciona clientes cujo nome começa com 'A' OU 'B' E o endereço contém 'Centro'.

SELECT NOME, ENDERECO

FROM CLIENTE

WHERE (NOME LIKE 'A%' OR NOME LIKE 'B%') AND ENDERECO LIKE '%Centro%';


-- Consulta 6.1: Seleciona pedidos que NÃO estão com o status 'Cancelado'.

SELECT ID_PEDIDO, STATUS, VALOR_TOTAL

FROM PEDIDOS

WHERE NOT STATUS = 'Cancelado';


-- Consulta 6.2: Seleciona produtos que NÃO são da categoria 'Bebida Quente' E NÃO tem valor menor ou igual a 5.00.
```

```
SELECT NOME, CATEGORIA, VALOR

FROM PRODUTO

WHERE NOT CATEGORIA = 'Bebida Quente' AND NOT VALOR <= 5.00;


-- Consulta 7.1: Seleciona pedidos com valor total entre R$ 10.00 e R$ 25.00.

SELECT ID_PEDIDO, VALOR_TOTAL, STATUS

FROM PEDIDOS

WHERE VALOR_TOTAL BETWEEN 10.00 AND 25.00;


-- Consulta 7.2: Seleciona produtos cujo nome começa com 'Ca' (case-insensitive, dependendo da collation do MySQL).

SELECT NOME, CATEGORIA

FROM PRODUTO

WHERE NOME LIKE 'Ca%';


-- Consulta 7.3: Seleciona clientes com ID 1, 3 ou 5.

SELECT NOME, ID_CLIENTE

FROM CLIENTE

WHERE ID_CLIENTE IN (1, 3, 5);


-- Consulta 8.1: Calcula o valor total de todos os pedidos registrados.

SELECT SUM(VALOR_TOTAL) AS SomaTotalDeTodosOsPedidos

FROM PEDIDOS;


-- Consulta 8.2: Calcula o valor médio dos produtos.

SELECT AVG(VALOR) AS ValorMedioProdutos

FROM PRODUTO;
```

```
-- Consulta 8.3: Encontra o valor do pedido mais caro e do pedido mais barato.
SELECT MAX(VALOR_TOTAL) AS PedidoMaisCaro, MIN(VALOR_TOTAL) AS PedidoMaisBarato
FROM PEDIDOS;

-- Consulta 9.1: Seleciona as entregas que estão agendadas para a data de hoje (considerando a data atual do servidor).
SELECT ID_ENTREGA, DATA, HORA, ENDERECO_ENTREGA
FROM ENTREGAS
WHERE DATA = CURDATE(); -- CURDATE() é similar a DATE(NOW()) para a data atual

-- Consulta 9.2: Conta o número de entregas por ano.
SELECT YEAR(DATA) AS AnoDaEntrega, COUNT(ID_ENTREGA) AS TotalDeEntregas
FROM ENTREGAS
GROUP BY AnoDaEntrega
ORDER BY AnoDaEntrega;

-- Consulta 10.1: Clientes que fizeram pedidos com valor total acima da média de todos os pedidos.
SELECT NOME, ENDERECO
FROM CLIENTE
WHERE ID_CLIENTE IN (
  SELECT FK_ID_CLIENTE
  FROM PEDIDOS
  WHERE VALOR_TOTAL > (SELECT AVG(VALOR_TOTAL) FROM PEDIDOS)
  GROUP BY FK_ID_CLIENTE -- Agrupamento para pegar IDs de clientes distintos
);
```

```
-- Consulta 10.2: Produtos que nunca foram vendidos (não aparecem em nenhum pedido).
SELECT NOME, DESCRICAO
FROM PRODUTO
WHERE ID_PRODUTO NOT IN (
  SELECT FK_PRODUTO_ID_PRODUTO
  FROM CONTER
);

-- Consulta 10.3: Pedidos que contêm o produto 'Café Expresso' e que foram feitos por clientes do 'Centro'.
SELECT P.ID_PEDIDO, P.VALOR_TOTAL, C.NOME AS NomeCliente, C.ENDERECO AS EnderecoCliente
FROM PEDIDOS AS P
JOIN CLIENTE AS C ON P.FK_ID_CLIENTE = C.ID_CLIENTE
WHERE P.ID_PEDIDO IN (
  SELECT FK_PEDIDOS_ID_PEDIDO
  FROM CONTER
  WHERE FK_PRODUTO_ID_PRODUTO = (SELECT ID_PRODUTO FROM PRODUTO WHERE NOME = 'Café Expresso')
) AND C.ENDERECO LIKE '%Centro%';

-- Consulta 11.1: Visualiza pedidos e os nomes dos clientes que os fizeram.
SELECT
  P.ID_PEDIDO,
  P.VALOR_TOTAL,
  P.STATUS,
  C.NOME AS NomeCliente,
  C.ENDERECO AS EnderecoCliente
```

```
FROM PEDIDOS AS P
JOIN CLIENTE AS C ON P.FK_ID_CLIENTE = C.ID_CLIENTE;

-- Consulta 11.2: Visualiza todos os itens de cada pedido, mostrando o nome do produto e a quantidade.
SELECT
  P.ID_PEDIDO,
  Pr.NOME AS NomeProduto,
  Ct.QUANTIDADE,
  Pr.VALOR AS ValorUnitario
FROM PEDIDOS AS P
JOIN CONTER AS Ct ON P.ID_PEDIDO = Ct.FK_PEDIDOS_ID_PEDIDO
JOIN PRODUTO AS Pr ON Ct.FK_PRODUTO_ID_PRODUTO = Pr.ID_PRODUTO;

-- Consulta 11.3: Visualiza detalhes das entregas, incluindo informações do pedido associado e do cliente.
SELECT
  E.ID_ENTREGA,
  E.DATA,
  E.HORA AS HoraEntrega,
  E.ENDERECO_ENTREGA,
  P.ID_PEDIDO,
  P.VALOR_TOTAL AS ValorPedido,
  C.NOME AS NomeCliente
FROM ENTREGAS AS E
JOIN PEDIDOS AS P ON E.FK_ID_PEDIDO = P.ID_PEDIDO
JOIN CLIENTE AS C ON P.FK_ID_CLIENTE = C.ID_CLIENTE;
```

```
-- Consulta 12.1: INNER JOIN - Pedidos que TÊM entregas (somente os correspondentes em ambas as tabelas).
SELECT
  P.ID_PEDIDO,
  P.VALOR_TOTAL,
  E.DATA AS DataEntrega,
  E.ENDERECO_ENTREGA
FROM PEDIDOS AS P
INNER JOIN ENTREGAS AS E ON P.ID_PEDIDO = E.FK_ID_PEDIDO;

-- Consulta 12.2: LEFT JOIN - Todos os pedidos e suas respectivas entregas, se houver.
-- Mostrará pedidos sem entrega com colunas de entrega como NULL.
SELECT
  P.ID_PEDIDO,
  P.STATUS,
  P.VALOR_TOTAL,
  E.ID_ENTREGA,
  E.DATA AS DataEntrega,
  E.ENDERECO_ENTREGA
FROM PEDIDOS AS P
LEFT JOIN ENTREGAS AS E ON P.ID_PEDIDO = E.FK_ID_PEDIDO;

-- Consulta 12.3: RIGHT JOIN - Todas as entregas e seus respectivos pedidos.
-- Neste modelo, como FK_ID_PEDIDO em ENTREGAS é NOT NULL, cada entrega SEMPRE terá um pedido,
-- então o RIGHT JOIN aqui se comportará como um INNER JOIN para essas duas tabelas especificamente.
-- É útil para mostrar um cenário onde a tabela da direita pode ter registros "órfãos" no LEFT JOIN.
```

```
SELECT
  E.ID_ENTREGA,
  E.DATA AS DataEntrega,
  E.ENDERECO_ENTREGA,
  P.ID_PEDIDO,
  P.STATUS AS StatusPedido
FROM PEDIDOS AS P
RIGHT JOIN ENTREGAS AS E ON P.ID_PEDIDO = E.FK_ID_PEDIDO;

-- Consulta 12.4: LEFT JOIN para mostrar todos os produtos e quantos pedidos eles estão (mesmo se zero).
SELECT
  Pr.NOME AS NomeProduto,
  COUNT(Ct.FK_PEDIDOS_ID_PEDIDO) AS TotalVendidoEmPedidos
FROM PRODUTO AS Pr
LEFT JOIN CONTER AS Ct ON Pr.ID_PRODUTO = Ct.FK_PRODUTO_ID_PRODUTO
GROUP BY Pr.NOME
ORDER BY TotalVendidoEmPedidos DESC;
```

1. Consultas com SELECT e WHERE

- **Explicação:** Este grupo de consultas demonstra o uso fundamental de SELECT para especificar quais colunas de dados queremos visualizar e WHERE para aplicar filtros, retornando apenas os registros que satisfazem condições específicas. Isso é essencial para focar em subconjuntos de dados relevantes para a análise.

---

2. Consultas com GROUP BY e ORDER BY com funções de agregação

- **Explicação:** Aqui, exploramos a capacidade de sumarizar dados utilizando GROUP BY para agrupar registros com valores comuns em uma ou mais colunas. As funções de agregação (como COUNT, SUM) são aplicadas a esses grupos, fornecendo métricas consolidadas. O ORDER BY é utilizado para organizar os resultados de forma crescente ou decrescente, facilitando a visualização e a análise de tendências ou rankings.

**3. Consultas com operadores aritméticos (+, -, , /)*

- **Explicação:** Este conjunto de consultas ilustra como os operadores aritméticos podem ser empregados diretamente nas instruções SELECT para realizar cálculos sobre os dados. Isso permite criar novas colunas calculadas dinamicamente, como simulações de aumento de preço ou estimativas de lucro, sem a necessidade de alterar os dados originais no banco.

---

4. Consultas com operadores de comparação (=, !=, <, >, etc.)

- **Explicação:** Neste critério, utilizamos operadores de comparação no WHERE para filtrar registros com base em relações de igualdade, diferença, maior que, menor que, entre outras. Essa técnica é crucial para refinar a seleção de dados, permitindo que a consulta retorne exatamente os registros que se encaixam em critérios de comparação específicos.

---

5. Consultas com operadores lógicos (AND, OR)

- **Explicação:** As consultas neste grupo demonstram a combinação de múltiplas condições de filtragem usando os operadores lógicos AND e OR. O AND exige que todas as condições conectadas sejam verdadeiras, enquanto o OR requer que pelo menos uma das condições seja verdadeira. Essa capacidade de encadear lógicas complexas é vital para criar filtros sofisticados e precisos.

---

6. Consultas com operadores lógicos e negação (NOT)

- **Explicação:** Este grupo foca no uso do operador NOT em conjunto com outras condições e operadores lógicos. O NOT inverte o resultado de uma expressão booleana, permitindo selecionar registros que *não* atendem a

uma determinada condição. É uma ferramenta poderosa para exclusões e para filtrar dados que não se enquadram em certos critérios.

---

7. Consultas com operadores auxiliares (IS NULL, BETWEEN, LIKE, IN)

- **Explicação:** Aqui, exploramos operadores auxiliares que facilitam a filtragem de dados de maneiras específicas. BETWEEN é usado para verificar se um valor está dentro de um intervalo. LIKE permite buscas por padrões de texto, útil para pesquisas flexíveis. IN verifica se um valor pertence a uma lista de opções. IS NULL (não usado diretamente nos exemplos fornecidos, mas fundamental) é para identificar registros com valores ausentes.

---

8. Consultas com funções de agregação (SUM(), AVG(), etc.)

- **Explicação:** Este critério reitera a importância das funções de agregação (SUM, AVG, MAX, MIN, COUNT) para realizar cálculos sumarizados sobre conjuntos de dados. Elas são essenciais para obter métricas gerais, como totais, médias, valores máximos e mínimos, fornecendo uma visão consolidada e quantitativa dos dados.

---

9. Consultas com funções de datas (NOW(), DATE(), YEAR(), etc.)

- **Explicação:** As consultas neste grupo demonstram a manipulação de dados de data e hora utilizando funções SQL específicas. Funções como CURDATE() (para a data atual) e YEAR() (para extrair o ano de uma data) são cruciais para filtragens baseadas em tempo e para analisar dados temporalmente, como o número de entregas por ano ou agendamentos para um dia específico.

---

10. Sub-consultas com agrupamento e união de dados

- **Explicação:** Este segmento aborda o uso avançado de subconsultas, onde uma consulta é aninhada dentro de outra. As subconsultas são usadas para resolver problemas que requerem múltiplos passos de processamento, como encontrar clientes que fizeram pedidos acima da média ou produtos que nunca foram vendidos. O agrupamento dentro das subconsultas ajuda a refinar os resultados intermediários, e a "união de

dados" se refere à capacidade de correlacionar informações entre tabelas de forma indireta através dessas construções aninhadas.

---

11. Consultas com JOIN e visualização de tabelas

- **Explicação:** As consultas com JOIN são fundamentais para combinar dados de duas ou mais tabelas relacionadas. Elas permitem "unir" as informações com base em colunas comuns (as chaves primárias e estrangeiras), proporcionando uma visão consolidada que não seria possível observando as tabelas isoladamente. Isso é crucial para entender o contexto completo dos dados, como ver os nomes dos clientes associados aos seus pedidos.

---

12. Consultas com tipos de JOIN: INNER, LEFT, RIGHT

- **Explicação:** Este critério aprofunda o uso de JOIN ao explorar seus diferentes tipos: INNER JOIN, LEFT JOIN e RIGHT JOIN. O INNER JOIN retorna apenas as linhas que têm correspondência em ambas as tabelas. O LEFT JOIN retorna todas as linhas da tabela da esquerda e as correspondências da direita (preenchendo com NULL onde não há correspondência). O RIGHT JOIN faz o oposto, retornando todas as linhas da tabela da direita e as correspondências da esquerda. A escolha do tipo de JOIN depende da necessidade de incluir ou excluir registros "órfãos" (sem correspondência) de um dos lados do relacionamento, oferecendo flexibilidade na recuperação de dados.

SALIENTO NESTE DOCUMENTO QUE FOI UTILIZADO A FERRAMENTA DE INTELIGÊNCIA ARTIFICIAL **“ GEMINI “.**

INFORMO, QUE, DIFERENTE DAS REGRAS ENSINADAS NA SALA DE AULA, FOI CRIADO UMA LÓGICA SUGERIDA POR AMBOS, CHATGPT E GEMINI, ONDE A CHAVE ESTRANGEIRA NÃO DEVERIA FICAR NA ENTIDADE CLIENTE, POIS LIMITARIA O NÚMERO DE PEDIDOS QUE UM CLIENTE PODERIA REALIZAR, ALTERANDO ASSIM, A REGRA DE NEGÓCIOS, SENDO MIGRADA A CHAVE ESTRANGEIRA PARA A ENTIDADE PEDIDOS.

INFORMO QUE O USO DAS FERRAMENTAS DE INTELIGÊNCIA ARTIFICIAL TEVE O INTUITO APENAS DE DAR CELERIDADE, E QUE ESTE ALUNO ABSORVEU TODO O CONTEÚDO! ENTENDE, QUE, NA BUSCA PELO APRENDIZADO, AS FERRAMENTAS DISPONÍVEIS REPRESENTAM UM AUXÍLIO IMPORTANTE PARA O AMBIENTE DE DESENVOLVIMENTO.

3º SGT PM 97349 MATHEUS HENRIQUE **MOTTA** RIBEIRO

ALUNO I CADSPM/2025