INSTITUTO SUPERIOR DO LITORAL DO PARANÁ

Prof. Luiz Efigênio

Administração de Banco de Dados II
Visão Geral e Conceitos Avançados de SQL

Disciplina: Administração de Banco de Dados II
Professor: Luiz Efigênio
Material de Revisão — Conteúdo até INNER JOIN

Instruções: responda cada questão com clareza. Para as questões que pedem consultas SQL, escreva a instrução SQL completa, usando a sintaxe padrão (ANSI SQL) e explicando brevemente a lógica quando necessário. Use os esquemas de tabelas fornecidos nas questões que demandam consulta.

Questões

Parte A — Fundamentos e consultas básicas

1. **DDL / DML**
   Explique a diferença entre **DDL**, **DML** e no SQL. Para cada categoria, dê um exemplo de comando real (um por categoria) e indique quando cada um deve ser usado.

2. **WHERE vs HAVING**
   Qual a diferença entre a cláusula `WHERE` e a cláusula `HAVING`? Apresente um exemplo (com `GROUP BY`) em que o uso de `HAVING` é obrigatório.

3. **Consulta com filtro e ordenação**
   Dada a tabela `Funcionarios(id, nome, salario, departamento_id)`, escreva uma consulta SQL que retorne o `nome` e o `salario` dos funcionários com salário maior que `3000`, ordenados em ordem decrescente de salário.

4. **Agregação e expressão**
   Considere a tabela `Vendas(id, produto, quantidade, valor_unitario)`. Escreva uma consulta que apresente para cada `produto` o **faturamento total** (soma de `quantidade * valor_unitario`) e ordene pelo faturamento decrescente.

5. **GROUP BY com HAVING**
   Usando a tabela `Vendas` do exercício anterior, escreva uma consulta que retorne apenas os produtos cujo faturamento total seja maior que `5000`. Explique por que `HAVING` é usado nesse caso.

Parte B — JOINs (mínimo 7 questões focadas em JOINs)

Esquemas de referência para as próximas questões:

- `Clientes(id, nome, email)`
- `Pedidos(id, cliente_id, data_pedido, total_pedido)`
- `Produtos(id, nome, preco)`
- `ItensPedido(id, pedido_id, produto_id, quantidade, preco_unitario)`

6. **INNER JOIN — básico**
   Escreva uma consulta que liste **id do pedido**, **data_pedido** e **nome do cliente** para todos os pedidos (apresente todas as colunas em inglês ou português, conforme preferir). Use `INNER JOIN` entre `Pedidos` e `Clientes`.

7. **INNER JOIN — múltiplas tabelas**
   Escreva uma consulta que exiba, para cada item de pedido: `pedido_id`, `data_pedido`, `nome_cliente`, `nome_produto`, `quantidade`, e `valor_total_linha` (quantidade × preco_unitario). Utilize `INNER JOIN` unindo as quatro tabelas necessárias.

8. **LEFT JOIN — encontrar não correspondentes**
   Escreva uma consulta que retorne todos os clientes (`Clientes`) e, quando existir, o `id` do último pedido (`Pedidos.id`) — mas incluía também clientes que **ainda não**

tenham feito pedidos. Use `LEFT JOIN` e explique por que `LEFT JOIN` é adequado aqui.

9. **RIGHT JOIN — conceito e equivalência**
   Escreva uma consulta com `RIGHT JOIN` que liste todos os produtos (`Produtos`) e os pedidos nos quais apareceram (caso existam). Em seguida, reescreva a mesma lógica utilizando `LEFT JOIN` (invertendo a ordem das tabelas) — explique por que as duas consultas produzem o mesmo resultado em um SGBD que suporte `RIGHT JOIN`.

10. **FULL OUTER JOIN — comportamento e emulação**
    Explique o comportamento de um `FULL OUTER JOIN`. Em ambientes (como MySQL) que não suportam `FULL OUTER JOIN`, descreva (em SQL) uma estratégia para **emular** um `FULL OUTER JOIN` entre `Clientes` e `Pedidos` mostrando todas as combinações (clientes com pedidos, pedidos sem cliente e clientes sem pedido). (Não é necessário executar; apresente a SQL via `UNION`).

11. **JOIN com condições adicionais (theta join)**
    Considere que exista uma tabela `Descontos(produto_id, inicio, fim, porcentagem)`. Escreva uma consulta que junte `ItensPedido` com `Descontos` retornando os itens cujo `data_pedido` (provinda de `Pedidos`) esteja entre `inicio` e `fim` do desconto. Mostre como a condição temporal é colocada no `ON` do `JOIN`.

12. **Ambiguidade de colunas e aliases**
    Suponha que tanto `Clientes` quanto `Pedidos` tenham uma coluna chamada `id`. Escreva uma consulta que selecione `Clientes.id` como `cliente_id`, `Pedidos.id` como `pedido_id`, e `Clientes.nome`, utilizando aliases para tabelas e colunas para evitar ambiguidade.

13. **Agregação com JOIN**
    Usando `Clientes`, `Pedidos` e `ItensPedido`, escreva uma consulta que retorne para cada cliente: `cliente_id`, `nome`, e `faturamento_total_cliente` (soma de `quantidade * preco_unitario` em todos os pedidos do cliente). Ordene pelo faturamento em ordem decrescente.

14. **LEFT JOIN e tratamento de NULL (COALESCE)**
    Escreva uma consulta que liste todos os produtos e a quantidade total vendida de cada produto (soma de `quantidade` em `ItensPedido`), retornando `0` quando o

produto não tiver vendas. Use `LEFT JOIN` e `COALESCE` para substituir `NULL` por `0`.

15. **Filtro após JOIN + ordenação**
    Escreva uma consulta que utilize `INNER JOIN` entre `Pedidos` e `Clientes` para retornar todos os pedidos realizados entre `2025-01-01` e `2025-06-30`, exibindo `pedido_id`, `data_pedido`, `cliente_nome` e `total_pedido`. Ordene por `data_pedido` crescente.

Observação final (do professor):
Responda cada questão com a instrução SQL e, quando pertinente, adicione uma breve explicação (1–2 linhas) sobre a lógica empregada. Para as questões teóricas (1 e 2), seja sucinto e objetivo. Nas questões de JOIN, sempre prefira usar **aliases** para tornar as consultas claras e evitar ambiguidade.

Assinatura,
Prof. Luiz Carlos Efigênio

```
-- ========================
-- CRIAÇÃO DO BANCO E TABELAS
-- ========================
DROP DATABASE IF EXISTS adbII_revisao;
CREATE DATABASE adbII_revisao;
USE adbII_revisao;

-- Clientes
CREATE TABLE Clientes (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE
);
```

```
-- Pedidos
CREATE TABLE Pedidos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    cliente_id INT,
    data_pedido DATE NOT NULL,
    total_pedido DECIMAL(10,2),
    FOREIGN KEY (cliente_id) REFERENCES Clientes(id)
);

-- Produtos
CREATE TABLE Produtos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    preco DECIMAL(10,2) NOT NULL
);

-- Itens de Pedido
CREATE TABLE ItensPedido (
    id INT AUTO_INCREMENT PRIMARY KEY,
    pedido_id INT,
    produto_id INT,
    quantidade INT,
    preco_unitario DECIMAL(10,2),
    FOREIGN KEY (pedido_id) REFERENCES Pedidos(id),
    FOREIGN KEY (produto_id) REFERENCES Produtos(id)
);
```

```
-- Descontos (para questão de JOIN com condição temporal)
CREATE TABLE Descontos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    produto_id INT,
    inicio DATE,
    fim DATE,
    porcentagem DECIMAL(5,2),
    FOREIGN KEY (produto_id) REFERENCES Produtos(id)
);

-- ========================
-- INSERÇÃO DE DADOS
-- ========================

-- Clientes
INSERT INTO Clientes (nome, email) VALUES
('Ana Souza', 'ana@email.com'),
('Bruno Lima', 'bruno@email.com'),
('Carla Mendes', 'carla@email.com'),
('Diego Rocha', 'diego@email.com'),
('Elisa Ramos', 'elisa@email.com');

-- Produtos
INSERT INTO Produtos (nome, preco) VALUES
('Notebook', 3500.00),
('Mouse Gamer', 120.00),
('Teclado Mecânico', 450.00),
('Monitor 24"', 900.00),
```

('Headset', 300.00),

('Cadeira Gamer', 1100.00);

-- Pedidos

INSERT INTO Pedidos (cliente_id, data_pedido, total_pedido) VALUES

(1, '2025-01-15', 3620.00),

(2, '2025-02-10', 1470.00),

(2, '2025-03-20', 120.00),

(3, '2025-05-05', 1550.00),

(4, '2025-06-25', 300.00);

-- Itens de Pedido

INSERT INTO ItensPedido (pedido_id, produto_id, quantidade, preco_unitario)
VALUES

-- Pedido 1 (Ana Souza)

(1, 1, 1, 3500.00), -- Notebook

(1, 2, 1, 120.00), -- Mouse Gamer

-- Pedido 2 (Bruno Lima)

(2, 3, 2, 450.00), -- Teclado Mecânico

(2, 4, 1, 900.00), -- Monitor 24"

-- Pedido 3 (Bruno Lima - só mouse)

(3, 2, 1, 120.00), -- Mouse Gamer

-- Pedido 4 (Carla Mendes)

(4, 5, 2, 300.00), -- Headset

-- Pedido 5 (Diego Rocha)

(5, 2, 1, 120.00), -- Mouse Gamer

(5, 5, 1, 300.00); -- Headset

-- Descontos

INSERT INTO Descontos (produto_id, inicio, fim, porcentagem) VALUES

(2, '2025-02-01', '2025-02-28', 10.00), -- Mouse Gamer em promoção fevereiro

(3, '2025-03-01', '2025-04-30', 15.00), -- Teclado em março/abril

(5, '2025-06-01', '2025-06-30', 20.00); -- Headset em junho