Questões Práticas

1. INNER JOIN
Consulta para listar nome dos clientes e valores dos pedidos, apenas clientes com pedidos:

```
SELECT c.nome, p.valor
FROM clientes c
INNER JOIN pedidos p ON c.id = p.cliente_id;
```

---

2. LEFT JOIN
Consulta para listar todos os clientes e seus pedidos (NULL se não houver pedido):

```
SELECT c.nome, p.valor
FROM clientes c
LEFT JOIN pedidos p ON c.id = p.cliente_id;
```

---

3. RIGHT JOIN
Consulta para listar todos os pedidos e nomes dos clientes (NULL se pedido sem cliente):

```
SELECT p.id, p.valor, c.nome
FROM pedidos p
RIGHT JOIN clientes c ON c.id = p.cliente_id;
```

---

4. FULL OUTER JOIN (emulação com UNION)

```
SELECT c.nome, p.valor
FROM clientes c
LEFT JOIN pedidos p ON c.id = p.cliente_id

UNION

SELECT c.nome, p.valor
FROM clientes c
RIGHT JOIN pedidos p ON c.id = p.cliente_id;
```

---

5. Subconsulta simples em SELECT
Nome do cliente com pedido de maior valor:

```
SELECT nome FROM clientes
WHERE id = (
   SELECT cliente_id FROM pedidos
   ORDER BY valor DESC LIMIT 1
);
```

---

6. Subconsulta simples em INSERT
Adicionar pedido para cliente com maior pedido:

```
INSERT INTO pedidos (cliente_id, valor)
VALUES (
   (SELECT cliente_id FROM pedidos ORDER BY valor DESC LIMIT 1),
   100.00
);
```

---

7. Subconsulta correlacionada
Nomes dos clientes e total dos pedidos usando subconsulta correlacionada:

```
SELECT c.nome,
      (SELECT SUM(valor) FROM pedidos p WHERE p.cliente_id = c.id) AS total_vendas
FROM clientes c;
```

---

8. Função COUNT
Número total de pedidos realizados:

```
SELECT COUNT(*) AS total_pedidos FROM pedidos;
```

---

9. Função de agregação com GROUP BY
Total de vendas por cliente:

```
SELECT c.nome, SUM(p.valor) AS total_vendido
FROM clientes c
JOIN pedidos p ON c.id = p.cliente_id
GROUP BY c.nome;
```

---

10. GROUP BY com HAVING
Clientes com vendas superiores a 200:

```
SELECT c.nome, SUM(p.valor) AS total_vendido
```

```
FROM clientes c
JOIN pedidos p ON c.id = p.cliente_id
GROUP BY c.nome
HAVING total_vendido > 200;
```

Questões Teóricas

1. O que é INNER JOIN?
INNER JOIN retorna somente os registros que têm correspondência em ambas as tabelas.
Exemplo: Listar clientes que possuem pedidos registrados.

---

2. Diferença entre LEFT JOIN e RIGHT JOIN

- **LEFT JOIN:** retorna todos os registros da tabela à esquerda, mesmo sem correspondência na direita.

- **RIGHT JOIN:** retorna todos os registros da tabela à direita, mesmo sem correspondência na esquerda.
  Exemplos:
  LEFT JOIN para listar todos os clientes (mesmo sem pedidos).
  RIGHT JOIN para listar todos os pedidos (mesmo sem cliente).

---

3. Como emular FULL OUTER JOIN no MySQL e por que não é nativo?
MySQL não suporta FULL OUTER JOIN diretamente. Ele pode ser emulado com UNION entre LEFT JOIN e RIGHT JOIN. A falta de suporte nativo se deve a limitações históricas e decisões de otimização.

---

4. O que são subconsultas simples?
Subconsultas simples são consultas aninhadas que retornam um valor ou conjunto para uso na consulta principal. São úteis quando a relação é direta e a subconsulta não depende de cada linha da consulta externa.
Exemplo: Buscar cliente com maior pedido sem necessidade de JOIN.

---

5. Diferença entre subconsulta simples e correlacionada

- **Simples:** executada uma vez, independente da consulta externa.

- **Correlacionada:** executada para cada linha da consulta externa, usando valores dessa linha para filtrar.

---

6. Conceito e principais funções de agregação
Funções que resumem dados em grupos, como:

- `COUNT()` — conta registros.
- `SUM()` — soma valores.
- `AVG()` — média.
- `MAX()` — valor máximo.
- `MIN()` — valor mínimo.

---

7. Função do GROUP BY
Agrupa resultados por uma ou mais colunas, para aplicar funções de agregação por grupo. Usado quando se quer analisar dados agrupados, como vendas por cliente.

---

8. O que é HAVING e diferença para WHERE

- **WHERE:** filtra linhas antes do agrupamento.
- **HAVING:** filtra grupos depois do agrupamento, usando condições sobre valores agregados.

---

9. Cenários práticos para usar GROUP BY e agregações

- Total de vendas por categoria de produto.
- Número de alunos matriculados por curso.

---

10. Importância dos JOINs em bancos relacionais

JOINs permitem relacionar dados entre tabelas, evitando redundâncias, mantendo integridade e possibilitando consultas eficientes e organizadas.