1. Diferença entre INSERT INTO e UPDATE

- **INSERT INTO** serve para inserir novos registros em uma tabela. Deve ser usado quando você quer adicionar dados novos.

- **UPDATE** altera dados existentes em uma tabela. Deve ser usado quando você quer modificar valores já cadastrados.

---

2. Função do comando DELETE e diferença para TRUNCATE

- **DELETE** remove linhas específicas de uma tabela, podendo usar cláusula WHERE para escolher quais linhas apagar. Pode ser revertido se estiver em transação.

- **TRUNCATE** remove todos os registros da tabela de forma rápida, sem registrar a exclusão linha a linha. Não pode usar WHERE e geralmente não pode ser revertido.

---

3. Importância da cláusula WHERE em UPDATE e DELETE
A cláusula WHERE limita quais registros serão afetados. Sem ela, o comando atualizaria ou apagaria todos os registros da tabela, o que geralmente não é desejado.

---

4. Utilidade da cláusula ORDER BY
ORDER BY serve para ordenar os resultados de uma consulta, podendo ser por uma ou mais colunas, em ordem crescente (ASC) ou decrescente (DESC).

Exemplo:

SELECT * FROM produtos ORDER BY preco DESC;

---

5. Palavra-chave DEFAULT em inserção de dados
DEFAULT indica que o valor padrão definido na tabela deve ser usado para uma coluna, caso nenhum valor seja informado.

Exemplo:

INSERT INTO produtos (nome, preco) VALUES ('Caneta', DEFAULT);

Se a coluna `preco` tiver valor padrão, ele será usado.

---

6. Finalidade da cláusula DISTINCT
DISTINCT elimina registros duplicados nos resultados de uma consulta, mostrando apenas valores únicos.

Exemplo:

SELECT DISTINCT categoria FROM produtos;

---

7. Funcionamento da cláusula GROUP BY e exemplo prático
GROUP BY agrupa linhas que têm valores iguais em uma ou mais colunas, permitindo usar funções agregadas como SUM, COUNT, AVG.

Exemplo:

SELECT categoria, COUNT(*) FROM produtos GROUP BY categoria;

Retorna a quantidade de produtos por categoria.

---

8. Impacto de não usar índices em consultas com filtros (WHERE)
Sem índices, o banco precisa fazer uma varredura completa (full table scan) para encontrar os dados, o que torna as consultas lentas, principalmente em tabelas grandes.

---

9. Operadores de comparação e lógicos em SQL

- Comparação:
    - `=` (igual a): compara se valores são iguais.
    - `<>` ou `!=` (diferente de): verifica se valores são diferentes.
- Lógicos:
    - `AND`: todas as condições precisam ser verdadeiras.
    - `OR`: pelo menos uma condição precisa ser verdadeira.

Exemplo:

SELECT * FROM produtos WHERE preco > 10 AND categoria = 'Escritório';

---

10. Diferença entre HAVING e WHERE; uso do HAVING com GROUP BY

- **WHERE** filtra linhas antes da agregação e do agrupamento.
- **HAVING** filtra grupos criados pelo GROUP BY, aplicando condições em valores agregados.

Exemplo:

SELECT categoria, COUNT(*) FROM produtos GROUP BY categoria HAVING COUNT(*) > 5;

Retorna apenas categorias com mais de 5 produtos.