ANOTAÇÕES DOS ESTUDOS DE BANCO DE DADOS (MYSQL)

FASE 3: Manipulação de Dados

Sumário

1. Comandos DDL (*Data Definition Language*) 3
   - 1.1. CREATE 3
   - 1.2. ALTER 3
   - 1.3. DROP 4
   - 1.4. TRUNCATE 4
   - 1.5. RENAME 4
2. Comandos DML (*Data Manipulation Language*) 5
   - 2.1. INSERT 5
   - 2.2. SELECT 6
   - 2.3. UPDATE 6
   - 2.4. DELETE 7
   - 2.5. MERGE (ou UPSERT) 7
3. Consultas básicas 7
   - 3.1. Selecionar todas as colunas 8
   - 3.2. Filtros com WHERE 8
   - 3.3. Ordenação de resultados (ORDER BY) 9
   - 3.4. Limitar resultados (LIMIIT) 9
   - 3.5. Uso de Alias (AS) 10
   - 3.6. Eliminando Duplicados (DISTINCT) 10
   - 3.7. Uso de Funções em Consultas básicas 10
     - 3.7.1. Funções de Agregação: 10
     - 3.7.2. Funções Numéricas 12
     - 3.7.3. Funções de Texto (Strings) 13
     - 3.7.4. Funções de Data e Hora 15
   - 3.8. Filtros Avançados 18

4. Exercício prático 01 20
5. Exercício Prático 02 23

1. Comandos DDL (*Data Definition Language*)

Os comandos DDL (Data Definition Language) são usados para criar, alterar e excluir estruturas de um banco de dados, como tabelas, índices e esquemas. Esses comandos definem a estrutura e organização dos dados e **não manipulam diretamente os dados** em si.

Os comandos DDL afetam diretamente a estrutura do banco de dados e geralmente exigem permissões administrativas.

Diferentemente de comandos DML (INSERT, UPDATE, DELETE), **as operações DDL são permanentes e não podem ser revertidas com um comando ROLLBACK.**

1.1. CREATE

O comando CREATE é usado para criar objetos no banco de dados, como tabelas, esquemas, índices, visualizações, entre outros.

```
CREATE TABLE cliente (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    email VARCHAR(150) UNIQUE,
    data_cadastro DATE DEFAULT CURRENT_DATE
);
```

1.2. ALTER

O comando ALTER é usado para modificar a estrutura de um objeto existente no banco de dados.

- **Alterar uma tabela: Adicionar colunas**

```
ALTER TABLE cliente ADD telefone VARCHAR(15);
```

Adiciona uma nova coluna chamada *telefone* à tabela cliente.

- **Alterar uma tabela: Modificar colunas**

```
ALTER TABLE cliente MODIFY telefone VARCHAR(20);
```

Modifica a largura da coluna *telefone* para 20 caracteres.

- **Alterar uma tabela: Renomear colunas**

```
ALTER TABLE cliente CHANGE telefone celular VARCHAR(15);
```

Renomeia a coluna *telefone* para *celular* e mantém o tipo VARCHAR(15).

- **Alterar uma tabela: Excluir colunas**

```
ALTER TABLE cliente DROP COLUMN celular;
```

Remove a coluna celular da tabela.

1.3. DROP

O comando DROP é usado para excluir objetos do banco de dados, como tabelas, índices ou esquemas.

- **Excluir uma tabela:**

```
DROP TABLE cliente;
```

- **Excluir um índice:**

```
DROP INDEX idx_email ON cliente;
```

1.4. TRUNCATE

O comando TRUNCATE é usado para remover todos os dados de uma tabela, mas mantém sua estrutura.

- **Exemplo de Uso:**

```
TRUNCATE TABLE cliente;
```

  - Exclui todos os registros da tabela cliente.
  - É mais rápido que DELETE porque não gera logs detalhados para cada linha excluída.

1.5. RENAME

O comando RENAME permite alterar o nome de tabelas ou outros objetos no banco de dados.

```
RENAME TABLE cliente TO cliente_antigo;
```

Renomeia a tabela *cliente* para *cliente_antigo*.

2. Comandos DML (*Data Manipulation Language*)

Os comandos DML (Data Manipulation Language) **são usados para manipular os dados dentro das tabelas de um banco de dados**. Diferente dos comandos DDL, que lidam com a estrutura do banco, **os comandos DML trabalham diretamente com os registros** (inserir, atualizar, excluir e consultar).

2.1. INSERT

O comando INSERT é usado para adicionar novos registros (linhas) em uma tabela.

```
INSERT INTO cliente (nome, email, telefone)
VALUES ('João Silva', 'joao@gmail.com', '99999-9999');
```

Adiciona um cliente chamado "João Silva" com e-mail e telefone fornecidos.

- **Inserindo múltiplos registros:**

```
INSERT INTO cliente (nome, email, telefone)
VALUES
('Ana Costa', 'ana@gmail.com', '88888-8888'),
('Pedro Santos', 'pedro@gmail.com', '77777-7777');
```

Insere várias linhas de uma só vez;

- **Inserindo apenas em algumas colunas:**
  - Se algumas colunas têm valores padrão ou aceitam NULL, você pode omitir essas colunas:

```
INSERT INTO cliente (nome, email)
VALUES ('Carlos Almeida', 'carlos@gmail.com');
```

O campo telefone será NULL ou usará o valor padrão.

2.2. SELECT

O comando SELECT é usado para consultar dados armazenados no banco.

- **Exemplo simples:**

```sql
SELECT nome, email FROM cliente;
```

- **Usando filtros com WHERE:**

```sql
SELECT nome, email FROM cliente WHERE telefone IS NOT NULL;
```

Retorna apenas clientes com telefone preenchido.

- **Ordenando resultados:**

```sql
SELECT nome, email FROM cliente ORDER BY nome ASC;
```

Retorna os clientes em ordem alfabética (ASC = *ascendente*).

- **Limitando resultados:**

```sql
SELECT * FROM cliente LIMIT 5;
```

Mostra apenas os 5 primeiros registros.

2.3. UPDATE

O comando UPDATE é usado para modificar registros existentes.

- **Exemplo prático:**

```sql
UPDATE cliente
SET email = 'joao_novo@gmail.com', telefone = '99999-1234'
WHERE id = 1;
```

- **Atenção quando usar o UPDATE sem um WHERE:**
  - Sem o WHERE, o comando atualizará **todos os registros** da tabela.

```sql
UPDATE cliente SET telefone = '99999-0000';
```

Isso altera o *telefone* de todos os *clientes* para '*99999-0000*'.

2.4. DELETE

O comando DELETE é usado para excluir registros de uma tabela.

- **Exemplo simples:**

```sql
DELETE FROM cliente WHERE id = 3;
```

Exclui o cliente com id = 3.

- **Excluindo todos os registros:**

```sql
DELETE FROM cliente;
```

- **Atenção quando usar o DELETE sem o WHERE:**
  - Se o WHERE for omitido, todos os registros da tabela serão removidos.

2.5. MERGE (ou UPSERT)

O comando *MERGE* combina ações de *INSERT* e *UPDATE*, mas não está disponível diretamente no MySQL (embora esteja em outros SGBDs, como SQL Server e Oracle). No MySQL, você pode usar *INSERT ... ON DUPLICATE KEY UPDATE* como alternativa.

```sql
INSERT INTO cliente (id, nome, email)
VALUES (1, 'João Silva', 'joao@gmail.com')
ON DUPLICATE KEY UPDATE email = 'joao@gmail.com';
```

Se o *cliente* com id = 1 já existir, apenas o e-mail será atualizado.

3. Consultas básicas

O objetivo das consultas básicas é acessar e manipular os dados em um banco de dados, utilizando o comando **SELECT**. As consultas podem variar de simples buscas em uma tabela até operações mais complexas que envolvem múltiplas tabelas, agregações e ordenações.

3.1. Selecionar todas as colunas

Se você deseja recuperar todas as colunas de uma tabela, pode usar o caractere *.

```
SELECT *
FROM cliente;
```

3.2. Filtros com WHERE

O WHERE é usado para filtrar os dados com base em condições específicas

- **Operadores de Comparação**

Operador	Descrição	Exemplo (`idade = 30`)
=	Igual	Idade igual a 30
!= ou <>	Diferente	Idade diferente de 30
>	Maior que	Idade maior que 30
<	Menor que	Idade menor que 30
>=	Maior ou igual	Idade maior ou igual a 30
<=	Menor ou igual	Idade menor ou igual a 30

  - Exemplo:

```
SELECT nome, idade
FROM cliente
WHERE idade > 25;
```

- **Filtros com operadores lógicos:**
  - Os operadores lógicos permitem combinar condições no WHERE:
    - **AND:** Todas as condições devem ser verdadeiras;
    - **OR:** Pelo menos uma condição deve ser verdadeira;
    - **NOT**: Inverte o resultado de uma condição.

```sql
SELECT nome, cidade
FROM cliente
WHERE cidade = 'São Paulo' AND idade >= 18;
```

Retorna clientes de São Paulo com 18 anos ou mais.

```sql
SELECT nome, cidade
FROM cliente
WHERE cidade = 'São Paulo' OR cidade = 'Rio de Janeiro';
```

Retorna clientes que estão em São Paulo **OU** no Rio de Janeiro.

3.3. Ordenação de resultados (ORDER BY)

O ORDER BY é usado para organizar os resultados em ordem crescente (*ASC*) ou decrescente (*DESC*).

```sql
SELECT nome, idade
FROM cliente
ORDER BY idade ASC;
```

Retorna os clientes organizados pela idade em ordem crescente.

```sql
SELECT nome, cidade
FROM cliente
ORDER BY cidade DESC, nome ASC;
```

Ordena pela cidade em ordem decrescente e, dentro da mesma cidade, ordena os nomes em ordem crescente.

3.4. Limitar resultados (LIMIT)

O LIMIT restringe a quantidade de linhas retornadas por uma consulta.

```sql
SELECT *
FROM cliente
LIMIT 10;
```

Retorna apenas os 10 primeiros registros

3.5. Uso de Alias (AS)

Os aliases são usados para renomear tabelas ou colunas nos resultados da consulta.

```
SELECT nome AS 'Nome do Cliente', cidade AS 'Cidade de Residência'
FROM cliente;
```

Renomeia as colunas nome e cidade no resultado da consulta.

3.6. Eliminando Duplicados (DISTINCT)

O DISTINCT é usado para retornar apenas valores únicos, eliminando duplicados.

```
SELECT DISTINCT cidade
FROM cliente;
```

Retorna todas as cidades cadastradas, sem repetição.

3.7. Uso de Funções em Consultas básicas

SQL fornece funções que podem ser aplicadas diretamente nos dados. Aqui estão algumas comuns:

3.7.1. Funções de Agregação:

As funções de agregação operam em um conjunto de valores e retornam um único valor.

Função	Descrição	Exemplo
`COUNT()`	Conta o número de registros	`SELECT COUNT(*) FROM alunos;`
`SUM()`	Soma os valores de uma coluna	`SELECT SUM(salario) FROM funcionarios;`
`AVG()`	Calcula a média dos valores	`SELECT AVG(nota) FROM notas;`
`MAX()`	Retorna o maior valor	`SELECT MAX(preco) FROM produtos;`
`MIN()`	Retorna o menor valor	`SELECT MIN(preco) FROM produtos;`

- Exemplo prático:

```sql
SELECT categoria, COUNT(*) AS total_produtos, AVG(preco) AS preco_medio
FROM produtos
GROUP BY categoria;
```

Conta quantos produtos existem por categoria e calcula o preço médio.

- **Exemplo com COUNT:**

```sql
SELECT COUNT(*) AS total_vendas FROM vendas;
```

Contar o total de vendas registradas.

- **Exemplo com SUM:**

```sql
SELECT SUM(quantidade) AS total_produtos FROM vendas;
```

Somar o total de produtos vendidos.

- **Exemplo com AVG:**

```sql
SELECT AVG(idade) AS idade_media
FROM cliente
WHERE cidade = 'São Paulo';
```

Calcula a idade média dos clientes que vivem em São Paulo.

```sql
SELECT AVG(preco) AS preco_medio FROM vendas;
```

Calcular o preço médio dos produtos.

- **Exemplo com MAX:**

```sql
SELECT MAX(preco) AS maior_preco FROM vendas;
```

Produto mais caro.

- **Exemplo com MIN:**

```sql
SELECT MIN(preco) AS menor_preco FROM vendas;
```

Produto mais barato.

3.7.2. Funções Numéricas

Usadas para operações matemáticas.

Função	Descrição	Exemplo
`ROUND(x, d)`	Arredonda o número `x` para `d` casas decimais	`SELECT ROUND(12.5678, 2);` → `12.57`
`CEIL(x)`	Arredonda para cima	`SELECT CEIL(12.3);` → `13`
`FLOOR(x)`	Arredonda para baixo	`SELECT FLOOR(12.9);` → `12`
`ABS(x)`	Retorna o valor absoluto	`SELECT ABS(-10);` → `10`
`MOD(a, b)`	Retorna o resto da divisão	`SELECT MOD(10, 3);` → `1`

- Exemplo prático:

```sql
SELECT nome, salario, ROUND(salario * 1.1, 2) AS novo_salario
FROM funcionarios;
```

Calcula um aumento de 10% nos salários e arredonda para 2 casas decimais.

- **Exemplo com ROUND:**

```sql
SELECT ROUND(12.5678, 2) AS valor_arredondado;
```

Arredondar um número para 2 casas decimais.

- **Exemplo com CEIL:**

```sql
SELECT CEIL(12.3) AS valor_arredondado;
```

Arredondar um número para cima.

- **Exemplo com FLOOR:**

```sql
SELECT FLOOR(12.9) AS valor_arredondado;
```

Arredondar um número para baixo.

- **Exemplo com ABS:**

```sql
SELECT ABS(-10) AS absoluto;
```

Valor absoluto de um número.

- **Exemplo com MOD:**

```sql
SELECT MOD(10, 3) AS resto;
```

Obter o resto da divisão de 10 por 3.

3.7.3. Funções de Texto (Strings)

Usadas para manipular textos.

Função	Descrição	Exemplo
`UPPER()`	Converte para maiúsculas	`SELECT UPPER('mysql');` → `'MYSQL'`
`LOWER()`	Converte para minúsculas	`SELECT LOWER('MYSQL');` → `'mysql'`
`LENGTH()`	Retorna o número de caracteres	`SELECT LENGTH('Texto');` → `5`
`LEFT(str, n)`	Retorna `n` caracteres da esquerda	`SELECT LEFT('abcdef', 3);` → `'abc'`
`RIGHT(str, n)`	Retorna `n` caracteres da direita	`SELECT RIGHT('abcdef', 3);` → `'def'`
`SUBSTRING(str, pos, n)`	Retorna `n` caracteres a partir de `pos`	`SELECT SUBSTRING('abcdef', 2, 3);` → `'bcd'`
`CONCAT(str1, str2, ...)`	Concatena strings	`SELECT CONCAT('João', ' ', 'Silva');` → `'João Silva'`

- Exemplo prático:

```sql
SELECT nome, CONCAT(UPPER(sobrenome), ', ', nome) AS nome_formatado
FROM clientes;
```

Formata o nome exibido “SOBRENOME, Nome”.

Para os exemplos abaixo com as funções, utilizaremos uma pequena tabela de exemplo:

id	nome	sobrenome
1	João	Silva
2	Maria	Santos

- **Exemplo com UPPER:**

```
SELECT UPPER(nome) FROM clientes;
```

Converter o nome para maiúsculas.
Resultado: JOÃO, MARIA

- **Exemplo com LOWER:**

```
SELECT LOWER(sobrenome) FROM clientes;
```

Converter o nome para minúsculas.
Resultado: silva, santos

- **Exemplo com LENGTH:**

```
SELECT sobrenome, LENGTH(sobrenome) AS tamanho FROM clientes;
```

Contar o número de caracteres do sobrenome.
Resultado:

sobrenome	tamanho
Silva	5
Santos	6

- **Exemplo com LEFT:**

```sql
SELECT LEFT(nome, 3) FROM clientes;
```

Pegar os 3 primeiros caracteres do nome.
Resultado: Joã, Mar

- **Exemplo com RIGHT:**

```sql
SELECT RIGHT(sobrenome, 2) FROM clientes;
```

Pegar os 2 últimos caracteres do sobrenome.
Resultado: va, os

- **Exemplo com SUBSTRING:**

```sql
SELECT SUBSTRING(nome, 2, 3) FROM clientes;
```

Obter uma parte específica do nome.
Resultado: ão, ari

- **Exemplo com CONCAT:**

```sql
SELECT CONCAT(nome, ' ', sobrenome) AS nome_completo FROM clientes;
```

Concatenar o nome completo.
Resultado: João Silva, Maria Santos

3.7.4. Funções de Data e Hora

Manipulam e formatam datas.

Função	Descrição	Exemplo
`NOW()`	Retorna a data e hora atual	`SELECT NOW();`
`CURDATE()`	Retorna a data atual	`SELECT CURDATE();`
`CURTIME()`	Retorna a hora atual	`SELECT CURTIME();`
`YEAR(data)`	Retorna o ano	`SELECT YEAR('2024-02-12');` → `2024`
`MONTH(data)`	Retorna o mês (1-12)	`SELECT MONTH('2024-02-12');` → `2`
`DAY(data)`	Retorna o dia do mês	`SELECT DAY('2024-02-12');` → `12`
`DATE_FORMAT(data, formato)`	Formata a data	`SELECT DATE_FORMAT(NOW(), '%d/%m/%Y');` → `'12/02/2024'`

- Exemplo prático:

```
SELECT nome, data_nascimento, YEAR(NOW()) - YEAR(data_nascimento) AS idade
FROM clientes;
```

Calcula a idade do cliente com base no ano de nascimento.

Para os exemplos abaixo com as funções, utilizaremos uma pequena tabela de exemplo:

id	nome	data_pedido
1	João	2024-01-20
2	Maria	2024-02-10

- **Exemplo com NOW:**

```
SELECT NOW() AS data_hora_atual;
```

Obter a data e hora atual.
Resultado: 2024-02-12 14:30:45

- **Exemplo com CURDATE:**

```
SELECT CURDATE() AS data_atual;
```

Obter apenas a data atual.
Resultado: 2024-02-12

- **Exemplo com CURTIME:**

```
SELECT CURTIME() AS hora_atual;
```

Obter apenas a hora atual.
Resultado: 14:30:45

- **Exemplo com YEAR:**

```
SELECT cliente, YEAR(data_pedido) AS ano_pedido FROM pedidos;
```

Obter o ano de um pedido.

Resultado:

cliente	ano_pedido
João	2024
Maria	2024

- **Exemplo com MONTH:**

```
SELECT cliente, MONTH(data_pedido) AS mes_pedido FROM pedidos;
```

Obter o mês do pedido.
Resultado:

cliente	mes_pedido
João	1
Maria	2

- **Exemplo com DAY:**

```
SELECT cliente, DAY(data_pedido) AS dia_pedido FROM pedidos;
```

Obter o dia do pedido.
Resultado:

cliente	dia_pedido
João	20
Maria	10

- **Exemplo com DATE_FORMAT:**

```
SELECT DATE_FORMAT(data_pedido, '%d/%m/%Y') AS data_formatada FROM pedidos;
```

Formatar a data para dd/mm/yyyy.
Resultado:

data_formatada
20/01/2024
10/02/2024

3.8. Filtros Avançados

- **BETWEEN:**
  - Filtra valores em um intervalo (comumente usado apenas com números):

```
SELECT nome, idade
FROM cliente
WHERE idade BETWEEN 18 AND 30;
```

Retorna clientes com idades entre 18 e 30 anos.

- **IN:**
  - Verifica se o valor está em uma lista:

```
SELECT nome, cidade
FROM cliente
WHERE cidade IN ('São Paulo', 'Rio de Janeiro', 'Belo Horizonte');
```

Retorna clientes de São Paulo, Rio de Janeiro ou Belo Horizonte.

- **LIKE:**
  - Usado para buscas com padrões:
    - **%**: Substitui zero ou mais caracteres.
    - **_**: Substitui exatamente um caractere.

```sql
SELECT nome
FROM cliente
WHERE nome LIKE 'Jo%';
```

*Retorna nomes que começam com "**Jo**"*

```sql
SELECT nome
FROM cliente
WHERE nome LIKE '_a%';
```

*Retorna nomes onde o segundo caractere é "**a**"*

4. Exercício prático 01

1. Criação de Tabelas

Cenário: Você está criando o banco de dados para um sistema de gerenciamento de uma loja de vendas. O sistema deve armazenar informações sobre os **produtos**, **clientes** e **vendas**.

1. Crie uma tabela chamada produto com as seguintes colunas:
    - **id** (inteiro, chave primária, auto incremento)
    - **nome** (texto, obrigatório)
    - **categoria** (texto, obrigatório)
    - **preco** (decimal, obrigatório)
    - **estoque** (inteiro, obrigatório)

2. Crie uma tabela chamada cliente com as seguintes colunas:
    - **id** (inteiro, chave primária, auto incremento)
    - **nome** (texto, obrigatório)
    - **email** (texto, único)
    - **cidade** (texto, obrigatório)

3. Crie uma tabela chamada venda com as seguintes colunas:
    - **id** (inteiro, chave primária, auto incremento)
    - **id_cliente** (inteiro, chave estrangeira referenciando cliente.id)
    - **id_produto** (inteiro, chave estrangeira referenciando produto.id)
    - **quantidade** (inteiro, obrigatório)
    - **data_venda** (data, obrigatório)

2. Inserção de Dados (DML)

Após criar as tabelas, insira os seguintes registros:

Produtos

Nome	Categoria	Preço	Estoque
Celular Samsung	Eletrônicos	1500.00	10
Notebook Dell	Eletrônicos	3500.00	5
Camisa Polo	Vestuário	80.00	20
Geladeira Brastemp	Eletrodomésticos	2400.00	3

Clientes

Nome	E-mail	Cidade
João Silva	joao@gmail.com	São Paulo
Maria Oliveira	maria@gmail.com	Belo Horizonte
Carlos Santos	carlos@gmail.com	Rio de Janeiro

Vendas

Cliente	Produto	Quantidade	Data Venda
João Silva	Celular Samsung	1	2025-01-20
Maria Oliveira	Camisa Polo	2	2025-01-19
Carlos Santos	Notebook Dell	1	2025-01-18

3. Consultas Básicas

Agora que as tabelas estão criadas e preenchidas, execute as seguintes consultas:

- **Parte 1: Consultando Produtos**
  1. Liste todos os produtos da tabela produto.
  2. Mostre apenas os produtos da categoria "Eletrônicos".
  3. Retorne os produtos cujo preço seja maior que R$ 1000, ordenados por preço em ordem decrescente.
  4. Exiba o nome e o estoque dos produtos que possuem estoque maior que 5 unidades.

- **Parte 2: Consultando Clientes**
  1. Liste o nome e a cidade de todos os clientes.
  2. Exiba os clientes que moram em "São Paulo" ou "Belo Horizonte".
  3. Encontre o cliente com o e-mail maria@gmail.com.

- **Parte 3: Consultando Vendas**
  1. Liste todas as vendas realizadas, exibindo o nome do cliente, o nome do produto e a quantidade.
  2. Filtre as vendas realizadas em 2025-01-19.
  3. Mostre as vendas de produtos da categoria "Eletrônicos".
  4. Calcule a quantidade total de produtos vendidos.
  5. Mostre a venda com a maior quantidade de produtos.

4. Desafios Avançados

1. Use um alias para exibir os resultados das consultas com nomes mais amigáveis. Por exemplo:

   SELECT nome AS "Nome do Cliente", cidade AS "Cidade de Residência"

   FROM cliente;

2. Crie uma consulta que mostre o total arrecadado em cada venda (multiplicando o preço do produto pela quantidade).

3. Mostre os produtos que **não foram vendidos**.

5. Exercício Prático 02

1. Crie uma tabela fictícia chamado produto com as colunas id, nome, categoria e preco.

2. Insira dados fictícios na tabela.

3. Faça as seguintes consultas:
    a. Retorne todos os produtos da tabela.

    b. Liste apenas os produtos da categoria "Eletrônicos".

    c. Mostre os produtos ordenados por preço em ordem decrescente.

    d. Calcule o preço médio de todos os produtos.