Documentação do Banco de Dados — Clientes e Pedidos

Introdução

Este documento apresenta a estrutura de um banco de dados simples composto pelas tabelas **clientes** e **pedidos**, bem como as consultas SQL solicitadas na atividade. Também é apresentada uma explicação breve do raciocínio utilizado na construção de cada consulta.

O banco foi desenvolvido utilizando o **MySQL Workbench**.

Estrutura do Banco de Dados

Tabela: clientes

A tabela clientes armazena as informações dos clientes cadastrados no sistema.

Campos:

- **id**: Identificador único do cliente (chave primária, auto incremento).
- **nome**: Nome do cliente.
- **email**: Email do cliente.

Tabela: pedidos

A tabela pedidos armazena as informações dos pedidos realizados pelos clientes.

Campos:

- **id**: Identificador único do pedido (chave primária, auto incremento).

- **id_cliente**: Identificador do cliente que realizou o pedido (chave estrangeira).
- **data**: Data em que o pedido foi realizado.
- **total**: Valor total do pedido.

Script SQL de Criação das Tabelas

```sql
create table clientes(
    id INT PRIMARY KEY AUTO_INCREMENT,
    nome VARCHAR(100),
    email VARCHAR(100)

);

create table pedidos (
    id INT PRIMARY KEY AUTO_INCREMENT,
    id_cliente INT,
    data DATE,
    total DECIMAL(10,2),
    FOREIGN KEY (id_cliente) REFERENCES clientes(id)
);

```

Consultas SQL

✅ Consulta 1

Enunciado:
Retornar todos os clientes que realizaram pedidos acima de R$ 100, ordenados pelo nome.

```sql
SELECT DISTINCT c.id, c.nome, c.email
FROM clientes c
INNER JOIN pedidos p ON c.id = p.id_cliente
WHERE p.total > 100
ORDER BY c.nome ASC;

```

Nesta consulta foi necessário relacionar as tabelas clientes e pedidos, pois o valor do pedido está armazenado na tabela pedidos, enquanto os dados do cliente estão na tabela clientes.

Foi utilizado um JOIN para fazer essa ligação através do campo `id_cliente`.

O filtro:

```
WHERE p.total > 100
```

garante que apenas pedidos com valor superior a R$ 100 sejam considerados.

O comando DISTINCT foi utilizado para evitar que um cliente apareça repetido caso tenha feito mais de um pedido acima desse valor.

Por fim, o ORDER BY c.nome organiza o resultado em ordem alfabética pelo nome do cliente.

✅ Consulta 2

Enunciado:
Retornar o total de pedidos realizados por cada cliente.

```
37 •    SELECT c.id, c.nome, COUNT(p.id) AS total_pedidos
38      FROM clientes c
39      LEFT JOIN pedidos p ON c.id = p.id_cliente
40      GROUP BY c.id, c.nome
41      ORDER BY c.nome ASC;
42
```

Nesta consulta foi utilizado um LEFT JOIN para garantir que todos os clientes apareçam no resultado, mesmo aqueles que não possuem pedidos cadastrados.

A função de agregação:

```
, COUNT(p.id)
```

É responsável por contar quantos pedidos cada cliente realizou.

O GROUP BY é necessário porque estamos utilizando uma função de agregação. Ele agrupa os resultados por cliente, permitindo calcular o total de pedidos individualmente.

```
AS total_pedidos
```

Foi utilizado para renomear a coluna de resultado, deixando o retorno mais claro e compreensível.