

# Introdução e configuração do ambiente

Henrique Oliveira Marques

Cuiabá, 10 de Novembro de 2025

# Planejamento de Atividades

10/Nov	Introdução e configuração do ambiente
10/Nov	Manipulação e modelagem: INSERT, UPDATE, DELETE, CREATE TABLE, Chaves primárias, estrangeiras e restrições
17/Nov	Linguagem SQL: SELECT e funções de agregação
24/Nov	SQL intermediário: JOINS, Subconsultas e CTEs
01/Dec	Avaliação 1
08/Dec	<b>Imaculada Conceição – Padroeira de Cuiabá (Feriado Municipal)</b>
15/Dec	Transações e controle: Conceitos ACID, COMMIT, ROLLBACK, SAVEPOINT, Locks e concorrência
02/Jan	Procedimentos armazenados e gatilhos: PL/pgSQL, funções, procedimentos, views e Triggers
09/Jan	Administração de banco e usuários: Criação de usuários e papéis, Permissões, Backup e restauração, Estimativa de carga e uso de índices para otimização
16/Jan	<b>Carnaval</b>
23/Jan	Avaliação 2
02/Fev	Bancos NoSQL + T3
09/Fev	Engenharia de Dados +T4
Total	<b>10</b>

# Avaliações

$$A1 + A2 + T1 + T2 + T3 + T4$$

A1: 3.5 pontos

A2: 4 pontos

T1: 0.5 ponto

T2: 0.5 ponto

T3: 0.75 ponto

T4: 0.75 ponto

# Avaliações

$$A1 + A2 + T1 + T2 + T3 + T4$$

A1: 3.5 pontos –

Prova SQL: Exercícios intermediários do LeetCode  
+ exercícios de criação e manipulação de dados

Consulta: Uma folha A4 com anotações escrita à mão (caneta)  
Identificada com nome, apenas um lado da folha

Data: 01/12/2025

# Avaliações

$$A1 + A2 + T1 + T2 + T3 + T4$$

A2: 4 pontos –

Criar e entregar um banco de dados funcional (5-7 tabelas)

Entrega: script .sql completo + documento (max 3 páginas) com diagrama, explicação do modelo, e consultas principais + apresentação

Dupla ou individual

Data: 23/02/2026

# Avaliações

$$A1 + A2 + T1 + T2 + T3 + T4$$

T1: 0.5 ponto –

Lista com 12 exercícios de SQL do LeetCode

Conteúdo:

SELECT, WHERE, ORDER BY, LIMIT, LIKE, BETWEEN, IN

Funções de agregação (COUNT, AVG, MAX, MIN, SUM).

GROUP BY e HAVING

Data: 23/11/2025

# Avaliações

$$A1 + A2 + T1 + T2 + T3 + T4$$

T2: 0.5 ponto –

Lista com 10 exercícios de SQL do LeetCode

Conteúdo:

INNER, LEFT, RIGHT, FULL JOIN

Subconsultas em WHERE e FROM

Common Table Expressions (WITH)

Data: 30/11/2025

# Avaliações

$$A1 + A2 + T1 + T2 + T3 + T4$$

T3: 0.75 ponto

NoSQL

Data provável: 02/03/2025

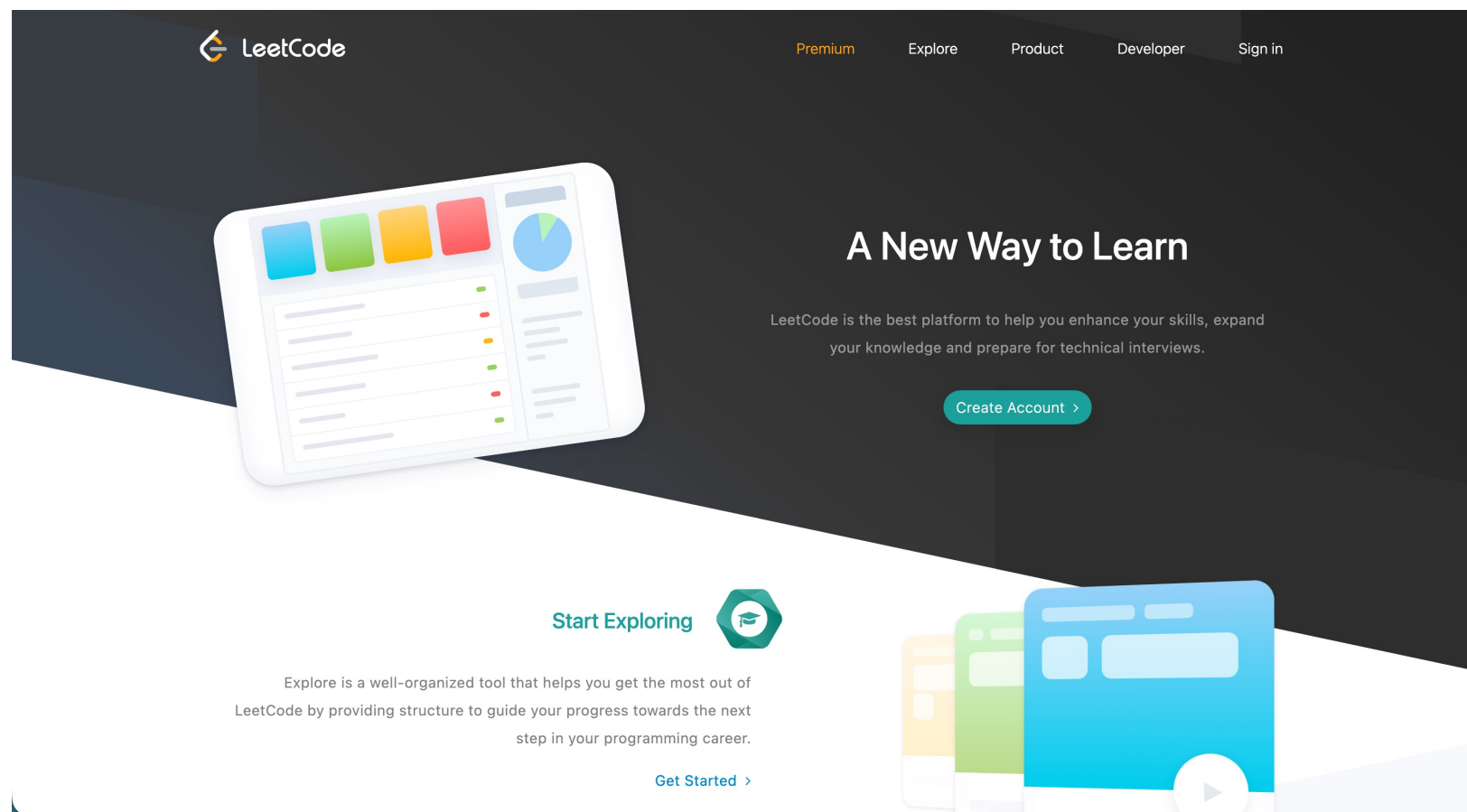
T4: 0.75 ponto

Engenharia de dados

Data provável: 09/03/2025



# Criar conta no LeetCode.com

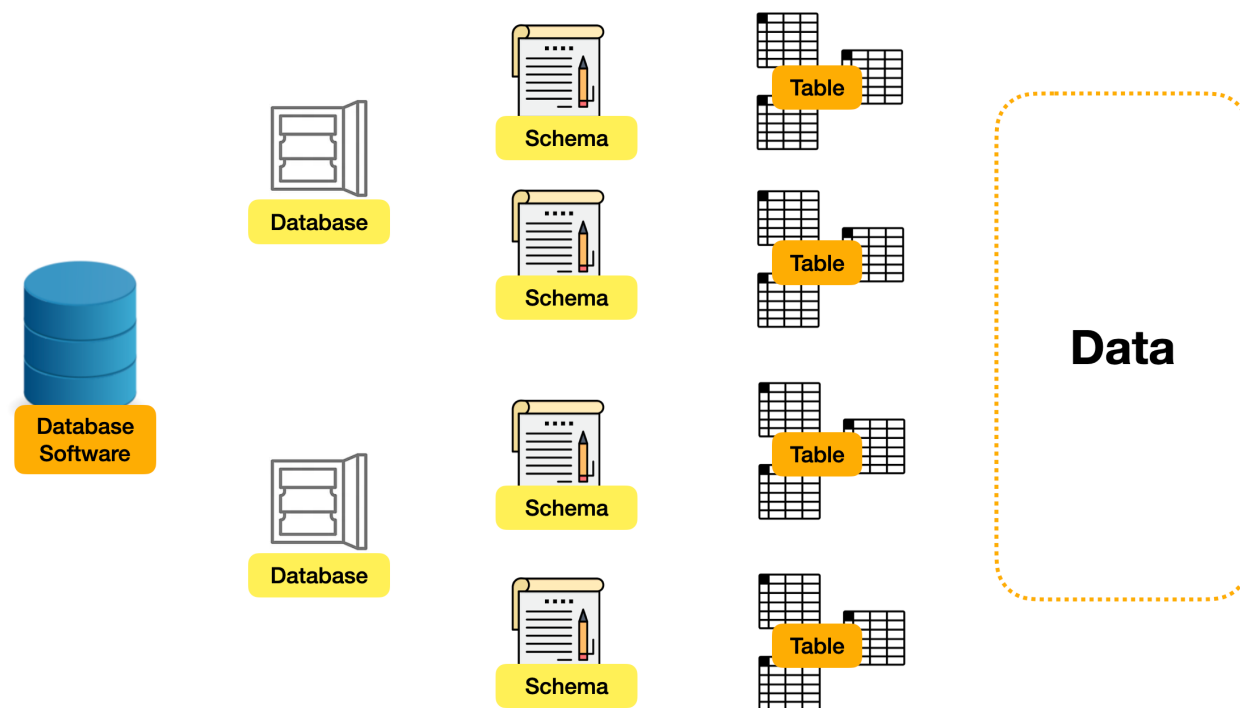


# Materiais da Disciplina



<https://github.com/homarques/LBD/>

# Estrutura do Banco de Dados



Fonte: LeetCode.com

# O que é um SGBD?

Definição:

Um SGBD (Sistema Gerenciador de Banco de Dados) é um software que permite criar, armazenar, organizar, consultar e manipular dados de forma estruturada, garantindo segurança, integridade e desempenho.

Exemplos de SGBDs:

Relacionais:

PostgreSQL

MySQL

Oracle Database

SQL Server

Não relacionais (NoSQL):

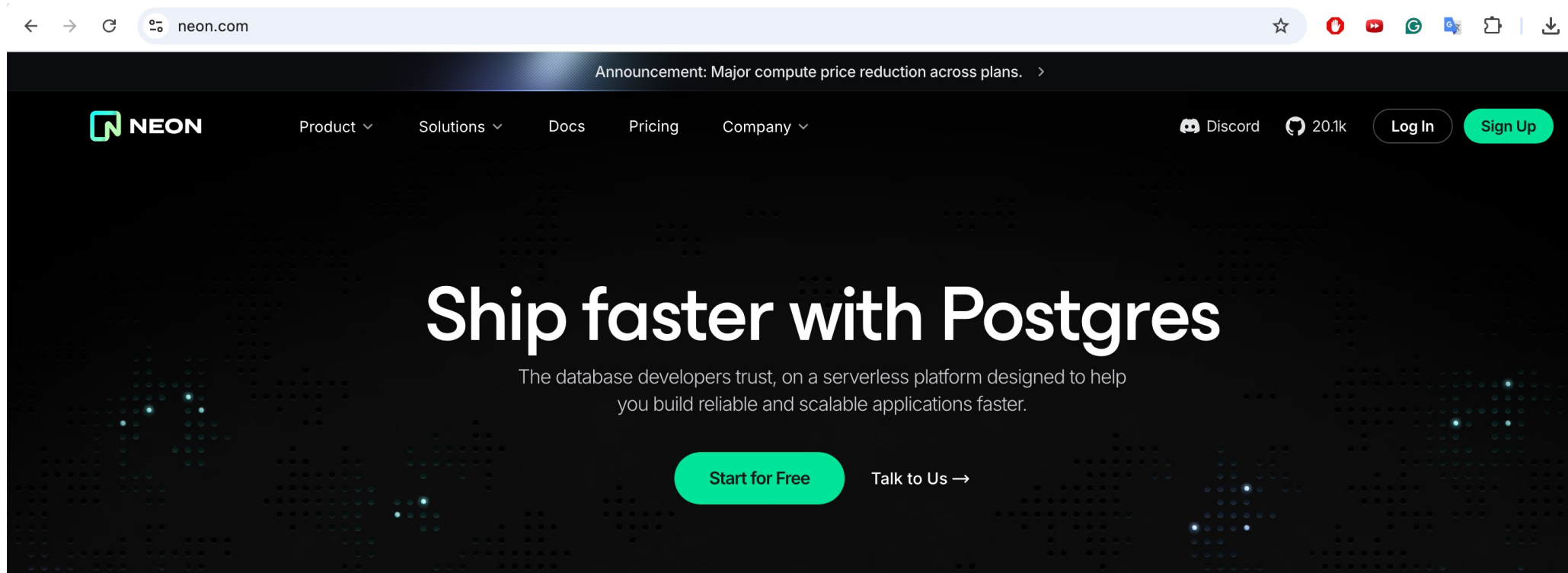
MongoDB

Cassandra

Redis


Neo4j

# Neon.com



← → ↻ neon.com ☆ 🖱️ 📺 🗺️ 📄 📁 ⬇️

Announcement: Major compute price reduction across plans. >


 [Product](#) [Solutions](#) [Docs](#) [Pricing](#) [Company](#) [Discord](#) 20.1k [Log In](#) [Sign Up](#)

# Ship faster with Postgres

The database developers trust, on a serverless platform designed to help you build reliable and scalable applications faster.



[Start for Free](#) [Talk to Us →](#)

# DBeaver.io



## DBeaver Community

Free Open-Source Database Management Tool

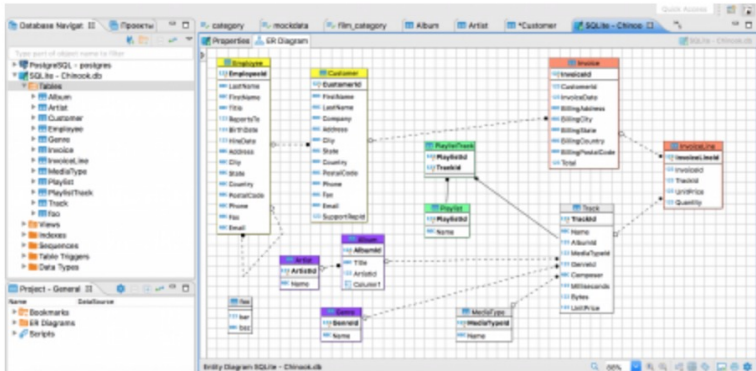
 Star 46,111  Follow @dbeaver\_news


search here ...

[Home](#) [About](#) [Download](#) [Documentation](#) [News](#) [Support](#) [DBeaver PRO](#) [CloudBeaver](#) [DBeaver Merch](#) [Join our team](#)

### Universal Database Tool

DBeaver Community is a free, open-source database management tool for personal projects. Manage and explore SQL databases like MySQL, MariaDB, PostgreSQL, SQLite, Apache Family, and more.





# Configurações de Conexão

Host: ep-royal-truth-ahd9kkox-pooler.c-3.us-east-1.aws.neon.tech

Database: lbd\_db

Username:

Password:

# Alterar Senha

```
ALTER USER <seu_usuario> WITH PASSWORD 'nova_senha';
```



# Banco de Dados

```
CREATE DATABASE nome_do_banco;
```

# Schema

```
CREATE SCHEMA nome_do_schema;
```

# Tabelas

Elemento fundamental do banco de dados: Uma boa modelagem de tabela melhora a manutenção e o desempenho.

## 1. Registro de metadados

- Tipo de dado de cada coluna
- Valores padrão
- Comentários e definições adicionais

# Tabelas

## 2. Gerenciamento de índices

- As tabelas controlam índices
- Índices aceleram buscas e consultas

## 3. Definição de relacionamentos

- Tabelas estabelecem vínculos entre si
- Ex.: um usuário (tabela *users*) pode ter vários pedidos (tabela *orders*)

## 4. Definição do local de armazenamento

- É possível configurar onde os dados serão armazenados fisicamente no disco

# Tipos de dados

Existem 3 tipos principais de dados:

## 1. Numéricos

- Inteiros
  - SMALLINT - 2 bytes ( $-32,768$  a  $32,767$ )
  - INTEGER ou INT - 4 bytes ( $-2,147,483,648$  a  $2,147,483,647$ )
  - BIGINT - 8 bytes ( $-9,22 \times 10^{18}$  a  $9,22 \times 10^{18}$ )
- Decimais
  - NUMERIC(p, s) ou DECIMAL(p, s) - Precisão exata
    - Ex.: DECIMAL(5,2) permite valores como 123.45 (5 dígitos no total, 2 depois da vírgula)
  - REAL (float de 4 bytes) – Não preciso
  - DOUBLE PRECISION (float de 8 bytes) – Não preciso
    - Valores como 2.5 podem ser armazenados como 2.50000000002

# Tipos de dados

Existem 3 tipos principais de dados:

## 2. Data e Hora

- DATE - Armazena apenas a data (ano, mês e dia)
- TIME - Armazena apenas horário
- TIMESTAMP - Armazena data e hora sem fuso horário
- TIMESTAMPTZ - Armazena data e hora com fuso horário
- INTERVAL - Representa durações (ex.: 5 dias, 3 horas)

# Tipos de dados

Existem 3 tipos principais de dados:

## 3. Texto

- CHAR(n): Texto de tamanho fixo;
  - Preenche espaços ao final.
- VARCHAR(n): Texto variável com limite definido.
- VARCHAR (sem n) e TEXT: Texto de tamanho ilimitado;
  - Ambos são eficientes e podem ser usados quando o tamanho máximo não é conhecido.

# Tipos de dados especiais

1. **BYTEA**: Usado para armazenar dados binários (imagens, documentos, vídeos).
  - É o equivalente PostgreSQL ao BLOB de outros sistemas.
2. **BOOLEAN**: Representa valores lógicos (TRUE ou FALSE).
3. **JSON e JSONB**: Tipos usados para armazenar documentos JSON.
  - JSONB é binário e otimizado para busca e indexação, sendo recomendado na maioria dos casos.



# Atributos de Coluna

## 1. NOT NULL

- A coluna não pode receber NULL.
- Obriga o usuário a sempre fornecer um valor.
- Ex.: `id INTEGER NOT NULL;`

## 2. SERIAL

- Cria automaticamente uma sequência associada
- Ex.: `id SERIAL PRIMARY KEY;`

## 3. DEFAULT

- Define um valor padrão para a coluna.
- Usado quando o usuário não envia nenhum valor.
- Ex.: `is_active BOOLEAN DEFAULT TRUE;`

# Criar Tabela

```
CREATE TABLE schema.nome_tabela (  
    id INT NOT NULL,  
    PRIMARY KEY (id)  
);
```

# Criar Coluna

```
ALTER TABLE schema.nome_tabela  
ADD COLUMN name VARCHAR(45);
```

# Atualizar Coluna

```
ALTER TABLE schema.nome_tabela  
RENAME COLUMN name TO user_name;
```

# Atualizar Coluna

```
ALTER TABLE schema.nome_tabela
```

```
ALTER COLUMN user_name TYPE VARCHAR;
```

# Atualizar Coluna

```
ALTER TABLE schema.nome_tabela
```

```
ALTER COLUMN user_name SET DEFAULT 'N/A';
```

# Inserir Dado

```
INSERT INTO schema.nome_tabela (id, user_name)  
VALUES (1, 'Pedro');
```

# Leitura

```
SELECT * FROM schema.nome_tabela;
```



# Inserir Múltiplos Dados

```
INSERT INTO schema.nome_tabela (id, user_name)  
VALUES (2, 'Maria'), (3, 'José');
```

# Leitura (Condição)

```
SELECT * FROM schema.nome_tabela  
WHERE id = 2;
```

# Find Customer Referee

Exercício 2 da Lista 1

# Atualização de Registro

```
UPDATE schema.nome_tabela SET user_name = 'Matheus'  
WHERE id = 1;
```

# Atualização de Registro (múltiplas colunas)

```
UPDATE schema.nome_tabela  
    SET user_name = 'Matheus', age = 17  
    WHERE id = 1;
```

# Apagar Registro

```
DELETE FROM schema.nome_tabela WHERE id = 1;
```

# Limpar Tabela

```
TRUNCATE schema.nome_tabela;
```

# Destruir Tabela

```
DROP TABLE schema.nome_tabela;
```



# Condicional

CASE

WHEN condição THEN resultado

WHEN outra\_condição THEN outro\_resultado

ELSE resultado\_padrão

END

# Swap Salary

<https://leetcode.com/problems/swap-salary/>