

ATIVIDADE DA AULA 2 - Banco de Dados - Relacional (2º Semestre)

ALUNA: Nadla Fernandes Ferreira
RA: 2581392613046

CÓDIGO:

```
CREATE DATABASE queimadas_db;
```

```
CREATE TABLE focos_calor(  
  id SERIAL PRIMARY KEY,  
  estado VARCHAR(50),  
  bioma VARCHAR(50),  
  data_ocorrencia DATE  
);
```

```
INSERT INTO focos_calor(estado, bioma, data_ocorrencia) VALUES  
( 'Amazonas', 'Amazonia', '2025-02-01'),  
( 'Mato Grosso', 'Cerrado', '2025-02-03'),  
( 'Pará', 'Amazônia', '2025-02-05');
```

```
SELECT * FROM focos_calor;
```

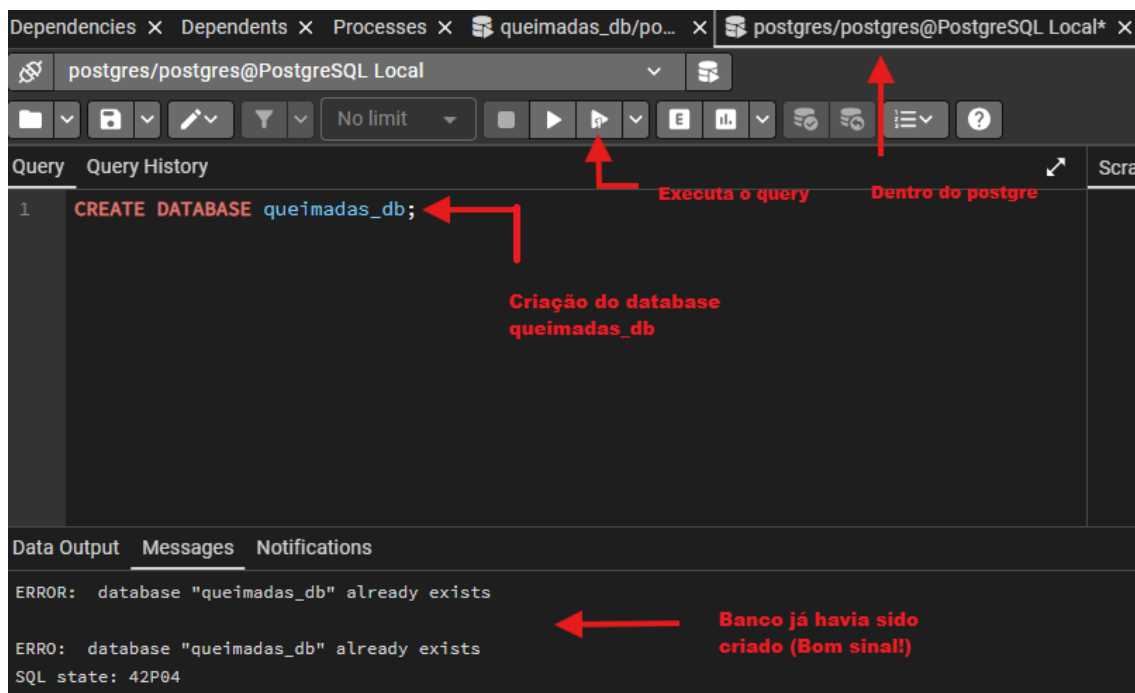
PROCESSO PARA CRIAÇÃO DO BANCO DE DADOS `queimadas_db`:

Esta é a primeira atividade da disciplina de Banco de Dados Relacional (2º semestre). A proposta consiste na execução de um código SQL no pgAdmin, com o objetivo de criar um banco de dados denominado `queimadas_db` e, a partir dele, gerar uma tabela. A seguir, apresentarei o passo a passo do processo, bem como o resultado obtido por mim e o resultado esperado.

Bom, com o PostgreSQL devidamente instalado no computador e seguindo as instruções apresentadas em aula, acessei o pgAdmin e criei um novo servidor.

Em seguida, dentro do database padrão "postgres", abri o Query Tool e executei a primeira parte do código SQL, responsável pela criação do banco de dados chamado "queimadas_db".

Esse procedimento foi realizado separadamente, pois no PostgreSQL o comando `CREATE DATABASE` não pode ser executado dentro de um bloco de transação.



Porém, quando executei o comando `CREATE DATABASE queimadas_db;`, apareceu a mensagem:

`ERROR: database "queimadas_db" already exists`

Isso aconteceu porque o banco já tinha sido criado anteriormente por mim. Mesmo assim, foi importante testar o comando para entender como o PostgreSQL funciona e confirmar que o banco estava criado corretamente.

Seguindo o passo a passo, acessei o database `queimadas_db` e comecei a escrever cada parte do código SQL responsável pela criação da tabela.

Foram utilizados comandos como CREATE TABLE, INSERT INTO e VALUES, além das variáveis/dados: estado, bioma e data_ocorrendia.

Cada parte do código foi executada separadamente, uma por vez, com o objetivo de garantir o funcionamento correto e obter o resultado esperado.

```
1 CREATE TABLE focos_calor(  
2   id SERIAL PRIMARY KEY,  
3   estado VARCHAR(50),  
4   bioma VARCHAR(50),  
5   data_ocorrendia DATE  
6 );  
7  
8 INSERT INTO focos_calor(estado, bioma, data_ocorrendia) VALUES  
9   ('Amazonas', 'Amazonia', '2025-02-01'),  
10  ('Mato Grosso', 'Cerrado', '2025-02-03'),  
11  ('Pará', 'Amazônia', '2025-02-05');  
12  
13 SELECT * FROM focos_calor;
```

Annotations in the image:

- Red arrow pointing to the first execution button: "Cada um é executado um query"
- Red arrow pointing to the first part of the code (CREATE TABLE): "1º Parte executada"
- Red arrow pointing to the second part of the code (INSERT INTO): "2º Parte executada"
- Red arrow pointing to the third part of the code (SELECT * FROM): "3º Parte executada"
- Red arrow pointing to the database name in the title bar: "Essa parte é executada no database queimadas_db"

Vale ressaltar que também foi utilizado o comando id SERIAL PRIMARY KEY.

Ele permite que seja criada uma identificação automática para cada registro inserido na tabela, tornando os dados mais organizados e facilitando sua identificação.

Por fim, a imagem abaixo mostra a tabela resultante, que corresponde ao resultado esperado após a execução do código.

	id [PK] integer	estado character varying (50)	bioma character varying (50)	data_ocorrendia date
1	1	Amazonas	Amazonia	2025-02-01
2	2	Mato Grosso	Cerrado	2025-02-03
3	3	Pará	Amazônia	2025-02-05

Annotations in the image:

- Red arrow pointing to the table data: "Aqui temos a tabela obtida com os dados!"

Esse foi o processo da atividade 1, realizada por mim. Obrigada pela atenção!