

PostgreSQL

Aula 06



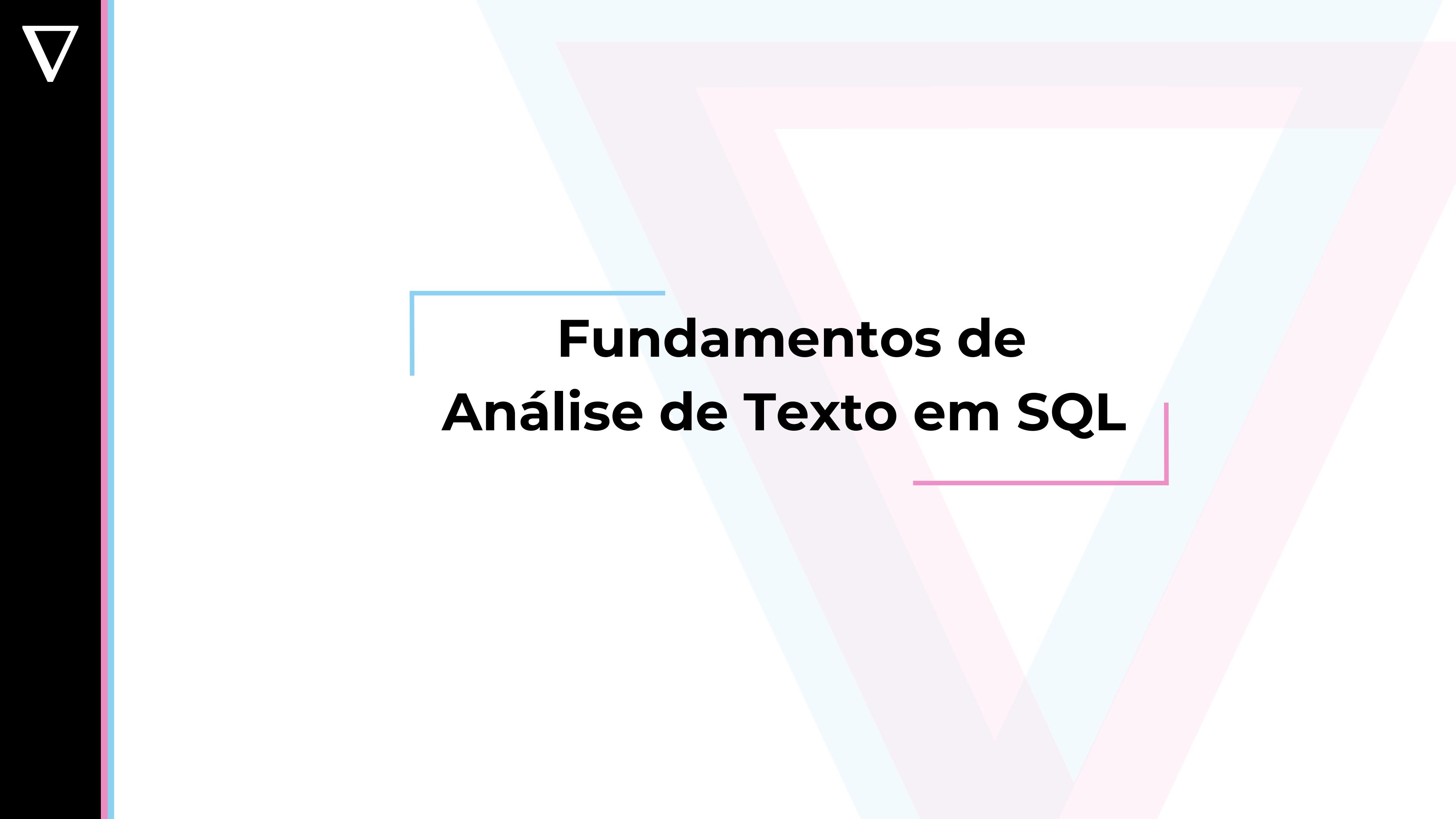
Presença

- Linktree: Presente na bio do nosso instagram
- Presença ficará disponível até 1 hora antes da próxima aula
- É necessário 70% de presença para obter o certificado



Presença





Fundamentos de Análise de Texto em SQL



Fundamentos de Análise de Texto em SQL

- Combina consultas relacionais com exploração de informações textuais
- **Objetivo:** compreender como o SQL pode ser usado para análise de textos
- Identificar quando o SQL é adequado — e quando não é eficiente

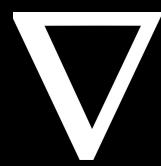
Por que usar SQL para Análise de Texto?

- Simplicidade e integração com bancos de dados relacionais
- Capacidade de lidar com grandes volumes de dados
- Ideal para análises exploratórias rápidas
- Fácil integração com outras ferramentas de análise



O que é Análise de Texto?

- Processo de identificar padrões e informações em textos
- **Inclui:**
- Contagem de palavras
- Busca por palavras-chave
- Extração de informações
- Transformações linguísticas

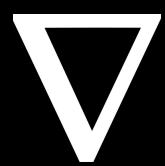


Limitações do SQL na Análise de Texto

- SQL não é ideal para tarefas de Processamento de Linguagem Natural (PLN)
- **Dificuldade em:**
 - Análise semântica profunda
 - Detecção de sentimento
 - Classificação textual avançada



Manipulação e Transformação de Texto

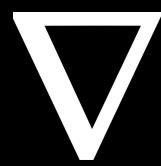


Manipulação e Transformação de Texto

- Operações fundamentais para trabalhar com textos no SQL
- Usadas para limpar, padronizar e extrair informações
- Essenciais antes de análises mais complexas

Funções de Transformação de Texto

- Funções nativas do PostgreSQL para manipular texto:
 - UPPER(text) → passa text para maiúsculas
 - LOWER(text) → passa text para minúsculas
 - INITCAP(text) → passa a inicial de text para maiúscula
 - REPLACE(text, from, to) → substitui partes
 - TRIM(), LTRIM(), RTRIM() → remove espaços
- Usadas em data cleaning e padronização



Exemplo:

- Padronizar a seguinte tabela de nomes:
- Código usando INITCAP (inicial maiúscula)
- e a função TRIM (tira os espaços no início ou fim

SELECT

```
    INITCAP(TRIM(nome)) AS nome_padronizado  
FROM clientes;
```

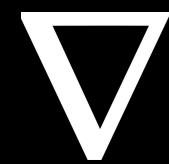
Resultado:

nome
'ana SILVA'
'joão'
'CARLOS souza'

nome_padronizado
Ana Silva
João
Carlos Souza

Características do Texto — Comprimento e Posição

- Entender a estrutura de strings:
 - LENGTH(text) – retorna o número de caracteres de uma string.
 - POSITION(substring IN text) – retorna a posição da primeira ocorrência de uma substring.
 - LEFT(text, n) / RIGHT(text, n) – retornam os primeiros ou últimos n caracteres.
 - SUBSTRING(text FROM start FOR length) – extrai parte específica do texto.



Encontrando Elementos em Blocos de Texto

- Localizar palavras ou expressões específicas
- Funções principais:
 - SUBSTRING(text, pattern) – extrai parte de um texto com base em um padrão (regex simples).
 - STRPOS(text, substring) – retorna a posição da substring.
 - SPLIT_PART(text, delimiter, index) – divide o texto em partes e retorna a parte desejada com base em um delimitador

Exemplo

- Extrair domínio de e-mail → SPLIT_PART(email, '@', 2)

```
SELECT  
    email,  
    SPLIT_PART(email, '@', 2) AS dominio  
FROM usuarios;
```

Resultado:

email	dominio
ana@gmail.com	gmail.com
joao@empresa.com	empresa.com

Boas práticas

- Normalize textos (LOWER() ou UPPER()) antes de comparar
- Use TRIM() para remover espaços invisíveis
- Prefira funções nativas (SPLIT_PART, POSITION) a LIKE
- Para textos longos → usar índices e busca textual (to_tsvector, to_tsquery)



▽

**Matching em
SQL**

Matching em SQL

Exploração e localização de padrões (strings) em um texto

Utilização

- Filtragem de dados de acordo com a presença ou ausência de strings
- Categorização de dados
- Substituição de strings por outras

Matching em SQL

Comandos **LIKE (NOT LIKE)** e **ILIKE (NOT ILIKE)**

- Fornece uma string ou conjunto de strings (atributo) e um string padrão para busca
- LIKE: **case-sensitive**
- ILIKE: **case-insensitive**

Dataset - UFO

Dataset com descrições textuais de fenômenos com ovnis, como cidade, país, descrição e formato

sighting_report	description
Occurred : 6/24/1980 14:00 (Entered as : 06/24/80 14:00)Reported: 4/6/2006 8:45:08 PM 20:45Posted: 5/15/2006Location: Mount W... Missing Time: Two PeopleMy mother and I have a long history of UFO sightings/involvement	
Occurred : 4/6/2006 02:05 (Entered as : 04/06/06 02:05)Reported: 4/6/2006 6:06:21 PM 18:06Posted: 5/15/2006Location: Ottoville, O... Bright lights near Ottoville, OhioHeading westward on SR 189 out of Ft. Jennings, Ohio at th	
Occurred : 9/11/2001 09:00 (Entered as : 9/11/01 10:00)Reported: 4/6/2006 4:04:27 PM 16:04Posted: 5/15/2006Location: Erie, PASh... Planes guided into the Trade Centers by UFOs.I can't believe not many people reported this	

Matching em SQL

Busca pelos dados em que a palavra “man” está presente

```
1  SELECT COUNT(*)
2  FROM ufo
3  WHERE description LIKE/ILIKE '%man%';
```

Com **LIKE**: buscamos apenas “man”

Com **ILIKE**: buscamos “man”, “MAN”, “Man”, etc

Matching em SQL

Simulação do comportamento de **ILIKE** com **LIKE**

- ILIKE não está presente em outros bancos, como MySQL e Oracle

```
1  SELECT COUNT(*)
2  FROM ufo
3  WHERE LOWER(description) LIKE '%man%';
```

LOWER: converte uma string em caixa-baixa

UPPER: converte uma string em caixa-alta

Matching em SQL

Voltando à busca pelos dados em que a palavra “man” está presente

```
1  SELECT COUNT(*)
2  FROM ufo
3  WHERE description LIKE/ILIKE '%man%';
```

Curinga “%”: pode ser substituído por qualquer conjunto de caracteres

- Sem o uso dele, somente textos unitários com a palavra “man” seriam aceitos

Matching em SQL

Busca pelos dados em que as palavras “man” e “men” estão presentes

```
1  SELECT COUNT(*)
2  FROM ufo
3  WHERE description LIKE '%m_n%';
```

Curinga “_”: substituido por um caractere qualquer

Matching em SQL

Além de filtragem, podemos usar LIKE e ILIKE para contagem do número de dados que possuem uma expressão ou palavra

```
1  SELECT
2      COUNT(CASE WHEN description ILIKE '%south%' THEN 1 END) AS South,
3      COUNT(CASE WHEN description ILIKE '%north%' THEN 1 END) AS North,
4      COUNT(CASE WHEN description ILIKE '%east%' THEN 1 END) AS East,
5      COUNT(CASE WHEN description ILIKE '%west%' THEN 1 END) AS West
6  FROM ufo;
```

Matching em SQL

Além de filtragem, podemos usar LIKE e ILIKE para contagem do número de dados que possuem uma expressão ou palavra

```
1  SELECT
2      COUNT(CASE WHEN description ILIKE '%south%' THEN 1 END) AS South,
3      COUNT(CASE WHEN description ILIKE '%north%' THEN 1 END) AS North,
4      COUNT(CASE WHEN description ILIKE '%east%' THEN 1 END) AS East,
5      COUNT(CASE WHEN description ILIKE '%west%' THEN 1 END) AS West
6  FROM ufo;
```

Obs: num texto/registro com 2 ou mais palavras “south”, contamos apenas uma vez. Não há contagem de número de ocorrências das palavras

Matching em SQL

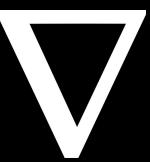
Problema

- Quando buscamos diversas expressões diferentes, podemos ter um conjunto extenso de CASEs
- Código mais propenso a erros e menos legível
- **Solução:** comandos IN e NOT IN

Matching em SQL

Problema

- Quando buscamos diversas expressões diferentes, podemos ter um conjunto extenso de CASEs
- Código mais propenso a erros e menos legível
- **Solução:** comandos IN e NOT IN



Matching em SQL

Sem IN

```
1  SELECT first_word, description
2  FROM
3  (
4      SELECT split_part(description, ' ', 1) as first_word
5      ,description
6      FROM ufo
7  ) a
8  WHERE first_word = 'Red'
9  OR first_word = 'Orange'
10 OR first_word = 'Yellow'
11 OR first_word = 'Green'
12 OR first_word = 'Blue'
13 OR first_word = 'Purple'
14 OR first_word = 'White';
```

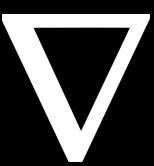
Matching em SQL

Com IN

```
1  SELECT first_word, description
2  FROM
3  (
4      SELECT split_part(description, ' ', 1) as first_word
5      ,description
6      FROM ufo
7  ) a
8 WHERE first_word IN ('Red','Orange','Yellow','Green','Blue','Purple','White')
```



Regex com SQL



Regex com SQL

O que é Regex?

As expressões regulares, ou RegEx (Regular Expressions), são padrões usados para encontrar, validar e manipular textos de maneira flexível. Em SQL, elas nos permitem realizar buscas muito mais precisas do que o operador tradicional LIKE.

Regex com SQL

Regex vs LIKE?

Quando usamos LIKE, conseguimos identificar padrões simples, como: `SELECT nome FROM clientes WHERE nome LIKE 'Jo%';`

Regex com SQL

Regex vs LIKE?

Quando usamos LIKE, conseguimos identificar padrões simples, como: `SELECT nome FROM clientes WHERE nome LIKE 'Jo%';`

Se quisermos:

- Buscar nomes que tenham um número no meio;
- identificar e-mails válidos, CEPs, telefones ou códigos específicos?

O LIKE não é suficiente para isso. É aí que entram as expressões regulares.

Regex com SQL

O PostgreSQL oferece operadores específicos para trabalhar com expressões regulares:

- `~` → corresponde a um padrão (case-sensitive);
- `~*` → corresponde a um padrão (sem diferenciar maiúsculas e minúsculas);
- `!~` → nega o padrão (não corresponde);
- `!~*` → nega o padrão (sem diferenciar maiúsculas e minúsculas).

Regex com SQL

`^` → início da string;

```
1  SELECT nome FROM clientes WHERE nome ~ '^A';
```

Seleciona nomes que começam com A, como “Ana” e “Amanda”.

Regex com SQL

\$ → fim da string;

```
1  SELECT nome FROM clientes WHERE nome ~ 'son$';
```

Seleciona nomes que terminam com “son”, como “Robson” e “Anderson”.

Regex com SQL

[] → conjunto de caracteres permitidos;

```
1   SELECT usuario FROM logins WHERE usuario ~ '[0-9]'
```

Retorna todos os usuários que possuem pelo menos um número no nome, como “juan123” ou “ana2”

Regex com SQL

- . → representa qualquer caractere;
- + → uma ou mais ocorrências;

```
1 ▾ SELECT email  
2 FROM usuarios  
3 WHERE email ~ '^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$';
```

Verifica se os e-mails seguem o formato padrão:
usuario@dominio.com.

Regex com SQL

```
1 ✓ SELECT telefone  
2 FROM contatos  
3 WHERE telefone ~ '^\\(([0-9]{2})[0-9]{5}-[0-9]{4})$';
```

Seleciona telefones como (11)98765-4321.

Regex com SQL

| → operador “ou”;

```
1 ✓ SELECT descricao  
2 FROM produtos  
3 WHERE descricao ~ 'data|info';
```

Retorna textos que tenham as palavras “data” ou “info”.



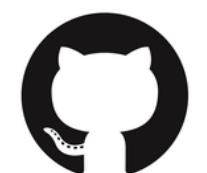
data@icmc.usp.br



[@data.icmc](https://www.instagram.com/@data.icmc)



[/c/DataICMC](https://www.youtube.com/c/DataICMC)



[/icmc-data](https://github.com/icmc-data)



data.icmc.usp.br

|| obrigado!