

Aprofundando em Banco de Dados Relacional e SQL

Aula 13: Conceitos avançados de banco de dados relacional

Prof. MSc. Kizzy Terra

PretaLab
Ciclo Formativo Intermediário
29/09/2025

Conteúdo

- Lidando com Arquivos, Pacotes e Módulos
- Banco de Dados Relacionais e SQL
- **Aprofundando em Banco de Dados Relacionais e SQL**
- Tratamento de Dados Utilizando Pandas e Numpy
- Estatística com Python - Probabilidade, Amostragem e Testes de Hipóteses
- Análise de Dados do Mundo Real - Projeto Final

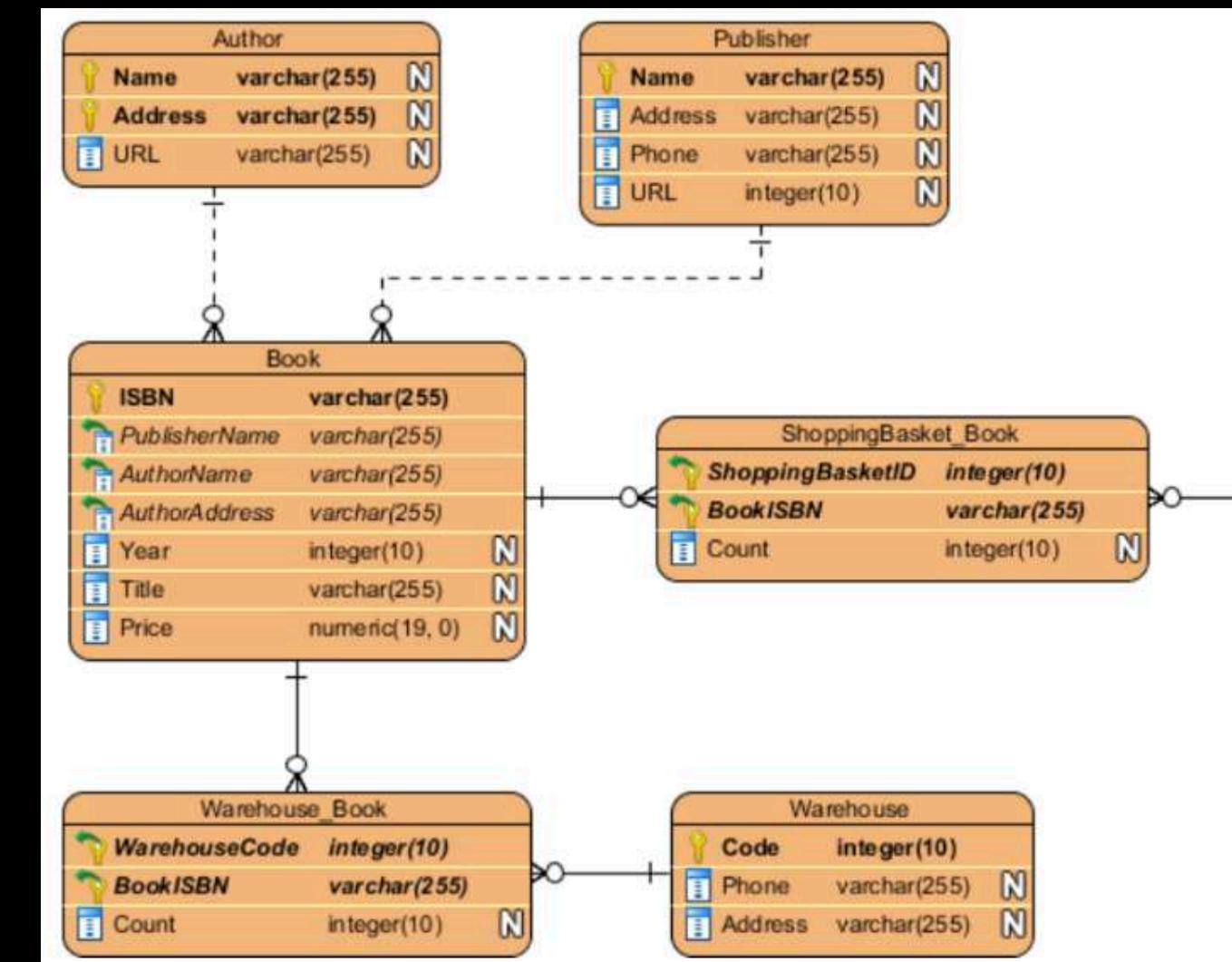
Semanalmente:

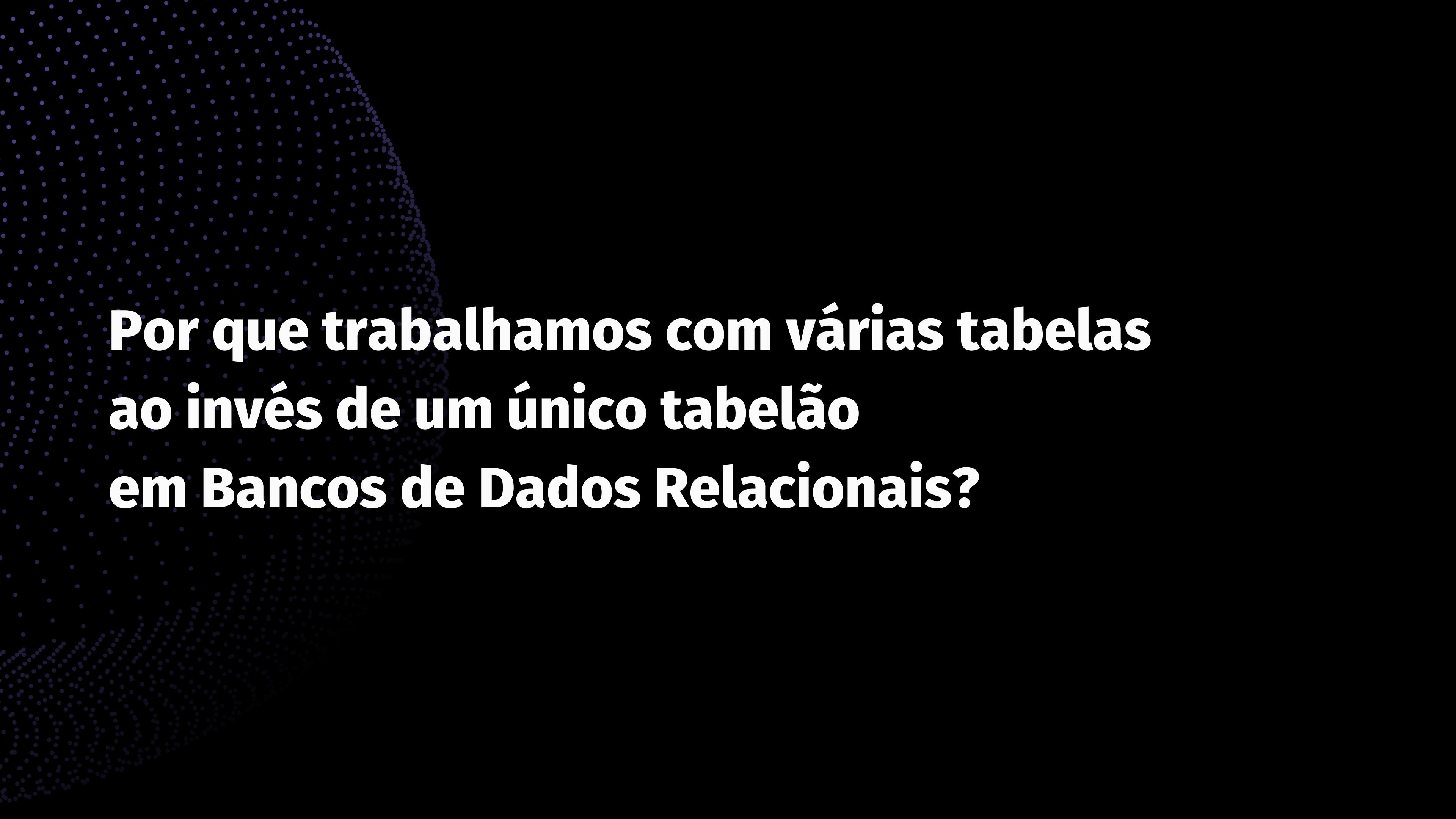
- Aulas teóricas (com prática) nas **segundas e terças**
- Aula focadas em exercícios práticos nas **quintas**
 - Se organize para isso!

Banco de Dados Relacional

Este é o tipo mais comum de banco de dados. Os **dados são organizados em tabelas** com linhas e colunas, e a **relação entre os dados é estabelecida por meio de chaves** primárias e estrangeiras.

Exemplos populares incluem MySQL, PostgreSQL, Oracle e Microsoft SQL Server.





**Por que trabalhamos com várias tabelas
ao invés de um único tabelão
em Bancos de Dados Relacionais?**

Imaginemos uma tabela única chamada **filmes_avaliacoes**:

filme_id	titulo	genero	ano	avaliacao_id	nota	comentario
1	Inception	Ação	2010	101	5	"Excelente, gostei!"
1	Inception	Ação	2010	102	4	"Bom, mas confuso"
2	Toy Story 3	Animação	2010	103	5	"Maravilhoso!"
2	Toy Story 3	Animação	2010	104	3	"Legal, mas longo"

⚠ Problemas:

- **Repetição:** O título, gênero e ano de “Inception” aparecem toda vez que alguém faz uma avaliação.
- **Atualização trabalhosa:** Se precisar corrigir o título (“Inception 2”), tem que alterar em todas as linhas.
- **Maior chance de erro:** Uma linha pode ter “Inception” e outra “Incption” (erro de digitação), quebrando consistência.

Agora dividimos em duas tabelas:

Tabela filmes			
id	titulo	genero	ano
1	Inception	Ação	2010
2	Toy Story 3	Animação	2010
Tabela avaliacoes			
id	filme_id	nota	comentario
101	1	5	"Excelente, gostei!"
102	1	4	"Bom, mas confuso"
103	2	5	"Maravilhoso!"
104	2	3	"Legal, mas longo"

✓ Vantagens:

- O título “Inception” aparece uma única vez.
- Se mudar o título, só precisamos alterar uma linha em filmes.
- As avaliações ficam independentes, mas ligadas corretamente.
-

Normalização dos dados



A normalização é focada na prevenção de problemas com repetição e atualização de dados, assim como o cuidado com a integridade dos dados.

Este conceito foi apresentado originalmente em um artigo científico publicado pela IBM de autoria do matemático Edgar F. Codd, intitulado "Um modelo de dados relacionais para grandes bancos de dados compartilhados" (1970).

Normalização dos dados

PESSOA

CPF	Nome	Sexo	Localização	Telefone
111	Ana	F	Rio de Janeiro, RJ	999-444, 999-000
222	Bruno	M	São Paulo, SP	888-888, 444-333
333	Carla	F	Belo Horizonte, MG	555-777
444	Diego	M	Vitória, ES	999-999

O atributo multivalorado da tabela original se transforma em uma coluna da tabela nova.

→

CPF	Telefone
111	999-444
111	999-000
222	888-888
222	444-333
333	555-777
444	999-999

PESSOA

CPF	Nome	Sexo	Localização
111	Ana	F	Rio de Janeiro, RJ
222	Bruno	M	São Paulo, SP
333	Carla	F	Belo Horizonte, MG
444	Diego	M	Vitória, ES

→

PESSOA

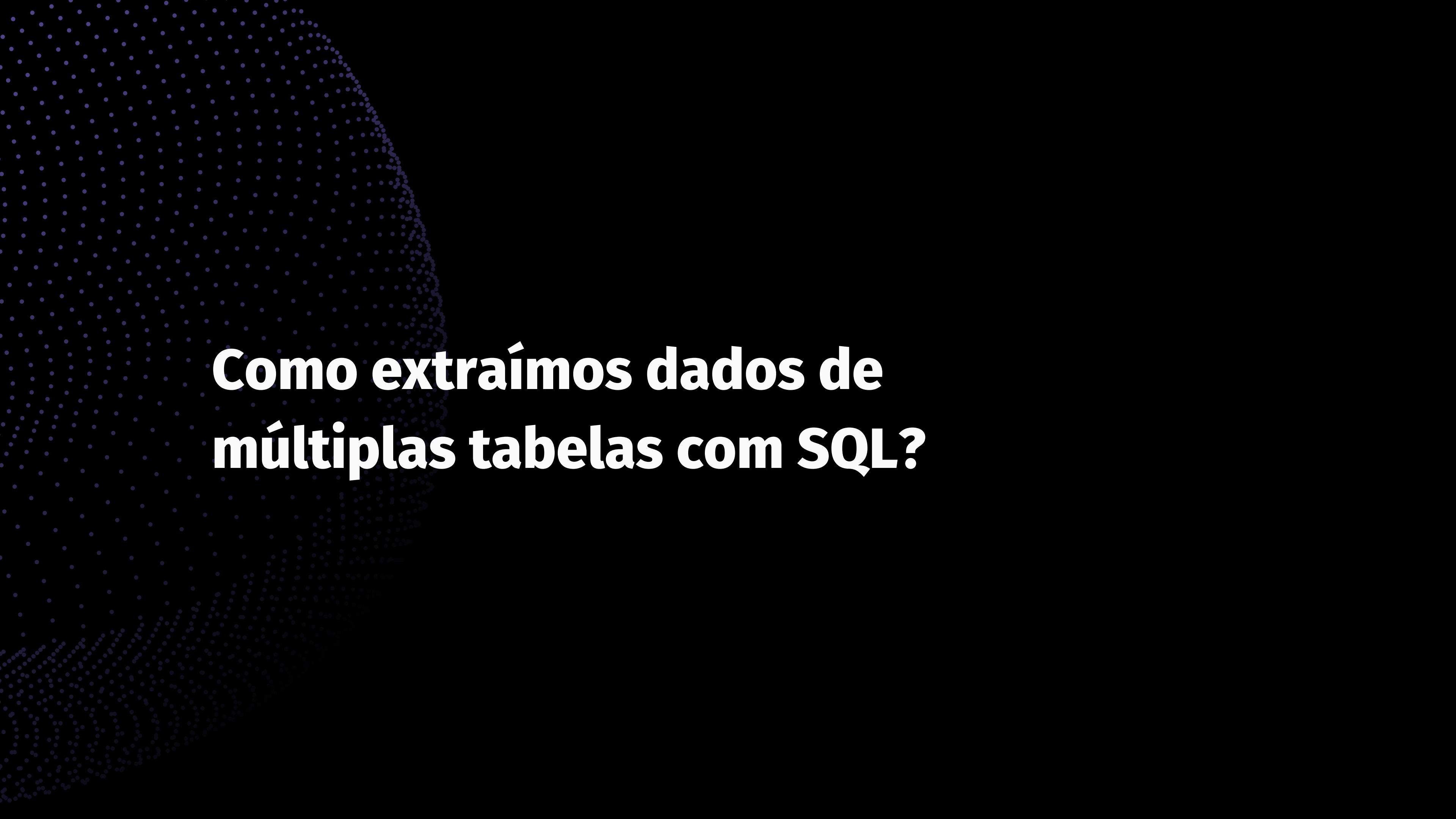
CPF	Nome	Sexo	Cidade	Estado
111	Ana	F	Rio de Janeiro	RJ
222	Bruno	M	São Paulo	SP
333	Carla	F	Belo Horizonte	MG
444	Diego	M	Vitória	ES

A coluna Localização foi dividida em duas, contendo a informação de Cidade e Estado separadamente.

Resumindo

Usamos várias tabelas em Bancos de Dados Relacionais porque:

- Evitamos duplicação e inconsistências
- Modelamos melhor os relacionamentos do mundo real
- Tornamos os dados mais fáceis de manter e atualizar
- Ganhamos eficiência nas consultas
- Melhoramos segurança e organização
- Facilitamos a evolução do sistema



Como extraímos dados de múltiplas tabelas com SQL?

Chave Primária

É um campo (ou conjunto de campos) usado para identificar de forma única cada registro em uma tabela do banco de dados.

Ela funciona como um “CPF” da linha: não pode se repetir e não pode ser nulo, garantindo que cada linha seja distinta das demais.

Chave Estrangeira

É um campo (ou conjunto de campos) em uma tabela que faz referência à chave primária de outra tabela.

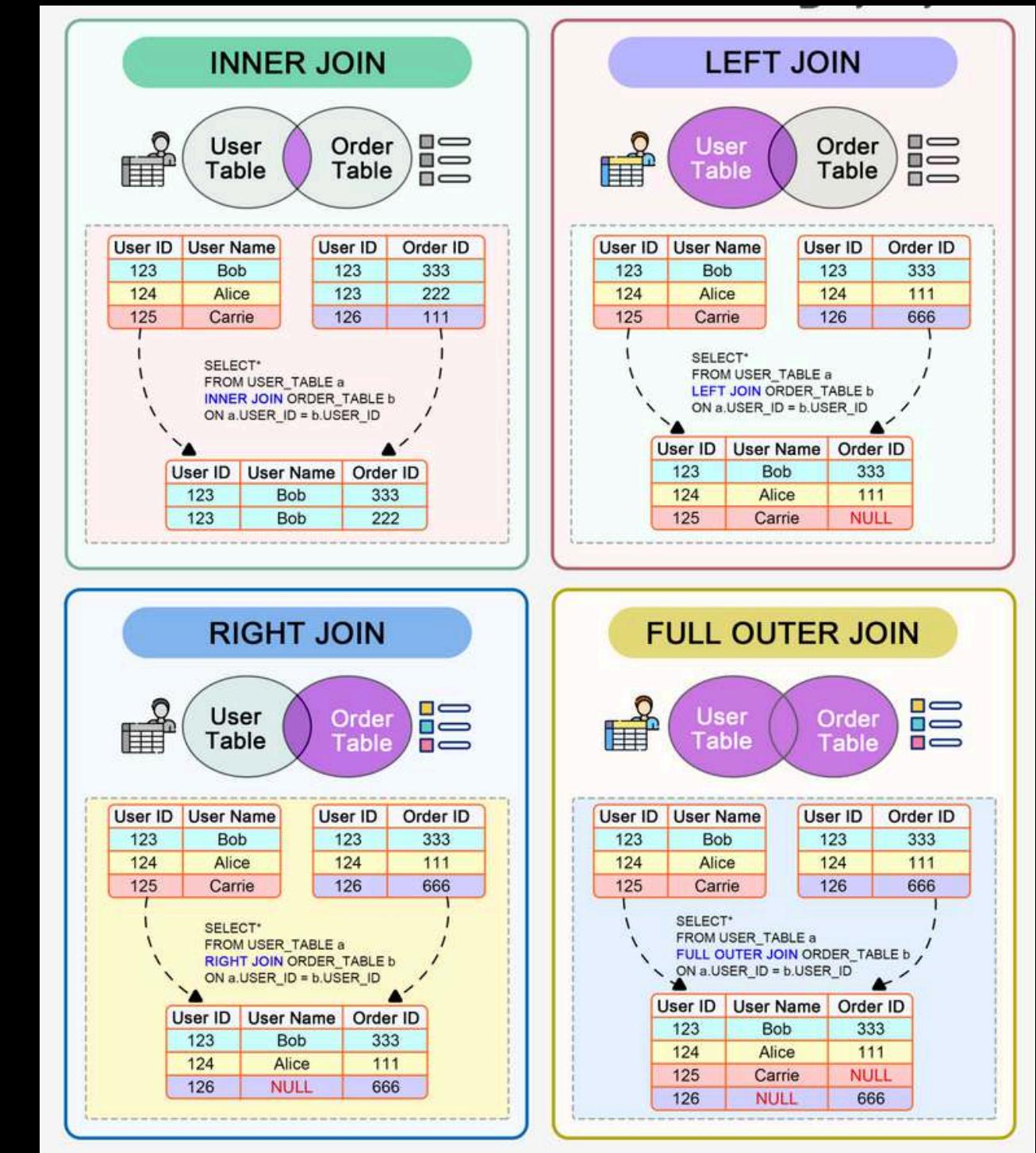
Sua função é criar relacionamentos entre tabelas e garantir a integridade referencial dos dados.

SQL Joins

Um JOIN é um meio de combinar colunas de uma ou mais tabelas, usando valores comuns a cada uma delas (geralmente chave estrangeira).

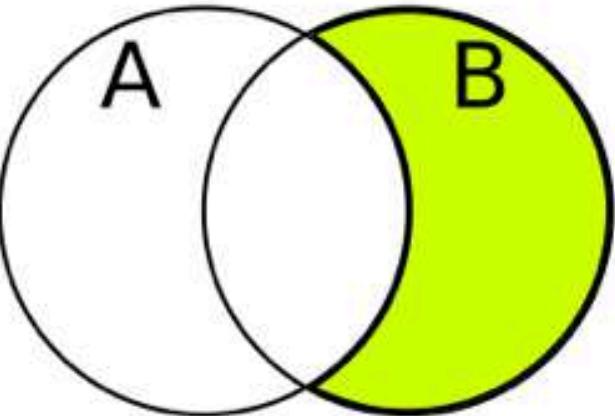
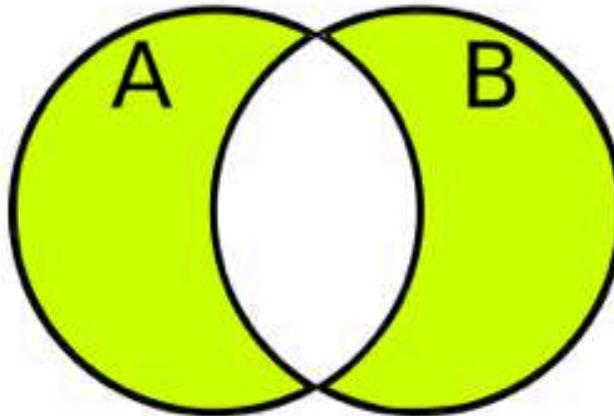
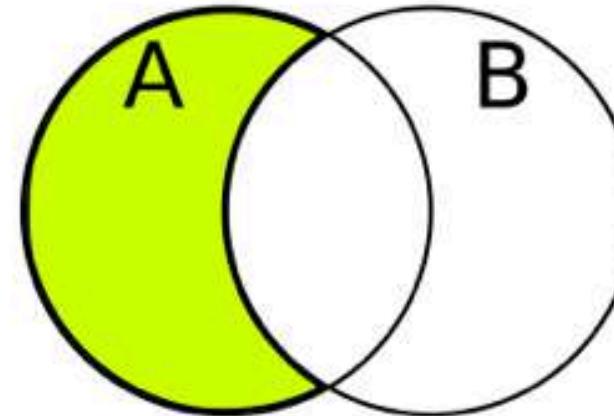
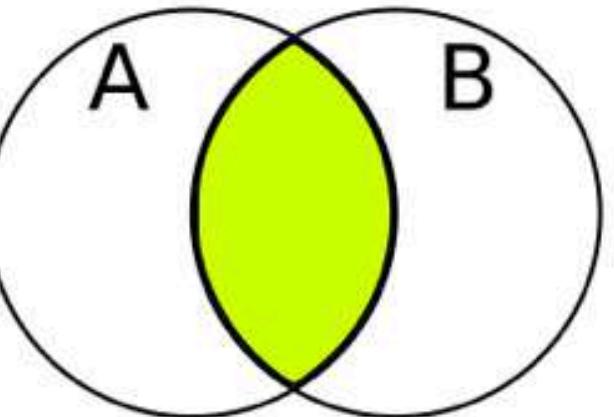
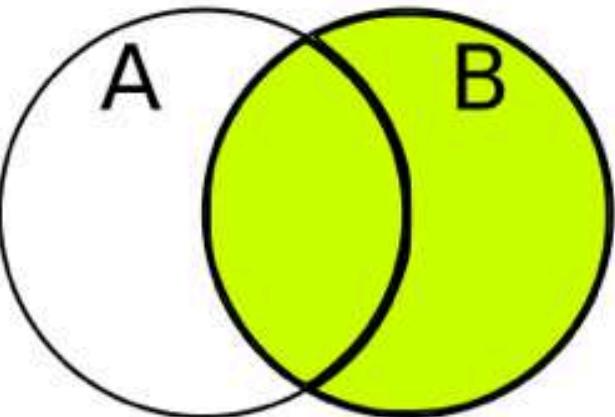
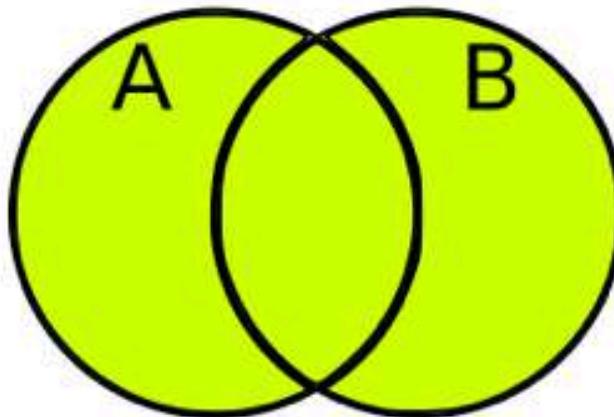
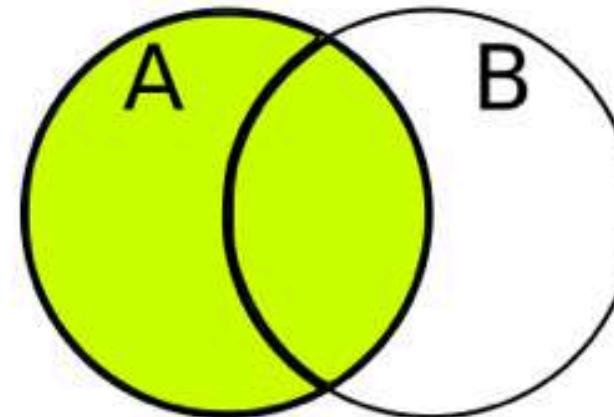
Existem quatro tipos principais de JOIN: INNER JOIN, LEFT JOIN, RIGHT JOIN e FULL JOIN.

Como um caso especial, uma tabela (tabela base, visão ou tabela juntada) pode se juntar a si mesma em uma auto-união (self-join).



SQL JOINS

Arranged in a Karnaugh Map by Jason Charney (jrcharney@gmail.com)

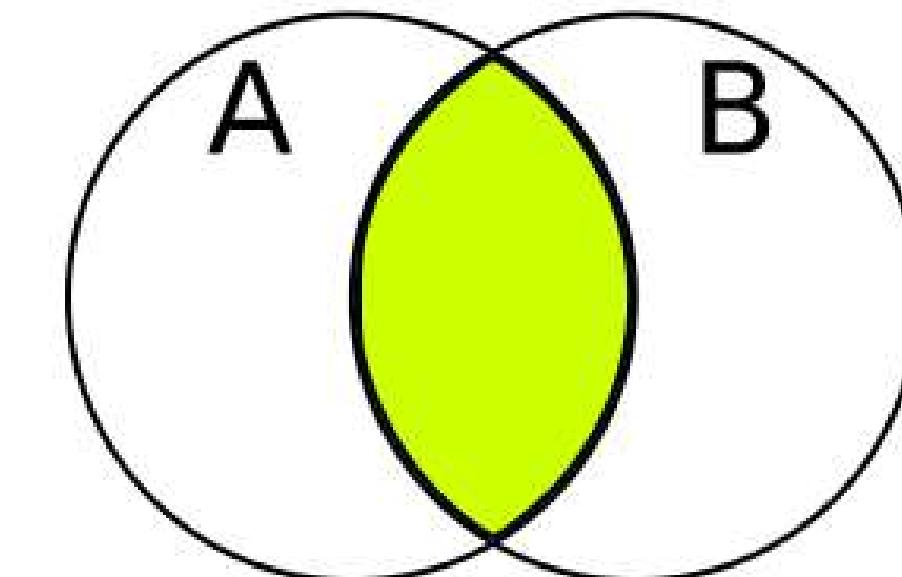
[Exclusive] Right Join ($\neg A$)	[Exclusive] Full Join ($A \oplus B$)	[Exclusive] Left Join ($\neg B$)
		
<pre>SELECT *\nFROM A\nRIGHT JOIN B ON A.key = B.key\nWHERE B.key IS NULL</pre>	<pre>SELECT *\nFROM A\nFULL JOIN B ON A.key = B.key\nWHERE B.key IS NULL\nOR A.key IS NULL</pre>	<pre>SELECT *\nFROM A\nLEFT JOIN B ON A.key = B.key\nWHERE B.key IS NULL</pre>
Inner Join ($A \wedge B$)	[Inclusive] Right Join (B)	[Inclusive] Full Join ($A \vee B$)
		
[Inclusive] Left Join (A)		
<pre>SELECT *\nFROM A\nINNER JOIN B ON A.key = B.key</pre>	<pre>SELECT *\nFROM A\nRIGHT JOIN B ON A.key = B.key</pre>	<pre>SELECT *\nFROM A\nFULL JOIN B ON A.key = B.key</pre>
		
		<pre>SELECT *\nFROM A\nLEFT JOIN B ON A.key = B.key</pre>

INNER JOIN

```
SELECT <colunas>  
FROM Tabela1  
INNER JOIN Tabela2  
ON Tabela1.chave_estrangeira =  
Tabela2.chave_primaria;
```

```
SELECT f.titulo, f.genero, a.notas, a.comentario  
FROM filmes f  
INNER JOIN avaliacoes a  
ON f.id = a.filme_id;
```

Inner Join (A \wedge B)



```
SELECT *  
FROM A  
INNER JOIN B ON A.key = B.key
```

O INNER JOIN mostra somente os registros que têm correspondência nas duas tabelas.

Prática 1 - INNER JOIN

👉 Liste os filmes e suas avaliações. Mostre apenas os filmes que possuem pelo menos uma avaliação.

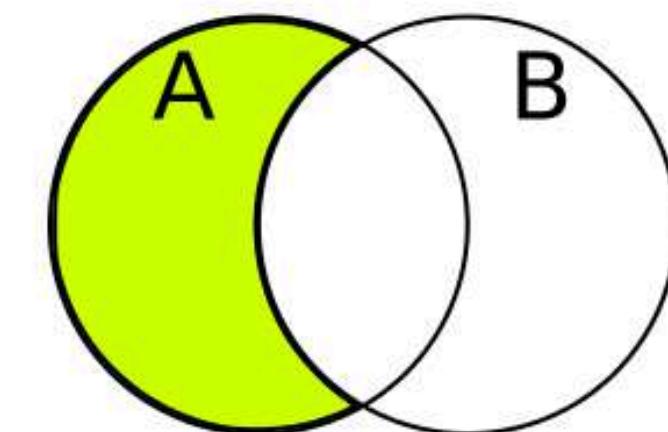
<https://www.db-fiddle.com/f/iCENLmdAvCtdTzeJmaFD2T/0>

LEFT JOIN

```
SELECT <colunas>  
FROM Tabela1  
LEFT JOIN Tabela2  
ON Tabela1.chave_estrangeira =  
Tabela2.chave_primaria;
```

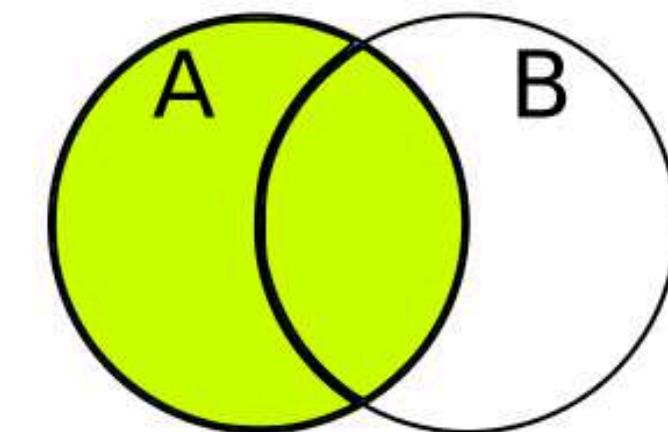
```
SELECT f.titulo, a.notas, a.comentario  
FROM filmes f  
LEFT JOIN avaliacoes a  
ON f.id = a.filme_id;
```

[Exclusive] Left Join ($\neg B$)



```
SELECT *  
FROM A  
LEFT JOIN B ON A.key = B.key  
WHERE B.key IS NULL
```

[Inclusive] Left Join (A)



```
SELECT *  
FROM A  
LEFT JOIN B ON A.key = B.key
```

O LEFT JOIN mostra todos os registros da tabela da esquerda, e os dados da outra tabela somente se houver correspondência. Quando não existe correspondência, os campos da outra tabela ficam vazios (NULL).

Prática 2 - LEFT JOIN

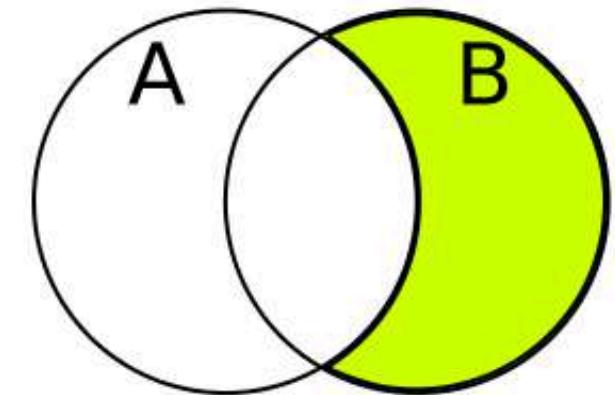
- 👉 Liste todos os filmes e avaliações, mas mostre aqueles que não possuem avaliação.

RIGHT JOIN

```
SELECT <colunas>
FROM Tabela1
RIGHT JOIN Tabela2
ON Tabela1.chave_estrangeira =
Tabela2.chave_primaria;
```

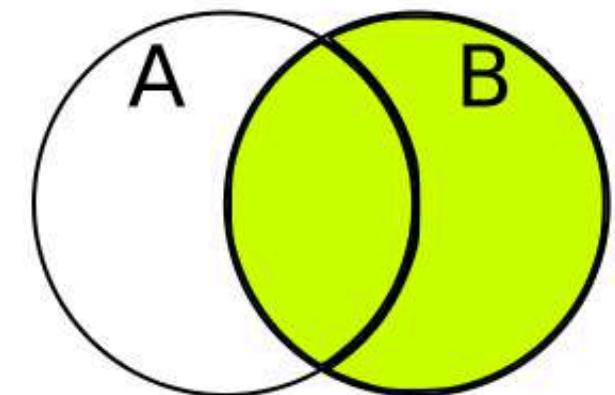
```
SELECT f.titulo, a.notas, a.comentario
FROM filmes f
RIGHT JOIN avaliacoes a
ON f.id = a.filme_id;
```

[Exclusive] Right Join ($\neg A$)



```
SELECT *
FROM A
RIGHT JOIN B ON A.key = B.key
WHERE B.key IS NULL
```

[Inclusive] Right Join (B)



```
SELECT *
FROM A
RIGHT JOIN B ON A.key = B.key
```

Esse JOIN é o contrário do LEFT JOIN, mostra todos os registros da tabela da direita, e os dados da tabela da esquerda apenas se houver correspondência.

Nem todos os bancos de dados suportam RIGHT JOIN (ex: SQLite não suporta, mas PostgreSQL e MySQL sim).

Prática 3 - RIGHT JOIN

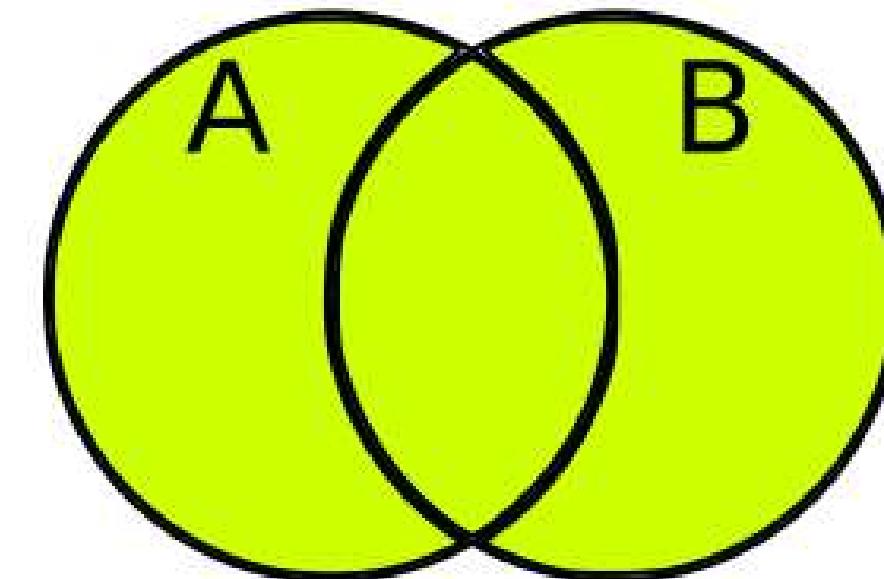
- 👉 Liste todas as avaliações, mesmo aquelas que fazem referência a filmes inexistentes.

FULL OUTER JOIN

```
SELECT <colunas>
FROM Tabela1
FULL OUTER JOIN Tabela2
ON Tabela1.chave_estrangeira =
Tabela2.chave_primaria;
```

```
SELECT f.titulo, a.notas, a.comentario
FROM filmes f
FULL OUTER JOIN avaliacoes a
ON f.id = a.filme_id;
```

[Inclusive] Full Join (A ∨ B)



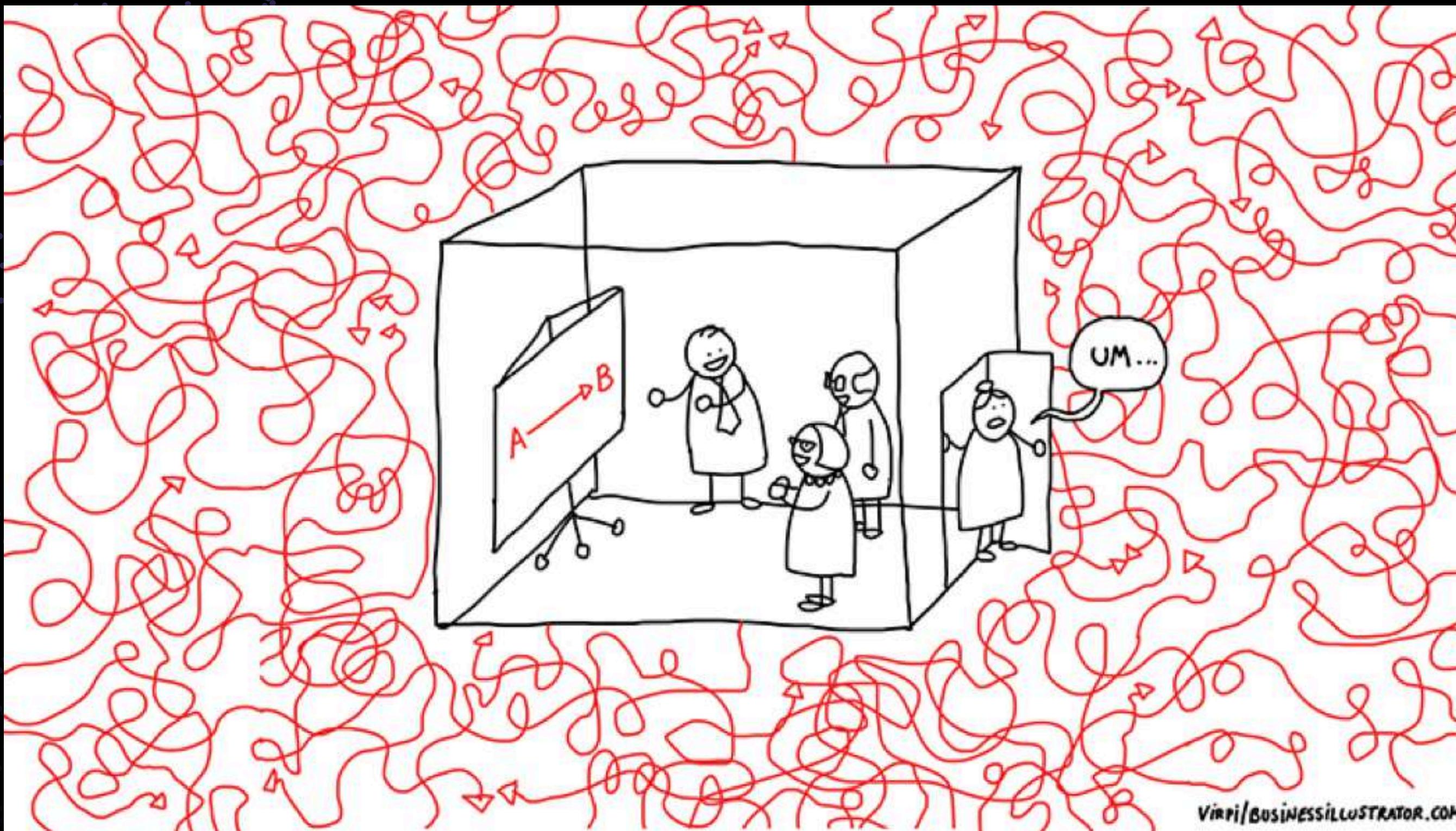
```
SELECT *
FROM A
FULL JOIN B ON A.key = B.key
```

O FULL JOIN mostra todos os registros das duas (ou mais) tabelas, combinando quando for possível, e preenchendo com NULL quando não houver correspondência.

Prática 4 - OUTER JOIN

- 👉 Liste todos os filmes e todas as avaliações, mesmo que não haja correspondência entre eles.

Como fazemos tudo isso com mais de duas tabelas?



Prática 5 - JOIN com 3 tabelas

- 👉 Liste os usuários que fizeram avaliações, junto com o filme avaliado e a nota dada.

Prática 6

Exercício 1 - INNER JOIN básico

Liste os pedidos mostrando o nome do cliente e o título do livro comprado.

Exercício 2 - LEFT JOIN

Liste todos os clientes e seus pedidos, incluindo os clientes que ainda não compraram nada.

Exercício 3 - RIGHT JOIN

Liste todos os pedidos e os clientes correspondentes, incluindo pedidos que possam estar sem cliente cadastrado.

Exercício 4 - FULL OUTER JOIN

Liste todos os clientes e todos os pedidos, mesmo que não haja correspondência entre eles.

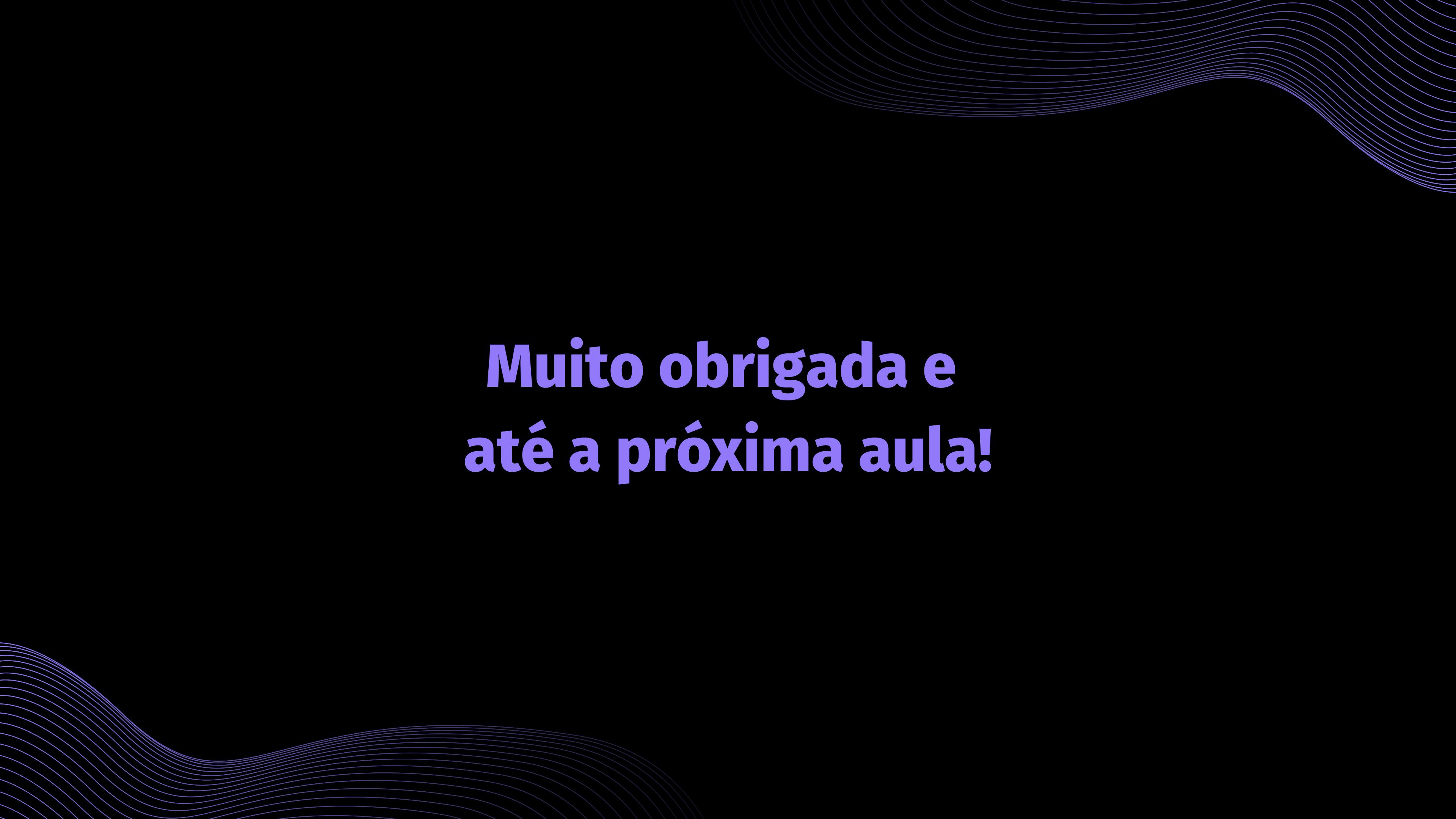
Exercício 5 - JOIN + agregação

Liste cada cliente junto com o total de pedidos realizados.

Solução dos exercícios: <https://www.db-fiddle.com/f/9UNpWtRSupe5Y9T2LpyHQa/13>



Dúvidas?



**Muito obrigada e
até a próxima aula!**