JOIN e Agregações

Aula 14 — Cruzando tabelas e resumindo dados

Aula 14 · 27/10/2026 · Indicadores 3, 5

UniSenac Centro Universitário RS
Senac
Fecomércio
IFEP
Sesc
Sindicatos Empresariais

O mapa de hoje

- **Bloco 1** — `INNER JOIN`: cruzar duas tabelas
- **Bloco 2** — `LEFT JOIN`: e quem não tem par?
- **Bloco 3** — agregações: `COUNT`, `SUM`, `AVG` e `GROUP BY`
- **Bloco 4** — mão na massa: perguntas que só o JOIN responde

> Esta é, com folga, a aula mais útil da UC para o mercado de trabalho. Quem domina `JOIN` e `GROUP BY` responde 80% do que se pede a um programador em relatório.

Retomando a Aula 13

- Qual a diferença entre o que o `SELECT` escolhe e o que o `WHERE` escolhe?
- Por que `= NULL` não funciona?
- Qual é o primeiro passo antes de escrever qualquer consulta?

Bloco 1

INNER JOIN — cruzando duas tabelas

O problema de ontem

`SELECT * FROM emprestimo` devolve:

id_emprestimo	data_retirada	id_cliente
7	20/10/2026	1
8	21/10/2026	2

O dono não quer ver `1`. Ele quer ver **"Ana Souza"**.

> O nome do cliente está na **outra** tabela. Precisamos das duas ao mesmo tempo.

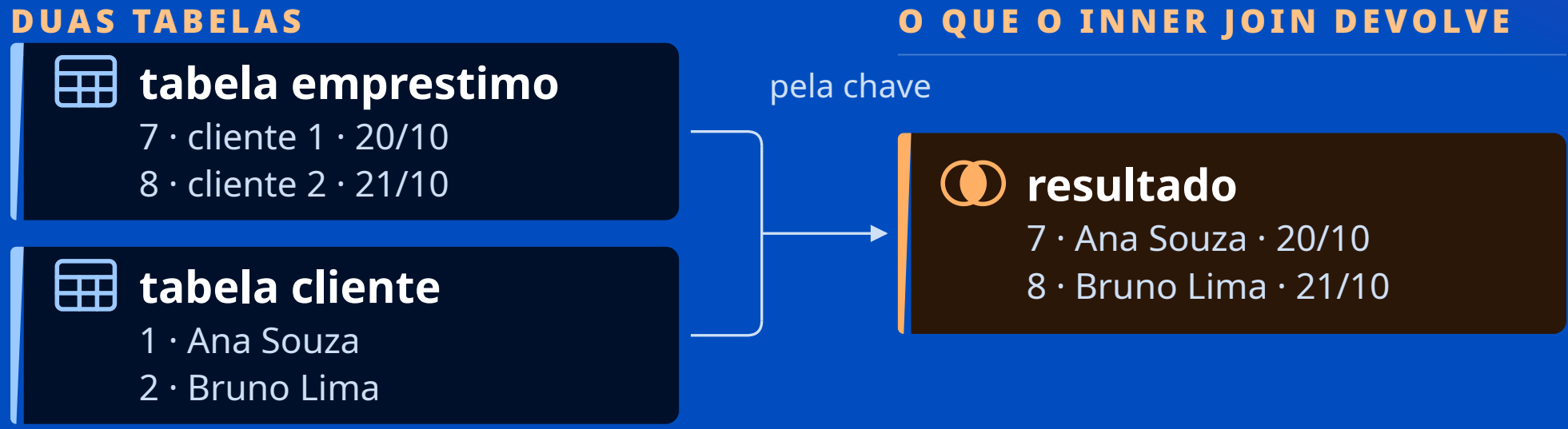
INNER JOIN — a solução

```
SELECT emprestimo.data_retirada,
       cliente.nome
FROM   emprestimo
INNER JOIN cliente ON emprestimo.id_cliente = cliente.id_cliente;
```

GLOSSÁRIO

JOIN	juntar — cruzar duas tabelas
INNER JOIN	junção interna — só as linhas que têm par nas duas
ON	sobre / usando — a condição que liga as duas tabelas

O que acontece por dentro



Para cada linha de `emprestimo`, o banco procura a linha de `cliente` com o mesmo id e **cola as duas**.

Apelidos de tabela — para não escrever tanto

```
SELECT e.data_retirada, c.nome
FROM   emprestimo e
INNER JOIN cliente c ON e.id_cliente = c.id_cliente;
```

`emprestimo e` quer dizer: *"a partir de agora, chame* `emprestimo` *de* `e` *"*.

Use apelidos **curtos e óbvios**: `e` para emprestimo, `c` para cliente, `j` para jogo. Apelido como `a`, `b`, `t1` deixa a consulta ilegível daqui a um mês.

JOIN com três tabelas

"Quais jogos saíram em cada empréstimo, e por quanto?"

```
SELECT e.data_retirada,
       j.titulo AS "Jogo",
       i.preco_pago
FROM   emprestimo e
INNER JOIN item_emprestimo i ON e.id_emprestimo = i.id_emprestimo
INNER JOIN jogo            j ON i.id_jogo       = j.id_jogo;
```

A tabela do **meio** — a do N:N, com uma chave estrangeira para cada lado — é a que liga as outras duas. Um `INNER JOIN ... ON` para cada ligação.

Checagem do Bloco 1

Complete o `ON`:

```
SELECT c.nome, e.data_retirada
FROM   emprestimo e
INNER JOIN cliente c ON _____________ ;
```

- E se eu quisesse também o **título do jogo** de cada empréstimo?

forms.gle/oD9QmiuXY9TQFn3T6

Intervalo

10 minutos

Bloco 2

LEFT JOIN — e quem não tem par?

⚠ O problema que o INNER esconde

"Liste todos os clientes e os empréstimos de cada um."

Com `INNER JOIN`, a **Carla** — que se cadastrou mas nunca alugou — **não aparece**.

> O `INNER JOIN` só mostra quem tem par nas **duas** tabelas. Quem não tem par simplesmente **some do resultado** — sem aviso.

LEFT JOIN — todo mundo da esquerda

INNER JOIN — SÓ QUEM TEM PAR

Ana · 2 empréstimos
aparece 2x

Bruno · 1 empréstimo
aparece 1x

Carla · nenhum
NÃO aparece

LEFT JOIN — TODOS DA ESQUERDA

Ana · 2 empréstimos
aparece 2x

Bruno · 1 empréstimo
aparece 1x

Carla · nenhum
aparece com NULL

O `LEFT JOIN` escrito

```
SELECT c.nome, e.data_retirada
FROM   cliente c
LEFT JOIN emprestimo e ON c.id_cliente = e.id_cliente;
```

A Carla aparece, com `NULL` na coluna do empréstimo.

Quando usar cada um

Pergunta	JOIN
"Quais jogos existem, com o nome da plataforma?"	`INNER` — todo jogo tem plataforma
"Todos os clientes e seus empréstimos"	`LEFT` — inclusive quem nunca alugou
"Quais clientes **nunca** alugaram?"	`LEFT` + `WHERE e.id_emprestimo IS NULL`

A terceira linha é um truque muito usado no mercado: `LEFT JOIN` e depois filtrar pelo `NULL` encontra exatamente **quem não tem par**.

⚠ Erro comum — o `WHERE` que mata o `LEFT JOIN`

```sql
-- Errado: vira INNER JOIN disfarçado
SELECT c.nome, e.data_retirada
FROM   cliente c
LEFT JOIN emprestimo e ON c.id_cliente = e.id_cliente
WHERE  e.data_retirada > '2026-10-01';
```

A Carla tem `NULL` em `data_retirada`, e `NULL > data` não é verdadeiro — então o `WHERE` joga ela fora de novo. Condição sobre a tabela da direita vai no `ON`, não no `WHERE`.

Checagem do Bloco 2

Qual JOIN você usaria?

- "Quais jogos foram alugados e por quem?"
- "Todos os jogos do catálogo, com a quantidade de vezes que foram alugados"
- "Quais jogos nunca saíram da prateleira?"



forms.gle/zTcGYrga5Mo5GvpBA

Intervalo

10 minutos

Bloco 3

Agregações — resumindo muitas linhas em uma

As funções de agregação

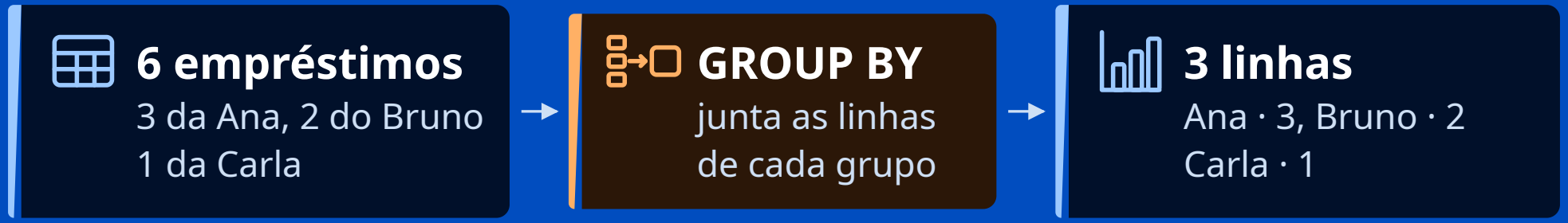
```sql
SELECT COUNT(*)              AS "Quantos jogos",
       AVG(ano_lancamento)   AS "Ano médio",
       MIN(ano_lancamento)   AS "Mais antigo",
       MAX(ano_lancamento)   AS "Mais novo"
FROM   jogo;
```

GLOSSÁRIO

`COUNT`	contar
`SUM`	somar
`AVG`	*average* — média
`MIN` / `MAX`	mínimo / máximo

GROUP BY — resumir por grupo

MUITAS LINHAS VIRAM UMA POR GRUPO




```sql
SELECT c.nome, COUNT(*) AS total_emprestimos
FROM   emprestimo e
INNER JOIN cliente c ON e.id_cliente = c.id_cliente
GROUP BY c.nome;
```

A regra do `GROUP BY`

Toda coluna que aparece no `SELECT` e **não** está dentro de uma função de agregação **precisa** estar no `GROUP BY`.

```sql
-- Erro: data_retirada não está no GROUP BY nem é agregada
SELECT c.nome, e.data_retirada, COUNT(*) FROM emprestimo e
INNER JOIN cliente c ON e.id_cliente = c.id_cliente
GROUP BY c.nome;
```

```
ERROR: column "e.data_retirada" must appear in the GROUP BY clause
or be used in an aggregate function
```

`HAVING` — filtrar depois de agrupar

```sql
SELECT c.nome, COUNT(*) AS total
FROM   emprestimo e
INNER JOIN cliente c ON e.id_cliente = c.id_cliente
GROUP BY c.nome
HAVING COUNT(*) > 2;
```

`WHERE` filtra **linhas**, antes de agrupar.
`HAVING` filtra **grupos**, depois de agrupar.
Não dá para usar `COUNT(*)` no `WHERE` — naquele momento a contagem ainda não existe.

Juntando tudo — a consulta completa

```
SELECT c.nome                  AS "Cliente",
       COUNT(e.id_emprestimo) AS "Empréstimos"
FROM   cliente c
LEFT JOIN emprestimo e ON c.id_cliente = e.id_cliente
GROUP BY c.nome
ORDER BY COUNT(e.id_emprestimo) DESC;
```

"Quantos empréstimos cada cliente fez, do que mais alugou para o que menos alugou — incluindo quem nunca alugou."

Checagem do Bloco 3

Escreva a consulta para cada pergunta da dona:

- "Quantos jogos eu tenho de cada plataforma?"
- "E só as plataformas com mais de dois?"
- "Quantos empréstimos cada cliente fez, **incluindo quem não fez nenhum**?"



forms.gle/wvhFdjbj4jTqqGYK7

Intervalo

10 minutos

MÃO NA MASSA

Perguntas que só o JOIN responde

1. Escrevam **5 perguntas do negócio** que exigem mais de uma tabela
2. Pelo menos uma com `INNER JOIN` de **duas** tabelas
3. Pelo menos uma com `INNER JOIN` de **três** tabelas
4. Pelo menos uma com `LEFT JOIN`, que inclua quem **não tem par**
5. Pelo menos uma com `GROUP BY` e `COUNT`
6. Rodem **duas** delas também no notebook, com `fetchall()`

MÃO NA MASSA

Lembrete de entrega

GLOSSÁRIO

`grupoNN-aulaNN-artefato.<ext>` `grupo03-aula14-consultas-join.sql`

Cada consulta com o comentário da **pergunta em português** em cima, como na Aula 13.

MÃO NA MASSA

Enquanto vocês consultam

O que eu vou conferir:

- O `ON` está na forma **FK = PK**?
- A consulta de três tabelas começou pela tabela do **meio**?
- No `LEFT JOIN`, o registro **sem par** apareceu mesmo no resultado?
- Toda coluna não agregada está no `GROUP BY`?
- Alguém usou `COUNT(*)` onde deveria contar a **coluna**?

Terminou? Tente: *"qual foi o mês com mais empréstimos?"* Vai precisar de `GROUP BY`, `ORDER BY` e `LIMIT 1` ao mesmo tempo.

MÃO NA MASSA

Devolutiva — passo de mesa em mesa

Eu passo em cada grupo. Deixem **uma** consulta com JOIN na tela:

- A pergunta em português
- O SQL, com o `ON` explicado — **de qual chave para qual chave**
- O resultado rodando

> Na hora do `ON`, aponte no DER de vocês: "essa linha do desenho é esse `ON` do SQL". É o fechamento de nove aulas de modelagem.

PARA CASA · OPCIONAL

Antes da próxima aula

Testar em casa: escreva a consulta *"qual o jogo mais alugado do catálogo?"*. Dica: `GROUP BY`, `ORDER BY ... DESC` e `LIMIT 1`.

Tarefa opcional: transforme uma das suas consultas de hoje numa pergunta que o dono do negócio faria de verdade, e escreva a resposta em uma frase em português — como num relatório.

Para treinar: `locadora-exercicios-14-juncoes.sql` — oito perguntas, incluindo a armadilha do `WHERE` que desfaz o `LEFT JOIN`. Como fica a entrega do grupo: `locadora-exemplo-14-consultas-join.sql`.

Para ver acontecer: demonstrações da Aula 14 — acompanhar o JOIN parear linha a linha, ver quem some no INNER, e o GROUP BY dobrando as linhas — e oito perguntas com resposta na hora. No site da UC, na página da aula.

Para a próxima aula

Vocês escreveram consultas ótimas hoje. **Como guardar uma consulta com nome, para não reescrever toda vez?**

UniSenac Centro Universitário RS
Senac
Fecomércio
IFEP
Sesc
Sindicatos Empresariais