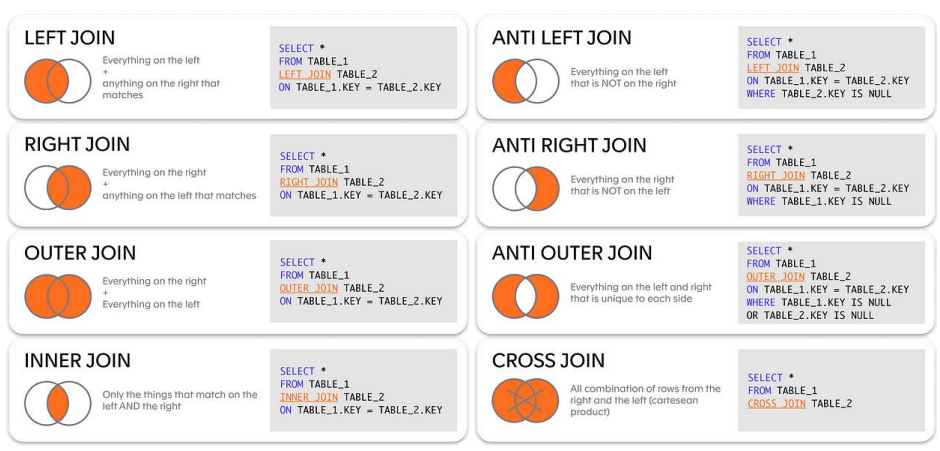
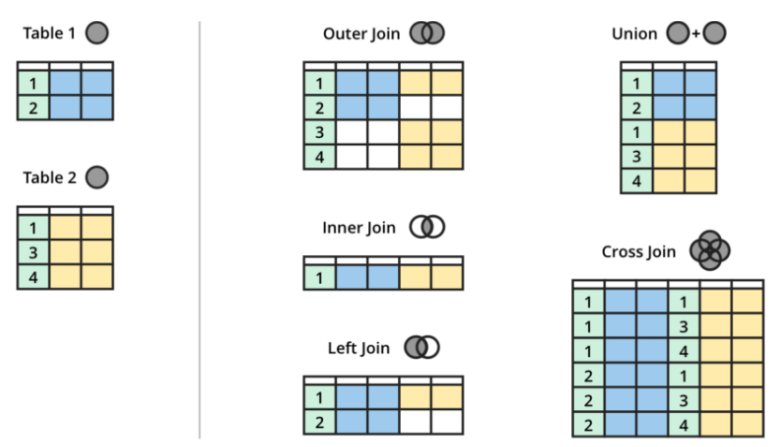
**BD2 – Prova - Resumo**

**SQL avançado**

**SQL JOINS: Joins servem para juntar dados de mais de uma tabela.**



**LEFT JOIN: pega tudo da tabela da esquerda e só o que tiver da tabela da direita -** (se não tiver, preenche NULL)

SELECT pessoas.nome, cidades.nome\_cidade

FROM pessoas -- mostra todas as pessoas, mesmo que elas não tenham cidade

LEFT JOIN cidades ON pessoas.id = cidades.id\_pessoa;

**RIGHT JOIN: pega tudo da tabela da direita e só o que tiver da tabela da esquerda**

SELECT pessoas.nome, cidades.nome\_cidade

FROM pessoas -- mostra todas as cidades, mesmo que não tenham pessoas associadas

RIGHT JOIN cidades ON pessoas.id = cidades.id\_pessoa;

**ANTI LEFT JOIN (simulação): mostra quem NÃO tem ligação -** simula com LEFT JOIN + WHERE campo IS NULL

SELECT pessoas.nome

FROM pessoas -- mostra pessoas que NÃO têm cidade associada

LEFT JOIN cidades ON pessoas.id = cidades.id\_pessoa

WHERE cidades.nome\_cidade IS NULL;

**Consultas com GROUP BY: agrupa resultados iguais**

SELECT genero, COUNT(\*) AS quantidade

FROM pessoas -- conta quantas pessoas de cada gênero existem

GROUP BY genero;

**Consultas com GROUP BY e HAVING: HAVING é para filtrar depois do agrupamento (não usa WHERE)**

SELECT genero, COUNT(\*) AS quantidade

FROM pessoas -- mostra apenas gêneros com mais de 2 pessoas

GROUP BY genero

HAVING COUNT(\*) > 2;

**Subconsultas dentro da cláusula WHERE: usar um SELECT dentro do WHERE**

SELECT nome

FROM pessoas

WHERE id IN (SELECT id\_pessoa FROM cidades WHERE nome\_cidade = 'São Paulo');

**Subconsultas dentro da cláusula FROM: um SELECT dentro do FROM**

* Sem JOIN:

SELECT sub.nome -- mostra nomes de pessoas do gênero feminino

FROM (SELECT nome FROM pessoas WHERE genero = 'F') AS sub;

* Com JOIN:

SELECT p.nome, c.nome\_cidade

FROM (SELECT \* FROM pessoas WHERE genero = 'F') AS p -- pega só mulheres e a cidade delas

JOIN cidades AS c ON p.id = c.id\_pessoa;

**Subconsultas em uma coluna: consulta dentro do SELECT para trazer um valor extra**

SELECT nome, -- mostra o nome da pessoa e quantas cidades ela está ligada

(SELECT COUNT(\*) FROM cidades WHERE cidades.id\_pessoa = pessoas.id) AS qtd\_cidades

FROM pessoas;

**CONCAT: serve para juntar (concatenar) textos (strings)**

SELECT CONCAT(nome, ' - ', genero) AS pessoa\_genero

FROM pessoas; -- vai mostrar algo como: Ana - F ou Bruno - M

**Subconsultas que retornam mais de uma tupla: quando o resultado da subconsulta tem várias linhas**

deve usar IN, ANY ou ALL

SELECT nome

FROM pessoas

WHERE id IN (SELECT id\_pessoa FROM cidades);

(Mostra pessoas que têm cidades cadastradas.)

**Funções de janela**

* São funções que permitem realizar cálculos sobre um conjunto de linhas relacionadas à linha atual.
* Diferente das funções de agregação normais (SUM, AVG, COUNT), que reduzem o número de linhas, as funções de janela mantêm cada linha no resultado.
* São usadas para fazer cálculos cumulativos, classificações, médias móveis e muito mais, sem precisar agrupar os dados.

Principais características

* Não reduzem o número de linhas: Cada linha continua aparecendo no resultado.
* Usam sempre OVER(): É obrigatório definir a janela de dados usando a cláusula OVER
* Não usadas para análises avançadas: Como rankings, somatórios acumulados, médias móveis, etc.

**ROW\_NUMBER(): numera cada linha, de acordo com uma ordem definida**

SELECT nome,

ROW\_NUMBER() OVER (ORDER BY nome) AS numero -- dá um número para cada pessoa baseado no nome

FROM pessoas;

**RANK(): cria um ranking, mas em caso de empate eles ficam na mesma posição e o próximo número pula**

SELECT nome, salario,

RANK() OVER (ORDER BY salario DESC) AS posição -- ranqueia salários do maior para o menor

FROM pessoas;

**SUM() OVER (...): soma acumulada das linhas, respeitando uma ordem**

SELECT nome, salario, -- acumula os salários à medida que os nomes aparecem

SUM(salario) OVER (ORDER BY nome) AS soma\_acumulada

FROM pessoas;

**PARTITION BY dentro do OVER: divide os dados em partes (grupos) antes de aplicar a função**

SELECT nome, genero, salario,

SUM(salario) OVER (PARTITION BY genero) AS soma\_por\_genero -- soma salários separados para cada gênero

FROM pessoas;

**OVER() + RANK() juntos: classifica dados dentro de um grupo**

SELECT nome, genero, salario, -- faz um ranking de salário dentro de cada gênero

RANK() OVER (PARTITION BY genero ORDER BY salario DESC) AS rank\_genero

FROM pessoas;