Nesse vídeo, vamos estudar sobre as **subconsultas**, que nada mais são do que consultas dentro de outras consultas. Para exemplificar, vamos usar duas tabelas. Na tabela 1, temos o campo "X" que apresenta letras e o campo "Y" que possui valores numéricos. Na tabela 2, temos apenas o campo "Y", também com números:

Nosso primeiro objetivo será fazer uma consulta que liste todos os registros da tabela 1 cujo Y está presente na tabela 2. Como sabemos que a tabela 2 contém apenas dois valores (1 e 2), uma opção seria usar a cláusula WHERE junto do IN:

Texto

O conteúdo gerado por IA pode estar incorreto.

A condição WHERE Y IN (1,2) significa que vamos filtrar registros cujo Y esteja **dentro** do conjunto de valores (1,2):

Assim, a consulta retornará exatamente o que queríamos. **Entretanto**, se um novo valor for adicionado à tabela 2, seremos obrigados a reescrever o script para que continue funcionando do modo que desejamos. Por exemplo, acrescentando um registro à tabela 2:

Precisaríamos embutir manualmente o novo valor no script:

Texto

O conteúdo gerado por IA pode estar incorreto.

Essa prática é um exemplo de ***hardcoding***. Caso ache interessante, você pode pesquisar mais sobre o assunto.

Essa situação se tornará insustentável, se a tabela 2 for modificada constantemente. Uma solução, então, é usar uma subconsulta:

Texto

O conteúdo gerado por IA pode estar incorreto.

Dessa forma, vamos filtrar registros cujo Y esteja dentro do conjunto de valores resultantes da subconsulta SELECT Y FROM tab2 (que, por sua vez, retorna todos os valores da tabela 2). Teremos uma consulta dentro de outra - ou seja, uma subconsulta. E o resultado seria o seguinte:

A seguir, criaremos outro exemplo. Vejamos a consulta a seguir e o seu retorno (que chamaremos de tabela 3):



Vamos supor que precisamos filtrar os valores da tabela 3 cujo "NEW\_Y" seja igual a 3. Usando uma subconsulta, poderíamos fazer o seguinte:

**SELECT** Z.X, Z.NEW\_Y **FROM** (**SELECT** X, SUM(Y) **as** NEW\_Y **FROM** tab1 **GROUP** **BY** X) Z **WHERE** Z.NEW\_Y = 3;

No trecho FROM (SELECT X, SUM(Y) as NEW\_Y FROM tab1 GROUP BY X) Z, estamos usando como subconsulta o SELECT que fizemos anteriormente (cujo retorno é a tabela 3). Além disso, declaramos "Z" como *alias* e, por fim, colocamos a condição desejada: WHERE Z.NEW\_Y = 3:

Para tornar a explicação mais clara, vamos praticar com alguns exemplos no nosso banco de dados "sucos\_vendas". Abriremos o MySQL Workbench, criaremos um novo script e começaremos consultando os bairros da tabela de vendedores:

Texto

O conteúdo gerado por IA pode estar incorreto.

O retorno mostrará que existem 4 bairros onde há escritórios de vendedores: Tijuca, Jardins, Copacabana e Santo Amaro. Em seguida, vamos selecionar os clientes cujos bairros têm escritórios de vendedores. Uma forma de buscar esses dados é a seguinte:

Interface gráfica do usuário, Texto

O conteúdo gerado por IA pode estar incorreto.

Com essa consulta, conseguiremos as informações que desejamos, porém digitar os nomes dos bairros um a um não é conveniente, visto que a qualquer momento podem surgir novos escritórios em outros bairros e seremos forçados a reescrever a consulta.

Então, vamos encontrar uma forma melhor de buscar esses dados. Sabemos que a lista **Tijuca, Jardins, Copacabana, Santo Amaro** é o retorno de SELECT DISTINCT BAIRRO FROM tabela\_de\_vendedores. Desse modo, podemos usar essa seleção como uma subconsulta:

Interface gráfica do usuário, Texto

O conteúdo gerado por IA pode estar incorreto.

Em outras palavras: da tabela de clientes, vamos selecionar clientes cujo bairro esteja **dentro** de que conjunto? Do resultado de SELECT DISTINCT BAIRRO FROM tabela\_de\_vendedores. Assim, obtemos o mesmo retorno, porém de uma forma mais sustentável - se os bairros dos vendedores mudarem, a consulta continua funcionando.

Vamos fazer outro exemplo, dessa vez consultando o maior preço de cada tipo de embalagem:

Texto

O conteúdo gerado por IA pode estar incorreto.

Assim, descobrimos que a garrafa mais cara custa R$13,31; o PET mais caro é R$38,01; e a lata, R$4,56. Então, faremos uma seleção das embalagens cujo preço mais caro é maior ou igual que R$10. Utilizaremos nossa última consulta como uma subconsulta com apelido "X":

**SELECT** X.EMBALAGEM, X.PRECO\_MAXIMO **FROM**

(**SELECT** EMBALAGEM, MAX(PRECO\_DE\_LISTA) **AS** PRECO\_MAXIMO **FROM** tabela\_de\_produtos

**GROUP** **BY** EMBALAGEM) X **WHERE** X.PRECO\_MAXIMO >= 10;

Assim, o MySQL está interpretando a subconsulta como se fosse uma tabela com *alias* "X", então X.EMBALAGEM e X.PRECO\_MAXIMO referem-se à seleção dos campos dessa tabela "X". Ao final, temos a cláusula WHERE que filtra apenas os preços maiores ou iguais a R$10. O resultado dessa consulta mostra que apenas embalagens PET e garrafas têm o preço máximo maior que R$10.

Com esse exemplo, constatamos que uma consulta pode assumir o papel de uma tabela - foi o que ocorreu com a subconsulta apelidada de "X". Inclusive, seria possível até usar JOIN e outros comandos nesses casos.