Técnico em Desenvolvimento de Sistemas

Consulta de dados: seleção, filtragem, operadores lógicos, operadores aritméticos, consultas simples e compostas, ordenação de resultados, cálculos em SQL, funções internas básicas (COUNT, SUM, AVG, MIN, MAX)

Para iniciar as consultas em SQL, você criará primeiramente um novo banco de dados chamado **super** e depois algumas tabelas com registros. Isso permitirá que você tenha uma experiência prática quando começar a fazer perguntas mais tarde.

```
CREATE DATABASE super;
```

Imagine o seguinte cenário: você foi contratado para realizar consultas de produtos de uma empresa do ramo de supermercado. A tabela principal para essa tarefa é **Produtos**, que contará com ID, nome, categoria, valor e quantidade em estoque. O *script* para criação é o seguinte:

```
CREATE TABLE Produtos(
idProduto INT NOT NULL AUTO_INCREMENT,
nomeProduto VARCHAR(200),
categoriaProduto VARCHAR(200),
valorVendaProduto DOUBLE(9,2),
quantidadeProduto INT,
PRIMARY KEY (idProduto)
);
```

Serão incluídos os seguintes dados para demonstração:

```
INSERT into Produtos
(nomeProduto,categoriaProduto,
```

```
valorVendaProduto,
quantidadeProduto) values
('Refrigerante','Bebidas','5.50',
300),
('Arroz 5kg','Mercearia','8.50',
100),
('Feijão','Mercearia','6.50',
800),
('Detergente','Limpeza','2.10',
100),
('Leite','Laticínios','2.70',600),
('Bolacha recheada',NULL,'1.50',200),
('Leite condensado','Mercearia','4.50',
500);
```

idProduto	nomeProduto	categoriaProduto	valorVendaProduto	quantidadeProduto
1	Refrigerante	Bebidas	5.50	300
2	Arroz 5kg	Mercearia	8.50	100
3	Feijão	Mercearia	6.50	800
4	Detergente	Limpeza	2.10	100
5	Leite	Laticínios	2.70	600
6	Bolacha recheada	NULL	1.50	200
7	Leite condensado	Mercearia	4.50	500

Tabela 1 – Listagem dos produtos

Após obter a definição da tabela de produtos com os registros de demonstração, você poderá avançar para conhecer e praticar os comandos de consulta na linguagem SQL (*structured query language*). Para melhor aproveitamento de seu estudo, faça, no MySQL Workbench ou no editor de sua preferência, as consultas exemplificadas no decorrer deste texto e verifique os

resultados obtidos.

Seleção de dados com SELECT

A instrução **SELECT** em SQL permite recuperar os dados a partir de uma tabela ou mais do banco de dados, de acordo com cláusulas (por exemplo **FROM** e **WHERE**) que especificam critérios.

A sintaxe para isso é:

SELECT coluna1, coluna2 FROM tabela1, tabela2 WHERE coluna2 = 'valor';

Aprenda um pouco mais sobre o que compõe a instrução SQL anterior:

Cláusula SELECT
Especifica uma ou mais colunas a serem exibidas. Para especificar várias colunas, use uma vírgula e um espaço entre os nomes das colunas. Para recuperar todas as colunas, use o caractere curinga * (asterisco).

Cláusula FROM
Especifica uma ou mais tabelas a serem consultadas. Use uma vírgula e espaço entre os nomes das tabelas ao especificar várias tabelas.

Cláusula WHERE
Seleciona apenas as linhas nas quais a coluna especificada contém o valor especificado. O valor, quando é textual, é colocado entre aspas simples (por exemplo, **WHERE nomeProduto = 'Leite'**).

Ponto e vírgula (;)

É o terminador da instrução. Tecnicamente, se você está executando apenas uma instrução, não precisa do terminador de instrução, entretanto se você estiver executando mais de uma cláusula, ele será necessário.

SQL não faz distinção entre maiúsculas e minúsculas (por exemplo, SELECT é o mesmo que select). Para melhor legibilidade, e porque é um padrão bastante usado, serão adotadas letras maiúsculas para cláusulas e comandos e letras minúsculas para todo o resto.

Confira a seguir alguns exemplos da instrução **SELECT** em SQL:

Para listar todas as informações que constam nas colunas da tabela de produtos, utilize a seguinte instrução:

Exemplo 1:

SELECT * FROM Produtos

idProduto	nomeProduto	categoriaProduto	valorVendaProduto	quantidadeProduto
1	Refrigerante	Bebidas	5.50	300
2	Arroz 5kg	Mercearia	8.50	100
3	Feijão	Mercearia	6.50	800
4	Detergente	Limpeza	2.10	100
5	Leite	Laticínios	2.70	600
6	Bolacha recheada	NULL	1.50	200
7	Leite condensado	Mercearia	4.50	500

Tabela 2 – Listagem de todos os produtos

Existem, porém, situações em que são necessárias apenas colunas específicas da(s) tabela(s) pesquisada(s) e, para isso, você deverá substituir o caractere asterisco pelo nome das colunas que deseja. Nesse caso, se você quiser que a pesquisa traga apenas os nomes dos produtos com seus respectivos valores, utilize **nomeProduto**, **valorVendaProduto**.

Exemplo 2:

SELECT nomeProduto, valorVendaProduto FROM Produtos

nomeProduto	valorVendaProduto
Refrigerante	5.50
Arroz 5kg	8.50
Feijão	6.50
Detergente	2.10
Leite	2.70
Bolacha recheada	1.50
Leite condensado	4.50

Tabela 3 – Listagem de todos os produtos com as colunas **nomeProduto** e **valorVendaProduto**

Filtrando dados com WHERE

Até agora, foram retornadas as informações de todos os registros da tabela. Este é o comportamento padrão de uma consulta. Para retornar um conjunto de dados mais seletivo, serão necessários filtros, com a utilização da cláusula **WHERE**.

Uma cláusula **WHERE** normalmente segue a seguinte sintaxe:

WHERE nome_coluna operador_de_comparacao valor

O operador de comparação em uma cláusula **WHERE** define como a coluna especificada deve ser comparada com o valor. Observe agora alguns operadores de comparação comuns:

Operador	O que faz
=	Teste de igualdade
<> ou !=	Teste de desigualdade (o operador != não é padronizado, mas diversos SGBDs o aceitam)
<	Teste para menor que
>	Teste para maior que
<=	Teste para menor ou igual a
>=	Teste para maior ou igual a
BETWEEN	Testa se um valor está dentro de um determinado intervalo
IN	Testa se o valor de uma linha está contido em um conjunto de valores especificados
EXISTS	Testa se existem linhas, dadas as condições especificadas
LIKE	Testa se um valor corresponde a uma string específica
IS NULL	Testa para NULL valores
IS NOT NULL	Testa para todos os valores, exceto NULL

O operador de comparação mais simples que pode ser usado é o operador de igualdade (=). Imagine que você quisesse encontrar os produtos cujo nome fosse “leite”:

Exemplo 3:

SELECT * FROM Produtos WHERE nomeProduto='Leite'

idProduto	nomeProduto	categoriaProduto	valorVendaProduto	quantidadeProduto
5	Leite	Laticínios	2.70	600

Tabela 4 – Produtos com o nome “leite”

Note que foi utilizada a cláusula **WHERE** seguida do nome da coluna pela qual se deseja

filtrar **nomeProduto**, em que o valor da coluna é igual a “leite”.

Para retornar apenas as colunas **nomeProduto** e **valorVendaProduto**, com base nos mesmos critérios mencionados, use a seguinte instrução:

Exemplo 4:

SELECT nomeProduto, valorVendaProduto FROM Produtos WHERE nomeProduto='Leite'

nomeProduto	valorVendaProduto
Leite	2.70

Tabela 5 – Produtos cujo nome seja “leite” nas colunas **nomeProduto** e **valorVendaProduto**

Semelhantemente, pode-se utilizar o operador <> para comparar diferenças:

Exemplo 5:

SELECT nomeProduto, categoriaProduto, valorVendaProduto FROM Produtos WHERE categoriaProduto <> 'Mercearia'

nomeProduto	categoriaProduto	valorVendaProduto
Refrigerante	Bebidas	5.50
Detergente	Limpeza	2.10
Leite	Laticínios	2.70

Tabela 6 – Produtos cuja categoria seja diferente de “mercearia”

Operadores lógicos AND, OR, NOT

Ainda sobre filtros nas pesquisas, é possível aplicar esses filtros utilizando operadores lógicos booleanos, do mesmo modo que a maioria das linguagens de programação: **AND** (operador de conjunção **E**), **OR** (operador de disjunção **OU**), e **NOT** (operador de negação **NÃO**).

Para consultar produtos em que o valor de venda (**valorVendaProduto**) seja, por exemplo, 1.50 e a quantidade (**quantidadeProduto**) seja 200, pode-se usar o operador **AND** na cláusula **WHERE**.

Exemplo 6:

SELECT * FROM Produtos WHERE valorVendaProduto = 1.50 AND quantidadeProduto = 200

idProduto	nomeProduto	categoriaProduto	valorVendaProduto	quantidadeProduto
6	Bolacha recheada		1.50	200

Tabela 7 – Produtos cujo valor de venda seja 1.50 e quantidade igual a 200

Note que a pesquisa só trará registros na tabela em que ambas as condições são satisfeitas.

Para encontrar produtos em que a categoria (**categoriaProduto**) seja “limpeza”, por exemplo, ou o valor de venda (**valorVendaProduto**) seja 5.50, pode-se usar o operador **OR**.

Exemplo 7:

SELECT * FROM Produtos WHERE categoriaProduto = 'Limpeza' OR valorVendaProduto = 5.50

idProduto	nomeProduto	categoriaProduto	valorVendaProduto	quantidadeProduto
1	Refrigerante	Bebidas	5.50	300
4	Detergente	Limpeza	2.10	100

Tabela 8 – Produtos da categoria “limpeza” ou com valor de venda de 5.50

As condições **AND** e **OR** também podem ser encadeadas. Considere, por exemplo, que você queira encontrar produtos da categoria (**categoriaProduto**) “limpeza” ou produtos da categoria (**categoriaProduto**) “mercearia”, que tenham, independentemente da categoria, o valor de venda

(**valorVendaProduto**) igual a 6.50.

Exemplo 8:

SELECT * FROM Produtos WHERE (categoriaProduto ='Limpeza' or categoriaProduto='Mercearia') and valorVendaProduto = 6.50

idProduto	nomeProduto	categoriaProduto	valorVendaProduto	quantidadeProduto
3	Feijão	Mercearia	6.50	800

Tabela 9 – Produtos da categoria “limpeza” ou “mercearia” e com valor de 6.50 independentemente da categoria

Note a necessidade de utilizar parênteses para priorizar a execução da condição “categoria igual à ‘limpeza’ ou ‘mercearia’”. Assim como em muitas linguagens de programação, a SQL prioriza a execução do operador AND sobre o operador OR.

SELECT * FROM Produtos WHERE categoriaProduto ='Limpeza' or categoriaProduto= 'Mercearia' AND valorVendaProduto = 6.50

Sem os parênteses, como o exemplo anterior, a primeira condição a ser avaliada seria categoriaProduto='Mercearia' AND valorVendaProduto = 6.50. Só depois se avaliaria categoriaProduto ='Limpeza' or.... O resultado prático disso é que a consulta retornaria produtos da categoria “limpeza” de qualquer valor, mas apenas os produtos da categoria “mercearia” com o valor de 6.50 seriam retornados. Isso contrariaria a necessidade enunciada anteriormente, que afirmava que o valor do produto deve ser de 6.50 independentemente da categoria. Daí a necessidade de primeiro comparar as categorias e depois o valor.

Em outro exemplo, se você quisesse uma listagem de todos os produtos, exceto os da categoria (**categoriaProduto**) “mercearia”, poderia usar o operador **NOT**.

Exemplo 9:

SELECT * FROM Produtos WHERE NOT categoriaProduto ='Mercearia'

idProduto	nomeProduto	categoriaProduto	valorVendaProduto	quantidadeProduto
1	Refrigerante	Bebidas	5.50	300
4	Detergente	Limpeza	2.10	100
5	Leite	Laticínios	2.70	600

Tabela 10 – Produtos cuja categoria não seja “mercearia”

No caso anterior, o operador **NOT** acaba resultando semelhantemente a uma comparação de diferença. É claro que o **NOT** pode ser usado para expressões mais complexas, geralmente envolvidas por parênteses.

Operadores de comparação (<, >, <=, >=)

Podem ser utilizados os operadores de comparação <, >, <=, >= para ampliar as opções de filtragem em suas consultas.

Imagine que você quer encontrar os produtos cuja quantidade em estoque seja maior ou igual a 300 unidades.

Exemplo 10:

SELECT * FROM Produtos WHERE quantidadeProduto >= 300

idProduto	nomeProduto	categoriaProduto	valorVendaProduto	quantidadeProduto
1	Refrigerante	Bebidas	5.50	300
3	Feijão	Mercearia	6.50	800
5	Leite	Laticínios	2.70	600
7	Leite condensado	Mercearia	4.50	500

Tabela 11 – Produtos cuja quantidade seja maior ou igual a 300

Considere agora um exemplo de consulta de produtos cujos valores de venda sejam

menores ou iguais a 5 reais.

Exemplo 11:

SELECT * FROM Produtos WHERE valorVendaProduto <= 5

idProduto	nomeProduto	categoriaProduto	valorVendaProduto	quantidadeProduto
4	Detergente	Limpeza	2.10	100
5	Leite	Laticínios	2.70	600
6	Bolacha recheada		1.50	200
7	Leite condensado	Mercearia	4.50	500

Tabela 12 – Produtos cujo valor de venda seja menor ou igual a 5

Operador BETWEEN

Esse operador permite que você selecione dados por meio de intervalos de valores ao retornar os resultados de uma consulta.

A sintaxe para seu uso é:

SELECT * FROM tabela WHERE coluna BETWEEN valor1 AND valor2;

Como exemplo, pode-se usar a cláusula **BETWEEN** para retornar os registros cujos preços estejam entre 5 e 10 reais.

Exemplo 12:

SELECT * FROM Produtos WHERE valorVendaProduto BETWEEN 5 AND 10

idProduto	nomeProduto	categoriaProduto	valorVendaProduto	quantidadeProduto
1	Refrigerante	Bebidas	5.50	300
2	Arroz 5kg	Mercearia	8.50	100
3	Feijão	Mercearia	6.50	800

Tabela 13 – Produtos cujos preços estejam entre 5 e 10 reais

Operadores de existência IN, NOT IN

Para filtrar o valor de um campo específico da tabela, considerando uma lista de possibilidades, você pode utilizar o operador **IN**. Observe que, diferentemente do operador de comparação de igualdade (=), que verifica se dois valores são iguais, o operador **IN** verifica se o valor de um campo se encontra em uma lista.

Abaixo, serão listados todos os produtos cuja categoria seja “limpeza”, “bebidas” e “laticínios”.

Exemplo 13:

SELECT * FROM Produtos WHERE categoriaProduto in ('Limpeza','Bebidas','Laticínios')

idProduto	nomeProduto	categoriaProduto	valorVendaProduto	quantidadeProduto
1	Refrigerante	Bebidas	5.50	300
4	Detergente	Limpeza	2.10	100
5	Leite	Laticínios	2.70	600

Tabela 14 – Produtos cuja categoria seja “limpeza”, “bebidas” e “laticínios”

Desta forma, é possível usar o **NOT IN** para negar essa condição:

Exemplo 14:

SELECT * FROM Produtos WHERE categoriaProduto NOT IN ('Limpeza','Bebidas','Laticínios')

idProduto	nomeProduto	categoriaProduto	valorVendaProduto	quantidadeProduto
2	Arroz 5kg	Mercearia	8.50	100
3	Feijão	Mercearia	6.50	800
7	Leite condensado	Mercearia	4.50	500

Tabela 15 – Produtos cuja categoria não seja “limpeza”, “bebidas” e “laticínios”

O operador **IN** pode ser usado também com subconsultas – ou seja, a lista de valores possíveis pode vir de outra cláusula **SELECT**. As subconsultas serão estudadas em seguida, ainda neste conteúdo.

Operador LIKE

O operador **LIKE** é utilizado para buscar um determinado texto dentro de um campo com valores textuais. Para efetuar esse tipo de consulta, utilize o caractere % (símbolo de porcentagem) para indicar uma espécie de curinga %, ou seja, um texto qualquer que pode aparecer em alguma parte do campo a ser pesquisado.

A sintaxe para isso é:

SELECT colunas FROM tabela WHERE campo LIKE 'termo'

Nessa sintaxe, o '**termo**' pode ser informado na pesquisa de diversas formas com o caractere %. Confira os exemplos a seguir:

'%termo%'

Serão retornados os registros que **contêm** em qualquer parte do campo buscado o termo informado. Considerando, por exemplo, que o nome do produto seja “leite condensado” e o termo a ser pesquisado seja **'%eite%'**, então isso atenderia ao filtro **LIKE '%eite%'**.

'%termo'

Retornará todos os registros em que o valor do campo filtrado **termine** com o termo informado. O % serve para indicar que, mesmo que existam outros termos no início do campo, se ele terminar com o valor do termo buscado, ele será retornado nessa consulta. Exemplo: **'%Condensado'**.

'termo%'

Serão retornados os registros cujo valor do campo filtrado **começa** com o termo informado. O % indica que pode haver qualquer texto depois do termo informado. Exemplo: **'Leite%'**.

Para selecionar as colunas **nomeProduto** e **valorVendaProduto** da tabela de produtos, para as linhas nas quais a coluna **nomeProduto** contém o termo "Leite" em qualquer parte do campo, use esta instrução:

Exemplo 15:

SELECT nomeProduto, valorVendaProduto FROM Produtos WHERE nomeProduto LIKE '%leite%'

O conjunto de resultados subsequente pode ser semelhante a:

nomeProduto	valorVendaProduto
Leite	2.70
Leite condensado	4.50

Tabela 16 – Lista de todos os produtos que tenham a palavra "leite" em qualquer parte de seu nome

Observe pelo exemplo anterior que as comparações textuais em SQL, por padrão, também independem de letras maiúsculas e minúsculas. Dessa forma, compara-se “leite” e obtém-se “Leite”, por exemplo. De qualquer maneira, é comum que os SGBDs (sistemas gerenciadores de bancos de dados) ofereçam opções para manter esse mecanismo ou passar a considerar maiúsculas e minúsculas.

No operador **LIKE**, além do símbolo %, ainda há o caractere especial _ (*underscore* ou sublinhado), no qual se pode efetuar filtros mais exatos, buscando por um termo em uma posição específica do campo da pesquisa. O _ indica a quantidade de casas/caracteres antes ou depois do termo buscado. Confira as principais situações:

'_eite%'

Serão retornados os registros que contêm um caractere qualquer no começo do termo pesquisado. Retorno: ‘L**eite** Condensado’.

'%s__o'

Serão retornados os registros que comecem com a letra “s”, contenham dois caracteres quaisquer e, em seguida, a letra “o”. Retorno: ‘Leite Conden**s**ad**o**’.

O operador **LIKE** também pode ser usado sem os caracteres curinga (**LIKE 'Leite'**, por exemplo). Nesse caso, seu comportamento será exatamente como o do operador de igualdade.

Outra forma de encontrar o produto “leite” poderia ser consultar pelo seu ID, cujo nome, na coluna da tabela de produtos, é **idProduto**.

Exemplo 16:

SELECT * FROM Produtos WHERE idProduto=5

idProduto	nomeProduto	categoriaProduto	valorVendaProduto	quantidadeProduto
5	Leite Laticínios	2.70	600	

Tabela 17 – Listando os dados do produto cujo **idProduto** seja 5

Aqui, apenas um registro correspondeu à condição de **idProduto**=5, portanto, só uma linha foi recuperada.

Operador IS NULL e IS NOT NULL

Em uma consulta em SQL, caso deseje verificar se um campo está com valor **NULL**, ao invés de utilizar o operador de igualdade, você pode utilizar o operador **IS NULL**.

Exemplo 17:

SELECT * FROM Produtos WHERE categoriaProduto IS NULL

idProduto	nomeProduto	categoriaProduto	valorVendaProduto	quantidadeProduto
6	Bolacha recheada	NULL	1.50	200

Tabela 18 – Listando todos os produtos cuja categoria esteja com **NULL**

Caso precise encontrar todos os registros cujo campo não seja preenchido com **NULL**, use o operador **IS NOT NULL**.

Exemplo 18:

SELECT * FROM Produtos WHERE categoriaProduto IS NOT NULL

idProduto	nomeProduto	categoriaProduto	valorVendaProduto	quantidadeProduto
1	Refrigerante	Bebidas	5.50	300
2	Arroz 5kg	Mercearia	8.50	100
3	Feijão	Mercearia	6.50	800
4	Detergente	Limpeza	2.10	100
5	Leite	Laticínios	2.70	600
7	Leite condensado	Mercearia	4.50	500

Tabela 19 – Listando todos os produtos cuja categoria não seja **NULL**

O operador de igualdade e de diferença ignoram valores nulos – veja no exemplo 5 que o item de categoria nula não aparece no resultado da consulta. Assim, muitas vezes, é preciso usar o operador **OR** para completar a consulta. Revisitando o exemplo 5, seria possível adaptar a consulta para o seguinte:

Exemplo 19:

SELECT nomeProduto, categoriaProduto, valorVendaProduto FROM Produtos WHERE categoriaProduto <> 'Mercearia' OR categoriaProduto IS NULL

nomeProduto	categoriaProduto	valorVendaProduto
Refrigerante	Bebidas	5.50
Detergente	Limpeza	2.10
Leite	Laticínios	2.70
Bolacha recheada	NULL	1.50

Tabela 20 – Listando produtos cuja categoria seja diferente de “mercearia” ou NULL

Dessa maneira, o resultado traria todos os registros em que a categoria do produto seria nula ou diferente de “mercearia”.

Alias

Imagine que você esteja trabalhando com um banco de dados em que os nomes das colunas ou tabelas são longos ou difíceis de ler. Em situações como essa, você pode utilizar um **alias** para tornar esses nomes mais legíveis. Pode-se considerar os alias (lê-se “álias”) como apelidos para colunas ou tabelas.

Para definir um alias, utiliza-se a palavra-chave **AS**. A criação de um alias com **AS** é temporária e só existe enquanto durar a consulta para a qual foram criados:

Exemplo 20:

SELECT nomeProduto AS nome, valorVendaProduto AS valor FROM Produtos

nome	valor
Refrigerante	5.50
Arroz 5kg	8.50
Feijão	6.50
Detergente	2.10
Leite	2.70
Bolacha recheada	1.50
Leite condensado	4.50

Tabela 21 – Listando todos os produtos e inserindo um alias nas colunas: **nomeProduto** para nome e **valorVendaProduto** para valor

Neste momento, foi “dito” ao SQL para exibir a coluna **nomeProduto** como nome, e a coluna **valorVendaProduto** como valor. Note que são os alias que aparecem na descrição das colunas no resultado.

Operadores aritméticos

O SQL permite executar operações matemáticas durantes as consultas ao banco de dados. Para criar expressões aritméticas, são utilizados os seguintes operadores:

Operador	O que faz
+	Somar
-	Subtrair
*	Multiplicar
/	Dividir

Os operadores citados podem ser usados apenas em colunas do tipo numérico e em qualquer cláusula, exceto na **FROM**.

Multiplicação

No exemplo a seguir estará a multiplicação do valor de venda de cada produto por sua quantidade. Para isso, serão adotados os parênteses externos e atribuídos a um alias **AS Valor_Total**.

Exemplo 21:

SELECT (valorVendaProduto*quantidadeProduto) AS Valor_Total FROM Produtos

Valor_Total
1650.00
850.00
5200.00
210.00
1620.00
300.00
2250.00

Tabela 22 – Multiplicação da coluna **valorVendaProduto** por **quantidadeProduto**

Divisão

O operador aritmético para divisão é /, e deverá ficar entre os dois campos nos quais deve ocorrer a divisão.

Exemplo 22:

SELECT (quantidadeProduto / valorVendaProduto) AS Valor_Divisao FROM Produtos

Valor_Divisao
54.545455
11.764706
123.076923
47.619048
222.222222
133.333333
111.111111

Tabela 23 – Divisão da coluna **quantidadeProduto** por **valorVendaProduto**

Subtração

Observe a cláusula **SELECT** utilizando a operação de subtração entre dois campos.

Exemplo 23:

SELECT (quantidadeProduto - valorVendaProduto) AS Valor_Subtracao FROM Produtos

Valor_Subtracao
294.50
91.50
793.50
97.90
597.30
198.50
495.50

Tabela 24 – Subtraindo a coluna **quantidadeProduto** por **valorVendaProduto**

Adição

Veja a cláusula **SELECT** utilizando a operação de adição entre dois campos.

Exemplo 24:

SELECT (quantidadeProduto + valorVendaProduto) AS Valor_Soma FROM Produtos

Valor_Soma
305.50
108.50
806.50
102.10
602.70
201.50
504.50

Tabela 25 – Somando a coluna **quantidadeProduto** com **valorVendaProduto**

Após esses exemplos de operadores aritméticos, saiba que também é possível aplicar porcentagem em um determinado campo. Para isso, selecione todos os produtos e aplique um aumento de 20% sobre o valor de venda do produto.

Exemplo 25:

SELECT (valorVendaProduto * 1.2) AS Valor_Reajustado FROM Produtos

Valor_Reajustado
6.60
10.20
7.80
2.52
3.24
1.80
5.40

Tabela 26 – Operação de reajuste

Semelhantemente aos cálculos aritméticos básicos, os operadores aritméticos no MySQL também têm a mesma precedência de operador. Se a expressão aritmética contiver mais de um operador, os operadores de multiplicação e divisão terão a prioridade mais alta e serão avaliados primeiro e, em seguida, os operadores de adição e subtração serão avaliados.

Quando dois operadores têm a mesma prioridade, a expressão será avaliada da esquerda para a direita no MySQL. Os parênteses no MySQL também podem ser usados para forçar uma operação a ter prioridade sobre quaisquer outros operadores. Os parênteses também são usados para melhorar a legibilidade do código.

Ordenação de resultados com ORDER BY

A ordenação dos resultados de uma pesquisa é possível utilizando a cláusula **ORDER BY**, tanto em ordem crescente quanto em ordem decrescente.

A sintaxe para esse caso é:

SELECT * FROM tabela ORDER BY campo DESC

Para isso, utiliza-se, logo após o nome do campo, **ASC** para ordem ascendente (padrão) ou **DESC**, para ordem decrescente. Veja como fazer uma ordenação ascendente em um campo.

Exemplo 26:

SELECT * FROM Produtos ORDER BY nomeProduto

idProduto	nomeProduto	categoriaProduto	valorVendaProduto	quantidadeProduto
2	Arroz 5kg	Mercearia	8.50	100
6	Bolacha recheada	NULL	1.50	200
4	Detergente	Limpeza	2.10	100
3	Feijão	Mercearia	6.50	800
5	Leite	Laticínios	2.70	600
7	Leite condensado	Mercearia	4.50	500
1	Refrigerante	Bebidas	5.50	300

Tabela 27 – Lista de todos os produtos em ordem alfabética na coluna de nome do produto

Para classificar produtos em ordem decrescente, basta adicionar **DESC** após o nome da coluna **nomeProduto**.

Exemplo 27:

SELECT * FROM Produtos ORDER BY nomeProduto DESC

idProduto	nomeProduto	categoriaProduto	valorVendaProduto	quantidadeProduto
1	Refrigerante	Bebidas	5.50	300
7	Leite condensado	Mercearia	4.50	500
5	Leite	Laticínios	2.70	600
3	Feijão	Mercearia	6.50	800
4	Detergente	Limpeza	2.10	100
6	Bolacha recheada	NULL	1.50	200
2	Arroz 5kg	Mercearia	8.50	100

Tabela 28 – Lista de todos os produtos em ordem decrescente na coluna de nome do produto

É possível classificar os resultados por mais de um campo, neste caso, use a vírgula para separá-los. Veja o exemplo a seguir:

Exemplo 28:

SELECT * FROM Produtos ORDER BY categoriaProduto, nomeProduto DESC

idProduto	nomeProduto	categoriaProduto	valorVendaProduto	quantidadeProduto
6	Bolacha recheada	NULL	1.50	200
1	Refrigerante	Bebidas	5.50	300
5	Leite	Laticínios	2.70	600
4	Detergente	Limpeza	2.10	100
7	Leite condensado	Mercearia	4.50	500
3	Feijão	Mercearia	6.50	800
2	Arroz 5kg	Mercearia	8.50	100

Tabela 29 – Lista de todos os produtos ordenados por categoria em ordem alfabética e por nome do produto em ordem decrescente

Note que, na tabela anterior, as categorias estão em ordem (NULL, no caso, é valor vazio, então vem primeiro, depois os itens bolacha, laticínios etc.). De acordo com essa ordenação, em valores repetidos de categoria, vai-se ordenar o nomeProduto de maneira decrescente. Observe que isso acontece com a categoria “mercearia”, que tem três produtos associados, os quais ficam ordenados de modo decrescente (leite, feijão arroz).

Elabore uma consulta que apresente o nome do produto, a categoria e o valor, exibindo os produtos que não sejam da categoria “mercearia” e cujo valor de venda seja superior a 2.00. Não esqueça de ordenar pelo nome do produto.

Funções internas básicas (COUNT, SUM, AVG, MIN, MAX)

Até este momento, você verificou as opções de consulta para listar dados. Nem sempre, porém, ao trabalhar com banco de dados, a necessidade será apenas listar os dados. Pode ser que você precise obter informações sobre esses dados.

Para casos como este, você dispõe de funções para interpretação e execução de cálculos com os dados. A sintaxe do SQL inclui funções para serem emitidas na cláusula **SELECT** para isso. Elas são conhecidas como **funções de agregação**: COUNT, SUM, AVG, MIN e MAX.

COUNT

É uma função que conta e retorna o número de linhas que correspondem a um determinado critério. Caso queira saber quantos de seus produtos tem o nome “leite”, você pode fazer esta consulta:

Exemplo 29:

SELECT COUNT(nomeProduto) FROM Produtos WHERE nomeProduto LIKE '%leite%'

COUNT(nomeProduto)
2

Tabela 30 – Obtendo a quantidade de produtos com o nome "leite"

AVG

É uma função que retorna o valor médio (média) de uma coluna. Utilizando a tabela de produtos como exemplo, você pode encontrar o valor médio dos produtos com esta função:

Exemplo 30:

SELECT AVG(valorVendaProduto) FROM Produtos

AVG(valorVendaProduto)
4.471429

Tabela 31 – Obtendo o valor médio entre todos os produtos

SUM

É usado para encontrar a soma total de uma determinada coluna. Por exemplo, se você quiser ver a quantidade de produtos do supermercado, pode executar esta consulta:

Exemplo 31:

SELECT SUM(quantidadeProduto) FROM Produtos

SUM(quantidadeProduto)
2600

Tabela 32 – Obtendo a soma de todas as quantidades dos produtos

Observe que as funções **AVG** e **SUM** só funcionarão corretamente se utilizadas com dados numéricos. Se você tentar usá-los em dados não numéricos, isso resultará em um erro ou apenas 0.

MIN

É usado para encontrar o menor valor em uma coluna especificada. Você pode usar essa consulta para ver qual é o valor do produto mais barato.

Exemplo 32:

SELECT MIN(valorVendaProduto) FROM Produtos

MIN(valorVendaProduto)
1.50

Tabela 33 – Obtendo o valor do produto de menor valor

MAX

É usado para encontrar o maior valor numérico em uma determinada coluna. Você pode usar essa consulta para obter o valor do produto mais caro.

Exemplo 33:

SELECT MAX(valorVendaProduto) FROM Produtos

MAX(valorVendaProduto)
8.50

Tabela 34 – Obtendo o valor do produto de maior valor

As funções **MIN** e **MAX** podem ser usadas para tipos de dados numéricos e alfabéticos (diferentemente das funções **SUM** e **AVG**, que só aceitam números).

Quando se executa a função **MIN** em uma coluna que contém valores de *string*, essa função retornará o primeiro valor em ordem alfabética:

Exemplo 34:

SELECT MIN(nomeProduto) FROM Produtos

MIN(nomeProduto)
Arroz 5kg

Tabela 35 – Utilizando a função **MIN** em um campo *string* (retorno do primeiro registro em ordem alfabética)

Da mesma forma, quando executada em uma coluna que contém valores de *string*, a função **MAX** mostrará o último valor em ordem alfabética:

Exemplo 35:

SELECT MAX(nomeProduto) FROM Produtos

MAX(nomeProduto)
Refrigerante

Tabela 36 – Utilizando a função **MAX** em um campo *string* (retorno do último registro em ordem alfabética)

As funções agregadas têm muitos usos, além dos que foram descritos nesta seção. Elas são particularmente úteis quando usadas com a cláusula **GROUP BY**, abordada no conteúdo **Consultas com múltiplas tabelas**, desta unidade curricular, e com várias outras cláusulas de consulta que afetam como os conjuntos de resultados são classificados.

O dono do estabelecimento precisa saber quanto em dinheiro consta em seu estoque, afinal, foi realizado um investimento para compras das mercadorias. Para isso, utilizando as funções internas básicas estudadas, implemente agora uma consulta que exiba o valor em estoque de todos os produtos por meio da multiplicação da quantidade pelo valor dos produtos.

Subconsulta em SQL (*subquery*)

Uma *subquery*, também conhecida como subconsulta ou *subselect*, é uma instrução do tipo **SELECT** aninhada dentro de outro comando SQL.

Uma subconsulta deve ser delimitada entre parênteses e é avaliada apenas uma vez. O resultado de uma subconsulta retorna um conjunto de linhas para a consulta principal, ou seja, a consulta mais externa depende da subconsulta.

Existem algumas formas de utilizar subconsultas e, neste material, você estudará os seguintes meios:

- Subconsulta como filtro de uma consulta (utilizando operadores de comparação e **IN**)
- Subconsulta com uma nova coluna de consulta **(SELECT (...) AS Novo_Campo**)

Para continuar, será necessária uma nova tabela, **Vendas**, que pode ser criada com o seguinte *script*:

```
CREATE TABLE Vendas (
IdVenda int auto_increment,
IdProduto int,
quantidadeVendida int,
valorVendido double(9,2),
dataVenda date,
PRIMARY KEY (IdVenda)
);
```

Serão incluídos os seguintes dados:

```
INSERT into Vendas
(IdProduto,quantidadeVendida,
valorVendido,dataVenda)
values
(2,10,8.5,'2022-12-31'),
(2,15,8.5,'2022-01-01'),
(1,3,5.50,'2022-01-15');
```

Exemplo 36:

```
SELECT * FROM Vendas
```

IdVenda	idProduto	quantidadeVendida	valorVendido	dataVenda
1	2	10	8.50	2022-12-31
2	2	15	8.50	2022-01-01
3	1	3	5.50	2022-01-15

Tabela 37 – Listagem de todas as vendas

Veja a seguir as duas formas de utilizar subconsultas:

Subconsulta como filtro de uma consulta

Uma subconsulta de valor único retorna **apenas um valor** e pode ser usada no lugar de qualquer expressão utilizando operadores (=, <, >, <>).

Suponha que você precise obter da tabela de produtos todos os dados do(s) produto(s) de maior valor:

Exemplo 37:

SELECT * FROM Produtos WHERE valorVendaProduto = (SELECT MAX(valorVendaProduto) FROM Produtos)

idProduto	nomeProduto	categoriaProduto	valorVendaProduto	quantidadeProduto
2	Arroz 5kg	Mercearia	8.50	100

Tabela 38 – Obtendo o produto de maior valor

Obtenha os dados de todos os produtos sem mostrar o produto de maior valor:

Exemplo 38:

SELECT * FROM Produtos WHERE valorVendaProduto <> (Select Max(valorVendaProduto) FROM Produtos)

idProduto	nomeProduto	categoriaProduto	valorVendaProduto	quantidadeProduto
1	Refrigerante	Bebidas	5.50	300
3	Feijão	Mercearia	6.50	800
4	Detergente	Limpeza	2.10	100
5	Leite	Laticínios	2.70	600
6	Bolacha recheada	NULL	1.50	200
7	Leite condensado	Mercearia	4.50	500

Tabela 39 – Listagem de todos os produtos, sem mostrar o produto de maior de valor

Entretanto, para que uma subconsulta retorne múltiplas linhas, ela somente pode ser usada em uma **WHERE**, utilizando cláusulas especiais, como o operador **IN**.

Considere o seguinte exemplo a seguir para exibir o nome de todos os produtos que foram vendidos.

Exemplo 39:

SELECT nomeProduto FROM Produtos WHERE IdProduto IN (SELECT IdProduto FROM Vendas)

nomeProduto
Arroz 5kg
Refrigerante

Tabela 40 – Produtos que tiveram alguma venda

Subconsulta como uma nova coluna da consulta

Uma das formas possíveis de realizar uma subconsulta é fazendo com que o resultado de outra consulta seja uma coluna dentro da sua consulta principal.

Neste exemplo, serão listados o nome e o total vendido de produtos, realizando assim uma consulta principal na tabela de produtos e uma subconsulta na tabela de vendas, o que gerará uma nova coluna, cujo nome será **Total_Vendido**.

Exemplo 40:

```
SELECT P.nomeProduto,
(select sum(V.quantidadeVendida * V.valorVendido)
FROM Vendas as V WHERE V.idProduto=P.idProduto)
as Total_Vendido
FROM Produtos as P
```

nomeProduto	Total_Vendido
Refrigerante	16.50
Arroz 5kg	212.50
Feijão	NULL
Detergente	NULL
Leite	NULL
Bolacha recheada	NULL
Leite condensado	NULL

Tabela 41 – Listagem dos produtos e o total vendido

Subconsulta como filtro de uma consulta

Uma subconsulta de valor único retorna **apenas um valor** e pode ser usada no lugar de qualquer expressão utilizando operadores (=, <, >, <>).

Suponha que você precise obter da tabela de produtos todos os dados do(s) produto(s) de maior valor:

Exemplo 37:

SELECT * FROM Produtos WHERE valorVendaProduto = (SELECT MAX(valorVendaProduto) FROM Produtos)

idProduto	nomeProduto	categoriaProduto	valorVendaProduto	quantidadeProduto
2	Arroz 5kg	Mercearia	8.50	100

Tabela 38 – Obtendo o produto de maior valor

Obtenha os dados de todos os produtos sem mostrar o produto de maior valor:

Exemplo 38:

SELECT * FROM Produtos WHERE valorVendaProduto <> (Select Max(valorVendaProduto) FROM Produtos)

idProduto	nomeProduto	categoriaProduto	valorVendaProduto	quantidadeProduto
1	Refrigerante	Bebidas	5.50	300
3	Feijão	Mercearia	6.50	800
4	Detergente	Limpeza	2.10	100
5	Leite	Laticínios	2.70	600
6	Bolacha recheada	NULL	1.50	200
7	Leite condensado	Mercearia	4.50	500

Tabela 39 – Listagem de todos os produtos, sem mostrar o produto de maior de valor

Entretanto, para que uma subconsulta retorne múltiplas linhas, ela somente pode ser usada

em uma **WHERE**, utilizando cláusulas especiais, como o operador **IN**.

Considere o seguinte exemplo a seguir para exibir o nome de todos os produtos que foram vendidos.

Exemplo 39:

SELECT nomeProduto FROM Produtos WHERE IdProduto IN (SELECT IdProduto FROM Vendas)

nomeProduto
Arroz 5kg
Refrigerante

Tabela 40 – Produtos que tiveram alguma venda

Subconsulta como uma nova coluna da consulta

Uma das formas possíveis de realizar uma subconsulta é fazendo com que o resultado de outra consulta seja uma coluna dentro da sua consulta principal.

Neste exemplo, serão listados o nome e o total vendido de produtos, realizando assim uma consulta principal na tabela de produtos e uma subconsulta na tabela de vendas, o que gerará uma nova coluna, cujo nome será **Total_Vendido**.

Exemplo 40:

```
SELECT P.nomeProduto,
(select sum(V.quantidadeVendida * V.valorVendido)
FROM Vendas as V WHERE V.idProduto=P.idProduto)
as Total_Vendido
FROM Produtos as P
```

nomeProduto	Total_Vendido
Refrigerante	16.50
Arroz 5kg	212.50
Feijão	NULL
Detergente	NULL
Leite	NULL
Bolacha recheada	NULL
Leite condensado	NULL

Tabela 41 – Listagem dos produtos e o total vendido

Revisando e aplicando

No vídeo a seguir, você poderá expandir seu conhecimento e sua experiência com novos exemplos de *scripts* SQL de consultas. Acompanhe os casos de estudo passo a passo, pause ou retroceda sempre que necessário.

Encerramento

Este material apresentou os principais operadores em SQL de que você precisará para realizar seleção e filtragem em consultas e subconsultas. Também foram estudados os operadores lógicos e aritméticos, a realização de cálculos em SQL e as funções de agregação de dados. Não esqueça de praticar os exemplos aqui apresentados. É fundamental praticar em seu ambiente de estudo.

Em outro conteúdo, haverá uma abordagem mais profunda em consultas com múltiplas tabelas em operações de união, intersecção, junção e agrupamento.