



[www.devmedia.com.br](http://www.devmedia.com.br)

[versão para impressão]

Link original: <https://www.devmedia.com.br/sql-funcoes-de-agregacao/38463>

## SQL: Funções de agregação

**Nesta documentação você aprenderá a utilizar as funções de agregação da linguagem SQL, que são aquelas que agregam um conjunto de valores em um único resultado.**

SQL: Funções de agregação



É comum que uma aplicação necessite de informações resumidas. Obter a menor/maior venda do dia é apenas um exemplo dessa situação. A linguagem SQL contém funções nativas para esse fim, que podem ser usadas para agregar um conjunto de valores em um único resultado.

Neste documento apresentaremos as **principais funções de agregação da linguagem SQL**.

### Tópicos

---

[Sintaxe](#)

[Tabelas de exemplo](#)

[MAX](#)

[MIN](#)

[SUM](#)

[AVG](#)

[COUNT](#)

[GROUP BY](#)

[HAVING](#)

[ALIAS](#)

[Exemplo prático](#)

## Sintaxe

Uma função de agregação processa um conjunto de valores contidos em uma única coluna de uma tabela e retorna um único valor como resultado. Sua sintaxe é semelhante aquela encontrada em muitas linguagens de programação. Contudo, o parâmetro informado é sempre a coluna cujos valores desejamos processar.

Vemos a seguir um exemplo da sintaxe dessa cláusula.

```
nome-da-funcao (coluna)
```

Podemos informar na listagem de colunas do comando *SELECT* uma ou mais funções de agregação, de acordo com a necessidade:

```
SELECT
    [função(ões) de agregação/coluna(s)]
FROM
    [tabela(s)]
```

## Tabelas de exemplo

Para acompanhar os exemplos apresentados nessa documentação, considere a **Tabela 1**:

	Codigo	Descricao	PrecoVenda	PrecoCusto	Categoria
	1	Caderno	5.45	1.00	1
	2	Caneta	1.20	0.50	1
	3	CD	1.00	0.10	2
	4	Mouse	14.00	5.00	2

**Tabela 1.** Tabela produtos

## MAX

A função *MAX* analisa um conjunto de valores e retorna o maior entre eles. No exemplo abaixo utilizamos essa função para encontrar o preço de venda mais alto:

```
SELECT
```

```
max (precovenda)
FROM
produtos
```

Ao final da execução desse comando será retornado o valor 14.00, como mostra a **Figura 1**:

	max(precovenda)
	14.00

**Figura 1.** Resultado com a função MAX

## MIN

---

*MIN* analisa um grupo de valores e retorna o menor entre eles. A seguir utilizamos essa função para conhecer o preço de venda mais baixo:

```
SELECT
min (precovenda)
FROM
produtos
```

Como retorno para esse comando, obteremos o valor 1.00 conforme mostra a **Figura 2**:

	min(precovenda)
	1.00

**Figura 2.** Resultado com a função MIN

## SUM

---

A função *SUM* realiza a soma dos valores em uma única coluna e retorna esse resultado. Para somar todos os preços de venda dos produtos de uma categoria, podemos utilizar essa função informando a coluna PrecoVenda, como exemplificado a seguir:

```
SELECT
sum (precovenda)
```

```
FROM
  produtos
WHERE
  categoria = 1
```

Ao final da execução desse comando, teremos o valor 6.65, como mostra a **Figura 3**.

	sum(precovenda)
	6.65

**Figura 3.** Resultado com a função SUM

*Nota:* Por padrão, a função SUM ignora valores do tipo null.

## AVG

Com a função *AVG* podemos calcular a média aritmética dos valores em uma única coluna. Usamos essa função no exemplo a seguir, tomando como parâmetro a coluna *PrecoVenda* da tabela *produtos*.

```
SELECT
  avg(precovenda)
FROM
  produtos
```

Ao final da execução desse comando teremos o valor aproximado 5.41, com mostra a **Figura 4**.

	avg(precovenda)
	5.412500

**Figura 4.** Resultado com a função AVG

*Nota:* Por padrão a função AVG ignora valores do tipo null.

## COUNT

A função *COUNT* retorna o total de linhas selecionadas. Ela pode receber por parâmetro o nome da coluna ou um asterisco. Por padrão, quando informado o nome de uma coluna,

valores do tipo *null* são ignorados, mas quando informado \* todas as linhas serão contabilizadas.

Para sabermos o total de produtos em uma categoria, podemos escrever uma consulta como esta:

```
SELECT
    count (precovenda)
FROM
    produtos
WHERE
    categoria = 1
```

## GROUP BY

Ao utilizar a cláusula **GROUP BY** dividimos os registros que serão agregados em **grupos de valores**. Essa mudança faz com que tenhamos mais de uma linha como resultado, pois o processamento será realizado uma vez sobre cada um desses grupos.

Para sabermos o produto com maior valor de venda de cada categoria, podemos escrever uma consulta como esta:

```
SELECT
    categoria,
    max (precovenda)
FROM
    produtos
GROUP BY categoria
```

Considerando a tabela de produtos, quando essa consulta for executada os dados serão divididos em dois grupos, um para cada categoria, como podemos observar nas **Figuras 5 e 6**.

	PrecoVenda	Categoria
	5.45	1
	1.20	1

**Figura 5.** Resultado com a função GROUP BY da categoria 1

	PrecoVenda	Categoria
	1.00	2
	14.00	2

**Figura 6.** Resultado com a função GROUP BY da categoria 2

Logo após, a função *MAX* será aplicada uma vez para cada um desses grupos, fazendo com que tenhamos como resultado os seguintes valores apresentados na **Figura 7**:

	categoria	max(preco venda)
	1	5.45
	2	14.00

**Figura 7.** Resultado com a função MAX nas categorias

Note que o primeiro registro se refere aos produtos com categoria 1, sendo o segundo o valor máximo dentre os preços de venda dos produtos de categoria 2.

## HAVING

Podemos usar a cláusula *HAVING* em conjunto com *GROUP BY* para filtrar os resultado que serão submetidos a agregação.

Vemos um exemplo desse filtro na consulta abaixo, que lista o maior preço de venda de cada categoria, incluindo apenas os produtos com preço de venda maior que 10:

```
SELECT
    categoria,
    max(preco venda)
FROM
    produtos
GROUP BY categoria
HAVING max(preco venda) > 10
```

Ao final dessa consulta obtemos os valores apresentados na **Figura 8**.

	categoria	max(preco venda)
	2	14.00

**Figura 8.** Resultado com a função HAVING

## ALIAS

A fim de facilitar a compreensão do SQL, podemos utilizar a palavra-chave *as* para criar um apelido para uma coluna. Uma vez que as funções de agregação são retornadas como tal, também podemos fazer uso desse recurso como exemplificado a seguir:

```
SELECT
    categoria,
    max(preco_venda) as maximo_preco_venda
FROM
    produtos
GROUP BY categoria
HAVING max(preco_venda) > 10
```

Ao final da execução desse código teremos a coluna *maximo\_preco\_venda* contendo o maior preço de venda, como mostra a **Figura 9**.

	categoria	maximo_preco_venda
	2	14.00

**Figura 9.** Resultado com a função ALIAS

## Exemplo prático

No exemplo a seguir listamos as categorias que contêm produtos cujas médias dos preços de custo exceda 2. Para isso agrupamos as categorias com a cláusula *GROUP BY* e filtramos os totais maiores que 2 com *HAVING*:

```
SELECT
    categoria,
    avg(preco_custo) as media_preco_custo
FROM
    produtos
GROUP BY categoria
HAVING avg(preco_custo) > 2
```

Uma vez que apenas a categoria 2 possui registro que atendem ao filtro estabelecido, receberemos um único registro como resposta, que pode ser observado na **Figura 10**.

	categoria	media_preco_custo
	2	2.550000

**Figura 10.** Resultado do exemplo prático