Sumário

FUNÇÃO DE JANELA1

PARTITION BY1

OVER2

ROWNUMBER3

ACÚMULO AO LONGO DO TEMPO4

ACÚMULO AO LONGO DO TEMPO POR PARTICIONAMENTO5

CALCULANDO ESTATÍSTICA – MÉDIA MÓVEL6

CALCULANDO ESTATÍSTICA – MÉDIA MÓVEL COM PARTICIONAMENTO6

# FUNÇÃO DE JANELA

As funções de janela, também conhecidas como funções de janela analítica ou "window functions" em inglês, são uma poderosa ferramenta disponível em bancos de dados relacionais como o SQL Server. Elas permitem executar cálculos agregados ou analíticos em um conjunto de linhas relacionadas em uma consulta, sem agrupar os dados ou criar subconsultas.

As funções de janela são usadas em combinação com a cláusula OVER para definir o conjunto de linhas ao qual a função será aplicada. A cláusula OVER especifica como as linhas devem ser particionadas e ordenadas antes de aplicar a função de janela.

As funções de janela são muito úteis para calcular classificações, agregações e comparações dentro de um conjunto de dados sem a necessidade de criar subconsultas complexas ou fazer várias junções de tabelas. Isso torna as consultas mais eficientes e legíveis.

## PARTITION BY

A função PARTITION BY é uma cláusula frequentemente utilizada em SQL em conjunto com funções analíticas, como SUM, AVG, ROW\_NUMBER, entre outras. Essa cláusula permite a divisão do conjunto de resultados em partições lógicas com base nos valores de uma ou mais colunas. Posteriormente, as funções analíticas são aplicadas a cada partição separadamente.

Quando a cláusula PARTITION BY é utilizada, a função analítica é calculada em grupos de linhas que compartilham o mesmo valor ou conjunto de valores nas colunas especificadas na cláusula.

A sintaxe geral é a seguinte:

<função\_analítica> OVER (PARTITION BY coluna1, coluna2, ...)

Essa cláusula é composta pelos seguintes elementos:

<função\_analítica>: Refere-se à função analítica que está sendo aplicada, como SUM, AVG, ROW\_NUMBER, etc.

OVER: Esta cláusula define o escopo da função analítica.

PARTITION BY: Indica as colunas pelas quais o conjunto de resultados será particionado.

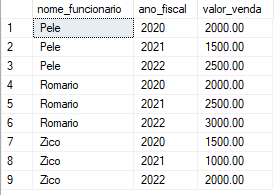
Por exemplo, utilizando a nossa base de dados exemplo, caso utilizemos o comando abaixo, teremos o seguinte resultado:

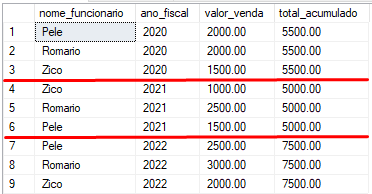
SELECT nome\_funcionario,

ano\_fiscal,valor\_venda,

SUM(valor\_venda) OVER (PARTITION BY ano\_fiscal) as total\_acumulado

FROM WindowFunction.Vendas





SUM(valor\_venda) OVER (PARTITION BY ano\_fiscal) calcula a soma acumulada das vendas para cada ano fiscal. A cláusula (PARTITION BY ano\_fiscal) divide o conjunto de resultados em partições separadas para cada ano fiscal, de modo que a soma acumulada seja calculada separadamente para cada ano, como temos 3 anos, 2020, 2021, 2022, o resultado é dividido em três partições.

## OVER

Quando utilizada sem argumentos, a função OVER() em SQL é uma cláusula que define o escopo da função analítica para todo o conjunto de resultados, sem agrupamento ou partição. Isso significa que a função será aplicada a todas as linhas do conjunto de resultados de forma global, sem distinção.

Aqui está uma explicação detalhada dos componentes:

OVER(): Esta é a cláusula que define o escopo da função analítica. Ao usar OVER() sem argumentos, estamos especificando que a função deve ser aplicada a todas as linhas do conjunto de resultados, sem distinção ou agrupamento.

A função analítica utilizada em conjunto com OVER() pode ser qualquer função analítica válida, como SUM, AVG, ROW\_NUMBER, entre outras. Quando usada sem argumentos, ela calcula o resultado para todas as linhas do conjunto de resultados.

Por exemplo, utilizando a nossa base de dados exemplo, caso utilizemos o comando abaixo, teremos o seguinte resultado:

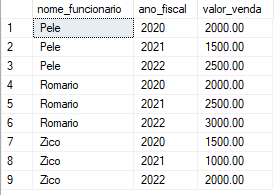
SELECT nome\_funcionario,

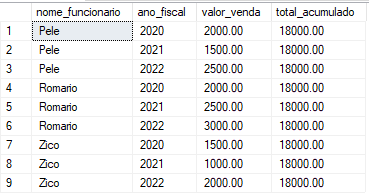
ano\_fiscal,

valor\_venda,

SUM(valor\_venda) OVER () as total\_acumulado

FROM WindowFunction.Vendas





Podemos notar que o resultado se repetiu por toda a coluna, isso ocorreu, pois não foi especificado nenhuma partição dentro da cláusula over(), dessa forma, o valor foi o mesmo para todas as linhas.

## ROWNUMBER

A função ROW\_NUMBER() atribui um número sequencial para cada linha de um conjunto de resultados, de acordo com a ordem especificada. Esta função é frequentemente utilizada em consultas SQL para classificar e paginar resultados, semelhante a um ranking.

A sintaxe básica da função ROW\_NUMBER é a seguinte:

ROW\_NUMBER() OVER (ORDER BY coluna1, coluna2, ...)

Aqui está uma explicação dos principais componentes:

ROW\_NUMBER(): A função em si que atribui um número de linha a cada registro.

OVER: A cláusula que define a janela analítica ou a parte do conjunto de resultados sobre a qual a função é aplicada.

(ORDER BY coluna1, coluna2, ...): A parte que especifica a ordem pela qual as linhas devem ser numeradas.

Podemos ordenar por uma ou mais colunas.

Aqui está um exemplo simples de como usar ROW\_NUMBER:

SELECT

ROW\_NUMBER() OVER (ORDER BY Nome) AS NumeroLinha,

Nome

FROM

ExemploTabela;

Neste exemplo, a função ROW\_NUMBER atribuirá um número sequencial às linhas da tabela ExemploTabela com base na ordem crescente dos valores na coluna Nome. Isso pode ser útil em situações em precisamos referenciar ou classificar as linhas de acordo com a ordem de uma coluna específica.

É importante observar que a função ROW\_NUMBER gera números sequenciais, mas não necessariamente únicos, mesmo que a cláusula ORDER BY inclua colunas com valores exclusivos.

## ACÚMULO AO LONGO DO TEMPO

Para realizar um acumulado ao longo do tempo em SQL, pode-se utilizar a função de janela SUM() em conjunto com a cláusula OVER() e especificar a ordem pela qual as linhas devem ser somadas. Essa abordagem é útil quando é necessário calcular uma soma acumulada de valores ao longo de uma sequência de datas, por exemplo.

Supondo que haja uma tabela de vendas denominada Vendas com as colunas DataVenda e ValorVenda, e o objetivo seja calcular o acumulado dos valores de vendas ao longo do tempo. Aqui está um exemplo de consulta SQL que realiza essa tarefa:

SELECT

DataVenda,

ValorVenda,

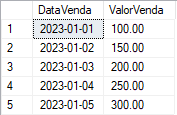
SUM(ValorVenda) OVER (ORDER BY DataVenda) AS Acumulado

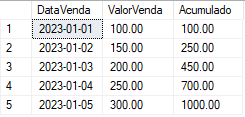
FROM

Vendas

Neste exemplo, a função SUM(ValorVenda) calcula a soma acumulada dos valores de venda. A cláusula OVER (ORDER BY DataVenda) especifica que a soma acumulada será calculada para cada linha do conjunto de resultados, com base na ordem das datas de venda (DataVenda).

Isso resultará em uma tabela que mostra a data da venda, o valor da venda e o acumulado dos valores de venda até aquela data. Cada linha terá o acumulado de todas as vendas que ocorreram até a data especificada naquela linha. Essa abordagem é útil para análises de tendências e acompanhamento do desempenho ao longo do tempo.



****

Isso resultará em uma tabela que mostra a data da venda, o valor da venda e o acumulado dos valores de venda até aquela data. Cada linha terá o acumulado de todas as vendas que ocorreram até a data especificada naquela linha. Essa abordagem é útil para análises de tendências e acompanhamento do desempenho ao longo do tempo.

## ACÚMULO AO LONGO DO TEMPO COM PARTICIONAMENTO

Para fazer um acumulado ao longo do tempo com particionamento em SQL, pode-se utilizar a função de janela SUM() em conjunto com a cláusula OVER() e especificar a ordem pela qual as linhas devem ser somadas, assim como a coluna pela qual deseja-se particionar os dados.

Ainda utilizando a base de dados anterior, agora o objetivo é calcular o acumulado dos valores de vendas ao longo do tempo, particionado por ano, adicionamos novos registros para o ano de 2024. Aqui está um exemplo de consulta SQL que realiza essa operação:

SELECT

DataVenda,

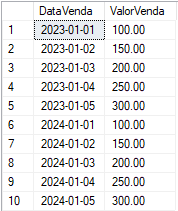
ValorVenda,

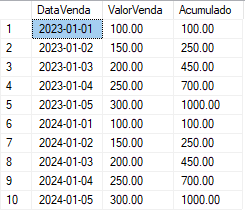
SUM(ValorVenda) OVER (PARTITION BY YEAR(DataVenda) ORDER BY DataVenda) AS Acumulado

FROM

Vendas

Neste exemplo, a função SUM(ValorVenda) calcula a soma acumulada dos valores de venda. A cláusula ”OVER (PARTITION BY YEAR(DataVenda) ORDER BY DataVenda)” especifica que a soma acumulada será calculada para cada linha do conjunto de resultados, particionada por ano (YEAR(DataVenda)) e ordenada pela data de venda (DataVenda).





Isso resultará em uma tabela que mostra a data da venda, o valor da venda e o acumulado dos valores de venda até aquela data, porém agora agrupado por ano. Cada linha terá o acumulado de todas as vendas que ocorreram até a data especificada naquela linha, mas separado por ano. Essa abordagem é útil para análises que exigem o acompanhamento do desempenho ao longo do tempo, considerando diferentes períodos de tempo separadamente.

## CALCULANDO ESTATÍSTICA – MÉDIA MÓVEL

No SQL Server, é possível calcular a média móvel com particionamento utilizando funções de janela (window functions) em conjunto com a cláusula OVER(). Isso permite segmentar os dados em grupos e calcular a média móvel dentro de cada grupo.

Supondo que temos uma tabela de vendas chamada Vendas, onde o objetivo é calcular a média móvel do valor das vendas. Podemos utilizar a instrução a seguir para verificar essa informação.

SELECT

DataVenda,

ValorVenda,

AVG(ValorVenda) OVER (PARTITION BY YEAR(DataVenda) ORDER BY DataVenda) AS MediaMovel

FROM

Vendas;

Neste exemplo, a média móvel do valor das vendas é calculada para cada ponto de dados na série temporal, considerando as vendas anteriores até a data atual. Isso permite visualizar como a média do valor das vendas evolui ao longo do tempo.

## CALCULANDO ESTATÍSTICA – MÉDIA MÓVEL COM PARTICIONAMENTO

Para calcular a média móvel com particionamento no SQL Server, utiliza-se funções de janela (window functions) em conjunto com a cláusula OVER(). Isso permite calcular a média móvel dentro de grupos específicos de dados, segmentados por uma determinada coluna.

Por exemplo, considerando a tabela anterior, é possível calcular a média móvel do valor das vendas para cada ano da seguinte forma:

SELECT

DataVenda,

ValorVenda,

AVG(ValorVenda) OVER (PARTITION BY YEAR(DataVenda) ORDER BY DataVenda) AS MediaMovel

FROM

Vendas;

Neste exemplo, a média móvel do valor das vendas é calculada para cada ponto de dados na série temporal, considerando as vendas anteriores até a data atual, dentro de cada ano. Isso permite visualizar como a média do valor das vendas evolui ao longo do tempo, mas agora segmentada por ano.