**FUNÇÕES NATIVAS**

Material de Apoio

Sumário

[1. Conceito 3](#_Toc79935814)

[2. Funções de Agregação 4](#_Toc79935815)

[3. FUNÇÕES ANALÍTICAS 13](#_Toc79935816)

[4. FUNÇÕES DE CLASSIFICAÇÃO 13](#_Toc79935817)

[5. FUNÇÕES DO CONJUNTO DE LINHAS 15](#_Toc79935818)

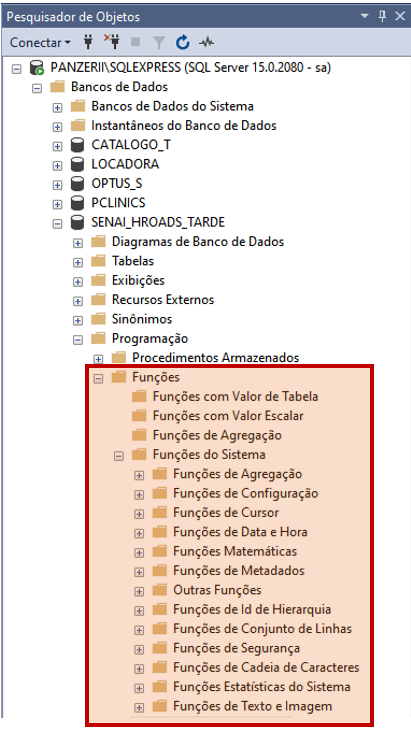
[6. FUNÇÕES ESCALARES 20](#_Toc79935819)

[Trello 22](#_Toc79935820)

# Conceito

Funções nativas, ou funções internas, como também são chamadas, são aquelas que já se encontram previamente disponíveis aos desenvolvedores para utilização no ambiente SQL. Elas já estão prontas e acessíveis desde a instalação, não necessitando serem desenvolvidas, como as functions que veremos adiante.

Assim como podemos utilizar diversas funções específicas ao carregar diferentes bibliotecas em C# (lembra do comando using, lá em cima do código, no VS Code?), no SQL Server temos várias funções prontas para o uso já presentes na estrutura de um banco de dados assim que ele é instanciado:



Em nosso projeto HROADS, já utilizamos algumas funções nativas, mesmo sem saber deste conceito. Vamos agora entender alguns conceitos importantes sobre os tipos de funções nativas e em quais categorias elas se agrupam/se subdividem. Também veremos alguns exemplos de situações em que podem ser úteis:

# Funções de Agregação

Funções agregadas executam um cálculo em um conjunto de valores de uma coluna e retornam um único valor, ou seja, ela faz uma agregação de dados através de contas com a finalidade de facilitar a análise, assim servindo para criar índices, medidas e indicadores a partir dos dados analisados.

Elas ignoram valores nulos e são muito usadas com a cláusula GROUP BY de uma instrução SELECT. As funções agregadas são usadas como expressão em uma lista de seleção como uma subconsulta ou cláusula HAVING. Todas as funções de agregação são funções determinísticas.

**SINTAXE**

nome-da-função(coluna)

Podemos utilizar as funções de agregação na listagem de colunas do comando SELECT de acordo com a finalidade, de acordo com o exemplo abaixo:

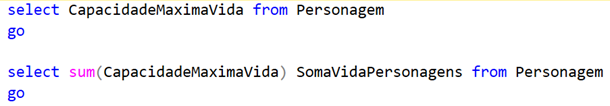
SELECT função-de-agregação(coluna) FROM tabela

**SUM**

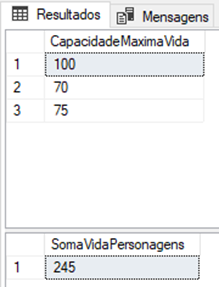
Realiza a **soma dos valores** de uma coluna e retorna o resultado obtido.



Demonstração no código:



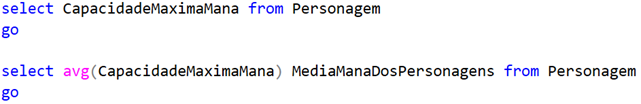
Retorno:



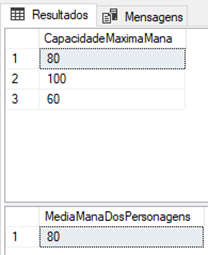
**AVG**

Uma função utilizada para retornar a **média aritmética dos valores** de uma coluna, dividindo a soma desses valores pela contagem de valores não nulos (ignora os valores nulos). Poderá ocorrer um erro caso a soma ultrapassar o valor máximo do tipo de dado retornado.

Demonstração no código:



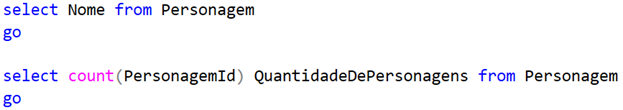
Retorno:



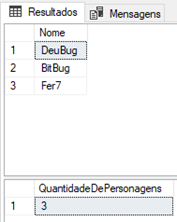
**COUNT**

Retorna o total de linhas selecionadas, ou seja, o **número de itens encontrados em um grupo**, sempre retorna um valor do tipo de dados int. Para retornar valores superiores a 2^31-1, COUNT retornará um erro. Nesses casos, use COUNT\_BIGem vez disso.

Demonstração no código:



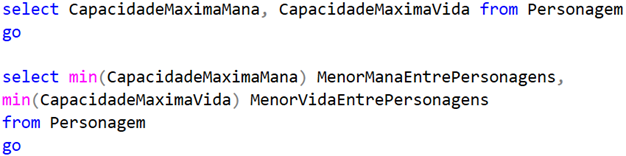
Retorno:



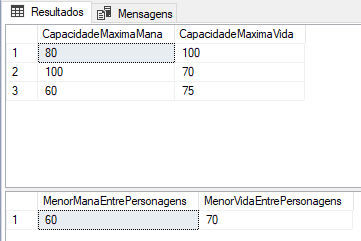
**MIN**

Retorna o **valor mínimo na expressão**, ou seja, analisa um grupo de valores e retorna o menor entre eles. MIN pode ser usado com colunas numeric, char, varchar, uniqueidentifier ou datetime. O retorno é igual ao tipo de valor analisado na expressão.

Demonstração no código:



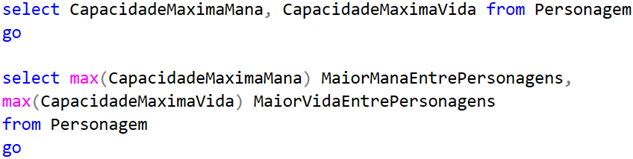
Retorno:



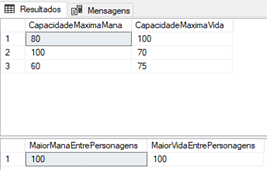
**MAX**

Retorna o **valor máximo na expressão**, ou seja, analisa um grupo de valores e retorna o maior entre eles. MAX pode ser usado com colunas numeric, character, uniqueidentifier e datetime, mas não com colunas bits. O retorno é igual ao tipo de valor analisado na expressão.

Demonstração no código:



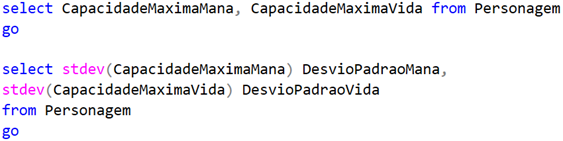
Retorno:



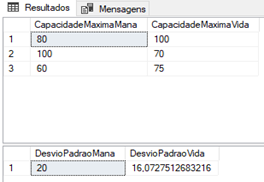
**STDEV**

Retorna o **desvio padrão estatístico de todos os valores do grupo especificado**. Retorna um valor do tipo float. STDEV pode ser usado exclusivamente com colunas numéricas.

Demonstração no código:



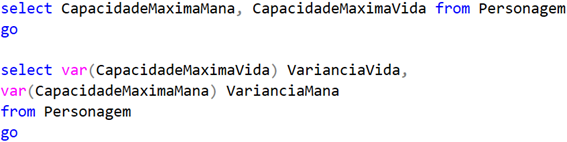
Retorno:



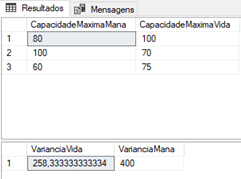
**VAR**

Retorna a **variância estatística** de todos os valores da expressão especificada. Retorna um valor do tipo float.  VAR pode ser usado exclusivamente com colunas numéricas.

Demonstração no código:



Retorno:

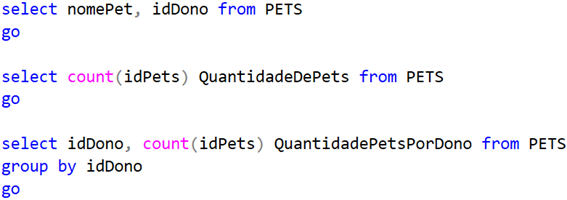


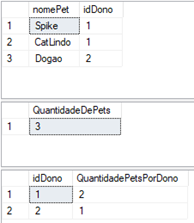
**Cláusulas**

**GROUP BY**

Na cláusula GROUP BY agrupa linhas baseado em semelhanças entre elas. Essa mudança faz com que tenhamos mais de uma linha como resultado, pois o processamento será realizado uma vez sobre cada um desses grupos.

Exemplo:



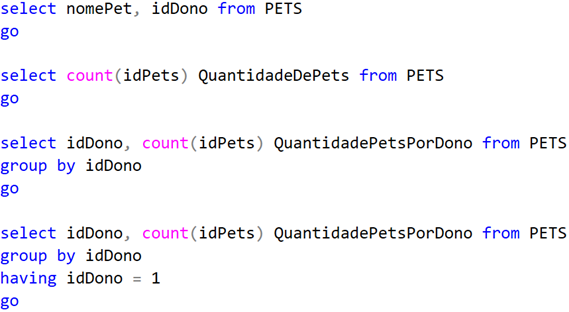


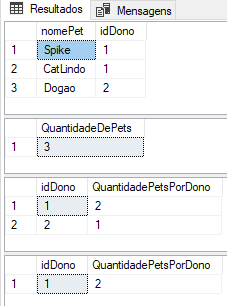
Pode se perceber que, na tabela QuantidadePetsPorDono, há a quantidade (COUNT) de pets para cada idDono, sendo assim há essa agregação pela semelhança do idDono presente nos pets.

**HAVING**

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

Exemplo:



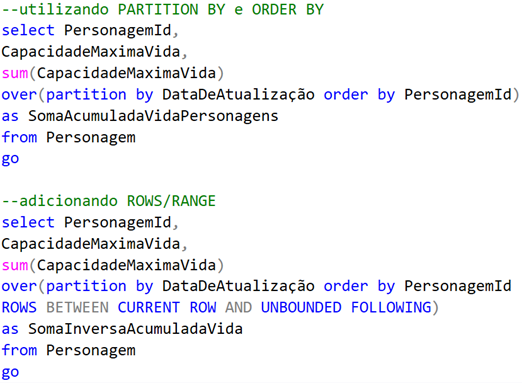


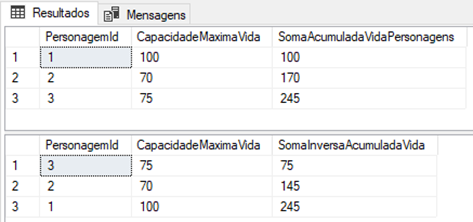
Neste exemplo, é feito o COUNT dos pets que possuem um idDono = 1, assim filtrando os dados pelo idDono.

**OVER**

Determina o **particionamento e a ordenação** de um conjunto de linhas antes da aplicação da função de janela associada. Utiliza se a cláusula OVER com funções para computar valores agregados como médias móveis, agregações cumulativas, somas acumuladas e outros. OVER possui alguns argumentos como PARTITION BY (divide o conjunto de resultados da consulta em partições), ORDER BY (define a ordem lógica das linhas dentro de cada partição do conjunto de resultados) e ROWS/RANGE (limita as linhas dentro da partição com a especificação de pontos iniciais e finais na partição, feito pela especificação de um intervalo de linhas em relação à linha atual. É necessário o ORDER BY para definir o ponto inicial).

Exemplo:

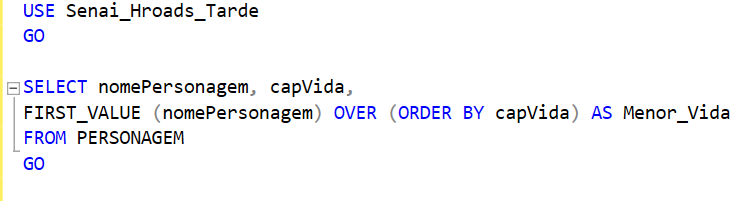


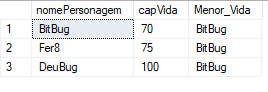


Neste exemplo, na primeira tabela é feito a soma da vida dos personagens e exibido de forma separada pelos PersonagemId (PARTITION BY) e acumulada, sendo a exibição ordenada pelos PersonagemId (ORDER BY). Na segunda tabela acontece o mesmo, porém a ordem da soma dos dados é inversa pois o ponto inicial e final fora limitado (ROWS/RANGE), somando então do personagem de ID = 3 até o ID = 1, obtendo o mesmo resultado da primeira tabela no final.

# FUNÇÕES ANALÍTICAS

A função de analítica calcula os valores agregados com base em um conjunto de linhas. As funções analíticas podem retornar várias linhas para cada grupo. Use a análise para calcular médias móveis, totais cumulativos, porcentagens ou os N principais resultados de um grupo. FIRST-VALUE Retorna o primeiro valor em um conjunto ordenado de valores.



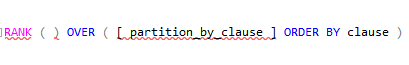


# FUNÇÕES DE CLASSIFICAÇÃO

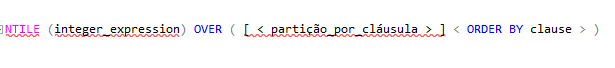
As funções de classificação retornam um valor de classificação para cada linha na partição. Dependendo da função usada, algumas linhas podem receber o mesmo valor que outras linhas. As funções de classificação são não determinísticas. A linguagem SQL fornece alguns comandos como:

RANK, NTILE, ROW\_NUMBER, DENSE\_RANK.

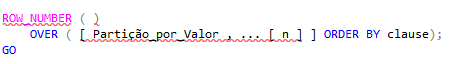
**RANK:** O RANK Retorna à classificação de cada linha na partição de um conjunto de resultados. A classificação de uma linha é um mais o número de classificações que vêm antes da linha em questão.



**NTILE:** O NTILEDistribui as linhas de uma partição ordenada em um número de grupos especificado. Os grupos são numerados, iniciando em um. Para cada linha, NTILE retorna o número do grupo ao qual a linha pertence.

integer\_expression:  
É uma expressão que especifica o número de grupos nos quais cada partição deve ser dividida.

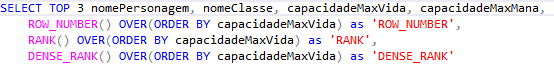
**ROW\_NUMBER:** ROW\_NUMBER numera todas as linhas em sequência (por exemplo 1, 2, 3, 4, 5). Retorna o número sequencial de uma linha em uma partição de um conjunto de resultados, começando em 1 na primeira linha de cada partição.

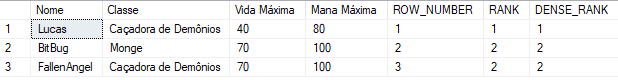


**DENSE\_RANK:** O DENSE\_RANKretorna à posição de cada linha dentro de uma partição do conjunto de resultados. A classificação de uma linha é um mais o número de valores de classificação.



**TOP:** A diretória **TOP** é utilizada para escolher a quantidade de registros que aparecerá no SELECT, escolhendo os primeiros registros, a partir do ORDER BY.





# FUNÇÕES DO CONJUNTO DE LINHAS

Um conjunto de linhas contém colunas de dados. Os conjuntos de linhas são objetos centrais que permitem que todos os provedores de dados OLE DB exponham dados de conjuntos de resultados em formato tabular.

**String\_Split**

Uma função com valor de tabela que divide uma cadeia de caracteres em linhas de subcadeias de caracteres com base em um caractere separador especificado.

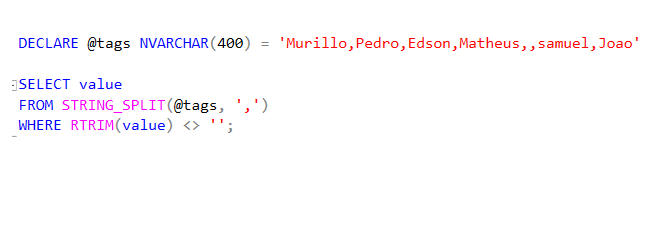
**Nível de compatibilidade 130**

Obs: STRING\_SPLIT requer que o nível de compatibilidade seja, no mínimo, 130. Quando o nível for inferior a 130, o SQL Server não conseguirá localizar a função STRING\_SPLIT.

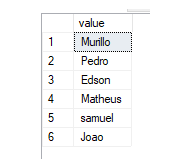
**Tipos de retorno**

Retorna uma tabela de coluna única cujas linhas são as subcadeias de caracteres. O nome da coluna é value. Retorna nvarchar se um dos argumentos de entrada são nvarchar ou nchar. Caso contrário, retorna varchar. O tamanho do tipo de retorno é o mesmo que o tamanho do argumento da cadeia de caracteres.

**Exemplo:**

****

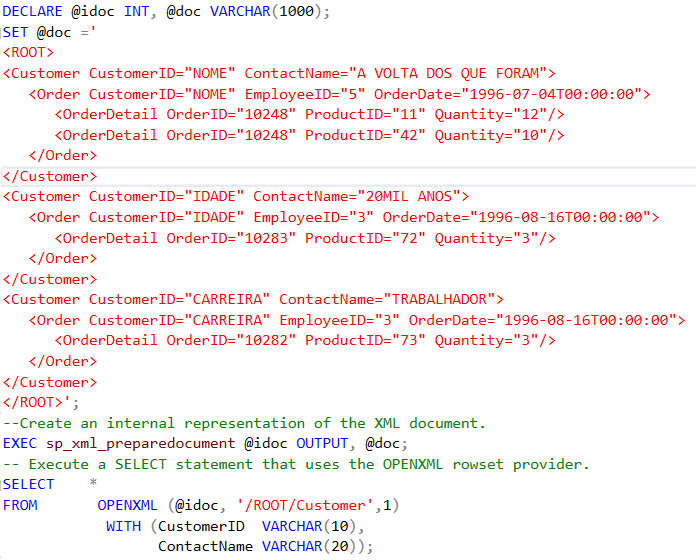
**Tabela:**

****

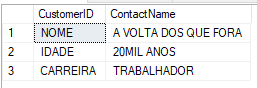
**Openxml**

OPENXML fornece uma exibição de conjunto de linhas em um documento XML. Como OPENXML é um provedor de conjunto de linhas, OPENROWSET pode ser usado em instruções Transact-SQL nas quais os provedores de conjunto de linhas, como uma tabela, exibição ou a função OPENROWSET, podem aparecer.

**EXEMPLO:**

****

**TABELA:**

****

**Containstable**

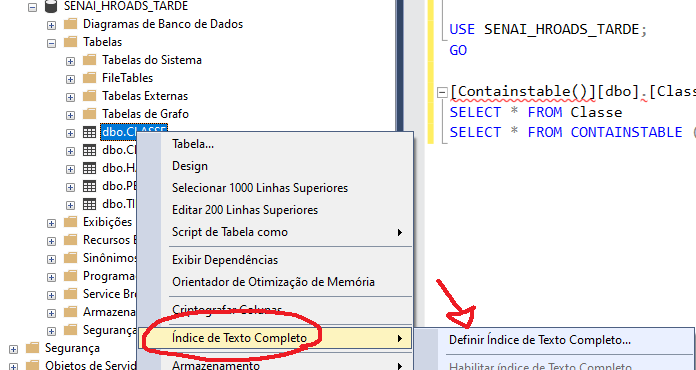
Retorna uma tabela de zero, uma ou mais linhas para as colunas que contêm correspondências exatas ou difusas (menos precisas) a palavras e frases únicas, a proximidade de palavras dentro de uma determinada distância uma da outra ou correspondências ponderadas

**EXEMPLO:**

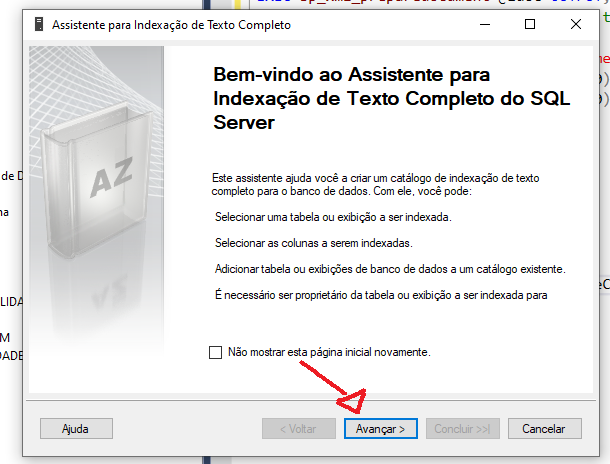
****

**ATIVANDO INDEX DA TABELA**

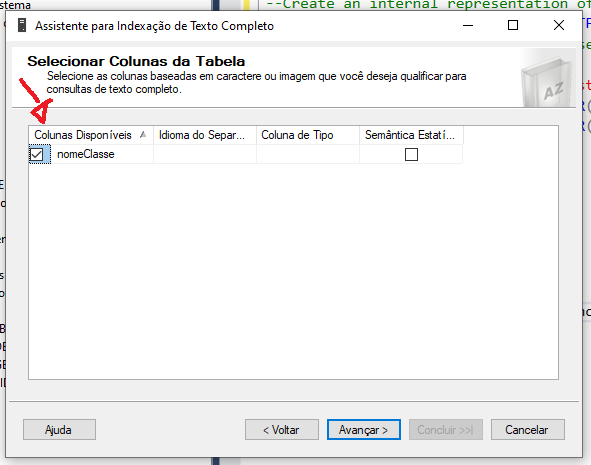
Clique com o direito em cima do nome de sua tabela, vá até Índice de Texto Completo, depois em Definir Índice:

****

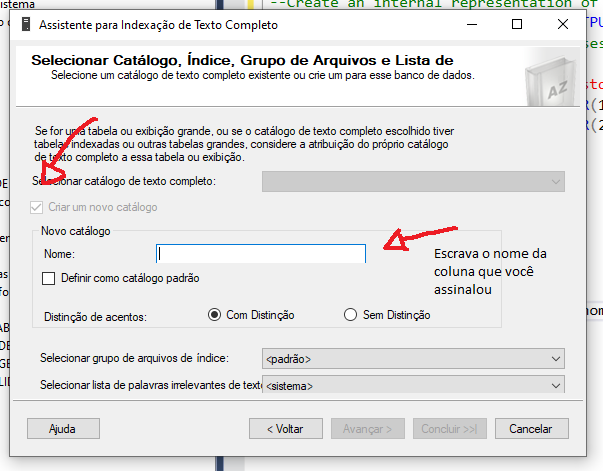
Após isso irá abrir uma aba . Clique em Avançar.

****

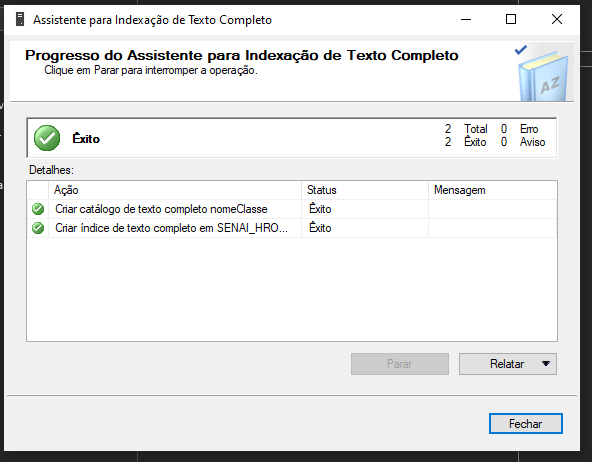
Clique em avançar ate checar nessa tela. Selecione a coluna que você deseja procurar.E após Clique em Avançar.



Clique em avançar até chegar nessa tela. Faça oque se pede. Após isso clique em Concluir.

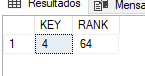


Depois em concluir novamente e estará completa a Ativação.



**TABELA:**

**Após executar o SELECT**

****

**OBS: KEY= ID**

**Openquery**

Para saber mais acesse:

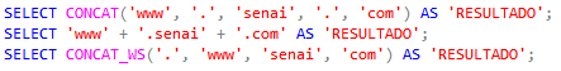
https://docs.microsoft.com/pt-br/sql/t-sql/functions/openquery-transact-sql?view=sql-server-ver15

# FUNÇÕES ESCALARES

Recebem valores únicos e retornam um valor único. As funções escalares podem ser usadas onde uma expressão é válida. Dentre as diversas subcategorias existentes, apresentaremos a seguir algumas funções que acreditamos serem úteis e interessantes.

**EXEMPLO DE UTILIZAÇÃO DA FUNÇÃO CONCAT**

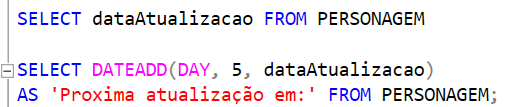
Para concatenar, ou seja, agrupar continuamente expressões em strings na forma de uma única sequencia de caracteres, podemos digitar os seguintes códigos:



Neste caso, o objetivo é obter a strings ”www.senai.com”. Perceba que o resultado para as três formas de digitação, seja utilizando o operador de concatenação “+” ou as funções CONCAT e CONCAT\_WS, será exatamente o mesmo. Nesta ultima, temos a opção de escolher o caractere separador especifico (no caso do exemplo, o ponto final “.”).

**EXEMPLO DE UTILIZAÇÃO DA FUNÇÃO DATEADD**

A função DATEADD permite somar um valor especifico de dias, meses ou anos a uma data pré-existente em colunas do banco de dados. Basta que selecionemos um dos parâmetros DAY, MONTH ou YEAR, seguido do numero correspondente a ser adicionado e o nome da coluna onde se encontra o valor de data que recebera o incremento, conforme o exemplo abaixo:



O resultado será a adição de 5 dias a data preexistente na coluna “dataAtualização” da tabela PERSONAGEM.

**EXEMPLO DE UTILIZAÇÃO DA FUNÇÃO GETDATE**

Outra função de Sistema interessante é a chamada ISNULL. Como seu nome sugere, ela identifica a presença de valores NULL em uma coluna e os substitui por uma expressão personalizada pelo usuário. Veja o exemplo:

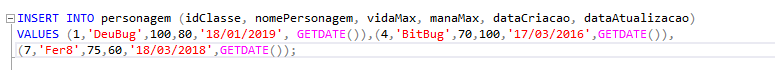
Texto

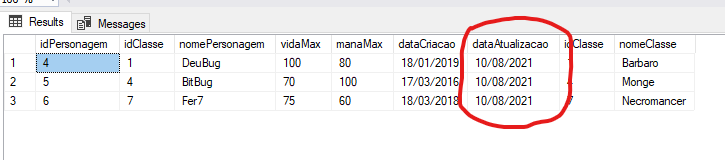
Descrição gerada automaticamente

Neste caso, os campos da tabela PERSONAGEM que corresponderem a um valor NULL serão apresentados com a expressão NÃO CADASTRADO, deixando o resultado final mais claro e legível para o usuário.

**EXEMPLO DE UTILIZAÇÃO DA FUNÇÃO GETDATE**

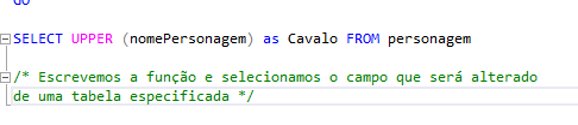
Quando criamos o projeto HROADS poderíamos utilizar a função GETDATE. Esta função pega a data atual do seu computador.

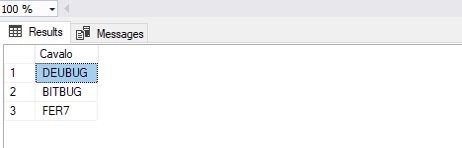




**EXEMPLO DE UTILIZAÇÃO DA FUNÇÃO UPPER**

Upper é uma função que nos retorna caracteres em caixa alta... Ele vai pegar todos caracteres de uma coluna/campo que os receba, retornar em caixa alta e quando visualizarmos o resultado, teremos o seguinte:





**DETERMINISMO DE FUNÇÃO**

As funções nativas do SQL Server são determinísticas ou não, mas o que isto quer dizer? As funções possuem retorno, porém existem as que sempre retornam o mesmo valor e as que podem retornar valores diferentes. Determinísticas serão todas as que retornam o mesmo valor, não determinísticas tem um retorno que pode mudar quando usadas. O GETDATE é um exemplo disto... Ele pega a data atual do nosso computador, então se usarmos em um outro dia, por exemplo, o retorno será diferente, classificando esta função como não determinística.

### Trello

https://github.com/MurilloAssis/SalaAulaInvertida\_FuncoesNativas

https://trello.com/b/YvSBjr3C/tarefas