FUNCTIONS IN SQLSERVER:(23-10-2024)
==========

- to perform some task as per the given input values and it must return a value.
- sqlserver supports the following two types of functions.
    1) Pre-defined functions
        - Use in SQL & T/SQL.
    2) User-defined functions
        - Use in T/SQL.

1) Pre-defined functions:
==========

- these functions are again classified into two types.
    i) Single row functions
    ii) Multiple row functions

i) Single row functions:
==========

- these functions are always return a single value only.
    > Mathematical functions
    > Character / String functions
    > Date & Time functions
    > Null function (i.e ISNULL())
    > Ranking functions
    > Conversion functions

syntax to call function:
==========
SELECT <FNAME>(value/(s))

Mathematical functions:
==========
ABS():
======

- to convert (-ve) sign values into (+ve) sign values.

syntax:
======

abs(n)

EX:
SELECT ABS(-12) AS RESULT;----------> 12
SELECT ENAME,SAL,COMM,ABS(COMM-SAL) AS RESULT FROM EMP;

CEILING():
=========
- it return a value which is equals to or greater than to the given expression.

syntax:
=======
ceiling(n)

EX:
SELECT CEILING(9.0) AS RESULT----------> 9
SELECT CEILING(9.4) AS RESULT----------> 10

FLOOR():
========
- it return a value which is equals to or less than to the given expression.

syntax:
=======
floor(n)

EX:
SELECT FLOOR(9.0) AS RESULT---------->9
SELECT FLOOR(9.8) AS RESULT---------->9

POWER():
========
- it return the power of the given expression.

syntax:
=======
power(m,n)

EX:
SELECT POWER(2,3) AS RESULT----------> 8
SELECT POWER(SAL,2) AS RESULT FROM EMP;

SQRT():
======
- it return the square root value of the given expression.

syntax:
======
sqrt(n)

EX:
SELECT SQRT(25) AS RESULT
SELECT SQRT(SAL) AS RESULT FROM EMP

SQUARE():

========
- it return the square value of the given expression.

syntax:
======
square(n)

EX:
SELECT SQUARE(10) AS RESULT
SELECT SQUARE(SAL) AS RESULT FROM EMP

24-10-2024:
===========
Character / String functions:
=======================
LEN():
======
- it return the length of the given string expression.

syntax:
=======
len(string)

EX:
SELECT LEN('HELLO') AS RESULT
SELECT LEN('WEL COME') AS RESULT

EX:
SELECT ENAME,LEN(ENAME) AS RESULT FROM EMP
SELECT * FROM EMP WHERE LEN(ENAME)=5
SELECT * FROM EMP WHERE LEN(ENAME)>5
SELECT * FROM EMP WHERE LEN(ENAME)<5

LTRIM():
=======
- to remove unwanted spaces(characters) from the left side of the given string expression.

syntax:
=======
ltrim(string)

EX:
SELECT LTRIM('   SMITH') AS RESULT
SELECT LEN(LTRIM('   SMITH')) AS RESULT

RTRIM():
=======
- to remove unwanted spaces(characters) from the right side of the given string expression.

syntax:
=======
rtrim(string)
EX:
SELECT RTRIM('SMITH ') AS RESULT
SELECT LEN(RTRIM('SMITH ')) AS RESULT

LOWER():
========
- to convert upper case characters into lower case characters.
syntax:
=======
lower(string)
EX:
SELECT LOWER('HELLO') AS RESULT
SELECT ENAME,LOWER(ENAME) FROM EMP;

UPDATE EMP SET ENAME=LOWER(ENAME) WHERE JOB='ANALYST';
UPDATE EMP SET ENAME=LOWER(ENAME);

UPPER():
========
- to convert lower case characters into upper case characters.
syntax:
=======
upper(string)

EX:
SELECT UPPER('hello') AS RESULT
UPDATE EMP SET ENAME=upper(ENAME);

LEFT():
======
- to return the no.of characters from the left side of the given string expression as per specified number.

syntax:

======
left(string,<number>)

EX:
SELECT LEFT('WELCOME',1) AS RESULT----------> W
SELECT LEFT('WELCOME',3) AS RESULT----------> WEL

RIGHT():
======
- to return the no.of characters from the right side of the given string expression as per specified number.

syntax:
======
right(string,<number>)

EX:
SELECT RIGHT('WELCOME',1) AS RESULT----------> E
SELECT RIGHT('WELCOME',3) AS RESULT----------> OME

REVERSE():
==========
- to reverse the characters from the given string expression.
syntax:
======
reverse(string)

EX:
SELECT REVERSE('SAI') AS RESULT ----------> IAS
SELECT ENAME,REVERSE(ENAME) FROM EMP;

REPLICATE():
===========
- to repeat the given string expression as per the given number.
syntax:
======
replicate(string,number)

EX:
SELECT REPLICATE('SMITH',5) AS RESULT
SELECT ENAME,REPLICATE(ENAME,3) FROM EMP;

REPLACE():

=========

- to replace the required old characters with new characters in the given string expression.

syntax:
======

replace(string,'<old characters>','<new characters>')

EX:
SELECT REPLACE('JACK AND JUE','J','BL') AS RESULT ----------> BLACK AND BLUE
SELECT REPLACE('HELLO','ELL','D') AS RESULT----------> HDO
SELECT REPLACE('HELLO','L','XYZ') AS RESULT----------> HEXYZXYZO

SUBSTRING():
============

- it return the required substring from the given string expression.

syntax:
======

substring(string,<starting position of character>,<length of characters>)

Expression :
==========

W E L C O M E
1 2 3 4 5 6 7

EX:
SELECT SUBSTRING('WELCOME',1,2) AS RESULT----------> WE
SELECT SUBSTRING('WELCOME',3,4) AS RESULT----------> LCOM
SELECT SUBSTRING('WELCOME',6,1) AS RESULT----------> M
SELECT SUBSTRING('WELCOME',4,2) AS RESULT----------> CO
SELECT SUBSTRING('WELCOME',-4,2) AS RESULT----------> (negative position are not allowed)

Date & Time functions:
==========

GETDATE():
==========

- it return the current date and time information of the system.

syntax:
======

getdate()

EX:
SELECT GETDATE() AS RESULT

DAY(),MONTH(),YEAR():
==========
- these functions are return day,month,year part from the given date expression.

syntax:
======
day(date),month(date),year(date)

EX:
SELECT DAY('2022-09-29') AS RESULT----------> 29
SELECT MONTH('2022-09-29') AS RESULT----------> 9
SELECT YEAR('2022-09-29') AS RESULT----------> 2022

DATENAME():
==========
- it return the name of day and month as per the given interval.

syntax:
=======
datename(<interval>,date)

EX:
SELECT DATENAME(DW,GETDATE()) AS RESULT
SELECT DATENAME(MM,GETDATE()) AS RESULT
SELECT DATENAME(YYYY,GETDATE()) AS RESULT

DATEADD():
=========
- to add no.of days,months and years to the given date expression.

syntax:
=======
dateadd(<interval>,<number>,date)

EX:
SELECT DATEADD(DD,30,GETDATE()) AS RESULT
SELECT DATEADD(MM,5,GETDATE()) AS RESULT
SELECT DATEADD(YYYY,2,GETDATE()) AS RESULT

DATEDIFF():
==========
- it return the no.of days,months and years are differences in between the given two date expressions.

syntax:
======
datediff(<interval>,<date1>,<date2>)

EX:
SELECT DATEDIFF(DD,'2022-10-24',GETDATE()) AS RESULT
SELECT DATEDIFF(MM,'2022-10-24',GETDATE()) AS RESULT
SELECT DATEDIFF(YYYY,'2022-10-24',GETDATE()) AS RESULT

DATEPART():
==========
- it return the specified interval from the given date expression.

syntax:
=======
datepart(<interval>,date)

EX:
SELECT DATEPART(DD,GETDATE()) AS RESULT
SELECT DATEPART(MM,GETDATE()) AS RESULT
SELECT DATEPART(YYYY,GETDATE()) AS RESULT

25-10-2024:
===========
Ranking functions:
================
- to assign rank numbers to each row wise / to each group of rows wise in the table.
- there two ranking functions in sqlserver.
    i) Rank()
    ii) Dense_Rank()

EX:

ENAME	SALARY	RANK()	DENSE_RANK()
A	85000	1	1
B	72000	2	2
C	72000	2	2
D	64000	4	3
E	55000	5	4
F	55000	5	4
G	48000	7	5
H	32000	8	6

- Rank(),Dense_Rank() functions are assigning same rank number to the same values but Rank() will skip the next sequence rank number in the order for every duplicate value whereas Dense_rank() will not skip the next sequence rank number in the order for every duplicate value.

syntax:
======
ranking function name()over([partition by <column name>] order by <column name> <asc/desc>)

Here,

partition by clause is optional

order by clause is mandatory

Examples without Partition by clause:
==========
EX:
SELECT ENAME,SAL,RANK()OVER(ORDER BY SAL DESC) AS RANKS FROM EMP
SELECT ENAME,SAL,DENSE_RANK()OVER(ORDER BY SAL DESC) AS RANKS FROM EMP

Examples with Partition by clause:
==========
EX:
SELECT ENAME,JOB,SAL,RANK()OVER(PARTITION BY JOB ORDER BY SAL DESC) AS RANKS FROM EMP
SELECT ENAME,JOB,SAL,DENSE_RANK()OVER(PARTITION BY JOB ORDER BY SAL DESC) AS RANKS FROM EMP

Conversion Function:
==========

CAST():
=======

- it is a pre-defined function which is used to convert from one datatype to another datatype.

syntax:
======

cast(<expression> as <target datatype>)

EX:
SELECT CAST(10.23 AS INT) AS RESULT ----------> 10
SELECT CAST(10.23 AS VARCHAR) AS RESULT ----------> '10.23'
SELECT CAST(10.23 AS MONEY) AS RESULT ----------> 10.23

Concatenation operator (+):
==========

- to add two or more than two string expressions.

syntax:
======

expression1 + expression2 + expression3 + ...;

EX:
SELECT 'THE EMPLOYEE'+' '+ENAME+' '+'IS WORKING AS A'+' '+JOB FROM EMP;
SELECT 'THE EMPLOYEE'+' '+ENAME+' '+'SALARY IS'+' '+CAST(SAL AS VARCHAR) FROM EMP
SELECT 'THE EMPLOYEE'+' '+ENAME+' '+'JOINED ON'+' '+CAST(HIREDATE AS VARCHAR) FROM EMP

ii) Multiple row functions:
==========
SUM():
=====
- it return total value.
Ex:
SELECT SUM(SAL) FROM EMP;
SELECT SUM(SAL) FROM EMP WHERE DEPTNO=20;

AVG():
=====
- it return average value.

Ex:
SELECT AVG(SAL) FROM EMP;
SELECT AVG(SAL) FROM EMP WHERE DEPTNO=20;

MAX():
=======
- it return maximum value.

EX:
SELECT MAX(SAL) FROM EMP;
SELECT MAX(SAL) FROM EMP WHERE JOB='MANAGER';
SELECT MAX(HIREDATE) FROM EMP;

MIN():
=====
- it return minimum value.
EX:
SELECT MIN(SAL) FROM EMP WHERE JOB='MANAGER';
SELECT MIN(HIREDATE) FROM EMP;

COUNT():
========
i) COUNT(*):

==========

- counting all rows including duplicates and nulls in a table.

EX:

SELECT COUNT(*) FROM EMP; ----------> 14

ii) COUNT(COLUMN NAME):

==========

- counting all values including duplicates but not nulls in a specific column.

EX:

SELECT COUNT(MGR) FROM EMP;----------> 13

iii) COUNT(DISTINCT <COLUMN NAME>):

==========

- counting the unique values from a column.(i.e no duplicates & no nulls)

EX:

SELECT COUNT(DISTINCT MGR) FROM EMP;----------> 6

NOTE:

======

- multiple row functions are also called as "Aggregative functions / Grouping functions" in SQLSERVER.