DBMS/SQL

Lesson 05 SQL (Single-row) Functions

Capgemini

Lesson Objectives

To understand the following topics:

- SQL (single-row) functions
  - Number functions
  - Character functions
  - Date functions
  - Conversion functions
  - Miscellaneous Single-row functions

Single Row Functions

Single-row functions return a single result row for every row of a queried Table or View.

- Single-row functions can appear in SELECT lists, WHERE clauses, START WITH and CONNECT BY clauses, and HAVING clauses.
- Different categories of single valued functions are:
  - Numerical functions
  - Character functions
  - Date and Time functions
  - Conversion functions
  - Miscellaneous Single-row functions

Single Row Functions - Characteristics

Manipulate data items
- Accept arguments and return one value
- Act on each row returned
- Return one result per row
- May modify the data type
- Can be nested
- Accept arguments which can be a column or an expression function_name [(arg1, arg2, …)]

Can be used in SELECT, WHERE, and ORDER BY clauses

Number / Integer Functions

Number/ Integer functions accept “numeric data” as argument, and returns “numeric values”.

ROUND (arg,n)	Returns “arg” rounded to “n” decimal places. If “n” is omitted, then “arg” is rounded as an integer.
CEIL (arg)	Returns the smallest integer greater than or equal to “arg”.
FLOOR (arg)	Returns the largest integer less than or equal to “arg”.
ABS (arg)	Returns the absolute value of “arg”.
POWER (arg, n)	Returns the argument “arg” raised to the n^{th} power.
SQRT (arg)	Returns the square root of “arg”.
SIGN (arg)	Returns –1, 0, or +1 according to “arg” which is negative, zero, or positive respectively.
MOD (arg1, arg2)	Returns the remainder obtained by dividing “arg1” by “arg2”.

Number / Integer Functions - Examples

Example 1:

```
SELECT ABS(-15) "Absolute"
FROM dual;
```

Absolute
15

Example 2:

```
SELECT POWER(3,2) "Raised"
FROM dual; .
```

Raised
9

Number/ Integer Functions - Examples

Example 3: ROUND(n,m): Returns n rounded to m places

```
SELECT ROUND(17.175,1) "Number"
    FROM dual;
```

17.2

Example 4: TRUNC(n,m): Returns n rounded to m places

```
SELECT TRUNC(15.81,1) "Number"
    FROM dual;
```

Number
15.8

Character/ String Functions

Character / String functions accept “character data” as argument, and returns “character” or “number” values.

LOWER (arg)	Converts alphabetic character values to lowercase.
UPPER (arg)	Converts alphabetic character values to uppercase.
INITCAP (arg)	Capitalizes first letter of each word in the argument string.
CONCAT (arg1, arg2)	Concatenates the character strings “arg1” and “arg2”.
SUBSTR (arg, pos, n)	Extracts a substring from “arg”, “n” characters long, and starting at position “pos”.
LTRIM (arg)	Removes any leading blanks in the string “arg”.
RTRIM (arg)	Removes any trailing blanks in the string “arg”.
LENGTH (arg)	Returns the number of characters in the string “arg”.
REPLACE (arg, str1, str2)	Returns the string “arg” with all occurrences of the string “str1” replaced by “str2”.
LPAD (arg, n, ch)	Pads the string “arg” on the left with the character “ch”, to a total width of “n” characters.
RPAD (arg, n, ch)	Pads the string “arg” on the right with the character “ch”, to a total width of “n” characters.
INSTR(string, pattern[, start [,occurrence]])	It returns the location of a character in a string.

Character/ String Functions - Examples

Example 1:

```
SELECT CONCAT('Hello ','World')
      "Concat"
      FROM Dual;
```

Concat
Hello World

Example 2:

```
SELECT SUBSTR('HelloWorld',1,5)
"SubString"
      FROM Dual;
```

SubSt
Hello

Date Functions

Date Functions operate on Date & Time datatype values

Name	Description
ADDDATE()	Add time values (intervals) to a date value
ADDTIME()	Add time
CURTIME()	Return the current time
DATE()	Extract the date part of a date or datetime expression
DATE_ADD()	Add time values (intervals) to a date value
PERIOD_ADD()	Add a period to a year-month
PERIOD_DIFF()	Return the number of months between periods

Date Functions- Examples

ADDDATE(*date*,INTERVAL *expr unit*), ADDDATE(*expr*,*days*)
When invoked with the INTERVAL form of the second argument, ADDDATE() is a synonym for DATE_ADD(). The related function SUBDATE() is a synonym for DATE_SUB(). For information on the INTERVAL *unit* argument

Example:

```
mysql> SELECT DATE_ADD('2008-01-02', INTERVAL 31 DAY);
          -> '2008-02-02’
mysql> SELECT ADDDATE('2008-01-02', INTERVAL 31 DAY);
          -> '2008-02-02'
```

When invoked with the *days* form of the second argument, MySQL treats it as an integer number of days to be added to *expr*.

```
mysql> SELECT ADDDATE('2008-01-02', 31);
          -> '2008-02-02'
```

Date Functions- Examples

ADDTIME(*expr1*,*expr2*)

ADDTIME() adds *expr2* to *expr1* and returns the result. *expr1* is a time or datetime expression, and *expr2* is a time expression.

Example :

```
mysql> SELECT ADDTIME('2007-12-31 23:59:59.999999', '1 1:1:1.000002');
        -> '2008-01-02 01:01:01.000001'
mysql> SELECT ADDTIME('01:00:00.999999', '02:00:00.999998');
        -> '03:00:01.999997'
```

CURDATE()

Returns the current date as a value in *'YYYY-MM-DD'* or *YYYYMMDD* format, depending on whether the function is used in a string or numeric context.

```
mysql> SELECT CURDATE();
        -> '2008-06-13'
mysql> SELECT CURDATE() + 0;
        -> 20080613
```

CURTIME()

Returns the current time as a value in *'hh:mm:ss'* or *hhmmss.uuuuuu* format, depending on whether the function is used in a string or numeric context. The value is expressed in the session time zone.

```
mysql> SELECT CURTIME();
        -> '23:50:26'
mysql> SELECT CURTIME() + 0;
        -> 235026.000000
```

Date Functions- Examples

DATE(*expr*)

Extracts the date part of the date or datetime expression *expr*.

```
mysql> SELECT DATE('2003-12-31 01:02:03');
        -> '2003-12-31'
```

DATE_ADD(*date*,INTERVAL *expr unit*), DATE_SUB(*date*,INTERVAL *expr unit*)

These functions perform date arithmetic. The *date* argument specifies the starting date or datetime value. *expr* is an expression specifying the interval value to be added or subtracted from the starting date. *expr* is evaluated as a string; it may start with a - for negative intervals. *unit* is a keyword indicating the units in which the expression should be interpreted. For more information about temporal interval syntax, including a full list of *unit* specifiers, the expected form of the *expr* argument for each *unit* value, and rules for operand interpretation in temporal arithmetic, see Temporal Intervals.

The return value depends on the arguments:

DATE if the *date* argument is a DATE value and your calculations involve only YEAR, MONTH, and DAY parts (that is, no time parts).

DATETIME if the first argument is a DATETIME (or TIMESTAMP) value, or if the first argument is a DATE and the *unit* value uses HOURS, MINUTES, or SECONDS.

String otherwise.

```
mysql> SELECT DATE_ADD('2018-05-01',INTERVAL 1 DAY);
        -> '2018-05-02'
mysql> SELECT DATE_SUB('2018-05-01',INTERVAL 1 YEAR);
        -> '2017-05-01'
```

Functions with Dates

PERIOD_ADD(*P*,*N*)

Adds *N* months to period *P* (in the format *YYMM* or *YYYYMM*). Returns a value in the format *YYYYMM*. Note that the period argument *P* is *not* a date value.

```
mysql> SELECT PERIOD_ADD(200801,2);
        -> 200803
```

PERIOD_DIFF(*P1*,*P2*)

Returns the number of months between periods *P1* and *P2*. *P1* and *P2* should be in the format *YYMM* or *YYYYMM*. Note that the period arguments *P1* and *P2* are *not* date values.

```
mysql> SELECT PERIOD_DIFF(200802,200703);
        -> 11
```

Conversion Functions

DATE_FORMAT(*date*,*format*)

Formats the *date* value according to the *format* string.

The specifiers shown in the following table may be used in the *format* string. The % character is required before format specifier characters. The specifiers apply to other functions as well: STR_TO_DATE(), TIME_FORMAT()

DATE_FORMAT() returns a string with a character set and collation given by character_set_connection and collation_connection so that it can return month and weekday names containing non-ASCII characters.

```
mysql> SELECT DATE_FORMAT('2009-10-04 22:23:00', '%W %M %Y');
    -> 'Sunday October 2009'
mysql> SELECT DATE_FORMAT('2007-10-04 22:23:00', '%H:%i:%s');
    -> '22:23:00'
mysql> SELECT DATE_FORMAT('1900-10-04 22:23:00', '%D %y %a %d %m %b %j');
    -> '4th 00 Thu 04 10 Oct 277'
mysql> SELECT DATE_FORMAT('1997-10-04 22:23:00',
    -> '%H %k %I %r %T %S %w');
    -> '22 22 10 10:23:00 PM 22:23:00 00 6'
mysql> SELECT DATE_FORMAT('1999-01-01', '%X %V');
    -> '1998 52'
mysql> SELECT DATE_FORMAT('2006-06-00', '%d');
    -> '00'
```

Conversion Functions

Specifier	Description
%D	Day of the month with English suffix (0th, 1st, 2nd, 3rd, …)
%d	Day of the month, numeric (00..31)
%H	Hour (00..23)
%h	Hour (01..12)
%I	Hour (01..12)
%i	Minutes, numeric (00..59)
%j	Day of year (001..366)
%Y	Year, numeric, four digits
%y	Year, numeric (two digits)
%M	Month name (January..December)
%m	Month, numeric (00..12)
%p	AM or PM
%S	Seconds (00..59)
%s	Seconds (00..59)

Conversion Functions - Examples

EXTRACT(*unit* FROM *date*)

The EXTRACT() function uses the same kinds of *unit* specifiers as DATE_ADD() or DATE_SUB(), but extracts parts from the date rather than performing date arithmetic. For information on the *unit* argument

Example :

```
mysql> SELECT EXTRACT(YEAR FROM '2019-07-02');
        -> 2019
mysql> SELECT EXTRACT(YEAR_MONTH FROM '2019-07-02 01:02:03');
        -> 201907
mysql> SELECT EXTRACT(DAY_MINUTE FROM '2019-07-02 01:02:03');
        -> 20102
mysql> SELECT EXTRACT(MICROSECOND -> FROM '2003-01-02
10:30:00.000123');
        -> 123
```

Miscellaneous Functions

COALESCE(*value*,...)

Returns the first non-NULL value in the list, or NULL if there are no non-NULL values.

The return type of COALESCE() is the aggregated type of the argument types.

```
mysql> SELECT COALESCE(NULL,1);
        -> 1
mysql> SELECT COALESCE(NULL,NULL,NULL);
        -> NULL
```

The Case Function

Case() function

- Conditional evaluation by doing work of an IF-THEN-ELSE statement
- Syntax

```
CASE expr when compare_expr1 then return_expr1
                [when compare_exprn then return_exprn
ELSE else_expr]
END
```

- Example

```
SELECT  staff_code, staff_name,
CASE dept_code WHEN 10 then 'Ten' ELSE 'Other' END
     FROM staff_master;
```

Example of Case using comparison operators

```
SELECT  ename, sal,
CASE
          WHEN sal >500 AND sal <1000 THEN 'OK'
          WHEN sal>1000 AND sal <2000 THEN 'Good'
          WHEN sal >2000 AND sal<3000 THEN 'Very Good'
          ELSE
          'Excellent'
END CASE FROM EMP
```

Quick Guidelines

If possible, try avoiding the SUBSTRING function in the WHERE clauses.

- Depending on how it is constructed, using the SUBSTRING function can force a table scan instead of allowing the Optimizer to use an Index (assuming there is one).
- Instead, use the LIKE condition, for better performance.

- For example: Use the second query instead of using the first query.

```
WHERE SUBSTRING(column_name,1,1) = 'b'
```

```
WHERE column_name LIKE 'b%'
```

Summary

In this lesson, you have learnt:

- SQL (single-row) functions
- Character functions
- Number functions
- Date functions
- Conversion functions
- Miscellaneous functions

Review – Questions

Question 1: Single row functions can be broadly classified as _____.

- Option 1: character functions
- Option 2: numeric functions
- Option 3: Date functions
- Option 4: all the above

Question 2: The function which returns the value after capitalizing the first character is ___.

Review – Questions

Question 3: The output of the following function will be ____.

- select to_char(17000,'$99,999.00') ‘Amount’ from dual;

Question 4: The decode function can convert NULL values to required type.

- True / False

Review – Questions

Question 5: The function which returns the last date of the month is ____.

- Option 1: LAST_DATE
- Option 2: LAST_DAY
- Option 3: MONTH_LAST_DATE
- Option 4: MONTH_LAST_DAY