```
REM Author :
REM Date :
REM Objective : Chapter 3. Built-in Function
REM Environment : Ubuntu Server 20.04 LTS, HeidiSQL 10.2.0, MySQL Community Server 5.7.34.0

REM   SQL function
-A function is a stored program that you can pass parameters into and then return a value.
1. Built Function(내장함수)
2. Stored Function(사용자 정의 함수)

REM 단일행 함수(Single Row function)
1. Syntax
   function_name(column | expression [ arg1, arg2...])

2. 종류
  1)제어흐름 함수
  2)숫자 함수
  3)날자시간 함수
  4)문자열 함수
  5)집합 함수
  6)변환 함수
  7)기타 함수


REM 제어 흐름 함수(Flow Control Functions)
1. IF()
   1)Definition
      -Returns a value if a condition is TRUE, or another value if a condition is FALSE.

   2)Syntax
      IF(expr1, expr2, expr3)

   3)만일 expr1이 참이면, expr2를 리턴한다.
   4)그렇지 않으면 expr3을 리턴한다.

   SELECT IF(1 > 2, 2, 3);   --> 3
   SELECT IF(1 < 2, 'yes', 'no') --> 'yes'


2. CASE
   1)Definition
      -Goes through conditions and return a value when the first condition is met.
      -like an IF-THEN-ELSE statement.
      -So, once a condition is true, it will stop reading and return the result.
      -If no conditions are true, it will return the value in the ELSE clause.
      -If there is no ELSE part and no conditions are true, it returns NULL.

   2)Syntax
      CASE
         WHEN compare_value1 THEN result1
         WHEN compare_value2 THEN result2
         WHEN compare_value3 THEN result3
         ...
         ELSE resultN
      END

   SELECT job, sal,
      CASE   WHEN  job = 'ANALYST' THEN sal * 1.1
```

```
            WHEN  job = 'CLERK'     THEN sal * 1.15
            WHEN  job = 'MANAGER'  THEN  sal * 1.2
            ELSE  sal
      END  AS "SALARY"
   FROM emp;


3. IFNULL()
   1)Definition
      -Returns a specified value if the expression is NULL.
      -If the expression is NOT NULL, this function returns the expression.

   2)Syntax
      IFNULL(expr1, expr2)
         -If expr1 is not NULL, IFNULL() returns expr1; otherwise it returns expr2.
         -expr1 : NULL
         -expr2 : 치환값
         -expr1값이 NULL 아니면 expr1 값을 그대로 사용
         -만약 expr1 값이 NULL이면, expr2 값으로 대체


4. NULLIF()
   1)Definition
      -Compares two expressions and returns NULL if they are equal. Otherwise, the first
       expression is returned.

   2)Syntax
      NULLIF(expr1, expr2)

   SELECT NULLIF(1,1);   --> NULL
   SELECT NULLIF(1,2);   --> 1
   SELECT NULLIF("Hello", "world"); --> 'Hello'



REM 숫자 함수(Numeric Functions)

1. ABS
   1) 숫자 값을 절대값으로 바꾼다.
   2)Syntax
      ABS(expression)

   SELECT ABS(-15)


2. CEIL(CEILING)
   1)Returns the smallest integer value that is bigger than or equal to a number.
   2)Syntax
      CEIL(number)

   SELECT CEIL(15.7)


3. DEGREES
   1)Convert radians to degrees
   2)Syntax
      DEGREES(number)

   SELECT DEGREES(PI()*2);  --> 360
   SELECT DEGREES(PI());  --> 180
```

```
   SELECT DEGREES(PI() / 2);  --> 90


4. FLOOR
   1)Returns the largest integer value that is smaller than or equal to a number.
   2)Syntax
      FLOOR(number)

   SELECT FLOOR(15.7)


5. MOD
   1)Returns the remainder of a number divided by another number.
   2)Syntax
      MOD(m, n)
         -m MOD n
         -m % n

   SELECT ename, sal, comm, MOD(sal, comm)
   FROM emp
   WHERE job = 'SALESMAN';

   SELECT 10 / 3, MOD(10, 3);
   SELECT sal, MOD(sal, 30);


6. PI
   SELECT PI();


7. POW(POWER)
   1)Returns the value of a number raised to the power of another number.

   SELECT POWER(3,2)


8. RADIANS
   1)Converts a degree value into radians.
   2)Syntax
      RADIANS(number)

   SELECT RADIANS(-45);  -->  -0.7853981633974483
   SELECT RADIANS(90);  -->  1.5707963267949


9. RAND
   1)Returns a random number between 0 (inclusive) and 1 (exclusive).
   2)Syntax
      RAND(seed)

   SELECT RAND();  --> 0.26097273012713784


10. ROUND
   1)Rounds a number to a specified number of decimal places.
   2)Syntax
      ROUND(column | expression, n)
   3) 열, 표현식, 값을 소수점 n째 자리로 반올림
   4) n을 지정하지 않은 경우 소수점 이하 값이 없어짐
   5) n이 음수이면 소수점 왼쪽 수가 반올림
```

```
178
179     SELECT ROUND(45.925, 2), ROUND(45.925, 0), ROUND(45.925, -1);
180     SELECT ROUND(-1.23);
181     SELECT ROUND(-1.58);
182     SELECT ROUND(1.298, 1);
183     SELECT ROUND(1.298, 0);
184
185
186 11. SIGN
187     1) 주어진 수가 양수이면 1, 0이면 0, 음수이면 -1
188
189     SELECT SIGN(-12);
190
191
192 12. SQRT
193     1)Returns the square root of a number.
194
195     SELECT SQRT(13);
196
197
198 13. TRUNCATE
199     1)Truncates a number to the specified number of decimal places.
200     2)열, 표현식, 값을 소수점 n째 자리까지 남기고 버린다.
201     3)Syntax
202         TRUNC (column | expression, n)
203
204     SELECT TRUNCATE(345.156, 0);   --> 345
205     SELECT TRUNCATE(1.223,1);
206     SELECT TRUNCATE(1.999,1);
207     SELECT TRUNCATE(122, -2);
208
209
210
211 REM 날짜 함수
212
213 1. 날짜데이터
214     1)MySQL은 표준 출력 형식으로 주어진 날짜 또는 시간 유형에 대한 값을 검색하지만 사용자가 제공하는 입
        력 값에 대한 다양한 형식을 해석하려고 시도한다.
215     2)다른 형식의 값을 사용하면 예측할 수 없는 결과가 발생할 수 있다.
216     3)MySQL은 여러 형식으로 값을 해석하려고 시도하지만 날짜 부분은 항상 월-일-년 또는 일-월-보다는 년-
        월-일 순서(예: '98-09-04')로 지정해야 한다.
217     4)다른 곳에서 일반적으로 사용되는 연도 순서(예: '09-04-98', '04-09-98'), 다른 순서의 문자열을 년-월
        -일 순서로 변환하려면 STR_TO_DATE() 함수가 유용할 수 있다.
218     5)2자리 연도 값을 포함하는 날짜는 세기를 알 수 없기 때문에 모호하다.
219     6)MySQL은 다음 규칙을 사용하여 2자리 연도 값을 해석한다.
220         -Year values in the range 70-99 become 1970-1999.
221         -Year values in the range 00-69 become 2000-2069.
222
223
224 2. ADDDATE
225     1)Adds a time/date interval to a date and then returns the date.
226     2)Syntax
227     ADDDATE(date, INTERVAL value addunit)
228     OR
229     ADDDATE(date, days)
230
231     SELECT ADDDATE("2017-06-15 09:34:21", INTERVAL 15 MINUTE);    -->
        2017-06-15 09:49:21
232     SELECT ADDDATE("2017-06-15 09:34:21", INTERVAL -3 HOUR);      --> 2017-06-15
        06:34:21
```

```
    SELECT ADDDATE("2017-06-15", INTERVAL -2 MONTH);            --> 2017-04-15
    SELECT DATE_ADD('2008-01-02', INTERVAL 31 DAY);            --> '2008-02-02'
    SELECT ADDDATE('2008-01-02', INTERVAL 31 DAY);             --> '2008-02-02'
    SELECT ADDDATE('2008-01-02', 31);                          --> '2008-02-02'


3. ADDTIME
    1)Adds a time interval to a time/datetime and then returns the time/datetime.
    2)Syntax
    ADDTIME(datetime, addtime)

    --Add 5 seconds and 3 microseconds to a time and return the datetime:
    SELECT ADDTIME("2017-06-15 09:34:21.000001", "5.000003");   --> 2017-06-15
    09:34:26.000004

    --Add 2 hours, 10 minutes, 5 seconds, and 3 microseconds to a time and return the
    datetime:
    SELECT ADDTIME("2017-06-15 09:34:21.000001", "2:10:5.000003");  --> 2017-06-15
    11:44:26.000004

    -Add 5 days, 2 hours, 10 minutes, 5 seconds, and 3 microseconds to a time and return
    the datetime:
    SELECT ADDTIME("2017-06-15 09:34:21.000001", "5 2:10:5.000003");  -->
    2017-06-20 11:44:26.000004

    --Add 2 hours, 10 minutes, 5 seconds, and 3 microseconds to a time and return the time:
    SELECT ADDTIME("09:34:21.000001", "2:10:5.000003");  --> 11:44:26.000004


4. CURDATE
    1)Returns the current date.
    2)The date is returned as "YYYY-MM-DD" (string) or as YYYYMMDD (numeric).
    3)This function equals the CURRENT_DATE() function.
    4)Syntax
       CURDATE()

    SELECT CURDATE() + 1;  --> 20210831
    SELECT CURDATE();       --> '2021-08-30'
    SELECT CURDATE() + 0;  --> 20210830


5. CURRENT_DATE
    1)Returns the current date.
    2)Syntax
       CURRENT_DATE()

    SELECT CURRENT_DATE() + 1;   --> 20210831


6. CURRENT_TIME
    1)Returns the current time.
    2)The time is returned as "HH-MM-SS" (string) or as HHMMSS.uuuuuu (numeric).
    3)This function equals the CURTIME() function.
    4)Syntax
       CURRENT_TIME()

    SELECT CURRENT_TIME() + 1;   --> 224909
    SELECT CURTIME();  --> --> '22:49:58'
    SELECT CURTIME() + 0;  --> 224958.000000
```

7. **CURRENT_TIMESTAMP**
   1)**Returns** the **current** date **and** time**.**
   2)The date **and** time **is** returned **as** "YYYY-MM-DD HH-MM-SS" **(**string**) or as** YYYYMMDDHHMMSS**.**uuuuuu **(**numeric**).**

```
SELECT CURRENT_TIMESTAMP();  --> '2021-08-30 22:52:13'
SELECT CURRENT_TIMESTAMP() + 1 --> 20210830225329
```

8. DATE
   1)Extracts the date part **from** a datetime expression**.**
   2)Syntax
      DATE**(**expression**)**

```
SELECT DATE("2017-06-15 09:34:21");  --> '2017-06-15'
```

9. **DATEDIFF**
   1)**Returns** the number **of** days **between** two date **values.**
   2)Syntax
      DATEDIFF(date1**,** date2**)**

```
SELECT DATEDIFF("2017-06-25 09:34:21", "2017-06-15 15:25:35");  --> 10
SELECT DATEDIFF("2017-01-01", "2016-12-24"); --> 8
```

10. **DATE_FORMAT**
   1)Formats a date **as** specified**.**
   2)Syntax
      DATE_FORMAT(date**,** **format)**

```
SELECT DATE_FORMAT("2017-06-15", "%M %d %Y");  --> June 15 2017
SELECT DATE_FORMAT("2017-06-15", "%W %M %e %Y"); --> Thursday June 15 2017
```

11. DAY
   1)**Returns** the day **of** the **month for** a given date **(**a number **from** 1 **to** 31**).**
   2)This **function equals** the DAYOFMONTH**() function.**
   3)Syntax
      DAY**(**date**)**

```
SELECT DAY("2017-06-15 09:34:21");  --> 15
SELECT DAY(CURDATE()); --> 30
```

12. DAYNAME
   1)**Returns** the weekday name **for** a given date**.**
   2)Syntax
      DAYNAME**(**date**)**

```
SELECT DAYNAME("2017-06-15 09:34:21");  --> Thursday
SELECT DAYNAME(CURDATE());  --> Monday
```

13. **LAST_DAY**
   1)Extracts the **last** day **of** the **month for** a given date**.**
   2)Syntax
      LAST_DAY(date**)**

```
   SELECT LAST_DAY("2017-02-10 09:34:00");   --> 2017-02-28


14. MAKEDATE
   1)Creates and returns a date based on a year and a number of days value.
   2)Syntax
      MAKEDATE(year, day)

   SELECT MAKEDATE(2017, 175);  --> 2017-06-24


15. MAKETIME
   1)Creates and returns a time based on an hour, minute, and second value.
   2)Syntax
      MAKETIME(hour, minute, second)

   SELECT MAKETIME(16, 1, 0);  --> 16:01:00


16. NOW
   1)Returns the current date and time.

   SELECT NOW();


17. PERIOD_ADD
   1)Adds a specified number of months to a period.
   2)Return the result formatted as YYYYMM.
   3)Syntax
      PERIOD_ADD(period, number)

   SELECT PERIOD_ADD(201703, 15);  --> 201806


18. PERIOD_DIFF
   1)Returns the difference between two periods. The result will be in months.
   2)Syntax
      PERIOD_DIFF(period1, period2)

   SELECT PERIOD_DIFF(201703, 201803);  --> -12
   SELECT PERIOD_DIFF(1703, 1612);  --> 3


19. QUARTER
   1)Returns the quarter of the year for a given date value (a number from 1 to 4).
   2)Syntax
      QUARTER(date)

   SELECT QUARTER("2017-01-01 09:34:21");   --> 1


20. STR_TO_DATE
   1)Returns a date based on a string and a format.

   SELECT STR_TO_DATE('01,5,2013','%d,%m,%Y');  --> '2013-05-01'
   SELECT STR_TO_DATE('May 1, 2013','%M %d,%Y');  --> '2013-05-01'


REM 문자 함수
```

1. **ASCII**, CHAR
   1)**Returns** the **ASCII value for** the **specific character.**
   2)**Returns** the String **value for** the **specific ASCII** code.
   3)Syntax
      ASCII(str)
      CHAR(number)

   SELECT ASCII('2'); --> 50
   SELECT CHAR(77,121,83,81,'76'); --> 'MySQL'

2. BIT_LENGTH
   1)**Returns** the **length of** the string **str in** bits.
   2)Syntax
      BIT_LENGTH(**str**)

   SELECT BIT_LENGTH('hello'); --> 40
   SELECT BIT_LENGTH('안녕'); --> 48

3. CHAR_LENGTH
   1)**Returns** the **length of** the string **str,** measured **in** characters.
   2)Syntax
      CHAR_LENGTH(**str**)

   SELECT CHAR_LENGTH("SQL Tutorial"); --> 12
   SELECT CHAR_LENGTH("안녕"); --> 2

4. **LENGTH**
   1)**Returns** the **length of** a string **(in** bytes**).**
   2)Syntax
      LENGTH(str)

   SELECT LENGTH("SQL Tutorial"); --> 12
   SELECT CHAR_LENGTH("안녕"); --> 6

5. **FORMAT**
   1)The **FORMAT() function** formats a number **to** a **format like** "#,###,###.##", rounded **to** a specified number **of** decimal places**, then** it **returns** the **result as** a string.
   2)Syntax
      FORMAT(number**,** decimal_places**)**

   SELECT FORMAT(250500.5634**,** 0**);** --> '250,501'
   SELECT FORMAT(12332.123456**,** 4**);** --> '12,332.1235'
   SELECT FORMAT(12332.1**,**4**);** --> '12.332.1000'
   SELECT FORMAT(12332.2**,**0**);** --> '12,332'
   SELECT FORMAT(12332.2**,**2**,**'de_DE'**);** --> '12.332,20'
      -If no locale **is** specified**,** the **default is** 'en_US'

6. **LOWER**
   1) 소문자로 변환
   2) Syntax
   LOWER(column | expression**)**

   SELECT empno**,** ename
   FROM emp

```
   WHERE LOWER(ename) = 'scott';


7. UPPER
   1) 대문자로 변환
   2) Syntax
   UPPER (column | expression)

   SELECT empno, ename, deptno
   FROM emp
   WHERE ename = 'blake';

   SELECT empno, ename, deptno
   FROM emp
   WHERE ename = UPPER('blake');


8. CONCAT
   1)Adds two or more expressions together.
   2)Syntax
   CONCAT(expression1, expression2, expression3,...)

   SELECT CONCAT("SQL ", "Tutorial ", "is ", "fun!")


9. SUBSTR[ING]
   1)Extracts a substring from a string (starting at any position).
   2)Syntax
   SUBSTR(string, start, length)

   SELECT SUBSTRING('Quadratically',5);  --> 'ratically'
   SELECT SUBSTRING('foobarbar' FROM 4); --> 'barbar'
   SELECT SUBSTRING('Quadratically',5,6);  --> 'ratica'
   SELECT SUBSTRING('Sakila', -3); --> 'ila'
   SELECT SUBSTRING('Sakila', -5, 3);  --> 'aki'


10. INSTR
   1)Returns the position of the first occurrence of substring substr in string str.
   2)Syntax
   INSTR(str,substr)

   SELECT INSTR('foobarbar', 'bar');    --> 4
   SELECT INSTR('xbar', 'foobar');      --> 0


11. LPAD | RPAD
   1)Left-pads a string with another string, to a certain length.
   2)Syntax
   LPAD(string, length, lpad_string)

   SELECT LPAD("SQL Tutorial", 20, "ABC");  --> ABCABCABSQL Tutorial


12. LTRIM | RTRIM
   1)Removes leading spaces from a string.
   2)Syntax
   LTRIM(string)
```

```
   SELECT LTRIM("    SQL Tutorial");  --> SQL Tutorial



13. REPLACE
   1)Replaces all occurrences of a substring within a string, with a new substring.
   2)Syntax
   REPLACE(string, substring, new_string)

   SELECT REPLACE("SQL Tutorial", "SQL", "HTML");  --> HTML Tutorial


14. REPEAT
   1)Repeats a string as many times as specified.
   2)Syntax
   REPEAT(string, number)


   SELECT REPEAT("SQL Tutorial", 3);  --> SQL TutorialSQL TutorialSQL Tutorial


15. REVERSE
   1)Reverses a string and returns the result.
   2)Syntax
   REVERSE(string)

   SELECT REVERSE("SQL Tutorial");  --> lairotuT LQS


16. SPACE
   1)Returns a string of the specified number of space characters.
   2)Syntax
      SPACE(number)

   SELECT SPACE(6);   --> '      '



REM 변환함수
1. CAST
   1)Converts a value (of any type) into the specified datatype.
   2)Syntax
      CAST(value AS datatype)

   SELECT CAST(150 AS CHAR);  --> '150'
   SELECT CAST("14:06:10" AS TIME);  --> 14:06:10


2. CONVERT
   1)Converts a value into the specified datatype or character set.
   2)Syntax
      CONVERT(value, type)
      OR
      CONVERT(value USING charset)

   SELECT CONVERT(150, CHAR);   --> '150'
```