Built in functions

Built in Functions

The Multi row functions are categorized according to the mode of action and argument`s data type into the following :

- Comparison Functions
- Control Flow Functions
- Cast Functions
- Managing Different Types of Data

Comparison functions

Comparison Functions

- Test relative values or membership value
- Functions
  - LEAST( ) returns the smallest value from a set
  - GREATEST( ) returns the largest value from a set
- Examples

```
SELECT LEAST(4,3,8,-1,5), LEAST('cdef','ab','ghi');
+-------------------+--------------------------+
| LEAST(4,3,8,-1,5) | LEAST('cdef','ab','ghi') |
+-------------------+--------------------------+
|                -1 | ab                       |
+-------------------+--------------------------+

SELECT GREATEST(4,3,8,-1,5),
 GREATEST('cdef','ab','ghi');
+----------------------+-----------------------------+
| GREATEST(4,3,8,-1,5) | GREATEST('cdef','ab','ghi') |
+----------------------+-----------------------------+
|                    8 | ghi                         |
+----------------------+-----------------------------+
```

Control Flow functions

Flow Control Functions (IF / Case)

- Choose between different values based on the result of an expression
- IF() tests the expression
  - Examples

True

```
SELECT IF(1 > 0, 'YES', 'NO');
```

False

```
+-------------------------+
| IF(1 > 0, 'YES', 'NO') |
+-------------------------+
| YES                     |
+-------------------------+
1 row in set (#.## sec)
```

Flow Control Functions

- CASE/WHEN provides branching flow control
- General syntax

```
CASE
  WHEN when_expr THEN result

  [WHEN when_expr THEN result] ...

  [ELSE result]
END
```

Managing Data Types

Managing Data Types

According to the input data type they can be classified into

String functions:

- ASCII() Functions
- CHAR_LENGTH(), CHARACTER_LENGTH(), and LENGTH() Functions
- CHARSET() and COLLATION() Functions
- CONCAT() and CONCAT_WS() Functions
- INSTR() and LOCATE() Functions
- LCASE(), LOWER(), UCASE(), and UPPER() Functions
- LEFT() and RIGHT() Functions
- REPEAT() and REVERSE() Functions
- SUBSTRING() Function

Managing Data Types

According to the input data type they can be classified into

- Numeric functions
  - CEIL(), CEILING(), and FLOOR() Functions
  - COT() Functions
  - MOD() Function
  - POW() and POWER() Functions
  - ROUND() and TRUNCATE() Functions
  - SQRT() Function

Managing Data Types

According to the input data type they can be classified into

- Date time functions

  - ADDDATE(), DATE_ADD(), SUBDATE(), DATE_SUB(),and EXTRACT() Functions
  - DATE(), MONTH(), MONTHNAME(), and YEAR() Functions
  - DATEDIFF() and TIMEDIFF() Functions
  - DAY(), DAYOFMONTH(), DAYNAME(), DAYOFWEEK(), and DAYOFYEAR() Functions
  - SECOND(), MINUTE(), HOUR(), and TIME() Functions

String functions

String Functions

- INSTR(), LOCATE() and POSITION()

```
SELECT INSTR('Alice and Bob', 'and'),
LOCATE('and', 'Alice and Bob'),
POSITION('and' IN 'Alice and Bob')\G
```

7

7

7

String Functions

- Perform operations on strings
- LENGTH()

```
SELECT LENGTH('MySQL')
```

5

String Functions

- CONCAT() and CONCAT_WS() examples

```
SELECT CONCAT('See','spot','run');
+-----------------------------+
| CONCAT('See','spot','run')|
+-----------------------------+
| Seespotrun                  |
+-----------------------------+

SELECT CONCAT_WS(' ','See','spot','run');
+-----------------------------------+
| CONCAT_WS(' ','See','spot','run') |
+-----------------------------------+
| See spot run                      |
+-----------------------------------+
```

String Functions

- SUBSTRING()

```
SELECT SUBSTRING('Alice and Bob', 1, 5);
+----------------------------------+
| SUBSTRING('Alice and Bob', 1, 5) |
+----------------------------------+
| Alice                            |
+----------------------------------+
```

String Functions

- LEFT() and RIGHT()

```
SELECT LEFT('Alice and Bob', 5);
+--------------------------+
| LEFT('Alice and Bob', 5) |
+--------------------------+
| Alice                    |
+--------------------------+

SELECT RIGHT('Alice and Bob', 3);
+---------------------------+
| RIGHT('Alice and Bob', 3) |
+---------------------------+
| Bob                       |
+---------------------------+
```

String Functions

- INSERT() and REPLACE()

```
SELECT REPLACE('Alice & Bob', '&', 'and');
+-----------------------------------+
| REPLACE('Alice & Bob', '&', 'and') |
+-----------------------------------+
| Alice and Bob                     |
+-----------------------------------+

SELECT INSERT('Alice and Bob', 6, 5, ', Carol & ');
+--------------------------------------------+
| INSERT('Alice and Bob', 6, 5, ', Carol & ') |
+--------------------------------------------+
| Alice, Carol & Bob                         |
+--------------------------------------------+
```

Numeric functions

Numeric Functions (1/4)

- Mathematical operations
- Common functions
  - TRUNCATE()
  - FLOOR()
  - CEILING()
  - ROUND()
  - ABS()
  - SIGN()
  - SIN(), COS(), TAN()

Numeric Functions (2/4)

- **ROUND** examples

```
SELECT ROUND(28.5), ROUND(-28.5);
+-------------+--------------+
| ROUND(28.5) | ROUND(-28.5) |
+-------------+--------------+
| 29          | -29          |
+-------------+--------------+
```

Numeric Functions (3/4)

- FLOOR/CEILING examples

```
SELECT FLOOR(-14.7), FLOOR(14.7);
+--------------+-------------+
| FLOOR(-14.7) | FLOOR(14.7) |
+--------------+-------------+
|          -15 |          14 |
+--------------+-------------+

SELECT CEILING(-14.7), CEILING(14.7);
+----------------+---------------+
| CEILING(-14.7) | CEILING(14.7) |
+----------------+---------------+
|            -14 |            15 |
+----------------+---------------+
```

Numeric Functions (4/4)

- ABS/SIGN examples

```
SELECT ABS(-14.7), ABS(14.7);
+------------+-----------+
| ABS(-14.7) | ABS(14.7) |
+------------+-----------+
| 14.7       | 14.7      |
+------------+-----------+

SELECT SIGN(-14.7), SIGN(14.7), SIGN(0);
+-------------+------------+---------+
| SIGN(-14.7) | SIGN(14.7) | SIGN(0) |
+-------------+------------+---------+
|          -1 |          1 |       0 |
+-------------+------------+---------+
```

Date/Time functions

Temporal Functions (1/5)

- Time, Date, Year
- Perform many operations
- Functions

Functions	Definition
`NOW()`	*Current date and time* as set on the client host (in **DATETIME** format)
`CURDATE()`	*Current date* as set on the client host (in **DATE** format)
`CURTIME()`	*Current time* as set on the client host (in **TIME** format)
`YEAR()`	*Year* in **YEAR** format, per value indicated (can use **NOW()** function within parenthesis to get current year per client)
`MONTH()`	*Month of the year* in *integer* format, per value indicated (can use **NOW()** as above)
`DAYOFMONTH()` *or* `DAY()`	*Day of the month* in integer format, per value indicated (can use **NOW()** as above)
`DAYNAME()` *(English)*	*Day of the week* in *string* format, per value indicated (can use **NOW()** as above)
`HOUR()`	*Hour of the Day* in *integer* format, per value indicated (can use **NOW()** as above)
`MINUTE()`	*Minute of the Day* in *integer* format, per value indicated (can use **NOW()** as above)
`SECOND()`	*Second of the Minute* in *integer* format, per value indicated (can use **NOW()** as above)
`GET_FORMAT()`	Returns a *date format string*, per values indicated for date-type and international format.

Temporal Functions (2/5)

- View current date and time

```
SELECT NOW();
+---------------------+
| NOW()               |
+---------------------+
| 2004-04-30 11:59:15 |
+---------------------+
1 row in set (#.## sec)
```

Temporal Functions (3/5)

- Extracting parts of date/time examples

```
SELECT YEAR('2010-04-15'), MONTH('2010-04-15'),
DAYOFMONTH('2010-04-15');
```

```
+--------------------+---------------------+--------------------------+
| YEAR('2010-04-15') | MONTH('2010-04-15') | DAYOFMONTH('2010-04-15') |
+--------------------+---------------------+--------------------------+
|               2010 |                   4 |                       15 |
+--------------------+---------------------+--------------------------+
```

Temporal Functions (3/5)

- Extracting parts of date/time examples

```
SELECT DAYOFYEAR('2010-04-15');
```

```
+--------------------------+
| DAYOFYEAR('2010-04-15') |
+--------------------------+
|                      105 |
+--------------------------+
```

```
SELECT HOUR('09:23:57'), MINUTE('09:23:57'), SECOND('09:23:57');
+------------------+--------------------+--------------------+
| HOUR('09:23:57') | MINUTE('09:23:57') | SECOND('09:23:57') |
+------------------+--------------------+--------------------+
|                9 |                 23 |                 57 |
+------------------+--------------------+--------------------+
```

Temporal Functions (4/5)

- Composite dates/times examples

```
SELECT MAKEDATE(2010,105);
+--------------------+
| MAKEDATE(2010,105) |
+--------------------+
| 2010-04-15         |
+--------------------+

SELECT MAKETIME(9,23,57);
+-------------------+
| MAKETIME(9,23,57) |
+-------------------+
| 09:23:57          |
+-------------------+
```

Temporal Functions (5/5)

- Current dates/times examples

```
SELECT CURRENT_DATE(),
       CURRENT_TIME(),
       CURRENT_TIMESTAMP();
+----------------+----------------+---------------------+
| CURRENT_DATE() | CURRENT_TIME() | CURRENT_TIMESTAMP() |
+----------------+----------------+---------------------+
| 2005-05-31     | 21:40:18       | 2005-05-31 21:40:18 |
+----------------+----------------+---------------------+
```

NULL-Related Functions

- Specifically for use with NULL
- ISNULL()/IFNULL() examples

```
SELECT ISNULL(NULL), ISNULL(0), ISNULL(1);
+--------------+-----------+-----------+
| ISNULL(NULL) | ISNULL(0) | ISNULL(1) |
+--------------+-----------+-----------+
|            1 |         0 |         0 |
+--------------+-----------+-----------+
```

```
SELECT IFNULL (NULL,'a'), IFNULL(0,'b');
+------------------+---------------+
| IFNULL(NULL,'a') | IFNULL(0,'b') |
+------------------+---------------+
| a                | 0             |
+------------------+---------------+
```

Comments in SQL Statements

- MySQL supports three forms of syntax
  - '#'
  - /* or /*!
  - --
- Examples

```
/* this is a comment */

/*
  this
  is a
  comment,
  too
*/
```