CHAPTER 6
FUNGSI PADA MYSQL

Fungsi-Fungsi Umum pada MySQL

Fungsi-fungsi yang dibahas di lampiran ini hanya fungsi-fungsi umum yang sering digunakan saja. Fungsi-fungsi tersebut dibagi menjadi beberapa kategori dan disusun secara alfabet sebagai berikut:

- Fungsi String (*String Functions*)
- Fungsi Numerik (*Numeric Functions*)
- Fungsi Tanggal dan Waktu (*Date and Time Functions*)
- Fungsi Pembanding (*Comparison Functions*)
- Fungsi Agregate (*Aggregate Functions*)
- Fungsi Lain-lain (*Miscellaneous Functions*)

Penulisan fungsi yang benar adalah NamaFungsi( ), dan bukannya NamaFungsi ( ). Perhatikan bahwa posisi tanda kurung ( ) selalu menempel dengan nama fungsinya. Bila tidak, maka MySQL akan menampilkan pesan error.

Contoh:

```
mysql> SELECT NOW ( ) ;
ERROR 1064: You have an error in your SQL syntax near '( )' at line 1
```

Kita coba dengan tanda kurung menempel pada nama fungsinya:

```
mysql> SELECT  NOW( ) ;
+---------------------+
| NOW( )              |
+---------------------+
| 2003-04-02  22:27:47 |
+---------------------+
1 row in set (0.01 sec)
```

Beberapa simbol digunakan dalam penulisan fungsi dengan maksud sebagai berikut:

- Simbol kurung siku "[" dan "]" digunakan untuk mengapit fungsi yang bersifat pilihan (optional). Bila ada lebih dari satu fungsi di dalam tanda kurung siku, maka hanya salah satu dari fungsi tersebut yang dapat dipilih.
- Simbol kurung kurawal "{" dan "}" digunakan untuk mengapit fungsi yang bersifat mutlak. Bila ada lebih dari satu fungsi di dalam tanda kurung kurawal, maka Anda harus memilih salah satu dari pilihan yang ada.
- Simbol garis tegak lurus "|" digunakan sebagai pemisah antara pilihan yang ada.
- Simbol titik yang berulang ",,,,," menandakan adanya pengulangan fungsi sebelumnya.
- Fungsi MySQL ditulis dengan huruf kapital tebal. Sedangkan nama database, tabel dan sebagainya ditulis dengan huruf biasa miring.

Fungsi String (String Functions)

✍ **ASCII(**string**)**

Fungsi ini digunakan untuk memberikan nilai ASCII dari sebuah karakter. Bila yang dimasukkan adalah sebuah kata atau sederetan huruf/karakter, maka yang diberikan nilai ASCII-nya adalah karakter pada urutan paling kiri.

Contoh:

```
mysql> SELECT ASCII('A'), ASCII('a');
+------------+------------+
| ASCII('A') | ASCII('a') |
+------------+------------+
|         65 |         97 |
+------------+------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT ASCII(1), ASCII(NULL), ASCII(' ')
+----------+-------------+------------+
| ASCII(1) | ASCII(NULL) | ASCII(' ') |
+----------+-------------+------------+
|       49 |        NULL |         32 |
+----------+-------------+------------+
1 row in set (0.00 sec)
```

✍ **CHAR(**numerik1**)**

Fungsi ini akan memberikan bentuk ASCII dari suatu bilangan. Bila yang dimasukkan adalah NULL, maka hasilnya selalu diabaikan.

Contoh:

```
mysql> SELECT CHAR(65), CHAR(97), CHAR(3);
+----------+----------+---------+
| CHAR(65) | CHAR(97) | CHAR(3) |
+----------+----------+---------+
| A        | a        | ♥       |
+----------+----------+---------+
1 row in set (0.05 sec)
```

✍ **BIN(**numerik**)**

Fungsi ini akan memberikan bentuk binari dari suatu bilangan. Bila yang dimasukkan adalah karakter huruf, maka hasilnya selalu 0 (nol).

Contoh:

```
mysql> SELECT BIN(33), BIN(6), BIN('25'), BIN(3456), BIN('arbie') ;
+--------+--------+-----------+--------------+--------------+
| BIN(1) | BIN(6) | BIN('25') | BIN(3456)    | BIN('arbie') |
+--------+--------+-----------+--------------+--------------+
| 1      | 110    | 11001     | 110110000000 | 0            |
+--------+--------+-----------+--------------+--------------+
```

✍ **OCT(**numerik**)**

Fungsi ini akan menghasilkan **n** menjadi format bilangan oktal (bilangan basis 10). Alternatif fungsi ini adalah dengan menggunakan fungsi CONV( ).

Contoh:

```
mysql> SELECT OCT(15), OCT(16), OCT(x) ;
+--------+--------+--------+
| OCT(8) |OCT(16) |OCT(x)  |
+--------+--------+--------+
| 10     | 20     | 26     |
+--------+--------+--------+
```

✍ **HEX(**numerik**)**

Fungsi ini akan menghasilkan parameter numerik menjadi format bilangan hexadesimal. Alternatif fungsi ini adalah dengan menggunakan fungsi CONV( ).

Contoh:

```
mysql> SELECT HEX(17), HEX(123), HEX(10000) ;
+----------+----------+------------+
| HEX(255) | HEX(123) | HEX(10000) |
+----------+----------+------------+
| FF       | 7B       | 2710       |
+----------+----------+------------+
1 row in set (0.00 sec)
```

✍ **CONV(**numerik, basis_bilangan_awal, basis_bilangan_tujuan**)**

Fungsi ini akan mengubah numerik (atau karakter) dari basis_bilangan_awal menjadi basis_bilangan_tujuan. Nilai basis_bilangan berkisar dari bilangan basis 2 (biner) sampai dengan bilangan basis 36.

Contoh:

Mengubah angka 25 (dalam bentuk biner adalah 11001) menjadi bilangan hexadesimal:

```
mysql> SELECT CONV(11001, 2, 16);
+--------------------+
| CONV(11001, 2, 16) |
+--------------------+
| 19                 |
+--------------------+
1 row in set (0.00 sec)
```

11110000 → DES
240 → BIN
$15_{(8)}$ → ? $_{(16)}$

Mengubah bentuk biner 11111111 menjadi bilangan basis 10:

```
mysql> SELECT CONV(11111111,2,10) ;
+-----------------------+
| CONV(11111111, 2, 10) |
+-----------------------+
| 255                   |
+-----------------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT CONV(255,10,16), CONV(123,10,16), CONV(10000,10,16) ;
+-----------------+-----------------+-------------------+
| CONV(255,10,16) | CONV(123,10,16) | CONV(10000,10,16) |
+-----------------+-----------------+-------------------+
| FF              | 7B              | 2710              |
+-----------------+-----------------+-------------------+
```

mysql> SELECT CONV('FF',16,10), CONV('7B',16,10), CONV('2710',16,10) ;

```
+------------------+------------------+--------------------+
| CONV('FF',16,10) | CONV('7B',16,10) | CONV('2710',16,10) |
+------------------+------------------+--------------------+
| 255              | 123              | 10000              |
+------------------+------------------+--------------------+
```

1 row in set (0.00 sec)

✍ **CHARACTER_LENGTH(**string**) atau CHAR_LENGTH(**string**) atau LENGTH(string)**

Fungsi ini akan menghasilkan jumlah/panjang karakter dari suatu teks. Fungsi ini sama dengan fungsi LENGTH( ) dan CHAR_LENGTH( ).

Contoh:

mysql>SELECT LENGTH('Indonesia merdeka');

```
+------------------------+-----------------------------+
| CHARACTER_LENGTH('abc') | CHARACTER_LENGTH('matahari') |
+------------------------+-----------------------------+
|                      3 |                           8 |
+------------------------+-----------------------------+
```

1 row in set (0.00 sec)

✍ **CHAR_LENGTH(**string**)**

Fungsi ini akan menghasilkan jumlah (panjang) karakter dari suatu teks. Fungsi ini sama dengan fungsi LENGTH( ) dan CHARACTER_LENGTH( ).

Contoh:

mysql> SELECT CHAR_LENGTH('abc'), CHAR_LENGTH('matahari') ;

```
+--------------------+-------------------------+
| CHAR_LENGTH('abc') | CHAR_LENGTH('matahari') |
+--------------------+-------------------------+
|                  3 |                       8 |
+--------------------+-------------------------+
```

1 row in set (0.00 sec)

✍ **CONCAT(**string1, string2, …)

Fungsi ini akan menggabungkan semua argumen teks yang ada, kecuali bila ada salah satu argumen yang NULL.

Contoh:

mysql> SELECT CONCAT('mata','hari');

```
+----------------------+-------------------------+
| CONCAT('mata','hari') | CONCAT('matahari',NULL) |
+----------------------+-------------------------+
| matahari              | NULL                    |
+----------------------+-------------------------+
```

1 row in set (0.00 sec)

✍ **ELT(**numerik, string1, string2, …**)**

Fungsi ini akan menghasilkan argumen yang ke-n dari daftar argumen yang ada. Kecuali bila nilai ke-n adalah 0, maka hasilnya menjadi NULL.

Contoh:

```
mysql> SELECT ELT(2,'x','y','z','a','b','c');
+-------------------------------+-------------------------------+
| ELT(2,'x','y','z','a','b','c') | ELT(8,'x','y','z','a','b','c') |
+-------------------------------+-------------------------------+
| y                             | NULL                          |
+-------------------------------+-------------------------------+
1 row in set (0.00 sec)
```

✍ **FIELD(**string, str1, str2, …**)**

Fungsi ini akan menghasilkan posisi karakter string pada daftar karakter *(str1, str2,…)* yang ada.

Contoh:

```
mysql> SELECT FIELD('d','a','c','d','z');
+--------------------------+----------------------+
| FIELD('z','a','c','d','z') | FIELD('z','a','c','d' |
+--------------------------+----------------------+
|                        4 |                    0 |
+--------------------------+----------------------+
1 row in set (0.00 sec)
```

✍ **FIND_IN_SET(**str1, daftar_string**)**

Fungsi ini akan menghasilkan posisi karakter string pada daftar string *(string_list)* yang ada. Bila string yang dicari tidak ada dalam daftar, akan menghasilkan nilai 0 (nol).

Sedangkan bila ada salah satu nilai yang NULL, maka akan menghasilkan nilai NULL.

Contoh:

```
mysql> SELECT FIND_IN_SET('ada','bunga, mawar, ada, lima') ;
+-------------------------------------------- +
| FIND_IN_SET('ada','bunga, mawar, ada, lima') |
+-------------------------------------------- +
|                                           3  |
+-------------------------------------------- +
1 row in set (0.00 sec)
```

✍ **FORMAT(**X, D**)**

Fungsi ini akan menghasilkan format bilangan X menjadi bilangan D desimal. Format bilangan di sini menggunakan format internasional (tanda desimal sebagai titik, bukannya koma), sehingga berbeda dengan format bahasa Indonesia.

Contoh:

```
mysql> SELECT FORMAT(123456789,0),FORMAT(123456789,2);
+---------------------+---------------------+
| FORMAT(123456789,0) | FORMAT(123456789,2) |
+---------------------+---------------------+
| 123,456,789         | 123,456,789.00      |
+---------------------+---------------------+
1 row in set (0.00 sec)
```

✍ **INSERT(**string_lama, posisi_awal, jumlah_karakter, string_baru**)**

Fungsi ini akan mengganti string_lama dengan string_baru, dimulai dari posisi_awal sebanyak jumlah_karakter. Bila nilai pada posisi_awal adalah 0, maka tidak ada penggantian string yang

terjadi.

Contoh:

```
mysql> SELECT INSERT("matahari", 5, 2, "NA") ;
+-----------------------------------+
| INSERT("matahari", 5, 3, "herna") |
+-----------------------------------+
| mathernai                         |
+-----------------------------------+
1 row in set (0.00 sec)

mysql> SELECT INSERT("matahari terbenam", 1, 0, "sebelum ") ;
+-----------------------------------------------+
| INSERT("matahari terbenam", 1, 0, "sebelum ") |
+-----------------------------------------------+
| sebelum matahari terbenam                     |
+-----------------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT INSERT("matahari terbenam", 0, 1, "sebelum ") ;
+-----------------------------------------------+
| INSERT("matahari terbenam", 0, 1, "sebelum ") |
+-----------------------------------------------+
| matahari terbenam                             |
+-----------------------------------------------+
1 row in set (0.00 sec)
```

✍ **INSTR**(string, substr)

Fungsi ini akan menghasilkan nilai posisi suatu string di dalam suatu kata/kalimat *substr* tertentu. Bila posisi yang dihasilkan ada banyak, maka nilai yang ditampilkan adalah yang pertama kali ditemukan. Fungsi ini membedakan penulisan huruf besar dengan huruf kecil.

Contoh:

```
mysql> SELECT INSTR("matahari", "t");
+-----------------------+-----------------------+
| INSTR("matahari", "t") | INSTR("matahari", "a") |
+-----------------------+------------------- ----+
|                     3 |                     2 |
+-----------------------+-----------------------+
1 row in set (0.00 sec)

mysql> SELECT INSTR("matahari", "T"), INSTR("matahari", "A");
+-----------------------+-----------------------+
| INSTR("matahari", "T") | INSTR("matahari", "A") |
+-----------------------+-----------------------+
|                     0 |                     0 |
+-----------------------+-----------------------+
1 row in set (0.00 sec)
```

✍ **LOCATE**(substr, string)

Fungsi ini akan menghasilkan nilai posisi suatu string di dalam suatu kata/kalimat *substr* tertentu. Bila posisi yang dihasilkan ada banyak, maka nilai yang ditampilkan adalah yang pertama kali ditemukan. Fungsi ini membedakan penulisan huruf besar dengan huruf kecil. Fungsi LOCATE( ) mirip dengan

fungsi INSTR( ), hanya saja penulisan argumennya terbalik.

Contoh:

```
mysql> SELECT LOCATE('t',"matahari");
+-----------------------+-----------------------+
| locate("t","matahari") | INSTR("matahari",'t')  |
+-----------------------+-----------------------+
|                     3 |                     4 |
+-----------------------+-----------------------+
1 row in set (0.00 sec)
```

✍ **LEFT(**string, jumlah_karakter**)**

Fungsi ini akan menampilkan karakter sebanyak jumlah_karakter dari posisi paling kiri string. Bila jumlah_karakter melebihi jumlah yang ada, maka seluruh string akan ditampilkan.

Contoh:

```
mysql> SELECT LEFT("ADA APA DENGAN CINTA",6) ;
+------------------------------+
| LEFT("ADA APA DENGAN CINTA",7)|
+------------------------------+
| ADA APA                      |
+------------------------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT LEFT("ADA APA DENGAN CINTA",7) ;
+---------------------------------+
| LEFT("ADA APA DENGAN CINTA",100) |
+---------------------------------+
| ADA APA DENGAN CINTA             |
+---------------------------------+
1 row in set (0.00 sec)
```

✍ **MID(**string, posisi_awal, panjang_karakter **)**

Fungsi ini akan memotong (atau menampilkan) string sebanyak *panjang_karakter* dimulai dari *posisi_awal*.

Contoh:

```
mysql> SELECT MID("COKLAT RASA STRAWBERI",8,4) ;
+---------------------------------+
| MID("COKLAT RASA STRAWBERI",8,4) |
+---------------------------------+
| TRAWBER                         |
+---------------------------------+
1 row in set (0.00 sec)
```

✍ **LCASE(**string**) atau LOWER(string)**

Fungsi ini akan mengubah penulisan string menjadi huruf kecil. Kecuali pada NULL, yang akan menghasilkan NULL juga.

Contoh:

```
mysql> SELECT LCASE("ADA APA DENGAN CINTA"), LCASE('NULL'), LCASE(NULL) ;
+------------------------------+--------------+-------------+
| LCASE("ADA APA DENGAN CINTA") | LCASE('null') | LCASE(null) |
+------------------------------+--------------+-------------+
| ada apa dengan cinta          | null          | NULL        |
+------------------------------+--------------+-------------+
1 row in set (0.00 sec)
```

✍ **LPAD(**string, jumlah_pengulangan, karakter_pengulang**)**

Fungsi ini akan mengisi *karakter_pengulang* sebanyak *jumlah_pengulangan* dengan posisi di sebelah kiri string. Perhatikan bahwa panjang (jumlah) string ikut diperhitungkan dalam jumlah_pengulangan.

Contoh:

```
mysql> SELECT RPAD("!!!!!",9,char(3));
+-----------------+------------------+-------------------+
| LPAD("A",5,".") | LPAD('AB',5,".") | LPAD('ABC',5,".") |
+-----------------+------------------+-------------------+
| ....A           | ...AB            | ..ABC             |
+-----------------+------------------+-------------------+
1 row in set (0.00 sec)

mysql> SELECT LPAD("ABCD",5,"."), LPAD('ABCDE',5,"."),
   -> LPAD('ABCDEF',5,".") ;
+--------------------+---------------------+----------------------+
| LPAD("ABCD",5,".") | LPAD('ABCDE',5,".") | LPAD('ABCDEF',5,".") |
+--------------------+---------------------+----------------------+
| .ABCD              | ABCDE               | ABCDE                |
+--------------------+---------------------+----------------------+
1 row in set (0.00 sec)
```

✍ **LTRIM(**string**)**

Fungsi ini akan membuang spasi kosong yang ada di sebelah kiri string.

Contoh:

```
mysql> SELECT  LENGTH(LTRIM("  ADA APA?   "));

mysql> SELECT  LENGTH("  ADA APA?   ");

+-------------------------+----------------------+
| LTRIM("   ADA APA?    ") | LTRIM("ADA APA?    ") |
+-------------------------+----------------------+
| ADA APA?                | ADA APA?             |
+-------------------------+----------------------+
1 row in set (0.00 sec)
```

RTRIM(string **)**

Fungsi ini akan membuang spasi kosong yang ada di sebelah kanan string.

Contoh:

```
mysql> SELECT RTRIM("   ADA APA?   "), RTRIM("ADA APA?   ") ;
+-------------------------+----------------------+
| RTRIM("   ADA APA?    ") | RTRIM("ADA APA?    ") |
+-------------------------+----------------------+
|     ADA APA?            | ADA APA?             |
+-------------------------+----------------------+
```

✍ **OCTET_LENGTH(**string**)**

Fungsi ini akan menghasilkan jumlah (panjang) karakter dari suatu teks. Fungsi ini sama dengan fungsi CHARACTER_LENGTH( ), CHAR_LENGTH( ), dan LENGTH( ).

Contoh:

```
mysql> SELECT OCTET_LENGTH('abc'), OCTET_LENGTH('matahari') ;
+---------------------+--------------------------+
| OCTET_LENGTH('abc') | OCTET_LENGTH('matahari') |
+---------------------+--------------------------+
|                   3 |                        8 |
+---------------------+--------------------------+
1 row in set (0.00 sec)
```

✍ **POSITION(**substr **IN** string**)**

Fungsi ini akan menghasilkan nilai posisi suatu *substr* di dalam suatu kata/kalimat *substr* tertentu. Bila posisi yang dihasilkan ada banyak, maka nilai yang ditampilkan adalah yang pertama kali ditemukan. Fungsi ini membedakan penulisan huruf besar dengan huruf kecil. Fungsi POSITION( ) mirip dengan fungsi LOCATE( ), hanya saja penulisan argumennya ditambah kata IN.

Contoh:

```
mysql> SELECT POSITION("t" IN "matahari"), POSITION("a" IN "matahari");
+-----------------------------+-----------------------------+
| POSITION("t" in "matahari") | POSITION("a" in "matahari") |
+-----------------------------+-----------------------------+
|                           3 |                           2 |
+-----------------------------+-----------------------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT POSITION("T" IN "matahari"), POSITION("A" IN "matahari");
+-----------------------------+-----------------------------+
| POSITION("T" IN "matahari") | POSITION("A" IN "matahari") |
+-----------------------------+-----------------------------+
|                           0 |                           0 |
+-----------------------------+-----------------------------+
1 row in set (0.00 sec)
```

✍ **REPEAT(**string, jumlah_pengulangan**)**

Fungsi ini akan menghasilkan pengulangan string sebanyak *jumlah_pengulangan*.

Contoh:

```
mysql> SELECT REPEAT('ada',3), REPEAT('ada ',3) ;
+---------------+----------------+
| REPEAT('ada') | REPEAT('ada ') |
+---------------+----------------+
| adaadada      | ada ada ada    |
+---------------+----------------+
1 row in set (0.00 sec)
```

✍ **REPLACE(**string, string_awal, string_pengganti**)**

Fungsi ini akan mengganti *string_awal* yang ada pada sebuah string dengan *string_pengganti*. Fungsi ini membedakan penulisan huruf besar dengan huruf kecil (*case sensitive*).

Contoh:

```
mysql> SELECT REPLACE('Ada Apa Dengan Cinta','A','I') ;
+---------------------------------------+
| REPLACE('Ada Apa Dengan Cinta','A','I') |
+---------------------------------------+
| Ida Ipa Dengan Cinta                  |
+---------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT REPLACE('Ada Apa Dengan Cinta','a','I') ;
+---------------------------------------+
| REPLACE('Ada Apa Dengan Cinta','a','i') |
+---------------------------------------+
| AdI ApI DengIn CintI                  |
+---------------------------------------+
1 row in set (0.00 sec)
```

✍ **REVERSE**(string)

Fungsi ini akan membalikkan susunan huruf dari suatu string.

Contoh:

```
mysql> SELECT REVERSE('Ada Apa Dengan Cinta') ;
+---------------------------------+
| REVERSE('Ada Apa Dengan Cinta') |
+---------------------------------+
| atniC nagneD apA adA            |
+---------------------------------+
1 row in set (0.00 sec)
```

✍ **RIGHT**(string, jumlah_karakter)

Fungsi ini akan menampilkan karakter sebanyak *jumlah_karakter* dari posisi paling kanan string. Bila *jumlah_karakter* ditampilkan.

Contoh:

melebihi jumlah yang ada, maka seluruh string akan

```
mysql> SELECT RIGHT("ADA APA DENGAN CINTA",7) ;
+---------------------------------+
| RIGHT("ADA APA DENGAN CINTA",5) |
+---------------------------------+
| CINTA                           |
+---------------------------------+
1 row in set (0.00 sec)

mysql> SELECT RIGHT("ADA APA DENGAN CINTA",100) ;
+----------------------------------+
| RIGHT("ADA APA DENGAN CINTA",100) |
+----------------------------------+
| ADA APA DENGAN CINTA             |
+----------------------------------+
1 row in set (0.00 sec)
```

✍ **RPAD**(string, jumlah_pengulangan, karakter_pengulang)

Fungsi ini akan mengisi *karakter_pengulang* sebanyak *jumlah_pengulangan* dengan posisi di sebelah kanan string. Perhatikan bahwa panjang (jumlah) string ikut diperhitungkan dalam jumlah_pengulangan.

Contoh:

```
mysql> SELECT RPAD("A",5,."), RPAD('AB',5,."), RPAD('ABC',5,.") ;
+-----------------+-----------------+------------------+
| RPAD("A",5,.") | RPAD('AB',5,.") | RPAD('ABC',5,.") |
+-----------------+-----------------+------------------+
| A....           | AB...           | ABC..            |
+-----------------+-----------------+------------------+
1 row in set (0.00 sec)

mysql> SELECT RPAD("ABCD",5,."), RPAD('ABCDE',5,."),
   -> RPAD('ABCDEF',5,.") ;
+-------------------+-------------------+--------------------+
| RPAD("ABCD",5,.") | RPAD('ABCDE',5,.") | RPAD('ABCDEF',5,.") |
+-------------------+-------------------+--------------------+
| ABCD.             | ABCDE             | ABCDE              |
+-------------------+-------------------+--------------------+
1 row in set (0.00 sec)
```

✍ **SPACE**(numerik)

Fungsi ini akan menambah atau membuat spasi kosong sebanyak **n** karakter.

Contoh:

```
mysql> SELECT SPACE(0), SPACE(6), SPACE(10) ;
+----------+----------+--------------+
| SPACE(0) | SPACE(6) | SPACE(10)    |
+----------+----------+--------------+
| ""       | "------" | "----------" |
+----------+----------+--------------+
1 row in set (0.00 sec)
```

Catatan: Karena space tidak akan tampil dalam bentuk yang bisa dilihat, maka dalam contoh ini satu space diwakili oleh satu tanda '-'.

✍ **SUBSTR(**string, posisi_awal **)** atau **SUBSTR(**string, posisi_awal, jumlah_karakter**)** atau **SUBSTR(**string **FROM** posisi_awal**)** atau **SUBSTR(**string **FROM** posisi_awal **FOR** jumlah_karakter**)**

Fungsi ini akan memotong (atau menampilkan) string sebanyak *jumlah_karakter* dimulai dari *posisi_awal*.

Contoh:

```
mysql> SELECT SUBSTRING("COKLAT RASA STRAWBERI",8) ;
+-------------------------------------+
| SUBSTRING("COKLAT RASA STRAWBERI",8) |
+-------------------------------------+
| RASA STRAWBERI                      |
+-------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT SUBSTRING("COKLAT RASA STRAWBERI" FROM 8) ;
+-----------------------------------------+
| SUBSTRING("COKLAT RASA STRAWBERI" FROM 8) |
+-----------------------------------------+
| RASA STRAWBERI                          |
+-----------------------------------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT SUBSTRING("COKLAT RASA STRAWBERI",8, 4) ;
+---------------------------------------+
| SUBSTRING("COKLAT RASA STRAWBERI",8, 4) |
+---------------------------------------+
| RASA                                  |
+---------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT SUBSTRING("COKLAT RASA STRAWBERI" FROM 8 FOR 4) ;
+-------------------------------------------------+
| SUBSTRING("COKLAT RASA STRAWBERI" FROM 8 FOR 4) |
+-------------------------------------------------+
| RASA                                            |
+-------------------------------------------------+
1 row in set (0.00 sec)
```

✍ **SUBSTRING_INDEX(**string, pembatas, n**)**

Fungsi ini akan memotong (atau menampilkan) string yang berada sebelum posisi *pembatas* yang *k e-n*. Bila nilai ke-n positif, maka semua string yang berada di sebelah kiri pembatas ke-n akan ditampilkan. Bila nilai ke-n negatif, maka semua string yang berada di sebelah kanan pembatas ke-n akan ditampilkan.

Contoh:

```
mysql> SELECT SUBSTRING_INDEX('www.coba-coba.com','.',1) ;
+-------------------------------------------+
| SUBSTRING_INDEX('www.coba-coba.com','.',1) |
+-------------------------------------------+
| www                                       |
+-------------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT SUBSTRING_INDEX('www.coba-coba.com','.',-1) ;
+-------------------------------------------- +
| SUBSTRING_INDEX('www.coba-coba.com','.',-1) |
+-------------------------------------------- +
| com                                        |
+-------------------------------------------- +
1 row in set (0.00 sec)
```

✍ **TRIM([[LEADING | TRAILING | BOTH] [trim_string] FROM string)**

Fungsi ini akan memotong/menghilangkan karakter tertentu/spasi kosong pada posisi tertentu dari suatu string. Pilihan LEADING akan memotong pada posisi di depan string. Pilihan TRAILING akan memotong pada posisi di belakang string. Pilihan BOTH akan memotong pada posisi di depan dan di belakang string. Bila pilihan tidak disebutkan, maka akan sama dengan pilihan BOTH.

Contoh:

```
mysql> SELECT TRIM('*' FROM '***Bintang kecil***') ;
+-------------------------------------+
| TRIM('*' FROM '***Bintang kecil***') |
+-------------------------------------+
| Bintang kecil                       |
+-------------------------------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT TRIM(BOTH '*' FROM '***Bintang kecil***') ;
+-----------------------------------------+
| TRIM(BOTH '*' FROM '***Bintang kecil***') |
+-----------------------------------------+
| Bintang kecil                           |
+-----------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT TRIM(LEADING '*' FROM '***Bintang kecil***') ;
+-------------------------------------------+
| TRIM(LEADING '*' FROM '***Bintang kecil***') |
+-------------------------------------------+
| Bintang kecil***                          |
+-------------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT TRIM(TRAILING '*' FROM '***Bintang kecil***') ;
+-------------------------------------------+
| TRIM(TRAILING '*' FROM '***Bintang kecil***') |
+-------------------------------------------+
| ***Bintang kecil                          |
+-------------------------------------------+
1 row in set (0.00 sec)
```

- **UCASE(string) atau UPPER(string)**

Fungsi ini akan mengubah penulisan string menjadi huruf besar. Kecuali pada NULL, yang akan menghasilkan NULL juga. Fungsi UCASE( ) identik dengan fungsi UPPER( ).

Contoh:

```
mysql> SELECT UCASE("warna merah"), UCASE('Merah'), UCASE(NULL) ;
+----------------------+----------------+-------------+
| UCASE("warna merah") | UCASE('Merah') | UCASE(null) |
+----------------------+----------------+-------------+
| WARNA MERAH          | MERAH          | NULL        |
+----------------------+----------------+-------------+
1 row in set (0.00 sec)
```

- **UPPER(string)**

Fungsi ini akan mengubah penulisan string menjadi huruf besar. Kecuali pada NULL, yang akan menghasilkan NULL juga. Fungsi UPPER( ) identik dengan fungsi UCASE( ).

Contoh:

```
mysql> SELECT UPPER("warna merah"), UPPER('Merah'), UPPER(NULL) ;
+----------------------+----------------+-------------+
| UPPER("warna merah") | UPPER('Merah') | UPPER(null) |
+----------------------+----------------+-------------+
| WARNA MERAH          | MERAH          | NULL        |
+----------------------+----------------+-------------+
```

Fungsi Numerik (Numeric Functions)

- **ABS (x)**

Fungsi ini akan menghasilkan nilai mutlak (absolut) dari suatu nilai x, atau bisa dikatakan juga akan

mengubah suatu bilangan menjadi jenis UNSIGNED.

Contoh:

```
mysql> SELECT ABS(-1), ABS(2), ABS(-2.3), ABS( -101.4) ;
+---------+--------+------------+-------------+
| ABS(-1) | ABS(2) | ABS(-2.3)) | ABS(-101.4) |
+---------+--------+------------+-------------+
|       1 |      2 |        2.3 |       101.4 |
+---------+--------+------------+-------------+
1 row in set (0.00 sec)
```

- **ACOS(x)**

Fungsi ini akan menghasilkan nilai arccosinus dari suatu nilai x, dengan kisaran nilai x antara 1 dan -1. Di luar kisaran itu akan menghasilkan NULL.

Contoh:

```
mysql> SELECT ACOS(1), ACOS(2), ACOS(0), ACOS( -1), ACOS(0.5) ;
+----------+---------+----------+----------+-----------+
| ACOS(1)  | ACOS(2) | ACOS(0)  | ACOS(-1) | ACOS(0.5) |
+----------+---------+----------+----------+-----------+
| 0.000000 |    NULL | 1.570796 | 3.141593 | 1.047198  |
+----------+---------+----------+----------+-----------+
1 row in set (0.00 sec)
```

- **ASIN(x)**

Fungsi ini akan menghasilkan nilai arcsinus dari suatu nilai x, dengan kisaran nilai x antara 1 dan -1. Diluar kisaran itu, akan menghasilkan NULL.

Contoh:

```
mysql> SELECT ASIN(1), ASIN(2), ASIN(0), ASIN( -1), ASIN(0.5) ;
+----------+---------+----------+-----------+-----------+
| ASIN(1)  | ASIN(2) | ASIN(0)  | ASIN(-1)  | ASIN(0.5) |
+----------+---------+----------+-----------+-----------+
| 1.570796 |    NULL | 0.000000 | -1.570796 | 0.523599  |
+----------+---------+----------+-----------+-----------+
1 row in set (0.00 sec)
```

- **ATAN(x)**

Fungsi ini akan menghasilkan nilai arctangen suatu nilai x, dengan kisaran nilai x antara 1 dan -1. Di luar kisaran itu akan menghasilkan NULL.

Contoh:

```
mysql> SELECT ATAN(1), ATAN(2), ATAN(0), ATAN( -1), ATAN(0.5) ;
+----------+---------+----------+-----------+-----------+
| ATAN(1)  | ATAN(2) | ATAN(0)  | ATAN(-1)  | ATAN(0.5) |
+----------+---------+----------+-----------+-----------+
| 0.785398 |    NULL | 0.000000 | -0.785398 | 0.463648  |
+----------+---------+----------+-----------+-----------+
1 row in set (0.00 sec)
```

- **ATAN2 (x, y)**

Fungsi ini hampir sama dengan ATAN( ), hanya saja menggunakan parameter x dan y untuk menentukan kuadrannya, dengan kisaran nilai x dan y antara 1 dan -1. Di luar kisaran itu akan menghasilkan NULL.

Contoh:

```
mysql> SELECT ATAN2(1,1), ATAN2(1,-1), ATAN2(-1,1), ATAN2(-1,-1) ;
+------------+-------------+-------------+--------------+
| ATAN2(1,1) | ATAN2(1,-1) | ATAN2(-1,1) | ATAN2(-1,-1) |
+------------+-------------+-------------+--------------+
| 0.785398   | 2.356194    | -0.785398   | -2.356194    |
+------------+-------------+-------------+--------------+
1 row in set (0.00 sec)
```

- **CEILING(x)**

Fungsi ini menghasilkan nilai bilangan bulat yang tidak lebih kecil dari nilai x, atau pembulatan bilangan ke nilai di atasnya.

Contoh:

```
mysql> SELECT CEILING(4.5), CEILING(2.3), CEILING(7.8) ;
+--------------+--------------+--------------+
| CEILING(4.5) | CEILING(2.3) | CEILING(7.8) |
+--------------+--------------+--------------+
|            5 |            3 |            8 |
+--------------+--------------+--------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT CEILING(-4.5), CEILING(-2.3), CEILING(-7.8) ;
+---------------+---------------+---------------+
| CEILING(-4.5) | CEILING(-2.3) | CEILING(-7.8) |
+---------------+---------------+---------------+
|            -4 |            -2 |            -7 |
+---------------+---------------+---------------+
1 row in set (0.00 sec)
```

- **COS(x)**

Fungsi ini akan menghasilkan nilai cosinus dari suatu nilai x, dengan kisaran nilai x antara 1 dan -1. Di luar kisaran itu akan menghasilkan NULL.

Contoh:

```
mysql> SELECT COS(1), COS(2), COS(0), COS(-1), COS(0.5) ;
+----------+-----------+----------+----------+-----------+
| COS(1)   | COS(2)    | COS(0)   | COS(-1)  | COS(0.5)  |
+----------+-----------+----------+----------+-----------+
| 0.540302 | -0.416147 | 1.000000 | 0.540302 | 0.877583  |
+----------+-----------+----------+----------+-----------+
1 row in set (0.00 sec)
```

- **COT(x)**

Fungsi ini akan menghasilkan nilai cotangen dari suatu nilai x, di mana nilai x menggunakan satuan radian.

Contoh:

```
mysql> SELECT COT(PI()/2), COT(PI()/3), COT(PI()/4) ;
+-------------+-------------+-------------+
| COT(PI()/2) | COT(PI()/3) | COT(PI()/4) |
+-------------+-------------+-------------+
| 0.00000000  | 0.57735027  | 1.00000000  |
+-------------+-------------+-------------+
1 row in set (0.00 sec)
```

- **DEGREES(x)**

Fungsi ini akan menghasilkan nilai derajat dari suatu nilai x, di mana nilai x menggunakan satuan radian.

Contoh:

```
mysql> SELECT DEGREES(PI()), DEGREES(PI()/2),
   -> DEGREES(PI()/3), DEGREES(PI()/4) ;
+---------------+-----------------+-----------------+-----------------+
| DEGREES(PI()) | DEGREES(PI()/2) | DEGREES(PI()/3) | DEGREES(PI()/4) |
+---------------+-----------------+-----------------+-----------------+
|           180 |              90 |              60 |              45 |
+---------------+-----------------+-----------------+-----------------+
1 row in set (0.00 sec)

mysql> SELECT DEGREES(-PI()), DEGREES(PI()*2),
   -> DEGREES(-PI()/2), DEGREES(-PI()/4) ;
+----------------+-----------------+------------------+-----------------+
| DEGREES(-PI()) | DEGREES(PI()*2) | DEGREES(-PI()/3) | DEGREES(-PI()/4)|
+----------------+-----------------+------------------+-----------------+
|           -180 |             360 |              -60 |             -45 |
+----------------+-----------------+------------------+-----------------+
1 row in set (0.00 sec)
```

- **EXP(x)**

Fungsi ini akan menghasilkan nilai eksponential dari suatu nilai x (e^x), di mana e adalah bilangan logaritma natural.

Contoh:

```
mysql> SELECT EXP(0), EXP(0.5), EXP(1), EXP(2) ;
+----------+----------+----------+----------+
| EXP(0)   | EXP(0.5) | EXP(1)   | EXP(2)   |
+----------+----------+----------+----------+
| 1.000000 | 1.648721 | 2.718282 | 7.389056 |
+----------+----------+----------+----------+
1 row in set (0.00 sec)

mysql> SELECT EXP(-0.5), EXP(-1), 1/EXP(1), 1/EXP(-1) ;
+-----------+----------+------------+------------+
| EXP(-0.5) | EXP(-1)  | 1/EXP(1)   | 1/EXP(-1)  |
+-----------+----------+------------+------------+
| 0.606531  | 0.367879 | 0.36787944 | 2.71828183 |
+-----------+----------+------------+------------+
1 row in set (0.00 sec)
```

- **FLOOR(x)**

Fungsi ini menghasilkan nilai bilangan bulat yang tidak lebih besar dari nilai x, atau pembulatan bilangan ke nilai di bawahnya. Fungsi FLOOR( ) merupakan kebalikan dari fungsi CEILING( ).

Contoh:

```
mysql> SELECT FLOOR(4.5), FLOOR(2.3), FLOOR(7.8) ;
+------------+------------+------------+
| FLOOR(4.5) | FLOOR(2.3) | FLOOR(7.8) |
+------------+------------+------------+
|          4 |          2 |          7 |
+------------+------------+------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT FLOOR(-4.5), FLOOR(-2.3), FLOOR(-7.8) ;
+-------------+-------------+-------------+
| FLOOR(-4.5) | FLOOR(-2.3) | FLOOR(-7.8) |
+-------------+-------------+-------------+
|          -5 |          -3 |          -8 |
+-------------+-------------+-------------+
1 row in set (0.00 sec)
```

✍ **LOG(x)**

Fungsi ini akan menghasilkan nilai logaritma dari suatu nilai x (bilangan basis e).

Contoh:

```
mysql> SELECT LOG(0), LOG(0.5), LOG(1), LOG(2) ;
+--------+-----------+----------+----------+
| LOG(0) | LOG(0.5)  | LOG(1)   | LOG 2)   |
+--------+-----------+----------+----------+
| NULL   | -0.693147 | 0.000000 | 0.693147 |
+--------+-----------+----------+----------+
1 row in set (0.00 sec)
```

✍ **LOG10(x)**

Fungsi ini akan menghasilkan nilai logaritma dari suatu nilai x ke bilangan basis 10.

Contoh:

```
mysql> SELECT LOG10(0), LOG10(0.5), LOG10(1), LOG10(10) ;
+----------+-----------+----------+-----------+-----------+
| LOG10(0) | LOG10(0.5) | LOG10(1) | LOG10(10) | LOG10(100) |
+----------+-----------+----------+-----------+-----------+
| NULL     | -0.301030  | 0.000000 | 1.000000  | 2.000000   |
+----------+-----------+----------+-----------+-----------+
1 row in set (0.00 sec)
```

```
mysql> SELECT LOG10(1000), LOG10(-0.5), LOG10( -1), LOG10(-10) ;
+-------------+-------------+-----------+------------+
| LOG10(1000) | LOG10(-0.5) | LOG10(-1) | LOG10(-10) |
+-------------+-------------+-----------+------------+
| 3.000000    | NULL        | NULL      | NULL       |
+-------------+-------------+-----------+------------+
1 row in set (0.00 sec)
```

✍ **MOD**(m, n)

Fungsi ini akan menghasilkan nilai hasil bagi (modulus) dari bilangan m dengan bilangan n. Fungsi MOD(m,n) hasilnya sama dengan operator **%** (modulus).

Contoh:

```
mysql> SELECT MOD(10,2), MOD(7,3), MOD(11,3), MOD(3,0) ;
+-----------+----------+-----------+----------+
| MOD(10,2) | MOD(7,3) | MOD(11,3) | MOD(3,0) |
+-----------+----------+-----------+----------+
|         0 |        1 |         2 |  NULL    |
+-----------+----------+-----------+----------+
1 row in set (0.00 sec)
```

✍ **PI( )**

Fungsi ini akan menghasilkan nilai dari ? .

Contoh:

```
mysql> SELECT PI( );
```

```
+-----------+
| PI( )     |
+-----------+
| 3.141593  |
+-----------+
1 row in set (0.00 sec)
```

- **POW(x, y)**

Fungsi ini akan menghasilkan nilai hasil pangkat bilangan y terhadap x.

Contoh:

```
mysql> SELECT POW(2,2), POW(2,3), POW(10,2), POW(3,3) ;
+----------+----------+------------+-----------+
| POW(2,2) | POW(2,3) | POW(10,2)  | POW(3,3)  |
+----------+----------+------------+-----------+
| 4.000000 | 8.000000 | 100.000000 | 27.000000 |
+----------+----------+------------+-----------+
1 row in set (0.00 sec)
```

- **POWER (x, y)**

Fungsi ini akan menghasilkan nilai hasil pangkat bilangan y terhadap x. Fungsi POWER( ) ini sama dengan fungsi POW( ) di atas.

Contoh:

```
mysql> SELECT POWER(2,2), POWER(2,3), POWER(10,2), POWER(3,3) ;
+------------+------------+-------------+------------+
| POWER(2,2) | POWER(2,3) | POWER(10,2) | POWER(3,3) |
+------------+------------+-------------+------------+
| 4.000000   | 8.000000   | 100.000000  | 9.000000   |
+------------+------------+-------------+------------+
1 row in set (0.00 sec)
```

- **RADIANS(x)**

Fungsi ini akan menghasilkan nilai radians dari bilangan x yang berasal dari satuan derajat.

Contoh:

```
mysql> SELECT RADIAN(0), RADIAN(90), RADIAN(180), RADIANS(360) ;
+-----------+-----------+------------+--------------+
| RADIAN(0) | RADIAN(90) | RADIAN(180) | RADIANS(360) |
+-----------+-----------+------------+--------------+
| 0         | 1.570796   | 3.141593    | 6.283185     |
+-----------+-----------+------------+--------------+
1 row in set (0.00 sec)
```

- **RAND( ) atau RAND(**numerik**)**

Fungsi RAND( ) adalah menghasilkan bilangan pecahan secara acak dari kisaran nilai 0.0 sampai dengan 1.0. Sedangkan fungsi RAND(numerik) hampir sama dengan fungsi RAND( ). Perbedaannya fungsi RAND(numerik) akan menggunakan bilangan n sebagai acuannya, dan hasil acaknya akan tetap sama. Perhatikan perbedaannya pada contoh di bawah ini.

Contoh:

```
mysql> SELECT RAND( ), RAND( ), RAND( ), RAND( ) ;
+----------+----------+----------+----------+
| RAND( )  | RAND( )  | RAND( )  | RAND( )  |
+----------+----------+----------+----------+
| 0.523334 | 0.391736 | 0.388675 | 0.768166 |
+----------+----------+----------+----------+
1 row in set (0.00 sec)
```

mysql> SELECT RAND(1), RAND(2), RAND(3), RAND(5), RAND(10) ;

```
+----------+----------+----------+----------+----------+
| RAND(1)  | RAND(2)  | RAND(3)  | RAND(5)  | RAND(10) |
+----------+----------+----------+----------+----------+
| 0.181090 | 0.181090 | 0.181090 | 0.181090 | 0.181090 |
+----------+----------+----------+----------+----------+
```

1 row in set (0.00 sec)

- **ROUND(x)** atau **ROUND(x, d)**

Fungsi ROUND(x) akan menghasilkan pembulatan bilangan bulat x. Sedangkan fungsi ROUND(x,d) akan menghasilkan pembulatan bilangan bulat x dengan d desimal.

Contoh:

mysql> SELECT ROUND(3.3), ROUND(2.5), ROUND(5.8), ROUND(-7.3) ;

```
+------------+------------+------------+-------------+
| ROUND(3.3) | ROUND(2.5) | ROUND(5.8) | ROUND(-7.3) |
+------------+------------+------------+-------------+
|          3 |          2 |          6 |          -7 |
+------------+------------+------------+-------------+
```

1 row in set (0.00 sec)

mysql> SELECT ROUND(3.3, 2), ROUND(2.5, 3), ROUND(5.8,0), ROUND(-7.3, 1);

```
+---------------+---------------+--------------+----------------+
| ROUND(3.3, 2) | ROUND(2.5, 3) | ROUND(5.8,0) | ROUND(-7.3, 1) |
+---------------+---------------+--------------+----------------+
|          3.30 |         2.500 |            6 |           -7.3 |
+---------------+---------------+--------------+----------------+
```

1 row in set (0.00 sec)

- **SIGN(x)**

Fungsi ini akan menghasilkan 1 bila x adalah bilangan positif, 0 bila x adalah 0, -1 bila x adalah bilangan negatif.

Contoh:

mysql> SELECT SIGN(0), SIGN(5), SIGN(-10), SIGN(10) ;

```
+---------+---------+-----------+----------+
| SIGN(0) | SIGN(5) | SIGN(-10) | SIGN(10) |
+---------+---------+-----------+----------+
|       0 |       1 |        -1 |        1 |
+---------+---------+-----------+----------+
```

1 row in set (0.00 sec)

- **SIN(x)**

Fungsi ini akan menghasilkan nilai sinus dari bilangan x yang menggunakan satuan radian.

Contoh:

mysql> SELECT SIN(PI()), SIN(0), SIN(PI()/2), SIN(PI()*2);

```
+----------------------+--------+-------------+-----------------------+
| SIN(PI())            | SIN(0) | SIN(PI()/2) | SIN(PI()*2)           |
+----------------------+--------+-------------+-----------------------+
| 1.2246063538224e-016 |      0 |           1 | -2.4492127076448e-016 |
+----------------------+--------+-------------+-----------------------+
```

1 row in set (0.11 sec)

- **SQRT(x)**

Fungsi ini akan menghasilkan nilai akar pangkat dari bilangan positif x.

Contoh:

```
mysql> SELECT SQRT(25), SQRT(100), SQRT(-25), SQRT(400) ;
+----------+-----------+-----------+-----------+
| SQRT(25) | SQRT(100) | SQRT(-25) | SQRT(400) |
+----------+-----------+-----------+-----------+
| 5.000000 | 10.000000 |      NULL | 20.000000 |
+----------+-----------+-----------+-----------+
1 row in set (0.00 sec)
```

- **TAN(x)**

Fungsi ini akan menghasilkan nilai tangen dari bilangan x yang menggunakan satuan radian.

Contoh:

```
mysql> SELECT TAN(PI()),TAN(0),TAN(PI()/4),TAN(PI()*2);
+-----------------------+--------+-------------+-----------------------+
| TAN(PI())             | TAN(0) | TAN(PI()/4) | TAN(PI()*2)           |
+-----------------------+--------+-------------+-----------------------+
| -1.2246063538224e-016 |      0 |           1 | -2.4492127076448e-016 |
+-----------------------+--------+-------------+-----------------------+
1 row in set (0.00 sec)
```

- **TRUNCATE(x, d)**

Fungsi ini akan menghasilkan pemotongan tampilan suatu bilangan x dengan d desimal. Tidak ada perlakuan pembulatan bilangan di sini

Contoh:

```
mysql> SELECT TRUNCATE(4.567,0), truncate(4.567,1), truncate(4.567,2) ;
+-------------------+-------------------+-------------------+
| TRUNCATE(4.567,0) | TRUNCATE(4.567,1) | TRUNCATE(4.567,2) |
+-------------------+-------------------+-------------------+
|                 4 |               4.5 |              4.56 |
+-------------------+-------------------+-------------------+
1 row in set (0.00 sec)

mysql> SELECT TRUNCATE(4.567,4), truncate(4.567,-1), truncate(4.567,-2) ;
+-------------------+--------------------+--------------------+
| TRUNCATE(4.567,4) | TRUNCATE(4.567,-1) | TRUNCATE(4.567,-2) |
+-------------------+--------------------+--------------------+
|            4.5670 |                  0 |                  0 |
+-------------------+--------------------+--------------------+
1 row in set (0.00 sec)
```

Fungsi Tanggal dan Waktu (Date and Time Functions)

- **CURDATE( )**

Fungsi ini akan menghasilkan tanggal sistem/komputer saat ini, dengan format tampilan 'yyyy-mm-dd' atau 'yyyymmdd'.

Contoh:

```
mysql> SELECT CURDATE( ), CURDATE( )+1, CURDATE( )-1, CURDATE( )+0 ;
+------------+-------------+-------------+-------------+
| CURDATE( ) | CURDATE()+1 | CURDATE()-1 | CURDATE()+0 |
+------------+-------------+-------------+-------------+
| 2003-05-03 |    20030504 |    20030502 |    20030503 |
+------------+-------------+-------------+-------------+
1 row in set (0.00 sec)
```

✍ **CURRENT_DATE**

Fungsi ini akan menghasilkan tanggal sistem saat ini, dengan format tampilan 'yyyy-mm-dd' atau 'yyyymmdd'. Fungsi CURRENT_DATE sama dengan fungsi CURDATE( ), hanya saja tidak menggunakan tanda kurung seperti layaknya format penulisan fungsi lainnya, dan tidak bisa melakukan proses matematis penambahan dan pengurangan tanggal
seperti pada contoh fungsi CURDATE( ).

Contoh:

```
mysql> SELECT CURRENT_DATE ;
+---------------+
| CURRENT_DATE  |
+---------------+
| 2003-05-03    |
+---------------+
1 row in set (0.00 sec)

mysql> SELECT CURRENT_DATE+1 ;
ERROR 1064: You have an error in your SQL syntax near '+1' at line 1
```

✍ **CURRENT_TIME**

Fungsi ini akan menghasilkan jam sistem saat ini, dengan format tampilan 'hh-mm-ss' atau 'hhmmss'. Fungsi CURRENT_TIME sama dengan fungsi CURTIME( ), hanya saja tidak menggunakan tanda kurung seperti layaknya format penulisan fungsi lainnya, dan tidak bisa melakukan proses matematis penambahan dan pengurangan jam seperti pada contoh fungsi CURTIME( ).

Contoh:

```
mysql> SELECT CURRENT_TIME ;
+---------------+
| CURRENT_TIME  |
+---------------+
| 12:57:36      |
+---------------+
1 row in set (0.00 sec)

mysql> SELECT CURRENT_TIME+1 ;
ERROR 1064: You have an error in your SQL syntax near '+1' at line 1
```

✍ **CURRENT_TIMESTAMP**

Fungsi ini akan menghasilkan tanggal dan jam sistem saat ini, dengan format tampilan 'yyyy-mm-dd hh-mm-ss'. Fungsi CURRENT_TIMESTAMP sama dengan fungsi NOW( ), hanya saja tidak menggunakan tanda kurung seperti layaknya format penulisan fungsi lainnya, dan tidak bisa melakukan proses matematis penambahan dan pengurangan jam seperti pada contoh fungsi NOW( ).

Contoh:

```
mysql> SELECT CURRENT_TIMESTAMP ;
+---------------------+
| CURRENT_TIMESTAMP   |
+---------------------+
| 2003-05-03 12:57:36 |
```

```
+---------------------+
```
1 row in set (0.00 sec)

```
mysql> SELECT CURRENT_TIMESTAMP+1 ;
ERROR 1064: You have an error in your SQL syntax near '+1' at line 1
```

✍ **CURTIME( )**

Fungsi ini akan menghasilkan jam sistem saat ini, dengan format tampilan 'hh-mm-ss' atau 'hhmmss'.

Contoh:

```
mysql> SELECT CURTIME( ), CURTIME( )+10, CURTIME( ) -10, CURTIME( )+0 ;
+------------+-------------+-------------+-------------+
| CURTIME( )  | CURTIME( )+10 | CURTIME( )-10 | CURTIME( )+0 |
+------------+-------------+-------------+-------------+
| 12:57:36   |       125746 |       125756 |      125736 |
+------------+-------------+-------------+-------------+
```
1 row in set (0.00 sec)

✍ **DATE_ADD**(tanggal, **INTERVAL** ekspresi**)**

Fungsi ini akan menghasilkan penjumlahan dari tanggal yang ditentukan. Tanggal ditulis dengan format 'yyyy-mm-dd', 'yyyymmdd' atau 'yymmdd'. Sedangkan ekspresi yang digunakan adalah:

EKSPRESI	
SECOND	**:** Untuk satuan detik
MINUTE	**:** Untuk satuan menit
HOUR	**:** Untuk satuan jam
DAY	**:** Untuk satuan hari
MONTH	**:** Untuk satuan bulan
YEAR	**:** Untuk satuan tahun
MINUTE_SECOND	**:** Untuk satuan menit dan detik
HOUR_MINUTE	**:** Untuk satuan jam dan menit
DAY_HOUR	**:** Untuk satuan hari dan jam
YEAR_MONTH	**:** Untuk satuan tahun dan bulan
HOUR_SECOND	**:** Untuk satuan jam dan detik
DAY_MINUTE	**:** Untuk satuan hari dan menit
DAY_SECOND	**:** Untuk satuan hari dan detik

Contoh:

```
mysql> SELECT DATE_ADD('2003-05-01 11:59:59', INTERVAL 1 SECOND) ;
+-----------------------------------------------------+
| DATE_ADD('2003-05-01 11:59:59', INTERVAL 1 SECOND) |
+-----------------------------------------------------+
| 2003-05-01 12:00:00                                 |
+-----------------------------------------------------+
```
1 row in set (0.00 sec)

```
mysql> SELECT DATE_ADD('2003-05-01 11:50:50', INTERVAL 10 SECOND) ;
+-----------------------------------------------------+
| DATE_ADD('2003-05-01 11:50:50', INTERVAL 10 SECOND) |
+-----------------------------------------------------+
| 2003-05-01 11:51:00                                 |
+-----------------------------------------------------+
```
1 row in set (0.00 sec)

```
mysql> SELECT DATE_ADD('2003-05-01 12:00:00', INTERVAL 10 MINUTE) ;
+-----------------------------------------------------+
| DATE_ADD('2003-05-01 12:00:00', INTERVAL 10 MINUTE) |
+-----------------------------------------------------+
| 2003-05-01 12:10:00                                 |
+-----------------------------------------------------+
1 row in set (0.00 sec)


mysql> SELECT DATE_ADD('2003-05-01 12:00:00', INTERVAL 1 HOUR) ;
+----------------------------------------------------+
| DATE_ADD('2003-05-01 12:00:00', INTERVAL 1 HOUR) |
+----------------------------------------------------+
| 2003-05-01 13:00:00                                |
+----------------------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT DATE_ADD('2003-05-01 12:00:00', INTERVAL 1 DAY) ;
+---------------------------------------------------+
| DATE_ADD('2003-05-01 12:00:00', INTERVAL 1 DAY) |
+---------------------------------------------------+
| 2003-05-02 12:00:00                               |
+---------------------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT DATE_ADD('2003-05-01 12:00:00', INTERVAL 3 MONTH) ;
+-----------------------------------------------------+
| DATE_ADD('2003-05-01 12:00:00', INTERVAL 3 MONTH) |
+-----------------------------------------------------+
| 2003-08-01 12:00:00                                 |
+-----------------------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT DATE_ADD('2003-05-01 12:00:00', INTERVAL 1 YEAR) ;
+----------------------------------------------------+
| DATE_ADD('2003-05-01 12:00:00', INTERVAL 1 YEAR) |
+----------------------------------------------------+
| 2004-05-01 12:00:00                                |
+----------------------------------------------------+
1 row in set (0.00 sec)


mysql> SELECT DATE_ADD('2003-05-01 12:00:00',
   -> INTERVAL '5:5' MINUTE_SECOND) ;
+-------------------------------------------------------------+
| DATE_ADD('2003-05-01 12:00:00', INTERVAL '5:5' MINUTE_SECOND) |
+-------------------------------------------------------------+
| 2003-05-01 12:05:05                                         |
+-------------------------------------------------------------+
1 row in set (0.00 sec)
mysql> SELECT DATE_ADD('2003-05-01 12:00:00',
   -> INTERVAL '5:5' HOUR_MINUTE) ;
+-----------------------------------------------------------+
| DATE_ADD('2003-05-01 12:00:00', INTERVAL '5:5' HOUR_MINUTE) |
+-----------------------------------------------------------+
| 2003-05-01 17:05:00                                       |
+-----------------------------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT DATE_ADD('2003-05-01 12:00:00',
   -> INTERVAL '5:0:5' HOUR_SECOND) ;
+-------------------------------------------------------------+
| DATE_ADD('2003-05-01 12:00:00', INTERVAL '5:0:5' HOUR_SECOND) |
+-------------------------------------------------------------+
| 2003-05-01 17:00:05                                         |
+-------------------------------------------------------------+
1 row in set (0.00 sec)


mysql> SELECT DATE_ADD('2003-05-01 12:00:00',
```

```
   -> INTERVAL '5 5' DAY_HOUR) ;
+-----------------------------------------------------------+
| DATE_ADD('2003-05-01 12:00:00', INTERVAL '5 5' DAY_HOUR) |
+-----------------------------------------------------------+
| 2003-05-06 17:00:00                                       |
+-----------------------------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT DATE_ADD('2003-05-01 12:00:00',
   -> INTERVAL '5 0:5' DAY_MINUTE) ;
+-------------------------------------------------------------+
| DATE_ADD('2003-05-01 12:00:00', INTERVAL '5 0:5' DAY_MINUTE) |
+-------------------------------------------------------------+
| 2003-05-06 12:05:00                                         |
+-------------------------------------------------------------+
1 row in set (0.00 sec)

 mysql> SELECT DATE_ADD('2003-05-01 12:00:00',
   -> INTERVAL '5 0:0:5' DAY_SECOND) ;
+--------------------------------------------------------------+
| DATE_ADD('2003-05-01 12:00:00', INTERVAL '5 0:0:5' DAY_SECOND) |
+--------------------------------------------------------------+
| 2003-05-06 12:00:05                                          |
+--------------------------------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT DATE_ADD('2003-05-01 12:00:00',
   -> INTERVAL '5 5' YEAR_MONTH) ;
+-----------------------------------------------------------+
| DATE_ADD('2003-05-01 12:00:00', INTERVAL '5 5' YEAR_MONTH) |
+-----------------------------------------------------------+
| 2008-10-01 12:00:00                                       |
+-----------------------------------------------------------+
1 row in set (0.00 sec)
```

✍ **DATE_FORMAT**(tanggal, format)

Fungsi ini akan mengganti format tampilan tanggal yang ditentukan sesuai dengan kriteria format yang diberikan. Bentuk standar DATE biasanya ditulis dengan format 'yyyy-mm-dd', 'yyyymmdd' atau 'yymmdd'. Dengan adanya fungsi DATE_FORMAT ini kita bisa mengubahnya sesuai dengan kebutuhan kita.

KODE FORMAT

%M : Nama bulan seperti January, February dan seterusnya.
%W : Nama hari seperti Sunday, Monday dan seterusnya.
%D : Urutan bulan seperti 1st untuk bulan Januari, 3rd untuk bulan Maret dan seterusnya.
%Y : Tahun dengan penulisan 4 digit. Tahun 2002 ditulis 2002 dan seterusnya.
%y : Tahun dengan penulisan 2 digit. Tahun 2002 ditulis dengan 02 dan seterusnya.
%X : Urutan minggu dalam setahun, di mana perhitungannya dimulai dari hari Minggu (Sunday). Kisaran nilai dari 01 sampai dengan 53. Digunakan bersamaan dengan '%V'
%x : Urutan minggu dalam setahun, di mana perhitungannya dimulai dari hari Senin (Monday). Kisaran nilai dari 01 sampai dengan 53. Digunakan bersamaan dengan '%v'
%a : Singkatan nama hari. Seperti Sunday ditulis dengan Sun, Wednesday ditulis dengan Wed dan sebagainya.
%d : Urutan tanggal pada bulan dengan penulisan 2 digit. Seperti 01, 05, 21, 31, dan sebagainya.
%e : Urutan tanggal pada bulan dengan penulisan 1 digit. Seperti 1, 5, 21, 31, dan sebagainya.
%m : Urutan bulan dengan penulisan 2 digit. Seperti 01 untuk Januari, 05 untuk Mei, 10 untuk Oktober dan sebagainya.
%c : Urutan bulan dengan penulisan 1 digit. Seperti 1 untuk Januari, 5 untuk Mei, 10 untuk Oktober, dan sebagainya.
%b : Singkatan nama bulan. Seperti Jan untuk January, Dec untuk December, Aug untuk August, dan sebagainya.
%j : Urutan hari dalam setahun. 1 Januari ditulis 001, 2 Januari ditulis 002, 31 Desember ditulis 366, dan sebagainya.
%H : Urutan jam dengan penulisan 2 digit. Seperti 00 sampai dengan 23.
%k : Urutan jam dengan penulisan 1 digit. Seperti 0 sampai dengan 23.

%h : Urutan jam dengan penulisan 2 digit. Seperti 01 sampai dengan 12.
%I : Urutan jam dengan penulisan 2 digit. Seperti 01 sampai dengan 12.
%l : Urutan jam dengan penulisan 1 digit. Seperti 1 sampai dengan 12.

%i : Urutan menit dengan penulisan 2 digit. Seperti 00 sampai dengan 59.
%r **:** Urutan jam dalam bentuk penulisan 12 jam-an, (hh:mm:ss [AM | PM])
%T **:** Urutan jam dalam bentuk penulisan 24 jam-an. (hh:mm:ss)
%S **:** Urutan detik dalam penulisan 2 digit, dari 00 sampai dengan 59.
%p **:** Untuk penulisan kode AM atau PM pada jam
%w **:** Nomor urut hari yang dimulai dari hari Senin (0=Sunday…6=Saturday).
%U : Urutan minggu dalam setahun, yang perhitungannya dimulai dari hari Minggu (Sunday). Kisaran nilai dari 00 sampai dengan 53.
%u : Urutan minggu dalam setahun, yang perhitungannya dimulai dari hari Senin (Monday). Kisaran nilai dari 00 sampai dengan 53.
%V : Urutan minggu dalam setahunnya, yang perhitungannya dimulai dari hari Minggu (Sunday). Kisaran nilai dari 01 sampai dengan 53. Digunakan bersamaan dengan '%X'
%v : Urutan minggu dalam setahunnya, yang perhitungannya dimulai dari hari Senin (Monday). Kisaran nilai dari 01 sampai dengan 53. Digunakan bersamaan dengan '%x'
%% : Bentuk literal untuk tanda %

Contoh:

```
mysql> SELECT DATE_FORMAT('2003-05-01 11:00:00', '%W, %d %M %Y, %T') ;
+------------------------------------------------------------+
| DATE_FORMAT('2003-05-01 11:00:00', '%W, %d %M %Y, %T') |
+------------------------------------------------------------+
| Wednesday, 01 January 2003, 11:00:00                       |
+------------------------------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT DATE_FORMAT('2003-05-01 11:00:00', '%a, %b %D, %Y, %r') ;
+---------------------------------------------------------+
| DATE_FORMAT('2003-05-01 11:00:00', '%a, %b %D, %Y, %r') |
+---------------------------------------------------------+
| Wed, Jan 1st, 2003, 11:00:00 AM                         |
+---------------------------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT DATE_FORMAT('2003-05-01 11:00:00', '%d, %m %Y') ;
+-----------------------------------------------+
| DATE_FORMAT('2003-05-01 11:00:00', '%d %m %Y') |
+-----------------------------------------------+
| 01 01 2003                                    |
+-----------------------------------------------+
1 row in set (0.00 sec)
```

✍ **DATE_SUB**(tanggal, **INTERVAL** ekspresi interval**)**

Fungsi ini akan menghasilkan pengurangan dari tanggal yang ditentukan. Tanggal ditulis dengan format ‘yyyy-mm-dd’, ‘yyyymmdd’ atau ‘yymmdd’. Sedangkan ekspresi yang digunakan dan contoh selengkapnya dapat dilihat pada fungsi DATE_ADD( ) di atas, dengan penyesuaian menggunakan fungsi DATE_SUB( ).

Contoh:

```
mysql> SELECT DATE_SUB('2003-05-01 12:00:00', INTERVAL 1 SECOND) ;
+-------------------------------------------------+
| DATE_SUB('2003-05-01 12:00:00', INTERVAL 1 SECOND) |
+-------------------------------------------------+
| 2003-05-01 11:59:59                             |
+-------------------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT DATE_SUB('2003-05-01 12:00:00', INTERVAL 10 SECOND) ;
+-------------------------------------------------+
| DATE_SUB('2003-05-01 12:00:00', INTERVAL 10 SECOND) |
+-------------------------------------------------+
| 2003-05-01 11:50:00                             |
+-------------------------------------------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT DATE_SUB('2003-05-01 12:00:00', INTERVAL 10 MINUTE) ;
+-----------------------------------------------------+
| DATE_SUB('2003-05-01 12:00:00', INTERVAL 10 MINUTE) |
+-----------------------------------------------------+
| 2003-05-01 11:50:00                                 |
+-----------------------------------------------------+
1 row in set (0.00 sec)
```

✍ **DAYNAME**(tanggal)

Fungsi ini akan menghasilkan nama hari dari tanggal yang ditentukan. Tanggal ditulis dengan format 'yyyy-mm-dd', 'yyyymmdd' atau 'yymmdd'.

Contoh:

```
mysql> SELECT DAYNAME('2003-01-01'), DAYNAME('20030224'),
  -> DAYNAME('030817') ;
+------------------------+--------------------+--------------------+
| DAYNAME('2003-01-01') | DAYNAME('20030224') | DAYNAME('030817') |
+------------------------+--------------------+---- ---------------+
| Wednesday              | Monday             | Sunday             |
+------------------------+--------------------+--------------------+
1 row in set (0.00 sec)
```

✍ **DAYOFMONTH**(tanggal)

Fungsi ini akan menghasilkan urutan tanggal dari tanggal yang ditentukan. Tanggal ditulis dengan format 'yyyy-mm-dd', 'yyyymmdd' atau 'yymmdd'.

Contoh:

```
mysql> SELECT DAYOFMONTH('2003-01-01'), DAYOFMONTH('20030224') ;
+--------------------------+-----------------------+
| DAYOFMONTH('2003-01-01') | DAYOFMONTH('20030224') |
+--------------------------+-----------------------+
|                        1 |                    24 |
+--------------------------+-----------------------+
1 row in set (0.00 sec)
```

✍ **DAYOFWEEK**(tanggal)

Fungsi ini akan menghasilkan urutan nama hari dari tanggal yang ditentukan. Tanggal ditulis dengan format 'yyyy-mm-dd', 'yyyymmdd' atau 'yymmdd'. Urutan pertama adalah untuk Minggu, urutan ke-2 untuk Senin dan seterusnya hingga urutan ke-7 untuk Sabtu.

Contoh:

```
mysql> SELECT DAYNAME('2003-01-01'), DAYOFWEEK('2003-01-01') ;
+------------------------+-------------------------+
| DAYNAME('2003-01-01') | DAYOFWEEK('2003-01-01') |
+------------------------+-------------------------+
| Wednesday              |                       4 |
+------------------------+-------------------------+
1 row in set (0.00 sec)

mysql> SELECT DAYNAME('20030224'), DAYOFWEEK('20030224') ;
+---------------------+-----------------------+
| DAYNAME('20030224') | DAYOFWEEK('20030224') |
+---------------------+-----------------------+
| Wednesday           |                     2 |
+---------------------+-----------------------+
1 row in set (0.00 sec)

mysql> SELECT DAYNAME('030503'), DAYOFWEEK('030503') ;
+--------------------+----------------------+
| DAYNAME('030503') | DAYOFWEEK('030503') |
+--------------------+----------------------+
| Saturday           |                    7 |
```

```
+------------------+--------------------+
1 row in set (0.00 sec)
```

✍ **DAYOFYEAR(**tanggal**)**

Fungsi ini akan menghasilkan urutan tanggal dari tanggal yang ditentukan. Tanggal ditulis dengan format 'yyyy-mm-dd', 'yyyymmdd' atau 'yymmdd'. Urutan pertama adalah untuk 1 Januari dan seterusnya hingga urutan (maksimum) 366 untuk 31 Desember pada tahun kabisat. Selain tahun kabisat urutan maksimum menjadi 365 untuk tanggal 31 Desember.

Contoh:

```
mysql> SELECT DAYOFYEAR('2003-01-01'), DAYOFYEAR('20030224') ;
+-------------------------+----------------------+
| DAYOFYEAR('2003-01-01') | DAYOFYEAR('20030224') |
+-------------------------+----------------------+
|                       1 |                   55 |
+-------------------------+----------------------+
1 row in set (0.00 sec)

mysql> SELECT DAYOFYEAR('2000-12-31'), DAYOFYEAR('2003-12-31') ;
+-------------------------+-------------------------+
| DAYOFYEAR('2000-12-31') | DAYOFYEAR('2003-12-31') |
+-------------------------+-------------------------+
|                     366 |                     365 |
+-------------------------+-------------------------+
1 row in set (0.00 sec)
```

✍ **EXTRACT(**interval **FROM** tanggal_waktu**)**

Fungsi ini akan memisahkan bagian-bagian dari tanggal sesuai dengan kriteria interval. Interval yang digunakan sama dengan pada fungsi DATE_ADD( ).

Contoh:

```
mysql> SELECT EXTRACT(SECOND FROM '2003-01-01 11:12:25') ;
+-------------------------------------------+
| EXTRACT(SECOND FROM '2003-01-01 11:12:25') |
+-------------------------------------------+
|                                        25 |
+-------------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT EXTRACT(MINUTE '2003-01-01 11:12:25') ;
+-------------------------------------------+
| EXTRACT(MINUTE FROM '2003-01-01 11:12:25') |
+-------------------------------------------+
|                                        12 |
+-------------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT EXTRACT(HOUR FROM '2003-01-01 11:12:25') ;
+-----------------------------------------+
| EXTRACT(HOUR FROM '2003-01-01 11:12:25') |
+-----------------------------------------+
|                                      11 |
+-----------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT EXTRACT(HOUR_MINUTE FROM '2003-01-01 11:12:25') ;
+-------------------------------------------------+
| EXTRACT(HOUR_MINUTE FROM '2003-01-01 11:12:25') |
+-------------------------------------------------+
|                                            1112 |
```

```
+-----------------------------------------------+
```
1 row in set (0.00 sec)

mysql> SELECT EXTRACT(DAY FROM '2003-05-01 11:12:25') ;
```
+---------------------------------------+
| EXTRACT(DAY FROM '2003-05-01 11:12:25') |
+---------------------------------------+
|                                     1 |
+---------------------------------------+
```
1 row in set (0.00 sec)

mysql> SELECT EXTRACT(MONTH FROM '2003-05-01 11:12:25') ;
```
+-----------------------------------------+
| EXTRACT(MONTH FROM '2003-05-01 11:12:25') |
+-----------------------------------------+
|                                      05 |
+-----------------------------------------+
```
1 row in set (0.00 sec)

mysql> SELECT EXTRACT(YEAR FROM '2003-05-01 11:12:25') ;
```
+----------------------------------------+
| EXTRACT(YEAR FROM '2003-05-01 11:12:25') |
+----------------------------------------+
|                                   2003 |
+----------------------------------------+
```
1 row in set (0.00 sec)

✍ **FROM_DAYS**(numerik**)**

Fungsi ini akan menghasilkan tanggal dari bilangan n yang diberikan. Anda bisa menggunakan bersamaan dengan fungsi TO_DAYS( ) untuk memeriksa kebenaran hasilnya.

Contoh:

mysql> SELECT TO_DAYS('2003-05-01') ;
```
+----------------------+
| TO_DAYS('2003-05-01') |
+----------------------+
|               731701 |
+----------------------+
```
1 row in set (0.00 sec)

mysql> SELECT FROM_DAYS(731701) ;
```
+-------------------+
| FROM_DAYS(731701) |
+-------------------+
| 2003-05-01        |
+-------------------+
```
1 row in set (0.00 sec)

✍ **FROM_UNIXTIME**(unix_timestamp, format)

Fungsi ini akan menghasilkan jumlah detik yang dihitung sejak '1970 -01-01 00:00:00' GMT hingga tanggal yang ditentukan pada argumen fungsi.

Contoh:

mysql> SELECT UNIX_TIMESTAMP('2003-01-01') ;
```
+-----------------------------+
| UNIX_TIMESTAMP('2003-01-01') |
+-----------------------------+
|                  1041354000 |
+-----------------------------+
```
1 row in set (0.06 sec)

mysql> SELECT FROM_UNIXTIME(1041354000) ;
```
+--------------------------+
| FROM_UNIXTIME(1041354000) |
+--------------------------+
| 2003-01-01 00:00:00      |
```

```
+--------------------------+
1 row in set (0.06 sec)
```

```
mysql> SELECT FROM_UNIXTIME(1041354000,'%d %M %Y') ;
+--------------------------------------+
| FROM_UNIXTIME(1041354000, '%d %M %Y') |
+--------------------------------------+
| 01 January 2003                      |
+--------------------------------------+
1 row in set (0.06 sec)
```

✍ **HOUR**(waktu)

Fungsi ini akan menghasilkan urutan jam dari jam yang ditentukan. Kisaran nilai yang dihasilkan antara 1 sampai dengan 23.

Contoh:

```
mysql> SELECT HOUR('09:25:45'), HOUR('17:32:59'), HOUR('23:05:05') ;
+-----------------+-----------------+-----------------+
| HOUR('09:25:45') | HOUR('17:32:59') | HOUR('23:05:05') |
+-----------------+-----------------+-----------------+
|                9 |              17 |              23 |
+-----------------+-----------------+-----------------+
1 row in set (0.00 sec)
```

✍ **MINUTE**(waktu)

Fungsi ini akan menghasilkan urutan menit dari jam yang ditentukan. Kisaran nilai yang dihasilkan antara 1 sampai dengan 59.

Contoh:

```
mysql> SELECT MINUTE('09:25:45'), MINUTE('17:32:59'),MINUTE('23:05:05') ;
+-------------------+-------------------+-------------------+
| MINUTE('09:25:45') | MINUTE('17:32:59') | MINUTE('23:05:05') |
+-------------------+-------------------+-------------------+
|                25 |                32 |                05 |
+-------------------+-------------------+-------------------+
1 row in set (0.00 sec)
```

✍ **MONTH**(tanggal)

Fungsi ini akan menghasilkan urutan bulan dari tanggal yang ditentukan. Tanggal ditulis dengan format 'yyyy-mm-dd', 'yyyymmdd' atau 'yymmdd'.

Contoh:

```
mysql> SELECT MONTH('2003-01-01'), MONTH('20031224') ;
+--------------------+-------------------+
| MONTH('2003-01-01') | MONTH('20031224') |
+--------------------+-------------------+
|                  1 |                12 |
+--------------------+-------------------+
1 row in set (0.00 sec)
```

✍ **MONTHNAME**(tanggal)

Fungsi ini akan menghasilkan nama buan dari tanggal yang ditentukan. Tanggal ditulis dengan format 'yyyy-mm-dd' , 'yyyymmdd' atau 'yymmdd'.

Contoh:

```
mysql> SELECT MONTHNAME('2003-01-01'), MONTHNAME('20031224') ;
+------------------------+----------------------+
```

```
| MONTHNAME('2003-01-01') | MONTHNAME('20031224') |
+-------------------------+-----------------------+
| January                 | December              |
+-------------------------+-----------------------+
1 row in set (0.00 sec)
```

✍ **NOW( )**

Fungsi ini akan menghasilkan tanggal dan jam sistem saat ini, dengan format tampilan 'yyyy-mm-dd hh:mm:ss' atau 'yyyymmddhhmmss'.

Contoh:

```
mysql> SELECT NOW( ), NOW( )+10, NOW( )-10 ;
+---------------------+----------------+----------------+
| NOW( )              | NOW( )+10      | NOW( )-10      |
+---------------------+----------------+----------------+
| 2003-05-03 12:57:36 | 20030503125746 | 20030503125756 |
+---------------------+----------------+----------------+
1 row in set (0.00 sec)
```

✍ **PERIOD_ADD**(periode, n-bulan)

Fungsi ini akan menghasilkan penjumlahan *n-bulan* pada periode yang ditentukan. Periode ditulis dengan format 'yyyymm' atau 'yymm' saja. Nilai yang dihasilkan akan ditulis dalam format 'yyyymm'.

Contoh:

```
mysql> SELECT PERIOD_ADD('200301', 12), PERIOD_ADD('0301', 12) ;
+-------------------------+-----------------------+
| PERIOD_ADD('200301', 12) | PERIOD_ADD('0301', 12) |
+-------------------------+-----------------------+
|                  200401 |                200401 |
+-------------------------+-----------------------+
1 row in set (0.00 sec)
```

✍ **PERIOD_DIFF**(periode1, periode2)

Fungsi ini akan menghasilkan selisih bulan antara dua periode yang ditentukan. Periode ditulis dengan format 'yyyymm' atau 'yymm' saja.

Contoh:

```
mysql> SELECT PERIOD_DIFF('200312', '200301') ;
+---------------------------------+
| PERIOD_DIFF('200312', '200301') |
+---------------------------------+
|                              11 |
+---------------------------------+
1 row in set (0.00 sec)

mysql> SELECT PERIOD_DIFF('200301', '200312') ;
+---------------------------------+
| PERIOD_DIFF('200301', '200312') |
+---------------------------------+
|                             -11 |
+---------------------------------+
1 row in set (0.00 sec)
```

✍ **QUARTER**(tanggal)

Fungsi ini akan menghasilkan nilai pada periode triwulan keberapa tanggal yang ditentukan tersebut. Tanggal ditulis dengan format 'yyyy-mm-dd', 'yyyymmdd' atau 'yymmdd'.

Contoh:

mysql> SELECT QUARTER(‘2003-01-01’), QUARTER(‘2003-07-01’) ;

```
+----------------------+----------------------+
| QUARTER('2003-01-01') | QUARTER('2003-07-01') |
+----------------------+----------------------+
|                    1 |                    3 |
+----------------------+----------------------+
```

1 row in set (0.00 sec)

✍ **SECOND**(waktu)

Fungsi ini akan menghasilkan urutan detik dari jam yang ditentukan. Kisaran nilai yang dihasilkan antara 1 sampai dengan 59.

Contoh:

mysql> SELECT SECOND(’09:25:45’), SECOND(’17:32:59’),SECOND(’23:05:05’) ;

```
+--------------------+--------------------+--------------------+
| MINUTE('09:25:45') | MINUTE('17:32:59') | MINUTE('23:05:05') |
+--------------------+--------------------+--------------------+
|                 45 |                 59 |                 05 |
+--------------------+--------------------+--------------------+
```

1 row in set (0.00 sec)

✍ **SEC_TO_TIME**(detik)

Fungsi ini akan menghasilkan jam:menit:detik dari sejumlah detik yang ditentukan.

Contoh:

mysql> SELECT SEC_TO_TIME(11111), SEC_TO_TIME(55555) ;

```
+--------------------+--------------------+
| SEC_TO_TIME(11111) | SEC_TO_TIME(55555) |
+--------------------+--------------------+
| 03:05:11           | 15:25:55           |
+--------------------+--------------------+
```

1 row in set (0.00 sec)

✍ **SUBDATE**(tanggal, **INTERVAL** ekspresi interval)

Fungsi ini identik dengan fungsi DATE_SUB( ).

Contoh:

mysql> SELECT SUBDATE('2003-05-01 12:00:00', INTERVAL 1 SECOND) ;

```
+-----------------------------------------------------+
| SUBDATE('2003-05-01 12:00:00', INTERVAL 1 SECOND)   |
+-----------------------------------------------------+
| 2003-05-01 11:59:59                                 |
+-----------------------------------------------------+
```

1 row in set (0.00 sec)

mysql> SELECT SUBDATE('2003-05-01 12:00:00', INTERVAL 10 SECOND) ;

```
+-----------------------------------------------------+
| SUBDATE('2003-05-01 12:00:00', INTERVAL 10 SECOND)  |
+-----------------------------------------------------+
| 2003-05-01 11:50:00                                 |
+-----------------------------------------------------+
```

1 row in set (0.00 sec)

mysql> SELECT SUBDATE('2003-05-01 12:00:00', INTERVAL 10 MINUTE) ;

```
+-----------------------------------------------------+
| SUBDATE('2003-05-01 12:00:00', INTERVAL 10 MINUTE)  |
+-----------------------------------------------------+
| 2003-05-01 11:50:00                                 |
+-----------------------------------------------------+
```

1 row in set (0.00 sec)

✍ **SYSDATE( )**

Fungsi ini akan menghasilkan tanggal dan jam sistem saat ini, dengan format tampilan 'yyyy-mm-dd hh:mm:ss' atau 'yyyymmddhhmmss'. Fungsi ini sama dengan fungsi NOW ().

Contoh:

```
mysql> SELECT SYSDATE( ), SYSDATE( )+10, SYSDATE( ) -10 ;
+---------------------+----------------+----------------+
| SYSDATE( )          | SYSDATE( )+10  | SYSDATE( )-10  |
+---------------------+----------------+----------------+
| 2003-05-03 12:57:36 | 20030503125746 | 20030503125756 |
+---------------------+----------------+----------------+
1 row in set (0.00 sec)
```

✍ **TIME_FORMAT**(waktu, format)

Fungsi ini akan mengganti format tampilan waktu yang ditentukan sesuai dengan kriteria format yang diberikan. Bentuk format yang digunakan sama dengan pada fungsi DATE_FORMAT( ). Hanya saja penerapannya adalah pada hal-hal yang berkenaan dengan waktu.

Contoh:

```
mysql> SELECT TIME_FORMAT('12:30:15','%H.%I Waktu Bandung') ;
+-----------------------------------------------+
| TIME_FORMAT('12:30:15','%H.%I Waktu Bandung') |
+-----------------------------------------------+
| 12.30 Waktu Bandung                           |
+-----------------------------------------------+
1 row in set (0.03 sec)

mysql> SELECT TIME_FORMAT(123015,'%H.%I Waktu Bandung') ;
+-----------------------------------------+
| TIME_FORMAT(123015,'%H.%I Waktu Bandung') |
+-----------------------------------------+
| 12.30 Waktu Bandung                     |
+-----------------------------------------+
1 row in set (0.03 sec)
```

✍ **TIME_TO_SEC**(waktu)

Fungsi ini akan menghasilkan nilai detik dari parameter waktu yang diberikan.

Contoh:

```
mysql> SELECT TIME_TO_SEC('00:00:59'), TIME_TO_SEC('00:02:00') ;
+--------------------------+-------------------------+
| TIME_TO_SEC('00:00:59') | TIME_TO_SEC('00:02:00') |
+--------------------------+-------------------------+
|                       59 |                     120 |
+--------------------------+-------------------------+
1 row in set (0.06 sec)

mysql> SELECT TIME_TO_SEC('00:10:00'), TIME_TO_SEC('01:00:00') ;
+--------------------------+-------------------------+
| TIME_TO_SEC('00:10:00') | TIME_TO_SEC('01:00:00') |
+--------------------------+-------------------------+
|                      600 |                    3600 |
+--------------------------+-------------------------+
1 row in set (0.06 sec)
```

✍ **TO_DAYS**(tanggal)

Fungsi ini akan menghasilkan jumlah hari yang dihitung sejak tahun 0 hingga tanggal yang diberikan pada argumen tersebut.

Contoh:

```
mysql> SELECT TO_DAYS('2003-01-01'), TO_DAYS('2003-02-01') ;
+----------------------+----------------------+
| TO_DAYS('2003-01-01') | TO_DAYS('2003-02-01') |
+----------------------+----------------------+
|               731581 |               731612 |
+----------------------+----------------------+
1 row in set (0.06 sec)
```

✍ **UNIX_TIMESTAMP**(tanggal)

Fungsi ini akan menghasilkan jumlah detik yang dihitung sejak '1970 -01-01 00:00:00' GMT hingga tanggal yang ditentukan pada argumen fungsi.

Contoh:

```
mysql> SELECT UNIX_TIMESTAMP( ), UNIX_TIMESTAMP('2003-01-01') ;
+------------------+-----------------------------+
| UNIX_TIMESTAMP( ) | UNIX_TIMESTAMP('2003-01-01') |
+------------------+-----------------------------+
|       1054479013 |                  1041354000 |
+------------------+-----------------------------+
1 row in set (0.06 sec)
```

✍ **WEEK(**tanggal**) atau WEEK**(tanggal, urutan_pertama)

Fungsi WEEK(tanggal) akan menghasilkan urutan minggu dari tanggal yang ditentukan. Fungsi WEEK(tanggal, urutan_pertama) akan menghasilkan urutan minggu dari tanggal yang ditentukan, di mana urutan_pertama adalah urutan minggu mulainya. Secara default hitungan satu minggu dimulai dari hari Minggu (urutan 0), kemudian Senin (urutan 1) dan seterusnya. Tanggal ditulis dengan format 'yyyy-mm-dd', 'yyyymmdd' atau 'yymmdd'.

Contoh:

```
mysql> SELECT WEEK('2003-01-01'),WEEK('2003-01-31'),WEEK('2003-12-31') ;
+--------------------+--------------------+--------------------+
| WEEK('2003-01-01') | WEEK('2003-01-31') | WEEK('2003-12-31') |
+--------------------+--------------------+--------------------+
|                  1 |                  5 |                 53 |
+--------------------+--------------------+--------------------+
1 row in set (0.00 sec)

mysql> SELECT WEEK('2003-12-31',0), WEEK('2003 -12-31',1),
   -> WEEK('2003-01-01',6) ;
+---------------------+---------------------- +----------------------+
| WEEK('2003-12-31',0) | WEEK('2003-12-31',1) | WEEK('2003-01-01',6) |
+---------------------+---------------------- +----------------------+
|                  53 |                   53 |                    1 |
+---------------------+---------------------- +----------------------+
1 row in set (0.00 sec)
```

✍ **WEEKDAY**(tanggal)

Fungsi ini akan menghasilkan urutan nama hari dari tanggal yang ditentukan. Tanggal ditulis dengan format 'yyyy-mm-dd', 'yyyymmdd' atau 'yymmdd'. Urutan dimulai dari angka nol (0), di mana urutan ke-nol adalah Senin, urutan k e-satu untuk Selasa dan seterusnya hingga urutan ke-6 untuk Sabtu. Fungsi ini mirip dengan fungsi DAYOFWEEK( ), dengan perbedaan pada nilai urutan dan posisi urutan.

Contoh:

```
mysql> SELECT DAYNAME('2003-01-01'), WEEKDAY('2003-01-01') ;
+----------------------+----------------------+
| DAYNAME('2003-01-01') | WEEKDAY('2003-01-01') |
```

```
+----------------------+----------------------+
| Wednesday            |                    3 |
+----------------------+----------------------+
1 row in set (0.00 sec)

mysql> SELECT DAYNAME('20030224'), WEEKDAY('20030224') ;
+--------------------+---------------------- +
| DAYNAME('20030224') | DAYOFWEEK('20030224') |
+--------------------+---------------------- +
| Wednesday           |                    3 |
+--------------------+---------------------- +
1 row in set (0.00 sec)
```

✍ **YEAR**(tanggal)

Fungsi ini akan menghasilkan nilai angka tahun dari tanggal yang ditentukan, dengan kisaran nilai dari 1000 sampai dengan 9999 (berarti dari tahun 1000 sampai dengan tahun 9999). Tanggal ditulis dengan format 'yyyy-mm-dd', 'yyyymmdd' atau 'yymmdd'.

Contoh:

```
mysql> SELECT YEAR('2003-01-01'), YEAR('20031224') ;
+-------------------+------------------+
| YEAR('2003-01-01') | YEAR('20201224') |
+-------------------+------------------+
|               2003 |             2020 |
+-------------------+------------------+
1 row in set (0.00 sec)
```

Fungsi Pembanding (Comparison Functions)

✍ **GREATEST(***ekspresi1, ekspresi2, ...***)**

Fungsi ini akan membandingkan semua nilai yang ada dan menampilkan nilai terbesarnya.

Contoh:

```
mysql> SELECT GREATEST(10, 35, 33, 50, 77) ;
+------------------------------+
| GREATEST(10, 35, 33, 50, 77) |
+------------------------------+
|                           77 |
+------------------------------+
1 row in set (0.03 sec)
```

✍ **IF**(ekspresi1, ekspresi2, ekspresi3)

Fungsi ini akan menguji ekspresi1. Bila ekspresi1 adalah benar, maka fungsi akan menampilkan nilai eksepresi2. Sebaliknya bila salah, maka nilai ekspresi3 yang akan ditampilkan.

Contoh:

```
mysql> SELECT IF(25 > 10, 'Benar', 'Salah') ;
+------------------------------+
| IF(25 > 10, 'Benar', 'Salah') |
+------------------------------+
| Benar                        |
+------------------------------+
1 row in set (0.03 sec)

mysql> SELECT IF(25 < 10, 'Benar', 'Salah') ;
+------------------------------+
| IF(25 < 10, 'Benar', 'Salah') |
+------------------------------+
| Salah                        |
+------------------------------+
1 row in set (0.03 sec)
```

✍ **IFNULL**(ekspresi1, ekspresi2,…)

Bila nilai pada ekspresi1 adalah BUKAN NULL maka fungsi ini akan menampilkan nilai ekspresi1 tersebut. Bilai nilai ekspresi1 adalah NULL, maka fungsi ini akan menampilkan nilai ekspresi2. Singkatnya, fungsi ini akan menampilkan salah satu nilai yang BUKAN NULL.

Contoh:

```
mysql> SELECT IFNULL(NULL, 55), IFNULL(55, NULL) ;
+------------------+------------------+
| IFNULL(NULL, 55) | IFNULL(55, NULL) |
+------------------+------------------+
| 55               | 55               |
+------------------+------------------+
1 row in set (0.03 sec)
```

✍ **INTERVAL**(n1, n2, n3,…)

Fungsi ini akan menghasilkan 0 bila nilai n1<n2, menghasilkan 1 bila n1<n3, menghasilkan 2 bila n1<n4, dan seterusnya. Tetapi akan menghasilkan –1 bila n adalah NULL. Fungsi ini hanya bisa dijalankan bila nilai n1<n2<n3<n… Jika syarat ini tidak dipenuhi, maka akan menghasilkan nilai yang tidak benar.

Contoh:

```
mysql> SELECT INTERVAL(1, 2, 3, 4, 5, 6, 7, 8, 9, 10) ;
+-----------------------------------------+
| INTERVAL(1, 2, 3, 4, 5, 6, 7, 8, 9, 10) |
+-----------------------------------------+
|                                       0 |
+-----------------------------------------+
1 row in set (0.05 sec)

mysql> SELECT INTERVAL(1, 0, 3, 4, 5, 6, 7, 8, 9, 10) ;
+-----------------------------------------+
| INTERVAL(1, 0, 3, 4, 5, 6, 7, 8, 9, 10) |
+-----------------------------------------+
|                                       1 |
+-----------------------------------------+
1 row in set (0.05 sec)

mysql> SELECT INTERVAL(1, 0, 0, 4, 5, 6, 7, 8, 9, 10) ;
+-----------------------------------------+
| INTERVAL(1, 0, 0, 4, 5, 6, 7, 8, 9, 10) |
+-----------------------------------------+
|                                       2 |
+-----------------------------------------+
1 row in set (0.05 sec)

mysql> SELECT INTERVAL(1, 0, 0, 0, 5, 6, 7, 8, 9, 10) ;
+-----------------------------------------+
| INTERVAL(1, 0, 0, 0, 5, 6, 7, 8, 9, 10) |
+-----------------------------------------+
|                                       3 |
+-----------------------------------------+
1 row in set (0.05 sec)
```

✍ **ISNULL**(ekspresi)

Bila nilai pada ekspresi adalah NULL, maka fungsi akan menampilkan angka 1. Bila tidak NULL, maka fungsi akan menampilkan angka 0.

Contoh:

```
mysql> SELECT ISNULL(NULL), ISNULL(55) ;
+--------------+------------+
| ISNULL(NULL) | ISNULL(55) |
+--------------+------------+
|            1 |          0 |
```

```
+--------------+------------+
1 row in set (0.03 sec)
```

✍ **LEAST**(ekspresi1, ekspresi2, …)

Fungsi ini akan membandingkan semua nilai yang ada dan menampilkan nilai terkecilnya.

Contoh:

```
mysql> SELECT LEAST(10, 35, 33, 50, 77) ;
+--------------------------+
| LEAST(10, 35, 33, 50, 77) |
+--------------------------+
|                       10 |
+--------------------------+
1 row in set (0.03 sec)
```

✍ **STRCMP**(string1, string2)

Fungsi ini akan membandingkan string1 dengan string2. Jika string1 tepat sama dengan string2, maka fungsi akan menghasilkan nilai 0. Jika string1 lebih pendek daripada string2, maka fungsi akan menghasilkan nilai –1. Selain dari kondisi di atas, maka fungsi akan menghasilkan nilai 1.

Contoh:

```
mysql> SELECT STRCMP('saya','saya'), STRCMP('saya','kalian') ;
+----------------------+-------------------------+
| STRCMP('saya','saya') | STRCMP('saya','kalian') |
+----------------------+-------------------------+
|                     0 |                       1 |
+----------------------+-------------------------+
1 row in set (0.03 sec)

mysql> SELECT STRCMP('kalian','saya'), STRCMP('saya','abdi') ;
+-----------------------+----------------------+
| STRCMP('kalian,'saya') | STRCMP('saya','abdi') |
+-----------------------+----------------------+
|                    -1 |                    1 |
+-----------------------+----------------------+
1 row in set (0.03 sec)

mysql> SELECT STRCMP('abdi','abdi'), STRCMP('abdi','ibda') ;
+----------------------+-----------------------+
| STRCMP('abdi,'abdi') | STRCMP('abdi','ibda') |
+----------------------+-----------------------+
|                    0 |                    -1 |
+----------------------+-----------------------+
1 row in set (0.03 sec)
```

Fungsi Agregat (Aggregate Functions)

Fungsi-fungsi agregat ini digunakan pada saat kita ingin menampilkan hasil agregat dari sebuah kolom. Sebagai contoh kita akan menggunakan data-data pada Database **kesiswaandb** dan Tabel **nilai**.

✍ **AVG(***ekspresi***)**

Fungsi ini akan menghasilkan nilai rata-rata dari sekumpulan data pada kolom tertentu.

Contoh:

```
mysql> SELECT AVG(nilai) FROM nilai ;
+------------+
| AVG(nilai) |
```

```
+------------+
|    79.3844 |
+------------+
```
1 row in set (0.00 sec)

✍ **BIT_AND**(ekspresi)

Fungsi ini akan menghasilkan nilai dari operasi bitwise AND pada ekspresi (bukan NULL) yang diberikan.

Contoh:

```
mysql> SELECT BIT_AND(nilai) FROM nilai ;
+---------------------------+
| BIT_AND(nilai) FROM nilai |
+---------------------------+
|                         0 |
+---------------------------+
```
1 row in set (0.00 sec)

✍ **BIT_OR**(ekspresi)

Fungsi ini akan menghasilkan nilai dari operasi bitwise OR pada ekspresi (bukan NULL) yang diberikan.

Contoh:

```
mysql> SELECT BIT_OR(nilai) FROM nilai ;
+--------------------------+
| BIT_OR(nilai) FROM nilai |
+--------------------------+
|                        0 |
+--------------------------+
```
1 row in set (0.00 sec)

✍ **COUNT** (ekspresi)

Fungsi ini akan menghasilkan nilai jumlah data (record) yang ada dari kriteria tertentu.

Contoh:

Berapa jumlah record yang ada di dalam tabel nilai?

```
mysql> SELECT COUNT(*) FROM nilai ;
+----------+
| COUNT(*) |
+----------+
|      180 |
+----------+
```
1 row in set (0.00 sec)

Berapa jumlah record yang ada di dalam Tabel **siswa**?

```
mysql> SELECT COUNT(*) FROM siswa ;
+----------+
| COUNT(*) |
+----------+
|       30 |
+----------+
```
1 row in set (0.00 sec)

✍ **MAX**(ekspresi)

Fungsi ini akan menghasilkan nilai maksimum dari sekumpulan data.

Contoh:

Berapa nilai maksimum siswa yang ada di dalam Tabel **nilai** ?

```
mysql> SELECT MAX(nilai) FROM nilai ;
+-----------+
```

```
| MAX(nilai)|
+-----------+
|        96 |
+-----------+
1 row in set (0.00 sec)
```

Kapan tanggal kelahiran siswa yang paling muda?

```
mysql> SELECT MAX(tgl_lahir) FROM siswa ;
+----------------+
| MAX(tgl_lahir) |
+----------------+
| 1986-12-18     |
+----------------+
1 row in set (0.00 sec)
```

✍ **MIN(**ekspresi**)**

Fungsi ini akan menghasilkan nilai minimum dari sekumpulan data.

Contoh:

Berapa nilai maksimum siswa yang ada di dalam tabel nilai?

```
mysql> SELECT MIN(nilai) FROM nilai ;
+-----------+
| MIN(nilai)|
+-----------+
|        45 |
+-----------+
1 row in set (0.00 sec)
```

Kapan tanggal kelahiran siswa yang paling tua?

```
mysql> SELECT MIN(tgl_lahir) FROM siswa ;
+----------------+
| MIN(tgl_lahir) |
+----------------+
| 1986-02-04     |
+----------------+
1 row in set (0.00 sec)
```

✍ **STD(**ekspresi**)**

Fungsi ini akan menghasilkan nilai statistika standar deviasi dari sekumpulan data.

Contoh:

```
mysql> SELECT STD(nilai) FROM nilai ;
+------------+
| STD(nilai) |
+------------+
|    10.2921 |
+------------+
1 row in set (0.00 sec)
```

✍ **STDDEV(**ekspresi**)**

Fungsi ini akan menghasilkan nilai statistika standar deviasi dari sekumpulan data. Sama dengan fungsi STD( ).

Contoh:

```
mysql> SELECT STDDEV(nilai) FROM nilai ;
+---------------+
| STDDEV(nilai) |
+---------------+
|       10.2921 |
+---------------+
```

1 row in set (0.00 sec)

✍ **SUM(**ekspresi**)**

Fungsi ini akan menghasilkan jumlah keseluruhan dari sekumpulan data (kecuali NULL).

Contoh:

```
+------------+
| SUM(nilai) |
+------------+
|      14291 |
+------------+
1 row in set (0.00 sec)
```

Fungsi Lain-lain (Miscellaneous Functions)

✍ **BENCHMARK(***nPengulangan, ekspresi***)**

Fungsi ini digunakan untuk menguji kecepatan MySQL dalam melaksanakan suatu perintah sebanyak nPengulangan kali. Nilai yang dihasilkan selalu Nol.

Contoh:

```
mysql> SELECT BENCHMARK(1000000, ENCODE('Apa','Kabar') ;
+-----------------------------------------+
| BENCHMARK(1000000, ENCODE('Apa','Kabar') |
+-----------------------------------------+
|                                       0 |
+-----------------------------------------+
1 row in set (3.68 sec)
```

✍ **BIT_COUNT(**numerik**)**

Fungsi ini menghasilkan jumlah bit dari parameter n.

Contoh:

```
mysql> SELECT BIT_COUNT(1), BIT_COUNT(10), BIT_COUNT(100) ;
+--------------+---------------+----------------+
| BIT_COUNT(1) | BIT_COUNT(10) | BIT_COUNT(100) |
+--------------+---------------+----------------+
|            1 |             2 |              3 |
+--------------+---------------+----------------+
1 row in set (0.00 sec)
```

✍ **DATABASE( )**

Fungsi ini akan menampilkan informasi database yang sedang aktif.

Contoh:

```
mysql> SELECT DATABASE( ) ;
+-------------+
| DATABASES( ) |
+-------------+
| latih1db    |
+-------------+
1 row in set (0.00 sec)
```

✍ **DECODE(**string, kata_password**)**

Fungsi ini akan mengembalikan hasil dari fungsi ENCODE( ) ke bentuk aslinya.

Contoh:

```
mysql> SELECT DECODE(ENCODE('saya','password'),'password') ;
+-------------------------------------------+
```

```
| DECODE(ENCODE('saya','password'),'password') |
+----------------------------------------------+
| saya                                         |
+----------------------------------------------+
1 row in set (0.00 sec)
```

✍ **ENCODE(**string, kata_password**)**

Fungsi ini akan mengacak string dengan kata_password sebagai kode sandinya.

Contoh:

```
mysql> SELECT ENCODE('saya','password') ;
+---------------------------+
| ENCODE('saya','password') |
+---------------------------+
| B3-y                      |
+---------------------------+
1 row in set (0.00 sec)
```

✍ **ENCRYPT(**string**)**

Fungsi ini akan mengacak string dengan memanggil fasilitas Encrypt pada sistem operasi Unix. Bila Encrypt ini tidak ada, maka hasilnya adalah NULL (terutama pada Windows).

Contoh:

```
mysql> SELECT ENCRYPT('apa kabar') ;
+----------------------+
| ENCRYPT('apa kabar') |
+----------------------+
| NULL                 |
+----------------------+
1 row in set (0.00 sec)
```

✍ **GET_LOCK(**string, lama_penguncian **)**

Fungsi ini digunakan untuk mengunci sebuah string yang ditentukan dalam waktu lama_penguncian tertentu. Jangan disalahartikan bahwa fungsi ini akan mengunci database, tabel maupun kolom tertentu. Fungsi ini tidak untuk yang berkenaan dengan database, tetapi hanya pada string tertentu. Untuk membuka penguncian digunakan fungsi RELEASE_LOCK( ). Fungsi akan memberikan nilai 1 bila proses penguncian berhasil, 0 bila gagal dan NULL jika terjadi kesalahan.

Contoh:

Mengunci string 'cobakunci' selama 10 detik.

```
mysql> SELECT GET_LOCK('cobakunci',10) ;
+--------------------------+
| GET_LOCK('cobakunci',10) |
+--------------------------+
|                        1 |
+--------------------------+
1 row in set (0.00 sec)
```

✍ **LAST_INSERT_ID(**ekspresi**)**

Fungsi ini akan menampilkan nomor urut terakhir pada suatu kolom yang menggunakan

AUTO_INCREMENT.

Contoh:

```
mysql> INSERT jadwal_tes (jadwal_tes_ID)
    -> VALUES (NULL) ;
Query OK, 1 row affected (0.00 sec)

mysql> SELECT LAST_INSERT_ID( )
+------------------+
| LAST_INSERT_ID( ) |
+------------------+
|                25 |
+------------------+
1 row in set (0.00 sec)
```

✍ **LOAD_FILE**(nama_file)

Fungsi akan membaca sebuah file teks yang berada di dalam server dan kemudian memasukkan isinya ke dalam sebuah tabel (bila memang ditentukan demikian). Lokasi file harus berada di dalam server dan diberi alamat lokasi/direktori penyimpanan file selengkap mungkin. Karena itu, Anda harus memiliki izin akses FILE untuk dapat menjalankan fungsi ini.

Contoh:

```
mysql> UPDATE                    nilai
  -> SET                       nilai=LOAD_FILE('/temporer/data_nilai/nilaisiswa.txt'
  -> WHERE                       siswa_id = '199901123';

mysql> UPDATE                    nilai
  -> SET                       nilai=LOAD_FILE('C:/temporer/data_nilai/nilaisiswa.txt'
  -> WHERE                       siswa_id = '199901123';
```

✍ **PASSWORD**(string)

Fungsi ini akan mengacak string yang dimasukkan dengan kode acak tertentu. Biasanya digunakan untuk membuat password.

Contoh:

```
mysql> SELECT PASSWORD('apaaja'), PASSWORD('dimanaaja') ;
+-------------------+-----------------------+
| PASSWORD('apaaja')| PASSWORD('dimanaaja') |
+-------------------+-----------------------+
| 67f947ac0ba10fe7  | 53c5996460959eab      |
+-------------------+-----------------------+
1 row in set (0.00 sec)
```

✍ **RELEASE_LOCK**(string)

Fungsi ini digunakan untuk membuka penguncian sebuah string yang telah dilakukan sebelumnya dengan fungsi GET_LOCK( ). Fungsi akan memberikan nilai 1 bila proses pembukaan kunci berhasil, 0 bila gagal dan NULL jika terjadi kesalahan.

Contoh:

Membuka penguncian string 'cobakunci'.

```
mysql> SELECT RELEASE_LOCK('cobakunci') ;
+---------------------------+
| RELEASE_LOCK('cobakunci') |
+---------------------------+
|                         1 |
+---------------------------+
1 row in set (0.00 sec)
```

✍ **SESSION_USER( )**

Fungsi ini akan menampilkan nama user yang sedang aktif. Sama dengan fungsi SYSTEM_USER( ) dan USER().

Contoh:

```
mysql> SELECT SESSION_USER( ) ;
+-----------------+
| SESSION_USER( )  |
+-----------------+
| arbie@localhost |
+-----------------+
1 row in set (0.00 sec)
```

✍ **SYSTEM_USER( )**

Fungsi ini akan menampilkan nama user yang sedang aktif. Sama dengan fungsi SESSION_USER( ) dan USER( ).

Contoh:

```
mysql> SELECT SYSTEM_USER( ) ;
+-----------------+
| SYSTEM_USER( )   |
+-----------------+
| arbie@localhost |
+-----------------+
1 row in set (0.00 sec)
```

✍ **USER( )**

Fungsi ini akan menampilkan nama user yang sedang aktif. Sama dengan fungsi SYSTEM_USER( ) dan SESSION_USER( ).

Contoh:

```
mysql> SELECT USER( ) ;
+-----------------+
| USER( )          |
+-----------------+
| arbie@localhost |
+-----------------+
1 row in set (0.00 sec)
```

✍ **VERSION( )**

Fungsi ini akan menampilkan versi software MySQL server yang digunakan.

Contoh:

```
mysql> SELECT VERSION( ) ;
+-----------+
| VERSION( ) |
+-----------+
| 4.0.14    |
+-----------+
1 row in set (0.00 sec)
```