FUNCTIONS:

Here is the list of all important MySQL functions. Each function has been explained along with suitable example.

- **MySQL Group By Clause** – The MySQL GROUP BY statement is used along with the SQL aggregate functions like SUM to provide means of grouping the result dataset by certain database table column(s).
- **MySQL IN Clause** – This is a clause, which can be used along with any MySQL query to specify a condition.
- **MySQL BETWEEN Clause**– This is a clause, which can be used along with any MySQL query to specify a condition.
- **MySQL UNION Keyword** – Use a UNION operation to combine multiple result sets into one.
- **MySQL COUNT Function** – The MySQL COUNT aggregate function is used to count the number of rows in a database table.
- **MySQL MAX Function** – The MySQL MAX aggregate function allows us to select the highest (maximum) value for a certain column.
- **MySQL MIN Function** – The MySQL MIN aggregate function allows us to select the lowest (minimum) value for a certain column.
- **MySQL AVG Function** – The MySQL AVG aggregate function selects the average value for certain table column.
- **MySQL SUM Function** – The MySQL SUM aggregate function allows selecting the total for a numeric column.
- **MySQL SQRT Functions** – This is used to generate a square root of a given number.

You can use SQRT function to find out square root of various records as well. To understand **SQRT** function in more detail, consider an **employee_tbl** table, which is having following records −

```
mysql> SELECT * FROM employee_tbl;
+------+------+------------+--------------------+
|  id  | name | work_date  | daily_typing_pages |
+------+------+------------+--------------------+
|   1  | John | 2007-01-24 |       250          |
|   2  | Ram  | 2007-05-27 |       220          |
|   3  | Jack | 2007-05-06 |       170          |
|   3  | Jack | 2007-04-06 |       100          |
|   4  | Jill | 2007-04-06 |       220          |
|   5  | Zara | 2007-06-06 |       300          |
|   5  | Zara | 2007-02-06 |       350          |
+------+------+------------+--------------------+
7 rows in set (0.00 sec)
```

Now, suppose based on the above table you want to calculate square root of all the dialy_typing_pages, then you can do so by using the following command −

```
msql> SELECT name, SQRT(daily_typing_pages)
   -> FROM employee_tbl;
+------+--------------------------+
| name | SQRT(daily_typing_pages) |
+------+--------------------------+
| John |        15.811388         |
| Ram  |        14.832397         |
| Jack |        13.038405         |
| Jack |        10.000000         |
| Jill |        14.832397         |
| Zara |        17.320508         |
| Zara |        18.708287         |
+------+--------------------------+
7 rows in set (0.00 sec)
```

- **MySQL RAND Function –** This is used to generate a random number using MySQL command.

MySQL has a **RAND** function that can be invoked to produce random numbers between 0 and 1 −

```
mysql>  SELECT RAND( ), RAND( ), RAND( );
+------------------+-----------------+------------------+
|      RAND( )     |      RAND( )    |      RAND( )     |
+------------------+-----------------+------------------+
| 0.45464584925645 | 0.1824410643265 | 0.54826780459682 |
+------------------+-----------------+------------------+
1 row in set (0.00 sec)
```

When invoked with an integer argument, RAND( ) uses that value to seed the random number generator. Each time you seed the generator with a given value, RAND( ) will produce a repeatable series of numbers −

```
mysql>  SELECT RAND(1), RAND( ), RAND( );
+------------------+------------------+------------------+
|     RAND(1 )     |      RAND( )     |      RAND( )     |
+------------------+------------------+------------------+
| 0.18109050223705 | 0.75023211143001 | 0.20788908117254 |
+------------------+------------------+------------------+
1 row in set (0.00 sec)
```

You can use **ORDER BY RAND()** to randomize a set of rows or values as follows −

To understand **ORDER BY RAND()** function, consider an **employee_tbl** table, which is having the following records −

```
mysql> SELECT * FROM employee_tbl;
+------+------+------------+--------------------+
|  id  | name | work_date  | daily_typing_pages |
+------+------+------------+--------------------+
|   1  | John | 2007-01-24 |        250         |
|   2  | Ram  | 2007-05-27 |        220         |
|   3  | Jack | 2007-05-06 |        170         |
|   3  | Jack | 2007-04-06 |        100         |
|   4  | Jill | 2007-04-06 |        220         |
|   5  | Zara | 2007-06-06 |        300         |
|   5  | Zara | 2007-02-06 |        350         |
+------+------+------------+--------------------+
7 rows in set (0.00 sec)
```

Now, use the following commands −

```
mysql> SELECT * FROM employee_tbl ORDER BY RAND();
+------+------+------------+--------------------+
|  id  | name | work_date  | daily_typing_pages |
+------+------+------------+--------------------+
|   5  | Zara | 2007-02-06 |        350         |
|   5  | Zara | 2007-06-06 |        300         |
|   3  | Jack | 2007-05-06 |        170         |
|   2  | Ram  | 2007-05-27 |        220         |
|   4  | Jill | 2007-04-06 |        220         |
|   5  | Zara | 2007-02-06 |        350         |
|   1  | John | 2007-01-24 |        250         |
+------+------+------------+--------------------+
7 rows in set (0.01 sec)

mysql> SELECT * FROM employee_tbl ORDER BY RAND();
+------+------+------------+--------------------+
|  id  | name | work_date  | daily_typing_pages |
+------+------+------------+--------------------+
|   5  | Zara | 2007-02-06 |        350         |
|   5  | Zara | 2007-06-06 |        300         |
|   3  | Jack | 2007-05-06 |        170         |
|   2  | Ram  | 2007-05-27 |        220         |
|   4  | Jill | 2007-04-06 |        220         |
|   5  | Zara | 2007-02-06 |        350         |
|   1  | John | 2007-01-24 |        250         |
+------+------+------------+--------------------+
7 rows in set (0.00 sec)
```

- **MySQL CONCAT Function** – This is used to concatenate any string inside any MySQL command.

MySQL **CONCAT** function is used to concatenate two strings to form a single string. Try out the following example −

```
mysql> SELECT CONCAT('FIRST ', 'SECOND');
+----------------------------+
| CONCAT('FIRST ', 'SECOND') |
+----------------------------+
|         FIRST SECOND       |
+----------------------------+
1 row in set (0.00 sec)
```

To understand **CONCAT** function in more detail, consider an **employee_tbl** table, which is having the following records −

```
mysql> SELECT * FROM employee_tbl;
+------+------+------------+--------------------+
|  id  | name | work_date  | daily_typing_pages |
+------+------+------------+--------------------+
|   1  | John | 2007-01-24 |        250         |
|   2  | Ram  | 2007-05-27 |        220         |
|   3  | Jack | 2007-05-06 |        170         |
|   3  | Jack | 2007-04-06 |        100         |
|   4  | Jill | 2007-04-06 |        220         |
|   5  | Zara | 2007-06-06 |        300         |
|   5  | Zara | 2007-02-06 |        350         |
+------+------+------------+--------------------+
7 rows in set (0.00 sec)
```

Now, suppose based on the above table you want to concatenate all the names employee ID and work_date, then you can do it using the following command −

```
mysql> SELECT CONCAT(id, name, work_date)
   -> FROM employee_tbl;
+-----------------------------+
| CONCAT(id, name, work_date) |
+-----------------------------+
|      1John2007-01-24        |
|      2Ram2007-05-27         |
|      3Jack2007-05-06        |
|      3Jack2007-04-06        |
|      4Jill2007-04-06        |
|      5Zara2007-06-06        |
|      5Zara2007-02-06        |
+-----------------------------+
7 rows in set (0.00 sec)
```

- MySQL DATE and Time Functions–

1.ADDDATE(date,INTERVAL expr unit), ADDDATE(expr,days)

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

```
mysql> SELECT DATE_ADD('1998-01-02', INTERVAL 31 DAY);
+---------------------------------------------------------+
|         DATE_ADD('1998-01-02', INTERVAL 31 DAY)         |
+---------------------------------------------------------+
|                        1998-02-02                       |
+---------------------------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT ADDDATE('1998-01-02', INTERVAL 31 DAY);
+---------------------------------------------------------+
|          ADDDATE('1998-01-02', INTERVAL 31 DAY)         |
+---------------------------------------------------------+
|                        1998-02-02                       |
+---------------------------------------------------------+
1 row in set (0.00 sec)
```

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

```
mysql> SELECT ADDDATE('1998-01-02', 31);
+---------------------------------------------------------+
|         DATE_ADD('1998-01-02', INTERVAL 31 DAY)         |
+---------------------------------------------------------+
|                        1998-02-02                         |
+-----------------------------------------------------------+
1 row in set (0.00 sec)
```

2.ADDTIME(expr1,expr2)

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

```
mysql> SELECT ADDTIME('1997-12-31 23:59:59.999999','1 1:1:1.000002');
+-----------------------------------------------------------+
| DATE_ADD('1997-12-31 23:59:59.999999','1 1:1:1.000002') |
+-----------------------------------------------------------+
|                1998-01-02 01:01:01.000001                 |
+-----------------------------------------------------------+
1 row in set (0.00 sec)
```

3.CONVERT_TZ(dt,from_tz,to_tz)

This converts a datetime value dt from the time zone given by from_tz to the time zone given by to_tz and returns the resulting value. This function returns NULL if the arguments are invalid.

```
mysql> SELECT CONVERT_TZ('2004-01-01 12:00:00','GMT','MET');
+-----------------------------------------------------------+
| CONVERT_TZ('2004-01-01 12:00:00','GMT','MET')            |
+-----------------------------------------------------------+
|                  2004-01-01 13:00:00                      |
+-----------------------------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT CONVERT_TZ('2004-01-01 12:00:00','+00:00','+10:00');
+-----------------------------------------------------------+
| CONVERT_TZ('2004-01-01 12:00:00','+00:00','+10:00')      |
+-----------------------------------------------------------+
|                  2004-01-01 22:00:00                      |
+-----------------------------------------------------------+
1 row in set (0.00 sec)
```

4.CURDATE()

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

```
mysql> SELECT CURDATE();
+-----------------------------------------------------------+
|                        CURDATE()                          |
+-----------------------------------------------------------+
|                       1997-12-15                          |
+-----------------------------------------------------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT CURDATE() + 0;
+---------------------------------------------------------+
|                      CURDATE() + 0                      |
+---------------------------------------------------------+
|                         19971215                        |
+---------------------------------------------------------+
1 row in set (0.00 sec)
```

5.CURRENT_DATE and CURRENT_DATE()

CURRENT_DATE and CURRENT_DATE() are synonyms for CURDATE()

To know more functions in details refer -- https://www.tutorialspoint.com/mysql/mysql-date-time-functions.html

- **MySQL Numeric Functions** –

1. ABS(X)

The ABS() function returns the absolute value of X. Consider the following example −

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

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

2.ACOS(X)

This function returns the arccosine of X. The value of X must range between .1 and 1 or NULL will be returned. Consider the following example −

```
mysql> SELECT ACOS(1);
```

```
+-----------------------------------------------------------+
|                          ACOS(1)                          |
+-----------------------------------------------------------+
|                         0.000000                          |
+-----------------------------------------------------------+
1 row in set (0.00 sec)
```

3.ASIN(X)

The ASIN() function returns the arcsine of X. The value of X must be in the range of .1 to 1 or NULL is returned.

```
mysql> SELECT ASIN(1);
+-----------------------------------------------------------+
|                          ASIN(1)                          |
+-----------------------------------------------------------+
|                     1.5707963267949                       |
+-----------------------------------------------------------+
1 row in set (0.00 sec)
```

4.ATAN(X)

This function returns the arctangent of X.

```
mysql> SELECT ATAN(1);
+-----------------------------------------------------------+
|                         ATAN(1)                           |
+-----------------------------------------------------------+
|                    0.78539816339745                       |
+-----------------------------------------------------------+
1 row in set (0.00 sec)
```

```
To know more functions in details refer --
https://www.tutorialspoint.com/mysql/mysql-numeric-functions.htm
```

- **MySQL String Functions –**

1.ASCII(str)

Returns the numeric value of the leftmost character of the string str. Returns 0 if str is the empty string. Returns NULL if str is NULL. ASCII() works for characters with numeric values from 0 to 255.

```
mysql> SELECT ASCII('2');
+---------------------------------------------------------+
|                       ASCII('2')                        |
+---------------------------------------------------------+
|                           50                            |
+---------------------------------------------------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT ASCII('dx');
+---------------------------------------------------------+
|                       ASCII('dx')                       |
+---------------------------------------------------------+
|                          100                            |
+---------------------------------------------------------+
1 row in set (0.00 sec)
```

2.BIN(N)

Returns a string representation of the binary value of N, where N is a longlong (BIGINT) number. This is equivalent to CONV(N,10,2). Returns NULL if N is NULL.

```
mysql> SELECT BIN(12);
+---------------------------------------------------------+
|                          BIN(12)                        |
+---------------------------------------------------------+
|                           1100                          |
+---------------------------------------------------------+
1 row in set (0.00 sec)
```

3.BIT_LENGTH(str)

Returns the length of the string str in bits.

```
mysql> SELECT BIT_LENGTH('text');
+---------------------------------------------------------+
|                   BIT_LENGTH('text')                    |
+---------------------------------------------------------+
|                          32                             |
+---------------------------------------------------------+
1 row in set (0.00 sec)
```

4.CHAR(N,... [USING charset_name])

CHAR() interprets each argument N as an integer and returns a string consisting of the characters given by the code values of those integers. NULL values are skipped.

```
mysql> SELECT CHAR(77,121,83,81,'76');
+---------------------------------------------------------+
|                 CHAR(77,121,83,81,'76')                 |
+---------------------------------------------------------+
|                          MySQL                          |
+---------------------------------------------------------+
1 row in set (0.00 sec)
```

5.CHAR_LENGTH(str)

Returns the length of the string str, measured in characters. A multi-byte character counts as a single character. This means that for a string containing five two-byte characters, LENGTH() returns 10, whereas CHAR_LENGTH() returns 5.

```
mysql> SELECT CHAR_LENGTH("text");
+---------------------------------------------------------+
|                   CHAR_LENGTH("text")                   |
+---------------------------------------------------------+
|                            4                            |
+---------------------------------------------------------+
1 row in set (0.00 sec)
```

Creating Own Function:

Just as you can create functions in other languages, you can create your own functions in MySQL. Let's take a closer look.

Syntax

The syntax to create a function in MySQL is:

```
CREATE FUNCTION function_name [ (parameter datatype [, parameter datatype]) ]
RETURNS return_datatype

BEGIN

   declaration_section

   executable_section

END;
```

function_name

The name to assign to this function in MySQL.

parameter

One or more parameters passed into the function. When creating a function, all parameters are considered to be **IN parameters** (not OUT or INOUT parameters) where the parameters can be referenced by the function but can not be overwritten by the function.

return_datatype

The data type of the function's return value.

declaration_section

The place in the function where you declare local variables.

executable_section

The place in the function where you enter the code for the function.

Example

Let's look at an example that shows how to create a function in MySQL:

```
DELIMITER //

CREATE FUNCTION CalcIncome ( starting_value INT )
RETURNS INT

BEGIN

   DECLARE income INT;

   SET income = 0;

   label1: WHILE income <= 3000 DO
     SET income = income + starting_value;
   END WHILE label1;

   RETURN income;

END; //

DELIMITER ;
```

You could then reference your new function as follows:

```
SELECT CalcIncome (1000);
```

Drop Function:

Once you have created your function in MySQL, you might find that you need to remove it from the database.

Syntax

The syntax to a drop a function in MySQL is:

```
DROP FUNCTION [ IF EXISTS ] function_name;
```

function_name

The name of the function that you wish to drop.

Example

Let's look at an example of how to drop a function in MySQL.

For example:

```
DROP FUNCTION CalcIncome;
```

This example would drop the function called *CalcIncome*.

PROCEDURES:

In MySQL, a procedure is a stored program that you can pass parameters into. It does not return a value like a function does.

Create Procedure

Just as you can create procedures in other languages, you can create your own procedures in MySQL. Let's take a closer look.

Syntax

The syntax to create a procedure in MySQL is:

```
CREATE PROCEDURE procedure_name [ (parameter datatype [, parameter datatype])
]
```

```
BEGIN

   declaration_section

   executable_section

END;
```

procedure_name

The name to assign to this procedure in MySQL.

parameter

Optional. One or more parameters passed into the procedure. When creating a procedure, there are three types of parameters that can be declared:

1. **IN** - The parameter can be referenced by the procedure. The value of the parameter can not be overwritten by the procedure.
2. **OUT** - The parameter can not be referenced by the procedure, but the value of the parameter can be overwritten by the procedure.
3. **IN OUT** - The parameter can be referenced by the procedure and the value of the parameter can be overwritten by the procedure.

declaration_section

The place in the procedure where you declare local variables.

executable_section

The place in the procedure where you enter the code for the procedure.

Example

Let's look at an example that shows how to create a procedure in MySQL:

```
DELIMITER //

CREATE procedure CalcIncome ( OUT ending_value INT )

BEGIN

   DECLARE income INT;

   SET income = 50;

   label1: WHILE income <= 3000 DO
     SET income = income * 2;
   END WHILE label1;

   SET ending_value = income;

END; //
```

```
DELIMITER ;
```

You could then reference your new procedure as follows:

```
CALL CalcIncome (@variable_name);

SELECT @variable_name;
```

Drop procedure

Once you have created your procedure in MySQL, you might find that you need to remove it from the database.

Syntax

The syntax to a drop a procedure in MySQL is:

```
DROP procedure [ IF EXISTS ] procedure_name;
```

procedure_name

The name of the procedure that you wish to drop.

Example

Let's look at an example of how to drop a procedure in MySQL.

For example:

```
DROP procedure CalcIncome;
```

This example would drop the procedure called *CalcIncome*.