MySQL: Flow-Control Statements

Tushar B. Kute,
http://tusharkute.com

IF function

- The IF function is one of the parts of the MySQL control flow function, which returns a value based on the given conditions.
- In other words, the IF function is used for validating a function in MySQL.
- The IF function returns a value YES when the given condition evaluates to true and returns a NO value when the condition evaluates to false.
- It returns values either in a string or numeric form depending upon the context in which this function is used.
- Sometimes, this function is known as IF-ELSE and IF THAN ELSE function.

IF function

- The IF function takes three expressions, where the first expression will be evaluated.
- If the first expression evaluates to true, not null, and not zero, it returns the second expression. If the result is false, it returns the third expression.
- Syntax:

  IF (expression 1, expression 2, expression 3)

IF function

- Parameters:
- Expression 1
  - Required It is a value, which is used for validation.
- Expression 2
  - Optional It returns a value when the condition evaluates to true.
- Expression 3
  - Optional It returns a value when the condition evaluates to false.

IF function

- The return type of IF function can be calculated as follows:
- If expression 2 or expression 3 are both strings or produce a string, the result is always a string.
- If expression 2 or expression 3 gives a floating-point value, the result is always a floating-point value.
- If expression 2 or expression 3 is an integer, the result is always an integer.

Example:

- Example 1

  SELECT IF(200>350,'YES','NO') as result;
- In the above function, the (200>350) is a condition, which is evaluated.
- If the condition is true, it returns a value, YES, and if the condition is false, it returns NO.

Example:

- Example 2

  SELECT IF(251 = 251,' Correct','Wrong') as result;
- In the above function, the (251 = 251) is a condition, which is evaluated. If the condition is true, it returns value Correct, and if the condition is false, it returns Wrong output.

Example:

- Example 3

  SELECT IF(STRCMP('Sachin Tendulkar','Rahul Dravid')=0, 'Correct', 'Wrong') as value;
- The above example compares the two strings. If both the string is the same, it returns Correct. Otherwise, the IF function returns Wrong output.

Example with select

```
SELECT name, class,
    IF(marks>=60,"First Class","NO")
    As Result
FROM student;
```

Example with select

-
  ```
  SELECT *,
  IF(marks>=60 and class='TE',"YES","NO")
  As Result
  FROM student;
  ```

If statement

- The IF statement is used in stored programs that implement the basic conditional construct in MySQL. Based on a certain condition, it allows us to execute a set of SQL statements.
- It returns one of the three values True, False, or NULL.
- We can use this statement in three ways IF-THEN, IF-THEN-ELSE, IF-THEN-ELSEIF-ELSE clauses, and can terminate with END-IF.

If-then statement

- This statement executes a set of SQL queries based on certain conditions or expressions. The syntax of the IF-THEN statement is as follows:

```
IF condition THEN
   statements;
END IF;
```

- In the above syntax, we have to specify a condition for executing the code.
- If the statement evaluates to true, it will execute the statement between IF-THEN and END-IF. Otherwise, it will execute the statement following the END-IF.

Example:

```
DELIMITER $$
CREATE PROCEDURE myResult(original_rate float, OUT discount_rate
float)
    NO SQL
     BEGIN
       IF (original_rate>200) THEN
         SET discount_rate=original_rate*.5;
       ELSE
         SET discount_rate=original_rate;
       END IF;
       select discount_rate;
    END$$
DELIMITER ;
```

Example:

- Next, create two variables and set the value for both as below:

```
mysql> set @p = 150;
mysql> set @dp = 180;
```

- Now, call the stored procedure function to get the output.

```
mysql> call myResult(@p, @dp)
```

IF-THEN-ELSEIF-ELSE Statement

- If we want to execute a statement based on multiple conditions, this statement can be used. The syntax of the IF-THEN-ELSE statement is given below:

```
IF condition THEN
  statements;
ELSEIF elseif-condition THEN
  elseif-statements;
...
ELSE
  else-statements;
END IF;
```

Example:

```
DELIMITER $$
CREATE PROCEDURE myResult(original_rate float,OUT discount_rate float)
    NO SQL
     BEGIN
        IF (original_rate>500) THEN
           SET discount_rate=original_rate*.5;
       ELSEIF (original_rate<=500 AND original_rate>250) THEN
           SET discount_rate=original_rate*.8;
       ELSE
           SET discount_rate=original_rate;
        END IF;
        select discount_rate;
    END$$
DELIMITER ;
```

Example:

- Next, create two variables and set the value for both as below:

```
mysql> set @p = 150;
mysql> set @dp = 150;
```

- Now, call the stored procedure function to get the output.

```
mysql> call myResult(@p, @dp)
```

Case statements

- The CASE expression validates various conditions and returns the result when the first condition is true.
- Once the condition is met, it stops traversing and gives the output. If it will not find any condition true, it executes the else block.
- When the else block is not found, it returns a NULL value.
- The main goal of MySQL CASE statement is to deal with multiple IF statements in the SELECT clause.

Case statements

- Syntax:

```
CASE value
    WHEN [compare_value] THEN result
    [WHEN [compare_value] THEN result ...]
    [ELSE result]
END
```

Example:

- Example

```
mysql> SELECT CASE 1 WHEN 1 THEN 'one'
WHEN 2 THEN 'two' ELSE 'more' END;
```

Searched CASE statement

- The second method is to consider a search_condition in the WHEN clauses, and if it finds, return the result in the corresponding THEN clause.
- Otherwise, it will return the else clause. If else clause is not specified, it will return a NULL value.
- Syntax:

```
CASE
    WHEN [condition] THEN result
    [WHEN [condition] THEN result ...]
    [ELSE result]
END
```

Searched CASE statement

- `mysql> SELECT CASE BINARY 'B' WHEN 'a' THEN 1 WHEN 'b' THEN 2 END;`

Searched CASE statement

```
SELECT id, name, marks, class,
CASE class
   WHEN 'TE' THEN '3rd'
   WHEN 'BE' THEN 'Final'
   WHEN 'SE' THEN '2nd'
   ELSE 'NONE'
END AS year from student;
```

MySQL Function

- A stored function in MySQL is a set of SQL statements that perform some task/operation and return a single value.
- It is one of the types of stored programs in MySQL. When you will create a stored function, make sure that you have a CREATE ROUTINE database privilege.
- Generally, we used this function to encapsulate the common business rules or formulas reusable in stored programs or SQL statements.

MySQL Function

- The stored function is almost similar to the procedure in MySQL, but it has some differences that are as follows:
  - The function parameter may contain only the IN parameter but can't allow specifying this parameter, while the procedure can allow IN, OUT, INOUT parameters.
  - The stored function can return only a single value defined in the function header.
  - The stored function may also be called within SQL statements.
  - It may not produce a result set.

MySQL Function

- DELIMITER $$
  CREATE FUNCTION fun_name(fun_parameter(s))
  RETURNS datatype
  [NOT] {Characteristics}
  fun_body;

MySQL Function

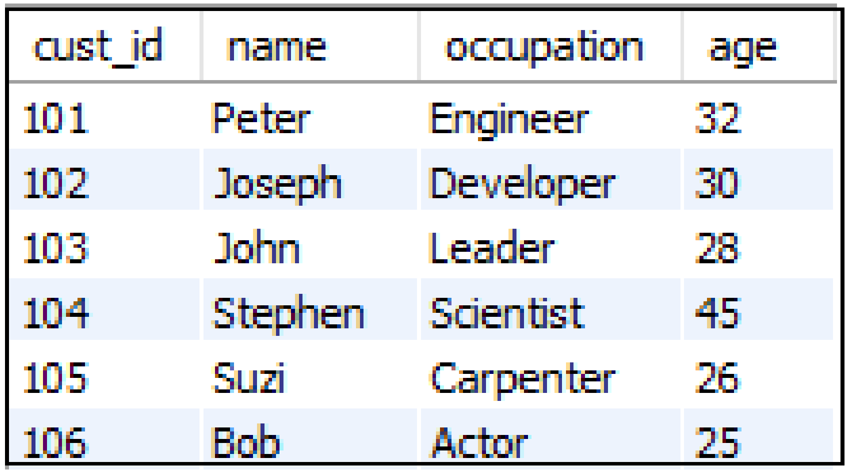
- fun_name
  - It is the name of the stored function that we want to create in a database. It should not be the same as the built-in function name of MySQL.
- fun_parameter
  - It contains the list of parameters used by the function body. It does not allow to specify IN, OUT, INOUT parameters.
- datatype
  - It is a data type of return value of the function. It should any valid MySQL data type.

MySQL Function

- characteristics
  - The CREATE FUNCTION statement only accepted when the characteristics (DETERMINISTIC, NO SQL, or READS SQL DATA) are defined in the declaration.
- fun_body
  - This parameter has a set of SQL statements to perform the operations. It requires at least one RETURN statement.
  - When the return statement is executed, the function will be terminated automatically. The function body is given below: BEGIN -- SQL statements END $$ DELIMITER

Example:

- Let us understand how stored function works in MySQL through the example.
- Suppose our database has a table named "customer" that contains the following data:



cust_id	name	occupation	age
101	Peter	Engineer	32
102	Joseph	Developer	30
103	John	Leader	28
104	Stephen	Scientist	45
105	Suzi	Carpenter	26
106	Bob	Actor	25

Example:

```
DELIMITER $$
CREATE FUNCTION Customer_Occupation(
    age int
)
RETURNS VARCHAR(20)
DETERMINISTIC
BEGIN
    DECLARE customer_occupation VARCHAR(20);
    IF age > 35 THEN
        SET customer_occupation = 'Scientist';
    ELSEIF (age <= 35 AND
            age >= 30) THEN
        SET customer_occupation = 'Engineer';
    ELSEIF age < 30 THEN
        SET customer_occupation = 'Actor';
    END IF;
    -- return the customer occupation
    RETURN (customer_occupation);
END$$
DELIMITER;
```

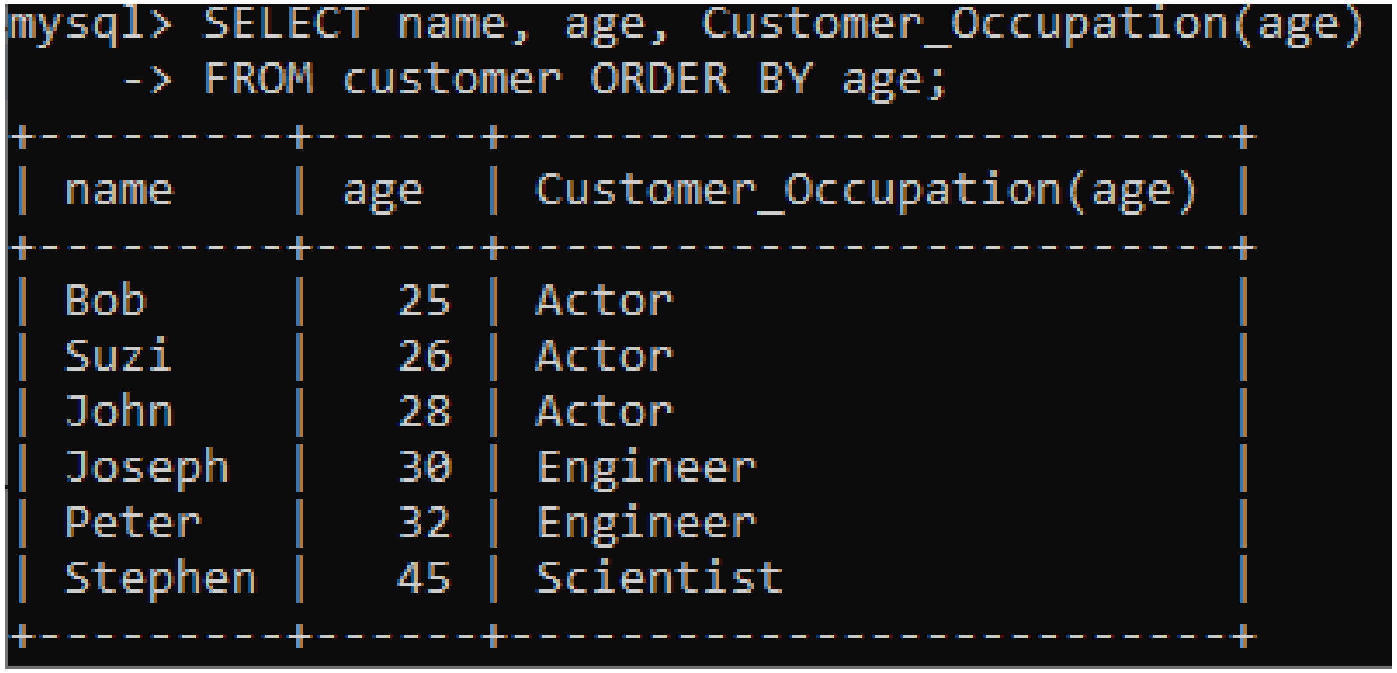
Example:

- Now, we are going to see how stored function is called with the SQL statement.
- The following statement uses customer_occupation stored function to get the result:

```
SELECT name, age, Customer_Occupation(age)
FROM customer ORDER BY age;
```

- It will give the output as below.

Example:



```
mysql> SELECT name, age, Customer_Occupation(age)
    -> FROM customer ORDER BY age;
+---------+------+--------------------------+
| name    | age  | Customer_Occupation(age) |
+---------+------+--------------------------+
| Bob     |   25 | Actor                    |
| Suzi    |   26 | Actor                    |
| John    |   28 | Actor                    |
| Joseph  |   30 | Engineer                 |
| Peter   |   32 | Engineer                 |
| Stephen |   45 | Scientist                |
+---------+------+--------------------------+
```

Loop

- Similar to other programming languages MySQL provides support for the flow control statements such as IF, CASE, ITERATE, LEAVE LOOP, WHILE, and REPEAT.
- You can use these statements in the stored programs (procedures), and RETURN in stored functions. You can use one Flow Control Statement with in another.
- The LOOP is a compound MySQL statement which is used to execute a single or set of statements repeatedly.

Loop

- Following is the syntax of the loop statement is MySQL –

```
begin_label: LOOP
  statement_list
END LOOP end_label
```

- Where, statement_list is a single or set of statements that are to be repeated. begin_label and end_label are the optional labels of the LOOP statement.
- The statement(s) in the LOOP are executed repeatedly till the loop is terminated. You can terminate the LOOP using the LEAVE statement.
- When used in a function the LOOP can also be terminated using the RETURN statement. Each statement in the LOOP ends with a semi colon (or. the current delimiter).

Loop

```
mysql> Delimiter //
mysql> CREATE procedure loopDemo()
   label:BEGIN
      DECLARE val INT ;
      DECLARE result VARCHAR(255);
      SET val =1;
      SET result = '';
         loop_label: LOOP
         IF val > 10 THEN
            LEAVE loop_label;
         END IF;
         SET result = CONCAT(result,val,',');
         SET val = val + 1;
         ITERATE loop_label;
      END LOOP;
      SELECT result;
      END//
mysql> Delimiter ;
```

While loop

- The WHILE is a compound MySQL statement which is used to execute a single or set of statements repeatedly as long as the specified condition is TRUE.
- Syntax:

```
begin_label: WHILE search_condition DO
  statement_list
END WHILE end_label
```

- Where, statement_list is a single or set of statements that are to be repeated. begin_label and end_label are the optional labels of the WHILE statement.

While loop

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE while_loop()
   BEGIN
      DECLARE num INT default 1;
      DECLARE res Varchar(50) default '';
      WHILE num < 78125 DO
         SET res = CONCAT(res,num,',');
         SET num = num*5;
      END While;
      SELECT res;
   END //
Query OK, 0 rows affected (0.38 sec)
mysql> DELIMITER ;
```

Repeat statement

- The REPEAT statement in MySQL is used to repeat the given set of statements (or statement) until the value of the given search condition is TRUE.
- The statement(s) in the LOOP ends with a semi colon (or. the current delimiter).
- Syntax:

```
begin_label: REPEAT
   statement_list
UNTIL search_condition
END REPEAT end_label
```

Repeat statement

```
DELIMITER //
CREATE PROCEDURE RepeatExample()
BEGIN
   DECLARE val INT;
   DECLARE squares INT;
   DECLARE res VARCHAR(100);
   SET val=1;
   SET squares=1;
   SET res = '';
   REPEAT
      SET squares = val*val;
      SET res = CONCAT(res, squares,',');
      SET val = val + 1;
   UNTIL val >= 10
   END REPEAT;
   SELECT res;
END//

DELIMITER ;
```

Thank you

@mitu_skillologies

@mITuSkillologies

@mitu_group

@mitu-skillologies

@MITUSkillologies

kaggle

@mituskillologies

Web Resources
https://mitu.co.in
http://tusharkute.com

@mituskillologies

contact@mitu.co.in

tushar@tusharkute.com