MySQL IF()

we are going to learn how **IF() function** works in MySQL. 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](https://www.javatpoint.com/mysql-functions). 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.

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**

1. IF ( expression 1, expression 2, expression 3)

### **Parameter**

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Requirement** | **Descriptions** |
| 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. |

* **SELECT** IF(200>350,'YES','NO');

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**.

**Output:**

NO

* **SELECT** IF(251 = 251,' Correct','Wrong');

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.

**Output:**

Correct

* **SELECT** IF(STRCMP('Rinky Ponting','Yuvraj Singh')=0, 'Correct', 'Wrong');

The above example compares the two strings. If both the string is the same, it returns **Correct**. Otherwise, the IF function returns **Wrong**output.

**Output:**

Wrong

In the given SQL query, the STRCMP function is used to compare two strings: 'Rinky Ponting' and 'Yuvraj Singh'. The STRCMP function returns an integer value that indicates the result of the comparison:

If the two strings are equal, STRCMP returns 0.

If the first string is greater than the second string, STRCMP returns a positive integer.

If the first string is less than the second string, STRCMP returns a negative integer.

The IF function is then used to create a conditional statement based on the result of the STRCMP function. If the result of STRCMP is 0 (i.e., the two strings are equal), the IF function returns 'Correct'. Otherwise, it returns 'Wrong'.

**CREATE TABLE student (**

**student\_id INT PRIMARY KEY,**

**first\_name VARCHAR(50),**

**last\_name VARCHAR(50),**

**class VARCHAR(10),**

**age INT**

**);**

**INSERT INTO student (student\_id, first\_name, last\_name, class, age)**

**VALUES**

**(1, 'John', 'Doe', '12th', 18),**

**(2, 'Jane', 'Smith', '11th', 20),**

**(3, 'Michael', 'Johnson', '10th', 22),**

**(4, 'Emma', 'Williams', '12th', 23),**

**(5, 'James', 'Brown', '11th', 24);**

**SELECT** last\_name,

IF(age>20,"Mature","Immature")

**As** Result

**FROM** student;

CREATE TABLE student1 (

student\_id INT AUTO\_INCREMENT PRIMARY KEY,

student\_name VARCHAR(50),

marks INT

);

INSERT INTO student1 (student\_name, marks)

VALUES

('John', 80),

('Sarah', 65),

('Michael', 95),

('Emily', 50),

('David', 70);

SELECT

student\_name,

marks,

CASE

WHEN marks >= 90 THEN 'Excellent'

WHEN marks >= 70 THEN 'Good'

ELSE 'Average'

END AS grade

FROM

student1;

# MySQL IFNULL()

This section helps you to learn about the MySQL IFNULL() function. The IFNULL function is a part of the MySQL control flow function used for handling NULL values.

The IFNULL function accepts two expressions, and if the first expression is **not null**, it returns the first arguments. If the first expression is **null**, it returns the second argument. This function returns either string or numeric value, depending on the context where it is used.

### **Syntax**

We can use the IFNULL function with the following syntax:

IFNULL (Expression1, Expression2)

It returns expression1 when the expression1 is not null. Otherwise, it will return expression2.

**SELECT** IFNULL(0,5);

In the above function, the MySQL statement checks the first expression. If the first expression is not NULL, it will return the first expression, which is zero.

**Output:**

0

**SELECT** IFNULL("Hello", "javaTpoint");

The above MySQL statement checks the first expression. If the first expression is not NULL, it will return the first expression, which is **'Hello'** value.

**Output:**

Hello

**SELECT** IFNULL(NULL,5);

The following MySQL statement checks the first expression. If the first expression is not NULL, it will return the first expression. Otherwise, it will return the second expression, which is five (5).

**Output:**

5

CREATE TABLE student\_contacts (

studentid INT UNSIGNED PRIMARY KEY,

contactname VARCHAR(45),

cellphone VARCHAR(20),

homephone VARCHAR(20)

);

INSERT INTO student\_contacts (studentid, contactname, cellphone, homephone)

VALUES

(2, 'Will', '3245712398', NULL),

(3, 'John', NULL, '4378957489'),

(4, 'Peter', NULL, '2123457870'),

(5, 'Kelly', '5678345609', NULL),

(6, 'Freeda', '4563482354', NULL);

**SELECT**

    contactname, IFNULL(cellphone, homephone) phone

**FROM**

    student\_contacts;

CREATE TABLE employeess (

employee\_name VARCHAR(50),

salary INT,

salary\_category VARCHAR(20)

);

INSERT INTO employeess (employee\_name, salary, salary\_category)

VALUES

('John', 50000, 'Low Salary'),

('Sarah', 60000, 'High Salary'),

('Michael', 75000, 'High Salary'),

('Emily', 45000, 'Low Salary'),

('David', 80000, 'High Salary');

SELECT

employee\_name,

salary,

IF(salary >= 80000, 'Very High Salary',

IF(salary >= 60000, 'High Salary',

IF(salary >= 40000, 'Moderate Salary', 'Low Salary')

)

) AS salary\_category

FROM

employeess;

**SQL AND, OR, AND OR**

| **employee\_id** | **employee\_name** | **department** | **salary** |
| --- | --- | --- | --- |
| 1 | John | IT | 50000 |
| 2 | Sarah | HR | 60000 |
| 3 | Michael | IT | 75000 |
| 4 | Emily | Finance | 45000 |
| 5 | David | IT | 80000 |

**SELECT employee\_name**

**FROM employees**

**WHERE department = 'IT' AND salary > 60000;**

**SELECT employee\_name, department**

**FROM employees**

**WHERE department = 'IT' OR salary > 60000;**

**SELECT employee\_name, department, salary**

**FROM employees**

**WHERE (department = 'IT' AND salary > 60000) OR department = 'Finance';**

MySQL Create Trigger

In this article, we are going to learn how to create the first trigger in MySQL. We can create a new trigger in MySQL by using the CREATE TRIGGER statement. It is to ensure that we have trigger privileges while using the CREATE TRIGGER command. The following is the basic syntax to create a trigger:

1. **CREATE** **TRIGGER** trigger\_name  trigger\_time trigger\_event
2. **ON** table\_name **FOR** EACH ROW
3. **BEGIN**
4. --variable declarations
5. --trigger code
6. **END**;

### **Why we need/use triggers in MySQL?**

We need/use triggers in MySQL due to the following features:

* Triggers help us to enforce business rules.
* Triggers help us to validate data even before they are inserted or updated.
* Triggers help us to keep a log of records like maintaining audit trails in tables.
* SQL triggers provide an alternative way to check the integrity of data.
* Triggers provide an alternative way to run the scheduled task.
* Triggers increases the performance of SQL queries because it does not need to compile each time the query is executed.
* Triggers reduce the client-side code that saves time and effort.
* Triggers help us to scale our application across different platforms.
* Triggers are easy to maintain.

Example:

**CREATE** **TABLE** employee11(

**name** **varchar**(45) NOT NULL,

    occupation **varchar**(35) NOT NULL,

    working\_date **date**,

    working\_hours **varchar**(10)

);

**INSERT** **INTO** employee11 **VALUES**

('Robin', 'Scientist', '2020-10-04', 12),

('Warner', 'Engineer', '2020-10-04', 10),

('Peter', 'Actor', '2020-10-04', 13),

('Marco', 'Doctor', '2020-10-04', 14),

('Brayden', 'Teacher', '2020-10-04', 12),

('Antonio', 'Business', '2020-10-04', 11);

**Create** **Trigger** before\_insert\_empworkinghours

BEFORE **INSERT** **ON** employee11 **FOR** EACH ROW

**BEGIN**

IF NEW.working\_hours < 0 **THEN** **SET** NEW.working\_hours = 0;

**END** IF;

**END** //

**INSERT** **INTO** employee11 **VALUES**

('Markus', 'Former', '2020-10-08', 14);

**INSERT** **INTO** employee11 **VALUES**

('Alexander', 'Actor', '2020-10-012', -13);

Delimiter ;

SHOW TRIGGERS IN employee; (database)

**DROP** **TRIGGER** employeedb.before\_update\_salaries;

Subqueries

Create database department

CREATE TABLE departments (

department\_id INT PRIMARY KEY,

department\_name VARCHAR(50),

salary INT

);

INSERT INTO departments (department\_id, department\_name, salary)

VALUES

(1, 'HR', 60000),

(2, 'IT', 70000),

(3, 'Finance', 75000);

CREATE TABLE employees (

employee\_id INT PRIMARY KEY,

employee\_name VARCHAR(50),

department\_id INT,

salary INT

);

INSERT INTO employees (employee\_id, employee\_name, department\_id, salary)

VALUES

(1, 'John', 1, 50000),

(2, 'Sarah', 2, 60000),

(3, 'Michael', 1, 75000),

(4, 'Emily', 3, 45000),

(5, 'David', 2, 80000);

**SELECT department\_name, (SELECT AVG(salary) FROM employees WHERE department\_id = 2) AS avg\_salary**

**FROM departments**

**WHERE department\_id = 2;**

In this query:

The outer query is selecting the department name from the departments table where the department\_id is 2 (which corresponds to the IT department).

The subquery (SELECT AVG(salary) FROM employees WHERE department\_id = 2) is used to find the average salary of employees in the IT department. The subquery filters the rows in the employees table based on the department\_id being 2 (IT department) and then calculates the average of the salary column for those rows.

The result of the subquery is displayed as the column avg\_salary in the outer query.

The result of the query will be:

department\_name avg\_salary

IT 70000

The query uses a subquery to calculate the average salary for the IT department and then includes that result in the final output alongside the department name. Subqueries in MySQL can be powerful tools for performing more complex and insightful data analysis within your database.

**SELECT employee\_name, salary**

**FROM employees**

**WHERE salary > (SELECT AVG(salary) FROM employees);**

The outer query selects the employee\_name and salary columns from the employees table.

The subquery (SELECT AVG(salary) FROM employees) calculates the average salary of all employees in the employees table.

The main query uses the WHERE clause to filter the rows and only selects employees whose salary is higher than the calculated average salary.