IN and BETWEEN Operator

IN Operator in MySQL:

The IN Operator in MySQL is used to search for specified values that match any value in the set of multiple values it accepts. The “IN” operator evaluates multiple values on a single data column. It displays the data row if any one of the given values is matched with the data column value. If none of the values matches, the SQL statement won’t return that data row. We generally will use the IN operator with WHERE clause to compare column or variable values with a set of multiple values. The “IN” operator is written as the word ‘IN’ followed by multiple values separated by a comma inside brackets.

Syntax: expression IN (value1, value2,.. valuen); where expression specifies the value to test and value1, value2, or valuen are the mentioned values to be test against expressions. If any of the values match then it evaluates to true otherwise false.

BETWEEN Operator in MySQL:

The BETWEEN Operator in MySQL is used to get the values within a specified range. The BETWEEN operator is used to compare the value in a column. This operator tests a range of values with a column value. The BETWEEN operator is written as the word “BETWEEN” followed by a range of values.

**Syntax: expression BETWEEN value1 AND value2;**where expression specifies a particular column and value1, value2 defines an inclusive range that expression is compared to.

**Understanding IN and BETWEEN Operator in MySQL:**

Let us understand IN and BETWEEN Operator in MySQL with Examples. We are going to use the following Employee table to understand the IN and BETWEEN operators.

**Syntax: expression BETWEEN value1 AND value2;**where expression specifies a particular column and value1, value2 defines an inclusive range that expression is compared to.

**Understanding IN and BETWEEN Operator in MySQL:**

Let us understand IN and BETWEEN Operator in MySQL with Examples. We are going to use the following Employee table to understand the IN and BETWEEN operators.

CREATE DATABASE company;

USE company;

CREATE TABLE employee (

Id INT PRIMARY KEY,

Name VARCHAR(45) NOT NULL,

Department VARCHAR(45) NOT NULL,

Salary FLOAT NOT NULL,

Gender VARCHAR(45) NOT NULL,

Age INT NOT NULL,

City VARCHAR(45) NOT NULL

);

INSERT INTO employee (Id, `Name`, Department, Salary, Gender, Age, City) VALUES (1001, 'John Doe', 'IT', 35000, 'Male', 25, 'London');

INSERT INTO employee (Id, `Name`, Department, Salary, Gender, Age, City) VALUES (1002, 'Mary Smith', 'HR', 45000, 'Female', 27, 'London');

INSERT INTO employee (Id, `Name`, Department, Salary, Gender, Age, City) VALUES (1003, 'James Brown', 'Finance', 50000, 'Male', 28, 'London');

INSERT INTO employee (Id, `Name`, Department, Salary, Gender, Age, City) VALUES (1004, 'Mike Walker', 'Finance', 50000, 'Male', 28, 'London');

INSERT INTO employee (Id, `Name`, Department, Salary, Gender, Age, City) VALUES (1005, 'Linda Jones', 'HR', 75000, 'Female', 26, 'London');

INSERT INTO employee (Id, `Name`, Department, Salary, Gender, Age, City) VALUES (1006, 'Anurag Mohanty', 'IT', 35000, 'Male', 25, 'Mumbai');

INSERT INTO employee (Id, `Name`, Department, Salary, Gender, Age, City) VALUES (1007, 'Priyanla Dewangan', 'HR', 45000, 'Female', 27, 'Mumbai');

INSERT INTO employee (Id, `Name`, Department, Salary, Gender, Age, City) VALUES (1008, 'Sambit Mohanty', 'IT', 50000, 'Male', 28, 'Mumbai');

INSERT INTO employee (Id, `Name`, Department, Salary, Gender, Age, City) VALUES (1009, 'Pranaya Kumar', 'IT', 50000, 'Male', 28, 'Mumbai');

INSERT INTO employee (Id, `Name`, Department, Salary, Gender, Age, City) VALUES (1010, 'Hina Sharma', 'HR', 75000, 'Female', 26, 'Mumbai');

INSERT INTO employee (Id, `Name`, Department, Salary, Gender, Age, City) VALUES (1011, 'Pramod Panda', 'IT', 45000, 'Male', 27, 'Mumbai');

INSERT INTO employee (Id, `Name`, Department, Salary, Gender, Age, City) VALUES (1012, 'Preety Tiwary', 'HR', 55000, 'Female', 28, 'Mumbai');

INSERT INTO employee (Id, `Name`, Department, Salary, Gender, Age, City) VALUES (1013, 'Santosh Dube', 'IT', 52000, 'Male', 28, 'Mumbai');

INSERT INTO employee (Id, `Name`, Department, Salary, Gender, Age, City) VALUES (1014, 'Sara Talour', 'HR', 85000, 'Female', 26, 'London');

INSERT INTO employee (Id, `Name`, Department, Salary, Gender, Age, City) VALUES (1015, 'Pamela Kar', 'Finance', 70000, 'Female', 26, 'London');

IN Operator Example in MySQL:

If you want to fetch all the employees from the employee table whose department is either IT or Finance, then you need to write the SQL Query using IN Operator as shown below.

**SELECT \* FROM Employee WHERE Department IN (‘IT’, ‘Finance’);**

Once you execute the above SQL Query, you will get the following result set which includes only the IT and Finance departments employees.

**NOT IN Operator Example in MySQL:**

The NOT IN Operator in MySQL is just the opposite of IN Operator. The IN operator in MySQL takes a set of values and then returns the records whose column values matched with the values it has. But if you use the NOT keyword along with the IN operator in MySQL, then it will return data where the column value not in the set of values. For example, the following SQL query will return all the employees from the employee table where the Department not in IT and Finance. In other words, we can say that the following SQL Query is going to return only the HR department employees.

**SELECT \* FROM Employee WHERE Department NOT IN (‘IT’, ‘Finance’);**

**Advantages of IN Operator over OR Operator in MySQL**

Although both IN and OR going to provide the same results, IN condition is more preferable because it has the minimum number of codes as compared to OR condition.

For example, the following SQL query returns all the employees who belong to the IT or HR department using the MySQL OR Operator.  
**SELECT \* FROM Employee WHERE (Department = ‘IT’ OR Department = ‘HR’);**

The following SQL Query also returns all the employees who belong to the IT or HR department using the MySQL IN Operator.  
**SELECT \* FROM Employee WHERE Department IN (‘IT’, ‘HR’);**

**Example: Fetch all the employees whose age is either 25 or 26**

Our requirement is to fetch all the employees whose age is either 25 or 26. Then using IN operator we can get the results very easily. The following SQL Query will retrieve all the employees from the employee table where the employee age is either 25 or 26.

**SELECT \* FROM Employee WHERE Age IN (25, 26);**

**Between Operator Example in MySQL:**

If you want to fetch all the employees from the employee table where the employee age is between 25 and 27, then we could write the SQL Query using the Between Operator as shown below.

**SELECT \* FROM Employee WHERE Age BETWEEN 25 AND 27;**

Once you execute the above SQL Query, you will get the following result set which includes only the employees whose age is between 25 and 27.

**Points to Remember while working with Between Operator in MySQL:**

1. Between Operator in MySQL returns true if the operand is within a range.
2. The Between Operator will return records including the starting and ending values.
3. Between operator support only the AND operator in MySQL.
4. The BETWEEN Operator takes the values from small to big range in the query.

**NOT BETWEEN Operator Example in MySQL:**

The NOT BETWEEN Operator in MySQL is just the opposite of BETWEEN Operator. the BETWEEN operator in MySQL will return records within a range including the starting and ending values specified in the Between Operator. But if you use the NOT keyword along with the BETWEEN operator then it will return data where the column values not in between the range values.

For example, the following SQL query will return all the employees from the employee table where the Age not between 25 and 27. The point that you also need to remember is it is also not going to return the employees whose age is 25 and 37.

**SELECT \* FROM Employee WHERE Age NOT BETWEEN 25 AND 27;**