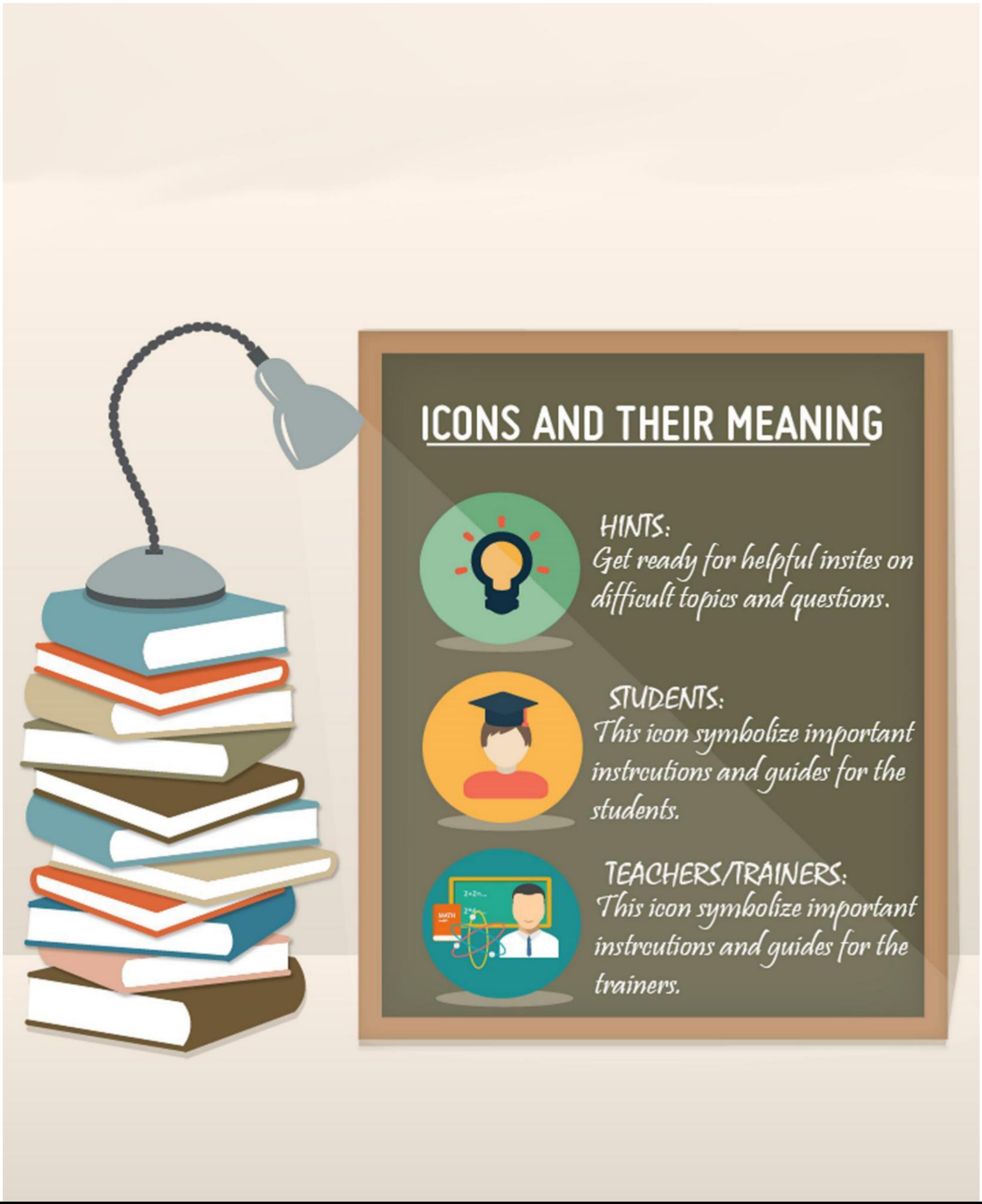
MySQL

MANUAL V8.3

MODULE CODE:

ANUDIP FOUNDATION


ICONS AND THEIR MEANING
HINTS:
Get ready for helpful insites on difficult topics and questions.
STUDENTS:
This icon symbolize important instrcutions and guides for the students.
TEACHERS/TRAINERS:
This icon symbolize important instrcutions and guides for the trainers.

Use of Boolean Operators BETWEEN, LIKE, IN, IS, IS NOT

Objective: After completing this lesson you will be able to : * Gain an understanding of the In, Between and Like operators in MySQL * Use the Limit clause * Use the Null operator	**Materials Required:** 1. Computer 2. Internet access
Theory Duration: 120 minutes	**Practical Duration:** 0 minute
Total Duration: 120 minutes	

Chapter 23

In Operator

MySQL IN operator enables users to determine if a value matches any value in a set, or matches a value returned by a subquery.

IN operator syntax -

```
SELECT
   column1,column2,...
FROM
   table_name
WHERE
    (expr|column_1) IN ('value1','value2',...);
```

* An expression (expr) or column with an IN operator in the WHERE clause.

* Values separated by commas (,)

* IN operator returns 1 if column_1 value or expression result is equal to a listed value. 0 is returned otherwise.

* MySQL performs some steps if listed values are constants

* Evaluates values based on column type of expression result

* Sorts values

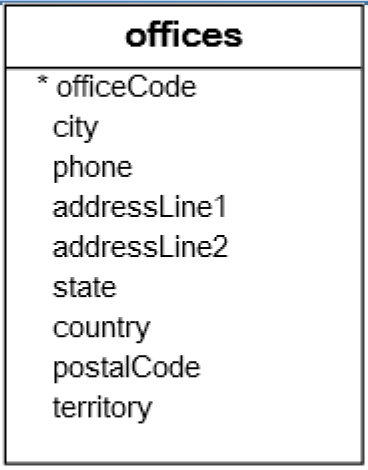
* Uses binary search algorithm for value searching

* The IN operator returns NULL if an expression or value in the list is NULL.

The IN operator and NOT operator can be combined to decide if a given value does not match in a subquery or a list of values. The IN operator can also be utilized within the WHERE clause of statements like DELETE and UPDATE.

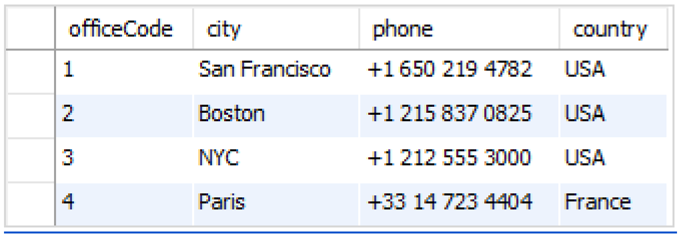
MySQL IN operator examples

IN operator example - ‘offices’ is the sample database in this example -



The user attempts to identify offices in France and the U.S. with the IN operator -

```
SELECT
    officeCode,
    city,
    phone,
    country
FROM
    offices
WHERE
    country IN ('USA' , 'France');
```



officeCode	city	phone	country
1	San Francisco	+1 650 219 4782	USA
2	Boston	+1 215 837 0825	USA
3	NYC	+1 212 555 3000	USA
4	Paris	+33 14 723 4404	France

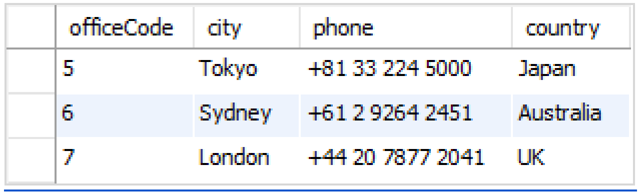
Using the OR operator can produce the same result -

```
SELECT

    officeCode,
    city,
    phone
FROM
    offices
WHERE
    country = 'USA' OR country = 'France';
```

But, using the OR operator is impractical if there are too many values in a list. This is why the IN operator is used for making the query shorter.

NOT IN can be used in the WHERE clause for showing offices that are not in France or USA. Query -

```
SELECT

    officeCode,
    city,
    phone
FROM
    offices
WHERE
    country NOT IN ('USA' , 'France');
```

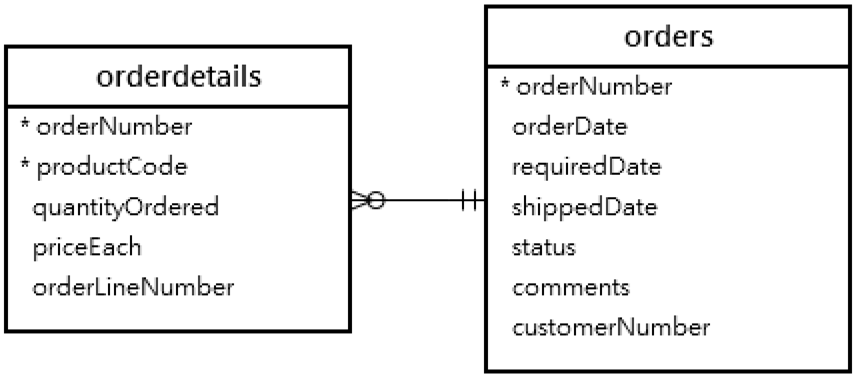


officeCode	city	phone	country
5	Tokyo	+81 33 224 5000	Japan
6	Sydney	+61 2 9264 2451	Australia
7	London	+44 20 7877 2041	UK

MySQL IN with a subquery

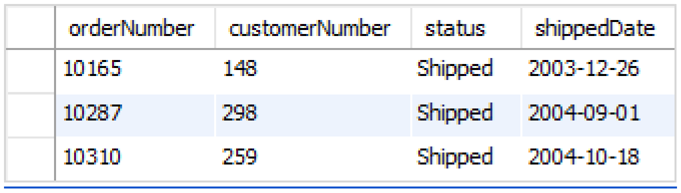
The IN operator can be used with a subquery. The subquery fetches a list of values from single or multiple tables. These values are then used as IN operator values.

Example - A sample database with the orders and orderDetails tables



Here, the user attempts to find orders with total values more than 60,000. Using the IN operator -

```
SELECT
    orderNumber,
    customerNumber,
    status,
    shippedDate
FROM
    orders
WHERE orderNumber IN
(
    SELECT
        orderNumber
    FROM
        orderDetails
    GROUP BY

        orderNumber
    HAVING SUM(quantityOrdered * priceEach) > 60000
);
```

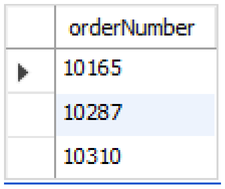


	orderNumber	customerNumber	status	shippedDate
	10165	148	Shipped	2003-12-26
	10287	298	Shipped	2004-09-01
	10310	259	Shipped	2004-10-18

The above query can be dissected into two different queries -

The subquery returns a list containing order numbers with values bigger than 60,000. This is done by utilizing the GROUP BY and HAVING clauses -

```
SELECT
    orderNumber
FROM
    orderDetails
GROUP BY
    orderNumber
HAVING
    SUM(quantityOrdered * priceEach) > 60000;
```



	orderNumber
▶	10165
	10287
	10310

In the other query, the IN operator is utilized within the WHERE clause by the other query for fetching data from the table -

```
SELECT
    orderNumber,
    customerNumber,
    status,
    shippedDate
FROM
    orders
WHERE
    orderNumber IN (10165,10287,10310);
```

Between Operator

The BETWEEN operator is a logical operator enabling users to specify whether a value is within a range, or not. It is used in the WHERE clause of the UPDATE, SELECT and DELETE statements.

BETWEEN operator syntax -

```
expr [NOT] BETWEEN begin_expr AND end_expr;
```

The expr expression is used for testing within the range between begin_expr and end_expr. All of these expressions have to have the same data type.

The BETWEEN operator gives a true output if the expr value is greater than or equal to (>=) the begin_expr value and less than or equal to (<=) the end_expr. It returns zero otherwise.

The NOT BETWEEN statement returns true if expr expression value is less than (<) the begin_expr value or greater than the end_expr value. It returns zero otherwise.

The BETWEEN operator returns NULL if any expression is NULL.

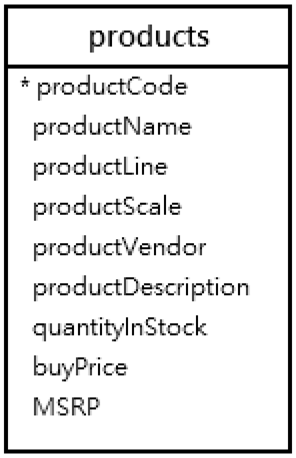
The greater than (>) and less than (<) operators can be used for specifying an exclusive range.

MySQL BETWEEN operator examples

BETWEEN operator examples -

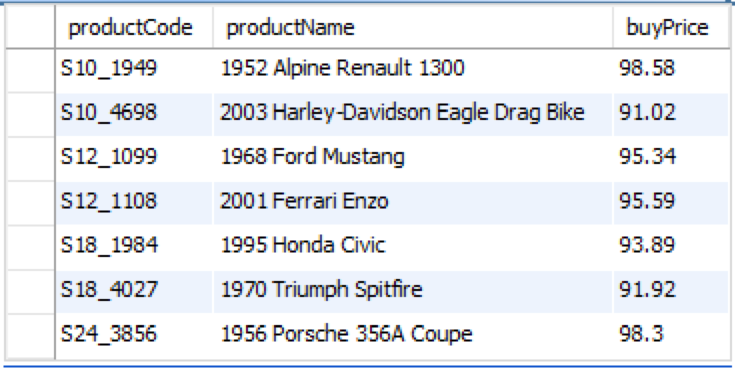
1) MySQL BETWEEN with number examples -

The sample products table -



The below example utilizes the BETWEEN operator for finding products with prices in the range of 90 to 100 -

```
SELECT
    productCode,
    productName,
    buyPrice
FROM
    products
WHERE
    buyPrice BETWEEN 90 AND 100;
```



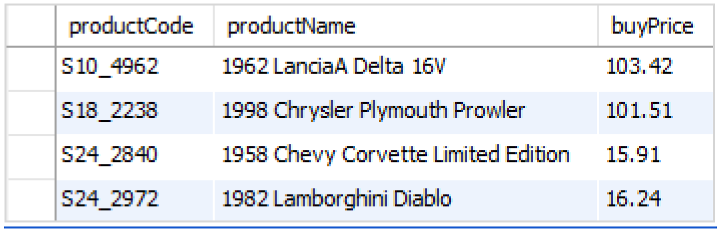
	productCode	productName	buyPrice
	S10_1949	1952 Alpine Renault 1300	98.58
	S10_4698	2003 Harley-Davidson Eagle Drag Bike	91.02
	S12_1099	1968 Ford Mustang	95.34
	S12_1108	2001 Ferrari Enzo	95.59
	S18_1984	1995 Honda Civic	93.89
	S18_4027	1970 Triumph Spitfire	91.92
	S24_3856	1956 Porsche 356A Coupe	98.3

The query below utilizes the less than or equal (<=) and greater than or equal (>=) operators instead of using the BETWEEN operator. The result is the same -

```
SELECT
    productCode,
    productName,
    buyPrice
FROM
    products
WHERE
    buyPrice >= 90 AND buyPrice <= 100;
```

In this part of the example, the BETWEEN and NOT operators are combined to find products with buy prices not between $20 and $100. Query example -

```
SELECT
    productCode,
    productName,
    buyPrice
FROM
    products
WHERE
    buyPrice NOT BETWEEN 20 AND 100;
```



	productCode	productName	buyPrice
	S10_4962	1962 LanciaA Delta 16V	103.42
	S18_2238	1998 Chrysler Plymouth Prowler	101.51
	S24_2840	1958 Chevy Corvette Limited Edition	15.91
	S24_2972	1982 Lamborghini Diablo	16.24

The query can be rewritten with the greater than (>), less than (<), and logical operators (AND). Query example -

```
SELECT
    productCode,
    productName,
    buyPrice
FROM

    products
```

```
WHERE
    buyPrice < 20 OR buyPrice > 100;
```

2) MySQL BETWEEN with dates example -

While using the BETWEEN operator with date values, a user should utilize type cast for explicit column type conversion to the DATE type.

The example returns orders which have the required between 02/01/2005 to 02/25/2005 -

```
SELECT
   orderNumber,
   requiredDate,
   status
FROM
   orders
WHERE
   requireddate BETWEEN
     CAST('2005-02-01' AS DATE) AND
     CAST('2005-02-25' AS DATE);
```

Output - The dates between the range are shown in the result. The CAST operator is used for converting the strings to DATE values.

Like Operator

A user can utilize the WHERE clause having an 'equal to' symbol is adequate for an exact match. However, there may be requirements for users to filter out results where a table has to have a particular value. The SQL LIKE clause can be used with the WHERE clause for such requirements.

The SQL LIKE clause can be used with the % character to function like the * character. The LIKE clause functions similar to the = sign without the % character.

Syntax

The code below has an SQL SELECT command syntax along with the LIKE clause for fetching MySQL table data.

```
SELECT field1, field2,...fieldN table_name1, table_name2...
WHERE field1 LIKE condition1 [AND [OR]] filed2 = 'somevalue'
```

* The LIKE clause can be used with the WHERE clause and also used instead of the equals to sign.

* LIKE paired with the % sign functions similar to a meta character search.

* WHERE...LIKE can be used with the SQL UPDATE or DELETE command for specifying a condition.

LIKE clause at the Command Prompt

Users can utilize the SQL SELECT command with WHERE...LIKE for retrieving chosen data from a MySQL table. In this example the tutorialz_tbl is used.

Example

In this example all records with author name ending with son are returned -

```
root@host# mysql -u root -p password;
Enter password:*******
mysql> use TUTORIALS;
Database changed
mysql> SELECT * from tutorials_tbl
   -> WHERE tutorial_author LIKE '%son';
                  +-------------+----------------+-----------------+-----------------+
                  | tutorial_id | tutorial_title | tutorial_author | submission_date |
                  +-------------+----------------+-----------------+-----------------+
                  |      3      |  HTML Tutorial |     Thompson     |    2009-06-22    |
                  +-------------+----------------+-----------------+-----------------+
                  1 rows in set (0.01 sec)


mysql>
```

LIKE clause inside PHP Script

The mysql_query() PHP function can utilize syntax like the WHERE...LIKE clause. This function enables the execution of the SQL command and the mysql_fetch_array() PHP function is used for retrieving data. This is possible if the WHERE...LIKE clause is utilized with the SELECT command.

However, a PHP function call is not required if the WHERE...LIKE clause is used with the UPDATE or DELETE command.

Example

This example with the tutorials_tbl table returns all records where author name had the term 'son'.

```
<?php
   $dbhost = 'localhost:3036';
   $dbuser = 'root';
   $dbpass = 'rootpassword';
   $conn = mysql_connect($dbhost, $dbuser, $dbpass);


   if(! $conn ) {
      die('Could not connect: ' . mysql_error());
   }
   $sql = 'SELECT tutorial_id, tutorial_title,
      tutorial_author, submission_date
      FROM tutorials_tbl
      WHERE tutorial_author LIKE "%son%"';


   mysql_select_db('TUTORIALS');
   $retval = mysql_query( $sql, $conn );

   if(! $retval ) {
      die('Could not get data: ' . mysql_error());

}
   while($row = mysql_fetch_array($retval, MYSQL_ASSOC)) {
      echo "Tutorial ID :{$row['tutorial_id']}  <br> ".
         "Title: {$row['tutorial_title']} <br> ".
         "Author: {$row['tutorial_author']} <br> ".
         "Submission Date : {$row['submission_date']} <br> ".
         "--------------------------------<br>";
   }
   echo "Fetched data successfully\n";
   mysql_close($conn);
?>
```

Limit Clause

The LIMIT clause of MySQL is utilized with the SELECT statement to limit the number of rows in a set of results. The LIMIT clause can accept one or two arguments which are count and offset. Both parameters can have zero or positive integer values.

Offset is used for specifying the first row's offset to be returned.

Count specifies the maximum number of rows which have to be returned.

The LIMIT clause accepts either one or two parameters.

When two parameters have been specified, the first is the offset and the second one is the count.

When one parameter is specified, the second parameter represents the number of rows which will be returned from the start of the result set.

Syntax:

```
SELECT column1, column2, ...
FROM table_name
LIMIT offset, count;
```

In the example below, the user has a table named 'Data'. The table has three columns named Firstname, Lastname and Age.

Firstname	Lastname	Age
Rupesh	Singh	26
Suresh	Sharma	29
Ram	Lal	24
Shyam	Kumar	30
Anup	Jain	40

In this step, the user attempts to fetch the first three rows of the 'Data' table -

```
SELECT * FROM Data LIMIT 3;
```

The query for fetching rows 2 to 3 -

```
SELECT * FROM Data LIMIT 1, 2;
```

PHP query implementation for displaying two rows of the table with the LIMIT clause -

***Limit Clause using Procedural Method**

```
< ? php $link = mysqli_connect("localhost", "root", "", "Mydb");

if ($link == = false) {
    die("ERROR: Could not connect. ".mysqli_connect_error());
}

$sql = "SELECT * FROM Data LIMIT 2";
if ($res = mysqli_query($link, $sql)) {
    if (mysqli_num_rows($res) > 0) {
        echo "<table>";
        echo "<tr>";
        echo "<th>Firstname</th>";
        echo "<th>Lastname</th>";
        echo "<th>Age</th>";
        echo "</tr>";
        while ($row = mysqli_fetch_array($res)) {
            echo "<tr>";
            echo "<td>".$row['Firstname']."</td>";
            echo "<td>".$row['Lastname']."</td>";
            echo "<td>".$row['Age']."</td>";
            echo "</tr>";
        }
        echo "</table>";
        mysqli_free_result($res);
    }
    else {
        echo "No matching records are found.";
    }
}
else {
    echo "ERROR: Could not execute $sql. ".mysqli_error($link);
}

mysqli_close($link);
? >
```

Output :

Firstname	Lastname	Age
Rupesh	Singh	26
Suresh	Sharma	29

Explanation:

* The “res” variable stores data returned by the mysql_query() function.

* The next row is returned from the res() set everytime mysqli_fetch_array() is invoked.

* Users can utilize the while loop for all rows of the 'data' table.

Limit Clause using Object Oriented Method

```
< ? php $mysqli = new mysqli("localhost", "root", "", "Mydb");

if ($mysqli == = false) {
   die("ERROR: Could not connect. ".$mysqli->connect_error);
}

$sql = "SELECT * FROM Data LIMIT 2";
if ($res = $mysqli->query($sql)) {
   if ($res->num_rows > 0) {
      echo "<table>";
      echo "<tr>";
      echo "<th>Firstname</th>";
      echo "<th>Lastname</th>";
      echo "<th>Age</th>";
      echo "</tr>";
      while ($row = $res->fetch_array()) {
         echo "<tr>";
         echo "<td>".$row['Firstname']."</td>";
         echo "<td>".$row['Lastname']."</td>";
         echo "<td>".$row['Age']."</td>";
         echo "</tr>";
      }


       echo "</table>";
      $res->free();
   }
   else {
      echo "No matching records are found.";
   }
}
else {
   echo "ERROR: Could not execute $sql. ".$mysqli->error;

}

$mysqli->close();
```

? >

Output :

Firstname	Lastname	Age
Rupesh	Singh	26
Suresh	Sharma	29

***Limit Clause using PDO Method**

Output :

Firstname	Lastname	Age
Rupesh	Singh	26
Suresh	Sharma	29

Is Null Operator

The IS NULL operator is utilized for testing if a value is NULL or not.

IS NULL operator syntax -

value IS NULL

The expression returns true if the value is NULL. Otherwise, a false is returned.

MySQL utilizes TINYINT(1) for representing BOOLEAN values. It translates to true meaning 1 and false meaning 0.

IS NULL is a comparison operators which can be used anywhere an operator is used i.e the WHERE and SELECT clauses.

Example -

```
SELECT 1 IS NULL,   -- 0

       0 IS NULL,   -- 0

       NULL IS NULL; -- 1
```

The IS NOT NULL operator is used for checking if a value is not NULL -

Syntax -

value IS NOT NULL

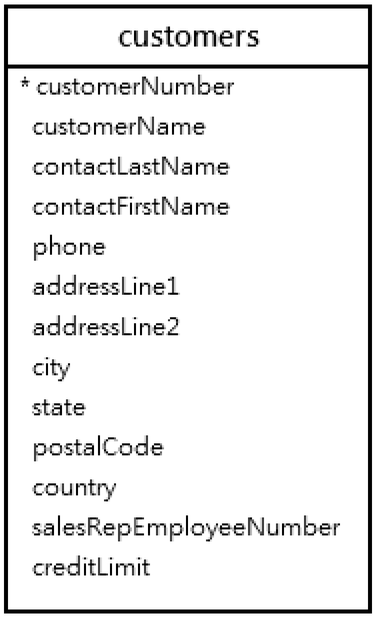
If value is not NULL, true (1) is returned. Otherwise, false (0) is returned.

Example -

```
SELECT 1 IS NOT NULL, -- 1
       0 IS NOT NULL, -- 1
       NULL IS NOT NULL; -- 0
```

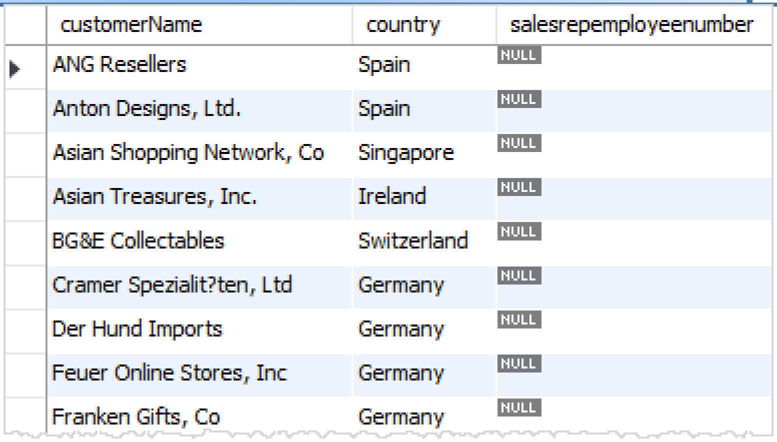
MySQL IS NULL examples

For this example, a table named customers is used -



IS NULL operator used for finding Customers without a representative -

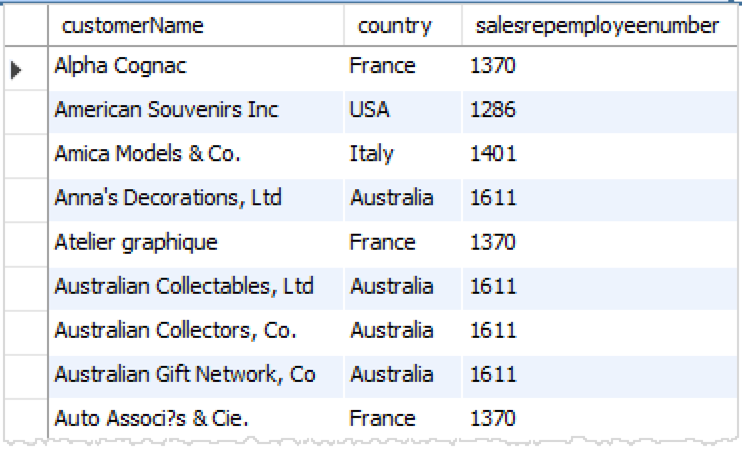
```
SELECT
    customerName,
    country,
    salesrepemployeenumber
FROM
    customers
WHERE
    salesrepemployeenumber IS NULL

ORDER BY
    customerName;
```



customerName	country	salesrepemployeenumber
ANG Resellers	Spain	NULL
Anton Designs, Ltd.	Spain	NULL
Asian Shopping Network, Co	Singapore	NULL
Asian Treasures, Inc.	Ireland	NULL
BG&E Collectables	Switzerland	NULL
Cramer Spezialit?ten, Ltd	Germany	NULL
Der Hund Imports	Germany	NULL
Feuer Online Stores, Inc	Germany	NULL
Franken Gifts, Co	Germany	NULL

The below example utilizes the IS NOT NULL operator for fetching customers with a sales rep -

```
SELECT
    customerName,
    country,
    salesrepemployeenumber
FROM
    customers
WHERE
    salesrepemployeenumber IS NOT NULL
ORDER BY
   customerName;
```



customerName	country	salesrepemployeenumber
Alpha Cognac	France	1370
American Souvenirs Inc	USA	1286
Amica Models & Co.	Italy	1401
Anna's Decorations, Ltd	Australia	1611
Atelier graphique	France	1370
Australian Collectables, Ltd	Australia	1611
Australian Collectors, Co.	Australia	1611
Australian Gift Network, Co	Australia	1611
Auto Associ?s & Cie.	France	1370

MySQL IS NULL – advanced features

MySQL offers support for several IS NULL advanced features for ODBC compatibility.

Example with the date ‘0000-00-00’

The IS NULL operator can be used for finding rows in a scenario where there is a NOT NULL constraint in a DATETIME or DATE column with the special date ‘0000-00-00’.

In this example, a table named projects is first created -

```
CREATE TABLE IF NOT EXISTS projects (

    id INT AUTO_INCREMENT,

    title VARCHAR(255),

    begin_date DATE NOT NULL,

    complete_date DATE NOT NULL,

    PRIMARY KEY(id)

);
```

The second step is inserting rows into the 'projects' table -

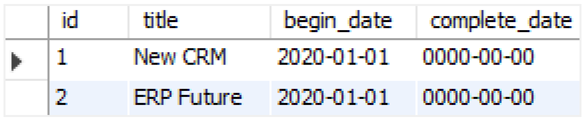
```
INSERT INTO projects(title,begin_date, complete_date)

VALUES('New CRM','2020-01-01','0000-00-00'),

      ('ERP Future','2020-01-01','0000-00-00'),

      ('VR','2020-01-01','2030-01-01');
```

In the third step, the IS NULL operator is utilized for selecting rows which has '0000-00-00' in the complete_date column.

SELECT *

FROM projects

WHERE complete_date IS NULL;



	id	title	begin_date	complete_date
▶	1	New CRM	2020-01-01	0000-00-00
	2	ERP Future	2020-01-01	0000-00-00

Instructions: The progress of students will be assessed with the exercises mentioned below.

MCQ

1. What does the IN operator in MySQL determine?

a) if value matches a set's value

b) if value matches subquery returned value

c) both a and b

d) none of the mentioned

2. What happens when listed values are constants?

a) values are sorted

b) utilizes binary search for value searching

c) both a and b

d) none of the mentioned

3. Which operator can the IN operator not be used with?

a) UPDATE

b) DELETE

c) WHERE

d) none of the mentioned

4. Which operator is used with the NOT operator if a value does not match any subquery value?

a) OR

b) IN

c) both a and b

d) none of the mentioned

5. Which operator is used for making a query shorter when there are too many values?

a) IN

b) OR

c) IN OR

d) none of the mentioned

6. Using a subquery with the _______ operator can help a user fetch values from multiple tables.

a) IN

b) OR

c) IN OR

d) none of the mentioned

7. Which of these clauses can be utilized to find values more than a specified value?

a) HAVING

b) GROUP BY

c) both a and b

d) none of the mentioned

8. Which of these operators is utilized for specifying if a value is not within a range?

a) SELECT

b) BETWEEN

c) SORT

d) none of the mentioned

9. The SQL SELECT command is used with __________ for fetching chosen a MySQL table's data.

a) WHERE

b) WHERE..SORT

c) WHERE...LIKE

d) none of the mentioned

10. The LIMIT clause can accept ____________.

a) two parameters

b) one parameter

c) both a and b

d) none of the mentioned