MySQLTUTORIAL

MySQL ORDER BY

Summary: in this tutorial, you will learn how to sort the rows in a result set using the MySQL `ORDER BY` clause.

Introduction to the MySQL ORDER BY clause

When you use the `SELECT (https://www.mysqltutorial.org/mysql-select-statement-query-data.aspx)` statement to query data from a table, the order of rows in the result set is unspecified. To sort the rows in the result set, you add the `ORDER BY` clause to the `SELECT` statement.

The following illustrates the syntax of the `ORDER BY` clause:

```
SELECT
   select_list
FROM
   table_name
ORDER BY
   column1 [ASC|DESC],
   column2 [ASC|DESC],
   ...;
```

In this syntax, you specify the one or more columns that you want to sort after the `ORDER BY` clause.

The `ASC` stands for ascending and the `DESC` stands for descending. You use `ASC` to sort the result set in ascending order and `DESC` to sort the result set in descending order respectively.

This `ORDER BY` clause sorts the result set by the values in the `column1` in ascending order:

```
ORDER BY column1 ASC;
```

And this `ORDER BY` clause sorts the result set by the values in the `column1` in descending order:

```
ORDER BY column1 DESC;
```

By default, the `ORDER BY` clause uses `ASC` if you don't explicitly specify any option. Therefore, the following `ORDER BY` clauses are equivalent:

```
ORDER BY column1 ASC;
```

and

```
ORDER BY column1;
```

If you want to sort the result set by multiple columns, you specify a comma-separated list of columns in the `ORDER BY` clause:

```
ORDER BY
   column1,
   column2;
```

In this case, the `ORDER BY` clause sorts the result set by `column1` in ascending order first and sorts the sorted result set by `column2` in ascending order.

It is possible to sort the result set by a column in ascending order and then by another column in descending order:

```
ORDER BY
    column1 ASC,
    column2 DESC;
```

In this case, the `ORDER BY` clause:

- First, sort the result set by the values in the `column1` in ascending order.
- Then, sort the sorted result set by the values in the `column2` in descending order. Note that the order of values in the `column1` will not change in this step, only the order of values in the `column2` changes.

When executing the `SELECT` statement with an `ORDER BY` clause, MySQL always evaluates the `ORDER BY` clause after the `FROM` and `SELECT` clauses:

MySQL ORDER BY examples

We'll use the `customers` table from the sample database (https://www.mysqltutorial.org/mysql-sample-database.aspx) for the demonstration.

A) Using MySQL ORDER BY clause to sort the result set by one column example

The following query uses the `ORDER BY` clause to sort the customers by their last names in ascending order.

```
SELECT
        contactLastname,
        contactFirstname
FROM
        customers
ORDER BY
        contactLastname;
```

Try It Out ›

Output:

```
+-----------------+------------------+
| contactLastname | contactFirstname |
+-----------------+------------------+
| Accorti         | Paolo            |
| Altagar,G M     | Raanan           |
| Andersen        | Mel              |
| Anton           | Carmen           |
| Ashworth        | Rachel           |
| Barajas         | Miguel           |
...
```

If you want to sort customers by the last name in descending order, you use the `DESC` after the `contactLastname` column in the `ORDER BY` clause as shown in the following query:

```
SELECT
        contactLastname,
        contactFirstname
FROM
        customers
ORDER BY
        contactLastname DESC;
```

Try It Out ›

Ouptut:

```
+-----------------+------------------+
| contactLastname | contactFirstname |
+-----------------+------------------+
| Young           | Jeff             |
| Young           | Julie            |
| Young           | Mary             |
| Young           | Dorothy          |
| Yoshido         | Juri             |
| Walker          | Brydey           |
| Victorino       | Wendy            |
| Urs             | Braun            |
| Tseng           | Jerry            |
....
```

B) Using MySQL ORDER BY clause to sort the result set by multiple columns example

If you want to sort the customers by the last name in descending order and then by the first name in ascending order, you specify both `DESC` and `ASC` in these respective columns as follows:

```
SELECT
    contactLastname,
    contactFirstname
FROM
    customers
```

```
ORDER BY
        contactLastname DESC ,
        contactFirstname ASC;
```

Try It Out ›

Output:

```
+-----------------+------------------+
| contactLastname | contactFirstname |
+-----------------+------------------+
| Young           | Dorothy          |
| Young           | Jeff             |
| Young           | Julie            |
| Young           | Mary             |
| Yoshido         | Juri             |
| Walker          | Brydey           |
| Victorino       | Wendy            |
| Urs             | Braun            |
| Tseng           | Jerry            |
| Tonini          | Daniel           |
...
```

In this example, the `ORDER BY` clause sorts the result set by the last name in descending order first and then sorts the sorted result set by the first name in ascending order to make the final result set.

C) Using MySQL ORDER BY clause to sort a result set by an expression example

See the following `orderdetails` table from the sample database (https://www.mysqltutorial.org/mysql-sample-database.aspx) .

The following query selects the order line items from the `orderdetails` table. It calculates the subtotal for each line item and sorts the result set based on the subtotal.

```
SELECT
    orderNumber,
    orderlinenumber,
    quantityOrdered * priceEach
FROM
    orderdetails
ORDER BY
   quantityOrdered * priceEach DESC;
```

Try It Out ›

```
+-------------+-----------------+------------------------------+
| orderNumber | orderlinenumber | quantityOrdered * priceEach |
+-------------+-----------------+------------------------------+
|       10403 |               9 |                     11503.14 |
|       10405 |               5 |                     11170.52 |
|       10407 |               2 |                     10723.60 |
|       10404 |               3 |                     10460.16 |
|       10312 |               3 |                     10286.40 |
...
```

To make the query more readable, you can assign the expression in the `SELECT` clause a column alias (https://www.mysqltutorial.org/mysql-alias/) and use that column alias in the `ORDER BY` clause as shown in the following query:

```
SELECT
    orderNumber,
    orderLineNumber,
    quantityOrdered * priceEach AS subtotal
FROM
    orderdetails
ORDER BY subtotal DESC;
```

Try It Out ›

```
+-------------+-----------------+----------+
| orderNumber | orderLineNumber | subtotal |
+-------------+-----------------+----------+
|       10403 |               9 | 11503.14 |
|       10405 |               5 | 11170.52 |
```

```
|       10407 |                2 | 10723.60 |
|       10404 |                3 | 10460.16 |
|       10312 |                3 | 10286.40 |
|       10424 |                6 | 10072.00 |
|       10348 |                8 |  9974.40 |
|       10405 |                3 |  9712.04 |
|       10196 |                5 |  9571.08 |
|       10206 |                6 |  9568.73 |
 ...
```

In this example, we use `subtotal` as the column alias (https://www.mysqltutorial.org/mysql-alias/) for the expression `quantityOrdered * priceEach` and sort the result set by the `subtotal` alias.

Since MySQL evaluates the `SELECT` clause before the `ORDER BY` clause, you can use the column alias specified in the `SELECT` clause in the `ORDER BY` clause.

Using MySQL ORDER BY clause to sort data using a custom list

The `FIELD()` function has the following syntax:

```
FIELD(str, str1, str2, ...)
```

The `FIELD()` function returns the position of the str in the str1, str2, ... list. If the str is not in the list, the `FIELD()` function returns 0. For example, the following query returns 1 because the position of the string 'A' is the first position on the list `'A'`, `'B'`, and `'C'`:

```
SELECT FIELD('A', 'A', 'B','C');
```

Output:

```
+--------------------------+
| FIELD('A', 'A', 'B','C') |
+--------------------------+
|                        1 |
+--------------------------+
1 row in set (0.00 sec)
```

And the following example returns 2:

```
SELECT FIELD('B', 'A','B','C');
```

Output:

```
+-------------------------+
| FIELD('B', 'A','B','C') |
+-------------------------+
|                       2 |
+-------------------------+
1 row in set (0.00 sec)
```

Let's take a more practical example.

See the following `orders` table from the sample database.

Suppose that you want to sort the sales orders based on their statuses in the following order:

- In Process
- On Hold
- Canceled
- Resolved
- Disputed
- Shipped

To do this, you can use the `FIELD()` function to map each order status to a number and sort the result by the result of the `FIELD()` function:

```
SELECT
    orderNumber, status
FROM
    orders
ORDER BY FIELD(status,
```

```
        'In Process',
        'On Hold',
        'Cancelled',
        'Resolved',
        'Disputed',
        'Shipped');
```

Try It Out ›

```
+-------------+------------+
| orderNumber | status     |
+-------------+------------+
|       10425 | In Process |
|       10421 | In Process |
|       10422 | In Process |
|       10420 | In Process |
|       10424 | In Process |
|       10423 | In Process |
|       10414 | On Hold    |
|       10401 | On Hold    |
|       10334 | On Hold    |
|       10407 | On Hold    |
...
```

MySQL ORDER BY and NULL

In MySQL, `NULL` comes before non-NULL values. Therefore, when you the `ORDER BY` clause with the `ASC` option, `NULLs` appear first in the result set.

For example, the following query uses the `ORDER BY` clause to sort employees by values in the `reportsTo` column:

```
SELECT
    firstName, lastName, reportsTo
FROM
    employees
ORDER BY reportsTo;
```

Output:

```
+-----------+-----------+-----------+
| firstName | lastName  | reportsTo |
+-----------+-----------+-----------+
| Diane     | Murphy    |      NULL |
| Mary      | Patterson |      1002 |
| Jeff      | Firrelli  |      1002 |
| William   | Patterson |      1056 |
| Gerard    | Bondur    |      1056 |
...
```

However, if you use the `ORDER BY` with the `DESC` option, `NULLs` will appear last in the result set. For example:

```
SELECT
    firstName, lastName, reportsTo
FROM
    employees
ORDER BY reportsTo DESC;
```

Output:

```
+-----------+-----------+-----------+
| firstName | lastName  | reportsTo |
+-----------+-----------+-----------+
| Yoshimi   | Kato      |      1621 |
| Leslie    | Jennings  |      1143 |
| Leslie    | Thompson  |      1143 |
| Julie     | Firrelli  |      1143 |
| ....
| Mami      | Nishi     |      1056 |
| Mary      | Patterson |      1002 |
| Jeff      | Firrelli  |      1002 |
| Diane     | Murphy    |      NULL |
+-----------+-----------+-----------+
23 rows in set (0.00 sec)
```

Summary

- Use the `ORDER BY` clause to sort the result set by one or more columns.
- Use the `ASC` option to sort the result set in ascending order and the `DESC` option to sort the result set in descending order.

- The `ORDER BY` clause is evaluated after the `FROM` and `SELECT` clauses.
- In MySQL, `NULL` is lower than non-NULL values