SQL for Citizen Data Scientists

SQL ORDER BY Clause - Learn By Example

SETScholars & WACAMLDS

In general, rows in the result set returned from a query are not in any particular order. If you want them in a particular order, you need to instruct the DBMS to sort the result using the ORDER BY clause.

The ORDER BY clause allows you to sort one or more columns in ascending or descending order.

Syntax

```
SELECT column_name(s)
FROM table_name
ORDER BY column ASC|DESC;
```

Sample Table

To help you better understand the examples, and enable you to follow along with the tutorial, we are going to use the following sample table.

This table is part of an 'Employee Management System' that contains basic information about employees.

ID	FirstName	LastName	Age	Job	Salary	HireDate
5	Max	Williams	26	Janitor	9000	2015-01-15
1	Bob	Smith	28	Manager	60000	2011-03-07
3	Eve	Jones	22	Developer	32000	2013-03-11
4	Joe	Smith	24	Developer	30000	2016-10-05
6	Sam	Jones	30	Janitor	*NULL*	2012-02-10
2	Kim	Johnson	26	Manager	55000	2014-04-25

Sorting Data

If you looked at the sample table, you would have discovered that the data is displayed in no order of any significance. Actually the retrieved data is not displayed in a mere random order. It is usually displayed in the order in which it was initially added to the tables. However, even if you enter data in an ordered way it can be affected if it is later updated or deleted. After all you should not rely on the default order if you don't explicitly control it.

So to explicitly sort the retrieved data, you can use the ORDER BY clause in your SELECT statement.

ORDER BY takes the name of the column by which to sort the output. For example, to sort the list of employees by 'Age' in ascending order, you could run this query:

```
SELECT *
FROM Employees
ORDER BY Age ASC;
```

ID	FirstName	LastName	Age	Job	Salary	HireDate
3	Eve	Jones	22	Developer	32000	2013-03-11
4	Joe	Smith	24	Developer	30000	2016-10-05
5	Max	Williams	26	Janitor	9000	2015-01-15
2	Kim	Johnson	26	Manager	55000	2014-04-25
1	Bob	Smith	28	Manager	60000	2011-03-07
6	Sam	Jones	30	Janitor	*NULL*	2012-02-10

Specifying ASC (or ASCENDING) allows data to be sorted in ascending order. However, in practice, ASC is not usually used because the ascending order is the default order.

You can use ORDER BY with data types other than numbers, such as text and dates. The following example sorts the data alphabetically by 'FirstName'.

```
SELECT *
FROM Employees
ORDER BY FirstName;
```

ID	FirstName	LastName	Age	Job	Salary	HireDate
1	Bob	Smith	28	Manager	60000	2011-03-07
3	Eve	Jones	22	Developer	32000	2013-03-11
4	Joe	Smith	24	Developer	30000	2016-10-05
2	Kim	Johnson	26	Manager	55000	2014-04-25
5	Max	Williams	26	Janitor	9000	2015-01-15
6	Sam	Jones	30	Janitor	*NULL*	2012-02-10

The examples below sort the data by the date the employees were hired.

```
SELECT *
FROM Employees
ORDER BY HireDate;
```

ID	FirstName	LastName	Age	Job	Salary	HireDate
1	Bob	Smith	28	Manager	60000	2011-03-07
6	Sam	Jones	30	Janitor	*NULL*	2012-02-10
3	Eve	Jones	22	Developer	32000	2013-03-11
2	Kim	Johnson	26	Manager	55000	2014-04-25
5	Max	Williams	26	Janitor	9000	2015-01-15
4	Joe	Smith	24	Developer	30000	2016-10-05

When specifying an ORDER BY clause, be sure that it is the last clause in your SELECT statement. Otherwise, an error will be generated.

```
--raises syntax error
SELECT *
FROM Employees
ORDER BY Age ASC
WHERE Age > 25;
```

Sorting by Nonselected Columns

Columns used in the ORDER BY clause are usually selected for display, but this is not really necessary. It is perfectly legal to sort data by a column that does not appear in the result.

For example, following query sorts the result-set by the 'Age' column even if it is not included in the SELECT list.

```
SELECT FirstName, Job, Salary
FROM Employees
ORDER BY Age;
```

FirstName	Job	Salary
Eve	Developer	32000
Joe	Developer	30000
Max	Janitor	9000
Kim	Manager	55000
Bob	Manager	60000
Sam	Janitor	*NULL*

Sorting Descending

Data sorting is not limited to ascending sort order. The ORDER BY clause can also be used to sort data in descending order.

To sort by descending order, the keyword `DESC` must be specified after the ORDER BY clause. Descending sorts are commonly used for ranking queries, for example, the following query sorts all employees by 'Salary' with the highest paid employee at the top:

```
SELECT *
FROM Employees
ORDER BY Salary DESC;
```

ID	FirstName	LastName	Age	Job	Salary	HireDate
1	Bob	Smith	28	Manager	60000	2011-03-07
2	Kim	Johnson	26	Manager	55000	2014-04-25
3	Eve	Jones	22	Developer	32000	2013-03-11
4	Joe	Smith	24	Developer	30000	2016-10-05
5	Max	Williams	26	Janitor	9000	2015-01-15
6	Sam	Jones	30	Janitor	*NULL*	2012-02-10

Sorting Columns with NULL Values

Most DBMS orders NULL values at the beginning of the result set if the order is ascending and orders them at the end if the order is descending.

In this example, as you can see that after sorting all the employees according to the salary, the record of the employee whose salary is NULL came out on top.

```
SELECT *
FROM Employees
ORDER BY Salary;
```

ID	FirstName	LastName	Age	Job	Salary	HireDate
6	Sam	Jones	30	Janitor	*NULL*	2012-02-10
5	Max	Williams	26	Janitor	9000	2015-01-15
4	Joe	Smith	24	Developer	30000	2016-10-05
3	Eve	Jones	22	Developer	32000	2013-03-11
2	Kim	Johnson	26	Manager	55000	2014-04-25
1	Bob	Smith	28	Manager	60000	2011-03-07

Sorting via Expressions

You usually sort the result-set using column data, but sometimes you may need to sort by something that is not stored in the database, and possibly does not appear in your query. To handle such situations, you can add an expression to your ORDER BY clause.

For example, the following query uses the built-in function `DAY()` to extract the day from the `HireDate` column and then sorts the rows accordingly.

```
SELECT *
FROM Employees
ORDER BY DAY(HireDate);
```

ID	FirstName	LastName	Age	Job	Salary	HireDate
4	Joe	Smith	24	Developer	30000	2016-10-05
1	Bob	Smith	28	Manager	60000	2011-03-07
6	Sam	Jones	30	Janitor	*NULL*	2012-02-10
3	Eve	Jones	22	Developer	32000	2013-03-11
5	Max	Williams	26	Janitor	9000	2015-01-15
2	Kim	Johnson	26	Manager	55000	2014-04-25

Sorting by Column Position

Along with being able to specify sort order using column names, ORDER BY also supports sorting columns by their position (in the SELECT clause) rather than by name.

For example, if you want to sort using the second column ('FirstName' in this case) returned by a query, you could do the following:

```
SELECT ID, FirstName, Job
FROM Employees
ORDER BY 2;
```

ID	FirstName	Job
1	Bob	Manager
3	Eve	Developer
4	Joe	Developer
2	Kim	Manager
5	Max	Janitor
6	Sam	Janitor

The main advantage of this technique is that it avoids rewriting column names. But there are some downsides as well.

- Not explicitly listing column names increases the likelihood of incorrectly specifying the wrong column.
- After making changes to the SELECT list, it is very easy to forget to make changes to the ORDER BY clause.
- This technique obviously cannot be used when sorting columns that do not appear in the SELECT list.

Sorting by Multiple Columns

There is often a need to sort data by more than one column. For example, while displaying an employee list, you might want to sort it first by last-name, and then within each last-name sort by first-name. This will be useful when many employees are with the same last name.

To sort by multiple columns, simply specify the column names separated by commas in the ORDER BY clause.

```
SELECT LastName, FirstName, Job, Age
FROM Employees
ORDER BY LastName, FirstName;
```

LastName	FirstName	Job	Age
Johnson	Kim	Manager	26
Jones	Eve	Developer	22
Jones	Sam	Janitor	30
Smith	Bob	Manager	28
Smith	Joe	Developer	24
Williams	Max	Janitor	26

Remember! The order in which the columns are specified determines the sort sequence.

Sorting Descending on Multiple Columns

If you want to sort by multiple columns in descending order, make sure that each column has its own DESC keyword. If you don't add the DESC keyword for any column, that column will be sorted in ascending order by default.

Here is the same previous query that has been modified to sort in descending order.

```
SELECT LastName, FirstName, Job, Age
FROM Employees
ORDER BY LastName DESC, FirstName DESC;
```

LastName	FirstName	Job	Age
Williams	Max	Janitor	26
Smith	Joe	Developer	24
Smith	Bob	Manager	28
Jones	Sam	Janitor	30
Jones	Eve	Developer	22
Johnson	Kim	Manager	26