SQL Learning Material

Introduction to the SQL Server `ORDER BY` clause

When you use the `SELECT` statement to query data from a table, the order of rows in the result set is not guaranteed. It means that SQL Server can return a result set with an unspecified order of rows.

The only way for you to guarantee that the rows in the result set are sorted is to use the `ORDER BY` clause. The following illustrates the `ORDER BY` clause syntax:

```
SELECT
    select_list
FROM
    table_name
ORDER BY
    column_name | expression [ASC | DESC ];
```
Code language: SQL (Structured Query Language) (sql)

In this syntax:

column_name | expression

First, you specify a column name or an expression on which to sort the result set of the query. If you specify multiple columns, the result set is sorted by the first column and then that sorted result set is sorted by the second column, and so on.

The columns that appear in the `ORDER BY` clause must correspond to either column in the select list or columns defined in the table specified in the `FROM` clause.

ASC | DESC

Second, use `ASC` or `DESC` to specify whether the values in the specified column should be sorted in ascending or descending order.

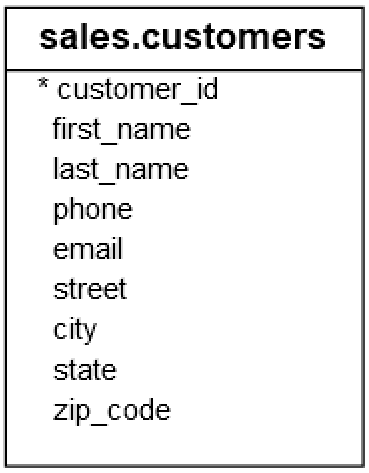
The `ASC` sorts the result from the lowest value to the highest value while the `DESC` sorts the result set from the highest value to the lowest one.

If you don't explicitly specify `ASC` or `DESC`, SQL Server uses `ASC` as the default sort order. Also, SQL Server treats NULL as the lowest value.

When processing the `SELECT` statement that has an `ORDER BY` clause, the `ORDER BY` clause is the very last clause to be processed.

SQL Server `ORDER BY` clause example

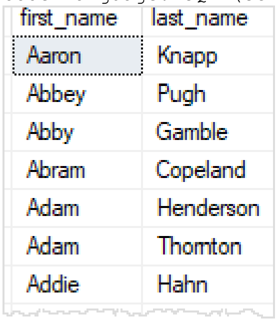
We will use the `customers` table in the sample database from the demonstration.



A) Sort a result set by one column in ascending order

The following statement sorts the customer list by the first name in ascending order:

```
SELECT
    first_name,
    last_name
FROM
    sales.customers
ORDER BY
    first_name;
Code language: SQL (Structured Query Language) (sql)
```



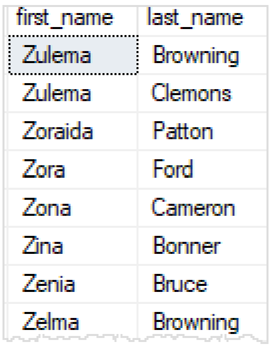
first_name	last_name
Aaron	Knapp
Abbey	Pugh
Abby	Gamble
Abram	Copeland
Adam	Henderson
Adam	Thornton
Addie	Hahn

In this example, because we did not specify `ASC` or `DESC`, the `ORDER BY` clause used `ASC` by default.

B) Sort a result set by one column in descending order

The following statement sorts the customer list by the first name in descending order.

```
SELECT
        firstname,
        lastname
FROM
        sales.customers
ORDER BY
        first_name DESC;
Code language: SQL (Structured Query Language) (sql)
```



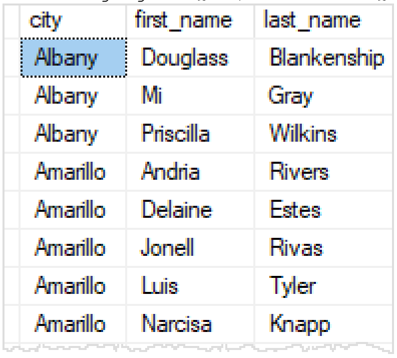
first_name	last_name
Zulema	Browning
Zulema	Clemons
Zoraida	Patton
Zora	Ford
Zona	Cameron
Zina	Bonner
Zenia	Bruce
Zelma	Browning

In this example, because we specified the `DESC` explicitly, the `ORDER BY` clause sorted the result set by values in the `first_name` column in descending order.

C) Sort a result set by multiple columns

The following statement retrieves the first name, last name, and city of the customers. It sorts the customer list by the city first and then by the first name.

```
SELECT
    city,
    first_name,
    last_name
FROM
    sales.customers
ORDER BY
    city,
    first_name;
```
Code language: SQL (Structured Query Language) (sql)

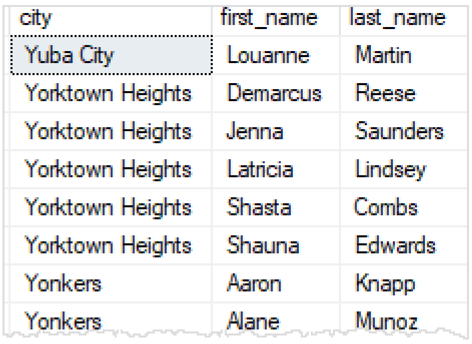


city	first_name	last_name
Albany	Douglass	Blankenship
Albany	Mi	Gray
Albany	Priscilla	Wilkins
Amarillo	Andria	Rivers
Amarillo	Delaine	Estes
Amarillo	Jonell	Rivas
Amarillo	Luis	Tyler
Amarillo	Narcisa	Knapp

D) Sort a result set by multiple columns and different orders

The following statement sorts the customers by the city in descending order and then sorts the sorted result set by the first name in ascending order.

```
SELECT
    city,
    first_name,
    last_name
FROM
    sales.customers
ORDER BY
    city DESC,
    first_name ASC;
```
Code language: SQL (Structured Query Language) (sql)

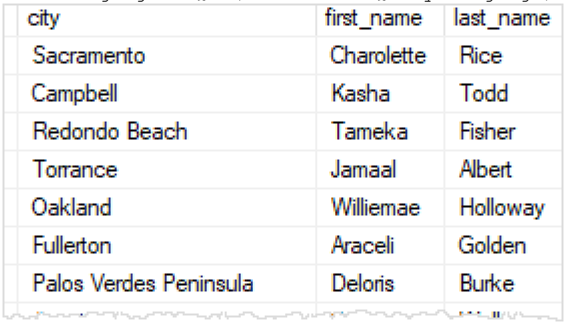


city	first_name	last_name
Yuba City	Louanne	Martin
Yorktown Heights	Demarcus	Reese
Yorktown Heights	Jenna	Saunders
Yorktown Heights	Latricia	Lindsey
Yorktown Heights	Shasta	Combs
Yorktown Heights	Shauna	Edwards
Yonkers	Aaron	Knapp
Yonkers	Alane	Munoz

E) Sort a result set by a column that is not in the select list

It is possible to sort the result set by a column that does not appear on the select list. For example, the following statement sorts the customer by the state even though the `state` column does not appear on the select list.

```
SELECT
    city,
    first_name,
    last_name
FROM
    sales.customers
ORDER BY
    state;
```
Code language: SQL (Structured Query Language) (sql)



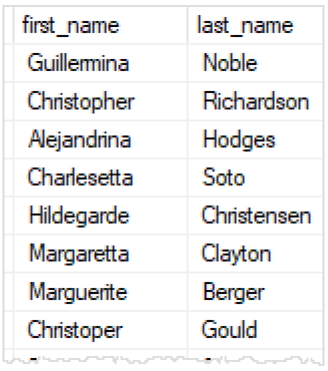
city	first_name	last_name
Sacramento	Charolette	Rice
Campbell	Kasha	Todd
Redondo Beach	Tameka	Fisher
Torrance	Jamaal	Albert
Oakland	Williemae	Holloway
Fullerton	Araceli	Golden
Palos Verdes Peninsula	Deloris	Burke

Note that the `state` column is defined in the `customers` table. If it was not, then you would have an invalid query.

F) Sort a result set by an expression

The `LEN()` function returns the number of characters in a string. The following statement uses the `LEN()` function in the `ORDER BY` clause to retrieve a customer list sorted by the length of the first name:

```
SELECT
    first_name,
    last_name
FROM
    sales.customers
ORDER BY
    LEN(first_name) DESC;
```
Code language: SQL (Structured Query Language) (sql)



first_name	last_name
Guillermina	Noble
Christopher	Richardson
Alejandrina	Hodges
Charlesetta	Soto
Hildegarde	Christensen
Margaretta	Clayton
Marguerite	Berger
Christoper	Gould

G) Sort by ordinal positions of columns

SQL Server allows you to sort the result set based on the ordinal positions of columns that appear in the select list.

The following statement sorts the customers by first name and last name. But instead of specifying the column names explicitly, it uses the ordinal positions of the columns:

```
SELECT
    first_name,
    last_name
FROM
    sales.customers
ORDER BY
    1,
    2;
Code language: SQL (Structured Query Language) (sql)
```

In this example, 1 means the `first_name` column, and 2 means the `last_name` column.

Using the ordinal positions of columns in the `ORDER BY` clause is considered a bad programming practice for a couple of reasons.

- First, the columns in a table don't have ordinal positions and need to be referenced by name.
- Second, when you modify the select list, you may forget to make the corresponding changes in the `ORDER BY` clause.

Therefore, it is a good practice to always specify the column names explicitly in the `ORDER BY` clause.

In this tutorial, you have learned how to use the SQL Server `ORDER BY` clause to sort a result set by columns in ascending or descending order.