Sorting data

This lesson demonstrates how to use ORDER BY in SQL to retrieve sorted data. You'll see how to sort data alphabetically as well as in ascending or descending order.

When you're looking at a long list of information, it can be hard to find what you need if the list is in random order. It's much easier and faster to find things if the list has some order to it. If you're looking at your monthly expenditures and want to see where you spent the most money, you probably want to see your expenses ordered from largest to smallest. Similarly, if you're trying to find certain items in a lengthy list of items, it may be most helpful to order the list alphabetically. In SQL, you can use the ORDER BY clause to sort data.

ORDER BY

The ORDER BY clause is used to order data that's retrieved by a query. An ORDER BY clause follows this structure:

```sql
SELECT column1, column2, …
FROM table_name
ORDER BY column_name;
```

Using ORDER BY is simple and straightforward. If you're ordering numeric data, the default order of ORDER BY is in ascending order, or smallest to largest. If you're ordering string data, the default order of ORDER BY is in alphabetical order, from A to Z.

But what if you want to view numeric data in descending order, from largest to smallest? Or what if you want to view string data in alphabetical order, but from Z to A? The addition of the keyword DESC (short for descending) does this for you. The structure of the ORDER BY query using DESC looks like this:

```sql
SELECT column1, column2, …
FROM table_name
ORDER BY column_name DESC;
```

Similarly, there is an ASC keyword, which sorts the data in ascending order—smallest to largest for numeric data, and A to Z for string data.

```sql
SELECT column1, column2, …
FROM table_name
ORDER BY column_name ASC;
```

Using ASC sorts data the same way as when you don't specify any order in your ORDER BY clause. But it's a good idea to specify ASC when you want to sort in ascending order; this helps to avoid any confusion, and it's especially helpful if you're doing two different sorts in one query.

Sorting numeric data

With some practice, it's easy to get the hang of ORDER BY. So, in the tree control pane, find the database called employees. Right-click this database and select Query Tool. In the Query Editor, write and execute the following query to pull all employee records, with their salaries shown in ascending order.

```sql
SELECT *
FROM emp
ORDER BY sal ASC;
```

Scroll through the Data Output. You'll see the salary field, called sal, sorted from smallest to largest.

Now, create a query that pulls all records while sorting the sal field from largest to smallest. Your query should look like this:

```sql
SELECT *
FROM emp
ORDER BY sal DESC;
```

Execute this query and look at your output. You will see the salary field sorted from largest to smallest.

Sorting string data

Now practice using ORDER BY on string data. Create and execute a query that pulls all data from the emp table, with the employee names (ename) in alphabetical order from A to Z. Your query should look like this:

```sql
SELECT *
FROM emp
ORDER BY ename ASC;
```

Scroll through your output. You will see all employee names ordered from A to Z.

Now do the same thing, but sort the employee names from Z to A. Your query should look like this:

```sql
SELECT *
FROM emp
ORDER BY ename DESC;
```

Now you should see the employee names ordered from Z to A, with the first record showing employee Ward.

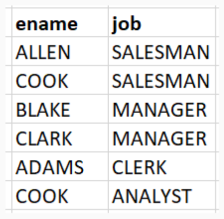
To use ORDER BY on string data, the field must be a character varying (varchar) data type. ORDER BY doesn't work on text data types. So, if you ever want to ORDER BY on a text data type field, convert the data type from text to varchar. This is shown in an upcoming lesson.

Using two sorts in one query

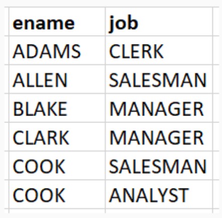
Finally, practice using two sorts in one query. Try creating a query that sorts ename from A to Z, and then sorts job from Z to A. Then compare your query to the one below. As you can see, to add a second sort, you just need to add a comma followed by the second sort.

```sql
SELECT *
FROM emp
ORDER BY ename ASC, job DESC;
```

If you look at the output for this query, it may look like you only sorted by employee name. Adding the second sort would only make a difference here if there were two employees with the same name. Then the two employees would be ordered so that their job titles are in descending order. The first image below shows the situation where two employees have the same name COOK, but different job titles. The second image shows how these are sorted by ename alphabetically, and then by job in descending order.



ename	job
ALLEN	SALESMAN
COOK	SALESMAN
BLAKE	MANAGER
CLARK	MANAGER
ADAMS	CLERK
COOK	ANALYST



ename	job
ADAMS	CLERK
ALLEN	SALESMAN
BLAKE	MANAGER
CLARK	MANAGER
COOK	SALESMAN
COOK	ANALYST