Types of ordering / sorting

In this reading, you'll explore using the ORDER BY clause for sorting data. You've learned about the purpose of the ORDER BY clause and the different forms in which it can be used to sort data. The main objective of this reading is to present some more examples related to some practical scenarios of using the ORDER BY clause for sorting data in a table.

The ORDER BY clause

The ORDER BY clause is useful when you want to sort or order the results obtained when running a SQL SELECT query. Data in a database become more meaningful when they are ordered or sorted in a specific order. Ordered data helps people make more accurate business decisions effectively and efficiently.

In SQL, there's the ORDER BY clause that can help you achieve this. If you run a SQL SELECT query, you get a set of unsorted results. If you want to sort them, you need to add the special ORDER BY clause into the SQL SELECT statement.

It can be used after the FROM clause as in:

```
SELECT *
FROM Employee
ORDER BY <order by column name>;
```

After the ORDER BY keyword, you need to specify the column name based on the data that needs to be sorted. Optionally, you can specify the keywords ASC or DESC after the column name. This is to indicate if the ordering should be in ascending or descending order.

Ascending and descending order are the two main types of ordering. If ASC or DESC are not specified, the data is sorted by default in ascending order. The ASC and DESC keywords sort the data based on the order by column, taking into consideration the data type of column or field, namely integer, numeric, text and dates.

Working with the ORDER BY clause

Let's review some example scenarios that use the ORDER BY clause using the tables in the sample database. You can give SQL SELECT statements using the ORDER BY clause with ASC and DESC keywords as required for these scenarios.

Sorting by a single column

In the customer table, the data is sorted by default in ascending order within the customer ID field. The customer ID field is numeric, so the data is sorted in ascending numeric order. Now let's examine how to order this data in the descending order of the Customer ID field.

To do this, you can run the following query:

```
SELECT *
FROM customers
ORDER BY CustomerId DESC;
```

Run

Reset

In the output, the records are sorted in descending order (largest to smallest) of the CustomerIdwhich has a numeric data type.

Now let's examine how sorting happens for a text data typed column. Consider sorting the data by the City column which has a text data type of VARCHAR. If you want to sort the customer data by city, use the following SELECT statement:

```
SELECT *
FROM customers
ORDER BY City;
```

Run

Reset

An ordering method like ASC or DESC wasn't specified in the ORDER BY clause. So, by default, the ordering happens in ascending order.
If you review the City column, you'll notice that the data is sorted in ascending alphabetical order (A to Z).

Let's now run the following SELECT statement:

```sql
SELECT *
FROM customers
ORDER BY City DESC;
```

Run

Reset

Now you'll notice that the records are ordered in descending alphabetical order (Z to A).

Let's examine another example of how data is ordered in a sort column that uses a DATE type field. This example uses the invoices table in the sample database. You can use the following SQL SELECT statement to sort the data by the invoice date column:

```sql
SELECT *
FROM invoices
ORDER BY InvoiceDate;
```

Run

Reset

If you review the InvoiceDate column, you'll notice that the date values are sorted from smallest to largest. That is, they're sorted in ascending order, which is the default sort order.Now let's try running this query with the DESC keyword added in the ORDER BY clause.

```sql
SELECT *
FROM invoices
ORDER BY InvoiceDate DESC;
```

Run

Reset

The data is now sorted from the largest to smallest date, which is descending order.

Ordering by multiple columns
You can also sort data by multiple columns and apply different sort orders to them. Let's say you want to sort invoice data by both billing city and invoice date. To do this, run the following query:

```sql
SELECT *
FROM invoices
ORDER BY BillingCity ASC, InvoiceDate DESC;
```

Run

Reset

You'll notice that the data is sorted in ascending order of BillingCity. That's why the data in the BillingCity column is sorted in alphabetical order. The data of the InvoiceDatecolumn is in turn sorted in descending order.
For example, if you review the records with the billing city of Amsterdam, the invoice dates are ordered in descending order from largest to smallest date. Similarly, if you examine the other sets of data closely, you'll observe the same.
The main types of ordering in SQL are ASC, ascending, and DESC, descending. How the data is ordered in these two cases would depend on the data type of the field or column being used as the sort column.