Ben Forta

FIFTH EDITION

FULL COLOR

Now with challenge questions to gauge and improve proficiency.

Sams Teach Yourself SQL in 10 Minutes

Ben Forta

Sams Teach Yourself SQL in 10 Minutes

Fifth Edition

SAMS

221 River Street, Hoboken, NJ 07030

Output ▼

```
vend_title
-----------------------------------------------------------
Bear Emporium (USA)
Bears R Us (USA)
Doll House Inc. (USA)
Fun and Games (England)
Furball Inc. (USA)
Jouets et ours (France)
```

The following is the same statement, but using the || syntax:

Input ▼

```
SELECT RTRIM(vend_name) || ' (' || RTRIM(vend_country) || ')'
 AS vend_title
FROM Vendors
ORDER BY vend_name;
```

And here is the equivalent for use with MySQL and MariaDB:

Input ▼

```
SELECT Concat(RTrim(vend_name), ' (',
       RTrim(vend_country), ')') AS vend_title
FROM Vendors
ORDER BY vend_name;
```

Analysis ▼

The `SELECT` statement itself is the same as the one used in the previous code snippet, except that here the calculated field is followed by the text `AS vend_title`. This instructs SQL to create a calculated field named `vend_title` containing the calculation specified. As you can see in the output, the results are the same as before, but the column is now named `vend_title`, and any client application can refer to this column by name, just as it would to any actual table column.

> NOTE: **`AS` Often Optional**
>
> Use of the `AS` keyword is optional in many DBMSs, but using it is considered a best practice.

TIP: **Other Uses for Aliases**

Aliases have other uses too. Some common uses include renaming a column if the real table column name contains illegal characters (for example, spaces) and expanding column names if the original names are either ambiguous or easily misread.

CAUTION: **Alias Names**

Aliases may be single words or complete strings. If the latter is used, then the string should be enclosed within quotes. This practice is legal but is strongly discouraged. While multiword names are indeed highly readable, they create all sorts of problems for many client applications—so much so that one of the most common uses of aliases is to rename multiword column names to single-word names (as explained above).

NOTE: **Derived Columns**

Aliases are also sometimes referred to as `derived columns`, so regardless of the term you run across, they mean the same thing.

Performing Mathematical Calculations

Another frequent use for calculated fields is performing mathematical calculations on retrieved data. Let's take a look at an example. The `Orders` table contains all orders received, and the `OrderItems` table contains the individual items within each order. The following SQL statement retrieves all the items in order number `20008`:

Input ▼

```
SELECT prod_id, quantity, item_price
FROM OrderItems
WHERE order_num = 20008;
```

Output ▼

```
prod_id       quantity       item_price
----------    -----------    ---------------------
RGAN01        5              4.9900
BR03          5              11.9900
BNBG01        10             3.4900
BNBG02        10             3.4900
BNBG03        10             3.4900
```

The item_price column contains the per unit price for each item in an order. To expand the item price (item price multiplied by quantity ordered), you simply do the following:

Input ▼

```
SELECT prod_id,
       quantity,
       item_price,
       quantity*item_price AS expanded_price
FROM OrderItems
WHERE order_num = 20008;
```

Output ▼

```
prod_id      quantity       item_price      expanded_price
----------   -----------    ------------    ------------------
RGAN01       5              4.9900          24.9500
BR03         5              11.9900         59.9500
BNBG01       10             3.4900          34.9000
BNBG02       10             3.4900          34.9000
BNBG03       10             3.4900          34.9000
```

Analysis ▼

The expanded_price column shown in the output above is a calculated field; the calculation is simply quantity*item_price. The client application can now use this new calculated column just as it would any other column.

SQL supports the basic mathematical operators listed in Table 7.1. In addition, you can use parentheses to establish order of precedence. Refer to Lesson 5, "Advanced Data Filtering," for an explanation of precedence.

TABLE 7.1 SQL Mathematical Operators

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division

TIP: **How to Test Calculations**

`SELECT` provides a great way to test and experiment with functions and calculations. Although `SELECT` is usually used to retrieve data from a table, the `FROM` clause may be omitted to simply access and work with expressions. For example, `SELECT 3 * 2;` would return `6`, `SELECT Trim('    abc    ');` would return `abc`, and `SELECT Curdate();` uses the `Curdate()` function to return the current date and time (on MySQL and MariaDB, for example). You get the idea: use `SELECT` to experiment as needed.

Summary

In this lesson, you learned what calculated fields are and how to create them. You used examples demonstrating the use of calculated fields for both string concatenation and mathematical operations. In addition, you learned how to create and use aliases so that your application can refer to calculated fields.

Challenges

1. A common use for aliases is to rename table column fields in retrieved results (perhaps to match specific reporting or client needs). Write a SQL statement that retrieves `vend_id`, `vend_name`, `vend_address`, and `vend_city` from `Vendors`, renaming `vend_name` to `vname`, `vend_city` to `vcity`, and `vend_address` to `vaddress`. Sort the results by vendor name (you can use the original name or the renamed name).
2. Our example store is running a sale and all products are 10% off. Write a SQL statement that returns `prod_id`, `prod_price`, and `sale_price` from the `Products` table. `sale_price` is a calculated field that contains, well, the sale price. Here's a hint: you can multiply by `0.9` to get 90% of the original value (and thus the 10% off price).

LESSON 8

Using Data Manipulation Functions

In this lesson, you'll learn what functions are, what types of functions DBMSs support, and how to use these functions. You'll also learn why SQL function use can be very problematic.

Understanding Functions

Like almost any other computer language, SQL supports the use of functions to manipulate data. Functions are operations that are usually performed on data, usually to facilitate conversion and manipulation, and they are an important part of your SQL toolbox.

An example of a function is `RTRIM()`, which we used in the last lesson to trim spaces from the end of a string.

The Problem with Functions

Before you work through this lesson and try the examples, you should be aware that, unfortunately, using SQL functions can be highly problematic.

Unlike SQL statements (for example, `SELECT`), which for the most part are supported by all DBMSs equally, functions tend to be very DBMS specific. In fact, very few functions are supported identically by all major DBMSs. Although all types of functionality are usually available in each DBMS, the function names or syntax can differ greatly. To demonstrate just how problematic this can be, Table 8.1 lists three commonly needed functions and their syntax as employed by various DBMSs: