**Grouping Data**

So far you’ve seen how to manipulate several tables into one larger table, and to sort and filter the resulting data. The SQL **GROUP BY** command lets you roll up rows based on similar columns and to perform calculations.

**GROUP BY** expects a comma-separated list of columns. Rows that have the same values in those columns are rolled up and counted by [*aggregate functions*](gloss01.html#gloss_09)in the **SELECT** clause. This query provides a list of authors and the number of books in the database:

[**Click here to view code image**](ch13_images.html#p386pro02a)

sqlite> **SELECT first\_name, last\_name, COUNT(title) AS books**  
**FROM author**  
**LEFT JOIN book ON author\_id=author.id**  
**GROUP BY first\_name, last\_name;**  
first\_name  last\_name        books  
----------  ---------------  -----  
Ada         Lovelace         0  
Ross        Brunson          2  
Sean        Walberg          2

The elements from the previous queries in this chapter are still there. The query selects some columns from **author** and joins in **book** with a left join. The difference is the **GROUP BY** clause and the **COUNT(title)**.

The grouping is what rolls up all rows with similar authors into one. COUNT(title) is an aggregate function that asks the database for the number of non-NULL titles in each grouping.

There are more aggregate functions, depending on your database:

**AVG(column)—**Returns the average of the column (NULLs are removed.)

**COUNT(column)—**Counts the number of non-NULL instances of column, or total rows if **\*** is used instead of a column name

**MIN(column)—**Returns the minimum non-NULL value in the column

**MAX(column)—**Returns the maximum non-NULL value in the column