Chapter 3

How to retrieve data   
from a single table

Exercises

Enter and run your own SELECT statements

In these exercises, you’ll enter and run your own SELECT statements.

1. Write a SELECT statement that returns four columns from the Products table: product\_code, product\_name, list\_price, and discount\_percent. Then, run this statement to make sure it works correctly.

Add an ORDER BY clause to this statement that sorts the result set by list price in descending sequence. Then, run this statement again to make sure it works correctly. This is a good way to build and test a statement, one clause at a time.

1. Write a SELECT statement that returns one column from the Customers table named full\_name that joins the last\_name and first\_name columns.

Format this column with the last name, a comma, a space, and the first name like this:

Doe, John

Sort the result set by the last\_name column in ascending sequence.

Return only the customers whose last name begins with letters from M to Z.

NOTE: When comparing strings of characters, ‘M’ comes before any string of characters that begins with ‘M’. For example, ‘M’ comes before ‘Murach’.

1. Write a SELECT statement that returns these columns from the Products table:

product\_name The product\_name column

list\_price The list\_price column

date\_added The date\_added column

Return only the rows with a list price that’s greater than 500 and less than 2000.

Sort the result set by the date\_added column in descending sequence.

1. Write a SELECT statement that returns these column names and data from the Products table:

product\_name The product\_name column

list\_price The list\_price column

discount\_percent The discount\_percent column

discount\_amount A column that’s calculated from the previous two columns

discount\_price A column that’s calculated from the previous three columns

Round the discount\_amount and discount\_price columns to 2 decimal places.

Sort the result set by the discount\_price column in descending sequence.

Use the LIMIT clause so the result set contains only the first 5 rows.

1. Write a SELECT statement that returns these column names and data from the Order\_Items table:

item\_id The item\_id column

item\_price The item\_price column

discount\_amount The discount\_amount column

quantity The quantity column

price\_total A column that’s calculated by multiplying the item price by the quantity

discount\_total A column that’s calculated by multiplying the discount amount by the quantity

item\_total A column that’s calculated by subtracting the discount amount from the item price and then multiplying by the quantity

Only return rows where the item\_total is greater than 500.

Sort the result set by the item\_total column in descending sequence.

Work with nulls and test expressions

1. Write a SELECT statement that returns these columns from the Orders table:

order\_id The order\_id column

order\_date The order\_date column

ship\_date The ship\_date column

Return only the rows where the ship\_date column contains a null value.

Chapter 4

How to retrieve data   
from two or more tables

Exercises

1. Write a SELECT statement that joins the Categories table to the Products table and returns these columns: category\_name, product\_name, list\_price.

Sort the result set by the category\_name column and then by the product\_name column in ascending sequence.

1. Write a SELECT statement that joins the Customers table to the Addresses table and returns these columns: first\_name, last\_name, line1, city, state, zip\_code.

Return one row for each address for the customer with an email address of allan.sherwood@yahoo.com.

1. Write a SELECT statement that joins the Customers table to the Addresses table and returns these columns: first\_name, last\_name, line1, city, state, zip\_code.

Return one row for each customer, but only return addresses that are the shipping address for a customer.

1. Write a SELECT statement that joins the Customers, Orders, Order\_Items, and Products tables. This statement should return these columns: last\_name, first\_name, order\_date, product\_name, item\_price, discount\_amount, and quantity.

Use aliases for the tables.

Sort the final result set by the last\_name, order\_date, and product\_name columns.

1. Write a SELECT statement that returns the product\_name and list\_price columns from the Products table.

Return one row for each product that has the same list price as another product.   
*Hint: Use a self-join to check that the product\_id columns aren’t equal but the list\_price columns are equal.*

Sort the result set by the product\_name column.

1. Write a SELECT statement that returns these two columns:

category\_name The category\_name column from the Categories table

product\_id The product\_id column from the Products table

Return one row for each category that has never been used. *Hint: Use an outer join and only return rows where the product\_id column contains a null value.*

Chapter 5

How to insert, update, and delete data

Exercises

To test whether a table has been modified correctly as you do these exercises, you can write and run an appropriate SELECT statement.

1. Write an INSERT statement that adds this row to the Categories table:

category\_name: Brass

Code the INSERT statement so MySQL automatically generates the category\_id column.

1. Write an UPDATE statement that modifies the row you just added to the Categories table. This statement should change the product\_name column to “Woodwinds”, and it should use the category\_id column to identify the row.
2. Write a DELETE statement that deletes the row you added to the Categories table in exercise 1. This statement should use the category\_id column to identify the row.

Chapter 6

How to code summary queries

Exercises

1. Write a SELECT statement that returns these columns:

The count of the number of orders in the Orders table

The sum of the tax\_amount columns in the Orders table

1. Write a SELECT statement that returns one row for each category that has products with these columns:

The category\_name column from the Categories table

The count of the products in the Products table

The list price of the most expensive product in the Products table

Sort the result set so the category with the most products appears first.

1. Write a SELECT statement that returns one row for each customer that has orders with these columns:

The email\_address column from the Customers table

The sum of the item price in the Order\_Items table multiplied by the quantity in the Order\_Items table

The sum of the discount amount column in the Order\_Items table multiplied by the quantity in the Order\_Items table

Sort the result set in descending sequence by the item price total for each customer.

1. Write a SELECT statement that returns one row for each customer that has orders with these columns:

The email\_address column from the Customers table

A count of the number of orders

The total amount for each order (*Hint: First, subtract the discount amount from the price. Then, multiply by the quantity.*)

Return only those rows where the customer has more than 1 order.

Sort the result set in descending sequence by the sum of the line item amounts.

1. Modify the solution to exercise 4 so it only counts and totals line items that have an item\_price value that’s greater than 400.