How to code summary queries

Exercises

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.

Chapter 7

How to code subqueries

Exercises

1. Write a SELECT statement that returns the same result set as this SELECT statement, but don’t use a join. Instead, use a subquery in a WHERE clause that uses the IN keyword.

SELECT DISTINCT category\_name

FROM categories c JOIN products p

ON c.category\_id = p.category\_id

ORDER BY category\_name

1. Write a SELECT statement that answers this question: Which products have a list price that’s greater than the average list price for all products?

Return the product\_name and list\_price columns for each product.

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

1. Write a SELECT statement that returns three columns: email\_address, order\_id, and the order total for each customer. To do this, you can group the result set by the email\_address and order\_id columns. In addition, you must calculate the order total from the columns in the Order\_Items table.

Write a second SELECT statement that uses the first SELECT statement in its FROM clause. The main query should return two columns: the customer’s email address and the largest order for that customer. To do this, you can group the result set by the email\_address. Sort the result set by the largest order in descending sequence.

1. Write a SELECT statement that returns the name and discount percent of each product that has a unique discount percent. In other words, don’t include products that have the same discount percent as another product.

Sort the result set by the product\_name column.

Chapter 12

How to create views

Exercises

1. Create a view named customer\_addresses that shows the shipping and billing addresses for each customer.

This view should return these columns from the Customers table: customer\_id, email\_address, last\_name and first\_name.

This view should return these columns from the Addresses table: bill\_line1, bill\_line2, bill\_city, bill\_state, bill\_zip, ship\_line1, ship\_line2, ship\_city, ship\_state, and ship\_zip.

1. Write a SELECT statement that returns these columns from the customer\_addresses view that you created in exercise 1: customer\_id, last\_name, first\_name, bill\_line1. The rows in the result should be sorted by the last\_name and then first\_name columns.
2. Write an UPDATE statement that updates the Customers table using the customer\_addresses view you created in exercise 1. Set the first line of the shipping address to “1990 Westwood Blvd.” for the customer with an ID of 8.
3. Create a view named order\_item\_products that returns columns from the Orders, Order\_Items, and Products tables.

This view should return these columns from the Orders table: order\_id, order\_date, tax\_amount, and ship\_date.

This view should return the product\_name column from the Products table.

This view should return these columns from the Order\_Items table: item\_price, discount\_amount, final\_price (the discount amount subtracted from the item price), quantity, and item\_total (the calculated total for the item).

1. Create a view named product\_summary that uses the view you created in exercise 4. This view should return summary information about each product.

Each row should include product\_name, order\_count (the number of times the product has been ordered) and order\_total (the total sales for the product).

1. Write a SELECT statement that uses the view that you created in exercise 5 to get total sales for the five best selling products. Sort the result set by the order\_total column in descending sequence.