Klasseopgaver: Joins, summary queries & subqueries

Joins

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 category_name and then by product_name in ascending sequence.

2. 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.

3. 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.

Summary queries

4. 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

5. 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.

6. 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.

Subqueries

7. 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
```

8. 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 results by the list_price column in descending sequence.

9. Write a SELECT statement that returns the category_name column from the Categories table.

   Return one row for each category that has never been assigned to any product in the Products table. To do that, use a subquery introduced with the NOT EXISTS operator.