Hand-in assignment on joins, summery queries and subqueries

How to retrieve data from two or more tables

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 last_name, order_date, and product_name.

2. 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 product_name.

3. 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.*

4. Use the UNION operator to generate a result set consisting of three columns from the Orders table:

ship_status	A calculated column that contains a value of SHIPPED or NOT SHIPPED
order_id	The order_id column
order_date	The order_date column

   If the order has a value in the ship_date column, the ship_status column should contain a value of SHIPPED. Otherwise, it should contain a value of NOT SHIPPED.

   Sort the final result set by order_date.

Summery queries

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

   The email_address 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.

6. 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.

7. Write a SELECT statement that answers this question: What is the total amount ordered for each product? Return these columns:

   The product name from the Products table

   The total amount for each product in the Order_Items (*Hint: You can calculate the total amount by subtracting the discount amount from the item price and then multiplying it by the quantity*)

   Use the WITH ROLLUP operator to include a row that gives the grand total.

   Note: Once you add the WITH ROLLUP operator, you may need to use MySQL Workbench's Execute SQL Script button instead of its Execute Current Statement button to execute this statement.

8. Write a SELECT statement that answers this question: Which customers have ordered more than one product? Return these columns:

   The email address from the Customers table

   The count of distinct products from the customer's orders

Subqueries

9. 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.

10. 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 results by the product_name column.

11. Use a correlated subquery to return one row per customer, representing the customer's oldest order (the one with the earliest date). Each row should include these three columns: email_address, order_id, and order_date.