

CSI 300: Problem Set 3

For this problem set you'll be using the **Murach** provided database created from the script *"create_my_guitar_shop.sql"* that you can find on Canvas. Run that script before you attempt any of the exercises below.

1. Write a SELECT statement that joins the Categories table to the Products table and returns these columns:
 - a. category_name
 - b. product_name
 - c. 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, Orders, Order_Items, and Products tables. This statement should return these columns:
 - a. last_name
 - b. first_name
 - c. order_date
 - d. product_name
 - e. item_price
 - f. discount_amount
 - g. quantity

Use aliases of your choice for the tables. Sort the final result set by last_name, order_date, and product_name.

3. 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.

4. Write a SELECT statement that returns these two columns:
- a. category_name (The category_name column from the Categories table)
 - b. 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.

5. Write a SELECT statement that returns these columns:
- a. The count of the number of orders in the Orders table
 - b. The sum of the tax_amount columns in the Orders table
6. Write a SELECT statement that returns one row for each customer that has orders with these columns:
- a. The email_address from the Customers table
 - b. count of the number of orders
 - c. 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.

7. Write a SELECT statement that answers this question: Which customers have ordered more than one product? Return these columns:
- a. The email address from the Customers table
 - b. The count of distinct products from the customer's orders
8. 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
```

9. Write a SELECT statement that returns three columns:
- a. email_address
 - b. order_id
 - c. 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:

- a. the customer's email address
- b. the largest order for that customer.

To do this, you can group the result set by the email_address.

10. Use the UNION operator to generate a result set consisting of three columns from the Orders table:
- a. ship_status
 - b. order_id
 - c. order_date
 - d. A calculated column that contains a value of SHIPPED or NOT SHIPPED.

Sort the final result set by order_date.