**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:
   1. category\_name
   2. product\_name
   3. list\_price

Sort the result set by category\_name and then by product\_name in

ascending sequence.

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

Use aliases of your choice for the tables. Sort the final result set by

last\_name, order\_date, and product\_name.

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 product\_name.

1. Write a SELECT statement that returns these two columns:
   1. category\_name (The category\_name column from the Categories table)
   2. 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.

1. Write a SELECT statement that returns these columns:
   1. The count of the number of orders in the Orders table
   2. The sum of the tax\_amount columns in the Orders table
2. Write a SELECT statement that returns one row for each customer that has

orders with these columns:

1. The email\_address from the Customers table
2. count of the number of orders
3. 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. Write a SELECT statement that answers this question: Which customers have ordered more than one product? Return these columns:
   1. The email address from the Customers table
   2. The count of distinct products from the customer’s orders
2. 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 returns three columns:
   1. email\_address
   2. order\_id
   3. 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:

1. the customer’s email address
2. the largest order for that customer.

To do this, you can group the result set by the email\_address.

1. Use the UNION operator to generate a result set consisting of three

columns from the Orders table:

* 1. ship\_status
  2. order\_id
  3. order\_date
  4. A calculated column that contains a value of SHIPPED or NOT SHIPPED.

Sort the final result set by order\_date.