CIS4331 Spring 2019 Lab 8
Subquery

1. Objectives

This lab will help you to

- Learn how to write a query that contains a subquery
- Enhance your skills to write summary queries using aggregation functions

2. Tasks to Complete

Complete the questions about subqueries on the **MGS Database** included in the later part of this document. These queries use the tables in **user mgs**. This lab is **required to submit by 11:50pm, Tuesday, March 19**.

NOTE: the links to online Oracle SQL Language references are available in the Modules\Resources folder on Canvas.

NOTE: The scripts you need in this practice can be downloaded from the item "*Murach Book Example Code"* in the Modules\Resources folder on Canvas.

3. Submission Requirements

NOTES:

- **TO EARN ANY CREDIT, you MUST SUBMIT YOUR WORK TO CANVAS.**
- **YOU MUST WRITE your FULL NAME on the first page.**

Please place your SQL statements in a text file with the extension .sql. Mark each query based on the question number.

Then submit this SQL script file by attaching it to the link **Lab 8** in folder **Assignments\Labs** on Canvas.

NOTE:

Because this lab is for you to practice how to use subqueries, you MUST use subquery to answer each question. Otherwise, you will get a 0.

Queries Containing Subquery on MGS Database

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
   ```

2. Print the product name and actual price of products whose actual price is above the average actual price of all products. The actual price is the price after the discount. Sort the result such that the product with the highest actual price appears first.

3. Print the ids and names of all categories that currently don't have any product.
   You must use NOT IN and a subquery.

4. Print the ids, codes, product names, and actual prices of the top 3 products that are most expensive based on the actual price, which is the price after the discount.
   You MUST use a subquery in the FROM clause in your answer.

5. Print the ids, codes, product names, **category names**, and actual prices of the top 3 products that are most expensive based on the actual price, which is the price after the discount.
   You MUST use a subquery in the FROM clause in your answer.

6. Print the category ids, product counts of the top 3 categories with the most products.
   You MUST use a subquery in the FROM clause in your answer.

7. Print 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 in increasing sequence.
   HINT: In the subquery, count the number of products in each discount percent and return those with product count being 1.

8. Write a query that prints 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 query that uses the first query in its FROM clause. The main query should return 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.