Murach 3e Chapter 7 Coding Assignment Instructions

Exercises to be Completed
Please complete Exercises 1 – 6. You will find those exercises on the attached sheet(s).

General Instructions
My expectations for your work on coding assignment exercises will grow as we progress through the course. In addition to applying any new coding techniques that have been covered in the current chapter, I will be expecting you to follow all of the good practices that we have adopted in the preceding weeks. Here is a quick summary of good practices that we have covered so far:

- Begin each script file that accesses the database with a USE statement (e.g., USE my_guitar_shop;).
- Use the *beautify* feature of the MySQL Workbench to *pretty-print* your code.
- End each statement in your script with a semicolon.
- Use the SQL features requested in the exercise description and/or covered in the chapter.
- Always include an ORDER BY in SELECT statements unless directed otherwise. If the exercise instructions ask for a particular order, then use that. Otherwise, choose any reasonable order.
- In SELECT statements that use JOIN, always use the explicit (ANSI) JOIN syntax implemented in the FROM clause. Do NOT use the implicit JOIN syntax implemented using the WHERE clause.
- When testing SELECT statements that use summary functions, always test with the ONLY_FULL_GROUP_BY setting set to ON.
- Do NOT include extra or unnecessary code in the script.

Tools
Use MySQL Workbench to create and test all scripts.

Submission Method
Use the following process to submit your work for this assignment:

- Locate the properly named directory associated with your assignment in the file system (see *File and Directory Naming,* below).
- Compress that directory into a single .ZIP file using a utility program. NOTE: Only one file may be submitted. File types other than .ZIP will not be accepted and will receive a grade of zero.
- Submit the properly named zip file to the submission activity for this assignment.

File and Directory Naming

Please note that file and directory names must be in all lower case. Deductions will be made for submissions that do not follow this standard.

Please use the following naming scheme for the directory that holds your scripts:

```
surname_givenname_mgs_chap_07
```

If this were my own project, I would name my PyCharm project as follows:

```
trainor_kevin_mgs_chap_07
```

A separate solution script file must be submitted for each exercise. Solution scripts must be named using the following form: ex_xx_yy.sql (where xx is the two-digit chapter number [04] and yy is the two-digit exercise number [01]). So, an example of a properly formed solution script file example would be:

```
ex_07_01.sql
```

Use a zip utility to create one zip file that contain the PyCharm project directory. The zip file should be named according to the following scheme:

```
surname_givenname_mgs_chap_07.zip
```

If this were my own project, I would name the zip file as follows:

```
trainor_kevin_mgs_chap_07.zip
```

Due By

Please submit this assignment by the date and time shown in the Weekly Schedule.

Last Revised

2021-02-24

Please see the exercises on the attached sheets

Chapter 7
How to code subqueries

Exercises

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. 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 result set by the list_price column in descending sequence.

3. 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.

4. 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. Sort the result set by the largest order in descending sequence.

5. 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 result set by the product_name column.

6. 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.

   Sort the result set by the order_date and order_id columns.