Murach 4e Chapter 3 Coding Assignment Instructions

Exercises to be Completed
Complete exercises as follows:

- Exercise 1 through 7 are regular exercises.
- Exercise 8 is the challenge exercise.

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 sort order, then use that order. Otherwise, choose any reasonable order.
- 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 graded.
- 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_03`

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

`trainor_kevin_mgs_chap_03`

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 [03] and yy is the two-digit exercise number [01]). So, an example of a properly formed solution script file example would be:

`ex_03_01.sql`

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

`surname_givenname_mgs_chap_03.zip`

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

`trainor_kevin_mgs_chap_03.zip`

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

Last Revised
2025-01-26

Please see the exercises on the attached sheets!

Chapter 3

How to retrieve data from a single table

Exercises

Enter and run your own SELECT statements

In these exercises, you'll enter and run your own SELECT statements.

1. Write a SELECT statement that returns four columns from the Products table: product_code, product_name, list_price, and discount_percent. Then, run this statement to make sure it works correctly.

   Add an ORDER BY clause to this statement that sorts the result set by list price in descending sequence. Then, run this statement again to make sure it works correctly. This is a good way to build and test a statement, one clause at a time.

2. Write a SELECT statement that returns one column from the Customers table named full_name that joins the last_name and first_name columns.

   Format this column with the last name, a comma, a space, and the first name like this:

   ```
   Doe, John
   ```

   Sort the result set by the last_name column in ascending sequence.

   Return only the customers whose last name begins with letters from M to Z.

   NOTE: When comparing strings of characters, 'M' comes before any string of characters that begins with 'M'. For example, 'M' comes before 'Murach'.

3. Write a SELECT statement that returns these columns from the Products table:

product_name	The product_name column
list_price	The list_price column
date_added	The date_added column

   Return only the rows with a list price that's greater than 500 and less than 2000.

   Sort the result set by the date_added column in descending sequence.

4. Write a SELECT statement that returns these column names and data from the Products table:

product_name	The product_name column
list_price	The list_price column
discount_percent	The discount_percent column
discount_amount	A column that's calculated from the previous two columns
discount_price	A column that's calculated from the previous three columns

   Round the discount_amount and discount_price columns to 2 decimal places.

   Sort the result set by the discount_price column in descending sequence.

   Use the LIMIT clause so the result set contains only the first 5 rows.

5. Write a SELECT statement that returns these column names and data from the Order_Items table:

item_id	The item_id column
item_price	The item_price column
discount_amount	The discount_amount column
quantity	The quantity column
price_total	A column that's calculated by multiplying the item price by the quantity
discount_total	A column that's calculated by multiplying the discount amount by the quantity
item_total	A column that's calculated by subtracting the discount amount from the item price and then multiplying by the quantity

   Only return rows where the item_total is greater than 500.

   Sort the result set by the item_total column in descending sequence.

Work with nulls and test expressions

6. Write a SELECT statement that returns these columns from the Orders table:

order_id	The order_id column
order_date	The order_date column
ship_date	The ship_date column

   Return only the rows where the ship_date column contains a null value.

7. Write a SELECT statement without a FROM clause that uses the NOW and CURRENT_DATE functions to create a row with these columns:

today_unformatted	The value of the NOW function unformatted
today_formatted	The value of the NOW function in this format: DD-Mon-YYYY

   This displays a number for the day, an abbreviation for the month, and a four-digit year.

8. [*Challenge Exercise*] Write a SELECT statement without a FROM clause that creates a row with these columns:

price	100 (dollars)
tax_rate	.07 (7 percent)
tax_amount	The price multiplied by the tax
total	The price plus the tax

   To calculate the fourth column, add the expressions you used for the first and third columns.