BASIC SQL PRACTICE

MURACH
Chapter 3

1. Start MySQL Workbench
2. Go to http://jackmyers.info/db/sql/
   Open the query named 3-02.sql. When it opens, you should see several queries. Note that each of these queries has a semicolon at the end of it.

   This exercise uses the ap schema.
3. Move the insertion point into the first query and press Ctrl+Enter or click on the Execute Current Statement button to run the query. This shows you the data that's in the Invoices table that you'll be working with.
4. Move the insertion point into the second query and run it.
5. Open and run any other Chapter 3 queries
6. Write a SELECT statement that returns three columns from the Vendors table: vendor_name, vendor_contact_last_name, and vendor_contact_first_name. Then, run this statement to make sure it works correctly.

   Add an ORDER BY clause to this statement that sorts the result set by last name and then first name, both in ascending 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.
7. Write a SELECT statement that returns one column from the Vendors table named full_name that joins the vendor_contact_last_name and vendor_contact_first_name columns.

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

   ```
   Doe, John
   ```

   Refer to Murach: pg. 86 or Dubois: 1.4.9.5. Sort the result set by last name and then first name in ascending sequence.

   Return only the contacts whose last name begins with the letter A, B, C or E. This sould retrieve 41 rows.
8. Write a SELECT statement that returns these column names and data from the Invoices table:

Column Name	Data Returned
Due Date	The invoice_due_date column
Invoice Total	The invoice_total column
10%	10% of the value of invoice_total
Plus 10%	The value of the invoice_total plus 10%

   Return only the rows with an invoice total that's greater than or equal to 500 and less than or equal to 1000. This should retrieve 12 rows.

   Sort the result set in descending sequence by invoice_due_date.

9. Write a SELECT statement that returns these columns from the Invoices table:

Column Name	Data Returned
invoice_number	The invoice_number column
invoice_total	The invoice_total column
payment_credit_total	Sum of the payment_total and credit_total columns
balance_due	The invoice_total column minus the payment_total and credit_total columns

Return only invoices that have a balance due that's greater than $50.

Sort the result set by balance due in descending sequence.

Use the LIMIT clause so the result set contains only the rows with the 5 largest balances.

10. Write a SELECT statement that returns these columns from the Invoices table:

Column Name	Data Returned
invoice_number	The invoice_number column
invoice_date	The invoice_date column
balance_due	The invoice_total column minus the payment_total and credit_total columns
payment_date	The payment_date column

Return only the rows where the payment_date column contains a null value. This should return 11 rows.

11. Write a SELECT statement without a FROM clause that uses the CURRENT_DATE function to return the current date in its default format. (Cf: Murach 88-89)

Use the DATE_FORMAT function to format the current date in this format: **`mm-dd-yyyy.`** This displays the month, day, and four-digit year of the current date.

Give this column an alias of current_date. To do that, you must enclose the alias in quotes, since that name is already used by the CURRENT_DATE function.

12. Write a SELECT statement without a FROM clause that creates a row with these columns:

Column Name	Data Returned
starting_principal	Starting principal of $50,000
interest	6.5% of the principal
principal_plus_interest	The principal plus the interest.

To calculate the third column, add the expression you used for the first two columns.