In-Class Exercise #1 (In breakout rooms)

Solutions to these exercises will be posted in Blackboard.

Enter and run your own SELECT statements In these exercises, you’ll enter and run your own SELECT statements. To do that, you can open the script for an example that is similar to the statement you need to write, copy the statement into a new SQL tab, and modify the statement. That can save you both time and syntax errors.

1. 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.Text

Description automatically generated

2. 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, JohnSort 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 should retrieve 41 rows.

Background pattern

Description automatically generated with medium confidence

Text

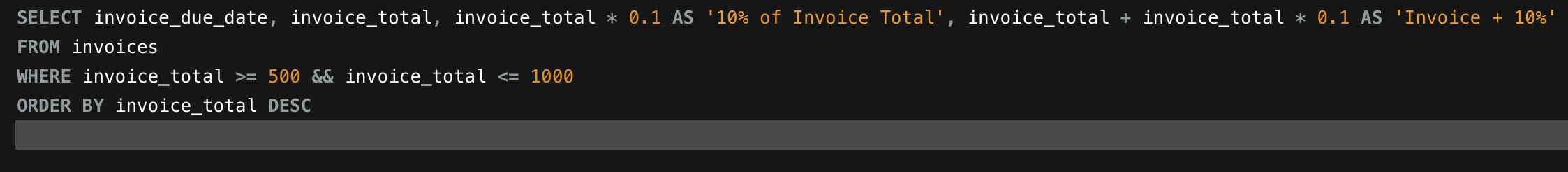
Description automatically generated

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

Due Date - The invoice\_due\_date column Invoice Total - The invoice\_total column

10% - 10% of the value of the invoice\_total Plus 10% - The value of 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.



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

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.

Work with nulls and test expressions

5. Write a SELECT statement that returns these columns from the Invoices table: invoice\_numberinvoice\_date balance\_duepayment\_dateThe invoice\_number column The invoice\_date columnThe invoice\_total column minus the payment\_total and credit\_total columnsThe payment\_date column

Return only the rows where the payment\_date column contains a null value. This should retrieve 11 rows.

6. Write a SELECT statement without a FROM clause that uses the CURRENT\_DATE function to return the current date in its default format. 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.

7. Write a SELECT statement without a FROM clause that creates a row with these columns: 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 expressions you used for the first two columns.