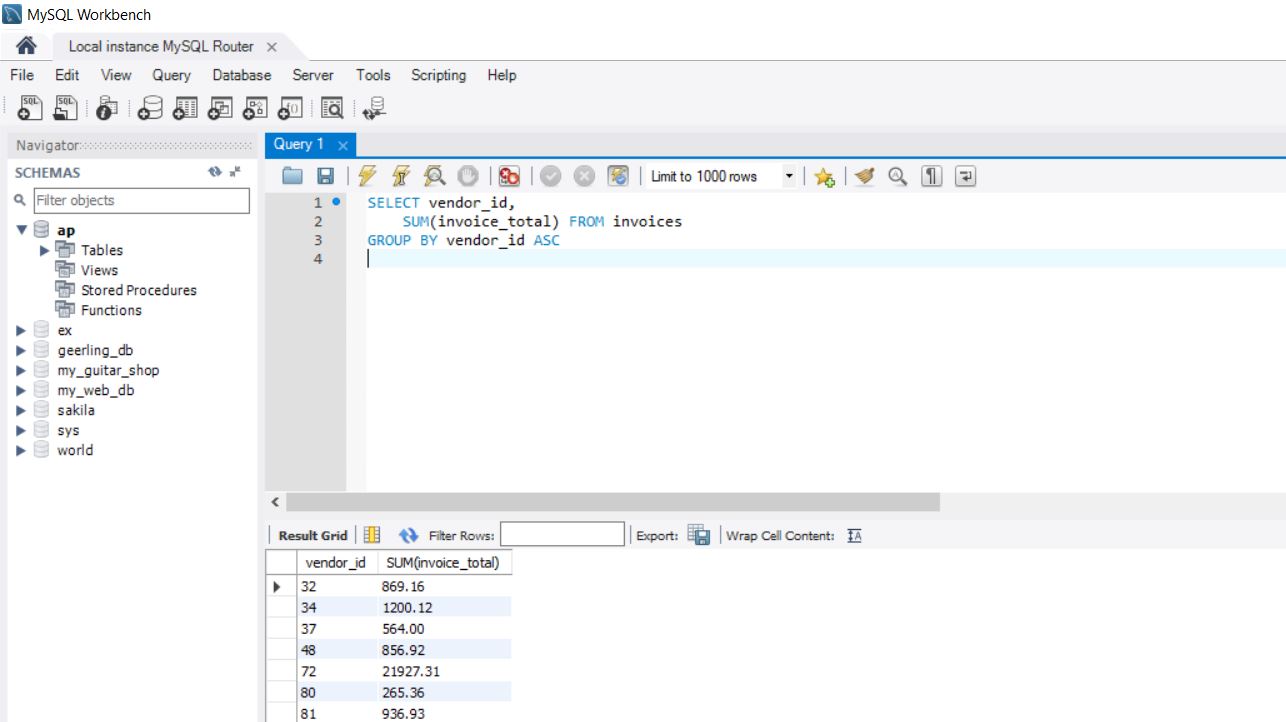
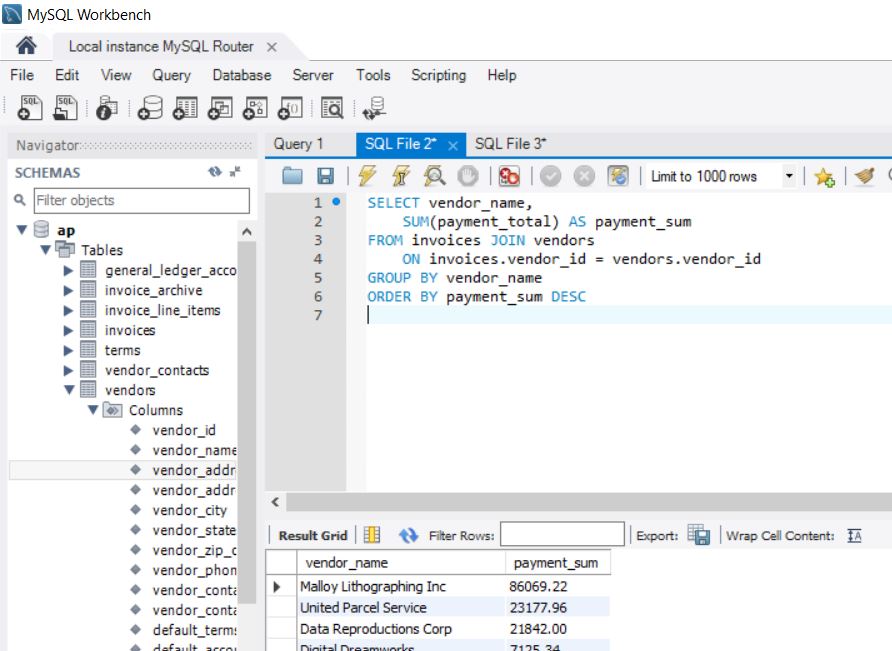
**Chapter 6:**

**Exercises**

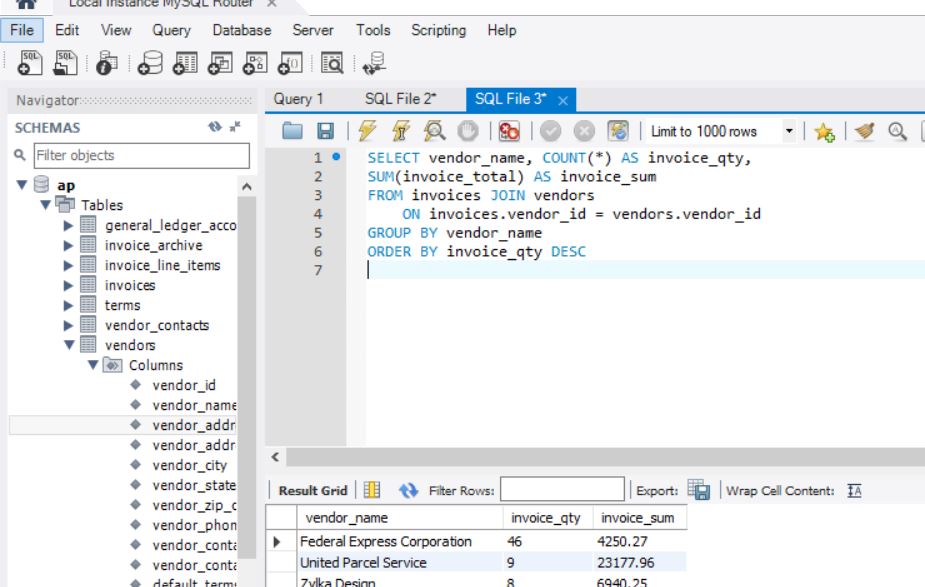
1. Write a SELECT statement that returns one row for each vendor in the Invoices table that contains these columns:
   1. The vendor\_id column from the Vendors table
   2. The sum of the invoice\_total columns in the Invoices table for that vendor
   3. This should return 34 rows.



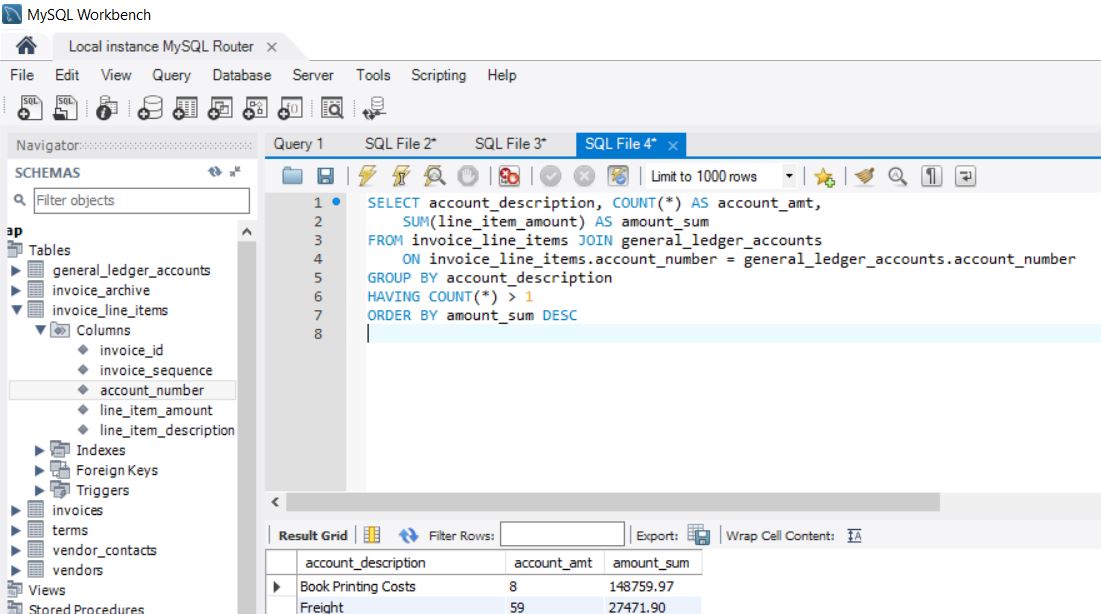
1. Write a SELECT statement that returns one row for each vendor that contains these columns:
   1. The vendor\_name column from the Vendors table
   2. The sum of the payment\_total columns in the Invoices table for that vendor
   3. Sort the result set in descending sequence by the payment total sum for each vendor.



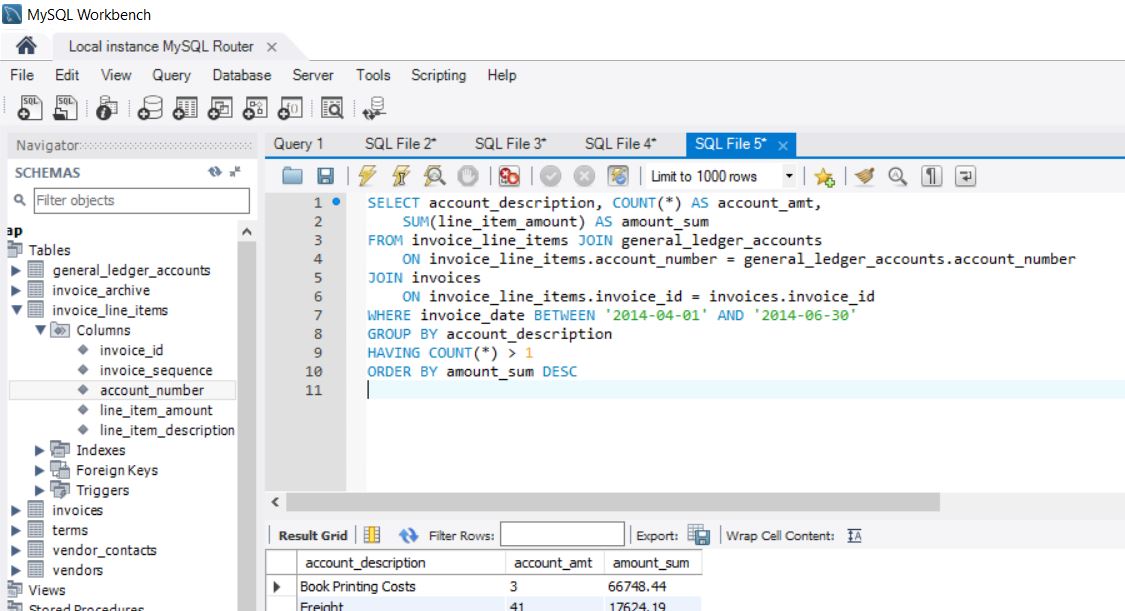
1. Write a SELECT statement that returns one row for each vendor that contains three columns:
   1. The vendor\_name column from the Vendors table
   2. The count of the invoices in the Invoices table for each vendor
   3. The sum of the invoice\_total columns in the Invoices table for each vendor
   4. Sort the result set so the vendor with the most invoices appears first.



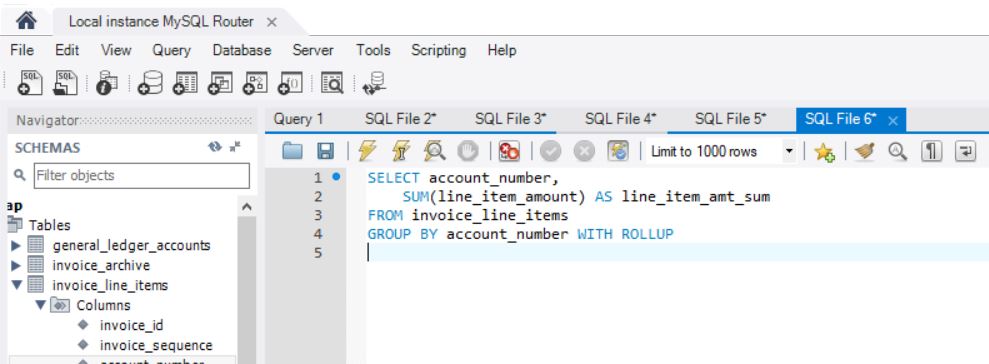
1. Write a SELECT statement that returns one row for each general ledger account number that contains three columns:
   1. The account\_description column from the General\_Ledger\_Accounts table
   2. The count of the items in the Invoice\_Line\_Items table that have the same account\_number
   3. The sum of the line\_item\_amount columns in the Invoice\_Line\_Items table that have the same account\_number
   4. Return only those rows where the count of line items is greater than 1. This should return 10 rows.
   5. Group the result set by account description.
   6. Sort the result set in descending sequence by the sum of the line item amounts.



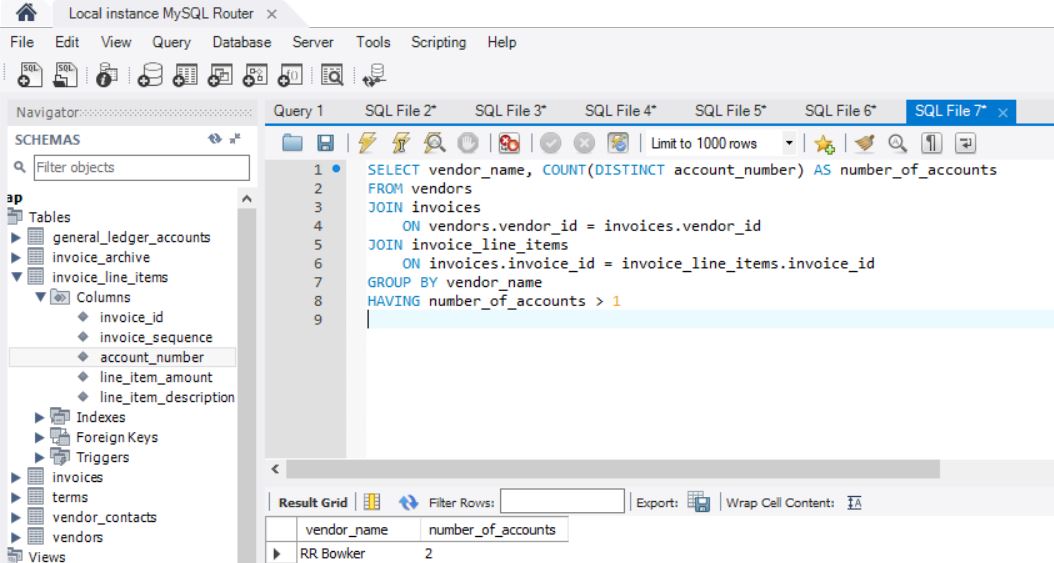
1. Modify the solution to exercise 4 so it returns only invoices dated in the second quarter of 2014 (April 1, 2014 to June 30, 2014.) This should still return 10 rows but with some different line item counts for each vendor. (Hint: Join to the Invoices table to code a search condition based on invoice\_date.)



1. Write a SELECT statement that answers this question: What is the total amount invoiced for each general ledger account number? Return these columns:
   1. The account number from the Invoice\_Line\_Items table
   2. The sum of the line item amounts from the Invoice\_Line\_Items table
   3. Use the WITH ROLLUP operator to include a row that gives the grand total. This should return 22 rows.
   4. Note: Once you add the WITH ROLLUP operator, you may need to use MySQL Workbench’s Execute SQL Script button instead of its Execute Current Statement button to execute this statement.



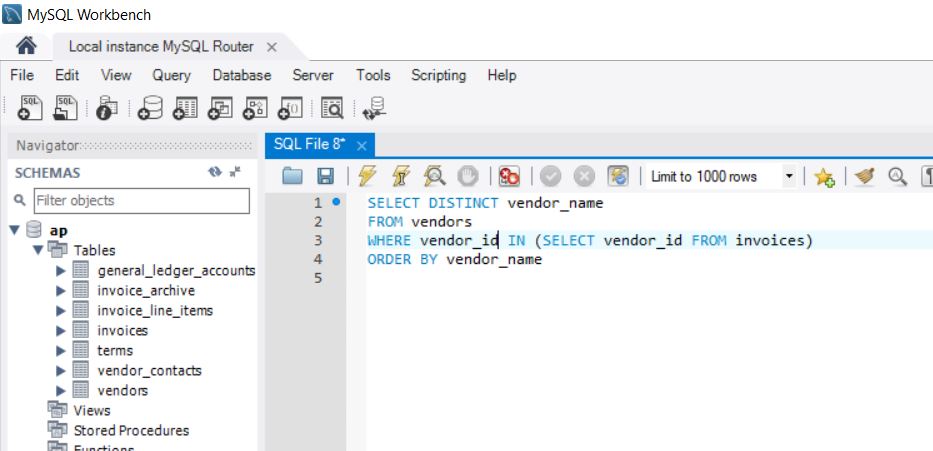
1. Write a SELECT statement that answers this question: Which vendors are being paid from more than one account? Return these columns:
   1. The vendor name from the Vendors table
   2. The count of distinct ledger accounts that apply to that vendor’s invoices
   3. This should return 2 rows.



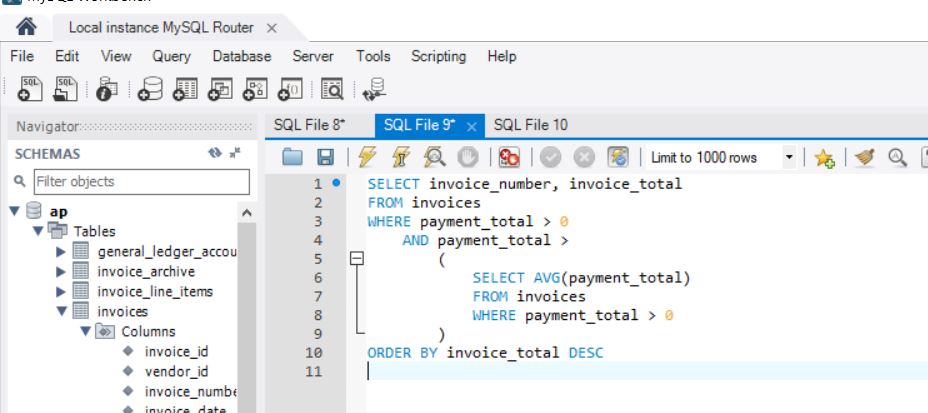
**Chapter 7:**

**Exercises**

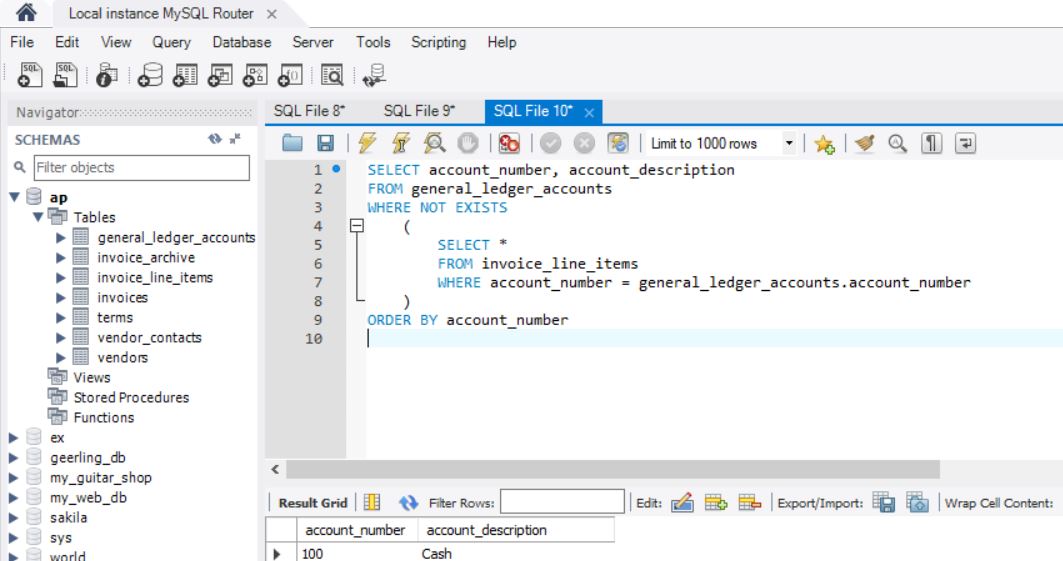
1. Write a SELECT statement that returns the same result set as this SELECT statement, but don’t use a join. Instead use a subquery in a WHERE clause that uses the IN keyword. [See statement in book]



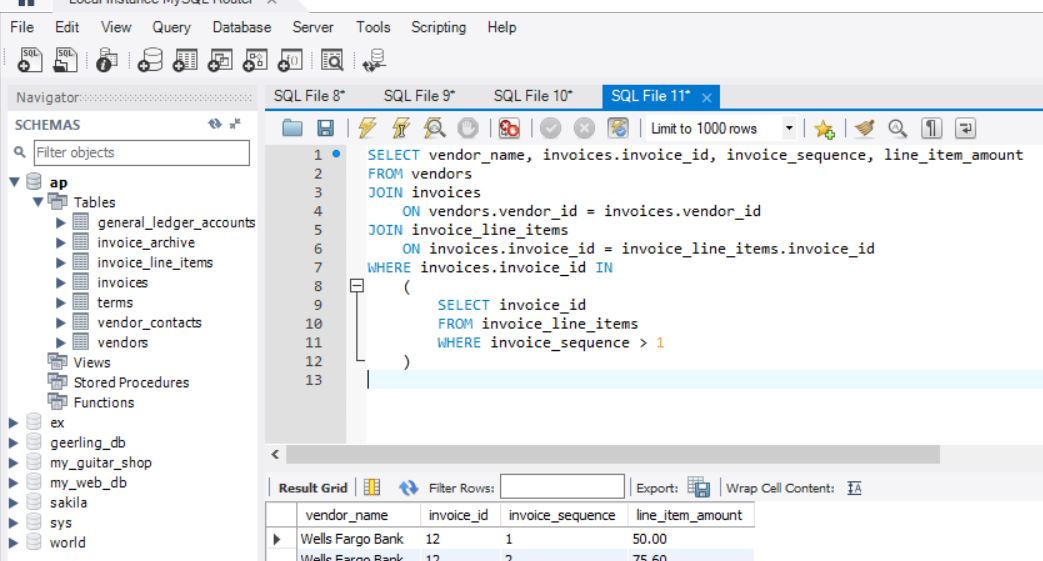
1. Write a SELECT statement that answers this question: Which invoices have a payment total that’s greater than the average payment total for all invoices with a payment total greater than 0?
   1. Return the invoice\_number and invoice\_total columns for each invoice. This should return 20 rows.
   2. Sort the results by the invoice\_total column in descending order.



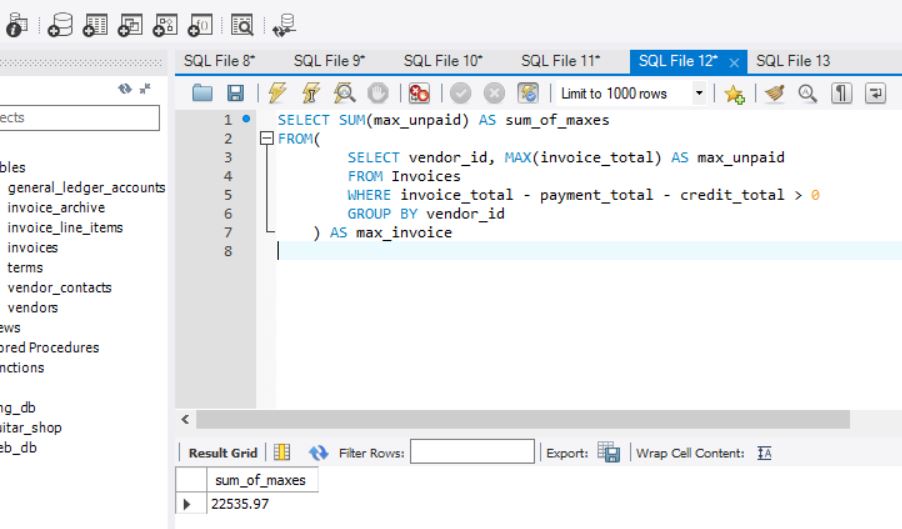
1. Write a SELECT statement that returns two columns from the General\_Ledger\_Accounts table: account\_number and account\_description.
   1. Return one row for each account number that has never been assigned to any line item in the Invoice\_Line\_Items table. To do that, use a subquery introduced with the NOT EXISTS operator. This should return 54 rows.
   2. Sort the results by the account\_number column.



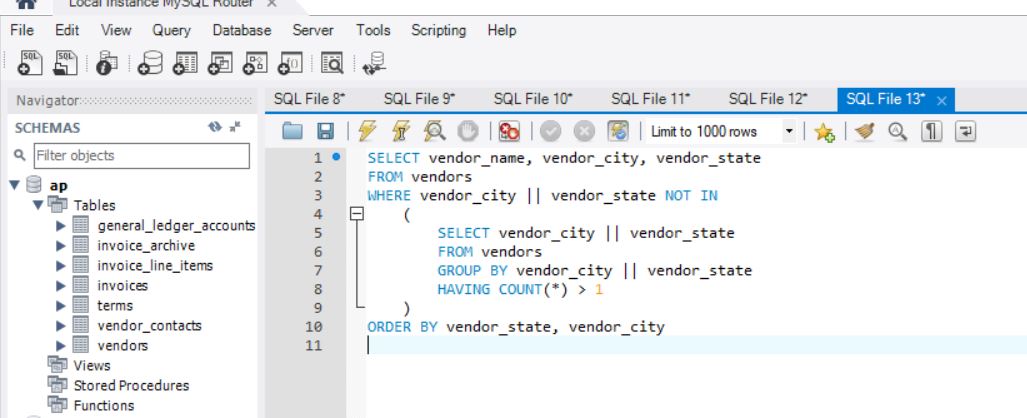
1. Write a SELECT statement that returns four columns: vendor\_name, invoice\_id, invoice\_sequence, and line\_item\_amount.
   1. Return a row for each line item of each invoice that has more than one line item in the Invoice\_Line\_Items table. Hint: Use a subquery that tests for invoice\_sequence > 1.



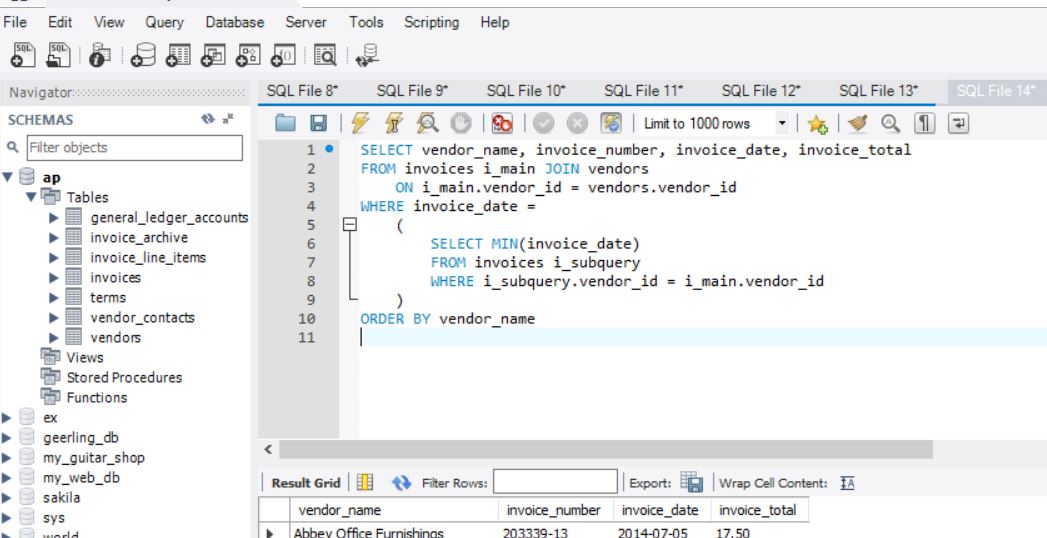
1. Write a SELECT statement that returns two columns: vendor\_id and the largest unpaid invoice for each vendor. To do this, you can group the result set by the vendor\_id column. This should return 7 rows.
   1. Write a second SELECT statement that uses the first SELECT statement in its FROM clause. The main query should return a single value that represents the sum of the largest unpaid invoices for each vendor.



1. Write a SELECT statement that returns the name, city, and state of each vendor that’s located in a unique city and state. In other words, don’t include vendors that have a city and state in common with another vendor. This should return 38 rows.
   1. Sort the results by the vendor\_state and vendor\_city columns.



1. Use a correlated subquery to return one row per vendor, representing the vendor’s oldest invoice (the one with the earliest date). Each row should include these four columns: vendor\_name, invoice\_number, invoice\_date, and invoice\_total. This should return 34 rows.
   1. Sort the results by the vendor\_name column.



1. Rewrite exercise 7 so it gets the same result but uses an inline view instead of a correlated subquery.

