**CCGC 5004 Database Systems**

**Lab Exercise 6 Summary Queries**

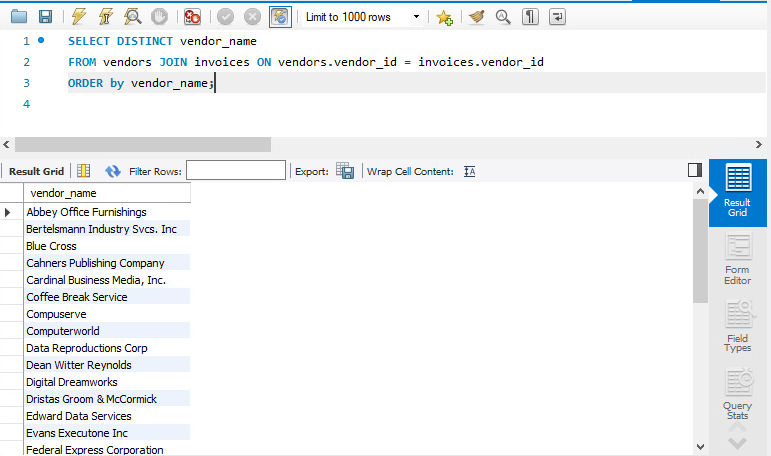
**Overview**

**To receive credit for this lab you must be present in today’s class. Late submissions are deducted 5% per day up to 5 days. Submissions received after 5 days will be given a grade of 0.**

**In this exercise you will download and run a script to populate your schema with the tables needed to perform the required queries.**

**You will write and execute queries as requested in the questions below.**

1. Write a **SELECT** statement that returns same result set as this **SELECT** statement, but do not use a join. Instead, use a subquery in a **WHERE** clause that uses the **IN** keyword. **Screen Capture 1**.



Returns 34 rows.

1. Write a **SELECT** statement that answers this question: Which invoices have a payment total that is greater than the average payment total for all invoices with a payment total > 0?

Return the **INVOICE\_NUMBER** and the **INVOICE\_TOTAL** columns for each invoice. This should return 20 rows

Sort the result by the **INVOICE\_TOTAL** column in descending order. **Screen Capture** **2**.

1. Write a **SELECT** statement that returns two columns from the **GENERAL\_LEDGER\_ACCOUNT** table: **ACCOUNT\_NUMBER** and **ACCOUNT\_DESCRIPTION**. **Screen Capture 3**.

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.

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**.

Return a row for each line item of each invoice that has more than one line in the **INVOICE\_LINE\_ITEMS** table. Hint, use a subquery that tests for **INVOICE\_SEQUENCE** > 1. This should return 6 rows.

Sort the results by the **VENDOR\_ID**, **INVOICE\_ID**, and **INVOICE\_SEQUENCE** columns.

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. **Screen capture 5.**

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**. Screen Capture 5A**.

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. **Screen capture 6.**

Sort the results by the **VENDOR\_NAME** column.

1. Rewrite exercise 6 so it gets the same result but uses an inline view instead of a correlated subquery. **Screen capture 7.**