**LAB 1**

**Please take screenshots after running each query and put all these screenshots on one Word file. Submit this file on Blackboard.**

**Exercise 1**

Write a SELECT statement that returns three columns from the Vendors table:

Vendor\_name, vendor\_contact\_last\_name, vendor\_contact\_first\_name. Then, run this statement to make sure it runs 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.

* Answer:

SELECT vendor\_name, vendor\_contact\_last\_name, vendor\_contact\_first\_name

FROM vendors

ORDER BY vendor\_contact\_last\_name, vendor\_contact\_first\_name

A screenshot of a computer

Description automatically generated

**Exercise 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, John**

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.

* Answer:

SELECT CONCAT(vendor\_contact\_last\_name, ', ', vendor\_contact\_first\_name) AS full\_name

FROM vendors

WHERE vendor\_contact\_last\_name < 'D' OR vendor\_contact\_last\_name LIKE 'E%'

ORDER BY vendor\_contact\_last\_name, vendor\_contact\_first\_name

A screenshot of a computer

Description automatically generated

**Exercise 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 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.

* Answer:

SELECT invoice\_due\_date AS "Due Date",

invoice\_total AS "Invoice Total",

invoice\_total / 10 AS "10%",

invoice\_total \* 1.1 AS "Plus 10%"

FROM invoices

WHERE invoice\_total >= 500 AND invoice\_total <= 1000

ORDER BY invoice\_due\_date DESC

A screenshot of a computer

Description automatically generated

**Exercise 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.

* Answer:

SELECT invoice\_number,

invoice\_total,

payment\_total + credit\_total AS payment\_credit\_total,

invoice\_total - payment\_total - credit\_total AS balance\_due

FROM invoices

WHERE invoice\_total - payment\_total - credit\_total > 50

ORDER BY balance\_due DESC

LIMIT 5;

A screenshot of a computer

Description automatically generated

**Exercise 5**

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

* Answer:

SELECT invoice\_number,

invoice\_date,

invoice\_total - payment\_total - credit\_total AS balance\_due,

payment\_date

FROM invoices

WHERE payment\_date IS NULL

A screenshot of a computer

Description automatically generated

**Exercise 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.

* Answer:

SELECT DATE\_FORMAT(CURRENT\_DATE, '%m-%d-%Y') AS "current\_date"

A screenshot of a computer

Description automatically generated

**Exercise 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.

* Answer:

SELECT 50000 AS starting\_principle,

50000 \* .065 AS interest,

(50000) + (50000 \* .065) AS principle\_plus\_interest

A screenshot of a computer

Description automatically generated