**CIS 386-DANG-HW3**

1. (5) Write a SELECT statement that returns these four columns where the balance due is greater than 200.

SELECT vendor\_name, invoice\_number, invoice\_date, (invoice\_total-payment\_total-credit\_total) AS balance\_due

FROM vendors v JOIN invoices i

ON v.vendor\_id = i.vendor\_id

WHERE (invoice\_total-payment\_total-credit\_total) > 200

ORDER BY balance\_due;

Graphical user interface, table

Description automatically generated

1. (5) Write a SELECT statement that returns three columns:

vendor\_id The vendor\_id column from the Vendors table

vendor\_name The vendor\_name column from the Vendors table

contact\_name A concatenation of the vendor\_contact\_first\_name and vendor\_contact\_last\_name columns with a space between

Return one row for each vendor whose contact has the same first name as another vendor’s contact. This should return 6 rows. Hint: Use a self-join to check that the vendor\_id columns aren’t equal but the vendor\_contact\_first\_name columns are equal. Sort the result set by vendor\_contact\_last\_name.

SELECT v.vendor\_id, v.vendor\_name, CONCAT(v.vendor\_contact\_first\_name," ", v.vendor\_contact\_last\_name) AS contact\_name

FROM vendors v JOIN vendors v1

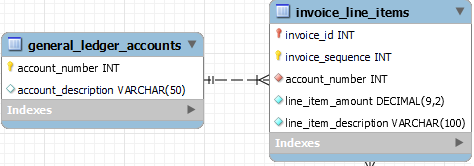
ON v.vendor\_contact\_first\_name = v1.vendor\_contact\_first\_name

AND v.vendor\_id <> v1.vendor\_id

ORDER BY v.vendor\_contact\_last\_name;

Graphical user interface, text, application

Description automatically generated

1. (5) Consider the general\_ledger\_accounts and invoice\_line\_items table.  
     
     
     
   Write a SELECT statement that returns these two columns, showing the accounts that do not appear on the invoice\_line\_items table.

account\_number The account\_number column from the General\_Ledger\_Accounts table

account\_description The account description column from the General\_Ledger\_Accounts table

Note that the returned account numbers have never been used.

SELECT g.account\_number, account\_description

FROM general\_ledger\_accounts g

LEFT JOIN invoice\_line\_items i

ON g.account\_number = i.account\_number

WHERE i.account\_number IS NULL;



1. (5) Show two columns from the Vendors table: vendor\_name and vendor\_phone. If the vendor has a phone number, the vendor\_phone value should be be its phone number. Otherwise, the vendor\_phone value should be “No Phone.” Sort the final result set by vendor\_name.

SELECT vendor\_name, "No Phone" AS vendor\_phone

FROM vendors

WHERE vendor\_phone IS NULL

UNION

SELECT vendor\_name, vendor\_phone AS vendor\_phone

FROM vendors

WHERE vendor\_phone IS NOT NULL

ORDER BY vendor\_name;

Table

Description automatically generated

1. (5) For each vendor, show vendor name, its invoice number, invoice date, invoice id, line item amount, and line item description. Sort the result by the invoice id.

SELECT v.vendor\_name, invoice\_number, invoice\_date, i.invoice\_id, line\_item\_amount, line\_item\_description

FROM vendors v JOIN invoices i

ON v.vendor\_id = i.vendor\_id

JOIN invoice\_line\_items il

ON i.invoice\_id = il.invoice\_id

ORDER BY invoice\_id;

Table

Description automatically generated

1. (5) Create a table called new\_terms from the terms table. Then, write an INSERT statement that adds this row to the new\_terms table:

terms\_id: 6terms\_description: Net due 150 days

terms\_due\_days: 150

Then, show the result of the following query: Select \* from terms;

CREATE TABLE new\_terms(

terms\_id INT AUTO\_INCREMENT PRIMARY KEY,

terms\_description varchar(255),

terms\_due\_date INT);

INSERT INTO new\_terms(terms\_id, terms\_description, terms\_due\_date)

VALUE (6, "Net due 150 days", 150);

SELECT \* FROM new\_terms;

Graphical user interface, application

Description automatically generated

1. (5) Write an UPDATE statement that modifies the row you just added to the Terms table. This statement should change the terms\_description column to “Net due 180 days”, and it should change the terms\_due\_days column to 180.

Then, show the result of the following query: Select \* from terms;

UPDATE new\_terms

SET terms\_description = "Net Due In 180 Days", terms\_due\_date = 180

WHERE terms\_id=6;

SELECT \* FROM new\_terms;

Graphical user interface

Description automatically generated with low confidence

1. (5) Write a DELETE statement that deletes the row you added to the Terms table (that is with the terms\_id 6).Then, show the result of the following query: Select \* from terms;

DELETE FROM new\_terms

WHERE terms\_id=6;

SELECT \* FROM new\_terms;

Text

Description automatically generated with medium confidence

1. (5) Write an UPDATE statement that modifies the Terms table. Change the terms\_description column to “Net due 100 days” and the terms\_due\_days column to 100 for the term with an ID of 5.

Then, show the result of the following query: Select \* from terms;

INSERT INTO new\_terms(terms\_id) VALUE (5);

UPDATE new\_terms

SET terms\_description = "Net Due In 100 Days", terms\_due\_date = 100

WHERE terms\_id =5;

SELECT \* FROM new\_terms;

Graphical user interface, application

Description automatically generated

1. (5) Write a SELECT statement that returns one row for each vendor in the Invoices table that contains these columns:

The vendor\_name column from the Vendors table

The sum of the invoice\_total columns in the Invoices table for that vendor

SELECT vendor\_name, SUM(invoice\_total) AS invoice\_total

FROM vendors v JOIN invoices i

ON v.vendor\_id = i.vendor\_id

GROUP BY vendor\_name;

Table

Description automatically generated

1. (10) Consider the following ER diagram.

Diagram, table

Description automatically generated with medium confidence

1. (2) What is the relationship between vendors and general\_ledger\_accounts? (Show whether it is 1:1, 1:Many, Many:1 or 1:1.  
    ***Many or One : One***
2. (4) Show the account\_description that has the most number invoices. What is the total invoice for the account? Show the query that finds the answer.;

Vendor with ID 123 has the most number invoices with 47 invoices.

SELECT i.vendor\_id, i.account\_description, MAX(total\_invoice) AS total\_invoice

FROM

(SELECT i.vendor\_id, COUNT(\*) AS total\_invoice, g.account\_description

FROM invoices i JOIN vendors v

ON i.vendor\_id = v.vendor\_id

JOIN general\_ledger\_accounts g

ON v.default\_account\_number = g.account\_number

GROUP BY vendor\_id

ORDER BY total\_invoice DESC) i;

1. (4) Which account\_description has the highest total invoice? Show the query that finds the answer.

Vendor with ID 110 has the highest invoices total with $118,982.41.

SELECT tab.vendor\_id, tab.account\_description, MAX(total\_invoice\_amt) AS total\_invoice

FROM(

SELECT i.vendor\_id, SUM(invoice\_total) AS total\_invoice\_amt, g.account\_description

FROM invoices i JOIN vendors v

ON i.vendor\_id = v.vendor\_id

JOIN general\_ledger\_accounts g

ON v.default\_account\_number = g.account\_number

GROUP BY vendor\_id

ORDER BY total\_invoice\_amt DESC) tab;



1. (10)
2. (5) Show the vendor name, line\_item\_description, and account number for each line items. Find vendors which use more than one account number from the result.

SELECT vendor\_id, account\_number, ili.line\_item\_description

FROM invoices i

JOIN invoice\_line\_items ili

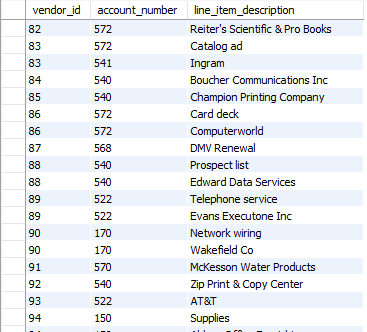
ON i.invoice\_id = ili.invoice\_id

UNION

(SELECT vendor\_id, default\_account\_number, vendor\_name

FROM vendors)

ORDER BY vendor\_id;



1. (5) Write a query to find out vendors which use more than one account.

SELECT tab.vendor\_id, tab.account\_number, tab.line\_item\_description

FROM (

SELECT vendor\_id, account\_number, ili.line\_item\_description

FROM invoices i

JOIN invoice\_line\_items ili

ON i.invoice\_id = ili.invoice\_id

UNION

(SELECT vendor\_id, default\_account\_number, vendor\_name

FROM vendors)

ORDER BY vendor\_id

) tab

GROUP BY tab.vendor\_id

HAVING COUNT(DISTINCT tab.account\_number) > 1 ;

Graphical user interface, application

Description automatically generated

1. (10) Find a vendor that had most late payments. How many times the payment was late? Consider the payment for each invoice was late if invoice\_due\_date < payment\_date.  
   Show the query which backs up your answer.

SELECT i.vendor\_id, MAX(a.late\_payment) AS number\_of\_late\_payments

FROM invoices i JOIN (

SELECT vendor\_id, COUNT(\*) AS late\_payment

FROM invoices

WHERE payment\_date - invoice\_due\_date > 0

GROUP BY vendor\_id) a;



1. (3) Write a query to find the average payment total for all invoices with a payment total greater than 0.  
   SELECT AVG(payment\_total) AS avg\_payment

FROM invoices

HAVING SUM(payment\_total)>0;



1. (7) Then, write a query that shows invoice\_id and its payment\_total where the payment\_total is greater than the average found in step (a).

SELECT i.invoice\_id, i.payment\_total, a.avg\_payment

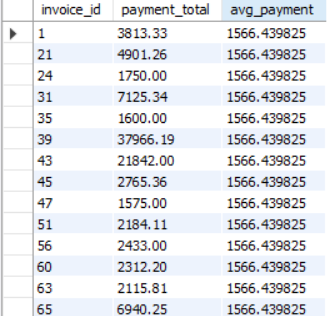
FROM invoices i

JOIN (SELECT AVG(payment\_total) AS avg\_payment

FROM invoices) a

WHERE i.payment\_total > a.avg\_payment

ORDER BY a.avg\_payment;



1. (10) Write a SELECT statement that returns two columns from the General\_Ledger\_Accounts table: account\_number and account\_description. 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. Sort the results by the account\_number column.

SELECT gen.account\_number, account\_description

FROM general\_ledger\_accounts gen

WHERE NOT EXISTS(

SELECT \* FROM invoice\_line\_items inv

WHERE gen.account\_number = inv.account\_number)

ORDER BY account\_number;

Table

Description automatically generated