

|  |
| --- |
| CS 1103 - FR02B  Assignment 3 |
| By Ngoc Phuong Anh Nguyen - 3712361  February 2021 |

# **Review Questions:**

## Questions 1: Explain why it would be preferable to use a DATE data type to store date data instead of a character data type.

it would be preferable to use a DATE data type to store date data instead of a character data type because it will be not regconized as numberic value if character data type is given. Otherwise, it is impossible if we calculate the date arithmetic if the data type is character.

## Question 7: Rewrite the following WHERE clause without the use of the IN special operator: WHERE V\_STATE IN (‘TN’, ‘FL’, ‘GA’)

Where V\_STATE = ‘TN’ or V\_STATE = ‘FL’ or V\_STATE = ‘GA’;

## Question 9: Explain why the following two commands produce different results: SELECT DISTINCT COUNT (V\_CODE) FROM PRODUCT; SELECT COUNT (DISTINCT V\_CODE) FROM PRODUCT;

The first commands put the key word “DISTINCT” before COUNT(), so the key word is applied to COUNT. In this case, “DISTINCT” is useless.

The second command has “DISTINCT” inside COUNT(), so it applies that key word to the V\_CODE. Therefore, only unique values are counted.

## Question 11: In a SELECT query, what is the difference between a WHERE clause and a HAVING clause?

* The WHERE clause selects rows before grouping. The HAVING clause selects row after grouping.
* The WHERE clause cannot contain aggregate functions, while the HAVING clause can contain them.
* The WHERE clause can be used without the GROUP BY clause. The HAVING clause cannot be used without the GROUP BY clause.

# Problems:

## The structure and contents of the Ch07\_SaleCo database are shown in Figure P7.9. Use this database to answer the following problems.

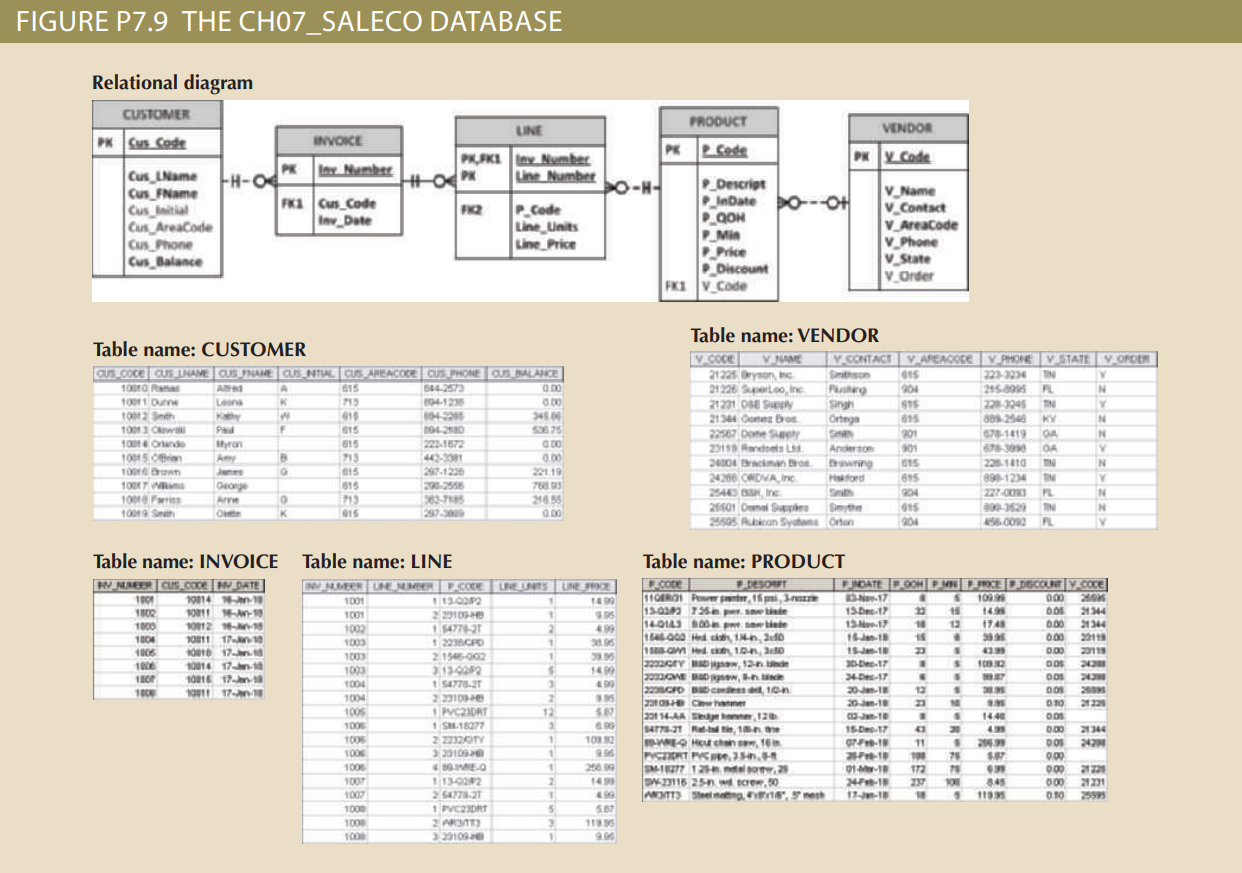


Figure P7.9 The Ch\_07\_SaleCo database

### Problems 9: Write a query to count the number of invoices.

select count(INV\_NUMBER) as NUMBER\_OF\_INVOICES from INVOICE;



Figure 1: Output result of Query 9

### Problem 10: Write a query to count the number of customers with a balance of more than $500.

select count(CUS\_CODE) as NUMBER\_OF CUSTIOMERS from CUSTOMER where CUS\_BALANCE >500;



Figure 2: Output result of Query 10

### Problem 12: Using the output shown in Figure P7.12 as your guide, generate a list of customer purchases, including the subtotals for each of the invoice line numbers. The subtotal is a derived attribute calculated by multiplying LINE\_UNITS by LINE\_PRICE. Sort the output by customer code, invoice number, and product description. Be certain to use the column aliases as shown in the figure

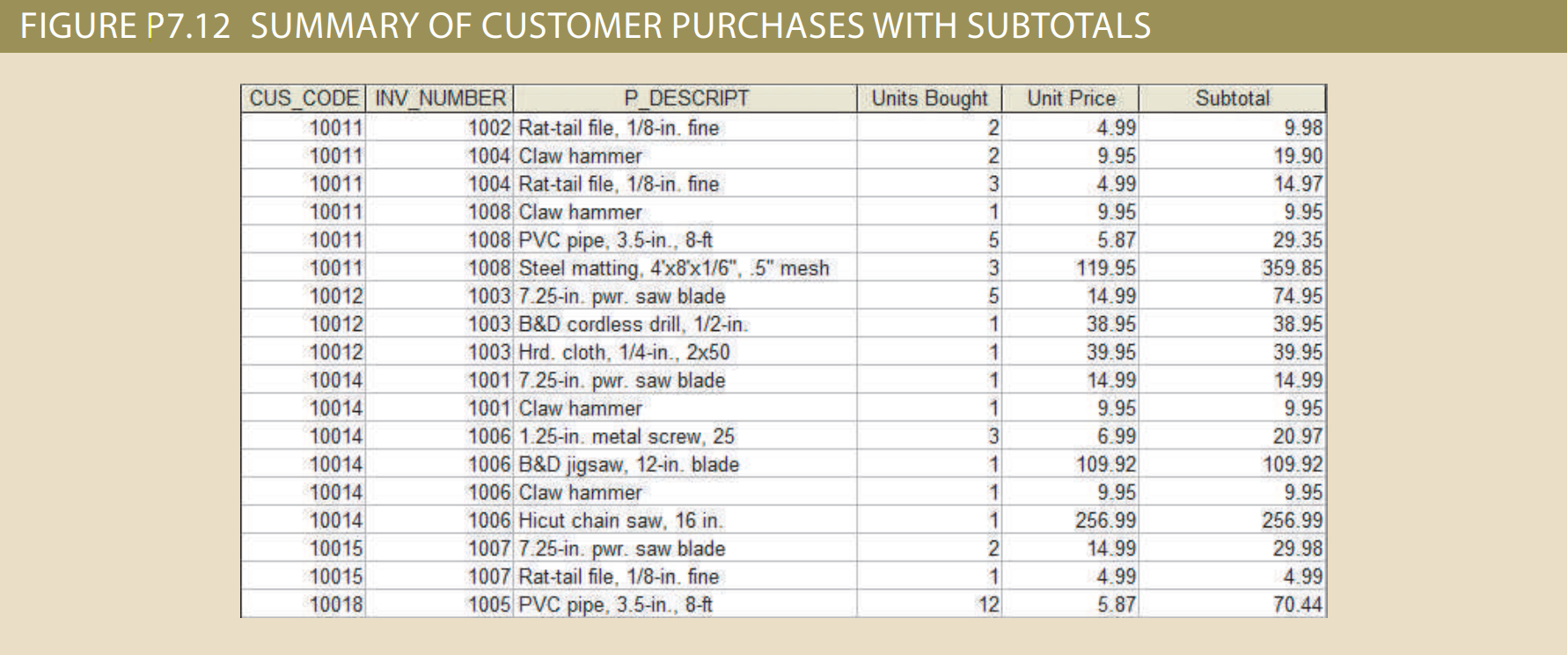


Figure P7.12: Summary of Customer Purchases with Subtotals

select C.CUS\_CODE, I.INV\_NUMBER,P.P\_DESCRIPT, L.LINE\_UNIT as 'Units Bought', L.LINE\_PRICE,(L.LINE\_UNIT\*L.LINE\_PRICE) as 'Subtotal'

from CUSTOMER as C, INVOICE as I, LINE as L, PRODUCT as P

where C.CUS\_CODE = I.CUS\_CODE and I.INV\_NUMBER = L.INV\_NUMBER and L.P\_CODE = P.P\_CODE

order by C.CUS\_CODE asc, I.INV\_NUMBER asc, P.P\_DESCRIPT asc;



Figure 3: Output result of Query 12

### Problem 15: Use a query to compute the total of all purchases, the number of purchases, and the average purchase amount made by each customer. Your output values must match those shown in Figure P7.15. Sort the results by customer code.

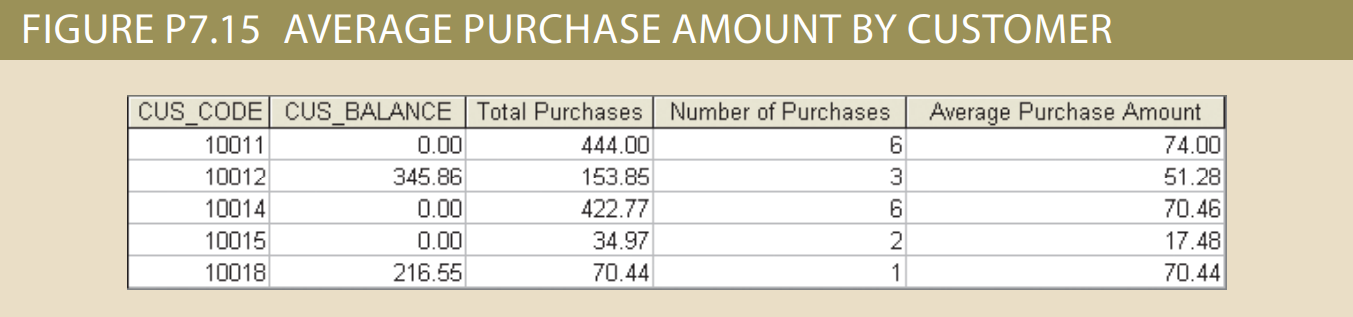


Figure P7.15: Average Purchase Amount by Customer

select CUS\_CODE, CUS\_BALANCE, sum(LINE\_PRICE\*LINE\_UNIT) as 'Total Purchases', count(LINE\_UNIT) as 'Number of Purchases', avg(LINE\_PRICE\*LINE\_UNIT) as 'Average Purchase Amount'

from INVOICE natural join LINE natural join CUSTOMER

group by CUS\_CODE;

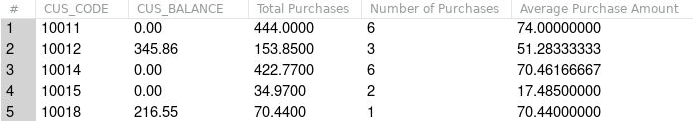


Figure 4: Output result of Query 15

### Problem 23: Find the listing of customers who did not make purchases during the invoicing period. Sort the results by customer code. Your output must match the output shown in Figure P7.23.

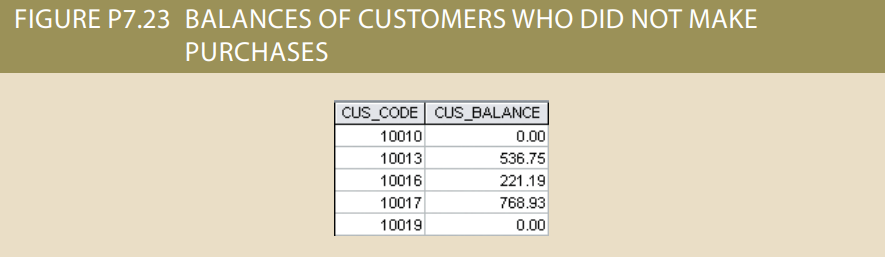


Figure P7.23: Balances Of Customers Who Did Not Make Purchases

select CUS\_CODE, CUS\_BALANCE

from CUSTOMER natural left join INVOICE

where INV\_NUMBER is null

order by CUS\_CODE;

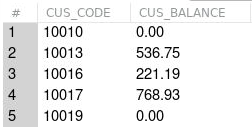
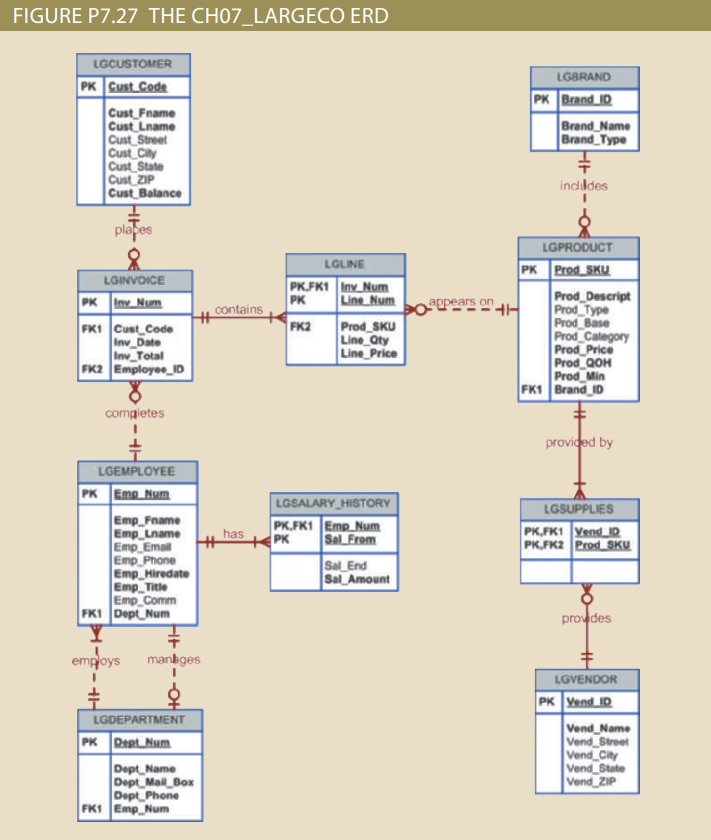


Figure 5: Output result of Query 23

## The Ch07\_LargeCo database (see Figure P7.27) stores data for a company that sells paint products. The company tracks the sale of products to customers. The database keeps data on customers (LGCUSTOMER), sales (LGINVOICE), products (LGPRODUCT), which products are on which invoices (LGLINE), employees (LGEMPLOYEE), the salary history of each employee (LGSALARY\_HISTORY), departments (LGDEPARTMENT), product brands (LGBRAND), vendors (LGVENDOR), and which vendors supply each product (LGSUPPLIES). Some of the tables contain only a few rows of data, while other tables are quite large; for example, there are only eight departments, but more than 3,300 invoices containing over 11,000 invoice lines. For Problems 28–55, a figure of the correct output for each problem is provided. If the output of the query is very large, only the first several rows of the output are shown.



P7.27: The Ch07\_LargeCo ERD

### Problem 27: Write a query to display the eight departments in the LGDEPARTMENT table sorted by department name.

select \* from LGDEPARTMENT

ORDER BY DEPT\_NAME;

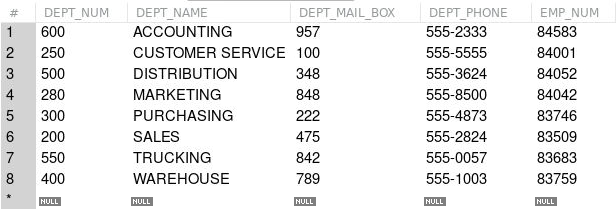


Figure 6: Output result of Query 27

### Problem 28: Write a query to display the SKU (stock keeping unit), description, type, base, category, and price for all products that have a PROD\_BASE of Water and a PROD\_ CATEGORY of Sealer (Figure P7.28).

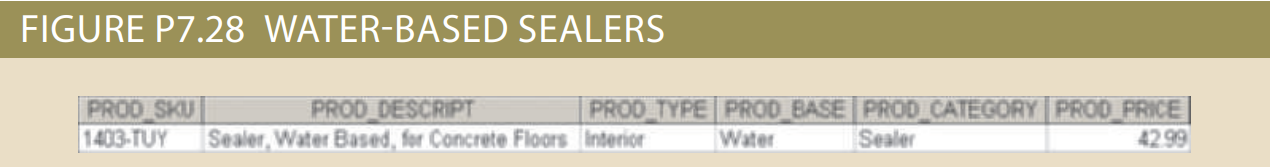


Figure P7.28 Water-Based Sealers

select PROD\_SKU as 'SKU', PROD\_DESCRIPT as 'Description', PROD\_TYPE as 'Type', PROD\_BASE as 'Base', PROD\_CATEGORY as 'Category', PROD\_PRICE as 'Price'

from LGPRODUCT

where PROD\_BASE = 'Water' and PROD\_CATEGORY = 'Sealer';



Figure 7: Output result of Query 28

### Problem 32: Write a query to display the first name, last name, street, city, state, and zip code of any customer who purchased a Foresters Best brand top coat between July 15, 2015, and July 31, 2015. If a customer purchased more than one such product, display the customer’s information only once in the output. Sort the output by state, last name, and then first name (Figure P7.32).

