**NAME**: Aimee Valladares

**Chapter 7 Questions**

**1.**  Using the data dictionary below write the MySQL code that will create the table structure for both the JOB and EMP\_1 tables. I expect that both primary and foreign keys are added to these tables as indicated in the data dictionary. I also expect if you use a foreign key that you create an INDEX for that foreign key (see lecture slide 57). Also add a CONSTRAINT CHECK or a CHECK that checks the JOB\_CODE in EMP\_1 is either 500, 501, or 502 – remember that JOB\_CODE is a CHAR(3) and not a number.

For example, if my name was Jane Smith, I would login the classroom VM and to MySQL and once in the mysql environment I would choose database *jsmith* with the command: *use jsmith;* In this jsmith database I will be able to create and delete tables.

|  |  |  |  |
| --- | --- | --- | --- |
| **Table** | **Key Type** | **Attribute (Field) Name** | **Data Declaration** |
| EMP\_1 | PK | EMP\_NUM | CHAR(3) |
|  |  | EMP\_LNAME | VARCHAR(15) |
|  |  | EMP\_FNAME | VARCHAR(15) |
|  |  | EMP\_INITIAL | CHAR(1) |
|  |  | EMP\_HIREDATE | DATE |
|  | FK | JOB\_CODE | CHAR(3) |
| JOB | PK | JOB\_CODE | CHAR(3) |
|  |  | JOB\_DESCRIPTION | VARCHAR(25) |
|  |  | JOB\_CHG\_HOUR | DECIMAL(12,2) |
|  |  | JOB\_LAST\_UPDATE | DATE |

create table JOB(

JOB\_CODE char(3) primary key not null,

JOB\_DESCRIPTION varchar(25) not null,

JOB\_CHG\_HOUR decimal(12,2) not null,

JOB\_LAST\_UPDATE date

);

create table EMP\_1(

EMP\_NUM char(3) primary key not null,

EMP\_LNAME varchar(15) not null,

EMP\_FNAME varchar(15) not null,

EMP\_INITIAL char(1),

EMP\_HIREDATE date not null,

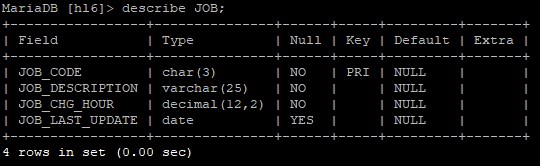
JOB\_CODE char(3),

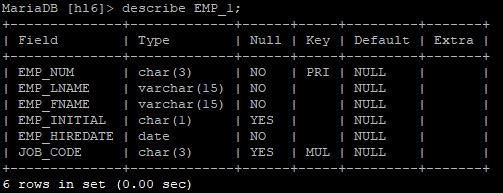
foreign key(JOB\_CODE) references JOB(JOB\_CODE) ON UPDATE CASCADE,

index (JOB\_CODE),

constraint JOB\_CK1 check(JOB\_CODE in('500', '501', '502'))

);





**2.** Now add the data into these two tables using INSERT commands and the two csv files I provided with the data records for each of the tables. On CANVAS you will find EMP\_1.csv with the 9 records for EMP\_1 table and JOB.csv file with 11 records for JOB table. ***For this problem you need to turn in is a paste of the INSERT statements only for each table EMP\_1 and JOB or the code you used to load the data into the tables inserted as plain text into your homework. In addition, you need to turn in the tables that appear after running the following two commands (including rows at bottom of the table) SELECT \* FROM EMP\_1; SELECT \* FROM JOB; inserted as images into your homework*** **(4 points)**

insert into JOB

(JOB\_CODE, JOB\_DESCRIPTION, JOB\_CHG\_HOUR, JOB\_LAST\_UPDATE)

values

(‘500’, ‘Programmer’, ‘35.75’, ‘2013-11-20’);

insert into JOB

(JOB\_CODE, JOB\_DESCRIPTION, JOB\_CHG\_HOUR, JOB\_LAST\_UPDATE)

values

('501', 'Systems Analyst', '96.75', '2013-11-20');

insert into JOB

(JOB\_CODE, JOB\_DESCRIPTION, JOB\_CHG\_HOUR, JOB\_LAST\_UPDATE)

values

('502', 'Database Designer', '125', '2014-03-24');

insert into JOB

(JOB\_CODE, JOB\_DESCRIPTION, JOB\_CHG\_HOUR, JOB\_LAST\_UPDATE)

values

('503', 'Electrical Engineer', '84.5', '2013-11-20');

insert into JOB

(JOB\_CODE, JOB\_DESCRIPTION, JOB\_CHG\_HOUR, JOB\_LAST\_UPDATE)

values

('504', 'Mechanical Engineer', '67.9', '2013-11-20');

insert into JOB

(JOB\_CODE, JOB\_DESCRIPTION, JOB\_CHG\_HOUR, JOB\_LAST\_UPDATE)

values

('505', 'Civil Engineer', '55.78', '2013-11-20');

insert into JOB

(JOB\_CODE, JOB\_DESCRIPTION, JOB\_CHG\_HOUR, JOB\_LAST\_UPDATE)

values

('506', 'Clerical Support', '26.87', '2013-11-20');

insert into JOB

(JOB\_CODE, JOB\_DESCRIPTION, JOB\_CHG\_HOUR, JOB\_LAST\_UPDATE)

values

('507', 'DSS Analyst', '45.95', '2013-11-20');

insert into JOB

(JOB\_CODE, JOB\_DESCRIPTION, JOB\_CHG\_HOUR, JOB\_LAST\_UPDATE)

values

('508', 'Applications Designer', '48.1', '2014-03-24');

insert into JOB

(JOB\_CODE, JOB\_DESCRIPTION, JOB\_CHG\_HOUR, JOB\_LAST\_UPDATE)

values

('509', 'Bio Technician', '34.55', '2013-11-20');

insert into JOB

(JOB\_CODE, JOB\_DESCRIPTION, JOB\_CHG\_HOUR, JOB\_LAST\_UPDATE)

values

('510', 'General Support', '18.36', '2013-11-20');

insert into EMP\_1

(EMP\_NUM, EMP\_LNAME, EMP\_FNAME, EMP\_INITIAL, EMP\_HIREDATE, JOB\_CODE)

values

('101', 'News', 'John', 'G', '2000-11-08', '502');

insert into EMP\_1

(EMP\_NUM, EMP\_LNAME, EMP\_FNAME, EMP\_INITIAL, EMP\_HIREDATE, JOB\_CODE)

values

('102', 'Senior', 'David', 'H', '1989-07-12', '501');

insert into EMP\_1

(EMP\_NUM, EMP\_LNAME, EMP\_FNAME, EMP\_INITIAL, EMP\_HIREDATE, JOB\_CODE)

values

('103', 'Arbough', 'June', 'E', '1996-12-01', '500');

insert into EMP\_1

(EMP\_NUM, EMP\_LNAME, EMP\_FNAME, EMP\_INITIAL, EMP\_HIREDATE, JOB\_CODE)

values

('104', 'Ramoras', 'Anne', 'K', '1987-11-15', '501');

insert into EMP\_1

(EMP\_NUM, EMP\_LNAME, EMP\_FNAME, EMP\_INITIAL, EMP\_HIREDATE, JOB\_CODE)

values

('105', 'Johnson', 'Alice', 'K', '1993-02-01', '502');

insert into EMP\_1

(EMP\_NUM, EMP\_LNAME, EMP\_FNAME, EMP\_INITIAL, EMP\_HIREDATE, JOB\_CODE)

values

('106', 'Smithfield', 'William', '', '2004-06-22', '500');

insert into EMP\_1

(EMP\_NUM, EMP\_LNAME, EMP\_FNAME, EMP\_INITIAL, EMP\_HIREDATE, JOB\_CODE)

values

('107', 'Alonzo', 'Maria', 'D', '1993-10-10', '500');

insert into EMP\_1

(EMP\_NUM, EMP\_LNAME, EMP\_FNAME, EMP\_INITIAL, EMP\_HIREDATE, JOB\_CODE)

values

('108', 'Washington', 'Ralph', 'B', '1991-08-22', '501');

insert into EMP\_1

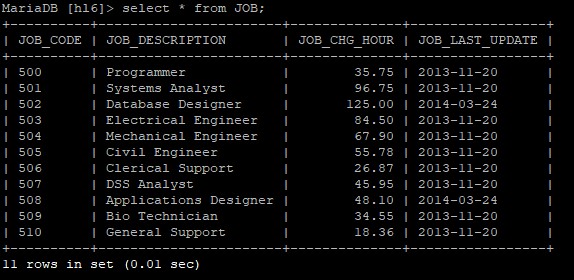
(EMP\_NUM, EMP\_LNAME, EMP\_FNAME, EMP\_INITIAL, EMP\_HIREDATE, JOB\_CODE)

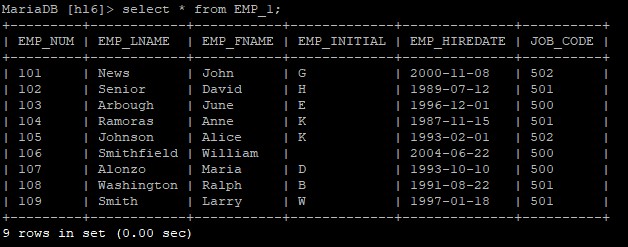
values

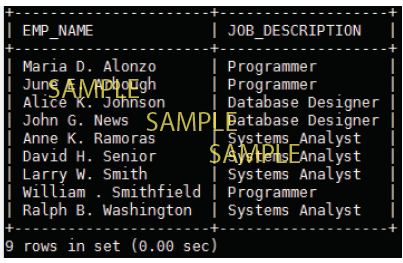
('109', 'Smith', 'Larry', 'W', '1997-01-18', '501');

select \* from JOB;

select \* from EMP\_1;





**3.** Now create and run a SQL query that displays only Employee Name (First name, Middle Initial, and Last name) and JOB\_DESCRIPTION sorted by Employee last name. See the picture below for the expected result. Note to accomplish this you must use CONCAT() function AND alias to get employee name grouped into one Attribute named EMP\_NAME along with sorting the data prior to display. See lecture slide 71 for CONCAT and Alias and lecture slides 89 & 90 for sorting a list. Additionally you will need to join EMP\_1 with JOB by JOB\_CODE but only select JOB\_DESCRIPTION to be displayed BESIDES the employee’s name. See lecture slides 100-104 and the textbook for details. ***For this problem you need to provide your MySQL code as plain text inserted into the homework and the resulting table pasted into your homework document as an image.*** **(3 points)**

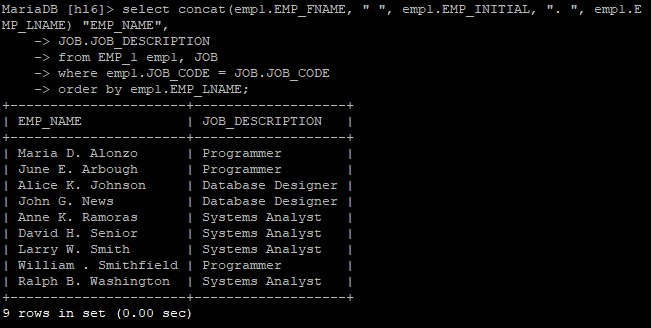
select concat(emp1.EMP\_FNAME, " ", emp1.EMP\_INITIAL, ". ", emp1.EMP\_LNAME) "EMP\_NAME",

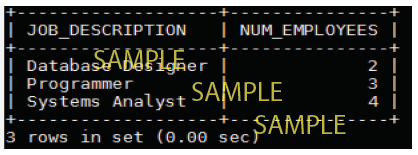
JOB.JOB\_DESCRIPTION

from EMP\_1 emp1, JOB

where emp1.JOB\_CODE = JOB.JOB\_CODE

order by emp1.EMP\_LNAME;



**4.** Now create and run SQL query that displays only JOB\_DESCRIPTION and the Number of employees in each job description sorted by JOB\_DESCRIPTION. See the picture below to know the expected result. Note to accomplish this you will need to join EMP\_1 with JOB by JOB\_CODE but only select JOB\_DESCRIPTION and you will need to use GROUP BY to get a count of how many employees have each JOB\_CODE/JOB\_DESCRIPTION. You will also need to use ORDER BY to sort the records by JOB\_DESCRIPTION. See lecture slides 95-97 and 100-104 and your textbook for more details. ***For this problem you need to provide your MySQL code as plain text inserted into the homework and the resulting table pasted as an image into your homework document.*** **(3 points)**

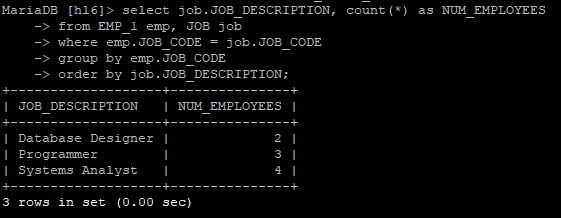
select job.JOB\_DESCRIPTION, count(\*) as NUM\_EMPLOYEES

from EMP\_1 emp, JOB job

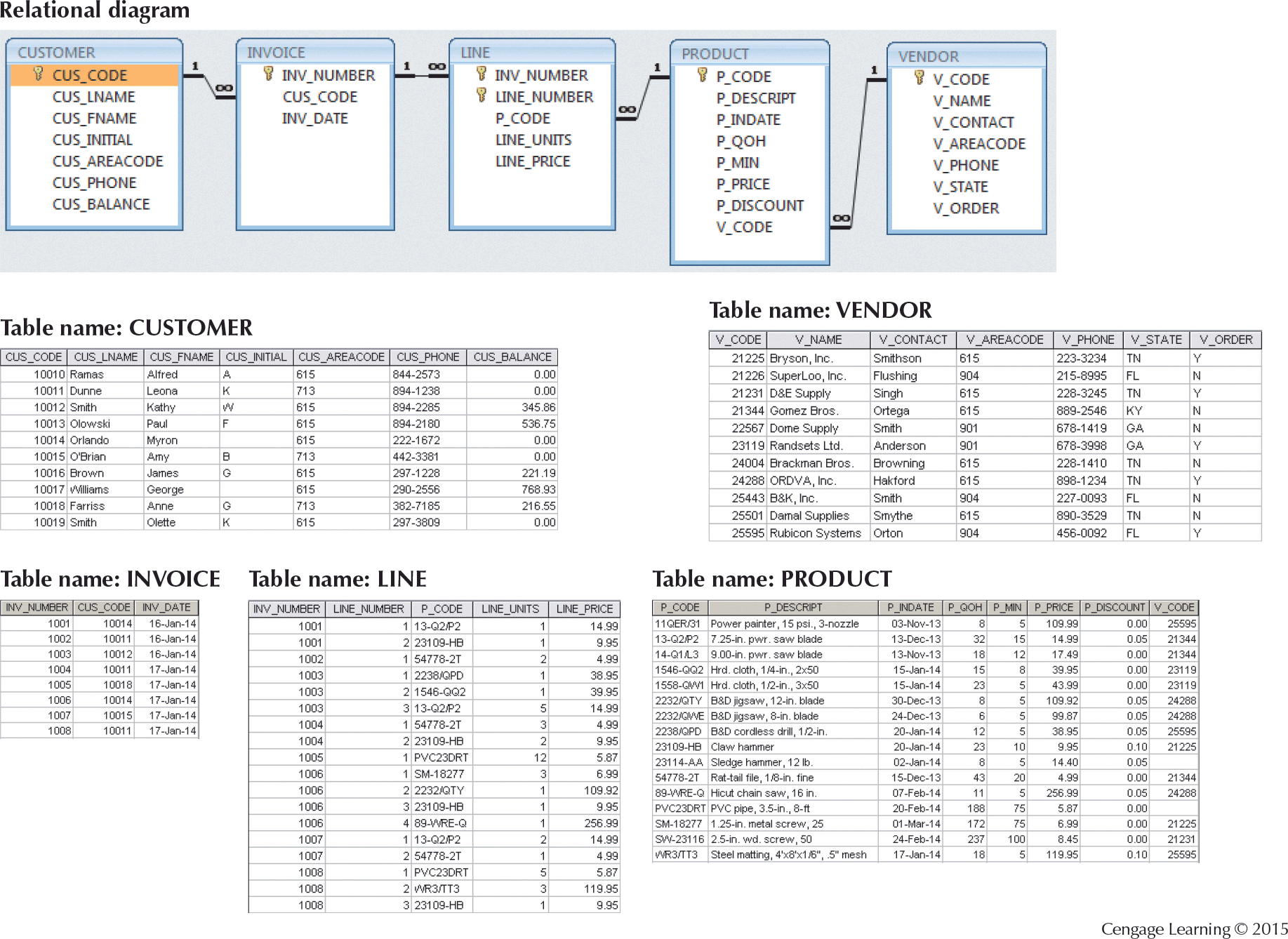
where emp.JOB\_CODE = job.JOB\_CODE

group by emp.JOB\_CODE

order by job.JOB\_DESCRIPTION;

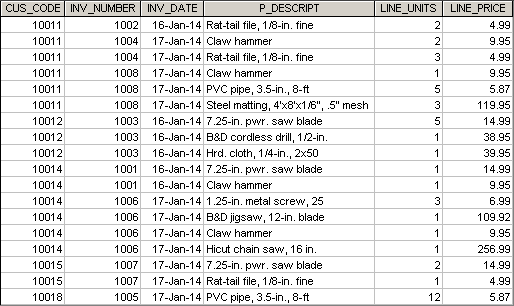


**PROBLEMS from CHAPTER 7 from the book use SALECO database:**

Login to the VM & MySQL, then select saleco database(SQL command: use saleco;) to answer these questions. Below is a relational diagram and tables from saleco (pg 299 in book). For these problems only provide the MySQL query you used to replicate the results that are shown in the tables below each problem. The questions are labeled 6-10 with their corresponding textbook question numbers indicated in parenthesis beside the question number.

**5**. **(28.)** Generate a listing of all purchases made by the customers, using the output shown in Figure P7.28 as your guide. (*Hint*: Use the ORDER BY clause to order the resulting rows as shown in Figure P7.28) ***Provide the MySQL query that gives results that match P7.28 below inserted as plain text into your homework AND provide the image of the table that results from your MySQL query pasted into your homework****.* **(4 points)**

**FIGURE P7.28 List of Customer Purchases**



select INVOICE.CUS\_CODE, INVOICE.INV\_NUMBER, INVOICE.INV\_DATE, PRODUCT.P\_DESCRIPT, LINE.LINE\_UNITS, LINE.LINE\_PRICE

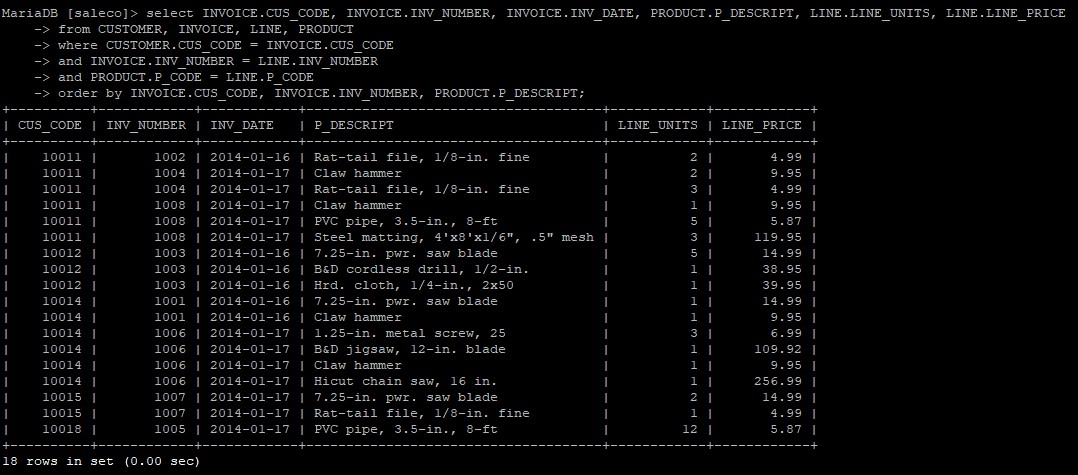
from CUSTOMER, INVOICE, LINE, PRODUCT

where CUSTOMER.CUS\_CODE = INVOICE.CUS\_CODE

and INVOICE.INV\_NUMBER = LINE.INV\_NUMBER

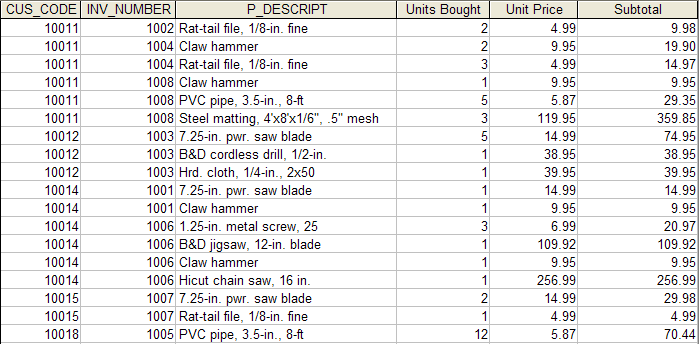
and PRODUCT.P\_CODE = LINE.P\_CODE

order by INVOICE.CUS\_CODE, INVOICE.INV\_NUMBER, PRODUCT.P\_DESCRIPT;



**6. (29.)** Using the output shown in Figure P7.29 as your guide, generate the listing of customer purchases, including the subtotals for each of the invoice line numbers. (*Hint*: Modify the query format used to produce the listing of customer purchases in Problem 18, delete the INV\_DATE column, and add the derived (computed) attribute LINE\_UNITS \* LINE\_PRICE to calculate the **subtotals** and rename LINE\_UNITS to Unit Bought and LINE\_PRICE to Unit Price.) ***Provide the MySQL query that gives results that match P7.29 below inserted as plain text into your homework AND provide the image of the table that results from your MySQL query pasted into your homework****.* **(4 points)**

**FIGURE P7.29 Summary of Customer Purchases with Subtotals**



select cus.CUS\_CODE, inv.INV\_NUMBER, pro.P\_DESCRIPT,

li.LINE\_UNITS "Units Bought", li.LINE\_PRICE "Unit Price",

round(li.LINE\_UNITS \* li.LINE\_PRICE, 2) "Subtotal"

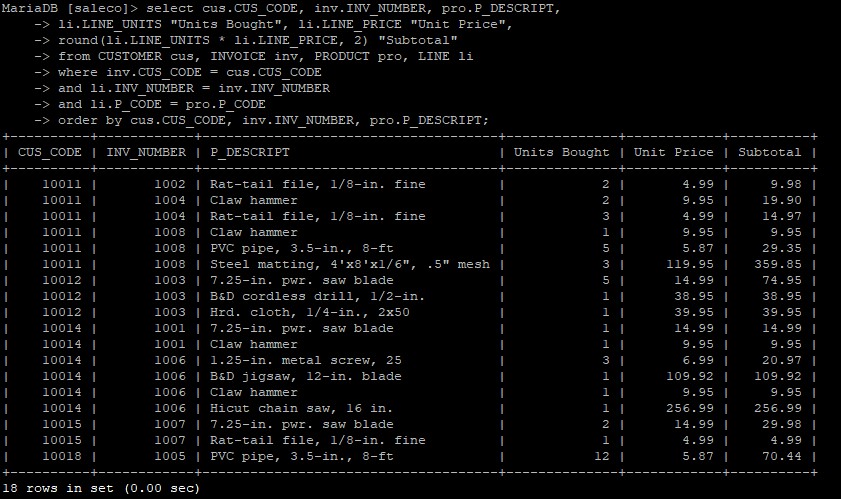
from CUSTOMER cus, INVOICE inv, PRODUCT pro, LINE li

where inv.CUS\_CODE = cus.CUS\_CODE

and li.INV\_NUMBER = inv.INV\_NUMBER

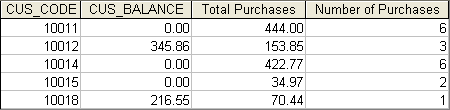
and li.P\_CODE = pro.P\_CODE

order by cus.CUS\_CODE, inv.INV\_NUMBER, pro.P\_DESCRIPT;



**7. (31.)** Modify the query in Problem 29 to provide a summary of the Total Purchases (Units Bought\* Unit Price) and Number of Purchases(Number of invoice lines – in the table from question 29, it’s number of times the CUS\_CODE appears in P7.29) per customer (CUS\_CODE). The result will include the number of individual product purchases made by each customer. (In other words, if the customer’s invoice is based on three products, one per LINE\_NUMBER, you would count three product purchases. If you examine the original invoice data, you will note that customer 10011 generated three invoices, which contained a total of six lines, each representing a product purchase.) Your output values must match those shown in Figure P7.31. Notice that besides the summary attributes of “Total Purchase” and “Number of Purchases”, your table only needs to include CUS\_BALANCE (see CUSTOMER) and CUS\_CODE. This means you will no longer need to merge with the PRODUCT table because you are no longer including P\_DESCRIPT in the resulting table. ***Provide the MySQL query that gives results that match P7.31 below inserted as plain text into your homework AND provide the image of the table that results from your MySQL query pasted into your homework****.* **(4 Points)**

**FIGURE P7.31 Customer Total Purchase Amounts and Number of Purchases**



select cus.CUS\_CODE, cus.CUS\_BALANCE,

sum(round(li.LINE\_UNITS \* li.LINE\_PRICE, 2)) "Total Purchases",

count(\*) as "Number of Purchases"

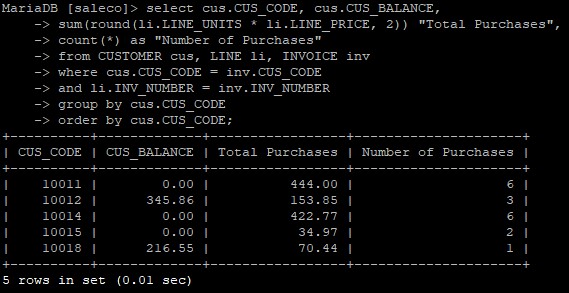
from CUSTOMER cus, LINE li, INVOICE inv

where cus.CUS\_CODE = inv.CUS\_CODE

and li.INV\_NUMBER = inv.INV\_NUMBER

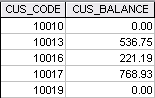
group by cus.CUS\_CODE

order by cus.CUS\_CODE;



**8. (40.)** Find the listing of customers who did not make purchases during the invoicing period – where all that is displayed is their CUS\_CODE and CUS\_BALANCE. Your output must match the output shown in Figure P7.40. ***Hint*** – to do this you need create a nested query where the inner query provides all distinct customers(CUS\_CODE) from the INVOICE table – because these are the customers that you will not show since they have had made recent purchase (e.g. they have INVOICES). To accomplish this you could simply want to display all CUS\_CODE from CUSTOMER where the CUS\_CODE is ***NOT IN*** with the inner query that identifies all unique CUS\_CODE from the INVOICE table. ***Provide the MySQL query that gives results that match P7.40 below inserted as plain text into your homework AND provide the image of the table that results from your MySQL query pasted into your homework****.* **(5 points)**

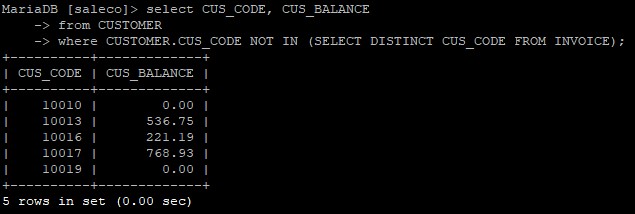
**FIGURE P7.40 Customer Balances for Customers Who Did Not Make Purchases**



select CUS\_CODE, CUS\_BALANCE

from CUSTOMER

where CUSTOMER.CUS\_CODE NOT IN (SELECT DISTINCT CUS\_CODE FROM INVOICE);



**9. (41)** Find the customer balance summary for all customers who have not made purchases during the current invoicing period. The results are shown in Figure P7.41. These are the same customers from P7.40 – question 9/40 above – except now you are using functions MIN, MAX, AVG, SUM to summarize CUS\_BALANCE for the whole group of customers shown above (CUS\_CODE = 10010, 10013, 10016, 10017, 10019). ***Provide the MySQL query that gives results that match P7.41 below inserted as plain text into your homework AND provide the image of the table that results from your MySQL query pasted into your homework****.* **(4 Points)**

**FIGURE P7.41 Summary of Customer Balances for Customers Who Did Not Make Purchases**

FigP7-41-Cust-Bal-for-Cust-not-Making-Purchases-Summ

select sum(cus.CUS\_BALANCE) "Total Balances",

min(cus.CUS\_BALANCE) "Minimum Balance",

max(cus.CUS\_BALANCE) "Maximum Balance",

round(avg(cus.CUS\_BALANCE),2) "Average Balance"

from CUSTOMER cus

where cus.CUS\_CODE not in(select inv.CUS\_CODE from INVOICE inv);

