CS 111 – Lab #13 – Subtotal queries using SQL

Please be sure you are finished with the previous lab before starting today’s activity. Lab 12 modeled a store and we made use of 5 tables in order to write interesting SELECT queries using SQL.

If you were having trouble with the last step of Lab 12, please note that in order to make use of the new Employee table, you needed to update the order table so that it also has a field for the Employee ID number. This field is what is in common to the two tables, so that the tables can be related to each other.

1. Log in to phpMyAdmin using your database username and password.

Today we are interested in doing subtotals. We have already seen that in SQL we can ask for a grand total. Let’s go back to a simple table – our original Employee table of financial service employees. Here is how we did a basic total:

Select sum(Salary) from Employee;

And we can replace the mathematical function “sum” with another aggregate function such as avg or count. In fact, we could ask for all three in the same query:

Select sum(Salary), count(Salary), avg(Salary) from Employee;

But what if we want to break a total down into subtotals? For instance, we may be interested in looking at salaries across different cities, or different job titles. To accomplish this, all we need to do is to tell SQL that we want to “group by” the field we want to subtotal by, and also include this field in the output. So, try this query:

Select **Location**, sum(Salary), count(Salary), avg(Salary) from Employee **group by Location**;

I’ve highlighted the part of the query that changed compared to the previous query you just did. (It’s a coincidence that all cities have 3 employees.)

If you replace “Location” with “Title” in the above subtotal query, you can find summary Salary data based on job title.

* 1. Write a select query that will determine, for each job title: the number of employees, the total of the salaries, the average, maximum and minimum salary.
	2. According to the result of the query, which job title has the lowest minimum salary?
1. It’s even possible to ask SQL to perform multiple levels of subtotals. Suppose we wanted to know how many employees there are each job function at each location? Writing a 2-level subtotal query is analogous to the subtotal queries you have already seen. All we need to do is specify the 2 fields we want to display and group by, like this:

Select **Location, Title**, count(Salary) from Employee group by **Location, Title**;

When you run this query, can you tell that there are some job titles missing from certain cities? Which ones don’t exist?

Note that since all we were interested in was just a count of the employees, we could have applied the “count” function to any of the fields.

1. Now we can apply our subtotal technique to the more interesting scenario of the multiple-table database. It is not hard to write queries that ask useful summary questions, such as how much money each customer has spent in total on orders. First, think about what fields we need to output in this query:
* Customer’s name, contained in the customer table
* Dollar amounts, contained in the product table
* Quantity of each product purchased in an order, contained in the order details table
* Order number, contained in the order table

It may help to first write a query that simply lists all the data we want, before worrying about the subtotals. That query would be similar to the work we did in the previous lab. Don’t forget that to write such a query, we must insist that the relevant “ID” fields match. In this case, we have a 4-table query, so there are 4 – 1 = 3 pairs of fields that need to match. Our initial query looks like this:

Select Lab12Customer.CID, Lab12Customer.First, Lab12Customer.Last, Lab12Orders.OID, Lab12Product.Name, Lab12Product.Price, Lab12OrderDetails.Quantity

from Lab12Customer, Lab12Orders, Lab12OrderDetails, Lab12Product

where Lab12Customer.CID = Lab12Orders.CID

and Lab12Orders.OID = Lab12OrderDetails.OID

and Lab12OrderDetails.PID = Lab12Product.PID;

Run this query to verify that the result makes sense to you. The result should show informative details about everything ordered by everybody.

Because we are eventually interested in adding up all the dollar values of the orders, we need to multiply the price by the quantity to produce a calculated field. A calculated field is a field that is not present in any table, but is useful in a query as output or for helping out with calculations later. In this case, a calculated field representing the total dollar value for each type of product ordered by a customer on a certain date is a necessary step.

How do we create a calculated field? This is also straightforward: in our Select statement, we simply tell SQL that we want to output another column of information. After printing the quantity, we want SQL to multiply the price and quantity values to produce the total $$$ for this product. Here is the updated query, with the new elements highlighted.

Select Lab12Customer.CID, Lab12Customer.First, Lab12Customer.Last, Lab12Orders.OID, Lab12Product.Name, Lab12Product.Price, Lab12OrderDetails.Quantity,

**Lab12Product.Price \* Lab12OrderDetails.Quantity**

from Lab12Customer, Lab12Orders, Lab12OrderDetails, Lab12Product

where Lab12Customer.CID = Lab12Orders.CID

and Lab12Orders.OID = Lab12OrderDetails.OID

and Lab12OrderDetails.PID = Lab12Product.PID;

When you run this query, in the result you’ll see a new column on the right containing the total dollar value of each record.

Unfortunately, when SQL multiplies real numbers it tends to give answers that are too precise. Earlier in the course we saw that many common real numbers cannot be exactly represented by the computer because they are not exact powers of 2. To fix this situation, we can use SQL’s round() function. This function takes 2 parameters. The first parameter is a real number we wish to round, and the second parameter is a number of decimal places. So, in this query above, change the 3rd line to this:

Round(Lab12Product.Price \* Lab12OrderDetails.Quantity, 2)

Edit and re-run the query. The output should look much better. In fact, for the case where the dollar amount is a whole number of dollars, SQL will display “.00” so that all the monetary values are consistent and aligned with one another. You may want to also apply the round() function to the Price column as well.

There is one thing left to do here: subtotals. We would like to find the total amount of money spent by each customer. Before we type in this new query, we should note that our result does not need so many columns. What information should we print out? It’s a good idea to have the customer’s name, and probably the customer ID as well. But we don’t need individual product information printed. That would defeat the purpose of doing the subtotal. We basically want to tell SQL to do this:

“Show me the customer’s name and total dollar amount, and subtotal by each name.”

While we don’t have do display as many columns of output, we still need to keep all of the “where” clauses in order to enforce all of the relationships among the tables. Our resulting query is:

Select Lab12Customer.CID, Lab12Customer.First, Lab12Customer.Last, round(Lab12Product.Price \* Lab12.OrderDetails.Quantity, 2)

from Lab12Customer, Lab12Orders, Lab12OrderDetails, Lab12Product

where Lab12Customer.CID = Lab12Orders.CID

and Lab12Orders.OID = Lab12OrderDetails.OID

and Lab12OrderDetails.PID = Lab12Product.PID

**group by Lab12Customer.CID**;

It’s very important to remember the “group by” clause. Why do we group by the CID field? Because we are guaranteed that every customer has a unique ID number. First names and last names would not be suitable because they are not necessarily unique in the database.

Looking at the result of the query, we can instantly see which customers spend the most money at the store. Very useful information!

1. Based on the previous example, create similar subtotal queries that answer these questions. As you did in the previous lab, you may prefer to copy/paste your query into a word processor or editor program rather than transcribe your (long) answers by hand.
	1. We are interested in knowing which sales persons have conducted the most ($$$) business. For each employee, show us the total dollar value of sales.
	2. Modify the previous query by computing a new calculated field that will represent the total commission earned by the employee. Assume the commission rate is 5% of the total sales by that employee.