CS 111 – Lab #12 – SQL queries for larger databases

If you have not done so already, please complete the previous lab, which was an introduction to SQL queries and phpMyAdmin. In today’s lab, we will practice working with multiple tables. You will probably find it helpful to refer to your Computer Store handout from class.

Throughout this lab, you will have several opportunities to write your own SQL queries. Because writing a lot of queries in addition to typing them on the keyboard can be tedious, you may find it more convenient to answer the questions in this lab electronically, such as in a plain text file, or an electronic version of this document. That way, you can copy/paste your answers rather than write them all by hand.

1. Begin by copying the 4 CSV files from the file server (Customers.csv, OrderDetails.csv, Orders.csv, Products.csv)
2. Log in to phpMyAdmin. You will need to create 4 new tables, one at a time.

Here are the details about the tables you need to create.

|  |  |  |
| --- | --- | --- |
| Name of table | How many fields | Field names, types, max values (if any) |
| Lab12Customer | 8 | CID = varchar, max length 3  First = varchar, max length 20  Last = varchar, max length 20  Address = varchar, max length 30  City = varchar, max length 20  State = varchar, max length 2  ZIP = varchar, max length 5  Phone = varchar, max length 10 |
| Lab12Product | 5 | PID = varchar, max length 3  Name = varchar, max length 30  UnitsAvailable = int  UnitsOnOrder = int  Price = float |
| Lab12Orders | 3 | OID = varchar, max length 3  CID = varchar, max length 3  Date = date |
| Lab12OrderDetails | 3 | OID = varchar, max length 3  PID = varchar, max length 3  Quantity = int |

After you have specified the format of each table, you need to import the data from the appropriate CSV file. At the top of the phpMyAdmin window, there is an Import button. Here, you can specify the CSV file. Don’t forget that in the section labeled “Format of imported file” you need to change the “Fields terminated by” to a comma instead of a semicolon. When you click the “Go” button, you should see a response on the next screen saying something like this: “Import has been successfully finished, 31 queries executed.” This number of “queries” actually refers to the number of records in your table.

After you have uploaded your 4 files, you should verify that the data in each table looks correct. Use the browse button to see the records. How many records does each table have?

Lab12Customer = \_\_\_\_\_\_

Lab12Product = \_\_\_\_\_\_

Lab12Orders = \_\_\_\_\_\_

Lab12OrderDetails = \_\_\_\_\_\_

1. Now let’s have some fun with simple 1-table queries.

Select \* from Lab12Customer where City = “Miami”;

Select First, Last, Phone from Lab12Customer where Phone like “305%”;

Select Name, Price from Lab12Product where Name like “%Monitor%”;

Now, it’s your turn. Write select queries that answer these questions. (You don’t need to write the answers to the questions.) Use phpMyAdmin to check your answers. Be sure to include enough fields in your query so that the result being returned to you is meaningful. But don’t include unnecessary fields that have nothing to do with the question.

1. Michael Ware is one of our customers. What is his telephone number?
2. Which products are priced at over $1000?
3. Does the store sell any laptops? If so, how much do they cost and how many are in stock?
4. How many units of product 14 were included in order 6?
5. Now, some 2-table queries. We’re interested in seeing the orders placed by Benjamin Lee. Note that the customer and order tables have a relationship because they share the same CID field. We can try the following query.

Select Lab12Customer.First, Lab12Customer.Last, Lab12Orders.OID, Lab12Orders.Date

from Lab12Customer, Lab12Orders

whereLab12Customer.CID = Lab12Orders.CID

and Lab12Customer.Last = “Lee”;

How many records are returned by the query?

Note that the above query had 2 criteria. We insisted that the CID’s match and that the last name of the customer be Lee. What records would get returned if we omit either of these criteria? Write your answers below:

If we want the CID’s to match, but we omit the Lee criterion, the effect is…..

If we want the last name Lee but we don’t check the CID’s, the effect is…….

Here is another example: Let’s suppose we want an itemized listing of what products were included in order #1. We’d like to print out the name of each product, along with its price, and how many of each product were ordered. The information we need is contained in two tables: Lab12Product and Lab12OrderDetails. These two tables have a one-to-many relationship because they share the PID field. Therefore, when we write our query, we need to insist that the corresponding PID fields match. The other criterion we need is to restrict our search to just order #1. As in the previous example, both criteria are necessary. Omitting one would produce too many records.

Here is the resulting query. Enter this query in SQL and run it to verify it works correctly.

Select Lab12Product.Name, Lab12Product.Price, Lab12OrderDetails.Quantity

from Lab12Product, Lab12OrderDetails

where Lab12Product.PID = Lab12OrderDetails.PID

and Lab12OrderDetails.OID = “O01”;

Okay, let’s practice. How would you write queries that answer these questions? Try to make the result of the query as informative as possible, without printing irrelevant information.

1. For order number 3, which customer placed that order and what was the date?
2. Who from Miami placed an order on April 25, 1999?
3. How would you modify the above query if we wanted the names of all customers from Miami and Coral Gables who bought something on April 25, 1999?
4. What is the name of each product included in order number 4?
5. Which orders (i.e. which order numbers) include a joystick?
6. We can write even more powerful queries if we collect information from several tables. Consider this query:

Select Lab12Orders.OID, Lab12Product.Name

from Lab12Orders, Lab12OrderDetails, Lab12Product

where Lab12Orders.OID = Lab12OrderDetails.OID

and Lab12OrderDetails.PID = Lab12Product.PID

and Lab12Orders.OID = “O01”;

Basically, this query says that we want the names of all the products from order #1. Sounds simple, so why must the query include so much detail? As before, we need to tell SQL that our tables have a relationship we want to enforce. We are making use of 3 tables in the query: the products, orders and order details. Note that we are “using” the Lab12OrderDetails table even though we are not printing out anything from it. We need this table because it is “between” the other two tables. The product and order tables do not have a direct relationship between each other, but each has a 1-many relationship (i.e. sharing a field) with order details.

Next, we can ask questions about customers buying certain products. Writing such a query would entail using 4 tables. To travel from the customer to the product table means we have to first travel through orders, and then order details:

Lab12Customer 🡪 Lab12Orders 🡪 Lab12OrderDetails 🡪 Lab12Product

And between each pair of tables along the way, there is a one-to-many relationship due to a shared column. For review, answer these questions:

Which field is shared by customers and orders?

Which field is shared by orders and order details?

Which field is shared by order details and products?

When we write a query involving customers and products, we need to tell SQL that these respective fields must match, or else the result of our query won’t make sense.

Now we’re ready to try some queries using 4 tables. Let’s start by asking for the names of customers who ordered an Ink Jet Printer. It would be minimally sufficient to just ask for the customer’s first and last name, but for clarity (and to make it easier to check correctness), the query will also give the order number and the product name.

Select Lab12Customer.First, Lab12Customer.Last, Lab12Orders.OID, Lab12Product.Name

from Lab12Customer, Lab12Orders, Lab12OrderDetails, Lab12Product

where Lab12Customer.CID = Lab12Orders.CID

and Lab12Orders.OID = Lab12OrderDetails.OID

and Lab12OrderDetails.PID = Lab12Product.PID

and Lab12Product.Name = "Ink Jet Printer";

Enter and run this query to verify that the answer is correct. You should be able to check the result against the data on your handout.

Now, let’s experiment with other multiple table queries. Adapt the previous example to write a queries to answer these questions

1. Benjamin Lee is one of our customers (and the only one with last name Lee). What products has he ever bought from us? Show the product names and the dates each was bought.
2. Which customers have bought products priced over $2,000? Print out their names and the cities where they live.
3. Let’s print a comprehensive summary of all items ever ordered. We’d like something as thorough as the Lab12OrderDetails table, but with the following information: order number, name of customer, date of sale, product name, price, quantity. (Note that if a field resides in two different tables, it does not matter which one you ask SQL to display. For example, at the beginning of the SQL statement you can “select” Lab12Orders.OID or Lab12OrderDetails.OID. In your criteria you are guaranteeing they are the same anyway).
4. Finally, let’s add the 5th table of this database: the Lab12Employee table.
   1. Use the data on your handout to create this table. Decide on names and types of fields that you need. Then, enter the data into the table.
   2. What is an example of a query that can make use of several tables, including the new employee table? The most interesting example would be a query making use of all five tables.