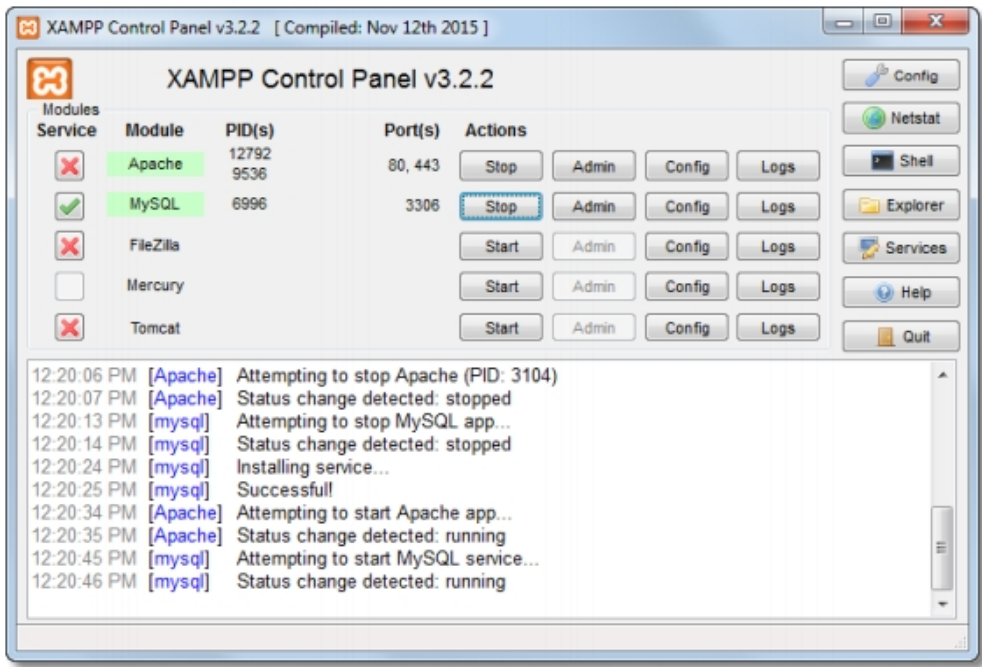
Week 1 – Lab exercises and set up

1. Editor download and installation:
   1. If you use Windows download and install Notepad++ editor to enter and edit HTML, CSS, and PHP files: <https://notepad-plus-plus.org/downloads/>
   2. If you use Mac OS X download and install BBEdit as your editor: <https://www.barebones.com/products/bbedit/>
2. Installing XAMPP software package (includes Apache, MySQL, and PHP): <https://www.apachefriends.org/index.html>. you need to make sure that both Apache and MySQL are running before you test your applications. To do that, you can click on the Start buttons in the XAMPP Control Panel as shown in figure below. If you want to start the Apache and MySQL servers each time your computer starts, you can use the check box to its left to install it as a service. In this figure, for example, the MySQL server is installed as a service. As a result, it will start every time this computer starts.



**How to start the XAMPP control panel**

* On a Windows system, select the XAMPP Control Panel item from the Windows Start menu or double-click on the XAMPP icon on your desktop.
* On a Mac OS X system, open the Applications folder, open the XAMPP folder, and double-click the manager-osx application.
* On a Linux system, open a terminal, use the cd command to change to the opt/lampp directory, and use the sudo command to execute the manager-linux-x64.run or manager-linux-x86.run file.

**How to start and stop Apache or MySQL**

* Click on its Start or Stop button.
* To start Apache or MySQL automatically when your computer starts, please refer to the appendixes.

**How to deploy a PHP application**

When you deploy an application, you make it accessible from a browser. To do that on your own computer or on a local server when running Windows, you copy all of the directories and files for the application to the \xampp\htdocs directory for the Apache server because that’s where Apache looks for applications. See figure below where the top-level directory for a guitar store application has been copied to the htdocs directory..



In this example, only four subdirectories are shown, including a directory for CSS files and a directory for storing the images that are required by the application. In practice, though, a large application is likely to contain many subdirectories at several different levels. Note here, that the htdocs directory is called the document root directory, while the guitar\_store directory is called the application root directory. This just means that htdocs is the root directory for all applications, while guitar\_store is the root directory for one application.

To deploy an application on an Internet server, you copy the application directories and files from the local server to your root directory on the Internet server. To do that, you can use an FTP program, which uses File Transfer Protocol to upload the files from the local server to the Internet server. To make this work, of course, the Internet server must be running Apache and a PHP interpreter.

In practice, PHP applications are usually developed and tested on a local server before they are uploaded to an Internet server. To make that manageable, the directory structure on the local server is exactly the same as (it “mirrors”) the directory structures on the Internet server. That also makes it easier to maintain and enhance a web application later on.

**How to deploy a PHP application on a local server**

* Copy all of the directories and files for an application to the \xampp\htdocs direc-tory on the server. This is where Apache looks for PHP applications.

**How to deploy the downloadable applications on a local server**

* Copy the book\_apps and ex\_starts directories and all their contents to the \xampp\htdocs directory on the server.

**How to deploy an application on an Internet server**

* Use an FTP (File Transfer Protocol) program to upload the tested directories and files to the htdocs directory of the Apache web server.

**How to test and debug a PHP page**

When you finish editing a file for a web page, you need to save it. Then, you can test the web page or application by running it in a browser. See figure below:

***URLs for requesting applications from a local web server***

**A request for the default page in an application directory** http://localhost/book\_apps/ch01\_product\_discount/

**A request for a directory that doesn’t contain a default page**

http://localhost/book\_apps/

When the page is displayed in your web browser, you can test it by entering

any required data and performing the actions indicated by the controls on the page.

**How to test a PHP page for the first time**

1. Make sure the Apache and MySQL servers are running.
2. Start a web browser and enter the URL for the application as shown in the last figure.
3. Test the page by entering both valid and invalid data, clicking on all links, and so on.

**How to retest a PHP page after you change the source code**

* Click the Reload or Refresh button in the browser. Then, test the page again.

***Complete the following exercises:***

**Open the files for the Product Discount application**

1. 1. Make sure the Apache server is running. Then, start your browser and run the application in this directory:

**xampp\htdocs\ex\_starts\ch02\_ex1**

To do that, you can use the index of exercise starts that you bookmarked in exercise 1-2 or you can use this URL:

**http://localhost/ex\_starts/ch02\_ex1**

1. Enter valid data in the three text boxes, and click the Calculate Discount button. Note that the resulting web page indicates that it is under construction.
2. Use your IDE or text editor to open the display\_discount.php file for this application. It is in the same folder as the index.html file. Then, note that this file doesn’t contain the PHP statements that this application needs.
3. Use your IDE or text editor to open the index.html file for this application. Then, note the name attributes for the three text boxes.

**Add the PHP statements**

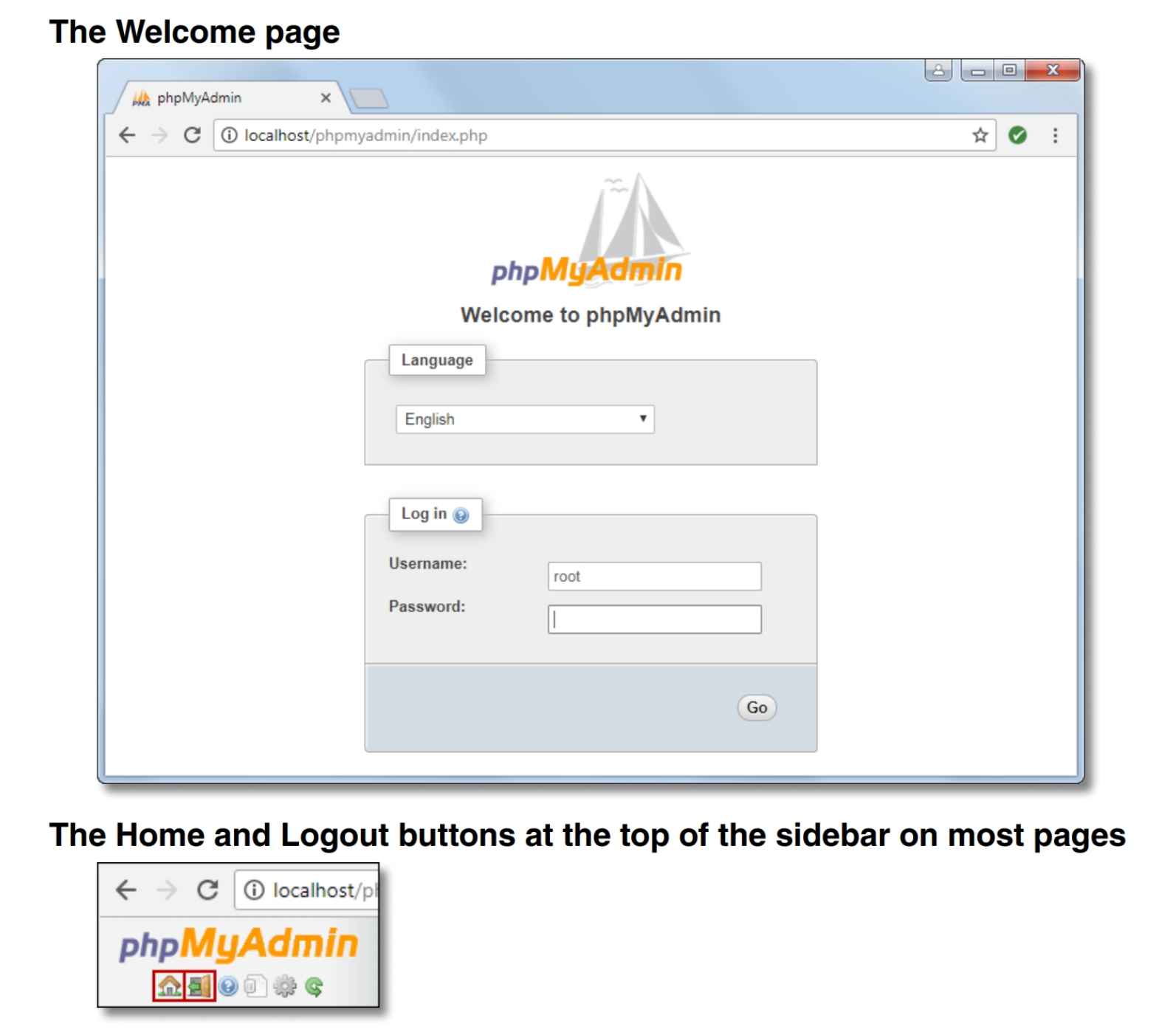
1. Switch to the display\_discount.php file and add statements that use the $\_POST array to get the product description, list price, and discount percent from the form. Then, modify the first three echo statements so they display this data, save the file, and run the application again. This should display unformatted versions of the data that you enter.
2. Switch to the display\_discount.php file and change the statements that use the $\_POST array to use the filter\_input() function instead. Save the file and run the application to make sure it still works as expected.
3. Switch to the display\_discount.php file and add statements that calculate the discount amount and discount price. Then, modify the last two echo state-ments so they display this data, save the file, and run the application. This time, the application should display unformatted versions of the calculated data.
4. Switch to the display\_discount.php file and add statements that format the numeric variables with the currency and percentage formats. Then, modify the last four echo statements so they display the formatted data, save the file, and run the application. Now, the application should display the formatted data.
5. 9. Keep the application running and navigate to the index.html page if you’re not already there. In the Product Description textbox, enter “<b>Guitar</b>”, and in the Discount Percent textbox, enter “<i>10</i>”. Click the Calculate Discount button, and notice that the Product Description value is displayed in bold, and the Standard Discount value is displayed in italics.
6. Switch to the display\_discount.php file and add the htmlspecialchars() function to the echo statements for Product Description, List Price, and Standard Discount. Then save the file, run the application, and repeat step 9. Notice that this time, the HTML tags are displayed as HTML entities rather than being implemented by the browser.
7. Switch to the display\_discount.php file and fix the heading for the display\_discount.php file so it says “Product Discount Calculator”. Then, test this change, and close the files.

**Use phpMyAdmin with a database**

This exercise will give you a chance to use phpMyAdmin to review the data-bases that are used with this book and to test the types of SQL statements that you will use in your PHP applications.

If necessary, set a password for the root user

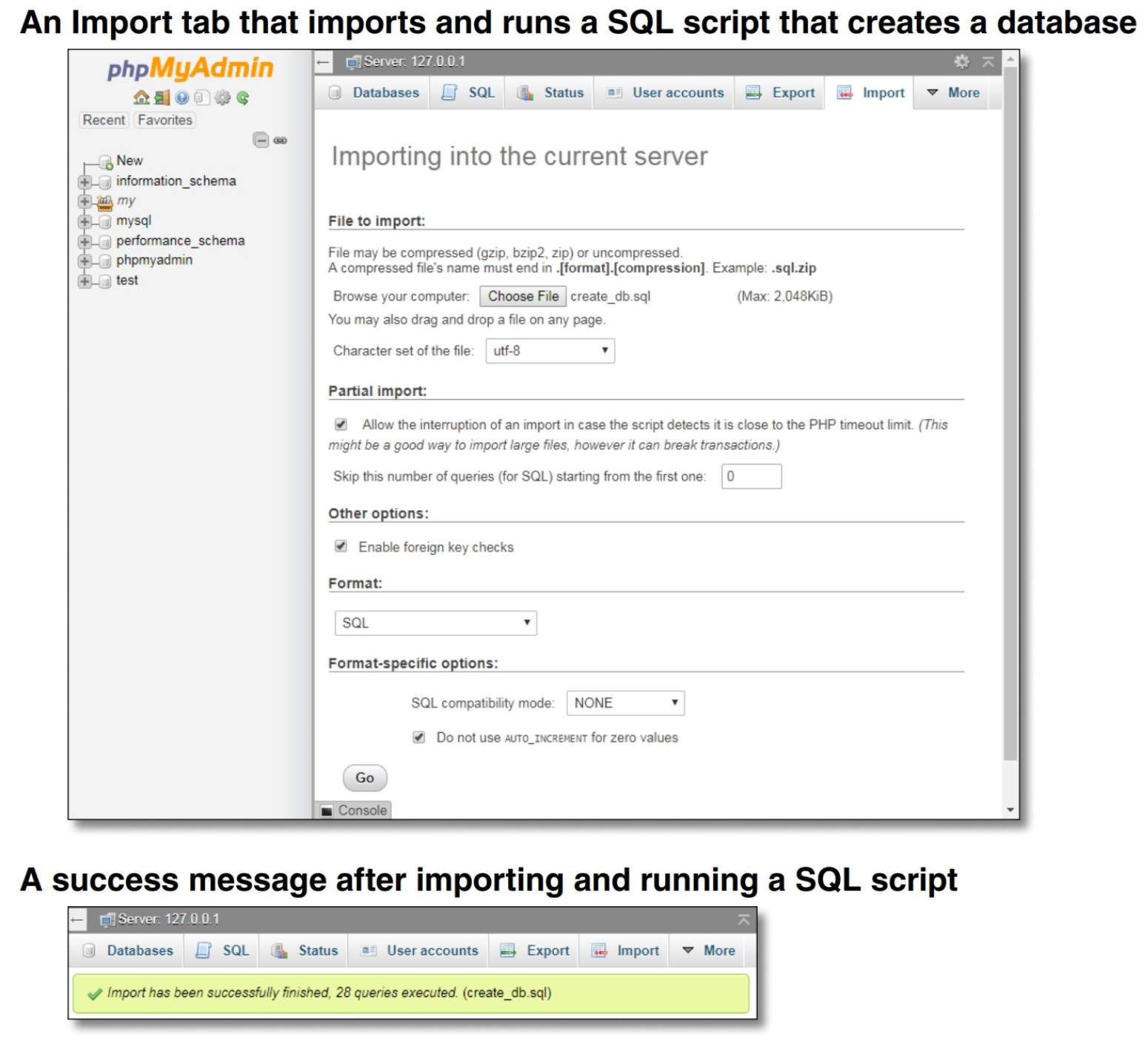
1. Start Apache, MySQL, and phpMyAdmin as shown in figure 3-9. Then, log in as the root user.



1. If you haven’t already set a password for the root user, use phpMyAdmin to do that. If you’re only using MySQL for working with the applications in this book, use a password like “sesame” so it’s easy to remember. Otherwise, use a more secure password.
2. Click the “Log out” button (the Exit sign) to log out. This should take you back to the Welcome page. Then, log in as the root user with the new password. That will take you to the Home page.

**Run the script for creating the book databases**

1. On the Home page, review the list of databases that are available in the sidebar. Then, click the Databases tab to see the same databases. Note that these databases include the databases that MySQL and phpMyAdmin use to manage their own operations. Are the my\_guitar\_shop1 and my\_guitar\_shop2 databases both available?
2. Use the procedure in figure 3-10 to import and run the script that creates the databases for this book. This will create or recreate the two databases for this book. As a result, the my\_guitar\_shop1 and my\_guitar\_shop2 databases will both be shown in the sidebar and in the Databases tab.

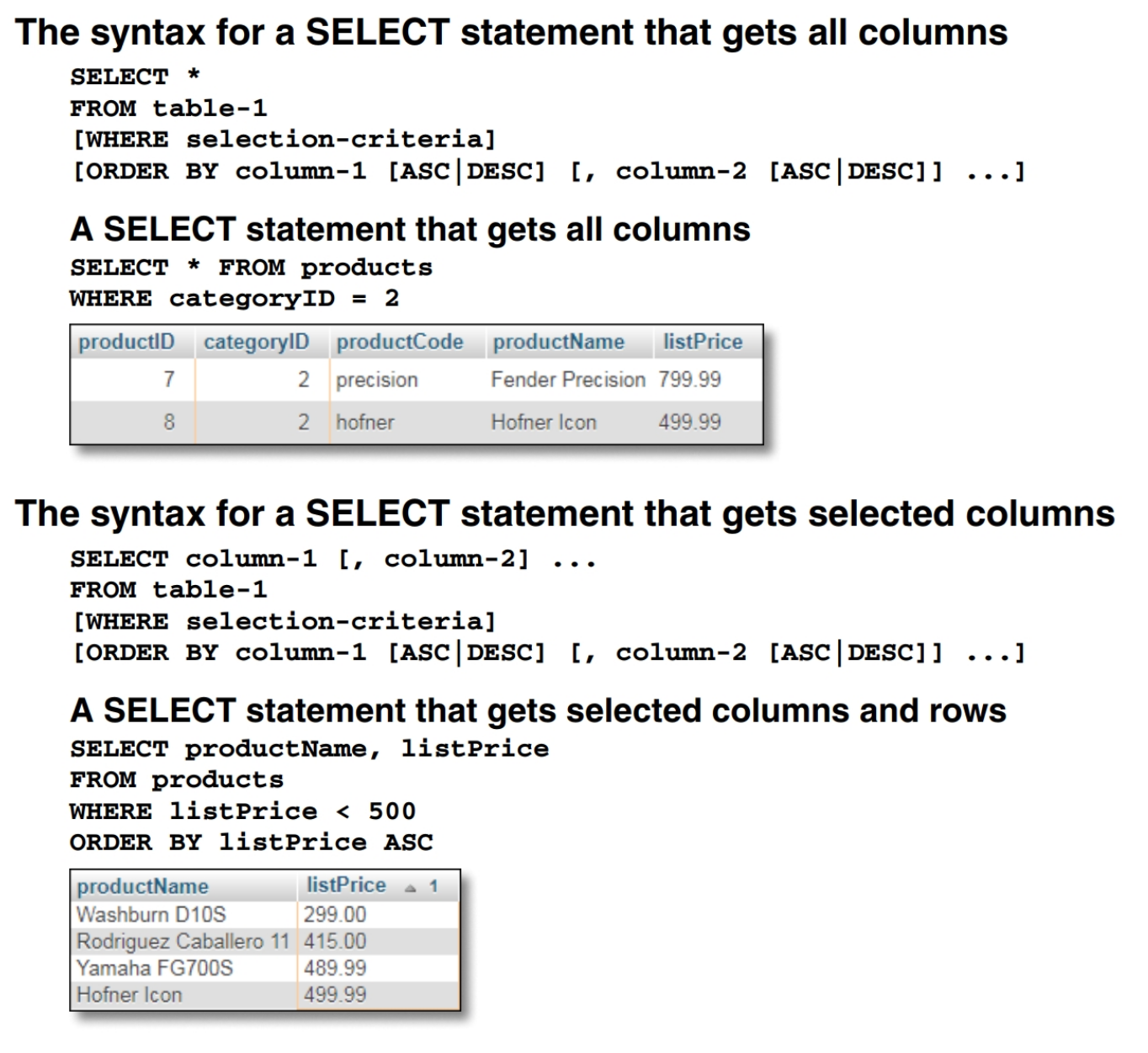


**Review the my\_guitar\_shop1 database**

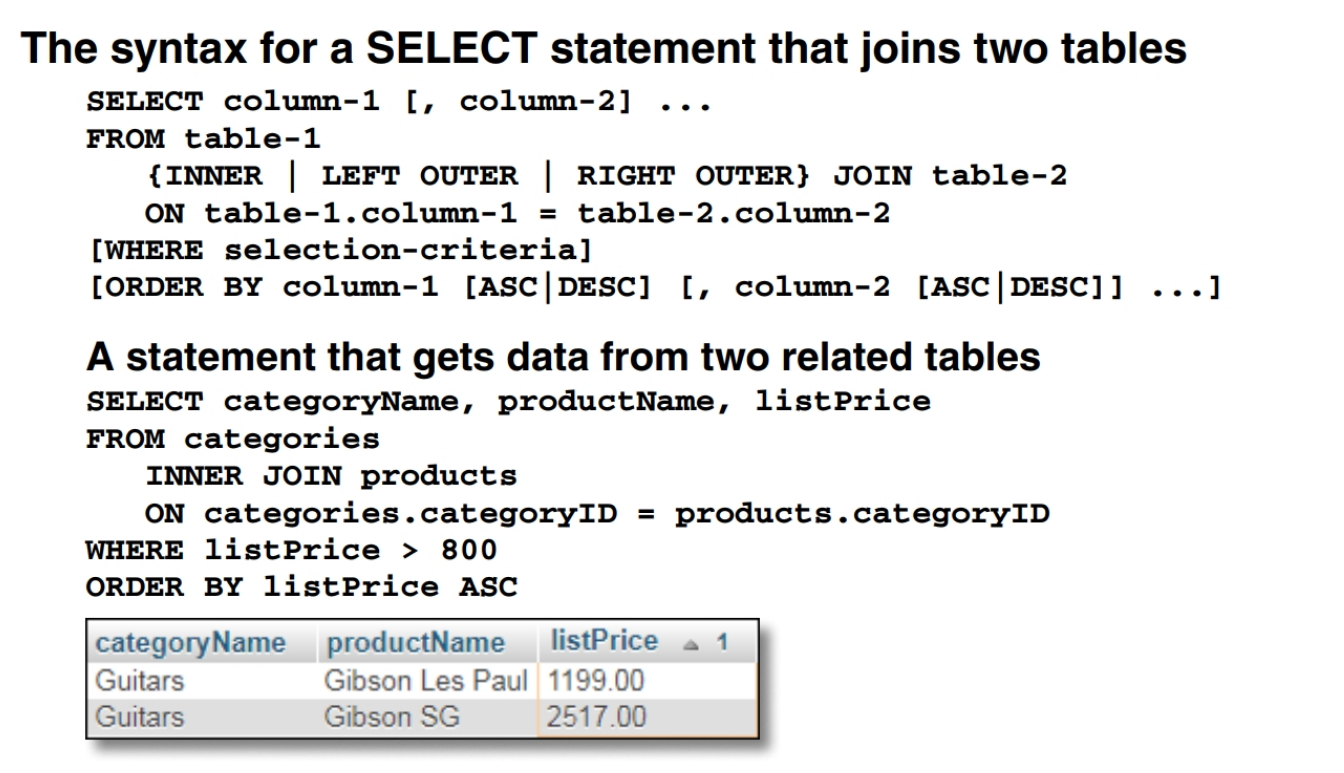
1. Select the my\_guitar\_shop1 database to display the tables for this database, and click the Browse button to view the data for the products table.
2. Click the Structure tab to view the column definitions for the products table. Note that none of the columns allows nulls or provides default values.

**Run SQL statements against the my\_guitar\_shop1 database**

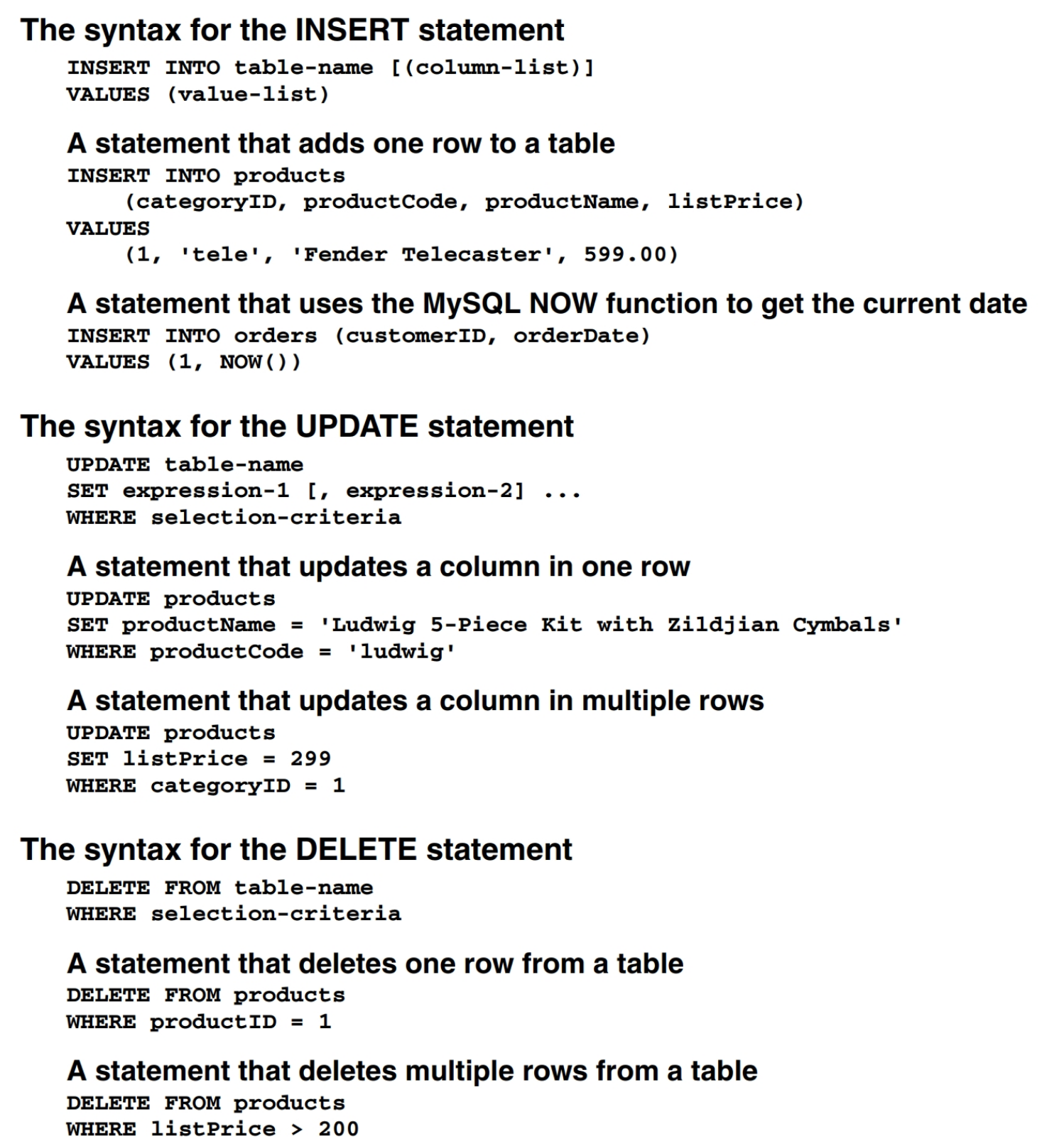
1. Use the SQL tab to run the first query in figure 3-4. Then, run the second query.



1. Run the query in figure 3-5. Then, modify the list price value in the query so it only selects products with a price that’s less than 400, and run the query again.



1. Run the first query in figure 3-6 to add a row to the products table. Then, browse the products table to view the new row. Last, run a DELETE statement to delete the new row.



1. Continue to experiment until you’re sure that you know how to code the SQL queries that your PHP applications will use.

**Log in as a different user and check that user’s privileges**

1. Log out of phpMyAdmin, and log back in as mgs\_tester with pa55word as the password. This user was created by the SQL script that you ran in step 5 of this exercise.
2. Use the SQL tab to run this SELECT statement:

**SELECT \* FROM categories**

This statement should be refused because mgs\_tester can only run statements against the products table.

**Experiment**

1. Continue to experiment until you’re confident that you understand the use of phpMyAdmin and the types of SQL statements that you’ll use in your PHP applications

**Enhance the Product Manager application**

This exercise has you enhance the Product Manager application by adding a page that lets you add and delete categories.

**Test the Product Manager application**

1. Start the Chrome browser and run the application in the ex\_starts\ch04\_ex1 directory. To do that, you can use this URL:

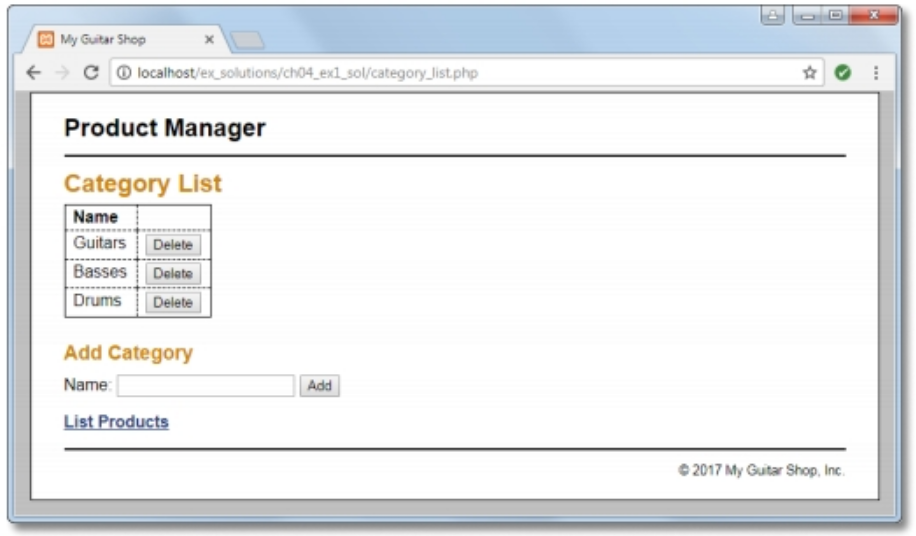
<http://localhost/ex_starts/ch04_ex1/>

This should display the products for the first category in the database named my\_guitar\_shop1.

1. View the products in each of the categories.
2. Add a new product to the database. When you add this product, make sure to enter valid values for a product. Then, delete the product you just added from the database.
3. Click on the List Categories link at the bottom of the page. Note that this link leads to a page that’s under construction. However, the link back to the Product List page does work.

**Add a Category List page**

In the rest of this exercise, you’ll add a page that looks like this:



1. Open the category\_list.php file that’s in the directory for this application. It contains some of the headings and a link back to the Product List page.
2. In the category\_list.php file, write the code that creates the category table shown above with all of the category names in the first column and Delete buttons in the second column, similar to how the index.php file works. Then, test that this table is displayed correctly.
3. In the category\_list.php file, write the code that lets the user add a category to the database. This code should consist of a form that accepts the name for a new category followed by a Submit button that displays “Add”, similar to how the add\_product\_form.php file works. Then, test that this form is displayed correctly.
4. Create an add\_category.php file that adds a category to the database and a delete\_category.php file that deletes a category from the database. These files should display the Category List page after they add or delete a category, similar to how the add\_product.php and delete\_product.php files work.
5. Test the application by adding two categories. Then, navigate to the Product List page and note that the list of categories includes the new categories. Next, navigate to the Add Product page and note that the drop-down list includes the new categories.
6. Test the application by deleting the categories that you just added. However, don’t delete any of the existing categories because that will lead to products without categories. If necessary, though, you can restore the database by running the create\_db.sql script again as described in the appendixes.