Lab Homework 6: SQL Search Strategies

In this lab you'll practice two methods for implementing search for a website backed by a database. You'll also practice inserting a record into the database.

1. Learning Objectives

- Implement different search strategies against a database.
- Practice *filtering input, escaping output* for *values* in SQL queries.
- Develop a better understanding of how PHP and SQL work together.
- Exposure to an example that you can use as a reference for Project 2.

2. Deadline

Lab Homework	Deadline	Slip Days	Credit	Solution
All Parts	Sun 3/8, 11:59pm	Max: 2 days	20 points (completion)	Provided

3. Instructions

1. Clone your lab repository.

   Clone the following URL:
   `git@github.coecis.cornell.edu:info2300-2020sp/YOUR_GITHUB_USERNAME-lab06.git`
   Replace **YOUR_GITHUB_USERNAME** in the URL with **your Cornell GitHub username**.

2. **Sign the attendance sheet.**

   It is your responsibility to sign the attendance sheet before you leave. If it wasn't handed to you, **ask to sign it**. No signature on the attendance sheet will result in an immediate 0 for this lab's attendance grade. Forgetting to sign the attendance sheet will not be considered a valid excuse to have your lab attendance corrected.

3. Work together.

   Work with your peers to complete this lab. Talk aloud, discuss, troubleshoot problems. **You are encouraged to work together so long as you do your own work and you don't give away answers.**

4. Submit.

   When you're finished, follow the instructions in **submit.md** to submit your assignment.

4. A Word About Project 2

This lab is designed to be similar to Project 2. This is intentional. Use this lab to help you complete your Project 2. Just remember, you aren't allowed to copy and paste code. **Write all of your own code.** You will learn more if you do.

Part I: Working with Existing Code

Learning to work with existing code can be challenging. It's also a critical skill to learn. Most of the code that you will work with outside of school won't be code that you wrote yourself.

Objective: For this part, review **shoes.php** with your peers. Carefully study the code to learn what it does before you start making changes to it.

Objective: Once your team has reviewed the code, join in the class discussion about what this code does and how it does it.

Part II: Search a Database

Objective: In Lab 5 you learned how to write database queries for searching records using SQL's `LIKE`. In this lab, we'll continue building the **2300 Shoe Review** you started in Lab 5. However, we'll focus on the 2 different methods for implementing *search* functionality. First, we'll search over just one field. Lastly, we'll search across all fields in a table.

Hint: Practice your queries in *DB Browser for SQLite* before you implement them in PHP. It easier to see if you have an incorrect SQL query if you test it **outside** of PHP. Otherwise you often can't tell if your SQL query is incorrect or your PHP PDO code is incorrect.

1. Search *only* One Field

Take a look at the search form for shoes.php in your browser. This search form is designed to support two types of searches: 1) search over just one field (e.g. "Search Reviewers") and 2) search across all fields at the same time ("Search Everything"). In this part, you'll implement searching across only one a user selected field:

Search Dropdown	Database Field
"Search Reviewers"	`reviewer`
"Search Ratings"	`rating`
"Search Products"	`product_name`
"Search Comments"	`comment`

1.1. Plan Your Query

Objective: With your peers plan a query that searches for `$search` across the user specified field, `$search_field`. Test out your queries in DB Browser for SQLite.

Tip: You will need to use a wild card search: `... WHERE (field LIKE '%search_term%')`

Hint: To test that your wild card search is working, search for "Fly" or "Air" under the Product category.

1.2. Implement Your Query

Objective: Look for the **TODO**s in *shoes.php*. **Implement your planned query and test it for all fields.**

Gotcha I: Recall from class that parameter markers only work for *values*. You cannot use parameter markers for table or field names.

This means if you need to programmatically set a field name or table name, you **must** filter that value **very** well. Conveniently, in the above section we already filter this input for you by checking whether `$search_field` is one of 4 valid field names. Because of this careful filtering, you can now just use PHP's concatenation operator (`.`) to use `$search_field` directly in a query. There's no way for a hacker to break this since there's only 4 valid options and we don't permit *anything* else.

Gotcha II: Since `$search` is a value, you can just use a parameter marker to escape it (`:search`) but this is tricky for wild card searches.

You will need to use the SQL concatenation operator (`||`) to perform a wild card search with a parameter marker: `'%' || :search || '%'`. Note: this is **not** valid SQL with parameter markers `%:search%`. Remember that the concatenation operator "glues" strings together. So really you're just gluing `%`, `:search`, and `%` together to make: `%:search%`, but you're doing it in such a way that PHP will understand your parameter marker!

2. Search Across *All* Fields at Once

Next, you'll implement search across **many** fields at once. So when a users selects **Search Everything** you will search across the `reviewer`, `rating`, `product_name`, and `comment` fields **all at the same time** using **one** SQL query.

Search Dropdown	Database Fields
"Search **Everything**"	`reviewer`, `rating`, `product_name`, `comment`

2.1. SQL Logical Operators

Just like we use logical operators to evaluate multiple conditional expressions in PHP, we can do the same in PHP. SQL *generally* supports these common logical operators:

Operator	Type	Usage	Example
`AND`	Evaluates to TRUE if both conditional expressions are TRUE	(*cond. expr*) AND (_cond. expr)	`... WHERE (position = 1) AND (age > 55)`
`OR`	Evaluates to TRUE if **one** conditional express is TRUE (or both)	(*cond. expr*) AND (_cond. expr)	`... WHERE (age < 18 ) OR (age > 55)`

To learn more you can check-out the SQLite reference documentation: https://sqlite.org/lang_expr.html. Unfortunately, it's not very understandable if you're new to SQLite. Because SQL is mostly standardized you can often review MySQL documentation: https://dev.mysql.com/doc/refman/5.7/en/logical-operators.html. However, please note SQLite does not use the `&&` and `||` as logical operators (`||` is the concatenation operator in SQLite.) **Lastly, you can review third-party SQLite documentation: https://www.tutorialspoint.com/sqlite/sqlite_and_or_clauses.htm.**

2.2. Plan Your Query

Objective: Plan one query that searches for `$search` across all fields at the same time. Test out your queries in DB Browser for SQLite.

Tip: Use the reference documentation for logical operators.

Hint: To test that your wild card search is working across multiple fields, search for *rare* characters like *z*. (There is a Z in Air**Z**oom and si**z**e.)

2.3. Implement Your Query

Objective: Look for the **TODO**s in *shoes.php*. **Implement your planned query and test it against data in *all* fields.**

Part III: Inserting a Review into the Database

Objective: In this part, you'll write the SQL query to insert a review into the shoe database.

1. Plan Your Query

Constructing the SQL Query for adding an entry uses the SQL `INSERT INTO` statement:

```
INSERT INTO table (field1, field2, ...) VALUES (value1, value2, ...);
```

For example, if we were to reuse the employee database for Willa Bend's Popcorn Stand from Lab 5. Adding a new manager, Mindy Kaling (mk497) would be done with the following query:

```
INSERT INTO employees (employee_id, first_name, last_name, position) VALUES ('mk497', 'Mindy',
'Kaling', 2);
```

Observe that we did not include the surrogate key, `id`. `id` is a primary key with the `AUTOINCREMENT` constraint set, this means the database will automatically set value for `id` for us!

Objective: Plan a query that inserts a review into the table.

2. Implement Your Query

Objective: Look for the **TODO**s in *shoes.php*. With the above plan as your guide, implement adding a review to the database. You will need to write a SQL query to insert the review into the database.

3. Test Adding Reviews

Objective: Thoroughly test that you're adding the reviews to the database.

Part IV: Project 2 Work-time

When you're finished with this lab, work on your Project 2. Feel free to ask your peers to review your code or your design. Feel free to ask for help. etc.

Reminder: All code for Project 2 must be **your own work**. You may not copy and paste any part of this lab into your Project. You may use it as a reference.

Contributors

The following individuals made contributions to this assignment.

- Kyle Harms
- Sharon Jeong