

Week 11: SQL Part 1

Agenda

- Overview
 - What's Due?
- Introduction to SQL (Structured Query Language)
 - What is SQL?
 - Structure of a Database
 - SQL Syntax
 - Setting Up a Database on 000webhost
 - Combining SQL and PHP
- Database Challenge
- Database Structure (Two Posts)
- Next Week

Overview

Last week, we went over making our pages more secure and easier to maintain. This included adding PHP to our action attribute and using another method for checking input from an external page. We also covered AJAX and how it could be used with PHP. This week, we will learn about SQL and how we can use it with PHP to create tables and manipulate them.

PHP is a server-side language and our UA Albany server spaces don't allow for us to use those types of pages. 000webhost will be the platform we use for the rest of the semester. You should have created a 000webhost account to complete last week's assignment. If you haven't done that already, please make sure to do so for this week.

Make sure to use a unique password for this website as it has a history of data breaches. The goal is to use it for testing basic PHP features with our web pages. You are free to use other platforms if you'd like, they should have a similar interface.

What's Due

- Database Challenge (Extra Credit Available)
- Database Structure (Two Posts)

Introduction to SQL (Structured Query Language)

In the following sections, I will briefly go over some concepts related to SQL and how it can be used on a web page. Specifically, we'll cover databases, SQL syntax, and how to tie them in to PHP so we can add more complicated features to our pages.

What is SQL?

As I'm sure you're aware, humans create a lot of data every day and it is only increasing with time. Raw data on its own is mostly meaningless and is not very useful for creating applications or other tools on the web. **Databases** provide a structure for this data, so it is easier to access and read. They are collections of related or unrelated tables which combine to give meaning to data and allow a website to contain more complex features.

Most databases are manipulated with some sort of **Database Management Systems (DBMS)** such as MySQL, Oracle, SQL Server, Microsoft Access, etc. Each of them has their strengths

and weaknesses, but for this course, we will work with MySQL. It is free, open source, and documented very well, so it is good for beginners learning to work with backend technologies.

SQL pronounced “sequel,” stands for **Structured Query Language** and is a language used in DBMS to manipulate databases. You run “queries” on the database, and they produce results which may include selecting data, inserting data, updating existing data, or even deleting data. It has its own set of rules regarding syntax/usage just like any other language. Before we get into the syntax, there are a few database concepts to learn which will help with creating SQL statements.

Structure of a Database

Every database has several components which work together for the entire system.

Understanding these components will help you with the creation of queries to get or push data into databases. Use the **Customers Table** and **Pets Table** below to understand the definitions provided. These are tables that store information about users and their dogs.

Customers Table

UserID	Name	Age	Email
1	Chris Velez	30	cvelez@albany.edu
2	John Doe	23	jodoe@mail.com
3	Jane Doe	25	Jadobe@mail.com

Pets Table

PetID	PetName	PetType	Age	OwnerID
1	Onyx	Cat	8	1
2	Smokey	Cat	10	1
3	Spot	Dog	5	3
4	Spot	Dog	7	2

Tables - These are collections of data arranged into columns/fields and rows/records.

Column/Field - Any vertical area in a table. In the Customers table above, UserID, Name, Age, and Email are columns. PetId, PetName, etc. are also columns in the Pets table.

Row/Record – Any horizontal area in a table. 1, Chris Velez, 30, and cvelez@albany.edu are all values in the same row of the Customers table.

Primary Key – This is a column used to uniquely identify a single row/record. In our Pets table, we have pets with the name “Spot” in multiple rows. How would we retrieve information about one of them instead of both? We could use PetID in our Pets table as a primary key since it has a unique value. One dog named Spot has a PetID of 3 and the other as a PetID of 4. The same thing is applicable to our Customers table; we can use one single column to identify any row individually and, in this case, it would be UserID. In other tables we might use a single field or combinations of fields.

Foreign Key – This is a column or columns used to link tables together. These columns typically contain values that are primary keys in another table. In our Pets table, we have a column called OwnerID. This OwnerID column contains the same values as the UserID column in our Customers table. The pets in the Pets table would then be linked to their owners by that foreign key value. Based on the tables, you can see that Chris Velez owns two cats, and the other two users each have one dog.

Data Types – There are many types of data for SQL but some of the most common ones you will encounter are Numeric (int, bit, decimal, etc.), Date (datetime, date, time, etc.), and Character (char, varchar, text). The various subtypes might seem similar, but their functions differ across DBMS platforms. In our Customers table, UserID and Age might be int while name and email are varchar or text.

Constraints – These are used to specify rules for data in a table. They limit the types of data that can be inserted and help to maintain accuracy/integrity of data in the table. Some common constraints might be NOT NULL, UNIQUE, PRIMARY KEY, and FOREIGN KEY, and data types. Not null means the column cannot have empty data. Unique means it must have a unique value. Primary and Foreign key tells the database if that column will serve as a primary or foreign key for the table. These constraints play a big role in table creation, and you will see them in action in the SQL Syntax **CREATE** section.

SQL Syntax

Just like any other language, SQL has a set of “rules” or syntax that it follows. SQL statements typically start with a keyword such as SELECT, INSERT, UPDATE, etc. and end with a semicolon (;). The keywords are case-insensitive meaning they can be SELECT, SeLeCt, etc. and it will be evaluated the same. However, it is good practice to keep them all uppercased.

SQL has its own set of operators which sometimes function like the ones found in other languages such as JavaScript or PHP. I have listed a few of them below with the explanations in parenthesis.

* (This operator means “all” like when we use it in CSS)

Numeric: +, -, *, /, %

Comparison:

= (equal to)

!= (not equal to)

<> (not equal to)

> (greater than)

< (less than)

>= (greater than or equal to)

<= (less than or equal to)

!> (not greater than)

!< (not less than)

Logical: ALL, AND, BETWEEN, EXISTS, NOT, LIKE, OR, IS NULL

When using these keywords with column/row data and various operators, we can create basic or complex statements that will manipulate our database. A full statement that is executed is known

as an SQL command. Many commands exist but we are going to review common ones often referred to as CRUD operations (Create, Read, Update, Delete).

For the purposes of this class:

Create is equivalent to INSERT/CREATE (creating new databases, tables, rows of data, etc.)

Read is equivalent to SELECT (selecting/reading data from database)

Update is equivalent to UPDATE (update existing data)

Delete is equivalent to DELETE (delete databases, tables, rows, etc.)

To use any of these commands, you need to specify pieces of information such as the table name, column name, or even row data. In the examples below, I demonstrate the syntax for basic SQL commands followed by explanations. Anything inside of square brackets is a placeholder value to indicate formatting for the query/command. Parentheses contain the output in terms of rows/columns retrieved.

SELECT (Based on One Condition)

```
SELECT [column_name, column_name2]
FROM [table_name, table_name2]
WHERE [some condition];
```

SELECT (Based on Multiple Conditions)

```
SELECT [column_name, column_name2]
FROM [table_name, table_name2]
WHERE [some condition] AND [some other condition];
```

In the code above, I broke up the query into multiple lines for readability. This is what you might type if you did a query directly inside of a DBMS. From this point onward, I will be writing my queries in one line.

```
SELECT [column_name, column_name2] FROM [table_name,
table_name2] WHERE [some condition];
```

This tells the database to select some column from a specific table based on a condition. We could name multiple columns or tables by separating them with commas. If we had data being pulled in from multiple tables, we would have to specify each one in the FROM part. The WHERE part of the statement is sort of like an if statement. We want to select the values from the table **only if** they match the conditions we set. You can specify multiple conditions by using the AND keyword. Each AND must be followed by a valid condition.

If you wanted to query one column from two tables but the column had the same name in both tables, the syntax would be different so that SQL can understand the column you want to get. You would need to use the table name and then follow it with a period before the name of the column. For example, if you had two tables with a name column, the SELECT statement would look like this:

```
SELECT table1.name, table2.name FROM table1, table2;
```

This would select all names from both tables and wouldn't be a problem. If you left out the "table1." and "table2." portions, it would be an error in some systems since there are two references to the same column name.

Select all columns from the Customers table:

```
SELECT * FROM Customers;
```

Select the name column from the Customers table (Chris Velez, Jon Doe, Jane Doe):

```
SELECT Name FROM Customers;
```

Select two columns from one table based on a single condition (Jane Doe/Chris Velez):

```
SELECT Name, Age FROM Customers WHERE Age >= 25;
```

Select two columns from one table based on multiple conditions (Spot with PetID of 4):

```
SELECT PetID, Name FROM Customers WHERE Age > 5 AND PetType =  
'Dog';
```

INSERT

```
INSERT INTO [table] (column1, column2,...) VALUES ('value1',  
'value2',...);
```

This tells the database to insert a new row into the table with specified values for varying rows. Every column provided must match the column in the database exactly and must have an accompanying value to go with it. If you supply 3 columns, 3 values should also be provided. The order of the columns in the query should follow the column order of your table as well.

Insert a new row into our Customers table:

```
INSERT INTO Customers (UserID, Name, Age, Email) VALUES (4,  
'Sasha Velez', 9, 'svelez@mail.com');
```

If your table's primary key is set to "auto-increment," meaning that it will automatically increase the numerical value when a new row is entered, then you wouldn't need to insert a value for it in your command. You would just supply Name, Age, and Email along with 'Sasha Velez', 9, and 'svelez@mail.com'.

UPDATE

```
UPDATE [table] SET [column1] = [value1], [column2] = [value2],  
WHERE [column] = [some value];
```

This tells the database to update a specific row's data. We can update as many columns in that row as we want but each column/value pair must be separated by a comma. Omitting the WHERE clause will cause the command to update the entire table since no specific condition was set.

Update the age for the user with UserID of 1 (only Chris Velez):

```
UPDATE Customers SET Age = 29 WHERE UserID = 1;
```

Update the name and email for the user with UserID of 2 (only John Doe):

```
UPDATE Customers SET Name = 'Tony Stark', Email =  
'tstark@mail.com' WHERE UserID = 2;
```

Update the age for everyone (omits WHERE clause – it updates every row which is bad unless desired):

```
UPDATE Customers SET Age = 99;
```

DELETE

```
DELETE FROM [table] WHERE [column] = [value];
```

This allows a user to delete records from a table. If the WHERE part is omitted, the entire table may be deleted. 'DELETE *' will also delete everything in a table.

Delete Chris Velez from Customers table:

```
DELETE FROM Customers WHERE UserID = 1;
```

CREATE

This is used to create tables or databases. You should probably avoid allowing others to create databases on your website so we can avoid this command for our course except in examples/assignments. When you create a table, you can create it with basic constraints on data types or you can specify more complicated things such as primary key, data lengths, etc.

Creating a database:

```
CREATE DATABASE [database_name];
```

Creating a table:

```
CREATE TABLE [table_name] (  
    [Column1] [datatype],  
    [Column2] [datatype],  
    ...  
);
```

Creating our Customers table from earlier:

```
CREATE TABLE Customers (  
    UserID int AUTO_INCREMENT PRIMARY KEY,  
    Name varchar(255) NOT NULL,  
    Age int NOT NULL,  
    Email varchar(255),  
);
```

In the example above, I create a table call Customers. Each line represents a different column being created by the table along with constraints. In our first line, we are creating a column called "UserID" that is a type of int (number). We want it to serve as the primary key, so we use the PRIMARY KEY keywords. We also want it to automatically increment (increase by 1) every time a new record is added so we say AUTO_INCREMENT. If it doesn't automatically increment, then we'd have to manually provide this primary key each time a record was inserted.

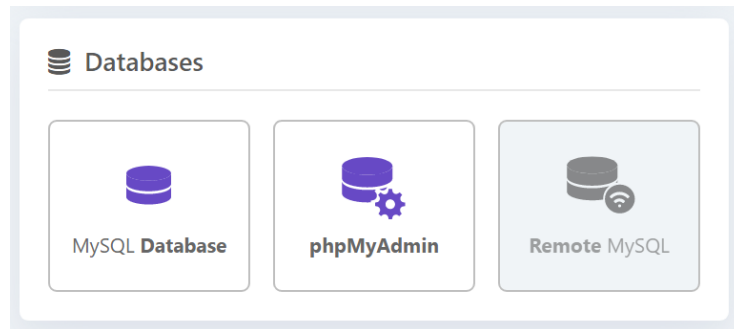
The Name column is created with a datatype of varchar which is just variable character. We use 255 and NOT NULL to specify that this column's value length can't exceed 255 characters and it can't have empty values. After all, each user should have a name and age. However, they may not have an email which is why I don't use NOT NULL on the email line. Each line is separated by a comma and wrapped inside of Customers(); Table creation is essentially a function which takes in the table structure as the argument.

Setting Up a Database on 000webhost

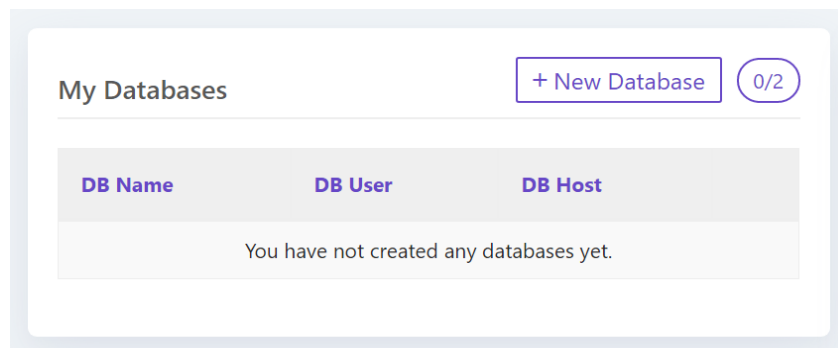
Before we can get fancy with PHP and SQL, we need to set up our database on 000webhost.

Creating a database on 000webhost is quick and easy to do. Log on to your 000webhost account.

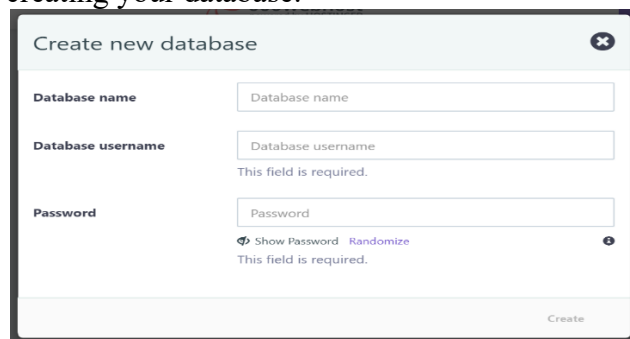
In your website's dashboard area, which contains your File Manager, scroll down to the section titled "Databases" and click the first option, "MySQL Database."



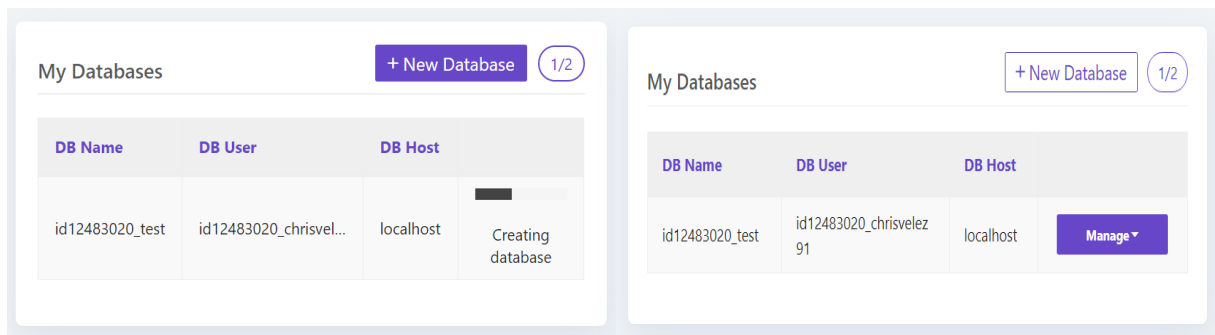
After clicking "MySQL Database" you will be brought to a screen with the following content on it. Click "+ New Database" to create a new database. Another screen will appear asking for credentials for your database.



On the next screen called "Create new database," you will add a database name, username, and password. **MAKE SURE YOU STORE THIS INFO SOMEWHERE.** This information is vital for establishing a connection to your database using PHP code. Once your info is typed in, click "Create" to finish creating your database.

A screenshot of a "Create new database" form. It contains three input fields: "Database name", "Database username", and "Password". Below the "Database username" field, there is a message "This field is required." Below the "Password" field, there is a message "This field is required." and two links: "Show Password" and "Randomize". At the bottom right of the form is a "Create" button.

After clicking "Create," the screen will close and one like the following will appear. You must wait for the "Creating database" progress bar to be finished before your database is available. That section will change to a purple "Manage" button with a dropdown arrow. The images below depict a before/after view of when your database is loaded versus when it is finished.



Combining PHP and SQL

Once your database has been created, you can start using PHP to pass SQL commands to it. To connect our PHP and database, you'll need the credentials which were established during the "Setting up a database on 000webhost" step. In total, you will need four pieces of information: the database name, the server name, the username, and the password for the user. It is very similar to how you used FileZilla to send files, except that information is being used to connect to a database.

We use a function which tries to connect to the database with those four pieces of information as arguments. If the connection is successful, we are then able to write queries. After we create our query, we test it against the connection. If the result isn't an error, we can then close the connection and choose to output our results.

If you've already created a database, please download the class-example files, and place them somewhere on your 000webhost site. The pwd.php file needs to be filled out with the pieces of information mentioned above. There are comments which explain where the info should be placed in the code.

Look through the code, especially at the \$sql variables. Those each contain SQL statements which can be executed. If you are confused about anything (such as what \$conn->query(\$sql) means), visit the following link and scroll down the menu on the left-hand side to see specific code explanations: https://www.w3schools.com/php/php_ref_mysqli.asp

If you have trouble and need to check your PHP, use this link:
<https://phpcodechecker.com/>

It will generally tell you where your error is and even if you don't understand what it means, look at the line numbers around the error and check for syntax problems.

Examples

Online Resources

- <https://www.sqltutorial.org/> (Sections 1 – 3 are solid)
- <https://www.tutorialspoint.com/sql/sql-syntax.htm> (List of general syntax)
- <https://www.php.net/manual/en/book.mysqli.php> (Very detailed reference on MySQL Improved Edition aka mysqli)
- <https://www.tutorialspoint.com/sql/sql-rdbms-concepts.htm> (RDBMS Concepts)
- https://www.w3schools.com/php/php_ref_mysqli.asp (MySQL Reference)

A new video will be released on Tuesday afternoon. This video will cover PHP and SQL in the context of this week's assignment. When it is available, the link will be added to the assignment description.

My Examples

This week's class example is different from any of the other ones. My live page will have limited functionality in terms of the database commands. This is to prevent any unwanted data being inserted or updated in my database. The class example provided today inside the zip folder has all the functionality built in except for inserting data manually. You'll need to enter your credentials for your database into the pwd.php file. Place the pwd.php and index.php file in the same folder on your 000webhost account and the page should work for you

- <https://cinf362.000webhostapp.com/examples/>

PowerPoint documents have also been released under Blackboard → Course Materials → Resources → PHP & SQL. One of them offers a brief introduction to PHP/SQL while the other will serve as a debugging guide.

You won't be able to right-click the page and view the source this week to view my PHP code. Since PHP is processed by the server and rendered as HTML, the user can't see the PHP code you're using. The code for the example below is in the "examples" folder that was included with this week's lecture notes zip folder. You can take all the contents of the examples folder and place them somewhere on your 000webhost account and the pages should work. There are comments in the code to explain what is happening. Feel free to look at the live link to see the page in action.

Here's a link to **all** course examples:

<https://cinf362.000webhostapp.com/examples/> (PHP/SQL Examples)

<https://www.albany.edu/~cv762525/cinf362/examples/> (HTML, CSS, and JS examples)

<https://www.albany.edu/~cv762525/cinf362/videos/> (videos)

<https://drive.google.com/drive/folders/13sh0oaUeE9di4aZuTYczqNdpzdb4kzKE?usp=sharing>
(more videos)

Before completing the assignment(s) for this week, please read the "Viewing Your Web Pages.docx" file on Blackboard. You will not be able to submit anything for the assignment without completing that portion first. It is located directly in the Lecture Notes folder.

Database Challenge

Due Monday, April 18th at midnight

Download the "Database-Challenge.zip" folder from Blackboard under this week's lecture notes. Inside of the folder will be two PHP files to use for this assignment. Your task this week will be to use SQL to query a database and write down the queries to submit. In completing this assignment, you'll learn a little bit about the interface to a database system and how to manipulate the data inside of it.

There are three parts to this assignment. In the first part, you will create a database, add a table to that database, and then fill it with data. You'll also have the option to delete the database if you'd like. In the second part, we'll navigate to the area which will allow us to run queries on the database. This is what most generic database interfaces look like. In the third part, you'll try the query challenges listed.

Part One

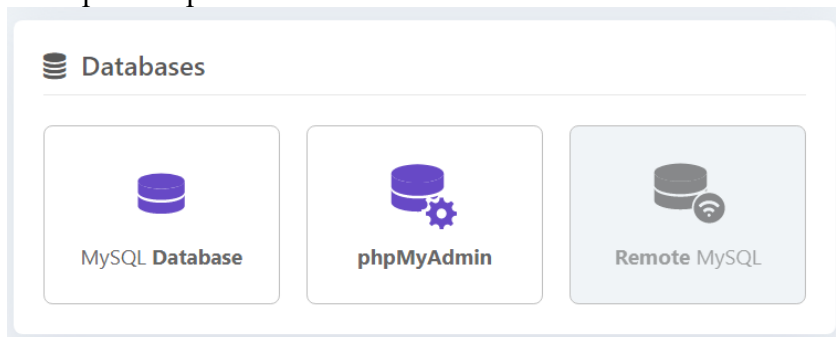
1. If you haven't done so already, create a database as shown in the instructions under the "Setting Up a Database on 000webhost" section.
2. When you have created a database and have your information (username, password, host, and database name), use it to fill out the password file (pwd.php).
3. Place the database-challenge.php and pwd.php files in your 000webhost area
4. View the database-challenge.php page
5. Click the "Create Table" and "Populate Table" buttons on the database-challenge page to create a table and add information to it. You should also click "View Table" to see its contents.

The delete button is available if you wish to quickly delete your database. Please note that some features won't work fully in some situations. For example, if you try to populate, view, or delete the table when it doesn't exist, an error will be output to your page.

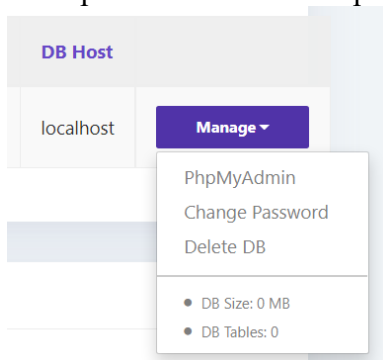
Part Two

After you have confirmed that your database and table are up and running, you'll go into the administrative area of your database so that you can manipulate it. Follow the instructions/screenshots below to access the database directly inside of 000webhost.

In the Dashboard of your 000webhost area (same place where you can go to the File Manager), scroll down until you reach the "Databases" section and click "MySQLDatabase" which is the first option depicted below.



On the new screen, click the purple "Manage" button with the dropdown menu and then select "PhpMyAdmin." The DB Name, DB User, and DB Host credentials are shown on this page. Your password will not display by default.

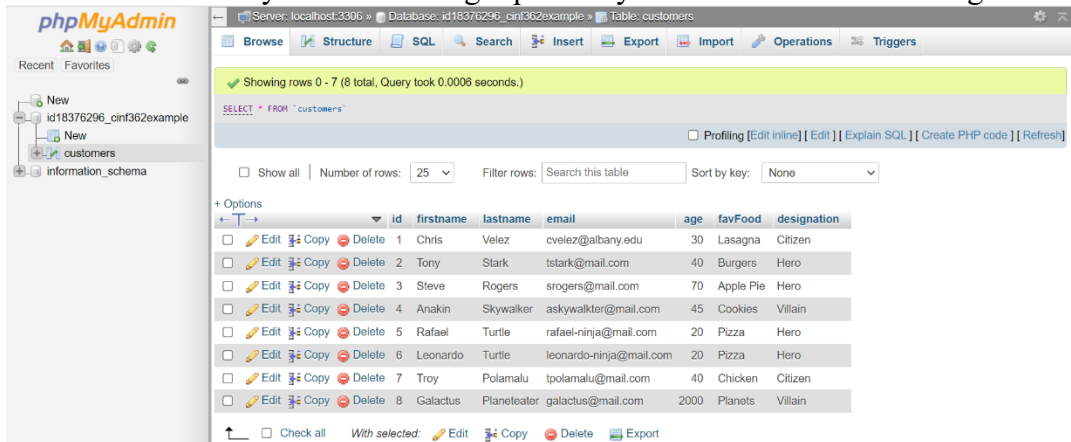


Once you're in the phpMyAdmin area, you'll be able to access your databases and tables. This can be done through the dropdown on the left side or through the "Databases" tab at the top.

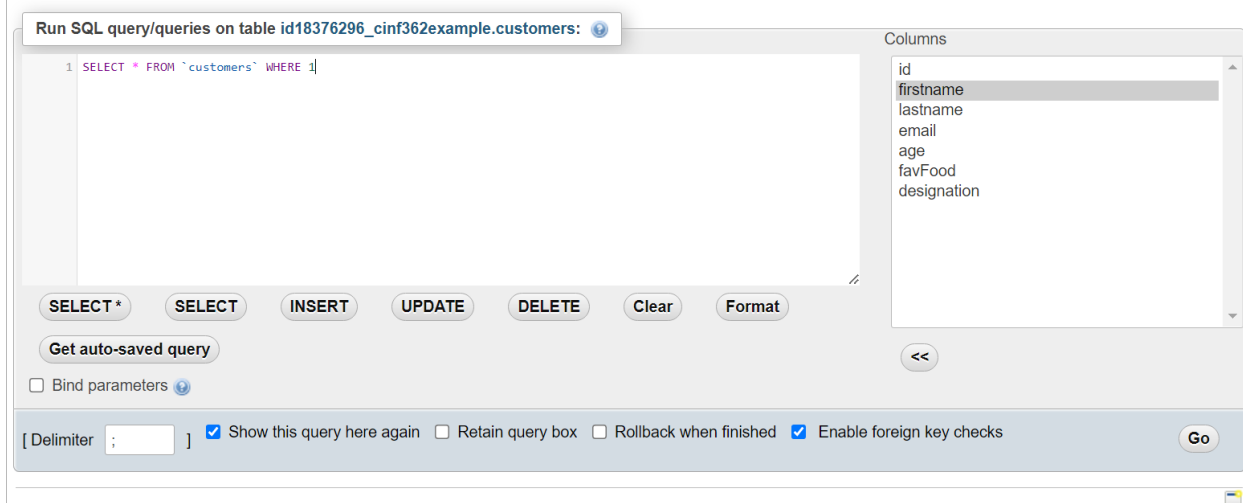
On the left side, you would click the plus signs to expand the dropdown menus until you found the customers table. You could then click it and the table will display. **This option will be the easiest.**

As an alternative, at the top, you would click “Databases,” and this will pull up a page with all your databases. You could then click the name of your database, and on the next screen, you’d click on “customers.”

You will know that you are in the right place if your screen looks something like below:



From here, you can see the database data and manipulate it. Your next step will be to click “SQL” at the top (3rd tab). It will bring you to a page which looks like the image below. This is where you can use SQL to directly manipulate the table.



Click “Clear” to empty the query box and check the box which says, “Retain query box.”. After this, you can press any of the other buttons such as “SELECT *” or SELECT to auto-populate the query box with an SQL statement related to its function. You can use these buttons to quickly put together basic SQL queries for this assignment or type them in manually. The “Columns” section on the left will let you add column names to the query. You just need to click the “<<” button to add them.

The results of your queries will appear directly below in an area which says “Showing rows” with a yellow background (assuming your query was correctly formatted/valid). Your query box may hide, so you can click “Show query box” to bring it back. If the query was unsuccessful, a red box will appear with an explanation.

Part Three

Now that you're all set up, your task will be to try and run queries on the database based on the challenges below. The answer to each challenge will be an SQL statement that could be executed to achieve the results mentioned. You'll add your SQL statement to the query box, click "Go" to execute your code, and the results will appear below.

For example, if the challenge was "Display all data for all customers who have Lasagna as their favorite food," the answer would be:

SELECT * FROM customers WHERE favFood = "Lasagna"

The results of the query would be this:

+ Options

			id	firstname	lastname	email	age	favFood	designation	
<input type="checkbox"/>	Edit	Copy	Delete	1	Chris	Velez	cvelez@albany.edu	30	Lasagna	Citizen
	<input type="checkbox"/> Check all	With selected:		Edit	Copy	Delete	Export			

I use the customers table and name of the column (favFood) based on the database created by our page. You can test your queries inside of the big box and then click the "Go" button to execute them. Make sure you review the names of the columns before executing your queries.

These are the 8 queries to try for this challenge. When I say display, you should use a SELECT statement to retrieve the data based on the conditions provided.

1. Display the first name and email for each customer
2. Display the emails for all customers that are 40 years old or younger
3. Display the first names for all customers with a designation of hero that have pizza as their favorite food
4. Display the first name, last name, and age of all customers with the last name "Turtle"
5. Update customers with the last name Turtle to have a last name of "Ninja" instead
6. Display the first name, last name, and age for all customers with a designation of citizen
7. Display all columns for any customer with a designation of villain whose favorite food is cookies
8. Insert a new customer with your own information (don't include primary key in insert statement)

To submit your answers, simply provide a challenge number along with your SQL statement to run the query mentioned in a text file or as a comment on Blackboard.

Make sure your PHP files are on a server that supports them. The UAlbany servers don't, so you should be using 000webhost or another platform such as byethost. If your PHP doesn't work, put your code in the PHP Checker link provided in the lecture notes above. It should give you a line number and a reason why it's failing. Check the line number (if provided) and adjust the code as necessary. Also, make sure your file has a .php extension. Without this extension, the browser won't know to treat it like a PHP file.

Your webpage and queries are **due on Monday, April 18th at midnight**. To submit the work for this exercise, visit Blackboard → Course Materials → Lectures Notes for this week's class. You can also go into the "Assignments" folder and the submission area will be titled "Database Challenge." You should be submitting a list of queries to me in addition to a link to the Database-Challenge.php" page. The queries can be done as a comment or in some text document. The work submitted will be evaluated based on the rubric explained below.

Database Challenge Rubric – 10pts

- Live link was provided – 1pt
- Database was created – 1pt
- Query code is correct – 8pts
 - Each challenge is 1pt

Database Structure (Two Posts)

Initial Post due Friday, April 15th at midnight

Most of the websites you have discussed so far have probably used one or more databases to add interactive features to their pages. Try to visit a website you haven't previously discussed which utilizes databases. Think about the structure that might be implemented for one of that website's features/databases and answer the following questions.

- What are some databases that may be used by this website?
- What tables would be needed for the database? Will any of them be linked?
- What columns/fields would those tables have?
 - What data types would those columns have?
 - Are there any constraints you think would be needed?
- What would be the primary or foreign keys of these tables?
- What pieces of information going into this database are sensitive/personal and need to be protected?

This discussion is entirely hypothetical so there is not an exact right or wrong answer. However, it should be evident from your post that you thought about tables, data, and how they might be connected. There are many ways tables and data can be set up for applications on a website so try to be as thorough as possible with explaining your ideas. Do not create a post in question/answer format. You should be writing complete sentences to express your thoughts in about two small paragraphs.

If you're unsure of what kind of websites to look for, use the list below for a reference:

- Restaurant
- Social media
- Sports
- Government

Submission

The initial post is **due Friday, April 15th at midnight**. To submit, go to Blackboard → Course Materials → Lecture Notes for this week's class. There will be a discussion area called "Database Structure" where you can post your initial post. You can also visit the Discussion Board area directly from the Course Materials folder.

Response Post due April 18th at midnight

In your response post, visit a website another student has discussed that you haven't already mentioned and in one paragraph, talk about their database structure. Do you agree with the information they've provided? Are there any other databases on their websites? What other tables, columns, or rows might be important to their database structure? Can you think of any additional constraints on their data?

The response post is **due Sunday, April 18th at midnight**. To submit, go to Blackboard → Course Materials → Lecture Notes for this week's class. There will be a discussion area called "Database Structure" where you can post your response post. You can also visit the Discussion Board area directly from the Course Materials folder.

Database Structure Rubric – 2pts

The initial post is worth 1.5pts and the response post is worth .5pt for a total of 2 points. I will be evaluating your posts based on the following criteria:

- Was a website with a database or potential database feature evaluated?
- Did you mention specific information related to database structure?
 - Were tables and columns mentioned?
 - What data is important?
- Did your response post contribute to the original post?
 - Avoid summarizing the other person's post or simply saying that it was a good post.
 - Were new tables, columns, or rows mentioned?

Next Week

Next week, we will learn more about SQL and other commands we can use to manipulate a database or filter results. We'll also go over how we can use PHP and MySQL together in more detail.