WEB DESIGN AND PROGRAMMING

class 9 exercise & homework

Read here first

For this in class activity, you will need to submit the following

1) Ap.sql
2) Customer_alter.sql
3) This PDF file (fill your answer in the answer box and save it)

Please put all of the files in a folder name "XXX_class09", zip the folder and **submit the zip file. (XXX is your student ID)** **through SCOMB**

This in class activity requires you to work in your own local server (not the university server) so please use XAMPP.

Part 0

XAMPP Installation

Note: Nothing to submit here

Be aware that the default installation of XAMPP is not secure. As a result, it should not be used as a production server until it is properly secured.

Before you work with the current date and time in PHP, you need to **set the default time zone so the correct date and time are returned**. In most cases, if you don't set the time zone, the date and time that are returned by PHP will reflect Greenwich Mean Time (GMT), also called Universal Time Coordinated (UTC).

Set the default time zone for PHP

1. Open the `php.ini` file in a text editor such as VS Code, Notepad++, Sublime Text etc. The file is stored as follow.

   `c:\xampp\php\php.ini` (open as administrator)

   `/Applications/XAMPP/xamppfiles/etc/php.ini`

2. Search the file for the second occurrence of 'timezone'. (The first one will be commented out.)
   In my computer, it is as follows

```
[Date]
; Defines the default timezone used by the date functions
; http://php.net/date.timezone
date.timezone=Europe/Berlin
```

3. Set the timezone **correctly to your region**.

   For example, for Tokyo Time, you can set the time zone like this:

```
date.timezone=Asia/Tokyo
```

Check for timezone here https://www.php.net/manual/en/timezones.php
You will need the administrator password to save the file.

How to configure phpMyAdmin

XAMPP includes a web-based tool for working with MySQL named phpMyAdmin. By default, phpMyAdmin stores the username and password for MySQL's root user in its configuration file, which is a plain text file. Since this isn't secure, it's generally considered a good practice to modify these default settings. We recommend changing the authentication type for phpMyAdmin from `config` to `cookie`

When you use the `cookie` authentication type, phpMyAdmin does not store any usernames or passwords in its configuration file. Instead, it prompts the user for a username and password when it starts, and it stores the username and password in a cookie in the user's browser.

Once you've modified the configuration file for phpMyAdmin, you can use the second procedure in this figure to make sure the MySQL server is working correctly and to make sure that phpMyAdmin is configured correctly.

The default location of the config file for **phpMyAdmin**

```
C:\xampp\phpMyAdmin\config.inc.php
/Applications/XAMPP/xamppfiles/phpmyadmin/config.inc.php
```

How to configure authentication for phpMyAdmin

1. Open the `config.inc.php` file (in a text editor such as VS Code, etc.) for editing.
2. Normally, the '`blowfish_secret`' option should be set to a random string up to 46 characters. This specifies the encryption key for the cookie. However, in this class you can leave it as it is (`xampp`).
3. Set the '`auth_type`' option to a value of '`cookie`'.
4. Set the '`user`' and '`password`' options to `empty strings`.
5. Save your changes.

The default settings

```php
declare(strict_types=1);

/**
 * This is needed for cookie based authentication to encrypt password in
 * cookie. Needs to be 32 chars long.
 */
$cfg['blowfish_secret'] = 'xampp'; /* YOU SHOULD CHANGE THIS FOR A MORE SECURE COOKIE AUTH! */

/**
 * Servers configuration
 */
$i = 0;

/**
 * First server
 */
$i++;
/* Authentication type */
$cfg['Servers'][$i]['auth_type'] = 'config';
$cfg['Servers'][$i]['user'] = 'root';
$cfg['Servers'][$i]['password'] = '';
```

Figure 1: `config.inc.php` before change

The settings after phpMyAdmin has been configured for authentication
Note that **blowfish_secret** is not changed in this class, but you should change when deploy a real web application.

```php
/* Authentication type */
$cfg['Servers'][$i]['auth_type'] = 'cookie';
$cfg['Servers'][$i]['user'] = '';
$cfg['Servers'][$i]['password'] = '';
```

Figure 2: `config.inc.php` after change

Part 1

Using phpMyAdmin to test the MySQL server

1. Use the XAMPP Control Panel to start the Apache and MySQL servers
2. Start Chrome and enter this URL: `http://localhost/phpmyadmin/`
3. This should start phpMyAdmin and prompt you for a username and password. If it doesn't, click the "Log out" link to get to the log in page.
4. Log in as the root user by specifying a username of 'root' and the password for the root user.
5. By default, the root user doesn't have a password, so you don't need to enter one. When you successfully log in as the root user, phpMyAdmin should display its Home page. At this point, both the MySQL server and phpMyAdmin are working correctly.

How to log in, log out, and change your password

Since the `root` user is the default admin user for MySQL, logging in as the `root` user gives you access to all databases on the server. By default, XAMPP doesn't set a password for the root user.

In general you should set a password for the root user before you store any sensitive information in the database. To do that, log in without entering a password to go to the Home page and then click the *Change Password* link. For "cookie" authentication, phpMyAdmin stores the username and password that you enter in a cookie in the browser. As a result, you are only prompted for the username and password when you start phpMyAdmin for the first time.

On subsequent starts, phpMyAdmin uses the cookie to log in automatically, skips the Welcome page, and displays the Home page. The exception is if you close your browser, in which case you need to log in again. That is usually what you want. If you're logged in as one user and you want to log in as another user, you can use the Log out button in the toolbar. Then, phpMyAdmin displays the Welcome page so you can log in as another user. You may also want to log out for security reasons.

If, for example, you're using a shared computer, you should log out when you're done.

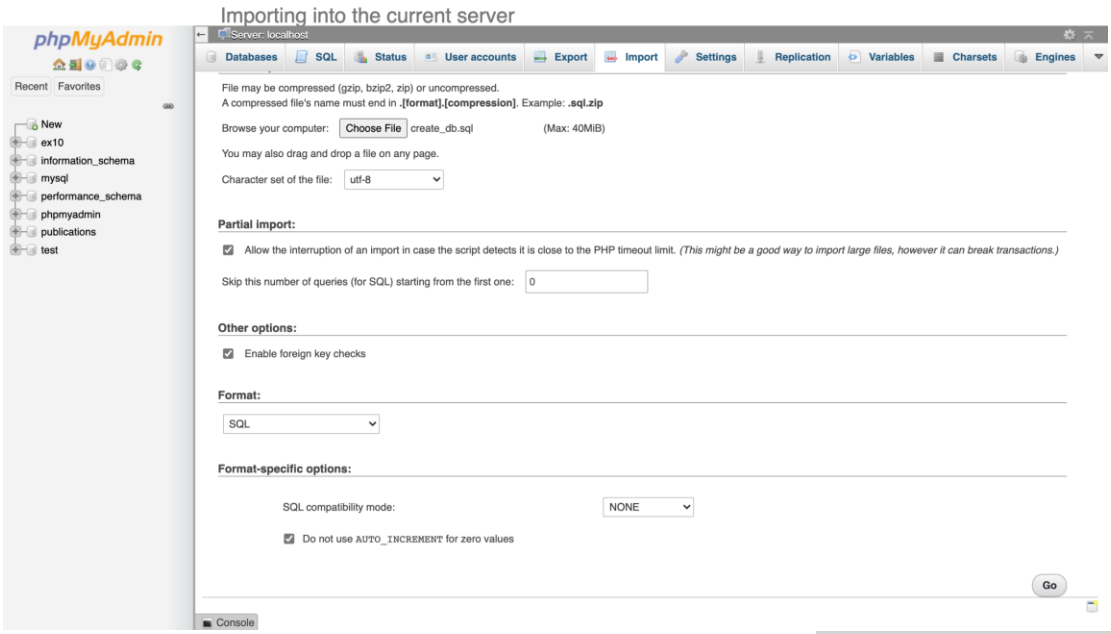
Please remember the password that you set, you will need to use it everytime when you need to access your databases. If you are only using MySQL in your localhost for working with exercises in this class, you don't need to set the password (leave it default which is nothing), or you can set to something very easy to remember such as "webpro".

Use phpMyAdmin with a database

Note: Write the answers in the textbox

Run the script for creating the book databases

1. On the phpMyAdmin 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.
2. Click Import Tab, go to the "File to Import" section. Choose File and select the file containing MySQL script (`create_db.sql`) to import.
3. Click "Go" button. This runs the script that is in the file.



When the import is successful, it will show the result similar to **phpMyAdmin_result.jpg** (see the file in this exercise folder)

4. This will create or recreate the two databases. 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*

5. 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.
6. 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
7. Use the SQL tab to run the query as follows
   `SELECT * FROM products WHERE categoryID = 2`

What is the result?
(capture the result using screen capture in the following box – only the relevant part is needed)

Then, run the second query as follows

```
SELECT productName, listPrice FROM products
WHERE listPrice < 500 ORDER BY listPrice ASC
```

What is the result?

8. Run the following query.

```
SELECT categoryName, productName, listPrice FROM categories
INNER JOIN products
ON categories.categoryID = products.categoryID WHERE
listPrice > 800 ORDER BY listPrice ASC
```

What is the result?

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.
What is your updated query?

Run the following query to add a row to the products table.

```
INSERT INTO products
(categoryID, productCode, productName, listPrice)
VALUES
 (1, 'tele', 'Fender Telecaster', 599.00)
```

Then, browse the products table to view the new row. Last, run a `DELETE` statement to delete the new row.

9. 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
10. 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 2 of this exercise.
11. Use the SQL tab to run this SELECT statement: `SELECT * FROM categories` What happen?.
    What is the result?

Experiment

12. 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.

Part 2

Exercise 1: Create a database and alter it

Note: Submit `customers_alter.sql` by putting it in the folder

In this exercise, you will run a script that creates a database named `my_guitar_shop2`. Then, you will create your own script that contains some statements that alter that database.

Create a database by executing an existing script

1. Use a text editor to open the script named `my_guitar_shop2.sql`.

   (Don't worry if you already have it from the previous exercise, the script will delete the existing `my_guitar_shop2` and create a new one for you)
2. Review the code and note how it creates the database named `my_guitar_shop2`.
3. Use phpMyAdmin to execute this script. Then, view the structure of the database.

Create a script that alters the database

4. Use your text editor to create a file named `customers_alter.sql` that you'll use for storing a script.
5. Write an `ALTER TABLE` statement that adds a column named `middleInitials` to the customers table. This column should store up to 3 characters, allow `NULL` values, and be added after the `firstName` column.
6. Write an `ALTER TABLE` statement that modifies the customers table so the `lastName` column can store up to `100` characters.
7. Use phpMyAdmin to test this script. Then, use phpMyAdmin to view the structure of the database. Check to make sure the `middleInitials` column has been added and that the data type for the `lastName` column has been changed.

Exercise 2: Create a simple database

Note: Submit `ap.sql` by putting it in the folder

In this exercise, you will write a script that creates a simple Accounts Payable (AP) database.

Write a script that creates a database and its tables

1. Start your text editor and create a file named `ap.sql` that you'll use for storing a script.
2. Write the `CREATE DATABASE` statement needed to create a database named `ap`. If the database already exists, drop it.
3. Test these statements by cutting and pasting them into phpMyAdmin.
4. Write the `USE` statement that selects the database.
5. Write `CREATE TABLE` statements to create the following tables in the `ap` database:

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
☐	1	**vendorID**	int(11)			No	*None*		AUTO_INCREMENT
☐	2	**vendorName**	varchar(45)	utf8mb4_general_ci		No	*None*		
☐	3	**vendorAddress**	varchar(45)	utf8mb4_general_ci		No	*None*		
☐	4	**vendorCity**	varchar(45)	utf8mb4_general_ci		No	*None*		
☐	5	**vendorState**	varchar(45)	utf8mb4_general_ci		No	*None*		
☐	6	**vendorZipCode**	varchar(10)	utf8mb4_general_ci		No	*None*		
☐	7	**vendorPhone**	varchar(20)	utf8mb4_general_ci		No	*None*		

Figure 3 vendors table

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
☐	1	**invoiceID**	int(11)			No	*None*		AUTO_INCREMENT
☐	2	**vendorID**	int(11)			No	*None*		
☐	3	**invoiceNumber**	varchar(45)	utf8mb4_general_ci		No	*None*		
☐	4	**invoiceDate**	datetime			No	*None*		
☐	5	**invoiceTotal**	decimal(10,0)			No	*None*		
☐	6	**paymentTotal**	decimal(10,0)			Yes	*NULL*		

Figure 4 invoices table

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
☐	1	**lineItemID**	int(11)			No	*None*		AUTO_INCREMENT
☐	2	**invoiceID**	int(11)			No	*None*		
☐	3	**description**	varchar(45)	utf8mb4_general_ci		No	*None*		
☐	4	**quantity**	int(11)			No	*None*		
☐	5	**price**	int(11)			No	*None*		
☐	6	**lineItemTotal**	decimal(10,0)			No	*None*		

Figure 5 lineitems table

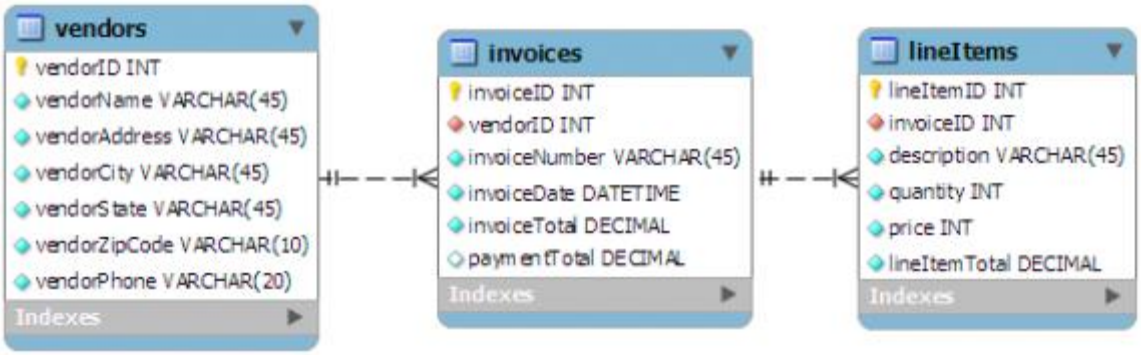


Figure 6 The ER diagram

When you create these tables, be sure to use the correct data types, to identify the primary key for each table, and to identify the foreign key constraints for the invoices and lineItems tables. Also, make sure to automatically generate a value for each primary key.
For each column, include any `UNIQUE`, `NOT NULL`, or `DEFAULT` attributes you think are necessary.

6. Use phpMyAdmin to run the script. Then, view the structure of the database.

Add statements to the script to create some indexes and a user

7. Add `CREATE INDEX` statements to the end of the script to create indexes for the foreign keys of the `invoices` and `lineItems` tables. Also, write a `CREATE INDEX` statement to create an index for the `invoiceNumber` column.
8. Add a `GRANT` statement to the end of the script that creates a user named `ap_user` with a password of "`sesame`" and grants this user privileges to `select`, `insert`, or `update` data from any table in the database. *However, don't allow this user to delete any data from the database.*
9. Test the entire `ap.sql` script to make sure it runs correctly.

Exercise 3: Work with the data in a database

Note: Write your answers in the textbox

Run some of the examples

1. Start phpMyAdmin.
2. Use your text editor to open the script named `class09_part2_ex3a.sql`. Then, use phpMyAdmin to run the first `SELECT` statement in this script. To do that, you can select that statement in your text editor and copy it into phpMyAdmin. (Don't forget to use phpMyAdmin to select the correct database.) Then, run the next three statements to limit the number of columns and rows.
3. Open the script named `class09_part2_ex3b.sql` in your text editor, and use phpMyAdmin to run the first `SELECT` statement in this script. Note how this statement selects data from two tables. Then, run the second `SELECT` statement. Note how this statement selects data from four tables.

Write your own SELECT statements

4. Use phpMyAdmin to write and test a `SELECT` statement that selects the `productName`, `description`, and `listPrice` columns for all rows in the `products` table. Add code to this statement so it sorts the result set by list price. Then, run this statement again to make sure it works correctly. This is a good way to build and test a statement, one clause at a time. Add code to this statement so it only selects rows that have the word "`electric`" in the `description` column, and run this statement again to make sure it works correctly.

5. Write a `SELECT` statement that joins data from the `customers` and `addresses` tables. This statement should select these columns: `firstName`, `lastName`, `line1`, `line2`, `city`, `state`, `zipCode`. It should only select the billing addresses for customers who have a last name of "`sherwood`".

6. Write a `SELECT` statement that returns a count of the number of products in the category that has a name of "`Guitars`". To do this, use a subquery to get the category ID.

Write your own INSERT, UPDATE, and DELETE statements

7. Write an `INSERT` statement that adds a customer named `John Smith` to the customers table. Use an email address of "`johnsmith@example.com`" and a password of "`sesame`".

8. Write an `UPDATE` statement that changes the password for `John Smith` to "`5e5ame!`".

9. Write a `DELETE` statement that deletes the customer named `John Smith`.

10. To restore the database to the way it was initially, run the creation script `my_guitar_shop2.sql` again.