ЗВІТ

про виконання лабораторної роботи № 1

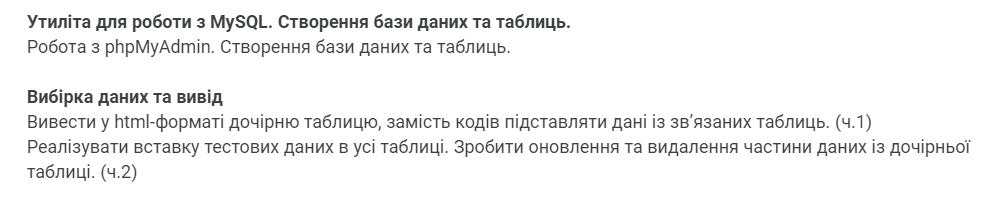
Утиліта для роботи з MySQL. Створення бази даних та таблиць.

з дисципліни «Веб-програмування»

студента групи КН-2226б

Медвіль Дмитро Степанович

Завдання:



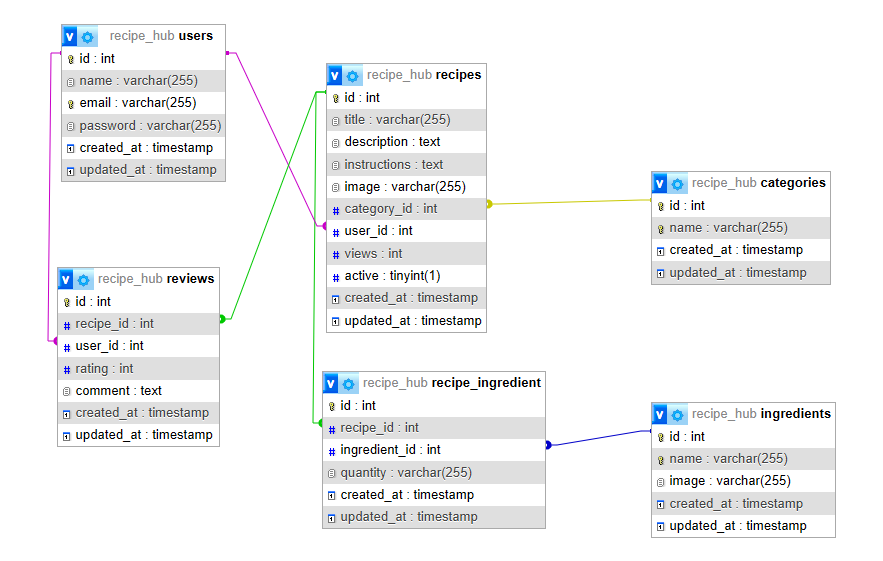
Код:

<?php  
// Підключення конфігураційного файлу з параметрами підключення до бази даних.  
require\_once 'config/database.php';  
  
*/\*\*  
 \* Функція для встановлення з'єднання з базою даних.  
 \** ***@return*** *PDO|null Повертає об'єкт PDO при успішному підключенні або null у разі помилки.  
 \*/*function getConnection() {  
 try {  
 // Створення нового об'єкта PDO з параметрами підключення з конфігураційного файлу.  
 $pdo = new PDO("mysql:host=".*DB\_HOST*.";dbname=".*DB\_NAME*, *DB\_USERNAME*, *DB\_PASSWORD*);  
 // Встановлення режиму помилок на виключення для кращого відловлювання помилок.  
 $pdo->setAttribute(PDO::*ATTR\_ERRMODE*, PDO::*ERRMODE\_EXCEPTION*);  
 return $pdo; // Повертає об'єкт з'єднання.  
 } catch (PDOException $e) {  
 // Виведення повідомлення про помилку у разі невдалого підключення.  
 echo 'Connection failed: ' . $e->getMessage();  
 return null; // Повертає null у разі невдалого підключення.  
 }  
}  
  
*/\*\*  
 \* Функція для виконання SQL-запиту SELECT з необов'язковими параметрами.  
 \** ***@param*** *string $query SQL-запит для виконання.  
 \** ***@param*** *array $params Параметри для підготовленого запиту (за замовчуванням порожній масив).  
 \** ***@return*** *array Повертає масив рядків з результатами запиту.  
 \*/*function all($query, $params = []) {  
 $pdo = getConnection(); // Отримання з'єднання з базою даних.  
 if ($pdo === null) {  
 return []; // Повертає порожній масив, якщо підключення не вдалося.  
 }  
  
 // Підготовка та виконання SQL-запиту.  
 $stmt = $pdo->prepare($query);  
 $stmt->execute($params);  
 // Повертає всі результати запиту у вигляді асоціативного масиву.  
 return $stmt->fetchAll(PDO::*FETCH\_ASSOC*);  
}  
  
*/\*\*  
 \* Функція для вставки даних у таблицю бази даних.  
 \** ***@param*** *string $table Назва таблиці для вставки.  
 \** ***@param*** *array $data Масив даних для вставки у форматі [назва\_колонки => значення].  
 \** ***@return*** *bool Повертає true при успішній вставці або false у разі помилки.  
 \*/*function insert($table, $data) {  
 $pdo = getConnection(); // Отримання з'єднання з базою даних.  
 if ($pdo === null) {  
 return false; // Повертає false, якщо підключення не вдалося.  
 }  
  
 // Формування списку колонок та плейсхолдерів для вставки.  
 $columns = implode(', ', array\_keys($data));  
 $placeholders = implode(', ', array\_fill(0, count($data), '?'));  
 // Створення SQL-запиту на вставку даних.  
 $sql = "INSERT INTO $table ($columns) VALUES ($placeholders)";  
  
 // Підготовка та виконання запиту.  
 $stmt = $pdo->prepare($sql);  
 return $stmt->execute(array\_values($data)); // Виконує запит з переданими значеннями.  
}  
  
*/\*\*  
 \* Функція для оновлення даних у таблиці бази даних.  
 \** ***@param*** *string $table Назва таблиці для оновлення.  
 \** ***@param*** *array $data Масив даних для оновлення у форматі [назва\_колонки => значення].  
 \** ***@param*** *array $conditions Умови для оновлення у форматі [назва\_колонки => значення].  
 \** ***@return*** *bool Повертає true при успішному оновленні або false у разі помилки.  
 \*/*function update($table, $data, $conditions) {  
 $pdo = getConnection(); // Отримання з'єднання з базою даних.  
 if ($pdo === null) {  
 return false; // Повертає false, якщо підключення не вдалося.  
 }  
  
 // Формування списку колонок та значень для оновлення.  
 $set = [];  
 foreach ($data as $column => $value) {  
 $set[] = "$column = ?";  
 }  
 $setClause = implode(', ', $set); // Створення частини запиту SET.  
  
 // Формування умов для оновлення (WHERE).  
 $where = [];  
 foreach ($conditions as $column => $value) {  
 $where[] = "$column = ?";  
 }  
 $whereClause = implode(' AND ', $where); // Створення частини запиту WHERE.  
  
 // Створення SQL-запиту на оновлення даних.  
 $sql = "UPDATE $table SET $setClause WHERE $whereClause";  
  
 // Підготовка та виконання запиту.  
 $stmt = $pdo->prepare($sql);  
 // Виконує запит з переданими значеннями для оновлення та умов.  
 return $stmt->execute(array\_merge(array\_values($data), array\_values($conditions)));  
}  
  
*/\*\*  
 \* Функція для видалення даних з таблиці бази даних.  
 \** ***@param*** *string $table Назва таблиці для видалення.  
 \** ***@param*** *array $conditions Умови для видалення у форматі [назва\_колонки => значення].  
 \** ***@return*** *bool Повертає true при успішному видаленні або false у разі помилки.  
 \*/*function dbDelete($table, $conditions) {  
 $pdo = getConnection(); // Отримання з'єднання з базою даних.  
 if ($pdo === null) {  
 return false; // Повертає false, якщо підключення не вдалося.  
 }  
  
 // Формування умов для видалення (WHERE).  
 $where = [];  
 foreach ($conditions as $column => $value) {  
 $where[] = "$column = ?";  
 }  
 $whereClause = implode(' AND ', $where); // Створення частини запиту WHERE.  
  
 // Створення SQL-запиту на видалення даних.  
 $sql = "DELETE FROM $table WHERE $whereClause";  
  
 // Підготовка та виконання запиту.  
 $stmt = $pdo->prepare($sql);  
 return $stmt->execute(array\_values($conditions)); // Виконує запит з переданими значеннями умов.  
}  
  
*/\*\*  
 \* Функція для виконання довільного SQL-запиту з необов'язковими параметрами.  
 \** ***@param*** *string $sql SQL-запит для виконання.  
 \** ***@param*** *array $params Параметри для підготовленого запиту (за замовчуванням порожній масив).  
 \** ***@return*** *array|bool Повертає результати запиту або false у разі помилки.  
 \*/*function query($sql, $params = []) {  
 $pdo = getConnection(); // Отримання з'єднання з базою даних.  
 if ($pdo === null) {  
 return false; // Повертає false, якщо підключення не вдалося.  
 }  
  
 // Підготовка та виконання SQL-запиту.  
 $stmt = $pdo->prepare($sql);  
 $stmt->execute($params);  
 // Повертає всі результати запиту у вигляді асоціативного масиву.  
 return $stmt->fetchAll(PDO::*FETCH\_ASSOC*);  
}  
?>

<?php  
include\_once "database.php";  
function recipeList(){  
 $sql = 'SELECT r.id, r.title, r.image, r.description, r.created\_at, r.views, c.name as category\_name, COUNT(rv.id) AS total\_reviews FROM recipes r   
 LEFT JOIN categories c ON r.category\_id = c.id   
 LEFT JOIN reviews rv ON rv.recipe\_id = r.id   
 WHERE r.active = ? GROUP BY r.id';  
 $params = [1];  
  
 return query($sql, $params);  
}  
  
function delete($id){  
 dbDelete('recipes', ['id' => $id ]);  
 header("Location:/recipes.php");  
}  
  
  
function show($id){  
 $sql = 'SELECT r.title, r.image, r.description, r.created\_at, r.views, c.name as category\_name, COUNT(rv.id) AS total\_reviews , r.instructions as instructions FROM recipes r   
 LEFT JOIN categories c ON r.category\_id = c.id   
 LEFT JOIN reviews rv ON rv.recipe\_id = r.id   
 WHERE r.active = ? AND r.id = ? GROUP BY r.id';  
 $params = [1, $id];  
 $recipe = query($sql, $params);  
  
 $sql = 'SELECT ri.quantity, i.name FROM recipe\_ingredient ri   
 LEFT JOIN ingredients i ON ri.ingredient\_id = i.id  
 WHERE ri.recipe\_id = ? ';  
 $params = [$id];  
 $ingredients = query($sql, $params);  
  
 return ['recipe' =>$recipe[0], 'ingredients' => $ingredients];  
  
}

function create($title, $description, $instructions, $category\_id, $user\_id) {  
 // SQL запит на вставку нового рецепту  
 $sql = "INSERT INTO recipes (title, description, instructions, category\_id, user\_id)   
 VALUES (:title, :description, :instructions, :category\_id, :user\_id)";  
  
 // Параметри для запиту  
 $params = [  
 ':title' => $title,  
 ':description' => $description,  
 ':instructions' => $instructions,  
 ':category\_id' => $category\_id,  
 ':user\_id' => $user\_id  
 ];  
  
 // Виклик функції query для виконання запиту  
 return query($sql, $params);  
}

База даних:



CREATE TABLE categories (

id int NOT NULL AUTO\_INCREMENT PRIMARY KEY,

name varchar(255) UNIQUE,

created\_at timestamp DEFAULT CURRENT\_TIMESTAMP,

updated\_at timestamp DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP

) ENGINE=InnoDB;

CREATE TABLE ingredients (

id int NOT NULL AUTO\_INCREMENT PRIMARY KEY,

name varchar(255) UNIQUE,

image varchar(255),

created\_at timestamp DEFAULT CURRENT\_TIMESTAMP,

updated\_at timestamp DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP

) ENGINE=InnoDB;

CREATE TABLE users (

id int NOT NULL AUTO\_INCREMENT PRIMARY KEY,

name varchar(255),

email varchar(255) UNIQUE,

password varchar(255),

created\_at timestamp DEFAULT CURRENT\_TIMESTAMP,

updated\_at timestamp DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP

) ENGINE=InnoDB;

CREATE TABLE recipes (

id int NOT NULL AUTO\_INCREMENT PRIMARY KEY,

title varchar(255),

description text,

instructions text,

image varchar(255) DEFAULT 'assets/images/none\_image.jpg',

category\_id int,

user\_id int,

views int DEFAULT 0,

active tinyint(1) DEFAULT 0 NOT NULL,

created\_at timestamp DEFAULT CURRENT\_TIMESTAMP,

updated\_at timestamp DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

FOREIGN KEY (category\_id) REFERENCES categories(id),

FOREIGN KEY (user\_id) REFERENCES users(id)

) ENGINE=InnoDB;

CREATE TABLE recipe\_ingredient (

id int NOT NULL AUTO\_INCREMENT PRIMARY KEY,

recipe\_id int,

ingredient\_id int,

quantity varchar(255),

created\_at timestamp DEFAULT CURRENT\_TIMESTAMP,

updated\_at timestamp DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

FOREIGN KEY (recipe\_id) REFERENCES recipes(id) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (ingredient\_id) REFERENCES ingredients(id) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB;

CREATE TABLE reviews (

id int NOT NULL AUTO\_INCREMENT PRIMARY KEY,

recipe\_id int,

user\_id int,

rating int,

comment text,

created\_at timestamp DEFAULT CURRENT\_TIMESTAMP,

updated\_at timestamp DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

FOREIGN KEY (recipe\_id) REFERENCES recipes(id) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (user\_id) REFERENCES users(id)

) ENGINE=InnoDB;

-- 1. Додавання категорії

INSERT INTO categories (name) VALUES ('Салати');

-- 2. Додавання інгредієнтів

INSERT INTO ingredients (name) VALUES

('Помідор'),

('Огірок'),

('Сир фета'),

('Оливки');

-- 3. Додавання користувача

INSERT INTO users (name, email, password)

VALUES ('Іван Іваненко', 'ivan@example.com', 'пароль123');

-- 4. Додавання рецепту

INSERT INTO recipes (title, description, instructions, category\_id, user\_id)

VALUES (

'Грецький салат',

'Смачний салат із свіжих овочів та сиру фета.',

'1. Нарізати овочі. 2. Додати сир та оливки. 3. Перемішати.',

1, -- ID категорії 'Салати'

1 -- ID користувача 'Іван Іваненко'

);

-- 5. Додавання інгредієнтів до рецепту

INSERT INTO recipe\_ingredient (recipe\_id, ingredient\_id, quantity)

VALUES

(1, 1, '2 шт.'), -- Грецький салат, Помідор

(1, 2, '1 шт.'), -- Грецький салат, Огірок

(1, 3, '100 г'), -- Грецький салат, Сир фета

(1, 4, '50 г'); -- Грецький салат, Оливки

-- 6. Додавання відгуку

INSERT INTO reviews (recipe\_id, user\_id, rating, comment)

VALUES (1, 1, 5, 'Дуже смачний салат!');

Оновлення

UPDATE recipes

SET

title = 'Оновлений грецький салат',

description = 'Оновлений рецепт смачного салату з овочів, сиру фета та оливок.',

instructions = '1. Нарізати свіжі овочі. 2. Додати більше сиру фета та оливок. 3. Додати спеції та перемішати.',

image = 'assets/images/updated\_greek\_salad.jpg', -- Оновлена картинка

views = views + 1, -- Наприклад, оновлення кількості переглядів

active = 1 -- Встановлюємо активний статус

WHERE id = 13; -- Оновлюємо рецепт з ID 1

Результат:

