PHP-Laravel-Migrations-Industry

**Laravel Assignment**

MODULE – 8

**Submitted To:-**

**Rajesh Nagar Sir**

**Submitted By:-**

**Ankita Sahu**

MODULE – 8(Migration)

**1. How to do config database in laravel ?**

**Ans:-**

1. First, ensure you have a database set up and have the credentials (username, password, hostname, etc.) ready. Laravel supports several database systems, including MySQL, PostgreSQL, and SQLite.
2. Open the config/database.php file and find the connections array. This array contains all of the available connection configurations.
3. Choose a connection type from the array (e.g., mysql, sqlite, pgsql, etc.) and update the corresponding configuration options with your database credentials. For example, to configure a MySQL database, you might update the Default Database Connection Name:

'default' => env('DB\_CONNECTION', 'mysql'),

1. Open the .env file in the root directory of your Laravel project. Set the DB\_CONNECTION variable in this file to match the connection type you chose in the previous step. For example, if you are using a MySQL database, you would set the DB\_CONNECTION variable to mysql.

DB\_CONNECTION=mysql

DB\_HOST=127.0.0.1

DB\_PORT=3306

DB\_DATABASE=laravel

DB\_USERNAME=root

DB\_PASSWORD=

1. Set the DB\_HOST variable to the hostname of your database server. For example, if you are using a local development server, you might set this to localhost.
2. Set the DB\_DATABASE variable to the name of your database.
3. Set the DB\_USERNAME and DB\_PASSWORD variables to the username and password for your database.
4. If you are using a MySQL or PostgreSQL database, you may also need to set the DB\_PORT variable in your .env file to the correct port number for your database server.
5. If you are using a MySQL database, you may also need to set the DB\_SOCKET variable in your .env file if your MySQL server uses a socket other than the default.
6. Save the .env file and exit.

**2. Call MySQLi Store Procedure from Laravel?**

**Ans:-**

<?php

**use** Illuminate\Database\Migrations\Migration;

**use** Illuminate\Database\Schema\Blueprint;

**use** Illuminate\Support\Facades\Schema;

**class** CreatePostProcedure **extends** Migration

{

/\*\*

\* Run the migrations.

\*

\* @return void

\*/

**public** **function** up()

{

$procedure = "DROP PROCEDURE IF EXISTS `get\_posts\_by\_userid`;

CREATE PROCEDURE `get\_posts\_by\_userid` (IN idx int)

BEGIN

SELECT \* FROM posts WHERE user\_id = idx;

END;";

\DB::unprepared($procedure);

}

/\*\*

\* Reverse the migrations.

\*

\* @return void

\*/

**public** **function** down()

{

}

}

Route::**get**('call-procedure', **function** () {

$postId = 1;

$getPost = DB::**select**(

'CALL get\_posts\_by\_userid('.$postId.')'

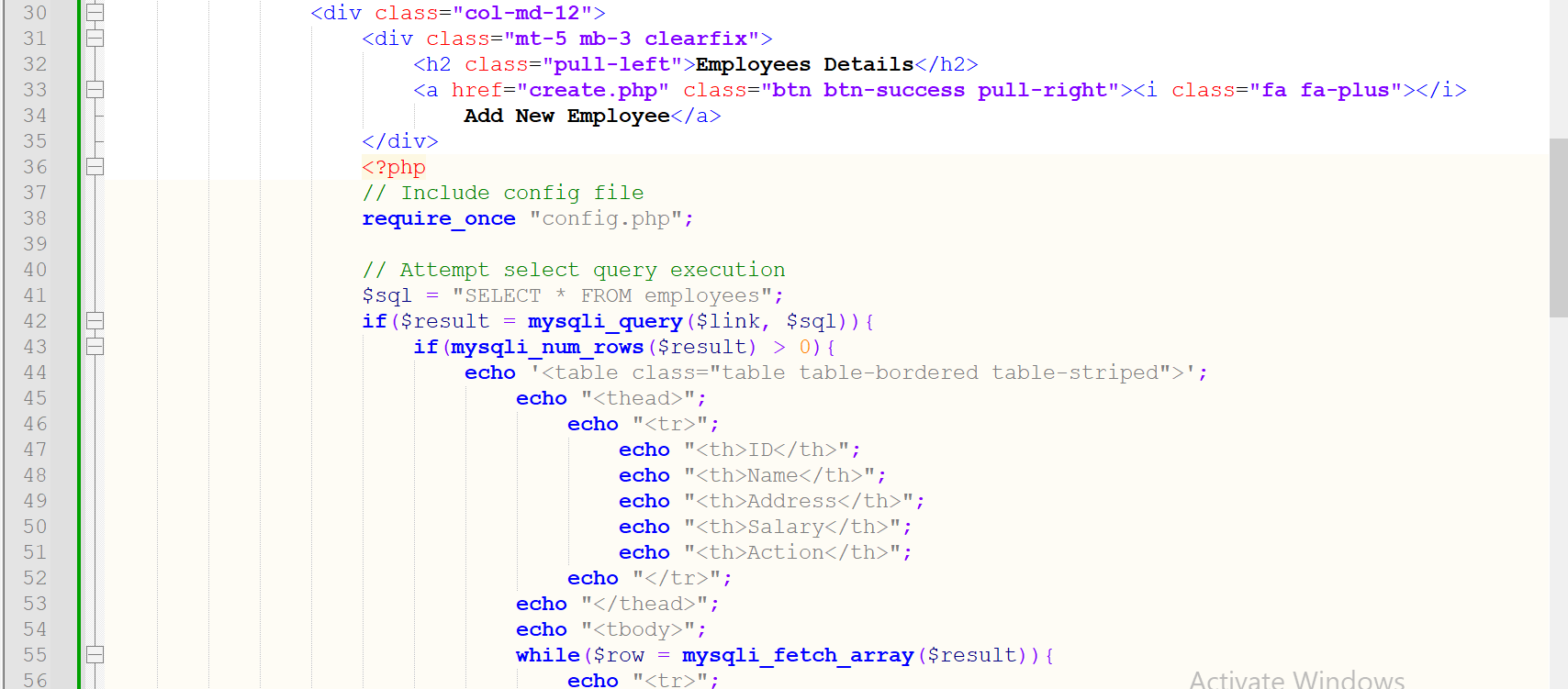
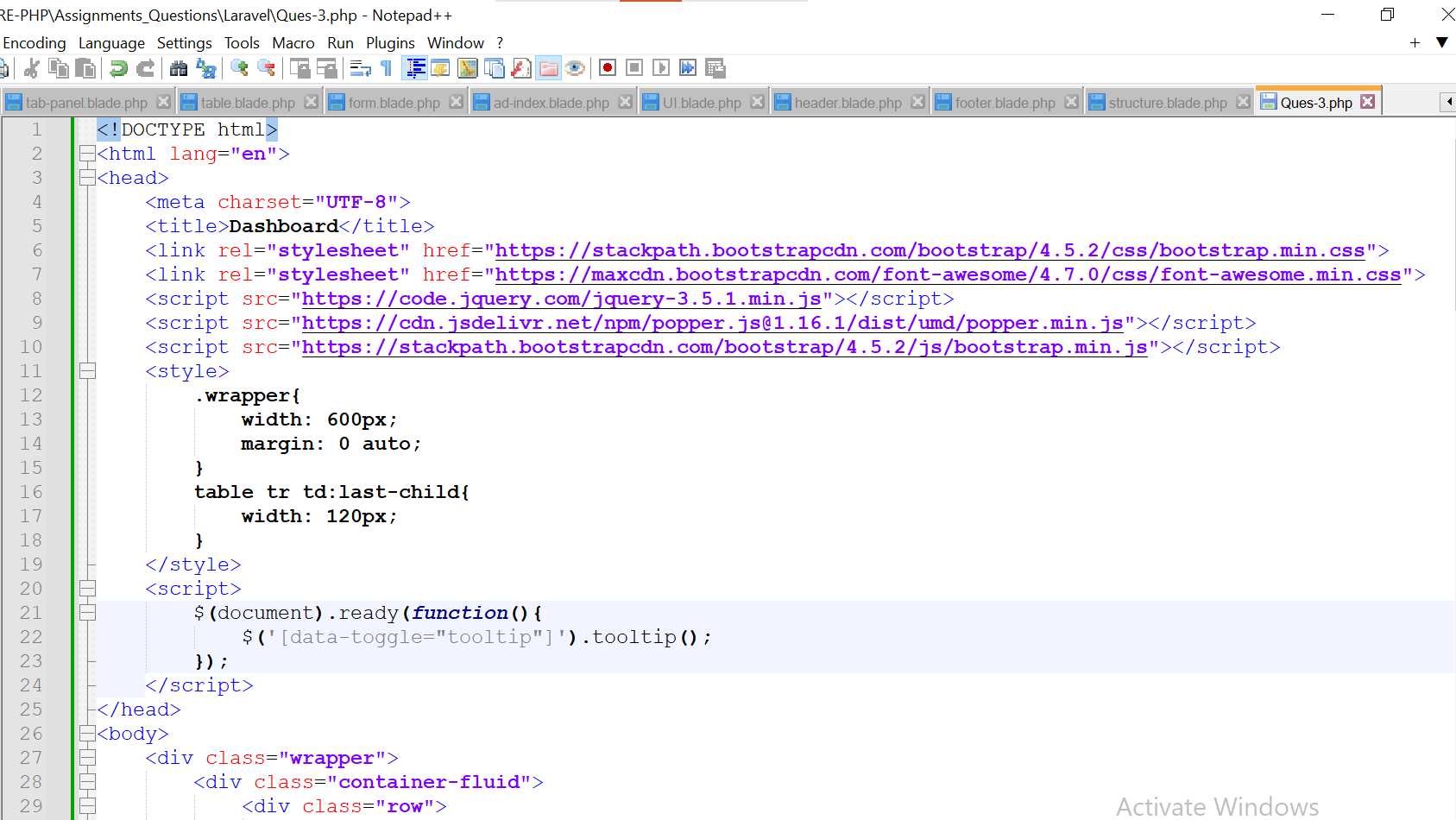
);

dd($getPost);

});

**3. Apply Curd Operation through Query Builder for Employee Management?**

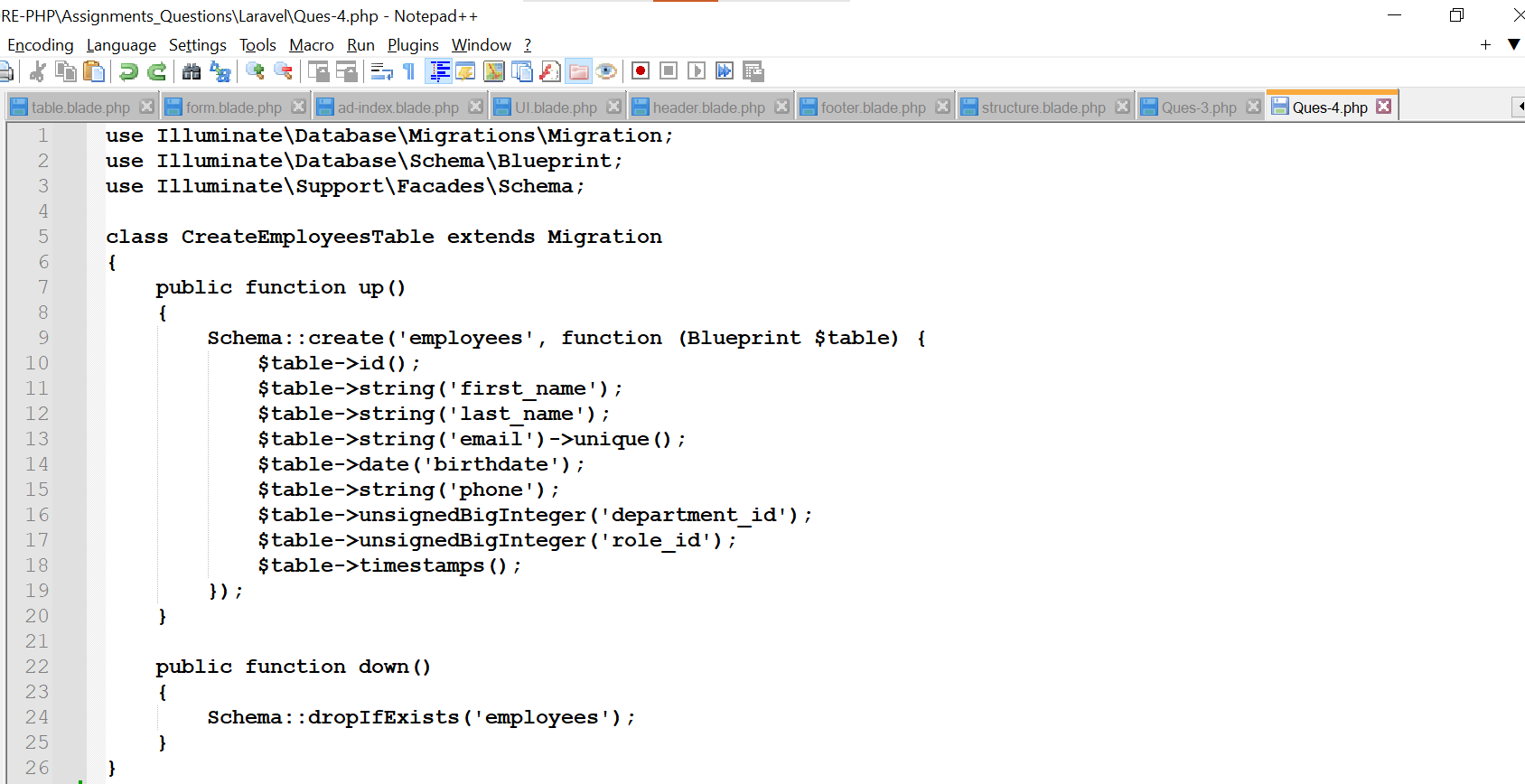
**Ans:-**

****

****

**4. Create All Migration for Employee management?**

**Ans:-**

**xoxoxoxoxoxoxoxo**