Ostad

Ostad PHP Laravel - Assignment on Module 17
Assignment Name: Laravel Query Builder

Student: Shafayat Hossain

Before Started:
1. Create a laravel project name ***"LaravelQuerybuilder"*** with terminal via writing ***"composer create-project laravel/laravel LaravelQuerybuilde"*** command
2. Create a migration table into mysql server using terminal via writing ***"php artisan make:migration create_posts_table"*** command.

Question 1. Explain what Laravel's query builder is and how it provides a simple and elegant way to interact with databases

Laravel's query builder is a powerful feature in the Laravel framework that simplifies database interactions. It provides an intuitive and readable syntax for constructing queries, allowing developers to write database-agnostic code. The query builder automatically handles syntax and functionality differences between various database systems, making it easier to switch between them. It also offers protection against SQL injection attacks by escaping user input and using parameter binding. With convenient methods for common database operations, the query builder improves productivity and reduces the time and effort required for database tasks.

Question 2. Write the code to retrieve the "excerpt" and "description" columns from the "posts" table using Laravel's query builder. Store the result in the $posts variable. Print the $posts variable.

```
$posts = DB::table('posts')
  ->select('excerpt', 'description')
  ->get();
```

Question 3. Describe the purpose of the distinct() method in Laravel's query builder. How is it used in conjunction with the select() method?

The ***distinct()*** method in Laravel's query builder is used to retrieve unique values from a column in a database table.

When used in conjunction with the ***select()*** method, the ***distinct()*** method allows you to specify which column or columns should have distinct values in the query results.

Ostad

Question 4. Write the code to retrieve the first record from the "posts" table where the "id" is 2 using Laravel's query builder. Store the result in the $posts variable. Print the "description" column of the $posts variable.

```
$posts = DB::table('posts')
  ->where('id', '==', '2')
  ->select('description')
  ->first();
```

Question 5. Write the code to retrieve the "description" column from the "posts" table where the "id" is 2 using Laravel's query builder. Store the result in the $posts variable. Print the $posts variable.

```
$posts = DB::table('posts')
  ->where('id', '==', '2')
  ->pluck('description');
```

Question 6. Explain the difference between the first() and find() methods in Laravel's query builder. How are they used to retrieve single records?

In Laravel's query builder, the ***first()*** and ***find()*** methods are used to retrieve single records from a database table. However, they have slight differences in their behavior and usage.

The ***first()*** method is used to retrieve the first record that matches the query criteria. It is commonly used when you want to retrieve the earliest or oldest record based on a specific order

On the other hand, the ***find()*** method is used to retrieve a record by its primary key. It expects the primary key value as its parameter and fetches the corresponding record

Question 7. Write the code to retrieve the "title" column from the "posts" table using Laravel's query builder. Store the result in the $posts variable. Print the $posts variable.

```
$posts = DB::table('posts')
  ->select('title')
  ->get();
```

Ostad

Question 8. Write the code to insert a new record into the "posts" table using Laravel's query builder. Set the "title" and "slug" columns to 'X', and the "excerpt" and "description" columns to 'excerpt' and 'description', respectively. Set the "is_published" column to true and the "min_to_read" column to 2. Print the result of the insert operation.

```
function insertNew(Request $request){
    //Q8
    if($request->validate()){
        $posts = DB::table('posts')
        ->insert([
            'title'=> $request->input('title'),
            'slug'=> $request->input('slug'),
            'excerpt'=> $request->input('excerpt'),
            'description'=> $request->input('description'),
            'is_published'=> $request->input('is_published'),
            'min_to_read'=> $request->input('min_to_read'),
        ]);
        if(!$posts){
            $request->json([
                'success' => false,
                'msg'=> 'Data not created',
            ]);
        }else{
            $request->json([
                'success' => true,
                'msg'=> $posts,
            ], 201);
        }
    }
}
```

Question 9. Write the code to update the "excerpt" and "description" columns of the record with the "id" of 2 in the "posts" table using Laravel's query builder. Set the new values to 'Laravel 10'. Print the number of affected rows.

```
  $updatedRows = DB::table( 'posts' )
        ->where( 'id', 2 )
        ->update( ['excerpt' => 'Laravel 10', 'description' =>
                    'Laravel 10'] );

    echo "Number of affected rows: " . $updatedRows;
```

Ostad

Question 10. Write the code to delete the record with the "id" of 3 from the "posts" table using Laravel's query builder. Print the number of affected rows.

```php
$deletedRows = DB::table( 'posts' )
      ->where( 'id', 3 )
      ->delete();

  echo "Number of deleted rows: " . $deletedRows;
```

Question 11. Explain the purpose and usage of the aggregate methods count(), sum(), avg(), max(), and min() in Laravel's query builder. Provide an example of each.

```php
$posts = DB::table('users')->count();
$posts = DB::table('posts')->sum('min_to_read');
$posts = DB::table('posts')->avg('min_to_read');
$posts = DB::table('posts')->max('min_to_read');
$posts = DB::table('posts')->min('min_to_read');
```

The ***count()*** method returns the number of records that match the query criteria.
The ***sum()*** method calculates the sum of the values in a specific column.
The ***avg()*** method calculates the average value of a specific column.
The ***max()*** method retrieves the maximum value from a specific column.
The ***min()*** method retrieves the minimum value from a specific column.

Question 12. Describe how the whereNot() method is used in Laravel's query builder. Provide an example of its usage.

```php
$posts = DB::table('posts')
  ->whereNot('is_published', '0')
  ->get();
```

The ***whereNot()*** method is used to add a "where not" condition to a query. It allows you to exclude records that do not meet a specified condition from the result set

Ostad

Question 13. Explain the difference between the exists() and doesntExist() methods in Laravel's query builder. How are they used to check the existence of records?

```php
$hasActiveUsers = DB::table( 'users' )
      ->where( 'status', 'active' )
      ->exists();

  if ( $hasActiveUsers ) {
      echo "There are active users.";
  } else {
      echo "No active users found.";
  }
```

```php
$noInactiveUsers = DB::table( 'users' )
      ->where( 'status', 'inactive' )
      ->doesntExist();
  if ( $noInactiveUsers ) {
      echo "There are no inactive users.";
  } else {
      echo "Inactive users found.";
  }
```

The ***exists()*** method returns ***true*** if at least one record matching the query criteria exists; otherwise, it returns ***false***.

The ***doesntExist()*** method is the negation of the ***exists()*** method. It returns ***true*** if no records matching the query criteria exist; otherwise, it returns ***false***.

Question 14. Write the code to retrieve records from the "posts" table where the "min_to_read" column is between 1 and 5 using Laravel's query builder. Store the result in the $posts variable. Print the $posts variable.

```php
$posts = DB::table( 'posts' )
      ->whereBetween( 'min_to_read', [1, 5] )
      ->get();

  print_r( $posts );
```

Ostad

Question 15. Write the code to increment the "min_to_read" column value of the record with the "id" of 3 in the "posts" table by 1 using Laravel's query builder. Print the number of affected rows.

```php
$affectedRows = DB::table( 'posts' )
      ->where( 'id', 3 )
      ->increment( 'min_to_read', 1 );

  echo "Number of affected rows: " . $affectedRows;
```