Assignment: Module-17

Name: Md. Ruhul Kuddus

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

Answer:

Laravel's query builder is an API provided by the Laravel framework that simplifies database operations. It allows developers to interact with databases using a fluent and expressive syntax. With the query builder, you can construct and execute database queries without writing raw SQL statements. It provides methods for selecting, inserting, updating, and deleting records, as well as for filtering, sorting, joining, and aggregating data.

The query builder abstracts the underlying database engine, making your code portable across different database systems. It also offers features like parameter binding to prevent SQL injection, automatic query execution, and result formatting. Overall, Laravel's query builder provides a simple and elegant way to work with databases, improving code readability and developer productivity.

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.

Route:

```
Route::get( '/getExcerptDescription', [AssignmentController::class,
'getExcerptDescription'] );
```

Controller:

```
function getExcerptDescription() {
    $posts = DB::table( 'posts' )->select( 'excerpt', 'description' )->get();
    return $posts;
    }
```

Question-3:

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

Answer:

The distinct() method in Laravel's query builder is used to retrieve only unique rows from a table. It ensures that the query results do not contain any duplicate rows.

When used in conjunction with the select() method, distinct() modifies the select statement to return only distinct (unique) values for the specified columns. It filters out any duplicate rows based on the selected columns.

Example:

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

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.

Route:

```
Route::get( '/firstRecordDescription',
[AssignmentController::class,'firstRecordDescription'] );
```

Controller:

```
function firstRecordDescription() {
        $posts = DB::table( 'posts' )->where( 'id', '=', '2' )->first();
        return $posts->description;
        }
```

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.

Route:

```
Route::get( '/getDescription', [AssignmentController::class, 'getDescription'] );
```

Controller:

```
function getDescription() {
            $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?

Answer:

The first() method is used to retrieve the first record that matches the query conditions. It returns a single object containing the record's data.

Example:

```
$firstPost = DB::table( 'posts' )->first();
```

The find() method is used to retrieve a record based on its primary key value. It assumes that the primary key column in the table is named "id" by default.

Example:

```
$post = DB::table('posts')->find(2);
```

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.

Route:

```
Route::get( '/getAllTitle', [AssignmentController::class, 'getAllTitle'] );
```

Controller:

```
function getAllTitle() {
            $posts = DB::table( 'posts' )->select( 'title' )->get();
            return $posts;
            }
```

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.

Route:

```
Route::post( '/insertPost', [AssignmentController::class, 'insertPost'] );
```

Controller:

```
function insertPost() {
        $result = DB::table( 'posts' )->insert( [
        'title' => 'X',
        'slug' => 'x',
        'excerpt' => 'excerpt',
        'description' => 'description',
        'is_published' => true,
        'min_to_read' => 2,
        ] );
        return $result;
        }
```

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.

Route:

```
Route::patch( '/update', [AssignmentController::class, 'update'] );
```

Controller:

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

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.

Route:

```
Route::delete( '/delete', [AssignmentController::class, 'delete'] );
```

Controller:

```
function delete() {
        $affectedRows = DB::table( 'posts' )->where( 'id', 3 )->delete();
    }
```

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.

count(): The count() method is used to retrieve the total count of records that match a specific condition. It returns the number of rows in the result set.

Example:

```
$count = DB::table( 'posts' )->count();
```

sum(): The sum() method calculates the sum of a specific column's values. It is typically used on numeric columns.

Example:

```
$sum = DB::table( 'orders' )->sum( 'amount' );
```

avg(): The avg() method calculates the average value of a specific column. It is commonly used on numeric columns.

Example:

```
$avg = DB::table( 'orders' )->avg( 'amount' );
```

max(): The max() method retrieves the maximum value from a specific column. It is typically used on numeric or date/time columns.

Example:

```
$max = DB::table( 'orders' )->max( 'amount' );
```

min(): The min() method retrieves the minimum value from a specific column. It is commonly used on numeric or date/time columns.

Example:

```
$min = DB::table( 'orders' )->min( 'amount' );
```

Question-12:

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

Answer:

whereNot(): The whereNot() method is typically used in combination with the where() method to add additional conditions to the query. It accepts two parameters: the column name and the value to compare against.

Example:

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

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?

Answer:

exists(): The exists() method is used to check if any records exist in a table that match a specific condition. It returns true if at least one record is found, and false otherwise.

Example:

```
$hasPublishedPosts = DB::table( 'posts' )->where( 'is_published', true )-
>exists();
```

doesntExist(): The doesntExist() method is the opposite of exists(). It is used to check if no records exist in a table that match a specific condition. It returns true if no records are found, and false otherwise.

Example:

```
$noPublishedPosts = DB::table( 'posts' )->where( 'is_published', false )-
>doesntExist();
```

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.

Route:

```
Route::get( '/getPostByMinRead', [AssignmentController::class,
'getPostByMinRead'] );
```

Controller:

```
  function getPostByMinRead() {
        $posts = DB::table( 'posts' )->whereBetween( 'min_to_read', [1, 5] )-
>get();
        return $posts;
        }
```

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.

Route:

```
:post( '/increments', [AssignmentController::class, 'increments'] );
```

Controller:

```
function increments() {
        $posts = DB::table( 'posts' )->where( 'id', 3 )->increment(
'min_to_read', 1);
        return $posts;
        }
```