মডিউল ১৭ এর এসাইনমেন্ট

Assignment: Query Builder in Laravel

Instructions:

Read the given code snippets and questions carefully.

Write the code that best answers each question.

Make sure to provide clear and concise answers.

Submit your completed assignment within the given time frame.

Questions:

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 feature of the Laravel framework that provides a convenient and expressive way to interact with databases. It allows developers to build and execute database queries using a fluent, chainable interface, rather than writing raw SQL statements.

The query builder provides a set of methods that represent different parts of a database query, such as selecting columns, specifying conditions, ordering results, and joining tables. By chaining these methods together, developers can easily construct complex queries in a readable and maintainable manner.

One of the key advantages of Laravel's query builder is its simplicity. It abstracts away the complexities of writing raw SQL statements and provides a more intuitive and developer-friendly API. This makes it easier to write and understand database queries, especially for developers who are not well-versed in SQL.

The query builder also promotes code reusability and modularity. Developers can create query builder instances and reuse them across different parts of their application, reducing code duplication. Additionally, the fluent interface allows for method chaining, enabling developers to progressively build queries without repeating code.

Laravel's query builder also incorporates features like parameter binding, which helps prevent SQL injection attacks by automatically escaping user input. It also provides convenient methods for pagination, aggregations, and other common database operations.

Another advantage of the query builder is its database agnosticism. It supports multiple database systems, including MySQL, PostgreSQL, SQLite, and SQL Server. Developers can write queries using the query builder syntax and Laravel will generate the appropriate SQL statements for the specific database being used.

In summary, Laravel's query builder simplifies and enhances the process of interacting with databases. It offers a clean and expressive syntax, promotes code reusability, provides security features, and supports multiple database systems. These features make it a powerful tool for developers working with databases in Laravel applications.

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();

return $posts;
```

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 only unique values from a column or set of columns in a database table. It ensures that duplicate records are eliminated, and only distinct values are returned in the result set.

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

return $posts;
```

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)->first();

return $posts;
```

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');

return $posts;
```

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

first() is used to retrieve the first record based on the query criteria, whereas find() is used to retrieve a record by its primary key. The first() method is more flexible and can be used with various query conditions, while find() is specifically designed for retrieving records by their IDs.

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

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

return $posts;
```

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')->pluck('title');

return $posts;
```

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.

```
DB::table('posts')->insert([
    'user_id' => 1,
    'title' => 'X',
    'slug' =>  'X',
    'description' => 'description',
    'excerpt' => 'excerpt',
    'is_published' => true,
    'min_to_read' => 2,
]);
```

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.

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

]);
```

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.

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

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.

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

$totalTime = DB::table('posts')->sum('min_to_read');

$minTime = DB::table('posts')->min('min_to_read');

$maxTime = DB::table('posts')->max('min_to_read');

$avgTime = DB::table('posts')->avg('min_to_read');

return [$postsNumbers, $totalTime, $minTime, $maxTime, $avgTime];
```

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

```
$postsN = DB::table('posts')->whereNot('id', '=', 51)->sum('min_to_read');

return $postsN;
```

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?

The exists() method is used to check if there are any records in the table that match the query criteria.

It returns a boolean value (true or false) indicating whether any matching records exist.

```
$postExists = DB::table('posts')->where('id', '=', 51)->exists();
if ($postExists) {
   echo 'this post is exists';
} else {
   echo 'this post doesnot exists';
}
```

The doesntExist() method is the opposite of exists(). It is used to check if there are no records in the table that match the query criteria.

It returns a boolean value (true or false) indicating whether there are no matching records.

```
$postNotExists = DB::table('posts')->where('id', '=', 151)->doesntExist();

if ($postNotExists) {

    echo 'this post doesnot exists';

} else {

    echo 'this post  exists';

}
```

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.

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

return $postsBetween;
```

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.

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

return $postIncrement;
```