Md Nahid Hossen

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 a database abstraction layer that provides a convenient and intuitive way to interact with databases in Laravel applications. It allows developers to construct database queries using a fluent and expressive API, without the need for writing raw SQL statements.

By using the query builder, developers can build complex queries easily by chaining methods together. The query builder provides methods for various database operations such as selecting, inserting, updating, and deleting records. It also supports advanced features like joins, subqueries, conditional clauses, ordering, and grouping.

One of the key benefits of Laravel's query builder is its simplicity and elegance. It abstracts away the complexities of writing SQL queries, making the code more readable and maintainable. Developers can express their intent in a more natural and intuitive way using method calls and expressive syntax, resulting in cleaner and more understandable code.

Additionally, the query builder handles parameter binding automatically, which helps prevent SQL injection attacks by properly escaping user input. It also supports multiple database systems, allowing developers to write database-agnostic code that can seamlessly work with different database backends.

Overall, Laravel's query builder simplifies the process of interacting with databases by providing a clean and concise API. It enhances developer productivity, promotes code readability, and ensures secure and reliable database operations 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.

Answer:

```
///php

$posts = DB::table('posts')

        ->select('excerpt', 'description')

        ->get();

print_r($posts);
```

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 distinct or unique values from a column in a database table. It eliminates duplicate rows from the result set based on the specified column(s).

When used in conjunction with the select() method, the distinct() method allows us to specify that we want to retrieve unique values for the selected columns.

Here's an example to illustrate its usage:

```
$distinctTitles = DB::table('posts')

        ->select('title')

        ->distinct()

        ->get();
```

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.

Answer:

Here's 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, and print the "description" column of the $posts variable:

```
///php


$posts = DB::table('posts')

`       ->where('id', 2)

        ->first();


if ($posts) {
    echo $posts
            ->description;
} else {
    echo "No record found.";
}
```

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.

Answer:

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

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

Answer:

first(): The first() method is used to retrieve the first record that matches the given query conditions. It returns a single instance of the model or a plain PHP object if not using Eloquent. The first() method is typically used when you want to retrieve the first record based on specific conditions or when you want to retrieve a single record without specifying the primary key.

```
$post = DB::table('posts')
          ->where('category', 'News')
          ->first();
```

In this example, the first() method is used to retrieve the first record from the "posts" table where the "category" column is equal to 'News'.

find(): The find() method is used to retrieve a record by its primary key. It returns a single instance of the model or a plain PHP object if not using Eloquent. The find() method is commonly used when you know the primary key value of the record you want to retrieve.

```
///php
$post = DB::table('posts')
          ->find(1);
```

In this example, the find() method is used to retrieve the record from the "posts" table with a primary key value of 1.

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.

Answer:

```
///php

$posts = DB::table('posts')

            ->pluck('title');

print_r($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.

Answer:

```
///php

$data = [

    'title' => 'X',

    'slug' => 'X',

    'excerpt' => 'excerpt',

    'description' => 'description',
    'is_published' => true,

    'min_to_read' => 2

];


$result = DB::table('posts')->insert($data);


if ($result) {

    echo "Record inserted successfully.";

} else {

    echo "Failed to insert the record.";

}
```

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.

Answer:

```
///php

$posts = DB::table('posts')

 ->where('id', 2)

 ->update([

 'excerpt' => 'Laravel 10',
```

'description' => 'Laravel 10 is the latest version of Laravel, the popular PHP framework. It includes a

number of new features and improvements, including a new Blade compiler, a new routing system, and a

new database abstraction layer.',

]);

$affectedRows = $posts->rowCount();

echo "Number of affected rows: $affectedRows";

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.

Answer:

```
///php

DB::table('posts')

 ->where('id', 3)

 ->delete();

$affectedRows = DB::affectedRows();

echo $affectedRows;
```

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.

Answer:

````count()` - This method returns the number of records that match the query conditions.

///php

$totalUsers = DB::table('users')->count();

‘’’

- `sum()` - This method calculates the sum of the values in a specified column.

///php

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

‘’’’

- `avg()` - This method calculates the average of the values in a specified column.

///php

$averageRating = DB::table('reviews')->avg('rating');

‘’’’

- `max()` - This method retrieves the maximum value from a specified column.

///php

$highestPrice = DB::table('products')->max('price');

‘’’’

- `min()` - This method retrieves the minimum value from a specified column.
````

```
///php

$lowestAge = DB::table('users')->min('age');
```

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

Answer:

The whereNot() method in Laravel's query builder is used to add a "not equal" condition to the query. It allows you to retrieve records where a specific column's value does not match the provided value. Here's an example to illustrate its usage:

```
$users = DB::table('users')
        ->whereNot('status', 'inactive')
        ->get();
```

In this example, we have a table called "users" and we want to retrieve all the users whose "status" column is not equal to 'inactive'.

The whereNot() method takes two arguments: the column name and the value to compare against. In this case, we are comparing the "status" column against the value 'inactive'. The method will generate a SQL query that retrieves all the records where the "status" column is not equal to 'inactive'.

The get() method is used to execute the query and retrieve the results. In this example, the $users variable will contain a collection of all the users whose "status" is not 'inactive'.

The whereNot() method can be used in combination with other query builder methods to further refine the query conditions and retrieve the desired results.

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()` - This method checks if there are any records that match the query conditions and returns a

boolean value indicating whether the records exist or not.

```
/////php

if (DB::table('users')->where('name', 'John')->exists()) {

 // Records with name 'John' exist

} else {

 // No records with name 'John' exist
```

}

‘’’

- `doesntExist()` - This method checks if there are no records that match the query conditions and

returns a boolean value indicating whether the records don't exist.

///php

```
if (DB::table('users')->where('name', 'John')->doesntExist()) {
 // No records with name 'John' exist
} else {
 // Records with name 'John' exist
}
```

‘’’’

By using exists() and doesntExist() methods, i can easily validate the existence of records in my database

tables and make decisions based on the results.

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.

Answer:

///php

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

foreach ($posts as $post) {

 echo $post->title . ' ' . $post->min_to_read . PHP_EOL;

}
```

This code will first create a new `Post` object and then use the `whereBetween` method to filter the

results to only include posts where the `min_to_read` column is between 1 and 5. The results will then

be stored in the `$posts` variable. Finally, the `$posts` variable will be looped through and each post's

title and `min_to_read` value will be printed.

///php

```
Title Min To Read

------- --------

Post 1 1

Post 2 2

Post 3 3

Post 4 4

Post 5 5
```

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.

Answer:

```
///php

DB::table('posts')

 ->where('id', 3)

 ->increment('min_to_read', 1);

$affectedRows = DB::table('posts')

 ->where('id', 3)

 ->count();

echo $affectedRows . ' rows affected';
```

This code will first increment the `min_to_read` column value of the record with the `id` of 3 by 1. Then,

it will count the number of rows that were affected by the update. Finally, it will print the number of affected rows