Assignment For Module-17: Query Builder

Questions & Answers

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 that allows you to interact with databases in a simple and elegant way. It provides a fluent syntax for constructing and executing database queries, making it easier to work with databases in Laravel. The query builder's expressive methods and seamless integration with other Laravel features simplify database interactions and promote efficient and maintainable code.

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 specific column or a combination of columns in the result set of a query. It eliminates duplicate rows from the query result.

When used in conjunction with the select() method, distinct() modifies the behavior of the select() method by applying the "DISTINCT" keyword to the generated SQL query.

```
$uniqueNames = DB::table('users')->select('name')->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.

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

```
}
```

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.

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

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

first() method: The first() method retrieves the first record that matches the query conditions. It returns an instance of the object representing the record or null if no matching record is found.

```
$user = DB::table('users')->where('email', 'example@example.com')->first();
```

find() method: The find() method retrieves a record by its primary key. It expects the primary key value as an argument and returns an instance of the object representing the record or null if no matching record is found.

```
$user = DB::table('users')->find(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.

```
$posts = DB::table('posts')->pluck('title');
print_r($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.

```
$result = DB::table('posts')->insert([
    'title' => 'X',
    'slug' => 'X',
    'excerpt' => 'excerpt',
    'description' => 'description',
    'is_published' => true,
```

```
    'min_to_read' => 2
]);

echo $result;
```

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.

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

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.

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

echo "Number of affected rows: " . $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.

1. count() - It returns the number of records that match the query conditions.
   `$count = DB::table('users')->where('status', 'active')->count();`
2. sum() - It calculates the sum of a specific column's values.
   `$totalAmount = DB::table('orders')->sum('amount');`
3. avg() - It calculates the average (mean) of a specific column's values.
   `$averageRating = DB::table('reviews')->where('product_id', 1)->avg('rating');`
4. max() - It retrieves the maximum value of a specific column.
   `$maxPrice = DB::table('products')->max('price');`
5. min() - It retrieves the minimum value of a specific column.
   `$minStock = DB::table('products')->min('stock');`

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

The whereNot() method in Laravel's query builder is used to add a "not equal" condition to a query. It allows you to retrieve records that do not match a specific value or column.

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

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?

exists() method: The exists() method checks if any records exist that match the specified query conditions. It returns true if at least one record is found, and false otherwise.

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

doesntExist() method: The doesntExist() method checks if no records exist that match the specified query conditions. It returns true if no records are found, and false if at least one record is found.

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

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.

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

print_r($posts);
```

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.

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

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