Assignment

Syed Sabbir Hasan

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

Ans: Laravel's query builder is a feature of the Laravel framework that provides a simplified and elegant way to interact with databases. It offers a fluent and expressive API, allowing developers to construct database queries using a readable syntax. The query builder is database-agnostic, supporting multiple database systems. It automatically handles parameter binding to prevent SQL injection attacks. The query builder provides methods for various types of queries, including selection, insertion, update, and deletion, along with advanced features like joins and subqueries. It integrates seamlessly with Eloquent, Laravel's ORM system, providing an additional layer of abstraction. Overall, the query builder simplifies database interactions, making them more manageable and less error-prone.

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.

Ans: The code to retrieve the "excerpt" and "description" columns from the "posts" table using Laravel's query builder are given bellow:

```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?

Ans: The distinct () method in Laravel's query builder is used to retrieve only unique or distinct values from a specific column or set of columns in a database table. It eliminates duplicate rows from the result set, ensuring that each row returned is unique based on the specified column(s).

When used in conjunction with the select () method, the distinct () method narrows down the uniqueness constraint to the columns specified in the select () method. It allows us to retrieve distinct values based on specific columns while still selecting other columns as well.

Example:

```php
$uniqueNames = DB::table('users')
    ->select('name')
    ->distinct()
    ->get();
```

In this code, we query the "users" table and specify that we want to select the "name" column. By chaining the distinct () method after the select () method, we indicate that we only want to retrieve distinct names from the "users" table. The resulting $uniqueNames variable will contain a collection of unique names.

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.

Ans:

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

echo $posts->description;
```

Here, we use the DB::table() method to specify the "posts" table. Then, we use the where () method to add a condition that checks for the "id" column to be equal to 2. The first () method is used to retrieve the first record that matches the condition.

The resulting record is stored in the $posts variable. To print the "description" column of the $posts variable, we use 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.

Ans:

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

print_r($posts);
```

Here, the pluck () method is used to retrieve the value of the "description" column from the matching record.

The resulting "description" value is stored in the $posts variable. To print the $posts variable, we use print_r($posts).

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

Ans: In Laravel's query builder, the first () and find () methods are both used to retrieve single records from a database table, but they have some differences in their functionality and usage.

❖ **first () Method:**

- The first () method is used to retrieve the first record that matches the query conditions.
- It is commonly used when you want to retrieve a single record based on specific conditions, such as fetching the oldest record, the newest record, or the record with the lowest or highest value in a column.
- It returns an instance of the query builder's result object or null if no matching record is found.
- The first () method does not require the primary key to be specified explicitly in the query. It retrieves the first record based on the ordering of the records in the table or the conditions provided in the query.

Example:

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

❖ **find () Method:**

- The find () method is specifically used to retrieve a record by its primary key value.
- It expects the primary key value as an argument and retrieves the record with the matching primary key value.
- It is commonly used when you know the primary key value and want to retrieve the corresponding record directly.
- It returns an instance of the query builder's result object or null if no matching record is found.

Example:

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

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.

Ans:

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

print_r($posts);
```

Here, we use the DB::table () method to specify the "posts" table. Then, we use the select () method to specify the "title" column that we want to retrieve. The get() method is used to execute the query and retrieve the results.

The resulting records containing only the "title" column will be stored in the $posts variable. To print the contents of the $posts variable, we use print_r($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.

Ans:

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

print_r($result);
```

Here, we use the DB::table () method to specify the "posts" table. The insert() method is then used to insert a new record into the table. We pass an associative array where the keys represent the column names, and the values represent the corresponding data to be inserted.

The code sets the "title" and "slug" columns to 'X', the "excerpt" column to 'excerpt', the "description" column to 'description', the "is_published" column to true, and the "min_to_read" column to 2.

The insert () method returns a boolean value indicating the success of the insert operation. To print the result of the insert operation, we use print_r ($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.

Ans:

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

```
echo $affectedRows;
```

Here, we use the DB::table () method to specify the "posts" table. The where () method is then used to add a condition to match the record with the "id" of 2.

The update () method is used to update the specified columns with new values. We pass an associative array where the keys represent the column names to be updated, and the values represent the new values.

The update () method returns the number of affected rows, which represents the number of records that were successfully updated. We store this value in the **$affectedRows** variable and then print it using echo **$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.

Ans:

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

echo $affectedRows;
```

Here, we use the DB::table () method to specify the "posts" table. The where() method is then used to add a condition to match the record with the "id" of 3. The delete () method is used to delete the matching record from the table. It returns the number of affected rows, which represents the number of records that were successfully deleted. We store this value in the **$affectedRows** variable. Finally, we print the number of affected rows using echo **$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.

Ans:

♦ count():

- The count() method is used to retrieve the number of rows that match the query conditions.
- It returns the count as an integer value.

Example:

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

♦ sum():

- The sum() method is used to calculate the sum of a specific column's values that match the query conditions.
- It returns the sum as a numeric value.

Example:

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

♦ avg():

- The avg() method is used to calculate the average (mean) value of a specific column's values that match the query conditions.
- It returns the average as a numeric value.

Example:

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

♦ max():

- The max() method is used to retrieve the maximum value of a specific column that match the query conditions.
- It returns the maximum value as a numeric or string value, depending on the column type.

Example:

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

♦ **min():**

- The min() method is used to retrieve the minimum value of a specific column that match the query conditions.
- It returns the minimum value as a numeric or string value, depending on the column type.

Example:

```
$lowestStock = DB::table('products')->min('stock');
```

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

Ans: In Laravel's query builder, the whereNot() method is used to add a "not equal to" condition to a query. It allows us to retrieve records that do not match a specific value or set of values in a column. The whereNot() method takes two arguments: the column name and the value(s) to compare against. It generates a WHERE clause that excludes records where the specified column is equal to the given value(s).

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

Here, we query the "users" table and use the whereNot () method to specify the condition. The condition states that we want to retrieve users whose "status" column is not equal to 'active'. This will exclude users with a status of 'active' from the result.

```
$users = DB::table('users')
    ->whereNot('role', ['admin', 'superadmin'])
    ->get();
```

Here, whereNot() method is used to exclude users with a "role" of 'admin' or 'superadmin' from the result.

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?

Ans:

exists()	**doesntExist()**
1. The exists() method is used to check if any records exist in the table that match the specified query conditions.	1. The doesntExist() method is the opposite of exists(). It checks if no records exist in the table that match the specified query conditions.
2. It returns true if at least one record is found, and false otherwise.	2. It returns true if no records are found, and false if there is at least one matching record.
3. `$exists = DB::table('users')->where('status', 'active')->exists();`	3. `$doesntExist = DB::table('users')->where('status', 'deleted')->doesntExist();`

In both cases, we use the where () method to specify the query conditions. These conditions can include comparisons, such as equality (=), inequality (<>), or any other applicable operators.

The exists () method is useful when we want to determine if there are records that match specific conditions, while the doesntExist() method is helpful to check if no records match the specified conditions.

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.

Ans:

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

print_r($posts);
```

Here, we use the DB::table() method to specify the "posts" table. The whereBetween() method is then used to add a condition to retrieve records where the "min_to_read" column falls between 1 and 5 (inclusive). The whereBetween() method takes two arguments: the column name and an array of two values representing the lower and upper bounds of the range. The get() method is used to execute the query and retrieve the results. The resulting records that satisfy the condition will be stored in the $posts variable. Finally, we print the contents of the $posts variable using print_r($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.

Ans:

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

echo $affectedRows;
```

Here, we use the DB::table() method to specify the "posts" table. The where() method is then used to add a condition to match the record with the "id" of 3. The increment() method is used to increment the value of the "min_to_read" column by 1 for the matching record. It automatically updates the column value and returns the number of affected rows. We store the number of affected rows in the $affectedRows variable and then print it using echo $affectedRows.