Query Builder in Laravel

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 powerful feature of the Laravel PHP framework that provides a convenient way to build and execute database queries. It can be used to perform most database operations in an application and works perfectly with all Laravel's supported database systems.

Laravel’s query builder provides a simple and elegant way to interact with databases through several key features and design choices.

- ✓ Laravel query builder offers a fluent API for interacting with databases.
- ✓ It supports multiple database systems, making it adaptable to different environments.
- ✓ The syntax is simplified, making it easy to read and write complex queries.
- ✓ Parameter binding is automated to prevent SQL injection attacks.
- ✓ It seamlessly integrates with Laravel's Eloquent ORM for enhanced database interactions.

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.

```php
class PostController extends Controller
{
    function postAction(){
        $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?

=> distinct() method used to retrieve only unique values from single or multiple columns in the database table.

distinct() method can be used in conjunction with the select() method. Ex.

```php
class PostController extends Controller
{
    function postAction(){
        $posts = DB::table('posts')->select('title')->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.

```
class PostController extends Controller
{
    function postAction(){
        $posts = DB::table('posts')->where('id', 2)->first();
        return $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.

```
class PostController extends Controller
{
    function postAction(){
        $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() method: Using first() method we can retrieve first record of a data table based on conditions. It dose not require specifying the primary key.

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

find() method: Using find() method we can retrieve any record of a data table based on primary key. It's require the primary key value to be passed as an argument.

```
      $posts = 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.

```
class PostController extends Controller
{
    function postAction(){
        $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.

```
class PostController extends Controller
{
    function postAction(){
        $posts = DB::table('posts')
            ->insert([
                'title' => 'X',
                'slug' => 'X',
                'excerpt' => 'excerpt',
                'description' => 'description',
                'is_published' => true,
                'min_to_read' => 2
            ]);
        return $posts;
    }
}
```

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.

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

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.

```
class PostController extends Controller
{
    function postAction(){
        $deleteRow = DB::table('posts')
            ->where('id', '=', 3)
            ->delete();
        return $deleteRow;
    }
}
```

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.

=> Explaining the purpose and usage of the aggregate methods count(), sum(), avg(), max(), and min() in Laravel's query builder.

count(): This method calculate the number of records in a table.

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

sum(): It's calculate of a specific column's value.

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

avg(): This method used to calculate average value of a specific column.

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

max(): To retrieves the maximum value from a specific column.

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

min(): it's retrieves the minimum value from a specific column.

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

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's retrieving records where a column's value doesn't match a given value or array.

```
class PostController extends Controller
{
    function postAction(){
        $result = DB::table('posts')->whereNot('like', '=', 20)->get();
        return $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?

=> <u>exists() method:</u> The exists() method is used to check if any records exist in the table that satisfy the given condition. It returns true if at least one record matches the condition and false otherwise.

<u>doesntExist() method:</u> The doesntExist() method is the opposite of exists(). It checks if no records exist in the table that satisfy the given condition. It returns true if no record matches the condition and false if there is at least one matching record.

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.

```
class PostController extends Controller
{
    function postAction(){
        $result = DB::table('posts')->whereBetween('min_to_read', [1, 5])->get();
        return $result;
    }
}
```

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.

```
class PostController extends Controller
{
    function postAction(){
        $result = DB::table('posts')
            ->where('id', '=', 3)
            ->increment('min_to_read');
        return $result;
    }
}
```