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 component of the Laravel framework that enables developers to construct and execute database queries using a fluent and expressive API. It provides a simple and elegant way to interact with databases by offering a chainable interface that closely resembles SQL syntax. The query builder abstracts away the complexities of writing raw SQL queries, handles parameter binding to prevent SQL injection, and supports multiple database systems. With its intuitive syntax and seamless integration with Laravel's Eloquent ORM, the query builder offers a streamlined approach for interacting with databases, making database operations more readable, flexible, and secure.

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
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;

class PostController extends Controller
{
    public function posts(Request $request)
    {
        $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 only distinct (unique) values from a column or a set of columns in a database query result. It eliminates duplicate records, giving you a unique set of values.

When used in conjunction with the select() method, distinct() ensures that only unique values are returned for the selected columns. It helps to filter out duplicate values and provide a clean and concise result set.

To use distinct() with select(), you chain the distinct() method after the select() method in your query.

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 :

```
public function posts(Request $request)
    {
        $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.

Answer :

```
public function posts(Request $request)
    {
        $posts = DB::table('posts')
            ->where('id', 2)
            ->select('description')
```

```
            ->get();

        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?

Answer:
The first() and find() methods in Laravel's query builder are both used to retrieve single records from a table, but there is a difference in how they are used and the scenarios they are commonly used for:

first() method:

The first() method is used to retrieve the first record that matches the query criteria. It returns a single instance of the model or a plain PHP object if no model is associated with the query.
The first() method is typically used when you want to retrieve the first record that satisfies certain conditions, such as the first record in a table or the first record that meets specific criteria.

find() method:

The find() method is used to retrieve a record by its primary key value.
It returns a single instance of the model or a plain PHP object if no model is associated with the query.
The find() method is commonly used when you want to retrieve a specific record based on its primary key value.

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.

```
public function posts(Request $request)
  {
    $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.

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

     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.

```php
    public function posts(Request $request)
    {
        $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.

count() : The count() method is used to retrieve the number of records that match a specific query. 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 in a table. It returns the sum as a numeric value.

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

avg() : The avg() method is used to calculate the average value of a specific column in a table. It returns the average as a numeric value.
Example:

```
$averagePrice = DB::table('products')->avg('price');
```

max() : The max() method is used to retrieve the maximum value from a specific column in a table. It returns the maximum value as a numeric or string value, depending on the column type.
Example:

```
$highestScore = DB::table('students')->max('score');
```

min() : The min() method is used to retrieve the minimum value from a specific column in a table. It returns the minimum value as a numeric or string value, depending on the column type.
Example:

```
$lowestTemperature = DB::table('weather')->min('temperature');
```

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

In Laravel's query builder, the whereNot() method is used to add a "not equal" condition to a query. It allows you to retrieve records where a specific column's value is not equal to the provided value. This method is useful when you want to exclude certain records from the query results based on a specific condition.

Here's an example usage of the whereNot() method:

```
$users = DB::table('users')
        ->whereNot('status', 'inactive')
        ->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 is used to check if any records exist in the result set of a query.
It returns true if at least one record exists, and false otherwise.

doesntExist() method:

The doesntExist() method is used to check if no records exist in the result set of a query.
It returns true if no records exist, and false if at least one record 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.

Answer :

```
$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.

Answer :

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

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